Rolling Software Update

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Product(s) or Functional Area	cnBNG
Applicable Platform(s)	SMI
Feature Default Setting	Disabled - Configuration Required
Related Documentation	• Multiple Replica Support for cnBNG Services • High Availability and CP Reconciliation

Revision History

Table 2: Revision History

Revision Details	Release
Rolling software upgrade support for LAC and LNS sessions.	2022.03.0
First introduced.	2021.04.0

Feature Description

The cnBNG Rolling Software Update feature enables incremental update of pod instances with minimal downtime. In Kubernetes (K8s), this implementation is possible only with rolling updates.

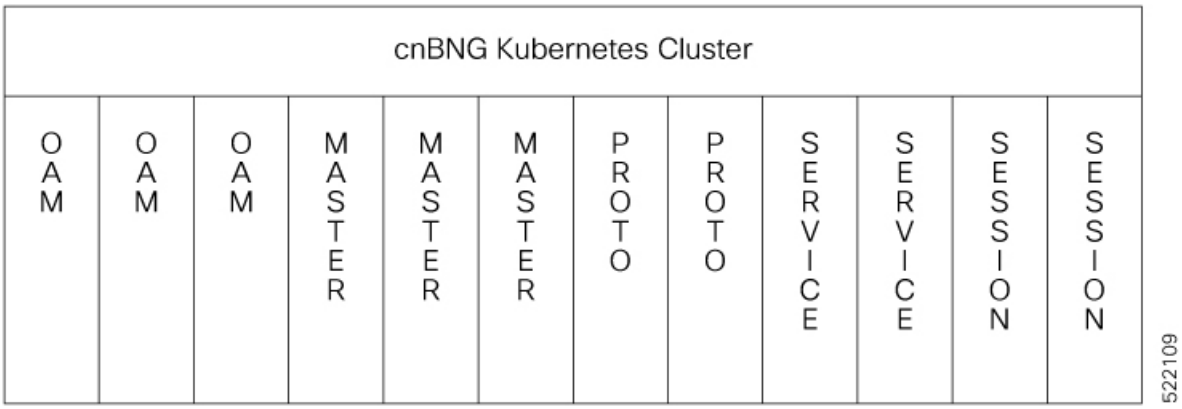
Subscriber Microservices Infrastructure (SMI) platform supports rolling software upgrade for cnBNG pods. The "Pod Restart and Reconciliation" and "Multiple Replica Support for cnBNG Services" features depend on this feature. For more information, see Multiple Replica Support for cnBNG Services and High Availability and CP Reconciliation.

The cnBNG has a three-tier architecture consisting 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 K8s master and worker nodes (including Operation and Management nodes).

For high availability (HA) and fault tolerance, cnBNG requires a minimum of two K8s worker nodes for each tier. Each worker node can have multiple replicas. K8s orchestrates the pods using the StatefulSets controller. The pods require a minimum of two replicas for fault tolerance.

The following figure depicts the cnBNG K8s Cluster with 12 nodes – three Master nodes, three Operations and Management (OAM) worker nodes, two Protocol worker nodes, two Service worker nodes, and two Session (data store) worker nodes.

Figure 1: cnBNG Kubernetes Cluster



Note

- OAM worker nodes—Host the Ops Center pods for configuration management and metrics pods for statistics and Key Performance Indicators (KPIs).
- Protocol worker nodes—Host the cnBNG protocol-related pods for UDP-based interfaces such as N4, RADIUS, and GTP.
- Service worker nodes—Host the cnBNG application-related pods that perform session and FSOL management.
- Session worker nodes—Host the database-related pods that store subscriber session data.

Rolling Upgrade Support for LAC and LNS

cnBNG supports the rolling upgrade procedure for LAC and LNS sessions in addition to PTA and IPoE sessions. Rolling upgrade for LAC and LNS clusters is supported without impacting the existing tunnels by modifying the L2TP service with an initial readiness delay of 60 seconds. The K8s infrastructure upgrades an instance of L2TP pod only after the active instance is successfully upgraded and resynchronizes all existing tunnel data in the peer L2TP instance.

