STEP 1 Start Minikube & Enable Ingress

minikube start
minikube addons enable ingress

Verify:

kubectl get pods -n ingress-nginx

```
ubuntu@ip-172-31-15-140:~$ minikube start
😄  minikube v1.37.0 on Ubuntu 24.04
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.48 ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
🔎  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
ubuntu@ip-172-31-15-140:~$ minikube addons enable ingress
💡  ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image registry.k8s.io/ingress-nginx/controller:v1.13.2
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.6.2
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.6.2
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
ubuntu@ip-172-31-15-140:~$ kubectl get pods -n ingress-nginx
NAME                                        READY   STATUS      RESTARTS   AGE
ingress-nginx-admission-create-dzd9c        0/1     Completed   0          98s
ingress-nginx-admission-patch-v2n4m         0/1     Completed   1          98s
ingress-nginx-controller-9cc49f96f-xhzpc    1/1     Running     0          98s
ubuntu@ip-172-31-15-140:~$
```

STEP 2 Create Custom Nginx Microservices

STEP 6 Apply All
kubectl apply -f .

kubectl get pods
kubectl get svc
kubectl get ingress

minikube ip

```
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$ kubectl apply -f .
ingress.networking.k8s.io/microservices-ingress created
configmap/ms1-html created
deployment.apps/microservice-1 created
service/microservice-1 created
configmap/ms2-html created
deployment.apps/microservice-2 created
service/microservice-2 created
configmap/ms3-html created
deployment.apps/microservice-3 created
service/microservice-3 created
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$ kubectl get pods
kubectl get svc
kubectl get ingress
NAME                            READY   STATUS    RESTARTS   AGE
microservice-1-d94466577-cnrhc   1/1     Running   0          18s
microservice-2-f859c57bf-gp95r   0/1     Running   0          18s
microservice-3-64465dc89c-69lx8   0/1     Running   0          17s
NAME             TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes       ClusterIP   10.96.0.1        <none>        443/TCP   16h
microservice-1   ClusterIP   10.103.237.229   <none>        80/TCP    18s
microservice-2   ClusterIP   10.106.103.92    <none>        80/TCP    18s
microservice-3   ClusterIP   10.99.253.152    <none>        80/TCP    17s
NAME                    CLASS   HOSTS                                                  ADDRESS        PORTS   AGE
microservices-ingress   nginx   service1.local.com,service2.local.com,service3.local.com   192.168.49.2   80      18s
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$
```

```
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$ kubectl get endpoints
Warning: v1 Endpoints is deprecated in v1.33+; use discovery.k8s.io/v1 EndpointSlice
NAME             ENDPOINTS           AGE
kubernetes       192.168.49.2:8443   16h
microservice-1   10.244.0.8:80       19m
microservice-2   10.244.0.9:80       19m
microservice-3   10.244.0.10:80      19m
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$ kubectl describe ingress microservices-ingress
Name:             microservices-ingress
Labels:           <none>
Namespace:        default
Address:          192.168.49.2
Ingress Class:    nginx
Default backend:  <default>
Rules:
  Host               Path  Backends
  ----               ----  --------
  service1.local.com
                     /   microservice-1:80 (10.244.0.8:80)
  service2.local.com
                     /   microservice-2:80 (10.244.0.9:80)
  service3.local.com
                     /   microservice-3:80 (10.244.0.10:80)
Annotations:          <none>
Events:
  Type    Reason  Age                 From                      Message
  ----    ------  ----                ----                      -------
  Normal  Sync    20m (x2 over 20m)   nginx-ingress-controller  Scheduled for sync
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$
```

