

Netris docs Documentation

Release Netris v3.0

© 2021, Netris

January 24, 2022

List of Tables	v
1 Introduction to Netris	3
2 Netris Architecture	5
2.1 Netris Controller	5
2.2 Netris Switch Agent	6
2.3 Netris SoftGate	6
3 Reference Network Architectures	9
3.1 Unmanaged Switch & Netris SoftGate.	9
3.2 Unmanaged Switch & SoftGate (HA).	10
3.3 Netris Managed Switch & SoftGate (HA)	10
3.4 Netris Managed Switch & SoftGate scalable data center (HA)	11
4 Controller Installation	13
4.1 Quickstart Installation	13
4.2 Virtual Machine Installation	14
4.3 Helm Chart Installation	17
5 Switch Agent Installation	31
5.1 Prerequisite Steps	31
5.2 Install the Netris Agent.	35
6 SoftGate Agent Installation	37
6.1 Minimum Hardware Requirements	37
6.2 BIOS Configuration.	37
6.3 Install the Netris Agent.	37
7 Definitions	39
8 IP Address Management	41
8.1 Allocations and Subnets	41
8.2 Add an Allocation	41
8.3 Add a Subnet	41
9 Inventory	43
9.1 Inventory Profiles.	43
9.2 Adding Switches	44
9.3 Adding SoftGates	45

9.4	Viewing Inventory	47
10	Topology Manager	49
10.1	Adding Links	49
10.2	Hairpin Links (Nvidia Cumulus only)	50
11	Switch Ports	53
12	Basic BGP	55
12.1	Adding BGP Peers	55
13	Advanced BGP	57
13.1	BGP Objects	57
13.2	BGP route-maps	59
14	Static Routing	61
15	NAT	63
15.1	Enabling NAT	63
15.2	Defining NAT rules	64
16	SiteMesh	67
17	Looking Glass	69
18	V-Net	73
18.1	V-Net Fields	73
19	L3 Load Balancer (Anycast LB)	77
19.1	Creating an L3 Load Balancer	77
20	L4 Load Balancer (L4LB)	79
20.1	Enabling L4LB service	79
20.2	Consuming L4LB service	80
21	Access Control Lists (ACL)	81
21.1	ACL Default Policy	81
21.2	ACL Rules	82
21.3	ACL Approval Workflow	84
21.4	ACL Processing Order	85
22	ROH (Routing on the Host)	87
22.1	Adding ROH Hosts	87
23	Kubernetes Integration	89
23.1	Install Netris Operator	89
23.2	Using Type 'LoadBalancer'	90
23.3	Using Netris Custom Resources	91
23.4	Calico CNI Integration	94
24	Visibility (Telescope)	97
24.1	Graph Boards	97
24.2	API Logs	99
24.3	Dashboard	99

25 Accounts	101
25.1 Users	101
25.2 Tenants	102
25.3 Permission Groups	103
25.4 User Roles	103
26 Release notes	105
27 SoftGate Performance	107

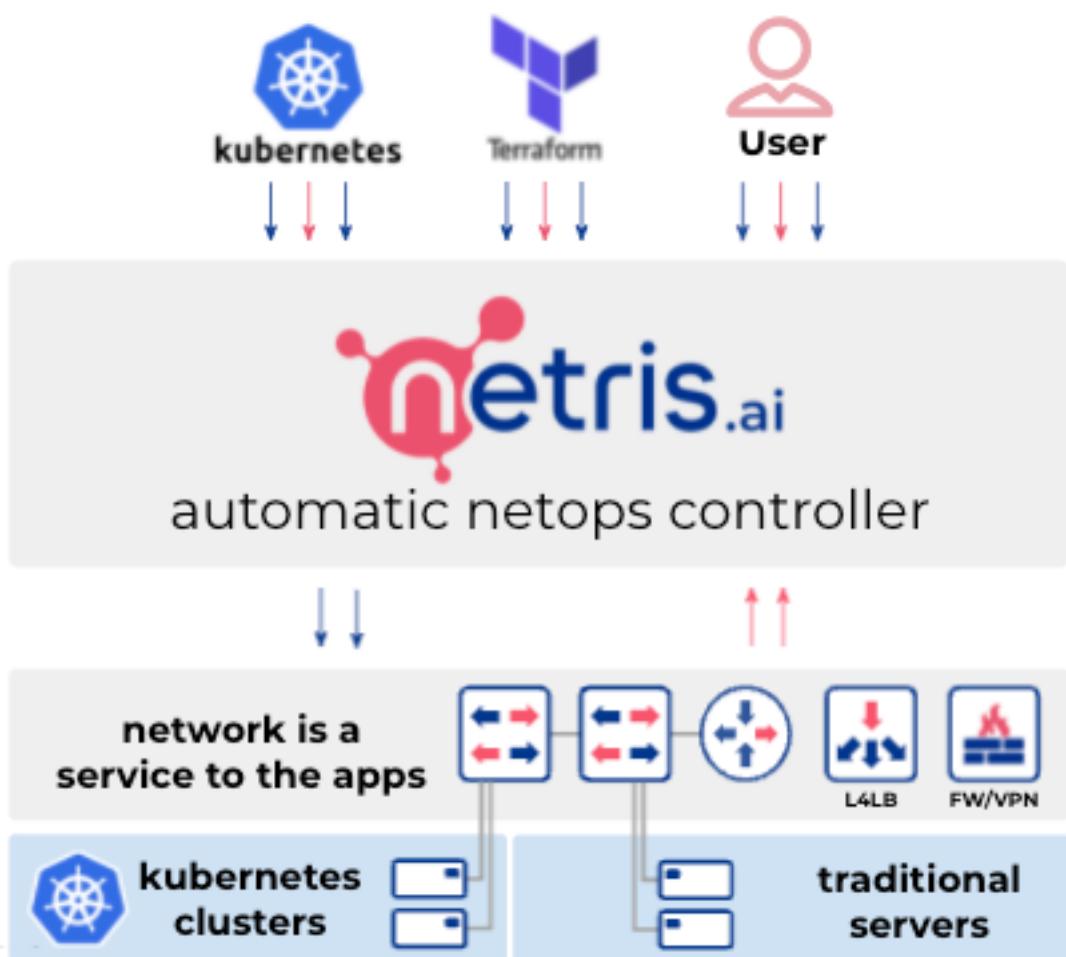
List of Tables

Table 8.1: Allocation Fields	41
Table 8.2: Subnet fields	42
Table 9.1: Inventory Profile Fields	43
Table 9.2: Add Inventory Fields - Switch	44
Table 9.3: Add Inventory Fields - SoftGate	46
Table 12.1: BGP Peer Fields.	55
Table 13.1: BGP Peer Fields - Advanced	57
Table 27.1: SoftGate Performance	107

Netris is the Automatic NetOps platform that runs the physical network and provides cloud-like user experience for NetOps and DevOps engineers.

Introduction to Netris

Netris is an automatic netops software for operating physical networks like it is a cloud. Netris automatically configures switching, routing, load-balancing, and network security based on user-defined services and policies. Netris continuously monitors the network's health and either applies software remediation or informs you of necessary actions if human intervention is required. Netris abstracts away the complexities of detailed network configuration, letting you perform efficiently by operating your physical network in a top down approach like a cloud – instead of the legacy box by box operation.



Netris Architecture

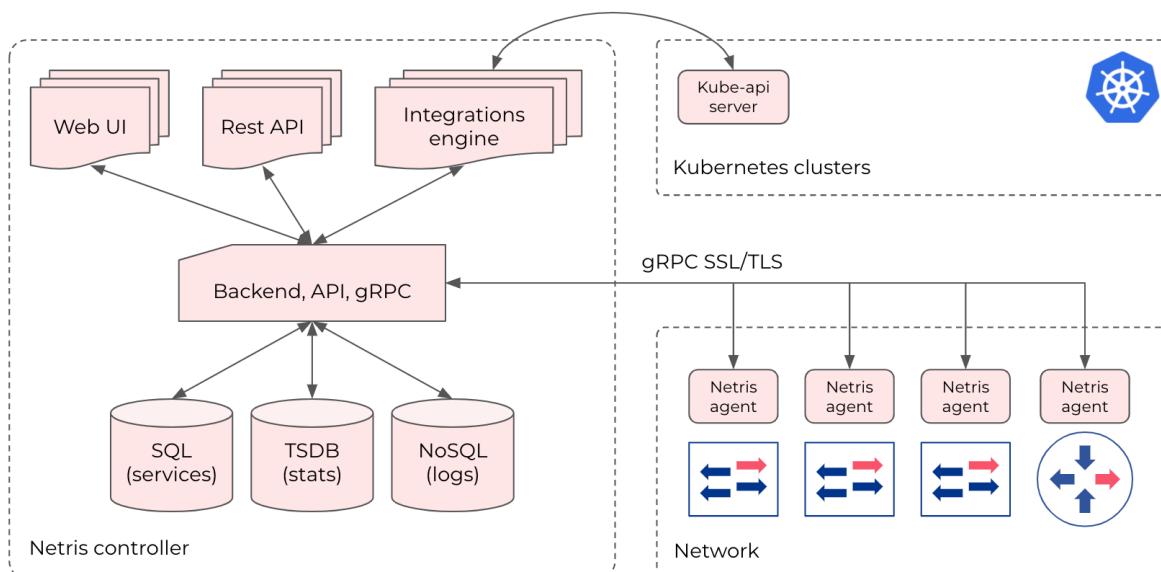
A Netris system is composed of 4 elements:

- Netris Controller
- Netris Switch Agent
- Netris SoftGate
- Customer Application Servers

2.1 Netris Controller

Netris Controller is the main operations control center for engineers using GUI/RestAPI/Kubernetes, systems, and network devices. The Netris Controller stores the data representing the user-defined network services and policies, health, statistics, analytics received from the network devices, and information from integration modules with external systems (Kubernetes). Netris Controller can run as a VM or container, on/off-prem, or in Netris cloud.

Diagram: High level Netris architecture



- **Controller HA** We highly recommend running more than one copy of the controller for database replication.
- **Multiple sites** Netris is designed to operate multiple sites with just a single controller with HA.

- **What if the controller is unreachable.** Netris operated switches/routers can tolerate the unreachability of the Netris Controller. Changes and stats collection will be unavailable during the controller unavailability window; however, switches/routers core operations will not be affected.

2.2 Netris Switch Agent

Netris Switch Agent is software running in the user space of the network operating system (NOS) of the switch and is responsible for automatically generating the particular switch configuration according to service requirements and policies defined in the Netris Controller. Netris Switch Agent uses an encrypted GRPC protocol for secure communication with the Netris Controller accessible through a local management network or over the Internet.

2.3 Netris SoftGate

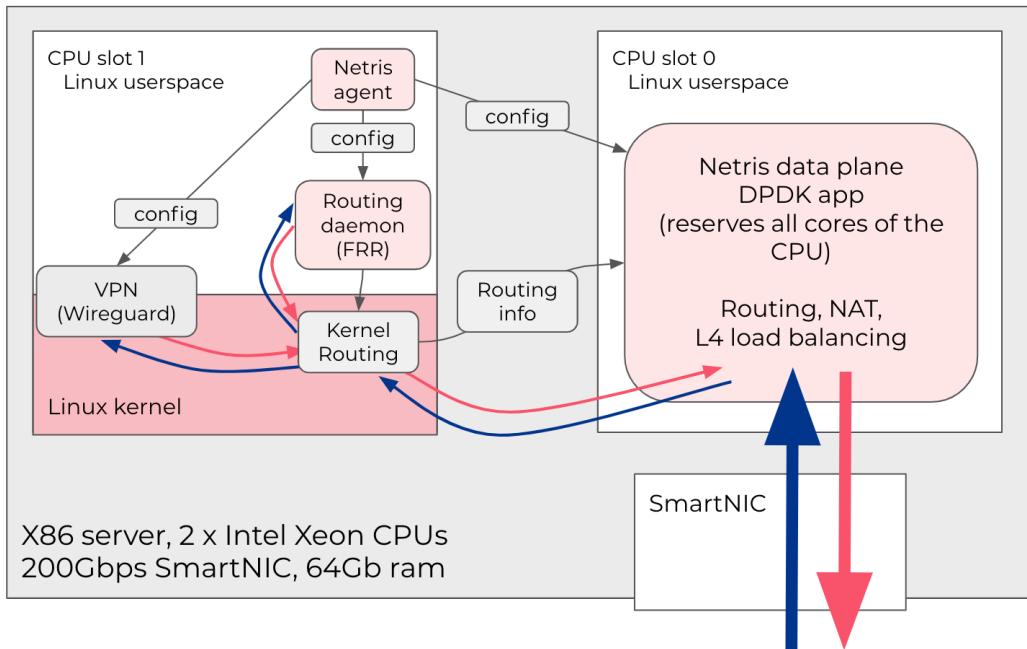
Netris SoftGate is automatic configuration software and reference architecture for enabling border routing, Layer-4 Load Balancing, Network Address Translation (NAT), and site-to-site VPN function on a regular x86 server with a SmartNIC card.

Netris SoftGate supports a high-performance DPDK data plane running in the user-space. It configures the system so that packets entering the NIC (network interface card) bypass Linux Kernel and go directly to the user space application. So traffic from the NIC travels through the PCIe bus to the closest CPU's last level cache and then into one of 8 cores, all reserved for the data-plane application. DPDK data-plane software processes the traffic for routing, load-balancing, NAT and makes necessary changes in the packet header (rewrites mac/VLAN-id) then returns the packet to the NIC, which sends it further into the switch for traveling further in Layer-2.

The server has to have 2 x Intel CPUs (8+ cores each). One CPU (closest to the SmartNIC card) is reserved for the data-plane process only (OS will report 100% CPU usage). Another CPU is used for running Linux OS, routing control plane (FRR), Netris agent, and other standard Linux utilities.

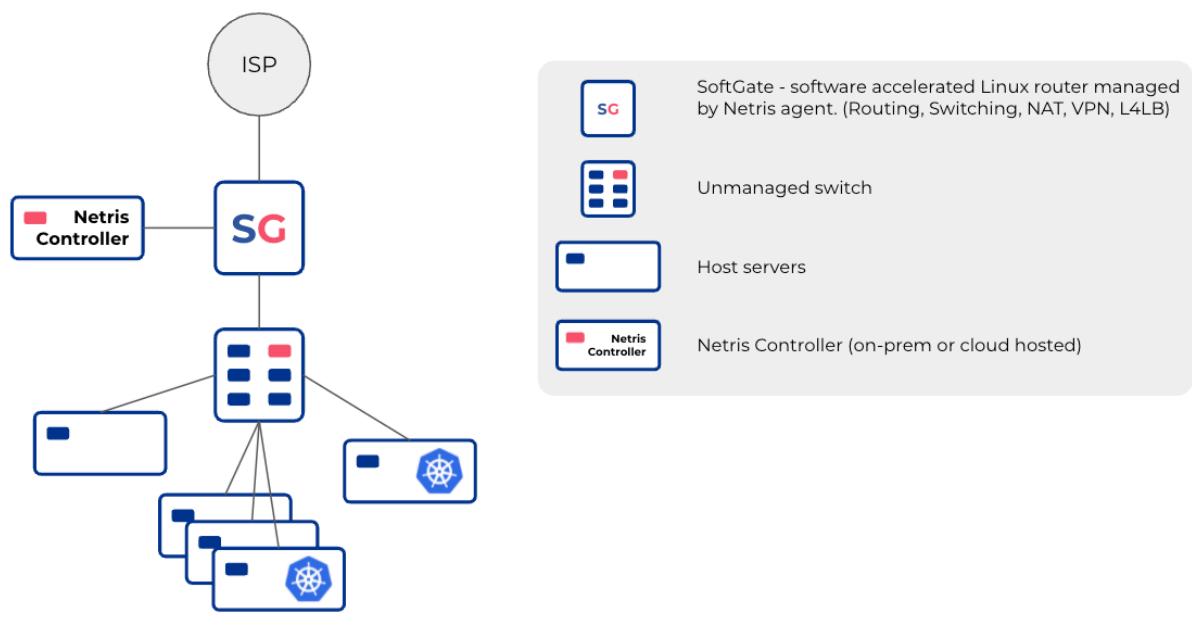
Netris agents can also configure Wireguard to form full mesh VPN tunnels between customer sites and then run necessary dynamic routing. So, servers and applications in multiple data centers can communicate over the Internet using encrypted tunnels.

