PCF Rolling Software Update

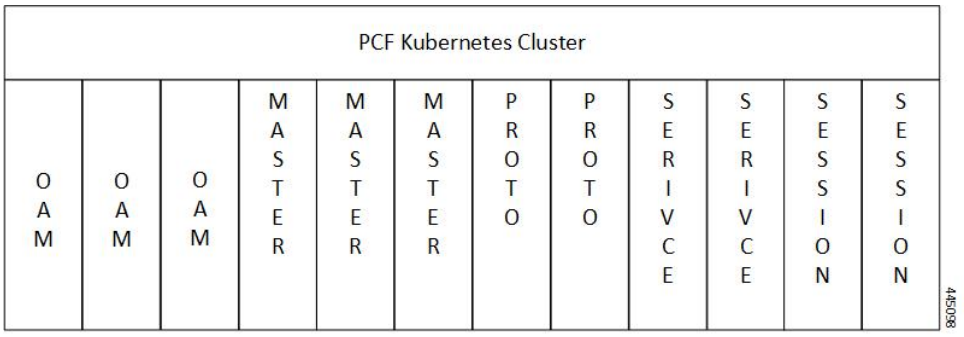
Introduction

The Cisco PCF has a three-tier architecture which consists of Protocol, Service, and Session tiers. Each tier includes a set of microservices (pods) for a specific functionality. Within these tiers, there exists a Kubernetes Cluster comprising of Kubernetes (K8s) master, and worker nodes (including Operation and Management nodes).

For high availability and fault tolerance, a minimum of two K8s worker nodes are required for each tier. You can have multiple replicas for each worker node. Kubernetes orchestrates the pods using the StatefulSets controller. The pods require a minimum of two replicas for fault tolerance.

The following figure depicts a PCF K8s Cluster with 12 nodes – 3 Master nodes, 3 Operations, and Management (OAM) worker nodes, 2 Protocol worker nodes, 2 Service worker nodes, 2 Session (data store) worker nodes.

Figure 1: PCF Kubernetes Cluster



Note

- OAM worker nodes - These nodes host the Ops Center pods for configuration management and metrics pods for statistics and Key Performance Indicators (KPIs).
- Protocol worker nodes - These nodes host the PCF protocol-related pods for service-based interfaces (N7, N28, N36, and NRF) and Diameter Rx Endpoint.
- Service worker nodes - These nodes host the PCF application-related pods that perform session management processing.
- Session worker nodes - These nodes host the database-related pods that store subscriber session data.

Updating PCF

The following section describes the procedure involved in updating the PCF software:

