



Research & Development Team

Kubernetes for Developer

www.pnpsw.com

[sommai.k@gmail.com](mailto:sommaik@gmail.com)

081-754-4663

Lineid : sommaik

Medium: @sommaikrangpanich

สมหมาย กรังพานิช

ตำแหน่งปัจจุบัน

- กรรมการสมาคมอุตสาหกรรมซอฟต์แวร์ไทย ATSI
- กรรมการผู้จัดการบริษัท พี เอ็น พี โซลูชั่น จำกัด

ด้านความเชี่ยวชาญ

- ผู้เชี่ยวชาญด้านการออกแบบและพัฒนา Software ด้วย Framework ดังนี้
 - Java, Springboot, Go, Node.js
 - Full Stack, Angular, Vue, React, Svelte
 - Apollo GraphQL
 - Docker, Docker Swarm, K8s
 - Flutter, Dart
 - MongoDB, Oracle, MySQL, SQL Server, DB2, PostgreSQL
- ผู้เชี่ยวชาญด้านการออกแบบและประยุกต์ใช้ DevOps ในการพัฒนา Software
- ผู้เชี่ยวชาญด้านการออกแบบและประยุกต์ใช้ Cloud Native ในการพัฒนา Software
- ผู้เชี่ยวชาญด้านการออกแบบและพัฒนาระบบงานในรูปแบบ Microservice



Software

INSTALLATION

การติดตั้ง

MINIKUBE

การติดตั้ง MINIKUBE

- เข้า website <https://minikube.sigs.k8s.io/docs/start/>
- เลือก Download Version ที่ต้องการ

1 Installation

Click on the buttons that describe your target platform. For other architectures, see [the release page](#) for a complete list of minikube binaries.

