

CLOUDSWTCH

Version 1.9.85

Table of Contents

cloudSwXtch

| | |
|------------------------|---|
| FAQ | 4 |
| Quotas | 5 |

cloudSwXtch > Getting Started

| | |
|--------------------------------------|---|
| Getting Started | 6 |
| What is cloudSwXtch? | 7 |

cloudSwXtch > Getting Started > cloudSwXtch Features

| | |
|---|----|
| cloudSwXtch Features | 9 |
| Multicast | 11 |
| Broadcast | 12 |
| High Availability | 13 |
| Mesh | 15 |
| Bridge | 17 |
| Precision Time Protocol | 18 |

cloudSwXtch > Installing cloudSwXtch

| | |
|---|----|
| cloudSwXtch System Requirements | 19 |
|---|----|

cloudSwXtch > Installing cloudSwXtch > cloudSwXtch on Azure

| | |
|---|----|
| cloudSwXtch on Azure | 20 |
| Validate Subnets on Azure | 22 |
| Create an Azure cloudSwXtch Template | 24 |
| Install cloudSwXtch on Azure | 27 |
| Install cloudSwXtch via Market Place | 32 |
| Upgrade cloudSwXtch on Azure | 40 |
| Deploy cloudSwXtch with Terraform on Azure | 41 |
| Install cloudSwXtch for an Air-Gapped Environment | 45 |

cloudSwXtch > Installing cloudSwXtch > cloudSwXtch on AWS

| | |
|---|----|
| cloudSwXtch on AWS | 55 |
| Validate Subnets on AWS | 56 |
| Verify Security Groups | 59 |
| Create SSH Key Pair | 61 |
| Install cloudSwXtch on AWS | 63 |
| Upgrade cloudSwXtch on AWS | 72 |
| Delete cloudSwXtch on AWS | 73 |
| Check Health of cloudSwXtch Instance on AWS | 74 |

cloudSwXtch > Installing cloudSwXtch Bridge

| | |
|---|----|
| Installing cloudSwXtch Bridge | 76 |
| Install cloudSwXtch Bridge V2 | 77 |
| Install cloudSwXtch Bridge V1 | 80 |

cloudSwXtch > Installing xNIC

| | |
|--|----|
| Installing xNIC | 83 |
| xNIC System Requirements | 84 |

cloudSwXtch > Installing xNIC > Install xNIC on Linux

| | |
|---------------------------------------|----|
| Install xNIC on Linux | 85 |
| xNIC Linux Uninstall | 88 |
| xNIC Linux Upgrade | 89 |

cloudSwXtch > Installing xNIC > Install xNIC on Windows

| | |
|---|----|
| Install xNIC on Windows | 90 |
| Uninstall xNIC on Windows | 93 |
| Upgrade xNIC on Windows | 94 |

cloudSwXtch > Installing xNIC > Install xNIC on AKS Cilium

| | |
|--|-----|
| Install xNIC on AKS Cilium | 95 |
| Create an Azure Kubernetes Service with Cilium | 97 |
| Install xNIC2 on AKS Cilium | 103 |
| Test xNIC2 with AKS | 113 |
| Upgrade xNIC nodes on AKS | 119 |

cloudSwXtch > Configuring cloudSwXtch

| | |
|--|-----|
| Mesh | 122 |
| Bridge | 127 |
| Protocol Conversion and Fanout | 128 |
| Licensing Information | 131 |

cloudSwXtch > Monitoring cloudSwXtch

| | |
|----------------------------------|-----|
| Azure Monitoring | 133 |
|----------------------------------|-----|

cloudSwXtch > Testing cloudSwXtch

| | |
|-------------------------------------|-----|
| Testing cloudSwXtch | 136 |
| swxtch-perf | 137 |
| swxtch-top | 142 |
| Troubleshooting | 151 |

cloudSwXtch > cloudSwXtch How To's

| | |
|---------------------------------------|-----|
| How to set MTU size | 152 |
| How to Find xNIC Logs | 158 |

cloudSwXtch > Market Use Cases > Media Use Cases

| | |
|---|-----|
| Media Use Cases | 163 |
| Media Multicast made easy with cloudswXtch | 164 |
| Hitless Merge - 2022-7 | 166 |
| Media support for Compressed and Uncompressed Workflows | 167 |
| Protocol Fanout | 169 |
| Disaster Recovery | 170 |

cloudSwXtch > cloudSwXtch API

| | |
|-----------------------------------|-----|
| Monitoring API | 171 |
| Configuration API | 183 |

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.

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

| Name | Core | Memory | Hard Drive | Bandwidth capacity (egress) | # Endpoints |
|--------|------|----------|------------|-----------------------------|-------------|
| Small | 8 | 16GB DDR | 64GB SSD | 2Gb/s | 10 |
| Medium | 16+ | 16GB DDR | 64GB SSD | 30 Gb/s | 50 |
| Large | 64+ | 16GB DDR | 64GB SSD | 96 Gb/s | 200 |

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 |

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 both Azure and AWS.
- Users **must** install an **xNIC** on every VM that needs to send or receive **cloudSwXtch** multicast or broadcast traffic.

[Azure cloudSwXtch Installation Guide](#)

[AWS cloudSwXtch Installation Guide](#)

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

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.

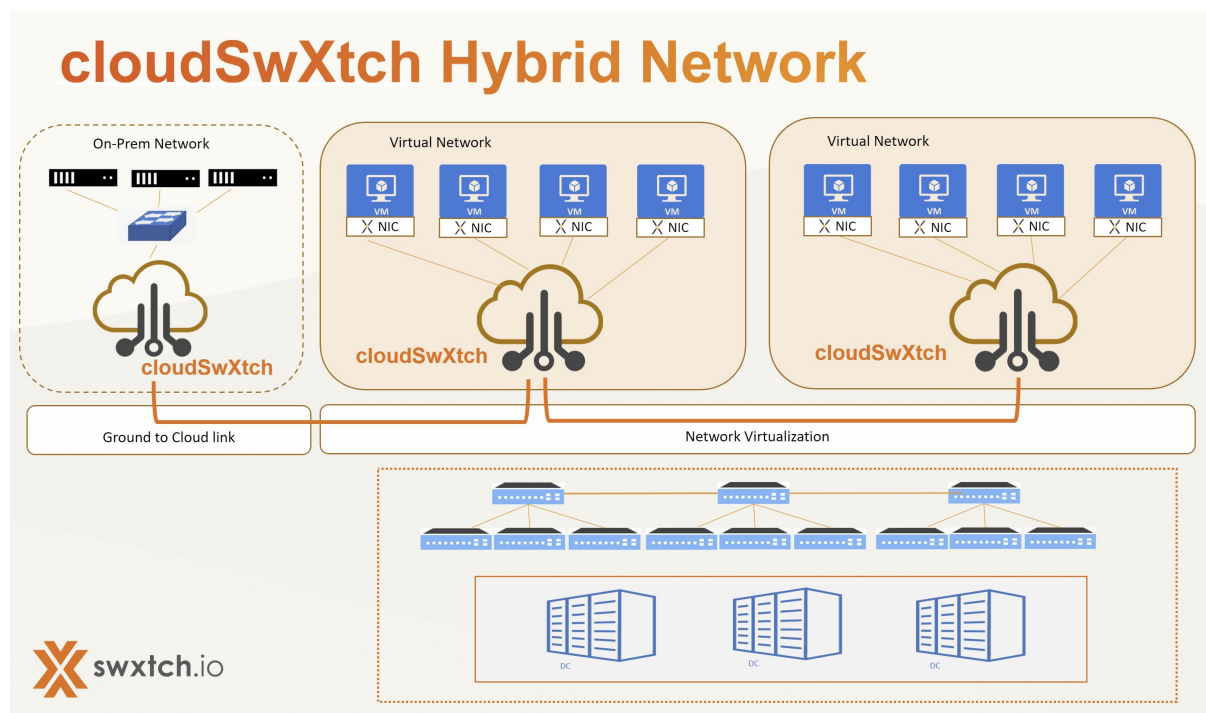
Supported Enviroments

- Microsoft's [Azure](#) Cloud
- Amazon's [AWS](#) Cloud
- 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 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.

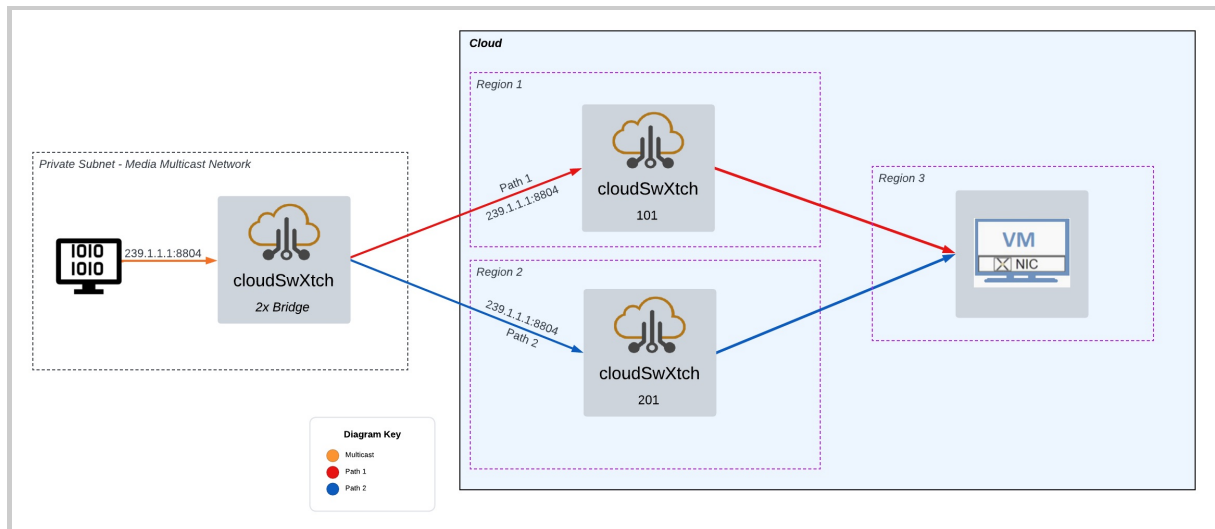
In addition, there is no limit to the number of multicast groups per data path. If one cloud, availability zone or region should go down, then the data is still sent in the other 2-8 paths, ensuring that the consumer gets the necessary data. Consumers can also be put into different clouds, availability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

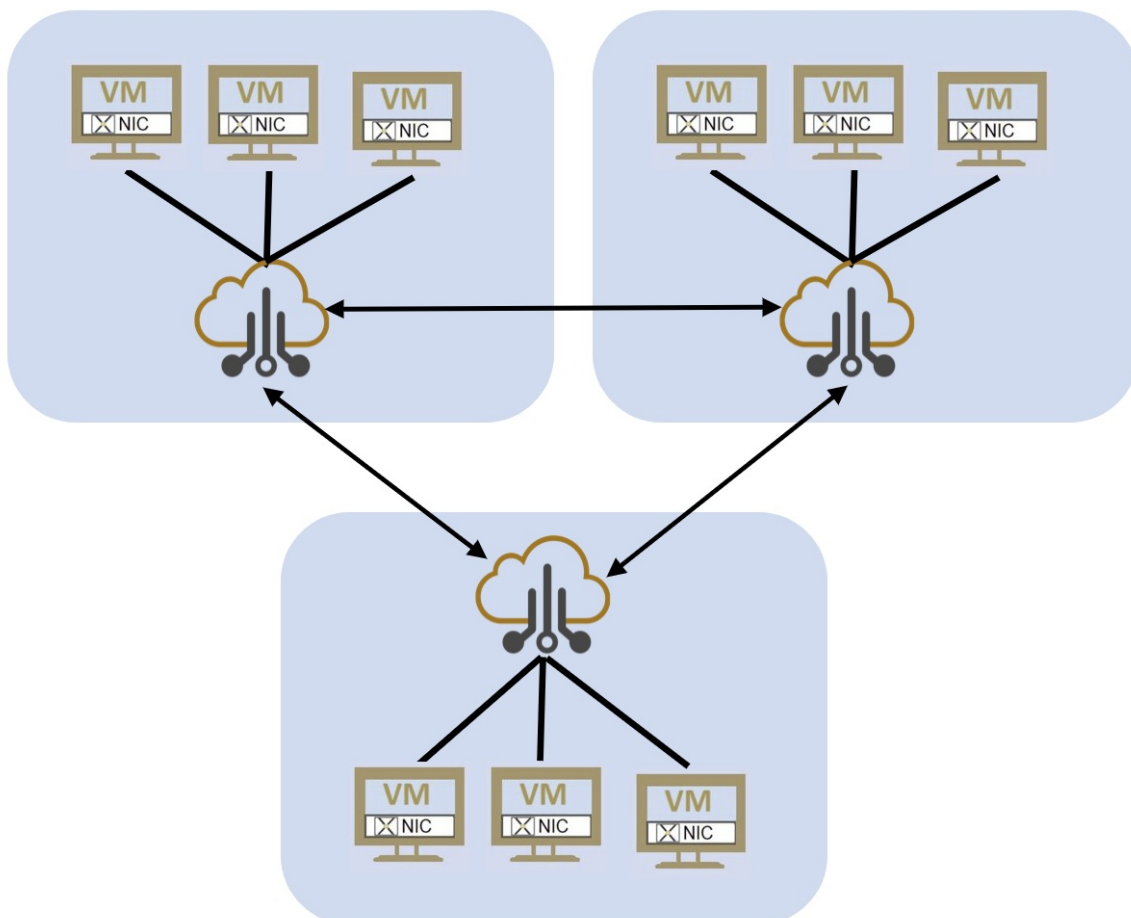
A mesh connects cloudSwXtches in a variety of dispersed network locations – different Vnets, regions, clouds, subnets, data centers, ect.). Additionally, a mesh is a way to group two or more swXtch's together to act as one to gain network performance.

Learn more about a Mesh

- See [wXcked Eye](#) for monitoring swXtches to understand existing capacity to know if you need to consider creating a swXtch mesh.
- See [cloudSwXtch Installation](#) for installing a swXtch
- See [Configuring a Mesh](#) for mesh configuration.

Mesh

A Mesh is formed by linking cloudSwXtches so that they are eligible to receive and transmit multicast traffic to other switches in the same mesh. [Configuring a mesh](#) 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 switch-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.

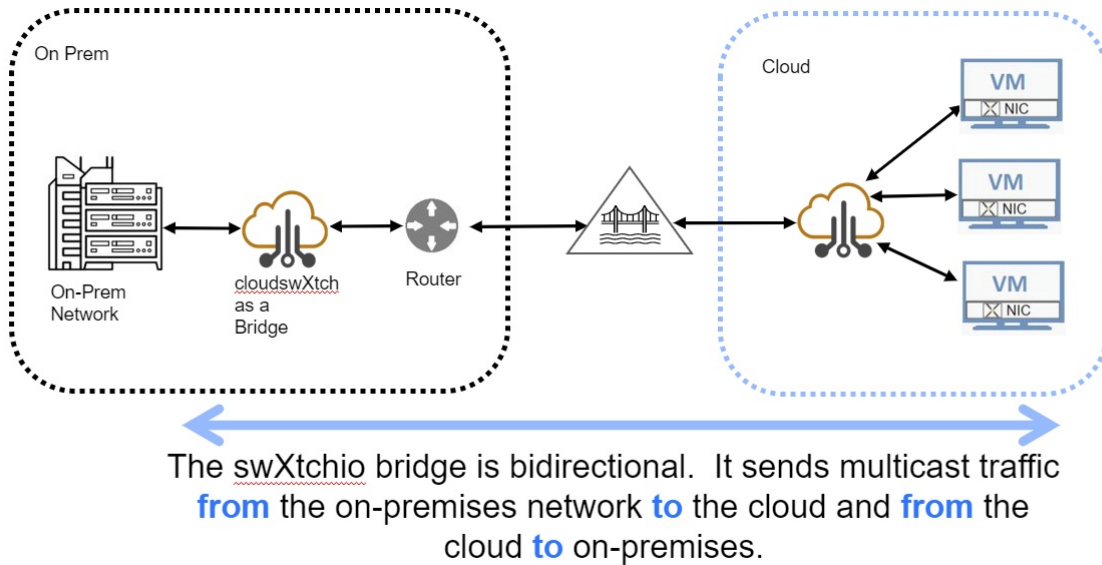
IMPORTANT

- 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 switch 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.



{height="" width=""}

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 [Bridge Configuration](#) on how to configure the bridge.

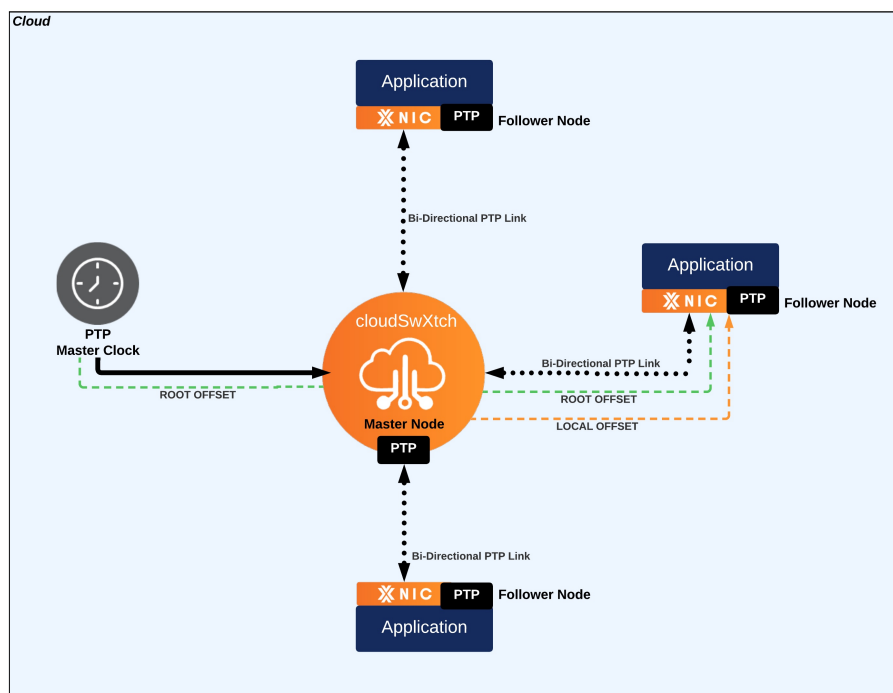
Precision Time Protocol

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.

cloudSwXtch System Requirements

Supported cloud environments

- Microsoft's [Azure](#) Cloud
- Amazon's [AWS](#) Cloud

Virtual Network

A virtual network is required to create a cloudSwXtch instance. This virtual network must meet the following characteristics:

- 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 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 cloudSwxtch must be a minimum of 4 Cores and 16 GiB memory.

Firewall and Security Group Rules

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

| subnet | protocol | ports | vm |
|-------------|----------|-------------|-------------|
| ctrl-subnet | http | 80 | cloudSwXtch |
| ctrl-subnet | udp | 10800-10803 | all |
| data-subnet | udp | 9999 | all |

Mesh and High Availability

Both Mesh and High Availability need special firewall exceptions in order to properly work in a user's cloudSwXtch environment. If you plan on using either 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 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.

Microsoft Azure

Home > Marketplace >

cloudSwXtch VM Image (preview) ✕ ...

swxtch.io LLC

cloudSwXtch VM Image (preview) Add to Favorites

swxtch.io LLC

Free trial

Plan

Small swXtch Create Start with a pre-set configuration

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

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

IMPORTANT

Use the provided templates in the link below to **easily** deploy a cloudSwXtch. These templates ensure correct setup of the VM and network interfaces.
If you deploy using the Azure Portal Web UI the resulting VM may not function properly.

Templates for deployment: [Templates](#)
For more installation instructions see: [docs.swxtch.io](#)

Benefits

Plug and Play Existing applications and services that expect standards-based IP multicast and broadcast will work in Azure without requiring any code changes.

wXtched Fast Moving high-performance systems to the cloud can be difficult. cloudSwXtch enables massive bandwidth with sub-millisecond latency for most applications.

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

Microsoft Azure

Home > Marketplace >

cloudSwXtch VM Image (preview) ✕ ...

swxtch.io LLC

cloudSwXtch VM Image (preview) Add to Favorites

swxtch.io LLC

Free trial

Plan

Small swXtch Create Start with a pre-set configuration

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

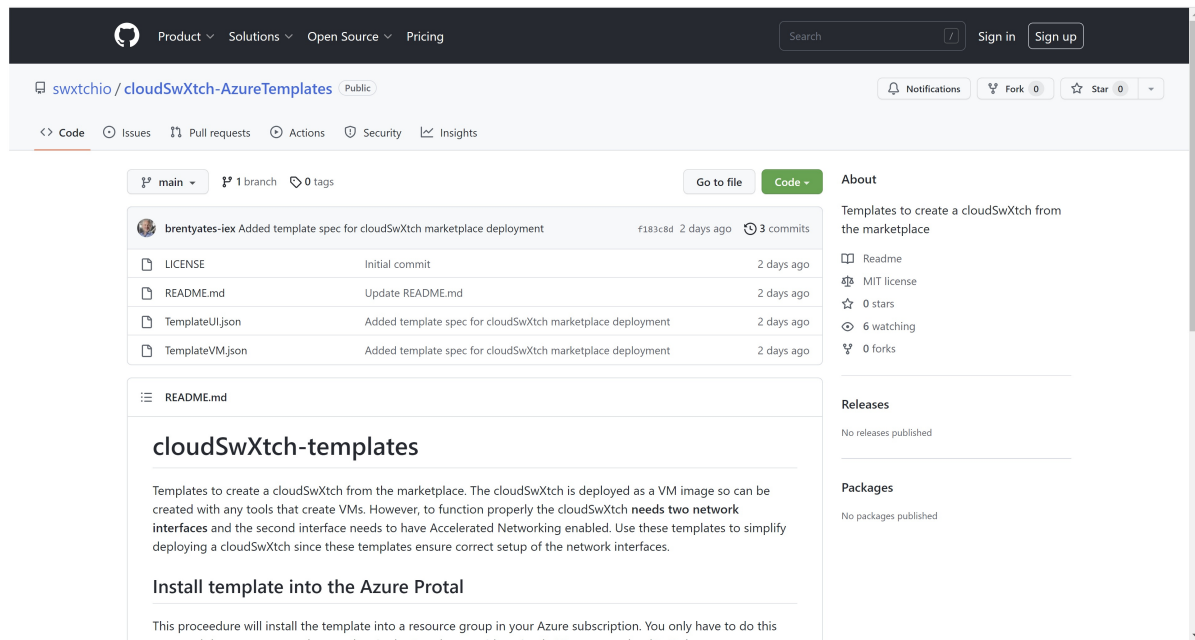
The cloudSwXtch is a network appliance that is deployed as a **vm image** so can be created with any tools that create VMs. However, to function properly the cloudSwXtch needs two network interfaces and the second interface needs to have Accelerated Networking enabled.

IMPORTANT

Use the provided templates in the link below to **easily** deploy a cloudSwXtch. These templates ensure correct setup of the VM and network interfaces.
If you deploy using the Azure Portal Web UI the resulting VM may not function properly.

Templates for deployment: [Templates](#)
For more installation instructions see: [docs.swxtch.io](#)

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

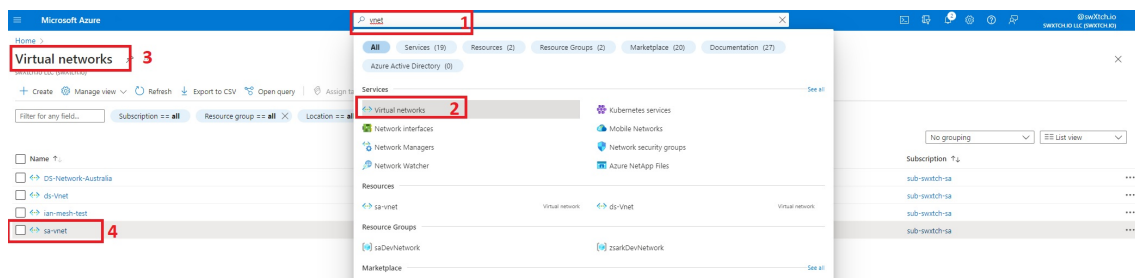
A virtual network must be created before deploying a cloudSwXtch instance.

- The VNet must contain two subnets: one that's used for control plane communication and another for data plane communication.
- Both subnets need 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.
- The subnets must be the same subnets used for the xNIC installations.

In this section, users will learn how to create both the ctrl- and the data-subnets for their virtual network in preparation for [cloudSwXtch installation on Azure](#).

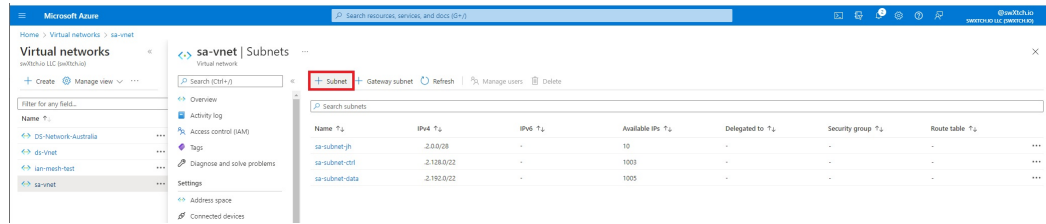
To validate:

1. Go to the Azure Console.
2. Search for "vnet".
3. Select Virtual Networks.
4. Select the vnet to be used for cloudSwXtch.



5. Name the subnets as "ctl" and "data" to distinguish between them when creating an VM instance.

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

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 "cloudSwXth-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$

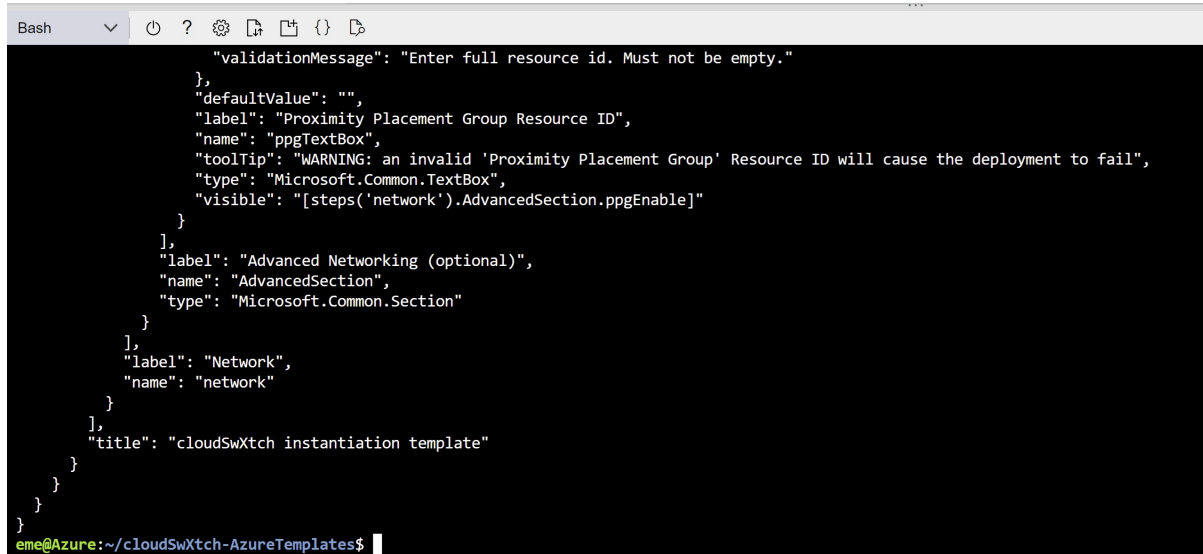
```

7. Create "cloudSwxtch-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.

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 you 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.

Install cloudSwXtch via Market Place

WARNING

This method is not suggested, but is documented as a valid method of creation.

The best method to use is via **template**, which is detailed in the "[Install cloudSwXtch on Azure](#)" guide.

Prerequisites

Before starting, ensure that you [validate your subnets](#) on Azure. Return to this page after completing that preliminary step.

Creating a Virtual Machine

1. **Log in to the Azure Portal.** You will need the following permissions to create and manage virtual machines and to create Managed Applications.

- **virtual-machine-contributor:** To create and manage virtual machines.
- **managed-application-contributor-role:** To create Managed Applications.

2. **Select "Marketplace."**

3. **Search for "cloudswXtch."**

4. **Select a plan.** For more information, see: [cloudSwXtch Pricing](#).

5. Click on the "cloudSwXtch VM Image" drop down menu to select a plan.

The screenshot shows the Microsoft Azure Marketplace interface. At the top, there's a search bar with 'cloudswxtch' entered. Below the search bar, it says 'Showing 1 to 1 of 1 results for 'cloudswxtch''. A card for 'cloudSwXtch VM Image' by swxtch.io LLC is displayed. The card includes a 'Free trial' badge, the product name, the provider name, and a description: 'Virtual Machine Multicast on Azure via a virtual network switch (overlay network with IGMP support)'. It also states 'Starts at \$2.00/hour'. A 'Create' button is visible, and a dropdown menu is open below it, showing three options: 'Small swXtch', 'Medium swXtch', and 'Large swXtch'. The left sidebar contains navigation links for 'Get Started', 'Service Providers', 'Management', 'Private Marketplace', 'Private Offer Management', 'My Marketplace', 'Favorites', 'Recently created', 'Private products', and 'Categories'.

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

This screenshot shows the 'cloudSwXtch VM Image' page. The 'Plan' dropdown menu is open, showing three options: 'Small swXtch', 'Medium swXtch', and 'Large swXtch'. The 'Small swXtch' option is selected. Below the dropdown, there are 'Create' and 'Start with a pre-set configuration' buttons. The page also includes an 'Overview' section with a description of the product and an 'IMPORTANT' note about deployment templates.

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

NOTE

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

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

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

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

9. Choose (or create) an "Resource Group."

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

11. Select a "Region."

12. Select the "Image" and choose an appropriate image based on the plan type small, medium or large selected.

Microsoft Azure

Search resources, services, and docs (G+)

Home > Marketplace > cloudSwXtch VM Image >

Create a virtual machine

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

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

14. Continue on the Networking Tab.

Networking Tab

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

15. Check "Delete public IP and NIC when VM is deleted".

16. OPTIONAL: Change values on other tabs.

17. Select ****Review and Create****.

18. Carefully review the plan pricing.

19. Read the Terms & Conditions.

20. Select "I agree" when ready.

Please note: The creation will take 2-3 minutes depending on Azure varieties.

Creating the Second Subnet ***REQUIRED***

21. Navigate to the newly created VM by selecting the "Go to Resource" button.

22. Click "Stop" at the top of the toolbar.

23. Select "Yes" when prompted.

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

Home > CreateVm-swxtch300-1614108926893.cloudswtch-vm--20220930112431 | Overview >

vm-swxtch300
Virtual machine

Search < Connect > Start < Restart < Stop < Capture < Delete < Refresh < Open in mobile < CU / PS

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Networking
Connect
Disks
Size
Microsoft Defender for Cloud
Advisor recommendations
Extensions + applications
Continuous delivery
Availability + scaling
Configuration
Identity
Properties
Locks
Operations
Bastion
Auto-shutdown
Backup
Disaster recovery
Updates
Inventory

Essentials
Resource group (move) : test-donna-300-rg
Status : Stopped (deallocated)
Location : East US
Subscription (move) : sub-swxtch-300
Subscription ID : b10209ad-ad22-4c26-8aef-be93b2f0bb58
Tags (edit) : CreatedBy : dsilveri@swxtch.io

Operating system : Linux
Size : Standard D4 v4 (4 vcpus, 16 GiB memory)
Public IP address : 40.112.52.133
Virtual network/subnet : test-donna-300-vnet/ds-subnet-ctrl-300
DNS name : Not configured

Properties Monitoring Capabilities (7) Recommendations Tutorials

Virtual machine
Computer name : dsd-vm-swxtch300
Health state : -
Operating system : Linux
Publisher : swxtch300-1614108926893
Offer : cloudswtch-vm-001-preview
Plan : swxtch-small-002
VM generation : V1
VM architecture : x64
Host group : None
Host : -
Proximity placement group : -
Colocation status : N/A
Capacity reservation group : -

Availability + scaling
Availability zone : -
Availability set : -
Scale Set : -

Networking
Public IP address : 40.112.52.133
Public IP address (IPv6) : -
Private IP address : 10.1.1.6
Private IP address (IPv6) : -
Virtual network/subnet : test-donna-300-vnet/ds-subnet-ctrl-300
DNS name : Configure

Size
Size : Standard D4 v4
vCPUs : 4
RAM : 16 GiB

Disk
OS disk : dsd-vm-swxtch300_OsDisk_1_ba053f9f18e54a10
Encryption at host : Disabled
Azure disk encryption : Not enabled
Ephemeral OS disk : N/A
Data disks : 0

Auto-shutdown

25. Select "Attach network interface."

Microsoft Azure

Search resources, services, and documentation

Home > CreateVm-swxtch300 | Overview > vm-swxtch300

vm-swxtch300 | Networking

Virtual machine

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Settings
- Networking**

Attach network interface Detach network interface

vm-swxtch300727

IP configuration ⓘ

ipconfig1 (Primary)

Network Interface: vm-swxtch300727 Effective security rules Troubleshoot

Virtual network/subnet: test-donna-300-vnet/ds-subnet-ctrl-300 NIC Public IP: 40.112

Inbound port rules Outbound port rules Application security groups Load balancers

Network security group dsd-vm-swxtch300-nsg (attached to network interface)

26. Select a "Resource Group" under Project Details.

27. Enter in a "Name" under Network Interface.

28. Select a "Subnet."

Please note: You can optionally change other data.

29. Select "Create"

Microsoft Azure

Search resources, services, and docs (G+)

Home > vm-swxtch300 | Networking >

Create network interface ...

Project details

Subscription ⓘ
sub-swX-sa

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

Location ⓘ
(US) East US

Network interface

Name *
vm-swxtch300-data ✓

Virtual network ⓘ
test-300-vnet

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

NIC network security group ⓘ
☐ None
☒ Basic
☐ Advanced

Public inbound ports * ⓘ
☒ None
☐ Allow selected ports

Select inbound ports
Select one or more ports

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

Private IP address assignment
☒ Dynamic ☐ Static

Create

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

Enabling "Accelerated Networking" *REQUIRED*

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

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

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

Home > vm-swxtch300

vm-swxtch300 | Networking

Search

Attach network interface Detach network interface

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

IP configuration ipconfig1 (Primary)

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

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

Inbound port rules Outbound port rules Application security groups Load balancing

Network security group basicNsgdsd-vm-swxtch300-data (attached to network interface: dsd-vm-swxtch300-data) Impacts 0 subnets, 1 network interfaces [Add inbound port rule](#)

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

Need help?

[Understand Azure load balancing](#)

[Quickstart: Create a public load balancer to load balance Virtual Machines](#)

[Quickstart: Direct web traffic with Azure Application Gateway](#)

3. Click "Edit accelerated networking."

Home > vm-swxtch300 | Networking >

vm-swxtch300-data

Search

Move Delete Refresh **Edit accelerated networking**

Essentials

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

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

Subscription (move): sub-swxtch300 Private IP address (IPv6): -

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

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

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

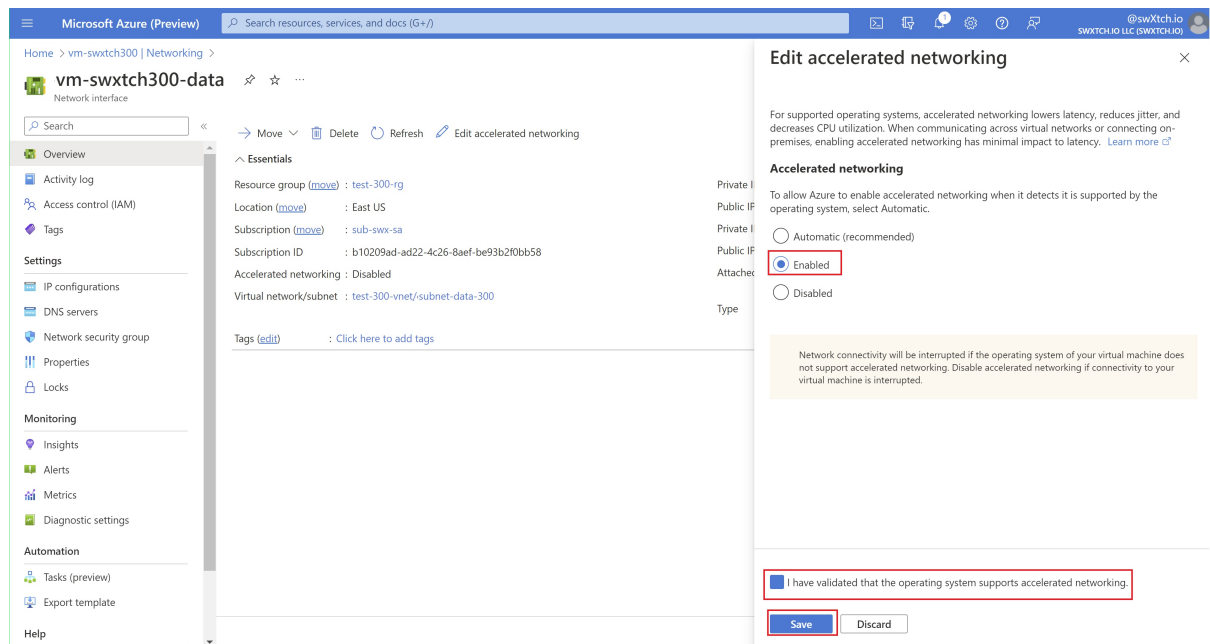
Type: Regular

Tags (edit): [Click here to add tags](#)

4. Select "Enable."

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

6. Click "Save."



7. Start the VM for use.

Important

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

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 the following steps:

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 v1.9.16 --ip 10.5.1.6 v1.9.16</pre> | |

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

 [azure_deploySwxtch.tf](#)

 [azure_var_deploySwxtch.tf](#)

 [azure_var_network.tf](#)

Attached are **azure_deploySwxtch.tf**, **azure_var_deploySwxtch.tf**, **azure_var_network.tf** files that will be deployed via Terraform to create a cloudSwXtch in your Azure network.

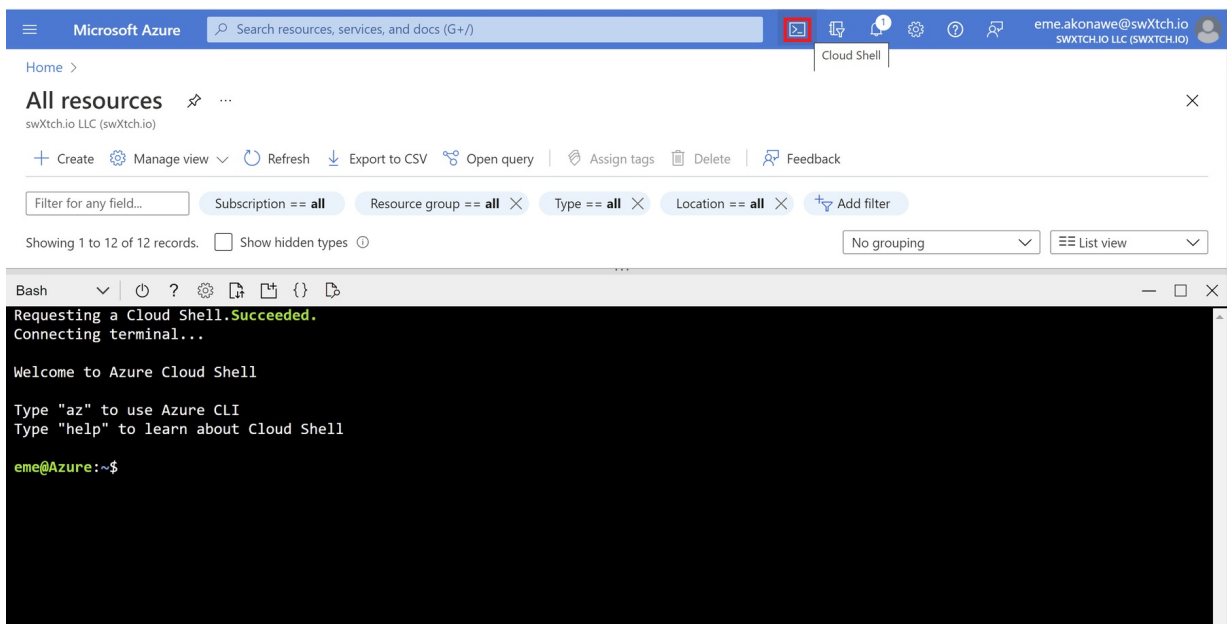
Note:

By default, the terraform script will spin up a "Small" Swxtch. You can make edits to the **azure_var_deploySwxtch.tf** file under the variable "sku_" to declare a different Swxtch size.

There is also an option to delegate static ip addresses on your Cloud Swxtch. Further details on how to do this can be found at the end of this document.

Steps to deploy this Terraform script are as follows:

1. Sign-in to your Azure portal under the subscription where you want to deploy the cloudSwXtch.
2. Update the "default" values in the **azure_var_deploySwxtch.tf** & **azure_var_network.tf** file to match your existing azure resources such as: resource group, virtual network, subnets, etc.
3. Open the Azure Cloud Shell interface and select the Bash environment as shown.



4. From the Cloud Shell terminal, upload the **azure_deploySwxtch.tf**, **azure_var_deploySwxtch.tf**, **azure_var_network.tf** file by clicking the icon shown below in the grey status bar.

Bash



5. Once the files are uploaded, they will be placed in the /home/user directory.
6. Move the files into a newly created folder (ex. "TerraformDeploy") where you can easily manage deployment files such as your .state files (which will be created) and any other environment specific files. Be sure to also add your public key file to this directory. You will need to update the azure_deploySwxtch.tf file, under the **admin_ssh_key** parameter to point to the correct directory.

None

Copy

