CLOUDSWXTCH

Version 2.3.2

Table of Contents

Getting Started

Getting Started 6
What is cloudSwXtch? 9
What is an xNIC? 12

cloudSwXtch Liscensable Features

cloudSwXtch Liscensable Features 14
Multicast 17
Source Specific Multicast 18
Broadcast 19
High Availability 20
Mesh 23
Bridge 25
Precision Time Protocol 27
Protocol Conversion and Fanout 29

Installing cloudSwXtch

cloudSwXtch System Requirements 32

cloudSwXtch on AWS

cloudSwXtch on AWS 36
Validate Subnets on AWS 37
Verify Security Groups 41
Create SSH Key Pair 44
Install cloudSwXtch on AWS 47
Deploy cloudSwXtch with Terraform on AWS 58
Check Health of cloudSwXtch Instance on AWS 62
Delete cloudSwXtch on AWS 64

cloudSwXtch on Azure

cloudSwXtch on Azure 65
Validate Subnets on Azure 69
Create an Azure cloudSwXtch Template 73
Install cloudSwXtch on Azure 77
Install cloudSwXtch via Azure Marketplace 84
Deploy cloudSwXtch with Terraform on Azure 96
Install cloudSwXtch for an Air-Gapped Environment 100

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

cloudSwXtch on GCP

cloudSwXtch on GCP 112
Install cloudSwXtch via GCP Marketplace 113

cloudSwXtch on OCI

cloudSwXtch on OCI 123
Install cloudSwXtch via OCI Marketplace 124
Cloud agnostic cloudSwXtch VM Install 136

Upgrading cloudSwXtch

Upgrading cloudSwXtch 138

Installing cloudSwXtch Bridge

Installing cloudSwXtch Bridge 140
Install cloudSwXtch Bridge Type 2 141
Install cloudSwXtch Bridge Type 1 146

Installing xNIC

Installing xNIC 148
xNIC System Requirements 149

Install xNIC on Linux

Install xNIC on Linux 151
xNIC Linux Uninstall 157
xNIC Linux Upgrade 158

Install xNIC on Windows

Install xNIC on Windows 160
Uninstall xNIC on Windows 167
Upgrade xNIC on Windows 168

Install xNIC on Kubernetes

Install xNIC on Kubernetes 170
Install xNIC on K8s Cluster 172
Uninstall xNIC on K8s 185
Test xNIC with K8s 186
Upgrade xNIC nodes on K8s 195

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Using wXcked Eye for cloudSwXtch

Using wXcked Eye for cloudSwXtch 196

Monitor cloudSwXtch with wXcked Eye

Monitor cloudSwXtch with wXcked Eye 199
wXcked Eye Cluster Page 202
wXcked Eye cloudSwXtch Page 205
wXcked Eye xNICs Page 209
wXcked Eye Topology Graph 213
wXcked Eye Topology Table 221
wXcked Eye Protocol Fanout Stats 224
wXcked Eye Timing Nodes 227
wXcked Eye Support Page 230

Configure cloudSwXtch with wXcked Eye

Configure cloudSwXtch with wXcked Eye 232
General 234
Mesh with wXcked Eye 236
High Availability with wXcked Eye 242
Protocol Conversion and Fanout with wXcked Eye 247
Protocol Conversion and Fanout Example 255
Channels 259
Stream and Nodes 264

Configuring cloudSwXtch

High Availability 270
Mesh 288
Bridge Type 2 294
Bridge Type 1 300
Protocol Conversion and Fanout 302

Monitoring cloudSwXtch

Azure Monitoring 303
Prometheus Monitoring 307

Testing cloudSwXtch

Testing cloudSwXtch 327
swxtch-perf 328
swxtch-top 333

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

swxtch-tcpdump 351
swxtch-where 353
Universal Third-Party Testing Tools 357
Troubleshooting 370

cloudSwXtch How To's

How to View cloudSwXtch Logs for Troubleshooting 371
How to Install xNIC Dependencies in an Air-Gapped Environment 378
How to View cloudSwXtch Bridge Logs 385
How to Find xNIC Logs 389
How to License a cloudSwXtch 396
How to Install a cloudSwXtch Licensing Server 397
How to set MTU size 401
How to Peer between VPCs in Different Regions for AWS 407

Media Use Cases

Media Use Cases 413
Media Multicast made easy with cloudswXtch 414
Hitless Merge - 2022-7 416
Media support for Compressed and Uncompressed Workflows 417
Protocol Fanout 419
Disaster Recovery 420

cloudSwXtch API

Monitoring API 422
Configuration API 472

cloudSwXtch

Release Notes 492

Resources

Resources 503
Quotas 504

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Getting Started

Quick Start Guide (for those in a hurry)

Introduction

swXtch.io implements a software-based network switch called cloudSwXtch. cloudSwXtch consists of a software network switch and virtual NIC service called xNIC . Together, these components create an overlay network on top of a standard cloud network. This overlay network adds many valuable network features, one of which is a seamless IP multicast experience. With cloudSwXtch, existing user applications and services, that expect standards-based IP multicast, will work on any cloud without requiring any code changes. This enables performance to approach that of bare metal.

Installing cloudSwXtch and xNIC

WHAT TO EXPECT

- In this section, users will be able to learn more about installing cloudSwXtch and the xNIC on AWS, Azure, GCP and OCI.
- Users **must** install an xNIC on every VM that needs to send or receive cloudSwXtch multicast or broadcast traffic.

Before you install cloudSwXtch, please review the System Requirements.

AWS cloudSwXtch Installation Guide

Azure cloudSwXtch Installation Guide

GCP cloudSwXtch Installation Guide

OCI cloudSwXtch Installation Guide

Before you install the xNIC, please review xNIC System Requirements.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

xNIC Installation Guide for Windows and Linux

xNIC Installation Guide for Kubernetes

---

wXcked Eye for Monitoring and Orchestration

wXcked Eye is a web-based monitoring and orchestration tool for cloudSwXtch. It presents users with a high-level view of their cloudSwXtch environment with an interactive network graph detailing connections to different endpoints. With an expansive look at performance metrics, users can ensure that their data is flowing as expected.

In addition, wXcked Eye unlocks the ability to configure Mesh, High Availability, Protocol Fanout, and Precision Time Protocol (PTP) from the comfort of a user's web browser.

For more information, see Using wXcked Eye for cloudSwXtch.

Testing

The xNIC installation includes the following utilities that can be used to verify both the functionality and performance of the network:

- **swxtch-top**: This utility shows detailed switch statistics in the console. For more information, click here.
- **swxtch-perf**: This utility can be used to produce and consume multicast traffic for testing. For more information, click here.
- **swxtch-tcpdump**: This utility helps capture multicast packets sent to and from the cloudSwXtch with logic to decode our own head and display the original MC payload.
- **swxtch-where**: This utility allows users to call for hardware information regarding their cloudSwXtch VM.

Each of the utilities can be run from a VM that has the xNIC software installed. Detailed usage information can be found for each by entering in the —help command-line argument.

Multicast Examples

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Users can find examples of the multicast by scrolling down to the Multicast Example section in the swxtch-perf article.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

What is cloudSwXtch?

WHAT TO EXPECT

In this article, users will get a deeper understanding of **cloudSwXtch** and how it can improve their networking capabilities. This article also gives users a preliminary introduction to the main features available while using cloudSwXtch.

Meet cloudSwXtch

cloudSwXtch creates a virtual overlay network that lets users add high performance networking to their cloud or edge applications with the touch of a button, requiring no code changes!

cloudSwXtch is available on AWS, Azure, GCP and OCI. It can be instantiated via their respective Marketplaces. cloudSwXtch is also available as a BYOL software install.

Supported Environments

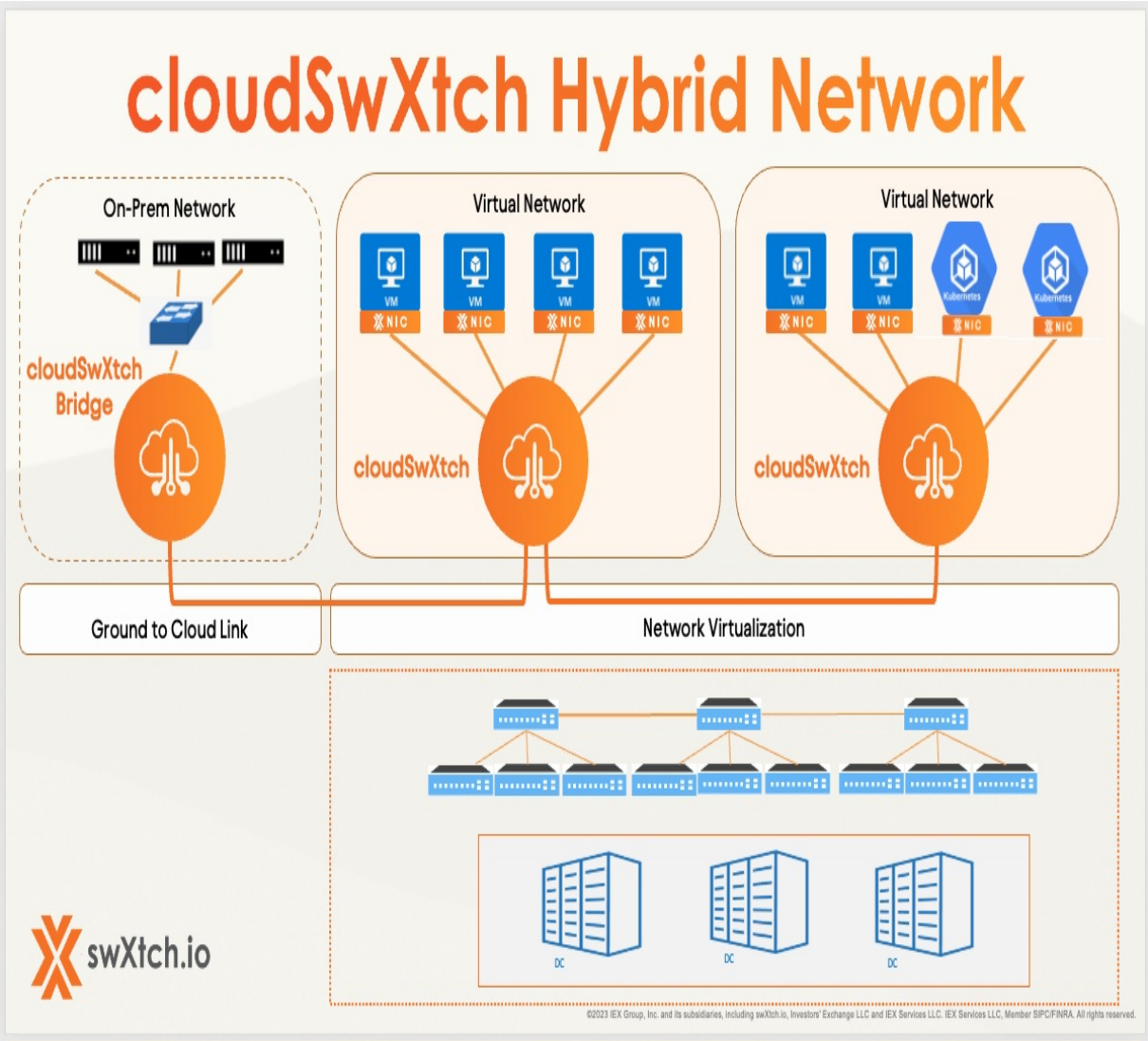
- Microsoft's **Azure**
- Amazon's **AWS**
- Google's **GCP**
- Oracle's **OCI**
- On-Premises with **cloudSwXtch Bridge**

What is a Virtual Overlay Network?

swXtch.io provides an application that implements a ***Cloud Based Virtual Switch - cloudSwXtch***. It consists of a software-based network switch and a virtual NIC service (*xNIC*). Together, these components create an overlay network on top of the standard cloud network.

This overlay network adds many valuable, high-performance network features that aren't traditionally available in the cloud; one of which is a **seamless IP multicast** experience.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



cloudSwXtch Instance

A cloudSwXtch instance running on a user's virtual machines will provide extremely low latency ($< 3\mu s$), high determinism, and elastic scalability. A user can build a 1,000-port switch or create a cloudSwXtch mesh of switches to optimize network reliability.

With *cloudSwXtch*, existing user applications and services that expect standards-based IP multicast will work in the cloud **without requiring any code changes.** This enables performance to approach that of bare metal.

Benefits of cloudSwXtch

- **Unblock Cloud Migrations** – Migrate critical workloads that couldn't move to the cloud because of missing network features or performance limitations.
- **Extend On-Prem Networks to the Cloud** – Create a single data plane across private networks and the cloud, traversing virtual networks, availability zones, and regions.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

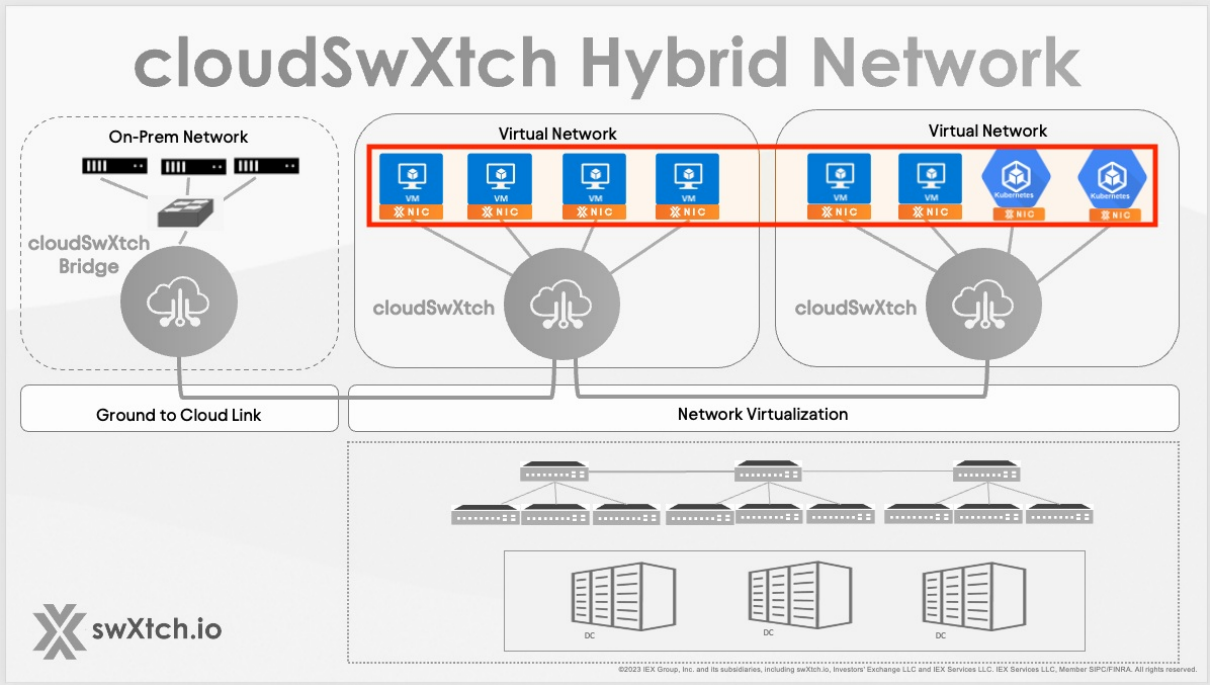
- **Massive Scale** – Extended networks with unlimited endpoints share identical features and sub-millisecond performance.
- **Enhanced Packet Monitoring** – The cloudSwXtch architecture provides a unique view into low level network traffic across the entire extended network.
- **Simplified and Flexible Network Configuration** – Add and remove endpoints dynamically from global networks as conditions dictate. Eliminate the need to reconfigure individual workloads.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

What is an xNIC?

WHAT TO EXPECT

In this article, you will learn how the xNIC, our virtual NIC, fits into the cloudSwXtch infrastructure.



A Fully Virtualized NIC

The **xNIC** is a software service deployed by swXtch.io that creates a fully virtualized **Network Interface Controller (NIC)** within a virtual machine's (VM) operating system. In tandem with the cloudSwXtch, the xNIC provides users with wXcked fast network connectivity and enables applications running on the VM to send and receive IP multicast traffic. It is seamless and transparent to all workloads, requiring no code changes.

xNIC installation for cloudSwXtch is required for multicast to work across all major clouds — **AWS**, **Azure**, **GCP**, **OCI**. It is available on both Linux and Windows machines as well as Kubernetes. You can find a more detailed list of **xNIC system requirements**, here.

In addition to multicast, the xNIC can be configured to perform high availability. At an endpoint, the xNIC compares packet reception from the multiple paths, detects dropped packets, and reconstructs the output stream in the correct order. For more information, see Configuring xNIC for High Availability.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Q: Can xNIC work on any IP within the multicast address space?

A: We recommend referring to the IANA **IPv4 Multicast Address Space Registry**. Currently, the xNIC can't join a multicast stream if it's on 224.0.0.1.

©2024 IEX Group, Inc. and its subsidiaries, including sw Xtch.io. All rights reserved.

cloudSwXtch Liscensable Features

WHAT TO EXPECT

The following section gives a preliminary introduction to the main features available while using a cloudSwXtch.

Licensing cloudSwXtch Features

In addition to Multicast and wXcked Eye, users can license the following features for their cloudSwXtch:

Protocol Fanout	**Protocol Conversion (e.g. SRT to Multicast)**	**Ground to Cloud/Cloud to Ground**	**Mesh**	**SMPTE ST 2022-7 & High Availability**
Precision Time Protocol (PTP)	**Tachyon LIVE!**	**Increase Bandwidth Capacity (Ingress)**	**Additional Endpoint Connections**	

Multicast

cloudSwXtch enables true and seamless IP-multicast. Using **multicast**, instead of unicast, optimizes a user's network configuration, reducing their cloud distribution and egress costs. In addition, receivers can dynamically subscribe and unsubscribe to a user's streams as workflows dictate. cloudSwXtch alleviates the need to have to constantly reconfigure unicast streams to accommodate downstream receivers. cloudSwXtch uses the industry standard IGMPv2/v3 for its management of multicast group membership.

For more information, check out the Multicast feature article.

Single Source Multicast (SSM)

Traditionally, Single Source Multicast (SSM) is a method for delivering multicast packets in which the only packets that are delivered to the receiver are those originating from a specific source address requested by the receiver. This can be accomplished as either a consumer command for swxtch-perf, the cloudSwXtch-based tool for simulating traffic movement, or via an external application.

Protocol Conversion and Fanout

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

cloudSwXtch supports a unique feature called **protocol conversion and fanout**. This feature is useful when a user's multicast application needs to stream to an endpoint that does not support multicast or it is not possible to install an xNIC in the endpoint. cloudSwXtch can map a multicast group address to a unicast address. Similarily, a unicast input to cloudSwXtch can be mapped to a multicast group enabling mulitple endpoints to consume the original unicast input stream. Protocol Fanout converts many packet protocols and distributes them out as if they were multicast; freely integrating multicast, unicast and Secure Reliable Transport (SRT) streams while making the network more efficient and reducing egress costs.

For more information, check out the Protocol Conversion and Fanout feature article.

SMPTE 2022-7 and High Availability (HA)

High Availability (HA) protects users against data path errors by sending the same stream through as many as eight different distributed data paths. It compares packet reception from the multiple paths, detects dropped packets and reconstructs the output stream in the correct order. This feature is compliant with SMPTE 2022-7 for media workflows.

For more information, check out the High Availability feature article.

Mesh

Multiple cloudSwXtches can be connected together in a mesh for routing throughout the cloud network. This includes cloudSwXtches in any topology across all dispersed network locations (different Vnets, regions, clouds, subnets, etc.). Additionally, a mesh allows cloudSwXtch to scale vertically.

For more information, check out the Mesh feature article.

Ground to Cloud <==> Cloud to Ground

A user can connect their On-Prem network to their cloudSwXtches in the Cloud via the bridge application.

For more information, check out the Installing cloudSwXtch Bridge guide.

wXcked Eye for Monitoring and Configuration

cloudSwXtch also provides its users with visibility down to the packet level for enhanced Monitoring and Quality of Service (QoS). **wXcked Eye** is the cloudSwXtch monitoring UI tool that enables users to deeply audit the performance of their cloudSwXtch network. Each cloudSwXtch performs complete packet capture.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

In addition, wXcked Eye also provides users with an additional avenue to configure their cloudSwXtch environment for mesh, high availability, protocol conversion and fanout, and precision time protocol.

A REST API is provided to help users manage and control their network in their own way.

For more information, please see the Using wXcked Eye for cloudSwXtch article.

Precision Time Protocol (PTP)

Precision Time Protocol (PTP) is a cloudSwXtch feature that facilitates clock synchronization between agents connected to the network. The cloudSwXtch acts as the **Master Node**, passing on the information gained from the true clock source to the **Follower Nodes** or agent end points.

For more information, please see the Precision Time Protocol (PTP) feature article.

Tachyon LIVE!

Tachyon LIVE! is a live, high-quality standards, format, and frame rate converter software stack that maximizes video quality across all conversions. It performs standards conversion including PAL/NTSC frame rate and format conversions, high-quality deinterlacing, and up/down rescaling from SD through HD, and it can process the highest-quality conversions for HD 59.97p to HD 50p, faster than real time in a VM with NVIDIA GPU-accelerated infrastructure.

This feature is an exclusive offer for cloudSwXtch from Cinnafilm and available as either an HD or UHD add-on.

For more information, please visit Cinnafilm's website.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Multicast

Multicast

Software defined multicast (SDMC™) is a feature of the **cloudSwXtch** overlay network. With SDMC, existing applications and services that expect standards-based, IP multicast will work **without requiring any code changes** and with performance that approaches that of bare metal.

At a high level, **cloudSwXtch** implements a ***software switch*** that serves the same role as a hardware switch. **cloudSwXtch** receives multicast packets from producers and sends a copy of each packet to every destination VM. The **cloudSwXtch** control plane uses the industry standard IGMPv2/3 specification for the management of group membership.

The **xNIC** service handles multicast traffic between the switch and the VM operating system. The xNIC service must be installed on every VM that needs to send or receive multicast traffic.

SUMMARY

The **cloudSwXtch** system consists of a software switch instantiated within a virtual network and a set of virtual machines that have an xNIC virtual interface.

Applications can send and receive IP multicast by targeting the virtual network interface. IGMP control packets are generated by the local operating system and the xNIC virtual interface seamlessly picks these up and sends them to the **cloudSwXtch** instance. Local applications will work in this environment just as they would on a similar bare-metal network.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Source Specific Multicast

WHAT TO EXPECT

In this section, we will go into detail about Source Specific Multicast, or SSM, and how it improves security when sending/receiving multicast packets.

Source Specific Multicast, or SSM, is defined as a method for delivering multicast packets in which the only packets that are delivered to the receiver are those originating from a specific source address requested by the receiver. Not only does this improve security within the cloudSwXtch but it also alleviates strain on the network since the sender will know to only send a multicast stream from the specified single source and not via other source addresses.

This feature can be tested using swxtch-perf, a cloudSwXtch-based tool for simulating consumer and producer traffic, as well as external applications that support SSM. For more information on using it for SSM, please see the swxtch-perf article under Testing cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Broadcast

Broadcast is a feature of the cloudSwXtch overlay network. With Broadcast, existing applications and services that expect standards-based broadcast will work without requiring any code changes and with performance that approaches that of bare metal.

At a high level, cloudSwXtch implements a software switch that serves the same role as a hardware switch. cloudSwXtch receives broadcast packets from producers and sends a copy of each packet to every destination VM.

The xNIC 2 service handles tunneling broadcast traffic between the cloudSwXtch and the VM operating system. The xNIC 2 service must be installed on every VM that needs to send or receive broadcast traffic.

SUMMARY

The cloudSwXtch system consists of a software switch instantiated within a virtual network and a set of virtual machines that have an xNIC 2 virtual interface.
Applications can send and receive broadcast by targeting the virtual network interface.
Broadcast packets are generated by the local operating system and the xNIC 2 virtual interface seamlessly picks these up and sends them to the cloudSwXtch instance. Local applications will work unchanged in this environment just as they would on a similar bare-metal network.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

High Availability

WHAT TO EXPECT

High Availability (HA) is an implementation of data path redundancy and stream duplication. It protects users from data loss by replicating and sending packets through multiple network paths. xNIC compares packets received from those multiple paths and automatically reconstructs the original stream.

In this section, users will learn more about the benefits of implementing the High Availability feature in their cloudSwXtch and understand how to leverage it for their future needs.

Creating A More Resilient Network

With High Availability, critical workloads can be configured to be more resilient, stretching across regions or availability zones in a single cloud. In addition, it can be used across multiple cloud providers. Although there can only be up to eight redundant paths, there are no limits to the number of consumers that can receive the HA stream, other than bandwidth constraints.

In addition, there is no limit to the number of multicast groups per data path. If one cloud, availability zone or region should go down, then the data is still sent in the other 2-8 paths, ensuring that the consumer gets the necessary data. Consumers can also be put into different clouds, availability zones or regions so that if a consumer becomes unavailable, users can still sign into a different cloud, availability zone or region and get the data desired.

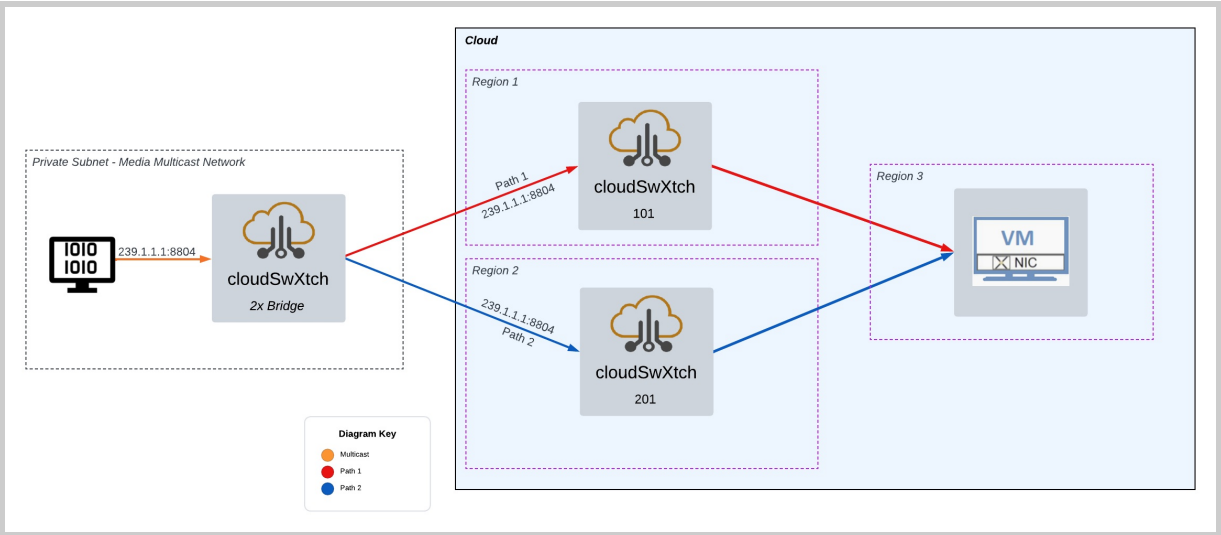
The HA feature forwards packets to the receiving application from any of the configured paths as soon as the "next" expected packet is received. Redundant packets from other paths are discarded. There is no additional latency imposed by the HA feature.

IMPORTANT

A cloudSwXtch configured in a HA path cannot be used in a cloudSwXtch mesh. They are mutually exclusive.

High Availability Example

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



The simple diagram above shows high availability with one multicast group 239.1.1.1:8804 originating from an on prem source. From the bridge, two paths are created with redundant packets being sent to alternate cloudSwXtches in different regions. The number of regions and cloud providers needed for High Availability will vary depending upon the customer's environment.

Independent path redundancy ensures no packet loss if every packet arrives at the consumer from at least one path. For example:

- **In the event that cloudSwXtch101 goes offline**, the consumer will still get the multicast traffic via cloudSwXtch201 (or vice versa).
- **In the event that there are network issues in Region 1 where some of the packets are lost in path one**, the consumer can still get the multicast traffic with High Availability pulling data from Region 2 in path two.
- **In the event that there are network issues in Region 1 and 2 where some of the packets are lost in both paths**, both consumers can still get the multicast since the high availability function will take the valid packets and reconstruct the multicast stream from Region 1 and 2.

In each example, despite losing paths, multicast data was still able to get to the end point using high availability with no packet loss. Configuring more paths will ensure higher availability of the multicast group.

HA can be monitored via swxtch-top, see swxtch-top section 4-6.

To configure the system for high availability, refer to: High Availability Configuration.

Installing cloudSwXtch - Firewall Exceptions

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

When installing the cloudSwXtch, high availability requires special firewall exceptions. To learn more, see **cloudSwXtch System Requirements**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Mesh

What is a Mesh?

A mesh connects cloudSwXtches in a variety of dispersed network locations – different Vnets, regions, clouds, subnets, data centers, ect.). Additionally, a mesh is a way to group two or more swXtch's together to act as one to gain network performance.

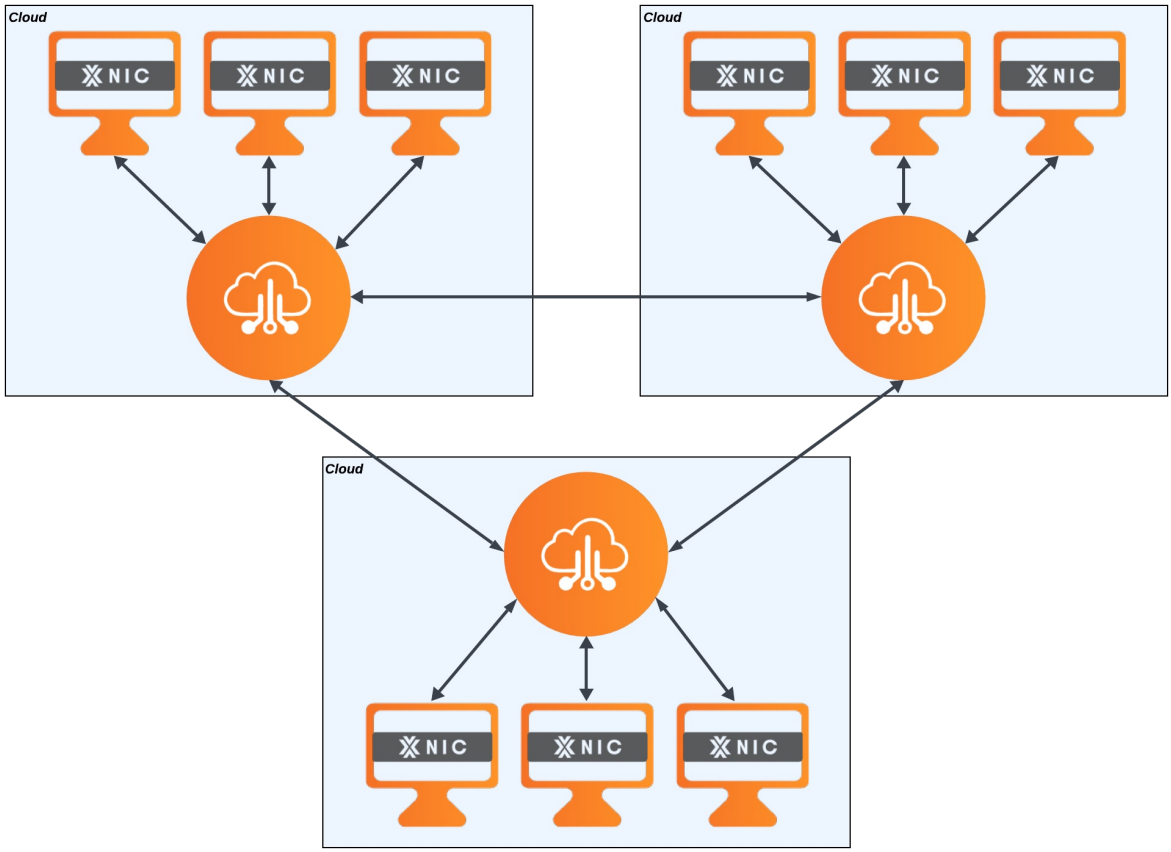
Learn more about a Mesh

- See **Monitor cloudSwXtch with wXcked Eye** to learn more about monitoring cloudSwXtches to understand existing capacity to know if you need to consider creating a cloudSwXtch mesh.
- See **cloudSwXtch Installation** for installing a cloudswXtch.
- See **Mesh with wXcked Eye** for mesh configuration.

Mesh

A Mesh is formed by linking cloudSwXtches so that they are eligible to receive and transmit multicast traffic to other cloudSwXtches in the same mesh. Configuring a mesh with wXcked Eye allows a user to create a network of cloudSwXtches across Availability Zones, Regions, and On-Prem Networks to manage multicast traffic.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



NOTE

- A member of a mesh is called a swxtch-node, or simply node.
- Mesh membership is managed by via a REST API and a CLI tool.
- A node can be added as long as it is reachable via IP traffic. This means a node can be in any other VNet as long as IP traffic can be routed between at least one other node in the mesh.

IMPORTANT

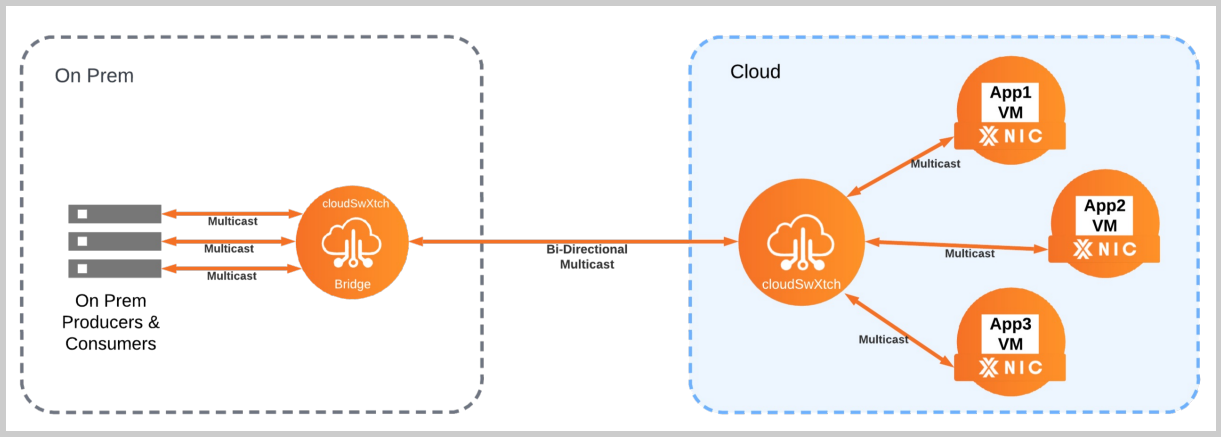
- A cloudSwXtch configured for mesh cannot be used in an HA path. Mesh and High Availability are mutually exclusive.
- Mesh membership doesn't mean that all multicast traffic is sent to every other node in the mesh. Packets destined for a multicast group are only sent to nodes that have consumers that have joined the same multicast group.
- A cloudSwXtch can only be a member of a single mesh.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bridge

cloudSwXtch Bridge

The **cloudSwXtch Bridge** application enables bi-directional multicast traffic between a non-cloud and cloud network. The source network can be bare-metal and on-premises. The destination network can be a cloud virtual network with a **cloudSwXtch** instance deployed. With **cloudSwXtch**, multicast traffic generated from the on-prem network can be received and processed in the cloud which then in turn can be sent back to the on-prem network.



The cloudSwXtch Bridge is bi-directional. It sends multicast traffic from the on-premises network to the cloud and from the cloud to on-premises.

From on-prem to the cloud, the bridge is dynamic. This means that users in the cloud can subscribe to a multicast group via IGMP joins. Then, the bridge will allow that traffic through. This ensures that only necessary traffic goes through the VPN or Express Route/Gateway into the cloud. It guarantees the best use of the gateway and incurs less ingress bandwidth into the cloud.

Operation

The operation of the cloudSwXtch Bridge varies based on direction.

Ground-->Cloud

For Ground to Cloud, a mesh must be configured between the cloudSwXtch and the cloudSwXtch Bridge at the ground. From then on, the operation is dynamic, meaning the user does not need to map multicast addresses to go into the cloud. Instead, when a user is in an application and use an IGMP join then a message is sent to the cloudSwXtch Bridge via the cloudSwXtch through the mesh and then the Bridge allows that traffic through. When the user stops using multicast group and does an IGMP leave, then the bridge stops sending multicast data.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Cloud-->Ground

For Cloud to ground there is no current support to propagate IGMP joins and leaves from cloudSwXtch to on-prem. In this case, multicast groups must be explicitly configured to let the bridge know what traffic is allowed.

See Bridge Installation on how to install the Bridge and the differences between Bridge Type 1 and Type2. See our configuration pages for Bridge Type 1 and Bridge Type 2.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

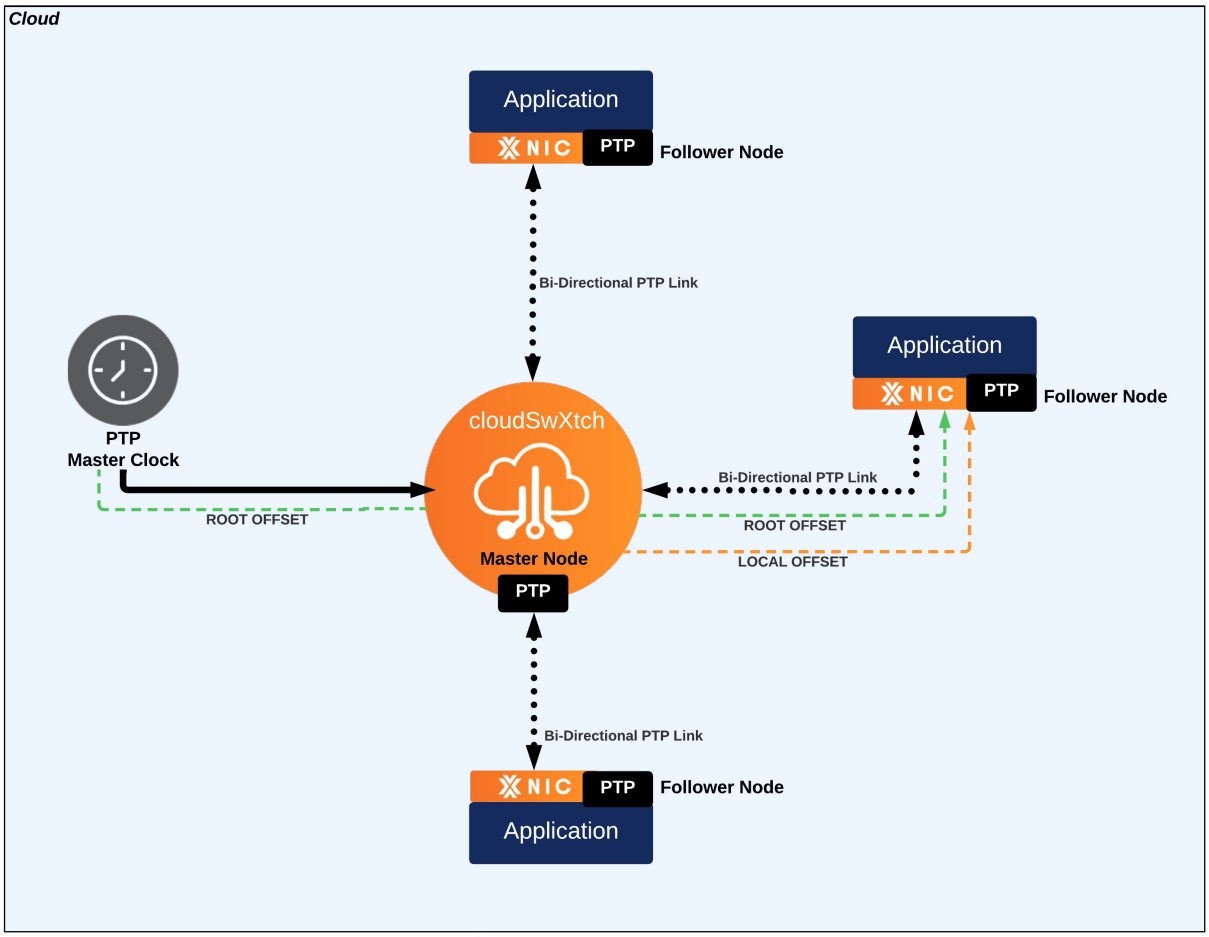
Precision Time Protocol

WHAT TO EXPECT

In this article, users will learn how Precision Time Protocol (PTP) works in a cloudSwXtch environment when the feature is activated.

What is Precision Time Protocol?

Precision Time Protocol (PTP) is a cloudSwXtch feature that facilitates clock synchronization between agents connected to the network. The cloudSwXtch acts as the **Master Node**, passing on the information gained from the true clock source to the **Follower Nodes** or agent end points.



Information regarding PTP will display in both swXtch-top under the PTP page and wXcked Eye under Timing Nodes. Both cloudSwXtch tools will show the local and root offset. The **local offset** denotes the offset in time from the cloudSwXtch to the xNIC. The **root offset** denotes the offset in time from the True Clock Source and the cloudSwXtch's follower nodes (xNICs). The root value will always be larger than the local since the distance between the follower node and the True Clock Source is greater than the offset between a cloudSwXtch and xNIC.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Protocol Conversion and Fanout

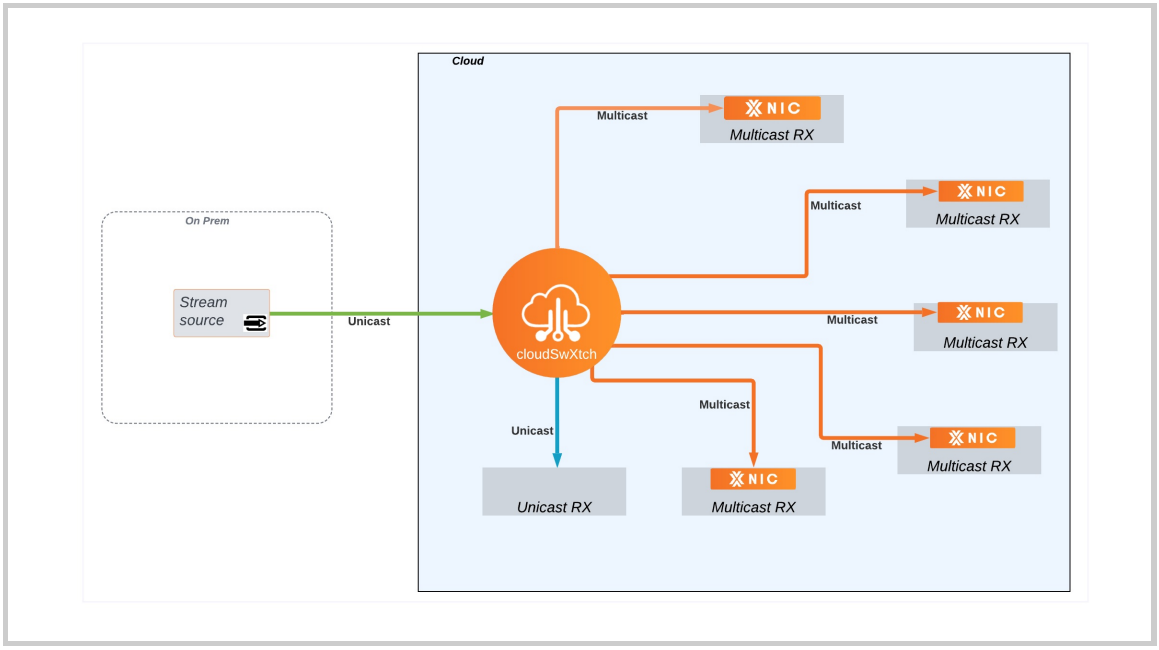
WHAT TO EXPECT

Protocol Conversion and Fanout allows users to send copies of a single input stream in any supported protocol to multiple destinations. This gives each destination the option of being a different protocol from the input stream. An example would be a UDP input being sent to a set of multicast destination and, additionally, to one using SRT.

In this section, users will become more familiar with how Protocol Conversion and Fanout works in a cloudSwXtch-enabled environment.

What is Protocol Conversion and Fanout?

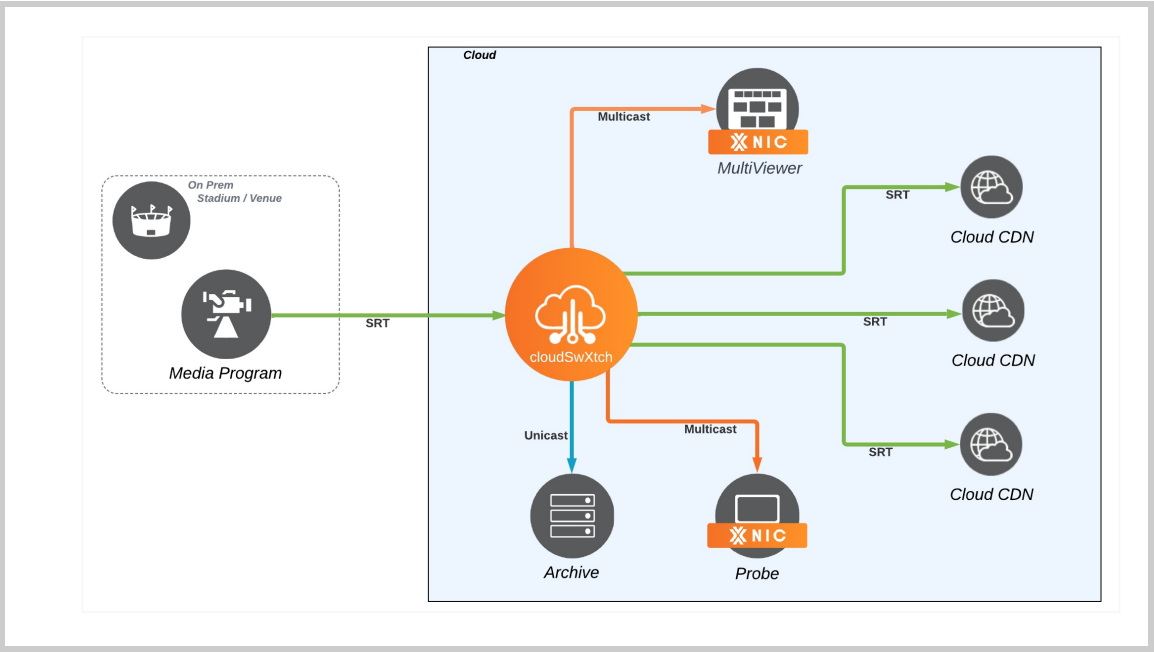
It is not usual for workflows to have many endpoints with each having their own protocols: SRT, Unicast and Multicast. Configuring each device can be a difficult and time consuming endeavor and up until now, impossible in the cloud. However, with cloudSwXtch, that is no longer the case. Users can convert protocols and send out multiple copies of payloads to different receivers regardless of protocol type without the need to add custom software or hardware.



This is a generic depiction of a Protocol Conversion and Fanout configuration.

In the above example, unicast data flows from the stream source into the cloudSwXtch. With Protocol Conversion and Fanout enabled, the cloudSwXtch can convert the unicast stream into multicast and fan it out to multiple receivers. In addition, the cloudSwXtch is sending out the stream to a unicast receiver. **Please note**: While it is not depicted in the above example, the cloudSwXtch can send the stream out to multiple unicast receivers.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



This is a media depiction of a Protocol Fanout configuration.

In the alternative example, SRT data flows from the stream source into the cloudSwXtch. With Protocol Fanout enabled, the cloudSwXtch can convert the SRT stream into multicast and fan it out to multiple receivers. In addition, the cloudSwXtch is sending out the stream to a unicast receiver and to multiple SRT receivers. **Please note**: While it is not depicted in the above example, the cloudSwXtch can send the stream out to multiple unicast receivers.

Understanding Endpoints

Workflows often have many endpoints: some requiring unicast, and some requiring multicast. Configuring for each device can be difficult and supporting both unicast and multicast for the same stream requires custom software or hardware. cloudSwXtch has the ability to map multicast streams to unicast streams and unicast streams to multicast streams allowing non-xNIC endpoints to participate in the cloudSwXtch network. This feature actualizes two different scenarios:

1. **Non-xNIC** producers, such as, those external to the cloud can send traffic to the cloudSwXtch, via unicast or SRT. The cloudSwXtch, then, can map that unicast or SRT stream to a multicast group for consumption within the cloudSwXtch network.

2. **Non-xNIC** consumers can receive traffic from a cloudSwXtch, as multicast streams can be mapped to unicast or SRT endpoints. This implies that non-xNIC consumers can receive packets created from a xNIC producer.

xNIC consumers and producers can consume SRT, unicast, and/or multicast based on consumer/producer workflow. For example, a VM may have 3 applications installed with each requiring a different protocol. The cloudSwXtch can send all three in the event that all three are needed.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Configuring Protocol Conversion and Fanout for cloudSwXtch

Users can configure Protocol Conversion and Fanout using two methods:

- Via wXcked Eye
- Via API - Please see the section on Protocol Fanout.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

cloudSwXtch System Requirements

WHAT TO EXPECT

In this article, users will learn about the system requirements needed to successfully deploy a cloudSwXtch. It is recommended for a user to review this page before installing a cloudSwXtch any of the four cloud platforms.

cloudSwXtch Sizing Guidelines

Sizing and Feature Selection For Your cloudSwXtch

The number of endpoints and bandwidth dictate cloudSwXtch sizing requirements. It is recommended for users **to contact a swXtch.io sales representative** to discuss cloudSwXtch sizing and additional features so that the appropriate license can be distributed.

Please note: A cloudSwXtch BYOL offering will not work without a license.

- **Sizing**: For bandwidth greater than 2 Gb/s and endpoints greater than 100, you will need different virtual CPUs/NIC sizing.
- **Adding Features:** Many additional licensable features are available for cloudSwXtch. For more information, see **cloudSwXtch Features**.

To contact sales, please visit **swXtch.io/contact**.

cloudSwXtch BYOL (Marketplace)

# Endpoints	Bandwidth	Core	Memory	Hard Drive
Up to 100	2 Gb/s (max)	16+	16GB DDR	64GB SSD
Up to 200	More than 2Gb/s	64+	16GB DDR	64GB SSD

cloudSwXtch PAYG for POCs/Trial

# Endpoints	Bandwidth	Core	Memory	Hard Drive
Unlimited	Unlimited	12+	48GB DDR	*

*Hard Drive is dependent on the user's needs. **Note**: cloudSwXtch Pay As You Go (PAYG) is a $30/hr offering.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Internet Connection

Installing and upgrading cloudSwXtch **requires** internet connection. Alternatively, if a user ***does not have access*** to the internet, they can use the Air-Gapped installation guide for Azure.

Supported Cloud Environments

- Amazon's AWS Cloud
- Microsoft's Azure Cloud
- Google's GCP Cloud
- Oracle's OCI Cloud

Virtual Network

A cloudSwXtch instance **must have 2 NICs**. However, both NICs can share a single subnet for control and data plane communications. This is the preferred method.

In the event that a user needs higher performance, a user can separate their subnets as described below.

- Contain a subnet for control plane traffic (referred to as the **ctrl-subnet** from here on).
- Contain a subnet for data plane traffic (referred to as the **data-subnet** from here on).

Please note: GCP does not allow for single subnet configuration. A user must have 2 separate subnets for their data and control NICs.

> **Subnet Selection**
>
> The subnets must be the same subnets used for the xNIC installations.

The virtual network and subnets may be shared with other services in addition to the **cloudSwXtch**. The size of each subnet should include at least 32 addresses.

Minimum CPU and Memory

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

A cloudSwXtch must be a minimum of 8 cores and 16 GiB memory.

Firewall and Security Group Rules

The xNIC software and the cloudSwXtch communicate with each other using the following protocols and ports. These firewall exceptions must be allowed in the xNIC VMs and the cloudSwXtch VM.

Subnet	Protocol	Ports	VM
ctrl-subnet	http	80	cloudSwXtch
ctrl-subnet	udp	10800-10803	all
data-subnet	udp	9999	all

Mesh and High Availability

Both Mesh and High Availability need special firewall exceptions in order to properly work in a user's cloudSwXtchenvironment. If you plan on using either features, please allow the following:

Mesh

Subnet	Protocol	Ports	VM
ctrl-subnet	tcp+udp	37856	cloudSwXtch

High Availability

Subnet	Protocol	Ports	VM
ctrl-subnet	tcp+udp	37856	cloudSwXtch

Reminder: HA Mesh are mutually exclusive and cannot be used together.

PTP

PTP needs special firewall exceptions in order to properly work in a user's cloudSwXtch environment. If you plan on using the feature, please allow the following:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Subnet	Protocol	Ports	VM
ctrl-subnet	http	80	cloudSwXtch
ctrl-subnet	udp	10800-10803	all
data-subnet	udp	9999	all

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

cloudSwXtch on AWS

Pre-Creation Steps

Before creating an EC2 instance with cloudSwXtch installed for AWS, users must already have an AWS account ***and*** a VPC (Virtual Private Cloud) already created.

Installation Method:

1. Review system requirements
2. Validate subnets on AWS
3. Verify security groups *Optional*
4. Create SSH key pair
5. Install cloudSwXtch on AWS

Disclaimers

- swxtch.io does not handle any policy access rights for deployment nor does it have any special IAM roles or policies that are needed. That being said, swxtch.io suggests using a policy of least privilege for all access granted as part of the deployment. Please refer to AWS for best practices for policy rights and IAM roles and policies: **AWS Identity**
- swxtch.io does not require any public resources for deployment such as Amazon S3 buckets.
- swxtch.io cloudSwXtch installation does not use any AWS Secrets in Secret Manager as swxtch.io does not natively store any customer sensitive data. Customers can encrypt their traffic and the cloudSwXtch will still be able to handle the network traffic.
- swxtch.io does not encrypt data. It pass through any data sent in the multicast which may be encrypted.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Validate Subnets on AWS

WHAT TO EXPECT

A virtual network must be created before deploying a cloudSwXtch EC2 instance.

- It must contain at least one subnet that's used for both the control and data plane communication.
    - It is recommended that it is **private facing** and **does not auto-assign public IPs**.
    - This single subnet will be used for xNIC installation.

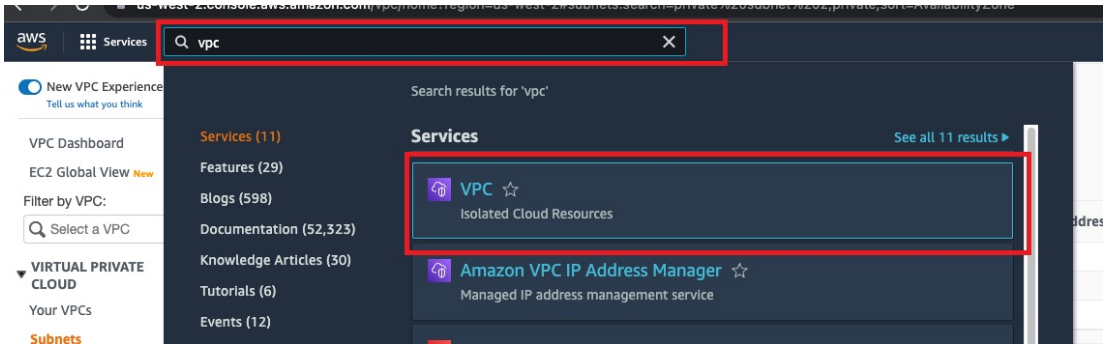
In this section, users will learn how to validate whether a subnet exists to be used as both the control and the data plane for their virtual network. This is in preparation for cloudSwXtch installation on AWS. We will also walk through an alternative method of using 2 subnets, separating the control and data plane.

Method #1: Single-subnet

Typically, when deploying a VPC, a user will automatically create a subnet. During the main installation process, this subnet can be used for both control and data plane communications. This is the **preferred** method and will be used by a majority of users. Before installing cloudSwXtch, users should validate that the control subnet exists.

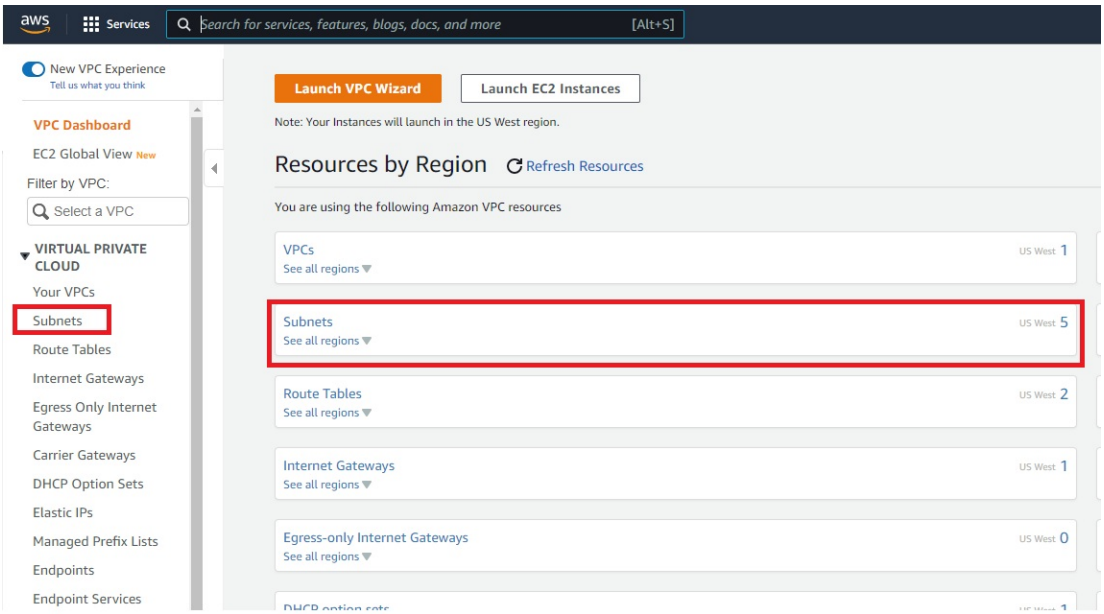
To validate:

1. **Navigate** to the VPC Console in AWS. In the example below, the user entered VPC in search field to find it under Services.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. **Select** "Subnets" under the Virtual Private Cloud tab or under Resources by Region in the VPC Dashboard.



3. **Check that the subnet you wish to use for the cloudSwXtch is listed.** In addition to the cloudSwXtch installation, this single subnet will be used during xNIC installation.

Method #2: Two Subnets

Alternatively, a user may decide that they want to have two separate subnets for their cloudSwXtch: one for the control plane and another for data. In addition, the same subnets must be used for the xNIC installations. This method is recommended for individuals who want higher performance.

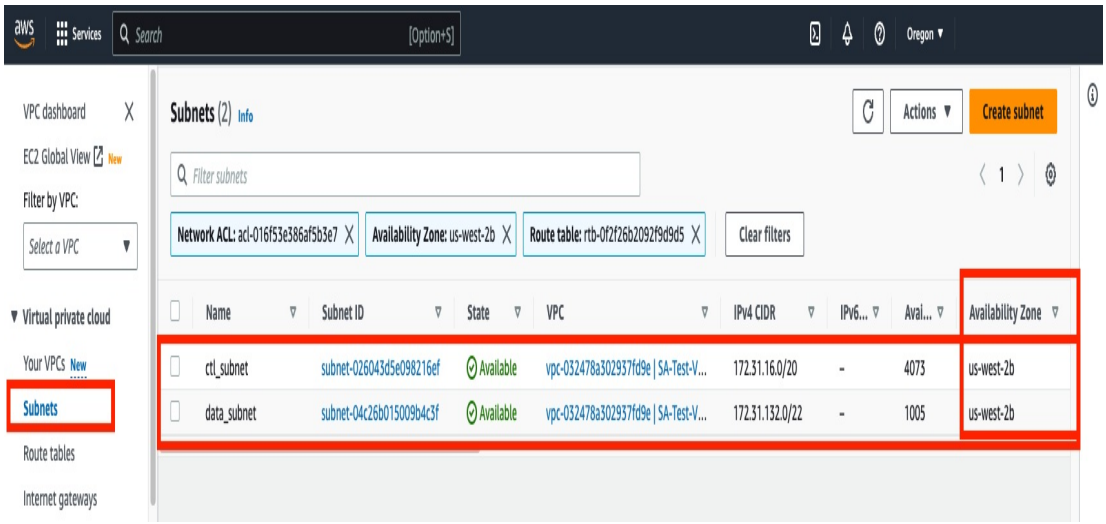
To accomplish this:

1. Navigate to the **VPC Console** in **AWS**.
2. Select **Subnets** under the **Virtual Private Cloud** tab or under **Resources by Region** in the **VPC Dashboard**.

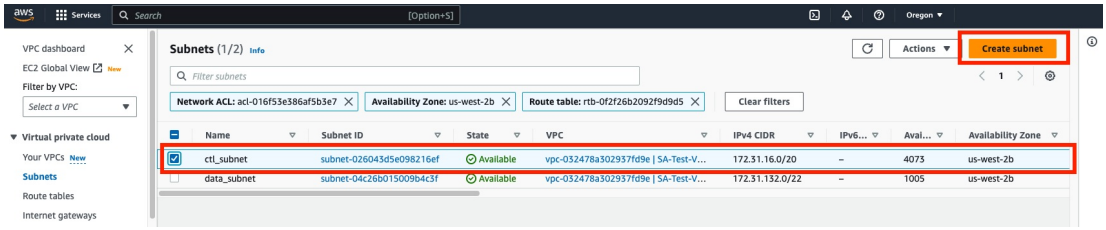
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. Check that 2 subnets exists: one for the data and another for the control plane. Ensure that both subnets are in the same **Availability Zone**. This allows be both NICs to be connected on the EC2 instance at the same time.

> **Naming your subnets**
>
> For ease of use, name the subnets are ctrl-subnet and data-subnet to distinguish between them when creating an EC2 instance with cloudSwXtch installed.

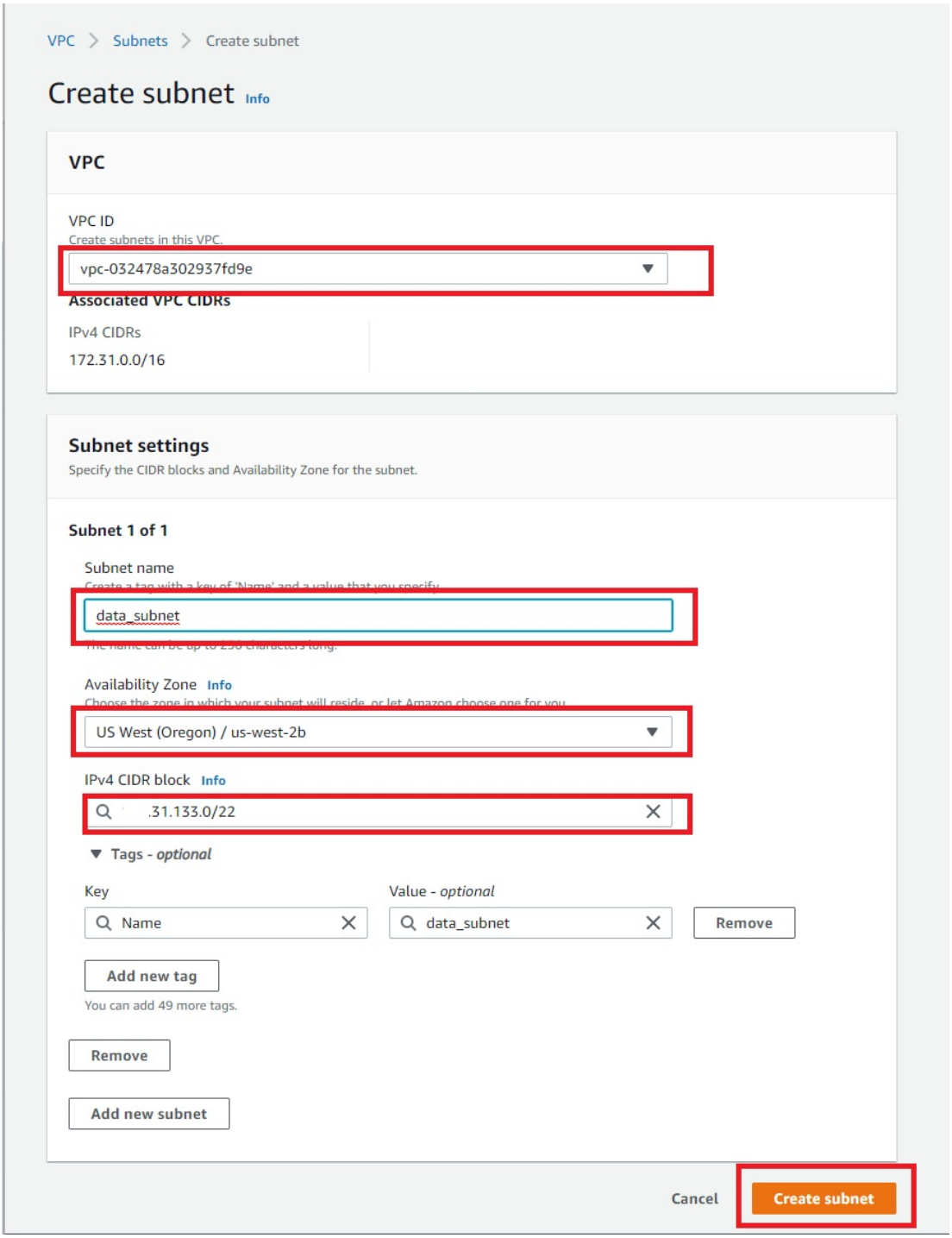


4. **If a second subnet does not exist**, select the orange **Create Subnet** button in the top right corner of the page.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

5. Fill in the **Create Subnet** form like the example shown below, ensuring that the subnet is in the same **VPC ID** and **Availability Zone** as your other subnet. In the example below, the user is creating their data subnet.



6. **Click "Create Subnet."** You should now have a new subnet on your list.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

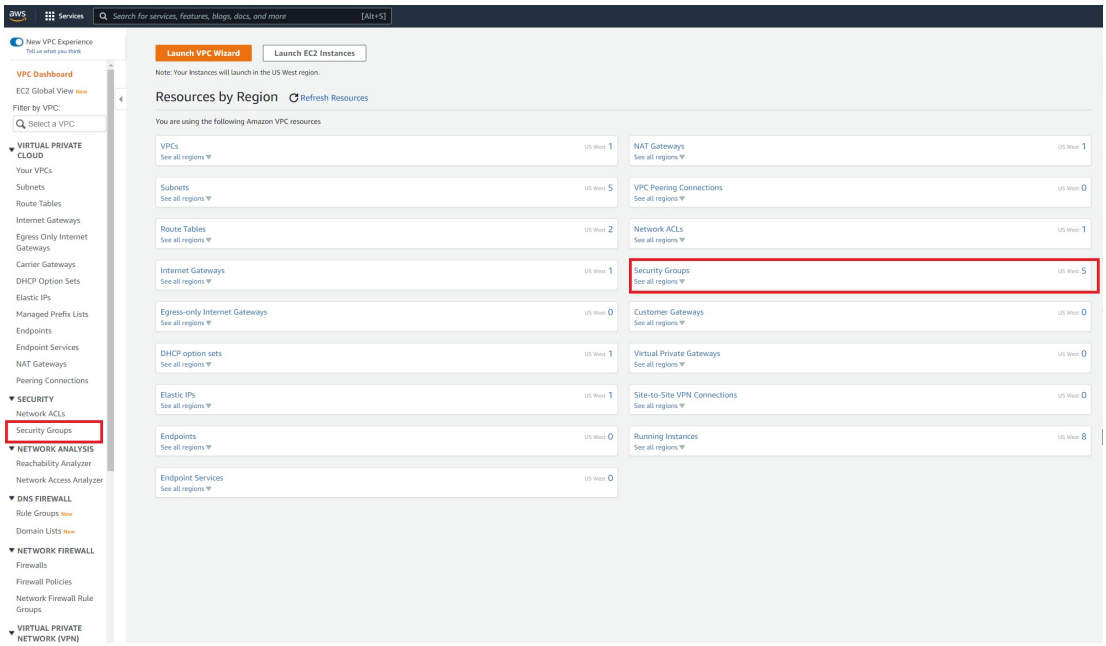
Verify Security Groups

The security group contains the firewall settings for EC2 instances and interfaces (xNICs).

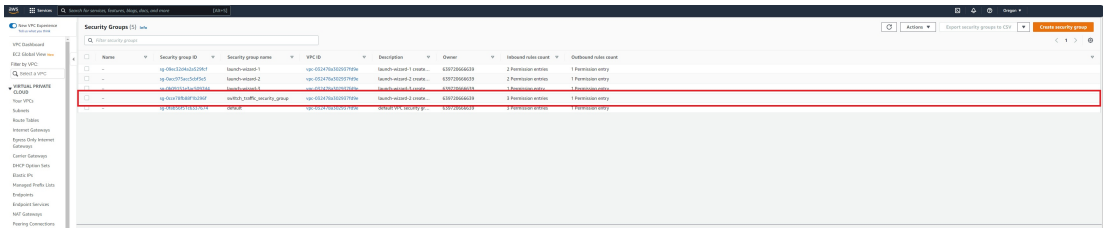
To ensure security groups are set up properly for cloudSwXtch:

1. **Navigate** to the VPC console.

2. **Select** the "Security Groups" link as shown below. (**Note**: There are multiple ways to get to the "Security Groups" page.)

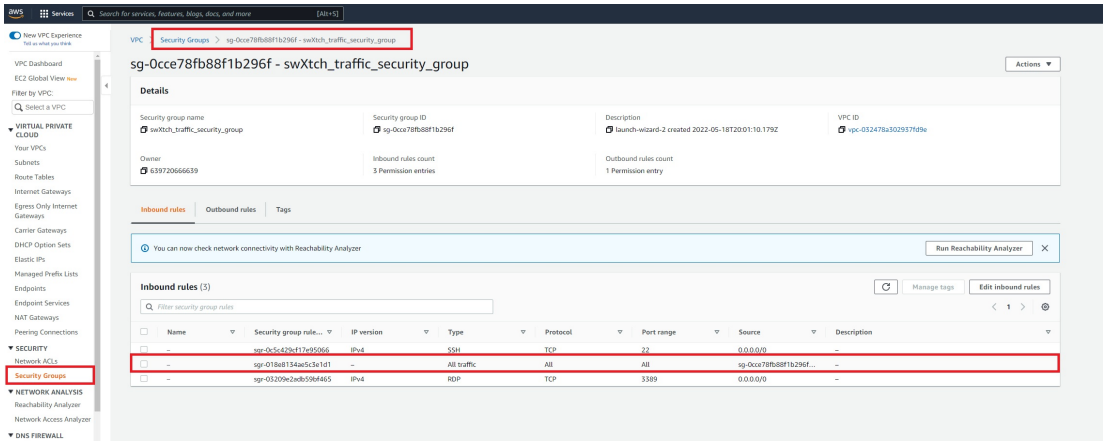


3. **Select** the Security Group that is normally used to create your EC2 instances for your application. (Note: The names in the example will be different in your environment.)

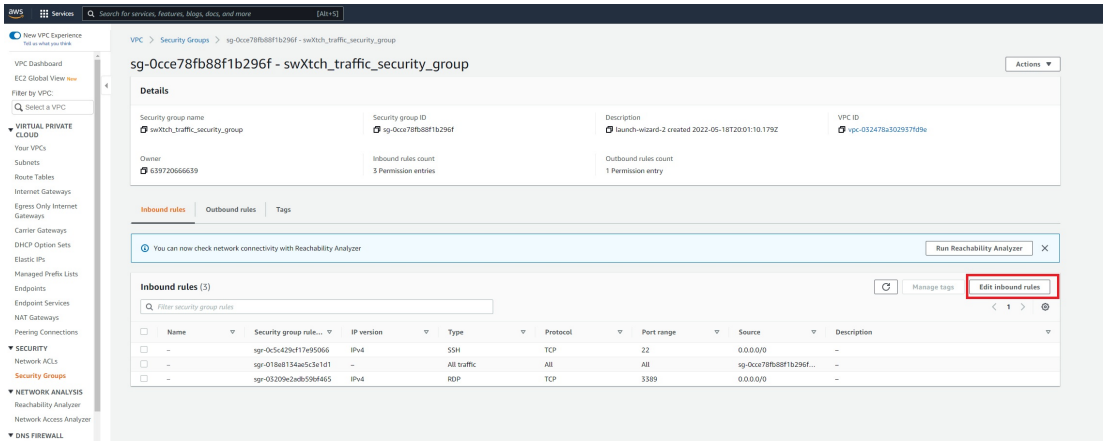


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. In order for certain features to work in your cloudSwXtch, you will need to add inbound rules to open specific ports originating from that security group. You can find the ports outlined in the cloudSwXtch System Requirements article under "Firewall and Security Group Rules."

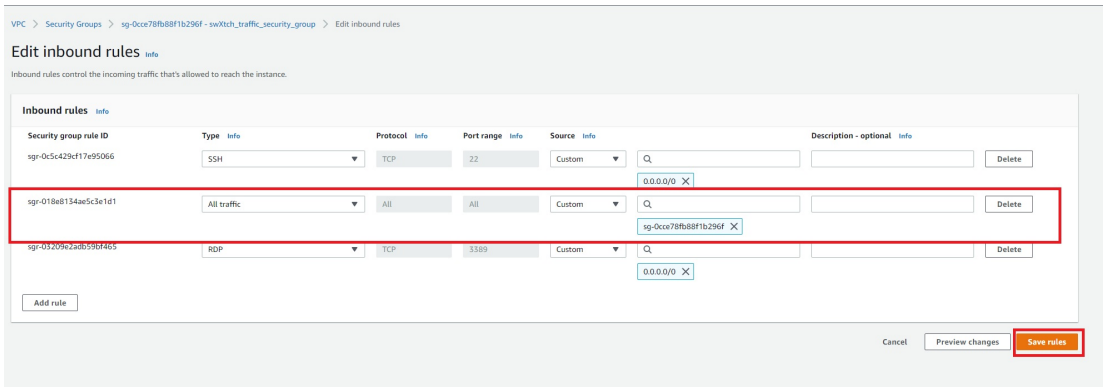


5. **If an inbound rule does not exist**, create it by selecting "Edit inbound rules."



6. **Select** "Add Rule."

7. **Enter** the information like the screenshot shown below verifying that the ID of the SG on Source matches the SG you are editing.



8. **Save** the rule.

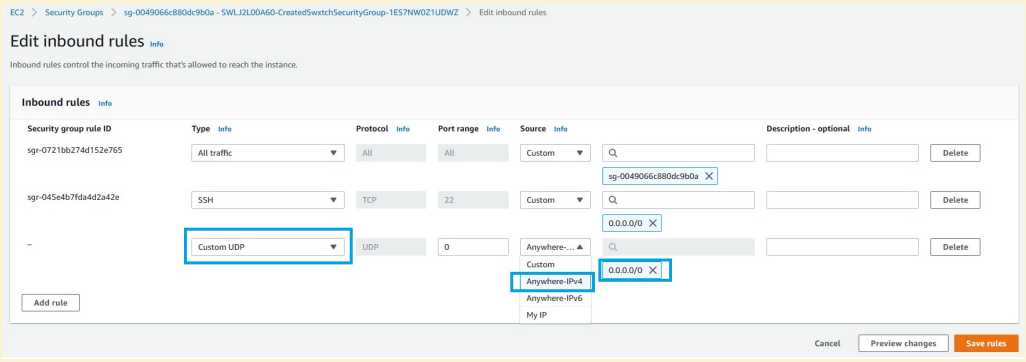
Additional Rules

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Mandatory Inbound Rule For Mesh

In order to use the Mesh feature bidirectionally between VPCs, users must also add the following inbound rule to each SG:

- **Type**: Custom UDP
- **Protocol**: UDP
- **Port Range**: 9999
- **Source**: Custom/Anywhere-IPv4 0.0.0.0/0



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

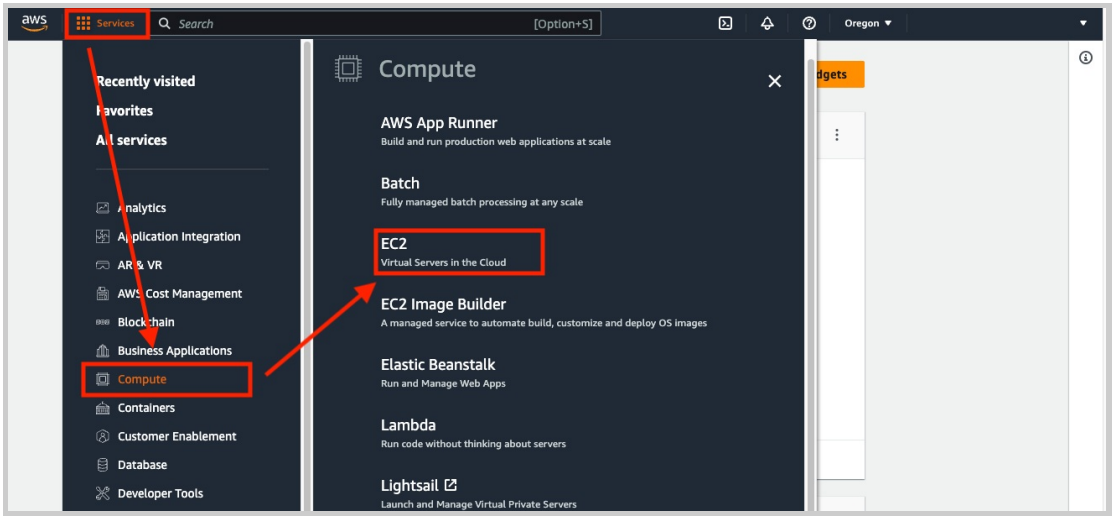
Create SSH Key Pair

WHAT TO EXPECT

An SSH key pair is necessary when accessing a cloudSwXtch EC2 instance. If you do not already have one imported, please create an SSH key pair before beginning the cloudSwXtch on AWS creation process.

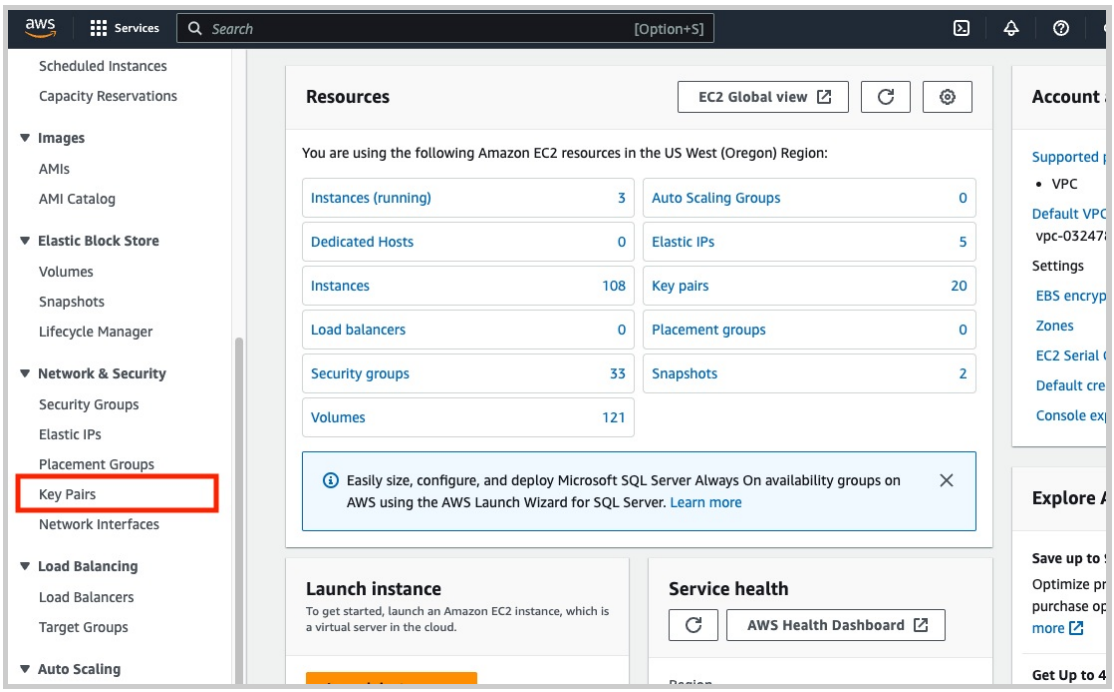
In the *AWS Management Console*, make sure you are in the region where you plan to use the cloudSwXtch instance.

1. **Navigate** to EC2
    - **Select** the “Services” menu in the AWS Management Console.
    - **Click** “Compute”
    - **Select** “EC2”



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. In EC2, **click** “Key Pairs” under the “Network & Security” tab in the menu on the left-hand side.



3. **Click** “Create Key Pair”. A new window should open.

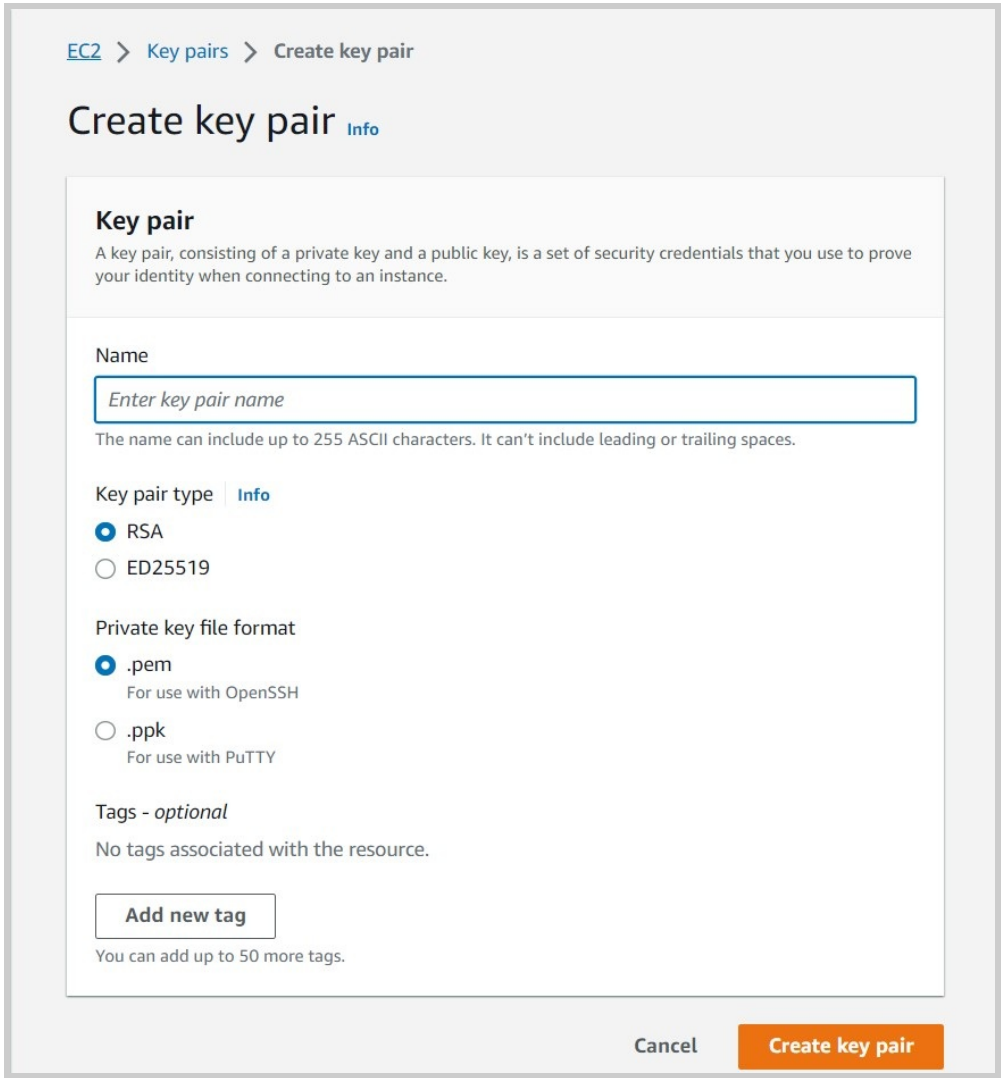
4. Under **Name**, enter something meaningful and descriptive for the key.

5. Depending on your needs, you have to choose **RSA** or **ED25519**, and **.pem** or **.ppk** (OpenSSH or PuTTY access).

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

6. Click on **Create Key Pair**.

    a. A file with the chosen extension will be downloaded to your computer (secret private key), and the other half of the pair will be store on AWS for later use (public key, used in conjunction with your private key to validate the access).



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install cloudSwXtch on AWS

WHAT TO EXPECT

Deployment of a cloudSwXtch consists of two parts: the creation of an EC2 instance containing cloudSwXtch and the installation of the xNIC software. The cloudSwXtch is considered "installed" once while the xNIC is installed on each agent instance that is a part of the network.

In this section, users will learn how to deploy cloudSwXtch for their AWS environment.

NOTE:

Root privileges are not required for deployment or operation. Our CloudFormation template allows an automated mechanism to update the installed cloudSwXtch version. This will deploy the latest version of the cloudSwXtch instead of the one packaged in the AMI, which requires root privileges to trigger the update from the product side. For upgrades, please see **Upgrade cloudSwXtch on AWS** on how to perform an upgrade from the client side. An upgrade from the client side does not require root privileges.

Creating a cloudSwXtch EC2 Instance

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Prerequisites

Before starting, a user must do the following:

1. **Review** **cloudSwXtch System Requirements**.
2. **Ensure** that you already have an AWS account.
3. **Create** a virtual network (VPC). **This *must* be created before deploying a cloudSwXtch.**
4. **Validate you have at least one subnet for your virtual network.** A single subnet can be used for the control and data plane.
5. **Verify a Security Group that allows access to all traffic inside the VPC**. If one is not created, use default when creating a cloudSwXtch.
6. **Create an SSH Key Pair**.

Post-Installation

The following instructions detail how to deploy a cloudSwXtch Multicast instance. However, if a user decides to deploy a **BYOL** instance, they will need to complete **the additional step of licensing their cloudSwXtch.**

If all prerequisites are met, a cloudSwXtch can be created via the Marketplace in any region in approximately 10 minutes. If multi-AZ or multi-region is required then see Mesh for details. The installer will create a CloudFormation Stack to include the following resources:

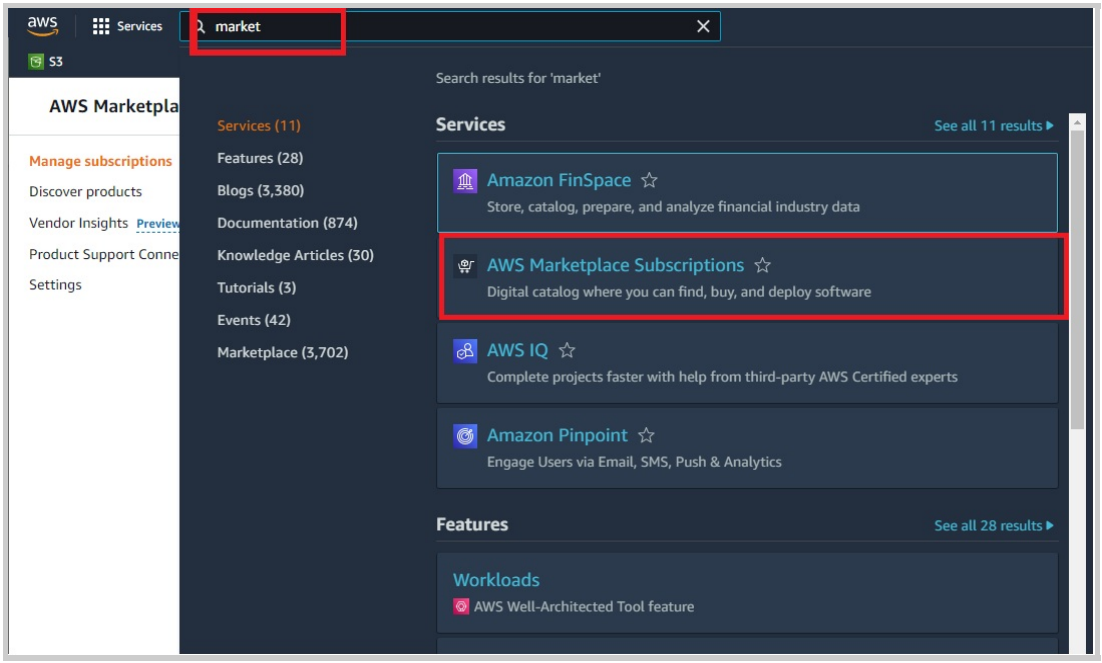
- ControlEni Networking Interface for control data
- DataENI Networking Interface for data such as Multicast
- EC2Instance in Linux for the cloudSwxtch to run on

In order to create a cloudSwXtch, please do the following steps.

1. **Sign** into AWS.

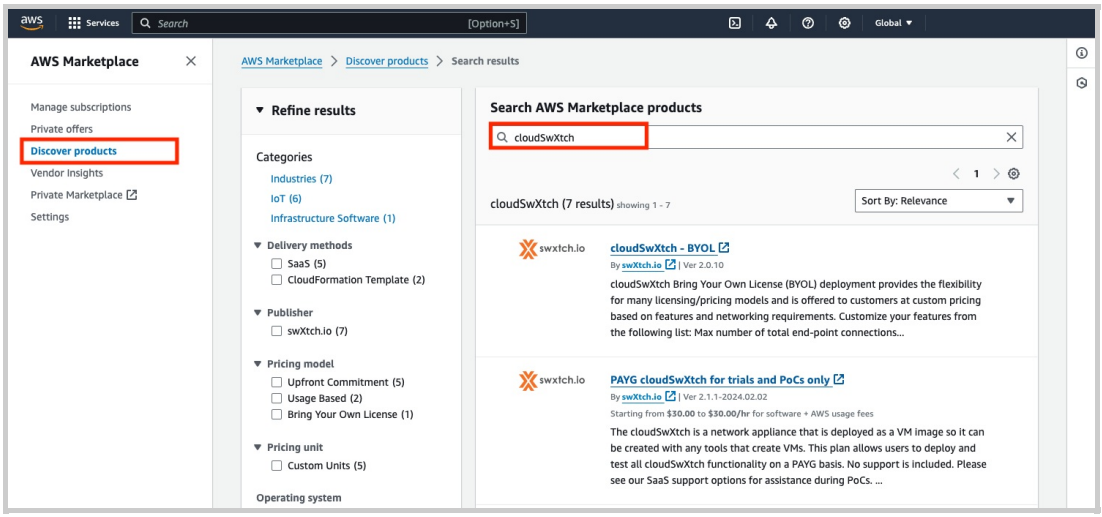
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. From the AWS console, **search** "Market" and **select** "AWS Marketplace Subscriptions" from the search results.



3. **Select** "Discover Products" in the AWS Marketplace menu on the left hand side.

4. **Search** for "cloudSwXtch."



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

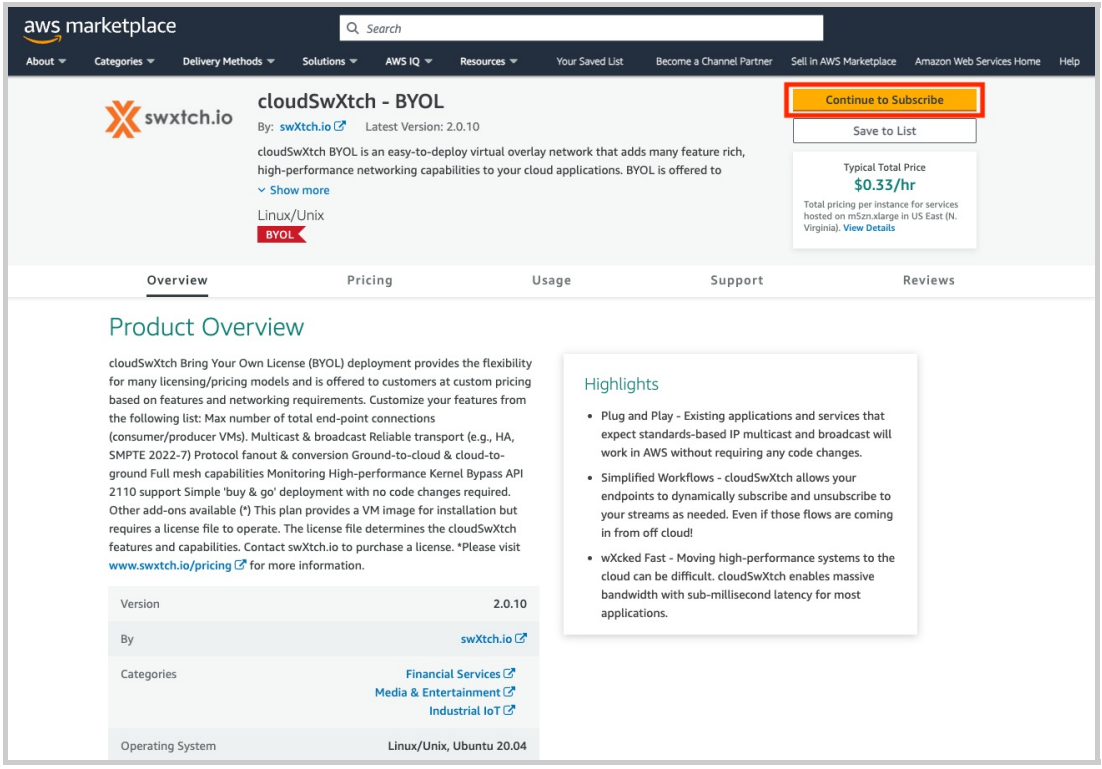
5. **Select** a Tier (cloudSwXtch - Multicast or cloudSwXtch - BYOL) based on your usage requirements and features needed.

    a. Please read the cloudSwXtch System Requirements article for more information regarding cloudSwXtch sizing.

    b. For the purpose of this guide, the next screenshots will be for a **cloudSwXtch Multicast** deployment.

> **Endpoint Connections Limit**
>
> Be mindful of the number of endpoints you connect to your cloudSwXtch after creation. For example, by selecting the "cloudSwXtch Multicast" tier, you will be limited to 10 endpoint connections. If you know you will need more than that, consider deploying a cloudSwXtch BYOL instance.
>
> If you need to increase the number of endpoints, please view the AWS instructions **here**. Note that if your new instance type exceeds the size of your tier, you must contact **support@swxtch.io** to update your license.

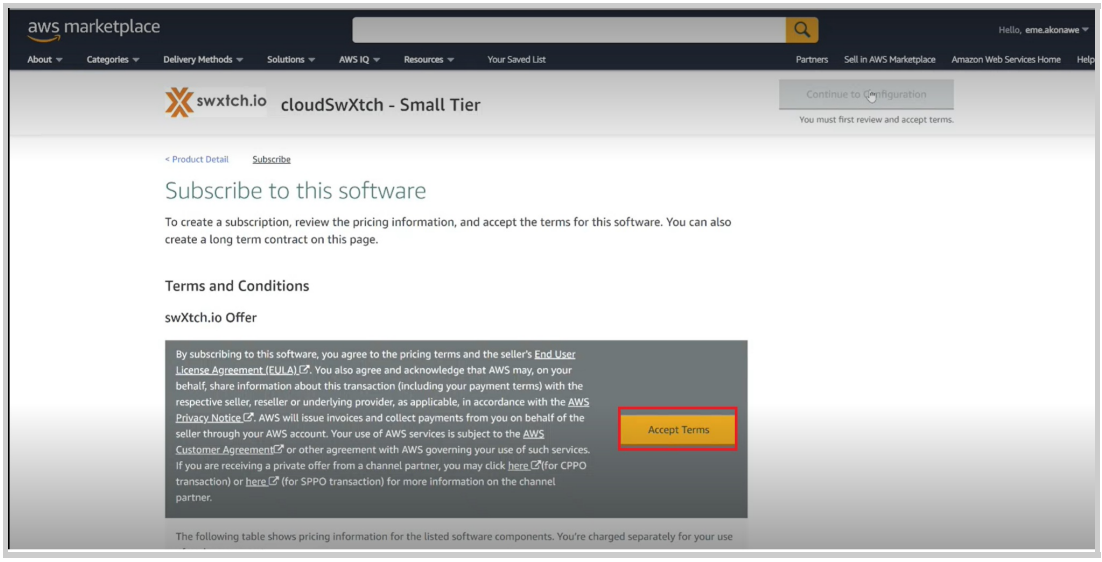
6. **Select** "Continue to Subscribe" after reviewing the product information. Note: The "Typical Total Price" is calculated with the recommended instance size included in the final monthly value and a utilization of 24x7. **Please note**: The cost in "Software Pricing Details" is for the cloudSwXtch and does not include costs for the AWS instance.



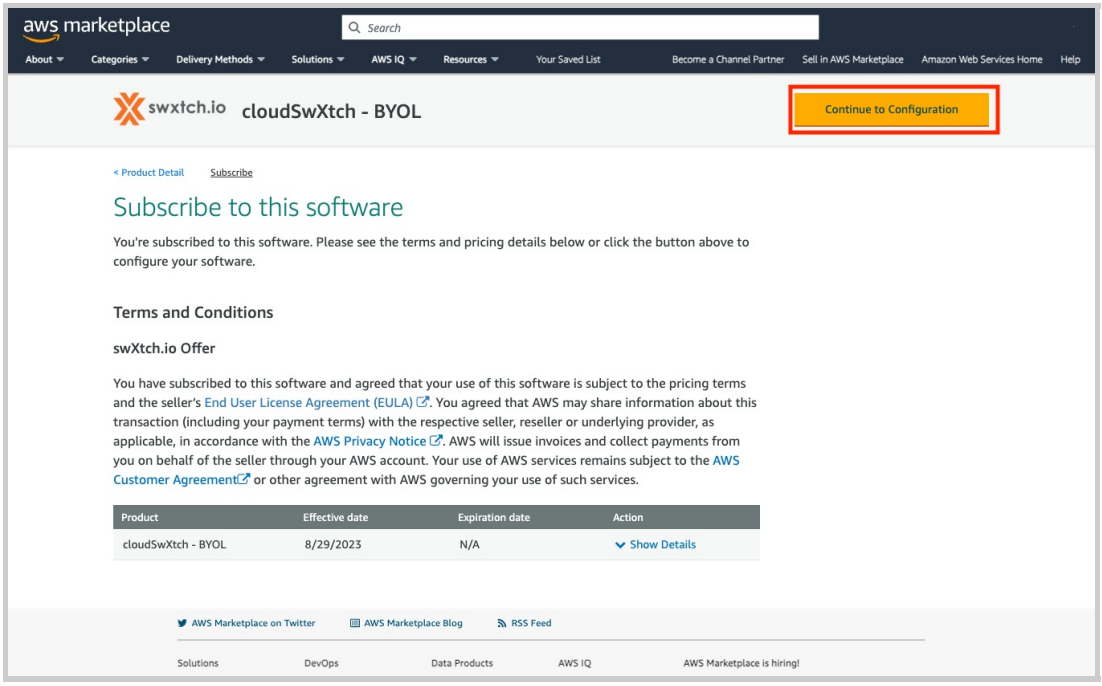
7. **Review** the Terms and Conditions.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

8. **Select** "Accept Terms" if they are acceptable.

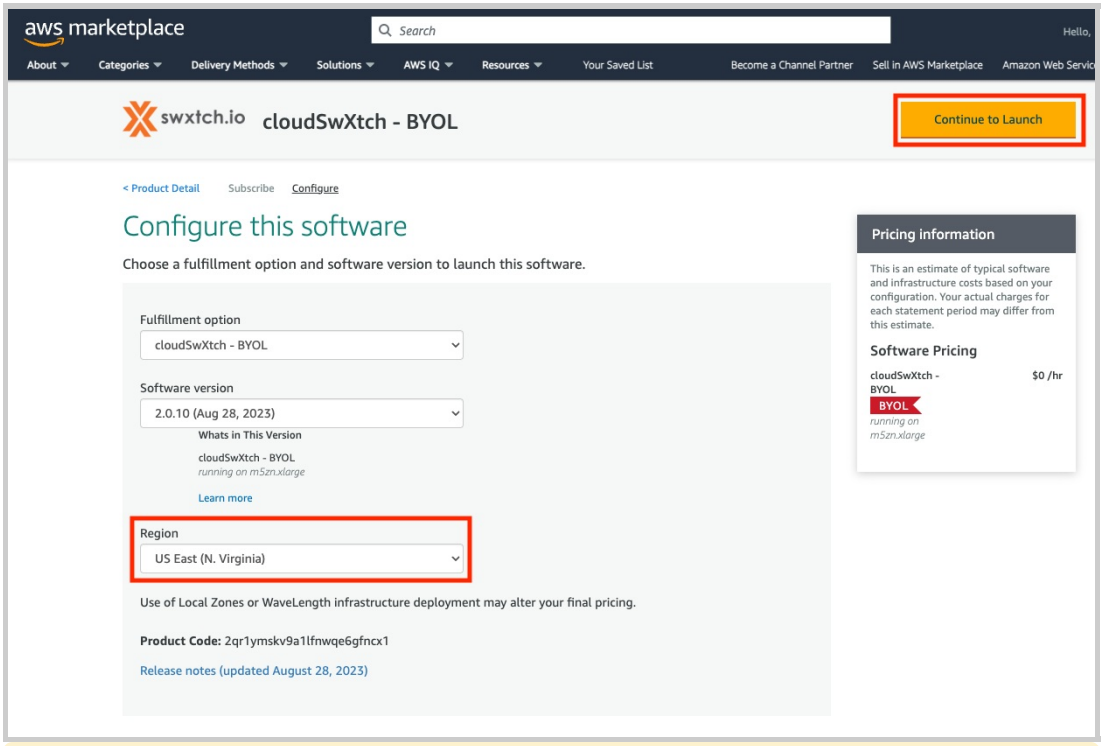


9. **Select** "Continue to Configuration" after reading the subscription and license management.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

10. **Select** the desired "Region" and then **select** “Continue to Launch”. (Note: If you select a region that does not match the region you began with, then it may not work even if selected here.)



INSTANCE TYPES

Note how the cloudSwXtch Marketplace install selects the appropriate VM size in the Fulfillment section based on the cloudSwXtch tier. Please ensure that the instance type matches one of the options below:

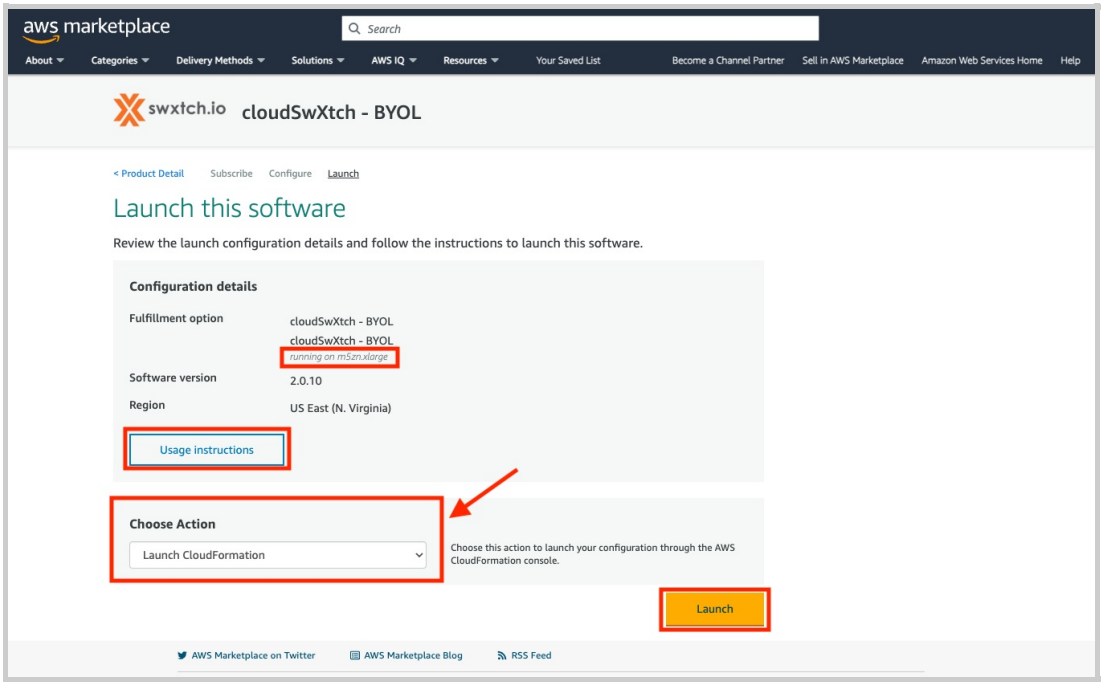
- m5.xlarge
- m5.2xlarge
- m5.4xlarge
- m5.12xlarge
- m5.16xlarge
- m5.24xlarge
- m5zn.xlarge
- m5zn.2xlarge
- m5zn.3xlarge
- m5zn.6xlarge
- m5zn.12xlarge

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

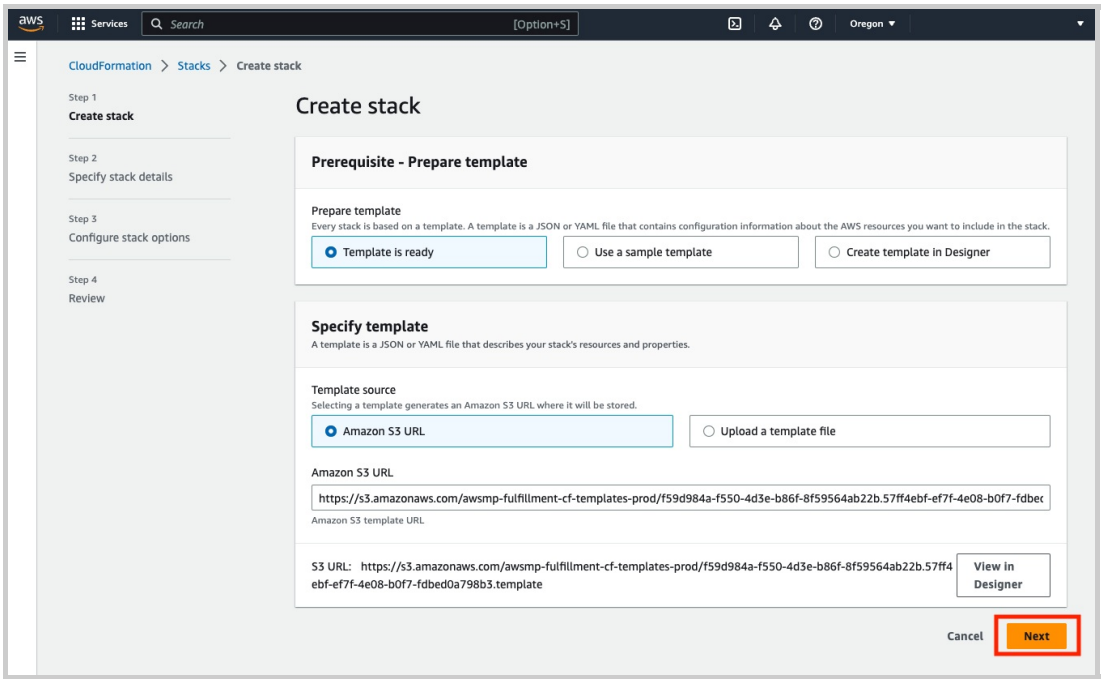
11. **Read** “Usage Instructions” if you desire.

12. **Use** the “Choose Action” dropdown menu and **select** “Launch CloudFormation”.

13. **Click** “Launch”.



14. **Keep** Settings on default on the "Create Stack" page and **select** "Next."



15. **On the Specify stack details page, complete the following**:

    a. Under “Stack name,” **enter** your desired name. Keep in mind that this will be used for everything added to the stack. For example: “resource name,” “security groups,” “EC2 instance name,” etc.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

b. Under “CidrIpForInboundOutboundTraffic,” **use** 0.0.0.0/0 so that you can SSH to the virtual machine from any IP address. You can also pick a more restrictive range if desired.

c. Under “ControlSubnet,” **use** the dropdown to find the control subnet you created (recommended: *ctrl-subnet*).

d. Under “DataSubnet,” **use** the dropdown to find the control subnet you created. Both control and data can share the same subnet.

    i. Alternatively, for better performance, a user can assign a separate subnet for their data subnet.

e. For “InstanceType,” there should be "Fulfillment" data from the earlier step.

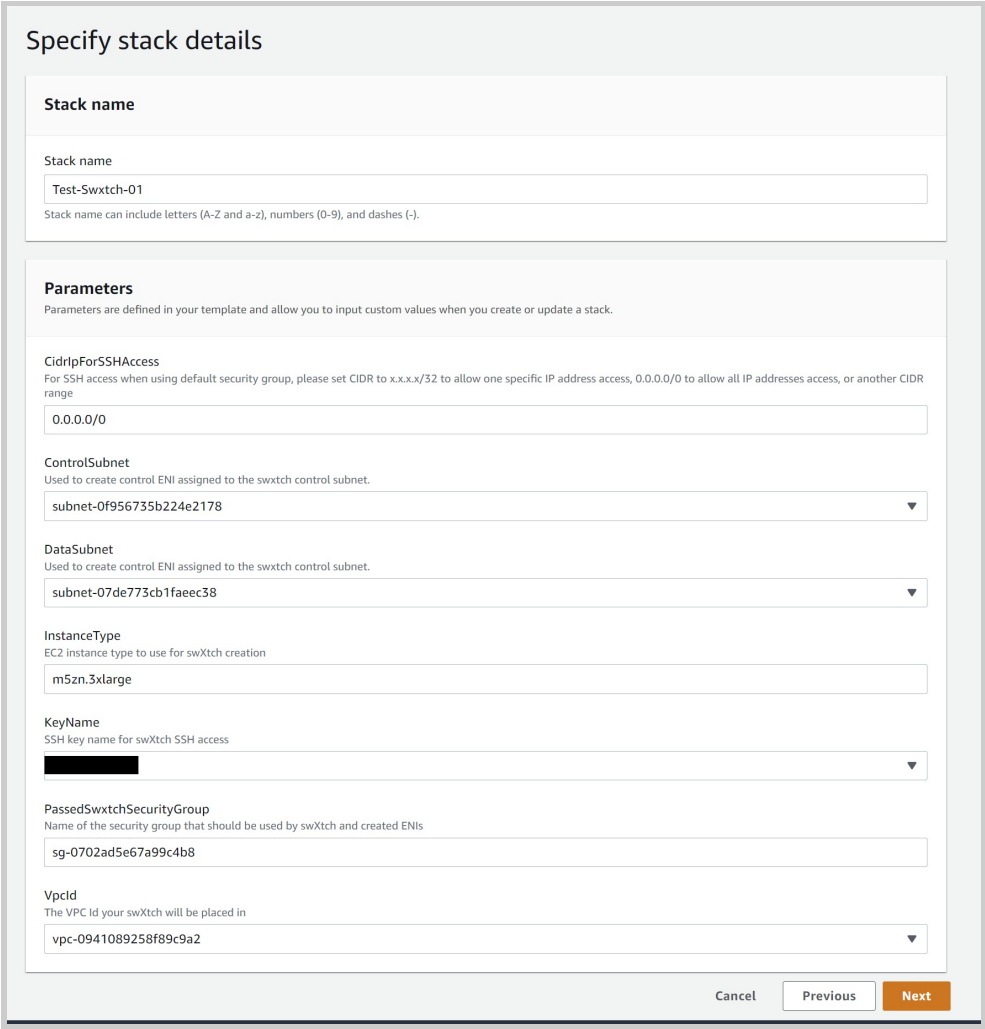
f. Under “KeyName,” **use** the dropdown to find your previously created or imported SSH key.

g. In “PassedSwxtchSecruityGroup,” **use** “default” and one will be created during the installation process. Alternatively, you can enter the ID of an already created security group. It will be something similar to "sg-009273855418af38d."

h. Under “VpcId,” **select** from the dropdown to find the already created VPC id.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

i. **Here is an example of how your template would look like:**



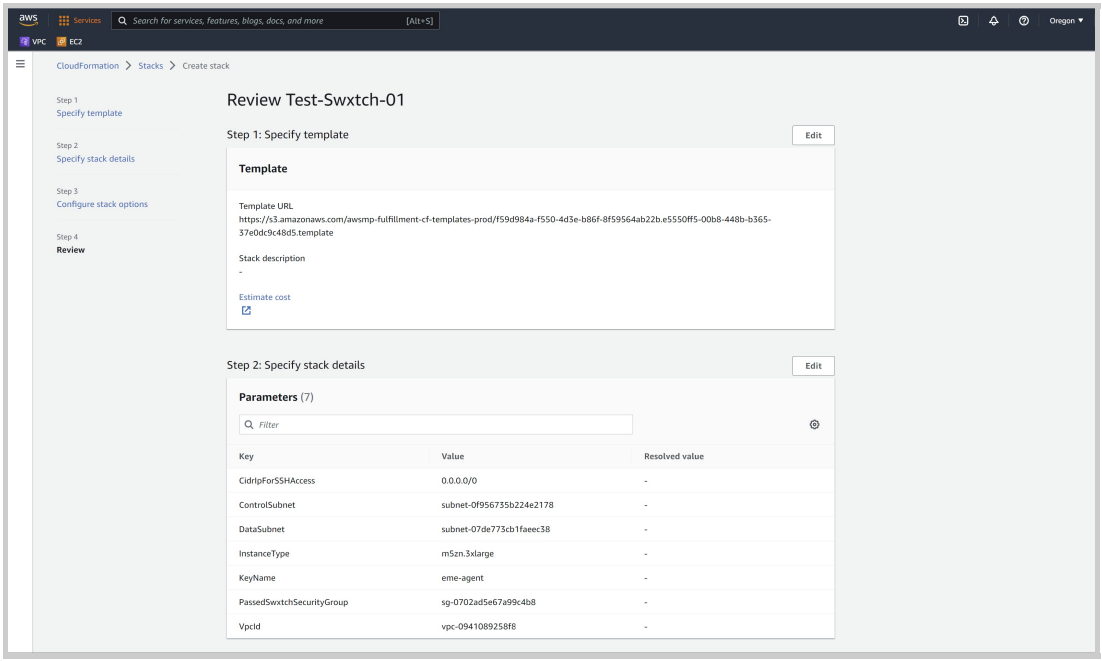
16. **Click** "Next."

17. **The "Configuring stack options" page is completely optional.** You can assign tags for your stack, set additional IAM permissions, stack failure options, etc.

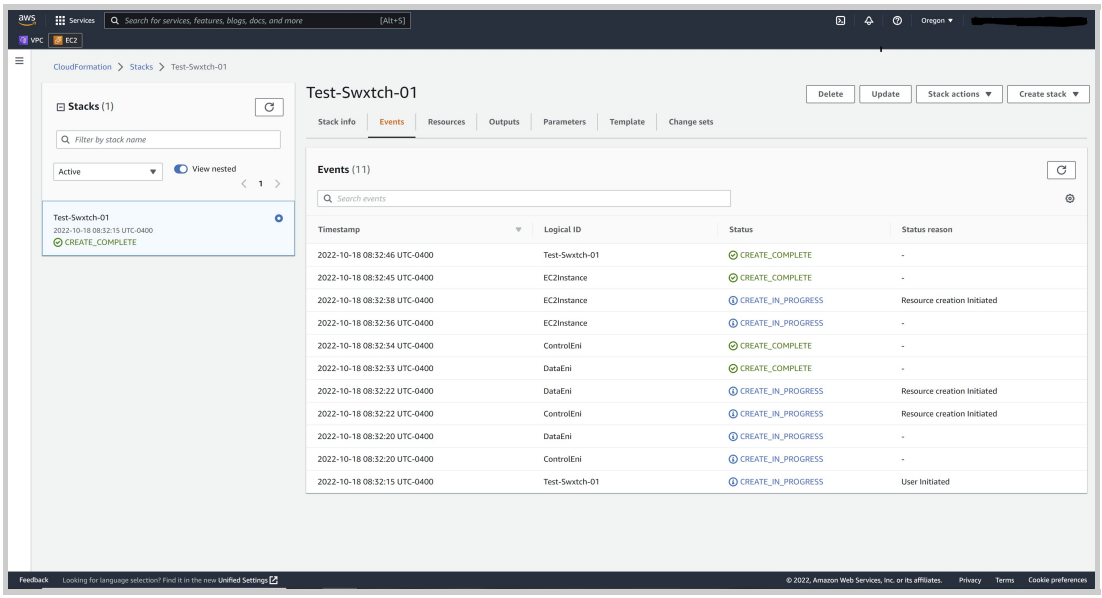
18. **Click** "Next" if you don't need to make any changes.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

19. **Verify** that your parameters are accurate on the final "Review" page. If you need to change anything, **select** "Edit."



20. **Click** "Submit." On the next page, you can view the creation of your stack.



Your EC2 instance has now been created. You can view it on the EC2/Instances list and connect to your cloudSwXtch from there.

21. Once you have connected with SSH to your cloudSwXtch as root user (sudo su), **navigate to the cloudSwXtch directory (cd /swxtch) then run the following command**:

Bash	Copy

```
sudo swxtch/swxtch-top dashboard --swxtch <cloudSwXtch-IP>
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

NOTE

Use the cloudSwXtch-name in place of the IP address if DNS resolution is setup or "localhost."

This will display the cloudSwXtch's swxtch-top dashboard. In "Status," you should see "OK." **This will let you know that your cloudSwXtch has been successfully deployed.** You can review more information regarding swxtch-top in the swxtch-top article.

INSTALLING AN XNIC

If this is a new installation, then each client that is expected to receive or transmit to the cloudSwXtch will need an xNIC installed.

If this is an existing cloudSwXtch replacement, then each client with an xNIC already installed will need to be upgraded to match the current cloudSwXtch version.

You can find more information about xNIC installation, **here**.

Required Step for BYOL: Contact swXtch.io for a license

Users deploying a BYOL instance of cloudSwXtch will need to contact swXtch.io for a license file. For more information, see How to License a cloudSwXtch.

Checking the Health of Your cloudSwXtch Instance

It is important to ensure your AWS system is healthy. AWS provides AWS CloudWatch as a way to check on the health of your system. To check on the cloudSwXtch EC2 instance, read more here.

Upgrading cloudSwXtch on AWS

It is important that your cloudSwXtch instance is up to date. To learn how to upgrade your cloudSwXtch, you can read more here.

Deleting cloudSwXtch on AWS

To learn how to delete your cloudSwXtch, you can read more here.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Deploy cloudSwXtch with Terraform on AWS

WHAT TO EXPECT

In this article, you will learn how to deploy a cloudSwXtch instance on AWS using a Terraform script.

Prerequisites:

For this script to work, you will need to have already provisioned your VPC, Subnets, and SSH Keys. You will plug those parameter values into your AWS/terraform/terraform.tfvars file.

Deploying a Terraform Script

1. Choose what platform you would like to run Terraform on. For this example, the user is on a Linux machine. Download intructions can be found at: terraform.io/downloads

2. Clone the repository using SSH. **Please note**: You will need an SSH key set up with GitHub.

Console	Copy
`$ git clone git@github.com:swxtchio/cloudSwXtch-support`	

Or, alternatively, you can clone with the HTTPS URL: (here on GitHub)

Console	Copy
`$ git clone https://github.com/swxtchio/cloudSwXtch-support.git`	

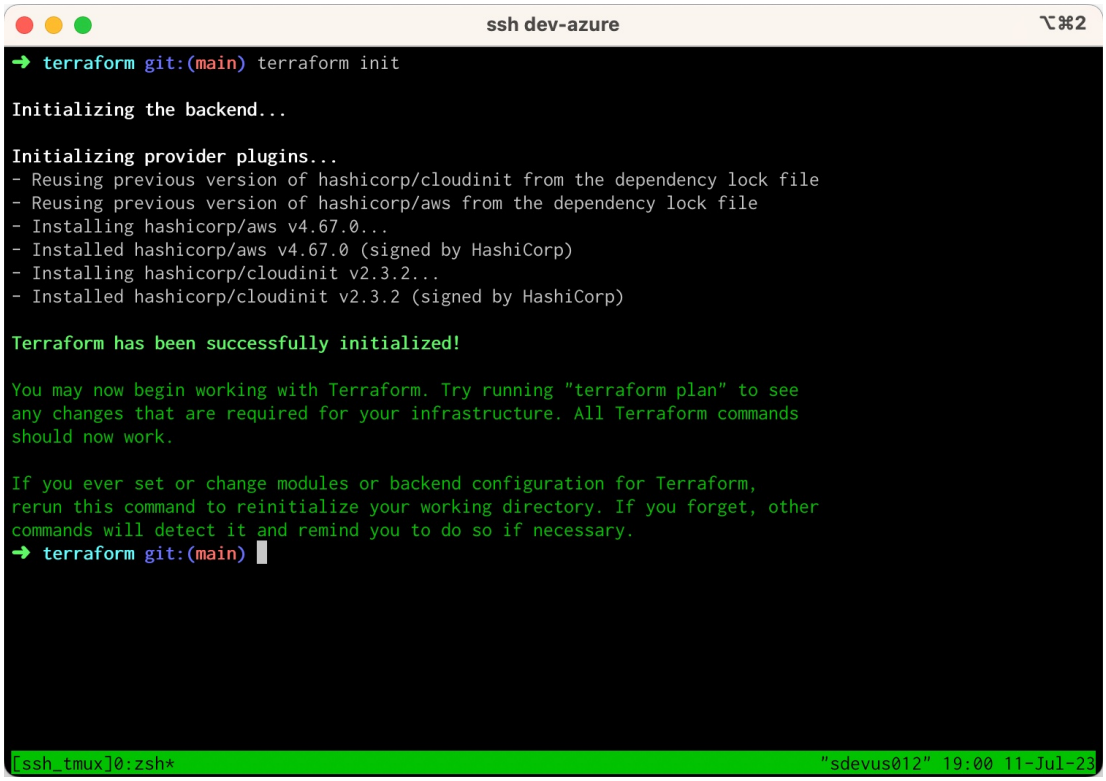
Then:

Console	Copy
`$ cd cloudSwXtch-support/AWS/terraform`	

3. Update the values in the AWS/terraform/terraform.tfvars file to match your existing AWS resources such as: VPC id, Subnet IDs, and SSH Key names.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. Run terraform init inside the AWS/terraform/ directory.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

5. For this step you'll need to have authenticated your AWS credentials inside the console. Or you can pass the credentials with environmental variables. One simple way to do this is using an Access Key (AK). If you don't have one, you can generate your AK in Amazon's IAM->Users->(your user), on the Security Credentials tab, Access Keys. Then, you have to export the following:

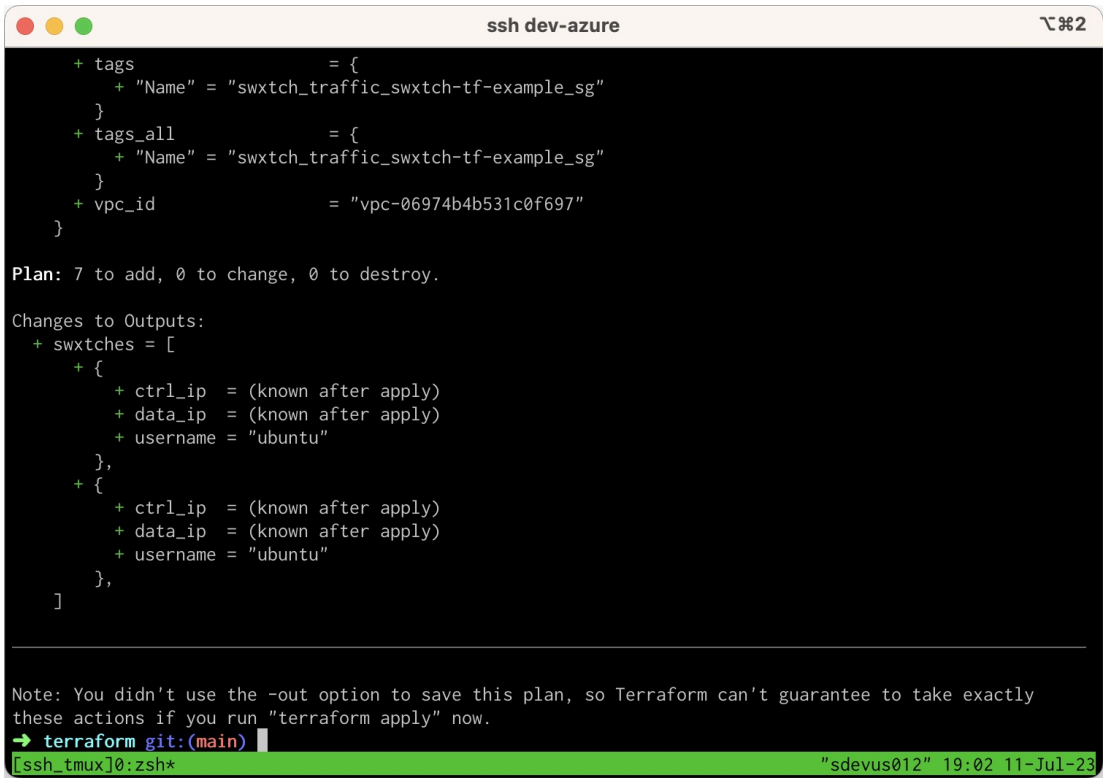
Console	Copy

```
$ export AWS_ACCESS_KEY_ID="anaccesskey"
$ export AWS_SECRET_ACCESS_KEY="asecretkey"
$ export AWS_REGION="us-west-2"
```

Now that Terraform has been initialized and you're authenticated, run this command to evaluate the config and confirm the desired output which will be shown:

Console	Copy

```
$ terraform plan
```



> This is the result of running terraform plan with 2 cloudSwXtches. The provided terraform example creates a security group for all of the deployed resources: two aws_network_interface for each of the cloudSwXtches and an aws_instance for each cloudSwXtch.

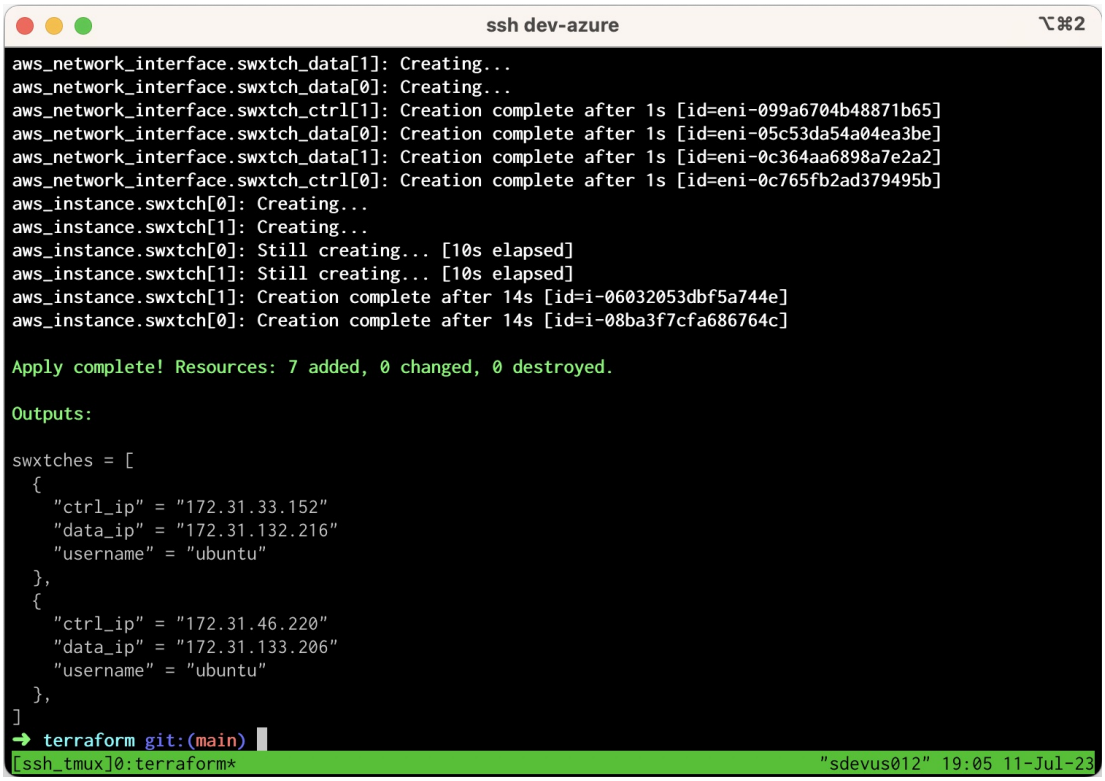
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

6. Run the Terraform apply command (followed by “yes” when prompted) to approve the action.

Console	Copy

