Pods and Services Reference

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Product(s) or Functional Area	AMF
Applicable Platform(s)	SMI
Feature Default Setting	Enabled - Always-on For Multiple Service Pods on Multiple Nodes: Disabled – Configuration required to enable
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
Multiple Service Pods on Multiple Nodes	2023.02.0
First introduced.	2021.04.0

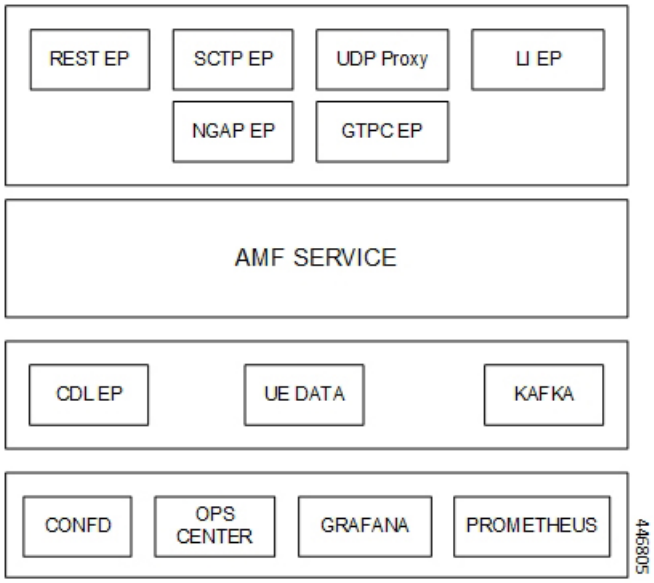
Feature Description

The AMF 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, AMF uses the construct that includes the components such as pods and services.

Depending on your deployment environment, the AMF 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 AMF 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



Pods

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

Kubernetes deploys one or multiple pods on a single or multiple nodes 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 tables list the AMF pod names and the Kubernetes node names 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 7.

Note Maximum number of pods that can be configured per node is 256.

Note In case of separate CDL deployment, CDL pods are visible under CDL namespace.

Table 3: AMF Pods

Pod Name	Description	Kubernetes Node Name
base-entitlement-amf	Supports Smart Licensing feature.	OAM
cache-pod	Operates as the pod to cache any sort of system information that will be used by other pods as applicable.	Protocol
cdl-ep-session-c1	Provides an interface to the CDL.	Session
cdl-index-session-c1	Preserves the mapping of keys to the session pods.	Session
cdl-slot-session-c1	Operates as the CDL Session pod to store the session data.	Session
etcd-amf-etcd-cluster	Hosts the etcd for the AMF application to store information, such as pod instances, leader information, NF-UUID, endpoints, and so on.	OAM
georeplication	Contains business logic for Geographic Redundancy (Currently, GR is not fully supported in AMF).	Protocol
grafana-dashboard-cdl	Contains the default dashboard of CDL metrics in Grafana.	OAM
grafana-dashboard-amf	Contains the default dashboard of AMF-service metrics in Grafana.	OAM
gtpc-ep	Operates as GTPC endpoint of AMF.	Protocol
kafka	Hosts the Kafka details for the CDL replication.	Protocol
nodemgr	Performs node level interactions, such as N4 link establishment, management (heart-beat). It also generates unique identifiers, such as NGAP-ID, TMSI, GUTI and so on.	Service

Pod Name	Description	Kubernetes Node Name
oam-pod	Operates as the pod to facilitate Ops Center actions, such as show commands, configuration commands, monitor protocol monitor subscriber, and so on.	OAM
ops-center-amf-ops-center	Acts as the AMF Ops Center.	OAM
smart-agent-amf-ops-center	Operates as the utility pod for the AMF Ops Center.	OAM
amf-service	Contains main business logic of AMF.	Service
amf-rest-ep	Operates as REST endpoint of AMF for HTTP2 communication.	Protocol
amf-protocol-ep	Processes NGAP/NAS Protocol Messages.	Protocol
amf-gosctp-lb	Operates as SCTP endpoint for AMF.	Protocol
udp-proxy	Operates as proxy for all UDP messages. Owns UDP client and server functionalities.	Protocol
swift-amf-ops-center	Operates as the utility pod for the AMF Ops Center.	OAM
zookeeper	Assists Kafka for topology management.	OAM
li-ep	Responsible for handling LI-IRI events for AMF.	OAM

Services

The AMF configuration is composed of several microservices that run on a set of discrete pods. Microservices are deployed during the AMF deployment. AMF 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 AMF services and the pod on which they run.

