

CLOUDSWXTCH

Version 2.0.34



Table of Contents

cloudSwXtch

Quotas	5
FAQ	6

cloudSwXtch > Getting Started

Getting Started	7
What is cloudSwXtch?	9

cloudSwXtch > Getting Started > cloudSwXtch Features

cloudSwXtch Features	11
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	24
---	----

cloudSwXtch > Installing cloudSwXtch > cloudSwXtch on AWS

cloudSwXtch on AWS	26
Validate Subnets on AWS	27
Verify Security Groups	31
Create SSH Key Pair	33
Upgrade cloudSwXtch on AWS	35
Deploy cloudSwXtch with Terraform on AWS	36
Check Health of cloudSwXtch Instance on AWS	40
Delete cloudSwXtch on AWS	42

cloudSwXtch > Installing cloudSwXtch > cloudSwXtch on Azure

cloudSwXtch on Azure	43
Validate Subnets on Azure	45
Create an Azure cloudSwXtch Template	47
Install cloudSwXtch on Azure	50
Upgrade cloudSwXtch on Azure	55
Deploy cloudSwXtch with Terraform on Azure	56
Install cloudSwXtch for an Air-Gapped Environment	60

cloudSwXtch > Installing cloudSwXtch > cloudSwXtch on GCP

cloudSwXtch on GCP	70
------------------------------------	----

cloudSwXtch > Installing cloudSwXtch > cloudSwXtch on OCI

cloudSwXtch on OCI	71
Install cloudSwXtch via OCI Marketplace	72

cloudSwXtch > Installing cloudSwXtch Bridge

Installing cloudSwXtch Bridge	83
---	----

Install cloudSwXtch Bridge Type 2	84
Install cloudSwXtch Bridge Type 1	87

cloudSwXtch > Installing xNIC

Installing xNIC	88
xNIC System Requirements	89

cloudSwXtch > Installing xNIC > Install xNIC on Linux

Install xNIC on Linux	90
xNIC Linux Uninstall	93
xNIC Linux Upgrade	94

cloudSwXtch > Installing xNIC > Install xNIC on Windows

Install xNIC on Windows	96
Uninstall xNIC on Windows	100

cloudSwXtch > Installing xNIC > Install xNIC on AKS Cillium

Install xNIC on AKS Cilium	101
Create an Azure Kubernetes Service with Cilium	103
Install xNIC2 on AKS Cilium Native	109
Uninstall xNIC2 on AKS	125
Test xNIC2 with AKS	126
Upgrade xNIC nodes on AKS	132

cloudSwXtch > Using wXcked Eye for cloudSwXtch

Using wXcked Eye for cloudSwXtch	134
--	-----

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

Monitor cloudSwXtch with wXcked Eye	136
wXcked Eye Network Graph	138
wXcked Eye cloudSwXtch Page	140
wXcked Eye xNICs Page	144

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

Configure cloudSwXtch with wXcked Eye	148
General	150
Mesh with wXcked Eye	151
High Availability with wXcked Eye	156
Protocol Conversion and Fanout with wXcked Eye	161
Aliases	165
Timing Nodes	168

cloudSwXtch > Configuring cloudSwXtch

High Availability	171
Mesh	181
Bridge Type 2	186
Bridge Type 1	190
Protocol Conversion and Fanout	192

cloudSwXtch > Monitoring cloudSwXtch

Azure Monitoring	193
Prometheus Monitoring	196
cloudSwXtch > Testing cloudSwXtch	
Testing cloudSwXtch	201
swxtch-top	202
swxtch-tcpdump	210
Troubleshooting	211
cloudSwXtch > cloudSwXtch How To's	
How to License a cloudSwXtch	212
How to set MTU size	213
How to Find xNIC Logs	219
How to Peer between VPCs in Different Regions for AWS	224
cloudSwXtch > Market Use Cases > Media Use Cases	
Media Use Cases	230
Media Multicast made easy with cloudswXtch	231
Hitless Merge - 2022-7	233
Media support for Compressed and Uncompressed Workflows	234
Protocol Fanout	236
Disaster Recovery	237
cloudSwXtch > cloudSwXtch API	
Monitoring API	238
Configuration API	250
cloudSwXtch > Resources	
Resources	261

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.

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

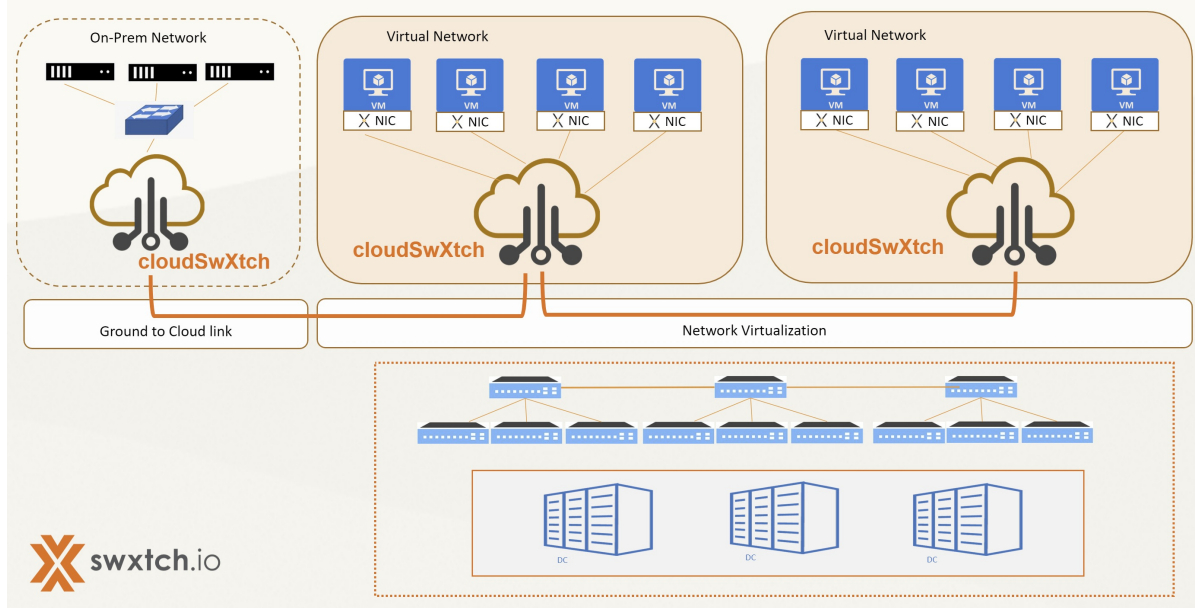
- Microsoft's [Azure](#)
- Amazon's [AWS](#)
- Google's [GCP](#)
- Oracle's [OCI](#)
- On-Premises for Bridge

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 (<3us), 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 Features

WHAT TO EXPECT

The following section gives a preliminary introduction to the main features available while using a cloudSwXtch. For additional information, please visit their respective articles.

- [Multicast](#)
- [Protocol Fanout](#)
- [High Availability](#)
- [Mesh](#)
- [Ground to Cloud <==> Cloud to Ground \(Bridge\)](#)
- [wXcked Eye for Monitoring](#)

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.

Protocol Fanout

cloudSwXtch supports a unique feature called **protocol 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 streams while making the network more efficient and reducing egress costs.

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.

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.

Ground to Cloud <==> Cloud to Ground

A user can connect their On-Prem network to their cloudSwXtches in the Cloud via the [bridge application](#).

wXcked Eye for Monitoring

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.

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

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.

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.

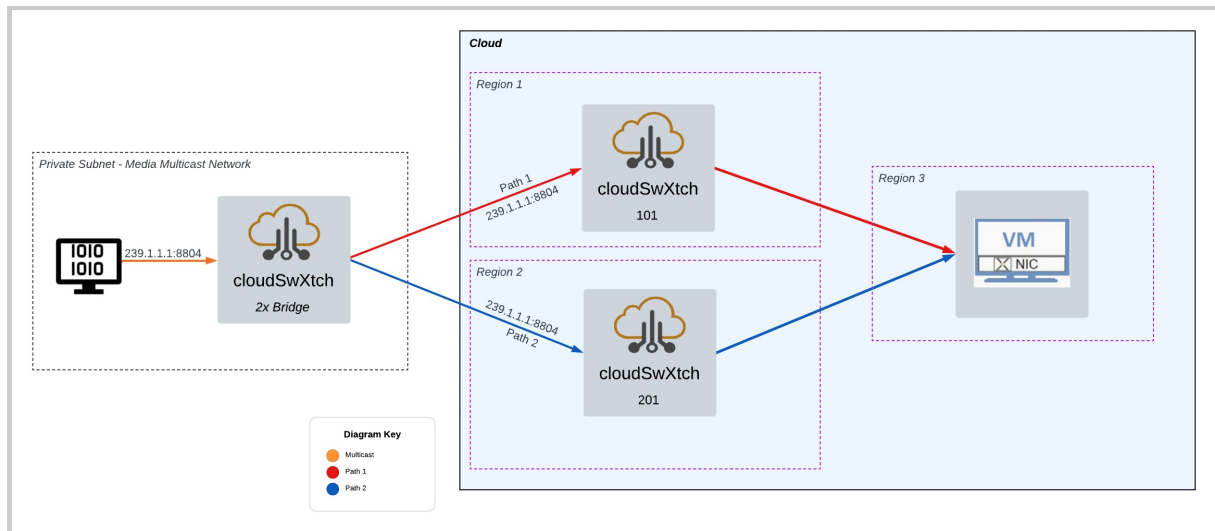
In addition, there is no limit to the number of multicast groups per data path. If one cloud, availability zone or region should go down, then the data is still sent in the other 2-8 paths, ensuring that the consumer gets the necessary data. Consumers can also be put into different clouds, availability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

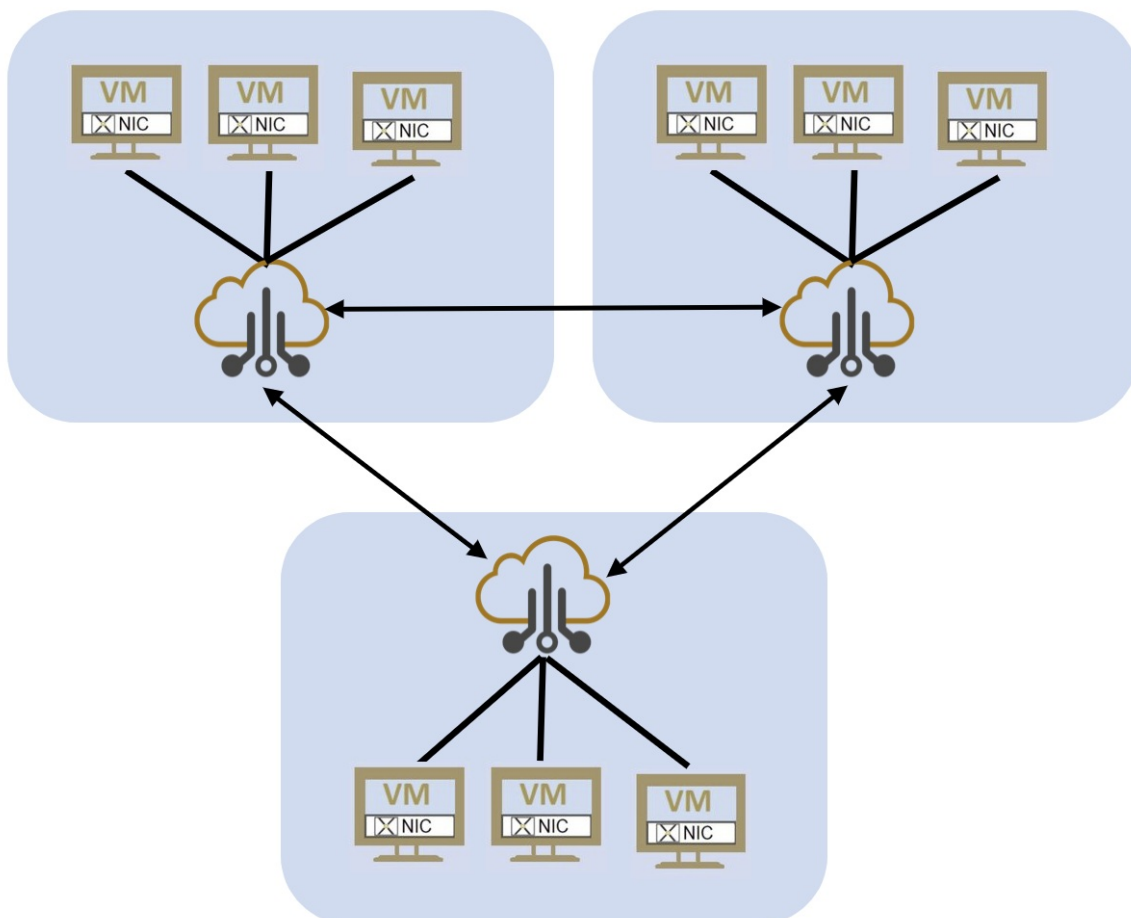
A mesh connects cloudSwXtches in a variety of dispersed network locations – different Vnets, regions, clouds, subnets, data centers, ect.). Additionally, a mesh is a way to group two or more swXtch's together to act as one to gain network performance.

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.

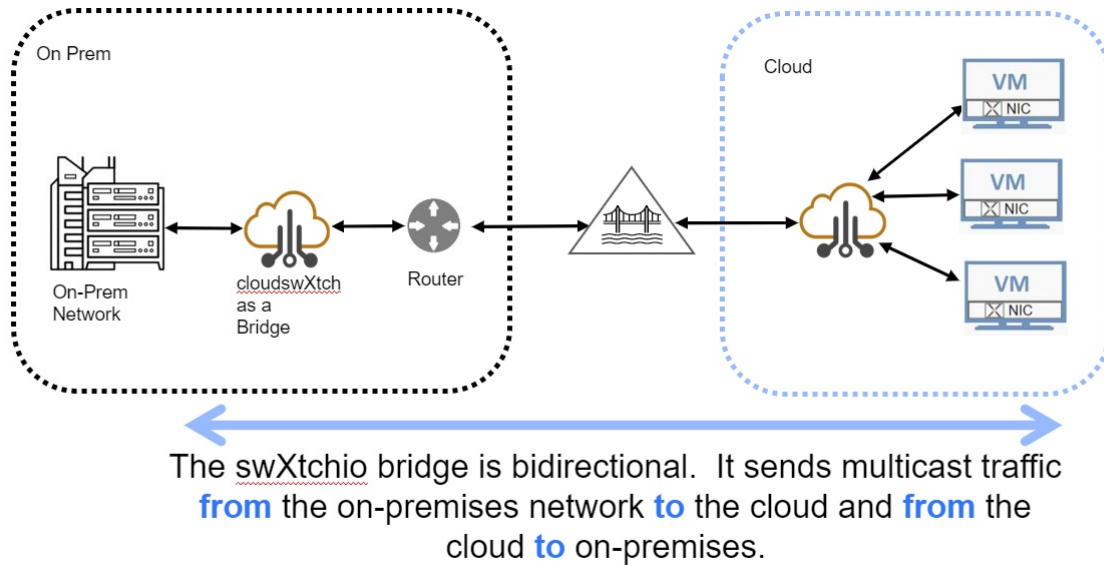
IMPORTANT

- 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 network and cloud network. The source network can be a bare-metal, on-prem network. The destination network can be a cloud virtual network with a **cloudSwXtch** instance. With **cloudswXtch**, multicast traffic generated on the on-prem network can be received and processed in the cloud which then in turn can be sent to the on-prem network.



From on-prem to the cloud the bridge is dynamic in that as users in the cloud subscribe to a multicast via IGMP joins and then the bridge allows that traffic through. This ensures that only needed traffic goes through the VPN or Express Route/Gateway into the cloud. This guarantees the best use of the gateway and incurs less ingress bandwidth into the cloud.

Operation

The operation of the Bridge varies based on direction.

Ground-->Cloud

For Ground to Cloud a mesh must be configured between the cloudSwXtch and the Bridge on the ground. From then on, the ground to the cloud they operation is dynamic, 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 bridge via the CloudSwXtch through the mesh and then the Bridge allows that traffic through. When the user stops using the multicast and does an IGMP leave then the bridge stops sending the multicast.

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 Type 1. See our configuration pages for [Bridge Type 1](#) and [Bridge Type 2](#).

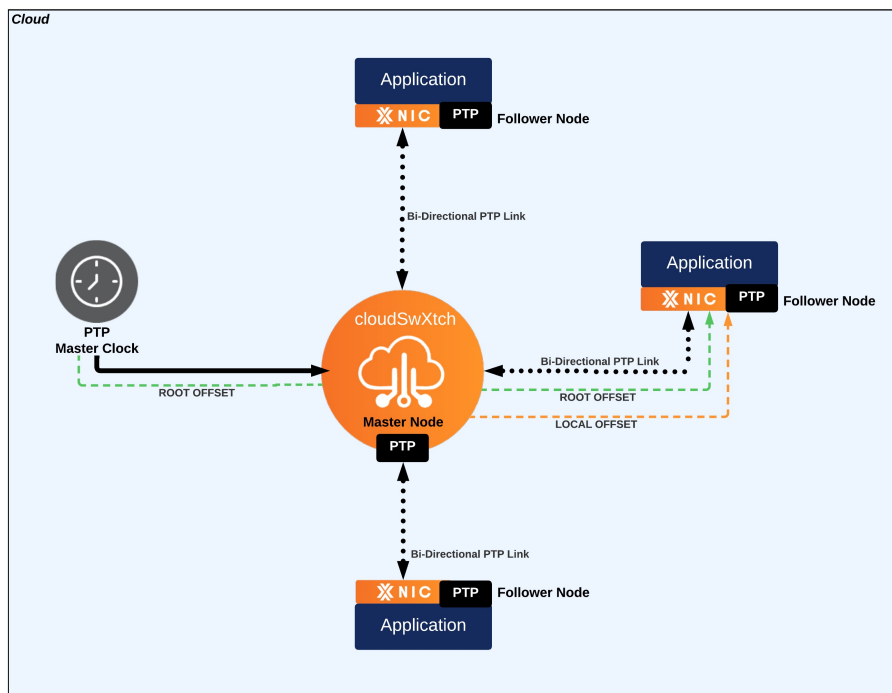
Precision Time Protocol

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

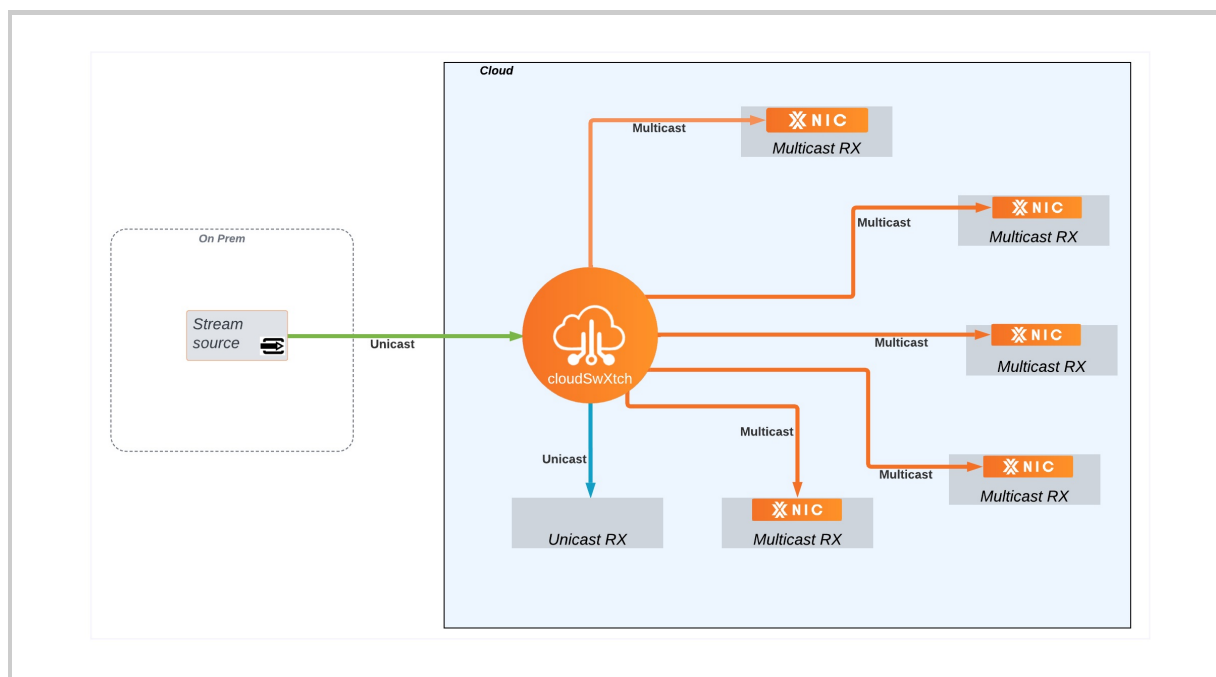
WHAT TO EXPECT

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.

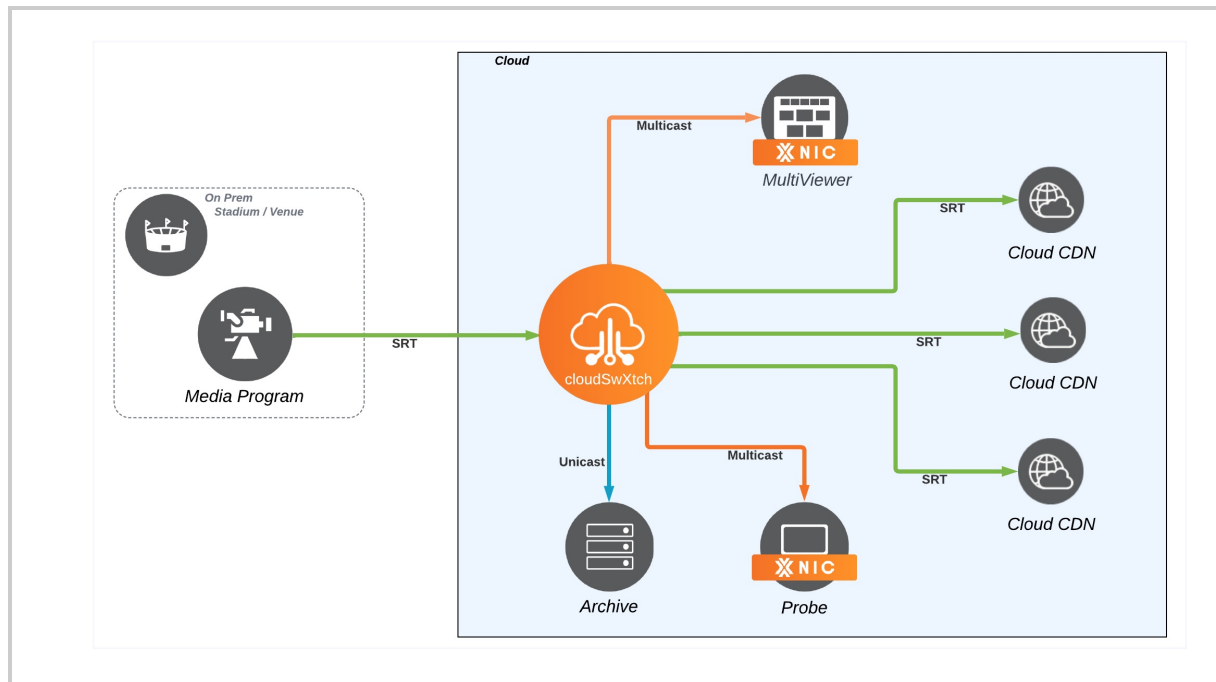
What is Protocol Conversion and Fanout?

It is not unusual 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 in 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 in 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

cloudSwXtch Sizing Guidelines

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

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](#).

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 the 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 cloudSwxctch must be a minimum of 4 Cores and 16 GiB memory.

Firewall and Security Group Rules

The xNIC software and the cloudSwxctch 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 feature, 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 and 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	tcp	65107	cloudSwXtch
ctrl-subnet	udp	319-320	cloudSwXtch
ctrl-subnet	tcp	9200	cloudSwXtch

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 subnets 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.

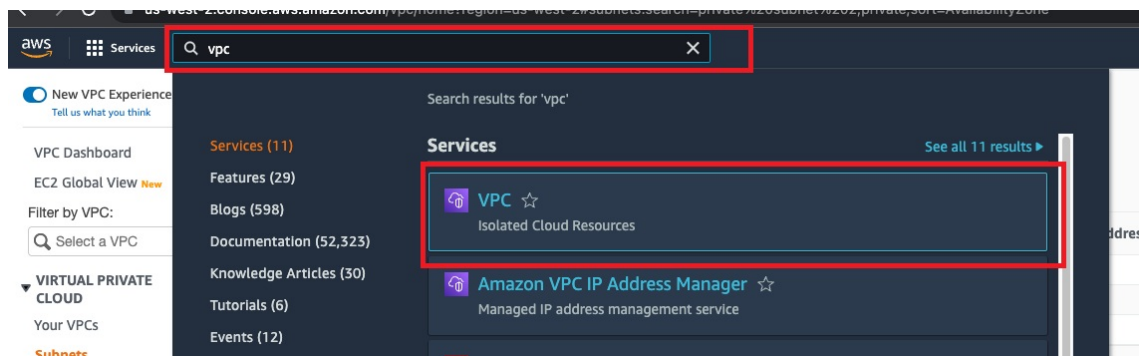
In this section, users will learn how to validate whether a subnet exists to be used as both the control and the data plane for their virtual network. This is in preparation for cloudSwXtch installation on AWS. We will also walk through an alternative method of using 2 subnets, separating the control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

Method #1: Single-subnet

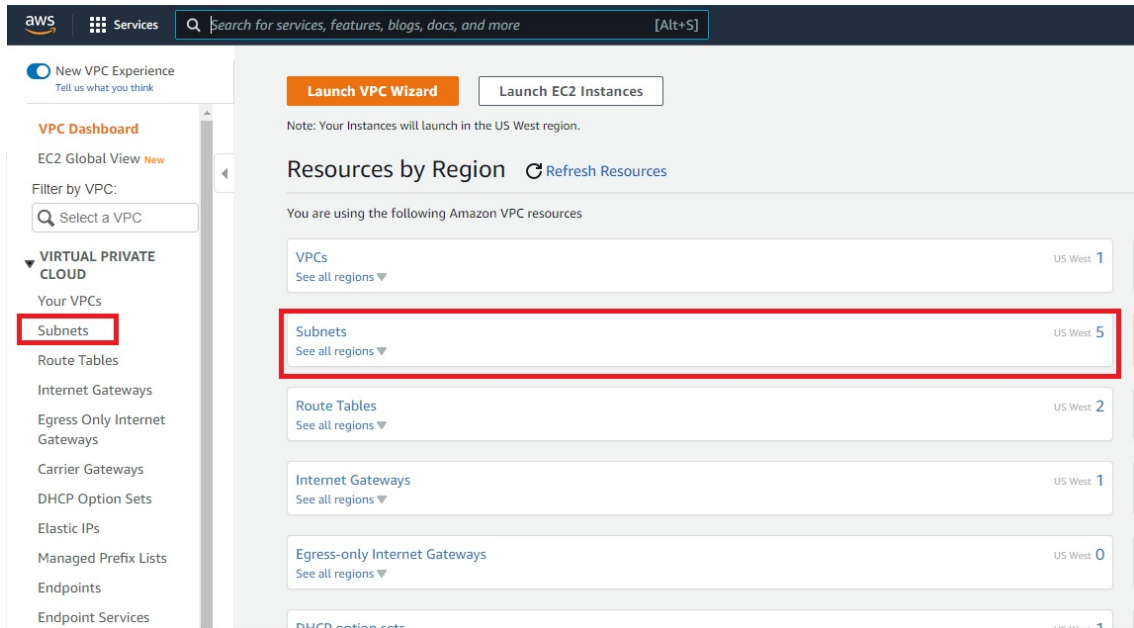
Typically, when deploying a VPC, a user will automatically create a subnet. During the main installation process, this subnet can be used for both control and data plane communications. This is the **preferred** method and will be used by a majority of users. Before installing cloudSwXtch, users should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

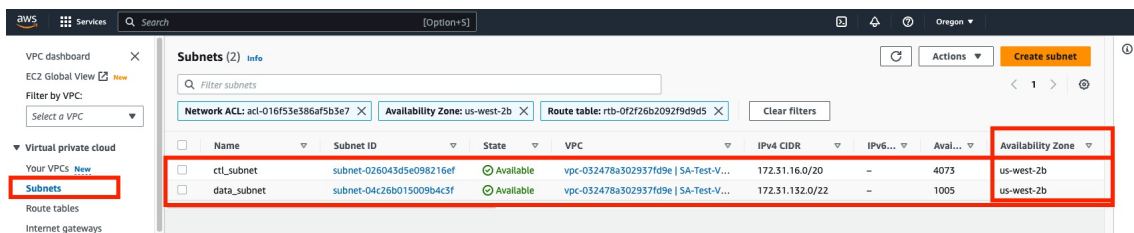
Alternatively, a user may decide that they want to have two separate subnets for their cloudSwXtch: one for the control plane and another for data. In addition, the subnets must be the same used for the xNIC installations. This method is recommended for individuals who want higher performance.

To accomplish this:

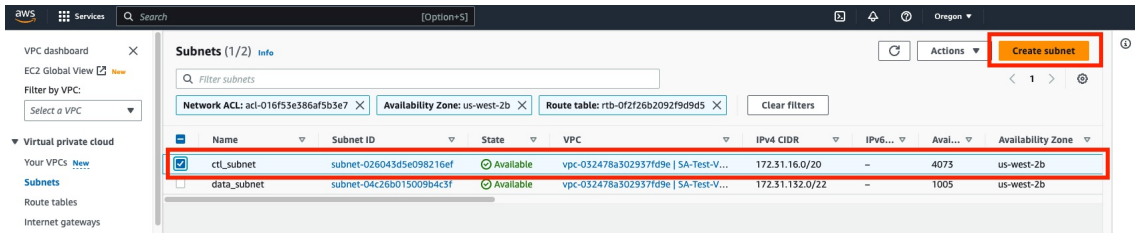
1. Navigate to the VPC Console in AWS.
2. Select Subnets under the Virtual Private Cloud tab or under Resources by Region in the VPC Dashboard.
3. Check that 2 subnets exists: one for the data and another for the control plane. Ensure that both subnets are in the same Availability Zone. This allows be both NICs to be connected on the EC2 instance at the same time.

Naming your subnets

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



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



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.

VPC > Subnets > Create subnet

Create subnet Info

VPC

VPC ID
Create subnets in this VPC.

vpc-032478a302937fd9e

Associated VPC CIDRs

IPv4 CIDRs

172.31.0.0/16

Subnet settings

Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

Subnet name
Create a tag with a key of 'Name' and a value that you specify.

data_subnet

The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

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

IPv4 CIDR block Info

.31.133.0/22

Tags - optional

Key	Value - optional
Name	data_subnet

Add new tag

You can add 49 more tags.

Remove

Add new subnet

Cancel **Create subnet**

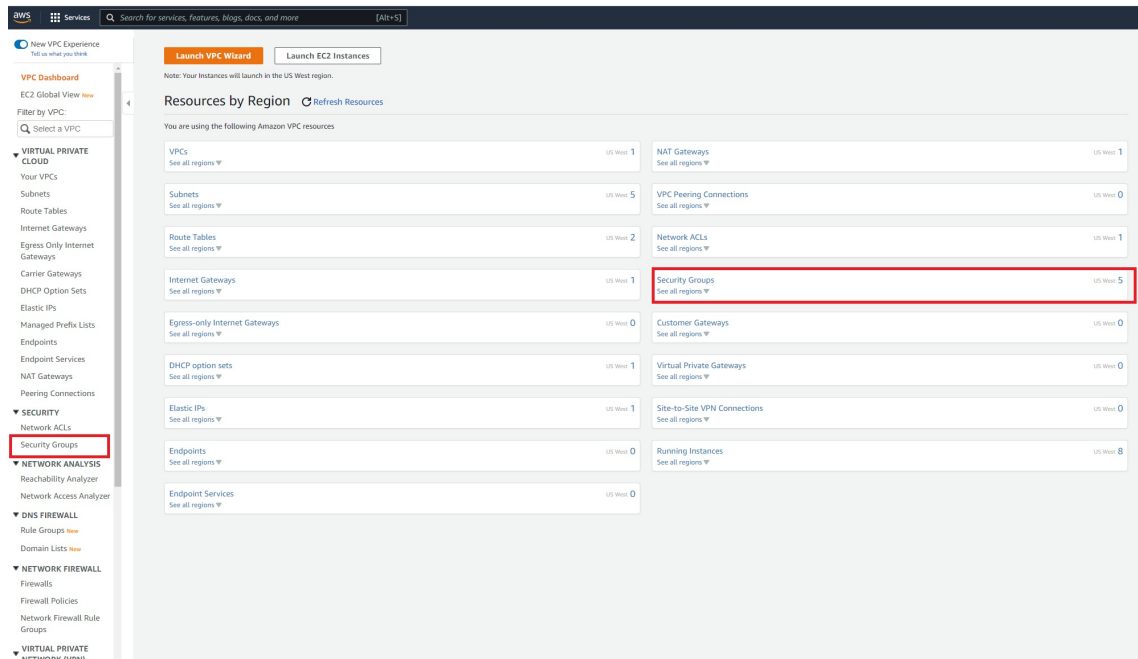
6. Click **"Create Subnet."** You should now have a new subnet on your list.

Verify Security Groups

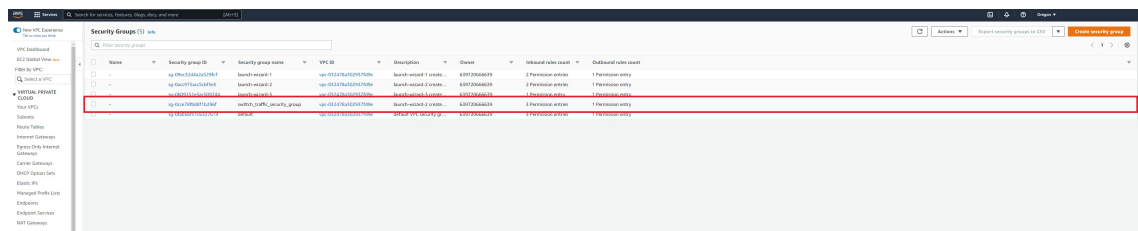
The security group contains the firewall settings for EC2 instances and interfaces (xNICs).

To ensure security groups are set up properly for cloudSwXtch:

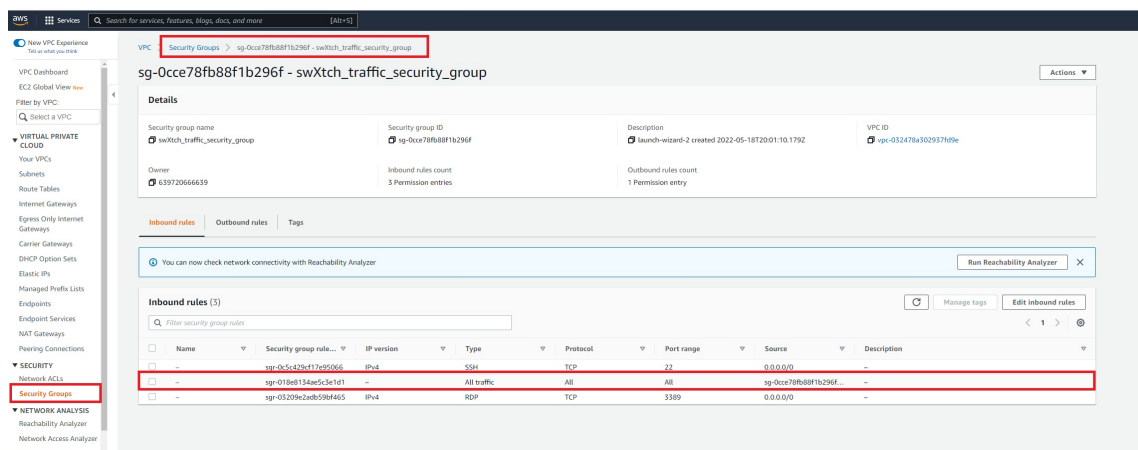
1. Navigate to the VPC console.
2. Select the "Security Groups" link as shown below. (Note: There are multiple ways to get to the "Security Groups" page.)



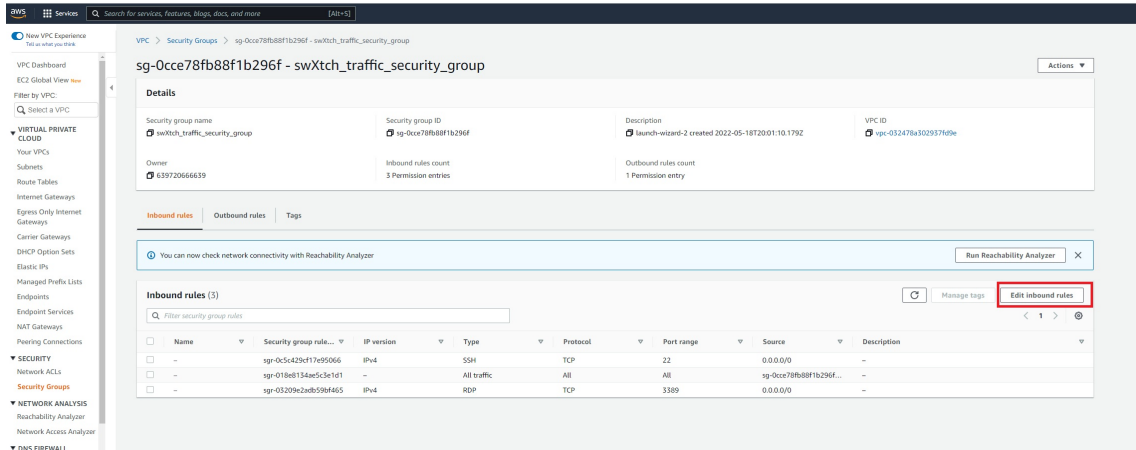
3. Select the Security Group that is normally used to create your EC2 instances for your application. (Note: The names in the example will be different in your environment.)



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."

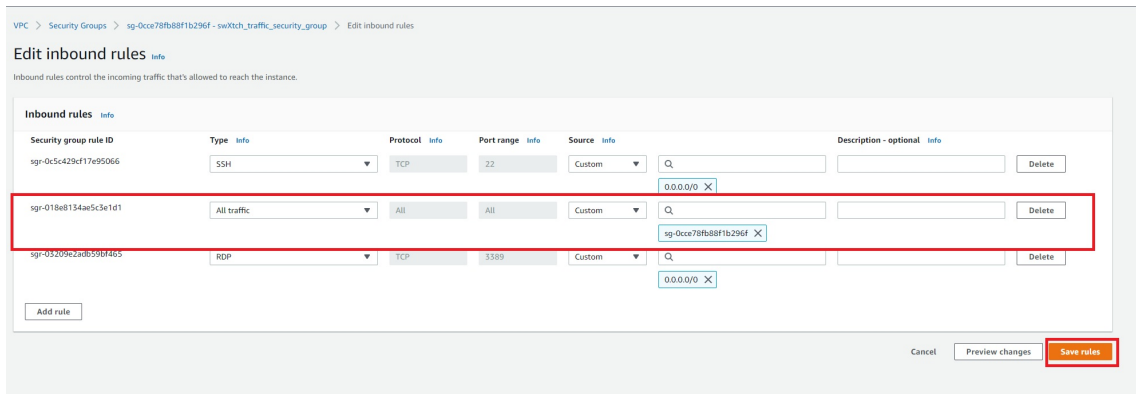


5. If an inbound rule does not exist, create it by selecting "Edit inbound rules."



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.



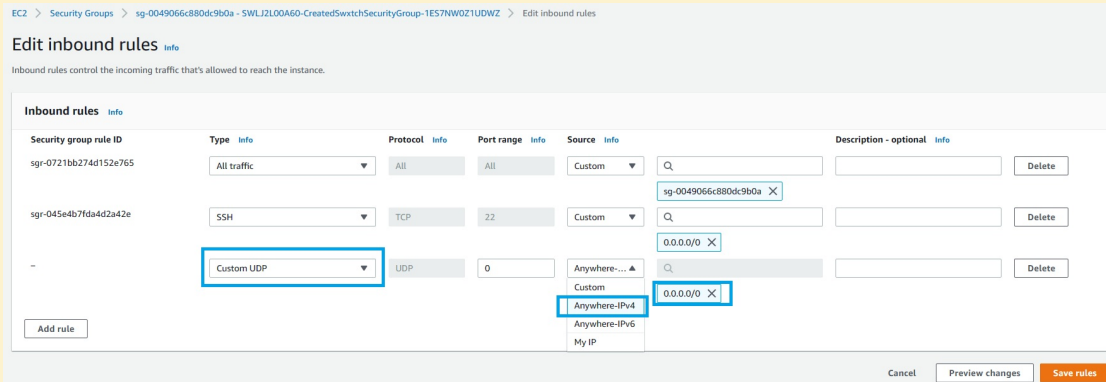
8. Save the rule.

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



Create SSH Key Pair

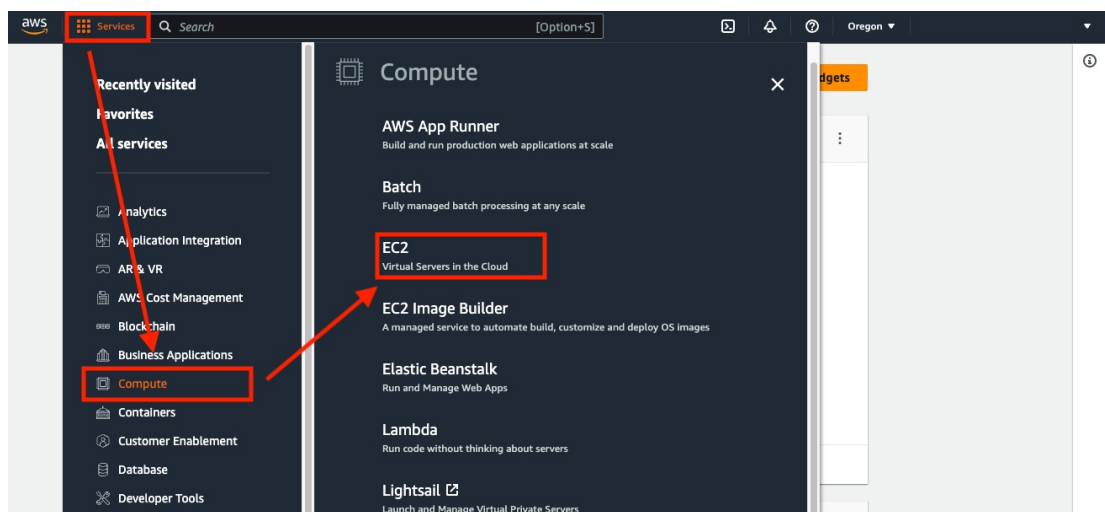
WHAT TO EXPECT

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.

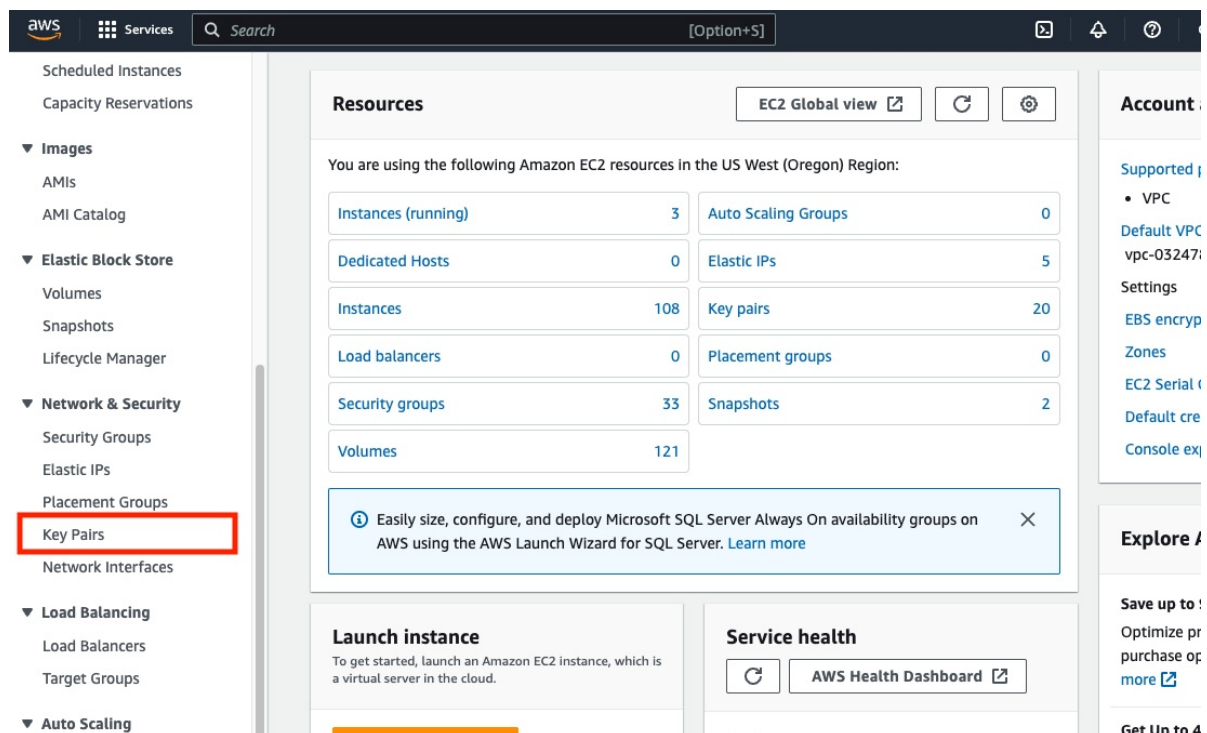
In the *AWS Management Console*, make sure you are in the region where you plan to use the cloudSwXtch instance.

1. Navigate to EC2.

1. Select the "Services" menu in the AWS Management Console.
2. Click "Compute."
3. 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**.
 1. A file with the desired extension will be downloaded to your computer (secret private key), and the other half of the pair will be stored on AWS for later use (public key, used in conjunction with your private key to validate the access).

[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

Enter key pair 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](#)

Upgrade cloudSwXtch on AWS

Keeping Your cloudSwXtch Up-to-date

When new versions are available in the Market Offering and a upgrade is desired, please use either options:

Upgrading cloudSwXtch via the cloudSwXtch.

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

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

Example:

Bash	Copy
<pre>sudo /swxtch/swx update -i localhost -v v2.0.34</pre>	

Upgrading cloudSwXtch via the xNIC

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

Text

None	Copy
<pre>swx update -v <desired version> --ip <ip of cloudSwXtch></pre>	

Example:

None	Copy
<pre>swx update -v v2.0.34 --ip 10.5.1.6</pre>	

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

Upgrade

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

Deploy cloudSwXtch with Terraform on AWS

WHAT TO EXPECT

In this article, you will learn how to deploy a cloudSwXtch instance on AWS by 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](#))

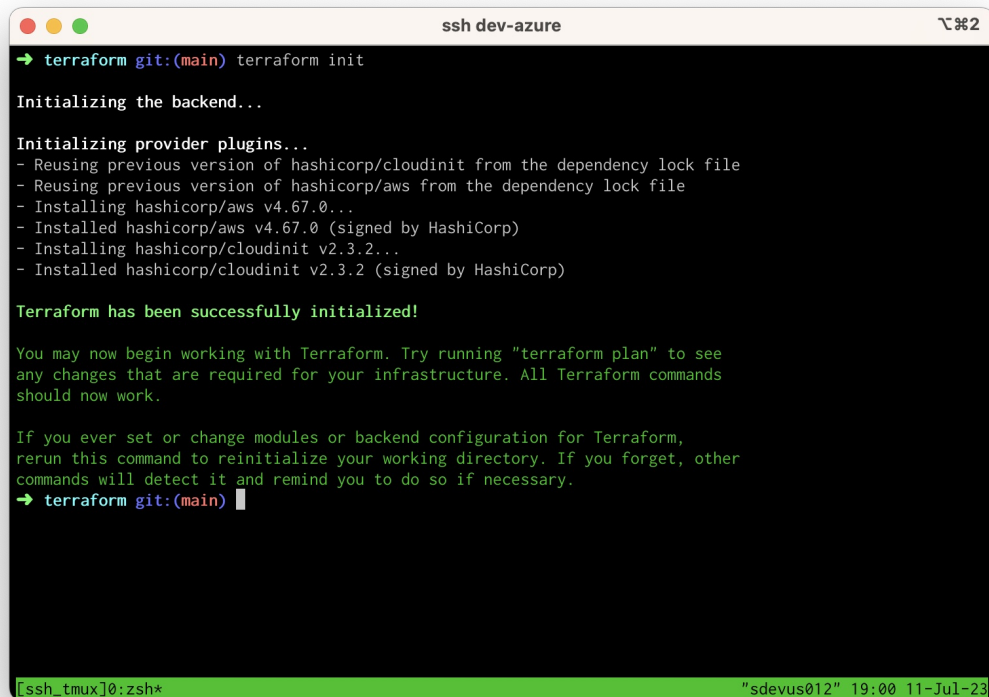
Console	Copy
<pre>\$ git clone https://github.com/swxtchio/cloudSwXtch-support.git</pre>	

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.



```
ssh dev-azure
→ 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)
```

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

+ 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"
+   },
+ ]

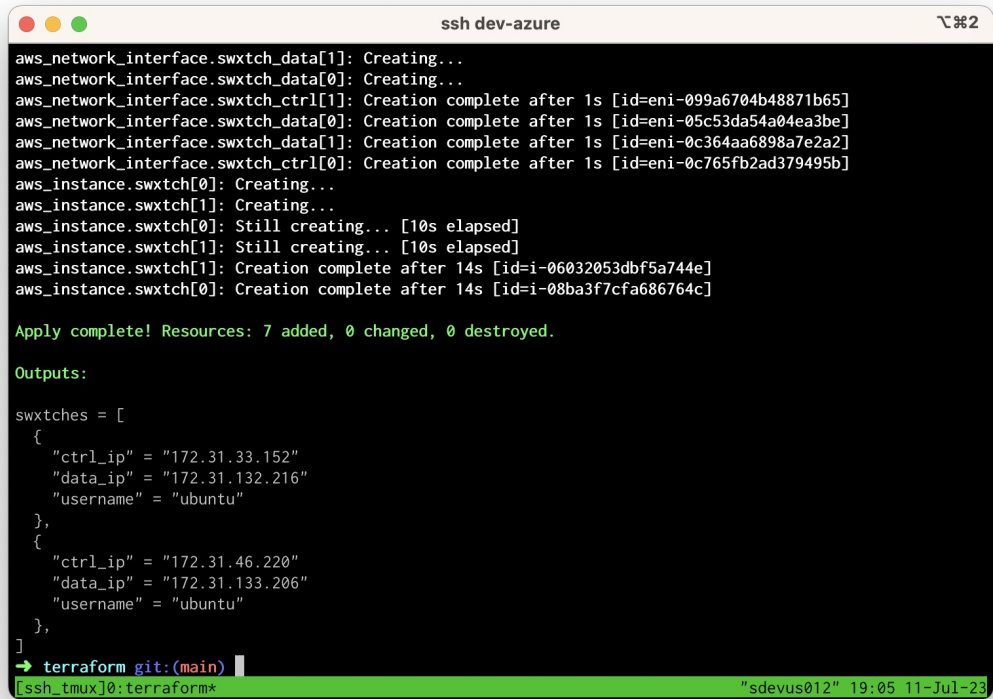
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly
these actions if you run "terraform apply" now.
→ terraform git:(main)
[ssh_tmux]0:zsh* "sdevus012" 19:02 11-Jul-23
```

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: 2 `aws_network_interface` for each of the cloudSwXtches and a `aws_instance` for each cloudSwXtch.

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

Console	Copy
terraform apply yes	

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



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

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

Outputs:

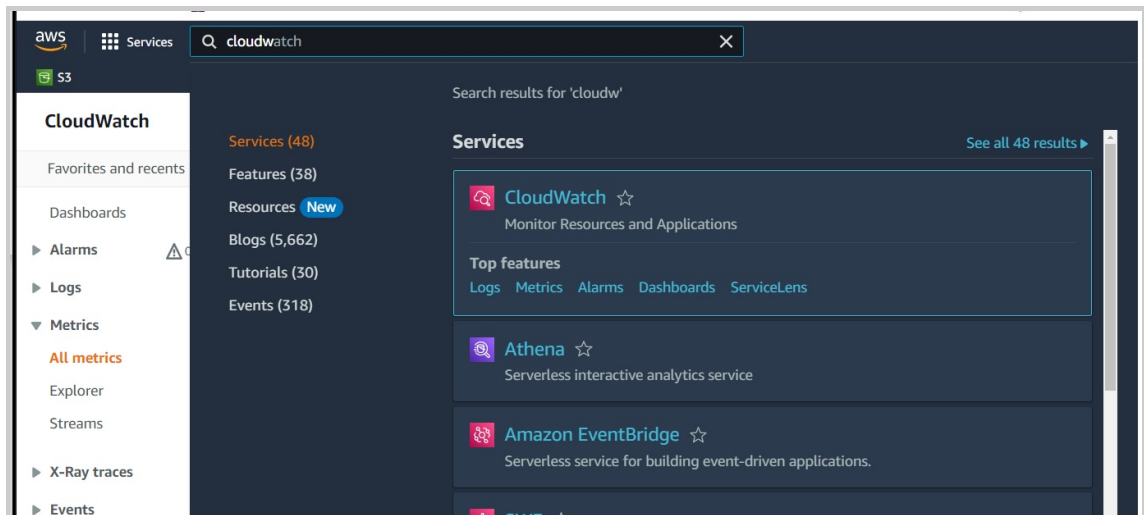
swtches = [
  {
    "ctrl_ip" = "172.31.33.152"
    "data_ip" = "172.31.132.216"
    "username" = "ubuntu"
  },
  {
    "ctrl_ip" = "172.31.46.220"
    "data_ip" = "172.31.133.206"
    "username" = "ubuntu"
  },
]
→ terraform git:(main) [ssh_tmux]0:terraform* "sdevus012" 19:05 11-Jul-23
```

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

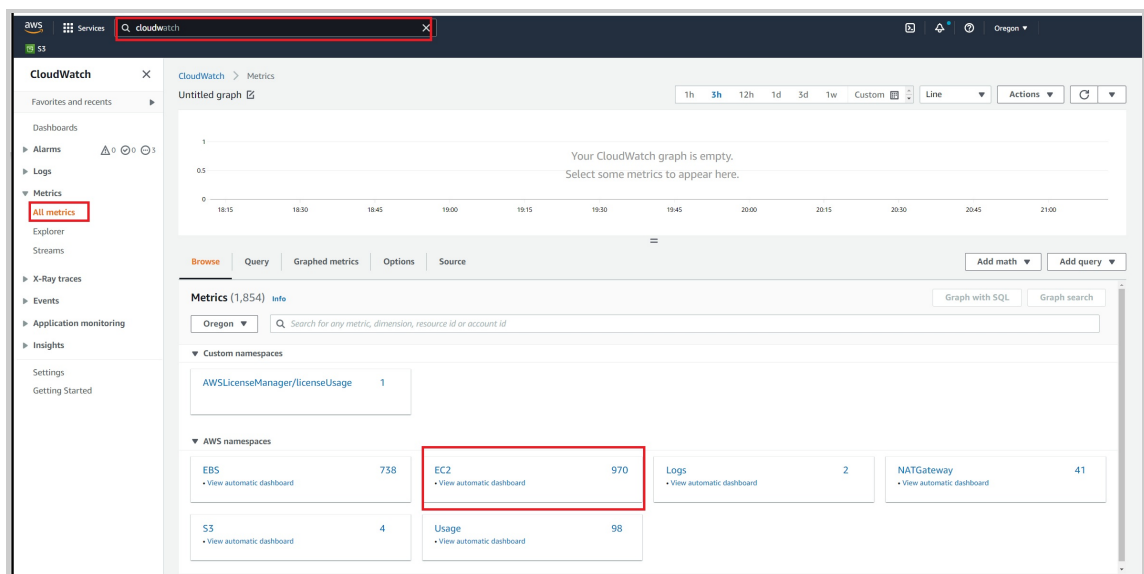
Check Health of cloudSwXtch Instance on AWS

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:

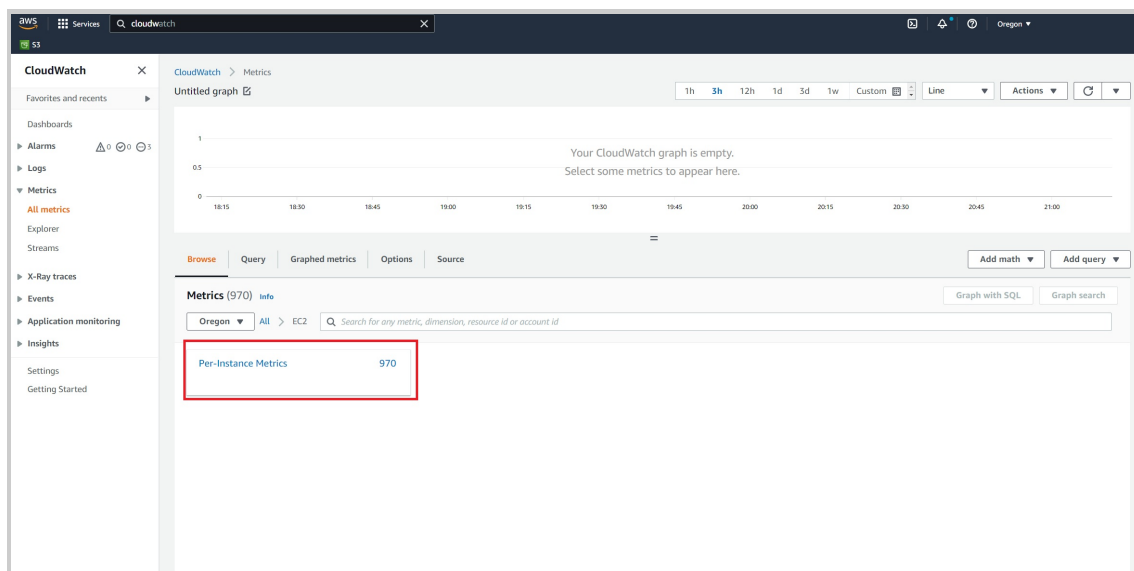
1. Search for "CloudWatch" in the AWS Search bar.