```
terraform apply
yes
```

7. Once the resources have been applied successfully, you should see an output similar to this:



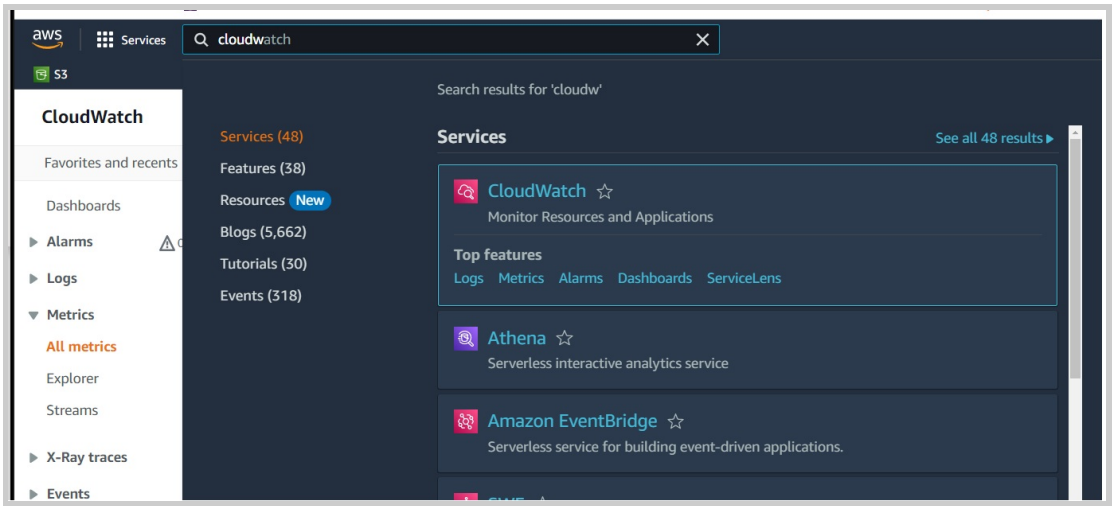
You can view the resources created from your AWS portal as confirmation of a successful deployment.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Check Health of cloudSwXtch Instance on AWS

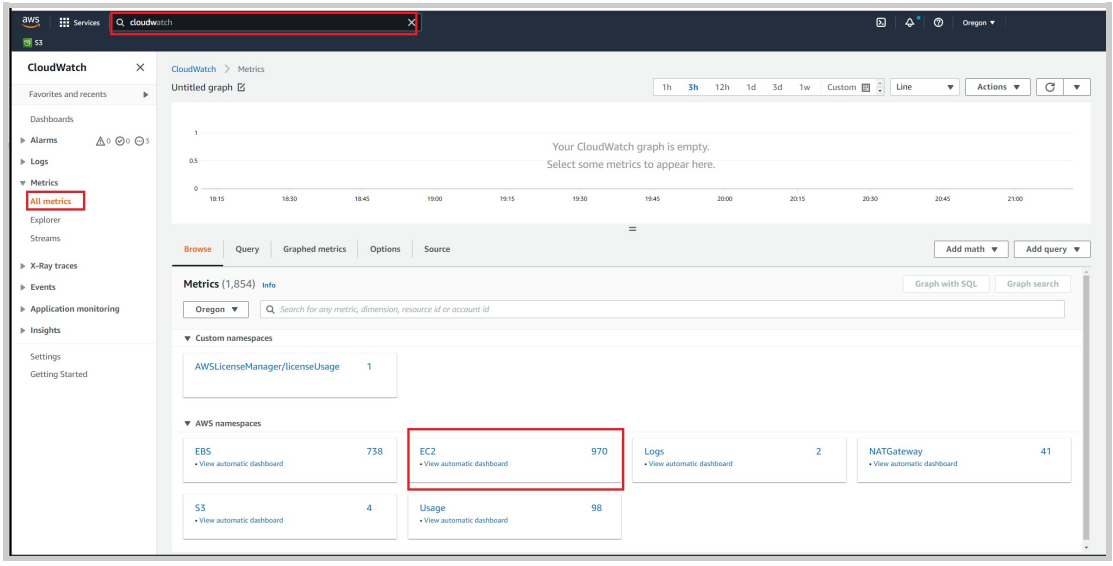
It is important to ensure your AWS system is healthy. AWS provides AWS CloudWatch as a way to check on the health of your system. To check on the cloudSwXtch EC2 instance:

1. **Search** for "CloudWatch" in the AWS Search bar.



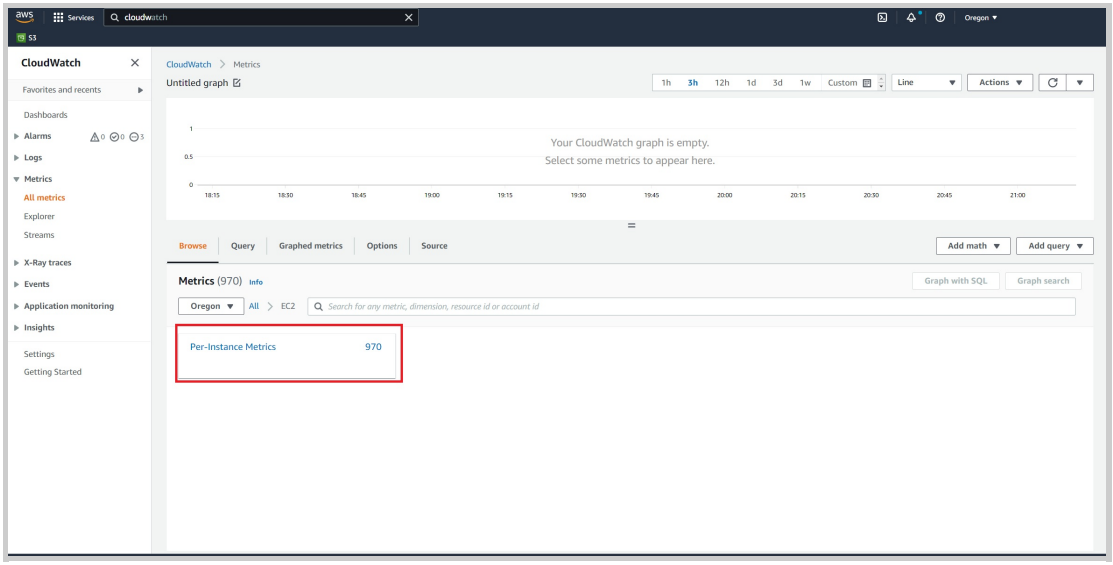
2. **Select** "All Metrics" on the left tree menu under "Metrics."

3. **Select** "EC2."



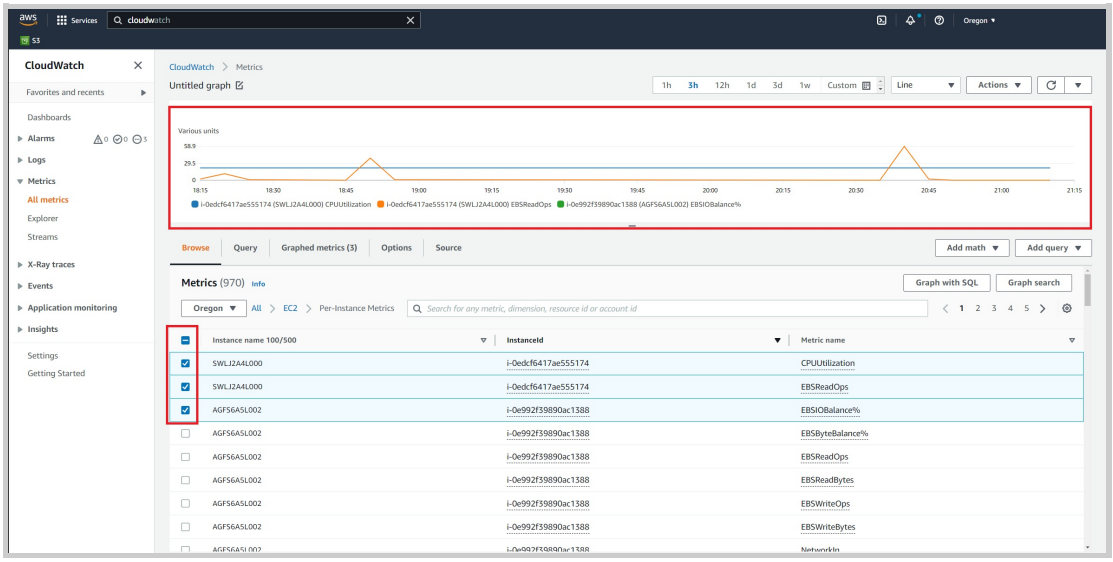
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. **Select** "Per-Instance Metrics."



5. **Sort** as desired. Instance ID works well.

6. **View** data in graph.



WARNING

A cloudSwXtch instance will consume CPU even when the connected agents are not producing/consuming data. This is because there are several vCPUs configured to constantly watch the interfaces.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Delete cloudSwXtch on AWS

WHAT TO EXPECT

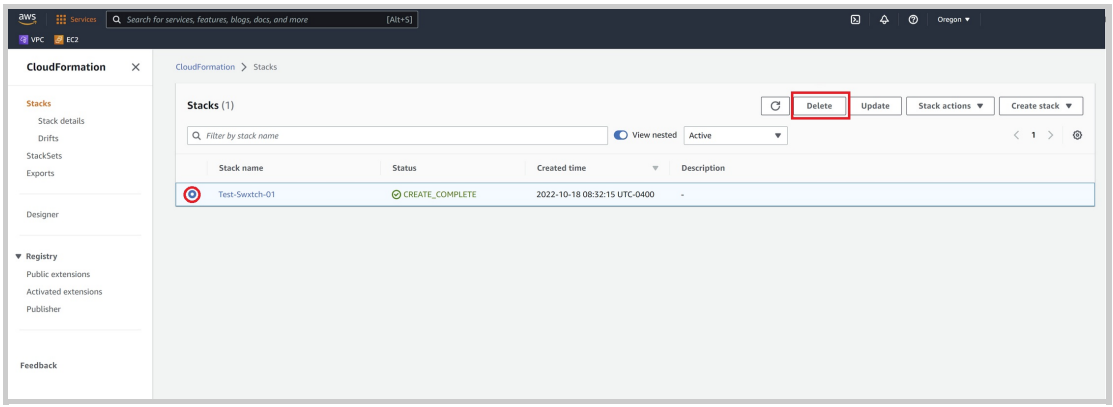
In this section, users will learn how to delete cloudSwXtch from their AWS environment.

Prior to deleting a cloudSwXtch, it is advised to uninstall any xNICs using it. See xNIC Installation.

It is important to note that since your cloudSwXtch was created using a Stack, you do not want to just delete the EC2 instance by itself. Rather, you will want to delete the Stack as whole, which will also delete all associated resources as well.

To delete a cloudSwXtch:

1. **Navigate** to your cloud stack: “Cloud Formation → Stacks”



2. **Select** the stack you want to delete.

3. **Click** "Delete" and then confirm on the popup window.

4. **Refresh the page after a minute or so to confirm the stack has been deleted.**

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

cloudSwXtch on Azure

Pre-installation Steps

There are three methods that a cloudSwXtch instance can be deployed using the Azure Portal: via template, via Terraform, and via the Market Place.

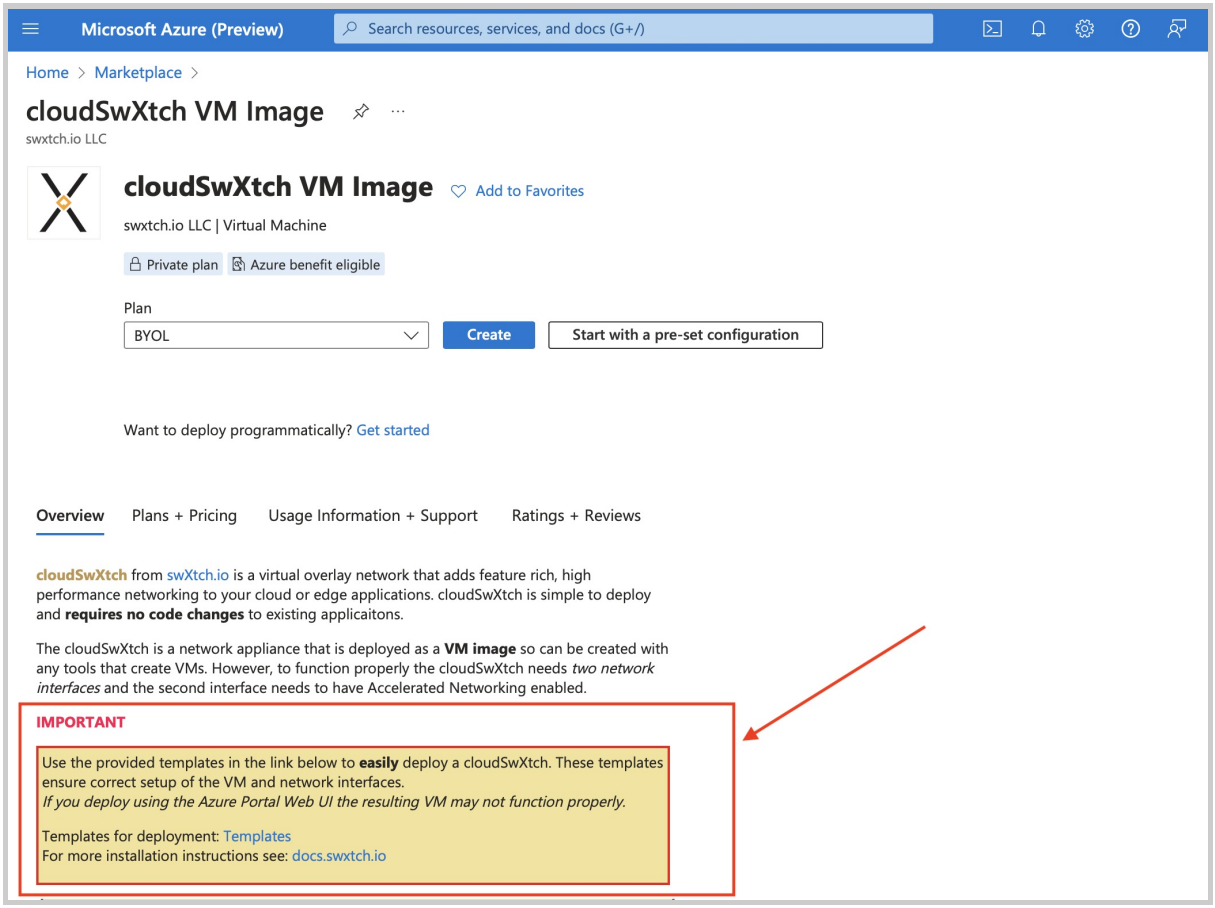
Out of those three options, the **preferred method is via template** as it will create the two subnets needed for a cloudSwXtch to operate. In addition, the Network Interface will have "Accelerated Networking" enabled.

Template Method (PREFERRED):

1. Review system requirements
2. Validate subnets on Azure
3. Create Azure cloudSwXtch Template
4. Install cloudSwXtch on Azure

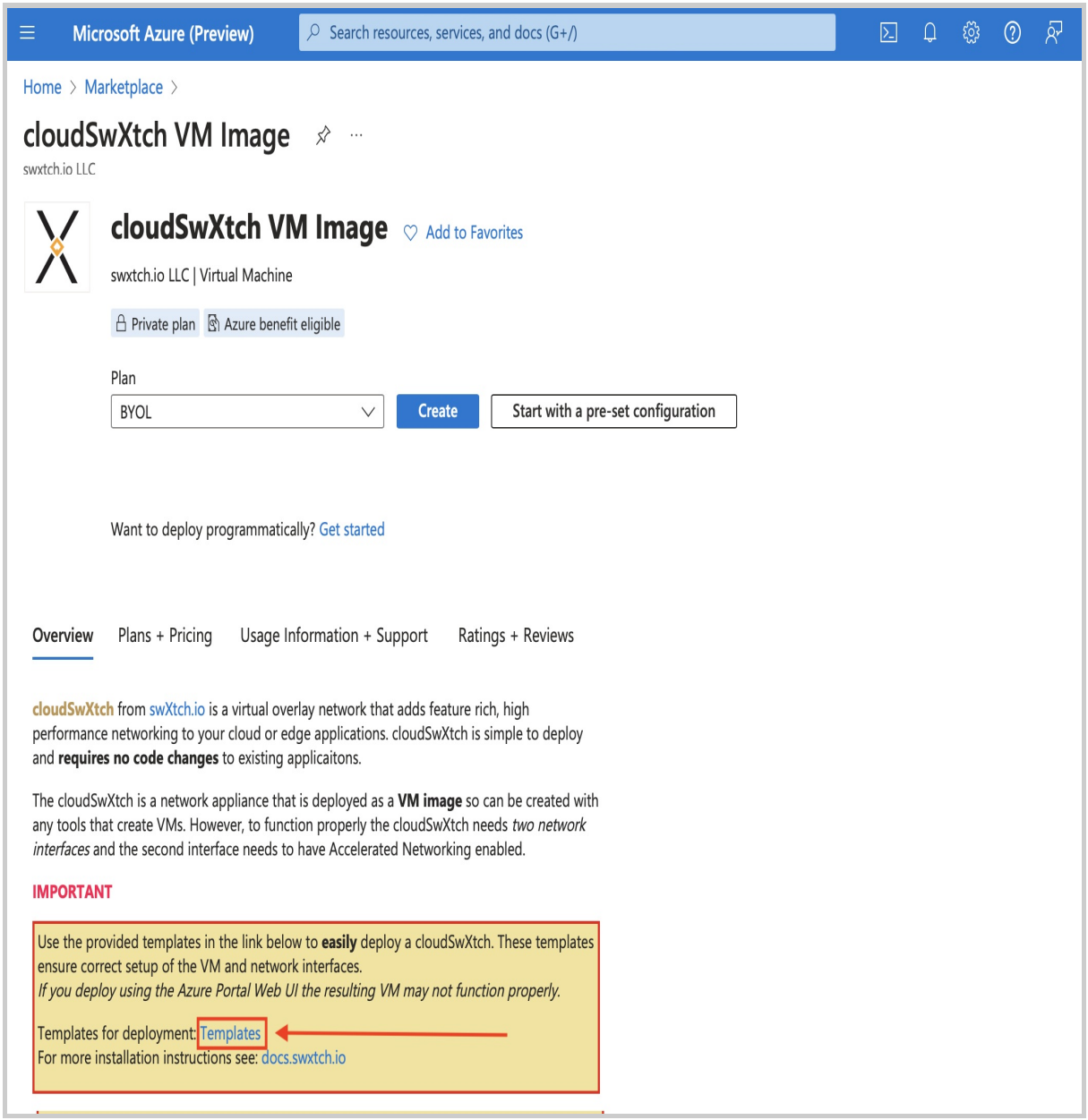
The template method is also mentioned in the Market Place cloudSwXtch installation as highlighted below:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



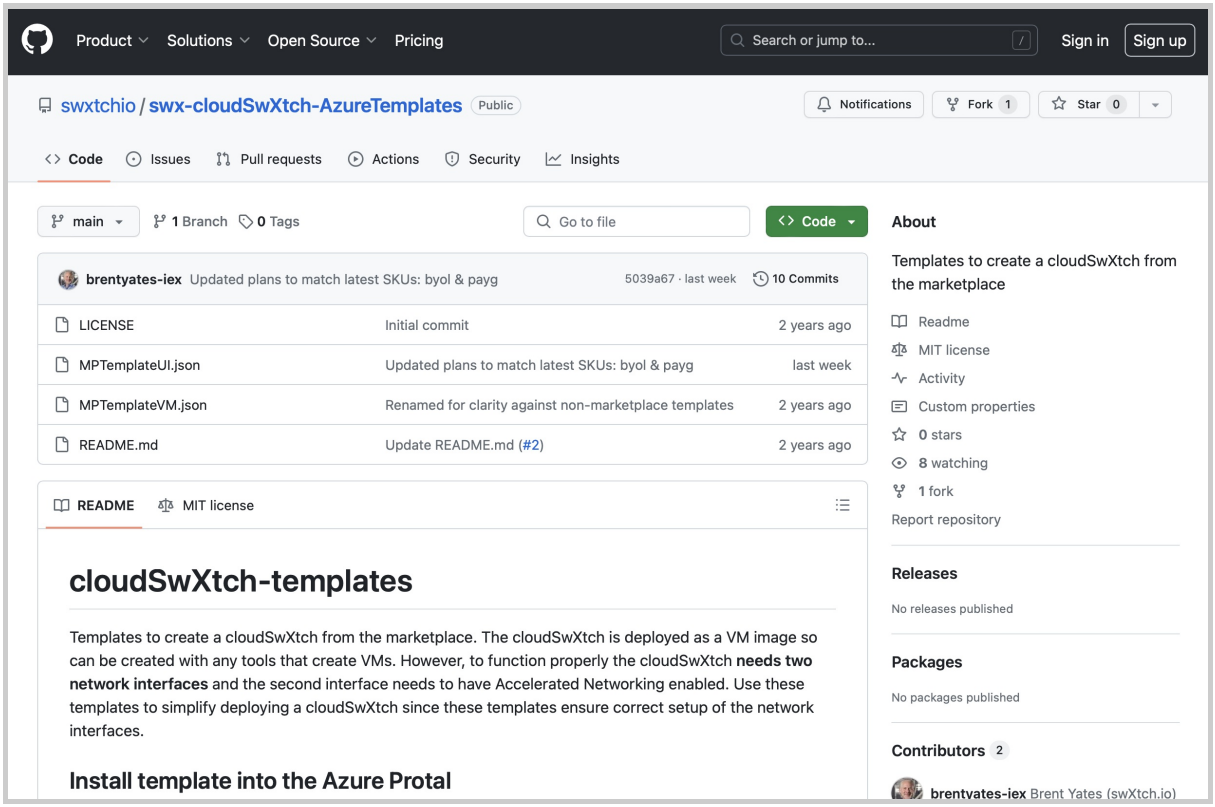
Clicking on the “Templates” hyperlink shows more information about the template creation method.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



Selecting this link will open the page below:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



Alternative Install Methods

Market Place

While a user can create a cloudSwXtch via the Market Place, it will require additional work in terms of maintenance. For example, the cloudSwXtch would have to be updated to add a second NIC and then have accelerated networking manually enabled. With the template method, users can bypass all this.

If you still wish to use the Market Place method, you can find more information here.

Terraform

If you wish to deploy cloudSwXtch via Terraform, you can find more information here.

Air-Gapped

For closed environments, users can follow the Azure Air-Gapped installation instructions here.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Validate Subnets on Azure

WHAT TO EXPECT

A virtual network must be created before deploying a cloudSwXtch EC2 instance.

- It must contain at least one subnet that's used for both the control and data plane communication.
    - It is recommended that it is **private facing** and **does not auto-assign public IPs**.
    - This single subnet will be used for xNIC installation.

The subnets will also be used xNIC installations.

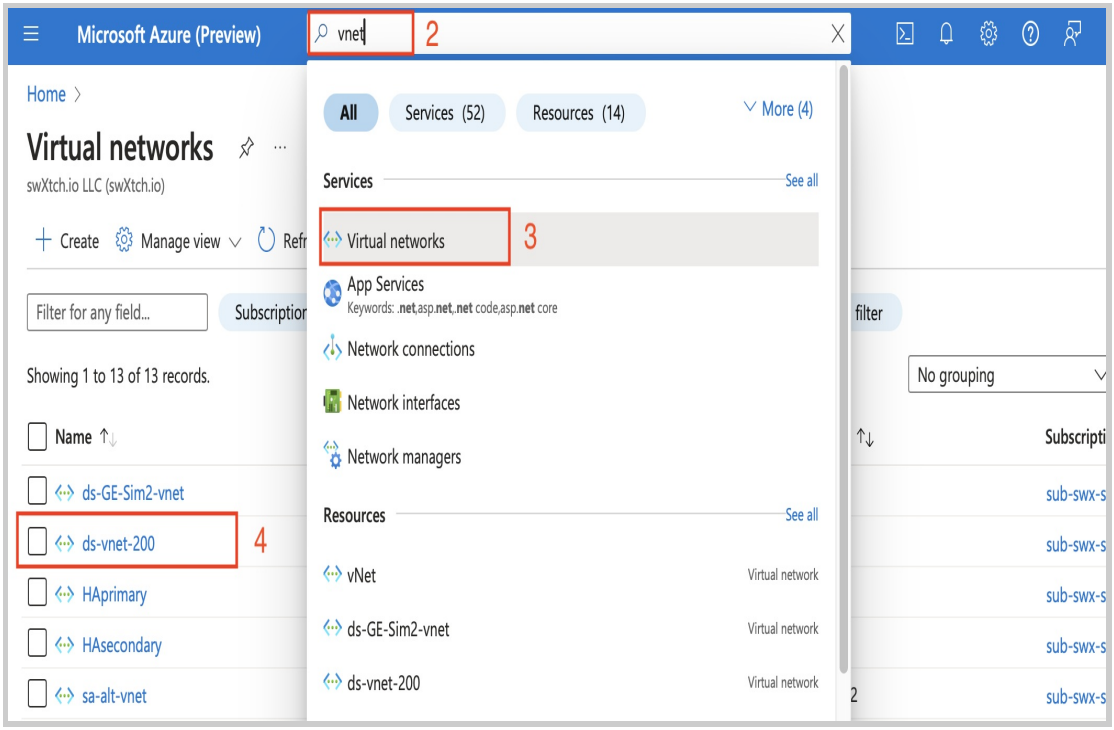
In this section, users will learn how to validate whether a subnet exists to be used as both the control and the data plane for their virtual network. This is in preparation for cloudSwXtch installation on Azure. We will also walk through an alternative method of using 2 subnets, separating the control and data plane.

To validate:

1. **Go** to the Azure Console.
2. **Search** for "vnet".
3. **Select** Virtual Networks.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

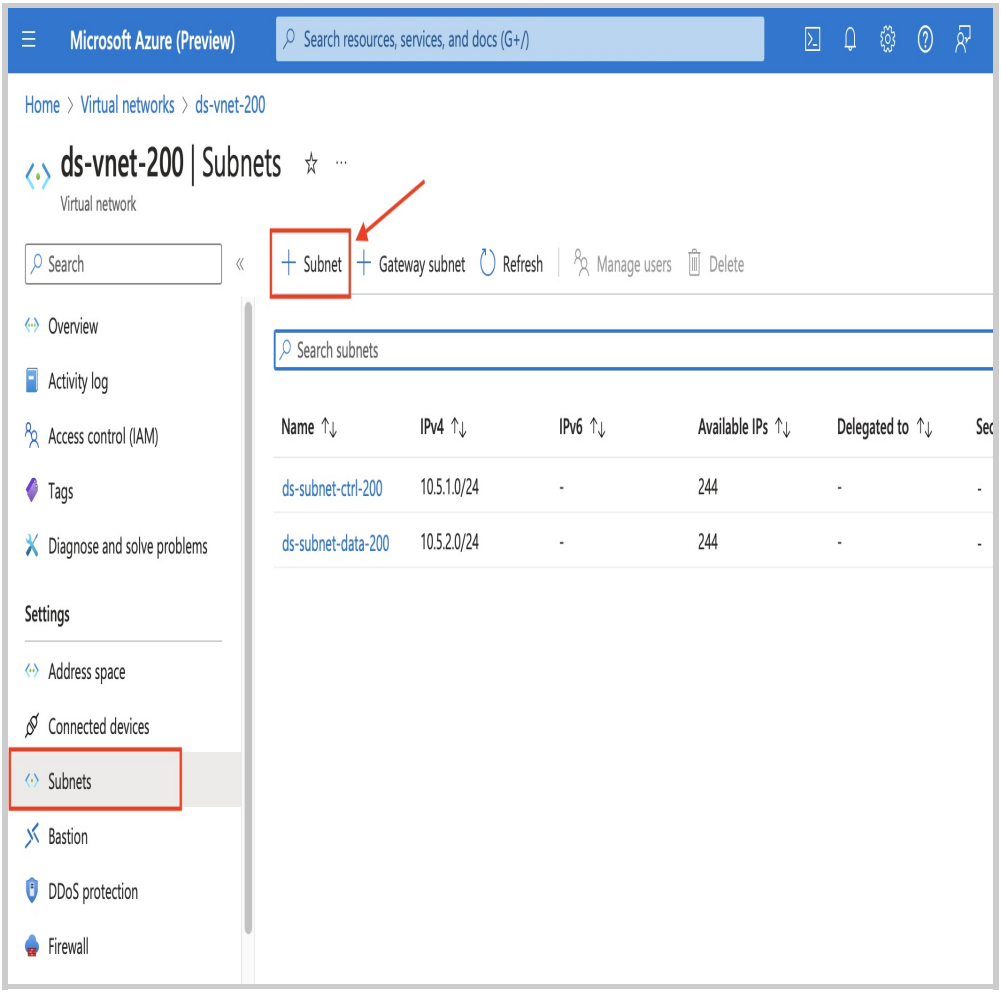
4. **Select** the vnet to be used for cloudSwXtch.



5. **If using a single-subnet, name** the subnet as "ctl." **If using two subnets**, name the second one "data" to distinguish between them when creating an VM instance. Using two subnets will require them to be in the same Region (for example, East US). This enables a single VM instance to have two NICs connected to both subnets at the same time.

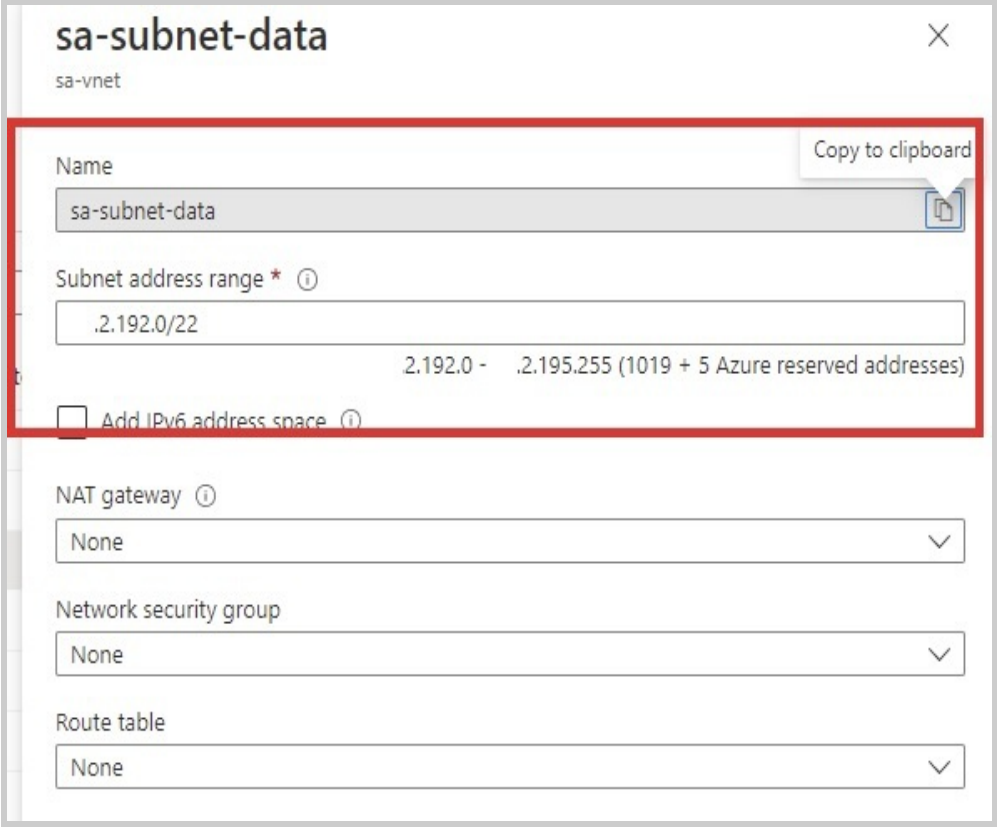
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

a. **In the event that the second subnet does not exist,** create it by selecting "+ Subnet."



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

b. **Enter** data as shown below making sure the subnet in the same VNET and Availability zone as shown below:



Endpoint Connections Limit

Please be mindful of the number of endpoints (virtual machines) you are allowed to connect to your cloudSwXtch after creation. For more information about sizing, please see **cloudSwXtch System Requirements**.

NEXT STEP: Creating an Azure cloudSwXtch Template

After validating the subnets on Azure, continue on to the **Create an Azure cloudSwXtch Template** guide. This is in preparation for **installing cloudSwXtch on Azure**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Create an Azure cloudSwXtch Template

WHAT TO EXPECT

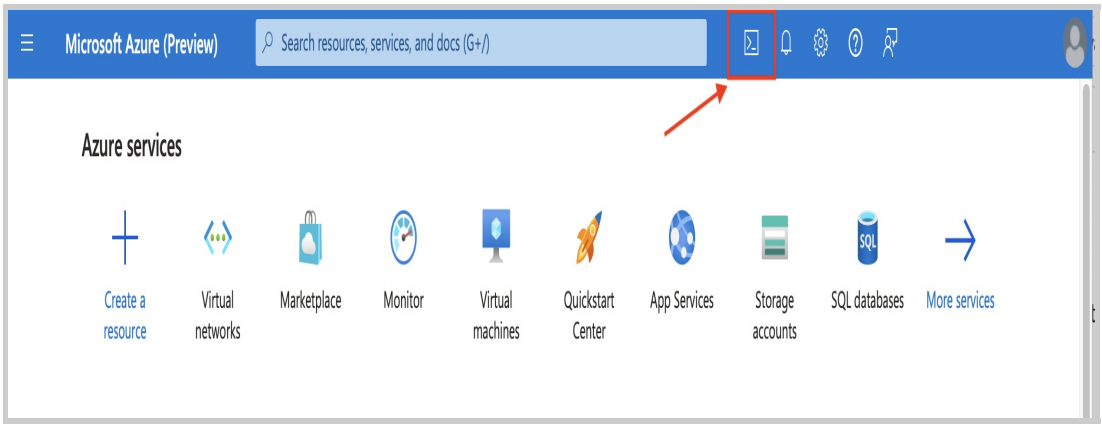
The easiest way to deploy a cloudSwXtch instance in your Azure environment is through the template method. **The following process is a one-time task per subscription.**

This section will walk you through the template creation process in preparation for Azure cloudSwXtch installation.

Template Creation

A cloudSwXtch template can be created by using the Azure Portal. This template will be used to create a cloudSwXtch "Creating cloudSwXtch via Template method". The template is not used during creation of a cloudSwXtch via the Market Place. The creation of the Template is a one-time task per subscription

1. Log into the Azure Portal. You will need permissions to create and manage virtual machines.
    - virtual-machine-contributor
2. **Open Cloud Shell.**



If you need help setting up your Azure cloud-shell, use the link below for setup instructions.
azure cloud-shell quick start

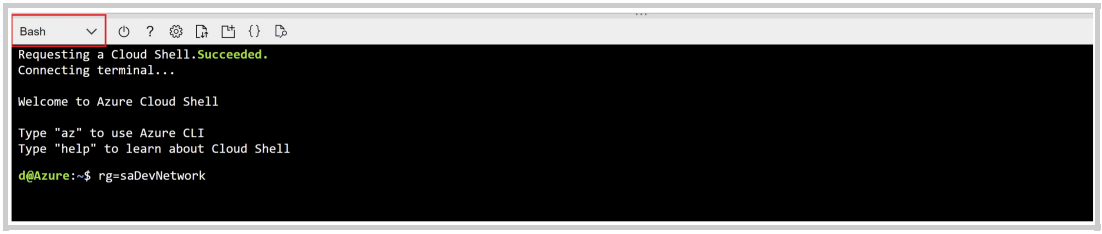
3. **Make sure** you are running your cloud shell terminal in Bash mode.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. **Enter** in the following command to get to the proper resource group:

```bash
rg="<your-rg-here>"
```

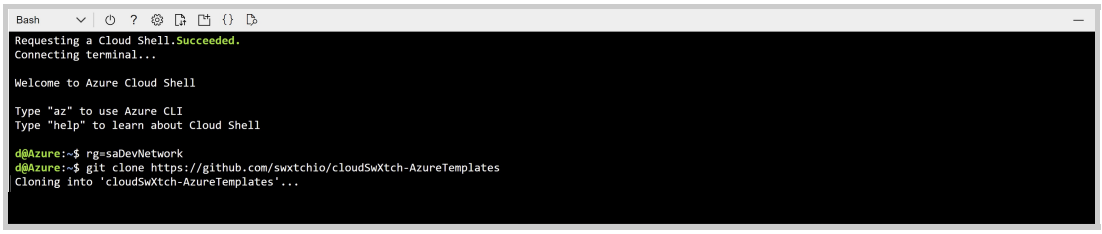
Example below:



5. **Enter** in the following command to clone the "cloudSwXth-AzureTemplates":

```bash
git clone https://github.com/swxtchio/cloudSwXtch-AzureTemplates
```

Example below:



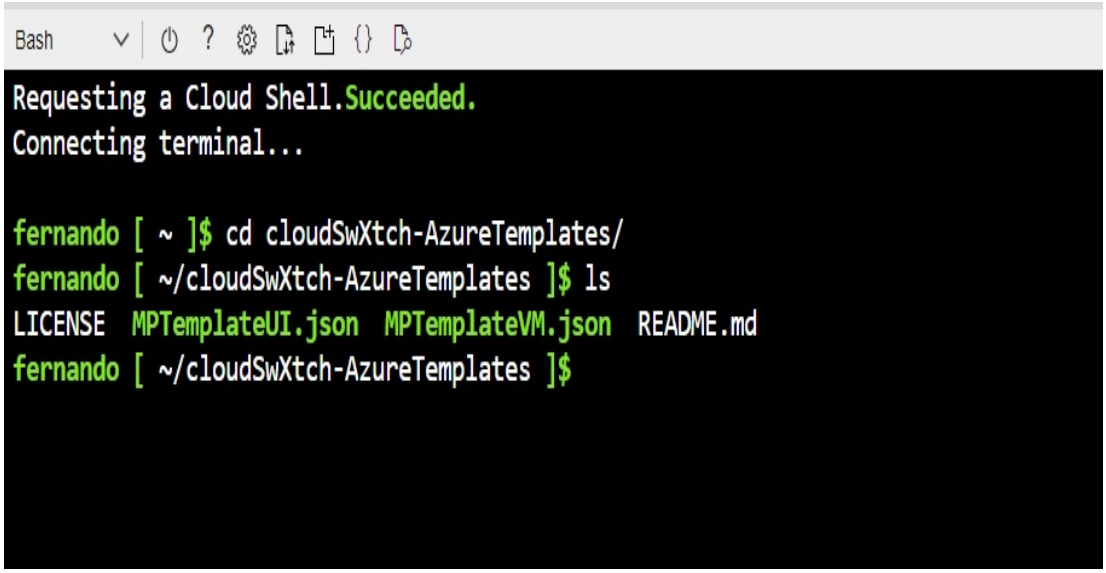
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

6. **Change** directory (cd) to "cloudSwXtch-AzureTemplates".

Bash	Copy

```
cd cloudSwXtch-AzureTemplates
```

If desired, use the "ls" command to see what is in the directory. Example below:

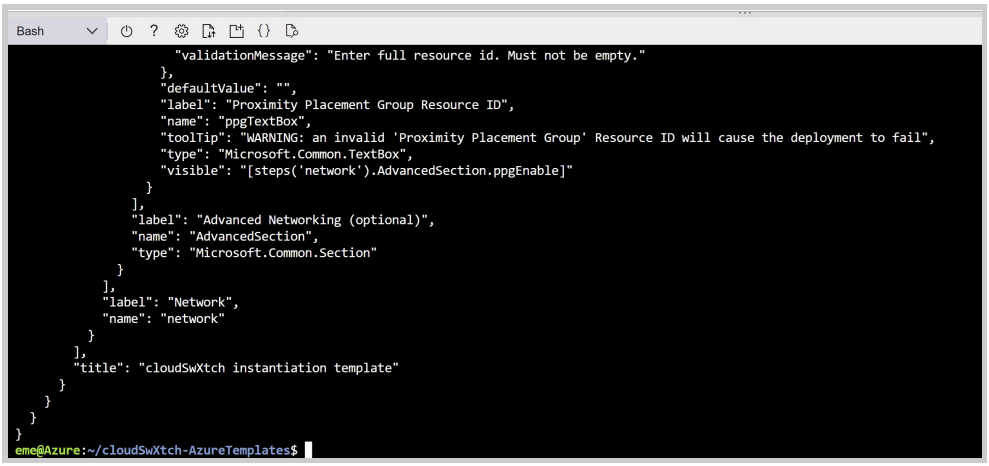


7. **Create** "cloudSwxtch-from-mp-image" using the following command:

Bash	Copy

```
az ts create -n cloudSwxtch-from-mp-image -g $rg -v 1 -f
MPTemplateVM.json --ui-form-definition MPTemplateUI.json
```

a. The output should look like the below screenshot:



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

NEXT STEP: Azure cloudSwXtch Installation

After completing template creation and **validating subnets**, continue on to the main **Azure cloudSwXtch Installation guide**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install cloudSwXtch on Azure

WHAT TO EXPECT

Installation of a cloudSwXtch instance consists of two parts: the cloudSwXtch and the xNIC software. The cloudSwXtch is instantiated once while the xNIC is installed on each VM that is part of the cloudSwXtch network.
In this section, users will learn how to install cloudSwXtch for their Azure environment through the template method.

Please note: This is the preferred method of installation. However, alternatively, you can do a manual install via the Marketplace. For more information on this method, please see the **Install cloudSwXtch via Market Place** guide.

NOTE:

Access to https://services.swxtch.io should be enabled for marketplace installation of the cloudSwXtch. For closed environments, swXtch.io offers a BYOL model to allow installation and operation for highly secure deployments. Please contact support@swxtch.io for more details.

Deploying a cloudSwXtch instance

Prerequisites

Before starting, a user must do the following:

1. **Review cloudSwXtch System Requirements**.
2. **Validate that there are two Subnets**: A virtual network must be created before creating a cloudSwXtch instance. This must contain two subnets, known as the ctrl- and data-subnet. In addition, the data subnet must have the "Network Acceleration" feature enabled.
3. **Create an Azure cloudSwXtch Template**: Creating a template will allows users to follow the easiest method for cloudSwXtch deployment detailed below.
4. **Make sure that your Azure subscription has the quota and access privileges to create the virtual machine instance used to run the cloudSwXtch.** Your instance will fail if you do not have the quota for the selected machine size.

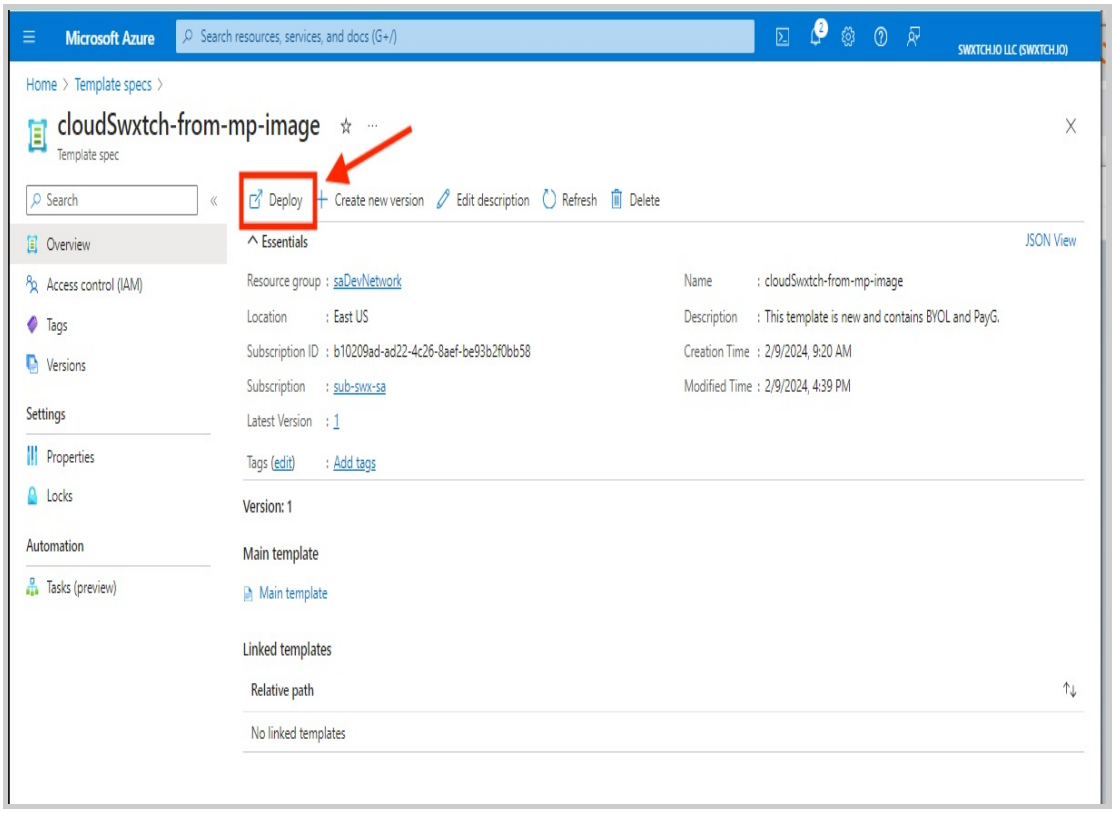
1. Log into the **Azure** Portal.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

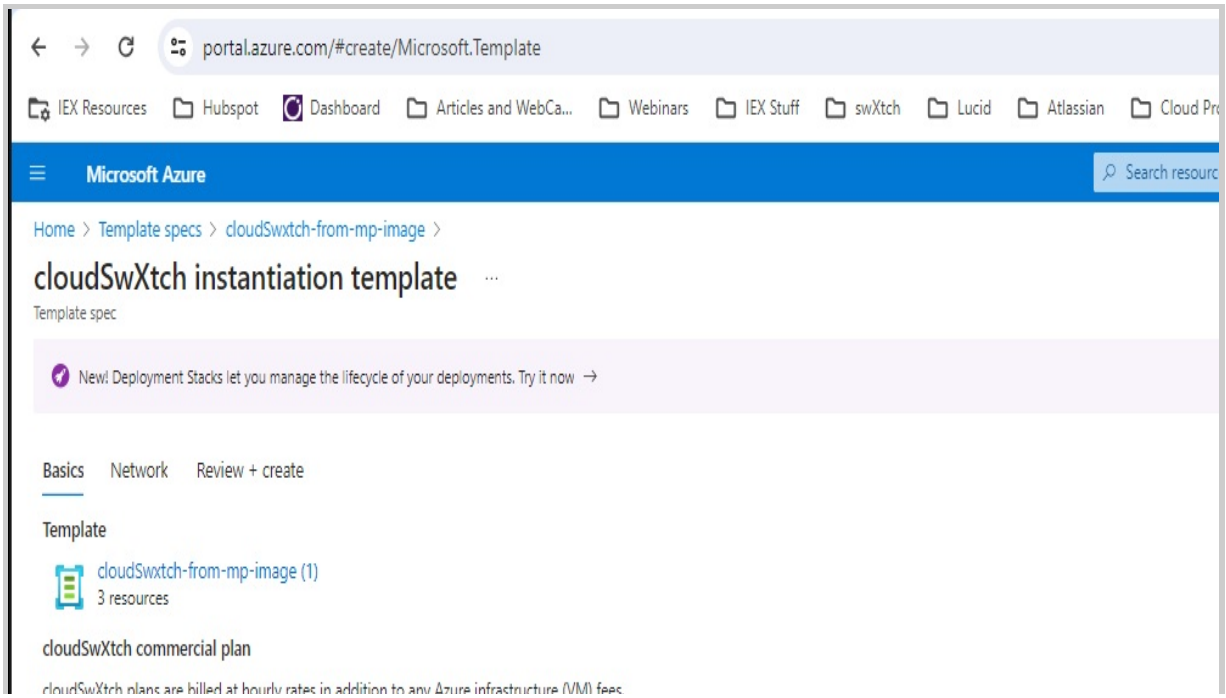
2. Find the template by using the **Search resource, services, and docs (G+/)** bar and enter **cloudSwxtch-from-mp-image** in the search. This will take to directly to the template.

3. Select the **template**.

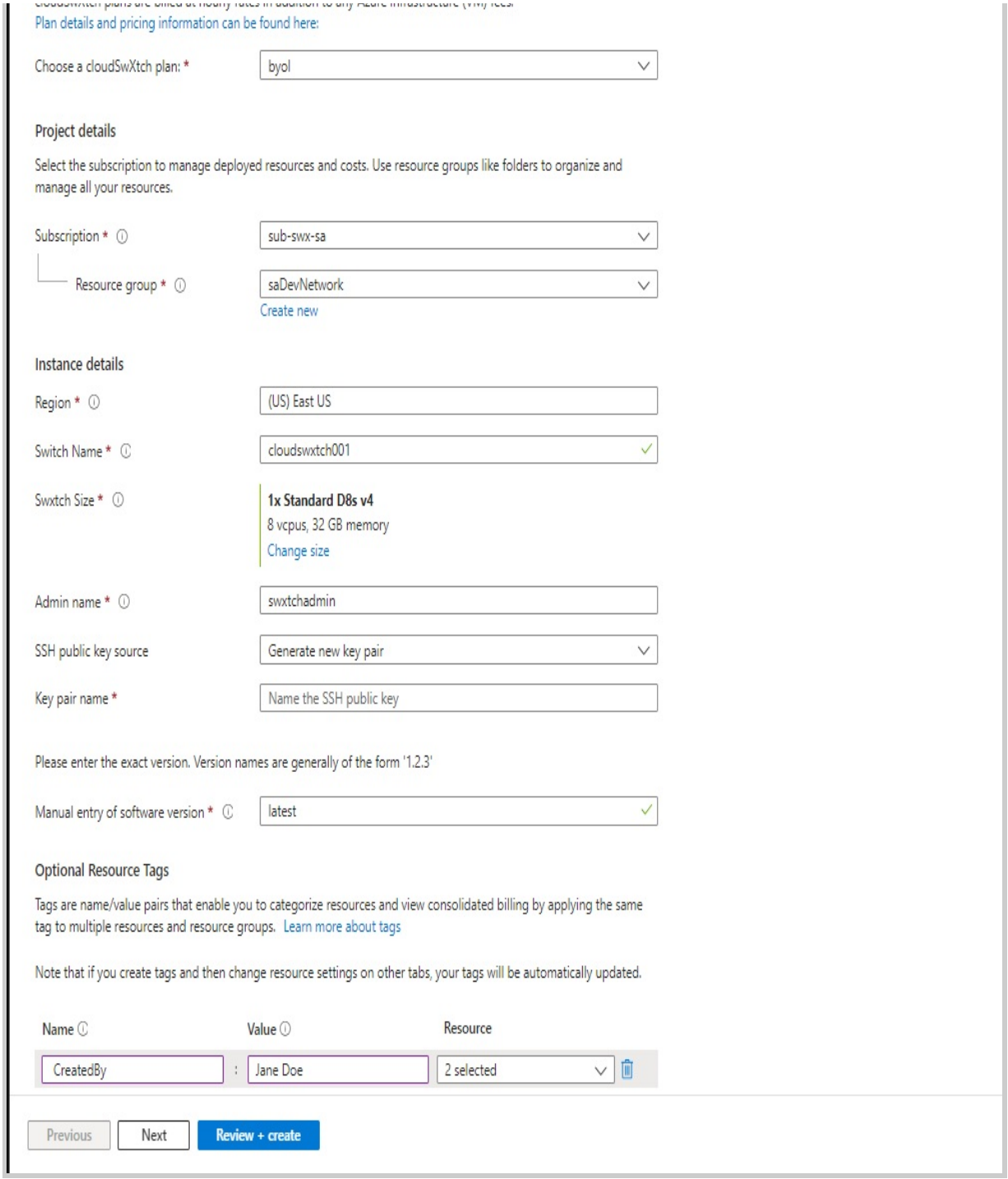
4. Click **Deploy** to launch the template UI.



5. In the **cloudSwXtch commercial plan** area, click on the **Choose a cloudSwXtch plan** dropdown and **select a plan** (BYOL or PAYG). For more information on plans see: cloudSwXtch System Requirements.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



1. In the **Project Details** area, select a **Subscription**.
2. Pick (or create) an **Azure Resource Group**.
3. In the **Instance details** area, notice how the **region** is filled in from the **Azure Resource Group**.
4. Assign the **Virtual Machine** a name. This name must be unique in both the **resource group** and the **virtual network** in which the instance will exist. It also must meet the requirements for a VM host name.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

5. Select the **cloudSwXtch size**.

> **cloudSwXtch Size Explained**
>
> The default size is **1x Standard D8s V4**. A minimum of 8 cores is suggested for cloudSwXtch deployment.

> **NOTE:**
>
> Please be aware that the owner of the Azure Subscription in which the cloudSwXtch instance is created is responsible for all cloud resources used by the switch. These fees are to the cloud provider and do not include any fees to swxtch.io for cloudSwXtch licensing.

6. Enter in an **Admin name.** This will default to **swxtchadmin**, but can be modified.

7. Enter in a **SSH public key source.** The options are:

    - **Generate new key pair.**
        - If selected, enter in **Key Pair Name**. This name must be unique among other key pairs in Azure.
    - **Use existing key stored in Azure.**
        - If selected, choose a **stored key** from the drop-down menu.
    - **Use existing public key.**
        - If selected, paste in a **SSH public Key** from Azure. Refer to https://learn.microsoft.com/en-us/azure/virtual-machines/ssh-keys-portal for how to get an existing public key.

8. Select the **software version**. The most common choice is **latest** which will use the most recent software release for this instance. For more control, a specific release version can be entered.

9. In the **Optional Resource Tags** area, optionally add **Tags**. Tags can be added to all Resources

10. Select **Next - Network.**

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

11. In the **Configure virtual networks** area**, select** a previously created virtual network.

> **WARNING**
>
> Due to an issue with Azure templates, **do not select the Create new option** for the network because the created network will not be accessible to you. **Always** select a previously created virtual network.

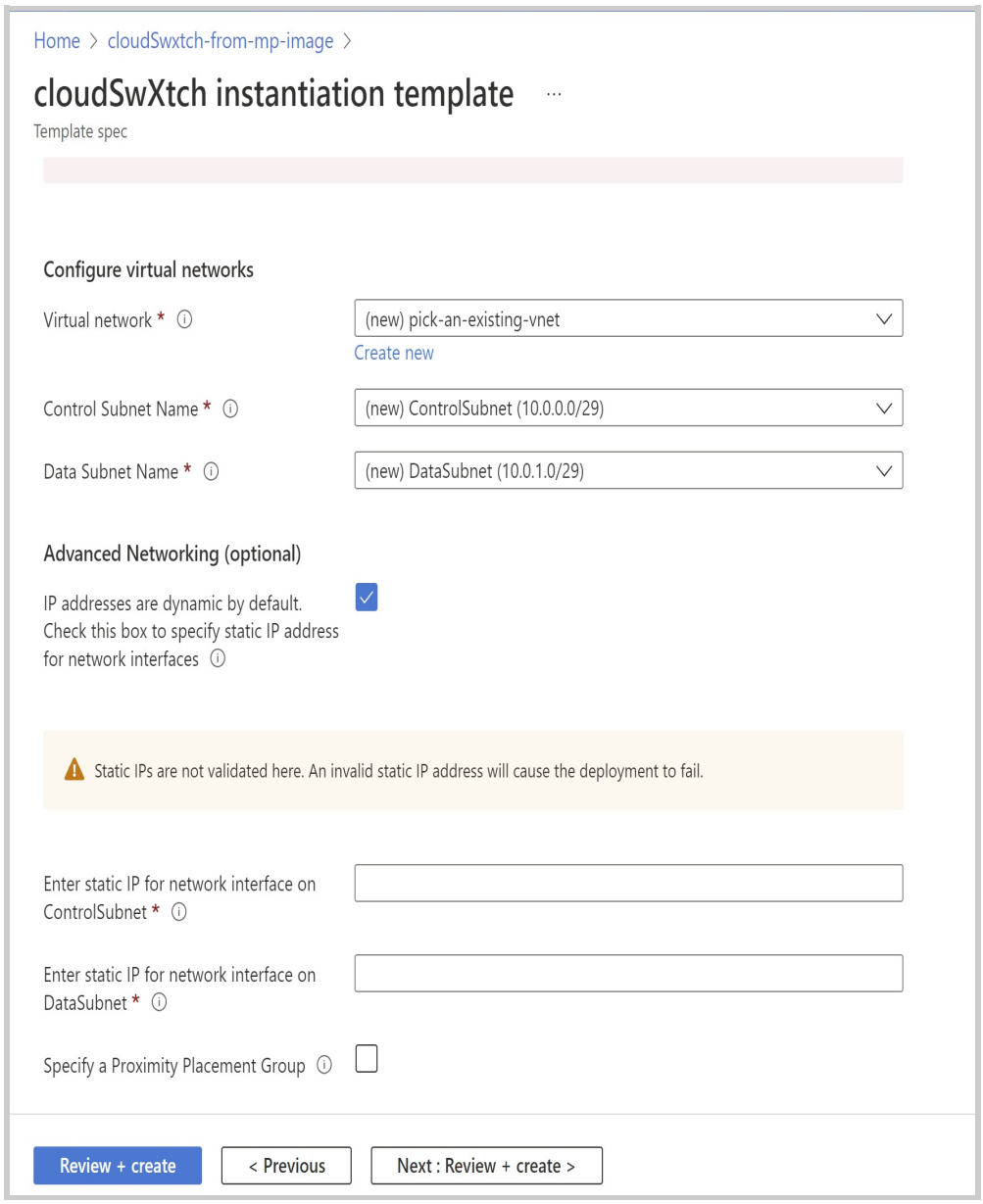
> **Network**
>
> The cloudSwXtch must be associated with a virtual network and the virtual network must have at least two subnets: one for control plane and one for data plane traffic. See “System Requirements” above for details.

12. In the **Configure virtual networks** area, select a **Control Subnet Name.**

13. Select a **Data Subnet Name.**

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

14. **OPTIONAL**: In the **Advanced Networking (optional)** section:
    - Add a static IP Address.
    - Specify a Proximity Placement Group.



15. Select **Review and Create.**
16. Review the **plan pricing.**
17. Read the **Terms & Conditions.**
18. Select **I agree** when ready.

The creation will take 1-3 minutes depending on Azure vagaries. When done, a cloudSwXtch instance shall exist within the selected Azure Resource Group. Your cloudSwXtch is now ready for use.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Post-Installation

- **IMPORTANT:** If this is a new install then each client that is expected to get traffic from the cloudSwXtch will need a xNIC installed. If this is a existing install then each client with an xNIC already installed will need to be upgraded. Please see **xNIC Installation**.
- For Windows-related OS/servers, It's important to reboot the machine, once the installation is complete, in order to be able to execute cloudSwXtch tools properly from any client's user home directory.

24/7 Operations

If the services need to be up and running 24/7 swXtch.io suggests that redundant systems exist for which will be referred to as "Main" and "Backup". During an upgrade the Backup system should be upgraded, then the traffic should be routed to the Backup while the Main is upgraded.

Uninstalling cloudSwXtch

Delete the cloudSwXtch instance as you would any other virtual machine.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install cloudSwXtch via Azure Marketplace

Installing cloudSwXtch via Template

The best method for deploying a cloudSwXtch on Azure is via a template. For more information on this method, please review the **Install cloudSwXtch on Azure** guide.

Prerequisites

Before starting, ensure that you have **validated your subnets** on Azure. Return to this page after completing that preliminary step.

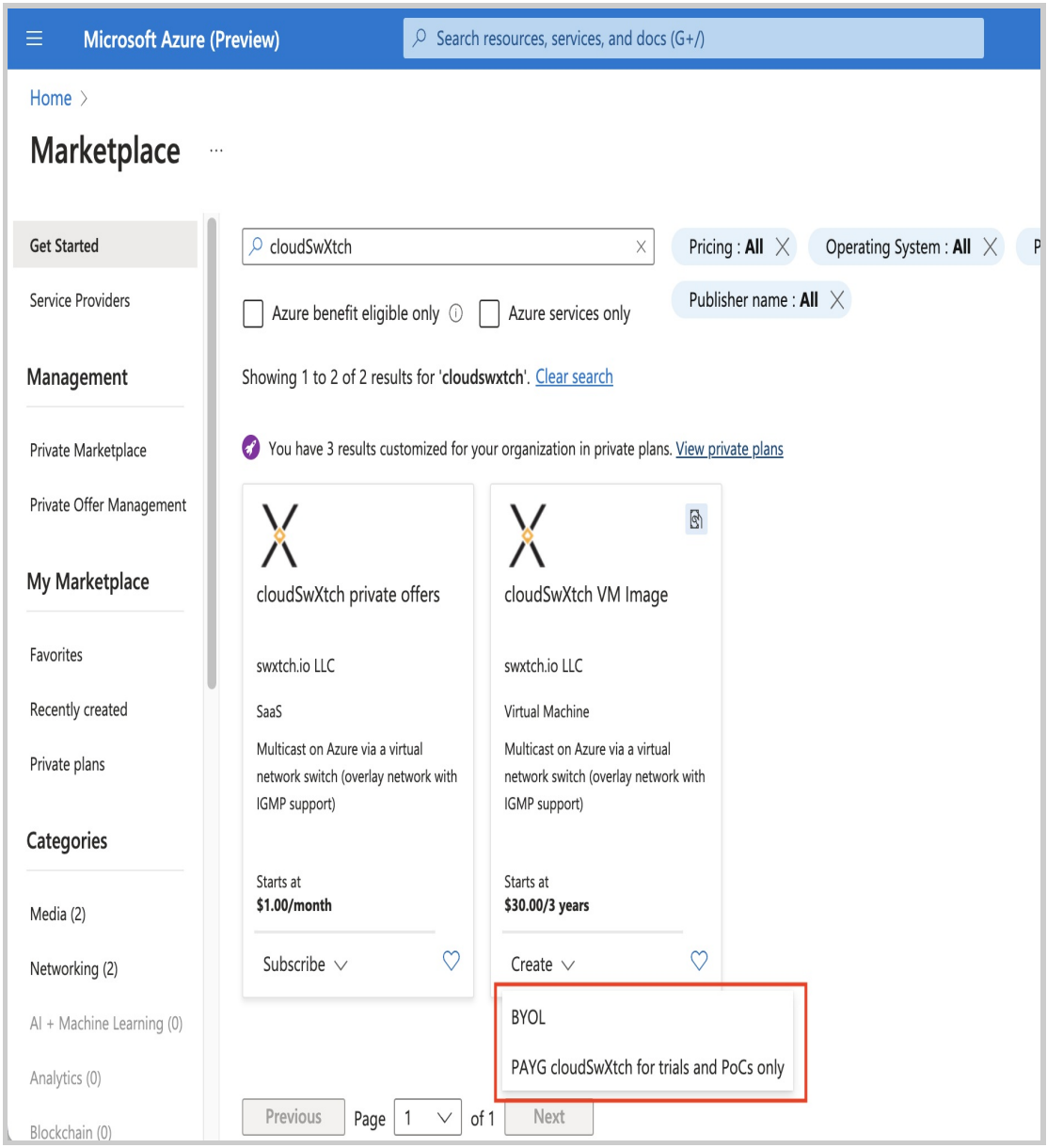
Creating a Virtual Machine

1. **Log in to the Azure Portal.** You will need the following permissions to create and manage virtual machines and to create Managed Applications.
    - **virtual-machine-contributor**: To create and manage virtual machines.
    - **managed-application-contributor-role:** To create Managed Applications.
2. Select **Marketplace**.
3. Search for **cloudswXtch**.
4. **Select** a plan. For more information, see: cloudSwXtch System Requirements.
5. Click on the **cloudSwXtch VM Image** drop down menu to select a plan. **Please note**: A BYOL instance will require users to obtain a license from swXtch.io. For more information, see here.

    cloudSwXtch BYOL

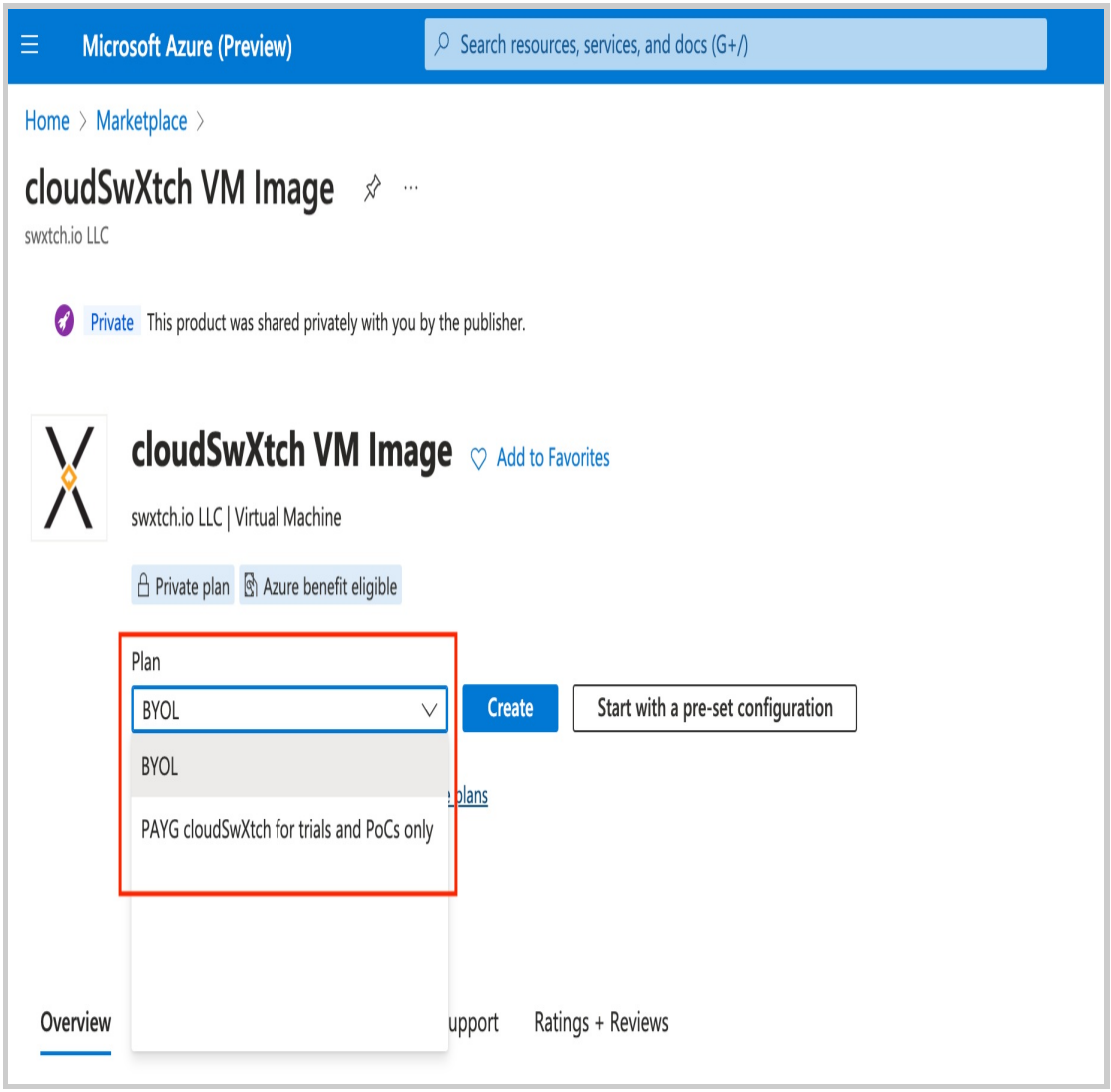
    It is recommended that users select **cloudSwXtch BYOL**, which allows for more customizability including expanded bandwidth, increased endpoint limit and additional licensable features. Users will need to request a license from support@swxtch.io. For more information, please see **cloudSwXtch System Requirements**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



The **Create a virtual machine** will open with the selected plan. If the plan was not selected in the previous screen, then the following screen will open to choose a plan.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



6. Select either **Create** or **Start with a pre-set configuration**.

NOTE

swxtch.io is just using the standard Azure Marketplace VM from Image method, this document will not go over all the tabs and fields in the tabs as they are not cloudSwxtch specific. Some things of note in the Azure Marketplace VM image creation are as follows:

- The **Start with a pre-set configuration** vs **Create** will eventually lead to the same UI where there are many tabs to enter data. However, the **Start with a pre-set configuration** will fill in certain fields based on the user's selections. For example, in the **Basics** tab it will fill in **Boot diagnostics**, **Availability options**, and **Size**. In addition, the **Disks** tab will fill in the OS disk type.
- **REMINDER**: This Market Place method will only create one NIC. The second required NIC will need to be added after creation.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

7. Follow the tabs and make appropriate selections – there are a number of fields that have to be filled in to create a cloudSwXtch instance.

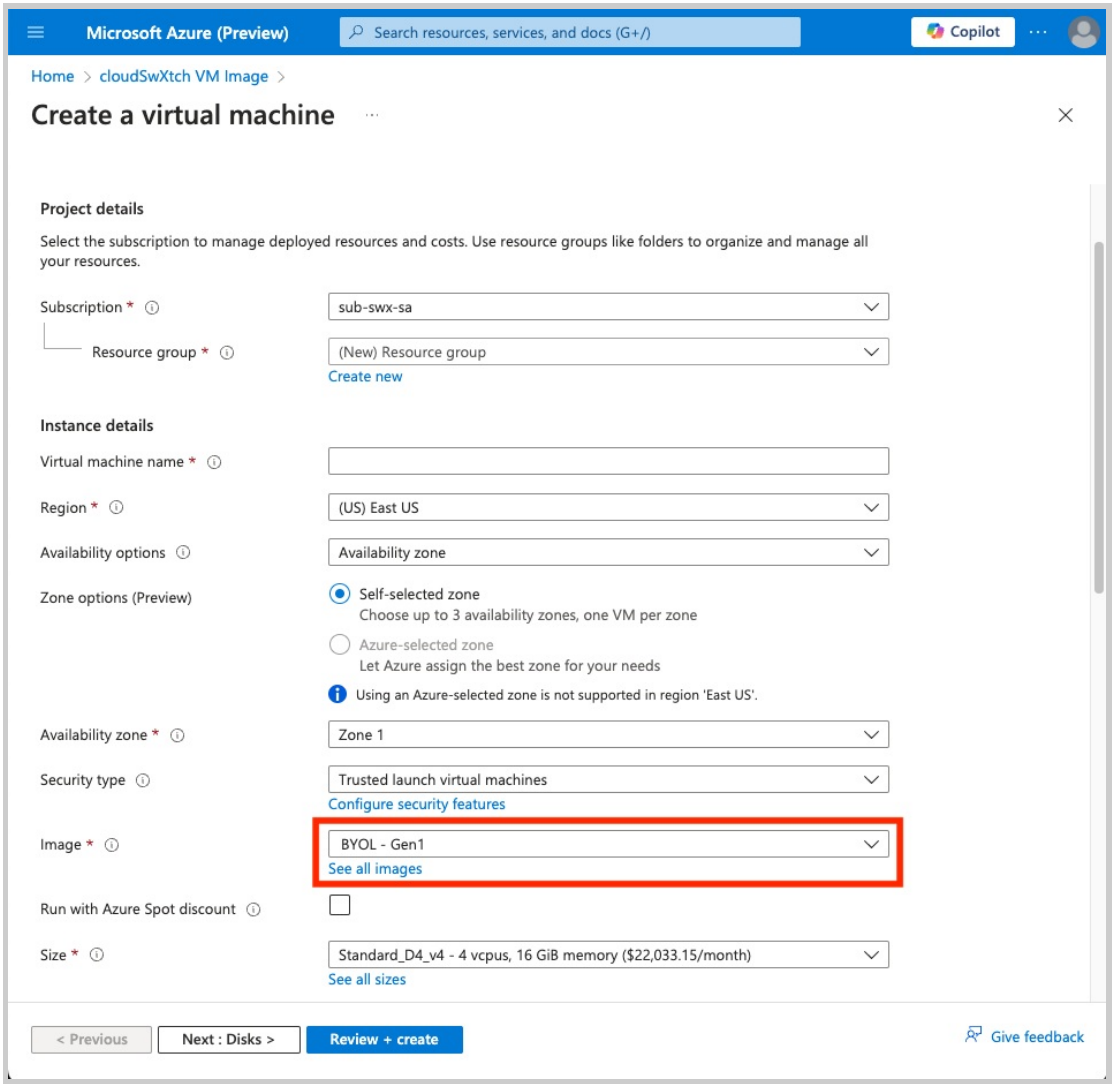
8. In the **Basics** tab, select a **Subscription**.

9. Choose (or create) a **Resource Group**.

10. Assign the **Virtual Machine Name**. This name must be unique.

11. Select a **Region**.

12. Confirm the **Image** is the correct one you selected when choosing a plan: **cloudSwXtch BYOL** or **cloudSwXtch PAYG**.



13. Select the **Software Version**. The most common choice is **latest**, which will use the most recent software release for this instance. For more control, a specific release version can be entered.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

14. Continue on the **Networking** Tab.

> **Networking Tab**
>
> The cloudSwXtch instance must be associated with a virtual network and the virtual network must have at least two subnets: one for control plane and one for data plane traffic. This user interface only allows attachment of one subnet. Below steps will describe how to add a second subnet after creation. See “System Requirements” above for details.

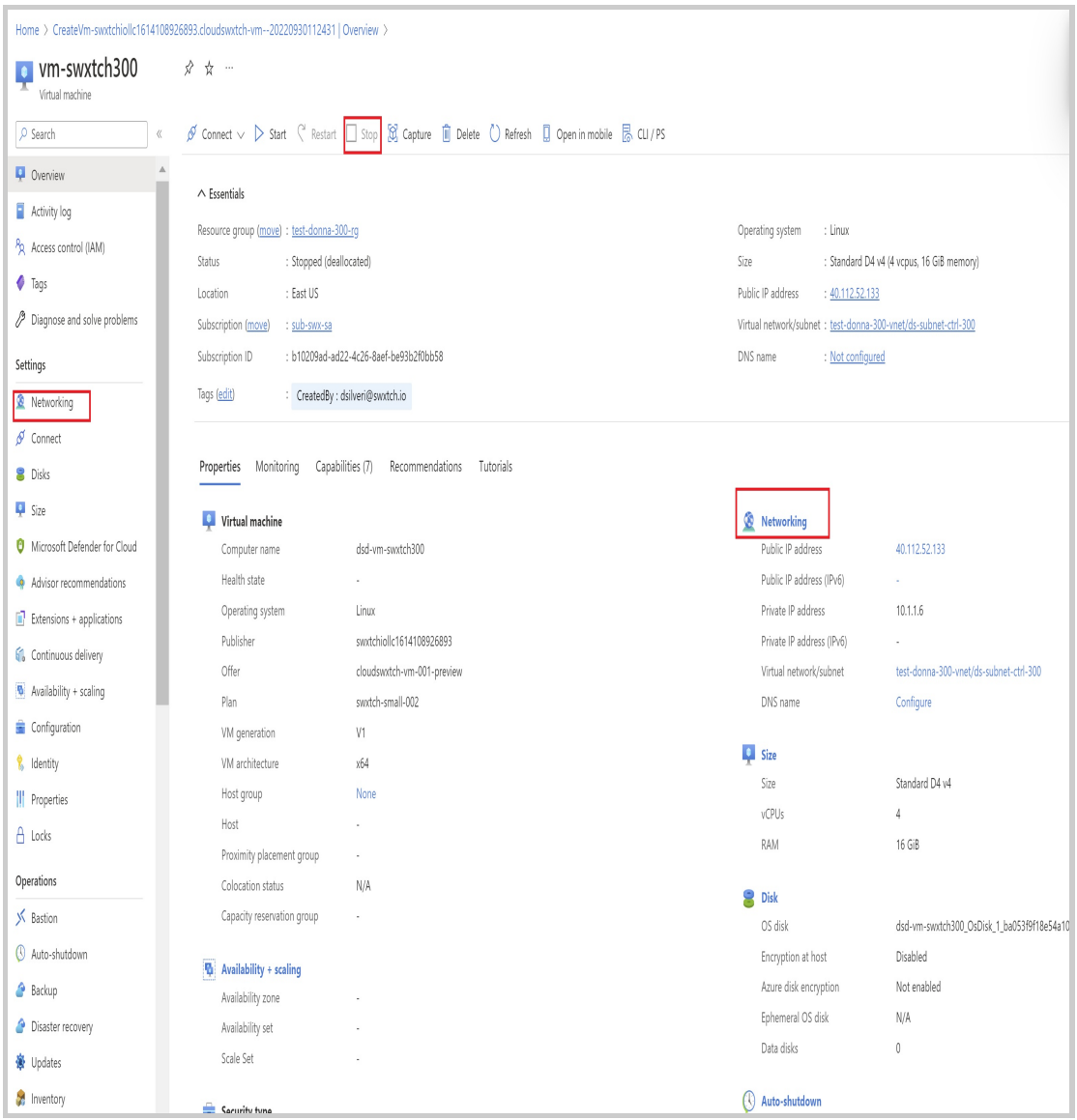
15. Check **Delete public IP and NIC when VM is deleted**.
16. **OPTIONAL**: Change values on other tabs.
17. Select **Review and Create**.
18. Carefully review the **plan pricing**.
19. Read the **Terms & Conditions**.
20. Select **I agree** when ready. **Note:** The creation will take 2-3 minutes depending on Azure varieties.

Creating the Second Subnet *REQUIRED*

1. Navigate to the newly created VM by selecting the **Go to Resource** button.
2. Click **Stop** at the top of the toolbar.
3. Select **Yes** when prompted.

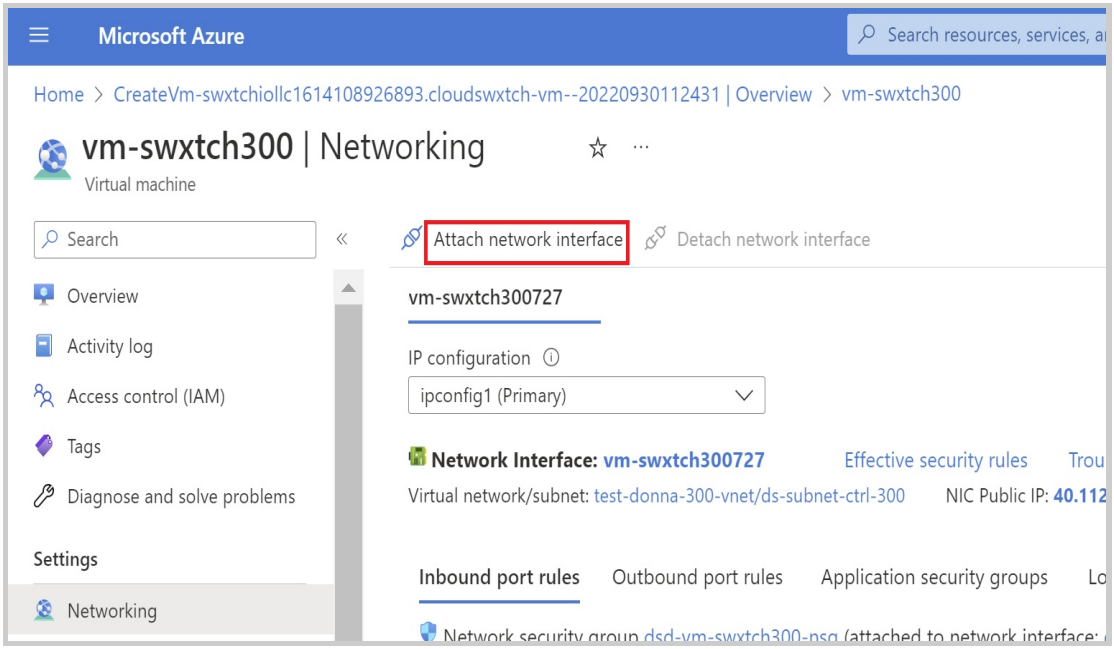
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. Click **Networking** on the left hand side under settings. Alternatively, you can select **Networking** in the main Properties page.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

5. Select **Attach network interface**.



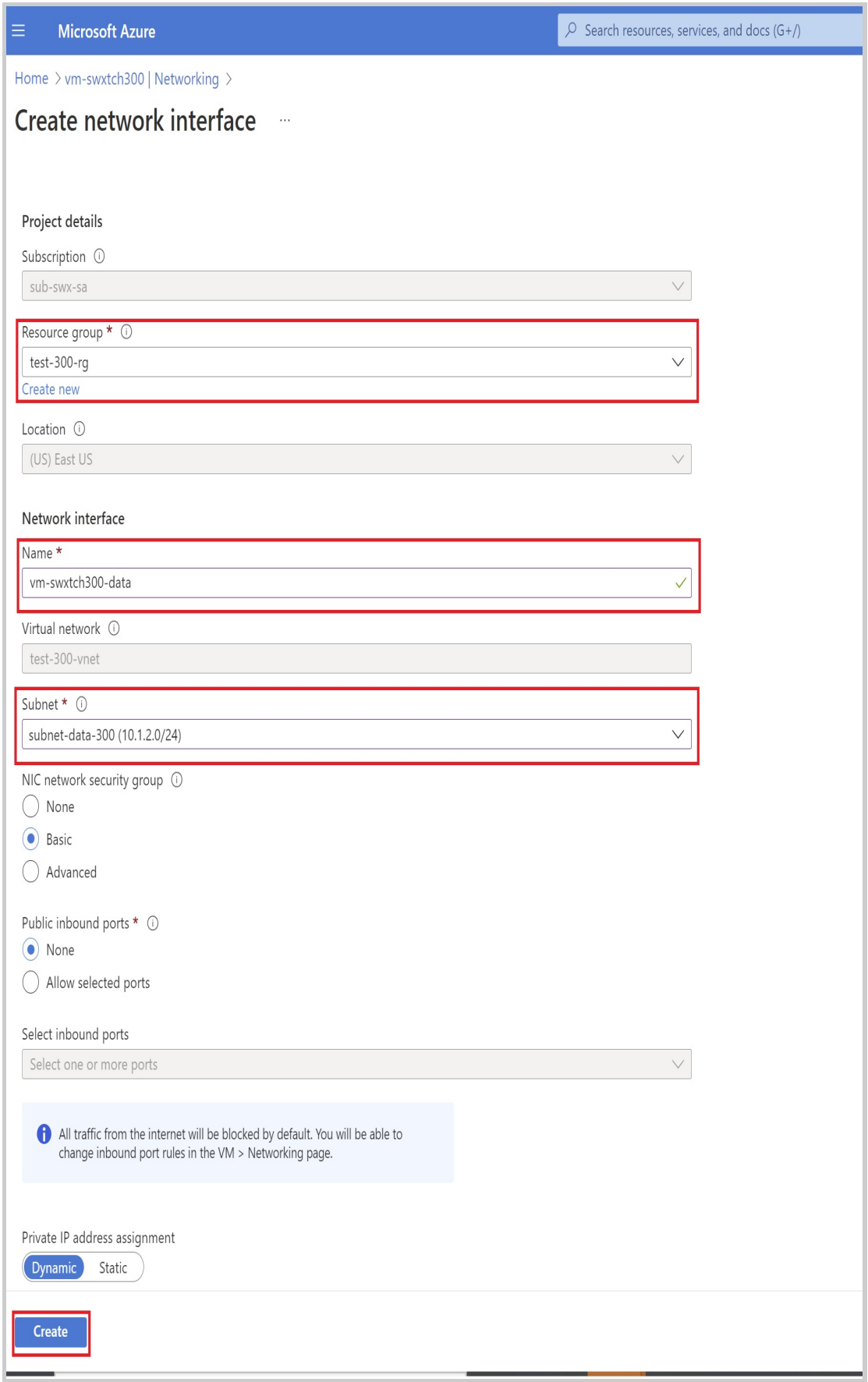
6. Select a **Resource Group** under **Project Details.**

7. Enter in a **Name** under **Network Interface.**

8. Select a **Subnet. Please note:** You can optionally change other data.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

9. Select **Create.**



10. Refresh the screen after completing the form and the second subnet should be added in a second tab.

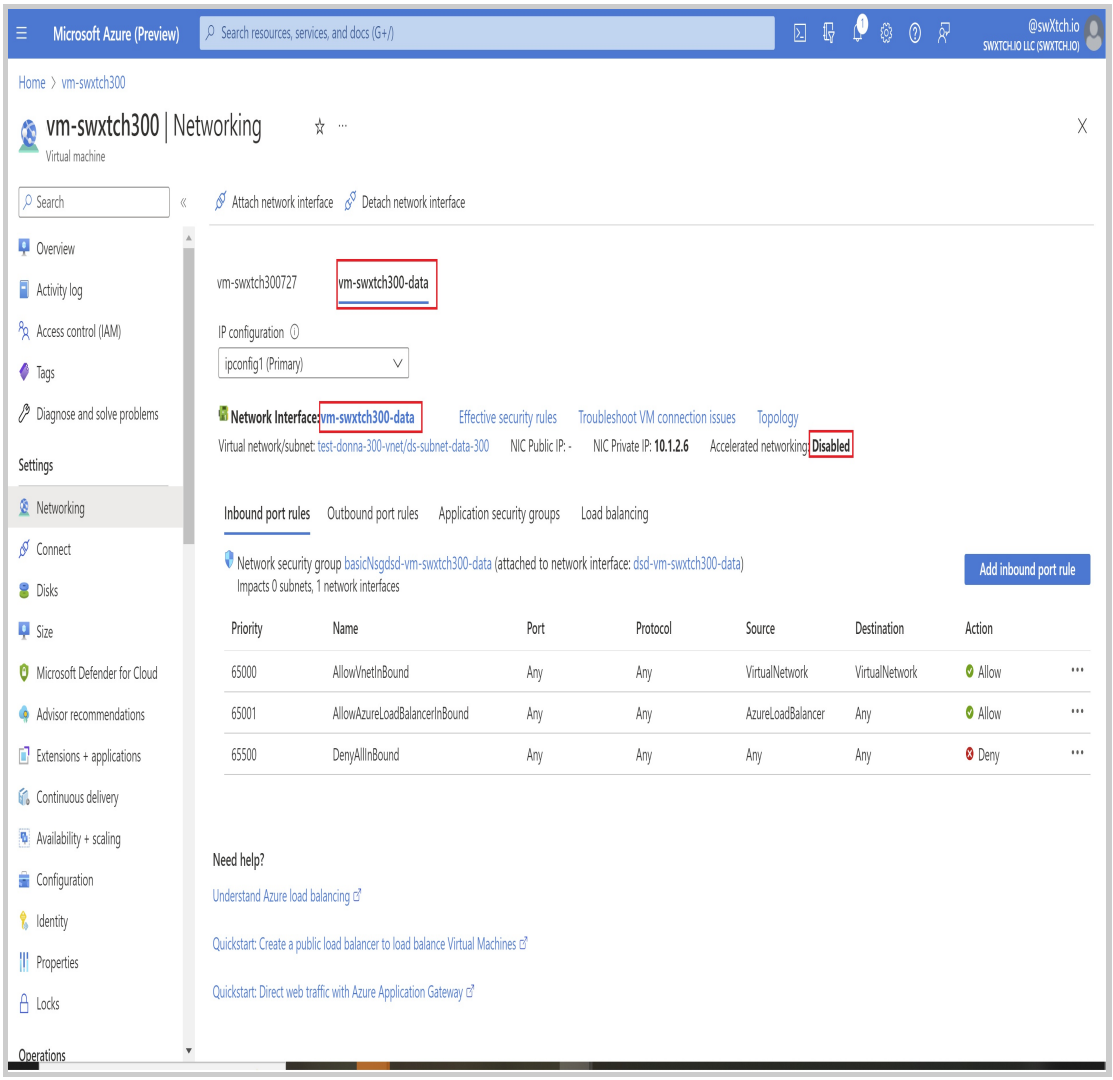
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Enabling "Accelerated Networking" *REQUIRED*

The newly created Network Interface needs to be updated to enable **Accelerated Networking** to do this follow the steps below:

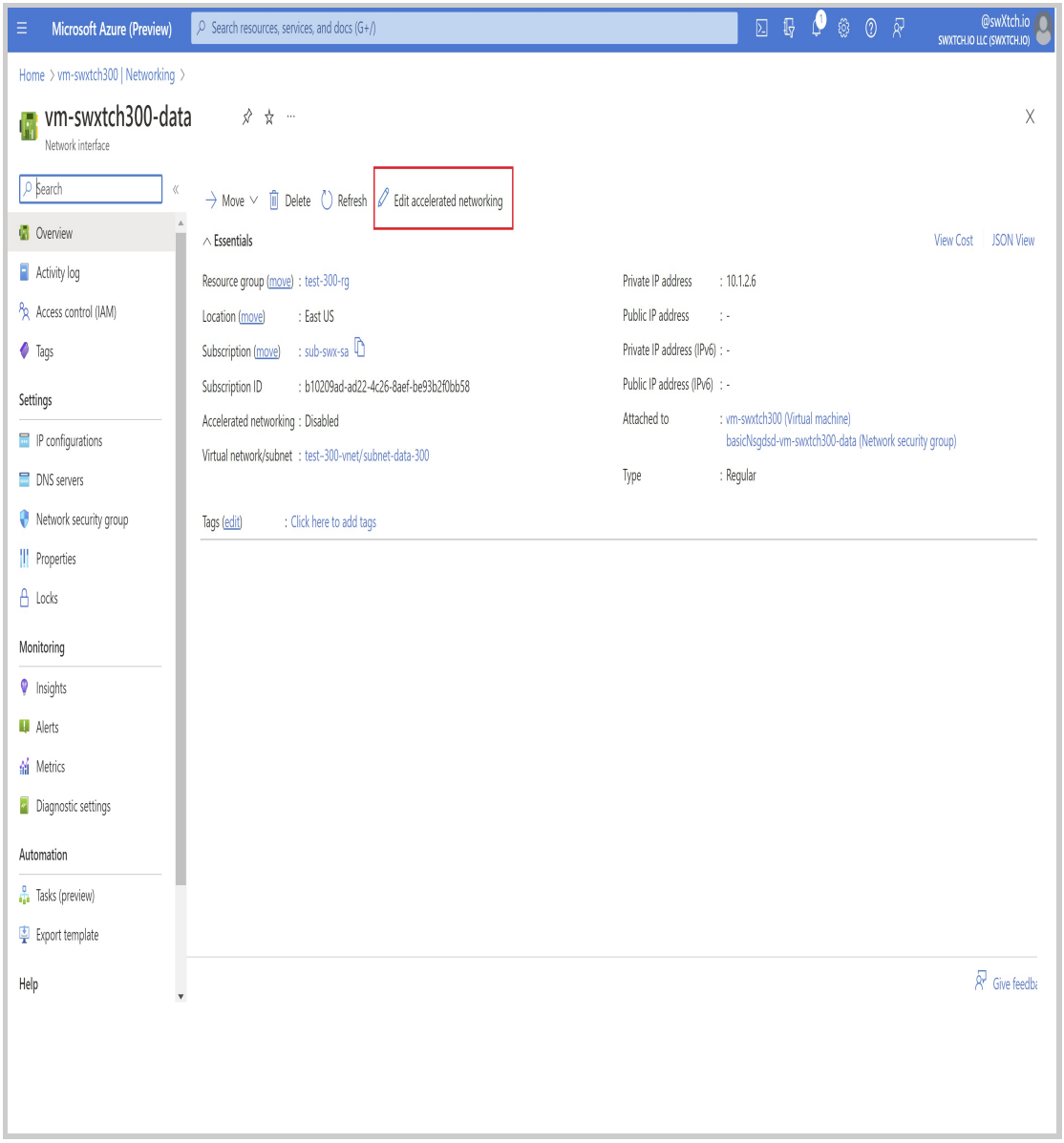
1. Select the **Network Interface**. In the example below, it is named **vm-swxtch-300-data**.

2. Click the blue link to the **Network Interface**.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. Click **Edit accelerated networking**.

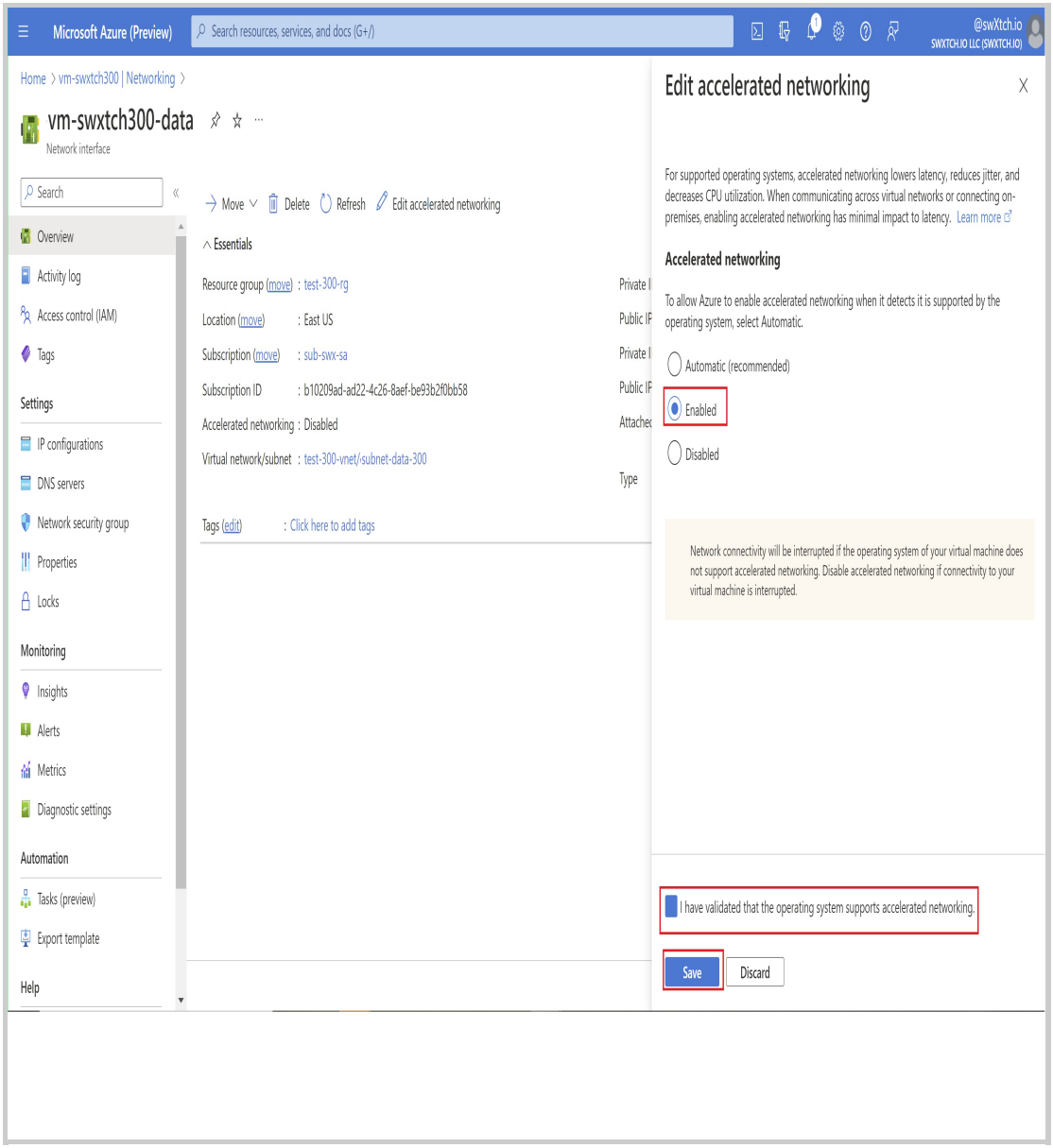


4. Select **Enable**.

5. Select **I have validated that the operating system supports accelerated networking**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

6. Click **Save**.



7. **Start** the VM for use.

> **Upgrading your cloudSwXtch Instance**
>
> After deployment, it is recommended to update your cloudSwXtch instance to latest. Please refer to the **Upgrading cloudSwXtch** article for more information.

Required Step for BYOL: Request a License from swXtch.io

Users deploying a BYOL instance of cloudSwXtch will need to contact swXtch.io for a license file. For more information, see How to License a cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Important

If this is a new install then each client that is expected to get traffic from the cloudSwXtch or send to the cloudSwXtch will need a xNIC installed. If this is a existing install then each client with an xNIC already installed will need to be upgraded. Please see **xNIC Installation**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Deploy cloudSwXtch with Terraform on Azure

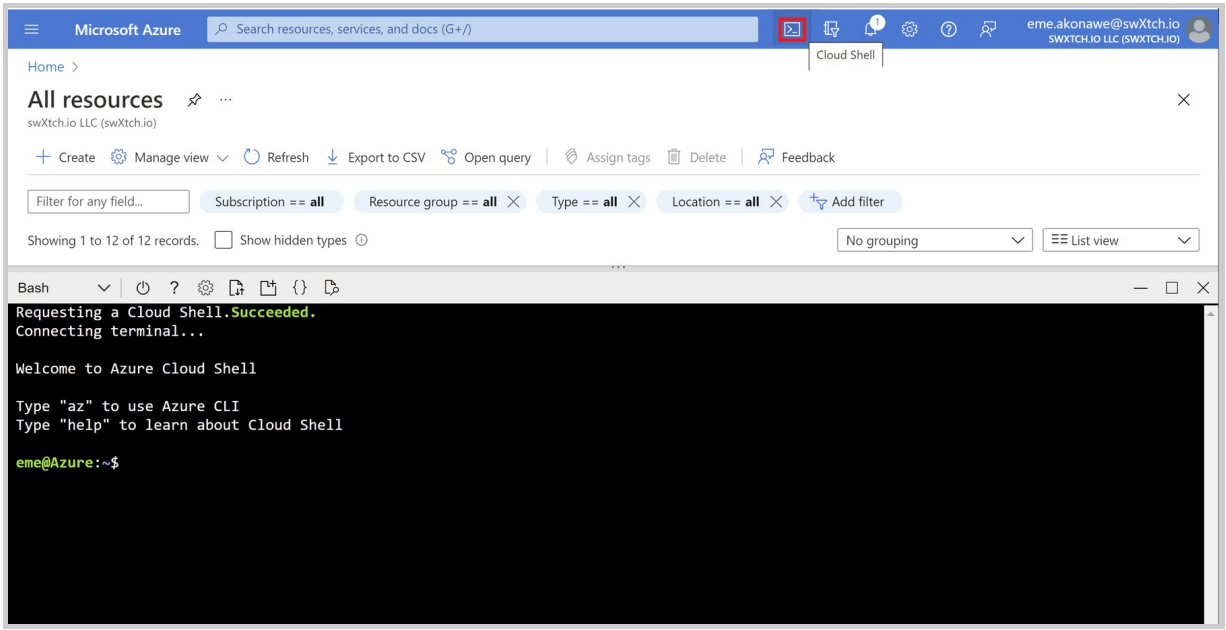
WHAT TO EXPECT

By default, the terraform script will spin up a "small" cloudSwXtch. You can make edits to the Azure/terraform/terraform.tfvarse file to declare a different cloudSwXtch size. There is also an option to delegate static ip addresses on your cloudSwXtch. Further details on how to do this can be found at the end of this article.

Deploying cloudSwXtch with Terraform on Azure

1. Sign-in to your Azure portal under the subscription where you want to deploy the cloudSwXtch.

2. Open the Azure Cloud Shell interface and select the Bash environment as shown.



3. Clone the example repository from GitHub.

You can do this either **via SSH** (requires setting up your SSH authentication with GitHub):

Bash	Copy

```
$ git clone git@github.com:swxtchio/cloudSwXtch-support.git
```

or via HTTPS:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
$ git clone https://github.com/swxtchio/cloudSwXtch-support.git
```

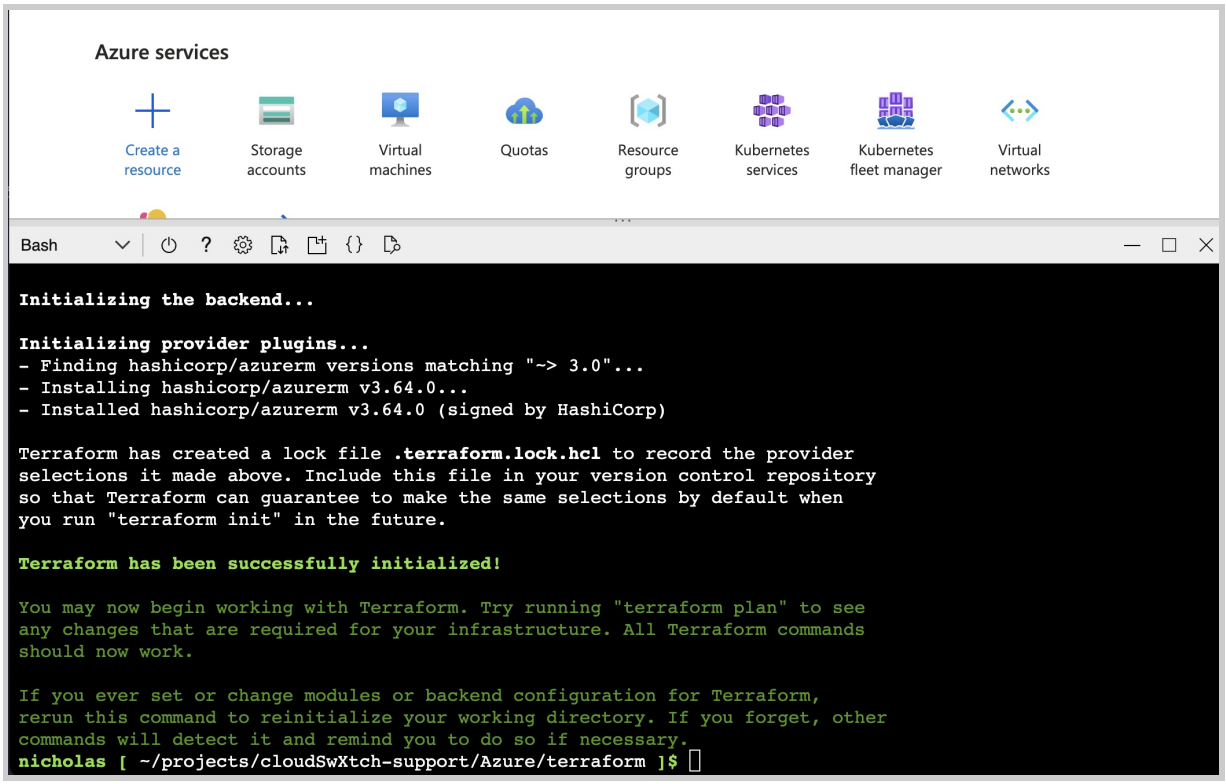
4. Update the values in the Azure/terraform/terraform.tfvarse file to match your existing azure resources such as: resource group, virtual network, subnets, etc.

> The format of the key file that the scripts can process is the ssh-rsa type. The content of the file should start with "ssh-rsa AAAAB3..."

5. In the Cloud Shell terminal, cd into the Azure/terraform directory and initialize the terraform environment:

Bash | Copy

```
$ cd Azure/terraform
$ terraform init
```

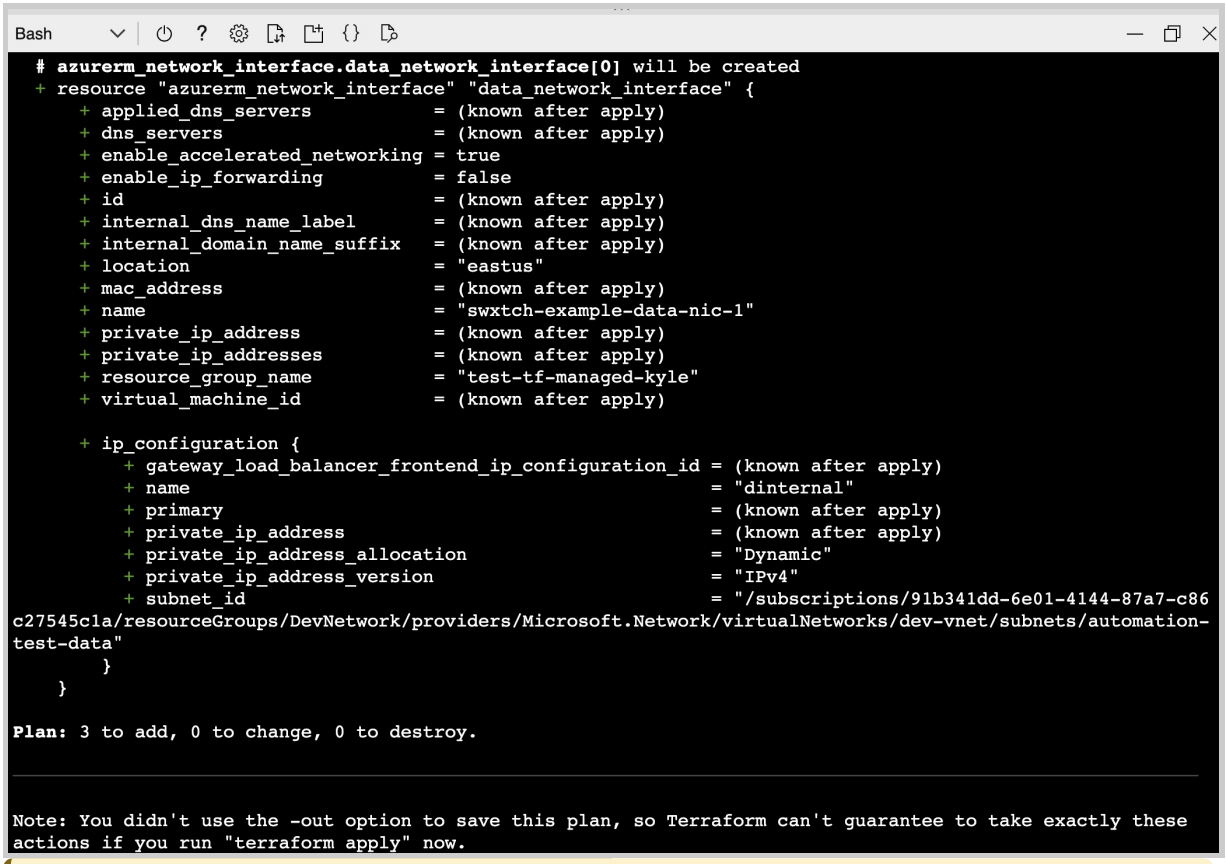


6. Now that Terraform has been initialized, run this command to evaluate the config and confirm the desired output which will be shown:

Bash | Copy

```
$ terraform plan
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



```
Bash
  # azurerm_network_interface.data_network_interface[0] will be created
  + resource "azurerm_network_interface" "data_network_interface" {
      + applied_dns_servers           = (known after apply)
      + dns_servers                   = (known after apply)
      + enable_accelerated_networking = true
      + enable_ip_forwarding          = false
      + id                            = (known after apply)
      + internal_dns_name_label       = (known after apply)
      + internal_domain_name_suffix   = (known after apply)
      + location                      = "eastus"
      + mac_address                   = (known after apply)
      + name                          = "swxtch-example-data-nic-1"
      + private_ip_address            = (known after apply)
      + private_ip_addresses          = (known after apply)
      + resource_group_name           = "test-tf-managed-kyle"
      + virtual_machine_id            = (known after apply)

      + ip_configuration {
          + gateway_load_balancer_frontend_ip_configuration_id = (known after apply)
          + name                                               = "dinternal"
          + primary                                            = (known after apply)
          + private_ip_address                                 = (known after apply)
          + private_ip_address_allocation                      = "Dynamic"
          + private_ip_address_version                         = "IPv4"
          + subnet_id                                          = "/subscriptions/91b341dd-6e01-4144-87a7-c86
c27545c1a/resourceGroups/DevNetwork/providers/Microsoft.Network/virtualNetworks/dev-vnet/subnets/automation-
test-data"
        }
    }

Plan: 3 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these
actions if you run "terraform apply" now.
```

> Since you are using all pre-existing resources to deploy your cloudSwXtch, there should only be 3 resources added - 1 cloudSwxtch, and 2 NICs - as can be seen at the bottom of the screenshot, “Plan: 3 to add, 0 to change, 0 to destroy.”

7. Run the Terraform apply command (followed by “yes” when prompted) to approve the action.

Bash | Copy

```
Terraform apply
yes
```

8. Once the resources have applied successfully you can view the resources created from your Azure portal as confirmation of a successful deployment.

STATIC IPs

If you'd like to deploy a cloudSwXtch using Static IPs, then you just need to make some small changes to the azure_deployswxtch.tf & terraform.tfvars files.

1. Un-comment the Parameter private_ip_address in the azure_deployswxtch.tf code file for both your data_network_interface & control_network_interface resources.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
source "azurerm_network_interface" "data_network_interface" {
count               = var.counter
name                = "${var.data_nic}-${count.index +1}"
location            = data.azurerm_resource_group.resource_group.location
resource_group_name = data.azurerm_resource_group.resource_group.name
enable_accelerated_networking = true

ip_configuration {
  name                          = "dinternal"
  subnet_id                     = data.azurerm_subnet.datasubnet.id
  private_ip_address_allocation = "Static"
  private_ip_address            = var.datanic_staticip
}
```

```
39  resource "azurerm_network_interface" "control_network_interface" {
40    count               = var.counter
41    name                = "${var.control_nic}-${count.index +1}"
42    location            = data.azurerm_resource_group.resource_group.locatio
43    resource_group_name = data.azurerm_resource_group.resource_group.name
44
45    ip_configuration {
46      name                          = "cinternal"
47      subnet_id                     = data.azurerm_subnet.ctrlsubnet.id
48      private_ip_address_allocation = "Static"
49      private_ip_address = var.controlnic_staticip
50    }
51  }
```

2. Set the parameter private_ip_address_allocation to "Static".

Your 2 lines of code should look like below for both network interface resources:

Bash | Copy

```
private_ip_address_allocation = “Static”
private_ip_address = var.datanic_staticip
```

Your terraform.tfvars file will have variables defined for your control and data NIC StaticIP definitions. You can update those values based on your subnet setup.

Note: This static IP address allocation will only work for swxtch_count of 1.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install cloudSwXtch for an Air-Gapped Environment

> **WHAT TO EXPECT**
>
> In this article, you will learn how to install a cloudSwXtch in an Air-Gapped (Closed Network) environment for Azure. For standard Azure installation instructions, please see the **cloudSwXtch on Azure** article.

Before You Start

Review VM Requirements for a cloudSwXtch Instance in cloudSwXtch System Requirements.

VM Image Creation

The cloudSwXtch software is delivered as a **Virtual Machine Disk Image**. This Image file can be added to an Azure **Image Gallery**. Images in an Image Gallery can be used to create Virtual Machines.

To assist with creation of VMs from images in a gallery, swXtch.io provides instructions on how to accomplish the following:

1. Get the VM Disk Image
2. Upload the VM Image into an Azure Storage Account
3. Create a VM Image from the Disk Image
4. Create cloudSwXtch from VM Image
5. License the cloudSwXtch

Complete all steps to successfully install cloudSwXtch for an Air-Gapped environment.

STEP ONE: Get the VM Disk Image

Log onto an environment that has access to the internet and download the following file (~30GB):

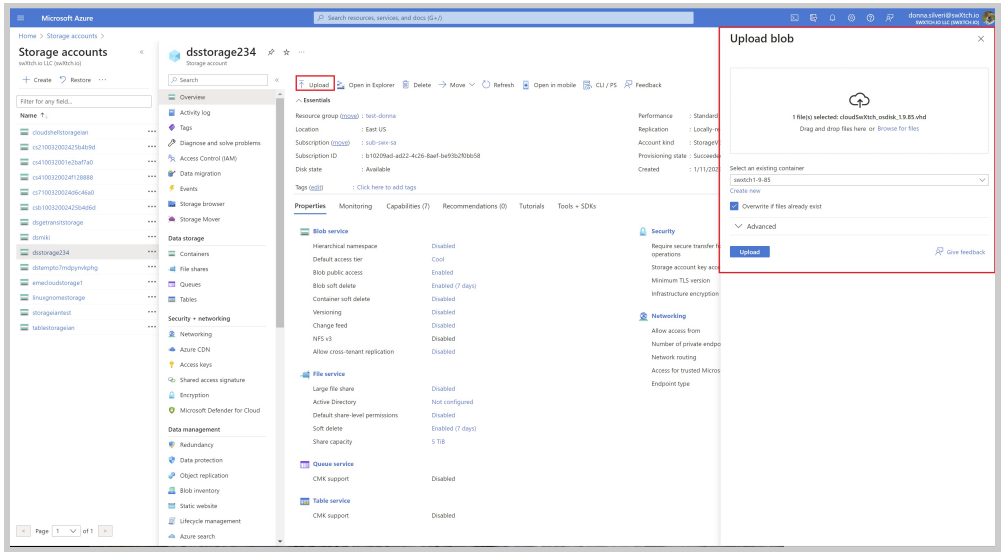
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Plaintext	Copy

```
https://swxtchpublic.blob.core.windows.net/3hwgfe98hfglsrdfh4/cloudSwXtch_osdisk_1.9.85.vhd
```

STEP TWO: Upload the VM Disk Image into an Azure Storage Account

1. Place the file onto a machine with access to the **Azure Air Gapped Environment**.
2. Upload the files into an Azure storage account in the **secure** Azure Environment.
    a. Log into the **Azure** Portal
    b. Navigate to **Storage Accounts**.
    c. Select the desired storage account.
    d. Select the desired Container or **create a new one**.
    e. Select **Upload** and select the VM Disk Image file you copied to the local PC.



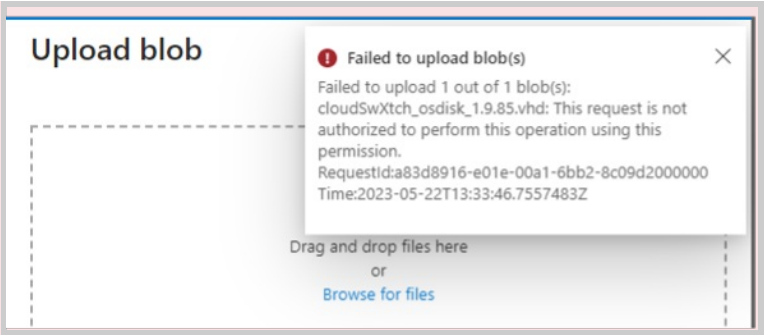
    f. Start the upload and wait for it to complete.

This may take some time to upload the file (up to an hour). When completed, the file should show with a green checkbox.

Failed to Upload Blob(s) Message

If you receive a "Failed to Upload Blob(s)" message when uploading the file in the Storage Account, select **Configuration** and validate the **Allow storage key access** is enabled.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



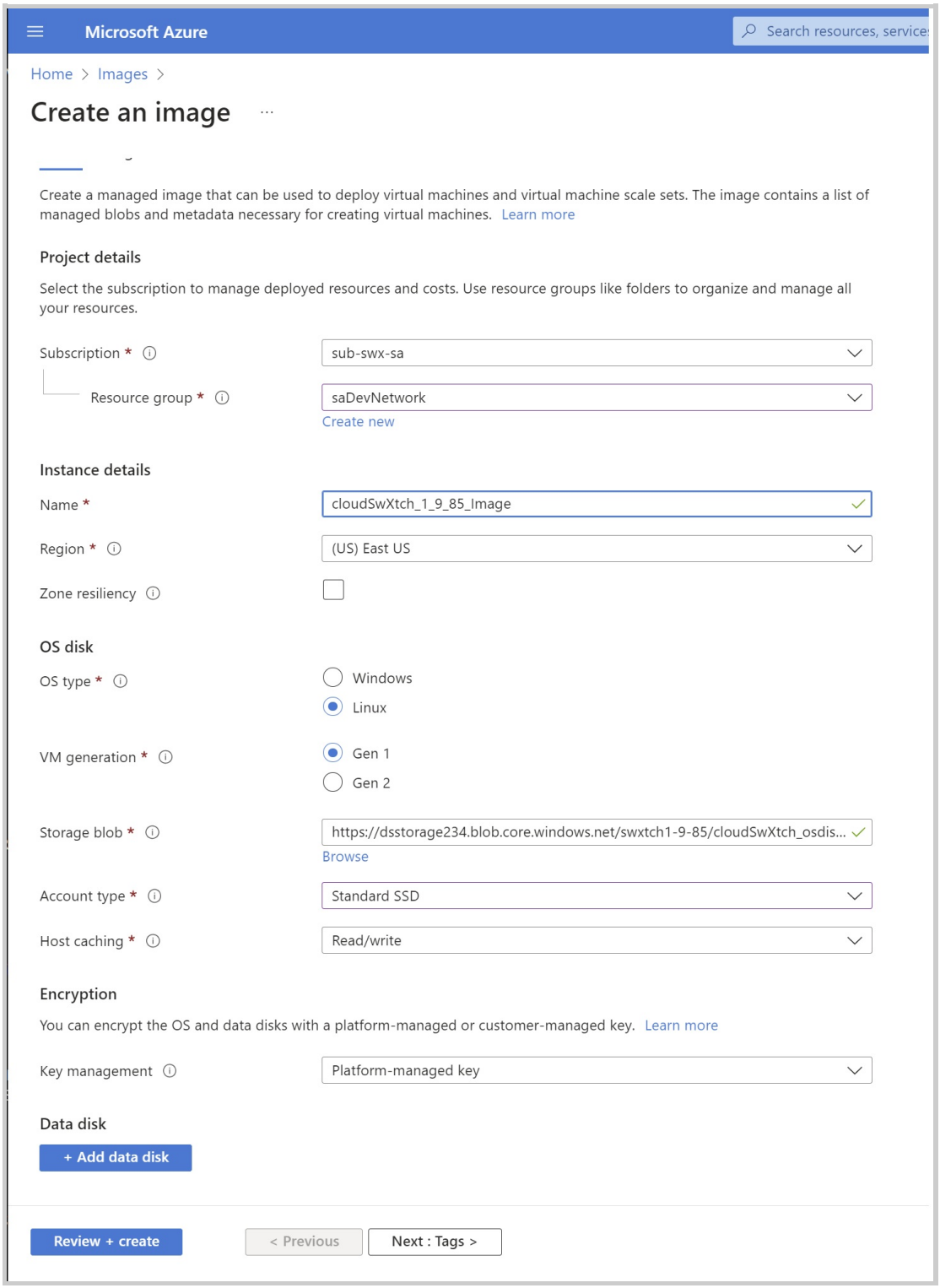
STEP THREE: Create a VM Image from the Disk Image

Once we have a disk image in storage, we can use it to create a VM image. A VM image is a *description of* a VM. The real VM will be created later. The VM Image only needs to be created once. Any number of VMs can be instantiated from a single VM image.

1. In the Azure Portal, **Search** for and **select Images**.
2. Select **Create**.
3. Select the appropriate **Resource Group**.
4. Give the VM Image a name. The cloudSwXtch instance will be created later with a different name. Pick a name with the cloudSwXtch software version in it as you may end up with multiple images after some time.
5. Ensure that the region is the same for the storage account holding the disk image.
6. Select **Linux** as the OS type
7. Select **Gen 1**.
8. Click **Browse** on the **Storage Blob**.
    a. In the new panel, navigate to the storage account and container holding the disk image.
    b. Select the file that was previously uploaded.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

9. For **Account Type**, select **Standard SSD**. See the example of the screen filled out completely.



10. If tags are desired, then select **Tags** and enter the required tags.

11. The other fields can be left as default.

12. Select **Review and create.**

13. When validation passes, select **Create**. When it is complete, click **Go to Resource** to see the image.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

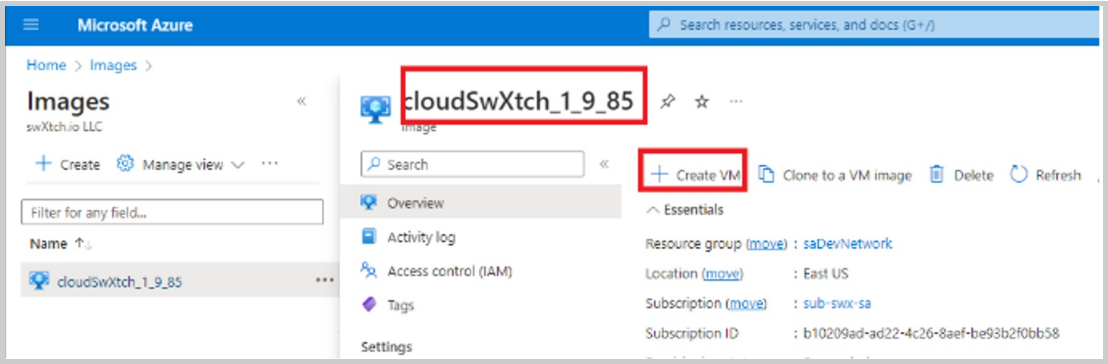
STEP FOUR: Create cloudSwXtch from VM Image

Now that we have a cloudSwXtch VM Image, we can use it to instantiate a cloudSwXtch.

1. Navigate to **Images**.

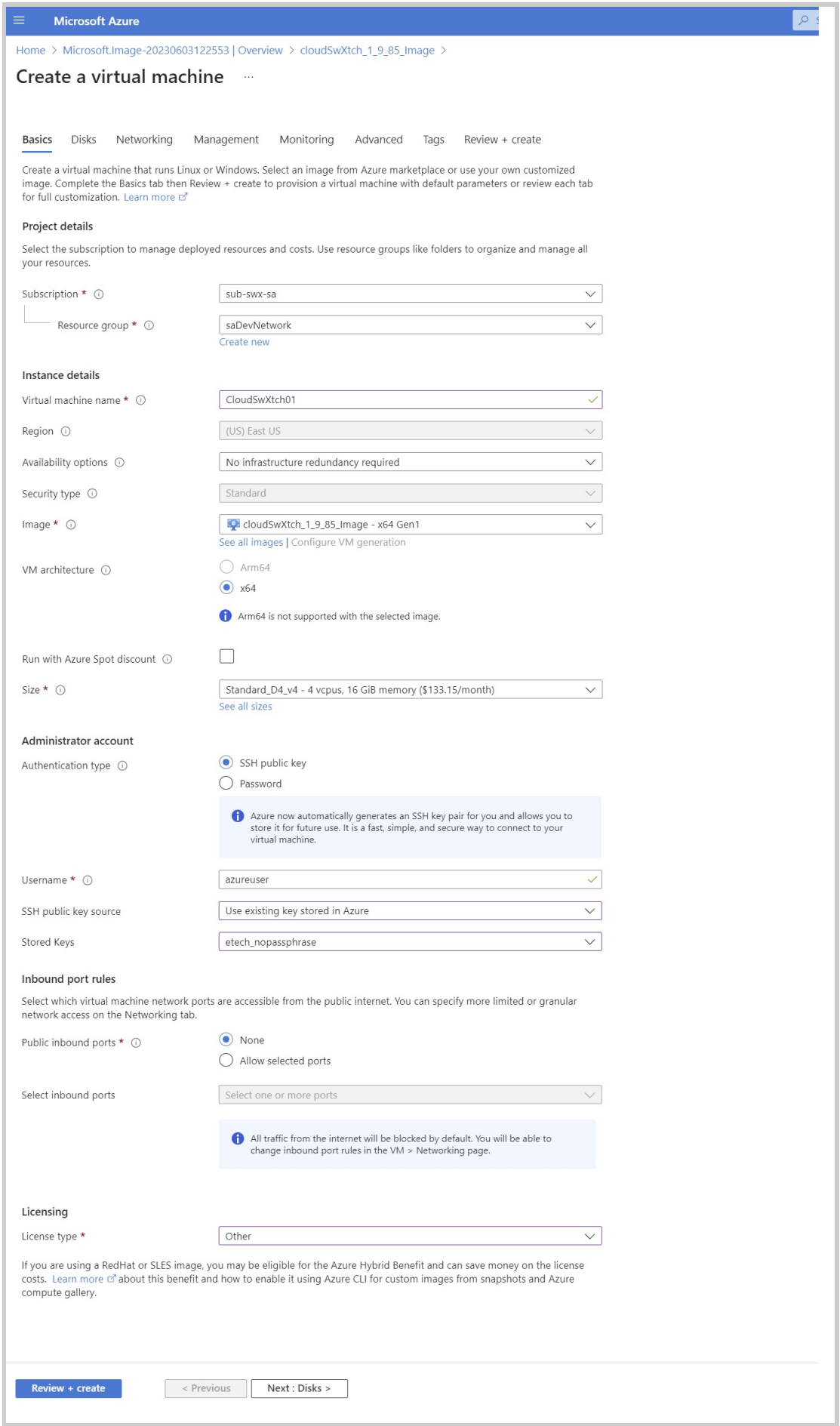
2. Select the image with the cloudSwXtch version you require.

3. Select **Create VM**.



4. Fill out the **Create Virtual machine** form like below:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



a. Set the **subscription** and **Resource Group** for where you want the cloudSwXtch instance to be located.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

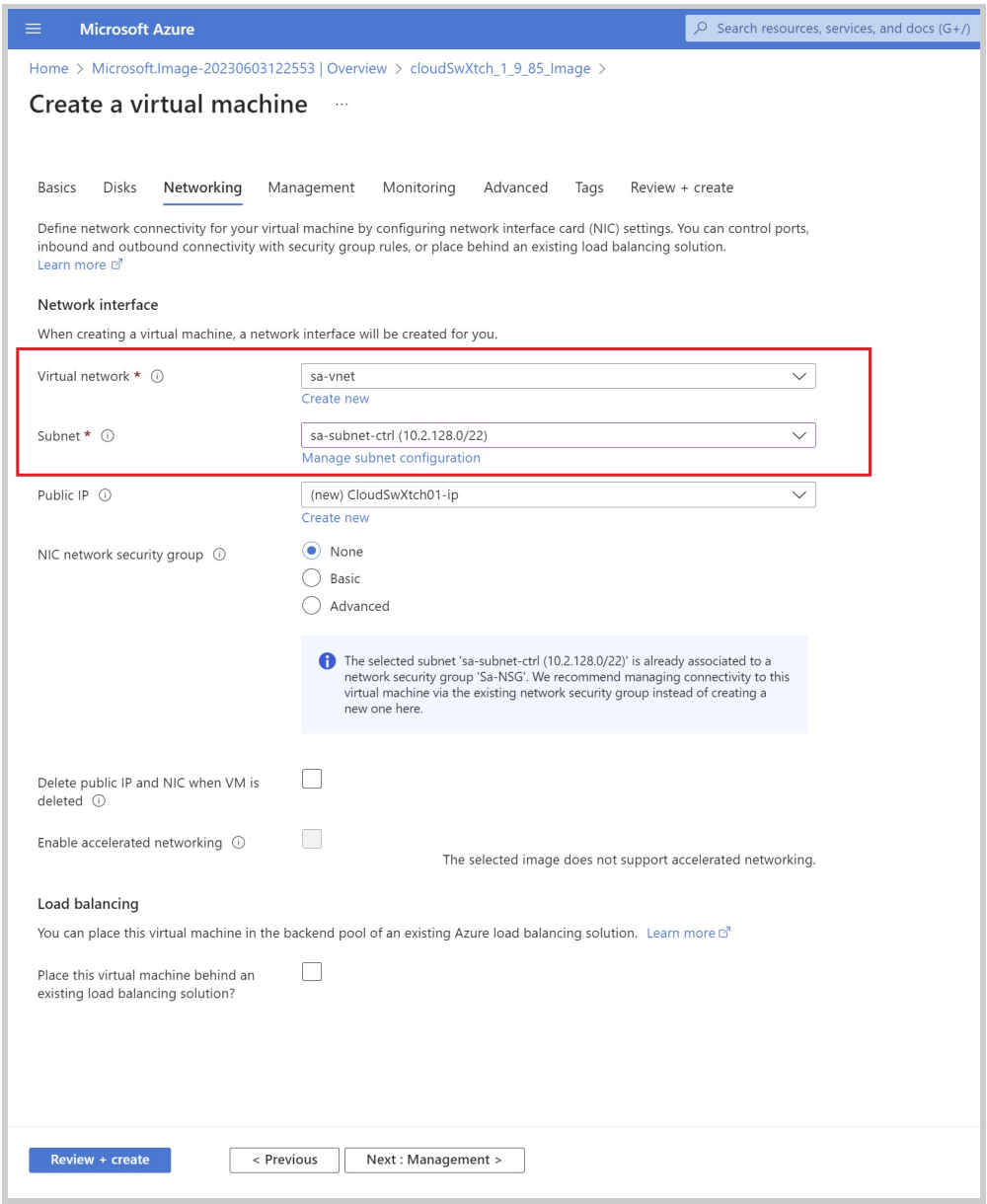
b. Name the Virtual Machine with a valid host name.

c. Select appropriate machine size. For recommendations based on features, endpoints, and bandwidth needs, read the Quotas article.

d. Use SSH for the authentication type. Enter your **SSH public key source.** Refer to ssh-keys-portal for details.

e. Set the **Licensing Type** to **Other.**

f. Navigate to the **Networking** tab and fill out the form like below:



i. Select the appropriate **Virtual Network**.

ii. Select the appropriate control subnet.

g. Navigate to other tabs as desired and enter in information as preferred. For example, some installations expect **Tags** to be entered.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

h. Select **Review + Create**.

i. When validation passes, select **Create**.

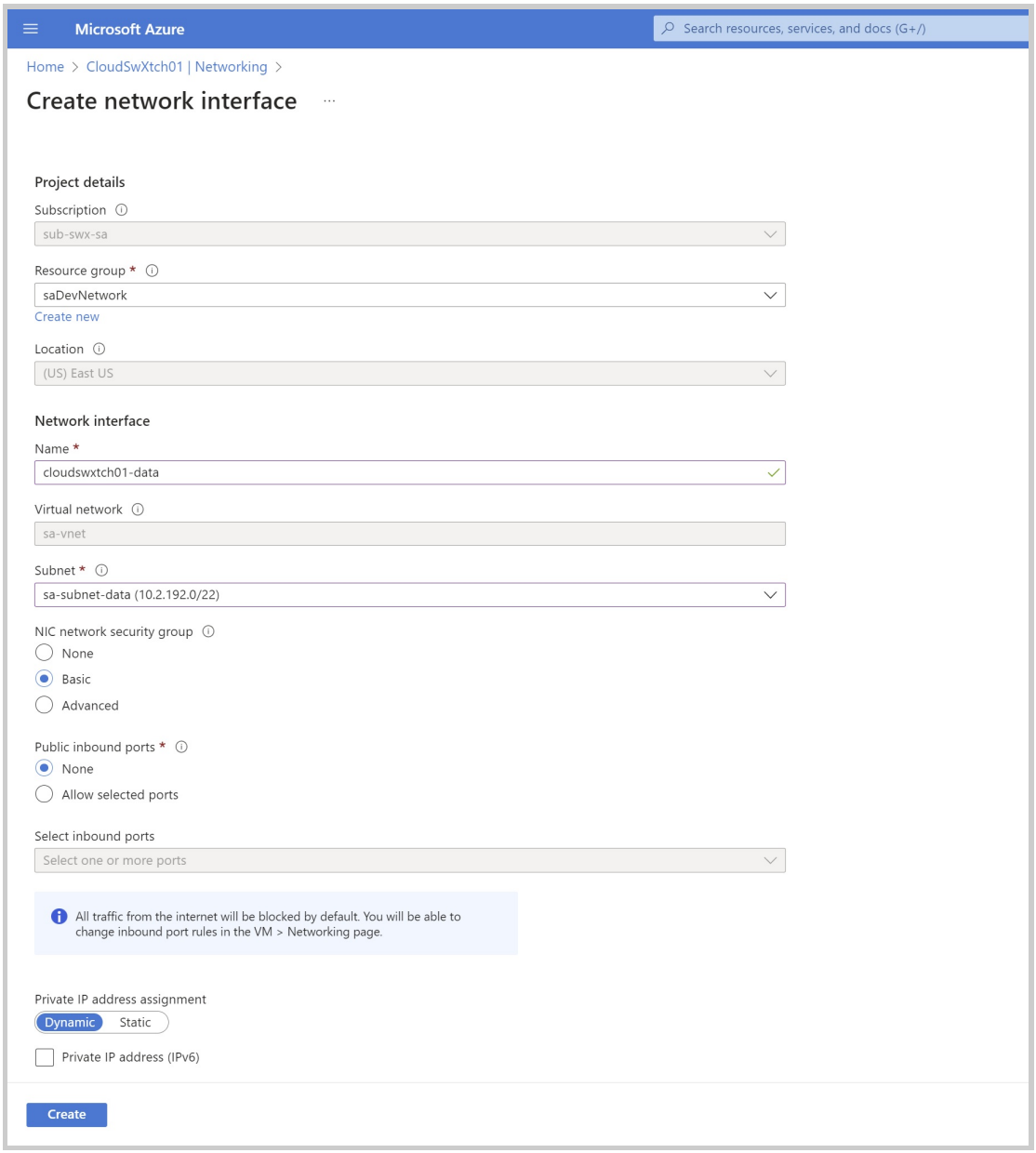
5. When the deployment is complete, select **Go to Resource**.

a. Select **Stop** to stop the VM.

6. Navigate to **Networking**.

7. Select **Attach network Interface**.

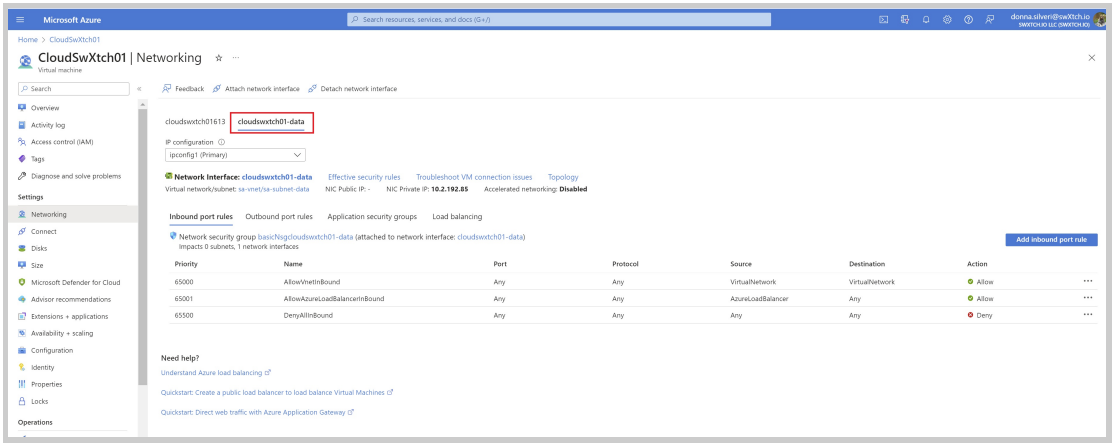
8. Select **Create and attach Network** and enter in data into the form to add a new NIC like shown.



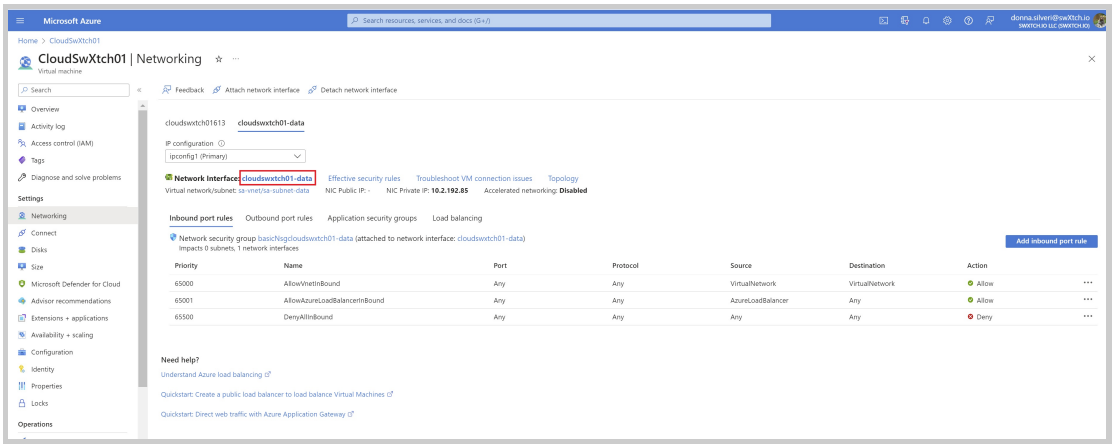
9. Select **Create**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

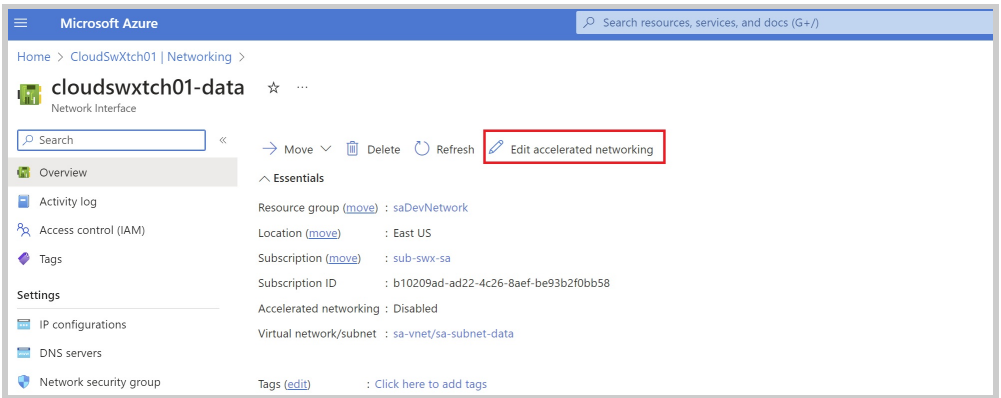
10. When it is done, refresh the screen. **There should now be a control and data interface.**



11. Select the data **Network Interface**.

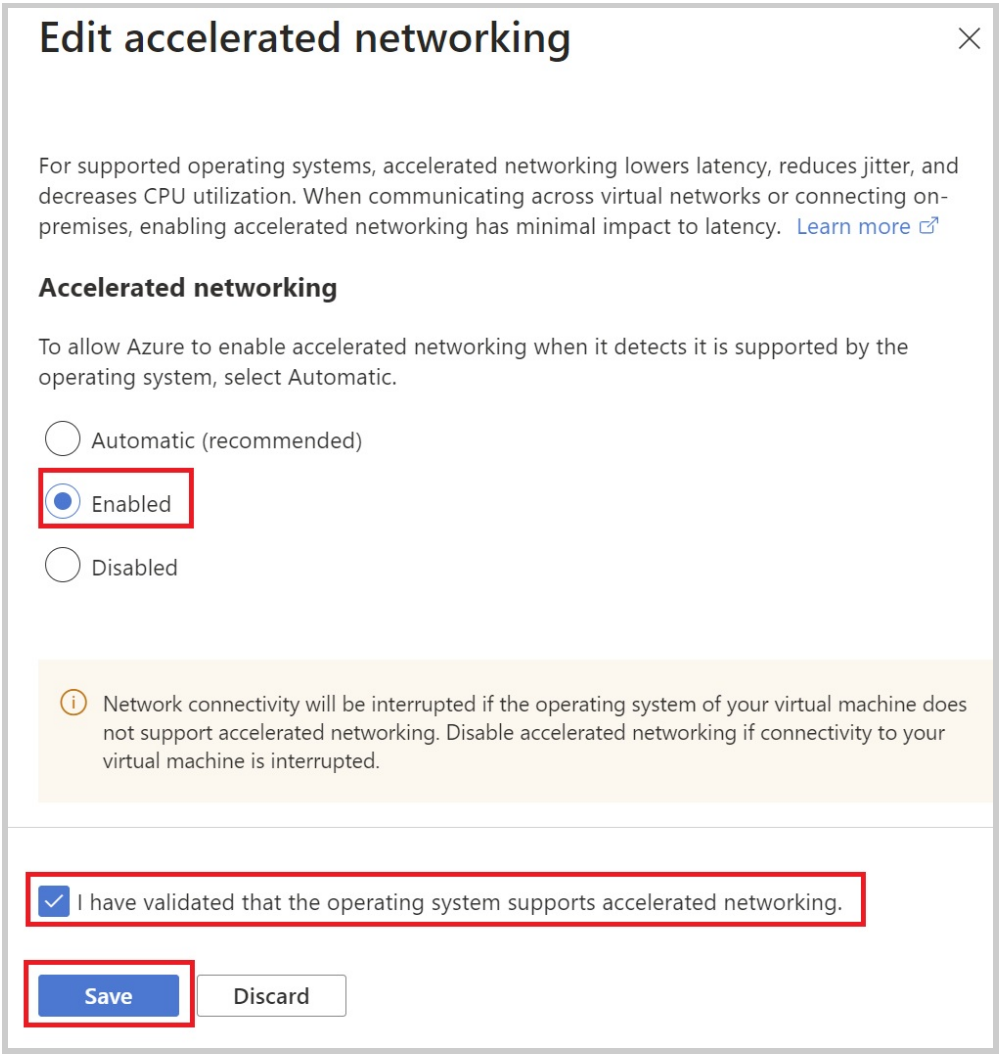


a. Select **Edit accelerated Networking**.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

b. A new window will display.

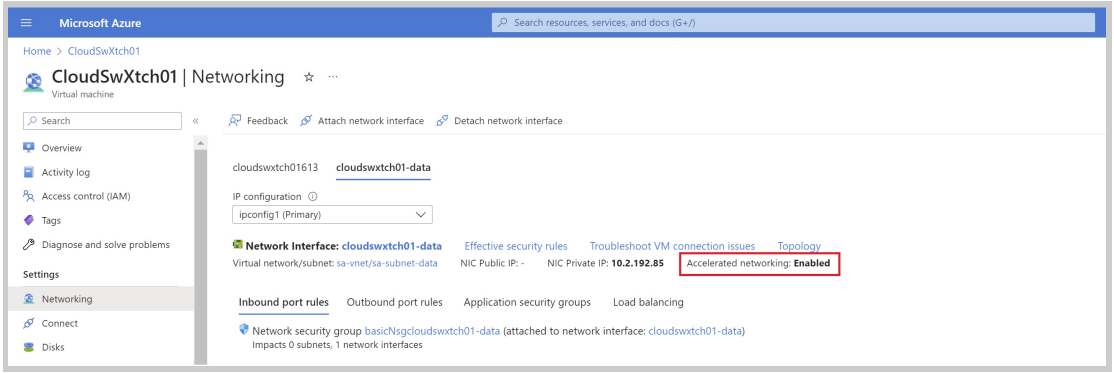


c. Select **Enabled**.

d. Check the agreement.

e. Select **Save**.

12. Refresh page and navigate back to Networking data tab to validate that Accelerated networking is **Enabled**.



13. Start the newly created cloudSwXtch VM.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

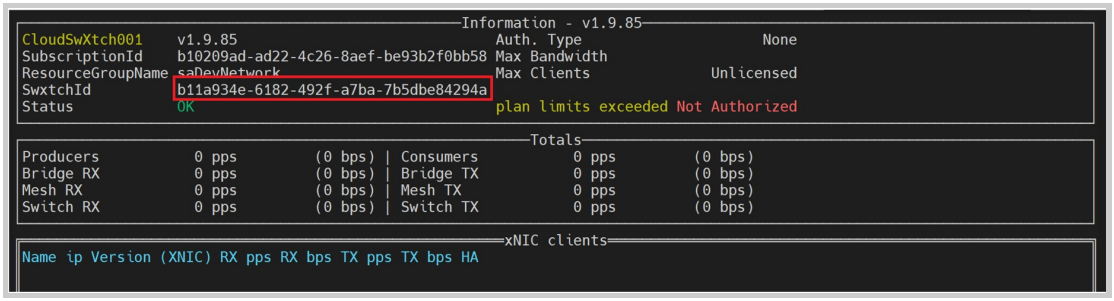
STEP FIVE: License the cloudSwXtch

1. Log onto the newly created VM.
2. Run this command:

   Text

   ```
   Bash
   sudo /swxtch/swxtch-top dashboard --swxtch localhost
   ```

3. **The swXtch-top dashboard will display.**



4. Copy the "**SwxtchId**" and email it to **support@swxtch.io** requesting a license.
5. When you receive the license file, **upload** it onto the cloudSwXtch VM.
6. Move the license.json file to the /swxtch directory using the following command replacing user with the appropriate value:

   Text

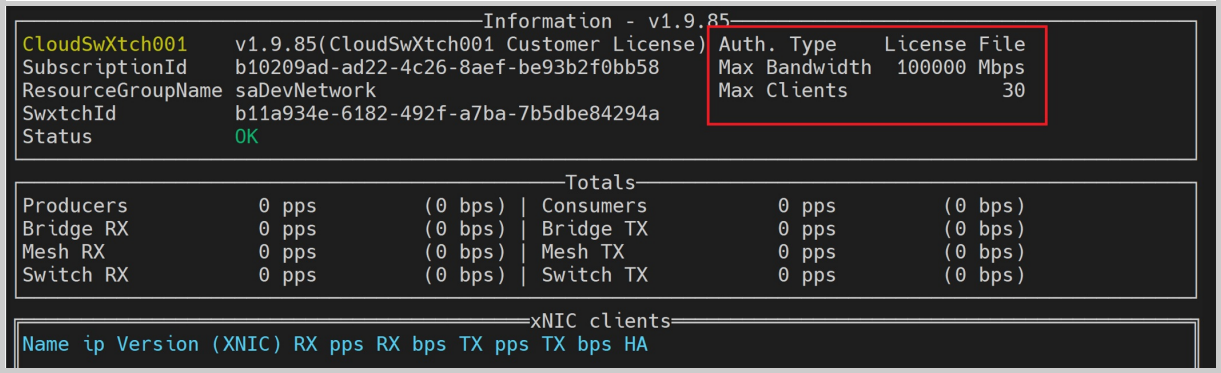
   ```
   Bash
   sudo mv /home/<user>/license.json /swxtch
   ```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

7. Reboot the cloudSwXtch and run swxtch-top again or journal to check the license took place:

Text

```
Bash
sudo journalctl  -u swxtch-ctrl.service -f -n 500
```



The cloudSwXtch is ready for use. IMPORTANT: Each client that is expected to get traffic from the cloudSwXtch will need an xNIC installed. See Installing xNIC for next steps in preparing clients (producers and consumers of Multicast).

> **Prerequisite: Installing Dependencies for Air-Gapped Clients**
>
> Before installing the xNIC, users will need to install the necessary packages on their air-gapped clients. Failure to complete this prerequisite will result in an unsuccessful xNIC deployment. To learn more, see **How to Install xNIC Dependencies in an Air-Gapped Environment**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

cloudSwXtch on GCP

> **WHAT TO EXPECT**
>
> In this article, users will find links to articles on deploying a cloudSwXtch on Google Cloud Platform (GCP).

Currently, there is two ways of deploying a cloudSwXtch on GCP:

- From the Google Cloud Marketplace
    - **Note**: This method will require users to contact swXtch.io for a license.
- Cloud agnostic cloudSwXtch VM Install
    - **Note**: This method will require the user to already have a Virtual Machine installed with Ubuntu 20.04 that adheres to all the **cloudSwXtch System Requirements**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install cloudSwXtch via GCP Marketplace

WHAT TO EXPECT

In this article, users will learn how to deploy a cloudSwXtch instance via the Google Cloud Platform (GCP) Marketplace.

- Prerequisites
- Step One: Navigate to cloudSwXtch in the GCP Marketplace
- Step Two: Configure cloudSwXtch deployment
- Step Three: Add SSH Key(s)
- Required Step for BYOL: Contact swXtch.io for a License

Prerequisites

A user needs the following to deploy a cloudSwXtch via the GCP Marketplace:

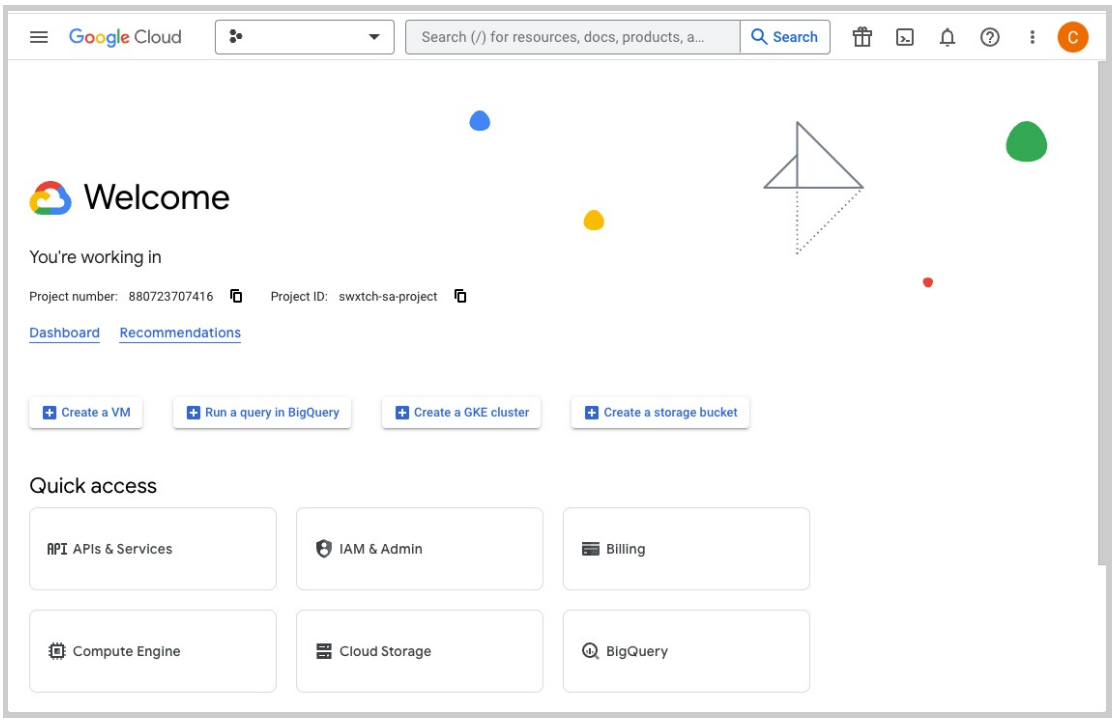
- An existing **Deployment Service Account** established in their Google Cloud Console project. Creating a new account is further detailed in Step Two.
- **Two (2) VPCs available** -- one for the Control NIC and another for Data. All cloudSwXtch installations require 2 NICs. Please review GCP documentation on how to create and modify VPCs.
    - **Note**: GCP does not allow you to have 2 NICs on the same VPC. It will return an error message. The Control and Data NIC must be assigned their own VPC.

Please review cloudSwXtch System Requirements for additional prerequisites.