Diagram: Netris SoftGate high level architecture

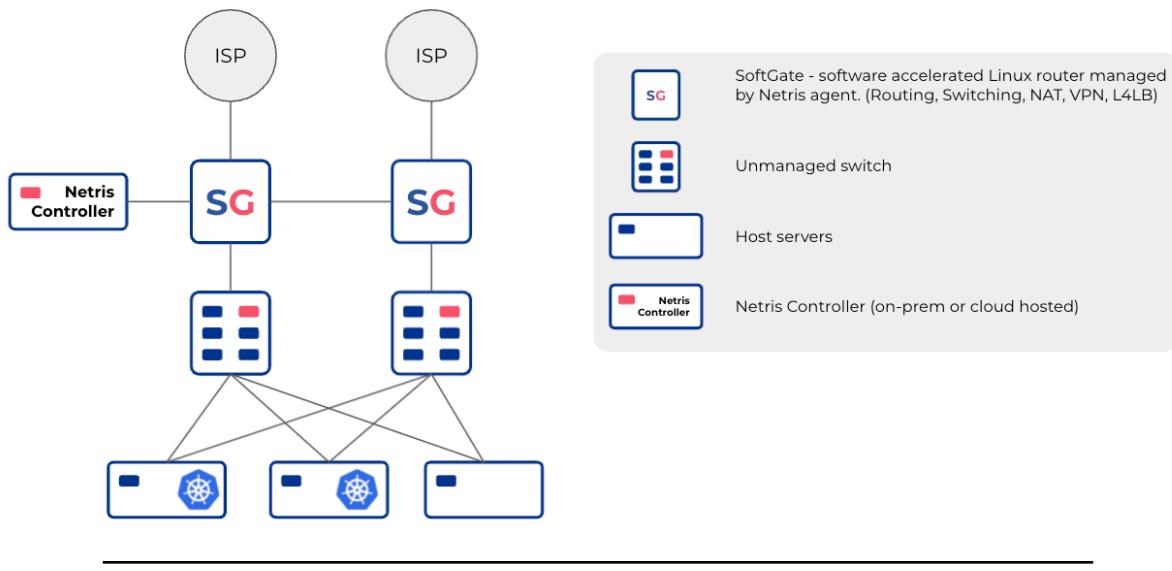


Reference Network Architectures

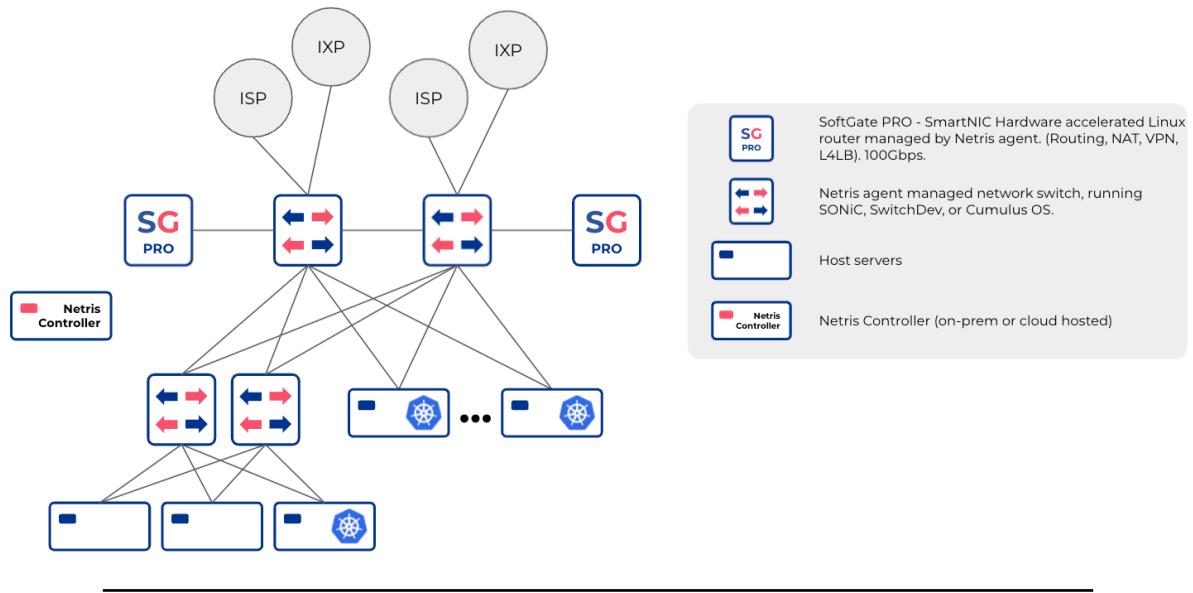
3.1 Unmanaged Switch & Netris SoftGate



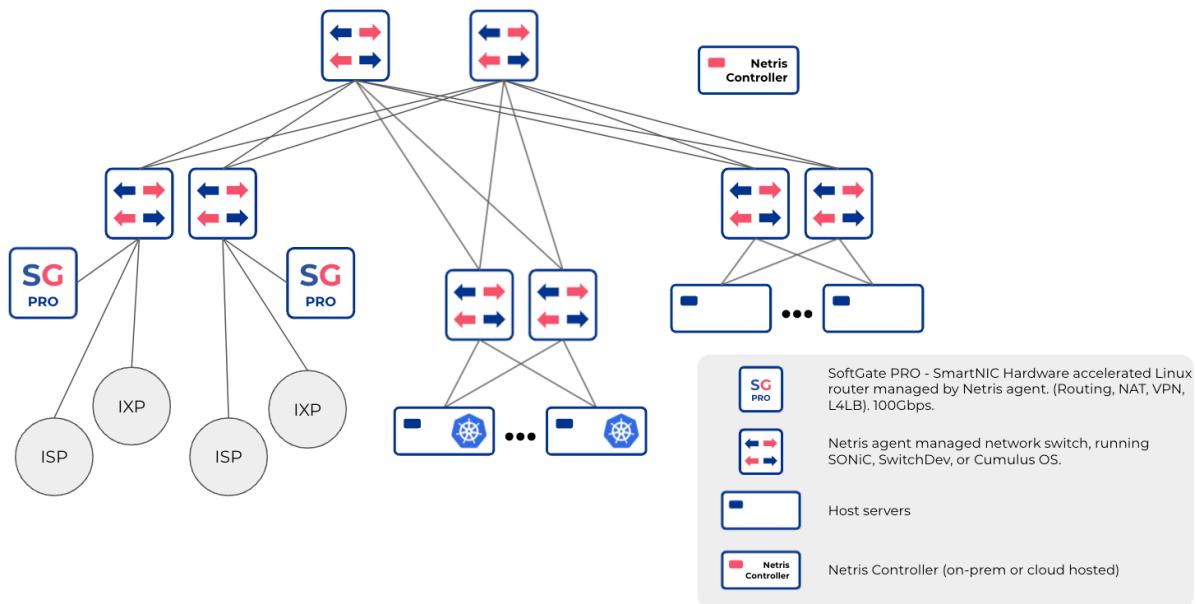
3.2 Unmanaged Switch & SoftGate (HA)



3.3 Netris Managed Switch & SoftGate (HA)



3.4 Netris Managed Switch & SoftGate scalable data center (HA)



Controller Installation

Netris Controller can be installed locally as a VM, deployed as a Kubernetes application, or hosted in the Netris Cloud. All three options provide the same functionality. Cloud-hosted Controller can be moved into on-prem anytime.

Note Select ONE installation option below.

4.1 Quickstart Installation

Netris offers a simplified deployment model for users who want to quickly install the Netris Controller in the shortest amount of time.

This installation process is streamlined for Linux servers that do not already have Kubernetes running. The install does the following:

- Installs k3s
- Installs the [Netris Controller Helm chart](#)

If you wish to install the controller on an existing Kubernetes cluster, follow [these instructions](#) instead of this Quickstart.

4.1.1 Quickstart Process

1. Install the Netris Controller w/ k3s by running the following command on your Linux server:

```
curl -sfL https://get.netris.ai | sh -
```

2. When the installation completes, you will be provided with the login for the web UI. Login with the provided credentials.
3. Navigate in the UI to Net→IPAM and add a new subnet that contains the desired management IP addresses you wish to use for your SoftGates and switches.
For example, if you are planning on using 192.168.1.100 as the IP address of your Ubuntu server, then create a subnet in Netris UI for 192.168.1.0/24.
Detailed configuration documentation is available here: [Netris IPAM](#).
4. Navigate in the UI to **Topology**
5. Click the **Add** in the upper right
6. Fill out the fields for the SoftGate you wish to add
7. Select the proper **Management IP address** from the subnet selector
8. Once the SoftGate is created in the Topology, **right-click** on the SoftGate and select the **Install**

Agent option

9. Copy the agent install command to your clipboard and run the command on the Ubuntu server you are using as your SoftGate
10. Congratulations. The SoftGate should now be connected to your controller.

4.2 Virtual Machine Installation

4.2.1 Requirements

Minimal system requirements for the VM:

- CPU - 4 Core
- RAM - 4 Gb
- Disk - 100Gb
- Network - 1 virtual NIC

Recommended system requirements for the VM:

- CPU - 8 Core
- RAM - 16 Gb
- Disk - 100Gb
- Network - 1 virtual NIC

4.2.2 KVM Hypervisor Installation

If KVM is not already installed, install Qemu/KVM on the host machine (example provided for Ubuntu Linux 18.04)

```
sudo apt-get install virt-manager
```

4.2.3 VM Controller Installation

1. Download the Netris Controller image. (contact Netris support for repository access permissions).

```
cd /var/lib/libvirt/images  
sudo wget http://img.netris.ai/netris-controller3.qcow2
```

2. Download VM definition file.

```
cd /etc/libvirt/qemu  
sudo wget http://img.netris.ai/netris-controller3.xml
```

3. Define the KVM virtual machine

```
sudo virsh define netris-controller3.xml
```

Note Netris Controller virtual NIC will bind to the “br-mgmt” interface on the KVM host machine.

See below for the network interface configuration example.

Example: Network configuration on host (hypervisor) machine.

Note replace <Physical NIC>, <host server management IP/prefix length> and <host server default gateway> with the correct NIC and IP for your host machine.

```
sudo vim /etc/network/interfaces
```

```
#Physical NIC connected to the management network
auto <Physical NIC>
iface <Physical NIC> inet static
    address 0.0.0.0/0

#bridge interface
auto br-mgmt
iface br-mgmt inet static
    address <host server management IP/prefix length>
    gateway <host server default gateway>
    bridge-ports <Physical NIC>

source /etc/network/interfaces.d/*
```

```
sudo ifreload -a
```

4. Set the virtual machine to autostart and start it.

```
sudo virsh autostart netris-controller
```

```
sudo virsh start netris-controller
```

4.2.4 Accessing the Netris Controller

By default, Netris Controller will obtain an IP address from a **DHCP** server.

Below steps describe how to configure a **Static IP** address for the Netris Controller.

1. Connecting to the VM console.

default credentials. **login:** netris **password:** newNet0ps

```
sudo virsh console netris-controller
```

Note Do not forget to change the default password (using passwd command).

2. Setting a static IP address.

Edit network configuration file.

```
sudo vim /etc/network/interfaces
```

Example: IP configuration file.

```
# The loopback network interface
auto lo
iface lo inet loopback
```

```
# The primary network interface
auto eth0
iface eth0 inet static
    address <Netris Controller IP/prefix length>
    gateway <Netris Controller default gateway>
    dns-nameserver <a DNS server address>

source /etc/network/interfaces.d/*
```

Reload the network config.

```
sudo ifreload -a
```

Note Make sure Netris Controller has Internet access.

3. Reboot the controller

```
sudo reboot
```

After reboot, the Netris Controller GUI should be accessible using a browser. Use netris/newNet0ps credentials.



4.2.5 Replacing the SSL certificate

1. Replace the below file with your SSL certificate file.

```
/etc/nginx/ssl/controller.cert.pem;
```

2. Replace the below file with your SSL private key.

```
/etc/nginx/ssl/controller.key.pem;
```

3. Restart Nginx service.

```
systemctl restart nginx.service
```

4.3 Helm Chart Installation

- Installs the automatic NetOps platform Netris Controller

4.3.1 Requirements

- Kubernetes 1.12+
- Helm 3.1+
- PV provisioner support in the underlying infrastructure

4.3.2 Get Repo Info

Add the Netris Helm repository:

```
helm repo add netrisai https://netrisai.github.io/charts  
helm repo update
```

4.3.3 Helm Chart Installation

To install the Helm chart, follow these steps:

1. Create the namespace for netris-controller:

```
kubectl create namespace netris-controller
```

2. Generate strong auth key:

```
export mystrongauthkey=$(date |base64 | md5sum | base64 | head -c 32)
```

3. Install helm chart with netris-controller:

```
helm install netris-controller netrisai/netris-controller \  
--namespace netris-controller \  
--set app.ingress.hosts={my.domain.com} \  
--set netris.authKey=$mystrongauthkey
```

4.3.4 Uninstalling the Helm Chart

To uninstall/delete the netris-controller helm release:

```
helm uninstall netris-controller
```

4.3.5 Chart Default Configuration

The following table lists the configurable parameters of the netris-controller chart and their default values.

Note These settings are already enabled, you do not need to do anything.

Common Parameters

Parameter	Description	Default
nameOverride	String to partially override common.names.fullname template with a string (will prepend the release name)	nil
fullnameOverride	String to fully override common.names.fullname template with a string	nil
serviceAccount.create	Create a serviceAccount for the deployment	true
serviceAccount.name	Use the serviceAccount with the specified name	" "
serviceAccount.annotations	Annotations to add to the service account	{ }
podAnnotations	Pod annotations	{ }
podSecurityContext	Pod Security Context	{ }
securityContext	Containers security context	{ }
resources	CPU/memory resource requests/limits	{ }
nodeSelector	Node labels for pod assignment	{ }
tolerations	Node tolerations for pod assignment	[]
affinity	Node affinity for pod assignment	{ }

Netris-Controller Common Parameters

Parameter	Description	Default
netris.webLogin	Netris Controller GUI default login	netris
netris.webPassword	Netris Controller GUI default password	newNet0ps
netris.authKey	Netris Controller agents authentication key	mystrongkey

Netris-Controller App Parameters

Parameter	Description	Default
app.replicaCount	Number of replicas in app deployment	1
app.image.repository	Image repository	netrisai/xcaas-xcaas-web
app.image.tag	Image tag. Overrides the image tag whose default is the chart appVersion	" "
app.image.pullPolicy	Image pull policy	IfNotPresent

app.imagePullSecrets	Reference to one or more secrets to be used when pulling images	[]
app.service.type	Kubernetes service type	ClusterIP
app.service.port	Kubernetes port where service is expose	80
app.service.portName	Name of the port on the service	http
app.ingress.enabled	Enables Ingress	true
app.ingress.annotations	Ingress annotations (values are templated)	kubernetes.io/ingress.class: nginx }
app.ingress.labels	Custom labels	{ }
app.ingress.path	Ingress accepted path	/
app.ingress.pathType	Ingress type of path	Prefix
app.ingress.hosts	Ingress accepted hostnames	["chart-example.local"]
app.ingress.tls	Ingress TLS configuration	[]

app.autoscaling.enabled	Option to turn autoscaling on for app and specify params for HPA. Autoscaling needs metrics-server to access cpu metrics	false
app.autoscaling.minReplicas	Default min replicas for autoscaling	1
app.autoscaling.maxReplicas	Default max replicas for autoscaling	100
app.autoscaling.targetCPUUtilizationPercent	The desired target CPU utilization for autoscaling	80

Netris-Controller gRPC Parameters

Parameter	Description	Default
grpc.replicaCount	Number of replicas in grpc deployment	1
grpc.image.repository	Image repository	netrisai/xcaas-agent-api-server

grpc.image.tag	Image tag. Overrides the image tag whose default is the chart appVersion	" "
grpc.image.pullPolicy	Image pull policy	IfNotPresent
grpc.imagePullSecrets	Reference to one or more secrets to be used when pulling images	[]
grpc.service.type	Kubernetes service type	ClusterIP
grpc.service.port	Kubernetes port where service is expose	443
grpc.service.portName	Name of the port on the service	grpc

<code>grpc.autoscaling.enabled</code>	Option to turn autoscaling on for app and specify params for HPA. Autoscaling needs metrics-server to access cpu metrics	false
<code>grpc.autoscaling.minReplicas</code>	Default min replicas 1 for autoscaling	
<code>grpc.autoscaling.maxReplicas</code>	Default max replicas 100 for autoscaling	
<code>grpc.autoscaling.targetCPUUtilizationPercentage</code>	The desired target CPU utilization for autoscaling	80

Netris-Controller Telescope Parameters

Parameter	Description	Default
<code>telescope.replicaCount</code>	Number of replicas in telescope deployment	1
<code>telescope.image.repository</code>	Image repository	<code>netrisai/xcaas-telescope-go</code>

telescope.image.tag	Image tag. Overrides the image tag whose default is the chart appVersion	" "
telescope.image.pullPolicy	Image pull policy	IfNotPresent
telescope.imagePullSecrets	Reference to one or more secrets to be used when pulling images	[]
telescope.service.type	Kubernetes service type	ClusterIP
telescope.service.port	Kubernetes port where service is exposed	80
telescope.service.portName	Name of the port on the service	ws
telescope.service.securePort	Kubernetes secure port where service is exposed	443
telescope.service.securePortName	Name of the secure port on the service	wss

<code>telescope.autoscaling.enabled</code>	Option to turn autoscaling on for app and specify params for HPA. Autoscaling needs metrics-server to access cpu metrics.
<code>telescope.autoscaling.minReplicas</code>	Default min replicas 1 for autoscaling
<code>telescope.autoscaling.maxReplicas</code>	Default max replicas 100 for autoscaling
<code>telescope.autoscaling.targetCPUUtilizationPercentage</code>	The desired target CPU utilization for autoscaling

Netris-Controller k8s-watcher Parameters

Parameter	Description	Default
<code>k8s-watcher.replicaCount</code>	Number of replicas in k8s-watcher deployment	1
<code>k8s-watcher.image.repository</code>	Image repository	<code>netrisai/xcaas-kuberis-k8s-api-agent</code>

k8s-watcher.image.tag	Image tag. Overrides the image tag "" whose default is the chart appVersion
k8s-watcher.image.pullPolicy	Image pullIfNotPresent policy
k8s-watcher.imagePullSecrets	Reference to one or more secrets [] to be used when pulling images
k8s-watcher.autoscaling.enabled	Option to turn autoscaling on for app and specify params for HPA. Autoscaling needs metrics-server to access cpu metrics
k8s-watcher.autoscaling.minReplicas	Default min replicas for autoscaling
k8s-watcher.autoscaling.maxReplicas	Default max replicas for autoscaling

k8s-watcher.autoscaling.targetCPUUtilizationPercentage	The desired target CPU utilization for autoscaling
--	--

Netris-Controller telescope-notifier Parameters

Parameter	Description	Default
telescope-notifier.replicaCount	Number of replicas in telescope-notifier deployment	1
telescope-notifier.image.repository	Image repository	netrisai/xcaas-xcaas-notifier
telescope-notifier.image.tag	Image tag. Overrides the image tag whose default is the chart appVersion	" "
telescope-notifier.image.pullPolicy	Image pull policy	IfNotPresent
telescope-notifier.imagePullSecrets	Reference to one or more secrets to be used when pulling images	[]

telescope-notifier.autoscaling.enabled	Option to turn autoscaling on for app and specify params for HPA. Autoscaling needs metrics-server to access cpu metrics
telescope-notifier.autoscaling.minReplicas	Default min replicas for autoscaling
telescope-notifier.autoscaling.maxReplicas	Default max replicas for autoscaling
telescope-notifier.autoscaling.targetCPUUtilizationPercentage	The desired target CPU utilization for autoscaling

MariaDB Parameters

Using default values from

Parameter	Description	Default
mariadb.image.repository	MariaDB image name. We extended bitnami's mariadb image with own plugin	netrisai/netris-mariadb
mariadb.image.tag	MariaDB image tag. (only 10.1 is supported)	10.1
mariadb.initdbScriptsConfigMap	ConfigMap with the initdb scripts.	netris-controller-initdb
mariadb.auth.database	Name for a database to create	netris
mariadb.auth.username	Name for a user to create	netris

mariadb.auth.password	Password for the new user	changeme
mariadb.auth.rootPassword	Password for the root user	changeme

Auth from existing secret not supported at the moment

MongoDB Parameters

Using default values from

Parameter	Description	Default
mongodb.useStatefulSet	Use StatefulSet instead of Deployment when deploying standalone	true
mongodb.initdbScriptsConfigMap	ConfigMap with the initdb scripts.	netris-controller-initdb-mongodb
mongodb.auth.database	Name for a database to create	netris
mongodb.auth.username	Name for a user to create	netris
mongodb.auth.password	Password for the new user	changeme
mongodb.auth.rootPassword	Password for the root user	changeme

Auth from existing secret not supported at the moment

Redis Parameters

Using default values from

Parameter	Description	Default
redis.cluster.enabled	Use master-slave topology	false
redis.usePassword	Use password	false

Auth not supported at the moment

SMTP Parameters

Using default values from

Parameter	Description	Default
smtp.config.DISABLE_IPV6	Disable IPv6	1

smtp.config.RELAY_NETWORKS	Rely networks. Change if your CNI use other subnets	:172.16.0.0/12:10.0.0.0/8:192.168.0.0/16
----------------------------	--	--

Haproxy Parameters

Using default values from

Parameter	Description	Default
haproxy.enabled	Enable HAProxy. Used for exposing netris agents ports from single loadbalancer ip. Disable if you can't have type:LoadBalancer service in cluster	true
haproxy.service.type	Kubernetes service type	LoadBalancer

Graphite Parameters

Using default values from

Parameter	Description	Default
graphite.configMaps	Netris-Controller supported graphite's config files	see in values.yaml
graphite.service.type	Kubernetes service type	ClusterIP

4.3.6 Usage

Specify each parameter using the `--set key=value[,key=value]` argument to helm install. For example,

```
helm install netris-controller netrisai/netris-controller \
--namespace netris-controller \
--set app.ingress.hosts={my.domain.com} \
--set netris.authKey=$mystrongauthkey \
--set mariadb.auth.rootPassword=my-root-password \
--set mariadb.auth.password=my-password \
--set mongodb.auth.rootPassword=my-root-password \
--set mongodb.auth.password=my-password
```

The above command sets netris-controller application ingress host to `my.domain.com` and sets generated `netris.authKey`. Additionally, it sets MariaDB and MongoDB root account password to `my-root-password` and user account password to `my-password`.