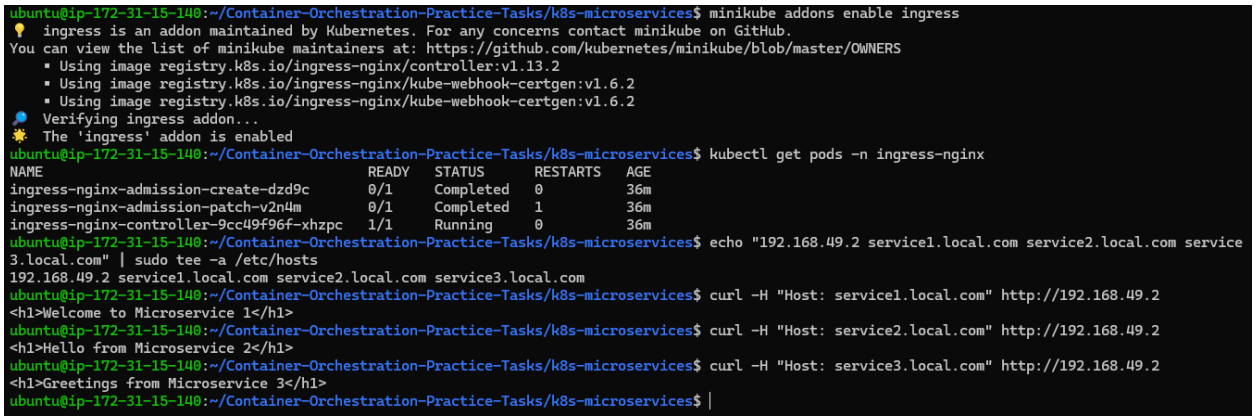
Test via Ingress (from outside the cluster)

Enable ingress addon in minikube
minikube addons enable ingress

Verify ingress controller is running
kubectl get pods -n ingress-nginx

Update /etc/hosts to point to minikube IP
echo "192.168.49.2 service1.local.com service2.local.com service3.local.com" | sudo tee -a /etc/hosts

Test the ingress
curl -H "Host: service1.local.com" http://192.168.49.2
curl -H "Host: service2.local.com" http://192.168.49.2
curl -H "Host: service3.local.com" http://192.168.49.2



Alternative method without modifying /etc/hosts:
Using curl with Host header
curl -H "Host: service1.local.com" http://$(minikube ip)
curl -H "Host: service2.local.com" http://$(minikube ip)
curl -H "Host: service3.local.com" http://$(minikube ip)

Health Checks

Detailed view of pods
kubectl get pods -o wide

Describe a specific pod
kubectl describe pod microservice-1-d94466577-cnrhc

Check pod logs
kubectl logs microservice-1-d94466577-cnrhc
kubectl logs microservice-2-f859c57bf-gp95r
kubectl logs microservice-3-64465dc89c-69lx8

Follow logs in real-time
kubectl logs -f microservice-1-d94466577-cnrhc

```
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$ kubectl get pods -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP            NODE       NOMINATED NODE   READINESS GATES
microservice-1-d94466577-cnrhc    1/1     Running   0          30m   10.244.0.8    minikube   <none>           <none>
microservice-2-f859c57bf-gp95r    1/1     Running   0          30m   10.244.0.9    minikube   <none>           <none>
microservice-3-64465dc89c-69lx8   1/1     Running   0          30m   10.244.0.10   minikube   <none>           <none>
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$ kubectl describe pod microservice-1-d94466577-cnrhc
Name:             microservice-1-d94466577-cnrhc
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Sat, 06 Dec 2025 06:54:59 +0000
Labels:           app=microservice-1
                  pod-template-hash=d94466577
Annotations:      <none>
Status:           Running
IP:               10.244.0.8
IPs:
  IP:           10.244.0.8
Controlled By:  ReplicaSet/microservice-1-d94466577
Containers:
  nginx:
    Container ID:   docker://11c22c064e94fe16d16246758412168e29906ece4f24f60d7790238d3316666e
    Image:          nginx:alpine
    Image ID:       docker-pullable://nginx@sha256:b3c656d55d7ad751196f21b7fd2e8d4da9cb430e32f646adcf92441b72f82b14
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 06 Dec 2025 06:55:04 +0000
    Ready:          True
    Restart Count:  0
    Readiness:      http-get http://:80/ delay=5s timeout=1s period=10s #success=1 #failure=3
    Environment:    <none>
    Mounts:
      /usr/share/nginx/html from html (rw)
```

```
    Type:      ConfigMap (a volume populated by a ConfigMap)
    Name:      ms1-html
    Optional:  false
  kube-api-access-ks26s:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    Optional:                false
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  30m   default-scheduler  Successfully assigned default/microservice-1-d94466577-cnrhc to minikube
  Normal  Pulling    30m   kubelet            Pulling image "nginx:alpine"
  Normal  Pulled     30m   kubelet            Successfully pulled image "nginx:alpine" in 3.483s (3.483s including waiting). Image size: 52840371 by
tes.
  Normal  Created    30m   kubelet            Created container: nginx
  Normal  Started    30m   kubelet            Started container nginx
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$ kubectl logs microservice-1-d94466577-cnrhc
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/12/06 06:55:04 [notice] 1#1: using the "epoll" event method
2025/12/06 06:55:04 [notice] 1#1: nginx/1.29.3
2025/12/06 06:55:04 [notice] 1#1: built by gcc 14.2.0 (Alpine 14.2.0)
```

