TECHNICAL WHITE PAPER - **AUGUST 2018**

PAS AND PKS WITH VMWARE NSX-T DATA CENTER

Reference Design Guide

1 Purpose

This reference design guide is tuned to cater to the requirements of Pivotal Cloud Foundry (PCF) running on VMware NSX-T Data Center covering both Pivotal Application Service (PAS) and Pivotal Container Service (PKS). As such, only the components and details of NSX-T Data Center relevant to running PCF are addressed in this design guide. Please refer to the NSX-T Data Center Reference Design guide, posted on communities, for a more complete coverage of the subject and NSX-T Data Center in general for all applications.

2 A Brief History of "Agile and Cloud Native"

"In today's fast-paced, fiercely competitive world of commercial new product development, speed and flexibility are essential."

–Hirotaka Takeuchi and Ikujiro Nonaka

Agile development methods and continuous delivery are new terms solving decades old problem of application delivery lag. Application delivery lag is the problem of application development taking anywhere from 2 – 20 years. Often, business needs change and the application may not be useful or relevant by the time it's ready to be deployed resulting in the application being scrapped because of lack of methods to refine dynamically as business needs evolve. Methods to speed up application delivery process were being explored as far back as 80s. One such methodology, Scrum is a result of such exploration.

Over the years, across the board, one methodology that has helped solve the application delivery lag is to parallelize at every level. That is, at a process level, moving away from the highly sequential process of various development stages, i.e., "concept development, feasibility testing, product design, development process, pilot production, and final production", to parallel methods where all the process teams interact and refine the process in parallel. At the development level, moving away from large monolithic application models to smaller teams with smaller goals. Microservices architectures are contrary to the traditional monolithic approach. In a microservices architecture, applications are deployed, upgraded, scaled, and restarted independently as multiple smaller services, each providing a business function.

In these new architectures, different teams work independently on different parts of the app using any programming language and libraries that they are most comfortable with or best suited for the job. In these smaller apps, a.k.a microservices, there is very little inter-process communication because these are not processes within a single monolithic app any more, but a number of small independent apps running within a container and using networks for communication.

Supporting multiple programming languages with diverse needs for frameworks and libraries needs a platform to abstract the underlying infrastructure.

Pivotal's Cloud Foundry (PCF), is a platform as a service that provides a platform to deploy these multitude of small applications without worrying about the runtime required to run these apps that may have been written in various languages with a variety of runtime requirements.

Pivotal's Cloud Foundry, by design, uses the micro-services architecture, one job per application deployed. Communication between the various components of an application is not just inter-process anymore but it would be done over a network.

PCF brings agile processes for application deployment to the compute layer. However, traditional networks fall far behind the demands of regular development lifecycles. Network and firewall changes are not trivial and could take weeks to months depending on how stringent the security policies of the concerned organization are. This in turn slows down the “agile” characteristics of PCF. To address the need for agile from network perspective, PCF has leveraged VMware’s NSX-T Data Center platform to provide on demand network services. NSX-T Data Center enables Network Platform operators to work in the same agile manner as the developers using the Platform.

3 PAS Overview

Prior to Pivotal Cloud Foundry, the closest “agile” mechanism required developers to not only develop their app but also to package it with the required runtimes to make sure it's ready for deployment. Linux containers enabled developers to create a lightweight package with the required runtime libraries. Containers are isolated environments provided by an operating system within which a microservice process is run. Basically, it's an operating system level virtualization. While containers helped, they also forced the developers to own not only the code they developed but the entire package which is encapsulated within the container. As such, the developers became the owners of the management of the package libraries shipped with the container that’s hosting the app for scenarios such as security and bug fixes etc., for the runtimes like for example JRE.

The PAS platform provides required application runtimes so the developer doesn’t have to package them with their applications. This allows for more scalable patching and scanning of the applications and their execution environments by Platform Operators.

PAS uses a separate isolated environment within a virtual machine for each instance of an application. These isolated environments are known as containers. Each container within PAS is used to only host one application instance. Again, a single application may have multiple instances. In which case, each of those instances of the application is hosted in a separate container. Similar to the micro-services architecture, there is no inter-process communication as such since these applications are all independent individual applications running within their own small blocked out area. Instead of inter-process communication, these applications communicate using standard REST API. Which means, every instance of an application within PCF ends up being a unique network endpoint identified by an IP and a port.

4 PAS Components

In the following section we will take a look at some of the components that go into a PAS install and how they relate to the network. To learn more about PAS in general, please check out their website: https://pivotal.io/learn

A single complete installation of PAS, with all the required components, is known as a PAS Foundation. Multiple Foundations are used for workload separation based on compliance requirements etc.,

4.1 Ops Manager

Ops Manager helps with the installation and configuration of Pivotal. BOSH is the first component, that's installed by Ops Manager. Do not worry if you are not aware of what BOSH is, it is covered a little later.

In Pivotal's current implementation, Ops Manager manages the state of PCF BOSH deployments, therefore it will override any manually modified BOSH deployment configuration. Pivotal recommends only interacting with BOSH directly when deploying BOSH add-ons, and troubleshooting with Pivotal Support.

4.2 Tiles

Ops Manager's Installation Dashboard shows all the components that are installed. Each one of these components is called a tile simply because they look like wall tiles. Tiles are used to install, configure and check status of the concerned component.

Based on the functionality provided and to make it simpler to understand, tiles may be categorized into two categories:

1. CF Tiles
2. Service Tiles

Note: This categorization is to help understand differentiate between PAS's built in tiles vs addon tiles. PAS itself does not have any such categories. Pivotal provides the core components of PAS and any PAS managed services as a tile.

4.3 CF Tiles

CF Tiles are various Cloud Foundry Pivotal Components that are part of the PAS foundation such as Bosh.

BOSH being a critical component of PAS, its pre-packaged and available with the Ops Manager for installation. In fact this is the only tile that is provided initially as all other tiles depend on the BOSH tile.

4.4 Service Tiles

Tiles may be used to install additional tools for the application instances such as access to a database like MySQL or messaging services such RabbitMQ. BOSH has a rich set of tiles to address various deployment requirements. These are known as service tiles.

There are two types of service tiles based on the method used for deployment

1. Pre-Provisioned
2. On Demand

Pre-provisioned service tiles are installed per PAS foundation and are shared across multiple application deployments. Based on the size of deployment, pre-Provisioned service may be deployed on a single large network or multiple smaller ones each addressing a different type of service. For example, a network for MySQL service and another for RabbitMQ

On Demand services are known as dynamic services within PAS. In this mode, services may be deployed on demand and are tied to a particular deployment. PAS creates each on-demand tiles when app developer creates a service instance. They are not shared across all the deployments within a PAS foundation. Hence, each deployment could potentially have a dedicated service and may end up needing a large subnet since the services would create additional VMs that would consume IPs - on Demand. Pivotal recommends a separate network for the dedicated services tile to help with future growth requirements.

4.5 BOSH

BOSH, a primary PAS tile, is an open source project that unifies release engineering, deployment, and lifecycle management of small and large-scale cloud software. Bosh was originally designed to deploy the open source Cloud Foundry.

BOSH is used within PAS to

1. Provision VMs
2. Deploy software to BOSH provisioned VMs
3. Monitoring
4. Recovery from failures
5. Software updates

Internally BOSH has the following components

4.5.1 Director

This is the core-orchestrating component, which is installed as a single VM and that controls VM creation and deployments as well as other software and service lifecycle events.

Either a Command line Interface (CLI) or Direct API may be used to interact with the Director.

4.5.2 Postgres DB

Director uses a Postgres database to store information about the desired state of deployment including information about stemcells, releases and deployments. This Postgres DB is internal to the Director VM.

4.5.3 Blobstore

Blobstore is basically a repository for any files, including large binary files. Director uses a blobstore to host the source code and related source and binaries used to compile and build the release. This is also an internal component of the Director VM and the releases that are built here are PAS components.

4.6 Stemcell

Stemcell is bare minimum operating system with a BOSH agent and few common utilities. By design, they are generic and do not contain anything specific to any type of programming language. Stemcells are used to capture and version operating system images. They are also easy to use across multiple IaaS platforms or VMs of different types.

Stemcells are used by BOSH to generate all the VMs within PAS. VMs used for everything including infra such as TCP Router and also the VMs used to host the containers such as Diego Cells (essentially the VMs that stage and run the application artifacts as application Instances or containers).

4.7 Release

A release is basically a versioned collection of configuration properties, configuration templates, startup scripts, source code, binary artifacts and anything else required to build and deploy software in a reproducible way.

PAS deploys the core components of Cloud Foundry and PAS using Releases for package management. Releases also provide an easy method for patching and keeping the PAS instances secure and up to date. While stemcells are generic, a release is very specific and provide all the software needed. For a given release, BOSH helps capture all needed configuration options and scripts for deployment, keeping track of dependencies and versioning,

Note: A release could be user provided code or any of the management and service related VMs such as MySQL servers for MySQL Tile.

4.8 Deployment

Deployment is the combination of the base operating system called the stemcell with the software that needs to be deployed called the release. In the case of the user code, this would also require making the configuration required for the GO routers etc.,

Deployments are hosted on vSphere infrastructure. Deployment could refer to the deploying a service such as MySQL or Diego Cells that would host containers with user deployed code.

The process for deploying user code is similar to the above process, with one key difference. When deploying user code, only a container is deployed inside an existing diego cell.

Diego cells will consume IPs from the Elastic Runtime subnet. Containers hosted within the Diego Cells, do not consume any IPs.

4.9 Diego Cells

Diego Cells are essentially the VMs that stage and run the application artifacts as application Instances or containers

4.10 Application Instance

In the PAS terminology, Application Instance (AI) is one copy of a user deployed application. A single application may have multiple instances for scale and redundancy purposes. Multiple copies of the same application are considered as multiple AIs. All the Application Instances of a given Application would only exist within a single PAS foundation. By default, communication is not allowed between the Application Instances. However, this behavior can be changed.

PAS applications are instantiated as one or more instances. PAS runs each of those application instances in its own dedicated container.

4.11 Gorouters

Gorouters, which is a router implemented in Go language and hence the name, helps route the incoming traffic to the appropriate component. This could be an operator addressing the cloud controller or an application user accessing an app running on a Diego Cell. Even though the name suggests some kind of routing, this is not a typical network router and does not do any network level routing.

4.12 Cluster

On vSphere, Pivotal uses the vSphere Cluster construct along with vSphere resource pools to deploy Availability Zones. Pivotal's ideal configuration is 3 vSphere compute clusters with 3 servers in each cluster. Pivotal creates 3 Availability Zones, each mapping to either one cluster or a subset of it using resource pools. Multiple Pivotal foundations could run on the same compute clusters using different availability zones.

Each PAS installation, containing a full deployment of the PAS, is known as a PAS foundation.

Pivotal's use of "3" clusters comes from the need to maintain quorum for key components, such as Galera and etcd, in the event of failure of one of the entities. A "3" node cluster can survive one node failure.

For sandbox type non-production clusters, one could potentially use just 3 servers with the following design options.

1. Three clusters with a single node for each cluster
2. Single cluster with all three nodes + enabling VMware DRS and HA

4.13 Multi-tenancy

In a typical enterprise application deployment, it is common to deploy application with segmentation or separation in mind. This can occur due to variety of reasons such as compliance, span of control, prod vs dev and security. One approach is to separate workload by using a separate PAS foundation.

5 NSX-T Data Center Container Plugin

PAS is fully integrated with NSX-T Data Center via the NSX-T Data Center Container Plugin (NCP). In the following section we take a look at the various components of NCP and how it integrates with PAS.

5.1 NCP Tile

NCP is a software component provided by VMware in the form of a BOSH add-on release and is installed as a PCF Ops Manager tile.

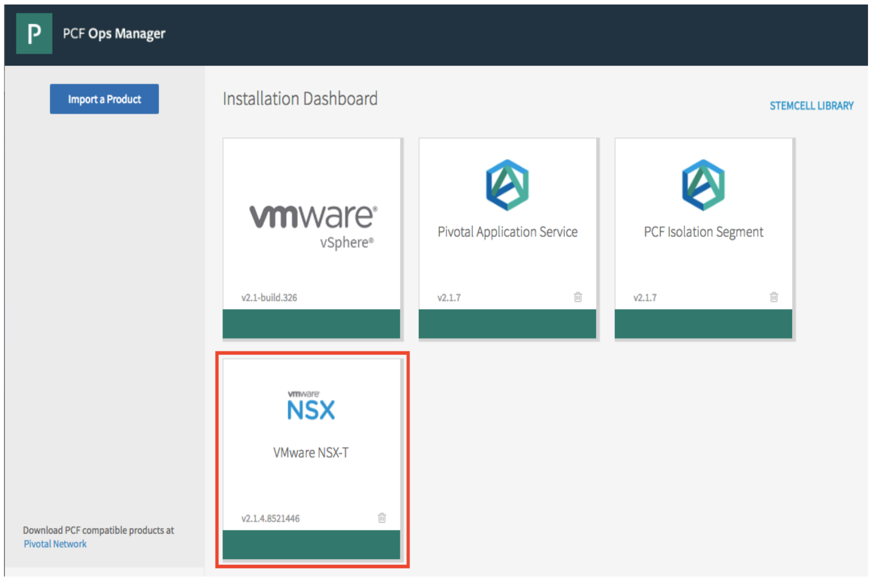


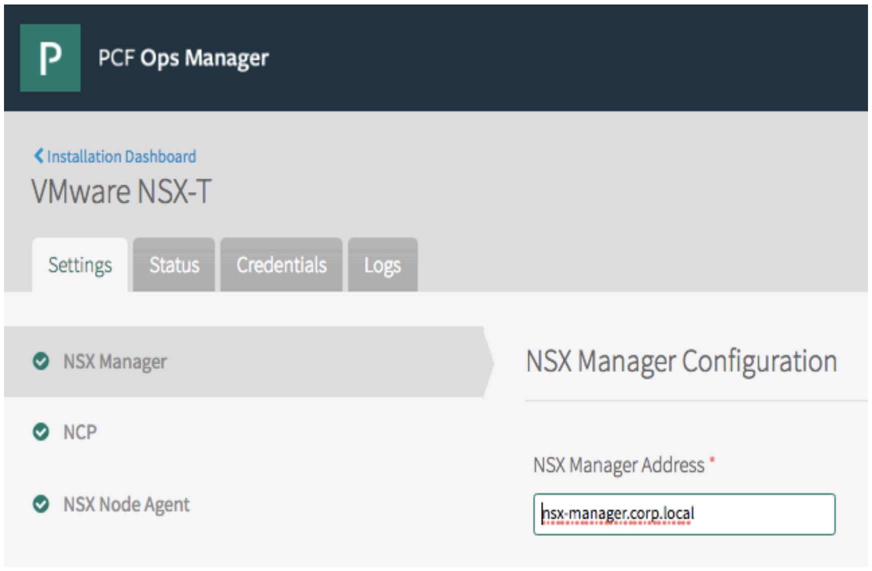
Figure 1 NSX-T Data Center Tile

Tile has 3 pages that focus on

1. NSX-T Data Center Manager details
2. Object references to be used within NSX-T Data Center Manager – for example the T0 router to be used for the current installation
3. Enabling logging

5.1.1 NSX-T Data Center Manager Details

NSX-T Data Center Manager page is used to provide the NSX-T Data Center manager address and access details. The following image shows a portion of the configuration. Note: The DNS name provided in the NSX-T Data Center Manager Address field needs to match the actual cert that's generated on the NSX-T Data Center manager.



NSX Manager Address *

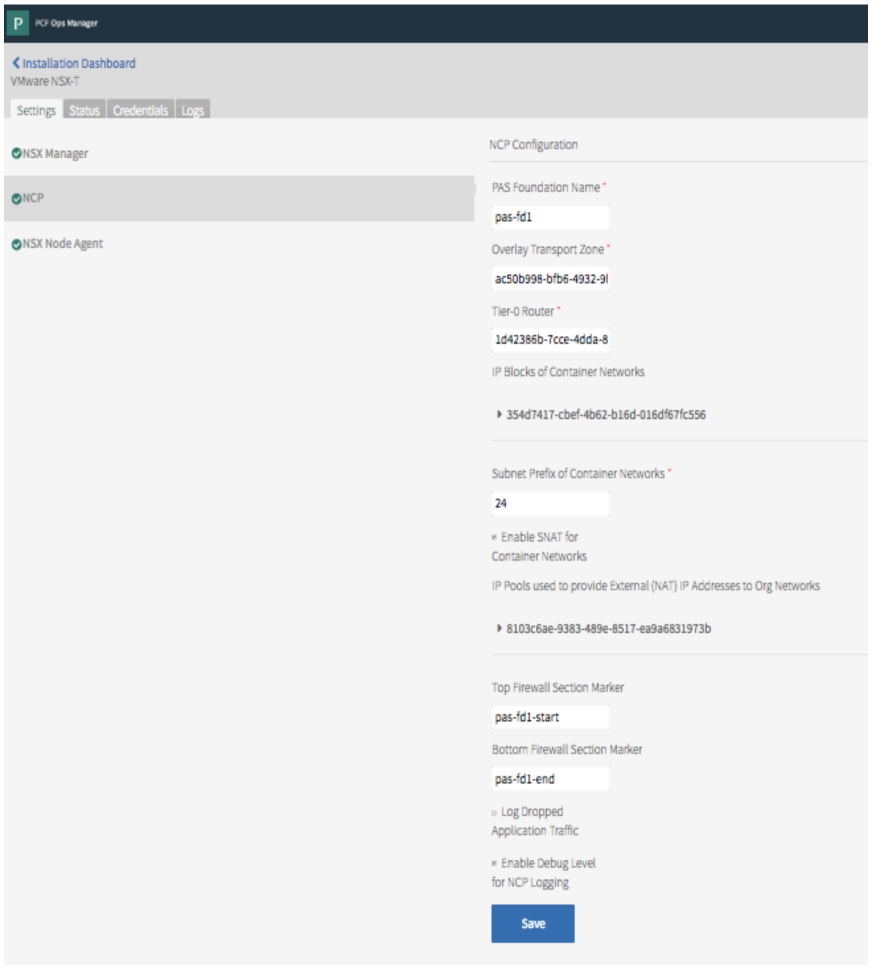
nsx-manager.corp.local

Figure 2 NSX-T Data Center Tile – NSX-T Data Center Manager Related Details

5.1.2 NCP

NCP page of the NSX-T Data Center Manager tile provides references to the various objects used by PAS when provisioning containers and setting up policy. This includes

1. T0 router associated with the specific PAS instance
2. Overlay Transport Zone
3. IP Blocks used for the container networks
4. SNAT pools if used
5. Firewall sections markers – which specify exactly where to place a particular foundation related policy within the full firewall rule table


PCF Ops Manager

Installation Dashboard
VMware NSX-T

Settings | Status | Credentials | Logs

NSX Manager
NCP
NSX Node Agent

NCP Configuration

PAS Foundation Name *
pas-fd1

Overlay Transport Zone *
ac50b998-bfb6-4932-9l

Tier-0 Router *
1d42386b-7cce-4dda-8

IP Blocks of Container Networks
354d7417-cbef-4b62-b16d-016df67fc556

Subnet Prefix of Container Networks *
24

Enable SNAT for Container Networks

IP Pools used to provide External (NAT) IP Addresses to Org Networks
8103c6ae-9383-489e-8517-ea9a6831973b

Top Firewall Section Marker
pas-fd1-start

Bottom Firewall Section Marker
pas-fd1-end

Log Dropped Application Traffic

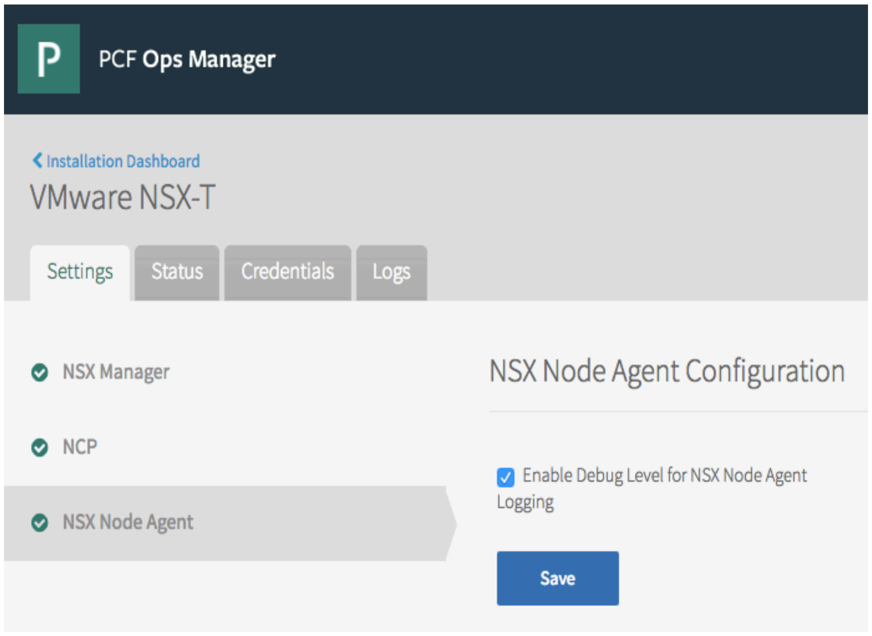
Enable Debug Level for NCP Logging

Save

Figure 3 NSX-T Data Center Tile - NCP Related Details

5.1.3 NSX-T Data Center Node Agent

Last page simply allows user to specify whether logging needs to be enabled:



☑ Enable Debug Level for NSX Node Agent Logging

Figure 4 NSX-T Data Center Tile – NSX-T Data Center Node Agent Logging

NCP is deployed as a pair of HA VMs as part of PAS.

