Cisco Prime Network Registrar on Kubernetes

Kubernetes is an open source container orchestration system for automating software deployment, scaling, and management. Starting from Cisco Prime Network Registrar 11.2, you can deploy the Cisco Prime Network Registrar instances on Kubernetes using the following Docker images:

- For regional instance deployment: **cpnr-regional-11.2-1.el8.x86_64_rhel_docker.tar.gz**
- For DHCP and DNS local instance deployment: **cpnr-local-11.2-1.el8.x86_64_rhel_docker.tar.gz**
- For CDNS local instance deployment: **cpnr-cdns-11.2-1.el8.x86_64_rhel_docker.tar.gz**

Note The names of the images will change with releases in future.

You need to have a private docker registry in Kubernetes to load the container image.

This chapter contains the following section:

Deploying Cisco Prime Network Registrar regional and local Instance on Kubernetes

You can deploy the Cisco Prime Network Registrar instances on Kubernetes using the YAML files. YAML is the standard format used in a Kubernetes configuration file. Cisco Prime Network Registrar kit **cpnr-11.2-1.el8.x86_64_kubernetes.tar.gz** contains examples of YAML files (cpnr-local-statefulset.yaml and cpnr-regional-statefulset.yaml), which depict one of the ways of deploying Cisco Prime Network Registrar on Kubernetes.

Note No special license is required to deploy Cisco Prime Network Registrar in the Kubernetes environment. It uses the existing container licenses.

For example, you can use the cpnr-local-statefulset.yaml in the Cisco Prime Network Registrar kit to create the Cisco Prime Network Registrar local instance on Kubernetes. This configuration uses StatefulSet and hostNetwork deployment. Instance created using this YAML will bind the Cisco Prime Network Registrar local instance to the configured worker node. This instance will run only on the configured worker node and it will not run on the other node.

When a pod is configured with **hostNetwork: true**, the applications running in the pod can directly see the network interfaces of the host machine where the pod was started. An application that is configured to listen on all network interfaces will in turn be accessible on all the network interfaces of the host machine.

YAML file consists of the following Kubernetes resources:

- Service

  In cpnr-local-statefulset.yaml, cpnr-local is the service name and since Headless service is used, the clusterIp is set to None.

- StatefulSet

  In cpnr-local-statefulset.yaml, cpnr-local is the StatefulSet name and is used to run the Cisco Prime Network Registrar 11.2 docker image with replica of one.

  StatefulSet is used in Cisco Prime Network Registrar for two reasons:

  - Constant pod names
  - When pods created using deployment are deleted, it creates a new pod before the old pod is completely terminated. Since hostNetwork is used before the old Cisco Prime Network Registrar process on the host node is termianted completely, the new Cisco Prime Network Registrar pod is created and the Cisco Prime Network Registrar services on the new pod is down since the Cisco Prime Network Registrar process on the old pod is not termianted completely.

    This is overcome by StatefulSet where the old pod is terminated completely and only then the new pod is created.

Note Cisco Prime Network Registrar has been tested using the hostNetwork, as other networking modes had issues with HA and failover pair. In the hostNetwork mode, when a pod starts, it uses the host network namespace and host IP address. This basically means that the pod can see all the network interfaces of the host. With hostNetwork mode, only one Cisco Prime Network Registrar instance can be deployed in a node and hence the replica in the YAML is set to 1. If multiple Cisco Prime Network Registrar pods are to be deployed in the same node using hostNetwork mode, then for each pod, you need to change all Cisco Prime Network Registrar related ports and adjust the replicas accordingly. However, this may not be useful.

Step 1 Configure the following parameters in the YAML file:

- NODE_NAME: Worker node name, where the Cisco Prime Network Registrar instance will run. For example, “cnr-k8s-worker2.*server*.com”. Use the **kubectl get nodes** command to get the node name.
- IMAGE: The Docker image location. For example “cnr-k8s-worker1.server.com/cpnr-local:11.2”, where cnr-k8s-worker1.*server*.com is the private registry and the image name is cpnr-local with tag 11.2.
- HOST_MOUNT_PATH: Directory path on the host machine. This directory will be used to store configuration files and data on the Cisco Prime Network Registrar instance. /var/nwreg2 of the pod is mapped to HOST_MOUNT_PATH

in the host machine. This is required to persist the data ofCisco Prime Network Registrar instance on the host machines.