Operating system

Linux

macOS

Windows

Architecture

x86-64

Release type

Stable

Beta

Installer type

.exe download

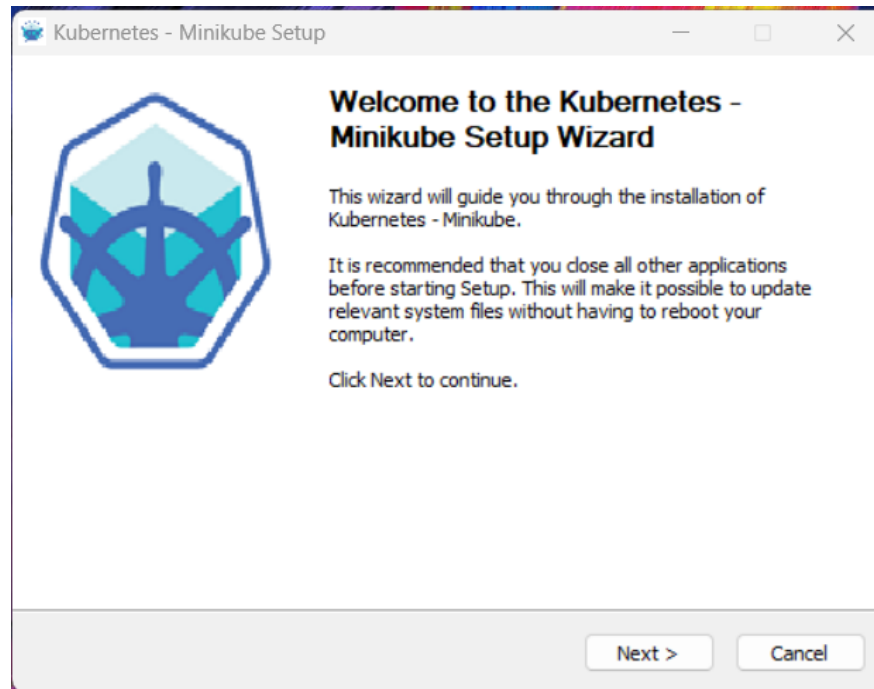
Windows Package Manager

Chocolatey

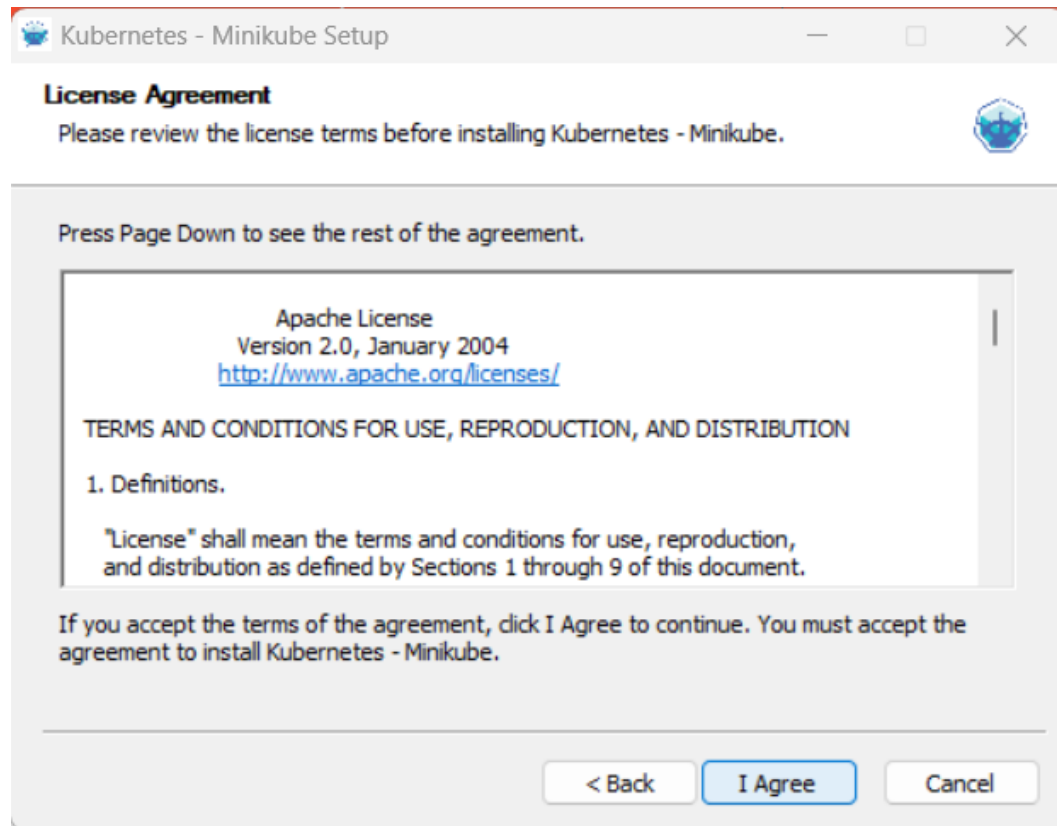
To install the latest minikube **stable** release on **x86-64 Windows** using **.exe download**:

การติดตั้ง MINIKUBE #2

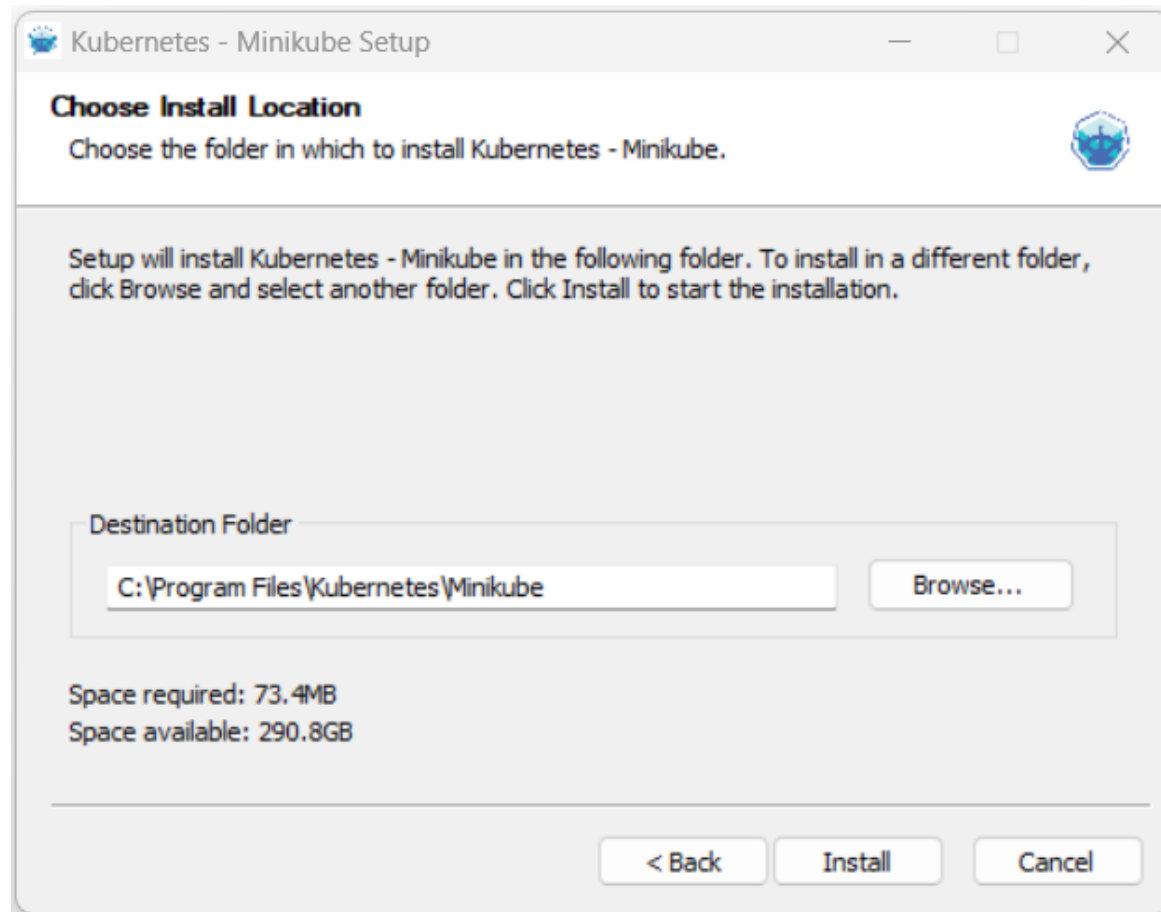
- ติดตั้งโดยใช้สิทธิ์ Administrator ในการติดตั้ง