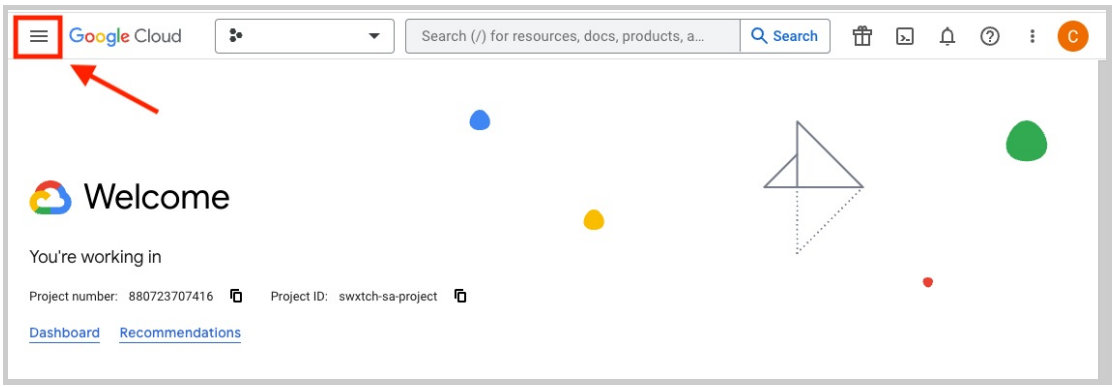
Step One: Navigate to cloudSwXtch in the GCP Marketplace

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

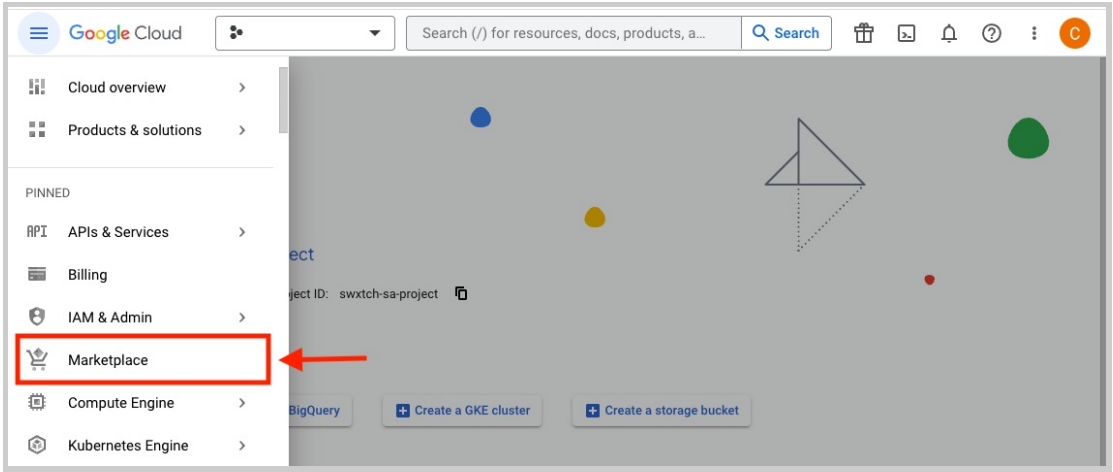
1. Log into the **Google Cloud Console**.



2. Navigate to the **Google Cloud Marketplace** using the **Navigation** menu at the top left corner.

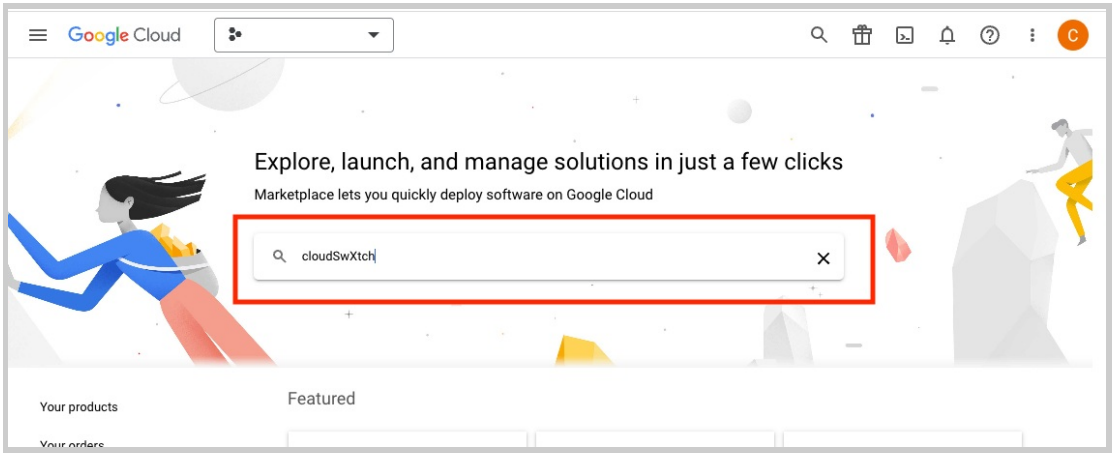


3. Select **Marketplace**.

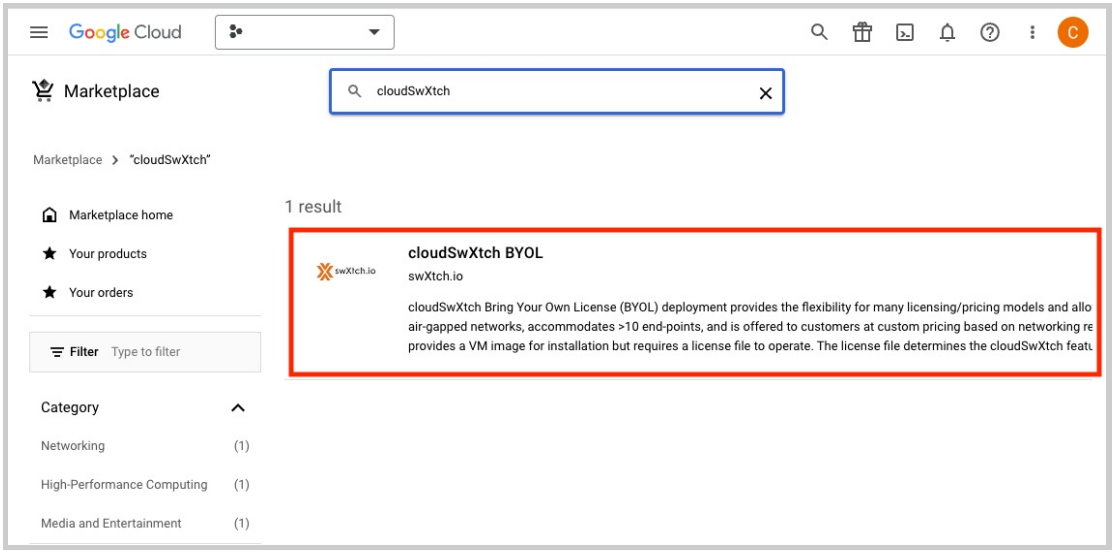


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

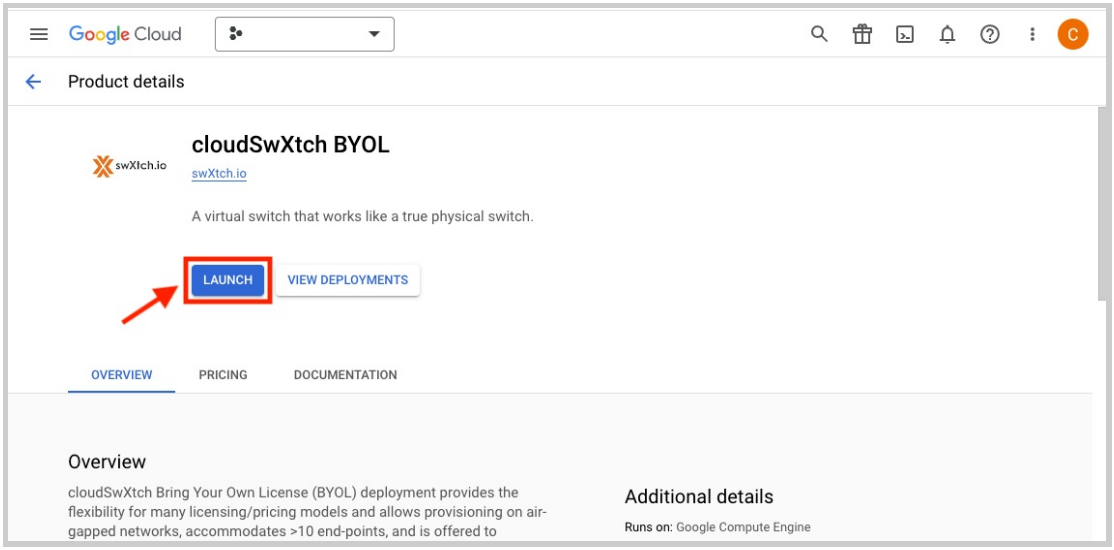
4. Search for **cloudSwXtch**.



5. Select the product, **cloudSwXtch BYOL**.



6. Click **Launch** to open the deployment configuration page.



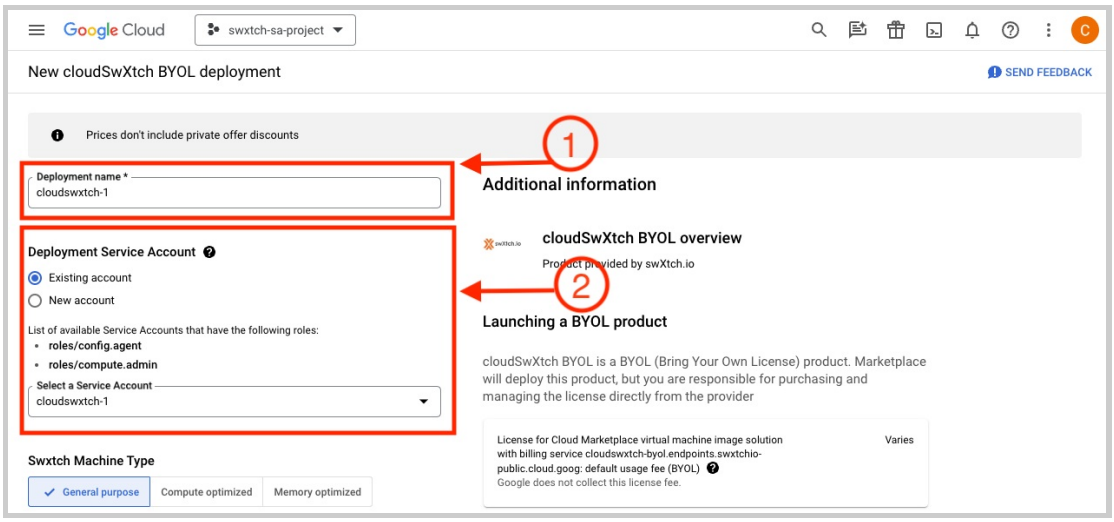
Enabling APIs

After hitting launch, a new window might open asking you to enable Google APIs. You must enable the suggested APIs to continue.

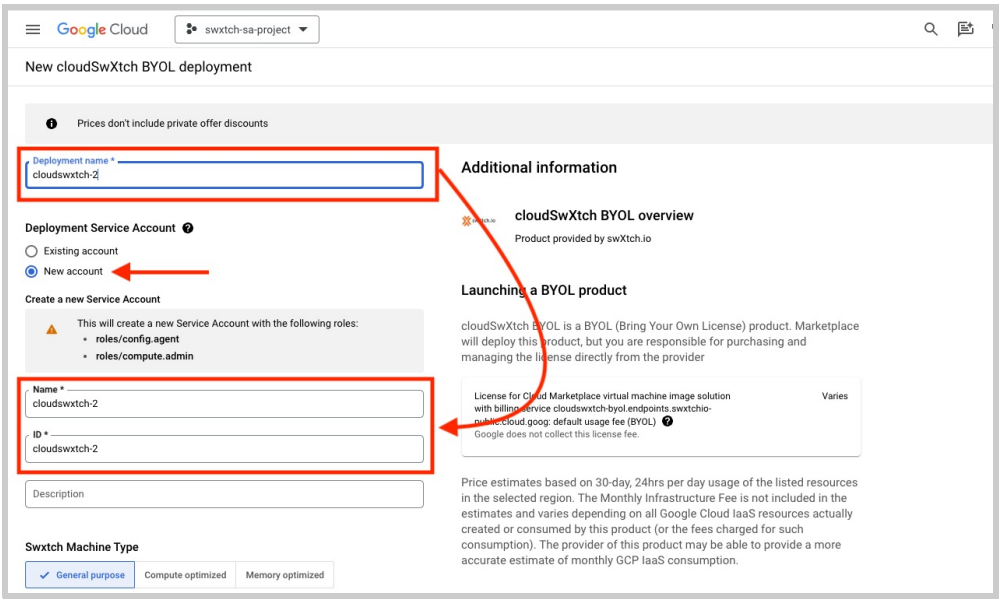
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Step Two: Configure cloudSwXtch Deployment

1. Enter a **Deployment name** for your cloudSwXtch.
2. Select an **Existing account** under Deployment Service Account.



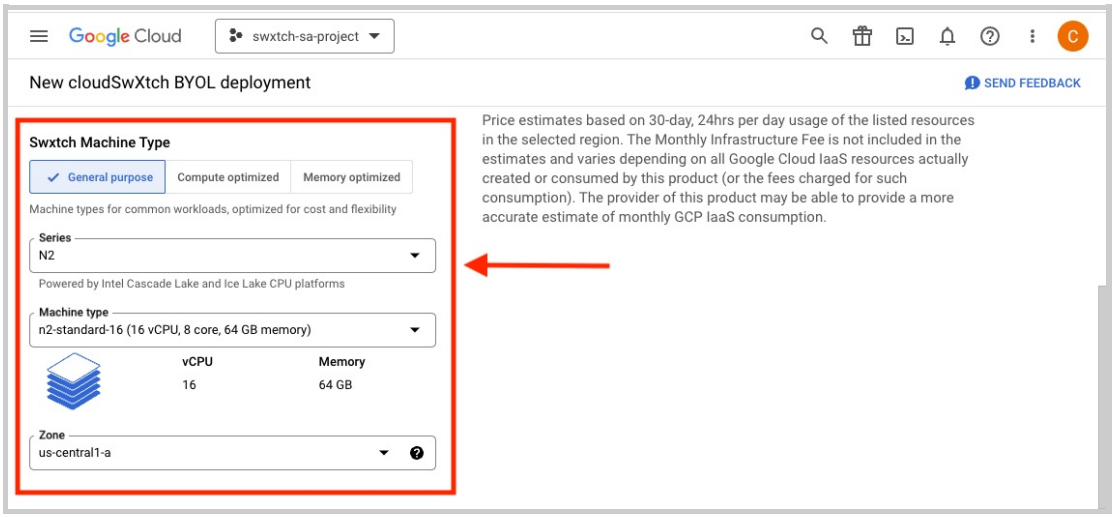
   a. If you do not have a **Deployment Service Account**, select **New Account**. You will need permissions from your Project IAM Admin (or someone with the resourcemanager.projects.setiampolicy permissions) to allow you to create a new Deployment Service Account.
   b. Enter the same name used for your cloudSwXtch Deployment for your New Account Name and ID. The names must match and only use lowercase letters and numbers.
   c. **The account will be created after you deploy your cloudSwXtch.** Once an account is created, users without permissions can use it as an Existing account option.



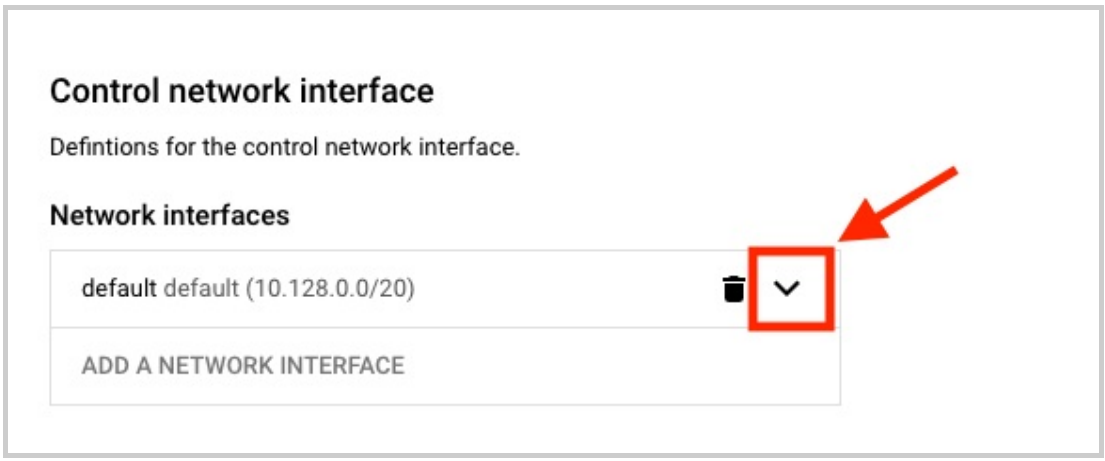
3. Under **SwXtch Machine Type**, confirm that **N2** is selected under **Series**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. Confirm your sizing under **Machine Type**. The **default** is set to **n2-standard-16,** which is 16 core. **A cloudSwXtch must have a minimum of 8 cores**. For cloudSwXtch Sizing guidelines, see cloudSwXtch System Requirements.
5. Confirm your desired **Zone**.

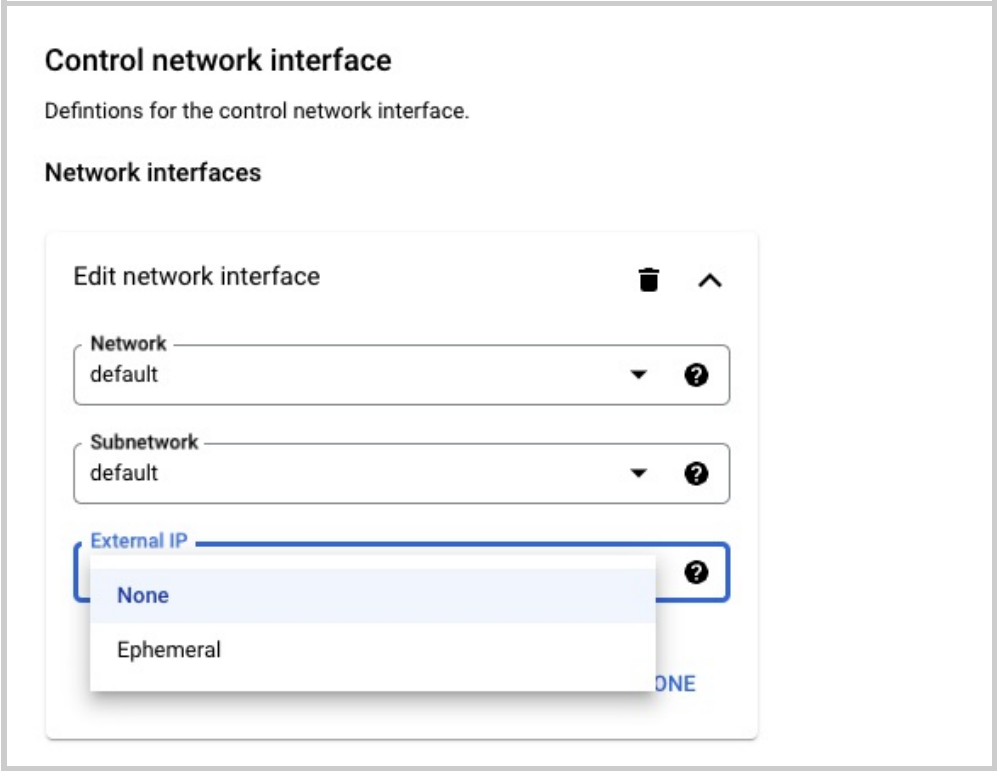


6. Use the dropdown arrow under **Control network interface** to open the configuration panel. If your default subnet is already selected and you do not wish to set a public IP, continue you on Step 10.

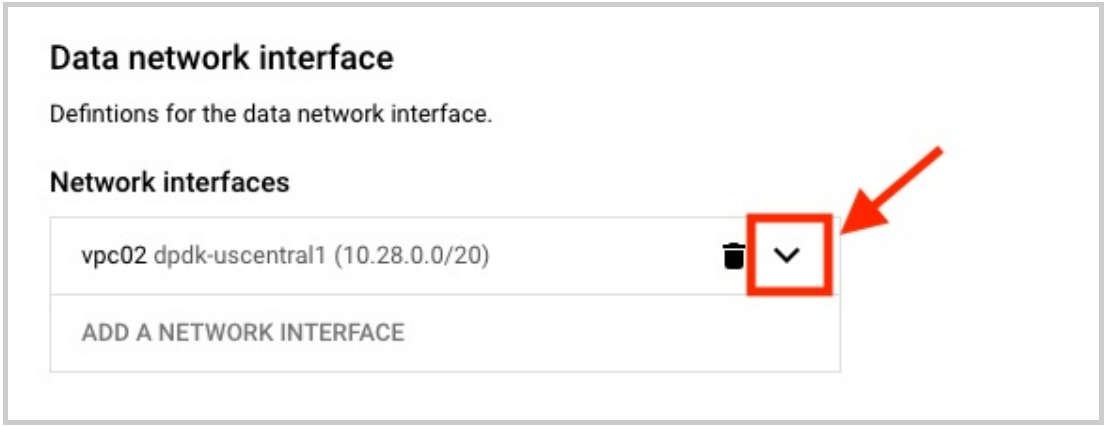


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

7. Select a **Network** and **Subnetwork**. This subnet will be used for your control plane communications.
    a. **Optional**: Users can select Ephemeral under External IP if they wish for their Control NIC to be assigned a randomized public IP address.

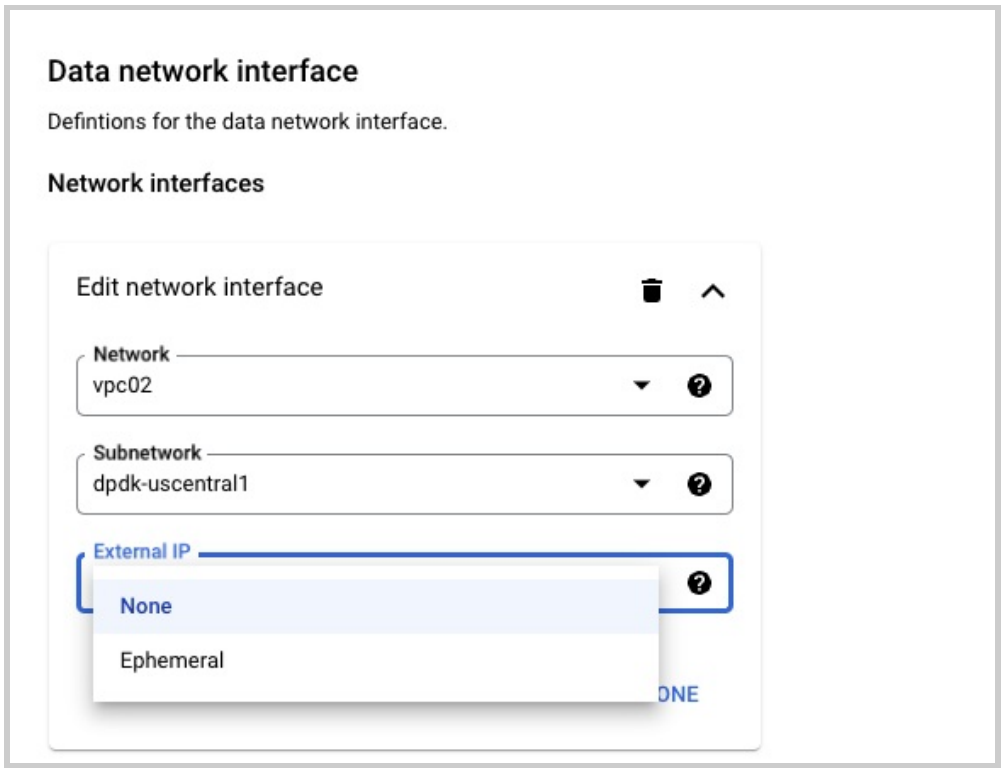


8. Click **Done** when you are happy with your selections.
9. Use the dropdown arrow under **Data network interface** to open the configuration panel.

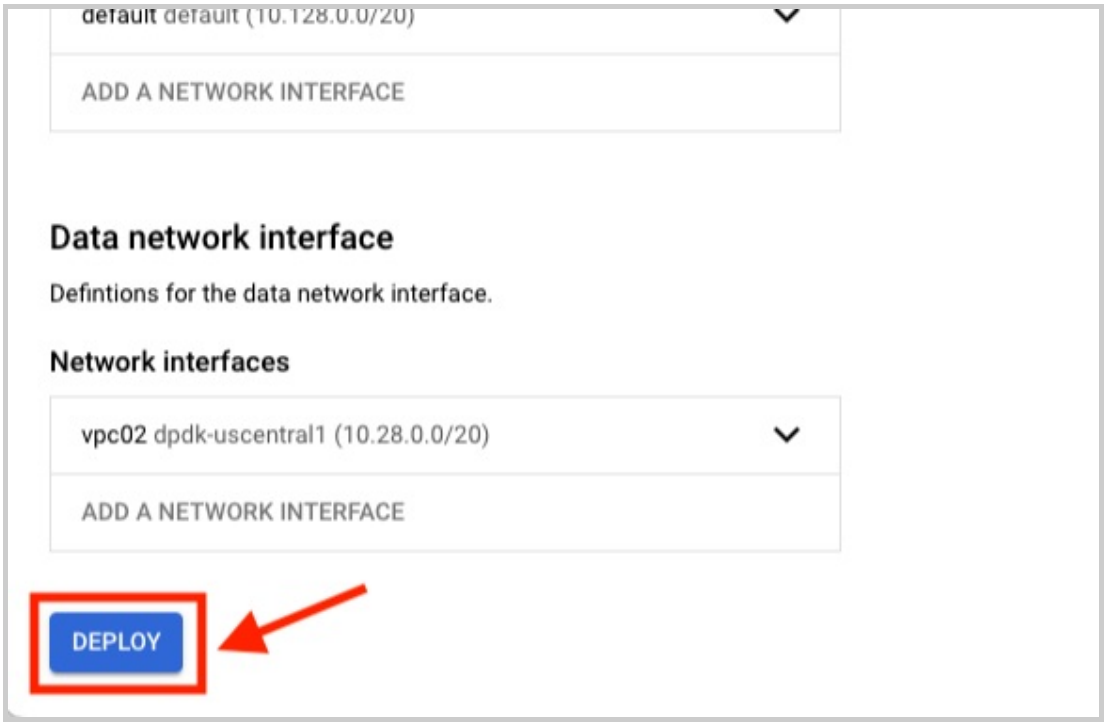


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

10. Select a **Network** and **Subnetwork**. This 2nd subnet will be used for your data plane communications and should've been created before starting the deployment process.
    a. **Please note**: The control and data NICs cannot share a subnet. They must have separate subnets.
    b. **Optional**: Users can select Ephemeral under External IP if they wish for their Data NIC to be assigned a randomized public IP address.

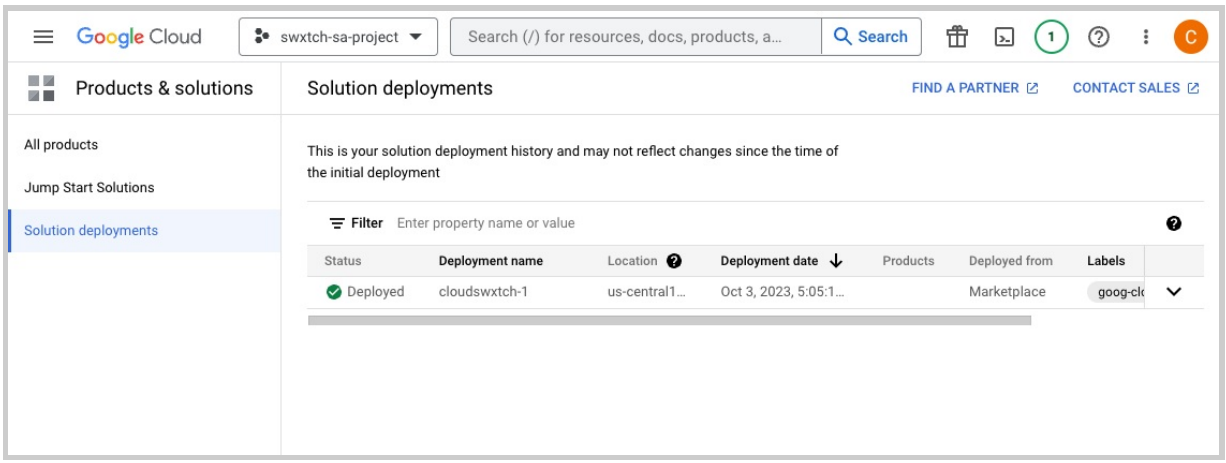


11. Click **Done** when you are happy with your selections.
12. Click **Deploy**.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

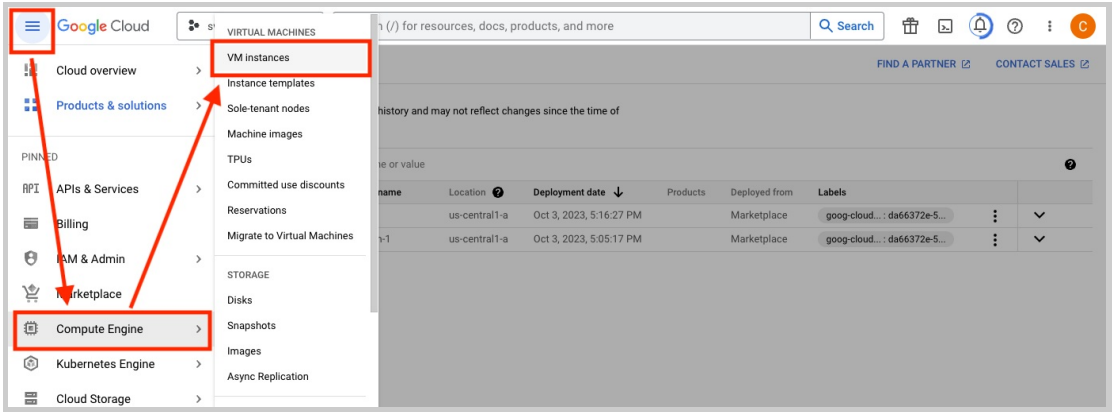
Your cloudSwXtch instance will now be deployed.



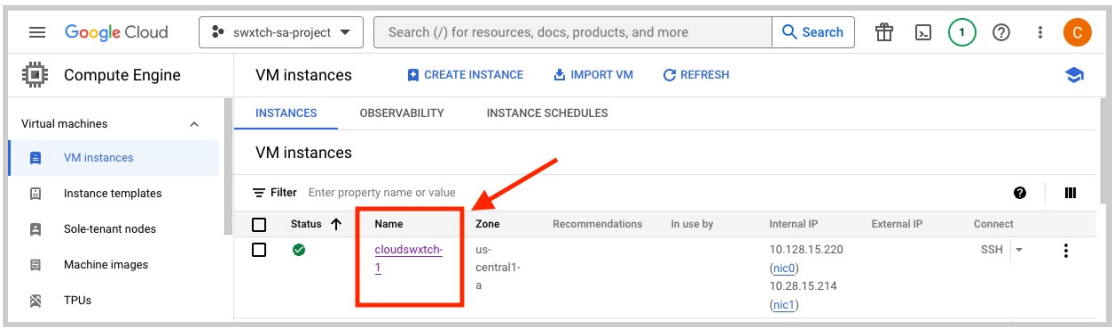
Step Three: Add SSH Key(s)

In order to access your Google Cloud VM instance, you will need to add an SSH key to your cloudSwXtch deployment.

1. Click on the **Navigation** menu on the left hand corner, highlight **Compute Engine** and select **VM instances**.

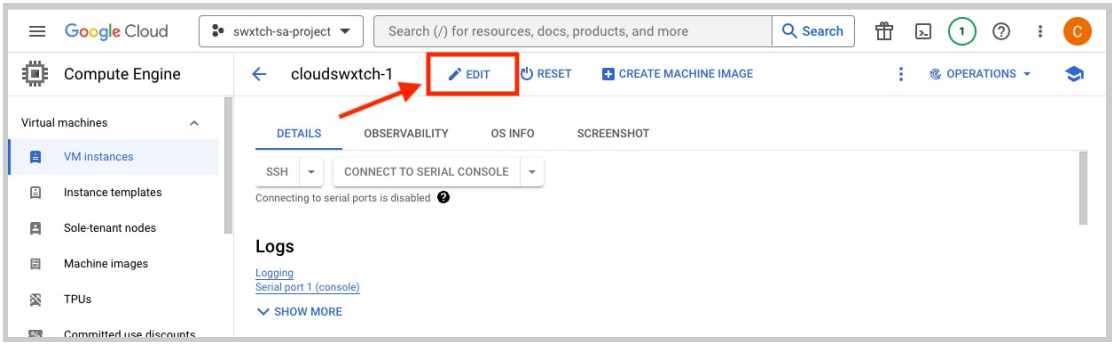


2. Select the **name of your cloudSwXtch deployment** to open its configuration page.

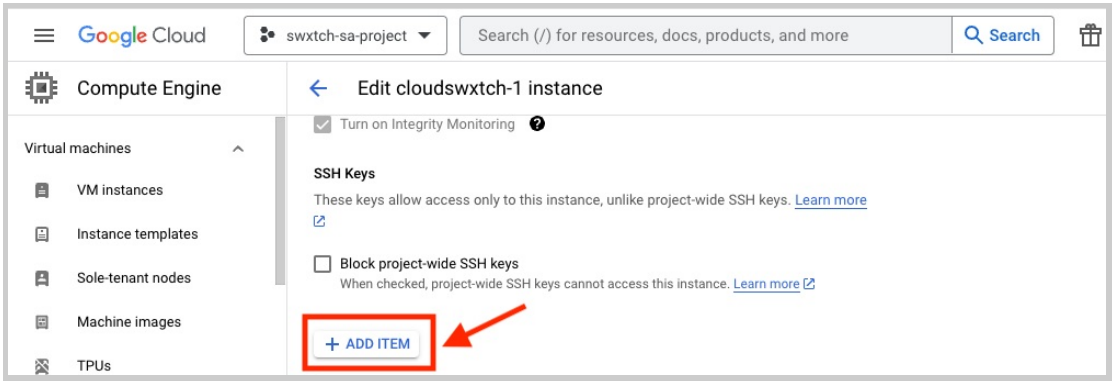


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

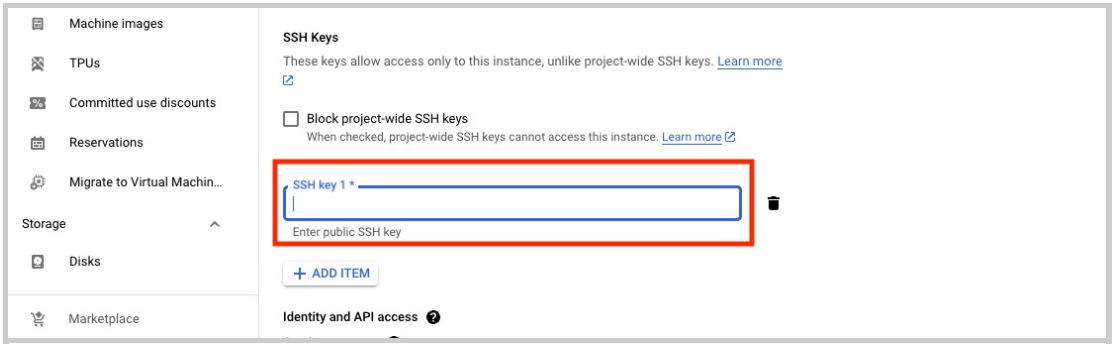
3. Choose **Edit** next to your cloudSwXtch deployment name.



4. Scroll down to **SSH Keys** under **Security and Access**.
5. Click **+Add Item**.



6. Enter your **SSH key**. You can add multiple.



7. Hit **Save** at the bottom of the page.

Upgrading your cloudSwXtch Instance

After deployment, it is recommended to update your cloudSwXtch instance to latest. Please refer to the **Upgrading cloudSwXtch** article for more information.

Required Step for BYOL: Contact swXtch.io for a license

Users deploying a BYOL instance of cloudSwXtch will need to contact swXtch.io for a license file. For more information, see How to License a cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

cloudSwXtch on OCI

WHAT TO EXPECT

In this article, users will find links to articles on deploying a cloudSwXtch in Oracle Cloud Infrastructure (OCI).

Currently, there are only two ways of deploying a cloudSwXtch on OCI:

- From the Oracle Cloud Marketplace
    - **Note**: This method will require users to contact swXtch.io for a license.
- Cloud agnostic cloudSwXtch VM Install
    - **Note**: This method will require the user to already have a Virtual Machine installed with Ubuntu 20.04 that adheres to all the cloudSwXtch System Requirements.

Please stay tuned for more information about alternative methods of installation.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install cloudSwXtch via OCI Marketplace

WHAT TO EXPECT

In this article, users will learn how to deploy a cloudSwXtch instance via the Oracle Cloud Marketplace.

- Step One: Navigate to cloudSwXtch in the Oracle Marketplace
- Step Two: Create Compute Instance
- Step Three: Attach Secondary VNIC
- Optional Step for BYOL: Contact swXtch.io for License

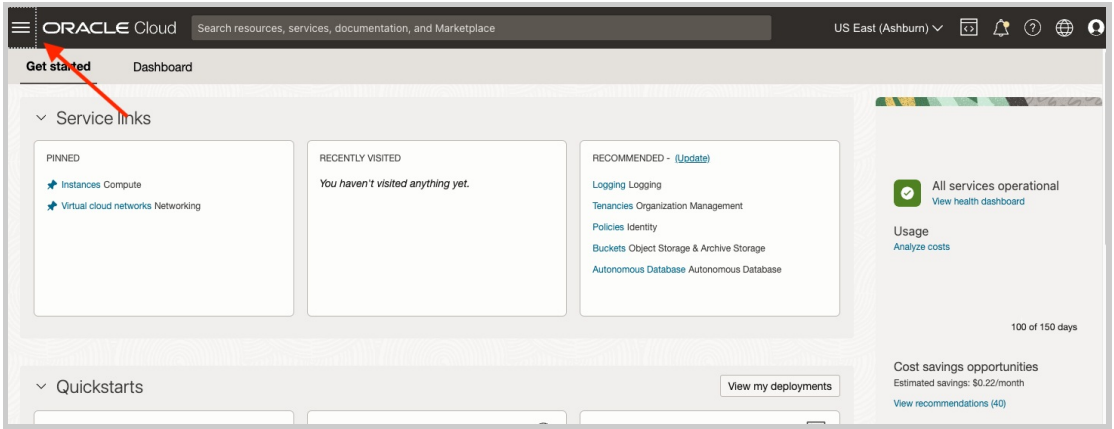
Please note: At this time, our only product offering in OCI is a BYOL instance of cloudSwXtch. This requires a user to contact swXtch.io for a license.

Prerequisites

A user should have a **Compartment** established in their Oracle Cloud console before they start to deploy a cloudSwXtch. For more information about compartments, please see the Managing Compartments page under Oracle Cloud Infrastructure Documentation.

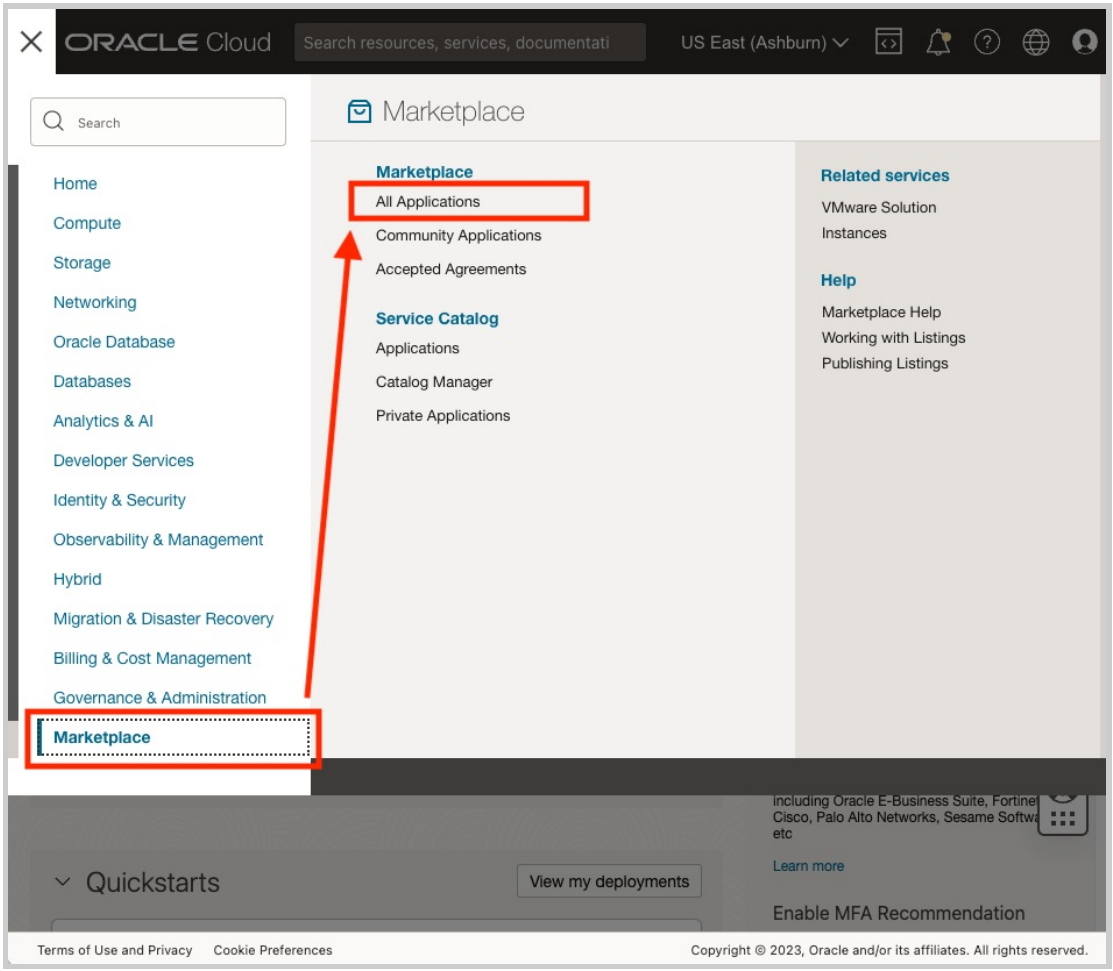
Step One: Navigate to cloudSwXtch in the Oracle Marketplace

1. Log into **Oracle Cloud**.
2. Navigate to the **Oracle Cloud Marketplace** using the **Navigation** menu at the top left corner.

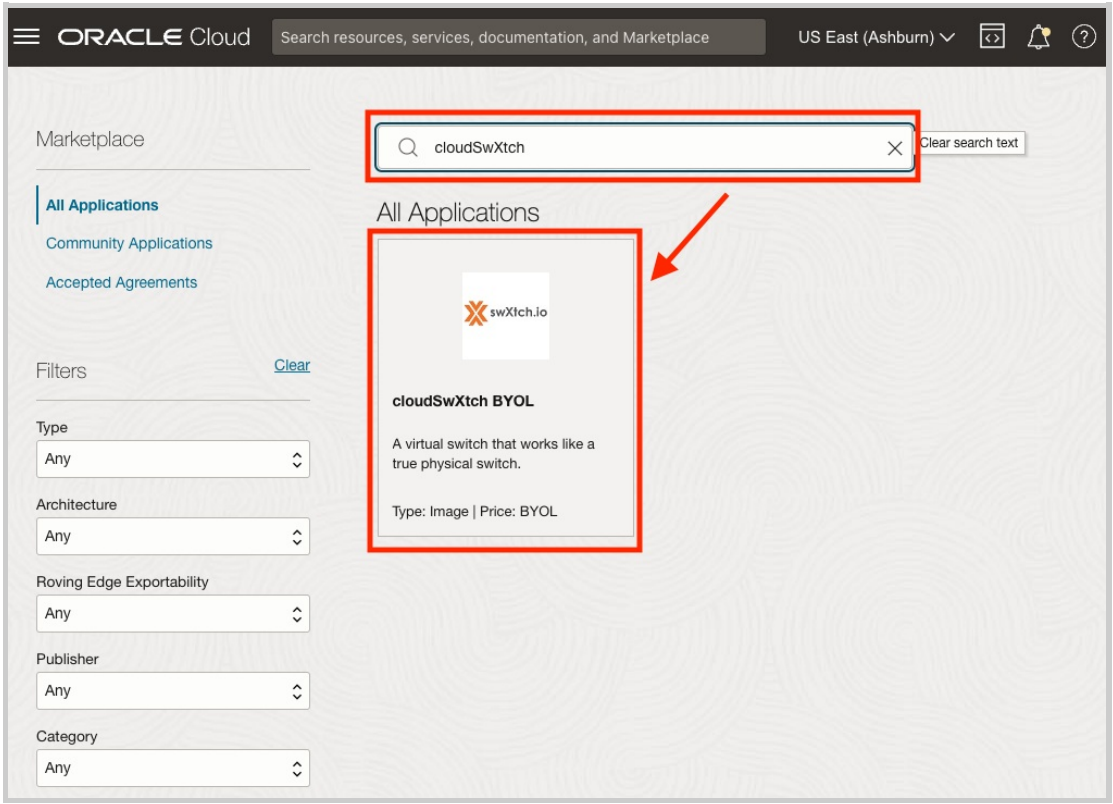


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. Select **Marketplace** and **All Applications**.

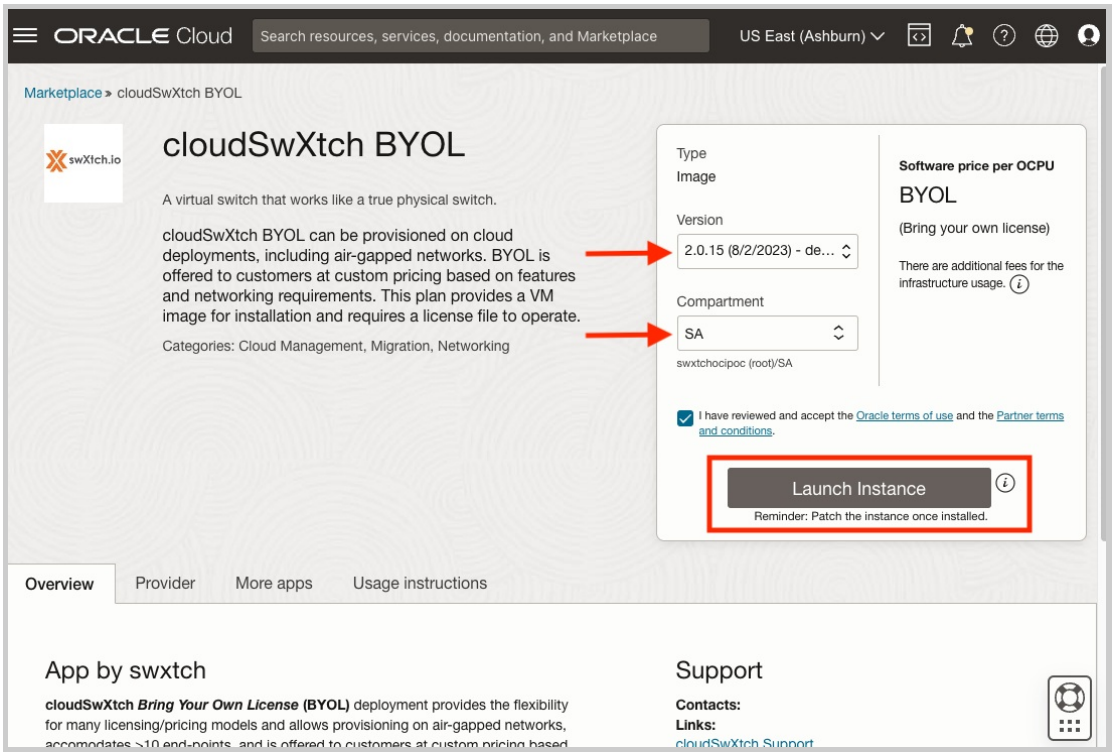


4. Search for **cloudSwXtch** and select the product, **cloudSwXtch BYOL**.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

5. Select the **Version** and the **Compartment** that you will like to use. It is best to use the default since it will be the most recent version.
6. Click **Launch Instance** when you're happy with your selections.

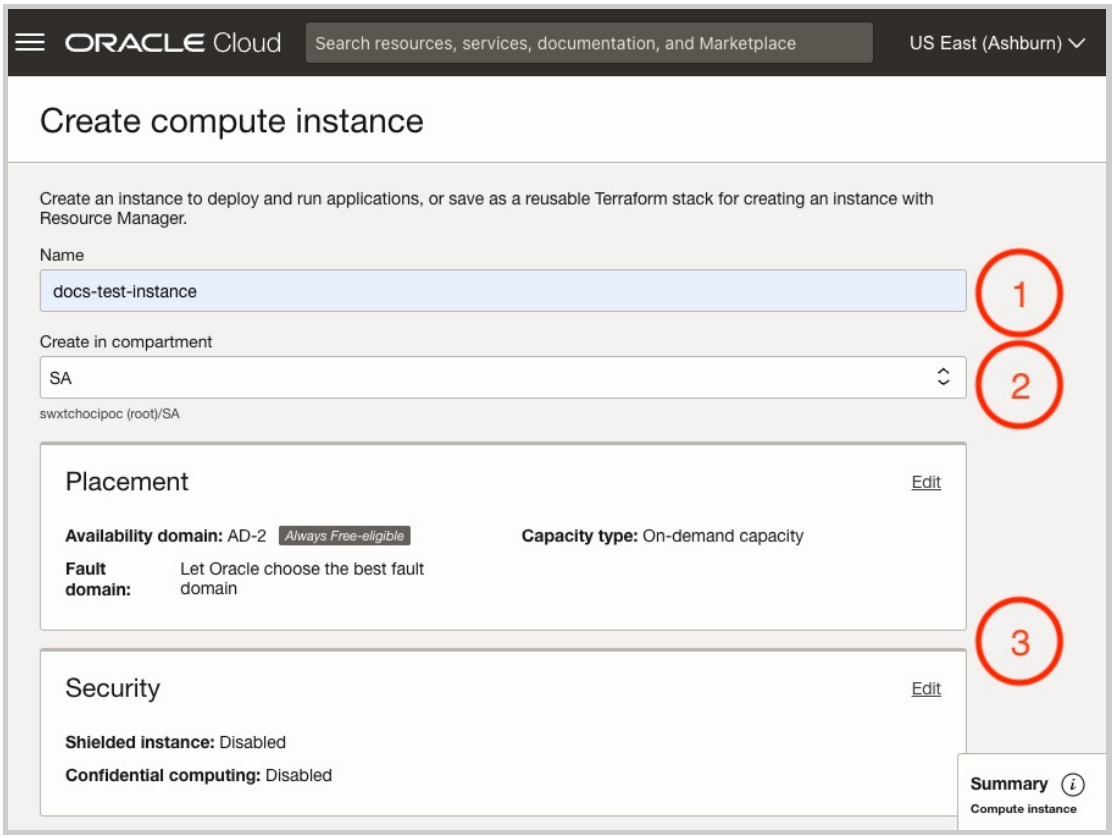


Step Two: Create Compute Instance

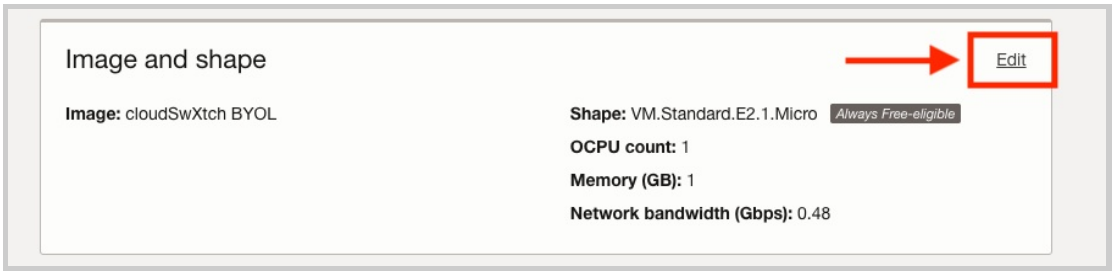
1. Give your **Compute Instance** a unique name.
2. Confirm that your desired **Compartment** is populated.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

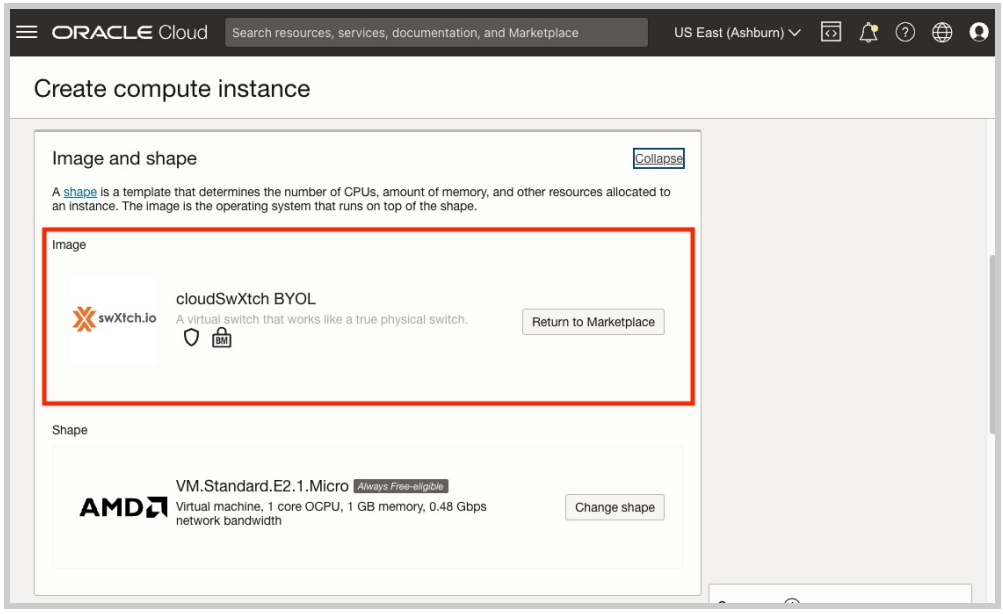
3. **Optional**: Edit selections for **Placement** and **Security**. This is dependent on a user's specific needs.



4. Select the **Edit** button for **Image and Shape**.

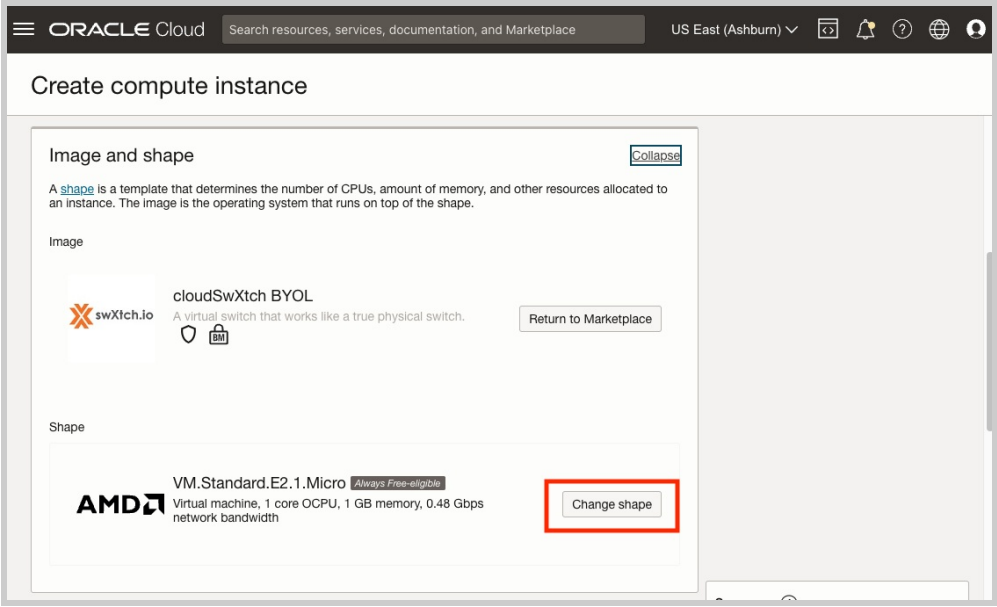


   a. Confirm that **cloudSwXtch BYOL** is selected for **Image**.

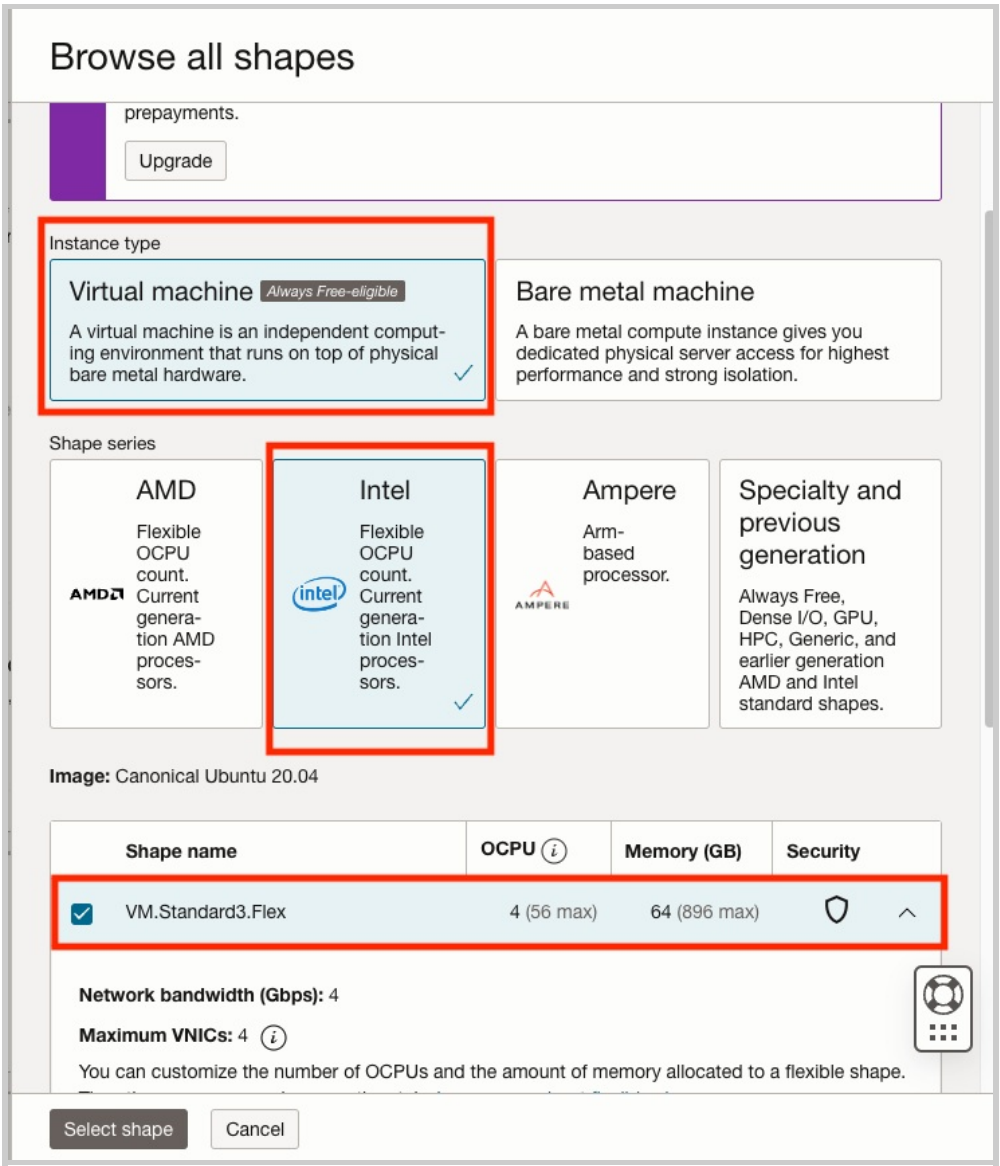


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

b. Click **Change Shape**.

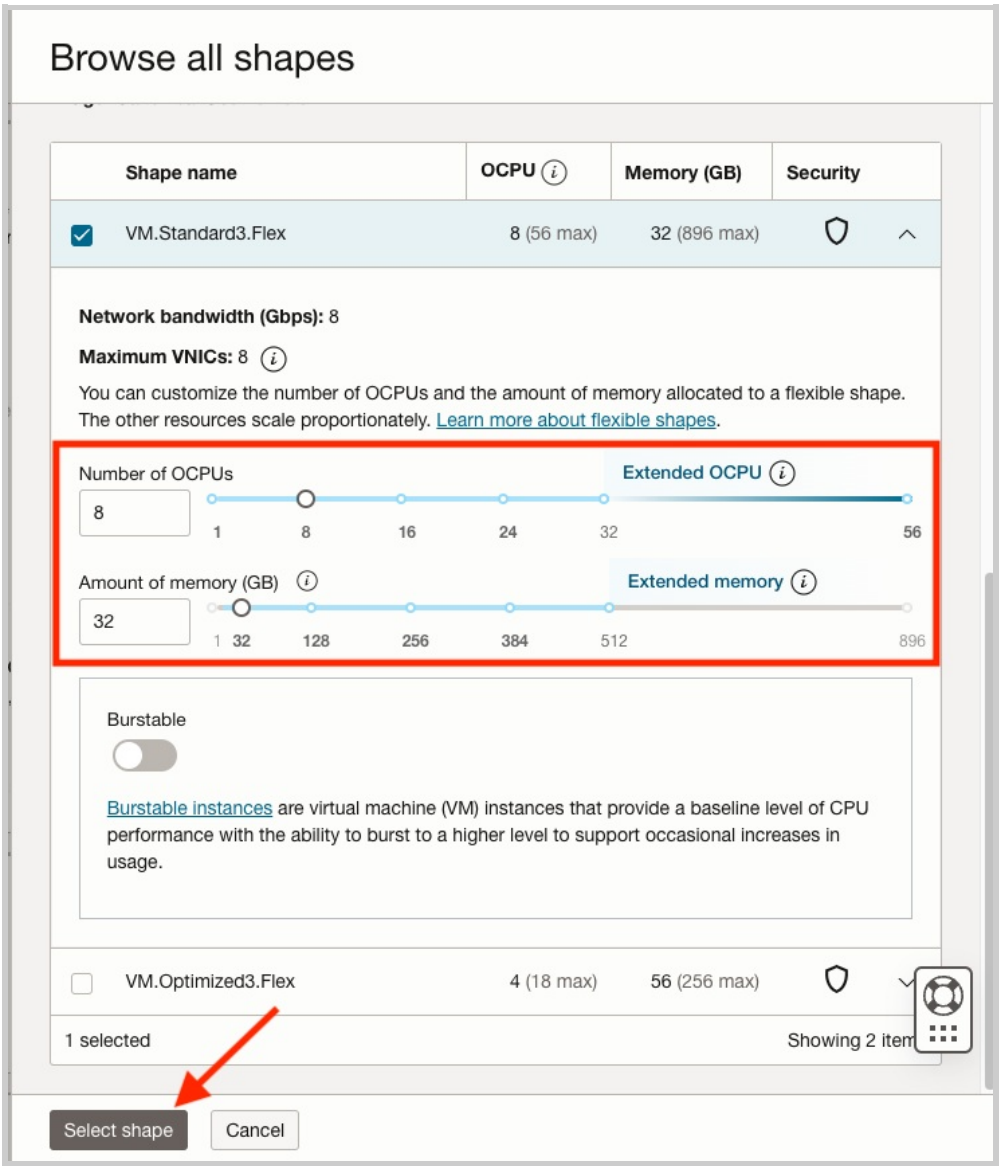


c. Choose **Intel** and **VM.Standard3.Flex**.



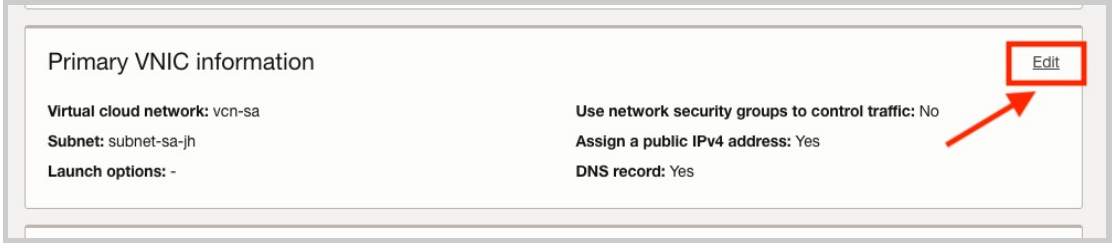
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

d. Configure the **Number of OCPUs** and **Amount of memory (GB)**. **Please note**: It is recommended to have **at least four (4) cores** for your cloudSwXtch instance. This is typical sizing for a small cloudSwXtch. **For more information on recommended sizing**, please see cloudSwXtch System Requirements.



e. Click **Select Shape** when you're happy with your selection.

5. Select the **Edit** button for **Primary VNIC information**.

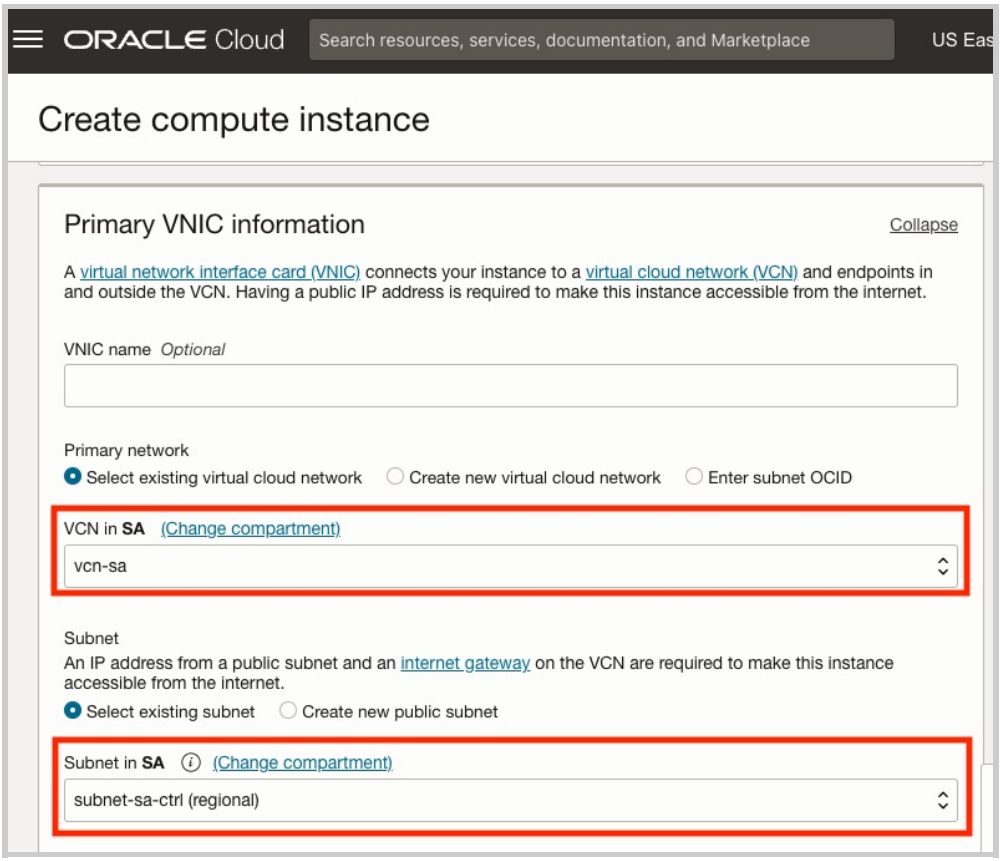


a. **Optional**: Add a name to your VNIC. If left blank, Oracle will assign it the name of your instance with a note that it is the Primary VNIC.

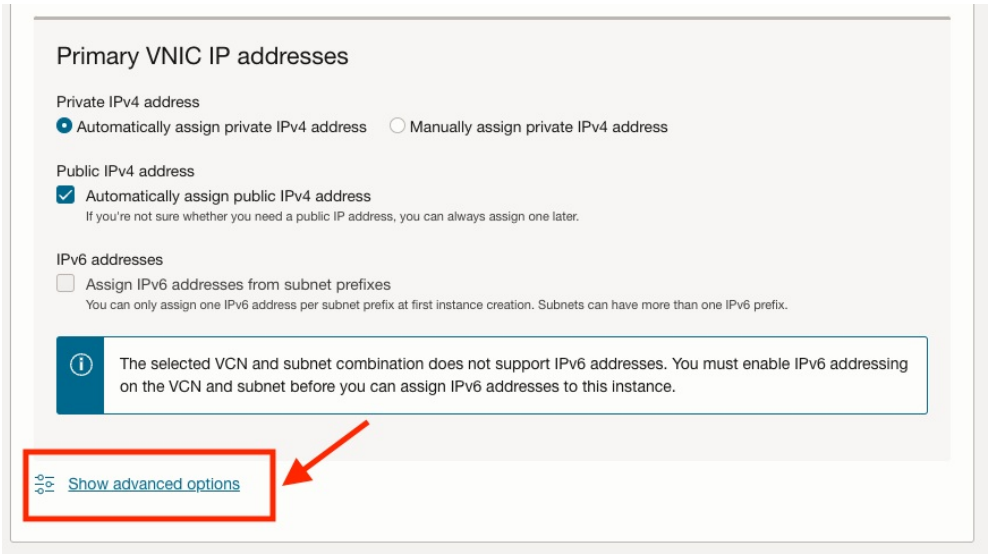
b. Assign a **VCN** to your **Primary VNIC**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

c. Select a **subnet**. **Please note**: This ctrl subnet will also be used for your secondary VNIC.

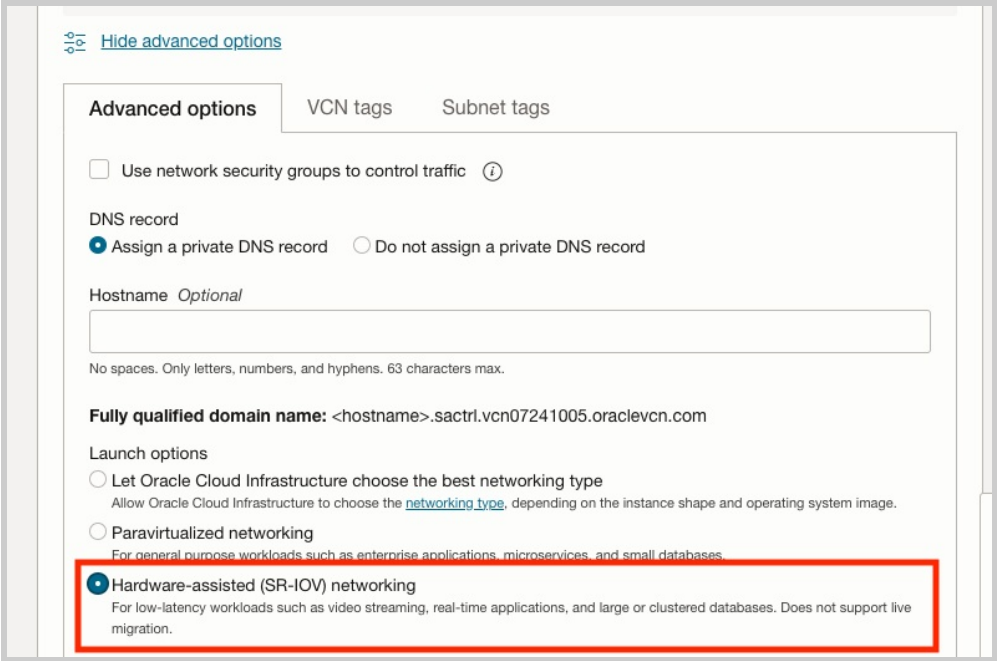


d. Click on **Show advanced options**.

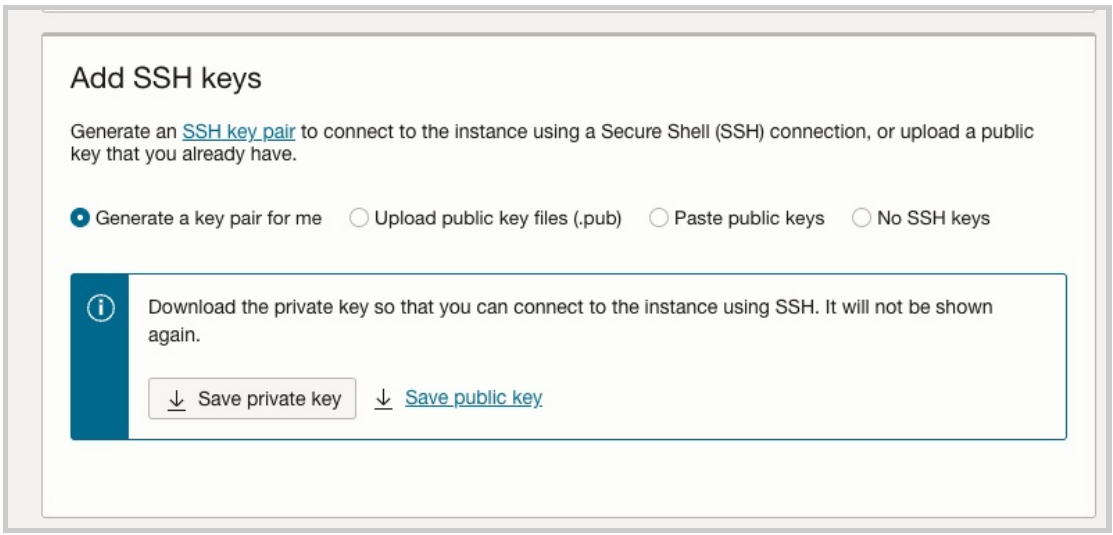


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

e. Select **Hardware-assisted (SR-IOV) networking** under **Launch** options.



6. Add an **SSH key**.



7. Hit **Create** button when you're happy with all of your selections.

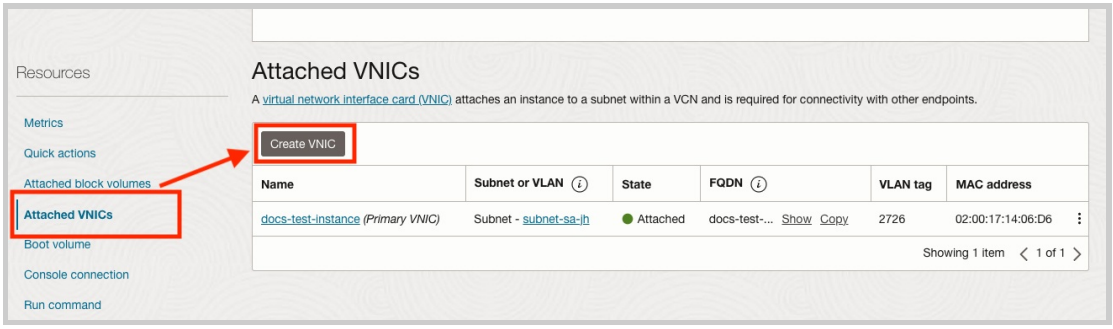
Step Three: Attach a Secondary VNIC

When deploying a cloudSwXtch, you will need two VNICs. Both can share a single subnet for control and data plane communications. In this step, we will walkthrough how to attach your secondary VNIC and how to manually add its IP to your cloudSwXtch instance.

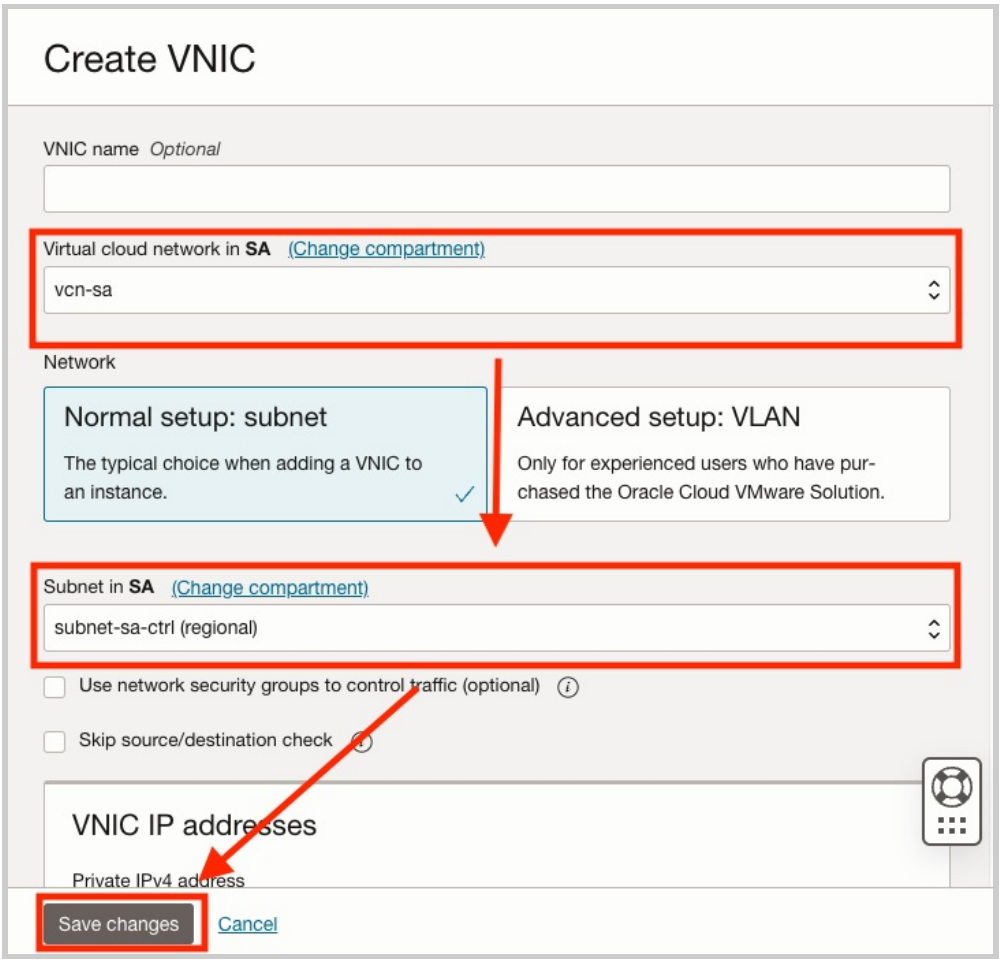
1. Make sure that your **Instance with cloudSwXtch installed** is **running**. You **cannot** attach a secondary VNIC if the machine is off.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Select **Attached VNICs** under **Resources**.

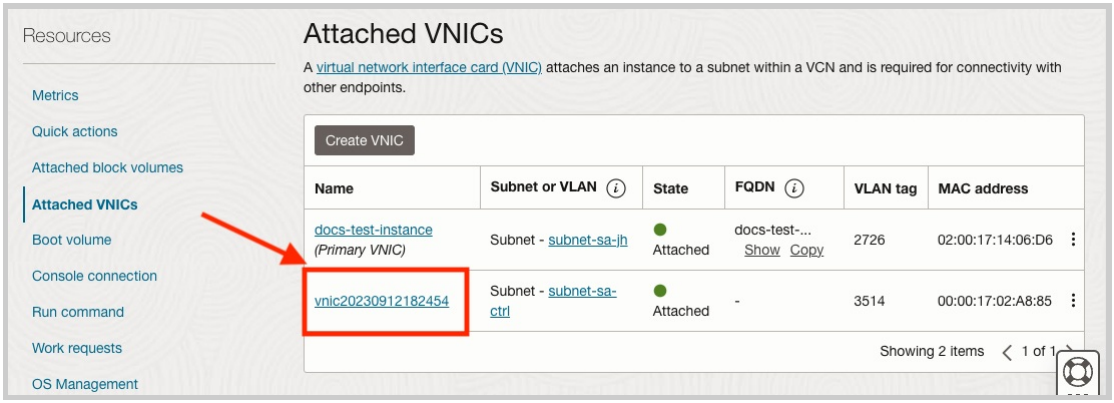


3. Click **Create VNIC**.
    a. **Pro-Tip**: Assign your secondary VNIC a user-friendly name. Otherwise, Oracle will assign a randomized ID.
    b. Choose the same **Virtual cloud network** and ctrl **Subnet** as your Primary VNIC.
    c. Select **Save Changes**.

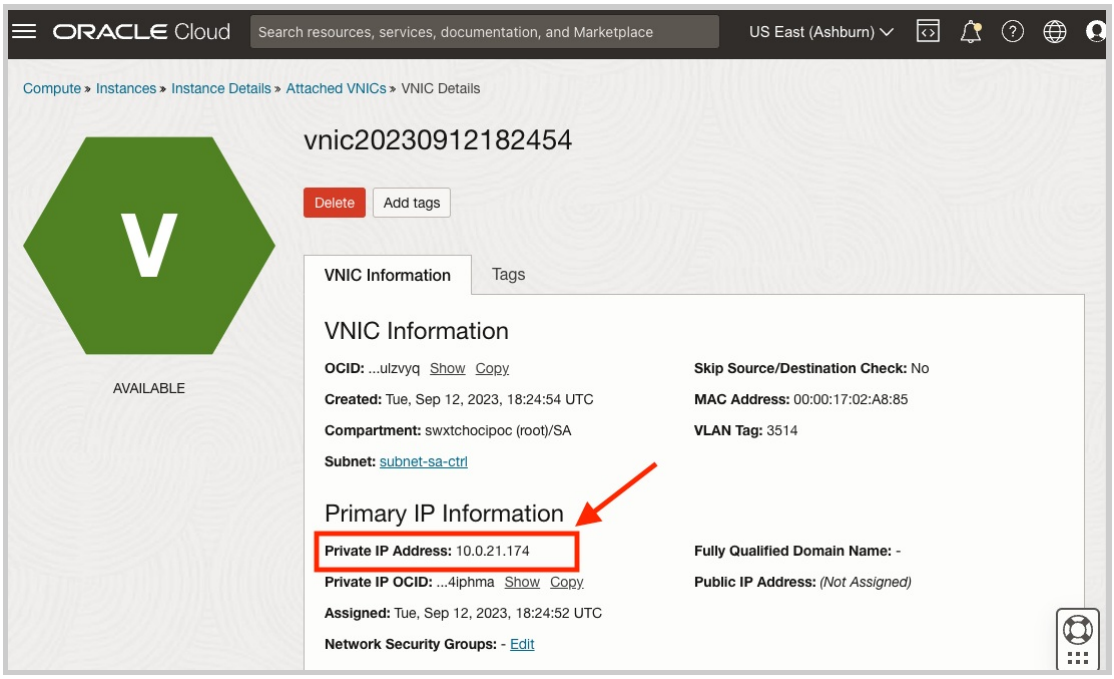


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. Click on the freshly created VNIC's name after it finishes attaching.



5. Record the **Private IP address**. **You will need it later.**



6. Log into your **Instancewith cloudSwXtch installed.**

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

7. Create the following file in the **/etc/netplan** folder and name it **02-datanic-static-config.yaml**. **Please note:** You will need to add the Private IP Address of the secondary VNIC into the file below.

```
Bash                                                    Copy

network:
                version: 2
                ethernets:
                    ens4:
                        match:
                            macaddress: 02:00:17:09:c2:cf
                        dhcp4: false
                        addresses:
                            - <ADD IP ADDRESS OF 2ND VNIC>/<XX>
```

   Where the <XX> is the net mask (or network mask) of ctrl-plane CIDR (in single-subnet configuration).
8. Apply the new config (**sudo netplan apply**).
9. Find the file **/etc/iptables/rules.v4** and open it in your editor.
10. Search for the following lines:

```
Bash                                                    Copy

-A INPUT -p all -s 10.0.128.0/24 -j ACCEPT
-A INPUT -p all -s 10.0.192.0/24 -j ACCEPT
```

11. Replace the **CIDRs** with your own **CIDRs**, corresponding to the ctrl and data subnets. These numbers can be the same if using a single-subnet configuration for both your VNICs.
12. Save file and reboot instance.

The secondary VNIC should now be successfully attached.

Upgrading your cloudSwXtch

After deployment, it is recommended to update your cloudSwXtch instance to latest. Please refer to the **Upgrading cloudSwXtch** article for more information.

Optional Step for BYOL: Contact swXtch.io for a license

Users deploying a BYOL instance of cloudSwXtch will need to contact swXtch.io for a license file. For more information. see How to License a cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

NEXT STEPS

The cloudSwXtch is ready to use. The next step is to install the xNIC on each client expected to get traffic from the cloudSwXtch. See **Installing xNIC** for more information on preparing clients.**How to License a cloudSwXtch**

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Cloud agnostic cloudSwXtch VM Install

WHAT TO EXPECT

In this article, you will learn how to install a cloudSwXtch instance on an existing Linux Ubuntu 20.04 virtual machine. This install process can be used on any cloud but requires a license file from swXtch.io. For more information about VM prerequisites, please see the **cloudSwXtch System Requirements**.

Pre-Installation Step: Create VM

Before installing cloudSwXtch, you will need to create **an Ubuntu 20.04 virtual machine** on your desired cloud **with connection to the internet.** In addition, it should encompass all the prerequisites outlined in the cloudSwXtch System Requirements

Step One: Install cloudSwXtch

In this step, users will execute commands in their VMs to manually install a cloudSwXtch instance.

1. Run your freshly created virtual machine using your desired tool.
2. Enter the following command to download the **cloudSwXtch installer script**:

   Shell

   Bash | Copy

   ```
   token="si=RDONLY&spr=https&sv=2021-06-08&sr=c&sig=xyPF7SyI1cagUAEIZViqCHz7RroFTy2Fkltn2wwvMzc%3D"
   curl -X GET -H "x-ms-date: $(date -u)" "https://sdmcdevstorage.blob.core.windows.net/imagebuilder/image_install.sh?$token" -o image_install.sh
   chmod +x image_install.sh
   ```

3. Use the following command to get the latest version of cloudSwXtch. **The latest release is 2.3.2**.

   Shell

   Bash | Copy

   ```
   ver="v2.3.2"
   ```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. Enter the following to download that version.

   Shell

   ```
   Bash
   Copy
   curl -X GET -H "x-ms-date: $(date -u)"
   "https://sdmcdevstorage.blob.core.windows.net/imagebuilder/install-
   ${ver}.tar.gz?$token" -o install-${ver}.tar.gz
   ```

5. Execute the installer.

   Shell

   ```
   Bash
   Copy
   sudo ./image_install.sh ${ver}
   ```

This will automatically reboot the machine.

Step Two: Contact swXtch.io for a license

Users will need to contact swXtch.io for a license file. For more information, see How to License a cloudSwXtch.

NEXT STEPS

The cloudSwXtch is ready to use. The next step is to install the xNIC on each client expected to get traffic from the cloudSwXtch. See **Installing xNIC** for more information on preparing clients.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Upgrading cloudSwXtch

WHAT TO EXPECT

In this article, users will learn how to update their cloudSwXtch when new versions are available. The following commands are cloud agnostic so they should work regardless of what cloud they're using.

There are two ways of ensuring your cloudSwXtch is up-to-date: via the cloudSwXtch or via the xNIC.

Upgrading cloudSwXtch via the cloudSwXtch

1. Sign onto the VM where the cloudSwXtch is running.
2. Run the following command:

   Shell

   ```
   Bash | Copy
   sudo /swxtch/swx update -i localhost -v v<desired version>
   ```

Example:

Shell

```
Bash | Copy
sudo /swxtch/swx update -i localhost -v v2.3.2
```

Upgrading cloudSwXtch via the xNIC

1. Connect to any VM where an xNIC is running.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Run the following command:

   Shell

   ```
   swx update -v <desired version> --ip <ip of cloudSwXtch>
   ```

Example:

```
swx update -v v2.3.2 --ip 10.5.1.6
```

Note: The <desired version> includes a "v" before the version number (e.g. v2.3.2).

> **Upgrading cloudSwXtch and xNICs**
>
> Make sure you upgrade all cloudSwXtches and xNICs in the environment to have the best functionality.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Installing cloudSwXtch Bridge

WHAT TO EXPECT

There are currently 2 types of cloudSwXtch Bridges: Type 2 and Type 1. Depending on their needs, it is suggested for users to use either cloudSwXtch Type 2 for most cases. Bridge Type 1 should really only be used for testing purposes.

In this article, users will learn about the difference between each cloudSwXtch Bridge Types and links on installation instructions for both.

Bridge Type 2

Type 2 of the cloudSwXtch Bridge is the more performant version of Bridge Type 1. It supports the following:

- Protocol Fanout only available for UDP Ingress
- **Bi-directional traffic between on-prem and the cloud**
- **Dynamic IGMP joins and leaves.** When an application in the cloud sends an IGMP join, then the cloudSwXtch in the cloud sends the information to the ground cloudSwXtch as a bridge, allowing the traffic to go through. Dynamic bridge is only supported from ground to cloud, not from cloud to ground.

See **Install cloudSwXtch Bridge Type 2** for installation instructions.

Bridge Type 1

Type 1 of the cloudSwXtch Bridge should **only be used for testing purposes** where there is no VPN or Express Route, only access via the Internet. Unlike Bridge Type 2, it **does not support bi-directional traffic between the on-prem and the cloud**. It only supports one direction: on-prem to the cloud. Additionally, it **does not** support dynamic bridge.

See **Install cloudSwXtch Bridge Type 1** for installation instructions.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install cloudSwXtch Bridge Type 2

PREREQUISITES

- A cloudSwXtch instance running in any cloud.
- Network connectivity from on-premises to the Virtual Network hosting the cloudSwXtch instance. A user should be able to ping the cloudSwXtch instance from the on-premises network.
- The cloudSwXtch Bridge can work with a single NIC. However, it is recommended to have at least 2 NICs with the non-primary NIC as a Mellanox card. This will result in better performance for high bandwidth traffic.
    - **Note**: ConnectX-5 cards are preferred for the non-primary NIC. For a list of recommendations, please visit the following link: **https://www.cisco.com/c/en/us/products/servers-unified-computing/third-party-adapters-listing.html?flt2_general-table0=Nvidia%2FMellanox**
- A VM or BareMetal bridge host machine running one of the following operating systems with **a minimum 4 cores, 8GB RAM and a recommended hard drive of 40GB (20GB min):**
    - Ubuntu 20.04 with Kernel 5.15 or greater
    - RHEL8/CentOS8 with Kernel of 5.11 or greater
    - RHEL9 with a Kernel of 5.11 or greater.
- The bridge host must be able to receive and/or send multicast traffic from the local network and send UDP packets to the cloud's Virtual Network using a VPN or Express Route. Internet only access is NOT viable for Type 2. (See **Install cloudSwXtch Bridge Type 1** if this is your only option.)

Firewall Rules

- cloudSwXtch Ctrl IP <-> Bridge Ctrl IP 80 (TCP)
- cloudSwXtch Data IP <-> Bridge Data IP 9999 (UDP)

Pre-Installation: Update Ubuntu 20.04 to Kernel 5.15

1. Use the following commands:

   Shell

Bash	Copy
`sudo apt-get install linux-generic-hwe-20.04`	

   Copy

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```bash
sudo apt update && sudo apt full-upgrade
```

Copy

3. Reboot your machine. If a user is running an Air-Gapped install, they will need to download and Install the package manually: https://vitux.com/how-to-install-latest-linux-kernel-5-15-on-ubuntu-20-04/
4. Use the following to verify the kernel version is at 5.15:

   Shell

   ```bash
   uname -r
   ```

   Copy

   The output will read:

   ```
   5.15.0-1023
   ```

   Copy

Updating RHEL8/CentOS8 with Kernel 5.11 or Greater

For more information on how to update your kernel for RHEL8/CentOS8, please read the following articles:

- https://www.ubuntumint.com/install-linux-kernel-rhel-8/
- https://www.2daygeek.com/changing-default-kernel-rhel-8-rhel-9/

WARNING: Update your kernel at your own risk for RHEL/CentOS8.

Installation

This method can be used to install the bridge application onto the bridge host machine. It will only work if the cloudSwXtch instance is up and running and the bridge host has network connectivity to the cloudSwXtch instance.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. **Open** a bash console on any VM that is on the same control plane network as the cloudSwXtch that you intend to use as the bridge host.
2. **Ping** the cloudSwXtch using your instance name or IP.

   Text

   None | Copy

   ```
   ping <swxtch-instance-ip or instance-name>
   ```

   Copy

   a. **If the ping fails to find the cloudSwXtch instance by name**, try pinging the IP address of the cloudSwXtch instance. If the IP works, they use the IP address in place of the name in all future commands. This can happen if the default DNS settings are changed for the Virtual Network.

3. **Run** the cloudSwXtch bridge installer script:

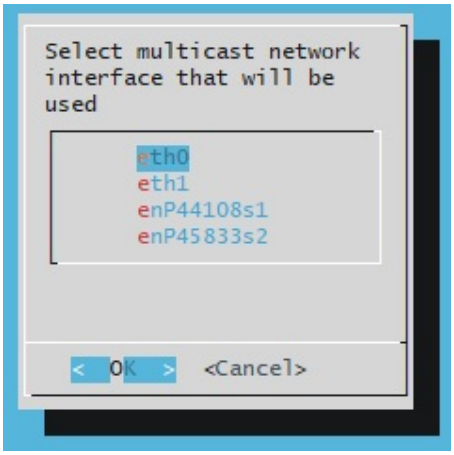
   Text

   None | Copy

   ```
   sudo sh -c "curl -s http://<swxtch-ip>/services/install/swxtch-bridge-install.sh | bash -s -- -t 2"
   ```

   Copy

   a. When prompted, **select** the network interface that will be used to receive and send multicast traffic (i.e. interface to/from on prem).



The service will be automatically initialized and the logs can be seen with:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

None	Copy

```
sudo journalctl -u swxtch-bridge2 -f -n 100
```

Copy

TROUBLESHOOT: Configuring Bridge Type 2 Interfaces or Gateway

In the event that your cloudSwXtch Bridge is not sending data to the cloudSwXtch, then it is recommended to review the Bridge Type 2 configuration json file to verify the correct interfaces have been selected. For more information on how to do this, please see the **Configuring cloudSwXtch Bridge Type 2 Instances** section.

Alternatively, there can be an issue with the gateway address. For more information, see **Using a Specific Gateway Address for Bridge Type 2**.

Configuring cloudSwXtch Bridge Type 2

There may be some scenarios that require special configuration for the cloudSwXtch Bridge. For more information, see Bridge Type 2 under Configuring cloudSwXtch.

---

cloudSwXtch Bridge Type 2 Commands

After deploying your cloudSwXtch Bridge Type 2, a user can execute commands to stop, start, and restart their instance. They can execute these commands in the command window of their cloudSwXtch Bridge.

STOP

Bash	Copy

```
sudo systemctl stop swxtch-bridge2.service
```

Copy

START

Bash	Copy

```
sudo systemctl start swxtch-bridge2.service
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Copy

RESTART

Bash | Copy

```
sudo systemctl restart swxtch-bridge2.service
```

Copy

Uninstalling cloudSwXtch Bridge Type 2

To uninstall your cloudSwXtch Bridge Type 2 application from your bridge host machine:

1. Execute the following command on the Bridge VM on-prem:

Bash | Copy

```
sudo sh -c "curl -s http://<swxtch-ip>/services/install/swxtch-bridge-install.sh | bash -s -- -u"
```

Copy

Your cloudSwXtch Bridge Type 2 instance should now be uninstalled.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install cloudSwXtch Bridge Type 1

PREREQUISITES

You will need:

- A cloudSwXtch instance running in a cloud.
- Network connectivity from on-premises to the Virtual Network hosting the cloudSwXtch instance. You should be able to ping the cloudSwXtch instance from the on-premises network.
- A VM or Bare Metal bridge host machine running RHEL 7+, CentOS 7+, or Ubuntu 18.04+ with a minimum of 2 cores, 4GB RAM.
- A bridge host that must be able to receive multicast traffic from the local network and send UDP packets to the cloud Virtual Network.

Direct installation to bridge host -T1

This method can be used to install the bridge application onto the **bridge host** machine. It will only work if the cloudSwXtch instance is up and running and the **bridge host** has network connectivity to the cloudSwXtch instance.

1. **Open** a shell script on any VM that is on the same control plane network as the cloudSwXtch that you intend to use as the bridge host.
2. **Ping** the cloudSwXtch using your instance name.

   Text

   ```
   Bash                                                    Copy
   sudo ping <swxtch-instance-name>
   ```

   a. **If the ping fails to find the switch instance by name**, try pinging the IP address of the cloudSwXtch instance. If the IP works, then use the IP address in place of the <swxtch-instance-name> in all future commands. This can happen if the default DNS settings are changed for the virtual network.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. **Run** the bridge installer script:

   Text

   Bash | Copy

```
curl http://<swxtch-instance-name>/services/install/swxtch-bridge-install.sh | bash -s -- -k -t 1
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Installing xNIC

SUMMARY

- The following article will explain how to install the xNIC component on your Windows and Linux system.
- **xNIC** is the software that runs on your VM to create a virtual NIC. The **xNIC** connects your VM to a cloudSwXtch instance.

xNIC System Requirements

There are some major feature considerations to make when deciding what xNIC version to use. These prerequisites are further detailed in the xNIC System Requirements article.

Linux Installation Guide

xNIC Linux Installation

The installer script will install the xNIC as a service as well as the utility applications used to verify the operation of the xNIC and cloudSwXtch instance network for a Linux system. See Testing.

Windows Installation Guide

xNIC Windows Installation

The installer script will install the xNIC as a service as well as the utility applications used to verify the operation of the xNIC and the cloudSwXtch instance network for a Windows system.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

xNIC System Requirements

A cloudSwxtch must exist to create a xNIC. See cloudSwXtch System Requirements for more information.

xNIC software

The xNIC software must be run on each virtual machine that is to be part of the IP multicast network and not a cloudSwXtch or a cloudSwXtch Bridge. This software can be installed on hosts which meet the following requirements:

Available Operating Systems

Linux	Windows
• AlmaLinux 8.8 • Amazon Linux 2023 • Centos 8 Minimum • Oracle Linux 8 • RHEL 8.8 • RHEL 9.2 • Rocky Linux 8 • Rocky Linux 9 • Ubuntu 20.04 • Ubuntu 22.04 *Minimum Kernel Version 4.18*	• Windows Server 2022 • Windows Server 2019 • Windows 11 Pro/Enterprise • Windows 10 Pro/Enterprise

CPU Architecture

x86_x64

Network Connectivity

1 NIC or 2 NICs for higher performance (one for each sub-net: ctrl-subnet and data-subnet)

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1 NIC vs. 2 NICs

An xNIC instance may have 1 or 2 NICs depending on the subnet configuration of the cloudSwXtch.

- **If a cloudSwXtch has 2 NICs sharing a single subnet**, an xNIC needs only 1 NIC (control). This NIC will share the same single subnet for control and data plane communications as the cloudSwXtch.
- **For high performance, a cloudSwXtch should have 2 NICs using 2 different subnets**, an xNIC will need 2 NICs connected to separate subnets:
    - A subnet for control plane traffic (referred to as the **ctrl-subnet** from here on).
    - A subnet for data plane traffic (referred to as the **data-subnet** from here on).

Subnet Selection

The subnets must be the same subnets used for the cloudSwXtch.

The install requires a simple command that installs the xNIC from the cloudSwXtch. The install typically takes less than one minute per host. See the installation sections for more details.

Tunnel network

The xNIC software must be installed on each virtual machine that is to send or receive multicast traffic. The xNIC software will create a tunnel network interface (called **swxtch-tun0** for Linux and **swxtch-tun** for Windows) that presents to the application a network subnet of 172.30.X.Y. Each virtual machine running the xNIC software will be assigned an IP address in this range.

NOTE:

The swxtch tunnel interface should only be used for multicast traffic. Any other network traffic should target other network interfaces.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install xNIC on Linux

WHAT TO EXPECT

The xNIC is a lightweight service that must be installed on every VM sending or receiving multicast traffic to and from a cloudSwXtch. An xNIC should not be installed on a cloudSwXtch or cloudSwXtch Bridge VM.

In this article, users will learn how to install the xNIC software in the Linux systems.

Installing xNIC for Linux

BEFORE YOU START

Review **xNIC System requirements**.

Network Acceleration

If using Azure, the data-subnet must have the "Network Acceleration" feature enabled.

Running the Install Script

To make installation easy, the xNIC is installed from the cloudSwXtch instance via a one-time shell command. The xNIC is matched to the attached cloudSwXtch instance and should be reinstalled if the cloudSwXtch version changes.

To run the install:

1. **Open** a terminal on the VM you wish to install the xNIC software on.
2. **Verify** network connectivity to the cloudSwXtch instance by "pinging" the switch.

```bash
ping <cloudSwXtch-instance-name>
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Ping Fails

If the ping fails to find the cloudSwXtch instance by name, try pinging the IP address of the cloudSwXtch instance. If the IP works, then use the IP address in place of the <switch-instance-name> in all further commands.

This can happen if the DNS settings are not configured for the virtual network.

TURNING OFF FIREWALL

It is **recommended** for users to turn off their firewall. The installer script will automatically open ports 10800 and 9999.

To open up additional ports for producing/consuming multicast traffic, use the following command:

Bash | Copy

```
sudo firewall-cmd --add-port=<port>/udp --permanent
sudo systemctl restart firewalld
```

3. **Run** the following installer script:

Bash | Copy

```
curl http://<cloudSwXtch-instance-name>/services/install/swxtch-xnic-install.sh | bash
```

Alternatively, you can run the install script after downloading it using the **wget** command:

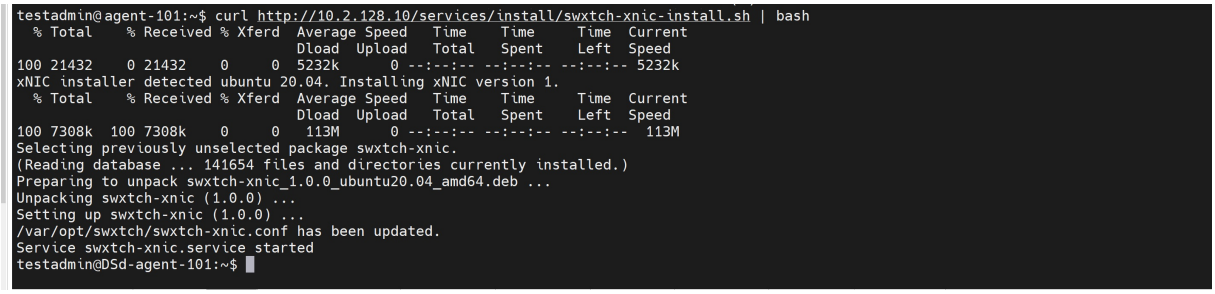
Bash | Copy

```
wget http://<cloudSwXtch-ip>/services/install/swxtch-xnic-install.sh
chmod +x swxtch-xnic-install.sh
./swxtch-xnic-install.sh
```

The installer script will install the xNIC as a service and a set of utility applications that can be used to verify the operation of your cloudSwXtch network. Refer to Testing for more details.

A successful install is shown below:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



IF THE INSTALL FAILS:

Your content goes here If the install fails, validate that the VM has at least two NICs and the NICs are on the same subnets for control and data as the cloudSwXtch. The ctrl-subnet should be assigned to the primary NIC.

If you are using Azure, validate that the data-subnet has "Network Acceleration" feature enabled.

Setting the rp_filter on Linux

During xNIC installation, the Linux rp_filter is set to loose mode by default at runtime. This allows for the xNIC to work on asymmetric networks, meaning that it can receive packets from machines outside of its subnet if the source is routable.

To opt out of this configuration, navigate to the **xnic.json** file after completing the xNIC installation process. This file can be found in **/var/opt/swxtch/xnic.json**. To edit the file, one option is to use nano as shown below:

Bash | Copy

```
sudo nano /var/opt/swxtch/xnic.json
```

Next to "overrideSourceIP" in the json file, change the parameter to **true**. **Save the file** and **restart** the xNIC VM. This will set the rp_filter back to the original mode and will remain like that for future reboots. Note that this means our software will do source network address translation on incoming packets.

Verifying Installer Files [Optional]

Prior to installing the xNIC, the user may need to verify the authenticity of swXtch.io's installer files. While it is not a necessary step in the installation process, it is still an available option to users with security protocols that require files to be validated. **Please note**: This option is only available in cloudSwXtch versions 2.2 or greater.

After downloading the **swxtch-xnic-install.sh** file on your VM, run the following steps:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. Download the public key from swXtch.io using the following command on your VM:

   Shell

   ```
   curl https://services/swxtch.io/assets/sig_public.pem
   ```

2. Move the public key into the same directory as the swxtch-xnic-install.sh file.
3. Run the install command with the **--verify** argument:

   Shell

   ```
   ./swxtch-xnic-install.sh --verify
   ```

The --verify argument will download the .sig file associated with the xNIC installer file based on the VM's operating system, verify with the public key, and proceed to install the xNIC onto the VM as a service.

Additional Arguments

There are additional arguments when installing the xNIC.

Note that the **ctrl-** and **data-** interfaces are from the VM the xNIC is installed. These will be set automatically by the installer. There may be some instances where you will need to specify them. For example, if you have three network interfaces and you want to specify what you want to use for ctrl or data, you can manually select them using the **-ctrl_interface <interface index>** or **-data_interface <interface index>** arguments. Also, these argument help in complex contexts where the agent is in a different vNet/VPC from the cloudSwXtch.

A full list of arguments is detailed below:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
$ ./swxtch-xnic-install.sh -h
Usage: ./swxtch-xnic-install.sh [OPTIONS]
  -t <1|2>                          xNIC type to install (default:
2 if supported in this OS, 1 otherwise)
  -u                                uninstall xNIC instances
  --ctrl_interface <interface name>   manual selection of the
Control interface
  --data_interface <interface name>   manual selection of the Data
interface
  --ptp <true|false>                installing of Precision Time
Protocol (default: true)
  --verify                          Verify install package with
respective signature file
  -h | --help                       shows this help
```

Note: There is an option for users to switch between xNIC Type 1 and Type 2, latter being the default. All installation instructions and system requirements are solely for Type 2. It is not recommended to use Type 1 unless otherwise suggested by swXtch.io Support.

Skipping PTP Installation

If you would like to skip Precision Time Protocol installation, they can run the following command:

Bash Copy

```
./swxtch-xnic-install.sh --ptp false
```

Testing

xNIC installation includes a set of utility applications that you can use to verify the operation of your cloudSwXtch network. Refer to Testing for details.

- **swxtch-top**: An application to display real-time statistics of the cloudSwXtch instance.
- **swxtch-perf**: An application to produce and consume unicast and multicast traffic for testing purposes.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Running swxtch-top on Linux

<swxtch-hostname>: name of your existing swxtch or "host" swxtch

Bash | Copy

```
swxtch-top dashboard --swxtch <swxtch-hostname>
```

Uninstalling xNIC on Linux

To uninstall xNIC on Linux, users can follow the steps in the xNIC Linux Uninstall Guide.

Upgrading xNIC on Linux

To upgrade xNIC on Linux, users can follow the steps in the xNIC Linux Upgrade Guide.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

xNIC Linux Uninstall

WHAT TO EXPECT

In this article, users will learn how to remove the xNIC from their Linux system for both Ubuntu and Redhat.

Uninstalling xNIC on Linux

1. **Open** a shell on the host VM. The host VM is the VM where you wish to uninstall the xNIC software.

2. **Run** the following command depending the xNIC version:

None | Copy

```
curl http://<swxtch-instance-name>/services/install/swxtch-xnic-install.sh | bash -s  -- -u
```

3. **The uninstall script will remove Linux xNIC.**

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

xNIC Linux Upgrade

BEFORE YOU START

When a cloudSwXtch has been updated, it's recommended to update connected xNICs as well.

In this article, users will be able to use the appropriate script to upgrade their xNIC.

Upgrading Linux xNIC

24/7 Operations

If the services need to be up and running 24/7, swXtch.io suggests that redundant systems exist for which will be referred to as "Main" and "Backup". During an upgrade the Backup system should be upgraded, then the traffic should be routed to the Backup while the Main is upgraded.

You can use the following command to uninstall the existing xNIC and upgrade it.

1. **Run** the installer script:

   Shell

   Bash | Copy

   ```
   curl http://<swxtch-instance-name/services/install/swxtch-xnic-install.sh | bash
   ```

Additional Arguments

There are additional options when installing the xNIC.

Note that the **ctrl-** and **data-** interfaces are from the VM the xNIC is installed. These will be set automatically by the installer. There may be some instances where you will need to specify them. For example, if you have three network interfaces and you want to specify what you want to use for ctrl or data, you can manually select them using the -ctrl_interface or -data_interface arguments. Also, these argument help in complex contexts where the agent is in a different vNet/VPC from the cloudSwXtch.

For xNIC 1 on Linux, multiple xNICs can be installed on one VM by using the -i and the --tun-subnet arguments. In this case, the control interface will be the same while the data interface will differ for each xNIC on the Linux VM.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

A full list of arguments is detailed below:

Bash Copy

```
$ ./swxtch-xnic-install.sh -h
Usage: ./swxtch-xnic-install.sh [OPTIONS]
  -t <1|2>                              xNIC type to install (default:
2 if supported in this OS, 1 otherwise)
  -u                                    uninstall xNIC instances
  --ctrl_interface <interface name>     manual selection of the
Control interface
  --data_interface <interface name>     manual selection of the Data
interface
  --ptp <true|false>                    installing of Precision Time
Protocol (default: true)
  -h | --help                           shows this help
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install xNIC on Windows

WHAT TO EXPECT

The xNIC is a lightweight service that must be installed on every VM sending or receiving multicast traffic to and from a cloudSwXtch. An xNIC should not be installed on a cloudSwXtch or a cloudSwXtch Bridge VM.

In this article, users will learn how to install the xNIC software on Windows systems.

Installing xNIC for Windows

BEFORE YOU START

Review **xNIC System Requirements**.

Firewall Restrictions
The Windows installation process adds rules to Windows Defender Firewall, which allow for traffic through the UDP ports 10800 and 9999. The rule names are SwXtchControl, SwXtchData, and SwXtchTun.

Network Acceleration
If using Azure, the data-subnet must have the "Network Acceleration" feature enabled.

Running the Install script

To make installation easy, the xNIC is installed from the cloudSwXtch instance via a one-line shell command. The xNIC is matched to the attached cloudSwXtch instance and should be reinstalled if the cloudSwXtch version changes.

The xNIC takes less than a minute to install on an existing VM.

To run the install:

1. **Open** a PowerShell terminal on the Windows VM that you aspire to install the xNIC software on.
    - **If you are working on Windows 11**, please use Windows Terminal instead for installation.
2. **Verify** network connectivity to the cloudSwXtch instance by “pinging” the switch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
ping <cloudSwXtch-instance-name>
```

Ping Fails

If the ping fails to find the cloudSwXtch instance by name, try pinging the IP address of the cloudSwXtch instance. If the IP works, then use the IP address in place of the `<switch-instance-name>` in all further commands.

This can happen if the default DNS settings are changed for the virtual network.

3. The installer script will automatically remove any firewall restrictions to UDP ports 10800 and 9999. The cloudSwXtch sends UDP packets to these ports as part of normal operation.

Special Rules for Windows Defender Firewall

It is recommended to simply turn off the firewall. Additionally, users can open up additional ports for producing/consuming multicast traffic by using the following command in PowerShell:

Bash | Copy

```
New-NetFirewallRule -Name 'rule_name' -DisplayName 'rule_name' -Enabled
True -Direction Inbound -Protocol UDP -Action Allow -LocalPort 1234
```

4. Ensure **Windows SecureBoot UEFI** is disabled. This can be enabled or disabled from the VM or EC2 configuration in your preferred cloud.

5. **Download** and **run** the installer script in Powershell as an administrator:

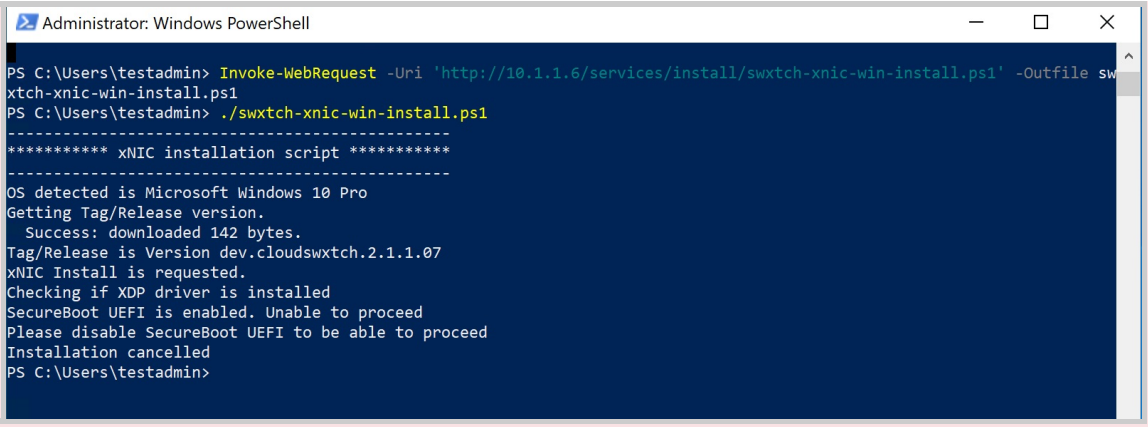
Bash | Copy

```
Invoke-WebRequest -Uri 'http://<cloudSwXtch-instance-
name>/services/install/swxtch-xnic-win-install.ps1' -Outfile swxtch-
xnic-win-install.ps1
```

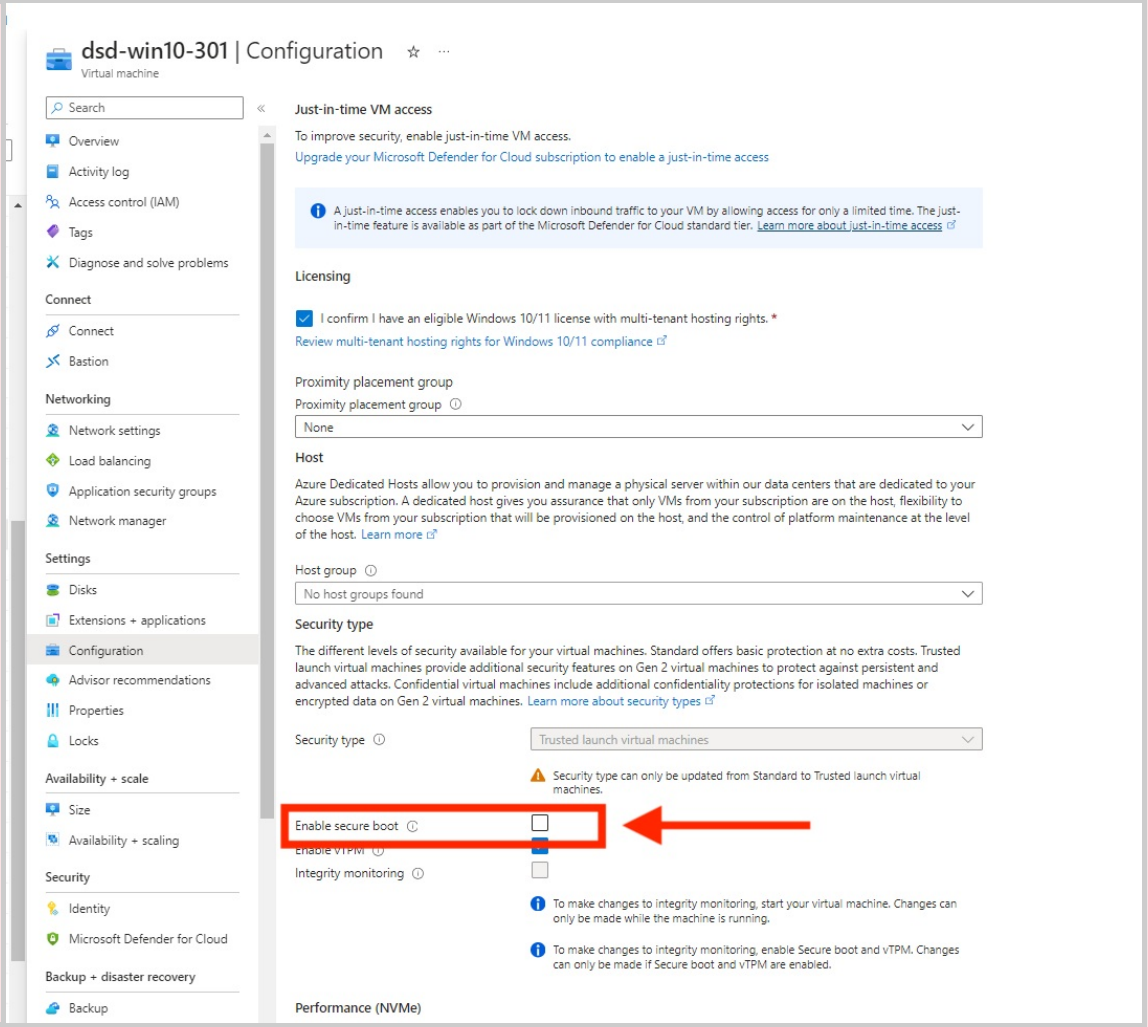
Secure Boot Error

The following error will appear if your Secure Boot is enabled or if you're not running as an administrator in Powershell. The below example shows running as an administrator, therefore it is likely that secure boot is incorrectly enabled on the VM.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



To disable Secure Boot on your Azure VM, go to the Microsoft Azure Portal and navigate to your VM. Under **Settings**, select **Configuration** and scroll down to **Security Type**. De-select **Enable secure boot** and then select **Apply** to confirm your changes.



To disable Secure Boot on your GCP VM, go to the Compute portal and navigate to your VM Instances and select your desired VM. Under Details, scroll down to Security and Access and validate that Secure Boot is off under Shielded VM. If it is enabled, stop your VM and select edit to disable the option. Save and restart VM.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Additional Arguments

There are additional options when installing the xNIC. To see these options, use the **-h** argument.

The **ctrl-** and **data-** interfaces are from the VM the xNIC is installed. These will be set automatically by the installer. There may be some instances where you will need to specify them. For example, if you have three network interfaces and you want to specify what you want to use for ctrl or data, you can manually select them using the **-ctrl_interface <interface index>** or **-data_interface <interface index>** arguments detailed below.

Note: There is an option (-t) for users to switch between xNIC Type 1 and xNIC Type 2 (default). All installation instructions and system requirements are solely for Type 2. It is not recommended to use Type 1 unless otherwise suggested by swXtch.io Support.

Bash Copy

```
PS C:\Users\testadmin> .\swxtch-xnic-win-install.ps1 -h
Usage: C:\Users\testadmin\swxtch-xnic-win-install.ps1 [OPTIONS]
  -t <1|2>                              xNIC type to install
(default: 2 if supported in this OS, 1 otherwise)
  -u                                    uninstall xcd xNIC only (no
other options allowed)
  -unattended                           unattended installation (in
case of reboot, the user will not be prompted)
  -ctrl_interface <interface index>     manual selection of the
Control interface
  -data_interface <interface index>     manual selection of the Data
interface
  -ptp <true|false>                     installing of Precision Time
Protocol (default: true)
  -h                                    shows this help
```

Please note: The **ctrl_interface** and **data_interface** commands should only be used in complex configurations where the installer cannot locate them. Contact support@swXtch.io for more information.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Getting the interface index for ctrl_interface and data_interface

To get the interface index, you can run the following command:

Bash | Copy

```
get-netipconfiguration
```

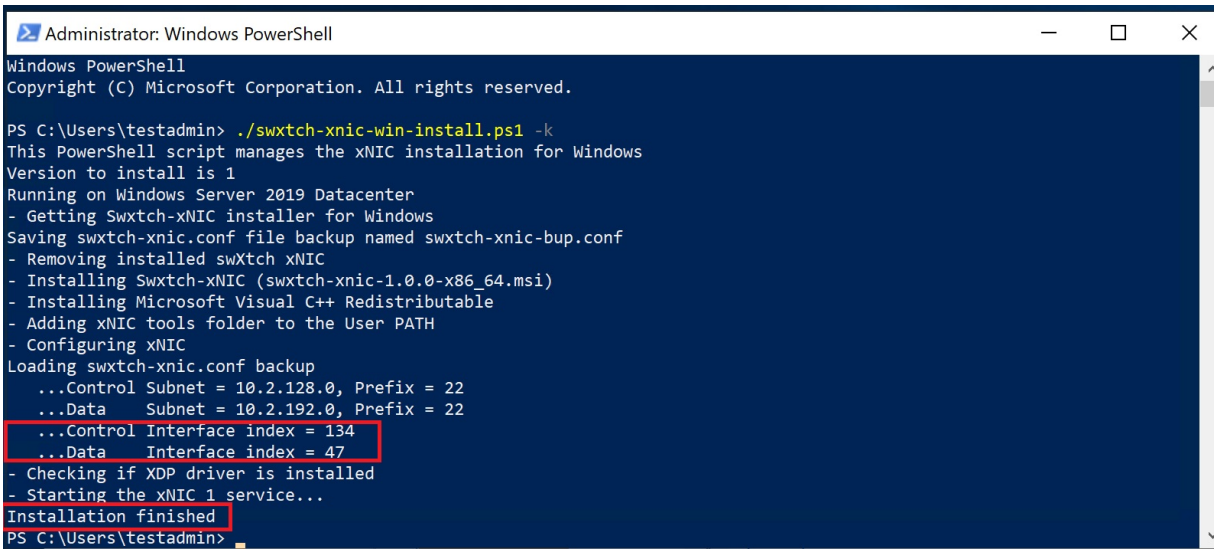
A sample of the response will be returned with the InterfaceIndex (***) listed for both:

Bash | Copy

```
PS C:\Users\testadmin> get-netipconfiguration

InterfaceAlias       : Ethernet
InterfaceIndex       : 36 ***
InterfaceDescription : Microsoft Hyper-V Network Adapter
NetProfile.Name      : Network
IPv4Address          : 10.2.128.75
IPv6DefaultGateway   :
IPv4DefaultGateway   : 10.2.128.1
DNSServer            : 168.63.129.16



InterfaceAlias       : Ethernet 2
InterfaceIndex       : 60 ***
InterfaceDescription : Microsoft Hyper-V Network Adapter #2
NetProfile.Name      : Unidentified network
IPv4Address          : 10.2.192.82
IPv6DefaultGateway   :
IPv4DefaultGateway   :
DNSServer            : 168.63.129.16
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

7. The installer script will install a **Windows service** called **swXtchNIC**:



8. **Reboot** your machine once the installation is complete. This will enable you to execute cloudSwXtch tools properly from your user home directory such as **swxtch-top**.

Errors

The control and data interfaces should have proper numbers. A 0, or negative number, indicates an error in the configuration of the control or data subnets for the xNIC. The control and data subnets of the cloudSwXtch and the xNICs should be the same.

If you are using Azure, validate that the data-subnet has "Network Acceleration" featured enabled.

Testing

The installation includes a set of utility applications that you can use to verify the operation of your cloudSwXtch network. Refer to Testing for details.

- `swxtch-top.exe` : An application to display real-time statistics of the cloudSwXtch instance.
- `swxtch-perf.exe` : An application to produce and consume multicast traffic for testing purposes.

Running swxtch-top on Windows

Bash | Copy

```
swxtch-top --swxtch <cloudSwXtch-hostname>
```

- **<cloudSwXtch-hostname>**: the name of your existing or "host" cloudSwXtch

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Uninstalling xNIC on Windows

To uninstall xNIC on Windows, users can follow the steps in the Uninstall xNIC on Windows guide.

Upgrading xNIC on Windows

To upgrade xNIC on Windows, users can follow the steps in the Upgrade xNIC on Windows guide.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Uninstall xNIC on Windows

WHAT TO EXPECT

In this article, users will learn how to remove the xNIC from their Windows system.

Uninstalling xNIC on Windows

When uninstalling xNIC on Windows, please **do not** uninstall using the Add/Remove Programs feature. It is important to use the command below instead for uninstall.

1. **Open** Powershell on your Windows system (command window if Windows 11).
2. Run the following command:

   Text

None	Copy
`.\swxtch-xnic-win-install.ps1 -u`	

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Upgrade xNIC on Windows

WHAT TO EXPECT

When a cloudSwXtch has been updated, their xNIC should be upgraded as well. This is very simple since you will only need to reinstall the script. The installer will automatically remove the older version of xNIC.

In this article, users will learn to use the appropriate script to upgrade their xNIC.

Make sure that you have the latest version of cloudSwXtch installed. You can find information about how to upgrade your cloudSwXtch by clicking here: Upgrading cloudSwXtch. You can also upgrade your cloudSwXtch by deleting and recreating the instance.

Upgrading xNIC on Windows

1. **Open** PowerShell. **If you are using Windows 11**, please use Windows Terminal.

2. **Download** the installer script:

Bash | Copy

```
Invoke-WebRequest -Uri 'http://<swxtch-instance-name>/services/install/swxtch-xnic-win-install.ps1' -Outfile swxtch-xnic-win-install.ps1
```

3. **Run** the script. Please use the appropriate command for your version. **Note**: xNIC Type 2 is the default.

None | Copy

```
./swxtch-xnic-win-install.ps1
```

The latest version of the Windows xNIC will be installed.

Remember to Reboot

Reboot the machine after the upgrade is complete. You must do this to be able to execute the cloudSwXtch tools properly from your user home directory.

Additional Arguments

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

To see all the options available for the xNIC installation/update script, use the **-h** argument.

Note that the **ctrl-** and **data-** interfaces are from the VM the xNIC is installed. These will be set automatically by the installer. There may be some instances where you will need to specify them. For example, if you have three network interfaces and you want to specify ctrl or data interfaces, you can manually select them using the **-ctrl_interface <interface index>** or **-data_interface <interface index>** arguments detailed below.

Bash Copy

```
PS C:\Users\testadmin> .\swxtch-xnic-win-install.ps1 -h
Usage: C:\Users\testadmin\swxtch-xnic-win-install.ps1 [OPTIONS]
  -t <1|2>                              xNIC type to install
(default: 2 if supported in this OS, 1 otherwise)
  -u                                    uninstall xcd xNIC only (no
other options allowed)
  -unattended                           unattended installation (in
case of reboot, the user will not be prompted)
  -ctrl_interface <interface index>     manual selection of the
Control interface
  -data_interface <interface index>     manual selection of the Data
interface
  -ptp <true|false>                     installing of Precision Time
Protocol (default: true)
  -h                                    shows this help
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install xNIC on Kubernetes

WHAT TO EXPECT

The xNIC is a lightweight service that must be installed on every VM sending or receiving cloudSwXtch traffic. This creates a virtual network interface within the node in a Kubernetes Cluster. Applications that use IP multicast should target this virtual network interface.

In this article, you will learn how to install xNIC on K8s.

Supported Types

Below is a list of supported K8s Clusters.

- AKS (Cilium or Kubernetes)
- EKS
- GKE

The following operating systems in a pod are supported for the xNIC on K8s: RHEL 8, CentOS 8 or Ubuntu 20.04.

Installation

The installation process can be split into three steps:

1. Create a Kubernetes Cluster (AKS with Cilium/Kubernetes, EKS or GKE)
2. Install xNIC on K8s Cluster
3. Test xNIC on K8s

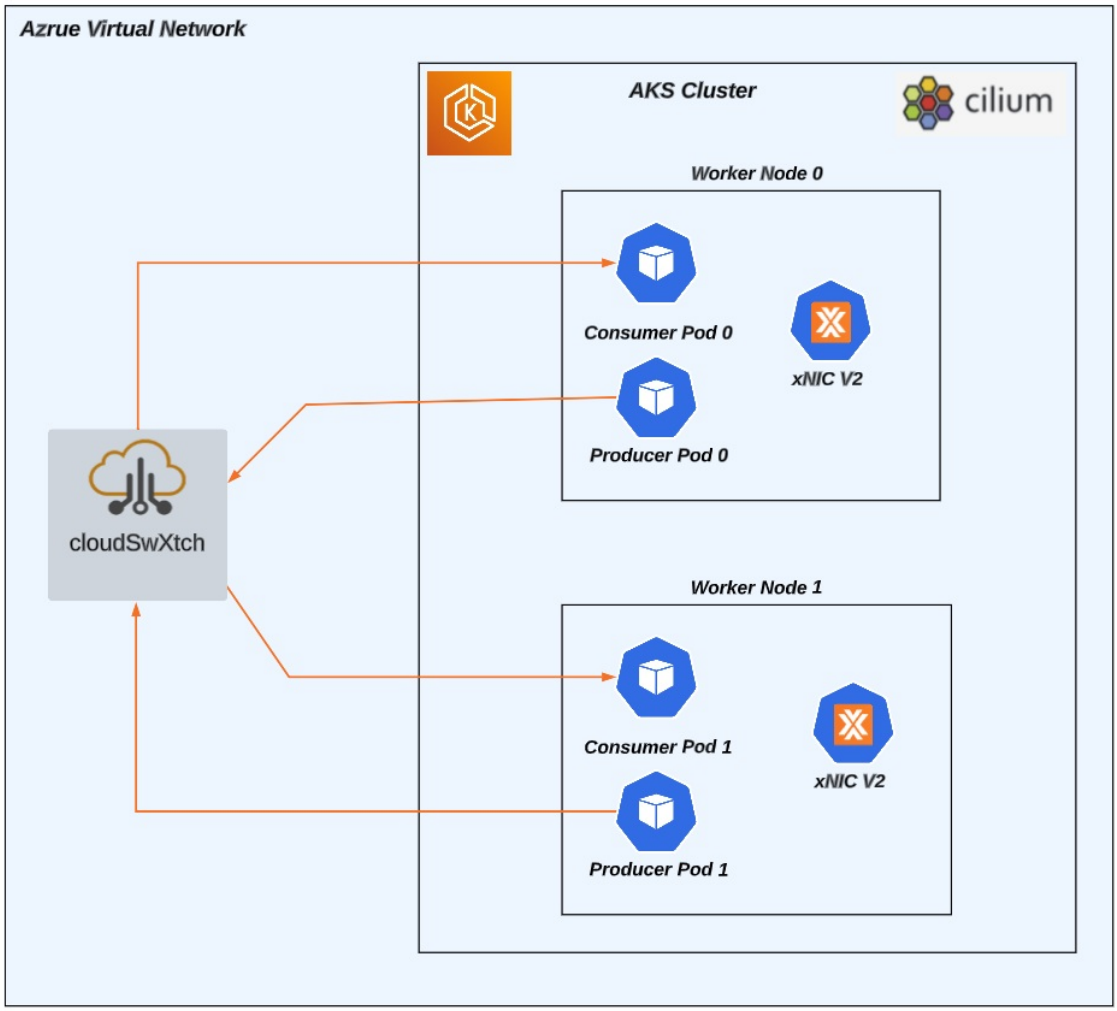
Post-Installation

You can learn how to upgrade your xNIC nodes on K8s, here.

xNIC Architecture Diagram

Below is an example architecture for an xNIC installed on AKS with Cilium with communication to and from a cloudSwXtch. Other Virtual Machines (not AKS) with xNICs installed could also communicate with the AKS worker nodes via cloudSwXtch and xNIC v2.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Install xNIC on K8s Cluster

> **WHAT TO EXPECT**
> xNIC is a lightweight service that must be installed on every Kubernetes cluster used for sending and/or receiving cloudSwXtch traffic. This creates a virtual network interface within the VM's operation system. Applications that use IP multicast should target this virtual network interface.
> **In this article**, users will learn how to install xNIC on one of the supported types of Kubernetes.

Overview

Unicast traffic will not be affected by this feature since it will work as it did before. The xNIC will only be used for Multicast traffic. The default interface xNIC will use is **eth0**. It can be installed via your preferred cloud's CloudShell or you can assign a VM as a manager to control your cluster. Either way, it is required to have access to the cloudSwXtch and the cluster.

In this document, we will discuss how to do it via the CloudShell. However, the commands below will work in either the CloudShell or on the VM managing the K8s cluster.

Running the Install Script

> **BEFORE YOU START**
> If you haven't already, please create a Kubernetes Cluster. This is a prerequisite before installing the xNIC.

To make installation easy, the xNIC is installed from the cloudSwXtch instance via a one-line shell command. The xNIC is matched to the attached cloudSwXtch and should be reinstalled if the cloudSwXtch version changes.

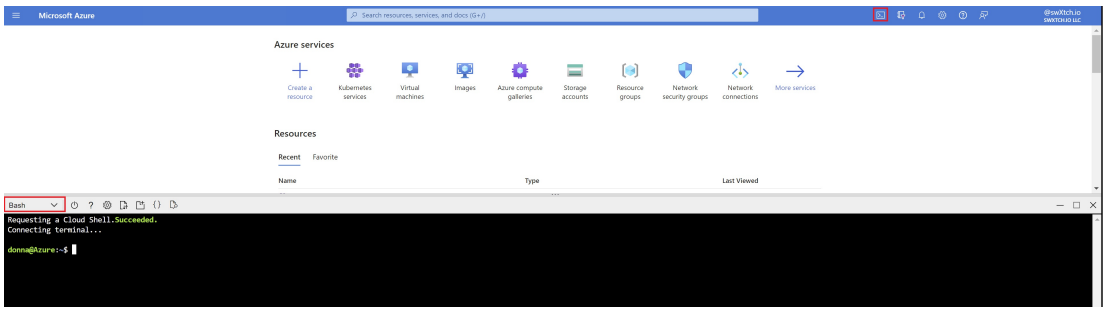
This process takes less than a minute to install on an existing K8s cluster.

To run the install:

1. Ensure your cloudSwXtch is version **2.0.89 or greater.** If it is not upgraded, see Upgrading cloudSwXtch.
2. Sign into **your desired cloud provider**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. Open **cloudShell** as Bash.



4. Paste in the following commands, replacing the <cloudSwXtch_IP> with your cloudSwXtch's Ctrl IP address.

```bash
kubectl run installer --image=busybox -- sh -c "wget http://<cloudSwXtch_IP>/services/install/xnic_ds_installer.sh; sleep 3650"
kubectl cp default/installer:/xnic_ds_installer.sh xnic_ds_installer.sh
kubectl delete po/installer --grace-period 1
chmod +x xnic_ds_installer.sh
```

5. Run one of the following scripts:

With Internet Access:

```bash
./xnic_ds_installer.sh
```

Without Internet Access:

```bash
./xnic_ds_installer.sh -ag
```

An example of a successful install without INTERNET access is shown below:

```bash
$ ./xnic_ds_installer.sh -ag

[i] Detected Cloud: AZURE
[i] Cilium Installation detected
[i] Setting CNI to CILIUM...
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
#########################################################################
###################################
        This script modifies the underlying configuration of Cilium
CNI to make it compatible
        with Multicast Networks.
        It also installs xNIC DaemonSet on the existing cluster.
#########################################################################
###################################


- RUNNING INSTALLER: Airgap

- IMAGE: 10.144.0.115:443/xnicv2:airgap

- CNI PLUGIN: CILIUM

- SWXTCH IP ADDRESS: 10.144.0.115

- AGENT TYPE: XNIC XCD

=========================================================
      Adjusting BPF filter priority on Cilium
=========================================================
Setting flag "bpf-filter-priority" to "50000"
configmap/cilium-config patched
Done!

=========================================================
             Restarting Cilium Agents
=========================================================
daemonset.apps/cilium restarted
daemonset.apps/cilium-node-init restarted
Waiting for Cilium Agents to be fully UP and Running......OK
Done!
Proceeding with xNIC Installation

=========================================================
             Creating xNIC ConfigMap
=========================================================
configmap/xnic-config created

=========================================================
                Installing xNIC
=========================================================
daemonset.apps/swxtch-xnic created
Done!

==================== Completed! ======================
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
Please allow a minute for the xNIC DaemonSet to fully spin up before
starting to use it.
Feel free to follow up on the xNIC Agents installation by running

 kubectl logs -n kube-system daemonsets/swxtch-xnic -f
```

6. Run the following script to view **kubectl** logs in the Bash window in Azure:

Bash | Copy

```
kubectl logs -n kube-system daemonsets/swxtch-xnic -f
```

7. Use the command below to follow the AKS node status in the Bash window in Azure and check if they have started:

Example:

Bash | Copy

```
user@Azure:~$ kubectl get pods -l app=swxtch-xnic -n kube-system
NAME                READY   STATUS    RESTARTS   AGE
swxtch-xnic-fc58t   1/1     Running   0          11d
swxtch-xnic-kn9hg   1/1     Running   0          11d
```

8. Sign into your cloudSwXtch and enter in the following command to see the new instances in swXtch-top.

Bash | Copy

```
sudo /swxtch/swxtch-top dashboard --swxtch localhost
```

Restarting xNIC services

To restart xNIC services for K8s, run the following command:

Bash | Copy

```
kubectl rollout restart ds/swxtch-xnic -n kube-system
```

Managing Multicast Traffic

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Following are some `tc` commands that can be useful when it comes to allowing/denying either incoming or outgoing multicast traffic on producer and consumer pods. You **must** run these commands **inside** the target producer/consumer pods so that the correct interface name (eth0 in the examples) is picked up.

By default, **ALL** multicast traffic is **allowed** on every pod.

For Outgoing (Traffic leaving the Pod)

Deny ALL outgoing multicast

To deny all outgoing multicast, use the following commands:

Specific syntax:

Bash Copy

```
# DENY ALL OUTGOING
tc qdisc add dev eth0 root handle 1: prio
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst
224.0.0.0/4 action drop
```

Alternatively, users can deny outgoing multicast to specific groups:

General Syntax:

Bash Copy

```
# DENY OUTGOING TO SPECIFIC GROUP(S)
tc qdisc add dev eth0 root handle 1: prio
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst
<multicast_group_0> action drop
...
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst
<multicast_group_n> action drop
```

Example: denying outgoing traffic to multicast group `239.0.0.1` :

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```bash
tc qdisc add dev eth0 root handle 1: prio
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst
239.0.0.1/32 action drop
```

Allow outgoing multicast to a specific group(s) - Deny any other

```bash
# DENY ALL OUTGOING
tc qdisc add dev eth0 root handle 1: prio
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst
224.0.0.0/4 action drop

# ALLOW SPECIFIC GROUP(S)
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst
<multicast_group_0> action ok
...
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst
<multicast_group_n> action ok
```

Example: allowing outgoing traffic ONLY to multicast group `239.0.0.1`:

```bash
tc qdisc add dev eth0 root handle 1: prio
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst
224.0.0.0/4 action drop
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst
239.0.0.1/32 action ok
```

Incoming (Traffic entering the Pod)

To deny ALL incoming multicast, use the following command:

Specific syntax:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
# DENY ALL INCOMING
tc qdisc add dev eth0 ingress
tc qdisc add dev eth0 parent ffff: protocol ip u32 match ip dst
224.0.0.0/4 action drop
```

Alternatively, users can deny incoming multicast for a specific group(s)

General syntax:

Bash Copy

```
# DENY INCOMING TO SPECIFIC GROUP(S)
tc qdisc add dev eth0 ingress
tc filter add dev eth0 parent ffff: protocol ip u32 match ip dst
<multicast_group_0> action drop
...
tc filter add dev eth0 parent ffff: protocol ip u32 match ip dst
<multicast_group_n> action drop
```

Example: denying incoming multicast traffic to multicast group `239.0.0.1` :

Bash Copy

```
tc qdisc add dev eth0 ingress
tc filter add dev eth0 parent ffff: protocol ip u32 match ip dst
239.0.0.1/32 action drop
```

In addition, users can specify allowing incoming multicast by a specific group(s) while denying any other:

General syntax:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
# DENY ALL INCOMING
tc qdisc add dev eth0 ingress
tc qdisc add dev eth0 parent ffff: protocol ip u32 match ip dst
224.0.0.0/4 action drop

# ALLOW SPECIFIC GROUP(S)
tc filter add dev eth0 parent ffff: protocol ip u32 match ip dst
<multicast_group_0> action ok
...
tc filter add dev eth0 parent ffff: protocol ip u32 match ip dst
<multicast_group_n> action ok
```

Example: allowing incoming traffic **ONLY** to multicast group `239.0.0.1`:

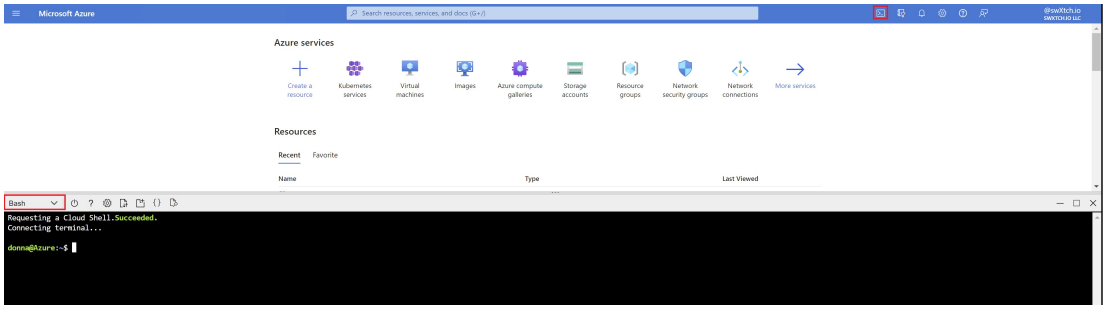
Bash Copy

```
tc qdisc add dev eth0 ingress
tc qdisc add dev eth0 parent ffff: protocol ip u32 match ip dst
224.0.0.0/4 action drop
tc filter add dev eth0 parent ffff: protocol ip u32 match ip dst
239.0.0.1/32 action ok
```

Accessing an xNIC Pod

At times, it is nice to be able to get into the pod and be able to run commands such as swxtch-tcpdump. To accomplish this, follow these steps:

1. Sign into **your desired cloud**.
2. Open **cloudShell** as Bash. In this example, the user is using Azure.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. Enter in the following command to get the pod name:

```bash
kubectl get pods -l app=swxtch-xnic -n kube-system
```

Example:

```bash
user@Azure:~$ kubectl get pods -l app=swxtch-xnic -n kube-system
NAME                READY   STATUS    RESTARTS   AGE
swxtch-xnic-fc58t   1/1     Running   0          11d
swxtch-xnic-kn9hg   1/1     Running   0          11d
```

4. Enter in the following command, replacing **Pod** with the pod name:

```bash
kubectl exec -it pod/swxtch-xnic-name -n kube-system -- bash
```

Example:

```bash
user@Azure:~$ kubectl exec -it pod/swxtch-xnic-kn9hg -n kube-system -
- bash
root@aks-nodepool1-23164585-vmss00000A:/
```

You can now enter in commands similar to any VM Node, such as "**ip a**" or "**sudo swxcth-tcpdump -i eth0**". Note that the pods created in this example do not have tools such as the standard tcpdump. However, **swxtch-tcpdump** will work. For testing use swxtch-perf, see swxtch-top under Testing cloudSwXtch.

