Certified Kubernetes Administrator

©Simplilearn. All rights reserved.

Kubernetes: Application Lifecycle Management

©Simplilearn. All rights reserved.

Learning Objectives

By the end of this lesson, you will be able to:

- Update deployment rolling using kubectl command and by editing the deployment file
- Check rollback status, pause, and resume rollback
- Create a pod with arguments and commands
- Create a configmap file
- Scale the applications up and down
- Demonstrate the use of initcontainers and self-healing

©Simplilearn. All rights reserved.

Rolling Updates and Rollbacks

©Simplilearn. All rights reserved.

Deployment Rolling Update Using Kubectl Rollout

Problem Statement: You are given a project to update deployment rolling using kubectl rollout.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Deployment Rolling Update by Editing the Deployment File

Problem Statement: You are given a project to update deployment rolling by editing the deployment file.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Rollout Status Check

Problem Statement: You are given a task to check the rollout status.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Pause and Resume the Rollout

Problem Statement: You are given a project to demonstrate the pausing and resuming of the rollout.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Rollback Using Command and Deployment File

Problem Statement: You are given a project to demonstrate the rollback using both command and deployment file.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Updating the Rollback Deployment Method

Problem Statement: You are given a project to update the rollback deployment method.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Configuring Applications

©Simplilearn. All rights reserved.

Configuring Applications with Commands

Follow the steps below to configure applications with commands:

```
kubectl config view # Show Merged kubeconfig settings.
# use multiple kubeconfig files at the same time and view merged config
KUBECONFIG=~/.kube/config:~/.kube/kubconfig2
kubectl config view
# get the password for the e2e user
kubectl config view -o jsonpath='{.users[?(@.name == "e2e")].user.password}'
kubectl config view -o jsonpath='{.users[].name}'    # get a list of users
kubectl config get-contexts          # display list of contexts
kubectl config current-context   # display the current-context
kubectl config use-context my-cluster-name  # set the default context to my-cluster-name
# add a new cluster to your kubeconf that supports basic auth
kubectl config set-credentials kubeuser/foo.kubernetes.com --username=kubeuser --
password=kubepassword
# permanently save the namespace for all subsequent kubectl commands in that context.
kubectl config set-context --current --namespace=ggckad-s2
# set a context utilizing a specific username and namespace.
kubectl config set-context gce --user=cluster-admin --namespace=foo \
  && kubectl config use-context gce
kubectl config unset users.foo        # delete user foo
```

©Simplilearn. All rights reserved.

simplilearn

Configuring Applications with Arguments

The characteristics of configuring applications with commands:

- Define a command and arguments for the containers that run in the pod, when you create a pod.
- Include the **command** field in the configuration file, to define a command.
- Include the **args** field in the configuration file, to define arguments for the command.
- Once the pod is created, the command and arguments cannot be changed.
- Do not define a command, if you are defining args, as the default command is used with your new arguments.

©Simplilearn. All rights reserved.

simplilearn

Configuring Applications with ConfigMap

ConfigMaps allow you to decouple configuration artifacts from image content to keep containerized applications portable.

It is used to store configuration settings for your code, connection strings, public credentials, hostnames, and URLs in your ConfigMap.

©Simplilearn. All rights reserved.

simplilearn

Configuring Applications with Secrets

Given below are the steps to configure an application with secrets (consuming secret in a volume in a pod):

- Create a secret or use an existing one. Multiple pods can reference the same secret.
- Modify your pod definition to add a volume under .spec.volumes[]. Name the volume anything and have a .spec.volumes[].secret.secretName field equal to the name of the secret object.
- Add a .spec.containers[].volumeMounts[]to each container that needs the secret. Specify .spec.containers[].volumeMounts[].readOnly = trueand .spec.containers[].volumeMounts[].mountPathto an unused directory name where you would like the secrets to appear.
- Modify your image and/or command line so that the program looks for files in that directory. Each key in the secret data map becomes the filename under mountPath.

©Simplilearn. All rights reserved.

simplilearn

Configuring Applications

A few best practices for configuring applications using Kubernetes:

- Specify the latest stable API version while defining configurations
- Write configuration files using YAML and not JSON
- Store the configuration files in version control before they are pushed to the cluster
- Group related objects in a single file
- Put the object descriptions in annotations to allow introspection

©Simplilearn. All rights reserved.

simplilearn

Commands and Arguments

©Simplilearn. All rights reserved.

Creating a Pod with echo host name Command

Problem Statement: You are given a project to create a pod with *echo host name* command.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Creating a Pod with sleep 3600 Arguments

Problem Statement: You are given a project to create a pod with *sleep 3600* arguments.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Configure Environmental Variables and Applications

©Simplilearn. All rights reserved.

