

# Pods and Services Reference


- Feature Summary and Revision History, on page 1
- Feature Description, on page 2
- Associating Pods to the Nodes, on page 8
- Configuring Pod Layout, on page 10
- Configuring Event Trace, on page 11


# 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 Changes in this Release | Subscriber Manager |
| Related Documentation | Not Applicable |

## Revision History

***Table 2: Revision History***

| Revision Details | Release |
|---|---|
| Enhancement Introduced:<br>cnBNG supports the pod layout configuration and event tracing configuration. | 2022.03.0 |

| Revision Details | Release |
|---|---|
| Enhancement Introduced:<br>The Subscriber Manager pod supports the charging functionality, that is, subscriber access and subscriber management. | 2021.03.0 |
| First introduced. | 2021.01.0 |

# Feature Description

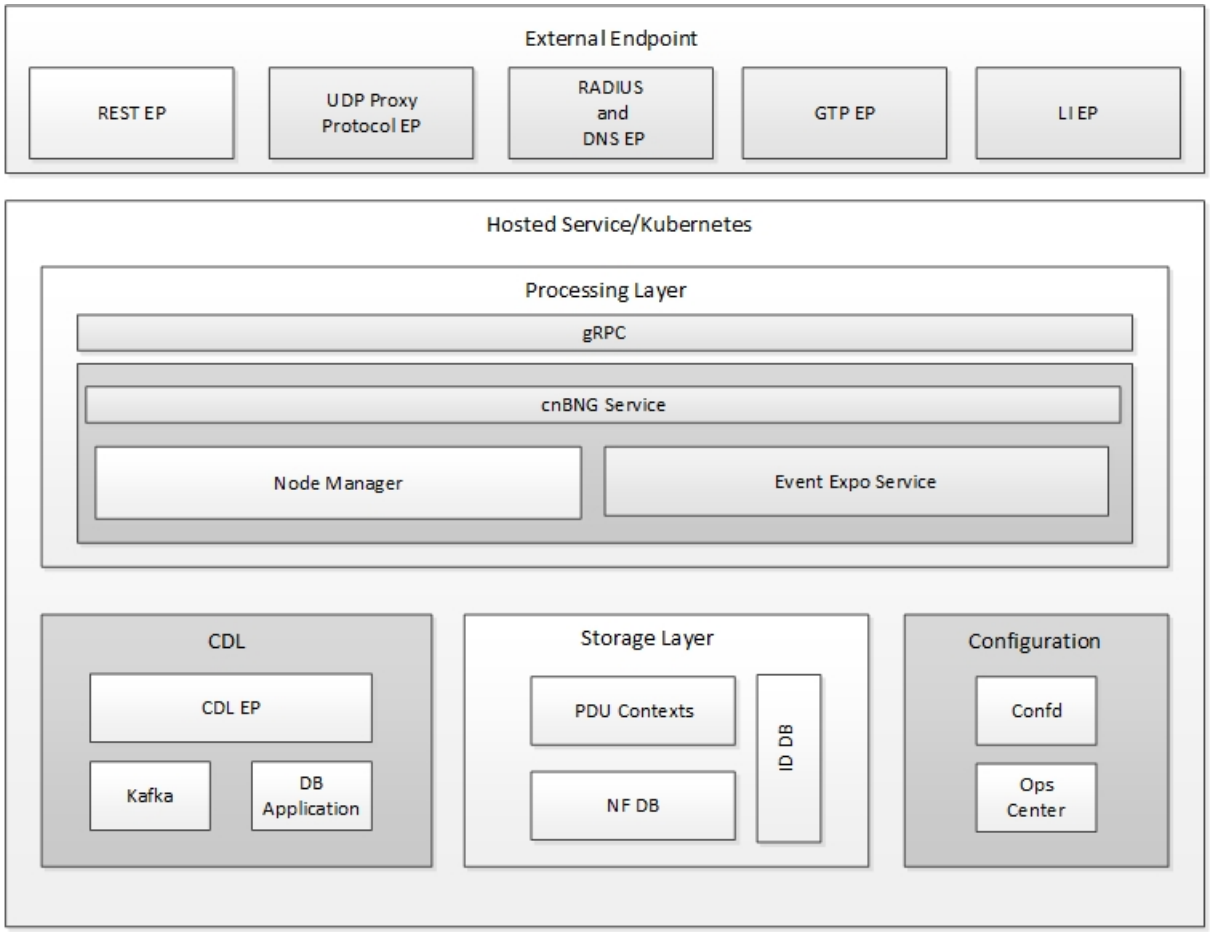
The SMI Ops Center is the platform to deploy cnBNG cluster with the offline or online repository. It is mandatory to deploy the SMI Ops Center to install the BNG Ops Center.

