

CLOUDSWXTCH

Version 2.1.2



Table of Contents

cloudSwXtch > Getting Started

Getting Started	5
What is cloudSwXtch?	7

cloudSwXtch > Getting Started > cloudSwXtch Features

cloudSwXtch Licensable Features	9
Multicast	12
Source Specific Multicast	13
Broadcast	14
High Availability	15
Mesh	17
Bridge	19
Precision Time Protocol	20
Protocol Conversion and Fanout	21

cloudSwXtch > Installing cloudSwXtch

cloudSwXtch System Requirements	23
---------------------------------------	----

cloudSwXtch > Installing cloudSwXtch > cloudSwXtch on AWS

cloudSwXtch on AWS	26
Validate Subnets on AWS	27
Verify Security Groups	31
Create SSH Key Pair	34
Install cloudSwXtch on AWS	37
Deploy cloudSwXtch with Terraform on AWS	46
Check Health of cloudSwXtch Instance on AWS	50
Delete cloudSwXtch on AWS	52

cloudSwXtch > Installing cloudSwXtch > cloudSwXtch on Azure

cloudSwXtch on Azure	53
Validate Subnets on Azure	56
Create an Azure cloudSwXtch Template	59
Install cloudSwXtch on Azure	62
Install cloudSwXtch via Azure Marketplace	68
Deploy cloudSwXtch with Terraform on Azure	76
Install cloudSwXtch for an Air-Gapped Environment	80

cloudSwXtch > Installing cloudSwXtch > cloudSwXtch on GCP

cloudSwXtch on GCP	90
Install cloudSwXtch via GCP Marketplace	91

cloudSwXtch > Installing cloudSwXtch > cloudSwXtch on OCI

cloudSwXtch on OCI	100
Install cloudSwXtch via OCI Marketplace	101
Cloud agnostic cloudSwXtch VM Install	112

cloudSwXtch > Upgrading cloudSwXtch

Upgrading cloudSwXtch	114
-----------------------------	-----

cloudSwXtch > Installing cloudSwXtch Bridge

Installing cloudSwXtch Bridge	115
Install cloudSwXtch Bridge Type 2	116
Install cloudSwXtch Bridge Type 1	120

cloudSwXtch > Installing xNIC

Installing xNIC	121
xNIC System Requirements	122

cloudSwXtch > Installing xNIC > Install xNIC on Linux

Install xNIC on Linux	124
xNIC Linux Uninstall	128
xNIC Linux Upgrade	129

cloudSwXtch > Installing xNIC > Install xNIC on Windows

Install xNIC on Windows	131
Uninstall xNIC on Windows	137
Upgrade xNIC on Windows	138

cloudSwXtch > Installing xNIC > Install xNIC on Kubernetes

Install xNIC on Kubernetes	140
Install xNIC on K8s Cluster	142
Uninstall xNIC on K8s	152
Test xNIC with K8s	153
Upgrade xNIC nodes on K8s	160

cloudSwXtch > Using wXcked Eye for cloudSwXtch

Using wXcked Eye for cloudSwXtch	161
--	-----

cloudSwXtch > Using wXcked Eye for cloudSwXtch > Monitor cloudSwXtch with wXcked Eye

Monitor cloudSwXtch with wXcked Eye	163
wXcked Eye Cluster Page	165
wXcked Eye cloudSwXtch Page	168
wXcked Eye xNICs Page	171
wXcked Eye Topology	175
wXcked Eye Protocol Fanout Stats	178
wXcked Eye Timing Nodes	181
wXcked Eye Support Page	183

cloudSwXtch > Using wXcked Eye for cloudSwXtch > Configure cloudSwXtch with wXcked Eye

Configure cloudSwXtch with wXcked Eye	185
General	187
Mesh with wXcked Eye	189
High Availability with wXcked Eye	194
Protocol Conversion and Fanout with wXcked Eye	199
Protocol Conversion and Fanout Example	206
Stream and Nodes	210

cloudSwXtch > Configuring cloudSwXtch

High Availability	215
Mesh	226
Bridge Type 2	231
Bridge Type 1	236
Protocol Conversion and Fanout	238

cloudSwXtch > Monitoring cloudSwXtch

Azure Monitoring	239
Prometheus Monitoring	242

cloudSwXtch > Testing cloudSwXtch

Testing cloudSwXtch	255
swxtch-perf	256
swxtch-top	262
swxtch-tcpdump	273
swxtch-where	274
Troubleshooting	278

cloudSwXtch > cloudSwXtch How To's

How to View cloudSwXtch Logs for Troubleshooting	279
How to View cloudSwXtch Bridge Logs	283
How to Find xNIC Logs	286
How to License a cloudSwXtch	292
How to Install a cloudSwXtch Licensing Server	293
How to set MTU size	296
How to Peer between VPCs in Different Regions for AWS	302

cloudSwXtch > Market Use Cases > Media Use Cases

Media Use Cases	308
Media Multicast made easy with cloudswXtch	309
Hitless Merge - 2022-7	311
Media support for Compressed and Uncompressed Workflows	312
Protocol Fanout	314
Disaster Recovery	315

cloudSwXtch > cloudSwXtch API

Monitoring API	316
Configuration API	330

cloudSwXtch > Resources

Resources	341
-----------------	-----

cloudSwXtch

Release Notes	342
Quotas	343
FAQ	344

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](#)

[xNIC Installation Guide for Windows and Linux](#)

wXcked Eye for Monitoring and Orchestration

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 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](#).

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

Multicast Examples

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

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

Supported Enviroments

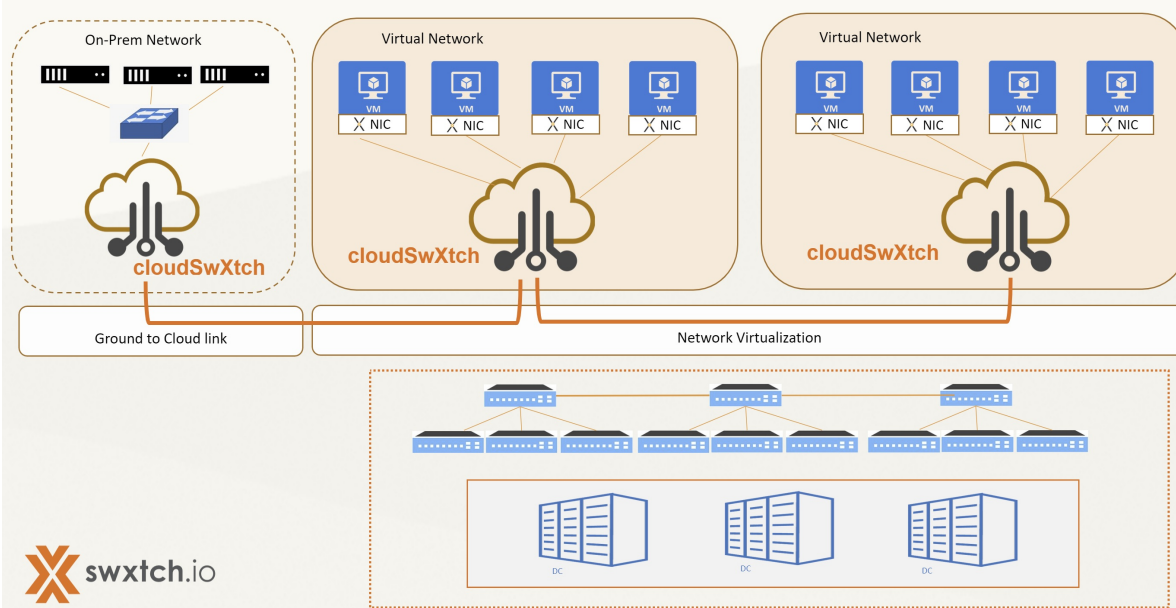
- Microsoft's [Azure](#)
- Amazon's [AWS](#)
- Google's [GCP](#)
- Oracle's [OCI](#)
- On-Premises with [cloudSwXtch Bridge](#)

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.

cloudSwXtch Hybrid Network



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.
- **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.

cloudSwXtch Licensable 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 swtch-perf, the cloudSwXtch-based tool for simulating traffic movement, or via an external application.

Protocol Conversion and Fanout

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. Similarly, a unicast input to cloudSwXtch can be mapped to a multicast group enabling multiple 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.

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](#).

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.

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.

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.

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.

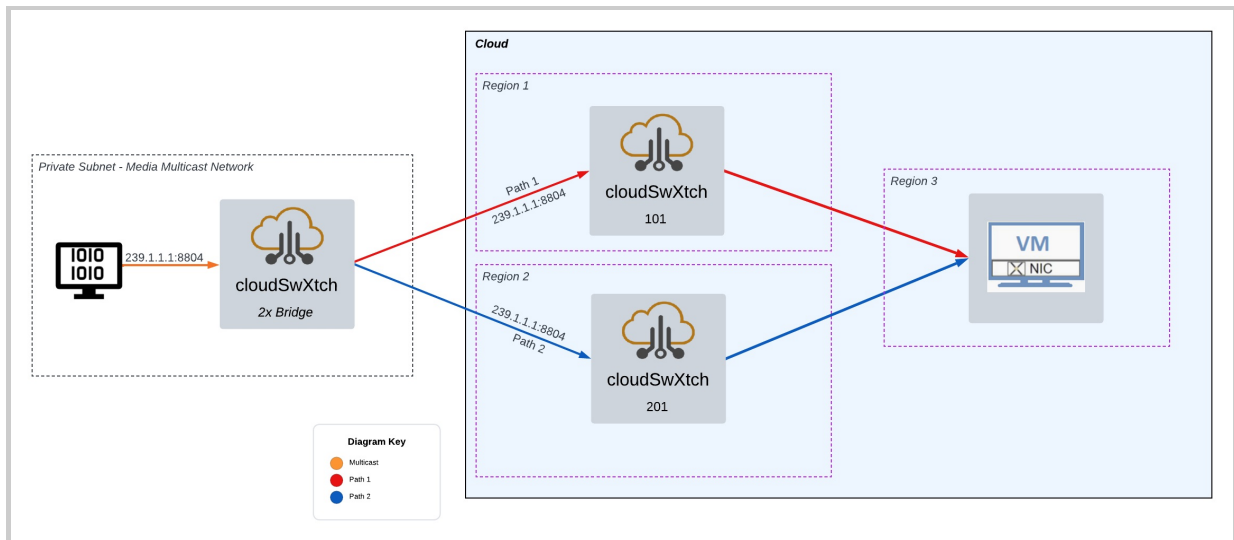
In addition, there is no limit to the number of multicast groups per data path. If one cloud, availability zone or region should go down, then the data is still sent in the other 2-8 paths, ensuring that the consumer gets the necessary data. Consumers can also be put into different clouds, availability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



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 swtch-top, see [swtch-top](#) section 4-6.

To configure the system for high availability, refer to: [High Availability Configuration](#).

Installing cloudSwXtch - Firewall Exceptions

When installing the cloudSwXtch, high availability requires special firewall exceptions. To learn more, see [cloudSwXtch System Requirements](#).

Mesh

What is a Mesh?

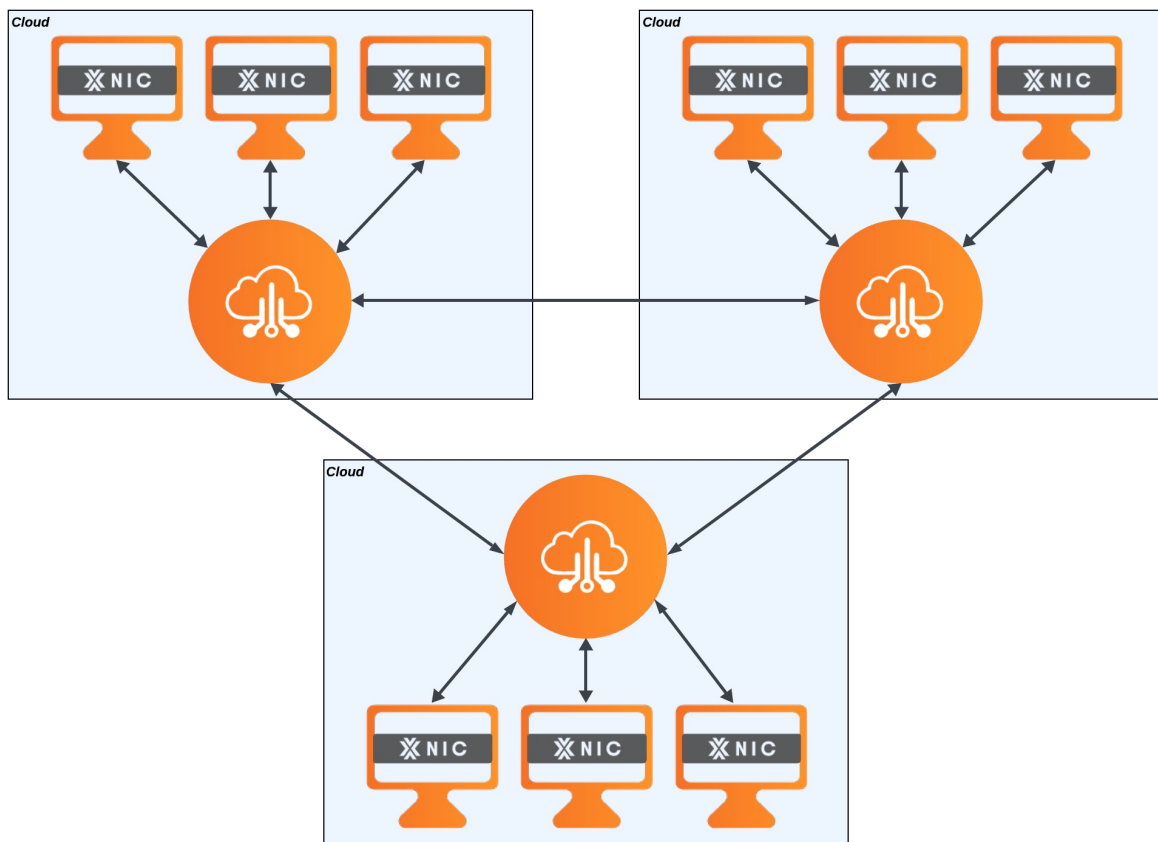
A mesh connects cloudSwXtches in a variety of dispersed network locations – different Vnets, regions, clouds, subnets, data centers, ect.). Additionally, a mesh is a way to group two or more swXtch's together to act as one to gain network performance.

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.



NOTE

- A member of a mesh is called a swtch-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.

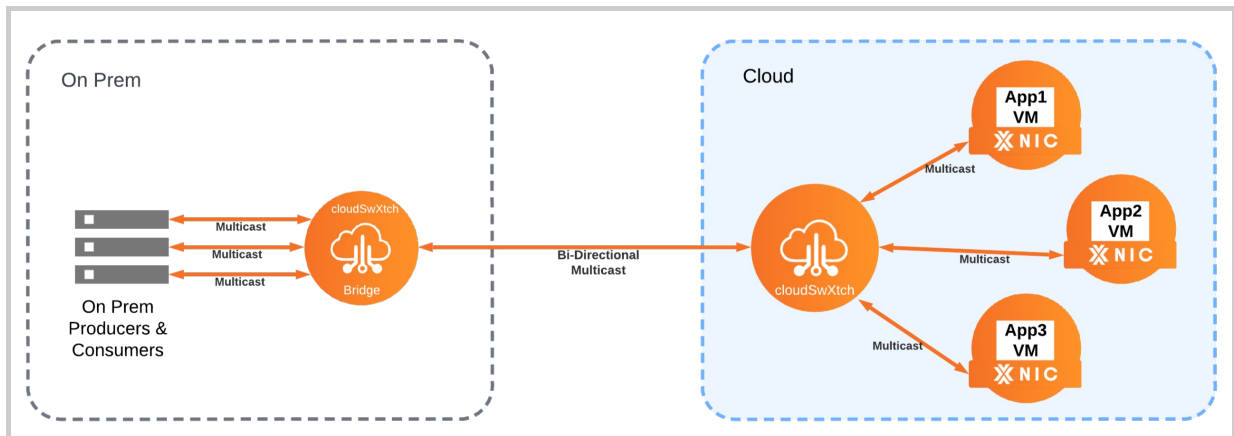
IMPORTANT

- 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.

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.

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](#).

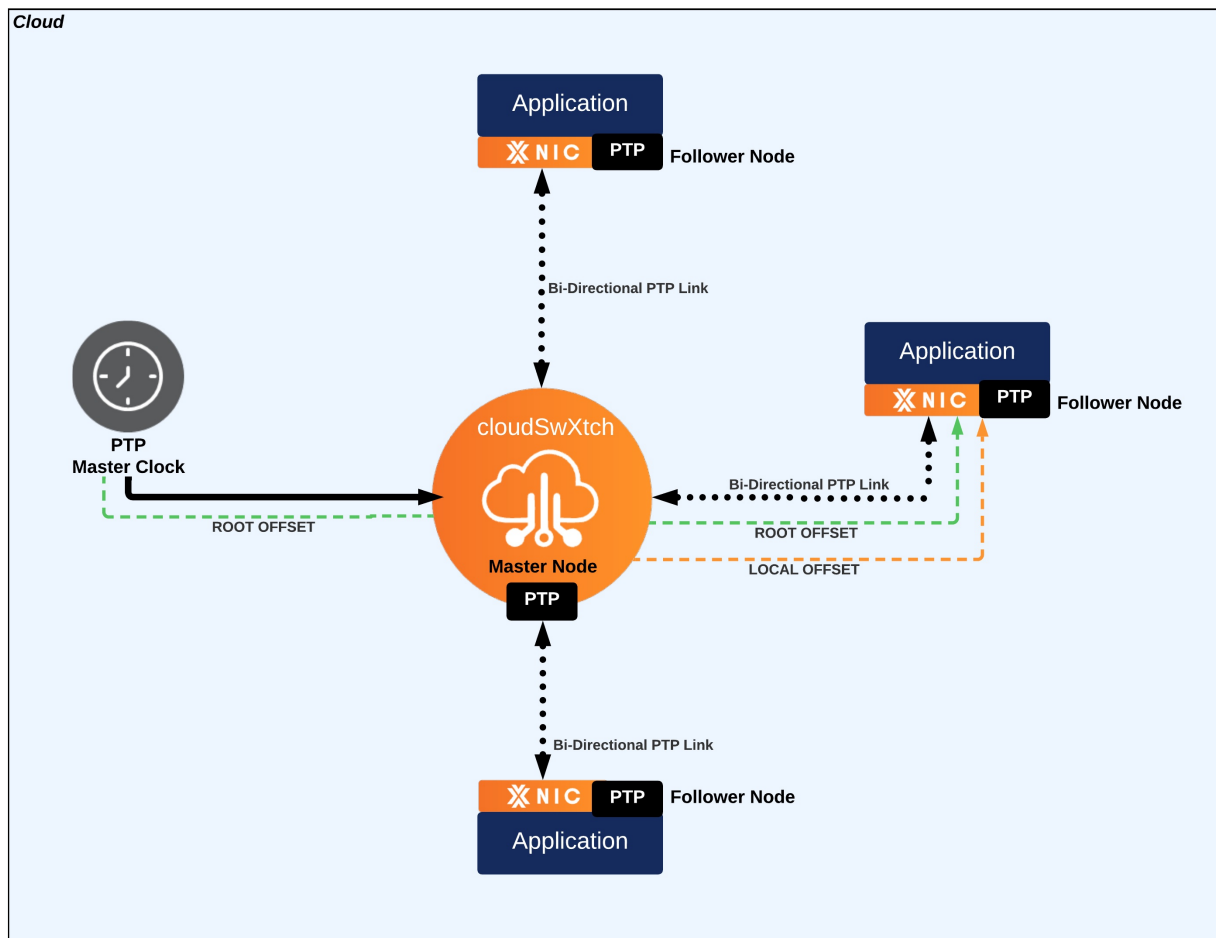
Precision Time Protocol

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.

Protocol Conversion and Fanout

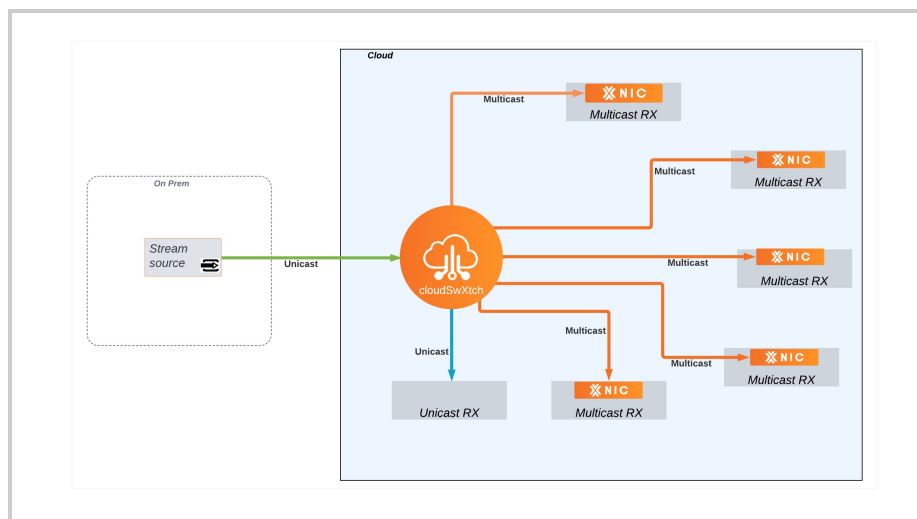
WHAT TO EXPECT

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.

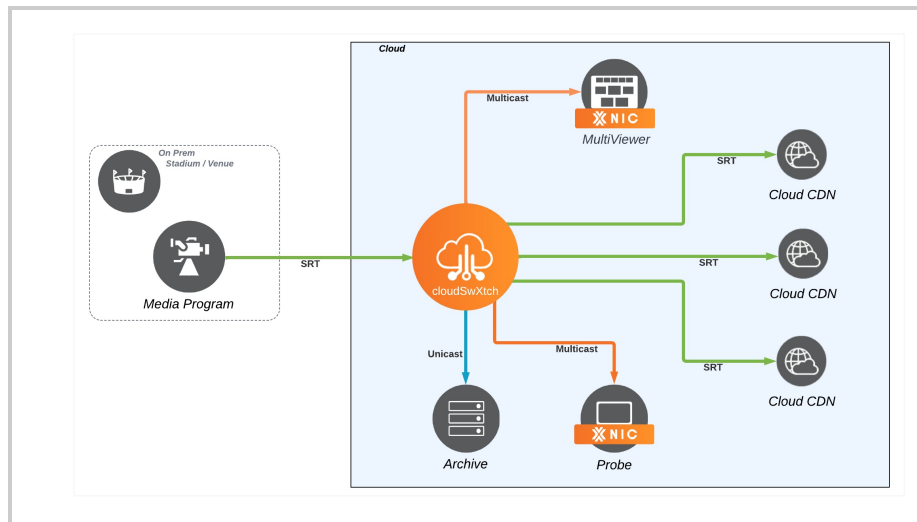
What is Protocol Conversion and Fanout?

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.



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.

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.

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 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

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.

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

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 cloudSwXtch environment. 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	42000	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:

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

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

- swtch.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, swtch.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](#)
- swtch.io does not require any public resources for deployment such as Amazon S3 buckets.
- swtch.io cloudSwXtch installation does not use any AWS Secrets in Secret Manager as swtch.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.
- swtch.io does not encrypt data. It pass through any data sent in the multicast which may be encrypted.

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.

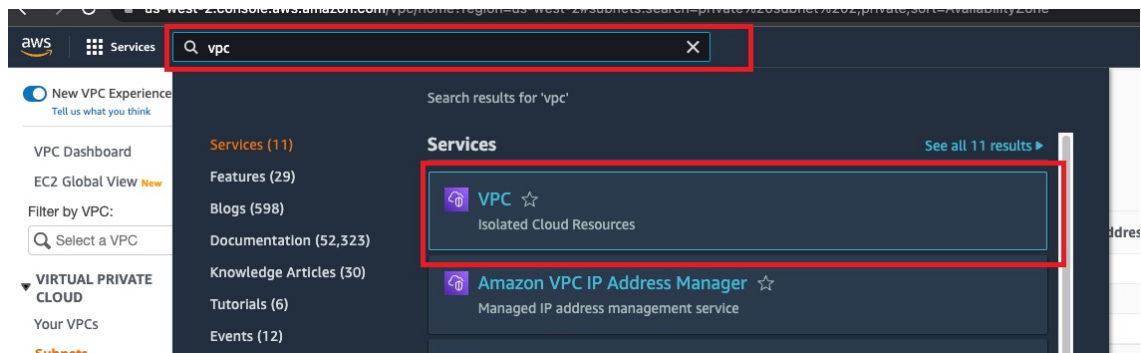
In this section, users will learn how to validate whether a subnet exists to be used as both the control and the data plane for their virtual network. This is in preparation for cloudSwXtch installation on AWS. We will also walk through an alternative method of using 2 subnets, separating the control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

Method #1: Single-subnet

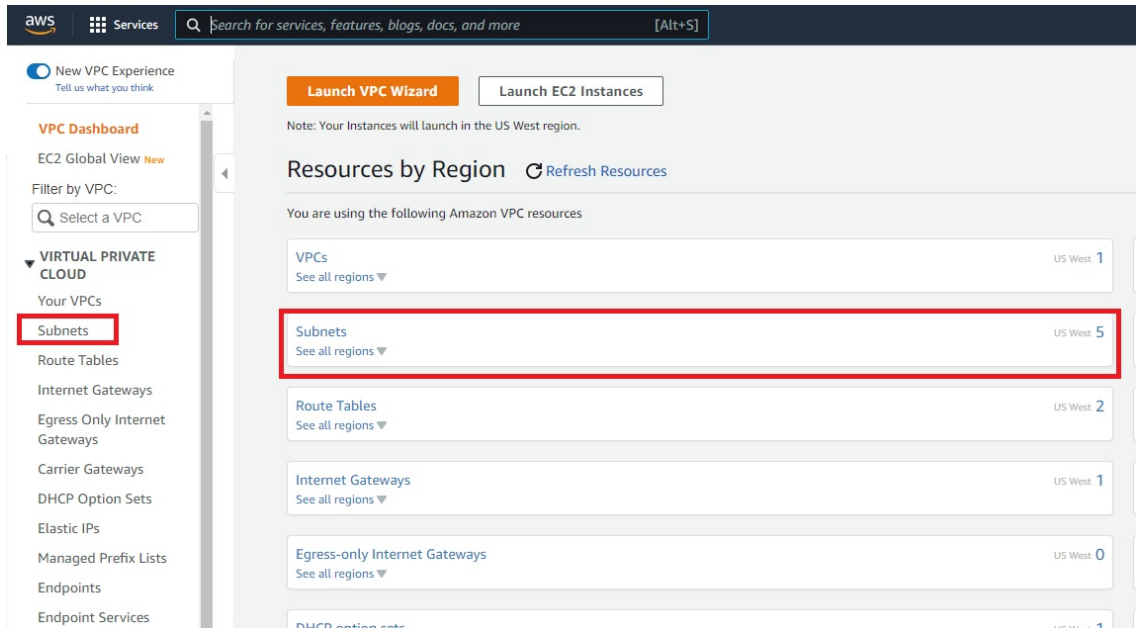
Typically, when deploying a VPC, a user will automatically create a subnet. During the main installation process, this subnet can be used for both control and data plane communications. This is the **preferred** method and will be used by a majority of users. Before installing cloudSwXtch, users should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



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.

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

Naming your subnets

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

The screenshot shows the AWS VPC console with the 'Subnets (2)' page. The left sidebar shows 'Subnets' selected under 'Virtual private cloud'. The main area displays a table of subnets with the following data:

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6...	Avail...	Availability Zone
ctrl_subnet	subnet-026043d5e098216ef	Available	vpc-032478a302937fd9e SA-Test-V...	172.31.16.0/20	-	4073	us-west-2b
data_subnet	subnet-04c26b015009b4c3f	Available	vpc-032478a302937fd9e SA-Test-V...	172.31.132.0/22	-	1005	us-west-2b

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

The screenshot shows the AWS VPC console with the 'Subnets (1/2)' page. The 'Create subnet' button in the top right corner is highlighted with a red box. The table of subnets shows the 'ctrl_subnet' selected with a checkbox.

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6...	Avail...	Availability Zone
ctrl_subnet	subnet-026043d5e098216ef	Available	vpc-032478a302937fd9e SA-Test-V...	172.31.16.0/20	-	4073	us-west-2b
data_subnet	subnet-04c26b015009b4c3f	Available	vpc-032478a302937fd9e SA-Test-V...	172.31.132.0/22	-	1005	us-west-2b

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.

The screenshot shows the AWS 'Create subnet' console. The breadcrumb navigation at the top reads 'VPC > Subnets > Create subnet'. The main heading is 'Create subnet' with an 'Info' link. The form is divided into two main sections: 'VPC' and 'Subnet settings'.

VPC Section:

- VPC ID:** A dropdown menu showing 'vpc-032478a302937fd9e'. This field is highlighted with a red box.
- Associated VPC CIDRs:** A list showing 'IPv4 CIDRs' with the value '172.31.0.0/16'.

Subnet settings Section:

Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

- Subnet name:** A text input field containing 'data_subnet'. This field is highlighted with a red box.
- Availability Zone:** A dropdown menu showing 'US West (Oregon) / us-west-2b'. This field is highlighted with a red box.
- IPv4 CIDR block:** A text input field containing '.31.133.0/22'. This field is highlighted with a red box.
- Tags - optional:** A section for adding tags. It includes a 'Key' field with 'Name' and a 'Value - optional' field with 'data_subnet'. There is a 'Remove' button and an 'Add new tag' button. Below this, there is a 'Remove' button and an 'Add new subnet' button.

At the bottom right of the form, there are two buttons: 'Cancel' and 'Create subnet'. The 'Create subnet' button is highlighted with a red box.

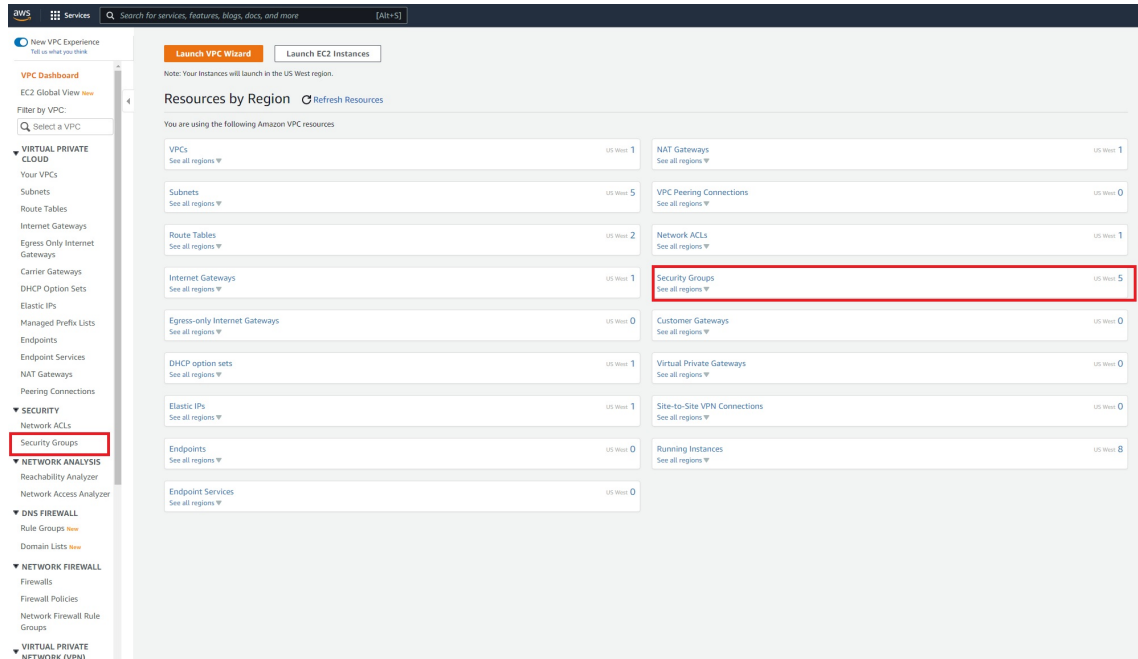
6. Click "Create Subnet." You should now have a new subnet on your list.

Verify Security Groups

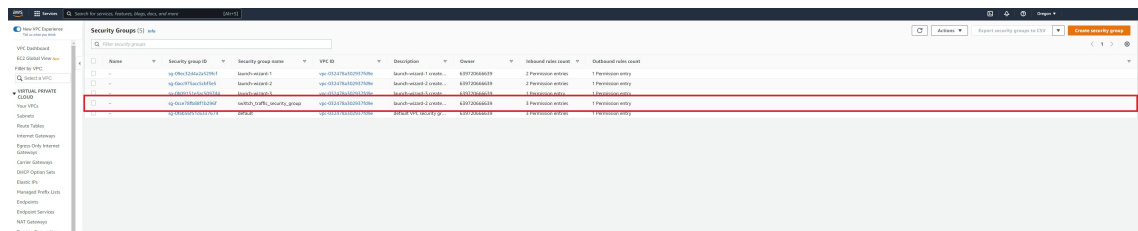
The security group contains the firewall settings for EC2 instances and interfaces (xNICs).

To ensure security groups are set up properly for cloudSwXtch:

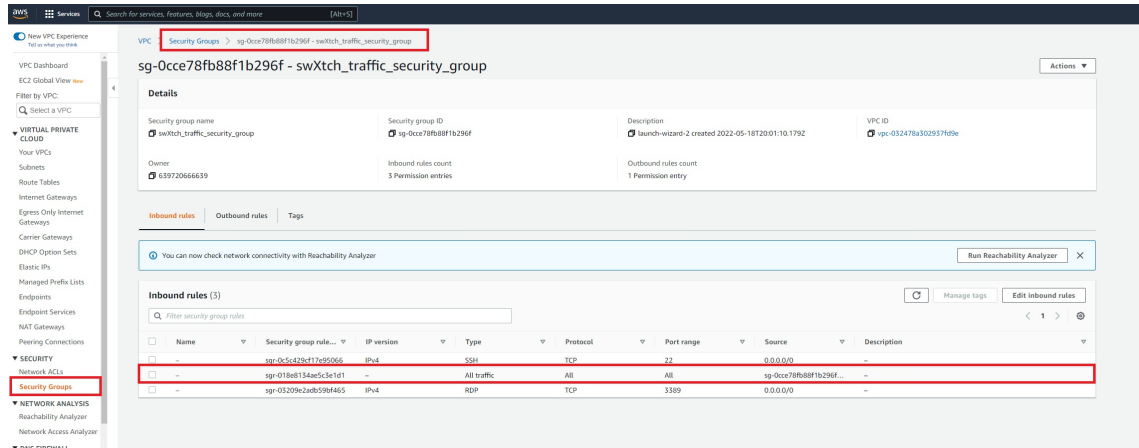
1. **Navigate** to the VPC console.
2. **Select** the "Security Groups" link as shown below. (Note: There are multiple ways to get to the "Security Groups" page.)



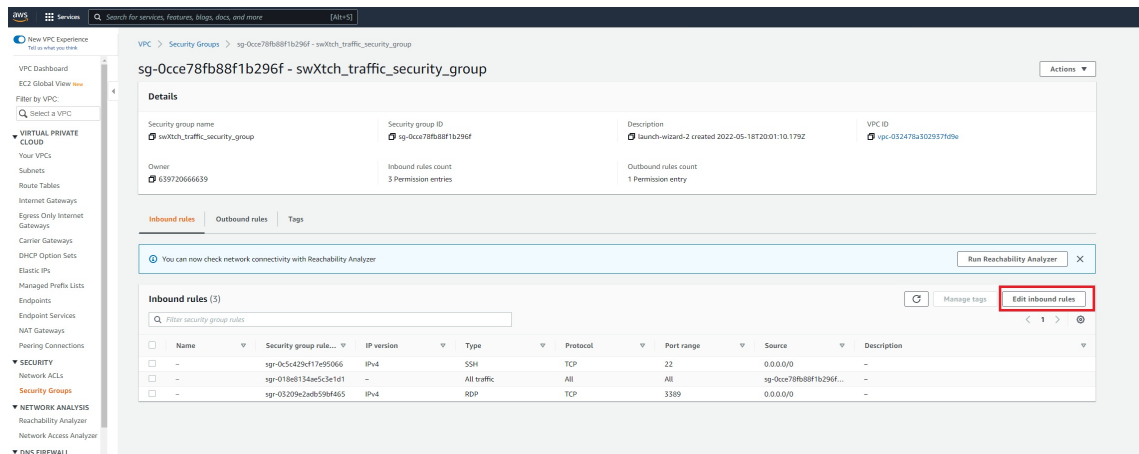
3. **Select** the Security Group that is normally used to create your EC2 instances for your application. (Note: The names in the example will be different in your environment.)



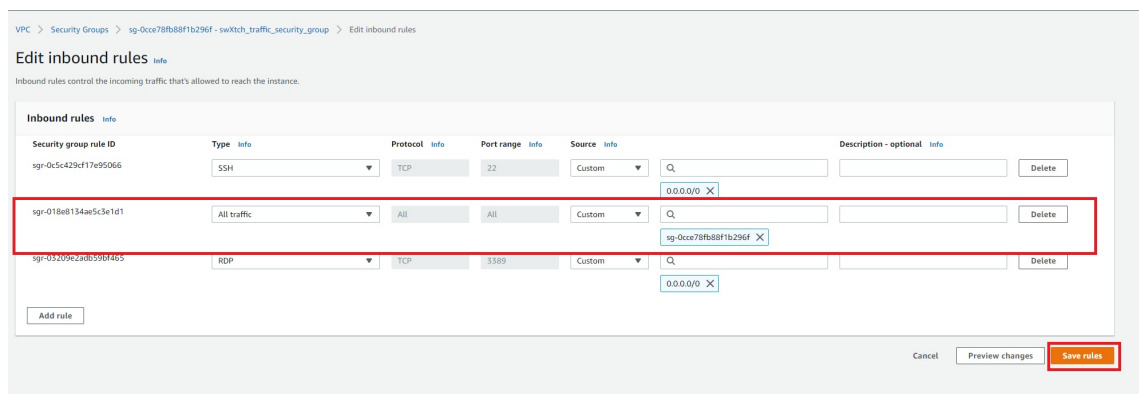
4. In order for certain features to work in your cloudSwXtch, you will need to add inbound rules to open specific ports originating from that security group. You can find the ports outlined in the [cloudSwXtch System Requirements](#) article under "Firewall and Security Group Rules."



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

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

EC2 > Security Groups > sg-0049066c880dc9b0a - SWLJ2L00A60-CreatedSwitchSecurityGroup-1E57NW0Z1UDWZ > Edit inbound rules

Edit inbound rules [info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type info	Protocol info	Port range info	Source info	Description - optional info	
sgr-0721bb274d152e765	All traffic	All	All	Custom	sg-0049066c880dc9b0a	Delete
sgr-045e4b7fda4d2a42e	SSH	TCP	22	Custom	0.0.0.0/0	Delete
-	Custom UDP	UDP	0	Anywhere-... Custom Anywhere-IPv4 Anywhere-IPv6 My IP	0.0.0.0/0	Delete

Create SSH Key Pair

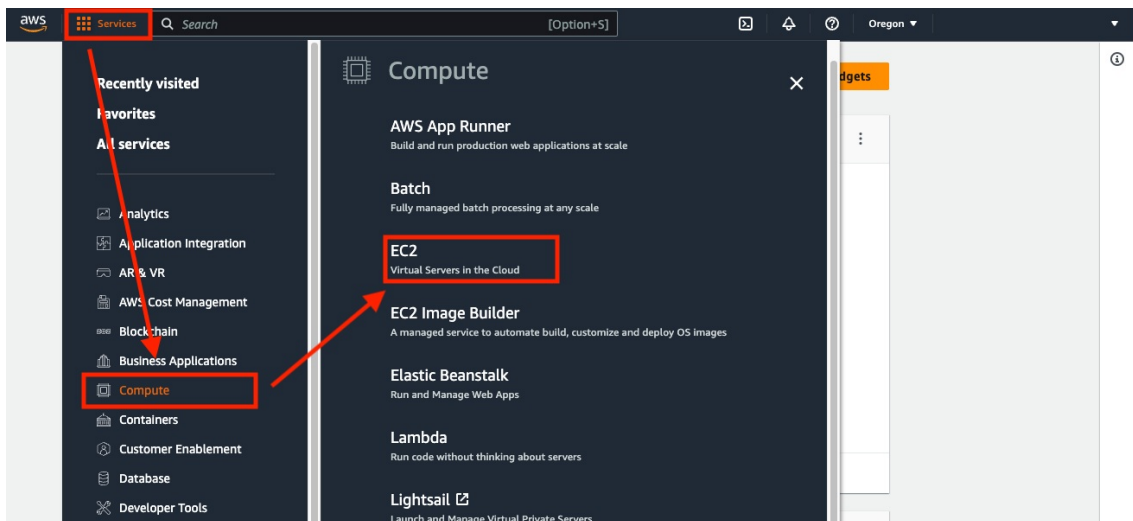
WHAT TO EXPECT

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.

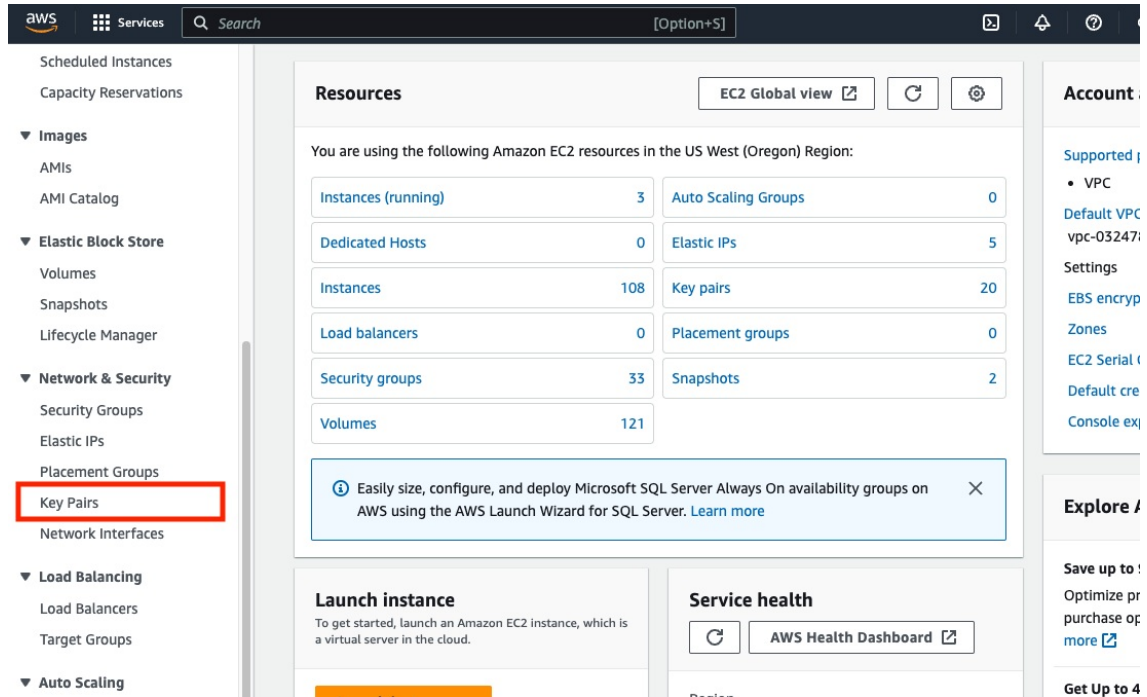
In the *AWS Management Console*, make sure you are in the region where you plan to use the cloudSwXtch instance.

1. Navigate to EC2

- Select the “Services” menu in the AWS Management Console.
- Click “Compute”
- Select “EC2”



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).

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

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).

The screenshot shows the 'Create key pair' page in the AWS Management Console. At the top, there is a breadcrumb trail: 'EC2 > Key pairs > Create key pair'. Below this is the main heading 'Create key pair' with an 'Info' link. The page is divided into several sections. The first section is titled 'Key pair' and contains a description: 'A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.' The second section is 'Name', which has a text input field with the placeholder 'Enter key pair name'. Below the input field is a note: 'The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.' The third section is 'Key pair type', which has an 'Info' link and two radio button options: 'RSA' (selected) and 'ED25519'. The fourth section is 'Private key file format', which has two radio button options: '.pem' (selected) and '.ppk'. Below the '.pem' option is the text 'For use with OpenSSH', and below the '.ppk' option is 'For use with PuTTY'. The fifth section is 'Tags - optional', which states 'No tags associated with the resource.' and includes an 'Add new tag' button. Below the button is the text 'You can add up to 50 more tags.' At the bottom right of the page are two buttons: 'Cancel' and 'Create key pair'.

EC2 > Key pairs > Create key pair

Create key pair [Info](#)

Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type [Info](#)

☒ RSA

☐ ED25519

Private key file format

☒ .pem
For use with OpenSSH

☐ .ppk
For use with PuTTY

Tags - optional

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#) [Create key pair](#)

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

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](#).

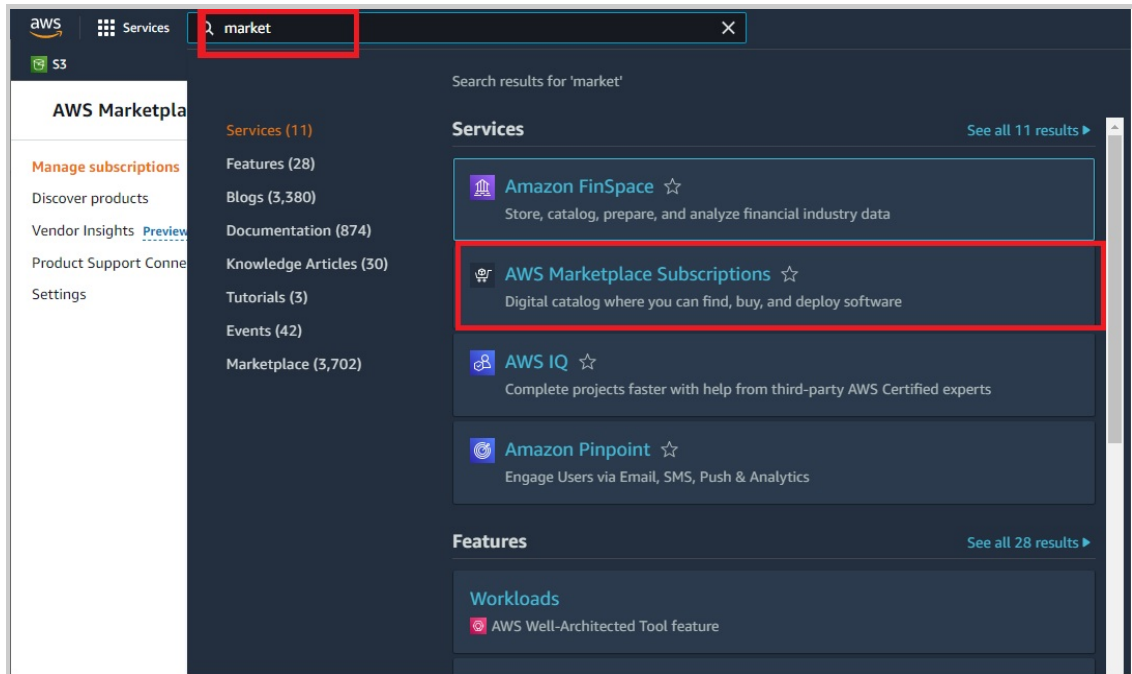
If all prerequisites are met, a cloudSwXtch can be created via the Marketplace in any region in approximately 10 minutes. If multi-AZ or multi-region is required then see [Mesh](#) for details. The installer will create a CloudFormation Stack to include the following resources:

- ControlEni Networking Interface for control data
- DataENI Networking Interface for data such as Multicast

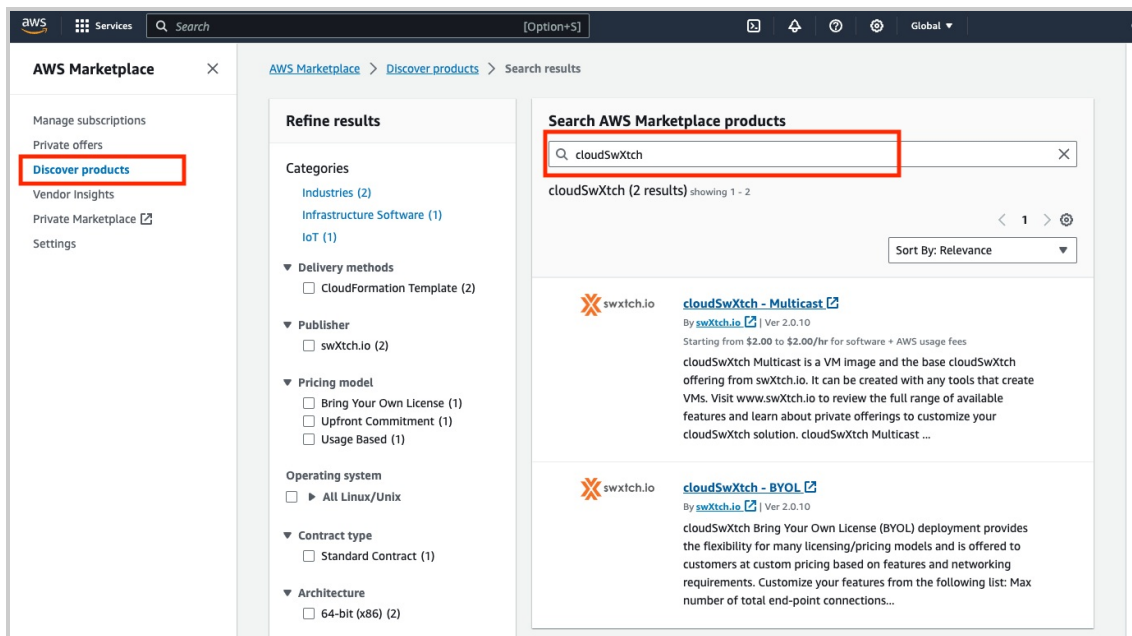
- EC2Instance in Linux for the cloudSwXtch to run on

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

1. Sign into AWS.
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."



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

1. Please read the [cloudSwXtch System Requirements](#) article for more information regarding cloudSwXtch sizing.
2. 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.

The screenshot shows the AWS Marketplace product page for cloudSwXtch - Multicast. The page includes a search bar, navigation links, and a product overview section. The 'Continue to Subscribe' button is highlighted with a red box. The pricing section shows a typical total price of \$2.991/hr. The product overview section includes a description of the product and a list of highlights.

cloudSwXtch - Multicast
By: [swxtch.io](#) Latest Version: 2.0.10
cloudSwXtch Multicast is an easy-to-deploy virtual overlay network that adds multicast and broadcast capabilities to cloud applications. Good for applications that require high-
[Show more](#)
Linux/Unix

Continue to Subscribe
Save to List

Typical Total Price
\$2.991/hr
Total pricing per instance for services hosted on m5zn.3xlarge in US East (N. Virginia). [View Details](#)

Product Overview
cloudSwXtch Multicast is a VM image and the base cloudSwXtch offering from swxtch.io. It can be created with any tools that create VMs. Visit [www.swxtch.io](#) to review the full range of available features and learn about private offerings to customize your cloudSwXtch solution.
cloudSwXtch Multicast includes the following features:

- Up to 100Mb ingress bandwidth
- Up to 10 total end-point connections (consumer/producer VMs).
- Multicast & Broadcast
- Simple "buy & go" deployment with no code changes required.

Monitoring - wXcked Eye is a monitoring UI that enables users to audit the performance of their cloudSwXtch network. A REST API is provided to manage and control your network your way.
Please visit [www.swxtch.io/pricing](#) for more information.

Highlights

- Plug and Play - Existing applications and services that expect standards-based IP multicast and broadcast will work in Azure without requiring any code changes.
- Simplified Workflows - cloudSwXtch allows your endpoints to dynamically subscribe and unsubscribe to your streams as needed. Even if those flows are coming in from off cloud!
- wXcked Fast - Moving high-performance systems to the cloud can be difficult. cloudSwXtch enables massive bandwidth with sub-millisecond latency for most applications.

7. Review the Terms and Conditions.

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

The screenshot shows the AWS Marketplace interface for the product 'cloudSwXtch - Small Tier' by swxtch.io. The page includes a navigation bar at the top with links like 'About', 'Categories', 'Delivery Methods', 'Solutions', 'AWS IQ', 'Resources', and 'Your Saved List'. The main content area has a 'Subscribe to this software' section with a paragraph explaining the subscription process. Below this is a 'Terms and Conditions' section with a 'swxtch.io Offer' box. Inside this box, there is a paragraph of legal text and a red-bordered button labeled 'Accept Terms'. At the bottom of the page, there is a note about pricing information.

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

The screenshot shows the AWS Marketplace interface for the product 'cloudSwXtch - Multicast' by swxtch.io. The page includes a navigation bar at the top. The main content area has a 'Subscribe to this software' section. Below this is a 'Save money by purchasing a software contract' section with a 'Configure contract' link. Further down is a 'License management' section with a 'Manage License' link. At the bottom is a 'Terms and Conditions' section with a 'swxtch.io Offer' box. To the right of the main content, there is a 'Software contract' sidebar showing the 'Total contract price' as '\$0 Due now' and a 'Create Contract' button. A red-bordered button labeled 'Continue to Configuration' is located at the top right of the main content area.

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.)

The screenshot shows the AWS Marketplace interface for configuring the 'cloudSwXtch - Multicast' software. The top navigation bar includes the AWS Marketplace logo, a search bar, and various links like 'About', 'Categories', 'Delivery Methods', 'Solutions', 'AWS IQ', 'Resources', 'Your Saved List', 'Become a Channel Partner', 'Sell in AWS Marketplace', 'Amazon Web Services Home', and 'Help'. The main header displays the 'swxtch.io' logo and the product name 'cloudSwXtch - Multicast'. A red box highlights the 'Continue to Launch' button in the top right corner. Below the header, the 'Configure this software' section is active, showing options for 'Fulfillment option' (Swxtch Configuration), 'Software version' (2.0.10 (Sep 04, 2023)), and 'Region' (US East (N. Virginia)). A red box highlights the 'Region' dropdown menu. To the right, there are sections for 'Software contract' (Total contract price: \$0) and 'Pricing information' (Software Pricing: cloudSwXtch - Multicast, \$0.001/hr OR \$1.00/1095 days). The bottom of the page shows the 'Product Code: ejh6oiwhxalojuo4octwweeqz'.

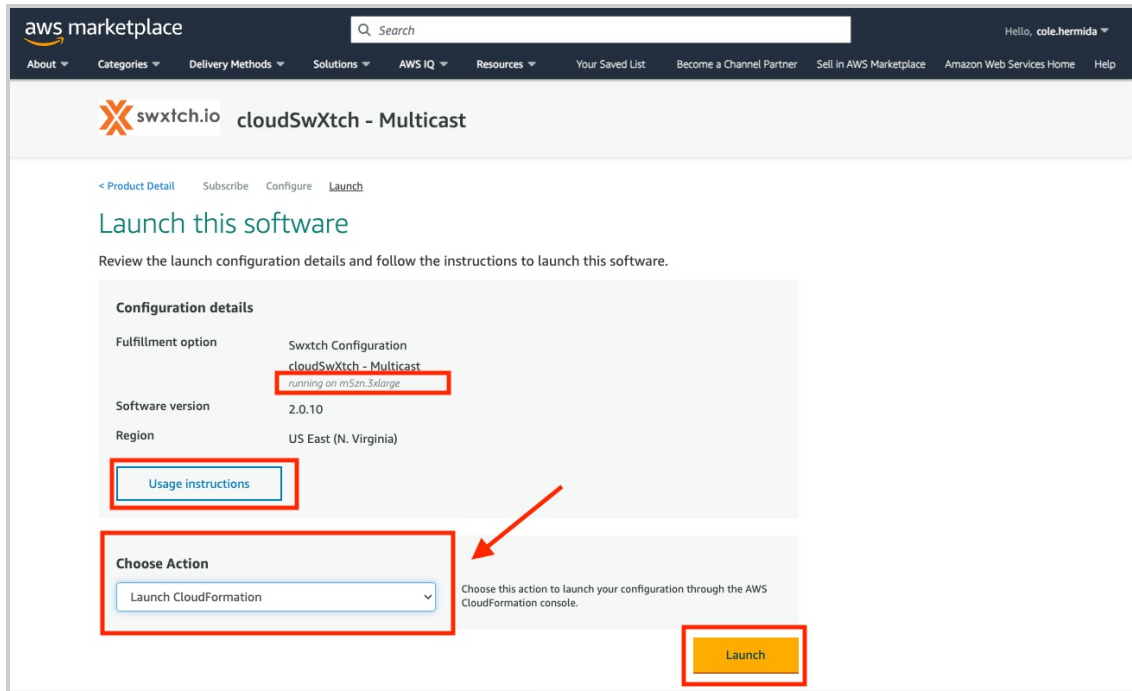
INSTANCE TYPES

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:

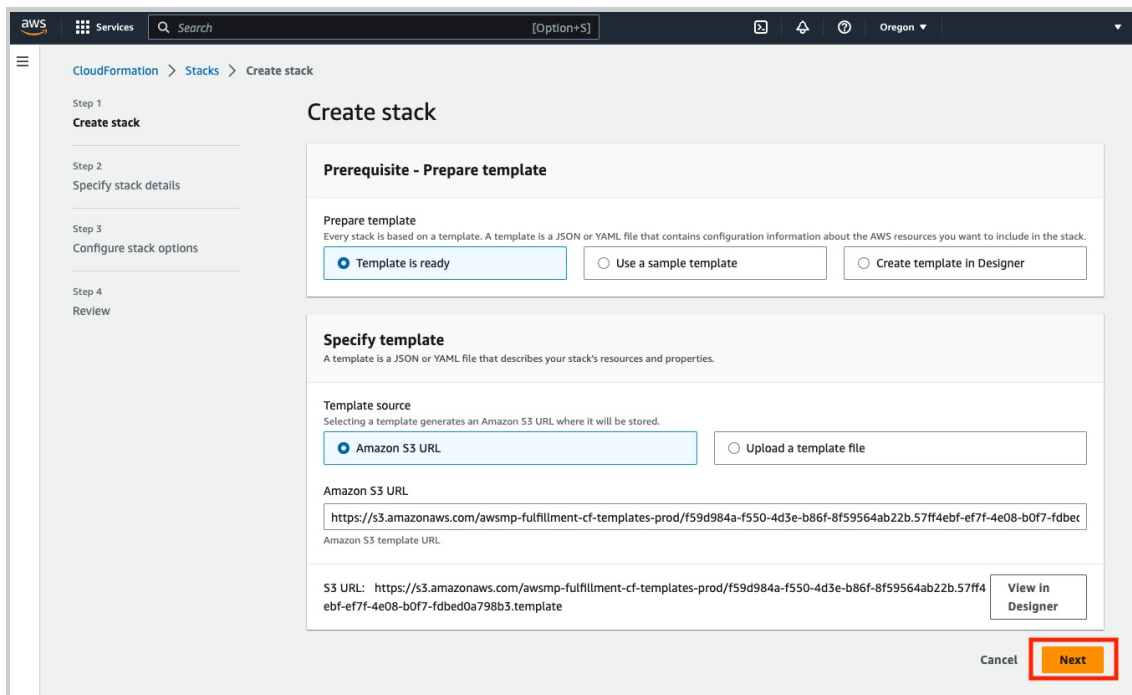
- m5.xlarge
- m5.2xlarge
- m5.4xlarge
- m5.12xlarge
- m5.16xlarge
- m5.24xlarge
- m5zn.xlarge
- m5zn.2xlarge
- m5zn.3xlarge
- m5zn.6xlarge
- m5zn.12xlarge

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:

1. 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.
2. 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.
3. Under "ControlSubnet," use the dropdown to find the control subnet you created (recommended: *ctrl-subnet*).

4. Under “DataSubnet,” **use** the dropdown to find the control subnet you created. Both control and data can share the same subnet.
 1. Alternatively, for better performance, a user can assign a separate subnet for their data subnet.
5. For “InstanceType,” there should be "Fulfillment" data from the earlier step.
6. Under “KeyName,” **use** the dropdown to find your previously created or imported SSH key.
7. In “PassedSwxtchSecurityGroup,” **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."
8. Under “VpcId,” **select** from the dropdown to find the already created VPC id.
9. **Here is an example of how your template would look like:**

Specify stack details

Stack name

Stack name

Test-Swxtch-01

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

CidripForSSHAccess

For SSH access when using default security group, please set CIDR to x.x.x.x/32 to allow one specific IP address access, 0.0.0.0/0 to allow all IP addresses access, or another CIDR range

0.0.0.0/0

ControlSubnet

Used to create control ENI assigned to the swxtch control subnet.

subnet-0f956735b224e2178

DataSubnet

Used to create control ENI assigned to the swxtch control subnet.

subnet-07de773cb1faecc38

InstanceType

EC2 instance type to use for swXtch creation

m5zn.3xlarge

KeyName

SSH key name for swXtch SSH access

PassedSwxtchSecurityGroup

Name of the security group that should be used by swXtch and created ENIs

sg-0702ad5e67a99c4b8

VpcId

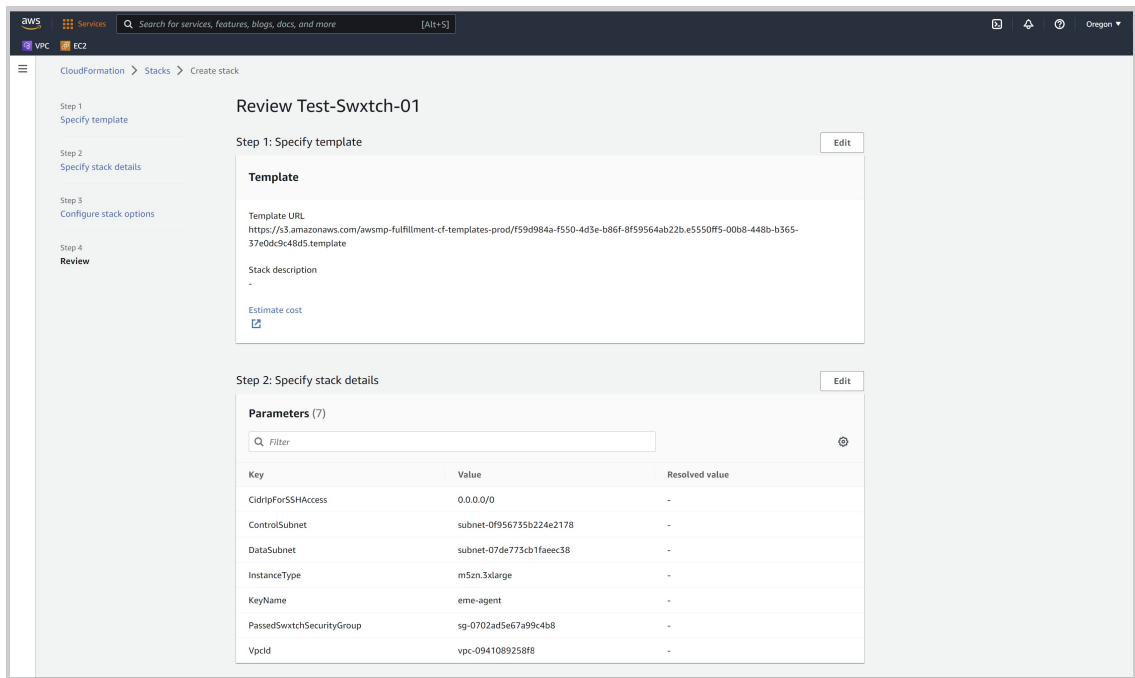
The VPC Id your swXtch will be placed in

vpc-0941089258f89c9a2

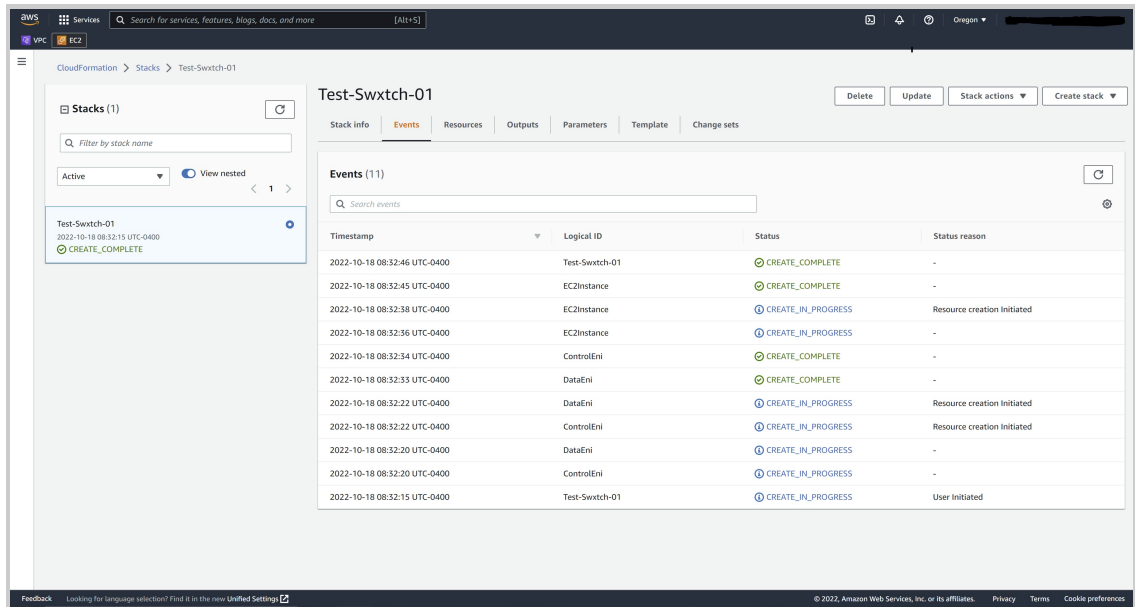
Cancel
Previous
Next

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.

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>
```

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](#).

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 instructions 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
<pre>\$ git clone git@github.com:swxtchio/cloudSwXtch-support</pre>	

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

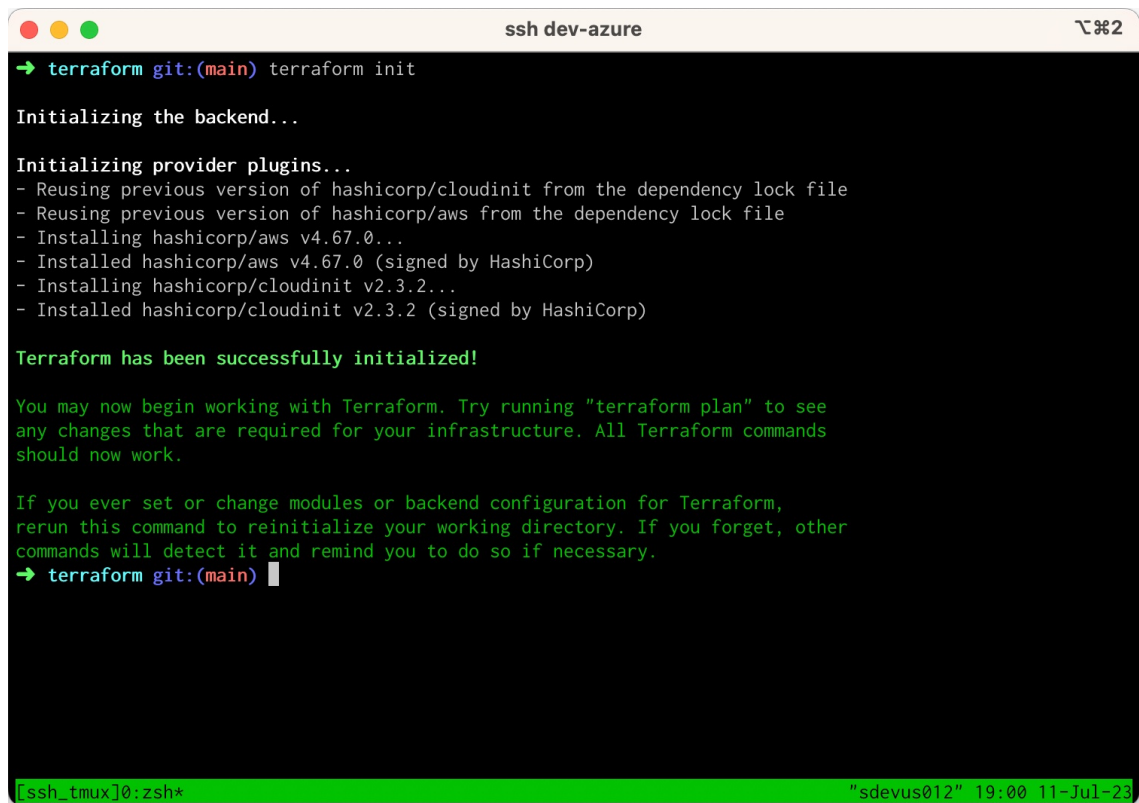
Console	Copy
<pre>\$ git clone https://github.com/swxtchio/cloudSwXtch-support.git</pre>	

Then:

Console	Copy
<pre>\$ cd cloudSwXtch-support/AWS/terraform</pre>	

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.

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

A terminal window titled "ssh dev-azure" with a window icon in the top-left corner. The terminal shows the command "terraform init" being executed. The output includes "Initializing the backend...", "Initializing provider plugins...", a list of actions (reusing previous versions, installing hashicorp/aws v4.67.0, and hashicorp/cloudinit v2.3.2), and a green message "Terraform has been successfully initialized!". It also provides instructions on how to use Terraform and how to reinitialize if configurations change. The prompt "terraform git:(main)" is shown at the bottom of the terminal content. The terminal's status bar at the bottom is green and shows "[ssh_tmux]0:zsh*" on the left and "\"sdevus012\" 19:00 11-Jul-23" on the right.