- Rolling Software Update Using SMI Cluster Manager

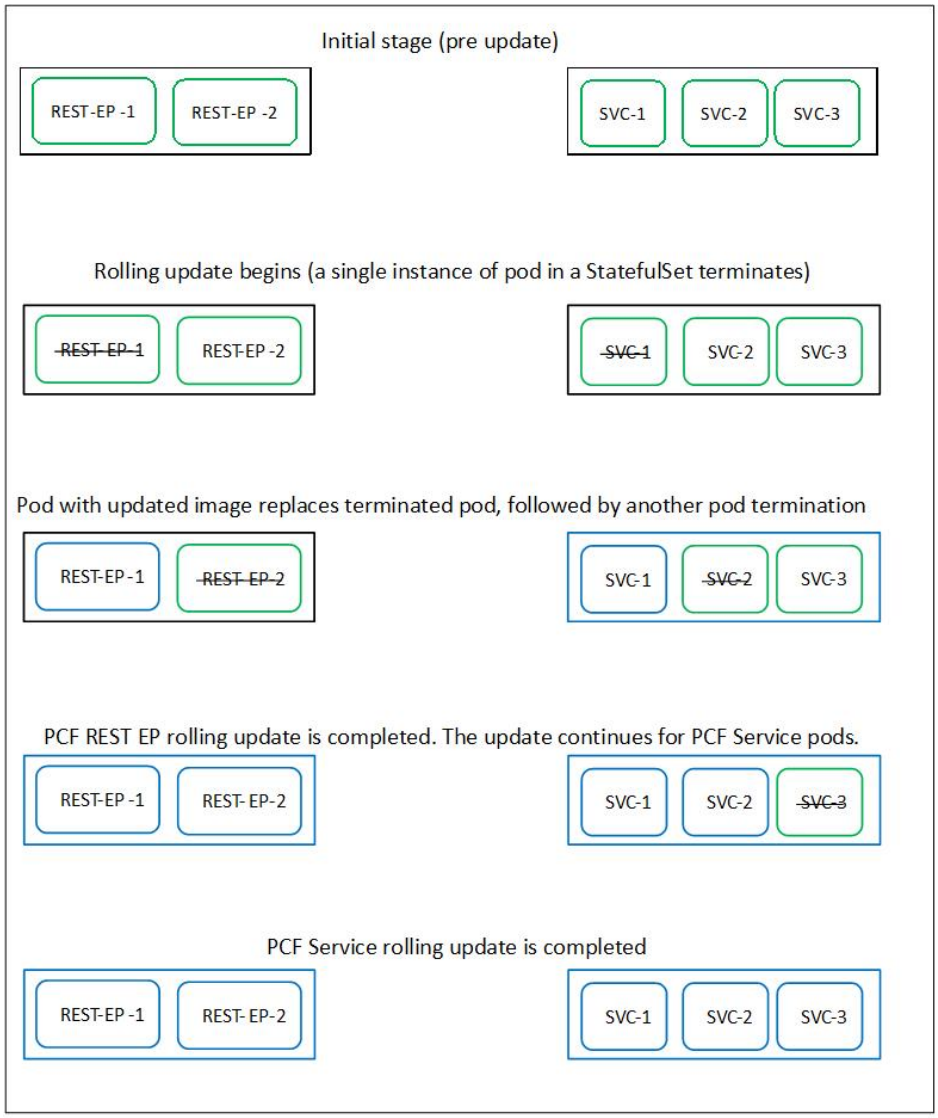
Rolling Software Update Using SMI Cluster Manager

The PCF software update or in-service update procedure utilizes the K8s rolling strategy to update the pod images. In K8s rolling update strategy, the pods of a StatefulSet are updates sequentially to ensure that the ongoing process remains unaffected. Initially, a rolling update on a StatefulSet causes a single pod instance to terminate. A pod with an updated image replaces the terminated pod. This process continues until all the replicas of the StatefulSet are updated. The terminating pods exit gracefully after competing all the ongoing processes. Other in-service pods continue to receive and process the traffic to provide a seamless software update. You can control the software update process through the Ops Center CLI.

Note Each pod needs a minimum of two replicas for high availability. For example, Policy Engine must have 2 Engine replicas. In a worst-case scenario, the processing capacity of the pod may briefly reduce to 50% while the software update is in-progress.

The following figure illustrates a PCF rolling update for PCF REST endpoint pods (two replicas) on Protocol worker nodes along with PCF Service pods (three replicas) on Service worker nodes.

Figure 2: PCF Rolling Update



Prerequisites

The prerequisites for upgrading PCF are:

- All the nodes – including all the pods in the node – are up and running.
- A patch version of the PCF software.

Note Currently, major versions do not support the rolling upgrade. The major version represents the release year, release number, and maintenance number. Cisco follows the versioning format as YYYY.RN.MN such as 2020.03.0.

Important Trigger rolling upgrade only when the CPU usage of the nodes is less than 50%.

PCF Health Check

You need to perform a health check to ensure that all the services are running and nodes are in ready state. To perform a health check:

1. Log in to master node and use the following configuration:

```
kubectl get pods -n smi
kubectl get nodes
kubectl get pod --all-namespaces -o wide
kubectl get pods -n pcf-wsp -o wide
kubectl get pods -n cee-wsp -o wide
kubectl get pods -n smi-vips -o wide
helm list
kubectl get pods -A | wc -l
```

Important Ensure that all the nodes are in the ready state before you proceed further. Use the kubectl get nodes command to display the node states.

