Create Serverless Apps Using Kubernetes and Knative: A Complete Guide

Serverless computing allows developers to focus solely on writing code without worrying about the infrastructure. With **Kubernetes** and **Knative**, you can create serverless applications that scale automatically, reduce operational complexity, and run seamlessly across any environment — on-premises or in the cloud. In this guide, we'll walk you through how to set up a serverless application using Kubernetes and Knative, from start to finish.

Table of Contents

1. What is Serverless Computing?

Serverless computing allows you to run applications without provisioning or managing servers. In a serverless model, the infrastructure scales dynamically based on the demand for the app, and developers pay only for the resources consumed. Kubernetes with Knative takes serverless to the next level by providing flexibility, scalability, and a powerful ecosystem to manage serverless workloads.

2. Overview of Kubernetes and Knative

Kubernetes:

Kubernetes is a powerful open-source platform for automating the deployment, scaling, and operation of containerized applications. It offers robust orchestration, which makes it easier to manage large clusters of containers.

Knative:

Knative is an open-source extension to Kubernetes designed to support serverless workloads. It simplifies the process of deploying and managing serverless apps on Kubernetes. Knative consists of two main components:

- **Knative Serving**: Manages the deployment of serverless containers and provides scaling, auto-scaling, and routing capabilities.
- **Knative Eventing**: This enables apps to consume and react to events in a loosely coupled manner.

3. Prerequisites

Before getting started, ensure you have the following:

- A Kubernetes cluster (You can set this up locally using **Minikube** or use a managed Kubernetes service like **GKE**, **AKS**, or **EKS**)
- **kubectl** installed (Kubernetes command-line tool)
- Docker installed for building container images
- Helm (optional but recommended for easier installations)
- Basic knowledge of Docker, Kubernetes, and YAML files

4. Setting Up Kubernetes

First, you need a Kubernetes cluster running. You can set up Kubernetes locally using **Minikube** or use a cloud provider's managed Kubernetes services like Google Kubernetes Engine (GKE), Amazon EKS, or Azure AKS.

Step 1: Install Minikube (Optional)

To run Kubernetes locally, install **Minikube**.

```
bash

# Install Minikube

curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube-lin
ux-amd64

sudo install minikube-linux-amd64 /usr/local/bin/minikube

# Start Minikube

minikube start --driver=docker
```

Step 2: Verify Kubernetes Cluster

Ensure your Kubernetes cluster is running by using the following command:

```
bash

Copy code

kubectl get nodes
```

You should see a list of nodes with the status `Ready`.

5. Installing Knative

Once your Kubernetes cluster is ready, the next step is to install Knative. We'll install **Knative Serving** for managing serverless workloads.

Step 1: Install Istio (for Networking)

Knative uses Istio as the default ingress and service mesh provider. Install Istio by following these steps:

```
bash

Copy code

kubectl apply -f https://github.com/knative/net-istio/releases/latest/download/release.yaml
```

Check if the Istio pods are up and running:

```
bash

Copy code

kubectl get pods -n istio-system
```

Step 2: Install Knative Serving

Next, install **Knative Serving**:

```
bash

Copy code

kubectl apply -f https://github.com/knative/serving/releases/latest/download/serving-crds.yaml

kubectl apply -f https://github.com/knative/serving/releases/latest/download/serving-core.yaml
```

Verify that Knative Serving is running:

```
bash

Copy code

kubectl get pods -n knative-serving
```

You should see pods like `controller`, `autoscaler`, and `webhook` running.

6. Deploying a Serverless Application

Now that Knative is installed, let's deploy a simple serverless application. We'll deploy a containerized application that responds with "Hello, World!"

Step 1: Create a Knative Service YAML File

Create a file called `service.yaml` with the following content:

```
yaml

Copy code

apiVersion: serving.knative.dev/v1

kind: Service

metadata:

 name: helloworld

spec:

 template:

   spec:

     containers:

       - image: gcr.io/knative-samples/helloworld-go

         env:

           - name: TARGET

             value: "Hello, Knative!"
```

Step 2: Apply the Service

Deploy the Knative Service using `kubectl`:

```
bash

Copy code

kubectl apply -f service.yaml
```

Knative will automatically create the necessary Kubernetes resources, including a `Pod` and a `Route`, to manage your application.

Step 3: Verify the Deployment

Check if the service has been deployed successfully:

```
bash

Copy code

kubectl get ksvc
```

7. Scaling and Autoscaling with Knative

Knative automatically scales the application based on traffic. You can simulate traffic to observe autoscaling in action.

Step 1: Send Requests to the Service

Find the URL of the service by running:

```
bash

Copy code

kubectl get ksvc
```

Then, use `curl` to send requests to the application:

```
bash

Copy code

curl http://<your-service-url>
```

Step 2: Autoscaling

Knative scales down to zero when no requests are being processed. You can monitor the scaling behavior by checking the number of pods:

```
bash

Copy code

kubectl get pods
```

8. Managing Traffic with Knative Routing

Knative also supports traffic splitting between different versions of your application. This is useful for A/B testing or canary releases.

Step 1: Deploy a New Revision

Modify the `service.yaml` file and update the environment variable `TARGET`:

```
yaml

Copy code
```

```
env:
 - name: TARGET
   value: "Hello, Knative Version 2!"
```

Deploy the updated version:

```
bash
Copy code
kubectl apply -f service.yaml
```

Step 2: Split Traffic

To split traffic between two revisions, modify the `traffic` section of your YAML file:

```
yaml
Copy code
spec:
 traffic:
   - revisionName: helloworld-00001
     percent: 50
   - revisionName: helloworld-00002
     percent: 50
```

9. Conclusion

Congratulations! You've successfully set up a serverless application using Kubernetes and Knative. With this setup, you can deploy, scale, and manage serverless applications efficiently. Knative makes it easy to build modern, scalable applications on top of Kubernetes, combining the power of containers with the simplicity of serverless.

By leveraging Knative's autoscaling and traffic management features, you can create robust applications that respond to demand while minimizing resource usage when idle.

Additional Resources

- Kubernetes Official Documentation
- Knative Documentation
- Minikube Setup Guide