การติดตั้ง MINIKUBE #3

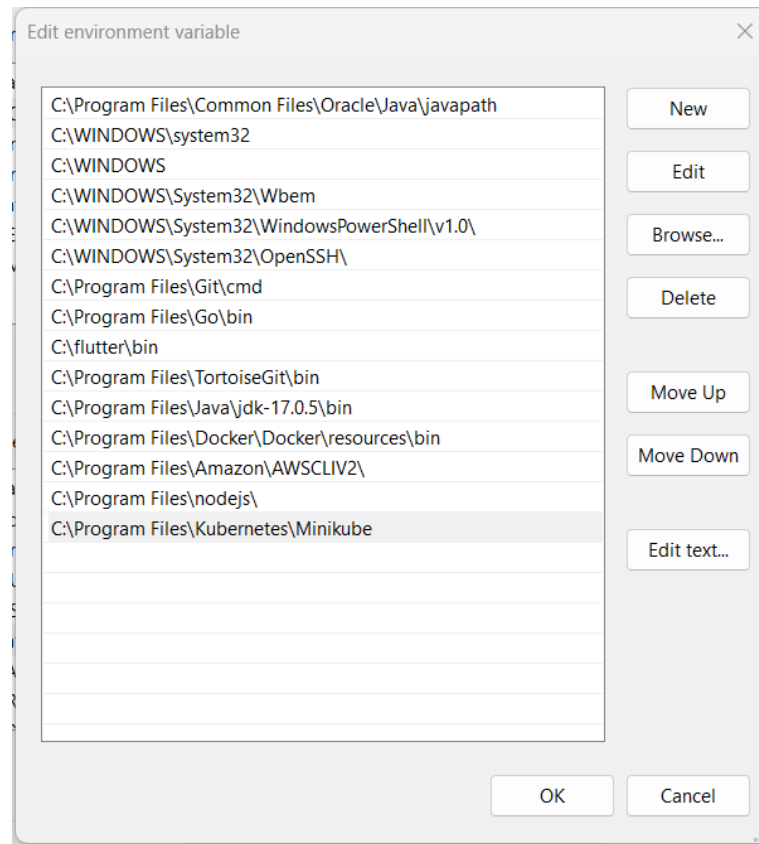


การติดตั้ง MINIKUBE #4



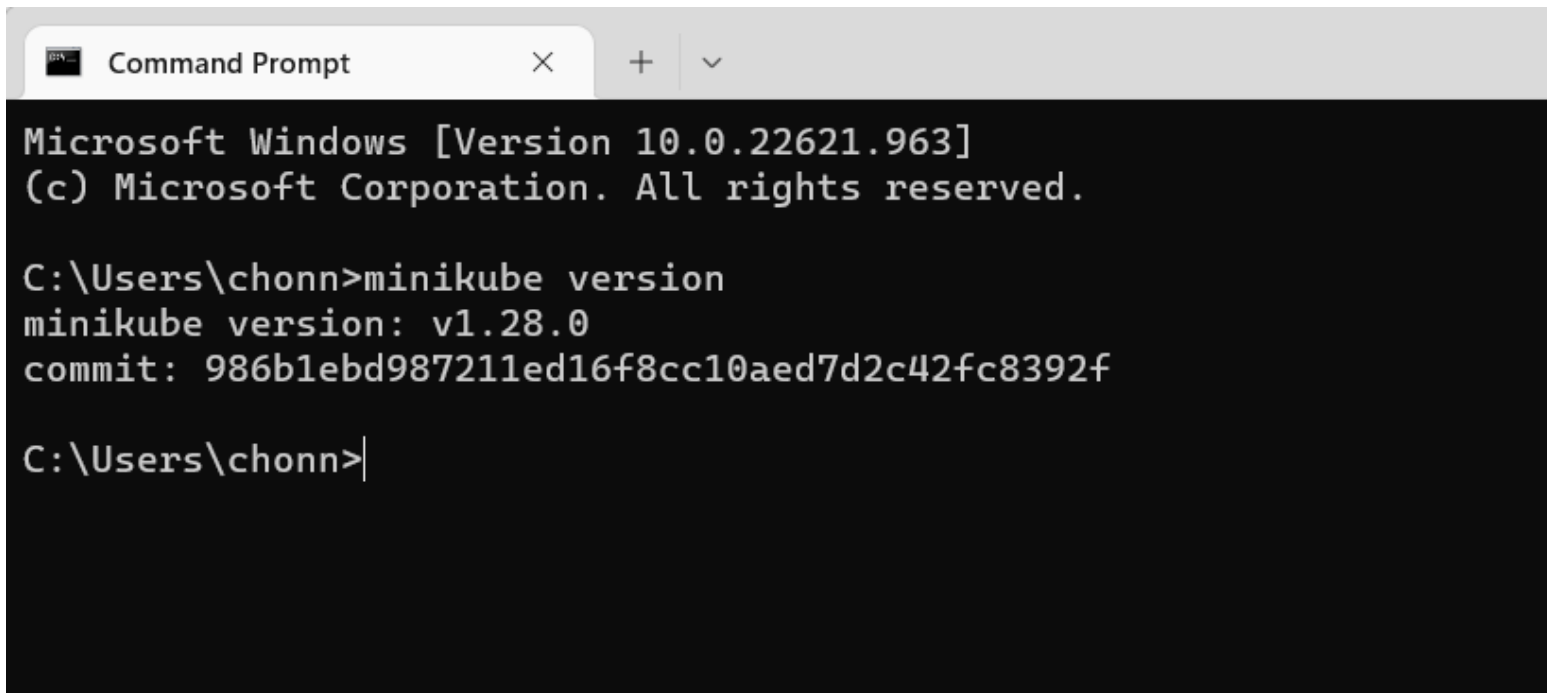
การติดตั้ง MINIKUBE #5

- Add the minikube.exe binary to your PATH



ทดสอบหลังการติดตั้ง MINIKUBE

- เปิดโปรแกรม cmd แล้วพิมพ์คำสั่ง minikube version



```
Microsoft Windows [Version 10.0.22621.963]
(c) Microsoft Corporation. All rights reserved.

C:\Users\chonn>minikube version
minikube version: v1.28.0
commit: 986b1ebd987211ed16f8cc10aed7d2c42fc8392f

C:\Users\chonn>
```

การติดตั้ง

KUBECTL

kubectl

1. Download

```
curl -o kubectl.exe https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.7/2022-10-31/bin/windows/amd64/kubectl.exe
```