```
ssh dev-azure 2

→ terraform git:(main) terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/cloudinit from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Installing hashicorp/aws v4.67.0...
- Installed hashicorp/aws v4.67.0 (signed by HashiCorp)
- Installing hashicorp/cloudinit v2.3.2...
- Installed hashicorp/cloudinit v2.3.2 (signed by HashiCorp)

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
→ terraform git:(main) █

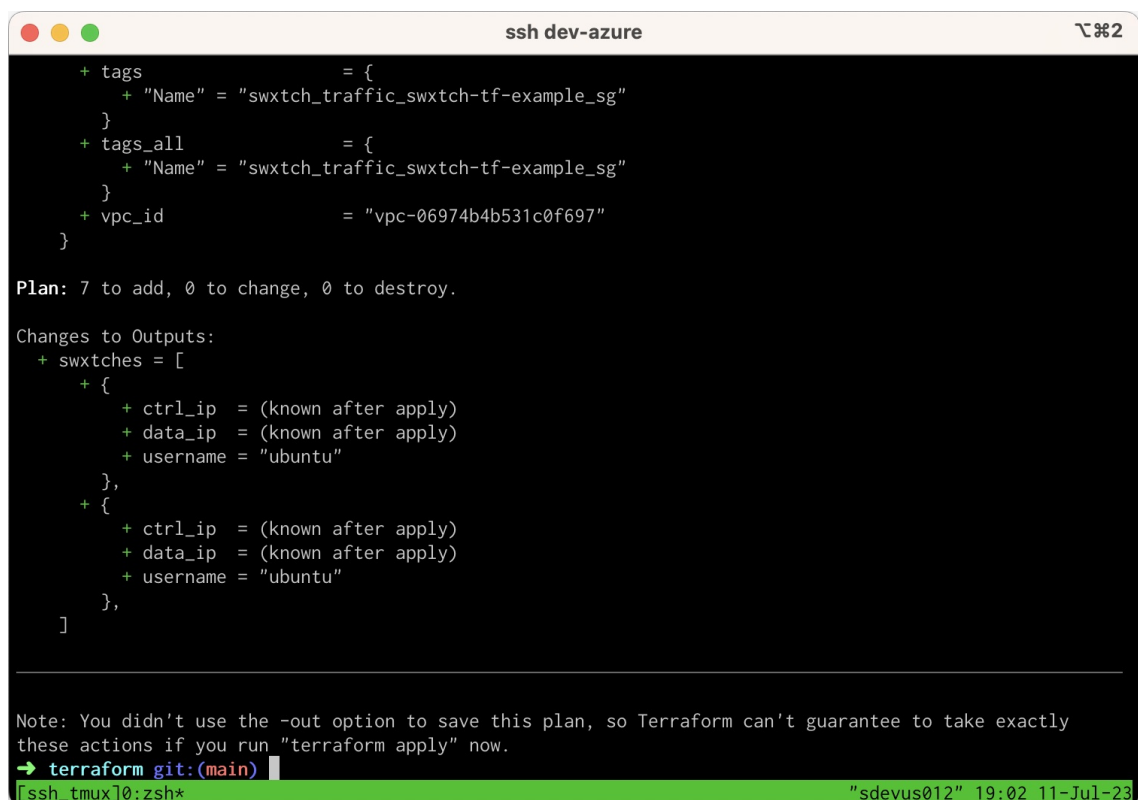
[ssh_tmux]0:zsh* "sdevus012" 19:00 11-Jul-23
```

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
<pre>\$ export AWS_ACCESS_KEY_ID="anaccesskey" \$ export AWS_SECRET_ACCESS_KEY="asecretkey" \$ export AWS_REGION="us-west-2"</pre>	

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
<pre>\$ terraform plan</pre>	



```
ssh dev-azure 2

+ tags              = {
+   "Name" = "swxtch_traffic_swxtch-tf-example_sg"
+ }
+ tags_all          = {
+   "Name" = "swxtch_traffic_swxtch-tf-example_sg"
+ }
+ vpc_id            = "vpc-06974b4b531c0f697"
+ }

Plan: 7 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ swxtches = [
+   {
+     ctrl_ip = (known after apply)
+     data_ip = (known after apply)
+     username = "ubuntu"
+   },
+   {
+     ctrl_ip = (known after apply)
+     data_ip = (known after apply)
+     username = "ubuntu"
+   },
+ ]

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.
→ terraform git:(main)
[ssh_tmux]0:zsh* "sdevus012" 19:02 11-Jul-23
```

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.

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

Console	Copy
<pre>terraform apply yes</pre>	

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

```
ssh dev-azure
aws_network_interface.swxtch_data[1]: Creating...
aws_network_interface.swxtch_data[0]: Creating...
aws_network_interface.swxtch_ctrl[1]: Creation complete after 1s [id=eni-099a6704b48871b65]
aws_network_interface.swxtch_data[0]: Creation complete after 1s [id=eni-05c53da54a04ea3be]
aws_network_interface.swxtch_data[1]: Creation complete after 1s [id=eni-0c364aa6898a7e2a2]
aws_network_interface.swxtch_ctrl[0]: Creation complete after 1s [id=eni-0c765fb2ad379495b]
aws_instance.swxtch[0]: Creating...
aws_instance.swxtch[1]: Creating...
aws_instance.swxtch[0]: Still creating... [10s elapsed]
aws_instance.swxtch[1]: Still creating... [10s elapsed]
aws_instance.swxtch[1]: Creation complete after 14s [id=i-06032053dbf5a744e]
aws_instance.swxtch[0]: Creation complete after 14s [id=i-08ba3f7cfa686764c]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

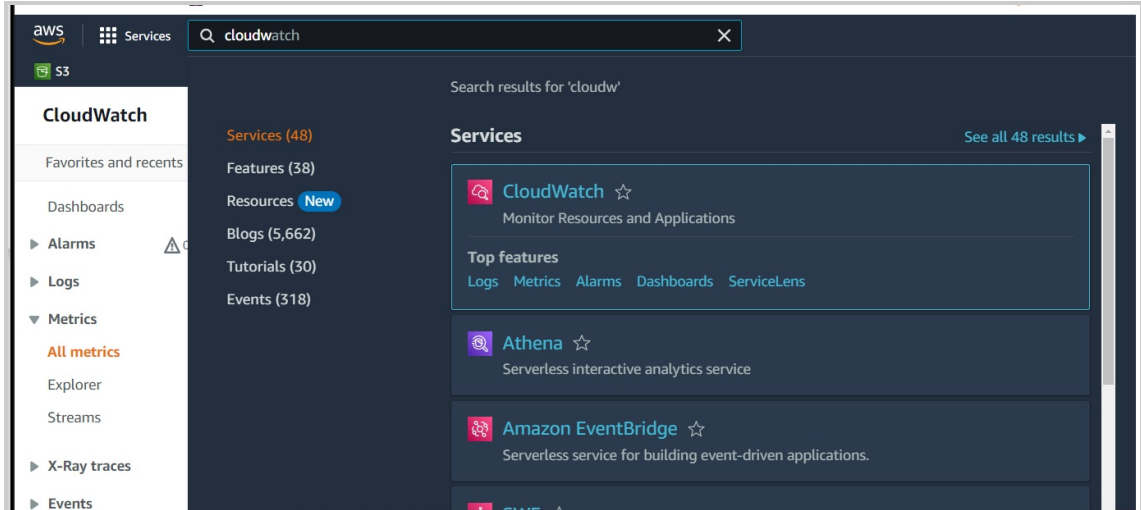
swxtches = [
  {
    "ctrl_ip" = "172.31.33.152"
    "data_ip" = "172.31.132.216"
    "username" = "ubuntu"
  },
  {
    "ctrl_ip" = "172.31.46.220"
    "data_ip" = "172.31.133.206"
    "username" = "ubuntu"
  },
]
→ terraform git:(main)
[ssh_tmux]0:terraform* "sdevus012" 19:05 11-Jul-23
```

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

Check Health of cloudSwXtch Instance on AWS

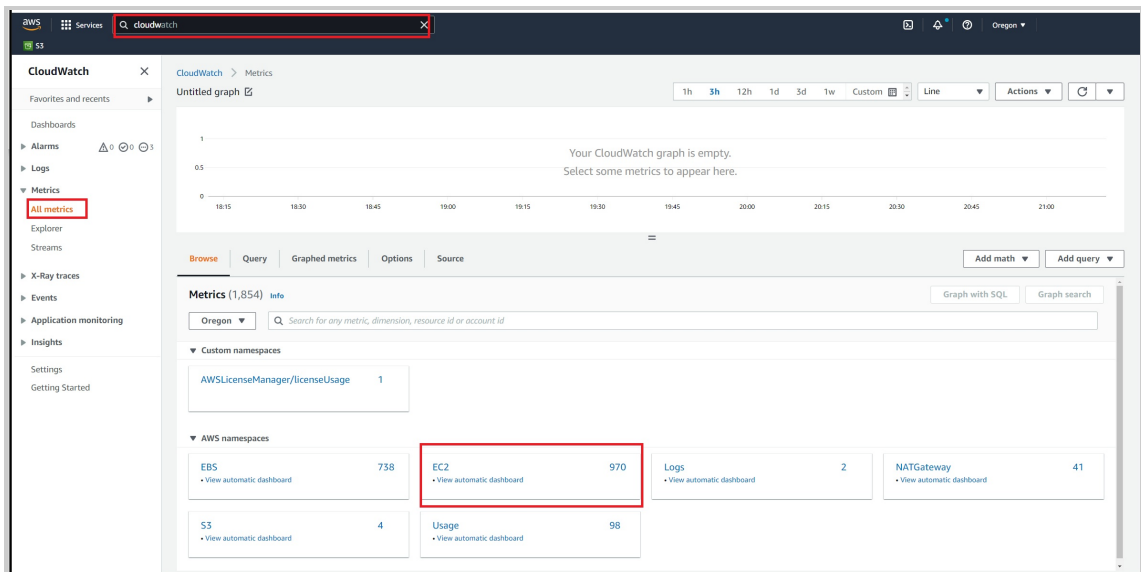
It is important to ensure your AWS system is healthy. AWS provides AWS CloudWatch as a way to check on the health of your system. To check on the cloudSwXtch EC2 instance:

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

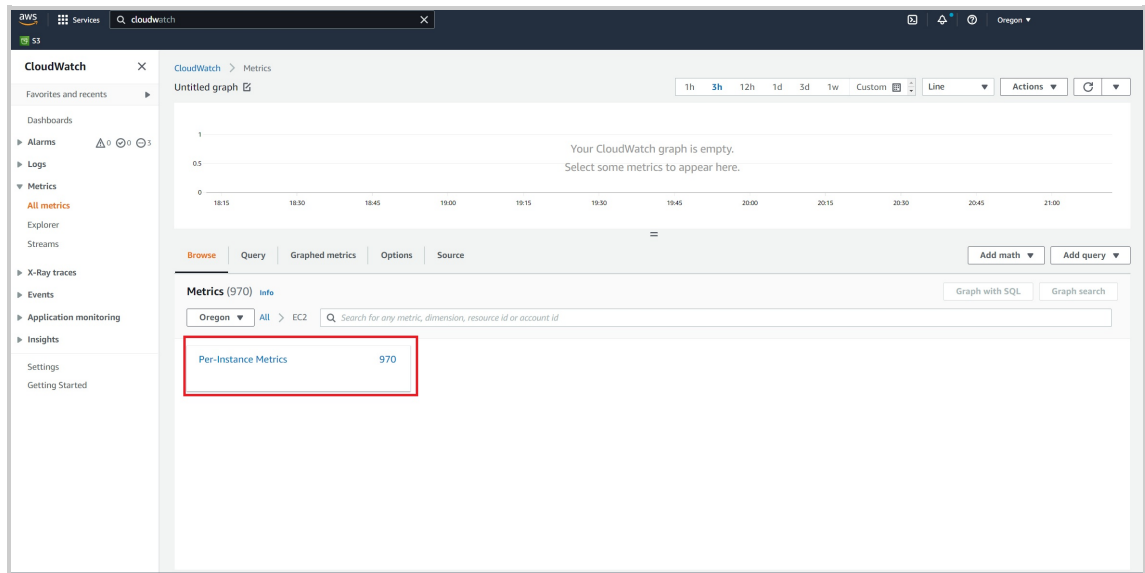


2. Select "All Metrics" on the left tree menu under "Metrics."

3. Select "EC2."

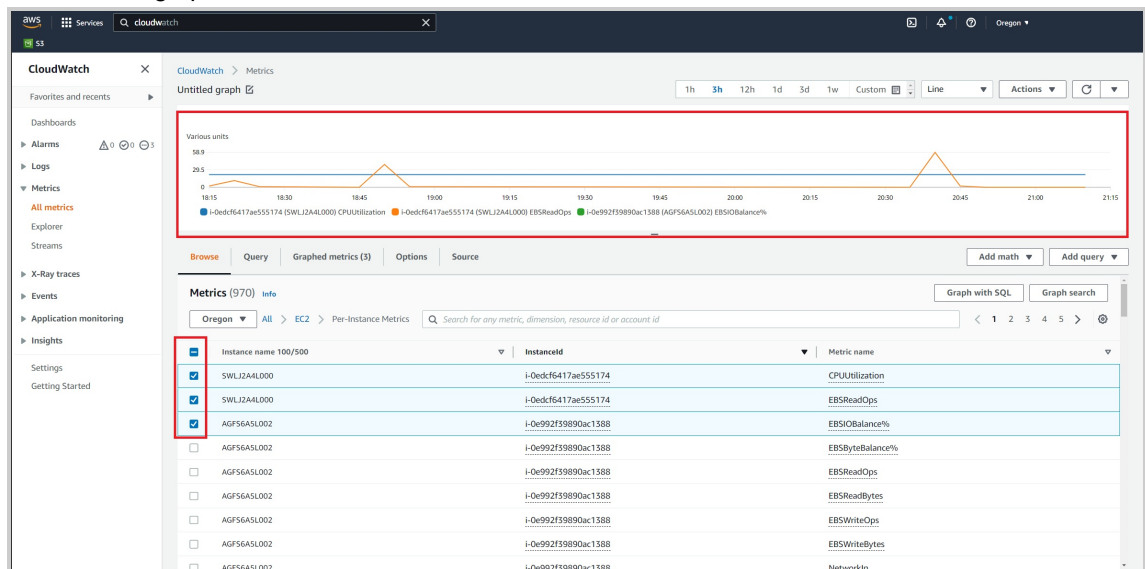


4. Select "Per-Instance Metrics."



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.

Delete cloudSwXtch on AWS

WHAT TO EXPECT

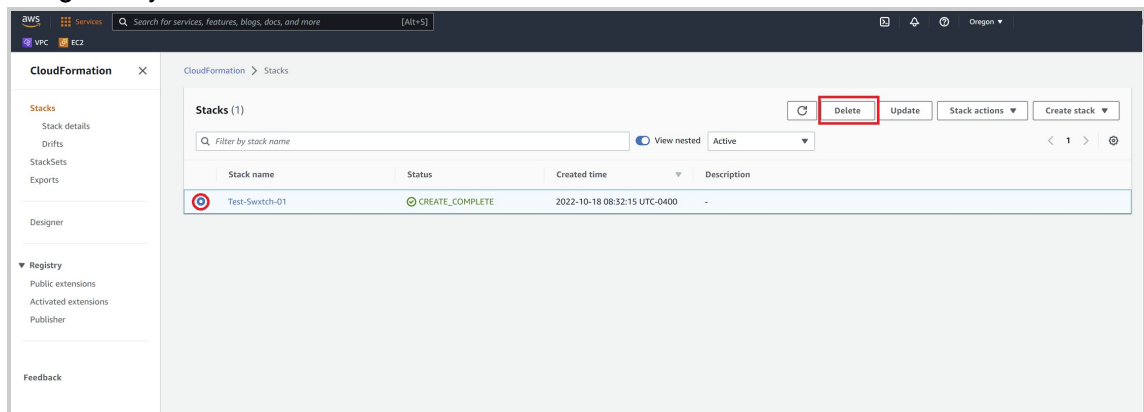
In this section, users will learn how to delete cloudSwXtch from their AWS environment.

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.

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:

The screenshot shows the Azure Marketplace page for 'cloudSwXtch VM Image' by swxtch.io LLC. The page includes a search bar, navigation links, and a 'Create' button. A red box highlights an 'IMPORTANT' note that states: 'Use the provided templates in the link below to **easily** deploy a cloudSwXtch. These templates ensure correct setup of the VM and network interfaces. If you deploy using the Azure Portal Web UI the resulting VM may not function properly. Templates for deployment: [Templates](#) For more installation instructions see: [docs.swxtch.io](#)'. A red arrow points from the text 'The template method is also mentioned in the Market Place cloudSwXtch installation as highlighted below:' to this box.

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

Microsoft Azure (Preview)

Search resources, services, and docs (G+/)

Home > Marketplace >

cloudSwXtch VM Image

swtch.io LLC

cloudSwXtch VM Image

swtch.io LLC | Virtual Machine

Private plan

Azure benefit eligible

Plan

BYOL

Create

Start with a pre-set configuration

Want to deploy programmatically? [Get started](#)

Overview

Plans + Pricing

Usage Information + Support

Ratings + Reviews

cloudSwXtch from swtch.io is a virtual overlay network that adds feature rich, high performance networking to your cloud or edge applications. cloudSwXtch is simple to deploy and **requires no code changes** to existing applicaitons.

The cloudSwXtch is a network appliance that is deployed as a **VM image** so can be created with any tools that create VMs. However, to function properly the cloudSwXtch needs *two network interfaces* and the second interface needs to have Accelerated Networking enabled.

IMPORTANT

Use the provided templates in the link below to **easily** deploy a cloudSwXtch. These templates ensure correct setup of the VM and network interfaces.

If you deploy using the Azure Portal Web UI the resulting VM may not function properly.

Templates for deployment: [Templates](#)

For more installation instructions see: docs.swtch.io

Selecting this link will open the page below:

Product

Solutions

Open Source

Pricing

Search or jump to...

7

Sign in

Sign up

swtch.io / swx-cloudSwXtch-AzureTemplates

Public

Notifications

Fork 1

Star 0

Code

Issues

Pull requests

Actions

Security

Insights

main

1 Branch

0 Tags

Go to file

Code

brentvates-iex

Updated plans to match latest SKUs: byol & payg

5039a67 · last week

10 Commits

LICENSE	Initial commit	2 years ago
MPTemplateUI.json	Updated plans to match latest SKUs: byol & payg	last week
MPTemplateVM.json	Renamed for clarity against non-marketplace templates	2 years ago
README.md	Update README.md (#2)	2 years ago

README

MIT license

cloudSwXtch-templates

Templates to create a cloudSwXtch from the marketplace. The cloudSwXtch is deployed as a VM image so can be created with any tools that create VMs. However, to function properly the cloudSwXtch **needs two network interfaces** and the second interface needs to have Accelerated Networking enabled. Use these templates to simplify deploying a cloudSwXtch since these templates ensure correct setup of the network interfaces.

Install template into the Azure Protal

About

Templates to create a cloudSwXtch from the marketplace

Readme

MIT license

Activity

Custom properties

0 stars

8 watching

1 fork

Report repository

Releases

No releases published

Packages

No packages published

Contributors 2

brentvates-iex

Brent Yates (swtch.io)

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io, Investors' Exchange LLC and IEX Services LLC. IEX Services LLC, Member SIPC/FINRA. All rights reserved.

54

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](#).

Validate Subnets on Azure

WHAT TO EXPECT

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

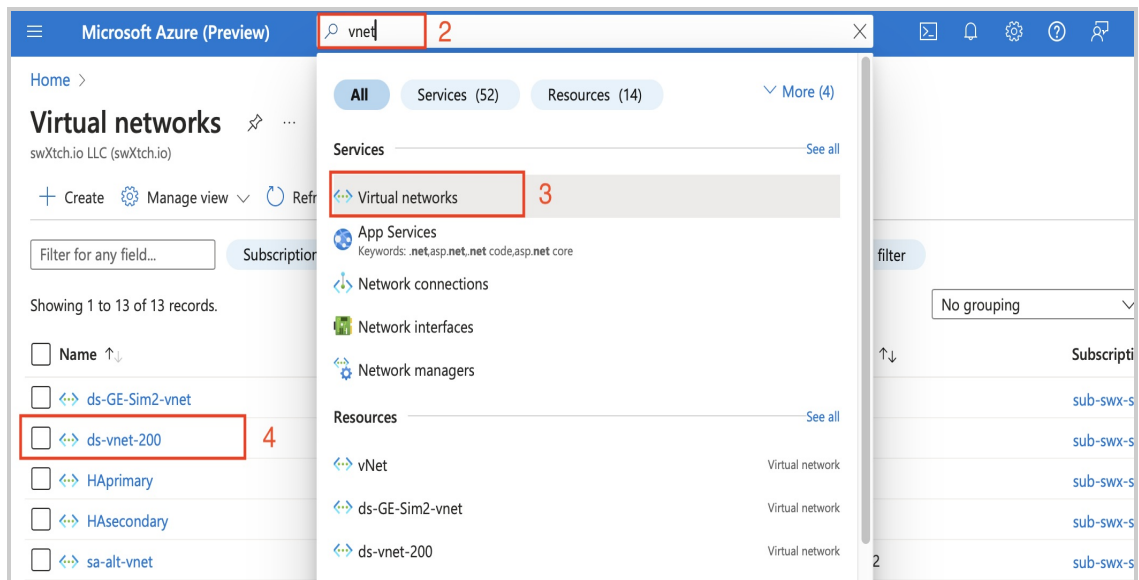
- It must contain at least one subnet that's used for both the control and data plane communication.
 - It is recommended that it is **private facing** and **does not auto-assign public IPs**.
 - This single subnet will be used for xNIC installation.

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.

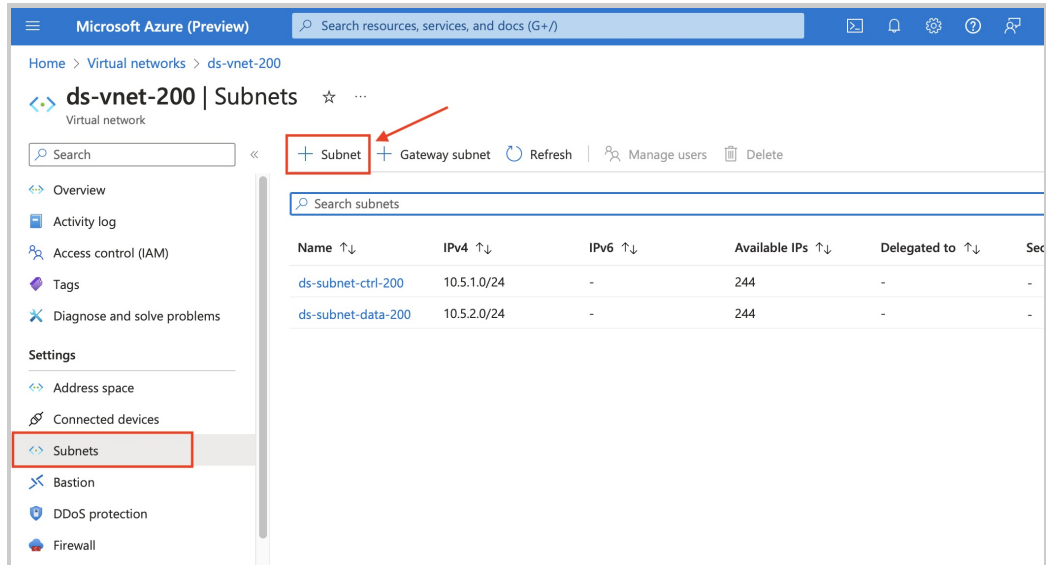
To validate:

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

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



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

sa-subnet-data

sa-vnet

Name: sa-subnet-data

Subnet address range: .2.192.0/22

Add IPv6 address space: ☐

NAT gateway: None

Network security group: None

Route table: None

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](#).

Create an Azure cloudSwXtch Template

WHAT TO EXPECT

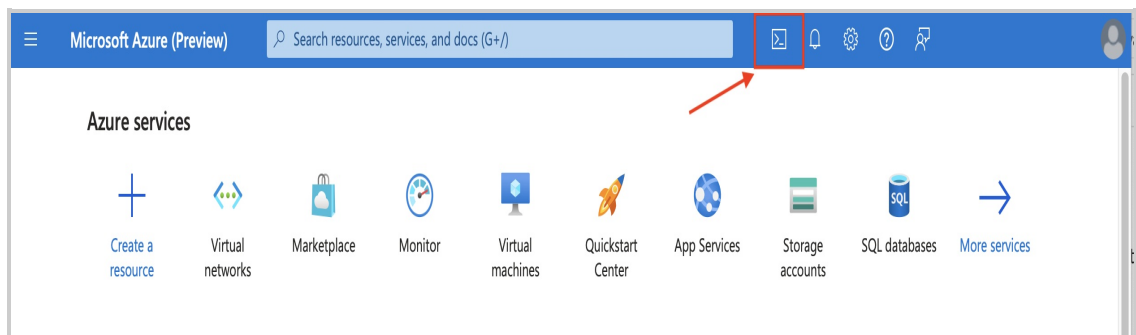
The easiest way to deploy a cloudSwXtch instance in your Azure environment is through the template method. The following process is a one-time task per subscription.

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.

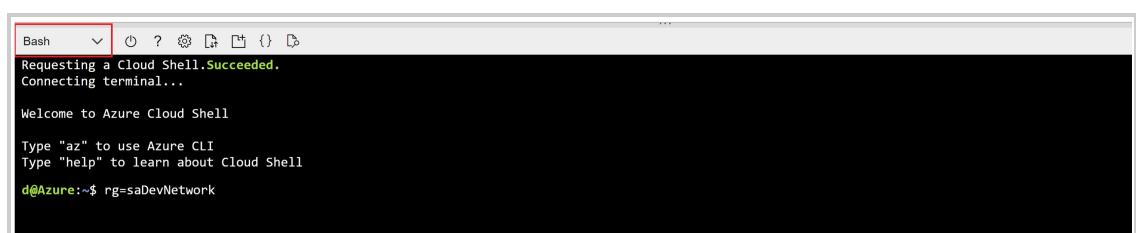


If you need help setting up your Azure cloud-shell, use the link below for setup instructions.
[azure cloud-shell quick start](#)

3. Make sure you are running your cloud shell terminal in Bash mode.
4. Enter in the following command to get to the proper resource group:

Bash	Copy
<pre>rg="<your-rg-here>"</pre>	

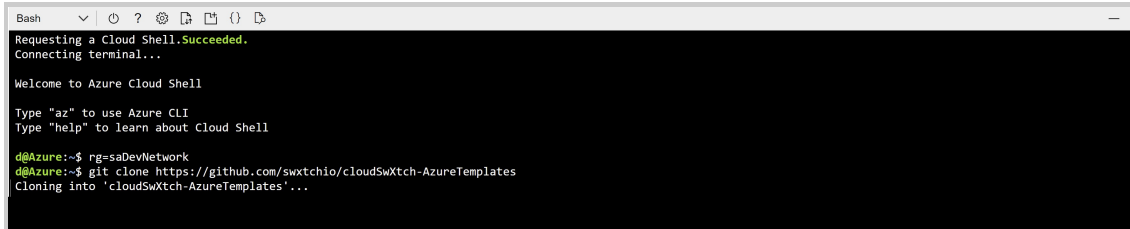
Example below:



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

Bash	Copy
<pre>git clone https://github.com/swxtchio/cloudSwXtch-AzureTemplates</pre>	

Example below:



```
Bash
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

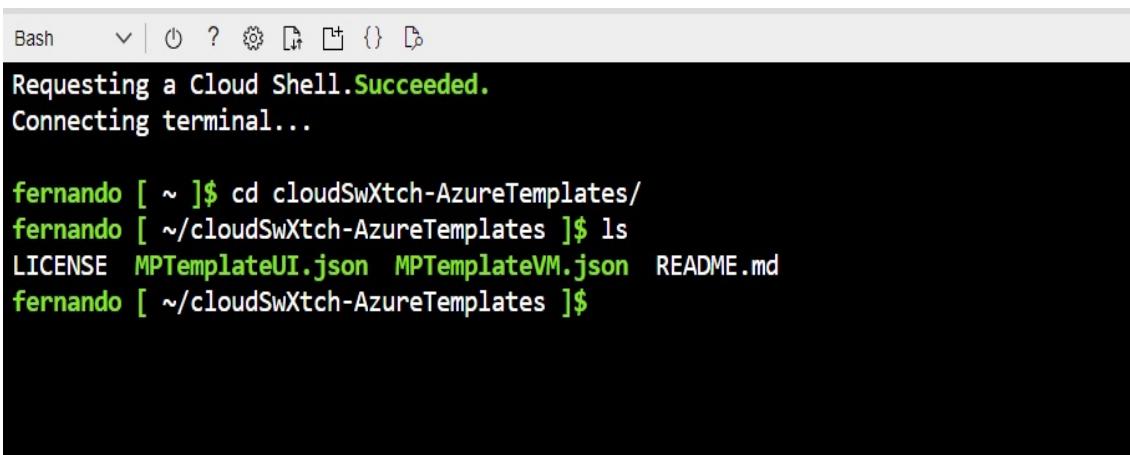
Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

d@Azure:~$ rg-saDevNetwork
d@Azure:~$ git clone https://github.com/swxtchio/cloudSwXtch-AzureTemplates
Cloning into 'cloudSwXtch-AzureTemplates'...
```

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

Bash	Copy
<pre>cd cloudSwXtch-AzureTemplates</pre>	

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



```
Bash
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

fernando [ ~ ]$ cd cloudSwXtch-AzureTemplates/
fernando [ ~/cloudSwXtch-AzureTemplates ]$ ls
LICENSE  MPTemplateUI.json  MPTemplateVM.json  README.md
fernando [ ~/cloudSwXtch-AzureTemplates ]$
```


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

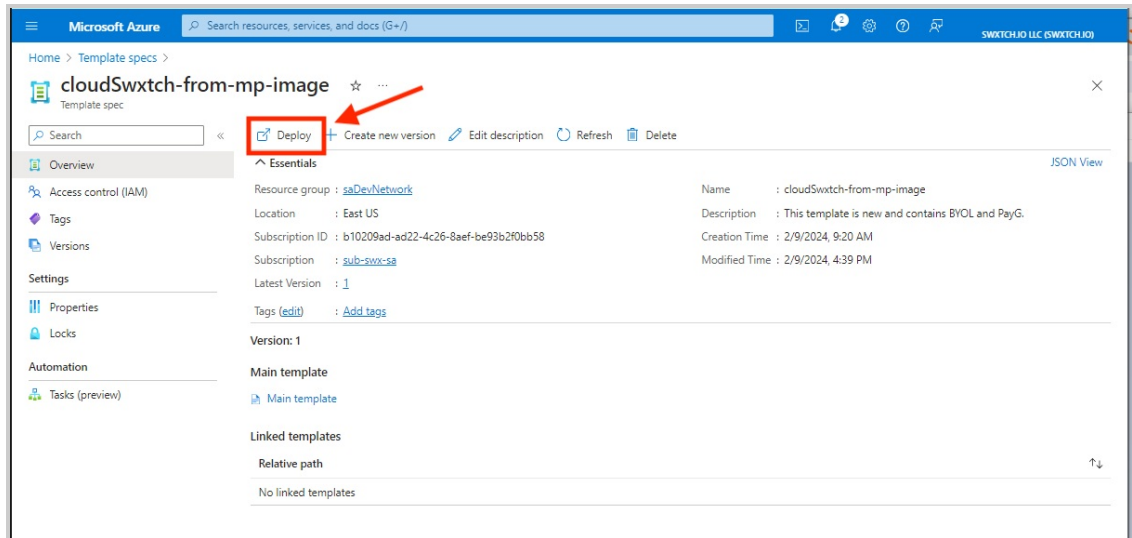
Prerequisites

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 allow 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
2. Find the template by using the "Search resource, services, and docs" bar (G+/) 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 (Multicast or PAYG). For more information on plans see: [cloudSwXtch System Requirements](#).

portal.azure.com/#create/Microsoft.Template

IEX Resources Hubspot Dashboard Articles and WebCa... Webinars IEX Stuff swXtch Lucid Atlassian Cloud Pro

Microsoft Azure Search resource

Home > Template specs > cloudSwxtch-from-mp-image >

cloudSwXtch instantiation template

Template spec

New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →

Basics Network Review + create

Template

cloudSwxtch-from-mp-image (1)
3 resources

cloudSwXtch commercial plan

cloudSwXtch plans are billed at hourly rates in addition to any Azure infrastructure (VM) fees.
[Plan details and pricing information can be found here:](#)

Choose a cloudSwXtch plan: * byol

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ sub-swX-sa

Resource group * ⓘ saDevNetwork
[Create new](#)

Instance details

Region * ⓘ (US) East US

Switch Name * ⓘ cloudswxtch001 ✓

Switch Size * ⓘ **1x Standard D8s v4**
8 vcpus, 32 GB memory
[Change size](#)

Admin name * ⓘ swxtchadmin

SSH public key source Generate new key pair

Key pair name * Name the SSH public key

Please enter the exact version. Version names are generally of the form '1.2.3'

Manual entry of software version * ⓘ latest ✓

Optional Resource Tags

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. [Learn more about tags](#)

Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

Name ⓘ	Value ⓘ	Resource
CreatedBy	Jane Doe	2 selected

Previous Next Review + create

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](#).

5. Select the cloudSwXtch size.

cloudSwXtch Size Explained

The default size is **1x Standard D4 V4**. The cloudSwXtch size should work well for testing purposes, for production the size should be carefully considered based on traffic egress and ingress into and out of the cloudSwXtch.

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 swtch.io for cloudSwXtch licensing.

6. Enter in an **"Admin name."** This will default to "swtchadmin," 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."**

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. "

14. **OPTIONAL:** In the "Advanced Networking (optional)" section:

- Add a static IP Address
- Specify a Proximity Placement Group

Home > cloudSwxtch-from-mp-image >

cloudSwXtch instantiation template

Template spec

Configure virtual networks

Virtual network * ⓘ (new) pick-an-existing-vnet

Control Subnet Name * ⓘ (new) ControlSubnet (10.0.0.0/29)

Data Subnet Name * ⓘ (new) DataSubnet (10.0.1.0/29)

Advanced Networking (optional)

IP addresses are dynamic by default. ☒ Check this box to specify static IP address for network interfaces ⓘ

 Static IPs are not validated here. An invalid static IP address will cause the deployment to fail.

Enter static IP for network interface on ControlSubnet * ⓘ

Enter static IP for network interface on DataSubnet * ⓘ

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.

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.

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 [validate 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](#).

The screenshot shows the Microsoft Azure Marketplace interface. The search bar contains 'cloudSwXtch'. The results show two items: 'cloudSwXtch private offers' and 'cloudSwXtch VM Image'. The 'cloudSwXtch VM Image' item is highlighted with a red box, and its 'Create' button is selected, showing a dropdown menu with options: 'BYOL' and 'PAYG cloudSwXtch for trials and PoCs only'.

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 display to choose a plan.

The screenshot shows the 'cloudSwXtch VM Image' page. The 'Plan' dropdown menu is open, showing options: 'BYOL' and 'PAYG cloudSwXtch for trials and PoCs only'. The 'Create' button is visible next to the dropdown.

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

NOTE

swtch.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.

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) an "Resource Group"

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

11. Select a "Region"

12. Select the "Image" and choose an appropriate image based on the plan type: cloudSwXtch Multicast or cloudSwXtch BYOL.

Microsoft Azure (Preview) Search resources, services, and docs (G+)

Home > cloudSwXtch VM Image >

Create a virtual machine

Subscription * ⓘ sub-sw-x-sa

Resource group * ⓘ (New) Resource group
[Create new](#)

Instance details

Virtual machine name * ⓘ

Region * ⓘ (US) East US

Availability options ⓘ Availability zone

Availability zone * ⓘ Zone 1

Security type ⓘ Trusted launch virtual machines
[Configure security features](#)

Image * ⓘ cloudSwXtch Multicast - Gen1
[See all images](#)

Run with Azure Spot discount ⓘ ☐

Size * ⓘ Standard_D4_v4 - 4 vcpus, 16 GiB memory (Price unavailable)
[See all sizes](#)

Administrator account

Authentication type ⓘ ☒ SSH public key
☐ Password

[Review + create](#) < Previous Next: Disks > [Give feedback](#)

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.
14. Continue on the Networking Tab.

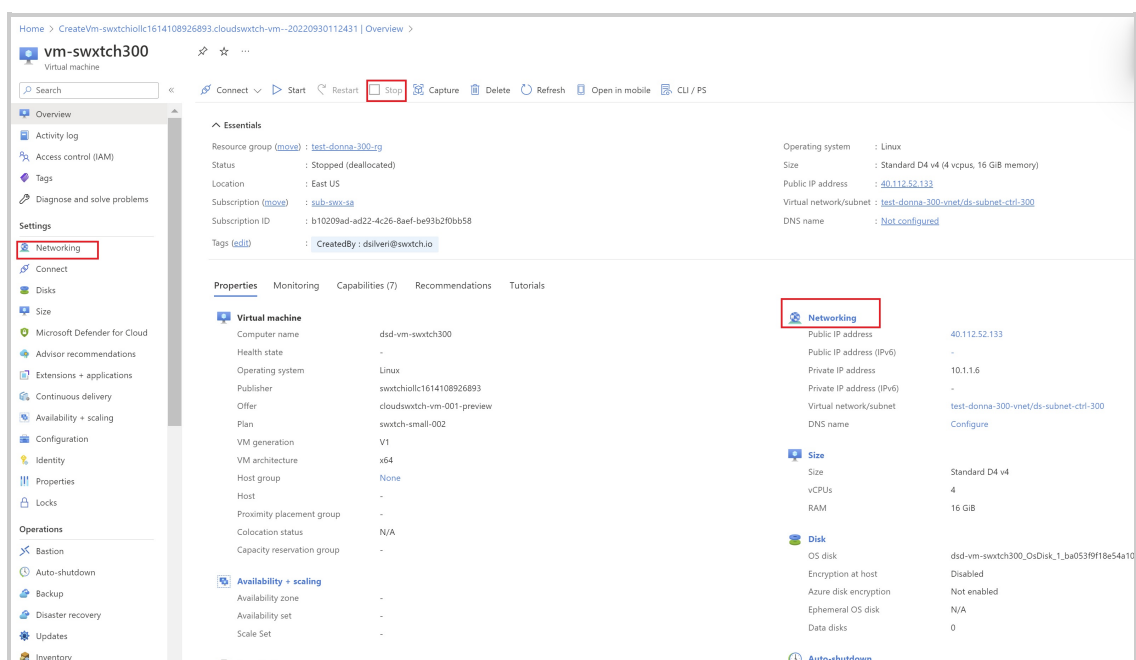
Networking Tab

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.

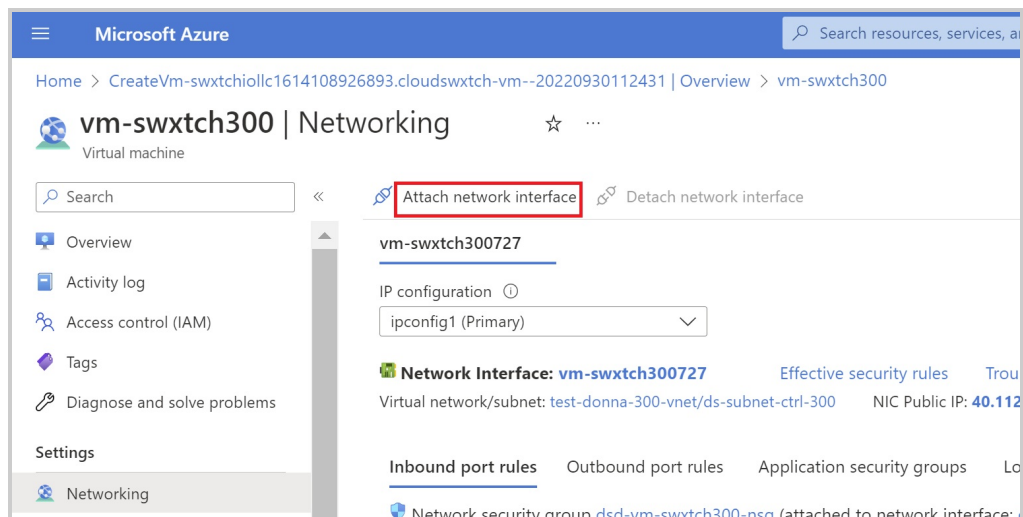
15. Check "Delete public IP and NIC when VM is deleted"
 16. **OPTIONAL:** Change values on other tabs.
 17. Select ****Review and Create****.
 18. **Carefully review** the plan pricing.
 19. Read the Terms & Conditions.
 20. Select "I agree" when ready.
- Please note: The creation will take 2-3 minutes 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.
4. Click "Networking" on the left hand side under settings. Alternatively, you can select "Networking" in the main Properties page.



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.

9. Select "Create"

Microsoft Azure

Home > vm-swxtch300 | Networking >

Create network interface

Project details

Subscription ⓘ
sub-swxt-sa

Resource group * ⓘ
test-300-rg
[Create new](#)

Location ⓘ
(US) East US

Network interface

Name *
vm-swxtch300-data ✓

Virtual network ⓘ
test-300-vnet

Subnet * ⓘ
subnet-data-300 (10.1.2.0/24)

NIC network security group ⓘ
☐ None
☒ Basic
☐ Advanced

Public inbound ports * ⓘ
☒ None
☐ Allow selected ports

Select inbound ports
Select one or more ports

All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.

Private IP address assignment
☒ Dynamic ☐ Static

Create

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

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."

Microsoft Azure (Preview) Search resources, services, and docs (G+/I)

Home > vm-swxtch300

vm-swxtch300 | Networking

Virtual machine

Search

Attach network interface Detach network interface

vm-swxtch300727 **vm-swxtch300-data**

IP configuration (Primary) ipconfig1 (Primary)

Network Interface: vm-swxtch300-data Effective security rules Troubleshoot VM connection issues Topology

Virtual network/subnet: test-donna-300-vnet/ds-subnet-data-300 NIC Public IP: - NIC Private IP: 10.1.2.6 Accelerated networking: **Disabled**

Inbound port rules Outbound port rules Application security groups Load balancing

Network security group basicNsgdsd-vm-swxtch300-data (attached to network interface: dsd-vm-swxtch300-data) Impacts 0 subnets, 1 network interfaces

Priority	Name	Port	Protocol	Source	Destination	Action
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Need help? Understand Azure load balancing Quickstart: Create a public load balancer to load balance Virtual Machines Quickstart: Direct web traffic with Azure Application Gateway

3. Click "Edit accelerated networking."

Microsoft Azure (Preview) Search resources, services, and docs (G+/I)

Home > vm-swxtch300 | Networking >

vm-swxtch300-data Network interface

Search

Move Delete Refresh **Edit accelerated networking**

Overview

Activity log Access control (IAM) Tags Settings

IP configurations DNS servers Network security group Properties Locks Monitoring

Insights Alerts Metrics Diagnostic settings Automation

Tasks (preview) Export template Help

Essentials

Resource group (move): test-300-rg Private IP address: 10.1.2.6

Location (move): East US Public IP address: -

Subscription (move): sub-swix-sa Private IP address (IPv6): -

Subscription ID: b10209ad-ad22-4c26-8aef-be93b2f0bb58 Public IP address (IPv6): -

Accelerated networking: Disabled Attached to: vm-swxtch300 (Virtual machine)

Virtual network/subnet: test-300-vnet/subnet-data-300 basicNsgdsd-vm-swxtch300-data (Network security group)

Type: Regular

Tags (edit): Click here to add tags

View Cost JSON View

Give feedback

4. Select "Enable."

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

6. Click "Save."

The screenshot shows the Microsoft Azure portal interface. On the left, there's a navigation pane with options like Overview, Activity log, Access control (IAM), Tags, Settings, IP configurations, DNS servers, Network security group, Properties, Locks, Monitoring, Insights, Alerts, Metrics, Diagnostic settings, Automation, Tasks (preview), Export template, and Help. The main area displays the 'vm-swxtch300-data' network interface. On the right, the 'Edit accelerated networking' dialog is open. It contains a description of accelerated networking, radio buttons for 'Automatic (recommended)', 'Enabled' (selected and highlighted with a red box), and 'Disabled'. Below these, a text box contains the statement 'I have validated that the operating system supports accelerated networking.' which is also highlighted with a red box. At the bottom of the dialog, there are 'Save' and 'Discard' buttons, with 'Save' being highlighted with a red box.

7. Start the VM for use.

For cloudSwXtch BYOL

Users will need to contact swXtch.io in order to obtain a license file. For more information, see [How to License a cloudSwXtch](#).

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](#).

Deploy cloudSwXtch with Terraform on Azure

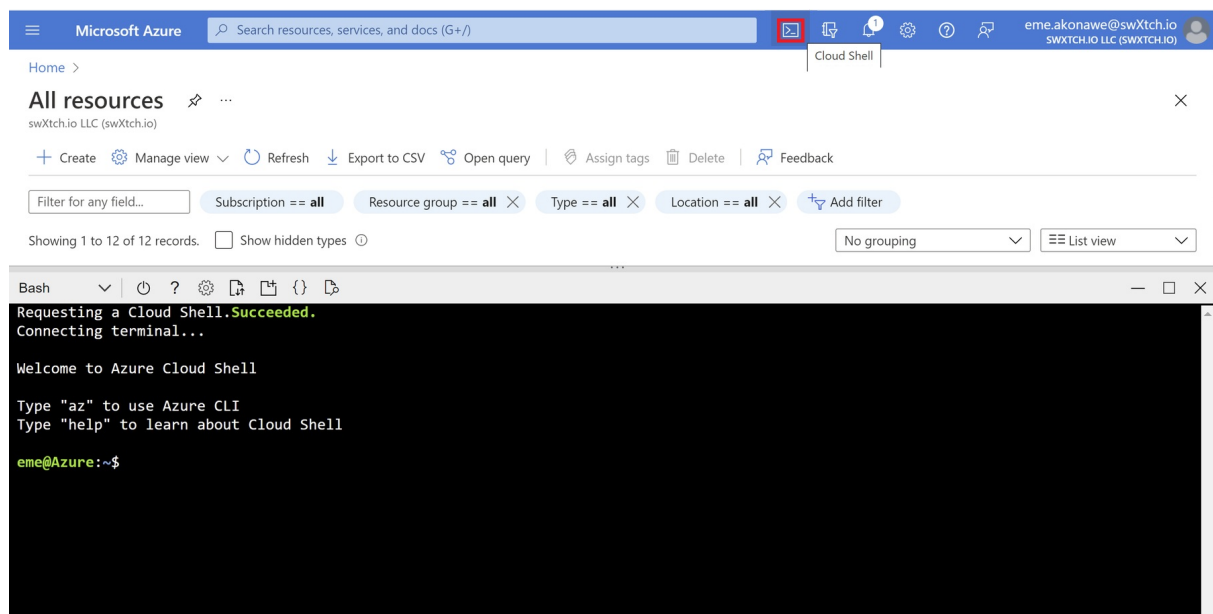
WHAT TO EXPECT

By default, the terraform script will spin up a "small" cloudSwStch. You can make edits to the [Azure/terraform/terraform.tfvars](#) 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):

Console	Copy
<pre>\$ git clone git@github.com:swxtch.io/cloudSwXtch-support.git</pre>	

or **via HTTPS**:

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

4. Update the values in the [Azure/terraform/terraform.tfvars](#) 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:

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

Azure services



Create a resource



Storage accounts



Virtual machines



Quotas



Resource groups



Kubernetes services



Kubernetes fleet manager



Virtual networks

Bash

```
Initializing the backend...
```

```
Initializing provider plugins...
```

```
- Finding hashicorp/azurerm versions matching "~> 3.0"...
- Installing hashicorp/azurerm v3.64.0...
- Installed hashicorp/azurerm v3.64.0 (signed by HashiCorp)
```

```
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.
```

```
Terraform has been successfully initialized!
```

```
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

```
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

```
nicholas [ ~/projects/cloudSwXtch-support/Azure/terraform ]$
```

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

```
$ terraform plan
```

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-c86c27545c1a/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.

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_deployswxthch.tf](#) & [terraform.tfvars](#) files.

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

