Kubernetes Full Demo 2

- Requirements
  - Install kubectl
  - Install Lens (kubernetes GUI)
  - Install Apache Bench
  - Install Helm (package manager)
  - Github repo
  - Activate Kubernetes on Docker Desktop
- Minikube
  - Start minikube
  - Show addons
  - Enable plugins
- Kubectl basic commands
  - Get all
  - Get (resource)
  - Describe (resource)
- Example 01: deployment
  - Deploy
  - Show pods
  - View logs
  - Shell
  - Update a deployment
  - Connecting from the outside
    - Linux and Mac
    - Windows
    - Windows with Docker Desktop
  - Extra: Pod quarantine
- And now: LENS
  - Deploy a new software version
  - Delete resources
- Example 02: HPA
  - Horizontal POD autoscaling.
  - Launch HPA
  - Test HPA
    - Linux and Mac
    - Windows
- Example 3: helm (package manager)
  - Show available packages
  - Install package (wordpress)

- Test Wordpress
  - Linux
  - Windows
- Uninstall package

Example 4: Cluster on AWS
- Requirements
- Procedure
- Destroy Resources
  - Load balancer
  - Destroy deployment
  - Destroy cluster

Example 5: Rancher
- Installation and launching

Example 6: Digital Ocean
- Create a K8s Cluster
- Download cluster credentials
- Connect Docker repository with cluster
- Delete cluster

Example 5: POD communications
- Load balancer
  - Get minikube load balancer url
- Ingress
  - Enable ingress on minikube

Requirements

Install kubectl

https://kubernetes.io/es/docs/tasks/tools/install-kubectl/

Install Lens (kubernetes GUI)

https://k8slens.dev/

Install Apache Bench

> Linux /Mac : install package apache-utils

> Windows: download full apache from https://www.apachelounge.com/download/ and copy ab.exe to

> Windows\System32 (or a directory in the path).

Install Helm (package manager)

https://helm.sh/docs/intro/install/

Github repo

Clone or download as zip.

https://github.com/impalah/k8s-demo

Activate Kubernetes on Docker Desktop

Just activate from Docker Desktop configuration.

Minikube

Start minikube

From the command line:

```
minikube start
```

(show console)

1. Starts the cluster
2. Configures the cluster into the kubeconfig file.

Show addons

Minikube has some addons to provide services for the clusters like:

- extended metrics
- load balancers
- ingress controllers ...

```
minikube addons list
```

Enable plugins

We can enable or disable plugins.

```
minikube addons enable metrics-server
```

Kubectl basic commands

Get all

Show all the defined resources (only the default namespace).

```
kubectl get all
```

Show all resources for the kube-system namespace.

```
kubectl -n kube-system get all
```

The **kube-system** namespace contains the resources needed to execute kubernetes.

Get (resource)

Returns information about the requested resource.

```
kubectl get nodes
```

Return the list of nodes

```
kubectl get nodes -o wide
```

Returns the list of nodes with MORE information

Describe (resource)

Describe the given resource type and name. You can see a lot of information.

```
kubectl describe nodes docker-desktop
```

> Important: **Events**

Example 01: deployment

> Github repository: https://github.com/press-any-key-tech/001.docker-k8s-demo

> Directory: 03.k8s-basic-application

Deploy

In this example, we will deploy a simple application.

Explain the deployment file.

> **Metadata**: tags applied to a resource.

> **Selector**: metadata field used to apply a resource to another. e.g. deployment applied to a pod definition.

The application we are deploying just shows its version using http.

(change to 01.basic-application directory)

`kubectl apply -f 01-hello-app-deployment.yaml`

Show pods

`kubectl get pods`

You can see “Container Creating” in the status line.

> Ready: how many of the POD containers are ready (remember “sidecar” with several containers in the same POD)

`kubectl get pods -o wide`

Show extended information like the node the pod is running on and the internal IP address.

`kubectl get all`

You can view the PODS, the replicaset, and the deployment.