```
mkdir TerraformDeploy
mv azure_deploySwxtch.tf azure_var_deploySwxtch.tf azure_var_network.tf
TerraformDeploy
```

7. Change working directory to TerraformDeploy & run Terraform init to initialize your working directory

None

Copy

```
cd TerraformDeploy
Terraform init
```

```
eme@Azure:~/TerraformDeploy$ terraform init
```

Initializing the backend...

Initializing provider plugins...

- Finding latest version of hashicorp/azurerm...
- Installing hashicorp/azurerm v2.97.0...
- Installed hashicorp/azurerm v2.97.0 (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.

```
eme@Azure:~/TerraformDeploy$
```

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

None

Copy

```
Terraform plan
```



```

Bash
+ 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 = "datanic501-1"
+ private_ip_address = (known after apply)
+ private_ip_addresses = (known after apply)
+ resource_group_name = "test-eme"
+ 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 = "10.2.192.94"
+   private_ip_address_allocation = "Static"
+   private_ip_address_version = "IPv4"
+   subnet_id = "/subscriptions/b10209ad-ad22-4c26-8aef-be93b2f0bb58/resourceGroups/saDevNetwork/providers/Microsoft.Network/subnets/dinternal"
}

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

```

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

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

| | |
|------------------------|------|
| None | Copy |
| Terraform apply yes | |

10. Once the resources have applied successfully you should see output similar to this:

```

Bash
+ gateway_load_balancer_frontend_ip_configuration_id = (known after apply)
+ name = "dinternal"
+ primary = (known after apply)
+ private_ip_address = "10.2.192.94"
+ private_ip_address_allocation = "Static"
+ private_ip_address_version = "IPv4"
+ subnet_id = "/subscriptions/b10209ad-ad22-4c26-8aef-be93b2f0bb58/resourceGroups/saDevNetwork/providers/Microsoft.Network/subnets/dinternal"
}

Plan: 3 to add, 0 to change, 0 to destroy.
azurerm_network_interface.control_network_interface[0]: Creating...
azurerm_network_interface.data_network_interface[0]: Creating...
azurerm_network_interface.control_network_interface[0]: Creation complete after 2s [id=/subscriptions/b10209ad-ad22-4c26-8aef-be93b2f0bb58/resourceGroups/test-eme/providers/Microsoft.Network/interfaces/control-network-interface]
azurerm_network_interface.data_network_interface[0]: Creation complete after 2s [id=/subscriptions/b10209ad-ad22-4c26-8aef-be93b2f0bb58/resourceGroups/test-eme/providers/Microsoft.Network/interfaces/data-network-interface]
azurerm_linux_virtual_machine.CloudSwxtch[0]: Creating...
azurerm_linux_virtual_machine.CloudSwxtch[0]: Still creating... [10s elapsed]
azurerm_linux_virtual_machine.CloudSwxtch[0]: Still creating... [20s elapsed]
azurerm_linux_virtual_machine.CloudSwxtch[0]: Still creating... [30s elapsed]
azurerm_linux_virtual_machine.CloudSwxtch[0]: Still creating... [40s elapsed]
azurerm_linux_virtual_machine.CloudSwxtch[0]: Still creating... [50s elapsed]
azurerm_linux_virtual_machine.CloudSwxtch[0]: Creation complete after 51s [id=/subscriptions/b10209ad-ad22-4c26-8aef-be93b2f0bb58/resourceGroups/test-eme/providers/Microsoft.Compute/virtualMachines/cloudswxtch]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
eme@Azure:~/Swxtch/staticip$

```

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 StaticIPs then you just need to make some small changes to the **azure_deploySwxtch.tf** & **azure_var_network.tf** files.

Un-comment the Parameter **privateipaddress** 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 }
```

And 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 **azure_var_network.tf** file will have variables defined for your control and data NIC StaticIP definitions, you can update those values based on your subnet setup.

```
50 #8 - Control Nic Static IP
51 variable "controlnic_staticip" {
52   description = "private ip address for control nic"
53   type = string
54   default = "10.2.128.93"
55 }
56
57 #9 - Data Nic Static IP
58 variable "datanic_staticip" {
59   description = "private ip address for data nic"
60   type = string
61   default = "10.2.192.94"
62 }
63
```

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

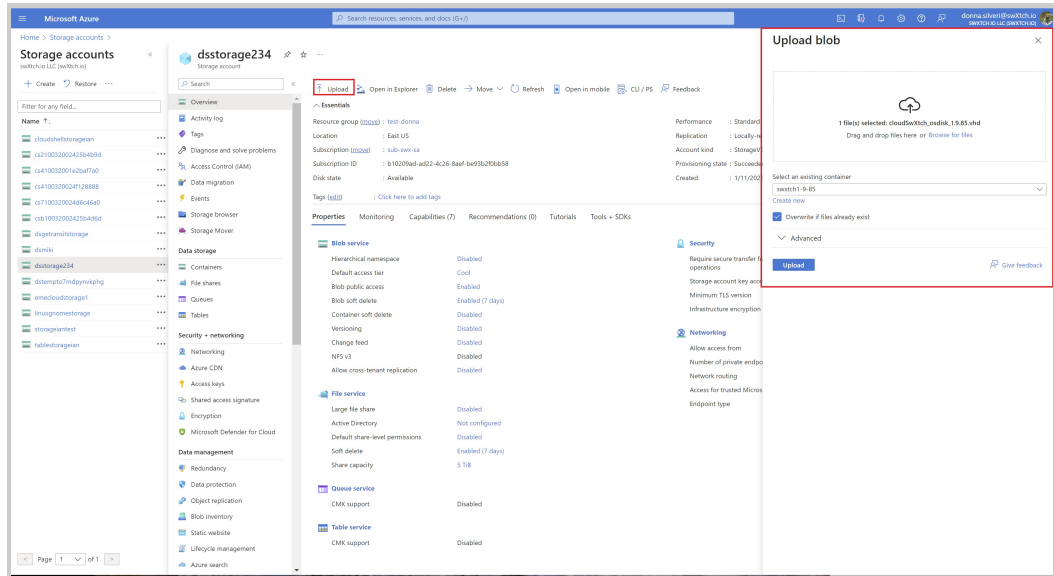
Log onto an environment that has access to the internet and download the following file (~30GB):

| | |
|---|------|
| 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.

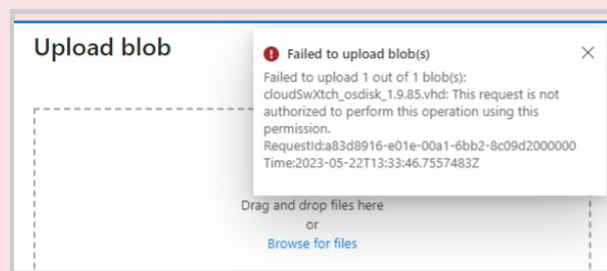


6. Start the upload and wait for it to complete.

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 >](#)

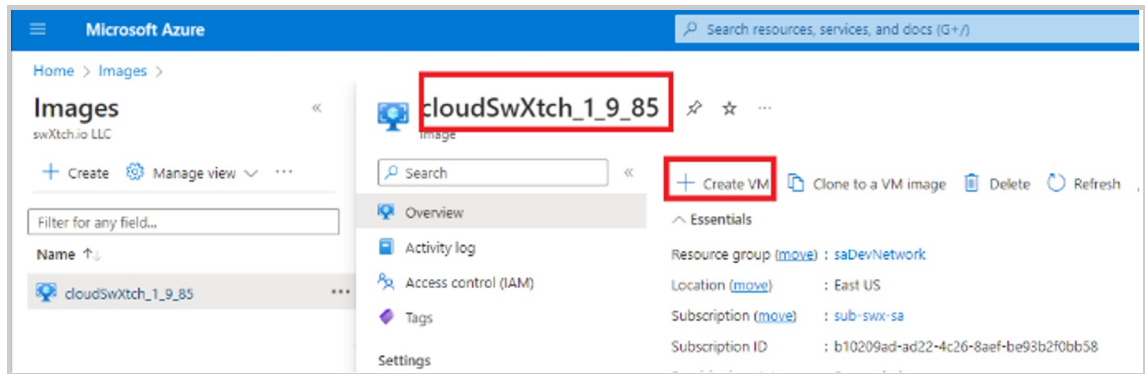
10. If tags are desired, then select **Tags** and enter the required tags.
11. The other fields can be left as default.
12. Select **Review and create**.
13. When validation passes, select **Create**. When it is complete, click **Go to Resource** to see the image.

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**.

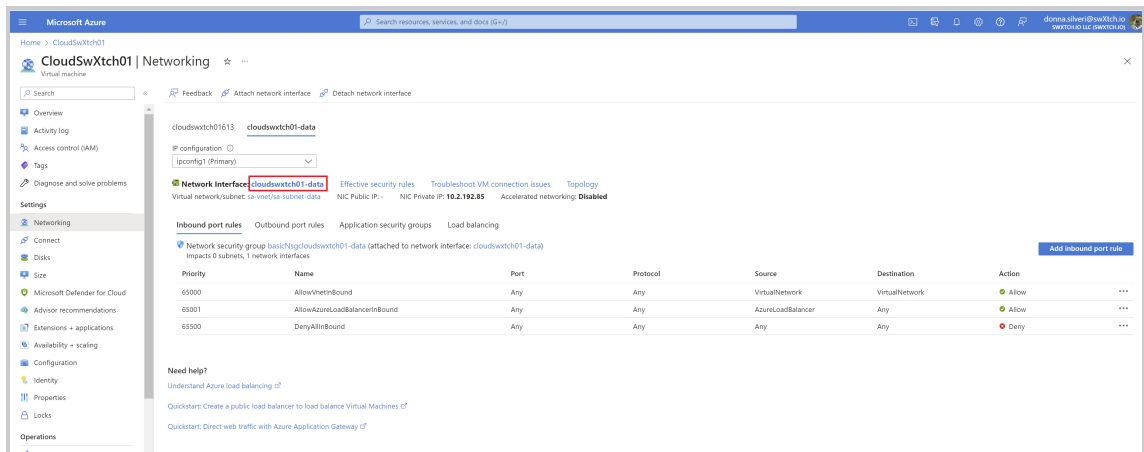
10. When it is done, refresh the screen. There should now be a control and data interface.

The screenshot shows the 'CloudSwXtch01 | Networking' page in the Microsoft Azure portal. The page has a sidebar with navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Networking, Connect, Disks, Size, Microsoft Defender for Cloud, Advisor recommendations, Extensions + applications, Availability + scaling, Configuration, Identity, Properties, Locks, and Operations. The main content area shows the 'cloudswtch01' virtual machine. The 'Network interface' section is highlighted, showing 'cloudswtch01-data' as the selected network interface. Below this, the 'Inbound port rules' section is expanded, showing a table of rules. The table has columns for Priority, Name, Port, Protocol, Source, Destination, and Action. The rules are:

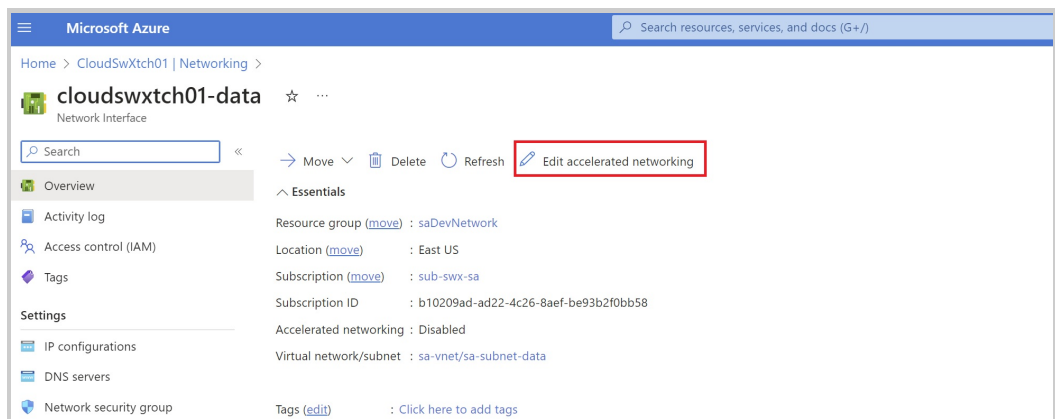
| Priority | Name | Port | Protocol | Source | Destination | Action |
|----------|-------------------------------|------|----------|-------------------|----------------|--------|
| 65000 | AllowVnetInbound | Any | Any | VirtualNetwork | VirtualNetwork | Allow |
| 65001 | AllowAzureLoadBalancerInbound | Any | Any | AzureLoadBalancer | Any | Allow |
| 65500 | DenyAllInbound | Any | Any | Any | Any | Deny |

Below the table, there is a 'Need help?' section with links to 'Understand Azure load balancing', 'Quickstart: Create a public load balancer to load balance Virtual Machines', and 'Quickstart: Direct web traffic with Azure Application Gateway'.

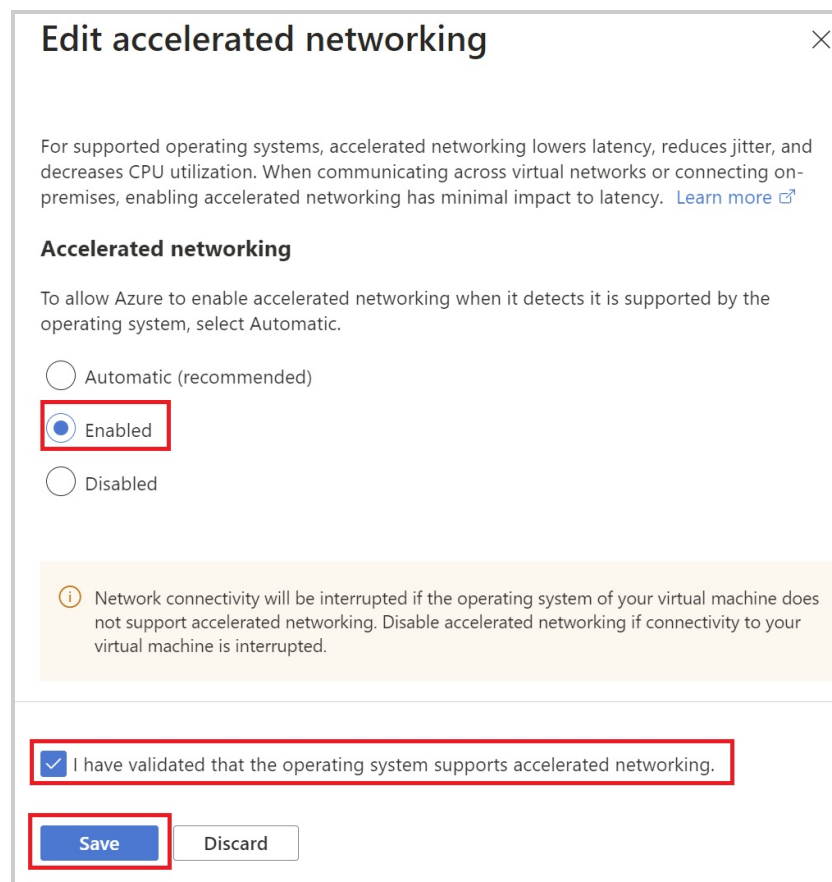
11. Select the data **Network Interface**.



1. Select Edit accelerated Networking.



2. A new window will display.

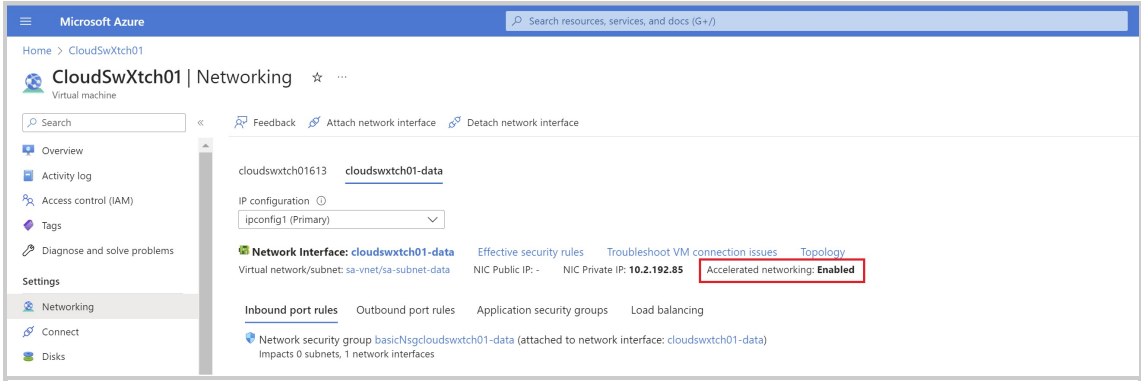


3. Select Enabled.

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 send it to 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

License File

Max Bandwidth

Max Clients

100000 Mbps

30

Totals

Producers

Bridge RX

Mesh RX

Switch RX

0 pps

0 pps

0 pps

0 pps

(0 bps)

(0 bps)

(0 bps)

(0 bps)

Consumers

Bridge TX

Mesh TX

Switch TX

0 pps

0 pps

0 pps

0 pps

(0 bps)

(0 bps)

(0 bps)

(0 bps)

xNIC clients

Name

ip

Version (XNIC)

RX pps

RX bps

TX pps

TX bps

HA

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

cloudSwXtch on 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

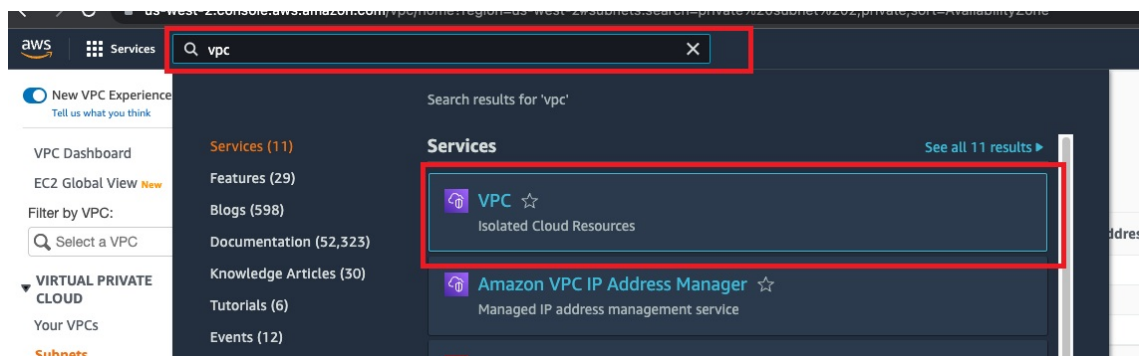
A virtual network must be created before deploying a cloudSwXtch EC2 instance.

- It must contain **two** subnets: one that's used for control plane communication and another for data plane communication.
 - It is recommended that both subnets are **private facing** and **do not auto-assign public IPs**.
- Both subnets need to be in the same Availability Zone (AZ). This allows be both NICs to be connected on the EC2 instance at the same time.
- The subnets must be the same subnets used for the xNIC installations.

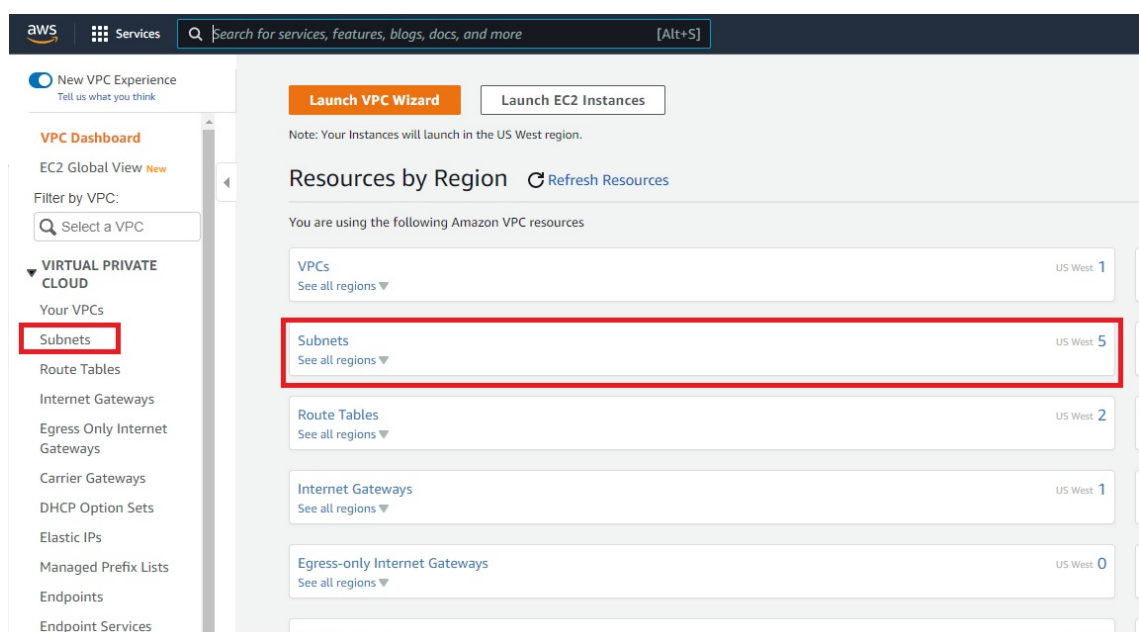
In this section, users will learn how to create both the control and the data subnets for their virtual network in preparation for cloudSwXtch installation on AWS.

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.



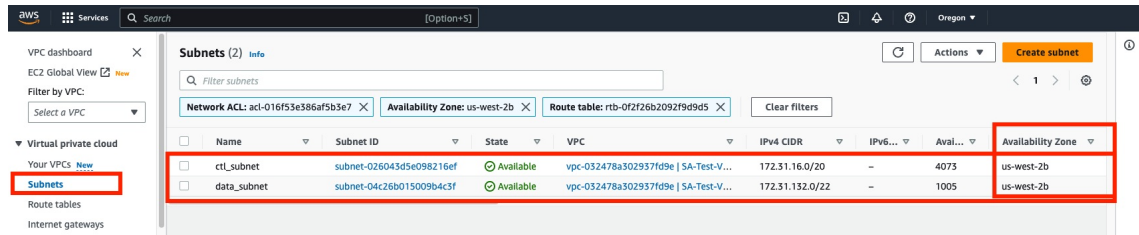
2. **Select "Subnets"** under the Virtual Private Cloud tab or under Resources by Region in the VPC Dashboard.



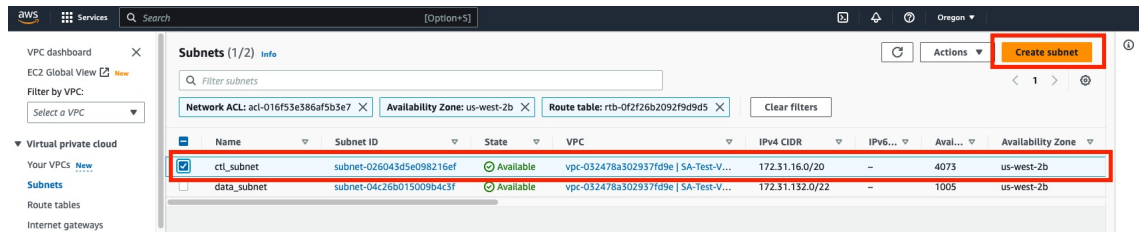
3. Check that the data and control plane subnets are in the same Availability Zone.

PRO TIP

For ease of use, name the subnets as ctrl-subnet and data-subnet to distinguish them when creating an EC2 instance.



4. Create a 2nd subnet if it does not exist by selecting 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 and Availability Zone as your other 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 | Remove |

Add new tag

You can add 49 more tags.

Remove

Add new subnet

Cancel **Create subnet**

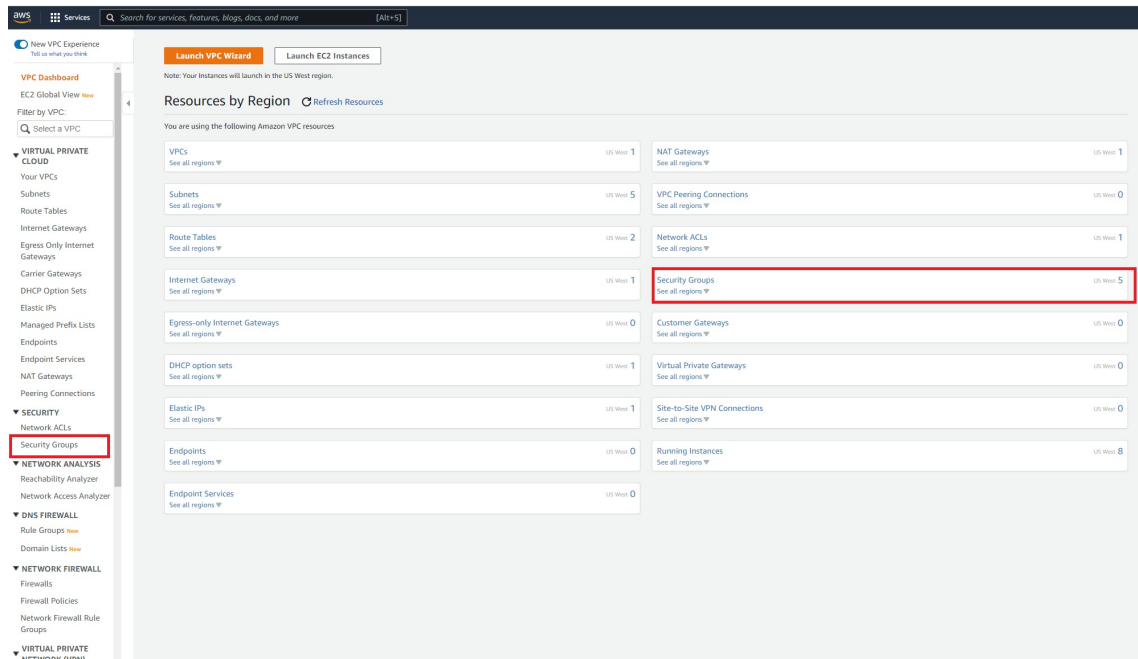
6. Click "Create Subnet." You should now have a new subnet in your list.