```
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.location
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:

```
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.

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

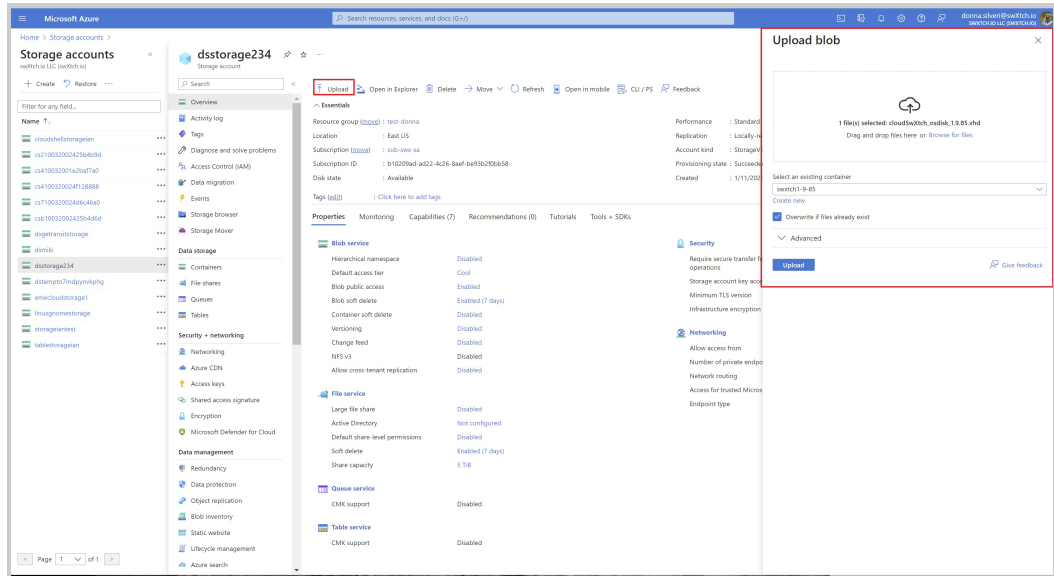
Log onto an environment that has access to the internet and download the following file (~30GB):

None	Copy
<code>https://swxtchpublic.blob.core.windows.net/3hwgfe98hfglsrdfh4/cloudSwXtch_osdisk_1.9.85.vhd</code>	

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.
 1. Log into the Azure Portal
 2. Navigate to **Storage Accounts**.
 3. Select the desired storage account.
 4. Select the desired Container or create a new one.
 5. Select **Upload** and select the VM Disk Image file you copied to the local PC.

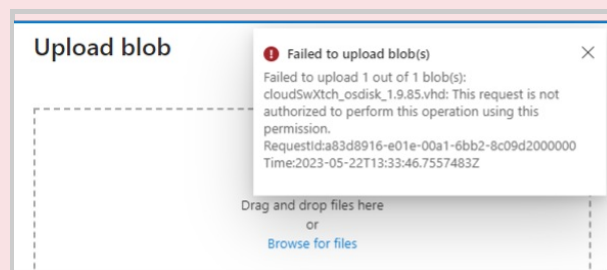


6. Start the upload and wait for it to complete.

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.



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**.
 1. In the new panel, navigate to the storage account and container holding the disk image.
 2. Select the file that was previously uploaded.
9. For **Account Type**, select **Standard SSD**. See the example of the screen filled out completely.

Microsoft Azure

Home > Images >

Create an image

Create a managed image that can be used to deploy virtual machines and virtual machine scale sets. The image contains a list of managed blobs and metadata necessary for creating virtual machines. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ sub-sw-x-sa

Resource group * ⓘ saDevNetwork [Create new](#)

Instance details

Name * cloudSwXtch_1_9_85_Image ✓

Region * ⓘ (US) East US

Zone resiliency ⓘ ☐

OS disk

OS type * ⓘ ☐ Windows ☒ Linux

VM generation * ⓘ ☒ Gen 1 ☐ Gen 2

Storage blob * ⓘ https://dsstorage234.blob.core.windows.net/swxtch1-9-85/cloudSwXtch_osdis... ✓ [Browse](#)

Account type * ⓘ Standard SSD

Host caching * ⓘ Read/write

Encryption

You can encrypt the OS and data disks with a platform-managed or customer-managed key. [Learn more](#)

Key management ⓘ Platform-managed key

Data disk

[+ Add data disk](#)

[Review + create](#) [< Previous](#) [Next : Tags >](#)

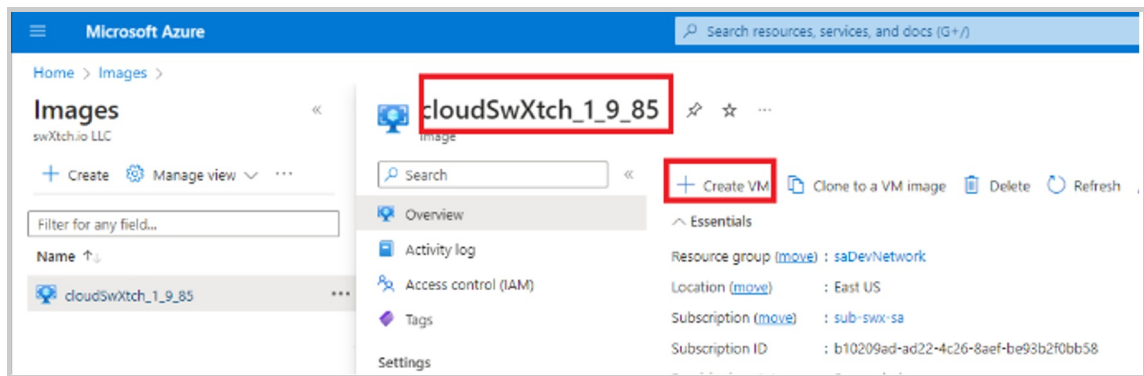
10. If tags are desired, then select **Tags** and enter the required tags.
11. The other fields can be left as default.
12. Select **Review and create**.
13. When validation passes, select **Create**. When it is complete, click **Go to Resource** to see the image.

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:

Microsoft Azure

Home > Microsoft.Image-20230603122553 | Overview > cloudSwXtch_1_9_85_Image >

Create a virtual machine

Basics Disks Networking Management Monitoring Advanced Tags Review + create

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group * [Create new](#)

Instance details

Virtual machine name *

Region

Availability options

Security type

Image * [See all images](#) | [Configure VM generation](#)

VM architecture ☐ Arm64 ☒ x64

Arm64 is not supported with the selected image.

Run with Azure Spot discount ☐

Size * [See all sizes](#)

Administrator account

Authentication type ☒ SSH public key ☐ Password

Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.

Username *

SSH public key source

Stored Keys

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * ☒ None ☐ Allow selected ports

Select inbound ports

All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.

Licensing

License type *

If you are using a RedHat or SLES image, you may be eligible for the Azure Hybrid Benefit and can save money on the license costs. [Learn more](#) about this benefit and how to enable it using Azure CLI for custom images from snapshots and Azure compute gallery.

[Review + create](#) [< Previous](#) [Next : Disks >](#)

1. Set the **subscription** and **Resource Group** for where you want the cloudSwXtch instance to be located.
2. Name the Virtual Machine with a valid host name.
3. Select appropriate machine size. For recommendations based on features, endpoints, and bandwidth needs, read the [Quotas](#) article.
4. Use SSH for the authentication type. Enter your **SSH public key source**. Refer to [ssh-keys-portal](#) for details.
5. Set the **Licensing Type** to **Other**.

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

Microsoft Azure

Home > Microsoft.Image-20230603122553 | Overview > cloudSwXtch_1_9_85_Image >

Create a virtual machine

Basics Disks **Networking** Management Monitoring Advanced Tags Review + create

Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution. [Learn more](#)

Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network * [Create new](#)

Subnet * [Manage subnet configuration](#)

Public IP [Create new](#)

NIC network security group ☒ None ☐ Basic ☐ Advanced

i The selected subnet 'sa-subnet-ctrl (10.2.128.0/22)' is already associated to a network security group 'Sa-NSG'. We recommend managing connectivity to this virtual machine via the existing network security group instead of creating a new one here.

Delete public IP and NIC when VM is deleted ☐

Enable accelerated networking ☐ The selected image does not support accelerated networking.

Load balancing

You can place this virtual machine in the backend pool of an existing Azure load balancing solution. [Learn more](#)

Place this virtual machine behind an existing load balancing solution? ☐

[Review + create](#) [< Previous](#) [Next : Management >](#)

1. Select the appropriate **Virtual Network**.

2. Select the appropriate control subnet.

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

8. Select **Review + Create**.

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

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

1. 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.

The screenshot shows the 'Create network interface' form in the Microsoft Azure portal. The form is titled 'Create network interface' and is located under 'Home > CloudSwXtch01 | Networking >'. The form is divided into several sections: 'Project details', 'Network interface', 'NIC network security group', 'Public inbound ports', and 'Private IP address assignment'. The 'Project details' section includes fields for 'Subscription' (sub-swX-sa), 'Resource group' (saDevNetwork), and 'Location' ((US) East US). The 'Network interface' section includes fields for 'Name' (cloudswxtch01-data), 'Virtual network' (sa-vnet), and 'Subnet' (sa-subnet-data (10.2.192.0/22)). The 'NIC network security group' section has radio buttons for 'None', 'Basic' (selected), and 'Advanced'. The 'Public inbound ports' section has radio buttons for 'None' (selected) and 'Allow selected ports'. The 'Private IP address assignment' section has radio buttons for 'Dynamic' (selected) and 'Static', and a checkbox for 'Private IP address (IPv6)'. A 'Create' button is at the bottom.

Microsoft Azure

Home > CloudSwXtch01 | Networking >

Create network interface

Project details

Subscription ⓘ
sub-swX-sa

Resource group * ⓘ
saDevNetwork
[Create new](#)

Location ⓘ
(US) East US

Network interface

Name *
cloudswxtch01-data ✓

Virtual network ⓘ
sa-vnet

Subnet * ⓘ
sa-subnet-data (10.2.192.0/22)

NIC network security group ⓘ
☐ None
☒ Basic
☐ Advanced

Public inbound ports * ⓘ
☒ None
☐ Allow selected ports

Select inbound ports
Select one or more ports

Private IP address assignment

☒ Dynamic ☐ Static

☐ Private IP address (IPv6)

[Create](#)

All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.

9. Select **Create**.

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

The screenshot shows the 'CloudSwXtch01 | Networking' page in the Microsoft Azure portal. The page is titled 'CloudSwXtch01 | Networking' and is located under 'Home > CloudSwXtch01 | Networking >'. The page is divided into several sections: 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Settings', 'Networking', 'Connect', 'Disks', 'Size', 'Microsoft Defender for Cloud', 'Advisor recommendations', 'Extensions + applications', 'Availability + scaling', 'Configuration', 'Identity', 'Properties', 'Locks', and 'Operations'. The 'Networking' section is selected, and the 'cloudswxtch01-data' network interface is highlighted. The 'cloudswxtch01-data' network interface is shown with its IP configuration (ipconfig1 (Primary)) and its network interface details (Network Interface: cloudswxtch01-data, Virtual network/subnet: sa-vnet/sa-subnet-data, NIC Public IP: -, NIC Private IP: 10.2.192.85, Accelerated networking: Disabled). The 'Inbound port rules' section is expanded, showing a table of rules. The table has columns for Priority, Name, Port, Protocol, Source, Destination, and Action. The rules are: 65000 AllowVnetInbound (Any, Any, VirtualNetwork, VirtualNetwork, Allow), 65001 AllowAzureLoadBalancerInbound (Any, Any, AzureLoadBalancer, Any, Allow), and 65000 DenyAllInbound (Any, Any, Any, Any, Deny). A 'Need help?' section is at the bottom, with links to 'Understand Azure load balancing', 'Quickstart: Create a public load balancer to load balance Virtual Machines', and 'Quickstart: Direct web traffic with Azure Application Gateway'.

Microsoft Azure

Home > CloudSwXtch01 | Networking

cloudswxtch01-data

IP configuration ⓘ
ipconfig1 (Primary)

Network Interface: cloudswxtch01-data
Virtual network/subnet: sa-vnet/sa-subnet-data NIC Public IP: - NIC Private IP: 10.2.192.85 Accelerated networking: Disabled

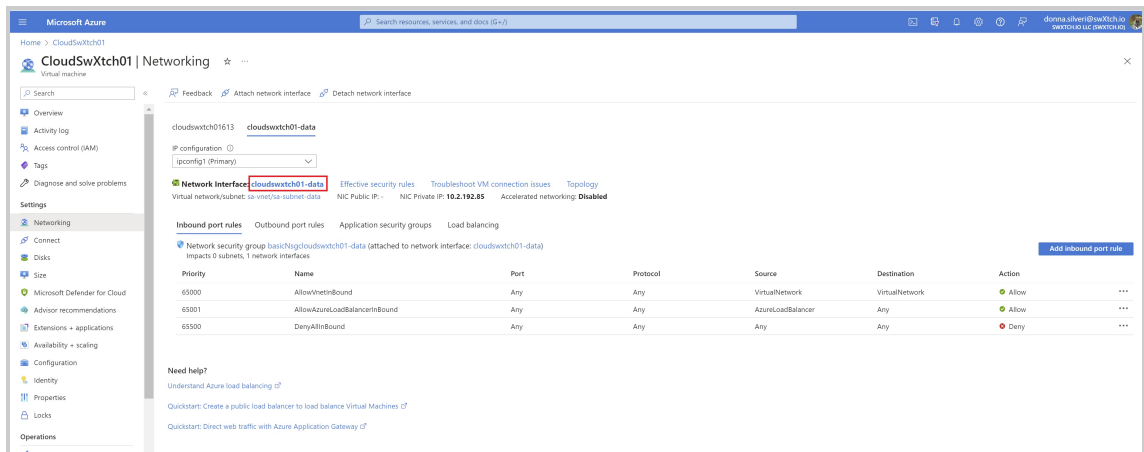
Inbound port rules Outbound port rules Application security groups Load balancing

Network security group basicRgCloudSwXtch01-data (attached to network interface: cloudswxtch01-data)
Impacts 0 subnets, 1 network interfaces

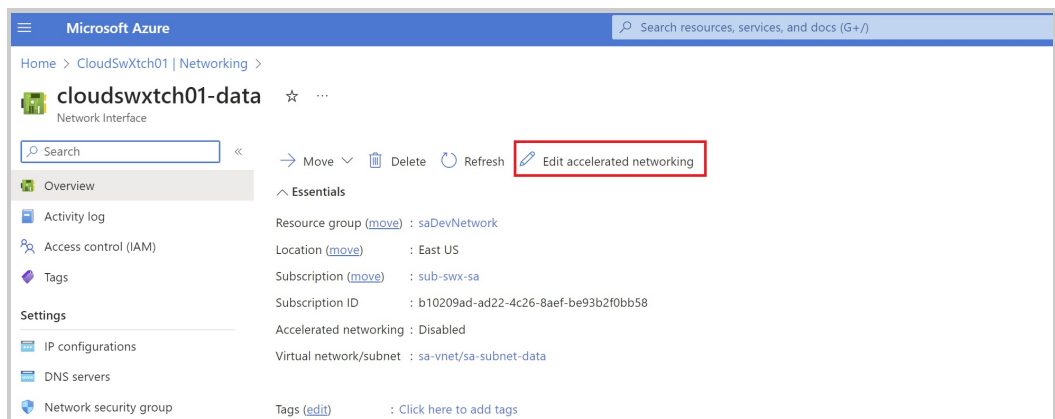
Priority	Name	Port	Protocol	Source	Destination	Action
65000	AllowVnetInbound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInbound	Any	Any	AzureLoadBalancer	Any	Allow
65000	DenyAllInbound	Any	Any	Any	Any	Deny

Need help?
[Understand Azure load balancing](#)
[Quickstart: Create a public load balancer to load balance Virtual Machines](#)
[Quickstart: Direct web traffic with Azure Application Gateway](#)

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



1. Select Edit accelerated Networking.



2. A new window will display.

Edit accelerated networking

For supported operating systems, accelerated networking lowers latency, reduces jitter, and decreases CPU utilization. When communicating across virtual networks or connecting on-premises, enabling accelerated networking has minimal impact to latency. [Learn more](#)

Accelerated networking

To allow Azure to enable accelerated networking when it detects it is supported by the operating system, select Automatic.

☐ Automatic (recommended)
 ☒ **Enabled**
☐ Disabled

Network connectivity will be interrupted if the operating system of your virtual machine does not support accelerated networking. Disable accelerated networking if connectivity to your virtual machine is interrupted.

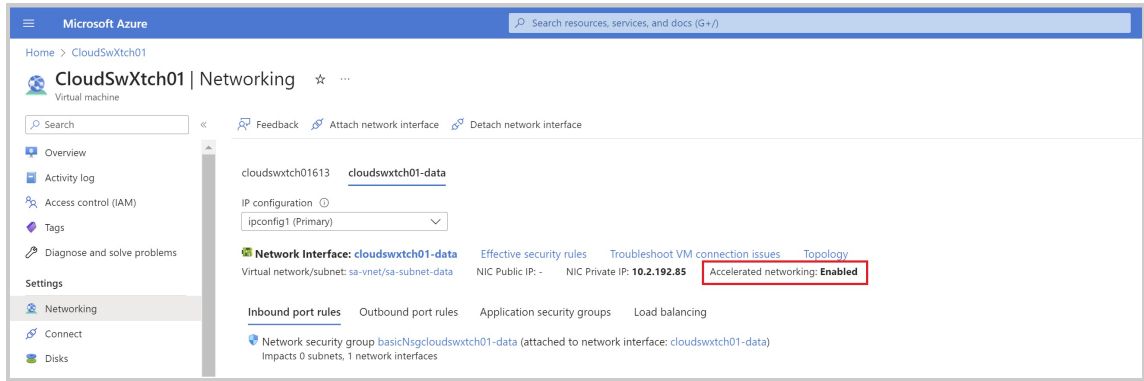
☒ **I have validated that the operating system supports accelerated networking.**

3. Select Enabled.

4. Check the agreement.

5. 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.

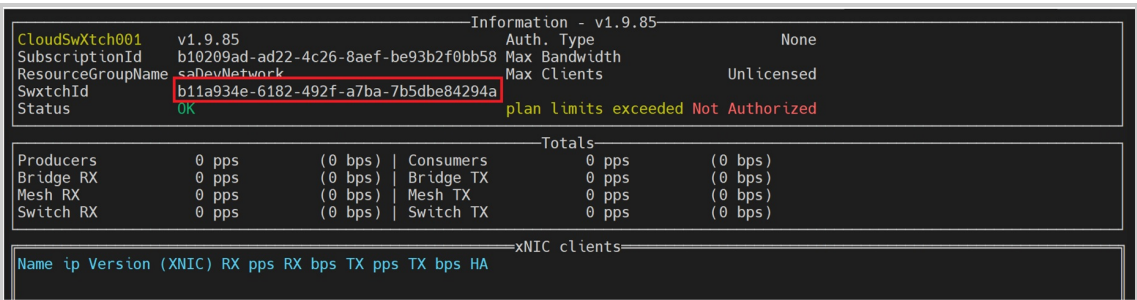
STEP FIVE: License the cloudSwXtch

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

Text

Bash	Copy
<pre>sudo /swxtch/swxtch-top dashboard --swxtch localhost</pre>	

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	Copy
<pre>sudo mv /home/<user>/license.json /swxtch</pre>	

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

Text

Bash	Copy
<pre>sudo journalctl -u swxtch-ctrl.service -f -n 500</pre>	

Information - v1.9.85

CloudSwXtch001

SubscriptionId

ResourceGroupName

SwxtchId

Status

v1.9.85(CloudSwXtch001 Customer License)

b10209ad-ad22-4c26-8aef-be93b2f0bb58

saDevNetwork

b11a934e-6182-492f-a7ba-7b5dbe84294a

OK

Auth. Type

License File

Max Bandwidth

Max Clients

100000 Mbps

30

Totals

Producers

Bridge RX

Mesh RX

Switch RX

0 pps

0 pps

0 pps

0 pps

(0 bps)

(0 bps)

(0 bps)

(0 bps)

Consumers

Bridge TX

Mesh TX

Switch TX

0 pps

0 pps

0 pps

0 pps

(0 bps)

(0 bps)

(0 bps)

(0 bps)

xNIC clients

Name

ip

Version (XNIC)

RX pps

RX bps

TX pps

TX bps

HA

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).

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](#).

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

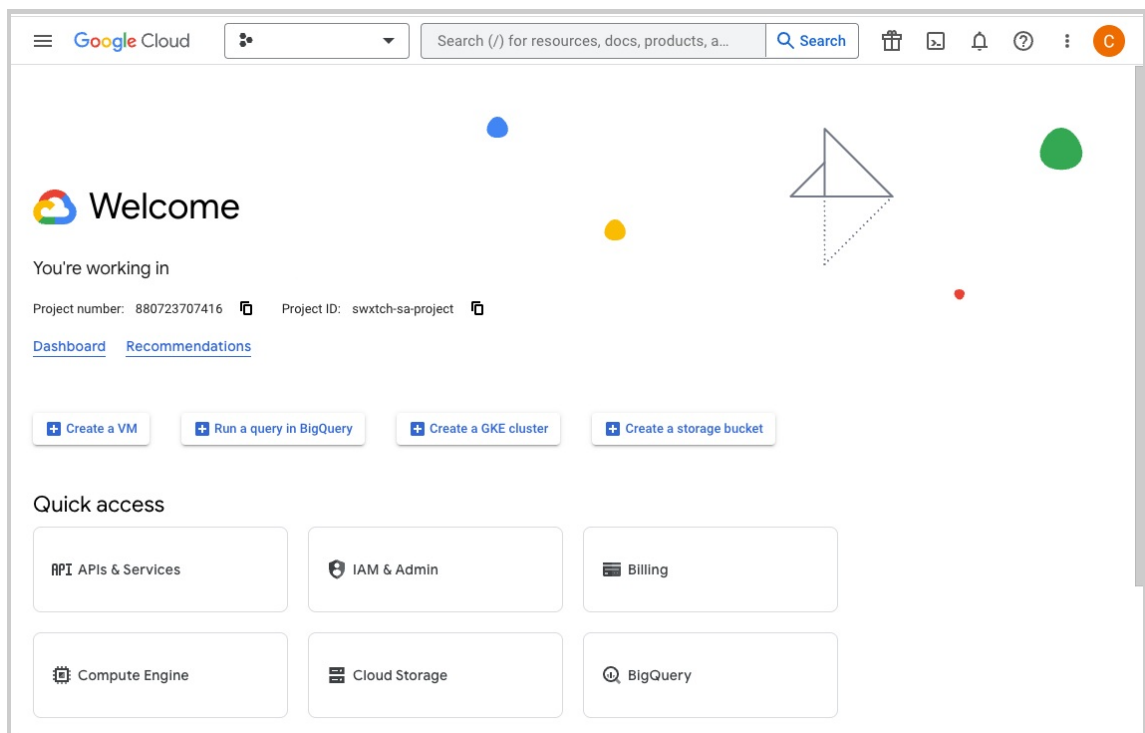
A user needs the following to deploy a cloudSwXtch via the GCP Marketplace:

- 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](#).

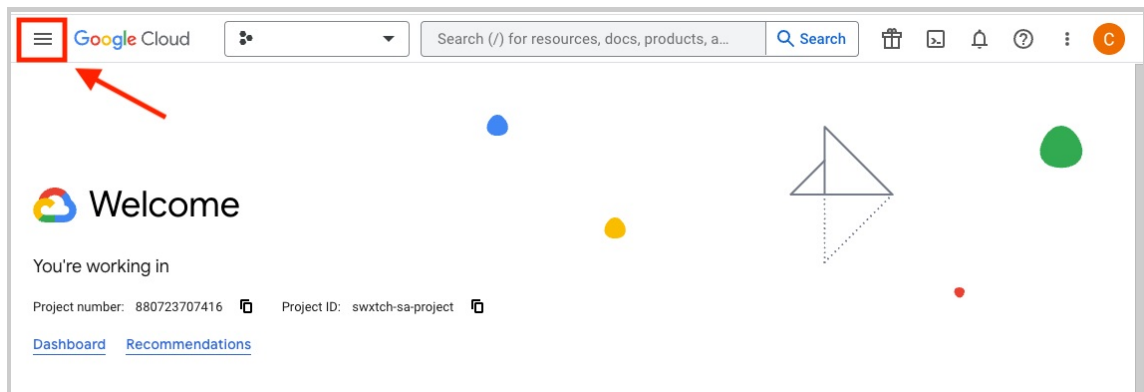
Please review [cloudSwXtch System Requirements](#) for additional prerequisites.

Step One: Navigate to cloudSwXtch in the GCP Marketplace

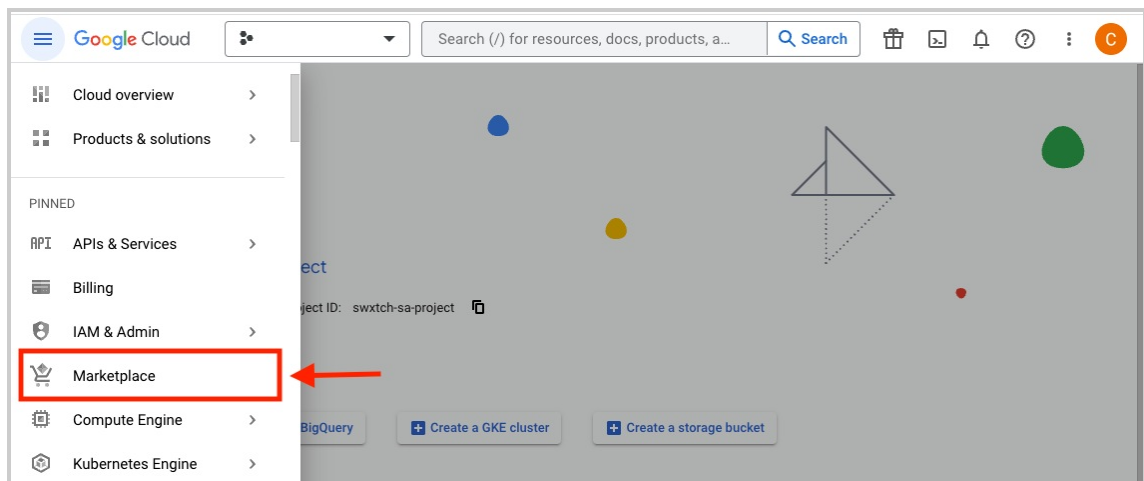
1. Log into the Google Cloud Console.



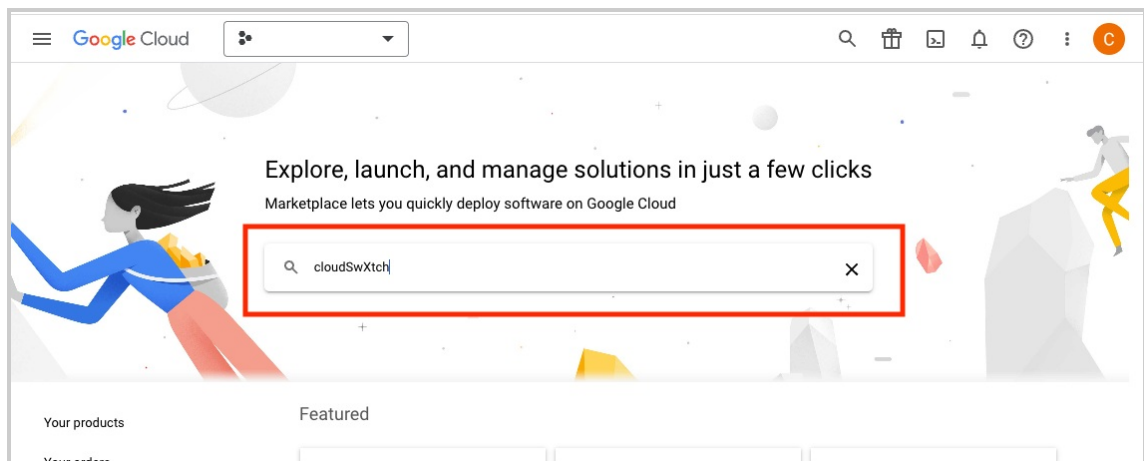
2. Navigate to the Google Cloud Marketplace using the Navigation menu at the top left corner.



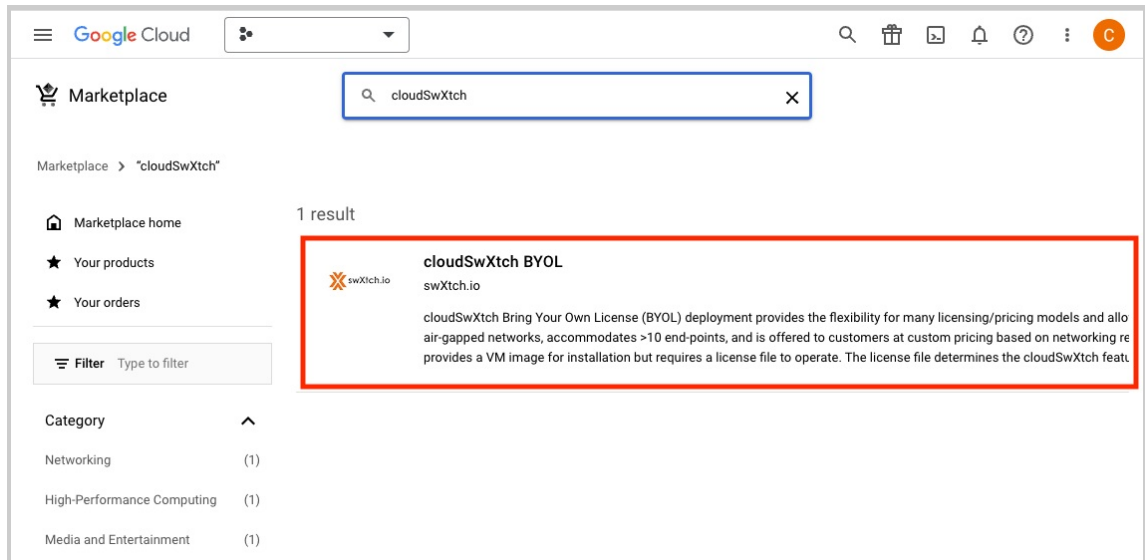
3. Select Marketplace.



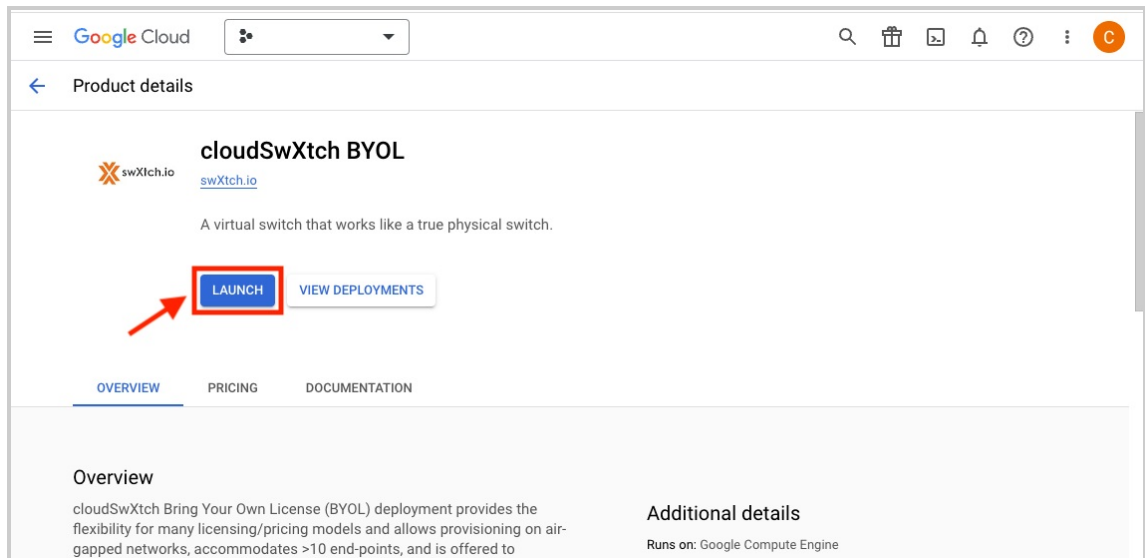
4. Search for cloudSwXtch.



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.

Step Two: Configure cloudSwXtch Deployment

1. Enter a **Deployment name** for your cloudSwXtch.

2. Select an Existing account under Deployment Service Account.

Prices don't include private offer discounts

Deployment name *
cloudswxtch-1

Deployment Service Account ②
☒ Existing account
☐ New account
List of available Service Accounts that have the following roles:
• roles/compute.admin
There are no Service Accounts matching the requirements above
Select a Service Account

Swxtch Image Id
cloudswxtch-2-0-34-2023-09-27

Swxtch Machine Type
☒ General purpose ☐ Compute optimized ☐ Memory optimized
Machine types for common workloads, optimized for cost and flexibility

Additional information
cloudSwXtch BYOL overview
Product provided by swXtch.io
Launching a BYOL product
cloudSwXtch BYOL is a BYOL (Bring Your Own License) product. Marketplace will deploy this product, but you are responsible for purchasing and managing the license directly from the provider
License for Cloud Marketplace virtual machine image solution with billing service cloudswxtch-byol.endpoints.swxtchio-public.cloud.goog: default usage fee (BYOL) ②
Google does not collect this license fee.
Price estimates based on 30-day, 24hrs per day usage of the listed resources in the selected region. The Monthly Infrastructure Fee is not included in the estimates and varies depending on all Google Cloud IaaS resources actually created or consumed by this product (or the fees charged for such consumption). The provider of this product may be able to provide a more accurate estimate of monthly GCP IaaS consumption.

1. 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 resource manager.projects.setIamPolicy permissions) to allow you to create a new Deployment Service Account.
2. 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.
3. 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.

Prices don't include private offer discounts

Deployment name *
cloudswxtch-2

Deployment Service Account ②
☐ Existing account
☒ New account
Create a new Service Account
This will create a new Service Account with the following roles:
• roles/compute.admin
Name *
cloudswxtch-2
ID *
cloudswxtch-2
Description
Swxtch Image Id
cloudswxtch-2-0-34-2023-09-27

Additional information
cloudSwXtch BYOL overview
Product provided by swXtch.io
Launching a BYOL product
cloudSwXtch BYOL is a BYOL (Bring Your Own License) product. Marketplace will deploy this product, but you are responsible for purchasing and managing the license directly from the provider
License for Cloud Marketplace virtual machine image solution with billing service cloudswxtch-byol.endpoints.swxtchio-public.cloud.goog: default usage fee (BYOL) ②
Google does not collect this license fee.
Price estimates based on 30-day, 24hrs per day usage of the listed resources in the selected region. The Monthly Infrastructure Fee is not included in the estimates and varies depending on all Google Cloud IaaS resources actually created or consumed by this product (or the fees charged for such consumption). The provider of this product may be able to provide a more accurate estimate of monthly GCP IaaS consumption.

3. Select the desired SwXtch Image ID. This will auto-populate with the most recent version of cloudSwXtch.
4. Under SwXtch Machine Type, confirm that N2 is selected under Series.

- 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](#).
- Confirm your desired **Zone**.

Google Cloud swxtch-sa-project

New cloudSwXtch BYOL deployment [SEND FEEDBACK](#)

Swxtch Machine Type

☒ General purpose ☐ Compute optimized ☐ Memory optimized

Machine types for common workloads, optimized for cost and flexibility

Series: N2

Powered by Intel Cascade Lake and Ice Lake CPU platforms

Machine type: n2-standard-16 (16 vCPU, 8 core, 64 GB memory)

	vCPU	Memory
	16	64 GB

Zone: us-central1-a

Price estimates based on 30-day, 24hrs per day usage of the listed resources in the selected region. The Monthly Infrastructure Fee is not included in the estimates and varies depending on all Google Cloud IaaS resources actually created or consumed by this product (or the fees charged for such consumption). The provider of this product may be able to provide a more accurate estimate of monthly GCP IaaS consumption.

- 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.

Control network interface

Definitions for the control network interface.

Network interfaces

default default (10.128.0.0/20)	 
---------------------------------	---

[ADD A NETWORK INTERFACE](#)

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

Control network interface

Definitions for the control network interface.

Network interfaces

Edit network interface  

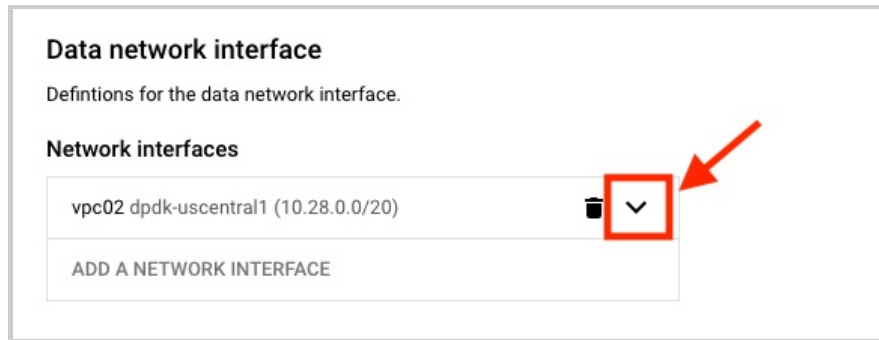
Network: default

Subnetwork: default

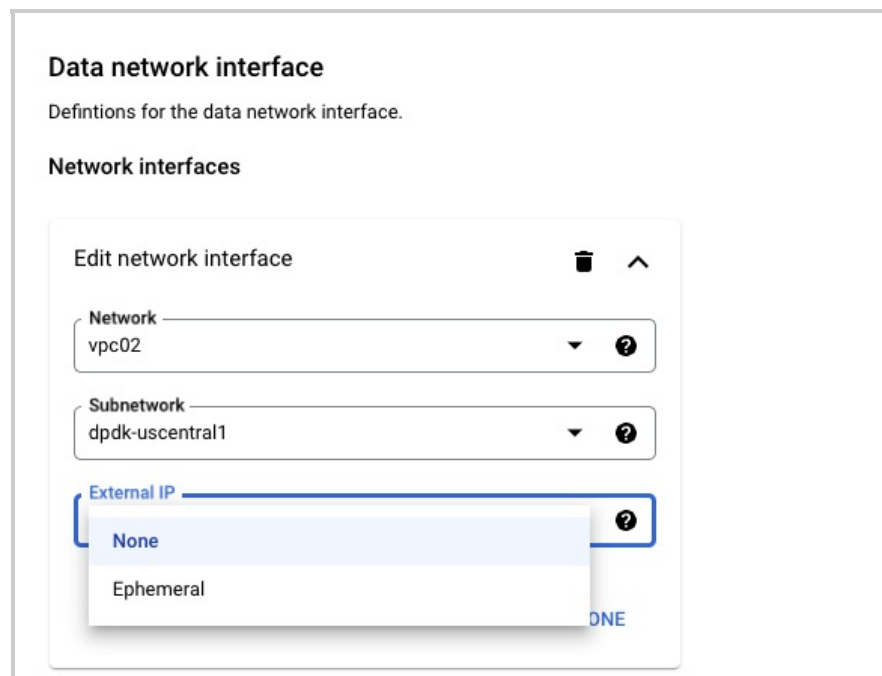
External IP: 

- None
- Ephemeral

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

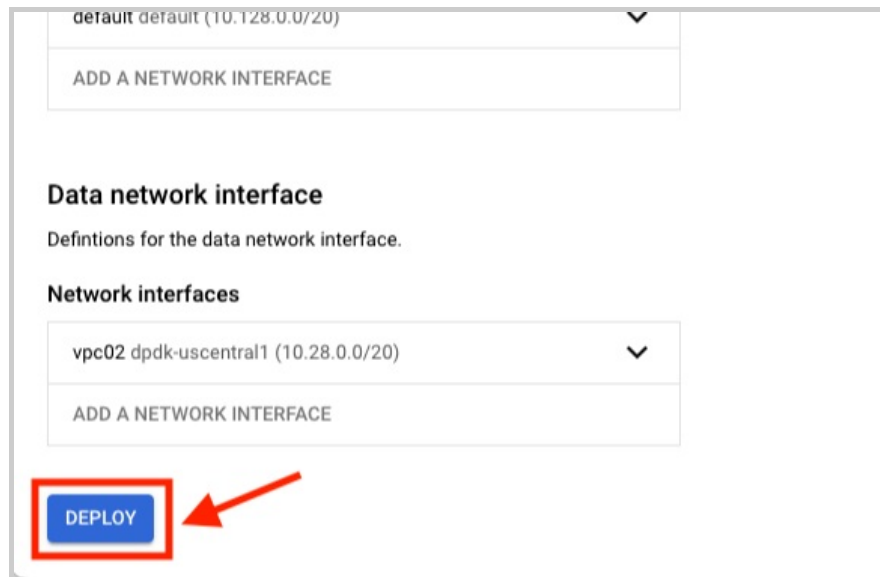


11. Select a **Network** and **Subnetwork**. This 2nd subnet will be used for your data plane communications and should've been created before starting the deployment process.
 1. **Please note:** The control and data NICs cannot share a subnet. They must have separate subnets.
 2. **Optional:** Users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

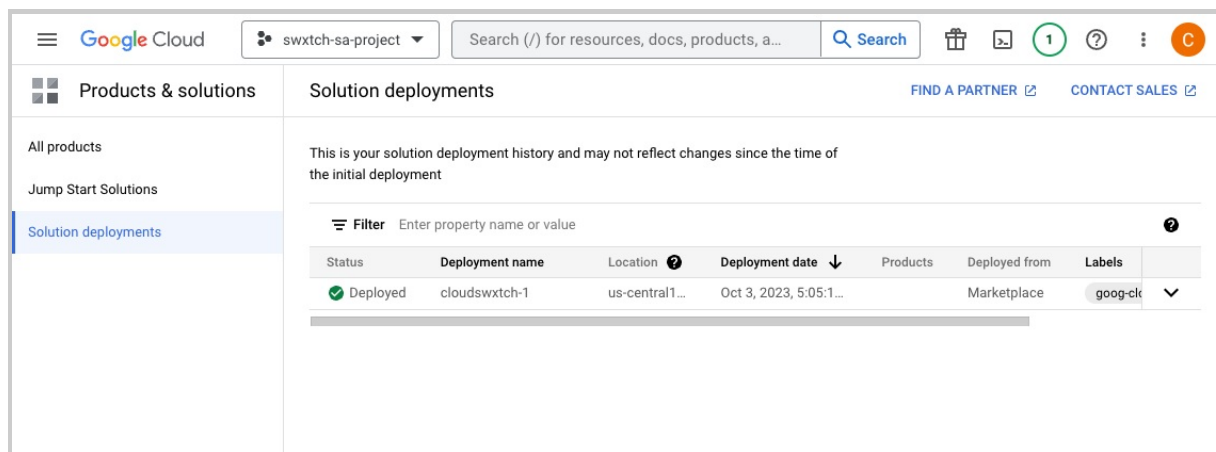


12. Click **Done** when you are happy with your selections.

13. Click Deploy.



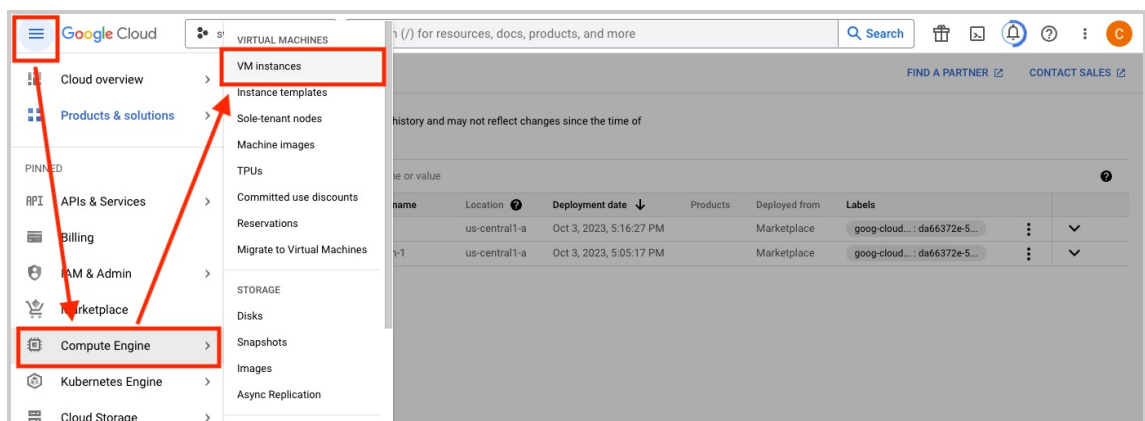
Your cloudSwXtch instance will now be deployed.



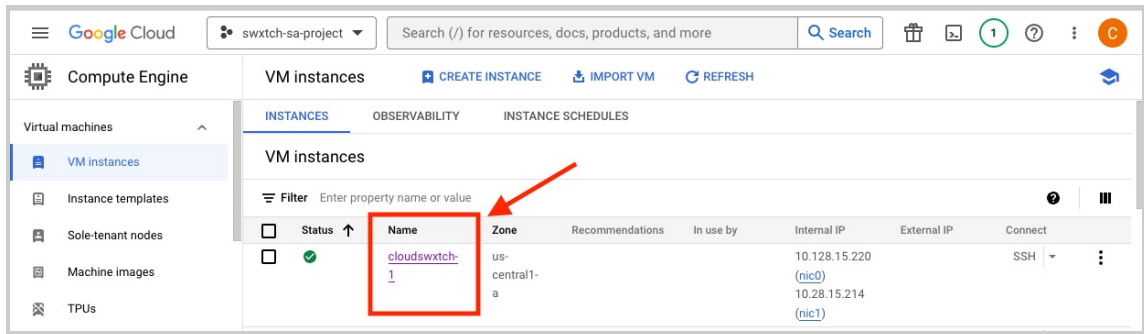
Step Three: Add SSH Key(s)

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

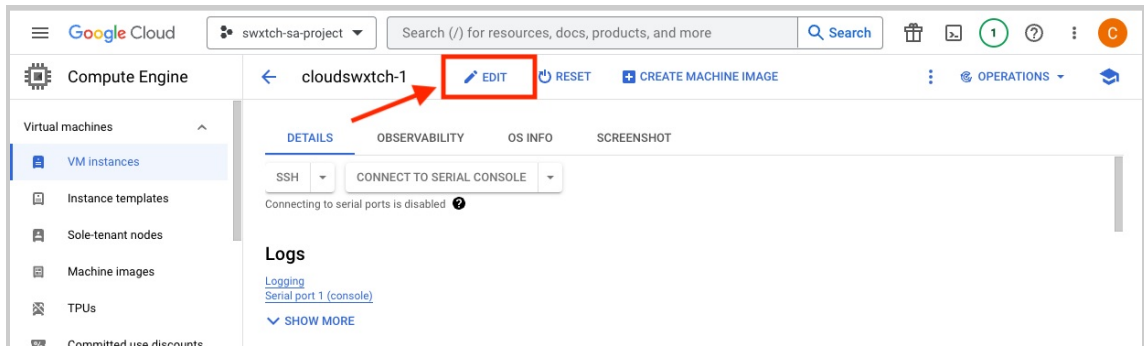
1. Click on the **Navigation** menu on the left hand corner, highlight **Compute Engine** and select **VM instances**.



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

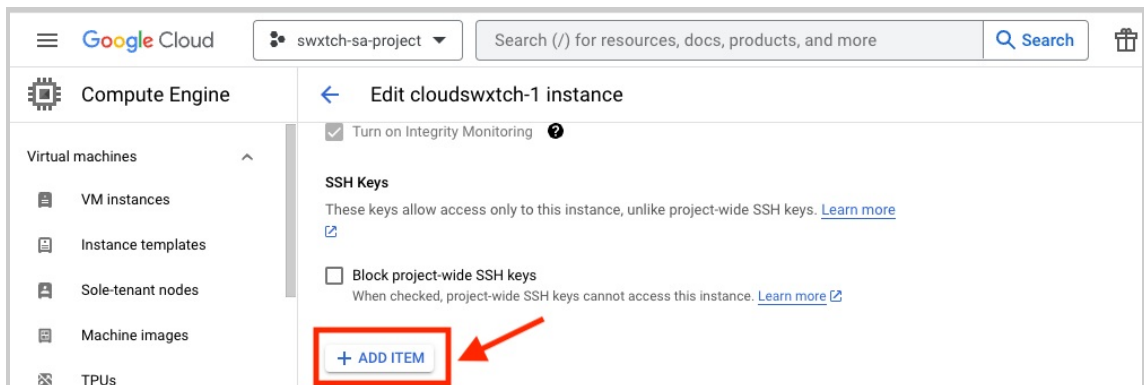


3. Choose Edit next to your cloudSwXtch deployment name.

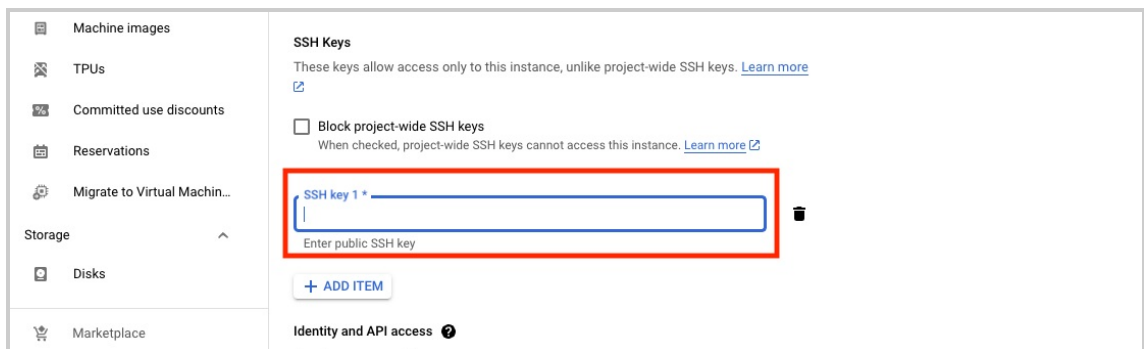


4. Scroll down to SSH Keys under Security and Access.

5. Click +Add Item.



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



7. Hit Save at the bottom of the page.

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](#).

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.

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](#)

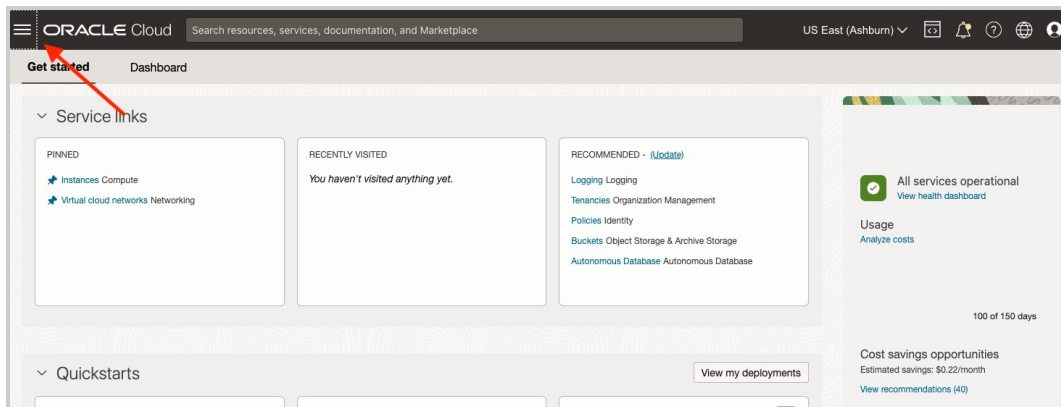
Please note: At this time, our only product offering in OCI is a BYOL instance of cloudSwXtch. This requires a user to contact swXtch.io for a license.

Prerequisites

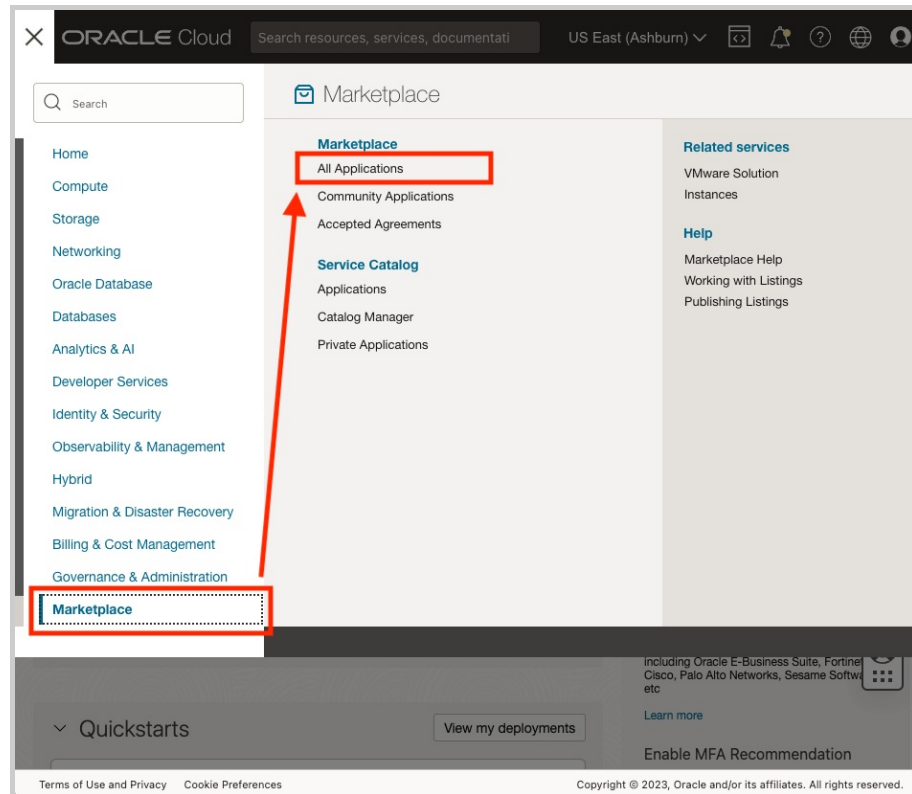
A user should have a **Compartment** established in their Oracle Cloud console before they start to deploy a cloudSwXtch. For more information about compartments, please see the [Managing Compartments](#) page under Oracle Cloud Infrastructure Documentation.

Step One: Navigate to cloudSwXtch in the Oracle Marketplace

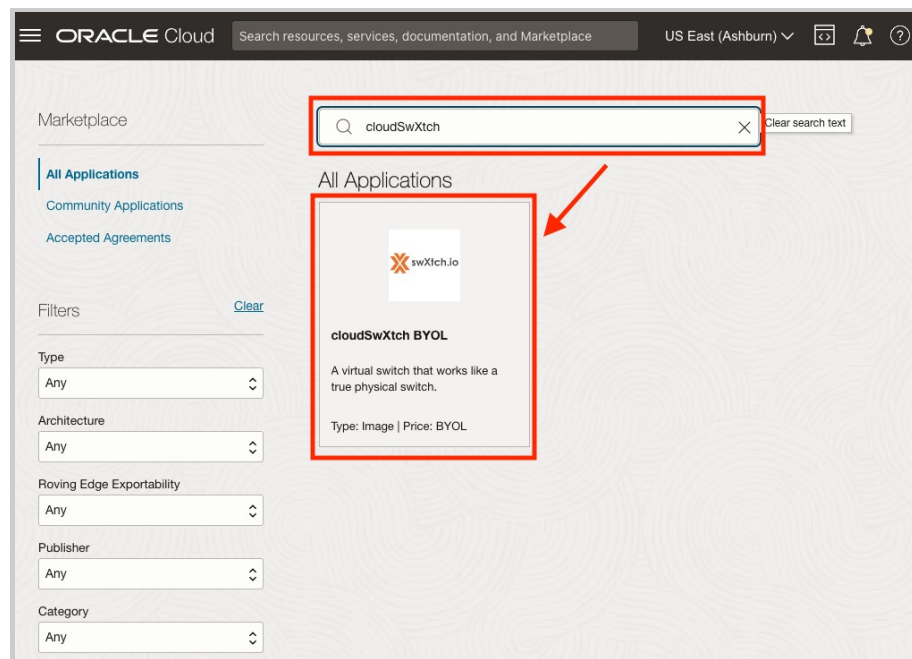
1. Log into Oracle Cloud.
2. Navigate to the Oracle Cloud Marketplace using the Navigation menu at the top left corner.



3. Select Marketplace and All Applications.

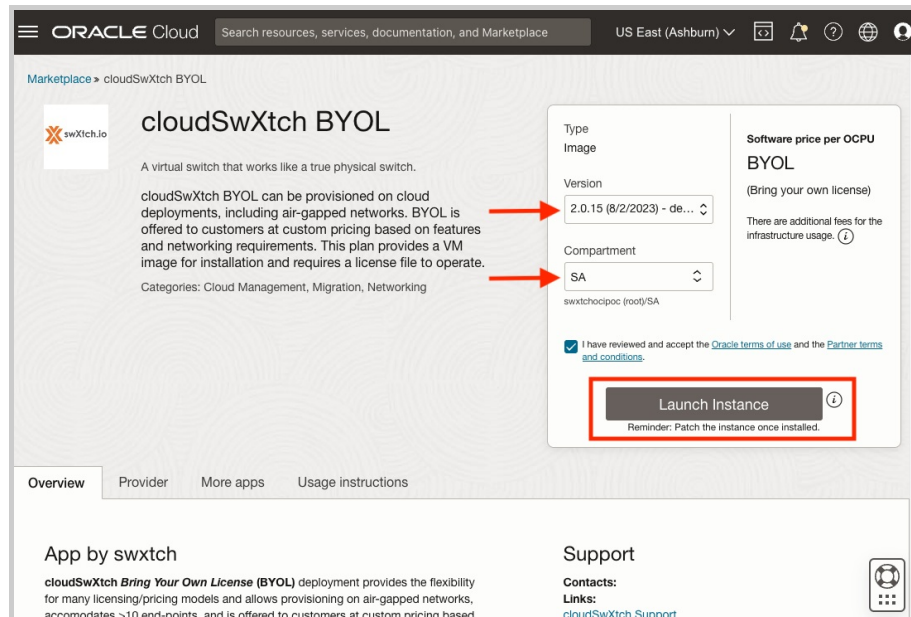


4. Search for cloudSwXtch and select the product, cloudSwXtch BYOL.



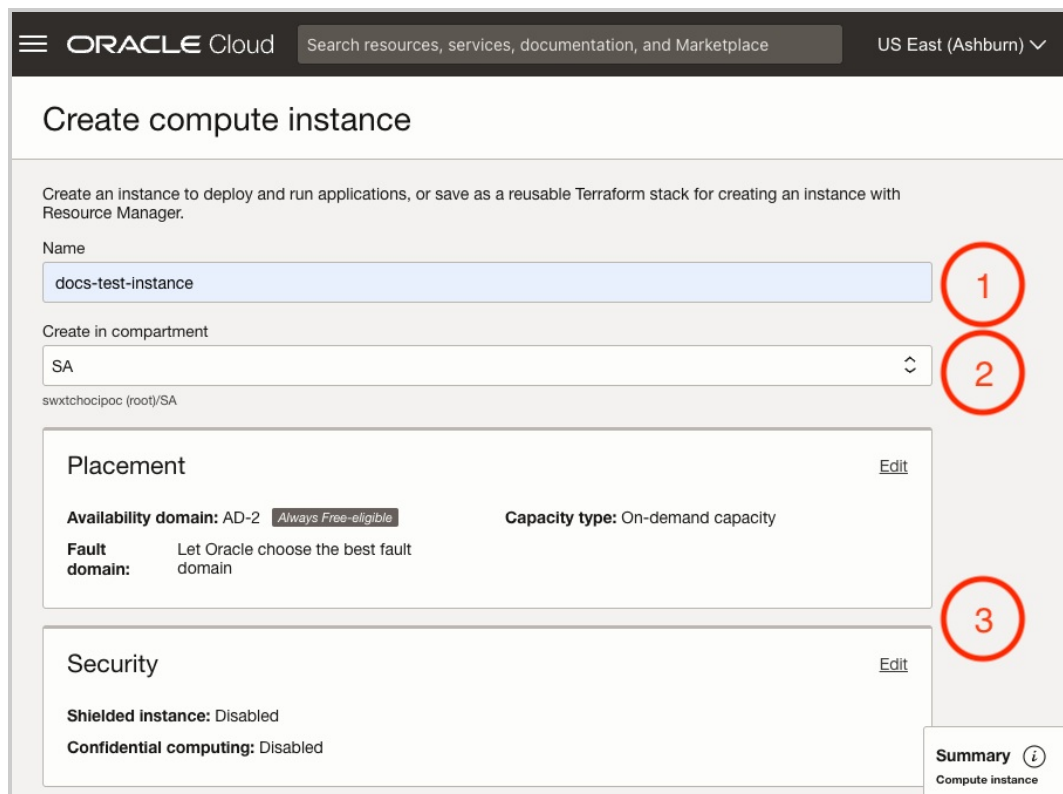
5. Select the Version and the Compartment that you will like to use. It is best to use the default since it will be the most recent version.

6. Click **Launch Instance** when you're happy with your selections.

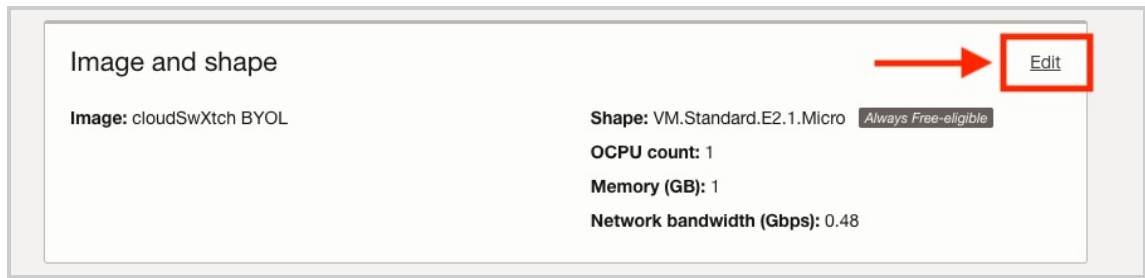


Step Two: Create Compute Instance

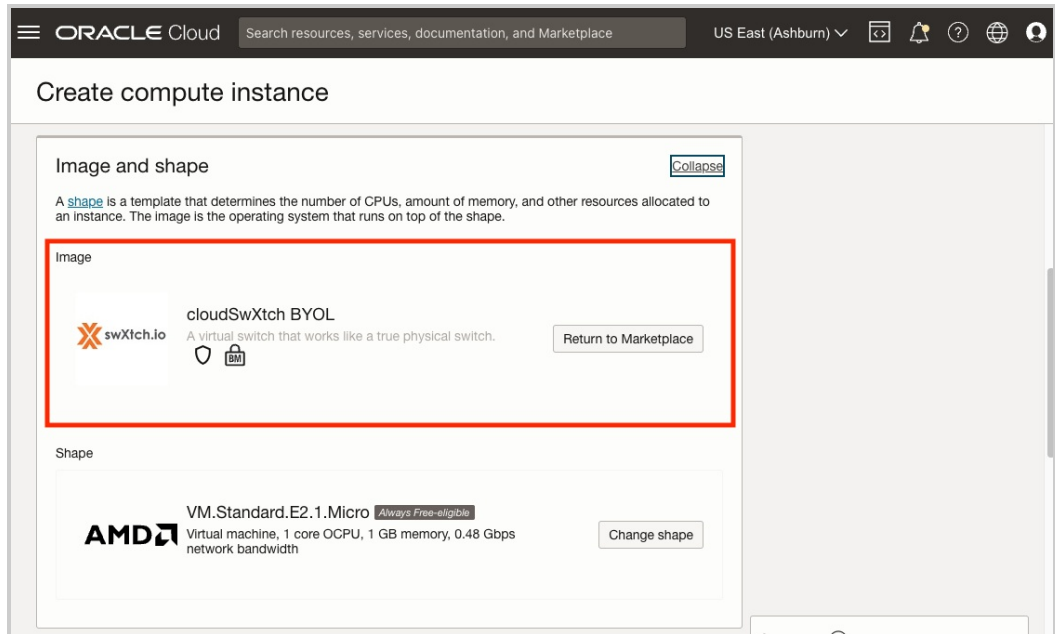
1. Give your **Compute Instance** a unique name.
2. Confirm that your desired **Compartment** is populated.
3. **Optional:** Edit selections for **Placement** and **Security**. This is dependent on a user's specific needs.



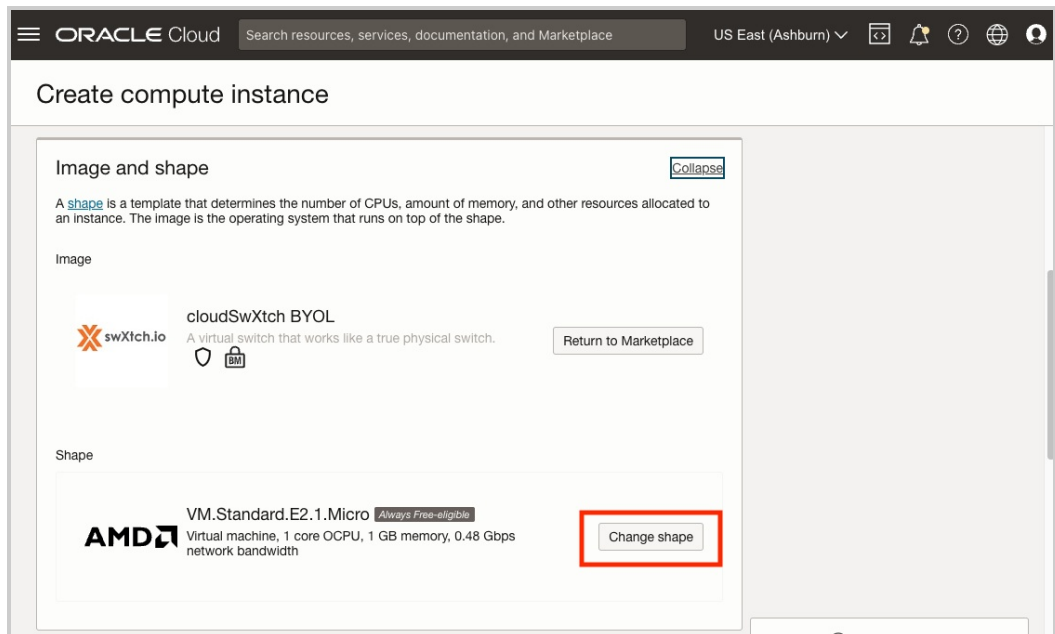
4. Select the **Edit** button for **Image and Shape**.



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



2. Click **Change Shape**.



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

Browse all shapes

prepayments.

Upgrade

Instance type

Virtual machine Always Free-eligible

A virtual machine is an independent computing environment that runs on top of physical bare metal hardware. ✓

Bare metal machine

A bare metal compute instance gives you dedicated physical server access for highest performance and strong isolation.

Shape series

AMD

Flexible OCPU count.
Current generation AMD processors.

Intel

Flexible OCPU count.
Current generation Intel processors. ✓

Ampere

Arm-based processor.

Specialty and previous generation

Always Free, Dense I/O, GPU, HPC, Generic, and earlier generation AMD and Intel standard shapes.

Image: Canonical Ubuntu 20.04

Shape name	OCPU i	Memory (GB)	Security
☒ VM.Standard3.Flex	4 (56 max)	64 (896 max)	

Network bandwidth (Gbps): 4

Maximum VNICS: 4 i

You can customize the number of OCPUs and the amount of memory allocated to a flexible shape.

Select shape

Cancel

©2024 IEX Group, Inc. and its subsidiaries, including swXtch.io, Investors' Exchange LLC and IEX Services LLC. IEX Services LLC, Member SIPC/FINRA. All rights reserved.

105

4. 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](#).

Browse all shapes

Shape name	OCPU ⓘ	Memory (GB)	Security
☒ VM.Standard3.Flex	8 (56 max)	32 (896 max)	🛡️ ^

Network bandwidth (Gbps): 8

Maximum VNICS: 8 ⓘ

You can customize the number of OCPUs and the amount of memory allocated to a flexible shape. The other resources scale proportionately. [Learn more about flexible shapes.](#)

Number of OCPUs

8 ⓘ

1 8 16 24 32 56

Extended OCPU ⓘ

Amount of memory (GB) ⓘ

32 ⓘ

1 32 128 256 384 512 896

Extended memory ⓘ

Burstable

☐

[Burstable instances](#) are virtual machine (VM) instances that provide a baseline level of CPU performance with the ability to burst to a higher level to support occasional increases in usage.

☐ VM.Optimized3.Flex 4 (18 max) 56 (256 max) 🛡️ ✓

1 selected Showing 2 items

Select shape Cancel

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

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

Primary VNIC information

Virtual cloud network: vcn-sa

Subnet: subnet-sa-jh

Launch options: -

Use network security groups to control traffic: No

Assign a public IPv4 address: Yes

DNS record: Yes

Edit

1. **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.
2. Assign a VCN to your **Primary VNIC**.

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

ORACLE Cloud Search resources, services, documentation, and Marketplace US East

Create compute instance

Primary VNIC information [Collapse](#)

A [virtual network interface card \(VNIC\)](#) connects your instance to a [virtual cloud network \(VCN\)](#) and endpoints in and outside the VCN. Having a public IP address is required to make this instance accessible from the internet.

VNIC name *Optional*

Primary network

☒ Select existing virtual cloud network ☐ Create new virtual cloud network ☐ Enter subnet OCID

VCN in SA [\(Change compartment\)](#)

vcn-sa

Subnet

An IP address from a public subnet and an [internet gateway](#) on the VCN are required to make this instance accessible from the internet.

☒ Select existing subnet ☐ Create new public subnet

Subnet in SA [\(Change compartment\)](#)

subnet-sa-ctrl (regional)

4. Click on **Show advanced options**.

Primary VNIC IP addresses

Private IPv4 address

☒ Automatically assign private IPv4 address ☐ Manually assign private IPv4 address

Public IPv4 address

☒ Automatically assign public IPv4 address

If you're not sure whether you need a public IP address, you can always assign one later.

IPv6 addresses

☐ Assign IPv6 addresses from subnet prefixes

You can only assign one IPv6 address per subnet prefix at first instance creation. Subnets can have more than one IPv6 prefix.

i The selected VCN and subnet combination does not support IPv6 addresses. You must enable IPv6 addressing on the VCN and subnet before you can assign IPv6 addresses to this instance.

[Show advanced options](#)

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

[Hide advanced options](#)

Advanced options VCN tags Subnet tags

☐ Use network security groups to control traffic ⓘ

DNS record

☒ Assign a private DNS record ☐ Do not assign a private DNS record

Hostname *Optional*

No spaces. Only letters, numbers, and hyphens. 63 characters max.

Fully qualified domain name: <hostname>.sactrl.vcn07241005.oraclevcn.com

Launch options

☐ Let Oracle Cloud Infrastructure choose the best networking type
Allow Oracle Cloud Infrastructure to choose the [networking type](#), depending on the instance shape and operating system image.

☐ Paravirtualized networking
For general purpose workloads such as enterprise applications, microservices, and small databases.

☒ **Hardware-assisted (SR-IOV) networking**
For low-latency workloads such as video streaming, real-time applications, and large or clustered databases. Does not support live migration.

6. Add an **SSH key**.

Add SSH keys

Generate an [SSH key pair](#) to connect to the instance using a Secure Shell (SSH) connection, or upload a public key that you already have.

☒ Generate a key pair for me ☐ Upload public key files (.pub) ☐ Paste public keys ☐ No SSH keys

Download the private key so that you can connect to the instance using SSH. It will not be shown again.

[Save private key](#) [Save public 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.

2. Select Attached VNICs under Resources.

Resources

Metrics

Quick actions

Attached block volumes

Attached VNICs

Boot volume

Console connection

Run command

Attached VNICs

A [virtual network interface card \(VNIC\)](#) attaches an instance to a subnet within a VCN and is required for connectivity with other endpoints.

Create VNIC

Name	Subnet or VLAN ⁽ⁱ⁾	State	FQDN ⁽ⁱ⁾	VLAN tag	MAC address
docs-test-instance (Primary VNIC)	Subnet - subnet-sa-jh	Attached	docs-test-... Show Copy	2726	02:00:17:14:06:D6

Showing 1 item < 1 of 1 >

3. Click Create VNIC.

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

Create VNIC

VNIC name *Optional*

Virtual cloud network in SA [\(Change compartment\)](#)

vcn-sa

Network

Normal setup: subnet

The typical choice when adding a VNIC to an instance. ✓

Advanced setup: VLAN

Only for experienced users who have purchased the Oracle Cloud VMware Solution.

Subnet in SA [\(Change compartment\)](#)

subnet-sa-ctrl (regional)

☐ Use network security groups to control traffic (optional) ⁽ⁱ⁾

☐ Skip source/destination check ⁽ⁱ⁾

VNIC IP addresses

Private IPv4 address

Save changes [Cancel](#)

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

Resources

Metrics

Quick actions

Attached block volumes

Attached VNICs

Boot volume

Console connection

Run command

Work requests

OS Management

Attached VNICs

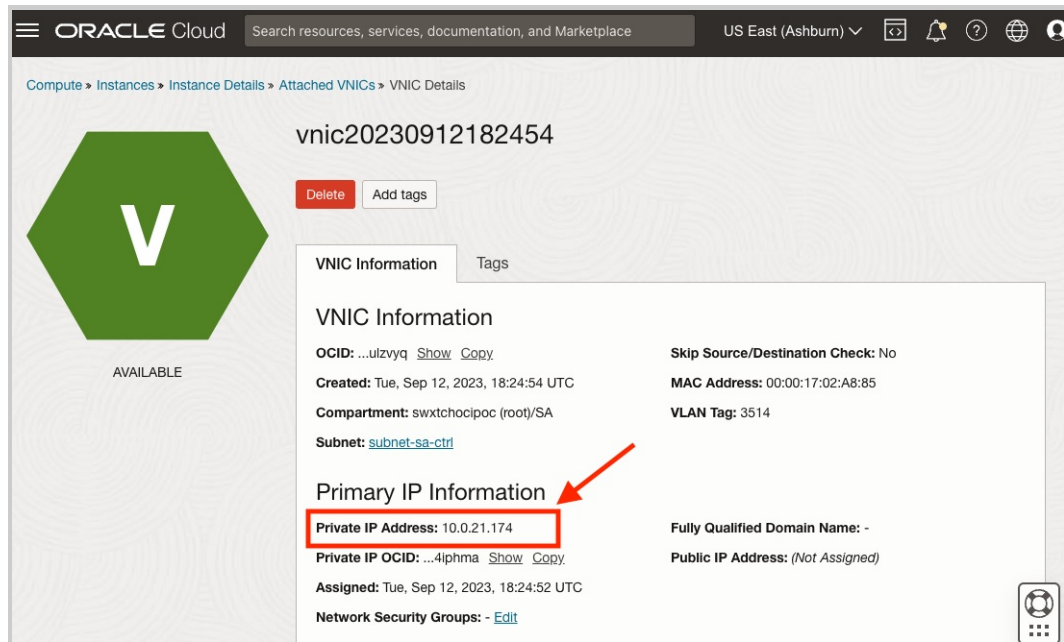
A [virtual network interface card \(VNIC\)](#) attaches an instance to a subnet within a VCN and is required for connectivity with other endpoints.

Create VNIC

Name	Subnet or VLAN ⁽ⁱ⁾	State	FQDN ⁽ⁱ⁾	VLAN tag	MAC address
docs-test-instance (Primary VNIC)	Subnet - subnet-sa-jh	Attached	docs-test-... Show Copy	2726	02:00:17:14:06:D6
vnic20230912182454	Subnet - subnet-sa-ctrl	Attached	-	3514	00:00:17:02:A8:85

Showing 2 items < 1 of 1 >

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



Oracle Cloud console showing the details of a secondary VNIC (vnic20230912182454). The VNIC is in an AVAILABLE state. The Primary IP Information section is highlighted, showing the Private IP Address: 10.0.21.174. A red arrow points to this value.

6. Log into your Instance with **cloudSwXtch** installed.

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
    ethernet:
        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.

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](#).

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](#)

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
<pre>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</pre>	

3. Use the following command to get the latest version of cloudSwXtch. **The latest release is 2.1.0.**

Shell

Bash	Copy
<pre>ver="v2.1.0"</pre>	

4. Enter the following to download that version.

Shell

Bash	Copy
<pre>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</pre>	

5. Execute the installer.

Shell

Bash	Copy
<pre>sudo ./image_install.sh \${ver}</pre>	

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.

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 v2.1.2
```

Upgrading cloudSwXtch via the xNIC

1. Connect to any VM where an xNIC is running.
2. Run the following command:

Shell

Bash

Copy

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

Example:

Bash

Copy

```
swx update -v v2.1.2 --ip 10.5.1.6
```

Note: The <desired version> includes a "v" before the version number (e.g. v2.0.34).

Upgrading cloudSwXtch and xNICs

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

Installing cloudSwXtch Bridge

WHAT TO EXPECT

There are currently 2 types of cloudSwXtch Bridges: Type 1 and Type 2. It is suggested for users to use cloudSwXtch Bridge 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:

- **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.

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 requires 2 NICs with the non-primary NIC as a Mellanox card.
 - **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 Ubuntu 20.04 with Kernel 5.15 or greater OR RHEL8/CentOS8 with Kernel of 5.11 or greater.
 - Minimum of 4 cores, 8GB RAM.
 - Hard drive: Minimum 20GB, Recommended: 40GB
- 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 V2. (See Install cloudSwXtch Bridge V1 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
<pre>sudo apt-get install linux-generic-hwe-20.04</pre>	
Bash	Copy
<pre>sudo apt update && sudo apt full-upgrade</pre>	

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	Copy
<pre>uname -r</pre>	

The output will read:

Bash	Copy
<pre>5.15.0-1023</pre>	

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.

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
<pre>ping <swxtch-instance-ip or instance-name></pre>	

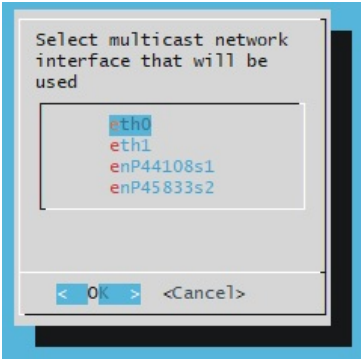
1. **If the ping fails to find the cloudSwXtch instance by name**, try pinging the IP address of the cloudSwXtch instance. If the IP works, they use the IP address in place of the name in all future commands. This can happen if the default DNS settings are changed for the Virtual Network.

3. Run the cloudSwXtch bridge installer script:

Text

None	Copy
<pre>sudo sh -c "curl -s http://<swxtch-ip>/services/install/swxtch-bridge-install.sh bash -s -- -v 2"</pre>	

1. 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:

None	Copy
<pre>sudo journalctl -u swxtch-bridge2 -f -n 100</pre>	

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
<pre>sudo systemctl stop swxtch-bridge2.service</pre>	

START

Bash	Copy
<pre>sudo systemctl start swxtch-bridge2.service</pre>	

RESTART

Bash	Copy
<pre>sudo systemctl restart swxtch-bridge2.service</pre>	

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

Your cloudSwXtch Bridge Type 2 instance should now be uninstalled.

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 -V1

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
<pre>sudo ping <swxtch-instance-name></pre>	

1. **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.

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

Text

Bash	Copy
<pre>curl http://<swxtch-instance-name>/services/install/swxtch-bridge-install.sh bash -s -- -k -v 1</pre>	

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.

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 - <i>Minimum Kernel Version 4.18</i> • Amazon Linux 2023 - <i>Minimum Kernel Version 5.14</i> • Centos 8 Minimum - <i>Minimum Kernel Version 4.18</i> • Oracle Linux 8 - <i>Minimum Kernel Version 4.18</i> • RHEL 8.8 - <i>Minimum Kernel Version 4.18</i> • RHEL 9.2 - <i>Minimum Kernel Version 5.14</i> • Rocky Linux 8 - <i>Minimum Kernel Version 4.18</i> • Rocky Linux 9 - <i>Minimum Kernel Version 5.14</i> • Ubuntu 20.04 - <i>Minimum Kernel Version 5.5</i> • Ubuntu 22.04 - <i>Minimum Kernel Version 6.2</i>	• 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)

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.

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

Copy

```
ping <switch-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 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://<swxtch-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://<swxtch-ip>/services/install/swxtch-xnic-install.sh
chmod +x swxtch-xnic-install.sh
./swxtch-xnic-install.sh
```

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:

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
```

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
```

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

A successful install is shown below.

```
testadmin@agent-101:~$ curl http://10.2.128.10/services/install/swxtch-xnic-install.sh | bash
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left    Speed
100 21432  0 21432  0  0  5232k  0  --:--:-- --:--:-- --:--:--  5232k
xNIC installer detected ubuntu 20.04. Installing xNIC version 1.
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left    Speed
100 7308k 100 7308k  0  0  113M  0  --:--:-- --:--:-- --:--:--  113M
Selecting previously unselected package swxtch-xnic.
(Reading database ... 141654 files and directories currently installed.)
Preparing to unpack swxtch-xnic_1.0.0_ubuntu20.04_amd64.deb ...
Unpacking swxtch-xnic (1.0.0) ...
Setting up swxtch-xnic (1.0.0) ...
/var/opt/swxtch/swxtch-xnic.conf has been updated.
Service swxtch-xnic.service started
testadmin@DSd-agent-101:~$
```

IF THE INSTALL FAILS:

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.

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.

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](#).

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

- 3. The uninstall script will remove Linux xNIC.

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

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.

A full list of arguments is detailed below:

```
$ ./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
```


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.

Bash

Copy

```
ping <switch-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:

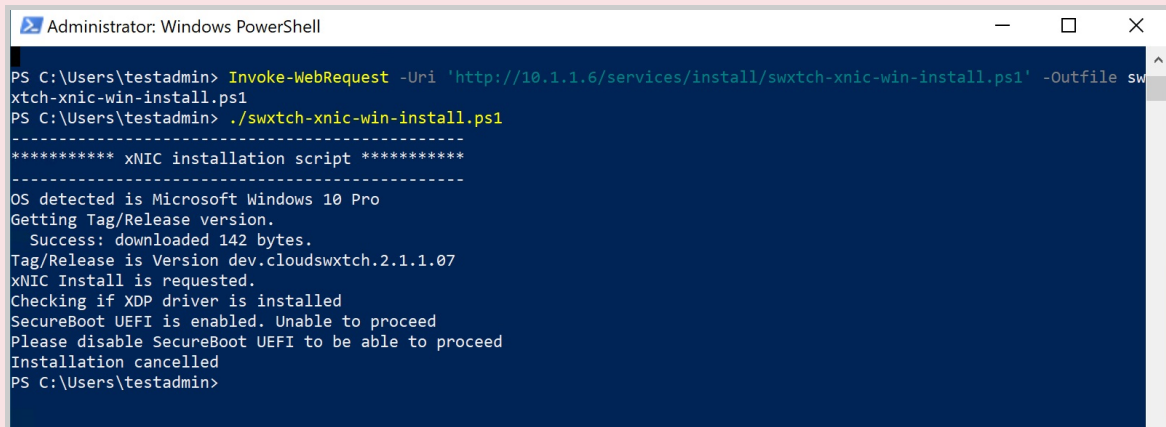
Bash

Copy

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

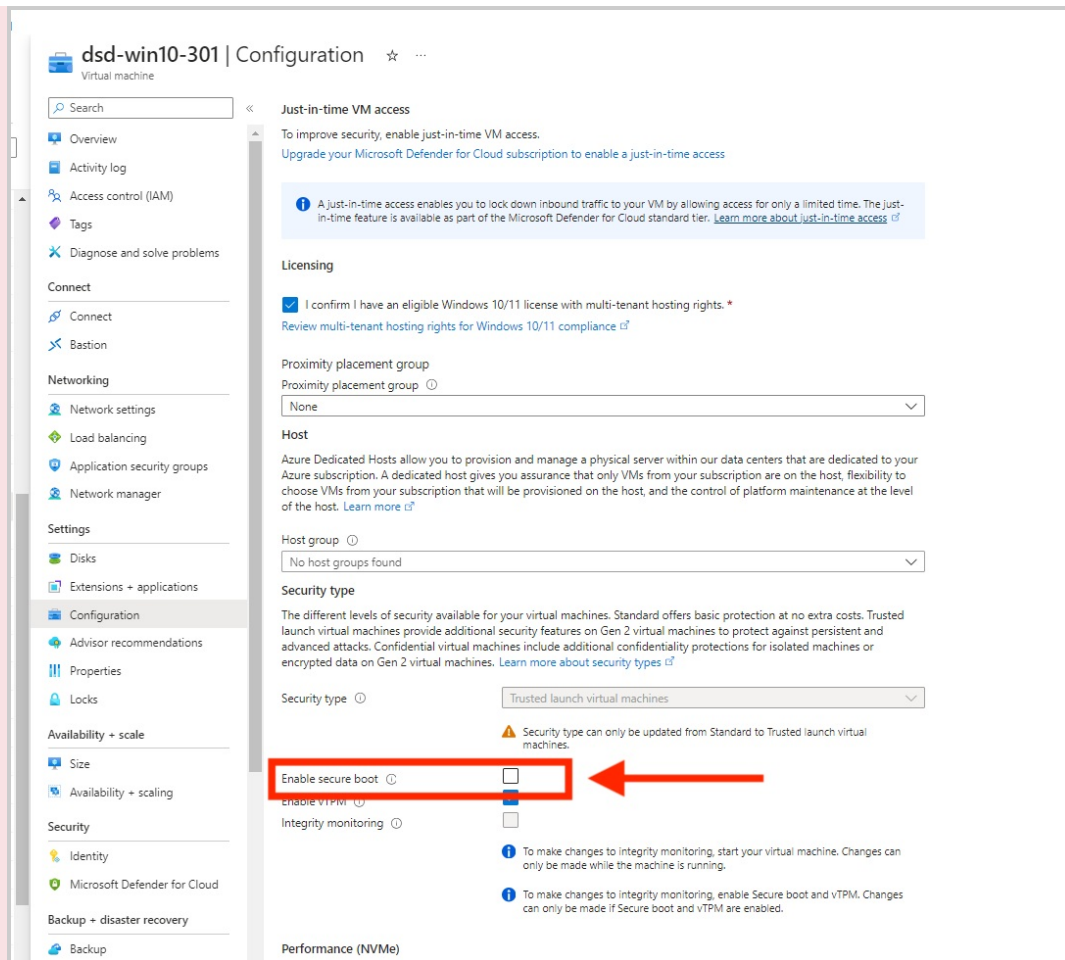
Secure Boot Error

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.



```
Administrator: Windows PowerShell
PS C:\Users\testadmin> Invoke-WebRequest -Uri 'http://10.1.1.6/services/install/swxtch-xnic-win-install.ps1' -Outfile sw
xtch-xnic-win-install.ps1
PS C:\Users\testadmin> ./swxtch-xnic-win-install.ps1
***** xNIC installation script *****
-----
OS detected is Microsoft Windows 10 Pro
Getting Tag/Release version.
  Success: downloaded 142 bytes.
Tag/Release is Version dev.cloudswxtch.2.1.1.07
xNIC Install is requested.
Checking if XDP driver is installed
SecureBoot UEFI is enabled. Unable to proceed
Please disable SecureBoot UEFI to be able to proceed
Installation cancelled
PS C:\Users\testadmin>
```

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.

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
<pre> 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 </pre>	

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.

Getting the interface index for `ctrl_interface` and `data_interface`

To get the interface index, you can run the following command:

Bash	Copy
<pre>get-netipconfiguration</pre>	

A sample of the response will be returned with the `InterfaceIndex` (***) listed for both:

Bash	Copy
<pre> 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 </pre>	

7. The installer script will install a Windows service called swXtchNIC:

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\testadmin> ./swxtch-xnic-win-install.ps1 -k
This PowerShell script manages the xNIC installation for Windows
Version to install is 1
Running on Windows Server 2019 Datacenter
- Getting Swxtch-xNIC installer for Windows
Saving swxtch-xnic.conf file backup named swxtch-xnic-bup.conf
- Removing installed swXtch xNIC
- Installing Swxtch-xNIC (swxtch-xnic-1.0.0-x86_64.msi)
- Installing Microsoft Visual C++ Redistributable
- Adding xNIC tools folder to the User PATH
- Configuring xNIC
Loading swxtch-xnic.conf backup
...Control Subnet = 10.2.128.0, Prefix = 22
...Data Subnet = 10.2.192.0, Prefix = 22
...Control Interface index = 134
...Data Interface index = 47
- Checking if XDP driver is installed
- Starting the xNIC 1 service...
Installation finished
PS C:\Users\testadmin>
```

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 <swxtch-hostname>
```

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

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.

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
<pre>.\swxtch-xnic-win-install.ps1 -u</pre>	

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

3. Run the script. Please use the appropriate command for your version. **Note:** xNIC Type 2 is the default.

None	Copy
<pre>./swxtch-xnic-win-install.ps1</pre>	

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

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.


```
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
```

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:

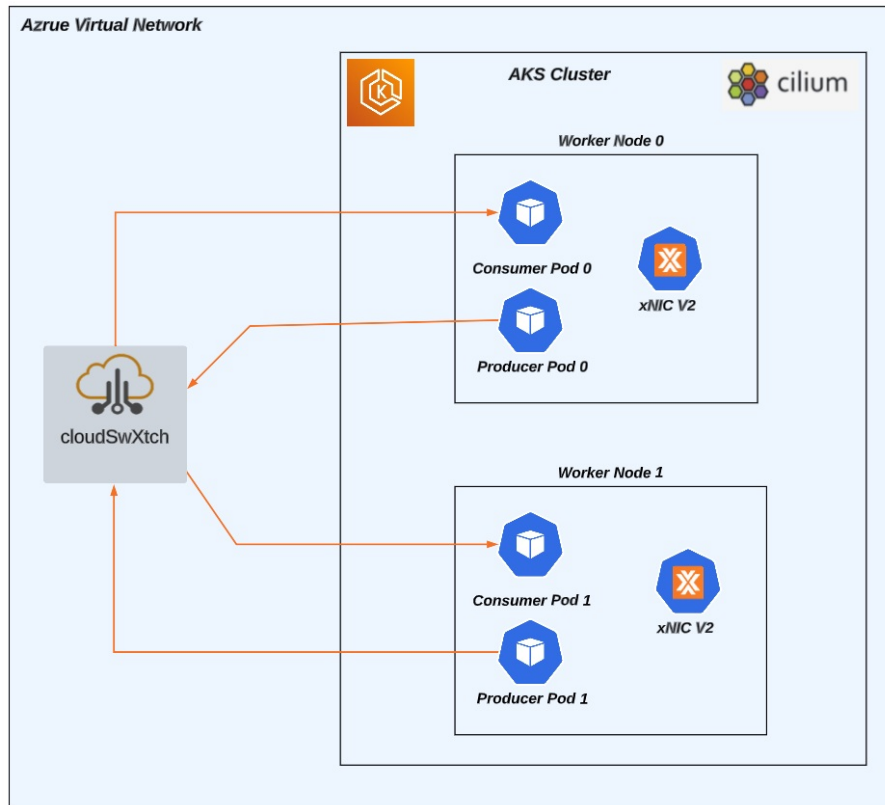
1. Create a Kubernetes Cluster (AKS with Cilium/Kubernetes, EKS or GKE)
2. [Install xNIC on K8s Cluster](#)
3. [Test xNIC on K8s](#)

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.



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

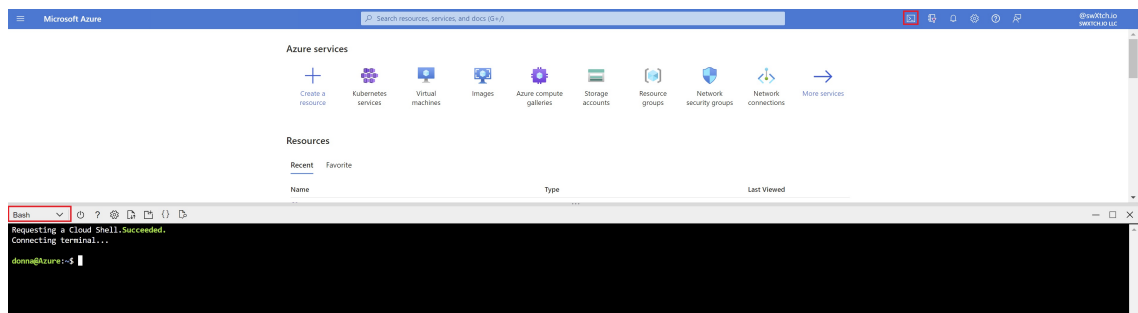
If you haven't already, please create a Kubernetes Cluster. This is a prerequisite before installing the xNIC.

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.
3. Open cloudShell as Bash.



4. Paste in the following commands, replacing the <cloudSwXtch_IP> with your cloudSwXtch's Ctrl IP address.

Bash	Copy
<pre>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</pre>	

5. Run one of the following scripts:

With Internet Access:

Bash	Copy
<pre>./xnic_ds_installer.sh</pre>	

Without Internet Access:

Bash	Copy
<pre>./xnic_ds_installer.sh -ag</pre>	

An example of a successful install without INTERNET access is shown below:

Bash	Copy
<pre>\$./xnic_ds_installer.sh -ag [i] Detected Cloud: AZURE [i] Cilium Installation detected [i] Setting CNI to CILIUM... ##### ##### 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</pre>	

```

=====
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/swtch-xnic created
Done!

