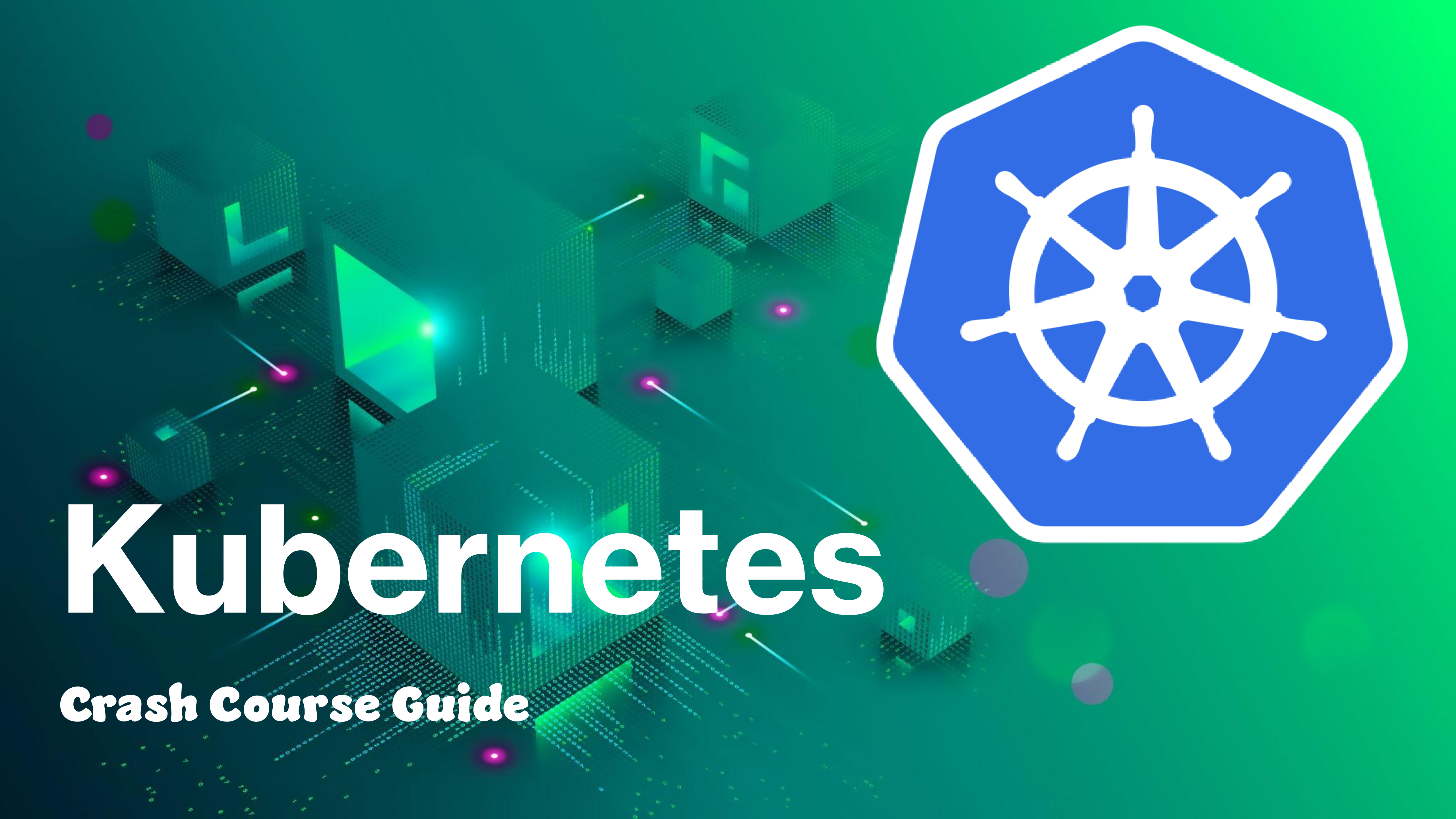
Kubernetes

Crash Course Guide



Why you should learn Kubernetes?