Alternatively, a YAML file that specifies the values for the parameters can be provided while installing the chart. For example,

```
helm install netris-controller netrisai/netris-controller --namespace
netris-controller -f values.yaml
```

After installation use EXTERNAL-IP of haproxy service as `--controller` parameter in `netris-setup`

```
kubectl get svc -nnetris-controller |grep haproxy
```

and `$mystrongauthkey` as `--auth` parameter in `netris-setup`

```
echo $mystrongauthkey
```

Also you can see overrides values from helm get values

```
helm get values netris-controller
```

Switch Agent Installation

5.1 Prerequisite Steps

5.1.1 Nvidia Cumulus Linux Devices

Requirements: * Fresh install of Cumulus Linux v. 3.7.(x) - Cumulus 4.X is in the process of validation and will be supported in the next Netris release.

Configure the OOB Management IP address

Configure internet connectivity via management port.

```
sudo vim /etc/network/interfaces
```

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address <management IP address/prefix length>
    gateway <gateway of management network>
    dns-nameserver <dns server>

source /etc/network/interfaces.d/*
```

```
sudo ifreload -a
```

Configure Nvidia Cumulus Linux License

```
sudo cl-license -i
```

Copy/paste the Cumulus Linux license string then press ctrl-d.

Continue to [Section 5.2](#) section.

5.1.2 Ubuntu SwitchDev Devices

Note Further installation requires a Console and Internet connectivity via management port!

1. NOS Uninstall

Uninstall current NOS using **Uninstall OS** from grub menu:

```
+-----+  
| ONIE: Install OS  
| ONIE: Rescue  
| *ONIE: Uninstall OS  
| ONIE: Update ONIE  
| ONIE: Embed ONIE  
|  
|  
|  
|  
|  
|  
+-----+
```

Once the uninstallation is completed, the switch will reboot automatically.

2. Update ONIE

Select **Update ONIE** from grub menu:

```
+-----+  
| ONIE: Install OS  
| ONIE: Rescue  
| ONIE: Uninstall OS  
| *ONIE: Update ONIE  
| ONIE: Embed ONIE  
|  
|  
|  
|  
|  
|  
+-----+
```

In case you don't have DHCP in the management network, then stop ONIE discovery service and configure IP address and default gateway manually:

```
onie-discovery-stop  
ip addr add <management IP address/prefix> dev eth0  
ip route add default via <gateway of management network>  
echo "nameserver <dns server>" > /etc/resolv.conf
```

Update ONIE to the supported version.

Note ONIE image available for Mellanox switches only!

```
onie-self-update https://repo.netris.ai/repo/onie-updater-x86_64-mlnx_x86-r0
```

3. NOS Install

Select **Install OS** from grub menu:

```
+-----+
| *ONIE: Install OS
| ONIE: Rescue
| ONIE: Uninstall OS
| ONIE: Update ONIE
| ONIE: Embed ONIE
|
|
|
|
|
|
+-----+
```

In case you don't have DHCP in the management network, then stop ONIE discovery service and configure IP address and default gateway manually:

```
onie-discovery-stop
ip addr add <management IP address/prefix> dev eth0
ip route add default via <gateway of management network>
echo "nameserver <dns server>" > /etc/resolv.conf
```

Install Ubuntu-SwitchDev from the Netris custom image:

```
onie-nos-install https://repo.netris.ai/repo/netris-ubuntu-18.04.1.bin
```

Default username/password

netris/newNet0ps

Configure the OOB Management IP address

Configure internet connectivity via management port.

```
sudo vim /etc/network/interfaces
```

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address <management IP address/prefix length>
    gateway <gateway of management network>
    dns-nameserver <dns server>

source /etc/network/interfaces.d/*
```

```
sudo ifreload -a
```

Continue to [Section 5.2](#) section.

5.1.3 EdgeCore SONiC Devices

Note Further installation requires a Console and Internet connectivity via management port!

1. NOS Uninstall

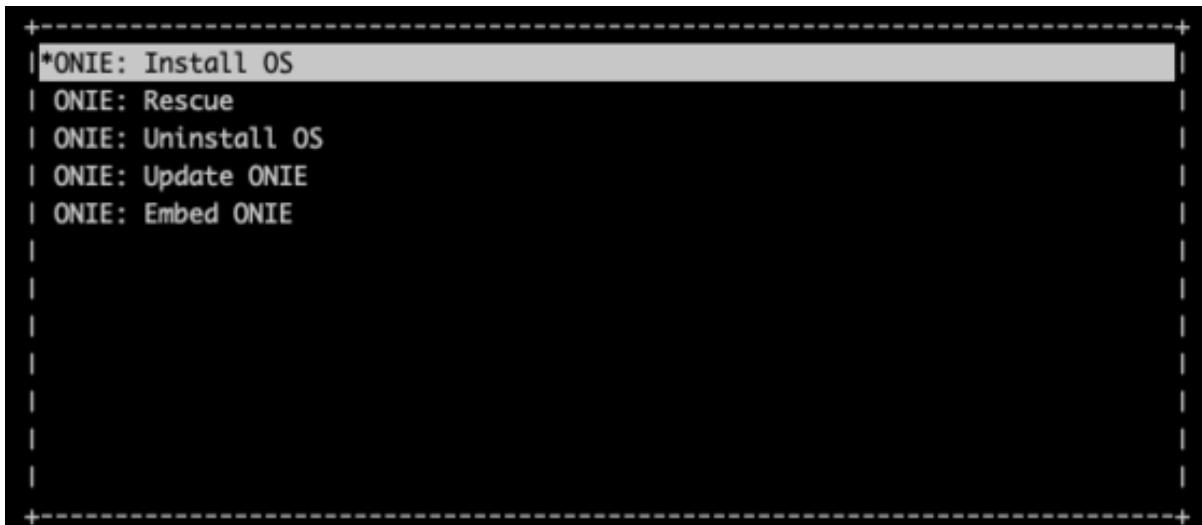
Uninstall current NOS using **Uninstall OS** from grub menu:



Once the uninstallation is completed, the switch will reboot automatically.

2. NOS Install

Select **Install OS** from grub menu:



In case you don't have DHCP in the management network, then stop ONIE discovery service and configure IP address and default gateway manually:

```
onie-discovery-stop
ip addr add <management IP address/prefix> dev eth0
ip route add default via <gateway of management network>
echo "nameserver <dns server>" > /etc/resolv.conf
```

Install EdgeCore SONiC image from the Netris repository:

```
onie-nos-install
https://repo.netris.ai/repo/Edgecore-SONiC_20211125_074752_ec202012_227.bin
```

Default username/password

admin/YourPaSsWoRd

Configure the OOB Management IP address

Disable Zero Touch Provisioning for time being.

```
ztp disable -y
```

Note This will take some time, please be patient.

Configure internet connectivity via management port.

```
ip addr add <management IP address/prefix> dev eth0
ip route add default via <gateway of management network>
echo "nameserver <dns server>" > /etc/resolv.conf
```

Continue to Section 5.2 section.

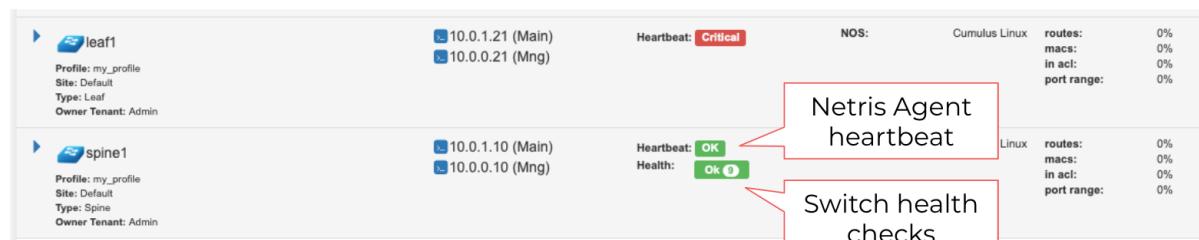
5.2 Install the Netris Agent

1. Add the Switch in the controller **Inventory**. Detailed configuration documentation is available here: [Section 9.2](#)
2. Once the Switch is created in the **Inventory**, click on **three vertical dots (?)** on the right side on the Switch and select the **Install Agent** option
3. Copy the agent install command to your clipboard and run the command on the Switch
4. Reboot the Switch when the installation completes

```
sudo reboot
```

Once the switch boots up you should see its heartbeat going from Critical to OK in Net→Inventory, Telescope→Dashboard, and switch color will reflect its health in Net→Topology

Screenshot: Net→Inventory



SoftGate Agent Installation

6.1 Minimum Hardware Requirements

- 2 x Intel Silver CPU
- 96 GB RAM
- 300 GB HDD
- Nvidia Mellanox Connect-X 5/6 SmartNIC card

6.2 BIOS Configuration

The following are some recommendations for BIOS settings. Different vendors will have different BIOS naming so the following is mainly for reference:

- Before starting consider resetting all BIOS settings to their defaults
- Disable all power saving options such as: Power performance tuning, CPU P-State, CPU C3 Report and CPU C6 Report
- Select Performance as the CPU Power and Performance policy
- Enable Turbo Boost
- Set memory frequency to the highest available number, NOT auto
- Disable all virtualization options when you test the physical function of the NIC, and turn off VT-d
- Disable Hyper-Threading

6.3 Install the Netris Agent

Requires freshly installed Ubuntu Linux 18.04 and internet connectivity configured from netplan via management port

1. Add the SoftGate in the controller **Inventory**. Detailed configuration documentation is available here: [Section 9.3](#)
2. Once the SoftGate is created in the **Inventory**, click on **three vertical dots (?)** on the right side on the SoftGate and select the **Install Agent** option
3. Copy the agent install command to your clipboard and run the command on the SoftGate

4. When the installation completes, review ifupdown configuration file (it was generated based on your netplan configuration)

```
sudo vim /etc/network/interfaces
```

5. If everything seems ok, remove/comment **Gateway** line and save the file.

Note If the Netris Controller is not in the same OOB network then add a route to Netris Controller. No default route or other IP addresses should be configured.

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address <Management IP address/prefix length>
    up ip ro add <Controller address> via <Management network gateway> #delete
this line if Netris Controller is located in the same network with the SoftGate
node.