If more than one L2TP instance exists, rolling upgrade updates only one instance at a time. The stateful set pairing must be done accordingly to avoid peer L2TP instance going down at the same time.

How it Works

This section describes how the cnBNG Rolling Software Update works.

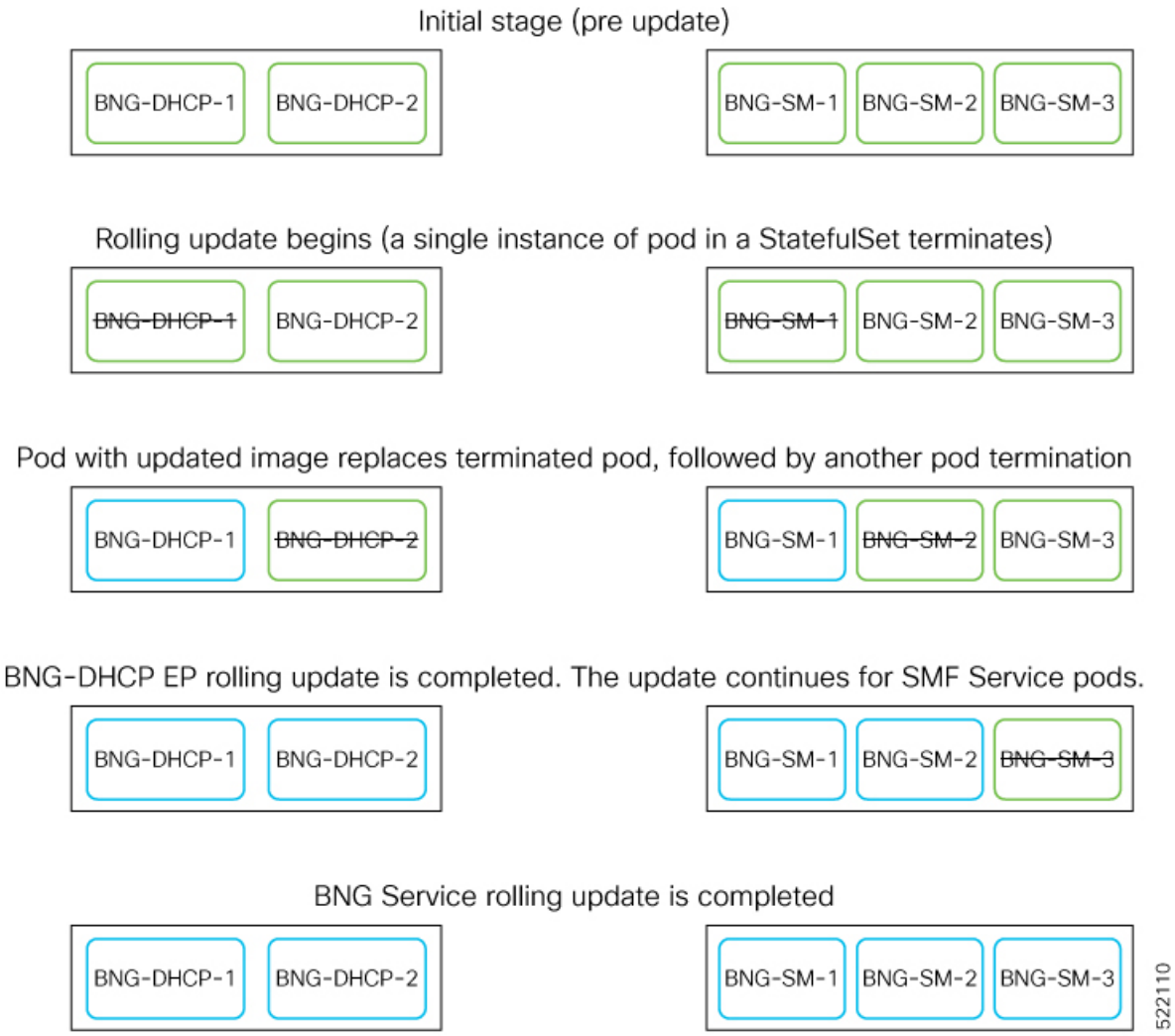
Rolling Software Update Using SMI Cluster Manager

