Introduction to Cloud Computing Report

Lab9: Knative Serving and Eventing Hands-on Lab

Name: Yin Yuang
Student ID: 202383930027
Class: Software Engineering Class 1

Introduction to Cloud Computing
(Autumn,2025)

Nanjing University of Information Science and Technology, China
Waterford Institute

I. Objective

The objective of this lab experiment is to gain hands-on experience with Knative for building, deploying, and managing serverless, cloud-native applications on Kubernetes.

Specifically, this experiment focuses on installing and configuring Knative Serving and Knative Eventing using official YAML manifests. Istio is deployed as the networking layer to support ingress traffic management for Knative services. In addition, Knative Functions are explored through local development and execution using a local Docker registry, while intentionally skipping function deployment to Kubernetes in accordance with the lab requirements.

II. Content

1. Environment Setup

The experiment was conducted on a local machine using Minikube to create a Kubernetes cluster, with Docker configured as the container runtime.

2. Knative Serving Installation

Knative Serving was installed using official manifests to deploy its core control plane components. Istio was configured as the networking layer to enable ingress traffic for Knative services.

3. Knative Eventing Installation

Knative Eventing was installed to support event-driven workloads, and its control plane components were verified to be running correctly.

4. Knative Functions Toolchain and Project Creation

The Knative Functions CLI was used to verify the local toolchain and to create a Go-based function project for local development.

5. Local Function Execution and Build

The function was executed and invoked locally using a local Docker registry to verify runtime behavior. The function image was then built and pushed to the local registry, while deployment to Kubernetes was intentionally skipped.

6. Knative Serving Service Deployment

A sample Hello World service was deployed to verify Knative Serving functionality, and the service reached a ready state.

III. Result

This section presents the experimental results in a logical order, demonstrating the successful installation, configuration, and verification of Knative Serving, Knative Eventing, and Knative Functions in a local Kubernetes environment.

1. Knative Serving Core Components

```
C:\Users\yin\Downloads\Cloud\Cloud\Videos\yaml>kubectl get pods -n knative-serving
NAME                                READY   STATUS    RESTARTS      AGE
activator-7bcd47489b-x44ql          1/1     Running   0             18h
autoscaler-65cf6767c4-5b6b7         1/1     Running   0             18h
controller-964dcf97b-4x7dq          1/1     Running   0             18h
net-istio-controller-794dbdb89d-ctqpd   1/1     Running   1 (17h ago)   17h
net-istio-webhook-bbf6f8fcd-s4l72   1/1     Running   0             17h
webhook-658b566b8-jq6d6             1/1     Running   0             18h
```

Figure 1: Knative Serving pods running in the knative-serving namespace

After applying the Knative Serving CRDs and core manifests, all Serving-related pods were observed to be running in the `knative-serving` namespace. This confirms that the Serving control plane components were initialized correctly and are able to manage Knative services.

Conclusion: Knative Serving was successfully installed and is operational.

2. Istio Networking Layer

```
C:\Users\yin\Downloads\Cloud\Cloud\Videos\yaml>kubectl get pods -n istio-system
NAME                                   READY   STATUS    RESTARTS   AGE
istio-ingressgateway-6c6444c759-cghgt   1/1     Running   0          17h
istio-ingressgateway-6c6444c759-ldz8m   1/1     Running   0          17h
istio-ingressgateway-6c6444c759-wtjqd   1/1     Running   0          17h
istiod-66cf796db8-4swnj                 0/1     Pending   0          17h
istiod-66cf796db8-j6h4m                 1/1     Running   0          17h
istiod-66cf796db8-z2h9p                 0/1     Pending   0          17h
```

Figure 2: Istio control plane pods running successfully

Knative Serving relies on a networking layer to handle ingress traffic. The screenshot shows that Istio control plane components are running normally in the `istio-system` namespace, providing traffic routing and gateway support for Knative services.

Conclusion: The Istio networking layer was deployed successfully and is ready to support Knative Serving.

3. Certificate Management Support

```
C:\Users\yin\Downloads\Cloud\Cloud\Videos\yaml>kubectl get pods -n cert-manager
NAME                                       READY   STATUS    RESTARTS      AGE
cert-manager-548f7cf98c-4j2lx              1/1     Running   2 (94m ago)   18h
cert-manager-cainjector-8798f647f-5hgps    1/1     Running   0             18h
cert-manager-webhook-6c8678dc46-7td8f      1/1     Running   0             18h
```

Figure 3: Cert-Manager components running successfully

Cert-Manager was installed to support certificate provisioning required by Istio. All related components are shown to be running correctly, indicating that certificate management capabilities are available within the cluster.