Preparing for Upgrade

This section describes the procedure involved creating a backup configuration, logs, and deployment files. To back up the files:

1. Log in to the SMI Cluster Manager Node as an **ubuntu** user.
2. Create a new directory for deployment.

   Example:

```
test@smipcf-cm01:~$ mkdir -p "temp_$(date +'%m%d%Y_T%H%M')" && cd "$_"
```

3. Move all the *pcf* deployment file into the newly created deployment directory.
4. Untar the *pcf* deployment file.

   Example:

```
test@smi1pcf01-cm01:~/temp_08072019_T1651$ tar -xzvf pcf.2020.01.0-1.SPA.tgz
./
./PCF_REL_KEY-CCO_RELEASE.cer
./cisco_x509_verify_release.py
./pcf.2020.01.0-1.tar
./pcf.2020.01.0-1.tar.signature.SPA
./pcf.2020.01.0-1.tar.SPA.README
```

5. Verify the downloaded image.

   Example:

```
test@smi1pcf01-cm01:~/temp_08072019_T1651$ cat pcf.2020.01.0-1.tar.SPA.README
```

Important Follow the procedure mentioned in the *SPA.README* file to verify the build before proceeding to the next step.

Back Up SVN, Policy, and CRD Data

This section describes the procedure involved in creating a backup of SVN, Policy, and CRD data. To perform a backup of SVN and Policy files:

1. Log in to the master node as an **ubuntu** user.
2. Use the following command to retrieve the Policy Builder URL.

   ```
   kubectl get ing -n $( kubectl get namespaces | grep -oP 'pcf-(\d+|\w+)'
    | cut -d\  -f1) | grep policy-builder | awk '{ print $2 }'
   pb.pcf-02-pcf-engine-app-blv02.ipv4address.nip.io
   ```

   Example:

   ```
   ubuntu@ mas01:~/backups_09182019_T2141$ kubectl get ing -n $( kubectl get namespaces |
   grep -oP 'pcf-(\d+|\w+)' | cut -d\  -f1) | grep policy-builder | awk '{ print $2 }'
   ```

   Sample output:

   ```
   pb.pcf-02-pcf-engine-app-blv02.ipv4address.nip.io
   ```

3. Navigate to the Policy Builder home page.
4. Click **Import/Export**.
5. Click **All Data**.
   - **Export URL** - Specifies the export URL.
   - **Export File Prefix** - Specify an appropriate name for the export file.
6. Click **Export**.

Important You can find the exported file in your local **Downloads** directory.

To perform a backup of CRD data:

1. Navigate to the Policy Builder Home page.
2. Click **Custom Reference Data.**
3. Click **Import/Export CRD data.**
4. Click **Export**.

Important You can find the CRD data in your Web browser's **Downloads** directory.

Back Up Ops Center Configuration

This section describes the procedure involved in creating a backup of the Ops Center configurations.

To perform a backup of the Ops Center configurations:

1. Log in to SMI Cluster Manager node as an **ubuntu** user.
2. Run the following command to backup the SMI Ops Center configuration to **/home/ubuntu/smiops.backup** file.

   ```
   ssh -p <port_number> admin@$(kubectl get svc -n smi | grep
   '.*netconf.*<port_number>' | awk '{ print $4 }') "show run | nomore"
   > smiops.backup_$(date +'%m%d%Y_T%H%M')
   ```

   NOTES:

   - **ssh -p <port_number>**: Specifies the port number of the system on which the SMI Ops Center service is running. Use the **Kubectl get service** command to display the ports on which is the services are running.
   - ***netconf.*<port_number>**': Specifies the port number of the system on which the Netconf service is running.

3. Run the following command to backup the CEE Ops Center configuration to **/home/ubuntu/ceeops.backup** file.

   ```
   ssh admin@<cee-vip> "show run | nomore" > ceeops.backup_$(date
   +'%m%d%Y_T%H%M')
   ```

   NOTES:

   - **cee-vip**: Specifies the CEE VIP that is configured in the SMI Ops Center. Use the **show running-config** to display the SMI Ops Center configuration.

4. Run the following command to backup the PCF Ops Center configuration to **/home/ubuntu/pcfops.backup** file.

   ```
   ssh admin@<pcf-vip> "show run | nomore" > pcfops.backup_$(date
   +'%m%d%Y_T%H%M')
   ```

   NOTES:

   - **pcf-vip**>: Specifies the PCF VIP that is configured in the SMI Ops Center. Use the **show running-config** to display the SMI Ops Center configuration.