Kubernetes is a powerful platform for deploying, scaling, and managing containerized applications.

In this crash course guide, I've covered some of the key concepts and features of Kubernetes, including pods, deployments, services, operators, networking, storage, and security.
By mastering these concepts, you'll be well on your way to becoming a Kubernetes expert and taking full advantage of this powerful platform for modern application development.

Overview

- This “**Kubernetes - Crash Course Guide**” is designed to introduce you to the basics of Kubernetes and give you hands-on experience deploying and managing applications on a Kubernetes cluster.

Prerequisites

Before starting this crash course, you should have:

- Basic knowledge of Linux command line
- Familiarity with containerization using Docker
- Access to a Kubernetes cluster or a local Kubernetes environment

Modules

- **Module 1: Introduction to Kubernetes**
  - In this module, you will learn the basics of Kubernetes, including its architecture, components, and key concepts.
- **Module 2: Deploying Applications on Kubernetes**
  - In this module, you will learn how to deploy and manage applications on a Kubernetes cluster using Kubernetes manifests and the kubectl command-line tool.
- **Module 3: Scaling and Load Balancing**
  - In this module, you will learn how to scale and load balance your applications on a Kubernetes cluster using Kubernetes deployments and services.
- **Module 4: Monitoring and Logging**
  - In this module, you will learn how to monitor and log your applications running on a Kubernetes cluster using Kubernetes metrics and logging tools.

Modules Quick Overview

Module 1: Introduction to Kubernetes

In this module, you will learn the basics of Kubernetes, including its architecture, components, and key concepts. Some of the topics covered in this module include:

- Kubernetes architecture: Learn about the components that make up a Kubernetes cluster, including the control plane, worker nodes, and etcd database.
- Kubernetes objects: Learn about the various Kubernetes objects you can use to deploy and manage applications on a cluster, including pods, services, deployments, and more.
- Kubernetes networking: Learn how Kubernetes networking works and how to configure networking for your applications.
- Kubernetes storage: Learn how Kubernetes provides persistent storage for your applications.

Module 2: Deploying Applications on Kubernetes

In this module, you will learn how to deploy and manage applications on a Kubernetes cluster using Kubernetes manifests and the kubectl command-line tool. Some of the topics covered in this module include:

- Kubernetes manifests: Learn how to write Kubernetes manifests to describe the desired state of your applications.
- kubectl: Learn how to use the kubectl command-line tool to interact with a Kubernetes cluster, including deploying and managing applications.
- Deployments: Learn how to use Kubernetes deployments to manage application replicas and rolling updates.
- Secrets: Learn how to use Kubernetes secrets to securely store and manage sensitive data.

Module 3: Scaling and Load Balancing

In this module, you will learn how to scale and load balance your applications on a Kubernetes cluster using Kubernetes deployments and services. Some of the topics covered in this module include:

- Scaling: Learn how to scale your applications horizontally and vertically using Kubernetes.
- Load balancing: Learn how to use Kubernetes services to load balance traffic to your applications.
- Service discovery: Learn how Kubernetes provides service discovery for your applications.

Module 4: Monitoring and Logging

In this module, you will learn how to monitor and log your applications running on a Kubernetes cluster using Kubernetes metrics and logging tools. Some of the topics covered in this module include:

- Kubernetes metrics: Learn how to use Kubernetes metrics to monitor the health and performance of your applications.
- Kubernetes logging: Learn how to use Kubernetes logging tools to collect and analyze logs from your applications.
- Monitoring and logging best practices: Learn about best practices for monitoring and logging your applications running on a Kubernetes cluster.

Modules Explanation in Detail

Module 1
Introduction to Kubernetes

Kubernetes is an open-source platform for managing containerized applications. In this module, we'll cover some of the key concepts and features of Kubernetes, and how it can help you manage your applications.

