Guided Exercise: Inspect Kubernetes Resources

Verify the state of an OpenShift cluster by querying its recognized resource types, their schemas, and extracting information from Kubernetes resources that are related to to OpenShift cluster services.

Outcomes

- List and explain the supported API resources for a cluster.
- Identify resources from specific API groups.
- Format command outputs in the YAML and JSON formats.
- Use filters to parse command outputs.
- Use JSONPath and custom columns to extract information from resources.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise. This command ensures that the cluster is accessible and that all resources are available for this exercise. It also creates a `myapp` application in the `cli-resources` project.

```
[student@workstation ~]$ lab start cli-resources
```

Instructions

1. Log in to the OpenShift cluster as the `developer` user with the `developer` password. Select the `cli-resources` project.

   Log in to the OpenShift cluster.

   ```
   [student@workstation ~]$ oc login -u developer -p developer \
     https://api.ocp4.example.com:6443
   Login successful.
   ...output omitted...
   ```

   Set the `cli-resources` project as the active project.

   ```
   [student@workstation ~]$ oc project cli-resources
   ...output omitted...
   ```

2. List the available cluster resource types with the `api-resources` command. Then, use filters to list namespaced and non-namespaced resources.

   List the available resource types with the `api-resources` command.

   ```
   [student@workstation ~]$ oc api-resources
   NAME                     SHORTNAMES   APIVERSION   NAMESPACED   KIND
   bindings                              v1           true         Binding
   componentstatuses        cs           v1           false        ComponentStatus
   configmaps               cm           v1           true         ConfigMap
   endpoints                ep           v1           true         Endpoints
   events                   ev           v1           true         Event
   limitranges              limits       v1           true         LimitRange
   namespaces               ns           v1           false        Namespace
   nodes                    no           v1           false        Node
   persistentvolumeclaims   pvc          v1           true         PersistentVolumeClaim
   persistentvolumes        pv           v1           false        PersistentVolume
   pods                     po           v1           true         Pod
   ...output omitted...
   controllerrevisions                   apps/v1      true         ControllerRevision
   daemonsets               ds           apps/v1      true         DaemonSet
   ...output omitted...
   cronjobs                 cj           batch/v1     true         CronJob
   jobs                                  batch/v1     true         Job
   ...output omitted...
   ```

   The `api-resources` command prints the supported API resources, including resource names, available shortnames, and the API versions.

   You can use the `APIVERSIONS` field to determine which API group provides the resource. The field lists the group followed by the API version of the resource. For example, the `jobs` resource type is provided by the `batch` API group, and `v1` is the API version of the resource.

   Use the `--namespaced` option to limit the output of the `api-resources` command to namespaced resources.

Then, determine the number of available namespaced resources. Use the `-o name` option to list the resource names, and then pipe the output to the `wc -l` command.

```
[student@workstation ~]$ oc api-resources --namespaced
NAME                     SHORTNAMES   APIVERSION   NAMESPACED   KIND
bindings                              v1           true         Binding
configmaps               cm           v1           true         ConfigMap
endpoints                ep           v1           true         Endpoints
events                   ev           v1           true         Event
limitranges              limits       v1           true         LimitRange
persistentvolumeclaims   pvc          v1           true         PersistentVolumeClaim
pods                     po           v1           true         Pod
podtemplates                          v1           true         PodTemplate
replicationcontrollers   rc           v1           true         ReplicationController
resourcequotas           quota        v1           true         ResourceQuota
secrets                               v1           true         Secret
...output omitted...
[student@workstation ~]$ oc api-resources --namespaced -o name | wc -l
122
```

The cluster has 122 namespaced cluster resource types, such as the `pods`, `deployments`, and `services` resources.

Limit the output of the `api-resources` command to non-namespaced resources.

Then, determine the number of available non-namespaced resources. To list the resource names, use the `-o name` option and then pipe the output to the `wc -l` command.

```
[student@workstation ~]$ oc api-resources --namespaced=false
NAME                              SHORTNAMES   APIVERSION                     ...
componentstatuses                 cs           v1                             ...
namespaces                        ns           v1                             ...
nodes                             no           v1                             ...
persistentvolumes                 pv           v1                             ...
mutatingwebhookconfigurations                  admissionregistration.k8s.io/v1  ...
validatingwebhookconfigurations                admissionregistration.k8s.io/v1  ...
customresourcedefinitions         crd,crds     apiextensions.k8s.io/v1        ...
...output omitted...
[student@workstation ~]$ oc api-resources --namespaced=false -o name | wc -l
129
```

The cluster has 129 non-namespaced cluster resource types, such as the `nodes`, `images`, and `project` resources.

3. Identify and explain the available cluster resource types that the core API group provides. Then, describe a resource from the core API group in the `cli-resources` project.