2. Select "All Metrics" on the left tree menu under "Metrics."
3. Select "EC2."

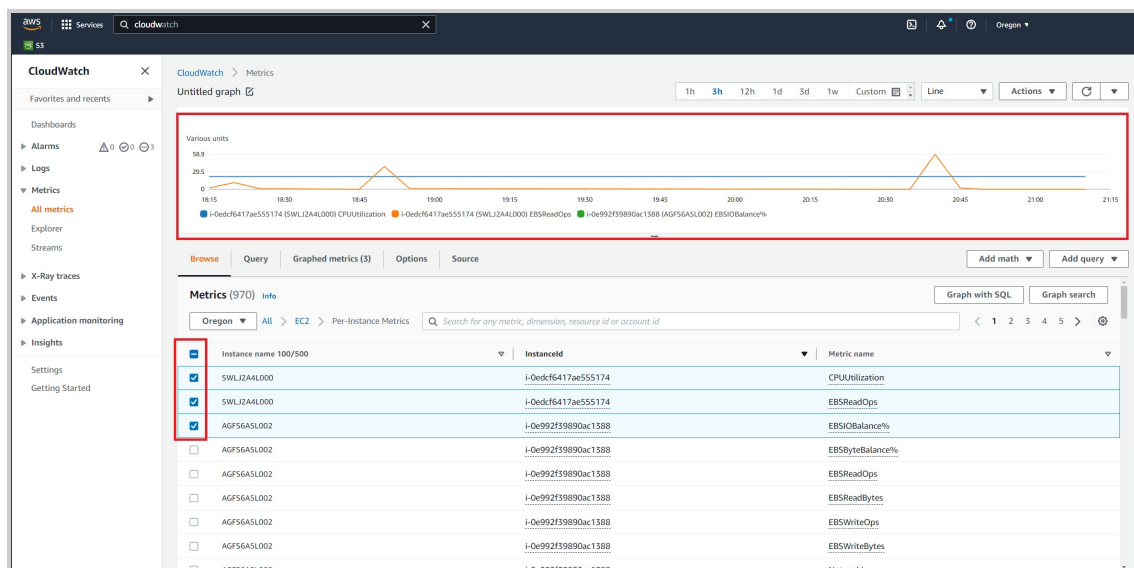


4. Select "Per-Instance Metrics."



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

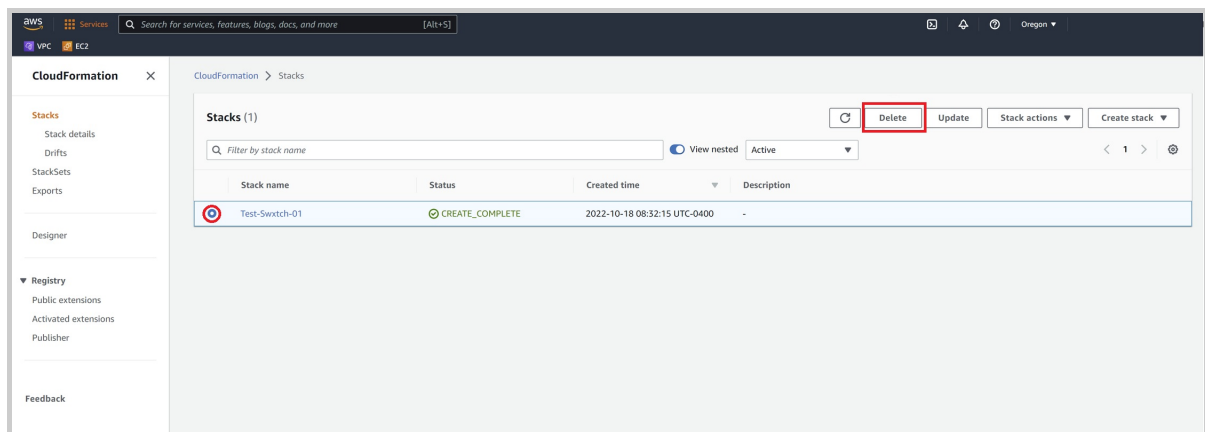
In this section, users will learn how to delete cloudSwXtch from their AWS environment.

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 a 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

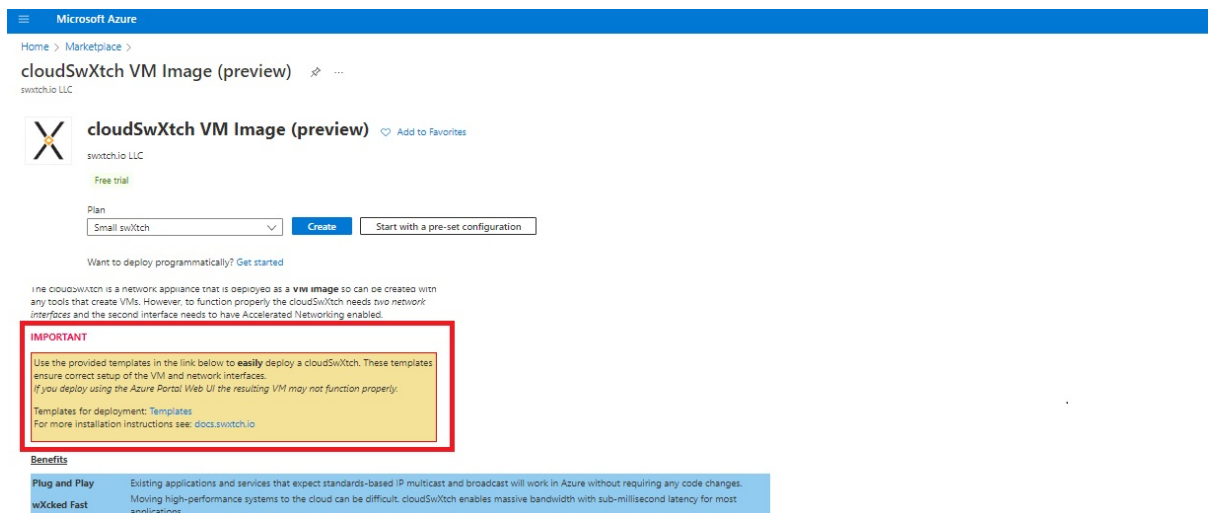
There are three methods that a cloudSwXtch instance can be deployed using the Azure Portal: via template, via Terraform, and via the Market Place.

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.

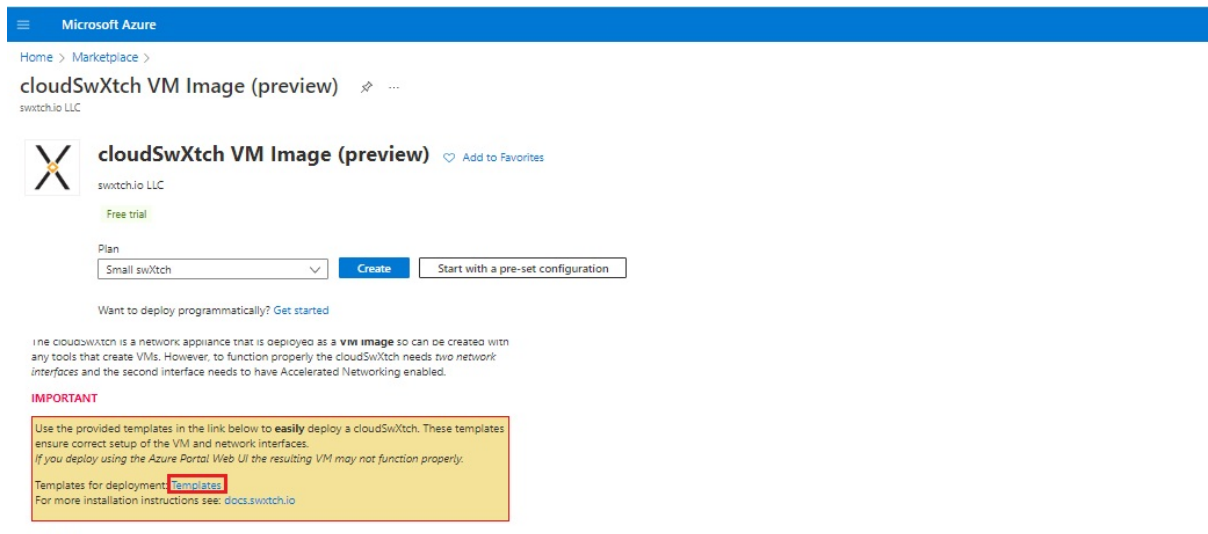
Template Method (PREFERRED):

1. [Review system requirements.](#)
2. [Validate subnets on Azure.](#)
3. [Create Azure cloudSwXtch Template.](#)
4. [Install cloudSwXtch on Azure.](#)

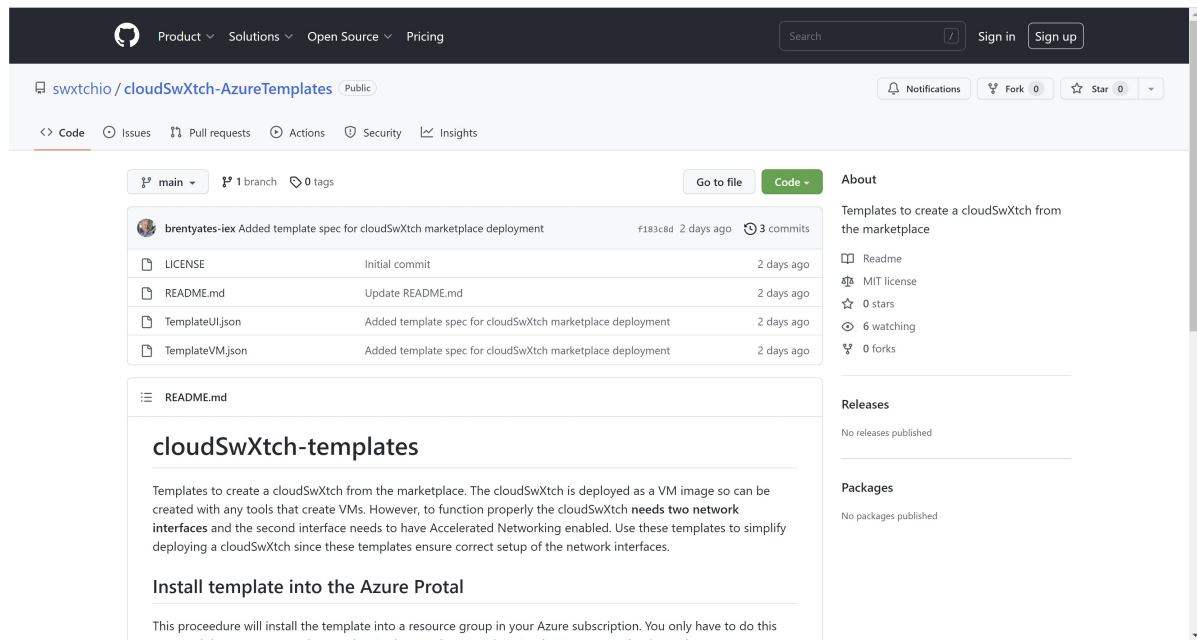
The template method is also mentioned in the Market Place cloudSwXtch installation as highlighted below.



This screen also shows more information about the template creation method by selecting the hyperlink below.



Selecting this link will open the page below:



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.

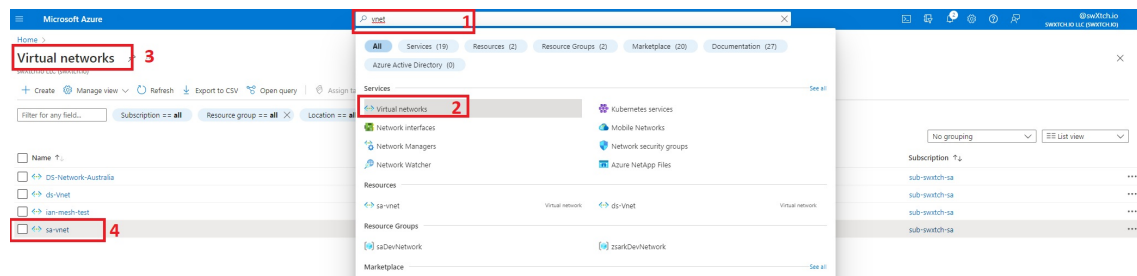
- It must contain at least one subnets that's used for both the control and data plane communication.
 - It is recommended that it is **private facing** and **does not auto-assign public IPs**.
 - This single subnet will be used for xNIC installation.

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.

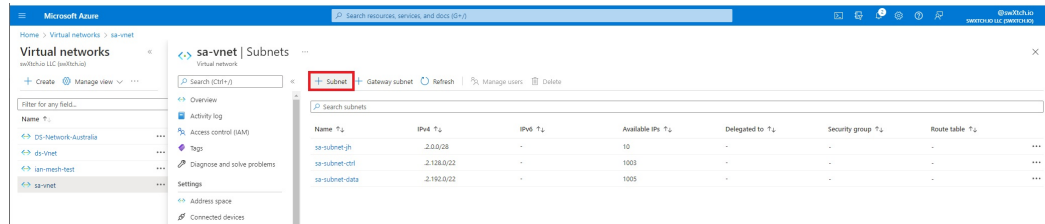
To validate:

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 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 (2.192.0 - .2.195.255 (1019 + 5 Azure reserved addresses))

☐ 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 example, for the *small* tier, users will be limited to 10 endpoint connections. If you know you will need more than that, consider deploying a larger sized cloudSwXtch as you walk through the deployment steps below.

NEXT STEP: Creating an Azure cloudSwXtch Template

After validating the subnets on Azure, continue you 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 the 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 in to the Azure Portal.** You will need permissions to create and manage virtual machines.
 - a. virtual-machine-contributor

2. **Open Cloud Shell.**



If you need help setting up your Azure cloud-shell, use the below link 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:

None	Copy
<pre>rg="<your-rg-here>"</pre>	


Example below:

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

Home > Marketplace >

cloudSwXtch VM Image (preview) ...

swtch.io LLC



cloudSwXtch VM Image (preview) Add to Favorites

swtch.io LLC

Free trial

Plan

Small swXtch Start with a pre-set configuration

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

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 the provided templates in the link below to simplify deploying a cloudSwXtch. These templates ensure correct setup of the network interfaces.

Templates to ease deployment: [Templates](#)
For documentation see: [docs.swtch.io](#)

```
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:~$ ng=saDevNetwork
```

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

None	Copy
<pre>git clone https://github.com/swtch.io/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:~$ ng=saDevNetwork
d@Azure:~$ git clone https://github.com/swtch.io/cloudSwXtch-AzureTemplates
Cloning into 'cloudSwXtch-AzureTemplates'...
```

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

None	Copy
<pre>cd cloudSwXtch-AzureTemplates</pre>	

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

```
Bash
Connecting terminal...

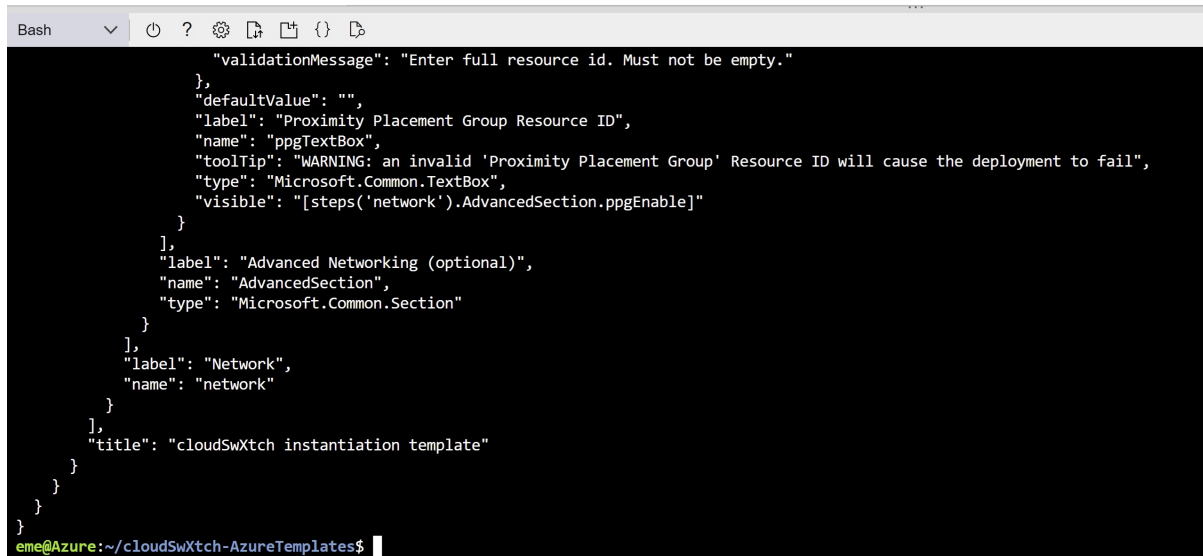
d@Azure:~$ cd cloudSwXtch-AzureTemplates
d@Azure:~/cloudSwXtch-AzureTemplates$ ls
LICENSE  README.md  TemplateUI.json  TemplateVM.json
d@Azure:~/cloudSwXtch-AzureTemplates$
```

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

None	Copy
------	------

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

The output should look like the below screenshot:



```
    "validationMessage": "Enter full resource id. Must not be empty.",
  },
  "defaultValue": "",
  "label": "Proximity Placement Group Resource ID",
  "name": "ppgTextBox",
  "toolTip": "WARNING: an invalid 'Proximity Placement Group' Resource ID will cause the deployment to fail",
  "type": "Microsoft.Common.TextBox",
  "visible": "[steps('network').AdvancedSection.ppgEnable]"
}
],
"label": "Advanced Networking (optional)",
"name": "AdvancedSection",
"type": "Microsoft.Common.Section"
}
],
"label": "Network",
"name": "network"
}
],
"title": "cloudSwxtch instantiation template"
}
}
}
}
}
eme@Azure:~/cloudSwxtch-AzureTemplates$
```

NEXT STEP: Azure cloudSwXtch Installation

After completing the template creation and [validating subnets](#), continue on to the main [Azure cloudSwXtch Installation guide](#).

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.

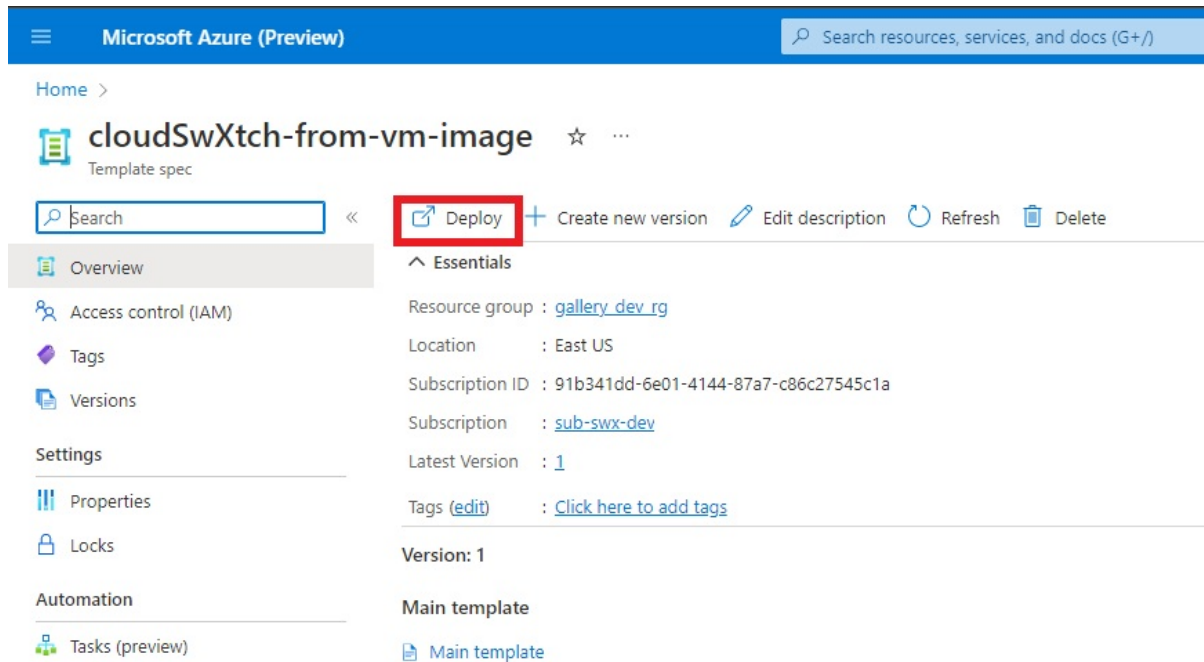
Deploying a cloudSwXtch instance

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.



In the cloudSwXtch commercial plan area, click on the “Choose a cloudSwXtch plan” dropdown and select a plan (small, medium or large). For more information on plans see: [cloudSwXtch Pricing](#)

6. In the "Project Details" area, select a Subscription.

7. Pick (or create) an Azure Resource Group.

8. In the "Instance details" area, notice how the region is filled in from the Azure Resource Group.

9. 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](#).

10. 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 swxtch.io for cloudSwXtch licensing.

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

12. 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.

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

15. Select **"Next - Network."**

16. 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.

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

17. Select a **"Data Subnet Name."**

18. **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 * ⓘ	<div>(new) pick-an-existing-vnet</div> <div>Create new</div>
Control Subnet Name * ⓘ	<div>(new) ControlSubnet (10.0.0.0/29)</div>
Data Subnet Name * ⓘ	<div>(new) DataSubnet (10.0.1.0/29)</div>

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 * ⓘ	<input type="text"/>
Enter static IP for network interface on DataSubnet * ⓘ	<input type="text"/>
Specify a Proximity Placement Group ⓘ	<input type="checkbox"/>

[Review + create](#)

[< Previous](#)

[Next : Review + create >](#)

19. Select "Review and Create."

20. Review the plan pricing.

21. Read the "Terms & Conditions."

22. 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.

Upgrade cloudSwXtch on Azure

Keeping Your cloudSwXtch Up-to-date

When new versions are available in the Market Offering and a upgrade is desired, please use either options:

Upgrading cloudSwXtch via the cloudSwXtch

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

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

Example:

Bash	Copy
<pre>sudo /swxtch/swx update -i localhost -v v2.0.34</pre>	

Upgrading cloudSwXtch via the xNIC

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

None	Copy
<pre>swx update -v <desired version> --ip <ip of cloudSwXtch></pre>	

Example:

None	Copy
<pre>swx update -v v2.0.34 --ip 10.5.1.6</pre>	

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

Why Upgrade?

To ensure that you experience the best functionality, upgrade all cloudSwXtches and xNICs whenever there is a new release.

Deploy cloudSwXtch with Terraform on Azure

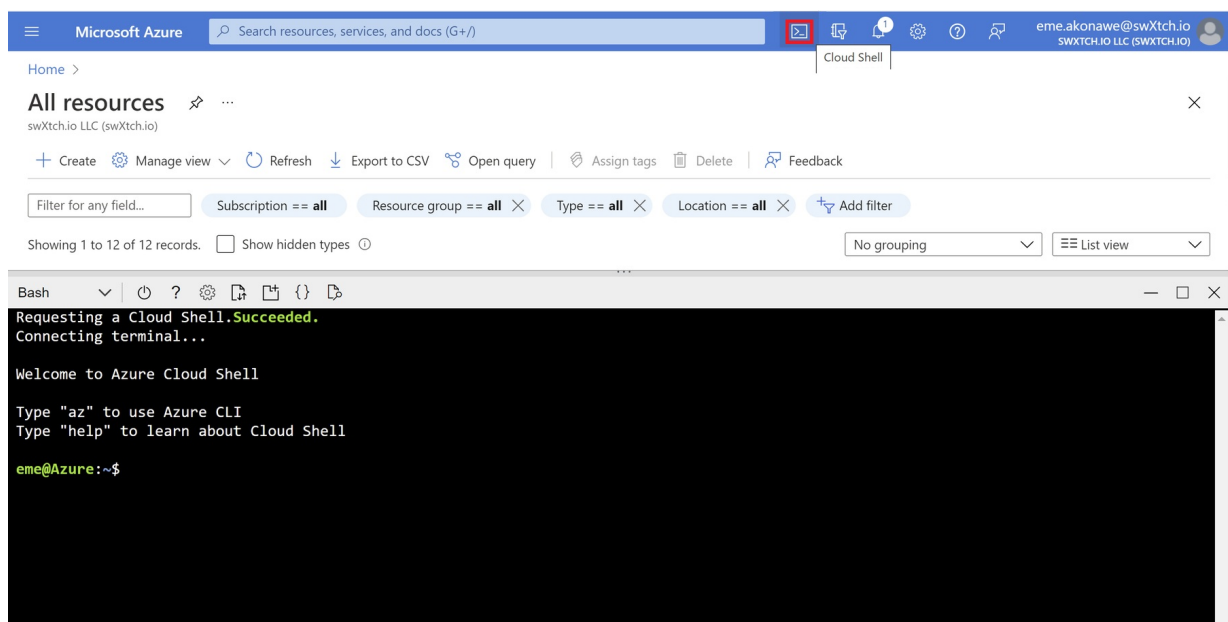
WHAT TO EXPECT

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**:

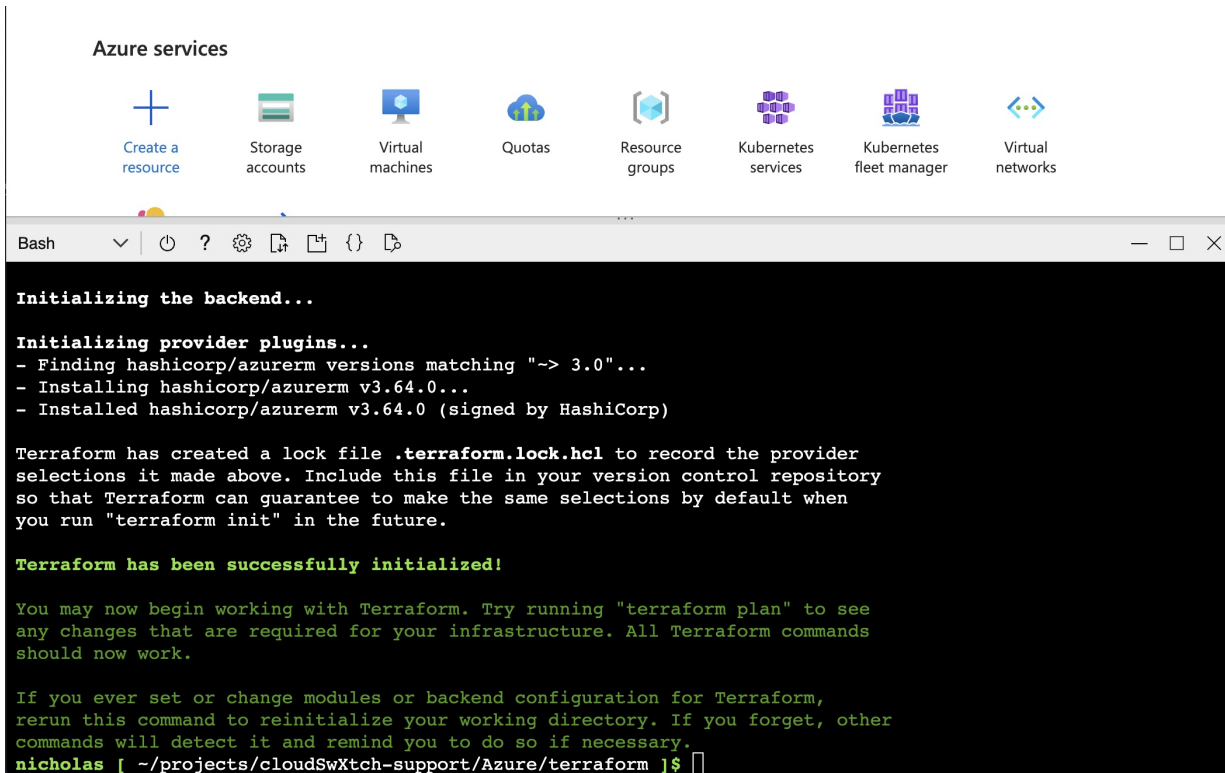
None	Copy
<pre>\$ git clone https://github.com/swxtch.io/cloudSwXtch-support.git</pre>	

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:

Console	Copy
<pre>\$ cd Azure/terraform \$ terraform init</pre>	



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 ]$
```

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

None	Copy
<pre>\$ terraform plan</pre>	

```

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.

None	Copy
Terraform apply yes	

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

STATIC IPs

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

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


```

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:

None	Copy
<pre>private_ip_address_allocation = "Static" private_ip_address = var.datanic_staticip</pre>	

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

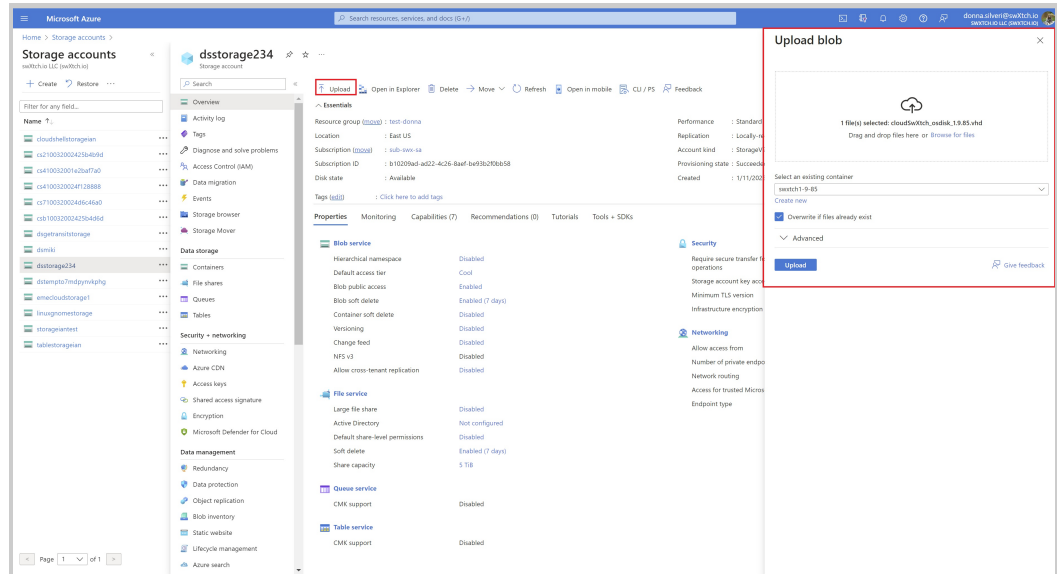
Log onto an environment that has access to the internet and download the following file (~30GB):

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

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

1. Place the file onto a machine with access to the Azure Air Gapped Environment.

2. Upload the files into an Azure storage account in the secure Azure Environment.
 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.

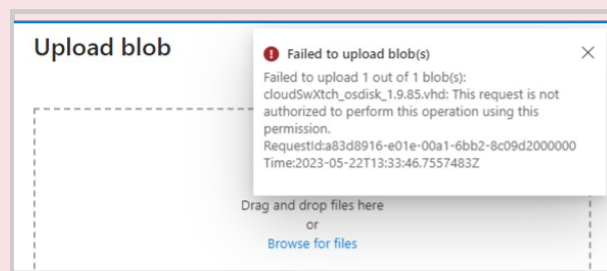


6. Start the upload and wait for it to complete.

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 *

Resource group * [Create new](#)

Instance details

Name *

Region *

Zone resiliency ☐

OS disk

OS type * ☒ Windows ☒ Linux

VM generation * ☒ Gen 1 ☐ Gen 2

Storage blob * [Browse](#)

Account type *

Host caching *

Encryption

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

Key management

Data disk

[+ Add data disk](#)

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

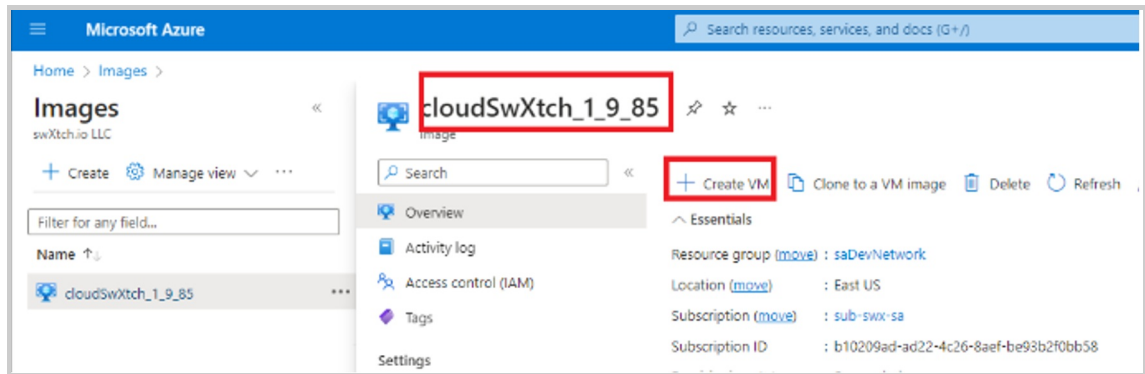
10. If tags are desired, then select **Tags** and enter the required tags.
11. The other fields can be left as default.
12. Select **Review and create**.
13. When validation passes, select **Create**. When it is complete, click **Go to Resource** to see the image.

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 has a search bar at the top. The 'Project details' section includes fields for 'Subscription' (sub-sw-5a), 'Resource group' (saDevNetwork), and 'Location' ((US) East US). The 'Network interface' section includes fields for 'Name' (cloudswtch01-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 'Select inbound ports' section has a dropdown menu with the text 'Select one or more ports'. A blue information box states: 'All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.' 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.

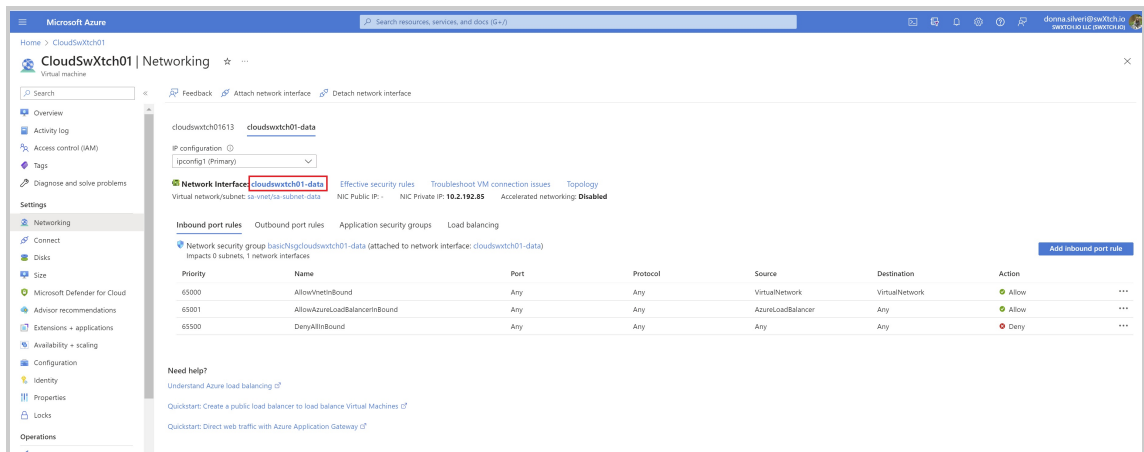
9. Select **Create**.

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

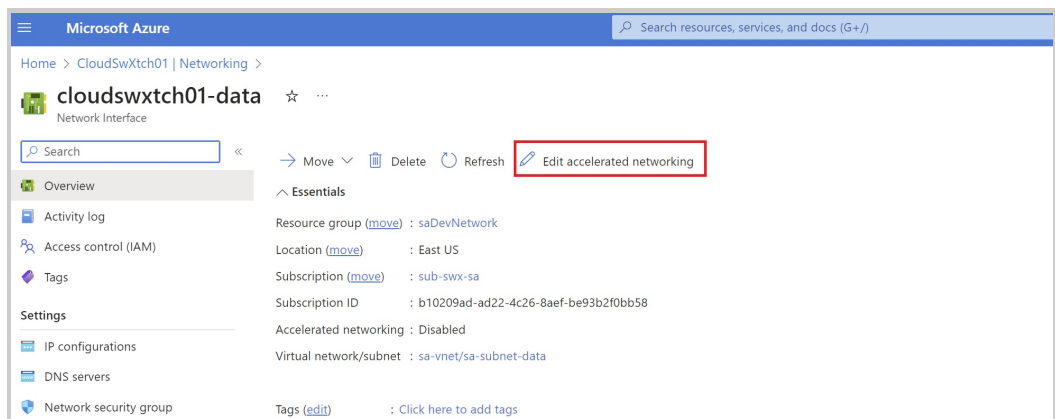
The screenshot shows the Microsoft Azure portal with the 'cloudswtch01-data' network interface selected. The interface is highlighted with a red box. The page shows the 'Overview' tab for the network interface. The 'IP configuration' section shows the 'ipconfig1 (Primary)' configuration. The 'Network interface: cloudswtch01-data' section shows the 'Virtual network/subnet: sa-vnet/sa-subnet-data' and 'NIC Public IP: 10.2.192.85'. The 'Inbound port rules' section shows a table of rules. The 'Need help?' section provides 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'.

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

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.

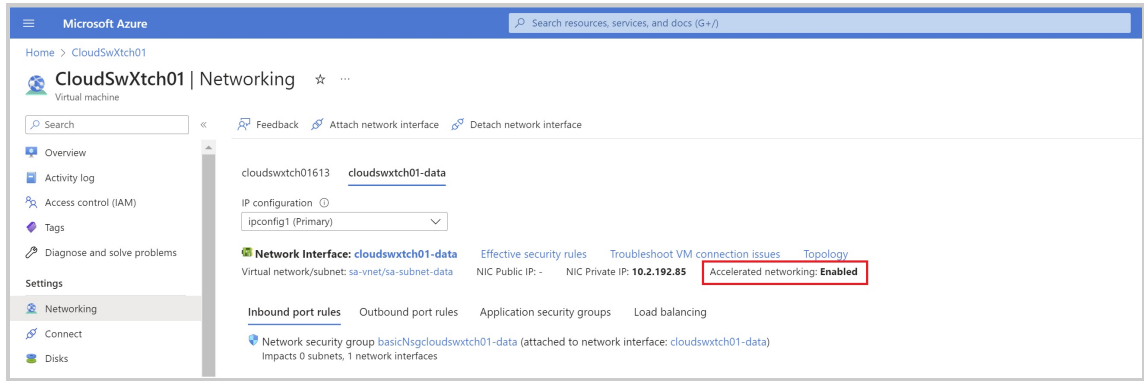
☒ I have validated that the operating system supports accelerated networking.

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

None	Copy
<pre>sudo /swxtch/swxtch-top dashboard --switch localhost</pre>	

3. The swXtch-top dashboard will display.

Information - v1.9.85			
CloudSwXtch001	v1.9.85	Auth. Type	None
SubscriptionId	b10209ad-ad22-4c26-8aef-be93b2f0bb58	Max Bandwidth	
ResourceGroupName	saDevNetwork	Max Clients	Unlicensed
SwxtchId	b11a934e-6182-492f-a7ba-7b5dbe84294a		
Status	OK	plan limits exceeded	Not Authorized

Totals					
Producers	0 pps	(0 bps)	Consumers	0 pps	(0 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	0 pps	(0 bps)	Switch TX	0 pps	(0 bps)

xNIC clients							
Name	ip	Version (XNIC)	RX pps	RX bps	TX pps	TX bps	HA

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

None	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

None	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

Max Bandwidth

Max Clients

License File

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 in Google Cloud Platform (GCP).

Currently, there is only one way of deploying a cloudSwXtch on GCP:

- [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.

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](#)

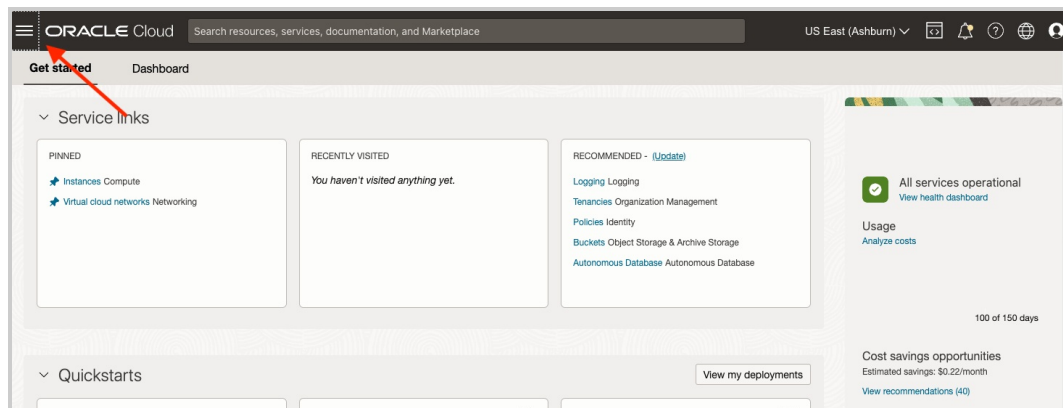
Please note: At this time, our only product offering in OCI is a BYOL instance of cloudSwXtch. This requires a user to contact swXtch.io for a license.

Prerequisites

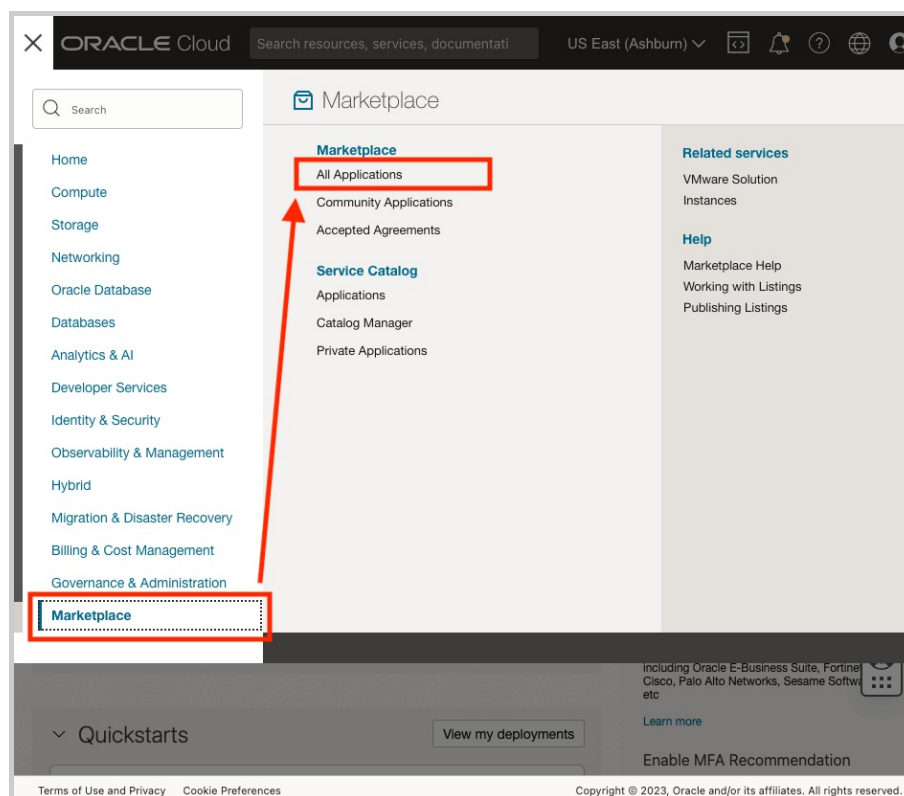
A user should have a **Compartment** established in their Oracle Cloud console before they start to deploy a cloudSwXtch. For more information about compartments, please see the [Managing Compartments](#) page under Oracle Cloud Infrastructure Documentation.

Step One: Navigate to cloudSwXtch in the Oracle Marketplace

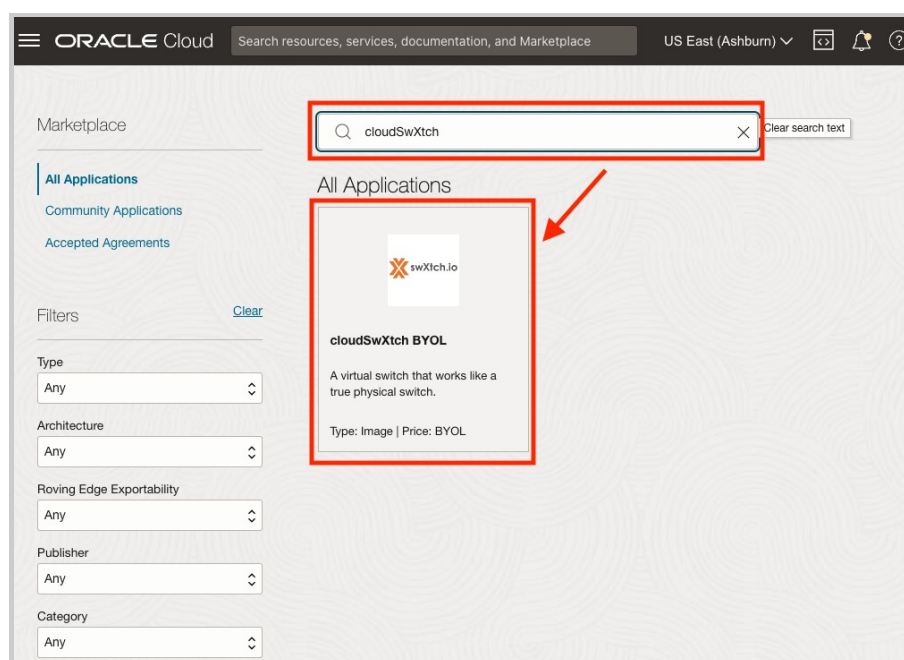
1. Log into Oracle Cloud.
2. Navigate to the Oracle Cloud Marketplace using the Navigation menu at the top left corner.



3. Select Marketplace and All Applications.

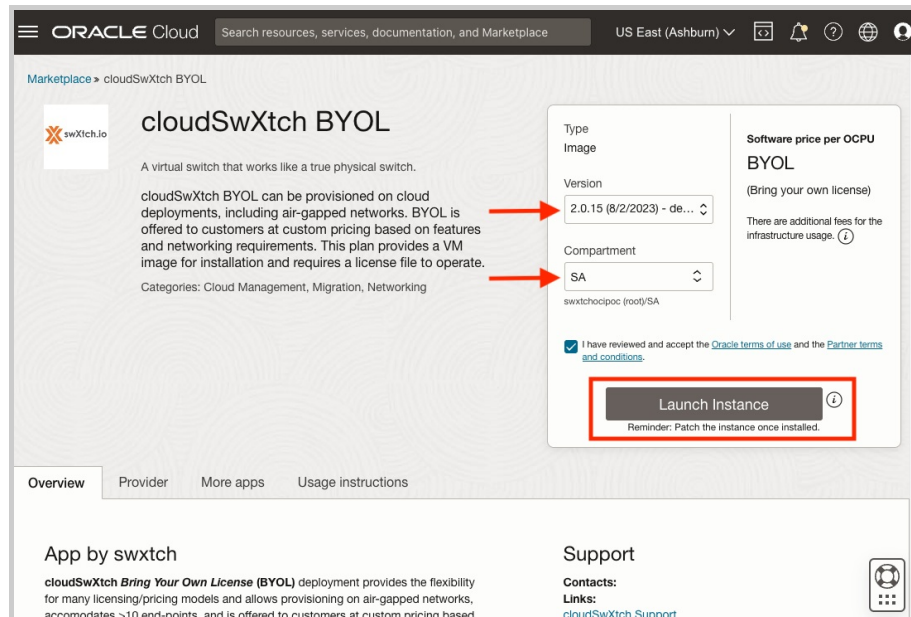


4. Search for cloudSwXtch and select the product, cloudSwXtch BYOL.



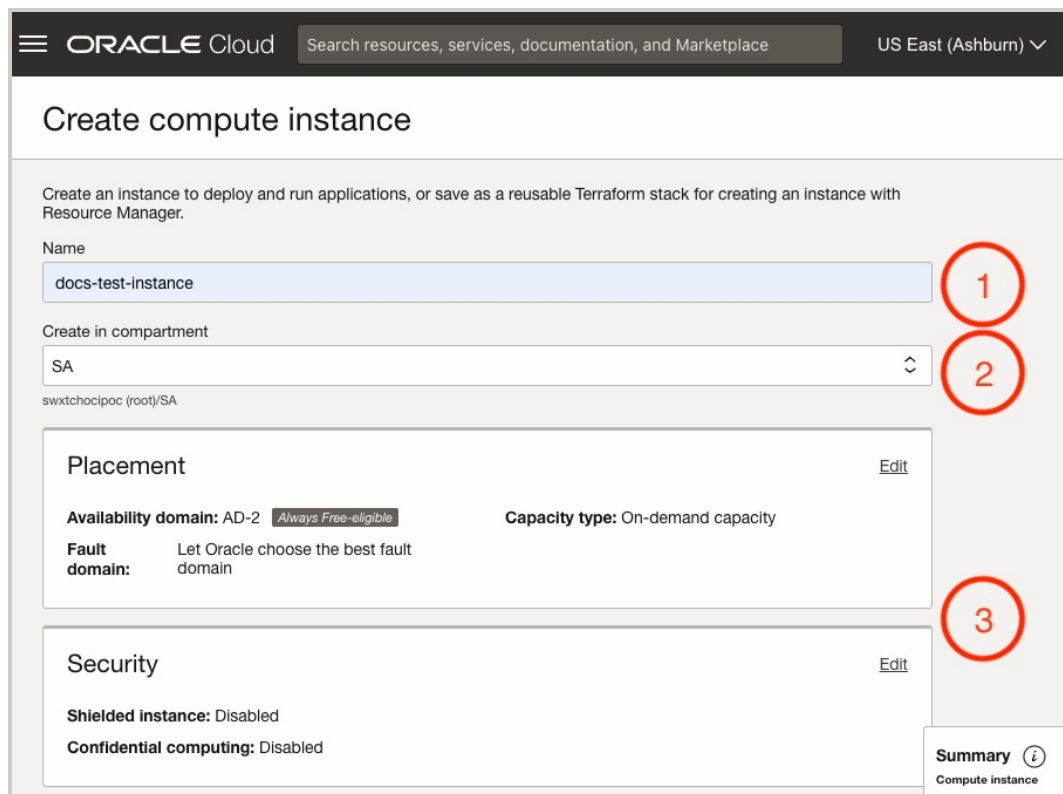
5. Select the Version and the Compartment that you will like to use. It is best to use the default since it will be the most recent version.