Kubernetes architecture

Kubernetes is composed of several components that work together to create a highly available and scalable platform for deploying and managing containerized applications. These components include:

- The control plane: This is the brain of the Kubernetes cluster and is responsible for managing the state of the cluster. It includes several components, such as the API server, etcd, scheduler, and controller manager.
- Worker nodes: These are the machines that run your applications. Each worker node runs a container runtime, such as Docker or containerd, and a kubelet agent that communicates with the control plane to ensure that the desired state of the cluster is being maintained.
- etcd database: This is a distributed key-value store that stores the configuration and state of the Kubernetes cluster.

Kubernetes objects

Kubernetes uses several objects to define and manage the state of your applications. These objects include:

- Pods: A pod is the smallest unit of deployment in Kubernetes. It represents a single instance of your application and can contain one or more containers.
- Deployments: A deployment manages a set of replicas of your application. It provides declarative updates for your application, such as rolling updates and rollbacks.
- Services: A service is an abstraction layer that provides a stable IP address and DNS name for a set of pods. It allows other applications to access your application without needing to know the exact IP addresses of the pods.
- ConfigMaps: A ConfigMap is a way to store configuration data as key-value pairs. It allows you to separate your application configuration from your application code.
- Secrets: A Secret is a way to store sensitive information, such as passwords or API keys, as key-value pairs. It provides a secure way to manage your application secrets.

Kubernetes networking

Kubernetes provides a powerful networking model that allows your applications to communicate with each other and with the outside world. Some of the key networking concepts in Kubernetes include:

- Pods: Each pod gets its own IP address and can communicate with other pods in the same cluster using that IP address.
- Services: A service provides a stable IP address and DNS name for a set of pods. It allows other applications to access your application without needing to know the exact IP addresses of the pods.
- Ingress: An Ingress is a way to expose HTTP and HTTPS routes from outside the cluster to services within the cluster.
- Network policies: Network policies allow you to control traffic between pods in a more granular way. You can define rules that allow or deny traffic based on source and destination IP addresses, ports, and protocols.

Kubernetes storage

Kubernetes provides several ways to manage persistent storage for your applications. Some of the key storage concepts in Kubernetes include:

- Volumes: A volume is a way to store data in a pod that persists beyond the life of the pod. Kubernetes supports several types of volumes, such as local, network-attached, and cloud storage volumes.
- Persistent Volumes: A Persistent Volume is a way to abstract the underlying storage technology from the application. It allows you to manage storage resources separately from the application and provides a way to reuse storage resources across multiple applications.
- Persistent Volume Claims: A Persistent Volume Claim is a way to request storage resources for your application. It allows you to specify the desired storage capacity and other storage-related parameters.