2. ตรวจสอบหลังติดตั้งด้วยคำสั่งดังนี้

```
kubectl version --short --client
```

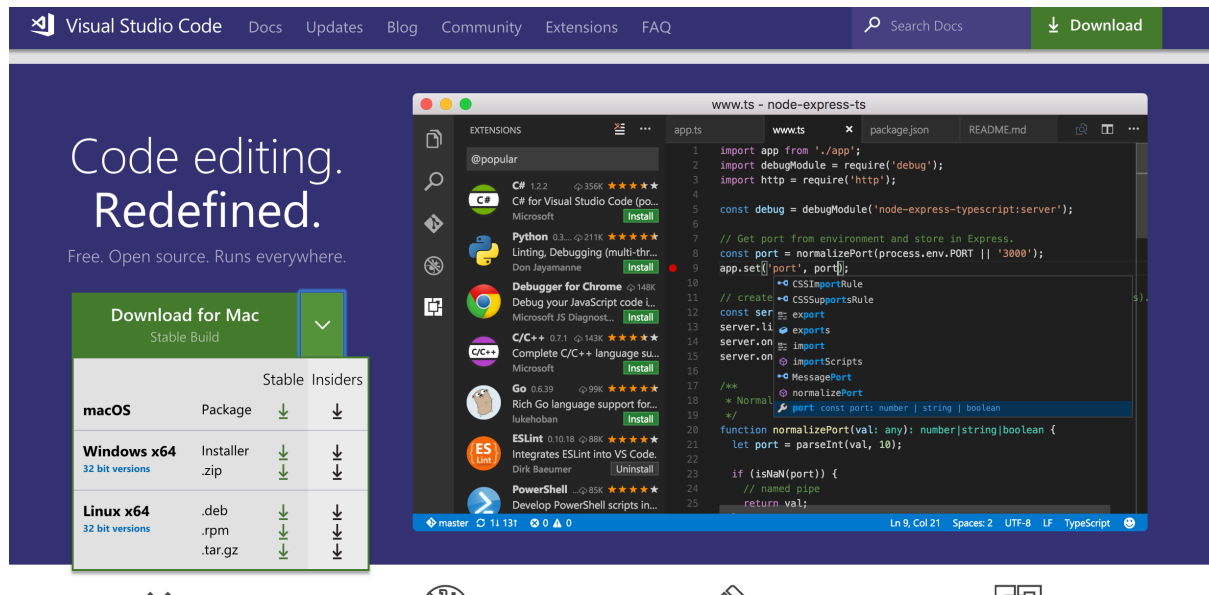
```
Flag --short has been deprecated, and will be removed in the future. The --short output will become the default.  
Client Version: v1.26.0  
Kustomize Version: v4.5.7
```

การติดตั้ง

VISUAL STUDIO CODE

การติดตั้ง VSC

- เข้าไปที่ website <https://code.visualstudio.com/>
- เลือก download สำหรับ windows (stable)



Introduction to containerization

WHAT IS CONTAINERIZATION

What is containerization

Each container that you run is repeatable; the standardization from having dependencies included means that you get the same behavior wherever you run it.