Note In case of separate CDL deployment, CDL related services are visible under CDL namespace.

Table 4: AMF Services and Pods

Service Name	Pod Name	Description
alert-frwd-ops-center	ops-center-amf-ops-center	Responsible for forwarding SNMP alerts.

Service Name	Pod Name	Description
amf-gosctp-lb	amf-gosctp-lb	Responsible for receiving incoming traffic over SCTP from N1 interface.
amf-nrf-service	amf-rest-ep	Responsible for providing API for NRF CLIs.
amf-protocol-ep	amf-protocol-ep	Responsible for inter-pod communication with amf-protocol-ep pod.
amf-rest-ep	amf-rest-ep	Responsible for inter-pod communication with amf-rest-ep pod.
amf-sbi-service	amf-rest-ep	Responsible for routing incoming SBI messages to REST-EP pods.
amf-service	amf-service	Responsible for inter-pod communication with amf-service pod.
base-entitlement-amf	ops-center-amf-ops-center	Supports Smart Licensing feature.
bgpspeaker-pod	georeplication-pod-0	Responsible for providing Geo replication support.
datastore-ep-session	cdl-ep-session	Responsible for the CDL session.
datastore-notification-ep	amf-rest-ep	Responsible for sending the notifications from the CDL to the smf-service through amf-rest-ep.
datastore-tls-ep-session	cdl-ep-session	Responsible for the secure CDL connection.
documentation	documentation	Responsible for the AMF documents.
etcd	etcd-cluster	Responsible for pod discovery within the namespace.
etcd-amf-ins1-etcd-cluster-0	etcd-cluster	Responsible for synchronization of data among the ETCD cluster.
etcd-amf-ins1-etcd-cluster-1	etcd-cluster	Responsible for synchronization of data among the ETCD cluster.
etcd-amf-ins1-etcd-cluster-2	etcd-cluster	Responsible for synchronization of data among the ETCD cluster.
grafana-dashboard-amf	grafana-dashboard-amf	Responsible for the default dashboard of AMF-service metrics in Grafana.
grafana-dashboard-cdl-cdl-amf	grafana-dashboard-cdl	Responsible for the default dashboard of CDL metrics in Grafana.

Service Name	Pod Name	Description
grafana-dashboard-etcd-amf	grafana-dashboard-etcd	Responsible for the default dashboard of ETCD metrics in Grafana.
gtpc-ep	gtpc-ep	Responsible for inter-pod communication with GTP-C pod.
kafka	kafka	Processes the Kafka messages.
local-ldap-proxy-amf-ins1-ops-center	ops-center-amf-ops-center	Responsible for leveraging Ops Center credentials by other applications, such as Grafana.
netconf-ops-center-amf-ins1-ops-center	ops-center-amf-ops-center	Responsible for providing/exposing netconf interface to configure AMF.
nodemgr	nodemgr	Responsible for inter-pod communication with nodemgr pod.
oam-pod	oam-pod	Responsible to facilitate Exec commands on the Ops Center.
ops-center-amf-ops-center	ops-center-amf-ops-center	Operates as the utility pod for the SMF Ops Center.
prometheus-rules-etcd	prometheus-rules-etcd	Responsible for the default Prometheus rules of ETCD in Prometheus.
smart-agent-amf-ops-center	smart-agent-amf-ops-center	Responsible for smart licensing.
ssh-ops-center-amf-ops-center	ops-center-amf-ops-center	To access AMF Ops Center using SSH IP.
zookeeper	zookeeper	Assists Kafka for topology management.
zookeeper-service	zookeeper	Assists Kafka for topology management.