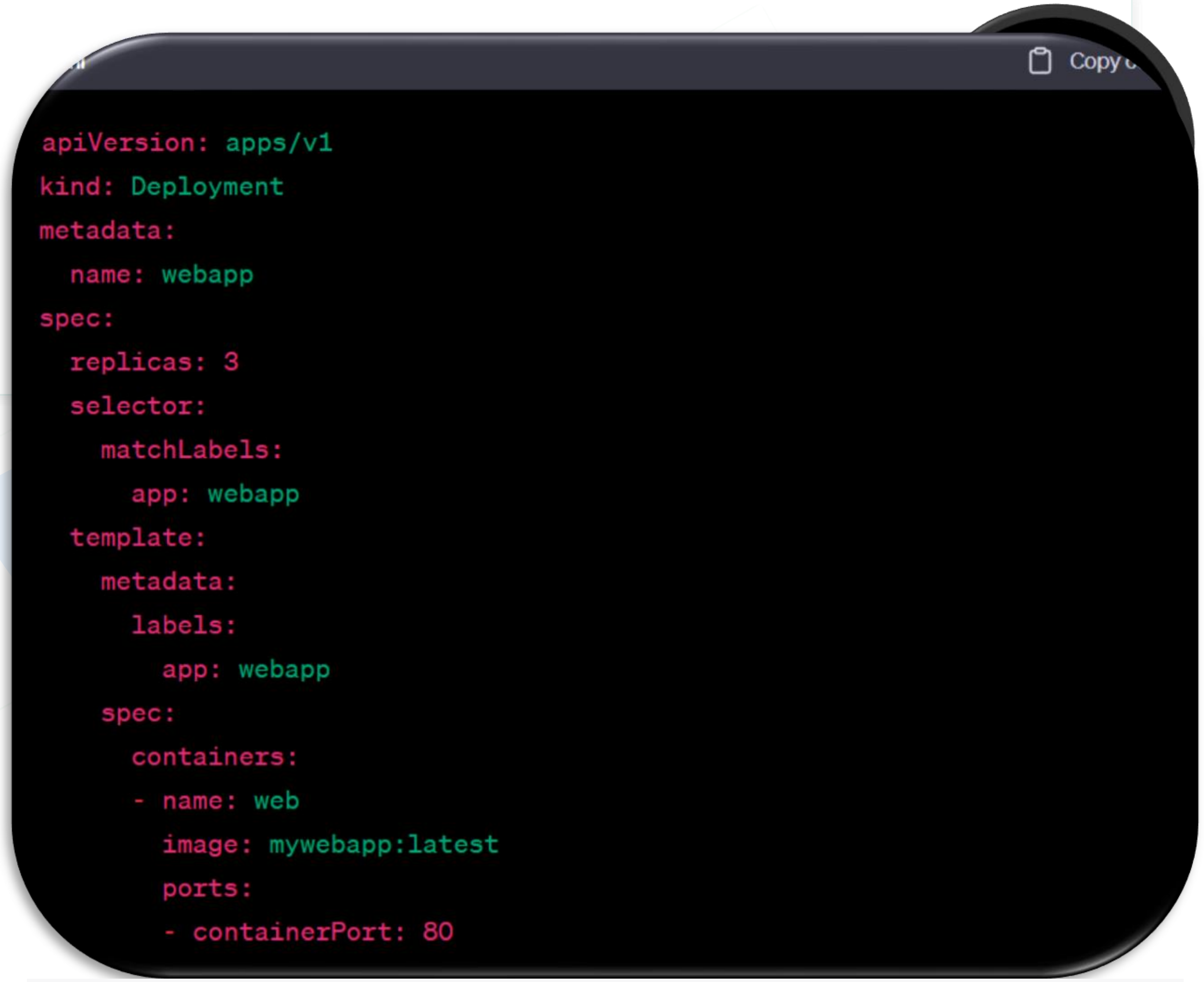
Module 2
Deploying Applications on Kubernetes

In this module, we'll cover the key concepts and tools for deploying applications on Kubernetes. Deploying applications on Kubernetes can seem complex at first, but once you understand the core concepts, it becomes much easier to manage your applications.

Kubernetes manifests

- A Kubernetes manifest is a YAML file that describes the desired state of your application. It contains all of the necessary information to create and manage your application on a Kubernetes cluster.

Here is an example of a simple Kubernetes manifest for a web application:



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: web
        image: mywebapp:latest
        ports:
        - containerPort: 80
```

In this example, we are defining a Deployment object that manages a set of replicas of our web application. We are requesting three replicas, and we have defined a selector to match the labels on the pods that belong to this deployment. The template section defines the pod template that will be used to create the replicas. It specifies that we want to use a container image called ***mywebapp:latest****, which exposes port 80.*

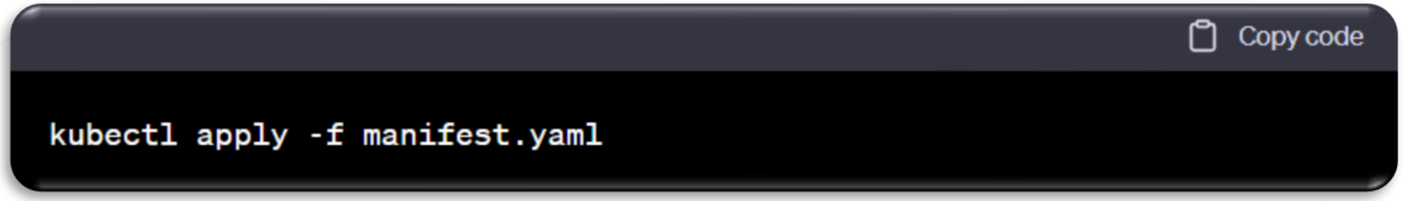
Deploying an application

To deploy your application on a Kubernetes cluster, you need to create a Kubernetes manifest file that describes the desired state of your application, and then use the ‘**kubectl**’ command-line tool to apply the manifest to the cluster. Here are the steps you need to follow:

1. Create a Kubernetes manifest file for your application. You can use the example manifest above as a starting point and modify it to suit your needs.

2. Save the manifest file to your local machine.

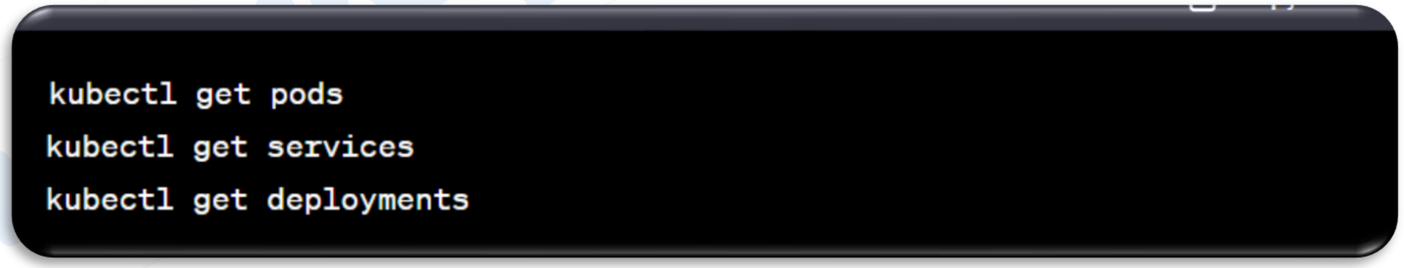
3. Use the ‘***kubectl apply***’ command to apply the manifest to the cluster:



```
kubectl apply -f manifest.yaml
```

Deploying an application (Continue)

4. Kubernetes will create the necessary resources to run your application, including pods, services, and deployments. You can use the kubectl get command to check the status of your resources:

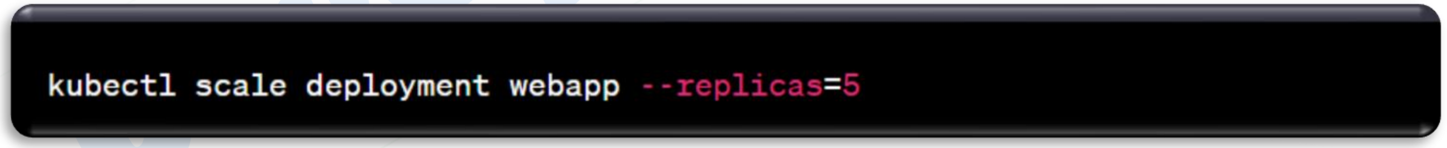