Back Up CEE and PCF Ops Center Configuration

This section describes the procedure involved in creating a backup of CEE and Ops Center configuration from the master node. To perform a backup of CEE and Ops Center configuration:

1. Log in to the master node as an **ubuntu** user.
2. Create a directory to backup the configuration files.

   ```
   mkdir backups_$(date +'%m%d%Y_T%H%M') && cd "$_"
   ```

3. Back up the PCF Ops Center configuration and verify the line count of the backup files.

```
ssh -p <port_number> admin@$(kubectl get svc -n $(kubectl get namespaces
| grep -oP 'pcf-(\d+|\w+)') | grep <port_number> | awk '{ print $3
}') "show run | nomore" > pcfops.backup_$(date +'%m%d%Y_T%H%M') && wc
-l pcfops.backup_$(date +'%m%d%Y_T%H%M')
```

Example:

```
ubuntu@popcf-mas01:~/backups_09182019_T2141$ ssh -p <port_number> admin@$(kubectl get
svc -n $(kubectl get namespaces | grep -oP 'pcf-(\d+|\w+)') | grep <port_number> | awk
'{ print $3 }') "show run | nomore" > pcfops.backup_$(date +'%m%d%Y_T%H%M') && wc -l
pcfops.backup_$(date +'%m%d%Y_T%H%M')
admin@<admin_ip_address> password: PCF-OPS-PASSWORD
334 pcfops.backup
```

4. Back up the CEE Ops Center configuration and verify the line count of the backup files.

```
ssh -p <port_number> admin@$(kubectl get svc -n $(kubectl get namespaces
| grep -oP 'cee-(\d+|\w+)') | grep <port_number> | awk '{ print $3
}') "show run | nomore" > ceeops.backup_$(date +'%m%d%Y_T%H%M') && wc
-l ceeops.backup_$(date +'%m%d%Y_T%H%M')
```

Example:

```
ubuntu@popcf-mas01:~/backups_09182019_T2141$ ssh -p <port_number> admin@$(kubectl get
svc -n $(kubectl get namespaces | grep -oP 'cee-(\d+|\w+)') | grep <port_number> | awk
'{ print $3 }') "show run | nomore" > ceeops.backup_$(date +'%m%d%Y_T%H%M') && wc -l
ceeops.backup_$(date +'%m%d%Y_T%H%M')
admin@<admin_ip_address> password: CEE-OPS-PASSWORD
233 ceeops.backup
```

5. Move the SMI Ops Center backup file (from the SMI Cluster Manager) to the backup directory.

```
scp $(grep cm01 /etc/hosts | awk '{ print $1
}'):/home/ubuntu/smiops.backup_$(date +'%m%d%Y_T%H%M') .
```

Example:

```
ubuntu@popcf-mas01:~/backups_09182019_T2141$ scp $(grep cm01 /etc/hosts | awk '{ print
$1 }'):/home/ubuntu/smiops.backup_$(date +'%m%d%Y_T%H%M') .
ubuntu@<admin_ip_address> password: SMI-CM-PASSWORD
smiops.backup                                                 100% 9346    22.3MB/s
   00:00
```

6. Verify the line count of the backup files.

Example:

```
ubuntu@popcf-mas01:~/backups_09182019_T2141$ wc -l *
  233 ceeops.backup
  334 pcfops.backup
  361 smiops.backup
  928 total
```

Upgrading the PCF

This section describes the procedures involved in upgrading PCF.

Staging a New PCF Image

This section describes the procedure involved in staging a new PCF image before initiating the upgrade.

To stage the new PCF image:

1. Download and verify the new PCF image.
2. Log in to the SMI Cluster Manager node as an **ubuntu** user.
3. Copy the images to **Uploads** directory.

   ```
   sudo mv <pcf_new_image.tar> /data/software/uploads
   ```

Note The SMI uses the new image present in the **Uploads** directory to upgrade.

