

Experiment 03

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Theory:

A Kubernetes Cluster is a set of nodes that run containerized applications. Containerizing applications packages an app with its dependencies and some necessary services.

They are more lightweight and flexible than virtual machines. In this way, Kubernetes Cluster allow for applications to be more easily developed, moved and managed.

Kubernetes Cluster allow containers to run across multiple machines and environments: virtual, physical, cloud-based and on-premises. Kubernetes containers are not restricted to a specific operating system unlike virtual machines. Instead, they are able to share operating systems and run anywhere.

Kubernetes clusters are comprised of one master node and a number of worker nodes. These nodes can either be physical computers or virtual machines depending on the cluster.

Outputs:

Installing the Kubernetes-cli using chocolatey.

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>choco install kubernetes-cli
Chocolatey v0.11.2
Installing the following packages:
kubernetes-cli
By installing, you accept licenses for the packages.
Progress: Downloading kubernetes-cli 1.22.2... 100%

kubernetes-cli v1.22.2 [Approved]
kubernetes-cli package files install completed. Performing other installation steps.
The package kubernetes-cli wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): y

Extracting 64-bit C:\ProgramData\chocolatey\lib\kubernetes-cli\tools\kubernetes-client-windows-amd64.tar.gz to C:\ProgramData\chocolatey\lib\kubernetes-cli\tools...
C:\ProgramData\chocolatey\lib\kubernetes-cli\tools
Extracting 64-bit C:\ProgramData\chocolatey\lib\kubernetes-cli\tools\kubernetes-client-windows-amd64.tar to C:\ProgramData\chocolatey\lib\kubernetes-cli\tools...
C:\ProgramData\chocolatey\lib\kubernetes-cli\tools
ShimGen has successfully created a shim for kubectl-convert.exe
ShimGen has successfully created a shim for kubectl.exe
The install of kubernetes-cli was successful.
Software installed to 'C:\ProgramData\chocolatey\lib\kubernetes-cli\tools'

Chocolatey installed 1/1 packages.
```

Installing minikube:

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>choco install minikube
Chocolatey v0.11.2
Installing the following packages:
minikube
By installing, you accept licenses for the packages.
Progress: Downloading Minikube 1.23.2... 100%

Minikube v1.23.2 [Approved]
minikube package files install completed. Performing other installation steps.
ShimGen has successfully created a shim for minikube.exe
The install of minikube was successful.
Software install location not explicitly set, it could be in package or
default install location of installer.

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).

C:\Windows\system32>
```

Spinning up Kubernetes cluster:

```
Administrator: Command Prompt
C:\Windows\system32>minikube version
minikube version: v1.23.2
commit: 0a0ad764652082477c00d51d2475284b5d39ceed

C:\Windows\system32>minikube start
* minikube v1.23.2 on Microsoft Windows 10 Home Single Language 10.0.19042 Build 19042
* Automatically selected the docker driver. Other choices: hyperv, virtualbox, ssh
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.22.2 preload ...
  > preloaded-images-k8s-v13-v1...: 511.84 MiB / 511.84 MiB 100.00% 9.23 MiB
  > gcr.io/k8s-minikube/kicbase: 355.40 MiB / 355.40 MiB 100.00% 4.59 MiB p/
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.22.2 on Docker 20.10.8 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 3.9666517s
* Restarting the docker service may improve performance.
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

C:\Windows\system32>
```

```
Administrator: Command Prompt
C:\Windows\system32>kubectl version
Client Version: version.Info{Major:"1", Minor:"21", GitVersion:"v1.21.5", GitCommit:"aea7bbadd2fc0cd689de94a54e5b7b758869d691", GitTreeState:"clean", BuildDate:"2021-09-15T21:10:45Z", GoVersion:"go1.16.8", Compiler:"gc", Platform:"windows/amd64"}
Server Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.2", GitCommit:"8b5a19147530eaac9476b0ab82980b4088bbc1b2", GitTreeState:"clean", BuildDate:"2021-09-15T21:32:41Z", GoVersion:"go1.16.8", Compiler:"gc", Platform:"linux/amd64"}

C:\Windows\system32>kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:52665
CoreDNS is running at https://127.0.0.1:52665/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

C:\Windows\system32>kubectl get nodes
NAME        STATUS    ROLES          AGE   VERSION
minikube    Ready     control-plane,master  84s   v1.22.2

C:\Windows\system32>
```

Conclusion: Kubernetes architecture was studied in detail and a Kubernetes cluster was spun up successfully.