> **Switching Contexts**
> If you have more than one AKS Kubernetes services, then you may need to change the context to work on the desired instance. For more information, please review the Changing K8s Context in Your Preferred Cloud section.

Signing into K8s node instance in Bash

If you want to sign into your preferred cloud instance and access the CloudShell as Bash, complete the following:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. Ensure the pods are running using this command:

```bash
kubectl get pods -l app=swxtch-xnic -n kube-system
```

Example:

```bash
user@Azure:~$ kubectl get pods -l app=swxtch-xnic -n kube-system
NAME                READY   STATUS    RESTARTS   AGE
swxtch-xnic-fc58t   1/1     Running   0          11d
swxtch-xnic-kn9hg   1/1     Running   0          11d
```

2. Run this command to sign into the pod:

```bash
kubectl exec -it pod/<cloudSwXtch-xnic-name> -n kube-system -- bash
```

Example:

```bash
user@Azure:~$ kubectl exec -it pod/swxtch-xnic-kn9hg -n kube-system -
- bash

root@aks-nodepool1-23164585-vmss00000A:/#
```

3. Now you can run any command for example "ip a" or run swxtch-perf (or a user application) as a consumer or producer to run tests.

Accessing xNIC Logs

You can get xNIC logs once signed in to the pod. See How to Find xNIC Logs and follow directions for xNIC.

Using xNIC config

Getting to the xNIC config is available once you're signed into the Pod. To get to the xNIC config, use the command below:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash	Copy

```
cat /var/opt/swxtch/swxtch-xnic.conf
```

Exiting the Pod

To exit the pod, enter in the following command:

Bash	Copy

```
exit
```

To Change K8s Context in Your Preferred Cloud

If there are more than one K8s instances in your preferred cloud then you may need to be able to switch between them to run commands in the CloudShell Bash. Below are steps to switch between K8s instances.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. Get a list of all K8s Contexts by using the following command:

```bash
kubectl config get-contexts
```

Example in Azure:

```
user@Azure:~$ kubectl config get-contexts

CURRENT   NAME                 CLUSTER              AUTHINFO
NAMESPACE

          cilium-sample        cilium-sample
clusterUser_saDevNetwork_cilium-sample

          cilium-sample-200    cilium-sample-200    clusterUser_test-
donna-200-rg_cilium-sample-200

          cilium-sample2       cilium-sample2
clusterUser_saDevNetwork_cilium-sample2

*         cilium-sample300     cilium-sample300     clusterUser_test-
donna-300-rg_cilium-sample300

          dsd-k8-cluster-100   dsd-k8-cluster-100
clusterUser_saDevNetwork_dsd-k8-cluster-100
```

Notice in the above list there are multiple context but only one has the asterisks (*).

2. To change context, run the following command. The example is changing to cilium-sample2.

```bash
kubectl config use-context cilium-sample2
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. Re-run the **get-context** command:

```bash
kubectl config get-contexts
```

Example in Azure:

```
user@Azure:~$ kubectl config get-contexts

CURRENT   NAME                 CLUSTER              AUTHINFO
NAMESPACE

          cilium-sample        cilium-sample
clusterUser_saDevNetwork_cilium-sample

          cilium-sample-200    cilium-sample-200    clusterUser_test-
donna-200-rg_cilium-sample-200

*         cilium-sample2       cilium-sample2
clusterUser_saDevNetwork_cilium-sample2

          cilium-sample300     cilium-sample300     clusterUser_test-
donna-300-rg_cilium-sample300

          dsd-k8-cluster-100   dsd-k8-cluster-100
clusterUser_saDevNetwork_dsd-k8-cluster-100
```

As you can see above, the asterisk (*) has changed positions to the desired context, cilium-sample2.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Uninstall xNIC on K8s

WHAT TO EXPECT

In this article, users will learn how to uninstall xNIC on Kubernetes (K8s).

To uninstall xNIC on K8s, please follow these steps:

1. Sign into **Cloud**.
2. Open **cloudShell** as Bash.
3. Run the following command in the terminal:

   Shell

Bash	Copy
`./xnic_ds_installer.sh -u`	

4. xNIC on K8s should now be uninstalled.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Test xNIC with K8s

WHAT TO EXPECT

Before running your application in your preferred cloud, it is a good idea to test with swXtch.io's provided tools/examples.

In this article, you will learn how to test xNIC with K8s. Please complete the installation process outlined in **Install xNIC on K8s** before you begin testing.

STEP ONE: Create A Consumer

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. Create a **TestConsumer.yaml** file using the example below.

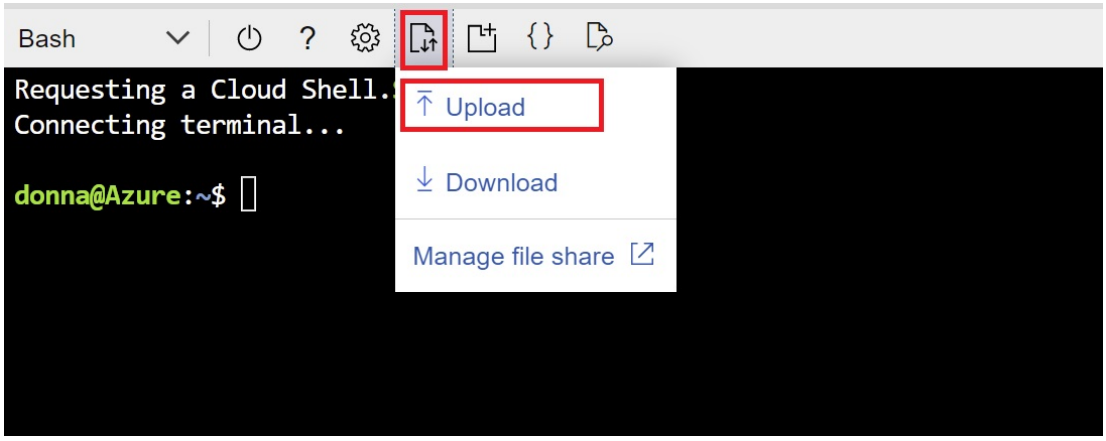
   Replace the **XNIC_SWXTCH_ADDR** with the cloudSwXtch control address.

Bash | Copy

```
apiVersion: v1
kind: Pod
metadata:
  name: consumer-a
  labels:
    app: consumer-a
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - producer-a
            - consumer-b
        topologyKey: kubernetes.io/hostname
  containers:
  - name: consumer-a
    image: ubuntu:20.04
    securityContext:
      privileged: true
    env:
    - name: IS_DAEMON
      value: "false"
    - name: PERF_TYPE
      value: "consumer"
    - name: PERF_NIC
      value: "eth0"
    - name: PERF_MCGIP
      value: "239.0.0.10"
    - name: PERF_MCGPORT
      value: "8410"
    - name: XNIC_SWXTCH_ADDR
      value: "10.224.0.115"
    command: ["/bin/bash"]
    args: ["-c", "apt update && apt install curl -y;
                  curl
http://$(XNIC_SWXTCH_ADDR)/services/install/swxtch-xnic-k8s-
install.sh --output swxtch-xnic-k8s-install.sh;
                  chmod +x swxtch-xnic-k8s-install.sh;
                  ./swxtch-xnic-k8s-install.sh -v 2;
                  sleep infinity"]
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Upload the file into the Azure CloudShell.



STEP TWO: Create a Producer

1. Create a **TestProducer.yaml** file using the example below.

   Replace **XNIC_SWXTCH_ADDR** with the cloudSwXtch control address.

   Shell

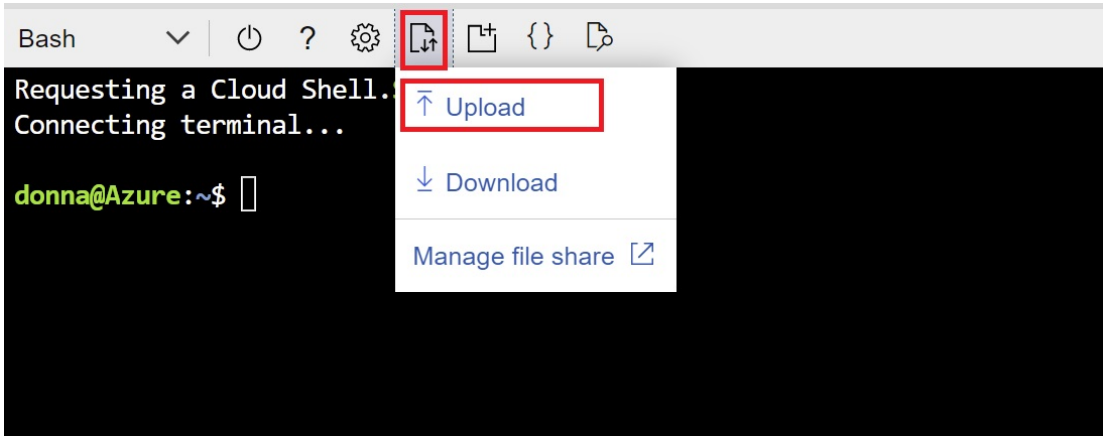
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
apiVersion: v1
kind: Pod
metadata:
  name: producer-a
  labels:
    app: producer-a
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - consumer-a
            - producer-b
        topologyKey: kubernetes.io/hostname
  containers:
  - name: producer-a
    image: ubuntu:20.04
    securityContext:
      privileged: true
    env:
    - name: IS_DAEMON
      value: "false"
    - name: PERF_TYPE
      value: "producer"
    - name: PERF_NIC
      value: "eth0"
    - name: PERF_MCGIP
      value: "239.0.0.10"
    - name: PERF_MCGPORT
      value: "8410"
    - name: PERF_PPS
      value: "100"
    - name: XNIC_SWXTCH_ADDR
      value: "10.224.0.115"
    command: ["/bin/bash"]
    args: ["-c", "apt update && apt install curl -y;
                  curl
http://$(XNIC_SWXTCH_ADDR)/services/install/swxtch-xnic-k8s-
install.sh --output swxtch-xnic-k8s-install.sh;
                  chmod +x swxtch-xnic-k8s-install.sh;
                  ./swxtch-xnic-k8s-install.sh -v 2;
                  sleep infinity"]
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Upload the file into the Azure CloudShell.



STEP THREE: Run Test

- Run the producer by running this command in your preferred cloud's cloudShell Bash window.

  Wait for the cursor to return to know it is fully created.

  Shell

Bash	Copy
`kubectl create -f TestProducer.yaml`	

- Run the consumer by running this command in your preferred cloud'scloudShell bash window.

  Wait for the cursor to return to know it is fully created.

Bash	Copy
`kubectl create -f TestConsumer.yaml`	

Validate they are running using this command:

Bash	Copy
`kubectl get pods -o wide -A`	

Below is an example in Azure showing the consumer-a and producer-a running:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
donna@Azure:~$ kubectl get pods -o wide -A
NAMESPACE     NAME                                  READY   STATUS
RESTARTS   AGE   IP             NODE
NOMINATED NODE   READINESS GATES
kube-system   cilium-node-init-kbql4                1/1     Running
0          27h   10.2.128.101   aks-nodepool1-23164585-vmss00000j
<none>           <none>
kube-system   cilium-node-init-sg4vc                1/1     Running
0          27h   10.2.128.100   aks-nodepool1-23164585-vmss00000i
<none>           <none>
kube-system   cilium-nx7vl                          1/1     Running
0          27h   10.2.128.100   aks-nodepool1-23164585-vmss00000i
<none>           <none>
kube-system   cilium-operator-6485c89c66-748tz      1/1     Running
0          27h   10.2.128.101   aks-nodepool1-23164585-vmss00000j
<none>           <none>
kube-system   cilium-vv4qs                          1/1     Running
0          27h   10.2.128.101   aks-nodepool1-23164585-vmss00000j
<none>           <none>
kube-system   cloud-node-manager-mncgk              1/1     Running
0          27h   10.2.128.100   aks-nodepool1-23164585-vmss00000i
<none>           <none>
kube-system   cloud-node-manager-qg5wf              1/1     Running
0          27h   10.2.128.101   aks-nodepool1-23164585-vmss00000j
<none>           <none>
kube-system   coredns-autoscaler-569f6ff56-qtqpr    1/1     Running
0          28h   10.0.0.121     aks-nodepool1-23164585-vmss00000i
<none>           <none>
kube-system   coredns-fb6b9d95f-blk6j               1/1     Running
0          28h   10.0.0.236     aks-nodepool1-23164585-vmss00000i
<none>           <none>
kube-system   coredns-fb6b9d95f-pxzh2               1/1     Running
0          28h   10.0.0.131     aks-nodepool1-23164585-vmss00000i
<none>           <none>
```

You can also validate it is working by running logs with this command:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash	Copy

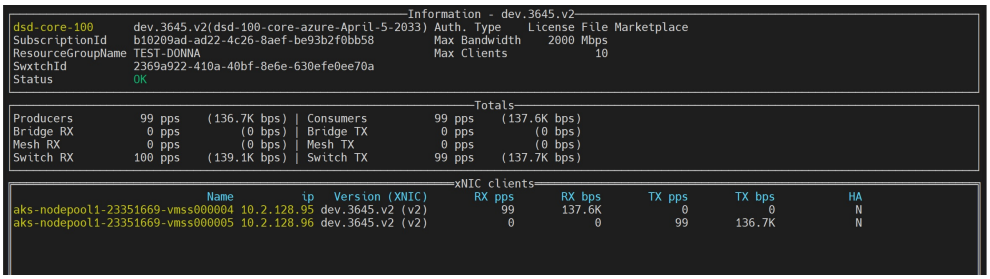
```
kubectl logs pods/producer-a -f
```

- Log into your cloudSwXtch and run this command:

Bash	Copy

```
sudo /swxtch/swxtch-top dashboard --swxtch localhost
```

1. swXtch-top should show the producer and the consumer. This may take a minute to completely show all metrics.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- Stop the test consumer by running this command back in your preferred cloud's CloudShell bash window.
    1. Wait for the cursor to return to know it is deleted fully.

```
Bash                                                    Copy

kubectl delete -f TestConsumer.yaml
```

    2. swXtch-top should no longer show the consumer. This may take a minute to display. Additionally, running **kubectl get pods -o wide** should now show just the test consumer as shown below:

```
Bash                                                    Copy

donna@Azure:~$ kubectl get pods -o wide -A
NAME                READY   STATUS        RESTARTS   AGE   IP
NODE                                NOMINATED NODE
READINESS GATES
producer-a          1/1     Terminating   0          15m
10.0.1.90     aks-nodepool1-23351669-vmss000005   <none>
<none>
swxtch-xnic-46qgg   1/1     Running       0          39m
10.2.128.96   aks-nodepool1-23351669-vmss000005   <none>
<none>
swxtch-xnic-szdk7   1/1     Running       0          40m
10.2.128.95   aks-nodepool1-23351669-vmss000004   <none>
<none>
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- Stop the test producer by running this command in your preferred cloud's CloudShell bash window.
    1. Wait for the cursor to return to know its fully deleted.

```
Bash                                                    Copy

kubectl delete -f TestProducer.yaml
```

    2. swXtch-top should no longer show the producer. This may take a minute to display. Additionally, running **kubectl get pods -o wide** should now show just the test producer as shown below:

```
Bash                                                    Copy

donna@Azure:~$ kubectl get pods -o wide -A
NAME                READY   STATUS    RESTARTS   AGE   IP
NODE                                  NOMINATED NODE
READINESS GATES
swxtch-xnic-46qgg   1/1     Running   0          42m
10.2.128.96   aks-nodepool1-23351669-vmss000005   <none>
<none>
swxtch-xnic-szdk7   1/1     Running   0          42m
10.2.128.95   aks-nodepool1-23351669-vmss000004   <none>
<none>
```

Now that the system is validated using swXtch.io, you can test with your K8s application.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Upgrade xNIC nodes on K8s

WHAT TO EXPECT

The nodes upgrade is automatic based on the restart of the nodes containing the xNIC. This can be done by **restarting the xNIC Daemonset**.

In this article, you will learn how you can use this method to upgrade your xNIC nodes on K8s to match the version of your cloudSwXtch.

Before you upgrade the xNIC nodes on K8s, you need to upgrade the cloudSwXtch to the latest version. See Upgrading cloudSwXtch for more information

Restarting the xNIC Daemon set

1. Sign into your preferred cloud's portal.
2. Open cloudShell as Bash.
3. Run the following command:

Bash	Copy
`kubectl rollout restart ds swxtch-xnic -n kube-system`	

    a. This will restart the Kubernetes swxtch-xnic daemon set and update the version of the xNIC to match that of the cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

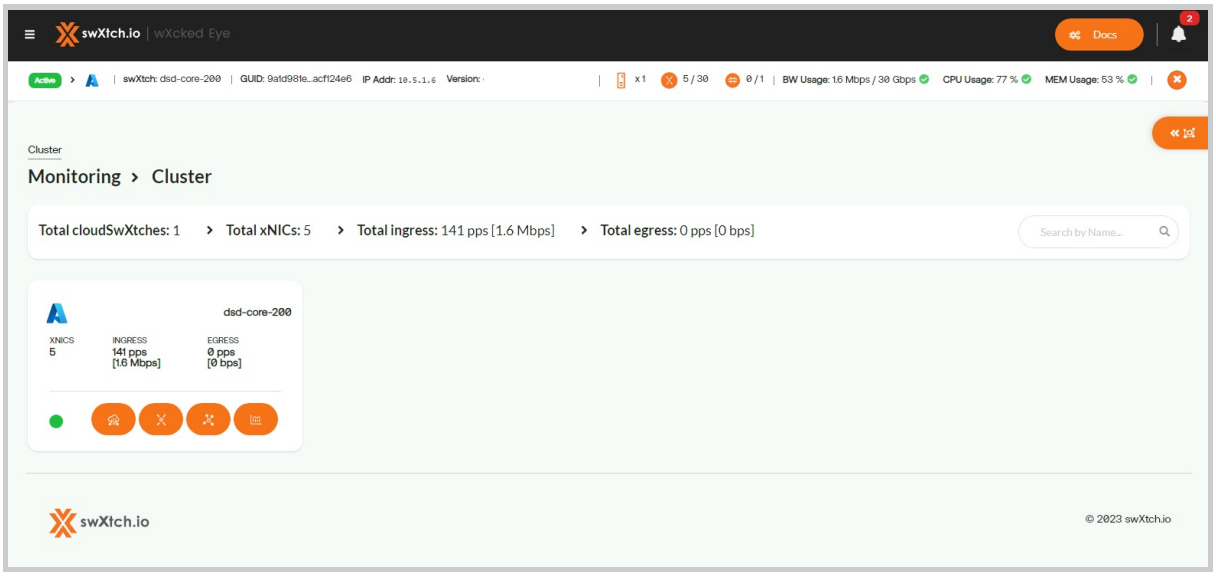
Using wXcked Eye for cloudSwXtch

WHAT TO EXPECT

In this article, you will learn more about wXcked Eye and the benefits of using it to configure and monitor your cloudSwXtch environment.

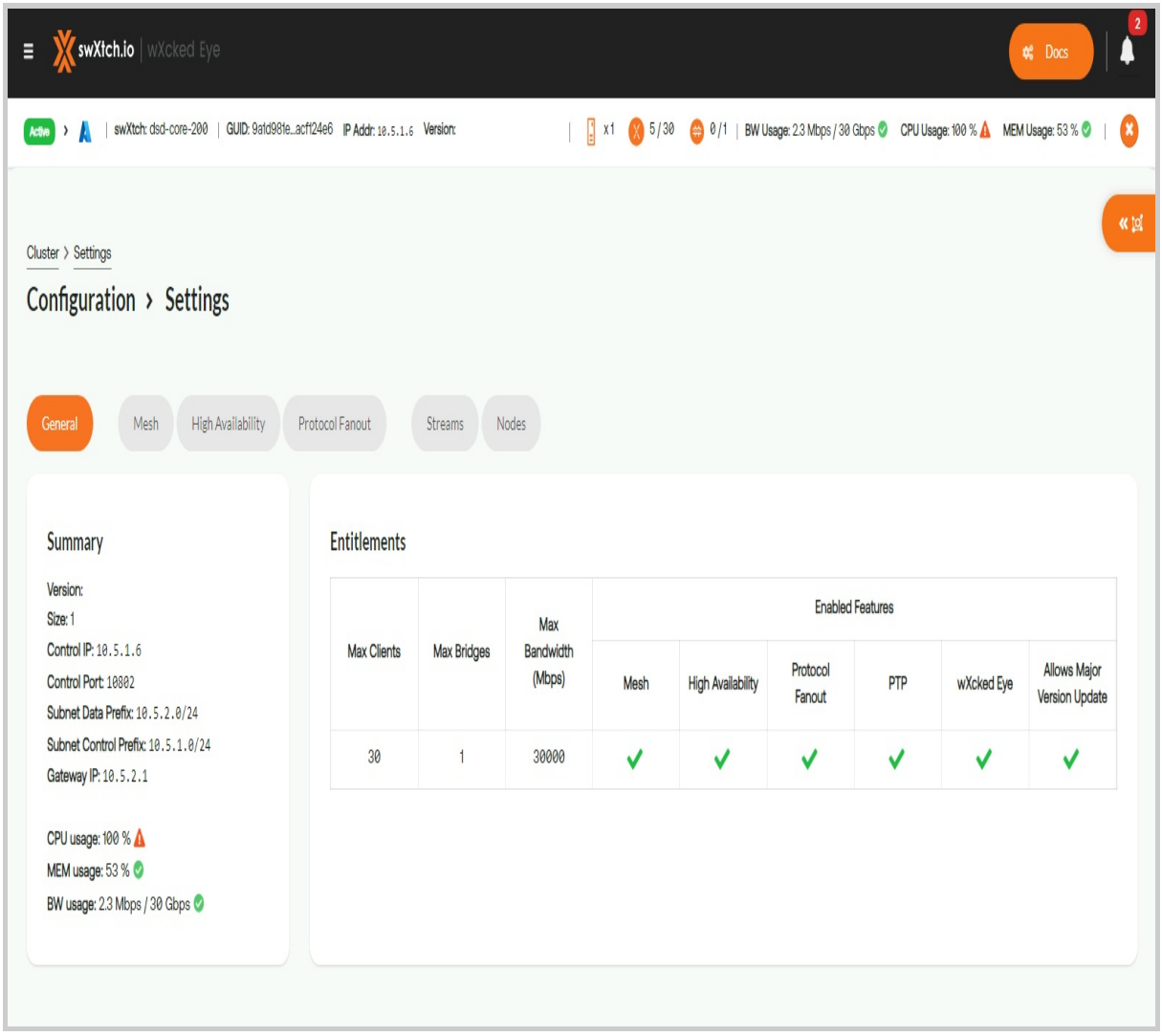
What is wXcked Eye?

wXcked Eye is a web-based monitoring and configuration tool for cloudSwXtch. It presents users with a high-level view of their cloudSwXtch environment with an interactive graph detailing connections to different endpoints. With an expansive look at performance metrics, users can ensure that their data is flowing as expected.



In addition, wXcked Eye unlocks the ability to configure Mesh, High Availability, Protocol Fanout and Conversion, and Precision Time Protocol (PTP) from the comfort of a web browser.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



How to Access wXcked Eye

To access the wXcked Eye UI, users will need to enter the following URL into a web browser of a VM in the cloudswXtch environment. They should use the IP address of their cloudSwXtch to prefix the URL.

Bash	Copy

```
<swxtch-ip-address>/wxckedeye/
```

Monitor cloudSwXtch with wXcked Eye

To learn more about the monitoring capabilities of wXcked Eye such as the Network Graph or the cloudSwXtch and xNIC metrics views, see Monitor cloudSwXtch with wXcked Eye.

Configure cloudSwXtch with wXcked Eye

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

To learn more about the configuration capabilities of wXcked Eye such as Mesh, High Availability, Protocol Fanout and Precision Time Protocol, see Configure cloudSwXtch with wXcked Eye.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Monitor cloudSwXtch with wXcked Eye

WHAT TO EXPECT

The **wXcked Eye** UI provides users with an additional way to monitor the performance of their cloudSwXtch network.

To learn more about how to configure your cloudSwXtch for mesh, high availability and protocol fanout with wXcked eye, please read the "**Configure cloudSwXtch with wXcked Eye**" article.

In this section, we will walk through the user interface, explaining overall functionality and how it provides users with additional control over their cloudSwXtch network.

Accessing the wXcked Eye UI

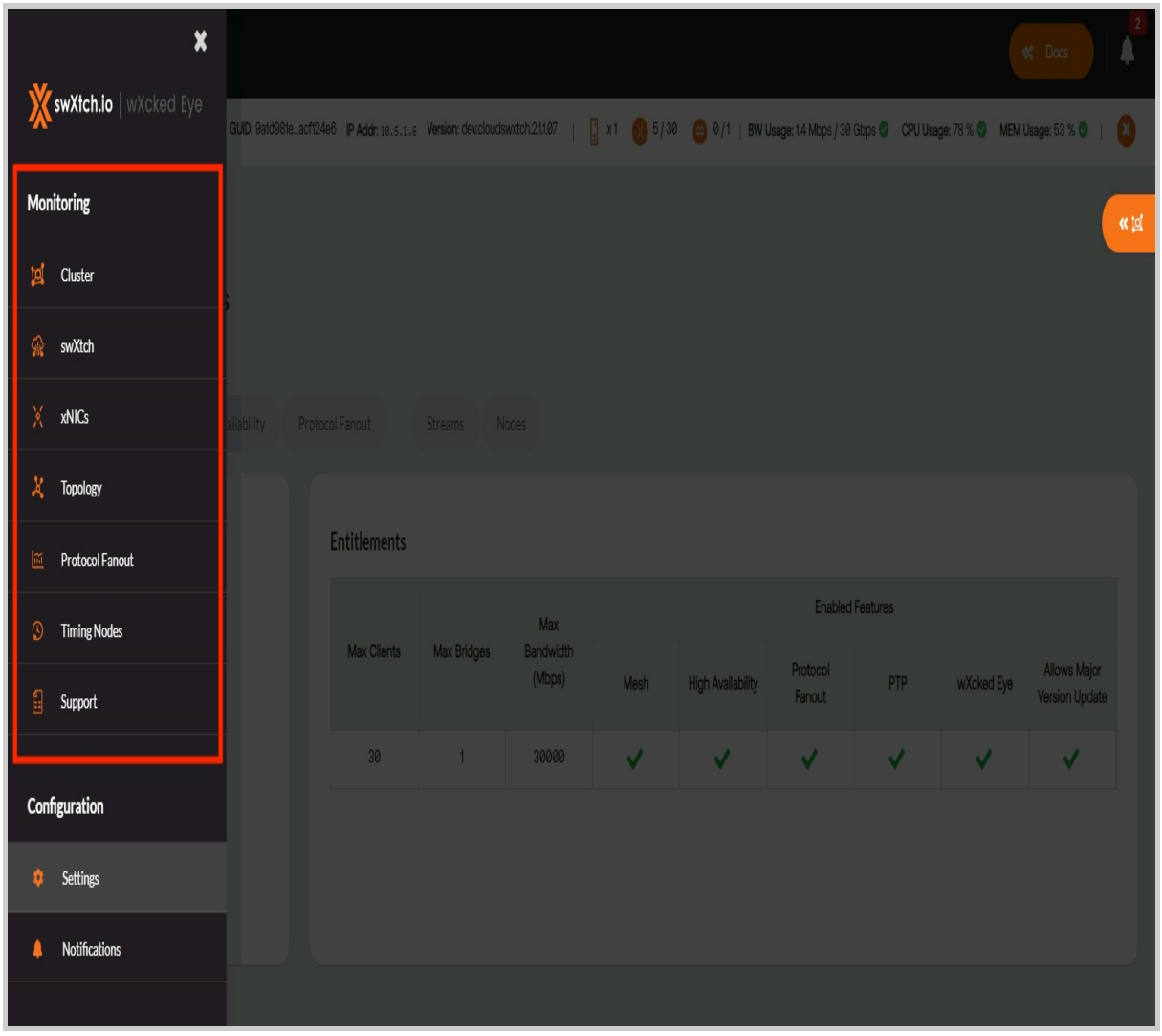
To access the wXcked Eye UI, users will need to enter the following URL into a web browser of a VM in their cloudswXtch environment. They should use the IP address of their cloudSwXtch to prefix the URL.

PowerShell | Copy

```
<cloudSwXtch-ip-address>/wxckedeye/
```

Navigating the Monitoring pages

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



The wXcked Eye's monitoring capabilities are organized into seven pages. For more information on a page's contents, please view their respective articles.

- Cluster View
- cloudSwXtch Stats
- xNICs Stats
- Topology
- Protocol Fanout Stats
- Timing Nodes
- Support

Note: The Cluster view is the default main page for wXcked Eye and will be the first thing users see when using the UI.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

wXcked Eye Cluster Page

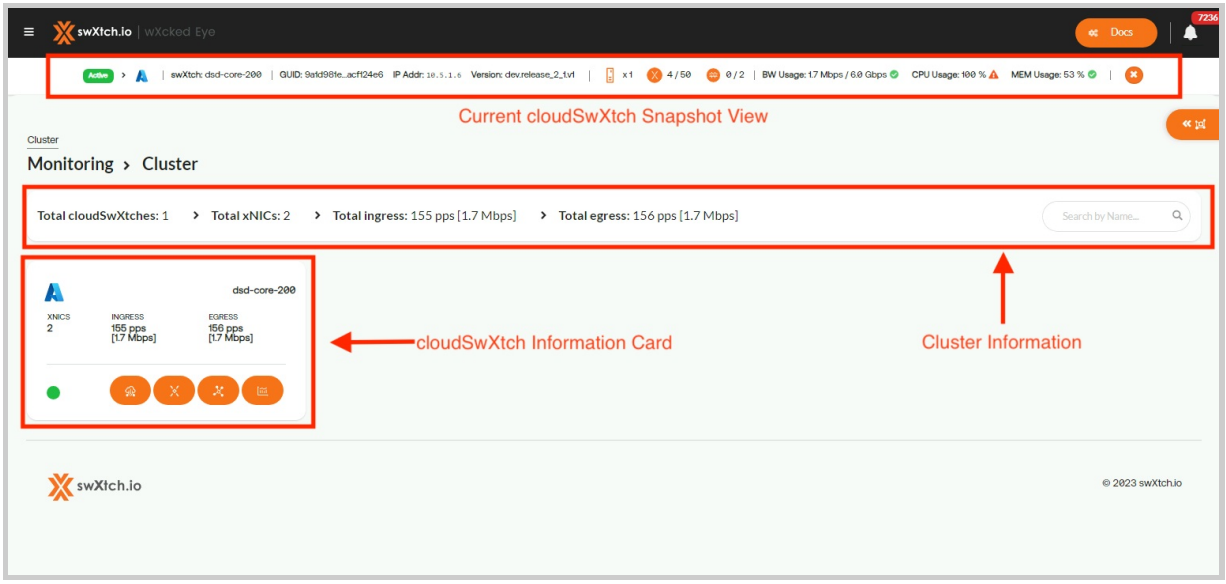
WHAT TO EXPECT

The Cluster page in wXcked Eye provides user with a high-level view of all their connected cloudSwXtches. Users can easily go between different cloudSwXtches and view their stats and the stats of associated xNICs. In addition, a new generalized side panel allows users to jump between cloudSwXtches.

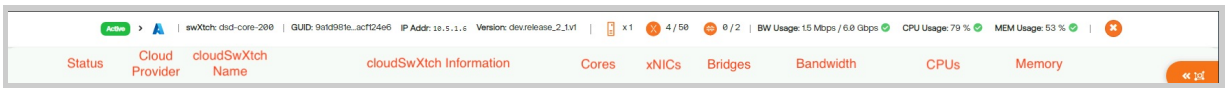
In this section, we will walk you through the Cluster page and how it improves navigation between cloudSwXtches.

Introduction

The **Cluster** page is the main page of wXcked Eye. It segments cloudSwXtches into byte sized cards with some top level information, such as the name, the cloud provider, the status, the number of attached xNICs, and the flow of data for both Ingress and Egress in packets per seconds (pps). The **Cluster Information panel** shows the total number of cloudSwXtches and xNICs with the cumulative total in pps for both Ingress and Egress.



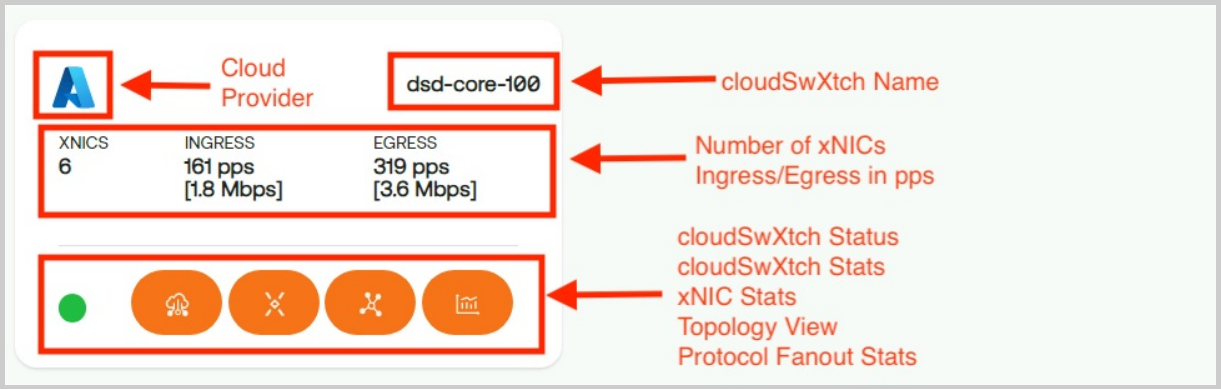
At the top of each page in wXcked Eye, the Current cloudSwXtch Snapshot banner will display relevant information regarding the cloudSwXtch a user is currently on.



A cloudSwXtch's membership in a Cluster is dependent on whether or not it is involved in a mesh or high availability configuration. For more information on how to set up mesh and high availability in wXcked Eye, please visit their respective articles under Configure cloudSwXtch in wXcked Eye.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

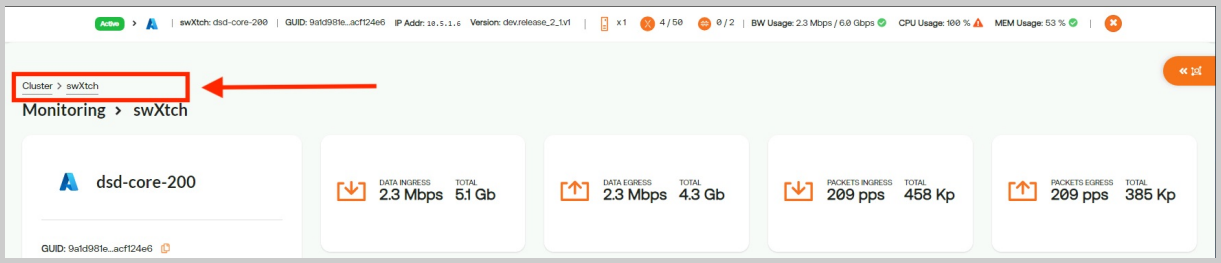
Navigating using cloudSwXtch Information Card



In each cloudSwXtch Information card, users will find four buttons linking you to additional information and metrics within wXcked Eye. This includes:

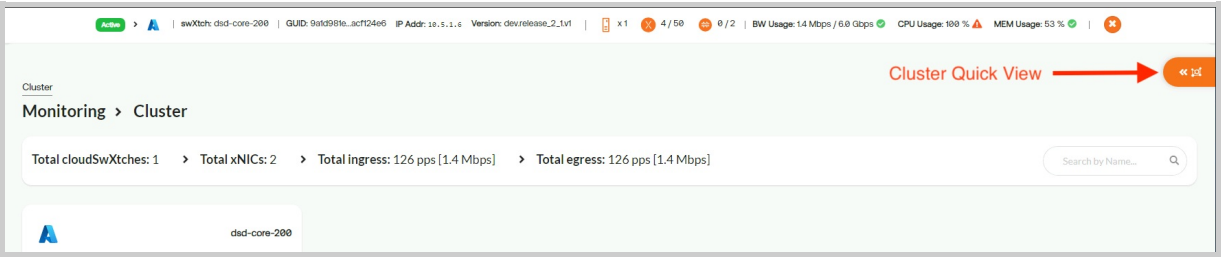
- cloudSwXtch Stats
- xNIC Stats
- Topology
- Protocol Fanout Stats

A user can return to the cluster view by selecting "Cluster" in the breadcrumb trail at the top of the page.



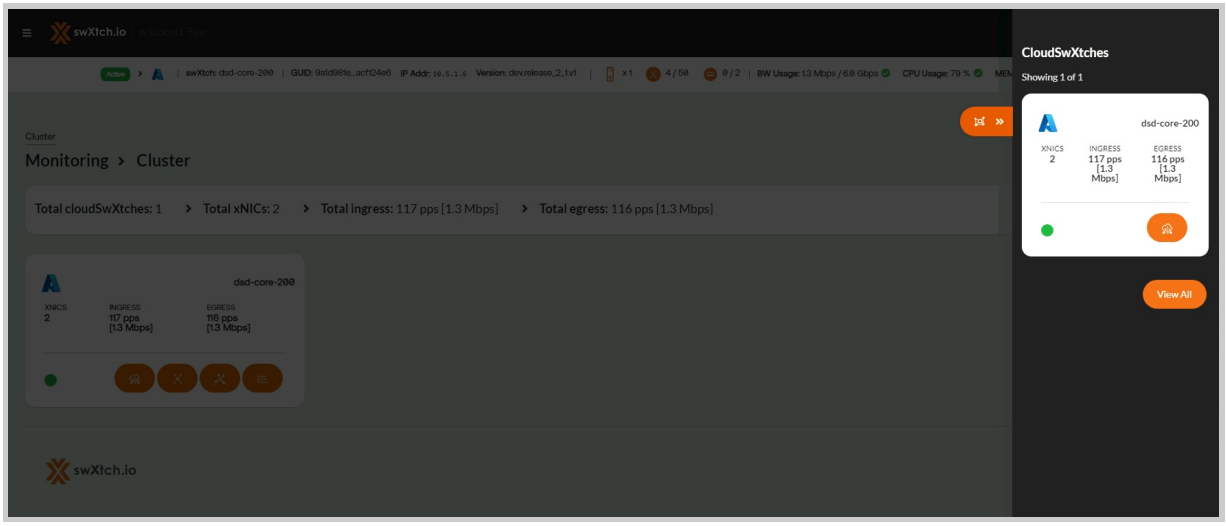
Cluster Quick View Panel

A user can quickly access a condensed list of up to 5 cloudSwXtches from any page in wXcked Eye by using the new **Cluster Quick View** side panel at the top right hand corner. The selected swXtch will be displayed at the top of the page.



The cloudSwXtch Information card will contain the same metrics displayed in the Cluster Main page and the cloudSwXtch Stats navigation button.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

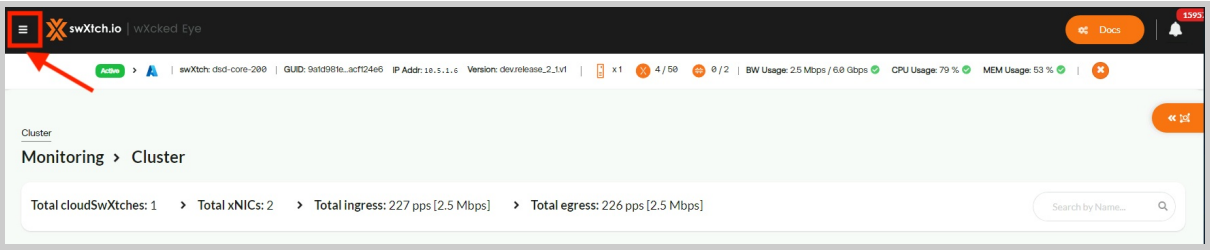
wXcked Eye cloudSwXtch Page

WHAT TO EXPECT

In this article, you will learn how to view the performance metrics for your cloudSwXtch and the xNICs associated with it.

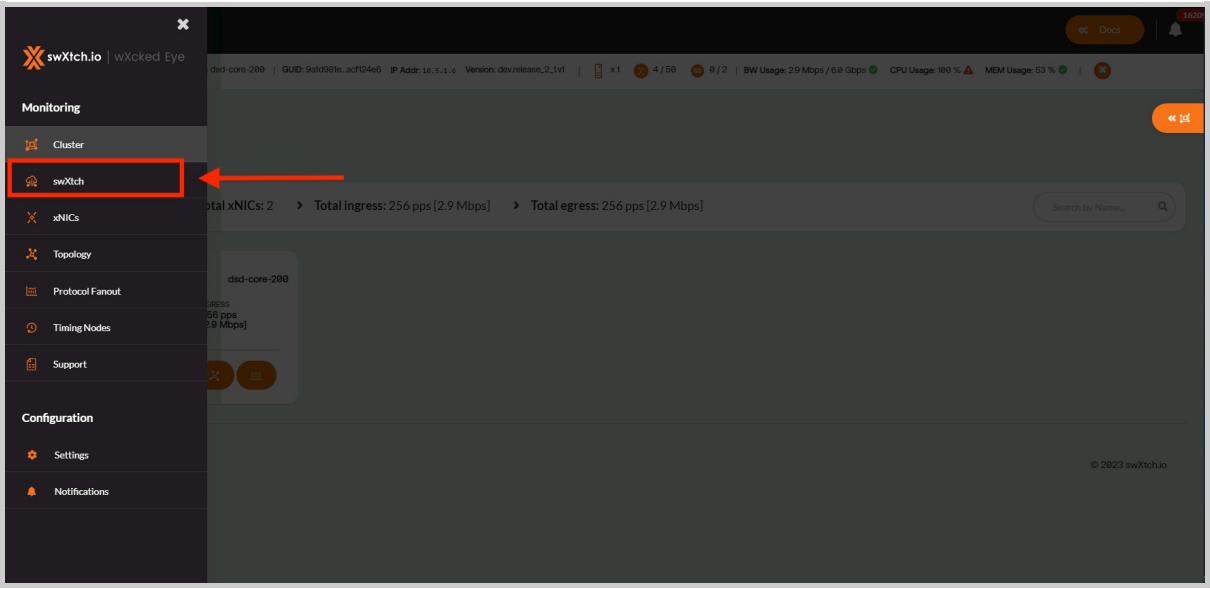
Locating the wXcked Eye cloudSwXtch Page

To navigate to the cloudSwXtch page, users will need to click on the menu (≡) option at the top left hand corner by the swXtch.io logo.



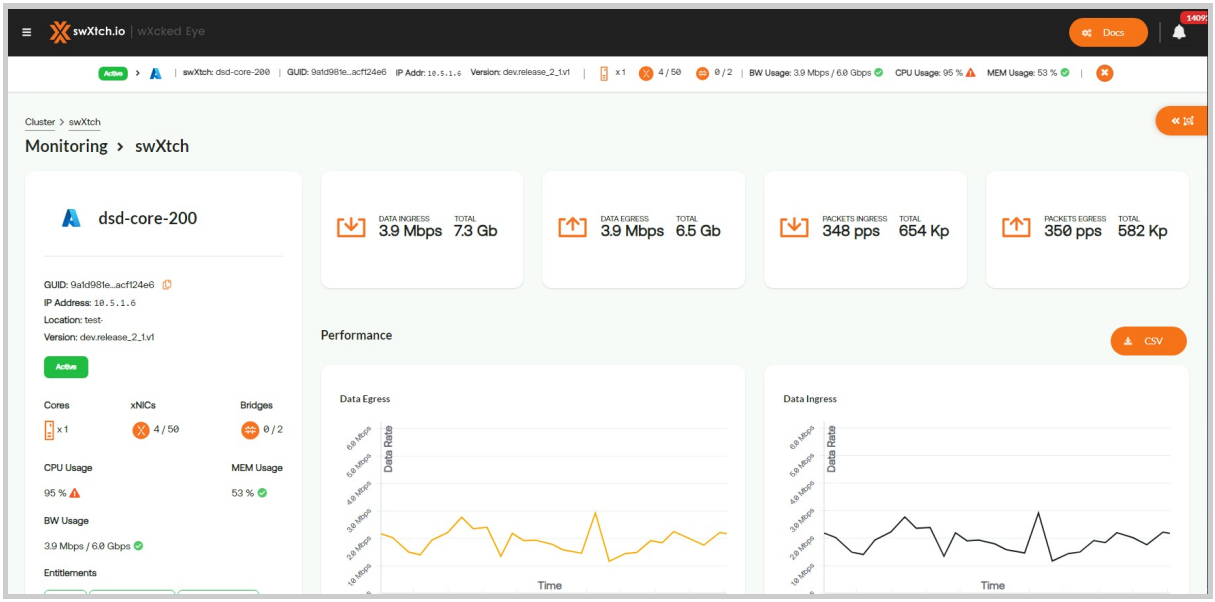
From there, select "**swXtch**" under "**Monitoring.**"

Alternatively, if a user is on the **Cluster** page, they can select the cloudSwXtch Stats button in a cloudSwXtch's Information card.



Navigating the wXcked Eye cloudSwXtch Page

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



cloudSwXtch Key Performance Metrics

Once the page loads, users will be presented a high-level view of their selected cloudSwXtch's data flow. This page provides detailed information regarding the cloudSwXtch and illustrates 4 key performance metrics:

- Data Egress
- Data Ingress
- Packets Egress
- Packets Ingress

Data egress/ingress are displayed in bits per second (bps) while packets egress/ingress are displayed in packets per second (pps). In addition to the rate, the total number of bits and packets are displayed for the user. These metrics are further explored in the **Performance** section with four related graphs and an additional Packets Dropped graph.

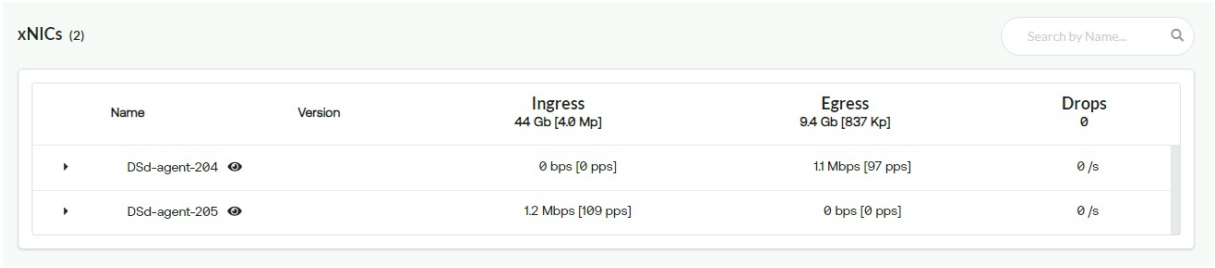
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

cloudSwXtch Information Panel

Important network information of the cloudSwXtch such as the **GUID**, **IP address**, **location (resource groups)**, **cloud provider**, **version**, & **replicator status** is shown in the top left card along with its name, which in this case is core-200. In addition, the **number of cores** and **number of associated xNICs** to the cloudSwXtch will also be displayed.

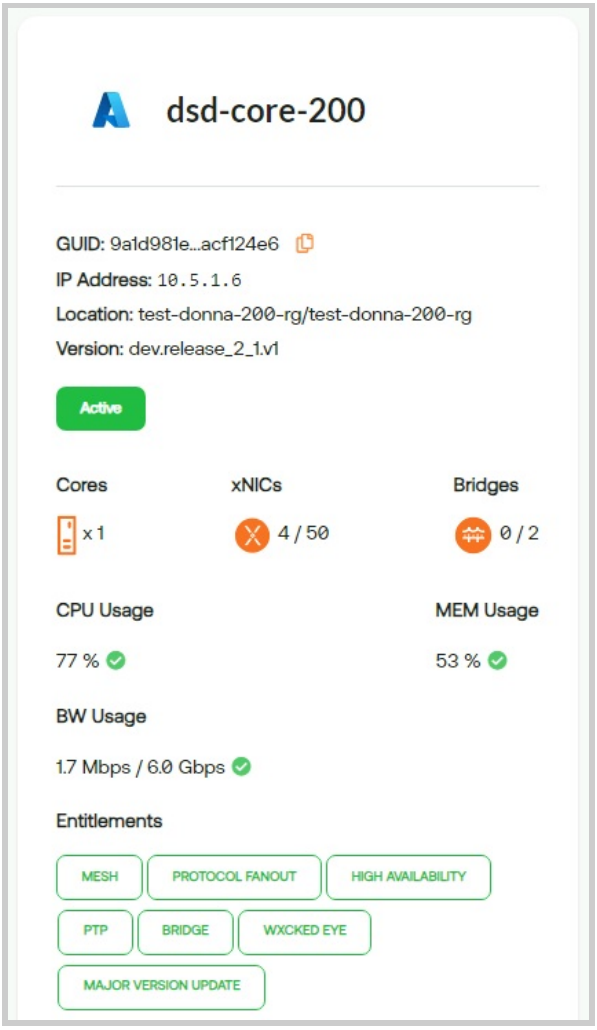
Bandwidth usage is also listed. In the event that a cloudSwXtch exceeds its allotted bandwidth, a warning symbol will appear.

xNICs Panel



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

In the bottom of the cloudSwXtch page, users will be able to see the **xNICs** panel. This panel lists the agents that are connected to the cloudSwXtch -- in this case, core-200. Each listed xNIC is accompanied with its version, ingress/egress rates, and packet drops. When using the dropdown feature for an agent, a user can see the multicast groups associated with the xNIC and its ingress/egress rates.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

wXcked Eye xNICs Page

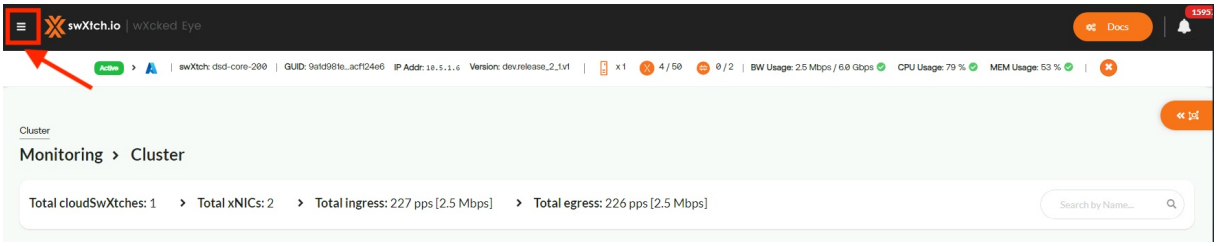
WHAT TO EXPECT

In this article, users will learn how to view performance metrics from the xNICs perspective.

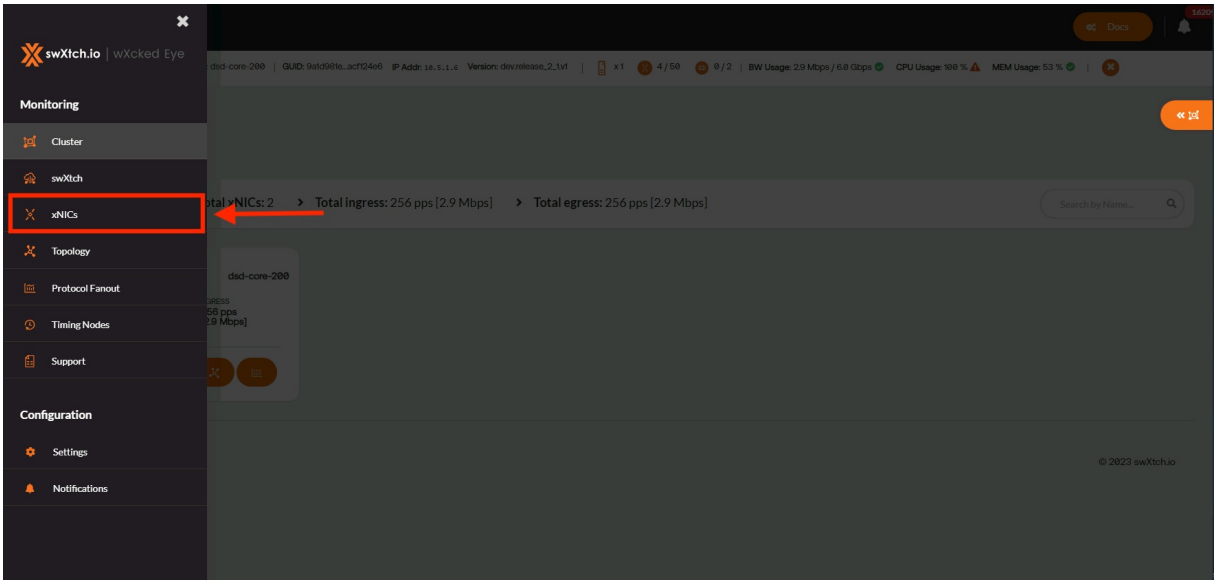
Locating the xNIC View Page

The xNIC page provides users with a look of their cloudSwXtch environment from the xNICs' perspective, breaking down performance at an agent's level.

To navigate to the xNIC page, users will need to click on the menu option at the top left hand corner by the swXtch.io logo.

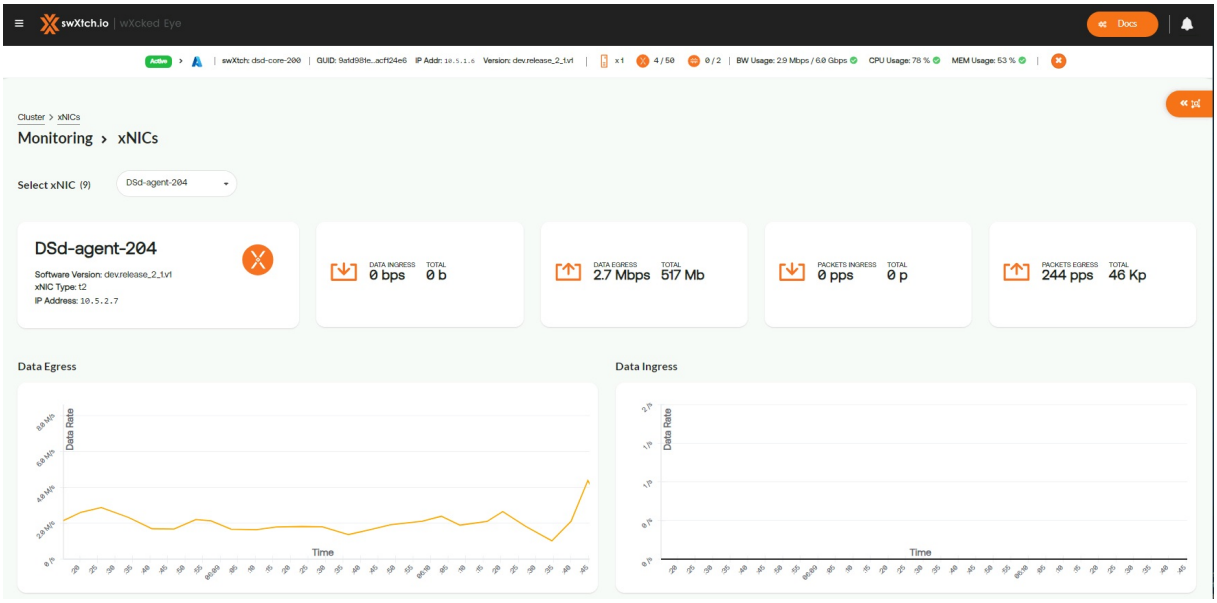


The navigation menu will open, revealing the other wXcked Eye pages. **Select xNICs** to view the xNIC page.



Navigating the xNIC page

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



At first glance, the xNIC page looks very similar to the cloudSwXtch view. However, instead of focusing on the cloudSwXtch, users are given key information and performance metrics for a single xNIC.

- **Data Ingress (bps)** - Data being consumed by the xNIC
- **Data Egress (bps)** - Data being sent from the xNIC
- **Packets Ingress (pps)** - Packets being consumed by the xNIC
- **Packets Egress (pps)** - Packets being sent from the xNIC

In the example above, one noticeable difference is the inclusion of the Select an xNIC dropdown menu next to "Select xNIC." Here, a user can select an agent they wish to monitor (DSd-agent-204).

After selecting an xNIC, the agent's information will display in the same area as the cloudSwXtch on the main page. The information includes the software version, xNIC version and the IP address.

Just like the xNIC panel in the wXcked Eye main page, users are able to see the Multicast Groups associated with the xNIC and their ingress/egress rates.

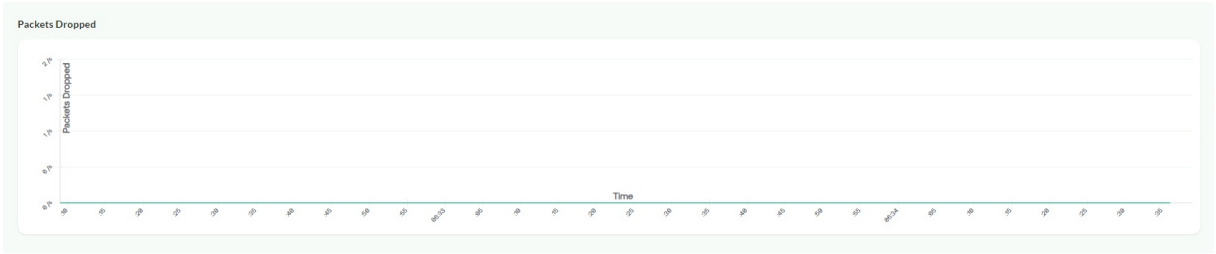
Performance

The xNIC view provides users with another way to visualize data flow. Towards the bottom of the page, users will be able to see the 5 key performance metrics displayed as active histograms. The first four deal with data and packet egress/ingress over 15 second increments.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Perfomance: Data Egress/Ingress and Packets Egress/Ingress

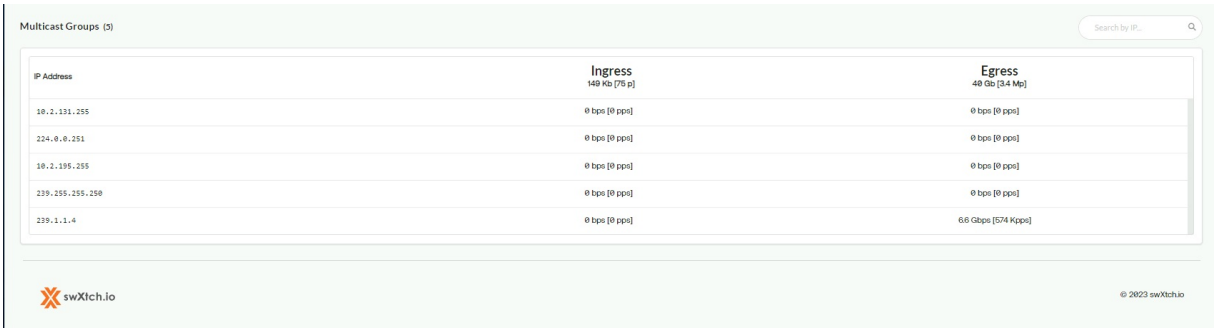
The bottom graph displays the number of packets dropped over time. A successful stream would show no packets dropping like the example below. The X-Axis is organized into 5 second increments.



Packets Dropped by Limit (:05 second increments)

Multicast Groups

The Multicast Groups panel lists the IP addresses of different data streams related to the cloudSwXtch with the ingress/egress rates displayed.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

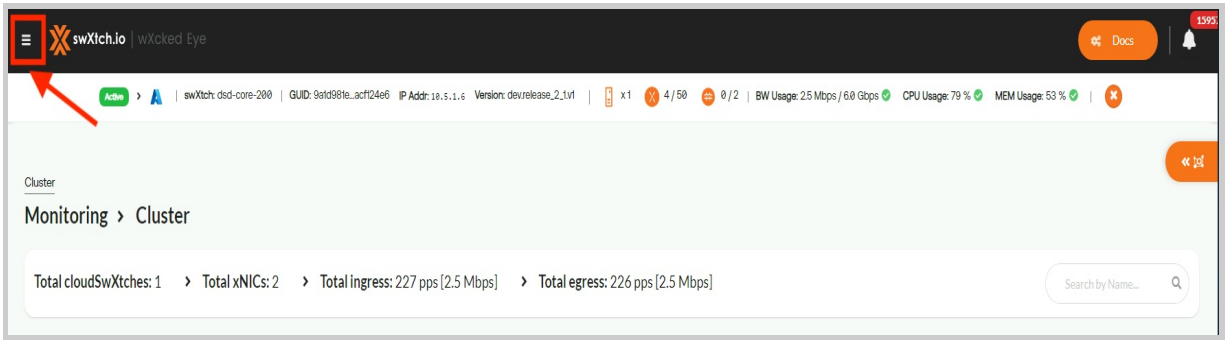
wXcked Eye Topology Graph

WHAT TO EXPECT

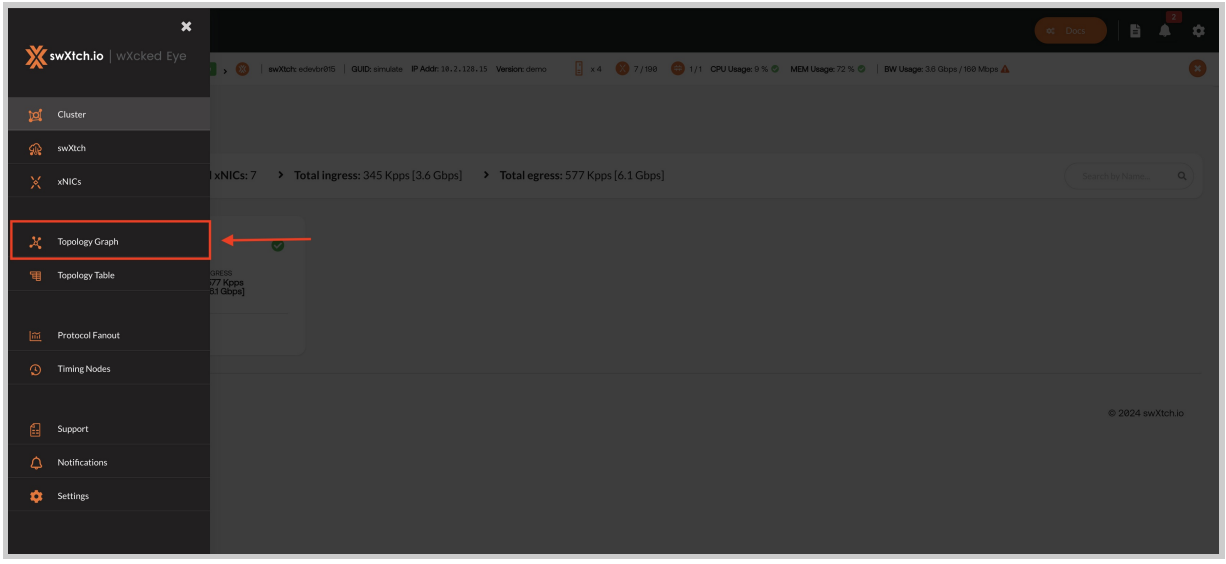
In this article, users will learn how to use the wXcked Eye Topology Graph and how to reformat it for their needs.

Locating the wXcked Eye Topology

To navigate to the wXcked Eye Topology Graph page, users will need to click on the menu (☰) option at the top right hand corner by the swXtch.io logo.

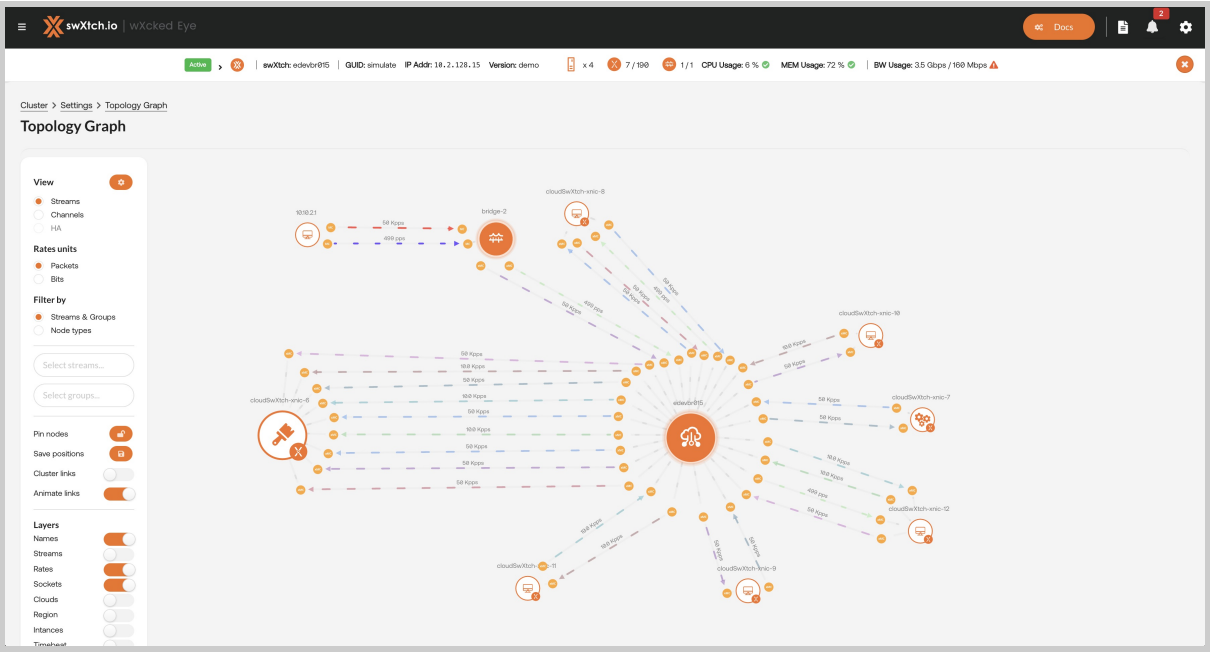


From there, select **Topology Graph**.



Using the wXcked Eye Topology

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



The **wXcked Eye Topology Graph page** displays a network graph, providing a high level view of the cloudSwXtch environment. On the top panel, users will find the name of the cloudSwXtch with a list of relevant network information. The center of the graph will display the cloudSwXtch you are currently on with user-assigned colored lines indicating traffic flowing either to or from it. Next to each line, users will be able to see the flow's transmission rates (either in pps or bps). The endpoints can be either xNICs, cloudSwXtch Bridges or other cloudSwXtch instances.

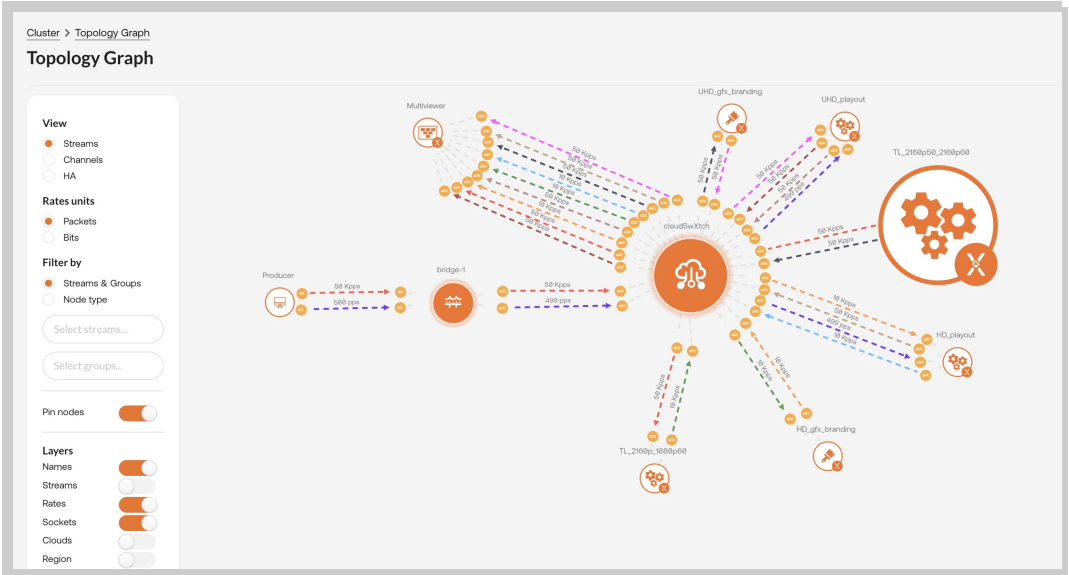
Reformatting the Topology Graph

It is very simple to alter the Topology Graph for a user's desired configuration. In addition to being able to physically drag and rearrange the icons in the graph, users can zoom in and out, refresh the page, and toggle labels on and off. In the left hand side, users will find several options to alter the appearance of their topology graph.

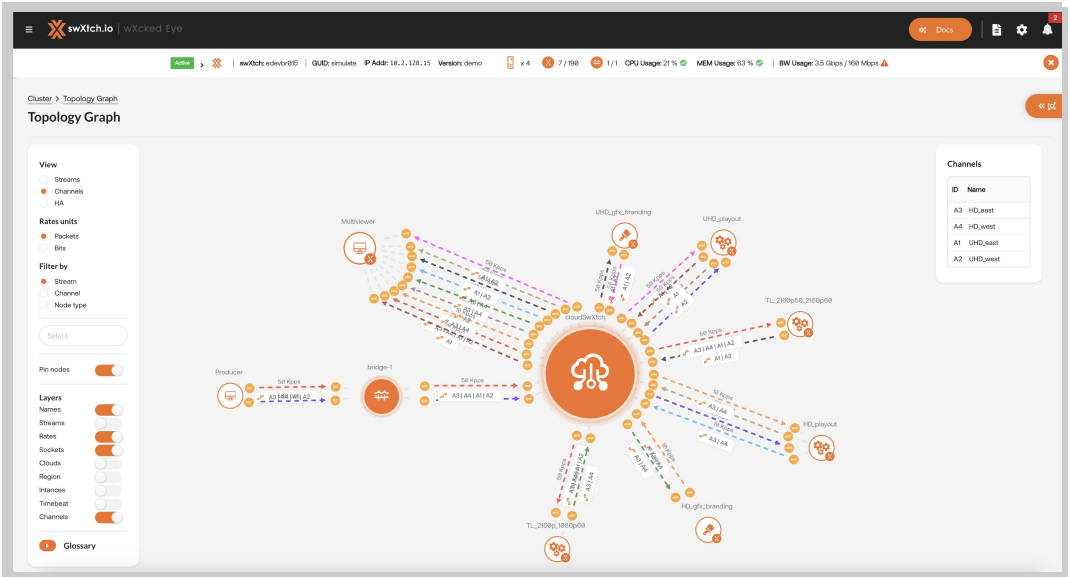
View

There are three views available for users: **Streams**, **Channels** and **HA**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



The Streams view provides users with a high level view of all the streams associated with the cloudSwXtch.



The Channels view highlights streams associated with a specific channel distinction. Each relevant line will have a Channel ID attached to it. A Channels key will also appear.

Finally, HA highlights the multiple paths in an HA configuration.

Streams, Channels and HA links can be formatted in the wXcked Eye Settings page.

Rates units

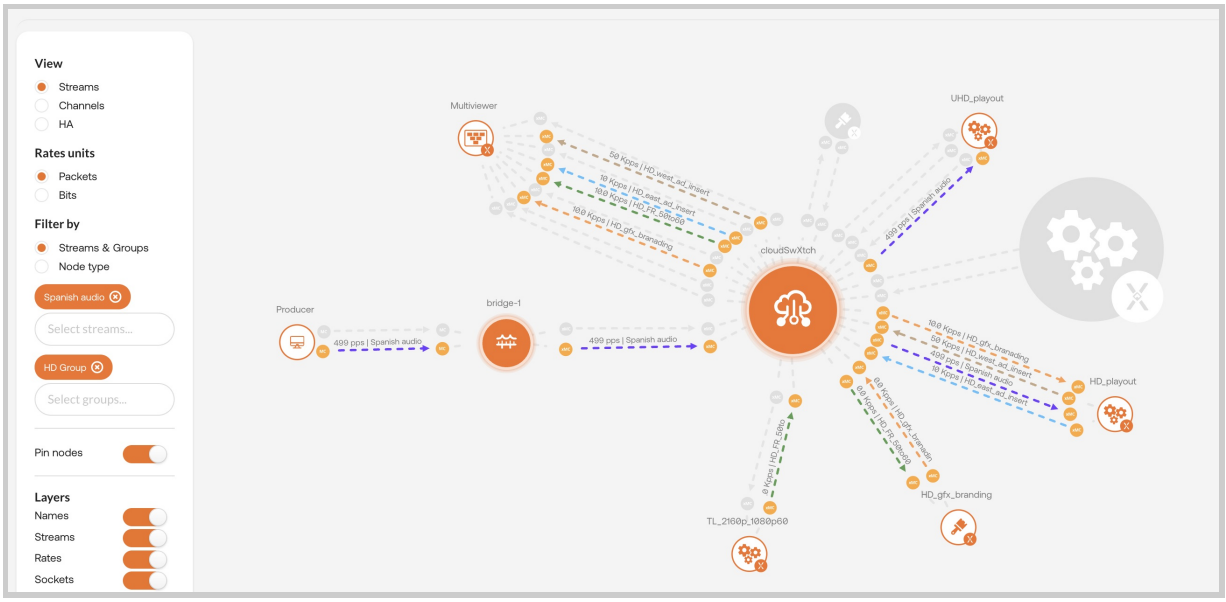
For Rates, users can select between Packets and Bits for their unit.

Filter by

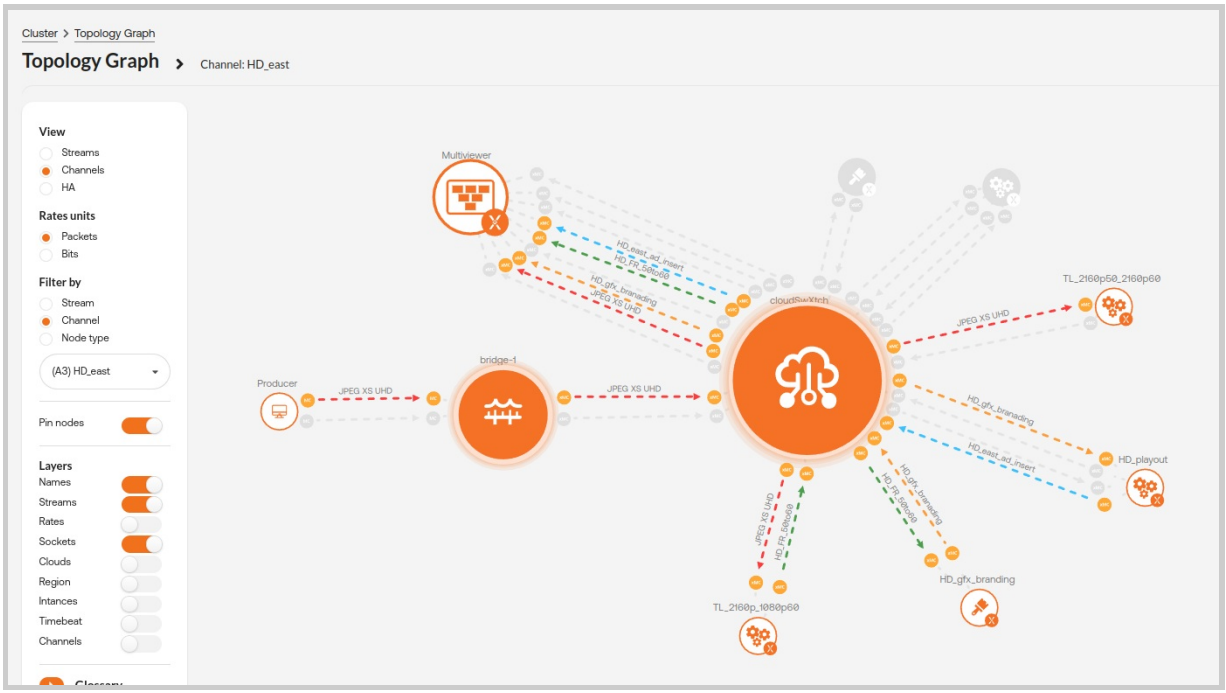
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Users can also filter by Streams & Groups and Node Types. Choosing either category will alter the subsequent “Select” prompts. Here, users can specify the streams, groups, channels, or node types they wish to see in the graph.

In this example, the user has "Spanish audio" and "HD Group" filtered on. The links related to that stream and group are highlighted in the Topology Graph.



When the View is set to Channels, an additional Filter By Channel option will activate, allowing users to highlight streams related to a specific channel. In the example below, the user filtered on the channel "(A3) HD_east." All related streams are highlighted in the graph.



Additional Options

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

The Topology Graph also provides users with some additional options to alter how the data is visualized.

- **Pin nodes**: When activated, this will lock all components in place.
- **Save positions**: Save positions will save the position of all components so that the next time wXcked Eye opens, they will remain locked in place.
- **Cluster links:** When activated, links between nodes will cluster depending on the number set in the topology settings.
- **Animate links**: When activated, the links illustrating data flow will animate the direction.

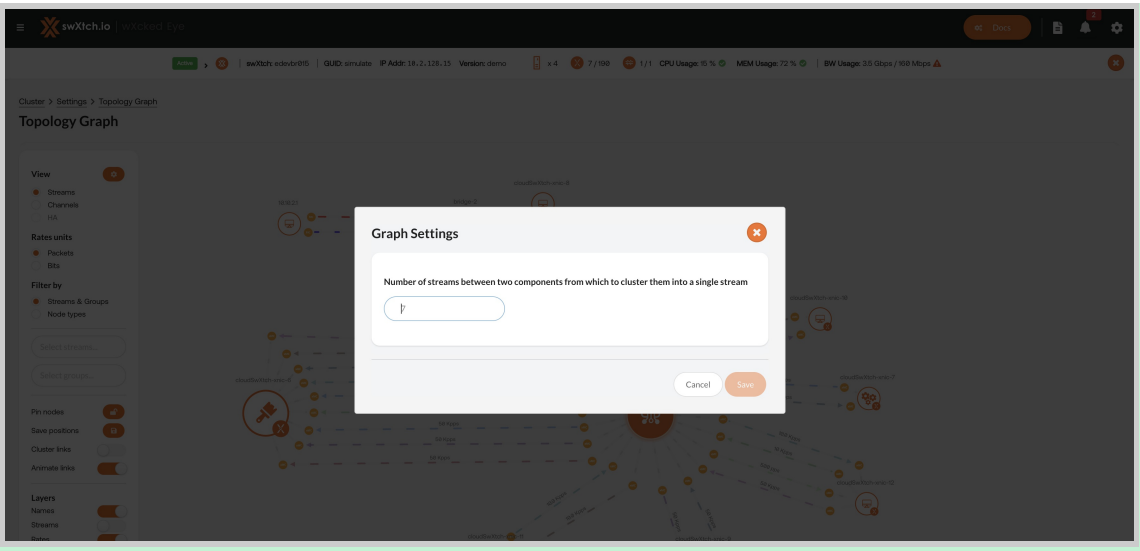
Layers in the network graph can be toggled on and off depending on the information a user wishes to display:

- **Names**
- **Streams**
- **Rates**
- **Sockets**
- **Clouds**
- **Regions**
- **Instances**
- **Timebeat**

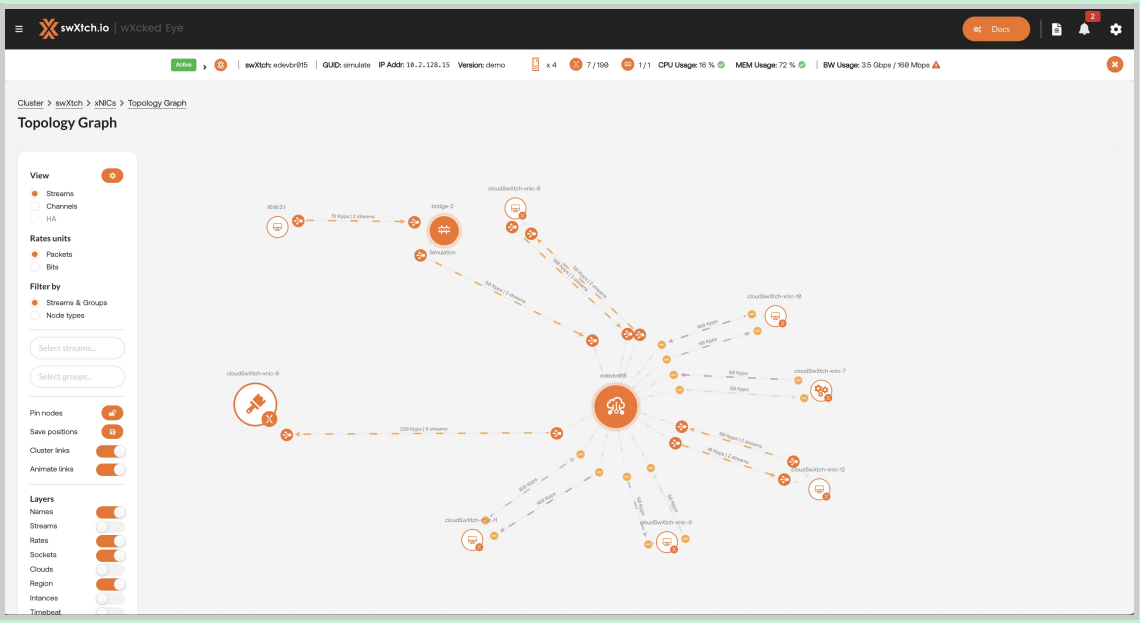
Setting Up Cluster Links

Cluster links groups together streams of a similar direction into a single link. Users can set a specific number to trigger this clustering feature. To do this, select the Gear symbol next to the View in the filter panel on the left-hand side of the Topology Graph. There, specify the number of links that will trigger a cluster.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



In the example below, the user set the number of streams to 7, causing the number of links to drastically decrease between each component.

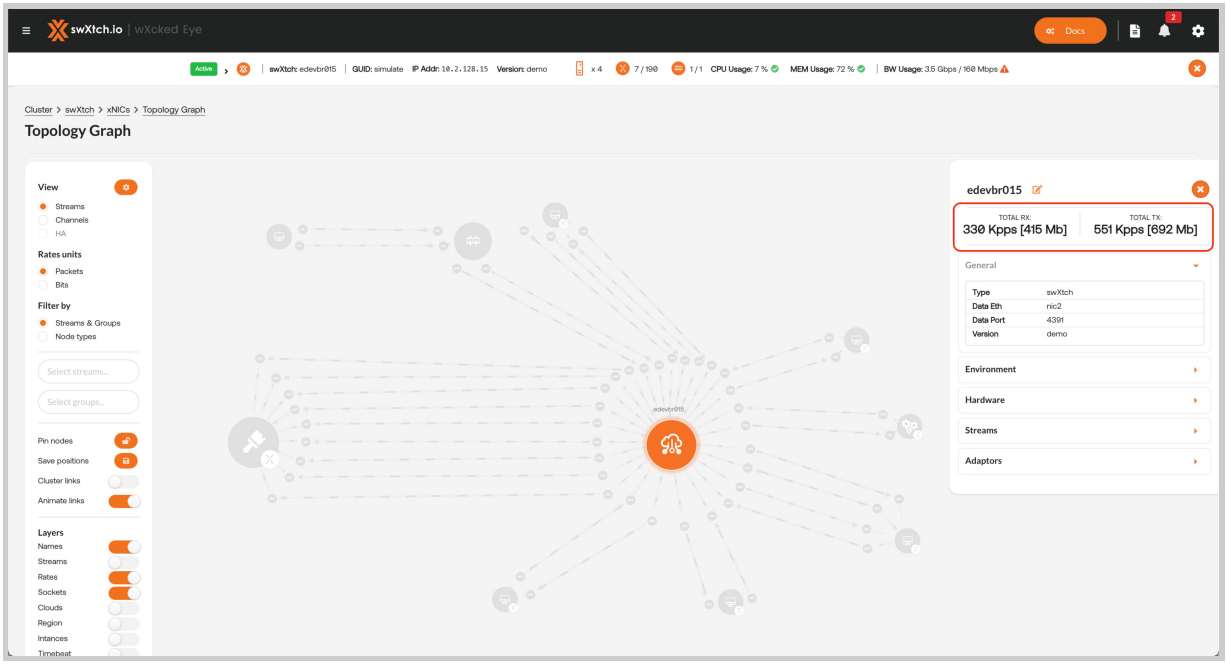


Component Information Sidebar

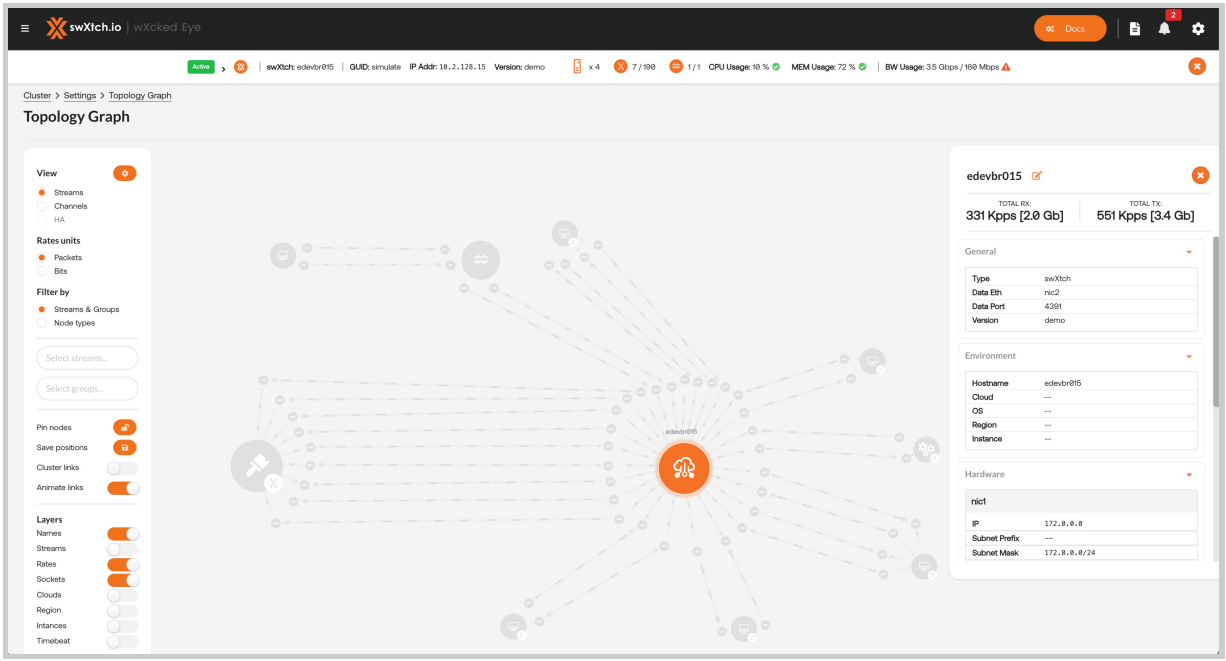
When selected, each component in the Topology Graph will open a information sidebar. This will display information regarding the component, organized into eight categories: Statistics, General, Environment, Hardware, Streams, Adaptors, High Availability, and Timebeat.

At the top of the component sidebar, users will see a high level view of Total Rx and Total Tx statistics from the perspective of the selected component. In the example below, the user has the cloudSwXtch in the center selected and can see the data being received and transmitted.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



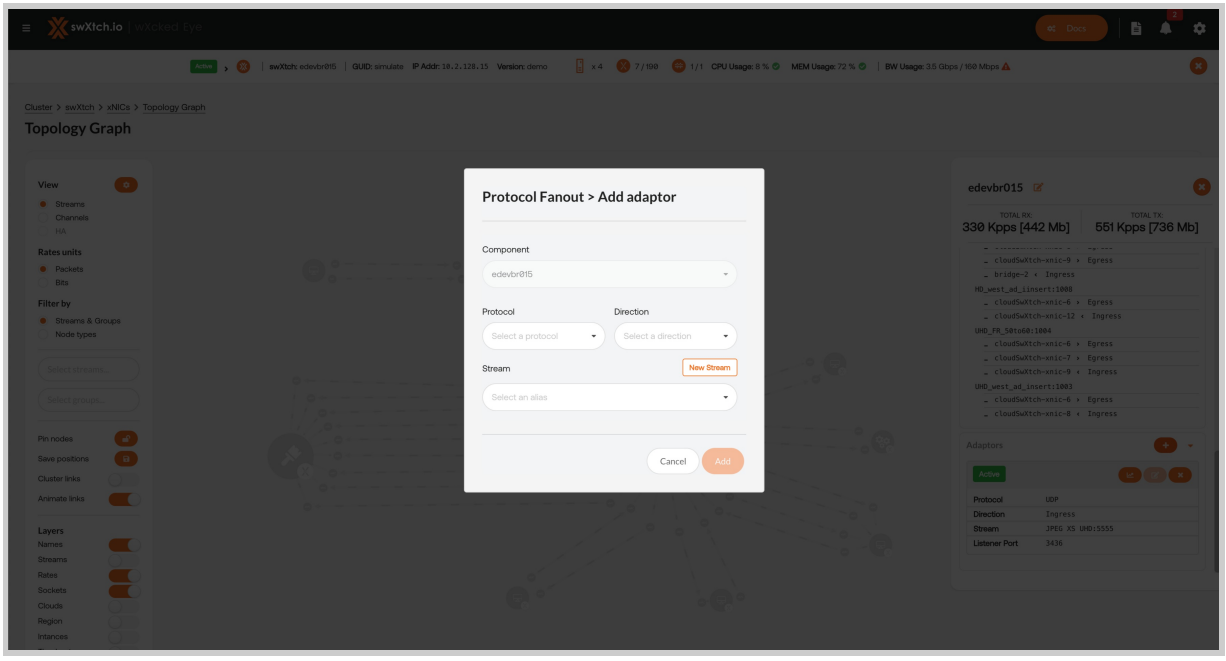
The **General** section details the type of component (cloudSwXtch, xNIC, cloudSwXtch Bridge, etc.), the data eth, data port and the version. The **Environment** provides information regarding the cloud provider, including the host name, cloud, OS, Region, and Instance type. The **Hardware** section goes deeper into the information regarding the virtual machine, specifically the NICs.



The **Streams** section provide users with a detailed list of all the streams going through the component and the direction of data flow.

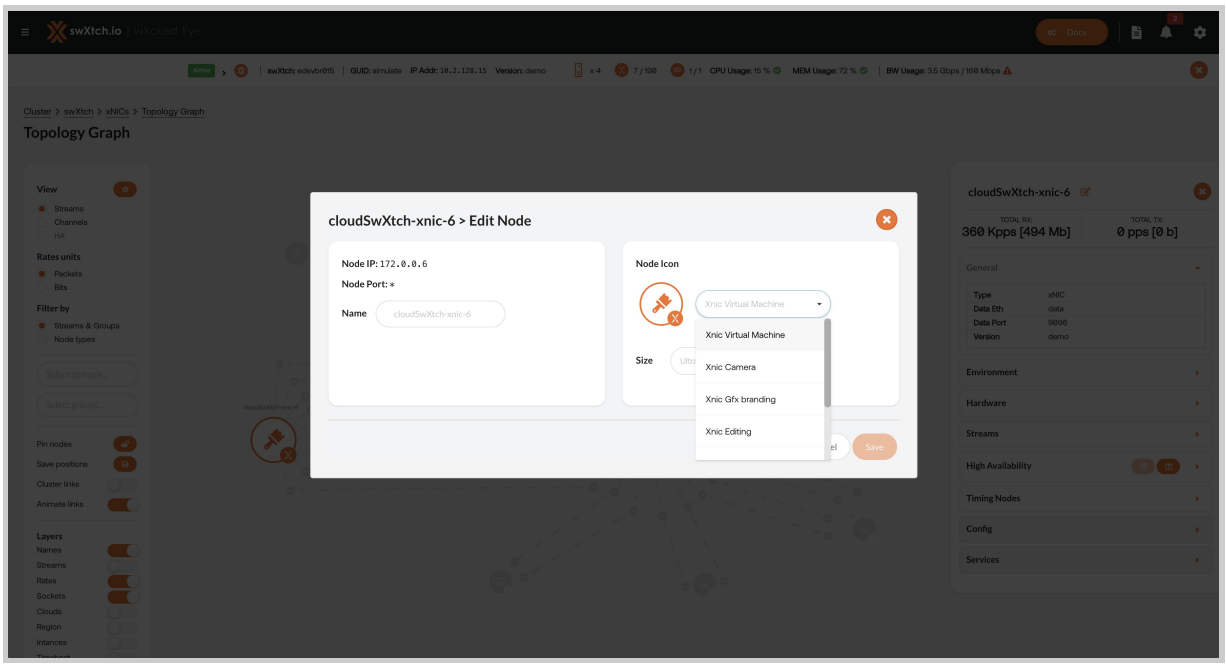
Under **Adaptors**, users can view adaptor details, add, edit, remove, and link to adaptor stats. Adding an adaptor will open a window directly in the Topology Graph.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



The **High Availability** and **Timebeat** section will populate with information if either are configured in the cloudSwXtch network.

Editing a Node's Appearance



Users can change the appearance of each node or component in the Topology Graph. By selecting the edit pad next to the component's name in the Information Sidebar, a user can edit the size, the name and the icon for each node.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

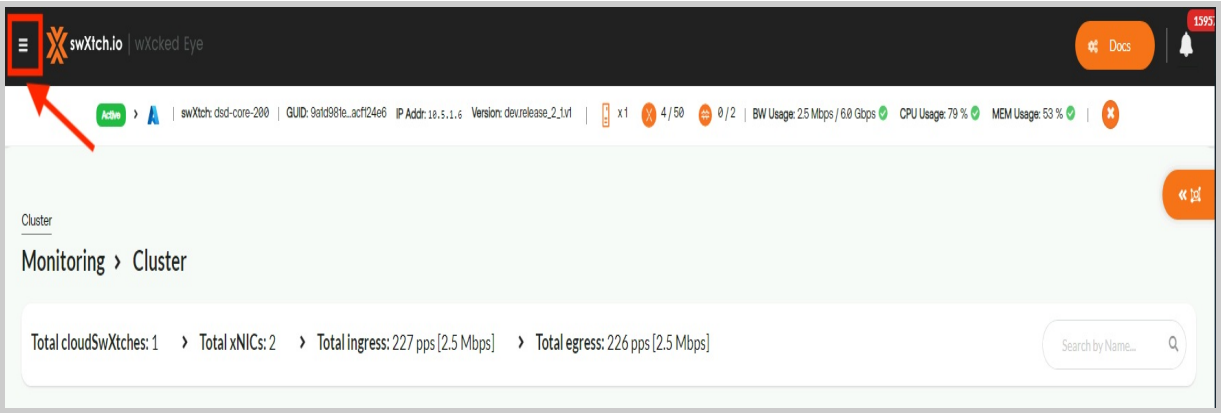
wXcked Eye Topology Table

WHAT TO EXPECT

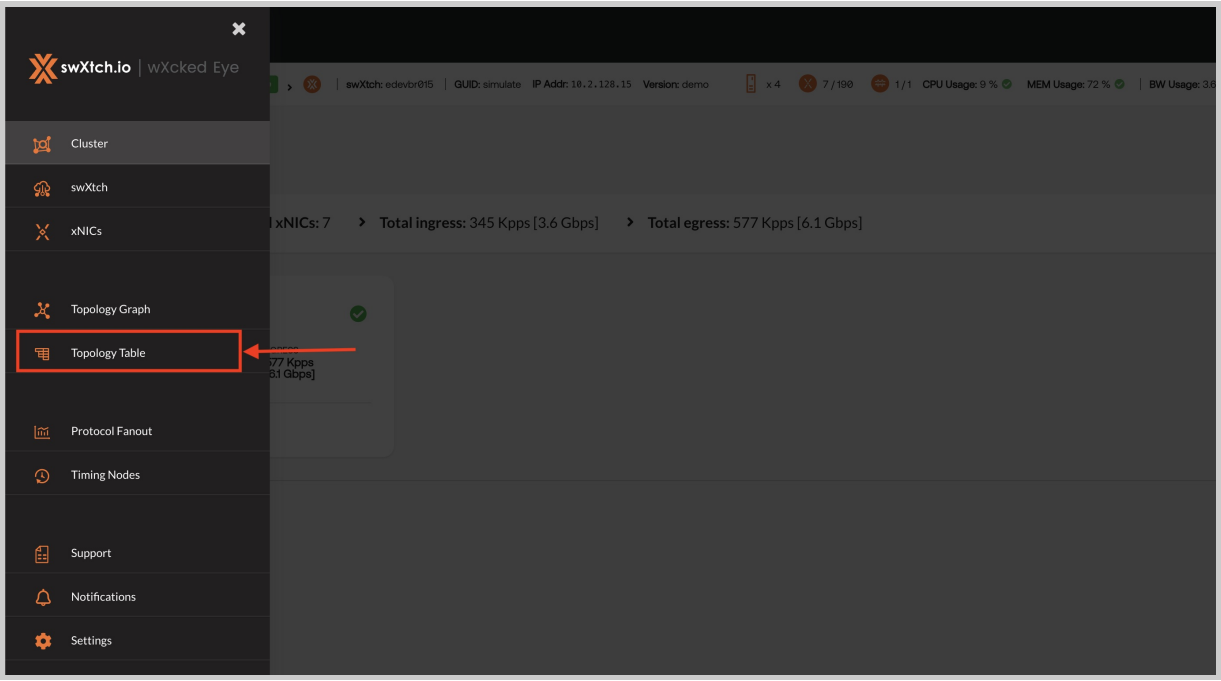
In this section, users will learn how to navigate the wXcked Eye Topology Table. Similar to the Topology Graph, the table organizes statistics by either Streams or Channels in a tree view.

Locating the wXcked Eye Topology Table

To navigate to the wXcked Eye Topology Table page, users will need to click on the menu (☰) option at the top right hand corner by the swXtch.io logo.



For there, select **Topology Table**.



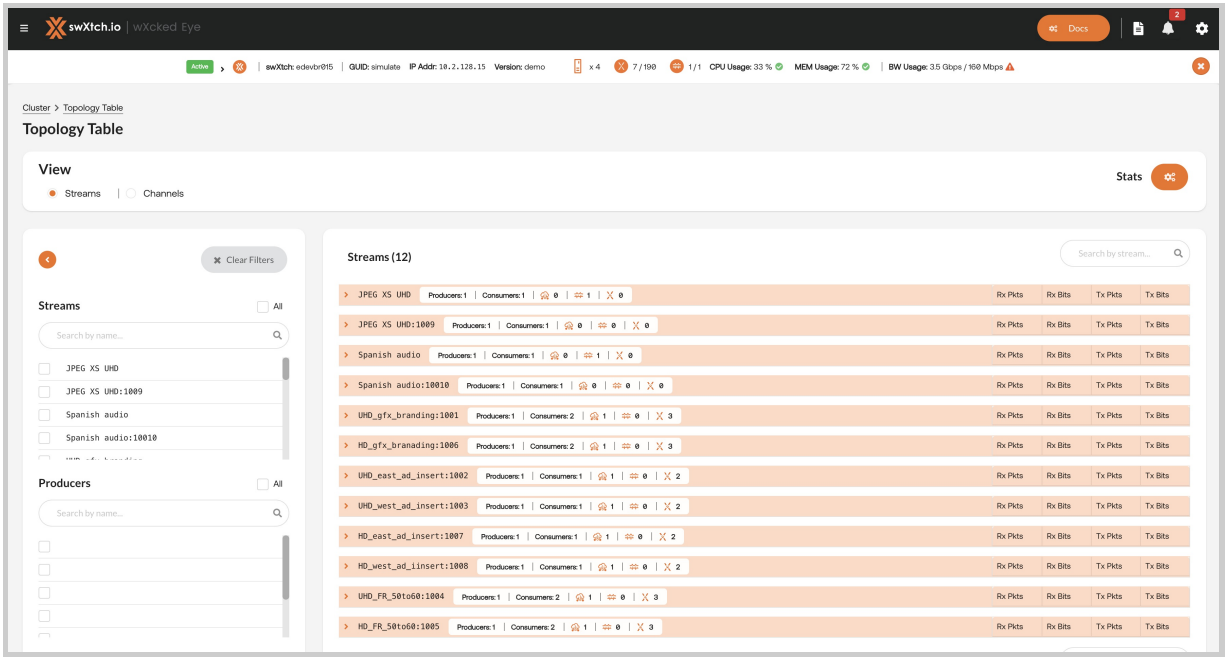
Using the wXcked Eye Topology Table

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

The wXcked Eye Topology Table provides users with an alternative way of viewing how data flows in and out of their cloudSwXtch. Instead of a Topology view with nodes and steam links, the table organizes streams and channels in tree view where they can easily see the origin of the data flow and how it moves through different components. This gives users a more focused look into a single stream or channel.

Streams

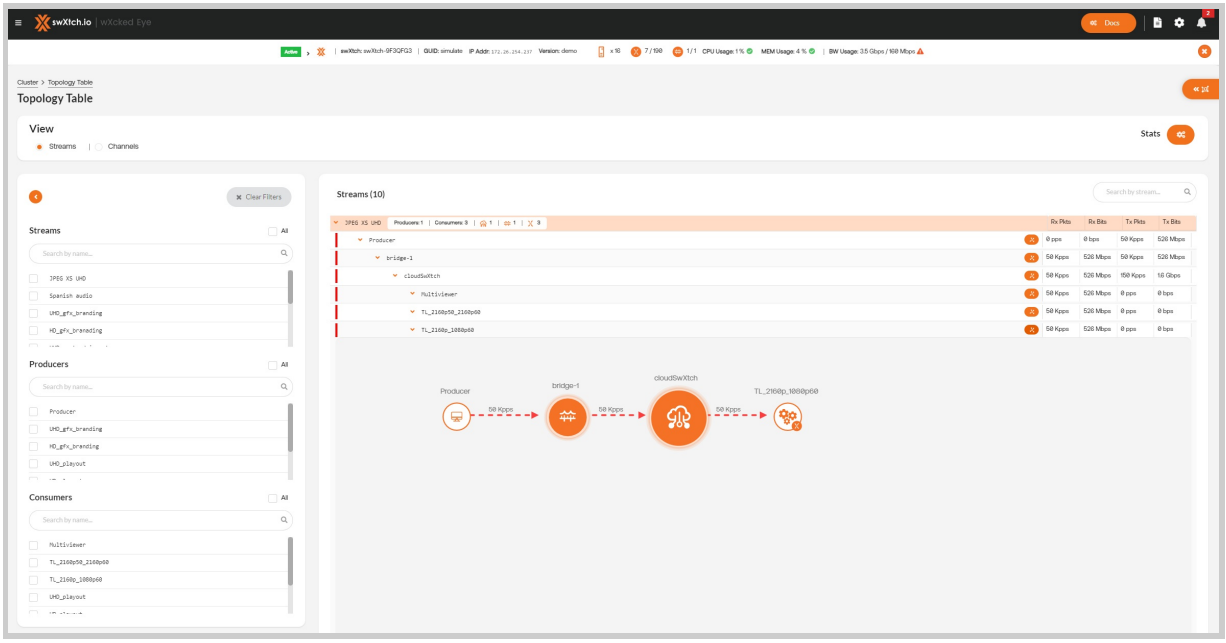
In the example below, the user is viewing a list of streams. Similar to the Topology Graph, a user can filter specific streams, consumers, and producers. The list will adapt to those selections. This can be helpful when dealing with a large amount of line items.



Each item in the stream list can be expanded to reveal a tree level view of the data flowing through the various components. The JPEG XS UHD stream is expanded to show how data flows through the various cloudSwXtch components. Starting from the producer, the stream moves through the cloudSwXtch Bridge up into the cloudSwXtch, where it feeds the stream into a multiviewer and Tachyon Live endpoints.

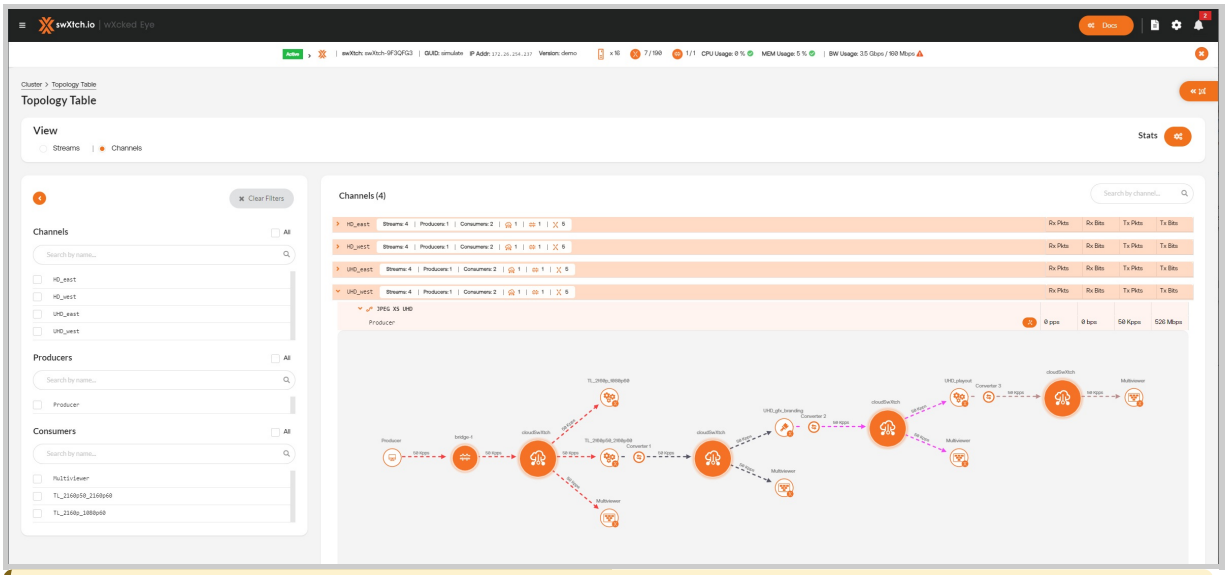
Selecting the Topology button next to the component name will reveal a visualization similar to the Topology graph up until that point.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



Channels

The Channels view organizes channels into a similar list for the Topology table. However, one big difference is that the number of streams associated with a single channel is also displayed. The table will list those streams in a tree view, displaying how they progress through the various cloudSwXtch components. In the example below, the UHD_west channel has been expanded, revealing the JPEG XS UHD stream with a topology graph view of data flow.



Multiple cloudSwXtch Nodes

In the Channels view example above, the cloudSwXtch node repeats. This does not mean that multiple cloudSwXtches are being used but rather the stream is flowing back and forth to the same cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

wXcked Eye Protocol Fanout Stats

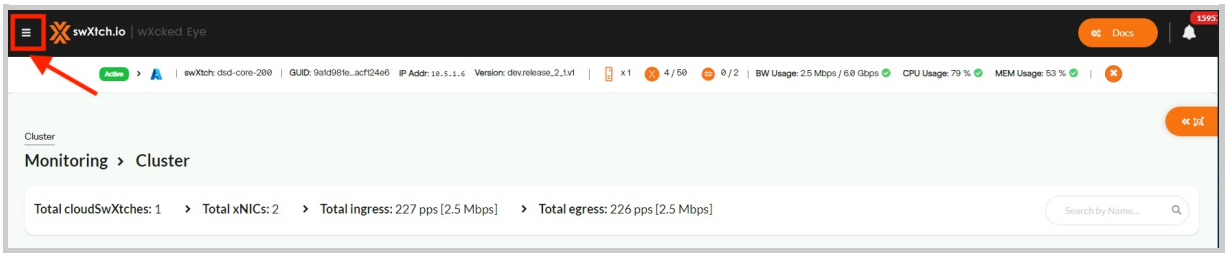
WHAT TO EXPECT

The **Protocol Fanout Stats** page allows users to see metrics for non-Multicast and non-Broadcast data flows. This includes protocols, such as **SRT Caller, SRT Listener, RIST Caller** and **RIST Listener.**

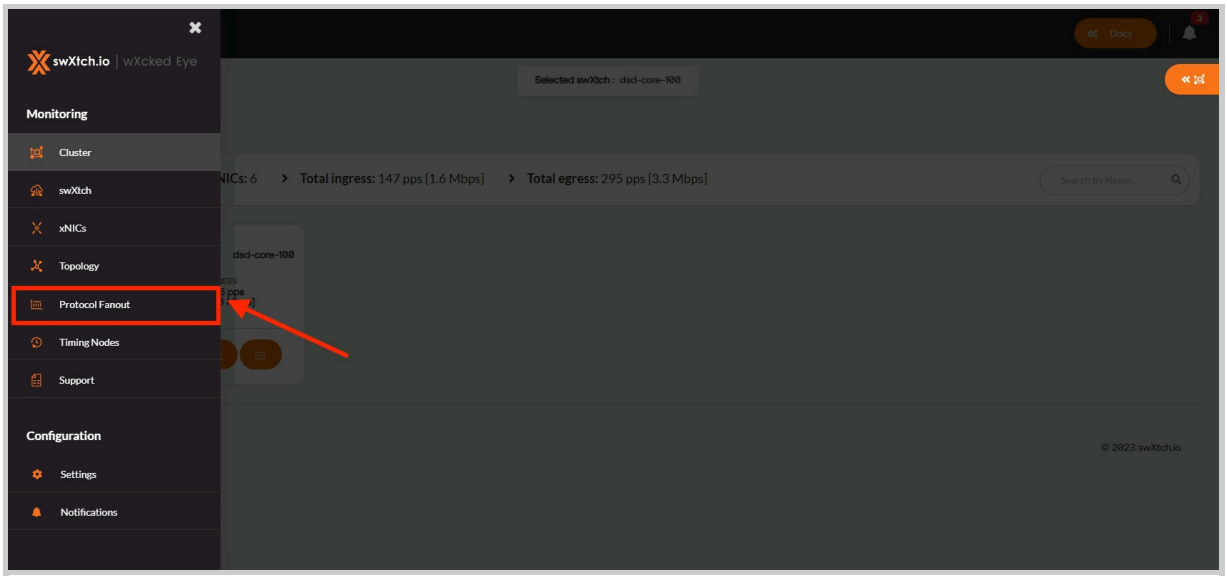
In this section, users will learn how to navigate between different protocols and their adaptors in order to better visualize their packets' movements. **Please note**: This page only shows the stats for protocol fanout configurations. To learn how to configure your cloudSwXtch for Protocol Fanout and Conversion in the wXcked Eye UI, please read **this article**.

Locating the Protocol Fanout Stats Page

To navigate to the **Protocol Fanout Stats** page, users will need to click on the menu option at the top right corner by the swXtch.io logo.



From there, select **Protocol Fanout** under **Monitoring**.

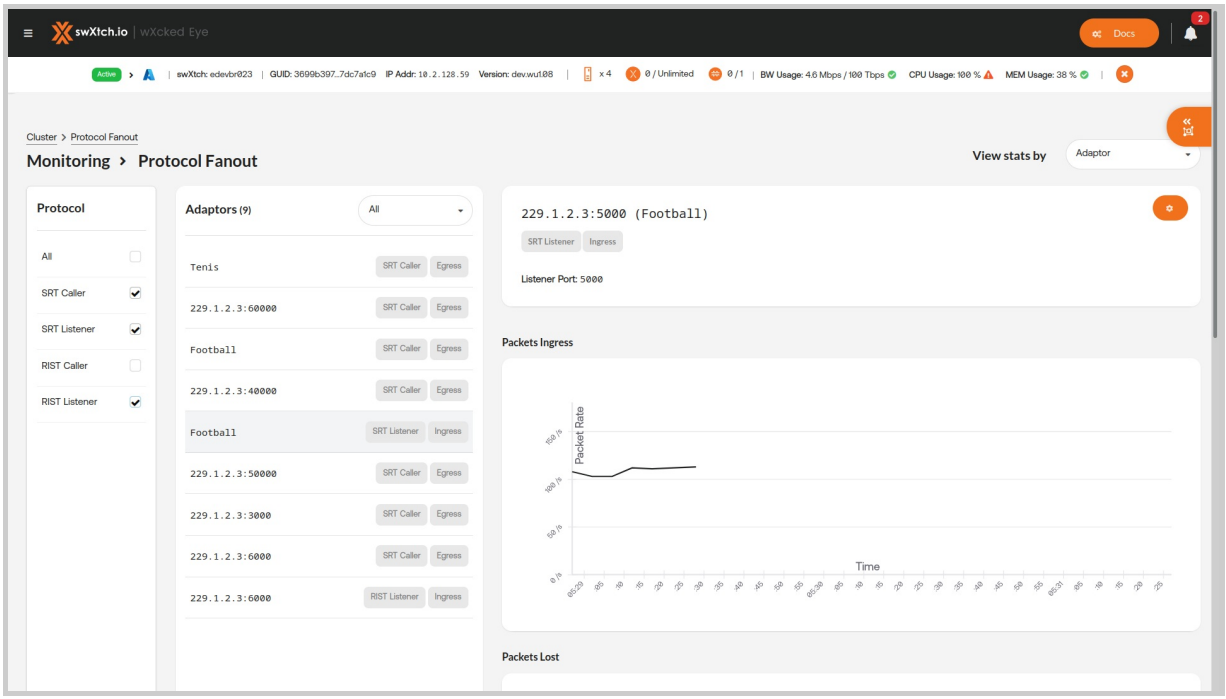


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Navigating the Protocol Fanout Stats page

Statistics displayed through the wXcked Eye UI focus primarily on multicast and broadcast data flow. The **Protocol Fanout Stats** page provides users with a dedicated area to see data flow for alternative protocols like SRT and RIST.

Adaptor View



Adaptor View

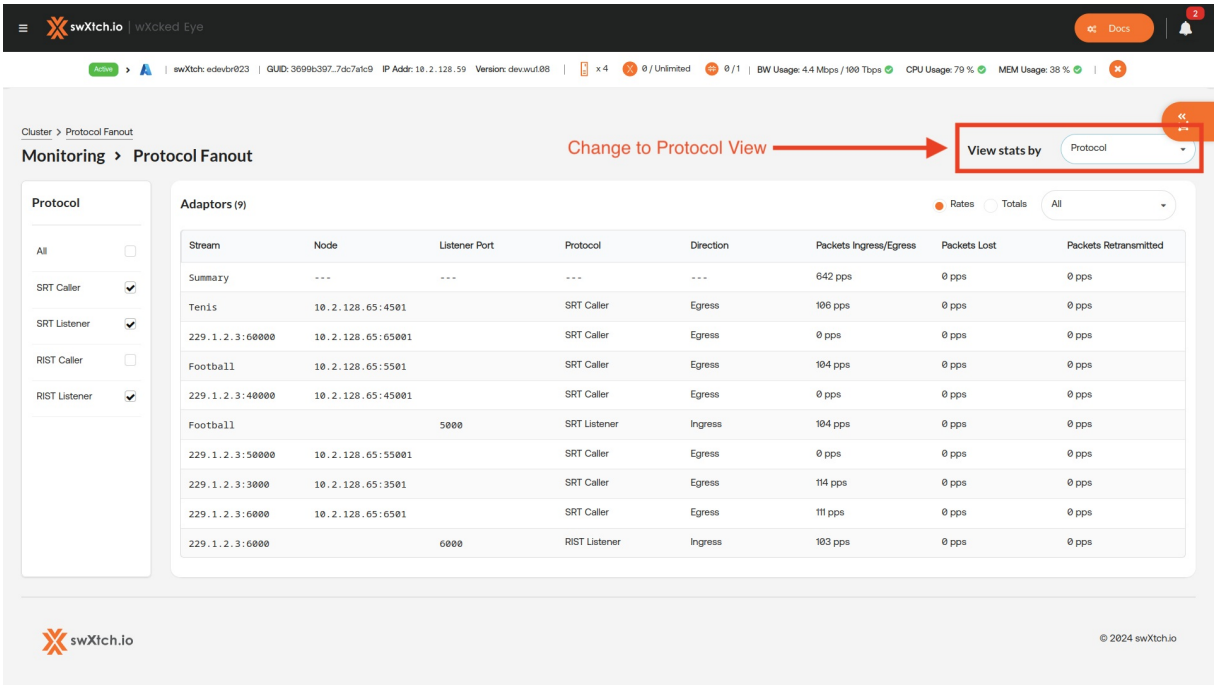
At start-up, the page will be in **Adaptors** view and display 3 graphs:

1. Packets Egress
2. Packets Lost
3. Packets Retransmitted

A user can select the **Protocols** they want listed in the left hand side to filter Adaptors or return to **"All"** to see them all listed. At the top of the Adaptors panel, users can filter the list further by direction -- either Ingress or Egress.

Protocol View

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



A user can select to **View Stats by Protocol** by clicking the dropdown menu in the upper right hand corner. This will provide users with a table view, listing adaptors by protocol. This allows for closer comparison between adaptors. The table can be configured to display in both **Rates (pps)** or **Total Packets**.

Similar to the Adaptors view, users can group adaptors by protocol. The Protocol panel on the left hand side enables users to filter our certain protocols so that they list in the center table. In addition, users can also group protocols by direction -- either Ingress or Egress.

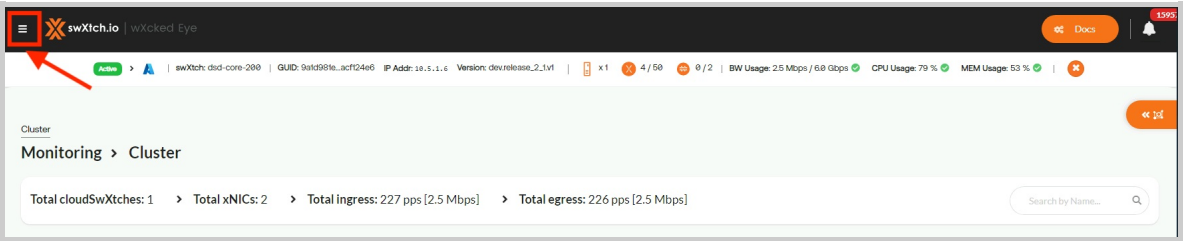
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

wXcked Eye Timing Nodes

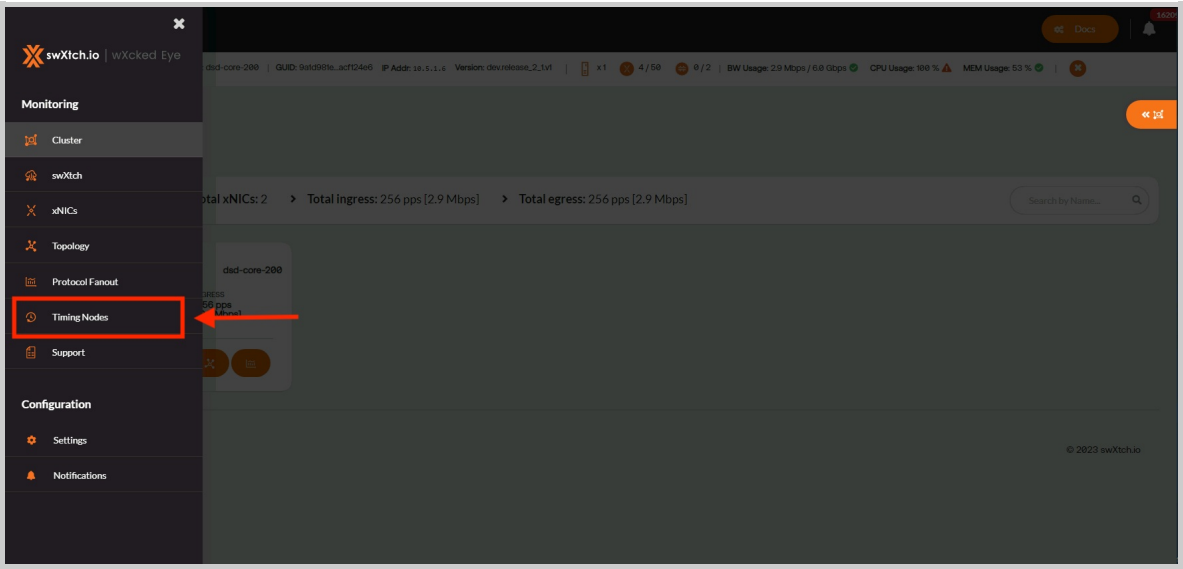
Finding the Timing Nodes page

To find the **Timing Nodes** page in the wXcked Eye UI:

1. **Click** the menu icon (☰) next to the swXtch.io logo.

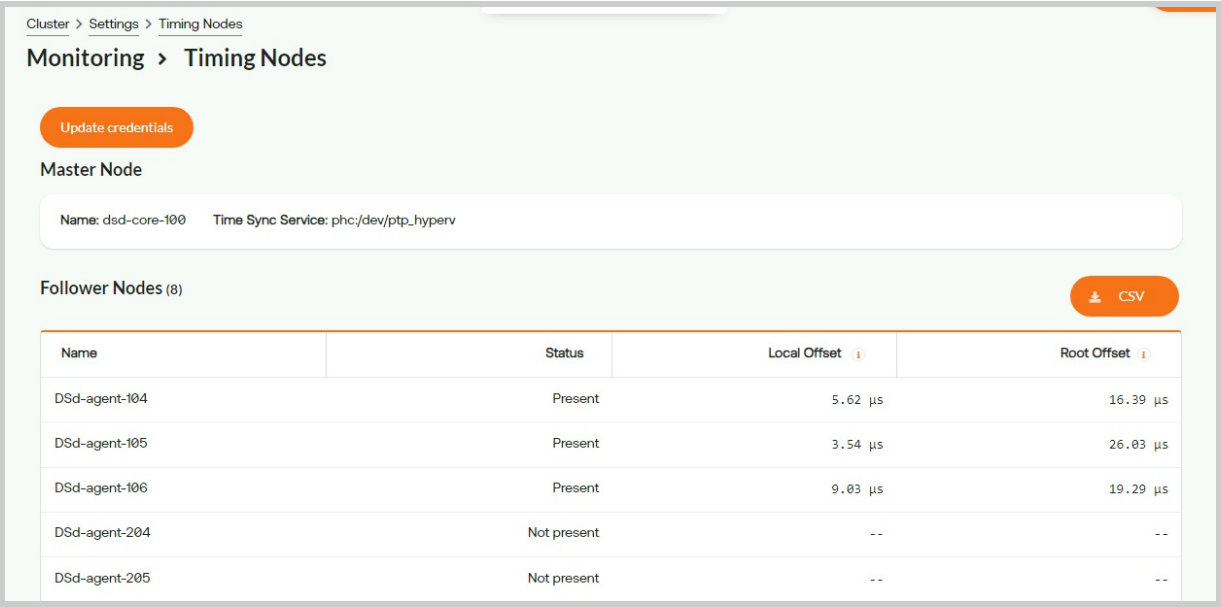


2. **Select Timing Nodes** under Monitoring.



Understanding the Timing Nodes page

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



The Timing Nodes page displays information regarding clock sync configuration for the cloudSwXtch. This page in wXcked Eye will only populate with information if the user has the PTP feature enabled.

In the example above, the cloudSwXtch (DSd-core-100) is acting as the Master Node.

- **Master Node**- The Master Node is what the PTP configuration sets as the most reliable time source. This will send the true time it receives from the source clock to the Follower Nodes.
    - **Name** - The name of the cloudSwXtch
    - **Time Sync Service** - The source clock

- **Follower Nodes**- The Follower Nodes lists the agents/VMs that subscribe to the Master Node for accurate timing.
    - **Name** - The name of the endpoints
    - **Status** - The status of the endpoints, noting if the node is active in the PTP configuration
    - **Local Offset** - The local offset denotes the offset in time from the cloudSwXtch to the xNIC.
    - **Root Offset** - The root offset denotes the offset in time from the GrandMaster clock to the cloudSwXtch and its follower nodes (xNIC). Note how the root offset is larger than the local offset. This is normal behavior since the distance between the follower node and the Grandmaster clock is greater than the offset between a cloudSwXtch and xNIC.

Timing Nodes Stabilization

After upgrading or rebooting your cloudSwXtch system, you may notice that the local and root offset values are much larger than they actually are. It can take up to 30 minutes for the values to stabilize and return back to normal levels.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Exporting your Timing Nodes

You can export your timing nodes by hitting the **CSV button** next to **Follower Nodes**.

Formatting CSV Timing Nodes file in Excel

To prevent incorrect formatting in your CSV Timing Nodes file in Excel, complete the following steps:

1. **Make sure** your Timing Nodes CSV file is already downloaded from wXcked Eye.
2. **Select** "Data" from the top ribbon of a new Excel spreadsheet.
3. **Click** "Get Data (Power Query)."
4. **Select** "Text/CSV" from the "Choose data source" options.
5. **Browse** for your file and click "Get Data."
6. **Click** "Next."
7. **Select** "Unicode (UTF-8) from the File Origin dropdown menu. This ensure your data displays as it was intended.
8. **Click** "Load."

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

wXcked Eye Support Page

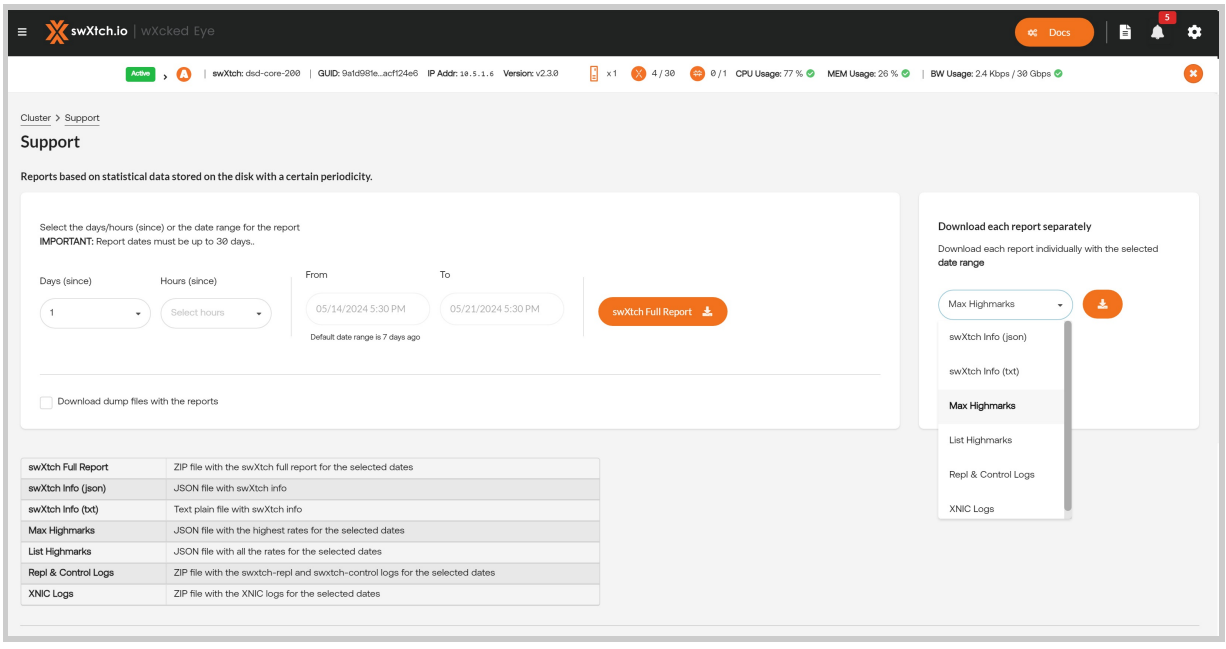
WHAT TO EXPECT

The wXcked Eye Support Page allows users to export a report detailing the statistical data stored within the cloudSwXtch over a set period of time. This report includes JSON files containing cloudSwXtch information, Max Highmarks, List Highmarks, Repl and Control logs, and xNIC Logs -- all in a compressed file. This report should be provided to swXtch.io Support when troubleshooting an issue.

In this section, we will walkthrough exporting both a full report and also individual files. Alternatively, you can use the swx support command in the cloudSwXtch VM to export a report. For more information, see How to View cloudSwXtch Logs for Troubleshooting.

Navigating the wXcked Eye Support Page

The wXcked Eye Support page can be located under Monitoring in the wXcked Eye navigation menu. The page has two functionalities: **exporting a full report** or **selecting individual JSON files to download separately**. When troubleshooting, it is **recommended** for users to send the complete report so that swXtch.io Support can fully understand the situation.



To do this, simply set a **start** and **end time** for the report and select the download button, **SwXtch full report.** User should set the duration to **at least 24 hours of time**, spanning from a little before the issue began to up until now. The output file will be named **swxtch-report-date_from.tar.gz**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

In the event that a user only wants a specific section of the report, they can use the dropdown menu after **Download each report separately** and download their desired JSON file. The wXcked Eye UI will then export using the time period set.

Contacting swXtch.io Support

For all troubleshooting requests, email the compressed file to support@swxtch.io for further instructions.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Configure cloudSwXtch with wXcked Eye

WHAT TO EXPECT

wXcked Eye allows users to configure their cloudSwXtch directly from a graphical user interface (GUI).

To learn how to use wXcked Eye to monitor your cloudSwXtch, please see the "Monitor cloudSwXtch with wXcked Eye" article.

In this article, users will learn how to navigate the "Settings" option in the wXcked Eye UI and to configure their cloudSwXtch for mesh, high availability and protocol fanout. To learn how to access the wXcked Eye UI, please review the following article.

Accessing wXcked Eye

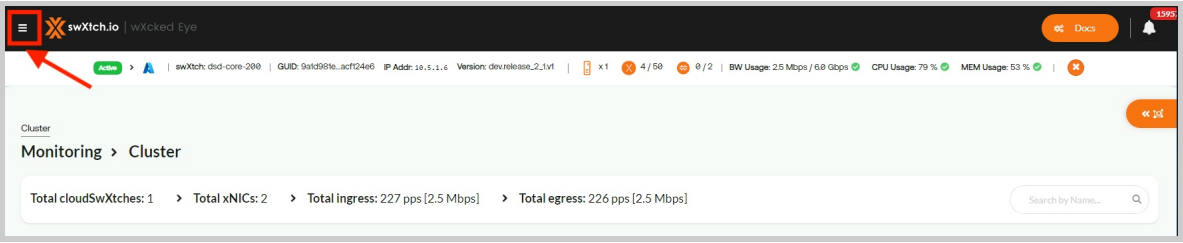
To access the wXcked Eye UI, users will need to enter the following URL into a web browser of a VM in their cloudswXtch environment. They should use the IP address of their cloudSwXtch to prefix the URL.

Custom	Copy
`<swxtch-ip-address>/wxckedeye/`	

Finding the Settings page

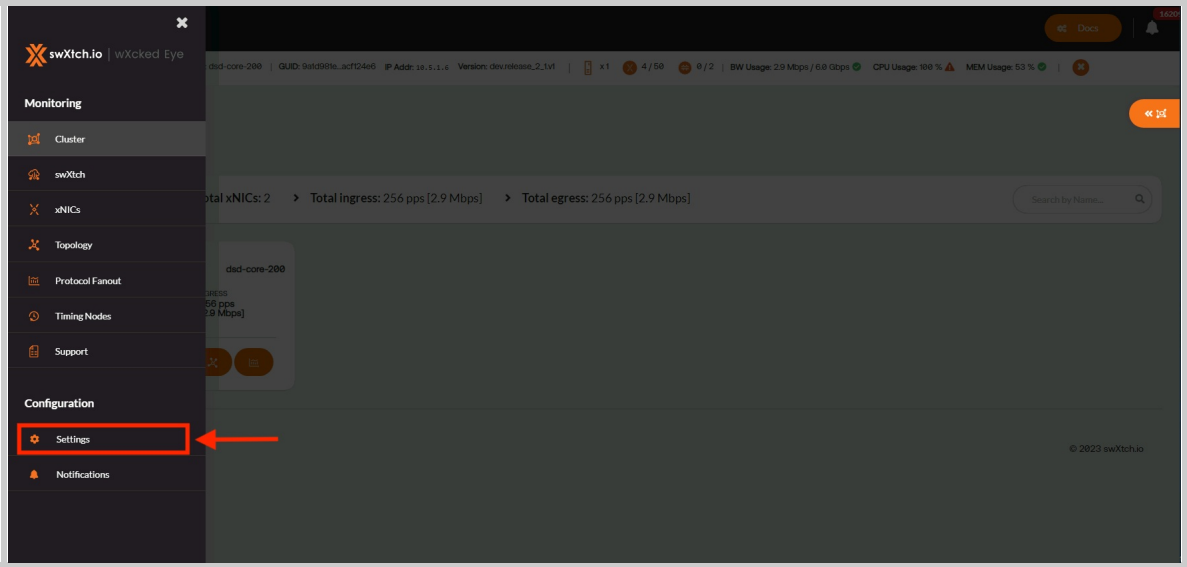
To find the Settings page in the wXcked Eye UI:

1. **Click** the menu icon (☰) next to the swXtch.io logo.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Select **Settings** under **Configuration**.



3. You should now be on the **Settings** page.

Navigating Settings

The Settings page is organized into five tabs with varying functionalities:

- General
- Mesh
- High Availability
- Protocol Conversion and Fanout
- Aliases

In this section, we will discuss each tab and how it offers the user additional control over their cloudSwXtch network.

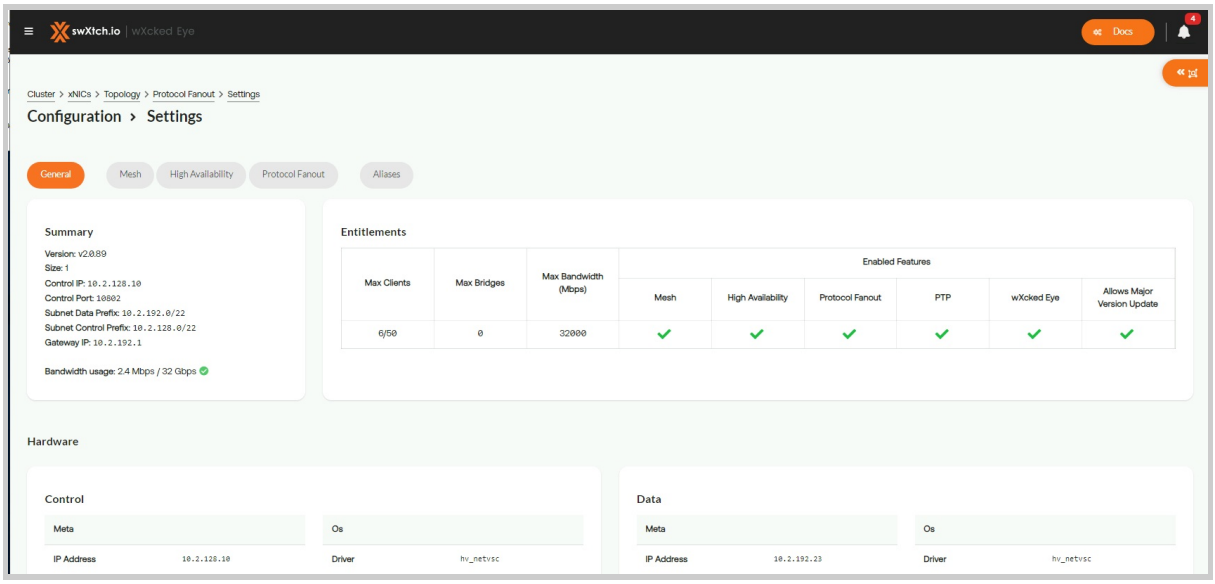
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

General

How to Navigate to the General tab

To learn how to navigate to Settings from the wXcked Eye main page, please review the **Configure cloudSwXtch with wXcked Eye** article.

Navigating the General Tab



The General tab is organized into four sections:

- cloudSwXtch Summary
- Entitlements
- Hardware
- Actions

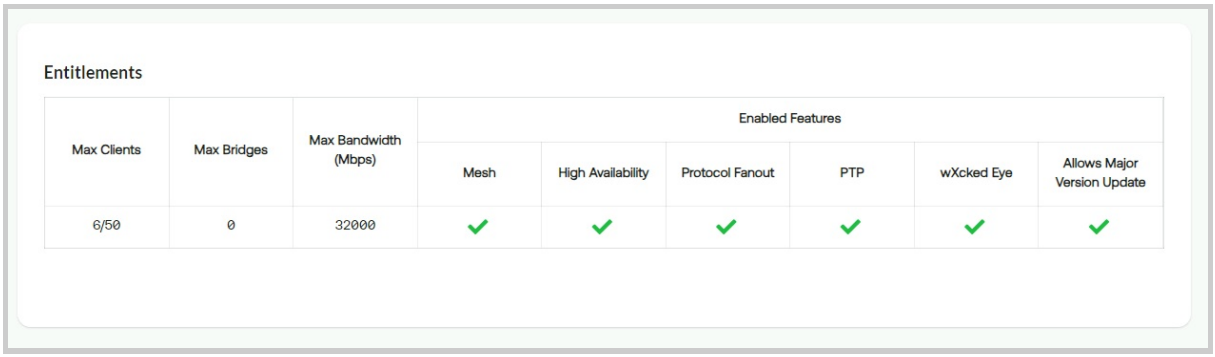
cloudSwXtch Summary

The Summary panel details basic information regarding the cloudSwXtch, specifically on the data and control subnets configured during installation. Similar to the cloudSwXtch view, the Summary panel also displays the amount of bandwidth currently in use.

