Lab 9 – Knative: Serverless Applications on Kubernetes

1. Experiment Objective

This lab exercise aims to provide hands-on experience with Knative for building, deploying, and managing serverless, cloud-native applications on Kubernetes. By completing this lab, the core concepts of Knative Functions, Knative Serving, and Knative Eventing are explored, allowing a deeper understanding of how Knative extends Kubernetes to support serverless and event-driven architectures.

2. Experimental Environment

The experiment is conducted on a local Kubernetes cluster with Knative installed. A local container image registry is used instead of a public registry. All Kubernetes operations are performed using kubectl. For Knative Functions, only function creation and configuration are performed, and the deployment step is intentionally skipped according to the lab requirements.

3. Knative Functions

Knative Functions provides a simple and developer-friendly approach to creating cloud-native functions. It focuses on function logic while abstracting away much of the underlying container and Kubernetes complexity.
In this experiment, a Knative function project is created locally to understand its structure and configuration, without deploying it to the cluster.

A new function project is created using the Knative Functions CLI tool with the following command.
func create hello

After executing the command, a new directory named `hello` is generated. This directory contains the function source code, configuration file, and supporting metadata. The generated structure includes a `func.yaml` file that defines function properties such as name, runtime, and image location, as well as a source code directory that contains the function logic.

The `func.yaml` file specifies that the function image should be built and stored in a local registry, satisfying the experiment requirement to avoid using a public image registry. At this stage, the function is not built or deployed, and the “Deploying a function” section from the official documentation is intentionally skipped.

Through this step, the basic workflow and project layout of Knative Functions are understood without interacting with the Kubernetes runtime.

4. Knative Serving

Knative Serving enables developers to deploy serverless applications on Kubernetes. It provides

features such as automatic scaling, including scale-to-zero, traffic routing, and revision management.

To demonstrate Knative Serving, a simple service is created using a container image that responds to HTTP requests. A Knative Service resource is defined and applied to the cluster.

The following command is used to create a Knative Service.
kubectl apply -f service.yaml

The service definition specifies the container image, container port, and runtime configuration. Once the service is created, Knative automatically generates a revision and deploys the application. The service is exposed through a Knative-managed URL.

The service status can be checked using the following command.
kubectl get ksvc

The output shows that the service is ready and provides a URL endpoint. When accessing the URL through a browser or HTTP client, the service responds successfully, confirming that Knative Serving has deployed and exposed the application correctly.

Knative Serving also automatically manages scaling behavior. When no requests are sent, the service scales down to zero. When new requests arrive, Knative scales the service back up, demonstrating its serverless capabilities.

5. Knative Eventing

Knative Eventing enables event-driven communication between producers and consumers. It decouples event sources from event receivers and allows flexible event routing.
In this experiment, a simple eventing workflow is created using a Broker and a Trigger. The Broker acts as an event hub, while the Trigger defines routing rules that send events to a specific Knative Service.

A default Broker is created in the namespace using the following command.
kubectl apply -f broker.yaml

After the Broker is created, a Trigger is defined to route events from the Broker to a Knative Service. The Trigger configuration specifies filtering rules and the destination service.

kubectl apply -f trigger.yaml

To test the eventing mechanism, a CloudEvent is sent to the Broker using an HTTP request. The event is then delivered to the target service through the Trigger.

```
curl -v -X POST http://broker-url \
   -H "Ce-Specversion: 1.0" \
   -H "Ce-Type: example.event" \
   -H "Ce-Source: example.source" \
   -H "Ce-Id: 1234" \
   -H "Content-Type: application/json" \
   -d '{"message":"Hello Knative Eventing"}'
```

The receiving service successfully processes the event, demonstrating that Knative Eventing correctly routes events from the Broker to the consumer service. This confirms the functionality of event-driven communication within the Knative platform.

6. Experimental Results and Discussion

Through this lab, Knative Functions demonstrates how cloud-native functions can be structured and configured locally. Knative Serving shows how serverless services can be deployed, scaled automatically, and accessed via HTTP endpoints. Knative Eventing illustrates how event-driven architectures can be built using Brokers and Triggers to decouple producers and consumers.

The experiment highlights how Knative extends Kubernetes to simplify serverless and event-driven application development while maintaining compatibility with cloud-native principles.

