Pods and Services

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Product(s) or Functional Area	5G-PCF
Applicable Platform(s)	SMI
Feature Default Setting	Enabled – Always-On
Related Changes in this Release	Not Applicable
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
First introduced.	2020.01.0

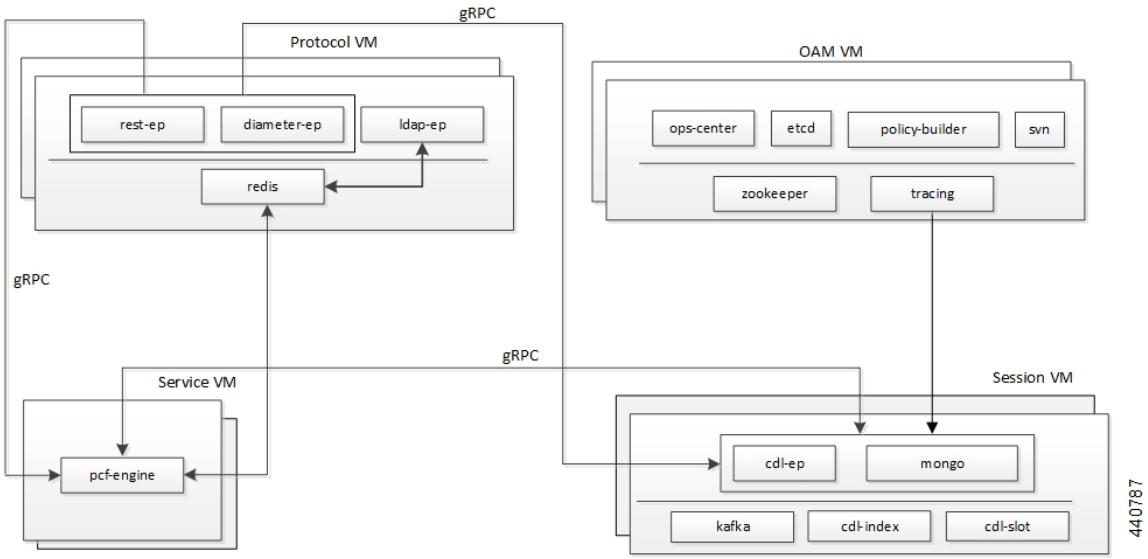
Feature Description

The PCF is built on the Kubernetes cluster strategy, which implies that it has adopted the native concepts of containerization, high availability, scalability, modularity, and ease of deployment. To achieve the benefits offered by Kubernetes, PCF uses the construct that includes the components such as pods and services.

Depending on your deployment environment, PCF deploys the pods on the virtual machines that you have configured. Pods operate through the services that are responsible for the intrapod communications. If the machine hosting the pods fail or experiences network disruption, the pods are terminated or deleted. However, this situation is transient and PCF spins new pods to replace the invalid pods.

The following workflow provides a high-level visibility into the host machines, and the associated pods and services. It also represents how the pods interact with each other. The representation might defer based on your deployment infrastructure.

Figure 1: Communication Workflow of Pods



The Protocol VM hosts the rest-ep, diameter-ep, and ldap-ep pod that governs the ingress (incoming) and egress (outgoing) traffic on the interfaces. The pods responsible for the operations and management processes reside in the OAM VM and, the Service VM hosts the pcf-engine. The session VMs hosts the pods that operate as the databases to store the data accessed by the pods. The illustration also depicts the services which the pods use to channel the interactions. The pods communicate over the gRPC interface.

Note Typically, multiple instances of the Protocol and OAM VMs are created to ensure resiliency.

Kubernetes deployment includes the kubectl command-line tool to manage the resources in the cluster. You can manage the pods, nodes, and services using the CLI.

For performing the maintenance activities, you can use the **kubectl drain** command to withdraw a node voluntarily. This command prepares the node by evicting or assigning the associated pods to another node with sufficient resources. You can run the **kubectl drain** on individual or multiple nodes concurrently.