4. Verify whether the image is picked up by the SMI for processing from the **Uploads** directory.

   ```
   sleep 30; ls /data/software/uploads
   ```

   Example:

   ```
   ubuntu@popcf-cm01:~/temp_08072019_T1651$ sleep 30; ls /data/software/uploads
   ubuntu@popcf-cm01:~/temp_08072019_T1651$
   ```

5. Verify whether the images were successfully picked up and processed.

   Example:

   ```
   auser@unknown:$ sudo du -sh /data/software/packages/*
   1.6G /data/software/packages/cee.2019.07
   5.3G /data/software/packages/pcf.2019.08-04
   16K /data/software/packages/sample
   ```

Note The SMI must unpack the images into the **packages** directory successfully to complete the staging.

Triggering the Rolling Software Upgrade

The PCF utilizes the SMI Cluster Manager to perform a rolling software update. To update PCF using SMI Cluster Manager, use the following configurations:

Important Before you begin, ensure that PCF is up and running with the current version of the software.

1. Log in to the SMI Cluster Manager console.
2. Run the following command to log in to the SMI Ops Center.

   ```
   ssh -p <port_number> admin@$(kubectl get svc -n smi | grep
   '.*netconf.*<port_number>' | awk '{ print $4 }')
   ```

   Example:

   ```
   ubuntu@popcf-cm01:~$ ssh -p <port_number> admin@$(kubectl get svc -n smi | grep
   '.*netconf.*<port_number>' | awk '{ print $4 }')
   admin@<admin_ip_address> password: SMI-CONSOLE-PASSWORD
   Welcome to the CLI
   admin connected from <admin_ip_address> using ssh on
   ops-center-smi-cluster-manager-85869cf9b6-7j64k
   ```

3. Download the latest TAR ball from the URL.

software-packages download *URL*

Example:

```
SMI Cluster Manager# software-packages download <URL>
```

NOTES:

- **software-packages download** *url* – Specifies the software packages to be downloaded through HTTP/HTTPS.

4. Verify whether the TAR balls are loaded.

   software-packages list

   Example:

   ```
   SMI Cluster Manager# software-packages list
   [ PCF-2019-08-21 ]
   [ sample ]
   ```

   NOTES:

   - **software-packages list** – Specifies the list of available software packages.

5. Update the product repository URL with the latest version of the product chart.

Note If the repository URL contains multiple versions, the Ops Center selects the latest version automatically.

```
config
  cluster cluster_name
   ops-centers app_name PCF_instance_name
        repository url
         exit
       exit
```

Example:

```
SMI Cluster Manager# config
SMI Cluster Manager(config)# clusters test2
SMI Cluster Manager(config-clusters-test2)# ops-centers PCF data
SMI Cluster Manager(config-ops-centers-PCF/data)# repository <url>
SMI Cluster Manager(config-ops-centers-PCF/data)# exit
SMI Cluster Manager(config-clusters-test2)# exit
```

NOTES:

- **cluster** – Specifies the K8s cluster.
- *cluster_name* – Specifies the name of the cluster.
- **ops-centers** *app_name instance_name* – Specifies the product Ops Center and instance. *app_name* is the application name. *instance_name* is the name of the instance.
- **repository** *url* – Specifies the local registry URL for downloading the charts.

6. Run the **cluster sync** command to update to the latest version of the product chart. For more information on **cluster sync** command, see the Important section.

clusters *cluster_name* **actions sync run**

Example:

```
SMI Cluster Manager# clusters test2 actions sync run
```

Important The cluster synchronization updates the PCF Ops Center, which in turn updates the application pods (through **helm sync** command) one at a time automatically.

NOTES:

- **cluster** – Specifies the K8s cluster.
- *cluster_name* – Specifies the name of the cluster.
- **actions** – Specifies the actions performed on the cluster.
- **sync run** – Triggers the cluster synchronization.

Monitoring the Upgrade

You can monitor the status of the upgrade through SMI Cluster Manager Ops Center. To monitor the upgrade status, use the following configurations:

```
config
  clusters cluster_name actions sync run debug true
  clusters cluster_name actions sync logs
  monitor sync-logs cluster_name
   clusters cluster_name actions sync status
   end
```

Example:

```
SMI Cluster Manager# clusters test1 actions sync run
SMI Cluster Manager# clusters test1 actions sync run debug true
SMI Cluster Manager# clusters test1 actions sync logs
SMI Cluster Manager# monitor sync-logs test1
SMI Cluster Manager# clusters test1 actions sync status
```

NOTES:

- **clusters** *cluster_name* – Specifies the information about the nodes to be deployed. *cluster_name* is the name of the cluster.
- **actions** – Specifies the actions performed on the cluster.
- **sync run** – Triggers the cluster synchronization.
- **sync logs** – Shows the current cluster synchronization logs.
- **sync status** – Shows the current status of the cluster synchronization. **debug true** – Enters the debug mode.
- **monitor sync logs** – Monitors the cluster synchronization process.

