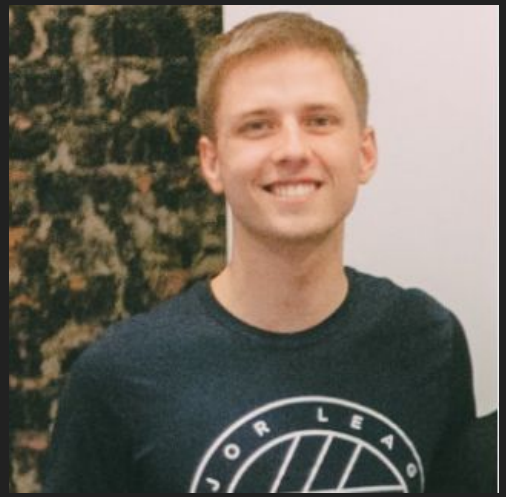
Introduction to Kubernetes Part 3

GHW Cloud Week - March 2025

Welcome!

I’m Stephen, your MLH Coach.



- Been hacking for almost ten years now!
- Super excited for GHW Cloud Week

Don’t forget to check in…
https://events.mlh.io/events/12345

Welcome Back!

Poll: How comfortable are you with Kubernetes so far?

1. Not at all
2. Getting there…
3. I’m cruisin’!

Part 1 Recap

- Learned the core components of Kubernetes: Pods, Nodes, Deployments, ReplicaSets, and Services.
- Practiced creating YAML files to deploy applications, scaled them, and performed rolling updates.

Question for you: What was one thing you found most surprising about creating your first Deployment?

Part 2 Recap

We advanced into networking, persistent storage, and security.

- Set up Services and Ingress for routing
- Configured Persistent Volume Claims (PVCs) for data persistence
- Learned about RBAC and Network Policies.

Advanced Storage & Configurations

- Today, we’re going deeper into how Kubernetes handles storage and configurations.
- We’ll discuss ConfigMaps, Secrets, and volume management.

ConfigMaps

- A ConfigMap is a way to decouple configuration data from your application code.
- Instead of hardcoding settings, you can store key-value pairs here and mount them into your Pods.
- For example, if you have an application that needs to know the URL of an API endpoint, you can store it in a ConfigMap.

Secrets

- Secrets are similar to ConfigMaps but are used for sensitive data such as passwords, tokens, or keys.
- They’re stored in an encoded form and should be handled with care.

Volume Management

- Volumes in Kubernetes provide a way to persist data beyond the life of a Pod.
- Persistent Volumes (PVs) and Persistent Volume Claims (PVCs) work together to give your application stable storage that is independent of Pod lifecycles.

Quiz

What might be one reason to use a ConfigMap instead of embedding configuration directly into your application?

Quiz

Now, what about Secrets? Why are they important?

Creating a ConfigMap

Let’s create a ConfigMap that stores an HTML file. This file will be served by an Nginx container later.

Create the YAML File

Open your code editor and create a file named *configmap.yaml*

Apply the ConfigMap

In your terminal, run:

kubectl apply -f configmap.yaml

This command creates a ConfigMap named nginx-config.

Now, type:

kubectl get configmaps nginx-config -o yaml

Quiz: What is the main purpose of using a ConfigMap?

Creating a Secret

Next, we'll create a Secret to store sensitive data. In this example, we're going to store a simple API key.

Create the Secret YAML File

Create a file called *secret.yaml*

Apply the Secret

Run:

kubectl apply -f secret.yaml

Then verify with:

kubectl get secret api-key-secret -o yaml

Quiz

Why do we store sensitive data in a Secret instead of a ConfigMap?

Creating a Persistent Volume Claim (PVC)

Create the PVC YAML File

Create a file called *pvc.yaml*

Apply the PVC

Run:

kubectl apply -f pvc.yaml

Then verify with:

kubectl get pvc nginx-pvc

Ensure that its status shows “Bound.”

Deploying an Nginx Pod with Persistent Storage

Now, we’ll update our Nginx Deployment to use this PVC. Open your existing nginx-deployment.yaml file and add a volume mount.

Update the Deployment YAML File

Within the container spec, add the following under volumeMounts.

```
volumeMounts:
  - name: config-volume
    mountPath: /usr/share/nginx/html
```

Update the Deployment YAML File

Then, under the Pod spec (at the same level as containers), add:

```
volumes:
  - name: config-volume
    persistentVolumeClaim:
      claimName: nginx-pvc
```

Apply the Updated Deployment

Run:

kubectl apply -f nginx-deployment.yaml

This will update your Deployment so that the Nginx Pods mount the persistent storage.

Now, run:

kubectl describe pod <pod-name>

(replacing <pod-name> with one of your Pod names)

Quiz

What is the primary role of a Persistent Volume Claim?

Security & Access Control

Let’s now test our knowledge on Kubernetes security and access control. We’re going to run a short interactive quiz focusing on RBAC, Network Policies, and best practices.

Quiz Questions

What does RBAC stand for?

Quiz Questions

Which command is used to create a Role in Kubernetes?

Quiz Questions

What is the main purpose of a Network Policy?

Quiz Questions

Why would you use Secrets instead of ConfigMaps for sensitive data?

Configuring a Secure Cluster

Creating a Role and RoleBinding

Create a Role YAML File

- Create a file called *role.yaml*

This Role allows the user to read Pods in the default namespace.

Create a RoleBinding YAML File

Next, create a file called *rolebinding.yaml*

Apply these files:

kubectl apply -f role.yaml

kubectl apply -f rolebinding.yaml

Then run:

kubectl get rolebinding read-pods-binding -o yaml

Configuring Network Policies for Security

Create a Network Policy YAML File

Create a file named *network-policy.yaml*

This policy denies all incoming and outgoing traffic by default.

Apply the Network Policy

Run:

kubectl apply -f network-policy.yaml

Then check:

kubectl get networkpolicy

Quiz

What is the purpose of applying a default-deny Network Policy?

Real-World Troubleshooting Session

Next, we’re going to look at some common issues you might face in a production cluster and how to troubleshoot them.

Common Issues and Live Debugging

Sometimes a Pod’s readiness probe fails. We can check our pods with:

kubectl get pods

kubectl describe pod <pod-name>

Look for events related to the readiness probe.

Summary Quiz

What command do you use to create an autoscaler for a deployment?

Summary Quiz

Which object in Kubernetes stores configuration data without embedding it in the container image?

Summary Quiz

Name one method to secure sensitive information in Kubernetes.

Summary Quiz

What command would you use to check detailed information about a Pod?

Summary Quiz

What is the main role of a Network Policy?

Thank you!

Next stream coming up…

Stay tuned for Abdullah's stream, next:

Building a Cloud-Native CI/CD Pipeline

18:00 - 20:00 ET

https://events.mlh.io/events/12346