Rolling software upgrade is a process of upgrading or migrating the build from an older to a newer version or upgrading the patch for the prescribed deployment set of application pods.

The cnBNG 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 updated sequentially to ensure that the ongoing process continues. 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 completing all the ongoing processes. Other in-service pods continue to receive and process the traffic with minimal impact. Use the Ops Center CLI. to control the software update process.

The following figure illustrates an cnBNG rolling update for bng-dhcp and bng-sm endpoint pods (two replicas) on Protocol worker nodes along with cnBNG Service pods (three replicas) on Service worker nodes.

Figure 2: cnBNG Rolling Update



Installing the Rolling Software Update

This section describes how to install the cnBNG Rolling Software Update feature.

The Rolling Software Update feature involves the following procedures.

Prerequisites

The prerequisites for installing the rolling software update for cnBNG are as follows:

- Ensure that all the nodes, including all the pods in the node, are up and running.
- Perform the cnBNG health check.
- Prepare for the upgrade.

- Backup the Ops Center configuration.
- Backup the CEE and BNG Ops Center configuration.
- Stage a new cnBNG image.

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

Performing the cnBNG Health Check

Perform the cnBNG health check to ensure that all the services are running and nodes are in ready state. To perform an health check, log in to the 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 bng-bng -o wide
kubectl get pods -n cee-global -o wide
kubectl get pods -n smi-vips -o wide
helm list -A
kubectl get pods -A | wc -l
```

Important Ensure that all the services are running and nodes are in ready state before proceeding further.

Note

- Static calls would not be impacted due to rolling upgrade.
- Inflight transactions and events will see failure during the rolling upgrade.
- For about 1-2 minutes downtime (that is, 100% transaction failures) is expected during upgrade with the suggested replica counts for 500K scale & 1000 CPS. Note: We can achieve zero downtime by increasing the number of replicas for each of the pods [dhcp, sm, pppoe], but that comes with the cost of additional resources.

Backing Up Ops Center Configuration

This section describes the procedure involved in creating a backup of the Ops Center configurations. To backup 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')
   ```

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')
   ```

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

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

Backing Up CEE and BNG Ops Center Configuration

This section describes the procedure involved in creating a backup of CEE and BNG Ops Center configuration from the master node. To perform a backup of CEE and BNG 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. Backup the BNG 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 'bng-(\d+|\w+)') | grep <port_number> | awk '{ print $3
   }') "show run | nomore" > bngops.backup_$(date +'%m%d%Y_T%H%M') && wc
    -l bngops.backup_$(date +'%m%d%Y_T%H%M')
   ```

   Example:

   ```
   ubuntu@pobng-mas01:~/backups_09182019_T2141$ ssh -p 2024 admin@$(kubectl get svc -n
   $(kubectl get namespaces | grep -oP 'bng-(\d+|\w+)') | grep <port_number> | awk '{ print
    $3 }') "show run | nomore" > bngops.backup_$(date +'%m%d%Y_T%H%M') && wc -l
   bngops.backup_$(date +'%m%d%Y_T%H%M')
   admin@<ipv4address>'s password: bng-OPS-PASSWORD
   334 bngops.backup
   ```