Open Ports and Services

The AMF uses different ports for communication. The following table describes the default open ports and the associated services.

Table 5: Open Ports and Services

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

Port	Service	Usage
6443	HTTP	SMI uses port to communicate with the Kubernetes API server.
9100	jetdirect	SMI uses TCP port to communicate with the Node Exporter. Node Exporter is a Prometheus exporter for hardware and OS metrics with pluggable metric collectors. It allows you to measure various machine resources, such as memory, disk, and CPU utilization.
10250	SSL/HTTP	SMI uses TCP port to communicate with Kubelet. Kubelet is the lowest level component in Kubernetes. It is responsible for what is running on an individual machine. It is a process watcher or supervisor focused on active container. It ensures the specified containers are up and running.
10256	HTTP	SMI uses TCP port to interact with the Kube proxy. Kube proxy is a network proxy that runs on each node in your cluster. Kube proxy maintains network rules on nodes. These network rules allow network communication to your pods from network sessions inside or outside of your cluster.
2024	SSH	AMF Ops Center uses this port to provide the ConfD CLI access.
9090	HTTP	AMF REST endpoint pods use this port to expose the APIs to support NRF interface specific CLIs.
8090	HTTP	AMF REST endpoint pods use this port for routing incoming SBI messages to REST-EP pods.
8890	gRPC/HTTP	AMF REST endpoint pods use this port to receive timer notification from CDL.
3179	Tcpwrapped	SMI uses this TCP port for Calico(Kubernetes networking). Calico is used for routing the networking packets to pods.

In addition to the preceding ports, AMF uses the ports that are destined for SMI for routing information between hosts. For more information on SMI ports, see *Ultra Cloud Core Subscriber Microservices Infrastructure Operations Guide*.

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 are deployed on the nodes that match the key-value pair.

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

```
config
  k8 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
    sctp-layer
      key key_value
      value value
      end
```

NOTES:

- **label { cdl-layer { key** *key_value* **| value** *value* **}**—Specify the key value pair for CDL.
- **oam-layer { key** *key_value* **| value** *value* **}**—Specify the key value pair for OAM layer.
- **protocol-layer { key** *key_value* **| value** *value* **}**—Specify the key value pair for protocol layer.
- **service-layer { key** *key_value* **| value** *value* **}**—Specify the key value pair for the service layer.
- **sctp-layer { key** *key_value* **| value** *value* **}**—Specify the protocol value. Example: For k8 label sctp-layer key value is smi.cisco.com/node-type-2 value protocol.

Note If you opt not to configure the labels, then AMF assumes the labels with the default key-value pair.

Viewing the Status and Pod Details

If the service requires extra pods, the AMF creates, and deploys those pods.

You can perform the following:

- View the list of pods that are participating in your deployment through the AMF Ops Center.
- Run the **kubectl** command from the Master node to manage the Kubernetes resources.

To view the comprehensive pod details, use the following command:

- **kubectl get pods -n** *amf_namespace 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 the pod is in the current state.