source /etc/network/interfaces.d/*
```

6. Reboot the SoftGate

```
sudo reboot
```

Once the server boots up you should see its heartbeat going from Critical to OK in Net→Inventory, Telescope→Dashboard, and SoftGate color will reflect its health in Net→Topology

Definitions

When configuring and operating a Netris system, the following nomenclature is important to understand:

- **User** - A user account for accessing Netris Controller through GUI, RestAPI, and Kubernetes. The default username is `netris`, with password `newNet0ps`.
- **Tenant** - IP addresses and Switch Ports are network resources assigned to different Tenants to have under their management. Admin is the default tenant, and by default, it owns all the resources. You can use different Tenants for sharing and delegation of control over the network resources. Network teams typically use Tenants to grant access to other groups to request and manage network services using the Netris Controller as a self-service portal or programmatically (with Kubernetes CRDs) via a DevOps/NetOps pipeline.
- **Permission Group** - List of permissions on a per section basis can be attached individually to a User or a User Role
- **User Role** - Group of user permissions and tenants for role-based access control (RBAC)
- **Site** - Each separate deployment (each data center) should be defined as a *Site*. All network units and resources are attached to a site. Netris Controller comes with a “default” site preconfigured. Site entry defines global attributes such as; AS numbers, default ACL policy, and Site Mesh (site to site VPN) type.
- **Subnet** - IPv4/IPv6 address resources linked to *Sites* and *Tenants*
- **Switch Port** - Physical ports of all switches attached to the system
- **Inventory** - Inventory of all network units that are operated using Netris Agent
- **E-BGP** - Defines all External BGP peers (iBGP and eBGP)

IP Address Management

Netris IPAM allows users to document their IP addresses and track pool usage. It is designed to have a tree-like view to provide opportunity to perform any kind of subnetting.

Purpose: Users define specific roles(purpose) for each subnet/address and only after that are allowed to use those subnets in services like V-net, NAT, etc...

8.1 Allocations and Subnets

There are 2 main types of IP prefixes - allocation and subnet. Allocations are IP ranges allocated to an organization via RIR/LIR or private IP ranges that are going to be used by the network. Subnets are prefixes which are going to be used in services. Subnets are always childs of allocation. Allocations do not have parent subnets.

8.2 Add an Allocation

1. Navigate to Net→IPAM
2. Click the **Add** button
3. Select **Allocation** from the bottom select box
4. Fill in the rest of the fields based on the requirements listed below
5. Click the **Add** button

Table 8.1. Allocation Fields

Name	Unique name for current allocation.
Prefix	Unique prefix for allocation, must not overlap with other allocations.
Tenant	Owner of the allocation.

8.3 Add a Subnet

1. Navigate to Net→IPAM
2. Click the **Add** button

3. Select **Subnet** from the bottom select box
4. Fill in the rest of the fields based on the requirements listed below
5. Click the **Add** button

Table 8.2. Subnet fields

Name	Unique name for current subnet.
Prefix	Unique prefix for subnet, ust be included in one of allocations.
Tenant	Owner of the subnet.
Purpose	<p>This field describes for what kind of services the current subnet can be used. It can have the following values:</p> <ul style="list-style-type: none"> • <i>common</i> - ordinary subnet, can be used in v-nets. • <i>loopback</i> - hosts of this subnet can be used only as loopback IP addresses for Netris hardware (switches and/or softgates). • <i>management</i> - subnet which specifies the out-of-band management IP addresses for Netris hardware (switches and softgates). • <i>load-balancer</i> - hosts of this subnet are used in LB services only. Useful for deploying on-prem kubernetes with cloud-like experience. • <i>nat</i> - hosts of this subnet or subnet itself can be used to define NAT services. • <i>inactive</i> - can't be used in any services, useful for reserving/documenting prefixes for future use.

Inventory

The Inventory section allows you to add/edit/delete network switches and SoftGates. Initial setup of a Netris managed network is a three part process:

1. Create Inventory Profiles
2. Adding Switches
3. Adding Softgates

9.1 Inventory Profiles

Inventory profiles allow security hardening of inventory devices. By default all traffic flow destined to switch/SoftGate is allowed. As soon as the inventory profile is attached to a device it denies all traffic destined to the device except Netris-defined and user-defined custom flows. Generated rules include:

- SSH from user defined subnets
- NTP from user defined ntp services
- DNS from user defined DNS servers
- Custom user defined rules

Table 9.1. Inventory Profile Fields

Name	Profile name
Description	Free text description
Allow SSH from IPv4	List of IPv4 subnets allowed to ssh (one address per line)
Allow SSH from IPv6	List of IPv6 subnets allowed to ssh (one address per line)
Timezone	Devices using this inventory profile will adjust their system time to the selected timezone.
NTP servers	List of domain names or IP addresses of NTP servers (one address per line). You can use your Netris Controller address as an NTP server for your switches and SoftGate.
DNS servers	List of IP addresses of DNS servers (one address per line). You can use your Netris Controller address as a DNS server for your switches and SoftGate.

Example: In this example Netris Controller is used to provide NTP and DNS services to the switches (common setup).

Add new Inventory profile

Name* Spine Switches

Description Description

Allow SSH from IPV4* 10.0.10.0/24

Allow SSH from IPV6

Timezone (GMT-08:00) Pacific Time

NTP servers us.pool.ntp.org

DNS servers 8.8.8.8
8.8.4.4

Custom Rules

+ Add

Source Subnet	Source port	Destination port	Protocol
10.0.20.0/24	1-65535, or empty for any	20-21	TCP

Cancel Add

9.2 Adding Switches

Every switch needs to be added to the Netris Controller inventory. You can add new devices with the following process:

1. Navigate to **Net→Inventory**
2. Click the **Add** button
3. Fill in the fields as described below
4. Click the **Add** button

Table 9.2. Add Inventory Fields - Switch

Name	Name of the device
Owner Tenant	Owner tenant of the device
Description	Description of the device
Type	There are 3 types of devices that users can add/edit - Switch/Soft-gate/Controller. Other types are added automatically when creating services like ROH
NOS	Operating system of the device; applicable for switches only
Site	Site where the devices reside
AS Number	Private AS number of the device; applicable for switches only; recommended to be assigned automatically
Profile	Inventory profile for current device. Profiles are used for security hardening the devices
Main IP address	Main loopback IP address for the device. Can be configured manually or assigned automatically from subnet with <i>loopback</i> purpose defined for current site.

Management IP address	Management IP address for the device. Can be configured manually or assigned automatically from subnet with <i>management</i> purpose defined for current site. This IP address is configured on the out-of-band management interface of the device.
MAC address	MAC address of the device; applicable for switches only
Preliminary port count	Used for definition of topology. When the device registers with the controller the real ports are synced with inventory
Add Link	Provides functionality to define the connections between devices; mandatory for Switch and Softgate physical interconnections

Example: Add a new Switch.

Add New Hardware x

Name*	spine1	
Tenant*	Admin	
Description	Description	
Type*	Switch	
NOS*	Ubuntu SwitchDev	
Site*	Santa Clara	
AS number*	Assign automatically	
Profile	Default	
Main IP address*	10.254.46.0/24	10.254.46.11
Management IP Address*	Assign automatically	
MAC address	Mac address	
Preliminary port count*	<input type="radio"/> 16 <input checked="" type="radio"/> 32 <input type="radio"/> 48 <input type="radio"/> 54 <input type="radio"/> 56 ?	

Links [?](#)

[+ Add](#)

Local port	Remote port
swp1(swp1)@spine1	swp8@SoftGate1 (Adm...)

[Cancel](#) [Add](#)

Note Repeat this process to define all your switches.

9.3 Adding SoftGates

Every SoftGate node needs to be added to the Netris Controller inventory. To add a SoftGate node:

1. Navigate to **Net→Topology**

2. Click Add
3. Fill in the fields as described below
4. Click the Add button

Table 9.3. Add Inventory Fields - SoftGate

Name	Descriptive name
Owner Tenant	Tenant(typically Admin); who administers this node
Description	Free text description
Hardware Type	Select SoftGate
Site	The data center where the current SoftGate node belongs.
Inventory Profile	Profile describing the timezone; DNS; NTP; and Security features
IP Address	IPv4 address for the loopback interface
Management IP address	IPv4 address for the out of band management interface
NFV Node Port	A physical port on a spine switch where the SoftGate node's first SmartNIC port is connected. Typically each spine switch has one SoftGate node connected to it.
+NAT address	Public IP addresses to be used as global IP for SNAT/DNAT. (check Enabling NAT section of Network Policies chapter)
+NAT address pool	Public IP address subnets to be used as rolling global IP addresses for SNAT. (check Enabling NAT section of Network Policies chapter)

Example: Adding a SoftGate Node to Topology.

Add New Hardware x

Name*	<input style="width: 100%; height: 25px; border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;" type="text"/> ?
Tenant*	<input style="width: 100%; height: 25px; border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;" type="text"/> ?
Description	<input style="width: 100%; height: 25px; border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;" type="text"/> ?
Type*	<input style="width: 100%; height: 25px; border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;" type="text"/> ?
Site*	<input style="width: 100%; height: 25px; border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;" type="text"/> ?
Profile	<input style="width: 100%; height: 25px; border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;" type="text"/> ?
Main IP address*	<input style="width: 50%; height: 25px; border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;" type="text"/> ?
Management IP Address*	<input style="width: 50%; height: 25px; border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;" type="text"/> ?
Links ?	
+ Add	
Cancel Add	

9.4 Viewing Inventory

Inventory Listing shows also Heartbeat and monitoring statuses of each device.

Heartbeat - Shows the status of device reachability. Health - Shows number of successful and failed checks on the device.

Name	Description	Address	Status	Details	Usage	Created Date	Modified Date
>  Controller		169.254.255.1 (main)	Health: Ok 3			29/Sep/2021 11:38	29/Sep/2021 11:38
Profile: None Site: Santa Clara Type: Controller Owner: Admin							
>  SoftGate1	SoftGate1	10.254.46.71 (main) 10.254.45.71 (mng)	Heartbeat: OK Health: Critical 1 Ok 5	Netris version: 3.0.0-alpha.0 OS: Ubuntu 18.04.6... Uptime: 11 hours, 54 mi... TimeZone: America/New_York Connected to: swp6@SoftGat...		30/Sep/2021 15:36	30/Sep/2021 15:36
Profile: None Site: Santa Clara Type: Softgate Owner: Admin							
>  SoftGate2		10.254.46.72 (main) 10.254.45.72 (mng)	Heartbeat: OK Health: Critical 1 Ok 5	Netris version: 2.9.0-10696 OS: Ubuntu 18.04.5... Uptime: 13 hours, 54 mi... TimeZone: America/New_York Connected to: swp6@SoftGat...		30/Sep/2021 22:29	30/Sep/2021 22:29
Profile: None Site: Santa Clara Type: Softgate Owner: Admin							

Note You can also add new devices in the Topology view.

Topology Manager

The topology manager is for describing and monitoring the desired network topology. Netris Switch Agents will configure the underlying network devices according to this topology dynamically and start watching against potential failures.

10.1 Adding Links

To define the links in the network:

1. Right-click on the spine switch
2. Click **Create Link**
3. Select the **From Port** and the **To Port**

See the example below:

❖ Create Link ×

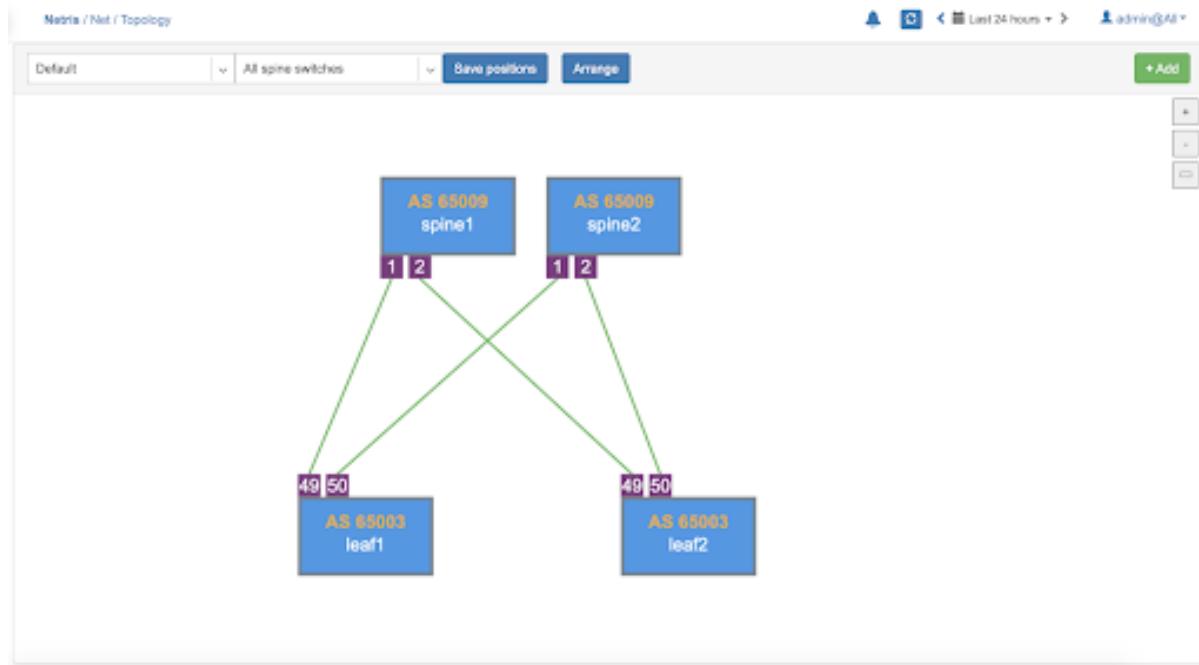
From

Device	spine1
Port	port2(port2)@spine1

To

Device	leaf2
Port	port49(port49)@leaf2

Cancel Create



Once the links have been defined, the network is automatically configured as long as physical connectivity is in place and Netris Agents can communicate with Netris Controller.

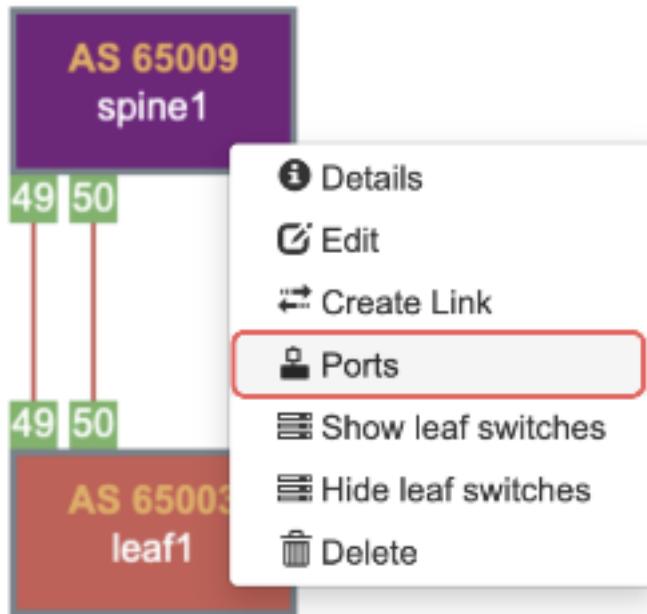
Tip You can drag/move the units to your desired positions and click “Save positions”.

10.2 Hairpin Links (Nvidia Cumulus only)

With Nvidia Cumulus Linux only, we need to loop two ports on spine switches (hairpin cable) in the current release, usually two upstream (higher capacity) ports. We are planning to lift this requirement in the next Netris release (v2.10).

To define what ports will be used as a hairpin, navigate to Net→Switch Ports, or right-click on the spine switch, click Ports in Net→Topology.

Example: Accessing Switch Ports from Net→Topology



For each spine switch, find the two ports that you are going to connect (loop/hairpin) and configure one port as a “hairpin l2” and another port as “hairpin l3”. The order doesn’t matter. The system needs to know which ports you have dedicated for the hairpin/loop on each spine switch. (do not do this for non-Cumulus switches) | Example: Editing Switch Port from Net→Switch Ports.

▶ □ port52	port52	Admin	spine1	standard	Unknown	⋮
▶ □ port53	port53	Admin	spine1	standard	Unknown	⋮
▶ □ port54	port54	Admin	spine1	standard	Unknown	⋮

Example: Setting port types to “hairpin l2” and “hairpin l3”.

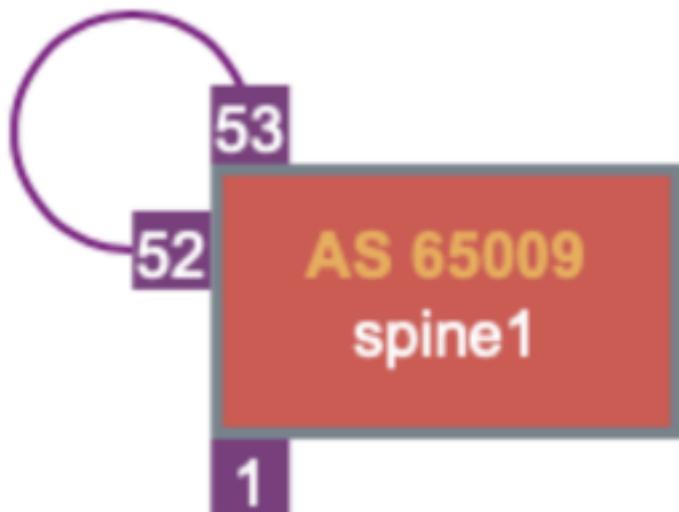
❖ Edit port52

Name	port52
Port type	hairpin l2
MTU	9000
Port Speed	Auto
Extension	None

❖ Edit port53

Name	port53
Port type	hairpin l3
MTU	9000
Port Speed	Auto
Extension	None

Screenshot: Hairpin visualized in Net→Topology



Switch Ports

Switch ports can be directly managed in the **Switch Port** UI section. Both physical and virtual ports (extended, aggregate, etc...) will appear in this section once they have been added to inventory. The Netris Controller will automatically sync the list of available ports that appear on each device.

The following options are available for editing on each port:

- Description - Description of the port.
- Tenant - Tenant to whom the port is assigned, by default it is the owner tenant of the device to whom the port belongs to.
- Breakout - Available only for physical switch ports, used to split physical ports into multiple physical ports. When there is a need to use other supported option supported by switch not shown in the dropdown list user must set breakout to "Manual" and configure breakout manually on the switch. For certain platforms some ports need to be disabled to support breakout into other ports, for that option use "Disable" mode of breakout. For Cumulus, after configuration, user must manually restart switchd daemon on the switch via command "systemctl restart switchd".
- MTU - Maximum transmission unit of the port.
- Autoneg - Toggle autonegotiation. Available only for physical ports.
- Speed - Toggle speed. Available only for physical ports.
- Duplex - Toggle duplex. Available only for physical ports.
- Extension - Create extension ports. Available for physical and aggregate ports.
- Extension Name - Name for new extension.
- VLAN Range - VLAN id range for new extension port.

Edit swp13(swp13)@leaf1 (Admin)

Description*

Tenant* ?

Breakout* ?

MTU* ?

AutoNeg* ?

Speed* ?

Duplex* ?

Extension* **ADD**

Cancel **Save**

Example: Edit physical port

Quick action menu provides following actions for ports (note that Bulk Action also available for multiple ports):

Edit - Edit the port. Admin UP/Down - Toggle admin status of the port. Add to V-net - Add selected port(s) to a V-net.

Add To LAG

Description* **Cancel**

Tenant* ?

MTU* ?

Extension* **ADD**

Member Ports

Cancel **Save**

Free Up Port - Detach port from all resources.

Basic BGP

BGP neighbors can be declared in the Net→E-BGP section. Netris will automatically generate and program the network configuration to meet the requirements.

12.1 Adding BGP Peers

1. Navigate to Net→E-BGP in the web UI
2. Click the **Add** button
3. Fill in the fields as described in the table below
4. Click the **Add** button

Table 12.1. BGP Peer Fields

Name	User assigned name of BGP session
Description	Free description
Site	Selects the site (data center) where this BGP session should be terminated on.
Softgate	Only if SoftGate nodes are in use. Define on which node BGP session should be terminated on.
Neighbor AS	Autonomous System number of the remote side. (Local AS is defined at Net→Sites section)
Terminate on switch	Typically used for setups without SoftGate. For connecting with upstream routers. Instructs the system to terminate the BGP session directly on the switch.
Switch port	Switch Port for the physical cable to the BGP neighbor (any port on the fabric). Optionally can bind to a V-NET service. Typically used for peering with IXPs or systems like GGC (Google Global Cache).
VLAN ID	Optionally tag with a VLAN ID (usually untagged)
IP Version	IPv4 or IPv6
Local IP	BGP peering IP address on Netris controlled side
Remote IP	BGP peering IP address on the remote end
State	Administrative state (Enabled/Disabled)
Advanced	Advanced policy settings are described in the next section

Example: Declare a basic BGP neighbor

Add new E-BGP

Name*	Iris1
Description	Description
Site*	Santa Clara
SoftGate*	sg01
Neighbor AS*	100
Terminate on switch	No
Switch port*	swp16(swp16)@leaf1
VLAN ID*	Untagged
IP Family*	IPv4
Local IP*	198.51.100.2
Remote IP*	198.51.100.1 /30
State*	Enabled

▶ Advanced

Cancel Add

Example2: Declare BGP neighbor terminated on V-Net. Netris will automatically configure BGP session on the switch closest to the remote IP. .. image:: images/add-bgp-basic-2.png

