😊 Kubernetes

Kubernetes, often abbreviated as K8s (because there are eight letters between the 'K' and the 's' in Kubernetes), is an open-source platform designed to automate deploying, scaling, and managing containerized applications. Originally developed by Google, Kubernetes is now maintained by the Cloud Native Computing Foundation (CNCF), which is a part of the Linux Foundation.

At its core, Kubernetes allows you to treat your infrastructure as a set of loosely coupled, self-healing units, enabling you to deploy, update, and scale applications with ease. It provides features for automatic load balancing, storage orchestration, automatic rollout and rollback of application updates, service discovery and routing, and more.

Kubernetes works by abstracting away the underlying infrastructure and providing a unified API to manage containerized workloads and services. It organizes containers into logical units called pods, which are the basic deployable units in Kubernetes. These pods can contain one or more containers that are tightly coupled and share resources such as networking and storage.

Overall, Kubernetes has become the de facto standard for container orchestration in the cloud-native ecosystem, enabling organizations to build and manage scalable, resilient, and portable applications.

😄 Use cases of Kubernetes:

Kubernetes is widely used across various industries and for different purposes. Some common use cases include:

1. **Microservices Orchestration:** Kubernetes excels at managing microservices-based applications. It allows you to deploy, scale, and update individual microservices independently, providing flexibility and agility in complex distributed systems.
2. **Containerized Applications Deployment:** Kubernetes simplifies the deployment process for containerized applications. It ensures consistent deployment across different environments, whether it's on-premises, in the cloud, or hybrid setups.
3. **Scalable Web Applications:** Kubernetes makes it easy to scale web applications horizontally by adding or removing instances based on demand. It automatically manages load balancing and routing traffic to the appropriate containers, ensuring optimal performance and resource utilization.
4. **CI/CD Pipeline Orchestration:** Kubernetes integrates seamlessly with continuous integration and continuous delivery (CI/CD) pipelines. It enables automated testing, building, and deployment of applications, streamlining the software delivery process.
5. **Batch Processing and Big Data Workloads:** Kubernetes can efficiently manage batch processing and big data workloads using frameworks like Apache Spark, Apache Hadoop, or TensorFlow. It dynamically allocates resources based on workload requirements, improving resource utilization and reducing costs.

6. **Edge Computing:** Kubernetes can be deployed at the edge to manage containerized applications running on IoT devices or in remote locations. It provides centralized management and orchestration capabilities, even in resource-constrained environments.
7. **Hybrid and Multi-cloud Deployments:** Kubernetes enables organizations to deploy applications across hybrid and multi-cloud environments seamlessly. It abstracts away the underlying infrastructure, allowing applications to run consistently across different cloud providers or on-premises data centers.
8. **Stateful Applications:** While Kubernetes is primarily associated with stateless applications, it also supports stateful workloads such as databases, caching systems, and key-value stores. StatefulSets and persistent volumes in Kubernetes facilitate the management of stateful applications, ensuring data persistence and high availability.
9. **Development and Testing Environments:** Kubernetes provides a consistent environment for development, testing, and production. Developers can easily spin up isolated Kubernetes clusters for testing new features or replicating production environments without affecting other workloads.
10. **Service Mesh Integration:** Kubernetes integrates with service mesh frameworks like Istio and Linkerd to enhance observability, security, and traffic management within microservices architectures. Service meshes provide features such as traffic routing, load balancing, and encryption, which are essential for modern cloud-native applications.

😄 Minikube for Kubernetes Setup:

Minikube is a tool that allows you to run Kubernetes locally on your computer. It's designed to simplify the process of setting up a single-node Kubernetes cluster for development and testing purposes.