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

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

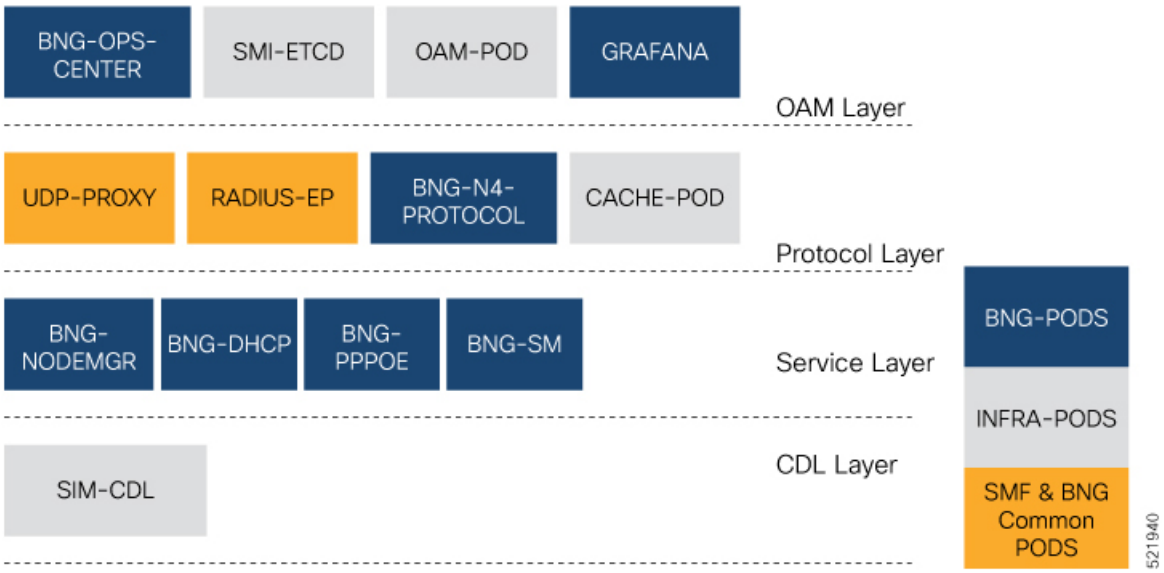
The following workflow provides a high-level visibility into the host machines, and the associated pods and services. It also represents how the pods interact with each other. The representation might defer based on your deployment infrastructure.

*Figure 1: Communication Workflow of Pods*




The following figure shows the cnBNG cluster pod layout.

*Figure 2: cnBNG Cluster Pod Layout*




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

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

The following sections provide more information on the Kubernetes components in cnBNG.

# Pods

A pod is a process that runs on your Kubernetes cluster. It encapsulates a granular unit known as a container. A pod contains 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 tables list the cnBNG and Common Execution Environment (CEE) 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: cnBNG Pods*

| Pod Name | Description | Host Name |
|---|---|---|
| api-bng-bng-ops-center | Functions as the *confD* API pod for the BNG Ops Center. | OAM |
| bng-dhcp-n0 | Operates as the DHCP server and handles all DHCP related control messages. | Service |
| bng-n4-protocol-n0 | Operates as encoder and decoder of application protocols (PFCP, GTP, RADIUS, and so on) whose underlying transport protocol is UDP. | Protocol |