Entitlements

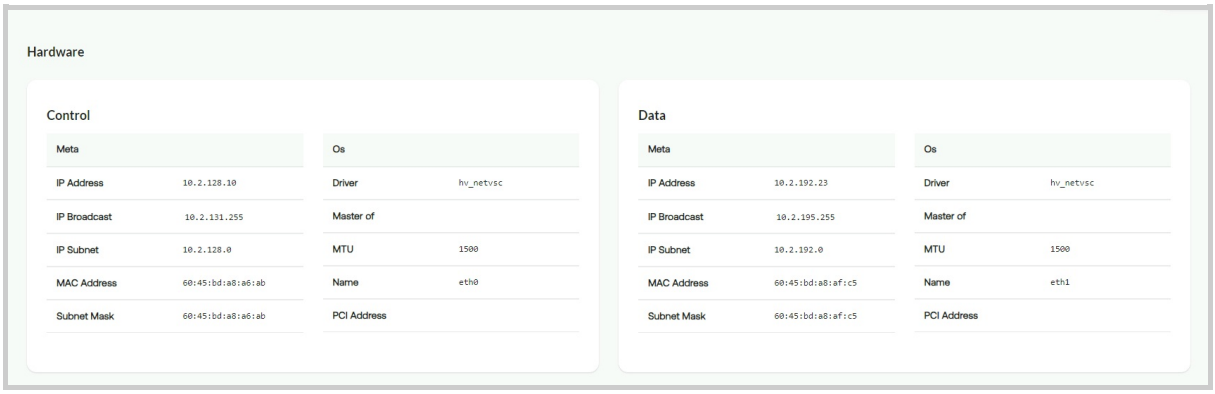
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

The **General** tab is designed to give users a detailed look into the entitlements associated with their network. In the example, the user has a license that enables mesh, high availability, protocol fanout, and PTP with a max of 50 clients, 0 cloudSwXtch bridges, and 32 Gbps max bandwidth.



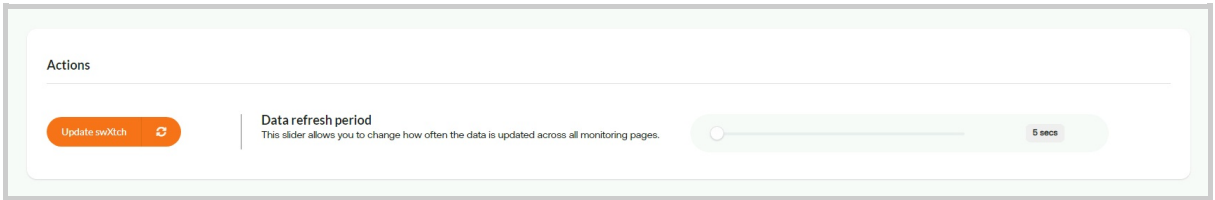
Hardware

The Hardware section of the General tab gives an extensive look at the Data and Control planes with each split into Meta and Operating System (OS) data.



Actions

The General tab also allows users to adjust the Data Refresh period for all Monitoring pages in wXcked Eye. This gives users control on how often the data is updating with the default value set to the minimum of 5 seconds.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Mesh with wXcked Eye

Navigating to the Mesh tab

The Mesh tab is located on the Settings page in wXcked Eye. To learn how to navigate there from the wXcked Eye main page, please review the **Configure cloudSwXtch with wXcked Eye** article.

The **Mesh** tab is organized into 4 functions:

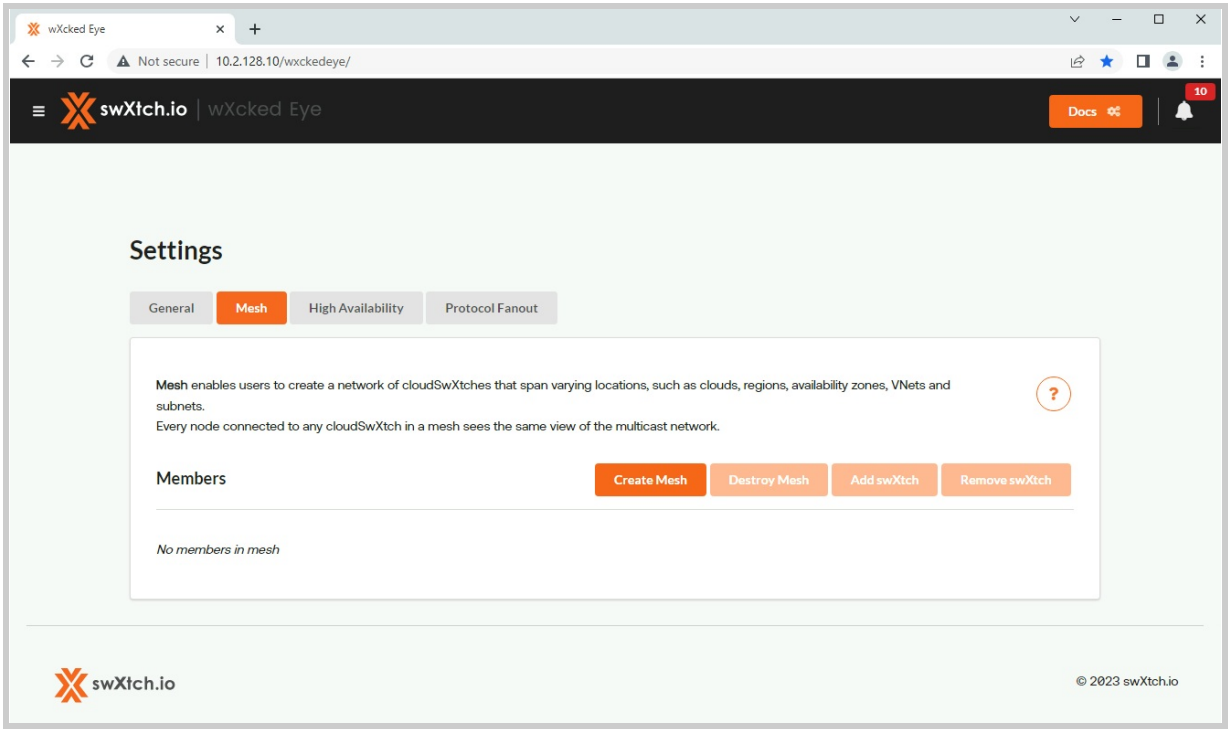
- Create Mesh
- Destroy Mesh
- Add SwXtch
- Remove SwXtch

When there is no mesh, the page will be blank as seen in the example above with a "No members in mesh" disclaimer. The "Create Mesh" button will be the only one activated. This section will explain how to create a mesh, add/remove cloudSwXtch(s) and destroy an existing mesh. If a user wishes to use commands in a terminal instead, please read the following article on configuring mesh.

Mesh Command-Line Alternatives

In addition to configuring your mesh through the wXcked Eye UI, users can also use swXtch specific commands in their terminal. To learn more, please see the Mesh article under Configuring cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



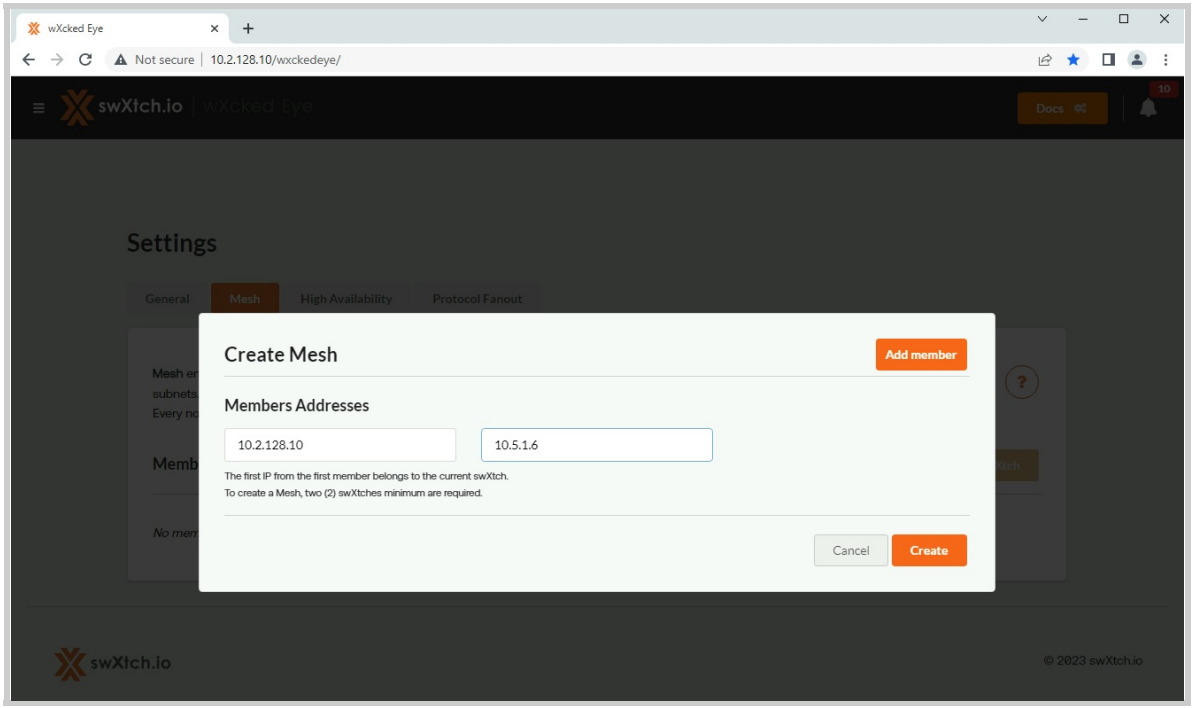
Mesh and High Availability

Both features are mutually exclusive and cannot be used together.

Create Mesh

Creating a mesh using the wXcked Eye UI is a relatively straight forward process. To start:

1. **Click** on the "Create Mesh" button. A new window will pop up.



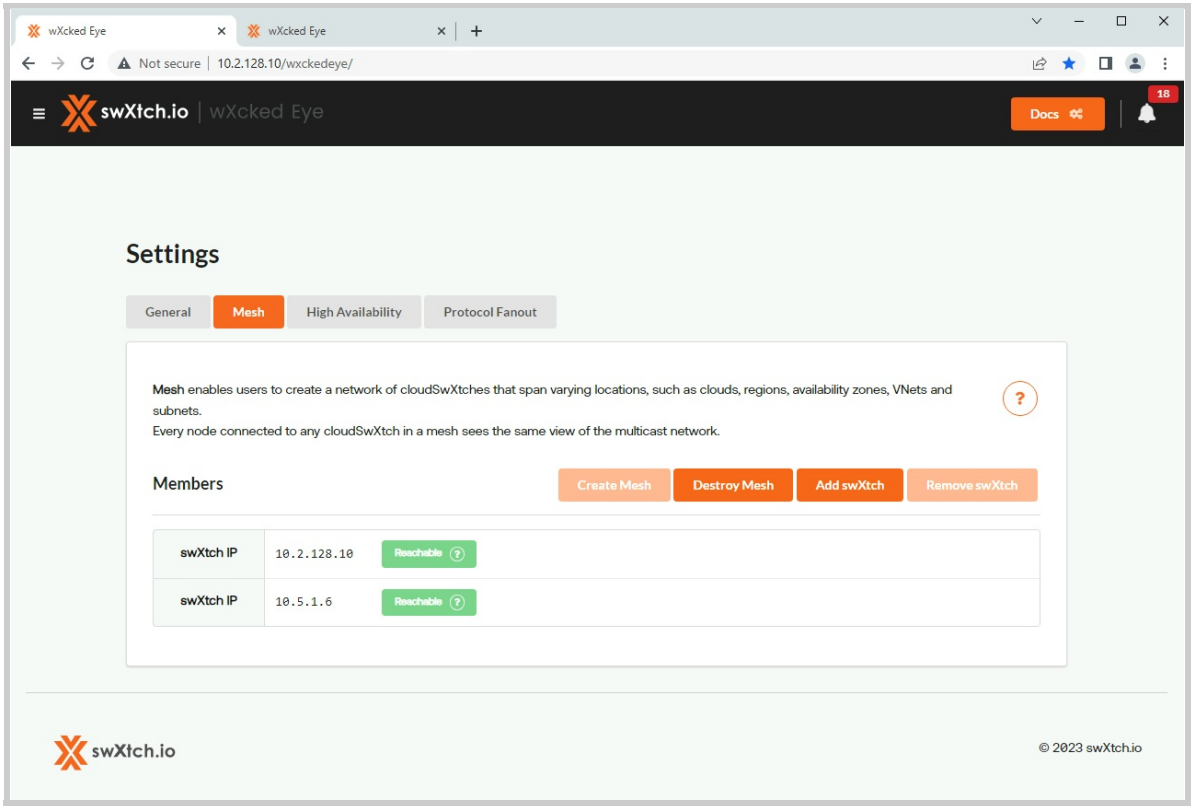
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Add the cloudSwXtch IP address you wish to add to the Mesh. The current cloudSwXtch you are on will **automatically** fill in the first slot. In this case, 10.2.128.10.

a. If you want to add additional cloudSwXtches to the Mesh, select the "Add Member" in the top right corner. This will display an additional IP address field.

b. **Please note:** You **must** enter **at least** 2 cloudSwXtches in order to successfully create a mesh.

3. **Click** "Create" after you have completed adding members. **The members should now be listed in the newly formed mesh. A tag on each swXtch IP will display whether or not the VM is reachable.**

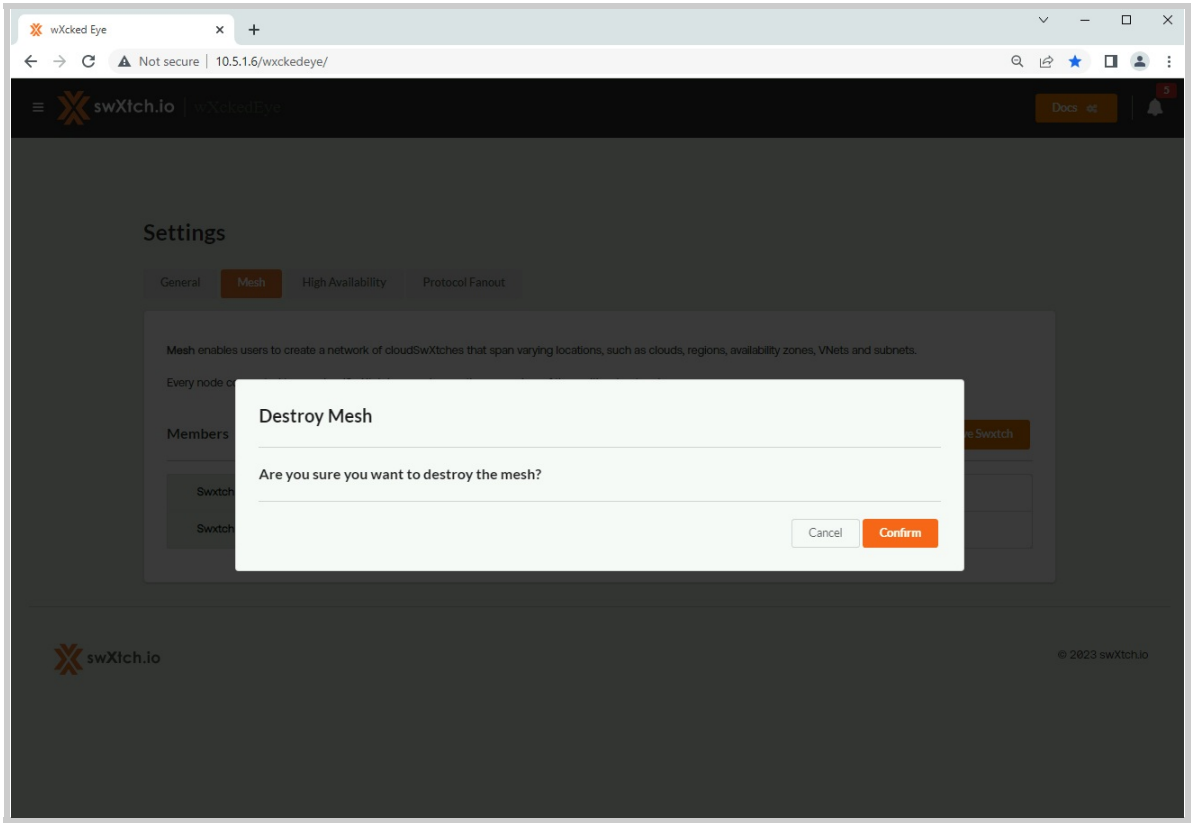


If a user were to view the wXcked Eye of another cloudSwXtch in the mesh, they would be able to see the other members as well. For example, if a user was on the wXcked Eye for cloudSwXtch 10.5.1.6 instead, they would see the same member list.

Destroy Mesh

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. **Click** "Destroy Mesh." A warning pop-up will appear.



2. **Select** "Confirm." **Your mesh and its members should no longer be listed in the Mesh tab.**

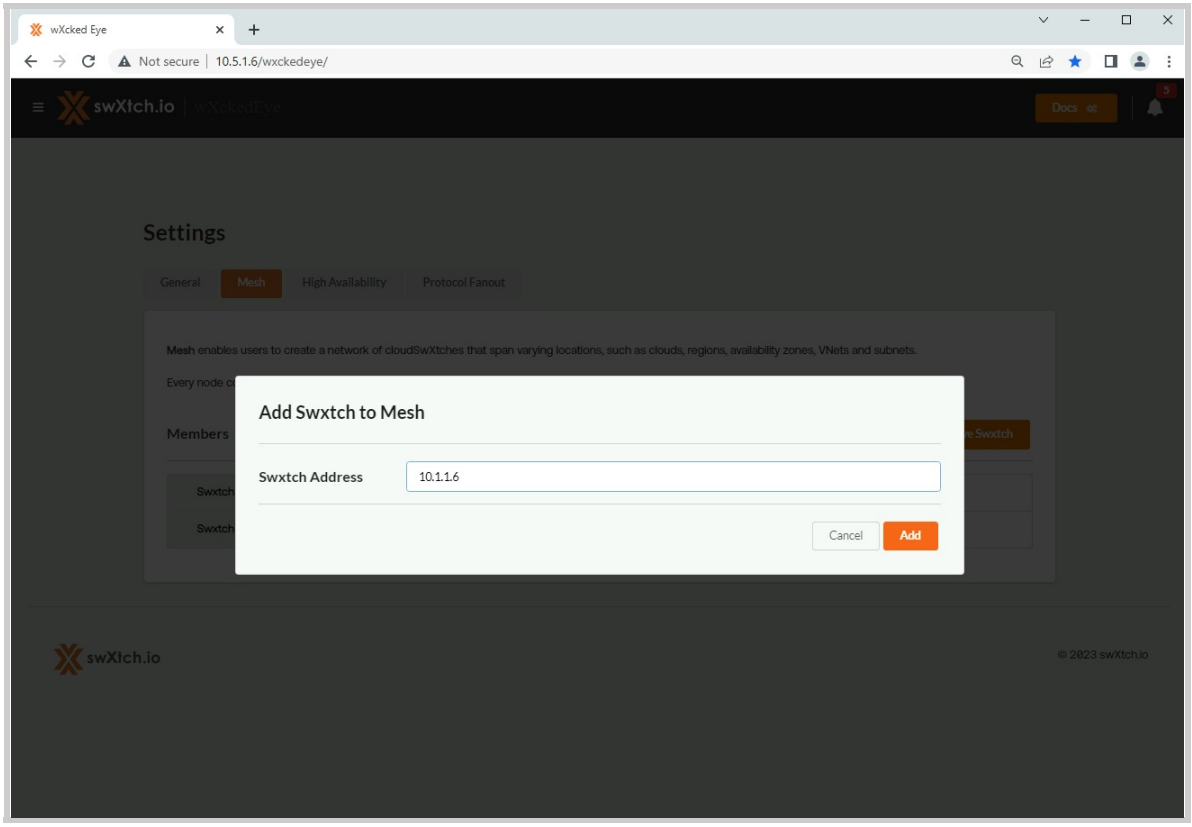
Add SwXtch

After creating a mesh, users may decide that they need additional cloudSwXtches. To add a cloudSwXtch:

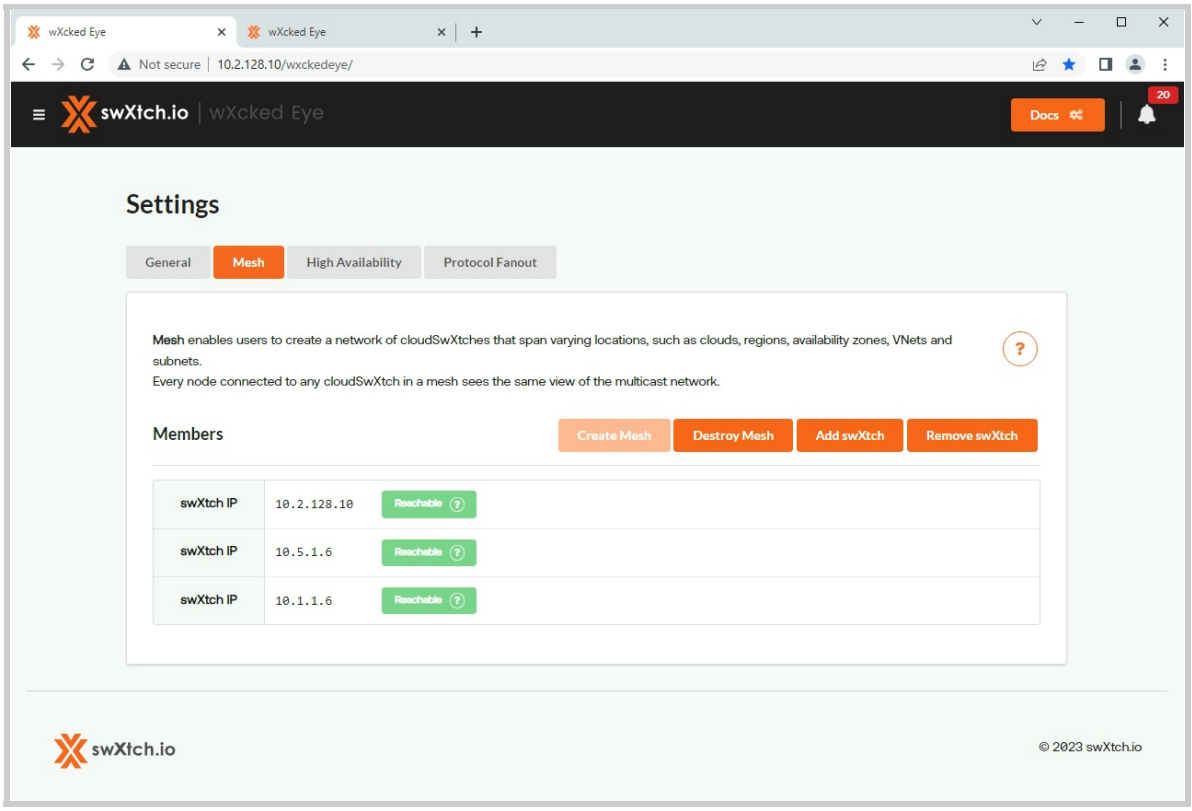
1. **Click** on the "Add SwXtch" button in the Mesh main page of any existing cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. **Enter** the IP address of the cloudSwXtch you wish to add to the mesh.



3. **Click** "Add." A new cloudSwXtch should now be listed in your mesh.

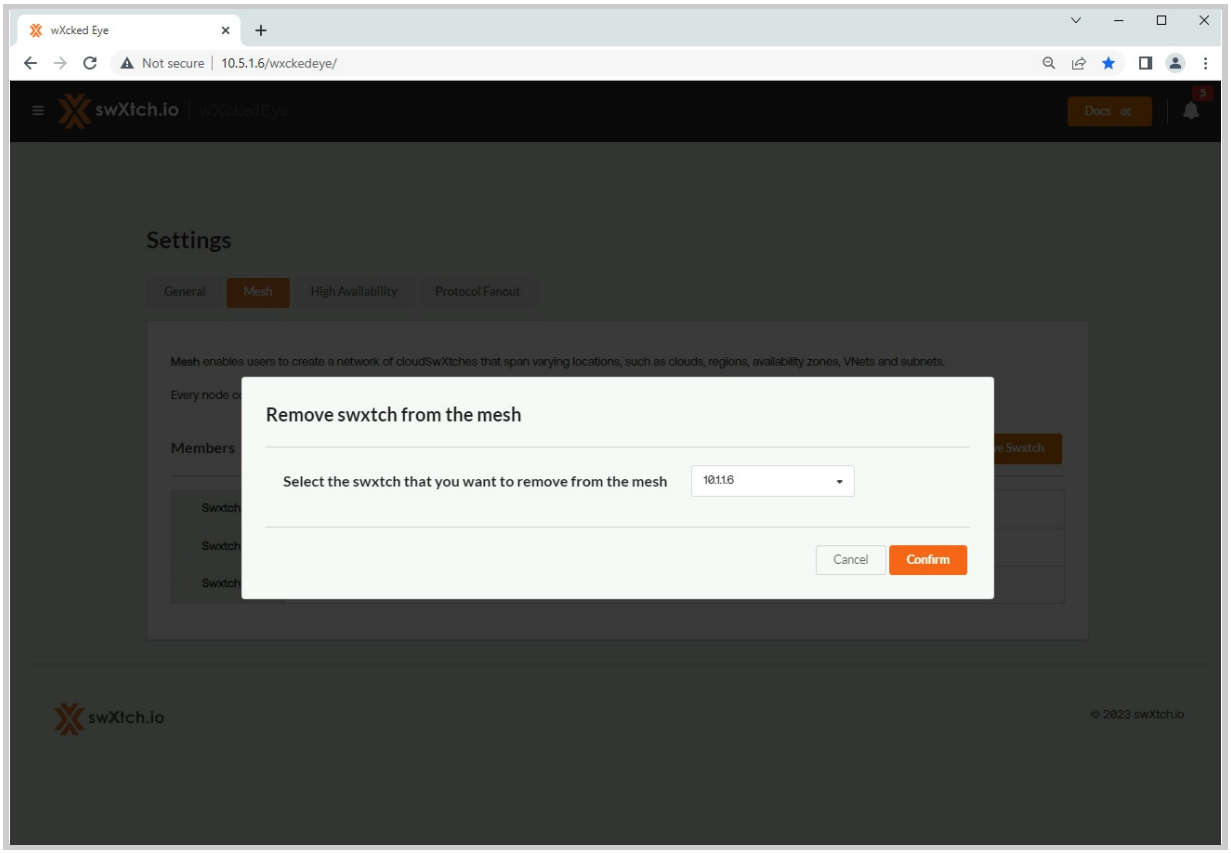


Remove SwXtch

1. **Click** the "Remove Swxtch" option. A new window will open confirming the removal.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. **Use** the dropdown menu to select the cloudSwXtch you would like to remove from your mesh configuration.



3. **Select** "Confirm." If you are on the mesh settings page of the cloudSwXtch you removed, the page should now appear blank. However, if you are on a different cloudSwXtch, the page should still be populated with the other remaining cloudSwXtches.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

High Availability with wXcked Eye

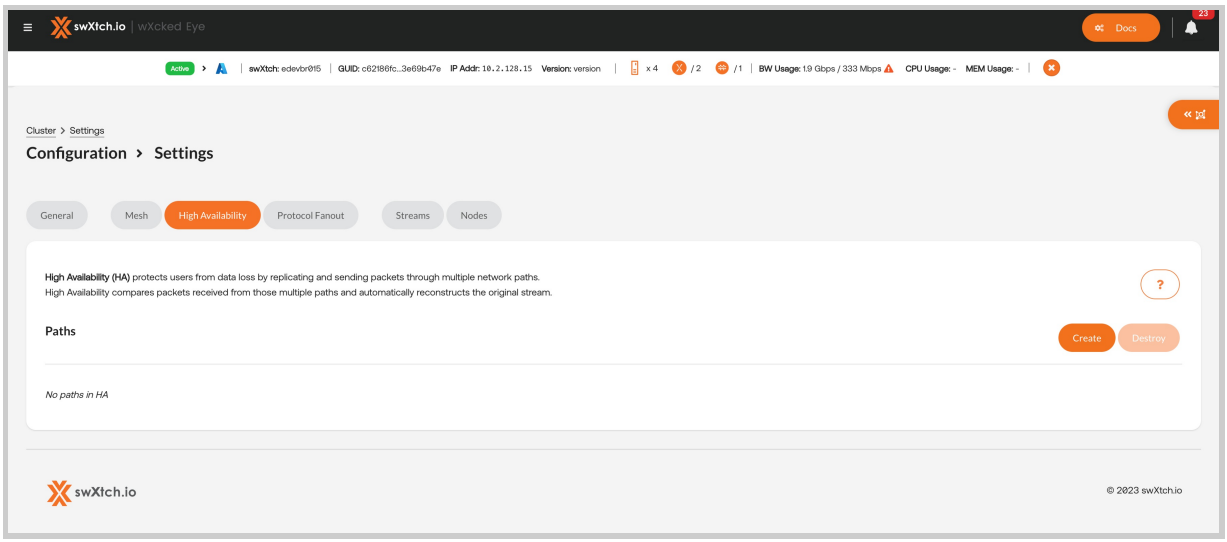
Navigating to the High Availability tab

The High Availability tab is located in the Settings page on wXcked Eye. To learn how to navigate there, please review the **Configure cloudSwXtch with wXcked Eye** article.

The High Availability tab is organized into 2 functions:

- Create HA
- Destroy HA

In this section, we will discuss each tab and how it offers a user additional control over their cloudSwXtch network.



High Availability Command-Line Alternatives

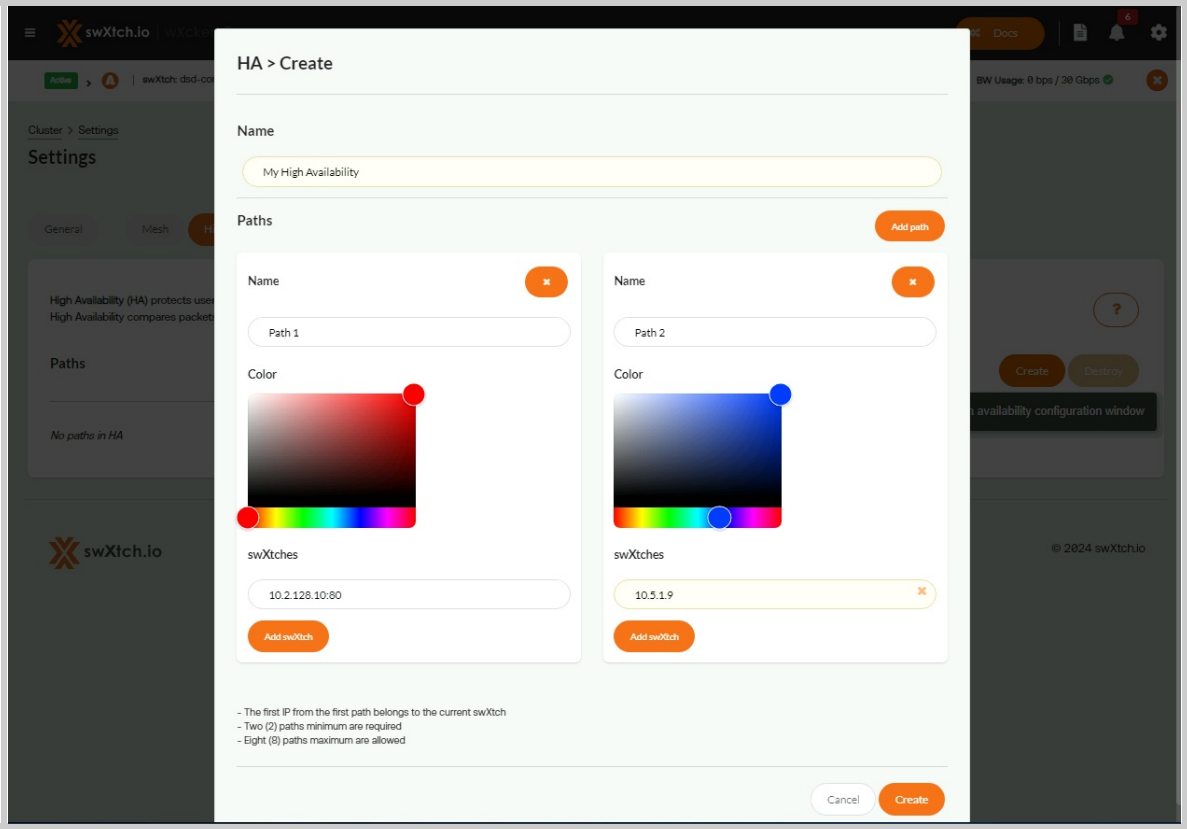
In addition to configuring your high availability through the wXcked Eye UI, users can also swXtch specific commands in their terminal. To learn more, please visit the **High Availability** article under Configuring cloudSwXtch.

Create HA

1. **Click** the "Create" button in the cloudSwXtch you wish to include in your high availability configuration. A pop-up will open.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. **Name** your HA configuration. In this example, the HA is named "My High Availability."



3. **Name** the first path. The IP address of the cloudSwXtch you are currently on will populate. Enter **IP addresses** for any additional relevant cloudSwXtches.

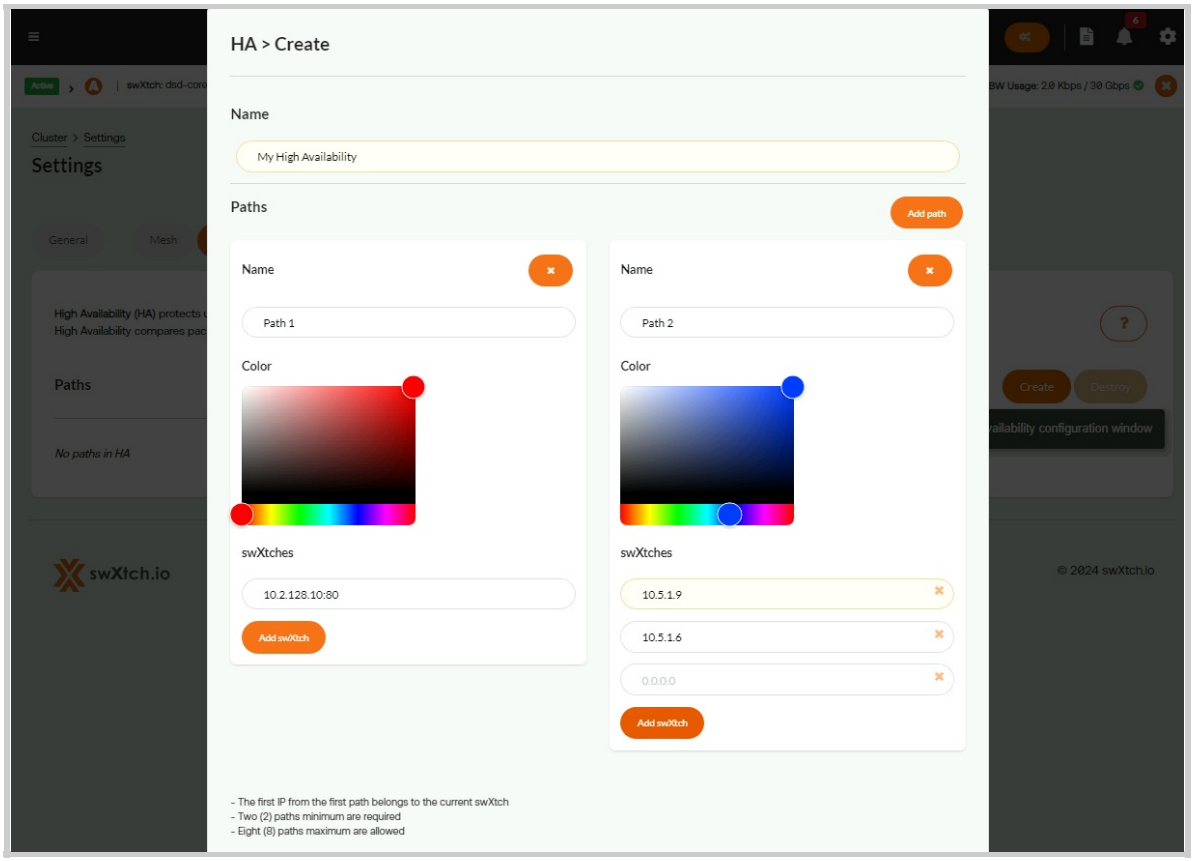
4. **Name** the second path and enter the **IP addresses** for the relevant cloudSwXtches.

Note: You must have *more than 1 path* in order to have a working HA flow.

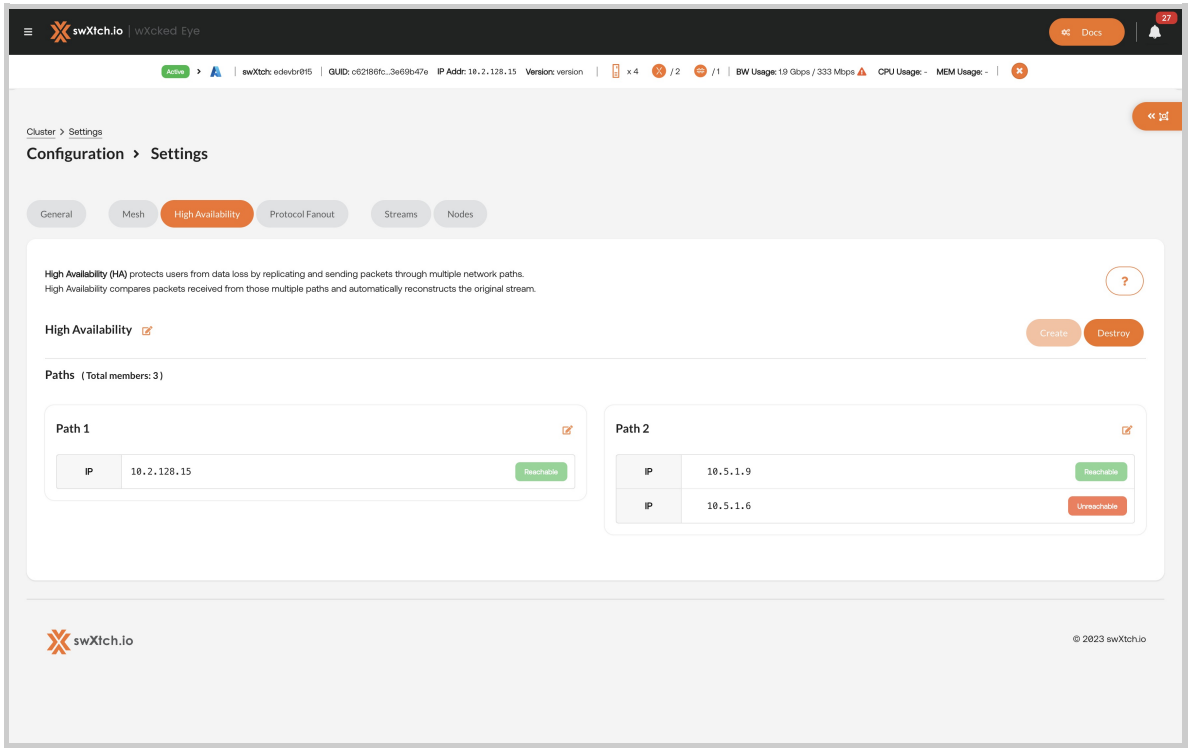
5. **OPTIONAL**: Add an additional cloudSwXtch to a Path by clicking "Add SwXtch." In this example, the user assigned 2 cloudSwXtches to Path 2.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

6. Assign a color for both paths. Here, the user selected Red for Path 1 and Blue for Path 2.



7. **Click** "Create." A new High Availability flow has been created.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

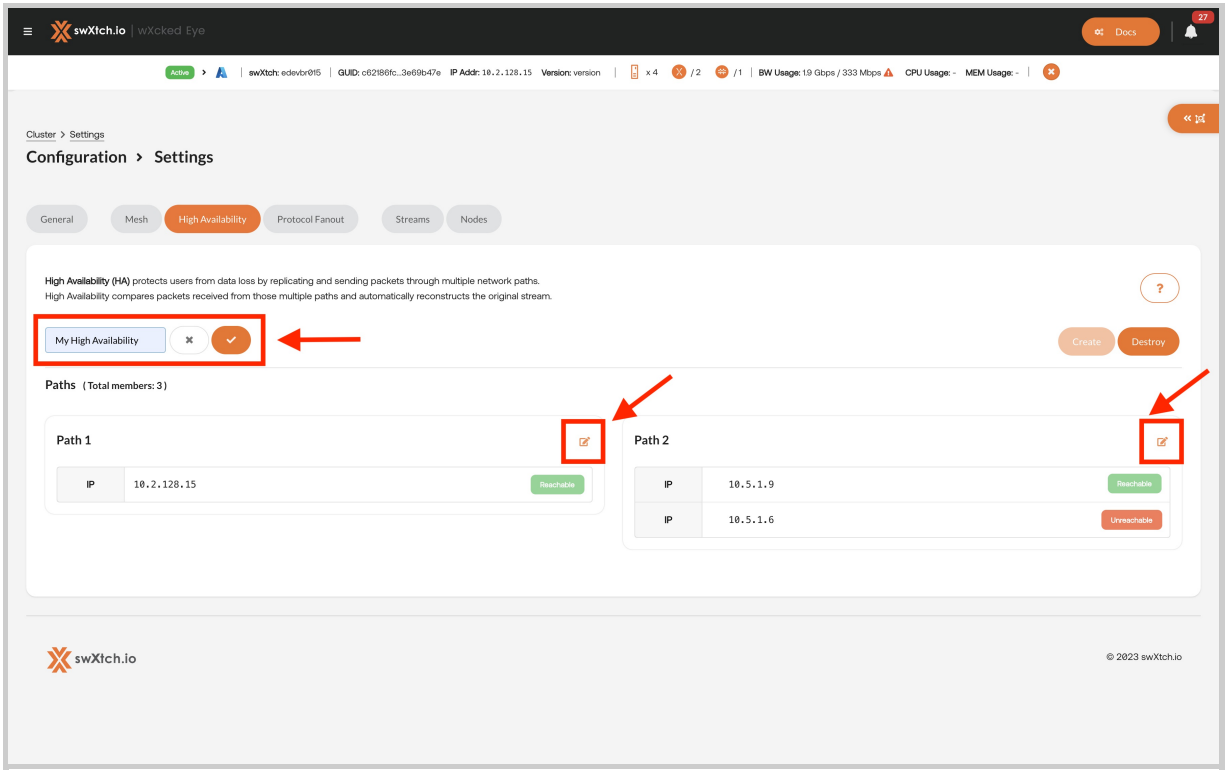
With High Availability now configured, users can switch between different cloudSwXtches in their HA, refresh the HA page and see the members listed with their associated paths. For example, if a user were to look at the wXcked Eye for cloudSwXtch 10.5.1.6 instead of the above 10.2.128.10, they will see the same My High Availability member list.

From any of the connected cloudSwXtches, users can destroy their HA configuration.

> **Warning**
>
> If you try to create another HA, the wXcked Eye UI will destroy the current HA and replace it with the new configuration.

Renaming High Availability and Path Names

Users can dynamically change their high availability and paths' names directly in the UI. To do this, simply click the edit button next to the paths.

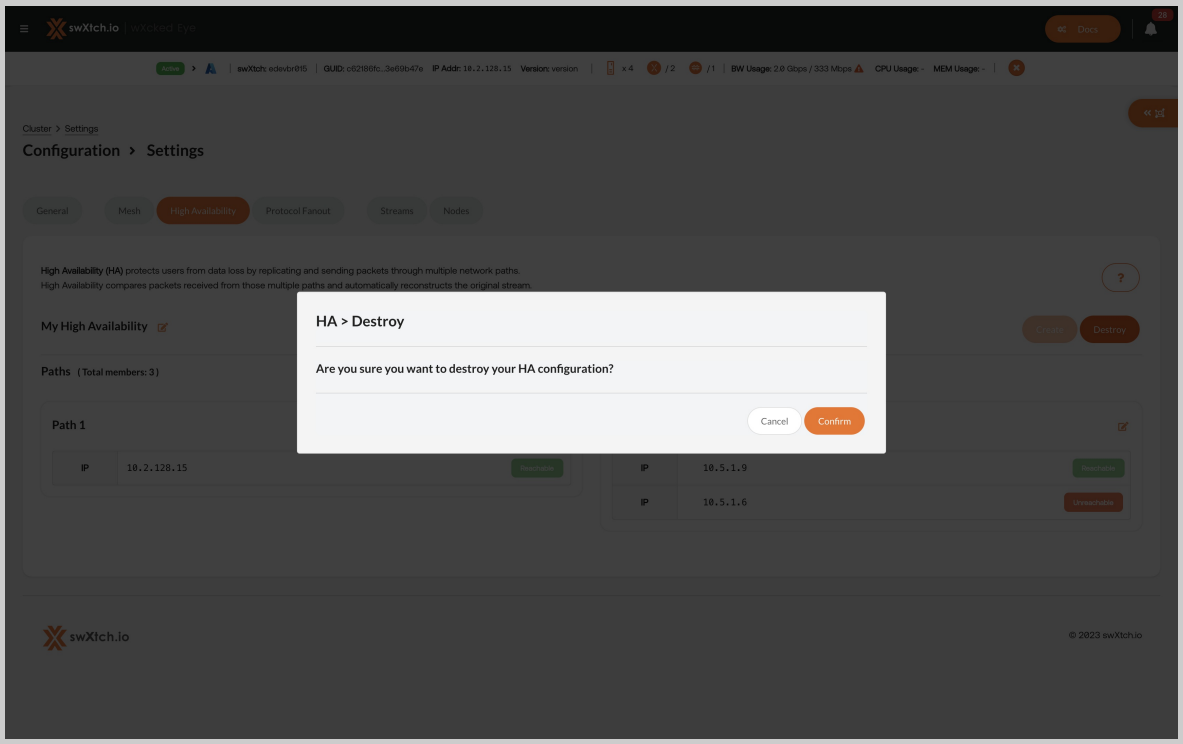


Destroy HA

To destroy an HA configuration, a user will need to go to the HA page of any associated cloudSwXtches.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. **Click** "Destroy HA." A new prompt will appear, asking you to confirm the action.



2. **Select** "Confirm."

Your HA is now destroyed. All associated cloudSwXtches will show a blank list for HA and your Cluster page will be empty.

Removing a cloudSwXtch from an HA configuration

To remove a cloudSwXtch from a user's HA configuration, they will need to delete and recreate their HA cluster without that cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Protocol Conversion and Fanout with wXcked Eye

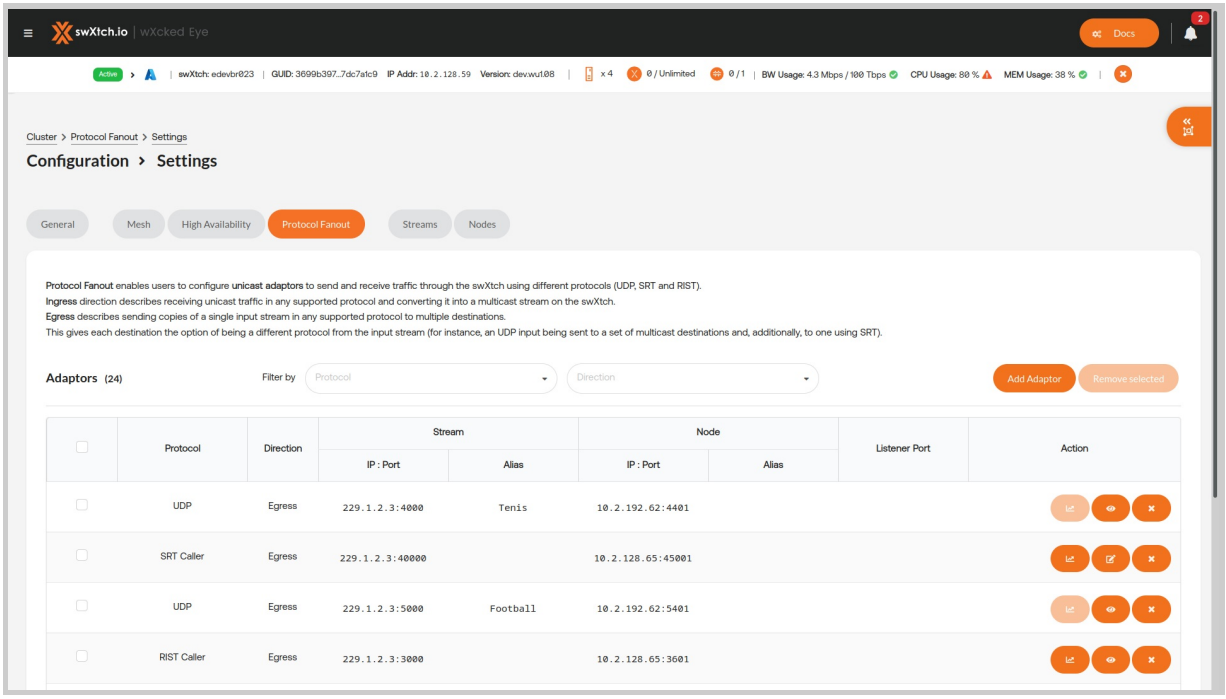
WHAT TO EXPECT

The Protocol Fanout tab can be found on the Settings page in wXcked Eye. To learn more about how to navigate there, please review the Configure cloudSwXtch with wXcked Eye article.

In this article, users will learn how to establish UDP, SRT and RIST connections via the Protocol Conversion and Fanout feature on wXcked Eye. It can also be used to convert multicast into one or more of these protocols. For example, imagine you have an SRT stream coming into the cloudSwXtch with five different clients requiring different protocols (one needing SRT, another RIST, another UDP and the last two for multicast). This can be accomplished by using this tool. **For a walkthrough on configuring Protocol Conversion and Fanout in wXcked Eye**, see the Protocol Conversion and Fanout Example article.

Setting Up Aliases

The Protocol Fanout tab utilizes Stream and Node names set up in the Aliases tab. For more information on how to do this, please see the Aliases article.



Protocol Fanout and Conversion is a cloudSwXtch feature that allows users to send copies of a single input stream in any supported protocol to multiple destinations. In wXcked Eye, users can send/receive UDP traffic or establish SRT/RIST caller/listener connection methods.

To add an adaptor, select "Add Adaptor." Here, a user can select between the three protocols.

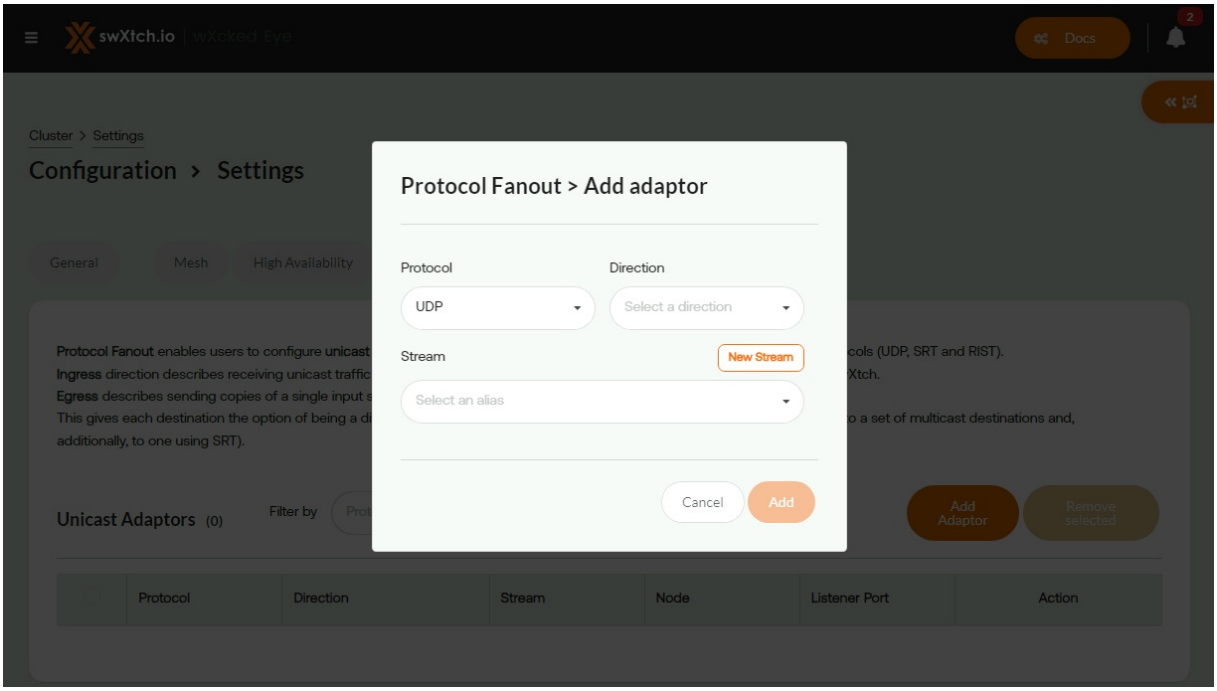
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

UDP

Ingress vs. Egress

Differentiating between ingress and egress can be difficult. It is important to imagine it in **relation to the cloudSwXtch**.

- **Ingress**: Data is coming into the cloudSwXtch.
- **Egress**: Data is leaving the cloudSwXtch.

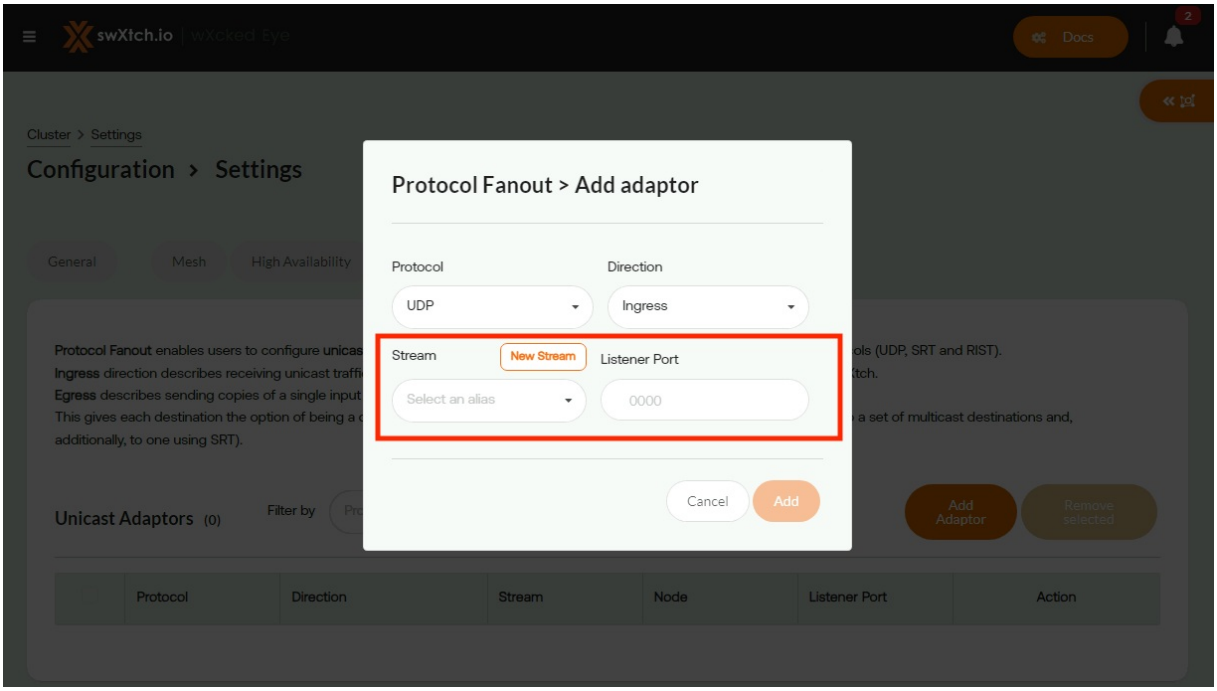


Selecting UDP under Protocol allows users to add mapping for UDP traffic entering and leaving the cloudSwXtch. Depending on the direction of the data, a user will have to add additional information to set up a successful connection.

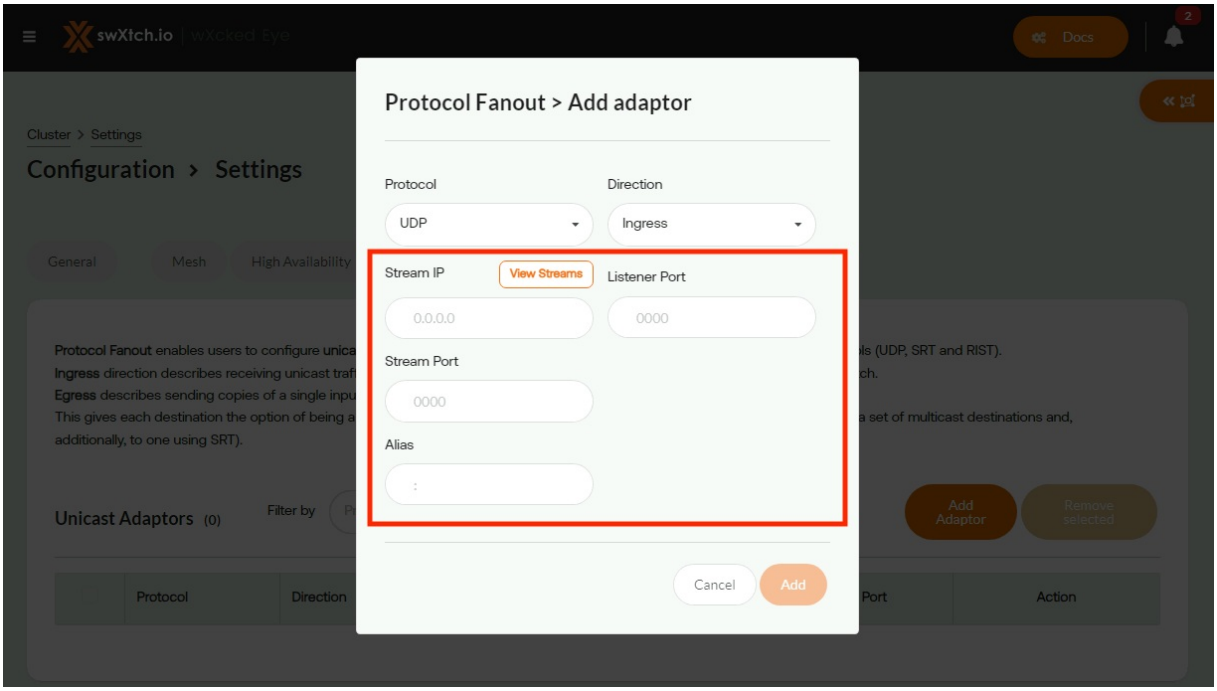
UDP Ingress

For Ingress, a user will select one of the **Streams** created in the **Aliases** tab from the dropdown and designate a **Listener Port**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



Alternatively, if a user does not have an Alias set up for their desired stream, they can manually specify a **Stream IP, Listener Port, a Stream Port and an Alias name** to add one. To do this, they would have to select the **New Stream** option in the panel.

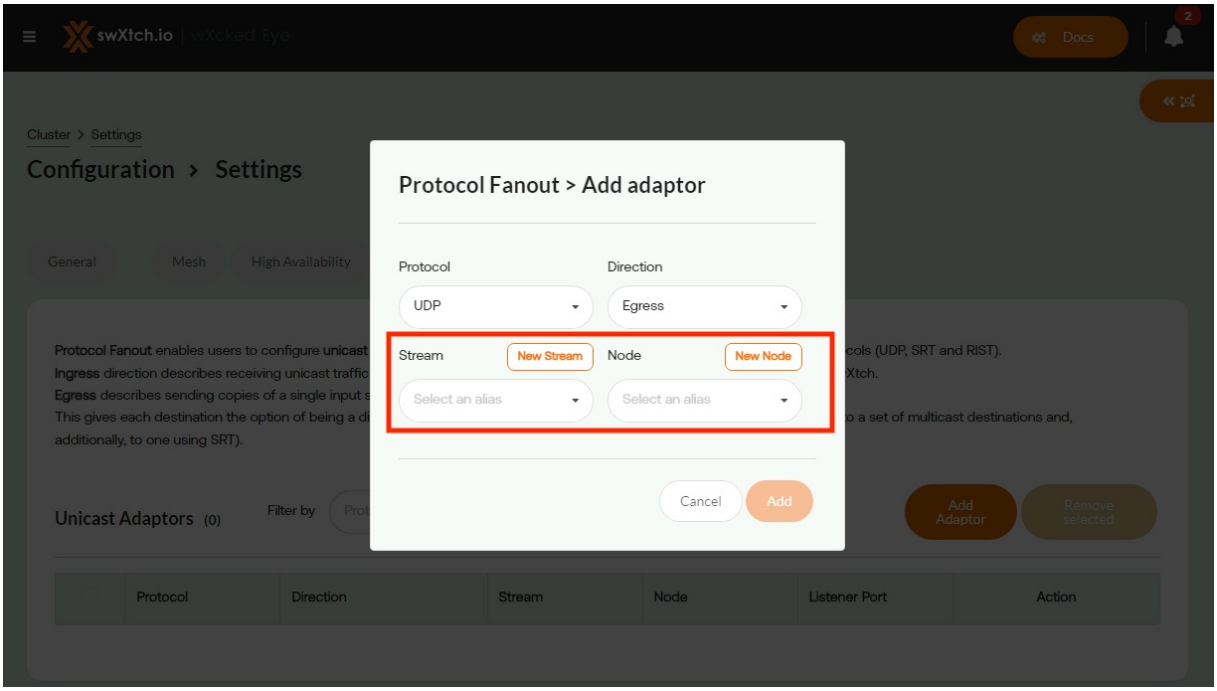


Both methods will allow endpoints to send unicast data. Once that connection has been established, the cloudSwXtch will be able to ingest the unicast data as multicast.

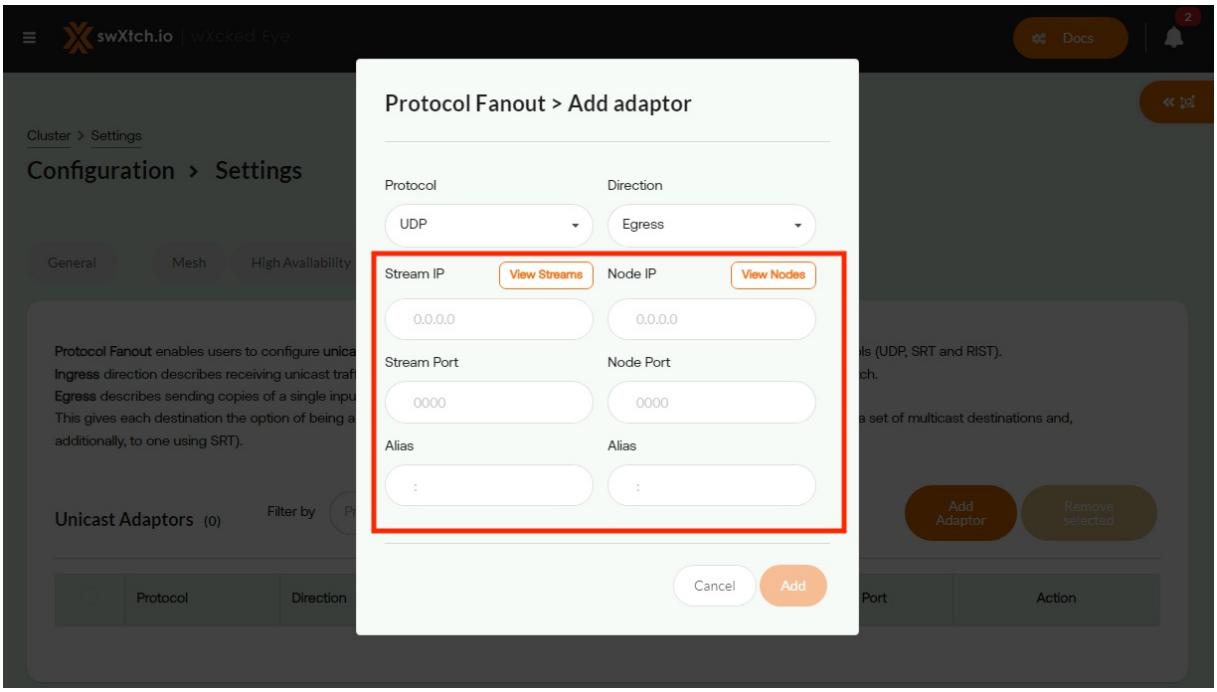
UDP Egress

For Egress, a user will set the parameters for fanning out a multicast stream as unicast. To do this, the cloudSwXtch would need to select a **Stream** and **Target Node** from their respective **Alias** dropdown.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



If an **Alias** was not created prior, they can manually add a **New Stream** and **New Node** by entering the appropriate information for both. To create a **New Stream**, a user will need the **Stream IP, Stream Port and Alias name**. Similarly, a **New Node** would require a **Node IP, Node Port and Alias name**.

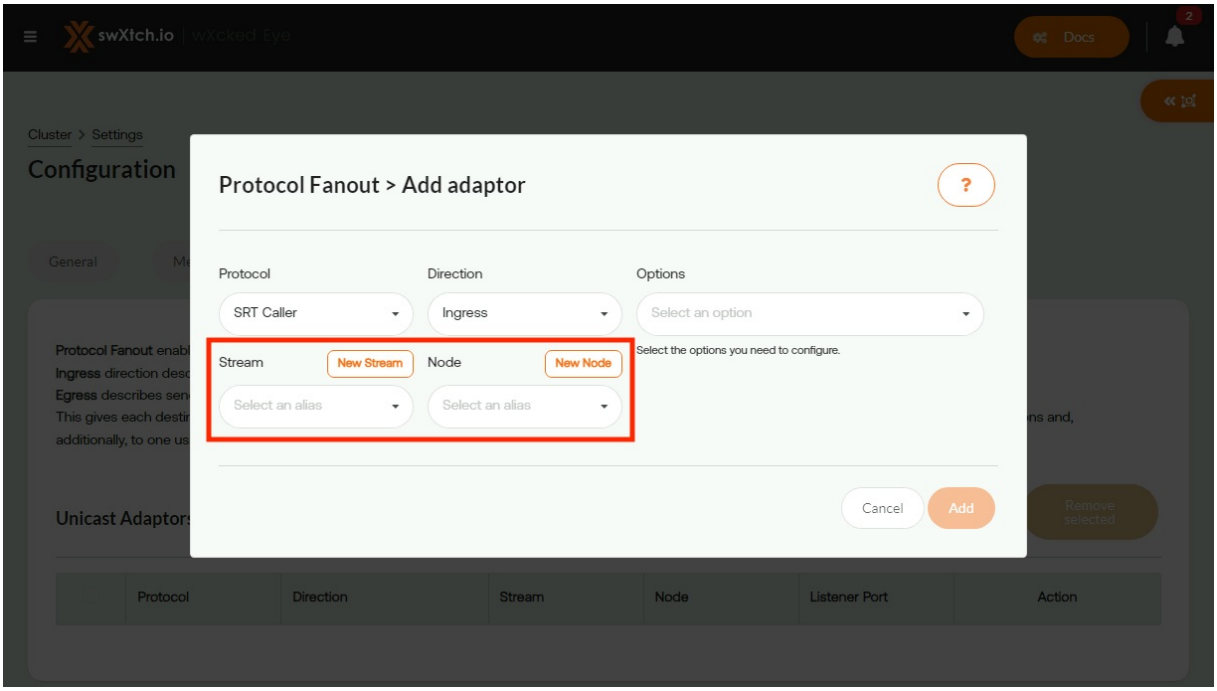


Whether it is from existing Aliases or ones manually created, this will allow the cloudSwXtch to transmit a multicast stream as unicast to a desired endpoint.

SRT and RIST Caller

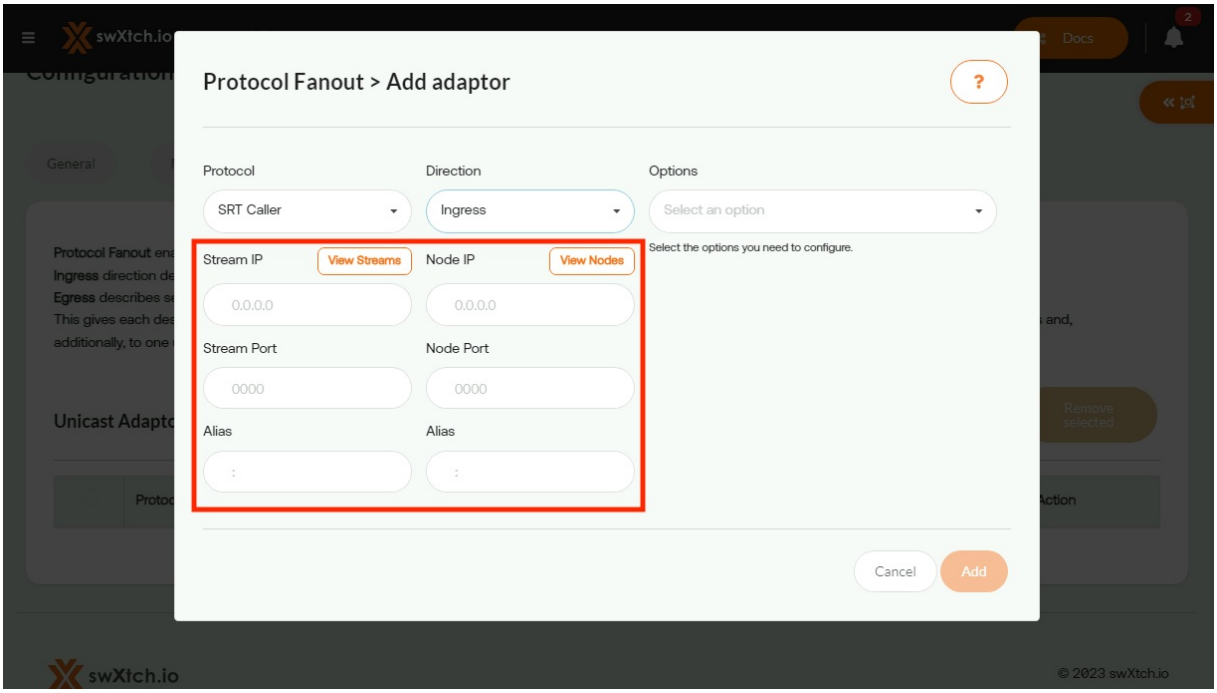
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

To set up **SRT** or **RIST Caller**, a user will need to choose what direction they will like their data to flow: Ingress or Egress. Regardless of their choice, both Ingress and Egress requires a selection for Stream and Node. These can either be selected from the Alias dropdown (assigned in the Alias tab under Settings) or manually created in the panel.



If a user wants to manually create a stream and node, they will need to select the "New Stream" and "New Node" buttons. This will reveal additional fields necessary for both. For Stream, a user will need to enter a Stream IP, Stream Port and Alias name. For Node, they will need a Node IP, Node Port and Alias. This will act as a Target Node. It is the source of where the traffic will be coming from outside the cloudSwXtch. This information is crucial since it will dictate where the cloudSwXtch sends the caller message.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



The **Options** dropdown allows for additional fine tuning based on standards of the protocol. Selection of an option will open another field.

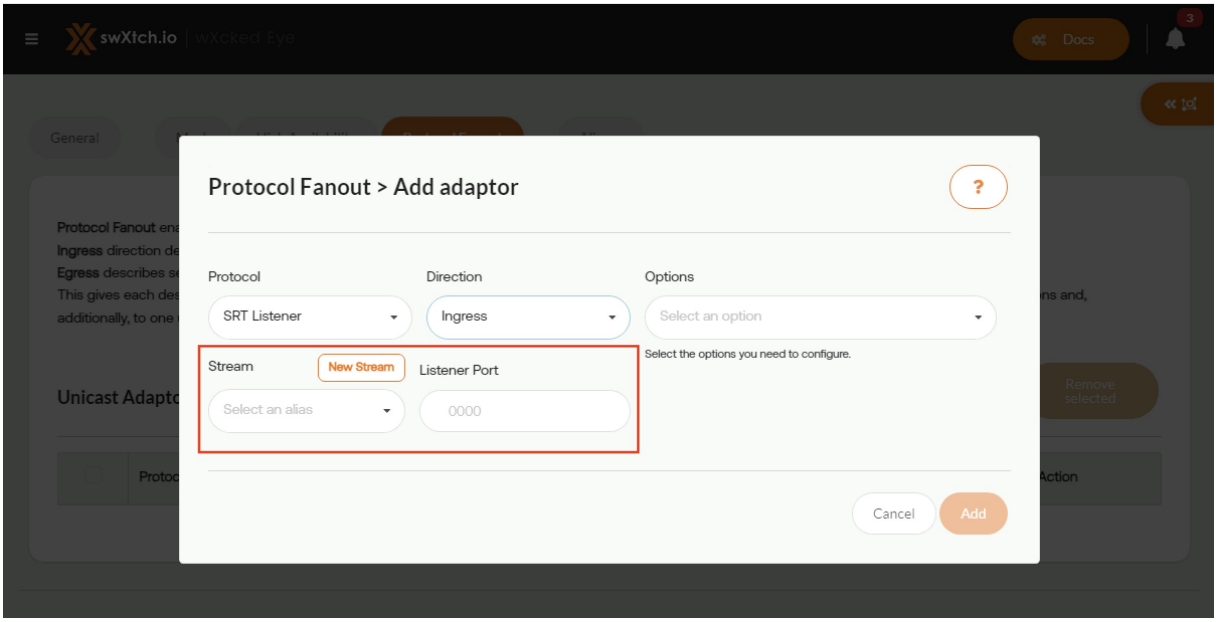
After filling out all the required fields, select "Add." The cloudSwXtch will then call out to the target source and receive multicast traffic through the designated node.

For Egress, the user will be specifying the Target Node for an endpoint to receive an SRT or RIST stream from the cloudSwXtch. The cloudSwXtch will then call to the Destination or Target Node to establish a connection before transmitting the stream.

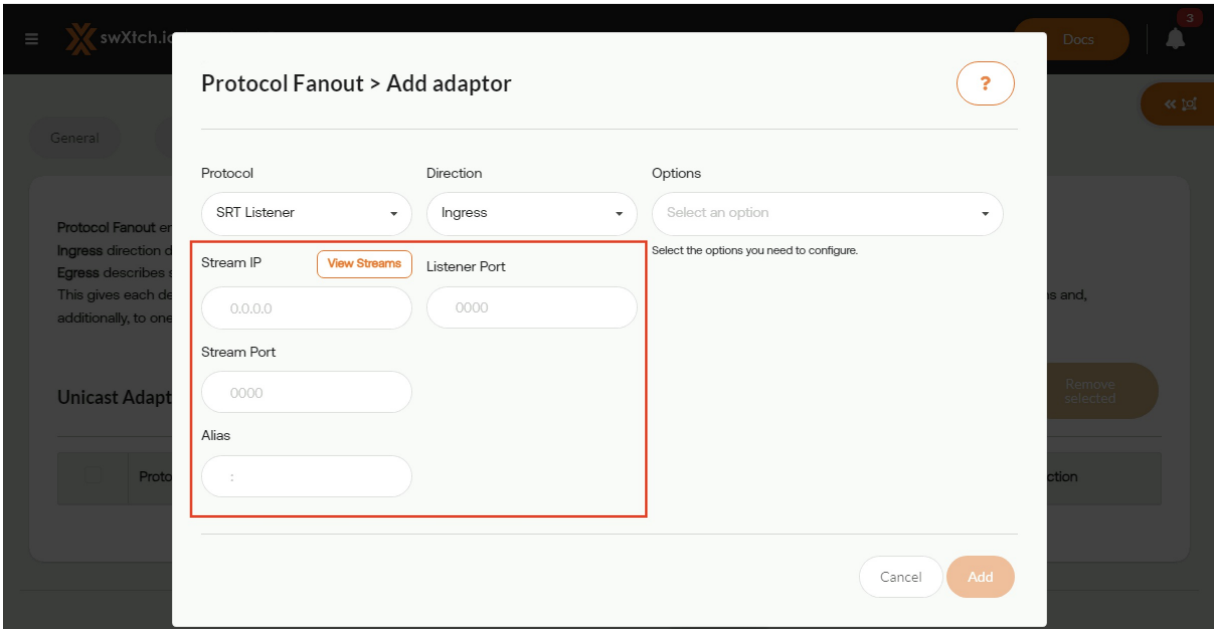
SRT and RIST Listener

Similar to the SRT or RIST Caller panel, Listener requires the user to specify the direction of their data flow: Ingress or Egress. However, what differs is that the SRT or RIST Listener is essentially "listening" for any incoming messages from endpoints ready to send/receive SRT or RIST data. This method of transmission is considered to be more user-friendly since a user will not have to worry about pointing to a specific IP address. It places the burden of targeting the endpoint instead.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



Both Ingress and Egress will require a user to select a multicast Stream from the Alias dropdown. Stream Aliases are assigned in the Alias tab under Settings. Alternatively, users can enter a new Stream by selecting the New Stream button and entering the following information: Stream IP, Stream Port and Alias Name. In addition to the Stream, a user will also need to specify a Listener Port where an endpoint can send data through.



The **Options** dropdown allows for additional fine tuning based on standards of the protocol. Selection of an option will open another field.

For Ingress, once the configuration is complete, the cloudSwXtch will now listen for producers of SRT or RIST traffic who connect to the port. When a connection has been established, the cloudSwXtch will begin ingesting data.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Likewise for Egress, the cloudSwXtch is listening for endpoints that are trying to receive SRT or RIST data. From there, depending on the user's bandwidth, they can create up to 32 Listener ports from which an endpoint can connect to the the steam. By setting the necessary parameters, a consumer will then be able locate a target port and begin streaming data from the cloudSwXtch.

Protocol Conversion and Fanout Example

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Protocol Conversion and Fanout Example

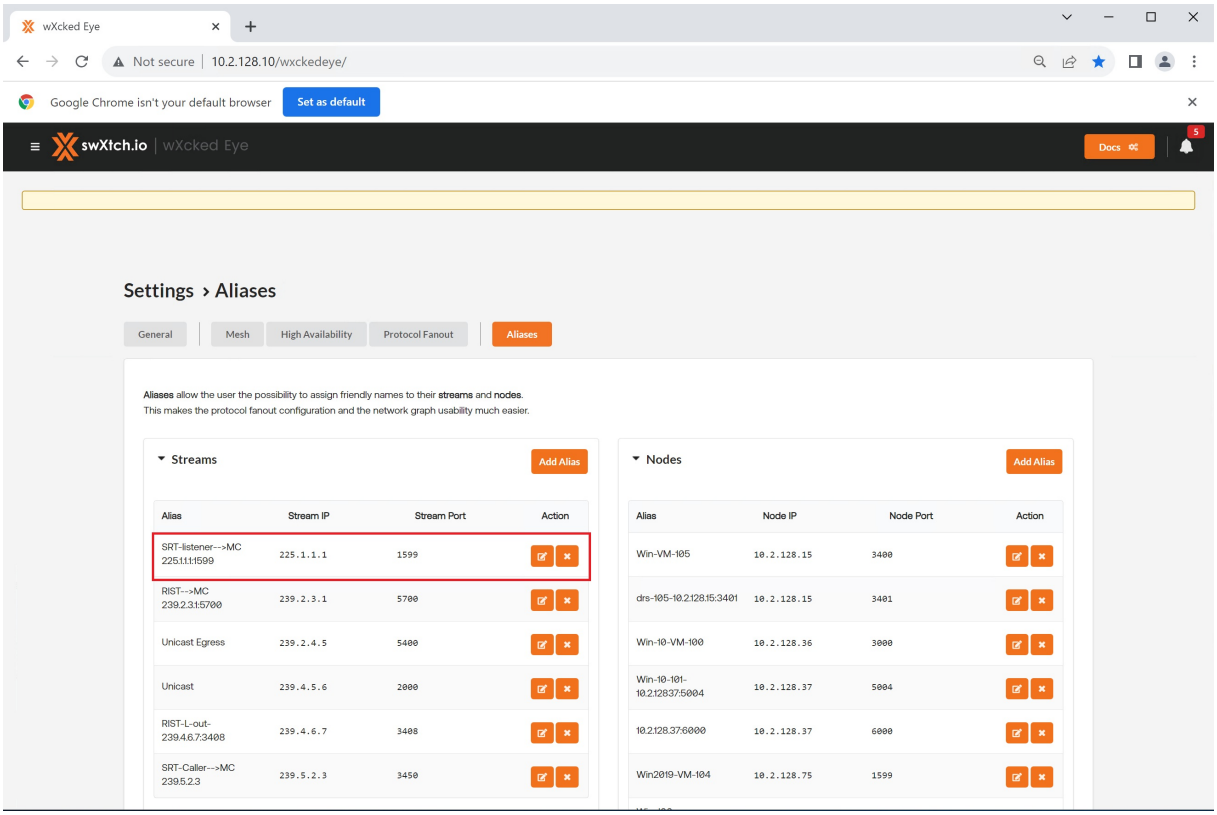
WHAT TO EXPECT

Navigating the Protocol Fanout and Conversion tab in wXcked Eye can be a little confusing when first starting out.

In this article, we will walk you through a typical SRT Listener configuration workflow to explain the various pieces that go into setting it up. We will look at the differences between ingress and egress and what that means in relation to the cloudSwXtch.

Step One: Setting up Your Aliases

The Aliases tab allows users to set friendly names or "aliases" for their streams and nodes so that it is easier to organize them in the Protocol Fanout tab. In this example, the user has named their stream **SRT-listener -> MC 225.1.1.1:1599,** inputting a stream IP of 225.1.1.1 and a Stream Port of 1599. This name is helpful for the user because it illustrates what they hope to do when setting up for Protocol Fanout. They are going to set up for an SRT to MC conversion using the stream IP and Stream Port assigned to the name.

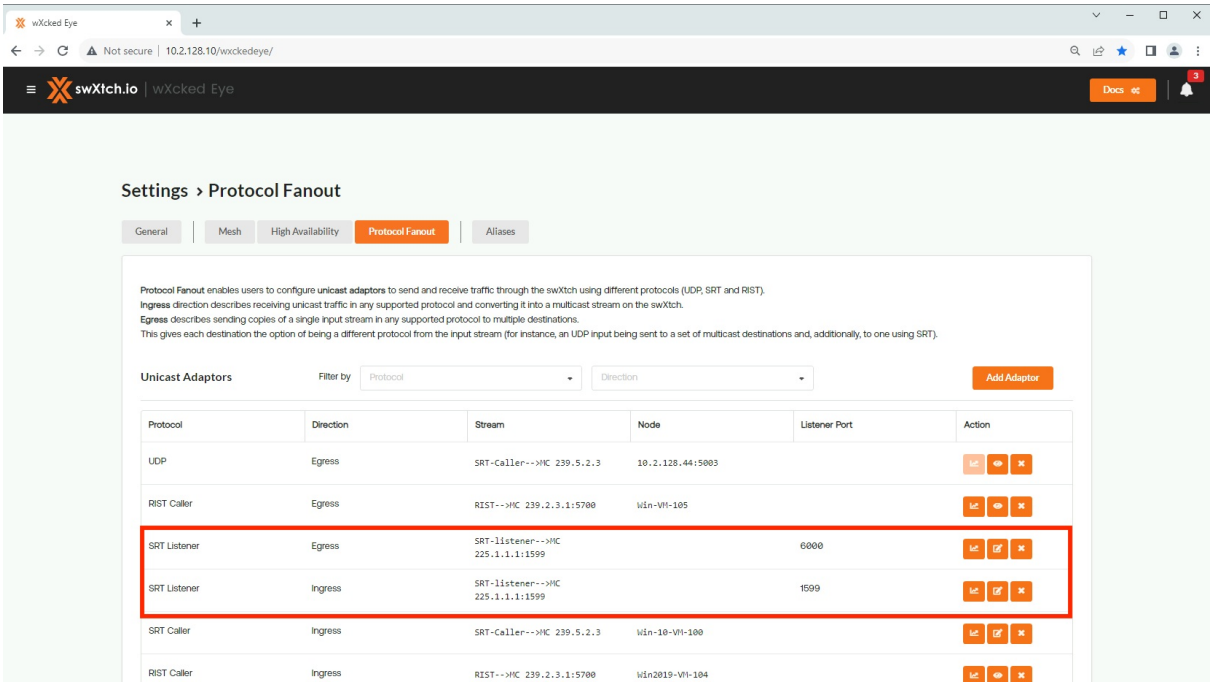


For more information about how aliases work, see the Aliases article under Configure cloudSwXtch with wXcked Eye.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Step Two: Adding Adaptors

In the Protocol Fanout Settings page, the user has set up two SRT Listeners. An SRT Listener configuration is telling the cloudSwXtch to listen for any incoming messages from endpoints ready to send/receive SRT data. This method of transmission is considered to be more user-friendly since a user will not have to worry about pointing to a specific IP address. It places the burden of targeting on the endpoint instead.



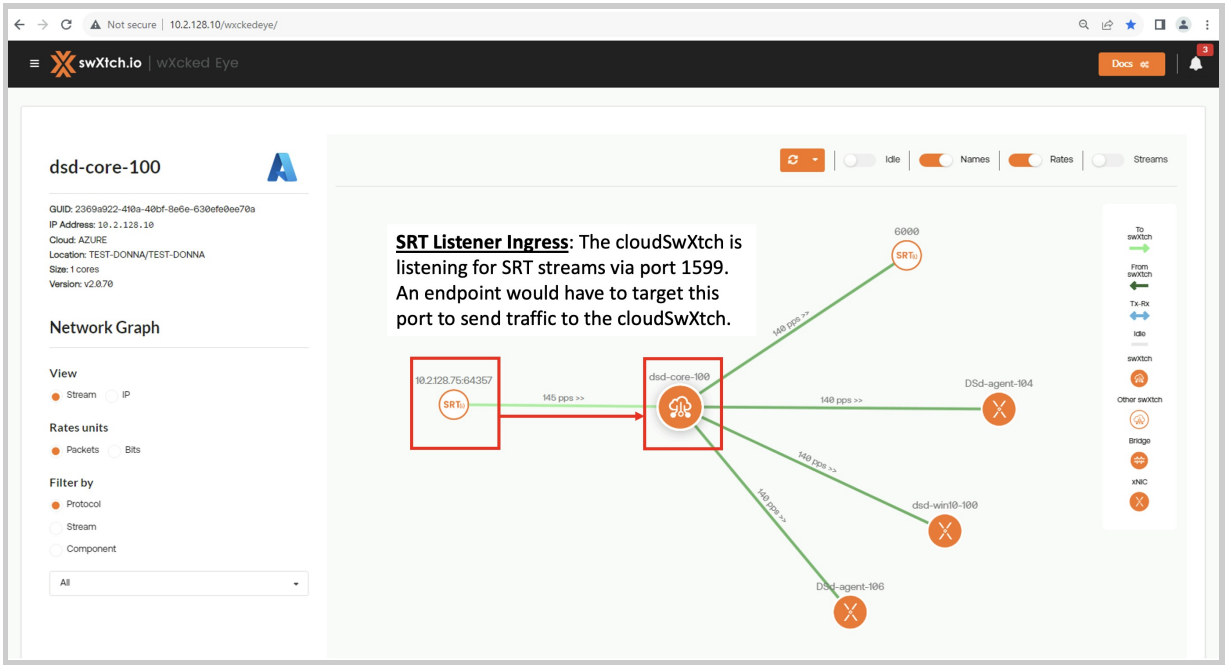
The user has set up SRT Listener for both Egress and Ingress using the Alias assigned earlier: **SRT-Listener -> Multicast 225.1.1.1:1599**. When differentiating between Egress and Ingress, always imagine it from the perspective of the cloudSwXtch:

SRT Listener Ingress

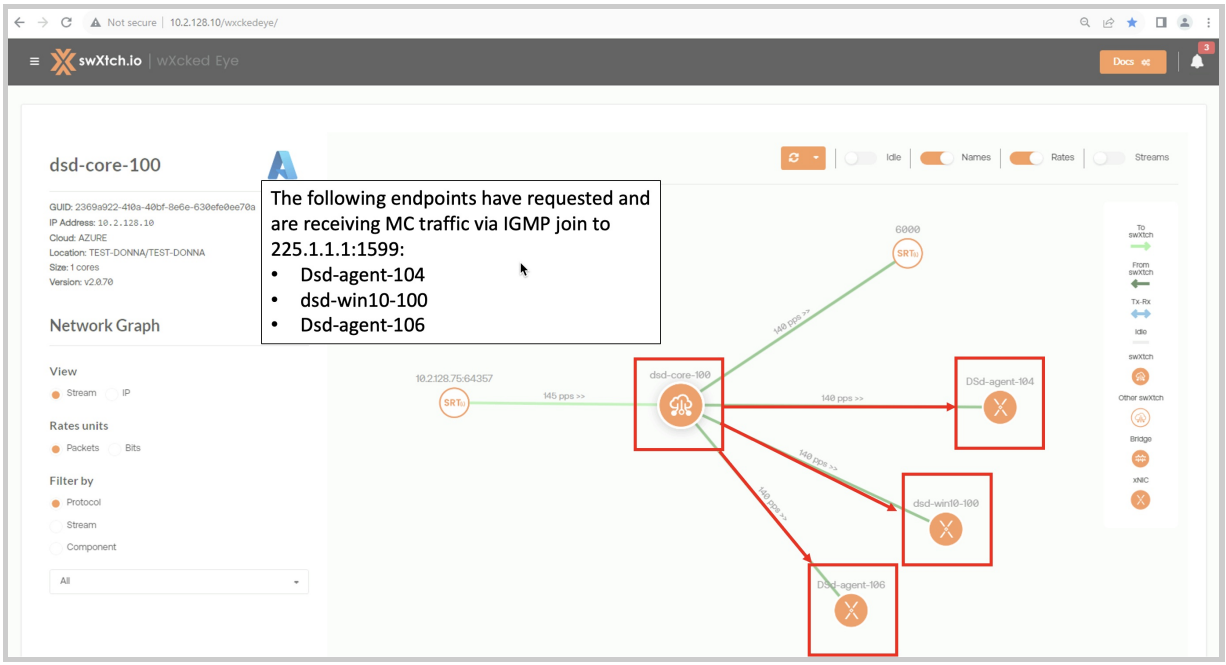
For Ingress, think about the cloudSwXtch **Ingesting** a stream. The user has set up an SRT Listener Ingress using the **SRT-Listener -> Multicast 225.1.1.1:1599** stream with **Listener Port 1599**. That means that an endpoint will have to target port 1599 to send SRT traffic to the cloudSwXtch. Since it is ingress, the cloudSwXtch will automatically convert the SRT stream it receives into multicast.

Using the Topology in wXcked Eye, you can see how SRT Listener Ingress is set up in relation to the cloudSwXtch below.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



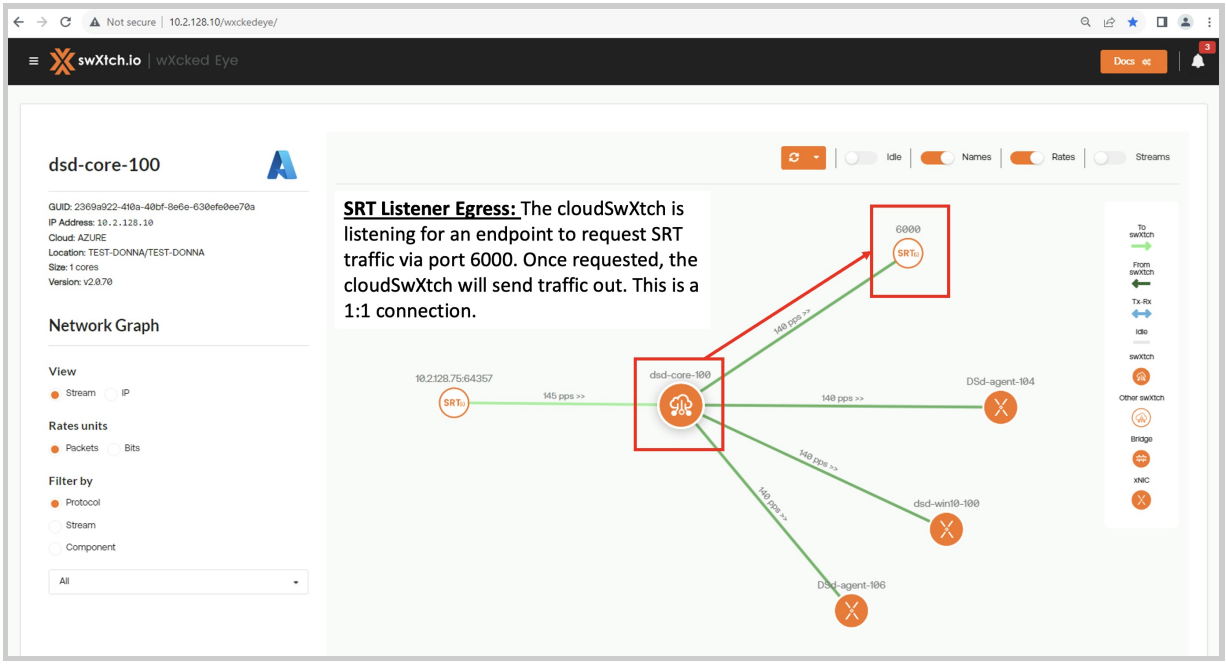
The endpoints highlighted will then request and receive multicast traffic via IGMP join to 225.1.1.1:1599.



SRT Listener Egress

For Egress, imagine the stream **EXITING** the cloudSwXtch (->). In this example, the user has set up SRT Listener Egress using the **SRT-Listener -> Multicast 225.1.1.1:1599** stream and opening Listener Port 6000. That means that an endpoint will have to target port 6000 and let the cloudSwXtch know that it would like to receive SRT traffic. Note that **this is a 1:1 connection**, meaning only one SRT endpoint can use the listener port. In the example below, an endpoint has requested the SRT traffic and the cloudSwXtch is sending it out 140pps via port 6000.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

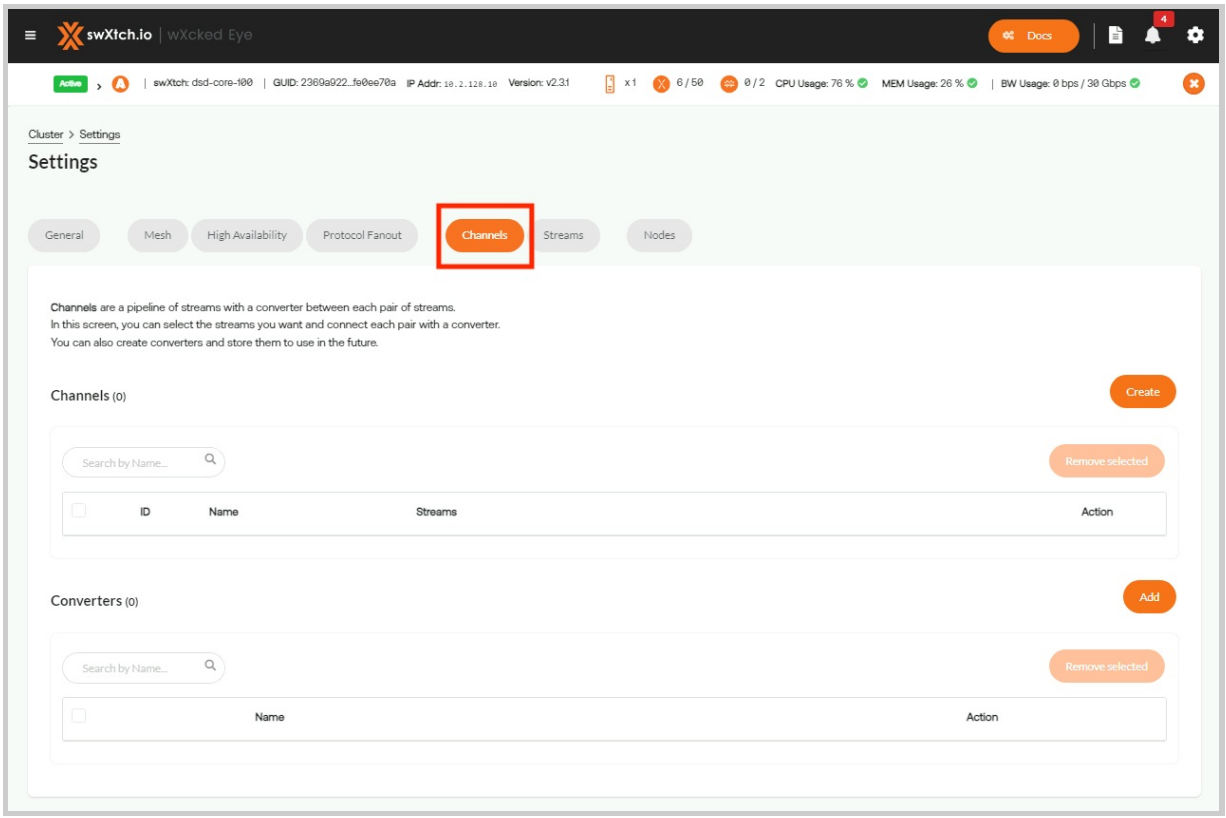


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Channels

WHAT TO EXPECT

The Channels tab on the Settings page allows users to create and remove channels from wXcked Eye. In addition, users can add and remove converters, which connect streams together to form a Channel.



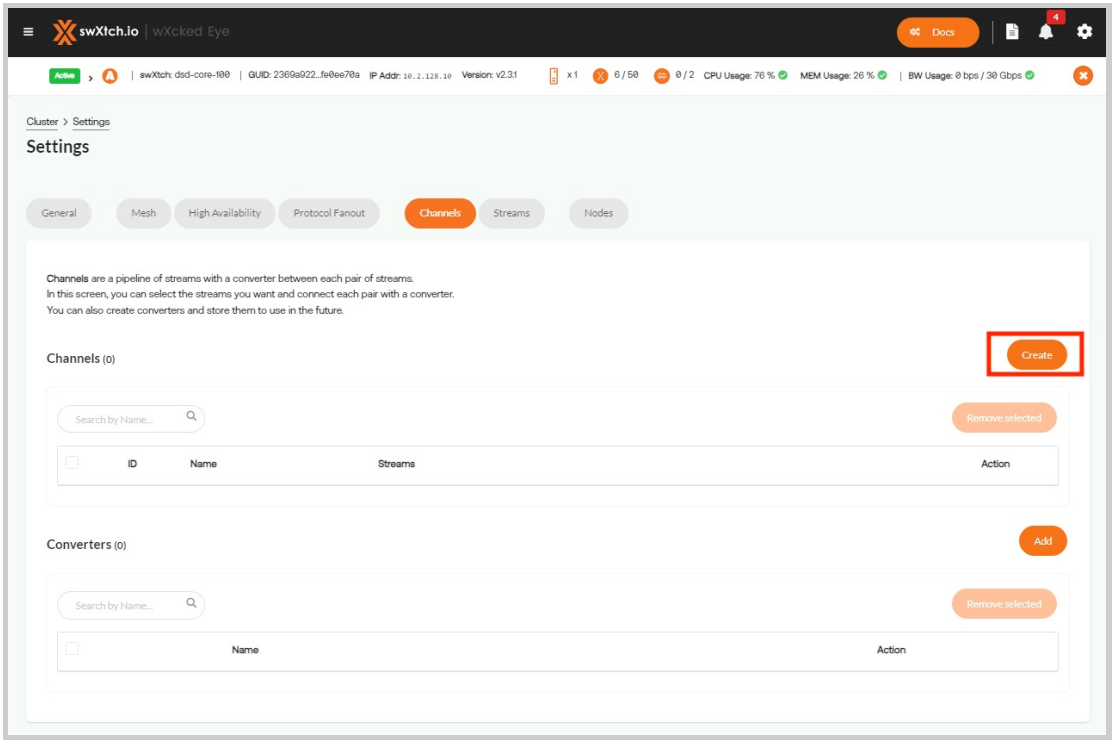
Channels

Channels are a pipeline of streams with a converter between each of them. On this page, a user will be able to Create Channels from a list of available streams and note the converters that connect them.

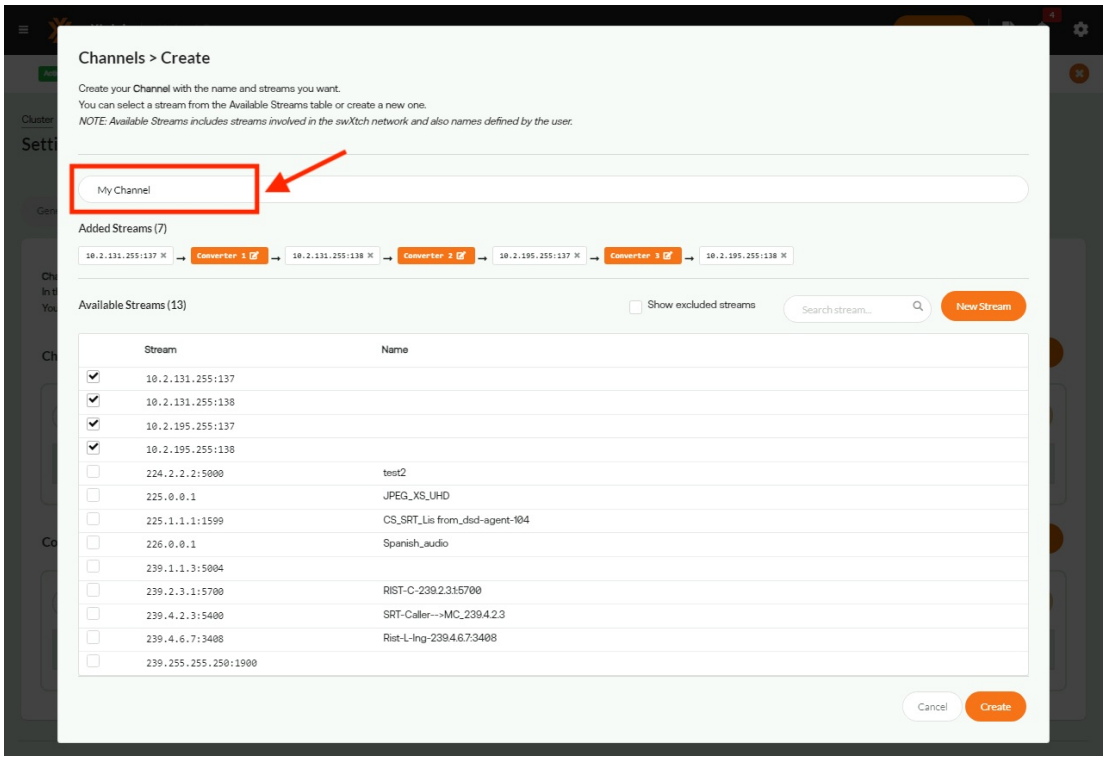
Creating Channels

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. Click **Create** above the Channels section. A new window will open.

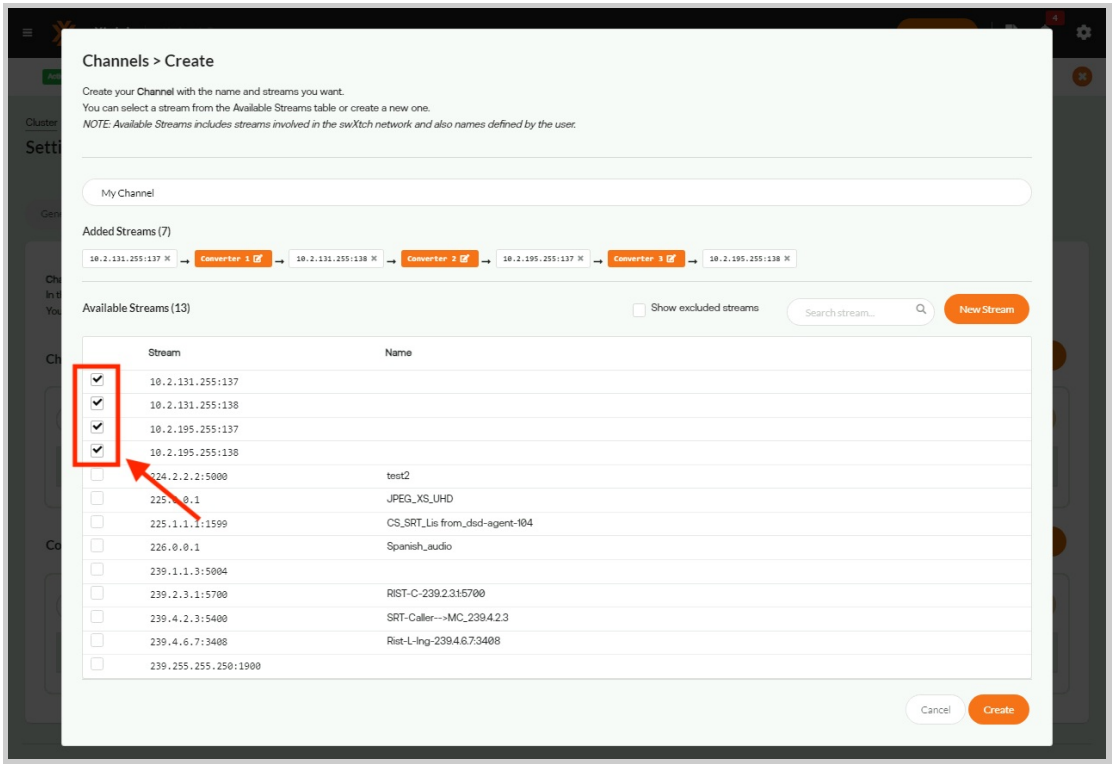


2. Assign a name to your new Channel.



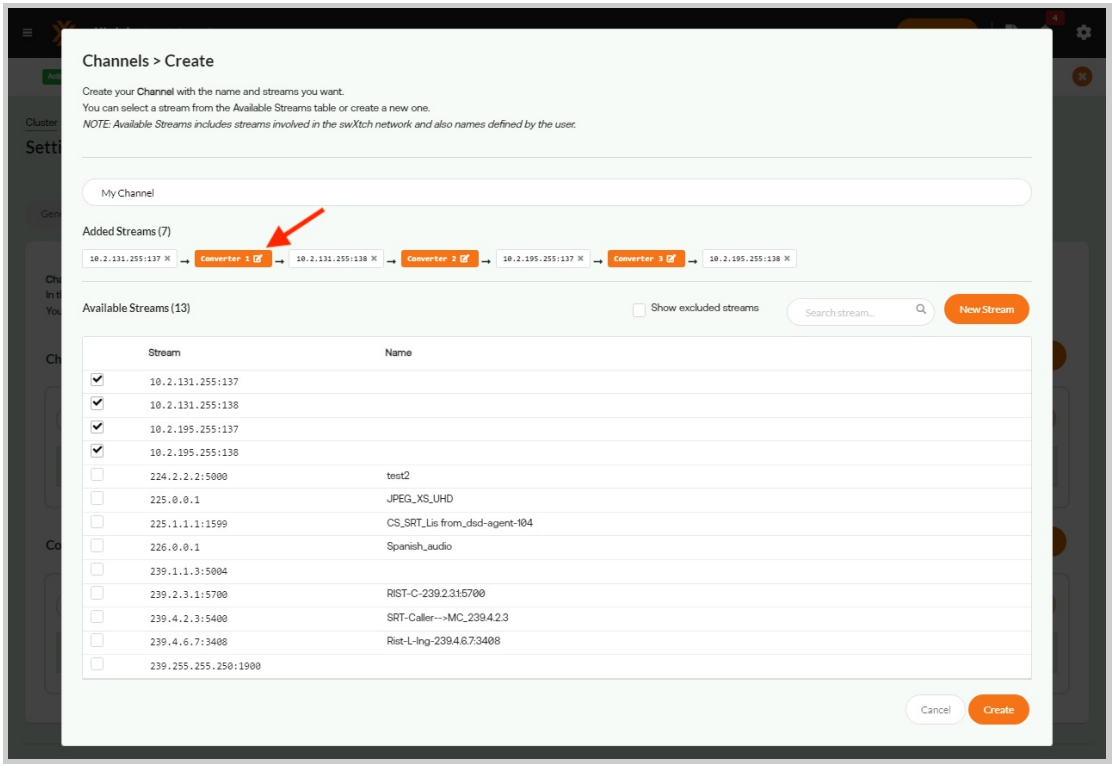
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. Select the streams that you wish to add to the Channel configuration from the Available streams list.



a. Alternatively, select New Stream to add a new stream to the list. This will open a new window where you can name and assign a color to a Stream IP.

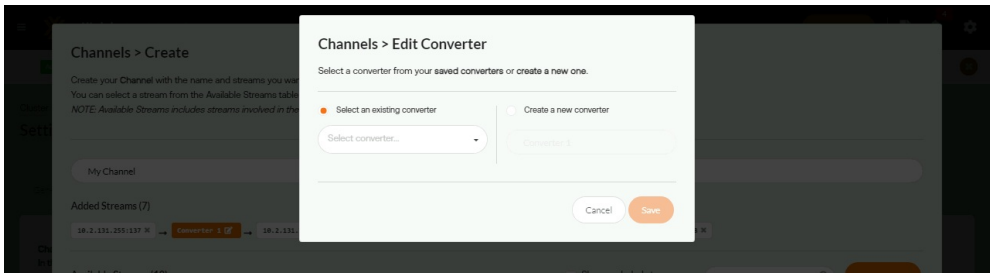
4. Edit the converter that links each stream by selecting the notepad by the placeholder.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

5. **Select an existing converter** from the dropdown menu.

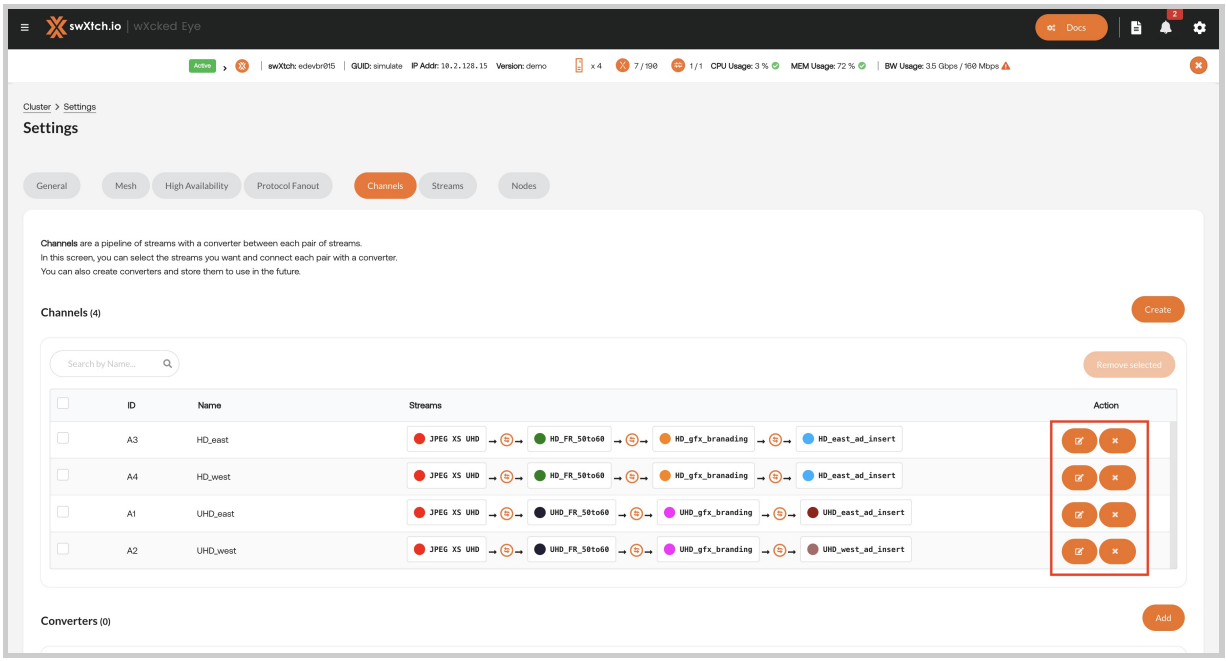
    a. Alternatively, select **Create a new converter** and name it.



    b. **Please note**: wXcked Eye does not create these converters. This feature only allows users to name the existing converters in their network that will connect these streams.

6. Click **Create**. A new Channel will appear.

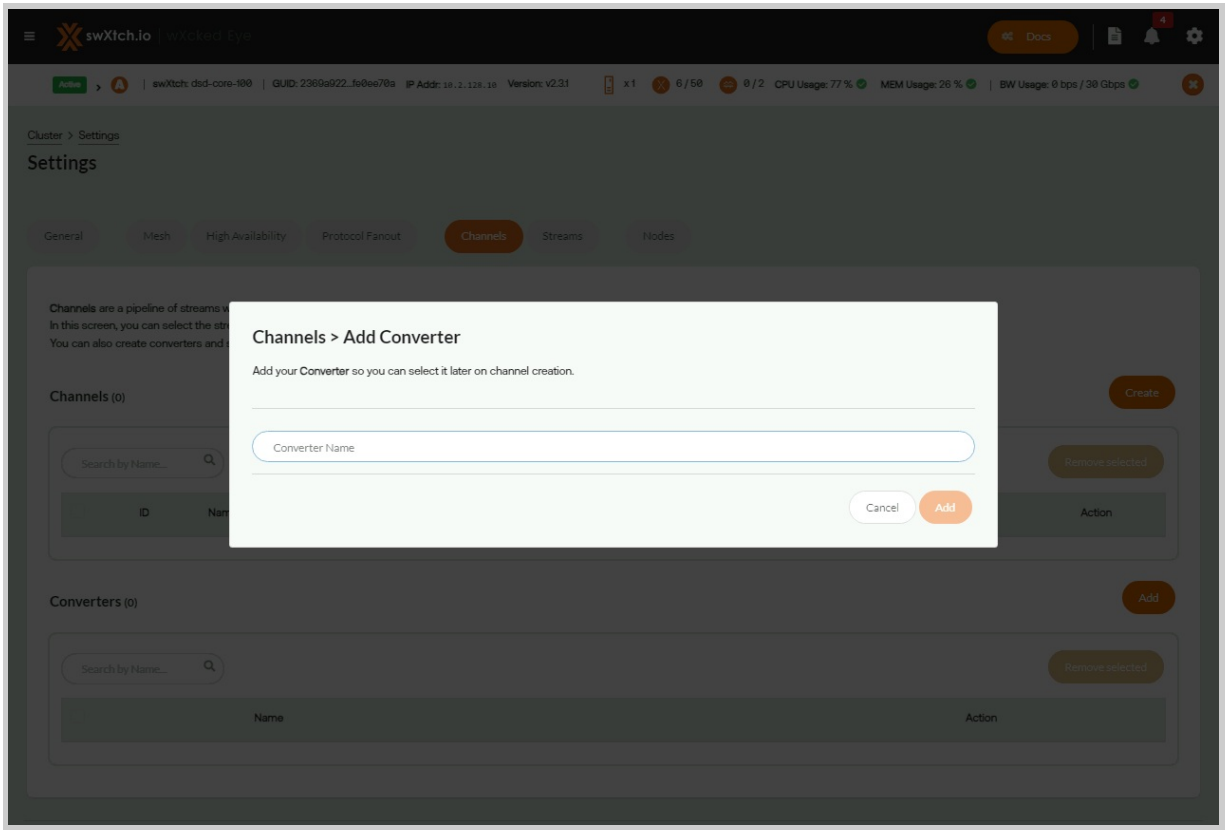
wXcked Eye will automatically assign an ID to the Channel. These channel configurations will carry throughout the UI. Users can edit or remove the channel using the action buttons on the right hand side.



Adding Converters

Users can add a converter by clicking the Add button in the Converter section. A new window will open where users can name existing converters in their network. These can be a range of different transformations, such as Tachyon LIVE. **Please note:** wXcked Eye does not create converters. This feature only allows users to name a placeholder to connect streams visually in the UI.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



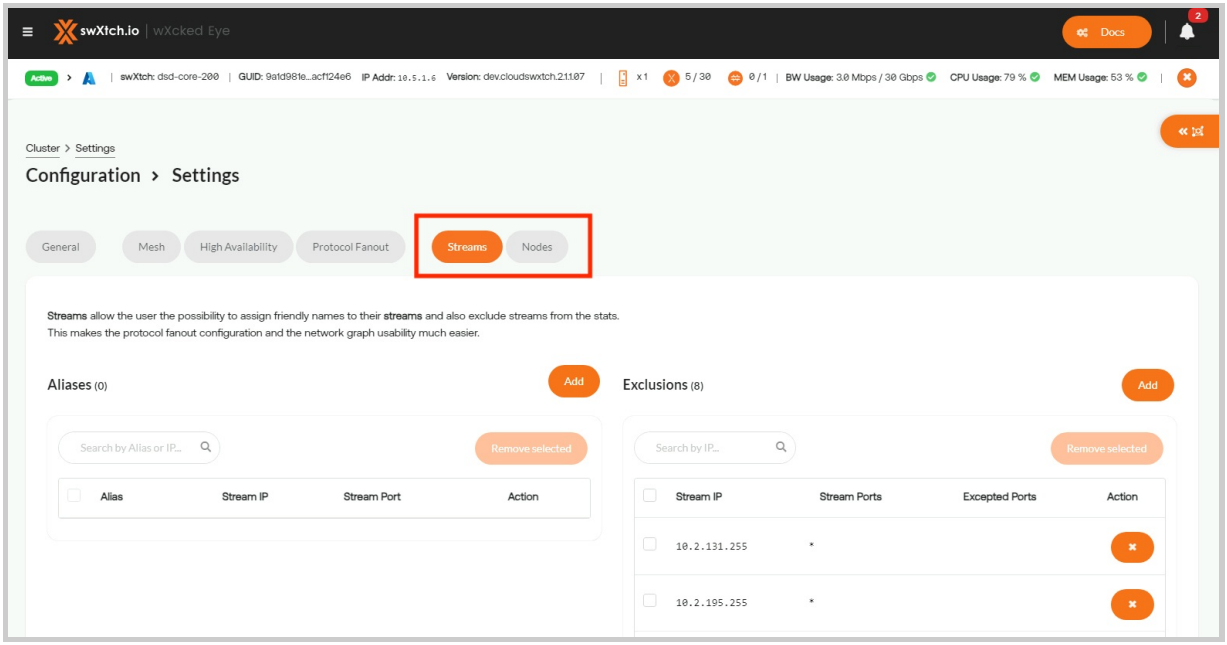
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Stream and Nodes

WHAT TO EXPECT

The Stream and Nodes tabs on the Settings page allows users to assign friendly names to their streams and nodes. In addition to streamlining protocol conversion and fanout configuration, assigning Aliases to your Streams and Nodes also makes the network graph easier to use.

In this section, users will learn how to add/edit streams and nodes in wXcked Eye. Additionally, they will learn how to exclude streams.



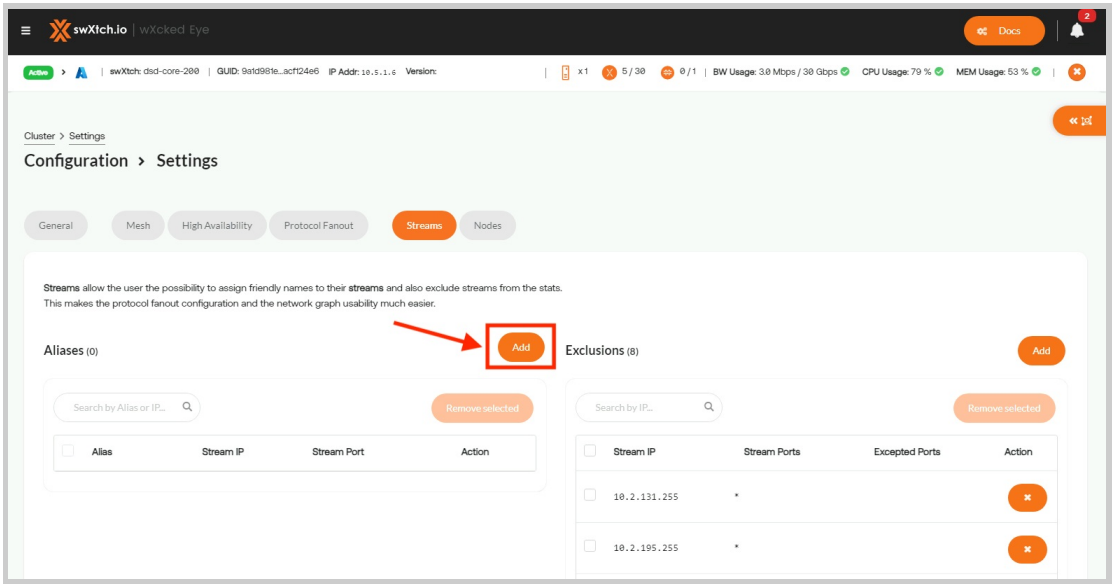
Streams

The Streams tab allows users to name their Multicast IP addresses. This creates a shortcut for users in dropdown menus throughout the Protocol Fanout tab, allowing them to easily differentiate between multiple streams.

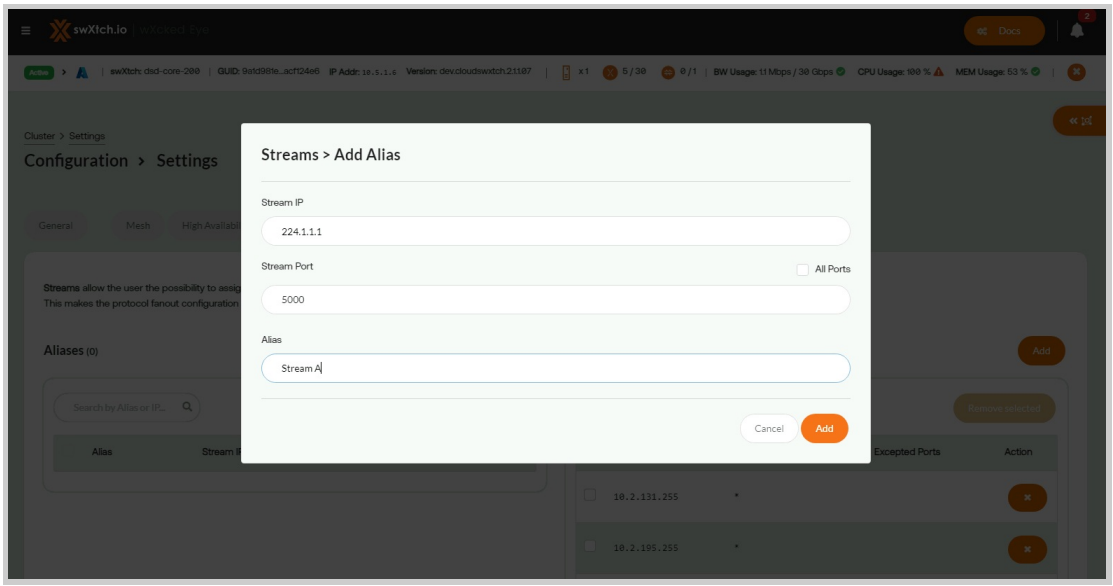
To add a new stream:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. Select **Add Alias** in the top right corner of the **Streams** panel. A pop-up will open.



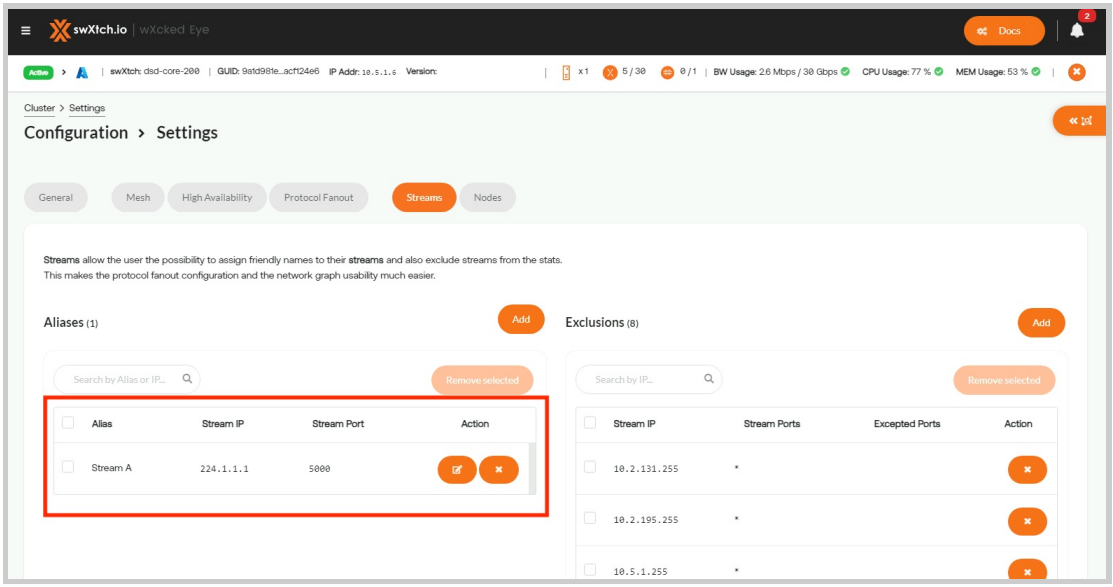
2. Enter the multicast **Stream IP, Stream Port** and **a user-friendly name under Alias**. In the example, the user assigns the Alias, **Stream A**. In addition to setting a single Stream Port, you may also open '**All Ports**' by selecting the checkbox.



3. Click **Add**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. A **new stream** will be added to the list.



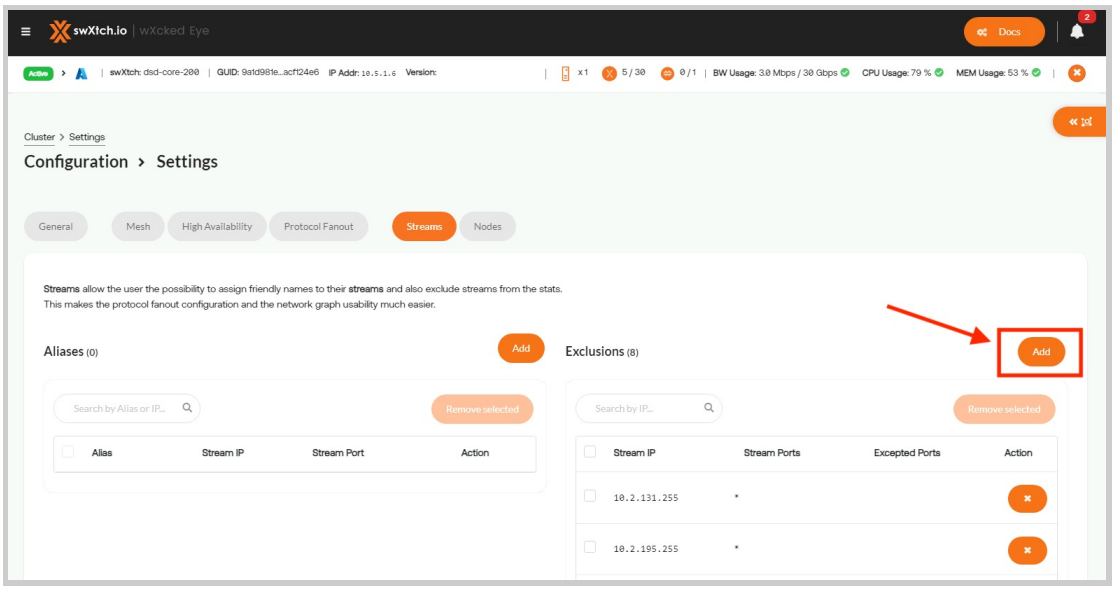
Users can edit the name of the stream or remove it from the list by clicking either buttons under the **Action** column.

Please note: The edit feature does not let you change the Group IP or Group Port. **To do this**, delete the stream and re-add.

Exclusions

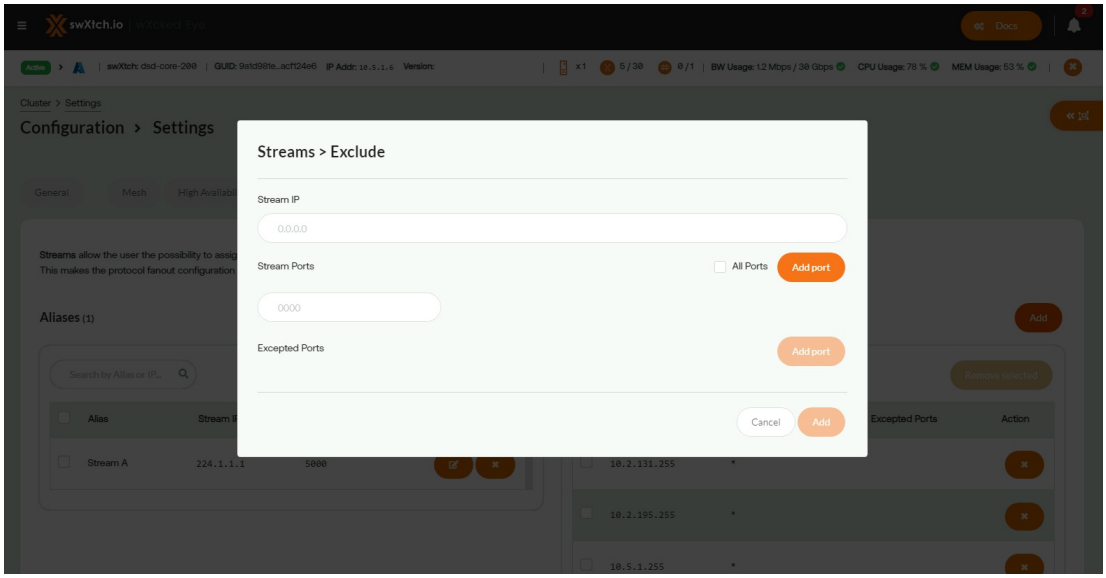
In addition to adding a stream, the Streams tab also includes the functionality to exclude streams from all monitoring related activities in wXcked Eye and swXtch-top. This is especially beneficial for users who use external software to generate additional streams that don't want this included when monitoring packet and data flow through the cloudSwXtch. To do this:

1. Click **Add** in **Exclusions**.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Enter the **Stream IP** and the **Stream Port** you would like to exclude. In addition to setting a single Stream Port, you may also open **All Ports** by selecting the checkbox.



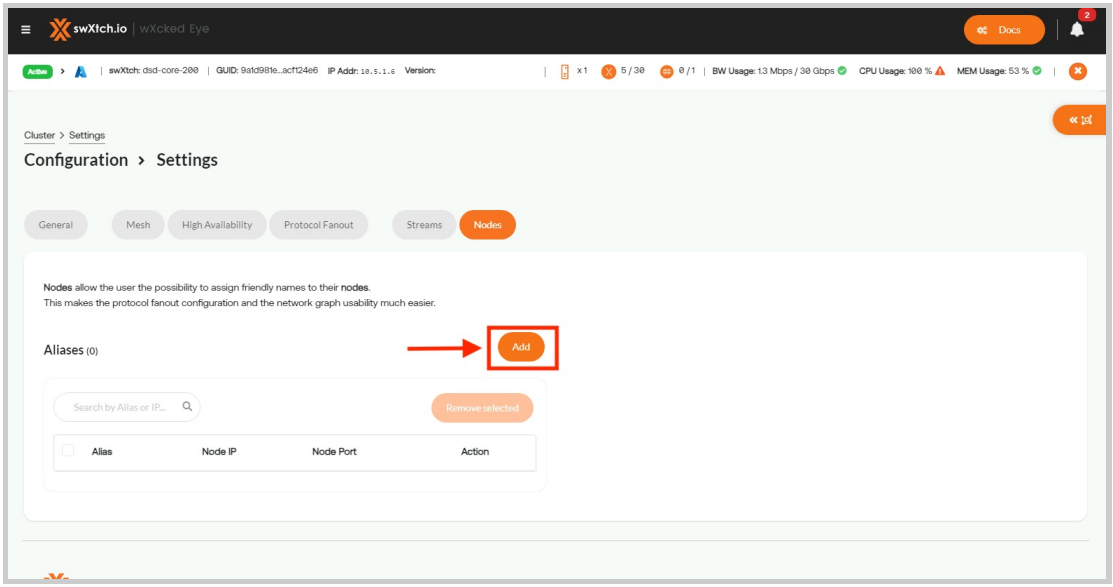
3. Click **Add** when you have made all your edits.
4. A new Stream will be added to your **Exclude Stream** list. **It will no longer appear in wXcked Eye and swXtch-top monitoring**.

Nodes

Similar to the Streams tab, users can set up aliases for their Nodes, avoiding the need to re-write the node IP and port during UDP, SRT or RIST mapping.

To add a new node:

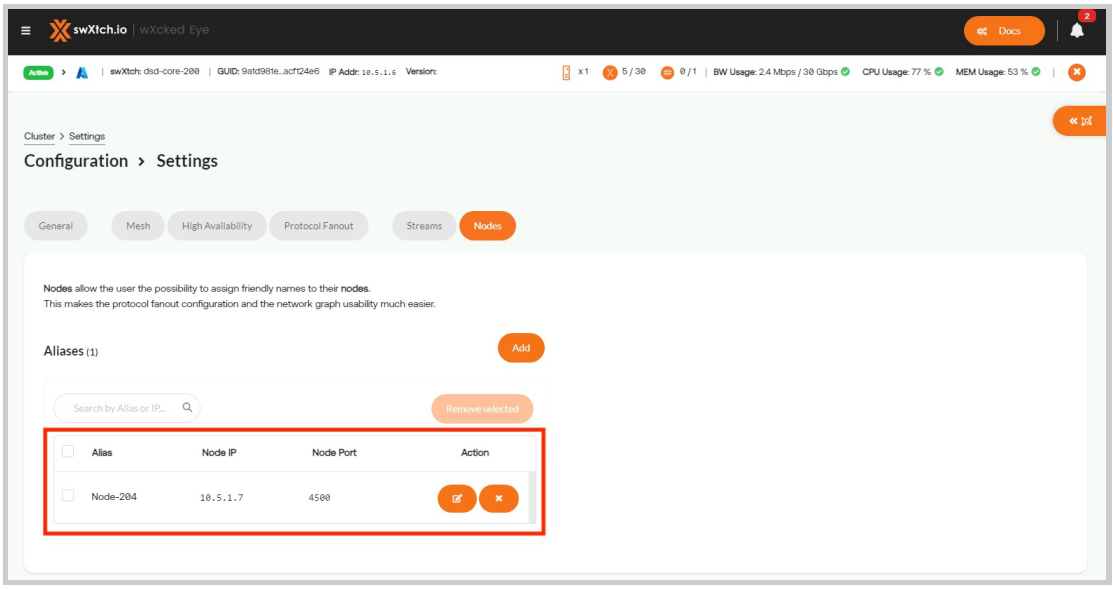
1. Click on the **Add** button in the top right corner of the **Nodes** panel. A new window will open.



2. Enter the **Node IP address, Node Port** and a **user-friendly name**. In this example, the user sets the Alias as Node-204.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. Click **Add** to confirm.
4. A **new node** will be added to the list.

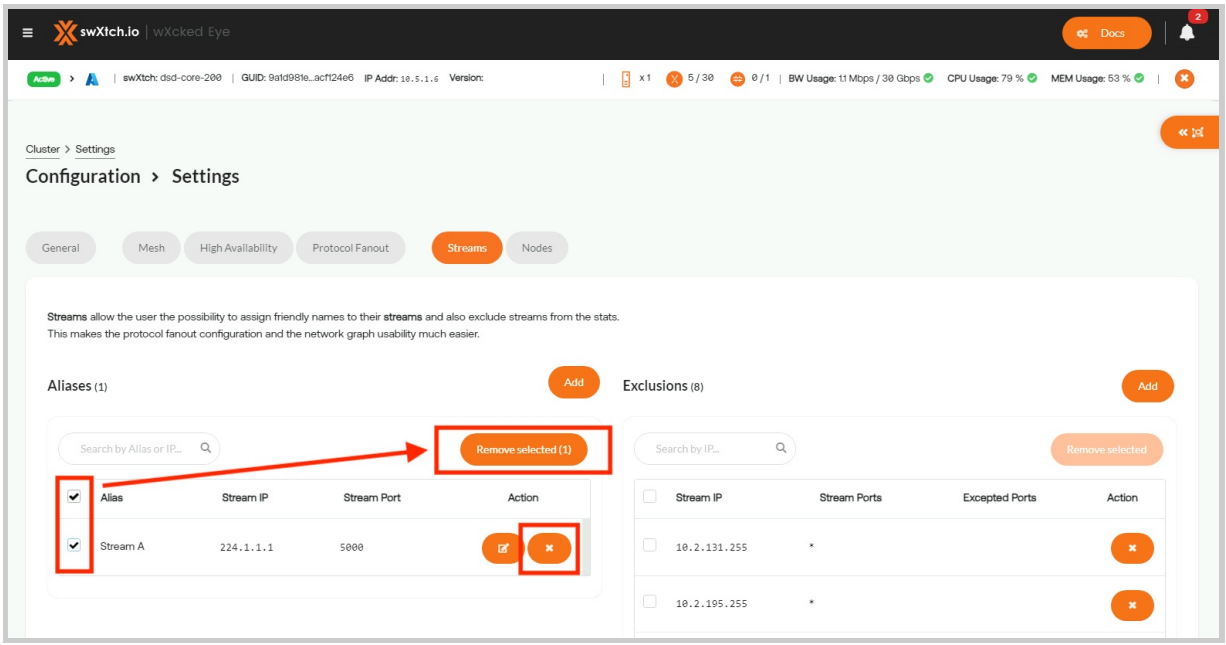


Users can edit the name of the node or remove the name by clicking the buttons under the **Action** column.

Please note: The edit feature does not let you change the Node IP or Node Port. To do this, delete the node and re-add.

Removing Aliases for Streams, Nodes and Exclusions

To remove an alias for a stream, node and/or exclusion, users can either select the "x" button next to each entry individually or do a batch select and click remove selected.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

High Availability

WHAT TO EXPECT

In this article, users will learn how to configure high availability for both the cloudSwXtch and the xNIC.

- **For the cloudSwXtch**, users will learn about the specific commands they can use to configure high availability. Alternatively, users can also configure the cloudSwXtch for high availability via wXcked Eye. **This is the preferred method**. For more information, see High Availability with wXcked Eye. cloudSwXtch configuration is the same for both xNIC and xNIC Classic.
- **For xNIC**, high availability is preconfigured after cloudSwXtch configuration. If Multiple Multicast IP addresses are required, those must be configured on each xNIC.

Configuring cloudSwXtch for High Availability

For High Availability to work, the cloudSwXtch must be configured. This allows for the system to know the paths through user naming and ultimately enables HA views in swXtch-top. In this section, we will review the various HA commands available to the user for a successful high availability configuration.

Users can also configure High Availability for the cloudSwXtch in wXcked Eye. This is the preferred method. For more information, see High Availability with wXcked Eye.