Creating a Pod in a *sample environment*

Problem Statement: You are given a project to create a pod in an environment labeled *sample environment.*

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Configuring ConfigMaps in Applications

©Simplilearn. All rights reserved.

Creating a ConfigMap from Values with Variables

Problem Statement: You are given a project to create a configmap from the values containing variables.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Creating a ConfigMap from a File

Problem Statement: You are given a project to create a configmap from a file.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Creating a ConfigMap from Env File

Problem Statement: You are given a project to create a configmap from the env file.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Configuring a ConfigMap as a File in the Pod

Problem Statement: You are given a project to demonstrate the configuration of a configmap as a file in the pod.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Configuring a ConfigMap as a Variable in the Pod

Problem Statement: You are given a project to demonstrate the configuration of a configmap as a variable in the pod.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Configuring a ConfigMap as a Volume in the Pod

Problem Statement: You are given a project to demonstrate the configuration of a configmap as a volume in the pod.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Scale Applications

©Simplilearn. All rights reserved.

Scaling Up Application with More Replicas

Problem Statement: You are given a task to scale up the application with more replicas.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Scaling Down Application with Less Replicas

Problem Statement: You are given a task to scale down the application with less replicas.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Scaling Up and Down with Kubectl Command

Problem Statement: You are given a task to scale the application up and down using kubectl command.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Multi-container Pods

©Simplilearn. All rights reserved.

What Is Multi-container Pod?

A multi-container pod is a pod that runs multiple containers that need to work together.

The pod comprises of an application that consists of multiple tightly-coupled containers.

Multiple pods must be used when you scale the application horizontally, that is, one pod for each instance.

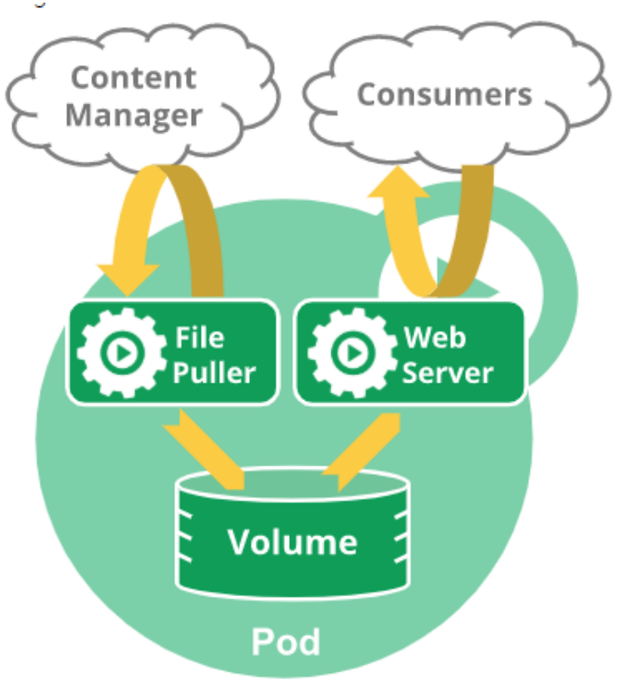
©Simplilearn. All rights reserved.

simplilearn

Multi-container Management by Pods

The pods automatically locate and schedule the containers on the same virtual machine in the cluster.

In pods, the containers can communicate with one another, share resources, and coordinate their way of termination.



©Simplilearn. All rights reserved.

simplilearn

Creating a Multi-container Pod that Accesses the Same Volume

Problem Statement: You are given a project to create a multi-container pod that accesses the same volume.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Multi-container Pods: Design Patterns

©Simplilearn. All rights reserved.

Sidecar Patterns

This pattern utilizes a sidecar container that consists of the main application and a helper container essential to the application.

In this pattern, the parts of applications like the logging code are kept on a separate pod so that an exception thrown by the code does not shut down the main application.

Examples of common sidecar containers:
➢ Logging utilities
➢ Watchers
➢ Sync services
➢ Monitoring agent

©Simplilearn. All rights reserved.

simplilearn

Adapter Patterns

This pattern is used to standardize the application and monitor data for aggregation.

The adapter container is used to adjust the output into the desired format.

Problem:

Say, we have an application that inputs data and produces the output in the format: [Date] – [Host] – [Duration]. The monitoring agent accepts the data only in the given format. The application writes the data in another format.

Solution:

The problem can be resolved using an adapter container to adjust the output into the desired format.

©Simplilearn. All rights reserved.

simplilearn

Ambassador Patterns

It is a pattern that allows containers to connect to different environments.

Ambassador container is a proxy container that connects the main container to a localhost. It then proxies that container to different environment according to the needs.