   Filter the output of the `api-resources` command to show only resources from the core API group. Use the `--api-group` option and set `''` as the value.

```
[student@workstation ~]$ oc api-resources --api-group ''
NAME                     SHORTNAMES   APIVERSION   NAMESPACED   KIND
bindings                              v1           true         Binding
componentstatuses        cs           v1           false        ComponentStatus
configmaps               cm           v1           true         ConfigMap
endpoints                ep           v1           true         Endpoints
events                   ev           v1           true         Event
limitranges              limits       v1           true         LimitRange
namespaces               ns           v1           false        Namespace
nodes                    no           v1           false        Node
persistentvolumeclaims   pvc          v1           true         PersistentVolumeClaim
persistentvolumes        pv           v1           false        PersistentVolume
pods                     po           v1           true         Pod
podtemplates                          v1           true         PodTemplate
replicationcontrollers   rc           v1           true         ReplicationController
resourcequotas           quota        v1           true         ResourceQuota
secrets                               v1           true         Secret
serviceaccounts          sa           v1           true         ServiceAccount
services                 svc          v1           true         Service
```

The core API group provides several resource types, such as nodes, events, and pods.

Use the `explain` command to list a description and the available fields for the `pods` resource type.

```
[student@workstation ~]$ oc explain pods
KIND:     Pod
VERSION:  v1

DESCRIPTION:
     Pod is a collection of containers that can run on a host. This resource is
     created by clients and scheduled onto hosts.

FIELDS:
   apiVersion   <string>
     APIVersion defines the versioned schema of this representation of an
     object. Servers should convert recognized schemas to the latest internal
     value, and may reject unrecognized values. More info:
...output omitted...
```

List all pods in the `cli-resources` project.

```
[student@workstation ~]$ oc get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-54fcdcd9d7-2h5vx   1/1     Running   0          4m25s
```

A single pod exists in the `cli-resources` project. The pod name might differ in your output.

Use the `describe` command to view the configuration and events for the pod. Specify the pod name from the previous step.

```
[student@workstation ~]$ oc describe pod myapp-54fcdcd9d7-2h5vx
Name:             myapp-54fcdcd9d7-2h5vx
Namespace:        cli-resources
...output omitted...
Status:           Running
IP:               10.8.0.127
IPs:
  IP:           10.8.0.127
Controlled By:  ReplicaSet/myapp-54fcdcd9d7
Containers:
  myapp:
    Container ID:   cri-o://e0da...669d
    Image:          registry.ocp4.example.com:8443/ubi8/httpd-24:1-215
    Image ID:       registry.ocp4.example.com:8443/ubi8/httpd-24@sha256:91ad...fd83
...output omitted...
Events:
  Type    Reason          Age    From               Message
  ----    ------          ----   ----               -------
  Normal  Scheduled       10m    default-scheduler  Successfully assigned cli-resources/myapp-54fcdcd9d7-2h5vx to master01
...output omitted...
```

Retrieve the details of the pod in a structured format. Use the `get` command and specify the output as the YAML format. Compare the results of the `describe` command versus the `get` command.

```
[student@workstation ~]$ oc get pod myapp-54fcdcd9d7-2h5vx -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
...output omitted...
  labels:
    app: myapp
    pod-template-hash: 54fcdcd9d7
  name: myapp-54fcdcd9d7-2h5vx
  namespace: cli-resources
...output omitted...
spec:
  containers:
  - image: registry.ocp4.example.com:8443/ubi8/httpd-24:1-215
    imagePullPolicy: Always
    name: myapp
    resources: {}
    securityContext:
...output omitted...
```

Using a structured format with the `get` command provides more details about a resource than the `describe` command.

4. Identify and explain the available cluster resource types that the Kubernetes `apps` API group provides. Then, describe a resource from the `apps` API group in the `cli-resources` project.

List the resource types that the `apps` API group provides.

```
[student@workstation ~]$ oc api-resources --api-group apps
NAME                  SHORTNAMES   APIVERSION   NAMESPACED   KIND
controllerrevisions                apps/v1      true         ControllerRevision
daemonsets            ds           apps/v1      true         DaemonSet
deployments           deploy       apps/v1      true         Deployment
replicasets           rs           apps/v1      true         ReplicaSet
statefulsets          sts          apps/v1      true         StatefulSet
```

Use the `explain` command to list a description and fields for the `deployments` resource type.

```
[student@workstation ~]$ oc explain deployments
GROUP:    apps
KIND:     Deployment
VERSION:  apps/v1

DESCRIPTION:
     Deployment enables declarative updates for Pods and ReplicaSets.

FIELDS:
   apiVersion   <string>
     APIVersion defines the versioned schema of this representation of an
     object. Servers should convert recognized schemas to the latest internal
     value, and may reject unrecognized values. More info:
...output omitted...
```

Use the `get` command to identify any `deployment` resources in the `cli-resources` project.

```
[student@workstation ~]$ oc get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myapp   1/1     1            1           25m
```

The `myapp` deployment exists in the `cli-resources` project. Use the `get` command and the `-o wide` option to identify the container name and the container image in the deployment.

```
[student@workstation ~]$ oc get deploy myapp -o wide
NAME  ... CONTAINERS IMAGES                                          SELECTOR
myapp ... myapp      registry.ocp4.example.com:8443/ubi8/httpd-24:1-215  app=myapp
```

The `myapp` deployment uses the `registry.ocp4.example.com:8443/ubi8/httpd-24:1-215` container image for the `myapp` container.