Step 2 Create the Cisco Prime Network Registrar instance on Kubernetes using the following command:

- For Cisco Prime Network Registrar local instance deployment:

  ```
  # kubectl create -f cpnr-local-statefulset.yaml
  ```

- For Cisco Prime Network Registrar regional instance deployment:

  ```
  # kubectl create -f cpnr-regional-statefulset.yaml
  ```

Step 3 Check the Cisco Prime Network Registrar instance details on Kubernetes using the following command:

```
# kubectl get all
```

Step 4 Log into the Cisco Prime Network Registrar instance pod using the following command:

```
# kubectl exec -it <pod name> -- bash
```

For example:

```
# kubectl exec -it cpnr-dhcp-dns-0 -- bash
# /opt/nwreg2/local/usrbin/nrcmd -s
100 Ok
```

Step 5 Set the user name and password, and register the local pods to regional pods.

If you want to delete the Cisco Prime Network Registrar instance on Kubernetes, use the following command:

- For Cisco Prime Network Registrar local instance:

  ```
  # kubectl delete -f cpnr-local-statefulset.yaml
  ```

- For Cisco Prime Network Registrar regional instance:

  ```
  # kubectl delete -f cpnr-regional-statefulset.yaml
  ```

To debug any pod failures, use the **kubectl logs** *podname* or **kubectl describe pod** *podname* command.

Note If you want to deploy Cisco Prime Network Registrar pod in another worker node, you need to make changes in the YAML file (for example, you need to change service.metadata.name and statefulset.metadata.name.).

Note Cisco Prime Network Registrar 11.2 Docker images have been tested with Kubernetes version 1.27. CNI used is Calico 3.22.0. The entire testing is carried out using the YAML files provided as examples. If you want to change the YAML file, it is your responsibility to test it before moving it to production.

Deploying Cisco Prime Network Registrar CDNS instance on Kubernetes

You can deploy the Cisco Prime Network Registrar Caching DNS Server instances on Kubernetes using the YAML files. **cpnr-11.2-1.el8.x86_64_kubernetes.tar.gz** contains examples of YAML files (cpnr-cdns-deployment.yaml).

Starting Cisco Prime Network Registrar 11.2, a separate CDNS container is deployed to make CDNS service more cloud native (smaller, stateless, and, horizontally scalable). CDNS on Kubernetes now supports auto scaling, CDNS service will start only if corresponding license is available on Regional Cluster, so it is highly recommended to use Smart License for CDNS deployment on Kubernetes to make scaling seamless. Before deploying the CDNS pod, make sure to which regional server this CDNS pod will be registered.

Note No special license is required to deploy Cisco Prime Network Registrar in the Kubernetes environment. It uses the existing container licenses.

In case if you prefer to use Flexlm license, there must be sufficient buffer of licenses on Regional Cluster to avoid CDNS registration/deregistration issues as license update is not instant. In this scenario if services exceed the licenses and the pods are stuck, you might need to manually clean-up the CDNS clusters which are not in use but have not been automatically deregistered due to license mismatch.

The CDNS container does not have any of the database such as CCM and no client interfaces like Web UI, or CLI, or REST API. All the configuration and management for CDNS happens from the regional and whatever configuration is required for CDNS, it has to be done from regional.

Note Hybrid mode and its configuration is not applicable to a CDNS service pod.

SNMP Server is not applicable for CDNS service pod, it is recommended to use the Open Source or Kubernetes native technologies like Prometheus, Grafana etc for monitoring, stats collection, and alerting.