4. Backup 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@pobng-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@<ipv4address>'s 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@pobng-mas01:~/backups_09182019_T2141$ scp $(grep cm01 /etc/hosts | awk '{ print
$1 }'):/home/ubuntu/smiops.backup_$(date +'%m%d%Y_T%H%M') .
ubuntu@<ipv4address>'s password: SMI-CM-PASSWORD
smiops.backup                                                 100% 9346    22.3MB/s
   00:00
```

6. Verify the line count of the backup files.

   Example:

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

Staging a New cnBNG Image

The SMI Deployer downloads the new image and verifies it. Provide sha256 details in the "software cnf" section:

```
software cnf bng.2021.04.0.i96
url
https://engci-maven-master.cisco.com/artifactory/mobile-cnat-charts-release/relengbuilds/2021.04.0/bng/2021.04.0.i96/bng.2021.04.0.i96-offline/bng.2021.04.0.i96.SSA.tgz
user testuser-deployer.gen
password $8$L1KSfQG9oMTkulzRxFjPTRsOH1O7S9qUVsLgDcFqJO4=
accept-self-signed-certificate true
sha256 d3a440be0e6080f2a83dc3d4e20121f2ceddadd0368a1d1bf41e567a397d35e0
exit
```

Performing Rolling Software Update

The cnBNG uses the SMI Cluster Manager to perform a rolling software update. To update cnBNG using the SMI Cluster Manager:

Important Ensure that cnBNG is up and running with the current version of the software.

1. Log in to SMI Cluster Manager Ops Center.
2. Update the product repository URL with the latest version of the product chart.

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

```
config
  cluster cluster_name
   ops-centers app_name bng_instance_name
       repository-local local_repository
        exit
      exit
```

Example:

```
SMI Cluster Manager# config
SMI Cluster Manager(config)# clusters test2
```

```
SMI Cluster Manager(config-clusters-test2)# ops-centers bng bng
SMI Cluster Manager(config-ops-centers-bng/bng)# repository-local <reference to the
locally downloaded image>
SMI Cluster Manager(config-ops-centers-bng/bng)# exit
SMI Cluster Manager(config-clusters-test2)# exit
```

3. Run the **cluster sync** command to update to the latest version of the product chart.

   clusters *cluster_name* **actions sync run**

   Example:

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

Important

- The cluster synchronization updates the BNG Ops Center, which in turn updates the application pods (through **helm sync** command) in sequence, automatically.
- When the rolling upgrade is in progress on a specific pod, the cnBNG avoids routing new calls to that pod.
- The cnBNG waits for 30 seconds before restarting the pod where rolling upgrade is initiated. Also, the cnBNG establishes all the in-progress calls completely within 30 seconds during the upgrade period (maximum call-setup time is 10 seconds).

Note

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

Monitoring the Rolling Software Update

Use the following sample configuration to monitor the status of the Rolling Software Update using the SMI Cluster Manager Ops Center:

```
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
  exit
```

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.

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
```

Important To view the pod details after the upgrade through CEE Ops Center, see Viewing the Pod Details, on page 9.

Viewing the Pod Details

Use the following sample configuration to view the details of the current pods through CEE Ops Center (in CEE Ops Center CLI):

`cluster pods` *`instance_name pod_name`* **`detail`**

NOTES:

- **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 udp-proxy-0 in the bng-bng instance.