Containers decouple applications from underlying host infrastructure. This makes deployment easier in different cloud or OS environments.

Each node in a Kubernetes cluster runs the containers that form the Pods assigned to that node. Containers in a Pod are co-located and co-scheduled to run on the same node

Introduction to containerization

WHAT IS KUBERNETES

What is Kubernetes

Kubernetes is an open-source system that automates container deployment tasks. It was originally developed at Google but is now maintained as part of the Cloud Native Computing Foundation (CNCF).

Kubernetes has risen to prominence because it solves many of the challenges around using containers in production. It makes it easy to launch limitless container replicas, distribute them across multiple physical hosts, and set up networking so users can reach your service.

Most developers begin their container journey with Docker. While this is a comprehensive tool, it's relatively low-level and relies on CLI commands that interact with one container at a time. Kubernetes provides much higher-level abstractions for defining applications and their infrastructure using declarative schemas you can collaborate on.

Introduction to containerization

KUBERNETES FEATURES

Kubernetes features

Kubernetes has a comprehensive feature set that includes a full spectrum of capabilities for running containers and associated infrastructure:

- **Automated rollouts, scaling, and rollbacks** – Kubernetes automatically creates the specified number of replicas, distributes them onto suitable hardware, and takes action to reschedule your containers if a node goes down. You can instantly scale the number of replicas on-demand or in response to changing conditions such as CPU usage.
- **Service discovery, load balancing, and network ingress** – Kubernetes provides a complete networking solution that covers internal service discovery and public container exposure.
- **Stateless and stateful applications** – While Kubernetes initially focused on stateless containers, it's now also got built-in objects to represent stateful apps too. You can run any kind of application in Kubernetes

