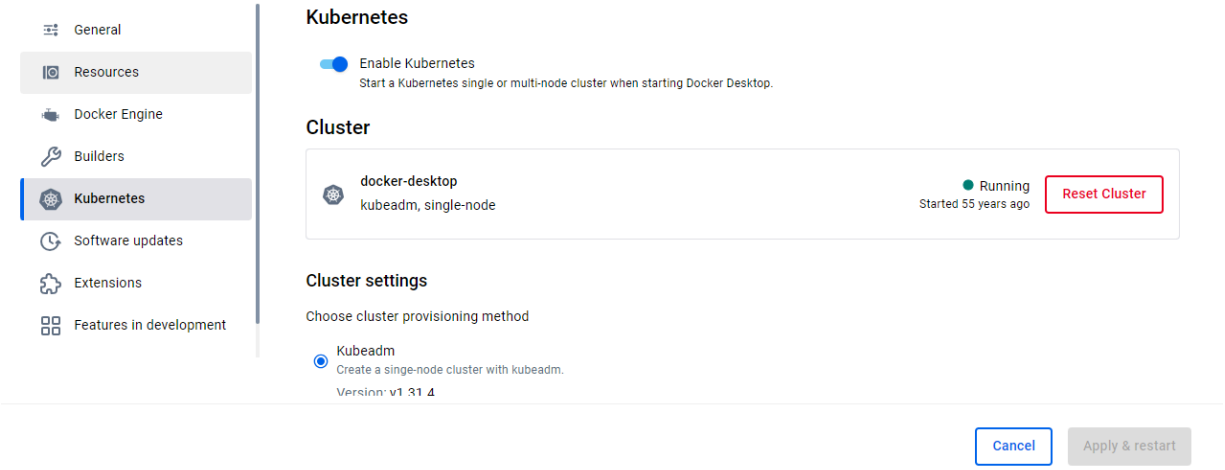
Kubernetes

Container Orchestration Tool which is helping to manage the containers, scaling, auto healing etc..
How To set up kubernetes.



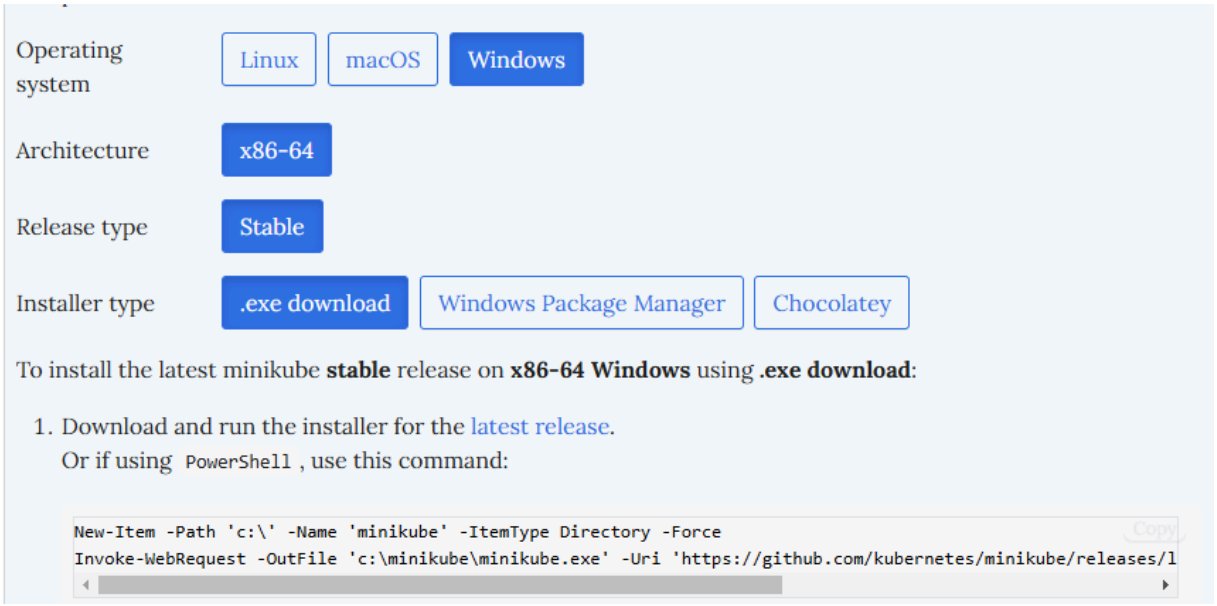
Enable Kubernetes and Apply and Restart.
Once the kubernetes service started you can verify using below commands.

```
C:\Users\NEW>kubectl version
Client Version: v1.31.4
Kustomize Version: v5.4.2
Server Version: v1.31.4

C:\Users\NEW>kubectl cluster-info
Kubernetes control plane is running at https://kubernetes.docker.internal:6443
CoreDNS is running at https://kubernetes.docker.internal:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

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

Minikube ?
Kubernetes by default provide one cluster.
If you want to setup minikube cluster to your local system you can download minikube first and install.



Click on latest Release and install.