> Deployment: ready (3/3) and up-to-date.

View logs

`kubectl logs -f pod-name`

“logs”: recover stdout/stderr

> -f = follow; locks console and recover all messages from the given pod.

Shell

You can access a pod shell using the command “exec”.

`kubectl -it exec pod-name sh`

It’s being deprecated

`kubectl -it exec pod-name -- sh`

> -it means “interactive” to allow the linux console to interact with the k8s console.

(you can exit the shell using “exit”)

Update a deployment

Using kubectl to modify the parameter we want.

`kubectl scale --replicas=5 deployment.apps/hello`

Scale up to five replicas.

Do a get all again

`kubectl get all`

Modify the file and apply again

Example: modify the file replicas = 10 and apply

Return to the previous situation.

> Kubernetes stores the cluster’s internal state.

> When you apply a configuration Kubernetes compares the desired situation with the current one and applies the changes (creating, modifying, or deleting resources)

Connecting from the outside

To connect with a POD from the outside we need to create a service.

We will use a NodePort service in this example, which will connect the pods of a deployment with a port in the node or nodes executing them.

`kubectl apply -f 02-hello-app-service-node-port.yaml`

Use a get all

`kubectl get all`

Linux and Mac

Linux and Mac

As we are using minikube and we only have one node we need the node IP address

`minikube ip`

And then connect with the service using the provided address.

`curl http://192.168.49.2:30000`

Execute curl several times to see how the node changes. K8s uses a round-robin algorithm to balance the nodes to the services.

If the ip does not works execute:

`minikube service hello --url`

Windows

You need to make a tunnel.

`minikube service hello --url`

And use Powershell

`Invoke-WebRequest -Uri MY_URL -UseBasicParsing`

In Windows you will rarely see the Round Robin algorithm working. It’s Windows after all ...

Windows with Docker Desktop

```
kubetctl get svc
```

Access the application using localhost and the port of the services (usually above 30000)

Extra: Pod quarantine

Imagine that you have a POD that is failing and you want to stop sending traffic to that pod but don’t want to stop the pod to diagnose the error later.

Just change the tag of the pod. Kubernetes will take out the pod from the deployment (and from the service) and will start up a new pod to fill the deployment replicas.

`kubectl label pod hello-777579f864-f8kkd role=quarantine --overwrite`

Get all (or get pods)

`kubectl get all --show-labels`

And now: LENS

Launch lens. Explain a bit. Explain kubeconfig (and contexts)

```
kubectl config get-contexts
```

Change context

```
kubectl config use-context [name]
```

Deploy a new software version

Just update the file and include the new docker version.

(execute the 03 file or change the container program version to 2.0)

`kubectl apply -f 03-hello-app-deployment.yaml`

`kubectl get pods`

The POD we changed the role to “quarantine” remains untouched!!!

Delete resources

As easy as:

`kubectl delete service/hello`

To delete all resources:

`kubectl delete -f .`

Example 02: HPA

> Github repository: https://github.com/press-any-key-tech/001.docker-k8s-demo

> Directory: 04.k8s-hpa

Horizontal POD autoscaling.

We won't test VPA or node scaling as it is longer to configure. If you want to see how it works: another day.

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/

Show page

Apply:

- Service
- Deployment

Or apply all files in the folder:

```
kubectl apply -f <folder>
```

```
kubectl apply -f .
```

```
kubectl get all
```

Launch HPA

Explain the file: min, max, utilization

Explain metrics-server has to be running: show namespace kube-system.

```
kubectl apply -f 03-php-apache-hpa.yaml
```

"Get all" and explain the metrics % and the min/max/replicas.

```
kubectl get all
```

We configured the deployment to have 3 replicas but the HPA for a minimum of 1. Kubernetes has to scale down to the minimum if cpu is below configured average.

> HPA has a cooldown period, configurable… look in the documentation. By default: 5 minutes.