Sample Output:

```
kubectl get pod -n amf-ins cache-pod-0 -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    cni.projectcalico.org/podIP: 209.165.201.3/32
    cni.projectcalico.org/podIPs: 209.165.201.3/32,4141:4141::d32/128
    prometheus.io/port: "10080"
    prometheus.io/scrape: "true"
    sidecar.istio.io/inject: "false"
  creationTimestamp: "2021-10-16T18:03:32Z"
  generateName: cache-pod-
  labels:
    component: cache-pod
    controller-revision-hash: cache-pod-56dc45d7df
    release: amf-ins1-infra-charts
    statefulset.kubernetes.io/pod-name: cache-pod-0
  name: cache-pod-0
  namespace: amf-ins1
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: StatefulSet
    name: cache-pod
    uid: 18dfdb38-ca20-47ab-b525-770be9ace57c
  resourceVersion: "5770907"
  uid: 088c4f8d-143b-4096-ad03-f95409c16db9
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: smi.cisco.com/node-type-2
            operator: In
            values:
            - protocol
.
.
.
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2021-10-16T18:03:47Z"
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2021-10-16T18:04:52Z"
    status: "True"
```

```
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2021-10-16T18:04:52Z"
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2021-10-16T18:03:32Z"
    status: "True"
    type: PodScheduled
  containerStatuses:
  - containerID: docker://68f5c45ed73ee311a05a32be4fadca0cb9fda0742a01d303fe5115dfa7573a48

    image:
docker.209.165.201.29.nip.io/amf.2021.04.m0.i80/mobile-cnat-app-infra/cache-pod/main/
cache_pod:0.1.0-32e359a
    imageID:
docker-pullable://docker.209.165.201.29.nip.io/amf.2021.04.m0.i80/mobile-cnat-app-infra/
cache-pod/main/cache_pod@sha256:d2c82e1af506cf92c04d93f40ef8ca1dfcf830d457bfeabd4dc8aba7b63ce894

    lastState: {}
    name: cache-pod
    ready: true
    restartCount: 0
    started: true
    state:
      running:
        startedAt: "2021-10-16T18:03:49Z"
  hostIP: 209.165.201.29
  phase: Running
  podIP: 209.165.201.3
  podIPs:
  - ip: 209.165.201.3
  - ip: 4141:4141::d32
  qosClass: Burstable
  startTime: "2021-10-16T18:03:47Z"
```

To view the summary of the pod details, use the following command.