```
kubectl get pods
kubectl get services
kubectl get deployments
```

Scaling an application

Kubernetes makes it easy to scale your application up or down depending on demand. To scale your application, you can use the **‘kubectl scale’** command.

Here's an example:



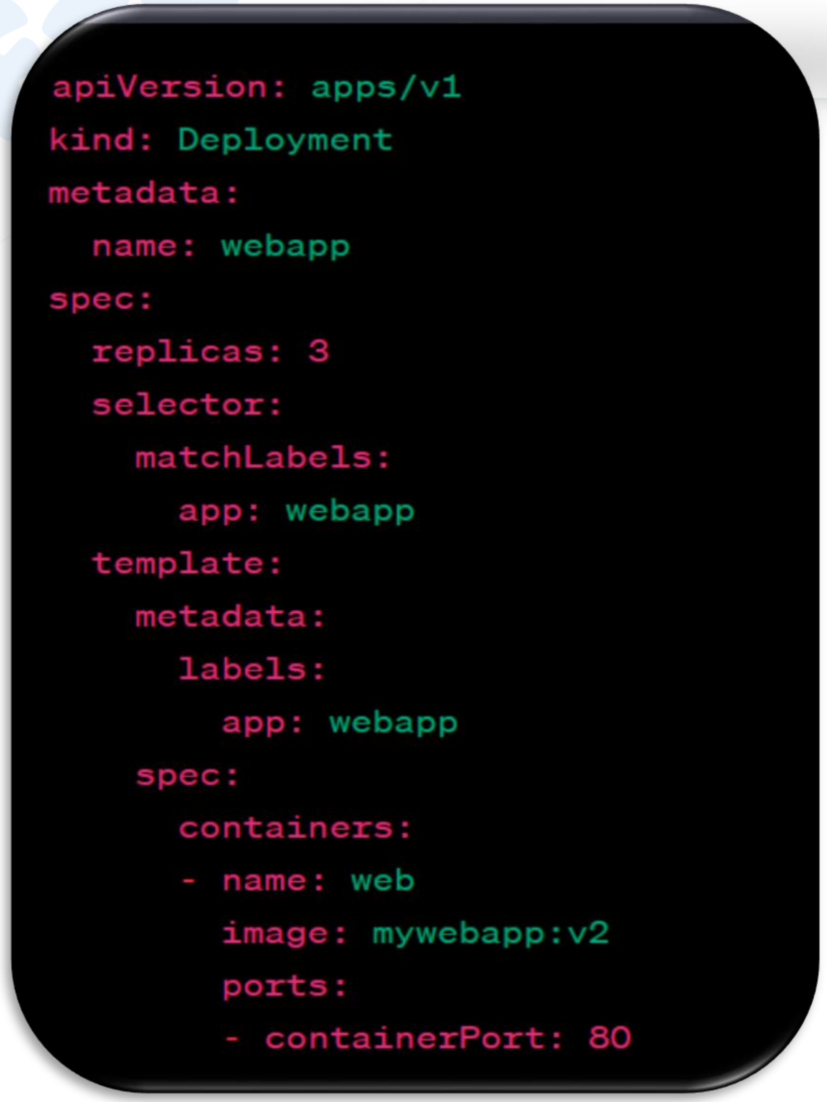
```
kubectl scale deployment webapp --replicas=5
```

In this example, we are scaling up our webapp deployment to five replicas.

Updating an application

Kubernetes provides a powerful way to update your application without downtime using rolling updates. To update your application, you can modify the Kubernetes manifest file to specify the new desired state of your application, and then use the ‘**kubectl apply**’ command to apply the changes to the cluster. Kubernetes will automatically create new replicas with the updated configuration and gradually replace the old replicas.

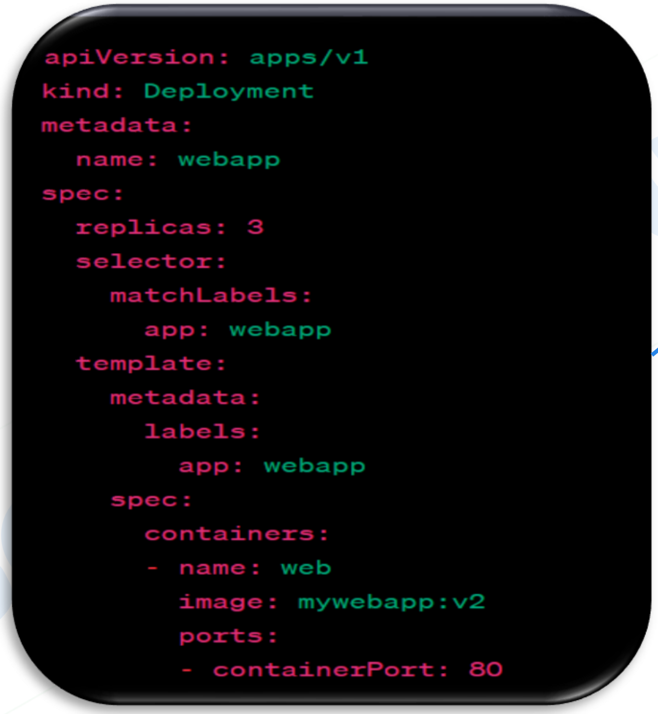
Here's an example:



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: web
        image: mywebapp:v2
        ports:
        - containerPort: 80
```