| Pod Name | Description | Host Name |
|---|---|---|
| bng-nodemgr-n0 | Performs node level interactions Service such as N4 link establishment, management (heart-beat), and so on. | Service |
| bng-pppoe-n0 | Runs the combined Control Plane (CP) for PPPoE and PPP. | Service |
| bng-sm-n0 | Manages subscriber access and subscriber management functions such as Authentication, Authorization, Accounting, and policy management.. | Service |
| cache-pod-0 | 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-d0 | Provides an interface to the CDL. | Session |
| cdl-index-session-c1-m1-0 | Preserves the mapping of keys to the session pods. | Session |
| cdl-slot-session-c1-m1-0 | Operates as the CDL Session pod Session to store the session data. | Session |
| documentation | Contains the documentation. | OAM |
| etcd-bng-bng-etcd-cluster-0 | Hosts the etcd for the BNG application to store information such as pod instances, leader information, NF-UUID, endpoints, and so on. | OAM |
| grafana-dashboard-app-infra | Contains the default dashboard of app-infra metrics in Grafana. | OAM |
| grafana-dashboard-bng | Contains the default dashboard of the cnBNG-service metrics in Grafana. | OAM |
| grafana-dashboard-cdl | Contains the default dashboard of CDL metrics in Grafana. | OAM |
| kafka | Hosts theKafka details for the CDL replication. | Protocol |
| oam-pod | Operates as the pod to facilitate Ops Center actions like show commands, configuration commands, monitor protocol monitor subscriber, and so on. | OAM |
| ops-center-bng-bng-ops-center | Acts as the BNG Ops Center. | OAM |
| prometheus-rules-cdl | Contains the default alerting rules and recording rules for Prometheus CDL. | OAM |
| radius-ep-n0-0 | Operates as RADIUS endpoint of cnBNG. | |
| smart-agent-bng-bng-ops-center | Operates as the utility pod for the BNG Ops Center. | OAM |
| bng-udp-proxy-0 | Operates as proxy for all UDP messages. Owns UDP client and server functionalities. | Protocol |
| swift-bng-bng-ops-center | Operates as the utility pod for the BNG Ops Center. | OAM |

| Pod Name | Description | Host Name |
|---|---|---|
| zookeeper | Assists Kafka for topology management. | OAM |

*Table 4: CEE Pods*

| Pod Name | Description | Host Name |
|---|---|---|
| alert-logger | Stores the history of active and resolved alerts. | OAM |
| alertmanager | Duplicates alerts and sends out resolution of alerts when they are resolved in Prometheus. | OAM |
| api-cee-global-ops-center | Functions as the confD API pod for the CEE Ops Center. | OAM |
| bulk-stats | Assists to retrieve bulkstats saved by Prometheus containers. | OAM |
| cee-global-product-documentation | Contains the product documentation (API, CLI, and so on). | OAM |
| core-retriever | Assists in retrieving the core dumps. | All the nodes except ETCD nodes. |
| documentation | Contains the documentation (metrics and usage). | OAM |
| grafana-dashboard-metrics | Assists in collating Grafana metrics on the dashboard. | OAM |
| grafana | Contains the Grafana metrics for CEE. | OAM |
| kube-state-metrics | Assists in generating metrics about the state of Kubernetes objects: node status, node capacity (CPU and memory), and so on. | OAM |
| logs-retriever | Assists in retrieving Kernel, Kubelet, and Container level logs through output to JournalD driver. | All the nodes except ETCD nodes. |
| node-exporter | Exports the node metrics. | All the nodes. |
| ops-center-cee-global-ops-center | Provides NETCONF and CLI interface to the application. | OAM |
| path-provisioner | Provisions the local storage volume. | All the nodes except ETCD nodes. |
| pgpool | *Pgpool* is a middleware that works between *PostgreSQL* servers and a *PostgreSQL* database. | OAM |
| postgres | Storage of alerts and Grafana dashboards. | OAM |

| Pod Name | Description | Host Name |
|---|---|---|
| prometheus-hi-res | Stores all metrics and generates alerts by alerting rules. | OAM |
| prometheus-rules | Contains the default alerting rules and recording rules for Prometheus. | OAM |
| prometheus-scrapeconfigs-synch | Synchronizes the Prometheus scrape configuration. | OAM |
| pv-manager | Provisions the local storage volume. | OAM |
| pv-provisioner | Provisions the local storage volume. | OAM |
| show-tac-manager | Assists in creating and deleting debug package. | OAM |
| smart-agent-cee-global-ops-center | Operates as the utility pod for the CEE Ops Center. | OAM |
| snmp-trapper | Sends the SNMP traps based on triggered alerts. | OAM |
| swift-cee-global-ops-center | Operates as the utility pod for the CEE Ops Center. | OAM |
| thanos-query-hi-res | Implements the Thanos query for Prometheus HA. | OAM |
| fluentbit | Assists in log forwarding to the external logs collector. | All the nodes except ETCD nodes. |