Following silk related processes are removed from the diego cells:

1. iptables-logger
2. netmon
3. silk-deamon
4. vxlan-policy-agent

and instead the following NSX-T Data Center related processes are installed and started:

1. Two OpenVSwitch user space deamons

    a. ovsdb-server
    b. ovs-vswitchd
2. NSX-Node-Agent

5.2 NCP Architecture

NCP is a modular, adapter based software that can be extended to support various CaaS and Paas Platforms. As shown in the following image, Cloud Foundry Adapter is used to interact with PAS.

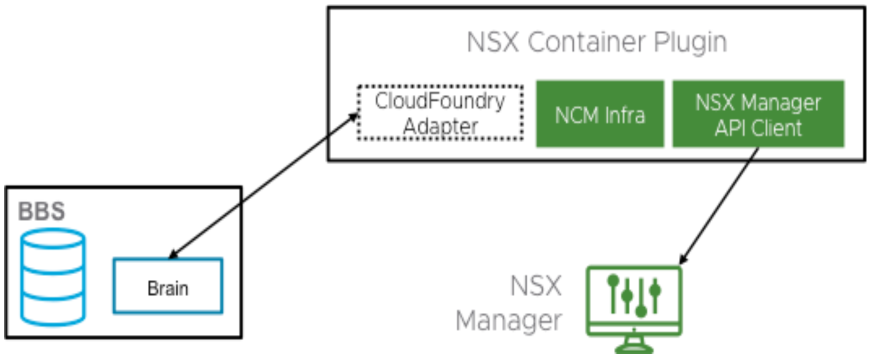


Figure 5 NCP Architecture

NCP, via CloudFoundry Adapter, listens to the BBS events for tasks and long running processes (LRP) and uses the NSX-T Data Center Manager API Client to create and configure the required infrastructure on network side.

Similarly, NCP also listens to policy related changes via the Policy server, triggered through “cf network policy…” and uses it to configure the right policy via the NSX-T Data Center Manager API Client.

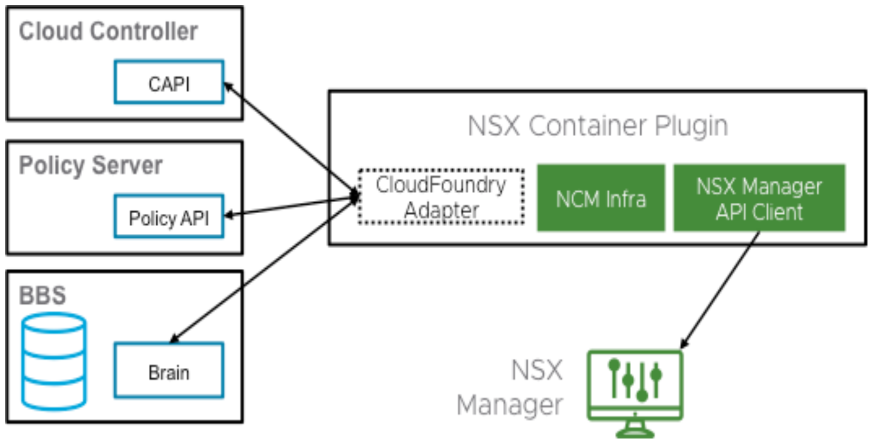


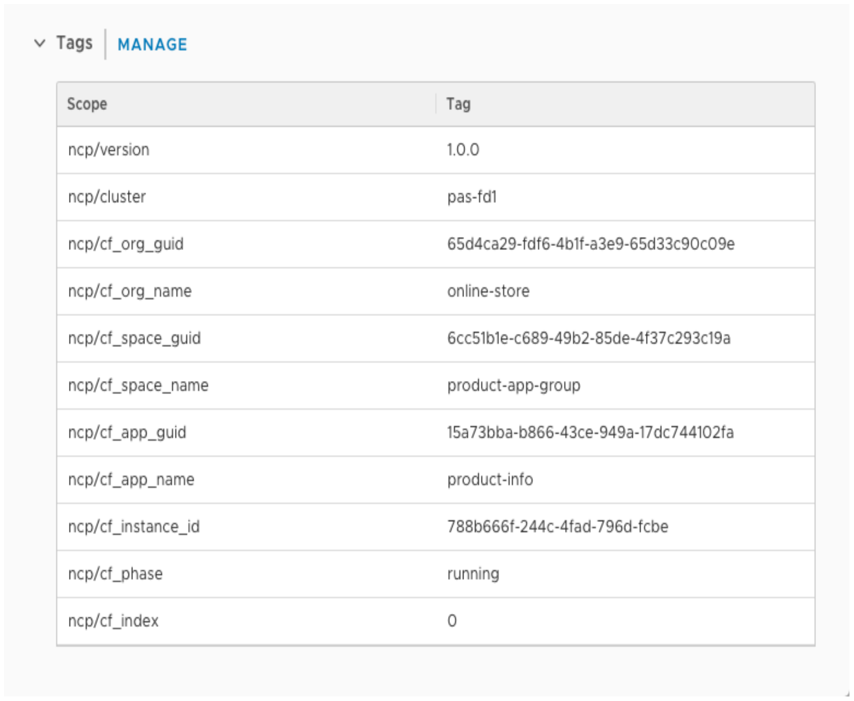
Figure 6 NCP Architecture - Cloud Foundry Adapter

5.3 NCP in Action – Deploying an Application

For example, when a user pushes a new Application, BBS announces new LRP. This will help NCP trigger a chain of actions on the NSX-T Data Center front that results in creating

1. T1 router for the Org that the App was pushed from, if it does not exist. This would be case if this is the very first App pushed within that Org. This is because; T1 Routers are not created on creation of Orgs and Spaces but on pushing an App in that Org.
2. Request an
    a. IP from the subnet allocated to that specific org
    b. MAC from the container MAC pool in NSX-T Data Center
    c. And picks a VLAN to assign for the Application
3. Once the above information is available, NCP then creates a logical port on the T1 associated with the Org that the app was pushed from and applies the IP, MAC and VLAN to the logical port
4. NCP also adds metadata such as PAS foundation name, Org Name, Space Name and Application

Name etc., as tags to the logical port. This tags help uniquely identify an application and are useful when constructing NSGroups for security posture. Following image shows various tags that are applied:



Tags MANAGE

Scope	Tag
ncp/version	1.0.0
ncp/cluster	pas-fd1
ncp/cf_org_guid	65d4ca29-fdf6-4b1f-a3e9-65d33c90c09e
ncp/cf_org_name	online-store
ncp/cf_space_guid	6cc51b1e-c689-49b2-85de-4f37c293c19a
ncp/cf_space_name	product-app-group
ncp/cf_app_guid	15a73bba-b866-43ce-949a-17dc744102fa
ncp/cf_app_name	product-info
ncp/cf_instance_id	788b666f-244c-4fad-796d-fcbe
ncp/cf_phase	running
ncp/cf_index	0

Figure 7 NCP Generated Application Tags

5. After the logical port is created, NSX-T Data Center Manger triggers NSX-T Data Center Local Control Pane (LCP) via NSX-T Data Center Controllers to create a logical port (LP) on the hypervisor. NSX-T Data Center LCP is a process running on the hypervisor.

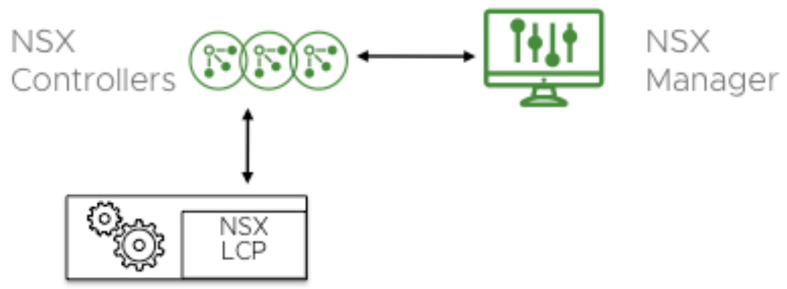


Figure 8 NSX-T Data Center LCP

6. NSX-T Data Center Hyperbus monitors the NSX-T Data Center LCP for new Container Interfaces (CIF) and its ID, MAC, IP and VLAN binding.

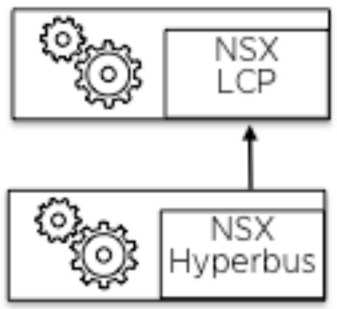


Figure 9 NSX-T Data Center - LCP to Hyperbus Communication

7. NSX-T Data Center Hyperbus triggers the Garden CNI which in turn executes the NSX-T Data Center CNI plugin to create the actual network wiring for the application. NSX-T Data Center CLI Plugin uses the CIFs ID, IP, MAC and VLAN binding data from the NSX-T Data Center Hyperbus over a isolated and secured channel that only allows one way connection from the hypervisor to the VM. The VLAN Tag is only relevant from inside the Application Instance to the Hypervisor. Once the packet from the Application Instance reaches the Hypervisor, VLAN tag is removed. Policy is applied based on the desired security posture and the packets are sent out towards

the LS and the T1 associated with the Org that app is pushed to.

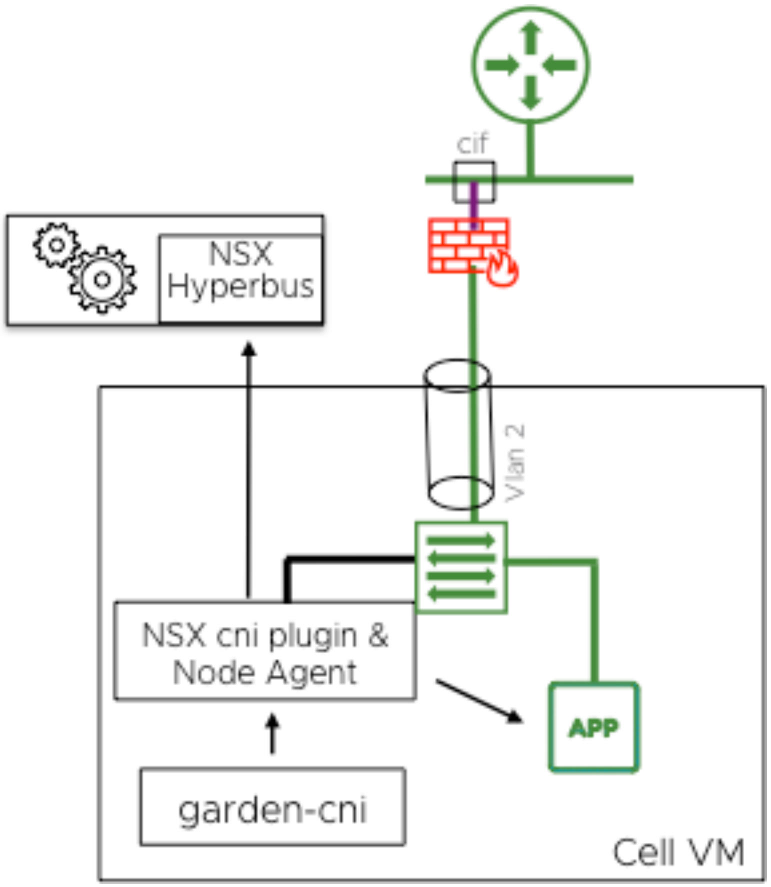


Figure 10 Container Port Attachment Workflow

Following image shows the full path of actions taken including the position of each element within the infra – hypervisor vs Cell VM:

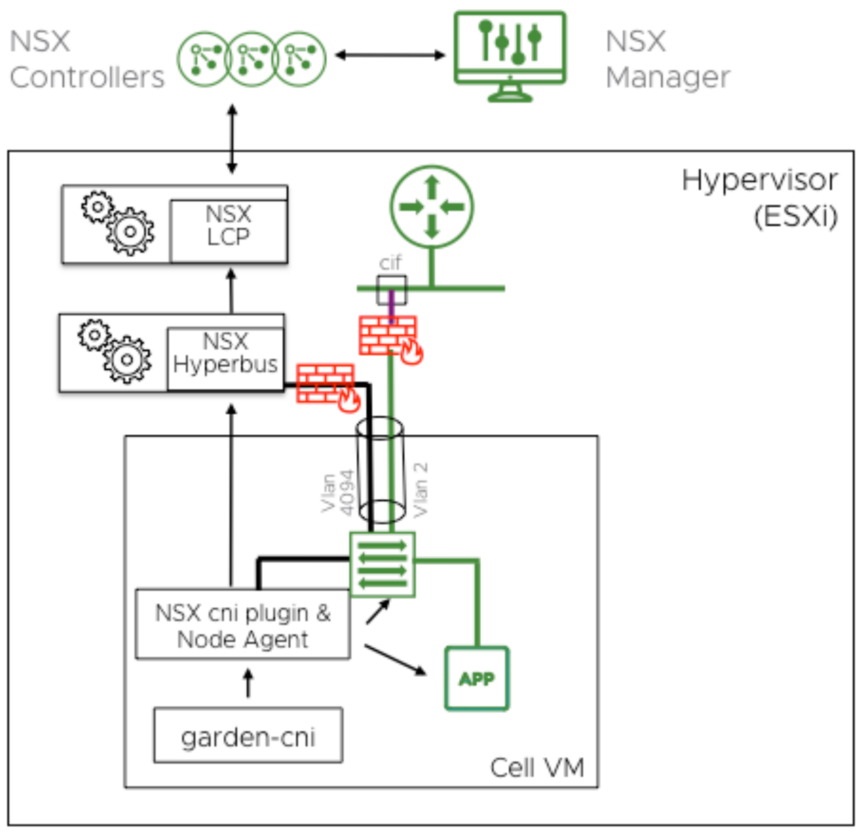


Figure 11 Container Logical Port Attachment Workflow - 2

8. Once the above actions are complete, Garden CNI is unblocked and the Application push succeeds.

6 PAS on NSX-T Data Center – Building Blocks

In this section we will take a look at two kinds of building blocks that play a role in design considerations.

- **Virtual Infrastructure**: Placement of virtual infrastructure components such as the vCenter, NSX-

T Data Center Manager, BOSH, OpsManager and PAS workload related virtual machines
- **Logical Infrastructure**: Design considerations around logical switching, routing, firewalls, NAT and Load Balancer.

6.1 NSX-T Data Center Management and Edge Design Considerations

The following section shows the basic building blocks of PAS running on NSX-T Data Center. Some of these components will be separated into a separate block based on hardware resource availability.

Following are some of the key blocks that are part of PAS on NSX-T Data Center running vSphere infrastructure:

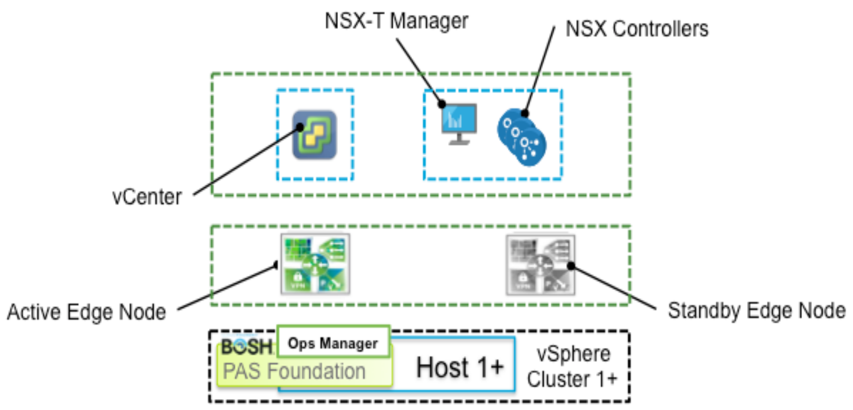


Figure 12 PAS on NSX-T Data Center - Base Components

Simplified PAS on NSX-T Data Center - components:

- vSphere Management: To manage the vSphere virtual infrastructure
- NSX-T Data Center Management: Containing NSX-T Data Center Manager and NSX-T Data Center Controllers
- Edge Node Cluster: Hosting T0 routers to provide North / South connectivity for the PAS Foundation
- Compute: Hosts used to run PAS Management components such as BOSH and Ops Manager and PAS workloads ie., infra to run the actual PAS driven Application Instances (AIs)

Recommendation is to have at least three clusters.

1. Mgmt Cluster: To host all the management related components of
    a. vSphere (vCenter)
    b. NSX-T Data Center (NSX-T Data Center Manager, NSX-T Data Center Controllers)

2. Edge Cluster: To host NSX-T Data Center Edges that provide north / south connectivity to the PAS Foundation. Based on performance requirements, this could either be a cluster of Virtual Edges or Bare Metal Edges.

3. Compute Cluster: To host the PAS Mgmt Components, BOSH and Ops Manager and Application Instances deployed with PAS. While a single Cluster would work – ideally should have 3 clusters with 3 or more hosts per cluster.

The following diagram shows a PAS deployment with the above components separated into their own clusters.