Example:

```
svi-cn-bng-tb4/global] cee# cluster pods bng-bng udp-proxy-0 detail
details apiVersion: "v1"
kind: "Pod"
metadata:
  annotations:
    prometheus.io/port: "8083"
    prometheus.io/scrape: "true"
    sidecar.istio.io/inject: "false"
  creationTimestamp: "2021-10-25T00:19:28Z"
  generateName: "udp-proxy-"
  labels:
    component: "udp-proxy"
    controller-revision-hash: "udp-proxy-5444cc5d74"
    instanceId: "1"
    release: "bng-bng-udp-proxy"
    statefulset.kubernetes.io/pod-name: "udp-proxy-0"
```

```
managedFields:
- apiVersion: "v1"
  fieldsType: "FieldsV1"
  fieldsV1:
    f:metadata:
      f:annotations:
        .: {}
        f:prometheus.io/port: {}
        f:prometheus.io/scrape: {}
        f:sidecar.istio.io/inject: {}
      f:generateName: {}
      f:labels:
        .: {}
        f:component: {}
        f:controller-revision-hash: {}
        f:instanceId: {}
        f:release: {}
        f:statefulset.kubernetes.io/pod-name: {}
      f:ownerReferences:
        .: {}
        k:{"uid":"914265b3-8b5b-4301-9433-c748e791c332"}:
          .: {}
          f:apiVersion: {}
          f:blockOwnerDeletion: {}
          f:controller: {}
          f:kind: {}
          f:name: {}
          f:uid: {}
    f:spec:
      f:affinity:
        .: {}
        f:nodeAffinity:
          .: {}
          f:requiredDuringSchedulingIgnoredDuringExecution:
            .: {}
            f:nodeSelectorTerms: {}
        f:podAntiAffinity:
          .: {}
          f:requiredDuringSchedulingIgnoredDuringExecution: {}
      f:containers:
        k:{"name":"udp-proxy"}:
          .: {}
          f:command: {}
          f:env:
            .: {}
            k:{"name":"APPLICATION_NAME"}:
              .: {}
              f:name: {}
              f:value: {}
            k:{"name":"CLUSTER_NAME"}:
              .: {}
              f:name: {}
              f:value: {}
            k:{"name":"COVERAGE_BUILD"}:
              .: {}
              f:name: {}
              f:value: {}
            k:{"name":"CPS_PATCH"}:
              .: {}
              f:name: {}
            k:{"name":"DATACENTER_NAME"}:
              .: {}
              f:name: {}
              f:value: {}
```

```
k:{"name":"ENABLE_ADD_DYNAMIC_BGP_ROUTE"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"ENABLE_RETRY_CONFIG"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"ENABLE_SGW_CACHE"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"ENABLE_TP_FEATURE"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"GOGC"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"GOMAXPROCS"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"GOTRACEBACK"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"GR_INSTANCE_ID"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"INFRA_ADMIN_PORT"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"INFRA_DIAG_PORT"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"INFRA_PROMETHEUS_PORT"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"INSTANCE_NODE_ID"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"IPC_EP_PORT"}:
  .: {}
  f:name: {}
  f:value: {}
k:{"name":"MY_POD_IP"}:
  .: {}
  f:name: {}
  f:valueFrom:
    .: {}
    f:fieldRef:
      .: {}
      f:apiVersion: {}
      f:fieldPath: {}
k:{"name":"MY_POD_NAME"}:
  .: {}
  f:name: {}
```

```
      f:valueFrom:
        .: {}
        f:fieldRef:
          .: {}
          f:apiVersion: {}
          f:fieldPath: {}
    k:{"name":"PPROF_EP_PORT"}:
      .: {}
      f:name: {}
      f:value: {}
    k:{"name":"PROTOCOL_POD"}:
      .: {}
      f:name: {}
      f:value: {}
    k:{"name":"PROXY_KEEPALIVED_PORT"}:
      .: {}
      f:name: {}
      f:value: {}
    k:{"name":"SERVICE_NAME"}:
      .: {}
      f:name: {}
      f:value: {}
    k:{"name":"SMF_PROFILE_CONFIGURED"}:
      .: {}
      f:name: {}
      f:value: {}
  f:image: {}
  f:imagePullPolicy: {}
  f:name: {}
  f:readinessProbe:
    .: {}
    f:failureThreshold: {}
    f:initialDelaySeconds: {}
    f:periodSeconds: {}
    f:successThreshold: {}
    f:tcpSocket:
      .: {}
      f:host: {}
      f:port: {}
    f:timeoutSeconds: {}
  f:resources:
    .: {}
    f:limits:
      .: {}
      f:cpu: {}
      f:memory: {}
    f:requests:
      .: {}
      f:cpu: {}
      f:memory: {}
  f:terminationMessagePath: {}
  f:terminationMessagePolicy: {}
  f:volumeMounts:
    .: {}
    k:{"mountPath":"/config/udp-proxy/coverage"}:
      .: {}
      f:mountPath: {}
      f:name: {}
      f:readOnly: {}
    k:{"mountPath":"/config/udp-proxy/flowcontrol"}:
      .: {}
      f:mountPath: {}
      f:name: {}
      f:readOnly: {}
```

```
      k:{"mountPath":"/config/udp-proxy/logging"}:
        .: {}
        f:mountPath: {}
        f:name: {}
        f:readOnly: {}
      k:{"mountPath":"/config/udp-proxy/system"}:
        .: {}
        f:mountPath: {}
        f:name: {}
        f:readOnly: {}
      k:{"mountPath":"/config/udp-proxy/vip-ip"}:
        .: {}
        f:mountPath: {}
        f:name: {}
        f:readOnly: {}
f:dnsPolicy: {}
f:enableServiceLinks: {}
f:hostNetwork: {}
f:hostname: {}
f:imagePullSecrets:
  .: {}
  k:{"name":"regcredbng"}:
    .: {}
    f:name: {}
f:restartPolicy: {}
f:schedulerName: {}
f:securityContext: {}
f:subdomain: {}
f:terminationGracePeriodSeconds: {}
f:volumes:
  .: {}
  k:{"name":"coverage-volume"}:
    .: {}
    f:configMap:
      .: {}
      f:defaultMode: {}
      f:items: {}
      f:name: {}
    f:name: {}
  k:{"name":"flowcontrol-volume"}:
    .: {}
    f:configMap:
      .: {}
      f:defaultMode: {}
      f:items: {}
      f:name: {}
      f:optional: {}
    f:name: {}
  k:{"name":"logging-volume"}:
    .: {}
    f:configMap:
      .: {}
      f:defaultMode: {}
      f:items: {}
      f:name: {}
    f:name: {}
  k:{"name":"system-volume"}:
    .: {}
    f:configMap:
      .: {}
      f:defaultMode: {}
      f:items: {}
      f:name: {}
    f:name: {}
```

```
            k:{"name":"vip-ip-volume"}:
              .: {}
              f:configMap:
                .: {}
                f:defaultMode: {}
                f:items: {}
                f:name: {}
              f:name: {}
      manager: "kube-controller-manager"
      operation: "Update"
      time: "2021-10-25T00:19:28Z"
    - apiVersion: "v1"
      fieldsType: "FieldsV1"
      fieldsV1:
        f:status:
          f:conditions:
            k:{"type":"ContainersReady"}:
              .: {}
              f:lastProbeTime: {}
              f:lastTransitionTime: {}
              f:status: {}
              f:type: {}
            k:{"type":"Initialized"}:
              .: {}
              f:lastProbeTime: {}
              f:lastTransitionTime: {}
              f:status: {}
              f:type: {}
            k:{"type":"Ready"}:
              .: {}
              f:lastProbeTime: {}
              f:lastTransitionTime: {}
              f:status: {}
              f:type: {}
          f:containerStatuses: {}