HA Help

To get a list of the HA commands, use -h or --help as shown below.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
PS C:\Users\testadmin> swx ha -h
High Availability cluster management tool (create, show, and destroy
the HA cluster)

Usage:
  swx ha [command]

Available Commands:
  create        Create or Update the HA cluster of swxtches using a
config file
  destroy       Destroy the HA cluster
  show          Show information about the HA cluster

Flags:
  -h, --help                          help for ha
  -s, --service-host-address string   Host swxtch address in the
form <host>[:port]
  -d, --show-error                    show-error - display
additional information for error messages.

Use "swx ha [command] --help" for more information about a command.
```

NOTE

The default port in which the cloudSwXtch listens for these swx configuration commands is port 80. You can safely omit the port in the "-s" parameter since 80 will be used. Do not use port 10802 (the one used in the config file), as it is intended for xNIC communications only. It will not work for swx commands.

HA Create

To create or update an HA cluster, a **HAconfig.json** file must exist. Note that the IP is for the control plane. The following shows the format:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
{
    "name": "MY High Availability",
    "paths": [
        {
            "name": "Path 1",
            "swxtches": [
                "10.2.128.10"
            ]
        },
        {
            "name": "Path 2",
            "swxtches": [
                "10.5.1.6"
            ]
        }
    ]
}
```

If there are multiple cloudSwXtches in a path then the last cloudSwXtch added will have the IP address in the configuration. The rest of the cloudSwXtches do not need to be listed.

The -h and --help commands will show the syntax for the "swx ha create" command.

Bash | Copy

```
PS C:\Users\testadmin> swx ha create -h
Create or Update the HA cluster of swxtches using a config file

Usage:
  swx ha create [flags]

Flags:
  -h, --help           help for create
  -i, --input string   JSON input file

Global Flags:
  -s, --service-host-address string   Host swxtch address in the
form <host>[:port]
  -d, --show-error                    show-error - display
additional information for error messages.
```

Below is the command:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
swx ha create -i <path_to_config> -s <cloudSwXtch_IP>
```

HA Destroy

To remove a cloudSwXtch from the High Availability flow, the "ha destroy" command can be used. The -h and --help commands will show the syntax for the "swx ha destroy" command.

Bash Copy

```
PS C:\Users\testadmin> swx ha destroy -h
Destroy the HA cluster

Usage:
  swx ha destroy [flags]

Flags:
  -h, --help   help for destroy

Global Flags:
  -s, --service-host-address string   Host swxtch address in the
form <host>[:port]
  -d, --show-error                    show-error - display
additional information for error messages.
```

Below is the command to leave:

Bash Copy

```
PS C:\Users\testadmin> swx ha destroy -s <swxtch name or control
data ip of a cloudSwXtch in the HA configuration>
```

Example:

Bash Copy

```
PS C:\Users\testadmin> swx ha destroy -s cloudswxtch101
Validating cluster deletion.
Successfully deleted the cluster.
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Removing a cloudSwXtch from an HA configuration

To remove a cloudSwXtch from a user's HA configuration, they will need to delete and recreate their HA cluster without that cloudSwXtch.

HA Show

To get a list of cloudSwXtches part of the HA flow, the "ha show" command can be used. The -h and --help commands will show the syntax for the "swx ha show" command.

Bash Copy

```
PS C:\Users\testadmin> swx ha show -h
Show information about the HA cluster

Usage:
  swx ha show [flags]

Flags:
  -h, --help   help for show

Global Flags:
  -s, --service-host-address string   Host swxtch address in the
form <host>[:port]
  -d, --show-error                    show-error - display
additional information for error messages.
```

Below is the command to list:

Bash Copy

```
swx ha show -s <swxtch name or control data IP of a cloudSwXtch in
the HA configuration>
```

Example below:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
PS C:\Users\testadmin> swx ha show -s  cloudswxtch101
{
    "clusterConfig": {
        "uid": "D20E820C-91DE-A571-4C7C-B60C1695973D",
        "name": "dsd-100-200-HA",
        "paths": [
            {
                "name": "Path1",
                "swxtches": [
                    "10.2.128.10"
                ]
            },
            {
                "name": "Path2",
                "swxtches": [
                    "10.5.1.6"
                ]
            }
        ]
    }
}
```

swxtch-top has options for High Availability. For more information, see the swxtch-top article.

Running Bridge 1 for High Availability

WHAT TO EXPECT

In order for a bridge to send multiple streams, it must have an separate instance running for each cloudSwXtch within the HA configuration. In the example above, the bridge was sending data to two cloudSwXtch, resulting in two separate cloudSwXtch Bridge 1 instances.

In this section, you will find the commands to run the Bridge 1 as separate instances.

Below are the swxtch-bridge arguments. **Note**: You can also find this list of arguments using **swxtch-bridge -h**

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
agent-101:~$ swxtch-bridge -h
Usage: swxtch-bridge [options]
Optional arguments:
-h --help                                shows help message and exits
[default: false]
-v --version                             prints version information
and exits [default: false]
-i --input                               input url: [udp|tcp|-]
(://<ip>:<port> [default: "-"]
-o --output                              input url: [udp|tcp|-]
(://<ip>:<port> [default: "-"]
-c --core                                start core [default: 1]
-n --core-count                          core count [default: 4]
-r --override-multicast-sender-ip        --override-multicast-sender-
ip xxx.xxx.xxx.xxx: override multicast sender ip to make it routable
[default: "0.0.0.0"]
-p --path-id                             path start identification
for HA flow [default: 0]
-f --first-leg                           first leg from repl, we need
to offset incoming packet for length field in IexHeader [default:
false]
-l --last-leg                            last leg to repl, we need to
stripe away the Length field in IexHeader [default: false]
```

The diagram above sends paths from two instances of the same cloudSwXtch Bridge 1 to two different cloudSwXtches. Below are example calls a user would use for each of the bridge instances.

Bridge Instance 1

Bash Copy

```
swxtch-bridge --input
multicast://239.1.1.4:172.30.0.4:8804,multicast://239.1.1.5:172.30.0
.4:8804,multicast://239.5.69.2:172.30.0.4:10000 --output
udp://10.2.192.23:9999 --last-leg --path-id 0 -c 0

OUTPUT:
set cpu: 0
set cpu: 1
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bridge Instance 2

```
swxtch-bridge --input
multicast://239.1.1.4:172.30.0.4:8804,multicast://239.1.1.5:172.30.0
.4:8804,multicast://239.5.69.2:172.30.0.4:10000 --output
udp://10.5.2.6:9999 --last-leg --path-id 1 -c 2

OUTPUT:
set cpu: 2
set cpu: 3
```

- **--input**: Multiple multicast groups can be entered as shown in both examples above.
- **--output**: This is the cloudSwXtch consumer's data NIC.
- **--path-id**: Here, a user will designate a path number, starting with 0 as Path 1. (Increment by 1 for each path. For example, 1 for Path 2, 2 for Path 3, etc.)

For all additional arguments, please see the **swxtch-bridge -h** list above.

Configuring xNIC for High Availability

After a user sets up their cloudSwXtch for High Availability, the xNIC will automatically configure itself to receive and/or send HA traffic for a single multicast group. Users can confirm high availability has been configured by viewing the JSON file in the VM where their xNIC resides.

Where to Find the xNIC JSON File

To configure the xNIC for high availability streams, the xnic.json file needs to be updated. See where to locate them below:

For Linux:
The file can be found in **/var/opt/swxtch/xnic.json**. Currently, only V1 is supported for Linux.
To edit the file, one option is to use nano as shown below:

```
sudo nano /var/opt/swxtch/xnic.json
```

For Windows:
The file can be found in **C:\Program Files\SwXtch.io\Swxtch-xNIC\xnic.json**

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Single Multicast Group

An example of the xnic.json file is shown below. Note the **ha section** has been added.

Bash | Copy

```
{
    "swxtch": "10.2.128.10",
    "controlInterface": "Ethernet 2",
    "dataInterface": "Ethernet",
    "dataPort": 9999,
    "xnicType": 2,
    "dataPlaneSpecs": {
        "verbosity": 0,
        "virtualInterface": {
            "name": "swxtch-tun0",
            "ip": "172.30.0.0",
            "subnet": "255.255.0.0",
            "mtu": 4096
        },
        "bpfPrograms": [
        ]
    },
    "ha": {
        "maxTimeToBufferPacketsMs": 50,
        "bufferSizeInPackets": 131072,
        "protocol": "swxtch"
    },
    "statsReportWait": 60
}
```

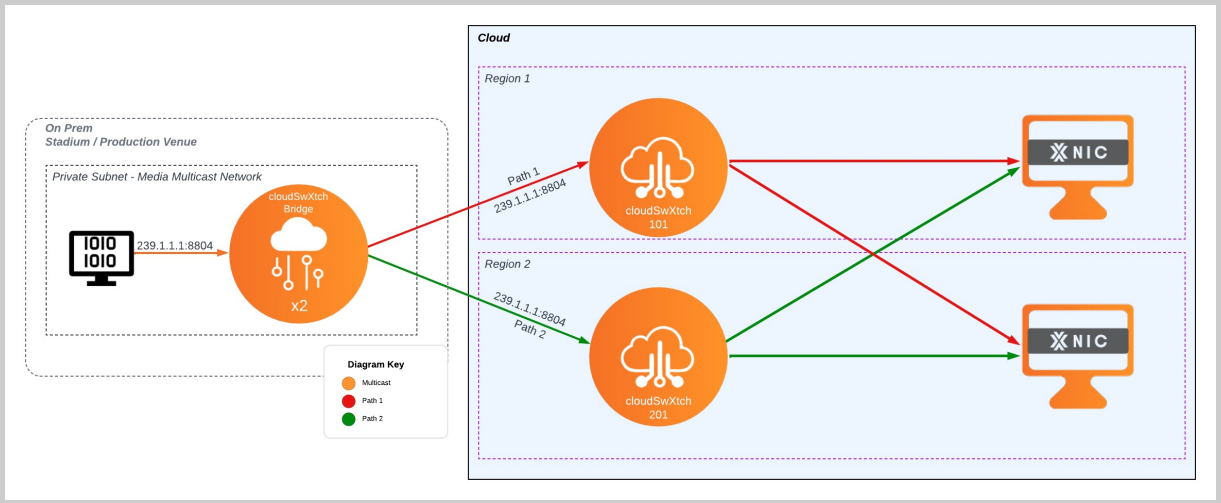
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

ha Section Explained

The ha section exposes variables that can alter the behavior of the hitless switching code. The values for MaxTimeToBufferPackets_ms and BufferSizeInPackets in the example are good, suggested values; however, they can be tweaked to meet desired high availability requirements.

- **MaxTimeToBufferPackets_ms** - how long to buffer packets before declaring it as lost.
- **BufferSizeInPackets**- the max number of packets that can be buffered.
- **Protocol**- how to parse the packet. The available options are **swxtch** or **rtp**.
    - ***swxtch*** = This can be used when the xNIC is duplicating or deduplicating the multicast. The xNIC will reconstruct based on the sequence count inside the cloudSwXtch packet header.
    - ***rtp*** = This should be used when processing RTP packets sent from a non-xNIC source. The xNIC will reconstruct based on the RTP timestamp information for Real-time Transport Protocol.

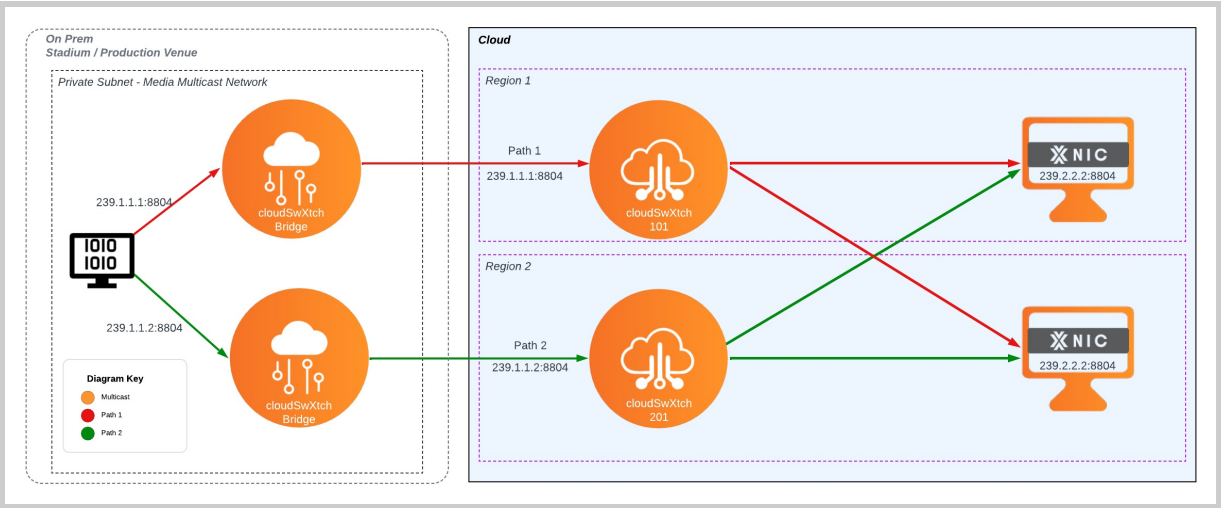
Below is an example of what a single multicast group configuration might look like. In this example, a user is sending the same multicast traffic (239.1.1.1:8804) via two paths with the xNIC consuming both and deduplicating at the end point.



Multiple Multicast for xNIC

If a user wants to set up for Multiple Multicast groups, they will need to manually configure the **xnic.json** file. Below is an example of what a multiple multicast group configuration might look like:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



In this example, you have two paths with the same multicast traffic with different IP addresses. Path 1 is 239.1.1.1 while Path 2 is 239.1.1.2. The application at the end point is listening to 239.2.2.2, which is grouping together Path 1 and Path 2. The xNIC at the end point is tasked with deduplication.

A sample **xnic.json** file of the diagram is shown below with a "streamSpecs" section added.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
{
    "swxtch": "10.2.128.10",
    "controlInterface": "Ethernet 2",
    "dataInterface": "Ethernet",
    "dataPort": 9999,
    "xnicType": 2,
    "dataPlaneSpecs": {
        "verbosity": 0,
        "virtualInterface": {
            "name": "swxtch-tun0",
            "ip": "172.30.0.0",
            "subnet": "255.255.0.0",
            "mtu": 4096
        },
        "bpfPrograms": [
        ]
    },
    "ha": {
        "maxTimeToBufferPacketsMs": 50,
        "bufferSizeInPackets": 131072,
        "protocol": "rtp"
    },
    "streamSpecs": {
        "mmcProducerEnable": false,
        "multipleMulticastGroups": [
            {
                "parent": "239.2.2.2:8804",
                "children": [
                    "239.1.1.1:8804",
                    "239.1.1.2:8804"
                ]
            }
        ]
    },
    "statsReportWait": 60
}
```

Here, the user is grouping together 2 multicast IPs (239.1.1.1 and 239.1.1.2) and assigning it a **"Parent"** IP address (239.2.2.2). The application at the endpoint is listening for 239.2.2.2, which the xNIC will deduplicate into a stream from 239.1.1.1 and 239.1.1.2. This was illustrated in the diagram above.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Please note: At this time, the ports for the Parent and the Children must be the same.

How to update xnic.json file for Multiple Multicast Groups

The user will have to make the following changes to their **xnic.json** file found in the single multicast group configuration to match the example above. These alterations to the xnic.json file should happen after Configuring the cloudSwXtch for High Availability.

1. Change the protocol under ha from "**swxtch**" to "**rtp**" including the quotation marks.

2. **For each multicast group**, add the following "**streamSpecs**" section as shown below with your **parent** and **child** groups listed. **Note**: A user can enter multiple multicast groups.

Bash Copy

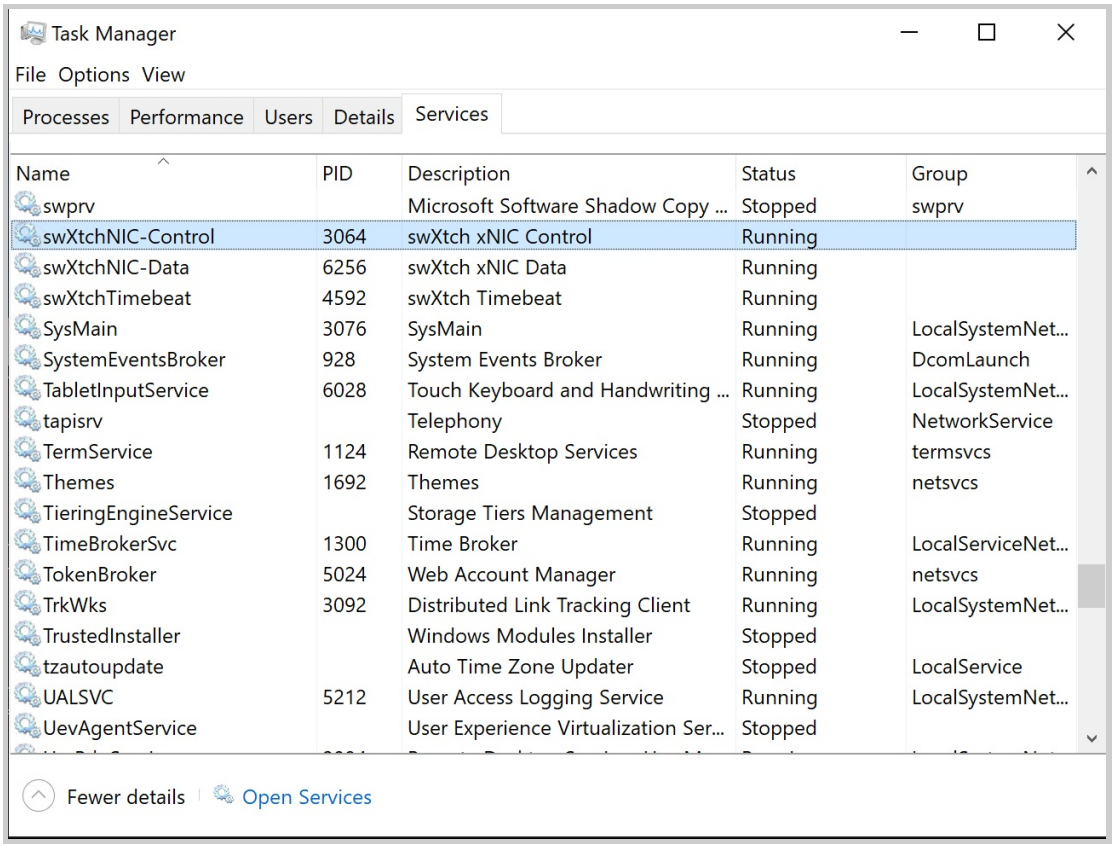
```
},
    "streamSpecs": {
        "mmcProducerEnable": false,
        "multipleMulticastGroups": [
            {
                "parent": "239.2.2.2:8804",
                "children": [
                    "239.1.1.1:8804",
                    "239.1.1.2:8804"
                ]
            },
            {
                "parent": "239.3.3.3:8804",
                "children": [
                    "239.1.1.3:8804",
                    "239.1.1.4:8804"
                ]
            }
        ]
    },
```

3. Save the **xnic.json** file.

4. Restart the **swXtch-NIC Control** service.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **For Windows**, go to the **Task Manager** and under the **Services** tab, select and restart swXtchNIC-Control.



- **For Linux**, use the following command:

Bash Copy

```
sudo service swxtch-xnic-control restart
```

Configuring cloudSwXtch Bridge Type 2 for High Availability

After a user sets up their cloudSwXtch for High Availability, the cloudSwXtch Bridge will automatically configure itself to receive and/or send HA traffic for a single multicast group. Users can confirm high availability has been configured by viewing the swxtch-bridge.json file in the VM where their cloudSwXtch Bridge resides.

Reminder: A cloudSwXtch Bridge can only exist on a Linux machine.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Where to Find the Bridge JSON File

To configure the cloudSwXtch Bridge for high availability streams, the swxtch-bridge.json file needs to be updated. See where to locate them below:

Linux:
The file can be found in /var/opt/swxtch/swxtch-bridge.json. To edit the file, use nano as shown below:

Bash | Copy

```
sudo nano /var/opt/swxtch/swxtch-bridge.json
```

Single Multicast Group

An example of the swxtch-bridge.json file is shown below. Note the ha section has been added. While it may look similar to the xnic.json file, there are a number of differences. One noteworthy addition is the "producer" and "consumer" fields in the ha section.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
{
    "swxtch": "10.2.128.10",
    "controlInterface": "Ethernet 2",
    "dataInterface": "Ethernet",
    "dataPort": 9999,
    "xnicType": 2,
    "dataPlaneSpecs": {
        "verbosity": 0,
        "virtualInterface": {
            "name": "swxtch-tun0",
            "ip": "172.30.0.0",
            "subnet": "255.255.0.0",
            "mtu": 4096
    },
    "ha": {
        "producer" : false,
        "consumer": false,
        "maxTimeToBufferPacketsMs": 50,
        "bufferSizeInPackets": 131072,
        "protocol": "swxtch"
    },
    "statsReportWait": 60
}
```

It is important to imagine configuration from the perspective of the cloudSwXtch Bridge. At default, both the producer and consumer is set to false. Switching either to true will change the direction of the high availability.

- **If the producer is set to true**, the cloudSwXtch Bridge is producing duplicate streams taken from on-prem applications and sending to cloudSwXtches in the cloud.
- **If the consumer is set to true**, the cloudSwXtch Bridge is receiving duplicate streams from the cloud. However, unlike the xNIC, it does not do the deduplication and will send both streams to the on-prem applications.

Whether it is producer or consumer, one of them has to be set to true in order to activate high availability via the cloudSwXtch Bridge.

Once a selection has been made, a user m**ust restart the cloudSwXtch Bridge control service** by running the following command:

Bridge Type 2:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash	Copy

```
sudo systemctl restart swxtch-bridge2.service
```

Multiple Multicast for cloudSwXtch Bridge

If a user wants to set up Multiple Multicast groups for cloudSwXtch Bridge, they will need to manually configure the **swxtch-bridge.json** file by changing the protocol under ha to "rtp" and adding the "streamSpecs" section. This process is similar to configuring the xNIC for multiple multicast groups.

How to update swxtch-bridge.json file for Multiple Multicast Groups:

The user will have to make the following changes to their **swxtch-bridge.json** file found in the single multicast group configuration to match the example above. These alterations to the swxtch-bridge.json file should happen after Configuring the cloudSwXtch for High Availability.

1. Change the protocol under ha from "**swxtch**" to "**rtp**" including the quotation marks.

2. **For each multicast group**, add the following "**streamSpecs**" section as shown below with your **parent** and **child** groups listed. **Note**: A user can enter multiple multicast groups.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
},
    "streamSpecs": {
        "mmcProducerEnable": false,
        "multipleMulticastGroups": [
            {
                "parent": "239.2.2.2:8804",
                "children": [
                    "239.1.1.1:8804",
                    "239.1.1.2:8804"
                ]
            },
            {
                "parent": "239.3.3.3:8804",
                "children": [
                    "239.1.1.3:8804",
                    "239.1.1.4:8804"
                ]
            }
        ]
    },
```

3. Save the swxtch-bridge.json file.

4. Restart the cloudSwXtch Bridge control service by running the following command in Linux:

Bridge 2

Bash Copy

```
sudo systemctl restart swxtch-bridge2.service
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Mesh

WHAT TO EXPECT

The following article details the available commands a user can input in order to create, destroy or modify a mesh configuration.

A user can also use the wXcked Eye UI to accomplish the same tasks. To learn more, visit the "**Configure with wXcked Eye**" article under Configuring cloudSwXtch.

Supported Versions:

Mesh commands below is supported in v1.9.16 or higher. For older versions contact support at support@swXtch.io.

Mesh

Mesh configuration with the commands below should only be done on a VM with an active xNIC running on it. Please note that these commands **should not** be done on the cloudSwXtch VM itself.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

None Copy

```
PS C:\Users\testadmin> swx mesh -h
Mesh management tool (create, destroy, members, add switch, remove
switch, print members & routes)

Usage:
  swx mesh [command]

Available Commands:
  add-swxtch    Add swxtch to the mesh
  create        Create the mesh of swxtches using a config file
  destroy       Destroy the mesh
  remove-swxtch Remove swxtch from the mesh
  show          Show information about the mesh

Flags:
  -h, --help                          help for mesh
  -s, --service-host-address string   Host swxtch address in the
form <host>[:port]

Use "swx mesh [command] --help" for more information about a
command.
```

CREATE

This command provides a mechanism to create a mesh using an input configuration file.

The configuration file describes the cloudSwXtches that will participate in the mesh. Each element in the list is the IP address for the cloudSwXtch's control interface.

Command:

`swx mesh create -i <config.json> -s <service-host-address>`

Arguments:

`-i, --input`

`-s, service-host-address of a cloudSwXtch to be included in the mesh.`

Example

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
swx mesh create -i meshconfig.json -s 10.2.128.5

OUTPUT:
Validating mesh..
Mesh succesfully created.
```

Below is an example of a meshconfig.json file:

```
{
  "name": "customer-mesh",
  "switches": ["10.2.128.5", "10.2.162.4"]
}
```

ADD cloudSwXtch to a Mesh

This command adds a cloudSwXtch to an already existing mesh configuration.

Command

`swx mesh add-swxtch -s <service-host-address of a cloudSwXtch in an existing Mesh configuration> -a <swxtch-addr>`

Arguments:

`-s, --service-host-address string Host swxtch address in the form <host>[:port]`

`-a, --swxtch-addr : ip address of the swxtch that is being added to the mesh`

Example

```
swx mesh add-swxtch -s 10.2.128.10 -a 10.1.1.6
Validating that the swxtch was added.
Swxtch successfully added to the mesh.
```

SHOW

This command reports a list of cloudSwXtches participating in the specified mesh. Any cloudSwXtch participating in the mesh is able to provide the current state of the mesh configuration. The query can be issued against any of them.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Command

```
swx mesh show -s <service-host-address for any cloudSwXtch in the Mesh configuration>
```

Arguments:

```
- -s, --service-host-address string Host swxtch address in the form <host>[:port]
```

Example

None | Copy


```
swx mesh show -s 10.2.128.10
{
    "routes": {
        "destinationMap": {
            "10.1.1.6": "10.1.1.6",
            "10.5.1.6": "10.1.1.6"
        }
    },
    "members": [
        "10.2.128.10",
        "10.5.1.6",
        "10.1.1.6"
    ],
    "subscriptions": {
        "groups": {
            "224.0.0.251": {
                "groupAddress": "224.0.0.251",
                "swxtches": {
                    "10.1.1.6": "10.1.1.6",
                    "10.5.1.6": "10.5.1.6"
                }
            },
            "224.0.0.252": {
                "groupAddress": "224.0.0.252",
                "swxtches": {
                    "10.1.1.6": "10.1.1.6",
                    "10.5.1.6": "10.5.1.6"
                }
            },
            "224.0.1.129": {
                "groupAddress": "224.0.1.129",
                "swxtches": {
                    "10.1.1.6": "10.1.1.6",
                    "10.5.1.6": "10.5.1.6"
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
                }
            },
            "239.1.1.1": {
                "groupAddress": "239.1.1.1",
                "swxtches": {
                    "10.5.1.6": "10.5.1.6"
                }
            },
            "239.1.1.2": {
                "groupAddress": "239.1.1.2",
                "swxtches": {
                    "10.1.1.6": "10.1.1.6"
                }
            },
            "239.1.1.3": {
                "groupAddress": "239.1.1.3",
                "swxtches": {
                    "10.1.1.6": "10.1.1.6"
                }
            },
            "239.1.1.4": {
                "groupAddress": "239.1.1.4",
                "swxtches": {
                    "10.1.1.6": "10.1.1.6"
                }
            },
            "239.255.255.250": {
                "groupAddress": "239.255.255.250",
                "swxtches": {
                    "10.1.1.6": "10.1.1.6"
                }
            }
        }
    }
}
```


Remove a cloudSwXtch from a Mesh

This command removes a given cloudSwXtch from the specified mesh.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Command

`swx mesh remove-swxtch -s <host-addr of the cloudSwXtch you wish to remove from the Mesh>`

Arguments:

`-s, --service-host-address string Host swxtch address in the form <host>[:port]`

Example

None | Copy

```
swx mesh remove-swxtch -s 10.1.1.6
Validating that the swxtch was removed.
Swxtch successfully removed from the mesh.
```

Destroy

This command will delete or destroy the entire mesh.

Comand:

`swx mesh destroy -s <host-addr for one of the cloudSwXtches in the Mesh you wish to destroy>`

Arguments:

`-s, --service-host-address string Host swxtch address in the form <host>[:port]`

Example

None | Copy

```
swx mesh destroy -s 10.2.128.10
Validating that the mesh was destroyed.
Mesh successfully destroyed.
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bridge Type 2

Configuring cloudSwXtch Bridge Type 2 Interfaces

By default, cloudSwXtch Bridge Type 2 installation will attempt to resolve the interface that is routable to the cloudSwXtch. However, if a user would like to do this manually, the cloudSwXtch Bridge Type 2 can be configured in one of two ways:

- For hairpin forwarding on a **single** interface
- For bridge in the middle redirection between **two** interfaces

This section will go into the changes a user would have to make to the cloudSwXtch Bridge Type 2 JSON configuration file to apply the above methods. The location of the JSON file is **/var/opt/swxtch/swxtch-bridge.json**.

Bash Copy

```
{
    "bridgeConfig": {
        "ctrlInterfaceName": "eth0",
        "dataInterfaceName": "eth1",
        "userInterfaceName": "eth0",
        "swxtchCtrlIp": "10.0.0.1",
        "swxtchCtrlPort": 80,
        "swxtchDataIp": "10.0.1.1",
        "swxtchDataPort": 9999,
        "pathId": 0,
        "groundToCloudSubscriptions": [],
        "cloudToGroundSubscriptions": [
        "225.0.23.182:12000",
        "225.0.23.183:12000",
        "225.0.23.184:12000",
        "225.0.23.185:12000"],
        "pollingIntervalMilliseconds": 1000
        "mtuSize": 1500,
        "xdpModeData": "auto",
        "xdpModeUser": "auto",
        "cloudTunIp": null
    }
}
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Fields Explained

Below are deeper explanations for certain fields in the cloudSwXtch Bridge Type 2 config file:

- **"ctrlInterfaceName"**: NIC used for control-plane communication with cloudSwXtch
- **"dataInterfaceName"**: NIC used for the data-plane communication with cloudSwXtch
- **"userInterfaceName"**: NIC used for multicast ground traffic
- **"pathId"**: Please set this to zero.
- **"groundToCloudSubscriptions"**: Please leave blank as it is no longer necessary since ground to cloud is done dynamically via IGMP joins from the cloud client.
- **"cloudToGroundSubscriptions"**: Traffic coming into the cloudSwXtch with these addresses will be forwarded to bridge and then to the userInterface.
- **"pollingIntervalMilliseconds":** Polling consists on a sync with the cloudSwXtch to exchange MC groups information.

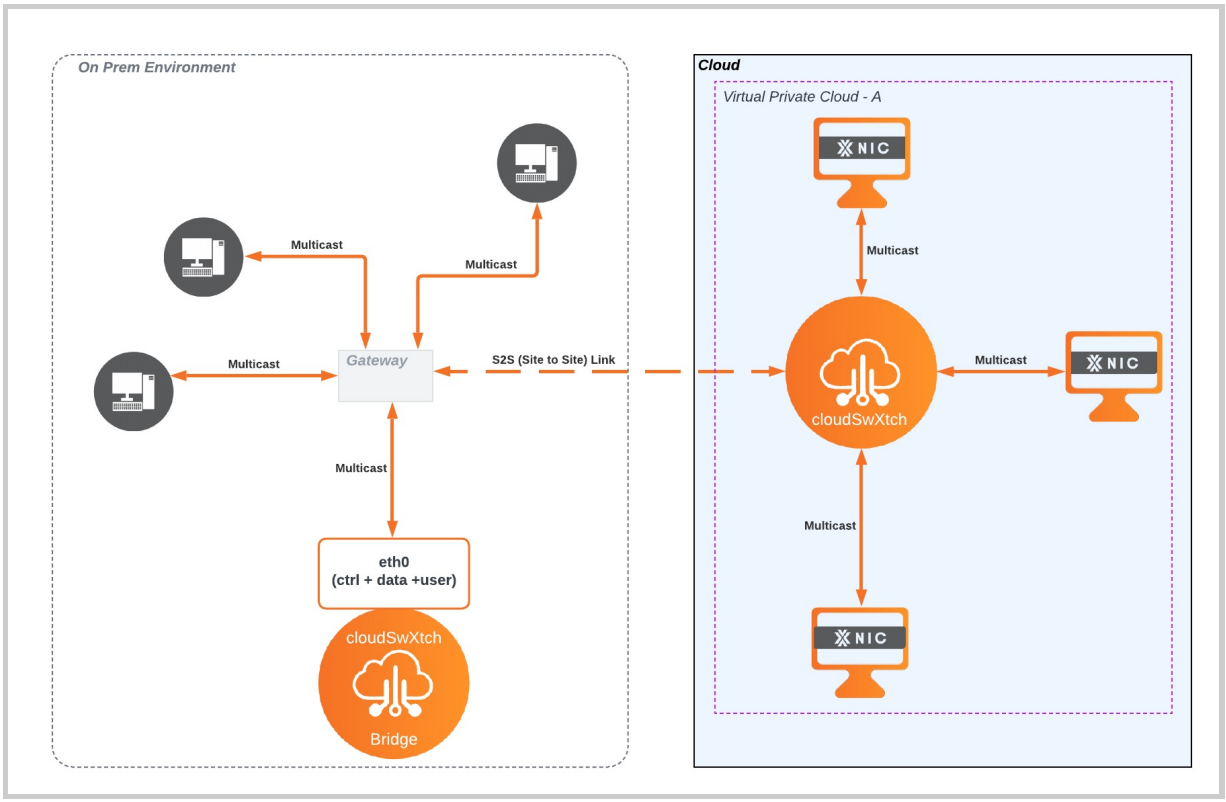
Using the status API call to get swxtch-bridge JSON file

Alternatively, users can use a **/status** API call to get the swxtch-bridge.json file. To do this, use the following command replacing the **<bridge-ctrl-IP>** with the IP address of the bridge:

Bash | Copy

```
curl -x GET http://<bridge-ctrl-ip>/status
```

For a single interface



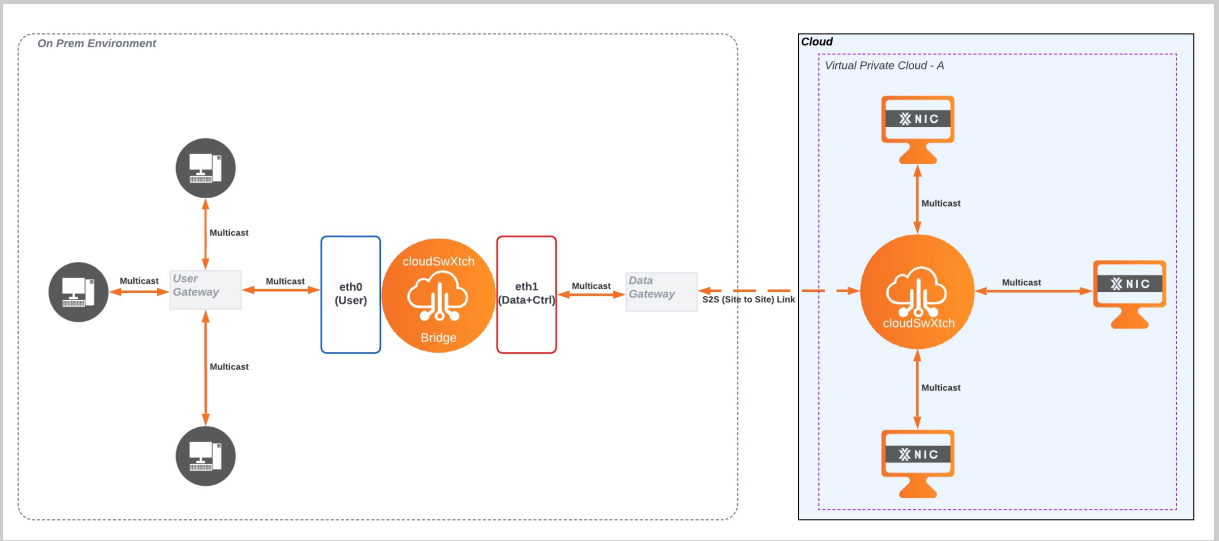
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

To accomplish a single interface configuration for your cloudSwXtch Bridge Type 2, users will need to specify the same InterfaceName for Ctrl, Data and User in the swxtch-bridge.json file. In the example, each are assigned to **eth0**.

Bash Copy

```
        "ctrlInterfaceName": "eth0",
        "dataInterfaceName": "eth0",
        "userInterfaceName": "eth0",
```

For bridge in the middle of two interfaces



Alternatively to the single interface approach, a user can decide to place the cloudSwXtch Bridge between two interfaces in order to redirect traffic from the user interface to the data interface. In the example below, the **ctrlInterfaceName** and the **dataInterfaceName** are the same (**eth1**) while the **userInterfaceName** differs (**eth0**).

Bash Copy

```
        "ctrlInterfaceName": "eth1",
        "dataInterfaceName": "eth1",
        "userInterfaceName": "eth0",
```

Using Bridge Type 2 cloud to ground API to Join/Leave

Bridge Type 2 has the capability to do join and leaves from ground to cloud via an HTTP endpoint on the bridge. This will enable the forwarding of multicast traffic from cloud to ground.

To Join:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
curl -X POST http://<BRIDGE_CTRL_IP>/addCloudToGround -d
'{"MulticastGroups":
["239.239.239.99:9000"],"UpdateConfigFile":false}'
```

To Leave:

Bash | Copy

```
curl -X POST http://<BRIDGE_CTRL_IP>/removeCloudToGround -d
'{"MulticastGroups":
["239.239.239.99:9000"],"UpdateConfigFile":false}'
```

Note: A user can set the UpdateConfigFile to **"true"** in order to make their configuration permanent. This means that the changes to cloudSwXtch Bridge Type 2 will persist between restarts.

Configuring Bridge Type 2 Static Subscriptions

The cloud to ground and ground to cloud flows are static based on entry into a json file. In order to do this, modify the bridge JSON configuration file and add the static multicast groups for either groundToCloudSubscriptions or cloudToGroundSubscriptions

The location of the configuration file is **/var/opt/swxtch/swxtch-bridge.json**. Modify the JSON array attribute for **"cloudToGroundSubscriptions" or "groundToCloudSubscriptions"** and add the appropriate multicast groups from either option.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
{
    "bridgeConfig": {
        "ctrlInterfaceName": "eth0",
        "dataInterfaceName": "eth1",
        "userInterfaceName": "eth0",
        "swxtchCtrlIp": "10.0.0.1",
        "swxtchCtrlPort": 80,
        "swxtchDataIp": "10.0.1.1",
        "swxtchDataPort": 9999,
        "pathId": 0,
        "groundToCloudSubscriptions": [
        "226.0.23.182:13000",
        "226.0.23.183:13000",
        "226.0.23.184:13000",
        "226.0.23.185:13000"],
        "cloudToGroundSubscriptions": [
        "225.0.23.182:12000",
        "225.0.23.183:12000",
        "225.0.23.184:12000",
        "225.0.23.185:12000"],
        "pollingIntervalMilliseconds": 1000
    }
}
```

After modifying the configuration file, restart the swxtch-bridge2 service with the following command:

Bash Copy

```
sudo systemctl restart swxtch-bridge2.service
```

In the example above, these multicast groups will now be sent from both cloud to ground and ground to cloud at startup for bridge.

Using a specific gateway address for Bridge Type 2

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

By default, Bridge Type 2 will resolve the data gateway MAC address by arping the first IP address of the subnet for the data interface. However, if the gateway IP address is not there, then the dataGatewayIP field can be added into the configuration file. This will force the Bridge to resolve the gateway MAC address by using the IP address specified. In the example below, the user inserted their own data gateway IP address.

Bash Copy

```
"dataGatewayIp": "192.168.1.2",
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bridge Type 1

Configuring cloudSwXtch Bridge Type 1

When launching the cloudSwXtch Bridge Type 1, the command line arguments for the input must specify the input multicast group IP address, the IP address to use within the multicast network, and the input multicast group port.

The format for the bridge **--input** argument is:

```
Bash
--input multicast://<multicast-group-ip>:<nic-ip>:<multicast-group-port>
```

Multiple multicast groups can be specified by separating them with commas:

```
PowerShell
--input multicast://<multicast-group-ip>:<nic-ip>:<multicast-group-port>,multicast://<multicast-group-ip>:<nic-ip>:<multicast-group-port>
```

The output parameter is the IP and Port of the cloudSwXtch instance where the multicast traffic will be sent to. The format for the bridge **--output** argument is:

```
PowerShell
--output udp://<swxtch-data-ip>:9999
```

The bridge application needs to know what IP address to use for the source of the multicast packets when those packets are injected into the tunnel network. This is because the network in which the bridge exists is not the same network as the tunnel network used by the application software for sending and receiving multicast traffic. This IP address should be a valid IP address in the 172.30.X.Y range and should be a unique address: i.e., not one used by another VM. This IP address in the 172.30.X.Y range shall be called the bridge source address. The format for the bridge **--override-multicast-sender-ip** argument is:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

PowerShell Copy

```
--override-multicast-sender-ip <bridge-source-address> --last-leg
```

Example: Bridge Type 2 from two multicast groups to a cloudSwXtch

In this example, the system is configured such that:

- cloudSwXtch is at 10.2.192.7 (data plane) and this IP address is reachable from the machine running the swxtch-bridge application.
- NIC IP address which the swxtch-bridge will use for receiving multicast traffic is at 169.192.0.4
- The multicast groups are 239.1.1.4:8804 and 239.1.1.1:8801
- The bridge source address was chosen to be 172.30.1.1

Example command to run the bridge:

PowerShell Copy

```
swxtch-bridge --input
multicast://239.1.1.4:169.192.0.4:8804,multicast://239.1.1.1:169.192.
0.4:8801 --output udp://10.2.192.7:9999 --override-multicast-sender-
ip 172.30.1.1 --last-leg
```

NOTE:

The bridge application assigns itself to use CPU cores 1 and 2 by default. This can be changed using the following command line parameters:

PowerShell Copy

```
--core 1 --core-count 2
```

Where core sets the starting core index (from 0) and core-count sets the number of cores to use.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Protocol Conversion and Fanout

Configuring Protocol Conversion and Fanout

There are two options for configuring Protocol Conversion and Fanout: via wXcked Eye or via the API. For more information, please see the following articles:

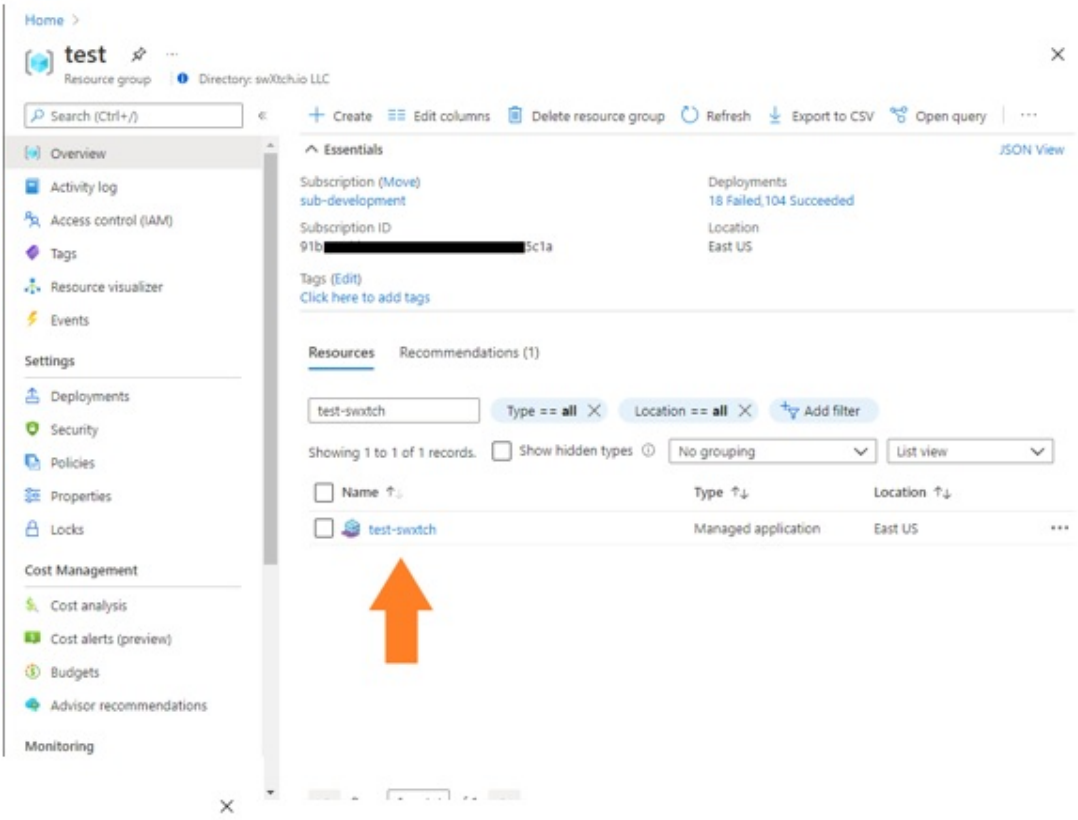
- **Protocol Conversion and Fanout with wXcked Eye**
- **Configuration API**

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Azure Monitoring

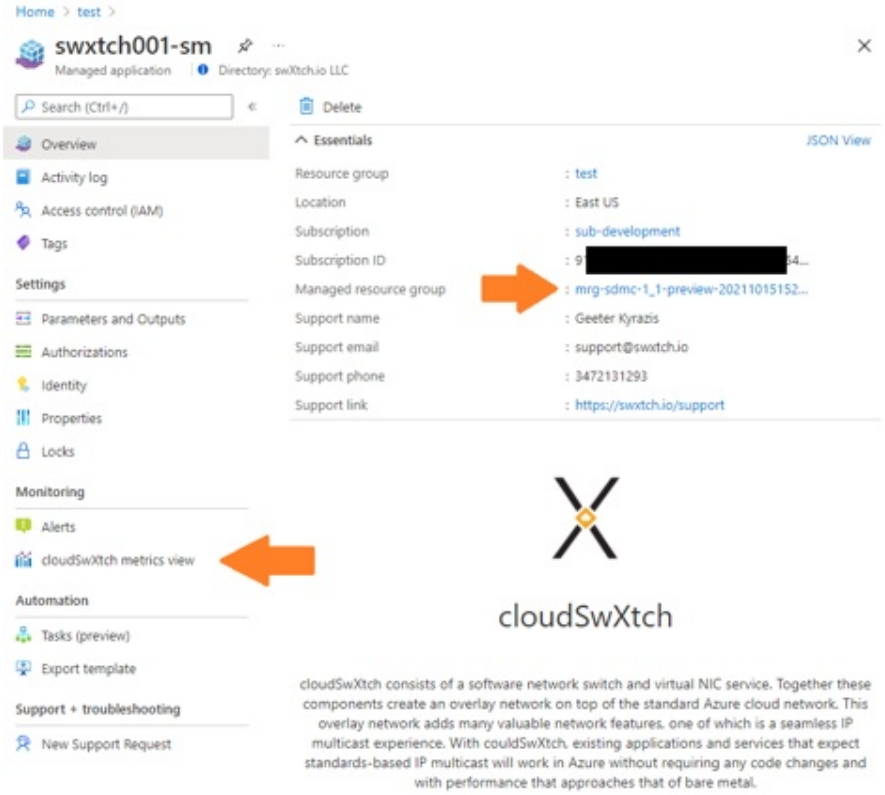
cloudSwXtch instances will show up in your Azure Resource Groups as “Managed applications” with the name given during creation. For example, the below image shows a cloudSwXtch instance with the name “test-switch” in the resource group “test”.

When you click on a cloudSwXtch instance in a resource group, you are taken to the cloudSwXtch information page for that instance. From this page you can view properties and other standard Azure component screens.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

In addition to the standard Azure component sections, this screen has two sections that are unique to the cloudSwXtch managed application: metrics view and managed application resource group.

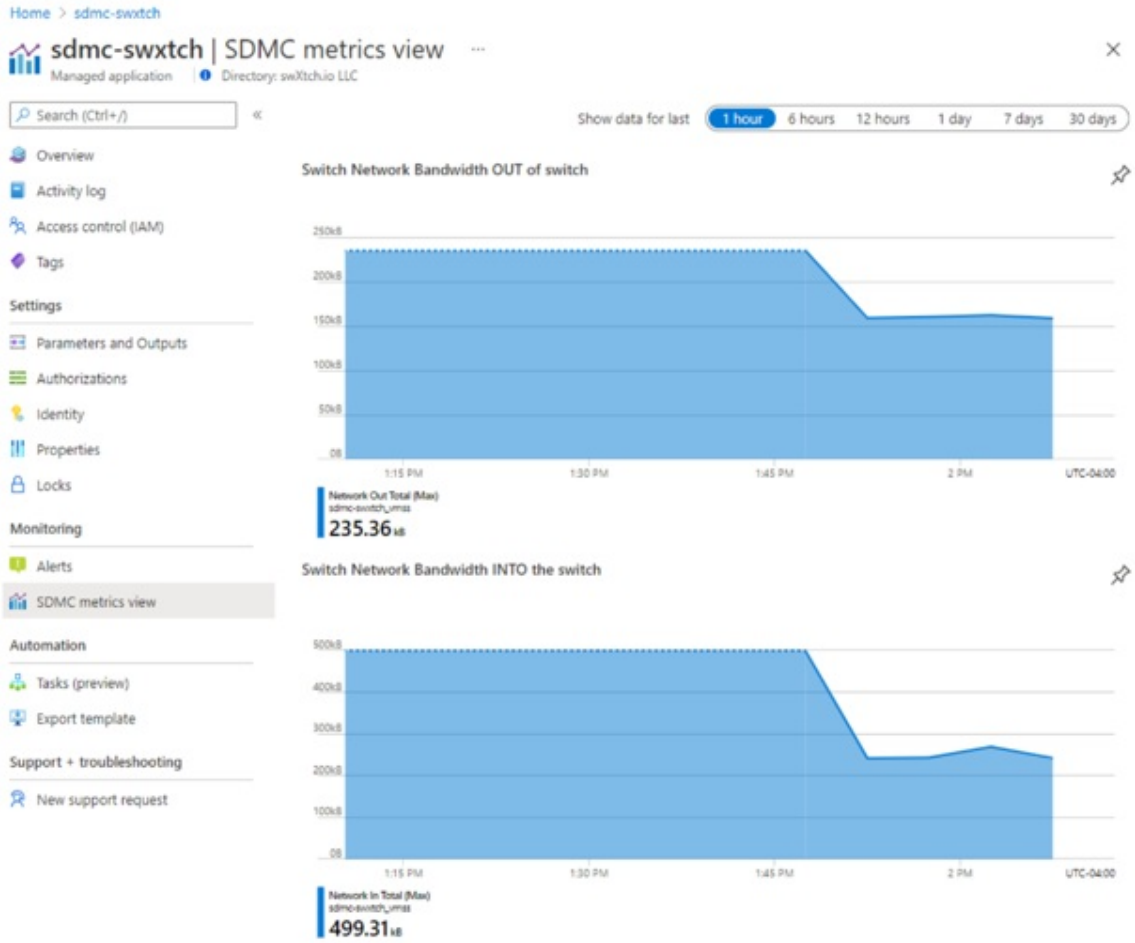


cloudSwXtch metrics view

The metrics view shows two simple graphs of the network activity of the cloudSwXtch instance. The metrics available are the total bandwidth into and out of the instance. The bandwidth units change based on the timescale chosen.

> **NOTE:**
>
> due to Azure idiosyncrasies, the metrics view will first show up around 15 minutes or so after a cloudSwXtch instance is first created. The swxtch-top application can be used immediately.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



Managed resource group

The cloudSwXtch product is delivered as a “managed application”. This means that a cloudSwXtch instance lives within the customer’s subscription and is made up of Azure resources (VMs, etc.) that are instantiated within the same subscription. These resources are directly billed to the subscription owner.

PRO TIP:

When a cloudSwXtch instance is created, it is assigned to the resource group selected by the creator and to an auto-generated resource group that holds the low-level components needed to compose the managed application. The creator of the instance has full access to the resource group that holds the instance and partial access to the auto-generated managed application resource group. The partial access allows the creator to see the various components and view their properties and metrics. It does not, however, allow the creator access to the internal VM instances that make up the managed application. The creator cannot directly control these resources from the portal, except to start/stop the VM.
For more details see:
Azure managed applications overview

Figure 2 - SDMC metrics view

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Changing xNIC configuration settings

All xNIC configuration values are normally set by the xNIC installation script. If manual changes are made to the configuration values, the xNIC service must be restarted:

```
sudo systemctl restart swxtch-xnic
```

The configuration settings for the xNIC are located at:

- Linux: `/var/opt/swxtch/swxtch-xnic.conf`
- Windows: `<tdb>`

The configuration file is a simple text file in `*.ini` format. The following values are available:

Key Name	Default value	Description and notes
SvcAddr	<ip-of-instance>	IPv4 address of the cloudSwXtch instance.
SvcPort	10802	Control port on cloudSwXtch instance.
VirtualInterfaceName	"swxtch-tun"	Base name of the virtual network interface. Must be < 15 characters.
VirtualInterfaceIpAddr	"172.30.0.0"	IPv4 subnet of the virtual network interface as seen from the host applications
VirtualInterfaceSubnet	"255.255.0.0"	IPv4 subnet mask
CtrlInterface	"eth0"	Network interface to use for control plane traffic.
DataInterface	"eth1"	Network interface to use for data plane traffic.
CtrlPort	10800	Local port used for control traffic *from* the SDMC switch
DataPort	9999	Local port used for data traffic *from* the SDMC switch

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Prometheus Monitoring

WHAT TO EXPECT

In this article, users will learn how to integrate Prometheus and Grafana as an additional way to monitor their cloudSwXtch environment.

PREREQUISITES

The following process assumes that you already have Prometheus and Grafana installed in a docker container.

STEP ONE: Validate cloudSwXtch can create Prometheus data

On the cloudSwXtch VM, run the following command:

Plaintext | Copy

```
curl http://localhost/prometheus/metrics
```

The output will list information about each metric with example output data. Metrics starting with **swx_core** are from the cloudSwXtch while metrics starting with **swx_xnic** are from xNICs. Since this example has only one cloudSwXtch but many VMs with xNICs, the xNIC data has multiple sample rows. Note that for brevity some of the xNIC rows returned have been deleted.

PowerShell | Copy

```
swxtchadmin@dsd-core-100:~$ curl http://localhost/prometheus/metrics
# HELP swx_core_droppedPacketCountByByteLimit Bytes dropped in the
swXtch
# TYPE swx_core_droppedPacketCountByByteLimit counter
swx_core_droppedPacketCountByByteLimit{category="swxtch_repl",host="
10.2.192.23"} 0
# HELP swx_core_droppedPacketCountByPacketLimit Packets dropped in
the swXtch
# TYPE swx_core_droppedPacketCountByPacketLimit counter
swx_core_droppedPacketCountByPacketLimit{category="swxtch_repl",host=
"10.2.192.23"} 0
# HELP swx_core_rxBridgeByteCount Bridge bytes received into the
swXtch
# TYPE swx_core_rxBridgeByteCount counter
swx_core_rxBridgeByteCount{category="swxtch_repl",host="10.2.192.23"
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
} 0
# HELP swx_core_rxBridgePacketCount Bridge packets received into the
swXtch
# TYPE swx_core_rxBridgePacketCount counter
swx_core_rxBridgePacketCount{category="swxtch_repl",host="10.2.192.2
3"} 0
# HELP swx_core_rxByteCount Bytes received into the swXtch
# TYPE swx_core_rxByteCount counter
swx_core_rxByteCount{category="swxtch_repl",host="10.2.192.23"}
8.797308e+07
# HELP swx_core_rxMeshByteCount Mesh bytes received into the swXtch
# TYPE swx_core_rxMeshByteCount counter
swx_core_rxMeshByteCount{category="swxtch_repl",host="10.2.192.23"}
0
# HELP swx_core_rxMeshPacketCount Mesh packets received into the
swXtch
# TYPE swx_core_rxMeshPacketCount counter
swx_core_rxMeshPacketCount{category="swxtch_repl",host="10.2.192.23"
} 0
# HELP swx_core_rxPacketCount Packets received into the swXtch
# TYPE swx_core_rxPacketCount counter
swx_core_rxPacketCount{category="swxtch_repl",host="10.2.192.23"}
505406
# HELP swx_core_rxUnicastByteCount Unicast bytes received into the
swXtch
# TYPE swx_core_rxUnicastByteCount counter
swx_core_rxUnicastByteCount{category="swxtch_repl",host="10.2.192.23"
} 0
# HELP swx_core_rxUnicastPacketCount Unicast packets received into
the swXtch
# TYPE swx_core_rxUnicastPacketCount counter
swx_core_rxUnicastPacketCount{category="swxtch_repl",host="10.2.192.
23"} 0
# HELP swx_core_sequence swXtch sequence number
# TYPE swx_core_sequence counter
swx_core_sequence{category="swxtch_repl",host="10.2.192.23"} 3480
# HELP swx_core_txBridgeByteCount Bridge bytes sent from the swXtch
# TYPE swx_core_txBridgeByteCount counter
swx_core_txBridgeByteCount{category="swxtch_repl",host="10.2.192.23"
} 0
# HELP swx_core_txBridgePacketCount Bridge packets sent from the
swXtch
# TYPE swx_core_txBridgePacketCount counter
swx_core_txBridgePacketCount{category="swxtch_repl",host="10.2.192.2
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
3"} 0
# HELP swx_core_txByteCount Bytes sent from the swXtch
# TYPE swx_core_txByteCount counter
swx_core_txByteCount{category="swxtch_repl",host="10.2.192.23"}
1.71498368e+08
# HELP swx_core_txMeshByteCount Mesh bytes sent from the swXtch
# TYPE swx_core_txMeshByteCount counter
swx_core_txMeshByteCount{category="swxtch_repl",host="10.2.192.23"}
0
# HELP swx_core_txMeshPacketCount Mesh packets sent from the swXtch
# TYPE swx_core_txMeshPacketCount counter
swx_core_txMeshPacketCount{category="swxtch_repl",host="10.2.192.23"
} 0
# HELP swx_core_txPacketCount Packets sent from the swXtch
# TYPE swx_core_txPacketCount counter
swx_core_txPacketCount{category="swxtch_repl",host="10.2.192.23"}
985630
# HELP swx_core_txUnicastByteCount Unicast bytes sent from the
swXtch
# TYPE swx_core_txUnicastByteCount counter
swx_core_txUnicastByteCount{category="swxtch_repl",host="10.2.192.23"
} 0
# HELP swx_core_txUnicastPacketCount Unicast packets sent from the
swXtch
# TYPE swx_core_txUnicastPacketCount counter
swx_core_txUnicastPacketCount{category="swxtch_repl",host="10.2.192.
23"} 0
# HELP swx_maxClientCount Maximum number of clients by license
# TYPE swx_maxClientCount gauge
swx_maxClientCount{category="swxtch"} 50
# HELP swx_numClientsConnected Number of client currently connected
# TYPE swx_numClientsConnected gauge
swx_numClientsConnected{category="swxtch"} 6
# HELP swx_xnic_byteCounters_rxMulticastCount Multicast bytes
received from the swXtch into the xNIC
# TYPE swx_xnic_byteCounters_rxMulticastCount counter
swx_xnic_byteCounters_rxMulticastCount{category="swxtch_xnic",host="
DSd-agent-101"} 7200
swx_xnic_byteCounters_rxMulticastCount{category="swxtch_xnic",host="
DSd-agent-102"} 7.0259222e+07
swx_xnic_byteCounters_rxMulticastCount{category="swxtch_xnic",host="
DSd-agent-104"} 6.9676152e+07
swx_xnic_byteCounters_rxMulticastCount{category="swxtch_xnic",host="
DSd-agent-105"} 7686
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
swx_xnic_byteCounters_rxMulticastCount{category="swxtch_xnic",host="
aks-nodepool1-23164585-vmss000018"} 7200
swx_xnic_byteCounters_rxMulticastCount{category="swxtch_xnic",host="
aks-nodepool1-23164585-vmss000019"} 7200
# HELP swx_xnic_byteCounters_rxTotalCount Total bytes received from
the swXtch into the xNIC
# TYPE swx_xnic_byteCounters_rxTotalCount counter
swx_xnic_byteCounters_rxTotalCount{category="swxtch_xnic",host="DSd-
agent-101"} 8864
swx_xnic_byteCounters_rxTotalCount{category="swxtch_xnic",host="DSd-
agent-102"} 8.6092278e+07
swx_xnic_byteCounters_rxTotalCount{category="swxtch_xnic",host="DSd-
agent-104"} 8.5377816e+07
swx_xnic_byteCounters_rxTotalCount{category="swxtch_xnic",host="DSd-
agent-105"} 9414
swx_xnic_byteCounters_rxTotalCount{category="swxtch_xnic",host="aks-
nodepool1-23164585-vmss000018"} 8864
swx_xnic_byteCounters_rxTotalCount{category="swxtch_xnic",host="aks-
nodepool1-23164585-vmss000019"} 8864
# HELP swx_xnic_byteCounters_txMulticastCount Multicast bytes sent
from the xNIC into the swXtch
# TYPE swx_xnic_byteCounters_txMulticastCount counter
swx_xnic_byteCounters_txMulticastCount{category="swxtch_xnic",host="
DSd-agent-101"} 0
swx_xnic_byteCounters_txMulticastCount{category="swxtch_xnic",host="
DSd-agent-102"} 0
swx_xnic_byteCounters_txMulticastCount{category="swxtch_xnic",host="
DSd-agent-104"} 96222
swx_xnic_byteCounters_txMulticastCount{category="swxtch_xnic",host="
DSd-agent-105"} 7686
swx_xnic_byteCounters_txMulticastCount{category="swxtch_xnic",host="
aks-nodepool1-23164585-vmss000018"} 7.1697362e+07
swx_xnic_byteCounters_txMulticastCount{category="swxtch_xnic",host="
aks-nodepool1-23164585-vmss000019"} 0
# HELP swx_xnic_byteCounters_txTotalCount Total bytes sent from the
xNIC into the swXtch
# TYPE swx_xnic_byteCounters_txTotalCount counter
swx_xnic_byteCounters_txTotalCount{category="swxtch_xnic",host="DSd-
agent-101"} 0
swx_xnic_byteCounters_txTotalCount{category="swxtch_xnic",host="DSd-
agent-102"} 0
swx_xnic_byteCounters_txTotalCount{category="swxtch_xnic",host="DSd-
agent-104"} 111006
swx_xnic_byteCounters_txTotalCount{category="swxtch_xnic",host="DSd-
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
agent-105"} 9414
swx_xnic_byteCounters_txTotalCount{category="swxtch_xnic",host="aks-
nodepool1-23164585-vmss000018"} 8.7854514e+07
swx_xnic_byteCounters_txTotalCount{category="swxtch_xnic",host="aks-
nodepool1-23164585-vmss000019"} 0
# HELP swx_xnic_latencies_count xNIC latency count
# TYPE swx_xnic_latencies_count gauge
swx_xnic_latencies_count{category="swxtch_xnic",host="DSd-agent-
101"} 0
swx_xnic_latencies_count{category="swxtch_xnic",host="DSd-agent-
102"} 0
swx_xnic_latencies_count{category="swxtch_xnic",host="DSd-agent-
104"} 0
swx_xnic_latencies_count{category="swxtch_xnic",host="DSd-agent-
105"} 0
swx_xnic_latencies_count{category="swxtch_xnic",host="aks-nodepool1-
23164585-vmss000018"} 0
swx_xnic_latencies_count{category="swxtch_xnic",host="aks-nodepool1-
23164585-vmss000019"} 0
# HELP swx_xnic_latencies_sum xNIC latency sum
# TYPE swx_xnic_latencies_sum gauge
swx_xnic_latencies_sum{category="swxtch_xnic",host="DSd-agent-101"}
0
swx_xnic_latencies_sum{category="swxtch_xnic",host="DSd-agent-102"}
0
swx_xnic_latencies_sum{category="swxtch_xnic",host="DSd-agent-104"}
0
swx_xnic_latencies_sum{category="swxtch_xnic",host="DSd-agent-105"}
0
swx_xnic_latencies_sum{category="swxtch_xnic",host="aks-nodepool1-
23164585-vmss000018"} 0
swx_xnic_latencies_sum{category="swxtch_xnic",host="aks-nodepool1-
23164585-vmss000019"} 0
# HELP swx_xnic_packetCounters_rxDroppedCount Lost packets received
from the swXtch into the xNIC
# TYPE swx_xnic_packetCounters_rxDroppedCount counter
swx_xnic_packetCounters_rxDroppedCount{category="swxtch_xnic",host="
DSd-agent-101"} 0
swx_xnic_packetCounters_rxDroppedCount{category="swxtch_xnic",host="
DSd-agent-102"} 0
swx_xnic_packetCounters_rxDroppedCount{category="swxtch_xnic",host="
DSd-agent-104"} 0
swx_xnic_packetCounters_rxDroppedCount{category="swxtch_xnic",host="
DSd-agent-105"} 0
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
swx_xnic_packetCounters_rxDroppedCount{category="swxtch_xnic",host="
aks-nodepool1-23164585-vmss000018"} 0
swx_xnic_packetCounters_rxDroppedCount{category="swxtch_xnic",host="
aks-nodepool1-23164585-vmss000019"} 0
# HELP swx_xnic_packetCounters_rxMulticastCount Multicast packets
received from the swXtch into the xNIC
# TYPE swx_xnic_packetCounters_rxMulticastCount counter
swx_xnic_packetCounters_rxMulticastCount{category="swxtch_xnic",host=
"DSd-agent-101"} 52
swx_xnic_packetCounters_rxMulticastCount{category="swxtch_xnic",host=
"DSd-agent-102"} 494783
swx_xnic_packetCounters_rxMulticastCount{category="swxtch_xnic",host=
"DSd-agent-104"} 490677
swx_xnic_packetCounters_rxMulticastCount{category="swxtch_xnic",host=
"DSd-agent-105"} 54
swx_xnic_packetCounters_rxMulticastCount{category="swxtch_xnic",host=
"aks-nodepool1-23164585-vmss000018"} 52
swx_xnic_packetCounters_rxMulticastCount{category="swxtch_xnic",host=
"aks-nodepool1-23164585-vmss000019"} 52
# HELP swx_xnic_packetCounters_rxTotalCount Total packets received
from the swXtch into the xNIC
# TYPE swx_xnic_packetCounters_rxTotalCount counter
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="DS
d-agent-101"} 52
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="DS
d-agent-102"} 494783
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="DS
d-agent-104"} 490677
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="DS
d-agent-105"} 54
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="ak
s-nodepool1-23164585-vmss000018"} 52
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="ak
s-nodepool1-23164585-vmss000019"} 52
# HELP swx_xnic_packetCounters_txDroppedCount Lost packets sent from
the xNIC into the swXtch
# TYPE swx_xnic_packetCounters_txDroppedCount counter
swx_xnic_packetCounters_txDroppedCount{category="swxtch_xnic",host="
DSd-agent-101"} 0
swx_xnic_packetCounters_txDroppedCount{category="swxtch_xnic",host="
DSd-agent-102"} 0
swx_xnic_packetCounters_txDroppedCount{category="swxtch_xnic",host="
DSd-agent-104"} 0
swx_xnic_packetCounters_txDroppedCount{category="swxtch_xnic",host="
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
DSd-agent-105"} 0
swx_xnic_packetCounters_txDroppedCount{category="swxtch_xnic",host="
aks-nodepool1-23164585-vmss000018"} 0
swx_xnic_packetCounters_txDroppedCount{category="swxtch_xnic",host="
aks-nodepool1-23164585-vmss000019"} 0
# HELP swx_xnic_packetCounters_txIgmpCount IGMP packets sent from
the xNIC into the swXtch
# TYPE swx_xnic_packetCounters_txIgmpCount counter
swx_xnic_packetCounters_txIgmpCount{category="swxtch_xnic",host="DSd
-agent-101"} 0
swx_xnic_packetCounters_txIgmpCount{category="swxtch_xnic",host="DSd
-agent-102"} 0
swx_xnic_packetCounters_txIgmpCount{category="swxtch_xnic",host="DSd
-agent-104"} 0
swx_xnic_packetCounters_txIgmpCount{category="swxtch_xnic",host="DSd
-agent-105"} 0
swx_xnic_packetCounters_txIgmpCount{category="swxtch_xnic",host="aks
-nodepool1-23164585-vmss000018"} 0
swx_xnic_packetCounters_txIgmpCount{category="swxtch_xnic",host="aks
-nodepool1-23164585-vmss000019"} 0
# HELP swx_xnic_packetCounters_txMulticastCount Multicast packets
sent from the xNIC into the swXtch
# TYPE swx_xnic_packetCounters_txMulticastCount counter
swx_xnic_packetCounters_txMulticastCount{category="swxtch_xnic",host=
"DSd-agent-101"} 0
swx_xnic_packetCounters_txMulticastCount{category="swxtch_xnic",host=
"DSd-agent-102"} 0
swx_xnic_packetCounters_txMulticastCount{category="swxtch_xnic",host=
"DSd-agent-104"} 462
swx_xnic_packetCounters_txMulticastCount{category="swxtch_xnic",host=
"DSd-agent-105"} 54
swx_xnic_packetCounters_txMulticastCount{category="swxtch_xnic",host=
"aks-nodepool1-23164585-vmss000018"} 504911
swx_xnic_packetCounters_txMulticastCount{category="swxtch_xnic",host=
"aks-nodepool1-23164585-vmss000019"} 0
# HELP swx_xnic_packetCounters_txTotalCount Total packets sent from
the xNIC into the swXtch
# TYPE swx_xnic_packetCounters_txTotalCount counter
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="DS
d-agent-101"} 0
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="DS
d-agent-102"} 0
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="DS
d-agent-104"} 462
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="DS
d-agent-105"} 54
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="ak
s-nodepool1-23164585-vmss000018"} 504911
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="ak
s-nodepool1-23164585-vmss000019"} 0
# HELP swx_xnic_rxMulticastGroups_byteCount Multicast group traffic
sent from the Swxtch into the xNIC - Multicast group byte count
# TYPE swx_xnic_rxMulticastGroups_byteCount counter
swx_xnic_rxMulticastGroups_byteCount{category="swxtch_xnic",cloudSwx
tchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host="D
Sd-agent-
101",hostAddress="10.2.128.29",index="3",osDistribution="Ubuntu
22.04",xNicType="t2"} 1116
swx_xnic_rxMulticastGroups_byteCount{category="swxtch_xnic",cloudSwx
tchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host="D
Sd-agent-
101",hostAddress="10.2.128.29",index="5",osDistribution="Ubuntu
22.04",xNicType="t2"} 1116
swx_xnic_rxMulticastGroups_byteCount{category="swxtch_xnic",cloudSwx
tchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host="D
Sd-agent-
101",hostAddress="10.2.128.29",index="6",osDistribution="Ubuntu
22.04",xNicType="t2"} 1100
swx_xnic_rxMulticastGroups_byteCount{category="swxtch_xnic",cloudSwx
tchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host="D
Sd-agent-
101",hostAddress="10.2.128.29",index="7",osDistribution="Ubuntu
22.04",xNicType="t2"} 1100
# HELP swx_xnic_rxMulticastGroups_packetCount Multicast group
traffic sent from the Swxtch into the xNIC - Multicast group packet
count
# TYPE swx_xnic_rxMulticastGroups_packetCount counter
swx_xnic_rxMulticastGroups_packetCount{category="swxtch_xnic",cloudS
wxtchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host=
"DSd-agent-
101",hostAddress="10.2.128.29",index="3",osDistribution="Ubuntu
22.04",xNicType="t2"} 9
swx_xnic_rxMulticastGroups_packetCount{category="swxtch_xnic",cloudS
wxtchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host=
"DSd-agent-
101",hostAddress="10.2.128.29",index="5",osDistribution="Ubuntu
22.04",xNicType="t2"} 9
swx_xnic_rxMulticastGroups_packetCount{category="swxtch_xnic",cloudS
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
wxtchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host=
"DSd-agent-
101",hostAddress="10.2.128.29",index="6",osDistribution="Ubuntu
22.04",xNicType="t2"} 4
swx_xnic_rxMulticastGroups_packetCount{category="swxtch_xnic",cloudS
wxtchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host=
"DSd-agent-
101",hostAddress="10.2.128.29",index="7",osDistribution="Ubuntu
22.04",xNicType="t2"} 4
# HELP swx_xnic_txMulticastGroups_byteCount Multicast group traffic
sent from the xNIC into the Swxtch - Multicast group byte count
# TYPE swx_xnic_txMulticastGroups_byteCount counter
swx_xnic_txMulticastGroups_byteCount{category="swxtch_xnic",cloudSwx
tchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host="D
Sd-agent-
104",hostAddress="10.2.128.75",index="100",osDistribution="Windows
Server 2019 Datacenter - Microsoft Windows [Version
10.0.17763.5329]",xNicType="t2"} 2988
swx_xnic_txMulticastGroups_byteCount{category="swxtch_xnic",cloudSwx
tchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host="D
Sd-agent-
104",hostAddress="10.2.128.75",index="27",osDistribution="Windows
Server 2019 Datacenter - Microsoft Windows [Version
10.0.17763.5329]",xNicType="t2"} 2853
# HELP swx_xnic_txMulticastGroups_packetCount Multicast group
traffic sent from the xNIC into the Swxtch - Multicast group packet
count
# TYPE swx_xnic_txMulticastGroups_packetCount counter
swx_xnic_txMulticastGroups_packetCount{category="swxtch_xnic",cloudS
wxtchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host=
"DSd-agent-
104",hostAddress="10.2.128.75",index="100",osDistribution="Windows
Server 2019 Datacenter - Microsoft Windows [Version
10.0.17763.5329]",xNicType="t2"} 18
swx_xnic_txMulticastGroups_packetCount{category="swxtch_xnic",cloudS
wxtchVersion="dev.cloudswxtch.2.1.1.10",groupIp="10.2.131.255",host=
"DSd-agent-
104",hostAddress="10.2.128.75",index="27",osDistribution="Windows
Server 2019 Datacenter - Microsoft Windows [Version
10.0.17763.5329]",xNicType="t2"} 9
```

If successful (there is an output), continue on to updating your Prometheus Directory for cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

STEP TWO: Update Prometheus Directory for cloudSwXtch

Attached is an example prometheus.yaml file with an cloudSwXtch job name configuration.

prometheus 732 Byte

1. Open the example **prometheus.yml** file and copy lines 26-36.
2. Paste those lines into your existing **prometheus.yml** file.
3. Update the cloudSwXtch **targets** line that has " **127.0.0.1:80**" and put in the IP address of the cloudSwXtch in place of the localhost IP.
    a. **Please note**: If Prometheus and cloudSwXtch are on the same VM, then the localhost IP (127.0.0.1) will still work.

```
26  - job_name: swxtch
27    honor_timestamps: true
28    scrape_interval: 5s
29    scrape_timeout: 2s
30    metrics_path: /prometheus/metrics
31    scheme: http
32    follow_redirects: true
33    enable_http2: true
34    static_configs:
35    - targets:
36        - 127.0.0.1:80
37
```

4. Run the following docker command to run it in VM. If you decide to run it this way, you will need to run it after every reboot or when you close your window. Please use this method when testing in order to limit the amount of records added to the Prometheus database.

Plaintext Copy

```
docker run --network host -v ~/prometheus/:/etc/prometheus
prom/prometheus
```

5. Use this command to run Prometheus automatically upon reboot. (Preferred method for a production environment.)

Plaintext Copy

```
docker run --network host --restart always -v
~/prometheus/:/etc/prometheus prom/prometheus
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

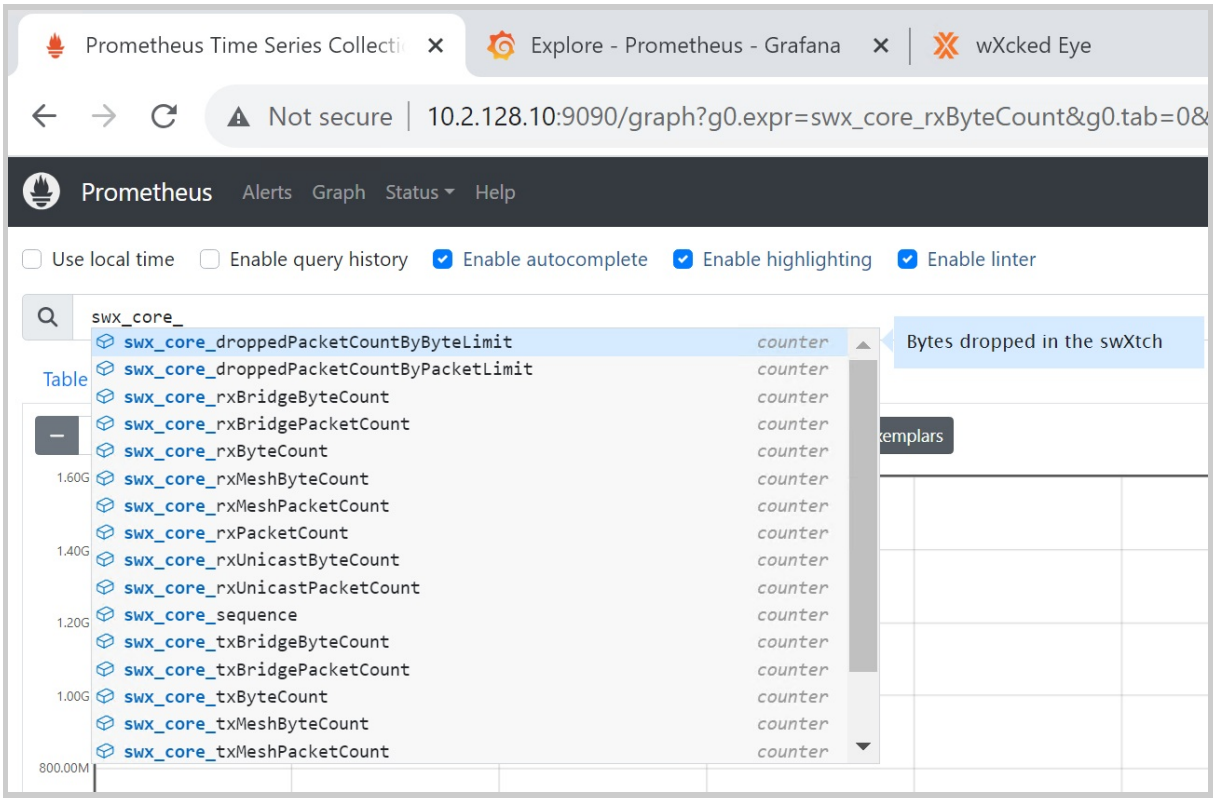
STEP THREE: Access Prometheus UI

In order to access the Prometheus UI, users should open a browser on their Windows machine in the same VNET and enter the following URL:

Plaintext	Copy
`http://<prometheus-IP>:9090/`	

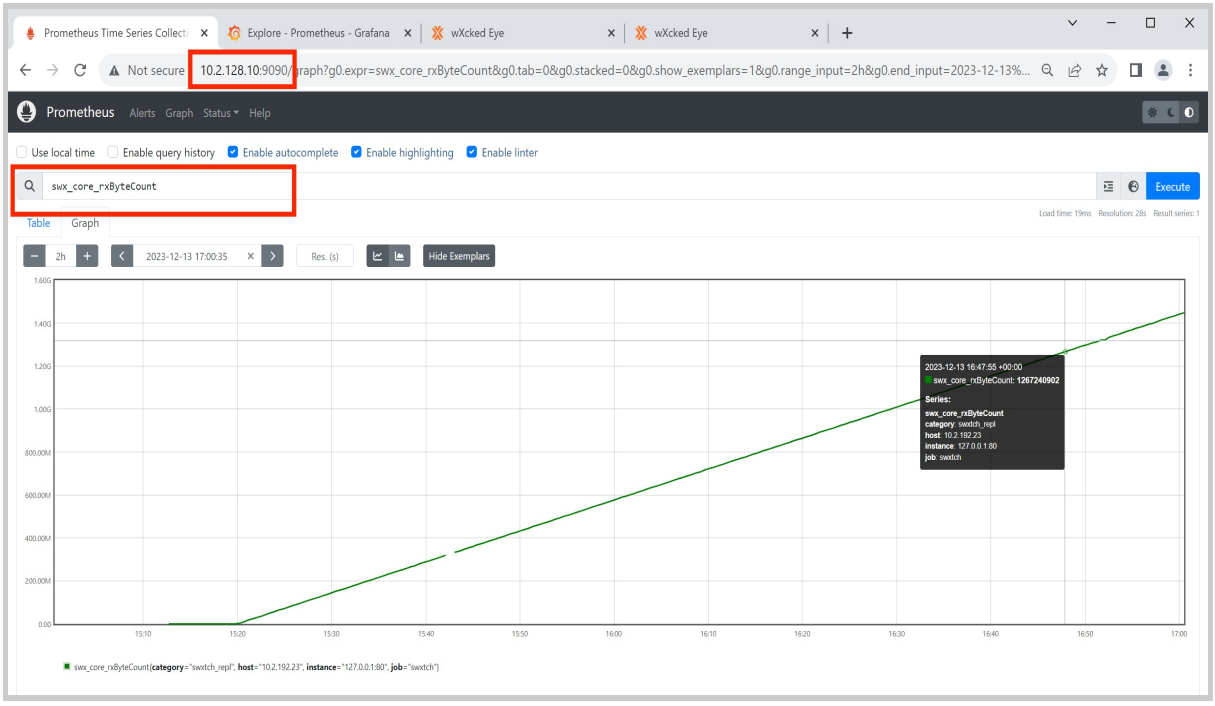
Note: Replace the <prometheus-IP> with the IP address of your Prometheus instance.

Users can enter the prefix “swx” into the search field to see a list of data fields related to the cloudSwXtch (swx_core) and its xNICs (swx_xnic).



In the example below, the user has chosen **swx_core_rxByteCount** as their data field.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



By selecting Execute, users will be able to populate the table with the desired data. **Note**: Producers and consumers must be running in order to see data. In the example above, a user can select Execute multiple times and notice that the number in the orange box grow in size.

For a list of available data fields, view the dropdown section here:

prometheus/metrics

URL

Plaintext	Copy
`http://<core>/prometheus/metrics`	

Parameters

None

Request

None

Response

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

200 - Success

Data Fields

Metric	Description	Type
swx_core_droppedPacketCountByByteLimit	Bytes dropped in the swXtch	counter
swx_core_droppedPacketCountByPacketLimit	Packets dropped in the swXtch	counter
swx_core_rxBridgeByteCount	Bridge bytes received into the swXtch	counter
swx_core_rxBridgePacketCount	Bridge packets received into the swXtch	counter
swx_core_rxByteCount	Bytes received into the swXtch	counter
swx_core_rxMeshByteCount	Mesh bytes received into the swXtch	counter
swx_core_rxMeshPacketCount	Mesh packets received into the swXtch	counter
swx_core_rxPacketCount	Packets received into the swXtch	counter
swx_core_rxUnicastByteCount	Unicast bytes received into the swXtch	counter
swx_core_rxUnicastPacketCount	Unicast packets received into the swXtch	counter
swx_core_sequence	swXtch sequence number	counter
swx_core_txBridgeByteCount	Bridge bytes sent from the swXtch	counter
swx_core_txBridgePacketCount	Bridge packets sent from the swXtch	counter
swx_core_txByteCount	Bytes sent from the swXtch	counter
swx_core_txMeshByteCount	Mesh bytes sent from the swXtch	counter
swx_core_txMeshPacketCount	Mesh packets sent from the swXtch	counter
swx_core_txPacketCount	Packets sent from the swXtch	counter
swx_core_txUnicastByteCount	Unicast bytes sent from the swXtch	counter
swx_core_txUnicastPacketCount	Unicast packets sent from the swXtch	counter
swx_xnic_activeConnectionCount	xNIC active connections	gauge
swx_xnic_byteCount		
swx_xnic_byteCounters_rxMulticastCount	Multicast bytes received from the swXtch into the xNIC	counter
swx_xnic_byteCounters_rxTotalCount	Total bytes received from the swXtch into the xNIC	counter

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

swx_xnic_byteCounters_txMulticastCount	Multicast bytes sent from the xNIC into the swXtch	counter
swx_xnic_byteCounters_txTotalCount	Total bytes sent from the xNIC into the swXtch	counter
swx_xnic_latencies_count	xNIC latency count	gauge
swx_xnic_latencies_sum	xNIC latency sum	guage
swx_xnic_maxActiveConnections	xNIC max number of active connections	guage
swx_xnic_packetCounters_rxDroppedCount	Lost packets received from the swXtch into the xNIC	counter
swx_xnic_packetCounters_rxMulticastCount	Multicast packets received from the swXtch into the xNIC	counter
swx_xnic_packetCounters_rxTotalCount	Total packets received from the swXtch into the xNIC	counter
swx_xnic_packetCounters_txDroppedCount	Lost packets sent from the xNIC into the swXtch	counter
swx_xnic_packetCounters_txIgmpCount	IGMP packets sent from the xNIC into the swXtch	counter
swx_xnic_packetCounters_txMulticastCount	Multicast packets sent from the xNIC into the swXtch	counter
swx_xnic_packetCounters_txTotalCount	Total packets sent from the xNIC into the swXtch	counter
swx_xnic_rxHaCounters_egressByteCount	HA bytes sent	counter
swx_xnic_rxHaCounters_egressPacketCount	HA packets sent	counter
swx_xnic_rxHaCounters_enqueueFailureCount	HA enqueue failure count	counter
swx_xnic_rxHaCounters_outputStreamLossCount	HA output stream lose packets	counter
swx_xnic_rxHaCounters_paths_ingressByteCount	HA Bytes received	counter
swx_xnic_rxHaCounters_paths_ingressPacketCount	HA Packets received	counter
swx_xnic_rxHaCounters_paths_missingPacketCount	HA Packets missing	counter
swx_xnic_rxHaCounters_paths_usedPacketCount	HA Packets used	counter

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

swx_xnic_rxMulticastGroups_byteCount	Multicast Groups bytes sent from the SwXtch into the xNIC	counter
swx_xnic_rxMulticastGroups_packetCount	Multicast Groups packets sent from the SwXtch into the xNIC	counter
swx_xnic_timestamp	timestamp	counter
swx_xnic_txMulticastGroups_byteCount	Multicast Groups bytes sent from the xNIC into the SwXtch	counter
swx_xnic_txMulticastGroups_packetCount	Multicast Groups packets sent from the xNIC into the SwXtch	counter
swx_xnic_usedPacketCount	Packets	counter

STEP FOUR: Access Grafana UI

Alternatively, users can also use Grafana as another method for viewing cloudSwXtch metrics.

1. In a Windows machine on the same VNET, open a browser and enter the following URL:

Plaintext Copy

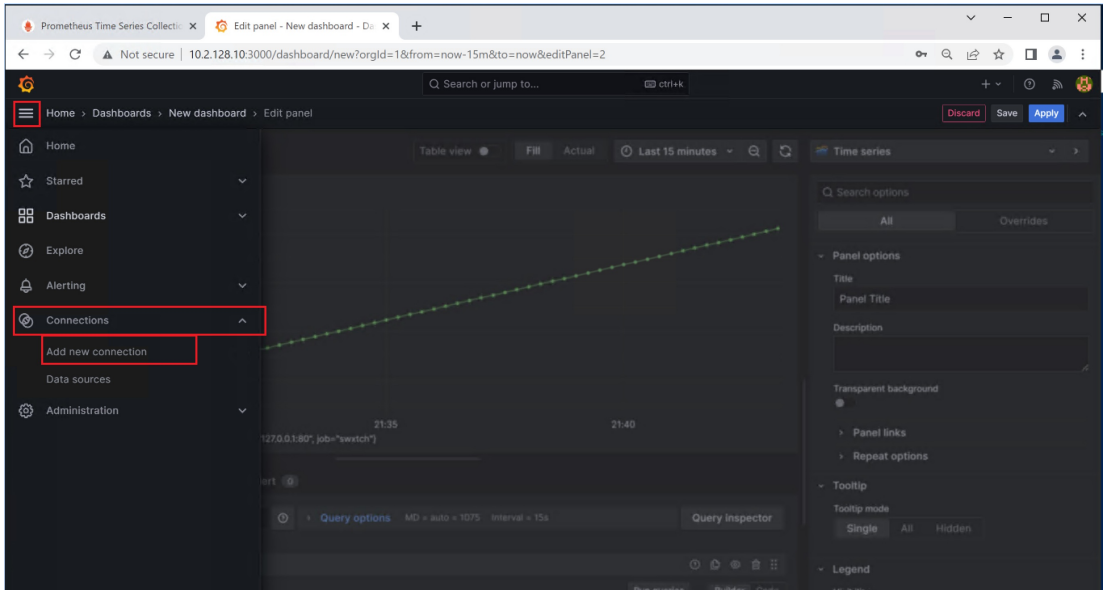
```
http://<Grafana-IP>:3000/
```

Note: Replace the <Grafana-IP> with the IP address of your Grafana instance.

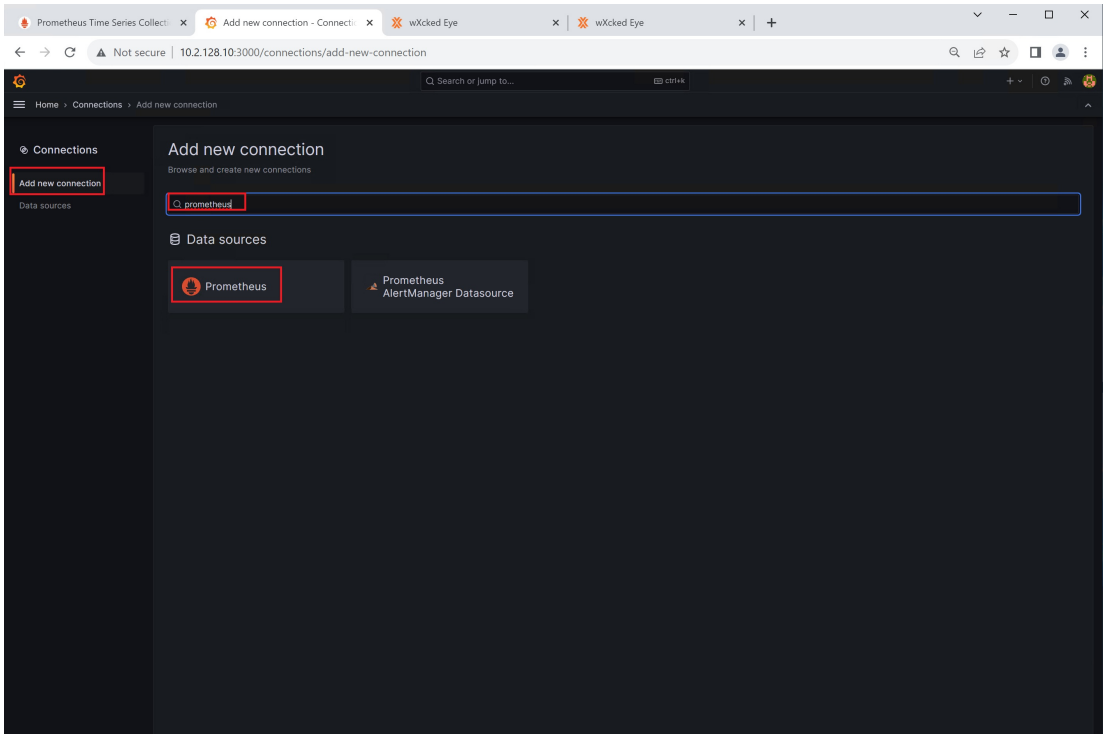
2. Sign in as a user **admin** and the password **admin**.
3. Click on the three horizontal lines next to **Home** to get additional options.
4. Select **Connection**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

5. Click **Add new connection**.

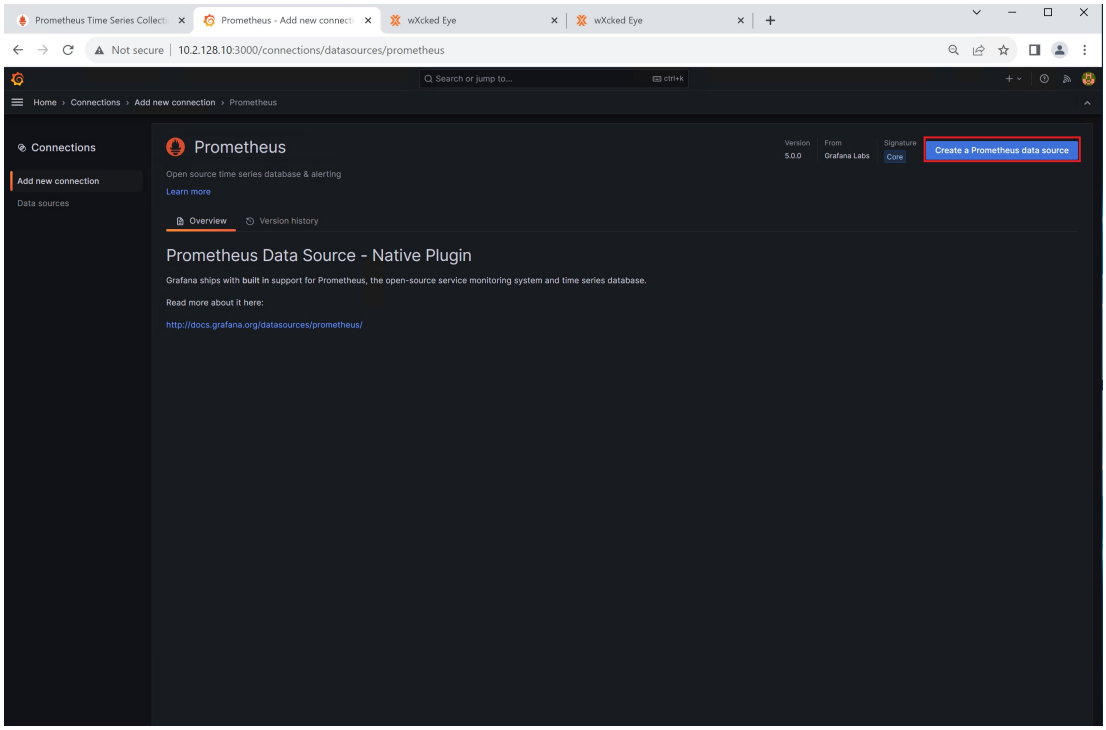


6. Search **Prometheus** on the **Add new connection** page and select it from the options.



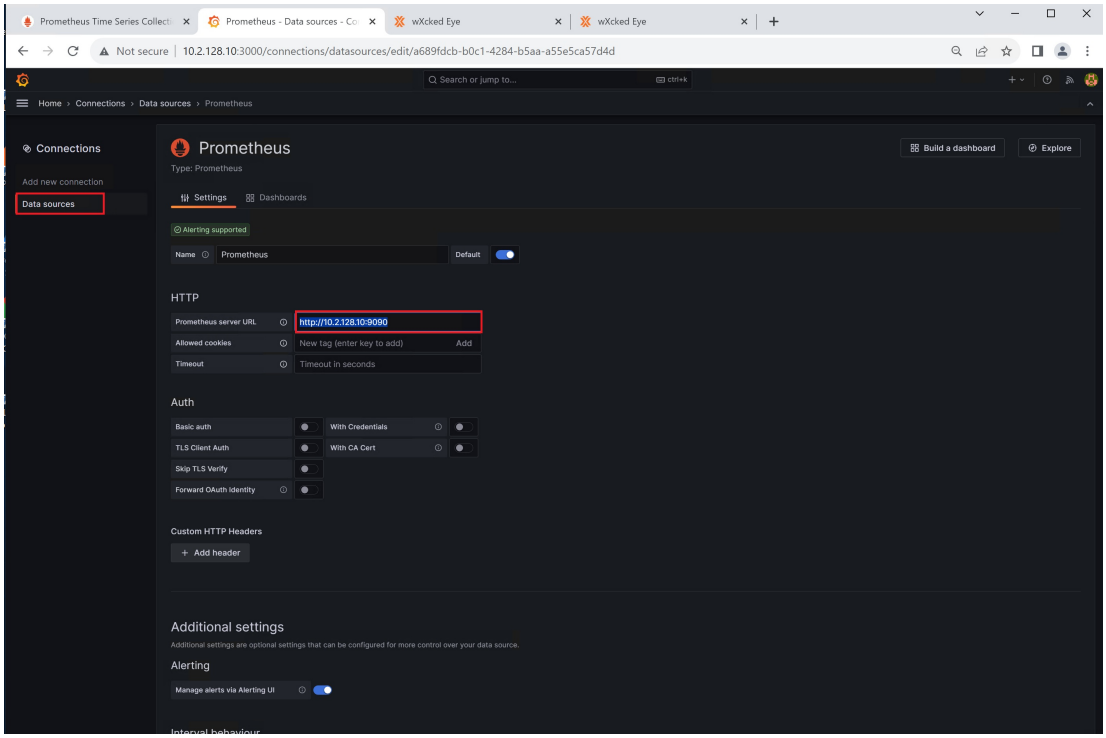
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

7. Click **Create a Prometheus data source**.



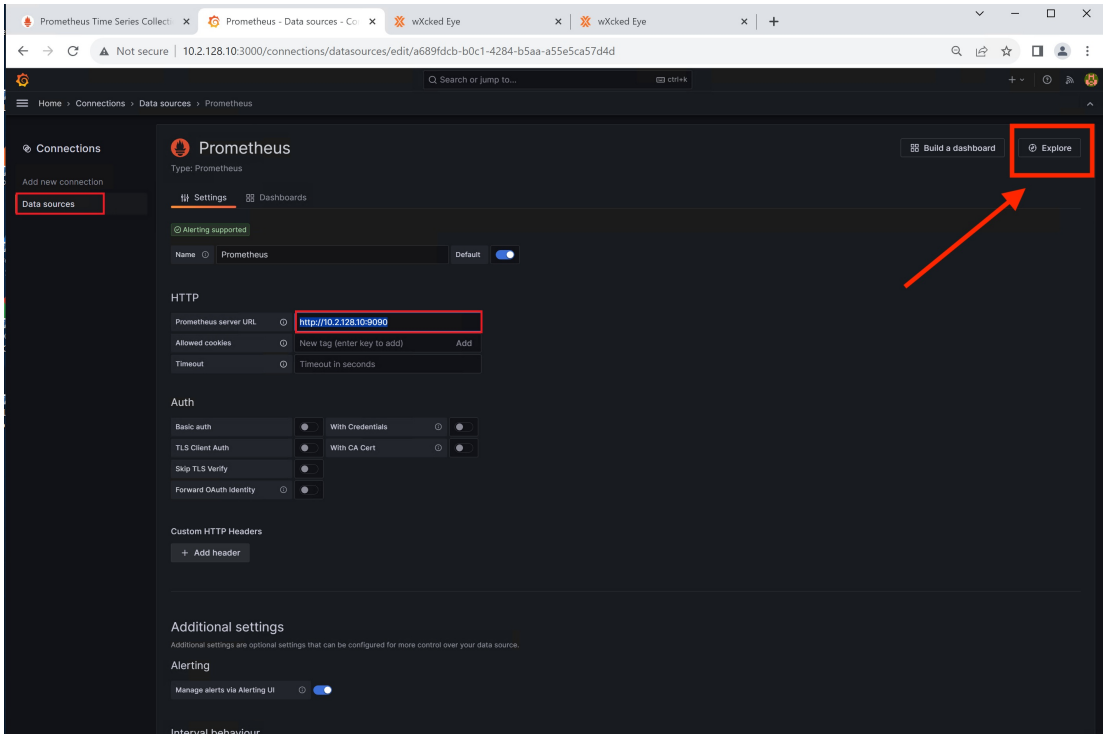
8. Go to **Data Sources** and select the **Prometheus** data source created.

9. Enter the **Prometheus server URL**. This should include the **IP address of your Prometheus instance**.

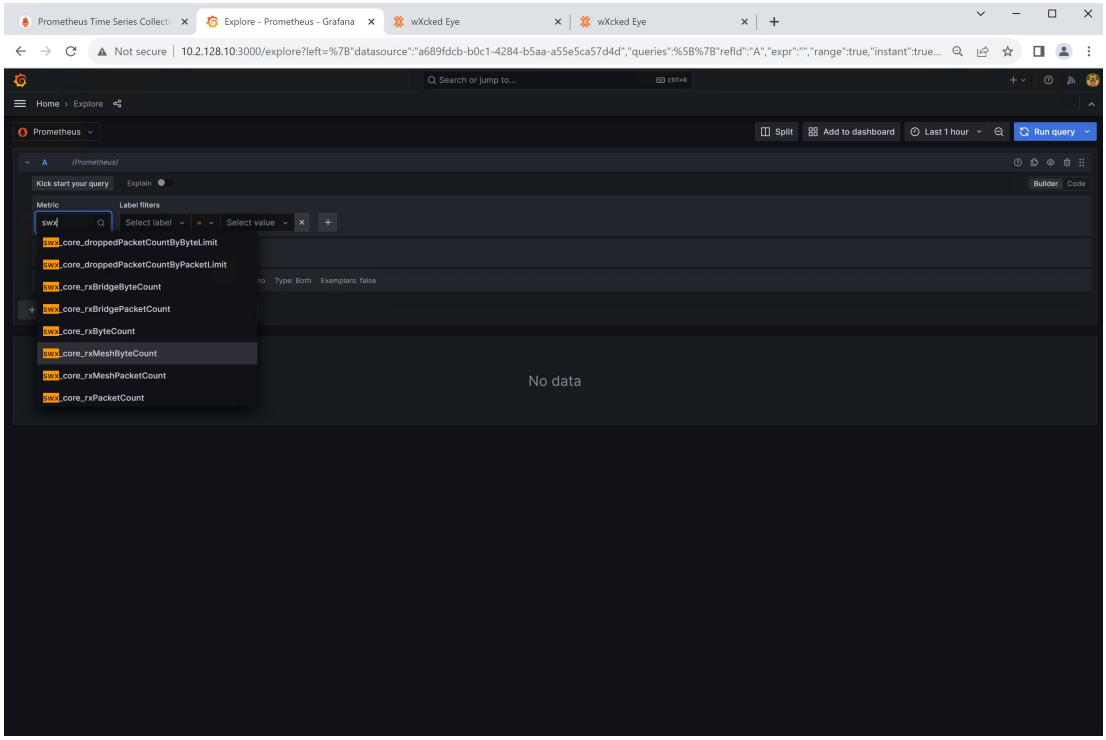


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

10. Click **Explore**.



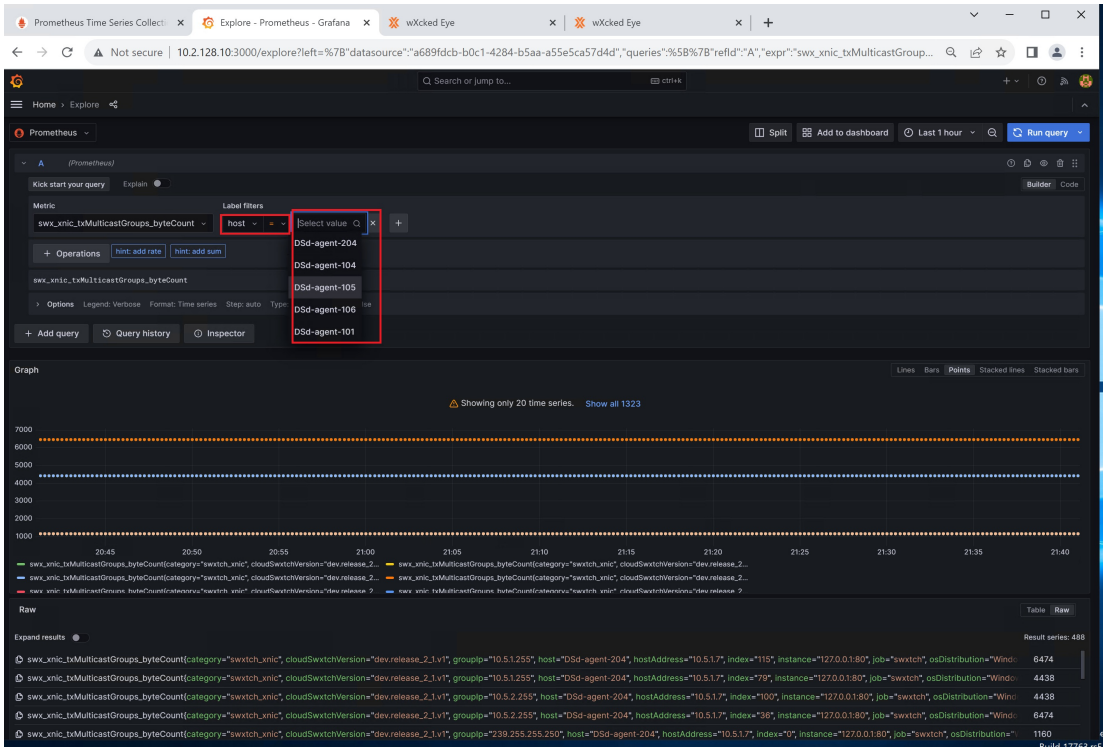
11. In the search bar under **Metric**, use the prefix “swx” to populate a list of potential data fields.



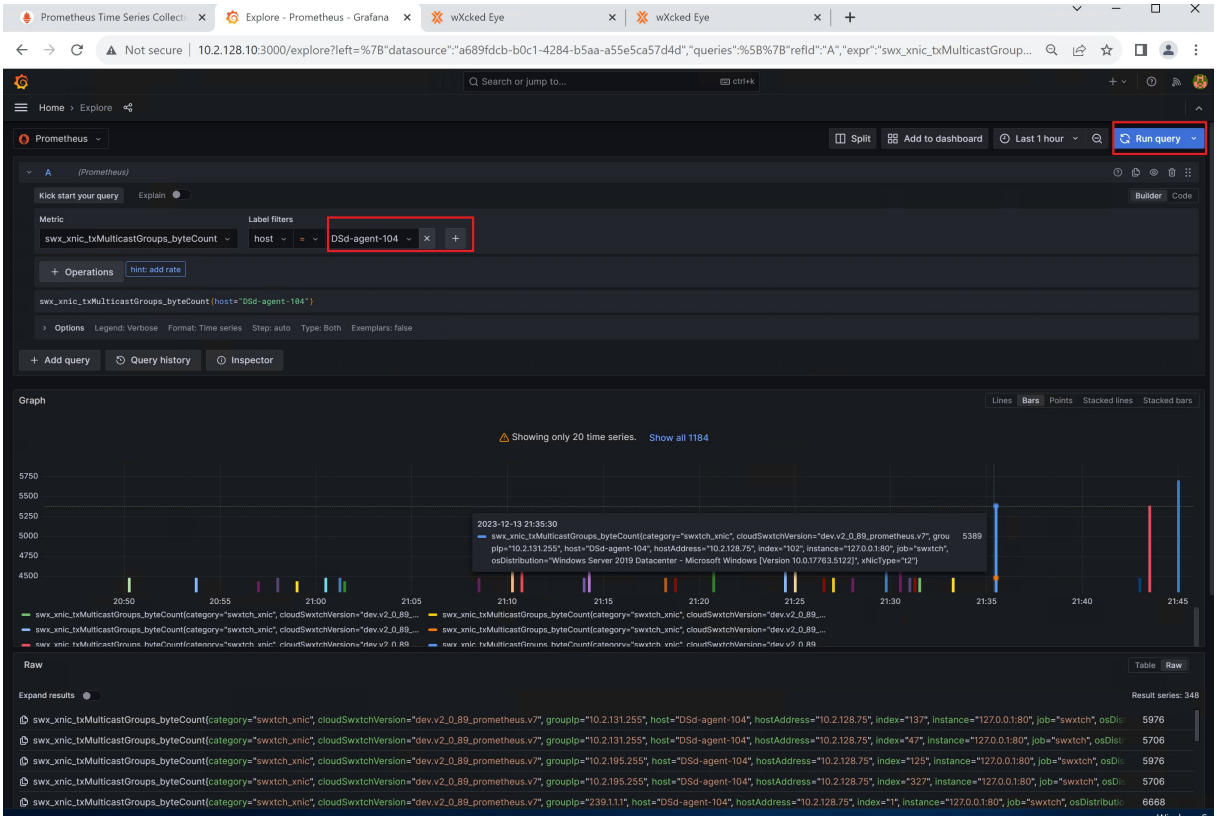
12. Scroll through the list and select what is desired.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

13. **Run the query**. The results will appear below.

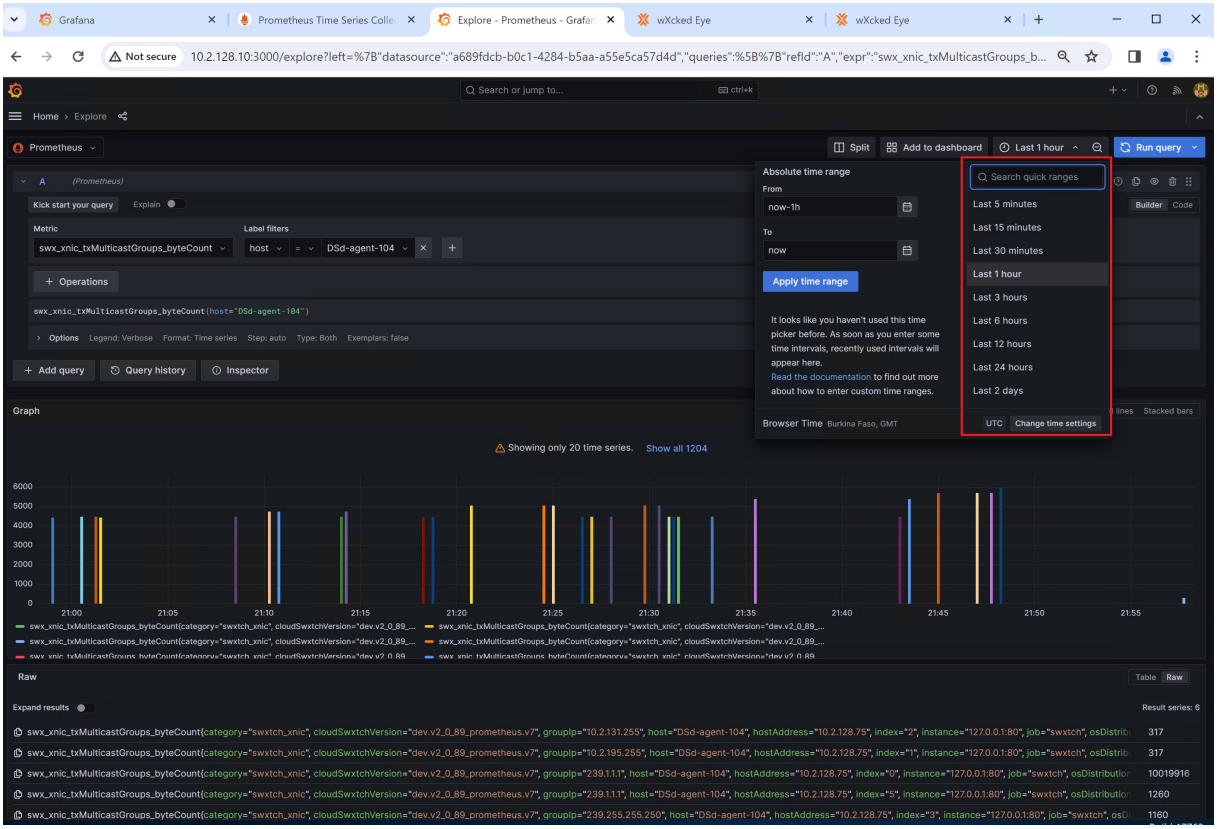


For xNIC-related metrics, users can specify a host under Label Filters and select an agent. After selecting an agent, a user will need to run the query again to populate the graph as shown above.



If desired, a user can also change the amount of time in the top of the window.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Testing cloudSwXtch

WHAT TO EXPECT

In this article, users will learn about the various tools that are available to them test their cloudSwXtch multicast network.

cloudSwXtch Testing Tools

Included within the xNIC installation are four utilities that can be used to verify both the functionality and performance of their cloudSwXtch network.

- **swxtch-perf**: Simulates traffic movement throughout the cloudSwXtch network assigning machines with the xNIC installed as producers and consumers
- **swxtch-top**: Displays detailed system statistics from the console of any VM that has the xNIC software installed. This includes data regarding the xNIC, the cloudSwXtch Bridge, streams, mesh, and high availability.
- **swxtch-tcpdump**: As our version of tcpdump, this tool helps capture multicast packets sent to and from the cloudSwXtch with logic to decode our own head and display the original MC payload.
- **swxtch-where**: Allows users to call for hardware information regarding their cloudSwXtch VM

Each of the utilities above can be run from a VM which has the xNIC software installed. Detailed information can be found for each by entering in the **--help** command-line argument.

wXcked Eye for Monitoring

In addition to viewing cloudSwXtch statistics in swxtch-top, users can also use wXcked Eye from the web browser of a VM in the cloudSwXtch network. For more information, see **Using wXcked Eye for cloudSwXtch**.

Universal Third-Party Test Tools

There are a number of universal third-party tools available to test the fidelity of your cloudSwXtch network. For more information on these alternative tools, see our Universal Third-Party Test Tools article.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

swxtch-perf

Overview

To simulate traffic movement throughout the cloudswxtch overlay network you can use swxtch-perf to create producer and consumers on machines with the xNIC installed.

swxtch-perf producer has multiple parameters that can be configured to generate different traffic flows. There can be multiple instances of swxtch-perf generating traffic on a single machine.

None	Copy

```
swxtch-perf producer --sendto <MC_ADDRESS:DEST_PORT> --nic
<NETWORK_INTERFACE>
```

swxtch-perf consumer will pick up the traffic generated by the producer(s) in the network.

None	Copy

```
swxtch-perf consumer --recvfrom <MC_ADDRESS:DEST_PORT> --nic
<NETWORK_INTERFACE>
```

NOTE

`<MC_ADDRESS>` = Multicast Address
`<DEST_PORT>` = Destination Port
`<NETWORK_INTERFACE>` = Network Interface where xNIC conncted to. The network interface does not have to be specified in xNic V1, but must be specified in xNic V2. (See xNIC Linux Installation for V1 and V2 differences.

swxtch-perf

For a quick view at the functionality and usage of **swxtch-perf** use `-h` or `-help`.

None	Copy

```
swxtch-perf -h
Usage: swxtch-perf [options] command

Positional arguments:
command                 [producer|consumer] suported commands
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
Optional arguments:
-h --help               shows help message and exits [default:
false]
-v --version            prints version information and exits
[default: false]
--nic                   name of NIC to use this is Mandatory for
swxtch-perf to work.
--recvfrom              IP:Port The IP and Port where packets come
from [default: "239.5.69.2:10000"]
--sendto                IP:Port The IP and Port where packets are
sent to [default: "239.5.69.2:10000"]
  --ssm_include           (consumer command only) List of SSM
addresses to include (i.e. 192.168.2.1 193.168.2.4) [nargs: 1 or
more]
  --ssm_exclude           (consumer command only) List of SSM
addresses to exclude (i.e. 192.168.2.1 193.168.2.4) [nargs: 1 or
more]
--payload_length        (producer command only) number of bytes for
the multicast udp payload [default: 100]
--total_pkts            Total packets to send/receive. To run
without this limit use 0 [default: 0]
--pps                   (producer command only) packet-rate or
packet per seconds [default: 1]
--seconds               Number of seconds to run the application. To
run without this limit use 0 [default: 0]
--loopback              Receives packets from --recvfrom and sends
packets to --sendto [default: false]
--generic               (consumer command only) to consume generic
packets [default: false]
--latency               Enables timestamp propagation and
measurement of latency [default: false]
--broadcast             Enables broadcast packets in NIC, this
overrides IP argument [default: false]
--generic-broadcast     Sends broadcast packets to 255.255.255.255,
valid only with --broadcast argument [default: false]
--broadcast-port        Port for broadcast traffic, valid only with
--broadcast argument [default: 10000]
--rtt-latency           Enables timestamp propagation and
measurement of RTT/2 where RTT = round trip time [default: false]
--one-way-latency       Enables timestamp propagation and
measurement of one way latency [default: false]
--latency-buckets       Enables histogram of latency. Use with --
latency [default: false]
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
--dbg                    Enables more information in the logs
[default: false]
--show-full-packet-bps   Shows the bps with all headers included
```

Parameters

Argument	Description	Default Value	Valid Range	Machine Type	Operating System
`h`	Shows commands that are available.				All
`v`	Shows version.			Both	All
`nic`	Specify which network interface xNIC will listen to this command is Mandatory.		--	Both	All
`recvfrom`	Specify the multicast group and port to listen for packets IPv4 addresses are valid; Ports: 1024 <= x <= 65535. Mandatory for Consumer Mode and Multicast.			Consumer	All
`sendto`	Specify the multicast group and port to send packets, mandatory for producer if using multicast.	All	IPv4 addresses are valid; Ports: 1024 <= x <= 65535 Mandatory for Producer Mode and Multicast.	Producer	All
`ssm_include`	List of SSM addresses to include (i.e. 192.168.2.1 193.168.2.4)		1 or more	Consumer	All
`ssm_exclude`	List of SSM addresses to exclude (i.e. 192.168.2.1 193.168.2.4)		1 or more	Consumer	All
`payload_length`	Number of bytes per packet.	100	8 and 65475	Producer	All
`total_pkts`	Number packets to receive or send before exiting iperf.	0	8 and 3750	Producer	Windows
`pps`	packet-rate or packets per second.	1	100000	Producer	All
`seconds`	Number of seconds to run the application, use 0 to run without a limit.	0		Both	Windows

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

`loopback`	Receives packets from recvfrom and sends packets to sendto.	false	true:false	Both	All
`generic`	Consume generic packets.	false	true:false	Consumer	All
`latency`	Enables timstamp propagation and measurement of latency.	false	true:false	Both	Linux
`broadcast`	Sets swxtch-perf to use normal broadcast mode, when sending it will use the IP of the --nic argument.	false	true:false	Both	All
`generic-broadcast`	Sets iperf to use broadcast mode using the IP of 255.255.255.255.	false	true:false	Both	All
`broadcast-port`	Sets port to be used for broadcast, and is only valid with --broadcast and --generic-broadcast arguement and is Mandatory for --broadcast --generic-broadcast .		Ports: 1024 <= x <= 65535	Both	All
`rtt-latency`	Enables timestamp propagation and measurement of RTT/2 where RTT = round trip time	false			Windows
`one-way-latency`	Enables timestamp propagation and measure of one way latency	false			Windows
`latency-buckets`	Enables histogram of latency. Use with --latency	false			Windows
`dbg`	Enables more information in the logs	false			All
`show-full-packet-bps`	Shows the bps with all headers included			Both	All

Multicast - Example

These examples can be run from one machine or across multiple machines. Parameters for NIC names assume default installation options.

EXAMPLE

Single Producer, Single Consumer, and one multicast group

Run this command on a VM to create a multicast group on the address `230.1.1.1` and port `3490` :

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

None Copy

```
Linux:
swxtch-perf producer --sendto 239.1.1.1:3490 --pps 1000 --nic
swxtch-tun0
Windows:
swxtch-perf producer --sendto 239.1.1.1:3490 --pps 1000 --nic
swxtch-tun
```

Example with results:

None Copy

```
swxtch-perf producer --sendto 239.1.1.1:3490 --pps 1000 --nic
swxtch-tun0
Trying to reach a packet-rate of 1000 pps
swxtch-perf producer threads started...  Ctrl+C to exit.
|-------------------------------------|-----------------------------
--|
|                 TOTALS              |         THIS PERIOD
|
|    TX PKTS    | TX BYTES | TX DROPS |  TX-PPS  |  TX-bps  | TX-DPS
|
|---------------|----------|----------|----------|----------|-------
--|
|         1,283 |    128KB |        0 |    1.28K |  1.0Mbps |
0 |
|         2,274 |    227KB |        0 |      991 |  792Kbps |
0 |
|         3,267 |    326KB |        0 |      993 |  794Kbps |
0 |
|         4,262 |    426KB |        0 |      995 |  796Kbps |
0 |
```

Run this command on one of the VMs to listen to traffic on the Multicast Address `230.1.1.1` port `13490` :

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

None Copy

```
Linux:
swxtch-perf consumer --recvfrom 239.1.1.1:3490 --nic swxtch-tun0
Windows:
swxtch-perf consumer --recvfrom 239.1.1.1:3490 --nic swxtch-tun
```

Example with results:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

None Copy

```
testadmin@DSd-agent-102:~$ swxtch-perf consumer --recvfrom
239.1.1.1:3490 --nic swxtch-tun0
swxtch-perf consumer threads started...  Ctrl+C to exit.
|-------------------------------------|-----------------------------
--|
|                 TOTALS              |         THIS PERIOD
|
|    RX PKTS    | RX BYTES | RX DROPS |  RX-PPS  |  RX-bps  | RX-DPS
|
|---------------|----------|----------|----------|----------|-------
--|
|             0 |       0B |        0 |        0 |     0bps |
0 |
|             0 |       0B |        0 |        0 |     0bps |
0 |
|             0 |       0B |        0 |        0 |     0bps |
0 |
|           330 |  33.00KB |        0 |      330 |  264Kbps |
0 |
|         1,326 |    132KB |        0 |      996 |  796Kbps |
0 |
|         2,328 |    232KB |        0 |    1.00K |  801Kbps |
0 |
|         3,330 |    333KB |        0 |    1.00K |  801Kbps |
0 |
|         4,332 |    433KB |        0 |    1.00K |  801Kbps |
0 |
|         5,328 |    532KB |        0 |      996 |  796Kbps |
0 |
|         6,330 |    633KB |        0 |    1.00K |  801Kbps |
0 |
```

- To add more consumers you simply run the same swxtch-perf command on new VMs.