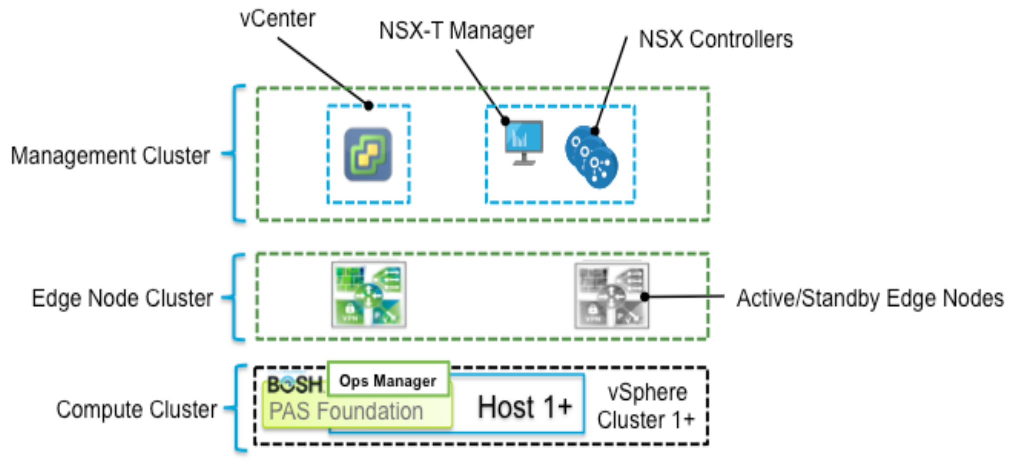


Figure 13 PAS on NSX-T Data Center – vSphere Clusters

6.1.1 Edge Size and Type Considerations

NSX-T Data Center provides two main variants of edge, which are: (Just recommend Virtual for PAS)

- Virtual Machine Edge: For most typical DC workloads, where 90% of packet sizes are over 1000 bytes, and cases where multi-tenancy is desired
- Bare Metal Edge: For enterprise workloads, which may require services such as NAT, firewall and Load Balancer at high performance levels.

VM Edge form factor further has 4 variants. Please refer to the NSX-T Data Center installation guide for details on various sizes. Small form factor Edge VM does not have enough resources to support a Load Balancer and hence is not recommended for PAS deployment. For more Load Balancer related considerations for when deciding on the Edge VM form factor size, please refer to the Load Balancer section of this document.

6.2 Logical Infrastructure

6.2.1 Switching

Pivotal's recommendation for traditional Elastic Run Time based architecture was to use 4 logical wires to host various components.

1. Management Components such as BOSH and Ops Manager
2. Elastic Run Time
3. Services such as MySQL
4. Dynamic Services that are deployed by developers

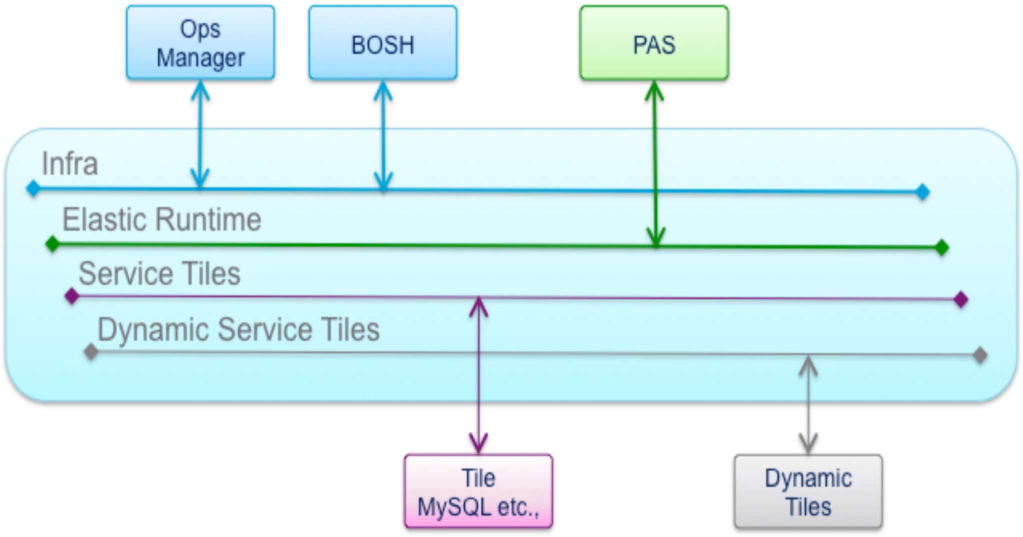


Figure 14 PAS on NSX-T Data Center - Logical Switch Requirements

With the NSX-T Data Center, Application Instances (containers) can connect directly on NSX-T Data Center logical switch ports. Multiple logical switches are used for PAS instead of the traditional single wire design that was used with

Elastic Run Time. This approach helps separate PAS workloads all the way down to the network level based on PAS's org. Each org gets its own logical wire, attached to an org specific t1 router.

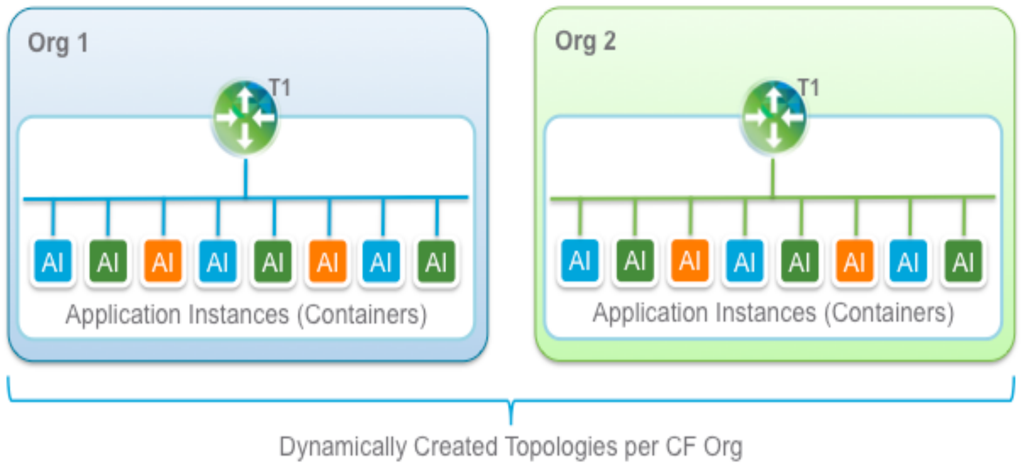

Org 1
T1
AI
Application Instances (Containers)
Org 2
T1
AI
Application Instances (Containers)
Dynamically Created Topologies per CF Org

Figure 15 T1 Router and Logical Switches for each Org

6.2.2 Routing

As mentioned earlier, each org has its own separate logical switch that's attached to a T1 router that's dedicated to that org. Every org in PAS will have a dedicated T1 router. Routing between the orgs and from PAS management components to workload components goes through the NSX-T Data Center T0 router.

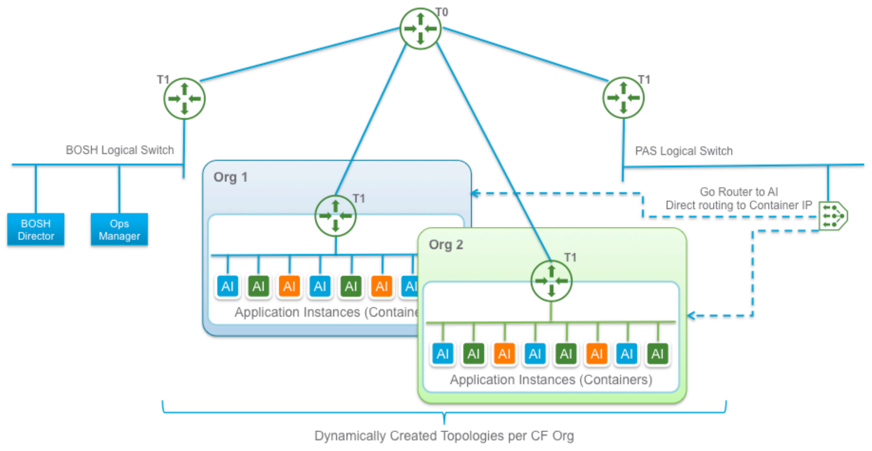

T0
T1
T1
BOSH Logical Switch
PAS Logical Switch
BOSH Director
Ops Manager
Org 1
T1
Go Router to AI
Direct routing to Container IP
Org 2
T1
AI
Application Instances (Containers)
Dynamically Created Topologies per CF Org

Figure 16 Routed Design

Fully routed designs are easier to implement compared to NAT. With NAT, due care must be taken to implement the NAT rules correctly to allow the right traffic into and out of the PAS foundation. More on NAT in the following section. Fully routed design also allows full visibility into all the applications at per application instance level even from outside NSX-T Data Center Domain.

6.2.3 NAT

NSX-T Data Center provides NAT as a standalone feature and also as part of the Load Balancer service.   The following section is focused on NAT feature. NAT runs on router node (T0 or T1) that it is connected to.

Network Address Translation (NAT) provides a mechanism to

1. Decrease the number of routable IPs required
2. Reuse IP address space across multiple PAS foundations

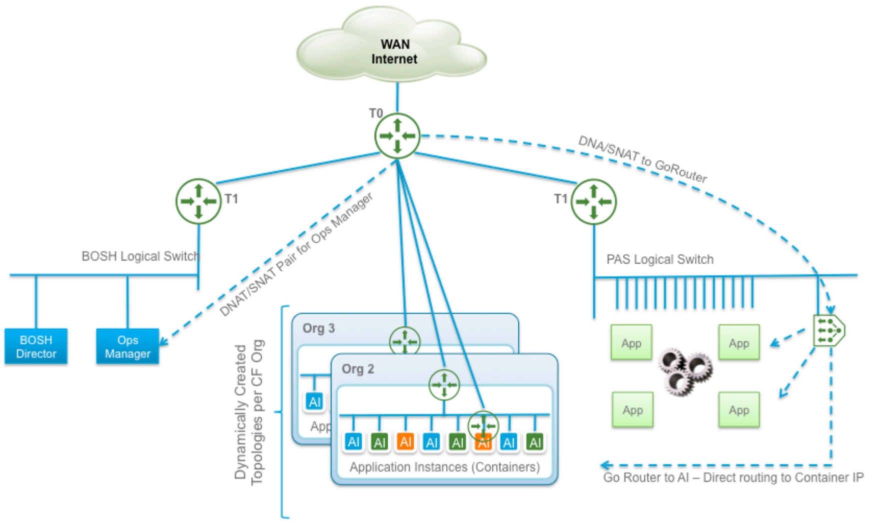

WAN Internet
T0
T1
T1
DNA/SNAT to GoRouter
DNAT/SNAT Pair for Ops Manager
BOSH Logical Switch
PAS Logical Switch
BOSH Director
Ops Manager
Dynamically Created Topologies per CF Org
Org 3
Org 2
AI
App
Application Instances (Containers)
Go Router to AI – Direct routing to Container IP

Figure 17 Design with NAT

NSX-T Data Center supports NAT on both T0 and T1 routers. Based on where NAT is implemented, due consideration should be given to how the IP space is being used/reused behind the NAT. Also, rules need to be configured to allow traffic in and out of the PAS foundation based on requirements. In this model, Ops Manager and the GoRouters will be reachable from the outside world via NAT'd IP address. There is no NAT within in the PAS infra, for communication between Ops Manager and Diego Cells etc., as they are all on routed NSX-T Data Center logical network. This helps avoid hair-pinning PAS internal traffic to the T0 Router instance running the stateful NAT service.

NAT cannot be used with an Active-Active Edge deployment as it is run as a stateful service. Hence, NSX-T Data Center Edge Cluster should be deployed in Active-Standby mode if planning to leverage NST-X NAT service.

6.2.4 Load Balancer

NSX-T Data Center, since release 2.1, comes with built-in Load Balancer functionality. NSX-T Data Center provides both L4 and SSL Termination capabilities.

With PAS, NSX-T Data Center's Load Balancer is used to load balance incoming requests to the set of GoRouters. SSL termination may also be done at the NSX-T Data Center load balancers.

Following image shows the placement of Load Balancer in PAS infrastructure.

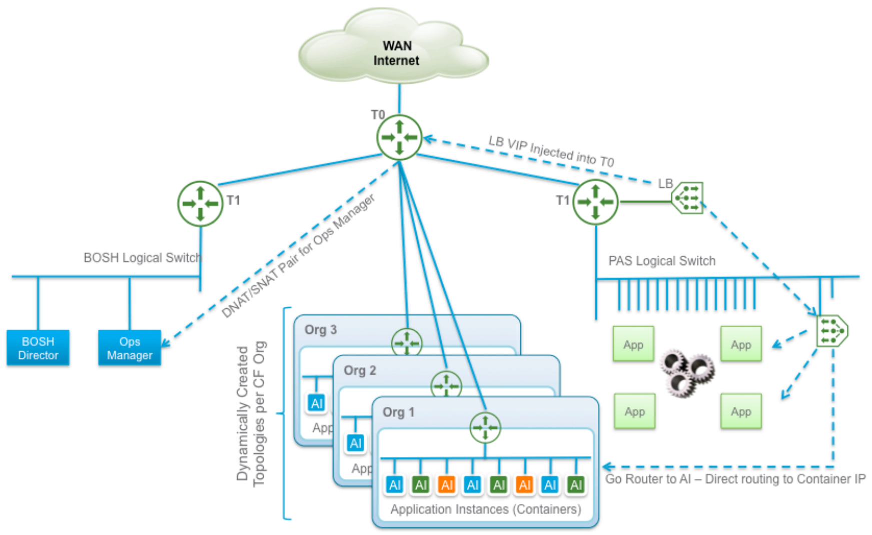

WAN
Internet
T0
LB VIP Injected into T0
LB
T1
T1
BOSH Logical Switch
PAS Logical Switch
DNAT/SNAT Pair for Ops Manager
BOSH Director
Ops Manager
Dynamically Created Topologies per CF Org
Org 3
Org 2
Org 1
AI
App
Application Instances (Containers)
Go Router to AI – Direct routing to Container IP

Figure 18 LB Design

Load Balancer cannot be used in Active-Active edge design as it is a stateful service. Hence, Edge's should be deployed in Active-Standby mode if planning to use Load Balancer.

SSL Termination

With PAS, SSL may be terminated at the Load Balancer or the GoRouters. If termination at the Load Balancer level is desired, then the following design of using a Edge Node cluster, helps achieve high SSL throughput.

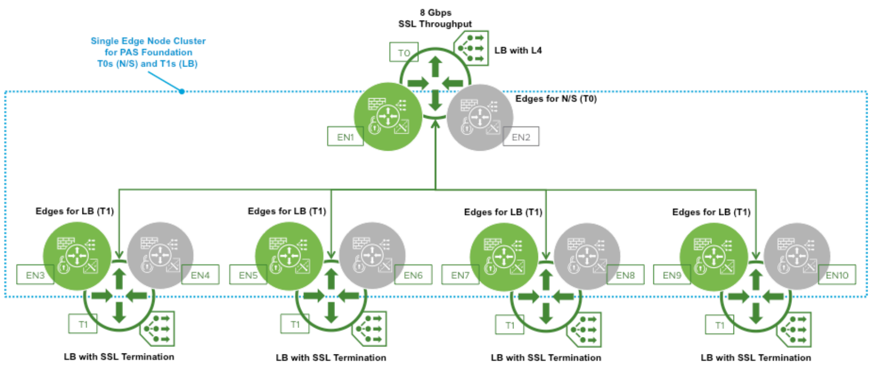


Figure 19 LB Design for High SSL Termination Throughput

In this design, one LB is used as a L4 Load Balancer to the other 4 LBs that are configured for SSL Termination.

6.2.5 Security

Security, traditionally, has been anything but agile. Hence, security in the PAS world is left at a bare minimum level, for example, securing only north/south traffic into PAS, for lack of better options. Often this translates to securing just the perimeter and not enforcing any further rules at the application layer.

NSX-T Data Center provides security at various layers with restrictions on the policy that can be applied based on users role. NSX-T Data Center enables traditional SecOps teams to setup umbrella policy and yet allow enough flexibility for the DevOps teams to open and close firewall ports as per their requirements.

6.2.5.1 NSGroups

NSX-T Data Center uses NSGroups to group applications. NSGroups are granular to the level of an Application in the PAS world, which is a set of Application Instances for a given application. Since NSGroups also allow-nesting, its possible to further group a bunch of similar applications into a larger set.

For example, the following image shows a group of applications that are related to Products. One of the Applications could be Product Info and another one Inventory. While they both may have their own independent security posture, they could also share common security posture because they are both also related to Products Application group.

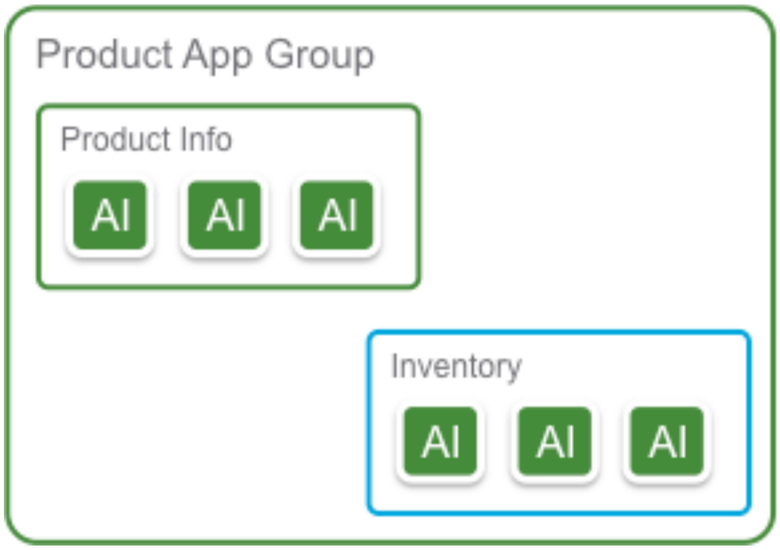
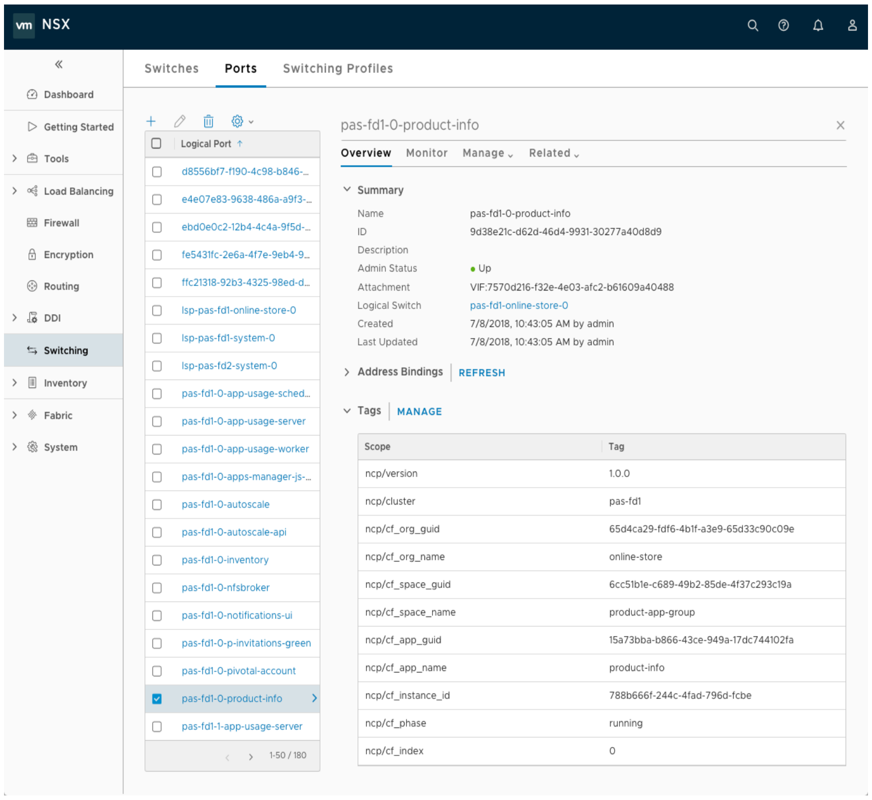


Figure 20 Example for an Org - Product App Group

6.2.5.2 Application Context

When applications are pushed via “cf push” in PAS, several tags are created to uniquely identify the application. For example, for the above Product Info, when the application is pushed in PAS, the following tags are created for the application's logical port in NSX.