- **kubectl get pods -n** *amf_namespace* **-o wide**

  Sample Output:

```
kubectl get pod -n amf-ins3 -o wide

NAME                                                READY   STATUS    RESTARTS
AGE     IP                NODE                       NOMINATED NODE   READINESS GATES
amf-ins3-amf-gosctp-lb-sctp-1-0                     1/1     Running   0
2d16h   209.165.201.29     amf-cndp-tb27d-master-1   <none>           <none>
amf-ins3-amf-gosctp-lb-sctp-1-1                     1/1     Running   0
2d16h   209.165.201.30                       amf-cndp-tb27d-master-2   <none>
  <none>
amf-ins3-amf-protocol-ep-default-0                  2/2     Running   0
2d16h   192.203.42.85     amf-cndp-tb27d-master-2   <none>           <none>
amf-ins3-amf-protocol-ep-default-1                  2/2     Running   0
2d16h   192.203.236.6     amf-cndp-tb27d-master-1   <none>           <none>
amf-ins3-amf-rest-ep-n0-0                           2/2     Running   0
2d19h   192.203.236.55    amf-cndp-tb27d-master-1   <none>           <none>
amf-ins3-amf-rest-ep-n0-1                           2/2     Running   0
2d19h   192.203.42.123    amf-cndp-tb27d-master-2   <none>           <none>
amf-ins3-amf-service-n0-0                           2/2     Running   1 (2d19h ago)
2d19h   192.203.211.179   amf-cndp-tb27d-master-3   <none>           <none>
amf-ins3-amf-service-n0-1                           2/2     Running   0
2d19h   192.203.211.131   amf-cndp-tb27d-master-3   <none>           <none>
amf-ins3-amf-service-n1-0                           2/2     Running   1 (2d19h ago)
```

```
2d19h   192.203.236.62    amf-cndp-tb27d-master-1   <none>           <none>
amf-ins3-amf-service-n1-1                           2/2     Running   0
2d19h   192.203.236.63    amf-cndp-tb27d-master-1   <none>           <none>
base-entitlement-amf-679fd785c-2zrnl                1/1     Running   0
3d12h   192.203.211.149   amf-cndp-tb27d-master-3   <none>           <none>
cache-pod-0                                         1/1     Running   0
2d19h   192.203.236.50    amf-cndp-tb27d-master-1   <none>           <none>
cache-pod-1                                         1/1     Running   0
2d19h   192.203.42.80     amf-cndp-tb27d-master-2   <none>           <none>
etcd-amf-ins3-etcd-cluster-0                        2/2     Running   0
3d11h   192.203.236.52    amf-cndp-tb27d-master-1   <none>           <none>
etcd-amf-ins3-etcd-cluster-1                        2/2     Running   0
3d11h   192.203.42.126    amf-cndp-tb27d-master-2   <none>           <none>
etcd-amf-ins3-etcd-cluster-2                        2/2     Running   0
3d11h   192.203.211.178   amf-cndp-tb27d-master-3   <none>           <none>
georeplication-pod-0                                1/1     Running   0
3d11h   209.165.201.31
                        amf-cndp-tb27d-master-2   <none>           <none>
grafana-dashboard-amf-774bdd8b6d-6t6kw              1/1     Running   0
3d11h   192.203.211.130   amf-cndp-tb27d-master-3   <none>           <none>
grafana-dashboard-etcd-amf-ins3-8597cf9fdc-72z7w    1/1     Running   0
3d11h   192.203.211.170   amf-cndp-tb27d-master-3   <none>           <none>
gtpc-ep-n0-0                                        2/2     Running   0
2d19h   192.203.236.22    amf-cndp-tb27d-master-1   <none>           <none>
gtpc-ep-n0-1                                        2/2     Running   0
2d19h   192.203.42.125    amf-cndp-tb27d-master-2   <none>           <none>
li-ep-n0-0                                          2/2     Running   0
3d11h   192.203.42.108    amf-cndp-tb27d-master-2   <none>           <none>
li-ep-n0-1                                          2/2     Running   0
3d11h   192.203.236.13    amf-cndp-tb27d-master-1   <none>           <none>
nodemgr-n0-0                                        2/2     Running   1 (2d19h ago)
2d19h   192.203.236.44    amf-cndp-tb27d-master-1   <none>           <none>
nodemgr-n0-1                                        2/2     Running   0
2d19h   192.203.211.150   amf-cndp-tb27d-master-3   <none>           <none>
oam-pod-0                                           2/2     Running   0
3d11h   192.203.211.146   amf-cndp-tb27d-master-3   <none>           <none>
ops-center-amf-ins3-ops-center-64479bd9d6-zqzzz     4/4     Running   0
3d12h   192.203.42.81     amf-cndp-tb27d-master-2   <none>           <none>
prometheus-rules-etcd-57688b5657-5gv5z              1/1     Running   0
3d11h   192.203.211.175   amf-cndp-tb27d-master-3   <none>           <none>
smart-agent-amf-ins3-ops-center-798d5f9884-zplrn    1/1     Running   0
3d12h   192.203.42.120    amf-cndp-tb27d-master-2   <none>           <none>
udp-proxy-0                                         1/1     Running   0
2d19h   198.51.100.10     amf-cndp-tb27d-master-1   <none>           <none>
udp-proxy-1                                         1/1     Running   0
2d19h   198.51.100.11     amf-cndp-tb27d-master-2   <none>           <none>
```

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

State	Description
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.

Viewing the Service Summary and Details

Use the following commands to view the service summary:

kubectl get svc -n *amf_namespace*

Sample Output:

```
kubectl get svc -n amf-ins1
NAME                                    TYPE        CLUSTER-IP        EXTERNAL-IP
PORT(S)                                       AGE
alert-frwd-ops-center                   ClusterIP   209.165.202.130   <none>
8080/TCP                                      29d
amf-gosctp-lb                           ClusterIP   209.165.202.140   <none>
7084/TCP                                      36h
amf-nrf-service                         ClusterIP   209.165.202.151   209.165.200.241
9090/TCP                                      36h
amf-protocol-ep                         ClusterIP   209.165.202.142   <none>
9003/TCP,8080/TCP                             36h
amf-rest-ep                             ClusterIP   209.165.202.145   <none>
9003/TCP,8080/TCP,9201/TCP                    36h
amf-sbi-service                         ClusterIP   209.165.202.156   209.165.200.241
8070/TCP                                      36h
amf-service                             ClusterIP   209.165.202.144   <none>
9003/TCP,8080/TCP                             36h
base-entitlement-amf                    ClusterIP   209.165.202.137   <none>
8000/TCP                                      29d
bgpspeaker-pod                          ClusterIP   209.165.202.154   <none>
9008/TCP,7001/TCP,8879/TCP                    36h
datastore-notification-ep               ClusterIP   209.165.202.135   209.165.200.240
8012/TCP                                      36h
documentation                           ClusterIP   209.165.202.134   <none>
8080/TCP                                      29d
etcd                                    ClusterIP   None              <none>
2379/TCP,7070/TCP                             36h
etcd-amf-ins1-etcd-cluster-0            ClusterIP   209.165.202.143   <none>
2380/TCP,2379/TCP                             36h
etcd-amf-ins1-etcd-cluster-1            ClusterIP   209.165.202.139   <none>
2380/TCP,2379/TCP                             36h
etcd-amf-ins1-etcd-cluster-2            ClusterIP   209.165.202.131   <none>
2380/TCP,2379/TCP                             36h
grafana-dashboard-amf                   ClusterIP   209.165.202.138   <none>
```

```
9418/TCP                                        36h
grafana-dashboard-app-infra-amf-ins1     ClusterIP   209.165.202.133   <none>
9418/TCP                                        36h
grafana-dashboard-etcd-amf-ins1          ClusterIP   209.165.202.141   <none>
9418/TCP                                        36h
gtpc-ep                                  ClusterIP   209.165.202.149   <none>
9003/TCP,8080/TCP                               36h
ldap-proxy-amf-ins1-oam-pod              ClusterIP   209.165.202.132   <none>
636/TCP,389/TCP                                 36h
li-ep                                    ClusterIP   209.165.202.150   <none>
9003/TCP,8080/TCP                               36h
local-ldap-proxy-amf-ins1-ops-center     ClusterIP   209.165.202.148   <none>
636/TCP,369/TCP                                 29d
netconf-ops-center-amf-ins1-ops-center   ClusterIP   209.165.202.155   209.165.200.231
2024/TCP                                        29d
nodemgr                                  ClusterIP   209.165.202.153   <none>
9003/TCP,8884/TCP,8879/TCP,9201/TCP,8080/TCP    36h
oam-pod                                  ClusterIP   209.165.202.147   <none>
9008/TCP,7001/TCP,8879/TCP,10080/TCP,8080/TCP   36h
ops-center-amf-ins1-ops-center           ClusterIP   209.165.202.152   <none>
8008/TCP,8080/TCP,2024/TCP,2022/TCP,7681/TCP    29d
prometheus-rules-etcd                    ClusterIP   None              <none>
9419/TCP                                        36h
smart-agent-amf-ins1-ops-center          ClusterIP   209.165.202.129   <none>
8888/TCP                                        29d
ssh-ops-center-amf-ins1-ops-center       ClusterIP   209.165.202.136   209.165.200.231
2025/TCP                                        29d
```

Use the following commands to view the comprehensive service details:

kubectl get svc -n *amf_namespace service_name* **-o yaml**

Sample Output:

```
kubectl get svc amf-rest-ep -n amf-ins1 -o yaml
apiVersion: v1
kind: Service
metadata:
  annotations:
    meta.helm.sh/release-name: amf-ins1-amf-rest-ep
    meta.helm.sh/release-namespace: amf-ins1
  creationTimestamp: "2021-10-16T18:00:23Z"
  labels:
    app: amf-rest-ep
    app.kubernetes.io/managed-by: Helm
    chart: amf-rest-ep-0.1.0-main-2464-211014124230-2d34ce7
    component: amf-rest-ep
    heritage: Helm
    release: amf-ins1-amf-rest-ep
  name: amf-rest-ep
  namespace: amf-ins1
  resourceVersion: "5768444"
  uid: 65cb4204-8914-4b71-aa3c-809238dd755e
spec:
  clusterIP: 209.165.202.145
  clusterIPs:
  - 209.165.202.145
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: grpc
    port: 9003
    protocol: TCP
    targetPort: 9003
```

```
  - name: metrics
    port: 8080
    protocol: TCP
    targetPort: 8080
  - name: nrfrestep
    port: 9201
    protocol: TCP
    targetPort: 9201
  selector:
    component: amf-rest-ep
    release: amf-ins1-amf-rest-ep
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
```

Multiple Service Pods on Multiple Nodes

Feature Description

This feature helps in bringing up service pods on two or more different nodes.