Verify Security Groups

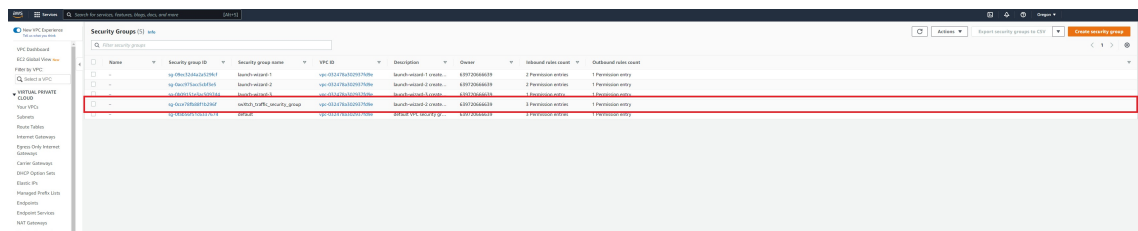
The security group contains the firewall settings for EC2 instances and interfaces (xNICs).

To ensure security groups are set up properly for cloudSwXtch:

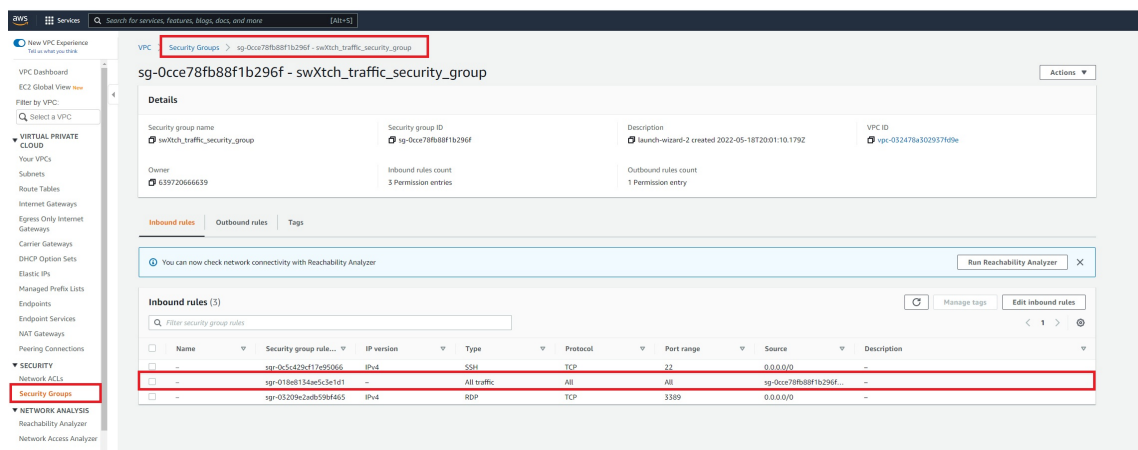
1. Navigate to the VPC console.
2. Select the "Security Groups" link as shown below. (Note: There are multiple ways to get to the "Security Groups" page.)



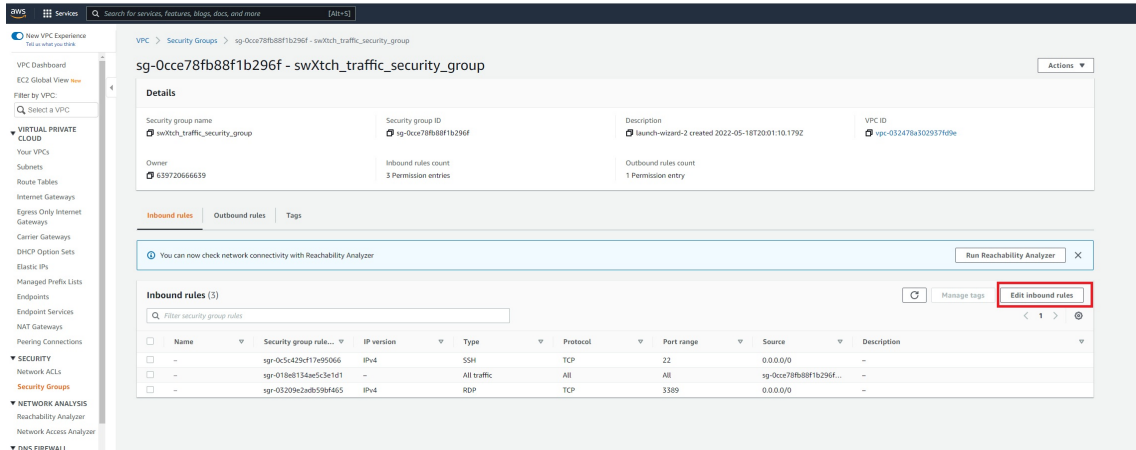
3. Select the Security Group that is normally used to create your EC2 instances for your application. (Note: The names in the example will be different in your environment.)



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."

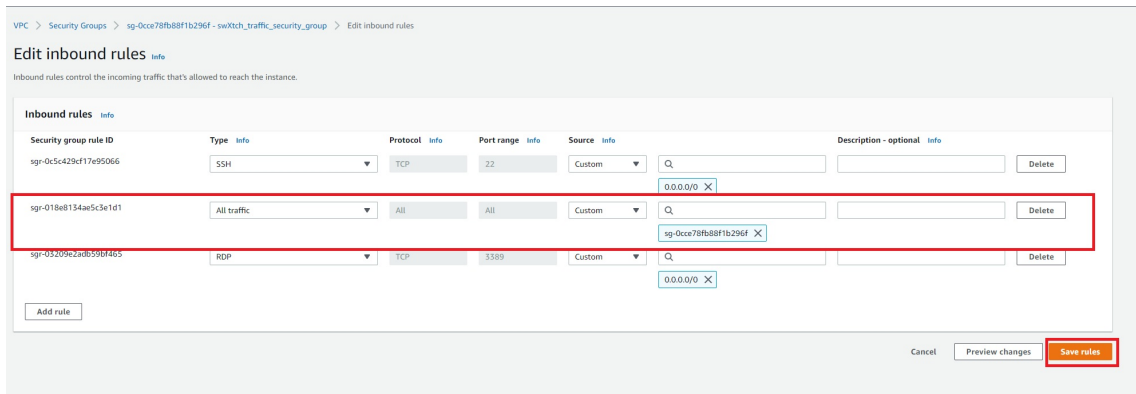


5. If an inbound rule does not exist, create it by selecting "Edit inbound rules."



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.



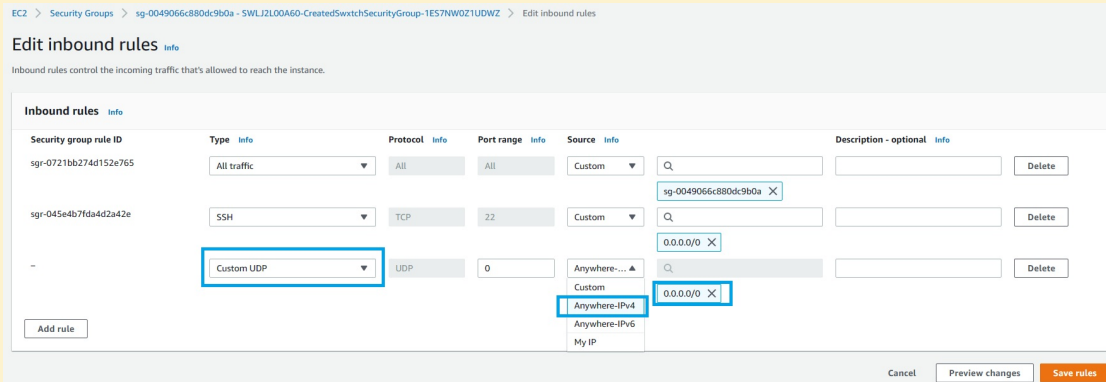
8. Save the rule.

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

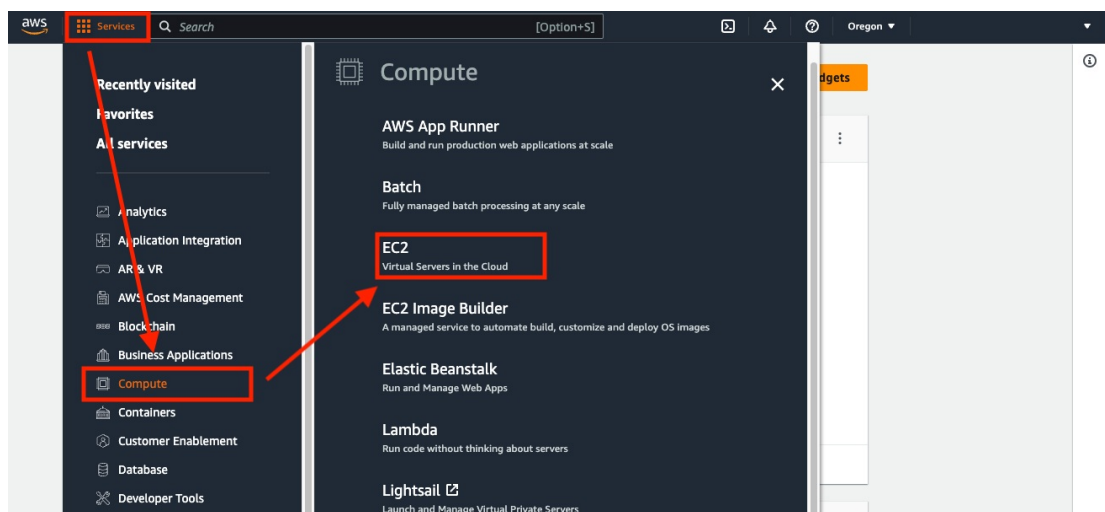
WHAT TO EXPECT

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.

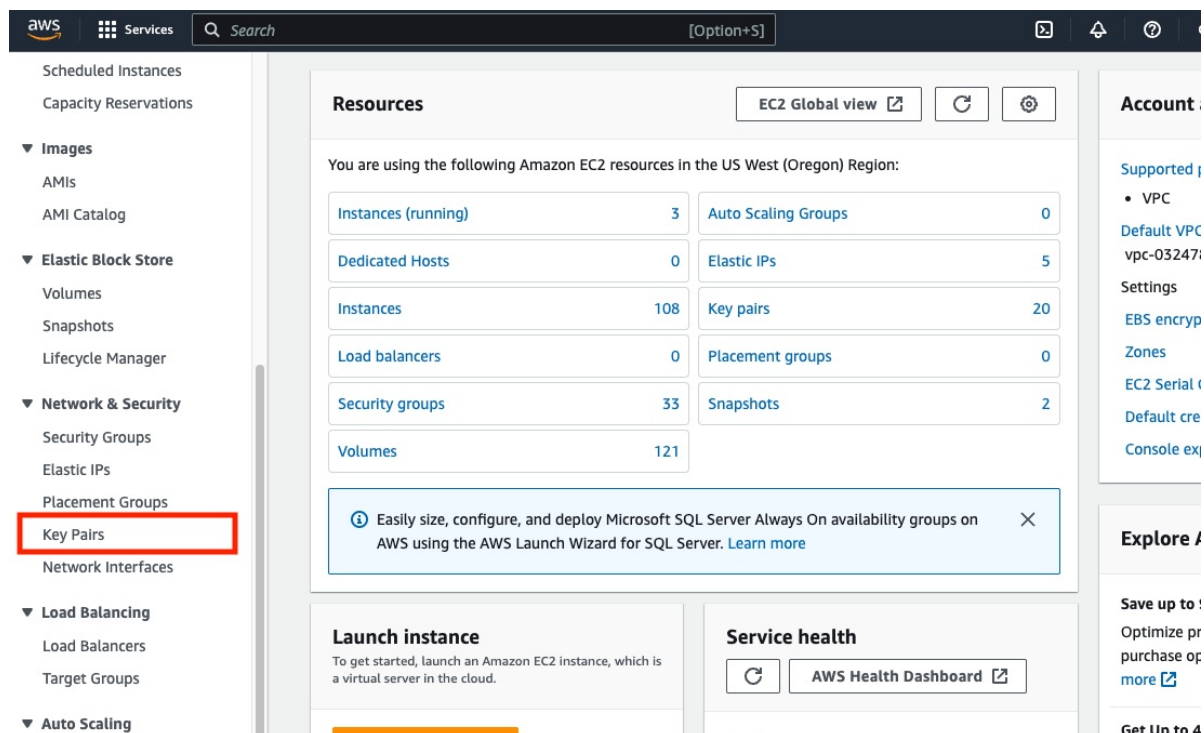
In the *AWS Management Console*, make sure you are in the region where you plan to use the cloudSwXtch instance.

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).

The screenshot shows the 'Create key pair' page in the AWS Management Console. The breadcrumb navigation at the top reads 'EC2 > Key pairs > Create key pair'. The main heading is 'Create key pair' with an 'Info' link. Below this is a section titled 'Key pair' with a descriptive paragraph: 'A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.' The form contains several fields and options: a 'Name' field with a placeholder 'Enter key pair name' and a note that the name can include up to 255 ASCII characters but no leading or trailing spaces; a 'Key pair type' section with radio buttons for 'RSA' (selected) and 'ED25519', accompanied by an 'Info' link; a 'Private key file format' section with radio buttons for '.pem' (selected, 'For use with OpenSSH') and '.ppk' ('For use with PuTTY'); a 'Tags - optional' section stating 'No tags associated with the resource' and an 'Add new tag' button with a note 'You can add up to 50 more tags.' At the bottom right are 'Cancel' and 'Create key pair' buttons.

EC2 > Key pairs > Create key pair

Create key pair [Info](#)

Key pair

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

Name

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

Key pair type [Info](#)

☒ RSA

☐ ED25519

Private key file format

☒ .pem
For use with OpenSSH

☐ .ppk
For use with PuTTY

Tags - optional

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

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

Install cloudSwXtch on AWS

WHAT TO EXPECT

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

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

NOTE:

Root privileges are not required for deployment or operation. Our CloudFormation template allows an automated mechanism to update the installed cloudSwXtch version. This will deploy the latest version versus the one packaged in the AML, which requires root privileges to trigger the update from the product side. For upgrades, please see [CloudSwXtch Upgrade](#) on how to perform an upgrade from the client side. An upgrade from the client side does not require root privileges.

Creating a cloudSwXtch EC2 Instance

Prerequisites

Before starting, a user must do the following:

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

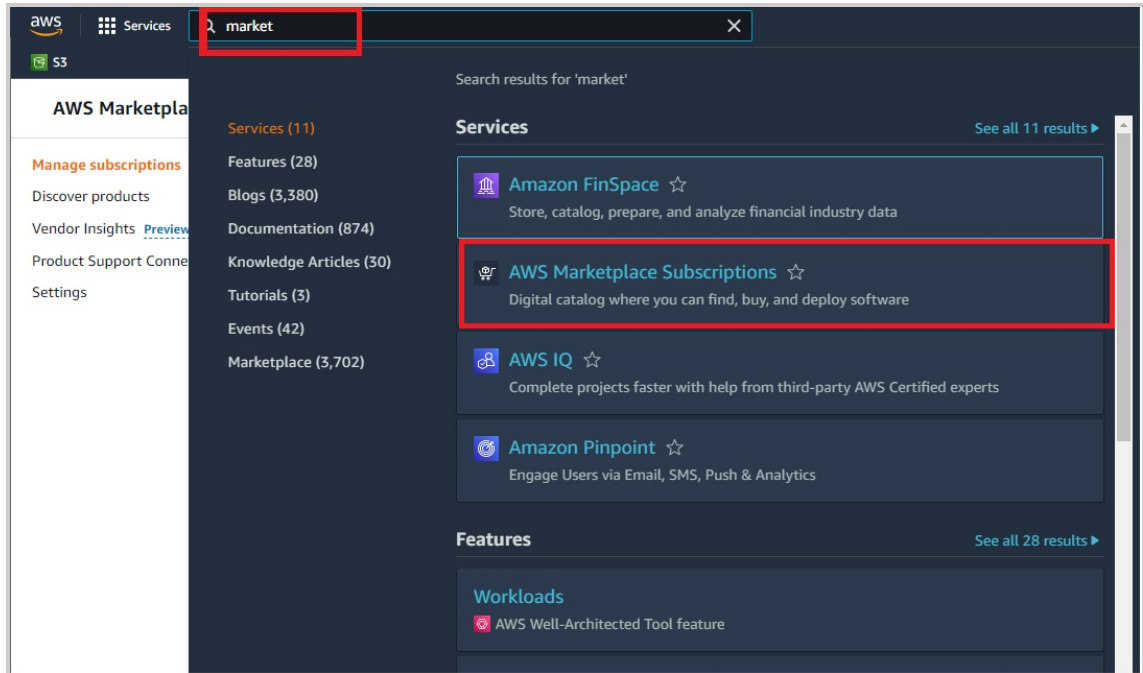
If all prerequisites are met, a cloudSwXtch can be created via the Marketplace in any region in approximately 10 minutes. If multi-AZ or multi-region is required then see [Mesh](#) for details. The installer will create a CloudFormation Stack to include the following resources:

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

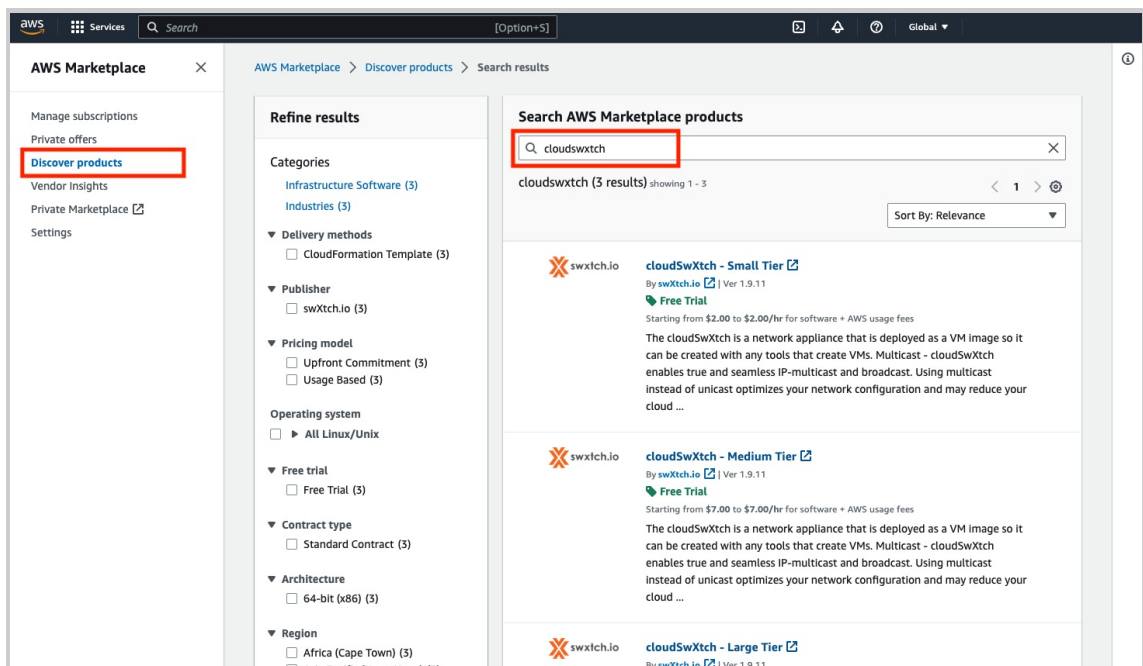
In order to create a cloudSwXtch, please do the following steps.

1. Sign into AWS.

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



3. Select "Discover Products" in the AWS Marketplace menu on the left hand side.
4. Search for "cloudSwXtch."



5. Select a Tier (Small, Medium or Large) based on your usage requirements and features needed. Please read the [Quotas](#) article for more information regarding cloudSwXtch sizing.

Endpoint Connections Limit

Be mindful of the number of endpoints you connect to your cloudSwXtch after creation. For example, by selecting the "Small" tier, you will be limited to 10 endpoint connections. If you know you will need more than that, consider deploying a medium (50 endpoints) or large (200 endpoints) sized cloudSwXtch.

If you need to increase the number of endpoints, please view the AWS instructions [here](#). Note that if your new instance type exceeds the size of your tier, you must contact support@swxtch.io to update your license.

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

The screenshot shows the AWS Marketplace product page for 'cloudSwXtch - Small Tier' by swxtch.io. The page includes a search bar, navigation links, and a 'Continue to Subscribe' button highlighted with a red box. Below the product name, there is a 'Free Trial' badge and a 'Typical Total Price' section. The 'Product Overview' section describes the product as a virtual overlay network. A 'Highlights' box lists three key features: Plug and Play, Simplified Workflows, and wXcked Fast. The 'Version' section shows the latest version is 1.9.11, with a link to 'Show other versions'.

7. Review the Terms and Conditions.
8. Select "Accept Terms" if they are acceptable.

The screenshot shows the 'Subscribe to this software' page for cloudSwXtch - Small Tier. It includes a 'Continue to Configuration' button with a note: 'You must first review and accept terms.' The 'Terms and Conditions' section is titled 'swXtch.io Offer' and contains a detailed text block about the subscription terms. A red box highlights the 'Accept Terms' button. At the bottom, there is a note about pricing information for software components.

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

The screenshot shows the AWS Marketplace interface for the product 'cloudSwXtch - Small Tier' by swtch.io. The page is titled 'Subscribe to this software' and includes a 'Continue to Configuration' button highlighted with a red box. The page also displays a 'Software contract' summary, 'License management' information, and 'Terms and Conditions'.

Software contract

Select contract option(s)

Total contract price \$0 Due now

Create Contract

License management

Your purchase of this product created a license that you can manage in AWS License Manager. This may include viewing, granting access, and tracking usage of your entitlements.

Manage License

Terms and Conditions

swtch.io Offer

You have subscribed to this software and agreed that your use of this software is subject to the pricing terms and the seller's [End User License Agreement \(EULA\)](#). You agreed that AWS may share information about this transaction (including your payment terms) with the respective seller, reseller or underlying provider, as applicable, in accordance with the [AWS Privacy Notice](#). AWS will issue invoices and collect payments from you on behalf of the seller through your AWS account. Your use of AWS services remains subject to the [AWS Customer Agreement](#) or other agreement with AWS governing your use of such services.

| Product | Effective date | Expiration date | Action |
|--------------------------|----------------|-----------------|------------------------------|
| cloudSwXtch - Small Tier | 9/30/2022 | N/A | Show Details |

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

The screenshot shows the AWS Marketplace configuration page for 'cloudSwXtch - Small Tier'. The page has a dark header with the AWS Marketplace logo and navigation links. Below the header, the product name 'cloudSwXtch - Small Tier' is displayed. A red box highlights the 'Continue to Launch' button in the top right corner. The main content area is titled 'Configure this software' and includes a section for 'Choose a fulfillment option and software version to launch this software.' This section contains two dropdown menus: 'Fulfillment option' (set to 'Swxtch Configuration') and 'Software version' (set to '1.9.11 (Dec 08, 2022)'). Below these, a 'Region' dropdown menu is highlighted with a red box, showing 'US West (Oregon)'. To the right of the configuration options, there are two panels: 'Software contract' and 'Pricing information'. The 'Software contract' panel shows a 'Total contract price' of '\$0' and a 'Create Contract' button. The 'Pricing information' panel provides details about the software pricing, including a 'Free Trial' offer for 'cloudSwXtch - Small Tier' running on 'm5zn.3xlarge'.

INSTANCE TYPES

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

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

11. Read "Usage Instructions" if you desire.
12. Use the "Choose Action" dropdown menu and select "Launch CloudFormation."

13. Click "Launch."

The screenshot shows the AWS Marketplace page for 'cloudSwXtch - Small Tier' by swtch.io. The page is titled 'Launch this software' and includes a navigation bar with links like 'Product Detail', 'Subscribe', 'Configure', and 'Launch'. The main content area displays configuration details: Fulfillment option (Swtch Configuration cloudSwXtch - Small Tier, running on m5.xlarge), Software version (1.9.11), and Region (US West (Oregon)). A 'Usage instructions' button is highlighted with a red box. Below this, the 'Choose Action' section shows a dropdown menu set to 'Launch CloudFormation'. At the bottom right, a yellow 'Launch' button is highlighted with a red box.

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

The screenshot shows the AWS CloudFormation 'Create stack' page. The page is divided into four steps: Step 1 (Create stack), Step 2 (Specify stack details), Step 3 (Configure stack options), and Step 4 (Review). The 'Specify template' section is active, showing the 'Prerequisite - Prepare template' and 'Specify template' sections. The 'Template source' section has 'Amazon S3 URL' selected. The 'Amazon S3 URL' field contains the URL: https://s3.amazonaws.com/awsmpl-fulfillment-cf-templates-prod/f59d984a-f550-4d3e-b86f-8f59564ab22b.57ff4ebf-ef7f-4e08-b0f7-fdbet. The 'Next' button is highlighted with a red box.

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

1. Under "Stack name," enter your desired name. Keep in mind that this will be used for everything added to the stack. For example: "resource name," "security groups," "EC2 instance name," etc.
2. Under "CidrIpForInboundOutboundTraffic," use 0.0.0.0/0 so that you can SSH to the virtual machine from any IP address. You can also pick a more restrictive range if desired.
3. Under "ControlSubnet," use the dropdown to find the control subnet you created (recommended: *ctrl-subnet*).
4. Under "DataSubnet," use the dropdown to find the data subnet you created (recommended: *data-subnet*).
5. For "InstanceType," there should be "Fulfillment" data from the earlier step.
6. Under "KeyName," use the dropdown to find your previously created or imported SSH key.
7. In "PassedSwxtchSecurityGroup," use "default" and one will be created during the installation process. Alternatively, you can enter the ID of an already created security group. It will be something similar to "sg-009273855418af38d."
8. Under "VpcId," select from the dropdown to find the already created VPC id.
9. Here is an example of how your template would look like:

The screenshot shows a web form titled "Specify stack details". It contains several sections:

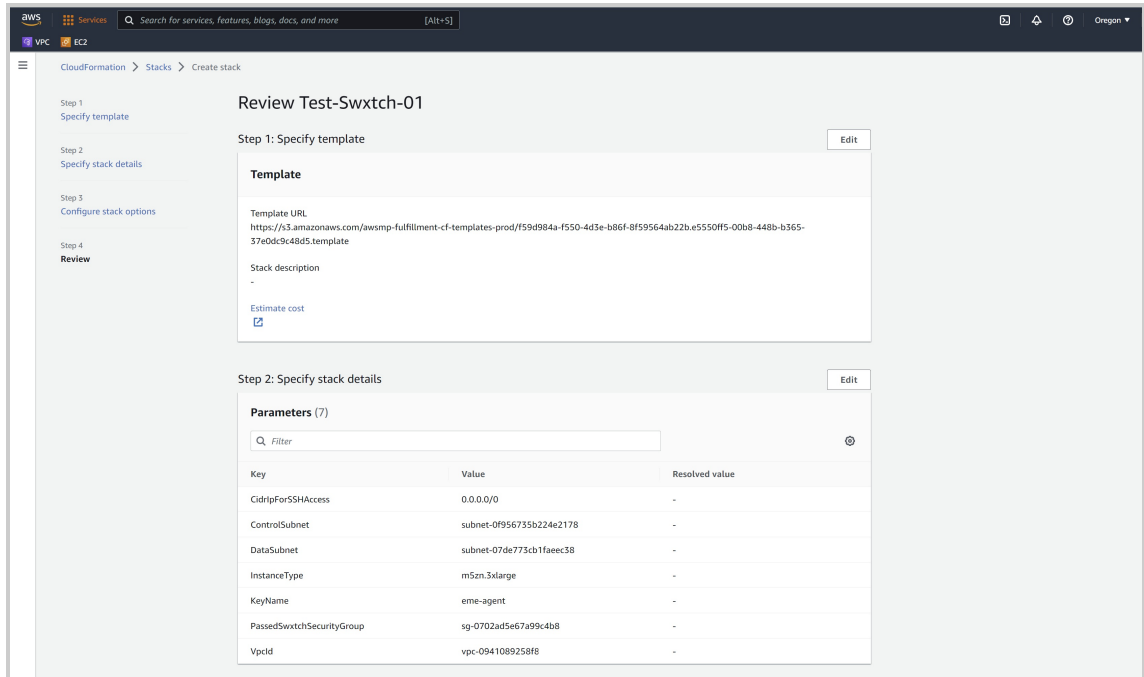
- Stack name:** A text input field with the value "Test-Swxtch-01". Below it is a note: "Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-)."
- Parameters:** A section with a subtitle "Parameters are defined in your template and allow you to input custom values when you create or update a stack." It contains several fields:
 - CidrIpForSSHAccess:** A text input field with the value "0.0.0.0/0". Above it is a note: "For SSH access when using default security group, please set CIDR to x.x.x.x/32 to allow one specific IP address access, 0.0.0.0/0 to allow all IP addresses access, or another CIDR range".
 - ControlSubnet:** A dropdown menu with the value "subnet-0f956735b224e2178". Above it is a note: "Used to create control ENI assigned to the swxtch control subnet."
 - DataSubnet:** A dropdown menu with the value "subnet-07de773cb1faec38". Above it is a note: "Used to create control ENI assigned to the swxtch control subnet."
 - InstanceType:** A text input field with the value "m5zn.3xlarge". Above it is a note: "EC2 instance type to use for swXtch creation".
 - KeyName:** A dropdown menu with a redacted value. Above it is a note: "SSH key name for swXtch SSH access".
 - PassedSwxtchSecurityGroup:** A text input field with the value "sg-0702ad5e67a99c4b8". Above it is a note: "Name of the security group that should be used by swXtch and created ENIs".
 - VpcId:** A dropdown menu with the value "vpc-0941089258f89c9a2". Above it is a note: "The VPC id your swXtch will be placed in".
- Navigation:** At the bottom right, there are three buttons: "Cancel", "Previous", and "Next".

16. Click "Next."

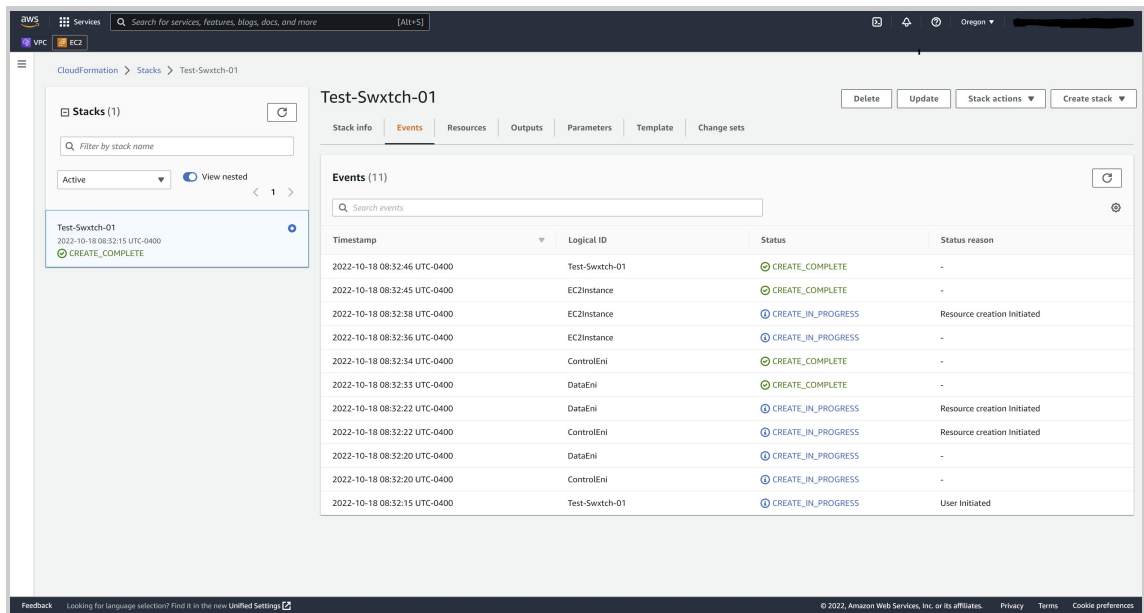
17. The "Configuring stack options" page is completely optional. You can assign tags for your stack, set additional IAM permissions, stack failure options, etc.

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

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



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



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

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

Text

| | |
|---|------|
| None | Copy |
| sudo swxtch/swxtch-top dashboard --switch <swXtch-ip> | |

NOTE

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

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

INSTALLING AN XNIC

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

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

You can find more information about xNIC installation, [here](#).

Checking the Health of Your cloudSwXtch Instance

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

Upgrading cloudSwXtch on AWS

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

Deleting cloudSwXtch on AWS

To learn how to delete your cloudSwXtch, you can read more [here](#).

Upgrade cloudSwXtch on AWS

cloudSwXtch upgrade

When new versions are available in the Market Offering, complete the following steps to upgrade cloudSwXtch:

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 v1.9.16 --ip 10.5.1.6</pre> | |

Upgrade

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

Delete cloudSwXtch on AWS

WHAT TO EXPECT

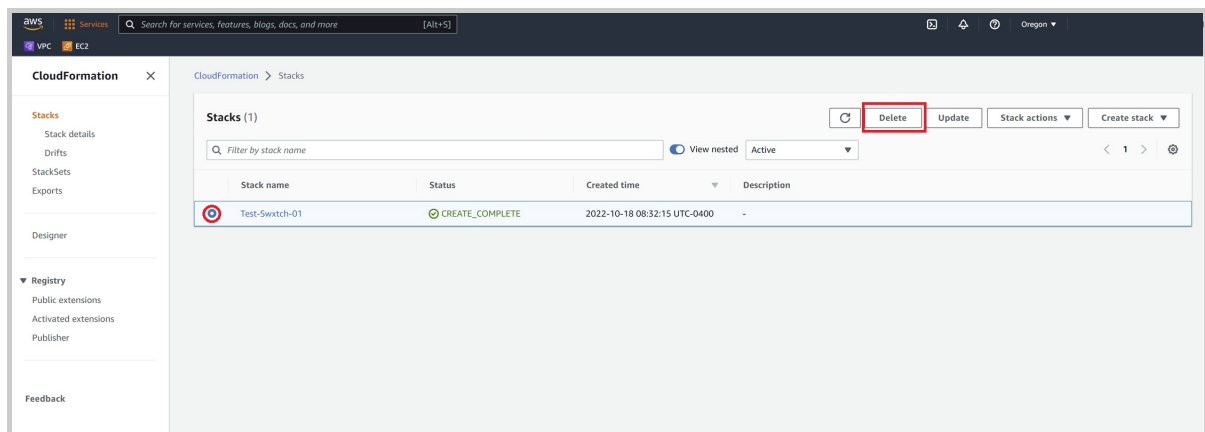
In this section, users will learn how to delete cloudSwXtch from their AWS environment.

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."

