Microservices Platform

Installation Overview

Cisco Policy Suite (CPS) architecture is now designed to use the Docker container technology.

This creates a platform that is more flexible, easier to adopt for new products, easier to deploy, and has much less rigidity with respect to its organization. Other benefits are easier upgrade and patching, and a much crisper separation of the “platform” common components versus the “application” components.

This document is intended to describe how the system is launched, and what inputs are required for the collection of Docker engine VMs to self-organize.

Before You Begin

This document makes the following assumptions:

- OpenStack as a target environment.
- Use of Nova as it is the most basic of all deployment approaches.

For more information about OpenStack, see the OpenStack documentation at https://docs.openstack.org/.

Before you set up CPS vDRA, perform the following steps:

1. Create the Project and User under which the system will be launched.
2. Upload the base VM image into Glance.

3. Upload the distribution ISO into Glance.
4. Create a Cinder volume from the distribution ISO.
5. Create an empty 20 GB Cinder volume for the Master VM.
6. Create one 80 GB/120 GB Cinder volume for each VM that will be supporting “control” services.
   - This will generally require three “control” Cinder volumes.
7. Create all the required Network entities, configured appropriately for the CPS Microservices platform.

Note Any config server addition or scale up must be done in maintenance window only as the operation may cause some timeouts in production.

VM Roles in CPS vDRA

The following table describes the VMs and their role in CPS vDRA:

Table 1: VM Names and Roles

VM Name	Role
Master	master
Control	control
DRA Director	dra-director
DRA Worker	dra-worker
Mongo DB (binding)	mongo-node - persistent router, Persistent DB.

Hardware Requirements for VMs

The following table describes the hardware requirements for binding and non-binding CPS vDRA VMs:

Table 2: Hardware Requirements

Role	CPU	RAM (GB)
master	8	32
control-0	8	32
control-1	8	32
dra-director	16	64

Role	CPU	RAM (GB)
dra-worker	8	32
persistence-router	8	32
persistence-db	8	64

A standard deployment of CPS vDRA includes the following VMs:

- For vDRA VNF: one cluster manager, two management VMs, nine DRA Directors, eight DRA Workers.
- For Binding vDRA VNF: one cluster manager, two management VMs, five Persistent Routers, 12 DB VMs.

The number of directors, workers, routers, and database VMs varies, depending upon deployment needs.

Note vDRA supports the failure of one virtual machine in the set of master, control-0, and control-1 virtual machines. Therefore, these virtual machines cannot reside on the same ESXi Server.

Network Requirements For VMs

The following table describes the network requirements for each VM:

Table 3: Network Requirements for VMs

VM	Networks Required
Cluster Manager	Internal Management Replication
Management	Internal Management
DRA Directors	Internal Management/Diameter Traffic VLAN
DRA Workers	Internal
Persistent Routers	Internal
Persistent Database	Internal Replication

VM	Networks Required
Arbiter	Internal Management Replication

Protocols and Port Ranges

For each tenant in OpenStack, configure a security group and the protocol with a port range.

The following table describes the port range for each protocol:

Table 4: Port Ranges for Protocols

Protocol	Port Range
TCP	22 (SSH)
TCP	80 (HTTP)
TCP	443 (HTTPS)
TCP	636
TCP	2024
TCP	2375-2376
TCP	3375-3376
TCP	3868
TCP	5000
TCP	5001
TCP	5003
TCP	6443
TCP	6783
TCP	7443
TCP	7946
TCP	8000
TCP	8008
TCP	8080
TCP	8400

Protocol	Port Range
TCP	8500
TCP	8888
TCP	9100
TCP	9210
TCP	9212
TCP	9213
TCP	9443
TCP	12375
TCP	27017-27047
UDP	4789
UDP	6783-6784
UDP	7946
TCP	12375

Launching VMs

To bring up the system you must launch the VMs.

CPS vDRA includes the following two types of VMs:

- Master VM : hosts critical central system services such as the Registry and the Orchestrator.
- Docker engine VM (also sometimes called a Worker)

All VMs are launched with cloud-init configuration injected into them. This is done with the “config drive” mechanism in OpenStack, and it conveys all the information needed for the VM to start successfully.

Launch Master VM

The Master VM is the main VM for the whole system and requires a cloud-init configuration file.

Some of the configuration details in a cloud-init configuration are:

- Users: non-root administrative user details for the ‘cps’ user that is created and assigned to the ‘docker’ group. The user also has an SSH key injected. The SSH key is user-supplied and should be unique for every installation.
- Password: password for the ‘cps’ user. The password may be removed for all production installations.
- Default path for cps.pem is /etc/puppet/modules/qps/templates/certs

- Configuration details in JSON format (swarm.json) for the system to start successfully.
- Informational file: details of the product and VM within that product they are working with when they log in via SSH.