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

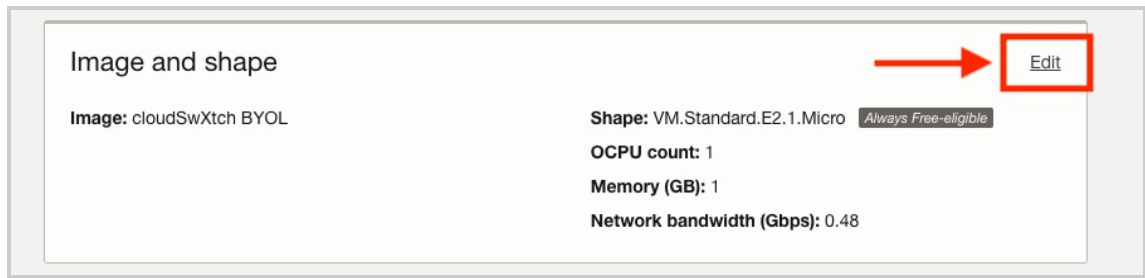


Step Two: Create Compute Instance

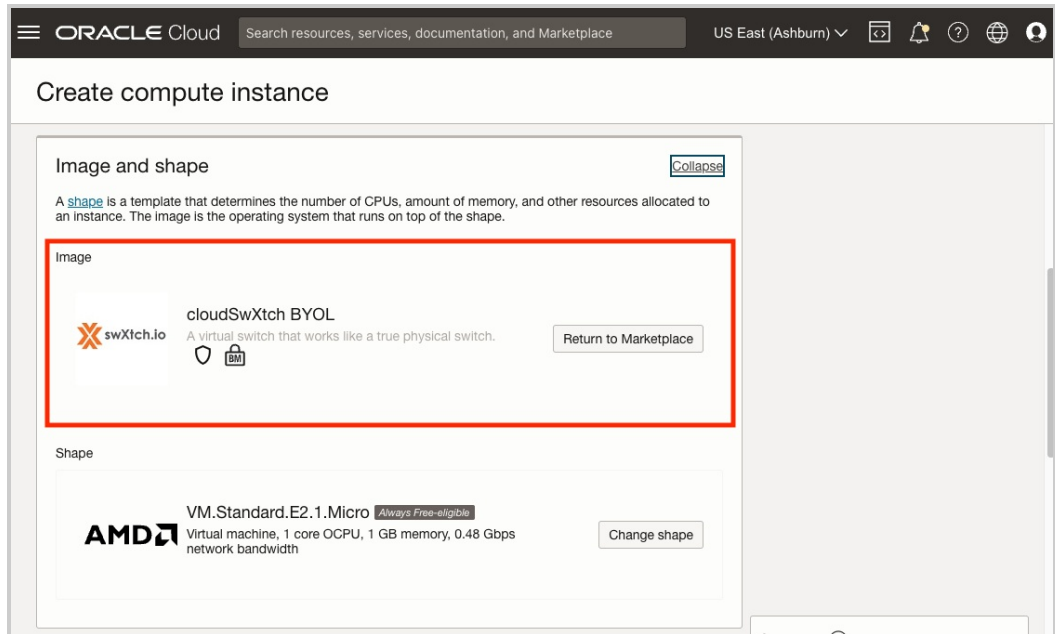
1. Give your **Compute Instance** a unique name.
2. Confirm that your desired **Compartment** is populated.
3. **Optional:** Edit selections for **Placement** and **Security**. This is dependent on a user's specific needs.



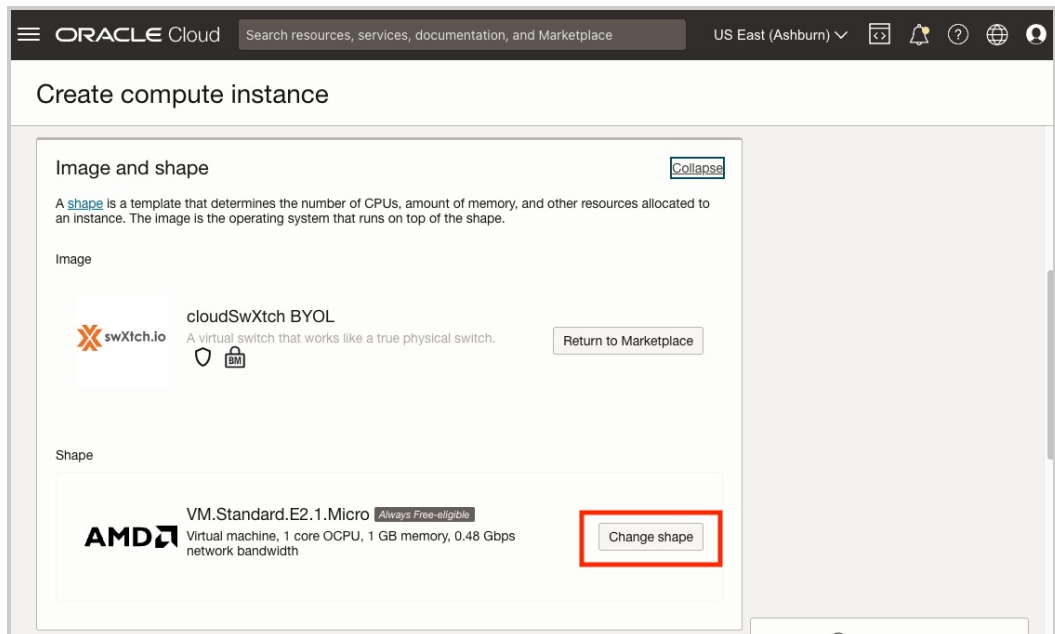
4. Select the **Edit** button for **Image and Shape**.



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 <small>i</small>	Memory (GB)	Security
<input checked="" type="checkbox"/> 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

©2023 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.

76

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
<input checked="" type="checkbox"/> 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	State	FQDN	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) ⓘ

☐ Skip source/destination check ⓘ

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

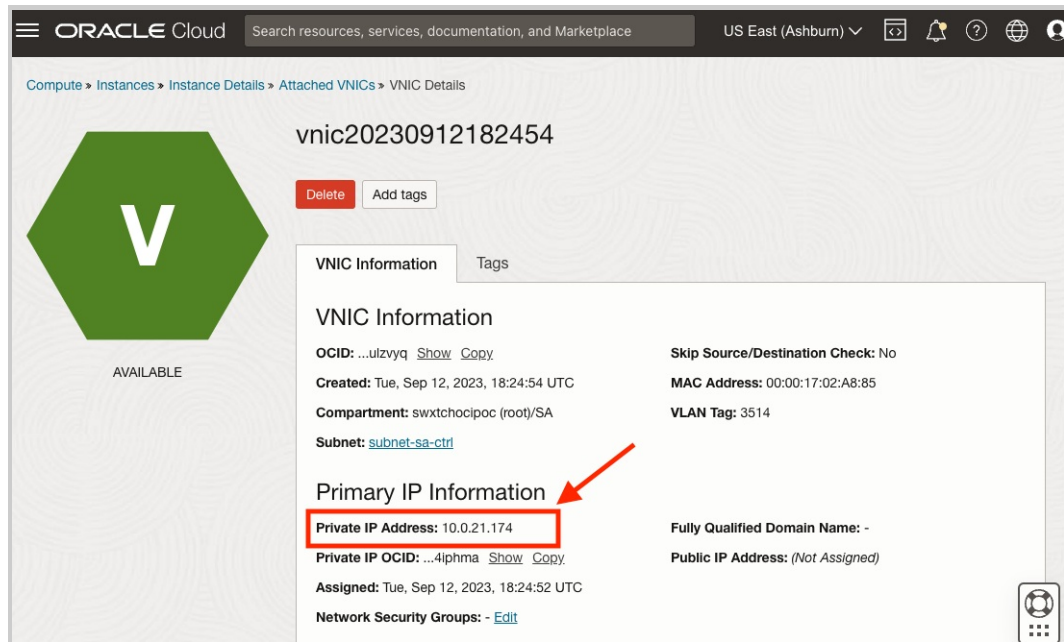
A [virtual network interface card \(VNIC\)](#) attaches an instance to a subnet within a VCN and is required for connectivity with other endpoints.

Create VNIC

Name	Subnet or VLAN	State	FQDN	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
<pre>network: version: 2 ethernet: ens4: match: macaddress: 02:00:17:09:c2:cf dhcp4: false addresses: - <ADD IP ADDRESS OF 2ND VNIC>/<XX></pre>	

1. 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
<pre>-A INPUT -p all -s 10.0.128.0/24 -j ACCEPT -A INPUT -p all -s 10.0.192.0/24 -j ACCEPT</pre>	

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.

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.
- 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 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 or IP.

Text

None	Copy
<pre>ping <swxtch-instance-ip></pre>	

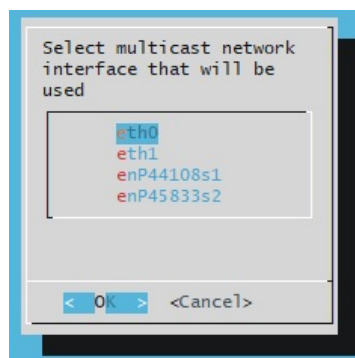
1. **If the ping fails to find the cloudSwXtch instance by name**, try pinging the IP address of the cloudSwXtch instance. If the IP works, they use the IP address in place of the name in all future commands. This can happen if the default DNS settings are changed for the Virtual Network.

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

Text

None	Copy
<pre>curl http://<swxtch-instance-ip>/services/install/swxtch-bridge-install.sh bash -s -- -k -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>	

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

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

START

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

RESTART

ActionScript	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>curl http://<swxtch-instance-name>/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

None	Copy
<pre>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

None	Copy
<pre>curl http://< swxtch-instance-name>/services/install/swxtch-bridge-install.sh bash</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](#)

xNIC software

The xNIC software must be run on each virtual machine that is to be part of the IP multicast network. This software can be installed on hosts which meet the following requirements:

- **Operating System:**
 - *Version 1:* RHEL 7+, CentOS 7+, or Ubuntu 18.04 | 20.04. Windows 10, Windows 11, Windows Server 2016+
 - *Version 2:* RHEL 8, CentOS 8, or Ubuntu 20.04, Windows 10, Windows 11, Windows Server 2019+
- **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. 2NICs

A 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:
 - 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).

Subnet Selection

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

The install is a simple command that installs 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. Version 1 of the xNIC software will create a tunnel network interface (called swtch-tun) 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. Version 2 does not create a tunnel interface and runs transparently in the kernel.

NOTE:

The swtch tunnel interface (swtch-tun) 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 cloudSwXtch traffic. For xNIC1, this creates a virtual network interface within the VM's operating system. In contrast, a virtual network interface will not appear for xNIC2 since users will only see their usual interfaces. Applications that use IP multicast should target this virtual network interface.

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](#).

Choosing an xNIC Version

There are two different xNIC versions for Linux: xNIC1 and xNIC2. xNIC2 is the default version. However, it is not available for all Linux versions. In those cases, the installer script will automatically download Version 1. swXtch.io's xNIC requires an OS from this list:

- Version 2: RHEL 8, CentOS 8, or Ubuntu 20.04.
- Version 1: RHEL 7+, CentOS 7+, or Ubuntu 18.04 | 20.04.

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 terminal on the host VM. The host VM is the VM in which you wish to install the xNIC software.
2. **Verify** network connectivity to the cloudSwXtch instance by "pinging" the switch.

None

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. If you would like to use **xNIC Version 2 (RHEL 8, CentOS8, or Ubuntu 20.04)**, run the following installer script:

None

Copy

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

4. If you would like to use **xNIC Version 1 (RHEL +7, CentOS 7+, or Ubuntu 18.04 | 20.04)**, run the following installer script:

Bash

Copy

```
curl http://<swxtch-instance-name>/services/install/swxtch-xnic-install.sh | bash -s -
- -v 1
```

Note: If you are using CentOS7 or Ubuntu 18.04 and try the xNIC Version 2 command, the installer script will detect the system and download xNIC Version 1.

Additional Arguments

In addition to the `-v` argument, there are additional options when installing the xNIC.

Note that the `ctrl-` and `data-` interfaces are from the VM the xNIC is installed. 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.

For xNIC1 on Linux, multiple xNICs can be installed on one VM by using the `-i` and the `--tun-subnet` arguments. However, in this case, the control and data interface must be different for each xNIC on the Linux VM for each cloudSwXtch.

A full list of arguments is detailed below:

Bash

Copy

```

$ ./swxtch-xnic-install.sh -h
Usage: ./swxtch-xnic-install.sh [OPTIONS]
  -v <1|2>                xNIC version to install (default: 1)
  -k                      kill existing swXtch processes and reinstall
package (default: no)
  -u                      uninstall xNIC instances, use -i to remove only
a specific instance
  -i <instance>           xNIC instance number (up to 5), valid only for
xNIC version 1
  --tun-subnet <ip>       virtual tun subnet IP. It is mandatory when
using -i.
  --ctrl_interface <interface name> manual selection of the Control interface
  --data_interface <interface name> manual selection of the Data interface
  -h | --help            shows this help

```

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

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

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, their xNIC has to be upgraded 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:

V2

None	Copy
<pre>curl http://<swxtch-instance-name>/services/install/swxtch-xnic-install.sh bash -s -- -k</pre>	

V1

None	Copy
<pre>curl http://<swxtch-instance-name>/services/install/swxtch-xnic-install.sh bash -s - - -k -v 1</pre>	

Additional Arguments

In addition to the -v argument, there are additional options when installing the xNIC.

Note that the ctrl- and data- interfaces are from the VM the xNIC is installed. 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:

None	Copy
<pre>\$./swxtch-xnic-install.sh -h</pre>	

```
Usage: ./swxtch-xnic-install.sh [OPTIONS]
  -v <2|1>                xNIC version to install (default: 2)
  -k                      kill existing swXtch processes and reinstall
package (default: no)
  -u                      uninstall xNIC instances, use -i to remove only
a specific instance
  -i <instance>           xNIC instance number (up to 5), valid only for
xNIC version 1
  --tun-subnet <ip>       virtual tun subnet IP. It is mandatory when
using -i.
  --ctrl_interface <interface name> manual selection of the Control interface
  --data_interface <interface name> manual selection of the Data interface
  -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 cloudSwXtch traffic. This creates a virtual network interface within the VM's operating system. Applications that use IP multicast should target this virtual network interface.

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](#).

Choosing an xNIC version

There are two different xNIC versions for Windows: xNIC1 and xNIC2. xNIC2 is the preferred method since it is the XDP (eXpress Data Path) version, which means it is a high performing and easily programmable. **Both versions are supported in Windows 10, Windows 11 and Server 2016+.**

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.

None	Copy
<pre>ping <switch-instance-name></pre>	

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. 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. However, if that is not possible, it is important to remove restrictions to UDP Ports 10800 and 9999 as mentioned above. 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. Download the installer script:

None

Copy

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

5. Run the installer script. **Note:** The installer script will differ depending on the version you decided to use.

None

Copy

```
V1
./swxtch-xnic-win-install.ps1
V2
./swxtch-xnic-win-install.ps1 -v 2
```

In addition to the `-v` argument, there are additional options when installing the xNIC. To see these options, use the `-h` argument. Note that the `ctrl-` and `data-` interfaces are from the VM the xNIC is installed. 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.

Bash

Copy

```
PS C:\Users\testadmin> ./swxtch-xnic-win-install.ps1 -h
This PowerShell script manages the xNIC installation for Windows
Usage: swxtch-xnic-win-install.ps1 [OPTIONS]
  -v [1|2]                xNIC version to install (default: 1)
  -u                      uninstall 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
-h                                     shows this help
```

Please note: The `ctrl_interface` and `data_interface` commands should only be used in complex configurations where the installer cannot locate them. Contact support@swXtch.io for more information.

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

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

Bash

[Copy](#)

```
get-netipconfiguration
```

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

Bash

[Copy](#)

```
PS C:\Users\testadmin> get-netipconfiguration

InterfaceAlias      : Ethernet
InterfaceIndex      : 36 ***
InterfaceDescription : Microsoft Hyper-V Network Adapter
NetProfile.Name     : Network
IPv4Address         : 10.2.128.75
IPv6DefaultGateway  :
IPv4DefaultGateway  : 10.2.128.1
DNSServer           : 168.63.129.16

InterfaceAlias      : Ethernet 2
InterfaceIndex      : 60 ***
InterfaceDescription : Microsoft Hyper-V Network Adapter #2
NetProfile.Name     : Unidentified network
IPv4Address         : 10.2.192.82
IPv6DefaultGateway  :
IPv4DefaultGateway  :
DNSServer           : 168.63.129.16
```


6. The installer script will install a Windows service called `swXtchNICWindowsService`:

```
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>
```

7. 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 from the cloudSwXtch and the NIC's 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 from the cloudSwXtch instance.
- `swxtch-perf.exe` : An application to produce and consume multicast traffic for testing purposes.

Running swxtch-top on Windows

swxtch-top dashboard --switch 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>	

Install xNIC on AKS Cilium

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 the Azure Kubernetes Service. Applications that use IP multicast should target this virtual network interface.

In this article, you will learn how to install xNIC2 on AKS Cilium.

The following operating systems in a pod are supported for the xNIC2 AKS installer: RHEL 8, CentOS 8 or Ubuntu 20.04.

Installation

The installation process can be split into three steps:

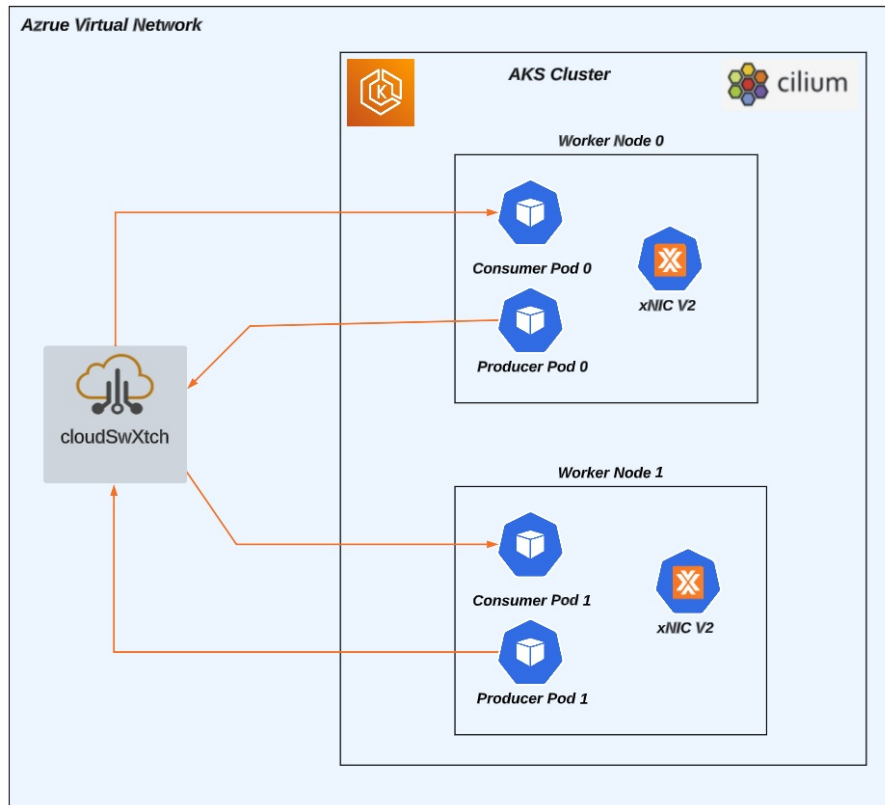
1. [Create an Azure Kubernetes Service with Cilium CNI](#)
2. [Install xNIC2 on AKS Cilium Native](#)
3. [Test xNIC2 with AKS](#)

Post-Installation

You can learn how to upgrade your xNIC nodes on AKS, [here](#).

xNIC Architecture Diagram

Below is an example of the architecture of 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 xNIC2.



Create an Azure Kubernetes Service with Cilium

WHAT TO EXPECT

In order to have an operational xNICv2 DaemonSet, a managed cluster using Azure Kubernetes Service (AKS) with Cilium CNI must be deployed.

Note: The AKS service **cannot be installed** using the Azure Kubernetes Service console, as the only network types that it supports are "Kubernetes" and "Azure CNI." The xNIC must be installed on AKS Cilium as the network.

In this article, you will learn the following:

- [How to Create an Azure Kubernetes Service \(AKS\)](#)
- [How to Install Cilium on AKS](#)

Pre-Creation Steps

If an AKS Cilium does not already exist, then the following steps can be used to create one. If it does exist but Cilium is not installed, then go to the section, [Cilium on AKS Installation](#).

Prior to running the script, ensure the following are already created in Azure:

- **Azure Resource Group** (must be same as the one used to create the cloudSwXtch)
- **Azure VNET** (must be same as the one used to create the cloudSwXtch)
- **Azure Control Subnet** (must be same as the one used to create the cloudSwXtch)

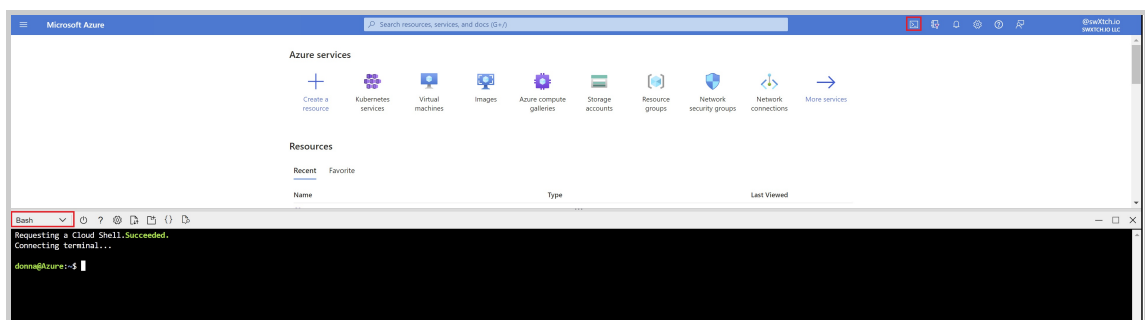
Each of these components **must match** the one used to create the cloudSwXtch on Azure.

Port 80 must be open for outbound traffic on the cluster.

AKS Creation

After verifying their match, complete these steps:

1. Log into Azure portal.
2. Open cloudShell as bash.



3. Run the following script to get security data:

ActionScript

ActionScript

Copy

```
az network vnet subnet show -g "Azure_Resource_Group" -n "Azure_subnet_ctr1" --  
vnet-name "Azure-VNET" --query id --output tsv
```

4. Copy the entire output for the next command:

Shell

Bash	Copy
<pre>az aks create \ --generate-ssh-keys \ --resource-group "Azure_Resource_Group" \ --network-plugin none \ --name cilium-sample \ --vnet-subnet-id "entire ouptput from above command" \ --node-vm-size Standard_Ds2_v2 \ --node-count 2</pre>	

1. The following will have to be replaced in the script:

- **The Resource Group:** "Azure_Resource_Group" - Change to the Resource group desired. This must be the same resource group as you cloudSwXtch.
- **The Name:** NAME="cilium-sample"- Change to an appropriate name for your AKS.
- **The vnet-subnet-id:** Will be the entire output of the previous command.

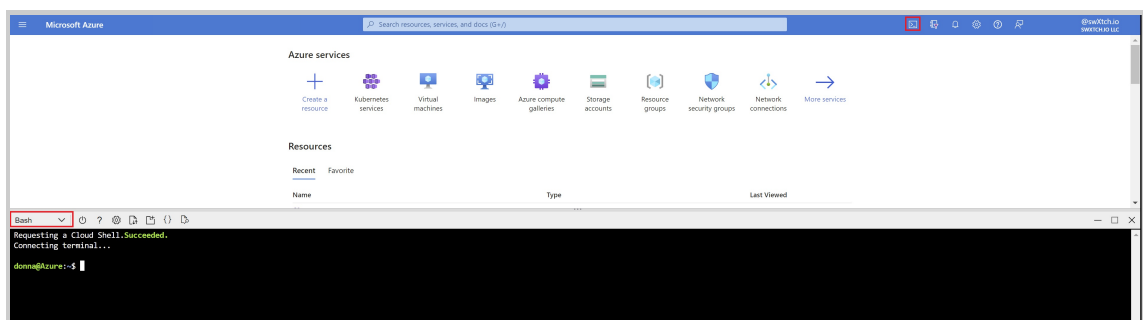
5. Edit the script with your desired values. Below is an example with filled in data:

ActionScript

ActionScript	Copy
<pre>az aks create \ --generate-ssh-keys \ --resource-group MyNetwork \ --network-plugin none \ --name cilium-sample2 \ --vnet-subnet-id /subscriptions/b10209ad-ad22-4c27-9aef-be93b2f0bb58/resourceGroups/MyNetwork/providers/Microsoft.Network/virtualNetworks/my-vnet/subnets/my-subnet-ctrl \ --node-vm-size Standard_Ds2_v2 \ --node-count 2</pre>	

1. This script will create 2 nodes. If more is desired, you can change it by increasing the node count.

6. Open cloudShell as bash.

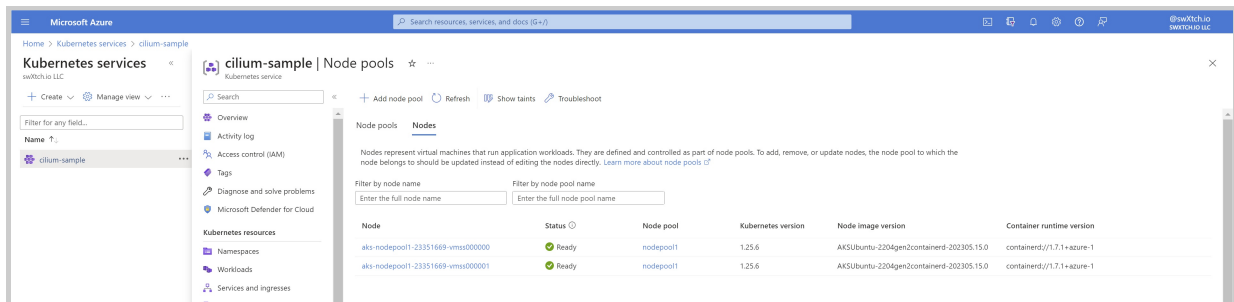


7. Paste in the script and hit Enter.

8. Wait for the script to complete.

1. You may get the following warning: "Could not create a role assignment for subnet. Are you an Owner on this subscription?" This is nothing to be concerned about.

The cluster is created with 1 node pool containing 2 nodes as shown below:



To ensure that credentials are properly set, run the following command where "Your-AKS-Name" is the name given in Step 4a:

Bash	Copy
<pre>az aks get-credentials --resource-group "Azure_Resource_Group" --name "Your-AKS-Name"</pre>	

To validate, run the following command:

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

Here is an example output:

Bash	Copy		
<pre>donna@Azure:~\$ kubectl config get-contexts</pre>			
CURRENT	NAME	CLUSTER	AUTHINFO
NAMESPACE			
*	cilium-sample-200	cilium-sample-200	clusterUser_test-donna-200-
	rg_cilium-sample-200		
	cilium-sample2	cilium-sample2	clusterUser_saDevNetwork_cilium-
	sample2		
	dsd-k8-cluster-100	dsd-k8-cluster-100	clusterUser_saDevNetwork_dsd-k8-
	cluster-100		

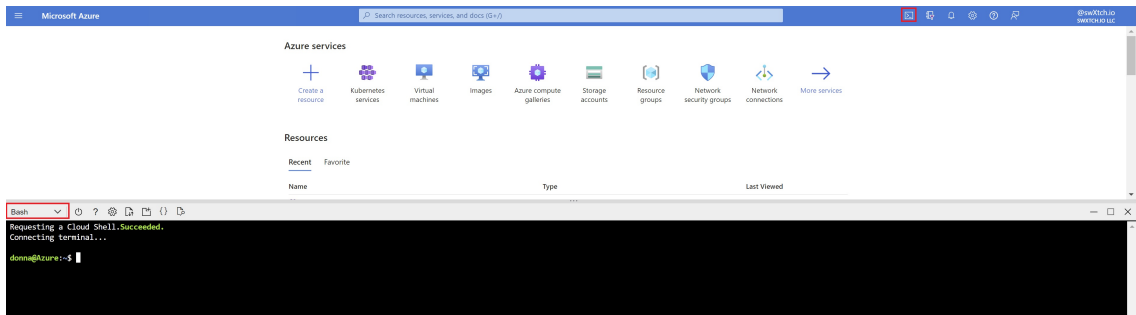
Cilium on AKS Installation

The section above created the AKS without a network. Cilium will provide network connectivity to the whole cluster. This section will describe how to install the Cilium CNI on an existing AKS without a network. Please refer to the [Cilium Official Documentation](#) or run the shell script provided next to install Cilium CLI:

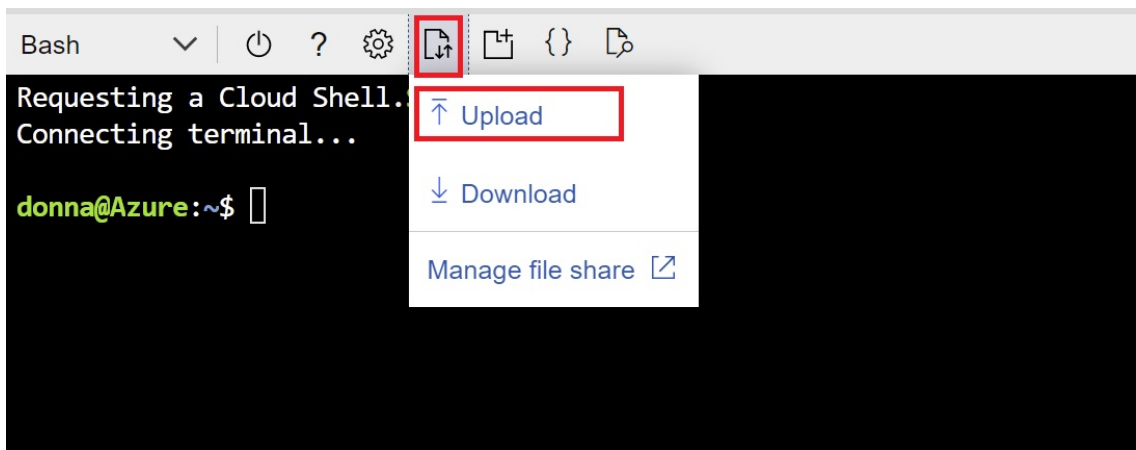
Bash	Copy
<pre>CILIUM_CLI_VERSION=\$(curl -s https://raw.githubusercontent.com/cilium/cilium- cli/master/stable.txt) CLI_ARCH=amd64 if ["\$(uname -m)" = "aarch64"]; then CLI_ARCH=arm64; fi</pre>	

```
curl -L --fail --remote-name-all https://github.com/cilium/cilium-
cli/releases/download/${CILIUM_CLI_VERSION}/cilium-linux-
${CLI_ARCH}.tar.gz{,.sha256sum}
sha256sum -c cilium-linux-${CLI_ARCH}.tar.gz.sha256sum
mkdir -p ~/bin
tar xzvf cilium-linux-${CLI_ARCH}.tar.gz -C /usr/local/bin -C ~/bin
rm cilium-linux-${CLI_ARCH}.tar.gz{,.sha256sum}
```

1. Copy the code above into a **Text Editor** and name the file **cilium-install.sh**.
2. Open **CloudShell** as **Bash**.



3. Upload the edited file using the **Bash Upload** feature.



4. Run the following command to change the script to an executable in the Azure Bash window:
Shell

Bash	Copy
<code>chmod +x cilium-install.sh</code>	

5. Run the following: command to run the executable in the Azure Bash window:
Shell

Bash	Copy
<code>./cilium-install.sh</code>	

- Run the script below, replacing the azure-resource group with the resource group used to create the AKS and that the cloudSwXtch was deployed with. This will install Cilium as the CNI for the AKS cluster.

Shell

Bash	Copy
<code>cilium install --azure-resource-group "your resource group"</code>	

- Validate that it was been successfully installed by running this command:

Shell

Bash	Copy
<code>cilium status</code>	

- An example of the validation is illustrated below:

Shell

Bash	Copy
<pre>\$ cilium status /--\ /--\ /--\ Cilium: OK \--\ \--\ Operator: OK /--\ /--\ Hubble Relay: OK \--\ \--\ ClusterMesh: disabled \--\ Deployment hubble-relay Desired: 1, Ready: 1/1, Available: 1/1 Deployment cilium-operator Desired: 1, Ready: 1/1, Available: 1/1 DaemonSet cilium Desired: 2, Ready: 2/2, Available: 2/2 Containers: cilium Running: 2 hubble-relay Running: 1 cilium-operator Running: 1 Cluster Pods: 10/10 managed by Cilium Image versions cilium quay.io/cilium/cilium:v1.13.1@sha256:428a09552707cc90228b7ff48c6e7a33dc0a 97fe1dd93311ca672834be25beda: 2 hubble-relay quay.io/cilium/hubble- relay:v1.13.1@sha256:ad7ce650c7877f8d769264e20bf5b9020ea778a9530cfae9d67a 5c9d942c04cb: 1 cilium-operator quay.io/cilium/operator- generic:v1.13.1@sha256:f47ba86042e11b11b1a1e3c8c34768a171c6d8316a3856253f 4ad4a92615d555: 1 \$</pre>	

Note: The Hubble Relay and ClusterMesh may not be enabled. This does not impact xNIC operation.

NEXT STEPS

The prerequisites have been completed. You can now continue onto the next step, [Install xNIC2 on AKS Cilium](#).

Install xNIC2 on AKS Cilium Native

WHAT TO EXPECT

xNIC2 is a lightweight service that must be installed on every AKS Cilium 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 xNIC2 on their Azure Kubernetes Service with Cilium CNI.

Overview

Unicast traffic will not be affected by this feature, it will work as it did before, the xNIC will only be used for Multicast traffic. The interface xNIC will use is swxbr0.

Running the Install Script

BEFORE YOU START

If you haven't already, please [Create an Azure Kubernetes Service with Cilium CNI](#) deployed. This is a prerequisite before installing xNIC2.

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 show be reinstalled if the cloudSwXtch version changes. This process takes less than a minute to install on an existing AKS cluster.

To run the install:

1. Ensure your cloudSwXtch is version 2.0.34. If it is not upgraded, see [Upgrade cloudSwXtch on Azure](#).
2. Copy the content below into a Text Editor, name the file `xnic_k8s_installer.sh` and Save.

Plaintext	Copy
<pre># This is the main installer of xNICv2 for both Air-Gapped and Open Networks # Supported Clouds: Azure, AWS and GCP. # Supported CNI plugins: Cilium, Amazon VPC, Calico and Azure Kubenet. #!/usr/bin/env bash set +e YELLOW='\033[1;33m' DEFAULT='\033[0m' function show_usage { echo -e "\nUsage: \$0 [OPTIONS]" echo -e "\nOPTIONS:" echo " -cni, --cni CNI plugin to be used [cilium, calico, aws, kubenet, gke, oci] (Default: auto)" echo " -ag, --airgap airgap installation (Default: false)" echo " -s, --switch <string> swXtch IP address" echo " -p, --port <string> docker Container Registry Port Number (Default: 5000)" echo " -u, --uninstall removes xNIC DaemonSet and all its configuration" echo " -h, --help shows this help"</pre>	

```

    echo ""
}

function parse_arguments() {
    CNI="AUTO"
    ALLOWED_CNIS="CILIUM CALICO AWS KUBENET GKE OCI"
    AIRGAP="false"
    XNIC_SWXTCH_PORT=5000
    if [ $# -eq 0 ]; then
        show_usage
        exit 1
    fi
    while [[ $# -gt 0 ]]
    do
        key="$1"
        case $key in
            -cni| --cni)
                CNI="$2"
                shift # shifts argument
                shift # shifts value
                ;;
            -ag| --airgap)
                AIRGAP="true"
                shift # shifts argument
                ;;
            -s| --switch)
                XNIC_SWXTCH_ADDR="$2"
                shift # shifts argument
                shift # shifts value
                ;;
            -p| --port)
                XNIC_SWXTCH_PORT="$2"
                shift # shifts argument
                shift # shifts value
                ;;
            -u| --uninstall)
                UNINSTALL="true"
                shift # shifts argument
                ;;
            -h| --help | *)
                show_usage
                exit 1
                ;;
        esac
    done
    CNI=${CNI^^}

    # Detect the Cloud // Default: AMAZON
    CLOUD="GENERIC"
    PROVIDERS="GOOGLE AMAZON AZURE ORACLE"
    ORACLE="oke.oraclecloud.com"
    GOOGLE="cloud.google.com"
    AMAZON="eks.amazonaws.com"

```

```

AZURE="kubernetes.azure.com"
for p in $PROVIDERS; do
    if [[ -n $(kubectl get no --show-labels | grep ${!p}) ]]; then
        CLOUD=$p
    fi
done
echo -e "\n[i] Detected Cloud: $CLOUD"

# Start uninstaller
if [[ $UNINSTALL == "true" ]]; then
    # Check if installer used Cilium
    USED_CNI=$(kubectl get -n kube-system cm xnic-config -o jsonpath={.data.CNI}
2>&1)
    if [[ $USED_CNI == "CILIUM" ]]; then
        echo "[i] Installed CNI: Cilium"
        echo ""
        echo "=====
        echo "           Restoring Cilium old configuration "
        echo "=====
        # Remove Cilium BPF flag and restart Cilium Agents
        echo -e "Deleting flag \"bpf-filter-priority\" from Cilium ConfigMap"
        kubectl patch -n kube-system cm/cilium-config \
        --patch ['{"op": "remove", "path": "/data/bpf-filter-priority" }'] \
        --type json
        kubectl rollout restart ds -l app.kubernetes.io/part-of=cilium -n kube-
system
    else
        echo "[i] Installed CNI: $USED_CNI"
        echo "[i] Proceeding with xNICv2 removal"
    fi
    # Continue removing xNICv2
    echo ""
    echo "=====
    echo "           Uninstalling xNICv2"
    echo "=====
    # Remove xNIC ConfigMap & DaemonSet K8S resources
    kubectl delete -n kube-system cm/xnic-config
    kubectl delete -n kube-system ds/swxtch-xnic
    echo "Done!"
    echo -e "\n===== Completed! ====="
    echo ""
    exit 0
fi
# End of uninstaller

# Continue parsing
if [ -z ${XNIC_SWXTCH_ADDR} ]; then
    echo -e "\nPlease specify the swXtch IP address.\n"
    exit 1
fi

if [[ $CNI == "AUTO" ]]; then
    kubectl get cm -n kube-system cilium-config > /dev/null 2>&1

```

```

if [[ $? == 0 ]]; then
    ## Cilium has been previously installed on the Cluster
    echo "[i] Cilium Installation detected"
    echo "[i] Setting CNI to CILIUM..."
    CNI="CILIUM"
else
    ## Any other CNI is present on the Cluster
    if [[ $CLOUD == "GENERIC" ]]; then
        echo "[i] Using Generic CNI (P2P)..."
        CNI="GENERIC"
    fi
    if [[ $CLOUD == "ORACLE" ]]; then
        echo "[i] Setting CNI to OCI VCN-Native..."
        CNI="OCI"
    fi
    if [[ $CLOUD == "GOOGLE" ]]; then
        kubect1 get ds -n kube-system calico-node > /dev/null 2>&1
        if [[ $? == 0 ]]; then
            echo "[i] Calico Installation detected"
            echo "[i] Setting CNI to CALICO..."
            CNI="CALICO"
        else
            echo "[i] Setting CNI to GKE Traditional..."
            CNI="GKE"
        fi
    fi
    if [[ $CLOUD == "AMAZON" ]]; then
        echo "[i] Setting CNI to AWS..."
        CNI="AWS"
    fi
    if [[ $CLOUD == "AZURE" ]]; then
        echo "[i] Setting CNI to KUBENET..."
        CNI="KUBENET"
    fi
fi
else
    for cni in $ALLOWED_CNIS; do
        if [[ $CNI == $cni ]]; then
            return
        fi
    done
    echo "Invalid CNI - Aborting"
    exit 1
fi
sleep 1
# End of parsing function
}

# Parse script arguments
parse_arguments "$@"

if [[ $CNI == "CILIUM" ]]; then
    echo ""

```

```

echo -e
"#####
#####"
echo -e "          This script modifies the underlying configuration of Cilium CNI
to make it compatible"
echo -e "          with Multicast Networks."
echo -e "          It also installs xNICv2 DaemonSet on the existing cluster. "
echo -e
"#####
#####"
echo ""
fi

sleep 2

if [[ $AIRGAP == "true" ]]; then
    echo ""
    echo -e "##### NOTICE - Airgap Installation
#####"
    echo -e "          Please ensure you have previously patched ALL the nodes with
the patch_containerd.sh script. "
    echo -e "          If you haven't done it, uninstall xNICv2 and restart the
process with the nodes patched."
    echo -e
"#####
#####"
    echo ""
    sleep 4
    echo -e "\n- RUNNING INSTALLER: Airgap"
    echo -e "\n- CONTAINER REGISTRY PORT: $XNIC_SWXTCH_PORT"
    IMAGE=${XNIC_SWXTCH_ADDR}:${XNIC_SWXTCH_PORT}/xnicv2:airgap
else

    echo -e "\n- RUNNING INSTALLER: Open Network with Internet Access"
    IMAGE='ubuntu:20.04'
    CMD='["/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;
sleep infinity"]'

fi

echo -e "\n- IMAGE: $IMAGE"

if [[ $CNI == "KUBENET" ]]; then
    SWX_K8S_BRIDGE_IF="cbr0"
else
    SWX_K8S_BRIDGE_IF="swxbr0-${CNI,,}"
fi

```

```

echo -e "\n- CNI PLUGIN: $CNI"
echo -e "\n- SWXTCH IP ADDRESS: $XNIC_SWXTCH_ADDR"

sleep 2

if [[ $CNI == "CILIUM" ]]; then
    echo ""
    echo "=====
    echo "          Adjusting BPF filter priority on Cilium "
    echo "=====
    echo -e "Setting flag \"bpf-filter-priority\" to \"50000\" "
    kubectl patch -n kube-system cm/cilium-config --patch '{"data": {"bpf-
filter-priority": "50000"}}'
    echo "Done!"

    echo ""
    echo "=====
    echo "          Restarting Cilium Agents"
    echo "=====
    kubectl rollout restart ds -l app.kubernetes.io/part-of=cilium -n kube-
system
    echo -n "Waiting for Cilium Agents to be fully UP and Running..."
    sleep 2
    while [[ -z $(kubectl get po -n kube-system -l
app.kubernetes.io/name=cilium-agent --field-selector status.phase=="Running" -o
NAME | head -1) ]]
    do
        echo -n "."
        sleep 2
    done
    echo "OK"
    echo "Done!"
    echo "Proceeding with xNICv2 Installation"

fi

sleep 2

echo ""
echo "=====
echo "          Creating xNIC ConfigMap"
echo "=====
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: xnic-config
  namespace: kube-system
data:
  XNIC_SWXTCH_ADDR: ${XNIC_SWXTCH_ADDR}
  XNIC_SWXTCH_PORT: "${XNIC_SWXTCH_PORT}"

```



```

IS_DAEMON: "true"
CNI: ${CNI}
SWX_K8S_BRIDGE_IF: ${SWX_K8S_BRIDGE_IF}
AIRGAP: "${AIRGAP}"
EOF

echo ""
echo "=====
echo "          Installing xNIC v2"
echo "=====
cat << EOF | kubectl apply -f -
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: swxtch-xnic
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: swxtch-xnic
  template:
    metadata:
      labels:
        app: swxtch-xnic
    spec:
      hostNetwork: true
      containers:
        - name: swxtch-xnic
          image: ${IMAGE}
          imagePullPolicy: Always
          securityContext:
            privileged: true
          envFrom:
            - configMapRef:
                name: xnic-config
          command: ${CMD}
          args: ${ARGS}
          lifecycle:
            preStop:
              exec:
                command: ["/bin/bash", "-c",
                          "/usr/bin/swxtch/tc-bpf-cleanup.sh;
                          ip link del dev ${SWX_K8S_BRIDGE_IF}"]
EOF

echo "Done!"

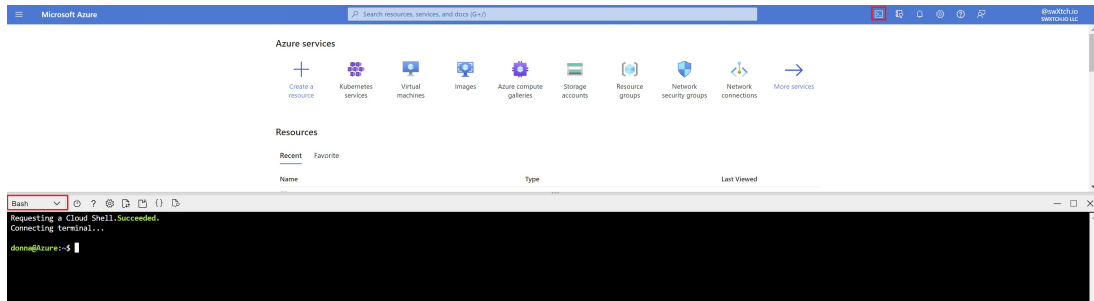
echo -e "\n===== Completed! ====="

echo -e "\nPlease allow a minute for the xNIC DaemonSet to fully spin up before
starting to use it."