Logical Port
d8556bf7-f190-4c98-b846-...
e4e07e83-9638-486a-a9f3-...
ebd0e0c2-12b4-4c4a-9f5d-...
fe5431fc-2e6a-4f7e-9eb4-9...
ffc21318-92b3-4325-98ed-d...
lsp-pas-fd1-online-store-0
lsp-pas-fd1-system-0
lsp-pas-fd2-system-0
pas-fd1-0-app-usage-sched...
pas-fd1-0-app-usage-server
pas-fd1-0-app-usage-worker
pas-fd1-0-apps-manager-js-...
pas-fd1-0-autoscale
pas-fd1-0-autoscale-api
pas-fd1-0-inventory
pas-fd1-0-nfsbroker
pas-fd1-0-notifications-ui
pas-fd1-0-p-invitations-green
pas-fd1-0-pivotal-account
✓ pas-fd1-0-product-info
pas-fd1-1-app-usage-server

1-50 / 180

pas-fd1-0-product-info

Summary

Name	pas-fd1-0-product-info
ID	9d38e21c-d62d-46d4-9931-30277a40d8d9
Description	
Admin Status	● Up
Attachment	VIF:7570d216-f32e-4e03-afc2-b61609a40488
Logical Switch	pas-fd1-online-store-0
Created	7/8/2018, 10:43:05 AM by admin
Last Updated	7/8/2018, 10:43:05 AM by admin

Address Bindings

Tags

Scope	Tag
ncp/version	1.0.0
ncp/cluster	pas-fd1
ncp/cf_org_guid	65d4ca29-fdf6-4b1f-a3e9-65d33c90c09e
ncp/cf_org_name	online-store
ncp/cf_space_guid	6cc51b1e-c689-49b2-85de-4f37c293c19a
ncp/cf_space_name	product-app-group
ncp/cf_app_guid	15a73bba-b866-43ce-949a-17dc744102fa
ncp/cf_app_name	product-info
ncp/cf_instance_id	788b666f-244c-4fad-796d-fcbe
ncp/cf_phase	running
ncp/cf_index	0

Figure 21 Application Tags

Tags include information such as the

- Cluster: pas-fd1 (PAS Foundation where the app was deployed)
- Org Name: online-store
- Space Name: product-app-group
- Application Name: product-info

Above tags are useful when creating NSGroups. Tags not only contain user readable names as listed above but also IDs. ID based tags such as the ncp/cf_app_guid are used by NCP when creating NSGroups dynamically on applying network policy via cf. More on this a little later in section on "On-Demand Security".

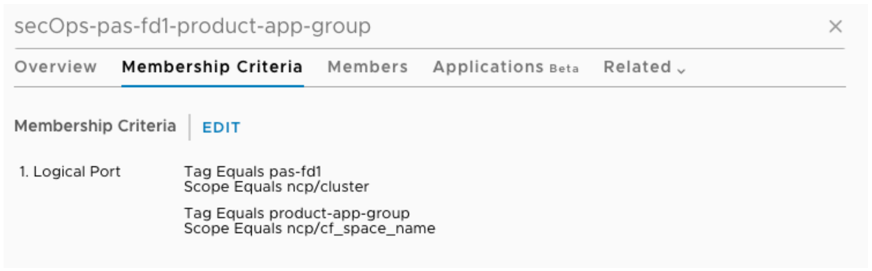
Granularity of the NSGroups may be as precise as application level.

6.2.5.3 Micro-segmentation Granularity

With NSX-T Data Center, micro-segmentation may be coarse-grained or fine-grained based on business needs. At a coarse level, applications may be grouped and security posture applied to the group as a whole. Or, based on business requirements, micro-segmentation may be applied at an application level.

6.2.5.3.1 Coarse-grained Micro-segmentation

NSGroup may be created to include all the apps within a App Group by using Space or Org related constructs. Following is an example NSGroup that includes all Apps within the Product App Group, i.e., both the Product Info and the Inventory Applications:



secOps-pas-fd1-product-app-group

Membership Criteria

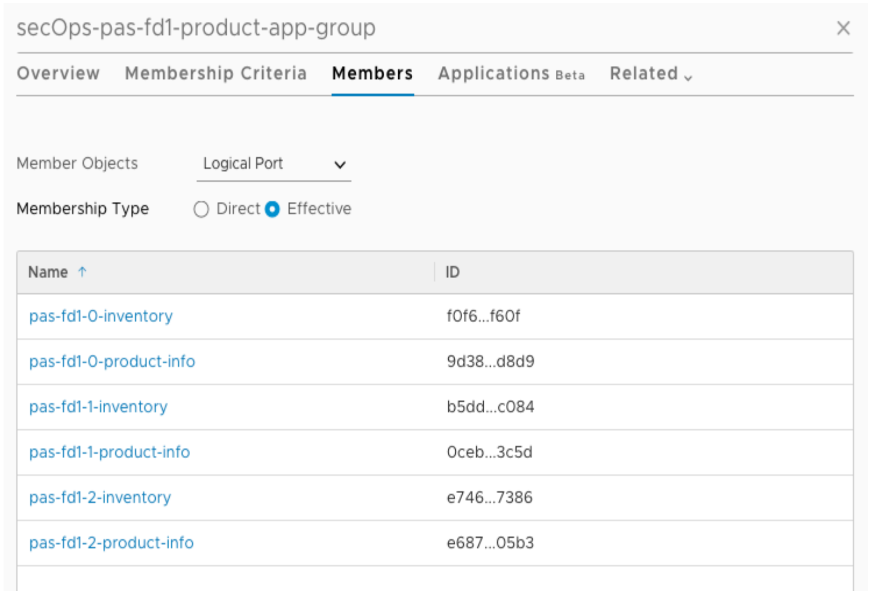
1. Logical Port

Tag Equals pas-fd1
Scope Equals ncp/cluster

Tag Equals product-app-group
Scope Equals ncp/cf_space_name

Figure 22 NSGroup Membership Criteria

Following image shows the members for the above group, essentially Application instances from both the Product Info App group and the Inventory App group:



secOps-pas-fd1-product-app-group

Member Objects: Logical Port

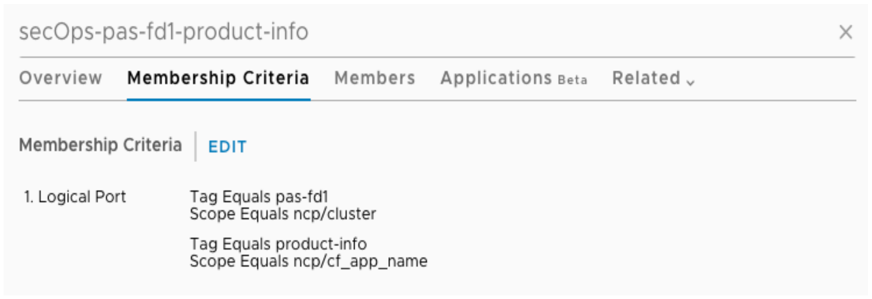
Membership Type: Direct [✓ Effective]

Name ↑	ID
pas-fd1-0-inventory	f0f6...f60f
pas-fd1-0-product-info	9d38...d8d9
pas-fd1-1-inventory	b5dd...c084
pas-fd1-1-product-info	0ceb...3c5d
pas-fd1-2-inventory	e746...7386
pas-fd1-2-product-info	e687...05b3

Figure 23 NSGroup Members

6.2.5.3.2 Fine-grained Micro-segmentation

NSX-T Data Center also allows fine-grained granularity of specifying security posture at a per application resolution and not just a group of applications. Following is an example NSGroup that may be created for "Product Info" App.

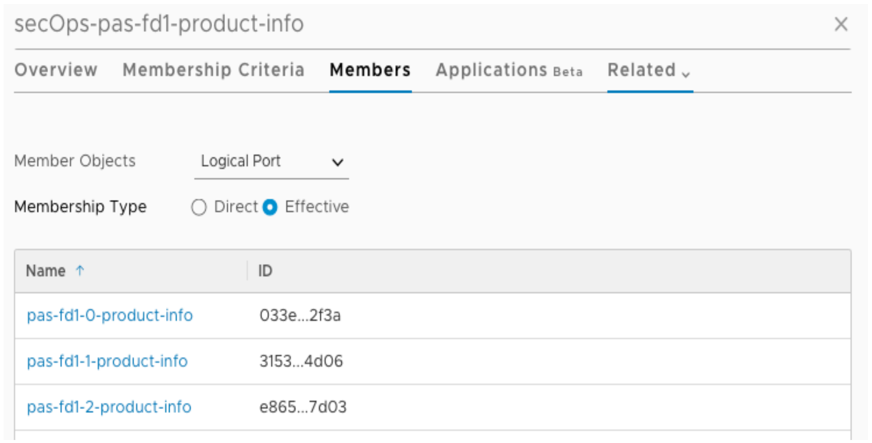


Membership Criteria

1. Logical Port — Tag Equals pas-fd1, Scope Equals ncp/cluster; Tag Equals product-info, Scope Equals ncp/cf_app_name

Figure 24 NSGroup Membership Criteria

Members for the above group would include all the instances of the "Product Info" application. In this case, "Product Info" has 3 instances.



Member Objects: Logical Port

Membership Type: Direct [✓ Effective]

Name ↑	ID
pas-fd1-0-product-info	033e...2f3a
pas-fd1-1-product-info	3153...4d06
pas-fd1-2-product-info	e865...7d03

Figure 25 NSGroup Members

6.2.5.4 Pre-Staged Security Posture

Since the Applications Tags are based on the actual Application name and its location within the PAS Org/Sapce, above NSGroups may be created by the SecOps even before the applications are actually deployed. Using this method, its possible for the SecOps to specify the security posture of the Applications that are yet to be deployed.

Above examples show how to create NSGroups for an App or an Org. One can use the same method to create a NSGroup for an entire PAS foundation by simply using the ncp/cluster tag and specifying the concerned PAS foundation or PKS Cluster.

Applications deployed in PAS foundation may need to access to external baremetal resources such Databases. In such cases, NSGroup with IP set may be used to help group the bare-metal resources.

6.2.5.5 On-Demand Security Posture

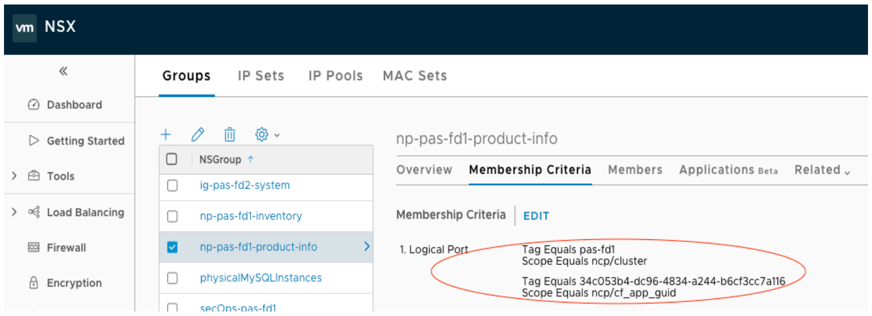
NSX-T Data Center also supports “cf network policy” that developers could use to specify the security posture once the applications are deployed. With “cf network policy” NSGroups are created dynamically when cf network policy is applied.

In the following example, product info app is allowed to access inventory app on TCP port 7007. When developed runs the following “cf add-network-policy” command

```
cf add-network-policy product-info --
destination-app inventory --port 7007 --
protocol tcp
```

Following two NSGroups are created dynamically.

NSGroup for the product-info applications:

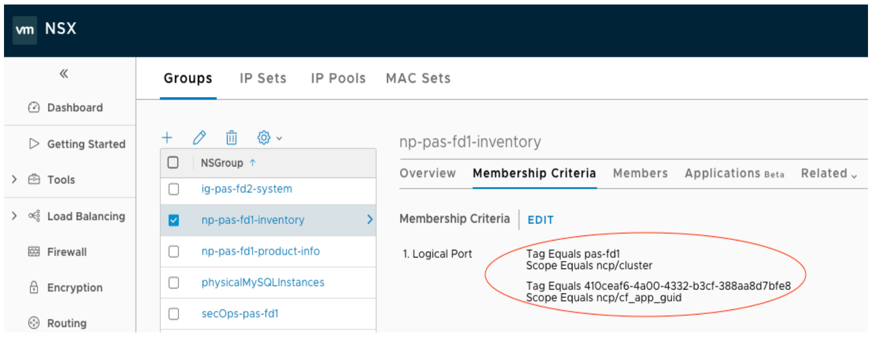


1. Logical Port
Tag Equals pas-fd1
Scope Equals ncp/cluster

Tag Equals 34c053b4-dc96-4834-a244-b6cf3cc7a116
Scope Equals ncp/cf_app_guid

Figure 26 Dynamically Created NSGroup via "cf add-network-policy"

NSGroup for the inventory application



1. Logical Port
Tag Equals pas-fd1
Scope Equals ncp/cluster

Tag Equals 410ceaf6-4a00-4332-b3cf-388aa8d7bfe8
Scope Equals ncp/cf_app_guid

Figure 27 Dynamically Created NSGroup via "cf add-network-policy"

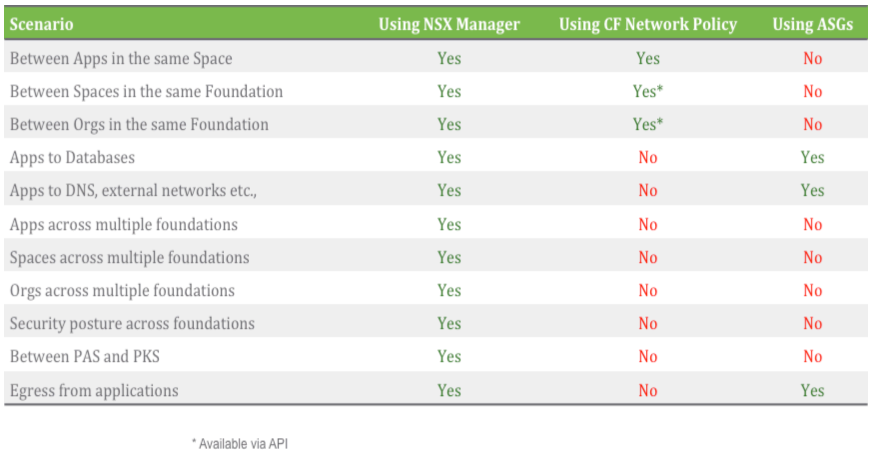
Notice the use of ncp/cf_app_guid instead of the ncp/cf_app_name in the above cases. Since, the applications are already deployed, “cf network policy” has access to the actual IDs of the application. This mechanism of using IDs is not available if pre-staging of the security groups is desired.

6.2.5.6 SecOps and DevOps

NSXT, for PAS and PKS, provides the following two mechanisms to specify security posture:

1. NSX-T Data Center Manager: Feature rich and granular controls over security posture allowing policy not only between application within a foundation but also between foundations and all other infrastructure managed NSX-T Data Center Domain and flows going in and out of NSX-T Data Center Domain to the external entities
2. CF Networking Policy: Tightly integrated into PAS and allows control of security policy between applications within a foundation

Following table shows what is achievable with the above two:

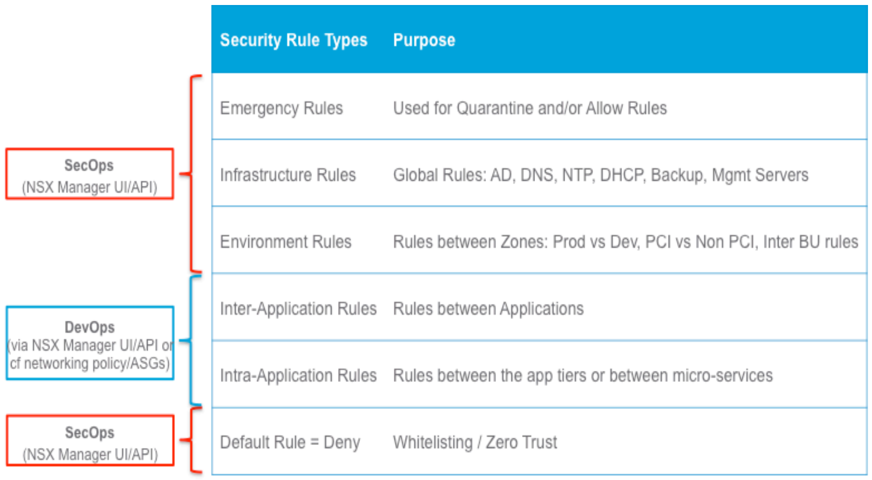


Scenario	Using NSX Manager	Using CF Network Policy	Using ASGs
Between Apps in the same Space	Yes	Yes	No
Between Spaces in the same Foundation	Yes	Yes*	No
Between Orgs in the same Foundation	Yes	Yes*	No
Apps to Databases	Yes	No	Yes
Apps to DNS, external networks etc.,	Yes	No	Yes
Apps across multiple foundations	Yes	No	No
Spaces across multiple foundations	Yes	No	No
Orgs across multiple foundations	Yes	No	No
Security posture across foundations	Yes	No	No
Between PAS and PKS	Yes	No	No
Egress from applications	Yes	No	Yes

* Available via API

Figure 28 Applying Policy via NSX-T Data Center Manager, CF Network Policy and ASGs

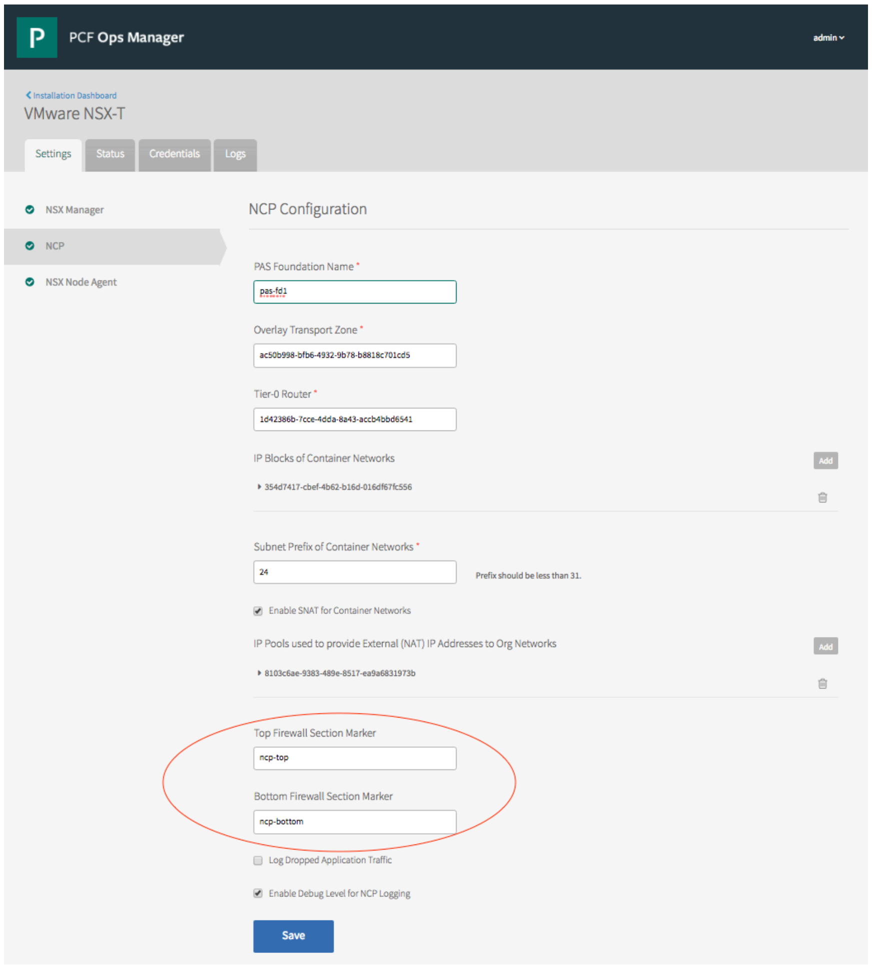
SecOps team, traditionally use the NSX-T Data Center Manager mechanism to specify umbrella rules over the entire infrastructure while the DevOps teams could use both the NSX-T Data Center Manager API directly for granular control over the entire application space or use the CF Network Policy to specify security posture between the applications within a foundation. Following image shows various categories of security related rules and the personas associated with each:



Security Rule Types	Purpose
Emergency Rules	Used for Quarantine and/or Allow Rules
Infrastructure Rules	Global Rules: AD, DNS, NTP, DHCP, Backup, Mgmt Servers
Environment Rules	Rules between Zones: Prod vs Dev, PCI vs Non PCI, Inter BU rules
Inter-Application Rules	Rules between Applications
Intra-Application Rules	Rules between the app tiers or between micro-services
Default Rule = Deny	Whitelisting / Zero Trust

Figure 29 Policy Grouping

NCP plugin also allows placement of the DevOps driven sections and rules within a specified region of the Firewall table by using section markers. Following image shows the section markers used to define where the firewall rules are placed.