Broadcast - Example

These examples can be run from one machine or across multiple machines. Parameters for NIC names assume default installation options.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

EXAMPLE

Single Producer, Single Consumer, and broadcast

Run this command on a VM to create a broadcast

None Copy

```
Linux:
swxtch-perf producer --broadcast --nic eth1 --pps 1000 --broadcast-
port 1234
Windows:
swxtch-perf producer --broadcast --nic 'Ethernet 2' --pps 1000 --
broadcast-port 1234
```

Example with results:

None Copy

```
PS C:\Users\testadmin> swxtch-perf producer --broadcast --nic
'Ethernet 2' --pps 1000 --broadcast-port 1234
Config:
        Sending traffic to  broadcast address.
        Ip Address: 10.2.195.255
        Port      : 10000
        Interface IP Address: 45
        Running without a total packet counter limit
        Running the application without a timing limit
Sent 972 total packets, throughput: 890.383 pkts/sec
Sent 2047 total packets, throughput: 993.128 pkts/sec
Sent 3123 total packets, throughput: 991.82 pkts/sec
Sent 4198 total packets, throughput: 990.419 pkts/sec
```

Run this command on one of the VMs to listen for broadcast

None Copy

```
Linux:
swxtch-perf consumer --broadcast --nic eth1--pps 1000
Windows:
swxtch-perf consumer --broadcast --nic 'Ethernet 3' --pps 1000
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

swxtch-top

WHAT TO EXPECT

swxtch-top is one of the utility applications included in xNIC installation. It can be run from the console of any VM that has the **xNIC** software installed, displaying real-time statistics of an attached cloudSwXtch instance. This includes data regarding mesh, high availability, multicast and PTP.

In this article, you will learn how to navigate through the different pages in swxtch-top and get better visibility on how data flows in your cloudSwXtch instance.

Running swxtch-top

Depending on your operating system, you can use certain commands to run swxtch-top on your VM.

For both Windows and Linux agents (xNICs), users can enter the following into the terminal:

Bash Copy

```
swxtch-top --swxtch <cloudSwXtch-IP>
```

Example:

Bash Copy

```
swxtch-top --swxtch 10.5.1.6
```

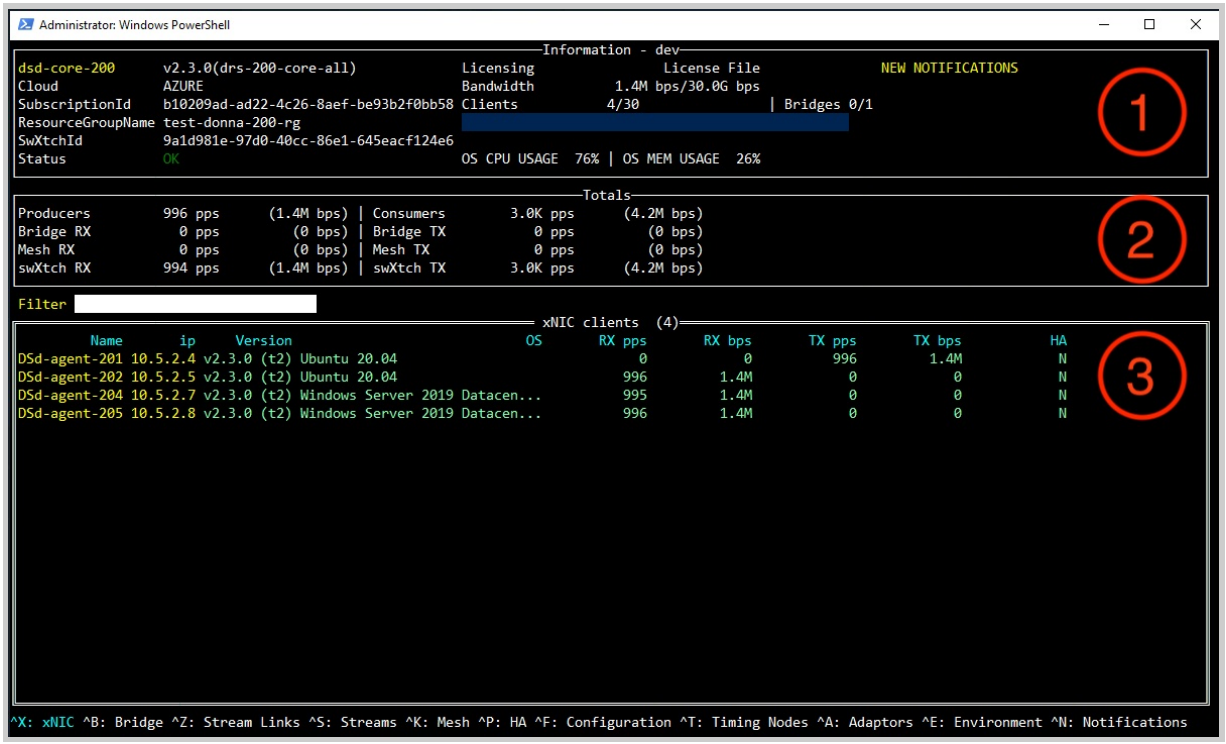
From the cloudSwXtch, users can enter the following command:

Bash Copy

```
sudo /swxtch/swxtch-top --swxtch localhost
```

Navigating swxtch-top Dashboard

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



The swxtch-top dashboard is organized into 3 panels as shown in the screenshot above. While the top 2 panels remain static, the third panel will change depending on the selected view. The swxtch-top dashboard has 11 different views:

1. ^X: xNIC
2. ^B: Bridge
3. ^Z: Stream Links
4. ^S: Streams
5. ^K: Mesh
6. ^P: HA
7. ^F: Configuration
8. ^T: Timing Nodes
9. ^A: Adaptors
10. ^D: Environment
11. ^N: Notifications

The default view is ^X: xNIC. To switch between views, **press CTRL + the associated letter**. For example, to toggle to "^D: Environment," enter CTRL + D.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

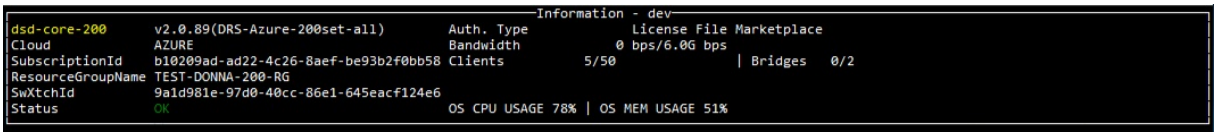
PLEASE NOTE

The following screenshots have been taken on the latest version of cloudSwXtch. To learn how to upgrade your cloudSwXtch, please see the article, **Upgrading cloudSwXtch**.

Panel 1: Information

The first panel of the swxtch-top dashboard provides users with information regarding their cloudSwXtch as well as their subscription plan. In the screenshot above, the cloudSwXtch is running on Azure. Each cloud provider will have alternative titles for some of the listed items but for the most part, the information is the same.

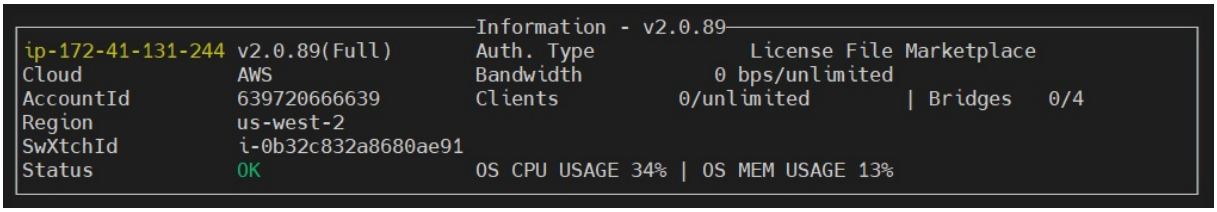
Azure



On the left side of the section, users will be able to read the name given to their cloudSwXtch, when it was instantiated as well as the cloud provider (Azure), version, cloud Subscription ID, ResourceGroupName, SwXtchID and Status.

On the right side, users can see the Authorization Type based on their cloudSwXtch license and the max bandwidth, clients, and bridges associated with that plan. The operating system's CPU and Memory usage is also displayed. For more information regarding licensing, please read the cloudSwXtch System Requirements article.

AWS

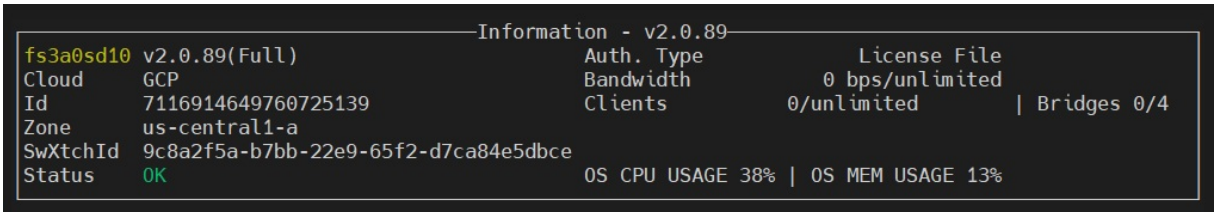


On the left side of the section, users will be able to read the name given to their cloudSwXtch with the version and the cloud provider (AWS). In addition, they can find the AccountID, Region, SwXtchID and cloudSwXtch status.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

On the right side, users can see the Authorization Type based on their cloudSwXtch license and the max bandwidth, clients, and bridges associated with that plan. The operating system's CPU and Memory usage is also displayed. For more information regarding licensing, please read the cloudSwXtch System Requirements article.

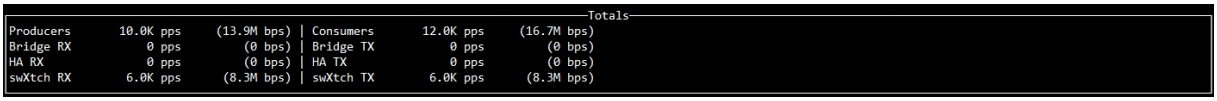
GCP



On the left side of the section, users will be able to read the name given to their cloudSwXtch with the version and the cloud provider (GCP). In addition, they can find the ID, Zone, SwXtchID and cloudSwXtch status.

On the right side, users can see the Authorization Type based on their cloudSwXtch license and the max bandwidth, clients, and bridges associated with that plan. The operating system's CPU and Memory usage is also displayed. For more information regarding licensing, please read the cloudSwXtch System Requirements article.

Panel 2: Totals



Panel 2 breaks down the statistics regarding data flow to and from the cloudSwXtch. Both the ingress and egress bandwidth will be displayed in both **packets per seconds (pps)** and **bits per second (bps**).

Please note: If your cloudSwXtch is part of a mesh or a bridge, the ingress/egress will show data in those sections.

- **Producers -** The total egress for all producers connected to the cloudSwXtch.
- **Consumers -** The total ingress for all consumers connected to the cloudSwxtch.
- **Bridge RX -** The total egress for the bridge that is connected to the cloudSwXtch (Ground-->Cloud).
- **Bridge TX -** The total ingress for the bridge that is connected to the cloudSwXtch (Cloud-->Ground).
- **HA RX -** The total egress for the entire high availability configuration of cloudSwXtches.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **HA TX -** The total ingress for the entire high availability configuration of cloudSwXtches.
- **swXtch RX -** The total ingress that the cloudSwXtch is receiving.
- **swXtch TX -** The total egress that the cloudSwXtch is transmitting.

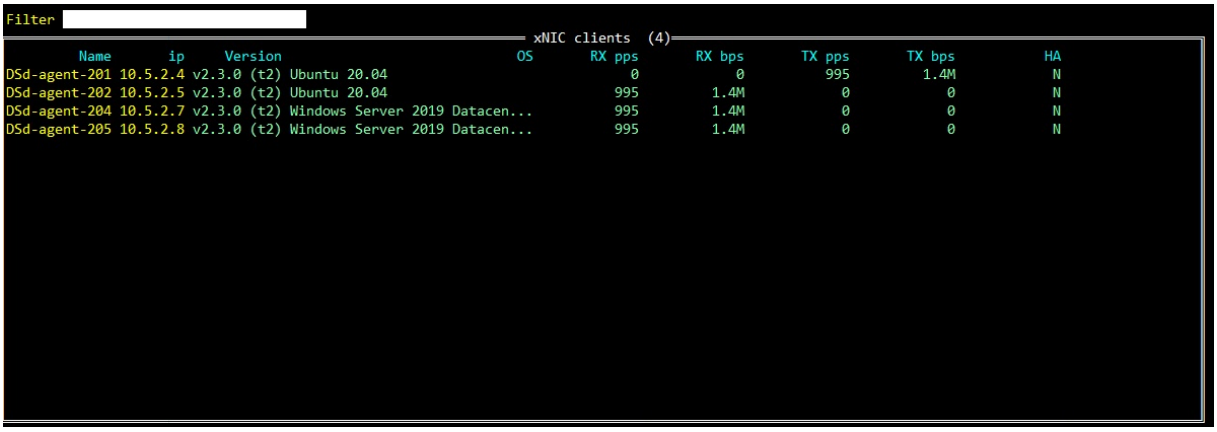
Panel 3: Views

Panel 3 defaults to "1: xNIC view" and is shown in the picture above. However, the display changes based on the selections at the bottom of the screen. To change views, key in
CTRL + the associated letter for that view.

Filter Bar

At the top of the xNIC, Streams and Notifications view panels, there is a white search bar. This allows users to filter through the contents the page. Simply type in the information you would like to filter the list by and hit Enter.

^X: xNIC view



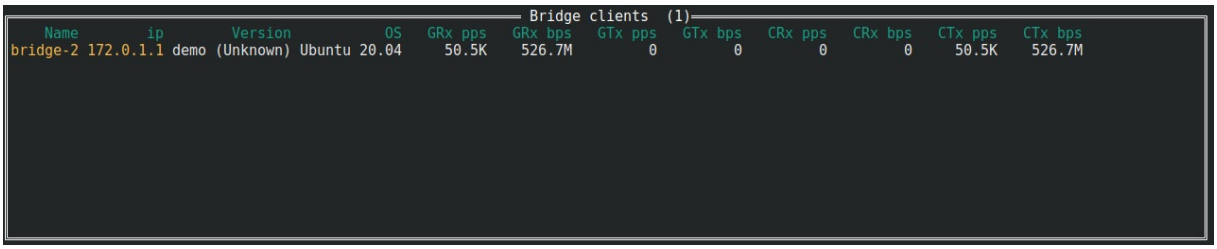
This view shows all the xNIC clients that are connected to the cloudSwXtch. To navigate to this view, press CTRL + N. This view includes:

- **Name -** Name of the Virtual Machine (Azure) or the HostName (AWS)
- **IP -** The IP of the data plane of the Virtual Machine.
- **Version -** The version of the xNIC. This value should match the cloudSwXtch's version.
- **XNIC -** The xNIC type: xNIC2 (T2) or xNIC1 (T1)
- **OS -** The operating system of the VM
- **RX pps -** The total ingress packets per second that the xNIC is receiving.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **RX bps -** The total ingress bits per second that the xNIC is receiving.
- **TX pps -** The total egress packets per second that the xNIC is transmitting.
- **TX bps -** The total egress bits per second that the xNIC is transmitting.
- **HA -** Whether the xNIC is configured for High Availability or not. This states how many cloudSwXtches are attached to this xNIC and if it is HA or not indicated by the -7. See also: High Availability Feature Description and High Availability Configuration

^B: Bridge



This view shows all the cloudSwXtch Bridges connected to the cloudSwXtch. It is divided into two parts: Bridge Information and Statistics. To navigate to this view, enter CTRL + B.

Bridge Information:

- **Name**: Name of the cloudSwXtch Bridge
- **IP**: IP address of the cloudSwXtch Bridge
- **Version**: cloudSwXtch Bridge version and type (t2 or t3)
- **OS**: cloudSwXtch Bridge Operating System

cloudSwXtch Bridge Statistics:

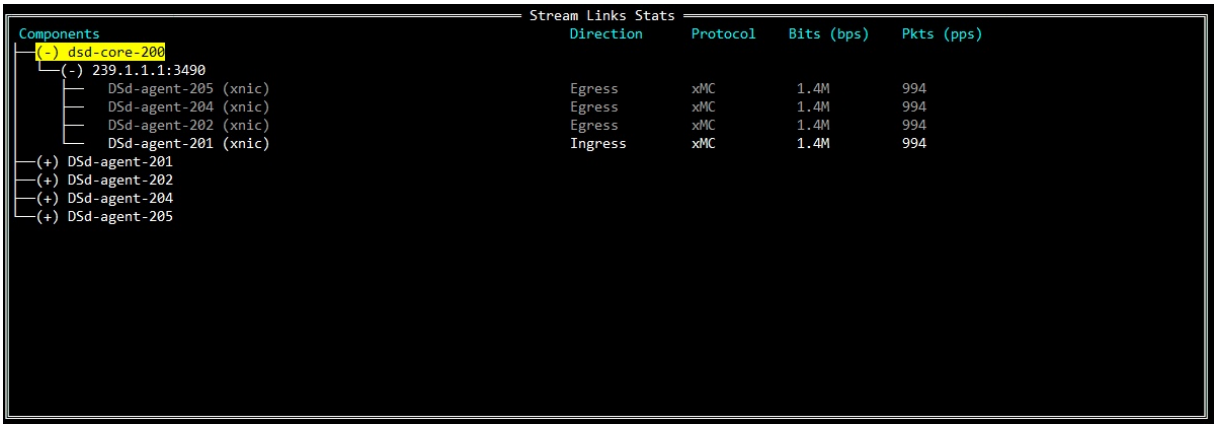
- **GRx pps** - Receiving traffic from the ground in packets per seconds
- **GRx bps** - Receiving traffic from the ground in bits per seconds
- **GTx pps** - Transmitting traffic to the ground from the cloudSwXtch in packets per seconds
- **GTx bps** - Transmitting traffic to the ground from the cloudSwXtch in packets per seconds
- **CRx pps** - Receiving traffic from the cloud in packets per seconds
- **CRx bps** - Receiving traffic from the cloud in bits per second
- **CTx pps** - Transmitting traffic to the cloud in packets per seconds

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **CTx bps** - Transmitting traffic to the cloud in bits per seconds

Think of these statistics from the perspective of the cloudSwXtch Bridge. For example, in the screenshot above, the cloudSwXtch Bridge is receiving traffic from the ground (**GRx**) and transmitting it to the cloud (**CTx**). That is why those columns are the only ones populated.

^Z: Stream Links



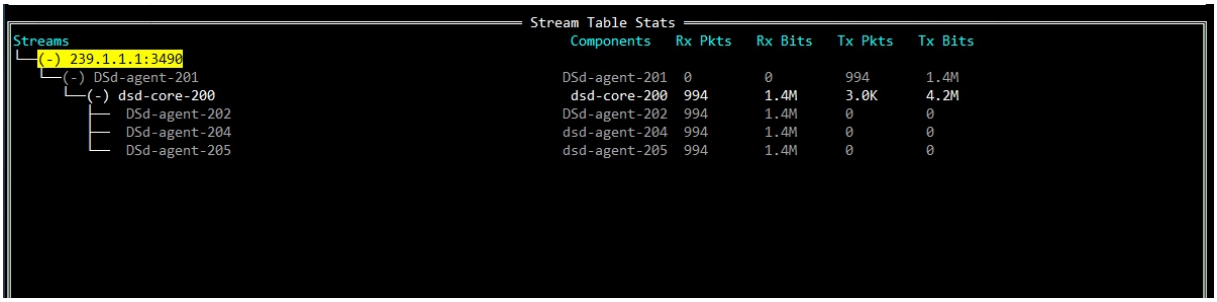
The Stream Links view displays all the cloudSwXtch network components (cloudSwXtch or xNIC) and the streams they are linked to. **To select this view**, press CTRL + Z. To expand a component, simply use the up and down keys on the keyboard and hit ENTER to select.

In this example, the cloudSwXtch (dsd-core-200) is selected with the singular stream 239.1.1.1:3490 expanded. Here, a user can see a list of endpoints receiving/transmitting the stream to/from the cloudSwXtch, dsd-core-200.

- **Components**: cloudSwXtch or endpoints (xNICs)
- **Direction**: Either Ingress or Egress, depending on whether data is being ingested or transmitted by the component
- **Protocol**: Multicast or Broadcast
- **Bits (bps)**: Bits per second
- **Pkts (pps)**: Packets per second

^S: Streams

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



This tree view shows all the multicast groups that are being received and/or transmitted by the cloudSwXtch and how to interacts with the various endpoints. **To navigate to this view**, enter CTRL + S.

This view includes:

- **Stream -** The name of the stream IP and Port. This could also be an Alias set in wXcked Eye. For example, BaseballCamera1 and BaseballCamera2 were Aliases assigned in wXcked Eye.
- **Components**: This could be the name of a cloudSwXtch or an agent (xNIC).
- **RX Pkts**: The total ingress packets per second being received by the component.
- **RX Bits**: The total ingress bits per second that is received by the component.
- **TX Pkts**: The total egress packets per second that is transmitted by the component.
- **TX Bits**: The total egress bits per second that that is transmitted by the component.

In the example above, stream 239.1.1.1: 3490 is listed. It is being transmitted by an endpoint (DSd-agent-201) to the cloudSwXtch (dsd-core-200), which is then send it to the other endpoints. That is why DSd-agent-202, DSd-agent-204 and DSd-agent-205 are receiving packets.

^K: Mesh

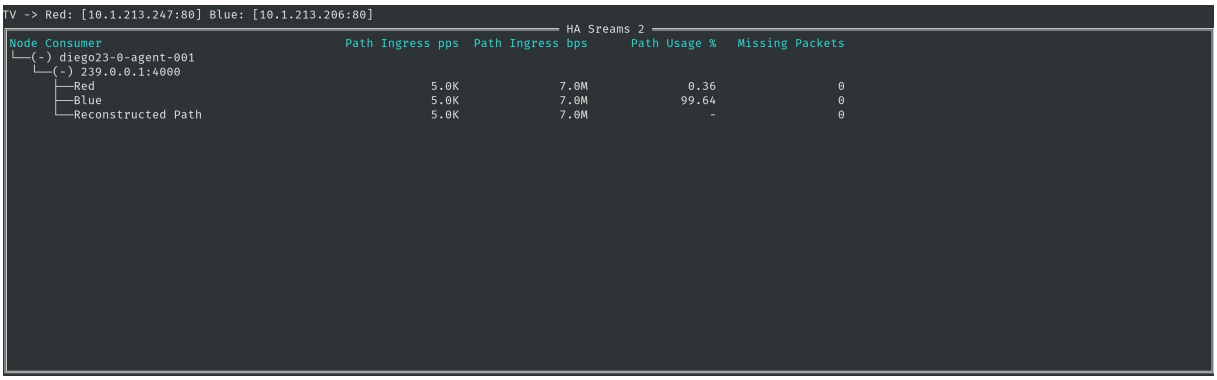
This view shows all the cloudSwXtches that are in a mesh. It only shows data if a mesh has been configured. To navigate to this view, enter CTRL + K. This view includes:

- **SwxtchAddress** - The IP address's of the cloudSwXtch(s) that is in the mesh with the cloudSwXtch that swxtch-top is set to.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **Gateway** - The IP address that serves as entry/exit point for traffic between networks.

^D: HA



The HA view shows additional details for high availability as the perspective of the consumer. It will only show data if High Availability has been configured. To navigate to this view, enter CTRL + P. P stands for paths. High Availability for configuration details.

This view is organized into two sections: **Node Consumer** and **packet/bits statistics**.

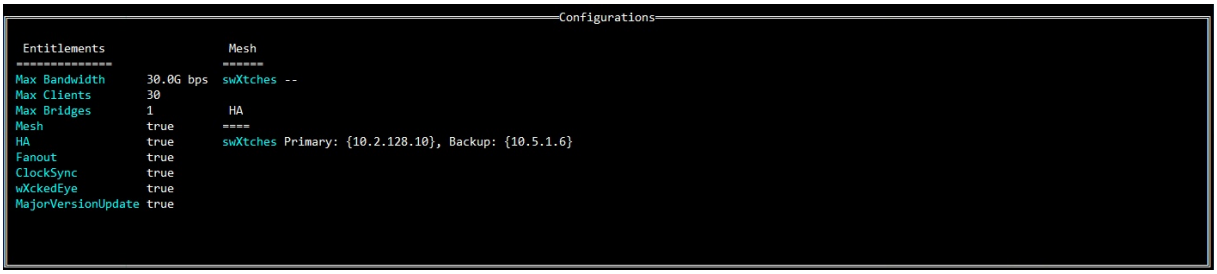
- **Node**: This is the component in the cloudSwXtch network that is consuming the HA paths. Expanding the consumer details will show the stream IP split into two paths (red and blue) and a reconstructed path.

The statistics by each line item is further explained below:

- **Path Ingress pps** - The total ingress packets per second that is received in the path for the multicast group.
- **Path Ingress bps** - The total ingress bits per second that is received in the path for the multicast group.
- **Path Usage %** - The percentage that the path is used in the highly available multicast group.
- **Missing packets** - The total number of missing packets for the path since the inception of the stream. If you stop the stream or cloudSwXtches, the number will stop increasing but will not reset.

^F: Configuration

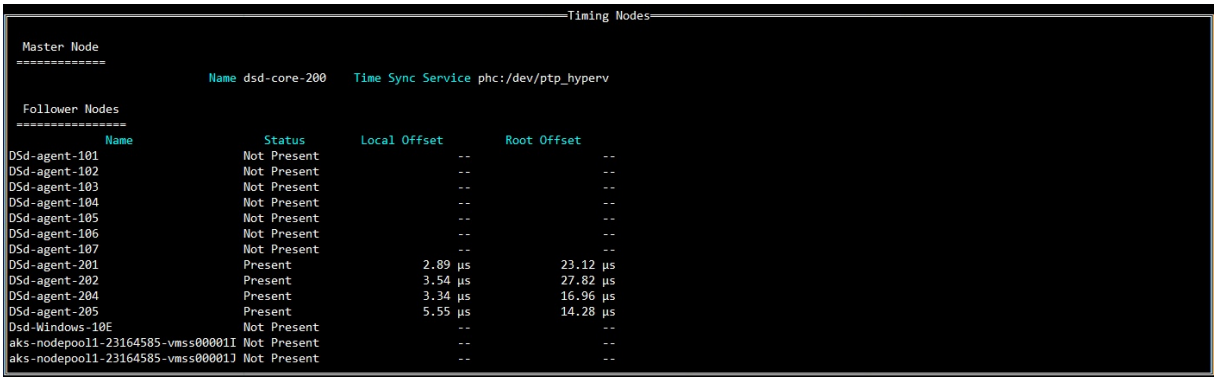
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



The Configuration view provides users with an expanded look at the licensing details found in the Information panel. In addition, they can see the cloudSwXtches connected in their mesh and HA configurations as well as details on their unicast. To navigate to this view, enter CTRL + F.

- **Entitlements**: Depending on their license, users will have a set number for their Max Bandwidth, Max Clients and Max bridges. In the example above, the user has a max bandwidth of 6 GBs with 50 clients max and 2 bridges max. This section will also show if a user has the following features enabled: Mesh, HA, Fanout, Clock Sync (PTP), wXckedEye, and Major Version Update.
- **Mesh**: This will list the IP addresses of the cloudSwXtches connected to a mesh.
- **HA**: This will list the Paths created for High Availability with each path showing the IP addresses of connected cloudSwXtches.

^T: Timing Nodes



The Timing Nodes view displays information regarding the clock sync configuration for the cloudSwXtch. The page in swxtch-top will only populate with information if the user has the PTP feature enabled. To navigate to this view, enter CTRL + T.

In the example above, the cloudSwXtch (dsd-core-200) is acting as the Master Node.

- **Master Node**- The Master Node is what the PTP configuration sets as the most reliable time source. This will send the true time it receives from the source clock to the Follower Nodes.
    - **Name** - The name of the cloudSwXtch
    - **Time Sync Service** - The source clock

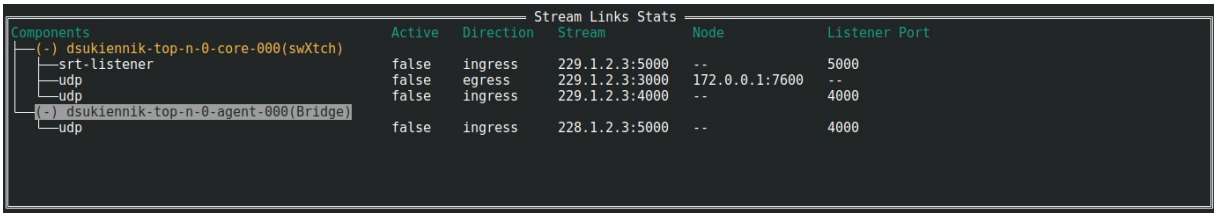
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **Follower Nodes**- The Follower Nodes lists the agents/VMs that subscribe to the Master Node for accurate timing.
    - **Name** - The name of the endpoints
    - **Status** - The status of the endpoints, noting if the node is active in the PTP configuration
    - **Local Offset** - The local offset denotes the offset in time from the cloudSwXtch to the xNIC.
    - **Root Offset** - The root offset denotes the offset in time from the GrandMaster clock to the cloudSwXtch and its follower nodes (xNIC). Note how the root is larger than the local. This is normal behavior since the distance between the follower node and the Grandmaster clock is greater than the offset between a cloudSwXtch and xNIC.

PTP Stabilization

After upgrading your cloudSwXtch system, you may notice that the local and root offset values are much larger than they actually are. It can take up to 30 minutes for the values to stabilize and return back to normal levels.

^A: Adaptors



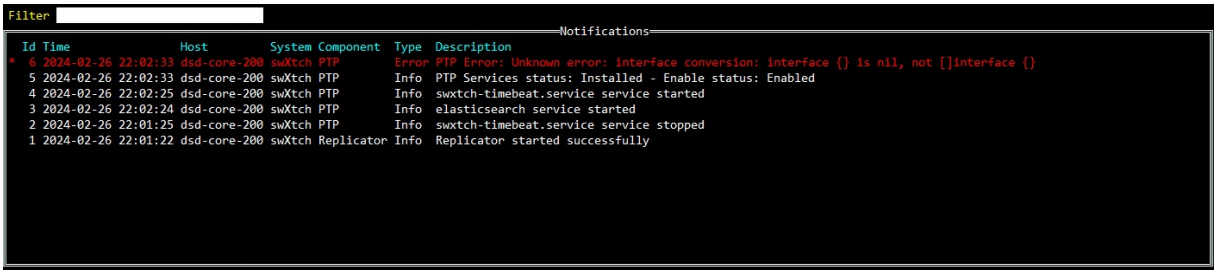
The Adaptors view displays information regarding Protocol Conversion and Fanout. It includes a detailed list of configured protocols (UDP, SRT Caller/Listener, or RIST Caller/Listener), their direction (ingress/egress), the stream, the node, and Listener Port. To navigate to this view, enter CTRL + A.

^E: Environment

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

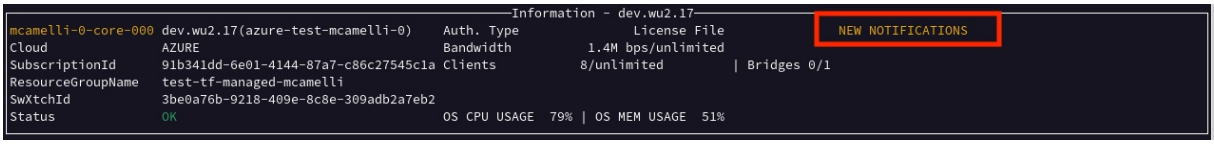
The Environment view displays information regarding the control and data subnets configured during cloudSwXtch instantiation. Both subnets are split into two sections: metadata and OS (Operating System). To navigate to this view, enter CTRL + E.

^N: Notifications



The Notifications view compiles a list of cloudSwXtch-related notifications. To navigate to this view, enter CTRL + H.

A **New Notifications** alert will appear in the Information panel in swXtch-top as shown below. Visiting the Notifications view will remove this alert.



Troubleshooting swxtch-top

1. **If the swxtch-top "Status" is showing that there is a "Connection error:"**

    a. Check that the cloudSwXtch is started.

    b. Check that you entered in the proper cloudswxtch name or IP when running the swxtch-top command.

    c. If name does not work when running the swxtch-top command then the DNS is not set-up correctly, use the IP address instead.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. **If an xNIC was installed but is not showing up in swxtch-top:**

    a. Navigate to the swxtch-nic.conf file and validate that the "SwxtchSvcAddr" is correct.

        - Windows can be found at "C:\Program Files\SwXtch.io\Swxtch-xNIC"
        - Linux can be found at "/var/opt/swxtch/swxtch-xnic.conf"

    b. Check that the firewall is open for the following ports:

subnet	protocol	ports	vm
ctrl-subnet	tcp	80	cloudSwXtch
ctrl-subnet	udp	10800-10803	all
data-subnet	udp	9999	all

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. **If a multicast group is not showing up then check that they have registered.**

    - **In Linux**, run this command:

      Text

Plaintext	Copy
`ip maddress show`	

    - **In Windows**, run this command in PowerShell:

      Text

Plaintext	Copy
`netsh.exe interface ipv4 show joins`	

    - If the joins are not showing here then the application is not joining the multi-cast group. In this case run swxtch-perf for the same IP:Port combination and then re-try in the program.

    - If the joins are not showing here then the application is not joining the multi-cast group. In this case run swxtch-perf for the same IP:Port combination and then re-try in the program.

    - If using Windows make use of Task Manager and view Performance to know where data is being sent/received.

    - Validate using TCPdump or Wireshark to identify where traffic is going as it could be going to the wrong network interface, it should be going to the Data Interface if xNIC2 and Swxtch-tun0 if xNIC1. An example is below:

Plaintext	Copy
`$ sudo tcpdump udp -X -i <interface>`	

NOTE

xNIC1 interface: `swxtch-tun0`
xNIC2 interface: data nic (usually `eth1 for Linux, and "Ethernet 2" for Windows`)

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- Validate that a firewall is not stopping the multicast and open up the firewall to include port exceptions.

swxtch-top on a cloudSwXtch

swxtch-top should be run from a virtual machine with an xNIC installed, it should be avoided to run it or anything else directly on a cloudSwXtch. That being said it can be done, but you must run it with sudo. Only run it on the cloudSwXtch if doing advanced troubleshooting.

sudo /swxtch/swxtch-top dashboard --swxtch localhost

Alternatively use 127.0.0.1 or swxtch-hostname or swxtch-IP in place of localhost

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

swxtch-tcpdump

WHAT TO EXPECT

Users can use a cloudSwXtch specific version of tcpdump called swxtch-tcpcump. This tool helps with capturing multicast packets sent to and from the cloudSwXtch. It is the same as tcpdump but with logic to decode our own header and display the original MC payload.

In this article, users will learn about the available arguments for swxtch-tcpdump.

Using swxtch-tcpdump

Execute the following command:

Bash | Copy

```
swxtch-tcpdump
```

Note: The default is **swxtch-tun (Windows)** or **swxtch-tun0 (Linux)**. If their multicast is running on a different interface, then a user will need to specify that interface. To get a list of interfaces for Windows, you can use **ip config**. For Linux, you can use **ip a**. After you get the name of the correct interface, you can use the -i argument followed by your desired interface name.

Example:

Bash | Copy

```
swxtch-tcpdump -i ens6
```

Additional arguments

Users can use the **-h** argument as shown below to get a list of available arguments for swXtch-tcpdump.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
ubuntu@ip-172-41-128-232:/var/opt$ swxtch-tcpdump -h
swxtch-tcpdump version 5.0.0-PRE-GIT
libpcap version 1.9.1 (with TPACKET_V3)
OpenSSL 1.1.1f  31 Mar 2020
Usage: swxtch-tcpdump [-AbdDefhHIJKlLnNOpqStuUvxX#] [ -B size ] [ -c
count ] [--count]
                [ -C file_size ] [ -E algo:secret ] [ -F file ] [ -G
seconds ]
                [ -i interface ] [ --immediate-mode ] [ -j
tstamptype ]
                [ -M secret ] [ --number ] [ --print ]
                [ --print-sampling nth ] [ -Q in|out|inout ] [ -r
file ]
                [ -s snaplen ] [ -T type ] [ --version ]
                [ -V file ] [ -w file ] [ -W filecount ] [ -y
datalinktype ]
                [ --time-stamp-precision precision ] [ --micro ] [ -
-nano ]
                [ -z postrotate-command ] [ -Z user ] [ expression ]
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

swxtch-where

WHAT TO EXPECT

swxtch-where allows users to call for hardware information regarding their cloudSwXtch VM.

In this article, users will learn about the different arguments they can use with swxtch-where and example outputs they should expect.

swXtch-where Cloud Type

Below are the Linux and Windows commands to call swxtch-where. An empty command (without an argument) like the examples below will only return the cloud type.

Windows

For Windows VM, swxtch-where must be called from the xNIC directory and have the .exe extension.

Bash | Copy

```
PS C:\Program Files\SwXtch.io\Swxtch-xNIC2> .\swxtch-where.exe
```

Linux

For Linux VM, a user would only need to input the following:

Bash | Copy

```
swxtch-where
```

swxtch-where Format

What is probably the most useful option in the swxtch-where argument list is **-f json** or **--format json**, which provides users with a json of hardware-related information regarding the cloudSwXtch VM. This is information is similar to the Hardware view in swxtch-top and Hardware panel in wXcked Eye's Settings' General tab. It presents a breakdown of the control and data subnet with information categorized as either metadata or operating system.

Windows

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
PS C:\Program Files\SwXtch.io\Swxtch-xNIC2> .\swxtch-where.exe --
format json
```

Linux

Bash Copy

```
swxtch-where -f json
```

Example Output in Windows:

Bash Copy

```
S C:\Program Files\SwXtch.io\Swxtch-xNIC2> .\swxtch-where.exe --
format json
{
  "cloudType": "AZURE",
  "nics": [
    {
      "computed": {
        "isPreferredControlNic": false,
        "isPreferredDataNic": true,
        "isSRIOV": false
      },
      "meta": {
        "ipAddress": "10.5.1.7",
        "ipBroadcast": "10.5.1.255",
        "ipSubnet": "10.5.1.0",
        "mac": "00:22:48:27:0e:dc",
        "subnetMask": "255.255.255.0"
      },
      "os": {
        "driver": "mlx5.sys",
        "mac": "00:22:48:27:0e:dc",
        "mtu": 1500,
        "name": "Ethernet 261",
        "pciAddress": "65025:00:02.0"
      }
    },
    {
      "computed": {
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
        "isPreferredControlNic": true,
        "isPreferredDataNic": false,
        "isSRIOV": false
      },
      "meta": {
        "ipAddress": "10.5.2.7",
        "ipBroadcast": "10.5.2.255",
        "ipSubnet": "10.5.2.0",
        "mac": "00:22:48:27:07:14",
        "subnetMask": "255.255.255.0"
      },
      "os": {
        "driver": "netvsc.sys",
        "ipAddress": "10.5.2.7",
        "ipBroadcast": "10.5.2.255",
        "ipSubnet": "10.5.2.0",
        "mac": "00:22:48:27:07:14",
        "mtu": 1500,
        "name": "Ethernet",
        "pciAddress": "",
        "subnetMask": "255.255.255.0"
      }
    }
  ]
}
```


swxtch-where Version

Using the **-v** or **--version** argument after the swxtch-where command will return the version.

Example in Windows:

Bash | Copy

```
PS C:\Program Files\SwXtch.io\Swxtch-xNIC2> .\swxtch-where.exe -v
1.0
```

swxtch-where Help

The swXtch-where -h or --help argument provides users with a detailed list of available arguments.

Example in Windows:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
PS C:\Program Files\SwXtch.io\Swxtch-xNIC2> .\swxtch-where.exe - h
Usage: C:\Program Files\SwXtch.io\Swxtch-xNIC2\swxtch-where.exe
[options]

swxtch-where utility.
Calling with no arguments simply returns the cloud type.


Optional arguments:
-h --help       shows help message and exits [default: false]
-v --version    prints version information and exits [default:
false]
-f --format     output format (example: "json")
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Universal Third-Party Testing Tools

WHAT TO EXPECT

While xNIC installation provides you with a number of useful tools to test the functionality of your cloudSwXtch network (swxtch-perf, swxtch-tcpdump, etc.), there is also a wealth of universal third-party testing tools at your disposal.

In this article, we will take a deeper dive into these alternative options and understand their benefits and pitfalls.

- VLC
- ffmpeg/ffplay
- iPerf (v2)
- sockperf
- Python/Go

Please Note: These tools should be used for testing purposes only.

VLC

VLC is a free and open-source cross-platform multimedia player and framework that plays most multimedia files, and various streaming protocols. As a highly visual tool, it can be used to demonstrate the delivery and fidelity of video streams from the cloudSwXtch to the xNIC.

Example

In the following example, a producer is streaming a .ts file using the RTP protocol to the multicast address 239.1.1.2 to port 1000 in a indefinite loop. The agent is consuming the stream.

Producer:

PowerShell | Copy

```
vlc file:///home/ubuntu/surfing.ts --sout
'#rtp{dst=239.1.1.2,port=10000,mux=ts}' --loop
```

Consumer:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

PowerShell Copy

```
vlc rtp://@239.1.1.2:10000
```

vlc vs. cvlc

When using Linux, you can use cvlc instead of vlc. cvlc is an alias for “vlc -I dummy”. It is VLC without the user interface, lighter and faster to open remotely, and will show a window playing the video only. **Note**: cvlc does not work for Windows.

ffmpeg/ffplay

ffmpeg/ffply are command line tools to demonstrate the delivery of a stream. **ffmpeg** is the streamer, **ffplay** is the command to see the stream. While VLC requires a file to stream from the producer to the consumer, ffmpeg/ffplay can produce video test patterns. In the following example, we will use "testsrc", which will generate a test video pattern. This will display a color pattern with a scrolling gradient and a timestamp. If a stream appears, it means this was a successful delivery. **Please note**: Other test patterns are available for use.

Example

In this example, the user is testing a stream signal with a resolution of 640×480 at 30fps, in MPEG using labfi (Libavfilter input virtual device) for 1000 seconds to the multicast address 239.1.1.1, port 10000 using a packet size of 1316 bytes.

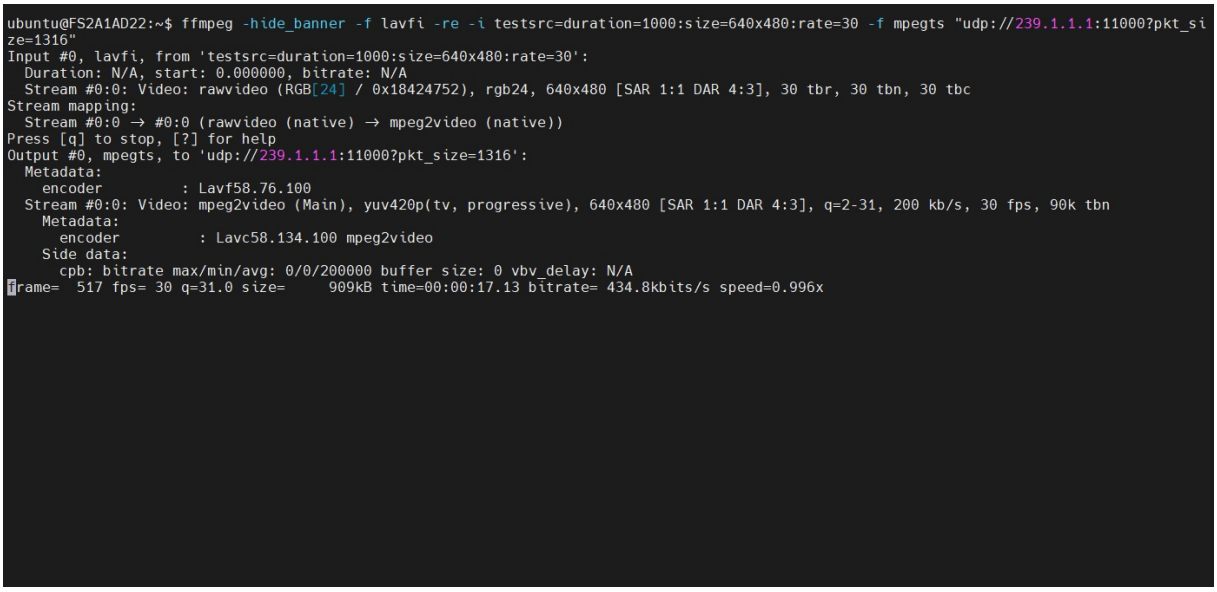
Producer

PowerShell Copy

```
ffmpeg -hide_banner -f lavfi -re -i
testsrc=duration=1000:size=640x480:rate=30 -f mpegts
"udp://239.1.1.1:11000?pkt_size=1316"
```

Below is an example of **ffmpeg producer output**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

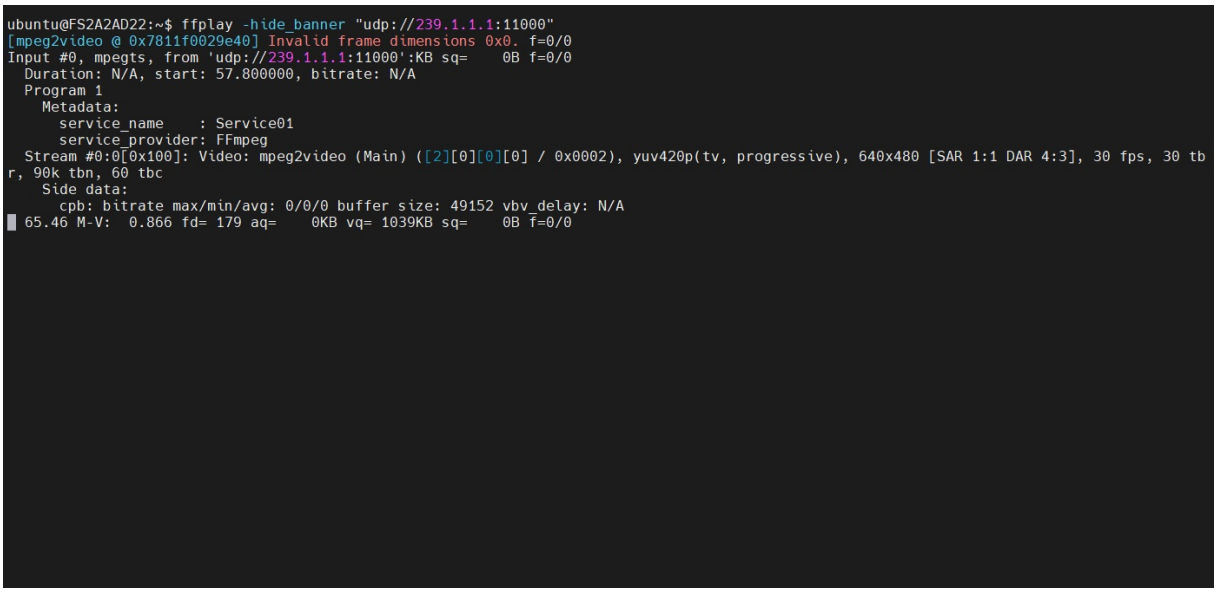


Consumer:

PowerShell Copy

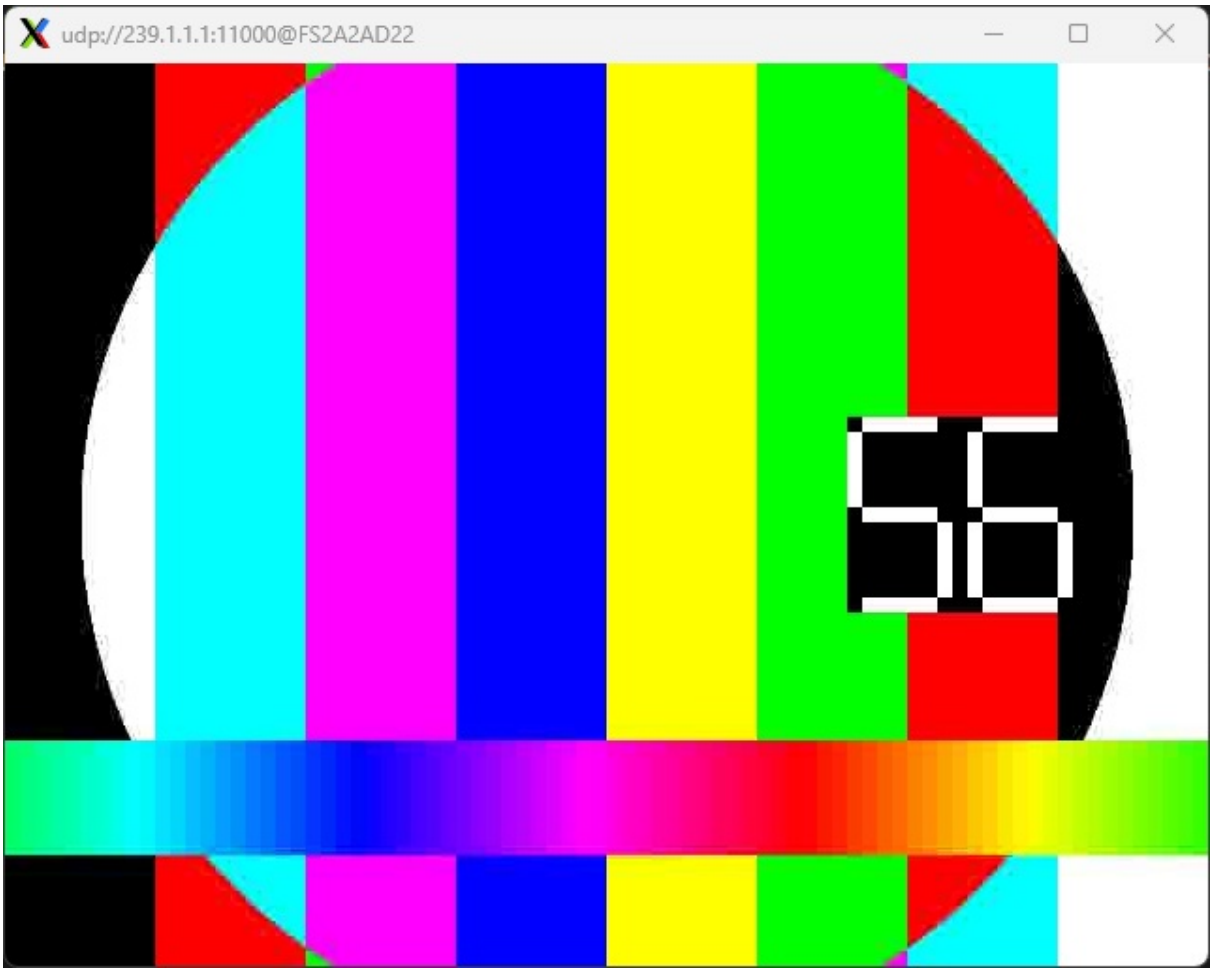
```
ffplay -hide_banner "udp://239.1..1:11000"
```

Below is an example ffplay consumer output.



A successful delivery of the stream will display the following stream in the consumer's VM:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



iPerf (v2)

Similar to swxtch-perf, iPerf is a multi-platform tool for network performance measurement and tuning. It is commonly used for bandwidth and latency measurements. However, what differs is that iPerf has additional arguments not found in swxtch-perf, our streamlined tool. Since iPerf (v3) does not support multicast, it is recommended to use iPerf (v2) for cloudSwXtch networking testing.

Example

In this example, the user is creating a UDP stream at multicast address 239.1.1.3, port 1000 for 120 seconds with a buffer length of 1000 bytes. The enhanced report will display stats for every second with enhanced reporting.

Producer:

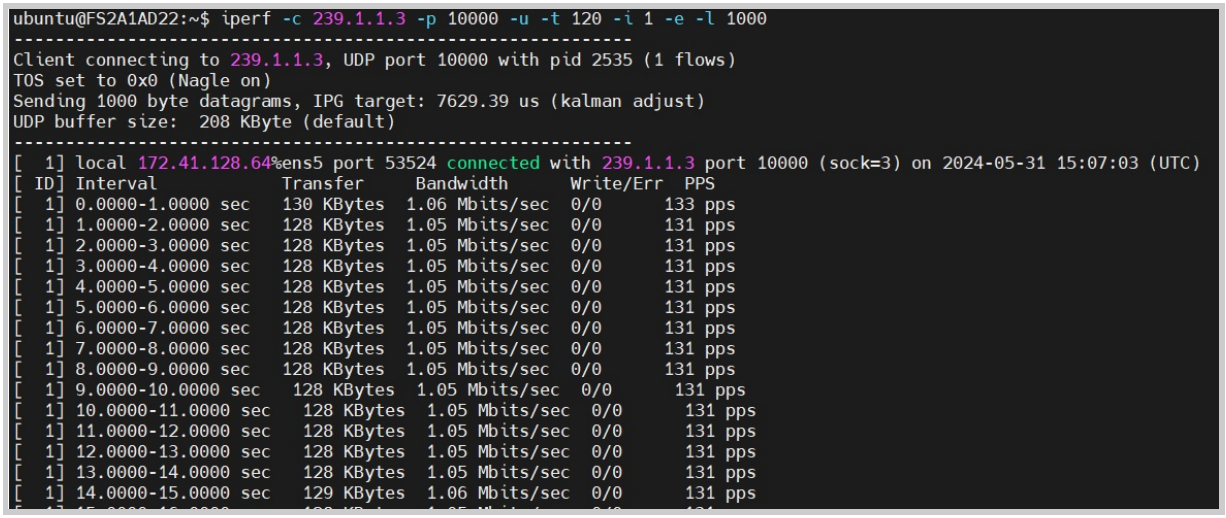
PowerShell Copy

```
iperf -c 239.1.1.3 -p 10000 -u -t 120 -i 1 -e -l 1000
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **Arguments Explained:**
    - **-c**: the machine as the producer, specifying the multicast address
    - **-p**: the port
    - **-u**: the producer will be sending UDP packets
    - **-t**: the duration of the stream in seconds
    - **-i**: the time interval stats will be reported. In this case, it is every second (1).
    - **-e**: enhanced reporting (interval, transfer, bandwidth, write/err, PPS)
    - **-l**: length of buffers to read or write (1000)

This is an example of an iPerf producer output.



Consumer:

PowerShell	Copy

```
iperf -s -u -b 239.1.1.3 -p 10000 -i 1
```

- **Arguments Explained:**
    - **-s**: server, signifying a consumer
    - **-u**: consumer will be taking in UDP packets
    - **-b**: bind, specifying the multicast address
    - **-p**: the port
    - **-i**: the time interval stats will be reported (1)

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

This is an example of an iPerf consumer output. Note how each line item coincides with the single second interval.

```
ubuntu@FS2A2AD22:~$ ./iperf -s -u -B 239.1.1.3 -p 10000 -i 1
------------------------------------------------------------
Server listening on UDP port 10000
Joining multicast group  239.1.1.3
Server set to single client traffic mode (per multicast receive)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 239.1.1.3 port 10000 connected with 172.41.128.64 port 56156
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.00-1.00 sec   129 KBytes  1.06 Mbits/sec   0.017 ms 7001/7133 (98%)
[  1] 1.00-2.00 sec   128 KBytes  1.05 Mbits/sec   0.014 ms 0/131 (0%)
[  1] 2.00-3.00 sec   128 KBytes  1.05 Mbits/sec   0.010 ms 0/131 (0%)
[  1] 3.00-4.00 sec   128 KBytes  1.05 Mbits/sec   0.016 ms 0/131 (0%)
[  1] 4.00-5.00 sec   128 KBytes  1.05 Mbits/sec   0.015 ms 0/131 (0%)
[  1] 5.00-6.00 sec   129 KBytes  1.06 Mbits/sec   0.017 ms 0/132 (0%)
[  1] 6.00-7.00 sec   128 KBytes  1.05 Mbits/sec   0.014 ms 0/131 (0%)
[  1] 7.00-8.00 sec   128 KBytes  1.05 Mbits/sec   0.027 ms 0/131 (0%)
[  1] 8.00-9.00 sec   128 KBytes  1.05 Mbits/sec   0.015 ms 0/131 (0%)
[  1] 9.00-10.00 sec   128 KBytes  1.05 Mbits/sec   0.017 ms 0/131 (0%)
```

No Consumer Output

On some older versions of iPerf 2, the consumer will not show any output. This can happen even when configured to do so. However, you can still check swxtch-top to see if the xNIC interface is getting traffic. To remedy this issue, ensure you have the latest version of iPerf 2.

sockperf

Another alternative to swxtch-perf is **sockperf**, a Linux-only network benchmarking utility over socket API that is designed for testing the latency and throughput of high-performance systems. While it is similar to both swxtch-perf and iPerf, it has far less commands than its counterparts. It is recommended to use sockperf when saturating the NIC and using the VM's maximum amount of bandwidth.

Example

In this example, the user will create a stream at multicast address 239.1.1.4, port 10000 using a message size of 1472, sending 2000 messages per second in 30 seconds.

Producer (sockperf throughput):

PowerShell | Copy

```
sockperf throughput --ip 239.1.1.4 --msg-size 1472 --port 10000 --
mps 2000 --time 30
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Consumer (sockperf server):

PowerShell	Copy

```
sockperf server --ip 239.1.1.4 --Activity 800 --port 10000
```

This will consume the traffic and print the activity for the last 800 messages processed. The consumer will continually wait for the producer until it is shut off.

Python/Go

Users may choose to also build their own tool to test their cloudSwXtch network. Below are two examples using Python or Go programming language to test multicast messaging. What is great about cloudSwXtch is that when we say it requires no code changes, the same applies for testing.

Python

As a pre-installed program on Linux-based machines, using Python is a simple way to test your xNIC.

Producer

The key object here is the socket module and the instruction used to send is sock.sendto. As you can see in the code, no modification to the code is necessary in order to make “compatible” with the xNIC/cloudSwXtch. The following code will work on any multicast-compatible environment.

1. Create a **producer.py** file on the producer VM. One method is to use nano. **Note**: You can name the file anything.

Plaintext	Copy

```
nano python.py
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Copy and paste the following script:

Profile	Copy

```
import socket
import struct
import time

def send_multicast_message(ip, port, message, delay):
    # Create the socket
    multicast_group = (ip, port)
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Set the time-to-live for messages to 1 so they do not go past
the local network segment
    ttl = struct.pack('b', 1)
    sock.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, ttl)

    try:
        while True:
            # Send the message
            print(f'Sending "{message}" to {ip}:{port}')
            sock.sendto(message.encode('utf-8'), multicast_group)

            # Delay between messages
            time.sleep(delay)
    finally:
        print('Closing socket')
        sock.close()

# Example usage
send_multicast_message('239.1.1.1', 10000, 'Hello, Multicast!',
2)Save the file and close.
```

3. Save and close.

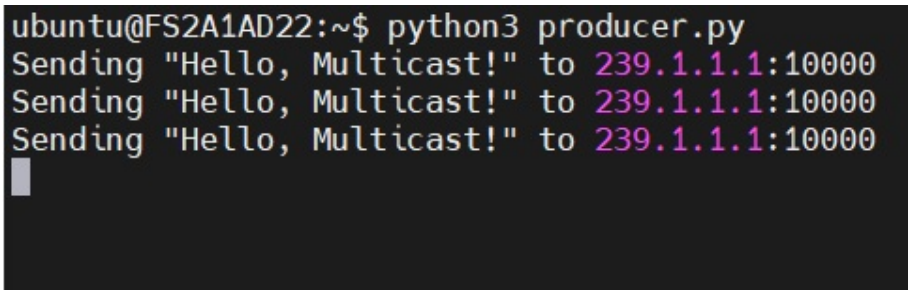
4. Run the script:

Profile	Copy

```
python3 producer.py
```

Below is an example of the output:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



Consumer

1. Create a **receiver.py** file on your consumer VM. One method is to use nano. **Note**: You can name the file anything.

Plaintext	Copy

```
nano consumer.py
```

2. Copy and paste the following script:

Profile	Copy

```
import socket
import struct

def receive_multicast_message(ip, port):
    # Create the socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Bind to the server address
    sock.bind(('', port))

    # Tell the operating system to add the socket to the multicast
group
    group = socket.inet_aton(ip)
    mreq = struct.pack('4sL', group, socket.INADDR_ANY)
    sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP,
mreq)

    # Receive/respond loop
    while True:
        print('Waiting to receive')
        data, address = sock.recvfrom(1024)

        print(f'Received "{data.decode("utf-8")}" from {address}')

# Example usage
receive_multicast_message('239.1.1.1', 10000)Save and close.
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

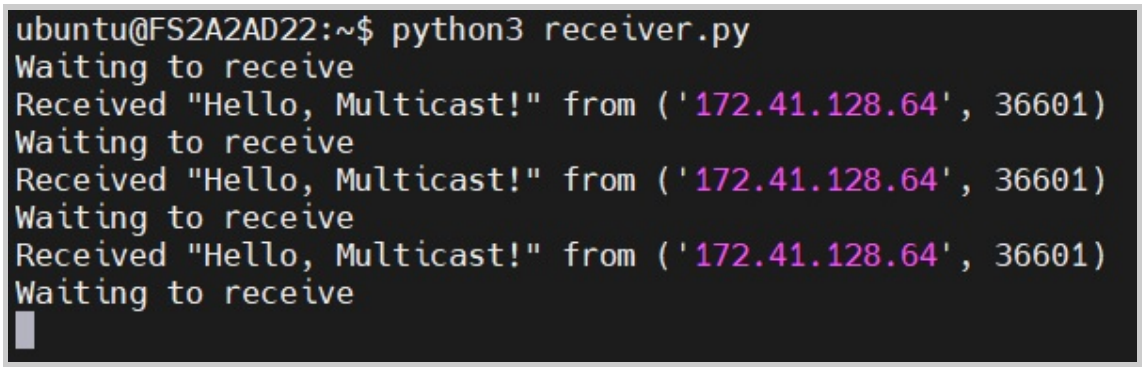
3. Save and exit.

4. Run the script:

PowerShell | Copy

```
python3 receiver.py
```

Below is an example of the output on the consumer VM:



Go

Go is an alternative programming language. However, unlike Python, it must be installed manually. Use the following example scripts to test.

Producer

The producer uses the command **net.Dial** to send a message using multicast.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Go Copy

```go
// producer.go
package main

import (
    "fmt"
    "net"
    "os"
    "time"
)

func main() {
    multicastAddr := "239.1.1.1:10000"

    conn, err := net.Dial("udp", multicastAddr)
    if err != nil {
        fmt.Println("Error connecting:", err)
        os.Exit(1)
    }
    defer conn.Close()

    for {
        message := "Hello, Multicast!"
        _, err = conn.Write([]byte(message))
        if err != nil {
            fmt.Println("Error sending message:", err)
            os.Exit(1)
        }
        fmt.Println("Message sent:", message)
        time.Sleep(2 * time.Second)
    }
}
```

Consumer

The receiver uses the command **net.ListenMulticastUDP**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Go | Copy

```go
// receiver.go
package main

import (
    "fmt"
    "net"
    "os"
)

func main() {
    multicastAddr := "239.1.1.1:10000"
    addr, err := net.ResolveUDPAddr("udp", multicastAddr)
    if err != nil {
        fmt.Println("Error resolving address:", err)
        os.Exit(1)
    }

    conn, err := net.ListenMulticastUDP("udp", nil, addr)
    if err != nil {
        fmt.Println("Error listening for multicast:", err)
        os.Exit(1)
    }
    defer conn.Close()

    buf := make([]byte, 1024)
    for {
        n, src, err := conn.ReadFromUDP(buf)
        if err != nil {
            fmt.Println("Error receiving message:", err)
            os.Exit(1)
        }
        message := string(buf[:n])
        fmt.Printf("Received message from %v: %s\n", src, message)
    }
}
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Other Programming Languages

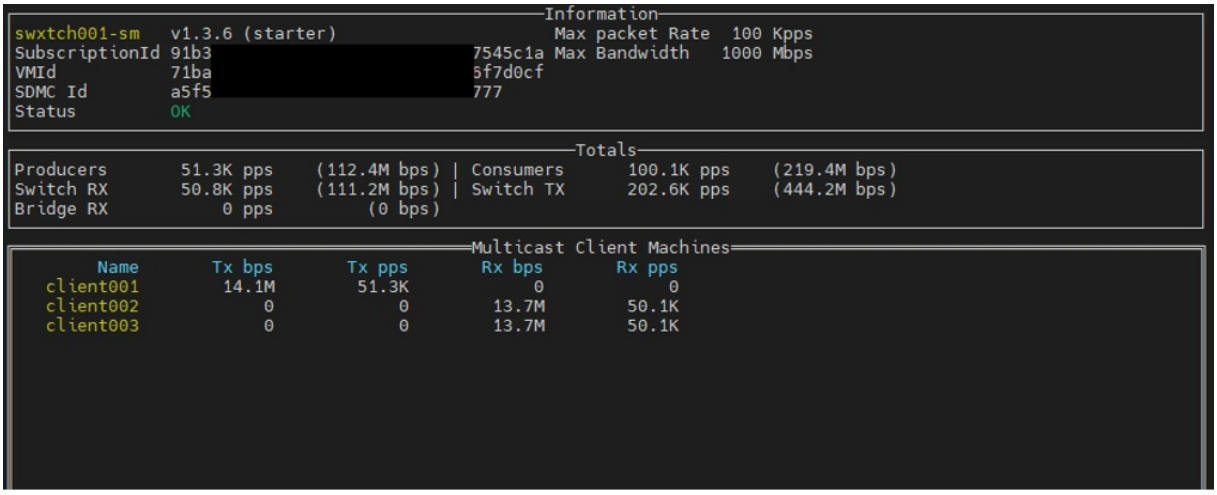
As with Python or Go, you can create programs in virtually any language with the same logic for sending multicast traffic and let the xNIC + cloudSwXtch take care of the rest. For example, in Javascript, the commands will be **server.send** and **server.on**. In C++, the commands will be **sendto()** and **recvfrom()**.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Troubleshooting

The swxtch-top program is the best way to quickly check system status. It can be run from any machine that has network access to the control subnet assigned to the switch instance. The swxtch-top program is automatically installed by the xNIC installer.

When run with no command line options, it connects to the switch instance associated with the local VM. There are command line arguments that allow you to specify the exact switch if more than one is reachable. Use the --help option for details.



Cannot ping the cloudSwXtch instance

If `ping <swxtch-instance-name>` fails, try directly pinging the IP address of the cloudSwXtch instance. If ping by IP address also fails, check to make sure that the VM from which you are running the ping command has its network configured properly: The host VM must have at least ***two NICs*** and the NICs must be on the ***same subnets*** for control and data as the SDMC switch.

Client machine doesn't show up in the switch list in swxtch-top

1. Verify that ping works from the client machine to the switch instance.
2. Check firewall settings (especially on RHEL). Remove any firewall restrictions to UDP ports 10800 and 9999. The cloudSwXtch sends UDP packets to these ports as part of normal operation.
3. Check xNIC log: `sudo journalctl -u swxtch-xnic`

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

How to View cloudSwXtch Logs for Troubleshooting

WHAT TO EXPECT

In this article, users will learn about accessing cloudSwXtch logs for the purpose of troubleshooting. We will break down two commands: **swx support** and **sudo journalctrl.** Common arguments that can be used when compiling information for the support team at swXtch.io will also be discussed.

Support may also request for you to send xNIC logs. For more information, see **How to Find xNIC Logs**.

swx support Logs

When troubleshooting, swXtch.io Support will request a report detailing the statistical data stored within the cloudSwXtch during a certain period. This report will include max highmarks, list highmarks, logs, swxtch Info, xNIC Logs, License and Config -- all in a compressed tar.gz file.

Accessing the Report

The following command will compile the data stored on the disk between a certain time period by executing all other commands, saving it in a compressed file.

Bash | Copy

```
./swx support -f "yyyy-mm-dd" -t "yyyy-mm-dd"  -s localhost
```

An example output file would be **swxtch-report-2024-05-22_15-58-55.tar.gz**.

Alternatively, you can export swx support logs using the wXcked Eye UI. To learn more, see the wXcked Eye Support Page article.

Available swx support Arguments

Exporting only xNIC Logs (xnic-logs)

Users can export xNIC logs by specifying it with the swx support command. In addition to the xnic-logs command, they must use the argument --xnic and specify the xNIC's control IP. In the example below, the xNIC's control IP is 10.5.1.4.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Example:

Bash	Copy

```
./swx support xnic-logs -since 2h -s localhost --xnic 10.5.1.4
```

An example output file would be **xnicLogs-10.5.1.4-2024-05-22_20-27-55.tar.gz**.

Date from (-f) and date to (-t)

- Following the **-f (date_from)**, enter the starting date.
- Following the **-t (date_to)**, enter the end date.
- Both dates can also include the time, using the ISO 8601 date format (Example: 2006-01-02T15:04:05Z)

Please note: Both "From" and "to" are optional. **If nothing is set, the default is over the past 7 days**.

Since hours/days (--since)

Following **-since**, users can specify number of days (d) or hours (h). For example, if a user would like to get logs since 8 hours ago, they would specify **-since 8h**. If they would like 4 days ago, they would specify **-since 2d**.

Example:

Bash	Copy

```
Hours
$ ./swx support -s localhost --since 8h

Days
$ ./swx support -s localhost --since 2d
```

Core dumps (--dump)

Add **-dump** if you wish to include the core dumps in the report.

Help (-h) for additional commands

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Add a **-h** to the command if you wish to see an extensive list of available commands. For example, you can specify a certain section of the report to export as a tar.gz file, similar to the xNIC logs example above.

Example:

Bash Copy

```
root@dsd-core-200:/swxtch# ./swx support -h
Swxtch support tool

Usage:
  swx support [flags]
  swx support [command]

Available Commands:
  bridge-log     Bridge log file
  list-highmarks List highmarks metrics within time range
  logs           Logs
  max-highmarks  Max highmarks metrics
  swxtch-info    Swxtch Info
  xnic-logs      xNIC logs file

Flags:
  -b, --bridge string                bridge address
  -f, --date-from string             Date from
  -t, --date-to string               Date to
  -d, --dump                         Dump enable
  -h, --help                         help for support
  -o, --path string                  Path location
  -s, --service-host-address string   Host swxtch address in the
form <host>[:port] (default "localhost")
  -c, --since string                 Since
  -x, --xnic string                  xnic

Use "swx support [command] --help" for more information about a
command.
```

cloudSwXtch Service Logs

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

In addition to the swx support logs, swXtch.io Support might request logs for the -ctrl/-repl services. For log requests, the standard command, **sudo journalctl -u,** can used with either **swxtch-ctrl.service** or **swxtch-repl.service** to get a detailed breakdown of cloudSwXtch activity.

- **swxtch-ctrl.service** - This will display information regarding the cloudSwXtch's control plane.
- **swxtch-repl.service** - This will display information regarding the cloudSwXtch's replicator app that is on the data plane.

In addition, the log request command can be used for **swxtch-bridge2 (Bridge Type 2)** and **swXtch-bridge (Bridge Type 1)** for bridge-related logs.

It is recommended for users to send logs from both services to support@swxtch.io. The logs should cover 24 hours worth of time, starting from before the issue to up until now.

Users can use any combination of the arguments below to create their logs.

Accessing and Following Logs (-f)

The following command will begin to display logs for either the swxtch-ctrl or swxtch-repl service at the time of the request. The -f argument will follow the logs and continually update. Logs prior to the call will not display.

Bash | Copy

```
sudo journalctl -u <swxtch-ctrl.service OR swxtch-repl.service> -f
```

Note: A user will need to choose between swxtch-ctrl or swxtch-repl. The .service is not necessary and will work with or without.

Listing a Certain Number of Lines in a Log (-n)

The following argument can be used to list a specific number of lines in a log.

Bash | Copy

```
sudo journalctl -u <swxtch-ctrl OR swxtch-repl> -n <number of lines>
```

Example:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
sudo journalctl -u swxtch-repl -n 200
```

Displaying Logs within a Timeframe (--since --until)

The following command will display logs within a set timeframe (between 2 dates).

Bash | Copy

```
sudo journalctl - u <swxtch-ctrl or swxtch-repl> --since <yyyy-mm-dd> --until <yyyy-mm-dd>
```

Example:

Bash | Copy

```
sudo journalctl - u swxtch-repl --since 2023-03-07 --until 2023-03-10
```

--since "Yesterday", "today", "now"

You can also use the words "yesterday," "today" or "now" after --since to get logs from that time period.

Bash | Copy

```
sudo journalctl - u swxtch-repl --since yesterday
```

Displaying Logs since Last Boot (-b)

To display logs since last boot, users can use the -b argument.

Bash | Copy

```
sudo journalctl -u <swxtch-ctrl OR swxtch-repl> -b
```

List boots (--list-boot)

The following argument can be used to list all the boots in date/time order.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```bash
sudo journalctl -u <swxtch-ctrl OR swxtch-repl> --list-boot
```

Exporting Logs (>)

The following command will export your logs to a .txt file. Logs should be emailed to support@swxtch.io.

```bash
sudo journalctl - u <swxtch-ctrl or swxtch-repl> > <file-name>.txt
```

Example:

```bash
sudo journalctl - u swxtch-repl > cloudswxtch-test.txt
```

You can also combine arguments to export logs from a timeframe or from last boot. It is recommended that logs should cover 24 hours worth of time, starting from before the issue to up until now.

Example:

```bash
LAST BOOT:
sudo journalctl - u swxtch-ctrl -b > cloudswxtch-test.txt

TIMEFRAME:
sudo journalctl - u swxtch-repl --since 2023-03-07 --until 2023-03-10 > cloudswxtch-test.txt
```

Change Logs to UTC (--utc)

To switch logs from local time to UTC, use the following argument:

```bash
sudo journalctl --utc
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

How to Install xNIC Dependencies in an Air-Gapped Environment

WHAT TO EXPECT

In this article, users will learn how to download and install the appropriate packages necessary to successfully deploy an xNIC in an air-gapped environment. Without the dependencies already on your air-gapped client, xNIC installation will fail.

Please follow the instructions based on your operating system:

- **RHEL-based Distros**
- **Ubuntu**

Installing Dependencies for RHEL-based Distros

The easiest way to install dependencies on RHEL-based distros is using the built-in tool called repotrack. By using this tool, users can download the necessary packages with their dependencies using a single command. Then, they can copy the rpm files from the **ancillary VM** to the **target agents** (the VMs within the air-gapped environment) and install them there.

List of dependencies

Based on the version of operating system, a user will have different package requirements.

Major Version	Packages list
RHEL 8 or 9	libpcap iproute-tc bpftool
Amazon Linux	libpcap iproute-tc bpftool wget

Make sure you download the correct packages for your operating system.

STEP ONE: Download All Packages and Dependencies onto your Ancillary VM

1. Select a VM that is connected to a repository or the internet that is similar to the one that will be used as an agent in your air-gapped environment. This will act as your "ancillary VM."

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Execute the following command, replacing <packages-list> with the list associated with the correct version.

   Shell

   ```
   Bash
   sudo repotrack <packages-list>
   ```

   a. This will download the packages and ALL their dependencies onto your ancillary VM.

STEP TWO: Copy the .rpm files to the Target VM

1. Secure copy the files now to the target machine on the air-gapped environment by issuing the following command:

   Shell

   ```
   Bash
   scp -i <private-key-path&file> *.rpm <target-username>@<target-hostname-or-IP>:<target-path>
   ```

   a. Example with replaced values:

      Shell

      ```
      Bash
      scp -i ~/.ssh/key.priv *.rpm swxtchadmin@172.41.130.20:/home/swxtchadmin
      ```

STEP THREE: Install the packages and dependencies on the Target VM

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. In the folder containing all the RPM files sent from the ancillary VM, issue the following command on the Target VM:

   Shell

   ```
   Bash                                                                    Copy
   sudo dnf install *.rpm --disablerepo=* --allowerasing --skip-broken
   ```

   a. This will analyze the system, skip what's already installed, give a report, and ask for confirmation.

You are now ready to install the xNIC.

Installing Dependencies for Ubuntu distros

Debian does not have a built-in tool to handle package installation in an air-gapped environment. While there are several 3rd-party tools available, **apt-offline** is recommended as the best "working" solution. Please note: Version 1.8.4 of apt-offline (the default for Ubuntu 22.04) fails on VMs with Python greater than 3.1.

List of dependencies

The following packages must be installed on the target VM in order to successfully deploy an xNIC in an air-gapped environment:

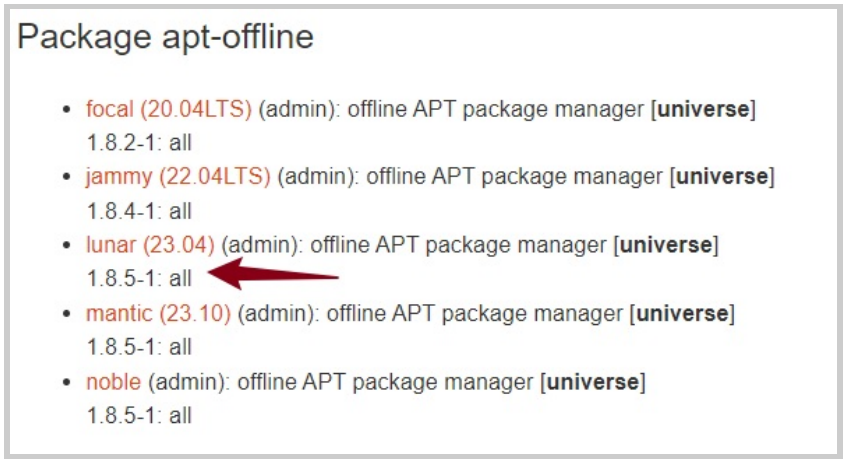
```
None                                                                    Copy
libpcap-dev binutils iproute2 linux-tools-generic
```

Pre-requisite: Download apt-offline tool on both the Ancillary and Target VMs

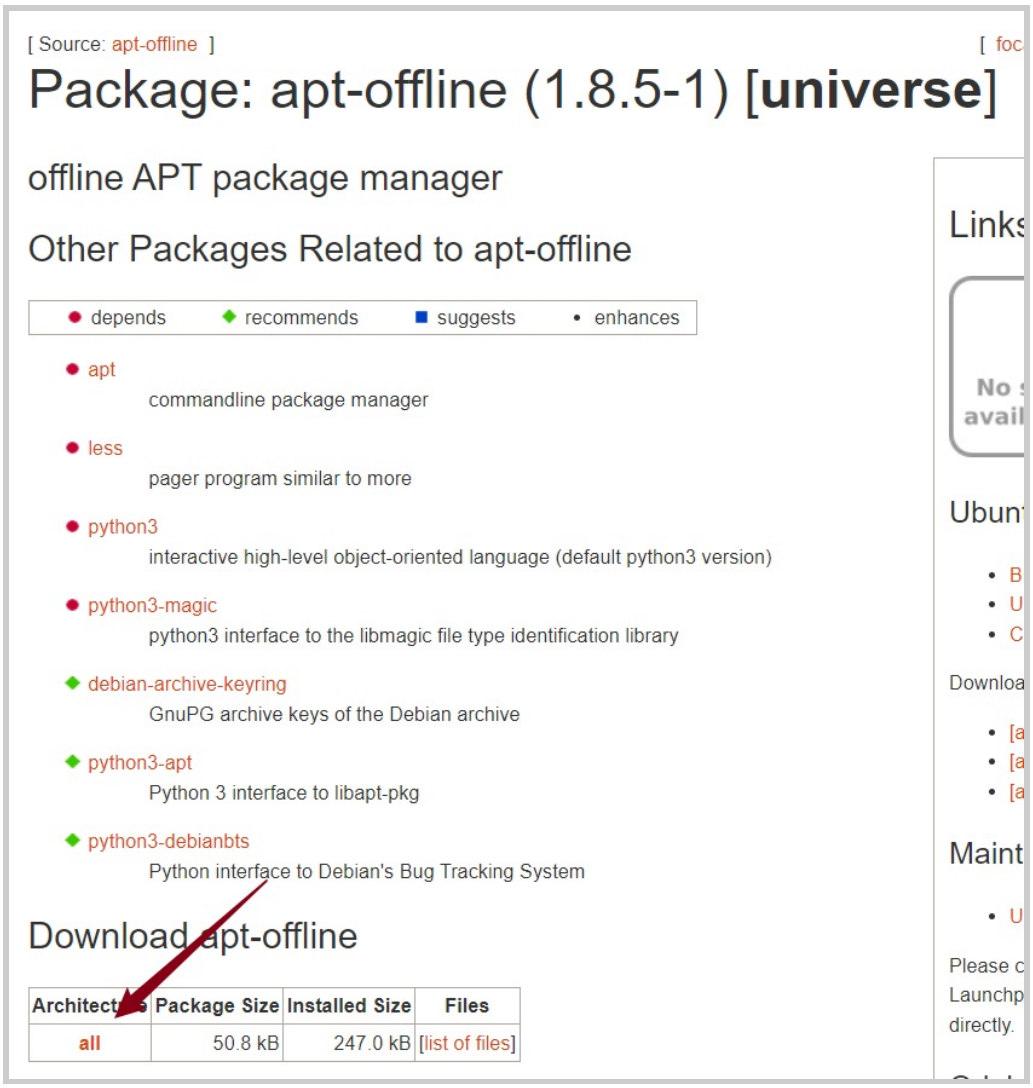
1. Go to the apt-offline package download page.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Select a version greater than 1.8.5 even if you have an older Ubuntu version. This will open a new package version page.

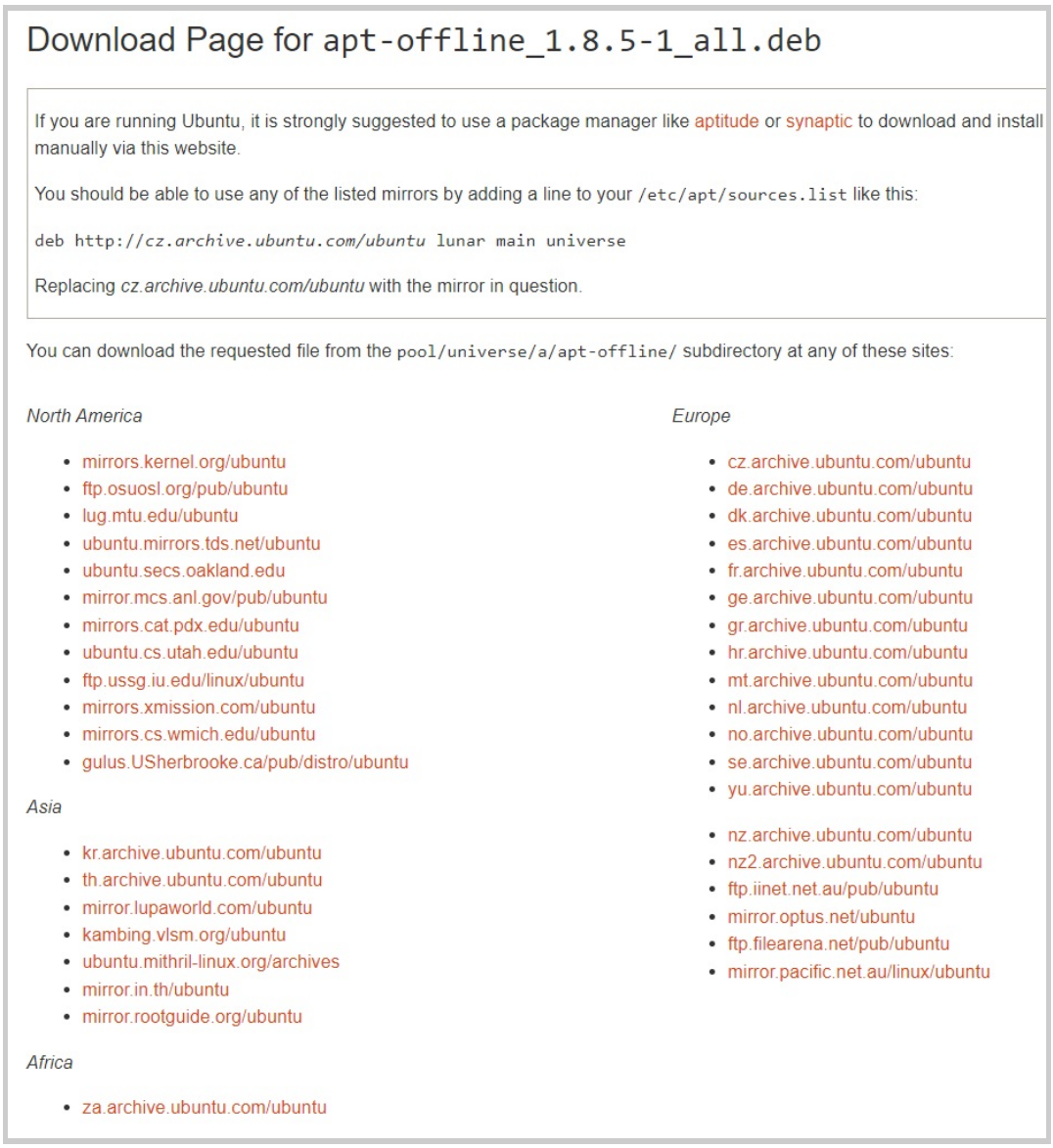


3. Click "all" on the selected version's page. This will open a page with the mirror list.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. Copy any URL from the mirrors list.



Download Page for apt-offline_1.8.5-1_all.deb

If you are running Ubuntu, it is strongly suggested to use a package manager like aptitude or synaptic to download and install manually via this website.

You should be able to use any of the listed mirrors by adding a line to your /etc/apt/sources.list like this:

deb http://cz.archive.ubuntu.com/ubuntu lunar main universe

Replacing cz.archive.ubuntu.com/ubuntu with the mirror in question.

You can download the requested file from the pool/universe/a/apt-offline/ subdirectory at any of these sites:

North America

- mirrors.kernel.org/ubuntu
- ftp.osuosl.org/pub/ubuntu
- lug.mtu.edu/ubuntu
- ubuntu.mirrors.tds.net/ubuntu
- ubuntu.secs.oakland.edu
- mirror.mcs.anl.gov/pub/ubuntu
- mirrors.cat.pdx.edu/ubuntu
- ubuntu.cs.utah.edu/ubuntu
- ftp.ussg.iu.edu/linux/ubuntu
- mirrors.xmission.com/ubuntu
- mirrors.cs.wmich.edu/ubuntu
- gulus.USherbrooke.ca/pub/distro/ubuntu

Asia

- kr.archive.ubuntu.com/ubuntu
- th.archive.ubuntu.com/ubuntu
- mirror.lupaworld.com/ubuntu
- kambing.vlsm.org/ubuntu
- ubuntu.mithril-linux.org/archives
- mirror.in.th/ubuntu
- mirror.rootguide.org/ubuntu

Africa

- za.archive.ubuntu.com/ubuntu

Europe

- cz.archive.ubuntu.com/ubuntu
- de.archive.ubuntu.com/ubuntu
- dk.archive.ubuntu.com/ubuntu
- es.archive.ubuntu.com/ubuntu
- fr.archive.ubuntu.com/ubuntu
- ge.archive.ubuntu.com/ubuntu
- gr.archive.ubuntu.com/ubuntu
- hr.archive.ubuntu.com/ubuntu
- mt.archive.ubuntu.com/ubuntu
- nl.archive.ubuntu.com/ubuntu
- no.archive.ubuntu.com/ubuntu
- se.archive.ubuntu.com/ubuntu
- yu.archive.ubuntu.com/ubuntu

- nz.archive.ubuntu.com/ubuntu
- nz2.archive.ubuntu.com/ubuntu
- ftp.iinet.net.au/pub/ubuntu
- mirror.optus.net/ubuntu
- ftp.filearena.net/pub/ubuntu
- mirror.pacific.net.au/linux/ubuntu

5. Select a VM that is connected to a repository or the internet that is similar to the one that will be used as an agent in your air-gapped environment. This will act as your "ancillary VM."

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

6. Issue the following command, replacing **<url-of-the-tool>** with the copied mirror URL:

   Shell

   ```
   Bash                                                    Copy

   wget <url-of-the-tool>
   ```

   a. **Example**:

      Shell

      ```
      Bash                                                 Copy

      wget http://mirrors.kernel.org/ubuntu/pool/universe/a/apt-
      offline/apt-offline_1.8.5-1_all.deb
      ```

   b. This will download the file **apt-offline_<something>** to the ancillary machine. **<something>** will depend on what version you are downloading.

7. Install that package by issuing the following command, replacing **apt-offline_<something>**with the actual file name you have:

   Shell

   ```
   Bash                                                    Copy

   sudo dpkg -i apt-offline_<something>
   ```

8. Copy the .deb file to the target VM and repeat the installation process.

STEP ONE: Generate a signature file for all the packages on the Target VM.

1. Issue the following command in the target VM:

   Shell

   ```
   Bash                                                    Copy

   sudo apt-offline set --install-packages libpcap-dev binutils iproute2
   linux-tools-generic --update apt-offline.sig
   ```

2. Secure copy the apt-offline.sig file to the ancillary VM.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

STEP TWO: Download the packages and dependencies

1. Enter the following command on the ancillary VM with the signature file to bundle the packages and dependencies:

   Shell

   ```
   Bash                                                    Copy
   apt-offline get --bundle bundle.zip apt-offline.sig
   ```

2. Secure copy the bundle.zip to the target VM.

STEP THREE: Install the packages and dependencies on the Target VM.

Enter the following command on the target VM to install the packages and dependencies bundle:

Shell

```
Bash                                                    Copy
sudo apt-offline install bundle.zip
```

All the packages and dependencies should now be installed on the target VM.

STEP FOUR: Configure bpftool

Issue the following command:

```
Bash                                                    Copy
sudo ln -sf "$(find /usr/lib/linux-tools/ -name bpftool | head -1)"
/usr/sbin/bpftool
```

You are now ready to install the xNIC.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

How to View cloudSwXtch Bridge Logs

WHAT TO EXPECT

In this article, users will learn how to access cloudSwXtch Bridge logs for the purpose of troubleshooting. Common arguments that can be used when compiling information for the Support team at swXtch.io will also be discussed.

Support may also request for you to send xNIC logs. For more information, see **How to Find xNIC Logs**.

cloudSwXtch Bridge Service Logs

For log requests, the standard command, **sudo journalctl -u** can be used with either **swxtch-bridge2** (Bridge Type 2) or **swxtch-bridge** (Bridge Type 1) to get a detailed breakdown of cloudSwXtch Bridge activity.

For example:

Bash | Copy

```
sudo journalctl -u swxtch-bridge2
```

It is recommended to send logs to support@swxtch.io, coverign 24 hours worth of time, starting form before the issue to up until now.

Users can use any combination of arguments below to create their logs.

Accessing and Following Logs (-f)

The following command will begin to display cloudSwXtch Bridge logs at the time of the request. The **-f** argument will follow the logs and continually update. Logs prior to the call will not display.

Bash | Copy

```
sudo journalctl -u <swxtch-bridge2 OR swxtch-bridge> -f
```

Note: A user will need to choose between **swxtch-bridge2** or **swxtch-bridge**.

Example:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash	Copy

```
sudo journalctl -u swxtch-bridge2 -n 200
```

Displaying Logs within a Timeframe (--since --until)

The following command will display logs within a set timeframe (between 2 dates).

Bash	Copy

```
sudo journalctl - u <swxtch-bridge2 or swxtch-bridge> --since <yyyy-mm-dd> --until <yyyy-mm-dd>
```

Example:

Bash	Copy

```
sudo journalctl - u swxtch-bridge2 --since 2023-03-07 --until 2023-03-10
```

--since "Yesterday", "today", "now"

You can also use the words "yesterday," "today", or "now" after **--since** to get logs from that time period.

Bash	Copy

```
sudo journalctl - u swxtch-bridge2 --since yesterday
```

Displaying Logs since Last Boot (-b)

To display logs since last boot, users can use the **-b** argument.

Bash	Copy

```
sudo journalctl -u <swxtch-bridge2 OR swxtch-bridge> -b
```

List boots (--list-boot)

The following argument can be used to list all the boots in date/time order.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```bash
sudo journalctl -u <swxtch-bridge2 or swxtch-bridge> --list-boot
```

Exporting Logs (>)

The following command will export your logs to a .txt file. Logs should be emailed to support@swxtch.io.

```bash
sudo journalctl -u <swxtch-bridge2 or swxtch-bridge> > <file-name>.txt
```

Example:

```bash
sudo journalctl -u swxtch-bridge2 > cloudswxtch-test.txt
```

You can also combine arguments to export logs from a timeframe or from last boot. It is recommended that logs should cover 24 hours worth of time, starting from before the issue to up until now.

Example:

```bash
LAST BOOT:
sudo journalctl - u swxtch-bridge2 -b > cloudswxtch-test.txt

TIMEFRAME:
sudo journalctl - u swxtch-bridge2 --since 2023-03-07 --until 2023-03-10 > cloudswxtch-test.txt
```

Change Logs to UTC (--utc)

To switch logs from local time to UTC, use the following argument:

```bash
sudo journalctl --utc
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

How to Find xNIC Logs

WHAT TO EXPECT

In this article, you will learn how to find xNICs logs on your VM and how to alter its verbosity level.

swXtch.io Support may also request for you to send cloudSwXtch logs. For more information, see How to View cloudSwXtch Logs for Troubleshooting.

Locating xNIC Logs

An xNIC installed on a virtual machine creates one .log file per day with the following naming structure: **swxtch-xnic-YYYYMMDD.log**. If the file size exceeds the maximum within the same day (16MB), it will be renamed by adding a counter as a suffix. Then, a new file will be created.

To find your logs, use the following file paths:

Windows

- C:\Users\Public\swx\logs\xnic-control
- C:\Users\Public\swx\logs\xnic-data

Linux

- /var/log/swx/xnic-control
- /var/log/swx/xnic-data

For Linux, logs can also be viewed by using either journalctl examples below:

xnic-control

Bash | Copy

```
sudo journalctl -u swxtch-xnic-control -n 500 -f
```

xnic-data

Bash | Copy

```
sudo journalctl -u swxtch-xnic-data -n 500 -f
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Please note: Standard journalctl arguments apply. The above examples use **-n** for number of lines and **-f** to follow.

Log File Deletion

Log files older than 30 days are automatically deleted.

What is verbosity?

Depending on the level of verbosity detailed in the xNIC config file, a log will contain different application messages and usage statistics. The default verbosity level after xNIC installation is 0, which means that no periodic statistics are being reported. It will only show start and stop information as well as critical errors.

A user can change the verbosity to pull more information out from their xNIC. The levels are detailed below:

- **Level 0**: Only show start and stop info as well as critical errors. This is the default.
- **Level 1**: Shows statistics and IGMP messages
- **Level 2**: Additional control messages
- **Level 3**: Hexadecimal dumps of control/config packages
- **Level 4**: Hexadecimal dumps of data packages

An average user would typically only need up to Level 2 for troubleshooting issues with their xNIC.

Verbosity and File Size

Please note that increasing the verbosity level of future logs will result in larger file sizes. It is recommended to revert back to the default Level 0 when testing and troubleshooting is complete.

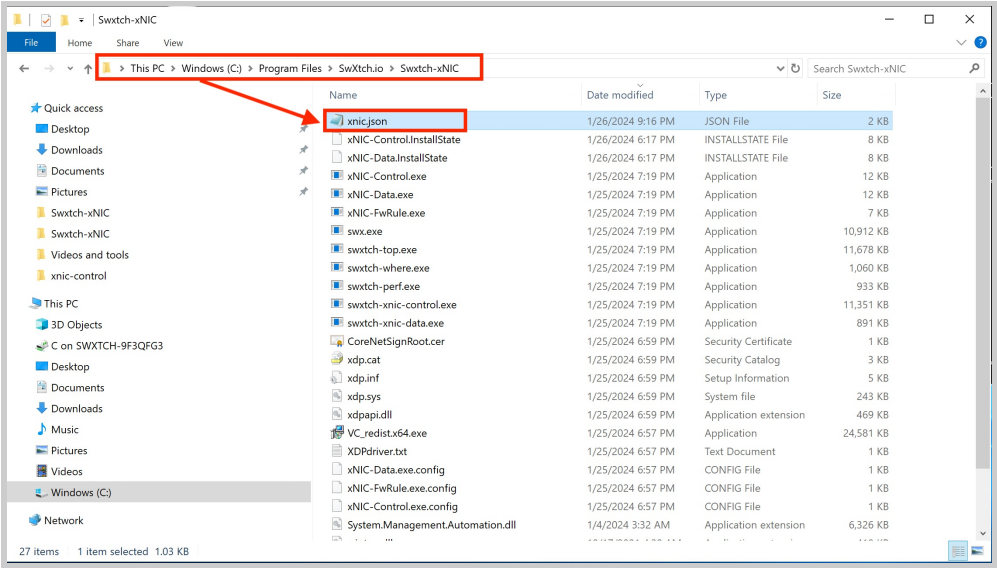
How to Change Verbosity

To change the verbosity, a user can manually edit the xNIC config file on their VM.

For Windows:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

1. **Go** to the Swxtch-xNIC folder on the VM you have an xNIC installed. Make sure it is the xNIC you want logs for.
    a. **C:\Program Files\SwXtch.io\Swxtch-xNIC**



2. **Open** the **xnic.json** file.
3. For xNIC-Data, **change** the number next to "verbosity" so that it matches the level you desire. The default is 0.

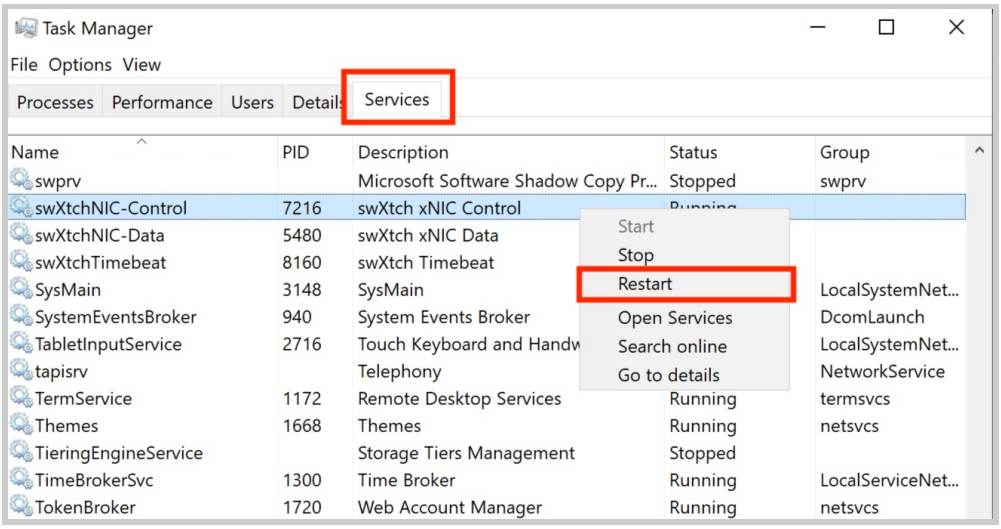
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. For xNIC-Control, modify the StatsReportWait to change the time in seconds of the report stats.

5. **Save and Close** the json file.
6. **Open** "Task Manager" and **go to** the "Services" tab towards the top of the window.
7. To apply changes from the json file, **scroll** down to "swXtchNIC-Control" and **right-click** on it.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

8. **Select**"Restart."
    a. **Please note**: Only xNIC-Control has to be restarted to update both control and data logs.



Your selection in verbosity will now be applied to future logs. This should only be changed if directed by swXtch.io and ensure that they are set back to default when troubleshooting is complete.

For Linux:

1. **Enter** the following command to view your config file in the Bash terminal. Make sure it is on the xNIC you want logs for.

    Text

    ```bash
    sudo nano /var/opt/swxtch/swxtch-xnic/swxtch-xnic.conf
    ```

2. For xNIC-Data, **change** the number next to "verbose" so that it matches the level you desire. The default is 0.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. For xNIC-Control, **modify** the StatsReportWait to change the time in seconds of the report stats.

4. **Save** and **Exit** the file.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

5. **Restart** your swxtch-xnic-control by using the following command. **Please note**: Only xNIC-Control has to be restarted to update both control and data logs.

   Text

   Bash | Copy

   ```
   sudo systemctl restart swxtch-xnic-control
   ```

   Your selection in verbosity will now be applied to future logs. This should only be changed if directed by swXtch.io and ensure that they are set back to default when troubleshooting is complete.

PRO-TIP

Rename your existing log file before restarting the xNIC service in order to differentiate it with the freshly generated log file containing the new verbosity data.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

How to License a cloudSwXtch

WHAT TO EXPECT

In this article, users will learn the appropriate steps for licensing their cloudSwXtch instance.

1. Log into the newly created cloudSwXtch VM.

2. Run the command:

```bash
sudo /swxtch/swxtch-top dashboard --swxtch localhost
```

3. The **swXtch-top dashboard** will display.

Alternatively, if you do not want to open swXtch-top, you can also use the following command to get the **SwXtchID**:

```bash
curl -s http://127.0.0.1/top/dashboard | grep -m 2 -Eo
'"fingerprint"[^,]*' | head -1
```

4. Copy the **SwXtchID** and email it to support@swxtch.io requesting a license.

5. When you receive the license, upload it onto the cloudSwXtch VM.

6. Move the **license.json** file to the **/swxtch** directory using the following command replacing user with the appropriate value:

```bash
sudo mv /home/<user>/license.json /swxtch
```

7. Return to the **swxtch-top dashboard** again check the license took hold.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

How to Install a cloudSwXtch Licensing Server

WHAT TO EXPECT

The Licensing Server allows users to circumvent the traditional license request process by installing a license bundle server directly onto a VM connected to the cloudSwXtch. Instead of having to email support@swXtch.io for a SwXtchID specific license file for every instance, the user themselves can grant a locked set of entitlements to a specific number of instances. This pair of maximum instances permitted and entitlements set is called a bundle.

In this article, users will learn how to install a self-hosted licensing server for their cloudSwXtch network.

Accessing the Licensing Server Installer

Prerequisites

The Licensing Server is supported on Ubuntu 20.04 and 20.22. When creating your VM, it is recommended to have at least four (4) CPUs with 16 GBs or greater of memory. Example instance sizes:

- **AWS**: m5zn.xlarge
- **Azure**: Standard_D4s_v5

Installing the Licensing Server is a relatively straight-forward process.

1. Create the VM that will be your Licensing Server based on the prerequisites. **Note**: This must be on a separate VM from the cloudSwXtch.
2. Run the following command to get the VM's unique system identifier.

```
Bash                                                    Copy
sudo dmidecode -s system-uuid
```

3. Send the uuid to support@swxtch.io, requesting both the **Licensing Server installer** and the **license.dat**.
4. Upload the **Licensing Server installer** and **license.dat** file to your Licensing Server VM.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

5. Run the following command.

   Shell

   ```
   sudo dpkg -i swxtch-license-svr_1.0.0_amd64.deb
   ```

   This will initiate the installer and create a folder that would hold the license bundles (**/swxtch/bundles**). At install, the folder will be empty.

6. Copy the license.dat file into the bundle folder located under **/swxtch/bundles** using the following command:

   ```
   sudo cp license.dat /swxtch/bundles
   ```

7. Run the following command to start the Licensing Server. The Licensing Server will automatically start when you boot the VM in the future.

   ```
   sudo systemctl start swxtch-license-svr
   ```

You should now be ready to configure the cloudSwXtch.

Configuring the cloudSwXtch

After installing the Licensing Server on your designated VM, a user will need to configure their cloudSwXtch to point at it when looking for a license file.

1. Create the **remote-licensing.json** file in your cloudSwXtch directory. It ***MUST*** have the name: **remote-licensing.json**

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Enter the following information in the remote-licensing.json file, replacing <IP-license-server> with the IP address of the VM selected to be the licensing server.

   Shell

   ```
   {
           "Url": "http://<IP-license-server>:8081/swxtch/bundles",
           "bundleName": "license"
   }
   ```

   a. **Note**: In versions 2.1.0+, there is currently only one bundle file available with specific entitlements named "license".

3. Save and close the file.

4. Restart the cloudSwXtch, specifically the swxtch-ctrl.service.

The cloudSwXtch is now ready and pointing to the license server.

Viewing License Bundles

A user can view their bundles by running the following command on their cloudSwXtch, its agents or the licensing server VM:

```
curl http://<IP-license-server>:8081/swxtch/bundles/show
```

Please note that there is currently only one bundle file available with specific entitlements named "license". The example below shows what that output will look like:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash | Copy

```
testadmin@agent-000:/swxtch$ curl
http://localhost:8081/swxtch/bundles/show
[
    {
        "bundleName": "license",
        "poolSize": 5,
        "consumedCount": 1,
        "authorized": [
            {
                "Fingerprint": "86224545-12aa-42af-9a24-
374a6c1a8a7c",
                "UpdateTime": "2023-12-28T16:35:46.214579179Z"
            }
        ]
    }
]
```

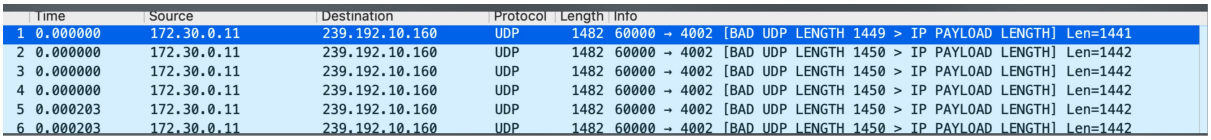
poolSize lists the amount of cloudSwXtches allowed on this bundle. In this example, 5 cloudSwXtches are allows with 1 already consuming a license (**consumedCount**).

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

How to set MTU size

In some cases the MTU Size of the multicast group may exceed the 1500 set limit in Windows and Linux virtual machines. This article will explain how to increase the MTU size if this should occur.

To know if the MTU size has been exceeded Wireshark or tcpdump can be used. Below is an example from Wireshark.



	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1449 > IP PAYLOAD LENGTH] Len=1441
2	0.000000	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1450 > IP PAYLOAD LENGTH] Len=1442
3	0.000000	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1450 > IP PAYLOAD LENGTH] Len=1442
4	0.000000	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1450 > IP PAYLOAD LENGTH] Len=1442
5	0.000203	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1450 > IP PAYLOAD LENGTH] Len=1442
6	0.000203	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1450 > IP PAYLOAD LENGTH] Len=1442

your title goes here

your content goes here
:::Note: The UDP length error shows it is exceeding the Length.

Linux update MTU Size:

First check MTU current size by running the following command:

None Copy

```
ifconfig | grep mtu
```

Example:

None Copy

```
someadmin@my-agent-101:~$ ifconfig | grep mtu
enP43852s1: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST>  mtu 1500
enP4589s2: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST>  mtu 1500
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
```

Note: The MTU size of eth1 = 1500

To change MTU size to 2000 for example - use the command below:

None Copy

```
sudo ifconfig eth1 mtu 2000 up
```

Validate it is set to new value in this case 2000 by running this command:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
None
Copy

ifconfig | grep mtu
```

Example:

```
None
Copy

someadmin@my-agent-101:~$ ifconfig | grep mtu
enP43852s1: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST>  mtu 1500
enP4589s2: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST>  mtu 2000
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 2000
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
```

Note: The MTU size of eth1 is now = 2000

Windows Update MTU Size

1. Check MTU Size by running this command:

```
None
Copy

netsh interface ipv4 show subinterfaces
```

You will see a list of network interfaces.
2. Set the MTU Size (in this case to 2000) using the following commands:

```
None
Copy

netsh
```

```
None
Copy

interface
```

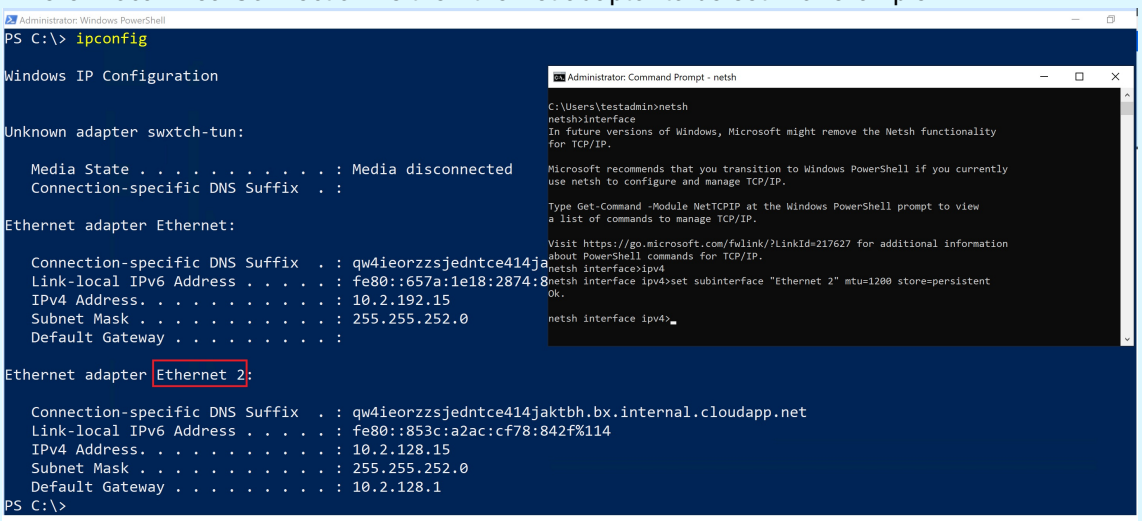
```
None
Copy

ipv4
```

and finally to acturally set the value:

```
None
Copy

set subinterface “Local Area Connection” mtu=2000 store=persistent
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Where “Local Area Connection” is the Ethernet adaptor to be set - for example:



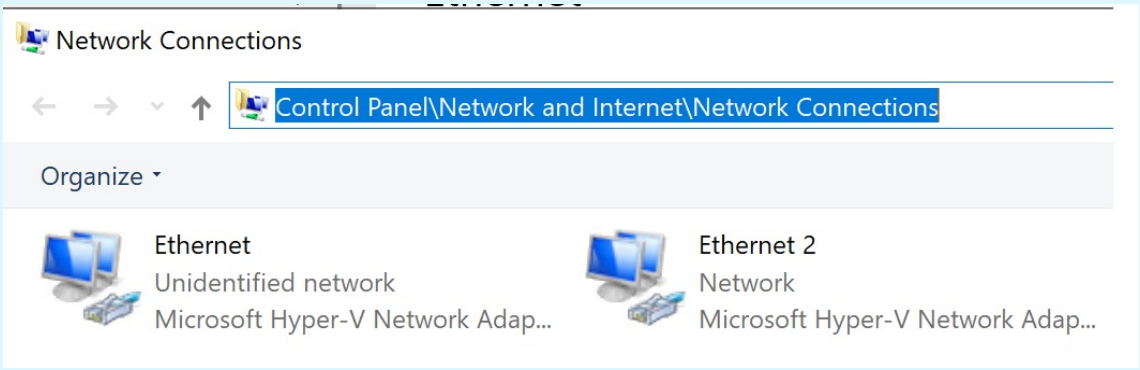
{height="" width=""}

2a. If you get the following error then follow steps after error:

None	Copy

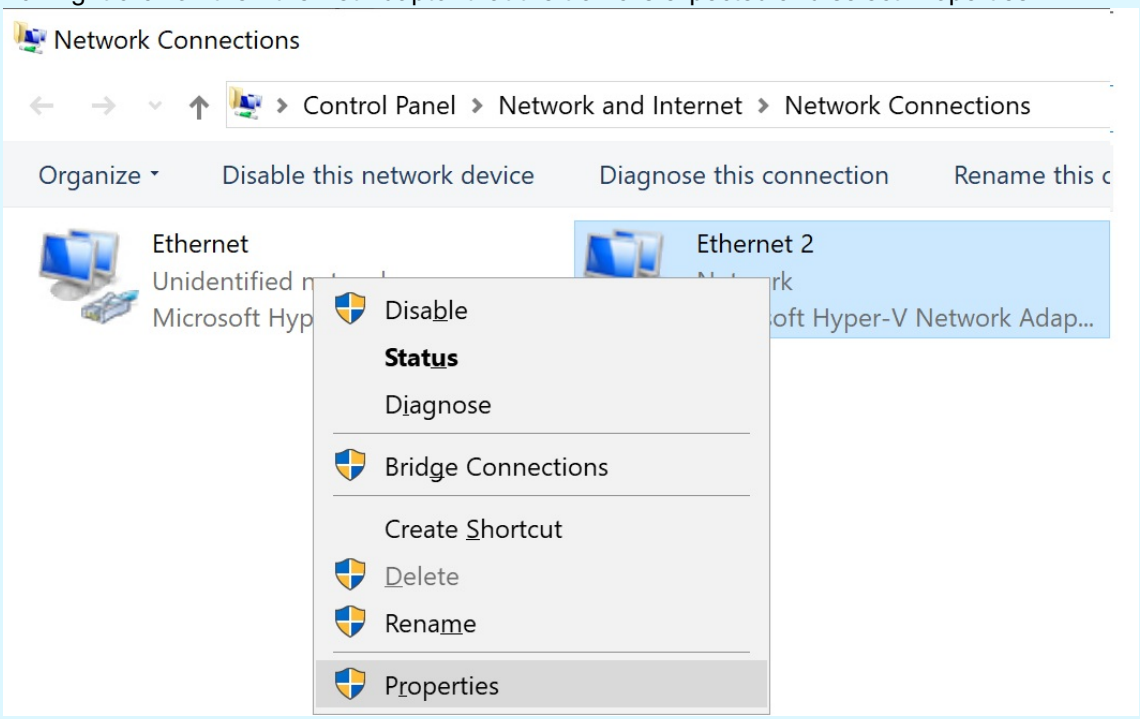
```
netsh interface ipv4>set subinterface "Ethernet 2" mtu=2000
store=persistent
The parameter is incorrect.
```

2b. Go to the Network connection → “Control Panel\Network and Internet\Network Connections”

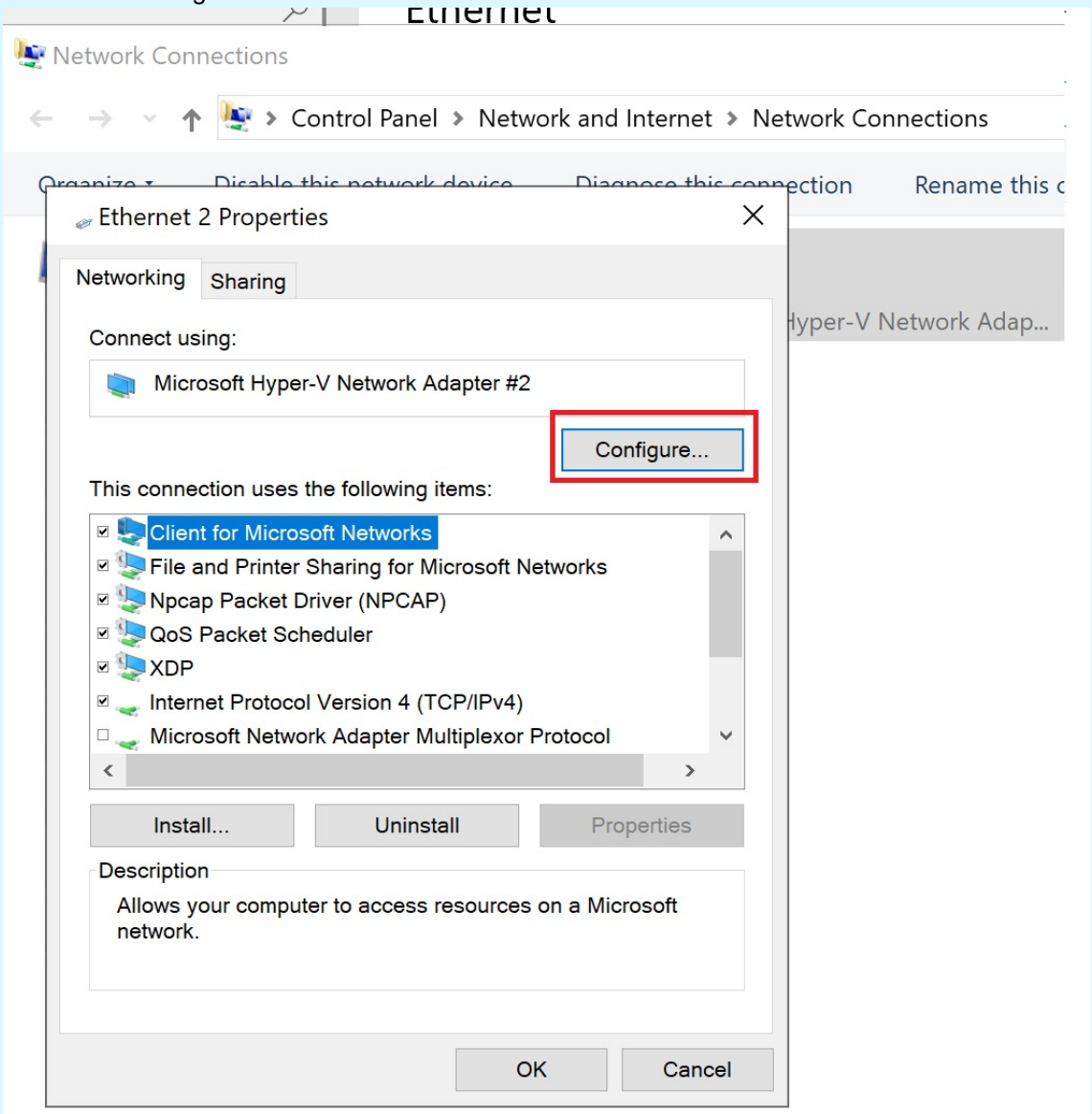


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2c. Right click on the Ethernet Adaptor that the traffic is expected and select Properties.

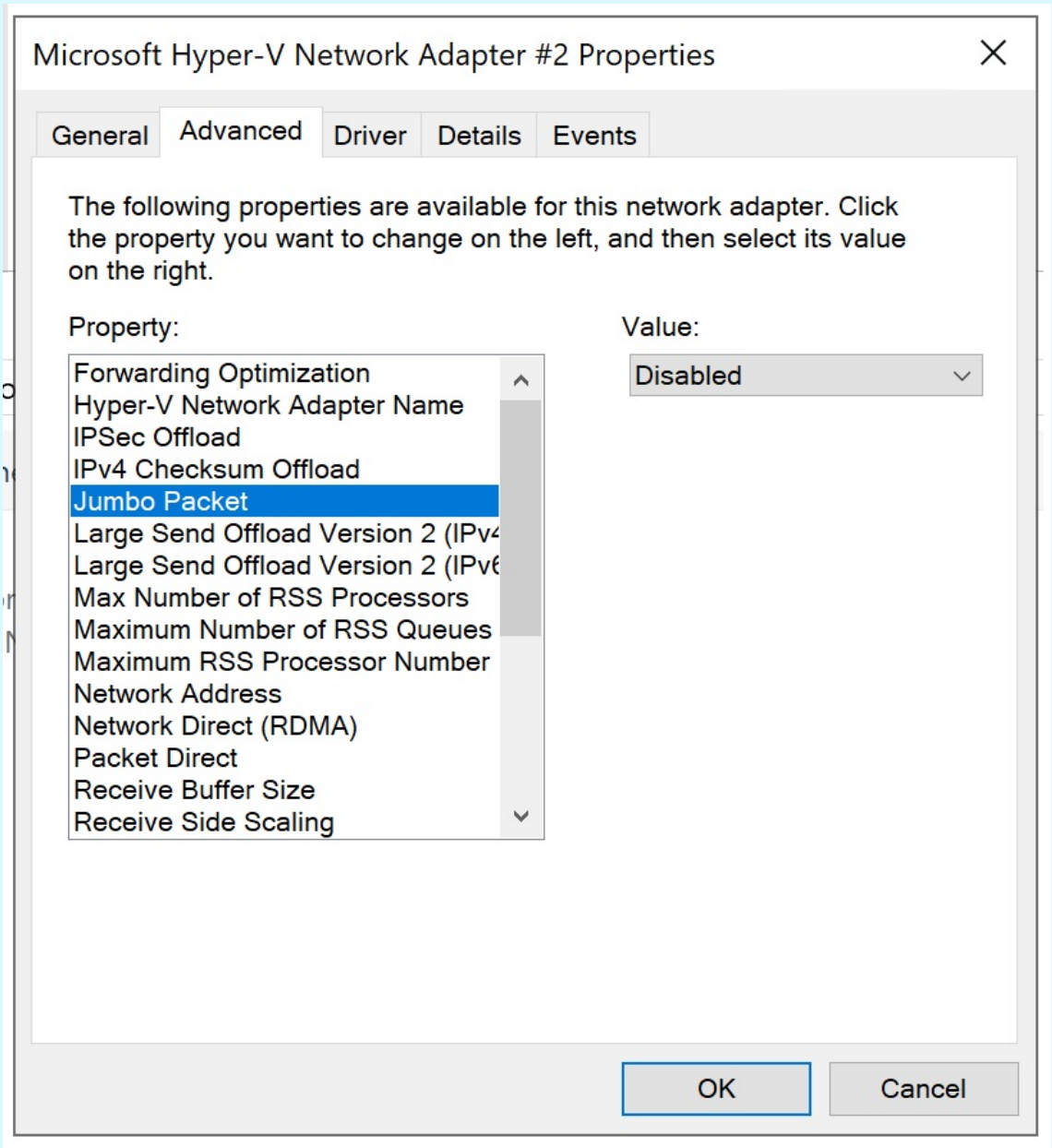


2d. Select “Configure”



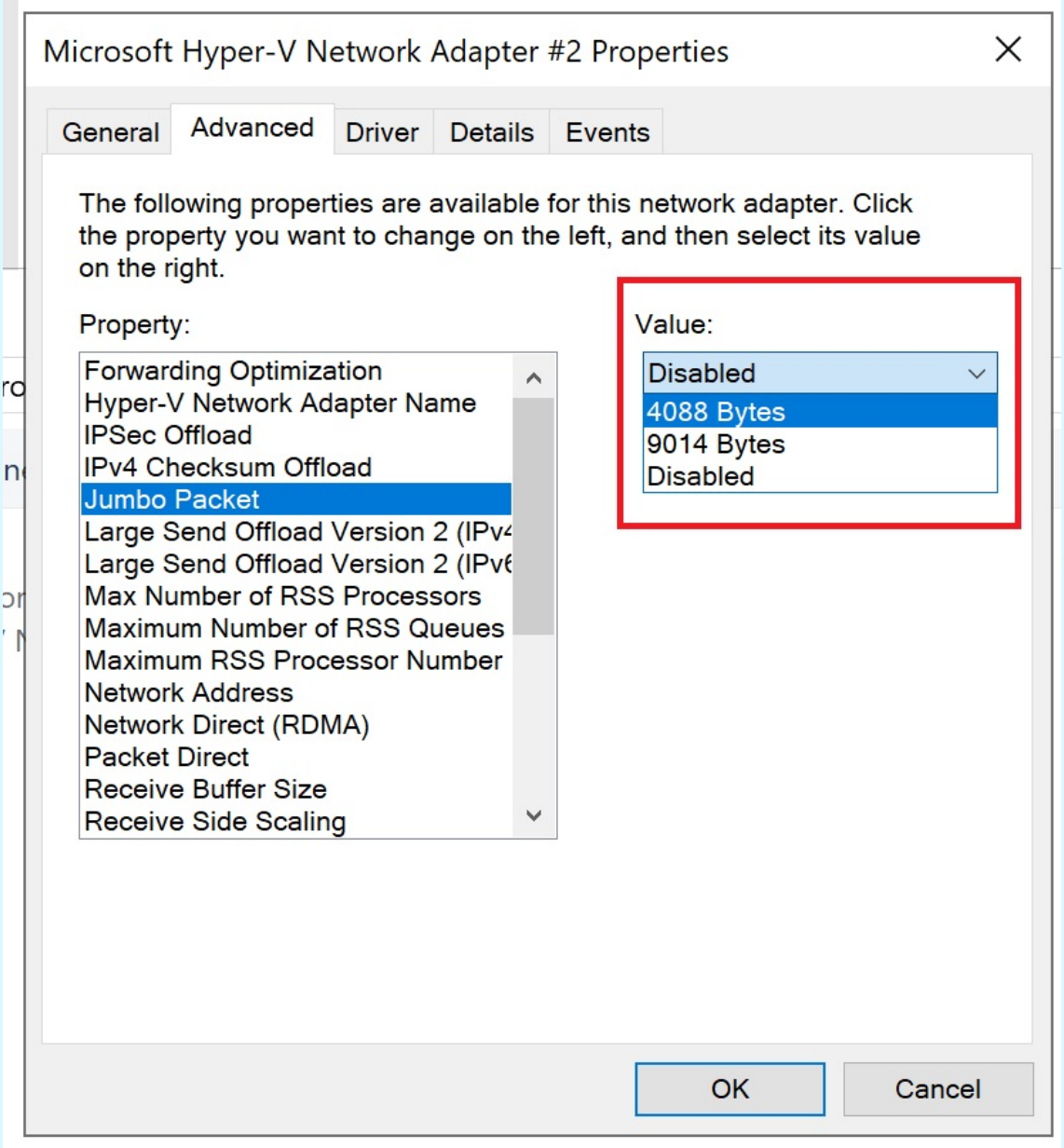
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2e. Select "Jumbo Packet"



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2f. Select "4088 Bytes" then select "OK".



{height="" width=""}

3. Re-run step 2 to set MTU size

5. Reboot your computer

6. Re-run step 1 to validate the MTU size is correct

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

How to Peer between VPCs in Different Regions for AWS

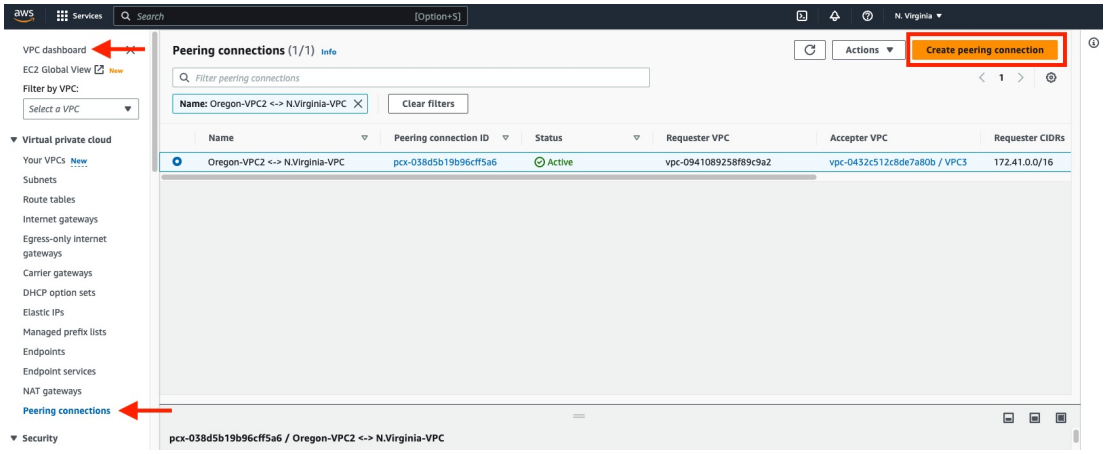
WHAT TO EXPECT

In order to successfully do Peering Connections between VPCs in different regions on AWS, a user must configure their route tables to allow traffic between instances. This will ensure that packets destined for a specific network segment on the other region/VPC/subnet are correctly routed.

In this article, users will learn how to Create a Peering Connection between Different Regions, Modify Route Tables and Edit Subnet Associations. Step 2 and 3 of the process will need to be repeated for both regions.

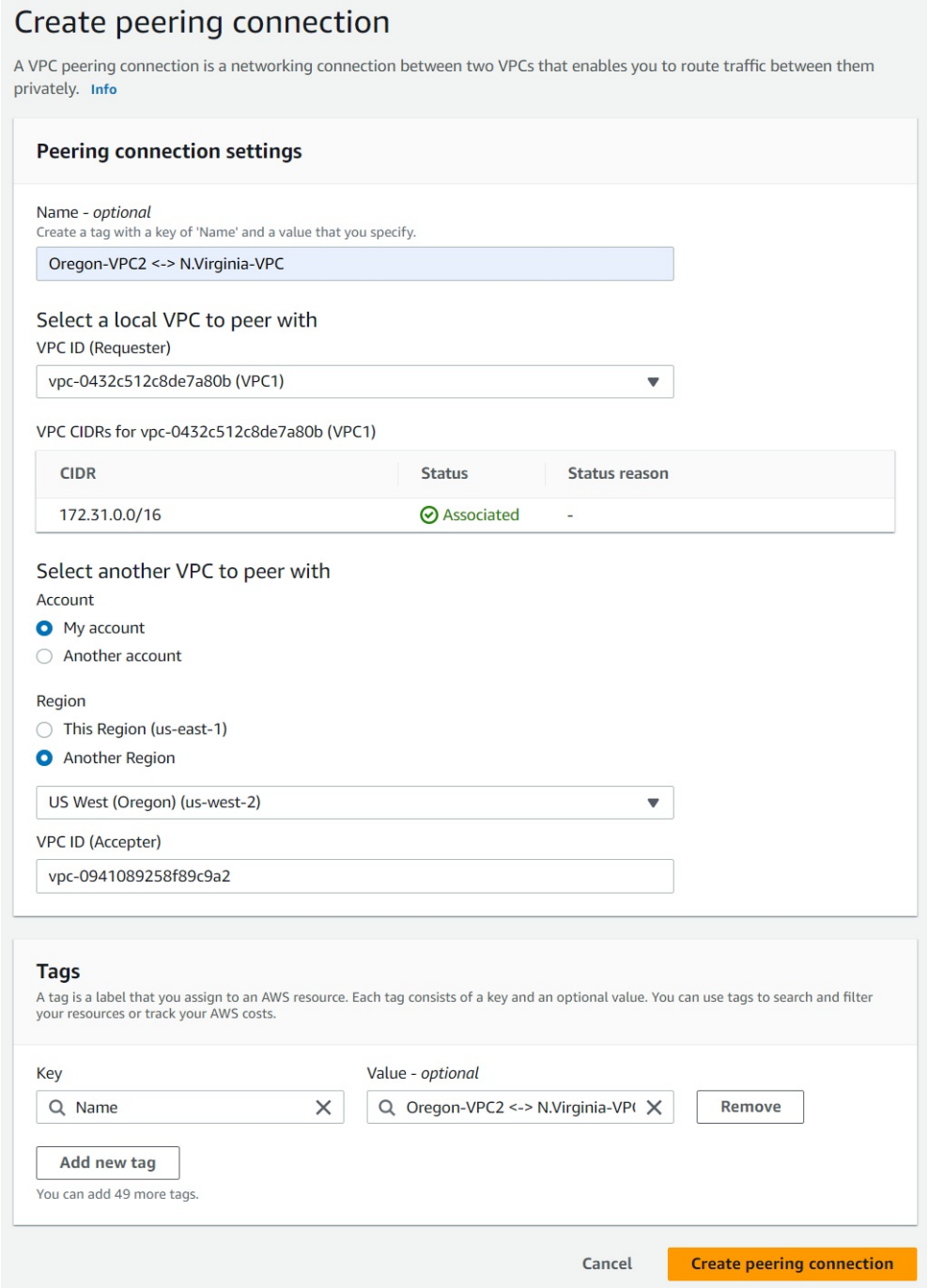
STEP ONE: Create a Peering Connection between Different Regions

1. Go to the **VPC Dashboard** and select **Peering Connections**.
2. Click **Create peering connection**.



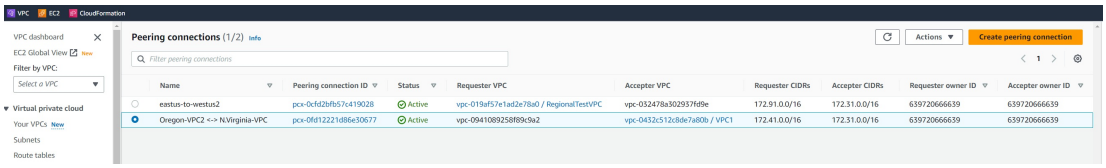
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

3. Edit the following in the **Create peering connection** form:
    a. Set a descriptive name. In the example, the user lists the connection between VPCs from Oregon and N. Virginia.
    b. Select the VPC of the instance you want to connect from.
    c. Select **Another Region** and select the destination region from the dropdown menu.
    d. Enter the **VPC ID** of the target VPC in the target region.
    e. Add any tag needed for organization purposes.



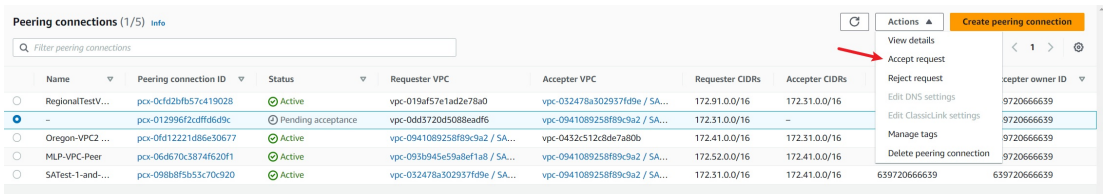
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. Click **Create peering connection**. A new Peering Connection should now be listed for the region you're on. **Please note**: A "mirrored connection" will be created on the "destination" region. It must be accepted manually to be active.



5. Change to the other region.
6. Go to **Peering Connections**.
7. Select the new **Peering Connection** listed as "Pending acceptance."

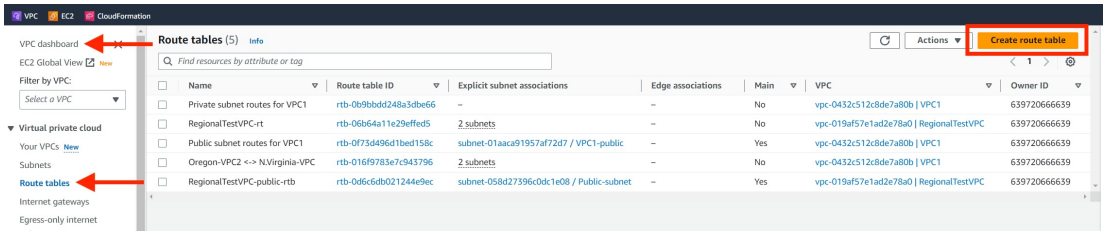
8. Under the Actions dropdown, select **Accept request**.



STEP TWO: Modify Route Tables in Both Regions

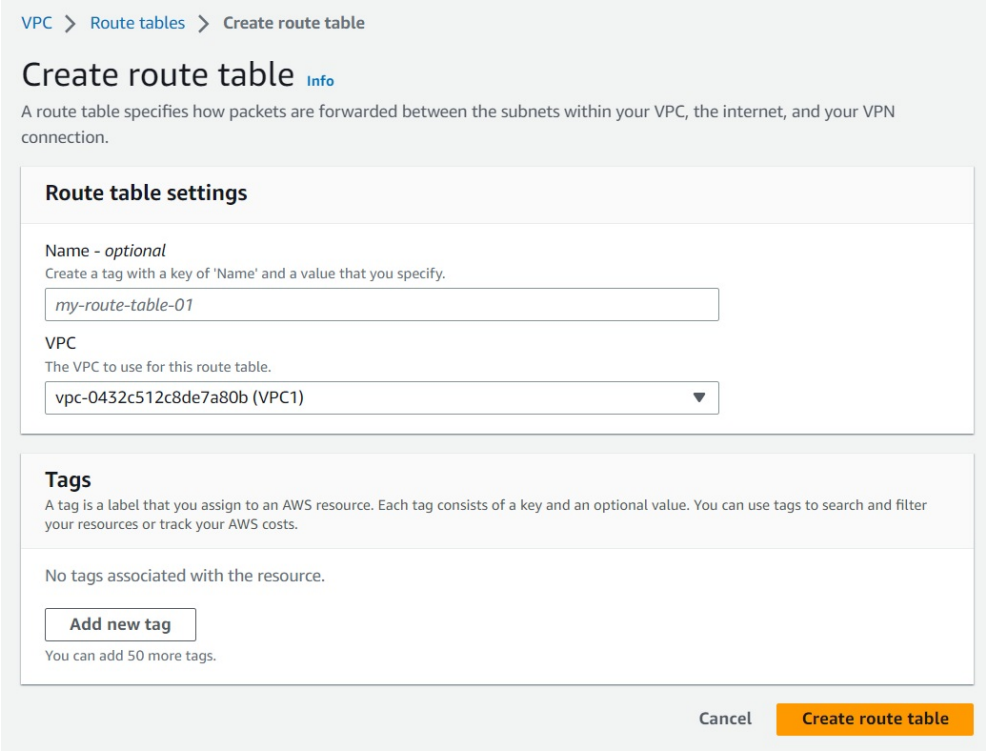
Once the peering connections are created, the route table must be modified in both regions. Start with the 1st region and complete STEP TWO and STEP THREE.

1. Go to the **VPC Dashboard**.
2. Click on **Route tables** in the Virtual private cloud section.
3. Select **Create route table** button.

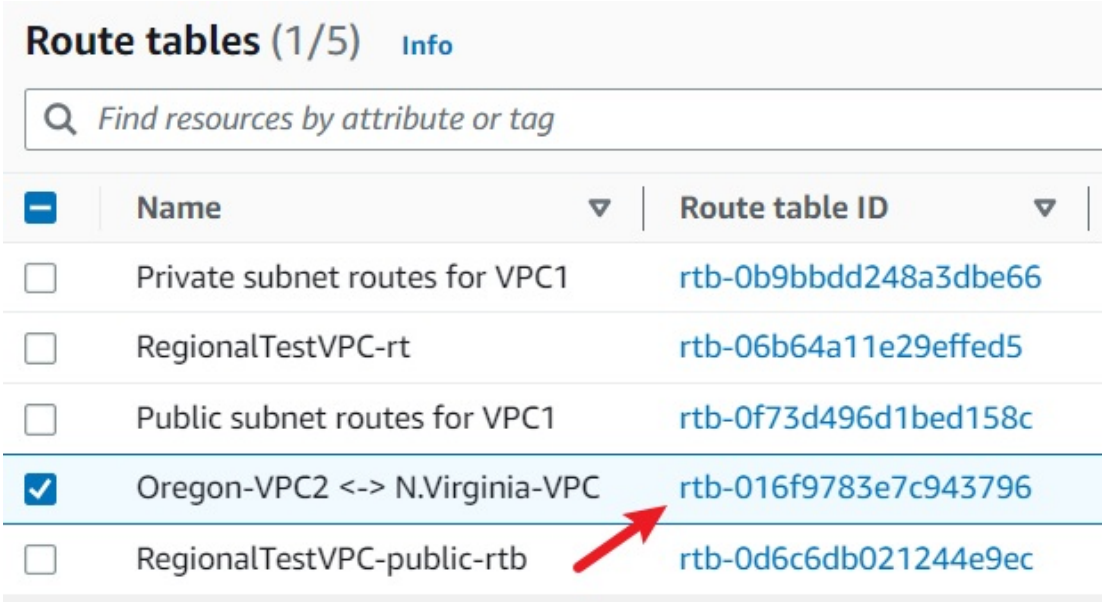


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

4. Edit the following in the **Create route table** form:
    a. Enter a descriptive name.
    b. Select the correct VPC.
    c. Add any necessary tags.

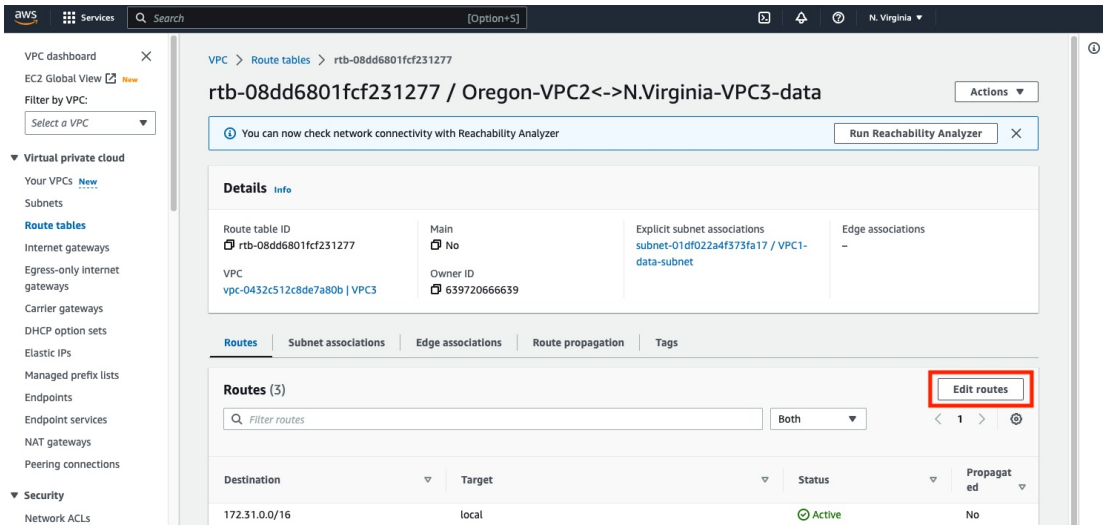


5. Select **Create route table**.
6. Select the **Route table ID** of the route table you just created.

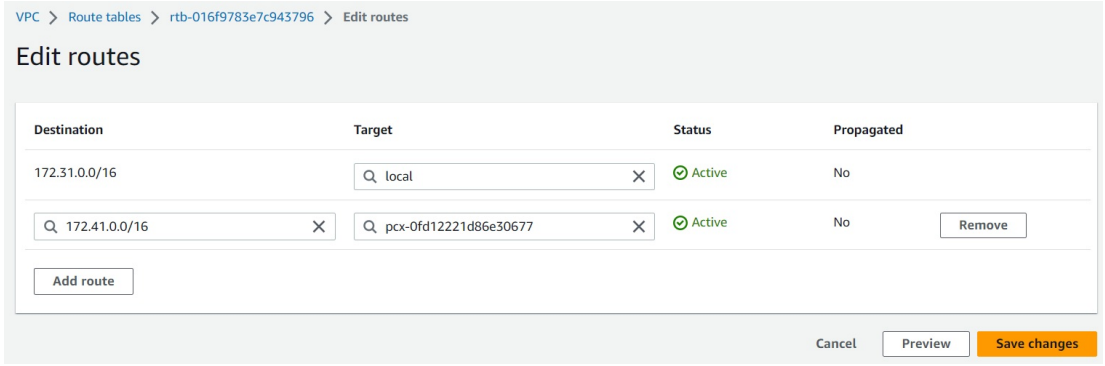


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

7. Select **Edit routes** button the next screen.



8. Add the **Destination** by entering the CIDR of the destination network.
9. Under Target, select the recently created **Peering Connection** from the list.



10. Click the **Save changes** button.

Internet Access

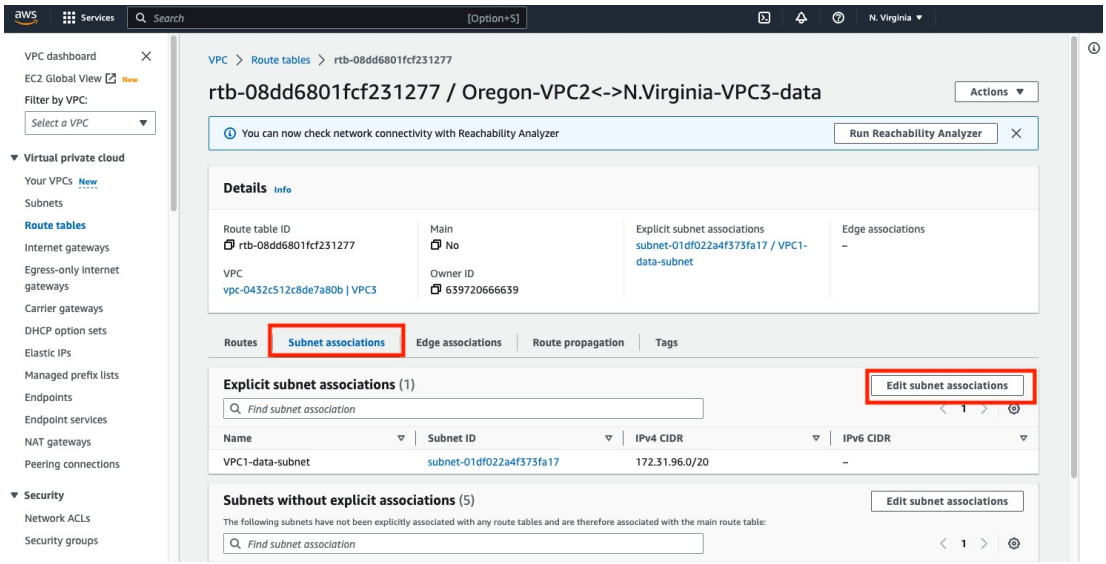
If you need the agents to have access to the internet, you will also need to add the route for the 0.0.0.0/0 towards the NAT gateway.

STEP THREE: Edit Subnet Associations

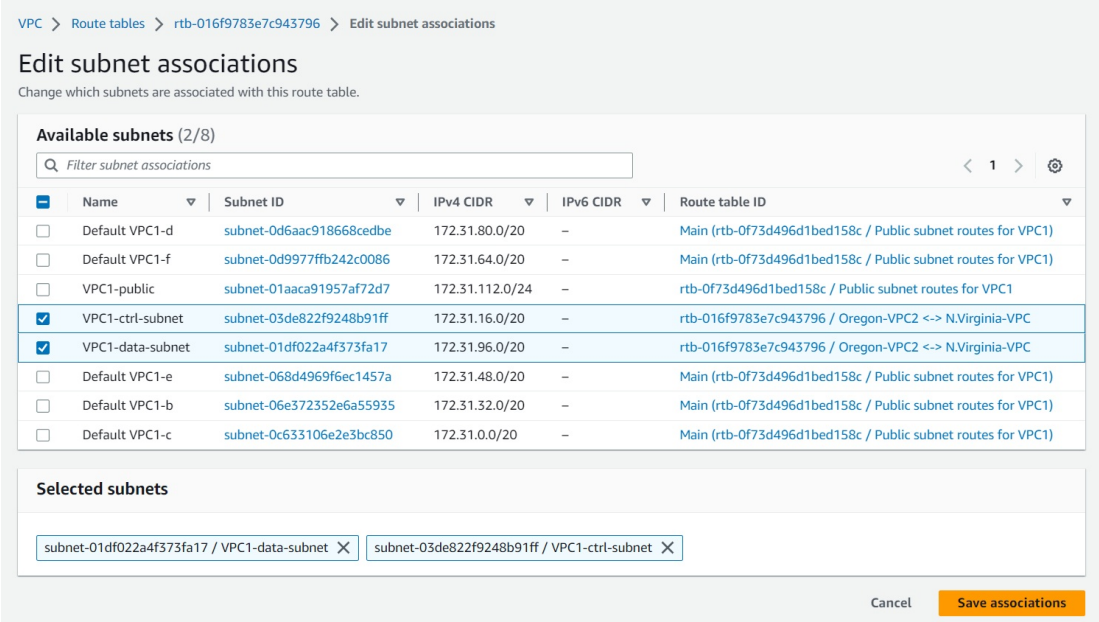
1. Select **Subnet associations** tab.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

2. Select **Edit subnet associations** button under the **Explicit subnet associations** box.



3. Select the subnet(s) of the instance that must be connected to the destination.



4. Click the **Save associations** button.

Security Groups

It is important that security groups on each EC2 and on each Subnet on both Regions match and should both encompass the port exceptions listed in the [cloudSwXtch System Requirements] article.

Repeat STEP TWO and THREE for the Other Region

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Media Use Cases

The media market can take advantage of several cloudSwXtch features such as:

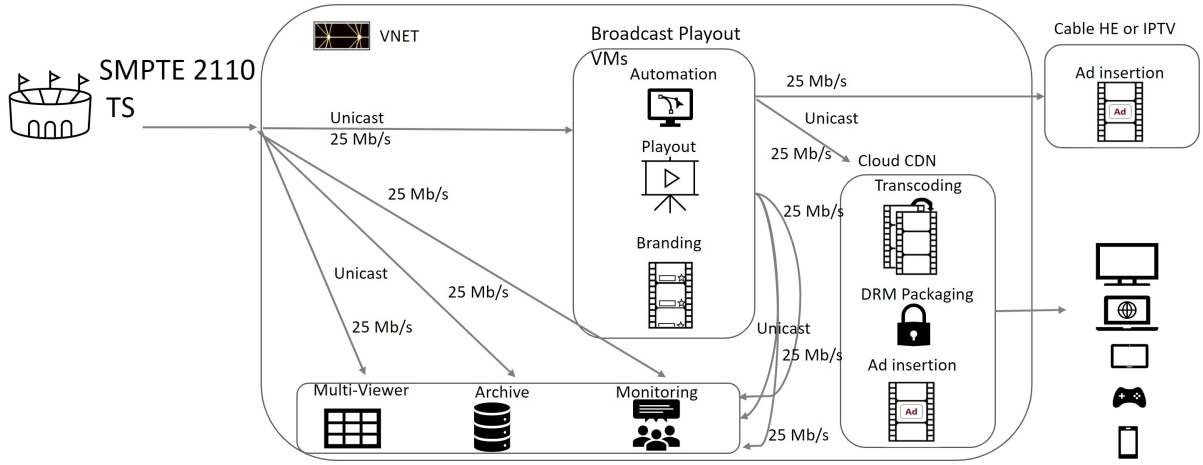
- Multicast
- Hitless Merge
- Compression support
- Protocol Fanout
- Disaster Recovery

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Media Multicast made easy with cloudswXtch

Media companies want to build dynamic workflows on the cloud, but clouds only support unicast workflows. This makes media workflows cumbersome as each stream would need to be configured for each reciever. Network provisioning and administration is complex, distributed, difficult to modify and must be be replicated for every workflow as shown below:

Unicast Playout in cloud without cloudSwXtch



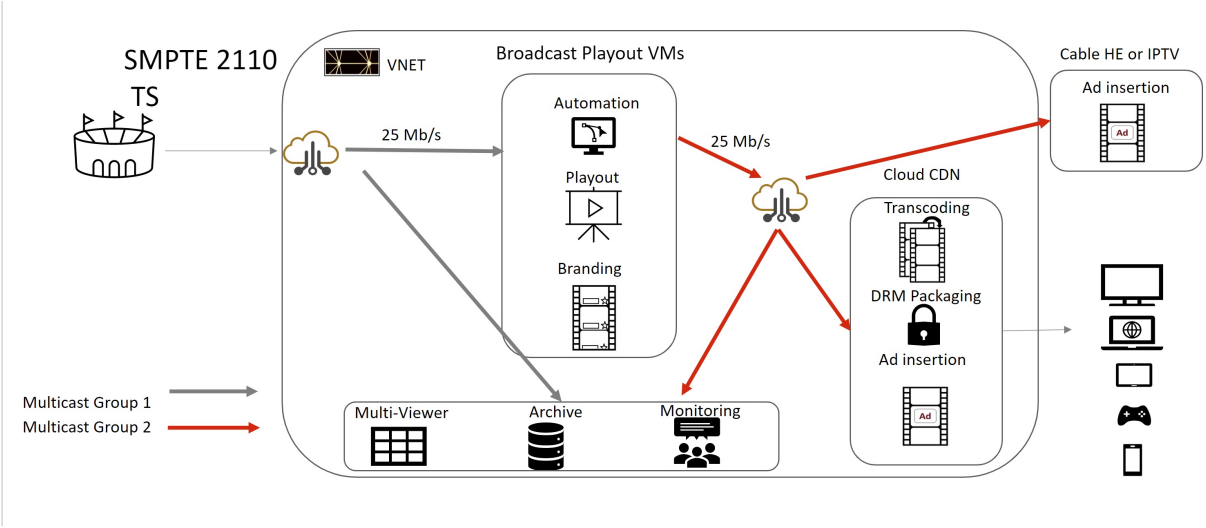
With unicast there are a number of issues:

- Network provisioning and administration is complex, distributed, difficult to modify and must be replicated for each channel or workflow
- The users cannot add endpoints without reconfiguring servers
- Larger VMs are required to support unicast which equates to higher cloud costs.
- Disaster Recovery is difficult to execute
- The load to the network is much larger
- SMPTE 2110 - 100+x more bandwidth

Multicast Playout in cloud with cloudSwXtch Multicast

cloudSwXtch enables true and seamless IP-multicast. Using multicast instead of unicast optimizes your network configuration and reduces your cloud distribution and egress costs. In addition, receivers can dynamically subscribe and unsubscribe to your streams as workflows dictate. cloudSwXtch eliminates having to configure and unconfigure unicast streams to accommodate configuration changes.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



With **cloudSwXtch** Multicast:

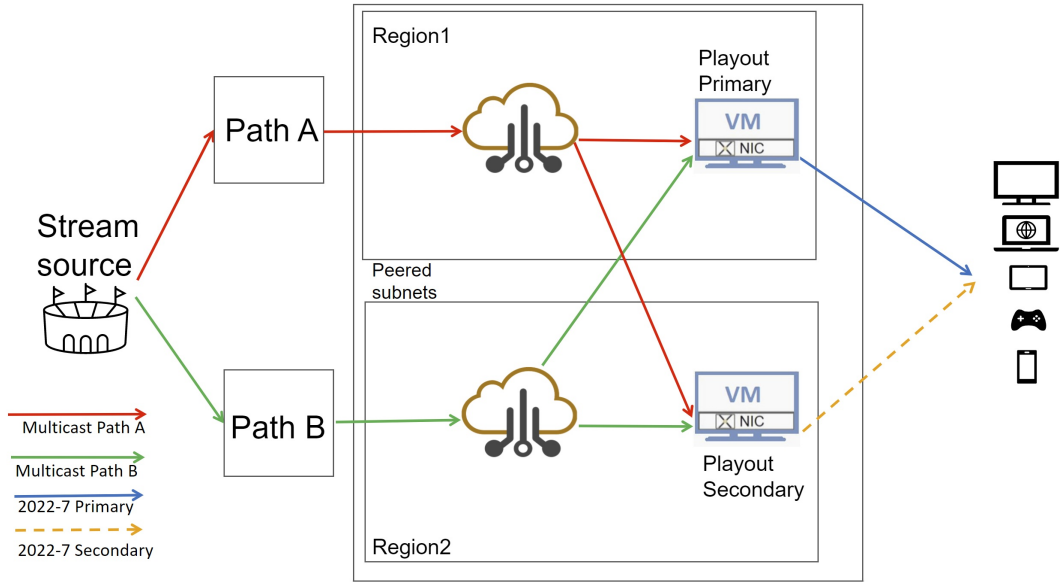
- Network may be modified and extended simply by joining multicast groups, with powerful centralized control and monitoring.
- Users can dynamically add new endpoints without playout server (or any other workflows/products) involvement.
- VM Sizes are minimized to workflow/product needs
- Disaster recovery is easy to set-up
- Minimal network load

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Hitless Merge - 2022-7

It is never good enough to have one broadcast instance, we all know things can and will go wrong. The show must always go on, media companies are used to having primary and backup streams to ensure the best user experience with NO downtime.

cloudSwXtch SMPTE 2022-7 Hitless Merge protects against data path failures by supporting two or more data paths. It compares packet reception from the multiple streams, detecting dropped packets, and reconstructs the output stream in the correct packet order.



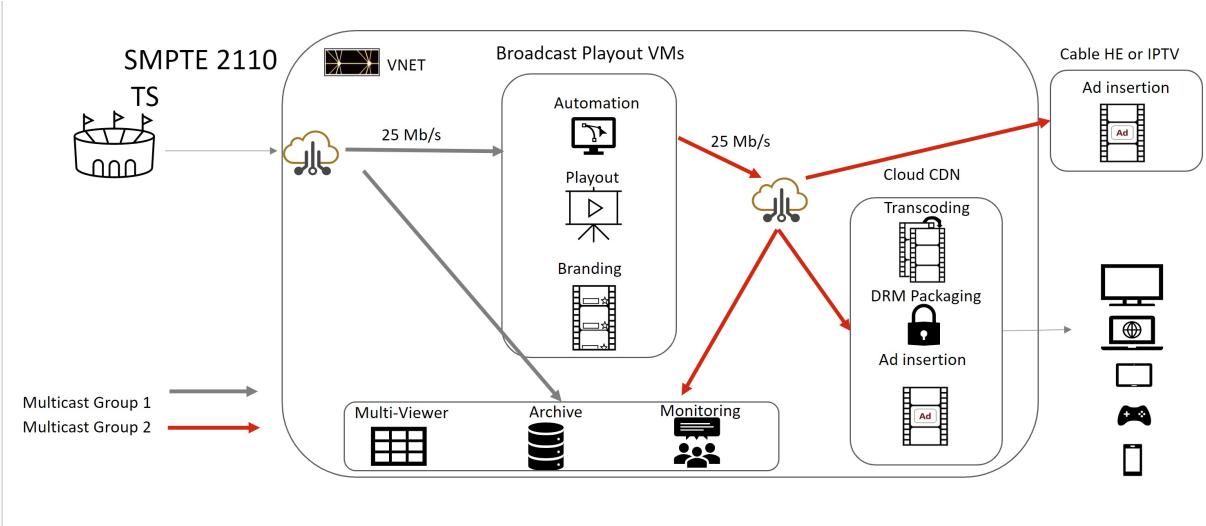
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Media support for Compressed and Uncompressed Workflows

At **swXtch.io** we know that the media companies rely highly on both compressed and uncompressed content. **cloudSwXtch** has SMPTE 2110 support without the necessity of additional gateways or other on-ramp/off-ramp appliances. The **cloudSwXtch** architecture is designed to treat content the same whether it is compressed or uncompressed. This means the ingest of streams from on-prem to the cloud and the streaming of content within the cloud, whether unicast or multicast, is the same regardless of the content type. No SDK is required for uncompressed video, and the cloud network becomes an extension of your broadcast network.

There are two workflow examples below, one is a compressed workflow and the other is an uncompressed workflow. The compressed workflow is a typical playout scenario where compressed inputs come into the cloud environment and are distributed via multicast to the necessary VM workloads by **cloudSwXtch**. All that is required is for the workloads to subscribe to the necessary multicast group(s). This eliminates the need to continually update unicast configurations to ensure your streams get to where they need to go. However, if there are workloads that only work with unicast, **cloudSwXtch** can map multicast streams to unicast devices.

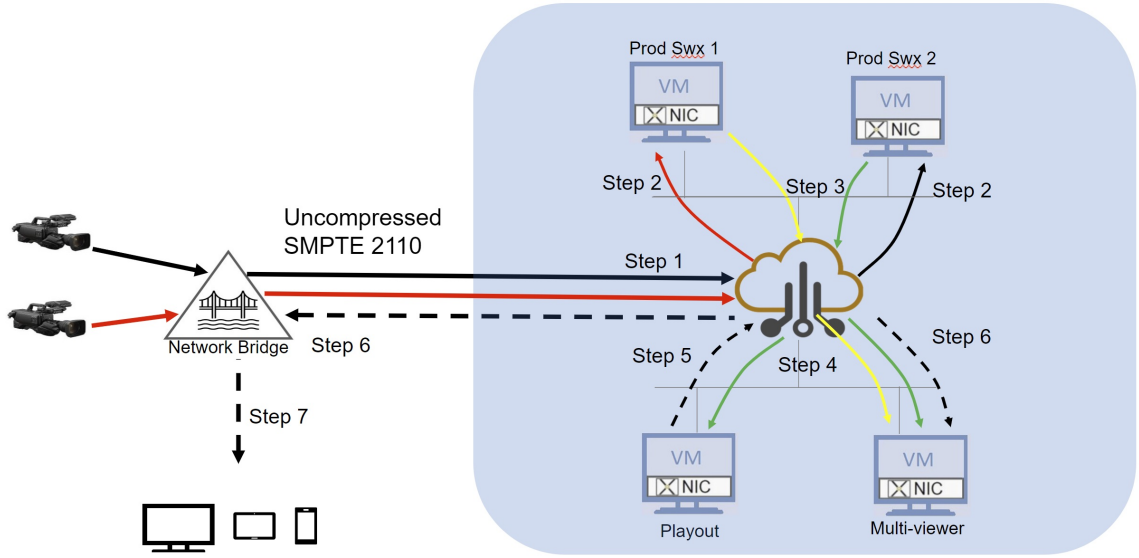
Example Compressed Playout in the Cloud with SMPTE 2110 Multicast TS



Example Uncompressed Playout in the Cloud with SMPTE 2110 Multicast

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Consider the following production workflow:



The workflow consists of a playout server which receives multiple camera feeds via 2 production switchers and determines which camera's to take to air. The **cloudSwXtch** is used to deliver the various streams via mulitcast to the workloads that subscribe to the stream:

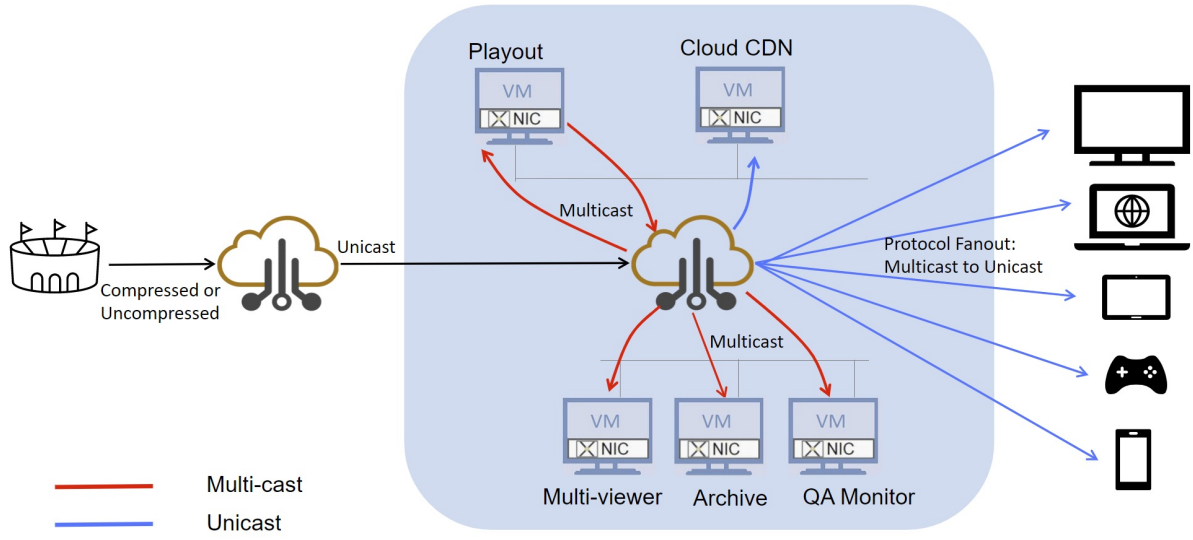
Step 1: Two inputs red and black go from Network Bridge into **cloudSwXtch**.
Step 2: Red stream goes from **cloudSwXtch** to Production Switcher 1 and black stream goes to production switcher 2.
Step 3: The modified ouptut stream from production switcher 1 is represented by the yellow path and the modified output stream from production switcher 2 is represented by the green path to the **cloudSwXtch**.
Step 4: All streams are multicasted to the multiviewer, via **cloudSwXtch**, so the director can make operational decisions.
Step 5: The playout server is directed to process and output one of the switcher outputs as represented by the dotted black to the **cloudSwXtch**.
Step 6: **cloudSwXtch** outputs the stream to the multiviewer, and the network bridge.
Step 7: The network bridge distributes to the clients for viewing consumption.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Protocol Fanout

****Media companies have many devices. Some require unicast, and some require multicast. Configuring for each device can be difficult and supporting both unicast and multicast for the same stream is impossible. Additionally multicast is not offered in the cloud see .

swXtch.io has the answer to your needs with the 'Protocol FanOut' feature which can take non-multicast packet protocols and fan them out in the same way that multicast does. It can forward a stream to many interested receivers or distribute a multicast stream to many unicast devices. This integrates unicast and multicast workflows in a way that hasn't been possible in the cloud.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Disaster Recovery

Disaster Recovery Scenerio

Coupling Hitless Merge - 2022-7 with redundant media workloads ensures high availability uptime for critical content and provides a new method to create highly available disaster recovery pathways in and between clouds.

There are many configurations that **cloudSwXtch** can recommend for redundancy, one is depicted below.

Path Redundancy

- The **cloudswXtch** in Region 1 can recieve the stream from Path A to Region 2.
- The **cloudswXtch** in Region 2 can recieve the stream from Path B to Region 1.
- If either path were to fail then the stream is still available in both Region 1 and 2 due to the redundancy.

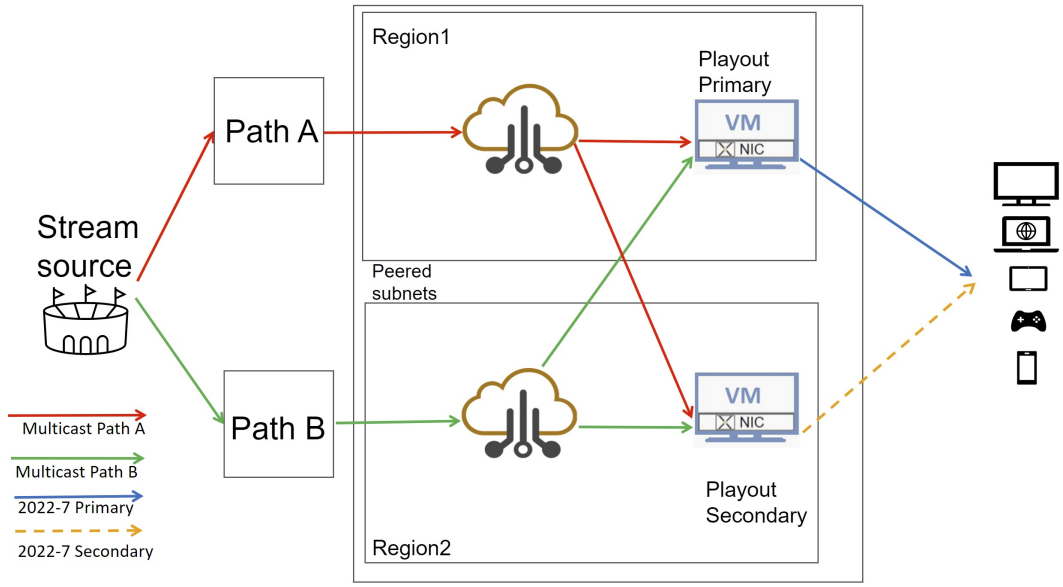
Playout Redundancy

- Each Region has a playout system, "Primary Playout" in Region 1 and "Secondary Playout" in Region 2.
- If the "Primary Playout" should fail, the stream is still playing out in the "Secondary Playout".
- As long as it is just the playout server that fails, then there is still stream redundancy from Path A and Path B.

Region Redundancy

If one region should fail the playout should still succeed in the other Region.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



This depiction only shows two stream paths, there could be a third or more. In any of these scenarios the paths could be in differnet regions or different clouds. This is done by using a **cloudSwXtch** as a Bridge between clouds or from on-prem to cloud.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Monitoring API

Overview

The cloudSwXtch Monitoring API is intended for use to integrate the cloudSwXtch data with third party tools for monitoring and dashboard purposes within customer user interfaces. This section will outline the API, with examples of data results. **Unless otherwise noted, these API calls are only applicable to cloudSwXtch versions 2.0.10 or greater.**

> **Prerequisites**
> A cloudSwXtch must exist as well as two or more agents with xNICs. To have data, agents must be producing and consuming data via the cloudSwXtch. By using a GET command, data will be provided in the response.

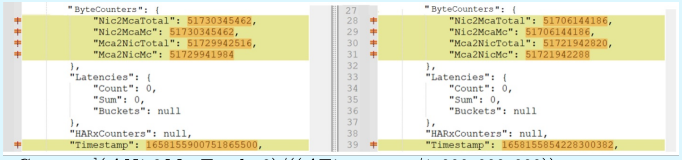
Monitoring API

The monitoring API allows developers to get data from the cloudSwXtch as well as data about its xNIC's. This data is broken down into 5 categories:

- Cloud information
- cloudSwXtch information
- cloudSwXtch status information
- xNIC status information
- xNIC totals information

To track time as a running total of seconds, the **Timestamps** are in **Unix Timestamp**. This count starts at the Unix Epoch on January 1st, 1970, at UTC. At any point in time, the API can be run, and certain metrics can be obtained from the response payload by calculating certain counter and timestamp Delta values.

> **How Monitoring Calculations Are Done (Example):**
> Below describes how to calculate from one timestamp to another using Data Ingress as an example.



$DataIngress = [xnicTotals][ByteCounters](\Delta Nic2McaTotal * 8)/((\Delta Timestamp/1,000,000,000))$

Taking the above expression and putting in the data from this call is shown below.

Timestamp from the call 2, 1658155900751865500 and the timestamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 in nanoseconds. Then, dividing the Δ by 1,000,000,000, we get the Δ of 46.523565312 in seconds, which we will use to calculate the data rate in bits per second.

$\Delta Nic2McaTotal$ is$51730345462 - 51706144186 = 7999676bits$

$[xnicTotals][ByteCounters](\Delta Mca2NicTotal * 8)/((\Delta Timestamp/1,000,000,000)) = (7999676 * 8)/46.523565312 = 4161551.392 \approx 4.2Mbps$

Cloud Information

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

GET

/swxtch/monitoring/v1/info/cloud

- Get information of the cloud for which the cloudSwXtch is installed on

URL:

Bash | Copy