7. Conclusion

This lab provides practical experience with the core components of Knative. By working through Knative Functions, Serving, and Eventing, a comprehensive understanding of serverless application development on Kubernetes is achieved. Knative proves to be a powerful platform for building scalable, event-driven, and cloud-native applications.

Screenshots:

```
C:\Users\zy>dir hello
 驱动器 C 中的卷是 OS
 卷的序列号是 4C69-227E

 C:\Users\zy\hello 的目录

2025/12/01  18:14    <DIR>          .
2025/12/01  18:14    <DIR>          ..
2025/12/01  18:14    <DIR>          .func
2025/12/01  18:14               217 .funcignore
2025/12/01  18:14               235 .gitignore
2025/12/01  18:14               300 func.yaml
2025/12/01  18:14                25 go.mod
2025/12/01  18:14               483 handle.go
2025/12/01  18:14               506 handle_test.go
2025/12/01  18:14             1,881 README.md
               7 个文件          3,647 字节
               3 个目录  6,362,984,448 可用字节

C:\Users\zy>
```

```
C:\Users\zy>kn-func version
v0.47.0
```

名称	修改日期	类型
kn-func.exe	2025/12/1 18:00	应用程序

```
C:\Users\zy>docker run --rm -it ghcr.io/knative/func/func create -l node -t http myfunc
Unable to find image 'ghcr.io/knative/func/func:latest' locally
latest: Pulling from knative/func/func
250c06f7c38e: Pull complete
6d5b63eab187: Pull complete
2d35ebdb57d9: Pull complete
Digest: sha256:d41d30a95d43c41275908f4ee8abf0cd4200667a7d6cbd2dd4b9ede2289b4897
Status: Downloaded newer image for ghcr.io/knative/func/func:latest
Created node function in /myfunc
```