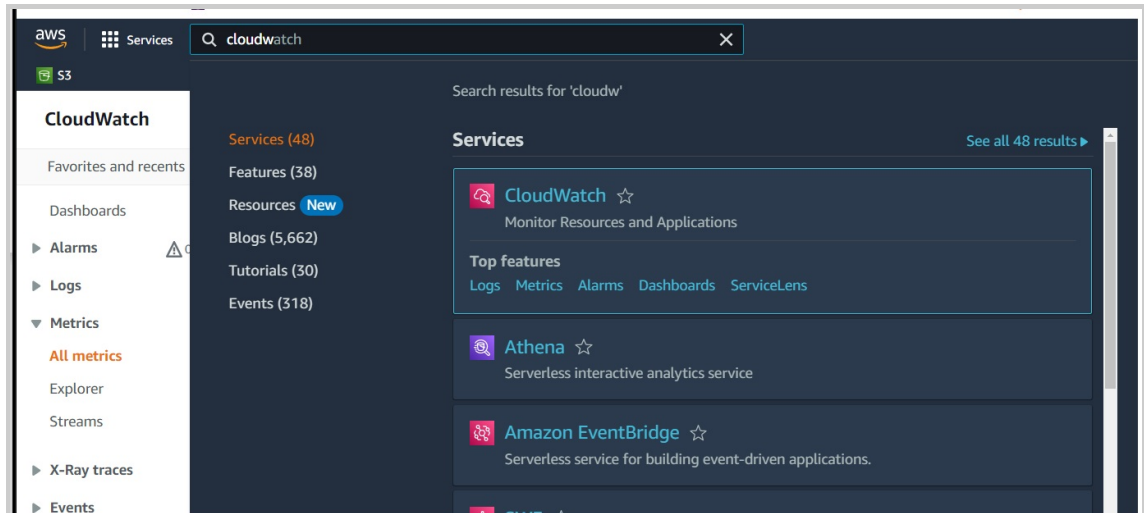


2. **Select** the stack you want to delete.
3. **Click "Delete"** and then confirm on the popup window.
4. **Refresh** the page after a minute or so to confirm the stack has been deleted.

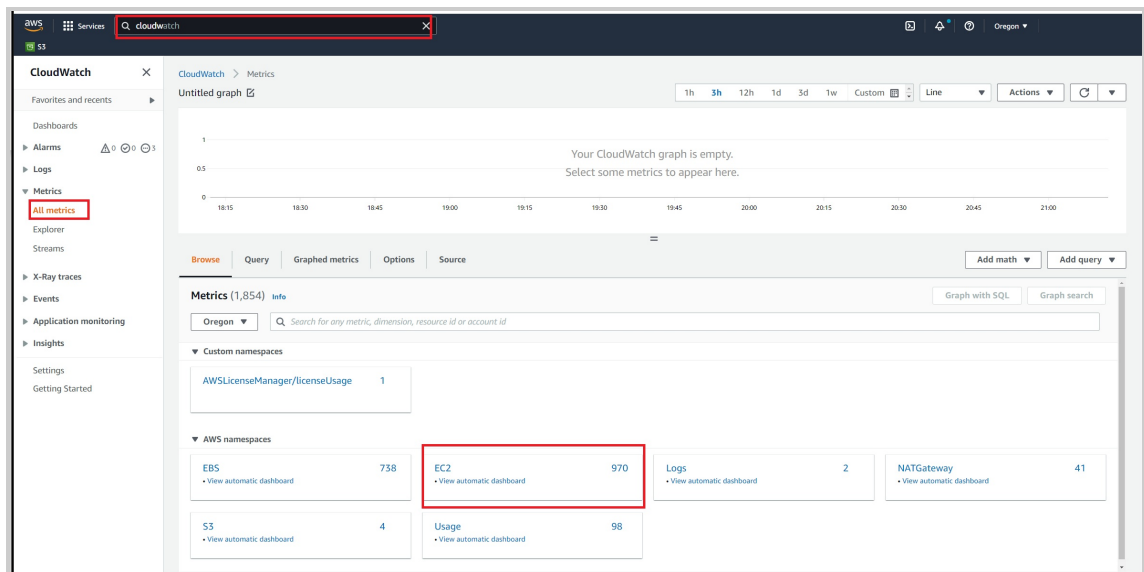
Check Health of cloudSwXtch Instance on AWS

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:

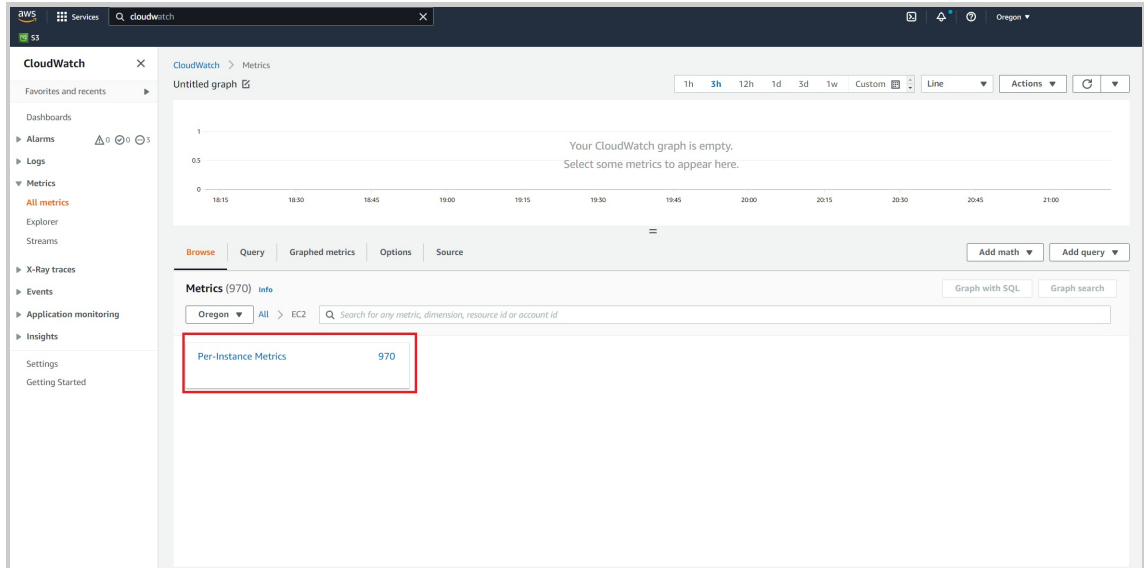
1. Search for "CloudWatch" in the AWS Search bar.



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

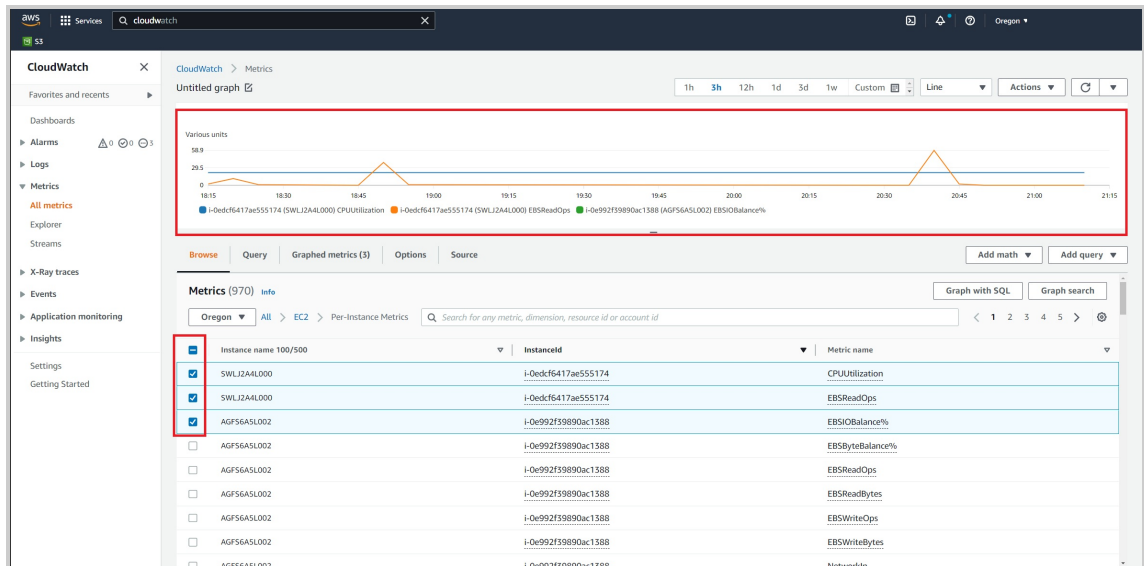


4. Select "Per-Instance Metrics."



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.

Installing cloudSwXtch Bridge

WHAT TO EXPECT

There are currently 2 variations of cloudSwXtch Bridges: V1 and V2. It is suggested for users to use cloudSwXtch Bridge V2 for most cases. However, there is still some benefit to adopting cloudSwXtch Bridge V1.

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

Bridge V2

Variation 2 of the cloudSwXtch Bridge **supports bi-directional traffic between on-prem and the cloud**. Additionally, it supports 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 V2](#) for installation instructions.

Bridge V1

Variation 1 of the cloudSwXtch Bridge **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. However, it may still be useful for testing scenarios where there is no VPN or Express Route only access via Internet. cloudSwXtch Bridge V2 does not support internet only access. It requires a VPN or Express Route.

See [Install cloudSwXtch Bridge V1](#) for installation instructions.

Install cloudSwXtch Bridge V2

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, CentOS8 with Kernel of 5.15 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 Ctrl IP <-> Bridge Ctrl IP 37856 (TCP+UDP)
- cloudSwXtch Data IP <-> Bridge Data IP 9999 (UDP)

Pre-Installation: Update Ubuntu 20.04 to Kernel 5.15

Use the following commands:

| | |
|---|------|
| None | Copy |
| <pre>sudo apt-get install linux-generic-hwe-20.04</pre> | |

| | |
|--|------|
| None | Copy |
| <pre>apt update && sudo apt full-upgrade</pre> | |

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/>

After running the commands, use the following to check the kernel version is at 5.15:

| | |
|-----------------------------------|------|
| None | Copy |
| <pre>>uname -r</pre> | |
| <p>OUTPUT:</p> <p>5.15.0-1023</p> | |

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> | |

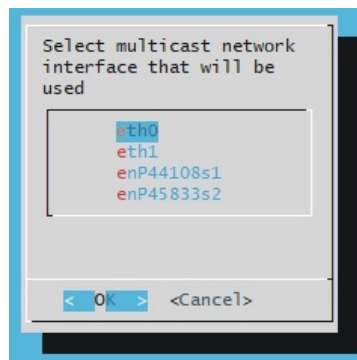
1. **If the ping fails to find the cloudSwXtch instance by name**, try pinging the IP address of the cloudSwXtch instance. If the IP works, they use the IP address in place of the name in all future commands. This can happen if the default DNS settings are changed for the Virtual Network.

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 V2 Commands

After deploying your cloudSwXtch Bridge V2, 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</pre> | |

START

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

RESTART

| | |
|--|------|
| ActionScript | Copy |
| <pre>sudo systemctl restart swxtch-bridge2</pre> | |

Please note: `swXtch-bridge2` can also be replaced with `swXtch-bridge2.service` and still work as expected.

Uninstalling cloudSwXtch Bridge V2

To uninstall your cloudSwXtch Bridge V2 application from your bridge host machine:

1. Use the **STOP** command to stop the bridge.
2. Execute the following command on the Bridge VM on-prem:

| | |
|---|------|
| ActionScript | Copy |
| <pre>sudo apt-get remove swxtch-bridge2</pre> | |

Your cloudSwXtch Bridge V2 instance should now be uninstalled.

Install cloudSwXtch Bridge V1

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 20.04+ with a minimum of 4 cores, 8GB 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.

Installation option #1: 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> | |

Installation option #2: Download of installer -V1

With this method, the bridge installer file (deb or rpm) is downloaded from any VM that has network connectivity to the SDMC switch. The installer file can then be run manually on the bridge host machine at a later time.

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 download script:

1. Ubuntu 18.04

Text

| None | Copy |
|--|------|
| <pre>wget http://<swxtch-instance-name>/services/install/swxtch-bridge_1.0.0_ubuntu18.04_amd64.deb</pre> | |

2. Ubuntu 20.04

Text

| None | Copy |
|--|------|
| <pre>wget http://<swxtch-instance-name>/services/install/swxtch-bridge_1.0.0_ubuntu20.04_amd64.deb</pre> | |

3. RHEL/CentOS

Text

| None | Copy |
|---|------|
| <pre>wget http://<swxtch-instance-name>/services/install/swxtch-bridge-1.0.0-centos8-1.x86_64.rpm</pre> | |

4. Run the installer packet on the bridge host:

1. Ubuntu 18.04

Text

| | |
|---|------|
| None | Copy |
| <pre>sudo dpkg -i swxtch-bridge_1.0.0_ubuntu18.04_amd64.deb</pre> | |

2. Ubuntu 20.04

Text

| | |
|---|------|
| None | Copy |
| <pre>sudo dpkg -i swxtch-bridge_1.0.0_ubuntu20.04_amd64.deb</pre> | |

3. RHEL/CentOS 8.1

Text

| | |
|---|------|
| None | Copy |
| <pre>sudo rpm -i swxtch-bridge-1.0.0-1.x86_64.rpm</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 2016+
- **CPU architecture:** x86_x64
- **Network connectivity:** 2 NICs (one for each sub-net: ctrl-subnet and data-subnet)

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 swxtch-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 swxtch tunnel interface (swxtch-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. 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 in the Linux systems.

Installing xNIC for Linux

BEFORE YOU START

1. Review [xNIC System requirements](#).
2. Ensure the following:
 1. The host VM must have at least two NICs.
 2. The NICs must be on the same subnets for control and data as the cloudSwXtch.
 3. The ctrl-subnet should be assigned to the primary NIC.

Choosing an xNIC Version

There are two different xNIC versions for Linux: xNIC1 and xNIC2. xNIC2 is preferred but it is not available for all Linux versions. swXtch.io's xNIC requires an OS from this list:

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

For Redhat RHEL users, this following is required:

| | |
|--|------|
| None | Copy |
| <pre>sudo firewall-cmd --zone=public --add-port=10800/udp --permanent sudo firewall-cmd --zone=public --add-port=9999/udp --permanent sudo firewall-cmd --reload</pre> | |

3. If you selected xNIC Version 1 (RHEL 7+, CentOS 7+, or Ubuntu 18.04 | 20.04), run the following installer script:

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

4. If you selected xNIC Version 2 (RHEL 8, CentOS 8, or Ubuntu 20.04), run the following installer script:

| | |
|--|------|
| None | Copy |
| <pre>curl http://<swxtch-instance-name>/services/install/swxtch-xnic-install.sh bash -s - - -v 2</pre> | |

5. 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@D5d-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:

V1

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

V2

| | |
|---|------|
| None | Copy |
| <pre>curl http://<swxtch-instance-name>/services/install/swxtch-xnic-install.sh bash -s - - -k -v 2</pre> | |

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

1. Review xNIC System Requirements.
2. Ensure the following:
 1. The host VM must have at least 2 NICs.
 2. The NICs must be on the same subnets for control and data as the cloudSwXtch.
 3. The ctrl-subnet should be assigned as the primary NIC.

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

```
ping <switch-instance-name>
```


Ping Fails

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

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

3. Remove any firewall restrictions to UDP ports 10800 and 9999. The cloudSwXtch sends UDP packets to these ports as part of normal operation.

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
```

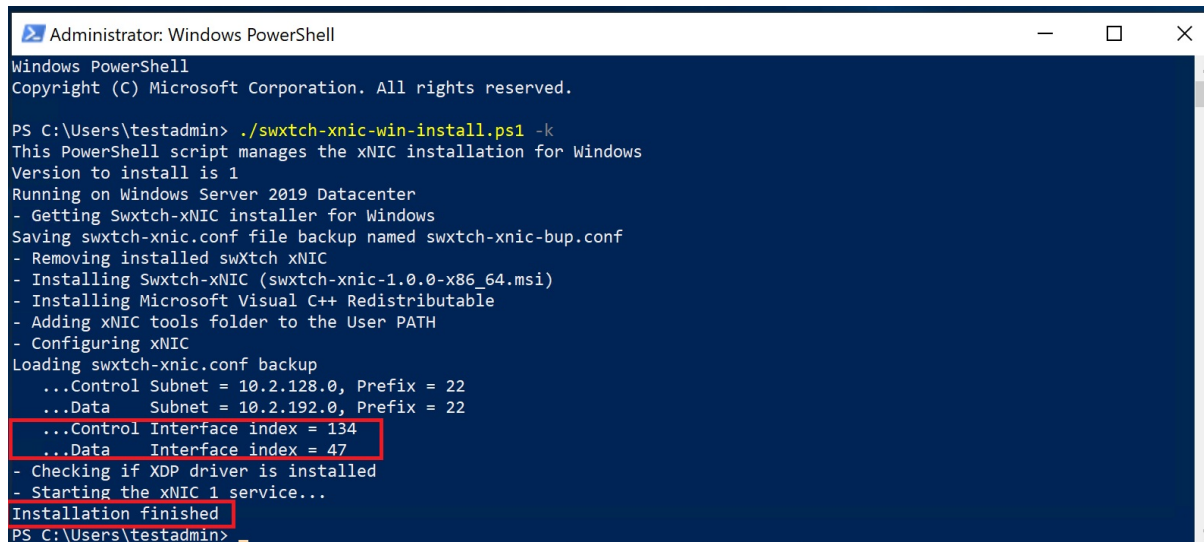
5. Run the installer script. Note: The installer script will differ depending on the version you decided to use.

None

Copy

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

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 commands below instead for uninstall.

For xNIC 1:

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> | |

For xNIC 2:

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 -v 2</pre> | |

Upgrade xNIC on Windows

WHAT TO EXPECT

When a cloudSwXtch has been updated, their xNIC should be upgraded as well.

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

Make sure that you have the latest version of cloudSwXtch installed. You can find information about how to upgrade your cloudSwXtch by clicking here:

- [Azure](#)
- [AWS](#)

You can also upgrade your cloudSwXtch by deleting and recreating the instance.

Upgrading xNIC on Windows

1. **Open PowerShell.** If you are using Windows 11, please use Windows Terminal.
2. **Download** the installer script:

| | |
|---|------|
| None | Copy |
| <pre>Invoke-WebRequest -Uri 'http://<swxtch-instance-name>/services/install/swxtch-xnic-win-install.ps1' -Outfile swxtch-xnic-win-install.ps1</pre> | |

3. **Run** the script. Please use the appropriate command for your version.

| | |
|---|------|
| None | Copy |
| <pre>V1 ./swxtch-xnic-win-install.ps1 -k V2 ./swxtch-xnic-win-install.ps1 -k -v 2</pre> | |

The latest version of the Windows xNIC will be installed.

IMPORTANT

Remember to reboot the machine after the upgrade is complete. You must do this to execute the cloudSwXtch tools properly from your user home directory.

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:

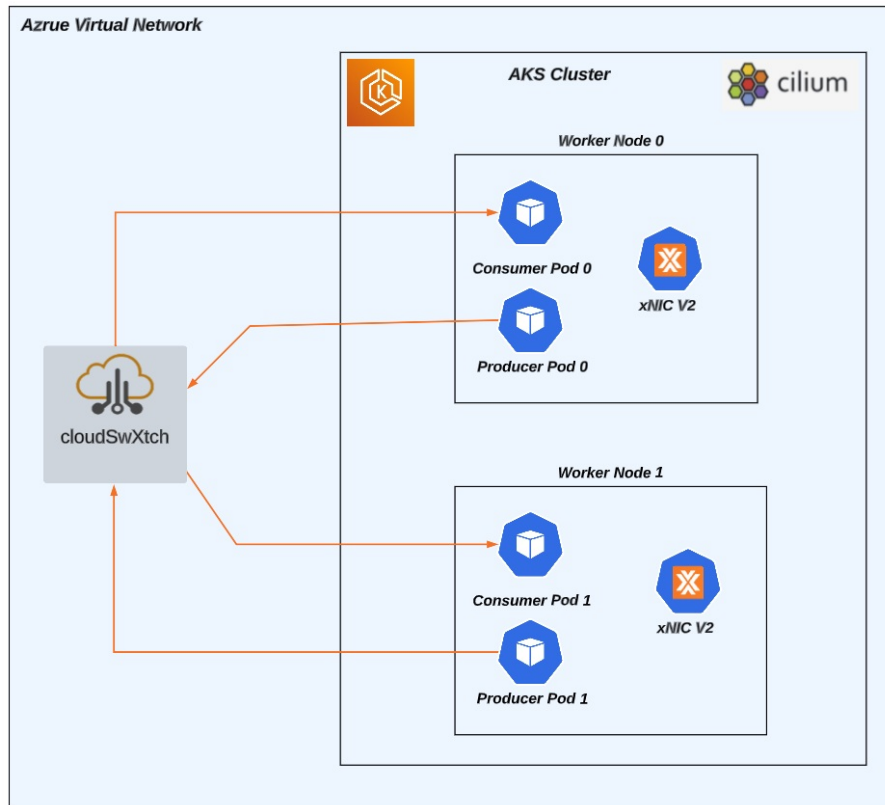
1. [Create an Azure Kubernetes Service with Cilium CNI](#)
2. [Install xNIC2 on AKS Cilium](#)
3. [Test xNIC2 with AKS](#)

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)

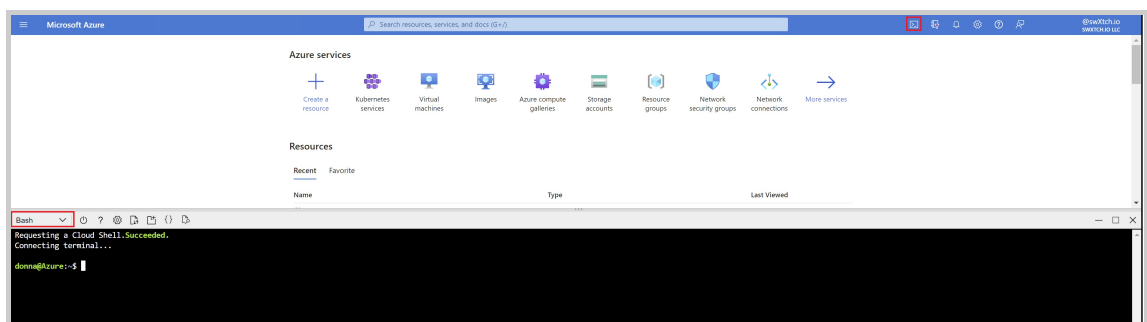
Each of these components **must match** the one used to create the cloudSwXtch on Azure.

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.

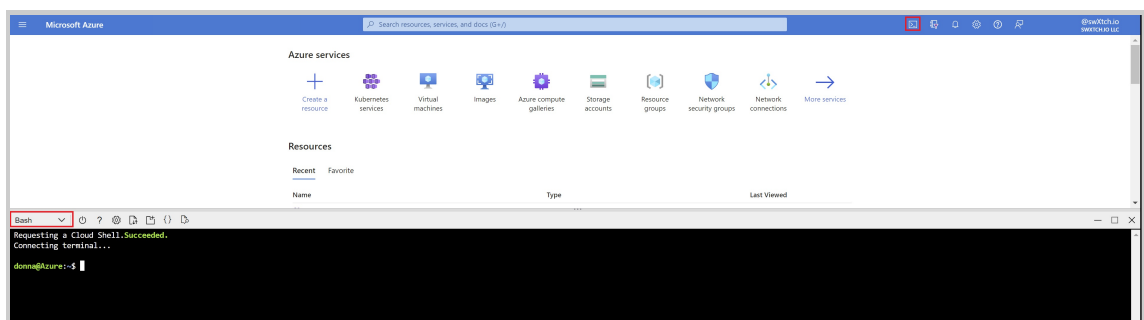
5. Edit the script with your desired values. Below is an example with filled in data:

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.

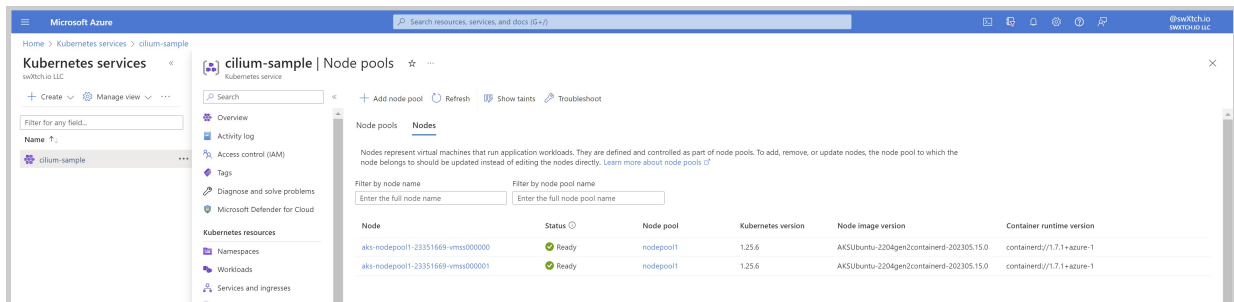


7. Paste in the script and hit Enter.

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 |
| | rg_cilium-sample-200 |
| | cilium-sample2 |
| | sample2 |
| | dsd-k8-cluster-100 |
| | cluster-100 |

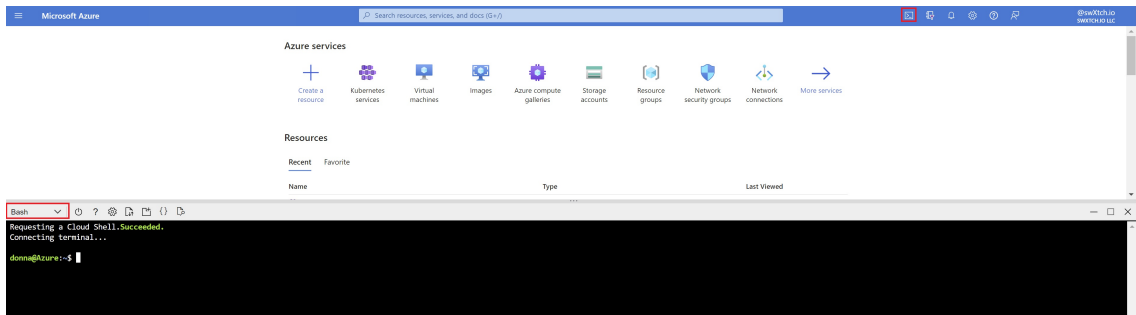
Cilium on AKS Installation

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:

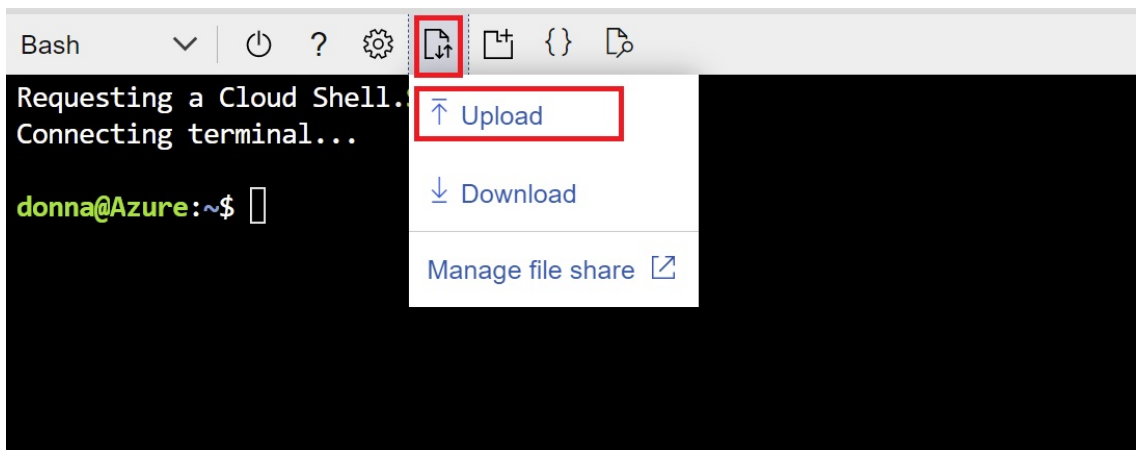
| | |
|--|------|
| Bash | Copy |
| <pre>CILIUM_CLI_VERSION=\$(curl -s https://raw.githubusercontent.com/cilium/cilium- cli/master/stable.txt) CLI_ARCH=amd64 if ["\$(uname -m)" = "aarch64"]; then CLI_ARCH=arm64; fi</pre> | |

```
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

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.

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 **dev.aks**. If it is not upgraded see [Upgrade cloudSwXtch on Azure](#).
2. Copy the content below into a Text Editor, name the file **xnic_cilium_install.sh** and Save.

Shell

| Bash | Copy |
|---|------|
| <pre>#!/usr/bin/env bash set -e # This is the main installer of xNICv2 for a Clustered environment # running Azure Kubernetes Service (AKS)and Cilium CNI. function show_usage { echo -e "\nUsage: \$0 [OPTIONS]" echo " -s, --switch swXtch IP address" echo " -h, --help shows this help" } function parse_arguments() { if [\$# -eq 0]; then echo -e "\nPlease specify the swXtch IP address." exit 1 else key="\$1" case \$key in -s --switch) SWXTCH_IP="\$2" shift # shifts argument ;; esac fi }</pre> | |

```

        shift # shifts value
    ;;
    -h| --help | *)
        show_usage
        exit 1
    ;;
esac
fi
}

# Parse script arguments
parse_arguments "$@"

if [ -n "$(kubectl get ds/foo -o NAME 2>/dev/null)" ]; then
    echo ""
    echo "=====
    echo "                Removing old installation"
    echo "=====
    kubectl delete ds/foo >/dev/null 2>&1
    echo "Done!"
fi

echo ""
echo "=====
echo "                Installing Multus CNI"
echo "=====

# Checking if Multus needs to be installed
if [[ -z $(kubectl get ds -n kube-system -l app=multus -o NAME | head -1) ]];
then
    rm -rf /tmp/multus-cni/
    cd /tmp/
    git clone https://github.com/k8snetworkplumbingwg/multus-cni.git && cd multus-
cni
    cat ./deployments/multus-daemonset.yml | kubectl apply -f -
fi

# Proceeding with installing the CRD - NetworkAttachmentDefinition resource type
cat << EOF | kubectl apply -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: eth0-bridge
  namespace: kube-system
spec:
  config: '{
    "cniVersion": "0.3.1",
    "type": "bridge",
    "bridge": "swxbr0",
    "ipMasq": true,
    "isGateway": true,
    "hairpinMode": true,
    "ipam": {

```

```

        "type": "host-local",
        "subnet": "10.10.0.0/16"
    }
}'
EOF

echo "Done!"

echo ""
echo "=====
echo "                Installing xNIC v2"
echo "=====
cat << EOF | kubectl apply -f -
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: swxtch-xnic
spec:
  selector:
    matchLabels:
      app: swxtch-xnic
  template:
    metadata:
      labels:
        app: swxtch-xnic
    spec:
      hostNetwork: true
      containers:
      - name: swxtch-xnic
        image: ubuntu:20.04
        imagePullPolicy: Always
        securityContext:
          privileged: true
        env:
        - name: SWX_K8S_BRIDGE_IF
          value: "swxbr0"
        - name: IS_DAEMON
          value: "true"
        command: ["/bin/bash"]
        args: ["-c", "apt update && apt install curl -y;
                    curl http://${SWXTCH_IP}/services/install/swxtch-xnic-
install.sh --output swxtch-xnic-install.sh;
                    chmod +x swxtch-xnic-install.sh;
                    ./swxtch-xnic-install.sh -v 2;
                    sleep infinity"]
EOF
echo "Done!"

echo ""
echo "=====
echo "                Restarting CNI Agents"
echo "=====
kubectl rollout restart -n kube-system ds/cilium-node-init ds/cilium

```

```

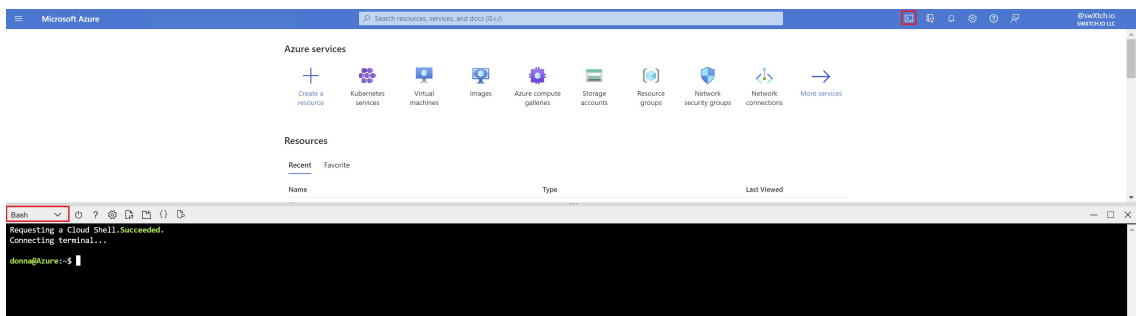
echo -n "Waiting for Cilium Agent to be fully UP..."
sleep 2
while [[ -z $(kubectl get po -n kube-system -l app.kubernetes.io/name=cilium-agent --field-selector status.phase=="Running" -o NAME | head -1) ]]
do
    echo -n "."
    sleep 2
done
echo "OK"
kubectl rollout restart -n kube-system ds/kube-multus-ds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

echo ""
echo "=====
echo "           Cleaning up and finishing installation"
echo "=====
echo "Removing temporary files"
rm -rf /tmp/multus-cni/
echo -e "\n===== Completed! ====="

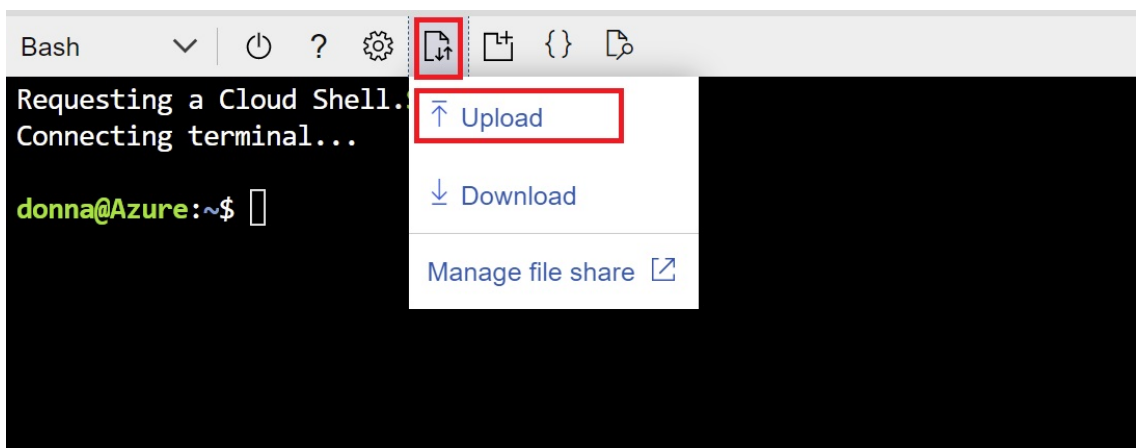
echo -e "\nPlease allow a minute for the xNIC DaemonSet to fully spin up before
starting to use it."
YELLOW='\033[1;33m'
echo -e "Feel free to follow up on the xNIC Agents installation by running
\n\n\t${YELLOW}kubectl logs daemonsets/swxtch-xnic -f"