For generic information on the Kubernetes concepts, see the Kubernetes documentation.

For more information on the Kubernetes components in PCF, see the following.

- Pods, on page 3
- Services, on page 5

Pods

Pod is a process that runs on your Kubernetes cluster. Pod encapsulates a granular unit known as a container. A pod can contain one or multiple containers.

Kubernetes deploys one or multiple pods on a single node which can be a physical or virtual machine. Each pod has a discrete identity with an internal IP address and port space. However, the containers within a pod can share the storage and network resources.

The following table lists the pod names and the hosts on which they are deployed depending on the labels that you assign. For information on how to assign the labels, see Associating Pods to the Nodes, on page 8.

Table 3: PCF Pods

Pod Name	Description	Host Name
admin-db	Acts as the MongoDB router pod for the Admin database.	Session
api-pcf-ops-center	Functions as the confD API pod for the PCF Ops Center.	OAM
cdl-ep-session-c1	Provides an interface to the CDL. **Note** Configuration changes to the CDL endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.	Session
cdl-index-session	Preserves mapping information of the keys to the session pods.	Session
cdl-slot-session-c1	Operates as the CDL Session pod to store the session data.	Session
cps-license-manager	Acts as the PCF License Manager.	OAM
crd-api-pcf-pcf-engine-app-pcf-<*n*>-mjgxp	Hosts the CRD APIs.	Protocol
db-admin	Acts as the replica set pod for the Admin database.	Session
db-admin-config	Acts as the replica set pod that stores the Admin database configuration.	Session
db-spr-config	Operates as the replica set pod that stores the SPR database configuration.	Session
db-spr1	Functions as the replica set pod that preserves the SPR database.	Session

Pod Name	Description	Host Name
diameter-ep-rx-rx	Contains the Diameter stack details and acts as the endpoint. **Note** Configuration changes to the diameter endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.	Protocol
documentation	Contains the documentation.	OAM
etcd-pcf-etcd-cluster	Hosts the etc-d for the PCF application.	OAM
grafana-dashboard-cdl	Contains the Grafana metrics for CDL.	OAM
grafana-dashboard-pcf	Contains the Grafana metrics for PCF.	OAM
kafka	Hosts the Kafka details for the CDL replication.	Protocol
ldap-ep	Operates as an LDAP client to establish communication with an external LDAP server. **Note** Configuration changes to the LDAP endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.	Protocol
network-query	Operates as the utility pod to determine the route IP for the Diameter outbound messages.	OAM
ops-center-pcf-ops-center	Acts as the PCF Ops Center.	OAM
patch-server-pcf-cnat-cps-infrastructure	Operates as the utility pod for patching the PCF JAR files.	OAM
pcf-day0-config-pcf-pcf -engine-<*n*>-rchg2	Dedicated for performing the Day-0 configuration for PCF.	OAM
pcf-engine-pcf-pcf-engine-app-pcf	Operates as the PCF Engine. **Note** Configuration changes to the PCF Engine endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.	Service

Pod Name	Description	Host Name
pcf-rest-ep	Operates as a REST endpoint for PCF. **Note** Configuration changes to the REST endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.	Protocol
policy-builder-pcf-pcf-engine-app	Operates as the Policy Builder for PCF.	OAM
redis-keystore	Operates as the REDIS Index.	Protocol
redis-queue	Operates as the REDIS IPC.	Protocol
rs-controller-admin	Responsible for the replication controller for Admin database.	Session
rs-controller-admin-config	Operates as a replication controller for the Admin database configuration.	Session
rs-controller-spr-config	Operates as a replication controller for SPR database configuration.	Session
rs-controller-spr1	Operates as a replication controller for the SPR database.	Session
smart-agent-pcf-ops-center	Operates as the utility pod for the PCF Ops Center.	OAM
svn	Stores all the PCF XMI configuration files.	OAM
svn-ldap	Stores the LDAP endpoint configuration which is configured through the ops-center.	Protocol
swift-pcf-ops-center	Operates as the utility pod for the PCF Ops Center.	OAM
traceid-pcf-pcf-engine	Stores the subscriber tracing details.	OAM
zookeeper	Assigned for the Zookeeper.	OAM

Services

The PCF configuration is composed of several microservices that run on a set of discrete pods. Microservices are deployed during the PCF deployment. PCF uses these services to enable communication between the pods. When interacting with another pod, the service identifies the pod's IP address to initiate the transaction and acts as an endpoint for the pod.

