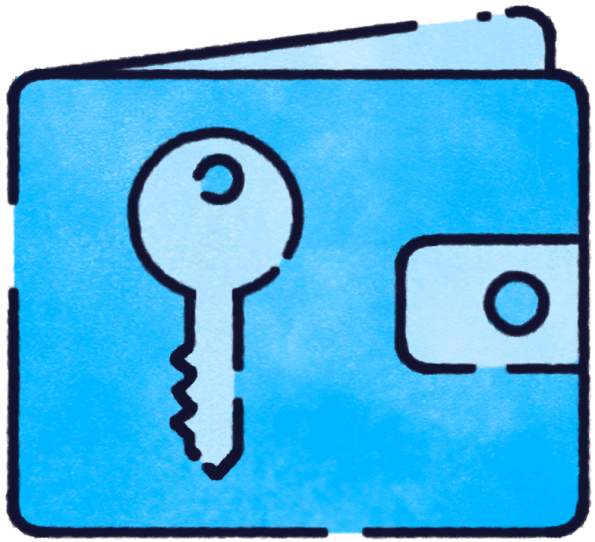
Understand ConfigMaps

Elle Krout
Principal Course Author, Pluralsight

Kubernetes ConfigMaps

ConfigMaps store key-value pairs, letting you decouple application configuration from the container image.

ConfigMaps



Can be updated without needing to rebuild the image

– Restart the pod for changes to populate

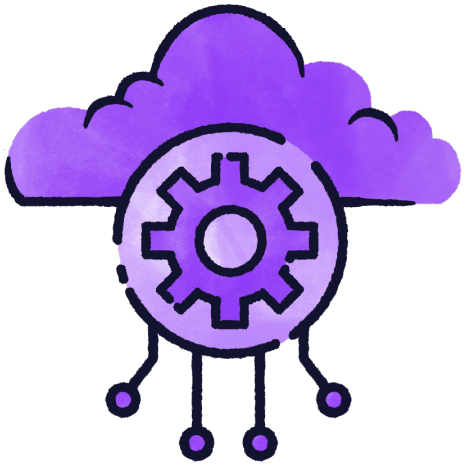
Namespace-scoped

Values can be multiline

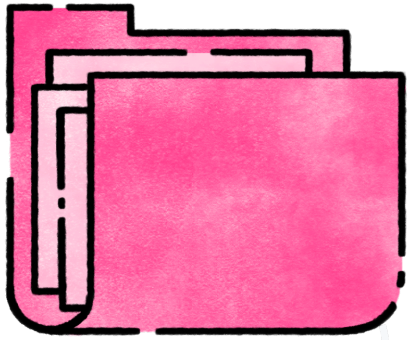
Values must be under 1MB in size

ConfigMaps Can Be Used As...

Arguments



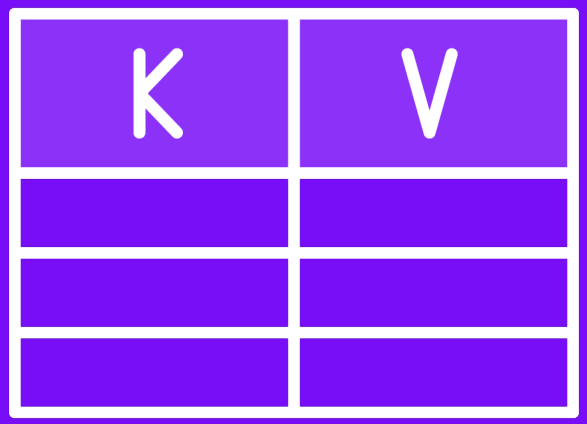
Environmental Variables



Files

ConfigMap Use Cases

- Environment-specific configurations
- Feature flags
- Storing centralized configurations
- System utility or init container configurations
- Application startup parameters



ConfigMaps

Provide a build-in mechanism to manage configuration data in reliable, repeatable ways.

Creating ConfigMaps

ConfigMap Manifest

app-config-map.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  APP_MODE: production
  LOG_LEVEL: debug
binaryData:
  config.bin: |
    U29tZSBiaW5hcnkgY29udGVudCB3aXRoIG5vbi1wcmludGFib
    GUgY2hhcnMh
```

Create ConfigMap via Command Line

```
> kubectl create configmap app-config --from-literal=APP_MODE=production --from-literal=LOG_LEVEL=debug
```

Create ConfigMap via Command Line (Dry Run)

```
> kubectl create configmap app-config --from-literal=APP_MODE=production --from-literal=LOG_LEVEL=debug
```