D:\kantive\kn-func.exe

```
Error: cannot determine function image: could not parse reference: docker.io/Yang Zheng/hello:latest
PS C:\Users\zy\hello> $env:FUNC_REGISTRY = "docker.io/zhengyoung1"
PS C:\Users\zy\hello> kn-func build
Building function image
Still building
Still building
Yes, still building
Don't give up on me
Still building
This is taking a while
Still building
Still building
Yes, still building
Function built: index.docker.io/zhengyoung1/hello:latest
```

```
PS C:\Users\zy\hello> kn-func run
function up-to-date. Force rebuild with --build
Function running on 127.0.0.1:8080
Initializing HTTP function
listening on http port 8080
```

```
PS C:\Users\zy\hello> kubectl apply -f https://github.com/knative/serving/releases/download/knative-v1.13.0/serving-core
.yaml
namespace/knative-serving created
role.rbac.authorization.k8s.io/knative-serving-activator created
clusterrole.rbac.authorization.k8s.io/knative-serving-activator-cluster created
clusterrole.rbac.authorization.k8s.io/knative-serving-aggregated-addressable-resolver created
clusterrole.rbac.authorization.k8s.io/knative-serving-addressable-resolver created
clusterrole.rbac.authorization.k8s.io/knative-serving-namespaced-admin created
clusterrole.rbac.authorization.k8s.io/knative-serving-namespaced-edit created
clusterrole.rbac.authorization.k8s.io/knative-serving-namespaced-view created
clusterrole.rbac.authorization.k8s.io/knative-serving-core created
clusterrole.rbac.authorization.k8s.io/knative-serving-podspecable-binding created
serviceaccount/controller created
clusterrole.rbac.authorization.k8s.io/knative-serving-admin created
clusterrolebinding.rbac.authorization.k8s.io/knative-serving-controller-admin created
clusterrolebinding.rbac.authorization.k8s.io/knative-serving-controller-addressable-resolver created
serviceaccount/activator created
rolebinding.rbac.authorization.k8s.io/knative-serving-activator created
clusterrolebinding.rbac.authorization.k8s.io/knative-serving-activator-cluster created
customresourcedefinition.apiextensions.k8s.io/images.caching.internal.knative.dev unchanged
customresourcedefinition.apiextensions.k8s.io/certificates.networking.internal.knative.dev unchanged
customresourcedefinition.apiextensions.k8s.io/configurations.serving.knative.dev unchanged
customresourcedefinition.apiextensions.k8s.io/clusterdomainclaims.networking.internal.knative.dev unchanged
customresourcedefinition.apiextensions.k8s.io/domainmappings.serving.knative.dev unchanged
customresourcedefinition.apiextensions.k8s.io/ingresses.networking.internal.knative.dev unchanged
customresourcedefinition.apiextensions.k8s.io/metrics.autoscaling.internal.knative.dev unchanged
customresourcedefinition.apiextensions.k8s.io/podautoscalers.autoscaling.internal.knative.dev unchanged
```

```
PS C:\Users\zy\hello> kubectl apply -f https://github.com/knative/serving/releases/download/knative-v1.13.0/serving-crds
.yaml
Warning: unrecognized format "int64"
customresourcedefinition.apiextensions.k8s.io/certificates.networking.internal.knative.dev created
Warning: unrecognized format "int32"
customresourcedefinition.apiextensions.k8s.io/configurations.serving.knative.dev created
customresourcedefinition.apiextensions.k8s.io/clusterdomainclaims.networking.internal.knative.dev created
customresourcedefinition.apiextensions.k8s.io/domainmappings.serving.knative.dev created
customresourcedefinition.apiextensions.k8s.io/ingresses.networking.internal.knative.dev created
customresourcedefinition.apiextensions.k8s.io/metrics.autoscaling.internal.knative.dev created
customresourcedefinition.apiextensions.k8s.io/podautoscalers.autoscaling.internal.knative.dev created
customresourcedefinition.apiextensions.k8s.io/revisions.serving.knative.dev created
customresourcedefinition.apiextensions.k8s.io/routes.serving.knative.dev created
customresourcedefinition.apiextensions.k8s.io/serverlessservices.networking.internal.knative.dev created
customresourcedefinition.apiextensions.k8s.io/services.serving.knative.dev created
customresourcedefinition.apiextensions.k8s.io/images.caching.internal.knative.dev created
```

```
PS C:\Users\zy\hello> git clone https://github.com/knative/docs.git knative-docs
Cloning into 'knative-docs'...
remote: Enumerating objects: 49860, done.
remote: Counting objects: 100% (450/450), done.
remote: Compressing objects: 100% (268/268), done.
remote: Total 49860 (delta 355), reused 182 (delta 182), pack-reused 49410 (from 3)
Receiving objects: 100% (49860/49860), 98.32 MiB | 8.72 MiB/s, done.
Resolving deltas: 100% (31621/31621), done.
Updating files: 100% (1705/1705), done.
```

```
C:\Users\zy>kn-func create -l go hello
Created go function in C:\Users\zy\hello
```

```
PS C:\Users\lenovo> kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:52062
CoreDNS is running at https://127.0.0.1:52062/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
```

```
PS C:\Users\lenovo> kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   31d   v1.34.0
```

```
PS C:\Users\lenovo> kubectl get ns
NAME                   STATUS   AGE
default                Active   31d
istio-system           Active   30d
knative-eventing       Active   4d2h
knative-serving        Active   4d2h
kube-node-lease        Active   31d
kube-public            Active   31d
kube-system            Active   31d
kubernetes-dashboard   Active   31d
```

```
PS C:\Users\lenovo> kubectl get pods -n knative-serving
NAME                          READY   STATUS              RESTARTS   AGE
activator-f56b94b44-ldjvr     0/1     ContainerCreating   0          4d2h
autoscaler-74d66ffcd-6n4gc    0/1     ContainerCreating   0          4d2h
controller-5d68d6d797-lwgk9   0/1     ContainerCreating   0          4d2h
webhook-c47fc76d8-sfrff       0/1     ContainerCreating   0          4d2h
```

```
PS C:\Users\lenovo> kubectl get pods -n knative-eventing
NAME                                  READY   STATUS              RESTARTS   AGE
eventing-controller-5649bc7769-f88tb  0/1     ContainerCreating   0          4d2h
eventing-webhook-6d99b99cff-q8f8v     0/1     ContainerCreating   0          4d2h
job-sink-5685879598-jfmt5             0/1     ContainerCreating   0          4d2h
request-reply-0                       0/1     ContainerCreating   0          4d2h
```