          f:hostIP: {}
          f:phase: {}
          f:podIP: {}
          f:podIPs:
            .: {}
            k:{"ip":"208.208.208.21"}:
              .: {}
              f:ip: {}
          f:startTime: {}
      manager: "kubelet"
      operation: "Update"
      time: "2021-10-25T00:19:38Z"
    name: "udp-proxy-0"
    namespace: "bng-bng"
    ownerReferences:
    - apiVersion: "apps/v1"
      kind: "StatefulSet"
      blockOwnerDeletion: true
      controller: true
      name: "udp-proxy"
      uid: "914265b3-8b5b-4301-9433-c748e791c332"
    resourceVersion: "1557892"
    uid: "d519c85b-baae-4131-925b-df46e72757ac"
  spec:
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
          - matchExpressions:
```

```
          - key: "smi.cisco.com/vm-type"
            operator: "In"
            values:
            - "protocol"
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: "component"
            operator: "In"
            values:
            - "udp-proxy"
        topologyKey: "kubernetes.io/hostname"
  containers:
  - command:
    - "/usr/local/bin/run-app"
    env:
    - name: "GOGC"
      value: "200"
    - name: "GOTRACEBACK"
      value: "crash"
    - name: "GOMAXPROCS"
      value: "12"
    - name: "CPS_PATCH"
    - name: "SERVICE_NAME"
      value: "udp-proxy"
    - name: "INFRA_PROMETHEUS_PORT"
      value: "8083"
    - name: "INFRA_ADMIN_PORT"
      value: "8879"
    - name: "INFRA_DIAG_PORT"
      value: "8979"
    - name: "PPROF_EP_PORT"
      value: "8850"
    - name: "IPC_EP_PORT"
      value: "9004"
    - name: "PROXY_KEEPALIVED_PORT"
      value: "28000"
    - name: "ENABLE_RETRY_CONFIG"
      value: "true"
    - name: "COVERAGE_BUILD"
      value: "false"
    - name: "DATACENTER_NAME"
      value: "DC"
    - name: "CLUSTER_NAME"
      value: "Local"
    - name: "APPLICATION_NAME"
      value: "BNG"
    - name: "INSTANCE_NODE_ID"
      value: "0"
    - name: "GR_INSTANCE_ID"
      value: "1"
    - name: "SMF_PROFILE_CONFIGURED"
      value: "false"
    - name: "ENABLE_TP_FEATURE"
      value: "true"
    - name: "ENABLE_ADD_DYNAMIC_BGP_ROUTE"
      value: "true"
    - name: "ENABLE_SGW_CACHE"
      value: "true"
    - name: "PROTOCOL_POD"
      value: "bng-n4-protocol"
    - name: "MY_POD_IP"
      valueFrom:
```

```
        fieldRef:
          apiVersion: "v1"
          fieldPath: "status.podIP"
    - name: "MY_POD_NAME"
      valueFrom:
        fieldRef:
          apiVersion: "v1"
          fieldPath: "metadata.name"
    image:
"docker.10.81.103.113.nip.io/bng.2021.04.0.i105/mobile-cnat-cn/udp-proxy/rel-2021.04/udp_proxy:0.1.1-16b9200-fe3d3ad-27f9489"

    imagePullPolicy: "IfNotPresent"
    name: "udp-proxy"
    readinessProbe:
      failureThreshold: 3
      initialDelaySeconds: 6
      periodSeconds: 5
      successThreshold: 1
      tcpSocket:
        host: "127.0.0.1"
        port: 28000
      timeoutSeconds: 1
    resources:
      limits:
        cpu: "3"
        memory: "32Gi"
      requests:
        cpu: "3"
        memory: "8Gi"
    terminationMessagePath: "/dev/termination-log"
    terminationMessagePolicy: "File"
    volumeMounts:
    - mountPath: "/config/udp-proxy/logging"
      name: "logging-volume"
      readOnly: true
    - mountPath: "/config/udp-proxy/vip-ip"
      name: "vip-ip-volume"
      readOnly: true
    - mountPath: "/config/udp-proxy/system"
      name: "system-volume"
      readOnly: true
    - mountPath: "/config/udp-proxy/flowcontrol"