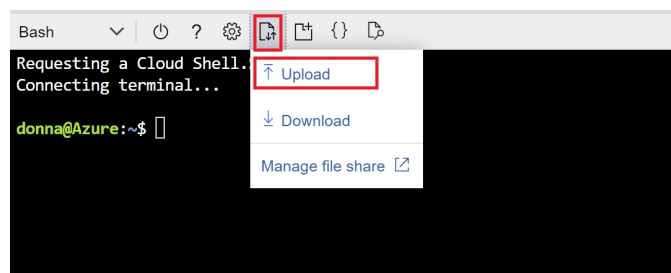
```

```
echo -e "Feel free to follow up on the xNIC Agents installation by running  
\n\n\t${YELLOW}kubectl logs -n kube-system daemonsets/swxtch-xnic -f \n"
```

3. Sign into Azure.
4. Open cloudShell as Bash.



5. Upload the **xnic_k8s_installer.sh** file using the Bash Upload feature.



6. Run the following command to change the script to an executable in the Azure Bash window:

Plaintext

Copy

```
chmod +x xnic_k8s_installer.sh
```

7. Run the following script replacing the **<cloudSwXtch-instance-IP>** with your cloudSwXtch control IP:

Plaintext

Copy

```
./xnic_k8s_installer.sh -s <cloudSwXtch-instance-ip>
```

1. A successful install is shown below:

Plaintext	Copy
<pre>user@Azure:~\$./xnic_k8s_installer.sh -s 10.2.128.10 ##### ##### This script modifies the underlying configuration of Cilium CNI to make it compatible with Multicast Networks. It also installs xNICv2 DaemonSet on the existing cluster. ##### ##### - RUNNING INSTALLER: Open Network with Internet Access - SWXTCH IP ADDRESS: 10.2.128.10 ===== Adjusting BPF filter priority on Cilium ===== Setting flag "bpf-filter-priority" to "50000" configmap/cilium-config configured 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 xNICv2 Installation ===== Installing xNIC v2 ===== daemonset.apps/swxtch-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 daemonsets/swxtch-xnic -f</pre>	

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

Plaintext	Copy
<pre>kubectl logs daemonsets/swxtch-xnic -f</pre>	

9. Use the command below to follow the AKS node status in the Bash window in Azure and check if they have started:

1. Example:

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

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

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

PowerShell	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:

PowerShell	Copy
<pre># DENY OUTGOING TO SPECIFIC GROUP(S)</pre>	

```
tc qdisc add dev eth0 root handle 1: prio
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst <multicast_group_0>
action drop
...
tc filter add dev eth0 parent 1: protocol ip u32 match ip dst <multicast_group_n>
action drop
```

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

Plaintext	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

Plaintext	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** :

Plaintext	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:

Plaintext	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:

--

Plaintext	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** :

Plaintext	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:

Plaintext	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** :

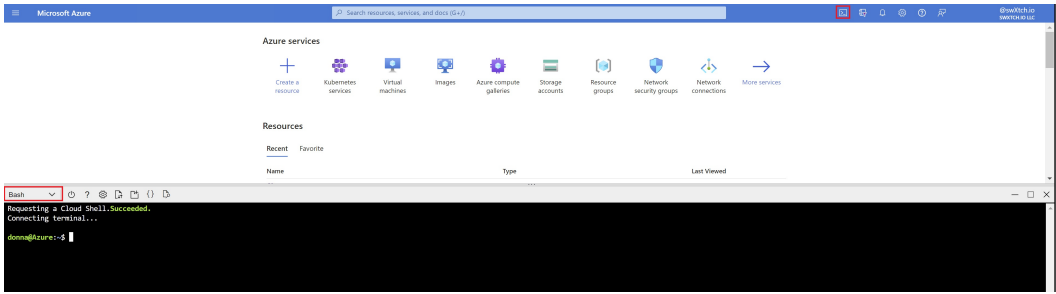
Plaintext	Copy
<pre>tc qdisc add dev eth0 ingress tc qdisc add dev eth0 parent ffff: protocol ip u32 match ip dst 224.0.0.0/4 action drop tc filter add dev eth0 parent ffff: protocol ip u32 match ip dst 239.0.0.1/32 action ok</pre>	

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 Azure.

2. Open cloudShell as Bash.



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

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

1. Example:

Plaintext	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>	

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

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

1. Example:

Plaintext	Copy
<pre>user@Azure:~\$ kubectl exec -it pod/swxtch-xnic-kn9hg -n kube-system -- bash root@aks-nodepool11-23164585-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 AKS Context in Azure](#) section.

Signing into AKS node instance in Azure Bash

If you want to sign into the AKS instance via the Azure Bash, complete the following:

1. Ensure the pods are running using this command:

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

1. Example:

Plaintext	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:

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

1. Example:

Plaintext	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 xNIC2.

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:

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

Exiting the Pod

To exit the pod, enter in the following command:

Plaintext	Copy
<pre>Exit</pre>	

To Change AKS Context in Azure

If there are more than one AKS instances in Azure then you may need to be able to switch between them to run commands in Azure Bash. Below are steps to switch between AKS instances.

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

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

1. Example:

Plaintext	Copy
<pre>user@Azure:~\$ kubectl config get-contexts</pre>	
<pre>CURRENT NAME CLUSTER AUTHINFO NAMESPACE</pre>	
<pre> cilium-sample cilium-sample clusterUser_saDevNetwork_cilium-sample</pre>	
<pre> cilium-sample-200 cilium-sample-200 clusterUser_test- donna-200-rg_cilium-sample-200</pre>	
<pre> cilium-sample2 cilium-sample2 clusterUser_saDevNetwork_cilium-sample2</pre>	
<pre>* cilium-sample300 cilium-sample300 clusterUser_test- donna-300-rg_cilium-sample300</pre>	
<pre> dsd-k8-cluster-100 dsd-k8-cluster-100 clusterUser_saDevNetwork_dsd-k8-cluster-100</pre>	

2. 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.

Plaintext	Copy
<pre>kubectl config use-context cilium-sample2</pre>	

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

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

1. Example:

Plaintext	Copy
<pre>user@Azure:~\$ kubectl config get-contexts</pre>	
<pre>CURRENT NAME CLUSTER AUTHINFO</pre>	
<pre>NAMESPACE</pre>	
<pre> cilium-sample cilium-sample</pre>	
<pre>clusterUser_saDevNetwork_cilium-sample</pre>	
<pre> cilium-sample-200 cilium-sample-200</pre>	
<pre>donna-200-rg_cilium-sample-200</pre>	
<pre>* cilium-sample2 cilium-sample2</pre>	
<pre>clusterUser_saDevNetwork_cilium-sample2</pre>	
<pre> cilium-sample300 cilium-sample300</pre>	
<pre>donna-300-rg_cilium-sample300</pre>	
<pre> dsd-k8-cluster-100 dsd-k8-cluster-100</pre>	
<pre>clusterUser_saDevNetwork_dsd-k8-cluster-100</pre>	

2. As you can see above, the asterisk (*) has changed positions to the desired context, cilium-sample2.

Uninstall xNIC2 on AKS

WHAT TO EXPECT

In this article, users will learn how to uninstall xNIC2 on Kubernetes (K8's).

To uninstall xNIC2 on AKS, please follow these steps:

- 1. Sign into Cloud.
- 2. Open cloudShell as Bash.
- 3. Run the following command in the terminal:

Shell

Bash	Copy
<pre>./ xnic_k8s_installer.sh -u</pre>	

- 4. xNIC2 on K8's should now be uninstalled.

Test xNIC2 with AKS

WHAT TO EXPECT

Before running your application in AKS, it is a good idea to test with swXtch.io provided tools/examples.

In this article, you will learn how to test xNIC2 with AKS. Please complete the installation process outlined in [Install xNIC2 on AKS Cilium](#) before you begin testing.

STEP ONE: Create A Consumer

1. Create a **TestConsumer.yaml** file using the example below.
 1. Replace the **XNIC_SWXTCH_ADDR** with the cloudSwXtch control address.

Shell

Bash	Copy
<pre>apiVersion: v1 kind: Pod metadata: name: consumer-a annotations: k8s.v1.cni.cncf.io/networks: eth0-bridge@swx0 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</pre>	

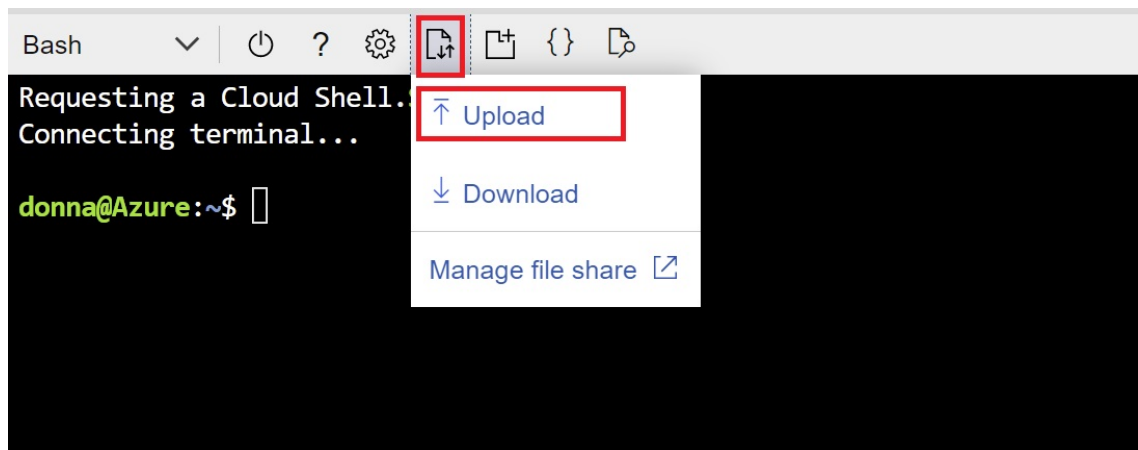
```

    value: "8410"
  - name: XNIC_SWXTCH_ADDR
    value: "xx.x.xxx.xx"
    command: ["/bin/bash"]
    args: ["-c", "apt update && apt install curl -y;
               curl
http://$(XNIC_SWXTCH_ADDR)/services/install/swxtch-xnic-install.sh --
output swxtch-xnic-install.sh;
               chmod +x swxtch-xnic-install.sh;
               ./swxtch-xnic-install.sh -v 2"]

initContainers:
- name: init-command-container
  securityContext:
    privileged: true
  image: ubuntu:20.04
  command: ["sh", "-c"]
  args: ["apt update && apt install iproute2 -y;
         tc qdisc add dev eth0 root handle 10: prio;
         tc filter add dev eth0 parent 10: protocol ip u32 match ip dst
224.0.0.0/4 action mirred egress redirect dev swx0;
         tc filter add dev eth0 parent 10: protocol ip u32 match ip
protocol 2 0xff action mirred egress redirect dev swx0;
         tc qdisc add dev swx0 ingress;
         tc filter add dev swx0 parent ffff: protocol ip u32 match ip
dst 224.0.0.0/4 action mirred ingress redirect dev eth0"]

```

2. Upload the file into the Azure CloudShell using the upload option.



STEP TWO: Create a Producer

1. Create a `TestProducer.yaml` file using the example below.
 1. Replace `XNIC_SWXTCH_ADDR` with the cloudSwXtch control address.

Bash	Copy
<pre> apiVersion: v1 kind: Pod metadata: name: producer-a annotations: k8s.v1.cni.cncf.io/networks: eth0-bridge@swx0 </pre>	

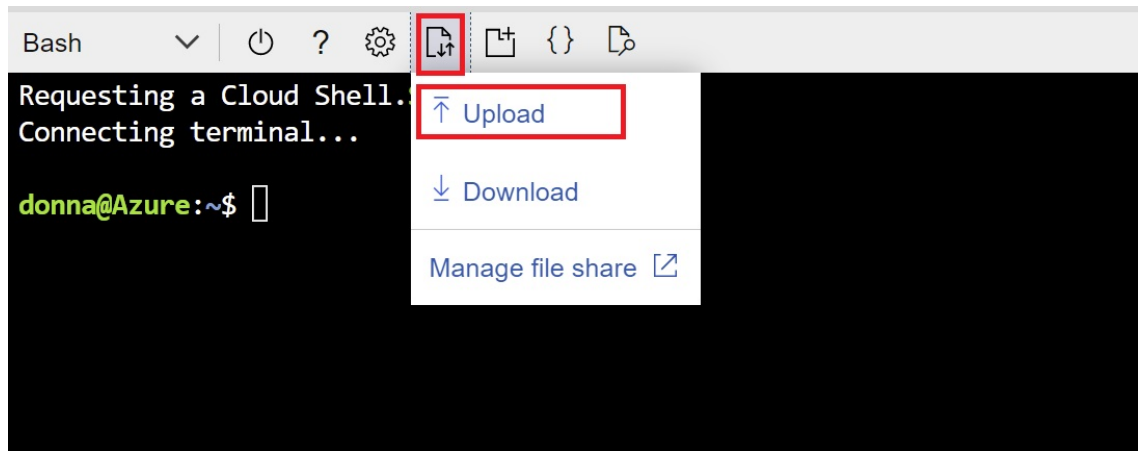
```

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: "xx.xx.xxx.xx"
      command: ["/bin/bash"]
      args: ["-c", "apt update && apt install curl -y;
                curl
                http://$(XNIC_SWXTCH_ADDR)/services/install/swxtch-xnic-install.sh --
                output swxtch-xnic-install.sh;
                chmod +x swxtch-xnic-install.sh;
                ./swxtch-xnic-install.sh -v 2"]
  initContainers:
    - name: init-command-container
      securityContext:
        privileged: true
      image: ubuntu:20.04
      command: ["sh", "-c"]
      args: ["apt update && apt install iproute2 -y;
                tc qdisc add dev eth0 root handle 10: prio;
                tc filter add dev eth0 parent 10: protocol ip u32 match ip dst
                224.0.0.0/4 action mirrored egress redirect dev swx0;
                tc filter add dev eth0 parent 10: protocol ip u32 match ip
                protocol 2 0xff action mirrored egress redirect dev swx0;"]

```

```
tc qdisc add dev swx0 ingress;  
tc filter add dev swx0 parent ffff: protocol ip u32 match ip  
dst 224.0.0.0/4 action mirred ingress redirect dev eth0"]
```

2. Upload the file into the Azure CloudShell using the upload option.



STEP THREE: Run Test

1. Run the producer by running this command in the Azure cloudShell bash window.
 1. Wait for the cursor to return to know it is fully created.

Shell

Bash	Copy
kubectl create -f TestProducer.yaml	

2. Run the consumer by running this command in the Azure cloudShell bash window.
 1. Wait for the cursor to return to know it is fully created.

Shell

Bash	Copy
kubectl create -f TestConsumer.yaml	

3. Validate they are running using this command:

Shell

Bash	Copy
kubect1 get pods -o wide	

1. Below is an example showing the consumer-a and producer-a running:

Shell

Bash	Copy
<pre>donna@Azure:~\$ kubect1 get pods -o wide NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES consumer-a 1/1 Running 0 17h 10.0.0.196 aks- nodepool1-23351669-vmss000003 <none> <none> producer-a 1/1 Running 0 17h 10.0.1.153 aks- nodepool1-23351669-vmss000002 <none> <none> swxtch-xnic-dwx2d 1/1 Running 0 17h 10.2.128.96 aks- nodepool1-23351669-vmss000002 <none> <none> swxtch-xnic-dzpf1 1/1 Running 0 17h 10.2.128.95 aks- nodepool1-23351669-vmss000003 <none> <none> donna@Azure:~\$</pre>	

2. You can also validate it is working by running logs with this command:

Shell

Bash	Copy
kubect1 logs pods/producer-a -f	

4. Log into your cloudSwXtch and run this command:

Shell

Bash	Copy
sudo /swxtch/swxtch-top dashboard --switch localhost	

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

5. Stop the test consumer by running this command back in the Azure CloudShell bash window.

1. Wait for the cursor to return to know it is deleted fully.

Shell

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:

Shell

Bash

Copy

```
donna@Azure:~$ kubectl get pods -o wide
```

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>	

6. Stop the test producer by running this command in the Azure CloudShell bash window.

1. Wait for the cursor to return to know its fully deleted.

Shell

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:

Shell

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

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

Upgrade xNIC nodes on AKS

WHAT TO EXPECT

The nodes upgrade is automatic based on the restart of the nodes containing the xNIC. This can be done in one of two ways: by [restarting the xNIC Daemonset](#) or by [restarting the AKS cluster in Azure](#).

In this article, you will learn how you can use either one of the methods to upgrade your xNIC nodes on AKS to match the version of your cloudSwXtch.

Before you upgrade the xNIC nodes on AKS, you need to upgrade the cloudSwXtch to the latest version. See Upgrade cloudSwXtch on Azure.

Restarting the xNIC Daemonset

1. Sign into Azure portal.
2. Open cloudShell as Bash.
3. Run the following command:

Shell

Bash

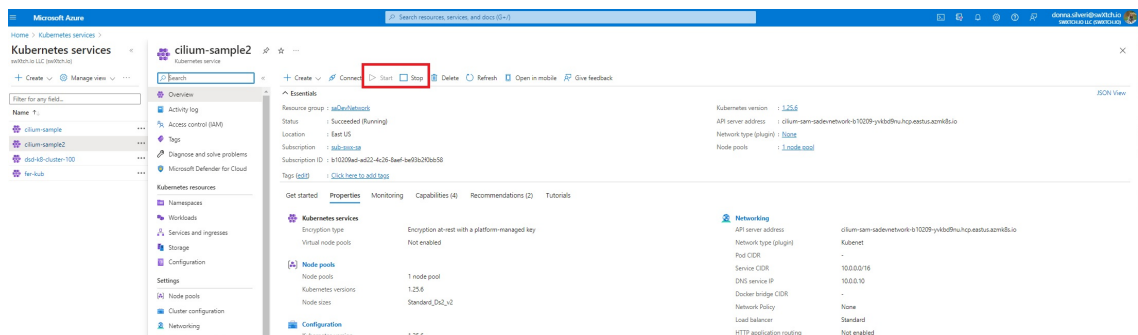
Copy