VMware NSX-T

NCP Configuration

PAS Foundation Name *: pas-fd1

Overlay Transport Zone *: ac50b998-bfb6-4932-9b78-b8818c701cd5

Tier-0 Router *: 1d42386b-7cce-4dda-8a43-accb4bbd6541

IP Blocks of Container Networks: 354d7417-cbef-4b62-b16d-016df67fc556

Subnet Prefix of Container Networks *: 24 Prefix should be less than 31.

[✓] Enable SNAT for Container Networks

IP Pools used to provide External (NAT) IP Addresses to Org Networks: 8103c6ae-9383-489e-8517-ea9a6831973b

Top Firewall Section Marker: ncp-top

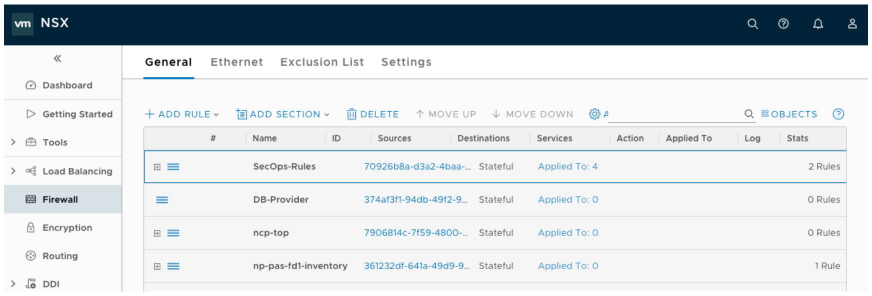
Bottom Firewall Section Marker: ncp-bottom

[ ] Log Dropped Application Traffic

[✓] Enable Debug Level for NCP Logging

Figure 30 PAS Related Policy Placement

Following image shows how the rules from NCP, that are driven by DevOps are placed in reference to the overall rules in the Firewall table:



#	Name	ID	Sources	Destinations	Services	Action	Applied To	Log	Stats
	SecOps-Rules	70926b8a-d3a2-4baa-...		Stateful	Applied To: 4				2 Rules
	DB-Provider	374af3f1-94db-49f2-9...		Stateful	Applied To: 0				0 Rules
	ncp-top	7906814c-7f59-4800-...		Stateful	Applied To: 0				0 Rules
	np-pas-fd1-inventory	361232df-641a-49d9-9...		Stateful	Applied To: 0				1 Rule

Figure 31 PAS Policy Placement

SecOps could have overarching rules on top of “ncp-top” section that get executed first and hence enforce overall security posture while still allowing the DevOps to define their own security posture for the specific applications.

With this model, its possible to have other sections on top of the DevOps driven NCP sections, that enforce their own security posture. For example, DB providers could allow access only on specific ports and leave the implementation of which applications are allowed to access the databases to the Application Developers (DevOps). More on this in a later section.

6.2.5.7 Application Level Access Control using NSX-T Data Center UI/API

NSX-T Data Center provides granular access control at the Application level.

Using either method discussed above, it is possible to security posture between the two applications. Product Info and Inventory. By default, all communication between the various applications is denied. If communication between any two applications or application groups is desired, then DevOps teams or the SecOps team need to setup that policy.

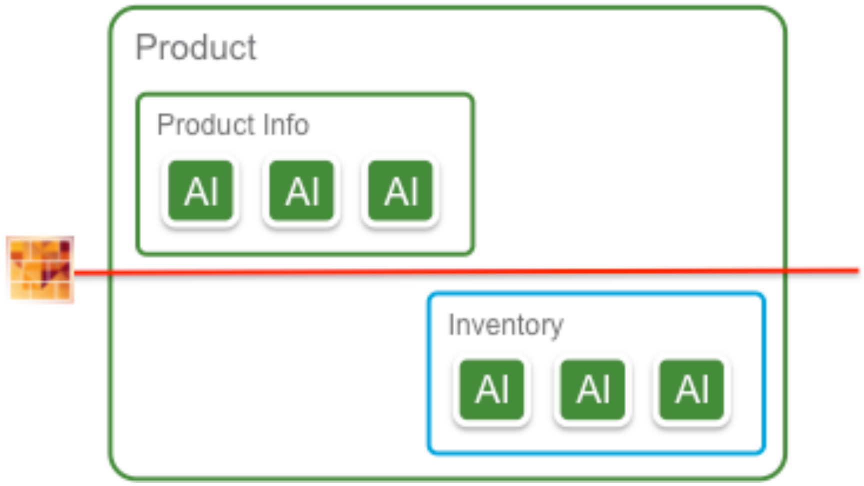


Figure 32 Security Between Apps within the same Org

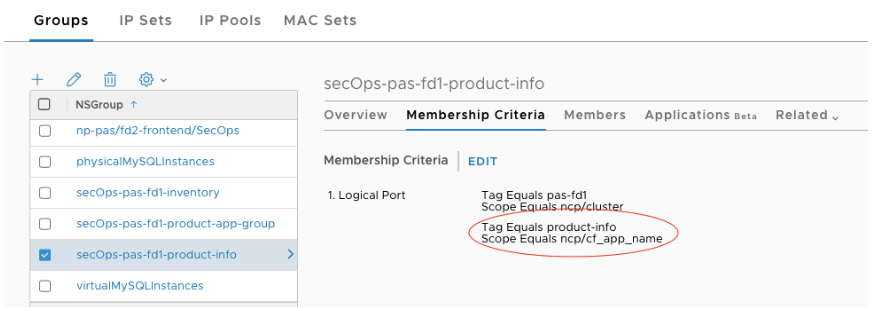
As shown in the previous section, SecOps an the DevOps team may define the security posture between various applications, in this example between Product Info and Inventory, even before the applications are actually deployed.

By default, access between applications is closed. If access needs to be allowed between Product Info and Inventory:

1. Create one NSGroup for each of the Applications as described in the NSGroup section
2. Setup a security policy between the two groups

In the following image, Product Info application is allowed access to the Inventory application:

Following is the membership criteria for Product Info applications:



	NSGroup
☐	np-pas/fd2-frontend/SecOps
☐	physicalMySQLInstances
☐	secOps-pas-fd1-inventory
☐	secOps-pas-fd1-product-app-group
☑	secOps-pas-fd1-product-info
☐	virtualMySQLInstances

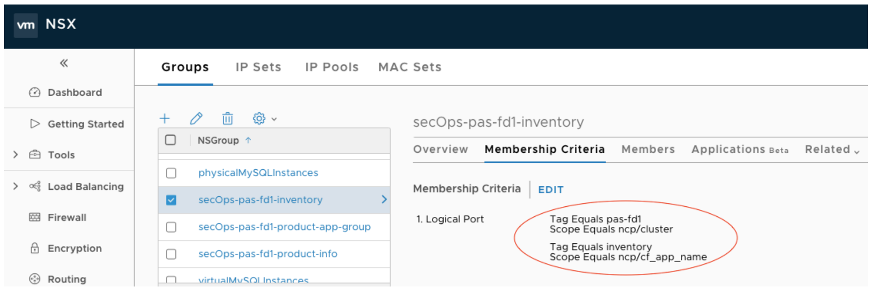
secOps-pas-fd1-product-info

Membership Criteria | EDIT

1. Logical Port
Tag Equals pas-fd1
Scope Equals ncp/cluster
Tag Equals product-info
Scope Equals ncp/cf_app_name

Figure 33 Example NSGroup for product-info Org

Following is the membership criteria for Inventory:



	NSGroup
☐	physicalMySQLInstances
☑	secOps-pas-fd1-inventory
☐	secOps-pas-fd1-product-app-group
☐	secOps-pas-fd1-product-info
☐	virtualMySQLInstances

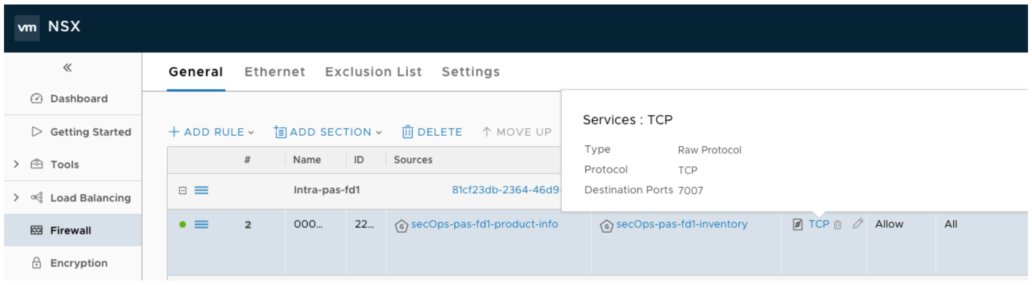
secOps-pas-fd1-inventory

Membership Criteria | EDIT

1. Logical Port
Tag Equals pas-fd1
Scope Equals ncp/cluster
Tag Equals inventory
Scope Equals ncp/cf_app_name

Figure 34 Example NSGroup for inventory Org

Using the above groups, security policy may be specified between the two applications. The following image shows the policy between the two applications – allowing Product Info application access to Inventory application on TCP port 7007.



#	Name	ID	Sources				
	Intra-pas-fd1	81cf23db-2364-46d9					
2	000...	22...	secOps-pas-fd1-product-info	secOps-pas-fd1-inventory	TCP	Allow	All

Services : TCP

Type	Raw Protocol
Protocol	TCP
Destination Ports	7007

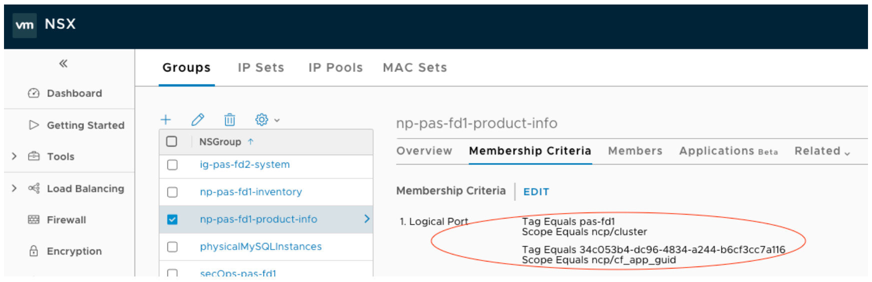
Figure 35 Security Posture between two Orgs

6.2.5.8 Application Level Access Control using "CF Network Policy"

Similarly, for the above example, if the on-demand approach via "cf network policy" is used where security is applied after the applications are deployed – then the NSGroups are deployed dynamically and the firewall is created automatically after running the following command:

```
cf add-network-policy product-info --
destination-app inventory --port 7007 --
protocol tcp
```

NSGroup for the product-info applications:



NSGroup
ig-pas-fd2-system
np-pas-fd1-inventory
np-pas-fd1-product-info
physicalMySQLInstances
secOps-pas-fd1

np-pas-fd1-product-info

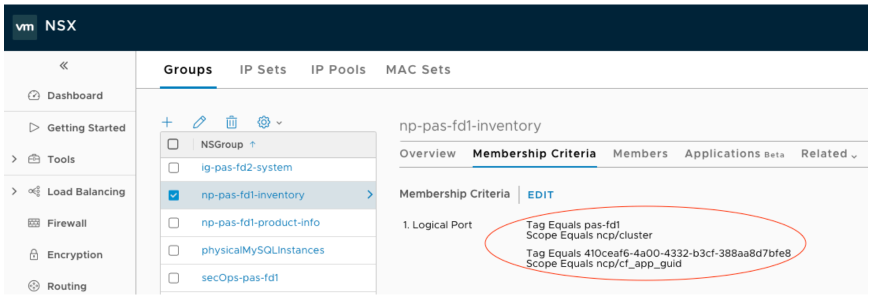
Membership Criteria | EDIT

1. Logical Port
Tag Equals pas-fd1
Scope Equals ncp/cluster

Tag Equals 34c053b4-dc96-4834-a244-b6cf3cc7a116
Scope Equals ncp/cf_app_guid

Figure 36 Dynamically Created NSGroup via "cf add-network-policy"

NSGroup for the inventory application



	NSGroup
☐	ig-pas-fd2-system
☑	np-pas-fd1-inventory
☐	np-pas-fd1-product-info
☐	physicalMySQLInstances
☐	secOps-pas-fd1

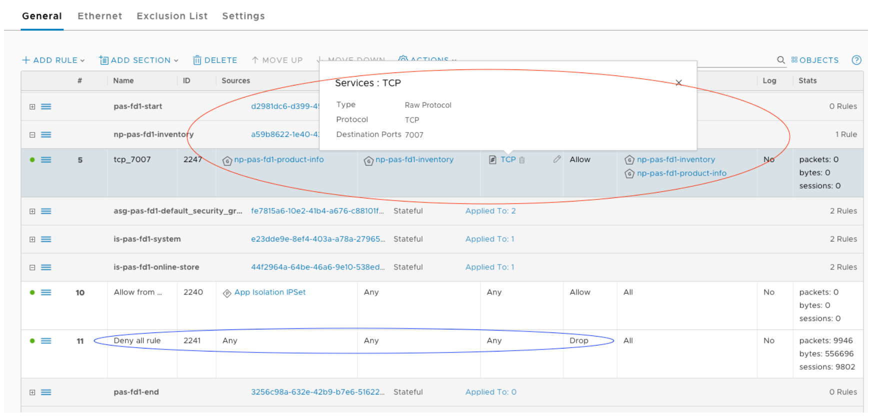
np-pas-fd1-inventory

Membership Criteria

1. Logical Port
Tag Equals pas-fd1
Scope Equals ncp/cluster
Tag Equals 410ceaf6-4a00-4332-b3cf-388aa8d7bfe8
Scope Equals ncp/cf_app_guid

Figure 37 Dynamically Created NSGroup via "cf add-network-policy"

And the following firewall rule is automatically created:



Services : TCP

Type	Raw Protocol
Protocol	TCP
Destination Ports	7007

#	Name	ID	Sources					Log	Stats
	pas-fd1-start		d2981dc6-d399-4						0 Rules
	np-pas-fd1-inventory		a59b8622-1e40-4						1 Rule
5	tcp_7007	2247	np-pas-fd1-product-info	np-pas-fd1-inventory	TCP	Allow	np-pas-fd1-inventory np-pas-fd1-product-info	No	packets: 0 bytes: 0 sessions: 0
	asg-pas-fd1-default_security_gr...		fe7815a6-10e2-41b4-a676-c88101f...	Stateful	Applied To: 2				2 Rules
	is-pas-fd1-system		e23dde9e-8ef4-403a-a78a-27965...	Stateful	Applied To: 1				2 Rules
	is-pas-fd1-online-store		44f2964a-64be-46a6-9e10-538ed...	Stateful	Applied To: 1				2 Rules
10	Allow from ...	2240	App Isolation IPSet	Any	Any	Allow	All	No	packets: 0 bytes: 0 sessions: 0
11	Deny all rule	2241	Any	Any	Any	Drop	All	No	packets: 9946 bytes: 556696 sessions: 9802
	pas-fd1-end		3256c98a-632e-42b9-b7e6-51622...	Stateful	Applied To: 0				0 Rules

Figure 38 Security Posture via NSGroups Created by "cf add-network-policy"

These sections/rules are placed; within the demarcation points for that foundation and towards the top (marked in red) before the default drop rules (marked in blue) for that foundation.

6.2.5.9 Application Group Level Access Control

Since NSGroups allow grouping of objects, its possible to easily create a security posture between various application groups by grouping them together. The following example shows how one can enforce policy between two application groups, (1) application group User Info and (2) application group Products.

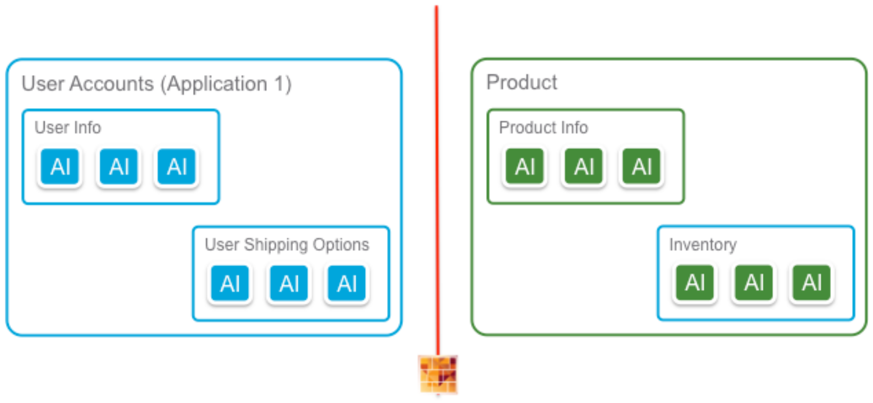


Figure 39 Security Posture Between Orgs

In cases where the User Accounts and Product fall in a different Orgs (PAS), NSX-T Data Center also enables creating access rules between Applications hosted on them.