      name: "flowcontrol-volume"
      readOnly: true
    - mountPath: "/config/udp-proxy/coverage"
      name: "coverage-volume"
      readOnly: true
    - mountPath: "/var/run/secrets/kubernetes.io/serviceaccount"
      name: "kube-api-access-hn2p5"
      readOnly: true
  dnsPolicy: "ClusterFirstWithHostNet"
  enableServiceLinks: true
  hostNetwork: true
  hostname: "udp-proxy-0"
  imagePullSecrets:
  - name: "regcredbng"
  nodeName: "svi-cn-bng-tb4-proto2"
  preemptionPolicy: "PreemptLowerPriority"
  priority: 100000000
  priorityClassName: "default-application"
  restartPolicy: "Always"
  schedulerName: "default-scheduler"
  securityContext: {}
  serviceAccount: "default"
```

```
serviceAccountName: "default"
subdomain: "udp-proxy-i1"
terminationGracePeriodSeconds: 30
tolerations:
- effect: "NoExecute"
  key: "node.kubernetes.io/not-ready"
  operator: "Exists"
  tolerationSeconds: 30
- effect: "NoExecute"
  key: "node.kubernetes.io/unreachable"
  operator: "Exists"
  tolerationSeconds: 30
volumes:
- configMap:
    defaultMode: 420
    items:
    - key: "logging"
      path: "logging.yaml"
    name: "infra-logging-conf"
  name: "logging-volume"
- configMap:
    defaultMode: 420
    items:
    - key: "endpointIp"
      path: "endpointIp.yaml"
    name: "udp-proxy-vip-ip-conf"
  name: "vip-ip-volume"
- configMap:
    defaultMode: 420
    items:
    - key: "system"
      path: "system.yaml"
    name: "infra-system-conf"
  name: "system-volume"
- configMap:
    defaultMode: 420
    items:
    - key: "flowcontrol"
      path: "flowcontrol.yaml"
    name: "udp-proxy-flowcontrol-conf"
    optional: true
  name: "flowcontrol-volume"
- configMap:
    defaultMode: 420
    items:
    - key: "coverage"
      path: "coverage.yaml"
    name: "udp-proxy-coverage-conf"
  name: "coverage-volume"
- name: "kube-api-access-hn2p5"
  projected:
    defaultMode: 420
    sources:
    - serviceAccountToken:
        expirationSeconds: 3607
        path: "token"
    - configMap:
        items:
        - key: "ca.crt"
          path: "ca.crt"
        name: "kube-root-ca.crt"
    - downwardAPI:
        items:
        - fieldRef:
```

```
              apiVersion: "v1"
              fieldPath: "metadata.namespace"
            path: "namespace"
status:
  conditions:
  - lastTransitionTime: "2021-10-25T00:19:28Z"
    status: "True"
    type: "Initialized"
  - lastTransitionTime: "2021-10-25T00:19:38Z"
    status: "True"
    type: "Ready"
  - lastTransitionTime: "2021-10-25T00:19:38Z"
    status: "True"
    type: "ContainersReady"
  - lastTransitionTime: "2021-10-25T00:19:28Z"
    status: "True"
    type: "PodScheduled"
  containerStatuses:
  - containerID: "docker://9365e5d78de9e7edf427ee92f3aa7e74c4fdf5070874c89045079a1586199358"

    image:
"docker.10.81.103.113.nip.io/bng.2021.04.0.i105/mobile-cnat-cn/udp-proxy/rel-2021.04/udp_proxy:0.1.1-16b9200-fe3d3ad-27f9489"

    imageID:
"docker-pullable://docker.10.81.103.113.nip.io/bng.2021.04.0.i105/mobile-cnat-cn/udp-proxy/rel-2021.04/udp_proxy@sha256:caeb2c6b510b07029590c25a8a8a1a98168c68bcb6bc647e2c38f1de6e7af90c"

    lastState: {}
    name: "udp-proxy"
    ready: true
    restartCount: 0
    started: true
    state:
      running:
        startedAt: "2021-10-25T00:19:29Z"
  hostIP: "208.208.208.21"
  phase: "Running"
  podIP: "208.208.208.21"
  podIPs:
  - ip: "208.208.208.21"
  qosClass: "Burstable"
  startTime: "2021-10-25T00:19:28Z"

[svi-cn-bng-tb4/global] cee#
```