```
kubectl rollout restart ds swtch-xnic -n kube-system
```

1. This will restart the Kubernetes swtch-nic daemon set and update the version of the xNIC to match the cloudSwXtch.

Restarting the AKS Cluster in Azure

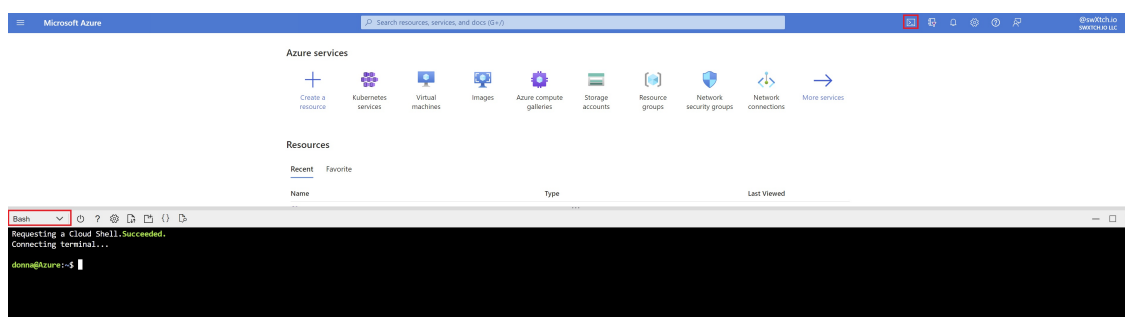
1. Sign into the Azure portal.
2. Search for **Kubernetes Services** and select the service where the xNIC is installed.
3. Select **Stop**.
4. Select **Start**.
5. Wait for the AKS and its nodes to start.



Validate the Nodes are running

1. Sign into Azure portal

2. Open cloudShell as Bash.



3. Validate xNIC's are running

To ensure xNIC is running run the following command:

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

Below is a sample output with the above commands:

Bash

Copy

user@Azure:~\$ kubectl get pods -l app=swxtch-xnic -n kube-system -n kube-system

NAME	READY	STATUS	RESTARTS	AGE
swxtch-xnic-fc58t	1/1	Running	0	11d
swxtch-xnic-qn9hg	1/1	Running	0	11d

In the above example, there are 2 pods. Once they are both set to **Running** again, they are ready for use. Note that it deletes and recreates the pods and therefore, the names, IP addresses and Node may be different prior to the restart.

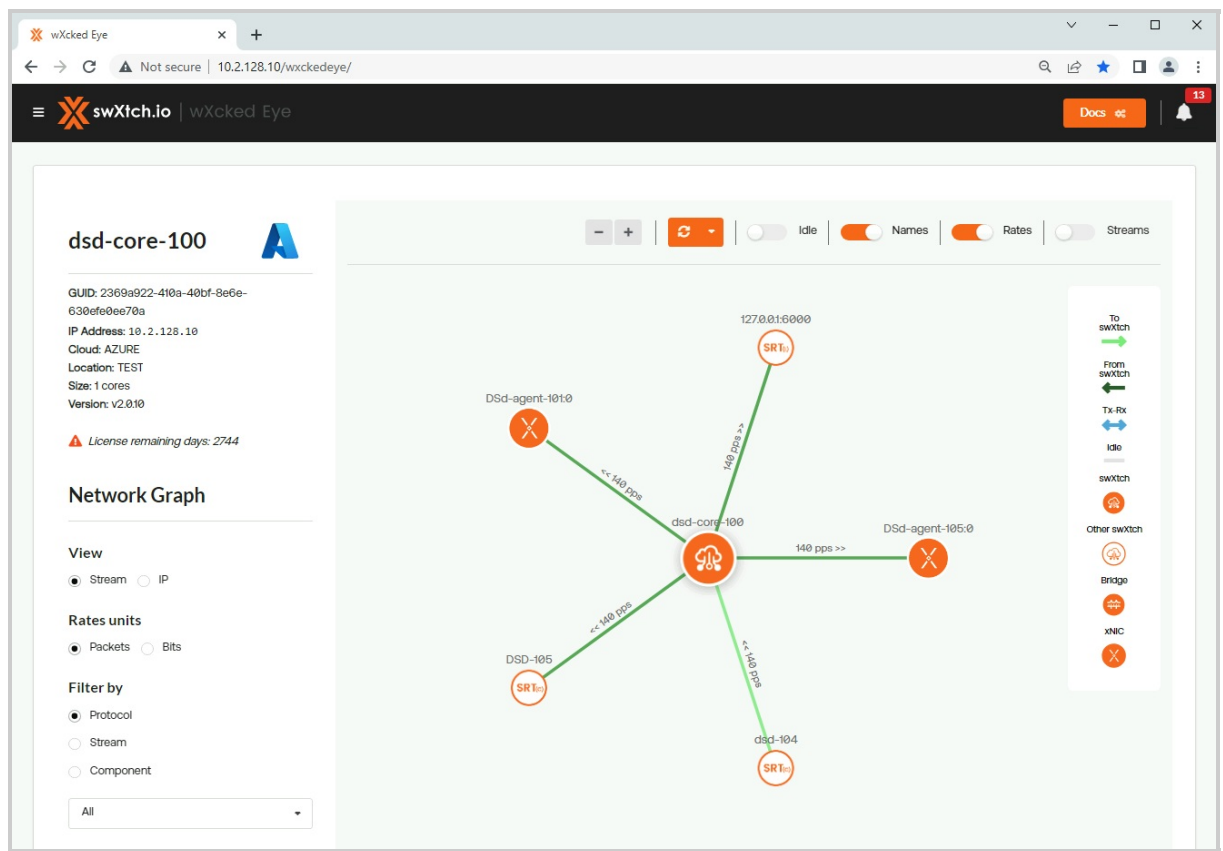
Using wXcked Eye for cloudSwXtch

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 Precision Time Protocol (PTP) from the comfort of a user's 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 on their VM. They should use the same IP address of their cloudSwXtch to prefix the URL.

None	Copy
<swxtch-ip-address>/wxckedeye/	

Monitor cloudSwXtch with wXcked Eye

To learn more about the monitoring capabilities of wXcked Eye such as the Network Graph or the cloudSwXtch and xNIC metrics views, see [Monitor cloudSwXtch with wXcked Eye](#).

Configure cloudSwXtch with wXcked Eye

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 with wXcked eye, please read the "[Configure cloudSwXtch with wXcked Eye](#)" article.

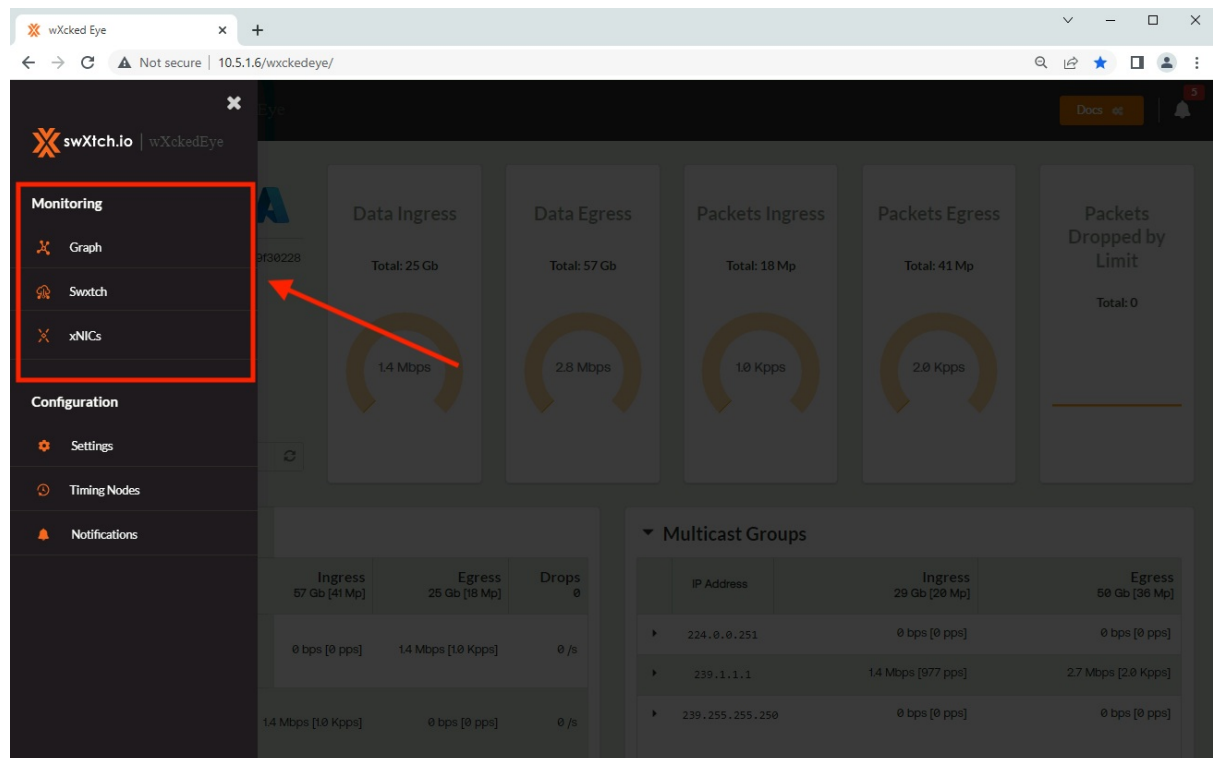
In this section, we will walk through the user interface, explaining overall functionality and how it provides users with additional control over their cloudSwXtch network.

Accessing the wXcked Eye UI

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

None	Copy
<code><swxtch-ip-address>/wxckedeye/</code>	

Navigating the Monitoring pages



The wXcked Eye monitoring tool is organized into three pages: the Network Graph, cloudSwXtch view and xNICs view. The Network graph is the default main page for wXcked Eye and will be the first thing users see when using the UI.

To learn more about the individual pages, you can find their respective articles here:

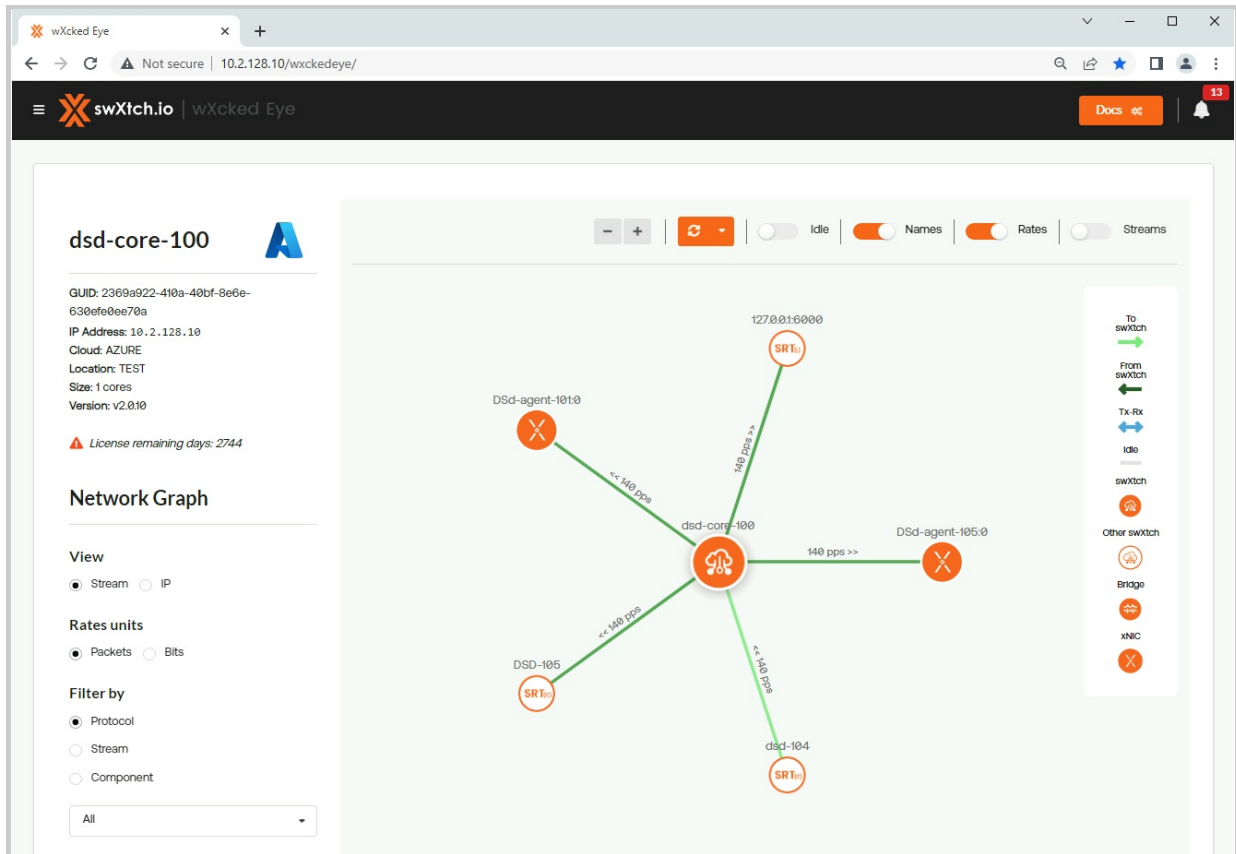
- [wXcked Eye Network Graph](#)
- [wXcked Eye cloudSwXtch view](#)
- [wXcked Eye xNICs view](#)

wXcked Eye Network Graph

WHAT TO EXPECT

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

Using the wXcked Eye Network Graph



The wXcked Eye main page displays a Network Graph to provide a high level view of the cloudSwXtch environment. On the top left hand 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 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 xNIC, UDP or SRT as detailed in the key on the right hand side of the graph.

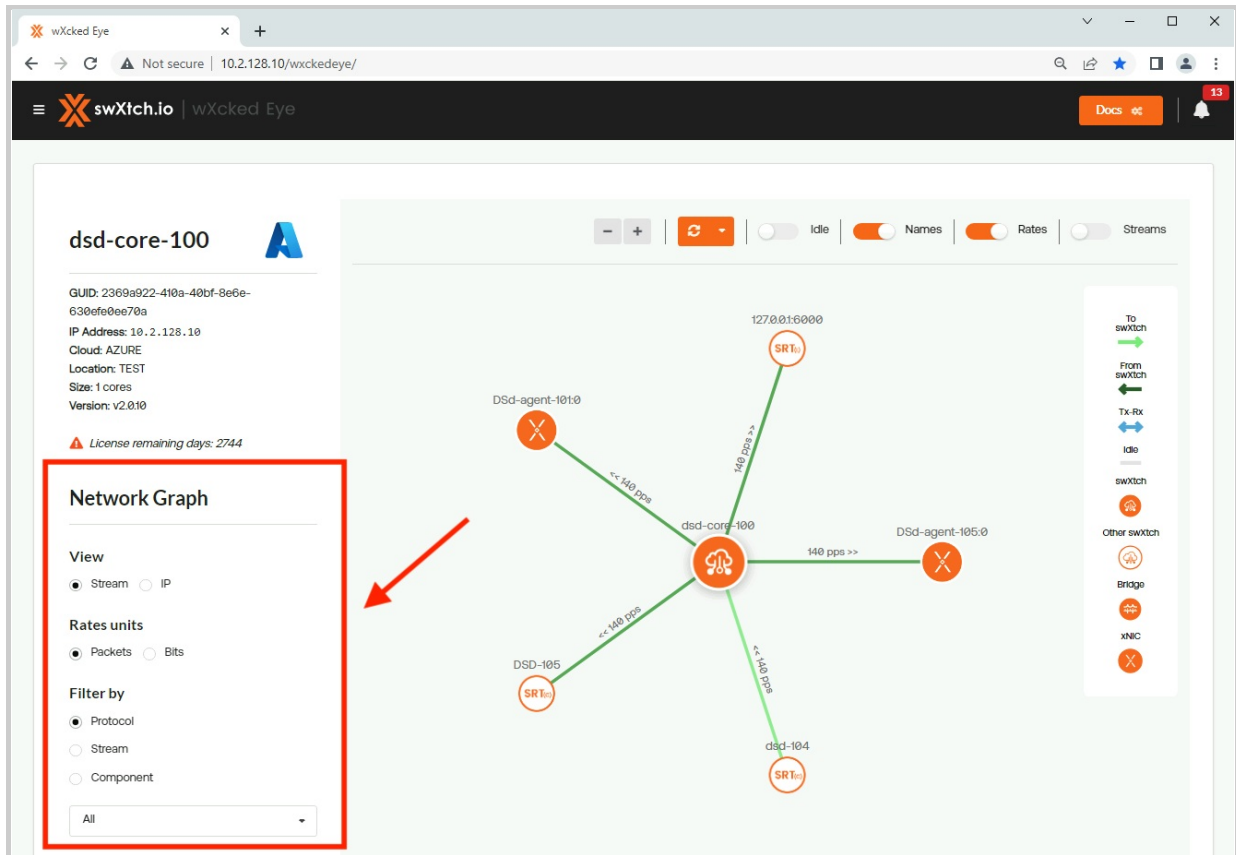
Reformatting the Graph

It is very simple to alter the Network Graph for a user's desired configuration. In addition to being able to physically drag and rearrange the icons in the graph, users can zoom in and out, refresh the page, and toggle 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.



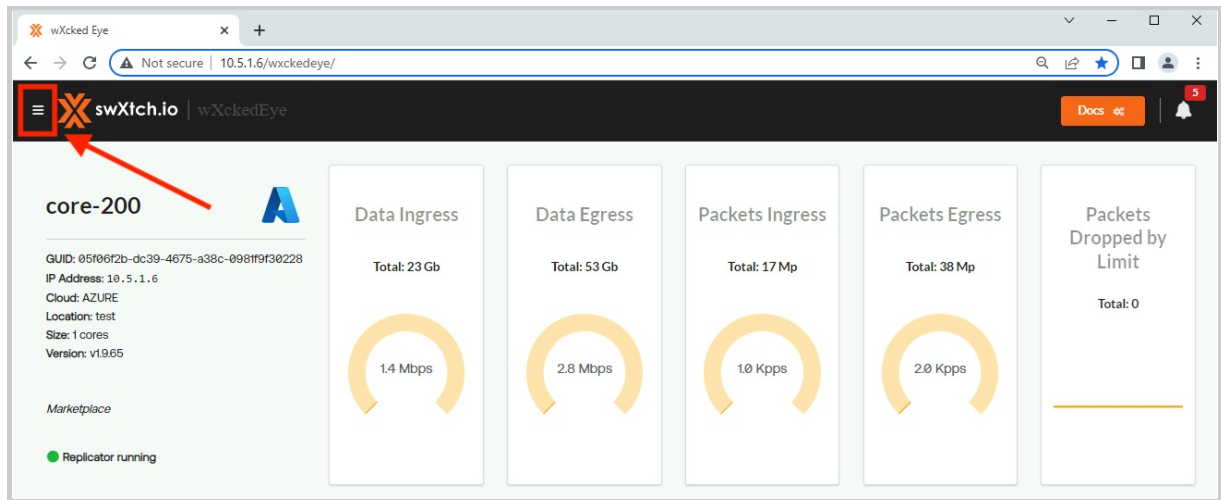
wXcked Eye cloudSwXtch Page

WHAT TO EXPECT

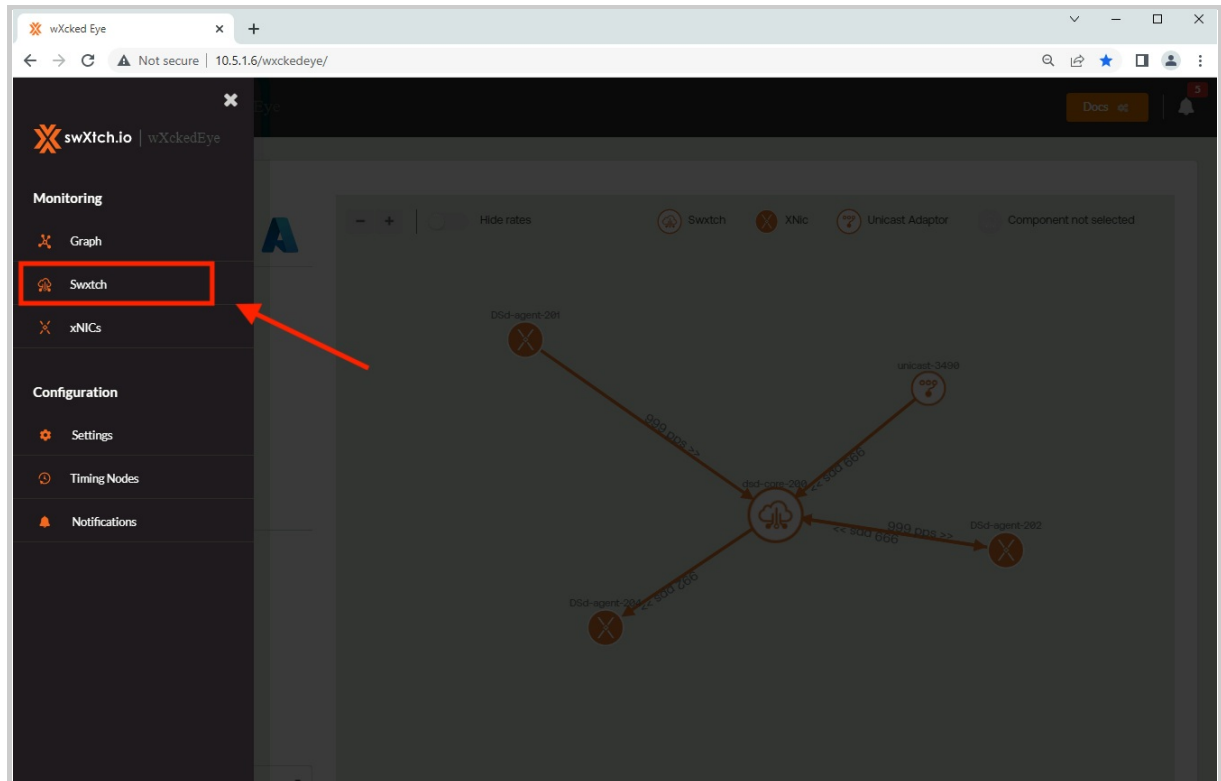
In this article, you will learn how to view the performance metrics for your cloudSwXtch and the xNICs associated with it.

Locating the wXcked Eye cloudSwXtch Page

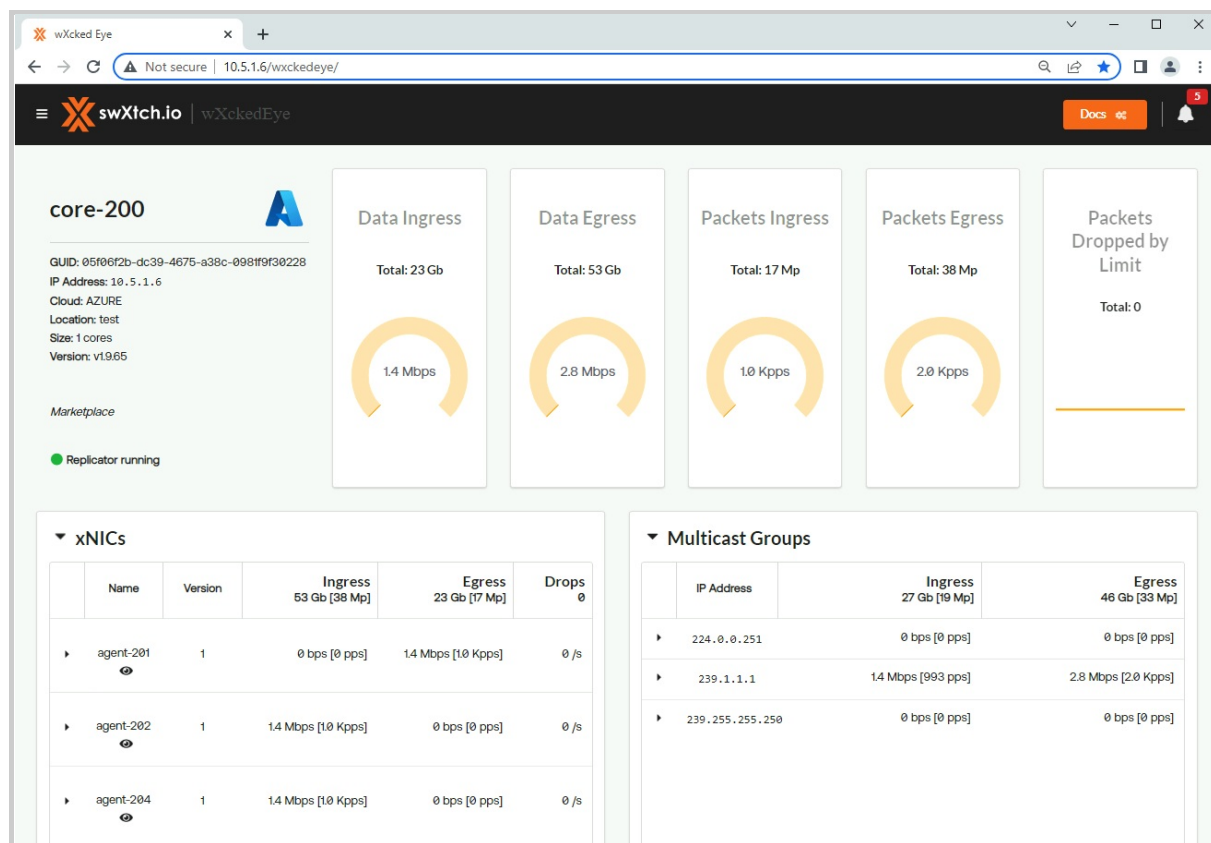
To navigate to the cloudSwXtch page, users will need to click on the menu option at the top right hand corner by the swxtch.io logo.



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



Navigating the wXcked Eye cloudSwXtch Page



cloudSwXtch Key Performance Metrics

Once the page loads, users will be given a cloudSwXtch view. This acts as the main page for the wXcked Eye user interface with detailed information regarding the cloudSwXtch and gauges to illustrate key performance metrics. The gauges are organized into 5 categories:

- Data Ingress
- Data Egress
- Packets Ingress
- Packets Egress
- Packets Dropped by Limit

Data ingress/egress are displayed in megabits per second (Mbps) while packets ingress/egress are displayed in kilo packets per second (Kpps). In addition to the rate, the total number of megabits and kilo packets are displayed for the user.

cloudSwXtch Information Panel

core-200

GUID: 05f06f2b-dc39-4675-a38c-0981f9f30228

IP Address: 10.5.1.6

Cloud: AZURE

Location: test

Size: 1 cores

Version: v1.9.65

Marketplace

Replicator running

Update Swtch

Important network information of the cloudSwXtch such as location (resource groups), cloud, IP address, switch size & replicator status is shown in the top left card along with its name, which in this case is core-200.

xNICs and Multicast Group Panels

swXtch.io

wXckedEye

Docs

5

Replicator running

Update Swtch

xNICs

Name	Version	Ingress 55 Gb [40 Mp]	Egress 24 Gb [17 Mp]	Drops 0
agent-201	1	0 bps [0 pps]	1.4 Mbps [10 Kpps]	0 /s
Multicast Groups				
IP: 224.0.0.251		Ingress: 0 bps [0 pps]	Egress: 0 bps [0 pps]	
IP: 224.0.1.129		Ingress: 0 bps [0 pps]	Egress: 0 bps [0 pps]	
IP: 239.1.1.1		Ingress: 0 bps [0 pps]	Egress: 1.4 Mbps [10 Kpps]	
agent-202	1	1.4 Mbps [10 Kpps]	0 bps [0 pps]	0 /s
agent-204	1	1.4 Mbps [10 Kpps]	0 bps [0 pps]	0 /s

Multicast Groups

IP Address	Ingress 28 Gb [20 Mp]	Egress 48 Gb [35 Mp]
224.0.0.251	0 bps [0 pps]	0 bps [0 pps]
xNICs		
Name: agent-201	Ingress: 0 bps [0 pps]	Egress: 0 bps [0 pps]
Name: agent-202	Ingress: 0 bps [0 pps]	Egress: 0 bps [0 pps]
Name: agent-204	Ingress: 0 bps [0 pps]	Egress: 0 bps [0 pps]
239.1.1.1	1.4 Mbps [10 Kpps]	2.8 Mbps [20 Kpps]
239.255.255.250	0 bps [0 pps]	0 bps [0 pps]

cloudSwXtch view - xNICs and Multicast Groups Panels

In the bottom half of the cloudSwXtch page, users will be able to see two additional panels: xNICs and Multicast Groups. The xNICs panel lists the agents that are connected to the cloudSwXtch -- in this case, test-core-100. 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.

Similar to the xNICs panel, **Multicast Groups** lists the IP addresses of different data streams related to the cloudSwXtch with the ingress/egress rates also displayed. Furthermore, using the dropdown feature will reveal the associated xNICs streaming that data.

wXcked Eye xNICs Page

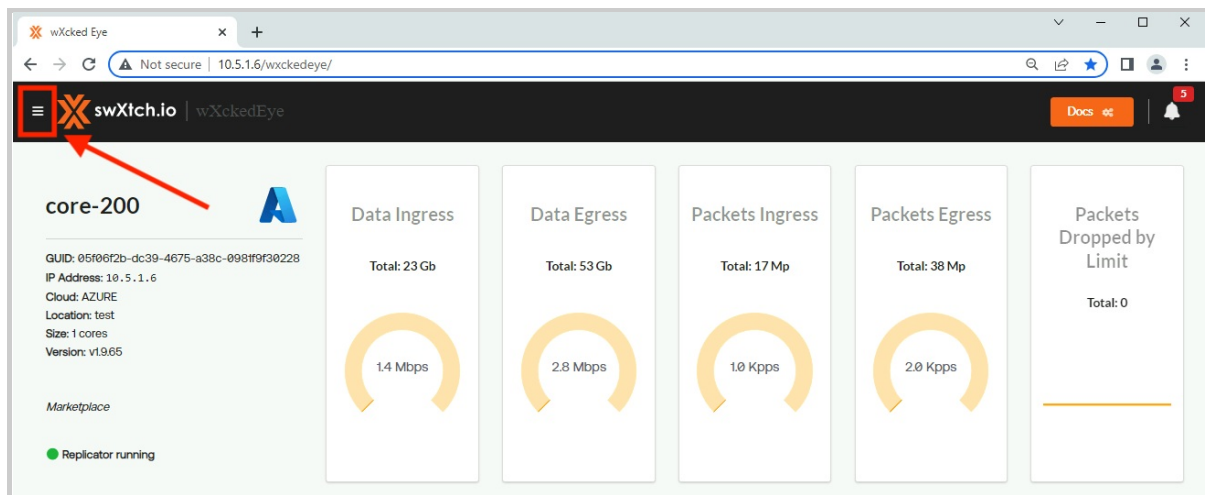
WHAT TO EXPECT

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

Locating the xNIC View Page

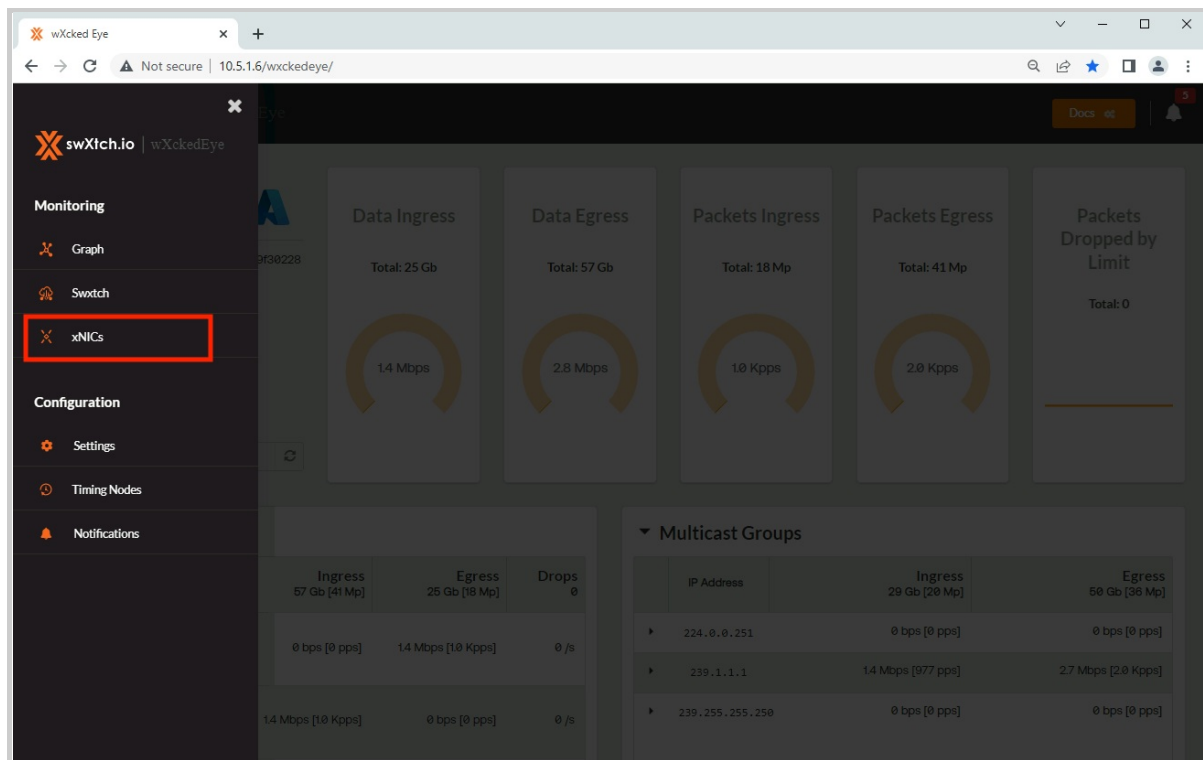
Similar to the yin and yang nature of the xNICs and Multicast Group panels, the xNIC page provides users with an opposite look of their cloudSwXtch from the xNICs perspective, breaking down performance at an agent's level.

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



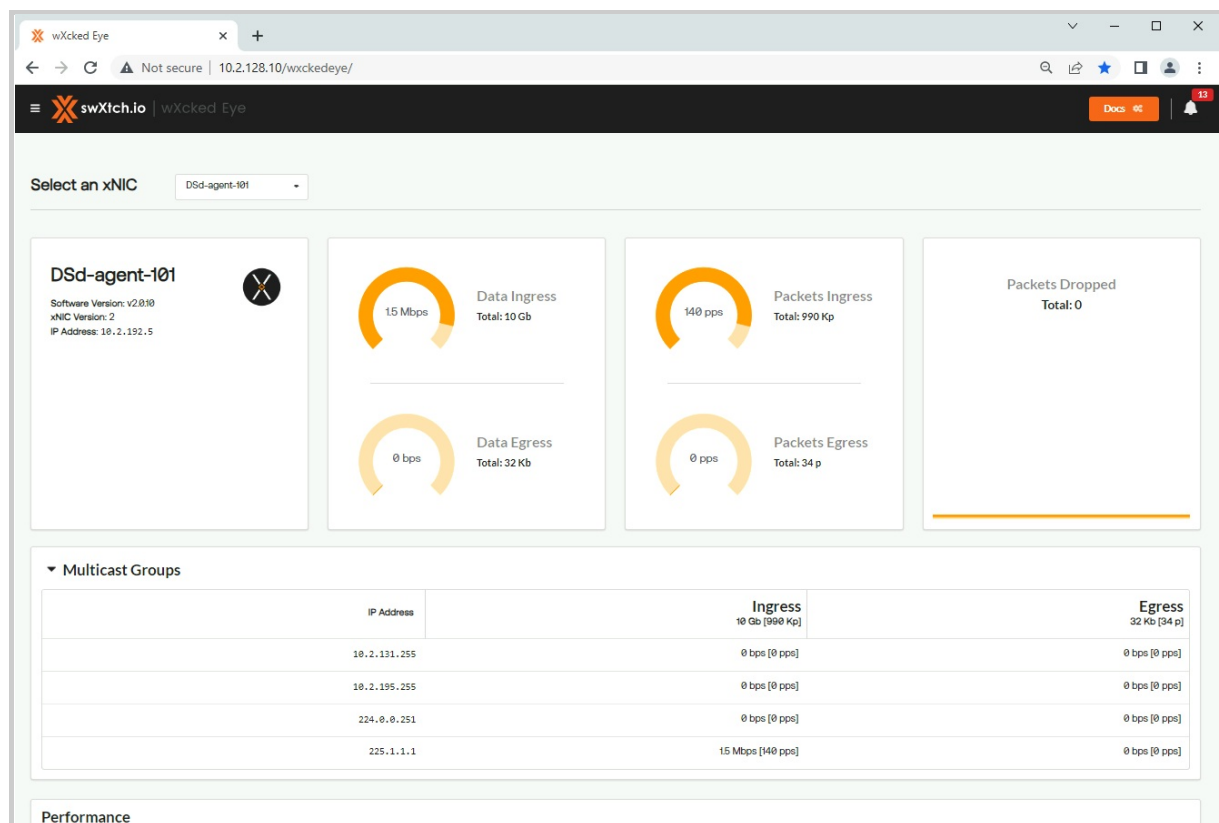
Howto access the navigation menu in the wXcked Eye UI

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



Selecting xNIC in the Navigation Menu will open the xNIC page.

Navigating the xNIC page



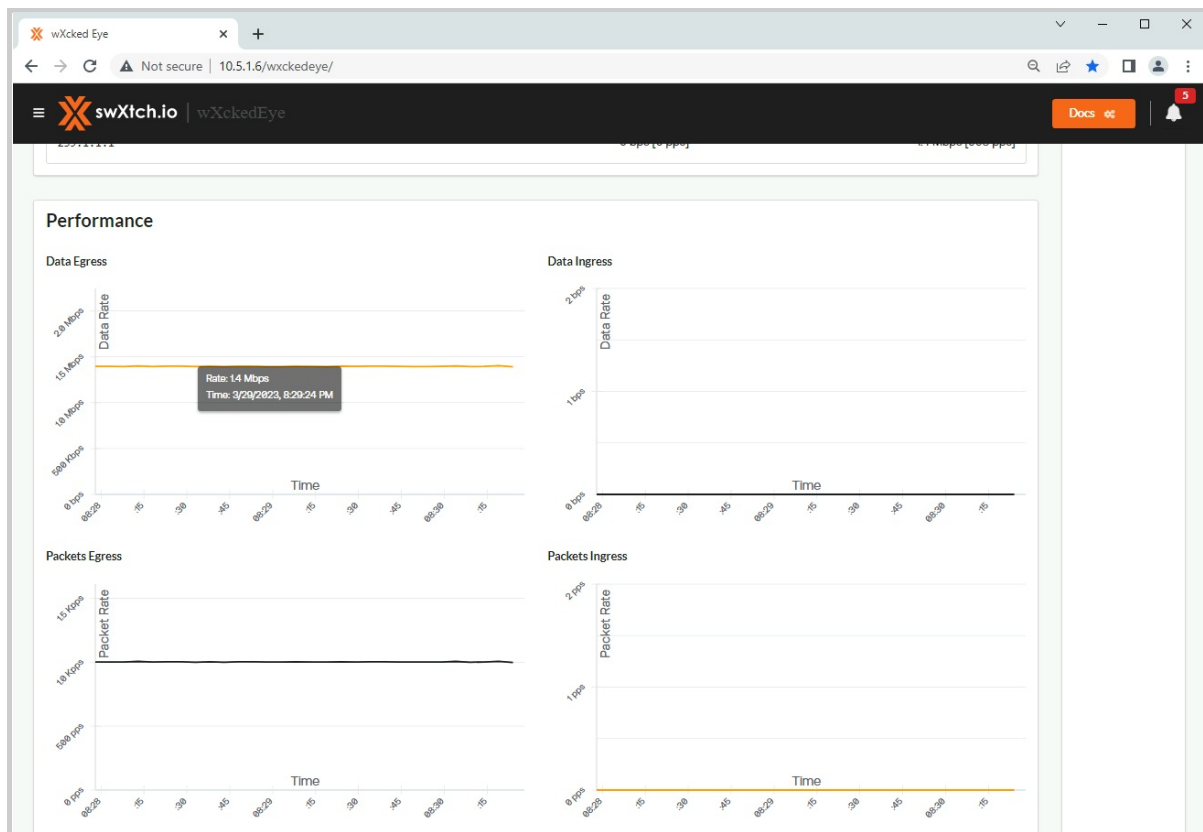
At first glance, the xNIC page looks very similar to the cloudSwXtch view. However, instead of focusing on the cloudSwXtch, users are given key information and performance metrics for a single xNIC. In the example above, one noticeable difference is the inclusion of the "Select an xNIC" dropdown menu at the top of the screen. Here, a user can select an agent they wish to monitor (agent-101).

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.

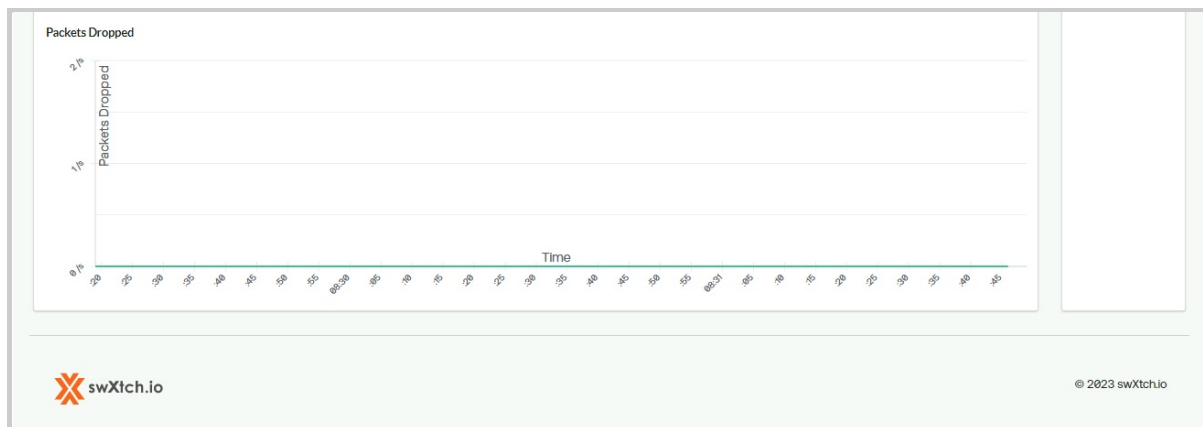
Performance

In addition to the 5 familiar gauges, the xNIC view also 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 egress/ingress and packets egress/ingress over :15 second increments.



Performance Panel: 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 :05 second increments



Performance Panel: Packets Dropped By Limit

Configure cloudSwXtch with wXcked Eye

WHAT TO EXPECT

wXcked Eye allows users to configure their cloudSwXtch directly from the user interface.

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 on their VM. They should use the same IP address of their cloudSwXtch to prefix the URL.

Bash

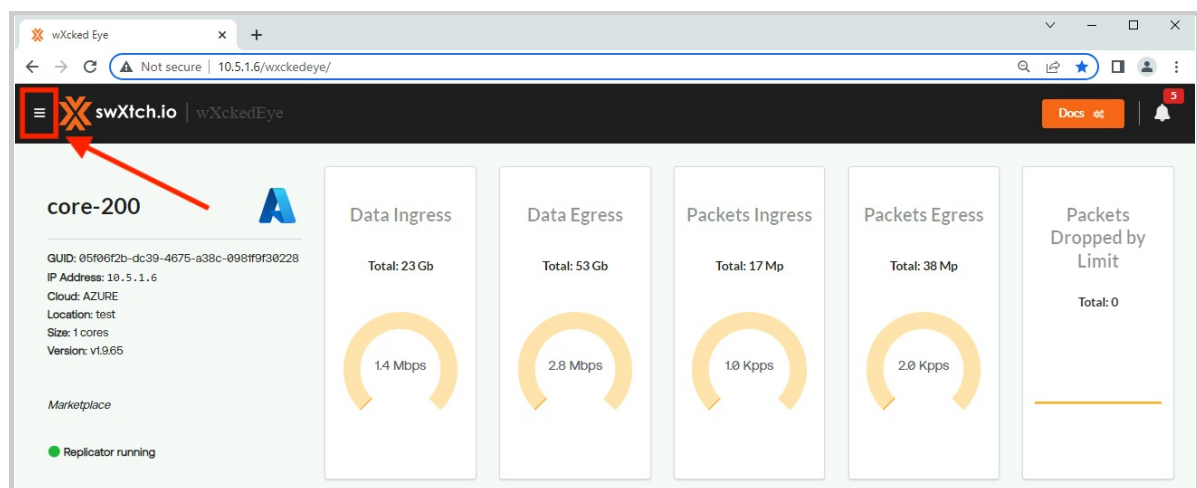
Copy

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

Finding the Settings page

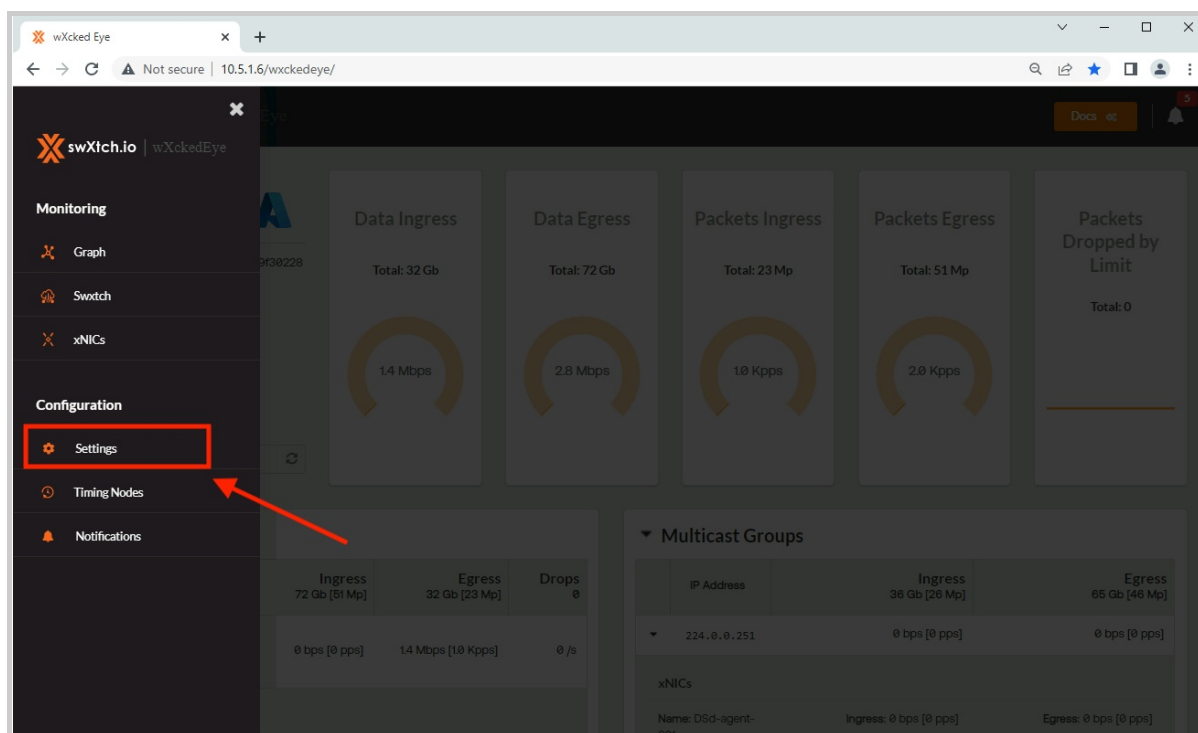
To find the Settings page in the wXcked Eye UI:

1. Click the three horizontal lines next to the swXtch.io logo.



Clicking on the three horizontal lines by the swXtch.io logo will open the Navigation menu.

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.

The **General** tab is designed to give users a detailed look into the entitlements associated with their network. In the example below, the user has a license that enables mesh, high availability, protocol fanout and clock sync with a max of 10 clients and 2000 Mbps bandwidth.

In addition to providing this information to users, the General tab also allows them to adjust the **Data Refresh Period**. This gives users control of how often the data is refreshing in real time. The default value is set to 5 seconds.

The screenshot shows the wXcked Eye web interface. The browser address bar shows '10.2.128.10/wxckedeye/'. The page title is 'wXcked Eye'. The 'Settings' section has four tabs: 'General' (selected), 'Mesh', 'High Availability', and 'Protocol Fanout'. Under 'General', there is a section for 'core-100' with the following information:

- Version: V2.0.10
- Size: 1
- Ctrl IP: 10.2.128.10
- Ctrl Port: 10002
- Subnet Data Prefix: 10.2.192.0/22
- Subnet Ctrl Prefix: 10.2.128.0/22
- Gateway IP: 10.2.192.1

Below this information is an 'Update swXtch' button. To the right is the 'Entitlements' section, which contains a table:

Max Clients	Max Bandwidth (Mbps)	Enabled Features					wXcked Eye
		Mesh	High Availability	Protocol Fanout	PTP	Bridge	
50	32000	✓	✓	✓	✓	✓	✓

Below the table is the 'Data refresh period' section, which includes a slider and a '5 secs' button. The footer shows the swXtch.io logo and copyright information: '© 2023 swXtch.io'.

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.

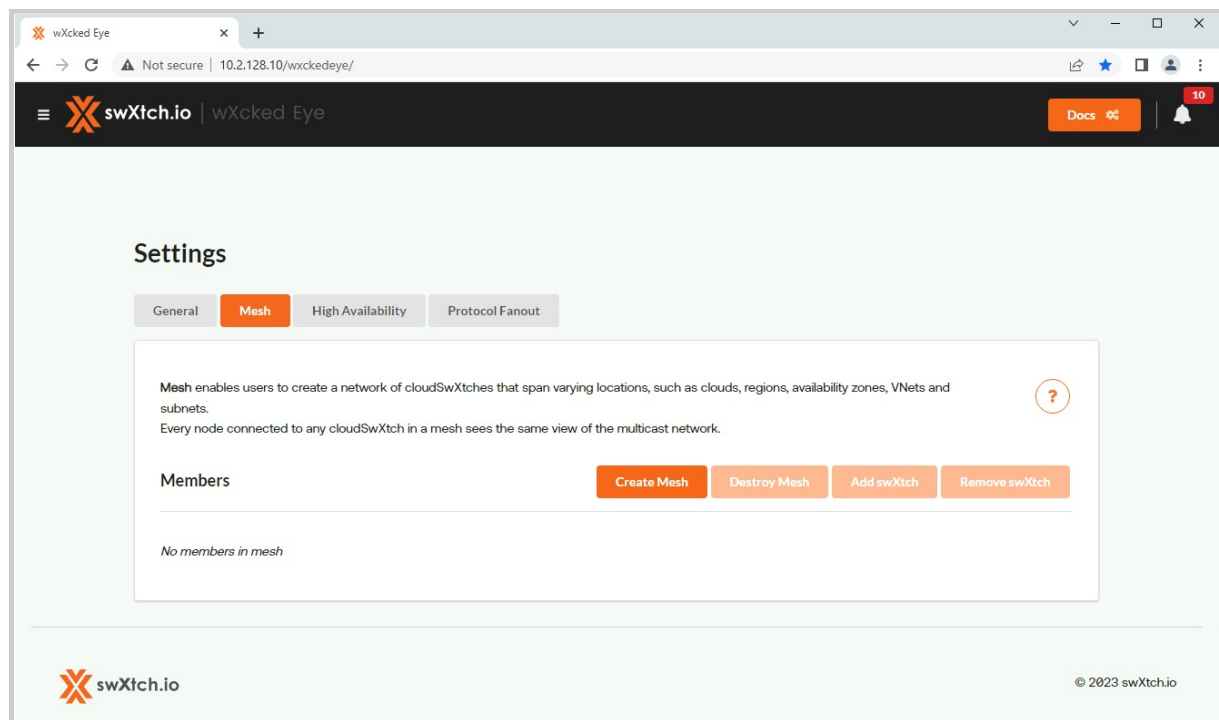
The **Mesh** tab is organized into 4 functions:

- [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.



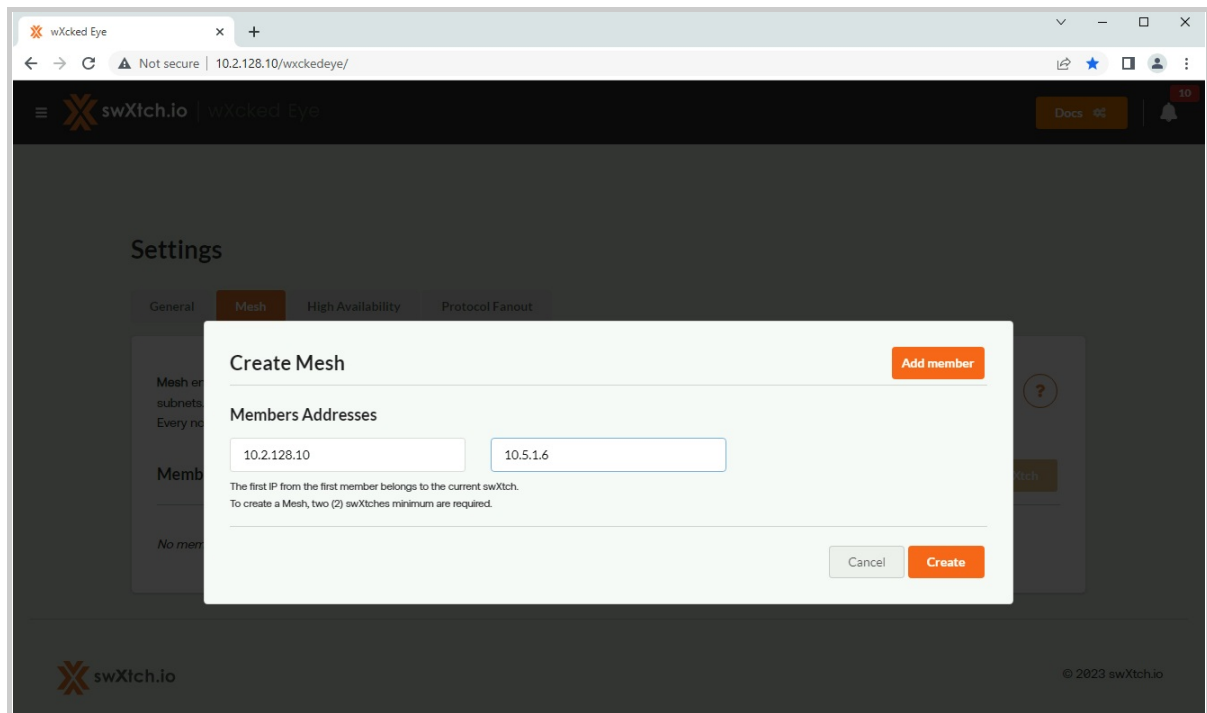
Mesh and High Availability

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:

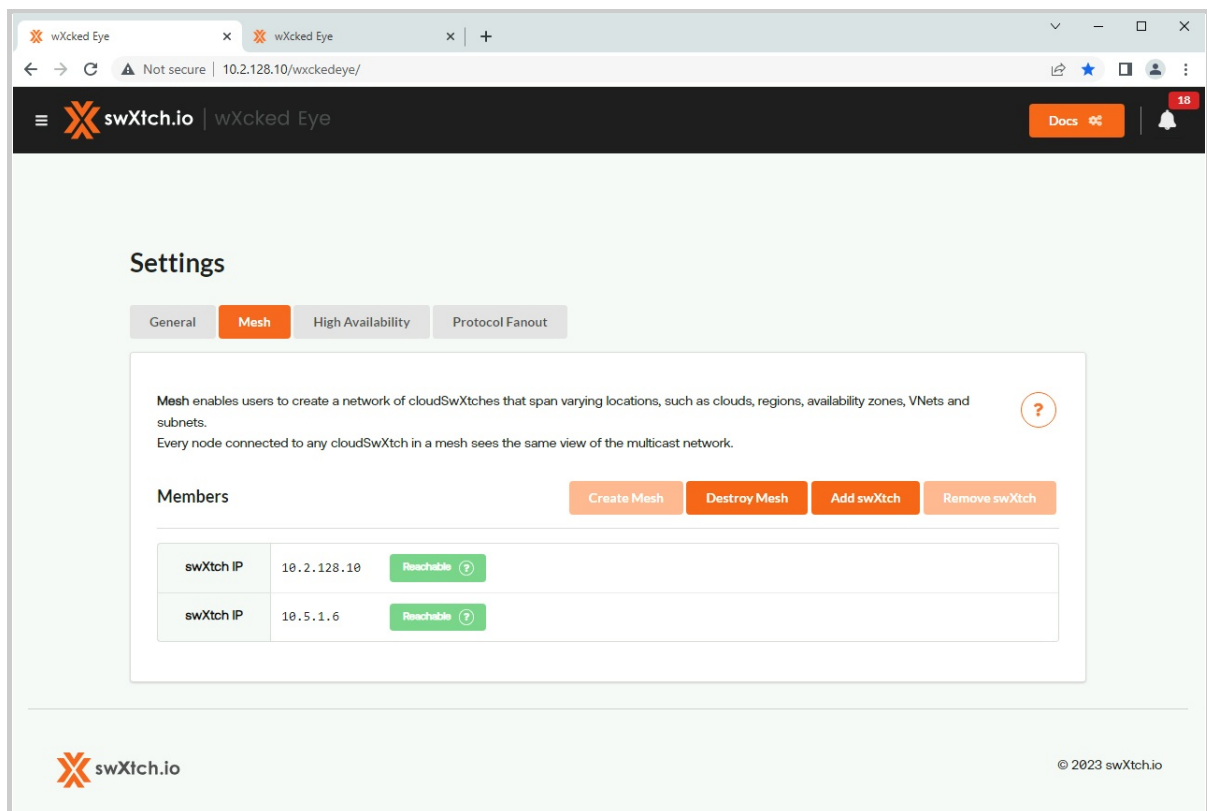
1. Click on the "Create Mesh" button. A new window will pop up.



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.

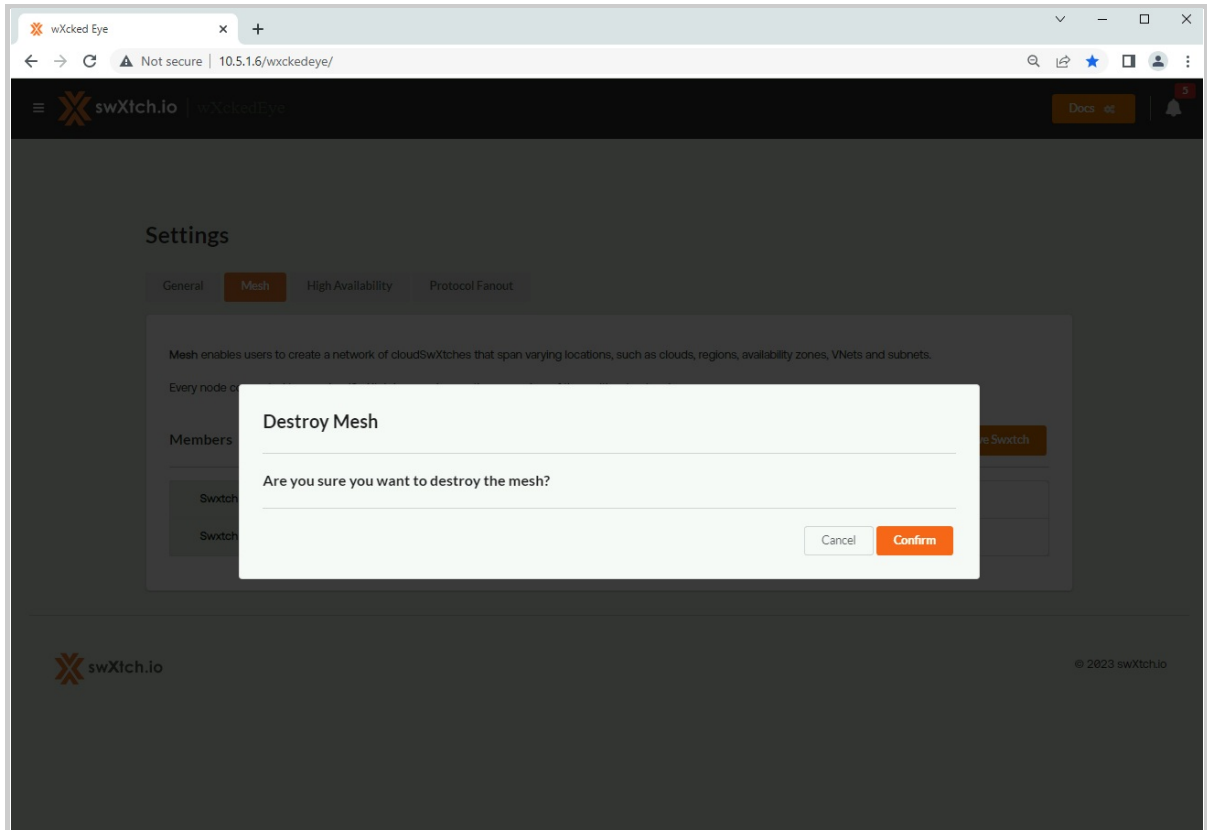
3. Click "Create" after you have completed adding members. The members should now be listed in the newly formed mesh. A tag on each swXtch IP will display whether or not the VM is reachable.



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.



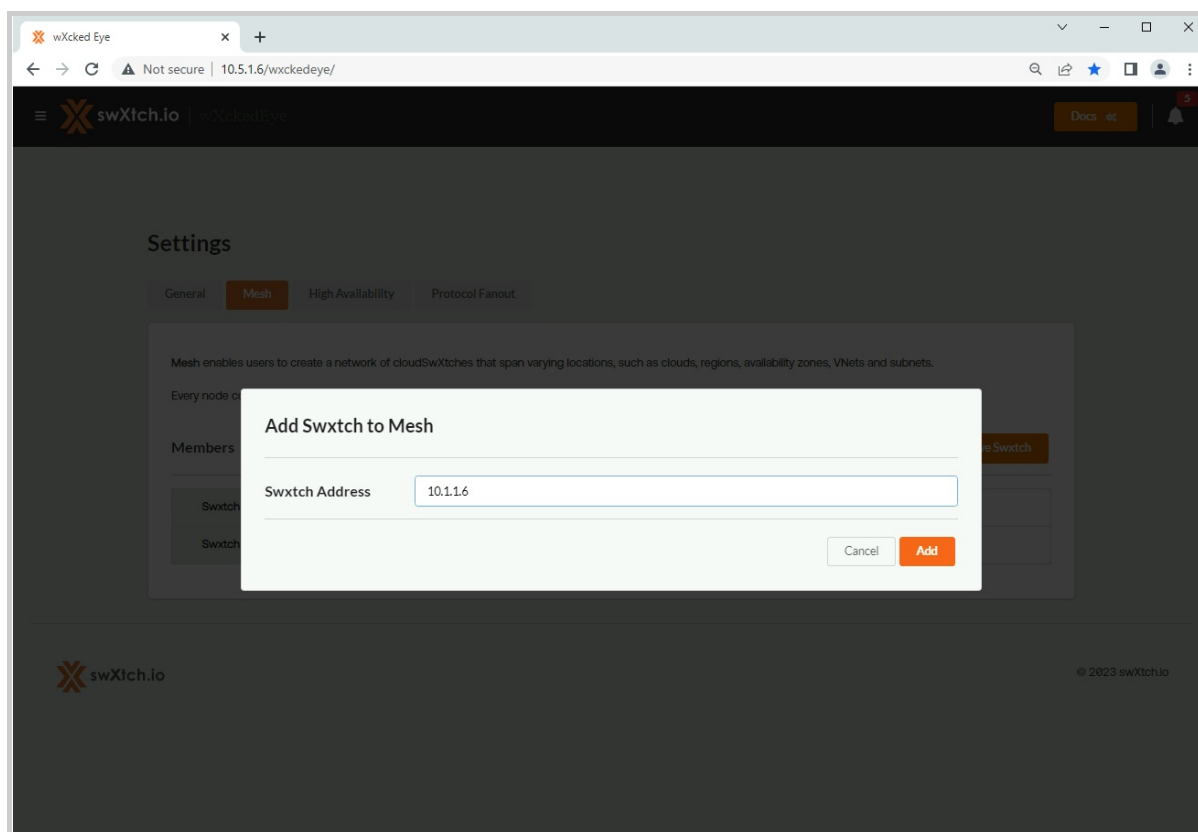
2. Select "Confirm." Your mesh and its members should no longer be listed in the Mesh tab.

Add SwXtch

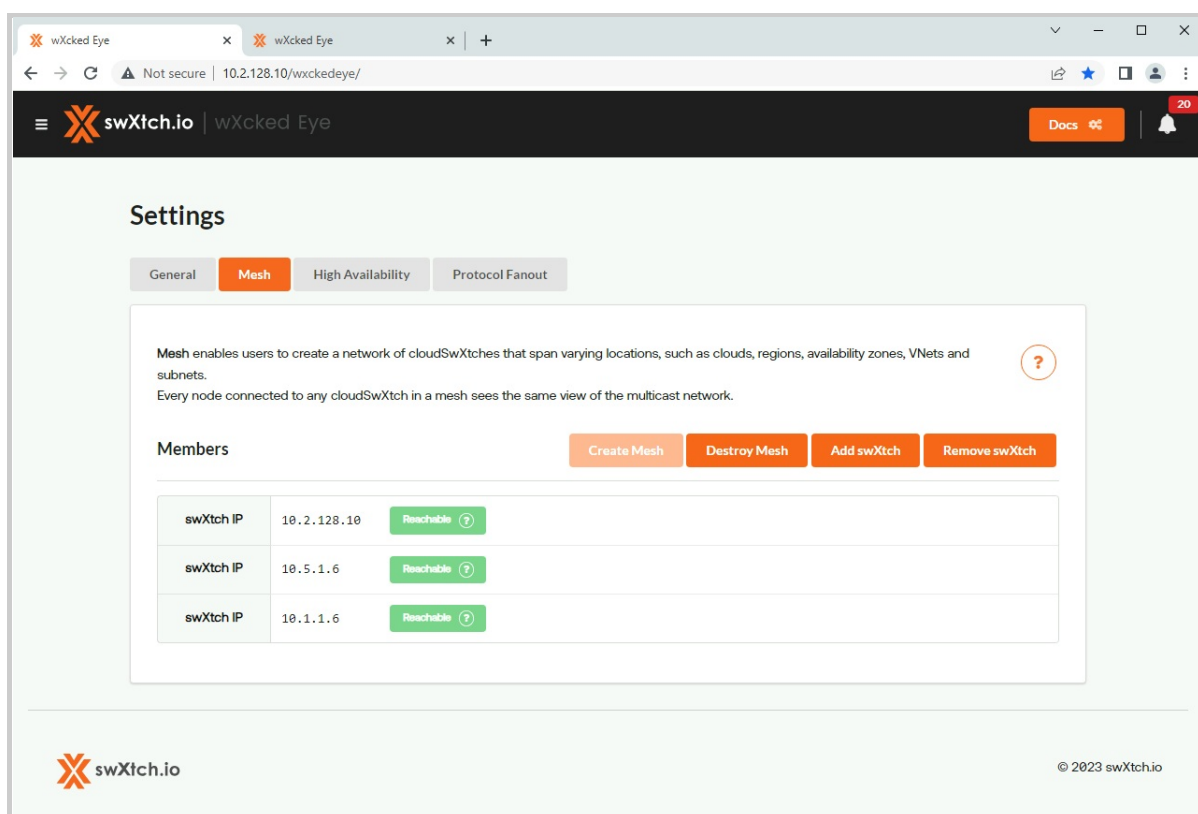
After creating a mesh, users may decide that they need additional cloudSwXtches. To add a cloudSwXtch:

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.



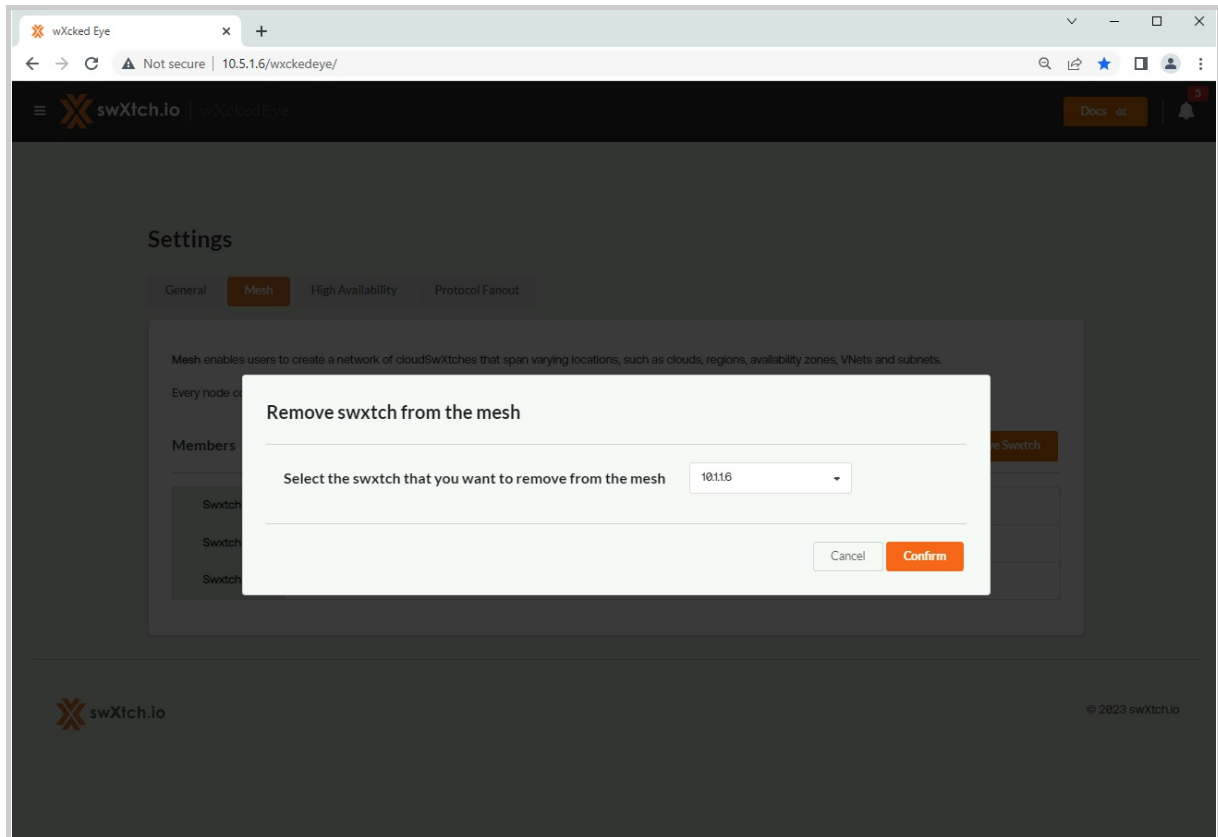
3. Click "Add." A new cloudSwXtch should now be listed in your mesh.



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

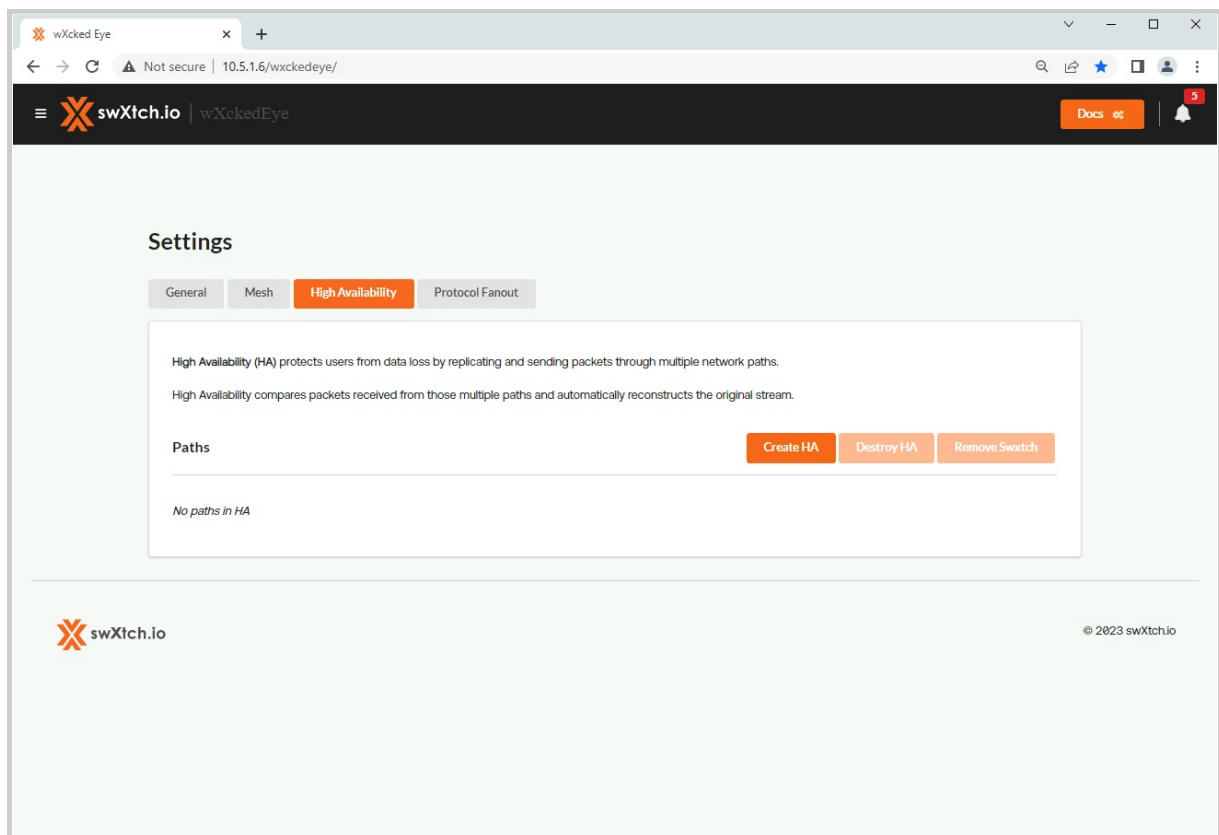
Navigating to the High Availability tab

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 3 functions:

- [Create HA](#)
- [Destroy HA](#)
- [Remove SwXtch](#)

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 HA" button in the cloudSwXtch you wish to include in your high availability configuration. A new window will open.

2. **Enter** a name for your HA configuration. In this example, the HA is named "My High Availability."

The screenshot shows the 'Create HA' dialog in the swXtch.io interface. The dialog has a 'Name' field containing 'My High Availability'. Below this, there is a 'Paths' section with an 'Add path' button. Two path cards are visible. The first card has a 'Name' field with 'Path 1' and a 'swXtches' field with '10.2.128.10'. The second card has a 'Name' field with 'Path name' and a 'swXtches' field with '000.0.0.0'. Each card has an 'Add Swtch' button. At the bottom of the dialog, there is a note: 'You need at least 2 (two) paths to create HA'. There are 'Cancel' and 'Create' buttons at the bottom right.

3. **Enter a name** for the first path and **the IP addresses** for the relevant cloudSwXtches.

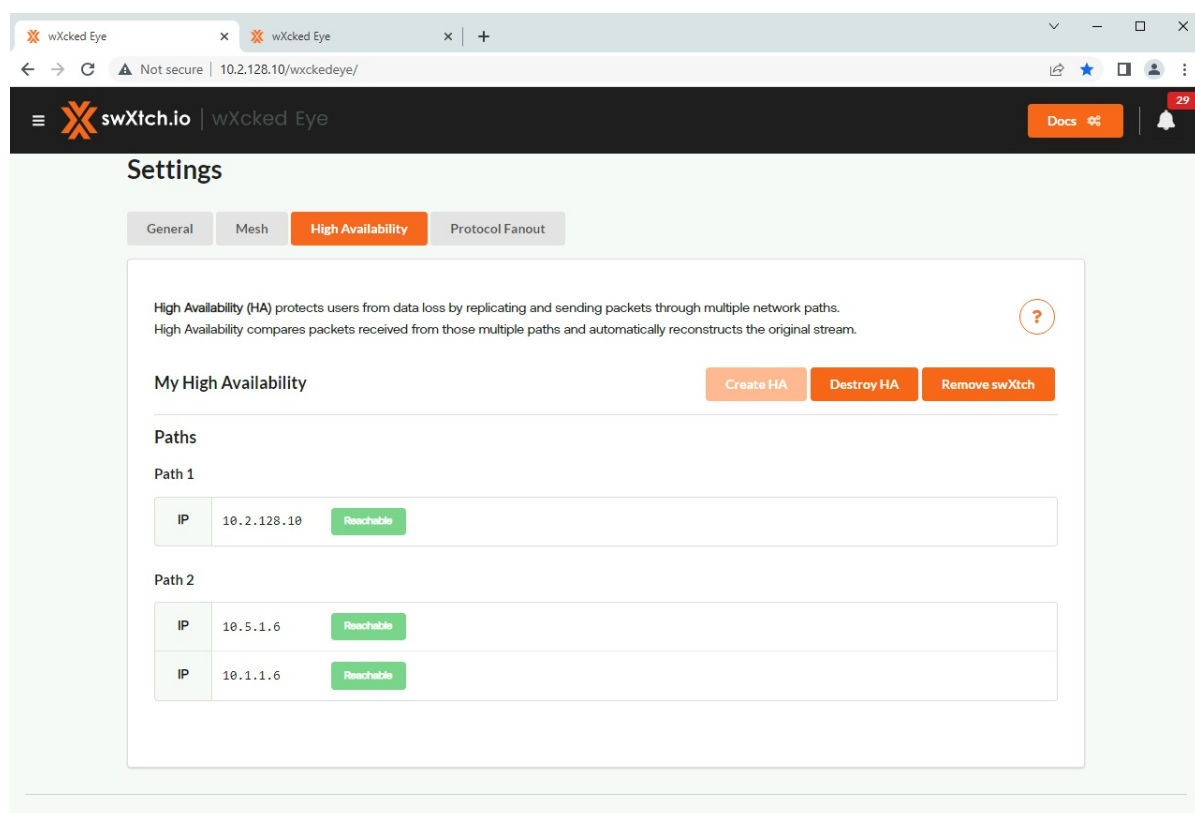
4. **Enter a name** for the second path and the IP addresses for the relevant cloudSwXtches.

a. **Please note:** You must have more than 1 path in order to have a working HA.

5. **OPTIONAL:** Add an additional cloudSwXtch to a Path by clicking "Add SwXtch." In this example, the user assigned 2 cloudSwXtches to Path 2.

The screenshot shows the 'Create HA' dialog in the swXtch.io interface, showing the next step. The 'Name' field still contains 'My High Availability'. Under 'Paths', the first card is unchanged. The second card now has 'Path 2' as the name and two swXtches: '10.5.1.6' and '10.1.1.6'. There are 'Add path' and 'Add swXtch' buttons. At the bottom of the dialog, there is a note: 'The first IP from the first path belongs to the current swXtch. You need at least 2 (two) paths to create HA'. There are 'Cancel' and 'Create' buttons at the bottom right.

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



With High Availability now formed, 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 HA or remove the cloudSwXtch.

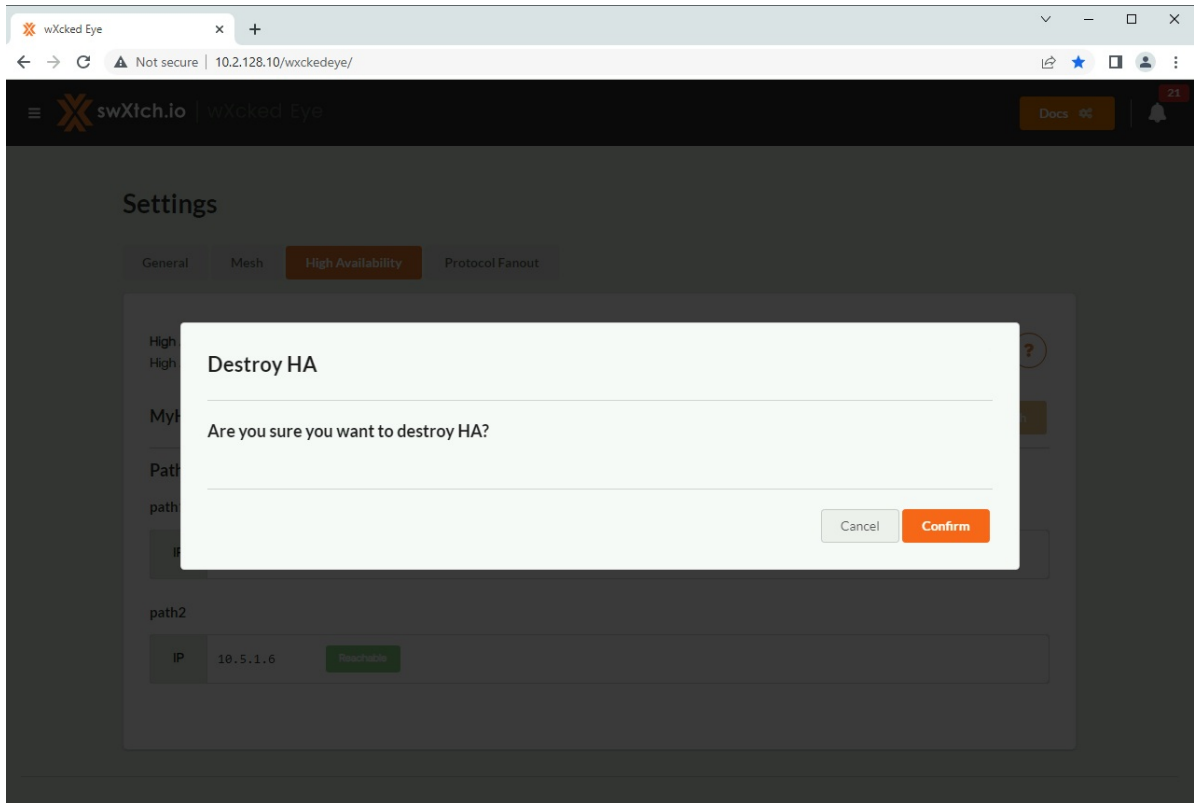
Warning

If you try to create another HA, the wXcked Eye UI will destroy the current HA and replace it with the new configuration.

Destroy HA

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

1. Click "Destroy HA." A new prompt will appear, asking you to confirm the action.



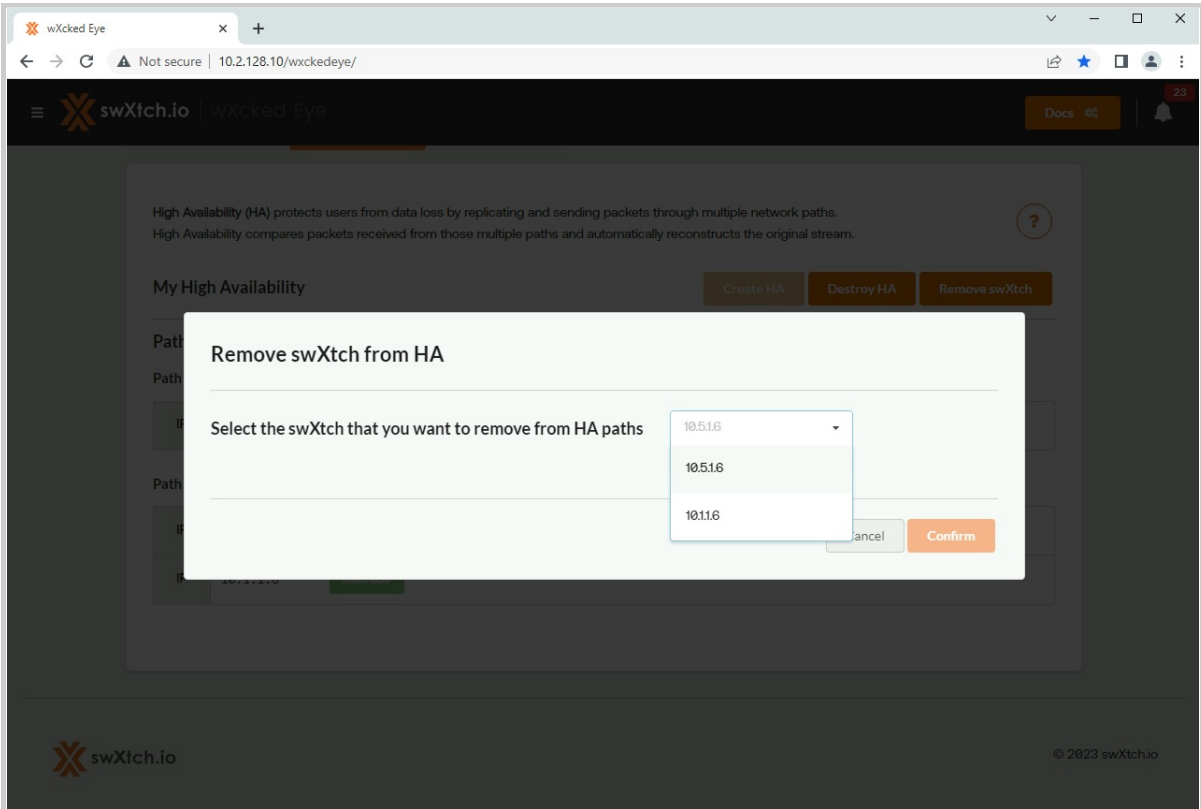
2. Select "Confirm."

Your HA is now destroyed. All associated cloudSwXtches will show a blank member list for HA.

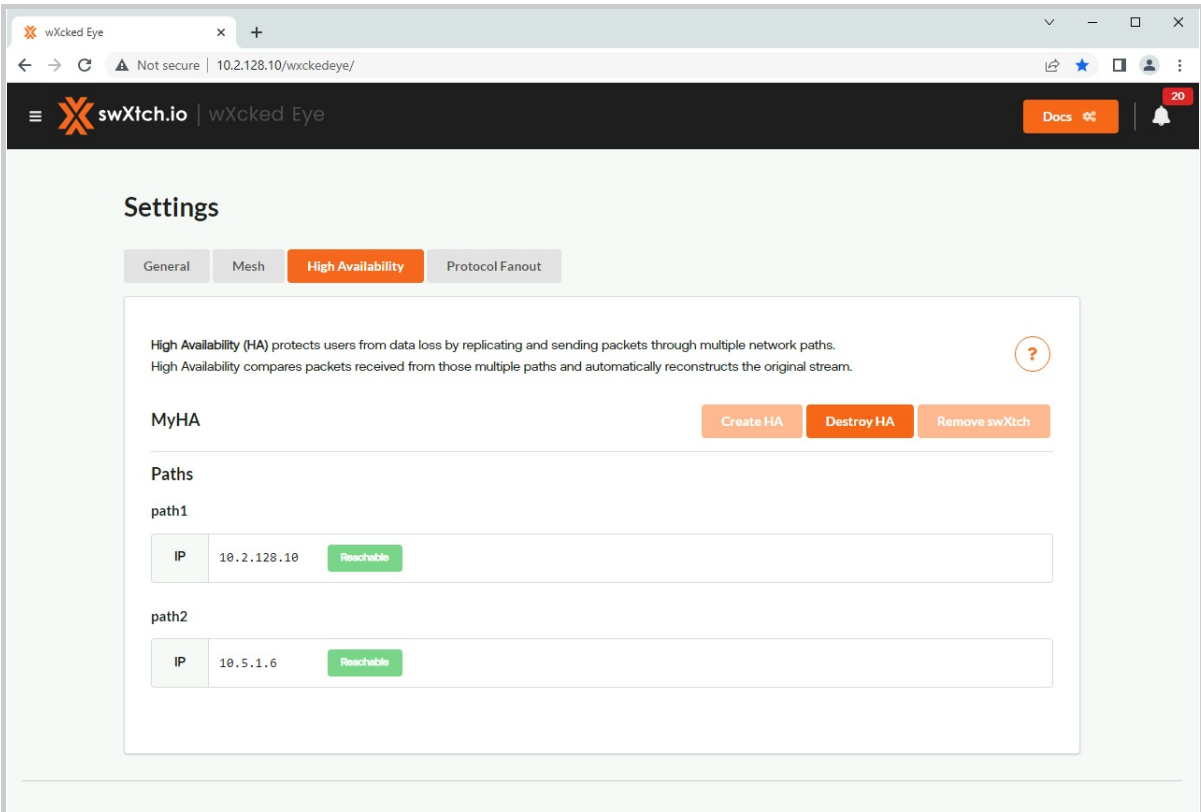
Remove swXtch

1. Click **Remove swXtch**. A new prompt will appear, asking you to confirm the removal of the current cloudSwXtch.

2. Select the cloudSwXtch you wish to remove from the HA in the dropdown menu. In this example, the user will be removing 10.1.1.6.



3. **Select "Confirm."** As you can see, cloudSwXtch 10.1.1.6 is now longer listed under Path 2 in the example below. If you were in the wXcked Eye of 10.1.1.6, the HA page should now be blank with the original members no longer visible.



Protocol Conversion and Fanout with wXcked Eye

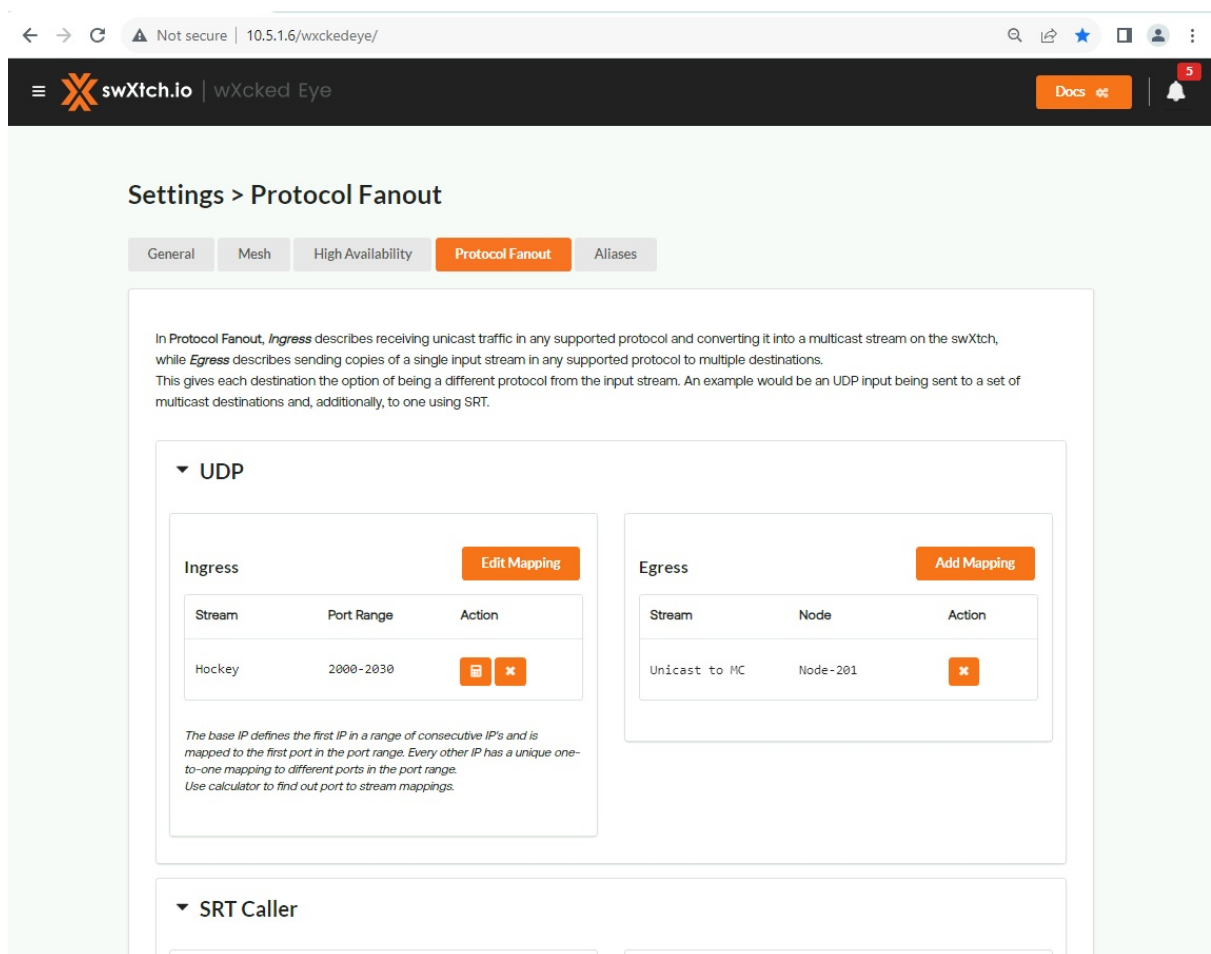
WHAT TO EXPECT

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 and SRT connections via the Protocol Conversion and Fanout feature on wXcked Eye.

Setting Up Aliases

The Protocol Fanout tab utilizes Stream and Node names set in the Aliases tab. For more information on how to do this, please see the [Aliases](#) article.



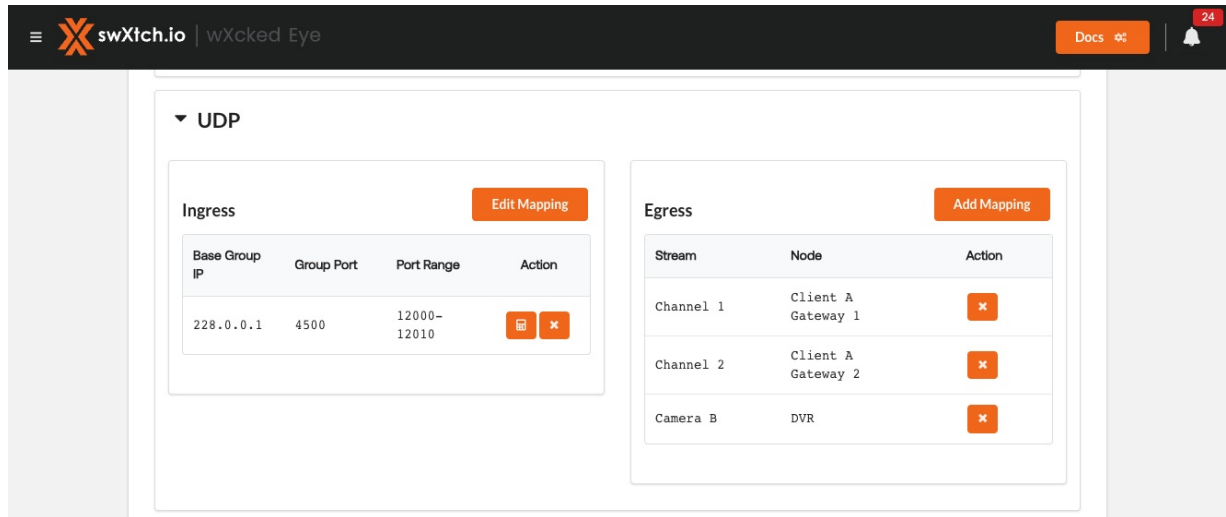
Protocol Fanout and Conversion is a cloudSwXtch feature that allows users to send copies of a single input stream in any supported protocol to multiple destinations. In wXcked Eye, users can send/receive UDP traffic and establish SRT caller/listener connection methods.

UDP

Ingress vs. Egress

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

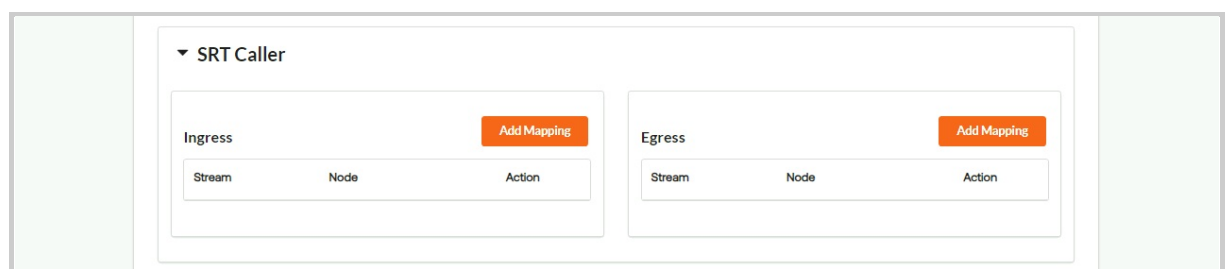
- **Ingress:** Data is coming into the cloudSwXtch.
- **Egress:** Data is leaving the cloudSwXtch.



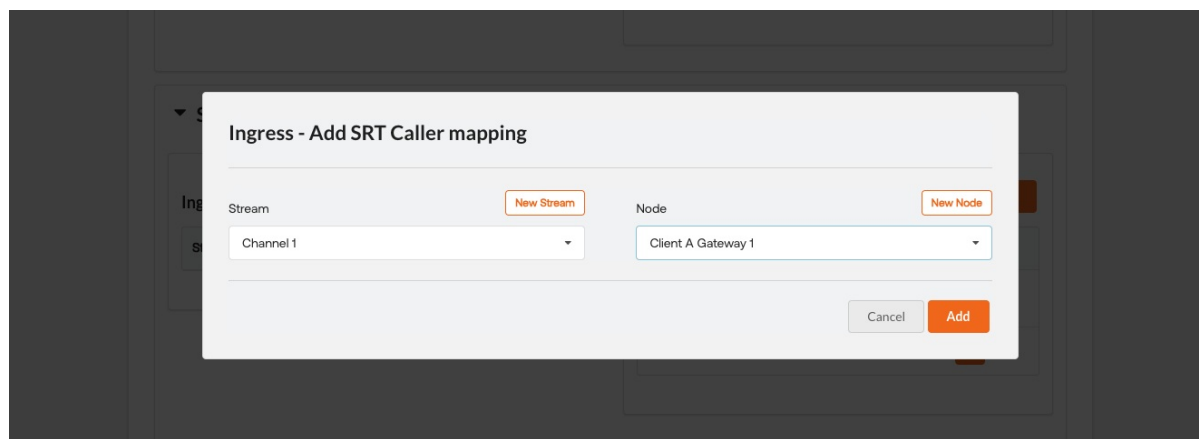
The UDP panel allows users to add mapping for UDP traffic entering and leaving the cloudSwXtch. For Ingress, a user will specify a Multicast Base Group IP, Group Port and a Port Range for endpoints to be able to send unicast data. Once that connection has been established, the cloudSwXtch will be able to ingest the unicast data as multicast.

For Egress, user will set the parameters for fanning out a multicast stream as unicast. To do this, the cloudSwXtch would need the user-named Stream or a new Multicast IP and the user-named Target Node for the endpoint to begin transmitting the data.

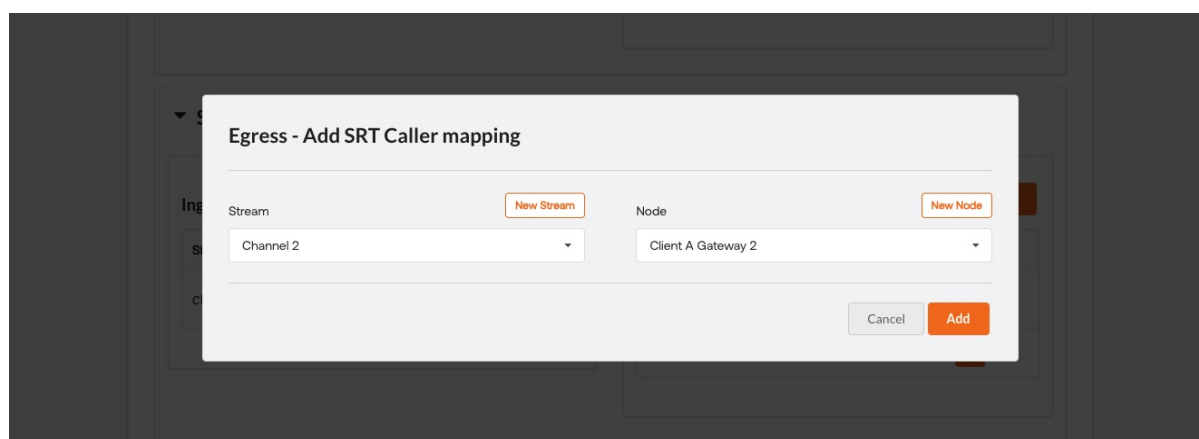
SRT Caller



The SRT Caller panel is organized into two sections: Ingress and Egress. Both house an "Add mapping" button. When selected, a new prompt will ask them to input the information necessary to make an SRT call.



For Ingress, users will need to specify a user-named Stream/Multicast IP and Target Node (Target IP and a Target Port.) The Target Node 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. After adding the mapping, the cloudSwXtch will then call out to the target source and receive multicast traffic through designated port.



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

SRT Listener

▼ SRT Listener

Ingress

Add mapping

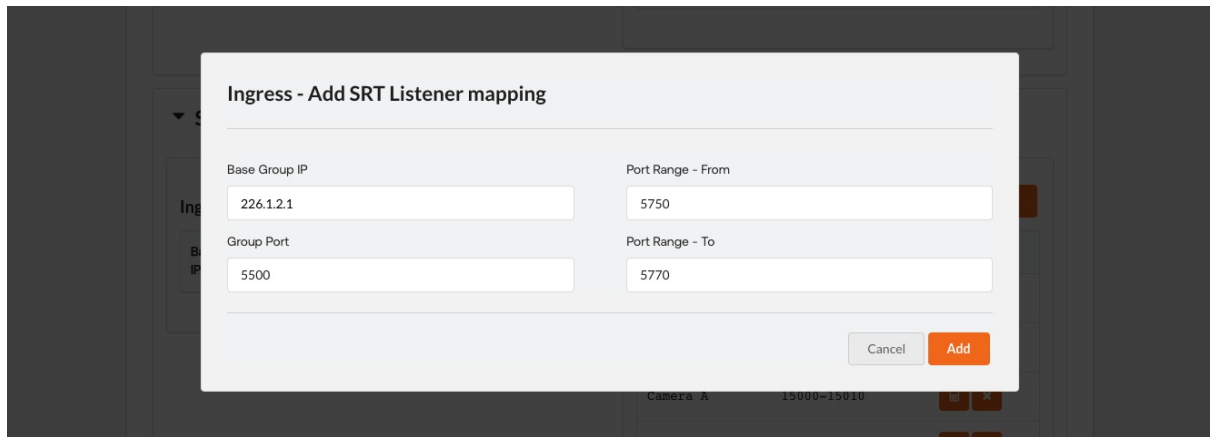
Multicast Base Address	Port Range	Action
225.1.1.1	1599-1602	

Egress

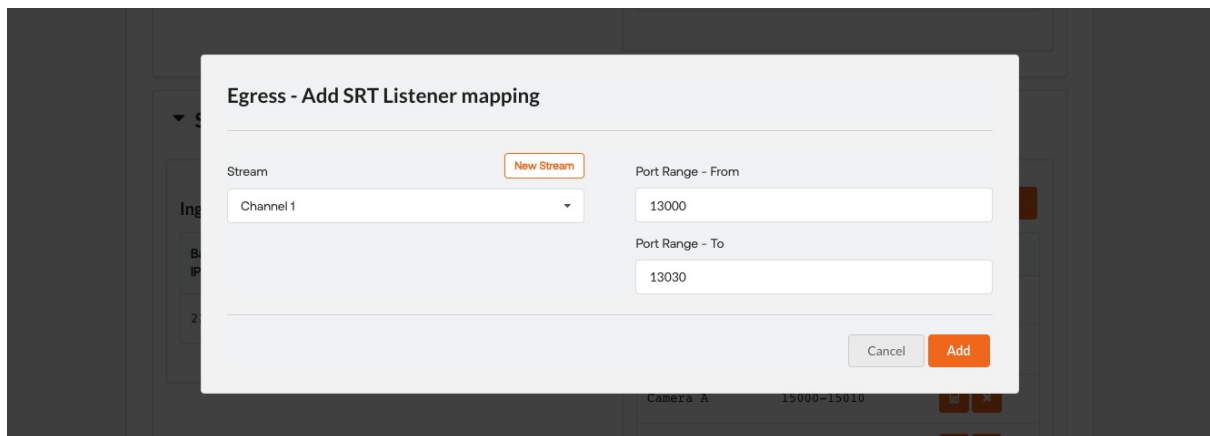
Add mapping

Stream / Multicast IP	Multicast Port	Port Range	Action
HD	1599	5250-5255	
UHD	1600	5300-5305	
OTS	1601	5400-5403	
Mosiac	1602	5500-5505	

Similar to the SRT Caller panel, SRT Listener contains two sections: Ingress and Egress. However, what differs is that the SRT Listener is essentially "listening" 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.

A screenshot of a web application showing a modal dialog box titled "Ingress - Add SRT Listener mapping". The dialog has a light gray background and is centered over a darker background. It contains four input fields arranged in a 2x2 grid. The top-left field is labeled "Base Group IP" and contains the text "226.1.2.1". The top-right field is labeled "Port Range - From" and contains the text "5750". The bottom-left field is labeled "Group Port" and contains the text "5500". The bottom-right field is labeled "Port Range - To" and contains the text "5770". At the bottom right of the dialog, there are two buttons: a gray "Cancel" button and an orange "Add" button.

For Ingress' "Add mapping" prompt, a user will specify the Base Group IP Address, Group Port and a port range of where an endpoint can send data to. Once the configuration is complete, the cloudSwXtch will now listen for producers of SRT traffic who connect to a port in that range. When a connection has been established, the cloudSwXtch will begin ingesting data.

A screenshot of a web application showing a modal dialog box titled "Egress - Add SRT Listener mapping". The dialog has a light gray background and is centered over a darker background. It contains a "Stream" dropdown menu on the left, which currently shows "Channel 1". Above the dropdown is a small orange button labeled "New Stream". To the right of the dropdown are two input fields for "Port Range - From" (containing "13000") and "Port Range - To" (containing "13030"). At the bottom right of the dialog, there are two buttons: a gray "Cancel" button and an orange "Add" button.

For Egress, the cloudSwXtch is listening for endpoints that are trying to receive SRT data. In the Egress Add Mapping prompt, a user will select either one of their multicast streams from the dropdown menu or specify a new Multicast IP. From there, depending on the user's bandwidth, they can set a Port Range of available entry points (up to 32) from which an endpoint can connect to the steam. By setting the necessary parameters, a consumer will then be able locate a target port and begin streaming data from the cloudSwXtch.

Aliases

WHAT TO EXPECT

The Aliases tab on the Settings page allows users to assign friendly names to their streams and nodes. In addition to streamlining protocol conversion and fanout configuration, the Aliases feature also makes the network graph easier to use.

In this section, users will learn how to add/edit streams and nodes in wXcked Eye.

swXtch.io | wXcked Eye

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.

▼ Streams Add Alias

Alias	Group IP	Group Port	Action
Hockey	239.1.1.2	2000	Edit Delete
Unicast to MC	239.1.1.3	3000	Edit Delete

▼ Nodes Add Alias

Alias	Node IP	Node Port	Action
Node-201	10.5.1.4	2000	Edit Delete
Node-202	10.5.1.5	3479	Edit Delete
Node-204	10.5.1.7	4500	Edit Delete





Streams

The Streams panel 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.

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.

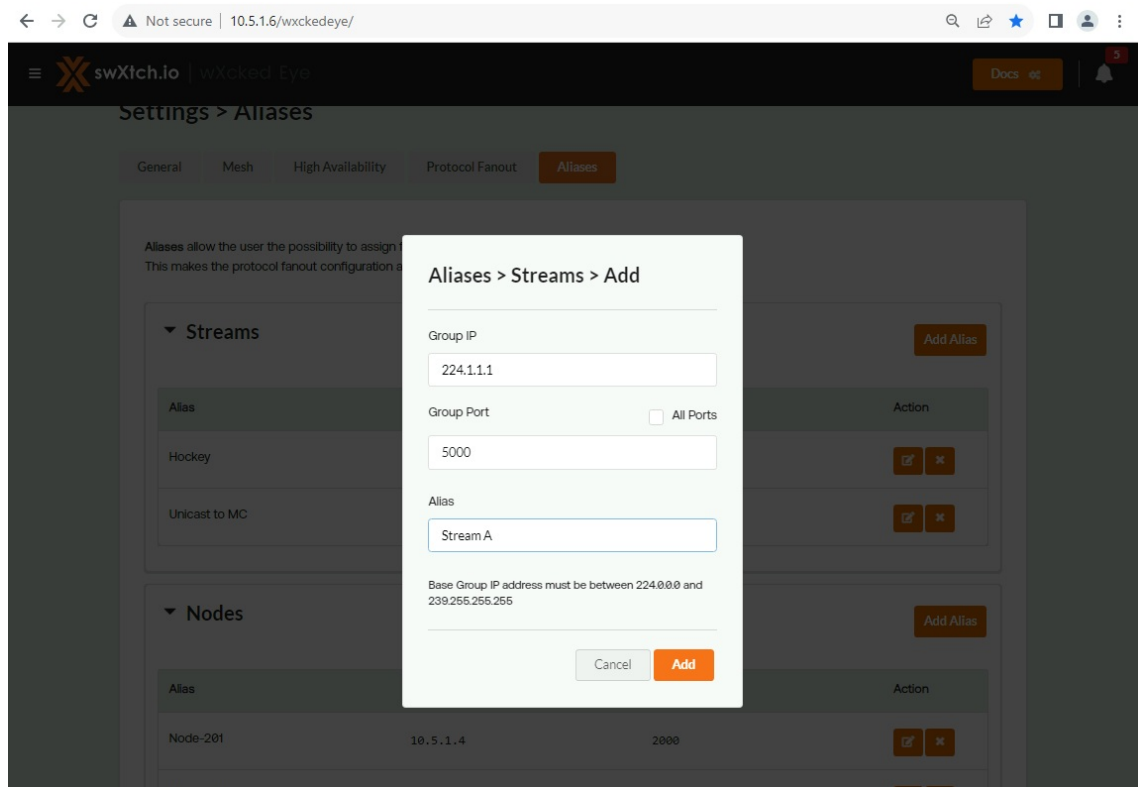
▼ Streams

Add Alias

Alias	Group IP	Group Port	Action
Hockey	239.1.1.2	2000	 
Unicast to MC	239.1.1.3	3000	 

To add a new stream:

1. Select **Add Alias** in the right hand corner of the **Streams** panel. A new window will open.



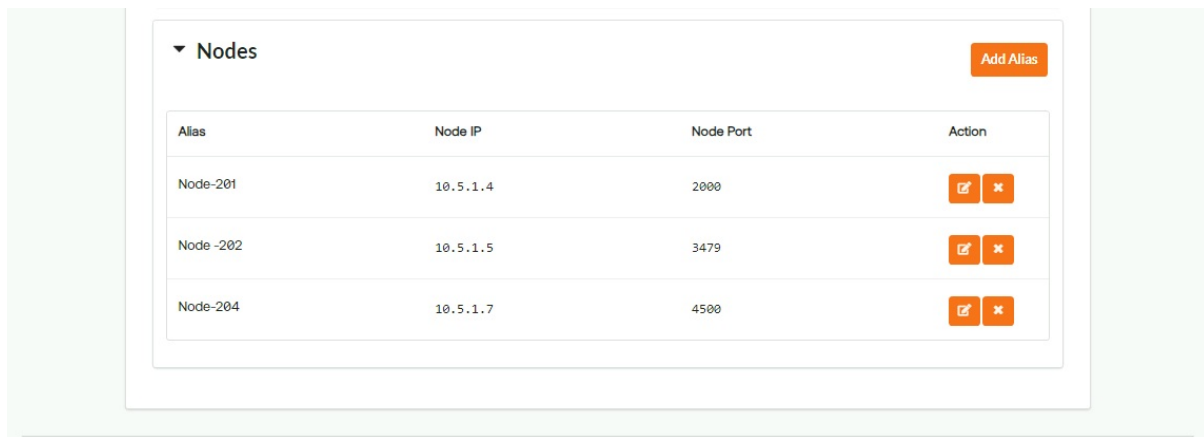
2. Enter the **Multicast Group IP**, **Group Port** and a **user-friendly name** under **Alias**. In the example, the user assigns the Alias, Stream A.
3. Click **Save**.
4. A new stream will be added to the list.

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

Nodes

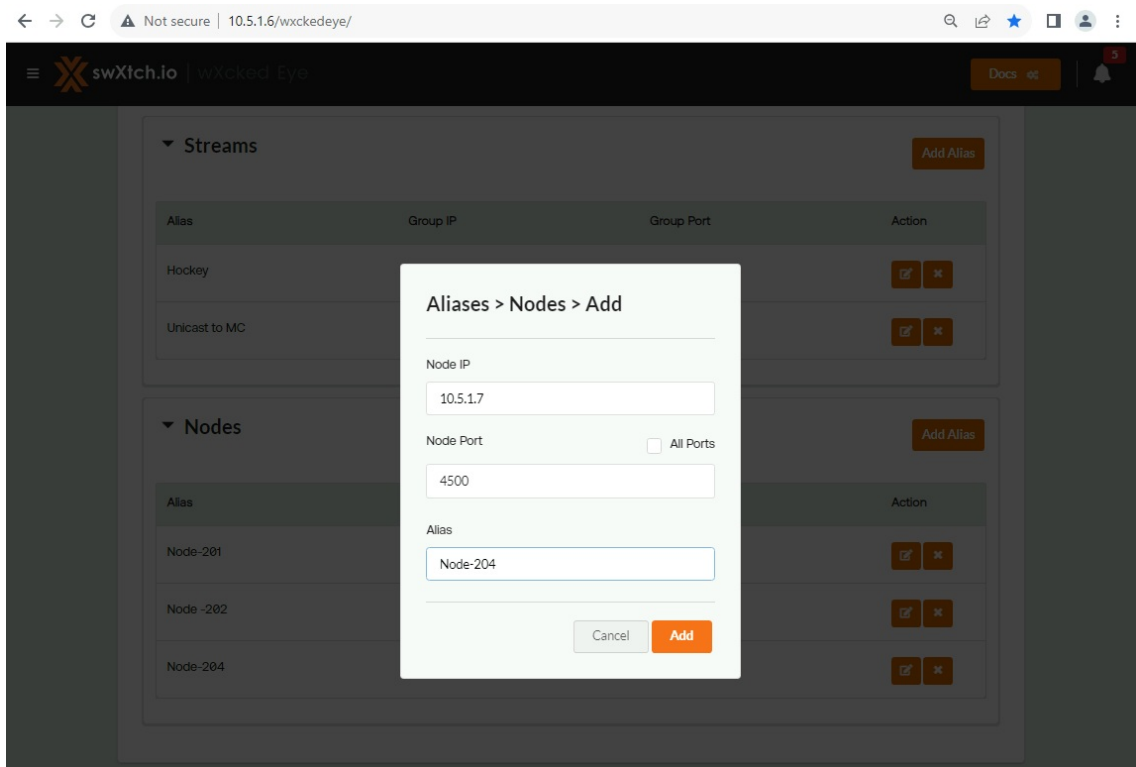
Similar to the Streams function, users can set aliases for their Nodes, avoiding the need to re-write the node IP and port during UDP or SRT mapping.



Alias	Node IP	Node Port	Action
Node-201	10.5.1.4	2000	
Node-202	10.5.1.5	3479	
Node-204	10.5.1.7	4500	

To add a new node:

1. Click on the **Add Alias** button in the right hand corner of the **Nodes** panel. A new window will open.



Aliases > Nodes > Add

Node IP: 10.5.1.7

Node Port: 4500 ☐ All Ports

Alias: Node-204

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 **Save** to confirm.
4. **A new node will be added to the list.**

Users can edit the name of the node or remove it from the list by clicking the buttons under the **Actions** column.

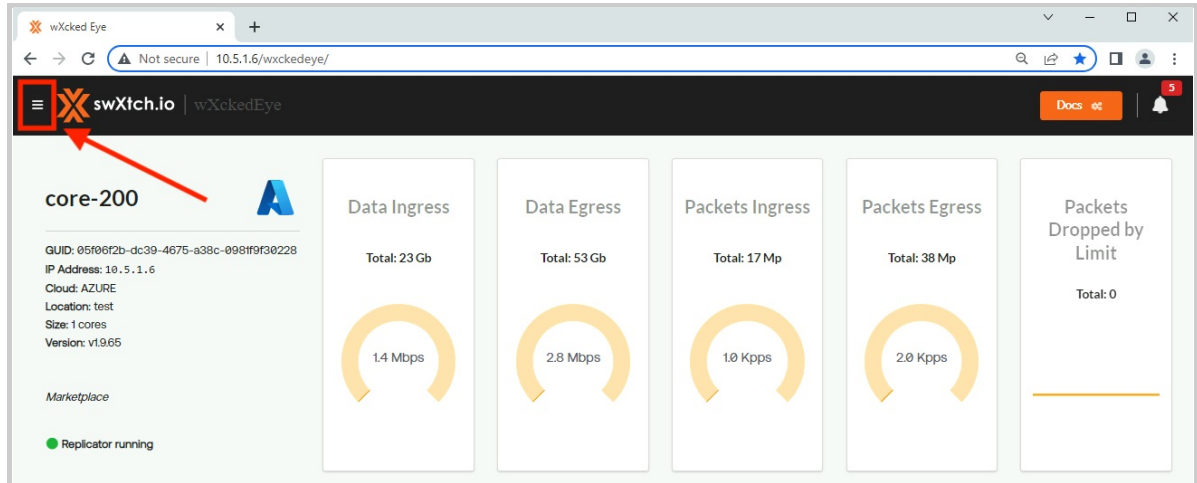
Please note: The edit feature does not let you change the Node IP or Node Port. To do this, delete the node and re-add.

Timing Nodes

Finding the Timing Nodes page

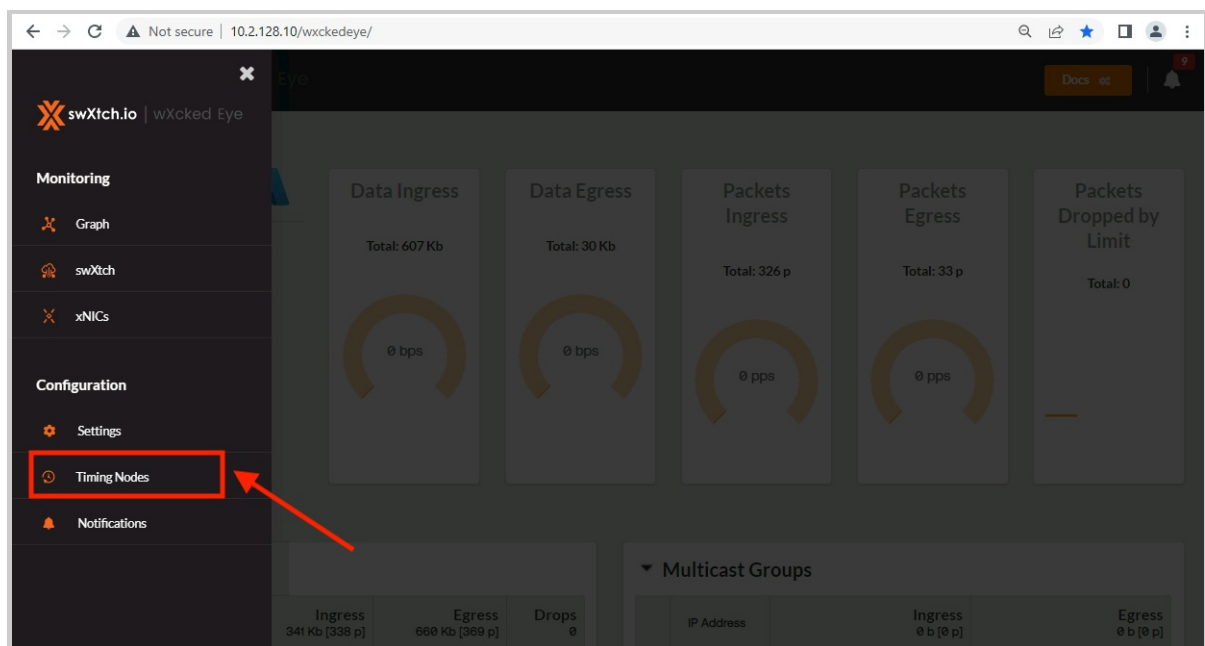
To find the Timing Nodes page in the wXcked Eye UI:

1. Click the three horizontal lines next to the swXtch.io logo.



Clicking on the three horizontal lines by the swXtch.io logo will open the Navigation menu.

2. Select Timing Nodes under Configuration.



Understanding the Timing Nodes page

The screenshot shows the 'Timing Nodes' page in the wXcked Eye interface. The page has a dark header with the swXtch.io logo and a navigation menu. The main content area is light green and contains the following information:

Timing Nodes (Update credentials button)

Master Node

Name: core-100 Time Sync Service: phc:/dev/ptp_hyperv

Follower Nodes (CSV button)

Name	Status	Local Offset	Root Offset
agent-101	Present	3.16 μ s	25.45 μ s
agent-102	Present	2.70 μ s	21.36 μ s
agent-104	Present	14.41 μ s	18.03 μ s
agent-105	Present	2.29 μ s	20.59 μ s
agent-204	Not present	--	--
wint-100	Present	2.88 μ s	15.11 μ s

swXtch.io © 2023 swXtch.io

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.

In the example above, the cloudSwXtch (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 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.

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."**

High Availability

WHAT TO EXPECT

In this article, users will learn how to configure high availability for both their xNIC and cloudSwXtch.

- [For the xNIC](#), users will edit the configuration file for either the producer or consumer and learn how to differentiate between single and multiple multicast groups.
- [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](#).

There are three items that need to be configured when using the High Availability feature in cloudSwXtch:

1. Any virtual machines that are consuming highly available multicast groups.
2. Any virtual machines that are producing highly available streams.
3. cloudSwXtch instances that are in one of the paths of the multicast groups.

The first section will discuss configuring the producers and consumers for xNIC. Each consumer or producer needs to be configured to send or receive multicast high availability traffic.

Please note: If the end consumer can deduplicate the multicast, then xNIC configuration is not required and the xNIC will send/receive both paths to the application.

Configuring xNIC for High Availability

Where To Find The Config File

To configure xNIC for High Availability streams, the **swXtch-xnic.conf** file needs to be updated for both the producer and consumer.

For Linux:

The file can be found in **/var/opt/swxtch/swxtch-xnic.conf**. Currently, only V1 is supported for Linux. To edit the file, one option is to use nano as shown below:

Bash

[Copy](#)