For example, the following images shows how User Info Application running on one org (PAS) or namespace (PKS) is allowed to access Product Info Application running in a different Org or namespace. However, shipping info is denied access to Product Info Application:

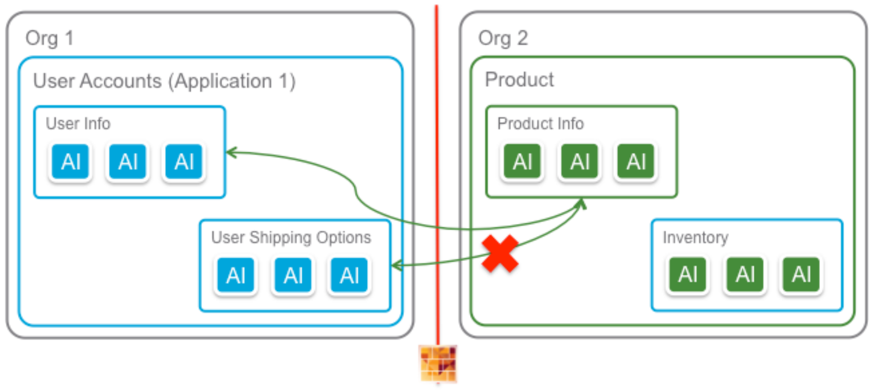


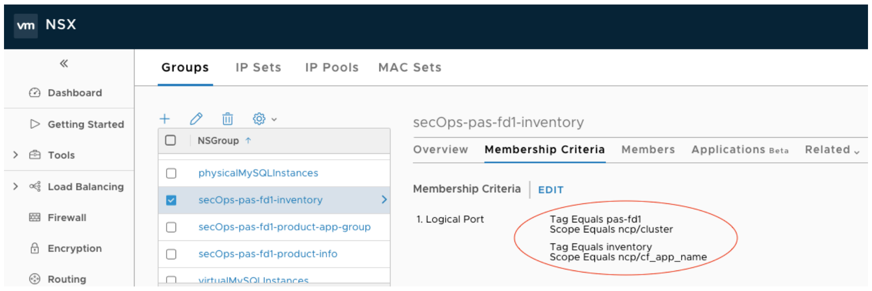
Figure 40 Security Posture between Applications across Orgs

By default, communication is not allowed between different groups. In this case, by default, User Shipping Options application is not allowed access to Product Info and vice versa. If User Info application needs access to Product Info application, it needs to be explicitly allowed.

Similar to the previous case of allowing access from Product Info to Inventory

1. Create NSGroup for each of the applications
2. Specify security policy

Following image shows the membership criteria for the Product Info security group:



physicalMySQLInstances
secOps-pas-fd1-inventory
secOps-pas-fd1-product-app-group
secOps-pas-fd1-product-info
virtualMySQLInstances

secOps-pas-fd1-inventory

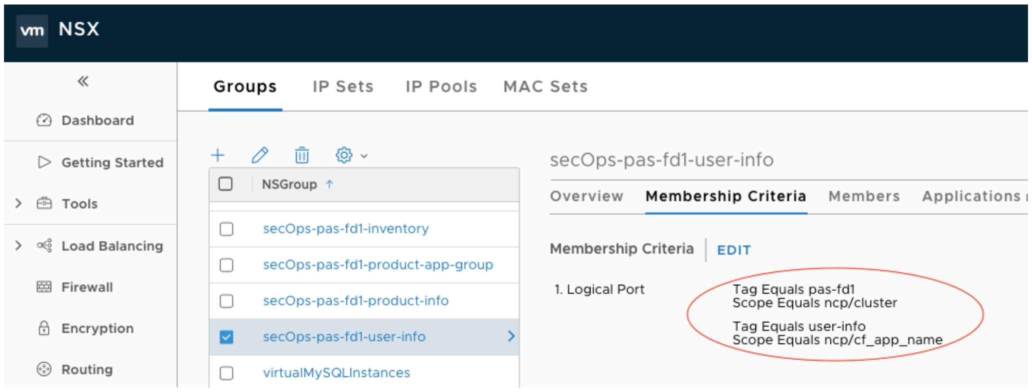
Membership Criteria

1. Logical Port

Tag Equals pas-fd1
Scope Equals ncp/cluster
Tag Equals inventory
Scope Equals ncp/cf_app_name

Figure 41 Example NSGroup using Application Name Tag

As shown in in the following image, create similar membership criteria for the User Info NSGroup:



secOps-pas-fd1-inventory
secOps-pas-fd1-product-app-group
secOps-pas-fd1-product-info
secOps-pas-fd1-user-info
virtualMySQLInstances

secOps-pas-fd1-user-info

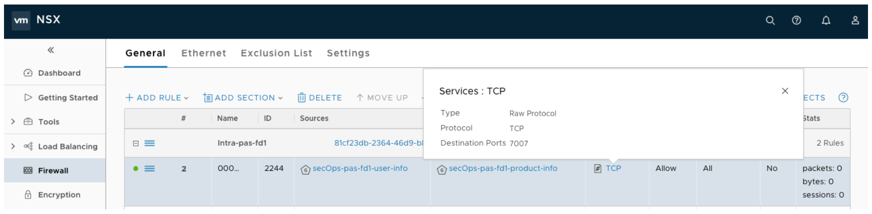
Membership Criteria

1. Logical Port

Tag Equals pas-fd1
Scope Equals ncp/cluster
Tag Equals user-info
Scope Equals ncp/cf_app_name

Figure 42 Example NSGroup using Application Name Tag

Use these groups to specify the security posture, similar to what was shown in the earlier section with access between apps within the same org:



Services : TCP

Type	Raw Protocol
Protocol	TCP
Destination Ports	7007

#	Name	ID	Sources						Stats
	Intra-pas-fd1		81cf23db-2364-46d9-b						2 Rules
2	000...	2244	secOps-pas-fd1-user-info	secOps-pas-fd1-product-info	TCP	Allow	All	No	packets: 0 bytes: 0 sessions: 0

Figure 43 Security Posture between Application across Orgs

6.2.5.10 Access to services such as Databases

The following image shows how NSX-T Data Center allows a security admin or a DB administration in this case, to setup over arching security related rules such as allow connections only on the database port as opposed to for example ssh on resources such as databases and yet leave the freedom to the dev/ops folks to determine whether they want to leverage that access for any applications. In this case, dev/ops folks have the flexibility to open ports on the firewall to allow DB access for their applications based on their requirements. However, they will only be able to use the ports that are already allowed.

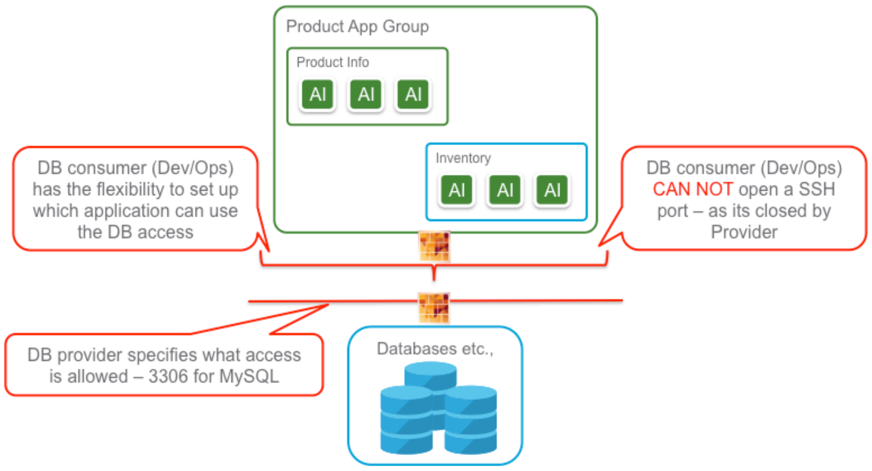
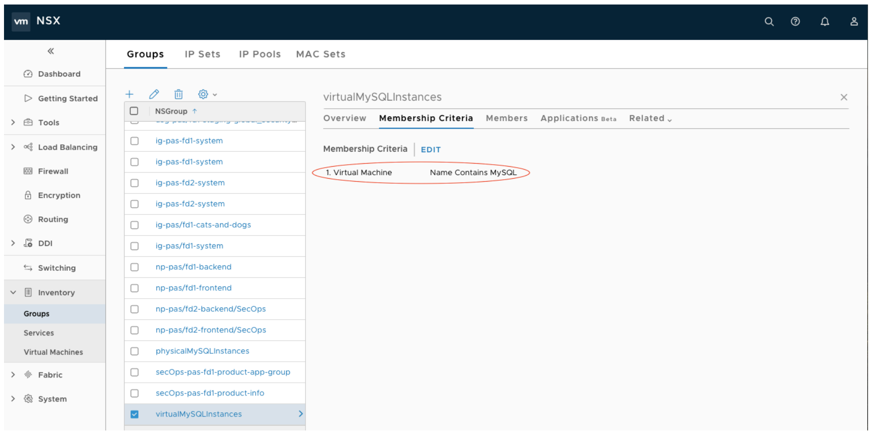

Product App Group
Product Info
AI AI AI
Inventory
AI AI AI
DB consumer (Dev/Ops) has the flexibility to set up which application can use the DB access
DB consumer (Dev/Ops) CAN NOT open a SSH port – as its closed by Provider
DB provider specifies what access is allowed – 3306 for MySQL
Databases etc.,

Figure 44 Security Posture for Databases

In the case of entities outside the foundation, based on whether the entities are within the NSX-T Data Center Domain or outside, one can use various mechanisms to build the NSGroups.

If the entities are within the NSX-T Data Center Domain, tags and properties of the VM may be used to populate the NSGroups. For example, in the following image we use the name of the VM to identify virtual MySQL instances:



NSGroup

- ig-pas-fd1-system
- ig-pas-fd1-system
- ig-pas-fd2-system
- ig-pas-fd2-system
- ig-pas/fd1-cats-and-dogs
- ig-pas/fd1-system
- np-pas/fd1-backend
- np-pas/fd1-frontend
- np-pas/fd2-backend/SecOps
- np-pas/fd2-frontend/SecOps
- physicalMySQLInstances
- secOps-pas-fd1-product-app-group
- secOps-pas-fd1-product-info
- virtualMySQLInstances

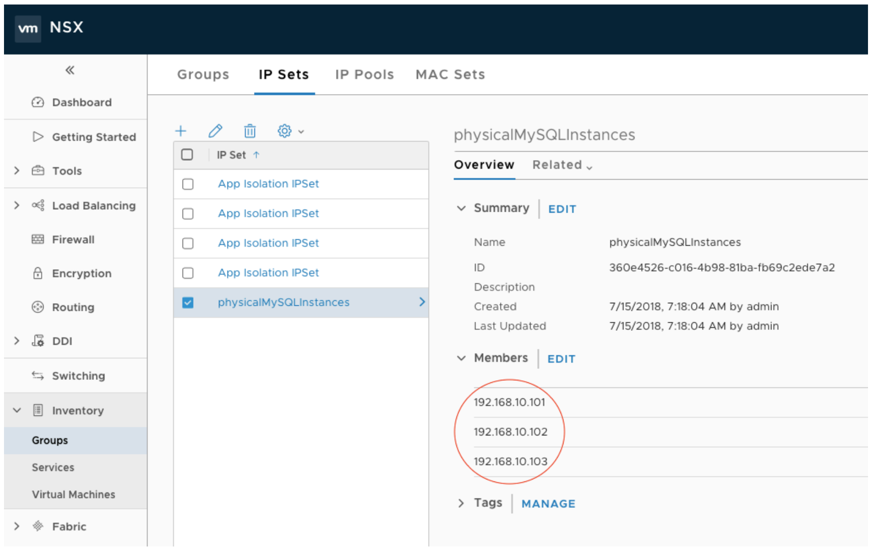
virtualMySQLInstances

Membership Criteria

1. Virtual Machine   Name Contains MySQL

Figure 45 Example NSGroup for MySQL VMs

If the entities are external to the NSX-T Data Center Domain, IP sets may be used to build the NS Groups. In the following example, first create IP set with the IPs (192.168.10.101-103) of the physical MySQL instances:



IP Set

- App Isolation IPSet
- App Isolation IPSet
- App Isolation IPSet
- App Isolation IPSet
- physicalMySQLInstances

physicalMySQLInstances

Summary

Name	physicalMySQLInstances
ID	360e4526-c016-4b98-81ba-fb69c2ede7a2
Description	
Created	7/15/2018, 7:18:04 AM by admin
Last Updated	7/15/2018, 7:18:04 AM by admin

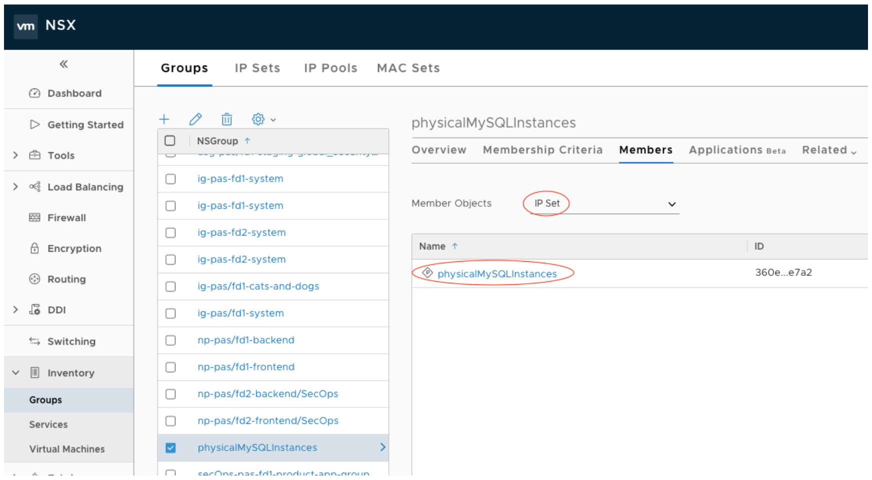
Members

192.168.10.101
192.168.10.102
192.168.10.103

Tags

Figure 46 Example NSGroup for Physical MySQL Database Servers

Above IP set may be used to create NSGroup for the physical MySQL instances. The following image shows how the above IP set could be used to create a NSGroup that points to the physical MySQL instances:



NSGroup
ig-pas-fd1-system
ig-pas-fd1-system
ig-pas-fd2-system
ig-pas-fd2-system
ig-pas/fd1-cats-and-dogs
ig-pas/fd1-system
np-pas/fd1-backend
np-pas/fd1-frontend
np-pas/fd2-backend/SecOps
np-pas/fd2-frontend/SecOps
[✓] physicalMySQLInstances
secOps-pas-fd1-product-app-group

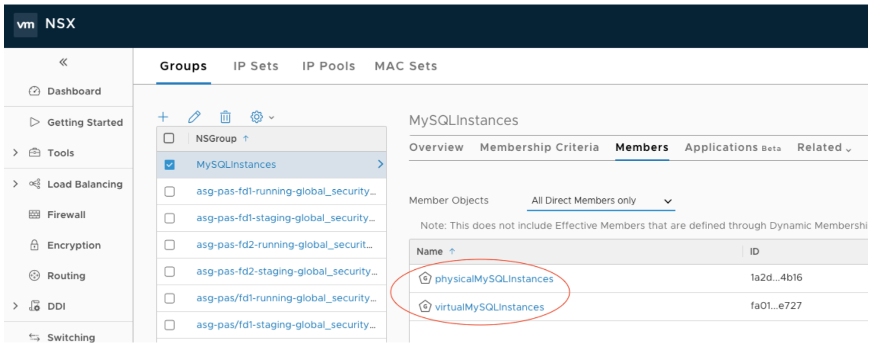
physicalMySQLInstances

Member Objects: IP Set

Name	ID
physicalMySQLInstances	360e...e7a2

Figure 47 Nested NSGroups

NSGroups allow nesting. This helps in combining the physical MySQL instances and the virtual MySQL instances to create a common policy across both. Following image shows a combined NSGroup “MySQLInstances” which is a combination of both physical and virtual MySQL instances.



NSGroup
MySQLInstances
asg-pas-fd1-running-global_security…
asg-pas-fd1-staging-global_security…
asg-pas-fd2-running-global_securit…
asg-pas-fd2-staging-global_security…
asg-pas/fd1-running-global_security…
asg-pas/fd1-staging-global_security…

MySQLInstances

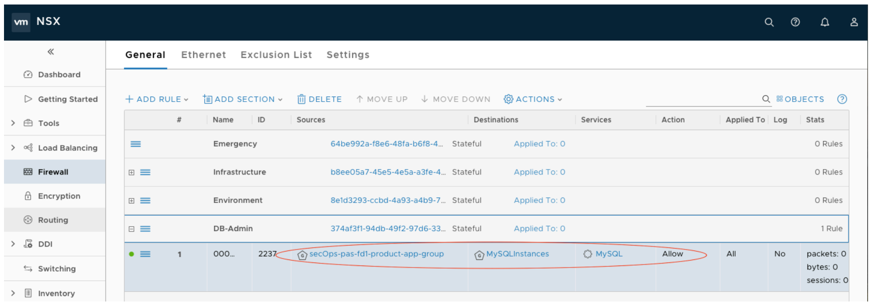
Member Objects: All Direct Members only

Note: This does not include Effective Members that are defined through Dynamic Membershi

Name	ID
physicalMySQLInstances	1a2d...4b16
virtualMySQLInstances	fa01...e727

Figure 48 Nested NSGroups

Using this group, “MySQLInstances”, a policy could be created that allows access to only specific ports on all physical and virtual MySQL instances. In the following example “SecOps-pas-fd1-product-app-group” is used to specify a security policy allowing the applications within the Product Application group to have access to the MySQL instances on MySQL service port:



#	Name	ID	Sources	Destinations	Services	Action	Applied To	Log	Stats
	Emergency		64be992a-f8e6-48fa-b6f8-4… Stateful	Applied To: 0					0 Rules
	Infrastructure		b8ee05a7-45e5-4e5a-a3fe-4… Stateful	Applied To: 0					0 Rules
	Environment		8e1d3293-ccbd-4a93-a4b9-7… Stateful	Applied To: 0					0 Rules
	DB-Admin		374af3f1-94db-49f2-97d6-33… Stateful	Applied To: 0					1 Rule
1	000…	2237	secOps-pas-fd1-product-app-group	MySQLInstances	MySQL	Allow	All	No	packets: 0 bytes: 0 sessions: 0

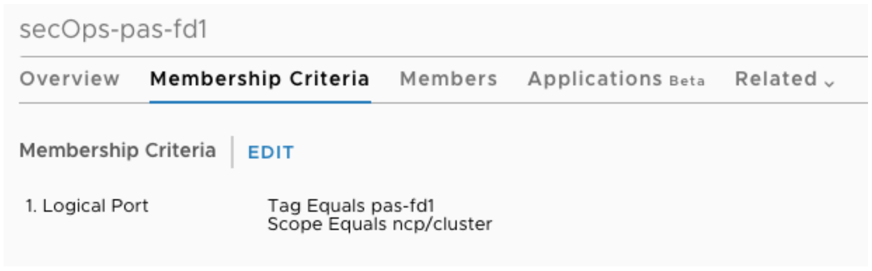
Figure 49 Security Posture for Databases

6.2.5.11 Security Posture Between Foundations

If we would like to allow all apps within foundation 1 to be able to access all the apps within foundation 2, the process is the same:

1. Create one NSGroup for each of the foundations
2. Specify the security posture between the two foundations

Following image shows the membership criteria for NSGroup for one of the foundations with name pas-fd1

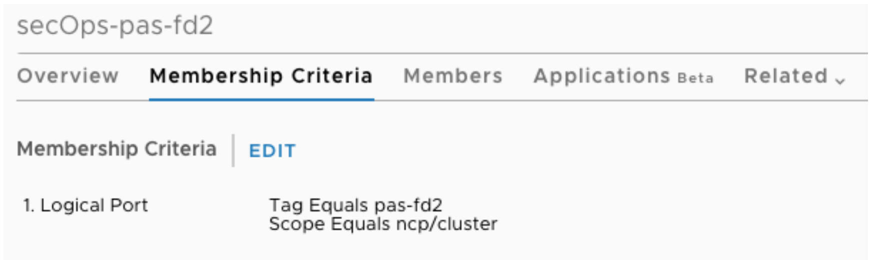


Membership Criteria

1. Logical Port — Tag Equals pas-fd1, Scope Equals ncp/cluster

Figure 50 Example Foundation level NSGroup

Similarly, create a NSGroup for foundation 2. In this example and in the image below, foundation 2's name is pas-fd2:

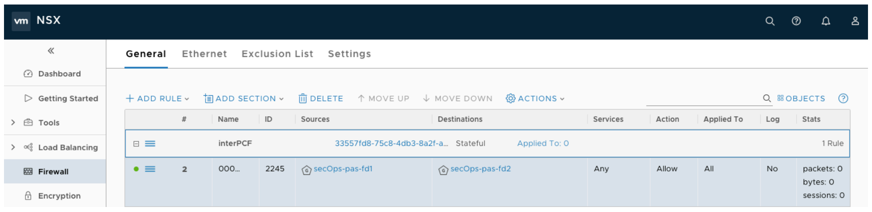


Membership Criteria

1. Logical Port — Tag Equals pas-fd2, Scope Equals ncp/cluster

Figure 51 Example Foundation Level NSGroup

And finally the security posture between the foundations – in this example – we allow all communication where the source is pas-fd1 and destination is pas-fd2



	#	Name	ID	Sources	Destinations	Services	Action	Applied To	Log	Stats
		interPCF		33557fd8-75c8-4db3-8a2f-a... Stateful	Applied To: 0					1 Rule
	2	000...	2245	secOps-pas-fd1	secOps-pas-fd2	Any	Allow	All	No	packets: 0 bytes: 0 sessions: 0

Figure 52 Security Posture Between Foundations

6.2.5.12 Section and Rule Placement

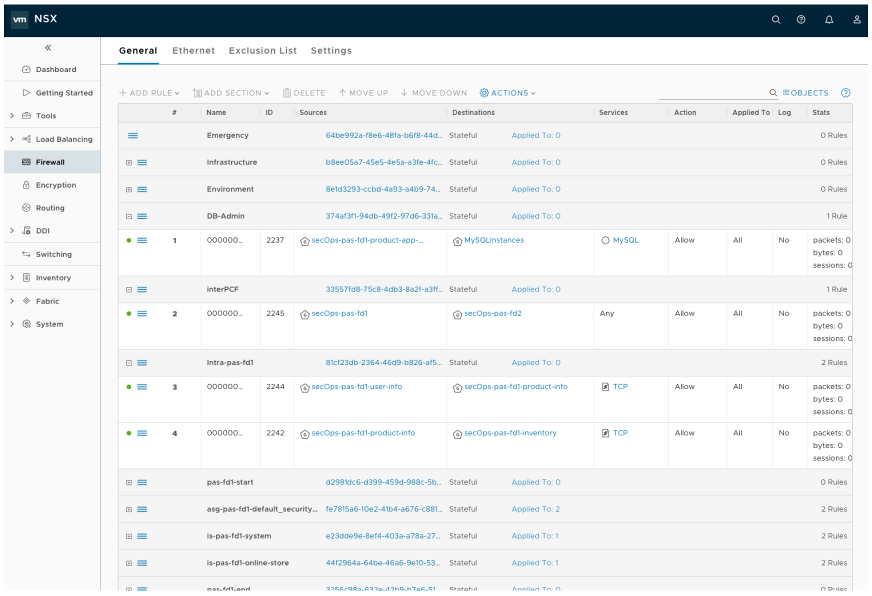
Above sections have shown the different rules that one can construct to define the security posture within a foundation and with other objects within and outside of NSX-T Data Center domain. To actually make these sections and rules effective, placement of these sections and rules within the Firewall table is crucial.

Following table shows how the sections and rules should be arranged to ensure that they are effective:

SecOps	Emergency	Over all rules across the infrastructure and environment
	Infrastructure	AD, DNS etc.,
	Environment	Prod vs Dev etc.,
DB-Admin	Resources external to PAS foundation – these resources may exist within the NSX-T Data Center Domain or even	Database related rules. There could be other rules that define security postures for any other resources used by the PAS foundations

	outside of it	
Inter-PCF		Security posture between foundations
Intra-PCF	DevOps Driven via NSX-T Data Center Manager	These are multiple sections that are specific to each foundation – that come before the NCP based sections and rules
	Developer Driven via NCP	Within specified sections – unique area for each foundation
SecOps Default Rule		Deny everything else

Following image from NSX-T Data Center Manager’s Firewall shows the layout of all the sections and rules discussed in the previous sections:



	#	Name	ID	Sources	Destinations	Services	Action	Applied To	Log	Stats
		Emergency		64be992a-f8e6-48fa-b6f8-44d...	Stateful Applied To: 0					0 Rules
		Infrastructure		b8ee05a7-45e5-4e5a-a3fe-4fc...	Stateful Applied To: 0					0 Rules
		Environment		8e1d3293-ccbd-4a93-a4b9-74...	Stateful Applied To: 0					0 Rules
		DB-Admin		374af3f1-94db-49f2-97d6-331a...	Stateful Applied To: 0					1 Rule
●	1	000000...	2237	secOps-pas-fd1-product-app-...	MySQLInstances	MySQL	Allow	All	No	packets: 0 bytes: 0 sessions: 0
		interPCF		33557fd8-75c8-4db3-8a2f-a3ff...	Stateful Applied To: 0					1 Rule
●	2	000000...	2245	secOps-pas-fd1	secOps-pas-fd2	Any	Allow	All	No	packets: 0 bytes: 0 sessions: 0
		Intra-pas-fd1		81cf23db-2364-46d9-b826-af5...	Stateful Applied To: 0					2 Rules
●	3	000000...	2244	secOps-pas-fd1-user-info	secOps-pas-fd1-product-info	TCP	Allow	All	No	packets: 0 bytes: 0 sessions: 0
●	4	000000...	2242	secOps-pas-fd1-product-info	secOps-pas-fd1-inventory	TCP	Allow	All	No	packets: 0 bytes: 0 sessions: 0
		pas-fd1-start		d2981dc6-d399-459d-988c-5b...	Stateful Applied To: 0					0 Rules
		asg-pas-fd1-default_security...		fe7815a6-10e2-41b4-a676-c881...	Stateful Applied To: 2					2 Rules
		is-pas-fd1-system		e23dde9e-8ef4-403a-a78a-27...	Stateful Applied To: 1					2 Rules
		is-pas-fd1-online-store		44f2964a-64be-46a6-9e10-53...	Stateful Applied To: 1					2 Rules
		pas-fd1-end		3256c98a-632e-42b9-b7e6-51	Stateful Applied To: 0					0 Rules

Figure 53 Firewall Section and Rule Placement

6.2.5.13 Ingress and Egress Access to the Foundation

NSX-T Data Center also offers traditional firewall type of mechanisms to only allows specific flows in and out of the entire PAS/PKS domain. These ingress/egress rules could be applied at the N/S layer on the T0 router.

Apart from the above, NSX-T Data Center also supports and is tightly integrated with PAS's legacy security architecture using Application Security Groups (ASGs). Design Options

7 Design Options

Following are some of the design options based on the above building blocks. These are tuned to address PAS specific requirements. Please refer to the NSX-T Data Center design guide for in-depth information on deployment recommendations.

7.1 PoC Labs

The first in this set of design options is purely meant for PoC and R&D. This design uses just two nodes – one for the mgmt. and edge and one for compute. All the management components including vSphere related, NSX-T Data Center related and BOSH related are placed in a single host. On the NSX-T Data Center front, in this design, it may be OK to use just one controller considering the resource constraints, as long as this is only used for PoC.

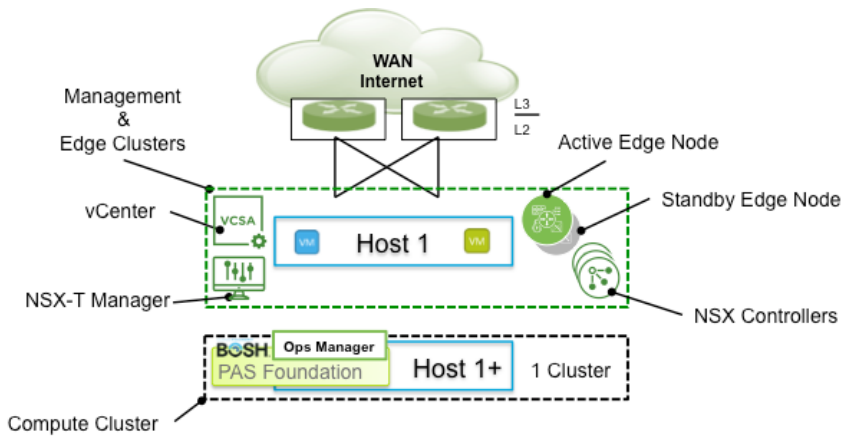


Figure 54 PAS on NSX-T Data Center – PoC Design

On the PAS front, small foot print version is recommended to conserve resource utilization.

7.2 Collapsed Management and Edge Resources Design

The following design uses multiple hosts for both the compute cluster and the mgmt/edge cluster. This design provides basic fault tolerance at a host level. However, since all the nodes are in the same rack, its not designed to tolerate rack level failures.

In this design, management and edge clusters are still collapsed into a single cluster. Hence, this is not designed for performance from the edge perspective.

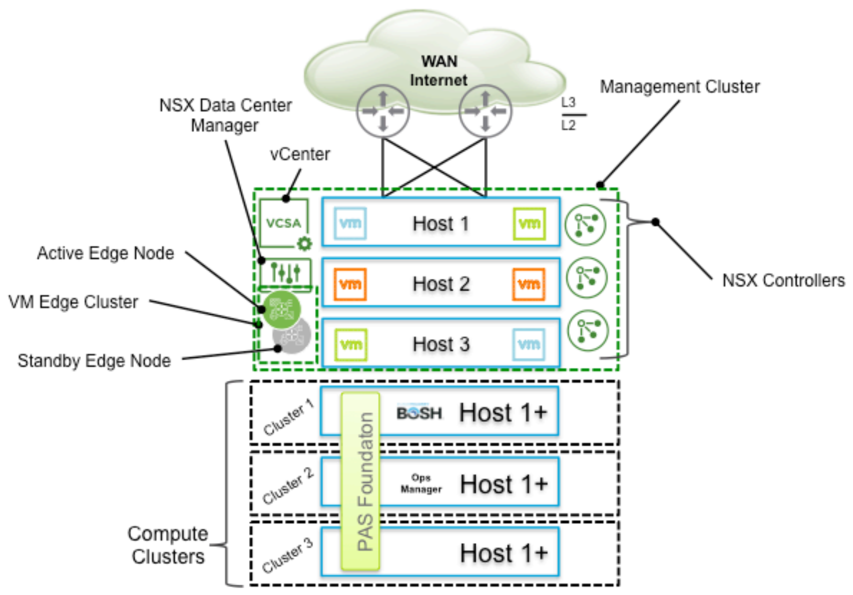


Figure 55 Collapsed Management and Edge Clusters Design

7.3 Dedicated Management and Edge Resources Design

Following is the recommended design for production deployments. Unlike the vSphere Enterprise Design guidelines, in this design, vCenter clusters are not striped across racks but are applied on one or more racks. This design requires at least 3 nodes for the vSphere management components. In this design,

- vCenter, NSX-T Data Center Manager are placed on Management cluster with vSphere HA enabled to protect from host failure and also provide resource reservation
- NSX-T Data Center Controllers are placed on separate hosts with anti-affinity setting and resource reservation

- Edge nodes are placed in their own cluster – spread across multiple hosts. Based on workload and performance requirements, edge nodes could be either virtual or bare metal.
- BOSH, Ops Manager and rest of the PAS components are placed in the compute cluster.

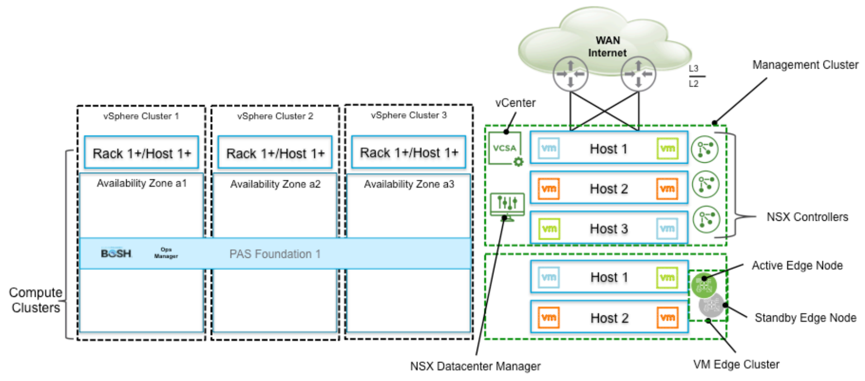


Figure 56 Dedicated Management and Edge Clusters Design

To deploy multiple foundations, either more clusters could be added to the same NSX-T Data Center domain, or reuse the existing clusters based on resource availability. Following image shows placement of multiple foundations on the same clusters.

If using the same cluster to run multiple PAS Foundations, resource pools dedicated to each Foundation needs to be created per vSphere cluster. Resource pools are used to create PAS availability zones.

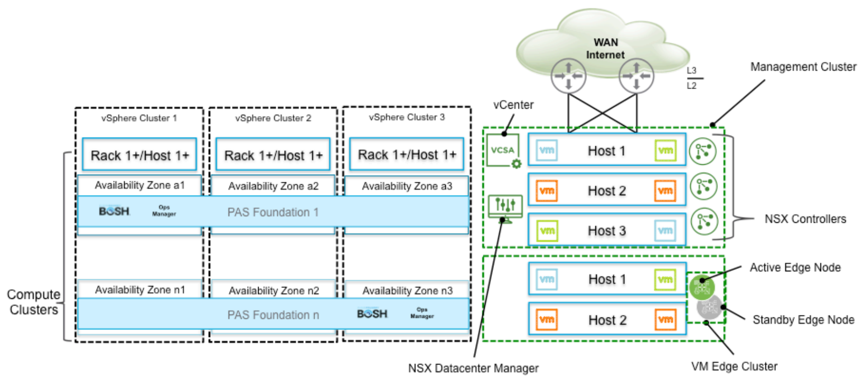


Figure 57 Multiple PAS Foundations

8 PKS Overview

Pivotal Container Service (PKS) is Pivotal's Kubernetes based production-grade offering. PKS allows leveraging the VMware SDDC as a unified infrastructure for containers and VMs. PKS is designed with focus on high availability, auto-scaling and supports rolling upgrades. It comes bundled with advanced networking and security features at the container level using NSX-T Data Center. Harbor, an open source registry server provides is also built into PKS and provides support for RBAC, LDAP/AD, Container image vulnerability scanning, policy based image replication, notary and auditing services apart from storing and distributing Docker images.

PAS is positioned as stateless and opinionated whereas PKS is positioned as stateful and not opinionated. Opinionated is merely a term to describe how strictly a certain model of micro-services architecture is followed. PAS enforces adherence to its model of deploying micro-services.

9 PKS Components

In the following section we will take a look at some of the components that go into a PKS install and how they relate to the network. To learn more about PKS in general, please check out their website: https://pivotal.io/learn

A single complete installation of PKS, with all the required components, is known as a PKS Management Plane.

In the Kubernetes world, Pods are used to contain 1 or more containers. Pods are created, destroyed and recreated on demand. On recreation, Pods may not have the same IP addresses that they had before. Kubernetes Service helps in keeping track of connecting the applications to the actual containers where the workload is running.

The following image shows the architecture of PKS with the underlying NSX, vSphere and vSAN Infrastructure.

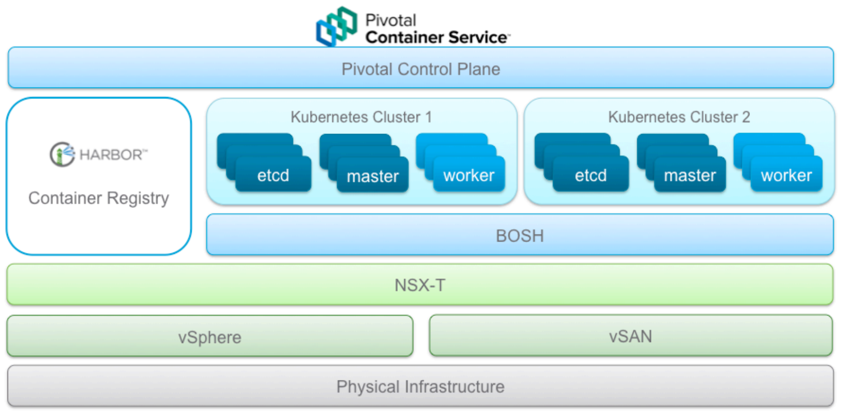


Figure 58 PKS Architecture

9.1 Ops Manager

Ops Manager helps with the installation and configuration of Pivotal Container Service (PKS). BOSH is the first component, that's installed by Ops Manager. BOSH

BOSH, a primary PKS tile, is an open source project that unifies release engineering, deployment, and lifecycle management of small and large-scale cloud software. Bosh was originally designed to deploy the open source Cloud Foundry.

BOSH is used within PKS to

1. Provision VMs
2. Deploy software to BOSH provisioned VMs
3. Monitoring
4. Recovery from failures
5. Software updates

Internally BOSH has the following components

9.1.1 Director

This is the core-orchestrating component, which is installed as a single VM and that controls VM creation and deployments as well as other software and service lifecycle events.

Either a Command line Interface (CLI) or Direct API may be used to interact with the Director.

9.1.2 Postgres DB

Director uses a Postgres database to store information about the desired state of deployment including information about stemcells, releases and deployments. This Postgres DB is internal to the Director VM.

9.1.3 Blobstore

Blobstore is basically a repository for any files, including large binary files. Director uses a blobstore to host the source code and related source and binaries used to compile and build the release. This is also an internal component of the Director VM and the releases that are built here are PKS components.

9.2 Master Node

In a Kubernetes cluster, Master node takes care of all the management tasks. It orchestrates workloads and is aware of where the actual services are running. API server runs on the master node and is the entry point for all the REST commands used to control the cluster. State persistence is provided via etcd. Etcd is a distributed key value store that's generally used for storing configuration.

Master Node uses the scheduler component to help with deployment of configured pods and services onto the nodes.

9.3 Worker Node

Worker nodes have the services necessary to run Pods and be managed by the master nodes. Services that run on the worker nodes include container runtime, kubelet and kube-proxy.

9.4 Harbor

Harbor is an an open source cloud native registry project that stores, signs, and scans content. Harbor extends the open source Docker Distribution by adding the functionalities usually required by users such as security, identity and management. Having a registry closer to the build and run environment can improve the image transfer efficiency. Harbor supports replication of images between registries, and also offers advanced security features such as user management, access control and activity auditing.

10 PKS on NSX-T Data Center – Building Blocks

In this section we will take a look at two kinds of building blocks that play a role in design considerations.

- **Virtual Infrastructure**: Placement of virtual infrastructure components such as the vCenter, NSX-T Data Center Manager, BOSH, OpsManager and PKS workload related virtual machines
- **Logical Infrastructure**: Design considerations around logical switching, routing, firewalls, NAT and Load Balancer.

10.1 NSX-T Data Center Management and Edge Design Considerations

The following section shows the basic building blocks of PKS running on NSX-T Data Center. Some of these components will be separated into a separate block based on hardware resource availability.