The following table describes the PCF services and the pod on which they run.

Table 4: PCF Services and Pods

Service Name	Pod Name	Description
admin-db	admin-db-0	Serves to process the MongoDB-specific router messages.
cps-diameter-inbound-rx-rx-rx	cps-diameter-ep	Transmits the Rx messages to the Diameter endpoint. You can set an external IP address for the service.
crd-api-pcf-pcf-engine-app-pcf	crd-api	Processes the CRD API calls.
datastore-ep	datastore-ep	Processes the CDL endpoint calls.
datastore-ep-session	ngn-datastore-ep	Responsible for the CDL session.
datastore-notification-ep	pcf-engine	Responsible for sending the notifications from the CDL to the engine.
diameter-engine	pcf-engine	Acts as the Diameter endpoint to pcf-engine.
documentation	documentation	Processes the documentation API calls.
etcd	pcf-etcd-cluster	Processes the etc-d API.
etcd-pcf-etcd-cluster-<n>	pcf-etcd-cluster	Processes the etc-d stateful sets.
grafana-dashboard-cdl	grafana-dashboard-cdl	Responsible for managing the Grafana dashboard for inputs from the CDL.
grafana-dashboard-pcf	grafana-dashboard-pcf	Manages the Grafana dashboard for PCF.
helm-api-pcf-ops-center	helm-api	Manages the Ops Center API.
kafka	kafka	Processes the Kafka messages.
mongo-admin-<n>	db-admin-0	Responsible for the Admin database stateful sets.
mongo-admin-config-<n>	db-admin-config-0	Responsible for the Admin database configuration stateful sets.
mongo-spr-config-<n>	db-spr-config-0	Responsible for the SPR database configuration stateful sets.
mongo-spr1-<n>	db-spr1-0	Responsible for the SPR database stateful sets
ops-center-pcf-ops-center	ops-center	Manages the PCF Ops Center.
patch-server-pcf-cnat-cps-infrastructure	patch-server	Maintains the patch repository.

Service Name	Pod Name	Description
pcf-day0-config-pcf-pcf-engine-app-pcf	pcf-day0-config	Manages the Day-0 configuration.
pcf-engine	pcf-engine	Processes the API calls to pcf-engine.
pcf-rest-ep	pcf-rest-ep	Acts as the http2 request/response to the REST endpoint. You can set an external IP address for the service.
policy-builder-pcf-pcf-engine-app-pcf	policy-builder	Manages the Policy Builder's request/response messages.
redis-keystore-<n>	redis-keystore-0	Manages the REDIS keystore stateful set.
redis-queue-<n>	redis-queue-0	Processes the REDIS queue stateful set.
rs-admin	replica-set admin	Manages the replica set for Admin database.
rs-admin-config	replica-set admin-config	Manages the replica set for the Admin database configuration.
rs-spr-config	replica-set spr-conifg	Manages the replica set for the SPR configuration.
rs-spr1	replica-set sp1	Manages the replica set for the SPR database.
smart-agent-pcf-ops-center	smart-agent-pcf-ops-center	Responsible for the Ops Center API.
svn	cps-subversion	Responsible for the SVN API calls.
swift-pcf-ops-center	swift-pcf-ops-center	Responsible for the Ops Center API.

Ports and Services

PCF uses different ports for communication purposes. The following table describes the default ports and the associated services.

Table 5: Ports and Services

Port	Service	Usage
22	SSH	SMI uses this port to communicate with the virtual machines.
80	HTTP	SMI uses this port for providing Web access to CLI, Documentation, and TAC.
443	SSL/HTTP	SMI uses this port for providing Web access to CLI, Documentation, and TAC.