# Services

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

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

***Table 5: BNG Services and Pods***

| Service Name | Pod Name | Description |
|---|---|---|
| bng-nodemgr | bng-nodemgr-n0 | Responsible for node level interactions Serv such as N4 link establishment, management (heart-beat), and so on. |
| bng-dhcp | bng-dhcp-n0 | Functions as the DHCP server and handles a DHCP related control messages. |
| bng-pppoe | bng-pppoe-n0 | Functions as the combined Control Plane (C PPPoE and PPP. |

| Service Name | Pod Name | Description |
|---|---|---|
| bng-sm | bng-sm-n0 | Manages subscriber access and subscriber management functions such as Authentication, Authorization, Accounting, and policy management.. |

## Open Ports and Services

cnBNG uses different ports for communication purposes. The following table describes the default open ports and the associated services in an SMI based cnBNG system.

### Application Infrastructure (App-infra)

| Port | Service |
|---|---|
| 8850 | Golang net/HTTP server TCP Golang net/HTTP server |
| 8879 | Golang net/HTTP server TCP Golang net/HTTP server |
| 8850 | DefaultPProfPort |
| 8879 | DefaultAdminEndPointPort |

### UDP

| Port | Service | CP to UP Interfaces |
|---|---|---|
| 2152 | GTPU | CPRi |
| 8805 | PFCP | SCi |

# 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:

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

```
config
k8 label protocol-layer key key_value vm-type value protocol
exit
k8 label service-layer key key_value vm-type value service
exit
```

```
k8 label cdl-layer key key_value vm-type value cdl
exit
k8 label oam-layer key key_value vm-type value oam
exit
```

**NOTES:**

- If you opt not to configure the labels, then BNG assumes the labels with the default key-value pair.
  - **k8 label protocol-layer key** *key_value* **vm-type** *value* **protocol**: Configures the key value pair for protocol layer.
  - **k8 label service-layer key** *key_value* **vm-type** *value* **service**: Configures the key value pair for the service layer.
  - **k8 label cdl-layer key** *key_value* **vm-type** *value* **cdl**: Configures the key value pair for CDL.
  - **k8 label oam-layer key** *key_value* **vm-type** *value* **oam**: Configures the key value pair for OAM layer.

# Viewing the Pod Details and Status

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

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

1. To view the comprehensive pod details, use the following command.

   ```
   kubectl get pods -n bng_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 pod is in the current state.

2. Use the following command to view the summary of the pod details.

   ```
   kubectl get pods -n bng_namespace -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.<br>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. |

# Configuring Pod Layout

To configure the cnBNG pod layout when the virtual machine is short of CPU and memory resources, use the following sample configuration:

```
config
   instance instance-id instance_id
     endpoint sm
        cpu { max-process max_os_threads | request cpu_resource_request }
        memory { limit max_memory_resource | request memory_resource_request }
        end
```

**NOTES:**

- **cpu { max-process** *max_os_threads* **| request** *cpu_resource_request* **}**: Enable the K8s pod CPU configuration.
  - **max-process** *max_os_threads*: Specify the maximum number of parallel OS threads to use. *max_os_threads* must be an integer in the range of 1 to 32.
  - **request** *cpu_resource_request*: Specify the CPU resource request in millicores. *cpu_resource_request* must be an integer in the range of 100 to 1000000.
- **memory { limit** *max_memory_resource* **| request** *memory_resource_request* **}**: Enable the K8s pod memory configuration.
  - **limit** *max_memory_resource*: Specify the maximum number of used memory resources in megabytes. *max_memory_resource* must be an integer in the range of 100 to 200000.
  - **request** *memory_resource_request*: Specify the memory resource request in megabytes. *memory_resource_request* must be an integer in the range of 100 to 200000.

# Configuring Event Trace

To configure event tracing, use the following sample configuration:

```
config
  subscriber [ event-trace-disable | event-trace-max-count event_trace_count
 ]
  end
```

**NOTES:**

- **event-trace-disable**: Disable subscriber event tracing. cnBNG uses event traces for session level event history in CDL records.
- **event-trace-max-count** *event_trace*: Specify the number of entries in event tracing.

  *event_trace* must be an integer in the range of 1 to 8192. Default value: 100.