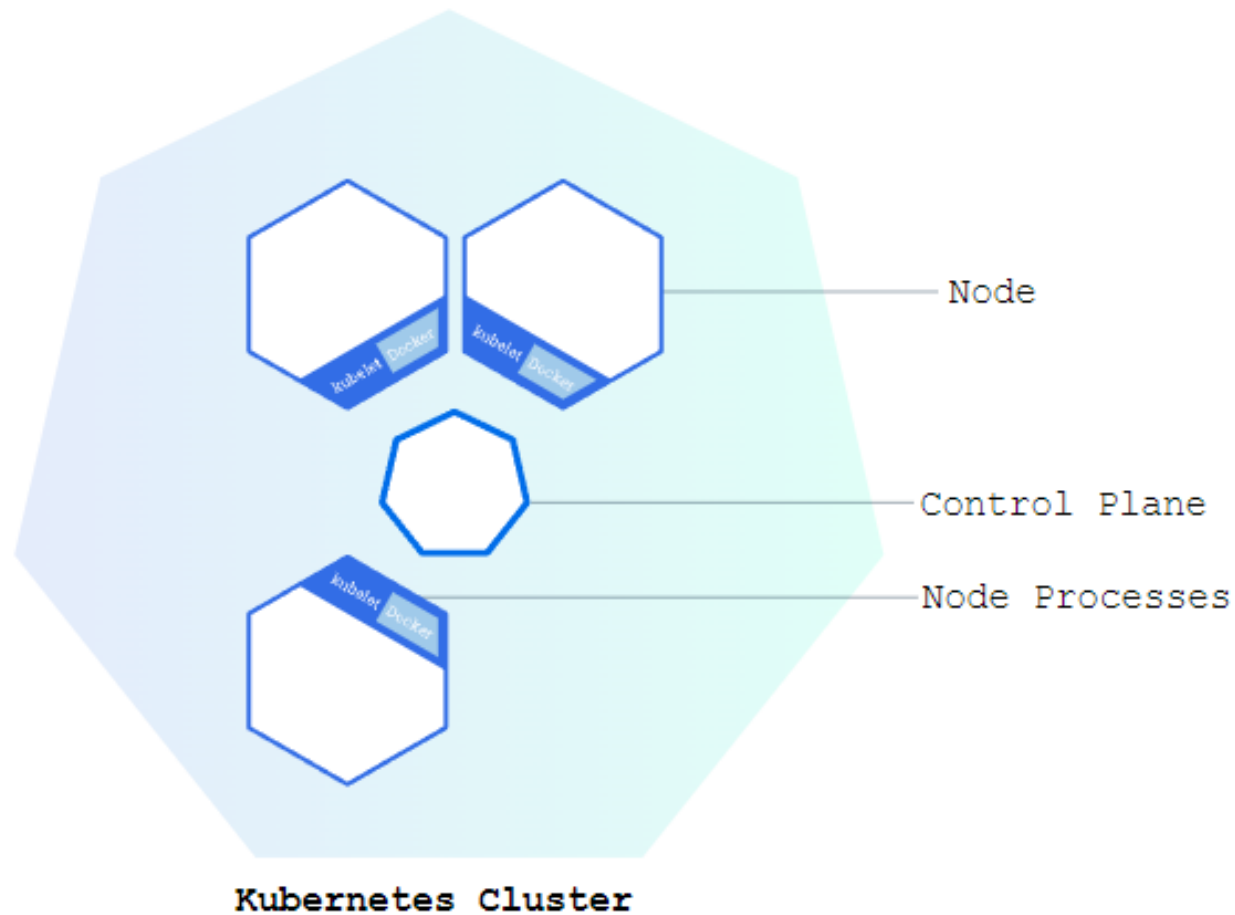
Kubernetes features

- **Storage management** – Persistent storage is abstracted by a consistent interface that works across providers, whether in the cloud, on a network share, or on a local filesystem.
- **Declarative state** – Kubernetes uses object manifests in YAML files to define the state you want to create in your cluster. Applying a manifest instructs Kubernetes to automatically transition the cluster to the target state. You don't have to manually script the changes you want to see.
- **Works across environments** – Kubernetes can be used in the cloud, at the edge, or on your developer workstation. Many different distributions are available to match different use cases. Major cloud providers like AWS and Google Cloud offer managed Kubernetes services, while single-node distributions such as Minikube and K3s are great for local use.
- **Highly extensible** – Kubernetes packs in a lot of functionality, but you can add even more using extensions. You can create custom object types, controllers, and operators to support your own workloads.

Introduction to containerization

ARCHITECTURE OF KUBERNETES CLUSTER

Architecture of Kubernetes cluster



KUBERNETES 101

Nodes and Pods

Nodes

Nodes represent the physical machines that form your Kubernetes cluster. They run the containers you create. Kubernetes tracks the status of your nodes and exposes each one as an object. You used Kubectl to retrieve a list of nodes in the example above.

While your fresh cluster has only one node, Kubernetes advertises support for up to 5,000 nodes. It's theoretically possible to scale even further.

Nodes and Pods

Pods

Pods are the fundamental compute unit in Kubernetes. A Pod is analogous to a container but with some key differences. Pods can contain multiple containers, each of which share a context. The entire Pod will always be scheduled onto the same node. The containers within a Pod are tightly coupled so you should create a new Pod for each distinct part of your application, such as its API and database.

In simple situations, Pods will usually map one-to-one with the containers your application runs. In more advanced cases, Pods can be enhanced with init containers and ephemeral containers to customize startup behavior and provide detailed debugging.

Deployments, jobs and services

Deployments

Deployments wrap ReplicaSets with support for declarative updates and rollbacks. They're a higher level of abstraction that's easier to control.

A Deployment object lets you specify the desired state of a set of Pods. This includes the number of replicas to run. Modifying the Deployment will automatically detect the required changes and scale the ReplicaSet as required. You can pause the rollout or revert to an earlier revision, features that aren't available with plain ReplicaSets.

Deployments, jobs and services

Jobs

A Kubernetes Job is an object that creates a set of Pods and waits for them to terminate. It will retry any failed Pods until a specified number have exited successfully. The Job's then marked as complete.

Jobs provide a mechanism for running ad-hoc tasks inside your cluster. Kubernetes also provides CronJobs that wrap Jobs with cron-like scheduling support. These let you automatically run a job on a regular cadence to accommodate batch activities, backups, and any other scheduled tasks your application requires.

Deployments, jobs and services

Services

Services are used to expose Pods to the network. They allow defined access to Pods either within your cluster or externally.

Ingresses are closely related objects. These are used to set up HTTP routes to services via a load balancer. Ingresses also support HTTPS traffic secured by TLS certificates.

Read more: [Kubernetes Ingress with NGINX Ingress Controller Example](#)

Labels, selectors and namespaces

- Labels are key/value pairs that are attached to objects, such as pods.
- Labels are intended to be used to specify identifying attributes of objects that are meaningful and relevant to users, but do not directly imply semantics to the core system.
- Labels can be used to organize and to select subsets of objects.
- Labels can be attached to objects at creation time and subsequently added and modified at any time.
- Each object can have a set of key/value labels defined.
- Each Key must be unique for a given object.

Labels, selectors and namespaces

Label selector, the client/user can identify a set of objects.

The label selector is the core grouping primitive in Kubernetes.

example:

```
kubectl get pods -l environment=production,tier=frontend
```

Labels, selectors and namespaces

Namespaces

Namespaces isolate different groups of resources. They avoid name collisions by scoping the visibility of your resources.

