Lesson 3 : Select a Storage Class for an Application

Match applications with storage classes that provide storage services to satisfy application requirements.

Storage Class Selection

- Type of storage
- Enable dynamic storage provisioning (on demand)
- Cluster Admin determine StorageClass
- Default storage class

- Use cases:
  - Development
  - Testing
  - Production
- Storage tier
  - Cost
  - Performance
  - Reliability
  - other functions

Kubernetes and Application Responsibilities

- Application responsible for data integrity, confidentiality and consistency
  - concurrent accesss to shared volume
  - data replicated across region
- Example :
  - PVC of iSCSI LUN configured as RWO (single pod access) but developer forcibly mount to two pods of same host.
  - Problematic depends on applications
- Single-node access (RWO) or shared access (RWX)
  - do not ensure files can be shared safely and reliably

Use Cases for Storage Classes

Storage volume modes A storage class with block volume mode support can increase performance for applications that can use raw block devices. Consider using a storage class with Filesystem volume mode support for applications that share files or that provide file access.

Quality of Service (QoS) levels A Solid State Drive (SSD) provides excellent speed and support for frequently accessed files. Use a lower cost and a slower hard drive (HDD) for files that are accessed less often.

Administrative support tier A production-tier storage class can include volumes that are backed up often. In contrast, a development-tier storage class might include volumes that are not configured with a backup schedule.

Kubernetes matches PVCs with best available PV

- PV that is not bound to any PVC
- PV that at least as large as the requested size in PVC
- PV that matches the other settings specified in PVC
  - Access Mode
  - Storage Class
- If PVC can't satisfied all criterias, enter **pending** state

Create a Storage Class

```
apiVersion: storage.k8s.io/v1 (1)
kind: StorageClass (2)
metadata:
  name: io1-gold-storage (3)
  annotations: (4)
    storageclass.kubernetes.io/is-default-class: 'false'
    description:'Provides RWO and RWOP Filesystem & Block volumes'
    ...
parameters: (5)
  type: io1
  iopsPerGB: "10"
    ...
provisioner: kubernetes.io/aws-ebs (6)
reclaimPolicy: Delete (7)
volumeBindingMode: Immediate (8)
allowVolumeExpansion: true (9)
```

```
[user@host ~]$ oc create -f <storage-class-filename.yaml>
```

1. (required) The current API version.
2. (required) The API object type.
3. (required) The name of the storage class.
4. (optional) Annotations for the storage class.
5. (optional) The required parameters for the specific provisioner; this object differs between plug-ins.
6. (required) The type of provisioner that is associated with this storage class.
7. (optional) The selected reclaim policy for the storage class.
8. (optional) The selected volume binding mode for the storage class.

Verify Storage Classes creation

Use the `oc get storageclass` command to view the storage class options that are available in a cluster.

```
[user@host ~]$ oc get storageclass
```

A regular cluster user can view the attributes of a storage class by using the `describe` command. The following example queries the attributes of the storage class with the name `lvms-vg1`.

```
[user@host ~]$ oc describe storageclass lvms-vg1
IsDefaultClass:        No
Annotations:           description=Provides RWO and RWOP Filesystem & Block volumes
Provisioner:           topolvm.io
Parameters:            csi.storage.k8s.io/fstype=xfs,topolvm.io/device-class=vg1
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
Events:                <none>
```

