Installing Gloo and Knative

This lesson outlines the process for integrating Gloo and Knative with Jenkins X.

We'll cover the following

We could visit the **Knative** documentation and follow the instructions to install it and configure it. Then we could reconfigure Jenkins X to use it. But we won't do any of that, because Jenkins X already comes with a method to install and integrate **Knative**. Jenkins X allows us to install the **Gloo** add-on, which, in turn, will install **Knative**.

What is Gloo?

Gloo is a Kubernetes Ingress controller and API gateway. The main reason for using it in our context is its ability to route requests to applications managed and autoscaled by **Knative**. The alternative to **Gloo** would be **Istio**, which, even though is very popular, is too heavy and complex.

Installing Gloo and Knative

Let's proceed by installing the `glooctl` CLI.

Please follow the Install command-line tool (CLI) instructions.

Now we can use `glooctl` to install **Knative**.

```
glooctl install knative \
    --install-knative-version=0.9.0
```

The process installed **Gloo** and **Knative** in our cluster.

Retrieving the IP of the Kubernetes service

There's one more thing missing for us to be able to run serverless applications using **Knative**. We need to configure it to use a domain (in this case `nip.io`). So, the first step is to get the IP of the **Knative** service. However, the command differs depending on whether you're using EKS or some other Kubernetes flavor.

For GKE and AKS

> ⚠ Please run the command that follows only if you are **NOT** using **EKS** (e.g., GKE, AKS, etc.).

```
KNATIVE_IP=$(kubectl \
    --namespace gloo-system \
    get service knative-external-proxy \
    --output jsonpath="{.status.loadBalancer.ingress[0].ip}")
```

For EKS

> ⚠ Please run the commands that follow only if you are using **EKS**.

```
KNATIVE_HOST=$(kubectl \
    --namespace gloo-system \
    get service knative-external-proxy \
    --output jsonpath="{.status.loadBalancer.ingress[0].hostname}")

export KNATIVE_IP="$(dig +short $KNATIVE_HOST \
    | tail -n 1)"
```

Just tTo be on the safe side, we'll output the retrieved IP.

```
echo $KNATIVE_IP
```

If the output is an IP, everything is working smoothly so far. If it's empty, the load balancer was probably not yet created. If that's the case, wait for a few moments and try it again.

Modifying Knative configuration to use the domain

Now we can change the **Knative** configuration.

```
echo "apiVersion: v1
kind: ConfigMap
metadata:
  name: config-domain
  namespace: knative-serving
data:
  $KNATIVE_IP.nip.io: \"\"" \
    | kubectl apply --filename -
```

We used the IP of the LoadBalancer Service that was created during the **Knative** installation as a `nip.io` address in the **Knative** configuration. From now on, all applications deployed using **Knative** will be exposed using that address.

Checking namespaces

Let's take a closer look at what we got by exploring the Namespaces.

```
kubectl get namespaces
```

The output is as follows.

```
NAME            STATUS AGE
default         Active 67m
gloo-system     Active 2m1s
jx              Active 66m
jx-production   Active 59m
jx-staging      Active 59m
knative-serving Active 117s
kube-public     Active 67m
kube-system     Active 67m
```

We can see that we got two new Namespaces. As you can probably guess, `gloo-system` contains **Gloo** components, while **Knative** runs in `knative-serving`. Keep in mind that we did not get all the **Knative** components, only `serving`, which is in charge of running Pods as serverless loads.

Now, I could go into detail and explain the function of every Pod, service, CRD, and other components running in `gloo-system` and `knative-serving` Namespaces, but I feel that would be a waste of time. You can get that information yourself by exploring Kubernetes resources running in those Namespaces, or by going through the official documentation. What matters, for now, is that we got everything Jenkins X needs to convert your applications into serverless deployments.

We're almost done with the setup. **Knative** is installed in our cluster, but we still need to tell Jenkins X to use it as a default deployment mechanism. We can do that with the command that follows.

Setting Knative as the default deployment mechanism

```
jx edit deploy \
    --team \
    --kind knative \
    --batch-mode
```

From this moment on, all new projects will be serverless, unless we say otherwise. If you choose to change your mind, please re-run the same command, with the `default` kind instead of `knative`.

In the next lesson, let's create a new project and check it out.