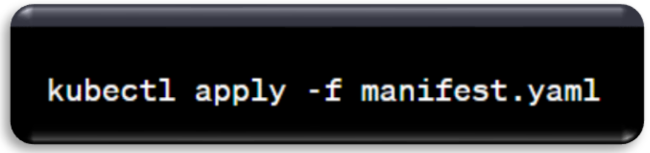
Updating an application (Continue)

Here's an example:



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: web
        image: mywebapp:v2
        ports:
        - containerPort: 80
```

In this example, we are updating our webapp deployment to use a new container image called '**mywebapp:v2**'. We can apply this change to the cluster using the '**kubectl apply**' command:



```
kubectl apply -f manifest.yaml
```

Kubernetes will automatically update the deployment without downtime.

Module 3
Advanced Kubernetes Concepts

In this module, we'll cover some of the more advanced concepts in Kubernetes, including networking, storage, and security.

Kubernetes Operators

Kubernetes Operators are a way to automate the management of complex applications on Kubernetes.

- An Operator is a custom controller that extends the Kubernetes API to create, configure, and manage instances of complex applications.
- Operators can be used to manage stateful applications, databases, message brokers, and other complex systems.
- Operators provide a way to automate the day-to-day tasks associated with managing these applications, such as backups, upgrades, and scaling.

For example, a database operator can automate the creation, scaling, and backup of a database cluster, using the Kubernetes API to manage the underlying resources. This allows developers to focus on writing code, rather than managing infrastructure.

Kubernetes Networking

Kubernetes networking can be complex, especially in multi-node clusters.

In Kubernetes, each pod has its own IP address, and pods can communicate with each other directly, even across different nodes in the cluster. To enable this communication, Kubernetes uses a networking model called "Container Networking Interface" (CNI).

CNI plugins are responsible for configuring networking for pods, and there are many different plugins available, such as ***Calico***, ***Flannel***, and ***Weave Net***. Each plugin has its own strengths and weaknesses, and the choice of plugin will depend on the specific requirements of your application.

In addition to pod-to-pod networking, Kubernetes also provides a way to expose services to the outside world using "***Services***". A service is an abstraction that defines a logical set of pods and a policy for accessing them. Services can be exposed using different types of "Service" objects, such as ***ClusterIP, NodePort,*** and ***LoadBalancer.***

Kubernetes Storage

Kubernetes provides a way to manage storage for your applications using ***"Persistent Volumes" (PVs)*** and ***"Persistent Volume Claims" (PVCs).***

PVs are resources in the cluster that provide storage for your applications, such as a network file system or a block storage device.

PVCs are requests for storage by a pod, and they can specify the capacity and access mode of the desired storage.

Kubernetes also provides a way to manage stateful applications using "Stateful Sets". Stateful Sets are similar to Deployments, but they provide guarantees about the ordering and uniqueness of pods. This is important for stateful applications, such as databases, that require stable network identities and persistent storage.

Kubernetes Security

Kubernetes provides several features to enhance the security of your applications. One important feature is "Role-Based Access Control" (RBAC), which allows you to define fine-grained access policies for users and applications in the cluster. RBAC can be used to control access to resources such as pods, services, and deployments.

Kubernetes also provides a way to encrypt network traffic within the cluster using "Transport Layer Security" (TLS). This is important for securing communication between pods and services within the cluster.

In addition, Kubernetes provides a way to run containers with restricted permissions using "Security Contexts". Security Contexts allow you to specify the user and group IDs of the container process, as well as other security settings such as AppArmor and Seccomp profiles.

Module 4
Monitoring & Logging