===== Completed! =====

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/swtch-xnic -f

```

- Run the following script to view **kubectl** logs in the Bash window in Azure:

Bash	Copy
<pre>kubectl logs -n kube-system daemonsets/swtch-xnic -f</pre>	

- 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
<pre> user@Azure:~\$ kubectl get pods -l app=swtch-xnic -n kube-system NAME READY STATUS RESTARTS AGE swtch-xnic-fc58t 1/1 Running 0 11d swtch-xnic-kn9hg 1/1 Running 0 11d </pre>	

8. Sign into your cloudSwXtch and enter in the following command to see the new instances in swXtch-top.

Bash	Copy
<pre>sudo /swxtch/swxtch-top dashboard --switch localhost</pre>	

Restarting xNIC services

To restart xNIC services for K8s, run the following command:

Bash	Copy
<pre>kubectl rollout restart ds/swxtch-xnic -n kube-system</pre>	

Managing Multicast Traffic

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
<pre># 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</pre>	

Alternatively, users can deny outgoing multicast to specific groups:

General Syntax:

Bash	Copy
<pre># 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</pre>	

Example: denying outgoing traffic to multicast group `239.0.0.1` :

Bash	Copy
<pre>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</pre>	

Allow outgoing multicast to a specific group(s) - Deny any other

Bash	Copy
<pre># 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</pre>	

Example: allowing outgoing traffic **ONLY** to multicast group **239.0.0.1** :

Bash	Copy
<pre>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</pre>	

Incoming (Traffic entering the Pod)

To deny **ALL** incoming multicast, use the following command:

Specific syntax:

Bash	Copy
<pre># 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</pre>	

Alternatively, users can deny incoming multicast for a specific group(s)

General syntax:

Bash	Copy
<pre># 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</pre>	

Example: denying incoming multicast traffic to multicast group 239.0.0.1 :

Bash	Copy
<pre>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</pre>	

In addition, users can specify allowing incoming multicast by a specific group(s) while denying any other:

General syntax:

Bash	Copy
<pre># 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</pre>	

Example: allowing incoming traffic ONLY to multicast group 239.0.0.1 :

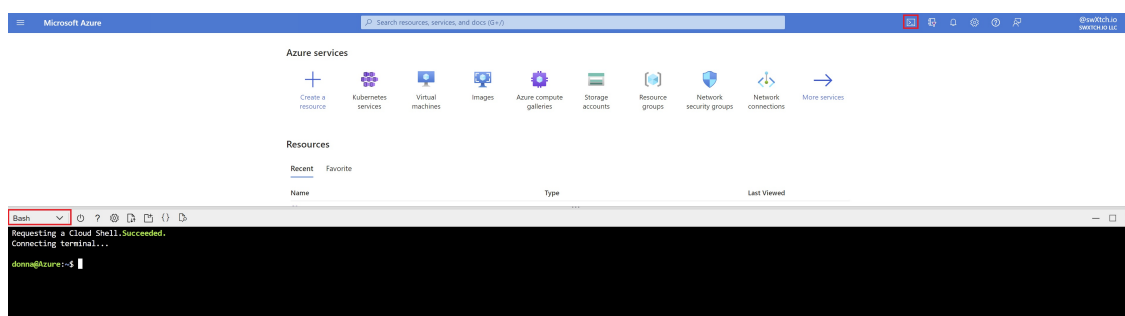
Bash	Copy
<pre>tc qdisc add dev eth0 ingress tc qdisc add dev eth0 parent ffff: protocol ip u32 match ip dst 224.0.0.0/4 action drop tc filter add dev eth0 parent ffff: protocol ip u32 match ip dst 239.0.0.1/32 action ok</pre>	

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 `swtch-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.



3. Enter in the following command to get the pod name:

Bash	Copy
<pre>kubectl get pods -l app=swxtch-xnic -n kube-system</pre>	

Example:

Bash	Copy
<pre>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-qn9hg 1/1 Running 0 11d</pre>	

4. Enter in the following command, replacing **Pod** with the pod name:

Bash	Copy
<pre>kubectl exec -it pod/swxtch-xnic-name -n kube-system -- bash</pre>	

Example:

Bash	Copy
<pre>user@Azure:~\$ kubectl exec -it pod/swxtch-xnic-qn9hg -n kube-system -- bash root@aks-nodepool1-23164585-vmss00000A:/</pre>	

You can now enter in commands similar to any VM Node, such as "ip a" or "sudo swxtch-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:

1. Ensure the pods are running using this command:

Bash	Copy
<pre>kubectl get pods -l app=swxtch-xnic -n kube-system</pre>	

Example:

Bash	Copy
<pre>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</pre>	

2. Run this command to sign into the pod:

Bash	Copy
<pre>kubectl exec -it pod/<swxtch-xnic-name> -n kube-system -- bash</pre>	

Example:

Bash	Copy
<pre>user@Azure:~\$ kubectl exec -it pod/swxtch-xnic-kn9hg -n kube-system -- bash root@aks-nodepool11-23164585-vmss00000A:/#</pre>	

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:

Bash	Copy
<pre>cat /var/opt/swxtch/swxtch-xnic.conf</pre>	

Exiting the Pod

To exit the pod, enter in the following command:

Bash	Copy
<code>exit</code>	

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.

1. Get a list of all K8s Contexts by using the following command:

Bash	Copy
<code>kubectl config get-contexts</code>	

Example in Azure:

Bash	Copy
<pre>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</pre>	

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	Copy
<code>kubectl config use-context cilium-sample2</code>	

3. Re-run the **get-context** command:

Bash	Copy
<pre>kubectl config get-contexts</pre>	

Example in Azure:

Bash	Copy																																												
<pre>user@Azure:~\$ kubectl config get-contexts</pre> <table border="1"><thead><tr><th>CURRENT</th><th>NAME</th><th>CLUSTER</th><th>AUTHINFO</th></tr></thead><tbody><tr><td></td><td>cilium-sample</td><td>cilium-sample</td><td></td></tr><tr><td></td><td colspan="3">clusterUser_saDevNetwork_cilium-sample</td></tr><tr><td></td><td>cilium-sample-200</td><td>cilium-sample-200</td><td>clusterUser_test-donna-200-</td></tr><tr><td></td><td colspan="3">rg_cilium-sample-200</td></tr><tr><td>*</td><td>cilium-sample2</td><td>cilium-sample2</td><td></td></tr><tr><td></td><td colspan="3">clusterUser_saDevNetwork_cilium-sample2</td></tr><tr><td></td><td>cilium-sample300</td><td>cilium-sample300</td><td>clusterUser_test-donna-300-</td></tr><tr><td></td><td colspan="3">rg_cilium-sample300</td></tr><tr><td></td><td>dsd-k8-cluster-100</td><td>dsd-k8-cluster-100</td><td></td></tr><tr><td></td><td colspan="3">clusterUser_saDevNetwork_dsd-k8-cluster-100</td></tr></tbody></table>		CURRENT	NAME	CLUSTER	AUTHINFO		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		
CURRENT	NAME	CLUSTER	AUTHINFO																																										
	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.

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.

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.

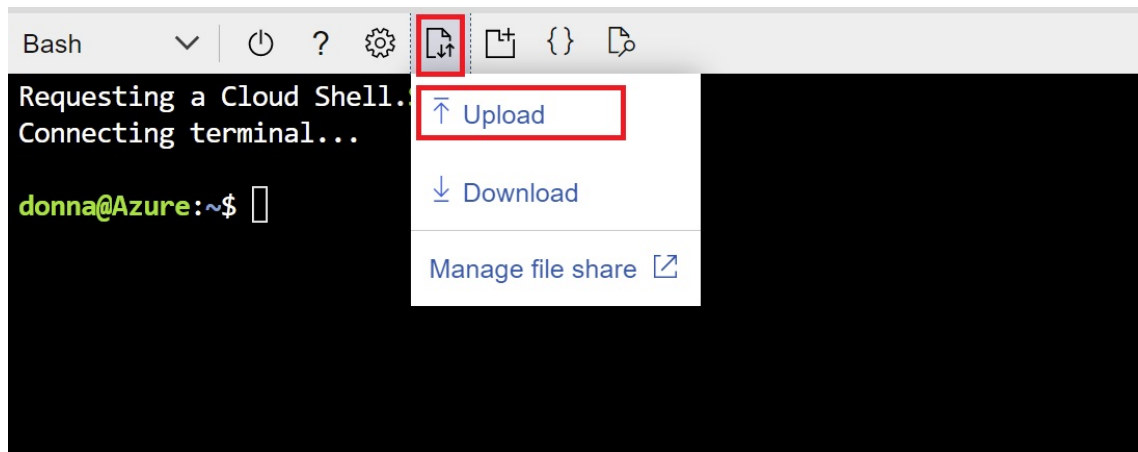
STEP ONE: Create A Consumer

1. Create a **TestConsumer.yaml** file using the example below.

Replace the **XNIC_SWXTCH_ADDR** with the cloudSwXtch control address.

Bash	Copy
<pre>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"]</pre>	

2. Upload the file into the Azure CloudShell.



STEP TWO: Create a Producer

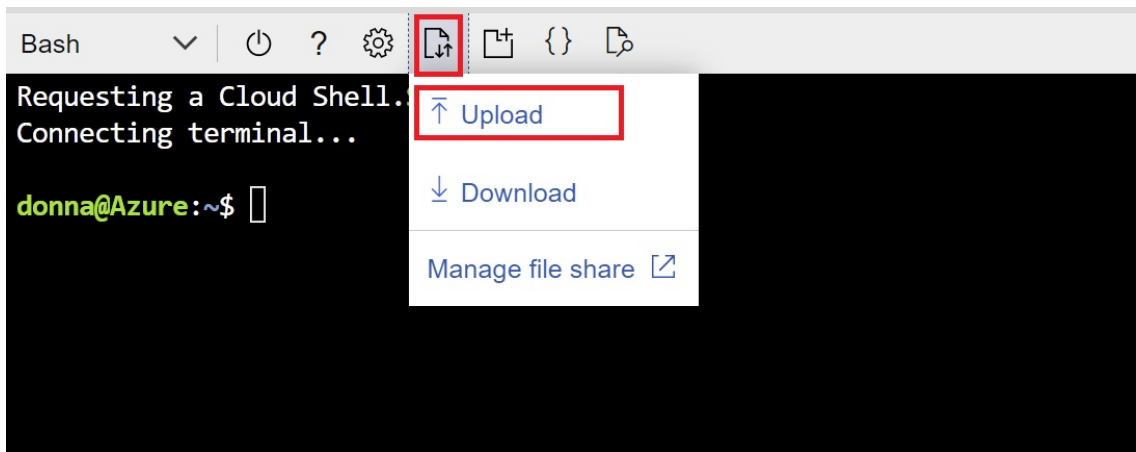
1. Create a **TestProducer.yaml** file using the example below.

Replace **XNIC_SWXTCH_ADDR** with the cloudSwXtch control address.

Shell

Bash	Copy
<pre>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"]</pre>	

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
<pre>kubectl create -f TestProducer.yaml</pre>	

- Run the consumer by running this command in your preferred cloud's cloudShell bash window. Wait for the cursor to return to know it is fully created.

Bash	Copy
<pre>kubectl create -f TestConsumer.yaml</pre>	

Validate they are running using this command:

Bash	Copy
<pre>kubectl get pods -o wide -A</pre>	

Below is an example in Azure showing the consumer-a and producer-a running:

Bash						Copy
<pre>donna@Azure:~\$ kubectl get pods -o wide -A</pre>						
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-569f6fff56-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:

Bash						Copy
<pre>kubectl logs pods/producer-a -f</pre>						

- Log into your cloudSwXtch and run this command:

Bash						Copy
<pre>sudo /swxtch/swxtch-top dashboard --swxtch localhost</pre>						

1. swXtch-top should show the producer and the consumer. This may take a minute to completely show all metrics.

Information - dev.3645.v2

dsd-core-100	dev.3645.v2(dsd-100-core-azure-April-5-2033)	Auth. Type	License File	Marketplace
SubscriptionId	b10209ad-ad22-4c26-8aef-be93b2f0bb58	Max Bandwidth	2000 Mbps	
ResourceGroupName	TEST-DONNA	Max Clients	10	
SwxtchId	2369a922-410a-40bf-8e6e-630efe0ee70a			
Status	OK			

Totals

Producers	99 pps	(136.7K bps)	Consumers	99 pps	(137.6K bps)
Bridge RX	0 pps	(0 bps)	Bridge TX	0 pps	(0 bps)
Mesh RX	0 pps	(0 bps)	Mesh TX	0 pps	(0 bps)
Switch RX	100 pps	(139.1K bps)	Switch TX	99 pps	(137.7K bps)

xNIC clients

Name	ip	Version (XNIC)	RX pps	RX bps	TX pps	TX bps	HA
aks-nodepool1-23351669-vmss000004	10.2.128.95	dev.3645.v2 (v2)	99	137.6K	0	0	N
aks-nodepool1-23351669-vmss000005	10.2.128.96	dev.3645.v2 (v2)	0	0	99	136.7K	N

- 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
<pre>kubectl delete -f TestConsumer.yaml</pre>	

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
<pre>donna@Azure:~\$ kubectl get pods -o wide -A</pre>	
<pre>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></pre>	

- 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
<pre>kubectl delete -f TestProducer.yaml</pre>	

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
<pre>donna@Azure:~\$ kubectl get pods -o wide -A</pre>	
<pre>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></pre>	

Now that the system is validated using swXtch.io, you can test with your K8s application.

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
<pre>kubectl rollout restart ds swxtch-xnic -n kube-system</pre>	

1. This will restart the Kubernetes swxtch-xnic daemon set and update the version of the xNIC to match that of the cloudSwXtch.

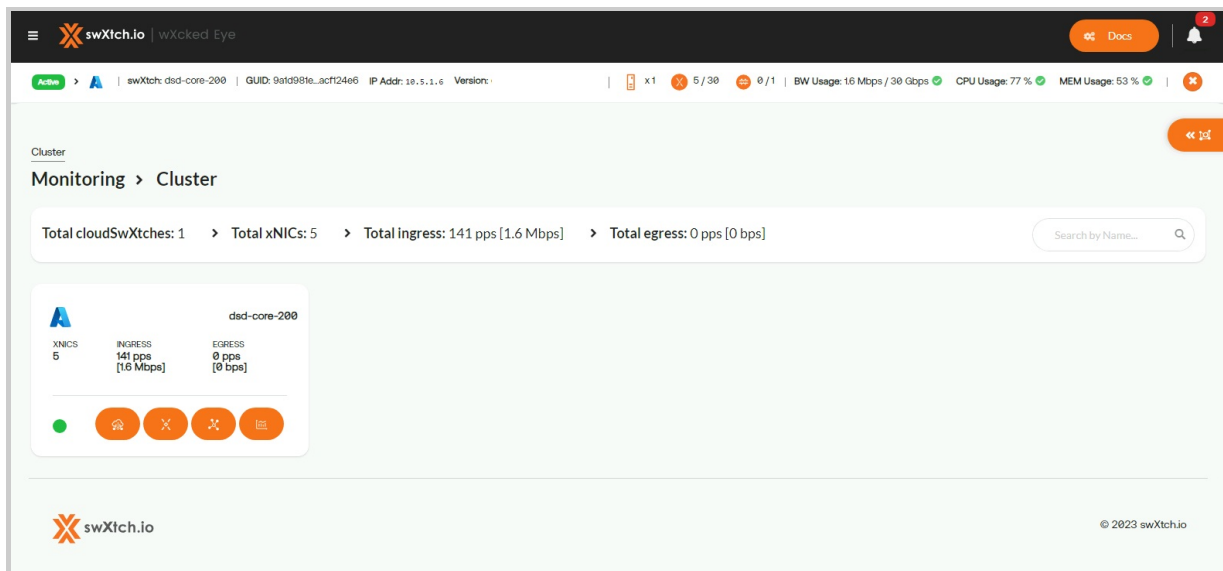
Using wXcked Eye for cloudSwXtch

WHAT TO EXPECT

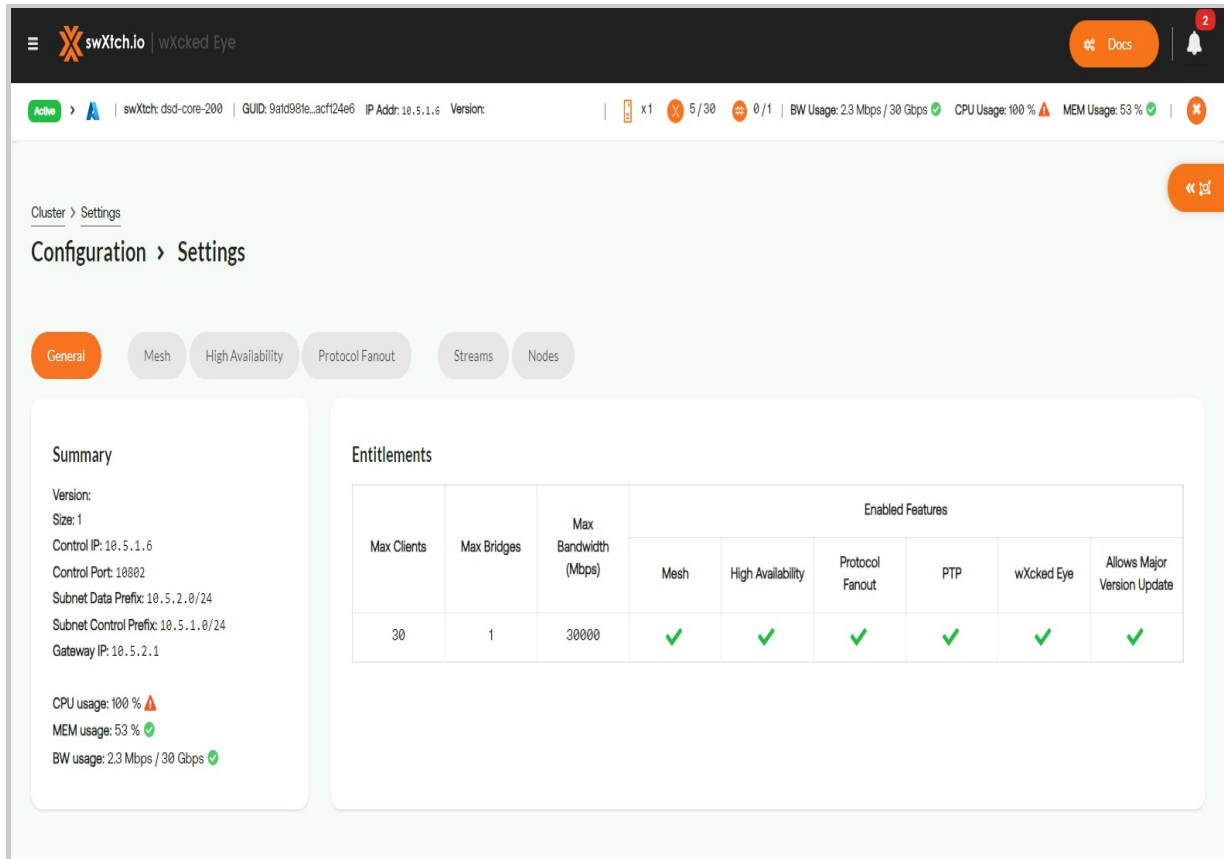
In this article, you will learn more about wXcked Eye and the benefits of using it to configure and monitor your cloudSwXtch environment.

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.



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
<pre><swxtch-ip-address>/wxckedeye/</pre>	

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

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](#).

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

The screenshot shows the wXcked Eye web interface. On the left, a dark sidebar contains a navigation menu. The 'Monitoring' section is highlighted with a red rectangular box and includes the following items: Cluster, swXtch, xNICs, Topology, Protocol Fanout, Timing Nodes, and Support. Below this, the 'Configuration' section includes Settings and Notifications. The main content area has a top status bar showing GUID, IP Address, Version, and various resource usage metrics (CPU, MEM, BW). Below the status bar, there are tabs for Availability, Protocol Fanout, Streams, and Nodes. The 'Entitlements' section contains a table with system capabilities.

Max Clients	Max Bridges	Max Bandwidth (Mbps)	Enabled Features					
			Mesh	High Availability	Protocol Fanout	PTP	wXcked Eye	Allows Major Version Update
30	1	30000	✓	✓	✓	✓	✓	✓

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.

wXcked Eye Cluster Page

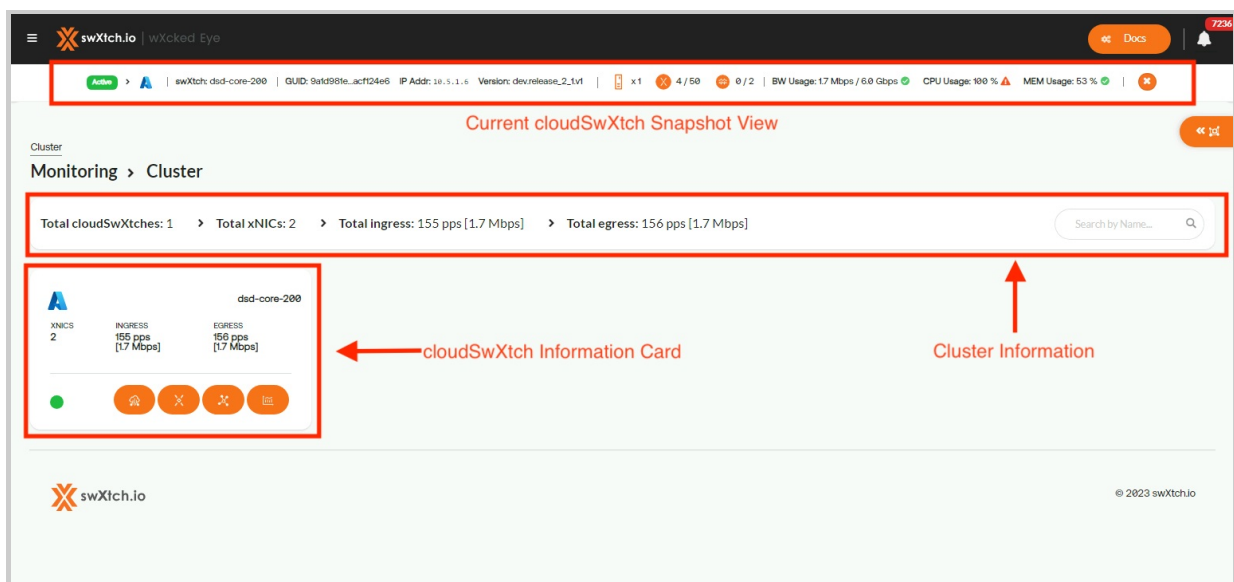
WHAT TO EXPECT

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.

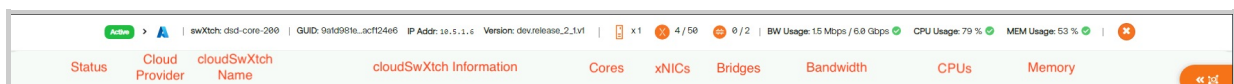
In this section, we will walk you through the Cluster page and how it improves navigation between cloudSwXtches.

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.

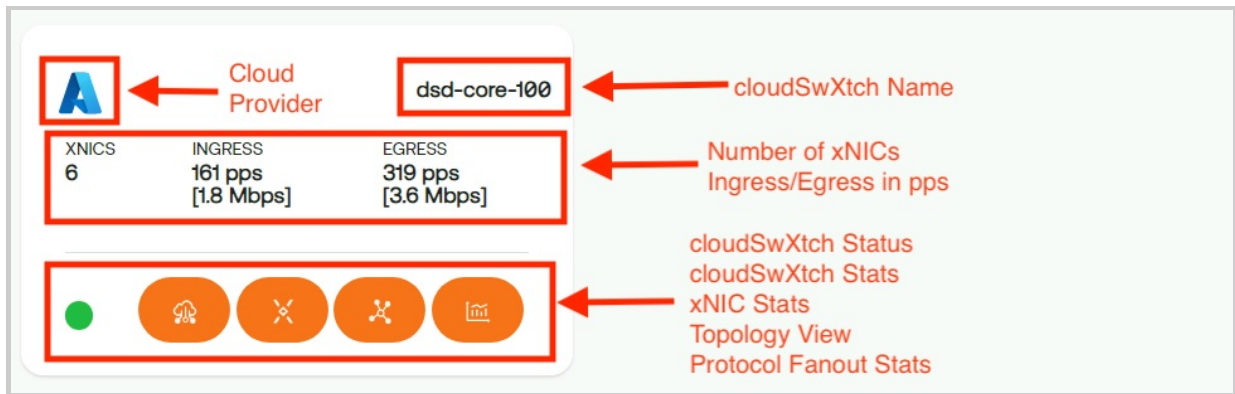


At the top of each page in wXcked Eye, the Current cloudSwXtch Snapshot banner will display relevant information regarding the cloudSwXtch a user is currently on.



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.

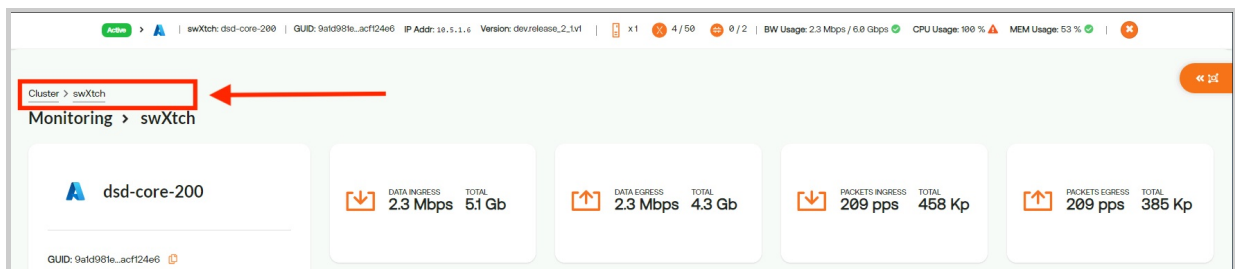
Navigating using cloudSwXtch Information Card



In each cloudSwXtch Information card, users will find four buttons linking you to additional information and metrics within wXcked Eye. This includes:

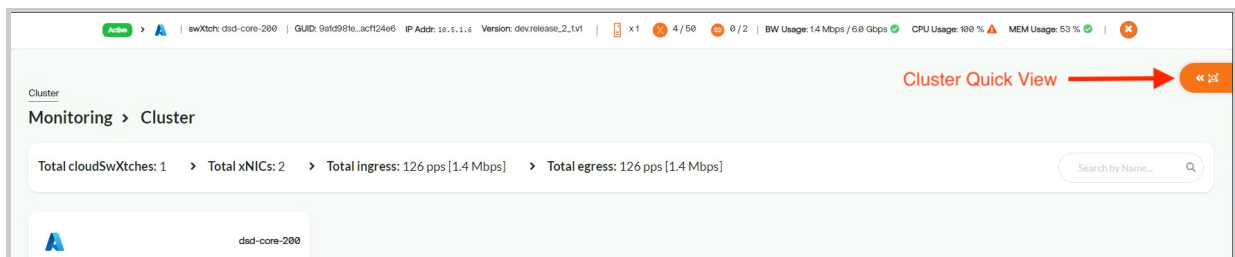
- [cloudSwXtch Stats](#)
- [xNIC Stats](#)
- [Topology](#)
- [Protocol Fanout Stats](#)

A user can return to the cluster view by selecting "Cluster" in the breadcrumb trail at the top of the page.

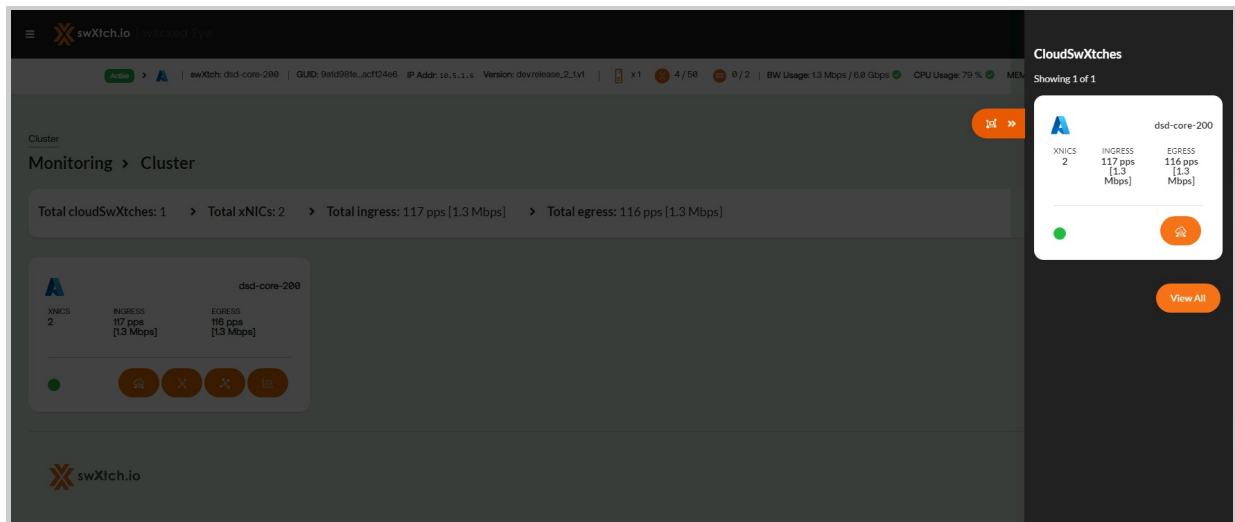


Cluster Quick View Panel

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.



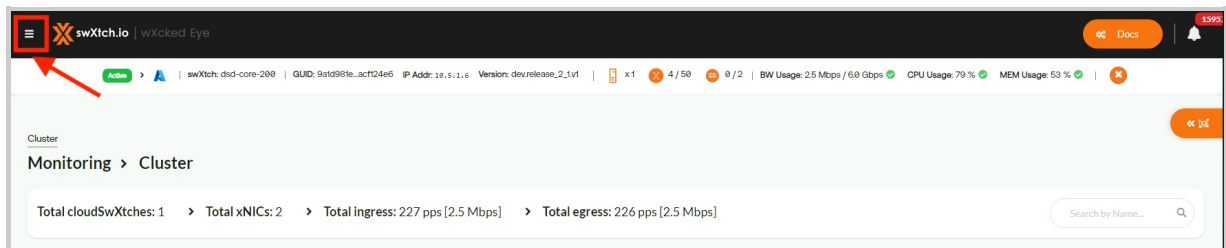
wXcked Eye cloudSwXtch Page

WHAT TO EXPECT

In this article, you will learn how to view the performance metrics for your cloudSwXtch and the xNICs associated with it.

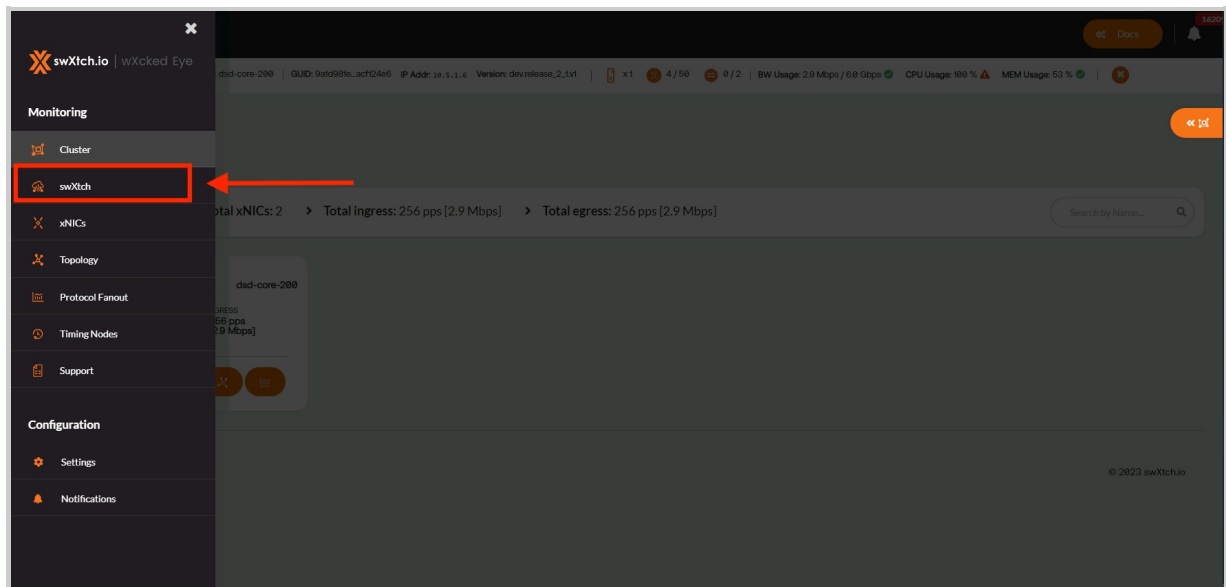
Locating the wXcked Eye cloudSwXtch Page

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.

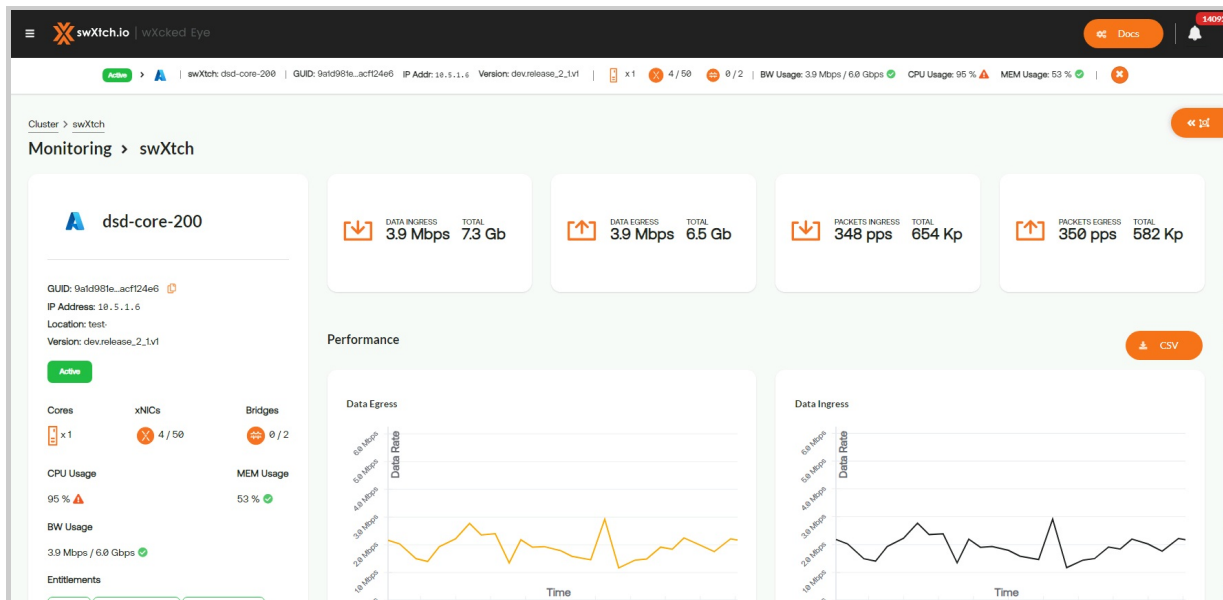


From there, select "swXtch" under "Monitoring."

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

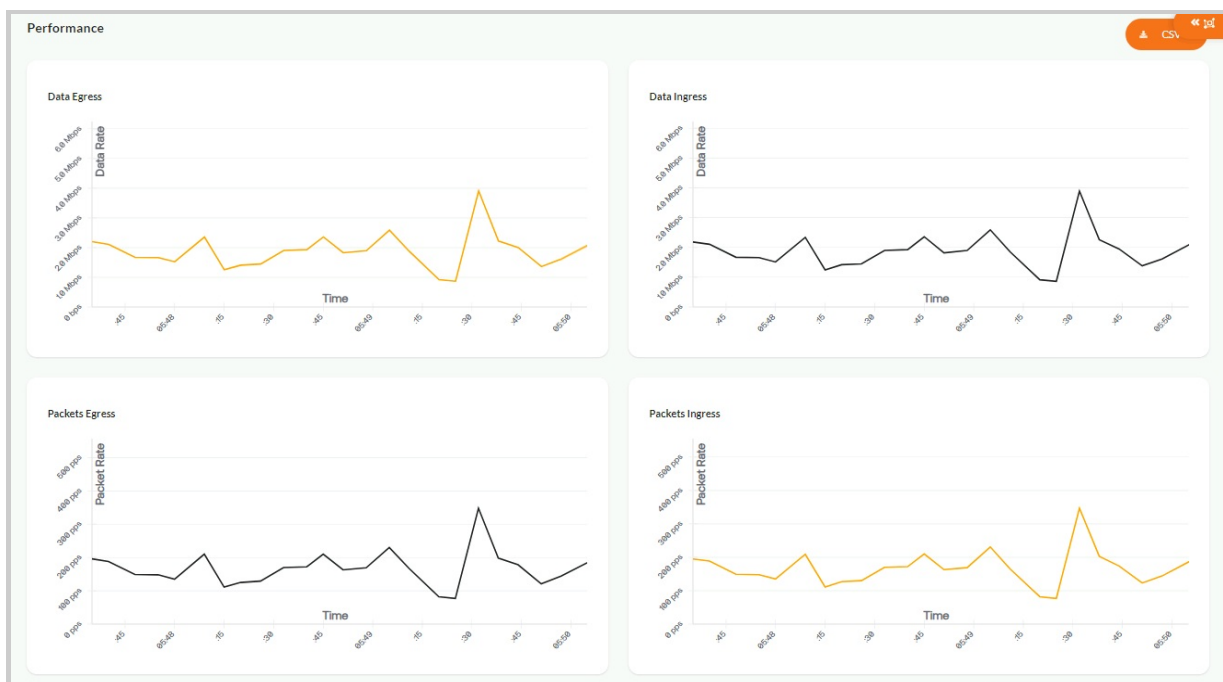


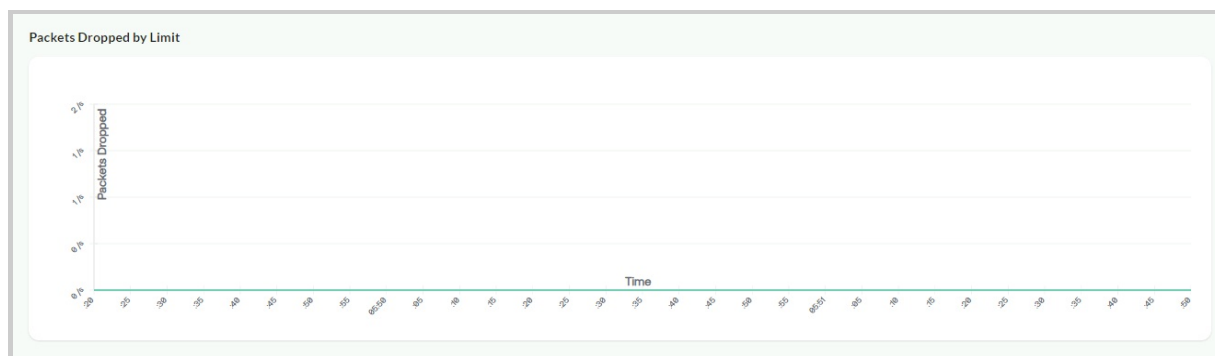
cloudSwXtch Key Performance Metrics

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.





cloudSwXtch Information Panel

 dsd-core-200

GUID: 9a1d981e...acf124e6 

IP Address: 10.5.1.6

Location: test-donna-200-rg/test-donna-200-rg

Version: dev.release_2.1.v1

Active

Cores

 x 1

xNICs

 4 / 50

Bridges

 0 / 2

CPU Usage

77 % 

MEM Usage

53 % 

BW Usage

1.7 Mbps / 6.0 Gbps 

Entitlements

MESH

PROTOCOL FANOUT

HIGH AVAILABILITY

PTP

BRIDGE

WICKED EYE

MAJOR VERSION UPDATE

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

xNICs (2) Search by Name...

Name	Version	Ingress 44 Gb [4.0 Mp]	Egress 9.4 Gb [837 Kp]	Drops 0
▶ DSd-agent-204 		0 bps [0 pps]	11 Mbps [97 pps]	0 /s
▶ DSd-agent-205 		1.2 Mbps [109 pps]	0 bps [0 pps]	0 /s

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.

wXcked Eye xNICs Page

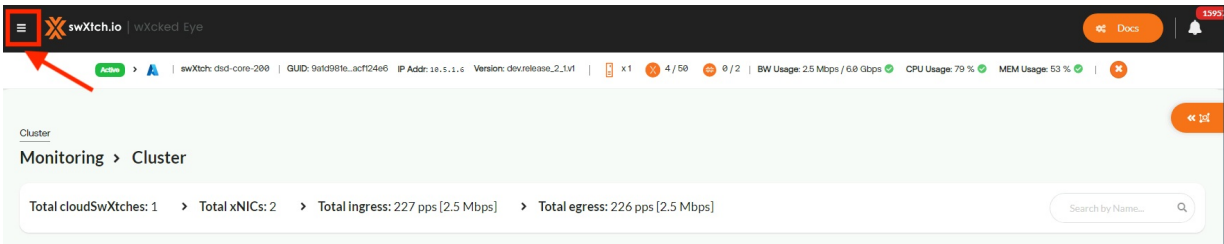
WHAT TO EXPECT

In this article, users will learn how to view performance metrics from the xNICs perspective.

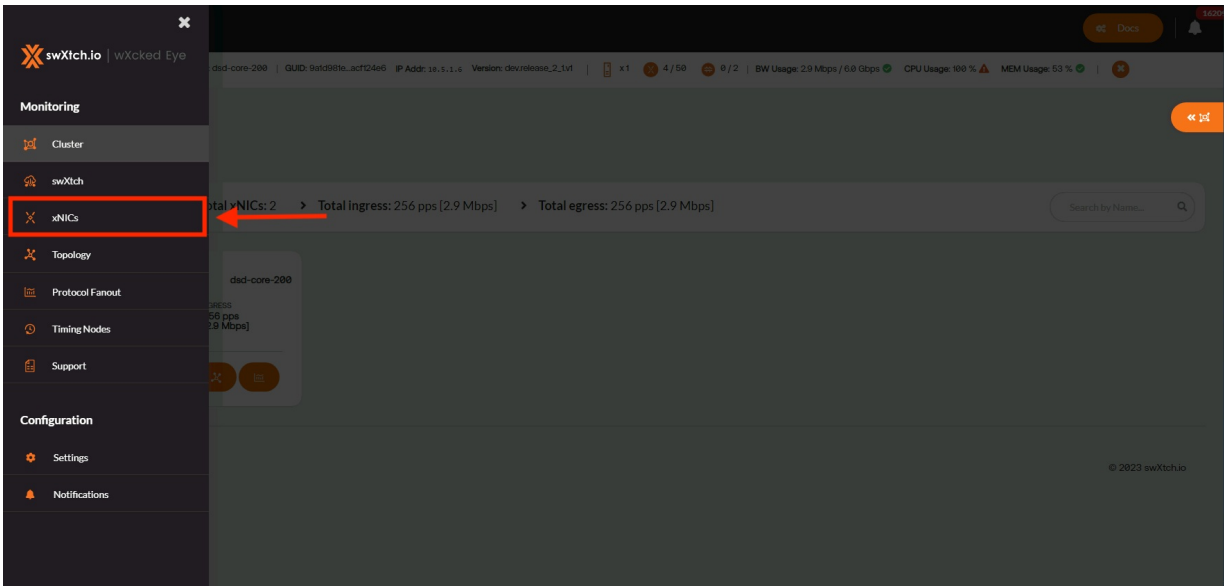
Locating the xNIC View Page

The xNIC page provides users with a look of their cloudSwXtch environment from the xNICs' perspective, breaking down performance at an agent's level.

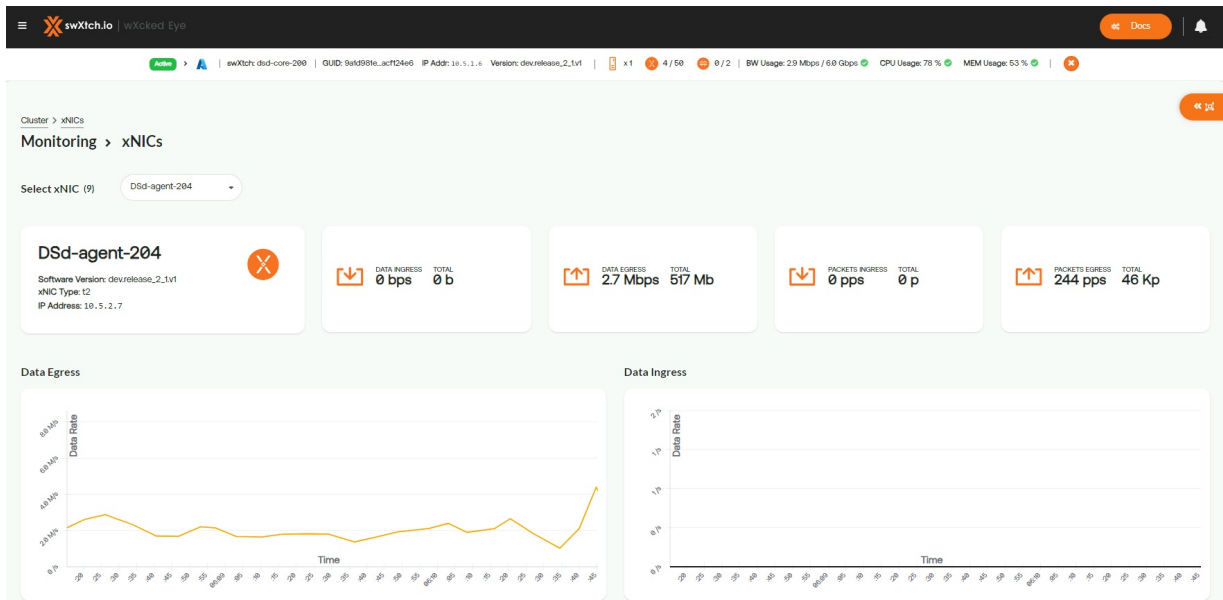
To navigate to the xNIC page, users will need to click on the menu option at the top left hand corner by the swXtch.io logo.



The navigation menu will open, revealing the other wXcked Eye pages. Select xNICs to view the xNIC page.



Navigating the xNIC page



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

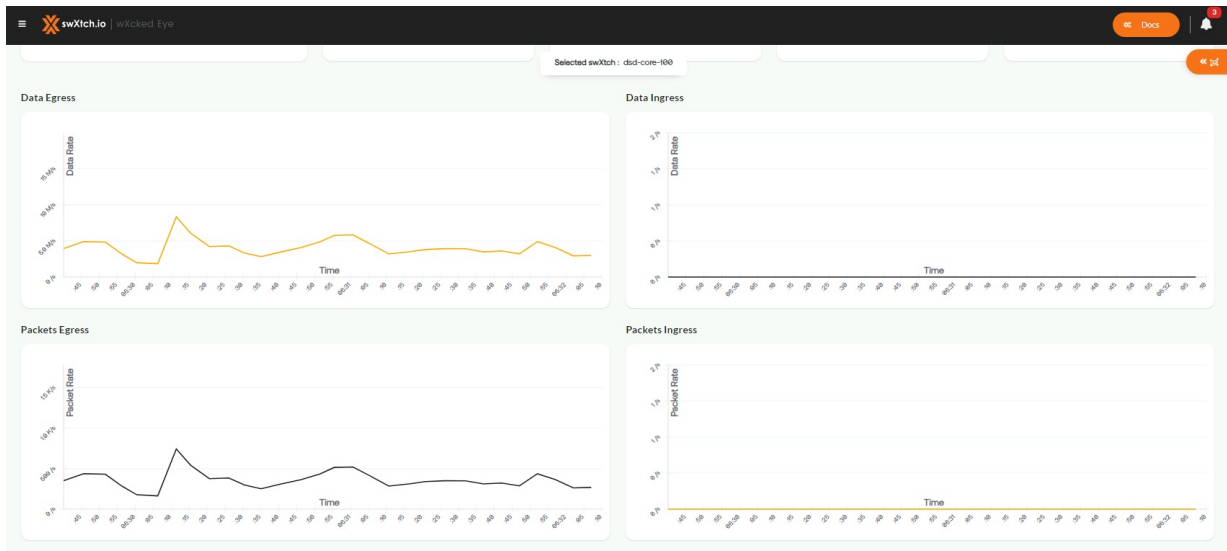
In the example above, one noticeable difference is the inclusion of the Select an xNIC dropdown menu next to "Select xNIC." Here, a user can select an agent they wish to monitor (DSd-agent-204).

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.

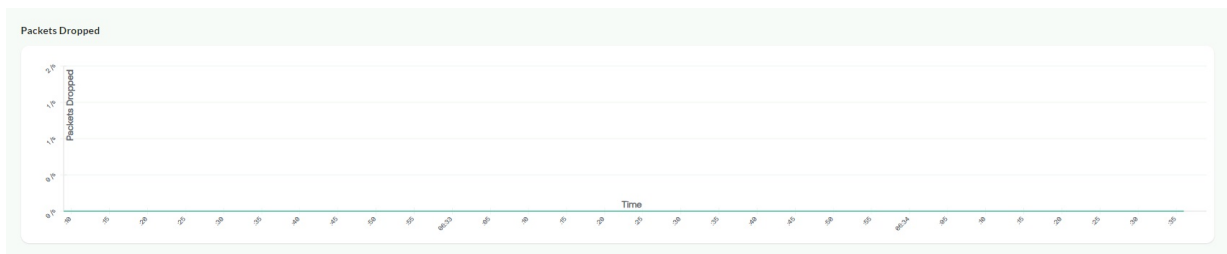
Performance

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.



Performance: 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.

Multicast Groups (5)		
IP Address	Ingress 149 Kb [75 p]	Egress 48 Kb [3.4 Mp]
10.2.131.255	0 bps [0 pps]	0 bps [0 pps]
224.0.0.251	0 bps [0 pps]	0 bps [0 pps]
10.2.195.255	0 bps [0 pps]	0 bps [0 pps]
239.255.255.258	0 bps [0 pps]	0 bps [0 pps]
239.1.1.1	0 bps [0 pps]	6.6 Gbps [574 Kpps]

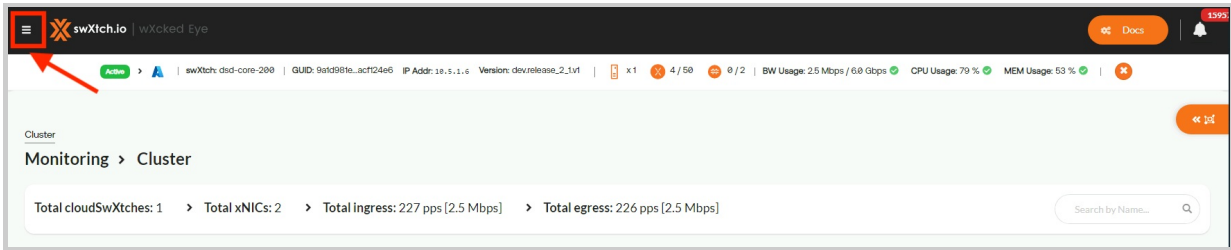
wXcked Eye Topology

WHAT TO EXPECT

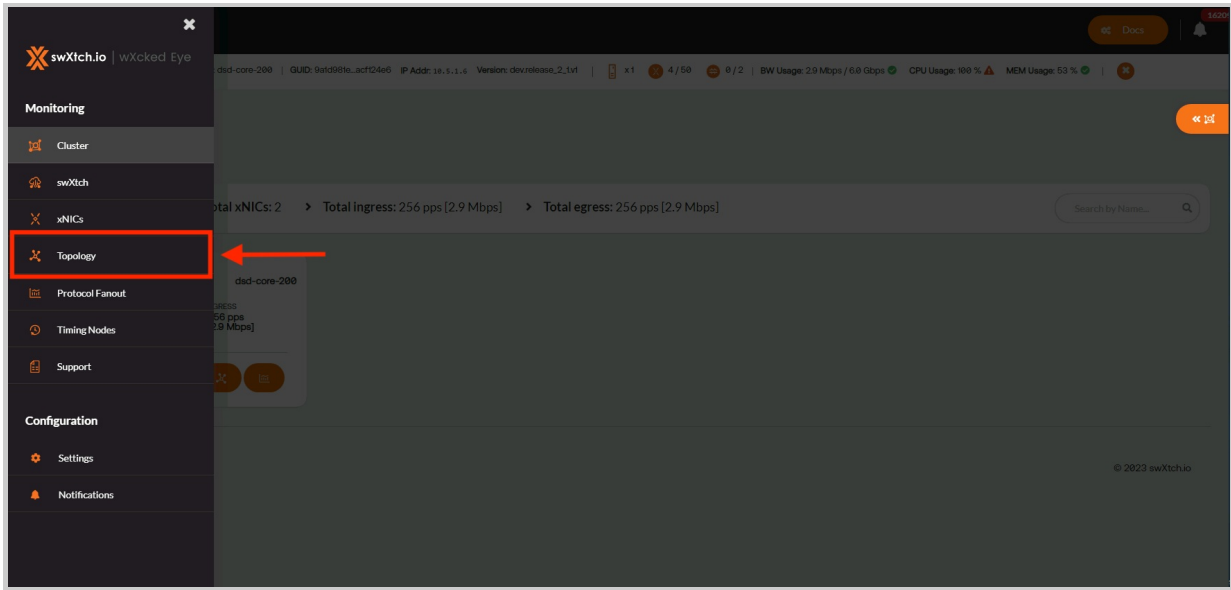
In this article, users will learn how to use the wXcked Eye Topology and reformat it for their needs.

Locating the wXcked Eye Topology

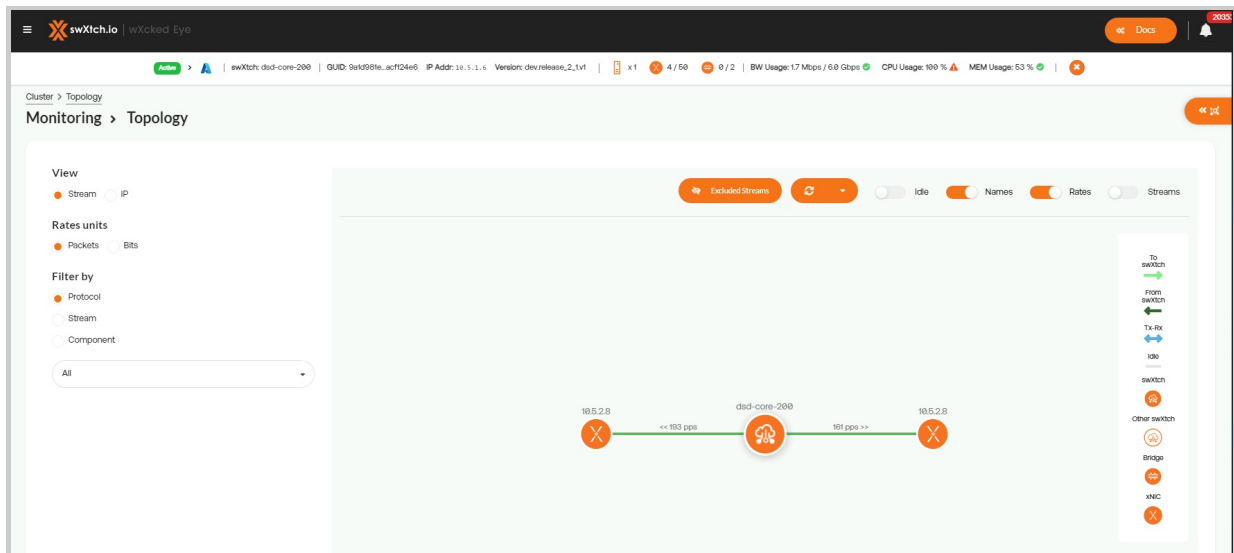
To navigate to the wXcked Eye Topology page, users will need to click on the menu (≡) option at the top right hand corner by the swXtch.io logo.



From there, select "Topology" under "Monitoring."



Using the wXcked Eye Topology



The **wXcked Eye Topology** 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 green lines indicating traffic flowing either to or from it. Next to each line, users will be able to see the flow's direction with the transmission rates (either in pps or bps). The endpoints can be either xNICs, cloudSwXtch Bridges or other cloudSwXtch instances as detailed in the key on the right hand side of the graph.

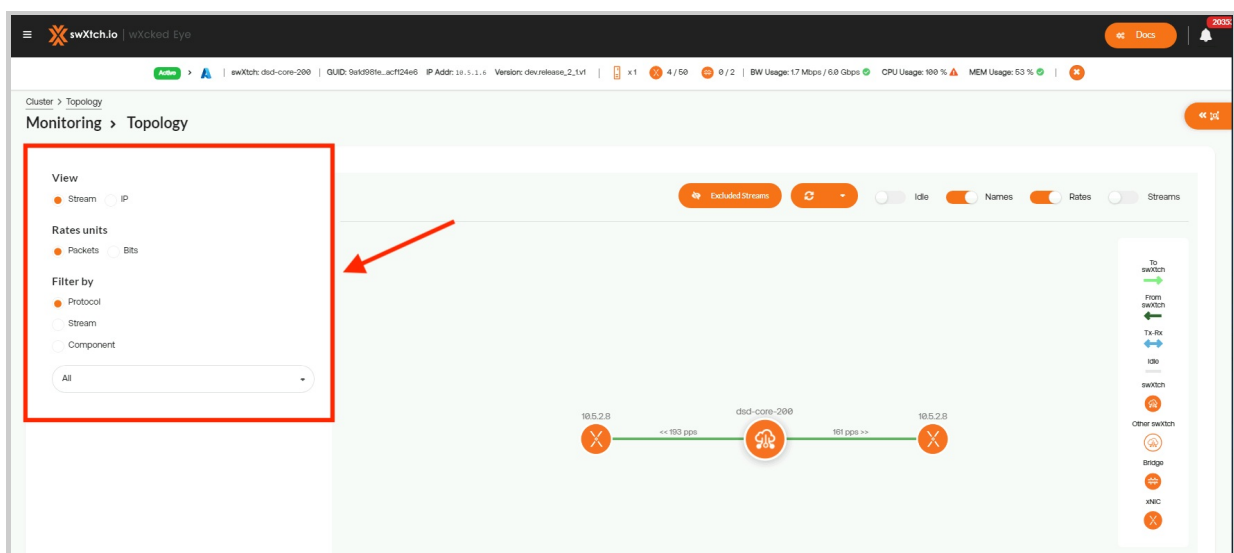
Reformatting the Topology

It is very simple to alter the Topology 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 names/rates on and off. These options are available next to the graph key.

For Rates, users can select between Packets and Bits for their unit.

Users can also filter by components, protocols, and streams. Selecting one of these options will change the list in the dropdown menu.

- **Components** - This will allow users to highlight specific icons like cloudSwXtch, xNICs, UDP, and SRT.
- **Protocol** - Multicast, UDP, SRT Caller, and SRT Listener
- **Stream** - This will allow users to highlight a specific data stream from producer to consumer.



wXcked Eye Protocol Fanout Stats

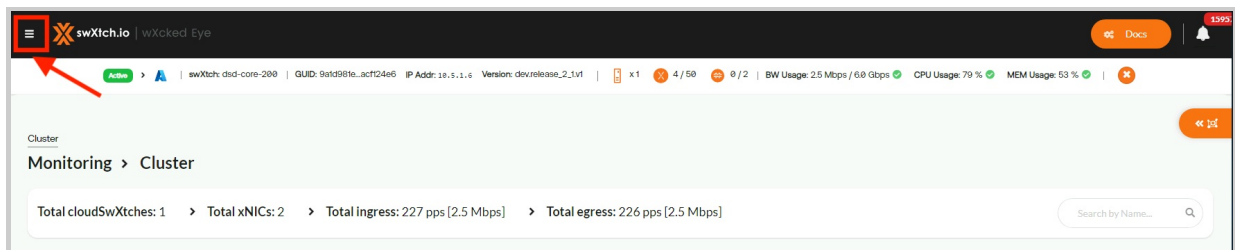
WHAT TO EXPECT

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.

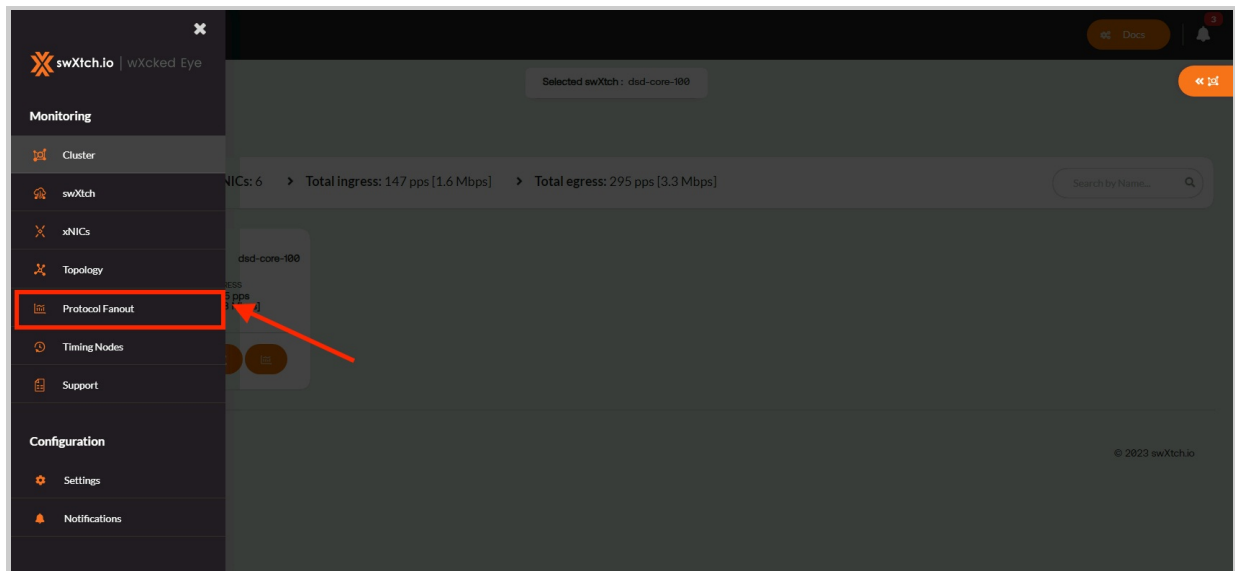
In this section, users will learn how to navigate between different protocols and their adaptors in order to better visualize their packets' movements. **Please note:** This page only shows the stats for protocol fanout configurations. To learn how to configure your cloudSwXtch for Protocol Fanout and 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.



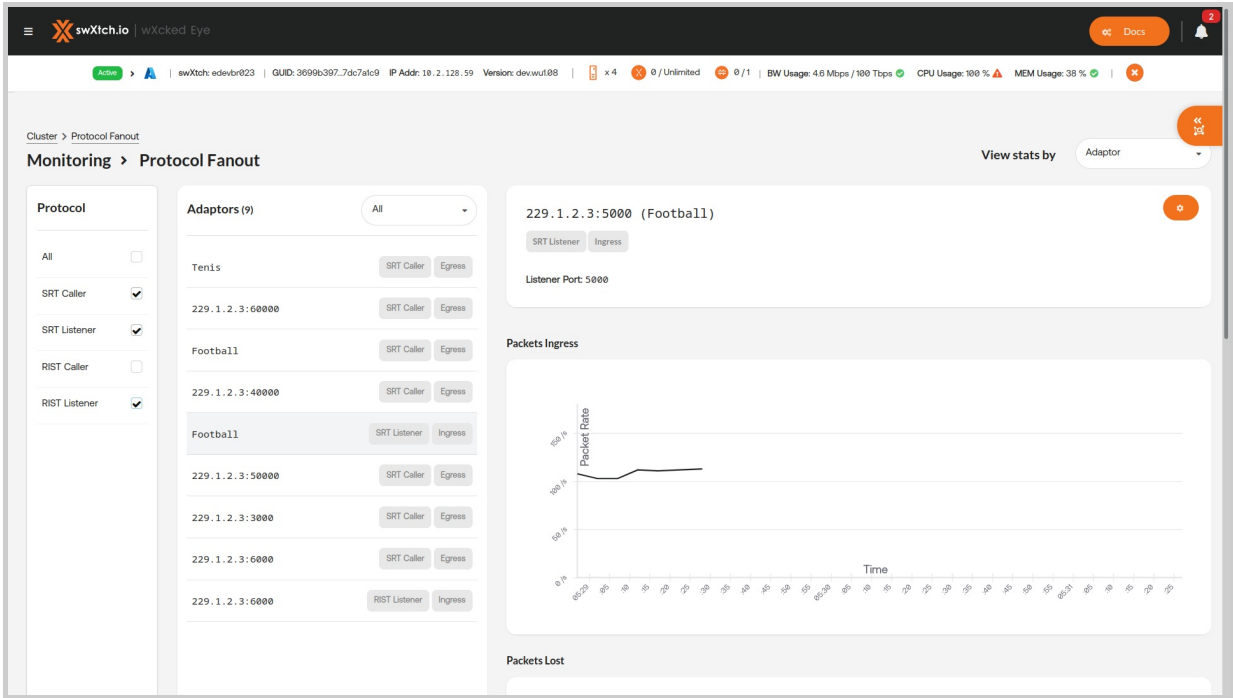
From there, select **Protocol Fanout** under **Monitoring**.



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



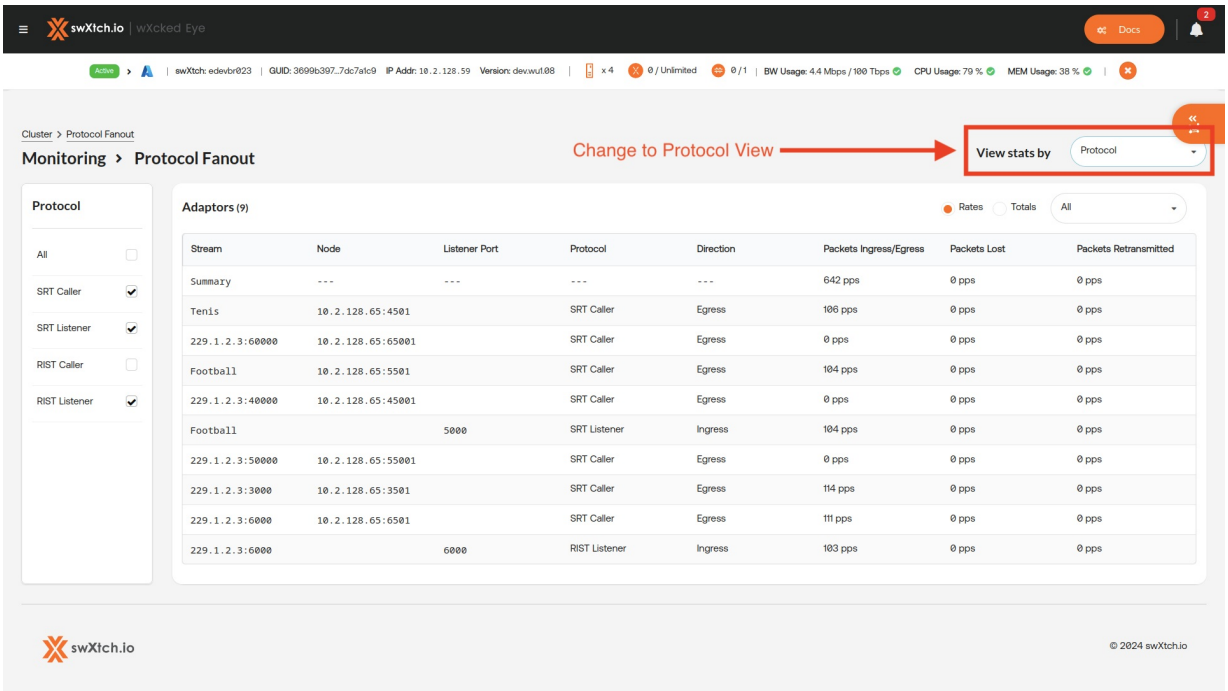
Adaptor View

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



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**.

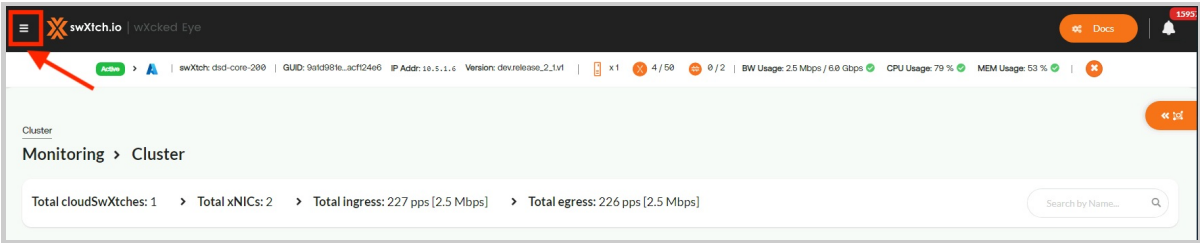
Similar to the Adaptors view, users can group adaptors by protocol. The Protocol panel on the left hand side enables users to filter out certain protocols so that they list in the center table. In addition, users can also group protocols by direction -- either Ingress or Egress.