```
sudo nano /var/opt/swxtch/swxtch-xnic.conf
```

For Windows:

V1 - The file can be found in **C:\Program Files\SwXtch.io\Swxtch-xNIC\swXtch-xnic.conf**

V2 -The file can be found in **C:\Program Files\SwXtch.io\Swxtch-xNIC2**

The xNIC needs to be configured to send and receive high availability traffic. Part of this configuration includes defining which cloudSwXtches are to be used. To do this, users can edit their config file similar to the example below. It displays HA configuration for single and multiple multicast IPs within a group.

Bash

[Copy](#)

```
[SWXTCH_1]
SwxtchSvcAddr="10.1.1.1"
SwxtchSvcPort=10802
[SWXTCH_2]
```

```

SwxtchSvcAddr="11.1.1.1"
SwxtchSvcPort=10802

[HA_MULTIPLEGROUP_1]
DestinationGroup="239.2.2.2"
GroupList="239.1.1.1, 239.1.1.2"

[HA_MULTIPLEGROUP_2]
DestinationGroup="239.3.3.3"
GroupList="239.1.1.3, 239.1.1.4"

[HIGH_AVAILABILITY]
MaxTimeToBufferPackets_ms=50
BufferSizeInPackets=131072
Protocol="rtp"

```

A user can have both configurations in one file (Single MC and Multiple MC) or choose one over the other depending on their needs. The examples below will detail the difference between these configurations. It is worth mentioning that these examples are very high level and simplified for the purpose of this document. For questions on more complex configurations, please contact support@swXtch.io.

REMINDER: The xNIC must be restarted for changes to take place

For Windows:

Go to the Task Manager and restart the swXtchNIC service.

For Linux:

To restart:

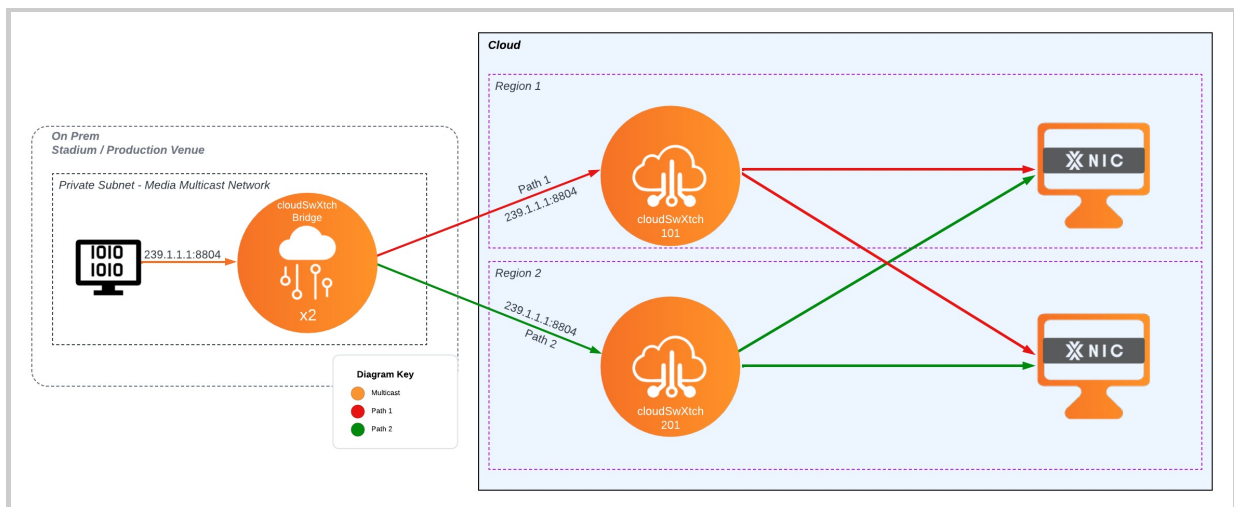
Bash

[Copy](#)

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

Single MC

The example configurations for the xNIC for Producer and Consumer are based on the simple picture below:



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. A sample file for this configuration is detailed below:

Bash	Copy
<pre> [SWXTCH_1] SwxtchSvcAddr="10.1.1.1" SwxtchSvcPort=10802 [SWXTCH_2] SwxtchSvcAddr="11.1.1.1" SwxtchSvcPort=10802 [HIGH_AVAILABILITY] MaxTimeToBufferPackets_ms=50 BufferSizeInPackets=131072 Protocol="rtp" </pre>	

Depending on if this file is for a producer or consumer, [SWXTCH_1] and [SWXTCH_2] is telling the xNIC to send/receive all packets via cloudSwXtches 10.1.1.1 and 11.1.1.1. Note that [SWXTCH_1] will already be in the file since it is created during xNIC install. You can have up to 8 paths total. This part is **required** for all high availability configurations.

The [HIGH_AVAILABILITY] section must be added to the consumer's config file. The protocol here can be set to "default" or "rtp." This section is consumer specific; if added to the producer config file, nothing will happen.

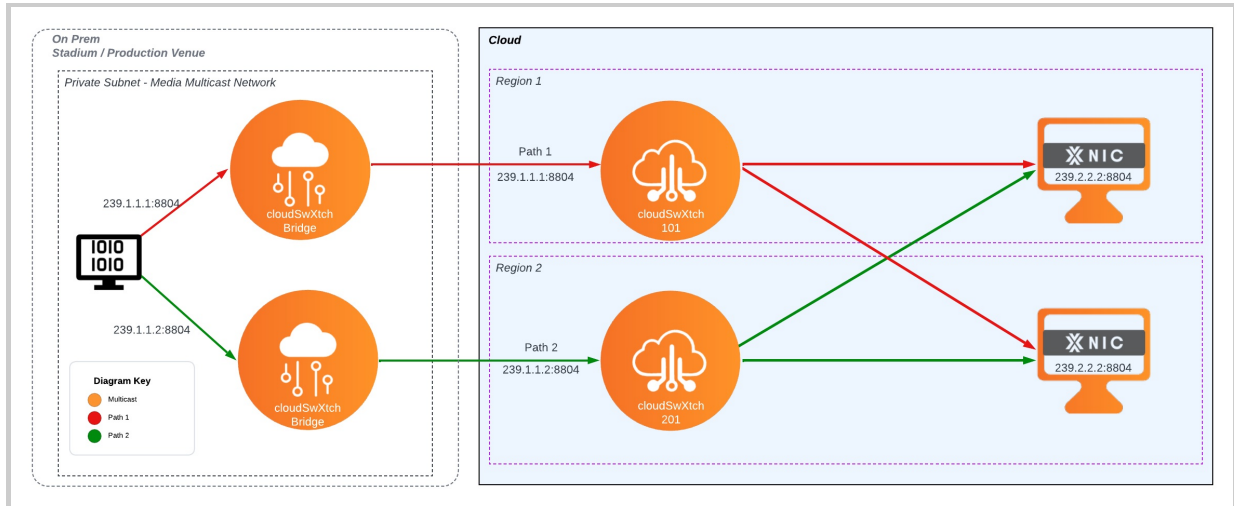
[HIGH AVAILABILITY] Explained

The [HIGH_AVAILABILITY] 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 default or RTP.
 - *Default* = This should be used when the producer is an xNIC. 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.

Multiple MC (mMC-7)

Alternatively, an xNIC can be configured for multiple multicast IPs within a group as illustrated below. This configuration will be referred to as Multiple MC or mMC-7.



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 file for this configuration is detailed below:

Bash	Copy
<pre>[SWXTCH_1] SwxtchSvcAddr="10.1.1.1" SwxtchSvcPort=10802 [SWXTCH_2] SwxtchSvcAddr="11.1.1.1" SwxtchSvcPort=10802 [HA_MULTIPLEGROUP_1] DestinationGroup="239.2.2.2" GroupList="239.1.1.1, 239.1.1.2" [HA_MULTIPLEGROUP_2] DestinationGroup="239.3.3.3" GroupList="239.1.1.3, 239.1.1.4" [HIGH_AVAILABILITY] MaxTimeToBufferPackets_ms=50 BufferSizeInPackets=131072 Protocol="rtp"</pre>	

Similar to the Single MC configuration, we have both [SWXTCH_1] and [SWXTCH_2] telling the xNIC, which cloudSwXtches to send or receive packets from. Note that [SWXTCH_1] will already be in the file since it is created during xNIC install. In total, you can have up to 8 paths. This part is **required** for all high availability configurations.

For the consumer's config file, the [HIGH_AVAILABILITY] section must be added. For **multiple multicast groups**, the protocol must be set to "rtp".

What differs from the Single MC configuration is the section with [HA_MULTIPLEGROUP_1]. Here, the user is grouping together 2 multicast IPs (239.1.1.1 and 239.1.1.2) and assigning it a DestinationGroup (239.2.2.2). The application at the end point 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 is illustrated in the diagram above.

Configuring cloudSwXtch for High Availability

The cloudSwXtch must be configured to for High Availability for the system to know the paths, additionally it will allow naming of Paths as well as enable the HA views in swxtch-top. This section will use the diagram above to create the two paths shown in the diagram.

Users can also configure High Availability for the cloudSwXtch in wXcked Eye. For more information, see [High Availability with wXcked Eye](#).

HA Help

To get a list of the HA commands use the -h or --help as shown below"

None	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 remove-swxtch Remove Swxtch from 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: 80 will be used. **Do not use port 10802** (the one used in the config file), as it is intended for xNIC communication only, therefore it will not work for swx commands.

HA Create

To create or update a HA cluster, a HAconfig.json file must exist note that the IP is for the control plane. The following shows the format:

```
{
```

```

"name": "MY High Availability",
"paths": [
  {
    "name": "Path 1",
    "swxtches": [
      "10.2.128.10"
    ]
  },
  {
    "name": "Path 2",
    "swxtches": [
      "10.5.1.6"
    ]
  }
]
}

```

If there are multiple cloudSwXtch(s) in a path then the last cloudSwXtch in the path will have the IP address in the configuration. The rest of the cloudSwXtches do not need to be listed.

The -h or --help commands will show the syntax for the “swx ha create” command.

None	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 which is part of the HA flow the “ha destroy” command can be used.

The -h or --help commands will show the syntax for the “swx ha destroy” command.

None	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:

None	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:

None	Copy
<pre> PS C:\Users\testadmin> swx ha destroy -s cloudswxtch101 Validating cluster deletion. Successfully deleted the cluster.</pre>	

HA Remove-swxtch

To remove a cloudSwXtch from a High Availability cluster, which is part of the HA flow, the " ha remove-swxtch" command can be used.

None	Copy
<pre> PS C:\Users\testadmin> swx ha remove-swxtch -h Remove Swxtch from the HA cluster Usage: swx ha remove-swxtch [flags] Flags: -h, --help help for remove-swxtch 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 remove the cloudSwXtch:

None	Copy
<pre> PS C:\Users\testadmin> swx ha remove-swxtch -s <swxtch name or control data ip of the cloudSwXtch you wish to remove></pre>	

Example:

None	Copy
<pre>PS C:\Users\testadmin> swx ha remove-swxtch -s cloudSwXtch101 Validating that the Swxtch left the cluster. Successfully left cluster. PS C:\Users\testadmin></pre>	

HA Show

To get a list of cloudSwXtch(s), which are part of the HA flow, the “ha show” command can be used.

The -h or –help commands will show the syntax for the “swx ha show” command.

None	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:

None	Copy
<pre>swx ha show -s <swxtch name or control data IP of a cloudSwXtch in the HA configuration></pre>	

Example below:

None	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"] }], }, }</pre>	


```

    {
      "name": "Path2",
      "swxtches": [
        "10.5.1.6"
      ]
    }
  ]
}
}

```

swxtch-top has options for High Availability - see [swxtch-top](#)

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**

None	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

None	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

None	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.

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> <p>OUTPUT: Validating mesh.. Mesh succesfully created.</p>	

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> <p>Validating that the swxtch was added. Swxtch successfully added to the mesh.</p>	

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> <p>Validating that the swxtch was removed.</p> <p>Swxtch successfully removed from the mesh.</p>	

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> <p>Validating that the mesh was destroyed.</p>	

Mesh successfully destroyed.

Bridge Type 2

Configuring Bridge Type 2 Interfaces

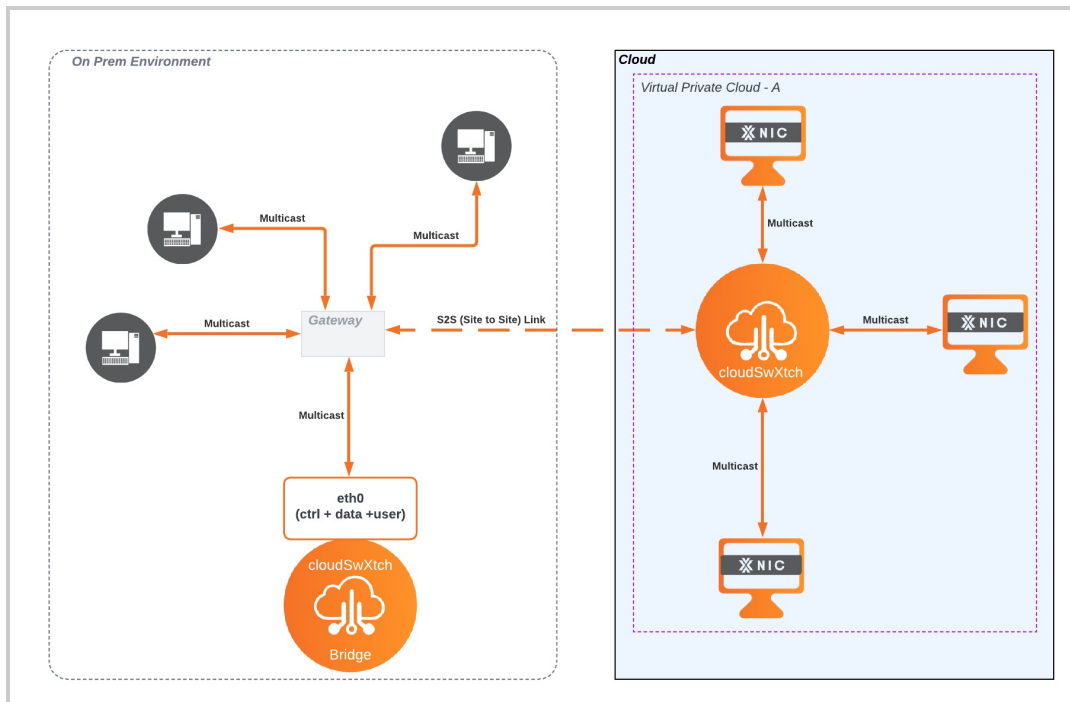
By default, Bridge Type 2 installation will attempt to resolve the interface the is routable to the cloudSwXtch. However, if a user would like to do this manually, the cloudSwXtch Bridge Type 2 can be configured in one of two ways:

- 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 Bridge Type 2 JSON configuration file to apply the above methods. The location of the configuration file is `/var/opt/swxtch/swxtch-bridge.json`.

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": [], "cloudToGroundSubscriptions": ["225.0.23.182:12000", "225.0.23.183:12000", "225.0.23.184:12000", "225.0.23.185:12000"], "pollingIntervalMilliseconds": 1000 } }</pre>	

For a single interface



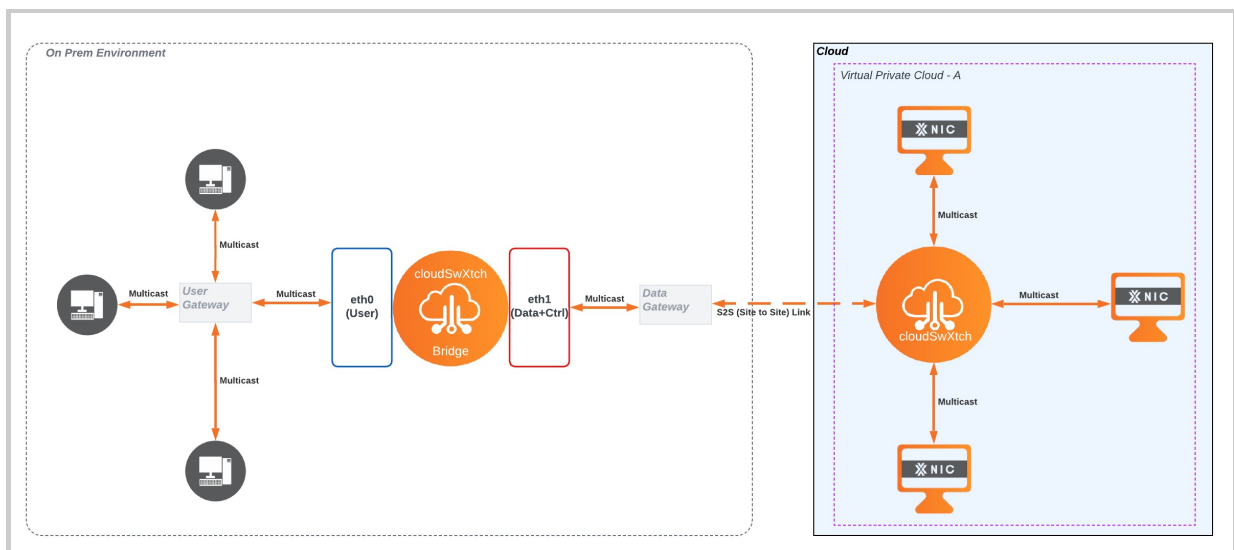
To accomplish a single interface configuration for your cloudSwXtch Bridge Type 2, users will need to specify the same InterfaceName for Ctrl, Data and User in the swxtch-bridge.json file. In the example, each are assigned to eth0.

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
<pre> "ctrlInterfaceName": "eth1", "dataInterfaceName": "eth1", "userInterfaceName": "eth0", </pre>	

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
<pre> curl -X POST http://<BRIDGE_CTRL_IP>/addCloudToGround -d '{"MulticastGroups": ["239.239.239.99"],"UpdateConfigFile":false}' </pre>	

To Leave:

Bash	Copy
<pre> curl -X POST http://<BRIDGE_CTRL_IP>/removeCloudToGround -d '{"MulticastGroups": ["239.239.239.99"],"UpdateConfigFile":false}' </pre>	

Configuring Bridge Type 2 Static Subscriptions

The cloud to ground and cloud to ground 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", </pre>	

```

        "225.0.23.184:12000",
        "225.0.23.185:12000"],
        "pollingIntervalMilliseconds": 1000
    }
}

```

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

Bash	Copy
<pre>sudo systemctl restart swtch-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>	

Using a specific source IP for Bridge Type 2

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.

To accomplish this, add the following parameter into the JSON configuration file with IP address to use for the source:

Bash	Copy
<pre>"overwriteSenderIp": "172.30.1.1",</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
<pre>--input multicast://<multicast-group-ip>:<nic-ip>:<multicast-group-port></pre>	

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

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

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
<pre>--output udp://<swxtch-data-ip>:9999</pre>	

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
<pre>--override-multicast-sender-ip <bridge-source-address> --last-leg</pre>	

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

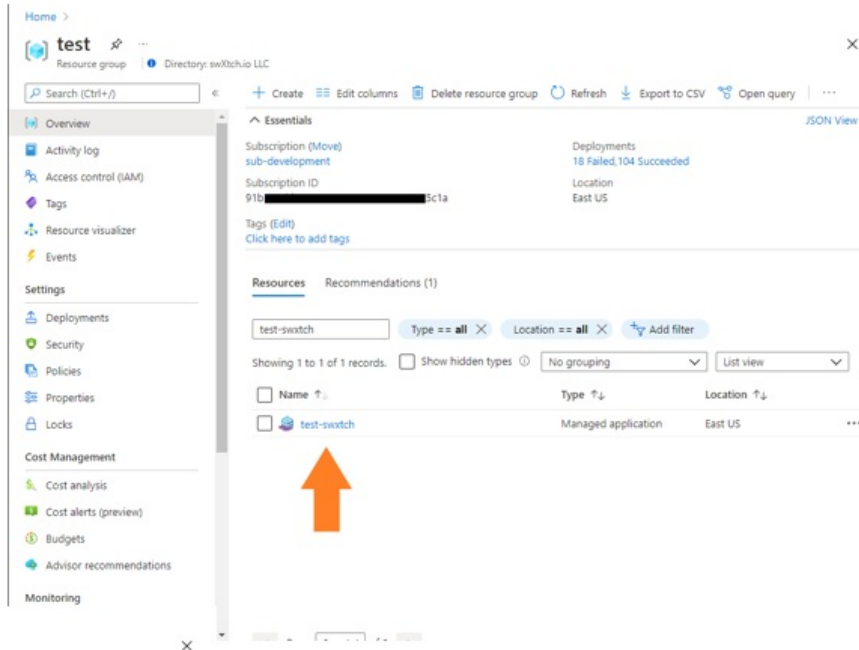
There are two options for configuring Protocol Conversion and Fanout: via wXcked Eye or via the API. For more information, please see the following articles:

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

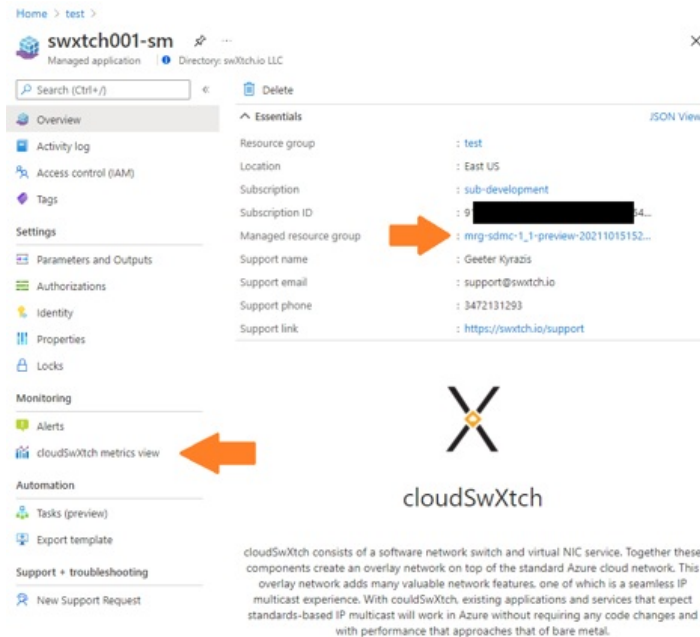
Azure Monitoring

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.

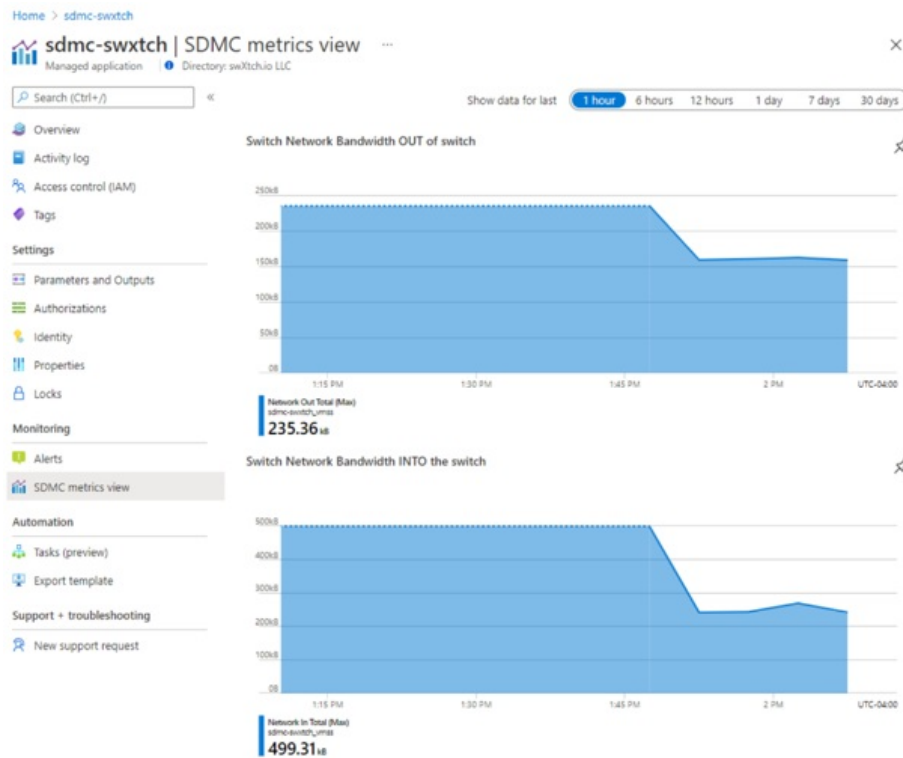


cloudSwXtch metrics view

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 similar to what's detailed below (cut is not full output, but just a sampling):

Plaintext	Copy
<pre>swxtch_xnics_dsd_win11_100_txMulticastGroups_0_pktsCount 2.020608e+06 # HELP swxtch_xnics_dsd_win11_100_txMulticastGroups_0_protocolType Protocol type # TYPE swxtch_xnics_dsd_win11_100_txMulticastGroups_0_protocolType counter swxtch_xnics_dsd_win11_100_txMulticastGroups_0_protocolType 0 # HELP swxtch_xnics_dsd_win11_100_txMulticastGroups_0_srcPort Source port # TYPE swxtch_xnics_dsd_win11_100_txMulticastGroups_0_srcPort counter swxtch_xnics_dsd_win11_100_txMulticastGroups_0_srcPort 0 # HELP swxtch_xnics_dsd_win11_100_xNicVersion xNIC version # TYPE swxtch_xnics_dsd_win11_100_xNicVersion counter swxtch_xnics_dsd_win11_100_xNicVersion 203426</pre>	

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.

- 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.

- 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
```

- Run the following docker command to run it in VM. If you decide to run it this way, you will need to run it after every reboot or when you close your window. Please use this method when testing in order to limit the amount of records added to the Prometheus database.