The utilities like cnr_exim and cnr_tactool are available and they connects to regional for configuration related data. A new utility cnr_cmd used is to reload the server without restarting the CDNS pod. The examples of the utilities used and debug settings is also shown below:

```
./cnr_tactool
user: admin
password:
Collecting CPNR Information:
Finding CNR Version...
Getting CNR Configuration File...
Searching for core files in '/var/nwreg2/cores/'...
Searching for Java HotSpot Error Logs...
Copying CNR Logs...
Exporting CNR Configuration...
Collecting System Information:
Gathering System Information...
Copying System Logs...
Getting System Processor Information...
Getting System Memory...
Getting Resource Limitation Information...
```

```
Getting IP Configuration Information...
Getting Cron Information...
Getting Process List...
Getting Services List...
Getting Socket Stats...
Getting Disk Space Utilization Information...
Getting Network Statistics...
Getting Kernel Routing Tables...
Getting Network Statistics...
Getting Kernel Parameters...
Getting IP Tables Information...
Getting nwreglocal.service journal entries...
Getting Curl Version Information...
Generating HTML File...
Successfully processed and archived data in
'/opt/nwreg2/local/temp/cnr_tac-9-19-2023-01.tar.gz'
```

```
./cnr_exim -e -x export-data
user: admin
password:
-----
```

```
./cnr_cmd -c cdns reload
```

```
./cnr_cmd -c "cdns setDebug AQW=1"
cdns debug set to AQW=1.
```

```
./cnr_cmd -c "cdns unsetDebug"
cdns debug cleared.
```

For CDNS container, the YAML file consists of the Kubernetes resource Service where there are multiple CDNS services available and that is called the node port type wherein it just runs as a normal Kubernetes cloud native service and can be changed it to a load balancer.

Refer to the sample code for the Kubernetes resource Service:

```
apiVersion: v1
kind: Service
metadata:
  name: cpnr-cdns-svc
  labels:
    app: cpnr-cdns-svc
spec:
  type: NodePort
  clusterIP: 10.8.7.6
  ports:
    - protocol: UDP
      name: cdns-tcp
      port: 53
      targetPort: 53
      nodePort: 53
    - protocol: TCP
      name: cdns-udp
      port: 53
      targetPort: 53
      nodePort: 53
    - protocol: TCP
      name: cdns-metrics
      port: 8000
      targetPort: 8000
      nodePort: 8000
  selector:
    app: cpnr-cdns
```

Step 1 Configure the following parameters in the YAML file:

- Regional IP: The regional IP where the CDNS server will be registered.
- Image: The Docker image location. For example “cnr-k8s-worker1.server.com/cpnr-local:11.2”, where cnr-k8s-worker1.server.com is the private registry and the image name is cpnr-local with tag 11.2.
- Host mount path: Directory path on the host machine. This directory will be used to store configuration files and data on the Cisco Prime Network Registrar instance. /var/nwreg2 of the pod is mapped to HOST_MOUNT_PATH in the host machine. This is required to persist the data of Cisco Prime Network Registrar instance on the host machines.

Step 2 Create the Cisco Prime Network Registrar CDNS local instance deployment on Kubernetes using the following command:

- **# kubectl create -f cpnr-cdns-deployment.yaml**

Step 3 Check the Cisco Prime Network Registrar CDNS local instance deployment on Kubernetes using the following command:

- **# kubectl get all**

Step 4 Log into the Cisco Prime Network Registrar instance pod using the following command:

```
# kubectl exec -it <pod name> -- bash
```

For example:

```
# kubectl exec -it cpnr-cdns-5795ff8d47-9zzzq -- bash
# /opt/nwreg2/local/usrbin/nrcmd -s
100 Ok
```

Step 5 Set the user name and password, and register the local pods to regional pods.

If you want to delete the Cisco Prime Network Registrar instance on Kubernetes, use the following command:

- For Cisco Prime Network Registrar CDNS instance:

```
# kubectl delete -f cpnr-cdns-deployment.yaml
```

To debug any pod failures, use the **kubectl logs** *podname* or **kubectl describe pod** *podname* command.