```

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



5. Upload the xnic_cilium_install.sh file using the Bash Upload feature.



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

Shell

| Bash | Copy |
|--|------|
| <pre>chmod +x xnic_cilium_install.sh</pre> | |

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

Shell

| Bash | Copy |
|---|------|
| <pre>./xnic_cilium_install.sh -s <cloudSwXtch-instance-ip> -v 2</pre> | |

1. A successful install is shown below:

Shell

| Bash | Copy |
|---|------|
| <pre>donna@Azure:~\$./xnic_cilium_install.sh -s 10.2.1.234 ===== Installing Multus CNI ===== networkattachmentdefinition.k8s.cni.cncf.io/eth0-bridge unchanged daemonset.apps/foo created Done! ===== Installing xNIC v2 ===== daemonset.apps/swxtch-xnic unchanged Done! ===== Restarting CNI Agents ===== daemonset.apps/cilium-node-init restarted daemonset.apps/cilium restarted Waiting for Cilium Agent to be fully UP.....OK daemonset.apps/kube-multus-ds restarted Done! ===== Cleaning up and finishing installation ===== Removing temporary files ===== 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 **kubectlogs** in the Bash window in Azure:

Shell

| Bash | 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:

Shell

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

1. Example:

Shell

| Bash | Copy | | | | | |
|---|-------|---------|----------|--------|-----------|-----------------------------------|
| <pre>onna@Azure:~\$ kubectl get pods -o wide -w</pre> | | | | | | |
| NAME | READY | STATUS | RESTARTS | AGE | IP | NODE |
| NOMINATED NODE READINESS GATES | | | | | | |
| swxtch-xnic-vhjdf | 1/1 | Running | 0 | 3m42s | 10.5.1.12 | aks-nodpool11-20941474-vmss000000 |
| | | <none> | | <none> | | |
| swxtch-xnic-wxlnn | 1/1 | Running | 0 | 4m14s | 10.5.1.11 | aks-nodpool11-20941474-vmss000001 |
| | | <none> | | <none> | | |

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

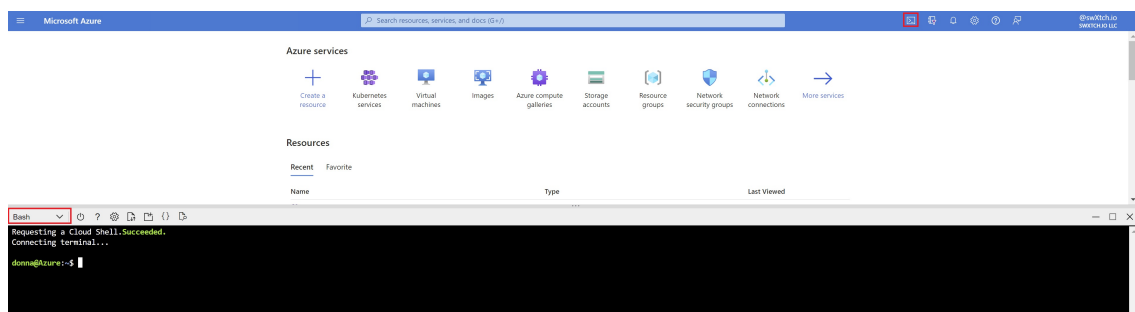
Shell

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

Shell

| Bash | Copy |
|--|------|
| donna@Azure:~\$ kubectl get pods -o wide | |

1. Example:

Shell

Bash

Copy

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

| NAME | READY | STATUS | RESTARTS | AGE | IP |
|-----------------------------------|-------|---------|-----------|------|-----------------|
| NODE | | | NOMINATED | NODE | READINESS GATES |
| consumer-a | 1/1 | Running | 0 | 127m | 10.0.0.147 |
| aks-nodepool1-40504797-vmss000004 | | | <none> | | <none> |
| producer-a | 1/1 | Running | 0 | 127m | 10.0.1.136 |
| aks-nodepool1-40504797-vmss000005 | | | <none> | | <none> |
| swxtch-xnic-26vmp | 1/1 | Running | 0 | 29m | 10.2.128.95 |
| aks-nodepool1-40504797-vmss000004 | | | <none> | | <none> |
| swxtch-xnic-x9s7r | 1/1 | Running | 0 | 28m | 10.2.128.96 |
| aks-nodepool1-40504797-vmss000005 | | | <none> | | <none> |

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

Shell

| Bash | Copy |
|--|------|
| kubectl exec -it pod/swxtch-pod-name -- bash | |

1. Example:

Shell

| Bash | Copy |
|---|------|
| kubectl exec -it pod/swxtch-xnic-26vmp -- bash root@aks-nodepool1-40504797-vmss000004:/# | |

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 more information about swxtch-tcpdump, see swxtch-tcpdump under Testing cloudSwXtch.

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:

| Bash | Copy |
|------|------|
|------|------|

```
cat /var/opt/swxtch/swxtch-xnic.conf
```

Exiting the Pod

To exit the pod, enter in the following command:

```
Bash
```

[Copy](#)

```
Exit
```

Other Common Kubernetes Commands

Here are a few other useful command examples with returns below for validation purposes.

To Get Kubernetes DaemonSets

```
Bash
```

[Copy](#)

```
kubectl get ds
```

An example of a return:

```
Bash
```

[Copy](#)

```
donna@Azure:~$ kubectl get ds
NAME          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
swxtch-xnic    2         2         2       2            2           <none>         17h
```

To Get Kubernetes Pods

```
Bash
```

[Copy](#)

```
kubectl get pods -o wide
```

An example of a return:

```
Bash
```

[Copy](#)

```
donna@Azure:~$ kubectl 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:~$
```


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> | |

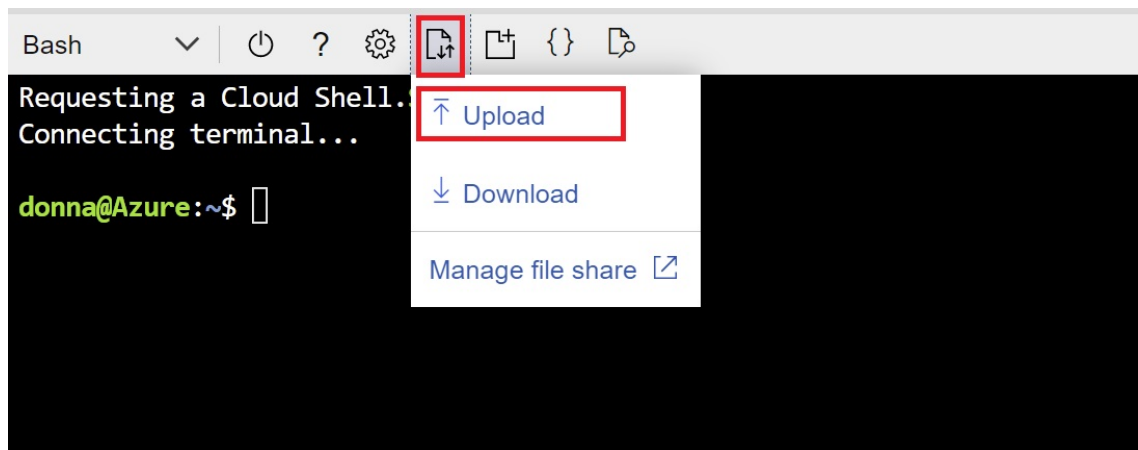
```

    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> | |

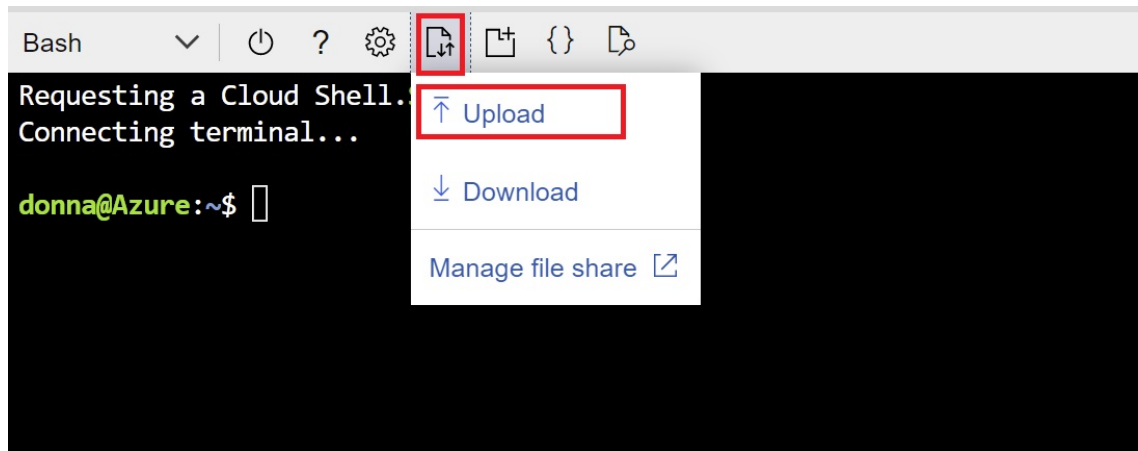

```

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"]
    - 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 |
|-------------------------------------|------|
| <pre>kubectl get pods -o wide</pre> | |

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

Shell

| Bash | Copy | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|-------|---------|----------|----------|-------------|-----------------------------------|------|--------------------------------|--|--|--|--|--|--|------------|-----|---------|---|-----|------------|-----------------------------------|--|--|--|--|--|--------|--------|------------|-----|---------|---|-----|------------|-----------------------------------|--|--|--|--|--|--------|--------|-------------------|-----|---------|---|-----|-------------|-----------------------------------|--|--|--|--|--|--------|--------|-------------------|-----|---------|---|-----|-------------|-----------------------------------|--|--|--|--|--|--------|--------|--|
| <pre>donna@Azure:~\$ kubectl get pods -o wide</pre> <table><tr><th>NAME</th><th>READY</th><th>STATUS</th><th>RESTARTS</th><th>AGE</th><th>IP</th><th>NODE</th></tr><tr><td colspan="7">NOMINATED NODE READINESS GATES</td></tr><tr><td>consumer-a</td><td>1/1</td><td>Running</td><td>0</td><td>17h</td><td>10.0.0.196</td><td>aks-nodepool1-23351669-vmss000003</td></tr><tr><td colspan="4"></td><td></td><td><none></td><td><none></td></tr><tr><td>producer-a</td><td>1/1</td><td>Running</td><td>0</td><td>17h</td><td>10.0.1.153</td><td>aks-nodepool1-23351669-vmss000002</td></tr><tr><td colspan="4"></td><td></td><td><none></td><td><none></td></tr><tr><td>swxtch-xnic-dwx2d</td><td>1/1</td><td>Running</td><td>0</td><td>17h</td><td>10.2.128.96</td><td>aks-nodepool1-23351669-vmss000002</td></tr><tr><td colspan="4"></td><td></td><td><none></td><td><none></td></tr><tr><td>swxtch-xnic-dzpf1</td><td>1/1</td><td>Running</td><td>0</td><td>17h</td><td>10.2.128.95</td><td>aks-nodepool1-23351669-vmss000003</td></tr><tr><td colspan="4"></td><td></td><td><none></td><td><none></td></tr></table> <pre>donna@Azure:~\$</pre> | 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> | |
| 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> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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

Shell

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

4. Log into your cloudSwXtch and run this command:

Shell

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

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

| | | | | | |
|-----------------------------------|--------------------------------------|--|--------------------------|---------------------------|-----------|
| dsd-core-100 | | dev.3645.v2(dsd-100-core-azure-April-5-2033) | | Information - dev.3645.v2 | |
| SubscriptionId | b10208ad-ad22-4c26-8aef-be93b2f0bb58 | Auth. Type | License File Marketplace | 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 |
| aks-nodepool1-23351669-vmss000004 | 10.2.128.95 | dev.3645.v2 (v2) | 99 | 137.6K | 0 |
| aks-nodepool1-23351669-vmss000005 | 10.2.128.96 | dev.3645.v2 (v2) | 0 | 0 | 99 |
| | | | | | TX bps |
| | | | | | 136.7K |
| | | | | | HA |
| | | | | | N |
| | | | | | 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 | | | | |
| swtch-xnic-46qgg | 1/1 | Running | 0 | 42m | 10.2.128.96 | aks-nodepool1-23351669-vmss000005 |
| | | <none> | | | <none> | |
| swtch-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

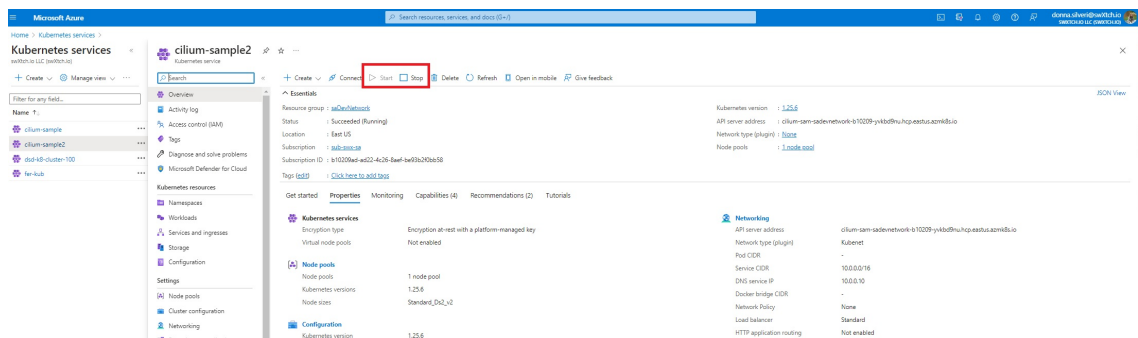
The nodes upgrade is automatic based on the restart of the nodes containing the xNIC. This can be done in one of two ways: [stopping the AKS cluster in Azure](#) and [restarting the Node](#).

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](#).

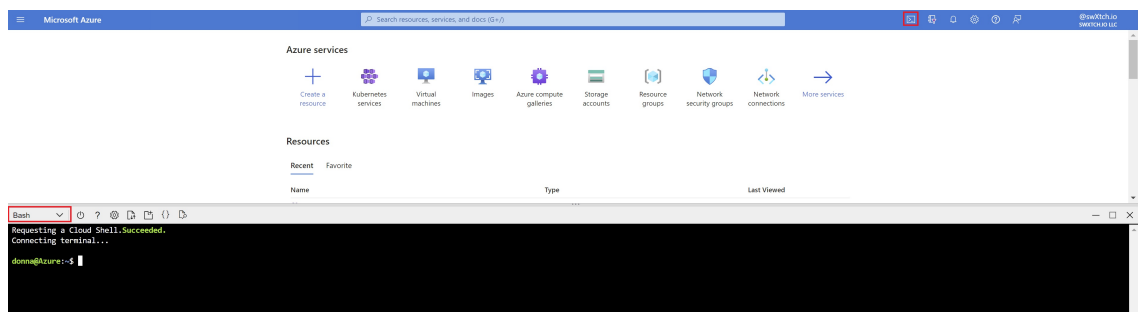
Stopping the AKS Cluster in Azure

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.



Restarting the Nodes

1. Sign into Azure portal
2. Open cloudShell as Bash.



3. Put in this command:
Shell

Bash

Copy

```
kubect1 get pods -o wide -w
```

4. Wait for the nodes within to start.

To ensure the nodes in the AKS are started, use the command below to follow the AKS node status:

| | |
|-----------------------------|------|
| Bash | Copy |
| kubect1 get pods -o wide -w | |

Below is a sample output with the above commands:

| | | | | | | | |
|---|-----------------|-------------------|----------|--------|-------------|--------|------|
| Bash | | | | | | | Copy |
| <pre>onna@Azure:~\$ kubectl rollout restart ds/swxtch-xnic daemonset.apps/swxtch-xnic restarted donna@Azure:~\$ kubectl get pods -o wide -w</pre> | | | | | | | |
| NAME | READY | STATUS | RESTARTS | AGE | IP | NODE | |
| NOMINATED NODE | READINESS GATES | | | | | | |
| consumer-a | 1/1 | Running | 0 | 97m | 10.0.0.147 | aks- | |
| nodepool1-40504797-vmss000004 | | <none> | | <none> | | | |
| producer-a | 1/1 | Running | 0 | 97m | 10.0.1.136 | aks- | |
| nodepool1-40504797-vmss000005 | | <none> | | <none> | | | |
| swxtch-xnic-2zx8r | 1/1 | Running | 0 | 100m | 10.2.128.96 | aks- | |
| nodepool1-40504797-vmss000005 | | <none> | | <none> | | | |
| swxtch-xnic-5bfgz | 1/1 | Terminating | 0 | 100m | 10.2.128.95 | aks- | |
| nodepool1-40504797-vmss000004 | | <none> | | <none> | | | |
| swxtch-xnic-5bfgz | 0/1 | Terminating | 0 | 100m | 10.2.128.95 | aks- | |
| nodepool1-40504797-vmss000004 | | <none> | | <none> | | | |
| swxtch-xnic-5bfgz | 0/1 | Terminating | 0 | 100m | 10.2.128.95 | aks- | |
| nodepool1-40504797-vmss000004 | | <none> | | <none> | | | |
| swxtch-xnic-5bfgz | 0/1 | Terminating | 0 | 100m | 10.2.128.95 | aks- | |
| nodepool1-40504797-vmss000004 | | <none> | | <none> | | | |
| swxtch-xnic-26vmp | 0/1 | Pending | 0 | 0s | <none> | <none> | |
| <none> | <none> | | | | | | |
| swxtch-xnic-26vmp | 0/1 | Pending | 0 | 0s | <none> | aks- | |
| nodepool1-40504797-vmss000004 | | <none> | | <none> | | | |
| swxtch-xnic-26vmp | 0/1 | ContainerCreating | 0 | 0s | 10.2.128.95 | aks- | |
| nodepool1-40504797-vmss000004 | | <none> | | <none> | | | |
| swxtch-xnic-26vmp | 1/1 | Running | 0 | 2s | 10.2.128.95 | aks- | |
| nodepool1-40504797-vmss000004 | | <none> | | <none> | | | |
| swxtch-xnic-2zx8r | 1/1 | Terminating | 0 | 100m | 10.2.128.96 | aks- | |
| nodepool1-40504797-vmss000005 | | <none> | | <none> | | | |
| swxtch-xnic-2zx8r | 0/1 | Terminating | 0 | 101m | 10.2.128.96 | aks- | |
| nodepool1-40504797-vmss000005 | | <none> | | <none> | | | |
| swxtch-xnic-2zx8r | 0/1 | Terminating | 0 | 101m | 10.2.128.96 | aks- | |
| nodepool1-40504797-vmss000005 | | <none> | | <none> | | | |
| swxtch-xnic-2zx8r | 0/1 | Terminating | 0 | 101m | 10.2.128.96 | aks- | |
| nodepool1-40504797-vmss000005 | | <none> | | <none> | | | |
| swxtch-xnic-x9s7r | 0/1 | Pending | 0 | 0s | <none> | <none> | |
| <none> | <none> | | | | | | |
| swxtch-xnic-x9s7r | 0/1 | Pending | 0 | 0s | <none> | aks- | |
| nodepool1-40504797-vmss000005 | | <none> | | <none> | | | |
| swxtch-xnic-x9s7r | 0/1 | ContainerCreating | 0 | 0s | 10.2.128.96 | aks- | |
| nodepool1-40504797-vmss000005 | | <none> | | <none> | | | |
| swxtch-xnic-x9s7r | 1/1 | Running | 0 | 1s | 10.2.128.96 | aks- | |
| nodepool1-40504797-vmss000005 | | <none> | | <none> | | | |

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.

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

Configuring The Bridge for Static Cloud to Ground

The 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.

The location of the configuration file is `/var/opt/swxtch/swxtch-bridge.json`.

Modify the JSON array attribute for **"cloudToGroundSubscriptions"** and add the multicast groups that need to go from the cloud to the ground.

```
{
  "bridgeConfig": {
    "ctrlInterfaceName": "eth0",
    "dataInterfaceName": "eth1",
    "userInterfaceName": "eth0",
    "swxtchCtrlIp": "10.0.0.1",
    "swxtchCtrlPort": 80,
    "swxtchDataIp": "10.0.1.1",
    "swxtchDataPort": 9999,
    "pathId": 0,
    "overwriteSenderIp": "172.30.1.1",
    "groundToCloudSubscriptions": [],
    "cloudToGroundSubscriptions": [
      "225.0.23.182:12000",
      "225.0.23.183:12000",
      "225.0.23.184:12000",
      "225.0.23.185:12000"],
    "pollingIntervalMilliseconds": 1000
  }
}
```

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

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

These multicast groups will now be sent from the cloud to the ground.

Protocol Conversion and Fanout

WHAT TO EXPECT

There are two functionalities to configure Protocol Conversion and Fanout:

- Enabling a cloudSwXtch to ingest unicast data
- Creating a unicast consumer

In this article, you will learn about both functionalities and the API calls associated with each.

Enabling a cloudSwXtch to Ingesting Unicast

To configure the cloudSwXtch to ingest a unicast stream, you must select a multicast base address and open a range of ports. The cloudSwXtch's protocol fanout feature will map the range of ports to its respective multicast group.

NOTE

Port Range: There is no required range, avoid ports used by the swxtch. Refer to [cloudSwXtch System Requirements](#) for prerequisites.

MC Base Address: If there is more than one unicast input within the range, each new port will map to its respective multicast group and will increment from this base address.

ENABLE

To enable the port range on the cloudSwXtch to map the unicast to multicast, execute this command in any terminal that has access to the cloudSwxtch:

Bash

Copy

```
curl http://<cloudSwXtch-IP>/swxtch/unicast-adaptor/enable -d '{"baseAddr": "XXX.X.XX.X", "portRange": [XXXX,XXXX]}'
```

Below you will find an example of what a successful call will look like.

Bash

Copy

```
curl http://NGd-core/swxtch/unicast-adaptor/enable -d '{"baseAddr": "239.5.69.2", "portRange": [2000,2005]}'
```

Once you execute this API request, the unicast adaptor will be enabled. If a user would want to subscribe to the new multicast feed above, they would enter the following:

Bash

Copy

```
MC Address: 239.5.69.2:2000
```

If we have a new unicast input, the multicast address and its port will change. Every new unicast input needs to be mapped from the port and the base multicast address, increasing incrementally by 1 as shown above for the initial multicast IP address and below as a subsequent one.

| | |
|-------------------------|------|
| None | Copy |
| NIC Address: 239.4.0.10 | |

SHOW-CONFIG

You can use the **show-config** command to display information regarding your protocol fanout configuration. To do so, execute the following command:

| | |
|---|------|
| Bash | Copy |
| <pre>curl http://localhost/swxtch/unicast-adaptor/show-config</pre> | |

Below is an example output with the status, multicast base address and port range displayed:

| | |
|--|------|
| Bash | Copy |
| <pre>{ "status": "Enable", "baseAddr": "239.4.0.10", "portRange": [10000, 12000] }</pre> | |

DISABLE

To **disable** the port range on the cloudSwXtch to map the unicast to multicast, execute this command in any terminal that has access to the cloudSwxtch:

| | |
|--|------|
| Bash | Copy |
| <pre>curl http://<cloudSwXtch-IP>/swxtch/unicast-adaptor/disable</pre> | |

Creating a Unicast Consumer

In order to properly configure Protocol Fanout, users will need to manually subscribe their xNIC-less machine(s) to the multicast group(s) that they should be receiving traffic from. There are three additional endpoints exposed for this step: one is for **adding** a machine to a multicast group, another for **removing**, and a final one for **providing a list** of all the non-xNIC agents that were added as well which groups they're subscribed to.

ADD

To **add a machine to a multicast group**, execute this command in any terminal that has access to the cloudSwxtch:

| | |
|--|------|
| Bash | Copy |
| <pre>curl http://<cloudSwXtch-ip>/swxtch/unicast-adaptor/join -d '{"targetIp": "XX.X.XXX.XX", "targetMac": "XX:XX:XX:XX:XX:XX", "groupIp": "XX.X.XXX.XX"}'</pre> | |

The cloudSwxtch will ensure that all packets destined to the specified address will receive a unicast packet irrespective of the source -- i.e. whether it came from a xNIC or non-xNIC producer.

REMOVE

To remove a machine from a multicast group, execute this command in any terminal that has access to the cloudSwxtch:

| Bash | Copy |
|---|------|
| <pre>curl http://<cloudswxtch-ip>/swxtch/unicast-adaptor/leave -d '{"targetIp": "XX.X.XXX.XX", "targetMac": "XX:XX:XX:XX:XX:XX", "groupIp": "XX.X.XXX.XX"}'</pre> | |

LIST

Finally, to list all of the non-xNIC machines and their respective group membership(s), execute this command in any terminal that has access to the cloudSwxtch:

| Bash | Copy |
|---|------|
| <pre>curl http://<cloudSwxtch-IP>/swxtch/unicast-adaptor/list</pre> | |

HTTP Request Arguments

- **targetIp** -- This represents the IP address of the machine that the cloudSwxtch will forward the traffic to.
 - **targetMac** -- The MAC address of the machine that the cloudSwxtch will forward the traffic to.
 - **groupIp** -- The IP address of the multicast group the non-xNIC consumer will join.
-

Licensing Information

Azure Market Place Licensing - Things to Know

Below is an illustration of our current cloudSwXtch type offerings which automatically generate licenses associated to them via the Azure Marketplace install. There is no manual effort required from the customer in terms of configuring licenses, but there are a few things that would be good to know which are touched on below.

| cloudSwXtch Choose Your Plan | | | | |
|----------------------------------|--------------|--------|---------|---------|
| Features | 30 Day Trial | Small | Medium | Large |
| Multicast | ✓ | ✓ | ✓ | ✓ |
| Protocol Fanout | ✓ | | ✓ | ✓ |
| Ground-to-Cloud | ✓ | | ✓ | ✓ |
| Cloud-to-Ground (In development) | | | ✓ | ✓ |
| Mesh | ✓ | | ✓ | ✓ |
| Stream Redundancy ST 2022-7 | ✓ | | ✓ | ✓ |
| Bandwidth Capacity (egress) | 96 Gb/s | 2 Gb/s | 30 Gb/s | 96 Gb/s |
| Endpoint Connections | Unlimited | 10 | 50 | 200 |
| | Free | \$2/hr | \$7/hr | \$15/hr |

How to Upgrade your cloudSwXtch

Currently with a Marketplace installation, a cloudSwXtch upgrade requires downtime as the running Swtch has to be shut down and recreated with the desired plan type. The licensing is provisioned behind the scenes and requires no action on the user end.

To avoid downtime when upgrading a cloud Swtch, it is recommended to have a secondary cloudSwXtch readily available to be switched over to when decomission a running cloudSwXtch for an upgrade.

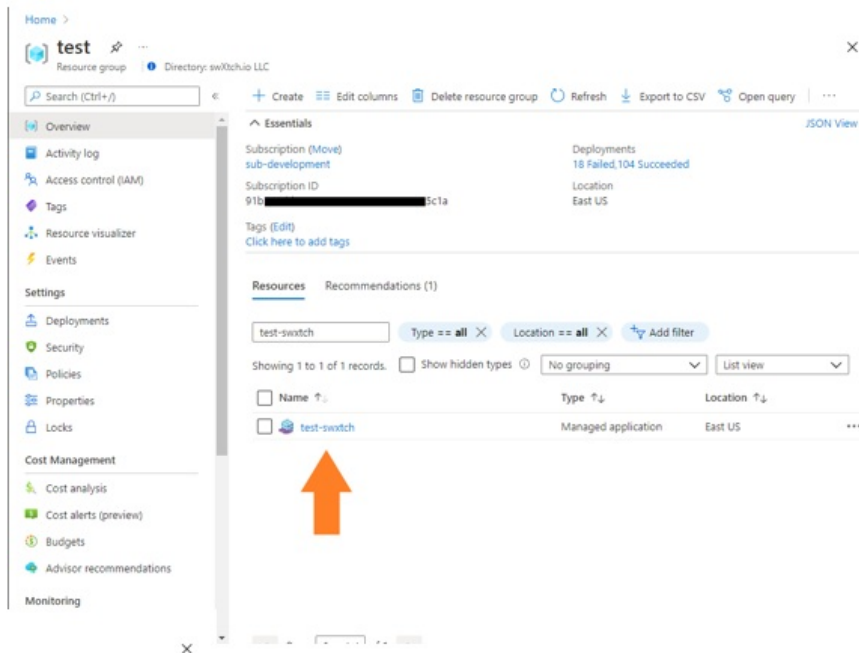
Exceding Endpoint Connections Limit

Be mindful of the number of endpoint connections you have setup with your Swtch. Based on the plan type being utilized, a small for example, you will recieve an error If you exceed the 10 endpoint limit.

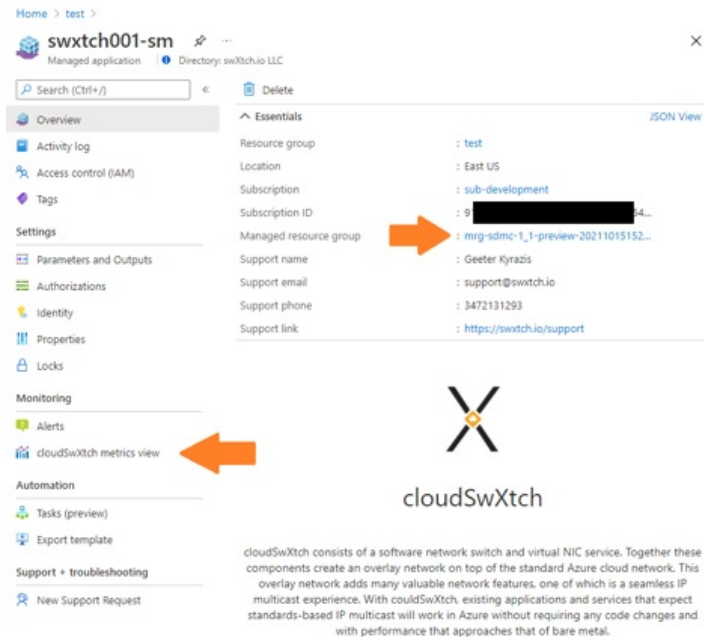
Azure Monitoring

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.