```
C:\Users\NEW>minikube version
minikube version: v1.35.0
commit: dd5d320e41b5451cdf3c01891bc4e13d189586ed-dirty
```

To start Minikube Cluster.

```
C:\Users\NEW>minikube start
* minikube v1.35.0 on Microsoft Windows 11 Pro 10.0.26100.3194 Build 26100.3194
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.46 ...
* Restarting existing docker container for "minikube" ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/re
ference/networking/proxy/
* Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

```
C:\Users\NEW>kubectl version
Client Version: v1.31.4
Kustomize Version: v5.4.2
Server Version: v1.32.0

C:\Users\NEW>kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:52755
CoreDNS is running at https://127.0.0.1:52755/api/v1/namespaces
y

To further debug and diagnose cluster problems, use 'kubectl cl

C:\Users\NEW>kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   8d    v1.32.0
```

Create Pod.yml file

```
apiVersion: v1
kind: Pod
metadata:
name: nginx #name of Pod
labels:
app: nginx # use this label to expose on service
spec:
containers:
- name: nginx
image: nginx:1.14.2
ports:
- containerPort: 80
```

```
rnetes>kubectl apply -f pod.yml
pod/nginx created
```

Verify pods:

```
C:\Users\NEW>kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          64s
```

Describe the pod details

```
C:\Users\NEW>kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Tue, 25 Feb 2025 11:48:12 +0530
Labels:           app=nginx
Annotations:      <none>
```

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  91s   default-scheduler  Successfully assigned default/nginx to minikube
  Normal  Pulling    90s   kubelet            Pulling image "nginx:1.14.2"
  Normal  Pulled     71s   kubelet            Successfully pulled image "nginx:1.14.2" in 19.192s
192s including waiting). Image size: 109129446 bytes.
  Normal  Created    71s   kubelet            Created container: nginx
  Normal  Started    70s   kubelet            Started container nginx
```

You can't access direct pod in browser.

For that we required Service to expose the same in browser. Create service.yml

```
apiVersion: v1
kind: Service
metadata:
name: my-service
spec:
type: NodePort # Service Type
selector:
app: nginx # label of your Pod
ports:
- protocol: TCP
port: 80 #Host Port
targetPort: 80 # Container Port
nodePort: 30007 # Service Port
```