Following are some of the key blocks that are part of PKS on NSX-T Data Center running vSphere infrastructure:

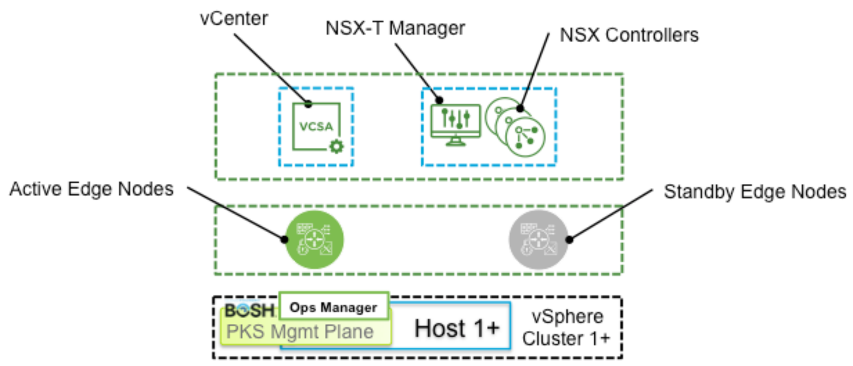


Figure 59 PKS Base Components

Simplified PKS on NSX-T Data Center - components:

- vSphere Management: To manage the vSphere virtual infrastructure
- NSX-T Data Center Management: Containing NSX-T Data Center Manager and NSX-T Data Center Controllers
- Edge Nodes: Containing T0 Routers to provide North / South connectivity for the PKS Cluster

- Compute: Hosts used to run the PKS workloads ie., infra to run the actual PKS driven Applications

Apart from the above, PKS management components placement should be considered. Our recommendation is to place them on the compute cluster.

Recommendation is to have at least three clusters.

1. Mgmt Cluster: To host all the management related components of
   a. vSphere (vCenter)
   b. NSX-T Data Center (NSX-T Data Center Manager, NSX-T Data Center Controllers)
2. Edge Cluster: To host NSX-T Data Center Edges that provide north / south connectivity to the PKS Cluster. Currently PKS only supports Virtual Edges.
3. Compute Cluster: To host the actual applications deployed with PKS. One management cluster for the management plane and 1 or more compute clusters for the worker nodes.

Following diagram shows a PKS deployment with the above components separated into their own clusters.

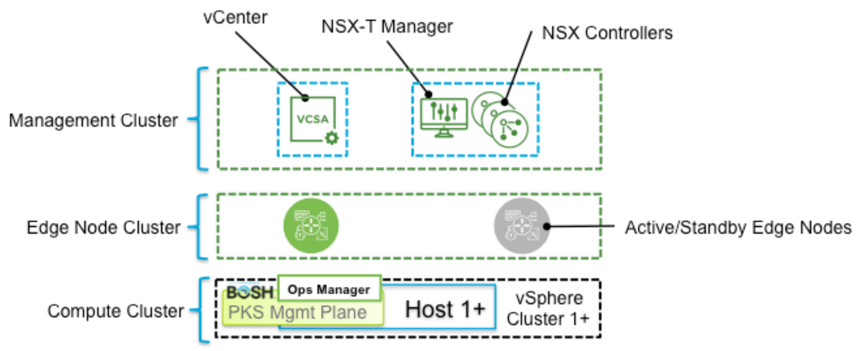


Figure 60 PKS - vSphere Clusters

10.1.1 Edge Size and Type Considerations

Large form factor is the recommended approach for the PKS especially when using the NSX-T Data Center provided Load Balancer functionality. This will provide flexibility in terms of the number of clusters that may be deployed.

10.2 Logical Infrastructure

10.2.1 Switching

Three logical switches are recommended for Pivotal Container Service:

1. Infrastructure: BOSH and Ops Manager are installed on this logical switch
2. Kubernetes Cluster Network: This logical switch will host the actual user workloads

Note: In the current PKS 1.0, each of these network will need an external IP. PKS 1.0 will by default use an IP from the provided External IP pool.

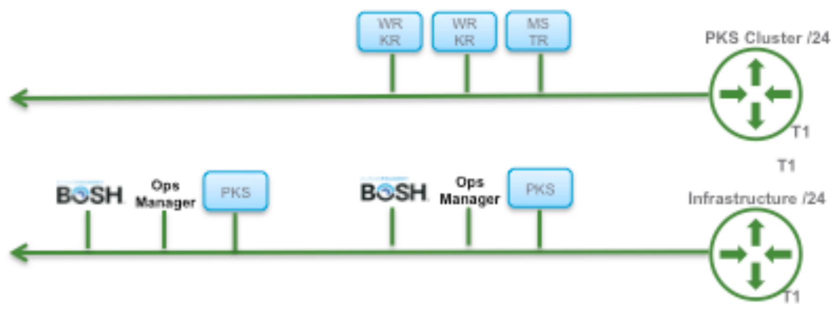


Figure 61 PKS - Logical Switches

Additional logical switches and T1 routers are created on demand as namespaces are created.

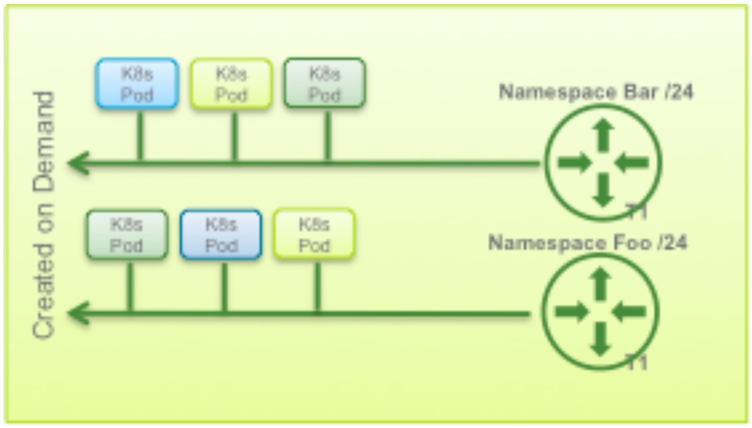


Figure 62 PKS - On Demand Logical Switches

Each name space would trigger the creation of a T1 router and a logical switch that is dedicated to that namespace.

10.2.2 Routing

As mentioned earlier, each namespace has its own separate logical switch that's attached to a T1 router that's dedicated to that namespace. Every namespace in PKS will have a dedicated T1 router. Routing between the namespaces and from PKS management components to workload components goes through the NSX-T Data Center T0 router.

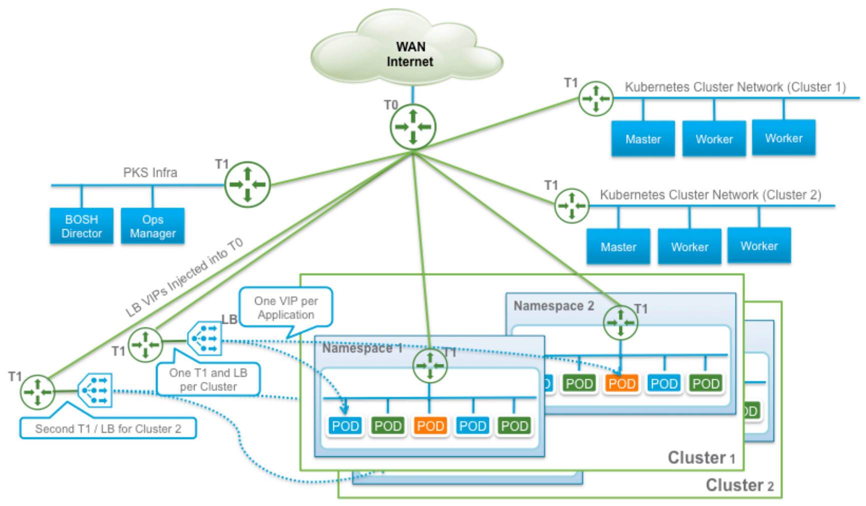


Figure 63 PKS - Routed Design

In routed topologies, namespaces may not need to be routed and could be a private network.

10.2.3 NAT

NSX-T Data Center provides NAT as a standalone feature and also as part of the Load Balancer service. The following section is focused on NAT feature. NAT runs on router node (T0 or T1) that it is connected to.

Network Address Translation (NAT) provides a mechanism to

1. Decrease the number of routable IPs required
2. Reuse IP address space across multiple PAS foundations

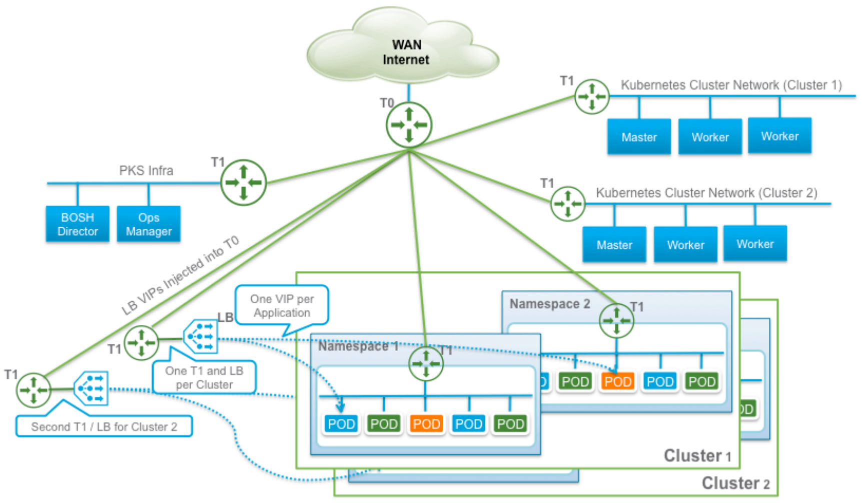


Figure 64 PKS NAT Design

While NSX-T Data Center supports NAT on both T0 and T1 routers, recommendation is to use the T0 router for NAT. PKS dynamically creates SNAT rules, for each namespace that is created and places them on the T0 Router.

NAT cannot be used with an Active-Active Edge deployment as it’s a service. Hence, Edge should be deployed in Active-Standby mode if planning to leverage NST-X NAT service.

10.2.4 Load Balancer

NSX-T Data Center, since release 2.1, comes with built-in Load Balancer functionality. NSX-T Data Center provides both L4 and L7 Load Balancer capabilities. However, L7 Load Balancer functionality is limited to HTTP 1.0 and 1.1.

Load Balancer cannot be used in Active-Active edge design as it’s a service. Hence, Edge's should be deployed in Active-Standby mode if planning to use Load Balancer.

PKS uses Load Balancer in both the Routed and NAT'd topologies.

11 Design Options

Following are some of the design options based on the above building blocks. These are tuned to address PKS specific requirements. Please refer to the NSX-T Data Center design guide for in-depth information on deployment recommendations.

11.1 Collapsed Management and Edge Resources Design

The following design uses multiple hosts for both the compute cluster and the mgmt/edge cluster. This design provides basic fault tolerance at a host level. However, since all the nodes are in the same rack, its not designed to tolerate rack level failures.

In this design, management and edge clusters are still collapsed into a single cluster. Hence, this is not designed for performance from the edge perspective.

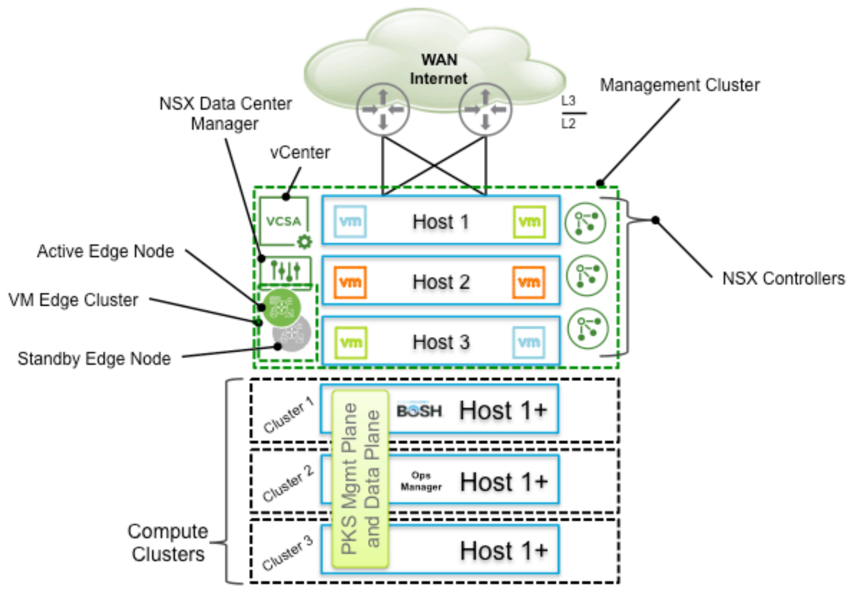


Figure 65 Collapsed Management and Edge Clusters Design

11.2 Dedicated Management and Edge Resources Design

Following is the recommended design for production deployments. It follows the vSphere enterprise design deployment. This design requires at least 3 nodes for the vSphere management components. In this design,

- vCenter, NSX-T Data Center Manager are placed on Management cluster with vSphere HA enabled to protect from host failure and also provide resource reservation
- NSX-T Data Center Controllers are placed on separate hosts with anti-affinity setting and resource reservation
- Edge nodes are placed in their own cluster – spread across multiple hosts. Based on workload and

performance requirements, edge nodes could be either virtual or bare metal.
- BOSH, Ops Manager and the all of the PKS components will reside on the compute cluster

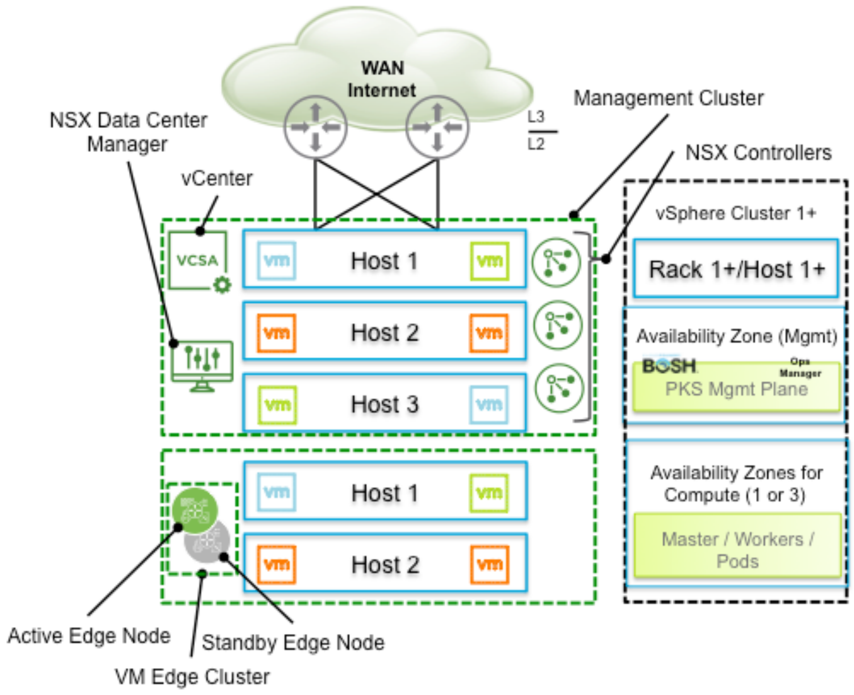


Figure 66 Dedicated Management and Edge Clusters Design

With this design approach, if using a single vSphere cluster, then use vSphere Resource Pools to carve out 1 Availability Zone for PKS Mgmt Plane components and 1 or 3 Availability Zones for PKS Data Plane components. If there are four vSphere clusters, then each cluster could have an independent resource pool that is used as the availability zone. One for the Mgmt Availability Zone for PKS Mgmt Plane and 3 for the Compute Availability Zones.

12 PAS and PKS on NSX-T Data Center

12.1 Logical Layout

Following image shows the placement of various logical components when combining PAS and PKS on the same NSX-T Data Center cluster.

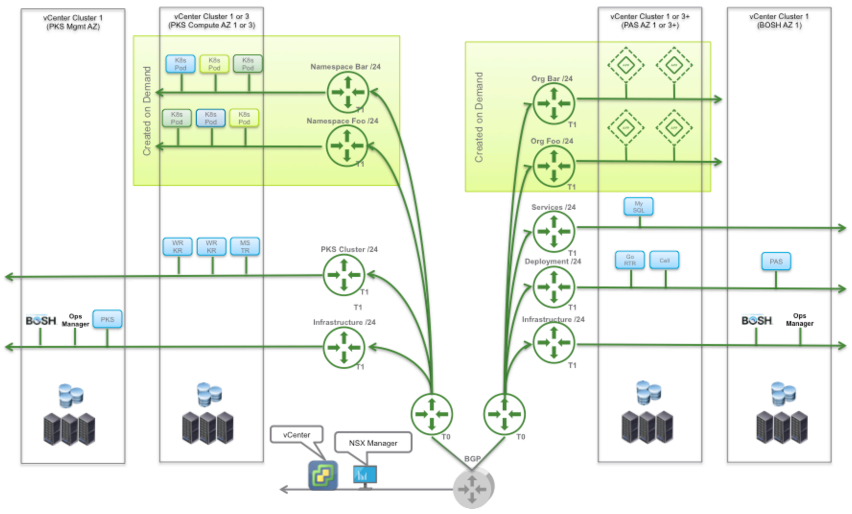


Figure 67 PAS and PKS on NSX-T Data Center Data Center

Above image shows only one PKS Management Plane and one PAS foundation. This may be easily extended to support multiple PAS foundations and one PKS Management Plane. Each cluster would have their own independent logical switches for infrastructure, services, deployment and cluster etc., and will share the same T0. Having multiple cluster and foundations under the same T0 helps keep the traffic within the same NSX-T Data Center domain.

12.2 Design Options

12.2.1 Dedicated Management and Edge Nodes Design

Following is the recommended design for production deployments. As mentioned earlier, unlike the vSphere

Enterprise Design guidelines, in this design, vCenter clusters are not striped across racks but are applied on one or more racks. For the PKS portion, recommendation is to stripe the cluster across the racks. PKS does not provide HA today and striping the cluster across provides some level of HA from the IaaS perspective.

This design requires at least 3 nodes for the vSphere management components. In this design,

- vCenter, NSX-T Data Center Manager are placed on Management cluster with vSphere HA enabled to protect from host failure and also provide resource reservation
- NSX-T Data Center Controllers are placed on separate hosts with anti-affinity setting and resource reservation
- Edge nodes are placed in their own cluster – spread across multiple hosts. Based on workload and performance requirements, edge nodes could be either virtual or bare metal. Note: PKS currently does not support Bare Metal Edge.
- BOSH, Ops Manager and rest of the PKS components are placed in the compute cluster. Note: BOSH is placed on a separate AZ using a separate resource pool on the vSphere Clusters.
- Unlike general design guidance, vSphere clusters here are not striped across the racks. This is acceptable only if running PAS and PKS workloads exclusively. If PAS and PKS workloads are mixed with traditional applications within the same cluster, then the recommendation is to go with the general design guidance.

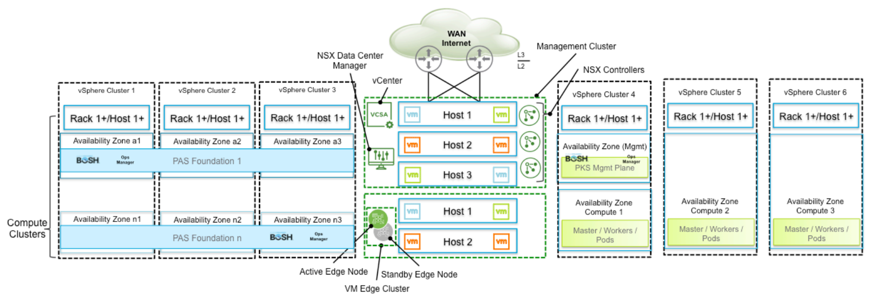


Figure 68 PAS and PKS on NSX-T Data Center Data Center

13 References

1. Agile Product Development https://hbr.org/1986/01/the-new-new-product-development-game
2. Pivotal: https://pivotal.io/
3. PKS Datasheet: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/datasheet/services/vmware-pivotal-container-service-datasheet.pdf
4. NSX-T Data Center Reference Design Guide: https://communities.vmware.com/docs/DOC-37591