wXcked Eye Timing Nodes

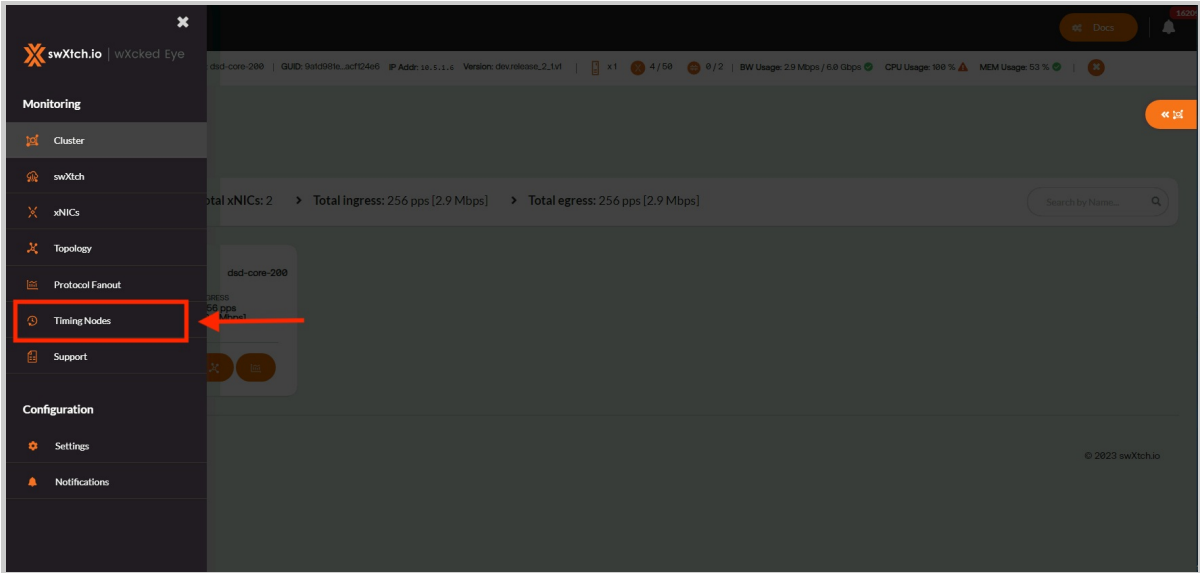
Finding the Timing Nodes page

To find the Timing Nodes page in the wXcked Eye UI:

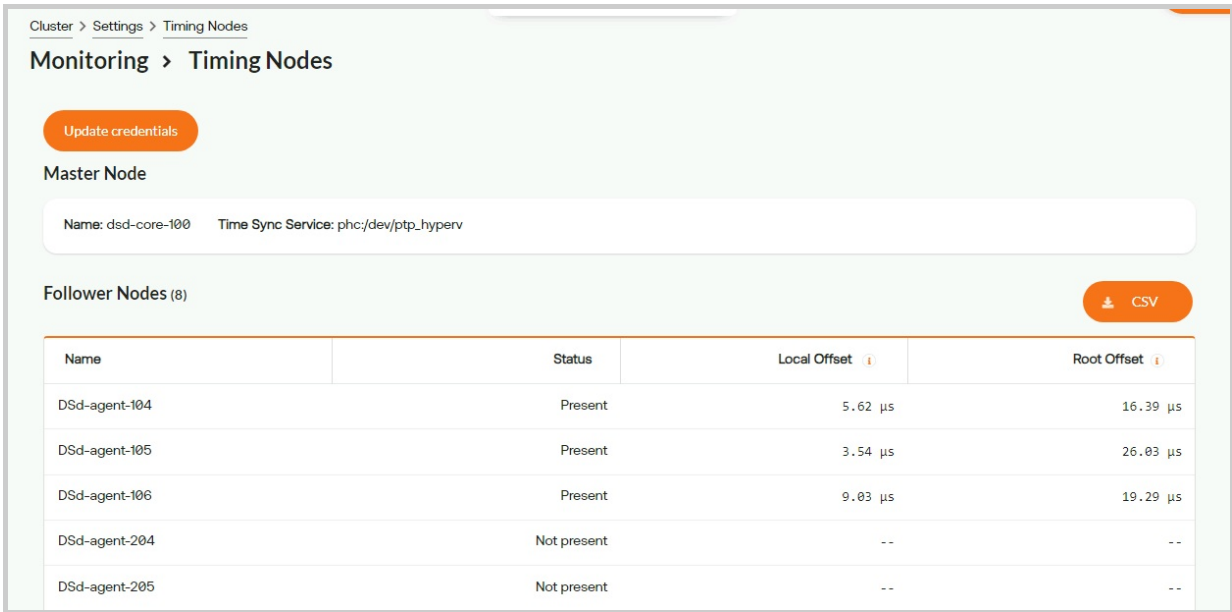
- 1. Click the menu icon (≡) next to the swXtch.io logo.



- 2. Select Timing Nodes under Monitoring.



Understanding the Timing Nodes page



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.

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."**

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, and 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.

Cluster > Settings > Support

Monitoring > Support

Reports based on statistical data stored on the disk with a certain periodicity.

From: 01/24/2024 7:18 PM To: 01/24/2024 7:18 PM

swXtch full report Download each report separately Select report...

swXtch Full Report	ZIP file with the swXtch full report for the selected dates
swXtch Info	JSON file with swXtch info
Max Highmarks	JSON file with the highest rates for the selected dates
List Highmarks	JSON file with all the rates for the selected dates
Logs	ZIP file with the swxrch-repl and swxrch-control logs for the selected dates

swXtch.io © 2023 swXtch.io

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.

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 in the **From** and **To** fields.

Contacting swXtch.io Support

For all troubleshooting requests, email the compressed file to support@swxtch.io for further instructions.

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

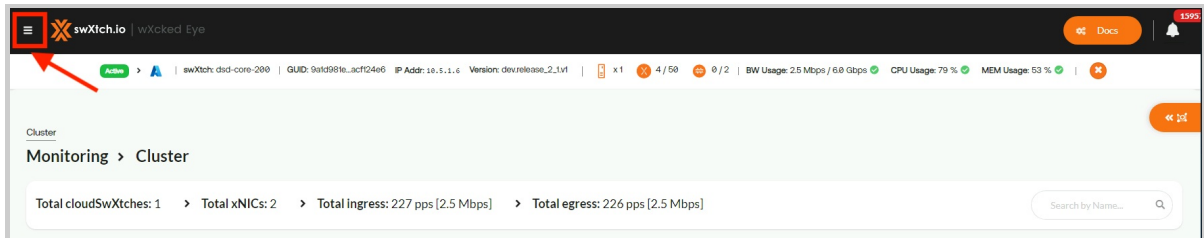
To access the wXcked Eye UI, users will need to enter the following URL into a web browser of a VM in their cloudswXtch environment. They should use the IP address of their cloudSwXtch to prefix the URL.

Custom	Copy
<swxtch-ip-address>/wxckedeye/	

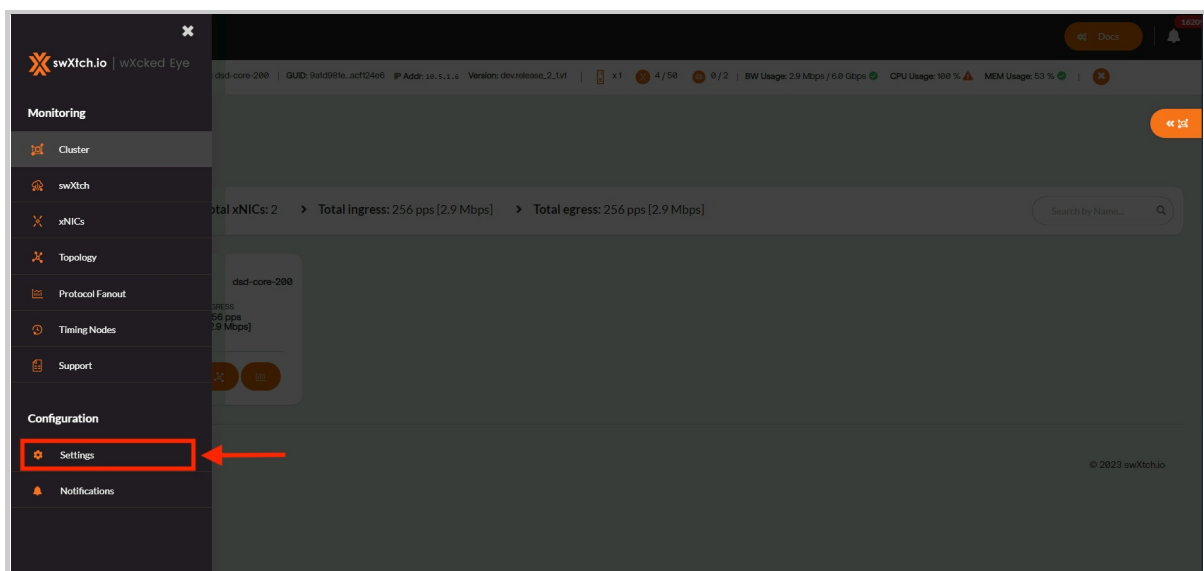
Finding the Settings page

To find the Settings page in the wXcked Eye UI:

1. Click the menu icon (≡) next to the swXtch.io logo.



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](#)

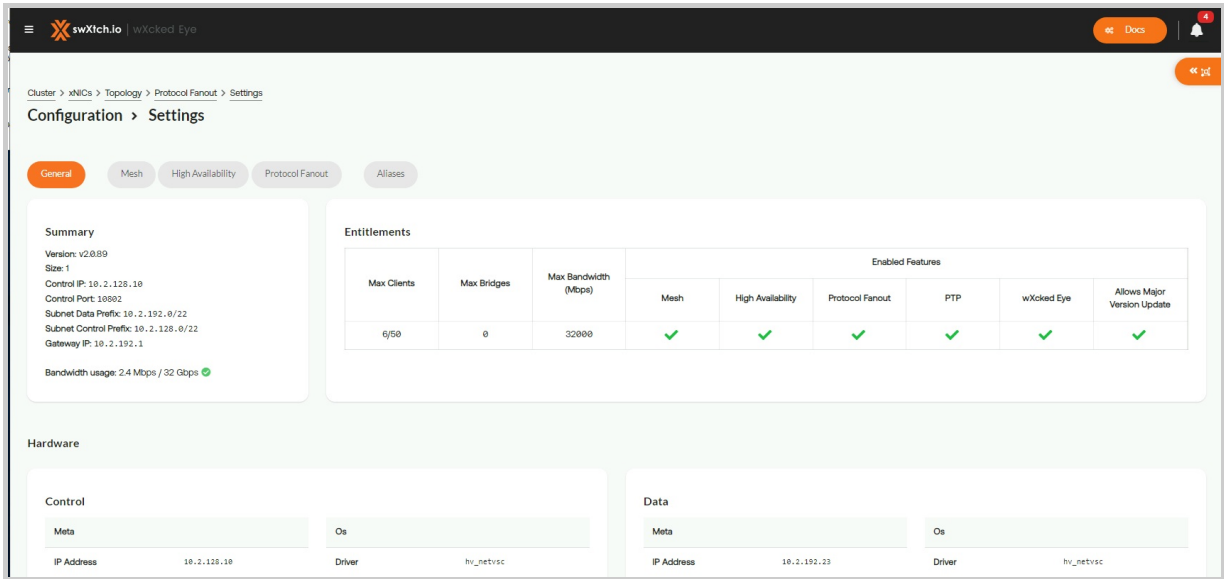
In this section, we will discuss each tab and how it offers the user additional control over their cloudSwXtch network.

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

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.

Entitlements								
Max Clients	Max Bridges	Max Bandwidth (Mbps)	Enabled Features					
			Mesh	High Availability	Protocol Fanout	PTP	wXcked Eye	Allows Major Version Update
6/50	0	32000	✓	✓	✓	✓	✓	✓

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.

Hardware							
Control				Data			
Meta		Os		Meta		Os	
IP Address	10.2.128.10	Driver	hv_netvsc	IP Address	10.2.192.23	Driver	hv_netvsc
IP Broadcast	10.2.131.255	Master of		IP Broadcast	10.2.195.255	Master of	
IP Subnet	10.2.128.0	MTU	1500	IP Subnet	10.2.192.0	MTU	1500
MAC Address	60:45:bd:a8:a6:ab	Name	eth0	MAC Address	60:45:bd:a8:af:c5	Name	eth1
Subnet Mask	60:45:bd:a8:a6:ab	PCI Address		Subnet Mask	60:45:bd:a8:af:c5	PCI Address	

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.

Actions	
Update swXtch	Data refresh period This slider allows you to change how often the data is updated across all monitoring pages. 5 secs

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.

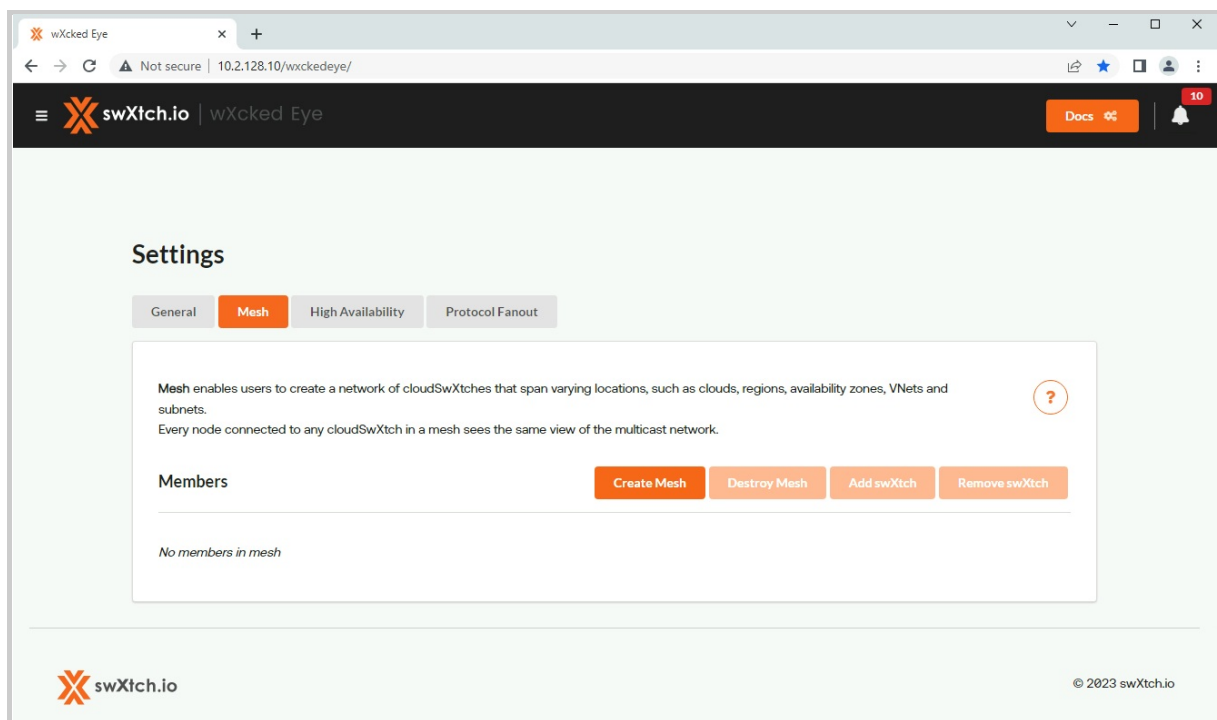
The **Mesh** tab is organized into 4 functions:

- [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.



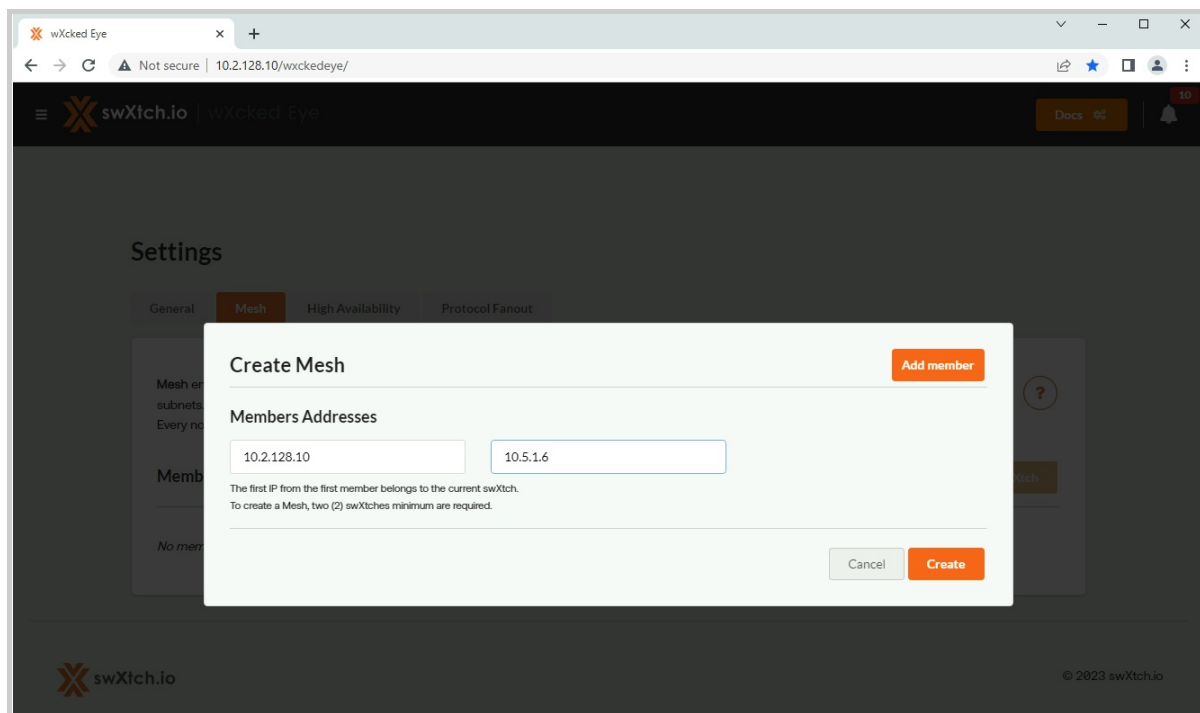
Mesh and High Availability

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:

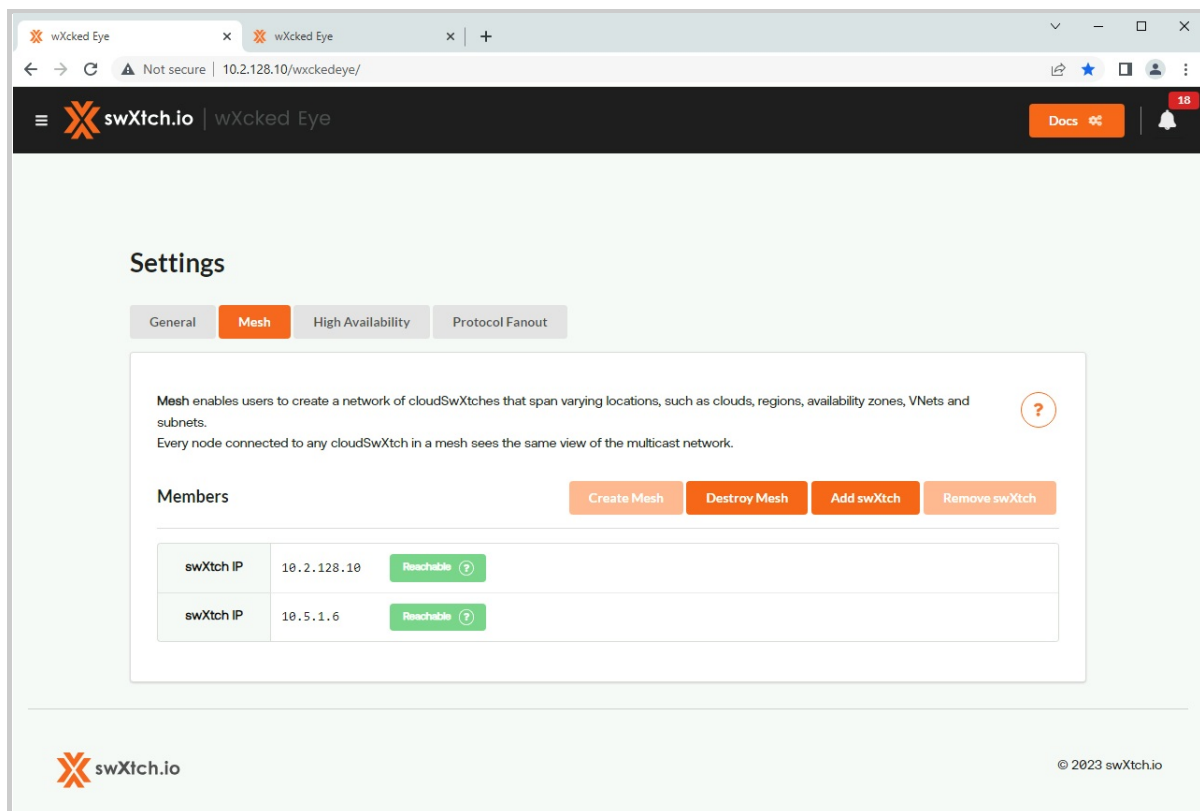
1. Click on the "Create Mesh" button. A new window will pop up.



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.

1. 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.
2. Please note: You must enter at least 2 cloudSwXtches in order to successfully create a mesh.

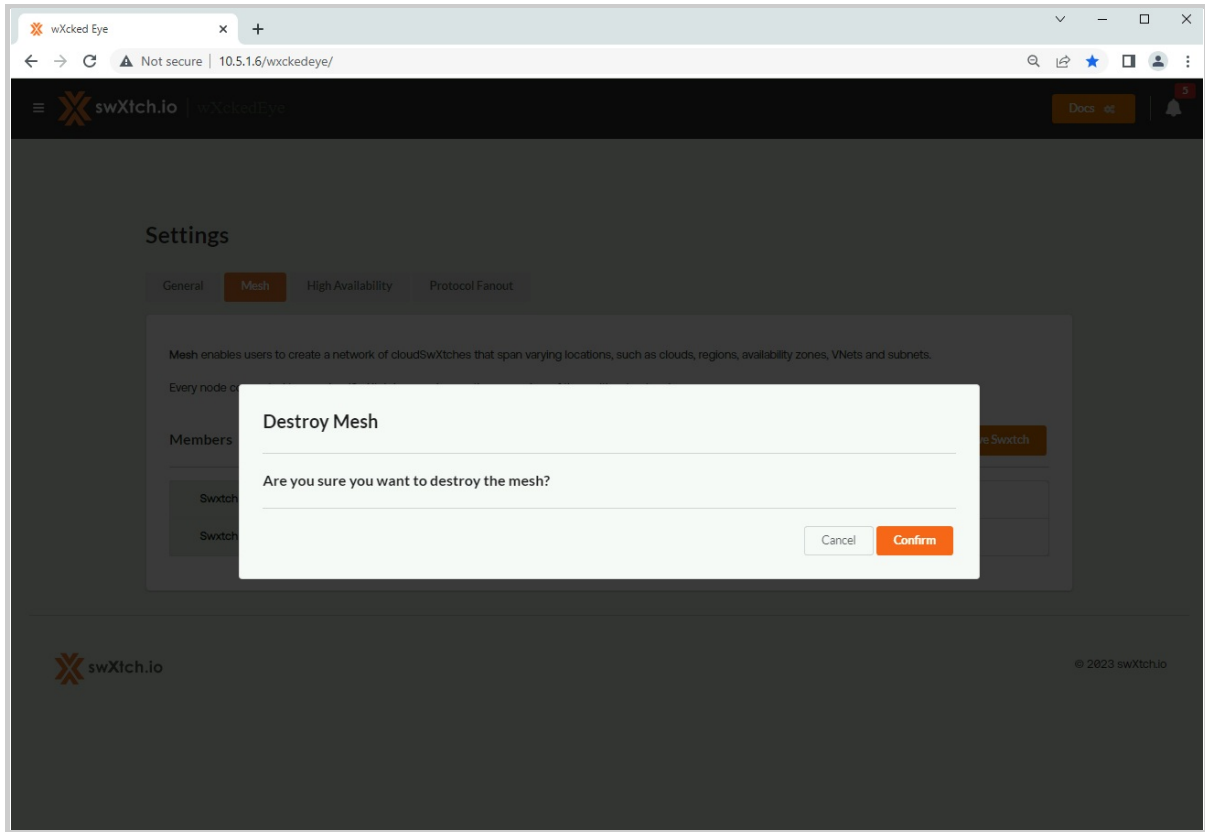
3. Click "Create" after you have completed adding members. The members should now be listed in the newly formed mesh. A tag on each swXtch IP will display whether or not the VM is reachable.



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

1. Click "Destroy Mesh." A warning pop-up will appear.



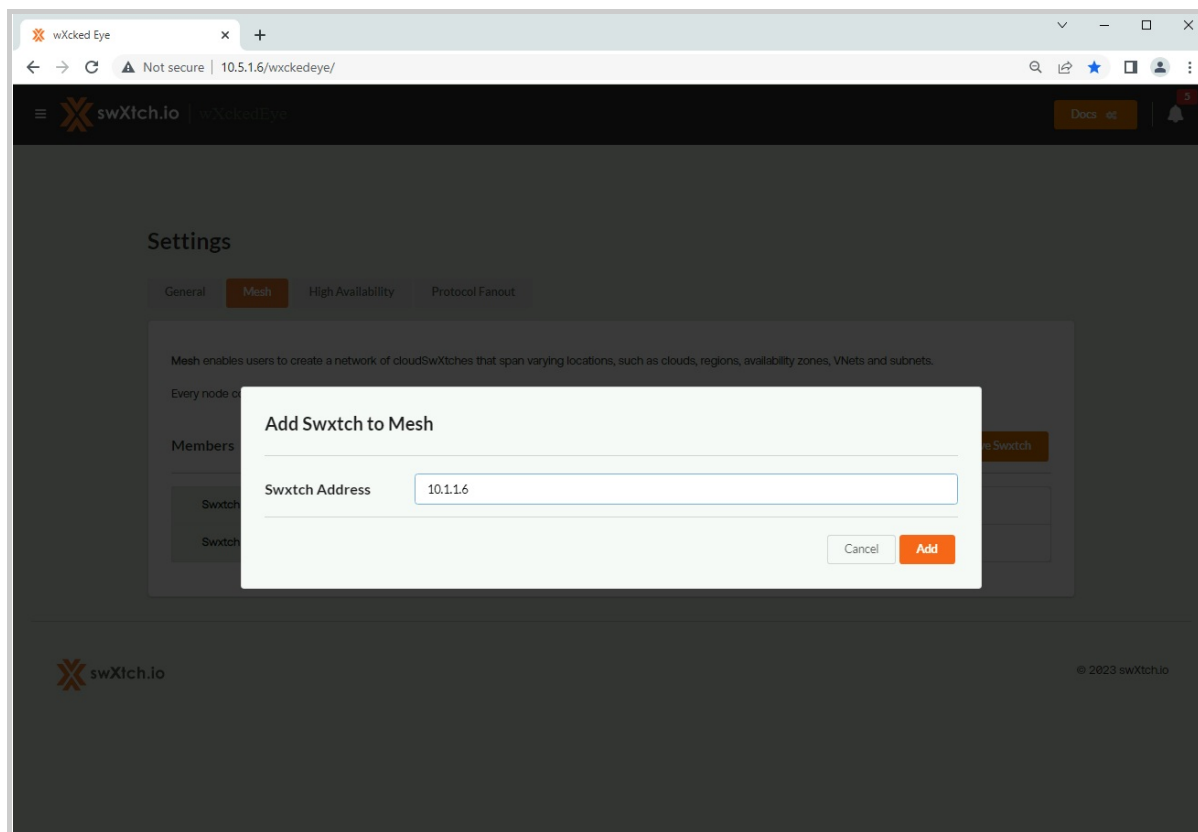
2. Select "Confirm." Your mesh and its members should no longer be listed in the Mesh tab.

Add SwXtch

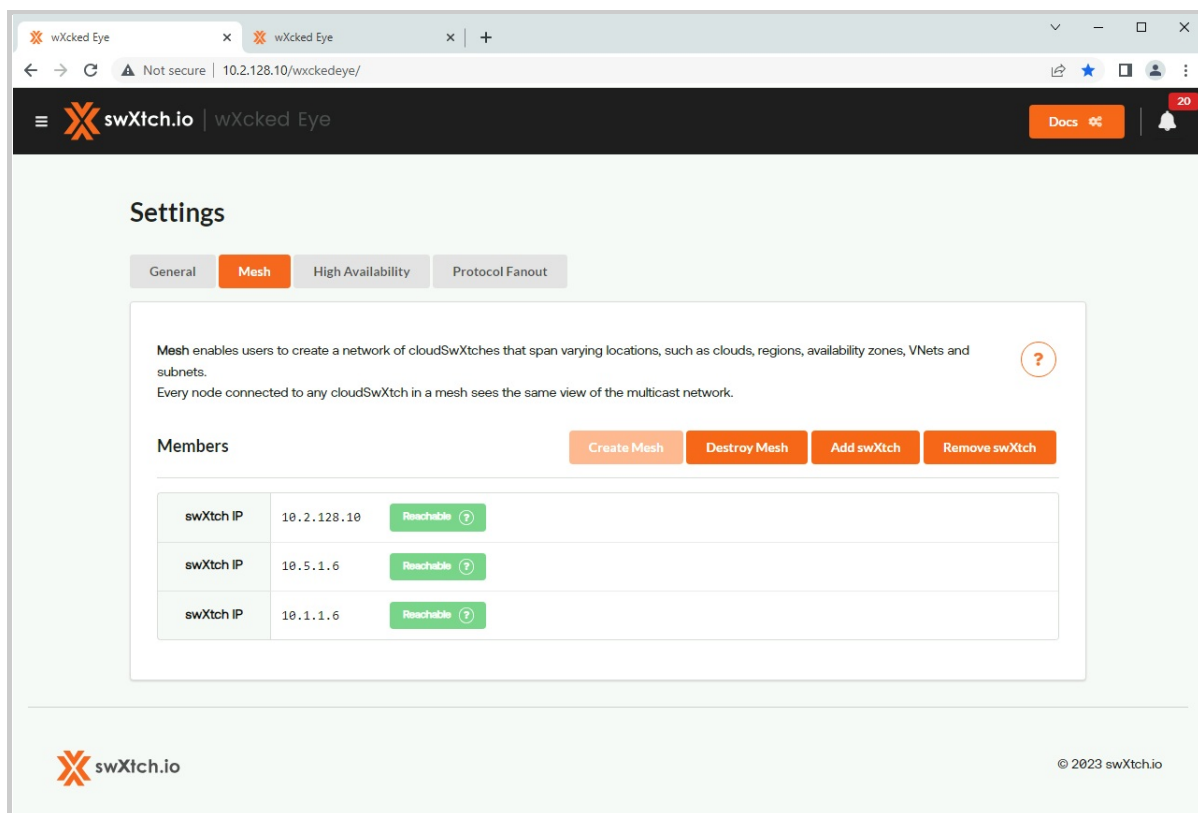
After creating a mesh, users may decide that they need additional cloudSwXtches. To add a cloudSwXtch:

1. Click on the "Add SwXtch" button in the Mesh main page of any existing cloudSwXtch.

2. Enter the IP address of the cloudSwXtch you wish to add to the mesh.



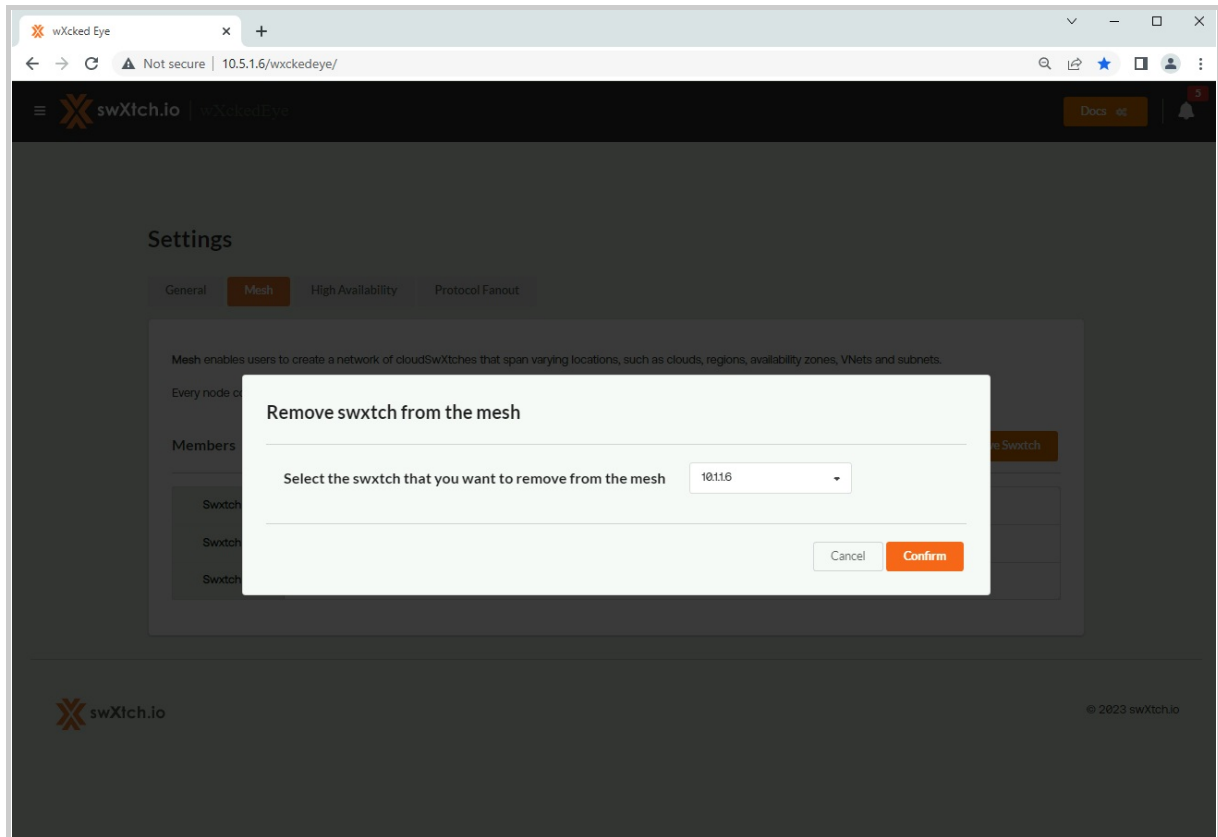
3. Click "Add." A new cloudSwXtch should now be listed in your mesh.



Remove SwXtch

1. Click the "Remove Swxtch" option. A new window will open confirming the removal.

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.

High Availability with wXcked Eye

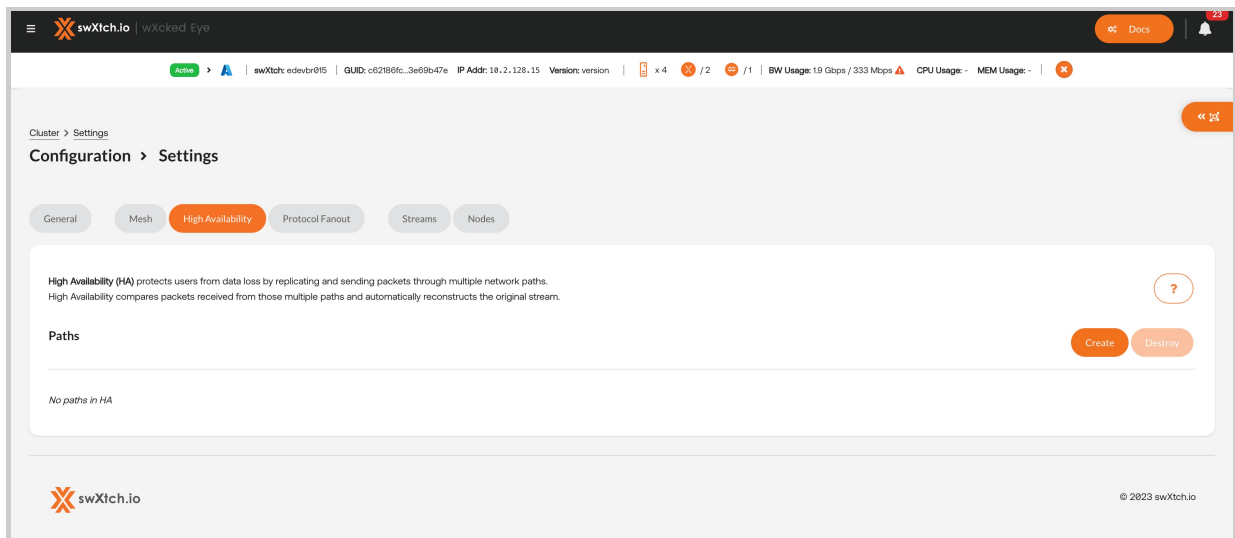
Navigating to the High Availability tab

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.

2. **Name** your HA configuration. In this example, the HA is named "My High Availability."

HA > Create

Name

My High Availability

Paths

Add path

Name

Path 1

swXtches

10.2.128.10

Add swXtch

Name

Path 2

swXtches

10.5.1.9

Add swXtch

10.5.1.6

- The first IP from the first path belongs to the current swXtch
- Two (2) paths minimum are required

Cancel Create

3. **Name** the first path and enter the **IP addresses** for the 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.

swXtch.io

Cluster > Settings

Configuration > Settings

General Mesh High Availability Protocol

High Availability (HA) protects users from data loss by replicating data across multiple nodes. High Availability compares packets received from those multiple nodes to ensure data integrity.

Paths

No paths in HA

swXtch.io

HA > Create

Name

High Availability

Paths

Add path

Name

Path 1

swXtches

10.2.128.15

Add swXtch

Name

Path 2

swXtches

10.5.1.9

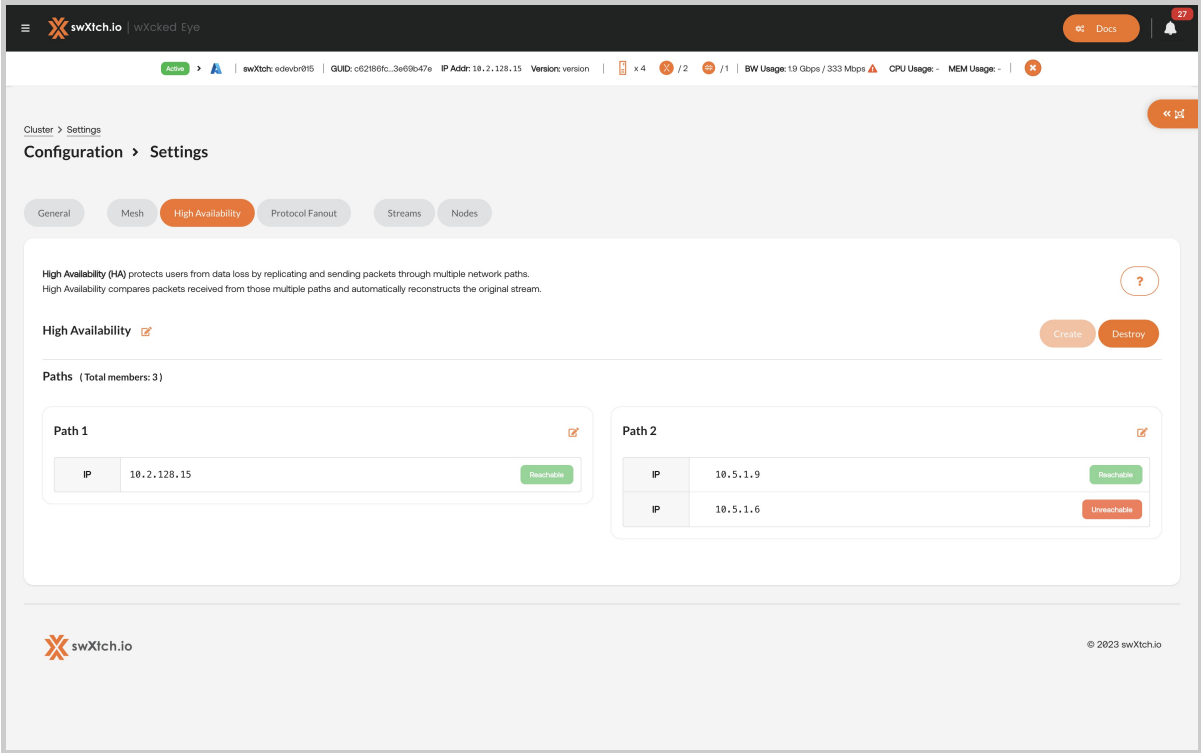
Add swXtch

10.5.1.6

- The first IP from the first path belongs to the current swXtch
- Two (2) paths minimum are required

Cancel Create

6. Click "Create." A new High Availability flow has been created.



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.

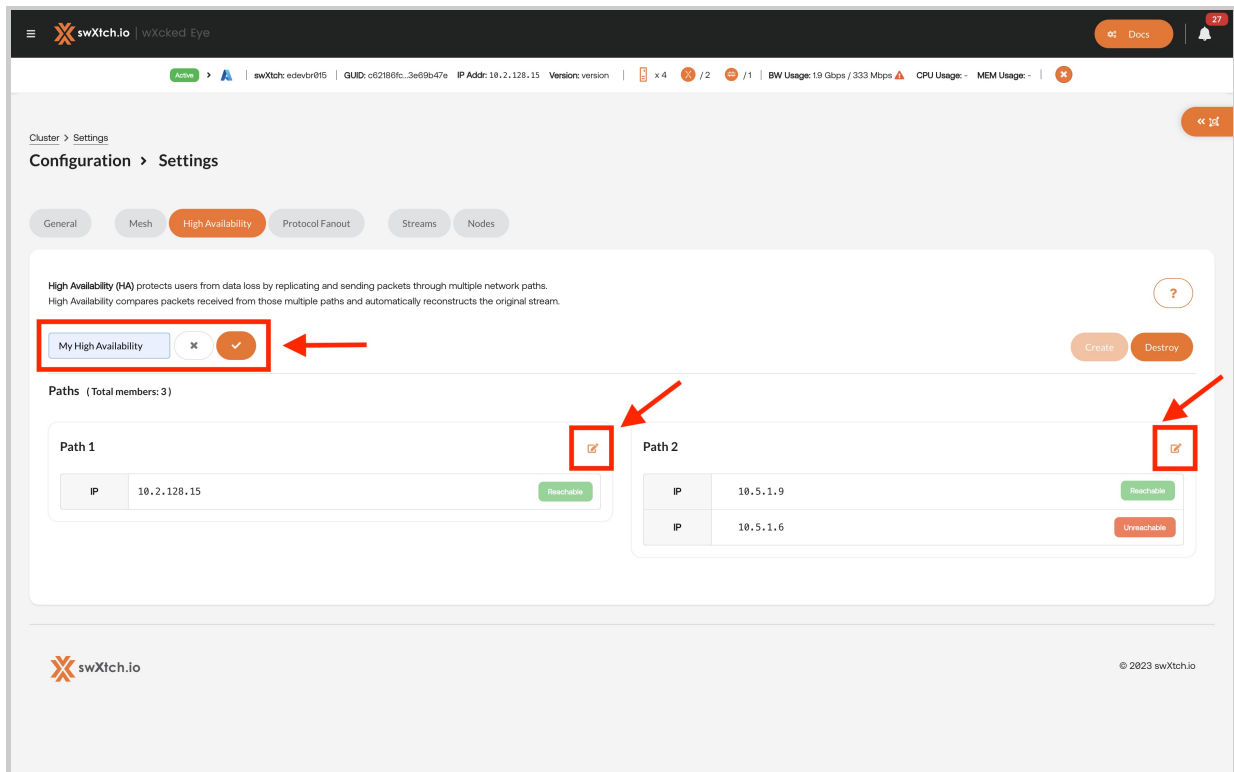
From any of the connected cloudSwXtches, users can destroy their HA configuration.

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

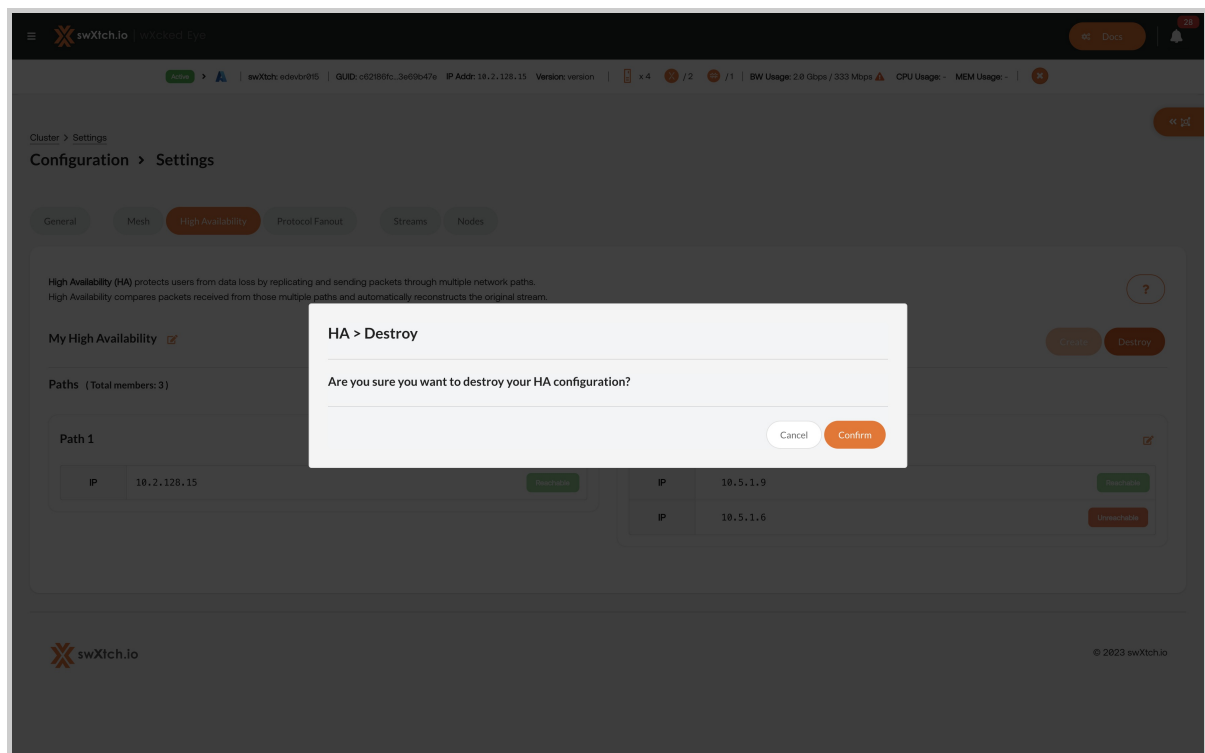
Users can dynamically change their high availability and paths' names directly in the UI. To do this, simply click the edit button next to the paths.



Destroy HA

To destroy an HA configuration, a user will need to go to the HA page of any associated cloudSwXtches.

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.

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.

Cluster > Protocol Fanout > Settings
Configuration > Settings

General Mesh High Availability **Protocol Fanout** Streams Nodes

Protocol Fanout enables users to configure unicast adaptors to send and receive traffic through the swXtch using different protocols (UDP, SRT and RIST).
Ingress direction describes receiving unicast traffic in any supported protocol and converting it into a multicast stream on the swXtch.
Egress describes sending 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 (for instance, an UDP input being sent to a set of multicast destinations and, additionally, to one using SRT).

Adaptors (24) Filter by Protocol Direction Add Adaptor Remove selected

	Protocol	Direction	Stream		Node		Listener Port	Action
			IP : Port	Alias	IP : Port	Alias		
☐	UDP	Egress	229.1.2.3:4000	Tennis	10.2.192.62:4401			☐ ☐ ☐
☐	SRT Caller	Egress	229.1.2.3:4000		10.2.128.65:45001			☐ ☐ ☐
☐	UDP	Egress	229.1.2.3:5000	Football	10.2.192.62:5401			☐ ☐ ☐
☐	RIST Caller	Egress	229.1.2.3:3000		10.2.128.65:3601			☐ ☐ ☐

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.

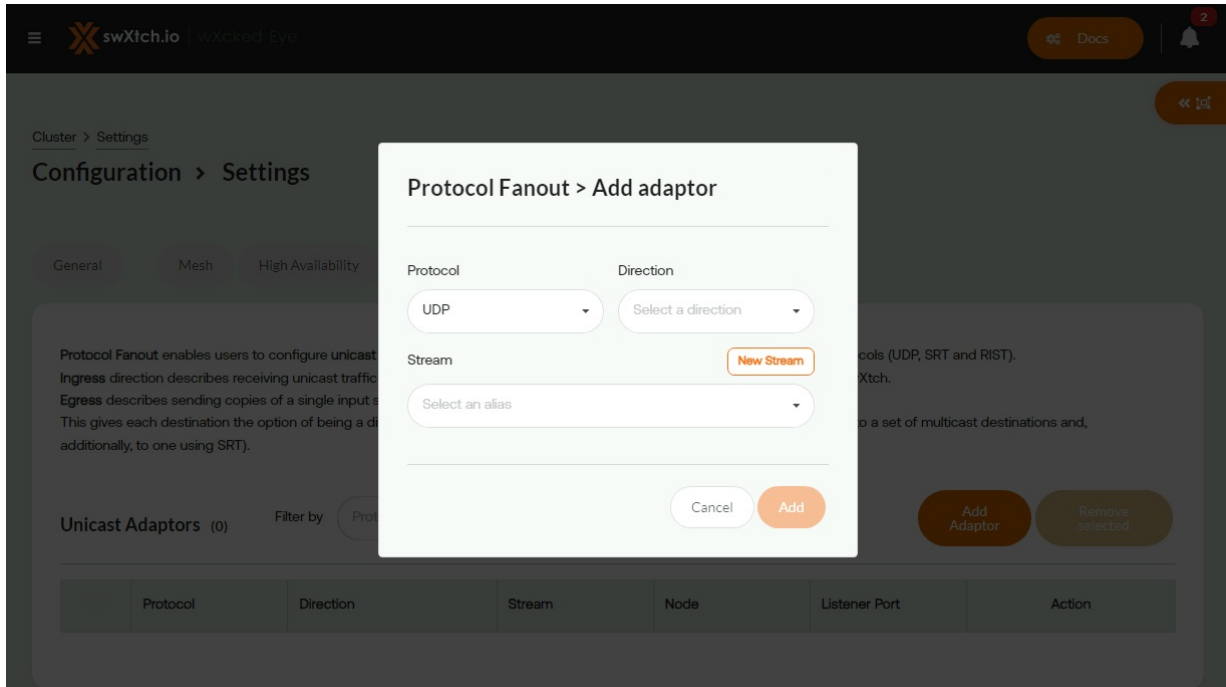
To add an adaptor, select "Add Adaptor." Here, a user can select between the three protocols.

UDP

Ingress vs. Egress

Differentiating between ingress and egress can be difficult. It is important to imagine it in relation to the cloudSwXtch.

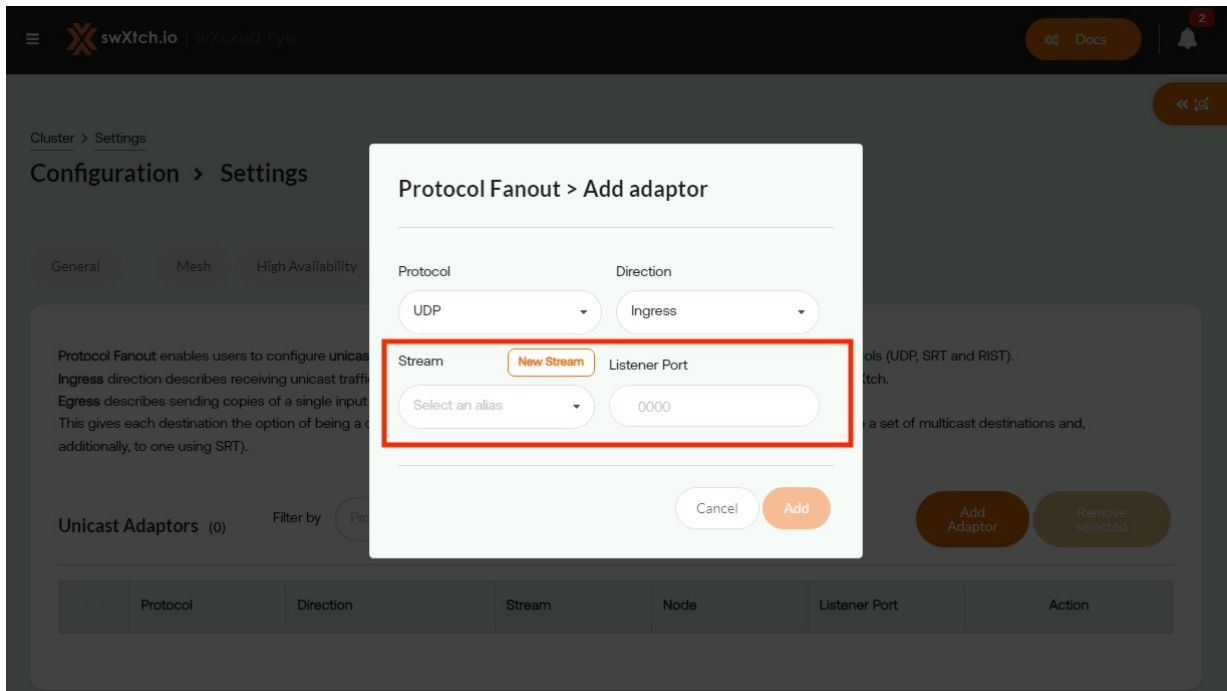
- **Ingress:** Data is coming into the cloudSwXtch.
- **Egress:** Data is leaving the cloudSwXtch.



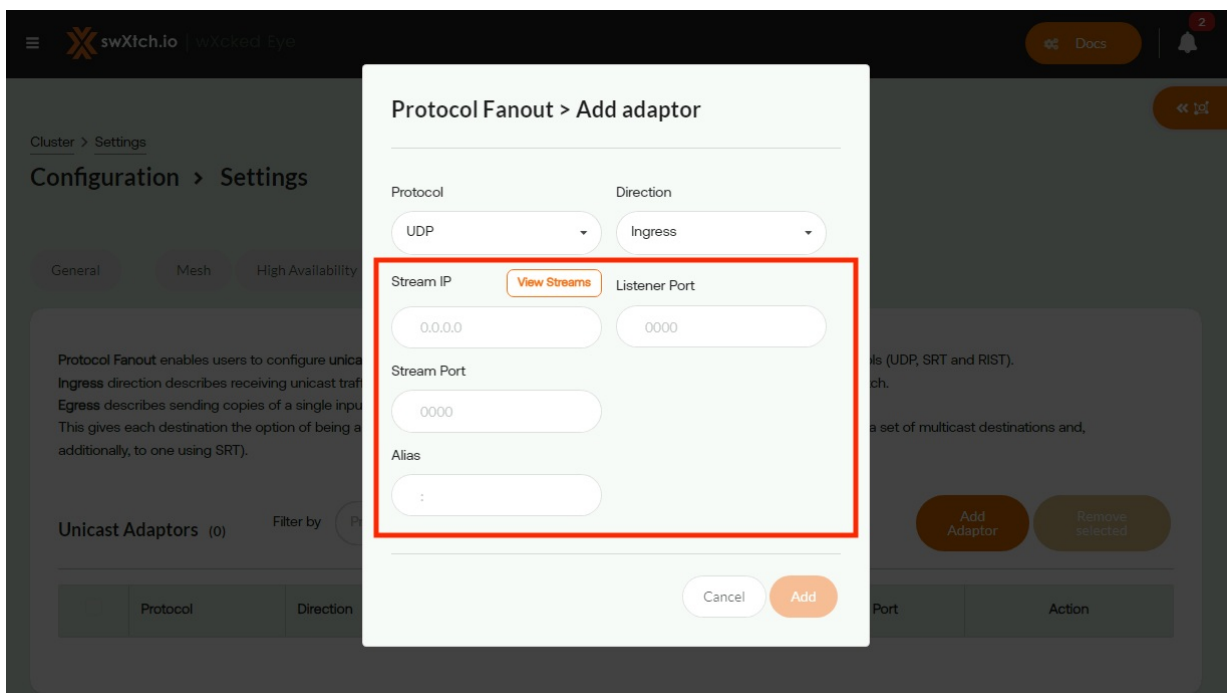
Selecting UDP under Protocol allows users to add mapping for UDP traffic entering and leaving the cloudSwXtch. Depending on the direction of the data, a user will have to add additional information to set up a successful connection.

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**.



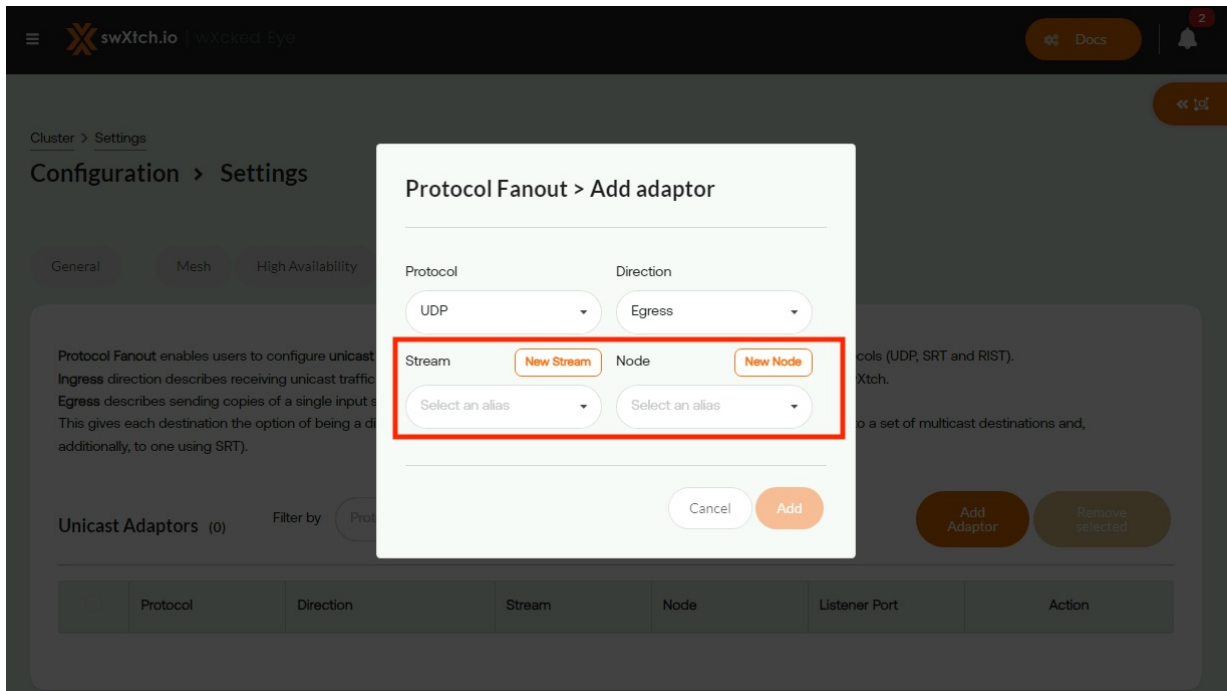
Alternatively, if a user does not have an Alias set up for their desired stream, they can manually specify a **Stream IP**, **Listener Port**, a **Stream Port** and an **Alias** name to add one. To do this, they would have to select the **New Stream** option in the panel.



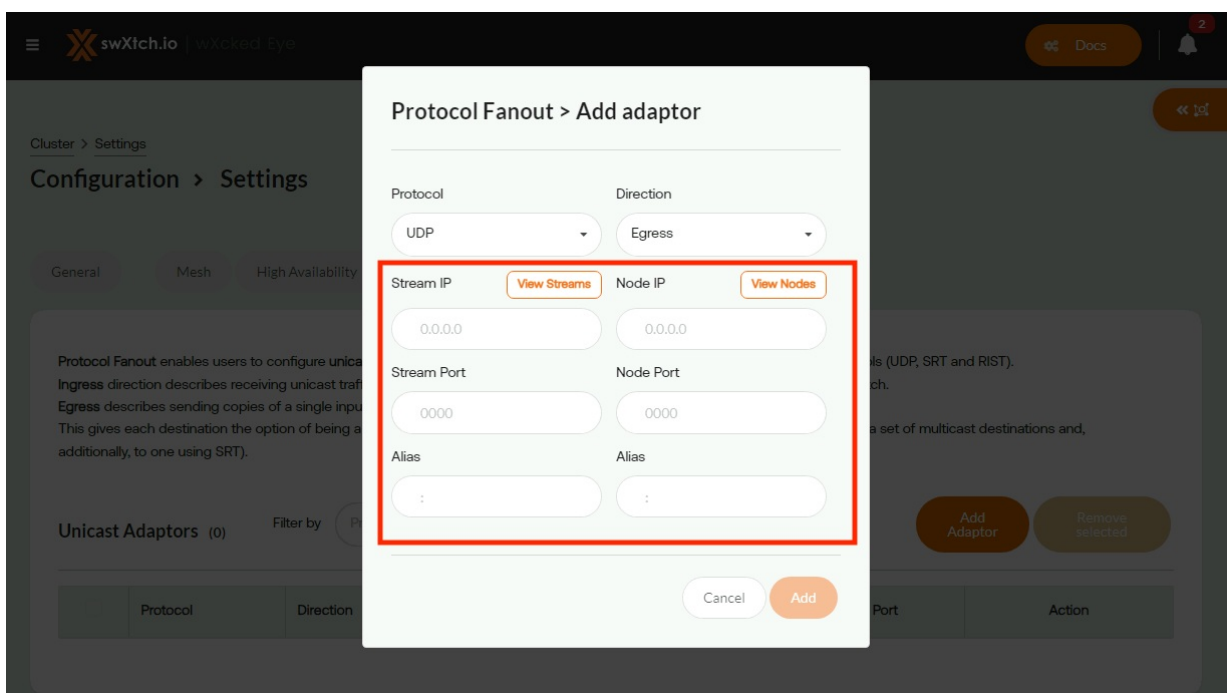
Both methods will allow endpoints to send unicast data. Once that connection has been established, the cloudSwXtch will be able to ingest the unicast data as multicast.

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.



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

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.

Protocol Fanout > Add adaptor

Protocol: SRT Caller | Direction: Ingress | Options: Select an option

Stream: [New Stream](#) | Node: [New Node](#)

Stream: Select an alias | Node: Select an alias

Buttons: Cancel, Add

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.

Protocol Fanout > Add adaptor

Protocol: SRT Caller | Direction: Ingress | Options: Select an option

Stream IP: 0.0.0.0 | Node IP: 0.0.0.0

Stream Port: 0000 | Node Port: 0000

Stream Alias: : | Node Alias: :

Buttons: Cancel, Add

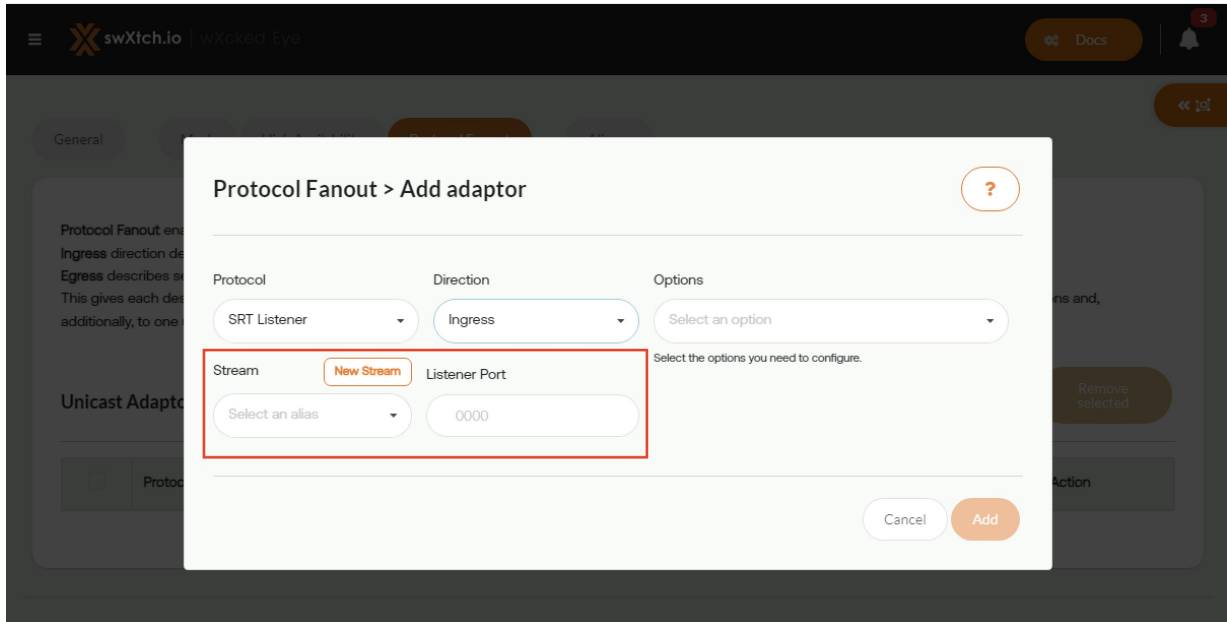
The **Options** dropdown allows for additional fine tuning based on standards of the protocol. Selection of an option will open another field.

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.

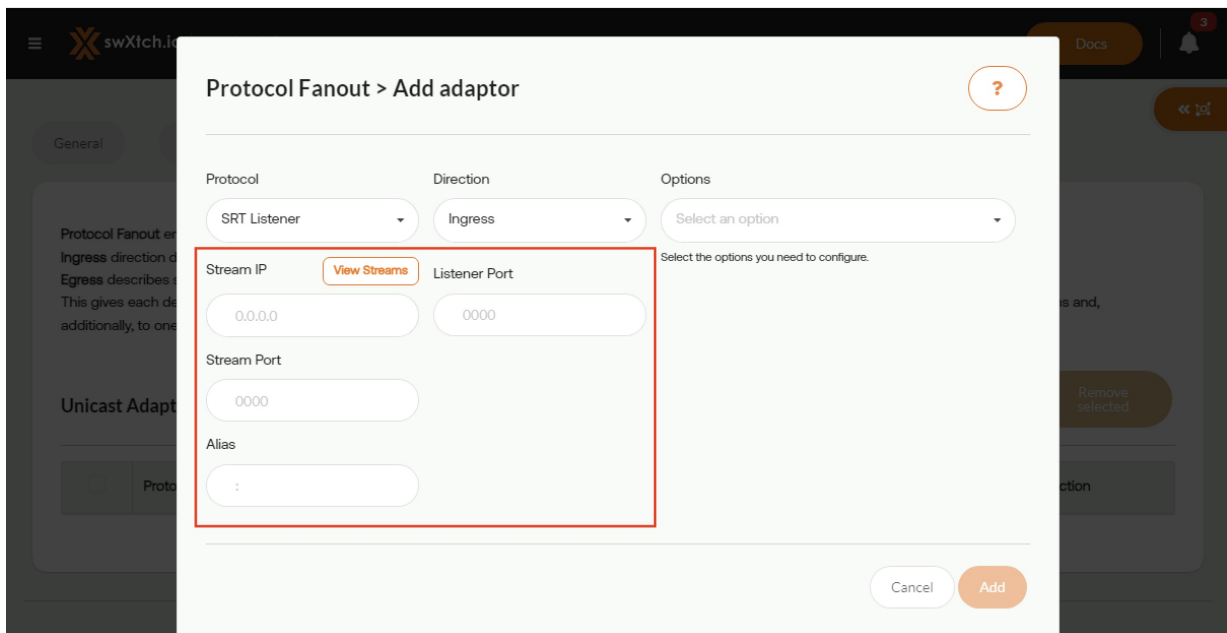
For Egress, the user will be specifying the Target Node for an endpoint to receive an SRT or RIST stream from the cloudSwXtch. The cloudSwXtch will then call to the Destination or Target Node to establish a connection before transmitting the stream.

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.



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.

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

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.

Settings > Aliases

General Mesh High Availability Protocol Fanout Aliases

Aliases allow the user the possibility to assign friendly names to their streams and nodes. This makes the protocol fanout configuration and the network graph usability much easier.

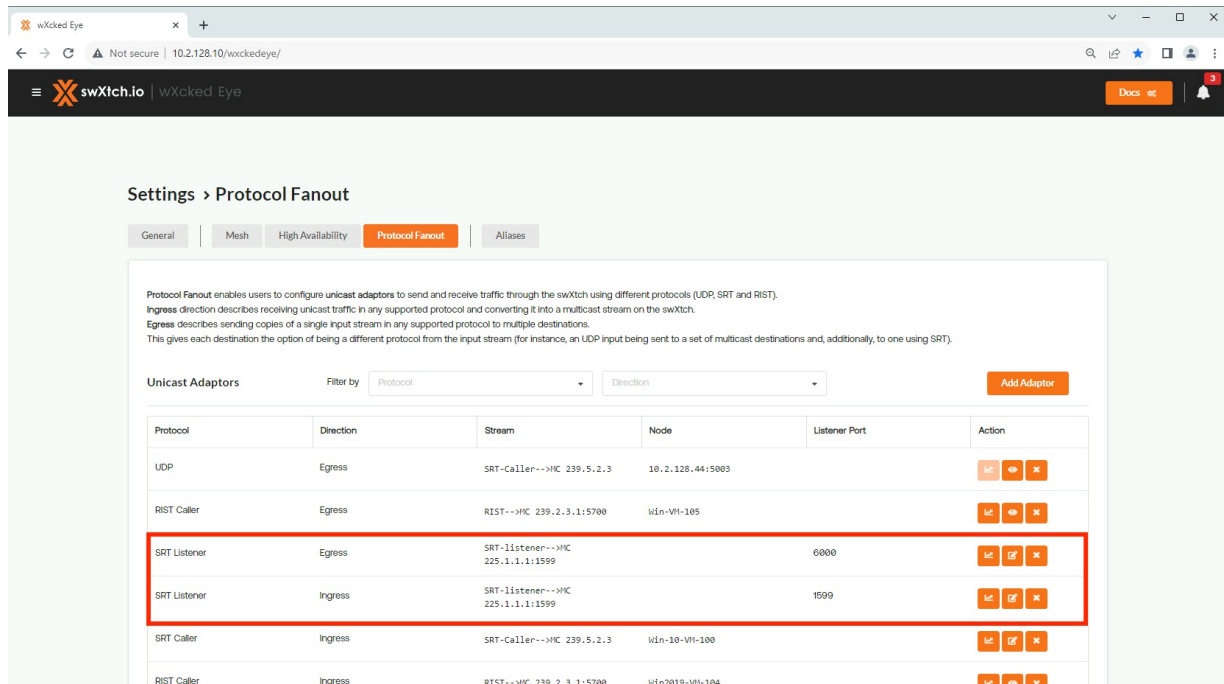
Alias	Stream IP	Stream Port	Action
SRT-listener-->MC 225.1.1.1:1599	225.1.1.1	1599	[Add] [Remove]
RIST-->MC 239.2.3.1:5700	239.2.3.1	5700	[Add] [Remove]
Unicast Egress	239.2.4.5	5400	[Add] [Remove]
Unicast	239.4.5.6	2000	[Add] [Remove]
RIST-L-out-239.4.6.7:3408	239.4.6.7	3408	[Add] [Remove]
SRT-Caller-->MC 239.5.2.3	239.5.2.3	3450	[Add] [Remove]

Alias	Node IP	Node Port	Action
Win-VM-105	10.2.128.15	3400	[Add] [Remove]
drs-105-10.2.128.15:3401	10.2.128.15	3401	[Add] [Remove]
Win-10-VM-100	10.2.128.36	3000	[Add] [Remove]
Win-10-101-10.2.128.37:5004	10.2.128.37	5004	[Add] [Remove]
10.2.128.37:6000	10.2.128.37	6000	[Add] [Remove]
Win2019-VM-104	10.2.128.75	1599	[Add] [Remove]

For more information about how aliases work, see the [Aliases](#) article under Configure cloudSwXtch with wXcked Eye.

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.

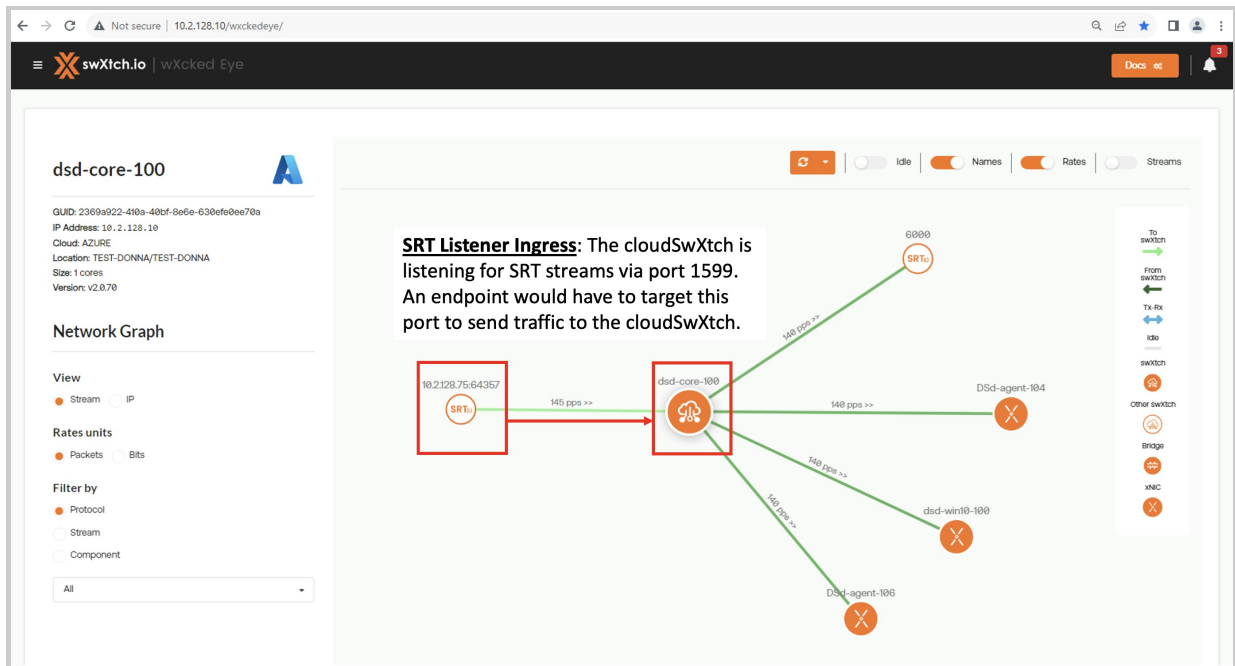


The user has set up SRT Listener for both Egress and Ingress using the Alias assigned earlier: **SRT-Listener - > Multicast 225.1.1.1:1599**. When differentiating between Egress and Ingress, always imagine it from the perspective of the cloudSwXtch:

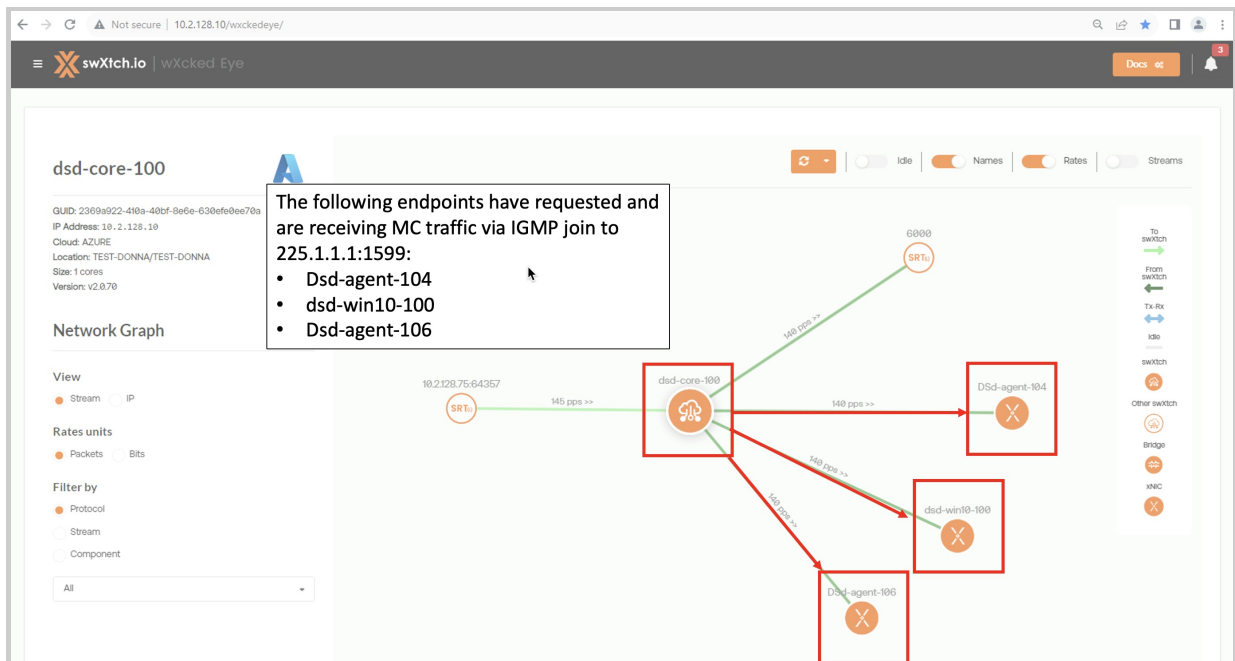
SRT Listener Ingress

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.

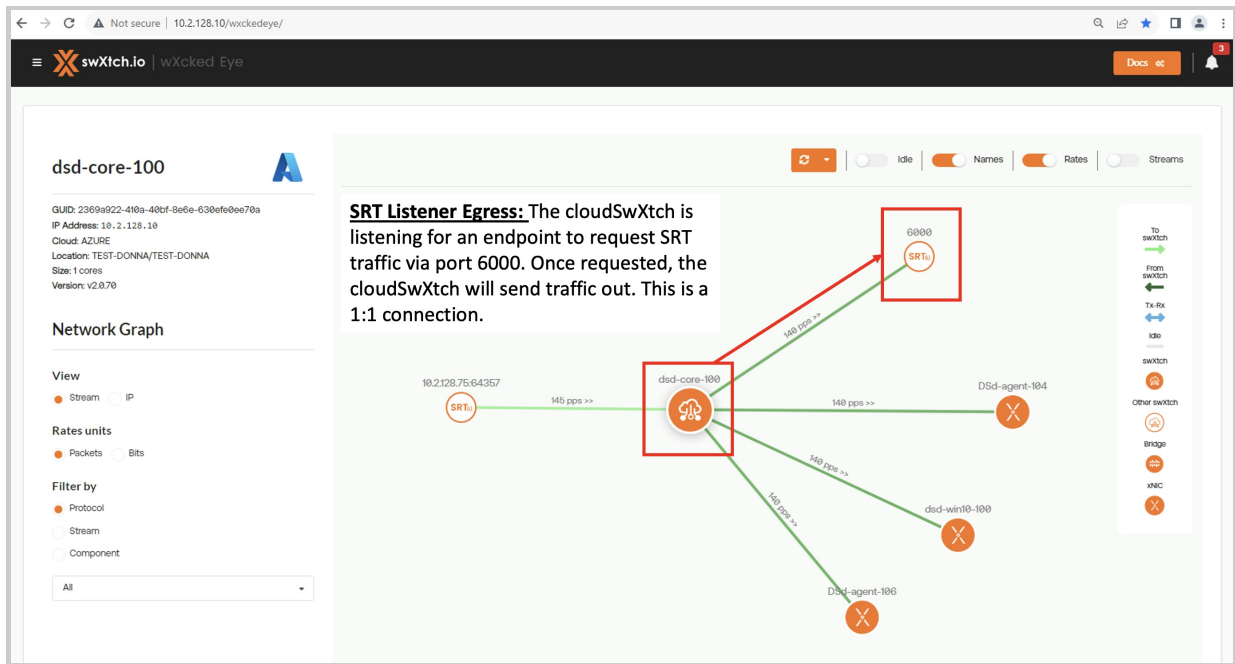


The endpoints highlighted will then request and receive multicast traffic via IGMP join to 225.1.1.1:1599.



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.

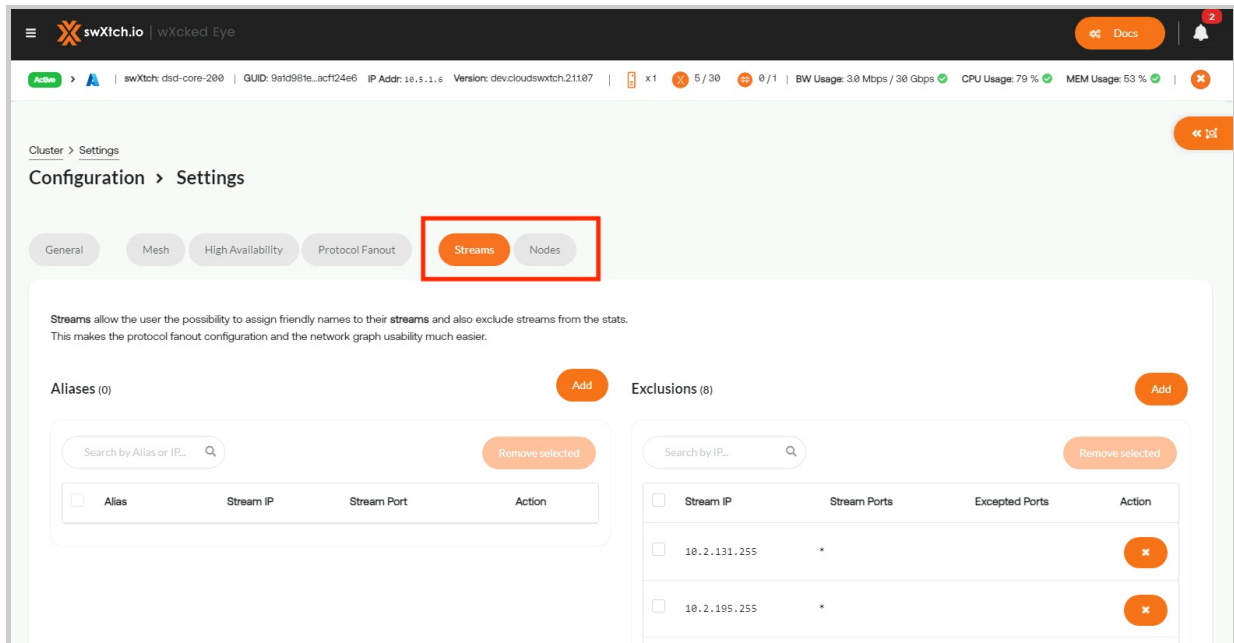


Stream and Nodes

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.

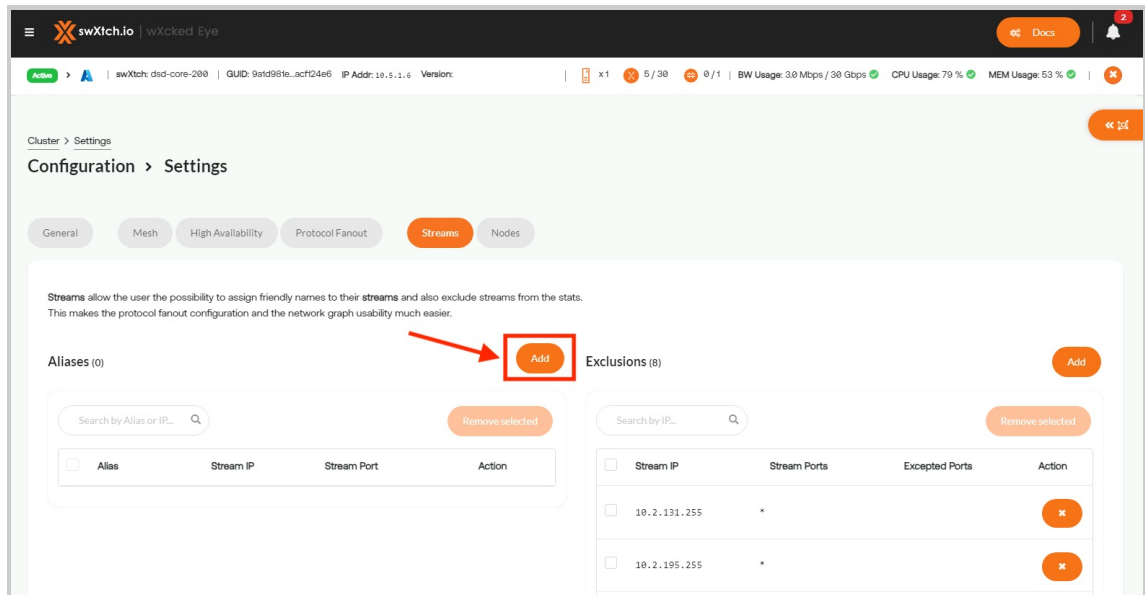


Streams

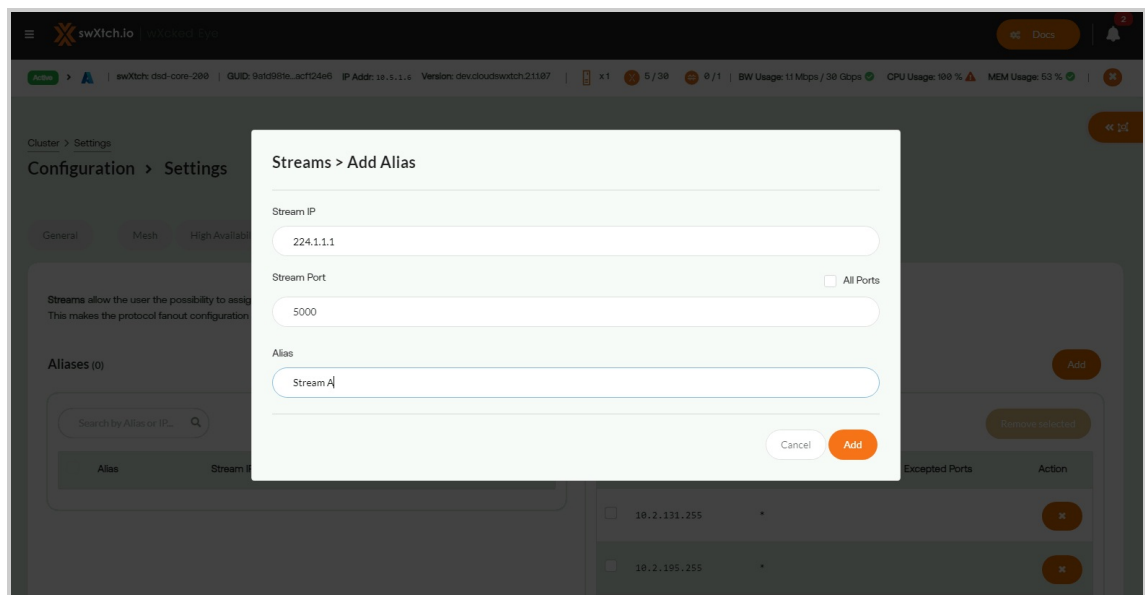
The Streams tab allows users to name their Multicast IP addresses. This creates a shortcut for users in dropdown menus throughout the Protocol Fanout tab, allowing them to easily differentiate between multiple streams.

To add a new stream:

1. Select **Add Alias** in the top right corner of the **Streams** panel. A pop-up will open.

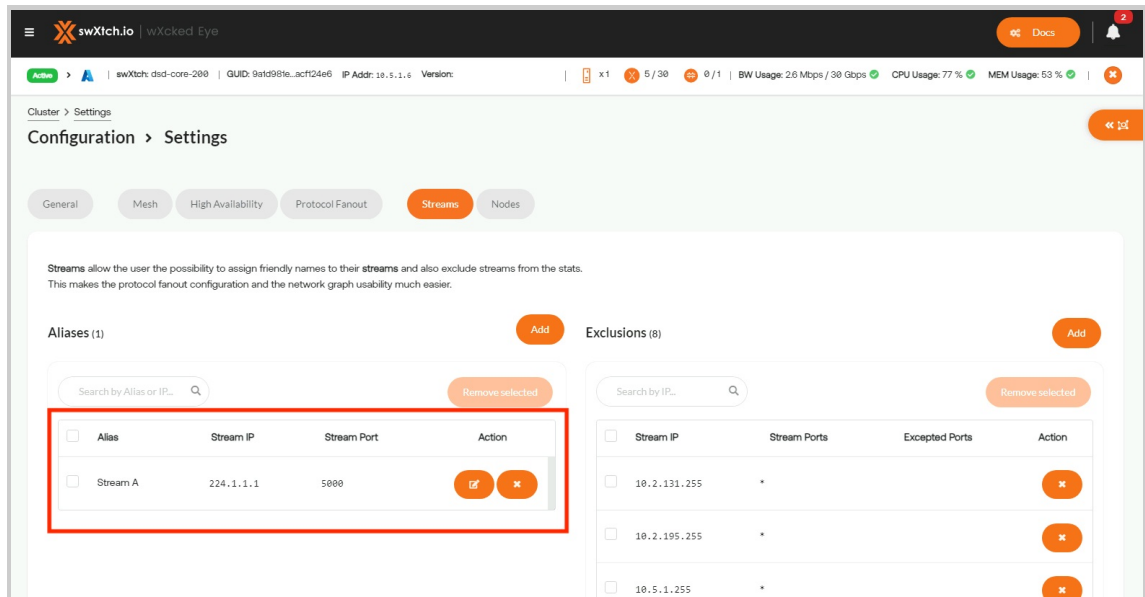


2. Enter the multicast **Stream IP**, **Stream Port** and a user-friendly name under **Alias**. In the example, the user assigns the Alias, **Stream A**. In addition to setting a single Stream Port, you may also open 'All Ports' by selecting the checkbox.



3. Click **Add**.

4. A new stream will be added to the list.



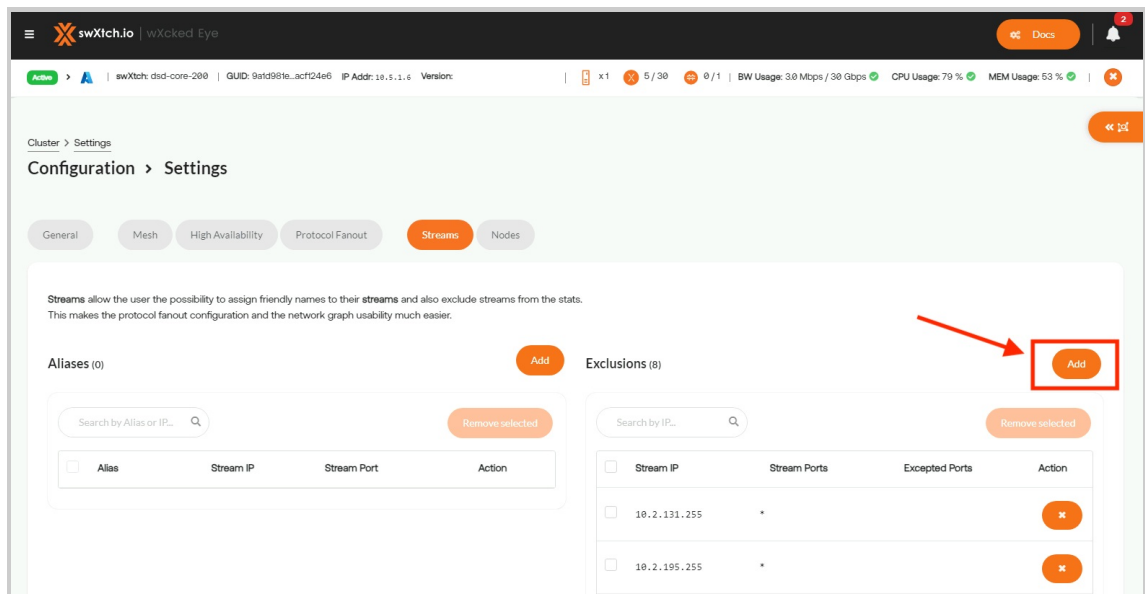
Users can edit the name of the stream or remove it from the list by clicking either buttons under the Action column.

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.

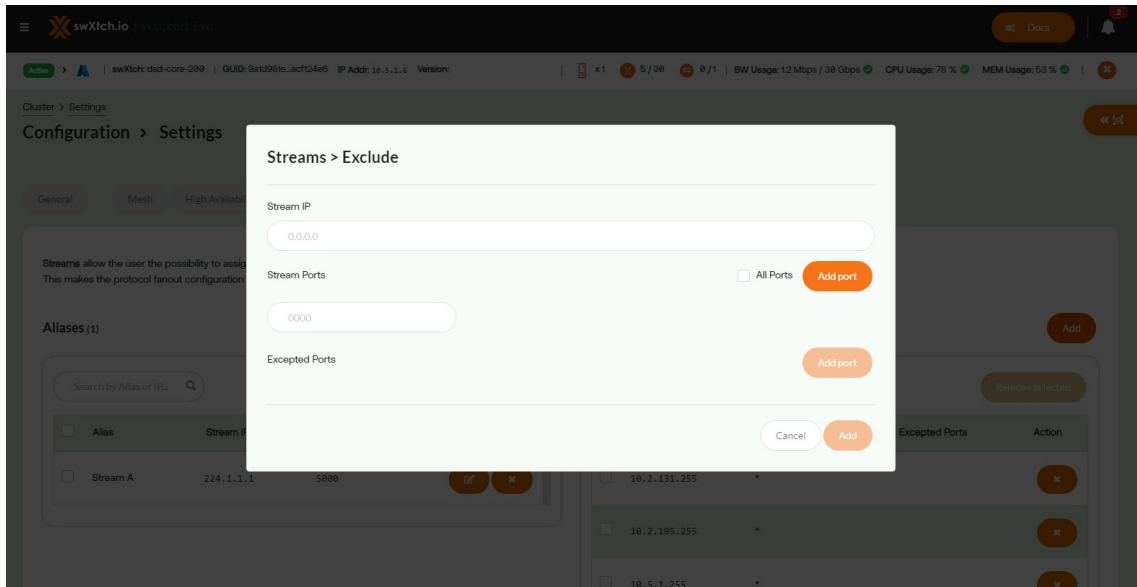
Exclusions

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.



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.



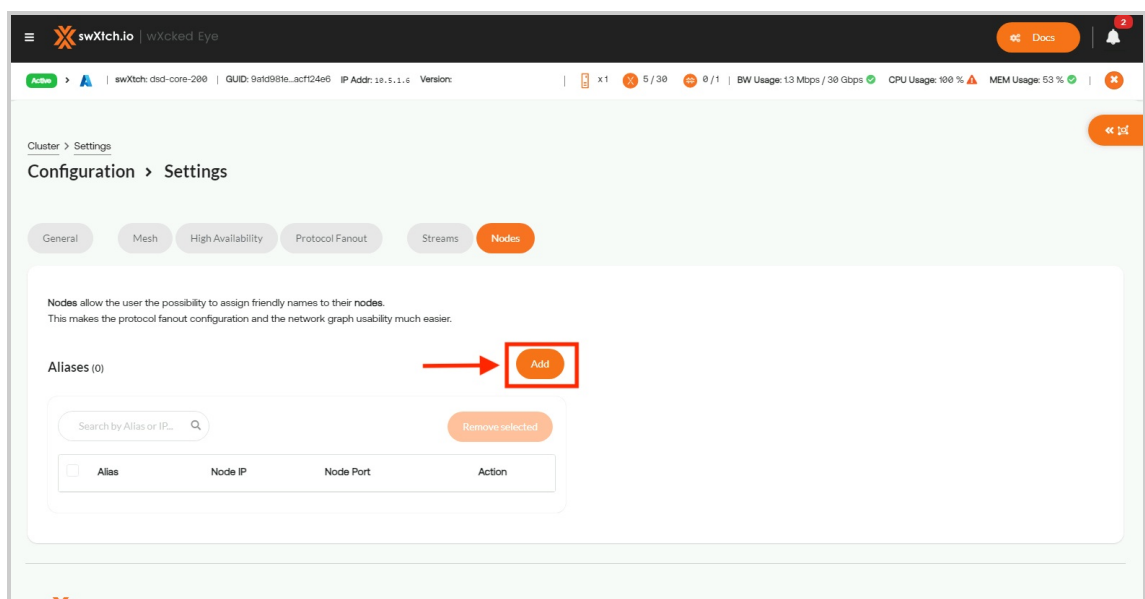
3. Click **Add** when you have made all your edits.
4. A new Stream will be added to your **Exclude Stream** list. It will no longer appear in **wXcked Eye** and **swXtch-top** monitoring.

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.

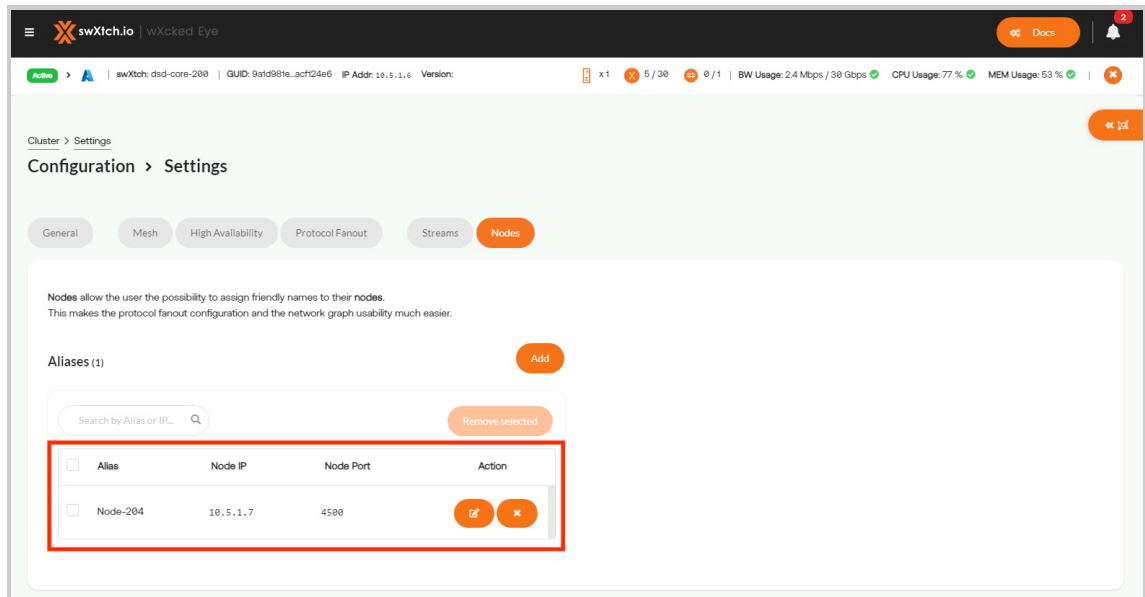
To add a new node:

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**.
3. Click **Add** to confirm.

4. A new node will be added to the list.

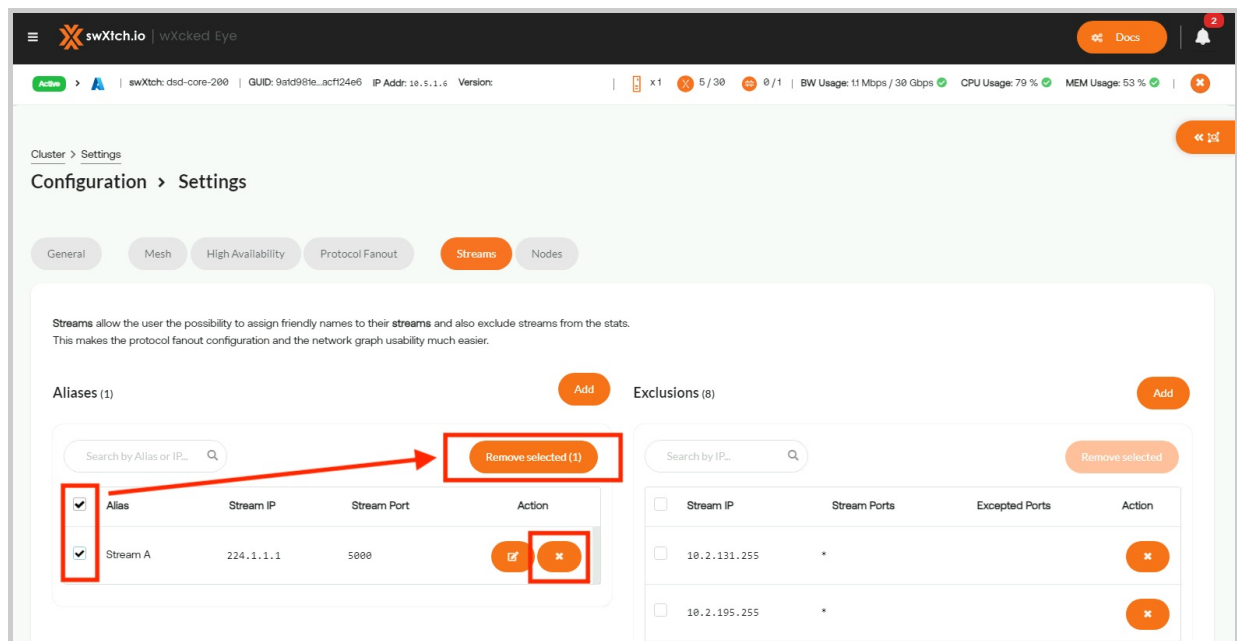


Users can edit the name of the node or remove the name by clicking the buttons under the **Action** column.

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.



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.

Bash	Copy
<pre>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.</pre>	

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:

Bash	Copy
<pre>{ "name": "MY High Availability", "paths": [{ "name": "Path 1", "swxtches": ["10.2.128.10"] }, { "name": "Path 2", "swxtches": ["10.5.1.6"] }] }</pre>	

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
<pre>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.</pre>	

Below is the command:

Bash	Copy
<pre>swx ha create -i <path_to_config> -s <cloudSwXtch_IP></pre>	

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
<pre>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.</pre>	

Below is the command to leave:

Bash	Copy
<pre>PS C:\Users\testadmin> swx ha destroy -s <swxtch name or control data ip of a cloudSwXtch in the HA configuration></pre>	

Example:

Bash	Copy
<pre>PS C:\Users\testadmin> swx ha destroy -s cloudswxtch101 Validating cluster deletion. Successfully deleted the cluster.</pre>	

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
<pre>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.</pre>	

Below is the command to list:

Bash	Copy
<pre>swx ha show -s <swxtch name or control data IP of a cloudSwXtch in the HA configuration></pre>	

Example below:

Bash	Copy
<pre>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"] }] } }</pre>	

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`

Bash	Copy
<pre>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]</pre>	

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
<pre>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</pre>	

Bridge Instance 2

Bash	Copy
<pre>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</pre>	

- **--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 xNIC for High Availability streams, the `xnic.json` file needs to be updated for both the producer and consumer.

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:

Bash	Copy
<pre>sudo nano /var/opt/swxtch/xnic.json</pre>	

For Windows:

The file can be found in `C:\Program Files\SwXtch.io\Swxtch-xNIC\xnic.json`

Single Multicast Group

An example of the `xnic.json` file is shown below. Note the `ha` section has been added.

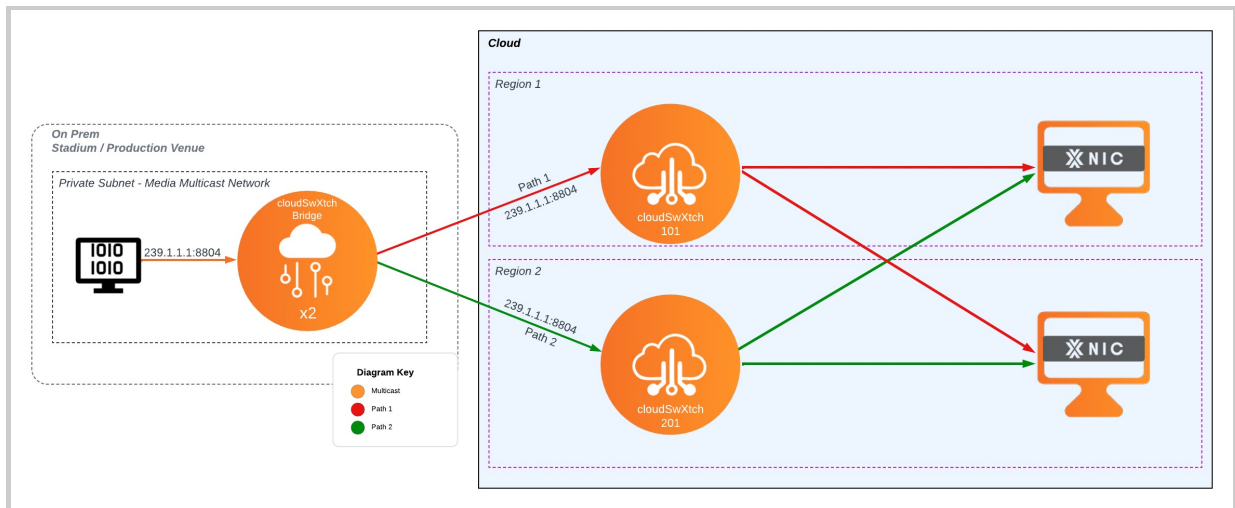
```
{
  "swxtch": "10.2.128.10",
  "controlInterface": "Ethernet 2",
  "dataInterface": "Ethernet",
  "dataPort": 9999,
  "xnicType": 2,
  "dataPlaneSpecs": {
    "verbosity": 0,
    "virtualInterface": {
      "name": "swxtch-tun0",
      "ip": "172.30.0.0",
      "subnet": "255.255.0.0",
      "mtu": 4096
    },
    "bpfPrograms": [
    ]
  },
  "ha": {
    "maxTimeToBufferPacketsMs": 50,
    "bufferSizeInPackets": 131072,
    "protocol": "swxtch"
  },
  "statsReportWait": 60
}
```

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.

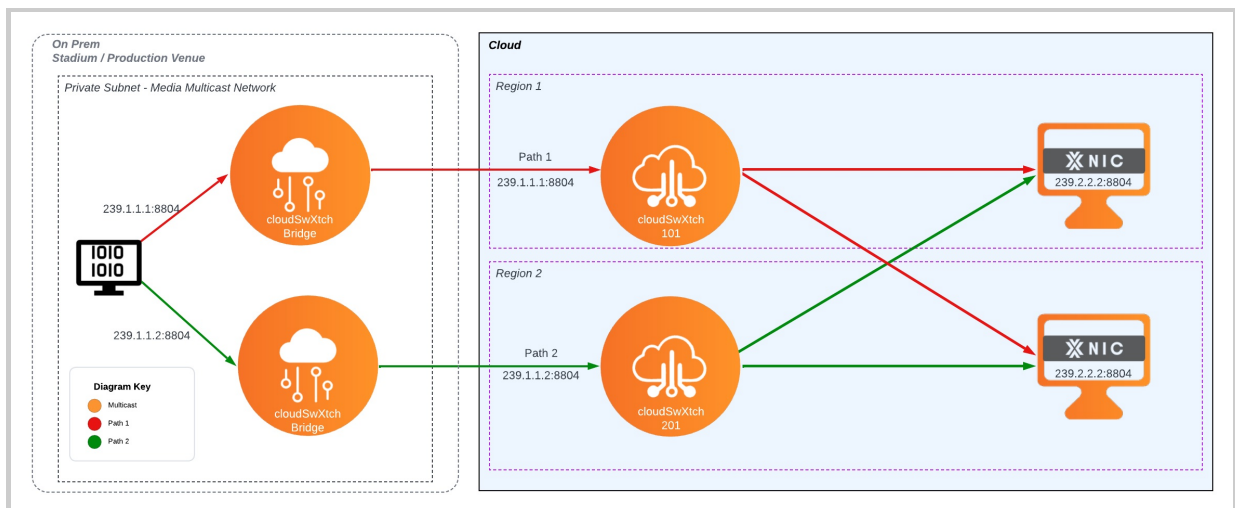
- **MaxTimeToBufferPackets_ms** - how long to buffer packets before declaring it as lost.
- **BufferSizeInPackets**- the max number of packets that can be buffered.
- **Protocol**- how to parse the packet. The available options are **swxtch** or **rtp**.
 - **swxtch** = This can be used when the xNIC 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:



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.