Kubernetes provides built-in support for monitoring and logging your applications. In this module, we'll cover some of the key concepts and tools for monitoring and logging in Kubernetes.

Monitoring

Monitoring is an essential part of managing any application. In Kubernetes, there are several tools available for monitoring your applications, such as:

- **Prometheus**: a popular open-source monitoring system that can collect metrics from your applications and infrastructure. Prometheus uses a powerful query language called PromQL to query and analyze metrics.
- **Grafana**: a visualization and dashboarding tool that can be used with Prometheus to create custom dashboards for monitoring your applications.
- **Kubernetes Dashboard**: a web-based dashboard for monitoring the health of your Kubernetes cluster and its components.
- **Container Runtime Interface (CRI) tools**: tools provided by your container runtime (such as Docker or containerd) for monitoring container-level metrics such as CPU and memory usage.

Using these tools, you can collect and analyze metrics for your applications & infrastructure and create custom dashboards to monitor their health.

Logging

In addition to monitoring, logging is also important for understanding the behavior of your applications. Kubernetes provides several tools for logging, such as:

- **Fluentd**: a popular open-source log collector that can collect logs from your applications and infrastructure and send them to various outputs, such as Elasticsearch or Google Cloud Storage.
- **Elasticsearch**: a powerful search and analytics engine that can be used with Fluentd to store and analyze logs.
- **Kibana**: a visualization and dashboarding tool that can be used with Elasticsearch to create custom dashboards for analyzing your logs.
- **Container logs**: Kubernetes automatically collects logs from your containers and makes them available through the Kubernetes API.

By using these tools, you can collect, store, and analyze logs from your applications and infrastructure, and gain valuable insights into their behavior.

Real World Examples (Monitoring & Logging)

Here are some real-world examples of how monitoring and logging can be used in Kubernetes:

- *An e-commerce website is using Kubernetes to deploy their application. They use Prometheus and Grafana to monitor the health of their application, including metrics such as response time and error rate. They also use Fluentd and Elasticsearch to collect and analyze logs from their application, and use Kibana to create custom dashboards for analyzing their logs.*
- *A financial services company is using Kubernetes to deploy their trading platform. They use CRI tools provided by their container runtime to monitor container-level metrics such as CPU and memory usage. They also use the Kubernetes Dashboard to monitor the health of their Kubernetes cluster and its components.*
- *A media streaming company is using Kubernetes to deploy their video transcoding application. They use Prometheus and Grafana to monitor the performance of their application, including metrics such as transcoding rate and video quality. They also use Fluentd and Elasticsearch to collect and analyze logs from their application, and use Kibana to create custom dashboards for analyzing their logs.*

By using monitoring and logging tools in Kubernetes, these companies can gain valuable insights into the behavior of their applications, and make informed decisions about how to optimize their performance and troubleshoot issues.

Kubernetes is a powerful platform for deploying, scaling, and managing containerized applications. In this crash course, we've covered some of the key concepts and features of Kubernetes, including pods, deployments, services, operators, networking, storage, and security. By mastering these concepts, you'll be well on your way to becoming a Kubernetes expert and taking full advantage of this powerful platform for modern application development.

THANK YOU FOR READING!

Join Now

For More FREE DevOps Resources

Learn DevOps for FREE