Important You can view the pod details after the upgrade through CEE Ops Center. For more information on pod details, see Viewing the Pod Details section.

Validating the Upgrade

This section describes the procedures involved in validating the upgrade process.

Viewing the Pod Details

You can view the details of the current pods through CEE Ops Center. To view the pod details, use the following command (in CEE Ops Center CLI):

cluster pods *instance_name pod_name* **detail**

Note

- **cluster pods** – Specifies the current pods in the cluster.
- *instance_name* – Specifies the name of the instance.
- *pod_name* – Specifies the name of the pod.
- **detail** – Displays the details of the specified pod.

The following example displays the details of the pod named *alertmanager-0* in the *PCF-data* instance.

Example:

```
cee# cluster pods PCF-data alertmanager-0 detail
details apiVersion: "v1"
kind: "Pod"
metadata:
  annotations:
    alermanager.io/scrape: "true"
    cni.projectcalico.org/podIP: "<ipv4address/subnet>"
    config-hash: "5532425ef5fd02add051cb759730047390b1bce51da862d13597dbb38dfbde86"
  creationTimestamp: "2020-02-26T06:09:13Z"
  generateName: "alertmanager-"
  labels:
    component: "alertmanager"
    controller-revision-hash: "alertmanager-67cdb95f8b"
    statefulset.kubernetes.io/pod-name: "alertmanager-0"
  name: "alertmanager-0"
  namespace: "PCF"
  ownerReferences:
  - apiVersion: "apps/v1"
    kind: "StatefulSet"
    blockOwnerDeletion: true
    controller: true
    name: "alertmanager"
    uid: "82a11da4-585e-11ea-bc06-0050569ca70e"
  resourceVersion: "1654031"
  selfLink: "/api/v1/namespaces/PCF/pods/alertmanager-0"
  uid: "82aee5d0-585e-11ea-bc06-0050569ca70e"
spec:
  containers:
  - args:
```

```
    - "/alertmanager/alertmanager"
    - "--config.file=/etc/alertmanager/alertmanager.yml"
    - "--storage.path=/alertmanager/data"
    - "--cluster.advertise-address=$(POD_IP):6783"
    env:
    - name: "POD_IP"
      valueFrom:
        fieldRef:
          apiVersion: "v1"
          fieldPath: "status.podIP"
    image: "<path_to_docker_image>"
    imagePullPolicy: "IfNotPresent"
    name: "alertmanager"
    ports:
    - containerPort: 9093
      name: "web"
      protocol: "TCP"
    resources: {}
    terminationMessagePath: "/dev/termination-log"
    terminationMessagePolicy: "File"
    volumeMounts:
    - mountPath: "/etc/alertmanager/"
      name: "alertmanager-config"
    - mountPath: "/alertmanager/data/"
      name: "alertmanager-store"
    - mountPath: "/var/run/secrets/kubernetes.io/serviceaccount"
      name: "default-token-kbjnx"
      readOnly: true
  dnsPolicy: "ClusterFirst"
  enableServiceLinks: true
  hostname: "alertmanager-0"
  nodeName: "for-smi-cdl-1b-worker94d84de255"
  priority: 0
  restartPolicy: "Always"
  schedulerName: "default-scheduler"
  securityContext:
    fsGroup: 0
    runAsUser: 0
  serviceAccount: "default"
  serviceAccountName: "default"
  subdomain: "alertmanager-service"
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: "NoExecute"
    key: "node-role.kubernetes.io/oam"
    operator: "Equal"
    value: "true"
  - effect: "NoExecute"
    key: "node.kubernetes.io/not-ready"
    operator: "Exists"
    tolerationSeconds: 300
  - effect: "NoExecute"
    key: "node.kubernetes.io/unreachable"
    operator: "Exists"
    tolerationSeconds: 300
  volumes:
  - configMap:
      defaultMode: 420
      name: "alertmanager"
    name: "alertmanager-config"
  - emptyDir: {}
    name: "alertmanager-store"
  - name: "default-token-kbjnx"
    secret:
```

```
      defaultMode: 420
      secretName: "default-token-kbjnx"
status:
  conditions:
  - lastTransitionTime: "2020-02-26T06:09:02Z"
    status: "True"
    type: "Initialized"
  - lastTransitionTime: "2020-02-26T06:09:06Z"
    status: "True"
    type: "Ready"
  - lastTransitionTime: "2020-02-26T06:09:06Z"
    status: "True"
    type: "ContainersReady"
  - lastTransitionTime: "2020-02-26T06:09:13Z"
    status: "True"
    type: "PodScheduled"
  containerStatuses:
  - containerID: "docker://821ed1a272d37e3b4c4c9c1ec69b671a3c3fe6eb4b42108edf44709b9c698ccd"

    image: "<path_to_docker_image>"
    imageID:
"docker-pullable:<path_to_docker_image>@sha256:c4bf05aa677a050fba9d86586b04383ca089bd784d2cb9e544b0d6b7ea899d9b"

    lastState: {}
    name: "alertmanager"
    ready: true
    restartCount: 0
    state:
      running:
        startedAt: "2020-02-26T06:09:05Z"
  hostIP: "<host_ipv4address>"
  phase: "Running"
  podIP: "<pod_ipv4address>"
  qosClass: "BestEffort"
  startTime: "2020-02-26T06:09:02Z"
cee#
```