```
D:\SFJ Solutions\Mindsprint-Foundatio
rnetes>kubectl apply -f service.yml
service/my-service created
```

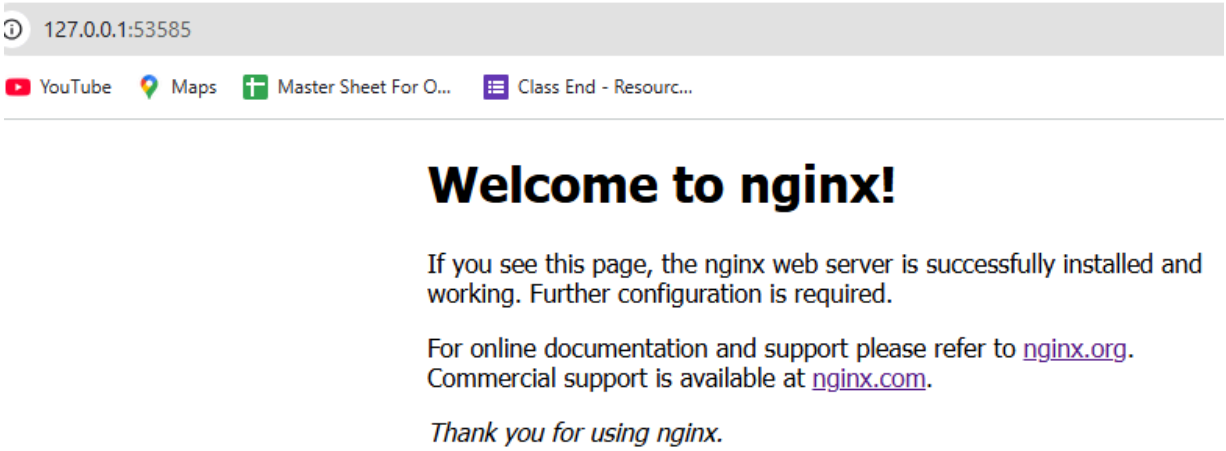
```
C:\Users\NEW>kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        8d
my-service   NodePort    10.108.16.131   <none>        80:30007/TCP   12m
```

```
C:\Users\NEW>kubectl describe service my-service
Name:                     my-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=nginx
```

Let's Access in browser using minikube service

```
C:\Users\NEW>minikube service my-service
|-----------|------------|-------------|--------------------------|
| NAMESPACE |    NAME    | TARGET PORT |            URL           |
|-----------|------------|-------------|--------------------------|
| default   | my-service |          80 | http://192.168.49.2:30007 |
|-----------|------------|-------------|--------------------------|
* Starting tunnel for service my-service.
|-----------|------------|-------------|------------------------|
| NAMESPACE |    NAME    | TARGET PORT |          URL           |
|-----------|------------|-------------|------------------------|
| default   | my-service |             | http://127.0.0.1:53585 |
|-----------|------------|-------------|------------------------|
* Opening service default/my-service in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

You can see the out put in Browser.



Deployment:
Deployment Features: self healing, rolling out and rollback feature, auto scaling and more updates to file.
create deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx-deployment
labels:
app: nginx
spec:
replicas: 3
selector:
matchLabels:
app: nginx
template:
metadata:
labels:
app: nginx
spec:
containers:
- name: nginx
image: nginx:1.14.2
ports:
- containerPort: 80
```

Execute Command:

```
rnetes>kubectl apply -f deployment.yml
deployment.apps/nginx-deployment created
```

```
C:\Users\NEW>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-647677fc66-4glgq   1/1     Running   0          97s
nginx-deployment-647677fc66-9p84z   1/1     Running   0          13s
nginx-deployment-647677fc66-jccjf   1/1     Running   0          94s
```

To expose this as service you require only one service. You can use type as LoadBalancer and it will manage the traffic between 3 replicas.
Let's understand Roll out:
kubectl set image deployment.v1.apps/nginx-deployment nginx=nginx:1.16.1

```
C:\Users\NEW>kubectl set image deployment.v1.apps/nginx-deployment nginx=nginx:1.16.1
deployment.apps/nginx-deployment image updated

C:\Users\NEW>kubectl rollout status deployment/nginx-deployment
deployment "nginx-deployment" successfully rolled out
```

You can also check the entire description

kubect describe deployment nginx-deployment (check Image version)

RollBack:

```
C:\Users\NEW>kubectl rollout undo deployment/nginx-deployment
deployment.apps/nginx-deployment rolled back
```

kubect describe deployment nginx-deployment (check Image version)

Clean up resources:

```
C:\Users\NEW>kubectl delete deployment nginx-deployment
deployment.apps "nginx-deployment" deleted
```

Stop minikube service.

```
C:\Users\NEW>minikube stop
* Stopping node "minikube"  ...
* Powering off "minikube" via SSH ...
* 1 node stopped.
```