```

align center
class with-shadow
```

Advanced BGP

BGP neighbor declaration can optionally include advanced BGP attributes and BGP route-maps for fine-tuning of BGP policies.

Click Advanced to expand the BGP neighbor add/edit window.

Table 13.1. BGP Peer Fields - Advanced

Neighbor address	IP address of the neighbor when peering with the loopback IP address instead of the interface IP address (aka Multihop).
Update source	When Multihop BGP peering is used it allows the operator to choose one of the loopback IP addresses of the SoftGate node as a BGP speaker source IP address.
BGP password	Password for the BGP session
Allowas-in	Define the number of allowed occurrences of the self AS number in the received BGP NLRI to consider it valid (normally 0).
Default Originate	Originate default route to the current neighbor
Prefix Inbound Max	Drop the BGP session if the number of received prefixes exceeds this max limit. For switch termination maximum allowed is 1000 prefixes and Soft-Gate termination can handle up to one million prefixes.
Inbound Route-Map	Apply BGP policies described in a route-map for inbound BGP updates
Outbound Route-Map	Apply BGP policies described in a route-map for outbound BGP updates
Local Preference	Set local preference for all inbound routes for the current neighbor
Weight	Set weight for all inbound routes for the current neighbor
Prepend Inbound (times)	How many times to prepend self AS number for inbound routes
Prepend Outbound (times)	How many times to prepend self AS number for outbound routes
Prefix List Inbound	List of IP addresses prefixes to permit or deny inbound
Prefix List Outbound	List of IP addresses prefixes to permit or deny outbound
Send BGP Community	List of BGP communities to send to the current neighbor

13.1 BGP Objects

Under Net→E-BGP objects, you can define various BGP objects referenced from a route-map to declare a dynamic BGP policy.

Supported objects include:

- IPv4 Prefix

- IPv6 Prefix
- AS-PATH
- Community
- Extended Community
- Large Community

13.1.1 IPv4 Prefix

Rules defined one per line.

Each line in IPv4 prefix list field consists of three parts:

- Action - Possible values are: permit or deny (mandatory).
- IP Prefix - Any valid IPv4 prefix (mandatory).
- Length - Possible values are: le <len>, ge <len> or ge <len> le <len>.

Example: Creating an IPv4 Prefix list.

Name*	Acme_Prefixes
Type*	IPv4 Prefix
IPv4 Prefix*	permit 198.51.100.0/24 le 32 permit 203.0.113.0/24 le 32

13.1.2 IPv6 Prefix

Rules defined one per line.

Each line in IPv6 prefix list field consists of three parts:

- Action - Possible values are: permit or deny (mandatory).
- IP Prefix - Any valid IPv6 prefix (mandatory).
- Keyword - Possible values are: le <len>, ge <len> or ge <len> le <len>.

Example: Creating an IPv6 Prefix list.

Name*	Acme_IPv6_Prefixes
Type*	IPv6 Prefix
IPv6 Prefix*	permit <u>2001:DB8:1::/48</u> permit <u>2001:DB8:2::/48</u> le 64 deny <u>2001:DB8::/32</u>

13.1.3 Community

Community field has two parts:

- Action - Possible values: permit or deny (mandatory).

- Community string - format is AA:NN, where AA and NN are any number from 0 to 65535 range or alternatively well known string (local-AS | no-advertise | no-export | internet | additive).

Example: Creating community.

Name*	Acme_community
Type*	Community
Community*	permit 44395:666

13.2 BGP route-maps

Under the Net→E-BGP Route-maps section, you can define route-map policies, which can be associated with the BGP neighbors inbound or outbound.

Description of route-map fields:

- Sequence Number** - Automatically assigned a sequence number. Drag and move sequences to organize the order.
- Description** - Free description.
- Policy** - Permit or deny the routes which match below all match clauses within the current sequence.
- Match** - Rules for route matching.
 - Type** - Type of the object to match: AS-Path, Community, Extended Community, Large Community, IPv4 prefix-list, IPv4 next-hop, Route Source, IPv6 prefix-list, IPv6 next-hop, local-preference, MED, Origin, Route Tag.
 - Object** - Select an object from the list.
- Action** - Action when all match clauses are met.
 - Action type** - Define whether to manipulate a particular BGP attribute or go to another sequence.
 - Attribute** - The attribute to be manipulated.
 - Value** - New attribute value.

Example: route-map

Add E-BGP Route-maps

route-map-1

+ Sequence

Sequence Number	5	 
Description	Prefer acme on this neighbor	
Policy	Permit	

+ Match

Match	IPv4 prefix-list	Acme_Prefixes	
Match	Origin	egp	

+ Action

Action	set	local-preference	300	
--------	-----	------------------	-----	---

route-map-1

+ Sequence

Sequence Number	10	 
Description	permit the rest	
Policy	Permit	

+ Match

+ Action

Cancel **Add**

Static Routing

Located under Net→Routes is a method for describing static routing policies that Netris will dynamically inject on switches and/or SoftGate where appropriate. We recommend using the Routes only if BGP is not supported by the remote end.

Typical use cases for static routing:

- To connect the switch fabric to an ISP or upstream router in a situation where BGP and dual-homing are not supported.
- Temporary interconnection with the old network for a migration.
- Routing a subnet behind a VM hypervisor machine for an internal VM network.
- Specifically routing traffic destined to a particular prefix through an out-of-band management network.

Add new static route fields description:

- **Prefix** - Route destination to match.
- **Next-Hop** - Traffic destined to the Prefix will be routed towards the Next-Hop. Note that static routes will be injected only on units that have the Next-Hop as a connected network.
- **Description** - Free description.
- **Site** - Site where Route belongs.
- **State** - Administrative (enable/disable) state of the Route.
- **Apply to** - Limit the scope to particular units. It's typically used for Null routes.

Example: Default route pointing to a Next-Hop that belongs to one of V-NETs.

❖ Edit static route x

Prefix*	0.0.0.0/0		
Next-Hop*	10.0.3.10	<input type="checkbox"/> Null	?
Description	Default route		
Site*	Default	▼	
State	Enabled	▼	

+Apply to ?

Cancel Save

Example: Adding a back route to 10.254.0.0/16 through an out-of-band management network.

❖ Add new static route x

Prefix*	10.254.0.0/16		
Next-Hop*	10.254.96.1	<input type="checkbox"/> Null	?
Description	OOB Management backroute		
Site*	Default	▼	
State	Enabled	▼	

+Apply to ?

Cancel Add

Screenshot: This Shows that my back route is actually applied on leaf1 and spine1.

Prefix	Next-Hop	State	Applied	Apply to	Description	Site
10.254.0.0/16	10.254.96.1	Active	leaf1 spine1		OOB Management backroute	Default

NAT

Netris SoftGate nodes are required to support NAT (Network Address Translation).

15.1 Enabling NAT

To enable NAT for a given site, you first need to create a subnet with NAT purpose in the IPAM section. NAT IP addresses can be used for SNAT or DNAT as a global IP address (the public IP visible on the Internet). NAT IP pools are IP address ranges that SNAT can use as a rolling global IP (for a larger scale, similar to carrier-grade SNAT). SNAT is always overloading the ports, so many local hosts can share one or just a few public IP addresses. You can add as many NAT IP addresses and NAT pools as you need.

1. Allocate a public IP subnet for NAT under Net→IPAM.

Example: Adding an IP allocation under Net→Subnets.

Name*	NAT POOL 1
Prefix*	198.51.100.0/27
Tenant*	Admin
Type*	Subnet
Purpose*	nat
Sites*	Santa Clara

Cancel Add

1. Attach NAT IP addresses and/or NAT IP Pools to just one SoftGate node. Other SoftGate Nodes on the same site will automatically add the same NAT IP/Pool resources for proper consistency and high availability.

Example: Adding NAT IP addresses and NAT IP Address Pools to a SoftGate node.

Add Rule x

Name*	SNAT for all local nets	State	Enabled
Site	Santa Clara	Action	SNAT
Protocol	ALL		
Source Address*	10.0.0.0/8	Destination Address*	0.0.0.0/0
SNAT to Pool <input checked="" type="checkbox"/> SNAT to IP 198.51.100.0/27 (SNAT...) 198.51.100.1 			
Comment <div style="border: 1px solid #ccc; height: 40px; width: 100%; margin-top: 5px;"></div>			
Cancel Add			

15.2 Defining NAT rules

NAT rules are defined under Net→NAT.

Example: SNAT all hosts on 10.0.0.0/8 to the Internet using 198.51.100.65 as a global IP.

◆ Add Rule x

Name*	SNAT for all local nets	Action	SNAT
Protocol	ALL		
Source Source* 10.0.0.0/8		Destination Destination* 0.0.0.0/0	
Nat IP*	198.51.100.65/32(Default)	Status	Enable
Comment <div style="border: 1px solid #ccc; height: 40px; width: 100%; margin-top: 5px;"></div>			
Cancel Add			

Example: Port forwarding. DNAT the traffic destined to 198.51.100.66:80 to be forwarded to the host 10.0.4.10 on port tcp/1080.

Edit DNAT to internal server x

Name*	DNAT to internal server	State	Enabled
Site	Santa Clara	Action	DNAT
Protocol	TCP		
Source Address*	0.0.0.0/0	Destination Address*	198.51.100.64/29 (DNA...)
Source Port*	1-65535	Destination Port*	80
DNAT to IP*	10.0.4.10/27		
DNAT to Port*	1080		
Comment	<input type="text"/>		

Cancel Save

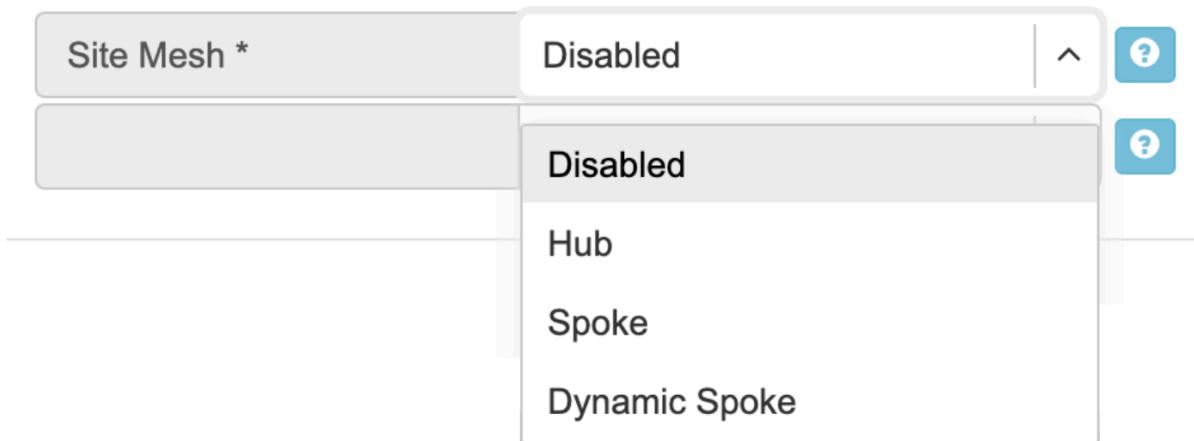
SiteMesh

SiteMesh is a Netris service for site-to-site interconnects over the public Internet. SiteMesh automatically generates configuration for WireGuard to create encrypted tunnels between participating sites and automatically generates a configuration for FRR to run dynamic routing. Hence, sites learn how to reach each other over the mesh WireGuard tunnels. The SiteMesh feature requires a SoftGate node at each participating site.

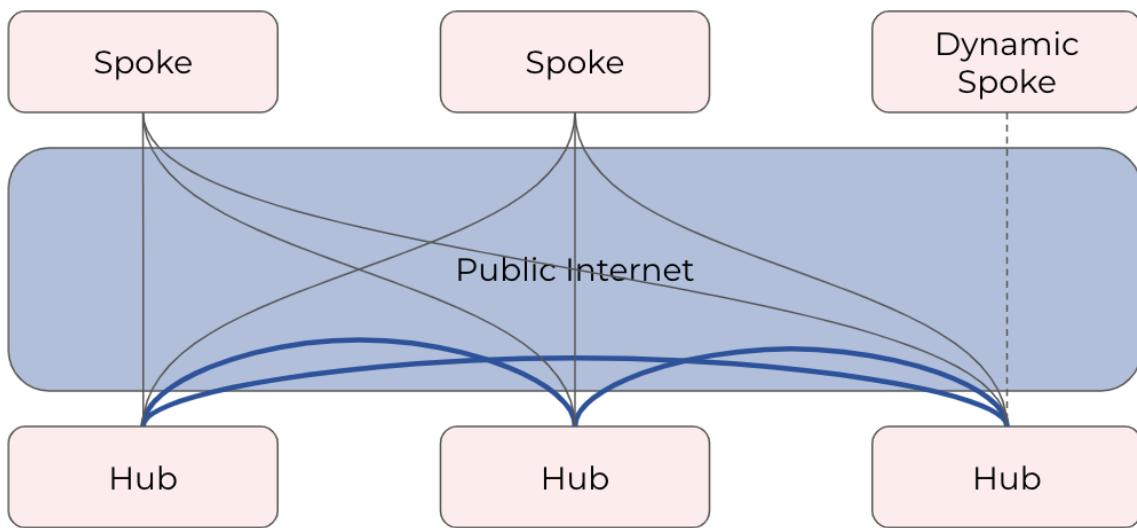
Edit Net->Sites, do declare what sites should form a SiteMesh. See SiteMesh types described below.

- **Disabled** - Do not participate in SiteMesh.
- **Hub** - Hub sites form full-mesh tunnels with all other sites (Hub and non-Hub) and can carry transit traffic for non-Hub sites. (usually major data center sites)
- **Spoke** - Spoke sites form tunnels with all Hub sites. Spoke to Spoke traffic will transit a Hub site. (small data center sites or major office sites)
- **Dynamic Spoke** - Dynamic Spoke is like Spoke, but it will maintain a tunnel only with one Hub site, based on dynamic connectivity measurements underneath and mathematical modeling. (small office sites)

Screenshot: Site Mesh parameter editing a Site under Net→Sites.



You only need to define your site-to-site VPN architecture policy by selecting SiteMesh mode for every site. Netris will generate the WireGuard tunnels (using randomly generated keys, and generate FRR rules to get the dynamic routing to converge).



Check the Net→Site Mesh section for the listing of tunnel statuses.

Screenshot: Listing of SiteMesh tunnels and BGP statuses (Net→Site Mesh)

Remote site	IT	Remote endpoint	IT	Status	IT	Last status cha...	IT	VPN enabled d...	IT
Silicon Valley	IT	SV-NFV1	IT	<div style="background-color: green; color: white; padding: 2px;">BGP: Established Prefix received: 1 Time: 4d13h3m</div>	IT	6/Sep/2020 18:05	IT	3/Sep/2020 04:56	IT

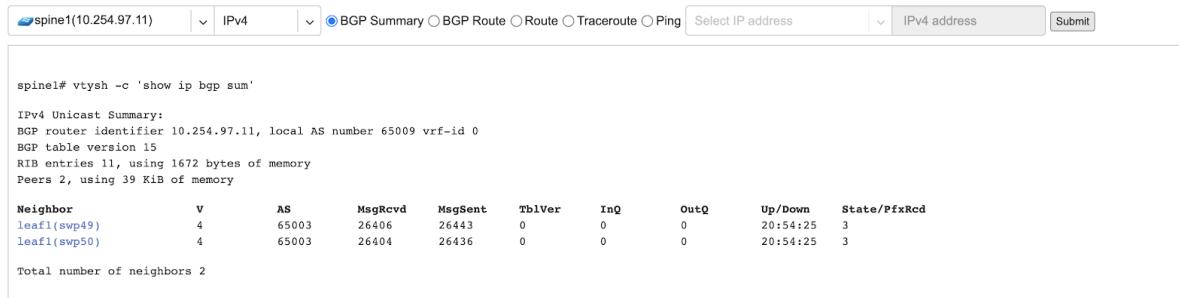
Looking Glass

The Looking Glass Is a GUI-based tool for looking up routing information from a switch or SoftGate perspective. You can access the Looking Glass either from Topology, individually for every device (right click on device → details → Looking Glass), or by navigating to Net→Looking Glass then selecting the device from the top-left dropdown menu.

Looking Glass controls described for IPv4/IPv6 protocol families.

- **BGP Summary** - Shows the summary of BGP adjacencies with neighbors, interface names, prefixes received. You can click on the neighbor name then query for the list of advertised/received prefixes.
- **BGP Route** - Lookup the BGP table (RIB) for the given address.
- **Route** - Lookup switch routing table for the given address.
- **Traceroute** - Conduct a traceroute from the selected device towards the given destination, optionally allowing to determine the source IP address.
- **Ping** - Execute a ping on the selected device towards the given destination, optionally allowing to select the source IP address.

Example: Spine1: listing BGP neighbors and number of received prefixes.



The screenshot shows the Looking Glass interface with the following details:

- Device: spine1(10.254.97.11)
- Protocol: IPv4
- Selected tab: BGP Summary (radio button selected)
- Buttons: Select IP address, IPv4 address, Submit

```

spine1# vtysh -c 'show ip bgp sum'

IPv4 Unicast Summary:
BGP router identifier 10.254.97.11, local AS number 65009 vrf-id 0
BGP table version 15
RIB entries 11, using 1672 bytes of memory
Peers 2, using 39 KiB of memory

Neighbor          V     AS   MsgRcvd   MsgSent    TblVer  InQ      OutQ     Up/Down  State/PfxRcd
leaf1(sw50)       4    65003  26406    26443      0        0        0        20:54:25  3
leaf1(sw49)       4    65003  26404    26436      0        0        0        20:54:25  3

Total number of neighbors 2
  
```

Example: BGP Route - looking up my leaf1 switch's loopback address from spine1's perspective. Spine1 is load balancing between two available paths.

The screenshot shows the BGP routing table entry for 10.254.97.12. The IP address 10.254.97.12 is highlighted with a red box in the search bar.

```

spine1# vtysh -c 'show ip bgp 10.254.97.12'
BGP routing table entry for 10.254.97.12/32
Paths: (2 available, best #2, table default)
Advertised to non peer-group peers:
leaf1(swp49) leaf1(swp50)
65003
  fe80::aa2b:b5ff:fed2:93e3 from leaf1(swp50) (10.254.97.12)
  (fe80::aa2b:b5ff:fed2:93e3) (used)
    Origin incomplete, metric 0, valid, external, multipath
    Community: 0:1
    AddPath ID: RX 0, TX 88
    Last update: Tue Jan 5 09:37:28 2021

  65003
  fe80::aa2b:b5ff:fed2:93df from leaf1(swp49) (10.254.97.12)
  (fe80::aa2b:b5ff:fed2:93df) (used)
    Origin incomplete, metric 0, valid, external, multipath, bestpath-from-AS 65003, best
    Community: 0:1
    AddPath ID: RX 0, TX 87
    Last update: Tue Jan 5 09:37:28 2021

```

Example: Ping.

The screenshot shows ping statistics between spine1 and 10.254.97.12. The IP address 10.254.97.12 is highlighted with a red box in the search bar.

```

spine1# ping -c 5 -I 10.254.97.11 10.254.97.12
PING 10.254.97.12 (10.254.97.12) from 10.254.97.11 : 56(84) bytes of data.
64 bytes from 10.254.97.12: icmp_seq=1 ttl=64 time=0.271 ms
64 bytes from 10.254.97.12: icmp_seq=2 ttl=64 time=0.236 ms
64 bytes from 10.254.97.12: icmp_seq=3 ttl=64 time=0.250 ms
64 bytes from 10.254.97.12: icmp_seq=4 ttl=64 time=0.278 ms
64 bytes from 10.254.97.12: icmp_seq=5 ttl=64 time=0.259 ms

--- 10.254.97.12 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.236/0.258/0.278/0.025 ms

```

Looking Glass controls described for the EVPN family.

- **BGP Summary** - Show brief summary of BGP adjacencies with neighbors, interface names, and EVPN prefixes received.
- **VNI** - List VNIs learned.
- **BGP EVPN** - List detailed EVPN routing information optionally for the given route distinguisher.
- **MAC table** - List MAC address table for the given VNI.

Example: Listing of adjacent BGP neighbors and number of EVPN prefixes received.

The screenshot shows the BGP summary output. The IP address 10.254.97.11 is highlighted with a red box in the search bar.

```

spine1# vtysh -c 'show ip bgp sum'

IPv4 Unicast Summary:
BGP router identifier 10.254.97.11, local AS number 65009 vrf-id 0
BGP table version 15
RIB entries 11, using 1672 bytes of memory
Peers 2, using 39 KiB of memory

Neighbor      V        AS      MsgRcvd      MsgSent      TblVer      InQ       OutQ      Up/Down      State/PfxRcd
leaf1(swp49)   4      65003     26406     26443         0          0          0      20:54:25      3
leaf1(swp50)   4      65003     26404     26436         0          0          0      20:54:25      3

Total number of neighbors 2

```

Example: Listing MAC addresses on VNI 2.

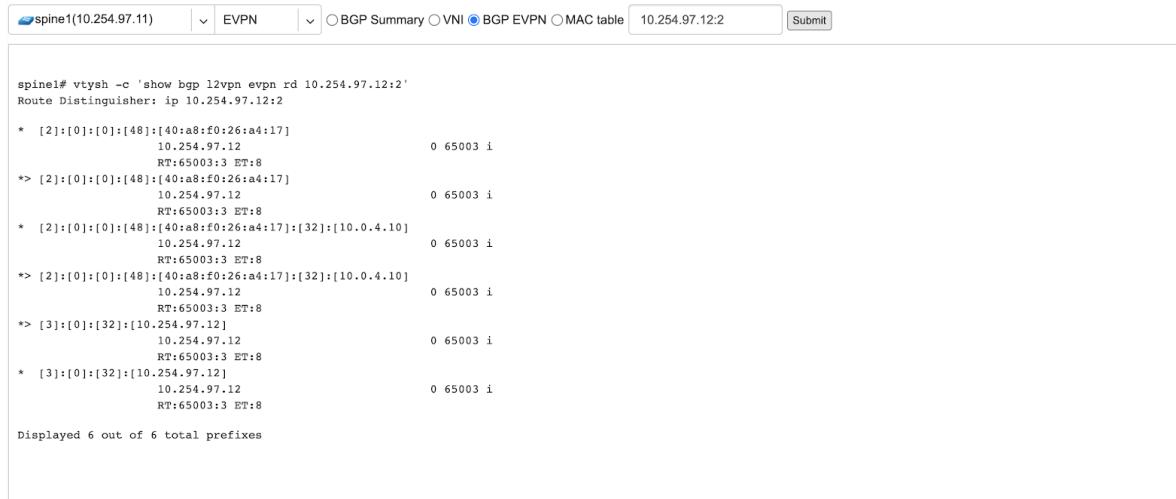
The screenshot shows the MAC addresses on VNI 2. The VNI 2 is highlighted with a red box in the search bar.