After 5 minutes only one pod will be up.

Test HPA

Linux and Mac

Get minikube ip address:

```
minikube ip
```

Launch a program to overload the pods:

```
ab -n 1000 -c 10 http://192.168.49.2:30000/
```

AB = apache bench

See how the metrics increase and the number of pods too. Then see the cooldown.

Windows

Start a tunnel and get ip address:

```
minikube service php-apache --url
```

Test:

```
Invoke-WebRequest -Uri MY_URL -UseBasicParsing
```

Launch a program to overload the pods:

```
ab -n 1000 -c 10 http://192.168.49.2:30000/
```

Be patient or increase the time (-c) to 20 seconds

Example 3: helm (package manager)

Show available packages

```
helm search hub [name]
```

```
helm search repo [name]
```

Install package (wordpress)

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

```
helm install my-wordpress bitnami/wordpress
```

It will show you how to use the package.

Show all the installed stuff

```
kubectl get all
```

Test Wordpress

Start a tunnel (maybe not needed)

```
minikube tunnel
```

Linux

Get minikube ip address:

```
minikube ip
```

Windows

Start a tunnel and get ip address:

```
minikube service my-wordpress --url
```

Uninstall package

First get installed packages

```
helm list
```

Then uninstall

```
helm uninstall my-wordpress
```

Example 4: Cluster on AWS

> Github repository: https://github.com/press-any-key-tech/001.docker-k8s-demo

> Directory: 05.k8s-aws-cluster

Requirements

- eksctl installed locally.
- Aws cli installed.
- Aws configured.

Procedure

From folder 03.aws.cluster

```
eksctl create cluster -f 01.simple-cluster.yaml
```

20 minutes!!!!

Deploy example (nginx)

`kubectl apply -f 02.nginx-deployment.yaml`

Deploy load balancer

`kubectl apply -f 03.nginx-lb.yaml`

Be patient (5 minutes)

Destroy Resources

Load balancer

`kubectl delete -f 03.nginx-lb.yaml`

Destroy deployment

`kubectl delete -f 02.nginx-deployment.yaml`

Destroy cluster

`eksctl delete cluster -f 01.simple-cluster.yaml`

Example 5: Rancher

Installation and launching

`docker run -d --restart=unless-stopped -p 80:80 -p 443:443 --privileged rancher/rancher`

Go lo localhost and complete installation

Rancher has to be accessible for the nodes so we need something like ngrok.

Provision a cluster and set master nodes, worker, etc.

Example 6: Digital Ocean

Create a K8s Cluster

```
doctl kubernetes cluster create \
   --region "nyc1" \
   --version "1.20.2-do.0" \
   --tag "demo" \
   --auto-upgrade \
```

```
    --maintenance-window="tuesday=20:00" \
    --node-pool="name=pool-001;size=s-1vcpu-2gb;count=2;tag=demo;auto-scale=true;min-no
  des=2;max-nodes=4"
    example-cluster-01
```

`doctl kubernetes cluster create --region "nyc1" --version "1.20.2-do.0" --tag "demo" --auto-upgrade --maintenance-window="tuesday=20:00" --node-pool="name=pool-001;size=s-1vcpu-2gb;count=2;tag=demo;auto-scale=true;min-nodes=2;max-nodes=4" example-cluster-01`

Download cluster credentials

`doctl kubernetes cluster kubeconfig save example-cluster-01`

Connect Docker repository with cluster

Use DO Registry control panel.

Delete cluster

`doctl kubernetes cluster delete example-cluster-01`

Example 5: POD communications

> Github repository: https://github.com/press-any-key-tech/001.docker-k8s-demo

> Directory: 05.k8s-aws-cluster

Load balancer

https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/

Change context to minikube and deploy the AWS cluster project.

Get minikube load balancer url

`minikube service example-service --url`

Ingress

From: https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/

Enable ingress on minikube

```
minikube addons enable ingress
```

 kubectl cheatsheet