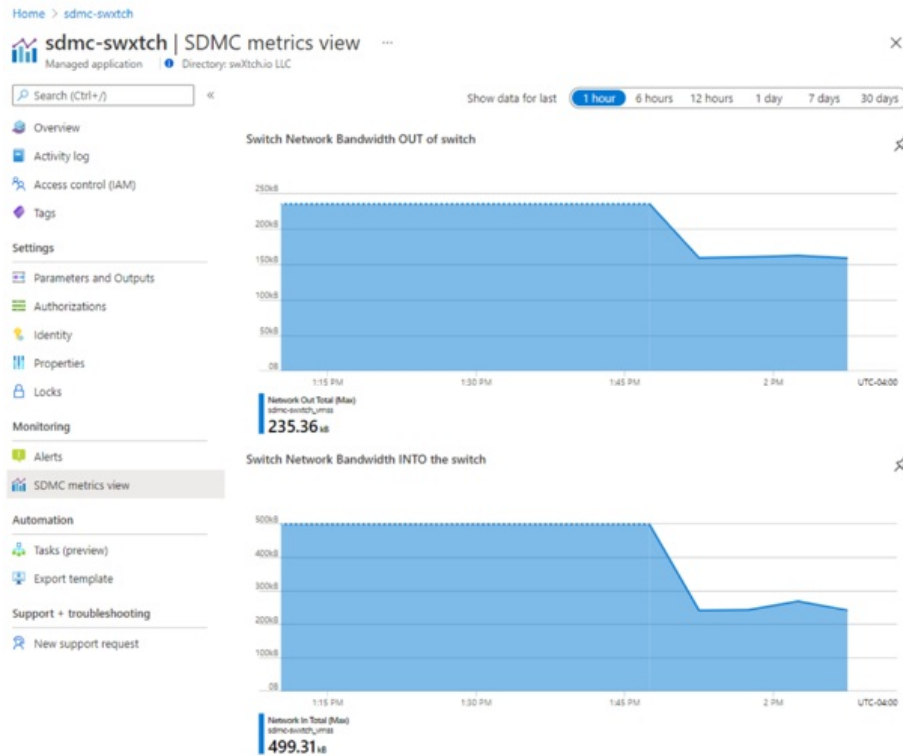


cloudSwXtch metrics view

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 |

Testing cloudSwXtch

Testing

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

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

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

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

swxtch-perf

Overview

To simulate traffic movement throughout the cloudswtch overlay network you can use swxtch-perf to create producer and consumers on machines with the xNIC installed.

swxtch-perf producer has multiple parameters that can be configured to generate different traffic flows. There can be multiple instances of swxtch-perf generating traffic on a single machine.

| | |
|---|------|
| None | Copy |
| <pre>swxtch-perf producer --sendto <MC_ADDRESS:DEST_PORT> --nic <NETWORK_INTERFACE></pre> | |

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

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

NOTE

<MC_ADDRESS> = Multicast Address
<DEST_PORT> = Destination Port
<NETWORK_INTERFACE> = Network Interface where xNIC conncted to. The network interface does not have to be specified in xNic V1, but must be specified in xNic V2. (See [xNIC Linux Installation](#) for V1 and V2 differences.

swxtch-perf

For a quick view at the functionality and usage of **swxtch-perf** use `-h` or `-help` .

| | |
|---|------|
| None | Copy |
| <pre>swxtch-perf -h Usage: swxtch-perf [options] command Positional arguments: command [producer consumer] suported commands Optional arguments: -h --help shows help message and exits [default: false] -v --version prints version information and exits [default: false] --nic name of NIC to use this is Mandatory for swxtch-perf to work. --recvfrom IP:Port The IP and Port where packets come from [default: "239.5.69.2:10000"] --sendto IP:Port The IP and Port where packets are sent to [default: "239.5.69.2:10000"] --payload_length (producer command only) number of bytes for the multicast udp payload [default: 100] --total_pkts Total packets to send/receive. To run without this limit use 0 [default: 0]</pre> | |

```

--pps                (producer command only) packet-rate or packet per seconds
[default: 1]
--seconds            Number of seconds to run the application. To run without this
limit use 0 [default: 0]
--loopback           Receives packets from --recvfrom and sends packets to --sendto
[default: false]
--generic            (consumer command only) to consume generic packets [default:
false]
--latency            Enables timestamp propagation and measurement of latency
[default: false]
--broadcast          Enables broadcast packets in NIC, this overrides IP argument
[default: false]
--generic-broadcast  Sends broadcast packets to 255.255.255.255, valid only with --
broadcast argument [default: false]
--broadcast-port     Port for broadcast traffic, valid only with --broadcast
argument [default: 10000]

```

Parameters

| Argument | Description | Default Value | Valid Range | Machine Type | Operating System |
|-----------------------------|--|---------------|--|--------------|------------------|
| <code>h</code> | Shows commands that are available. | | | | All |
| <code>v</code> | Shows version. | | | Both | All |
| <code>nic</code> | Specify which network interface xNIC will listen to this command is Mandatory. | | -- | Both | All |
| <code>recvfrom</code> | Specify the multicast group and port to listen for packets IPv4 addresses are valid; Ports: 1024 <= x <= 65535. Mandatory for Consumer Mode and Multicast. | | | Consumer | All |
| <code>sendto</code> | Specify the multicast group and port to send packets, mandatory for producer if using multicast. | All | IPv4 addresses are valid; Ports: 1024 <= x <= 65535 Mandatory for Producer Mode and Multicast. | Producer | All |
| <code>payload_length</code> | Number of bytes per packet. | 100 | 8 and 3750 | Producer | All |
| <code>total_pkts</code> | Number packets to receive or send before exiting iperf. | 0 | 8 and 3750 | Producer | Windows |
| <code>pps</code> | packet-rate or packets per second. | 1 | 100000 | Producer | All |
| <code>seconds</code> | Number of seconds to run the application, use 0 to run without a limit. | 0 | | Both | Windows |
| <code>loopback</code> | Receives packets from recvfrom and sends packets to sendto. | false | true:false | Both | Linux |
| <code>generic</code> | Consume generic packets. | false | true:false | Consumer | All |
| <code>latency</code> | Enables timestamp propagation and measurement of latency. | false | true:false | Both | Linux |

| | | | | | |
|--------------------------------|--|-------|---------------------------|------|-----|
| <code>broadcast</code> | Sets swtch-perf to use normal broadcast mode, when sending it will use the IP of the --nic argument. | false | true:false | Both | All |
| <code>generic-broadcast</code> | Sets iperf to use broadcast mode using the IP of 255.255.255.255. | false | true:false | Both | All |
| <code>broadcast-port</code> | Sets port to be used for broadcast, and is only valid with --broadcast and --generic-broadcast argument and is Mandatory for --broadcast --generic-broadcast . | | Ports: 1024 <= x <= 65535 | Both | All |

Multicast - Example

These examples can be run from one machine or across multiple machines. Parameters for NIC names assume default installation options.

EXAMPLE

Single Producer, Single Consumer, and one multicast group

Run this command on a VM to create a multicast group on the address `230.1.1.1` and port `3490` :

| | |
|---|------|
| None | Copy |
| Linux: swtch-perf producer --sendto 239.1.1.1:3490 --pps 1000 --nic swtch-tun0 Windows: swtch-perf producer --sendto 239.1.1.1:3490 --pps 1000 --nic swtch-tun | |

Example with results:

| | |
|--|------|
| None | Copy |
| swtch-perf producer --sendto 239.1.1.1:3490 --pps 1000 --nic swtch-tun0 Trying to reach a packet-rate of 1000 pps swtch-perf producer threads started... Ctrl+C to exit. <pre> ----- ----- TOTALS THIS PERIOD TX PKTS TX BYTES TX DROPS TX-PPS TX-bps TX-DPS ----- ----- ----- ----- ----- ----- 1,283 128KB 0 1.28K 1.0Mbps 0 2,274 227KB 0 991 792Kbps 0 3,267 326KB 0 993 794Kbps 0 4,262 426KB 0 995 796Kbps 0 </pre> | |

Run this command on one of the VMs to listen to traffic on the Multicast Address `230.1.1.1` port `13490` :

| | |
|--|------|
| None | Copy |
| Linux: swtch-perf consumer --recvfrom 239.1.1.1:3490 --nic swtch-tun0 Windows: | |

```
swtch-perf consumer --recvfrom 239.1.1.1:3490 --nic swtch-tun
```

Example with results:

None

Copy

testadmin@DSd-agent-102:~\$ swtch-perf consumer --recvfrom 239.1.1.1:3490 --nic swtch-tun0

swtch-perf consumer threads started... Ctrl+C to exit.

| TOTALS | | | THIS PERIOD | | |
|---------|----------|----------|-------------|---------|--------|
| RX PKTS | RX BYTES | RX DROPS | RX-PPS | RX-bps | RX-DPS |
| 0 | 0B | 0 | 0 | 0bps | 0 |
| 0 | 0B | 0 | 0 | 0bps | 0 |
| 0 | 0B | 0 | 0 | 0bps | 0 |
| 330 | 33.00KB | 0 | 330 | 264Kbps | 0 |
| 1,326 | 132KB | 0 | 996 | 796Kbps | 0 |
| 2,328 | 232KB | 0 | 1.00K | 801Kbps | 0 |
| 3,330 | 333KB | 0 | 1.00K | 801Kbps | 0 |
| 4,332 | 433KB | 0 | 1.00K | 801Kbps | 0 |
| 5,328 | 532KB | 0 | 996 | 796Kbps | 0 |
| 6,330 | 633KB | 0 | 1.00K | 801Kbps | 0 |

- To add more consumers you simply run the same swtch-perf command on new VMs.

Broadcast - Example

These examples can be run from one machine or across multiple machines. Parameters for NIC names assume default installation options.

EXAMPLE

Single Producer, Single Consumer, and broadcast

Run this command on a VM to create a broadcast

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

Example with results:

| None | Copy |
|--|------|
| <pre>PS C:\Users\testadmin> swtch-perf producer --broadcast --nic 'Ethernet 2' --pps 1000 --broadcast-port 1234 Config: Sending traffic to broadcast address.</pre> | |

```
Ip Address: 10.2.195.255
Port      : 10000
Interface IP Address: 45
Running without a total packet counter limit
Running the application without a timing limit
Sent 972 total packets, throughput: 890.383 pkts/sec
Sent 2047 total packets, throughput: 993.128 pkts/sec
Sent 3123 total packets, throughput: 991.82 pkts/sec
Sent 4198 total packets, throughput: 990.419 pkts/sec
```

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

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

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.

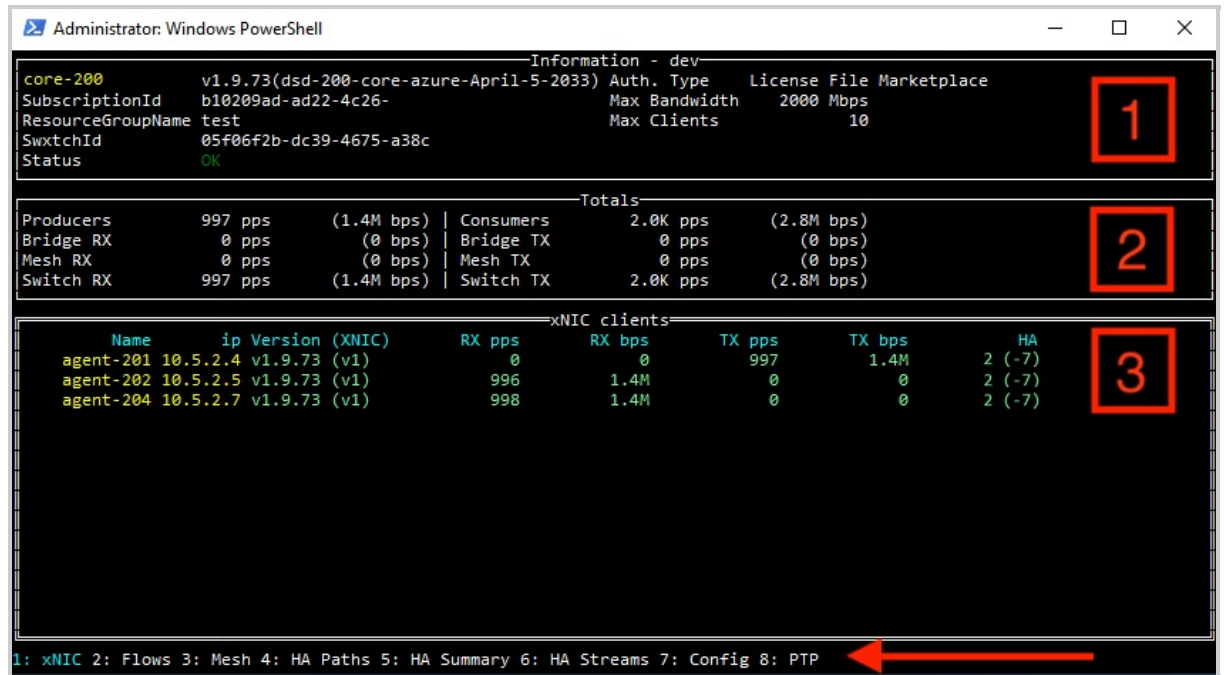
For both Windows and Linux agents, users can enter the following into the terminal:

| | |
|---------------------|------|
| 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



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. Flows
3. Mesh
4. HA Paths
5. HA Summary
6. HA Streams
7. Config
8. PTP

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)**.

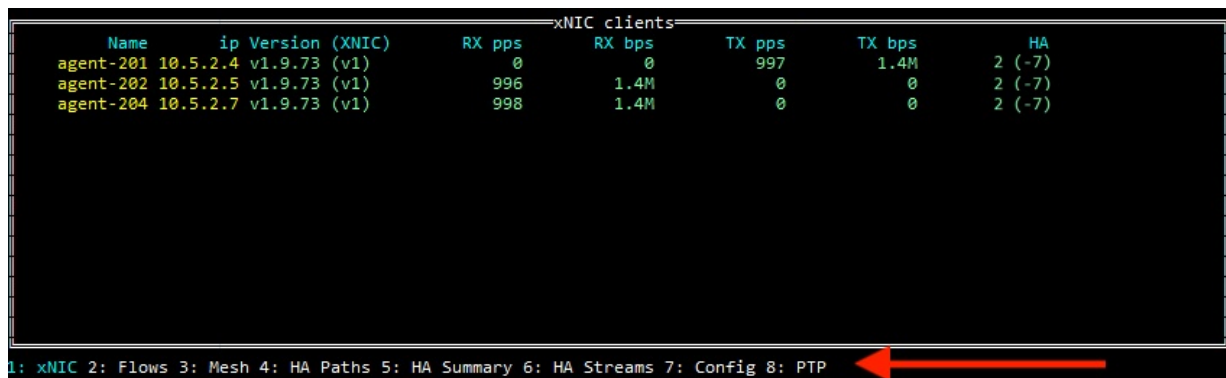
Please note: If your cloudSwXtch is part of a mesh or a bridge, the ingress/egress will show data in those sections.

- **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



| Name | ip | Version (XNIC) | RX pps | RX bps | TX pps | TX bps | HA |
|-----------|----------|----------------|--------|--------|--------|--------|--------|
| agent-201 | 10.5.2.4 | v1.9.73 (v1) | 0 | 0 | 997 | 1.4M | 2 (-7) |
| agent-202 | 10.5.2.5 | v1.9.73 (v1) | 996 | 1.4M | 0 | 0 | 2 (-7) |
| agent-204 | 10.5.2.7 | v1.9.73 (v1) | 998 | 1.4M | 0 | 0 | 2 (-7) |

1: xNIC 2: Flows 3: Mesh 4: HA Paths 5: HA Summary 6: HA Streams 7: Config 8: PTP

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

| Flow Stats | | | | | | | | |
|----------------|------------------|----------|------------------|--------|--------|--------|--------|--------------|
| Name | Src IP:Port | Protocol | MC Group IP:Port | RX pps | RX bps | TX pps | TX bps | Destinations |
| 239.1.1.1:3490 | 172.30.0.4:32712 | MC | 239.1.1.1:3490 | 1000 | 1.4M | 2.0K | 2.8M | 2 |

1: xNIC 2: Flows 3: Mesh 4: HA Paths 5: HA Summary 6: HA Streams 7: Config 8: PTP

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 | |
|---------------|-------------|
| SwitchAddress | Gateway |
| 10.2.128.10 | 10.2.128.10 |

1: xNIC 2: Flows 3: Mesh 4: HA Paths 5: HA Summary 6: HA Streams 7: Config 8: PTP

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 swtch-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 |
| Path 1 | core-100 |
| Path 2 | core-200 |

1: xNIC 2: Flows 3: Mesh 4: HA Paths 5: HA Summary 6: HA Streams 7: Config 8: PTP

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 | |
| agent-201 | Path 1 | 0 | 0 | 0.00 | 0 | |
| | Path 2 | 0 | 0 | 0.00 | 1 | |
| | Reconstructed Path | 0 | 0 | | 0 | |
| agent-202 | Path 1 | 994 | 1.0M | 100.00 | 0 | |
| | Path 2 | 0 | 0 | 0.00 | 192479 | |
| | Reconstructed Path | 994 | 1.0M | | 0 | |
| agent-204 | Path 1 | 1.0K | 1.1M | 100.00 | 0 | |
| | Path 2 | 0 | 0 | 0.00 | 178368 | |
| | Reconstructed Path | 1.0K | 1.1M | | 0 | |

1: xNIC 2: Flows 3: Mesh 4: HA Paths 5: HA Summary 6: HA Streams 7: Config 8: PTP

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 Streams

| HA Streams | | | | | | | | | |
|------------|---------------|-----------|--------------------|------------------|------------------|--------------|-----------------|--|--|
| Agent | Stream Src IP | Stream IP | Paths | Path Ingress pps | Path Ingress bps | Path Usage % | Missing Packets | | |
| agent-202 | 172.30.0.4 | 239.1.1.1 | Path 1 | 999 | 1.1M | 100.00 | 0 | | |
| | | | Path 2 | 0 | 0 | 0.00 | 1426210 | | |
| | | | Reconstructed Path | 999 | 1.1M | | 0 | | |
| agent-204 | 172.30.0.4 | 239.1.1.1 | Path 1 | 998 | 1.1M | 100.00 | 0 | | |
| | | | Path 2 | 0 | 0 | 0.00 | 1411934 | | |
| | | | Reconstructed Path | 998 | 1.1M | | 0 | | |

1: xNIC 2: Flows 3: Mesh 4: HA Paths 5: HA Summary 6: HA Streams 7: Config 8: PTP

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. Config

| Configurations | | | |
|-----------------|-----------|--------------|---|
| Entitlements | | Mesh | |
| ===== | | ===== | |
| Max Bandwidth | 2000 Mbps | Switches | 10.2.128.10, 10.5.1.6 |
| Max Clients | 10 | | |
| EnableMesh | true | HA | |
| EnableHA | true | ==== | |
| EnableUnicast | true | Switches | Path 1: {10.2.128.10}, Path 2: {10.5.1.6} |
| EnableClockSync | true | | |
| | | Unicast | |
| | | ===== | |
| | | Base Address | -- |
| | | Port Range | -- |
| | | Disable | -- |

1: xNIC 2: Flows 3: Mesh 4: HA Paths 5: HA Summary 6: HA Streams 7: Config 8: PTP

The Config 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, Unicast and Clock Sync (PTP).
- **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. PTP

Timing Nodes

Master Node

| | | | |
|------|----------|-------------------|---------------------|
| Name | core-200 | Time Sync Service | phc:/dev/ptp_hyperv |
|------|----------|-------------------|---------------------|

Follower Nodes

| Name | Status | Local Offset | Root Offset |
|-----------|---------|--------------|---------------|
| agent-201 | Present | 2.33 μ s | 19.10 μ s |
| agent-202 | Present | 2.65 μ s | 21.76 μ s |
| agent-204 | Present | 2.00 μ s | 15.89 μ s |

1: xNIC 2: Flows 3: Mesh 4: HA Paths 5: HA Summary 6: HA Streams 7: Config 8: PTP

The Precision Time Protocol (PTP) view displays information regarding the clock sync configuration for the cloudSwXtch. The page in swtch-top will only populate with information if the user has the PTP feature enabled.

In the example above, the cloudSwXtch (core-200) is acting as the Master Node.

- **Master Node-** The Master Node is what the PTP configuration sets as the most reliable time source. This will send the true time it receives from the source clock to the Follower Nodes.
 - **Name** - The name of the cloudSwXtch
 - **Time Sync Service** - The source clock
- **Follower Nodes-** The Follower Nodes lists the agents/VMs that subscribe to the Master Node for accurate timing.
 - **Name** - The name of the endpoints
 - **Status** - The status of the endpoints, noting if the node is active in the PTP configuration
 - **Local Offset** - The local offset denotes the offset in time from the cloudSwXtch to the xNIC.
 - **Root Offset** - The root offset denotes the offset in time from the GrandMaster clock to the cloudSwXtch and its follower nodes (xNIC). Note how the root is larger than the local. This is normal behavior since the distance between the follower node and the Grandmaster clock is greater than the offset between a cloudSwXtch and xNIC.

PTP Stabilization

After upgrading your cloudSwXtch system, you may notice that the local and root offset values are much larger than they actually are. It can take up to 30 minutes for the values to stabilize and return back to normal levels.

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 "SwtchSvcAddr" 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: `swxtch-tun0`

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

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

swtch-top on a cloudSwxtch

swtch-top should be run from a virtual machine with an xNIC installed, it should be avoided to run it or anything else directly on a cloudSwXtch. That being said it can be done, but you must run it with sudo. Only run it on the cloudSwXtch if doing advanced troubleshooting.

sudo /swtch/swtch-top dashboard --switch localhost

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

```
-----Timing Nodes-----
Master Node
=====
      Name      core-200      Time Sync Service phc:/dev/ptp_hyperv
Follower Nodes
=====
      Name      Status      Local Offset      Root Offset
agent-201      Present      2.33 µs      19.10 µs
agent-202      Present      2.65 µs      21.76 µs
agent-204      Present      2.00 µs      15.89 µs

1: xNIC 2: Flows 3: Mesh 4: HA Paths 5: HA Summary 6: HA Streams 7: Config 8: PTP
```

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 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.