How it Works

To enable this feature, ensure the following:

- Set the configuration to two nodes and the number of replicas on each node, so that each node in an RU system has the number of configured amf-service replicas.
- When a node gets removed, the system won't be able to spawn a new service replica on the node that already has one.
- If one node (RU) gets removed from the three nodes and one replica configuration, the service pod previously running on the removed node (RU) respawns on the other available node (RU).
- The same changes can be configured for amf-rest-ep also.

Feature Configuration

To configure this feature, use the following sample configuration:

```
config
   instance instance-id instance-id
      endpoint { sbi | service }
         nodes number_of_nodes
         replicas number_of_replicas
         end
```

NOTES:

- **instance instance-id** *instance-id*—Specify the endpoint instance ID.
- **endpoint { sbi | service }**—Specify the endpoint that must be configured.

- **nodes** *number_of_nodes*—Specify the number of nodes that must be used for resiliency.
- **replicas** *number_of_replicas*—Specify the number of replicas that must be created for the endpoint, on each node.

Configuration Example

The following is an example configuration.

```
config
    instance instance-id 1
        endpoint service
            nodes 2
            replicas 2
        exit
    exit
exit
```

Configuration Verification

To verify the configuration:

```
show running-config instance instance-id 1 endpoint service
```

Note To allow the spawning of pods on the nodes, it's necessary to label the nodes appropriately in the cluster.

To verify the label:

```
show running-config k8 label service-layer
```

GTPC Protocol Endpoint Merge with UDP Proxy Bypass

Feature Description

AMF introduces the bypass proxy to enable the GTP packets to directly land on GTPC-EP pod. This bypass process avoids the processing at UDP-proxy and reduces the one hop during the packet forwarding. All the rolling upgrade features supported by existing GTPC-EP and UDP-proxy are integrated in new merged path.

Configuring GTPC EP Merge Mode

Note We recommend deploying the GTPC-EP in merge mode for better performance optimization.

To configure the merge mode for the GTPC EP, use the following sample configuration:

```
config
  instance instance-id instance_id
    endpoint endpoint_name
      nodes nodes_count
```

```
retransmission { timeout timeout_value | max-retryretry_count}
internal-vip ip_address
vip-ip ip_address
interface interface_name
   vip-ip ip_address
exit
```

NOTES:

- **instance instance-id** *instance_id*: Specifies the creation or configuration of an instance. An instance can be a logical unit or a virtual instance within the device.
- **endpoint** *endpoint_name*: Specifies the creation or configuration of an endpoint for GPRS Tunneling Protocol (GTP).
- **nodes** *nodes_count*: Specifies the number of nodes associated with the endpoint.
- **retransmission { timeout** *timeout_value* **| max-retry***retry_value***}**: Specifies the number of nodes associated with the endpoint.
  - **retransmission**: Configures the retransmission settings.
  - **timeout** *timeout_value*: Specify the retransmission timeout value in seconds.
  - **max-retry** *retry_count*: Specify the maximum number of retry attempts.
  - Recommended value for retransmission configuration: (max-retry count+1)*retransmission timeout should be < 30 second.
- **internal-vip** *ip_address*: Specifies the internal virtual IP address for the instance.
- **vip-ip** *ip_address*: Specifies the virtual IP address for the instance.
- **interface** *interface_name*: Specifies the network interface for which the configuration settings apply.
- **vip-ip** *ip_address*: Specifies the virtual IP address associated with the specified interface.
- Enabling/Disabling this configuration requires pod restart.