Plaintext

Copy

```
docker run --network host -v ~/prometheus:/etc/prometheus prom/prometheus
```

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

Plaintext

Copy

```
docker run --network host --restart always -v ~/prometheus:/etc/prometheus
prom/prometheus
```

STEP THREE: Access Prometheus UI

In order to access the Prometheus UI, users should open a browser on their Windows machine in the same VNET and enter the following URL:

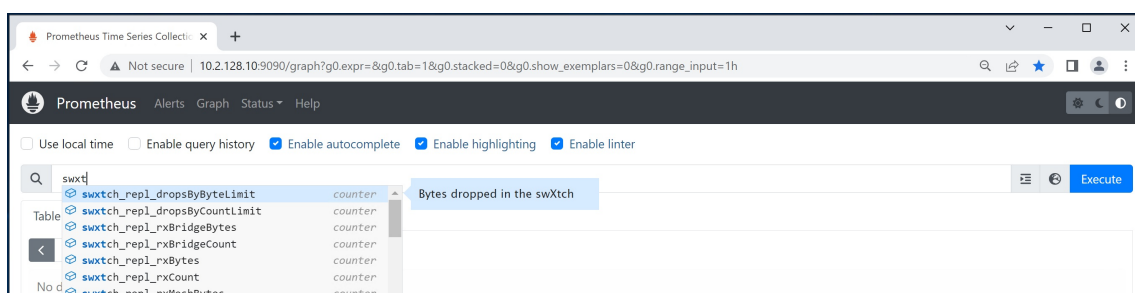
Plaintext

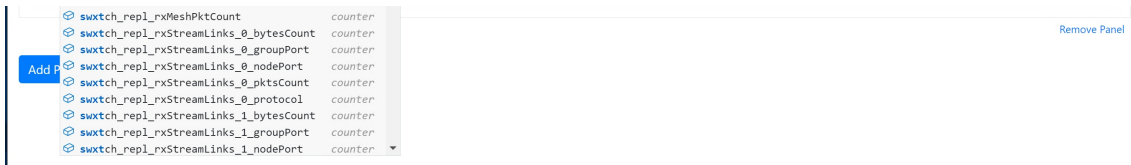
Copy

```
http://<prometheus-IP>:9090/
```

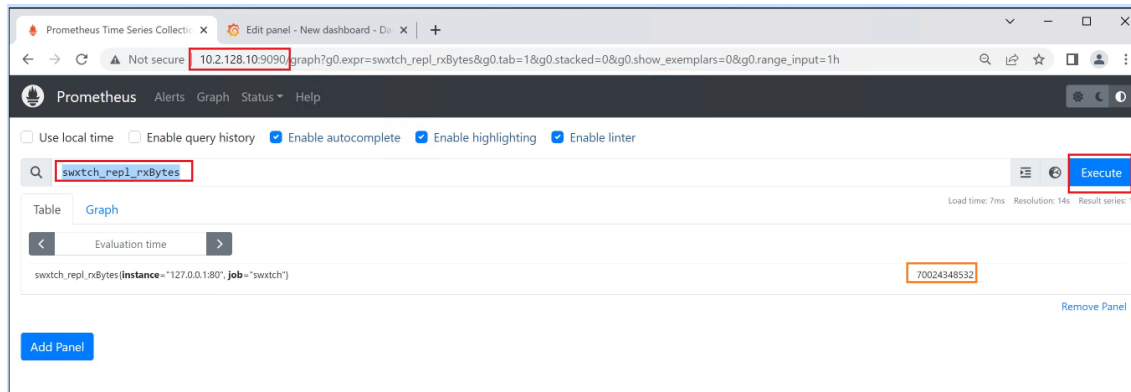
Note: Replace the <prometheus-IP> with the IP address of your Prometheus instance.

Users can enter the prefix “swxtch” into the search field to see a list of data fields related to the cloudSwXtch and its xNICs.





In the example below, the user has chosen `swtch_repl_rxBytes` 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.

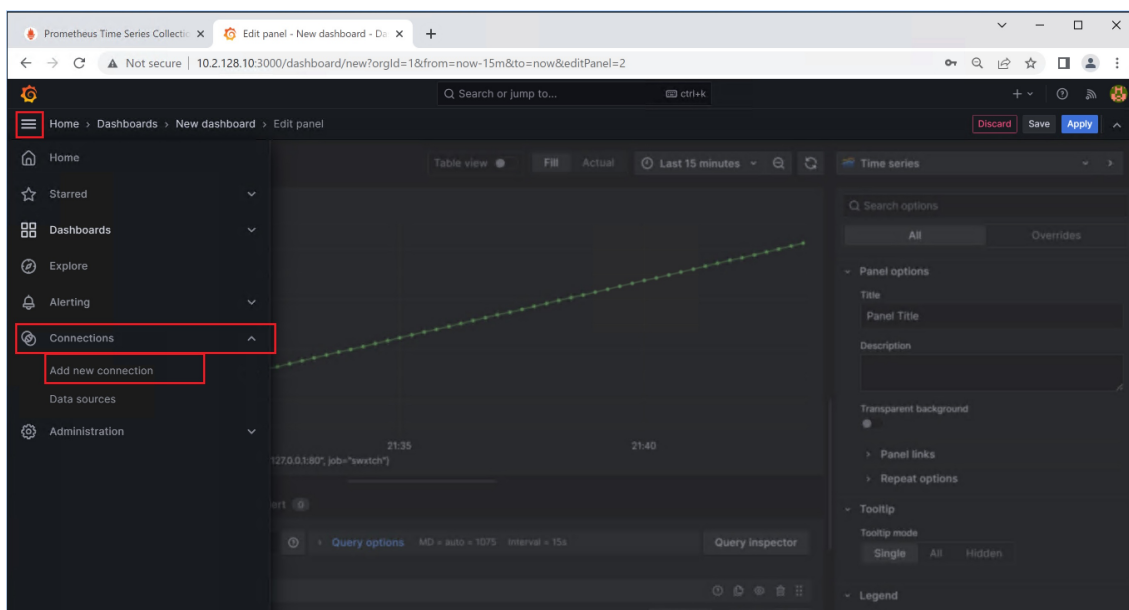
1. In a Windows machine on the same VNET, open a browser and enter the following URL:

Plaintext	Copy
http://<Grafana-IP>:3000/	

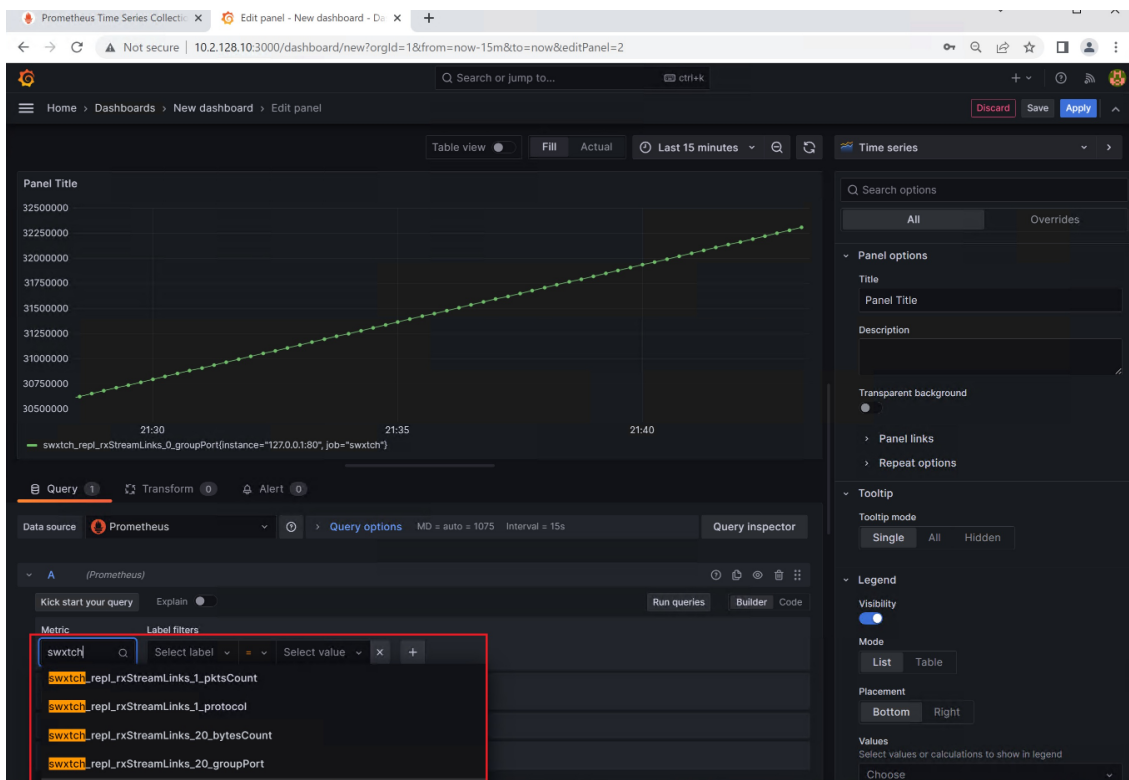
Note: Replace the <Grafana-IP> with the IP address of your Grafana instance.

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.

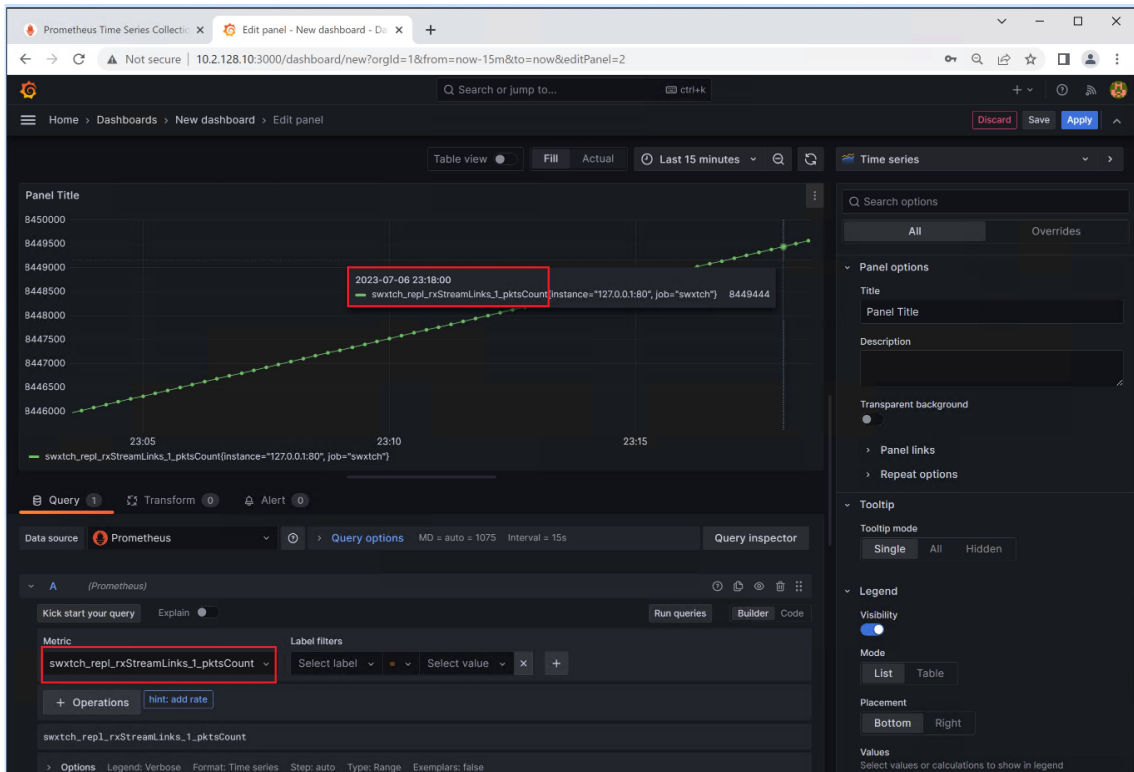


6. In the search bar under **Metric**, use the prefix “swtch” to populate a list of potential data fields.

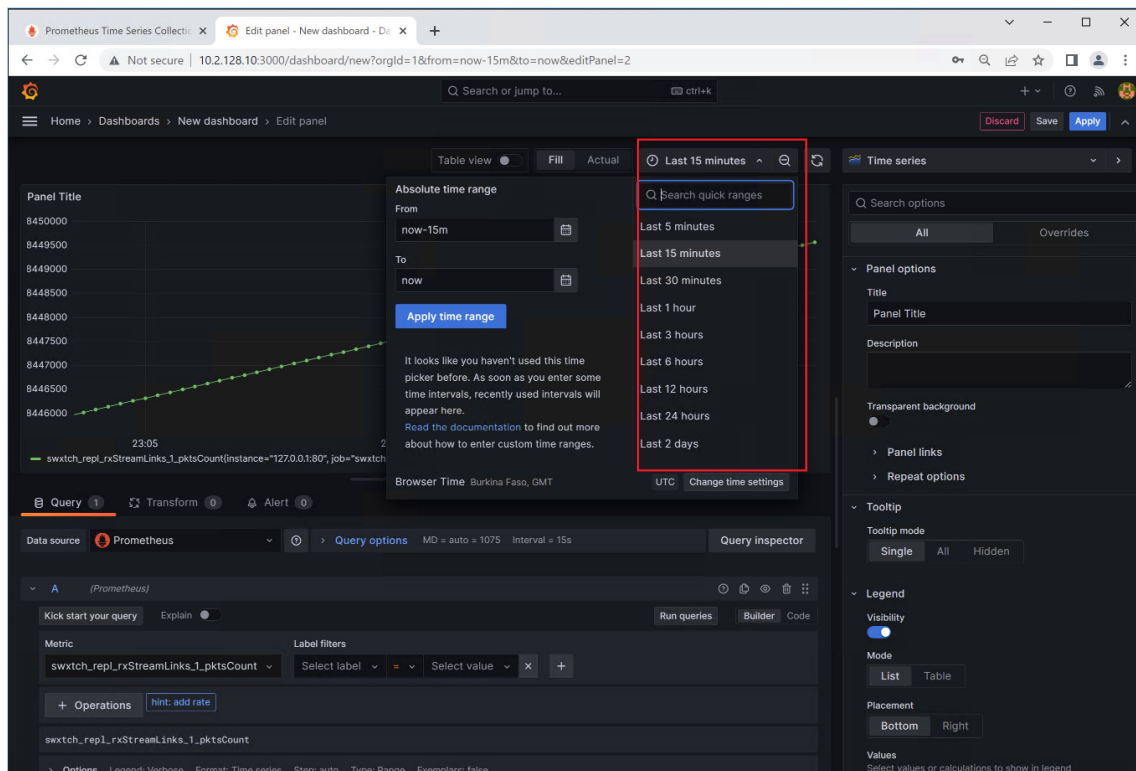


7. Scroll through the list and select what is desired.

8. Run the query. The results will appear in the top.



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

swtch-top

WHAT TO EXPECT

swtch-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 connected to mesh, high availability, multicast and PTP.

In this article, you will learn how to navigate through the different pages in swtch-top and get better visibility on how data flows in your cloudSwXtch instance.

Running swtch-top

Depending on your operating system, you can use certain commands to run swtch-top on your VM.

For both Windows and Linux agents, users can enter the following into the terminal:

None	Copy
swtch-top dashboard	

From the cloudSwXtch, users can enter the following command:

None	Copy
sudo /swtch/swtch-top dashboard --switch localhost	

Navigating swtch-top Dashboard

Administrator: Windows PowerShell

Information - dev

1

2

3

1: xNIC view 2: Streams 3: Mesh view 4: HA Paths 5: HA Summary 6: HA view 7: Configuration view 8: Timing Nodes view

```
dsd-core-100 v2.0.10(Dsd-100-50clients-1_21_2031) Auth. Type License File
SubscriptionId b10209ad-ad22-4c26-8aef-be93b2f0bb58 Max Bandwidth 32000 Mbps
ResourceGroupName TEST-DONNA Max Clients 50
SwxtchId 2369a922-410a-40bf-8e6e-630efe0ee70a
Status OK

Totals
Producers 473 pps (5.3M bps) Consumers 516 pps (5.6M bps)
Bridge RX 0 pps (0 bps) Bridge TX 0 pps (0 bps)
HA RX 250 pps (2.8M bps) HA TX 395 pps (4.3M bps)
Switch RX 639 pps (7.1M bps) Switch TX 1.3K pps (14.2M bps)

xNIC clients
Name ip Version (XNIC) RX pps RX bps TX pps TX bps HA
Dsd-agent-101 10.2.192.5 v2.0.10 (v2) 282 3.0M 0 0 2
Dsd-agent-102 10.2.192.11 v2.0.10 (v2) 0 0 0 0 N
Dsd-agent-104 10.2.192.82 v2.0.10 (v2) 0 0 473 5.3M 2 (-7)
Dsd-agent-105 10.2.192.15 v2.0.10 (v2) 235 2.6M 0 0 2 (-7)
Dsd-agent-201 10.5.2.4 v2.0.10 (v1) 0 0 0 0 2 (-7)
Dsd-agent-204 10.5.2.7 v2.0.10 (v2) 0 0 0 0 2 (-7)
aks-nodepool1-23164585-vmss000002 10.2.128.87 v2.0.10 (v2) 0 0 0 0 N
aks-nodepool1-23164585-vmss000003 10.2.128.95 v2.0.10 (v2) 0 0 0 0 N
```

The swtch-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 swtch-top dashboard has 8 different views:

1. xNIC
2. Streams

3. Mesh
4. HA Paths
5. HA Summary
6. HA
7. Configuration
8. Timing Nodes

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 "2: Flows," enter in the number 2 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 read the following article:

- [Upgrade cloudSwXtch on Azure](#)
- [Upgrade cloudSwXtch on AWS](#)

Panel 1: Information

The first panel of the swtch-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				
core-200	v1.9.77(dsd-200-core-azure-April-5-2033)	Auth. Type	License File	Marketplace
SubscriptionId	b10209ad-ad22-4c26-	Max Bandwidth	2000 Mbps	
ResourceGroupName	test	Max Clients	10	
SwxtchId	05f06f2b-dc39-4675-a38c-			
Status	OK			

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 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 associated with that plan. For more information regarding licensing, please read the [cloudSwXtch Pricing](#) article.

AWS

Information - v1.9.76				
ip-172-41-129-113	v1.9.76(large)	Auth. Type	Cloud Marketplace	
AccountId	6397206	Max Bandwidth	unlimited	
Region	us-west-2	Max Clients	unlimited	
SwxtchId	i-0183b5f1e96f			
Status	OK			

On the left side of the section, users will be able to read the name given to their cloudSwXtch with the version and the subscription tier. 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 associated with that plan. For more information regarding licensing, please read the [cloudSwXtch Pricing](#) article.

Panel 2: Totals

Panel 2 breaks down the statistics regarding data flow to and from the cloudSwXtch. Both the ingress and egress bandwidth will be displayed in both **packets per seconds (pps)** and **bits per second (bps)**.

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

- **Producers** - The total egress for all producers connected to the cloudSwXtch.
- **Consumers** - The total ingress for all consumers connected to the cloudSwXtch.
- **Bridge RX** - The total egress for the bridge that is connected to the cloudSwXtch (Ground-->Cloud).
- **Bridge TX** - The total ingress for the bridge that is connected to the cloudSwXtch (Cloud-->Ground).
- **Mesh RX** - The total egress for the entire Mesh of cloudSwXtches.
- **Mesh TX** - The total ingress for the entire Mesh of cloudSwXtches.
- **Switch RX** - The total ingress that the cloudSwXtch is receiving.
- **Switch 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

xNIC Clients								
Name	ip	Version (xNIC)	RX pps	RX bps	TX pps	TX bps	HA	
Dsd-agent-101	10.2.192.5	v2.0.10 (v2)	281	2.9M	0	0	2	
Dsd-agent-102	10.2.192.11	v2.0.10 (v2)	0	0	0	0	N	
Dsd-agent-104	10.2.192.82	v2.0.10 (v2)	0	0	439	4.9M	2 (-7)	
Dsd-agent-105	10.2.192.15	v2.0.10 (v2)	242	2.7M	0	0	2 (-7)	
Dsd-agent-201	10.5.2.4	v2.0.10 (v1)	208	2.3M	0	0	2 (-7)	
Dsd-agent-204	10.5.2.7	v2.0.10 (v2)	248	2.8M	0	0	2 (-7)	
aks-nodepool1-23164585-vmss000002	10.2.128.87	v2.0.10 (v2)	0	0	0	0	N	
aks-nodepool1-23164585-vmss000003	10.2.128.95	v2.0.10 (v2)	0	0	0	0	N	

1: xNIC view 2: Streams 3: Mesh view 4: HA Paths 5: HA Summary 6: HA view 7: Configuration view 8: Timing Nodes view

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: xNIC1 (V1) or xNIC2 (V2)
- **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: Flows

Stream Stats									
Stream	MC Group	IP:Port	Protocol	Node	Node IP:Port	RX pps	RX bps	TX pps	p...
239.1.1.4:8804	239.1.1.4:8804		MC	10.2.192.82:37874	10.2.192.82:37874	436	4.9M	423	7...
MC from SRT	225.1.1.1:1599		SRT(C)	dsd-104	10.2.128.75:1599	136	1.4M	136	4...
239.1.1.4:8805	239.1.1.4:8805		MC	10.2.192.82:38130	10.2.192.82:38130	0	0	0	...

1: xNIC view 2: Streams 3: Mesh view 4: HA Paths 5: HA Summary 6: HA view 7: Configuration view 8: Timing Nodes view

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 view 2: Streams 3: Mesh view 4: HA Paths 5: HA Summary 6: HA view 7: Configuration view 8: Timing Nodes view

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

- **SwitchAddress** - 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

HA Paths	
Name	Paths
path1	10.2.128.10
path2	10.5.1.6

1: xNIC view 2: Streams 3: Mesh view 4: HA Paths 5: HA Summary 6: HA view 7: Configuration view 8: Timing Nodes view

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

HA Summary						
Agent	Paths	Path Ingress pps	Path Ingress bps	Path Usage %	Missing Packets	
DSd-agent-104	path1	0	0	0.00	0	
	path2	0	0	0.00	2	
	Reconstructed Path	0	0		0	
DSd-agent-105	path1	752	8.3M	2.01	2002358711601	
	path2	481	5.4M	97.99	2002358718320	
	Reconstructed Path	246	2.7M		504796703896	
DSd-agent-201	path1	503	5.5M	0.00	0	
	path2	503	5.5M	100.00	1788	
	Reconstructed Path	252	2.7M		0	
DSd-agent-204	path1	0	0	0.00	0	
	path2	0	0	0.00	1788	
	Reconstructed Path	0	0		0	

1: xNIC view 2: Streams 3: Mesh view 4: HA Paths 5: HA Summary 6: HA view 7: Configuration view 8: Timing Nodes view

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 View

HA Streams									
Agent	Stream Src IP	Stream IP	Paths	Path Ingress pps	Path Ingress bps	Path Usage %	Missing Packets		
DSd-agent-105	10.2.128.75	225.1.1.1	path1	279	2.9M	0.00	2049538995469		
			path2	0	0	0.00	2049539000589		
			Reconstructed Path	0	0		500769909733		
DSd-agent-105	10.2.192.82	239.1.1.4	path1	530	5.9M	0.74	1		
			path2	530	5.9M	99.26	1788		
			Reconstructed Path	265	3.0M		0		
DSd-agent-201	10.2.192.82	239.1.1.4	path1	533	5.8M	0.00	0		
			path2	533	5.8M	100.00	1788		
			Reconstructed Path	267	2.9M		0		

1: xNIC view

2: Streams

3: Mesh view

4: HA Paths

5: HA Summary

6: HA view

7: Configuration view

8: Timing Nodes view

The HA streams 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

```

-----Configurations-----
Entitlements
=====
Max Bandwidth 32000 Mbps
Max Clients    50
Mesh           true
HA             true
Fanout         true
ClockSync      true
Bridge         true
WxckedEye      true

Mesh
=====
swXtches      10.2.128.10, 10.5.1.6, 10.1.1.6
HA            path1: {10.2.128.10}, path2: {10.5.1.6}
Unicast Udp
=====
Base Group Ip 239.1.1.1
Group Port    5000
Port Range    [4000, 4003]

1: xNIC view 2: Streams 3: Mesh view 4: HA Paths 5: HA Summary 6: HA view 7: Configuration view 8: Timing Nodes view

```

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 to their mesh and HA configurations as well as details on their unicast.

- **Entitlements:** Depending on their tier (Small, Medium and Large), users will have a set number for their Max Bandwidth and Max Clients. In the example above, the user has a max bandwidth of 2 GBs with 10 clients max. This section will also show if a user has the following features enabled: Mesh, HA, Fanout, Clock Sync (PTP), Bridge, and wxckedEye.
- **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.
- **Unicast:** This will list the unicast's Base Address and Port Range that are configured for Protocol Fanout.

8. Timing Nodes view

```

-----Timing Nodes-----

Master Node
=====
Name dsd-core-100   Time Sync Service phc:/dev/ptp_hyperv

Follower Nodes
=====
Name      Status      Local Offset      Root Offset
Dsd-agent-101 Present      2.69 µs          26.02 µs
Dsd-agent-102 Present      1.92 µs          17.91 µs
Dsd-agent-104 Present      4.32 µs          18.06 µs
Dsd-agent-105 Present      3.83 µs          51.06 µs
Dsd-agent-201 Not Present  --              --
Dsd-agent-204 Not Present  --              --

1: xNIC view 2: Streams 3: Mesh view 4: HA Paths 5: HA Summary 6: HA view 7: Configuration view 8: Timing Nodes view

```

The Timing Nodesview 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.

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 swtch-top:
 1. Navigate to the swtch-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 swtch-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 swtch-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 Swtch-tun0 if xNIC1. An example is below:

```
$ sudo tcpdump udp -X -i <interface>
```

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 cloudSwtch

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 --switch localhost
```

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

Timing Nodes				
Master Node				
=====				
Name	dsd-core-100	Time Sync Service	phc://dev/ptp_hyperv	
Follower Nodes				
=====				
Name	Status	Local Offset	Root Offset	
DSd-agent-101	Present	2.69 µs	26.02 µs	
DSd-agent-102	Present	1.92 µs	17.91 µs	
DSd-agent-104	Present	4.32 µs	18.06 µs	
DSd-agent-105	Present	3.83 µs	51.06 µs	
DSd-agent-201	Not Present	--	--	
DSd-agent-204	Not Present	--	--	

1: xNIC view 2: Streams 3: Mesh view 4: HA Paths 5: HA Summary 6: HA view 7: Configuration view 8: Timing Nodes view

swxtch-tcpdump

Users can use a special version of tcpdump that helps with capturing multicast packets sent to and from the cloudSwXtch. It is the same as the normal tcpdump but with logic to decode our own header and display the original MC payload.

To accomplish this, execute the following command:

Bash	Copy
<pre>swxtch-tcpdump</pre>	

Arguments for tcpdump will work for swXtch-tcpdump.

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 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 --switch 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, 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 set MTU size

In some cases the MTU Size of the multicast group may exceed the 1500 set limit in Windows and Linux virtual machines. This article will explain how to increase the MTU size if this should occur.

To know if the MTU size has been exceeded Wireshark or tcpdump can be used. Below is an example from Wireshark.

Time	Source	Destination	Protocol	Length	Info
1 0.000000	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1449 > IP PAYLOAD LENGTH] Len=1441
2 0.000000	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1450 > IP PAYLOAD LENGTH] Len=1442
3 0.000000	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1450 > IP PAYLOAD LENGTH] Len=1442
4 0.000000	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1450 > IP PAYLOAD LENGTH] Len=1442
5 0.000203	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1450 > IP PAYLOAD LENGTH] Len=1442
6 0.000203	172.30.0.11	239.192.10.160	UDP	1482	60000 → 4002 [BAD UDP LENGTH 1450 > IP PAYLOAD LENGTH] Len=1442

your title goes here

your content goes here

:::Note: The UDP length error shows it is exceeding the Length.

Linux update MTU Size:

First check MTU current size by running the following command:

None

Copy

```
ifconfig | grep mtu
```

Example:

None

Copy

```
someadmin@my-agent-101:~$ ifconfig | grep mtu
enP43852s1: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST> mtu 1500
enP4589s2: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST> mtu 1500
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
```

Note: The MTU size of eth1 = 1500

To change MTU size to 2000 for example - use the command below:

None

Copy

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

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

None

Copy

```
ifconfig | grep mtu
```

Example:

[None](#)[Copy](#)

```
someadmin@my-agent-101:~$ ifconfig | grep mtu
enP43852s1: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST> mtu 1500
enP4589s2: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST> mtu 2000
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 2000
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
```

Note: The MTU size of eth1 is now = 2000

Windows Update MTU Size

1. Check MTU Size by running this command:

[None](#)[Copy](#)

```
netsh interface ipv4 show subinterfaces
```

You will see a list of network interfaces.

2. Set the MTU Size (in this case to 2000) using the following commands:

[None](#)[Copy](#)

```
netsh
```

[None](#)[Copy](#)

```
interface
```

[None](#)[Copy](#)

```
ipv4
```

and finally to 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 swtch-tun:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : qw41eorzzsjedntce414ja
    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  . : qw41eorzzsjedntce414jaktbh.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>set subinterface "Ethernet 2" mtu=1200 store=persistent
Ok.

netsh interface ipv4>
```

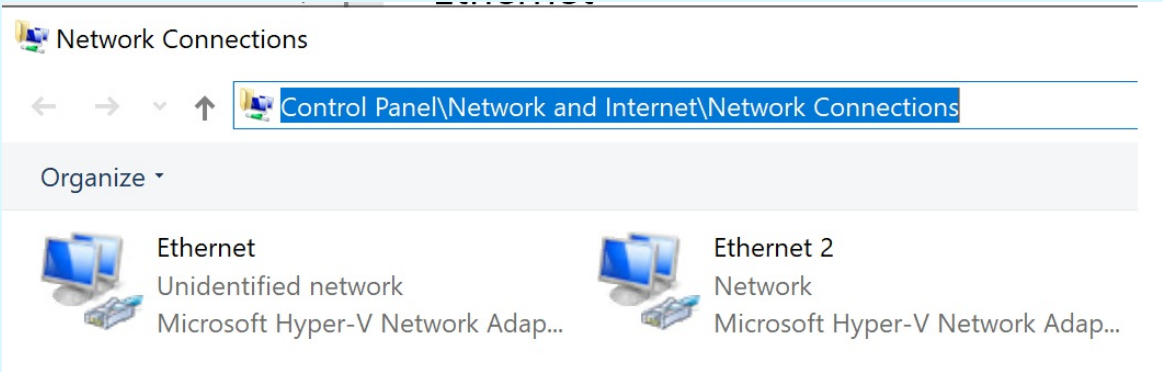
{height="" width=""}

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

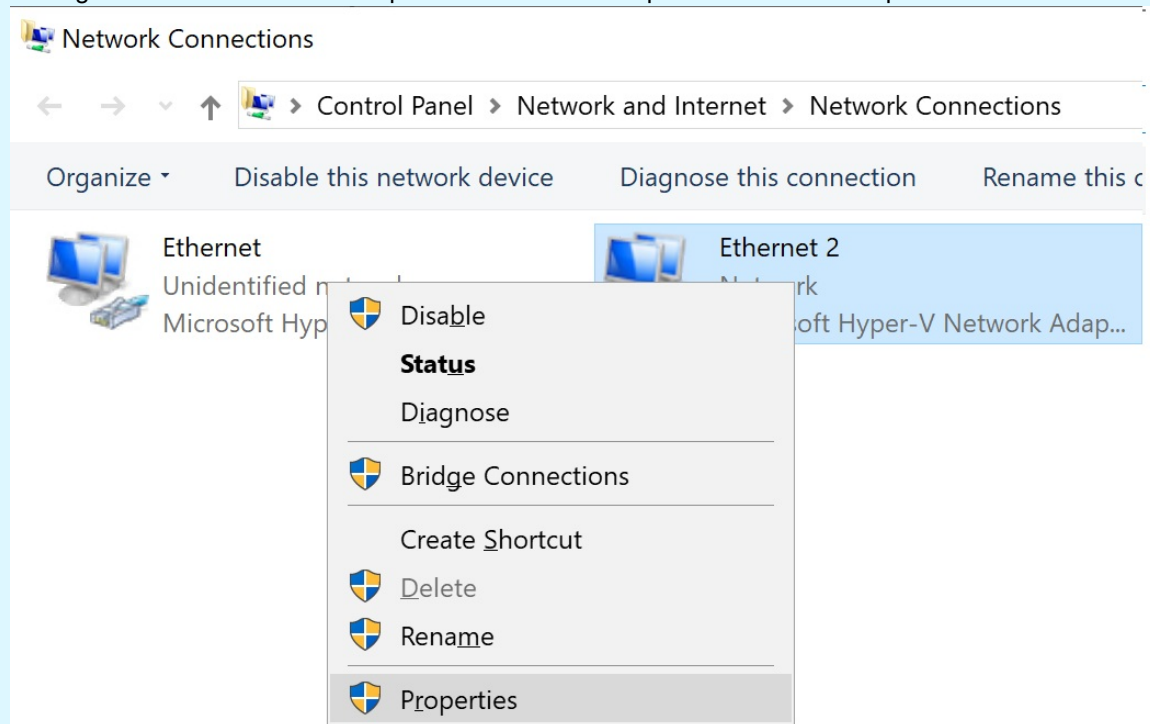
None Copy

```
netsh interface ipv4>set subinterface "Ethernet 2" mtu=2000 store=persistent
The parameter is incorrect.
```

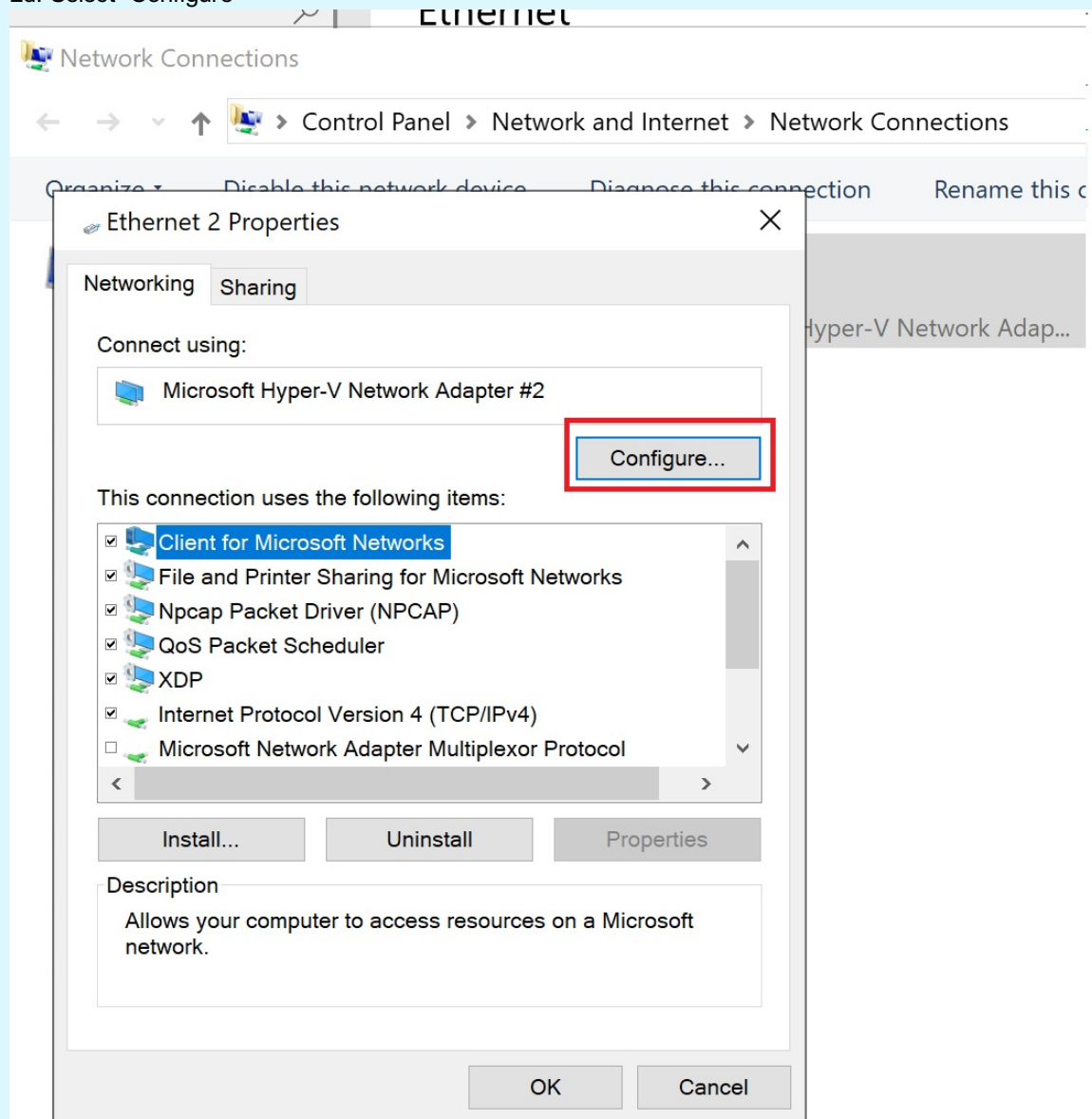
2b. Go to the Network connection → “Control Panel\Network and Internet\Network Connections”



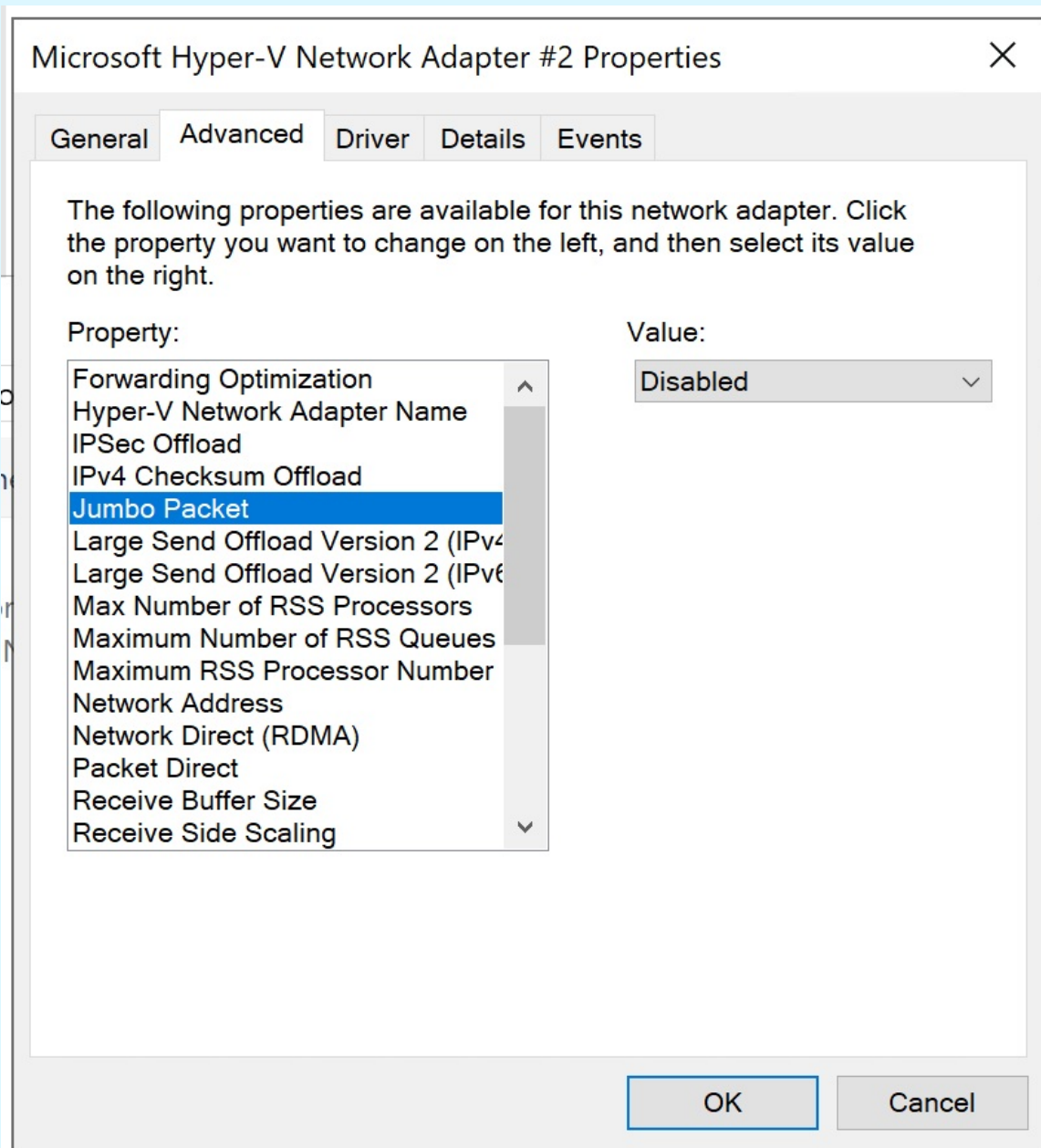
2c. Right click on the Ethernet Adaptor that the traffic is expected and select Properties.



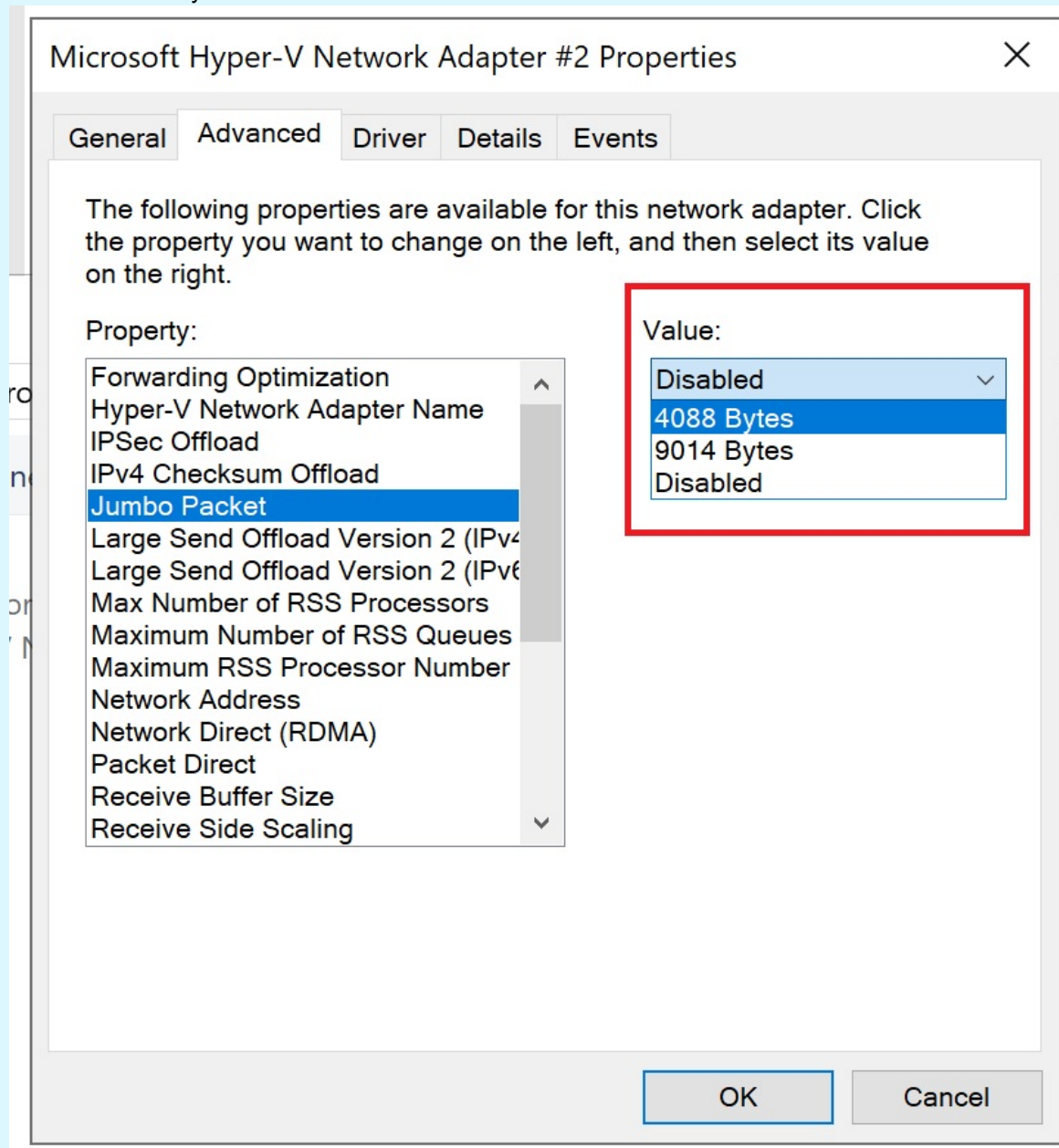
2d. Select "Configure"



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 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.

Locating xNIC Logs

An xNIC installed on a virtual machine creates one .log file per day with the following naming structure: **swtch-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\SwXtch.io\logs
 - Swtch-xNIC\ for xNIC1
 - Swtch-xNIC2\ for xNIC2
- **Linux:** /var/log/swtch
 - swtch-xnic for xNIC1
 - swtch-xnic2 for xNIC2

For Windows and Linux, you will see a folder for both versions of xNIC (1 and 2). Logs will only populate in the folder of the xNIC version you're using.

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.

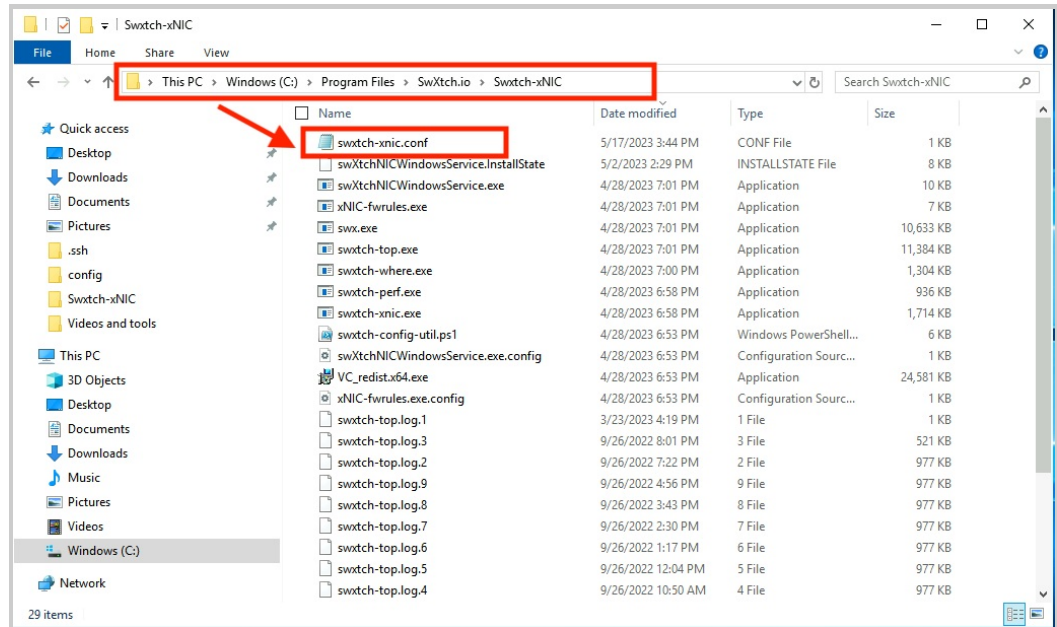
How to Change Verbosity

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

For Windows:

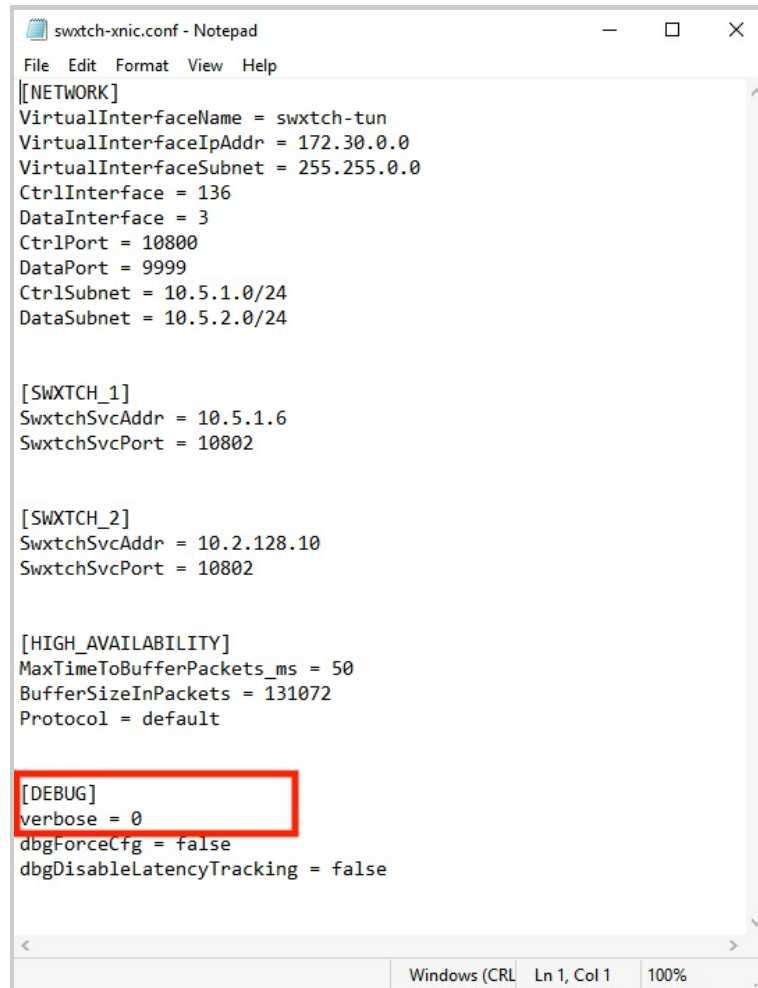
1. Go to the Swtch-xNIC folder on the VM you have an xNIC installed. Make sure it is the xNIC you want logs for.

1. For xNIC1: C:\Program Files\SwXtch.io\Swtch-xNIC
2. For xNIC2: C:\Program Files\SwXtch.io\Swtch-xNIC2



2. Open the "swtch-xnic.conf" file.

3. Change the number next to "verbose" so that it matches the level you desire. The default is 0.



```
swxtch-xnic.conf - Notepad
File Edit Format View Help
[NETWORK]
VirtualInterfaceName = swxtch-tun
VirtualInterfaceIpAddr = 172.30.0.0
VirtualInterfaceSubnet = 255.255.0.0
CtrlInterface = 136
DataInterface = 3
CtrlPort = 10800
DataPort = 9999
CtrlSubnet = 10.5.1.0/24
DataSubnet = 10.5.2.0/24

[SWXTCH_1]
SwxtchSvcAddr = 10.5.1.6
SwxtchSvcPort = 10802

[SWXTCH_2]
SwxtchSvcAddr = 10.2.128.10
SwxtchSvcPort = 10802

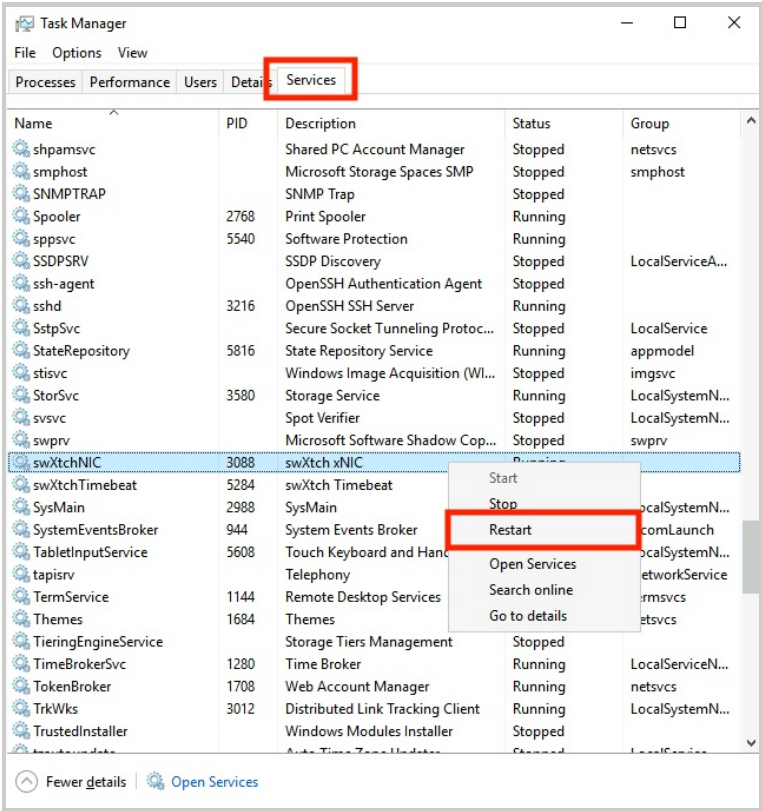
[HIGH_AVAILABILITY]
MaxTimeToBufferPackets_ms = 50
BufferSizeInPackets = 131072
Protocol = default

[DEBUG]
verbose = 0
dbgForceCfg = false
dbgDisableLatencyTracking = false

Windows (CRL Ln 1, Col 1 100%
```

4. Save and Close the config file.
5. Open "Task Manager" and go to the "Services" tab towards the top of the window.
6. Scroll down to "swXtchNIC" and right-click on it.

