Kubernetes Beginner-Friendly Documentation

Introduction

This documentation walks you through essential Kubernetes operations: creating a service using the nginx image, managing pods and services, deploying a StatefulSet, and implementing a DaemonSet. All technical terms are explained in simple language to make Kubernetes accessible even for beginners.

Task 1: Create a Kubernetes Service using nginx image

Command:
kubectl create deployment my-deployment1 --image=nginx

Explanation:

- kubectl: A command-line tool to interact with Kubernetes.

- create deployment: Creates a Deployment, which manages Pods and ensures they stay running.

- my-deployment1: Name for the deployment.

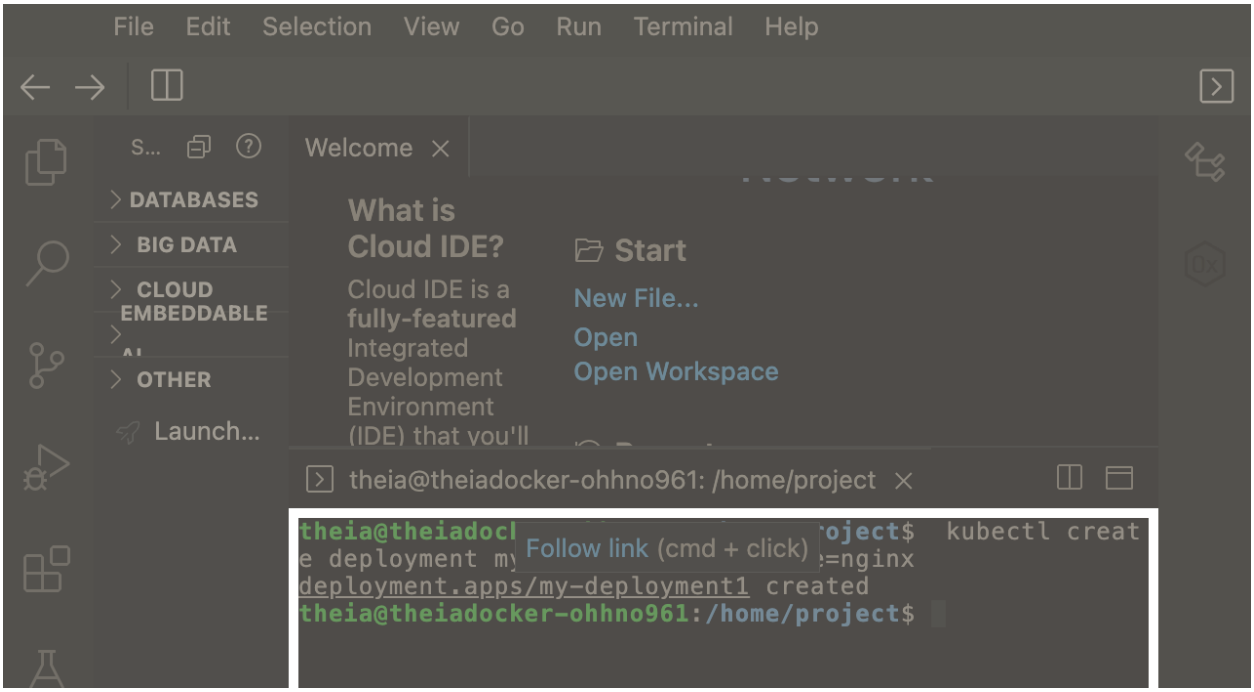
- --image=nginx: Uses the official nginx web server container image.

Command:
kubectl expose deployment my-deployment1 --port=80 --type=NodePort --name=my-service1

This command creates a Service named my-service1 that routes traffic to the nginx app on port 80. The type 'NodePort' exposes it externally on a high-numbered port.

Command:
kubectl get services

Lists all services. Services give consistent network access to Pods, even if Pods are restarted.




```
theia@theiadocker-ohhno961:/home/project$   kubectl expo
se deployment my-deployment1 --port=80 --type=NodePort -
-name=my-service1
service/my-service1 exposed
theia@theiadocker-ohhno961:/home/project$
```

```
theia@theiadocker-ohhno961:/home/project$  kubectl get s
ervices
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
my-service1   NodePort    172.21.155.178   <none>
80:30339/TCP   49s
theia@theiadocker-ohhno961:/home/project$
```

Task 2: Manage Kubernetes Pods and Services

Command:
kubectl get pods

Lists all Pods currently running.

Command:

kubectl get pod <pod-name> --show-labels

Displays metadata tags (labels) for a specific Pod. Labels are used for filtering and grouping.