```
> kubectl create configmap app-config --from-literal=APP_MODE=production –from-literal=LOG_LEVEL=debug --dry-run=client -o yaml > configmap.yaml
```

Create ConfigMap from File via Command Line

```
> kubectl create configmap app-startup --from-file=startup.sh
```

```
data:
  startup.sh: |
    #! /bin/sh
    <your_script_here>
```

```
> kubectl create configmap app-config --from-file=/configdir
```

Create ConfigMap from Env File via Command Line

```
> kubectl create configmap app-config --from-env-file=config.properties
```

Create ConfigMap from Env File via Command Line

```
> kubectl create configmap app-config --from-env-file=config.properties
```

config.properties

```
APP_MODE=production
LOG_LEVEL=debug
```

app-config.yaml

```
...
data:
  APP_MODE: production
  LOG_LEVEL: debug
```

Create ConfigMap from Env File via Command Line

```
> kubectl create configmap app-config --from-env-file=config.properties
```

config.yaml

```
app_mode: productioin
log_level: debug
```

app-config.yaml

```
...
data:
  APP_MODE: production
  LOG_LEVEL: debug
```

View Available ConfigMaps

```
> kubectl get configmaps

NAME                   DATA   AGE
app-config             2      3s
app-startup-scripts    1      25d
kube-root-ca.crt       1      53d
message-hello          1      19d
```

View Available ConfigMaps

```
> kubectl get cm

NAME                   DATA   AGE
app-config             2      3s
app-startup-scripts    1      25d
kube-root-ca.crt       1      53d
message-hello          1      19d
```

View Contents of ConfigMap

```
> kubectl describe configmap app-config
```

```
Name:         app-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
APP_MODE:
----
production

LOG_LEVEL:
----
debug


BinaryData
====
```

Using ConfigMaps

Using ConfigMaps

Pull in whole map as environment variables

Pull in part of the map as environment variables

Use as files as part of a volume

ConfigMap as Environment Variables

app-config-map.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
  - name: myapp
    image: busybox
    command: ["sh", "-c", "env"]
    envFrom:
      - configMapRef:
          name: app-config
```

ConfigMap as Environment Variables (Partial)

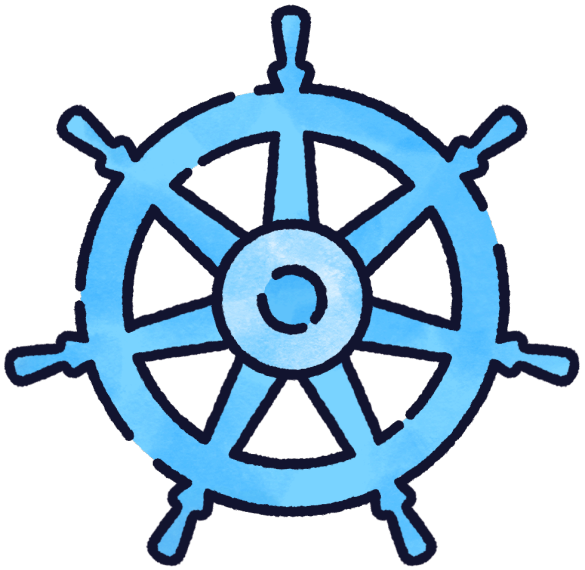
app-config-map.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
  - name: myapp
    image: busybox
    command: ["sh", "-c", "env"]
    env:
      - name: loglevel
        valueFrom:
          configMapKeyRef:
            name: app-config
            key: LOG_LEVEL
```

ConfigMap as Volumes

app-config-map.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
  - name: myapp
    image: busybox
    command: ["sh", "-c", “env"]
    volumeMounts:
      - name: config-volume
        mountPath: "/config"
        readOnly: true
    volumes:
      - name: config-volume
        configMap:
          name: app-config
```



When Using ConfigMaps

Choose the best option for your application, whether it's environmental variables or volumes

Demo: Working with Kubernetes ConfigMaps

Exam Scenario

You need to create a series of variables to be used across three pod definitions, `demo-pod.yaml`, `dev-demo-pod.yaml`, and `volume-pod.yaml`. The key-value pairs that need to saved are:

```
app_name=demo-app
log_level=debug
feature_flag=enabled
timeout=7
```

Create a ConfigMap that stores this data

Exam Scenario

Update the `demo-pod` manifest to pull in the ConfigMap as-is. The `dev-demo-pod` also uses this data, but each value needs to be pulled in individually, because the pod expects all environmental variables to have keys in ALL CAPS. Finally, update the `volume-pod` to pull in the ConfigMap as a volume

Deploy the pods. Confirm that the deploys were successful by viewing the `demo` pod logs and accessing the volume pod directly