Conclusion: Certificate management was configured successfully for the networking layer.

4. Knative Eventing Components

```
C:\Users\yin\Downloads\Cloud\Cloud\Videos\yaml>kubectl get pods -n knative-eventing
NAME                                   READY   STATUS    RESTARTS   AGE
eventing-controller-645c4bcd55-tvc9j   1/1     Running   0          17m
eventing-webhook-7fd9cb958f-5f76m      1/1     Running   0          17m
imc-controller-6b9fbb6487-ltpwp        1/1     Running   0          17m
imc-dispatcher-6c4b5856d-fdt2f         1/1     Running   0          17m
job-sink-5cc89b5d95-995pf              1/1     Running   0          17m
mt-broker-controller-568d6b9c59-9hm4s  1/1     Running   0          17m
mt-broker-filter-db66554c4-vc6d7       1/1     Running   0          17m
mt-broker-ingress-774547844d-sp66g     1/1     Running   0          17m
```

Figure 4: Knative Eventing pods running successfully

Following the installation of Knative Eventing, all Eventing-related pods entered the `Running` state. This demonstrates that the cluster is capable of supporting event-driven workloads in addition to request-driven services.

Conclusion: Knative Eventing was installed successfully and is ready for use.

5. Knative Functions Toolchain Verification

```
C:\Users\yin>kn func version
v0.47.1
```

Figure 5: Knative Functions CLI version verification

The Knative Functions CLI was verified using the `kn func version` command, confirming that the local development environment is properly configured for function-based development.

Conclusion: The Knative Functions CLI is available and functioning correctly.

6. Local Function Development and Execution

```
C:\Users\yin\Downloads\Cloud\Cloud\Videos\yaml>kn func create -l go hello
Created go function in C:\Users\yin\Downloads\Cloud\Cloud\Videos\yaml\hello
```

Figure 6: Creation of a Go-based Knative Function

A Go-based function named `hello` was created successfully using the Knative Functions CLI, generating the required project structure.

```
C:\Users\yin\Downloads\Cloud\Cloud\Videos\yaml\hello>func run --registry localhost:5001
function up-to-date. Force rebuild with --build
Function running on 127.0.0.1:54160
Initializing HTTP function
listening on http port 8080
```

Figure 7: Function running locally using a local Docker registry

The function was executed locally with a specified local Docker registry. The output confirms that the function started correctly, listened for HTTP requests, and processed incoming requests as expected.

Figure 8: Successful function invocation returning "Hello World"

Invocation results show that the function returned the expected response, verifying correct runtime behavior.

Conclusion: Knative Functions can be developed and executed locally without deployment to Kubernetes, as required by the lab instructions.

7. Function Image Build Result

Figure 9: Function image built and pushed to local registry

The function image was successfully built and pushed to the local Docker registry. Deployment to Kubernetes was intentionally skipped in accordance with the lab requirements.

Conclusion: The function build workflow using a local registry was completed successfully.

8. Knative Serving Hello World Service

Figure 10: Knative Service deployed successfully with READY=True

The Knative Hello World service reached the `READY=True` state after deployment, demonstrating that Knative Serving can correctly create, manage, and run serverless services on Kubernetes.

IV. Conclusion

This lab experiment provided hands-on experience with Knative for building, deploying, and managing serverless, cloud-native applications on Kubernetes. Knative Serving and Knative Eventing were successfully installed and configured using official YAML manifests, demonstrating the ability to support both request-driven and event-driven workloads.

Istio was deployed as the underlying networking layer, and Cert-Manager was used to provide certificate management support, ensuring that ingress traffic could be handled securely and reliably. The successful operation of all control plane components confirmed the correctness of the installation and configuration process.

In addition, Knative Functions were explored through local development and execution using a local Docker registry. By intentionally skipping function deployment to Kubernetes, this experiment focused on understanding the local function development workflow, which is an important part of serverless application development.

Overall, this experiment deepened the understanding of Knative's architecture and core components, and demonstrated how Knative enables scalable, flexible, and cloud-native serverless applications on top of Kubernetes.