Here are some key features and aspects of Minikube:

1. **Local Kubernetes Cluster:** Minikube enables you to run a lightweight Kubernetes cluster on your local machine, providing an environment similar to a production Kubernetes cluster but on a smaller scale.
2. **Single Node Cluster:** By default, Minikube sets up a single-node Kubernetes cluster, which is suitable for development and testing scenarios. This allows developers to experiment with Kubernetes features without needing access to a larger cluster.
3. **Easy Installation:** Minikube is easy to install and configure on various operating systems, including Windows, macOS, and Linux. You can download the Minikube binary and start a Kubernetes cluster with just a few commands.
4. **Support for Different Hypervisors:** Minikube supports multiple hypervisors, including VirtualBox, Hyper-V, KVM, and Docker. This allows you to choose the virtualization technology that best fits your development environment.
5. **Cluster Management:** Minikube provides commands to start, stop, and delete Kubernetes clusters, as well as options to configure cluster settings such as CPU, memory, and Kubernetes version.

6. **Add-Ons and Extensions:** Minikube supports various add-ons and extensions that enhance the functionality of the local Kubernetes cluster. These add-ons include features like Kubernetes Dashboard, Metrics Server, and Ingress Controller.
7. **Integration with kubectl:** Minikube seamlessly integrates with kubectl, the Kubernetes command-line tool. This allows you to manage the local Kubernetes cluster using familiar Kubernetes commands.

In this guide, we're setting up a local Kubernetes environment using Minikube and Chocolatey on a Windows machine. The end goal is to provide a development environment where users can experiment with Kubernetes features, deploy containerized applications, and learn how to manage Kubernetes clusters locally.

The guide covers the installation of necessary tools such as Chocolatey, Minikube, Kubernetes CLI (kubectl), and VirtualBox. It then walks through the process of starting a Minikube cluster, deploying a sample application, exposing it as a service, accessing it through a browser, and finally cleaning up the environment by deleting resources and stopping Minikube.

Overall, the aim is to enable users to gain hands-on experience with Kubernetes in a local development environment, which can serve as a foundation for further learning and experimentation with container orchestration technologies.

😄 To begin with the Lab:

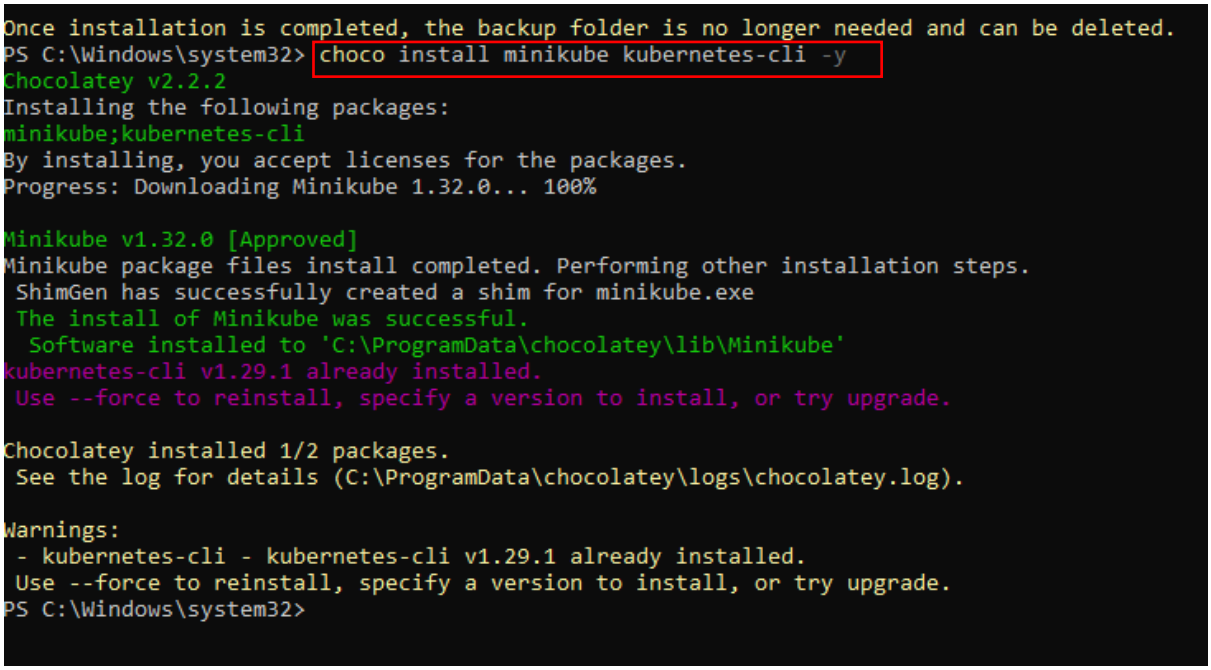
1. Now to work with Kubernetes first you need to set up an environment for it.
2. First you have to install chocolaty on your local machine. For that, you can use the below command. Now open your PowerShell as administrator and then paste the below script in it. Then you will see that chocolaty has been installed on your system.
3. If in case you are getting any error installing chocolaty on your local machine then you can turn off your anti-virus and then run the script again.

   Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

4. Now you are going to install Minikube on your system, for that you can run the command below.

   choco install minikube kubernetes-cli -y

5. Below you can see that Minikube has been installed.



6. Now you are going to install Oracle VM Virtual box on your system using Chocolaty. For that you can use the commands below.

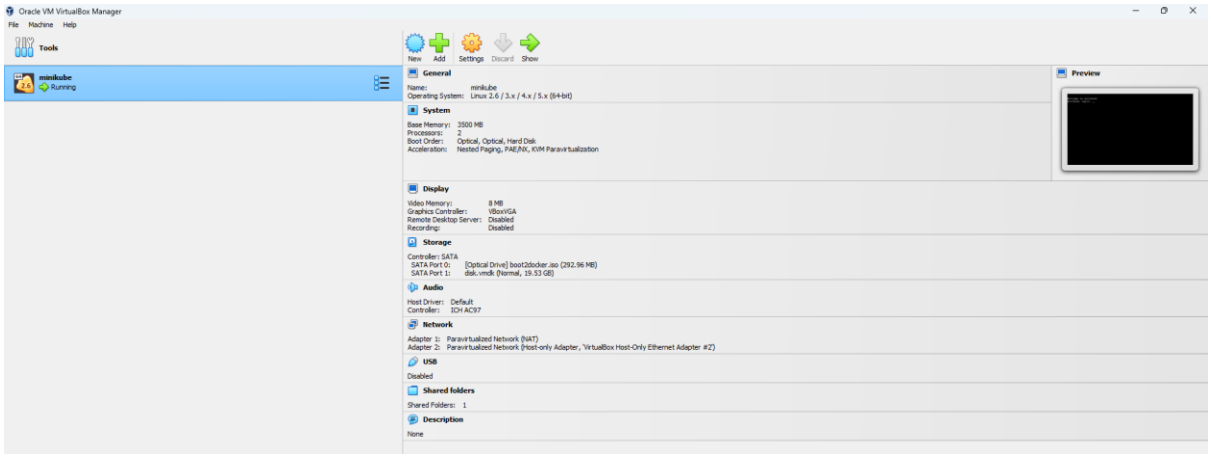
   choco install virtualbox

   vboxmanage –version

7. Once your Virtualbox has installed now you are going to start Minikube. For that you have to write **minikube start** in PowerShell or cmd.
8. If you are getting errors using minikube start command then you can use the command below.

   minikube start --no-vtx-check

9. Now it will take some time because an image is being downloaded on your virtual box.
10. And if you go inside your Virtual box then you can see that Minikube is running as expected.