```
http://<cloudSwXtch-control-IP>/swxtch/monitoring/v1/info/cloud
```

Example URL:

http://10.2.128.10/swxtch/monitoring/v1/info/cloud

Request:

Empty

Response:

200 - successful operation

Example Response:

ActionScript | Copy

```
{
    "azureData": {
        "subscriptionId": "b10209ad-ad22-4c26-9aef-be93b2f0bb58",
        "managedResourceGroupName": "TEST",
        "resourceGorupName": "TEST",
        "sku": null,
        "billingFields": null,
        "plan": "swxtch-small-002"
    },
    "awsData": null,
    "gcpData": null
}
```

cloudSwXtch Information

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

GET

/swxtch/monitoring/v1/info/swxtch

- Get information on the cloudSwXtch.

URL:

Bash | Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/monitoring/v1/info/swxtch
```

Example URL:

http://10.2.128.10/swxtch/monitoring/v1/info/swxtch

Request:

Empty

Response:

200 - successful response

Example Response:

PowerShell | Copy


```
{
    "cloudSwxtchVersion": "v2.0.7",
    "planType": "Dsd-100-50clients-1_21_2031",
    "cloudProvider": "AZURE",
    "ipAddr": "10.2.128.10",
    "swxtchGuid": "2369a922-410a-40bf-8e6e-630efe0ee70a",
    "swxtchName": "dsd-core-100",
    "hostName": "dsd-core-100",
    "numCores": 1,
    "replStatus": "running",
    "authorized": true,
    "validationResult": null,
    "isMarketplace": false,
    "expiresAt": "2031-01-21T21:51:00Z",
    "entitlements": {
        "maxClientCount": 50,
        "bandwidthMbps": 32000,
        "enableMesh": true,
        "enableUnicast": true,
        "enableHA": true,
        "enableClockSync": true,
        "enableBridge": true,
        "enableWxckedEye": true
    },
    "subnetDataPrefix": "10.2.192.0/22",
    "subnetCtrlPrefix": "10.2.128.0/22",
    "dataGatewayIp": "10.2.192.1",
    "ctrlIp": "10.2.128.10",
    "ctrlPort": 10802,
    "gatewayMacAddr": "12:34:56:78:9a:bc",
    "replInfo": {
        "ctrlIp": "1.0.0.127",
        "ctrlPort": 9996,
        "dataIp": "10.2.192.23",
        "dataPort": 9999,
        "dataMac": "YEW9qK/F"
    }
}
```


cloudSwXtch Status Information

GET

/swxtch/monitoring/v1/stats/swxtch

- Get status of the cloudSwXtch

URL:

Bash | Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/monitoring/v1/stats/swxtch
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Example URL:

http://10.2.128.10/swxtch/monitoring/v1/stats/swxtch

Request: Empty

Response:

200 - successful response

Example Response —>

PowerShell Copy


```
{
    "host": "10.2.192.23",
    "sequence": 5975,
    "rxCount": 384050,
    "txCount": 1150995,
    "rxBytes": 66853120,
    "txBytes": 200268998,
    "rxBridgeBytes": 0,
    "rxBridgeCount": 0,
    "timestamp": 1688930981555966182,
    "dropsByByteLimit": 0,
    "dropsByCountLimit": 0,
    "rxMeshPktCount": 0,
    "rxMeshBytes": 0,
    "txMeshPktCount": 0,
    "txMeshBytes": 0,
    "rxUnicastPktCount": 0,
    "rxUnicastBytes": 0,
    "txUnicastPktCount": 0,
    "txUnicastBytes": 0,
    "rxStreamLinks": [
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 64725,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:29:36.05267477Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 24558,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:27:36.052566528Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 57077,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:23:36.052356243Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 56821,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:23:36.052356243Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 49662,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:19:36.052141666Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 33275,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:17:36.051995754Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 3537,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:15:36.051889181Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 29182,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:13:36.051785945Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 2259,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:07:35.951458454Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 2003,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:07:35.951458454Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 39414,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:05:35.951336383Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 39670,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:05:35.951336383Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 28107,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:01:35.951089382Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 64742,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:59:35.950980186Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 1507,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:25:36.052463293Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 13053,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:55:35.950763225Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 1251,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:25:36.052463293Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 29438,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:13:36.051785945Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 12797,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:55:35.950763225Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 3281,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:15:36.051889181Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 36809,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:53:35.850654602Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 10475,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:51:35.850548028Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 27347,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:11:35.951684236Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 41412,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:49:35.850416183Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 27603,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:11:35.951684236Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 41668,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:49:35.850416183Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 53976,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:47:35.850300073Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 49913,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:45:35.850166797Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 49657,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:45:35.850166797Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 5089,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:43:35.850056409Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 4833,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:43:35.850056409Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 64469,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:29:36.05267477Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 30941,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:05:35.680929899Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 29138,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:57:35.950866937Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 32234,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:03:35.680820941Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 36553,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:53:35.850654602Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 28890,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:01:35.68071866Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 34247,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:17:35.648530721Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 9432,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:11:35.681354761Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 28634,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:01:35.68071866Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 3802,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:07:35.681049823Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 65275,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:15:35.731618165Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 27851,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:01:35.951089382Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 28109,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:37:35.849720001Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 47575,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:09:35.951563835Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 138,
            "nodeIp": "10.2.192.15",
            "nodePort": 35328,
            "protocol": 3,
            "pktsCount": 5,
            "bytesCount": 1375,
            "lastUpdate": "2023-07-09T18:49:54.150435686Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 137,
            "nodeIp": "10.2.192.15",
            "nodePort": 35072,
            "protocol": 3,
            "pktsCount": 18,
            "bytesCount": 2232,
            "lastUpdate": "2023-07-09T19:00:37.651032924Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 65531,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:15:35.731618165Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 3546,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:07:35.681049823Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 10731,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:51:35.850548028Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 25824,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T17:53:58.51932621Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 12486,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:21:36.052250146Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 42208,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:41:35.849951744Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 15050,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:13:35.68153882Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 49918,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:19:36.052141666Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 33019,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:17:36.051995754Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 15862,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T17:51:58.510430097Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 16118,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T17:51:58.510430097Z"
        },
        {
            "groupIp": "10.2.195.255",
            "groupPort": 138,
            "nodeIp": "10.2.192.82",
            "nodePort": 35328,
            "protocol": 3,
            "pktsCount": 11,
            "bytesCount": 3025,
            "lastUpdate": "2023-07-09T19:25:51.152475595Z"
        },
        {
            "groupIp": "10.2.195.255",
            "groupPort": 137,
            "nodeIp": "10.2.192.82",
            "nodePort": 35072,
            "protocol": 3,
            "pktsCount": 18,
            "bytesCount": 2232,
            "lastUpdate": "2023-07-09T18:59:35.850979804Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 64998,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:59:35.950980186Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 25568,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T17:53:58.51932621Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 12742,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:21:36.052250146Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 42464,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:41:35.849951744Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 15306,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:13:35.68153882Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 32221,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T17:55:58.530278444Z"
        },
        {
            "groupIp": "10.2.195.255",
            "groupPort": 138,
            "nodeIp": "10.2.192.15",
            "nodePort": 35328,
            "protocol": 3,
            "pktsCount": 5,
            "bytesCount": 1375,
            "lastUpdate": "2023-07-09T18:49:54.150435686Z"
        },
        {
            "groupIp": "10.2.195.255",
            "groupPort": 137,
            "nodeIp": "10.2.192.15",
            "nodePort": 35072,
            "protocol": 3,
            "pktsCount": 18,
            "bytesCount": 2232,
            "lastUpdate": "2023-07-09T19:00:39.951035022Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 25818,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:29:35.74923563Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 28882,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:57:35.950866937Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 31978,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:03:35.680820941Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 138,
            "nodeIp": "10.2.192.82",
            "nodePort": 35328,
            "protocol": 3,
            "pktsCount": 11,
            "bytesCount": 3025,
            "lastUpdate": "2023-07-09T19:25:51.152475595Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 137,
            "nodeIp": "10.2.192.82",
            "nodePort": 35072,
            "protocol": 3,
            "pktsCount": 19,
            "bytesCount": 2374,
            "lastUpdate": "2023-07-09T18:59:33.550977945Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 33758,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:35:35.749565124Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 62165,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:03:35.951204776Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 30685,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:05:35.680929899Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 24814,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:27:36.052566528Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 29120,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T17:59:35.780578374Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 47831,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:09:35.951563835Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 28864,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T17:59:35.780578374Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 33991,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:17:35.648530721Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 20928,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:19:35.748657226Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 20672,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:19:35.748657226Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 27109,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:09:35.681163787Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 59347,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:31:35.749353408Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 31168,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:21:35.748768568Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 26074,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:29:35.74923563Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 30912,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:21:35.748768568Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 61909,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T19:03:35.951204776Z"
        },
        {
            "groupIp": "239.1.1.1",
            "groupPort": 3490,
            "nodeIp": "10.2.192.82",
            "nodePort": 26828,
            "protocol": 3,
            "pktsCount": 383543,
            "bytesCount": 66736482,
            "lastUpdate": "2023-07-09T17:56:34.680418968Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 45268,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:23:35.748879194Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 53720,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:47:35.850300073Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 45012,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:23:35.748879194Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 9688,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:11:35.681354761Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 9446,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:25:35.749006271Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 64471,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:27:35.749121904Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 9190,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:25:35.749006271Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 32477,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T17:55:58.530278444Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 64215,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:27:35.749121904Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 26853,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:09:35.681163787Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 59091,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:31:35.749353408Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 43772,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:39:35.849824666Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 22725,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:33:35.849463701Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 22469,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:33:35.849463701Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 34014,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:35:35.749565124Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 27853,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:37:35.849720001Z"
        },
        {
            "groupIp": "239.255.255.250",
            "groupPort": 1900,
            "nodeIp": "10.2.192.82",
            "nodePort": 43516,
            "protocol": 3,
            "pktsCount": 4,
            "bytesCount": 996,
            "lastUpdate": "2023-07-09T18:39:35.849824666Z"
        }
    ],
    "txStreamLinks": [
        {
            "groupIp": "10.2.131.255",
            "groupPort": 138,
            "nodeIp": "10.2.192.82",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 5,
            "bytesCount": 1375,
            "lastUpdate": "2023-07-09T18:49:54.191912651Z"
        },
        {
            "groupIp": "10.2.195.255",
            "groupPort": 138,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "nodeIp": "10.2.192.82",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 5,
            "bytesCount": 1375,
            "lastUpdate": "2023-07-09T18:49:54.191912651Z"
        },
        {
            "groupIp": "10.2.195.255",
            "groupPort": 137,
            "nodeIp": "10.2.192.82",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 18,
            "bytesCount": 2232,
            "lastUpdate": "2023-07-09T19:00:39.892273902Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 137,
            "nodeIp": "10.2.192.82",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 18,
            "bytesCount": 2232,
            "lastUpdate": "2023-07-09T19:00:37.592268943Z"
        },
        {
            "groupIp": "10.2.195.255",
            "groupPort": 137,
            "nodeIp": "10.2.192.15",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 18,
            "bytesCount": 2232,
            "lastUpdate": "2023-07-09T18:59:35.792238473Z"
        },
        {
            "groupIp": "10.2.195.255",
            "groupPort": 137,
            "nodeIp": "10.2.192.5",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 36,
            "bytesCount": 4464,
            "lastUpdate": "2023-07-09T19:00:39.892273902Z"
        },
        {
            "groupIp": "239.1.1.1",
            "groupPort": 3490,
            "nodeIp": "10.5.2.4",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 383543,
            "bytesCount": 66736482,
            "lastUpdate": "2023-07-09T17:56:34.617266416Z"
        },
        {
            "groupIp": "239.1.1.1",
            "groupPort": 3490,
            "nodeIp": "10.2.192.5",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 383543,
            "bytesCount": 66736482,
            "lastUpdate": "2023-07-09T17:56:34.617266416Z"
        },
        {
            "groupIp": "239.1.1.1",
            "groupPort": 3490,
            "nodeIp": "10.2.192.15",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 383543,
            "bytesCount": 66736482,
            "lastUpdate": "2023-07-09T17:56:34.617266416Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 138,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "nodeIp": "10.2.192.5",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 16,
            "bytesCount": 4400,
            "lastUpdate": "2023-07-09T19:25:51.193130493Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 137,
            "nodeIp": "10.2.192.15",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 19,
            "bytesCount": 2374,
            "lastUpdate": "2023-07-09T18:59:33.492237567Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 138,
            "nodeIp": "10.2.192.15",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 11,
            "bytesCount": 3025,
            "lastUpdate": "2023-07-09T19:25:51.193130493Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 138,
            "nodeIp": "10.2.128.96",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 16,
            "bytesCount": 4400,
            "lastUpdate": "2023-07-09T19:25:51.193130493Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 137,
            "nodeIp": "10.2.128.96",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 37,
            "bytesCount": 4606,
            "lastUpdate": "2023-07-09T19:00:37.592268943Z"
        },
        {
            "groupIp": "10.2.195.255",
            "groupPort": 138,
            "nodeIp": "10.2.192.5",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 16,
            "bytesCount": 4400,
            "lastUpdate": "2023-07-09T19:25:51.193130493Z"
        },
        {
            "groupIp": "10.2.195.255",
            "groupPort": 138,
            "nodeIp": "10.2.192.15",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 11,
            "bytesCount": 3025,
            "lastUpdate": "2023-07-09T19:25:51.193130493Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 138,
            "nodeIp": "10.2.128.95",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 16,
            "bytesCount": 4400,
            "lastUpdate": "2023-07-09T19:25:51.193130493Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 137,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "nodeIp": "10.2.192.5",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 37,
            "bytesCount": 4606,
            "lastUpdate": "2023-07-09T19:00:37.592268943Z"
        },
        {
            "groupIp": "10.2.131.255",
            "groupPort": 137,
            "nodeIp": "10.2.128.95",
            "nodePort": 0,
            "protocol": 3,
            "pktsCount": 37,
            "bytesCount": 4606,
            "lastUpdate": "2023-07-09T19:00:37.592268943Z"
        }
    ]
}
```


xNIC Status Totals Information

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

GET

/swxtch/monitoring/v1/stats/xnicsTotals

- Get status of the xNIC totals

URL:

Bash Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/monitoring/v1/stats/xnicsTotals
```

Example URL:

http://10.2.128.10/swxtch/monitoring/v1/stats/xnicsTotals

Request:

Empty

Response:

200 - successful response

Example Response:

PowerShell Copy


```
{
    "activeConnections": 0,
    "byteCounters": {
        "rxMulticast": 8271583008,
        "rxTotal": 8271583008,
        "txMulticast": 254920,
        "txTotal": 272876
    },
    "cloudSwxtchVersion": "",
    "ipAddr": "",
    "latencies": {
        "buckets": {},
        "count": 0,
        "sum": 0
    },
    "maxActiveConnections": 0,
    "packetCounters": {
        "rxDrops": 391884,
        "rxMulticast": 37815581,
        "rxTotal": 37815581,
        "txDrops": 0,
        "txIgmp": 160,
        "txMulticast": 1152,
        "txTotal": 1306
    },
    "rxHaCounters": [],
    "rxMulticastGroups": [],
    "timestamp": 1688931006635816100,
    "txMulticastGroups": [],
    "virtualTunIpAddr": "",
    "xNicMode": "",
    "xNicVersion": 0
}
```


xNIC Status Information

GET

/swxtch/monitoring/v1/stats/xnics

- Get status of the xNICs

URL:

Bash Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/monitoring/v1/stats/xnics
```

Example URL:

http://10.2.128.10/swxtch/monitoring/v1/stats/xnics

Request:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Empty

Response:

200- successful response

Example Response —>

ActionScript Copy


```
{
    "DSd-agent-101": {
        "activeConnections": 2,
        "byteCounters": {
            "rxMulticast": 4772448644,
            "rxTotal": 4772448644,
            "txMulticast": 1428,
            "txTotal": 1236
        },
        "cloudSwxtchVersion": "v2.0.7",
        "ipAddr": "10.2.192.5",
        "latencies": {
            "buckets": {
                "0.000050": 0,
                "0.000100": 0,
                "0.000250": 0,
                "0.000500": 0,
                "0.001000": 0,
                "0.002000": 0,
                "0.005000": 0,
                "0.010000": 0,
                "0.011000": 0
            },
            "count": 0,
            "sum": 0
        },
        "maxActiveConnections": 2,
        "packetCounters": {
            "rxDrops": 0,
            "rxMulticast": 27427862,
            "rxTotal": 27427862,
            "txDrops": 0,
            "txIgmp": 6,
            "txMulticast": 12,
            "txTotal": 12
        },
        "rxHaCounters": [],
        "rxMulticastGroups": [
            {
                "bytesCount": 0,
                "groupIp": "224.0.1.129",
                "lastUpdate": "2023-07-09T19:30:25.688747882Z",
                "numberOfDestinations": 0,
                "pktsCount": 0,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 26028,
                "groupIp": "10.2.131.255",
                "lastUpdate": "2023-07-09T19:30:25.688749074Z",
                "numberOfDestinations": 0,
                "pktsCount": 144,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 4772394048,
                "groupIp": "239.1.1.1",
                "lastUpdate": "2023-07-09T19:30:25.688749313Z",
                "numberOfDestinations": 0,
                "pktsCount": 27427552,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
                "bytesCount": 26010,
                "groupIp": "10.2.195.255",
                "lastUpdate": "2023-07-09T19:30:25.688749313Z",
                "numberOfDestinations": 0,
                "pktsCount": 144,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 2558,
                "groupIp": "224.0.0.251",
                "lastUpdate": "2023-07-09T19:30:25.688749551Z",
                "numberOfDestinations": 0,
                "pktsCount": 22,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            }
        ],
        "timestamp": 1688931025688668047,
        "txMulticastGroups": [
            {
                "bytesCount": 1428,
                "groupIp": "224.0.0.251",
                "lastUpdate": "2023-07-09T19:30:25.688750743Z",
                "numberOfDestinations": 0,
                "pktsCount": 12,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            }
        ],
        "virtualTunIpAddr": "",
        "xNicMode": "Normal",
        "xNicVersion": 2
    },
    "DSd-agent-104": {
        "activeConnections": 2,
        "byteCounters": {
            "rxMulticast": 8374,
            "rxTotal": 8374,
            "txMulticast": 204578,
            "txTotal": 205328
        },
        "cloudSwxtchVersion": "v2.0.7",
        "ipAddr": "10.2.192.82",
        "latencies": {
            "buckets": {
                "0.000050": 56,
                "0.000100": 56,
                "0.000250": 56,
                "0.000500": 56,
                "0.001000": 56,
                "0.002000": 56,
                "0.005000": 56,
                "0.010000": 56,
                "0.011000": 56
            },
            "count": 56,
            "sum": 0
        },
        "maxActiveConnections": 2,
        "packetCounters": {
            "rxDrops": 0,
            "rxMulticast": 56,
            "rxTotal": 56,
            "txDrops": 0,
            "txIgmp": 7,
            "txMulticast": 860,
            "txTotal": 867
        },
        "rxHaCounters": [
            {
                "egressBytes": 684,
                "egressCount": 6,
                "enqueueFailureCount": 0,
                "groupIp": "224.0.0.251",
                "outputStreamLoss": 0,
                "paths": [
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
                {
                    "ingressBytes": 684,
                    "ingressCount": 6,
                    "missingPackets": 0,
                    "packetsUsed": 3
                },
                {
                    "ingressBytes": 684,
                    "ingressCount": 6,
                    "missingPackets": 0,
                    "packetsUsed": 3
                }
            ],
            "senderIp": "10.2.192.82"
        },
        {
            "egressBytes": 3607,
            "egressCount": 23,
            "enqueueFailureCount": 0,
            "groupIp": "10.2.195.255",
            "outputStreamLoss": 0,
            "paths": [
                {
                    "ingressBytes": 3607,
                    "ingressCount": 23,
                    "missingPackets": 0,
                    "packetsUsed": 14
                },
                {
                    "ingressBytes": 2491,
                    "ingressCount": 14,
                    "missingPackets": 9,
                    "packetsUsed": 9
                }
            ],
            "senderIp": "10.2.192.15"
        },
        {
            "egressBytes": 3607,
            "egressCount": 23,
            "enqueueFailureCount": 0,
            "groupIp": "10.2.131.255",
            "outputStreamLoss": 0,
            "paths": [
                {
                    "ingressBytes": 3607,
                    "ingressCount": 23,
                    "missingPackets": 0,
                    "packetsUsed": 13
                },
                {
                    "ingressBytes": 2491,
                    "ingressCount": 14,
                    "missingPackets": 9,
                    "packetsUsed": 10
                }
            ],
            "senderIp": "10.2.192.15"
        },
        {
            "egressBytes": 476,
            "egressCount": 4,
            "enqueueFailureCount": 0,
            "groupIp": "224.0.0.251",
            "outputStreamLoss": 0,
            "paths": [
                {
                    "ingressBytes": 476,
                    "ingressCount": 4,
                    "missingPackets": 0,
                    "packetsUsed": 4
                },
                {
                    "ingressBytes": 238,
                    "ingressCount": 2,
                    "missingPackets": 2,
                    "packetsUsed": 0
                }
            ],
            "senderIp": "10.2.192.5"
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            }
        ],
        "rxMulticastGroups": [
            {
                "bytesCount": 1160,
                "groupIp": "224.0.0.251",
                "lastUpdate": "2023-07-09T19:30:26.080212116Z",
                "numberOfDestinations": 0,
                "pktsCount": 10,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 3607,
                "groupIp": "10.2.131.255",
                "lastUpdate": "2023-07-09T19:30:26.080212116Z",
                "numberOfDestinations": 0,
                "pktsCount": 23,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 3607,
                "groupIp": "10.2.195.255",
                "lastUpdate": "2023-07-09T19:30:26.080212593Z",
                "numberOfDestinations": 0,
                "pktsCount": 23,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 0,
                "groupIp": "224.0.1.129",
                "lastUpdate": "2023-07-09T19:30:26.080212831Z",
                "numberOfDestinations": 0,
                "pktsCount": 0,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            }
        ],
        "timestamp": 1688931026080174100,
        "txMulticastGroups": [
            {
                "bytesCount": 5124,
                "groupIp": "10.2.131.255",
                "lastUpdate": "2023-07-09T19:30:26.080216407Z",
                "numberOfDestinations": 0,
                "pktsCount": 29,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 4982,
                "groupIp": "10.2.195.255",
                "lastUpdate": "2023-07-09T19:30:26.080216646Z",
                "numberOfDestinations": 0,
                "pktsCount": 28,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 91632,
                "groupIp": "239.255.255.250",
                "lastUpdate": "2023-07-09T19:30:26.080216646Z",
                "numberOfDestinations": 0,
                "pktsCount": 368,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 684,
                "groupIp": "224.0.0.251",
                "lastUpdate": "2023-07-09T19:30:26.080216884Z",
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
                "numberOfDestinations": 0,
                "pktsCount": 6,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            }
        ],
        "virtualTunIpAddr": "",
        "xNicMode": "HA",
        "xNicVersion": 2
    },
    "DSd-agent-105": {
        "activeConnections": 2,
        "byteCounters": {
            "rxMulticast": 1476244598,
            "rxTotal": 1476244598,
            "txMulticast": 48914,
            "txTotal": 60672
        },
        "cloudSwxtchVersion": "v2.0.7",
        "ipAddr": "10.2.192.15",
        "latencies": {
            "buckets": {
                "0.000050": 3936098,
                "0.000100": 3936098,
                "0.000250": 3936098,
                "0.000500": 3936098,
                "0.001000": 3936098,
                "0.002000": 3936098,
                "0.005000": 3936098,
                "0.010000": 3936098,
                "0.011000": 3936098
            },
            "count": 4500553,
            "sum": 12533584584855750000
        },
        "maxActiveConnections": 2,
        "packetCounters": {
            "rxDrops": 391884,
            "rxMulticast": 4500553,
            "rxTotal": 4500553,
            "txDrops": 0,
            "txIgmp": 87,
            "txMulticast": 280,
            "txTotal": 367
        },
        "rxHaCounters": [
            {
                "egressBytes": 684846774,
                "egressCount": 3935901,
                "enqueueFailureCount": 0,
                "groupIp": "239.1.1.1",
                "outputStreamLoss": 24,
                "paths": [
                    {
                        "ingressBytes": 3923642754,
                        "ingressCount": 22549671,
                        "missingPackets": 14199,
                        "packetsUsed": 2846789
                    },
                    {
                        "ingressBytes": 3879332958,
                        "ingressCount": 22295017,
                        "missingPackets": 155,
                        "packetsUsed": 1089112
                    }
                ],
                "senderIp": "10.2.192.82"
            },
            {
                "egressBytes": 15201,
                "egressCount": 92,
                "enqueueFailureCount": 0,
                "groupIp": "10.2.131.255",
                "outputStreamLoss": 0,
                "paths": [
                    {
                        "ingressBytes": 15201,
                        "ingressCount": 92,
                        "missingPackets": 0,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
                    "packetsUsed": 74
                },
                {
                    "ingressBytes": 10115,
                    "ingressCount": 56,
                    "missingPackets": 36,
                    "packetsUsed": 18
                }
            ],
            "senderIp": "10.2.192.82"
        },
        {
            "egressBytes": 15059,
            "egressCount": 91,
            "enqueueFailureCount": 0,
            "groupIp": "10.2.195.255",
            "outputStreamLoss": 0,
            "paths": [
                {
                    "ingressBytes": 15059,
                    "ingressCount": 91,
                    "missingPackets": 0,
                    "packetsUsed": 70
                },
                {
                    "ingressBytes": 10239,
                    "ingressCount": 57,
                    "missingPackets": 34,
                    "packetsUsed": 21
                }
            ],
            "senderIp": "10.2.192.82"
        },
        {
            "egressBytes": 684,
            "egressCount": 6,
            "enqueueFailureCount": 0,
            "groupIp": "224.0.0.251",
            "outputStreamLoss": 0,
            "paths": [
                {
                    "ingressBytes": 684,
                    "ingressCount": 6,
                    "missingPackets": 0,
                    "packetsUsed": 2
                },
                {
                    "ingressBytes": 684,
                    "ingressCount": 6,
                    "missingPackets": 0,
                    "packetsUsed": 4
                }
            ],
            "senderIp": "10.2.192.82"
        },
        {
            "egressBytes": 336749184,
            "egressCount": 240192,
            "enqueueFailureCount": 0,
            "groupIp": "239.1.1.4",
            "outputStreamLoss": 214148,
            "paths": [
                {
                    "ingressBytes": 282462342,
                    "ingressCount": 201471,
                    "missingPackets": 252869,
                    "packetsUsed": 196533
                },
                {
                    "ingressBytes": 231122504,
                    "ingressCount": 164852,
                    "missingPackets": 398897,
                    "packetsUsed": 43659
                }
            ],
            "senderIp": "172.30.0.1"
        },
        {
            "egressBytes": 455379414,
            "egressCount": 324807,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
                "enqueueFailureCount": 0,
                "groupIp": "239.1.1.5",
                "outputStreamLoss": 293739,
                "paths": [
                    {
                        "ingressBytes": 392498312,
                        "ingressCount": 279956,
                        "missingPackets": 384223,
                        "packetsUsed": 228446
                    },
                    {
                        "ingressBytes": 344886392,
                        "ingressCount": 245996,
                        "missingPackets": 506294,
                        "packetsUsed": 96361
                    }
                ],
                "senderIp": "172.30.0.1"
            },
            {
                "egressBytes": 714,
                "egressCount": 6,
                "enqueueFailureCount": 0,
                "groupIp": "224.0.0.251",
                "outputStreamLoss": 0,
                "paths": [
                    {
                        "ingressBytes": 714,
                        "ingressCount": 6,
                        "missingPackets": 0,
                        "packetsUsed": 4
                    },
                    {
                        "ingressBytes": 476,
                        "ingressCount": 4,
                        "missingPackets": 2,
                        "packetsUsed": 2
                    }
                ],
                "senderIp": "10.2.192.5"
            },
            {
                "egressBytes": 256,
                "egressCount": 2,
                "enqueueFailureCount": 0,
                "groupIp": "224.0.0.252",
                "outputStreamLoss": 0,
                "paths": [
                    {
                        "ingressBytes": 232704,
                        "ingressCount": 1818,
                        "missingPackets": 0,
                        "packetsUsed": 2
                    },
                    {
                        "ingressBytes": 0,
                        "ingressCount": 0,
                        "missingPackets": 2,
                        "packetsUsed": 0
                    }
                ],
                "senderIp": "172.30.0.7"
            }
        ],
        "rxMulticastGroups": [
            {
                "bytesCount": 0,
                "groupIp": "224.0.1.129",
                "lastUpdate": "2023-07-09T19:30:26.29428482Z",
                "numberOfDestinations": 0,
                "pktsCount": 0,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 15059,
                "groupIp": "10.2.195.255",
                "lastUpdate": "2023-07-09T19:30:26.294285297Z",
                "numberOfDestinations": 0,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "pktsCount": 91,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 15201,
            "groupIp": "10.2.131.255",
            "lastUpdate": "2023-07-09T19:30:26.294285535Z",
            "numberOfDestinations": 0,
            "pktsCount": 92,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 684846774,
            "groupIp": "239.1.1.1",
            "lastUpdate": "2023-07-09T19:30:26.294285774Z",
            "numberOfDestinations": 0,
            "pktsCount": 3935901,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 336373448,
            "groupIp": "239.1.1.4",
            "lastUpdate": "2023-07-09T19:30:26.294287204Z",
            "numberOfDestinations": 0,
            "pktsCount": 239924,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 454992462,
            "groupIp": "239.1.1.5",
            "lastUpdate": "2023-07-09T19:30:26.294287681Z",
            "numberOfDestinations": 0,
            "pktsCount": 324531,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 256,
            "groupIp": "224.0.0.252",
            "lastUpdate": "2023-07-09T19:30:26.294287681Z",
            "numberOfDestinations": 0,
            "pktsCount": 2,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 1398,
            "groupIp": "224.0.0.251",
            "lastUpdate": "2023-07-09T19:30:26.294287919Z",
            "numberOfDestinations": 0,
            "pktsCount": 12,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        }
    ],
    "timestamp": 1688931026294220300,
    "txMulticastGroups": [
        {
            "bytesCount": 13668,
            "groupIp": "10.2.195.255",
            "lastUpdate": "2023-07-09T19:30:26.294292211Z",
            "numberOfDestinations": 0,
            "pktsCount": 81,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 13668,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
                "groupIp": "10.2.131.255",
                "lastUpdate": "2023-07-09T19:30:26.294293403Z",
                "numberOfDestinations": 0,
                "pktsCount": 81,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            }
        ],
        "virtualTunIpAddr": "",
        "xNicMode": "HA",
        "xNicVersion": 2
    },
    "DSd-agent-201": {
        "activeConnections": 2,
        "byteCounters": {
            "rxMulticast": 1010860274,
            "rxTotal": 1010860274,
            "txMulticast": 0,
            "txTotal": 2616
        },
        "cloudSwxtchVersion": "v2.0.7",
        "ipAddr": "10.5.2.4",
        "latencies": {
            "buckets": {
                "0.000050": 3806272,
                "0.000100": 3806272,
                "0.000250": 3806272,
                "0.000500": 3806272,
                "0.001000": 3806272,
                "0.002000": 3806272,
                "0.005000": 3806272,
                "0.010000": 3806272,
                "0.011000": 3806272
            },
            "count": 4054895,
            "sum": 14075100747826217000
        },
        "maxActiveConnections": 2,
        "packetCounters": {
            "rxDrops": 0,
            "rxMulticast": 4054895,
            "rxTotal": 4054895,
            "txDrops": 0,
            "txIgmp": 28,
            "txMulticast": 0,
            "txTotal": 28
        },
        "rxHaCounters": [
            {
                "egressBytes": 462,
                "egressCount": 6,
                "enqueueFailureCount": 0,
                "groupIp": "224.0.0.251",
                "outputStreamLoss": 0,
                "paths": [
                    {
                        "ingressBytes": 462,
                        "ingressCount": 6,
                        "missingPackets": 0,
                        "packetsUsed": 2
                    },
                    {
                        "ingressBytes": 308,
                        "ingressCount": 4,
                        "missingPackets": 2,
                        "packetsUsed": 4
                    }
                ],
                "senderIp": "10.2.192.5"
            },
            {
                "egressBytes": 432,
                "egressCount": 6,
                "enqueueFailureCount": 0,
                "groupIp": "224.0.0.251",
                "outputStreamLoss": 0,
                "paths": [
                    {
                        "ingressBytes": 432,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
                    "ingressCount": 6,
                    "missingPackets": 0,
                    "packetsUsed": 4
                },
                {
                    "ingressBytes": 432,
                    "ingressCount": 6,
                    "missingPackets": 0,
                    "packetsUsed": 2
                }
            ],
            "senderIp": "10.2.192.82"
        },
        {
            "egressBytes": 502426056,
            "egressCount": 3806258,
            "enqueueFailureCount": 0,
            "groupIp": "239.1.1.1",
            "outputStreamLoss": 2014,
            "paths": [
                {
                    "ingressBytes": 2368540944,
                    "ingressCount": 17943492,
                    "missingPackets": 16238,
                    "packetsUsed": 3091703
                },
                {
                    "ingressBytes": 2337372576,
                    "ingressCount": 17707368,
                    "missingPackets": 4214,
                    "packetsUsed": 714555
                }
            ],
            "senderIp": "10.2.192.82"
        },
        {
            "egressBytes": 338127280,
            "egressCount": 248623,
            "enqueueFailureCount": 0,
            "groupIp": "239.1.1.4",
            "outputStreamLoss": 203985,
            "paths": [
                {
                    "ingressBytes": 277442720,
                    "ingressCount": 204002,
                    "missingPackets": 248606,
                    "packetsUsed": 191715
                },
                {
                    "ingressBytes": 228637760,
                    "ingressCount": 168116,
                    "missingPackets": 377704,
                    "packetsUsed": 56908
                }
            ],
            "senderIp": "172.30.0.1"
        },
        {
            "egressBytes": 454,
            "egressCount": 2,
            "enqueueFailureCount": 0,
            "groupIp": "224.0.0.251",
            "outputStreamLoss": 0,
            "paths": [
                {
                    "ingressBytes": 454,
                    "ingressCount": 2,
                    "missingPackets": 0,
                    "packetsUsed": 2
                },
                {
                    "ingressBytes": 0,
                    "ingressCount": 0,
                    "missingPackets": 2,
                    "packetsUsed": 0
                }
            ],
            "senderIp": "10.2.192.11"
        }
    ],
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
        "rxMulticastGroups": [
            {
                "bytesCount": 0,
                "groupIp": "224.0.1.129",
                "lastUpdate": "2023-07-09T19:30:26.623338222Z",
                "numberOfDestinations": 0,
                "pktsCount": 0,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 662288892,
                "groupIp": "239.1.1.1",
                "lastUpdate": "2023-07-09T19:30:26.623339891Z",
                "numberOfDestinations": 0,
                "pktsCount": 3806258,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 1936,
                "groupIp": "224.0.0.251",
                "lastUpdate": "2023-07-09T19:30:26.623340368Z",
                "numberOfDestinations": 0,
                "pktsCount": 14,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 348569446,
                "groupIp": "239.1.1.4",
                "lastUpdate": "2023-07-09T19:30:26.623340606Z",
                "numberOfDestinations": 0,
                "pktsCount": 248623,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            }
        ],
        "timestamp": 1688931026623304861,
        "txMulticastGroups": [],
        "virtualTunIpAddr": "",
        "xNicMode": "HA",
        "xNicVersion": 1
    },
    "DSd-agent-204": {
        "activeConnections": 2,
        "byteCounters": {
            "rxMulticast": 791824616,
            "rxTotal": 791824616,
            "txMulticast": 0,
            "txTotal": 3024
        },
        "cloudSwxtchVersion": "v2.0.7",
        "ipAddr": "10.5.2.7",
        "latencies": {
            "buckets": {
                "0.000050": 8802,
                "0.000100": 24927,
                "0.000250": 88361,
                "0.000500": 165728,
                "0.001000": 191754,
                "0.002000": 232660,
                "0.005000": 270569,
                "0.010000": 310098,
                "0.011000": 316318
            },
            "count": 564795,
            "sum": 15895895175075121000
        },
        "maxActiveConnections": 2,
        "packetCounters": {
            "rxDrops": 0,
            "rxMulticast": 564795,
            "rxTotal": 564795,
            "txDrops": 0,
            "txIgmp": 32,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "txMulticast": 0,
            "txTotal": 32
        },
        "rxHaCounters": [
            {
                "egressBytes": 470354640,
                "egressCount": 345849,
                "enqueueFailureCount": 0,
                "groupIp": "239.1.1.4",
                "outputStreamLoss": 283933,
                "paths": [
                    {
                        "ingressBytes": 415350800,
                        "ingressCount": 305405,
                        "missingPackets": 368669,
                        "packetsUsed": 248230
                    },
                    {
                        "ingressBytes": 341583040,
                        "ingressCount": 251164,
                        "missingPackets": 515891,
                        "packetsUsed": 97619
                    }
                ],
                "senderIp": "172.30.0.1"
            },
            {
                "egressBytes": 297747520,
                "egressCount": 218932,
                "enqueueFailureCount": 0,
                "groupIp": "239.1.1.5",
                "outputStreamLoss": 207786,
                "paths": [
                    {
                        "ingressBytes": 255231200,
                        "ingressCount": 187670,
                        "missingPackets": 239048,
                        "packetsUsed": 184291
                    },
                    {
                        "ingressBytes": 207783520,
                        "ingressCount": 152782,
                        "missingPackets": 384316,
                        "packetsUsed": 34641
                    }
                ],
                "senderIp": "172.30.0.1"
            },
            {
                "egressBytes": 462,
                "egressCount": 6,
                "enqueueFailureCount": 0,
                "groupIp": "224.0.0.251",
                "outputStreamLoss": 0,
                "paths": [
                    {
                        "ingressBytes": 462,
                        "ingressCount": 6,
                        "missingPackets": 0,
                        "packetsUsed": 4
                    },
                    {
                        "ingressBytes": 308,
                        "ingressCount": 4,
                        "missingPackets": 2,
                        "packetsUsed": 2
                    }
                ],
                "senderIp": "10.2.192.5"
            },
            {
                "egressBytes": 172,
                "egressCount": 2,
                "enqueueFailureCount": 0,
                "groupIp": "224.0.0.252",
                "outputStreamLoss": 0,
                "paths": [
                    {
                        "ingressBytes": 156348,
                        "ingressCount": 1818,
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
                    "missingPackets": 0,
                    "packetsUsed": 2
                },
                {
                    "ingressBytes": 0,
                    "ingressCount": 0,
                    "missingPackets": 2,
                    "packetsUsed": 0
                }
            ],
            "senderIp": "172.30.0.7"
        },
        {
            "egressBytes": 432,
            "egressCount": 6,
            "enqueueFailureCount": 0,
            "groupIp": "224.0.0.251",
            "outputStreamLoss": 0,
            "paths": [
                {
                    "ingressBytes": 432,
                    "ingressCount": 6,
                    "missingPackets": 0,
                    "packetsUsed": 4
                },
                {
                    "ingressBytes": 432,
                    "ingressCount": 6,
                    "missingPackets": 0,
                    "packetsUsed": 2
                }
            ],
            "senderIp": "10.2.192.82"
        }
    ],
    "rxMulticastGroups": [
        {
            "bytesCount": 1398,
            "groupIp": "224.0.0.251",
            "lastUpdate": "2023-07-09T19:30:26.194130182Z",
            "numberOfDestinations": 0,
            "pktsCount": 12,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 256,
            "groupIp": "224.0.0.252",
            "lastUpdate": "2023-07-09T19:30:26.194131135Z",
            "numberOfDestinations": 0,
            "pktsCount": 2,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 0,
            "groupIp": "224.0.0.1",
            "lastUpdate": "2023-07-09T19:30:26.194131851Z",
            "numberOfDestinations": 0,
            "pktsCount": 0,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 484880298,
            "groupIp": "239.1.1.4",
            "lastUpdate": "2023-07-09T19:30:26.194133281Z",
            "numberOfDestinations": 0,
            "pktsCount": 345849,
            "protocolType": 0,
            "srcIp": "",
            "srcPort": 0
        },
        {
            "bytesCount": 306942664,
            "groupIp": "239.1.1.5",
            "lastUpdate": "2023-07-09T19:30:26.194133758Z",
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
                "numberOfDestinations": 0,
                "pktsCount": 218932,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            },
            {
                "bytesCount": 0,
                "groupIp": "239.1.1.1",
                "lastUpdate": "2023-07-09T19:30:26.194133758Z",
                "numberOfDestinations": 0,
                "pktsCount": 0,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            }
        ],
        "timestamp": 1688931026194084100,
        "txMulticastGroups": [],
        "virtualTunIpAddr": "",
        "xNicMode": "HA",
        "xNicVersion": 1
    },
    "aks-nodepool1-40504797-vmss00000C": {
        "activeConnections": 1,
        "byteCounters": {
            "rxMulticast": 110098251,
            "rxTotal": 110098251,
            "txMulticast": 0,
            "txTotal": 0
        },
        "cloudSwxtchVersion": "v2.0.7",
        "ipAddr": "10.2.128.95",
        "latencies": {
            "buckets": {
                "0.000050": 0,
                "0.000100": 0,
                "0.000250": 0,
                "0.000500": 0,
                "0.001000": 0,
                "0.002000": 0,
                "0.005000": 0,
                "0.010000": 0,
                "0.011000": 0
            },
            "count": 0,
            "sum": 0
        },
        "maxActiveConnections": 1,
        "packetCounters": {
            "rxDrops": 0,
            "rxMulticast": 633710,
            "rxTotal": 633710,
            "txDrops": 0,
            "txIgmp": 0,
            "txMulticast": 0,
            "txTotal": 0
        },
        "rxHaCounters": [],
        "rxMulticastGroups": [
            {
                "bytesCount": 476512,
                "groupIp": "10.2.131.255",
                "lastUpdate": "2023-07-09T19:30:26.4050092697Z",
                "numberOfDestinations": 0,
                "pktsCount": 3699,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            }
        ],
        "timestamp": 1688931026404890904,
        "txMulticastGroups": [],
        "virtualTunIpAddr": "",
        "xNicMode": "Normal",
        "xNicVersion": 2
    },
    "aks-nodepool1-40504797-vmss00000D": {
        "activeConnections": 1,
        "byteCounters": {
```


©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
            "rxMulticast": 110098251,
            "rxTotal": 110098251,
            "txMulticast": 0,
            "txTotal": 0
        },
        "cloudSwxtchVersion": "v2.0.7",
        "ipAddr": "10.2.128.96",
        "latencies": {
            "buckets": {
                "0.000050": 0,
                "0.000100": 0,
                "0.000250": 0,
                "0.000500": 0,
                "0.001000": 0,
                "0.002000": 0,
                "0.005000": 0,
                "0.010000": 0,
                "0.011000": 0
            },
            "count": 0,
            "sum": 0
        },
        "maxActiveConnections": 1,
        "packetCounters": {
            "rxDrops": 0,
            "rxMulticast": 633710,
            "rxTotal": 633710,
            "txDrops": 0,
            "txIgmp": 0,
            "txMulticast": 0,
            "txTotal": 0
        },
        "rxHaCounters": [],
        "rxMulticastGroups": [
            {
                "bytesCount": 228377,
                "groupIp": "10.2.131.255",
                "lastUpdate": "2023-07-09T19:30:26.5260522362",
                "numberOfDestinations": 0,
                "pktsCount": 1804,
                "protocolType": 0,
                "srcIp": "",
                "srcPort": 0
            }
        ],
        "timestamp": 1688931026525962306,
        "txMulticastGroups": [],
        "virtualTunIpAddr": "",
        "xNicMode": "Normal",
        "xNicVersion": 2
    }
}
```


WxckedEye API

This API is available for backwards compatibility but will be deprecated in 2.1 version.

GET **/api/wxckedeye/v1/dashboard**

Request: Empty

Response:

code	description
200	successful operation

Example:

Bash Copy

```
curl http://<cloudSwXtch-control-IP>/api/wxckedeye/v1/dashboard
```

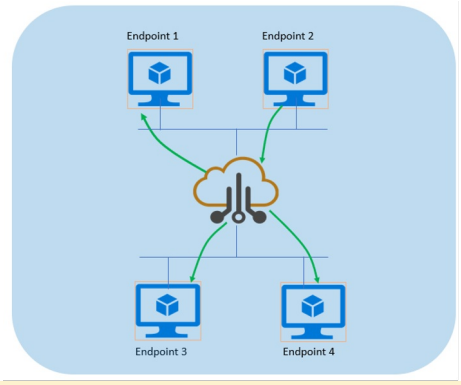
This cloudSwXtch API documentation will examine each section of the response and provide users a better understanding of each field.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

To track time as a running total of seconds, the **Timestamps** are in **Unix Timestamp**. This count starts at the Unix Epoch on January 1st, 1970, at UTC. At any point in time, the API can be run, and certain metrics can be obtained from the response payload by calculating certain counter and timestamp Delta values.

The example response comprises of one cloudSwXtch and the four agents connected to it. The response has been broken into several sections in the document. This will make each section easier to digest.

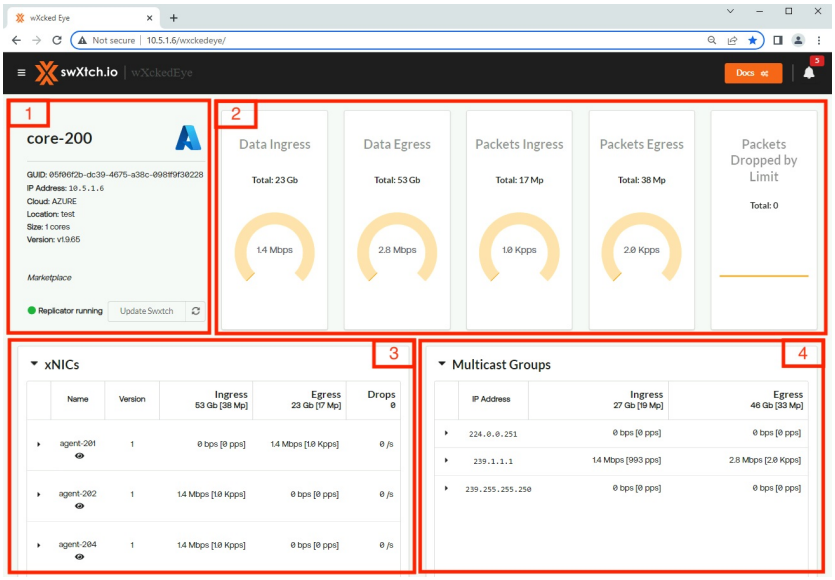
The example that will be used in this article refers to the network below, which has been simplified to help users understand the data returned in the API. As shown in Figure 1, Endpoint (Agent) 2 is sending data via multicast through the cloudSwXtch to Endpoints 1, 3 and 4.



A Note on Example Responses

Each example response will have notes on the right hand side in between asterisks (*). These notes explain what each value means to the reader. They will **not** appear in a typical response.

wXcked Eye User Interface



The figure above is a screenshot of the cloudSwXtch monitoring page in wXcked Eye, a web UI used to display data from the API. The following section will be broken into four subsections based on the numbering schema in the screenshot.

Section 1: Generic cloudSwXtch Information

Information about the cloudSwXtch instance such as cloudSwXtch Name GUID, cloud provider, managed resource group and resource group. As well as information about the subscription such as size, trial period, number of cores and active state can be found in the first and last sections of the response.

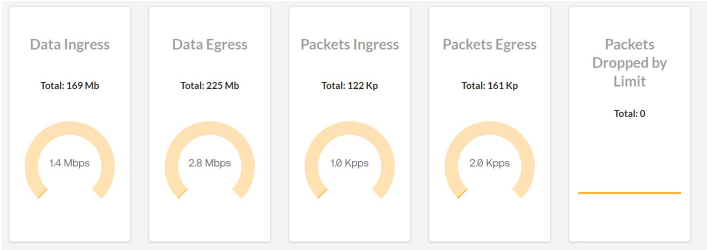
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
    "remfVersion": "v2.0.34",   *cloudSwXtch Software Version*
    "billingPlan": "trial",       *Plan Type*
    "cloud": "AZURE",             *Cloud Provider*
    "ipAddr": "10.2.128.27",
    "maxClientCount": "0",
    "maxPacketRateKpps": "0",
    "maxBandwidthMBPS": "0",
    "swxtchGuid": "07194da7a05d4ce3803d77eb77b0c29c",
    "swxtchName": "dsd-core-174",
    "managedResourceGroup": "mrg-sdmc-1_1-20220613202339",
    "resourceGroup": "test-resource",
    .
    .
    .
    "subscriptionId": null,
    "hostName": "ip-172-31-19-238",
    "numCores": 4,
    "replStatus": "running",
    "remainingDays": null,
    "authorized": true,
    "validationResult": null,
    "isMarketplace": true,
    "meshConfig": null,
    "expiresAt": null,
    "azureData": null,
    "awsData": {
        "accountId": "639720666639",
        "regionName": "us-east-1"
    },
    "alibabaData": null,
    "ociData": null,
    "gcpData": null
}
```

Section 2: cloudSwXtch Bytes and Packet Data

The cloudSwXtch wXcked Eye user interface displays egress and ingress data as shown in Figure 4. In addition, ingress and egress packets are also included.



The first part is high level as the cloudSwXtch. **xnics** represents agents and **xnicTotals** as well as **replTotals** represents the cloudSwXtch.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

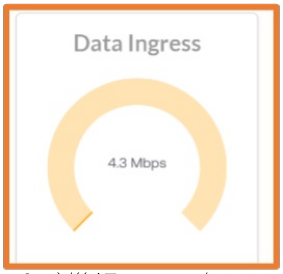
```
"xnicTotals": {                          *Statistics totals of agents*
       "PktCounters": {                   *Packet counters*
           "Nic2McaTotal": 30218,              *Packets from swxtch to agent*
           "Nic2McaMc": 30218,               *Multicast packets from swxtch to agent*
           "Mca2NicTotal": 31526,              *Packets from agent to swxtch*
           "Mca2NicMc": 31524,           *Multicast packets from agent to swxtch*
           "Mca2NicIgmp": 6,            *IGMP packets from agent to swxtch*
           "Mca2NicDrops": 0,           *Packets lost from agent to swxtch*
           "Nic2McaDrops": 0,            *Packets lost from swxtch to agent*
           "Mca2KniDrops": 0,            *Packets lost from agent to kernel NIC*
           "Kni2McaDrops": 0,            *Packets lost from kernel NIC to agent*
           "McaPktDrops": 0,            *Packets lost at agent*
           "McaBigPktDrops": 0           *Big size packets lost at agents*
       },
       "ByteCounters": {
           "Nic2McaTotal": 32347812,     *Bytes From swxtch to agent*
           "Nic2McaMc": 32347812,      *Multicast bytes from swxtch to agent*
           "Mca2NicTotal": 33795772,     *Bytes from agent to swxtch*
           "Mca2NicMc": 33795584      *Multicast bytes from agent to swxtch*
       },
       "Latencies": {
           "Count": 0,
           "Sum": 0,
           "Buckets": null
       },
       "HARxCounters": null,
       "Timestamp": 1697562240190902784,
       "IpAddress": "",
       "SoftwareVersion": "",
       "XnicVersion": 0,
       "RxMulticastGroups": null,
       "TxMulticastGroups": null,
       "XnicMode": "",
       "NumActiveConnections": 0,
       "MaxNumActiveConnections": 0,
       "VirtualTunIpAddress": ""

   },
```

cloudSwXtch network ingress and egress data are useful when shown in a custom user interface. To display the data like in the example of the user interface above, API calls should be made periodically. These points in time can then be used to compute data based on time.

$\Delta Something$ means the difference between the value of at time and $Something$ at time $t1$ and $Something$ $t2$. Example: If Egress has value 39000 at time $t1$ and value 19800 at time $t2$,

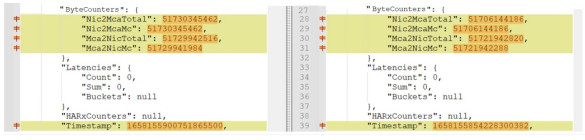
$\Delta Egress = (39000 - 19800) = result$

Calculating ByteCounters - Data Ingress



$DataIngress = [xnicTotals][ByteCounters](\Delta Nic2McaTotal * 8)/((\Delta Timestamp/1,000,000,000))$

Below is a cut from the response of the API called at two different times.



Taking the above expression and putting in the data from this call is shown below.

Timestamp from the call 2, 1658155900751865500 and the timestamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 in nanoseconds. Then, dividing the Δ by 1,000,000,000, we get the Δ of 46.523565312 in seconds, which we will use to calculate the data rate in bits per second.

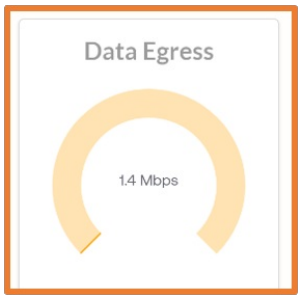
$\Delta Nic2McaTotal$ is $51730345462 - 51706144186 = 7999676 bits$

$[xnicTotals][ByteCounters](\Delta Mca2NicTotal * 8)/((\Delta Timestamp/1,000,000,000)) = (7999676 * 8)/46.523565312 = 4161551.392 \approx 4.2 Mbps$

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

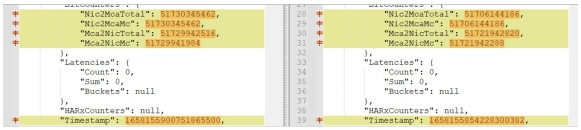
Please note: Since we calculated the average data ingress over a timespan of ~46 seconds, the value does not exactly match the UI screenshot. This is because the UI web app calls the API every 5 seconds by default, giving us more instantaneous rate compared to our calculation.

Calculating ByteCounters - Data Egress



$$DataEgress = [xnicTotals][ByteCounters](\Delta Mca2NicTotal * 8)/((\Delta Timestamp/1,000,000,000))$$

Below is a cut from the response of the API called at two different times.



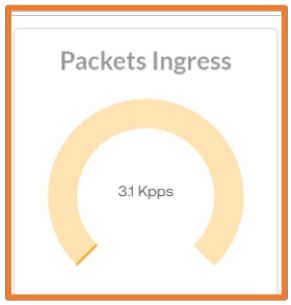
Taking the above expression and putting in data from this call is shown below.

Timestamp from call 2, 1658155900751865500 and time stamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 nanoseconds. Then dividing the Δ by 1,000,000,000, we get the Δ of 46.523565312 in seconds.

$\Delta Mca2NicTotal$ is $51729942516–51721942820 = 63997568 bits$

$[xnicTotals][ByteCounters](\Delta Mca2NicTotal * 8)/((\Delta Timestamp/1,000,000,000)) = (63997568 * 8)/46.523565312 = 1375594.66 \approx 1.4 Mbps$

Calculating PktCounters - Packets Ingress



$$PacketsIngress = [xnicTotals][PktCounters](\Delta Nic2McaTotal)/((\Delta Timestamp/1,000,000,000))$$

Below is a cut from the response of the API called at two different times.

Taking the above expression and putting in data from this call is shown below.

Timestamp from call 2, 1658155900751865500 and the timestamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 nanoseconds. Then, dividing the Δ by 1,000,000,000, we get a Δ of 46.523565312 in seconds.

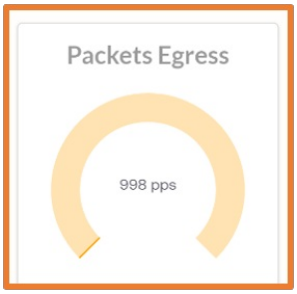
$\Delta Nic2McaTotal$ is $297300912 - 297161822 = 139090 Packets$

$[xnicTotals][PktCounters](\Delta Nic2McaTotal)/((\Delta Timestamp/1,000,000,000)) = 139090/46.523565312 = 2989.667689 \approx 3.0 Kpps$

Please note: Since we calculated the average data ingress over a timespan of ~46 seconds, the value does not exactly match the UI screenshot. This is because the UI web app calls the API every 5 seconds by default, giving us more instantaneous rate compared to our calculation.

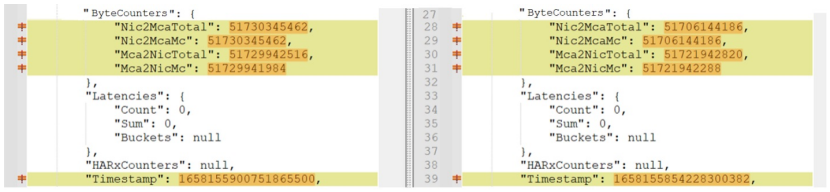
Calculating PktCounters - Packets Egress

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



$PacketsEgress = [xnicTotals][PktCounters](\Delta Mca2NicTotal)/((\Delta Timestamp/1,000,000,000))$

Below is a cut from the return at two different times with data this section focuses on in orange.



```
"ByteCounters": {
    "Nic2McaTotal": 51730345462,
    "Nic2McaMc": 51730345462,
    "Mca2NicTotal": 51729942516,
    "Mca2NicMc": 51729941984
},
"Latencies": {
    "Count": 0,
    "Sum": 0,
    "Buckets": null
},
"HARxCounters": null,
"Timestamp": 1658155900751865500,
```

```
"ByteCounters": {
    "Nic2McaTotal": 51706144186,
    "Nic2McaMc": 51706144186,
    "Mca2NicTotal": 51721942820,
    "Mca2NicMc": 51721942288
},
"Latencies": {
    "Count": 0,
    "Sum": 0,
    "Buckets": null
},
"HARxCounters": null,
"Timestamp": 1658155854228300382,
```

Taking the above expression and putting in data from this call is shown below.

Timestamp from call 2, 1658155900751865500 and the timestamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 nanoseconds. Then, dividing the Δ by 1,000,000,000, we get a Δ of 46.523565312 in seconds.

$\Delta Mca2NicTotal$ is $297298589–297252613 = 988.2303665 Packets$

$[xnicTotals][PktCounters](\Delta Mca2NicTotal)/((\Delta Timestamp/1,000,000,000)) = (988.2303665)/46.523565312 = 988.2303665 \approx 988pps$

Please note: Since we calculated the average data ingress over a timespan of ~46 seconds, the value does not exactly match the UI screenshot. This is because the UI web app calls the API every 5 seconds by default, giving us a more instantaneous rate compared to our calculation.

Section 3: Agents Bytes and Packet Data

This section will first show returns for four agents. Following the data will be a breakdown of how to calculate data like the section before:

- Ingress
- Egress
- Drops

Calculations for the data above is shown in each subsection. For this document, the calculations will only be for Ingress of DSd-agent-101 for and Egress of DSd-agent-102.

Agent #1

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
  "xnics": {                      *Stats of individual agents*
     "Agent-1": {
         "PktCounters": {         *Packet Counters*
             "Nic2McaTotal": 75,     *Packets from swxtch to agent*
             "Nic2McaMc": 75,      *Multicast packets from swxtch to agent*
             "Mca2NicTotal": 31516,    *Packets from agent to Swxtch*
             "Mca2NicMc": 31516,     *Multicast packets from agent to swxtch*
             "Mca2NicIgmp": 2,      *IGMP packets from agent to swxtch*
             "Mca2NicDrops": 0,      *Packets lost from agent to swxtch*
             "Nic2McaDrops": 0,         *Packets lost from swxtch to agent*
             "Mca2KniDrops": 0,      *Packets lost from agent to kernel NIC*
             "Kni2McaDrops": 0,       *Packets lost from kernel NIC to agent*
             "McaPktDrops": 0,        *Packets lost at agent*
             "McaBigPktDrops": 0       *Big size packets lost at agent*
         },
         "ByteCounters": {
             "Nic2McaTotal": 8754,     *Bytes From Swxtch To Agent*
             "Nic2McaMc": 8754,      *Multicast Bytes From Swxtch To Agent*
             "Mca2NicTotal": 33794704,  *Bytes From Agent To Swxtch*
             "Mca2NicMc": 33794704      *Multicast bytes from agent to swxtch*
         },
         "Latencies": {
             "Count": 0,
             "Sum": 0,
             "Buckets": {
                 "0.000050": 0,
                 "0.000100": 0,
                 "0.000250": 0,
                 "0.000500": 0,
                 "0.001000": 0,
                 "0.002000": 0,
                 "0.005000": 0,
                 "0.010000": 0,
                 "0.011000": 0
             }
         },
         "HARxCounters": [],              *High Availability Rx Stats*
         "Timestamp": 1657216501229157803,
         "SoftwareVersion": "v2.0.34",     *Swxtch Version*
         "XnicVersion": 2               *Agent Version*
        },
```

Latency Buckets

The API breaks out latencies into buckets. Bucket 0 contains the count of latencies in 0.000050 (50 microseconds). Bucket 1 contains latencies in 0.00100 (1 millisecond) and so on. Currently, these fields **will not** populate and are reserved for future use. Every bucket will remain at 0.

Agent #2

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
"Agent-2": {
  "PktCounters": {           *Packet Counters*
       "Nic2McaTotal": 30107,   *Packets from swxtch to agent*
       "Nic2McaMc": 30107,       *Multicast packets from swxtch to agent*
       "Mca2NicTotal": 0,       *Packets from agent to Swxtch*
       "Mca2NicMc": 0,        *Multicast packets from agent to swxtch*
       "Mca2NicIgmp": 2,        *IGMP packets from agent to swxtch*
       "Mca2NicDrops": 0,        *Packets lost from agent to swxtch*
       "Nic2McaDrops": 0,         *Packets lost from swxtch to agent*
       "Mca2KniDrops": 0,        *Packets lost from agent to kernel NIC*
       "Kni2McaDrops": 0,         *Packets lost from kernel NIC to agent*
       "McaPktDrops": 0,          *Packets lost at agent*
       "McaBigPktDrops": 0     *Big size packets lost at agent*
   },
   "ByteCounters": {
       "Nic2McaTotal": 32334918,  *Bytes From Swxtch To Agent*
       "Nic2McaMc": 32334918,    *Multicast Bytes From Swxtch To Agent*
       "Mca2NicTotal": 0,       *Bytes From Agent To Swxtch*
       "Mca2NicMc": 0         *Multicast bytes from agent to swxtch*
  },
  "Latencies": {
       "Count": 0,
       "Sum": 0,
       "Buckets": {
           "0.000050": 0,
           "0.000100": 0,
           "0.000250": 0,
           "0.000500": 0,
           "0.001000": 0,
           "0.002000": 0,
           "0.005000": 0,
           "0.010000": 0,
           "0.011000": 0
       }
   },
   "HARxCounters": [],           *High Availability Rx Stats*
   "Timestamp": 1657216500709825713,
   "SoftwareVersion": "v2.0.34",    *Swxtch Version*
   "XnicVersion": 2               *Agent Version*
 },
```

Agent #3:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
"Agent-3": {
  "PktCounters": {       *Packet Counters*
       "Nic2McaTotal": 18,   *Packets from swxtch to agent*
       "Nic2McaMc": 18,    *Multicast packets from swxtch to agent*
       "Mca2NicTotal": 9,   *Packets from agent to Swxtch*
       "Mca2NicMc": 8,    *Multicast packets from agent to swxtch*
       "Mca2NicIgmp": 1,    *IGMP packets from agent to swxtch*
       "Mca2NicDrops": 0,    *Packets lost from agent to swxtch*
       "Nic2McaDrops": 0,     *Packets lost from swxtch to agent*
       "Mca2KniDrops": 0,    *Packets lost from agent to kernel NIC*
       "Kni2McaDrops": 0,     *Packets lost from kernel NIC to agent*
       "McaPktDrops": 0,      *Packets lost at agent*
       "McaBigPktDrops": 0    *Big size packets lost at agent*
   },
   "ByteCounters": {
       "Nic2McaTotal": 2070,  *Bytes From Swxtch To Agent*
       "Nic2McaMc": 2070,    *Multicast Bytes From Swxtch To Agent*
       "Mca2NicTotal": 974,  *Bytes From Agent To Swxtch*
       "Mca2NicMc": 880     *Multicast bytes from agent to swxtch*
  },
  "Latencies": {
       "Count": 18,
       "Sum": 302211676294169800,
       "Buckets": {
           "0.000050": 0,
           "0.000100": 0,
           "0.000250": 0,
           "0.000500": 0,
           "0.001000": 0,
           "0.002000": 0,
           "0.005000": 0,
           "0.010000": 0,
           "0.011000": 0
       }
   },
   "HARxCounters": [],           *High Availability Rx Stats*
   "Timestamp": 1657216500559694400,
   "SoftwareVersion": "v2.0.34",   *Swxtch Version*
   "XnicVersion": 2                *Agent Version*
 },
```

Agent #4:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
"Agent-4": {
  "PktCounters": {       *Packet Counters*
       "Nic2McaTotal": 18,   *Packets from swxtch to agent*
       "Nic2McaMc": 18,    *Multicast packets from swxtch to agent*
       "Mca2NicTotal": 1,   *Packets from agent to Swxtch*
       "Mca2NicMc": 0,    *Multicast packets from agent to swxtch*
       "Mca2NicIgmp": 1,    *IGMP packets from agent to swxtch*
       "Mca2NicDrops": 0,    *Packets lost from agent to swxtch*
       "Nic2McaDrops": 0,     *Packets lost from swxtch to agent*
       "Mca2KniDrops": 0,    *Packets lost from agent to kernel NIC*
       "Kni2McaDrops": 0,     *Packets lost from kernel NIC to agent*
       "McaPktDrops": 0,      *Packets lost at agent*
       "McaBigPktDrops": 0    *Big size packets lost at agent*
   },
   "ByteCounters": {
       "Nic2McaTotal": 2070,  *Bytes From Swxtch To Agent*
       "Nic2McaMc": 2070,    *Multicast Bytes From Swxtch To Agent*
       "Mca2NicTotal": 94,  *Bytes From Agent To Swxtch*
       "Mca2NicMc": 0     *Multicast bytes from agent to swxtch*
  },
  "Latencies": {
       "Count": 18,
       "Sum": 302211676294180540,
       "Buckets": {
           "0.000050": 0,
           "0.000100": 0,
           "0.000250": 0,
           "0.000500": 0,
           "0.001000": 0,
           "0.002000": 0,
           "0.005000": 0,
           "0.010000": 0,
           "0.011000": 0
       }
   },
   "HARxCounters": [],            *High Availability Rx Stats*
   "Timestamp": 1657216500954028100,
   "SoftwareVersion": "v2.0.34",    *Swxtch Version*
   "XnicVersion": 2                 *Agent Version*
 },
```

xNICs Ingress Data

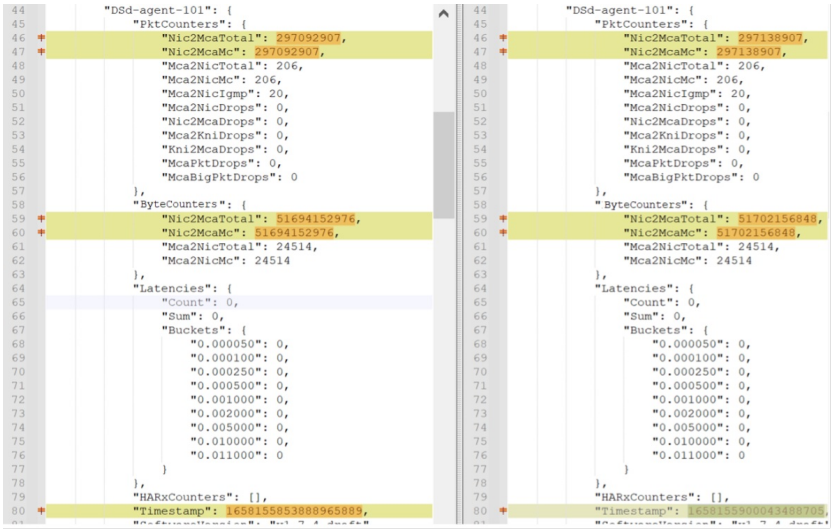
Name	Ingress 77 Mb [55 Kp]
DSd-agent-101	1.4 Mbps [999 pps]
DSd-agent-102	0 bps [0 pps]
DSd-agent-104	1.4 Mbps [1.0 Kpps]
DSd-agent-105	1.4 Mbps [1.0 Kpps]

1

- **Total Data**: $[xnicTotals][ByteCounters]Nic2McaTotal * 8$
- **Total Packets**: $[xnicTotals][PktCounters]Nic2McaTotal$
- **Data Ingress**: $([xnics][< agentName >][ByteCounters](\Delta Nic2McaTotal * 8))/((\Delta Timestamp/1,000,000,000))$
- **Pkts Ingress**: $([xnics][< agentName >][PktCounters](\Delta Nic2McaTotal))/((\Delta Timestamp/1,000,000,000))$

Below is a cut from the response of the API called at two different times.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



Taking the above expressions and putting in data from this call is shown below.

Timestamp from call 2, 1658155900043488705 and the timestamp from call 1, 1658155853888965889 gives us a Δ of 46154522880 nanoseconds. Then, dividing the Δ by 1,000,000,000, we get Δ of 46.15452288 in seconds.

Data Ingress: $\Delta Nic2McaTotal * 8 = 64030976 bits$

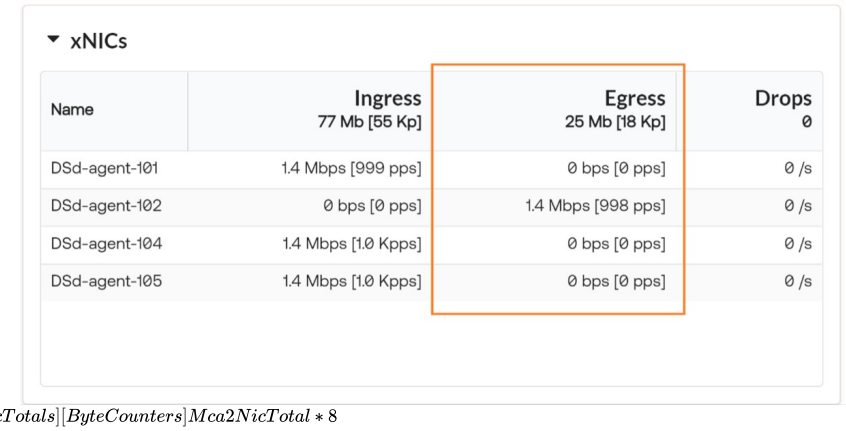
$[xnics][DSd - agent - 101][ByteCounters](\Delta Nic2McaTotal * 8)/((\Delta Timestamp/1,000,000,000)) = (64030976)/46.15452288 = 1387317.473 \approx 1.4 Mbps$

Pkts Ingress: $\Delta Nic2McaTotal$ is $297138907 – 297092907 = 46000 Packets$

$([xnics][DSd - agent - 101][PktCounters](\Delta Nic2McaTotal))/((\Delta Timestamp/1,000,000,000)) = 46000/46.15452288 = 996.65 \approx 997 pps$

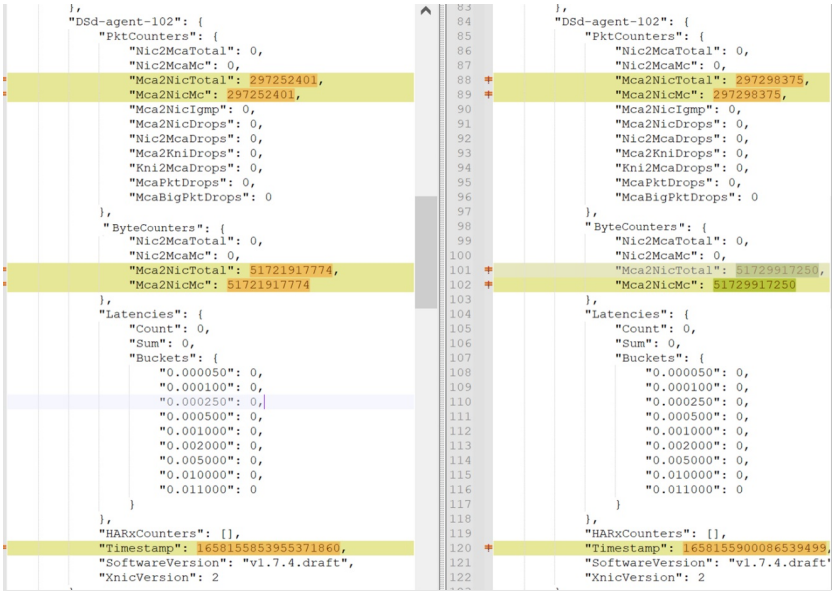
Please note: Since we calculated the average data ingress over a timespan of ~46 seconds, the value does not exactly match the UI screenshot. This is because the UI web app calls the API every 5 seconds by default, giving us a more instantaneous rate compared to our calculation.

xNICs Egress Data - Section 2:



- **Total Data:** $[xnicTotals][ByteCounters]Mca2NicTotal * 8$
- **Total Packets:** $[xnicTotals][PktCounters]Mca2NicTotal$
- **Data Egress:** $([xnics][< agentName >][ByteCounters](\Delta Mca2NicTotal * 8))/((\Delta Timestamp/1,000,000,000))$
- **Pkts Egress:** $([xnics][< agentName >][PktCounters](\Delta Mca2NicTotal))/((\Delta Timestamp/1,000,000,000))$

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



Taking the above expressions and putting in data from this call is shown below.

Timestamp from call 2, 1658155900086539499 and timestamp from call 1, 1658155853955371860 gives us a Δ of 46131167744 nanoseconds. Then, dividing the Δ by 1,000,000,000, we get Δ of 46.131167744 in seconds.

Data Egress: $\Delta Mca2NicTotal * 8$ is $(51729917250 - 51721917774) * 8 = 63995808 bits$

$[xnics][DSd - agent - 102][ByteCounters](\Delta Mca2NicTotal * 8)/((\Delta Timestamp/1,000,000,000))$
$= (63995808)/(46.131167744) = 1387257.49054 \approx 1.4 Mbps$

Pkts Egress: $\Delta Mca2NicTotal$ is $297298375 – 297252401 = 45974 Packets$

$([xnics][DSd - agent - 102][PktCounters](\Delta Mca2NicTotal))/((\Delta Timestamp/1,000,000,000)) = 45974/(46.131167744) = 996.593 \approx 997 pps$

Please note: Since we calculated the average data ingress over a timespan of ~46 seconds, the value does not exactly match the UI screenshot. This is because the UI web app calls the API every 5 seconds by default, giving us a more instantaneous rate compared to our calculation.

xNIC Drops - Section 3

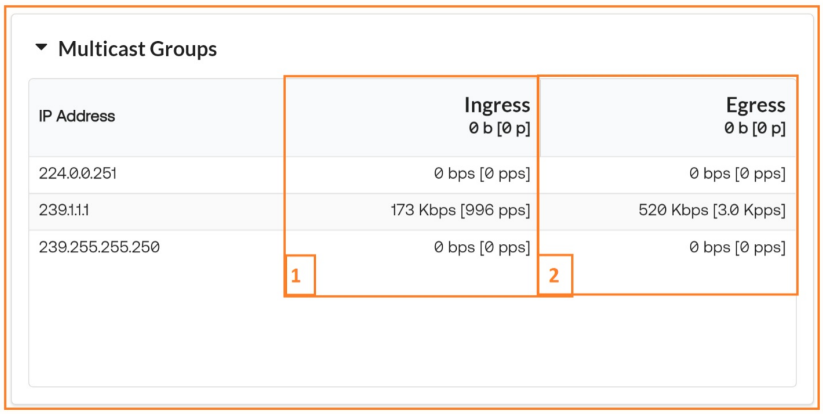
Total: $xnicTotals][PktCounters]Mca2NicDrops$

Drop(s): $([xnics][< agentName >][PktCounters](\Delta Mca2NicDrops))/((\Delta Timestamp/1,000,000,000))$

Since there were no drops at the time of the calls in this document, there are no examples for the calculations.

Section 4: Multicast Data

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



This section will discuss data available for multicast groups for a cloudSwXtch and will be broken down into two sections: Ingress and Egress. The multicast group 239.1.1.1 will be used for calculations below. Below is an example of the data stats, Multicast Group data and other statistics:

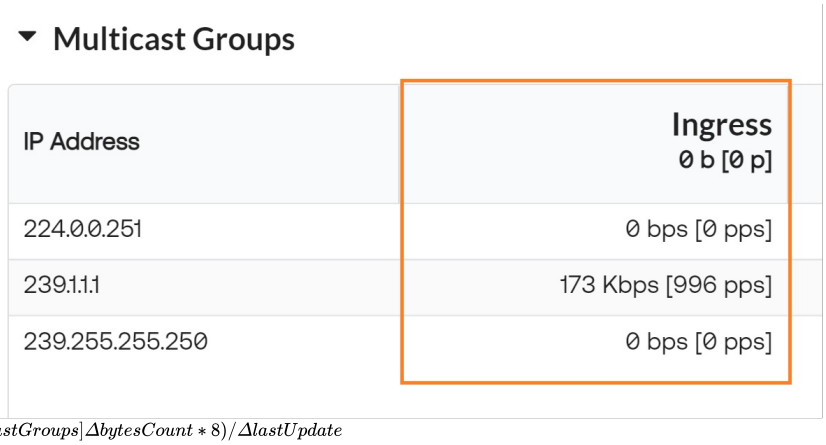
Bash Copy

```
},
   "replTotals": {                      *Data stats*
       "host": "",
       "sequence": 95924,
       "rxCount": 31535,           *Received packets*
       "txCount": 30332,           *Sent packets*
       "rxBytes": 33796794,         *Received bytes*
       "txBytes": 32361180,         *Sent bytes*
       "rxBridgeBytes": 0,          *Received bridge bytes*
       "rxBridgeCount": 0,          *Received bridge packets*
       "timestamp": 1657216500274611599,
       "dropsByByteLimit": 0,    *Dropped bytes*
       "dropsByCountLimit": 0,    *Dropped packets*
       "rxMeshPktCount": 0,      *Received mesh packets*
       "rxMeshBytes": 0,         *Received mesh bytes*
       "txMeshPktCount": 0,      *Sent mesh packets*
       "txMeshBytes": 0,         *Sent mesh bytes*
       "rxUnicastPktCount": 0,
       "rxUnicastBytes": 0,
       "txUnicastPktCount": 0,
       "txUnicastBytes": 0,
       "rxStreamLinks": [      *Receiving multicast groups*
           {
               "groupIp": "239.1.1.3",
               "groupPort": 3490
               "nodeIp": "172.30.0.6",
               "nodePort": 47040,
               "protocol": 3,
               "pktsCount": 31460,
               "bytesCount": 33788040,
               "lastUpdate": "2022-07-07T15:29:00.789006233"
           },
           {
               "groupIp": "239.0.0.251",
               "groupPort": 3490,
               "nodeIP": "172.30.0.6",
               "nodePort": 47040,
               "protocol": 3,
               "pktsCount": 75,
               "bytesCount": 8754,
               "lastUpdate": "2022-07-07T17:54:49.132519721Z"
           }
          ],
           "txStreamLinks": [
            {
               "groupIp": "239.1.1.1",
               "groupPort": 3490,
               "nodeIp": "10.2.192.82",
               "nodePort": 0,
               "protocol": 3,
               "pktsCount": 784125907,
               "bytesCount": 136437907818,
               "lastUpdate": "2023-10-18T17:06:55.270435571Z"
           }
          ]
         },
   "subscriptionId”: "d723b93f-112c-49fb-9fda-03d3ada867f3",
   "hostName": "dsd-core-174",
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

```
    "numCores": 8,
    "replStatus": "running",       *Replicator services status*
    "authorized": true,
    "validationResult": {
        "switch": 556,
        "authorized": true,
        "denialReason": null,
        "trialPeriod": {
            "startDate": "2021-12-22T19:38:33.565336Z",
            "endDate": "2022-08-21T19:38:33.565Z"
        },
        "applicationId": null
      "isMarketplace": true,
         "meshConfig": null,
         "expiresAt": null,
         "azureData": {
            "subscriptionId": "b10209ad-ad22-4c26-8aef-be93b2f0bb58",
            "managedResourceGroupName": "TEST-DONNA",
            "resourceGorupName": "TEST-DONNA",
            "sku": null,
            "billingFields": null,
            "plan": "swxtch-small-002"
        },
        "awsData": null,
        "alibabaData": null,
        "ociData": null,
        "gcpData": null
}
```

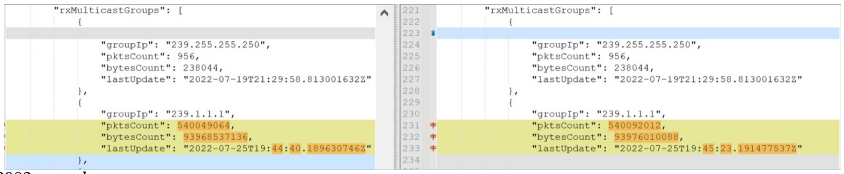

Multicast Ingress Data - Section 1



Data: $([xnics][RXMulticastGroups]\Delta bytesCount * 8)/\Delta lastUpdate$

Packets: $[xnics][RXMulticastGroups]\Delta pktsCount/\Delta lastUpdate$

Below is a cut from the return at two separate times with data for this section highlighted in orange.



$\Delta lastUpdate = 43.00003982 seconds$

Data Ingress:

$\Delta bytesCount * 8 = (93976010088–93968537136) * 8 =$ **59783616 bits**

$([xnics][RXMulticastGroups]\Delta bytesCount * 8)/\Delta lastUpdate = 59783616/43.00003982 = 1390315.364 \approx 1.4 Mbps.$

Note that the value in the figure above is wrongfully shown as Bytes/second as opposed to bits/s.

Packets Ingress:

$\Delta pktsCount = 540092012 - 540049064 = 42948 packets$

$[xnics][RXMulticastGroups]\Delta pktsCount/\Delta lastUpdate = 42948/43.00003982 = 998.789772749 \approx 999 pps$

Multicast Egress Data - Section 2

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

▼ Multicast Groups

IP Address	Ingress 0 b [0 p]	Egress 0 b [0 p]
224.0.0.251	0 bps [0 pps]	0 bps [0 pps]
239.1.1.1	173 Kbps [996 pps]	520 Kbps [3.0 Kpps]
239.255.255.250	0 bps [0 pps]	0 bps [0 pps]

Data: $([xnics][TXMulticastGroups]\Delta bytesCount * 8)/\Delta lastUpdate$

Packets: $[xnics][TXMulticastGroups]\Delta pktsCount/\Delta lastUpdate$

```
        ],
        "txMulticastGroups": [
            {
                "groupIp": "239.1.1.1",
                "pktsCount": 597605144,
                "bytesCount": 103983295056,
                "lastUpdate": "2022-07-25T19:44:40.1623785492"
            },
            {
                "groupIp": "224.0.0.251",
                "pktsCount": 969,
                "bytesCount": 115212,
                "lastUpdate": "2022-07-25T19:29:10.061601877Z"
            }
        ]
    },
```

```
        ],
        "txMulticastGroups": [
            {
                "groupIp": "239.1.1.1",
                "pktsCount": 597733970,
                "bytesCount": 104005710780,
                "lastUpdate": "2022-07-25T19:45:23.1624183652"
            },
            {
                "groupIp": "224.0.0.251",
                "pktsCount": 975,
                "bytesCount": 115872,
                "lastUpdate": "2022-07-25T19:44:41.1623795032"
            }
        ]
    },
```

Below is a cut from the return at two separate times with data for this section highlighted in orange.

$\Delta lastUpdate = 43seconds$

Data Egress: $\Delta bytesCount * 8 = (104005710780–103983295056) * 8 = 179325792bits$

$([xnics][TXMulticastGroups]\Delta bytesCount * 8)/\Delta lastUpdate = (179325792)/43 = 4170367.256 \approx 4.2Mbps.$

Note: The value in the figure above is wrongfully shown as Bytes/second as opposed to bits/s.

Packets Ingress: $\Delta pktsCount = 597733970 - 597605144 = 128826packets$

$[xnics][TXMulticastGroups]\Delta pktsCount/\Delta lastUpdate = 128826/43 = 2995.954 \approx 3.0Kpps$

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Configuration API

Overview

The cloudSwXtch Configuration API is intended for use to integrate the cloudSwXtch data with third party tools for configuring Mesh, High Availability, and Protocol Fanout.

Prerequisites

A cloudSwXtch must exist as well as two or more agents with xNICs. To have data, agents must be producing and consuming data via the cloudSwXtch. By using a GET command, data will be provided in the response.

GET **/services/settings**

Request: Empty

Response:

code	description
200	successful operation

Example:

Bash | Copy

```
curl http://<core>/services/settings
```

Note: Replace <core> with the control IP address of the cloudSwXtch.

The wXcked Eye settings API will give information based on the 4 tabs of the “Settings” page in the wXcked Eye UI: General, Mesh, High Availability, and Protocol Fanout. While this is one call to get all this information, the sections of the call will be broken out by the appropriate tab.

Below is an example response of the Settings call:

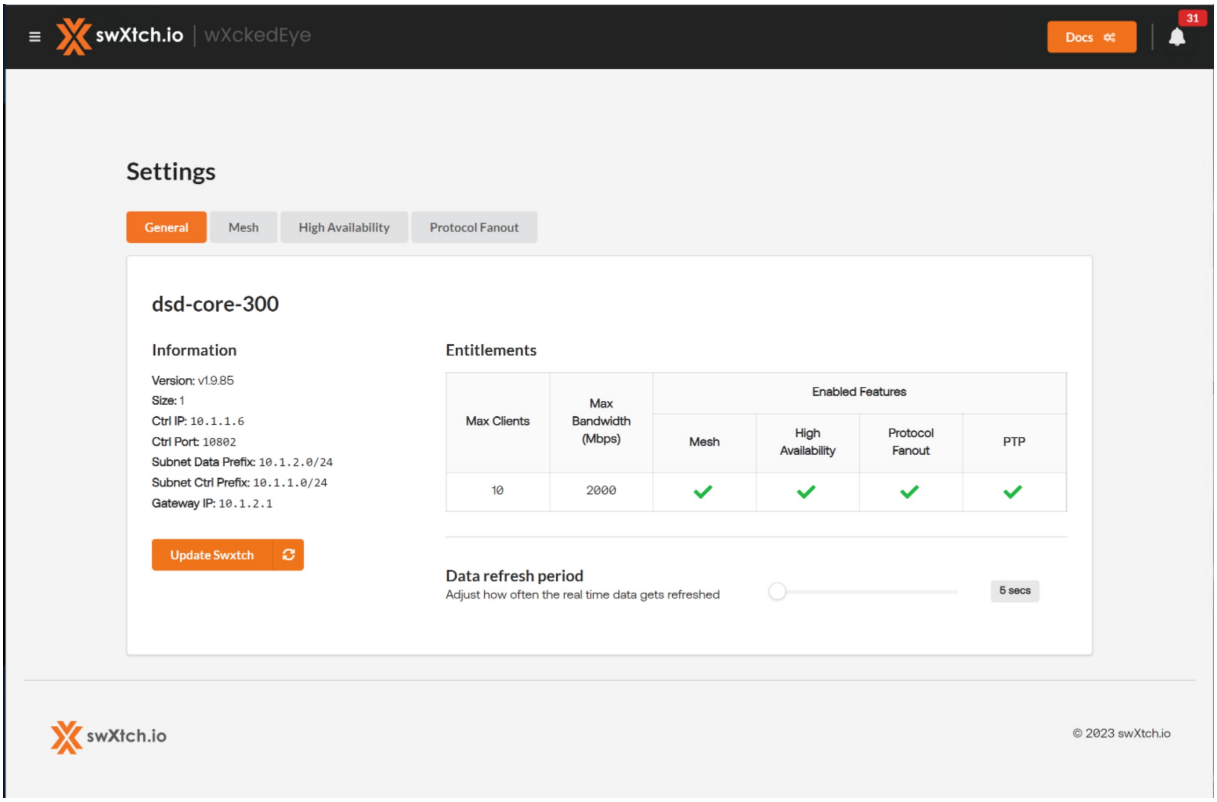
©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Mesh and HA Compatibility

Please note: In the above example, it displays information for both HA and Mesh settings. This is only for the purpose of this article. You **cannot** create a Mesh and HA configuration at the same time.

General

The general tab shows information about the cloudSwXtch networking and entitlements.



Below is a portion of the Settings call result detailing information in the General tab:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

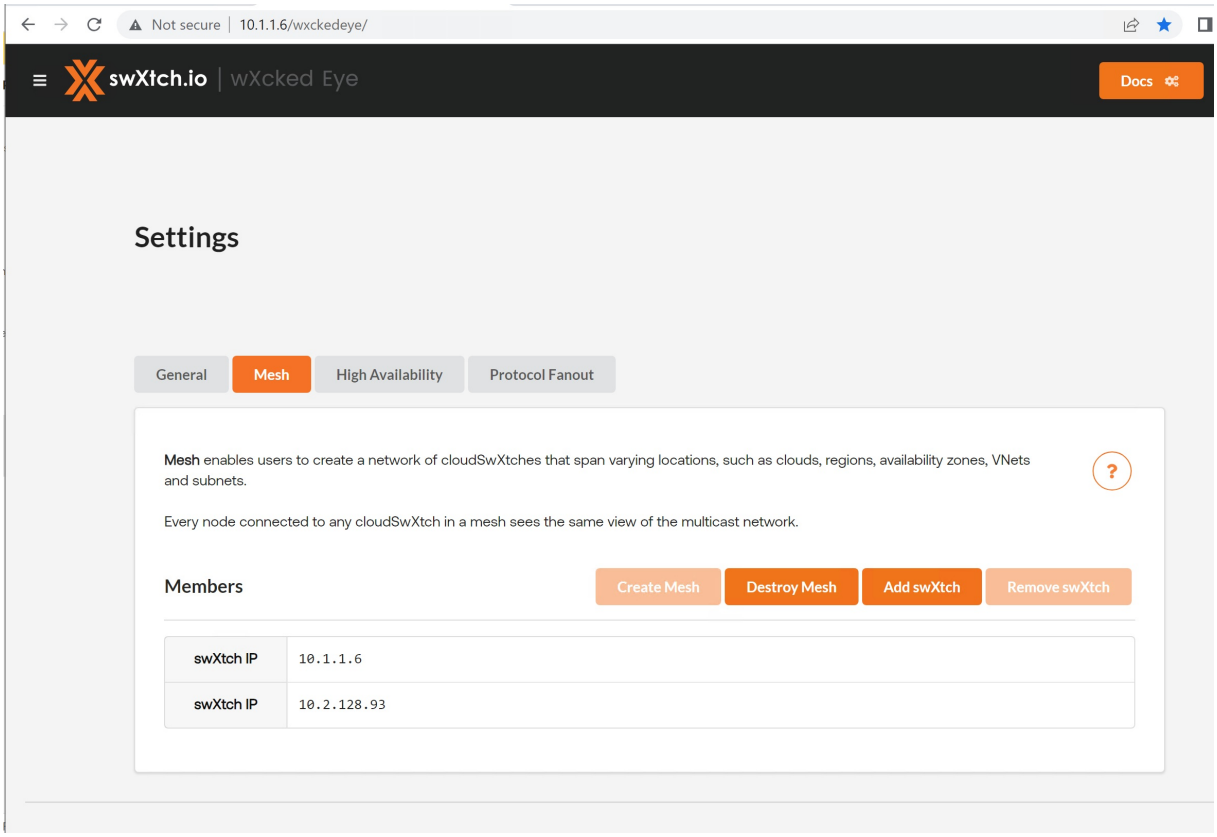
Bash Copy


```
{
    "generalSettings": {
        "hostName": "core-300",
        "switchName": "core-300",
        "version": "v1.9.85",
        "numCores": 1,
        "entitlements": {
            "maxClientCount": 10,
            "bandwidthMbps": 2000,
            "enableMesh": true,
            "enableUnicast": true,
            "enableHA": true,
            "enableClockSync": true
        },
        "subnetDataPrefix": "10.1.2.0/24",
        "subnetCtrlPrefix": "10.1.1.0/24",
        "dataGatewayIp": "10.1.2.1",
        "ctrlIp": "10.1.1.6",
        "ctrlPort": 10802,
        "gatewayMacAddr": "12:34:56:78:9a:bc",
        "replInfo": {
            "CtrlIp": "1.0.0.127",
            "CtrlPort": 9996,
            "DataIp": "10.1.2.6",
            "DataPort": 9999,
            "DataMac": "YEW9pzYf"
        }
    },
```


Mesh

The mesh tab shows the cloudSwXtches that are configured to be in a Mesh. Note that both High Availability and Mesh are configured here for example purposes. Mesh and High Availability are, however, not compatible with the same cloudSwXtches.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



Below is a portion of the Settings call response detailing information on the Mesh tab:

Bash Copy

```
"meshSettings": {
        "meshId": "86B62026-9222-0E62-03C2-6C5872CA64C5",
        "swxtches": [
            "10.1.1.6",
            "10.2.128.93"
        ],
        "uid": null
    },
```

Create Mesh

swxtch/mesh/v1/create

Show Mesh

swxtch/mesh/v1/show

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Add cloudSwXtch to Mesh

swxtch/mesh/v1/addSwxtch

Remove cloudSwXtch from Mesh

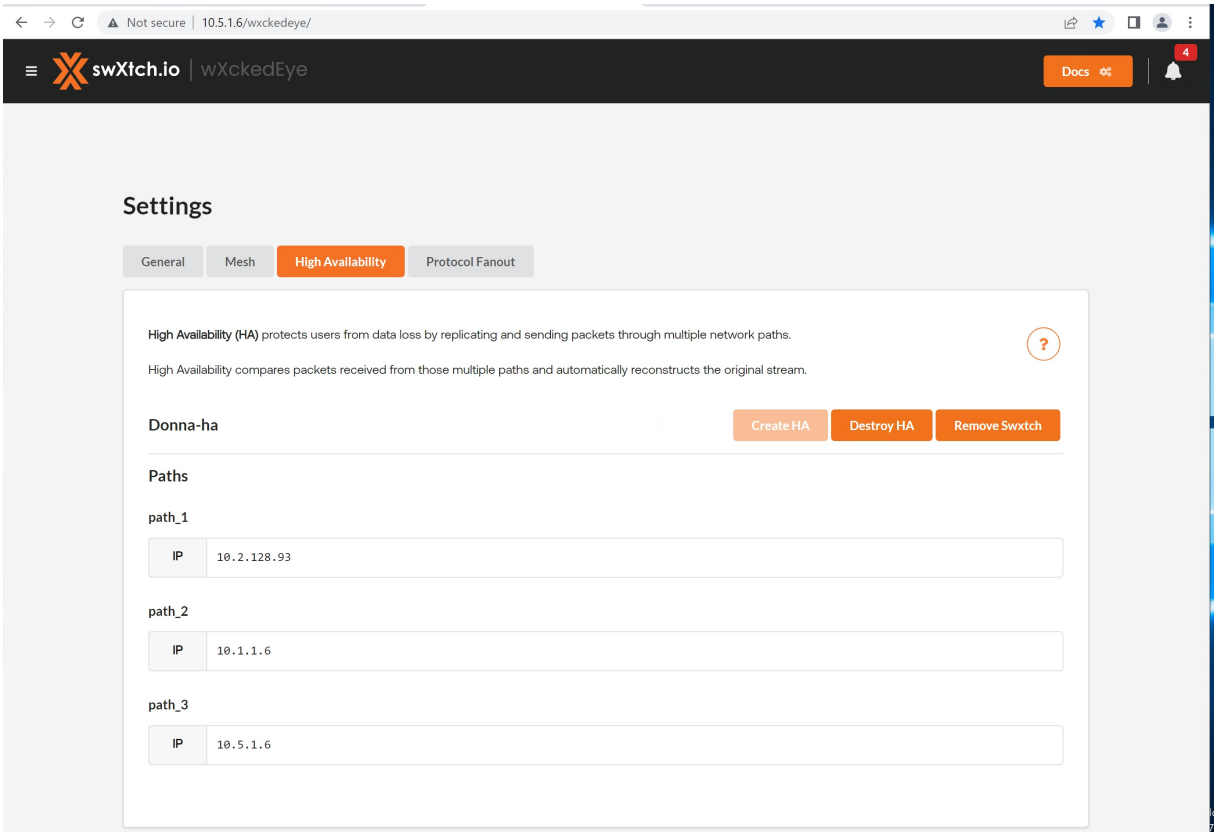
swxtch/mesh/v1/removeSwxtch

Destroy Mesh

swxtch/mesh/v1/destroy

High Availability

The High Availability tab shows the cloudSwXtches that are configured to be in a HA 2022-7 configuration. Note that both High Availability and Mesh are configured here for example purposes. Mesh and High Availability are, however, not compatible with the same cloudSwXtches.



©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Below is a portion of the Settings call response detailing information on High Availability:

Bash Copy


```
"haSettings": {
        "uid": "86A6BDD5-128D-E31D-4A7B-33D846C94CB2",
        "name": "ha",
        "paths": [
            {
                "name": "path_1",
                "swxtches": [
                    "10.2.128.93"
                ]
            },
            {
                "name": "path_2",
                "swxtches": [
                    "10.1.1.6"
                ]
            }
            {
                "name": "path_3",
                "swxtches": [
                    "10.5.1.6"
                ]
            }
```


Create HA

swxtch/ha/v1/create

Get HA List

swxtch/ha/v1/show

Delete cloudSwXtch from HA

swxtch/ha/v1/removeSwxtch

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

POST **swxtch/ha/v1/removeSwxtch**

- Removes a cloudSwXtch from a HA configuration

URL:

Bash Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/ha/v1/removeSwxtch
```

Example URL:

http://10.1.1.6/swxtch/ha/v1/removeSwxtch

Example Request Body:

Bash Copy

```
{
  "swxtch": "10.5.1.6"
}
```

Responses:

200 - successful operation

Example Value:

Bash Copy

```
{
    "Message": "Swxtch 10.5.1.6 removed sucessfully from HA"
}
```

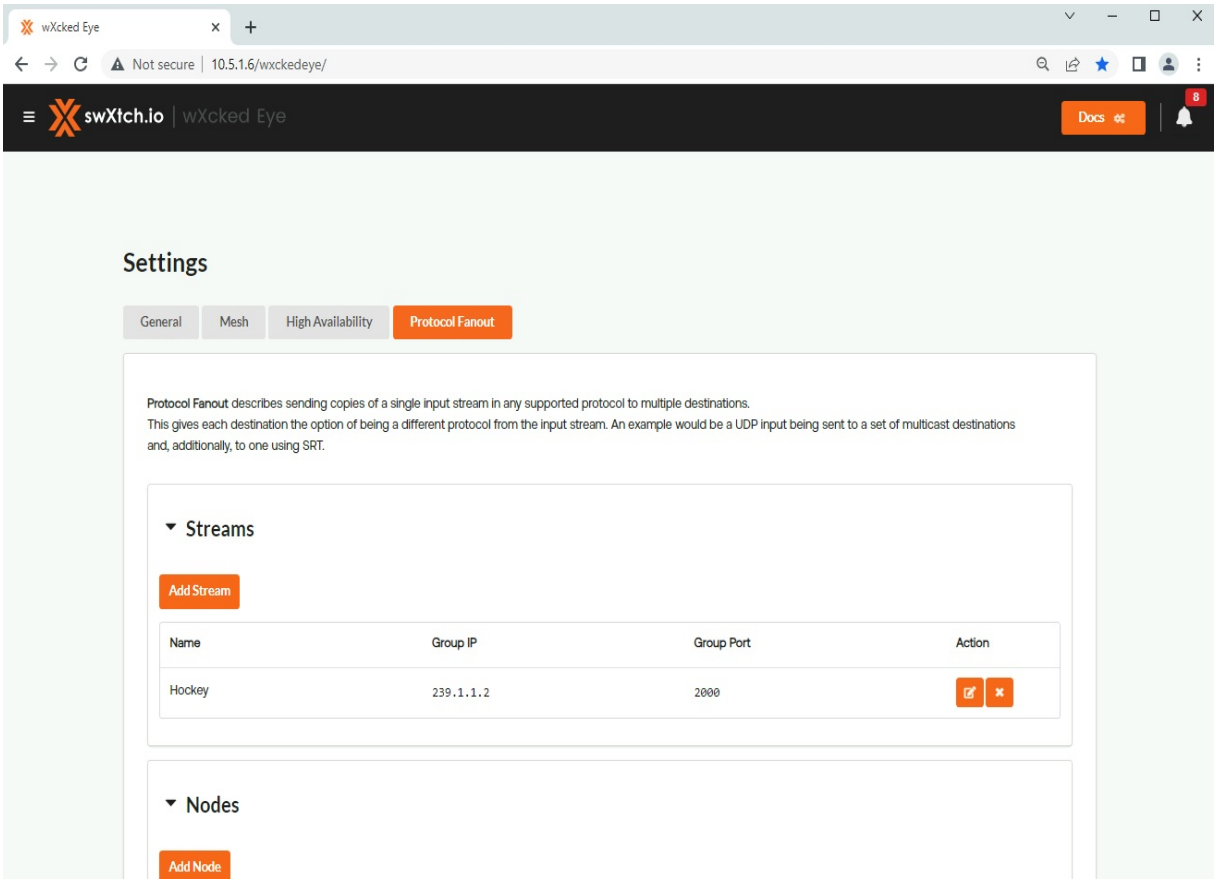
Destroy HA

swxtch/ha/v1/destroy

Protocol Fanout

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

The Protocol Fanout tab shows the cloudSwXtches that are configured for Protocol Fanout: multicast, UDP (Unicast), SRT Caller and SRT Listener.



Below is a portion of the Settings call response detailing information on Protocol Fanout.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
"unicastSettings": {
        "unicastToMulticast": {
            "baseAddr": "239.1.1.1",
            "portRange": [
                2000,
                2015
            ],
            "disable": false
        },
        "multicastToUnicast": {
            "adaptors": {
                "4009820932": {
                    "targetIp": "239.1.3.4",
                    "targetMac": "FF:FF:FF:FF:FF:FF",
                    "groupIp": "239.2.1.1"
                }
            }
        },
        "streams": {
            "streams": {
                "239.1.1.1": "Hockey"
            }
```

Streams

> Streams vs Nodes
> When configuring Protocol Fanout on wXcked Eye, users can assign names to their streams in order to easily configure Ingress/Egress SRT streams or unicast to multicast fanout. Nodes can be used in a similar manner when specifying an SRT Caller. However, both the streams and nodes are not required for the configuration of protocol fanout. They are more useful when configuring via wXcked Eye.

Add Streams

swxtch/entities/streams/v1/add

Update Stream

swxtch/entities/streams/v1/update

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Show Streams

swxtch/entities/streams/v1/show

Remove Streams

swxtch/entities/streams/v1/remove

Nodes

> Streams vs Nodes
> When configuring Protocol Fanout on wXcked Eye, users can assign names to their streams in order to easily configure Ingress/Egress SRT streams or unicast to multicast fanout. Nodes can be used in a similar manner when specifying an SRT Caller. However, both the streams and nodes are not required for the configuration of protocol fanout. They are more useful when configuring via wXcked Eye.

Add Nodes

swxtch/entities/nodes/v1/add

Update Node

swxtch/entities/nodes/v1/update

Show Node

swxtch/entities/nodes/v1/show

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Remove Node

swxtch/entities/nodes/v1/remove

UDP (Unicast Fanout)

UDP Enable

swxtch/protocols/udp/v1/ingress/enable

POST **swxtch/protocols/udp/v1/ingress/enable**

- Enables the ingress of unicast streams into the cloudSwXtch via Protocol Fanout by mapping a unicast port range to a multicast group and port.

URL:

Bash Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/protocols/udp/v1/ingress/enable
```

Example URL:

http://10.2.128.10/swxtch/protocols/udp/v1/ingress/enable

Example Request Body:

Bash Copy

```
{
  "baseGroupIp": "239.5.69.2",
  "groupPort": 5001,
  "portRange": [
    2000,
    2005
  ]
}
```

Example Response:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
Success: true{
    "Message": "Success"
}
```

UDP Egress Join

swxtch/protocols/udp/v1/egress/join

POST **swxtch/protocols/udp/v1/egress/join**

- Enables the egress of unicast streams from the cloudSwXtch via Protocol Fanout egress by mapping the multicast IP and port to a target node.

URL:

Bash Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/protocols/udp/v1/egress/join
```

Example URL:

http://10.2.128.10/swxtch/protocols/udp/v1/egress/join

Example Request Body:

Bash Copy

```
{
  "groupIp": "239.5.69.2",
  "groupPort": 5001,
  "targetIp": "10.5.1.6",
  "targetPort": 6001,
  "targetMac": "00:22:48:24:77:db"
}
```

Example Response:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
{
    "message": "Added 10.5.1.6:6001 to multicast group
239.5.69.2:5001"
}
```

UDP Ingress Show

swxtch/protocols/udp/v1/ingress/show

UDP Egress Show

swxtch/protocols/udp/v1/egress/show

UDP Leave

swxtch/protocols/udp/v1/egress/leave

UDP Disable

swxtch/protocols/udp/v1/ingress/disable

POST **swxtch/protocols/udp/v1/ingress/disable**

- Disables the ingress of unicast streams into cloudSwXtch.

URL:

Bash Copy

```
curl http://<cloudSwXtch-control-
IP>/swxtch/protocols/udp/v1/ingress/disable
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Example URL:

http://10.2.128.10/swxtch/protocols/udp/v1/ingress/disable

Example Request Body:

Example Response:

Bash Copy

```
Success: true{
    "Message": "Success"
}
```

SRT Caller

SRT Caller Ingress Enable

swxtch/protocols/srt-caller/v1/ingress/enable

POST **swxtch/protocols/srt-caller/v1/ingress/enable**

- Enables the ingress of SRT data into the cloudSwXtch via SRT Caller Protocol Fanout by mapping an target IP and port to a multicast group and port.

URL:

Bash Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/protocols/srt-caller/v1/ingress/enable
```

Example URL:

http://10.2.128.10/swxtch/protocols/srt-caller/v1/ingress/enable

Example Request Body:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bash Copy

```
{
  "groupIp": "225.1.1.1",
  "groupPort": 5500,
  "targetIp": "10.2.128.75",
  "targetPort": 1599
}
```

Example Response:

Bash Copy

```
Success: true{
    "Message": "Success"
}
```

SRT Caller Ingress Show

swxtch/protocols/srt-caller/v1/ingress/show

SRT Caller Egress Join

swxtch/protocols/srt-caller/v1/egress/join

POST **swxtch/protocols/srt-caller/v1/egress/join**

- Enables the egress of SRT data from the cloudSwXtch via SRT Caller Protocol

Fanout egress by mapping the multicast IP and port to a target node.

URL:

Bash Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/protocols/srt-caller/v1/egress/join
```

Example URL:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

http://10.2.128.10/swxtch/protocols/srt-caller/v1/egress/join

Example Request Body:

Bash Copy

```
{
  "groupIp": "225.1.1.1",
  "groupPort": 5500,
  "targetIp": "10.2.128.15",
  "targetPort": 1599
}
```

Example Response:

Bash Copy

```
{
    "message": "Added 10.2.128.15:6001 to multicast group
225.1.1.1:1599"
}
```

SRT Caller Egress Show

swxtch/protocols/srt-caller/v1/egress/show

SRT Caller Egress Leave

swxtch/protocols/srt-caller/v1/egress/leave

SRT Caller Ingress Disable

swxtch/protocols/srt-caller/v1/ingress/disable

SRT Listener

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

SRT Listener Ingress Enable

swxtch/protocols/srt-listener/v1/ingress/enable

POST **swxtch/protocols/srt-listener/v1/ingress/enable**

- Enables the ingress of SRT data into the cloudSwXtch via SRT Listener Protocol Fanout by mapping a port range to a multicast group and port.

URL:

Bash Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/protocols/srt-listener/v1/ingress/enable
```

Example URL:

http://10.2.128.10/swxtch/protocols/srt-listener/v1/ingress/enable

Example Request Body:

Bash Copy

```
{
  "baseGroupIp": "225.1.1.1",
  "groupPort": 5500,
  "portRange": [
    1599,
    1602
  ]
}
```

Example Response:

Bash Copy

```
Success: true{
    "message": "Success"
}
```

SRT Listener Ingress Show

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

swxtch/protocols/srt-listener/v1/ingress/show

SRT Listener Egress Join

swxtch/protocols/srt-listener/v1/egress/join

POST **swxtch/protocols/srt-listener/v1/egress/join**

- Enables the egress of SRT data from cloudSwXtch via SRT Listener Protocol Fanout egress by mapping the multicast IP and port to a port range.

URL:

Bash Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/protocols/srt-listener/v1/egress/join
```

Example URL:

http://10.2.128.10/swxtch/protocols/srt-listener/v1/egress/join

Example Request Body:

Bash Copy

```
{
  "portRange": [
    6000,
    6003
  ],
  "groupIp": "225.1.1.1",
  "groupPort": 6001
}
```

Example Response:

Bash Copy

```
{
    "message": "Added listener in range [6000,6003] to multicast group 225.1.1.1:6001"
}
```

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

SRT Listener Egress Show

swxtch/protocols/srt-listener/v1/egress/show

SRT Listener Egress Leave

swxtch/protocols/srt-listener/v1/egress/leave

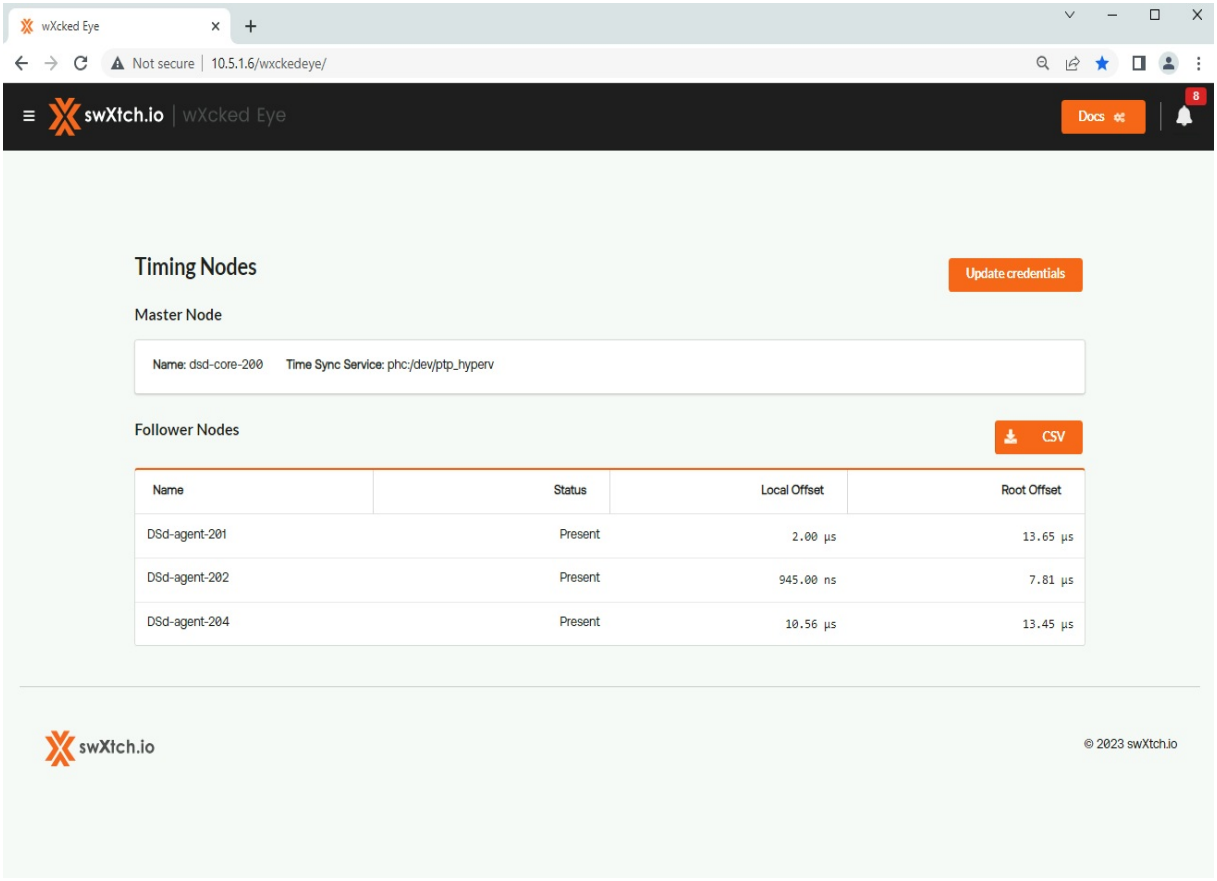
SRT Listener Ingress Disable

swxtch/protocols/srt-listener/v1/ingress/disable

Precision Timing Protocol (PTP)

The Timing Nodes page displays information regarding clock sync configuration for the cloudSwXtch. The page in wXcked Eye will only populate with information if the user has the PTP feature enabled.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.



PTP Master

swxtch/ptp/v1/master

PTP Slaves

swxtch/ptp/v1/slaves

Update PTP Credentials

swxtch/ptp/v1/updateTimebeatCredentials

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Release Notes

WHAT TO EXPECT

In this section, users will find release notes for current and past versions of our product. If you have any questions regarding your particular version, please contact us at info@swxtch.io. To see past versions of our documentation site, please check out our **Resources** page.

2.3.2 Release Notes - May 24, 2024

New Features:

- **A new and improved way of seeing things with wXcked Eye!**
    - The wXcked Eye Topology graph has been overhauled, introducing a slew of new features to allow for real-time monitoring with new streams and channels views.
        - Each graph component opens an information sidebar when selected. This information includes general, environment, hardware, total rx/tx, related streams, and related adaptors.
        - A new Layers toggle feature allows for users to view different combinations of data as required: name components, clouds, streams, rates, regions, sockets, instances and TimeBeat.
    - A new Topology Table view gives users a detailed look into individual streams and channels. Additionally, users can view streams in a tree view with components pulled directly from the topology graph.
    - An updated Support page allows users to define log ranges by number of days and/or hours.
    - The Settings page has a new Channels tab, allowing users to add channels based on streams linked by converter. In addition, users can specify certain colors for HA paths, streams, and channels to carry throughout the wXcked Eye UI.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **Stay on top of your cloudSwXtch metrics with swXtch-top!**
    - Similar to wXcked Eye, swXtch-top has been updated with additional views to provide additional metrics on channel data flow.
        - Streams and StreamsLinks views give users a tree view of data flow from the cloudSwXtch to relevant endpoints.
        - HA Paths, HA Summary and HA views has been consolidated into a singular HA page.
        - The Environment view has been updated to give an extensive hardware breakdown of the cloudSwXtch and relevant components.

Known Issues:

- There are IP conflicts when multiple xNICs on different VPCs have a ctrl IP with similar host names. (4857)
    - **Workaround**: Avoid using the same IP and hostname between VPCs.
- /stats/xnics can show multiple entries in RxMulticastGroups for the same Group Ip. (4893)
    - **Workaround**: None. This will be addressed in an upcoming release.
- When a cloudSwXtch is removed from an xNIC Type 2 cluster, the General Stats reset.

2.2.0 Release Notes - March 18, 2024

New Features:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **Say hello to updates to Bridge Type 2!**
    - **Protocol Fanout** is now available on the cloudSwXtch Bridge in Type 2. Previously, it was only available on the cloudSwXtch in the cloud. For more information on configuring the cloudSwXtch Bridge for Protocol Fanout, see the following article.
    - **2022-7 for High Availability** is now configurable on a cloudSwXtch Bridge in both Type 2. For more information, see the High Availability article under Configuring cloudSwXtch.
- **Keep things secure with our verifiable xNIC installers!**
    - We now provide an optional way to verify the legitimacy of our xNIC installer script and installation packages against signed hash files with a swXtch.io public key. For more information, see Verifying Installer Files in the Install xNIC on Linux articles.
        - Currently, only available on Linux.

New Improvements:

- **Improved performance for AWS:**
    - c6in or m6in EC2 instance families are now recommended for cloudSwXtch instances running version 2.2.0 or greater. This ensures that users get the best performance out of their cloudSwXtch network on AWS. Any current m5zn cloudSwXtch based instances should migrate to one of the two family types.
- **Updates to swXtch-top:**
    - A new Bridge view has been added to swXtch-top, displaying data flow stats specifically for Bridge Type 2. For more information, see swXtch-top article.

Bug Fixes:

- Fixed issue where a double join was occurring in Cloud-to-Ground.
- Fix stream stats in swxtch-top showing intermittent negative values when the replicator service is restarted.

Known Issues:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- There are IP conflicts when multiple xNICs on different VPCs have a ctrl IP with similar host names. (4857)
    - **Workaround**: Avoid using the same IP and hostname between VPCs.
- /stats/xnics can show multiple entries in RxMulticastGroups for the same Group Ip. (4893)
    - **Workaround**: None. This will be addressed in an upcoming release.

2.1.3 Release Notes - March 5, 2024

New Improvements:

- **Updates to swXtch-top**:
    - A new Search bar in the interface will allow users to filter items in the xNICs, Streams and Notification views. (4743)
    - Navigating between swXtch-top's views will require hitting both the CTRL button on a user's keyboard and the associated letter. For example, to get to the xNICs view, a user will need to key in CTRL + X to switch to that page.
    - The Hardware view is now called "Environment."
    - A new Notifications view has been added. (4709)
    - Both the HA Summary and HA view will only list in the navigation menu when HA is configured.
        - For more information, check out the swxtch-top page under Testing cloudSwXtch.
- **Updates to wXcked Eye**:
    - The wXcked Eye Navigation menu will now properly scroll if using the UI on a lower resolution screen. (5029) [Known Issue 2.1.2]
- **Updates to swx support**:
    - Users can now export xNIC logs using the swx support command. For more information, see How to View cloudSwXtch Logs for Troubleshooting. (4574)

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Bug Fixes:

- We corrected a bug that would happen when PTP service start-up fails and the master node in swxtch-top shows as empty. (5110) [Known Issue 2.1.2]

Known Issues:

- There are IP conflicts when multiple xNICs on different VPCs have a ctrl IP with similar host names. (4857)
    - **Workaround**: Avoid using the same IP and hostname between VPCs.
- /stats/xnics can show multiple entries in RxMulticastGroups for the same Group Ip. (4893)
    - **Workaround**: None. This will be addressed in an upcoming release.

2.1.2 Release Notes - February 15, 2024

New Improvements:

- **Updates to xNIC:**
    - Users can now download and install the xNIC without having a valid license authorizing their cloudSwXtch. However, to begin packet flow from the cloudSwXtch to the xNIC(s), a license is required. (4974)
- **Updates to swx support:**
    - New parameters have been added to the swx support command. For a complete list, please see How to View cloudSwXtch Logs for Troubleshooting. (4574)
- **Updates to wXcked Eye:**
    - The Protocol Fanout stats page will now display the Alias name by the stream IP address. The page also allows users to filter Adaptors by multiple protocols. (4572)
    - The Protocol Fanout tab in Settings will also now display the Alias name next to the Stream and Node IP addresses in the Adaptors table. (4572)
    - The Support page will now allow users to export xNIC logs.(4574)

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **Updates to Licensing Server:**
    - The Licensing Server will no longer require a manual start after initial install. (4919) [Known issue in 2.1.1]

Known Issues:

- **[Resolved 2.1.3]** PTP service start-up fails and swxtch-top shows that the master node is empty. (5110)
    - **Workaround**: Use the following four commands in succession to remedy this issue:

```
PowerShell                                                    Copy

sudo systemctl start elasticsearch.service
sudo systemctl enable elasticsearch.service
sudo systemctl start grafana-server.service
sudo systemctl enable grafana-server.service
```

- There are IP conflicts when multiple xNICs on different VPCs have a ctrl IP with similar host names. (4857)
    - **Workaround**: Avoid using the same IP and hostname between VPCs.
- **[Resolved 2.1.3]** wXcked Eye navigation menu does not scroll.
    - **Workaround**: Ensure you're using wXcked Eye in a high-res screen.
- /stats/xnics can show multiple entries in RxMulticastGroups for the same Group Ip. (4893)
    - **Workaround**: None. This will be addressed in an upcoming release.

2.1.1 Release Notes - February 5, 2024

New Improvements:

- **Updates to wXcked Eye:**
    - The Alias tab is now split into two separate pages, one for streams and exclusions and another for nodes.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **Updates to Prometheus API:**
    - swx_maxClientCount and swx_numClientsConnected are two new metrics added to Prometheus while both swx_xnic_maxActiveConnections and swx_xnic_activeConnectionCount have been deprecated.
    - When calling for metrics regarding nodes, only current active ones will display. Inactive or disconnected nodes will no longer list.
- **Updates to xNIC Logs:**
    - File paths in Windows and Linux for xNIC logs have been changed. Please see the article, How to Find xNIC Logs, for more information.
- **Updates to xnic.json file**:
    - “childs” has been updated to “children” in the xnic.json file for configuring HA paths. For more information, please see High Availability under Configuring cloudSwXtch.

Known Issues:

- **[RESOLVED in v2.1.2]** The Licensing Server does not automatically start after install. (4919)
    - **Workaround**: After the initial installation, please use the following command to start the licensing server.

      PowerShell | Copy

      ```
      sudo systemctl start swxtch-license-svr
      ```

      Subsequent start-ups will happen automatically when the licensing server VM is turned on.
- There are IP conflicts when multiple xNICs on different VPCs have a ctrl IP with similar host names. (4857)
    - **Workaround**: Avoid using the same IP and hostname between VPCs.
- wXcked Eye navigation menu does not scroll.
    - **Workaround**: Ensure you’re using wXcked Eye in a high-res screen.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- /stats/xnics can show multiple entries in RxMulticastGroups for the same Group Ip. (4893)
    - **Workaround**: None. This will be addressed in an upcoming release.

2.1.0 Release Notes - January 2024

New Features:

- **Users can now configure their High Availability for Multiple Multicast Groups (mMC-7)**. For more information on how to configure the xNIC for mMC-7, please see High Availability under Configuring cloudSwXtch. (4239)
- The cloudSwXtch can now perform **Source Specific Multicast (SSM)** as a method of delivering multicast packets where the receiver includes or excludes them through a specific source address.
- A new **swx support** command is now available for users to export a tar.gz file detailing statistical data stored within a cloudSwXtch during a set period of time. This is especially helpful when troubleshooting with swXtch.io Support. (4107, 4488)
- A self-hosted Licensing Server is now available for installation on your cloudSwXtch network. This allows users to self manage cloudSwXtch licenses instead of having to request one for each instance from swXtch.io. Read more about it here.
- Kubernetes support expanded to include GCP (GKE) and AWS (EKS) in addition to existing Azure (AKS) support.

New Improvements:

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **Updates to wXcked Eye:**
    - The new wXcked Eye Cluster view provides users with an expansive look of all cloudSwXtches connected in a high availability or mesh configuration. Serving as the new main page, users can easily navigate between cloudSwXtches for additional metrics. A universal Cluster quick view side panel has also been added, displaying up to 5 cloudSwXtches. (4568)
    - A new wXcked Eye Support Page has been added, allowing users to export a JSON file with a full cloudSwXtch report over a certain period of time. This is especially helpful when troubleshooting with swXtch.io Support. (4269)
    - A new batch select and delete feature has been added to the Aliases tab under Settings. (4320)
    - Users can now exclude certain streams via IP addresses from wXcked Eye and swXtch-top monitoring. (3732)
    - The Hardware panel in the General tab under Settings now lists metadata and OS for the control and data subnets. (4351, 4352)
    - A new Protocol Fanout Stats page displays SRT and RIST data flow to and from the cloudSwXtch. Previously, only multicast data flow was displayed across the UI. (4322)
    - A new cloudSwXtch information banner now displays at the top of each page in wXcked Eye. (4571)
    - The Network Graph is now called Topology.
    - Timing Nodes page has been relocated to under Monitoring.
    - wXcked Eye is fully scalable with responsive views and page resizing. (4328)
    - wXcked Eye and swXtch-top are both unified in their performance metrics and cloudSwXtch information.
    - Users can now return to previously visited pages in the UI using a navigation history tool at the top of each page.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **Updates to xNIC:**
    - **The xNIC installer script will automatically install xNIC Type 2** as the **default** and **recommended** xNIC. From now on, whenever the documentation refers to the xNIC, it means Type 2. (4249)
    - The xNIC installer script will automatically open recommended ports in a user's firewall.
    - HA Configuration for an xNIC is now automatic as long as only one multicast group is required. (4305, 4460)
    - The xNIC configuration file is now a .json.
    - Precision Time Protocol (PTP) install is now optional for both the cloudSwXtch and the xNIC. (4368)
    - The xNIC has been enhanced to work with a single-NIC configuration if the cloudSwXtch's ctrl and data NIC share a single subnet.
    - Support for the following Operating Systems has been added: AlmaLinux 8.8, Amazon Linux 2023, RHEL 9.2, Rocky Linux 8 + 9, and Ubuntu 22.04. For a comprehensive list, see xNIC System Requirements. (4414, 4314, 3371, 4261, 4154)
- **Updates to xNIC on K8s:**
    - The procedure to install the xNIC on a Kubernetes cluster has been updated with a new installer. See Install xNIC on K8s Cluster for more information. (3131, 4358)
- **Updates to Prometheus API:**
    - The Prometheus API metrics are now **swx_core** for the cloudSwXtch and **swx_xnic** for the xNICs. (4437)
    - The data for the xNICs are now grouped together, allowing users to filter by name.

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

- **Updates to swXtch-top:**
    - A new Hardware view has been added to swxtch-top, displaying metadata and OS for control and data subnets. (4351)
    - xNIC will now display as either Type 1 (t1) or Type 2 (t2). Type 2 is the default xNIC. (4445)
- **Updates to swXtch-perf:**
    - Single Source Multicast (SSM) include and exclude consumer commands have been added. In addition, a command to show the bps of all packet headers has also been included. For more information, see swxtch-perf article. (4386,
- **Bridge Installation**
    - Bridge installation command has been updated.

Bug Fixes:

- Fixed an issue with Protocol Fanout Egress Adaptors not working with xNIC Type 2. (4520)

2.0.34 Release Notes - October 2023

2.0.10 Release Notes - July 2023

1.9.85 Release Notes

1.9.73 Release Notes

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Resources

PDFs

cloudSwXtch User Guide v1.9.85

cloudSwXtch-v1-9-85 55.41 MB

cloudSwXtch User Guide v2.0.34

cloudSwXtch User Guide v2.1.0

cloudSwXtch User Guide v2.1.1

cloudSwXtch User Guide v2.1.2

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.

Quotas

cloudSwXtch

All bandwidth and packet per second values are aggregate values (i.e ingress + egress) unless otherwise noted.

Name	Default Value	Configurable
Multicast Packet Size	Up to 3750	Yes
Endpoint Connections	Unlimited	NA
Max Throughput per cloudSwXtch	Up to 100 Gb/s	No
Max Bandwidth per flow	Up to 15 Gb/s	No
Max Packets per second per cloudSwXtch	Up to 10M	No
Max cloudSwXtch instances per mesh	32	No
Max Bridge instances per cloudSwXtch	4	No
Max fanout outputs per cloudSwXtch	1000	No

cloudSwXtch Sizing

cloudSwXtch Multicast (Marketplace)

# Endpoints	Bandwidth	Core	Memory	Hard Drive
10 (max)	100 Mbps (max)	8	16GB DDR	64GB SSD

cloudSwXtch BYOL (Marketplace)

# Endpoints	Bandwidth	Core	Memory	Hard Drive
Up to 100	2 Gb/s (max)	16+	16GB DDR	64GB SSD
Up to 200	More than 2Gb/s	64+	16GB DDR	64GB SSD

xNIC

Name	Default Value	Configurable
Multicast Packet Size	Up to 3750	Yes
Multicast Groups	Unlimited	NA
Max cloudSwXtch Connections	4	No
Max Bandwidth	Up to 15 Gb/s	Yes

Powered by Document360

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io. All rights reserved.