For an example of a cloud-init configuration file for a Master VM, see Master Cloud Init Example.

swarm.json

The following table describes the configuration information that is included in a swarm.json file:

Table 5: JSON Parameters

Parameters	Description
role	Defines the “role” that this VM will be playing in the system. The role is used to map into the “deployment plan” to select the “scheduling slots” that are available on this particular VM.
identifier	This value combines the “role” with an “index number” on a per-role basis to create a unique identifying string for the deployed VM.
init	Name of the product-specific “initialization image”. This is the image to be launched by the boot scripts when the platform is established and it is time to start the application.
deployment_name	Name specific to a given deployment instance, used to identify the specific deployment, useful in the event that multiple instances of a given product (eg: PCRF) are deployed in the same environment. (This is also used by ESC.)
master	IP address of the Master VM, and must be communicated to all VMs
network	CIDR for the internal network.
registry	IP and port for the cluster-internal Docker registry. This is used by VMs to access the images deployed for the system.
scheduler	Style of scheduler to be used by the Orchestrator. Acceptable values are “ha” and “aio”, which launch the Orchestrator in High-Availability or All-In-One modes respectively. This defaults to “ha” if no orchestrator is specified.
reinitialize_data	Flag which, if set to 1, erases all existing data (primarily on the ‘control’ VMs) so that the system is started fresh. In most normal non-development situations this would be set to 0.

Parameters	Description
zing	Indicates whether, for those VMs where it is appropriate, the Azul Zing JVM should be used rather than the normal JVM.
weavePw	Password used to encrypt the Weave traffic.
tenant	(ESC only) Identifies the OpenStack tenant under which the system is being launched
esc	(ESC only) IP address of the ESC VM.
escUser	(ESC only) Username required to access the ConfD data on the ESC VM.
escPw	(ESC only) Password required to access the ConD data on the ESC VM.

Nova Boot Command

The Nova boot command to launch the Master VM (in this example) looks like the following:

```
nova boot --config-drive true --user-data=node-master-0.cfg \
  --flavor=cps.medium --image=docker-host-1.13.1-2 \
  --nic net-id="$Internal,v4-fixed-ip=172.16.2.11" \
  --nic net-id="$Management,v4-fixed-ip=172.18.11.121" \
  --block-device id=${iso},source=volume,dest=volume,device=/dev/vdb \
  --block-device id=${volume},source=volume,dest=volume,device=/dev/vdc \
  --availability-zone nova \
  --security-groups esc-security-group \
  docker-pcrf-master-0
```

The variables in the command are:

- $Internal: The OpenStack UUID of the “Internal” network.
- $Management: The OpenStack UUID of the “Management” network.
- $iso: The OpenStack UUID of the Cinder volume which contains the deployment ISO data.
- $volume: The OpenStack UUID of the Cinder volume to be used for persistent storage.

Launch Engine VM

The launching of a Docker engine VM (aka: Worker or Engine) is identical, though with less information injected into the ‘swarm.json’ file. Worker VMs might or might not have fixed IP addresses, and there is an assumption that if there isn’t a need for a fixed address that the interface will get it’s address via DHCP. This is primarily true for interfaces on the “internal” network, but there is no prohibition against doing it on other networks.

An example of the cloud-init file for an example Worker VM can be seen in Worker Cloud Init Example. The values included in the worker ‘swarm.json’ file are:

- role
- identifier

- master
- network

In the simplest case a VM is launched exclusively on the “internal” network, assumed to DHCP, and has no Cinder volumes. The Nova boot command for that situation looks like:

```
nova boot --config-drive true --user-data=node-pcrf-0.cfg \
  --flavor=cps.medium --image=docker-host-1.13.1-2 \
  --nic net-id="$Internal" \
  --availability-zone nova \
```

Deployment Matrix

The following tables describe the minimum deployment for the DRA and DRA Binding applications.

The instances are the default choices for a minimal deployment. For those instances that have “scalable” set to YES, new instances can be launched to additional capacity. Where Cinder volumes are indicated, it is the case that each instance will have its own Cinder volume as defined in the table.

DRA

Table 6: Minimum Deployment for DRA

Role	Instances	Cinder Volumes	Scalable
master	master-0	/mnt/iso - /dev/vdb /mnt/install - /dev/vdc /mnt/install - /dev/vdd	NO
control	control-0 control-1	/data - /dev/vdb	NO
diameter-endpoint	diameter-endpoint-0 diameter-endpoint-1	None	YES
binding	binding-0 binding-1	None	YES

DRA Binding

Table 7: Minimum Deployment for DRA Binding - Microservices Platform

Role	Instances	Cinder Volumes	Scalable
master	master-0	/mnt/iso - /dev/vdb /mnt/install - /dev/vdc /mnt/install - /dev/vdd	NO

Role	Instances	Cinder Volumes	Scalable
control	control-0 control-1	/data - /dev/vdb	NO
mongo-node	router-0 router-1	None	YES
mongo-node	db-app-0 db-app-1 db-app-2	None	YES

Startup Sequence

All VMs can be started at any time, in any order. The Worker VMs pause until the Master VM is ready to field their requests. Therefore, no matter the mechanism or VNFM used to launch the system ordering is not an issue. During the boot process normal OpenStack bootup activities occur. This includes all boot-time initialization, the execution of cloud-init, and the triggering of SystemD units. Those are all standard activities and the description of the startup sequence will begin upon the execution of the SystemD unit which is responsible for kicking off the CPS Microservices system. That unit is named “cpsinstall.service” and is staged to occur at the proper time in the Linux startup activities.