Creating two objects with the same name is forbidden within the same namespace. If you're in the default namespace, you can't create two Pods that are both called database, for example. Namespaces resolve this by providing logical separation of resources. Two namespaces called app-1 and app-2 could each contain a Pod called database without causing a conflict.

Namespaces are flexible and can be used in many different ways. It's a good idea to create a namespace for each workload in your cluster. You can also use namespaces to divide resources between users and teams by applying role-based access control.

Kubelet and Kube Proxy

Kubelet

Kubelet is a worker process that runs on each of your nodes. It maintains communication with the Kubernetes control plane to receive its instructions.

Kubelet is responsible for pulling container images and starting containers in response to scheduling requests.

Kube-proxy

Proxy is another component found on individual nodes. It configures the host's networking system so traffic can reach the services in your cluster.

Running

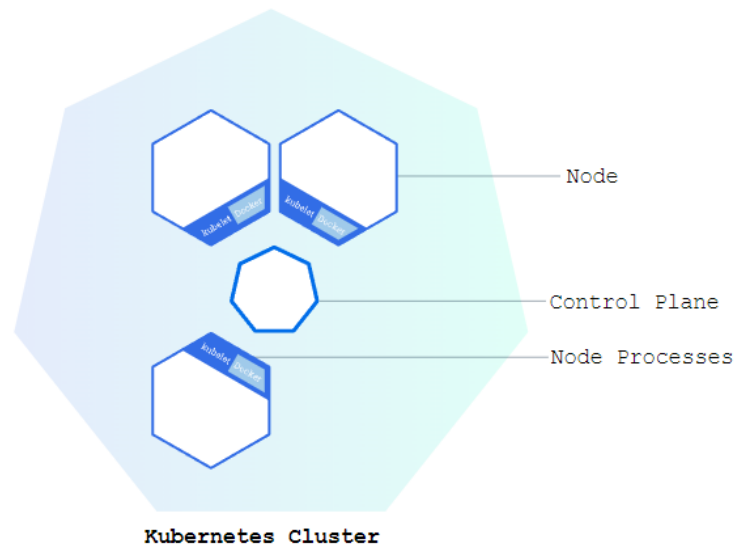
THE HELLO WORLD APPLICATION

Using Minikube to Create a Cluster

A Kubernetes cluster consists of two types of resources:

- The Control Plane coordinates the cluster
- Nodes are the workers that run applications

Cluster Diagram



Create a minikube cluster

Start minikube

- minikube start

```
C:\Windows\System32>minikube start
* minikube v1.28.0 on Microsoft Windows 11 Home Single Language 10.0.22621 Build 22621
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Updating the running docker "minikube" container ...
* Preparing Kubernetes v1.25.3 on Docker 20.10.20 ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Create Deployment

- `kubectl create deployment hello-node --image=nginx`
- `kubectl get deployment`

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
hello-node	1/1	1	1	12s

Create a Service

- minikube tunnel
- `kubectl expose deployment hello-node --type=LoadBalancer --port=8080 --target-port=80`
- `minikube service hello-node`
- `kubectl get svc`
- `kubectl port-forward service/hello-node 8080:8080 --address=0.0.0.0`

View Pods

- `kubectl get pods`

NAME	READY	STATUS	RESTARTS	AGE
hello-node-67949d9db-6x76c	1/1	Running	0	2m20s

Monitor

- `kubectl get events`
- `kubectl config view`

Scaling the Hello World application