```
{
  "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.

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

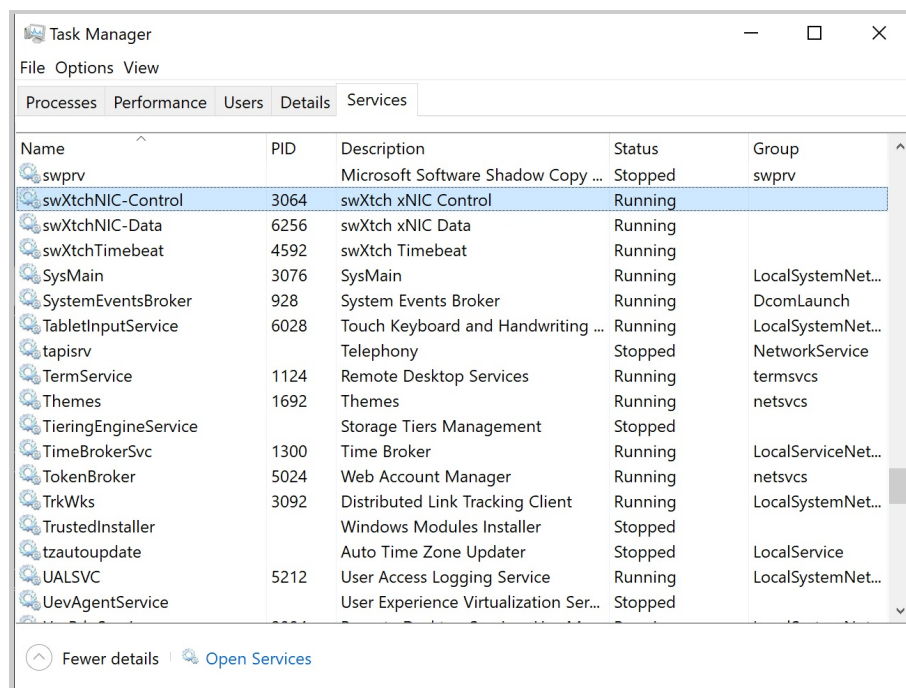
},
  "streamSpecs": {
    "mmcProducerEnable": false,
    "multipleMulticastGroups": [
      {
        "parent": "239.2.2.2:8804",
        "children": [
          "239.1.1.1:8804",
          "239.1.1.2:8804"
        ]
      },
      {
        "parent": "239.3.3.3:8804",
        "children": [
          "239.1.1.3:8804",
          "239.1.1.4:8804"
        ]
      }
    ]
  }
},
]

```

3. Save the **xnic.json** file.

4. Restart the **swXtch-NIC Control** service.

- 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
<pre>sudo service swxtch-xnic-control restart</pre>	

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.

None	Copy
<pre>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.</pre>	

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

None	Copy
<pre>swx mesh create -i meshconfig.json -s 10.2.128.5</pre> OUTPUT: Validating mesh.. Mesh succesfully created.	

Below is an example of a meshconfig.json file:

None	Copy
<pre>{ "name": "customer-mesh", "switches": ["10.2.128.5", "10.2.162.4"] }</pre>	

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

None	Copy
<pre>swx mesh add-swxtch -s 10.2.128.10 -a 10.1.1.6</pre> 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.

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"
        }
      },
      "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.

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
<pre>swx mesh remove-swxtch -s 10.1.1.6</pre> 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
<pre>swx mesh destroy -s 10.2.128.10</pre> Validating that the mesh was destroyed. Mesh successfully destroyed.	

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
  }
}
```

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

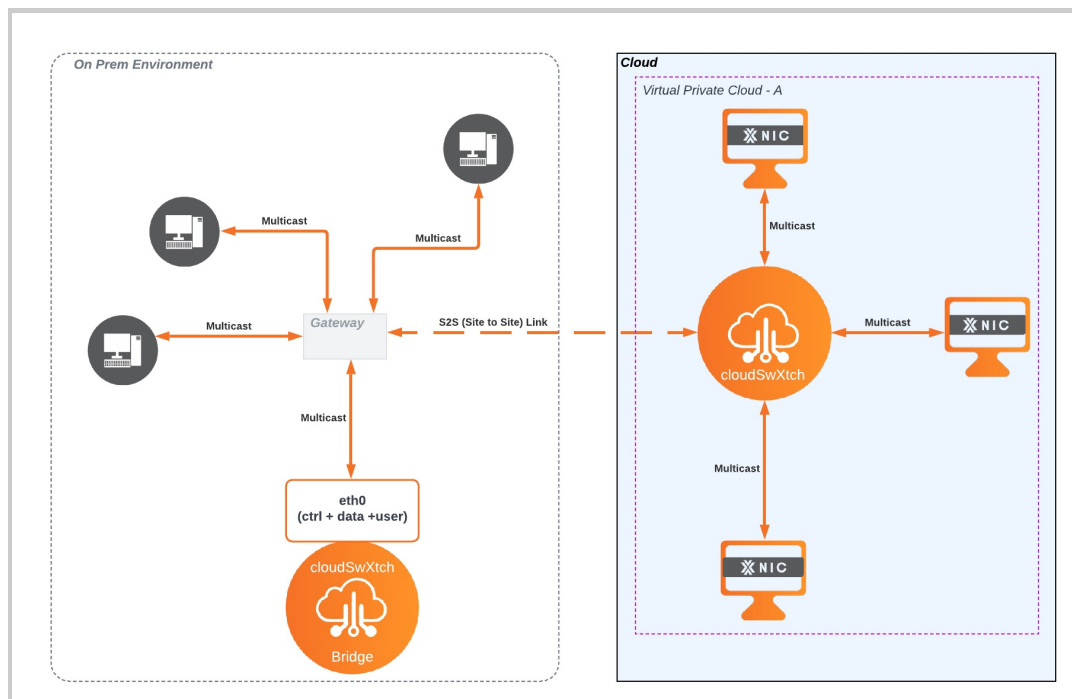
Alternatively, users can use a `/status` API call to get the `swxtch-bridge.json` file. To do this, use the following command replacing the `<bridge-ctrl-ip>` with the IP address of the bridge:

Bash

Copy

```
curl -x GET http://<bridge-ctrl-ip>/status
```

For a single interface



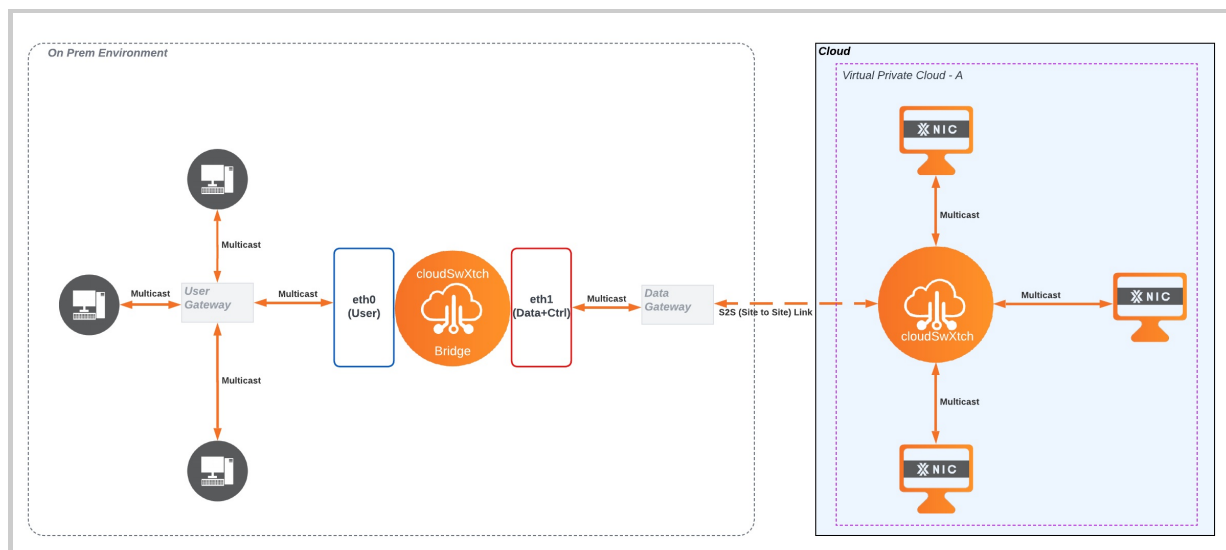
To accomplish a single interface configuration for your cloudSwXtch Bridge Type 2, users will need to specify the same InterfaceName for Ctrl, Data and User in the `swxtch-bridge.json` file. In the example, each are assigned to `eth0`.

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:

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.

Bash	Copy
<pre>{ "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 } }</pre>	

After modifying the configuration file, restart the swxtch-bridge2 service with the following command:

Bash	Copy
<pre>sudo systemctl restart swxtch-bridge2.service</pre>	

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

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
<pre>"dataGatewayIp": "192.168.1.2",</pre>	

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	Copy
<code>--input multicast://<multicast-group-ip>:<nic-ip>:<multicast-group-port></code>	

Multiple multicast groups can be specified by separating them with commas:

PowerShell	Copy
<code>--input multicast://<multicast-group-ip>:<nic-ip>:<multicast-group-port>,multicast://<multicast-group-ip>:<nic-ip>:<multicast-group-port></code>	

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	Copy
<code>--output udp://<swxtch-data-ip>:9999</code>	

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:

PowerShell	Copy
<code>--override-multicast-sender-ip <bridge-source-address> --last-leg</code>	

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
<pre>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</pre>	

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
<pre>--core 1 --core-count 2</pre>	

Where core sets the starting core index (from 0) and core-count sets the number of cores to use.

Protocol Conversion and Fanout

Configuring Protocol Conversion and Fanout

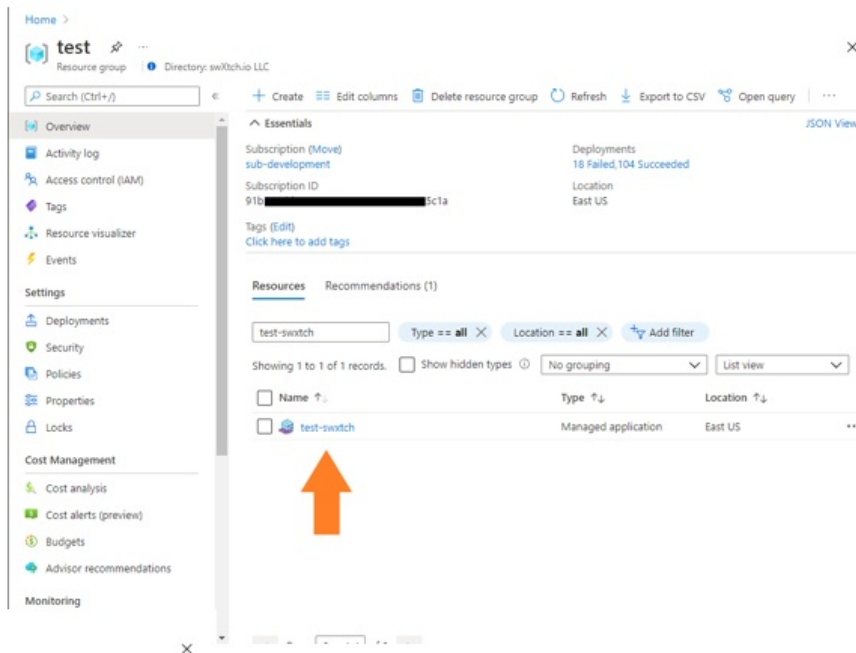
There are two options for configuring Protocol Conversion and Fanout: via wXcked Eye or via the API. For more information, please see the following articles:

- [Protocol Conversion and Fanout with wXcked Eye](#)
- [Configuration API](#)

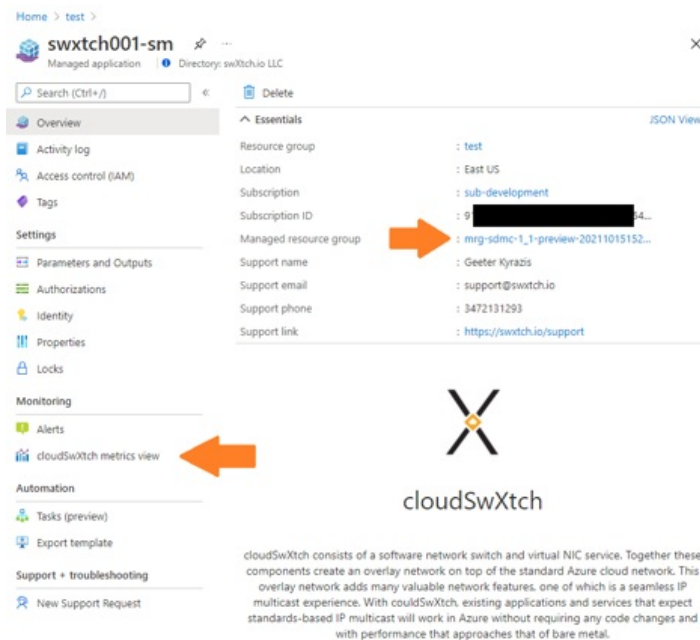
Azure Monitoring

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.



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.

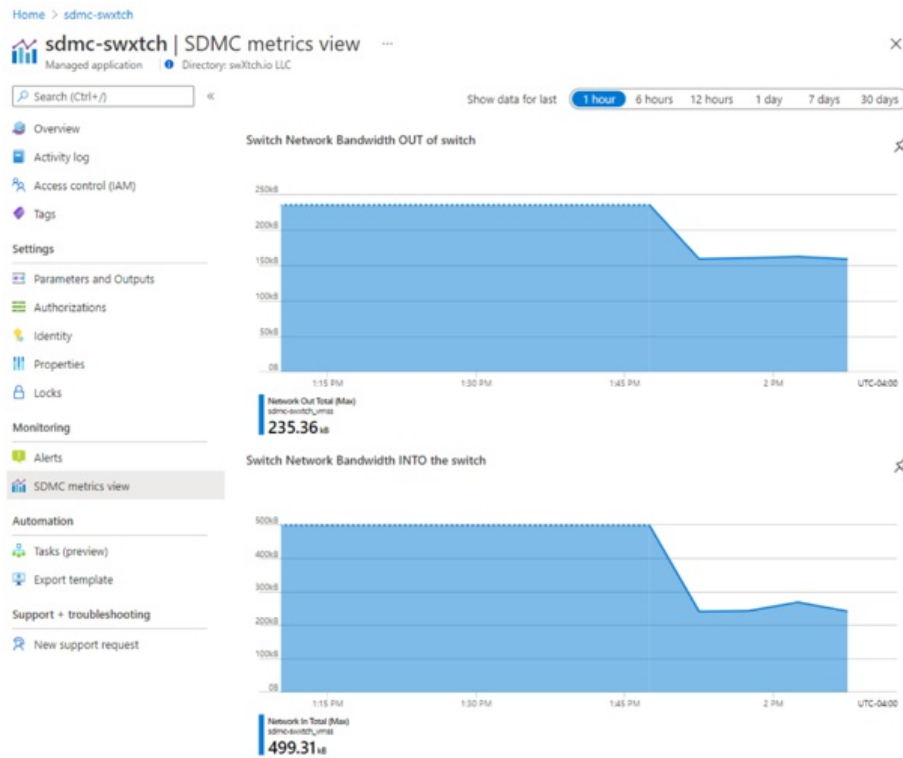


cloudSwXtch metrics view

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.



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

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 <i>from</i> the SDMC switch
DataPort	9999	Local port used for data traffic <i>from</i> the SDMC switch

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
<pre>curl http://localhost/prometheus/metrics</pre>	

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
<pre>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_rep1",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_rep1",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_rep1",host="10.2.192.23"} 0 # HELP swx_core_rxBridgePacketCount Bridge packets received into the swXtch # TYPE swx_core_rxBridgePacketCount counter swx_core_rxBridgePacketCount{category="swxtch_rep1",host="10.2.192.23"} 0 # HELP swx_core_rxByteCount Bytes received into the swXtch # TYPE swx_core_rxByteCount counter swx_core_rxByteCount{category="swxtch_rep1",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_rep1",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_rep1",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_rep1",host="10.2.192.23"} 505406 # HELP swx_core_rxUnicastByteCount Unicast bytes received into the swXtch # TYPE swx_core_rxUnicastByteCount counter</pre>	


```

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="10.2.192.23"} 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
swx_xnic_byteCounters_rxMulticastCount{category="swxtch_xnic",host="aks-nodepool11-
23164585-vmss000018"} 7200
swx_xnic_byteCounters_rxMulticastCount{category="swxtch_xnic",host="aks-nodepool11-
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-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
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="DSd-agent-101"} 52
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="DSd-agent-102"}
494783
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="DSd-agent-104"}
490677
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="DSd-agent-105"} 54
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="aks-nodepool1-
23164585-vmss000018"} 52
swx_xnic_packetCounters_rxTotalCount{category="swxtch_xnic",host="aks-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="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_txIcmpCount{category="swxtch_xnic",host="DSd-agent-105"} 0
swx_xnic_packetCounters_txIcmpCount{category="swxtch_xnic",host="aks-nodepool1-
23164585-vmss000018"} 0
swx_xnic_packetCounters_txIcmpCount{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="DSd-agent-101"} 0
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="DSd-agent-102"} 0
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="DSd-agent-104"} 462
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="DSd-agent-105"} 54
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="aks-nodepool1-
23164585-vmss000018"} 504911
swx_xnic_packetCounters_txTotalCount{category="swxtch_xnic",host="aks-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",cloudSwxtchVersion="dev.c1
oudswxtch.2.1.1.10",groupId="10.2.131.255",host="DSd-agent-
101",hostAddress="10.2.128.29",index="3",osDistribution="Ubuntu 22.04",xNicType="t2"}
1116
swx_xnic_rxMulticastGroups_byteCount{category="swxtch_xnic",cloudSwxtchVersion="dev.c1
oudswxtch.2.1.1.10",groupId="10.2.131.255",host="DSd-agent-
101",hostAddress="10.2.128.29",index="5",osDistribution="Ubuntu 22.04",xNicType="t2"}
1116
swx_xnic_rxMulticastGroups_byteCount{category="swxtch_xnic",cloudSwxtchVersion="dev.c1
oudswxtch.2.1.1.10",groupId="10.2.131.255",host="DSd-agent-
101",hostAddress="10.2.128.29",index="6",osDistribution="Ubuntu 22.04",xNicType="t2"}
1100
swx_xnic_rxMulticastGroups_byteCount{category="swxtch_xnic",cloudSwxtchVersion="dev.c1
oudswxtch.2.1.1.10",groupId="10.2.131.255",host="DSd-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",cloudSwxtchVersion="dev.
cloudswxtch.2.1.1.10",groupId="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",cloudSwxtchVersion="dev.
cloudswxtch.2.1.1.10",groupId="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",cloudSwxtchVersion="dev.
cloudswxtch.2.1.1.10",groupId="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",cloudSwxtchVersion="dev.
cloudswxtch.2.1.1.10",groupId="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",cloudSwxtchVersion="dev.cl
oudswxtch.2.1.1.10",groupId="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"} 2988
swx_xnic_txMulticastGroups_byteCount{category="swxtch_xnic",cloudSwxtchVersion="dev.cl
oudswxtch.2.1.1.10",groupId="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"} 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",cloudSwxtchVersion="dev.
cloudswxtch.2.1.1.10",groupId="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",cloudSwxtchVersion="dev.
cloudswxtch.2.1.1.10",groupId="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.

STEP TWO: Update Prometheus Directory for cloudSwXtch

Attached is an example prometheus.yml 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.

1. **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
<pre>docker run --network host -v ~/prometheus:/etc/prometheus prom/prometheus</pre>	

5. Use this command to run Prometheus automatically upon reboot. (Preferred method for a production environment.)

Plaintext	Copy
<pre>docker run --network host --restart always -v ~/prometheus:/etc/prometheus prom/prometheus</pre>	

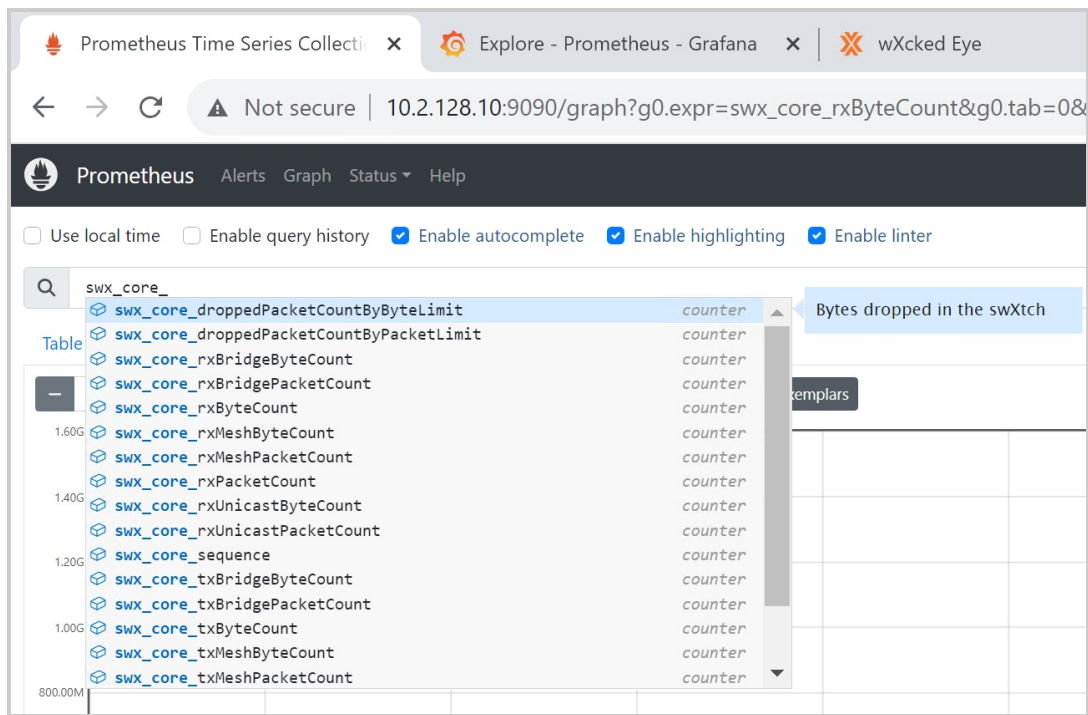
STEP THREE: Access Prometheus UI

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:

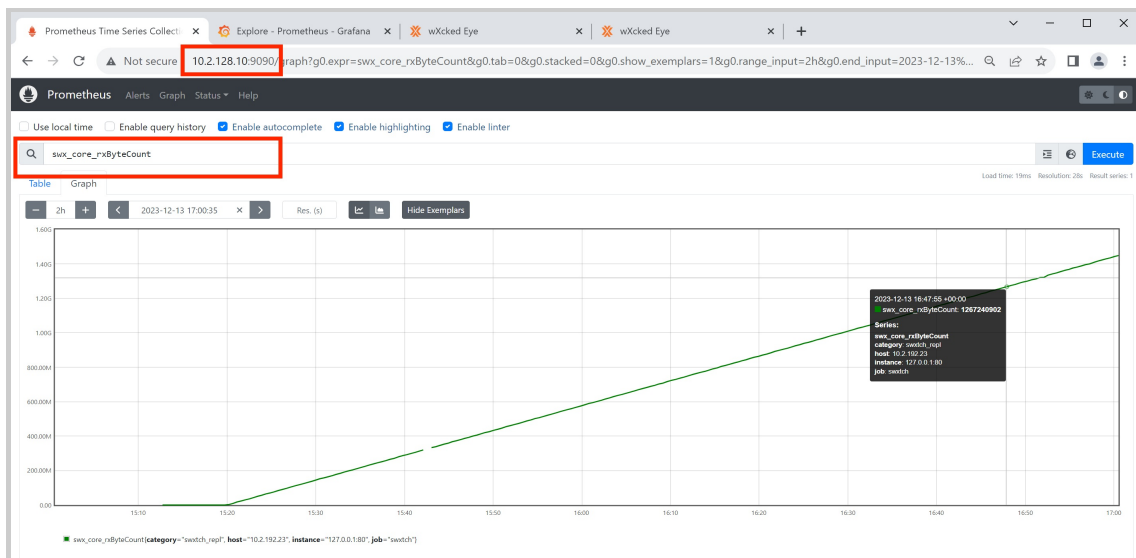
Plaintext	Copy
<pre>http://<prometheus-IP>:9090/</pre>	

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.



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](#)



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:

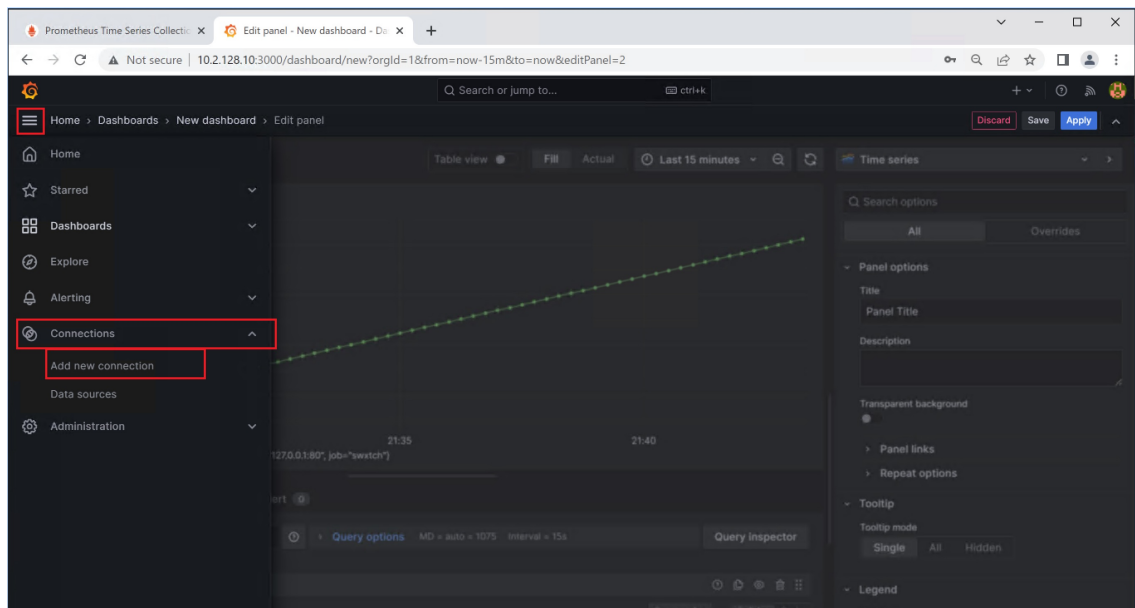
Plaintext

Copy

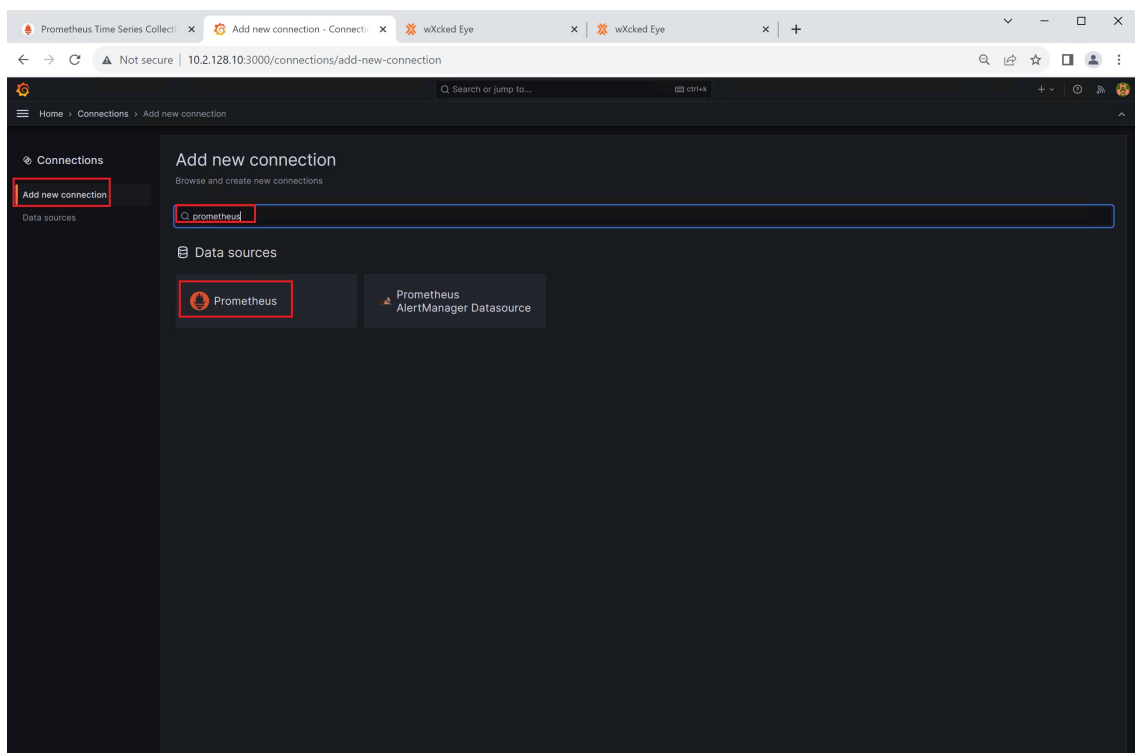
`http://<Grafana-IP>:3000/`

Note: Replace the <Grafana-IP> with the IP address of your Grafana instance.

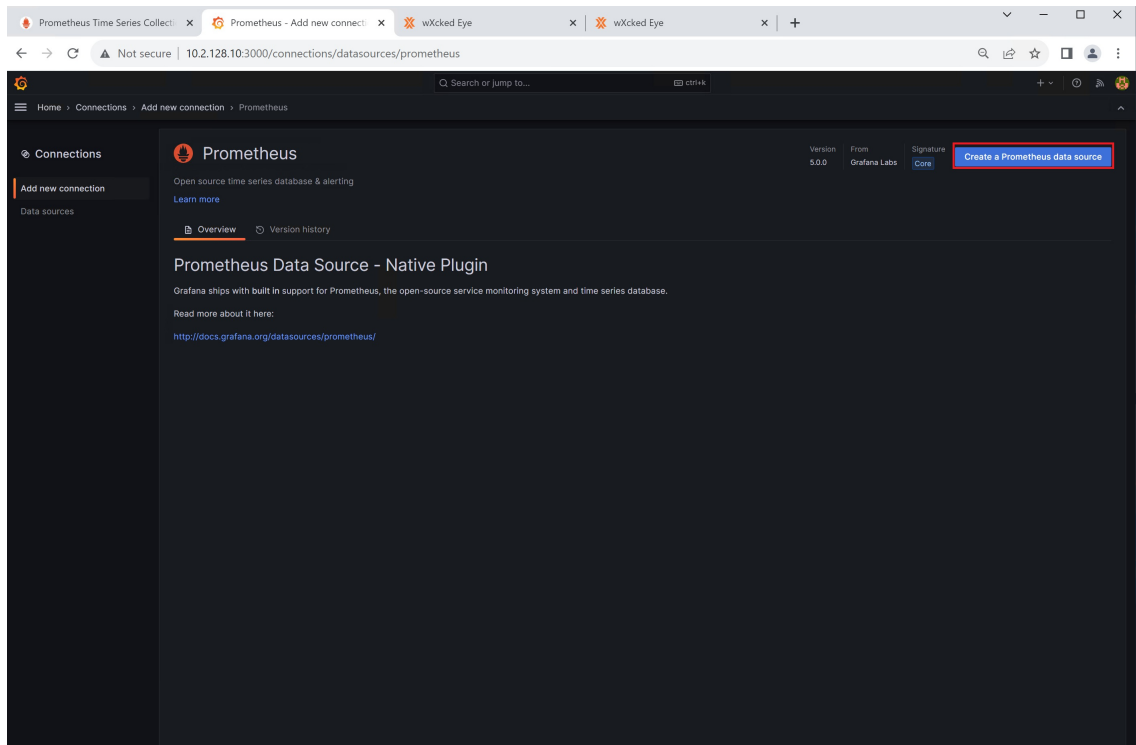
2. Sign in as a user **admin** and the password **admin**.
3. Click on the three horizontal lines next to **Home** to get additional options.
4. Select **Connection**.
5. Click **Add new connection**.



6. Search **Prometheus** on the **Add new connection** page and select it from the options.

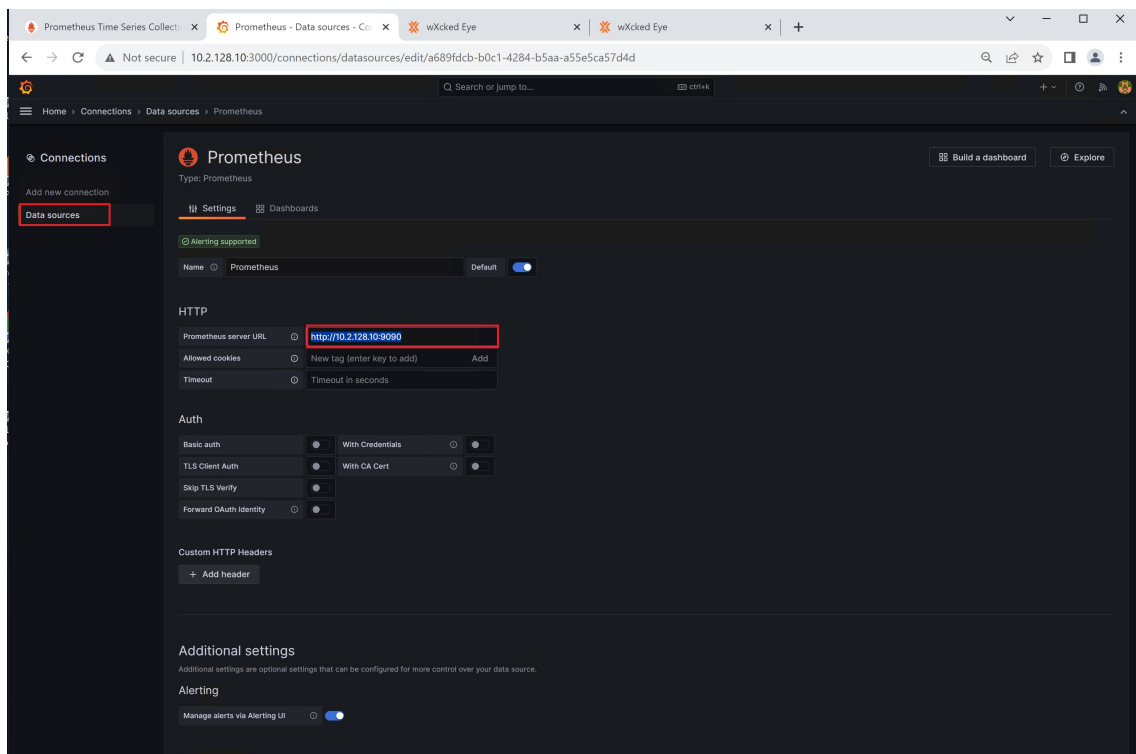


7. Click **Create a Prometheus data source**.

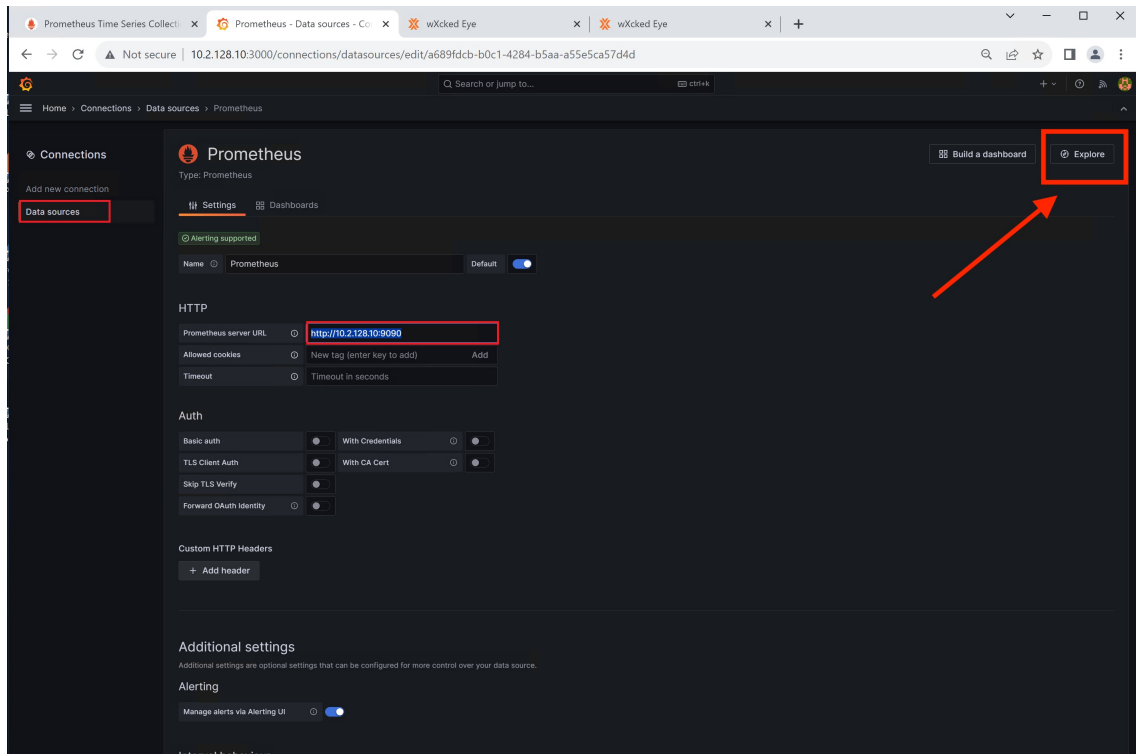


8. Go to **Data Sources** and select the **Prometheus** data source created.

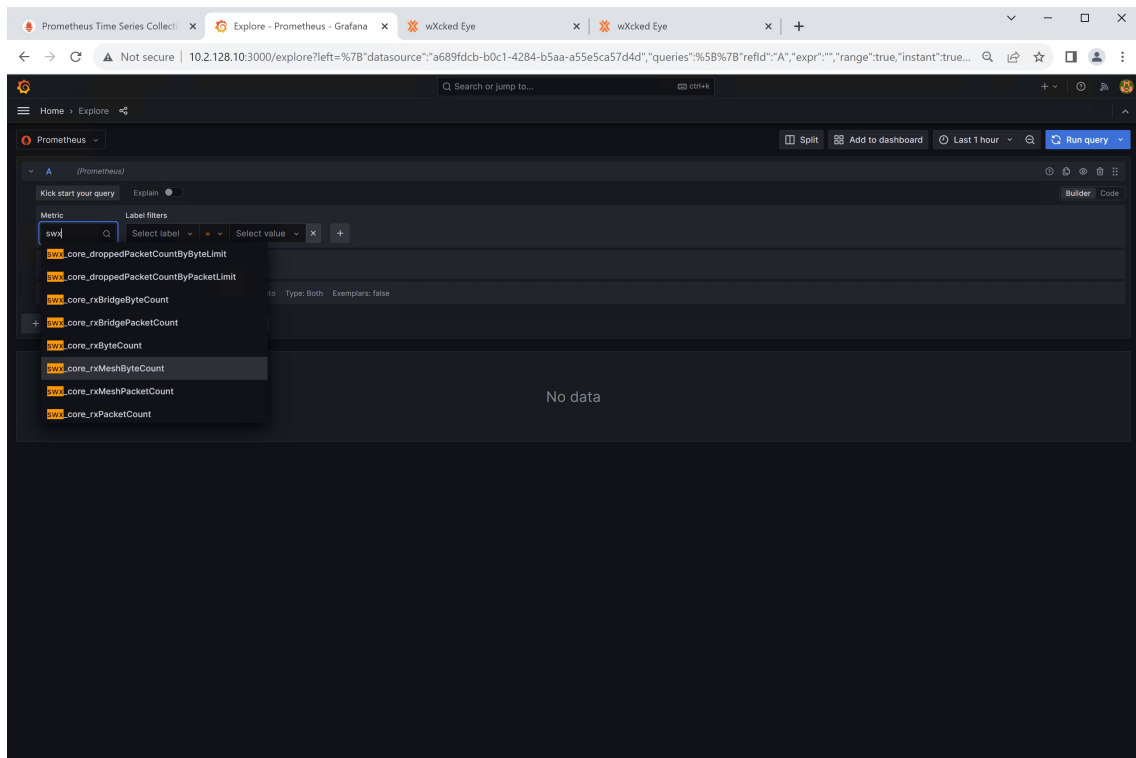
9. Enter the **Prometheus server URL**. This should include the **IP address** of your **Prometheus instance**.



10. Click Explore.

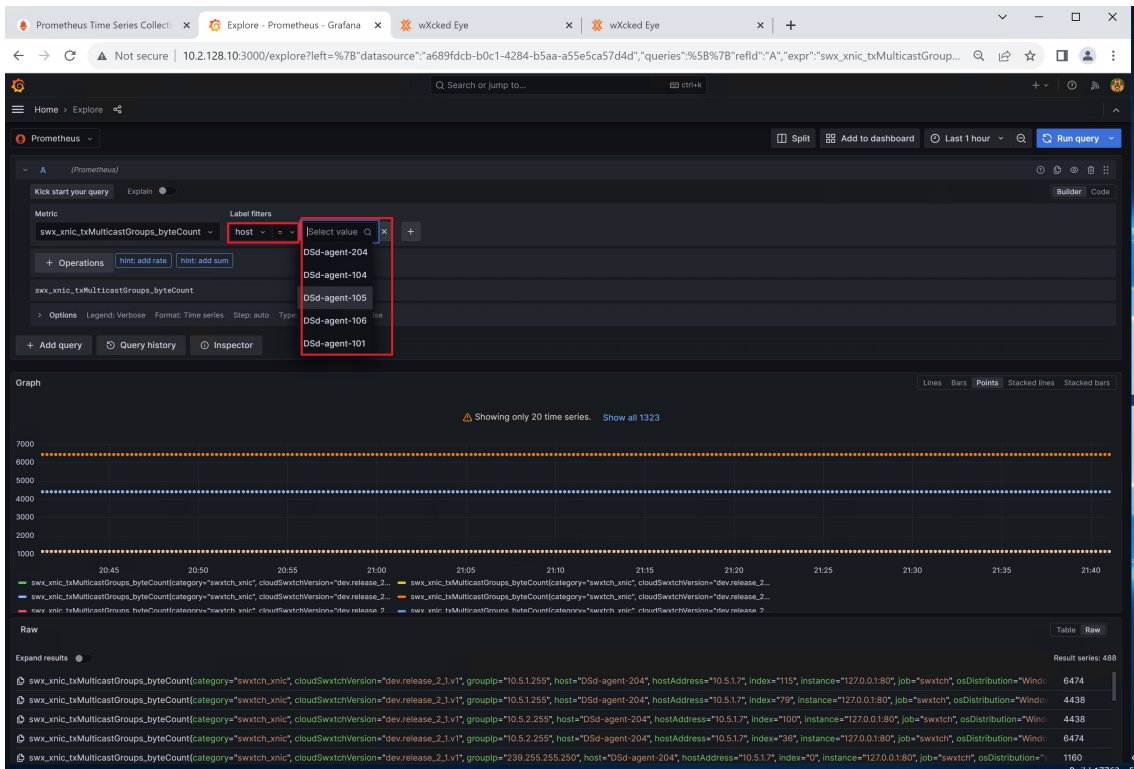


11. In the search bar under **Metric**, use the prefix “swx” to populate a list of potential data fields.

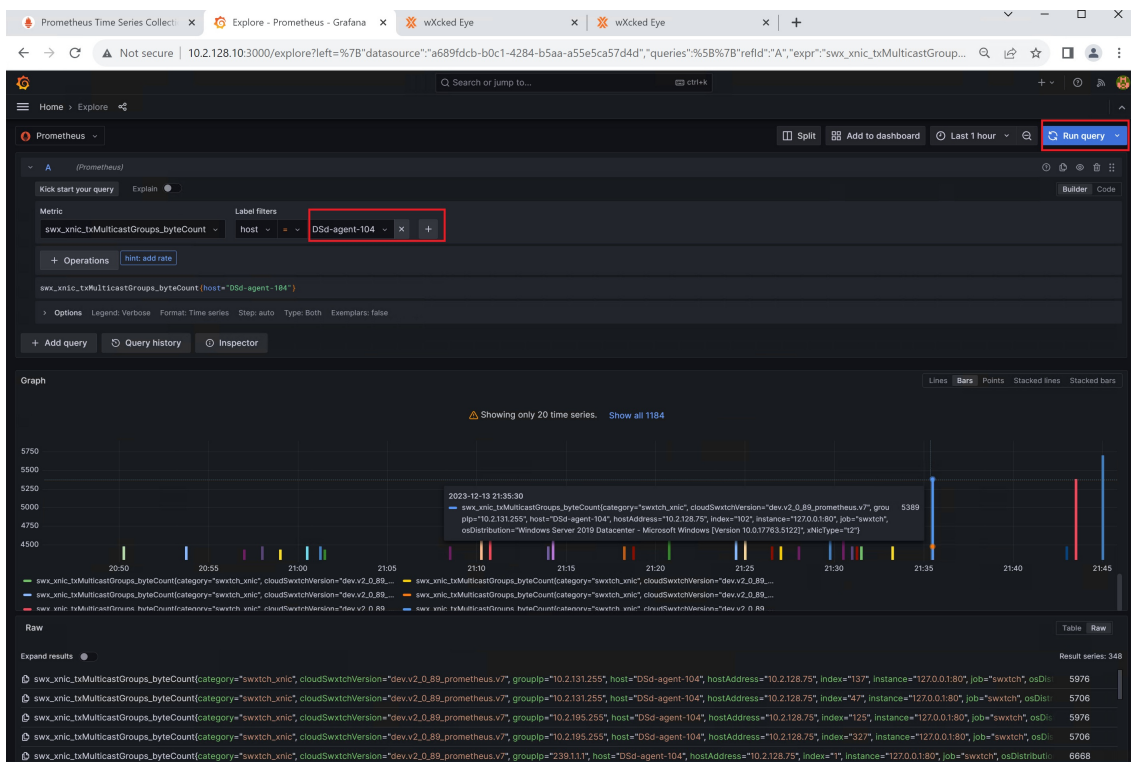


12. Scroll through the list and select what is desired.

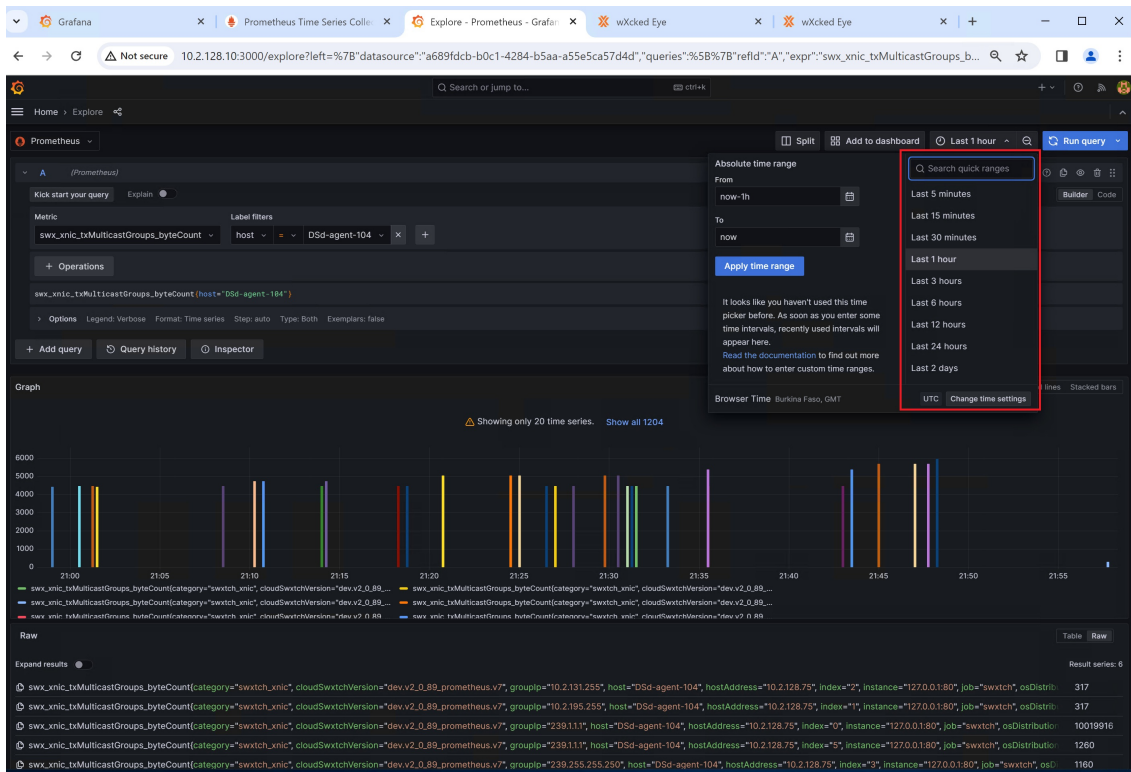
13. Run the query. The results will appear below.



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.



Testing cloudSwXtch

Testing

It is easy to test the functionality and performance of a cloudSwXtch multicast network. Included within the xNIC installation are utilities that can be used to verify both the functionality and performance of your network.

- `swtch-perf` – used to produce and consume unicast and multicast traffic
- `swtch-top` – shows detailed system statistics in the console

Additionally, the metrics view in the cloudSwXtch information page (see the Advanced cloudSwXtch Operation section below) shows global network traffic into and out of the cloudSwXtch instance.

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

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
<pre>swxtch-perf producer --sendto <MC_ADDRESS:DEST_PORT> --nic <NETWORK_INTERFACE></pre>	

swxtch-perf consumer will pick up the traffic generated by the producer(s) in the network.

None	Copy
<pre>swxtch-perf consumer --recvfrom <MC_ADDRESS:DEST_PORT> --nic <NETWORK_INTERFACE></pre>	

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
<pre> swxtch-perf -h Usage: swxtch-perf [options] command Positional arguments: command [producer consumer] suported commands 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] --dbg Enables more information in the logs [default: false] --show-full-packet-bps Shows the bps with all headers included </pre>	

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
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 timestamp propagation and measurement of latency.	false	true:false	Both	Linux
broadcast	Sets swtch-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 argument 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 :

None	Copy
<pre>Linux: swtch-perf producer --sendto 239.1.1.1:3490 --pps 1000 --nic swtch-tun0 Windows: swtch-perf producer --sendto 239.1.1.1:3490 --pps 1000 --nic swtch-tun</pre>	

Example with results:

None

Copy

swtch-perf producer --sendto 239.1.1.1:3490 --pps 1000 --nic swtch-tun0

Trying to reach a packet-rate of 1000 pps

swtch-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 :

None	Copy
<pre>Linux: swtch-perf consumer --recvfrom 239.1.1.1:3490 --nic swtch-tun0 Windows: swtch-perf consumer --recvfrom 239.1.1.1:3490 --nic swtch-tun</pre>	

Example with results:

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.

EXAMPLE

Single Producer, Single Consumer, and broadcast

Run this command on a VM to create a broadcast

None	Copy
<pre>Linux: swtch-perf producer --broadcast --nic eth1 --pps 1000 --broadcast-port 1234 Windows: swtch-perf producer --broadcast --nic 'Ethernet 2' --pps 1000 --broadcast-port 1234</pre>	

Example with results:

None	Copy
<pre>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</pre>	

Run this command on one of the VMs to listen for broadcast

None	Copy
<pre>Linux: swxtch-perf consumer --broadcast --nic eth1--pps 1000 Windows: swxtch-perf consumer --broadcast --nic 'Ethernet 3' --pps 1000</pre>	

swxtch-top

WHAT TO EXPECT

swxtch-top is one of the utility applications included in the xNIC installation. It can be run from the console of any VM that has an 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
<pre>swxtch-top --swxtch <cloudSwXtch-IP></pre>	

Example:

Bash	Copy
<pre>swxtch-top --swxtch 10.5.1.6</pre>	

From the cloudSwXtch, users can enter the following command:

Bash	Copy
<pre>sudo /swxtch/swxtch-top --swxtch localhost</pre>	

Navigating swxtch-top Dashboard