Note Force deletion of CDNS service pod using the following command does not remove cluster or service-pod entry on regional cluster.

```
#kubectl delete pod <name> --force
```

Cisco Prime Network Registrar 11.2 Docker images have been tested with Kubernetes version 1.28.2. CNI used is Calico 3.26.1. The entire testing is carried out using the YAML files provided as examples. If you want to change the YAML file, it is your responsibility to test it before moving it to production.

Container Logging

CDNS supports Kubernetes native logging and the logs like kubectl logs, and CDNS pods can be accessed. The CDNS logs can be configured with external logging system for proper log management. The CDNS logs are deployed and tested with Elastic search, Fluentd and Kibana (EFK) stack.

The command that can be used to retrieve the logs for a specific CDNS pod running in a Kubernetes cluster is:

```
kubectl logs <cdns-pod>
```

The CDNS logs example in Kubernetes cluster:

```
kubectl logs cpnr-cdns-7dfbc4bb79-68r6t
```

CDNS Metrics

The CDNS metrics (stats) are exposed in the cloud native way to make it accessible to a monitoring system such as Prometheus.

Note CDNS writes its stats periodically to a file named cdns_metrics for every 10 seconds.

All CDNS stats will be exported. Upstream services such as Prometheus and Grafana can do further filtering to fit the customer's monitoring requirements. This allows the customer to have the most flexibility over what metrics are monitored. Additionally, since the metrics are stored long term, it also allows the customer to make changes on the fly without needing to change anything about their deployment. The CDNS metrics has been tested using Prometheus.

CDNS Scaling

CDNS Service can be scaled up/down horizontally using Kubernetes HPA (Horizontal Pod Autoscaling). HPA can be configured for the CDNS deployment. Customers can create their own HPA based on the metric(s) they want to monitor for scaling the CDNS service.

Below example depicts HPA based on ‘avg_queries_per_second’ custom metric. This metric has been generated based on ‘queries_total’ CPNR metric by configuring appropriate ConfigMap in the Prometheus Adapter.

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: cdns-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: cpnr-cdns
  minReplicas: 2
  maxReplicas: 6
  metrics:
  - type: Pods
    pods:
      metric:
        name: avg_queries_per_second
      target:
        type: AverageValue
```

```
        averageValue: 75000

apiVersion: v1
kind: ConfigMap
metadata:
  name: adapter-config
  namespace: monitoring
data:
  config.yaml: |
    rules:
    - seriesQuery: 'queries_total'
      resources:
        overrides:
          kubernetes_namespace:
            resource: namespace
          kubernetes_pod_name:
            resource: pod
      name:
        matches: "queries_total"
        as: "queries_per_second_avg"
      metricsQuery: sum(rate(<<.Series>>{<<.LabelMatchers>>}[1m])) by (<<.GroupBy>>)
```

Upgrading Cisco Prime Network Registrar Kubernetes

To upgrade the existing Cisco Prime Network Registrar Kubernetes to the latest version of Cisco Prime Network Registrar Kubernetes, do the following:

Step 1 Check the Cisco Prime Network Registrar instance details on Kubernetes using the following command:

```
#kubectl get all
```

Step 2 Remove the existing Cisco Prime Network Registrar instance on Kubernetes using the following steps:

- For Cisco Prime Network Registrar local instance:

  ```
  #kubectl delete -f cpnr-local-statefulset.yaml
  ```

- For Cisco Prime Network Registrar regional instance:

  ```
  #kubectl delete -f cpnr-regional-statefulset.yaml
  ```

Note Make sure not to do the cleanup operations (that is, retain the data, cnr.conf, and so on). Directory mentioned as HOST_MOUNT_PATH in the yaml file must not be deleted.

Step 3 To avoid issues with Java upgrades, it is highly recommended to edit the cnr.conf file (under HOST_MOUNT_PATH) to replace the cnr.java-home entry path with /usr/bin/java.

Step 4 Deploy Cisco Prime Network Registrar 11.2 instance in Kubernetes. For installation instructions, see Deploying Cisco Prime Network Registrar CDNS instance on Kubernetes, on page 4.

Step 5 To debug any pod failures, use the following commands and take corrective actions based on the failure:

```
#kubectl logs podname
```

or

```
#kubectl describe pod podname
```