Describe the `myapp` deployment to view more details about the resource.

```
[student@workstation ~]$ oc describe deployment myapp
Name:                   myapp
Namespace:              cli-resources
CreationTimestamp:      Tue, 23 Sep 2025 18:41:39 -0500
Labels:                 my-app
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=myapp
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=myapp
  Containers:
   myapp:
    Image:      registry.ocp4.example.com:8443/ubi8/httpd-24:1-215
    Port:          <none>
    Host Port:     <none>
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   myapp-54fcdcd9d7 (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  30m   deployment-controller  Scaled up replica set myapp-54fcdcd9d7 to 1
```

5. Identify and explain the available cluster resource types that the OpenShift configuration API group provides. Then, describe a resource from the OpenShift configuration API group.

   List the resource types that the OpenShift configuration API group provides.

```
[student@workstation ~]$ oc api-resources --api-group config.openshift.io
NAME              SHORTNAMES APIVERSION           NAMESPACED KIND
apiservers                   config.openshift.io/v1  false   APIServer
authentications              config.openshift.io/v1  false   Authentication
builds                       config.openshift.io/v1  false   Build
clusteroperators      co     config.openshift.io/v1  false   ClusterOperator
clusterversions              config.openshift.io/v1  false   ClusterVersion
consoles                     config.openshift.io/v1  false   Console
dnses                        config.openshift.io/v1  false   DNS
featuregates                 config.openshift.io/v1  false   FeatureGate
imagecontentpolicies         config.openshift.io/v1  false   ImageContentPolicy
imagedigestmirrorsets idms   config.openshift.io/v1  false   ImageDigestMirrorSet
images                       config.openshift.io/v1  false   Image
imagetagmirrorsets    itms   config.openshift.io/v1  false   ImageTagMirrorSet
infrastructures              config.openshift.io/v1  false   Infrastructure
ingresses                    config.openshift.io/v1  false   Ingress
networks                     config.openshift.io/v1  false   Network
nodes                        config.openshift.io/v1  false   Node
oauths                       config.openshift.io/v1  false   OAuth
operatorhubs                 config.openshift.io/v1  false   OperatorHub
projects                     config.openshift.io/v1  false   Project
proxies                      config.openshift.io/v1  false   Proxy
schedulers                   config.openshift.io/v1  false   Scheduler
```

   The `config.openshift.io` API group provides multiple, non-namespaced resource types.

   Use the `explain` command to list a description and fields for the `projects` resource type.

```
[student@workstation ~]$ oc explain projects
GROUP:    project.openshift.io
KIND:     Project
VERSION:  v1

DESCRIPTION:
     Projects are the unit of isolation and collaboration in OpenShift. A
     project has one or more members, a quota on the resources that the project
     may consume, and the security controls on the resources in the project.
     Within a project, members may have different roles - project administrators
     can set membership, editors can create and manage the resources, and
     viewers can see but not access running containers. In a normal cluster
     project administrators are not able to alter their quotas - that is
     restricted to cluster administrators.

     Listing or watching projects will return only projects the user has the
     reader role on.
...output omitted...
```

Describe the `cli-resources` project.

```
[student@workstation ~]$ oc describe project cli-resources
Name:             cli-resources
Created:          10 minutes ago
Labels:           kubernetes.io/metadata.name=cli-resources
                  pod-security.kubernetes.io/audit=restricted
                  pod-security.kubernetes.io/audit-version=latest
                  pod-security.kubernetes.io/warn=restricted
                  pod-security.kubernetes.io/warn-version=latest
Annotations:      openshift.io/description=
                  openshift.io/display-name=
                  openshift.io/requester=system:admin
                  openshift.io/sa.scc.mcs=s0:c26,c25
                  openshift.io/sa.scc.supplemental-groups=1000710000/10000
                  openshift.io/sa.scc.uid-range=1000710000/10000
Display Name:     <none>
Description:      <none>
Status:           Active
Node Selector:    <none>
Quota:            <none>
Resource limits: <none>
```