Storage Class Usage

- The YAML manifest file

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-block-pvc
spec:
  accessModes:
  - RWO
  volumeMode: Block
  storageClassName: <storage-class-name>
  resources:
    requests:
      storage: 10Gi
```

$ oc create –f <manifest file>

$ oc set volume \
deployment/<deployment-name> \
--add –name <volume-name> \
--claim-name my-block-pvc \
--mount-path /var/tmp
--storageClassName: fast-disk

Guided Exercise: Provision Persistent Data Volumes

You should be able to:

- Deploy a MySQL database with persistent storage from a PVC.
- Identify the PV that backs the application.
- Identify the storage provisioner that created the PV

Lab: Deploy Managed and Networked Applications on Kubernetes

You should be able to:

- Deploy a MySQL database from a container image.
- Deploy a web application from a container image.
- Configure environment variables for a deployment.
- Expose the web application for external access.

Quiz 1

Which method for creating container images is recommended by the containers community?

a) Run commands inside a basic OS container, commit the container, and save or export it as a new container image.

b) Run commands from a Dockerfile and push the generated container image to an image registry.

c) Create the container image layers manually from tar files.

d) Run the podman build command to process a container image description in YAML format.

Quiz 1

Which method for creating container images is recommended by the containers community?

a) Run commands inside a basic OS container, commit the container, and save or export it as a new container image.

b) Run commands from a Dockerfile and push the generated container image to an image registry.

c) Create the container image layers manually from tar files.

d) Run the podman build command to process a container image description in YAML format.

Quiz 2

What are two advantages of using the standalone S2I process as an alternative to Dockerfiles? (Choose two.)

a) Requires no additional tools apart from a basic Podman setup.

b) Creates smaller container images, having fewer layers.

c) Reuses high-quality builder images.

d) Automatically updates the child image as the parent image changes (for example, with security fixes).

Quiz 2

What are two advantages of using the standalone S2I process as an alternative to Dockerfiles? (Choose two.)

a) Requires no additional tools apart from a basic Podman setup.

b) Creates smaller container images, having fewer layers.

c) Reuses high-quality builder images.

d) Automatically updates the child image as the parent image changes (for example, with security fixes).

Quiz 3

What are two typical scenarios for creating a Dockerfile to build a child image from an existing image? (Choose two.)

a) Adding new runtime libraries.

b) Setting constraints to a container's access to the host machine's CPU.

c) Adding internal libraries to be shared as a single image layer by multiple container images for different applications.

d) Creates images compatible with OpenShift, unlike container images created from Docker tools.

Quiz 3

What are two typical scenarios for creating a Dockerfile to build a child image from an existing image? (Choose two.)

a) Adding new runtime libraries.

b) Setting constraints to a container's access to the host machine's CPU.

c) Adding internal libraries to be shared as a single image layer by multiple container images for different applications.

a) Creates images compatible with OpenShift, unlike container images created from Docker tools.

Layering Image

- Applicable to other INSTRUCTIONs

```
LABEL version="2.0" \
      description="This is an example container image" \
      creationDate="01-09-2017"
```

```
ENV MYSQL_ROOT_PASSWORD="my_password" \
    MYSQL_DATABASE "my_database"
```

```
EXPOSE 8080, 9090, 9191
```

Chapter Summary

In this chapter, you learned:

Dockerfile contains instructions that specify how to construct a container image.

The Source-to-Image (S2I) process provides an alternative to Dockerfiles. S2I implements a standardized container image build process for common technologies from application source code. This allows developers to focus on application development and not Dockerfile development.

Container images provided by Red Hat Container Catalog or Quay.io are a good starting point for creating custom images for a specific language or technology.

Building an image from a Dockerfile using a three-step process

You should be able to:

- Create network policies to control communication between pods.
- Verify ingress traffic is limited to pods.

You should be able to:

- Modify a secret to add htpasswd entries for new users.
- Configure a new project with role-based access controls and resource quotas.
- Use an OperatorHub operator to deploy a database.
- Create a deployment, service, and route for a web application.
- Troubleshoot an application using events and logs.

In this chapter, you learned:

The OpenShift web console provides a GUI for visualizing and managing OpenShift resources.

Some resources feature a specialized page that makes creating and editing resources more convenient than writing YAML by hand, such as the Edit Key/Value Secret editor, which automatically handles Base64 encoding and decoding.

You can install partner and community operators from the embedded OperatorHub page.

Cluster-wide metrics such as CPU, memory, and storage usage are displayed on the Dashboards page.

Project Details pages display metrics specific to the project, such as the top ten memory consumers by pod and the current resource quota usage.