Command:

kubectl label pods <pod-name> environment=deployment

```
theia@theiadocker-ohhno961:/home/project$  kubectl get p
ods
NAME                                READY   STATUS    RESTA
RTS   AGE
my-deployment1-65b7c8bd8-qbccd   1/1     Running   0
      5m5s
theia@theiadocker-ohhno961:/home/project$
```

```
theia@theiadocker-ohhno961:/home/project$  kubectl get p
od my-deployment1-65b7c8bd8-qbccd --show-labels
NAME                                READY   STATUS    RESTA
RTS   AGE    LABELS
my-deployment1-65b7c8bd8-qbccd   1/1     Running   0
      7m4s   app=my-deployment1,pod-template-hash=65b7c8
bd8
theia@theiadocker-ohhno961:/home/project$
```

```
theia@theiadocker-ohhno961:/home/project$   kubectl labe
l pods my-deployment1-65b7c8bd8-qbccd environment=deploy
ment
pod/my-deployment1-65b7c8bd8-qbccd labeled
theia@theiadocker-ohhno961:/home/project$
```

```
theia@theiadocker-ohhno961:/home/project$  kubectl get p
od my-deployment1-65b7c8bd8-qbccd --show-labels
NAME                                READY   STATUS    RESTA
RTS   AGE   LABELS
my-deployment1-65b7c8bd8-qbccd   1/1     Running   0
      12m   app=my-deployment1,environment=deployment,po
d-template-hash=65b7c8bd8
theia@theiadocker-ohhno961:/home/project$
```

Adds a label to a Pod. Labels help organize and select Pods.

Command:
kubectl run my-test-pod --image=nginx --restart=Never

Runs a temporary Pod using the nginx image. '--restart=Never' means it won't auto-restart.

Command:
kubectl logs <pod-name>

Retrieves log output from a specific Pod to help with debugging.

```
theia@theiadocker-ohhno961:/home/project$   kubectl run
my-test-pod --image=nginx --restart=Never
pod/my-test-pod created
theia@theiadocker-ohhno961:/home/project$
```

```
theia@theiadocker-ohhno961:/home/project$ kubectl logs m
y-deployment1-65b7c8bd8-qbccd
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empt
y, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /doc
ker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10
-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the check
sum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on
 IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-
local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20
-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30
-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for
 start up
2025/08/07 15:27:32 [notice] 1#1: using the "epoll" even
t method
2025/08/07 15:27:32 [notice] 1#1: nginx/1.29.0
2025/08/07 15:27:32 [notice] 1#1: built by gcc 12.2.0 (D
ebian 12.2.0-14+deb12u1)
2025/08/07 15:27:32 [notice] 1#1: OS: Linux 6.8.0-63-gen
eric
2025/08/07 15:27:32 [notice] 1#1: getrlimit(RLIMIT_NOFIL
E): 1048576:1048576
2025/08/07 15:27:32 [notice] 1#1: start worker processes
2025/08/07 15:27:32 [notice] 1#1: start worker process 2
9
```

Task 3: Deploying a StatefulSet

StatefulSets are used for applications that need stable network identities and persistent storage.