7. Select "Restart."



Your selection in verbosity will now be applied to future logs.

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

None	Copy
xNIC1:	
sudo nano /var/opt/swxtch/swxtch-xnic/swxtch-xnic.conf	
xNIC2:	
sudo nano /var/opt/swxtch/swxtch-xnic2/swxtch-xnic.conf	

2. Change the number next to "verbose" so that it matches the level you desire. The default is 0.

```
GNU nano 4.8
[NETWORK]
CtrlInterface="eth0"
DataInterface="eth1"
CtrlPort=10800
DataPort=9999
CtrlSubnet="10.5.1.0/24"
DataSubnet="10.5.2.0/24"
InstanceNumber=0

[SWXTCH_1]
SwxtchSvcAddr="10.5.1.6"
SwxtchSvcPort=10802

[SWXTCH_2]
SwxtchSvcAddr="10.2.128.10"
SwxtchSvcPort=10802

[HIGH_AVAILABILITY]
MaxTimeToBufferPackets_ms=50
BufferSizeInPackets=131072
Protocol="default"

[DEBUG]
verbose=0
dbgForceCfg=false
dbgDisableLatencyTracking=true
```

3. Save and Exit the file.
4. Restart your xNIC by using the following command:

Text

None	Copy
sudo systemctl restart swxtch-xnic.service	

Your selection in verbosity will now be applied to future logs.

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 Peer between VPCs in Different Regions for AWS

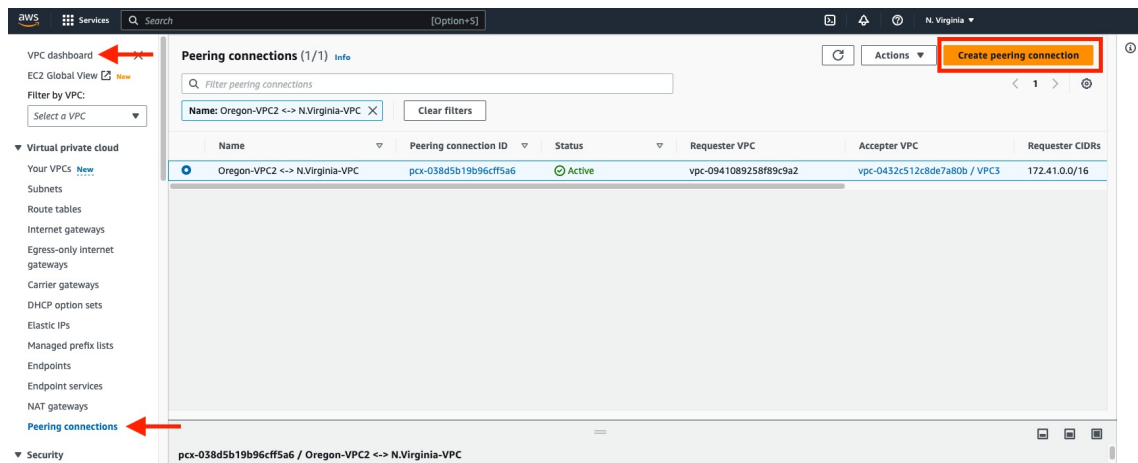
WHAT TO EXPECT

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	172.41.0.0/16	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
<input type="radio"/>	RegionalTestV...	pcx-0cfd2bfb57c419028	Active	vpc-019af57e1ad2e78a0	vpc-032478a302937fd9e / SA...	172.31.0.0/16	172.31.0.0/16	639720666639	639720666639
<input checked="" type="radio"/>	-	pcx-012996f2c0fffd69c	Pending acceptance	vpc-0dd3720d5088eadf6	vpc-0941089258f89c9a2 / SA...	172.31.0.0/16	-	639720666639	639720666639
<input type="radio"/>	Oregon-VPC2 ...	pcx-0fd12221d86e30677	Active	vpc-0941089258f89c9a2 / SA...	vpc-0432c512c8de7a80b	172.41.0.0/16	172.31.0.0/16	639720666639	639720666639
<input type="radio"/>	MLP-VPC-Peer	pcx-06d670c3874f620f1	Active	vpc-093b945e59a8ef1a8 / SA...	vpc-0941089258f89c9a2 / SA...	172.52.0.0/16	172.41.0.0/16	639720666639	639720666639
<input type="radio"/>	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
<input type="radio"/>	RegionalTestV...	pcx-0cfd2bfb57c419028	Active	vpc-019af57e1ad2e78a0	vpc-032478a302937fd9e / SA...	172.31.0.0/16	172.31.0.0/16	639720666639	639720666639
<input checked="" type="radio"/>	-	pcx-012996f2c0fffd69c	Pending acceptance	vpc-0dd3720d5088eadf6	vpc-0941089258f89c9a2 / SA...	172.31.0.0/16	-	639720666639	639720666639
<input type="radio"/>	Oregon-VPC2 ...	pcx-0fd12221d86e30677	Active	vpc-0941089258f89c9a2 / SA...	vpc-0432c512c8de7a80b	172.41.0.0/16	172.31.0.0/16	639720666639	639720666639
<input type="radio"/>	MLP-VPC-Peer	pcx-06d670c3874f620f1	Active	vpc-093b945e59a8ef1a8 / SA...	vpc-0941089258f89c9a2 / SA...	172.52.0.0/16	172.41.0.0/16	639720666639	639720666639
<input type="radio"/>	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
<input type="checkbox"/>	Private subnet routes for VPC1	rtb-0b9b0dd248a3db666	-	-	No	vpc-0432c512c8de7a80b VPC1	639720666639
<input type="checkbox"/>	RegionalTestVPC-rt	rtb-06b64a11e29effed5	2 subnets	-	No	vpc-019af57e1ad2e78a0 RegionalTestVPC	639720666639
<input type="checkbox"/>	Public subnet routes for VPC1	rtb-0f734496d1bed158c	subnet-01aaca91957af72d7 / VPC1-public	-	Yes	vpc-0432c512c8de7a80b VPC1	639720666639
<input type="checkbox"/>	Oregon-VPC2 -> N.Virginia-VPC	rtb-016f9783e7c943796	2 subnets	-	No	vpc-0432c512c8de7a80b VPC1	639720666639
<input type="checkbox"/>	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		
<input type="text" value="Find resources by attribute or tag"/>		
<input type="checkbox"/>	Name ▼	Route table ID ▼
<input type="checkbox"/>	Private subnet routes for VPC1	rtb-0b9bbdd248a3dbe66
<input type="checkbox"/>	RegionalTestVPC-rt	rtb-06b64a11e29effed5
<input type="checkbox"/>	Public subnet routes for VPC1	rtb-0f73d496d1bed158c
<input checked="" type="checkbox"/>	Oregon-VPC2 <-> N.Virginia-VPC	rtb-016f9783e7c943796
<input type="checkbox"/>	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' page 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 'Cancel', 'Preview', and 'Save changes' buttons are 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.

The screenshot shows the AWS Management Console interface for a route table. The left sidebar contains navigation options like 'Virtual private cloud', 'Security', and 'Network ACLs'. The main content area shows the 'Subnet associations' tab for the route table 'rtb-08dd6801fcf231277'. The 'Explicit subnet associations' section is highlighted with a red box, and the 'Edit subnet associations' button is also highlighted with a red box.

3. Select the subnet(s) of the instance that must be connected to the destination.

The screenshot shows the 'Edit subnet associations' dialog box in the AWS Management Console. The 'Available subnets' table lists various subnets, with 'VPC1-ctrl-subnet' and 'VPC1-data-subnet' selected. The 'Save associations' button is highlighted in orange.

	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input type="checkbox"/>	Default VPC1-d	subnet-0d6aac918668cedbe	172.31.80.0/20	-	Main (rtb-0f73d496d1bed158c / Public subnet routes for VPC1)
<input type="checkbox"/>	Default VPC1-f	subnet-0d9977ffb242c0086	172.31.64.0/20	-	Main (rtb-0f73d496d1bed158c / Public subnet routes for VPC1)
<input type="checkbox"/>	VPC1-public	subnet-01aaca91957af72d7	172.31.112.0/24	-	rtb-0f73d496d1bed158c / Public subnet routes for VPC1
<input checked="" type="checkbox"/>	VPC1-ctrl-subnet	subnet-03de822f9248b91ff	172.31.16.0/20	-	rtb-016f9783e7c943796 / Oregon-VPC2 <-> N.Virginia-VPC
<input checked="" type="checkbox"/>	VPC1-data-subnet	subnet-01df022a4f373fa17	172.31.96.0/20	-	rtb-016f9783e7c943796 / Oregon-VPC2 <-> N.Virginia-VPC
<input type="checkbox"/>	Default VPC1-e	subnet-068d4969f6ec1457a	172.31.48.0/20	-	Main (rtb-0f73d496d1bed158c / Public subnet routes for VPC1)
<input type="checkbox"/>	Default VPC1-b	subnet-06e372352e6a55935	172.31.32.0/20	-	Main (rtb-0f73d496d1bed158c / Public subnet routes for VPC1)
<input type="checkbox"/>	Default VPC1-c	subnet-0c633106e2e3bc850	172.31.0.0/20	-	Main (rtb-0f73d496d1bed158c / Public subnet routes for VPC1)

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

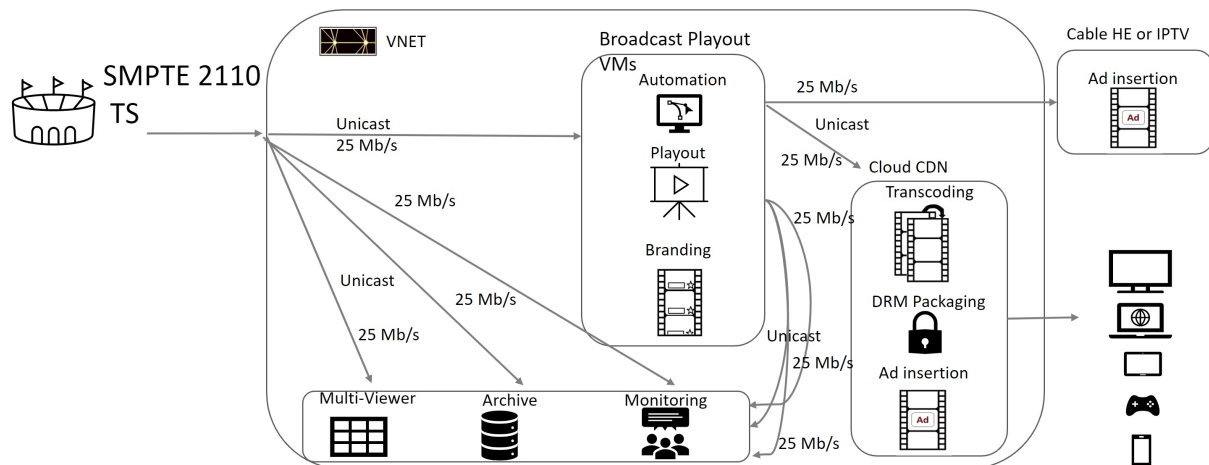
The media market can take advantage of several cloudSwXtch features such as:

- [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



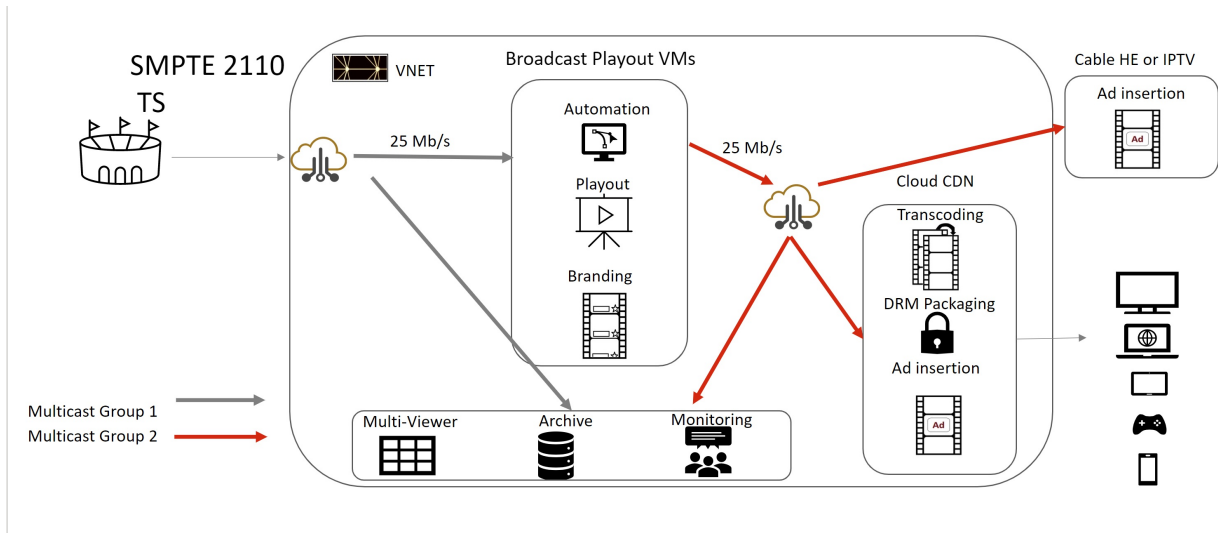
With unicast there are a number of issues:

- 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:

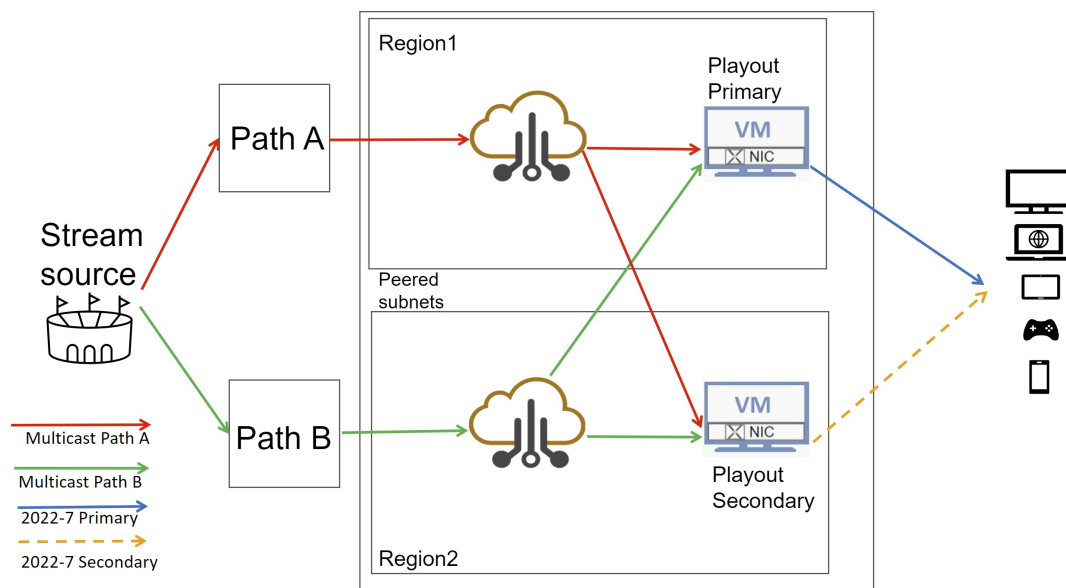
- Network may be modified and extended simply by joining multicast groups, with powerful centralized control and monitoring.
- Users can dynamically add new endpoints without playout server (or any other workflows/products) involvement.
- VM Sizes are minimized to workflow/product needs
- Disaster recovery 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

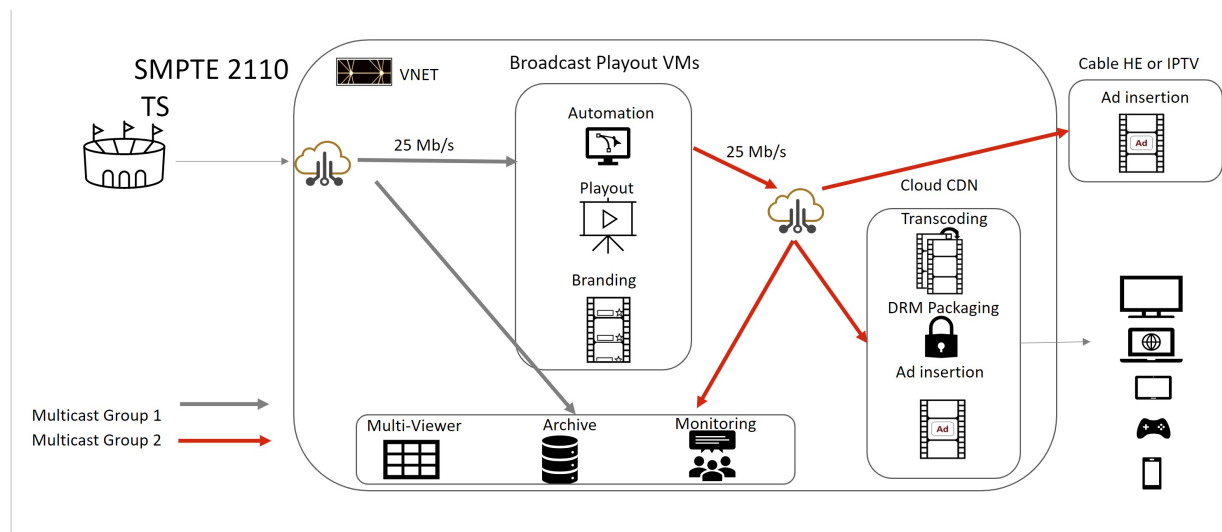


At **swXtch.io** we know that the media companies rely highly on both compressed and

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.

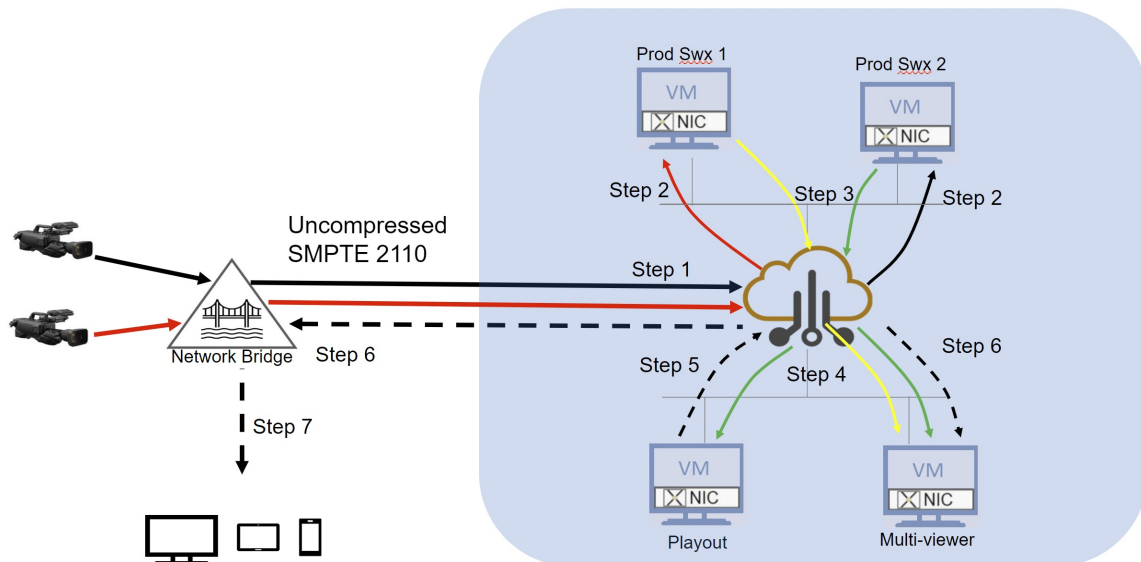
There are two workflow examples below, one is a compressed workflow and the other is an uncompressed workflow. The compressed workflow is a typical playout scenario where compressed inputs come into the cloud environment and are distributed via multicast to the necessary VM workloads by **cloudSwXtch**. 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.

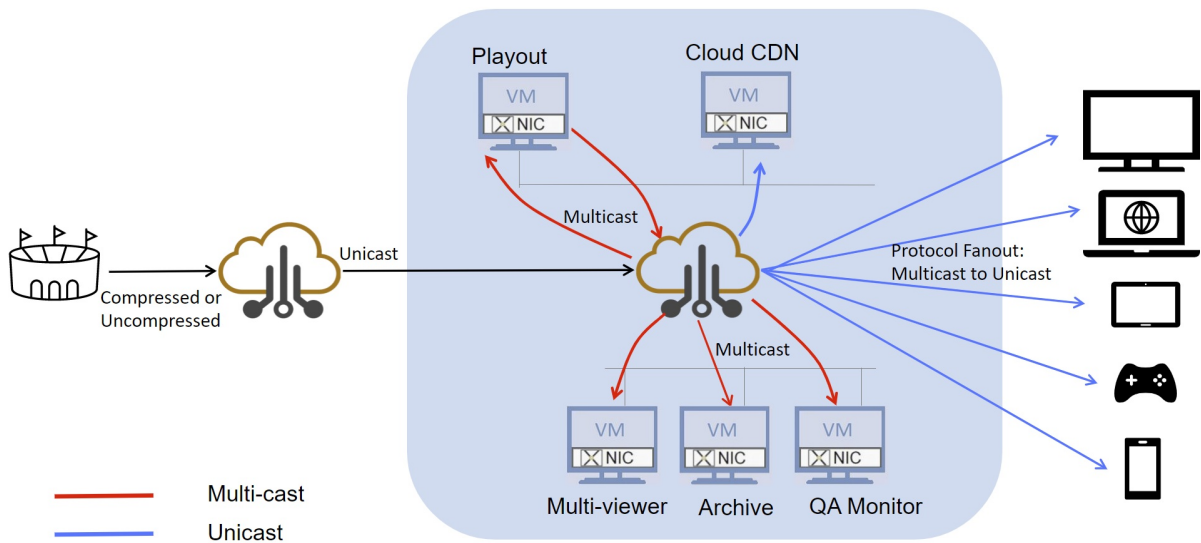
Step 7: The network bridge distributes to the clients for viewing consumption.

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

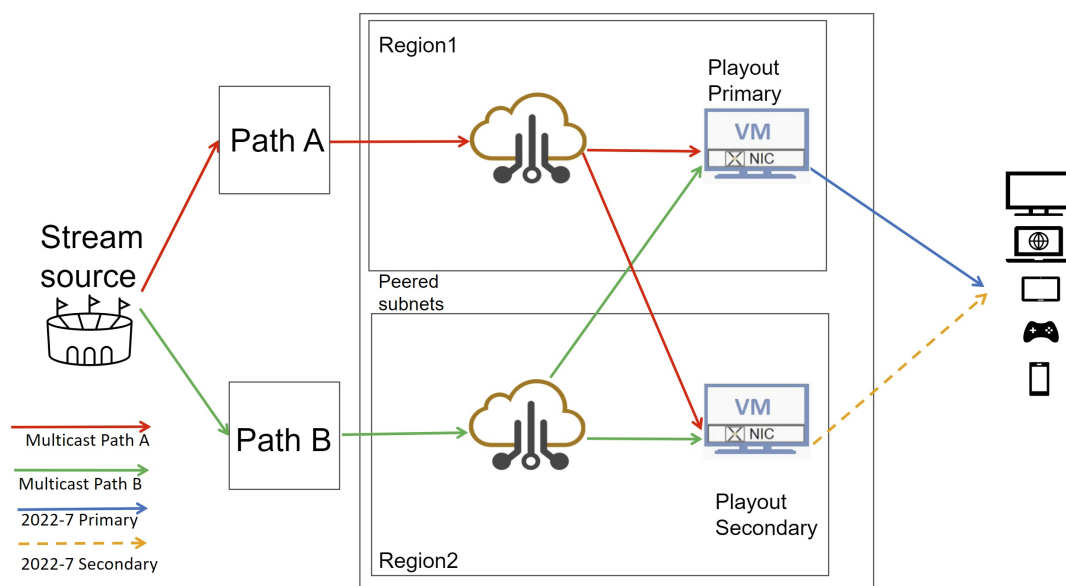
- The **cloudswXtch** in Region 1 can recieve the stream from Path A to Region 2.
- The **cloudswXtch** in Region 2 can recieve the stream from Path B to Region 1.
- If either path were to fail then the stream is still available in both Region 1 and 2 due to the redundancy.

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.

$\Delta Nic2McaTotal$ 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

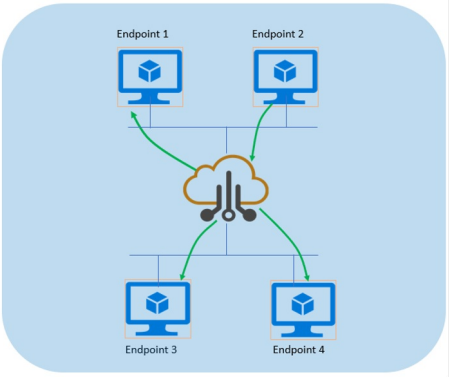
```
curl http://<cloudSwXtch-control-IP>/api/wxckedeye/v1/dashboard
```

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.

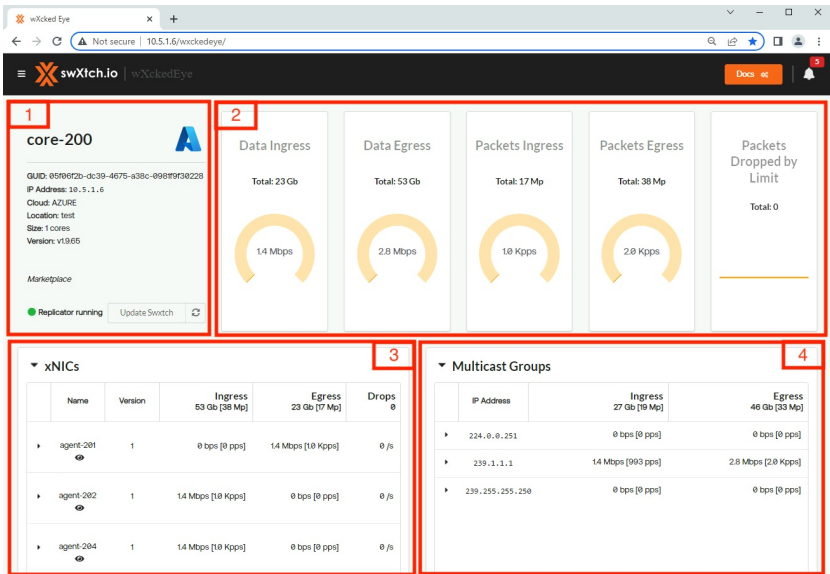
The example that will be used in this article refers to the network below, which has been simplified to help users understand the data returned in the API. As shown in Figure 1, Endpoint (Agent) 2 is sending data via multicast through the cloudSwXtch to Endpoints 1, 3 and 4.



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
------	------

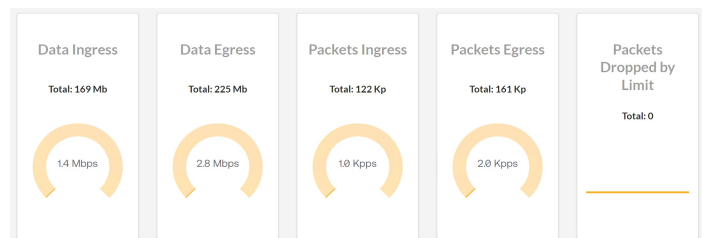
```

"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
},

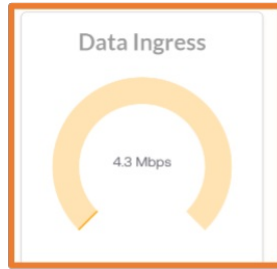
```

cloudSwXtch network ingress and egress data are useful when shown in a custom user interface. To display the data like in the example of the user interface above, API calls should be made periodically. These points in time can then be used to compute data based on time.

$\Delta Something$ means the difference between the value of at time and *Something* at time $t1$ and *Something* $t2$. Example: If Egress has value 39000 at time $t1$ and value 19800 at time $t2$,

$$\Delta Egress = (39000 - 19800) = result$$

Calculating ByteCounters - Data Ingress



$$DataIngress = [xnicTotals][ByteCounters](\Delta Nic2McaTotal * 8) / ((\Delta Timestamp / 1,000,000,000))$$

Below is a cut from the response of the API called at two different times.

27	"ByteCounters": {	27	"ByteCounters": {
28	"Nic2McaTotal": 51730345462,	28	"Nic2McaTotal": 51706144186,
29	"Nic2McaMc": 51730345462,	29	"Nic2McaMc": 51706144186,
30	"Mca2NicTotal": 51729942516,	30	"Mca2NicTotal": 51721942820,
31	"Mca2NicMc": 51729941994	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.

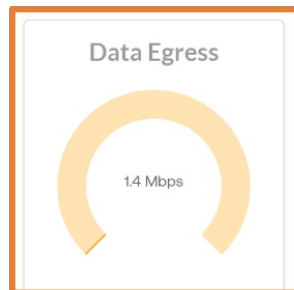
Timestamp from the call 2, 1658155900751865500 and the timestamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 in nanoseconds. Then, dividing the Δ by 1,000,000,000, we get the Δ of 46.523565312 in seconds, which we will use to calculate the data rate in bits per second.

$$\Delta Nic2McaTotal \text{ is } 51730345462 - 51706144186 = 7999676bits$$

$$[xnicTotals][ByteCounters](\Delta Mca2NicTotal * 8) / ((\Delta Timestamp / 1,000,000,000)) = (7999676 * 8) / 46.523565312 = 4161551.392 \approx 4.2Mbps$$

Please note: Since we calculated the average data ingress over a timespan of ~46 seconds, the value does not exactly match the UI screenshot. This is because the UI web app calls the API every 5 seconds by default, giving us more instantaneous rate compared to our calculation.

Calculating ByteCounters - Data Egress



$$DataEgress = [xnicTotals][ByteCounters](\Delta Mca2NicTotal * 8) / ((\Delta Timestamp / 1,000,000,000))$$

Below is a cut from the response of the API called at two different times.

27	"ByteCounters": {	27	"ByteCounters": {
28	"Nic2McaTotal": 51730345462,	28	"Nic2McaTotal": 51706144186,
29	"Nic2McaMc": 51730345462,	29	"Nic2McaMc": 51706144186,
30	"Mca2NicTotal": 51729942516,	30	"Mca2NicTotal": 51721942820,
31	"Mca2NicMc": 51729941994	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,

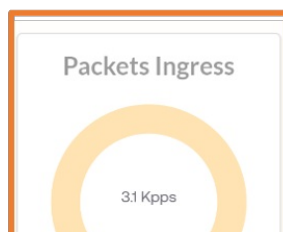
Taking the above expression and putting in data from this call is shown below.

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.

$$\Delta Mca2NicTotal \text{ is } 51729942516 - 51721942820 = 63997568bits$$

$$[xnicTotals][ByteCounters](\Delta Mca2NicTotal * 8) / ((\Delta Timestamp / 1,000,000,000)) = (63997568 * 8) / 46.523565312 = 1375594.66 \approx 1.4Mbps$$

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	"Mca2McaDrops": 0,	22	"Mca2McaDrops": 0,
23	"Mca2KniDrops": 0,	23	"Mca2KniDrops": 0,
24	"McaPktDrops": 0,	24	"McaPktDrops": 0,
25	"McaBigPktDrops": 0,	25	"McaBigPktDrops": 0,
26	},	26	},

Taking the above expression and putting in data from this call is shown below.

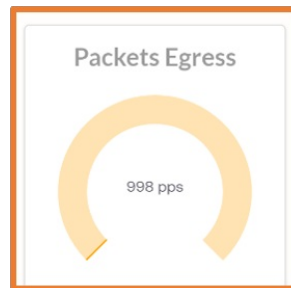
Timestamp from call 2, 1658155900751865500 and the timestamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 nanoseconds. Then, dividing the Δ by 1,000,000,000, we get a Δ of 46.523565312 in seconds.

$$\Delta Nic2McaTotal \text{ is } 297300912 - 297161822 = 139090 \text{Packets}$$

$$[xnicTotals][PktCounters](\Delta Nic2McaTotal)/((\Delta Timestamp/1,000,000,000)) = 139090/46.523565312 = 2989.667689 \approx 3.0Kpps$$

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	},	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 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.

$$\Delta Mca2NicTotal \text{ is } 517298589 - 517252613 = 988.2303665 \text{Packets}$$

$$[xnicTotals][PktCounters](\Delta Mca2NicTotal)/((\Delta Timestamp/1,000,000,000)) = (988.2303665)/46.523565312 = 988.2303665 \approx 988pps$$

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 [56 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	"0.004000": 0,	74	"0.004000": 0,

75	0.010000": 0,	75	0.010000": 0,
76	0.011000": 0	76	0.011000": 0
77		77	
78	"HARxCounters": {},	78	"HARxCounters": {},
79	"Timestamp": 1658155853888965889,	79	"Timestamp": 1658155900043488705,
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, 1658155853888965889 gives us a Δ of 46154522880 nanoseconds. Then, dividing the Δ by 1,000,000,000, we get Δ of 46.15452288 in seconds.

Data Ingress: $\Delta Nic2McaTotal * 8 = 64030976bits$

$[xnlcs][Dsd - agent - 101][ByteCounters](\Delta Nic2McaTotal * 8) / ((\Delta Timestamp / 1,000,000,000)) = (64030976) / 46.15452288 = 1387317.473 \approx 1.4Mbps$

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	"Mca2NicIgmP": 0,	90	"Mca2NicIgmP": 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	"HARxCounters": {},	118	"HARxCounters": {},
119	"Timestamp": 1658155853955371860,	119	"Timestamp": 1658155900086539499,
120	"SoftwareVersion": "v1.7.4.draft",	120	"SoftwareVersion": "v1.7.4.draft",
121	"XnicVersion": 2	121	"XnicVersion": 2
122		122	

Taking the above expressions and putting in data from this call is shown below.

Timestamp from call 2, 1658155900086539499 and timestamp from call 1, 1658155853955371860 gives us a Δ of 46131167744 nanoseconds. Then, dividing the Δ by 1,000,000,000, we get Δ of 46.131167744 in seconds.

Data Egress: $\Delta Mca2NicTotal * 8$ is $(51729917250 - 5172191774) * 8 = 63995808bits$

$[xnlcs][Dsd - agent - 102][ByteCounters](\Delta Mca2NicTotal * 8) / ((\Delta Timestamp / 1,000,000,000)) = (63995808) / (46.131167744) = 1387257.49054 \approx 1.4Mbps$

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]

This section will discuss data available for multicast groups for a cloudSwXtch and will be broken down into two sections: Ingress and Egress. The multicast group 239.1.1.1 will be used for calculations below. Below is an example of the data stats, Multicast Group data and other statistics:

Bash	Copy
<pre>}, "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": [{ "groupIp": "239.1.1.3", "pktsCount": 31460, "bytesCount": 33788040, </pre>	

```

        "lastUpdate": "2022-07-07T15:29:00.78906233Z"
      },
      "srcIp": null,
      "srcPort": 0,
      "protocolType": 0,
      "numberOfDestinations": 0
    },
    {
      "groupId": "239.0.0.251",
      "pktsCount": 75,
      "bytesCount": 8754,
      "lastUpdate": "2022-07-07T17:54:49.132519721Z"
    },
    "srcIp": null,
    "srcPort": 0,
    "protocolType": 0,
    "numberOfDestinations": 0
  },
  "subscriptionId": "d723b93f-112c-49fb-9fda-03d3ada867f3",
  "hostName": "dsd-core-174",
  "numCores": 8,
  "replStatus": "running",      *Replicator services status*
  "authorized": true,
  "validationResult": {
    "switch": 556,
    "authorized": true,
    "denialReason": null,
    "trialPeriod": {
      "startDate": "2021-12-22T19:38:33.565336Z",
      "endDate": "2022-08-21T19:38:33.565Z"
    },
    "applicationId": null
  },
  "isMarketplace": true,
  "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: $([x_{\text{nic}}][RXMulticastGroups]\Delta bytesCount * 8) / \Delta lastUpdate$

Packets: $[x_{\text{nic}}][RXMulticastGroups]\Delta pktsCount / \Delta lastUpdate$

Below is a cut from the return at two separate times with data for this section highlighted in orange.

"rxMulticastGroups": {	"rxMulticastGroups": {
{	{
"groupId": "239.255.255.250", "pktsCount": 956, "bytesCount": 238044, "lastUpdate": "2022-07-19T21:29:58.813001632Z"	"groupId": "239.255.255.250", "pktsCount": 956, "bytesCount": 238044, "lastUpdate": "2022-07-19T21:29:58.813001632Z"
{	{
"groupId": "239.1.1.1", "pktsCount": 840049064, "bytesCount": 93968537136, "lastUpdate": "2022-07-25T19:44:40.189630746Z"	"groupId": "239.1.1.1", "pktsCount": 840092012, "bytesCount": 93976010088, "lastUpdate": "2022-07-25T19:45:23.191477537Z"
}	}

$\Delta lastUpdate = 43.00003982 \text{ seconds}$

Data Ingress:

$\Delta bytesCount * 8 = (93976010088 - 93968537136) * 8 = 59783616 \text{ bits}$

$([x_{\text{nic}}][RXMulticastGroups]\Delta bytesCount * 8) / \Delta lastUpdate = 59783616 / 43.00003982 = 1390315.364 \approx 1.4 \text{ Mbps}$

Note that the value in the figure above is wrongfully shown as Bytes/second as opposed to bits/s.

Packets Ingress:

$$\Delta pktsCount = 540092012 - 540049064 = 42948 packets$$

$$[xnics][RXMulticastGroups]\Delta pktsCount / \Delta lastUpdate = 42948 / 43.00003982 = 998.789772749 \approx 999 pps$$

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: $([xnics][TXMulticastGroups]\Delta bytesCount * 8) / \Delta lastUpdate$

Packets: $[xnics][TXMulticastGroups]\Delta pktsCount / \Delta lastUpdate$

241	1,	241	1,
242	"txMulticastGroups": {	242	"txMulticastGroups": {
243	{	243	{
244	"groupId": "239.1.1.1",	244	"groupId": "239.1.1.1",
245	"pktsCount": 597605144,	245	"pktsCount": 597733970,
246	"bytesCount": 103983295056,	246	"bytesCount": 104005710780,
247	"lastUpdate": "2022-07-25T19:24:40.162378549Z"	247	"lastUpdate": "2022-07-25T19:24:43.162418365Z"
248	},	248	},
249	{	249	{
250	"groupId": "224.0.0.251",	250	"groupId": "224.0.0.251",
251	"pktsCount": 969,	251	"pktsCount": 979,
252	"bytesCount": 115212,	252	"bytesCount": 115872,
253	"lastUpdate": "2022-07-25T19:24:40.061601877Z"	253	"lastUpdate": "2022-07-25T19:24:43.162378503Z"
254	},	254	},
255	},	255	},
256	},	256	},

Below is a cut from the return at two separate times with data for this section highlighted in orange.

$$\Delta lastUpdate = 43 seconds$$

Data Egress: $\Delta bytesCount * 8 = (104005710780 - 103983295056) * 8 = 179325792 bits$

$$([xnics][TXMulticastGroups]\Delta bytesCount * 8) / \Delta lastUpdate = (179325792) / 43 = 4170367.256 \approx 4.2 Mbps.$$

Note: The value in the figure above is wrongfully shown as Bytes/second as opposed to bits/s.

Packets Ingress: $\Delta pktsCount = 597733970 - 597605144 = 128826 packets$

$$[xnics][TXMulticastGroups]\Delta pktsCount / \Delta lastUpdate = 128826 / 43 = 2995.954 \approx 3.0 Kpps$$

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.



dsd-core-300

Information

Version: v1.9.85
Size: 1
Ctrl IP: 10.1.1.6
Ctrl Port: 10802
Subnet Data Prefix: 10.1.2.0/24
Subnet Ctrl Prefix: 10.1.1.0/24
Gateway IP: 10.1.2.1

Update Swtch

Entitlements

Max Clients	Max Bandwidth (Mbps)	Enabled Features			
		Mesh	High Availability	Protocol Fanout	PTP
10	2000	✓	✓	✓	✓

Data refresh period

Adjust how often the real time data gets refreshed

5 secs

© 2023 swXtch.io

Below is a portion of the Settings call result detailing information in the General tab:

Bash

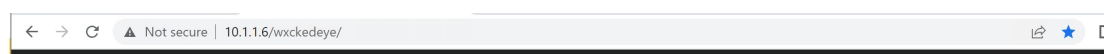
Copy

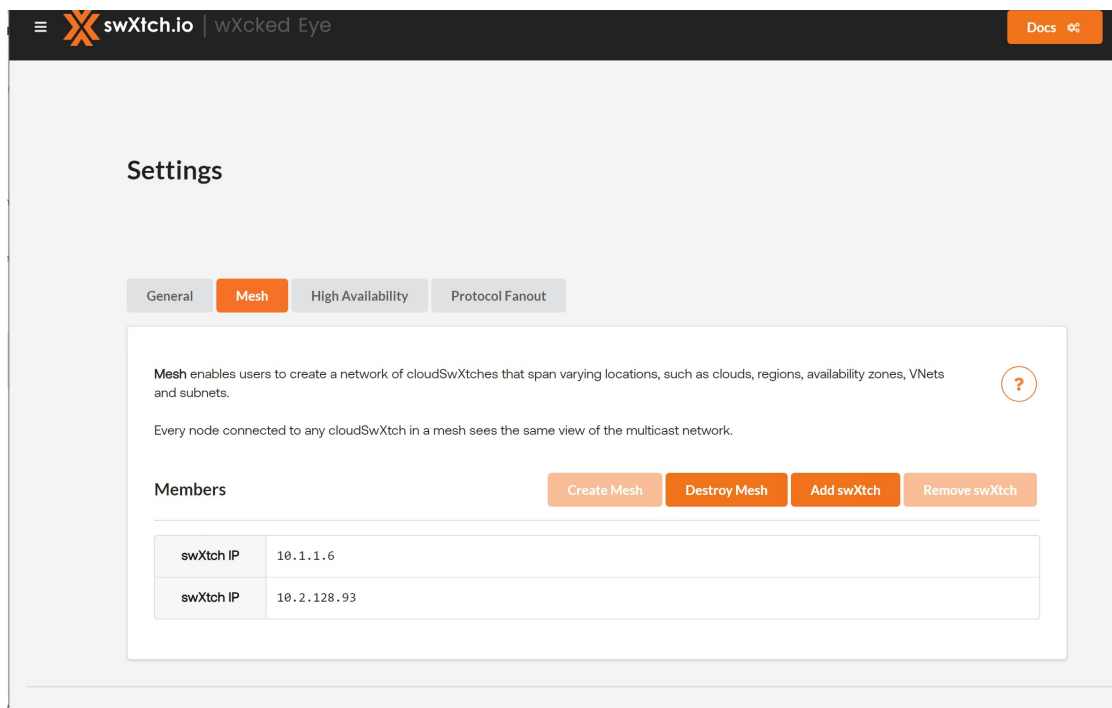
```

{
  "generalSettings": {
    "hostName": "core-300",
    "switchName": "core-300",
    "version": "v1.9.85",
    "numCores": 1,
    "entitlements": {
      "maxClientCount": 10,
      "bandwidthMbps": 2000,
      "enableMesh": true,
      "enableUnicast": true,
      "enableHA": true,
      "enableClockSync": true
    },
    "subnetDataPrefix": "10.1.2.0/24",
    "subnetCtrlPrefix": "10.1.1.0/24",
    "dataGatewayIp": "10.1.2.1",
    "ctrlIp": "10.1.1.6",
    "ctrlPort": 10802,
    "gatewayMacAddr": "12:34:56:78:9a:bc",
    "replInfo": {
      "CtrlIp": "1.0.0.127",
      "CtrlPort": 9996,
      "DataIp": "10.1.2.6",
      "DataPort": 9999,
      "DataMac": "YEW9pzYf"
    }
  },
}
```

Mesh

The mesh tab shows the cloudSwXtches that are configured to be in a Mesh. Note that both High Availability and Mesh are configured here for example purposes. Mesh and High Availability are, however, not compatible with the same cloudSwXtches.





Below is a portion of the Settings call response detailing information on the Mesh tab:

BashCopy

```
"meshSettings": {
  "meshId": "86B62026-9222-0E62-03C2-6C5872CA64C5",
  "swxtches": [
    "10.1.1.6",
    "10.2.128.93"
  ],
  "uid": null
},
```

Create Mesh

►
swxtch/mesh/v1/create



Show Mesh

►
swxtch/mesh/v1/show



Add cloudSwXtch to Mesh

►
swxtch/mesh/v1/addSwxtch



Remove cloudSwXtch from Mesh

►
`swxtch/mesh/v1/removeSwxtch`



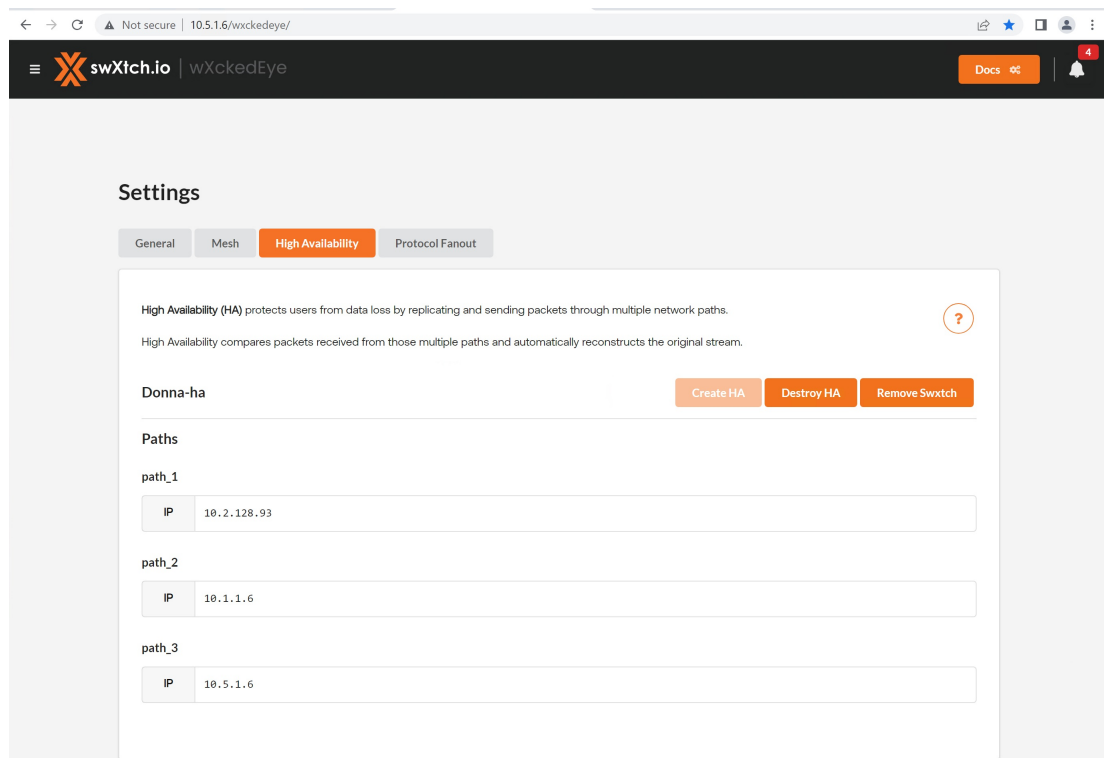
Destroy Mesh

►
`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>	

```

    {
      "name": "path_2",
      "swxtches": [
        "10.1.1.6"
      ]
    }
    {
      "name": "path_3",
      "swxtches": [
        "10.5.1.6"
      ]
    }
  ]
}

```

Create HA

►
swxtch/ha/v1/create

```

]

```

Get HA List

►
swxtch/ha/v1/show

```

]

```

Delete cloudSwXtch from HA

►
swxtch/ha/v1/removeSwxtch

```

]

```

Destroy HA

►
swxtch/ha/v1/destroy

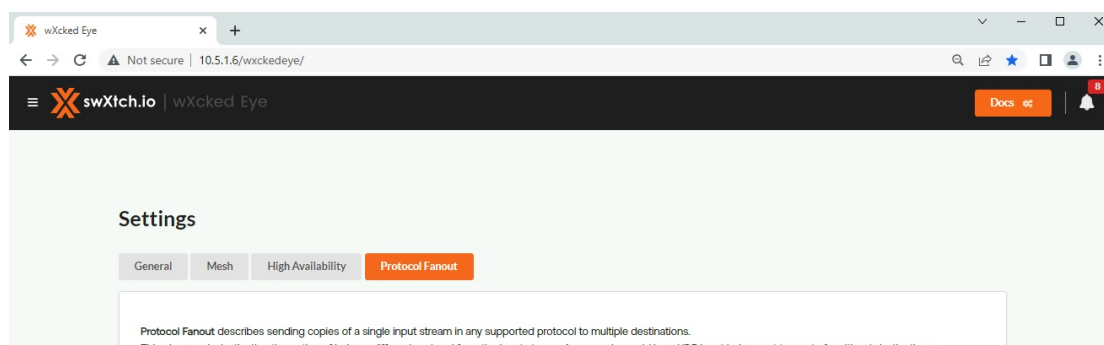
```

]

```

Protocol Fanout



The Protocol Fanout tab shows the cloudSwXtches that are configured for Protocol Fanout: multicast, UDP (Unicast), SRT Caller and SRT Listener.



and, additionally, to one using SRT.

▼ Streams

Add Stream

Name	Group IP	Group Port	Action
Hockey	239.1.1.2	2000	 

▼ Nodes

Add Node

Below is a portion of the Settings call response detailing information on Protocol Fanout.

BashCopy

```
"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

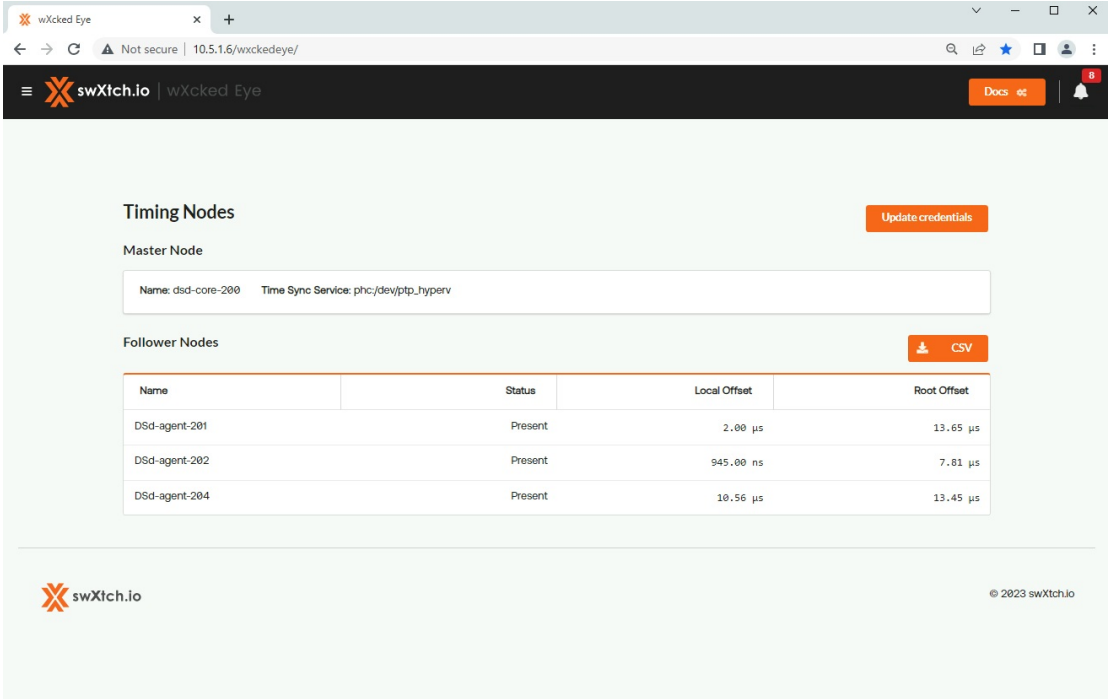
►
swxtch/protocols/srt-listener/v1/egress/leave

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.



The screenshot shows the wXcked Eye web interface. The header includes the swXtch.io logo and navigation links. The main content area is titled "Timing Nodes" and features an "Update credentials" button. Below this, there is a "Master Node" section with a form containing "Name: dsd-core-200" and "Time Sync Service: phc/dev/ptp_hyperv". A "Follower Nodes" section contains a table with three rows of data. A "CSV" button is located to the right of the table. The footer includes the swXtch.io logo and copyright information.

Name	Status	Local Offset	Root Offset
Dsd-agent-201	Present	2.00 μ s	13.65 μ s
Dsd-agent-202	Present	945.00 ns	7.81 μ s
Dsd-agent-204	Present	10.56 μ s	13.45 μ s

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