Problem:

When an application is developed, the developer wants to use the local database. But while testing, the developer wants a different database.

Solution:

The problem can be solved using an ambassador container wherein the application must always connect to the localhost. The ambassador container takes the responsibility of mapping the connection to the right database.

©Simplilearn. All rights reserved.

simplilearn

Init Containers

©Simplilearn. All rights reserved.

Init Containers

Init containers are the containers that run before the app container in a pod.

A pod can have more than one init containers that run before the start of app container.

Some characteristics of init containers:

- They always run on completion
- Each init container must run successfully before the next container starts

©Simplilearn. All rights reserved.

simplilearn

Init Containers

Here are the benefits of init containers:

- Contain custom code for setup
- Can run utilities securely that makes app image less secure
- Offer mechanism to block or delay app container startup
- Can run with a different view of filesystem than the app container in the same pod

©Simplilearn. All rights reserved.

Working of Init Containers

Problem Statement: You are given a project to demonstrate the workflow of init containers.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Self-Healing Applications

©Simplilearn. All rights reserved.

Self-Healing through ReplicaSet

Kubernetes provides self-healing capability that helps the application to continuously deliver the desired output.

ReplicaSet ensures that a specific number of replica pods are running. If one replica fails, ReplicaSet starts the other replica.

All replicas should be a part of the same node because if the node crashes down, the service will be lost.

©Simplilearn. All rights reserved.

simplilearn

Self-Healing through ReplicaSet

Problem Statement: You are given a project to demonstrate the workflow of self-healing through ReplicaSet.

ASSISTED PRACTICE

©Simplilearn. All rights reserved.

Key Takeaways

You are now able to:

- Update deployment rolling using kubectl command and by editing the deployment file
- Check rollback status, pause, and resume rollback
- Create a pod with arguments and commands
- Create a configmap file
- Scale the applications up and down
- Demonstrate the use of initcontainers and self-healing

©Simplilearn. All rights reserved.

Knowledge Check

©Simplilearn. All rights reserved.

Knowledge Check

1

Which of the following is NOT a best practice for configuring applications using Kubernetes?

a. Specify the latest stable API version while defining configurations

b. Write configuration files using YAML and not JSON

c. Group related objects in a single file

d. Write configuration files using JSON and not YAML

©Simplilearn. All rights reserved.

Knowledge Check

1

Which of the following is NOT a best practice for configuring applications using Kubernetes?

a. Specify the latest stable API version while defining configurations

b. Write configuration files using YAML and not JSON

c. Group related objects in a single file

d. Write configuration files using JSON and not YAML

The correct answer is **d**

Specifying the latest stable API version while defining configurations, writing configuration files using YAML and not JSON, and grouping related objects in a single file are the best practices for configuring applications using Kubernetes.

©Simplilearn. All rights reserved.

Knowledge Check 2	_____ are the containers that run before the app container.

a. Azure containers

b. Google containers

c. EC2 containers

d. Init containers

©Simplilearn. All rights reserved.

Knowledge Check 2

_____ are the containers that run before the app container.

a. Azure containers

b. Google containers

c. EC2 containers

d. Init containers

The correct answer is **d**

Init containers run before the app container.

©Simplilearn. All rights reserved.

Knowledge Check

3

Which of the following patterns consists of a container that adjusts the output into the desired format?

a. Sidecar pattern

b. Adapter pattern

c. Ambassador pattern

d. Proxy pattern

©Simplilearn. All rights reserved.

Knowledge Check 3

Which of the following patterns consists of a container that adjusts the output into the desired format?

a. Sidecar pattern

b. Adapter pattern

c. Ambassador pattern

d. Proxy pattern

The correct answer is **b**

Adaptor pattern consists of a container that adjusts the output into the desired format.

©Simplilearn. All rights reserved.

Knowledge Check

4

In which of the following patterns are the parts of applications like the logging code kept on a separate pod?

a. Sidecar pattern

b. Adapter pattern

c. Ambassador pattern

d. All of the above

©Simplilearn. All rights reserved.

Knowledge Check

4

In which of the following patterns are the parts of applications like the logging code kept on a separate pod?

a. Sidecar pattern

b. Adapter pattern

c. Ambassador pattern

d. All of the above

The correct answer is **a**

In sidecar pattern, the parts of applications like the logging code are kept on a separate pod.

©Simplilearn. All rights reserved.

Problem Statement: As a developer, how can you develop an application in which the cluster is capable of increasing the number of nodes as the demand for service response increases and how can you decrease the number of nodes as the requirement decreases?

Objective: Auto scale the application using Kubernetes and manage the available resources that would meet the existing infra requirements/resource management.

LESSON-END PROJECT

©Simplilearn. All rights reserved.