Port	Service	Usage
2024	SSH	SMI accesses the ConfdD CLI through this port.
3868	TCP	PCF uses this port as the default Diameter Endpoint on a public port.
6443	HTTP	SMI uses this port to communicate with the Kubernetes API server.
8080	HTTP	PCF uses this port to communicate with the Keep Alive API Interface on a public network.
9082	HTTP	PCF uses this port to access the SBI Interface on a public network. The Keepalive monitors the health of the container on this port. If the port is not accessible, then the kubectl restarts the container to restore the service.
9299	HTTP	SMI uses this port to communicate with the Prometheus Service.
9885	TCP	Default port that operates as the PCF Service gRPC endpoint on a private network.
10250	SSL/HTTP	SMI uses this port to communicate with Kubelet.
10256	HTTP	SMI uses this port to interact with the Kube proxy.

Limitations

In the current release, this feature has the following limitation introduced by Kubernetes:

When removing a node using the **kubectl drain** command, the pods managing the inbound traffic such as pcf-rest-ep, pcf-ldapserver-ep, and diameter-ep-rx-protocol cannot be assigned to another node. The workload of these pods' cannot be scheduled to another node since the traffic is routed through persistent connections that do not support load balance. As a result, the Grafana dashboard does not display the Transaction Per Second (TPS) for these pods.

Configuration Support for Pods and Services

This section describes how to associate pods to node and view the pod-related information using the following steps:

1. Associating Pods to the Nodes
2. Viewing the Pod Details and Status

Associating Pods to the Nodes

This section describes how to associate a pod to the node based on their labels.

After you have configured a cluster, you can associate pods to the nodes through labels. This association enables the pods to get deployed on the appropriate node based on the key-value pair.

Labels are required for the pods to identify the nodes where they must get deployed and to run the services. For example, when you configure the protocol-layer label with the required key-value pair, the pods get deployed on the nodes that match the key-value pair.

To associate pods to the nodes through the labels, use the following configuration:

```
config
  label
    cdl-layer
      key key_value
      value value
    oam-layer
      key key_value
      value value
    protocol-layer
      key key_value
      value value
    service-layer
      key key_value
      value value
      end
```

NOTES:

- If you opt not to configure the labels, then PCF assumes the labels with the default key-value pair.
- **cdl-layer** – Configures the key-value pair parameters for the CDL.
- **oam-layer** – Configures the key-value pair parameters for the OAM layer.
- **protocol-layer** – Configures the key-value pair parameters for the protocol layer.
- **service-layer** – Configures the key-value pair parameters for the service layer.

Viewing the Pod Details and Status

This section describes how to view the pod details.

If the service requires additional pods, PCF creates and deploys the pods. You can view the list of pods that are participating in your deployment through the PCF Ops Center.

You can run the kubectl command from the master node to manage the Kubernetes resources.

- To view the comprehensive pod details, use the following configuration:

  ```
  kubectl get pods -n pcf pod_name -o yaml
  ```

  The pod details are available in YAML format.

  The output of this command results in the following information:

  - The IP address of the host where the pod is deployed.
  - The service and application that is running on the pod.
  - The ID and name of the container within the pod
  - The IP address of the pod

- The current state and phase in which the pod is.
- The start time from which pod is in the current state.

- To view the summary of the pod details, use the following configuration:

```
kp -get pods -o wide
```

States

Understanding the pod's state lets you determine the current health and prevent the potential risks.

The following table describes the pod's states.

Table 6: Pod States

State	Description
Running	The pod is healthy and deployed on a node. It contains one or more containers.
Pending	The application is in the process of creating the container images for the pod.
Succeeded	Indicates that all the containers in the pod are successfully terminated. These pods cannot be restarted.
Failed	One ore more containers in the pod have failed the termination process. The failure occurred as the container either exited with non zero status or the system terminated the container.
Unknown	The state of the pod could not be determined. Typically, this could be observed because the node where the pod resides was not reachable.