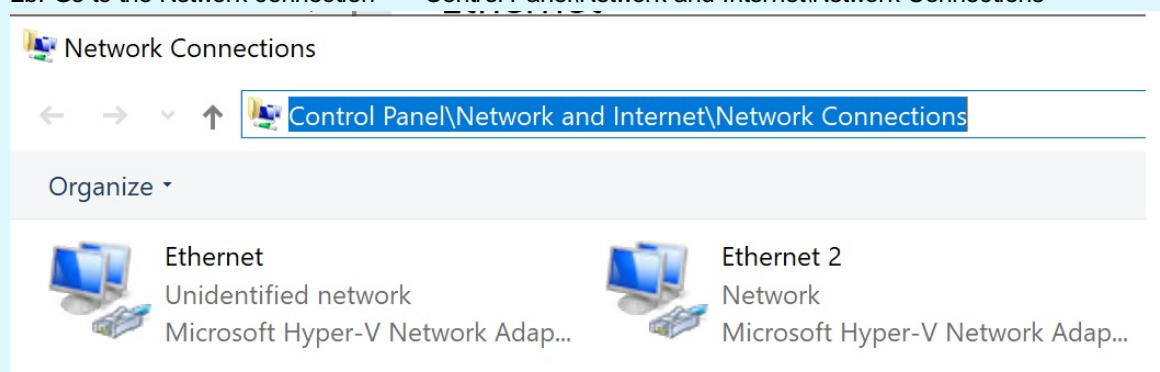
Visit https://go.microsoft.com/fwlink/?LinkId=217627 for additional information
about PowerShell commands for TCP/IP.
netsh interface ipv4>set subinterface "Ethernet 2" mtu=1200 store=persistent
Ok.
netsh interface ipv4>
```

{height="" width=""}

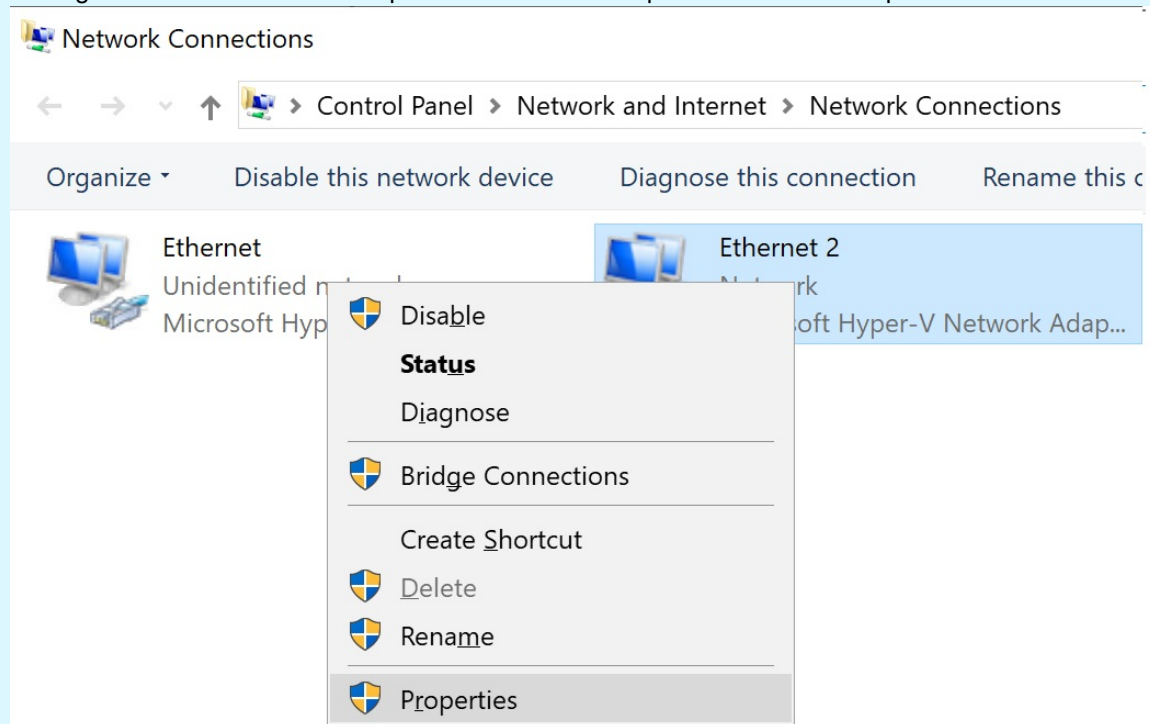
2a. If you get the following error then follow steps after error:

| | |
|--|------|
| None | Copy |
| <pre>netsh interface ipv4>set subinterface "Ethernet 2" mtu=2000 store=persistent The parameter is incorrect.</pre> | |

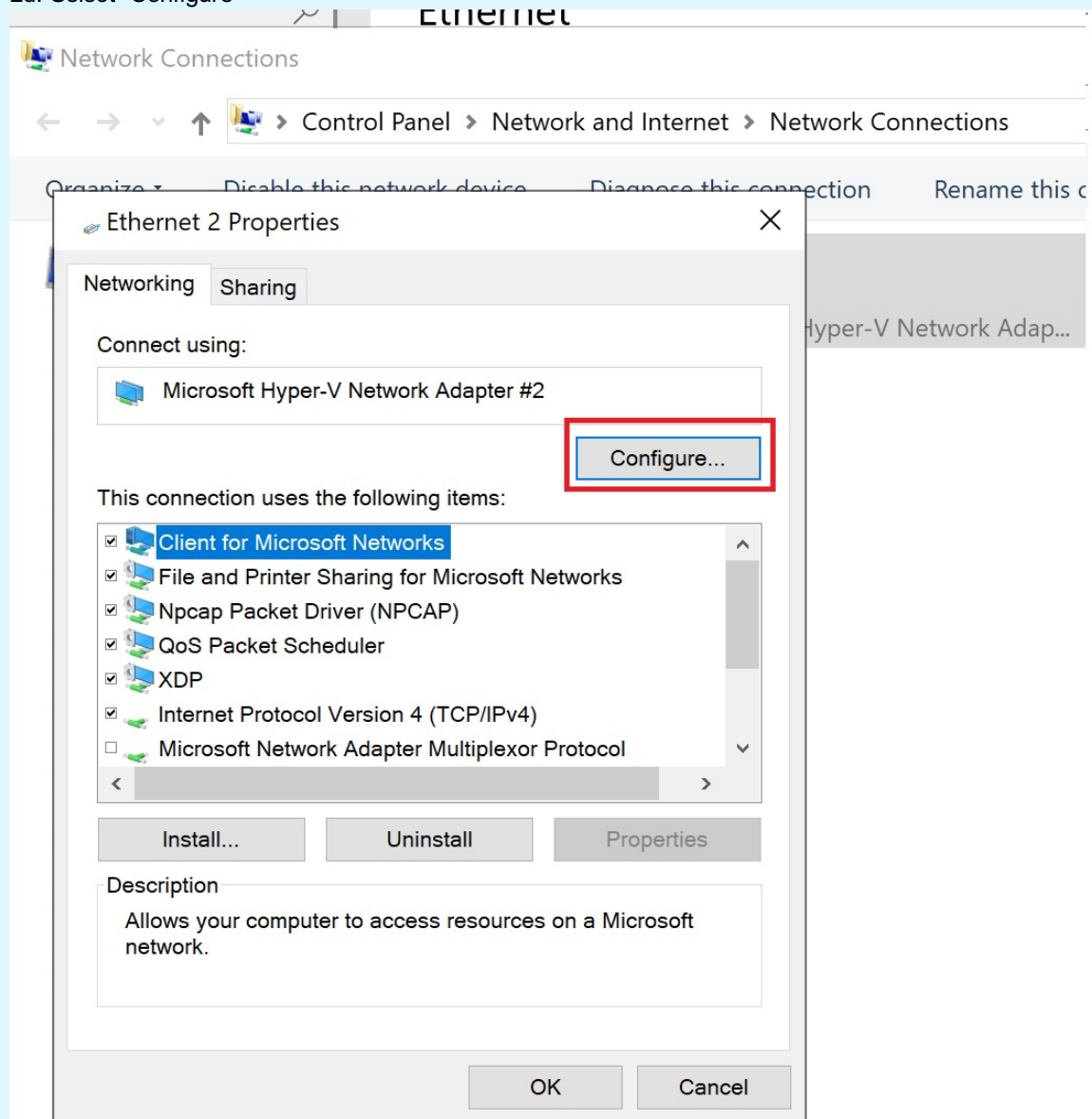
2b. Go to the Network connection → “Control Panel\Network and Internet\Network Connections”



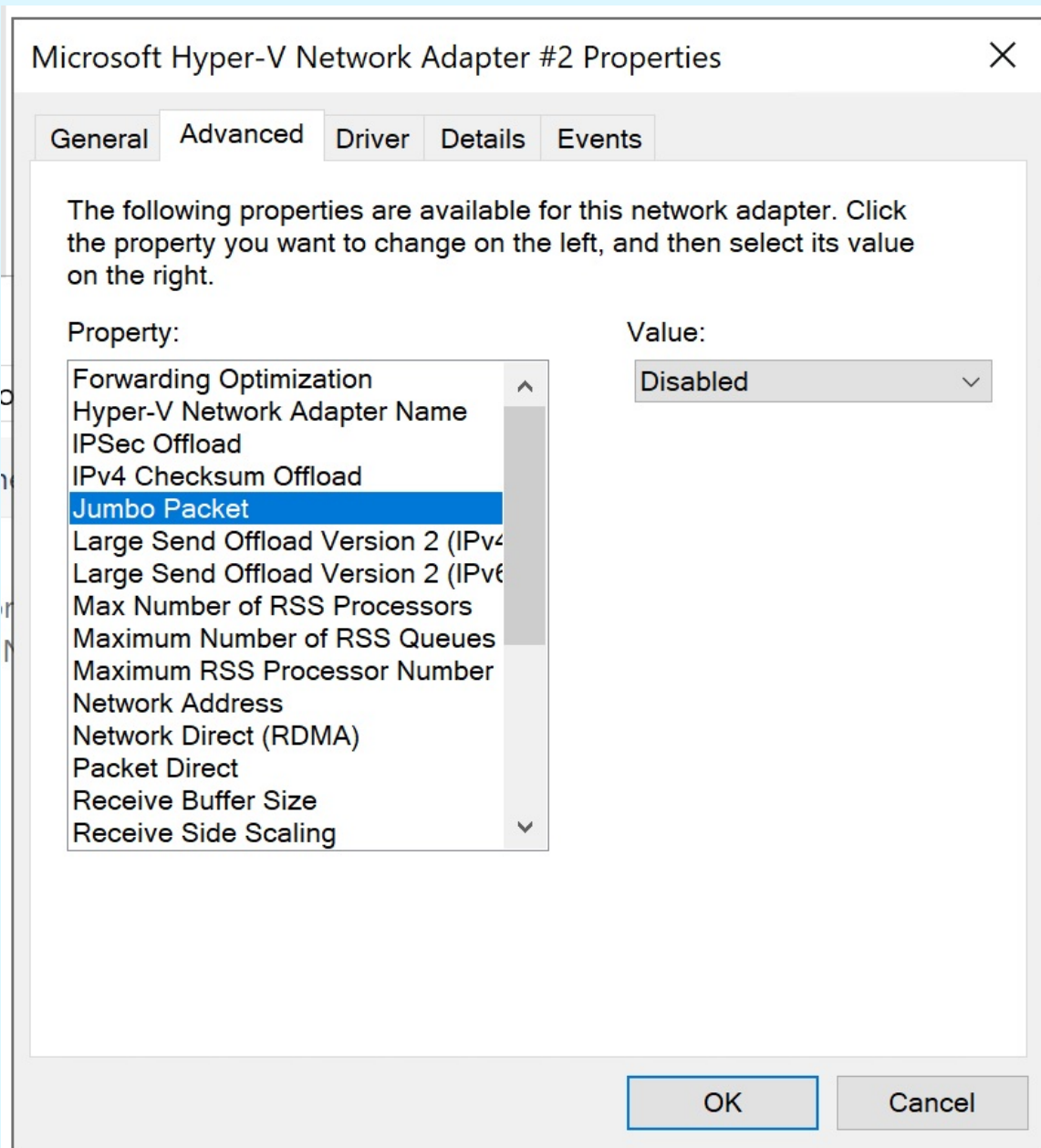
2c. Right click on the Ethernet Adaptor that the traffic is expected and select Properties.



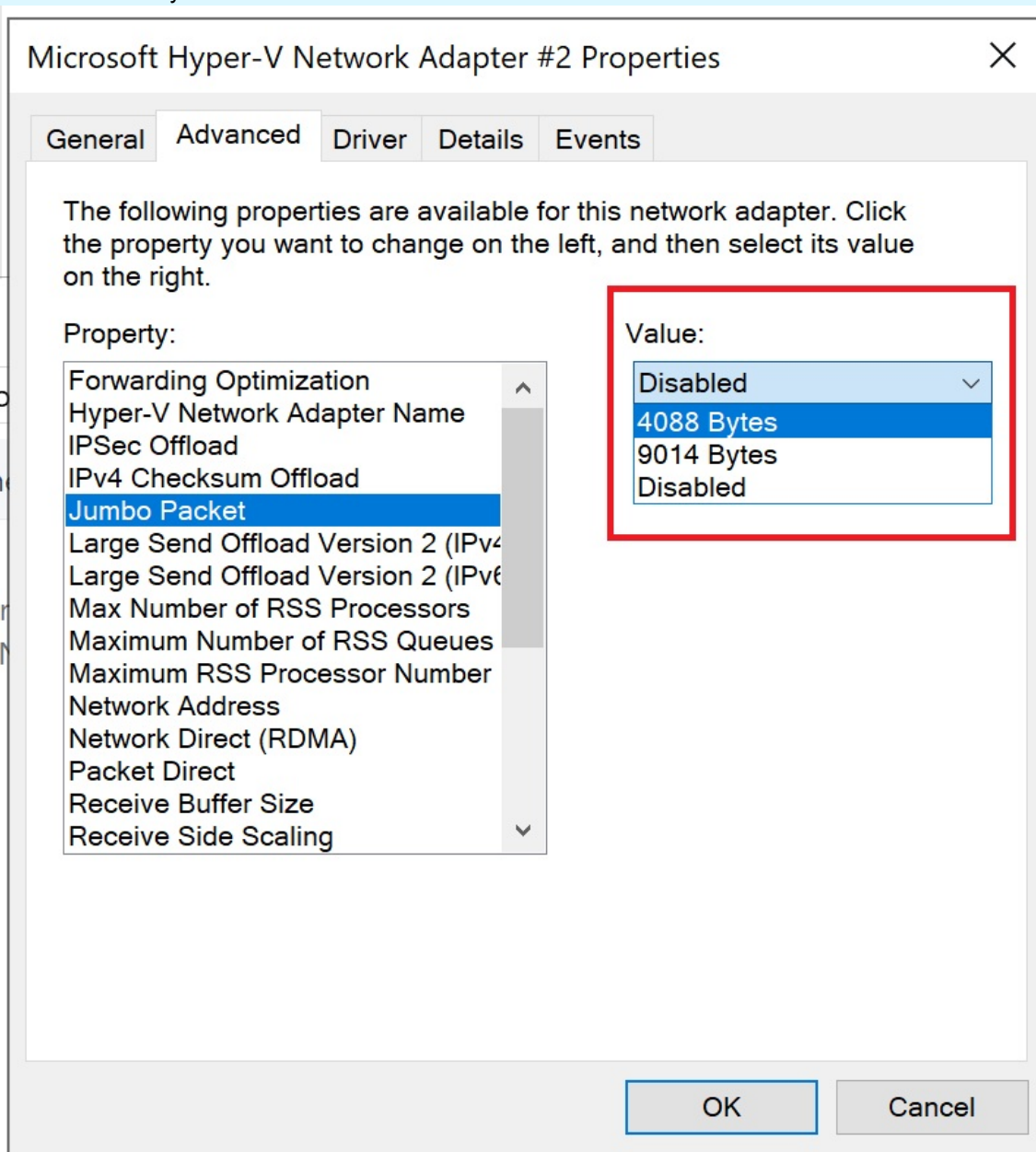
2d. Select "Configure"



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.

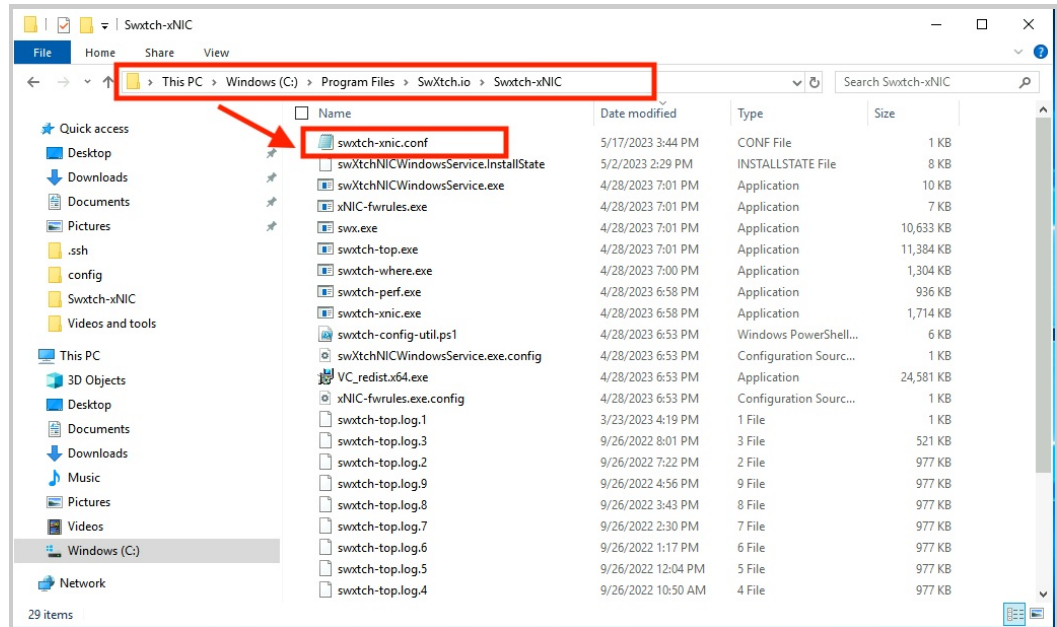
How to Change Verbosity

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

For Windows:

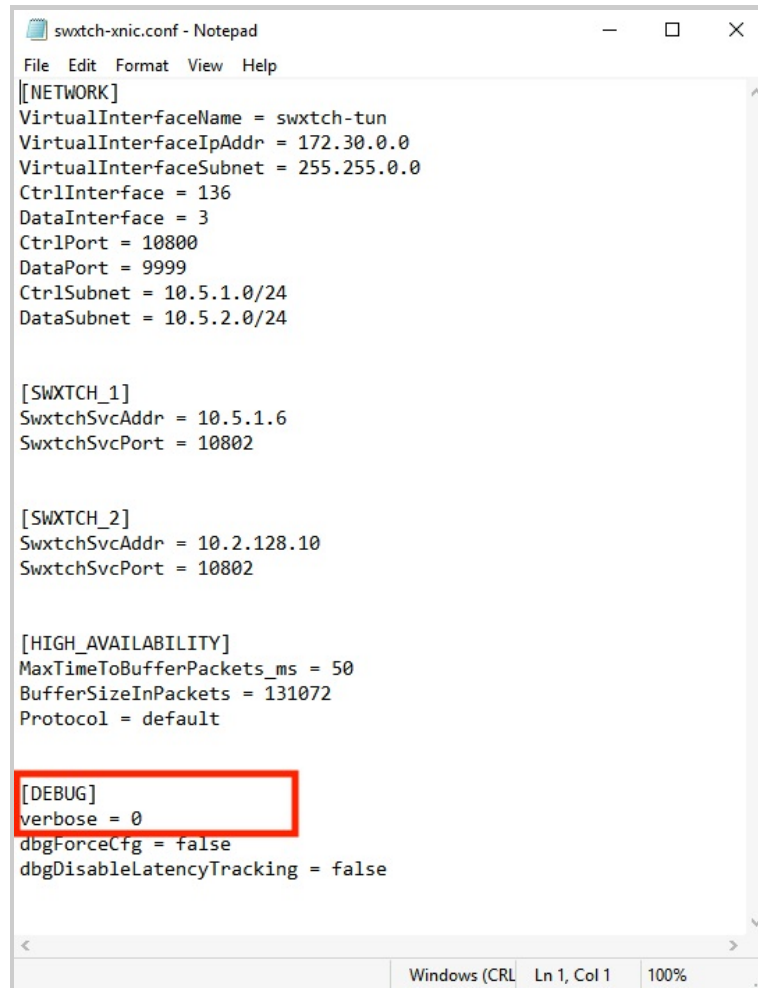
1. Go to the Swtch-xNIC folder on the VM you have an xNIC installed. Make sure it is the xNIC you want logs for.

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

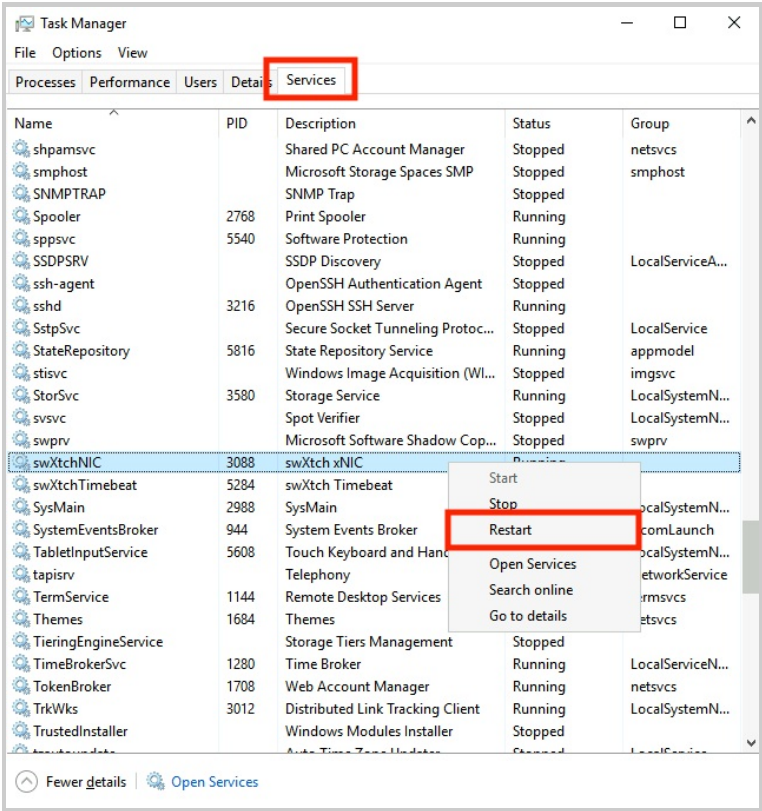
[HIGH_AVAILABILITY]
MaxTimeToBufferPackets_ms = 50
BufferSizeInPackets = 131072
Protocol = default

[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.

Media Use Cases

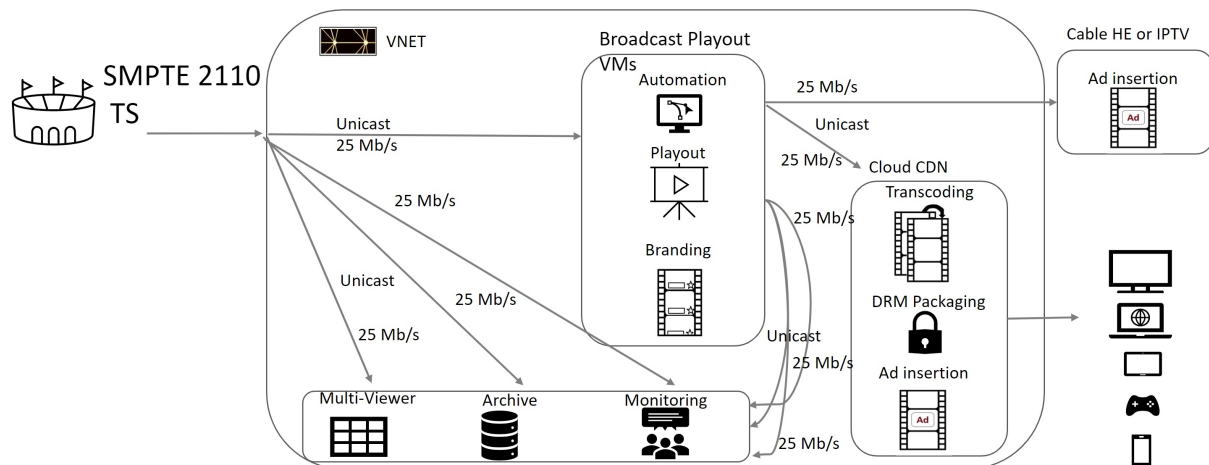
The media market can take advantage of several cloudSwXtch features such as:

- [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.



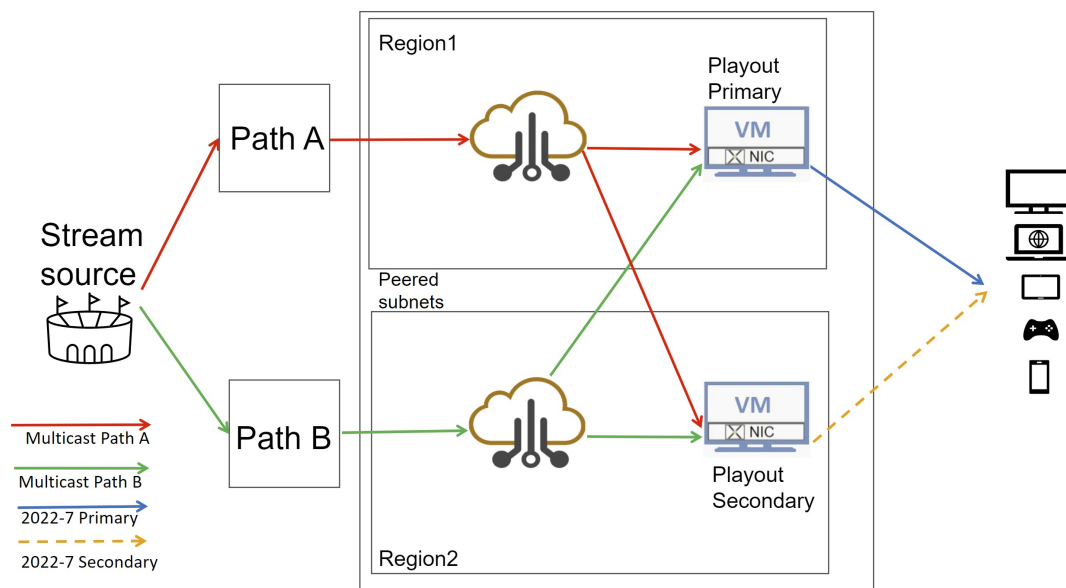
- ©2023 IEX Group, Inc. and its subsidiaries, including swXtch.io, Investors' Exchange LLC and IEX Services LLC. IEX Services LLC, Member SIPC/FINRA. All rights reserved.

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

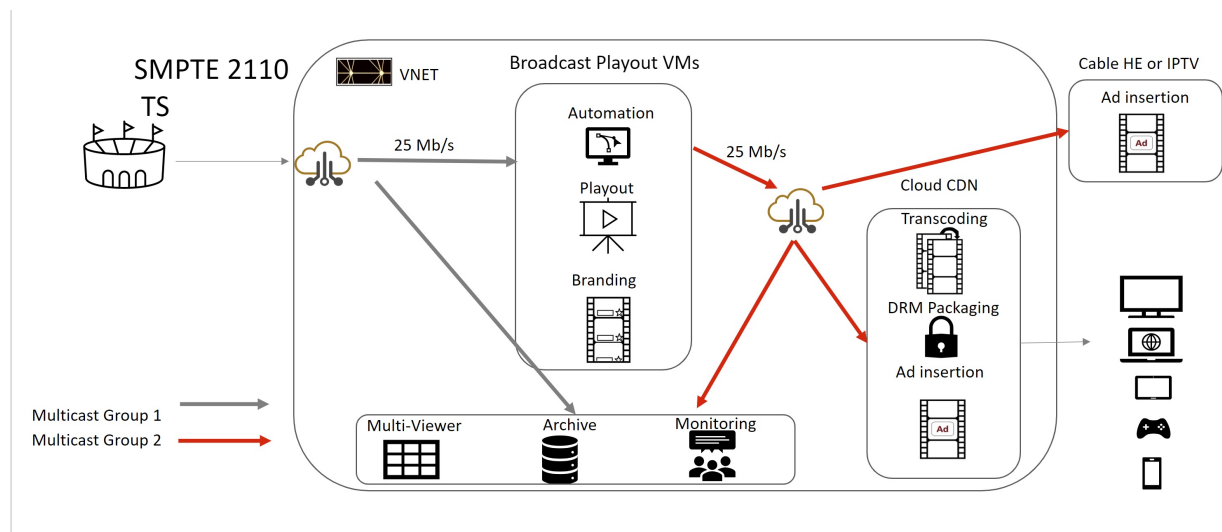


At **swXtch.io** we know that the media companies rely highly on both compressed and

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.

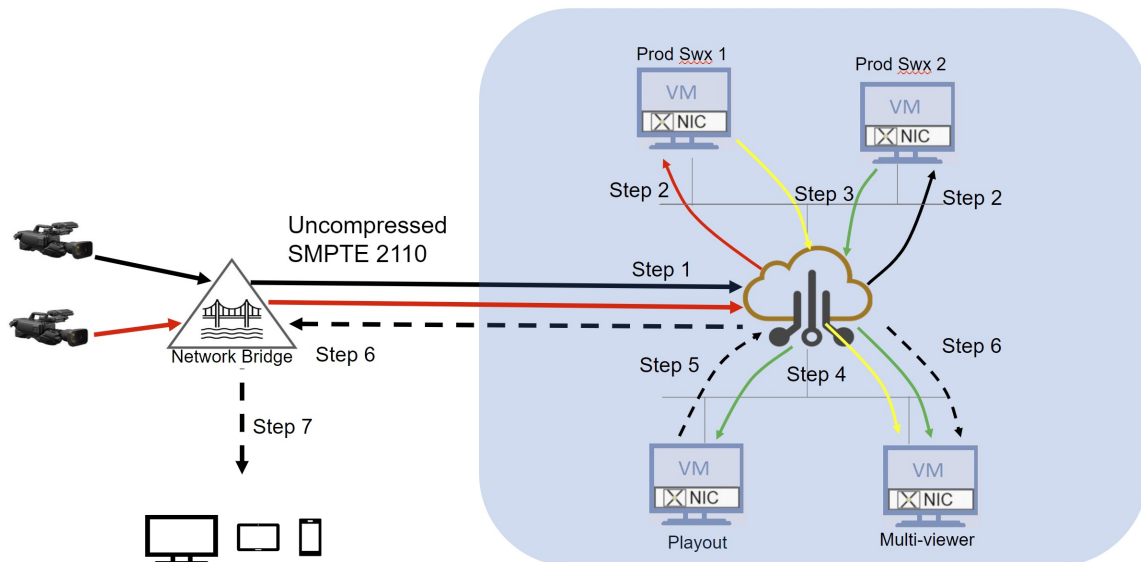
There are two workflow examples below, one is a compressed workflow and the other is an uncompressed workflow. The compressed workflow is a typical playout scenario where compressed inputs come into the cloud environment and are distributed via multicast to the necessary VM workloads by **cloudSwXtch**.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

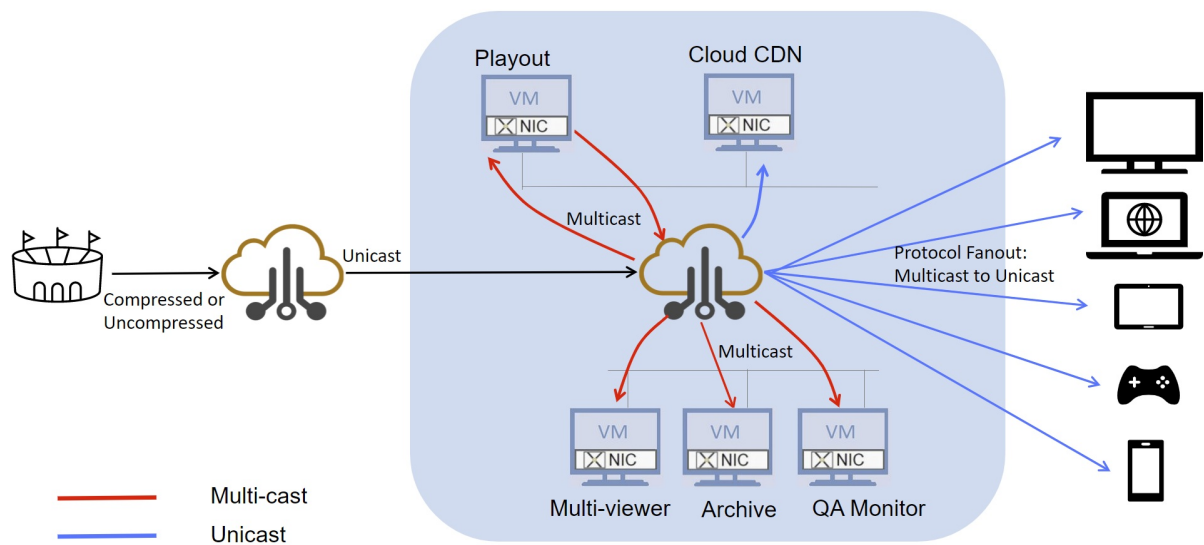
Step 7: The network bridge distributes to the clients for viewing consumption.

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

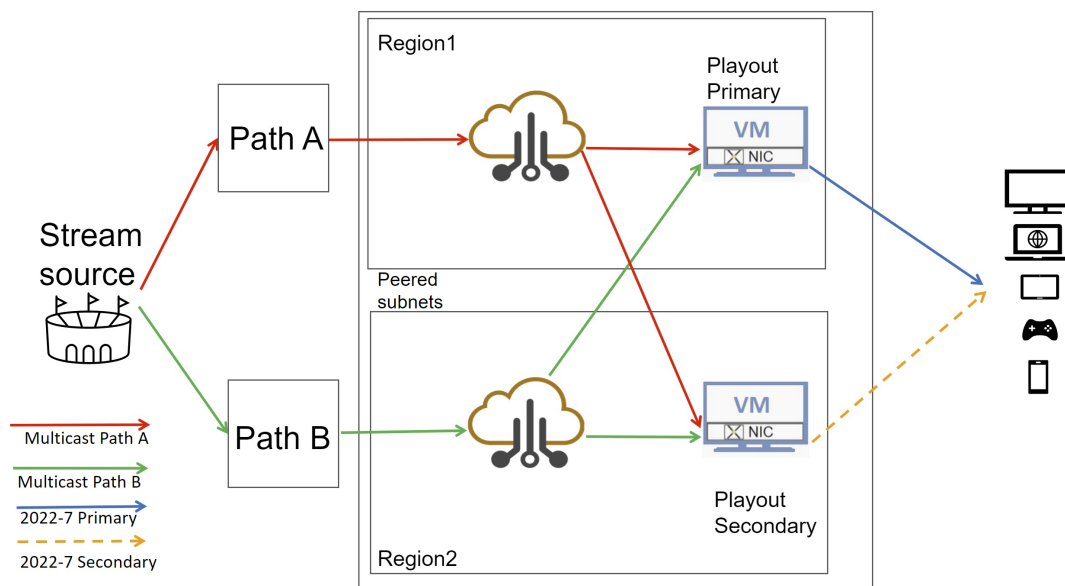
- The **cloudswXtch** in Region 1 can recieve the stream from Path A to Region 2.
- The **cloudswXtch** in Region 2 can recieve the stream from Path B to Region 1.
- If either path were to fail then the stream is still available in both Region 1 and 2 due to the redundancy.

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. Timestamps in the API are Epoch Unix Timestamp.

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

/api/wxckedeye/v1/dashboard

Request: Empty

Response:

| code | description |
|------|----------------------|
| 200 | successful operation |

Example:

Bash

Copy

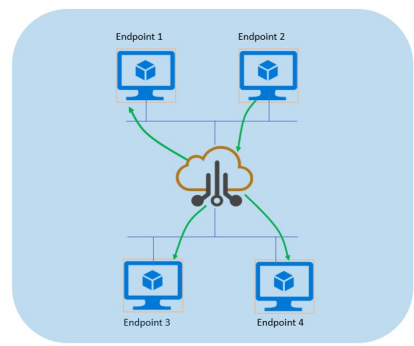
curl http://<cloudSwXtch-control-IP>/api/wxckedeye/v1/dashboard

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.

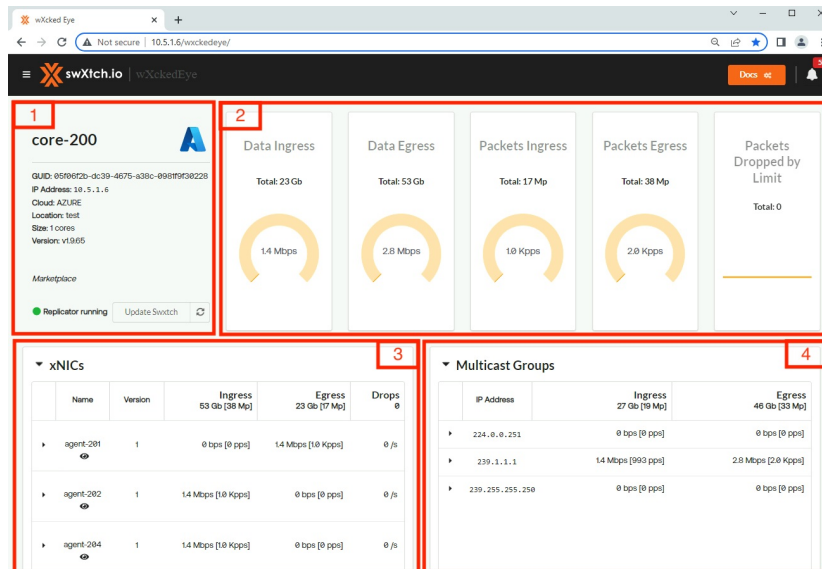
The example that will be used in this article refers to the network below, which has been simplified to help users understand the data returned in the API. As shown in Figure 1, Endpoint (Agent) 2 is sending data via multicast through the cloudSwXtch to Endpoints 1, 3 and 4.



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.

```

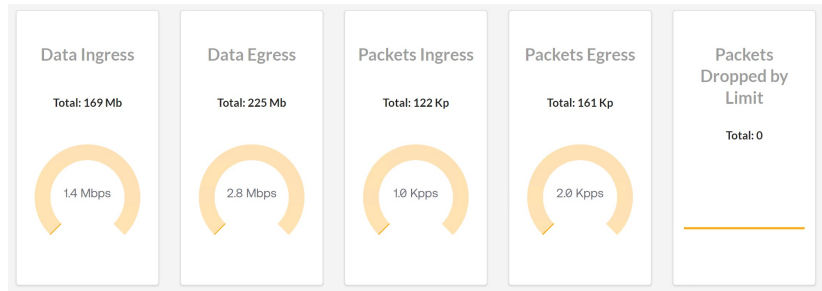
Bash Copy

{
  "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. xnic represents agents and xnicTotals as well as replTotals represents the cloudSwXtch.

```

Bash Copy
{
  "xnicTotals": {
    "PktCounters": {
      "Nic2McaTotal": 30218,
      "Nic2McaMc": 30218,
      "Mca2NicTotal": 31526,
      "Mca2NicMc": 31524,
      "Mca2NicIgmp": 6,
      "Mca2NicDrops": 0,
      "Nic2McaDrops": 0,
      "Mca2KniDrops": 0,
      "Kni2McaDrops": 0,
      "McaPktDrops": 0,
      "McaBigPktDrops": 0
    },
    "ByteCounters": {
      "Nic2McaTotal": 32347812,
      "Nic2McaMc": 32347812,
      "Mca2NicTotal": 33795772,
      "Mca2NicMc": 33795584
    },
    "Latencies": {
      "Count": 0,
      "Sum": 0,
      "Buckets": null
    },
    "HARxCounters": null,
    "Timestamp": 1657216501229157803,
    "SoftwareVersion": "",
    "XnicVersion": 0,
    "RxMulticastGroups": null,
    "TxMulticastGroups": null,
    "XnicMode": "",
    "NumConnections": 0
  },
  "Statistics totals of agents*",
  "Packet counters*",
  "Packets from swxtch to agent*",
  "Multicast packets from swxtch to agent*",
  "Packets from agent to swxtch*",
  "Multicast packets from agent to swxtch*",
  "IGMP packets from agent to swxtch*",
  "Packets lost from agent to swxtch*",
  "Packets lost from swxtch to agent*",
  "Packets lost from agent to kernel NIC*",
  "Packets lost from kernel NIC to agent*",
  "Packets lost at agent*",
  "Big size packets lost at agents*",
  "Bytes From swxtch to agent*",
  "Multicast bytes from swxtch to agent*",
  "Bytes from agent to swxtch*",
  "Multicast bytes from agent to swxtch*"
}

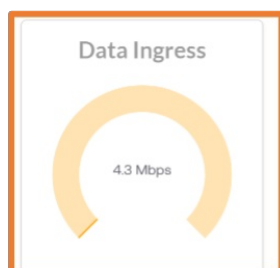
```

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.

Δ Something means the difference between the value of at time and Something at time t_1 and Something t_2 . Example: If Egress has value 39000 at time t_1 and value 19800 at time t_2 ,

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

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.

```

{
  "ByteCounters": {
    "Nic2McaTotal": 51730345462,
    "Nic2McaMc": 51730345462,
    "Mca2NicTotal": 51729942516,
    "Mca2NicMc": 51729941984
  },
  "Latencies": {
    "Count": 0,
    "Sum": 0,
    "Buckets": null
  },
  "HARxCounters": null,
  "Timestamp": 1658155900751865500,
  "ByteCounters": {
    "Nic2McaTotal": 51706144186,
    "Nic2McaMc": 51706144186,
    "Mca2NicTotal": 51721942820,
    "Mca2NicMc": 51721942288
  },
  "Latencies": {
    "Count": 0,
    "Sum": 0,
    "Buckets": null
  },
  "HARxCounters": null,
  "Timestamp": 1658155854228300382,
}

```

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

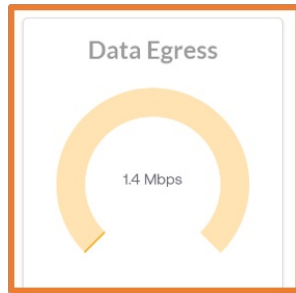
Timestamp from the call 2, 1658155900751865500 and the timestamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 in nanoseconds. Then, dividing the Δ by 1,000,000,000, we get the Δ of 46.523565312 in seconds, which we will use to calculate the data rate in bits per second.

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

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

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": 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, |

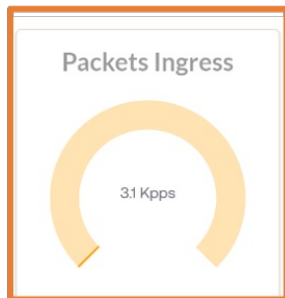
Taking the above expression and putting in data from this call is shown below.

Timestamp from call 2, 1658155900751865500 and 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": 297298583, | 18 | # | "Mca2NicMc": 297252607, |
| 19 | # | "Mca2NicKmp": 0, | 19 | # | "Mca2NicKmp": 0, |
| 20 | # | "Mca2NicDrops": 0, | 20 | # | "Mca2NicDrops": 0, |
| 21 | # | "Nic2McaDrops": 0, | 21 | # | "Nic2McaDrops": 0, |
| 22 | # | "Mca2KniDrops": 0, | 22 | # | "Mca2KniDrops": 0, |
| 23 | # | "Kni2McaDrops": 0, | 23 | # | "Kni2McaDrops": 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.

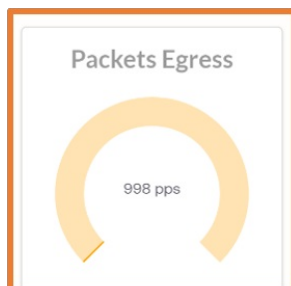
Timestamp from call 2, 1658155900751865500 and the timestamp from call 1, 1658155854228300382 gives us a Δ of 46523565312 nanoseconds. Then, dividing the Δ by 1,000,000,000, we get a Δ of 46.523565312 in seconds.