```
kubectl scale --replicas=2 deployment hello-node
```

```
kubectl get deployment
```

```
kubectl get pod
```

advance topic in

KUBERNETES

kubectl command list

- kubectl api-resources
- kubectl apply -f ./test.yaml
- kubectl create deployment **nginx** --image=**nginx**
- kubectl expose deployment nginx --type=LoadBalancer --port=80
- kubectl get services
- kubectl get pods
- kubectl get deployments
- kubectl get nodes
- kubectl set image deployment/**frontend** **www**=**image**:v2
- kubectl expose rc nginx --port=80 --target-port=8000
- kubectl autoscale deployment **foo** --min=2 --max=10
- kubectl get hpa
- kubectl scale --replicas=3 -f ./test.yaml
- kubectl delete pod,service baz foo

Deployment

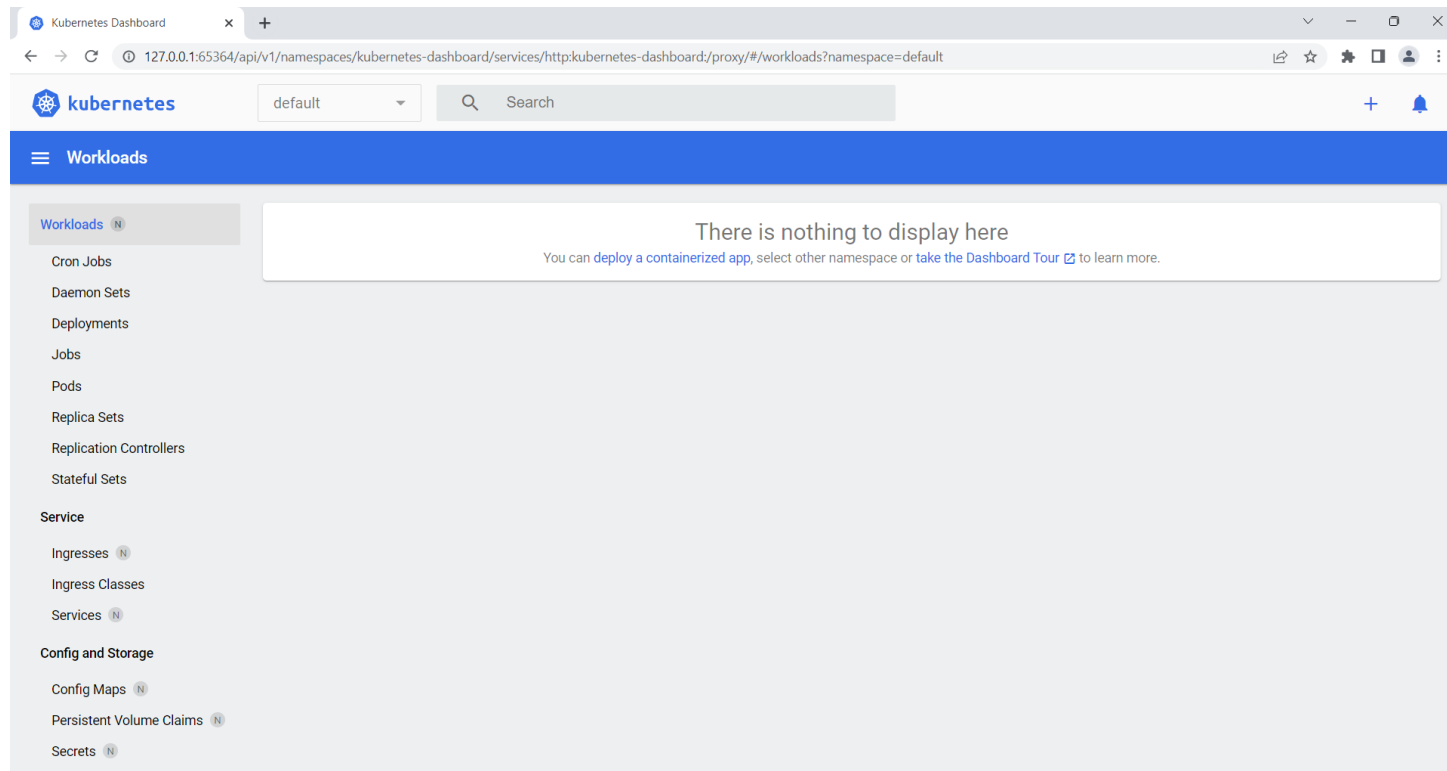
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world-server
          image: nginx:alpine
          ports:
            - containerPort: 80
```

Service

```
apiVersion: v1
kind: Service
metadata:
  name: helloworld
  labels:
    app: hello-world
spec:
  selector:
    app: hello-world
  ports:
    - name: http
      protocol: TCP
      port: 8080
      targetPort: 80
  type: ClusterIP
```

Kube dashboard

- minikube addons enable metrics-server
- minikube dashboard



Configuration data

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-config
data:
  root-password: "1234"
  database-name: "simple_db"
  user: "usr"
  password: "usr"
```

Deployment with config

```
env:
- name: MYSQL_ROOT_PASSWORD
  valueFrom:
    configMapKeyRef:
      name: mysql-config
      key: root-password
- name: MYSQL_DATABASE
  valueFrom:
    configMapKeyRef:
      name: mysql-config
      key: database-name
- name: MYSQL_USER
  valueFrom:
    configMapKeyRef:
      name: mysql-config
      key: user
- name: MYSQL_PASSWORD
  valueFrom:
    configMapKeyRef:
      name: mysql-config
      key: password
```


Dealing with application secret

```
kubectl create secret generic regcred  
  --from-file=.dockerconfigjson=<path/to/.docker/config.json>  
  --type=kubernetes.io/dockerconfigjson
```

OR

```
kubectl create secret docker-registry regcred  
  --docker-server=<your-registry-server>  
  --docker-username=<your-name>  
  --docker-password=<your-pword>  
  --docker-email=<your-email>
```

```
kubectl get secret
```

Dealing with application secret

```
spec:
  containers:
    - name: demo-api-server
      image: registry-sommaik.cloudystore.com/sommaik/demoapi
      ports:
        - containerPort: 3000
      resources:
        limits:
          memory: "128Mi"
          cpu: "500m"
      imagePullSecrets:
        - name: regcred
```

Any questions?

