T10 - Cloud

T-CLO-902

Kubi

Bootstrap

1.2

Docker

You are supposed to be familiar with Docker.
If it's not the case, start with the following tutorial and start your skill improvement.

It's never too late…

Local Kubernetes sandbox

Minikube is a local K8s cluster, with a single node.
It can be used as a sandbox to test several commands, such as:

```
$ minikube start --vm-driver virtualbox
           --extra-config=apiserver.service-node-port-range=1-30000

$ minikube status
$ minikube ip

$ kubectl config current-context
$ kubectl config use-context minikube

$ minikube stop
$ minikube delete
```

Play around with Minikube until you are ready to rumble.

Cloud Kubernetes sandbox

Most of IaaS offer Kubernetes as a Service, with free tier.
For instance:

- Digital Ocean (free tier: $100, 60 days)
- GCP (free tier: $300)

Before deploying things into Kubernetes, take some time to discover your cluster:

- list nodes, namepaces, pods…
- change your current namespace
- display k8s resources in a verbose way

For the rest of your learning, those Kubectl commands will be your best friends for debugging most things.

Kubernetes basics

Run Forest, run!

Install Docker and the Kubernetes CLI.
Both tools provide a bash autocomplete script.

Create your first pod with the following Docker image: samber/hello-world-nodejs.

There are 2 ways to run this pod into Kubernetes:

- with command line (similar to `$ kubectl run ...`), useful for fast debugging,
- with yaml (similar to `$ kubectl apply ...`) which is the right way to work with Kubernetes (*Infra as Code*).

Print the application logs (stdout+stderr), and some info about your container: ip, node, labels, uptime…

Then find a way to execute a shell command in this running container (uptime, date, echo, whatever…).

Now, delete this pod.

Communicate

Environment

Update your pod configuration and add the following environment variable: `PORT=8080`.
Find a way to display the environment variables of your running container.

Networking

Display the IP of this container. Is it reachable from outside the cluster?

Kubernetes provides an internal "overlay" network: a network shared by every node but not accessible from outside. Ask Kubernetes to expose the port 8080 to the cluster.

Forward pod port to localhost, then execute `$ curl localhost:8080` from your laptop terminal.

Port forwarding must be used for debugging only.

Addressing

When you destroy and recreate your pod many times, do you get the same container IP?

In any infrastructure, reaching a service with its IP address is the wrong way to communicate. You must always use DNS instead. Kubernetes embed a DNS server. Find a way to create an internal DNS for your hello-world container.

Can you see the pattern of DNS addresses ?

Start another container with "$ kubectl run" and send an HTTP request to hello-world DNS (using curl or netcat…).

If the server replies with "Hello world!", congratulations, you've done it!

Persist

Can you attach a 512MB volume to your container ?

Deployments

Take a look at Kubernetes "deployments". Do you see the difference with pods?

Starting from now, you won't start pods by yourself anymore.
Deployment is a much better abstraction.
Convert the "hello-world" pod into a Kubernetes Deployment.
Scale up!

In order to handle more requests on your hello-world API, start 3 instances of the container.

When you send HTTP requests to the hello-world DNS, it must be load balanced between your 3 instances.

Upgrade

The developers made a new release of the application.
It's time to upgrade the Docker image to `samber/hello-world-nodejs:v2`.
Check the deployment history of this hello-world Deployment.

Oops, developers inserted a bug.
Can you rollback to the previous version without downtime (using command line)?

Scheduling

Spread your 3 containers on 3 different Kubernetes nodes.

When your Kubernetes cluster have less than 3 nodes, what is the status of containers?

Find a way to reserve hardware resources to the container (such as 1GB of memory or some CPU).

Helm

Kubernetes is a highly-customizable container orchestrator, but for most deployments those yaml are way too verbose.**

In order to simplify deployment flows, the community built some tools that abstract configurations. You're going to discover Helm.

Install Helm and its bash autocompletion script.

If not done yet, check out the following tools: kubens, kubectx.

First deployment

Using Helm, deploy a PostgreSQL server.

At this phase, you don't need to create your own Helm chart.

- Can you list pods running in your cluster ?
- Can you send an SQL query to your database ?

Take some time to discover the Helm CLI: downgrade, upgrade, release history, undeploy…

Improve

The public Helm chart can be overridden by custom configurations.

- Can you change the username/password of the database ?
- For better availability, add 2 slaves to your PostgreSQL server

Now, you should be able to reach PostgreSQL slaves on a dedicated DNS endpoint.

The Helm chart you selecte might provide advanced PostgreSQL configurations.
Find a way to set the number of max connections To PostgreSQL instance, and the SQL query timeout.

Build

Now, create your first Helm template for the hello-world application you deployed earlier:

- deployment
- networking
- volumes
- …

Make your Helm chart generic: some settings should be customizable for people that are going to use it (instance count, Docker image, exposed port,…)

One more thing

Helm is a very good abstraction, for large infrastructure with dozens of micro-services. In this situation, you will be able to write 1 chart for many similar services.

In the company your working for, what level of abstraction would be appropriated ?

- 1 chart per language ?
- 1 chart per application type ?

Bonus: The sky is the limit

Helm is nice for creating deployment templates. But for a single application, you may need to set multiple configurations (for instance, replica count might be different in dev, staging, production).

It's time to take a look on Kustomize.

Can you write a generic/default description of your hello-world application ? Then a custom configuration for staging and production ?