11. In your command prompt also you can see that Minikube has successfully started.

```
* minikube v1.32.0 on Microsoft Windows 11 Home Single Language 10.0.22631.3447 Build 22631.3447
* Using the virtualbox driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Creating virtualbox VM (CPUs=2, Memory=3500MB, Disk=20000MB) ...
! This VM is having trouble accessing https://registry.k8s.io
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking
/proxy/
* Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Verifying Kubernetes components...
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

12. Now if you will run the command to check how many nodes you have then you can use the command below.

    kubectl get nodes

```
C:\Users>kubectl get nodes
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   3m3s   v1.28.3
```

13. Now you have a deployment command. This command created a deployment who's name is hello-minikube and it fetched an image from google.

    kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.10

```
$ kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.10
deployment.apps/hello-minikube created
```

14. Now run the command to check for pod and below you can see a pod running.

    kubectl.exe get pod

```
$ kubectl.exe get pod
NAME                              READY   STATUS    RESTARTS   AGE
hello-minikube-77d966488c-nxcrt   1/1     Running   0          85s
```

15. After that you are going to deploy it.

    kubectl.exe get deploy

```
$ kubectl.exe get deploy
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
hello-minikube   1/1     1            1           2m53s
```

16. Now to access this hello-minikube we are going to expose it.

    kubectl expose deployment hello-minikube --type=NodePort --port=8080

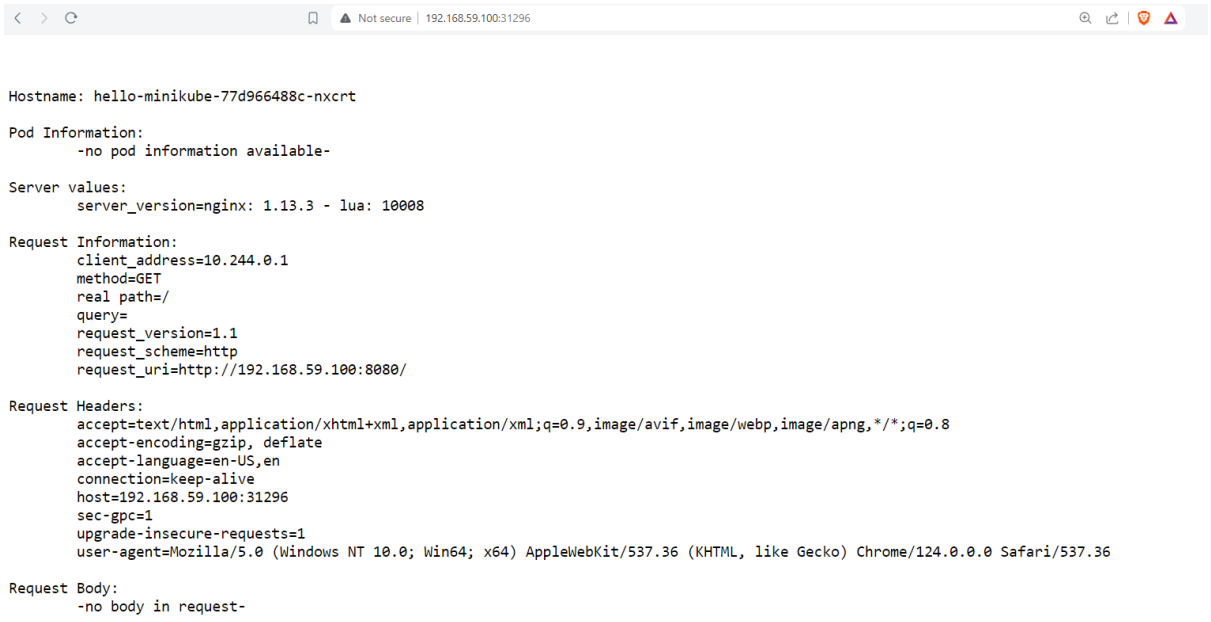
```
$ kubectl expose deployment hello-minikube --type=NodePort --port=8080
service/hello-minikube exposed
```

17. Now to access this service you will need a URL and for that you can run this command.

    minikube service hello-minikube –url

```
http://192.168.59.100:31296
```

18. Now copy this URL and paste it in browser. Then you will see this web page.



19. After that you are going to stop and delete these service.
20. For that you can use the commands below.

    kubectl.exe get svc

    kubectl.exe delete svc hello-minikube

kubectl.exe get deploy

kubectl.exe delete deploy hello-minikube

minikube.exe stop

minikube.exe delete