File: statefulset.yaml

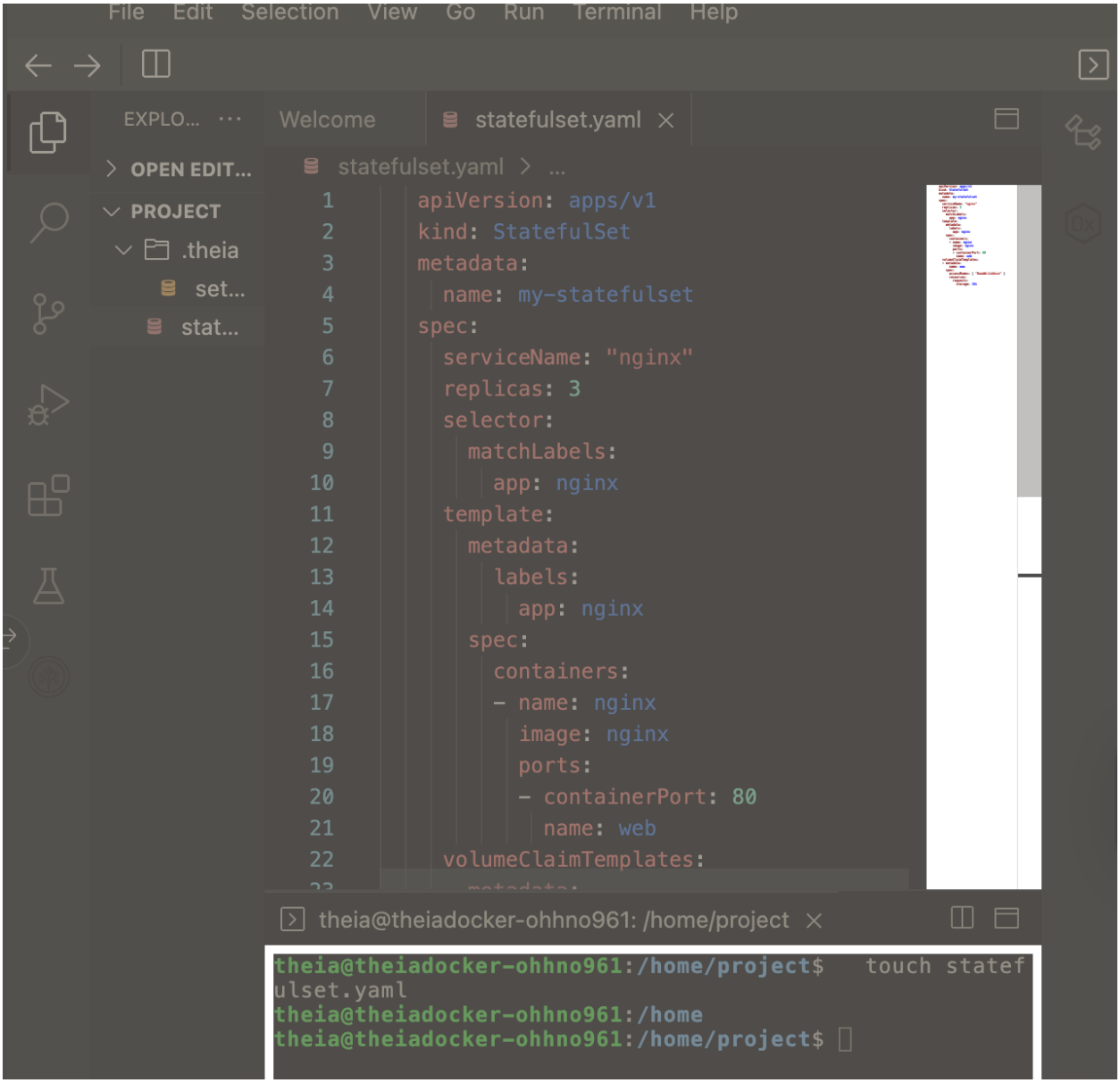
Key Terms:

- StatefulSet: Ensures each Pod has a unique, sticky identity.

- replicas: Number of Pods to run.

- volumeClaimTemplates: Creates persistent volumes for each Pod.

Command:
kubectl apply -f statefulset.yaml



Deploys the StatefulSet as described in the YAML file.

Command:

kubectl get statefulsets

Verifies the StatefulSet was successfully created.

```
theia@theiadocker-ohhno961:/home/project$    kubectl app
ly -f statefulset.yaml
statefulset.apps/my-statefulset created
theia@theiadocker-ohhno961:/home/project$    kubectl get
 statefulsets
NAME             READY   AGE
my-statefulset   0/3     16s
theia@theiadocker-ohhno961:/home/project$
```

Task 4: Implementing a DaemonSet

DaemonSets ensure a copy of a Pod runs on every node (or selected nodes). Useful for system-wide agents.

File: daemonset.yaml

Key Terms:

- DaemonSet: Runs one Pod per node.

- matchLabels: Determines which Pods are controlled by this DaemonSet.

Command:
kubectl apply -f daemonset.yaml

Applies the DaemonSet configuration.

Command:
kubectl get daemonsets

Lists details about the DaemonSet, such as how many Pods are ready and running.

File Edit Selection View Go Run Terminal Help

EXPLORER

OPEN EDITORS

PROJECT

.theia

settings.json

daemonset.yaml

statefulset.yaml

statefulset.yaml | daemonset.yaml

daemonset.yaml > {} spec > {} template > {}

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: my-daemonset
spec:
  selector:
    matchLabels:
      name: my-daemonset
  template:
    metadata:
      labels:
        name: my-daemonset
    spec:
```

theia@theiadocker-ohhno961: /home/proje

```
theia@theiadocker-ohhno961:/home/project$
touch daemonset.yaml
theia@theiadocker-ohhno961:/home/project$
```

```
theia@theiadocker-ohhno961:/home/project$  k
ubectl apply -f daemonset.yaml
daemonset.apps/my-daemonset created
theia@theiadocker-ohhno961:/home/project$
```

```
theia@theiadocker-ohhno961:/home/project$
 kubectl get daemonsets
NAME           DESIRED   CURRENT   READY   U
P-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
my-daemonset   7         6         6       6
            6           <none>          12s
theia@theiadocker-ohhno961:/home/project$
```

Conclusion

You now understand how to create services, manage pods, deploy stateful applications, and run DaemonSets across all nodes. These concepts are foundational for mastering Kubernetes in any real-world DevOps or cloud-native environment.