Verifying the Helm Status

This section describes the procedure involved in verifying the helm status. You need to determine whether the deployed helm chart is listed in the helm list successfully.

To determine the helm status:

1. Run the following on the master node to view the list of deployed helm charts.

   helm list

2. If the helm chart is not found, run the following in the operational mode to view the charts irrespective of their deployment status.

   show helm charts

Verifying the Pods

This section describes the procedure involved in determining the pod and container status after upgrading PCF. You need to ensure that the pods and containers are up and running.

Use the following commands to view the PCF pod logs.

kubectl describe pod *pod_name* **-n** *namespace*

Note If the **Status** column displays the state as *Running*, and the **Ready** column has the same number of containers on both sides of the forward-slash (/), then the pod is healthy and operational.

Rollback the Upgrade

You can rollback the upgrade if you encounter any issues during the upgrade process. This section describes the procedure involved rolling back the upgrade.

Reloading PCF Ops Center Configuration

This section describes the procedure involved in reloading the PCF Ops Center configuration from the backup file.

To reload the PCF Ops Center configuration:

1. Log in to the SMI console as an **ubuntu** user.
2. Untar the backup file created on SMI and move it into a directory.

   Example:

   ```
   ubuntu@popcf-cm01:~$ cd ~/backups && tar –zxf popcf-cfg-backup_110219-053530.tar.gz
   ubuntu@popcf-cm01 :~/backups$
   ```

3. Move the backup configuration file into the newly created **backups** directory.

   Example:

   ```
   ubuntu@popcf-cm01 :~/backups$ cd popcf-cfg-backup_110219-053530
   ubuntu@popcf-cm01 :~/backups/popcf-cfg-backup_110219-053530$
   ```

4. Convert the exported PCF Ops Center configuration into a clean file, which is ready for import.

   Example:

   ```
   ubuntu@popcf-cm01 :~/backups/popcf-cfg-backup_110219-053530$ cat pcfops*.cfg | perl -pe
    's/vendor.*\[(.*)\]/vendor $1/g' | perl -pe 's/(^\s+ips).*\[(.*)\]/$1$2/g' | perl -pe
   's/(\w)\s+(\w)/$1 $2/g' | perl -pe 's/^\s+//g' | grep -v "system mode run" > pcfops.txt
   ubuntu@popcf-cm01 :~/backups/popcf-cfg-backup_110219-053530$
   ```

Updating PCF Ops Center Configuration

This section describes the procedure involved in updating the PCF Ops Center configuration after restoring it. To update the PCF Ops Center configuration:

1. Log in to the master node as an **ubuntu** user.
2. Run the following command to log in to the PCF Ops Center CLI.

   Example:

   ```
   ubuntu@popcf-mas01:~$ ssh -p <port_number> admin@$(kubectl get svc -n $(kubectl get
   namespaces | grep -oP 'pcf-(\d+|\w+)') | grep <port_number> | awk '{ print $3 }')
   admin@<admin_ip_address> password: PCF-OPS-PASSWORD
   Welcome to the pcf CLI on popcf01
   admin connected from <admin_ip_address> using ssh on
   ops-center-pcf-01-ops-center-68dd9f588-htjdf
   ```

3. Paste the contents of the exported PCF configuration file (the **pcfops.txt** file mentioned in this example) in the PCF Ops Center.

   Example:

```
product pcf# config
Entering configuration mode terminal
product pcf(config)# <PASTE CONTENTS OF pcfops.txt AND RETURN TO 'config' mode. Don't
Paste Default Configuration>
product pcf(config)#
```

Important Fix any sections in the configuration file that did not import properly.

4. Ensure that the helm URLs are inline with the updated PCF image.

   Example:

```
product pcf(config)# helm repository base-repos
product pcf(config-repository-base-repos)# url <url>
product pcf(config-repository-base-repos)# exit
product pcf(config)# k8s registry <registry_url>
product pcf(config)# commit
Commit complete.
product pcf(config)#
```

Restoring the Configuration from Back Up

This section describes the procedure involved in restoring all the Policy Builder and CRD configuration files from the backup.

Restoring Policy Builder Configuration

1. Log in to the master node as an **ubuntu** user.
2. Retrieve the Cisco Policy Suite Central URL.

   Example:

```
ubuntu@popcf-mas01:~/backups_09182019_T2141$ kubectl get ing -n $( kubectl get namespaces
 | grep -oP 'pcf-(\d+|\w+)' | cut -d\  -f1) | grep policy-builder | awk '{ print $2
}'
pb.pcf-02-pcf-engine-app-blv02.<ipv4address>.nip.io
```

3. Navigate to the Cisco Policy Suite Central URL.
4. Log in with your user credentials.
5. Click **Import/Export**.
6. Click **Import** tab.
7. Click **File to Import**.
8. Select the exported policy backed up in the Back Up SVN, Policy, and CRD Data section.
9. In **Import URL**, specify the following URL:

   `http://svn/repos/configuration`

10. Enter a brief description in **Commit Message** text-box.

11. Click **Import**.
12. Log in to the master node as an **ubuntu** user.
13. Run the following command to retrieve the Cisco Policy Builder URL.

    Example:

```
kubectl get ing -n $(kubectl get namespaces | grep -oP 'pcf-(\d+|\w+)' | cut -d\  -f1)
 | grep policy-builder | awk '{ print "https://"$2"/pb" }'
https://pb.pcf-02-pcf-engine-app-blv02.<ipv4address>.nip.io/pb
ubuntu@popcf-mas01:~/backups_09182019_T2141$
```

14. Navigate to the Cisco Policy Builder URL.
15. Click **Build Policies using version controlled data**.
16. Select **Repository** from the drop-down list.
17. Click **OK**.
18. Log in with your user credentials.
19. Click **File**.
20. Click **Publish to Runtime Environment**.
21. Enter a brief description in **Commit Message**.
22. Click **OK.**

Restoring CRD Data

1. In CPS Central home page, click **Custom Reference Data**.
2. Check the **Export CRD to Golden Repository** check-box.
3. Specify the SVN host name in **Please enter valid server Hostname or IP** text-box.

Note For PCF the SVN host name value is *svn*.

4. Click **+**.
5. Click **Export**.

Note You receive a success message when the data is exported successfully.

Removing Temporary Files

1. Log in to SMI Cluster Manager as an **ubuntu** user.
2. Delete the temporary directory.

Note Ensure that a copy of the image is stored on OSPD before deleting.

Example:

```
ubuntu@popcf-cm01:~$ ls | grep temp
temp_09192019_T0143
ubuntu@popcf-cm01:~/temp_08072019_T1651$
ubuntu@popcf-cm01:~/temp_08072019_T1651$ rm -f temp_09192019_T0143
ubuntu@popcf-cm01:~/temp_08072019_T1651$
```