The “cpsinstall.service” is extremely simplistic, and performs one operation. It invokes the “/root/bootstrap.sh” script. This is true for Worker VMs as well as for the Master VM. Reading through this script will show you that it performs the following actions:

- Reads the values from the ‘swarm.json’ and sets them as environment variables
- Sets the queuing for RPS
- Determines if the VM is a Master VM, or Worker
- If the VM is a Master, do the following:
  - Mount the ISO from /dev/vdb
  - Mount the persistent Cinder volume from /dev/vdc
  - Perform ‘fsck’ and ‘resize2fs’ operations on the Cinder volume
  - Invoke the ‘install-master.py’ script
- If the VM is a Worker, do the following:
  - Wait for the Master to indicate it is alive
  - Mount the persistent Cinder volume from /dev/vdb
  - Perform ‘fsck’ and ‘resize2fs’ operations on the Cinder volume
  - Pull the ‘install-worker.py’ script from the Master with curl
  - Invoke the ‘install-worker.py’ script

The two install Python scripts perform the various install/startup operations for their VMs. Each of those actions will be detailed in its own section.

install-master.py

This script is specific to only the Master VM, and performs tasks that put the Master in play. The tasks will be outlined below as a sequence of bullet points.

- Condition the target system
  - If there is a /dev/vdc device first determine if it is formatted, and if not format the volume to be an EXT4 filesystem.
  - Remove all data from the mounted file system (/mnt/install) because the only time this runs on a Master VM is when it is booting, and that always calls for a fresh installation.
  - Copy all ISO data to the writable filesystem (/mnt/install).
  - Create the ‘cps-app’ system user.
- Load the configuration from the ‘/root/swarm.json’ file.
  - During loading combine the values from ‘/mnt/iso/release.json’ file
- Write the updated configuration to ‘/mnt/install/swarm.json’
- Stop the timesync daemon.
- Configure the Docker Engine daemon appropriately for operation on the Master.
- Restart the Docker service.
- Set the VM’s hostname to match the VM’s system-internal identifier.
- Install a locally-provided image from the saved image file ‘cps-registry-latest.tgz’.
- Launch the registry container.
- Launch the registry upgrade container.
- Launch Weave.
- Launch the Engine Proxy container.
- Launch the Personality service.
- Launch the product-specific initialization container identified by the ‘init’ value in ‘swarm.json’.

At this point control for the behavior of the system passes out of the startup processes and into the hands of the initialization container. For now, this ultimately leads to the invocation of the product specific Orchestrator. It is the Orchestrator that is responsible for launching all the remaining containers which make up the deployed application.

install-worker.py

This script is run for each of the Worker VMs that are launched. Operations on the Worker are very different from the Master, and that will be reflected in the steps detailed below.

- Load configuration information from the '/root/swarm.json' file for the VM.
- Read the release information from the Master VM, and incorporate it into an updated '/root/swrm.json' file.
- Determine the interface and IP for the Internal network
- If an 'identifier' value is not set in the provided '/root/swarm.json' file, create a unique engine identifier value.
- Create the 'cps-app' system user.
- If the role is 'control' then :
  - If there is a /dev/vdc device first determine if it is formatted, and if not format the volume to be an EXT4 filesystem.
  - Mount /dev/vdc on /data.
  - If 'reinitialize_data' is set, clear out the /data filesystem.
- If 'zing' is set and the role is 'policy', enable Zing operation.
- Set the hostname to match the 'identifier'.
- Stop the timesync daemon.
- Configure the Docker engine to include information about this Worker instance (it's identifier and it's network interface) so that the Orchestrator can know how to use the Worker in the system.
- Launch Weave

At this point the Worker VM is ready to accept containers. This will conclude the low-level startup process. From this point forward additional startup is handled by the Orchestrator and can vary from product to product.

External Port Matrix

The following table lists the services and ports are available to external users and applications in CPS vDRA .

It is recommended that connectivity to these ports be granted from the appropriate networks that require access to the below services.

Service	Port
Ssh	22/tcp
https	443/tcp
xinuexpansion4	2024/tcp
Upnp	5000/tcp
http	8008/tcp
http-proxy	8080/tcp