Kubernetes

Kubernetes (k8s) is an open-source platform designed to automate deploying, scaling, and operating application containers. It groups containers that make up an application into logical units for easy management and discovery. Here are the basics of Kubernetes:

Key Concepts

1. **Cluster**: A set of machines (nodes) running containerized applications managed by Kubernetes.
2. **Node**: A single machine (physical or virtual) in the Kubernetes cluster, which can be a master or worker node.
3. **Pod**: The smallest and simplest Kubernetes object. A pod represents a single instance of a running process in a cluster and can contain one or more containers.
4. **Service**: An abstraction that defines a logical set of pods and a policy to access them, typically via DNS.
5. **Namespace**: A way to divide cluster resources between multiple users via namespaces.
6. **Deployment**: A higher-level concept that manages a replicated application, defined by a Pod template. Deployments ensure that a specified number of pod replicas are running at any given time.
7. **ReplicaSet**: Ensures that a specified number of pod replicas are running at any given time. Deployments use ReplicaSets to manage pod lifecycles.
8. **ConfigMap**: Allows you to decouple configuration artifacts from image content to keep containerized applications portable.
9. **Secret**: Stores sensitive data, such as passwords, OAuth tokens, and ssh keys. Secrets can be mounted as files or environment variables in a pod.
10. **Ingress**: Manages external access to services, typically HTTP. Ingress can provide load balancing, SSL termination, and name-based virtual hosting.
11. **Volume**: Provides a way to persist data generated by and used by containers.

Basic Commands

kubectl: The command-line tool for interacting with the Kubernetes API.

```
# Get cluster information
kubectl cluster-info

# Get nodes in the cluster
```

```
kubectl get nodes

# Create a new pod from a YAML file
kubectl apply -f pod-definition.yaml

# Get the status of pods
kubectl get pods

# Describe a specific pod
kubectl describe pod <pod-name>

# Delete a pod
kubectl delete pod <pod-name>
```

- **Example: Deploying an Application**

Create a Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp-container
        image: myapp-image:latest
        ports:
        - containerPort: 80
```

Apply the deployment:

```
kubectl apply -f deployment.yaml
```

1. **Create a Service**

```
apiVersion: v1

kind: Service

metadata:

  name: myapp-service

spec:

  selector:

    app: myapp

  ports:

    - protocol: TCP

      port: 80

      targetPort: 80

  type: LoadBalancer
```

Apply the service:

```
kubectl apply -f service.yaml
```

2. Monitoring and Logging

- **kubectl logs**: View logs of a pod.
  ```
  kubectl logs <pod-name>
  ```
- **kubectl top**: Display resource (CPU/memory) usage.
  ```
  kubectl top pod
  ```

Scaling Applications

kubectl scale: Scale the number of pod replicas.

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

Rolling Updates and Rollbacks

kubectl rollout: Manage the rollout of a deployment.

```
# Check the status of a rollout
kubectl rollout status deployment/myapp-deployment

# Rollback to a previous deployment
kubectl rollout undo deployment/myapp-deployment
```

- **Helm**

Helm is a package manager for Kubernetes, allowing you to define, install, and upgrade even the most complex Kubernetes applications.

Install Helm

```
curl
https://raw.githubusercontent.com/helm/helm/main/scripts/get-hel
m-3 | bash
```

- **Using Helm Charts**
  ```
  # Add a Helm repository
  ```

```
helm repo add stable https://charts.helm.sh/stable

# Update your Helm repositories
helm repo update

# Install a chart
helm install myapp stable/mychart

# List installed charts
helm list

# Uninstall a chart
helm uninstall myapp
```

These basics should give you a solid foundation for working with Kubernetes and deploying containerized applications efficiently.

Setup Miniqube

Setting up Minikube is a straightforward way to run a local Kubernetes cluster for development and testing. Here's how you can set it up on various platforms:

Prerequisites

- **Virtualization**: Minikube requires a hypervisor to create virtual machines. Common options include VirtualBox, Hyper-V, VMware, and KVM. Ensure you have one installed.

kubectl: The Kubernetes command-line tool, `kubectl`, must be installed. You can install it via the following command:

```
# On macOS
brew install kubectl

# On Linux
curl -LO
"https://storage.googleapis.com/kubernetes-release/release/$(curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/
bin/linux/amd64/kubectl"
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl

# On Windows
choco install kubernetes-cli
```

- **Installing Minikube**

macOS

1. **Install Minikube via Homebrew:**
   `brew install minikube`
2. **Start Minikube:**
   `minikube start`

Linux

Download Minikube:

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

1. **Start Minikube:**
   `minikube start`

Windows

1. **Install Minikube via Chocolatey:**
   `choco install minikube`
2. **Start Minikube:**
   `minikube start`

Using Minikube

Once Minikube is installed and started, you can use `kubectl` to interact with your local Kubernetes cluster.

1. **Check the Status:**
   `minikube status`
2. **Get Cluster Info:**
   `kubectl cluster-info`
3. **Create a Deployment:**
   `kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.4`
4. **Expose the Deployment:**
   `kubectl expose deployment hello-minikube --type=NodePort --port=8080`
5. **Get the URL to Access the Service:**
   `minikube service hello-minikube --url`
6. **Stop Minikube:**
   `minikube stop`
7. **Delete the Minikube Cluster:**
   `minikube delete`

Advanced Usage

1. **Access Minikube Dashboard:**
   `minikube dashboard`
2. **Specify a Different Driver:**
   `minikube start --driver=virtualbox`
3. **Allocate More Resources:**
   `minikube start --memory=4096 --cpus=4`

4. **Enable Addons:**
   ```
   minikube addons enable <addon-name>
   ```

Troubleshooting

1. **Check Logs:**
   ```
   minikube logs
   ```
2. **Delete and Restart:**
   If Minikube fails to start or behaves unexpectedly, you can reset it:
   ```
   minikube delete

   minikube start
   ```

By following these steps, you should have a fully functional local Kubernetes cluster running with Minikube, enabling you to develop and test your applications in a Kubernetes environment.