```

spine1# vtysh -c 'show evpn mac vni 2'
Number of MACs (local and remote) known for this VNI: 2
MAC           Type     Intf/Remote VTEP      VLAN Seq #
02:00:5e:00:00:01 local     swp52.2          0/0
2c:76:8a:52:3c:4b remote  10.254.97.12      0/0

```

Example: EVPN routing information listing for a specified route distinguisher.



The screenshot shows a network configuration interface with the following details:

- Device: spine1 (IP: 10.254.97.11)
- Tab: EVPN
- Filter: BGP Summary (selected)
- Route Distinguisher: ip 10.254.97.12:2
- Address: 10.254.97.12
- RT: 65003:3
- ET: 8
- Count: 6 total prefixes displayed

```

spine1# vtysh -c 'show bgp l2vpn evpn rd 10.254.97.12:2'
Route Distinguisher: ip 10.254.97.12:2
* [2]:[0]:[0]:[48]:[40:a8:f0:26:a4:17]
    10.254.97.12          0 65003 i
    RT:65003:3 ET:8
*> [2]:[0]:[0]:[48]:[40:a8:f0:26:a4:17]
    10.254.97.12          0 65003 i
    RT:65003:3 ET:8
* [2]:[0]:[0]:[48]:[40:a8:f0:26:a4:17]:[32]:[10.0.4.10]
    10.254.97.12          0 65003 i
    RT:65003:3 ET:8
*> [2]:[0]:[0]:[48]:[40:a8:f0:26:a4:17]:[32]:[10.0.4.10]
    10.254.97.12          0 65003 i
    RT:65003:3 ET:8
*> [3]:[0]:[32]:[10.254.97.12]
    10.254.97.12          0 65003 i
    RT:65003:3 ET:8
* [3]:[0]:[32]:[10.254.97.12]
    10.254.97.12          0 65003 i
    RT:65003:3 ET:8
Displayed 6 out of 6 total prefixes

```

V-Net

V-Net is a virtual networking service that provide a Layer-2 (unrouted) or Layer-3 (routed) virtual network segments on switch ports anywhere on the switch fabric. V-NETs can be created and managed by a single tenant (single team) or they can be created and managed collaboratively by multiple tenants (different teams inside and/or outside the organization). Netris automatically configures a VXLAN with an EVPN control plane over an unnumbered BGP Layer-3 underlay network and organize the high availability for the default gateway behind the scenes.

18.1 V-Net Fields

- **Name** - Unique name for the V-Net
- **Owner** - Tenant, who can make any changes to current V-Net
- **V-Net state** - Active/Disable state for entire V-Net
- **VLAN aware** - Enable VLAN aware bridge, use only in rare cases, if otherwise is not possible
- **Guest tenants** - List of tenants allowed to add/edit/remove ports to the V-Net but not manage other parameters
- **Sites** - Ports from these sites will be allowed to participate in the V-Net. (Multi-site circuits would require backbone connectivity between sites).
- **IPv4 Gateway** - IPv4 address to be used as a default gateway in this V-Net. Should be configured under Net→IPAM as an assignment, assigned to the owner tenant, and available in the site where V-Net is intended to span.
- **IPv6 Gateway** - IPv6 address to be used as a default gateway in this V-Net. Should be configured under Net→IPAM as an assignment, assigned to the owner tenant, and available in the site or sites where V-Net is intended to span.
- **Port** - Physical Switch Port anywhere on the network. Switch Port should be assigned to the owner or guest tenant under Net→Switch Ports.
 - **Enabled** - Enable or disable individual Switch Port under current V-Net
 - **Port Name** - Switch Port format: <alias>(swp<number>)@<switch name>
 - **VLAN ID / Untag** - Specify a VLAN ID for tagging traffic on a per-port basis or set Untag not to use tagging on a particular port. VLAN tags are only significant on each port's ingress/egress unless VLAN aware mode is used.
 - **LAG Mode** - Allows for active-standby dual-homing, assuming LAG configuration on the remote end. Active/active dual homing will be enabled in future releases (dependence on SVI support by NOSes).

Tip Many switches can't autodetect old 1Gbps ports. If attaching hosts with 1Gbps ports to 10Gbps

switch ports, you'll need to change the speed for a given Switch Port from Auto(default) to 1Gbps. You can edit a port in Net→Switch Ports individually or in bulk.

Add new V-Net

Name*	Servers1
Owner*	Admin
V-Net state	Active
VLAN aware	<input type="checkbox"/>
Guest tenants	Select Tenants
Sites*	Santa Clara
IPv4 Gateway	198.51.100.240/28 (Servers1-su...)
IPv6 Gateway	
Add Port	
<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Search
<input checked="" type="checkbox"/> Enabled	swp1(swp1)@leaf1 (Admin)
<input checked="" type="checkbox"/> Enabled	swp2(swp2)@leaf1 (Admin)
<input checked="" type="checkbox"/> Enabled	swp3(swp3)@leaf1 (Admin)
Rows per page: 50 ▲	
<input type="button" value="Cancel"/> <input type="button" value="Add"/>	

Example: Adding a new V-Net.

V-Net	State	Gateways	Created Date	Modified Date
> Internet VXLAN ID: 4 Anycast MAC: 02:00:5e:00:00:00 Ports count: 1	Active	50.117.59.97/30	1/Oct/2021 05:10	1/Oct/2021 05:10
> V-net1 VXLAN ID: 2 Anycast MAC: 02:00:5e:00:00:00 Ports count: 1	Active	10.10.10.1/25	1/Oct/2021 04:43	1/Oct/2021 04:43
> VNet2 VXLAN ID: 3 Anycast MAC: 02:00:5e:00:00:00 Ports count: 1	Active	10.10.10.129/25	1/Oct/2021 04:46	1/Oct/2021 04:46

Example: Listing of V-Nets.

V-Net	State	Gateways	Created Date	Modified Date
Internet	Active	50.117.59.97/30	1/Oct/2021 05:10	1/Oct/2021 05:10
VXLAN ID: 4 Anycast MAC: 02:00:5e:00:00:00 Ports count: 1				
Owner: Admin Tenants: Sites: Santa Clara				
Show graphs: <input type="checkbox"/> all <input type="checkbox"/> throughput <input type="checkbox"/> packets <input type="checkbox"/> errors <input type="checkbox"/> optical <input type="checkbox"/> MAC count				
Brief view: <input type="checkbox"/>				
<input type="text"/> Search				
swp7@SoftGate1 (Admin) VLAN ID: 1071 Enabled Site: Santa Clara Tenant: Admin Show graphs: <input checked="" type="checkbox"/>				
MAC count 2 (show)		Port status Link Up (?)	Details Optical: Utilization: MTU: 9000 LAG Mode: Disabled	Dates Added: 1/Oct/2021 12:00 Modified: 1/Oct/2021 12:00
0 - 1 (1) records.				
Rows per page: 5 ▲				

Example: Expanded view of a V-Net listing.

L3 Load Balancer (Anycast LB)

L3 (Anycast) load balancer leverages ECMP load balancing and hashing capability of spine and leaf switches to deliver line-rate server load balancing with health checks.

ROH servers, besides advertising their unicast (unique) loopback IP address, need to configure and advertise an additional anycast (the same IP) IP address. Unicast IP address is used for connecting to each individual server.

End-user traffic should be destined to the anycast IP address. The switch fabric uses ECMP to load balance the traffic towards every server, and will hash sessions based on IP/Protocol/Port such that TCP sessions will exist between the given end-user and server pair for the lifetime of the session. Optional health checks are used to identify application failures and reroute traffic in the case of a server outage.

19.1 Creating an L3 Load Balancer

To configure L3 (Anycast) load balancing:

1. Navigate to **Services→Instances (ROH)** and locate an existing ROH instance
2. Click the ellipses and then the **Edit** button
3. Select an extra IPv4 address from the select box at the bottom, and check the **Anycast** option.
4. This will create a service under **Services→Load Balancer** and permit using the Anycast IP address in multiple ROH instances.

L4 Load Balancer (L4LB)

Netris L4 Load Balancer (L4LB) leverages SoftGate(Linux router) nodes to provide Layer-4 load balancing services, including on-demand cloud load balancing with native integration with Kubernetes.

20.1 Enabling L4LB service

L4 Load Balancer service requires at least one SoftGate node to be available in a given Site, as well as at least one IP address assignment (purpose=load balancer).

The IP address pool for L4LB can be defined in the Net→IPAM section by adding an Allocation and setting the purpose field to 'load-balancer'. You can define multiple IP pools for L4LB at any given site. See the below example.

Example: Adding a load-balancer IP pool assignment.

Add Allocation

Name*	<input type="text"/>
Prefix*	<input type="text"/>
Tenant*	<input type="text"/>
Type*	<input type="text"/> Allocation

Cancel Add

Screenshot: Listing of Net→IPAM after adding a load-balancer assignment

Prefix	Name	Type	Purpose	Utilization	Site	Tenant
> 10.0.0.0/24	ROH-test	Allocation		100% (subnets)		Admin
> 10.10.10.0/24	V-Net	Allocation		100% (subnets)		Admin
> 10.254.45.0/24	MGMT	Allocation		100% (subnets)		Admin
> 10.254.46.0/24	Loopback	Allocation		100% (subnets)		Admin
> 50.117.59.96/30	50.117.59.96/30	Allocation		125% (subnets)		Admin
> 50.117.59.160/28	Public	Allocation		100% (subnets)		Admin
> 169.254.255.1/32	Controller Loopback Allocation	Allocation		100% (subnets)		Admin
> 192.168.110.0/24	k8s-vnet	Allocation		100% (subnets)		Admin
▼ 198.51.100.0/24	Documentation	Allocation		6% (subnets)		Admin
198.51.100.0/28	L4LB IP pool	Subnet	load-balancer	0% (subnets)	Santa Clara	Admin

20.2 Consuming L4LB service

This guide describes how to request an L4 Load Balancer using GUI. For Kubernetes integration, check the Kubenet section.

Click +add under Services→L4 Load Balancer to request an L4LB service.

Add new L4 Load Balancer fields are described below:

General fields

- **Name** - Unique name.
- **Protocol** - TCP or UDP.
- **Tenant** - Requestor Tenant should have access to the backend IP space.
- **Site** - Site where L4LB service is being requested for. Backends should belong on this site.
- **State** - Administrative state.

Frontend

- **Address** - Frontend IP address to be exposed for this L4LB service. “Assign automatically” will provide the next available IP address from the defined load-balancer pool. Alternatively, users can select manually from the list of available addresses.
- **Port** - TCP or UDP port to be exposed.

Health-check

- **Type** - Probe backends on service availability.
 - **None** - load balance unconditionally.
 - **TCP** - probe backend service availability through TCP connect checks.
 - **HTTP** - probe backend service availability through HTTP GET checks.
- **Timeout(ms)** - Probe timeout in milliseconds.
- **Request path** - HTTP request path.

Backend

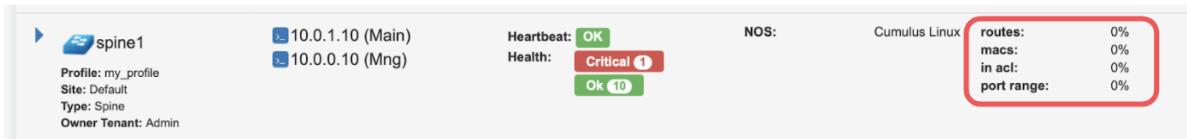
- **+Add** - add a backend host.
- **Address** - IP address of the backend host.
- **Port** - Service port on the backend host.
- **Enabled** - Administrative state of particular backend.

Access Control Lists (ACL)

Netris supports ACLs for switch network access control. (ACL and ACL2.0) ACL is for defining network access lists in a source IP: Port, destination IP: Port format. ACL2.0 is an object-oriented service way of describing network access.

Both ACL and ACL2.0 services support tenant/RBAC based approval workflows. Access control lists execute in switch hardware providing line-rate performance for security enforcement. It's important to keep in mind that the number of ACLs is limited to the limited size of TCAM of network switches.

Screenshot: TCAM utilization can be seen under Net→Inventory



Netris is applying several optimization algorithms to minimize the usage of TCAM while achieving the user-defined requirements.

21.1 ACL Default Policy

The ACL default policy is to permit all hosts to communicate with each other. You can change the default policy on a per Site basis by editing the Site features under Net→Sites. Once the "ACL Default Policy" is changed to "Deny," the given site will start dropping any traffic unless specific communication is permitted through ACL or ACL2.0 rules.

Example: Changing "ACL Default Policy" for the site "siteDefault".

❖ Edit Site siteDefault

Name *	siteDefault	
Border(Leaf) Switch ASN *	65008	
Spine Switch ASN *	65009	
TOR(Leaf) Switch ASN *	65003	
Hypervisor ASN *	65002	
ROH instance ASN *	65500	
ROH virtual instance ASN *	65501	
ROH Routing Profile *	Default	
Site Mesh *	Disabled	
ACL Default Policy	Permit	

21.2 ACL Rules

ACL rules can be created, listed, edited, approved under Services→ACL.

Description of ACL fields. General

- **Name** - Unique name for the ACL entry.
- **Protocol** - IP protocol to match.
 - All - Any IP protocols.
 - IP - Specific IP protocol number.
 - TCP - TCP.
 - UDP - UDP.
 - ICMP ALL - Any IPv4 ICMP protocol.
 - ICMP Custom - Custom IPv4 ICMP code.
 - ICMPv6 ALL - Any IPv6 ICMP protocol.
 - ICMPv6 Custom - Custom IPv6 ICMP code.
- **Active Until** - Disable this rule at the defined date/time.
- **Action** - Permit or Deny forwarding of matched packets.
- **Established/Reverse** - For TCP, also match reverse packets except with TCP SYN flag. For non-TCP, also generate a reverse rule with swapped source/destination.

Source/Destination - Source and destination addresses and ports to match.

- **Source IPv4/IPv6** - IPv4/IPv6 address.
- **Ports Type**
 - Port Range - Match on the port or a port range defined in this window.

- **Port Group** - Match on a group of ports defined under Services→ ACL Port Group.
- **From Port** - Port range starting from.
- **To Port** - Port range ending with.
- **Comment** - Descriptive comment, commonly used for approval workflows.
- **Check button** - Check if Another ACL on the system already permits the described network access.

Example: Permit hosts in 10.0.3.0/24 to access hosts in 10.0.5.0/24 by SSH, also permit the return traffic (Established).

Name*	commonServers_to_ROH																									
Protocol	TCP	<input type="button" value="▼"/>																								
<input type="checkbox"/> Active Until Mon Dec 28 2020 18:45:58 GMT																										
Action	Permit																									
<input checked="" type="checkbox"/> Established																										
Source <table border="1"> <tr> <td>Source*</td> <td colspan="2">10.0.3.0/24</td> </tr> <tr> <td>Ports Type</td> <td>Port Range</td> <td><input type="button" value="▼"/></td> </tr> <tr> <td>From port*</td> <td colspan="2">1</td> </tr> <tr> <td>To port*</td> <td colspan="2">65535</td> </tr> </table> Destination <table border="1"> <tr> <td>Destination*</td> <td colspan="2">10.0.5.0/24</td> </tr> <tr> <td>Ports Type</td> <td>Port Range</td> <td><input type="button" value="▼"/></td> </tr> <tr> <td>From port*</td> <td colspan="2">22</td> </tr> <tr> <td>To port*</td> <td colspan="2">22</td> </tr> </table>			Source*	10.0.3.0/24		Ports Type	Port Range	<input type="button" value="▼"/>	From port*	1		To port*	65535		Destination*	10.0.5.0/24		Ports Type	Port Range	<input type="button" value="▼"/>	From port*	22		To port*	22	
Source*	10.0.3.0/24																									
Ports Type	Port Range	<input type="button" value="▼"/>																								
From port*	1																									
To port*	65535																									
Destination*	10.0.5.0/24																									
Ports Type	Port Range	<input type="button" value="▼"/>																								
From port*	22																									
To port*	22																									
Comment <input type="text"/>																										
<input type="button" value="Cancel"/> <input type="button" value="Check"/> <input type="button" value="Save"/>																										

Example: “Check” shows that requested access is already provided by a broader ACL rule.

Name*	commonServers_to_ROH		<input type="checkbox"/> Active Until	Mon Dec 28 2020 19:04:55 GMT	
Protocol	TCP		Action	Permit	

Established

Source		Destination			
Source*	10.0.3.0/24		Destination*	10.0.5.15/32	
Ports Type	Port Range		Ports Type	Port Range	
From port*	1		From port*	22	
To port*	65535		To port*	22	

Comment

commonServers_to_R 10.0.3.0/24 1-65535 10.0.5.0/24 22 tcp permit

Cancel Check Save

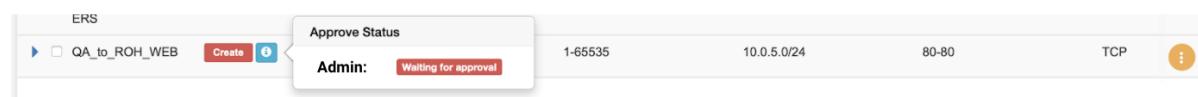
21.3 ACL Approval Workflow

When one tenant (one team) needs to get network access to resources under the responsibility of another tenant (another team), an ACL can be created but will activate only after approval of the tenant responsible for the destination address resources. See the below example.

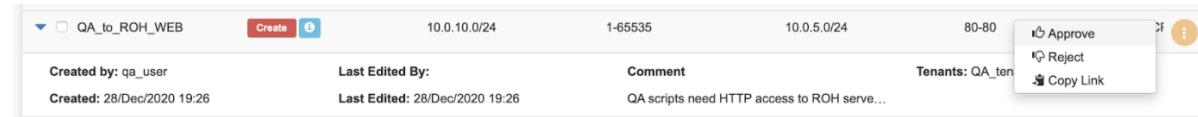
Example: User representing QA_tenant is creating an ACL where source belongs to QA_tenant, but destination belongs to the Admin tenant.

Name*	QA_to_ROH_WEB	
Protocol	TCP	<input type="checkbox"/> Active Until Mon Dec 28 2020 19:26:35 GMT
<input checked="" type="checkbox"/> Established		Action Permit
Source	Destination	
Source*	10.0.10.0/24	
Ports Type	Port Range	<input type="checkbox"/>
From port*	1	
To port*	65535	
Comment	QA scripts need HTTP access to ROH servers for auto testing.	
<input type="button" value="Cancel"/> <input type="button" value="Check"/> <input type="button" value="Save"/>		

Screenshot: ACL stays in “waiting for approval” state until approved.



Screenshot: Users of tenant Admin, receive a notification in the GUI, and optionally by email. Then one can review the access request and either approve or reject it.



Screenshot: Once approved, users of both tenants will see the ACL in the “Active” state, and soon Netris Agents will push the appropriate config throughout the switch fabric.



21.4 ACL Processing Order

1. User-defined Deny Rules
2. User-defined Permit Rules
3. Deny the rest

ROH (Routing on the Host)

To create more resilient and higher-performance data centers, some companies leverage the Linux ecosystem to run routing protocols directly on their servers. This is commonly known as ROH (Routing on the Host).

In ROH architectures, servers use a routing daemon to establish a BGP adjacency with the switch fabric on every physical link. ROH can run on bare metal servers, VMs, and even containers. The most commonly used routing daemon/suite is FRR.

Hosts connected to the network in ROH architecture don't have IP addresses on a shared Ethernet segment; instead an IP address is configured on the loopback interface and advertised over all BGP links towards switch fabric. This is a modern and optimal design, leveraging Layer-3 networking from the fabric to the servers.

By using only Layer-3 interfaces, Layer-2 protocols such as Spanning Tree (STP) can be minimized and the reliability of the network increases.

The ROH architecture that is configured by Netris allows for leveraging ECMP load balancing capabilities of the switching hardware for the high-performance server load balancing (described in L3 Load Balancer section). For each instance of ROH, you'll need to create a ROH entry in Netris Controller.

22.1 Adding ROH Hosts

1. Navigate in the Netris UI to **Services→Instances (ROH)**
2. Click the **Add** button
3. Fill out the form based on the fields in the table below.
4. Click the **Add** button

Description of ROH instance fields:

- **Name** - Unique name for the ROH instance
- **Site** - Site where the current ROH instance belongs
- **Type** - Physical Server, for all servers forming a BGP adjacency directly with the switch fabric. Hypervisor, for using the hypervisor as an interim router. Proxmox is currently the only supported hypervisor.
- **ROH Routing Profile** - ROH Routing profile defines what set of routing prefixes to be advertised to ROH instances
 - **Default route only (most common design)** - Will advertise 0.0.0.0/0 + loopback address of the physically connected switch
 - **Default + Aggregate** - Will add prefixes of defined assignments + “Default” profile

- **Full table** - Will advertise all prefixes available in the routing table of the connected switch
- **Inherit** - Will inherit policy from site objects defined under Net→Sites
- **Legacy Mode** - Switch from default zero-config mode to using /30 IP addresses. Used for MSFT Windows Servers or other OS that doesn't support FRR.
- **+Port** - Physical Switch Ports anywhere on the network.
- **+IPv4** - IPv4 addresses for the loopback interface.
- **+Inbound Prefix List** - List of additional prefixes that the ROH server may advertise. Sometimes used to advertise container or VM networks.

Tip Many switches can't autodetect old 1Gbps ports. If attaching hosts with 1Gbps ports to 10Gbps switch ports, you'll need to change the speed for a given Switch Port from Auto(default) to 1Gbps. You can edit a port in Net→Switch Ports individually or in bulk.

Kubenernetes Integration

Netris integrates with Kube API to provide on-demand load balancer and other Kubernetes specific networking features. Netris-Kubernetes integration is designed to complement Kubernetes CNI networking and provide a cloud-like user experience to local Kubernetes clusters.

23.1 Install Netris Operator

Integration between the Netris Controller and the Kubernetes API is completed by installing the Netris Operator. Installation can be accomplished by installing regular manifests or a helm chart:

23.1.1 Helm Chart Method

Instructions are available on Github: <https://github.com/netrisai/netris-operator/tree/master/deploy/charts/netris-operator#installing-the-chart>

23.1.2 Regular Manifest Method

1. Install the latest Netris Operator:

```
kubectl apply -f
https://github.com/netrisai/netris-operator/releases/latest/download/netris-operator.yaml
```

2. Create credentials secret for Netris Operator:

```
kubectl -n netris-operator create secret generic netris-creds \
--from-literal=host='**your-netris-controller-ip**' \
--from-literal=login='**your-netris-admin-username**' \
--from-literal=password='**your-netris-admin-password**'
```

3. Inspect the pod logs and make sure the operator is connected to Netris Controller:

```
kubectl -n netris-operator logs -l netris-operator=controller-manager
--all-containers -f
```

Example output demonstrating the successful operation of Netris Operator:

```
{"level": "info", "ts": 1629994653.6441543, "logger": "controller", "msg": "Starting workers", "reconcilerGroup": "k8s.netris.ai", "reconcilerKind": "L4LB", "controller": "l4lb", "worker count": 1}
```

Note After installing the Netris Operator, your Kubernetes cluster and physical network control

planes are connected.

23.2 Using Type ‘LoadBalancer’

In this scenario we will be installing a simple application that requires a network load balancer:

Install the application “Podinfo”:

```
kubectl apply -k https://github.com/stefanprodan/podinfo/kustomize
```

Get the list of pods and services in the default namespace:

```
kubectl get po,svc
```

As you can see, the service type is “ClusterIP”:

NAME	READY	STATUS	RESTARTS	AGE
pod/podinfo-576d5bf6bd-7z9jl	1/1	Running	0	49s
pod/podinfo-576d5bf6bd-nhlmh	1/1	Running	0	33s
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/podinfo	ClusterIP	172.21.65.106	<none>	9898/TCP, 9999/TCP
				50s

In order to request access from outside, change the type to “LoadBalancer”:

```
kubectl patch svc podinfo -p '{"spec": {"type": "LoadBalancer"}}'
```

Check the services again:

```
kubectl get svc
```

Now we can see that the service type changed to LoadBalancer, and “EXTERNAL-IP” switched to pending state:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
AGE				
podinfo	LoadBalancer	172.21.65.106	<pending>	9898:32584/TCP, 9999:30365/TCP
				8m57s

Going into the Netris Controller web interface, navigate to **Services / L4 Load Balancer**, and you may see L4LBs provisioning in real-time. If you do not see the provisioning process it is likely because it already completed. Look for the service with the name “**podinfo-xxxxxxxx**”

Service name	Frontend	Status	Backends	Site	Tenant
sandbox-k8s-apiservers	████████:6443	OK	Active	US/NYC	Admin
Protocol: TCP					
podinfo-default-████████...	████████:9999	Provisioning	Unhealthy	US/NYC	Admin
Protocol: TCP					

After provisioning has finished, inspect the service in k8s:

```
kubectl get svc
```

You can see that “EXTERNAL-IP” has been injected into Kubernetes:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
AGE				
podinfo	LoadBalancer	172.21.65.106	50.117.59.202	

```
9898:32584/TCP, 9999:30365/TCP    9m17s
```

23.3 Using Netris Custom Resources

23.3.1 Introduction to Netris Custom Resources

In addition to provisioning on-demand network load balancers, Netris Operator can also provide automatic creation of network services based on Kubernetes CRD objects. Let's take a look at a few common examples:

23.3.2 L4LB Custom Resource

In the previous section, when we changed the service type from “ClusterIP” to “LoadBalancer”, Netris Operator detected a new request for a network load balancer, then it created L4LB custom resources. Let's see them:

```
kubectl get l4lb
```

As you can see, there are two L4LB resources, one for each podinfo's service port:

NAME	PORT	SITE	TENANT	STATUS	AGE	STATE	FRONTEND
podinfo-default-66d44feb-0278-412a-a32d-73afe011f2c6-tcp-9898	50.117.59.202	9898/TCP	US/NYC	Admin	OK	33m	active
podinfo-default-66d44feb-0278-412a-a32d-73afe011f2c6-tcp-9999	50.117.59.202	9999/TCP	US/NYC	Admin	OK	32m	active

You can't edit/delete them, because Netris Operator will recreate them based on what was originally deployed in the service specifications.

Instead, let's create a new load balancer using the CRD method. This method allows us to create L4 load balancers for services outside of what is being created natively with the Kubernetes service schema. Our new L4LB's backends will be “srv04-nyc” & “srv05-nyc” on TCP port 80. These servers are already running the Nginx web server, with the hostname present in the index.html file.

Create a yaml file:

```
cat << EOF > srv04-5-nyc-http.yaml
apiVersion: k8s.netris.ai/v1alpha1
kind: L4LB
metadata:
  name: srv04-5-nyc-http
spec:
  ownerTenant: Admin
  site: US/NYC
  state: active
  protocol: tcp
  frontend:
    port: 80
  backend:
    - 192.168.45.64:80
    - 192.168.46.65:80
  check:
    type: tcp
    timeout: 3000
EOF
```

And apply it:

```
kubectl apply -f srv04-5-nyc-http.yaml
```

Inspect the new L4LB resources via kubectl:

```
kubectl get l4lb
```

As you can see, provisioning started:

NAME	PORT	SITE	TENANT	STATUS	AGE	STATE	FRONTEND
podinfo-default-d07acd0f-51ea-429a-89dd-8e4c1d6d0a86-tcp-9898	50.117.59.202	9898/TCP	US/NYC	Admin	OK	2m17s	
podinfo-default-d07acd0f-51ea-429a-89dd-8e4c1d6d0a86-tcp-9999	50.117.59.202	9999/TCP	US/NYC	Admin	OK	3m47s	
srv04-5-nyc-http	50.117.59.203	80/TCP	US/NYC	Admin	Provisioning	6s	active

When provisioning is finished, you should be able to connect to L4LB. Try to curl, using the L4LB frontend address displayed in the above command output:

```
curl 50.117.59.203
```

You will see the servers' hostname in curl output:

```
SRV04-NYC
```

You can also inspect the L4LB in the Netris Controller web interface:

Service name	Frontend	Status	Backends
▶ sandbox-k8s-apiservers Protocol: TCP	50.117.59.203:6443	OK	Active (2)
▶ podinfo-default- Protocol: TCP	50.117.59.203:80	OK	Active (1)
▶ podinfo-default- Protocol: TCP	50.117.59.203:9999	OK	Active (1)
▼ srv04-5-nyc-http Protocol: TCP	50.117.59.203:80	OK	Active (1) Unhealthy (1)

Health-check: TCP
Timeout: 3000
Backend addresses

Backend Status	Health-check response
Active	connection succeed
Unhealthy	dial tcp 192.168.1.65:80: i/o timeout

23.3.3 V-Net Custom Resource

You can also create Netris V-Nets (L2 segments) via Kubernetes with a simple manifest:

```
cat << EOF > vnet-customer.yaml
apiVersion: k8s.netris.ai/v1alpha1
kind: VNet
metadata:
  name: vnet-customer
spec:
  ownerTenant: Admin
  guestTenants: []
  sites:
    - name: US/NYC
      gateways:
        - 192.168.46.1/24
  switchPorts:
EOF
```

```
- name: swp2@sw22-nyc
EOF
```

And apply it:

```
kubectl apply -f vnet-customer.yaml
```

Let's check our VNet resources in Kubernetes:

```
kubectl get vnet
```

As you can see, provisioning for our new VNet has started:

NAME	STATE	GATEWAYS	SITES	OWNER	STATUS	AGE
vnet-customer	active	192.168.46.1/24	US/NYC	Admin	Provisioning	7s

After provisioning has completed, the L4LB's checks should work for both backend servers, and incoming requests should be balanced between them.

23.3.4 BGP Custom Resource

You can create BGP peers via Kubernetes manifests:

1. Create a yaml file:

```
cat << EOF > isp2-customer.yaml
apiVersion: k8s.netris.ai/v1alpha1
kind: BGP
metadata:
  name: isp2-customer
spec:
  site: US/NYC
  softgate: SoftGate2
  neighborAs: 65007
  transport:
    name: swp14@sw02-nyc
    vlanId: 1092
  localIP: 50.117.59.118/30
  remoteIP: 50.117.59.117/30
  description: Example BGP to ISP2
  prefixListOutbound:
    - permit 50.117.59.192/28 le 32
EOF
```

2. Apply the manifest file:

```
kubectl apply -f isp2-customer.yaml
```

3. Check created BGP:

```
kubectl get bgp
```

Allow up to 1 minute for both sides of the BGP sessions to come up:

NAME	STATE	BGP STATE	PORT STATE	NEIGHBOR AS	LOCAL ADDRESS
REMOTE ADDRESS	AGE				
isp2-customer	enabled			65007	50.117.59.118/30
50.117.59.117/30	15s				

Then check the state again:

```
kubectl get bgp
```

The output is similar to this:

NAME	STATE	BGP STATE	REMOTE ADDRESS	AGE	PORT
STATE	NEIGHBOR	AS	LOCAL ADDRESS		
isp2-customer		enabled	bgp: Established; prefix: 30; time: 00:00:51 65007	2m3s	UP
			50.117.59.118/30	50.117.59.117/30	

Feel free to use the import annotation for this BGP if you created it from the controller web interface previously.

Return to the Netris UI and navigate to [Net / Topology](#) to see the new BGP neighbor you created.

23.3.5 Importing existing resources from Netris Controller to Kubernetes

You can import any custom resources, already created from the Netris Controller to k8s by adding this annotation:

```
resource.k8s.netris.ai/import: "true"
```

Otherwise, if try to apply them w/out “import” annotation, the Netris Operator will complain that the resource with such name or specs already exists.

After importing resources to k8s, they will belong to the Netris Operator, and you won’t be able to edit/delete them directly from the Netris Controller web interface, because the Netris Operator will put everything back, as declared in the custom resources.

23.3.6 Reclaim Policy

There is also one useful annotation. So suppose you want to remove some custom resource from k8s, and want to prevent its deletion from the controller, for that you can use “reclaimPolicy” annotation:

```
resource.k8s.netris.ai/reclaimPolicy: "retain"
```

Just add this annotation in any custom resource while creating it. Or if the custom resource has already been created, change the “delete” value to “retain” for key `resource.k8s.netris.ai/reclaimPolicy` in the resource annotation. After that, you’ll be able to delete any Netris Custom Resource from Kubernetes, and it won’t be deleted from the controller.

See also

See all options and examples for Netris Custom Resources [here](#).

23.4 Calico CNI Integration

Netris Operator can integrate with Calico CNI. This annotation will automatically create BGP peering between cluster nodes and the leaf/TOR switch for each node, then to clean up it will disable Calico Node-to-Node mesh. To understand why you need to configure peering between Kubernetes nodes and the leaf/TOR switch, and why you should disable Node-to-Node mesh, review the [calico docs](#).

Integration is very simple, just need to add the annotation in calico’s `bgpconfigurations` custom resource. Before doing that, let’s see the current state of `bgpconfigurations`:

```
kubectl get bgpconfigurations default -o yaml
```

As we can see, `nodeToNodeMeshEnabled` is enabled, and `asNumber` is 64512 (it’s Calico default AS number):

```

apiVersion: crd.projectcalico.org/v1
kind: BGPConfiguration
metadata:
  annotations:
    ...
  name: default
  ...
spec:
  asNumber: 64512
  logSeverityScreen: Info
  nodeToNodeMeshEnabled: true

```

Let's enable the "netris-calico" integration:

```
kubectl annotate bgpconfigurations default manage.k8s.netris.ai/calico='true'
```

Let's check our BGP resources in k8s:

```
kubectl get bgp
```

Here are our freshly created BGPs, one for each k8s node:

NAME	STATE	BGP STATE	LOCAL ADDRESS	REMOTE ADDRESS	AGE
PORT	STATE	NEIGHBOR AS			
isp2-customer	enabled	bgp: Established; prefix: 28; time: 0:06:18	50.117.59.118/30	50.117.59.117/30	7m59s
0:06:18	UP	65007			
sandbox9-srv06-nyc-192.168.110.66	enabled	4200070000	192.168.110.1/24	192.168.110.66/24	26s
sandbox9-srv07-nyc-192.168.110.67	enabled	4200070001	192.168.110.1/24	192.168.110.67/24	26s
sandbox9-srv08-nyc-192.168.110.68	enabled	4200070002	192.168.110.1/24	192.168.110.68/24	26s

You might notice that peering neighbor AS is different from Calico's default 64512. The is because the Netris Operator is setting a particular AS number for each node.

Allow up to 1 minute for the BGP sessions to come up, then check BGP resources again:

```
kubectl get bgp
```

As seen our BGP peers are established:

NAME	STATE	BGP STATE	LOCAL ADDRESS	REMOTE ADDRESS	AGE
PORT	STATE	NEIGHBOR AS			
isp2-customer	enabled	bgp: Established; prefix: 28; time: 0:07:48	50.117.59.118/30	50.117.59.117/30	8m41s
0:07:48	UP	65007			
sandbox9-srv06-nyc-192.168.110.66	enabled	4200070000	192.168.110.1/24	192.168.110.66/24	68s
sandbox9-srv07-nyc-192.168.110.67	enabled	4200070001	192.168.110.1/24	192.168.110.67/24	68s
sandbox9-srv08-nyc-192.168.110.68	enabled	4200070002	192.168.110.1/24	192.168.110.68/24	68s

Now let's check if `nodeToNodeMeshEnabled` is still enabled:

```
kubectl get bgpconfigurations default -o yaml
```

It is disabled, which means the "netris-calico" integration process is finished:

```

apiVersion: crd.projectcalico.org/v1
kind: BGPConfiguration
metadata:
  annotations:
    ...

```

```

  manage.k8s.netris.ai/calico: "true"
  ...
  name: default
  ...
spec:
  asNumber: 64512
  nodeToNodeMeshEnabled: false

```

Note Netris Operator won't disable Node-to-Node mesh until k8s cluster all nodes' BGP peers are being established.

Finally, let's check if our earlier deployed "Podinfo" application is still working when Calico Node-to-Node mesh is disabled:

```
curl 50.117.59.202
```

Yes, it works:

```
{
  "hostname": "podinfo-576d5bf6bd-mfpdt",
  "version": "6.0.0",
  "revision": "",
  "color": "#34577c",
  "logo":
  "https://raw.githubusercontent.com/stefanprodan/podinfo/gh-pages/cuddle_clap.gif",
  "message": "greetings from podinfo v6.0.0",
  "goos": "linux",
  "goarch": "amd64",
  "runtime": "go1.16.5",
  "num_goroutine": "8",
  "num_cpu": "4"
}
```

23.4.1 Disabling Netris-Calico Integration

To disable "Netris-Calico" integration, delete the annotation from Calico's bgpconfigurations resource:

```
kubectl annotate bgpconfigurations default manage.k8s.netris.ai/calico-
```

or change its value to "false".

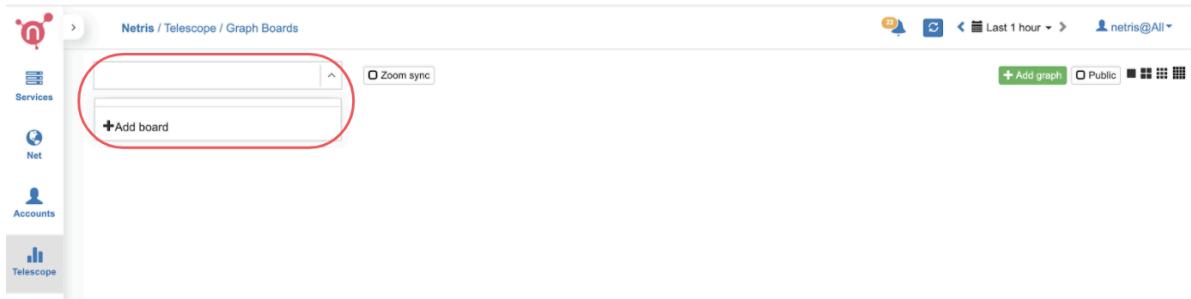
Visibility (Telescope)

24.1 Graph Boards

You can create custom graph boards with data sources available in different parts of the system. You can even sum multiple graphs and visualize them in a single view.

To start with Graph Boards, first, you need to add a new Graph Board.

1. Navigate to Telescope→Graph Boards, open the dropdown menu in the top left corner, then click +Add board.



2. Type a name and assign it to one of the tenants that you manage. Later on, you can optionally mark the Graph Board as public if you want the particular board to be visible to all users across multiple tenants.

The screenshot shows a modal dialog titled 'Create board'. Inside, there are two input fields: 'Name' containing 'my_graph_board' and 'Tenant' containing 'Admin'. Below the form are two buttons: 'Cancel' and a blue 'Add' button.

Now you can add graphs by clicking +Add graph.

Description of +Add graph fields:

- **Title** - Title for the new graph.
- **Type** - Type of data source.

- Bps - Traffic bits per second.
- Pps - Traffic packets per second.
- Errors - Errors per second.
- Optical - Optical signal statistics/history.
- MAC Count - History of the number of MAC addresses on the port.
- **Function** - Currently, only summing is supported.
- **+Member** - Add data sources by service (E-BGP, V-NET, etc..) or by Switch Port.

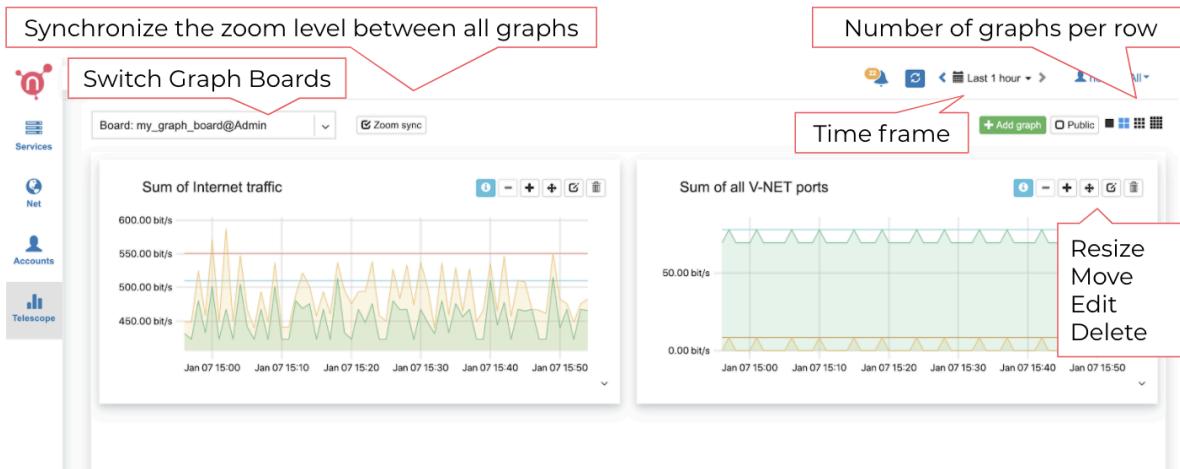
Example: Sum of traffic on two ISP(Iris1 + Iris2) links.

Title*	Sum of Internet traffic	
Type*	Bps	<input type="button" value="▼"/>
Function*	Sum	<input type="button" value="▼"/>
+Member <input type="button" value="▼"/>		
BGP bps	Iris1	<input type="button" value="Delete"/>
BGP bps	Iris2	<input type="button" value="Delete"/>

Example: Sum of the traffic on all ports under the service called “my V-NET”

Title*	Sum of all V-NET ports	
Type*	Bps	<input type="button" value="▼"/>
Function*	Sum	<input type="button" value="▼"/>
+Member <input type="button" value="▼"/>		
V-Net bps	my V-NET	<input type="button" value="Delete"/>

Screenshot: Listing of a Graph Board with the explanation of the controls.



24.2 API Logs

Comprehensive logging of all API calls sent to Netris Controller with the ability to search by various attributes, sort by each column, and filter by method type.

24.3 Dashboard

Netris, besides automatic configuration, also provides automatic monitoring of the entire network without the need for configuration of the monitoring systems.

Telescope→Dashboard summarizes Network Health, which can also be accessed by clicking on the Netris icon in the top left corner.

Description of the pie charts.

- **Hardware Health** - summary of CPU, RAM, disk utilization. Statuses of power supplies, fans, temperature sensors, critical system services, and time synchronization. Statuses of switch port link, utilization, optical signal levels, and BGP sessions.
- **E-BGP** - Statuses of external BGP sessions.
- **LB VIP** - Statuses of Load Balancer frontend / VIP availability.
- **LB Members** - Statuses of Load Balancer backend members.

By clicking on each title you can see the details of the checks on the right side.

Screenshot: Dashboard showing details of “Hardware Health.”

Monitoring summary grouped by type. Click on title to see details on the right column.

Filter by status: OK, Warning, Critical.

Name	Check name	Duration	Last check	Message
controller	check_port	12h 47m 23s	7/Jan/2021 17:41	swp10 port is UP, 0% RX Utilized of 1 Gbps, 0% TX Utilized of 1 Gbps
leaf1	check_port	12h 47m 23s	7/Jan/2021 17:41	swp11 port is DOWN, 0% RX Utilized of 1 Gbps, 0% TX Utilized of 1 Gbps
spine1	check_port	12h 47m 23s	7/Jan/2021 17:41	swp12 port is DOWN, 0% RX Utilized of 1 Gbps, 0% TX Utilized of 1 Gbps
spine1	check_port	12h 47m 23s	7/Jan/2021 17:41	swp13 port is DOWN, 0% RX Utilized of 1 Gbps, 0% TX Utilized of 1 Gbps
spine1	check_port	12h 47m 23s	7/Jan/2021 17:41	swp14 port is DOWN, 0% RX Utilized of 1 Gbps, 0% TX Utilized of 1 Gbps

Port up/down state can be set to “Save as normal.” So the system will alarm only if the actual state is different from the saved as the normal state.

Screenshot: “Save as normal” on selected ports.

Save as normal

Name	Check name	Duration	Last check	Message
controller	check_port	1h 45m 56s	29/Dec/2020 15:16	swp10 port is DOWN
spine1	check_port	1h 45m 56s	29/Dec/2020 15:16	swp11 port is DOWN
spine1	check_port	1h 45m 56s	29/Dec/2020 15:16	swp12 port is DOWN
spine1	check_port	1h 45m 56s	29/Dec/2020 15:16	swp13 port is DOWN
spine1	check_port	1h 45m 56s	29/Dec/2020 15:16	swp14 port is DOWN
spine1	check_port	1h 45m 56s	29/Dec/2020 15:16	swp15 port is DOWN
spine1	check_port	1h 45m 56s	29/Dec/2020 15:16	swp16 port is DOWN
spine1	check_port	1h 45m 56s	29/Dec/2020 15:16	swp17 port is DOWN

Accounts

The accounts section is for the management of user accounts, access permissions, and tenants.

25.1 Users

Description of User account fields:

- **Username** - Unique username.
- **Full Name** - Full Name of the user.
- **E-mail** - The email address of the user. Also used for system notifications and for password retrieval.
- **E-mail CC** - Send copies of email notifications to this address.
- **Phone Number** - User's phone number.
- **Company** - Company the user works for. Usually useful for multi-tenant systems where the company provides Netris Controller access to customers.
- **Position** - Position within the company.
- **User Role** - When using a User Role object to define RBAC (role-based access control), Permissions Group and Tenant fields will deactivate.
- **Permission Group** - User permissions for viewing and editing parts of the Netris Controller. (if User Role is not used)
- **+Tenant** - User permissions for viewing and editing services using Switch Port and IP resources assigned to various Tenants. (if User Role is not used)

Example: Creating a user with full access to all sections of Netris Controller, read-only access to resources managed by any Tenant, and full access to resources assigned to the Tenant Admin.

Username* acme

Full Name Acme

E-mail* acme@netris.ai

E-mail CC

Phone Number

Company Netris

Position

User Role * Permit All

Permission Group * Permit All

+Tenant

Cancel Add

Read-only access to resources of any Tenant

Full access to resources of Tenant Admin

Password: To set a password or email the user for a password form, go to the listing of usernames and click the menu on the right side.

Example: Listing of user accounts.

Username	Full Name	User Role	Permission Gr...	Tenants	Details	Created Date
acme	Acme	Permit All	<input type="checkbox"/> Edit All Tenants <input checked="" type="checkbox"/> Edit Admin	<input type="checkbox"/> Edit All Tenants <input checked="" type="checkbox"/> Edit Admin	<input type="checkbox"/> Edit Copy Link Set Password Send Password recovery Delete	2023-07-10
netris		Permit All	<input checked="" type="checkbox"/> Edit All Tenants	<input checked="" type="checkbox"/> Edit All Tenants	<input type="checkbox"/> Edit Copy Link Set Password Send Password recovery Delete	2023-07-10

25.2 Tenants

IP addresses and Switch Ports are network resources that can be assigned to different Tenants to have under their management. Admin is the default tenant, and by default, it owns all the resources. The concept of Tenants can be used for sharing and delegation of control over the network resources, typically used by network teams to grant access to other teams for requesting & managing network services using the Netris Controller as a self service portal or programmatically (with Kubernetes CRDs) as part of DevOps/NetOps pipeline.

A Tenant has just two fields, the unique name and custom description.

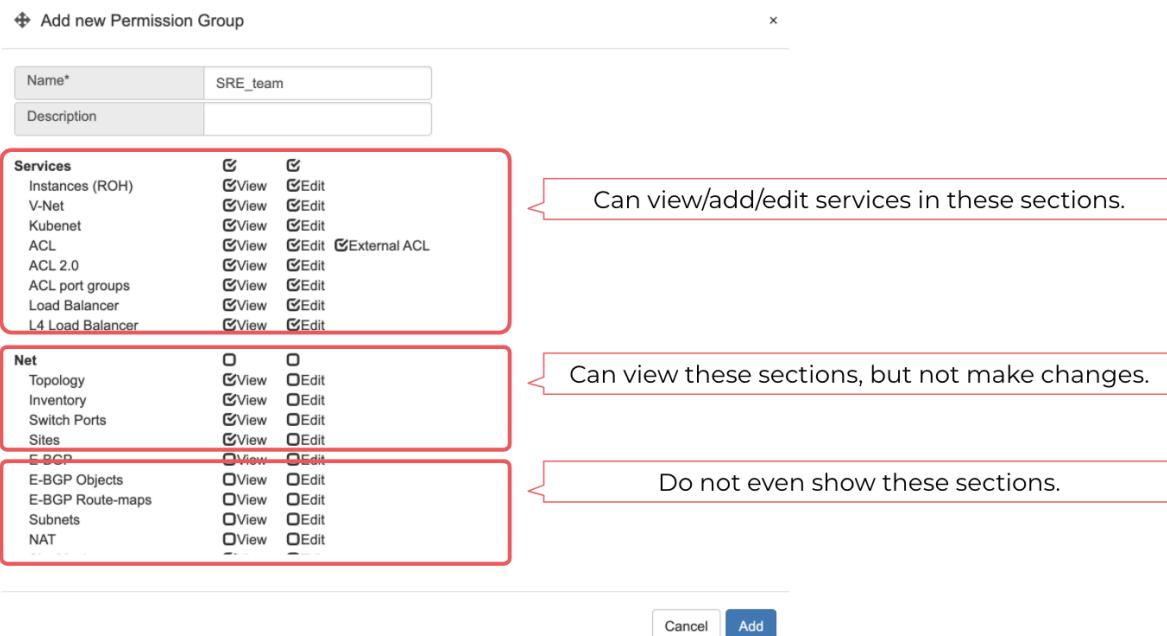
Example: Adding a tenant.

A screenshot of a "Add new tenant" dialog. It has fields for "Name*" (SRE_team) and "Description" (Tenant for the SRE team's resources). Buttons at the bottom are "Cancel" and "Add".

25.3 Permission Groups

Permission Groups are a list of permissions on a per section basis that can be attached individually to a User or a User Role. Every section has a View and Edit attribute. The view defines if users with this Permission Group can see the particular section at all. Edit defines if users with this Permission Group can edit services and policies in specific sections.

Example: Permission Group.

A screenshot of a "Add new Permission Group" dialog. It has fields for "Name*" (SRE_team) and "Description". Below is a table of permissions:

	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Services	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Instances (ROH)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
V-Net	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Kubenet	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ACL	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ACL 2.0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ACL port groups	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Load Balancer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
L4 Load Balancer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Net	<input type="checkbox"/>	<input type="checkbox"/>
Topology	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Inventory	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Switch Ports	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sites	<input checked="" type="checkbox"/>	<input type="checkbox"/>
E-BGP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
E-BGP Objects	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
E-BGP Route-maps	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Subnets	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
NAT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Annotations explain the permissions:
 - Services section: "Can view/add/edit services in these sections."
 - Net and E-BGP sections: "Can view these sections, but not make changes."
 - Subnets and NAT sections: "Do not even show these sections."

Buttons at the bottom are "Cancel" and "Add".

25.4 User Roles

Permission Groups and Tenants can be either linked directly to an individual username or can be linked to a User Role object which then can be linked to an individual username.

❖ Add new User Role x

Name*	SRE_users
Description	
Permission Group*	SRE_team ?

+ Tenant ?

<input type="checkbox"/>	<input checked="" type="checkbox"/> Edit SRE_team 	Delete	Full access to resources of Tenant SRE_team
<input type="checkbox"/>	<input type="checkbox"/> Edit Admin 	Delete	Read-only access to resources of Tenant Admin

Cancel Add

Release notes

- DPDK data plane support for SoftGate nodes. - Provides higher SoftGate performance. Up to 27Mpps, 100Gbps for L3 routing, 12Mpps with NAT rules on.
- L4 Load Balancer. - In addition to switch-based Anycast Load Balancer, we now support a Soft-Gate/DPDK-based L4 Load Balancer. L4LB integrates with Kubernetes providing cloud-like load balancer service (type: load-balancer).
- Kubenet - a network service purpose-built for Kubernetes cluster nodes. Kubenet integrates with Kube API to provide an on-demand load balancer and other Kubernetes specific networking features. Netris Kubenet is designed to complement Kubernetes CNI networking with modern physical networking.
- API logs - Comprehensive logging of all API calls sent to Netris Controller with the ability to search by various attributes, sort by each column, and filter by method type.
- Site Mesh - a Netris service for automatically configuring site-to-site interconnect over the public Internet. Site Mesh supports configuration for WireGuard to create encrypted tunnels between participating sites and automatically generates configuration for FRR to run dynamic routing. In a few clicks, services in one site get connectivity to services in other sites over a mesh of WireGuard tunnels.
- Ubuntu/SwitchDev updates - Removed the requirement for a hairpin loop cable. Removed the need for IP address reservation for V-NET, switching entirely to the anycast default gateway.
- Controller distributions - Netris controller, is now available in three deployment forms. 1) On-prem KVM virtual machine. 2) Kubernetes application. 3) Managed/Hosted in the cloud.
- Inventory Profiles - A construct for defining access security, timezone, DNS, NTP settings profiles for network switches and SoftGate nodes.
- Switch/SoftGate agents - New installer with easy initial config tool. Support for IP and FQDN as a controller address. Authentication key.
- GUI - Improved Net→Topology section, becoming the main and required place for defining the network topology. All sections got a column organizer, so every user can order and hide/show columns to their comfort.

SoftGate Performance

The following tested results are offered to help properly size the hardware needed for a SoftGate with various types of services:

Table 27.1. SoftGate Performance

	Routing with unmatched DNAT rules (mpps)	Routing with unmatched DNAT and SNAT rules (mpps)		DNAT	SNAT	DNAT with unmatched SNAT rules
Xeon E5-2660 CPU cores: (6 core for fw) Scheduler 0 Forward 1-7 KNI 8-10 OS 11-15	~27mpps	~17mpps	7mpps	9mpps	~2mpps	
Xeon Gold 6130 CPU cores: (6 core for fw) Scheduler 0 Forward 3-13(odd numbers) KNI 15 17 19	~31mpps	~22mpps	~15mpps	~11mpps	~3mpps (~3M conntrack count) ~7mpps (~20k conntrack count)	~11mpps
Xeon Gold 6130 CPU cores: (8 core for fw) Scheduler 0 Forward 3-17(odd numbers) KNI 19 21 23	~38mpps	~28mpps	19mpps	12mpps		~12mpps