```

Administrator: Windows PowerShell

Information - dev
dsd-core-200 v2.0.89(DRS-Azure-200set-all) Auth. Type License File Marketplace
Cloud AZURE Bandwidth 0 bps/6.06 bps
SubscriptionId b10209ad-ad22-4c26-8aef-be93b2f0bb58 Clients 5/50 | Bridges 0/2
ResourceGroupName TEST-DONNA-200-RG
SwXtchId 9a1d981e-97d0-40cc-86e1-645eacf124e6
Status OK OS CPU USAGE 87% | OS MEM USAGE 51%

Totals
Producers 107 pps (1.2M bps) | Consumers 117 pps (1.3M bps)
Bridge RX 0 pps (0 bps) | Bridge TX 0 pps (0 bps)
HA RX 0 pps (0 bps) | HA TX 0 pps (0 bps)
swXtch RX 0 pps (0 bps) | swXtch TX 0 pps (0 bps)

xNIC clients (5)
Name ip Version OS RX pps RX bps TX pps TX bps HA
DSd-agent-105 10.2.192.15 v2.0.89 (t2) Windows Server 2019 Datacen... 117 1.3M 0 0 N
DSd-agent-106 10.2.192.32 v2.0.89 (t2) Windows Server 2019 Datacen... 0 0 107 1.2M N
DSd-agent-201 10.5.2.4 v2.0.89 (t2) Ubuntu 20.04 0 0 0 0 2
DSd-agent-202 10.5.2.5 v2.0.89 (t2) Ubuntu 20.04 0 0 0 0 2
DSd-agent-204 10.5.2.7 v2.0.89 (t2) Windows Server 2019 Datacen... 0 0 0 0 N

1: xNIC 2: Streams 3: Mesh 4: HA Paths 5: HA Summary 6: HA 7: Configuration 8: Timing Nodes 9: Adaptors 0: Hardware

```

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 10 different views:

1. xNIC
2. Streams
3. Mesh
4. HA Paths
5. HA Summary
6. HA
7. Configuration
8. Timing Nodes
9. Adaptors
10. (0) Hardware

The default is the 1: xNIC view. To switch between them, simply enter the number that matches the view type. For example, to toggle to "0: Hardware," enter in the number 0 on your keypad.

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

Information - dev					
dsd-core-200	v2.0.89(DRS-Azure-200set-all)	Auth. Type	License File	Marketplace	
Cloud	AZURE	Bandwidth	0 bps/6.0G bps		
SubscriptionId	b10209ad-ad22-4c26-8aef-be93b2f0bb58	Clients	5/50	Bridges	0/2
ResourceGroupName	TEST-DONNA-200-RG				
SwXtchId	9a1d981e-97d0-40cc-86e1-645eac124e6				
Status	OK	OS CPU USAGE 78% OS MEM USAGE 51%			

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

Information - v2.0.89					
ip-172-41-131-244	v2.0.89(Full)	Auth. Type	License File	Marketplace	
Cloud	AWS	Bandwidth	0 bps/unlimited		
AccountId	639720666639	Clients	0/unlimited	Bridges	0/4
Region	us-west-2				
SwXtchId	i-0b32c832a8680ae91				
Status	OK	OS CPU USAGE 34% OS MEM USAGE 13%			

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.

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

Information - v2.0.89					
fs3a0sd10	v2.0.89(Full)	Auth. Type	License File		
Cloud	GCP	Bandwidth	0 bps/unlimited		
Id	7116914649760725139	Clients	0/unlimited	Bridges	0/4
Zone	us-central1-a				
SwXtchId	9c8a2f5a-b7bb-22e9-65f2-d7ca84e5dbce				
Status	OK	OS CPU USAGE 38% OS MEM USAGE 13%			

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

Totals					
Producers	106 pps	(1.2M bps)	Consumers	371 pps	(2.0M bps)
Bridge RX	27 pps	(297.3K bps)	Bridge TX	0 pps	(0 bps)
HA RX	0 pps	(0 bps)	HA TX	0 pps	(0 bps)
swXtch RX	134 pps	(1.5M bps)	swXtch TX	139 pps	(1.5M bps)

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)**.

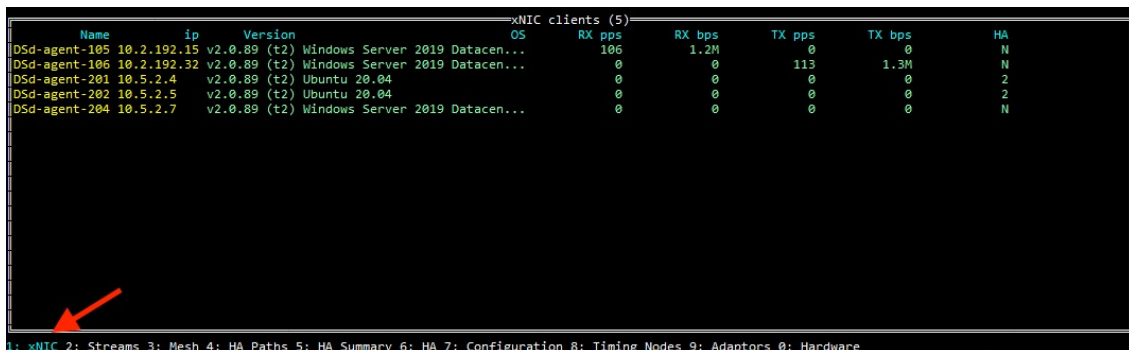
Please note: If your cloudSwXtch is part of a mesh or a bridge, the ingress/egress will show data in those sections.

- **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.
- **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 the numeric value for that view.

1: xNIC view



Name	ip	Version	OS	RX pps	RX bps	TX pps	TX bps	HA
DSd-agent-105	10.2.192.15	v2.0.89 (t2)	Windows Server 2019 Datacen...	106	1.2M	0	0	N
DSd-agent-106	10.2.192.32	v2.0.89 (t2)	Windows Server 2019 Datacen...	0	0	113	1.3M	N
DSd-agent-201	10.5.2.4	v2.0.89 (t2)	Ubuntu 20.04	0	0	0	0	2
DSd-agent-202	10.5.2.5	v2.0.89 (t2)	Ubuntu 20.04	0	0	0	0	2
DSd-agent-204	10.5.2.7	v2.0.89 (t2)	Windows Server 2019 Datacen...	0	0	0	0	N

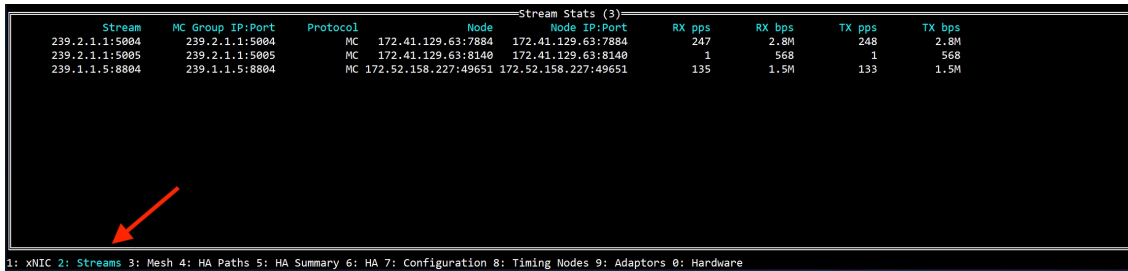
1: xNIC 2: Streams 3: Mesh 4: HA Paths 5: HA Summary 6: HA 7: Configuration 8: Timing Nodes 9: Adaptors 0: Hardware

This view shows all the xNIC clients that are connected to the cloudSwXtch. 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)
- **RX pps** - The total ingress packets per second that the xNIC is receiving.
- **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](#)

2: Streams



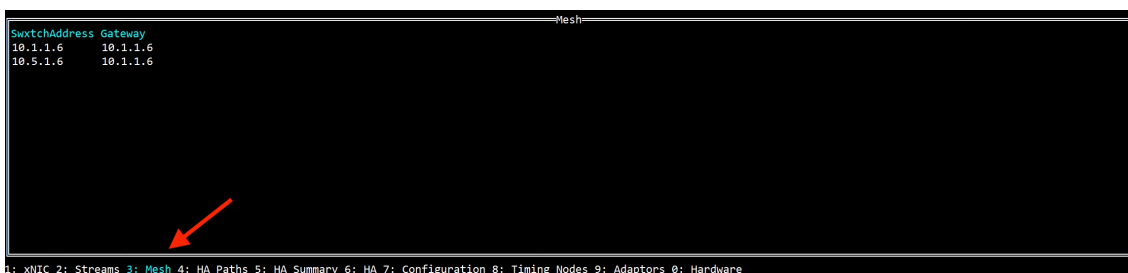
Stream Stats (3)									
Stream	MC Group	IP:Port	Protocol	Node	Node IP:Port	RX pps	RX bps	TX pps	TX bps
239.2.1.1:5004	239.2.1.1:5004		MC	172.41.129.63:7884	172.41.129.63:7884	247	2.8M	248	2.8M
239.2.1.1:5005	239.2.1.1:5005		MC	172.41.129.63:8140	172.41.129.63:8140	1	568	1	568
239.1.1.5:8804	239.1.1.5:8804		MC	172.52.158.227:49651	172.52.158.227:49651	135	1.5M	133	1.5M

1: xNIC 2: Streams 3: Mesh 4: HA Paths 5: HA Summary 6: HA 7: Configuration 8: Timing Nodes 9: Adaptors 0: Hardware

This view shows all the multicast groups that are being received and transmitted by the cloudSwXtch. This view includes:

- **Name** - The name is the stream IP and Port. For Multicast, it is the MC Group IP:Port. For broadcast, it is the Broadcast IP:Port.
- **Src IP:Port**: IP address of where the data is flowing from (the producer)
- **Protocol** - Multicast or Broadcast
- **RX pps**: The total ingress packets per second being received by the multicast group.
- **RX bps**: The total ingress bits per second that is received by the multicast group.
- **TX pps**: The total egress packets per second that is transmitted by the multicast group.
- **TX bps**: The total egress bits per second that that is transmitted by the multicast group.
- **Destinations**: The number of destinations receiving the multicast group.

3: Mesh



Mesh	
SwxtchAddress	Gateway
10.1.1.6	10.1.1.6
10.5.1.6	10.1.1.6

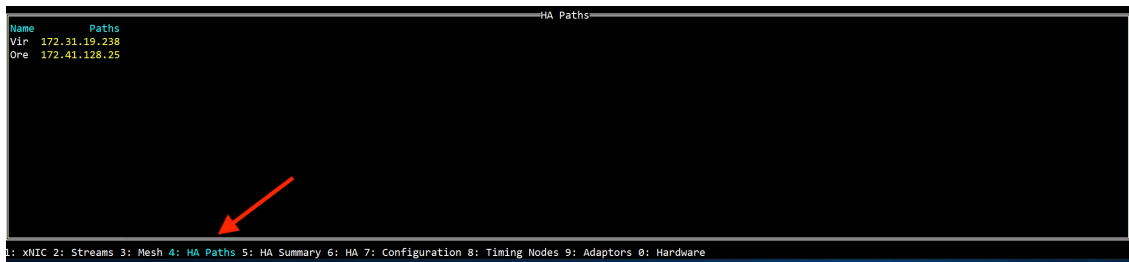
1: xNIC 2: Streams 3: Mesh 4: HA Paths 5: HA Summary 6: HA 7: Configuration 8: Timing Nodes 9: Adaptors 0: Hardware

This view shows all the cloudSwXtches that are in a mesh. It only shows data if a mesh has been configured.

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.
- **Gateway** - The IP address that serves as entry/exit point for traffic between networks.

4: HA Paths



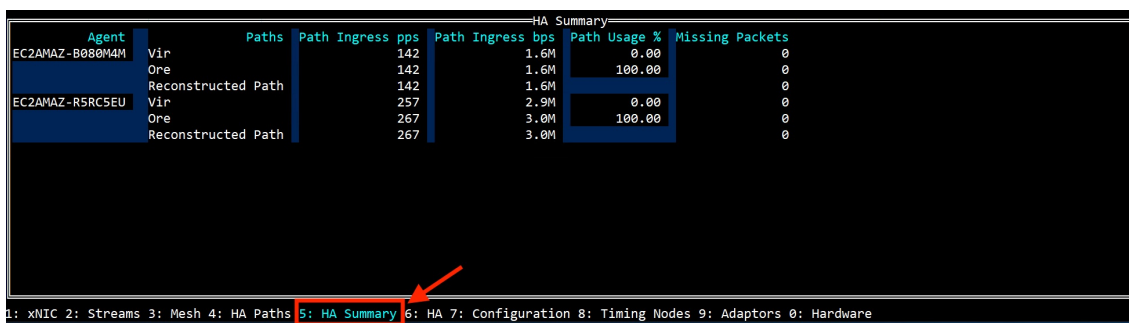
Name	Paths
Vir	172.31.19.238
Ore	172.41.128.25

1: xNIC 2: Streams 3: Mesh 4: HA Paths 5: HA Summary 6: HA 7: Configuration 8: Timing Nodes 9: Adaptors 0: Hardware

This view shows all the paths for high availability. It will only show data if High Availability has been configured. See [High Availability](#) for configuration details. This view includes:

- **Name** - The name of the Path
- **Paths** - The cloudSwXtches that are in the path. In this example, both paths have a single cloudSwXtch associated with it.

5: HA Summary



Agent	Paths	Path Ingress pps	Path Ingress bps	Path Usage %	Missing Packets
EC2AMAZ-B080M4M	Vir	142	1.6M	0.00	0
	Ore	142	1.6M	100.00	0
	Reconstructed Path	142	1.6M		0
EC2AMAZ-R5RC5EU	Vir	257	2.9M	0.00	0
	Ore	267	3.0M	100.00	0
	Reconstructed Path	267	3.0M		0

1: xNIC 2: Streams 3: Mesh 4: HA Paths 5: HA Summary 6: HA 7: Configuration 8: Timing Nodes 9: Adaptors 0: Hardware

The HA Summary view shows a breakdown of high availability for the cloudSwxtch. This will only display data if High Availability has been configured. See [High Availability](#) for configuration details. This view includes:

- **Agent** - The agent that is receiving the multicast traffic.
- **Paths** - The paths that the multicast is taking as well as an outcome of the reconstructed path.
- **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 of the path that is used in the High Availability 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 any of the cloudSwXtches, the number will stop increasing but will not reset.

6: HA

Agent	Stream Src IP	Stream IP	Paths	Path Ingress pps	Path Ingress bps	Path Usage %	Missing Packets
EC2AMAZ-BB8M4M	172.41.129.63	239.2.1.1	Vir	152	1.7M	98.43	0
			One	152	1.7M	9.57	0
			Reconstructed Path	152	1.7M		0
EC2AMAZ-R5RC5EU	172.52.158.227	239.1.1.5	Vir	126	1.4M	98.97	0
			One	138	1.4M	9.03	0
			Reconstructed Path	126	1.4M		0

The HA view shows additional details for high availability. It will only show data if High Availability has been configured. See [High Availability](#) for configuration details. This view includes:

- **Agent** - The agent that is receiving the multicast.
- **Stream Src. IP** - The IP address of where the stream is coming from (the producer).
- **Stream IP** - The IP of the multicast stream.
- **Paths** - The paths that the multicast is taking as well as an outcome of the reconstructed path.
- **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.
- **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.
- **Path Usage %** - The percentage that the path is used in the highly available multicast group.

7. Configuration

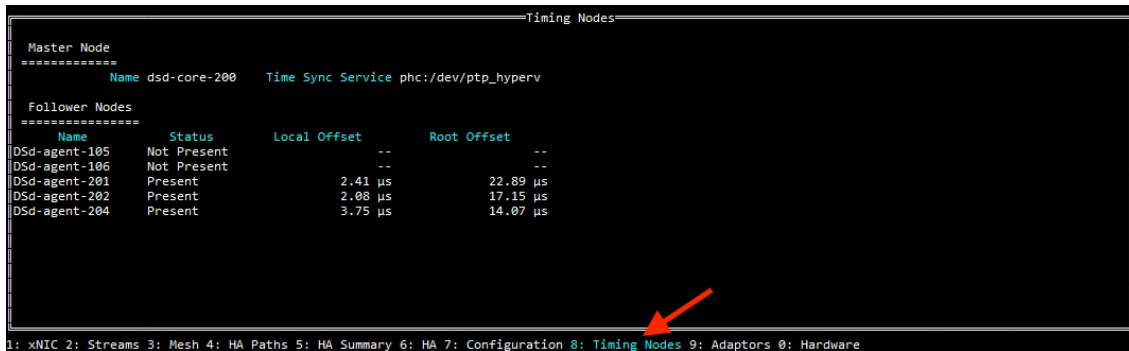
Entitlements	Mesh
Max Bandwidth	6.0G bps
Max Clients	50
Max Bridges	2
Mesh	true
HA	true
Fanout	true
ClockSync	true
wXckedEye	true
MajorVersionUpdate	true

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.

- **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.

8. Timing Nodes view



Timing Nodes			
Master Node			
=====			
Name	dsd-core-200	Time Sync Service	phc:/dev/ptp_hyperv
Follower Nodes			
=====			
Name	Status	Local Offset	Root Offset
DSd-agent-105	Not Present	--	--
DSd-agent-106	Not Present	--	--
DSd-agent-201	Present	2.41 µs	22.89 µs
DSd-agent-202	Present	2.08 µs	17.15 µs
DSd-agent-204	Present	3.75 µs	14.07 µs

1: xNIC 2: Streams 3: Mesh 4: HA Paths 5: HA Summary 6: HA 7: Configuration 8: **Timing Nodes** 9: Adaptors 0: Hardware

The Timing Nodes view displays information regarding the clock sync configuration for the cloudSwXtch. The page in swtch-top will only populate with information if the user has the PTP feature enabled.

In the example above, the cloudSwXtch (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
- **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.

9. Adaptors

Adaptors					
Protocol	Direction	Stream	Node	Listener	Port
RIST-LISTENER	ingress	239.4.6.7:3408	--	--	5687
UDP	egress	239.5.2.3:3450	10.2.128.44:5003	--	--
SRT-LISTENER	egress	225.1.1.1:1599	--	--	6000
RIST-CALLER	egress	239.2.3.1:5700	10.2.128.15:3401	--	--
RIST-LISTENER	egress	239.4.6.7:3408	--	--	5643
SRT-LISTENER	ingress	225.1.1.1:1599	--	--	1599
SRT-CALLER	ingress	239.5.2.3:3450	10.2.128.36:3000	--	--
RIST-CALLER	ingress	239.2.3.1:5700	10.2.128.75:1599	--	--
UDP	ingress	239.4.5.6:2000	--	--	5000
UDP	egress	239.5.2.3:3450	10.2.192.32:5001	--	--
SRT-CALLER	egress	239.5.2.3:3450	10.2.128.37:5004	--	--

1: xNIC 2: Streams 3: Mesh 4: HA Paths 5: HA Summary 6: HA 7: Configuration 8: Timing Nodes 9: Adaptors 0: Hardware

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.

0. Hardware

Hardware			
Control		Data	
meta		meta	
ipAddress	10.5.1.6	ipAddress	10.5.2.6
ipBroadcast	10.5.1.255	ipBroadcast	10.5.2.255
ipSubnet	10.5.1.0	ipSubnet	10.5.2.0
macAddress	00:22:48:24:75:06	macAddress	00:22:48:24:77:db
subnetMask	255.255.255.0	subnetMask	255.255.255.0
os		os	
driver	hv_netvsc	driver	mlx5_core
masterof		masterof	enp7964s2
mtu	1500	mtu	1500
name	eth0	name	eth1
pciAddress		pciAddress	1f1c:00:02:0

1: xNIC 2: Streams 3: Mesh 4: HA Paths 5: HA Summary 6: HA 7: Configuration 8: Timing Nodes 9: Adaptors 0: Hardware

The Hardware 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).

Troubleshooting swtch-top

1. If the swtch-top "Status" is showing that there is a "Connection error:"

1. Check that the cloudSwXtch is started.
2. Check that you entered in the proper cloudswxtch name or IP when running the swtch-top command.
3. If name does not work when running the swtch-top command then the DNS is not set-up correctly, use the IP address instead.

2. If an xNIC was installed but is not showing up in swxtch-top:

1. 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"

2. 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

3. If a multicast group is not showing up then check that they have registered.

- In Linux, run this command:

Text

None	Copy
<pre>ip maddress show</pre>	

- In Windows, run this command in PowerShell:

Text

None	Copy
<pre>netsh.exe interface ipv4 show joins</pre>	

- 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
<pre>\$ sudo tcpdump udp -X -i <interface></pre>	

NOTE

xNIC1 interface: `swtch-tun0`

xNIC2 interface: data nic (usually `eth1` for Linux, and "Ethernet 2" for Windows)

- Validate that a firewall is not stopping the multicast and open up the firewall to include port exceptions.

swtch-top on a cloudSwXtch

swtch-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 /swtch/swtch-top dashboard --swtch localhost`

Alternatively use `127.0.0.1` or `swtch-hostname` or `swtch-IP` in place of localhost

swxtch-tcpdump

WHAT TO EXPECT

Users can use a cloudSwXtch specific version of tcpdump called swxtch-tcpdump. 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
<pre>swxtch-tcpdump</pre>	

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
<pre>swxtch-tcpdump -i ens6</pre>	

Additional arguments

Users can use the **-h** argument as shown below to get a list of available arguments for swXtch-tcpdump.

Bash	Copy
<pre>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 [-AbdDefhHIJKlLnOpqStuUvxX#] [-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]</pre>	

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 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

Bash	Copy
PS C:\Program Files\SwXtch.io\Swxtch-xNIC2> .\swxtch-where.exe --format json	

Linux

Bash	Copy
<pre>swxtch-where -f json</pre>	

Example Output in Windows:

```
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": {
        "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
<pre>PS C:\Program Files\SwXtch.io\Swxtch-xNIC2> .\swxtch-where.exe -v 1.0</pre>	

swxtch-where Help

The `swXtch-where -h` or `--help` argument provides users with a detailed list of available arguments.

Example in Windows:

Bash	Copy
<pre>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")</pre>	

Troubleshooting

The swtch-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 swtch-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.

Information					
swtch001-sm	v1.3.6 (starter)		Max packet Rate	100 Kpps	
SubscriptionId	91b3	7545c1a	Max Bandwidth	1000 Mbps	
VMId	71ba	5f7d0cf			
SDMC Id	a5f5	777			
Status	OK				
Totals					
Producers	51.3K pps	(112.4M bps)	Consumers	100.1K pps	(219.4M bps)
Switch RX	50.8K pps	(111.2M bps)	Switch TX	202.6K pps	(444.2M bps)
Bridge RX	0 pps	(0 bps)			
Multicast Client Machines					
Name	Tx bps	Tx pps	Rx bps	Rx pps	
client001	14.1M	51.3K	0	0	
client002	0	0	13.7M	50.1K	
client003	0	0	13.7M	50.1K	

Cannot ping the cloudSwXtch instance

If `ping <swtch-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 swtch-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 swtch-xnic`

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 journalctl**. 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, License and Config -- all in a compressed 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
<pre>./swx support -f "yyyy-dd-mm" -t "yyy-dd-mm" -s localhost</pre>	

The output file will be named **swxtch-report-date_from.tar.gz**.

Available swx support Arguments

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 logs -s localhost:9095 -since "8h"	
Days \$./swx support logs -s localhost:9095 -since "2d"	

Core dumps (-dump)

Add **-dump** if you wish to include the core dumps in the report.

Export as text file (-plain)

Add **-plain** to the command if you wish to export the report as a plain text file instead of a json.

cloudSwXtch Service Logs

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
<pre>sudo journalctl -u <swxtch-ctrl.service OR swxtch-repl.service> -f</pre>	

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
<pre>sudo journalctl -u <swxtch-ctrl OR swxtch-repl> -n <number of lines></pre>	

Example:

Bash	Copy
<pre>sudo journalctl -u swxtch-repl -n 200</pre>	

Displaying Logs within a Timeframe (--since --until)

The following command will display logs within a set timeframe (between 2 dates).

Bash	Copy
<pre>sudo journalctl -u <swxtch-ctrl or swxtch-repl> --since <yyyy-mm-dd> --until <yyyy-mm-dd></pre>	

Example:

Bash	Copy
<pre>sudo journalctl -u swxtch-repl --since 2023-03-07 --until 2023-03-10</pre>	

--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
<pre>sudo journalctl -u swxtch-repl --since yesterday</pre>	

Displaying Logs since Last Boot (-b)

To display logs since last boot, users can use the -b argument.

Bash	Copy
<pre>sudo journalctl -u <swxtch-ctrl OR swxtch-repl> -b</pre>	

List boots (--list-boot)

The following argument can be used to list all the boots in date/time order.

Bash	Copy
<pre>sudo journalctl -u <swxtch-ctrl OR swxtch-repl> --list-boot</pre>	

Exporting Logs (>)

The following command will export your logs to a .txt file. Logs should be emailed to support@swxtch.io.

Bash	Copy
<pre>sudo journalctl - u <swxtch-ctrl or swxtch-repl> > <file-name>.txt</pre>	

Example:

Bash	Copy
<pre>sudo journalctl - u swxtch-repl > cloudswxtch-test.txt</pre>	

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	Copy
LAST BOOT: <pre>sudo journalctl - u swxtch-ctrl -b > cloudswxtch-test.txt</pre> TIMEFRAME: <pre>sudo journalctl - u swxtch-repl --since 2023-03-07 --until 2023-03-10 > cloudswxtch-test.txt</pre>	

Change Logs to UTC (--utc)

To switch logs from local time to UTC, use the following argument:

Bash	Copy
<pre>sudo journalctl --utc</pre>	

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
<pre>sudo journalctl -u swxtch-bridge2</pre>	

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
<pre>sudo journalctl -u <swxtch-bridge2 OR swxtch-bridge> -f</pre>	

Note: A user will need to choose between `swxtch-bridge2` or `swxtch-bridge`.

Example:

Bash	Copy
<pre>sudo journalctl -u swxtch-bridge2 -n 200</pre>	

Displaying Logs within a Timeframe (--since --until)

The following command will display logs within a set timeframe (between 2 dates).

Bash	Copy
<pre>sudo journalctl -u <swxtch-bridge2 or swxtch-bridge> --since <yyyy-mm-dd> --until <yyyy-mm-dd></pre>	

Example:

Bash	Copy
<pre>sudo journalctl -u swxtch-bridge2 --since 2023-03-07 --until 2023-03-10</pre>	

--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
<pre>sudo journalctl -u swxtch-bridge2 --since yesterday</pre>	

Displaying Logs since Last Boot (-b)

To display logs since last boot, users can use the -b argument.

Bash	Copy
<pre>sudo journalctl -u <swxtch-bridge2 OR swxtch-bridge> -b</pre>	

List boots (--list-boot)

The following argument can be used to list all the boots in date/time order.

Bash	Copy
<pre>sudo journalctl -u <swxtch-bridge2 or swxtch-bridge> --list-boot</pre>	

Exporting Logs (>)

The following command will export your logs to a .txt file. Logs should be emailed to support@swxtch.io.

Bash	Copy
<pre>sudo journalctl -u <swxtch-bridge2 or swxtch-bridge> > <file-name>.txt</pre>	

Example:

Bash	Copy
<pre>sudo journalctl -u swxtch-bridge2 > cloudswxtch-test.txt</pre>	

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	Copy
<pre>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</pre>	

Change Logs to UTC (--utc)

To switch logs from local time to UTC, use the following argument:

Bash	Copy
<pre>sudo journalctl --utc</pre>	

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
<pre>sudo journalctl -u swxtch-xnic-control -n 500 -f</pre>	

xnic-data

Bash	Copy
<pre>sudo journalctl -u swxtch-xnic-data -n 500 -f</pre>	

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.

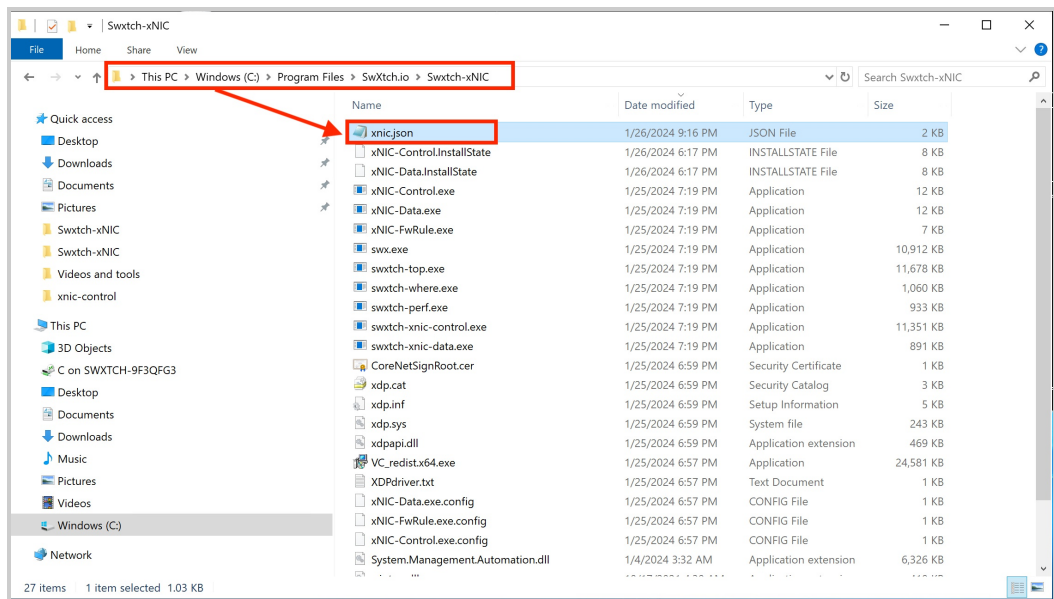
How to Change Verbosity

To change the verbosity, a user can manually edit the xNIC config file on their VM.

For Windows:

1. Go to the Swtch-xNIC folder on the VM you have an xNIC installed. Make sure it is the xNIC you want logs for.

1. C:\Program Files\SwXtch.io\Swtch-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.

4. For xNIC-Control, modify the StatsReportWait to change the time in seconds of the report stats.

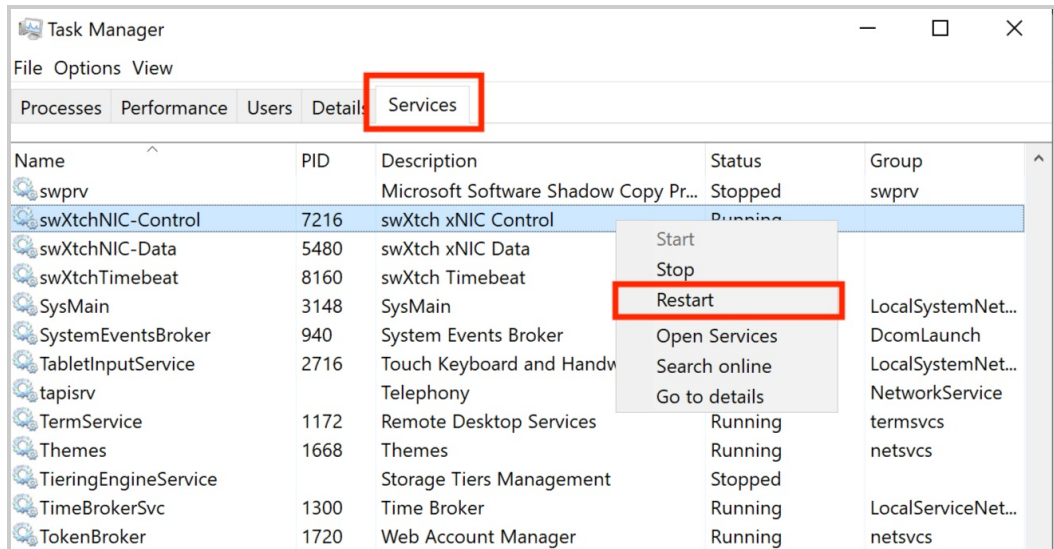


```
{
  "swtch": "10.2.128.10",
  "controlInterface": "Ethernet",
  "dataInterface": "Ethernet 2",
  "dataPort": 9999,
  "xnicType": 1,
  "dataPlaneSpecs": {
    "verbosity": 0,
    "virtualInterface": {
      "name": "swtch-tun0",
      "ip": "172.30.0.0",
      "subnet": "255.255.0.0",
      "mtu": 4096
    }
  },
  "ha": {
    "maxTimeToBufferPacketsMs": 50,
    "bufferSizeInPackets": 131072,
    "protocol": "rtp"
  },
  "streamSpecs": [
    {
      "mmcProducerEnable": true,
      "multipleMulticastGroups": [
        {
          "parent": "239.2.2.2:4000",
          "children": [
            "239.1.1.1:4000",
            "239.1.1.2:4000"
          ]
        }
      ]
    },
    {
      "parent": "239.3.3.3:4000",
      "children": [
        "239.1.1.3:4000",
        "239.1.1.4:4000"
      ]
    }
  ]
},
  "statsReportWait": 60
}
```

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.

8. Select "Restart."

1. **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	Copy
<pre>sudo nano /var/opt/swxtch/swxtch-xnic/swxtch-xnic.conf</pre>	

2. For xNIC-Data, change the number next to "verbose" so that it matches the level you desire. The default is 0.

3. For xNIC-Control, **modify** the StatsReportWait to change the time in seconds of the report stats.

```
GNU nano 4.8
{
  "swxtch": "10.2.128.10",
  "controlInterface": "eth0",
  "dataInterface": "eth1",
  "dataPort": 9999,
  "xnicType": 1,
  "dataPlaneSpecs": {
    "verbosity": 0,
    "virtualInterface": {
      "name": "swxtch-tun0",
      "ip": "172.30.0.0",
      "subnet": "255.255.0.0",
      "mtu": 4096
    }
  },
  "ha": {
    "maxTimeToBufferPacketsMs": 50,
    "bufferSizeInPackets": 131072,
    "protocol": "rtp"
  },
  "streamSpecs": {
    "mmcProducerEnable": true,
    "multipleMulticastGroups": [
      {
        "parent": "239.2.2.2:4000",
        "children": [
          "239.1.1.1:4000",
          "239.1.1.2:4000"
        ]
      },
      {
        "parent": "239.3.3.3:4000",
        "children": [
          "239.1.1.3:4000",
          "239.1.1.4:4000"
        ]
      }
    ]
  },
  "statsReportWait": 60
}
```

4. **Save** and **Exit** the file.
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
<pre>sudo systemctl restart swxtch-xnic-control</pre>	

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.

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	Copy
<pre>sudo /swxtch/swxtch-top dashboard --swxtch localhost</pre>	

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	Copy
<pre>curl -s http://127.0.0.1/top/dashboard grep -m 2 -Eo '"fingerprint"[^,]*' head -1</pre>	

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	Copy
<pre>sudo mv /home/<user>/license.json /swxtch</pre>	

7. Return to the **swxtch-top dashboard** again check the license took hold.

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
<pre>sudo dmidecode -s system-uuid</pre>	

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.
5. Run the following command.

Shell

Bash	Copy
<pre>sudo dpkg -i swxtch-license-svr_1.0.0_amd64.deb</pre>	

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:

Bash	Copy
<pre>sudo cp license.dat /swxtch/bundles</pre>	

7. Run the following command to start the Licensing Server. The Licensing Server will automatically start when you boot the VM in the future.

Bash	Copy
<pre>sudo systemctl start swxtch-license-svr</pre>	

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`
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

Bash	Copy
<pre>{ "url": "http://<IP-license-server>:8081/swxtch/bundles", "bundleName": "license" }</pre>	

1. **Note:** In version 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:

Bash	Copy
<pre>curl http://<IP-license-server>:8081/swxtch/bundles/show</pre>	

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:

Bash	Copy
<pre>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" }] }]</pre>	

poolSize lists the amount of cloudSwXtches allowed on this bundle. In this example, 5 cloudSwXtches are allowed with 1 already consuming a license (**consumedCount**).

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:

NoneCopy

```
ifconfig | grep mtu
```

Example:

NoneCopy

```
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:

NoneCopy

```
sudo ifconfig eth1 mtu 2000 up
```

Validate it is set to new value in this case 2000 by running this command:

NoneCopy

```
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 actually set the value:

None

[Copy](#)

```
set subinterface "Local Area Connection" mtu=2000 store=persistent
```

Where “Local Area Connection” is the Ethernet adaptor to be set - for example:

Administrator: Windows PowerShell

PS C:\> ipconfig

Windows IP Configuration

Unknown adapter swxthc-tun:

Media State : Media disconnected

Connection-specific DNS Suffix . :

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . : qw4ieorzzsjedntce414j

Link-local IPv6 Address : fe80::657a:1e18:2874:8

IPv4 Address. : 10.2.192.15

Subnet Mask : 255.255.252.0

Default Gateway :

Ethernet adapter Ethernet 2:

Connection-specific DNS Suffix . : qw4ieorzzsjedntce414jaktbh.bx.internal.cloudapp.net

Link-local IPv6 Address : fe80::853c:a2ac:cf78:842f%114

IPv4 Address. : 10.2.128.15

Subnet Mask : 255.255.252.0

Default Gateway : 10.2.128.1

PS C:\>

Administrator: Command Prompt - netsh

C:\Users\testadmin>netsh

netsh>interface

In future versions of Windows, Microsoft might remove the Netsh functionality for TCP/IP.

Microsoft recommends that you transition to Windows PowerShell if you currently use netsh to configure and manage TCP/IP.

Type Get-Command -Module NetTCPIP at the Windows PowerShell prompt to view a list of commands to manage TCP/IP.

Visit https://go.microsoft.com/fwlink/?LinkId=217627 for additional information about PowerShell commands for TCP/IP.

netsh interface>ipv4

netsh interface ipv4>set subinterface "Ethernet 2" mtu=1280 store=persistent

Ok.

netsh interface ipv4>

{height="" width=""}

2a. If you get the following error then follow steps after error:

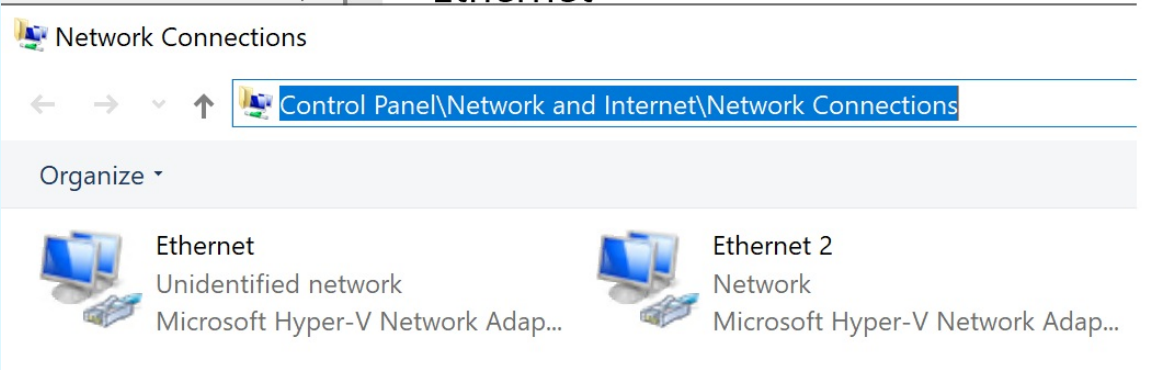
None

Copy

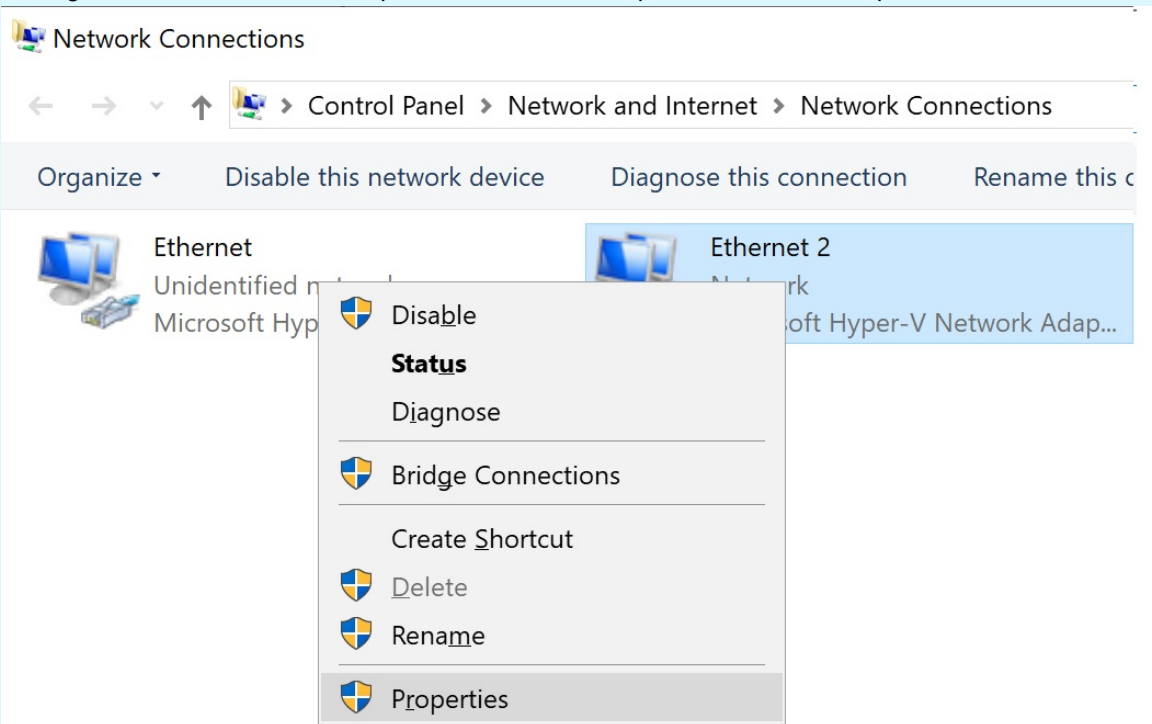
netsh interface ipv4>set subinterface "Ethernet 2" mtu=2000 store=persistent

The parameter is incorrect.

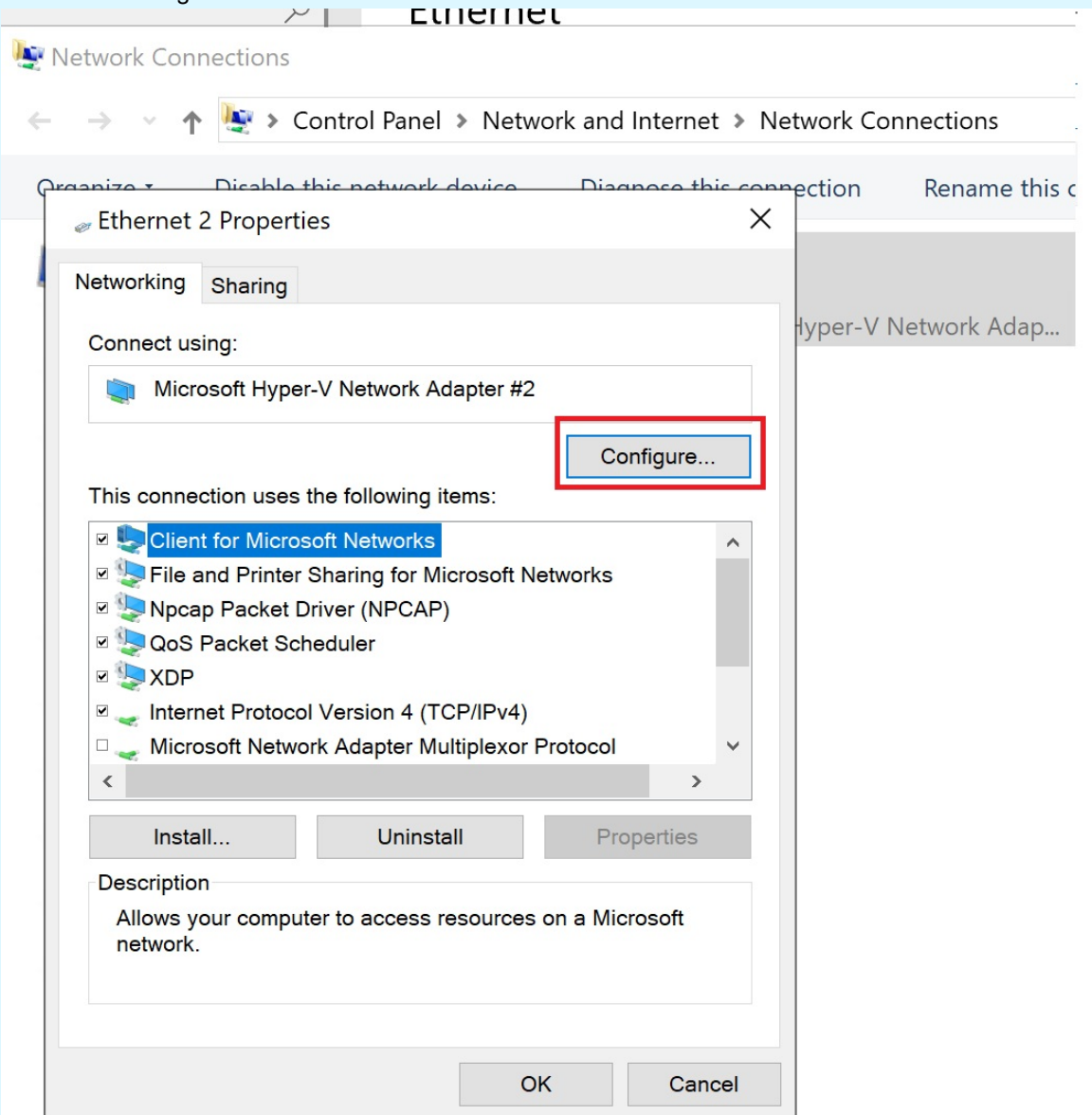
2b. Go to the Network connection → “Control Panel\Network and Internet\Network Connections”



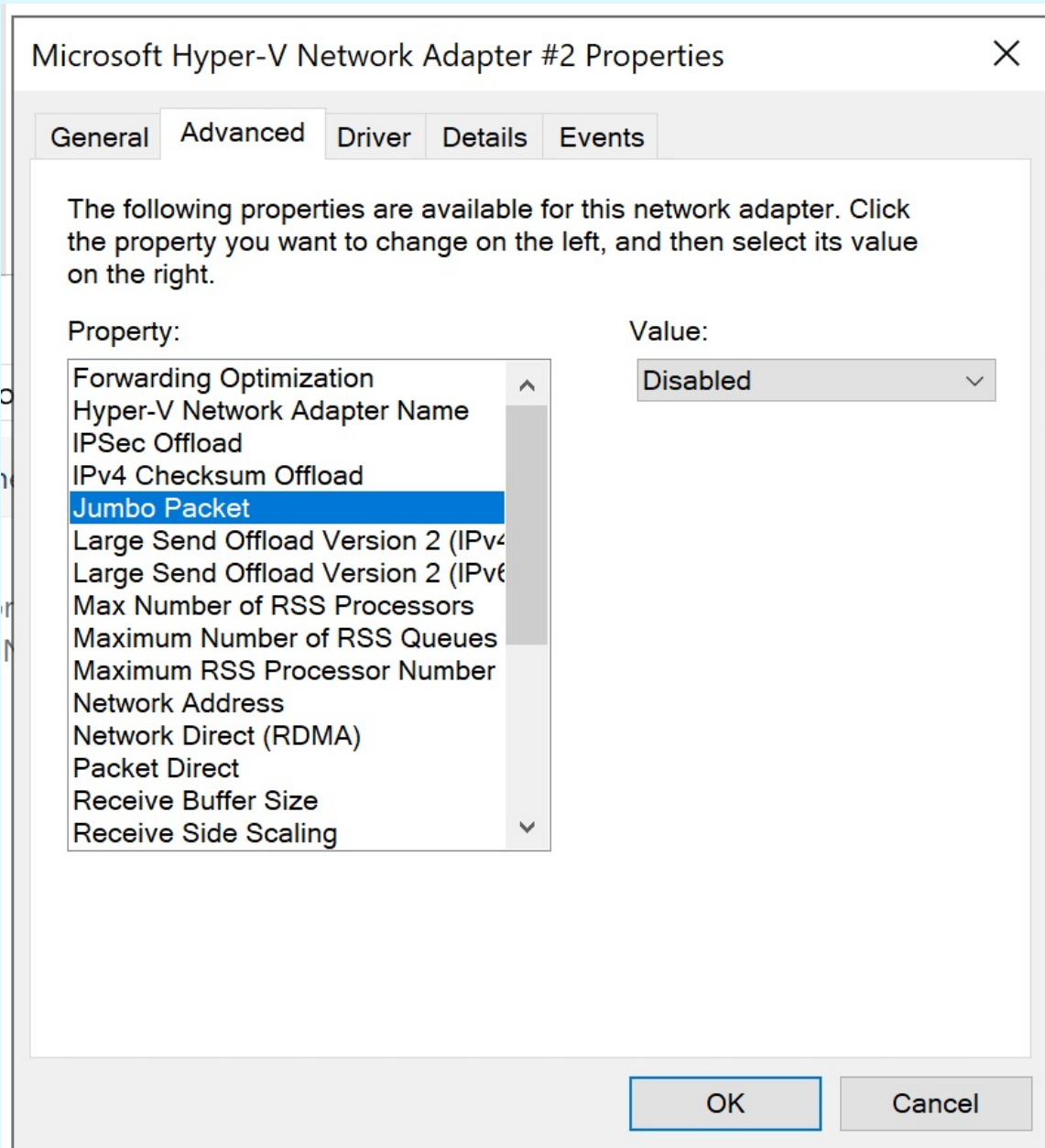
2c. Right click on the Ethernet Adaptor that the traffic is expected and select Properties.



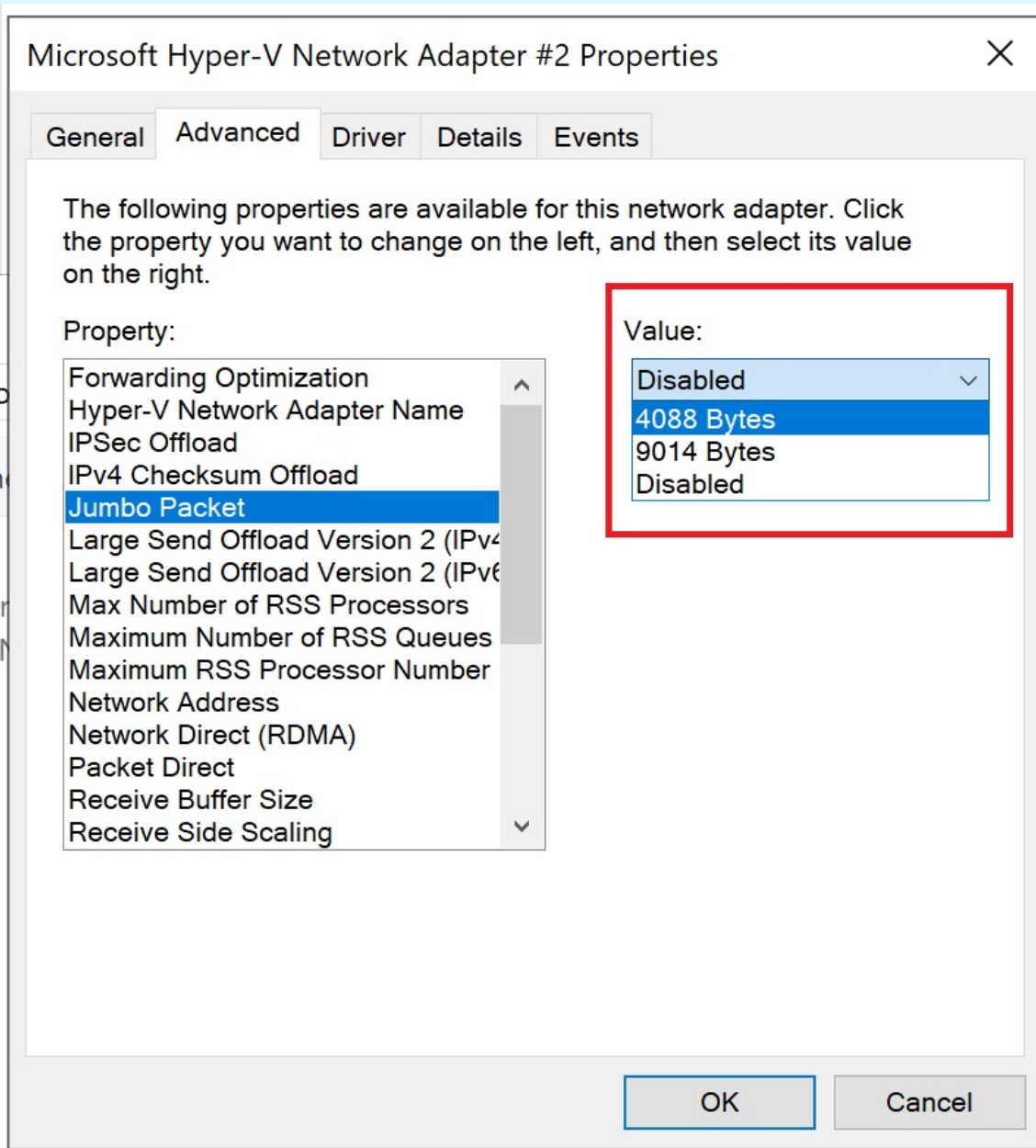
2d. Select "Configure"



2e. Select "Jumbo Packet"



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

How to Peer between VPCs in Different Regions for AWS

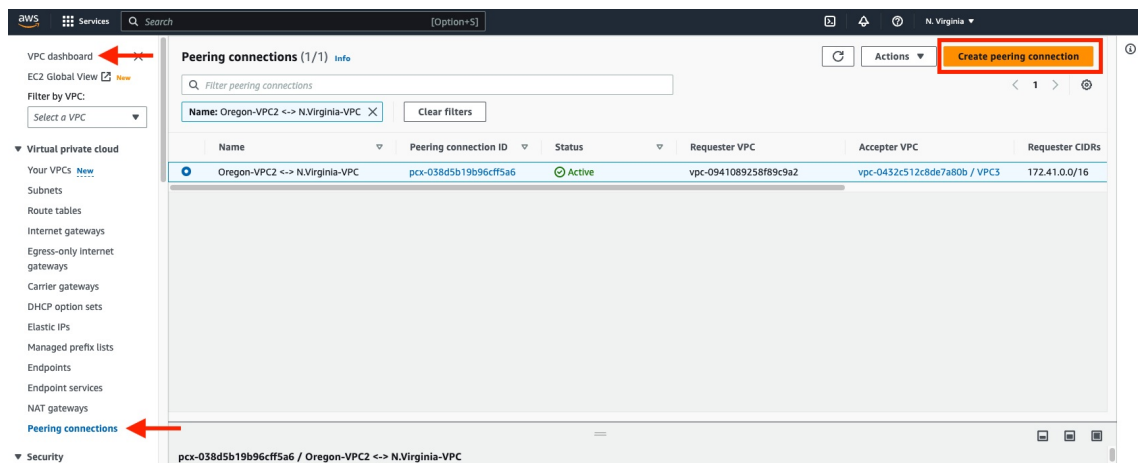
WHAT TO EXPECT

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.



3. Edit the following in the **Create peering connection** form:

1. Set a descriptive name. In the example, the user lists the connection between VPCs from Oregon and N. Virginia.
2. Select the VPC of the instance you want to connect from.
3. Select **Another Region** and select the destination region from the dropdown menu.
4. Enter the VPC ID of the target VPC in the target region.
5. Add any tag needed for organization purposes.

Create peering connection

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them privately. [Info](#)

Peering connection settings

Name - *optional*
Create a tag with a key of 'Name' and a value that you specify.

Oregon-VPC2 <-> N.Virginia-VPC

Select a local VPC to peer with
VPC ID (Requester)

vpc-0432c512c8de7a80b (VPC1)

VPC CIDRs for vpc-0432c512c8de7a80b (VPC1)

CIDR	Status	Status reason
172.31.0.0/16	✔ Associated	-

Select another VPC to peer with

Account

☒ My account
☐ Another account

Region

☐ This Region (us-east-1)
☒ Another Region

US West (Oregon) (us-west-2)

VPC ID (Accepter)

vpc-0941089258f89c9a2

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key

Q Name

✕

Value - *optional*

Q Oregon-VPC2 <-> N.Virginia-VPC

✕

Remove

Add new tag

You can add 49 more tags.

Cancel

Create peering connection

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.

VPC dashboard

EC2 Global View

Filter by VPC:

Select a VPC

Virtual private cloud

Your VPCs

Subnets

Route tables

Peering connections (1/2)

Filter peering connections

Name	Peering connection ID	Status	Requester VPC	Accepter VPC	Requester CIDRs	Accepter CIDRs	Requester owner ID	Accepter owner ID
eastus-to-westus2	pcx-0c4d2bfb57c419028	Active	vpc-019a57e1a1d2e78a0 / RegionalTestVPC	vpc-032478a302937f69e	172.31.0.0/16	172.31.0.0/16	639720666639	639720666639
Oregon-VPC2 <-> N.Virginia-VPC	pcx-0f612221d86e30677	Active	vpc-0941089258f89c9a2	vpc-0432c512c8de7a80b / VPC1	172.31.0.0/16	172.31.0.0/16	639720666639	639720666639

5. Change to the other region.
6. Go to **Peering Connections**.

7. Select the new Peering Connection listed as "Pending acceptance."

Peering connections (1/5) [Info](#)

Filter peering connections

	Name	Peering connection ID	Status	Requester VPC	Accepter VPC	Requester CIDRs	Accepter CIDRs	Requester owner ID	Accepter owner ID
☐	RegionalTestV...	pcx-0cfd2bfb57c419028	Active	vpc-019af57e1ad2e78a0	vpc-032478a302937fd9e / SA...	172.31.0.0/16	172.31.0.0/16	639720666639	639720666639
☒	-	pcx-012996f2c0fffd69c	Pending acceptance	vpc-0dd3720d5088eadf6	vpc-0941089258f89c9a2 / SA...	172.31.0.0/16	-	639720666639	639720666639
☐	Oregon-VPC2 ...	pcx-0fd12221d86e30677	Active	vpc-0941089258f89c9a2 / SA...	vpc-0432c512c8de7a80b	172.41.0.0/16	172.31.0.0/16	639720666639	639720666639
☐	MLP-VPC-Peer	pcx-06d670c3874f620f1	Active	vpc-093b945e59a8ef1a8 / SA...	vpc-0941089258f89c9a2 / SA...	172.52.0.0/16	172.41.0.0/16	639720666639	639720666639
☐	SA Test-1-and-...	pcx-098b8f5b53c70c920	Active	vpc-032478a302937fd9e / SA...	vpc-0941089258f89c9a2 / SA...	172.31.0.0/16	172.41.0.0/16	639720666639	639720666639

8. Under the Actions dropdown, select Accept request.

Peering connections (1/5) [Info](#)

Filter peering connections

	Name	Peering connection ID	Status	Requester VPC	Accepter VPC	Requester CIDRs	Accepter CIDRs	Requester owner ID	Accepter owner ID
☐	RegionalTestV...	pcx-0cfd2bfb57c419028	Active	vpc-019af57e1ad2e78a0	vpc-032478a302937fd9e / SA...	172.31.0.0/16	172.31.0.0/16	639720666639	639720666639
☒	-	pcx-012996f2c0fffd69c	Pending acceptance	vpc-0dd3720d5088eadf6	vpc-0941089258f89c9a2 / SA...	172.31.0.0/16	-	639720666639	639720666639
☐	Oregon-VPC2 ...	pcx-0fd12221d86e30677	Active	vpc-0941089258f89c9a2 / SA...	vpc-0432c512c8de7a80b	172.41.0.0/16	172.31.0.0/16	639720666639	639720666639
☐	MLP-VPC-Peer	pcx-06d670c3874f620f1	Active	vpc-093b945e59a8ef1a8 / SA...	vpc-0941089258f89c9a2 / SA...	172.52.0.0/16	172.41.0.0/16	639720666639	639720666639
☐	SA Test-1-and-...	pcx-098b8f5b53c70c920	Active	vpc-032478a302937fd9e / SA...	vpc-0941089258f89c9a2 / SA...	172.31.0.0/16	172.41.0.0/16	639720666639	639720666639

Actions

- View details
- Accept request
- Reject request
- Edit DNS settings
- Edit ClassicLink settings
- Manage tags
- Delete peering connection

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.