Retrieve more details of the `cli-resources` project. Use the `get` command, and format the output to use JSON.

```
[student@workstation ~]$ oc get project cli-resources -o json
{
    "apiVersion": "project.openshift.io/v1",
    "kind": "Project",
    "metadata": {
...output omitted....
        "labels": {
            "kubernetes.io/metadata.name": "cli-resources",
            "pod-security.kubernetes.io/audit": "restricted",
            "pod-security.kubernetes.io/audit-version": "latest",
            "pod-security.kubernetes.io/warn": "restricted",
            "pod-security.kubernetes.io/warn-version": "latest"
        },
        "name": "cli-resources",
        "resourceVersion": "705313",
        "uid": "53cbbe45-31ea-4b41-93a9-4ba5c2c4c1f3"
    },
...output omitted...
    "status": {
        "phase": "Active"
    }
}
```

The `get` command provides additional details, such as the `kind` and `apiVersion` attributes, of the project resource.

6. Verify the cluster status by inspecting cluster services. Format command outputs by using filters.

   Retrieve the list of pods for the `Etcd` operator. The `Etcd` operator is available in the `openshift-etcd` namespace. Specify the namespace with the `--namespace` or `-n` option.

```
[student@workstation ~]$ oc get pods -n openshift-etcd
Error from server (Forbidden): pods is forbidden: User "developer" cannot list resource "pods" in API group "" in the name
space "openshift-etcd"
```

The `developer` user cannot access resources in the `openshift-etcd` namespace. Regular cluster users, such as the `developer` user, cannot query resources in the `openshift-` namespaces.

Log in as the `admin` user with the `redhatocp` password. Then, retrieve the list of pods in the `openshift-etcd` namespace.

```
[student@workstation ~]$ oc login -u admin -p redhatocp
Login successful
...output omitted...
[student@workstation ~]$ oc get pods -n openshift-etcd
NAME                   READY   STATUS      RESTARTS   AGE
etcd-master01          5/5     Running     60         124d
installer-1-master01   0/1     Completed   0          124d
installer-2-master01   0/1     Completed   0          124d
```

Retrieve the image of the `etcd-master01` pod in the `openshift-etcd` namespace. Use filters to limit the output to the `.spec.containers` attribute of the pod to get the first element. Compare the outputs of the JSONPath, the `jq` filters, and the Go template format.

```
[student@workstation ~]$ oc get pods etcd-master01 -n openshift-etcd \
  -o=jsonpath='{.spec.containers[0].image}{"\n"}'
quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:24d9...ifca
```

```
[student@workstation ~]$ oc get pods -n openshift-etcd etcd-master01 \
  -o json | jq .spec.containers[0].image
"quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:24d9...ifca"
```

```
[student@workstation ~]$ oc get pods -n openshift-etcd etcd-master01 \
  -o go-template='{{(index .spec.containers 0).image}}'
quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:24d9b9d9d7fadacbc505c849a1e4b390b2f0fcd452ad851b7cce21e8cf
```

Retrieve the `condition` status of the `prometheus-k8s-0` pod in the `openshift-monitoring` namespace. Configure the output to use the YAML format, and then filter the output with the `yq` filter.

```
[student@workstation ~]$ oc get pods -n openshift-monitoring prometheus-k8s-0 \
  -o yaml |  yq '.status.conditions'
- lastProbeTime: null
  lastTransitionTime: "2025-09-23T13:30:26Z"
  status: "True"
  type: PodReadyToStartContainers
- lastProbeTime: null
  lastTransitionTime: "2025-09-22T14:16:09Z"
  status: "True"
  type: Initialized
- lastProbeTime: null
  lastTransitionTime: "2025-09-23T13:31:18Z"
  status: "True"
  type: Ready
- lastProbeTime: null
  lastTransitionTime: "2025-09-23T13:31:18Z"
  status: "True"
  type: ContainersReady
- lastProbeTime: null
  lastTransitionTime: "2025-05-22T11:13:27Z"
  status: "True"
  type: PodScheduled
```

Use the `get` command to retrieve detailed information for the pods in the `openshift-storage` namespace. Use the YAML format and custom columns to filter the output according to the following table:

Column title	Object
PodName	`metadata.name`
ContainerName	`spec.containers[].name`
Phase	`status.phase`
IP	`status.podIP`
Ports	`spec.containers[].ports[].containerPort`

```
[student@workstation ~]$ oc get pods -n openshift-storage \
  -o custom-columns=PodName:".metadata.name",\
ContainerName:"spec.containers[].name",\
Phase:"status.phase",\
IP:"status.podIP",\
Ports:"spec.containers[].ports[].containerPort"
PodName                     ContainerName       Phase    IP          Ports
lvms-operator-7fcd897cb-...  manager             Running  10.8.0.97   9443
vg-manager-z8g5k             vg-manager          Running  10.8.0.101  8081
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish cli-resources
```