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

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

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.

$$\Delta Mca2NicTotal \text{ is } 297298589 - 297252613 = 988.2303665Packets$$

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

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

| Bash | Copy |
|---|------|
| <pre>"xnics": { *Stats of individual agents* "Agent-1": { "PktCounters": { *Packet Counters* "Nic2McaTotal": 75, *Packets from swtch to agent* "Nic2McaMc": 75, *Multicast packets from swtch to agent* "Mca2NicTotal": 31516, *Packets from agent to Swtch* "Mca2NicMc": 31516, *Multicast packets from agent to swtch* "Mca2NicIcmp": 2, *IGMP packets from agent to swtch* "Mca2NicDrops": 0, *Packets lost from agent to swtch* "Nic2McaDrops": 0, *Packets lost from swtch to agent* "Mca2KniDrops": 0, *Packets lost from agent to kernel NIC* "Kni2McaDrops": 0, *Packets lost from kernel NIC to agent* "McaPktDrops": 0, *Packets lost at agent* "McaBigPktDrops": 0 *Big size packets lost at agent* }, "ByteCounters": { "Nic2McaTotal": 8754, *Bytes From Swtch To Agent* "Nic2McaMc": 8754, *Multicast Bytes From Swtch To Agent* "Mca2NicTotal": 33794704, *Bytes From Agent To Swtch* "Mca2NicMc": 33794704 *Multicast bytes from agent to swtch* }, "Latencies": { "Count": 0, "Sum": 0, "Buckets": { "0.000050": 0, "0.000100": 0, "0.000250": 0, "0.000500": 0, "0.001000": 0, "0.002000": 0, "0.005000": 0, "0.010000": 0, "0.011000": 0 } }, "HARxCounters": [], *High Availability Rx Stats* "Timestamp": 1657216501229157803, "SoftwareVersion": "v1.7.4.draft", *Swtch Version* "XnicVersion": 2 *Agent Version* }, }</pre> | |

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

Bash

Copy

```
"Agent-2": {
  "PktCounters": {
    *Packet Counters*
    "Nic2McaTotal": 30107, *Packets from swtch to agent*
    "Nic2McaMc": 30107, *Multicast packets from swtch to agent*
    "Mca2NicTotal": 0, *Packets from agent to Swtch*
    "Mca2NicMc": 0, *Multicast packets from agent to swtch*
    "Mca2NicIcmp": 2, *IGMP packets from agent to swtch*
    "Mca2NicDrops": 0, *Packets lost from agent to swtch*
    "Nic2McaDrops": 0, *Packets lost from swtch to agent*
    "Mca2KniDrops": 0, *Packets lost from agent to kernel NIC*
    "Kni2McaDrops": 0, *Packets lost from kernel NIC to agent*
    "McaPktDrops": 0, *Packets lost at agent*
    "McaBigPktDrops": 0 *Big size packets lost at agent*
  },
  "ByteCounters": {
    "Nic2McaTotal": 32334918, *Bytes From Swtch To Agent*
    "Nic2McaMc": 32334918, *Multicast Bytes From Swtch To Agent*
    "Mca2NicTotal": 0, *Bytes From Agent To Swtch*
    "Mca2NicMc": 0 *Multicast bytes from agent to swtch*
  },
  "Latencies": {
    "Count": 0,
    "Sum": 0,
    "Buckets": {
      "0.000050": 0,
      "0.000100": 0,
      "0.000250": 0,
      "0.000500": 0,
      "0.001000": 0,
      "0.002000": 0,
      "0.005000": 0,
      "0.010000": 0,
      "0.011000": 0
    }
  },
  "HARxCounters": [], *High Availability Rx Stats*
  "Timestamp": 1657216500709825713,
  "SoftwareVersion": "v1.7.4.draft", *Swtch Version*
  "XnicVersion": 2 *Agent Version*
},
```

Agent #3

Bash

Copy

```
"Agent-3": {
  "PktCounters": {
    *Packet Counters*
    "Nic2McaTotal": 18, *Packets from swtch to agent*
    "Nic2McaMc": 18, *Multicast packets from swtch to agent*
    "Mca2NicTotal": 9, *Packets from agent to Swtch*
    "Mca2NicMc": 8, *Multicast packets from agent to swtch*
    "Mca2NicIcmp": 1, *IGMP packets from agent to swtch*
    "Mca2NicDrops": 0, *Packets lost from agent to swtch*
    "Nic2McaDrops": 0, *Packets lost from swtch to agent*
    "Mca2KniDrops": 0, *Packets lost from agent to kernel NIC*
    "Kni2McaDrops": 0, *Packets lost from kernel NIC to agent*
    "McaPktDrops": 0, *Packets lost at agent*
    "McaBigPktDrops": 0 *Big size packets lost at agent*
  },
  "ByteCounters": {
    "Nic2McaTotal": 2070, *Bytes From Swtch To Agent*
    "Nic2McaMc": 2070, *Multicast Bytes From Swtch To Agent*
    "Mca2NicTotal": 974, *Bytes From Agent To Swtch*
    "Mca2NicMc": 880 *Multicast bytes from agent to swtch*
  },
  "Latencies": {
    "Count": 18,
    "Sum": 302211676294169800,
    "Buckets": {
      "0.000050": 0,

```

```

        "0.000100": 0,
        "0.000250": 0,
        "0.000500": 0,
        "0.001000": 0,
        "0.002000": 0,
        "0.005000": 0,
        "0.010000": 0,
        "0.011000": 0
    }
},
"HARxCounters": [],          *High Availability Rx Stats*
"Timestamp": 1657216500559694400,
"SoftwareVersion": "v1.7.4.draft",    *Swxtch Version*
"XnicVersion": 2              *Agent Version*
},

```

Agent #4

| Bash | Copy |
|--|------|
| <pre> "Agent-4": { "PktCounters": { *Packet Counters* "Nic2McaTotal": 18, *Packets from swxtch to agent* "Nic2McaMc": 18, *Multicast packets from swxtch to agent* "Mca2NicTotal": 1, *Packets from agent to Swxtch* "Mca2NicMc": 0, *Multicast packets from agent to swxtch* "Mca2NicIgmp": 1, *IGMP packets from agent to swxtch* "Mca2NicDrops": 0, *Packets lost from agent to swxtch* "Nic2McaDrops": 0, *Packets lost from swxtch to agent* "Mca2KniDrops": 0, *Packets lost from agent to kernel NIC* "Kni2McaDrops": 0, *Packets lost from kernel NIC to agent* "McaPktDrops": 0, *Packets lost at agent* "McaBigPktDrops": 0 *Big size packets lost at agent* }, "ByteCounters": { "Nic2McaTotal": 2070, *Bytes From Swxtch To Agent* "Nic2McaMc": 2070, *Multicast Bytes From Swxtch To Agent* "Mca2NicTotal": 94, *Bytes From Agent To Swxtch* "Mca2NicMc": 0 *Multicast bytes from agent to swxtch* }, "Latencies": { "Count": 18, "Sum": 302211676294180540, "Buckets": { "0.000050": 0, "0.000100": 0, "0.000250": 0, "0.000500": 0, "0.001000": 0, "0.002000": 0, "0.005000": 0, "0.010000": 0, "0.011000": 0 } }, "HARxCounters": [], *High Availability Rx Stats* "Timestamp": 1657216500954028100, "SoftwareVersion": "v1.7.4.draft", *Swxtch Version* "XnicVersion": 2 *Agent Version* }, </pre> | |

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: $([xnic][< agentName >][ByteCounters](\Delta Nic2McaTotal * 8))/((\Delta Timestamp/1,000,000,000))$
- Pkts Ingress: $([xnic][< agentName >][PktCounters](\Delta Nic2McaTotal))/((\Delta Timestamp/1,000,000,000))$

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 | "Mca2McaDrops": 0, | 52 | "Mca2McaDrops": 0, |
| 53 | "Mca2KniDrops": 0, | 53 | "Mca2KniDrops": 0, |
| 54 | "Mca2McaDrops": 0, | 54 | "Mca2McaDrops": 0, |
| 55 | "McaPktDrops": 0, | 55 | "McaPktDrops": 0, |
| 56 | "McaBigPktDrops": 0 | 56 | "McaBigPktDrops": 0 |
| 57 | }, | 57 | }, |
| 58 | "ByteCounters": { | 58 | "ByteCounters": { |
| 59 | "Nic2McaTotal": 51694152976, | 59 | "Nic2McaTotal": 51702156848, |
| 60 | "Nic2McaMc": 51694152976, | 60 | "Nic2McaMc": 51702156848, |
| 61 | "Mca2NicTotal": 24514, | 61 | "Mca2NicTotal": 24514, |
| 62 | "Mca2NicMc": 24514 | 62 | "Mca2NicMc": 24514 |
| 63 | }, | 63 | }, |
| 64 | "Latencies": { | 64 | "Latencies": { |
| 65 | "Count": 0, | 65 | "Count": 0, |
| 66 | "Sum": 0, | 66 | "Sum": 0, |
| 67 | "Buckets": { | 67 | "Buckets": { |
| 68 | "0.000050": 0, | 68 | "0.000050": 0, |
| 69 | "0.000100": 0, | 69 | "0.000100": 0, |
| 70 | "0.000250": 0, | 70 | "0.000250": 0, |
| 71 | "0.000500": 0, | 71 | "0.000500": 0, |
| 72 | "0.001000": 0, | 72 | "0.001000": 0, |
| 73 | "0.002000": 0, | 73 | "0.002000": 0, |
| 74 | "0.005000": 0, | 74 | "0.005000": 0, |
| 75 | "0.010000": 0, | 75 | "0.010000": 0, |
| 76 | "0.011000": 0 | 76 | "0.011000": 0 |
| 77 | }, | 77 | }, |
| 78 | }, | 78 | }, |
| 79 | "HARxCounters": {}, | 79 | "HARxCounters": {}, |
| 80 | "Timestamp": 1658155853888965889, | 80 | "Timestamp": 1658155900043488705, |

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: $\Delta Nic2McaTotal * 8 = 64030976bits$

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

Pkts Ingress: $\Delta Nic2McaTotal$ is $297138907 - 297092907 = 46000Packets$

$([xnic][DSd - agent - 101][PktCounters](\Delta Nic2McaTotal))/((\Delta Timestamp/1,000,000,000)) = 46000/46.15452288 = 996.65 \approx 997pps$

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: $[xnicTotals][ByteCounters]Mca2NicTotal * 8$
- Total Packets: $[xnicTotals][PktCounters]Mca2NicTotal$
- Data Egress: $([xnic][< agentName >][ByteCounters](\Delta Mca2NicTotal * 8))/((\Delta Timestamp/1,000,000,000))$
- Pkts Egress: $([xnic][< agentName >][PktCounters](\Delta Mca2NicTotal))/((\Delta Timestamp/1,000,000,000))$

| | |
|--|--|
| <pre> }, "Dsd-agent-102": { "PktCounters": { "Nic2McaTotal": 0, "Nic2McaMc": 0, "Mca2NicTotal": 297252401, "Mca2NicMc": 297252401, "Mca2NicIcmp": 0, "Mca2NicDrops": 0, "Nic2McaDrops": 0, "Mca2KniDrops": 0, "Kni2McaDrops": 0, "McaPktDrops": 0, "McaBigPktDrops": 0 }, "ByteCounters": { "Nic2McaTotal": 0, "Nic2McaMc": 0, "Mca2NicTotal": 5172191774, "Mca2NicMc": 5172191774 }, "Latencies": { "Count": 0, "Sum": 0, "Buckets": { "0.000050": 0, "0.000100": 0, "0.000250": 0, "0.000500": 0, "0.001000": 0, "0.002000": 0, "0.005000": 0, "0.010000": 0, "0.011000": 0 } }, "HARxCounters": [], "Timestamp": 165815585395371860, "SoftwareVersion": "v1.7.4.draft", "XnicVersion": 2 } }, "HARxCounters": [], "Timestamp": 165815585395371860, "SoftwareVersion": "v1.7.4.draft", "XnicVersion": 2 } </pre> | <pre> }, "Dsd-agent-102": { "PktCounters": { "Nic2McaTotal": 0, "Nic2McaMc": 0, "Mca2NicTotal": 297298375, "Mca2NicMc": 297298375, "Mca2NicIcmp": 0, "Mca2NicDrops": 0, "Nic2McaDrops": 0, "Mca2KniDrops": 0, "Kni2McaDrops": 0, "McaPktDrops": 0, "McaBigPktDrops": 0 }, "ByteCounters": { "Nic2McaTotal": 0, "Nic2McaMc": 0, "Mca2NicTotal": 51729917250, "Mca2NicMc": 51729917250 }, "Latencies": { "Count": 0, "Sum": 0, "Buckets": { "0.000050": 0, "0.000100": 0, "0.000250": 0, "0.000500": 0, "0.001000": 0, "0.002000": 0, "0.005000": 0, "0.010000": 0, "0.011000": 0 } }, "HARxCounters": [], "Timestamp": 1658155900086539499, "SoftwareVersion": "v1.7.4.draft", "XnicVersion": 2 } }, "HARxCounters": [], "Timestamp": 1658155900086539499, "SoftwareVersion": "v1.7.4.draft", "XnicVersion": 2 } </pre> |
|--|--|

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

Timestamp from call 2, 1658155900086539499 and timestamp from call 1, 165815585395371860 gives us a Δ of 46131167744 nanoseconds. Then, dividing the Δ by 1,000,000,000, we get Δ of 46.131167744 in seconds.

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

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

Pkts Egress: $\Delta Mca2NicTotal$ is $297298375 - 297252401 = 45974Packets$

$([xnic][Dsd - agent - 102][PktCounters](\Delta Mca2NicTotal))/((\Delta Timestamp/1,000,000,000)) = 45974/(46.131167744) = 996.593 \approx 997pps$

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

xNIC Drops - Section 3

| xNICs | | | |
|---------------|--------------------------|-------------------------|------------|
| Name | Ingress 77 Mb [55 Kp] | Egress 25 Mb [18 Kp] | Drops 0 |
| Dsd-agent-101 | 1.4 Mbps [999 pps] | 0 bps [0 pps] | 0 /s |
| Dsd-agent-102 | 0 bps [0 pps] | 1.4 Mbps [998 pps] | 0 /s |
| Dsd-agent-104 | 1.4 Mbps [1.0 Kpps] | 0 bps [0 pps] | 0 /s |
| Dsd-agent-105 | 1.4 Mbps [1.0 Kpps] | 0 bps [0 pps] | 0 /s |

Total: $xnicTotals[PktCounters]Mca2NicDrops$

Drop(s): $([xnic][< agentName >][PktCounters](\Delta Mca2NicDrops))/((\Delta Timestamp/1,000,000,000))$

Since there were no drops at the time of the calls in this document, there are no examples for the calculations.

Section 4: Multicast Data

| ▼ Multicast Groups | | |
|--------------------|----------------------|---------------------|
| IP Address | Ingress 0 b [0 p] | Egress 0 b [0 p] |
| 224.0.0.251 | 0 bps [0 pps] | 0 bps [0 pps] |
| 239.1.1.1 | 173 Kbps [996 pps] | 520 Kbps [3.0 Kpps] |
| 239.255.255.250 | 0 bps [0 pps] | 0 bps [0 pps] |

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

```

Bash Copy
},
"replTotals": {
    "host": "",
    "sequence": 95924,
    "rxCount": 31535,
    "txCount": 30332,
    "rxBytes": 33796794,
    "txBytes": 32361180,
    "rxBridgeBytes": 0,
    "rxBridgeCount": 0,
    "timestamp": 1657216500274611599,
    "dropsByByteLimit": 0,
    "dropsByCountLimit": 0,
    "rxMeshPktCount": 0,
    "rxMeshBytes": 0,
    "txMeshPktCount": 0,
    "txMeshBytes": 0,
    "rxUnicastPktCount": 0,
    "rxUnicastBytes": 0,
    "txUnicastPktCount": 0,
    "txUnicastBytes": 0,
    "rxMulticastGroups": [
        {
            "groupId": "239.1.1.3",
            "pktsCount": 31460,
            "bytesCount": 33788040,
            "lastUpdate": "2022-07-07T15:29:00.789006233Z"
        },
        {
            "groupId": "239.0.0.251",
            "pktsCount": 75,
            "bytesCount": 8754,
            "lastUpdate": "2022-07-07T17:54:49.132519721Z"
        }
    ],
    "subscriptionId": "d723b93f-112c-49fb-9fda-03d3ada867f3",
    "hostName": "dsd-core-174",
    "numCores": 8,
    "replStatus": "running",
    "authorized": true,
    "validationResult": {
        "switch": 556,
        "authorized": true,
        "denialReason": null,
        "trialPeriod": {
            "startDate": "2021-12-22T19:38:33.565336Z",
            "endDate": "2022-08-21T19:38:33.565Z"
        }
    },
    "applicationId": null,
    "isMarketplace": true,

```

```
    "meshes": {
      "uid": "DF91521E-7202-A3C2-8156-1FF5070545E9",
      "swxtches": [
        "10.2.128.10",
        "10.5.1.6"
      ]
    },
    "expiresAt": "2033-04-05T21:09:00Z"
  }
}
```

Multicast Ingress Data - Section 1

▼ Multicast Groups

| IP Address | Ingress 0 b [0 p] |
|-----------------|----------------------|
| 224.0.0.251 | 0 bps [0 pps] |
| 239.1.1.1 | 173 Kbps [996 pps] |
| 239.255.255.250 | 0 bps [0 pps] |

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

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

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

| | | |
|--|-----|--|
| "rxMulticastGroups": { | 221 | "rxMulticastGroups": { |
| { | 222 | { |
| "groupId": "239.255.255.250", | 223 | "groupId": "239.255.255.250", |
| "pktsCount": 956, | 224 | "pktsCount": 956, |
| "bytesCount": 238044, | 225 | "bytesCount": 238044, |
| "lastUpdate": "2022-07-19T21:29:50.013001632Z" | 226 | "lastUpdate": "2022-07-19T21:29:50.013001632Z" |
| }, | 227 | }, |
| { | 228 | { |
| "groupId": "239.1.1.1", | 229 | "groupId": "239.1.1.1", |
| "pktsCount": 540049064, | 230 | "pktsCount": 540092012, |
| "bytesCount": 93968537136, | 231 | "bytesCount": 93976010088, |
| "lastUpdate": "2022-07-25T19:44:40.189630746Z" | 232 | "lastUpdate": "2022-07-25T19:45:23.191477537Z" |
| }, | 233 | }, |
| }, | 234 | }, |

$\Delta lastUpdate = 43.00003982 seconds$

Data Ingress:

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

$([xnics][RXMulticastGroups]\Delta bytesCount * 8) / \Delta lastUpdate = 59783616 / 43.00003982 = 1390315.364 \approx 1.4 Mbps.$

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

Packets Ingress:

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

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

Multicast Egress Data - Section 2

▼ 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 | }, | 241 | }, |
| 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": 104003710780, |
| 247 | "lastUpdate": "2022-07-25T19:44:40.162376549Z" | 247 | "lastUpdate": "2022-07-25T19:45:23.162418365Z" |
| 248 | }, | 248 | }, |
| 249 | { | 249 | { |
| 250 | "groupId": "224.0.0.251", | 250 | "groupId": "224.0.0.251", |
| 251 | "pktsCount": 869, | 251 | "pktsCount": 875, |
| 252 | "bytesCount": 115212, | 252 | "bytesCount": 115872, |
| 253 | "lastUpdate": "2022-07-25T19:28:10.061601872Z" | 253 | "lastUpdate": "2022-07-25T19:44:41.162375932Z" |
| 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.

$\Delta lastUpdate = 43seconds$

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

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

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

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

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

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:

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

```

```

        "groupId": "239.2.1.1"
      }
    },
    "streams": {
      "streams": {
        "239.1.1.1": "Hockey"
      }
    }
  }
}

```

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.

The screenshot shows the 'Settings' page for a device named 'dsd-core-300'. The 'General' tab is selected, showing device information and entitlements.

Information:

- Version: v1.9.85
- Size: 1
- Ctrl IP: 10.1.1.6
- Ctrl Port: 19892
- Subnet Data Prefix: 10.1.2.0/24
- Subnet Ctrl Prefix: 10.1.1.0/24
- Gateway IP: 10.1.2.1

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. Slider set to 5 secs.

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": {

```

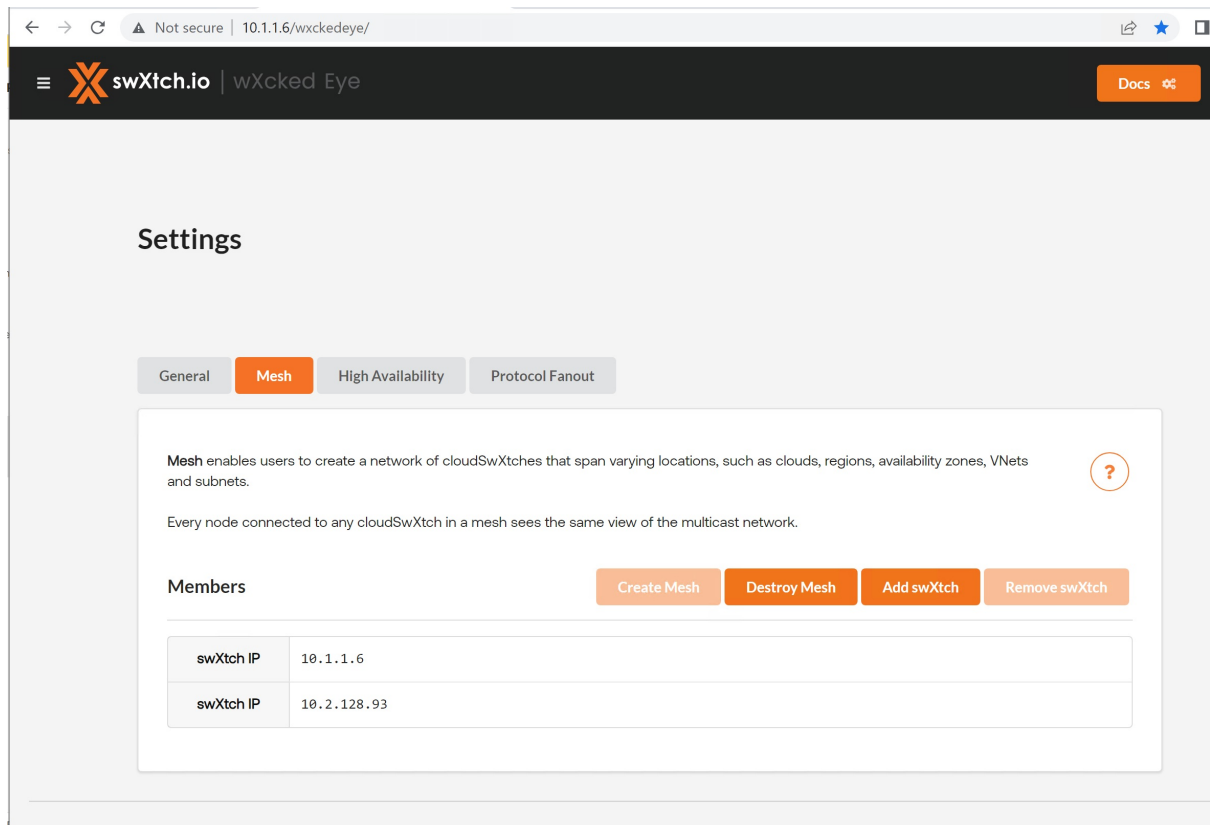
```

        "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:

| Bash | Copy |
|--|------|
| <pre> meshes: { "meshId": "86B62026-9222-0E62-03C2-6C5872CA64C5", "swxtches": ["10.1.1.6", "10.2.128.93"], "uid": null }, </pre> | |

Create Mesh

POST

swxtch/mesh/v1/

- Create a mesh of cloudSwXtches

URL:

Bash

Copy

```
curl http://<cloudSwXtch-control-IP>swxtch/mesh/v1/create
```

Example URL:

<http://10.1.1.6/swxtch/mesh/v1/addSwitch>

Request Body:

Bash

Copy

```

{
  "participants": [
    "10.1.1.6",
    "10.2.128.93"
  ]
}

```

Responses:

200 - successful operation

Example Value:

None

List Mesh Members

GET

swxtch/mesh/v1/listMemberAddress

- Returns a list of cloudSwXtches connected in a mesh configuration.

URL:

Bash

[Copy](#)

```
curl http://<cloudSwXtch-control-IP>/swxtch/mesh/v1/listMemberAddress
```

Example URL:

http://10.1.1.6/swxtch/mesh/v1/listMemberAddress

Request Body:

None

Responses:

200 - successful operation

Example Value:

Bash

[Copy](#)

```
[  
  "10.1.1.6",  
  "10.2.128.93"  
]
```

Add cloudSwXtch to Mesh

POST

swxtch/mesh/v1/addSwitch

- Adds a cloudSwXtch to a Mesh configuration

URL:

Bash

[Copy](#)

```
curl http://<cloudSwXtch-control-IP>/swxtch/mesh/v1/addSwitch
```

Example URL:

http://10.1.1.6/swxtch/mesh/v1/addSwitch

Example Request Body:

Bash

[Copy](#)

```
{
  "switch": "10.5.1.6"
}
```

Responses:

200 - successful operation

Example Value:

None

Remove cloudSwXtch from Mesh

POST

swxtch/mesh/v1/removeSwitch

- Removes a cloudSwXtch from a Mesh configuration

URL:

Bash

[Copy](#)

```
curl http://<cloudSwXtch-control-IP>/swxtch/mesh/v1/removeSwitch
```

Example URL:

http://10.1.1.6/swxtch/mesh/v1/removeSwitch

Example Request Body:

Bash

[Copy](#)

```
{
  "switch": "10.5.1.6"
}
```

Responses:

200 - successful operation

Example Value:

None

Destroy Mesh

POST

swxtch/mesh/v1/destroy

- Destroys a Mesh configuration

URL:

Bash

[Copy](#)

```
curl http://<cloudSwXtch-control-IP>/swxtch/mesh/v1/destroy
```

Example URL:

http://10.1.1.6/swxtch/mesh/v1/destroy

Example Request Body:

None

Responses:

200 - successful operation

Example Value:

None

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.

The screenshot shows the 'Settings' page for 'swXtch.io' with the 'High Availability' tab selected. The page title is 'Settings'. Below the title are four tabs: 'General', 'Mesh', 'High Availability' (active), and 'Protocol Fanout'. The 'High Availability' section contains a description: 'High Availability (HA) protects users from data loss by replicating and sending packets through multiple network paths. High Availability compares packets received from those multiple paths and automatically reconstructs the original stream.' Below this is a section for 'Donna-ha' with buttons 'Create HA', 'Destroy HA', and 'Remove Swtch'. Underneath is a 'Paths' section with three entries: 'path_1' with IP '10.2.128.93', 'path_2' with IP '10.1.1.6', and 'path_3' with IP '10.5.1.6'.

Below is a portion of the Settings call response detailing information on High Availability:

| Bash | Copy |
|--|------|
| <pre>"haSettings": { "uid": "86A6BDD5-128D-E31D-4A7B-33D846C94CB2", "name": "ha", "paths": [{ "name": "path_1", "swxtches": ["10.2.128.93"] }, { "name": "path_2", "swxtches": ["10.1.1.6"] }, { "name": "path_3", "swxtches": ["10.5.1.6"] }] }</pre> | |

Create HA

POST

swxtch/ha/v1/create

- Creates an High Availability configuration

URL:

Bash

[Copy](#)

```
curl http://<cloudSwXtch-control-IP>/swxtch/ha/v1/create
```

Example URL:

<http://10.1.1.6/swxtch/ha/v1/create>

Example Request Body:

Bash

[Copy](#)

```
{
  "uid": "0",
  "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"
      ]
    }
  ]
}
```

Responses:

200 - successful operation

Example Value:

Bash

[Copy](#)

```
{
  "joinClusterResultItems": null,
  "error": null
}
```

Get HA List

GET

swxtch/ha/v1/show

- Returns a list of cloudSwXtches connected in an HA configuration.

URL:

Bash

Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/ha/v1/show
```

Example URL:

http://10.1.1.6/swxtch/ha/v1/show

Example Request Body:

None

Responses:

200 - successful operation

Example Value:

Bash

Copy

```
{
  "clusterConfig": {
    "uid": "706FE124-7F3B-352A-30CE-39909673BDDC",
    "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"
        ]
      }
    ]
  },
  "memberData": {
    "10.1.1.6": {
      "ipAddr": "10.1.1.6",
      "isAlive": true,
      "host": "dsd-core-300"
    }
  }
}
```



```

    },
    "10.2.128.93": {
      "ipAddr": "10.2.128.93",
      "isAlive": true,
      "host": "dsd-core-101"
    },
    "10.5.1.6": {
      "ipAddr": "10.5.1.6",
      "isAlive": true,
      "host": "dsd-core-200"
    }
  },
  "xNICs": {
    "agent-102": {
      "name": "agent-102",
      "ctrlIpAddress": "10.2.192.11",
      "connectedTo": [
        "10.2.128.93"
      ],
      "mode": "Normal"
    },
    "agent-104": {
      "name": "agent-104",
      "ctrlIpAddress": "10.2.192.82",
      "connectedTo": [
        "10.2.128.93"
      ],
      "mode": "Normal"
    },
    "agent-105": {
      "name": "agent-105",
      "ctrlIpAddress": "10.2.192.15",
      "connectedTo": [
        "10.2.128.93"
      ],
      "mode": "Normal"
    },
    "agent-201": {
      "name": "agent-201",
      "ctrlIpAddress": "10.5.2.4",
      "connectedTo": [
        "10.5.1.6"
      ],
      "mode": "Normal"
    },
    "agent-202": {
      "name": "agent-202",
      "ctrlIpAddress": "10.5.2.5",
      "connectedTo": [
        "10.5.1.6"
      ],
      "mode": "Normal"
    },
    "agent-204": {
      "name": "agent-204",

```

```

        "ctrlIpAddress": "10.5.2.7",
        "connectedTo": [
            "10.1.1.6",
            "10.5.1.6"
        ],
        "mode": "HA"
    },
    "agent-301": {
        "name": "agent-301",
        "ctrlIpAddress": "10.1.2.4",
        "connectedTo": [
            "10.1.1.6"
        ],
        "mode": "Normal"
    },
    "agent-302": {
        "name": "agent-302",
        "ctrlIpAddress": "10.1.2.5",
        "connectedTo": [
            "10.1.1.6"
        ],
        "mode": "Normal"
    },
    "agent-304": {
        "name": "agent-304",
        "ctrlIpAddress": "10.1.1.7",
        "connectedTo": [
            "10.1.1.6",
            "10.5.1.6"
        ],
        "mode": "Normal"
    }
}

```

Delete cloudSwXtch from HA

POST

swxtch/ha/v1/removeSwitch

- Removes a cloudSwXtch from a HA configuration

URL:

Bash

[Copy](#)

```
curl http://<cloudSwXtch-control-IP>/swxtch/ha/v1/removeSwitch
```

Example URL:

http://10.1.1.6/swxtch/ha/v1/removeSwitch

Example Request Body:

Bash

[Copy](#)

```
{
  "switch": "10.5.1.6"
}
```

Responses:

200 - successful operation

Example Value:

Bash

[Copy](#)

```
{
  "Message": "Swxtch 10.5.1.6 removed sucessfully from HA"
}
```

Destroy HA

POST

swxtch/ha/v1/destroy

- Destroys a cloudSwXtch from the HA configuration

URL:

Bash

Copy

```
curl http://<cloudSwXtch-control-IP>/swxtch/ha/v1/destroy
```

Example URL:

http://10.1.1.6/swxtch/ha/v1/destroy

Example Request Body:

None

Responses:

200 - successful operation

Example Value:

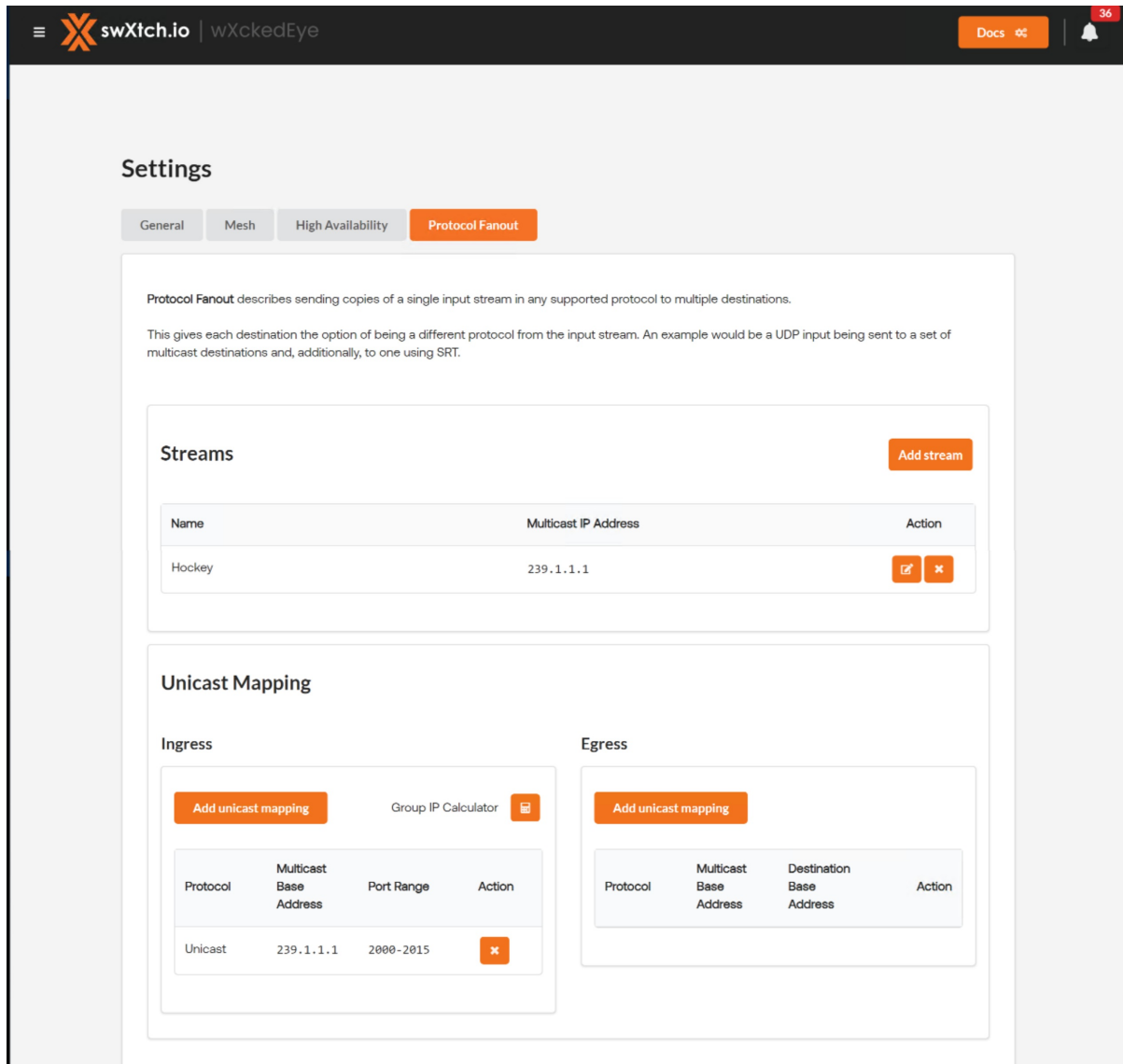
Bash

Copy

```
{
  "Message": "HA destroyed successfully"
}
```

Protocol Fanout

The Protocol Fanout tab shows the cloudSwXtches that are configured for Protocol Fanout: Multicast to Unicast and Unicast to Multicast.



Below is a portion of the Settings call response detailing information on Protocol Fanout. **Note:** In the API, it is called **unicast adaptor** while in the wXcked Eye UI, it is called Protocol Fanout.

```

Bash Copy

"unicastSettings": {
  "unicastToMulticast": {
    "baseAddr": "239.1.1.1",
    "portRange": [
      2000,
      2015
    ],
    "disable": false
  },
  "multicastToUnicast": {
    "adaptors": {
      "4009820932": {
        "targetIp": "239.1.3.4",
        "targetMac": "FF:FF:FF:FF:FF:FF",
        "groupIp": "239.2.1.1"
      }
    }
  }
},

```

```
"streams": {
  "streams": {
    "239.1.1.1": "Hockey"
  }
}
```

Enable Unicast

POST

swxtch/unicast-adaptor/enable

- Enables the port range on the cloudSwXtch to map the unicast to multicast

URL:

Bash

[Copy](#)

```
curl http://<cloudSwXtch-control-IP>/swxtch/unicast-adaptor/enable
```

Example URL:

http://10.1.1.6/swxtch/unicast-adaptor/enable

Example Request Body:

Bash

[Copy](#)

```
{
  "baseAddr": "239.1.1.1",
  "portRange": [
    2000,
    2015
  ],
  "disable": false
}
```

Responses:

200 - successful operation

Example Value:

Bash

[Copy](#)

```
Success: true{
  "Message": "Success"
}
```

Unicast Join

POST

swxtch/unicast-adaptor/join

- Add a machine to a multicast group

URL:

Bash

[Copy](#)

```
curl http://<cloudSwXtch-control-IP>/swxtch/unicast-adaptor/join
```

Example URL:

<http://10.1.1.6/swxtch/unicast-adaptor/join>

Example Request Body:

Bash

[Copy](#)

```
{
  "targetIp": "10.1.1.4",
  "targetMac": "ff:ff:ff:ff:ff:ff",
  "groupIp": "239.1.1.2"
}
```

Responses:

200 - successful operation

Example Value:

Bash

[Copy](#)

```
{
  "Message": "Added 10.1.1.4 to multicast group 239.1.1.2"
}
```

Unicast Leave

POST

swxtch/unicast-adaptor/leave

- Remove a machine from a multicast group

URL:

Bash

[Copy](#)

```
curl http://<cloudSwXtch-control-IP>/swxtch/unicast-adaptor/leave
```

Example URL:

http://10.1.1.6/swxtch/unicast-adaptor/leave

Example Request Body:

Bash

[Copy](#)

```
{
  "targetIp": "10.1.1.4",
  "targetMac": "ff:ff:ff:ff:ff:ff",
  "groupIp": "239.1.1.2"
}
```

Responses:

200 - successful operation

Example Value:

Bash

[Copy](#)

```
{
  "Message": "Removed 10.1.1.4 from multicast group 239.1.1.2"
}
```

Unicast Disable

POST

swxtch/unicast-adaptor/disable

- Disables the port range on the cloudSwXtch from mapping the unicast to multicast

URL:

Bash

[Copy](#)

```
curl http://<cloudSwXtch-control-IP>/swxtch/unicast-adaptor/disable
```

Example URL:

<http://10.1.1.6/swxtch/unicast-adaptor/disable>

Example Request Body:

None

Responses:

200 - successful operation

Example Value:

Bash

[Copy](#)

```
Success: true{
  "Message": "Success"
}
```