VPC dashboard

EC2 Global View [New](#)

Filter by VPC: [Select a VPC](#)

Virtual private cloud

Your VPCs [New](#)

Subnets

Route tables

Internet gateways

Egress-only internet

Route tables (5) [Info](#)

Find resources by attribute or tag

	Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC	Owner ID
☐	Private subnet routes for VPC1	rtb-0b9b0dd248a3db666	-	-	No	vpc-0432c512c8de7a80b VPC1	639720666639
☐	RegionalTestVPC-rt	rtb-06b64a11e29effed5	2 subnets	-	No	vpc-019af57e1ad2e78a0 RegionalTestVPC	639720666639
☐	Public subnet routes for VPC1	rtb-0f734496d1bed158c	subnet-01aaca91957af72d7 / VPC1-public	-	Yes	vpc-0432c512c8de7a80b VPC1	639720666639
☐	Oregon-VPC2 <-> N.Virginia-VPC	rtb-016f9783e7c943796	2 subnets	-	No	vpc-0432c512c8de7a80b VPC1	639720666639
☐	RegionalTestVPC-public-rtb	rtb-0d5c6db021244e9ec	subnet-058d27396c0dc1e08 / Public-subnet	-	Yes	vpc-019af57e1ad2e78a0 RegionalTestVPC	639720666639

Create route table

4. Edit the following in the **Create route table** form:

1. Enter a descriptive name.
2. Select the correct VPC.
3. Add any necessary tags.

VPC > Route tables > Create route table

Create route table [Info](#)

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Route table settings

Name - optional
Create a tag with a key of 'Name' and a value that you specify.

VPC
The VPC to use for this route table.

vpc-0432c512c8de7a80b (VPC1) ▼

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

[Add new tag](#)

You can add 50 more tags.

Cancel [Create route table](#)

5. Select **Create route table**.

6. Select the **Route table ID** of the route table you just created.

Route tables (1/5) Info		
Find resources by attribute or tag		
☐	Name ▼	Route table ID ▼
☐	Private subnet routes for VPC1	rtb-0b9bbdd248a3dbe66
☐	RegionalTestVPC-rt	rtb-06b64a11e29effed5
☐	Public subnet routes for VPC1	rtb-0f73d496d1bed158c
☒	Oregon-VPC2 <-> N.Virginia-VPC	rtb-016f9783e7c943796
☐	RegionalTestVPC-public-rtb	rtb-0d6c6db021244e9ec

7. Select **Edit routes** button the next screen.

The screenshot shows the AWS Management Console interface for a VPC. The left sidebar contains navigation links for VPC dashboard, EC2 Global View, and various VPC resources. The main content area displays the 'Route tables' page for a specific route table. The 'Routes' tab is selected, showing a list of routes. The 'Edit routes' button is highlighted with a red rectangle.

VPC > Route tables > rtb-08dd6801fcf231277

rtb-08dd6801fcf231277 / Oregon-VPC2<->N.Virginia-VPC3-data

You can now check network connectivity with Reachability Analyzer

Run Reachability Analyzer

Details

Route table ID: rtb-08dd6801fcf231277

Main: No

Explicit subnet associations: subnet-01df022a4f373fa17 / VPC1-data-subnet

Edge associations: -

VPC: vpc-0432c512c8de7a80b | VPC3

Owner ID: 639720666639

Routes (3)

Filter routes

Both

1

Edit routes

Destination	Target	Status	Propagated
172.31.0.0/16	local	Active	No

8. Add the **Destination** by entering the CIDR of the destination network.

9. Under **Target**, select the recently created **Peering Connection** from the list.

The screenshot shows the 'Edit routes' dialog box in the AWS Management Console. It displays a table with columns for Destination, Target, Status, and Propagated. Two routes are listed: one for 172.31.0.0/16 with target 'local', and another for 172.41.0.0/16 with target 'pcx-0fd12221d86e30677'. The 'Add route' button is visible at the bottom left, and the 'Save changes' button is at the bottom right.

VPC > Route tables > rtb-016f9783e7c943796 > Edit routes

Edit routes

Destination	Target	Status	Propagated
172.31.0.0/16	local	Active	No
172.41.0.0/16	pcx-0fd12221d86e30677	Active	No

Add route

Cancel Preview Save changes

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.

2. Select **Edit subnet associations** button under the **Explicit subnet associations** box.

VPC dashboard X

EC2 Global View [New](#)

Filter by VPC:

Select a VPC

Virtual private cloud

Your VPCs [New](#)

Subnets

Route tables

Internet gateways

Egress-only internet gateways

Carrier gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

Endpoint services

NAT gateways

Peering connections

Security

Network ACLs

Security groups

VPC > Route tables > rtb-08dd6801fcf231277

rtb-08dd6801fcf231277 / Oregon-VPC2<->N.Virginia-VPC3-data

Actions

You can now check network connectivity with Reachability Analyzer

Run Reachability Analyzer X

Details

Route table ID: rtb-08dd6801fcf231277

Main: No

Owner ID: 639720666639

VPC: vpc-0432c512c8de7a80b | VPC3

Explicit subnet associations: subnet-01df022a4f373fa17 / VPC1-data-subnet

Edge associations: -

Routes

Subnet associations

Edge associations

Route propagation

Tags

Explicit subnet associations (1)

Find subnet association

Edit subnet associations

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
VPC1-data-subnet	subnet-01df022a4f373fa17	172.31.96.0/20	-

Subnets without explicit associations (5)

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Find subnet association

3. Select the subnet(s) of the instance that must be connected to the destination.

VPC > Route tables > rtb-016f9783e7c943796 > Edit subnet associations

Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (2/8)

Filter subnet associations

	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
☐	Default VPC1-d	subnet-0d6aac918668cedbe	172.31.80.0/20	-	Main (rtb-0f73d496d1bed158c / Public subnet routes for VPC1)
☐	Default VPC1-f	subnet-0d9977ffb242c0086	172.31.64.0/20	-	Main (rtb-0f73d496d1bed158c / Public subnet routes for VPC1)
☐	VPC1-public	subnet-01aaca91957af72d7	172.31.112.0/24	-	rtb-0f73d496d1bed158c / Public subnet routes for VPC1
☒	VPC1-ctrl-subnet	subnet-03de822f9248b91ff	172.31.16.0/20	-	rtb-016f9783e7c943796 / Oregon-VPC2 <-> N.Virginia-VPC
☒	VPC1-data-subnet	subnet-01df022a4f373fa17	172.31.96.0/20	-	rtb-016f9783e7c943796 / Oregon-VPC2 <-> N.Virginia-VPC
☐	Default VPC1-e	subnet-068d4969f6ec1457a	172.31.48.0/20	-	Main (rtb-0f73d496d1bed158c / Public subnet routes for VPC1)
☐	Default VPC1-b	subnet-06e372352e6a55935	172.31.32.0/20	-	Main (rtb-0f73d496d1bed158c / Public subnet routes for VPC1)
☐	Default VPC1-c	subnet-0c633106e2e3bc850	172.31.0.0/20	-	Main (rtb-0f73d496d1bed158c / Public subnet routes for VPC1)

Selected subnets

subnet-01df022a4f373fa17 / VPC1-data-subnet X subnet-03de822f9248b91ff / VPC1-ctrl-subnet X

Cancel Save associations

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

Media Use Cases

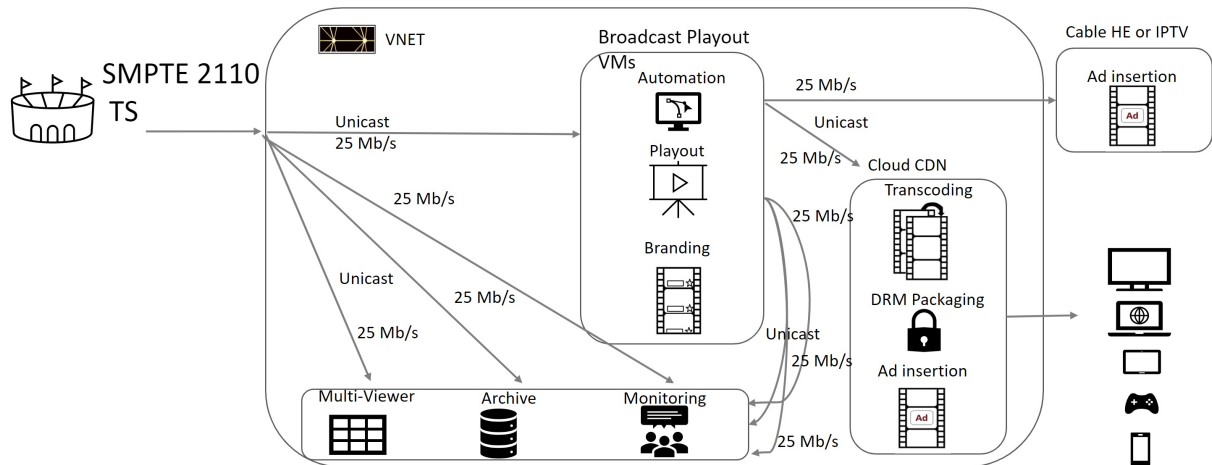
The media market can take advantage of several cloudSwXtch features such as:

- [Multicast](#)
- [Hitless Merge](#)
- [Compression support](#)
- [Protocol Fanout](#)
- [Disaster Recovery](#)

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 receiver. Network provisioning and administration is complex, distributed, difficult to modify and must be replicated for every workflow as shown below:

Unicast Playout in cloud without cloudSwXtch



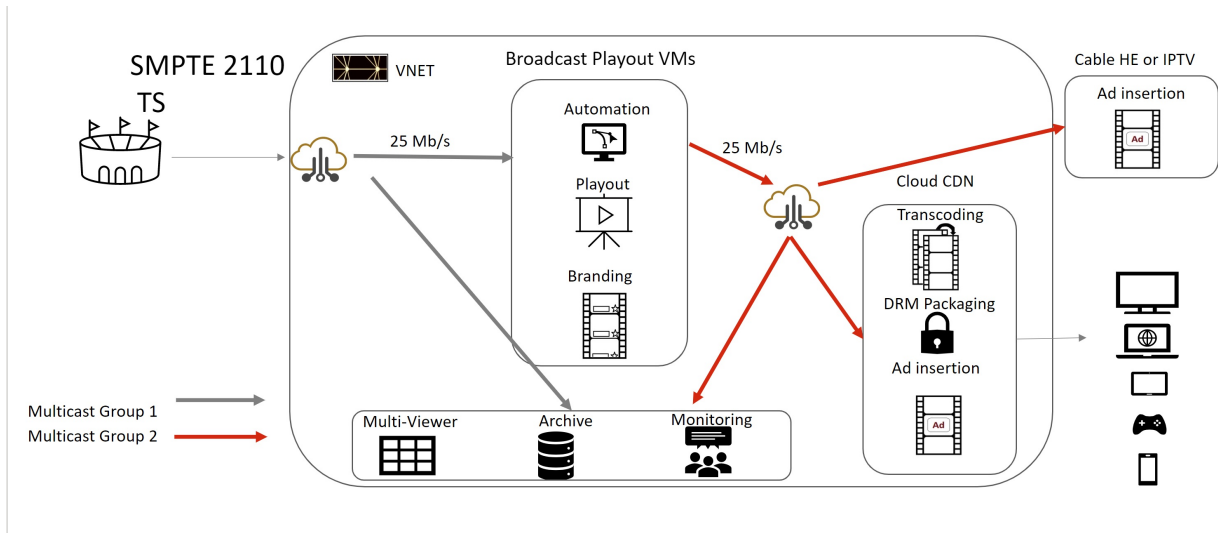
With unicast there are a number of issues:

- 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.



With cloudSwXtch Multicast:

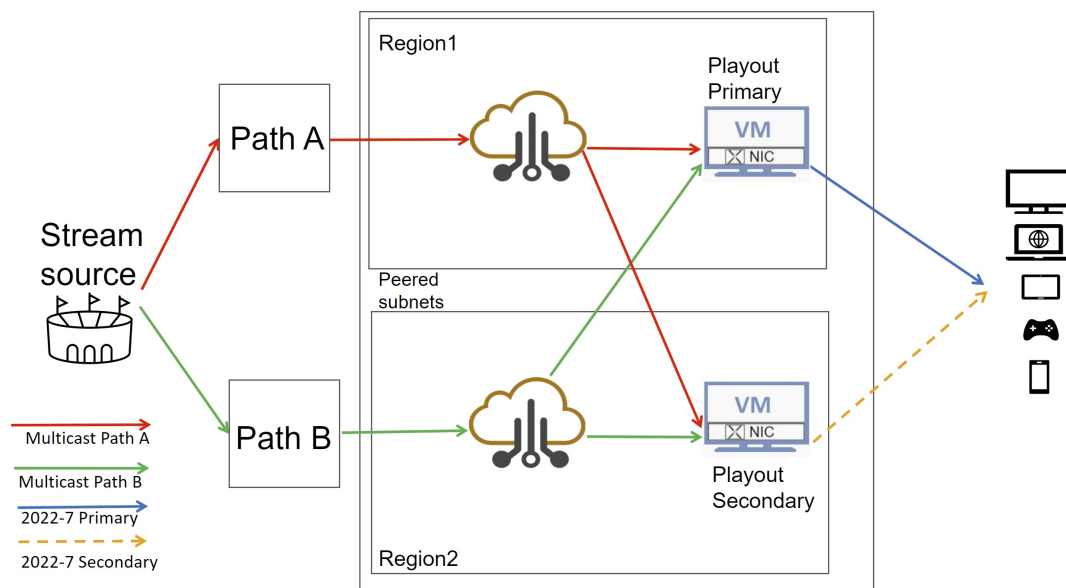
- Network may be modified and extended simply by joining multicast groups, with powerful centralized control and monitoring.
- Users can dynamically add new endpoints without playout server (or any other workflows/products) involvement.
- VM Sizes are minimized to workflow/product needs
- Disaster recovery is easy to set-up
- Minimal network load

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.



Media support for Compressed and Uncompressed Workflows

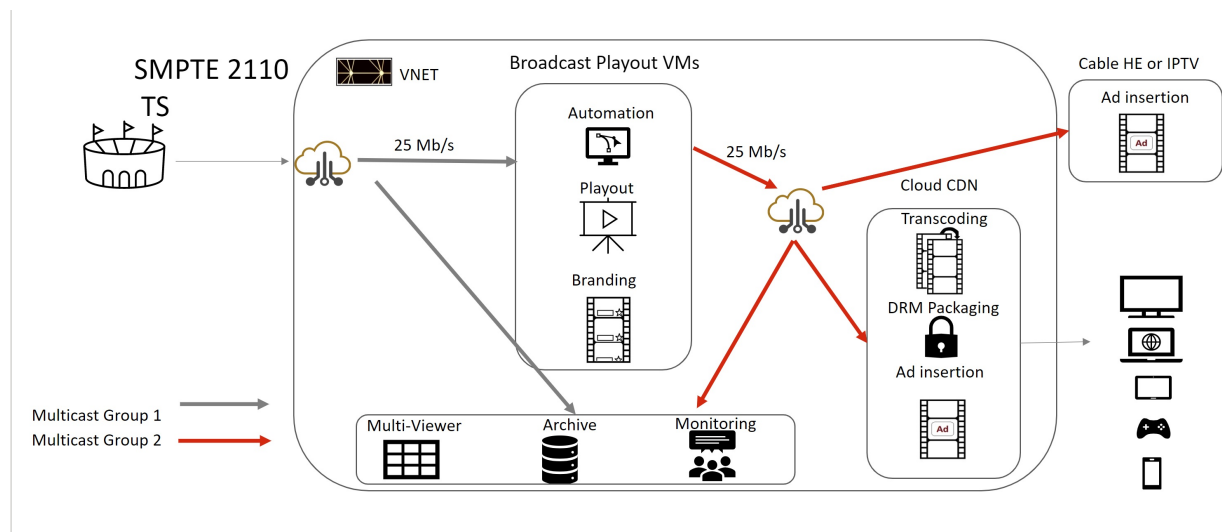


At **swXtch.io** we know that the media companies rely highly on both compressed and

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.

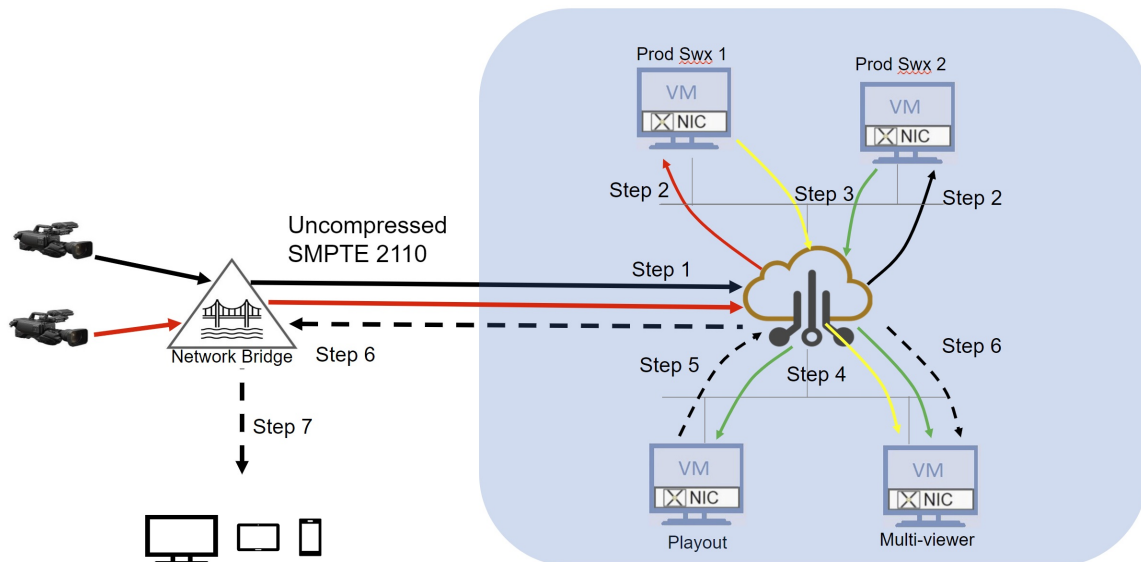
There are two workflow examples below, one is a compressed workflow and the other is an uncompressed workflow. The compressed workflow is a typical playout scenario where compressed inputs come into the cloud environment and are distributed via multicast to the necessary VM workloads by **cloudSwXtch**. 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

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 multicast 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 output 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.

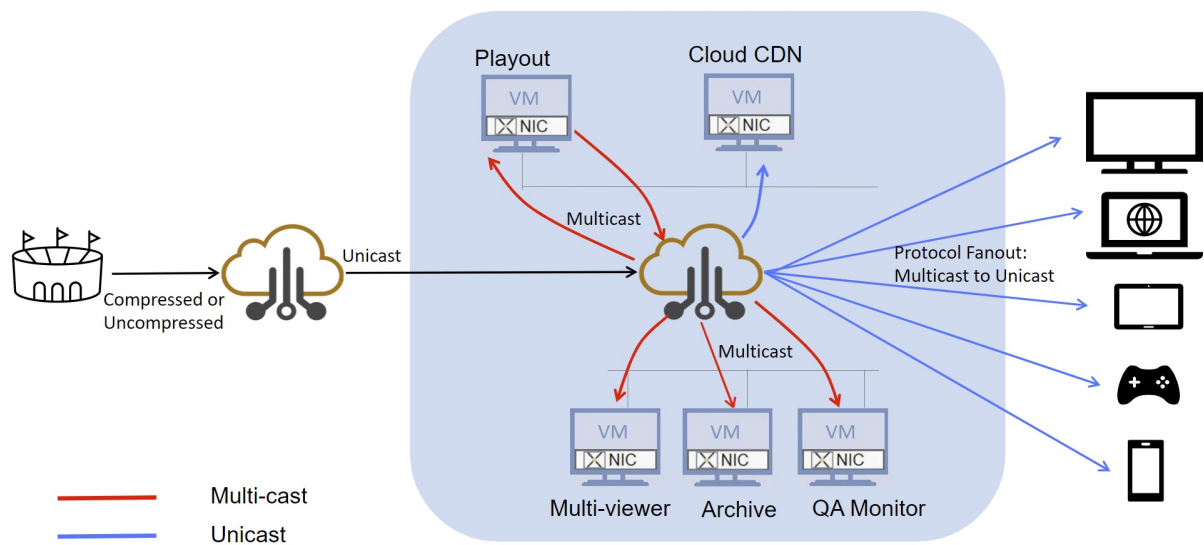
Step 7: The network bridge distributes to the clients for viewing consumption.

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.



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

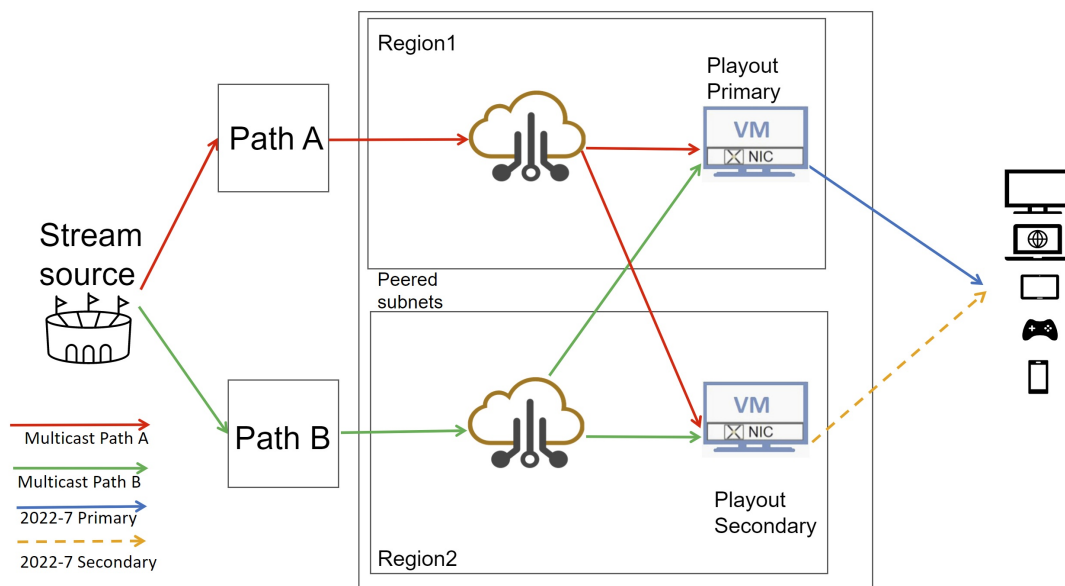
- The **cloudswXtch** in Region 1 can recieve the stream from Path A to Region 2.
- The **cloudswXtch** in Region 2 can recieve the stream from Path B to Region 1.
- If either path were to fail then the stream is still available in both Region 1 and 2 due to the redundancy.

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.



This depiction only shows two stream paths, there could be a third or more. In any of these scenarios the paths could be in different regions or different clouds. This is done by using a **cloudSwXtch** as a **Bridge** between clouds or from on-prem to cloud.

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.

27	28
29	30
31	32
33	34
35	36
37	38
39	40

$$DataIngress = [xnicTotals][ByteCounters](\Delta Nic2McaTotal * 8) / ((\Delta Timestamp / 1,000,000,000))$$

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.

$$\Delta Nic2McaTotal \text{ is } 51730345462 - 51706144186 = 7999676bits$$

$$[xnicTotals][ByteCounters](\Delta Mca2NicTotal * 8) / ((\Delta Timestamp / 1,000,000,000)) = (7999676 * 8) / 46.523565312 = 4161551.392 \approx 4.2Mbps$$

Cloud Information

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 —>

cloudSwXtch Information

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 —>



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
```

Example URL:

http://10.2.128.10/swxtch/monitoring/v1/stats/swxtch

Request: Empty

Response:

200 - successful response

▶ Example Response —>



xNIC Status Totals Information

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 —>



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:

Empty

Response:

200- successful response

▶ Example Response —>



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

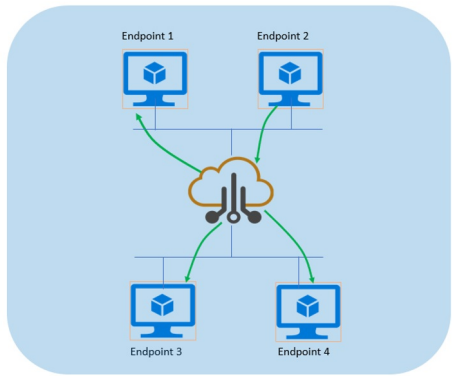
```
curl http://<cloudSwXtch-control-IP>/api/wxckedeye/v1/dashboard
```

This cloudSwXtch API documentation will examine each section of the response and provide users a better understanding of each field.

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.

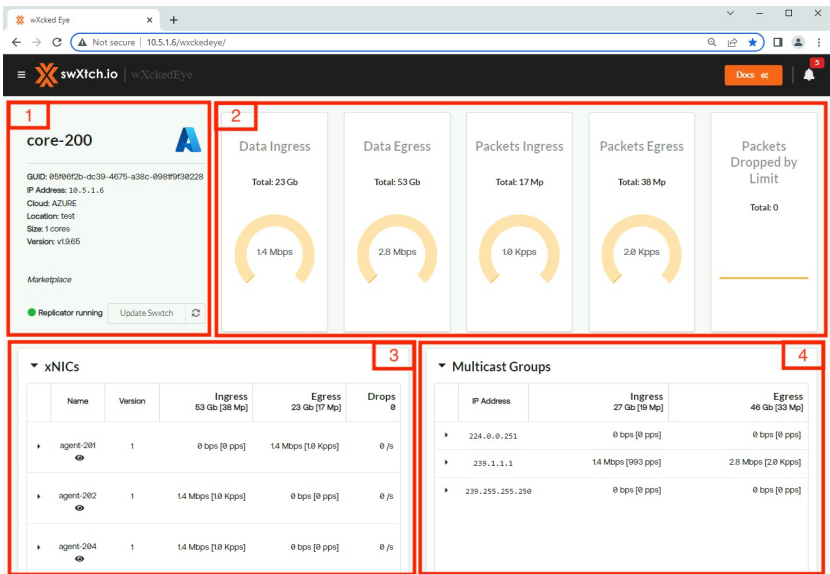
The example that will be used in this article refers to the network below, which has been simplified to help users understand the data returned in the API. As shown in Figure 1, Endpoint (Agent) 2 is sending data via multicast through the cloudSwXtch to Endpoints 1, 3 and 4.



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.

Bash

Copy

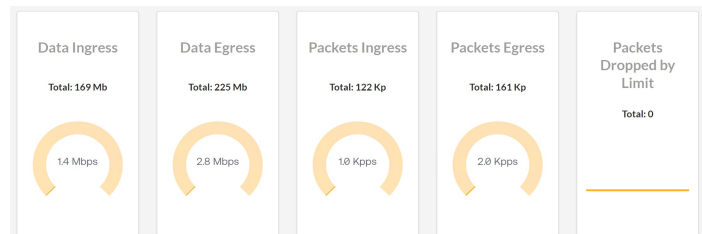
```

"remfVersion": "v1.7.4.draft",    *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": "c262fs1a-92c0-4346-as2f-547420127f313",
"hostName": " dsd-core-174",
"numCores": 4,
"replStatus": "running",
"authorized": true,
"validationResult": null,
"isMarketplace": true,

```

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.

Bash

Copy

```

"xnicTotals": {
  "PktCounters": {
    "Nic2McaTotal": 30218,
    "Nic2McaMc": 30218,
    "Mca2NicTotal": 31526,
    "Mca2NicMc": 31524,
    "Mca2NicIgmp": 6,
    "Mca2NicDrops": 0,
    "Nic2McaDrops": 0,
    "Mca2KniDrops": 0,
    "Kni2McaDrops": 0,
    "McaPktDrops": 0,
    "McaBigPktDrops": 0
  },
  "ByteCounters": {
    "Nic2McaTotal": 32347812,
    "Nic2McaMc": 32347812,
    "Mca2NicTotal": 33795772,
    "Mca2NicMc": 33795584
  },
  "Latencies": {
    "Count": 0,
    "Sum": 0,
    "Buckets": null
  },
  "HARxCounters": null,
  "Timestamp": 1657216501229157803,
  "SoftwareVersion": "",
  "XnicVersion": 0,
  "RxMulticastGroups": null,
  "TxMulticastGroups": null,
  "XnicMode": "",
  "NumConnections": 0
},

```

Statistics totals of agents

Packet counters

Packets from swxtch to agent

Multicast packets from swxtch to agent

Packets from agent to swxtch

Multicast packets from agent to swxtch

IGMP packets from agent to swxtch

Packets lost from agent to swxtch

Packets lost from swxtch to agent

Packets lost from agent to kernel NIC

Packets lost from kernel NIC to agent

Packets lost at agent

Big size packets lost at agents

Bytes From swxtch to agent

Multicast bytes from swxtch to agent

Bytes from agent to swxtch

Multicast bytes from agent to swxtch

SwxtchVersion

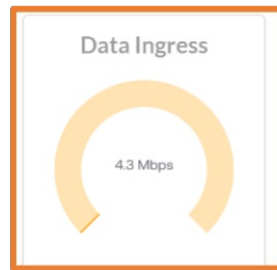
AgentVersion

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.

$\Delta \text{Something}$ means the difference between the value of at time and *Something* at time t_1 and *Something* t_2 . Example: If Egress has value 39000 at time t_1 and value 19800 at time t_2 ,

$$\Delta \text{Egress} = (39000 - 19800) = \text{result}$$

Calculating ByteCounters - Data Ingress



$$\text{DataIngress} = [\text{xnictotals}][\text{ByteCounters}](\Delta \text{Nic2McaTotal} * 8) / ((\Delta \text{Timestamp} / 1,000,000,000))$$

Below is a cut from the response of the API called at two different times.

27	"ByteCounters": {	27	"ByteCounters": {
28	"Nic2McaTotal": 51730345462,	28	"Nic2McaTotal": 51706144186,
29	"Nic2McaMc": 51720345462,	29	"Nic2McaMc": 51706144186,
30	"Mca2NicTotal": 51729942516,	30	"Mca2NicTotal": 51721942820,
31	"Mca2NicMc": 51729941904	31	"Mca2NicMc": 51721942288
32	},	32	},
33	"Latencies": {	33	"Latencies": {
34	"Count": 0,	34	"Count": 0,
35	"Sum": 0,	35	"Sum": 0,
36	"Buckets": null	36	"Buckets": null
37	},	37	},
38	"HARxCounters": null,	38	"HARxCounters": null,
39	"Timestamp": 1658155900751865500,	39	"Timestamp": 1658155854228300382,

Taking the above expression and putting in the data from this call is shown below.

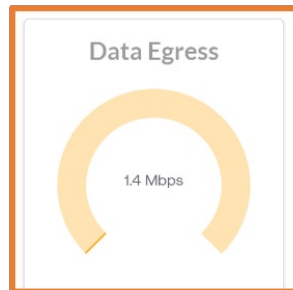
Timestamp from the call 2, 1658155900751865500 and the timestamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 in nanoseconds. Then, dividing the Δ by 1,000,000,000, we get the Δ of 46.523565312 in seconds, which we will use to calculate the data rate in bits per second.

$$\Delta \text{Nic2McaTotal} \text{ is } 51730345462 - 51706144186 = 7999676 \text{ bits}$$

$$[\text{xnictotals}][\text{ByteCounters}](\Delta \text{Mca2NicTotal} * 8) / ((\Delta \text{Timestamp} / 1,000,000,000)) = (7999676 * 8) / 46.523565312 = 4161551.392 \approx 4.2 \text{ Mbps}$$

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



$$\text{DataEgress} = [\text{xnictotals}][\text{ByteCounters}](\Delta \text{Mca2NicTotal} * 8) / ((\Delta \text{Timestamp} / 1,000,000,000))$$

Below is a cut from the response of the API called at two different times.

27	"ByteCounters": {	27	"ByteCounters": {
28	"Nic2McaTotal": 51730345462,	28	"Nic2McaTotal": 51706144186,
29	"Nic2McaMc": 51720345462,	29	"Nic2McaMc": 51706144186,
30	"Mca2NicTotal": 51729942516,	30	"Mca2NicTotal": 51721942820,
31	"Mca2NicMc": 51729941904	31	"Mca2NicMc": 51721942288
32	},	32	},
33	"Latencies": {	33	"Latencies": {
34	"Count": 0,	34	"Count": 0,
35	"Sum": 0,	35	"Sum": 0,
36	"Buckets": null	36	"Buckets": null
37	},	37	},
38	"HARxCounters": null,	38	"HARxCounters": null,
39	"Timestamp": 1658155900751865500,	39	"Timestamp": 1658155854228300382,

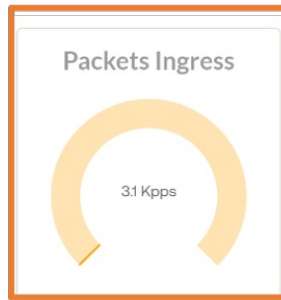
Taking the above expression and putting in data from this call is shown below.

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.

$$\Delta \text{Mca2NicTotal} \text{ is } 51729942516 - 51721942820 = 63997568 \text{ bits}$$

$$[\text{xnictotals}][\text{ByteCounters}](\Delta \text{Mca2NicTotal} * 8) / ((\Delta \text{Timestamp} / 1,000,000,000)) = (63997568 * 8) / 46.523565312 = 1375594.66 \approx 1.4 \text{ Mbps}$$

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.

14	"PktCounters": {	14	"PktCounters": {
15	"Nic2McaTotal": 297300912,	15	"Nic2McaTotal": 297161822,
16	"Nic2McaMc": 297300912,	16	"Nic2McaMc": 297161822,
17	"Mca2NicTotal": 297298589,	17	"Mca2NicTotal": 297252613,
18	"Mca2NicMc": 297298589,	18	"Mca2NicMc": 297252607,
19	"Mca2NicDrops": 0,	19	"Mca2NicDrops": 0,
20	"Mca2McaDrops": 0,	20	"Mca2McaDrops": 0,
21	"Mca2KniDrops": 0,	21	"Mca2KniDrops": 0,
22	"Mca2PktDrops": 0,	22	"Mca2PktDrops": 0,
23	"McaBigPktDrops": 0,	23	"McaBigPktDrops": 0,
24	"McaBigPktDrops": 0,	24	"McaBigPktDrops": 0,
25	"McaBigPktDrops": 0,	25	"McaBigPktDrops": 0,
26	},	26	},

Taking the above expression and putting in data from this call is shown below.

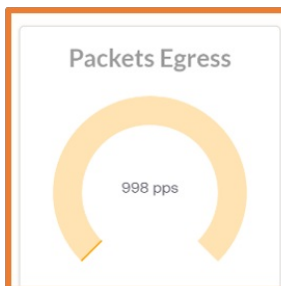
Timestamp from call 2, 1658155900751865500 and the timestamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 nanoseconds. Then, dividing the Δ by 1,000,000,000, we get a Δ of 46.523565312 in seconds.

$$\Delta Nic2McaTotal \text{ is } 297300912 - 297161822 = 139090 \text{Packets}$$

$$[xnicTotals][PktCounters](\Delta Nic2McaTotal)/((\Delta Timestamp/1,000,000,000)) = 139090/46.523565312 = 2989.667689 \approx 3.0Kpps$$

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



$$PacketsEgress = [xnicTotals][PktCounters](\Delta Mca2NicTotal)/((\Delta Timestamp/1,000,000,000))$$

Below is a cut from the return at two different times with data this section focuses on in orange.

27	"ByteCounters": {	27	"ByteCounters": {
28	"Nic2McaTotal": 51730345462,	28	"Nic2McaTotal": 51706144186,
29	"Nic2McaMc": 51730345462,	29	"Nic2McaMc": 51706144186,
30	"Mca2NicTotal": 51729942516,	30	"Mca2NicTotal": 51721942820,
31	"Mca2NicMc": 51729941984,	31	"Mca2NicMc": 51721942288,
32	"Latencies": {	32	"Latencies": {
33	"Count": 0,	33	"Count": 0,
34	"Sum": 0,	34	"Sum": 0,
35	"Buckets": null	35	"Buckets": null
36	"HarxCounters": null,	36	"HarxCounters": null,
37	"Timestamp": 1658155900751865500,	37	"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.

$$\Delta Mca2NicTotal \text{ is } 297298589 - 297252613 = 988.2303665 \text{Packets}$$

$$[xnicTotals][PktCounters](\Delta Mca2NicTotal)/((\Delta Timestamp/1,000,000,000)) = (988.2303665)/46.523565312 = 988.2303665 \approx 988pps$$

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

▼ xNICs			
Name	Ingress 77 Mb [55 Kp]	Egress 25 Mb [18 Kp]	Drops 0
DSd-agent-101	1.4 Mbps [999 pps]	0 bps [0 pps]	0 /s
DSd-agent-102	0 bps [0 pps]	1.4 Mbps [998 pps]	0 /s
DSd-agent-104	1.4 Mbps [1.0 Kpps]	0 bps [0 pps]	0 /s
DSd-agent-105	1.4 Mbps [1.0 Kpps]	0 bps [0 pps]	0 /s

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

Latency Buckets

The API breaks out latencies into buckets. Bucket 0 contains the count of latencies in 0.000050 (50 microseconds). Bucket 1 that contains latencies in 0.00100 (1 millisecond) and so on.

In the example, every bucket is 0. This means that there was no latencies in the period that this example was pulled from.

▶ Agent #2

▶ Agent #3

▶ Agent #4

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]

- **Total Data:** $[xnicTotals][ByteCounters]Nic2McaTotal * 8$
- **Total Packets:** $[xnicTotals][PktCounters]Nic2McaTotal$
- **Data Ingress:** $(([xnicTotals][< agentName >][ByteCounters](\Delta Nic2McaTotal * 8)))/((\Delta Timestamp/1,000,000,000))$
- **Pkts Ingress:** $(([xnicTotals][< agentName >][PktCounters](\Delta Nic2McaTotal)))/((\Delta Timestamp/1,000,000,000))$

Below is a cut from the response of the API called at two different times.

44	"Dsd-agent-101": {	44	"Dsd-agent-101": {
45	"PktCounters": {	45	"PktCounters": {
46	"Nic2McaTotal": 297092907,	46	"Nic2McaTotal": 297138907,
47	"Nic2McaMc": 297092907,	47	"Nic2McaMc": 297138907,
48	"Mca2NicTotal": 206,	48	"Mca2NicTotal": 206,
49	"Mca2NicMc": 206,	49	"Mca2NicMc": 206,
50	"Mca2NicIcmp": 20,	50	"Mca2NicIcmp": 20,
51	"Mca2NicDrops": 0,	51	"Mca2NicDrops": 0,
52	"Nic2McaDrops": 0,	52	"Nic2McaDrops": 0,
53	"Mca2KniDrops": 0,	53	"Mca2KniDrops": 0,
54	"Kni2McaDrops": 0,	54	"Kni2McaDrops": 0,
55	"McaPktDrops": 0,	55	"McaPktDrops": 0,
56	"McaBigPktDrops": 0	56	"McaBigPktDrops": 0
57	},	57	},
58	"ByteCounters": {	58	"ByteCounters": {
59	"Nic2McaTotal": 51694152976,	59	"Nic2McaTotal": 51702156848,
60	"Nic2McaMc": 51694152976,	60	"Nic2McaMc": 51702156848,
61	"Mca2NicTotal": 24514,	61	"Mca2NicTotal": 24514,
62	"Mca2NicMc": 24514	62	"Mca2NicMc": 24514
63	},	63	},
64	"Latencies": {	64	"Latencies": {
65	"Count": 0,	65	"Count": 0,
66	"Sum": 0,	66	"Sum": 0,
67	"Buckets": {	67	"Buckets": {
68	"0.000050": 0,	68	"0.000050": 0,
69	"0.000100": 0,	69	"0.000100": 0,
70	"0.000250": 0,	70	"0.000250": 0,
71	"0.000500": 0,	71	"0.000500": 0,
72	"0.001000": 0,	72	"0.001000": 0,
73	"0.002000": 0,	73	"0.002000": 0,
74	"0.005000": 0,	74	"0.005000": 0,
75	"0.010000": 0,	75	"0.010000": 0,
76	"0.011000": 0	76	"0.011000": 0
77	},	77	},
78	"McaRxCounters": {},	78	"McaRxCounters": {},
79	"Timestamp": 1658155900043488705,	79	"Timestamp": 165815588965889,
80	},	80	},

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, 165815585388965889 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: $\Delta Nic2McaTotal * 8 = 64030976bits$

$[xnlcs][Dsd-agent-101][ByteCounters](\Delta Nic2McaTotal * 8) / ((\Delta Timestamp / 1,000,000,000)) = (64030976) / 46.15452288 = 1387317.473 \approx 1.4Mbps$

Pkts Ingress: $\Delta Nic2McaTotal$ is $297138907 - 297092907 = 46000Packets$

$([xnlcs][Dsd-agent-101][PktCounters](\Delta Nic2McaTotal)) / ((\Delta Timestamp / 1,000,000,000)) = 46000 / 46.15452288 = 996.65 \approx 997pps$

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:

xNICs			
Name	Ingress 77 Mb [55 Kp]	Egress 25 Mb [18 Kp]	Drops 0
Dsd-agent-101	1.4 Mbps [999 pps]	0 bps [0 pps]	0 /s
Dsd-agent-102	0 bps [0 pps]	1.4 Mbps [998 pps]	0 /s
Dsd-agent-104	1.4 Mbps [1.0 Kpps]	0 bps [0 pps]	0 /s
Dsd-agent-105	1.4 Mbps [1.0 Kpps]	0 bps [0 pps]	0 /s

- **Total Data:** $[xnlcsTotals][ByteCounters]Mca2NicTotal * 8$
- **Total Packets:** $[xnlcsTotals][PktCounters]Mca2NicTotal$
- **Data Egress:** $([xnlcs][<agentName>][ByteCounters](\Delta Mca2NicTotal * 8)) / ((\Delta Timestamp / 1,000,000,000))$
- **Pkts Egress:** $([xnlcs][<agentName>][PktCounters](\Delta Mca2NicTotal)) / ((\Delta Timestamp / 1,000,000,000))$

83	},	83	},
84	"DSd-agent-102": {	84	"DSd-agent-102": {
85	"PktCounters": {	85	"PktCounters": {
86	"Nic2McaTotal": 0,	86	"Nic2McaTotal": 0,
87	"Nic2McaMc": 0,	87	"Nic2McaMc": 0,
88	"Mca2NicTotal": 297252401,	88	"Mca2NicTotal": 297298375,
89	"Mca2NicMc": 297252401,	89	"Mca2NicMc": 297298375,
90	"Mca2NicIcmp": 0,	90	"Mca2NicIcmp": 0,
91	"Mca2NicDrops": 0,	91	"Mca2NicDrops": 0,
92	"Nic2McaDrops": 0,	92	"Nic2McaDrops": 0,
93	"Mca2KniDrops": 0,	93	"Mca2KniDrops": 0,
94	"Kni2McaDrops": 0,	94	"Kni2McaDrops": 0,
95	"McaPktDrops": 0,	95	"McaPktDrops": 0,
96	"McaBigPktDrops": 0	96	"McaBigPktDrops": 0
97	},	97	},
98	"ByteCounters": {	98	"ByteCounters": {
99	"Nic2McaTotal": 0,	99	"Nic2McaTotal": 0,
100	"Nic2McaMc": 0,	100	"Nic2McaMc": 0,
101	"Mca2NicTotal": 5172191774,	101	"Mca2NicTotal": 51729917250,
102	"Mca2NicMc": 5172191774	102	"Mca2NicMc": 51729917250
103	},	103	},
104	"Latencies": {	104	"Latencies": {
105	"Count": 0,	105	"Count": 0,
106	"Sum": 0,	106	"Sum": 0,
107	"Buckets": {	107	"Buckets": {
108	"0.000050": 0,	108	"0.000050": 0,
109	"0.000100": 0,	109	"0.000100": 0,
110	"0.000250": 0,	110	"0.000250": 0,
111	"0.000500": 0,	111	"0.000500": 0,
112	"0.001000": 0,	112	"0.001000": 0,
113	"0.002000": 0,	113	"0.002000": 0,
114	"0.005000": 0,	114	"0.005000": 0,
115	"0.010000": 0,	115	"0.010000": 0,
116	"0.011000": 0	116	"0.011000": 0
117	}	117	}
118	},	118	},
119	"HARxCounters": [],	119	"HARxCounters": [],
120	"Timestamp": 165815585395371860,	120	"Timestamp": 1658155900086539499,
121	"SoftwareVersion": "v1.7.4.draft",	121	"SoftwareVersion": "v1.7.4.draft",
122	"XnicVersion": 2	122	"XnicVersion": 2

Taking the above expressions and putting in data from this call is shown below.

Timestamp from call 2, 1658155900086539499 and timestamp from call 1, 165815585395371860 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 $(51729917250 - 51721917774) * 8 = 63995808bits$

$[xnic][DSd - agent - 102][ByteCounters](\Delta Mca2NicTotal * 8)/((\Delta Timestamp/1,000,000,000))$
 $= (63995808)/(46.131167744) = 1387257.49054 \approx 1.4Mbps$

Pkts Egress: $\Delta Mca2NicTotal$ is $297298375 - 297252401 = 45974Packets$

$([xnic][DSd - agent - 102][PktCounters](\Delta Mca2NicTotal))/((\Delta Timestamp/1,000,000,000)) = 45974/(46.131167744) = 996.593 \approx 997pps$

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

▼ xNICs			
Name	Ingress 77 Mb [55 Kp]	Egress 25 Mb [18 Kp]	Drops 0
DSd-agent-101	1.4 Mbps [999 pps]	0 bps [0 pps]	0 /s
DSd-agent-102	0 bps [0 pps]	1.4 Mbps [998 pps]	0 /s
DSd-agent-104	1.4 Mbps [1.0 Kpps]	0 bps [0 pps]	0 /s
DSd-agent-105	1.4 Mbps [1.0 Kpps]	0 bps [0 pps]	0 /s

Total: $xnicTotals[PktCounters]Mca2NicDrops$

Drop(s): $([xnic][< agentName >][PktCounters](\Delta Mca2NicDrops))/((\Delta Timestamp/1,000,000,000))$

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

▼ 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]

1

2

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:

```

},
"replTotals": {
    "host": "",
    "sequence": 95924,
    "rxCount": 31535,
    "txCount": 30332,
    "rxBytes": 33796794,
    "txBytes": 32361180,
    "rxBridgeBytes": 0,
    "rxBridgeCount": 0,
    "timestamp": 1657216500274611599,
    "dropsByByteLimit": 0,
    "dropsByCountLimit": 0,
    "rxMeshPktCount": 0,
    "rxMeshBytes": 0,
    "txMeshPktCount": 0,
    "txMeshBytes": 0,
    "rxUnicastPktCount": 0,
    "rxUnicastBytes": 0,
    "txUnicastPktCount": 0,
    "txUnicastBytes": 0,
    "rxMulticastGroups": [
        {
            "groupId": "239.1.1.3",
            "pktsCount": 31460,
            "bytesCount": 33788040,
            "lastUpdate": "2022-07-07T15:29:00.789006233Z"
        },
        {
            "groupId": "239.0.0.251",
            "pktsCount": 75,
            "bytesCount": 8754,
            "lastUpdate": "2022-07-07T17:54:49.132519721Z"
        }
    ],
    "subscriptionId": "d723b93f-112c-49fb-9fda-03d3ada867f3",
    "hostName": "dsd-core-174",
    "numCores": 8,
    "replStatus": "running",
    "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,
    "meshes": {
        "uid": "DF91521E-7202-A3C2-8156-1FF5070545E9",
        "swxtches": [
            "10.2.128.10",
            "10.5.1.6"
        ]
    },
    "expiresAt": "2033-04-05T21:09:00Z"
}

```

Multicast Ingress Data - Section 1

▼ Multicast Groups

IP Address	Ingress 0 b [0 p]
224.0.0.251	0 bps [0 pps]
239.1.1.1	173 Kbps [996 pps]
239.255.255.250	0 bps [0 pps]

Data: $([x\text{nicos}][RXMulticastGroups]\Delta\text{bytesCount} * 8) / \Delta\text{lastUpdate}$

Packets: $[x\text{nicos}][RXMulticastGroups]\Delta\text{pktsCount} / \Delta\text{lastUpdate}$

Below is a cut from the return at two separate times with data for this section highlighted in orange.

"rxMulticastGroups": {	221	"rxMulticastGroups": {
{	222	{
"groupIp": "239.255.255.250",	223	"groupIp": "239.255.255.250",
"pktsCount": 956,	224	"pktsCount": 956,
"bytesCount": 238044,	225	"bytesCount": 238044,
"lastUpdate": "2022-07-19T21:29:58.813001632Z"	226	"lastUpdate": "2022-07-19T21:29:58.813001632Z"
},	227	},
{	228	{
"groupIp": "239.1.1.1",	229	"groupIp": "239.1.1.1",
"pktsCount": 540049064,	230	"pktsCount": 540092012,
"bytesCount": 93968537136,	231	"bytesCount": 93976010088,
"lastUpdate": "2022-07-25T19:48:40.189630746Z"	232	"lastUpdate": "2022-07-25T19:48:23.191477537Z"
},	233	},
},	234	},

$\Delta\text{lastUpdate} = 43.00003982\text{seconds}$

Data Ingress:

$\Delta\text{bytesCount} * 8 = (93976010088 - 93968537136) * 8 = 59783616\text{ bits}$

$([x\text{nicos}][RXMulticastGroups]\Delta\text{bytesCount} * 8) / \Delta\text{lastUpdate} = 59783616 / 43.00003982 = 1390315.364 \approx 1.4\text{Mbps}$.

Note that the value in the figure above is wrongfully shown as Bytes/second as opposed to bits/s.

Packets Ingress:

$\Delta\text{pktsCount} = 540092012 - 540049064 = 42948\text{packets}$

$[x\text{nicos}][RXMulticastGroups]\Delta\text{pktsCount} / \Delta\text{lastUpdate} = 42948 / 43.00003982 = 998.789772749 \approx 999\text{pps}$

Multicast Egress Data - Section 2

▼ 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: $([x\text{nicos}][TXMulticastGroups]\Delta\text{bytesCount} * 8) / \Delta\text{lastUpdate}$

Packets: $[x\text{nicos}][TXMulticastGroups]\Delta\text{pktsCount} / \Delta\text{lastUpdate}$

"txMulticastGroups": {	241	"txMulticastGroups": {
{	242	{
"groupIp": "239.1.1.1",	243	"groupIp": "239.1.1.1",
"pktsCount": 597605144,	244	"pktsCount": 597733970,
"bytesCount": 103983295056,	245	"bytesCount": 104005710780,
"lastUpdate": "2022-07-25T19:48:40.162378549Z"	246	"lastUpdate": "2022-07-25T19:48:23.162418365Z"
},	247	},
{	248	{
"groupIp": "224.0.0.251",	249	"groupIp": "224.0.0.251",
"pktsCount": 969,	250	"pktsCount": 975,
"bytesCount": 115872,	251	"bytesCount": 115872,
"lastUpdate": "2022-07-25T19:48:10.061601877Z"	252	"lastUpdate": "2022-07-25T19:48:41.162379503Z"
},	253	},
},	254	},
},	255	},
},	256	},

Below is a cut from the return at two separate times with data for this section highlighted in orange.

$\Delta\text{lastUpdate} = 43\text{seconds}$

Data Egress: $\Delta\text{bytesCount} * 8 = (104005710780 - 103983295056) * 8 = 179325792\text{bits}$

$([x\text{nicos}][TXMulticastGroups]\Delta\text{bytesCount} * 8) / \Delta\text{lastUpdate} = (179325792) / 43 = 4170367.256 \approx 4.2\text{Mbps}$.

Note: The value in the figure above is wrongfully shown as Bytes/second as opposed to bits/s.

Packets Ingress: $\Delta\text{pktsCount} = 597733970 - 597605144 = 128826\text{packets}$

$[xnics][TXMulticastGroups]\Delta pktsCount/\Delta lastUpdate = 128826/43 = 2995.954 \approx 3.0Kpps$

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:

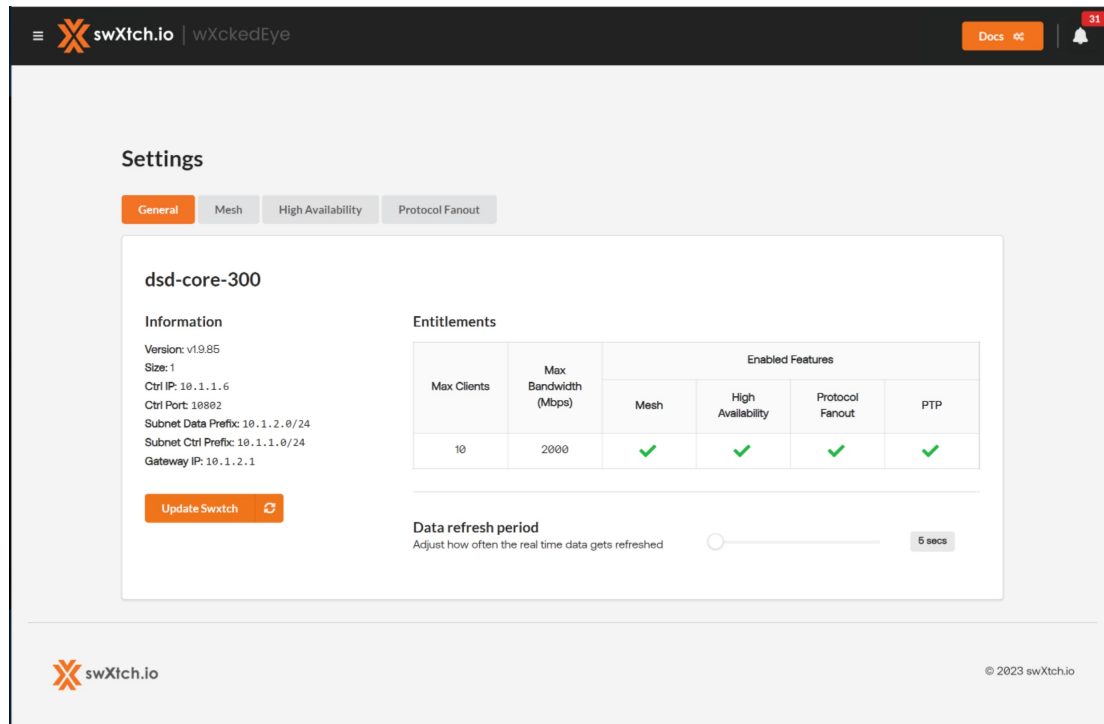


Mesh and HA Compatibility

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:

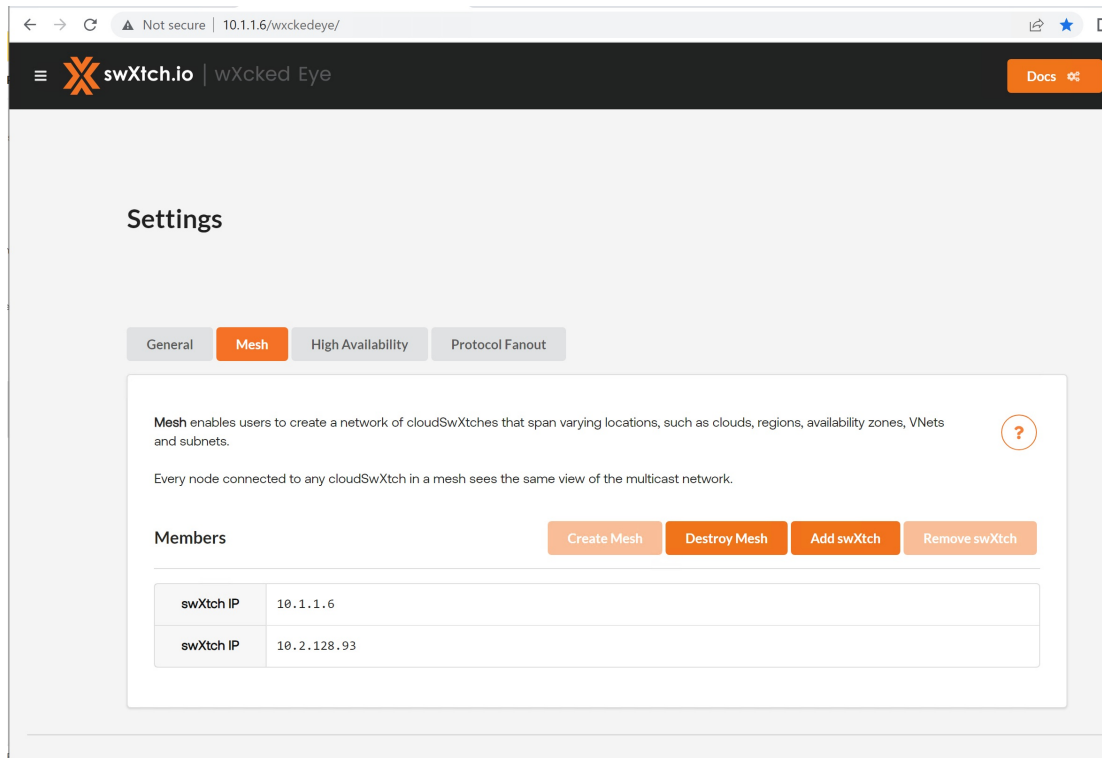
```

{
  "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.



Below is a portion of the Settings call response detailing information on the Mesh tab:

Bash	Copy
<pre>"meshSettings": { "meshId": "86B62026-9222-0E62-03C2-6C5872CA64C5", "swxtches": ["10.1.1.6", "10.2.128.93"], "uid": null },</pre>	

Create Mesh

►
swxtch/mesh/v1/create

]

Show Mesh

►
swxtch/mesh/v1/show

]

Add cloudSwXtch to Mesh

►
swxtch/mesh/v1/addSwxtch



Remove cloudSwXtch from Mesh

►
swxtch/mesh/v1/removeSwxtch



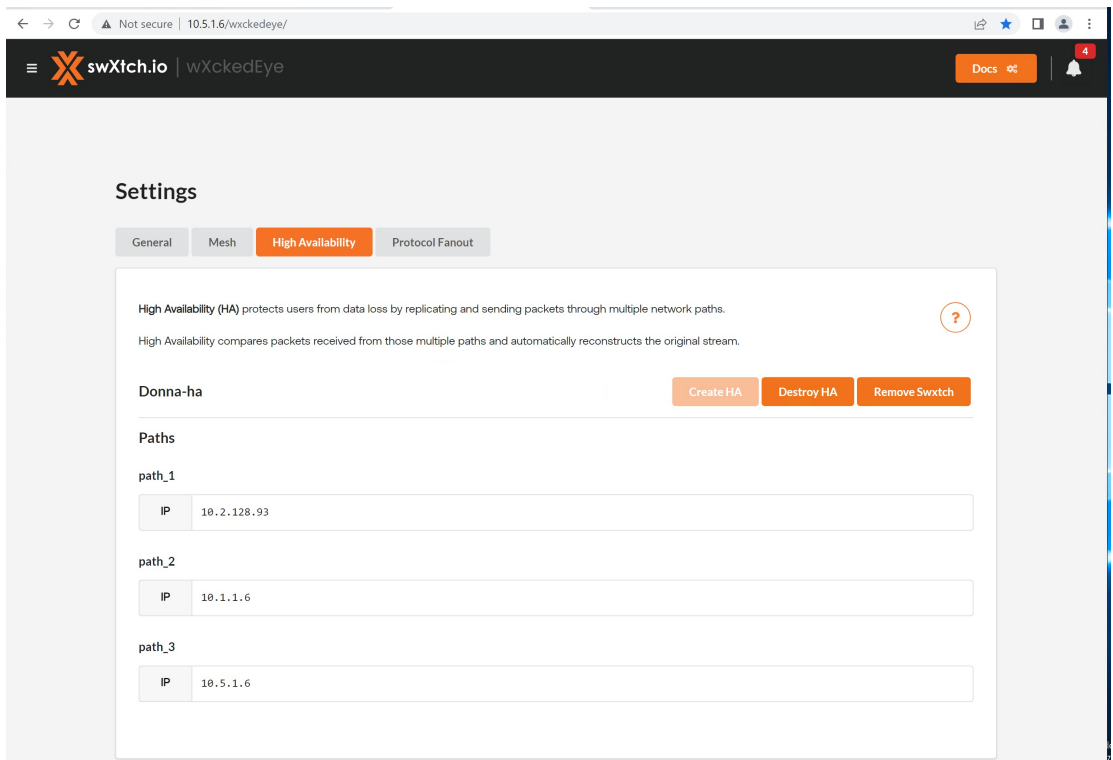
Destroy Mesh

►
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.



Below is a portion of the Settings call response detailing information on High Availability:

Bash	Copy
<pre>"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"] }] }</pre>	

Create HA

►
swxtch/ha/v1/create

]

Get HA List

►
swxtch/ha/v1/show

]

Delete cloudSwXtch from HA

►
swxtch/ha/v1/removeSwxtch

]

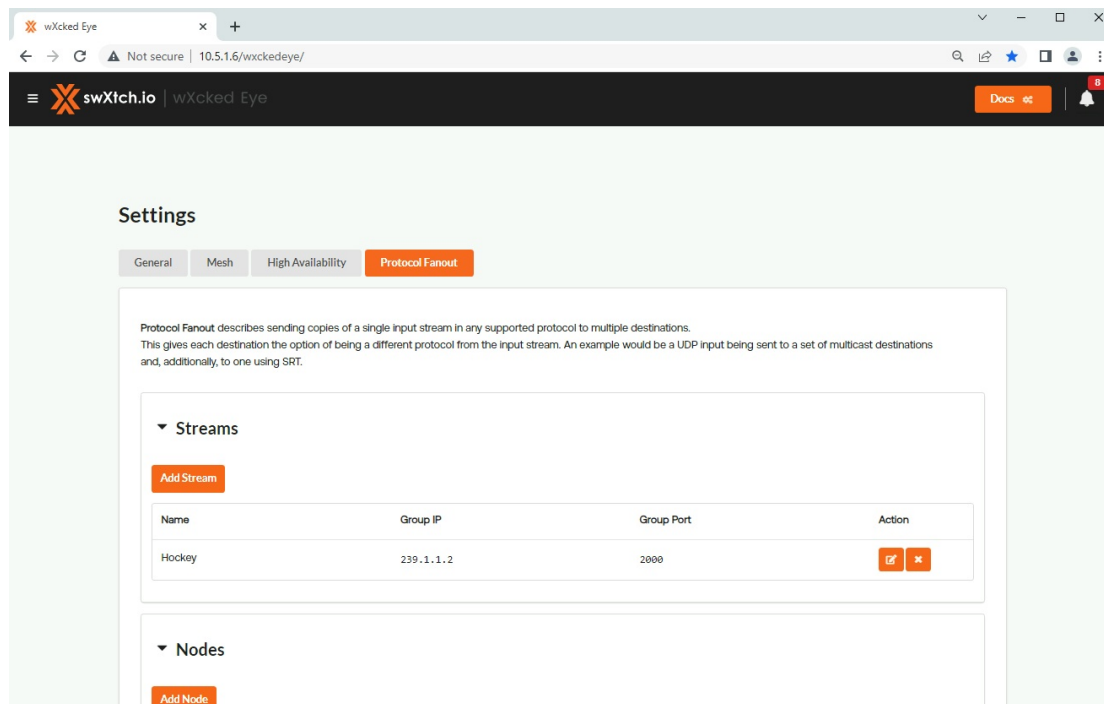
Destroy HA

►
swxtch/ha/v1/destroy

]

Protocol Fanout

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.

```
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



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



Remove Node

▶
swxtch/entities/nodes/v1/remove



UDP (Unicast Fanout)

UDP Enable

▶
swxtch/protocols/udp/v1/ingress/enable



UDP Egress Join

▶
swxtch/protocols/udp/v1/egress/join



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



SRT Caller

SRT Caller Ingress Enable

►
`swxtch/protocols/srt-caller/v1/ingress/enable`



SRT Caller Ingress Show

►
`swxtch/protocols/srt-caller/v1/ingress/show`



SRT Caller Egress Join

►
`swxtch/protocols/srt-caller/v1/egress/join`



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

SRT Listener Ingress Enable

►
`swxtch/protocols/srt-listener/v1/ingress/enable`



SRT Listener Ingress Show

►
swxtch/protocols/srt-listener/v1/ingress/show



SRT Listener Egress Join

►
swxtch/protocols/srt-listener/v1/egress/join



SRT Listener Egress Show

►
swxtch/protocols/srt-listener/v1/egress/show



SRT Listener Egress Leave

►
swxtch/protocols/srt-listener/v1/egress/leave



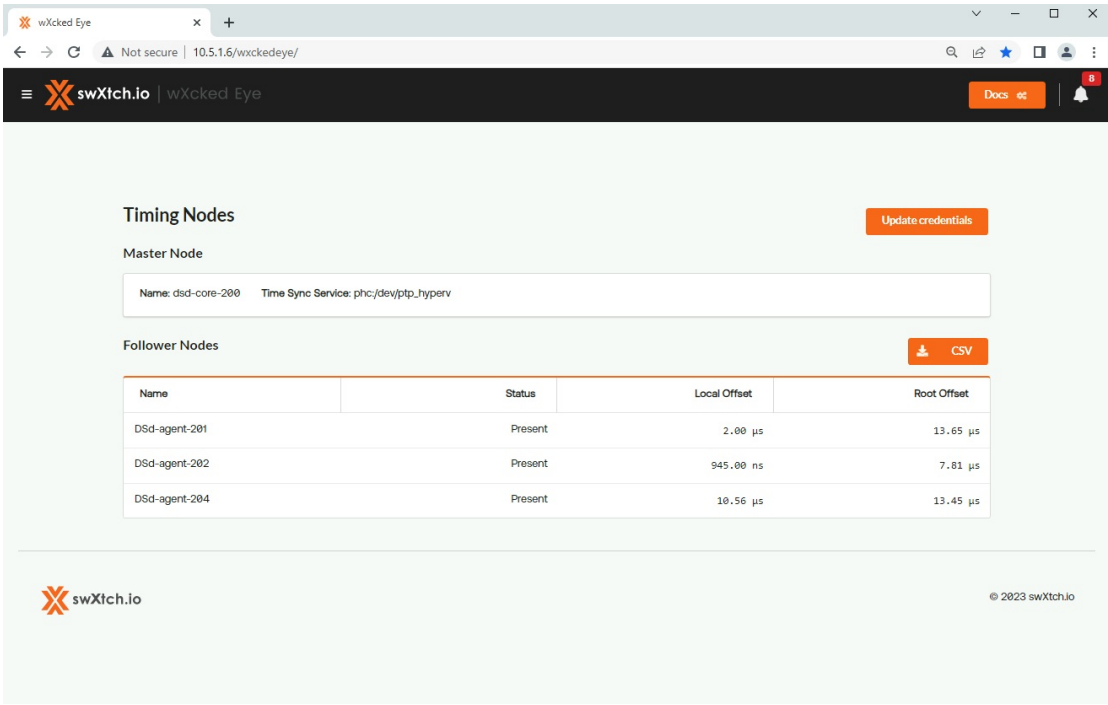
SRT Listener Ingress Disable

►
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.



PTP Master

►
swxtch/ptp/v1/master



PTP Slaves

►
swxtch/ptp/v1/slaves



Update PTP Credentials

►
swxtch/ptp/v1/updateTimebeatCredentials



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](#)

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.

▶
2.1.2 Release Notes - February 15, 2024



▶
2.1.1 Release Notes - February 5, 2024



▶
2.1.0 Release Notes - January 2024



▶
2.0.34 Release Notes - October 2023



▶
2.0.10 Release Notes - July 2023



▶
1.9.85 Release Notes



▶
1.9.73 Release Notes



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

FAQ

Please see the [swtch.io website FAQ page](#) for the most up-to-date version of the FAQ.

Q: *What is a cloudSwXtch?*

A: cloudSwXtch is a virtual overlay network that runs in your Azure tenant. It creates a standards-compliant network by deploying virtual network switches and virtual Network Interface Controllers (xNICs) that allow software workloads running on virtual machines to distribute their information as if they were running on a physical network. Many features not available on cloud networks, like multicast, PTP, packet pacing, custom packet filtering, and others may be implemented as features on this virtual network.

Q: *What is required to run cloudSwXtch?*

A: cloudSwXtch is available for workloads running on virtual machines that run RHEL 7 or later, CentOS 7 or later, and Ubuntu 18.04 or later. These operating systems must run on an x86_64 CPU. Each client VM must have two Network Interface Cards.

Q: *What happens when I run cloudSwXtch?*

A: cloudSwXtch creates a virtual switch architecture that behaves like a physical network switch. The switch runs on its own virtual machine(s) that is scaled for the network load that you require. Each virtual machine in your tenant that needs to access the multicast network must run a very small network interface application that communicates with the switch. Any workload that sends or receives multicast packets can join or leave a multicast group using standard IGMP calls.

Q: *What operating systems does xNIC support?*

A: It depends on the xNIC version.

Version 1: RHEL 7+, CentOS 7+, or Ubuntu 18.04 | 20.04, Windows 10, Windows Server 2016+

Version 2: RHEL 8, CentOS 8, or Ubuntu 20.04, Windows 10, Windows Server 2016+

Q: *Which version of IGMP does cloudSwXtch support?*

A: cloudSwXtch is fully compliant with IGMP Version 2, and partially compliant with IGMP Version 3. cloudSwXtch currently supports many of the features of IGMP Version 3 that are in common use, and will fully support IGMP version 3 in a future release.

Q: *Can I send multicast traffic across Azure vNets?*

A: cloudSwXtch is currently able to transfer packets between vNets or VPCs.

Q: *What resources are used by a cloudSwXtch?*

A: A cloudSwXtch uses only the VM resources in which it runs. The size of the VM determines the level of performance of the switch. The minimum VM size (core count) supported is 4 cores.

NOTE

You can select custom, to select a specific VM type and size.

NOTE:

Please be aware that the owner of the subscription in which the cloudSwXtch instance is created is responsible for all cloud resources used by the cloudSwXtch. These fees are to the cloud provider and do not include any fees to swtch.io for licensing.