Check service endpoints

Verify services are pointing to correct pods
kubectl get endpoints

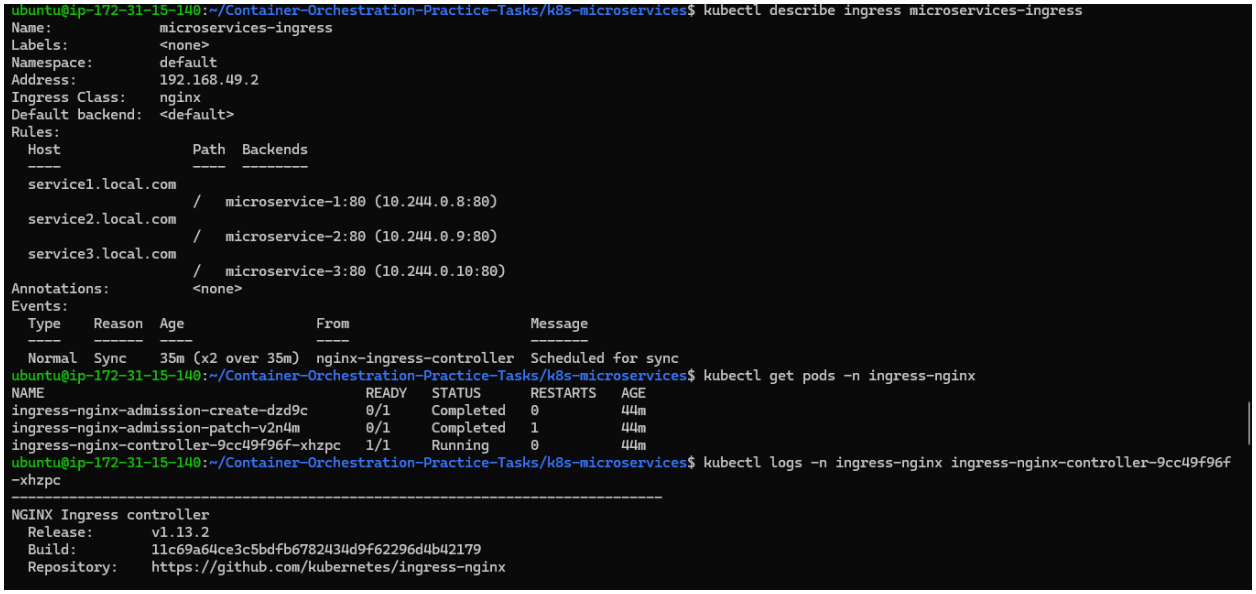
Describe services
kubectl describe svc microservice-1
kubectl describe svc microservice-2
kubectl describe svc microservice-3

```
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$ kubectl get endpoints
Warning: v1 Endpoints is deprecated in v1.33+; use discovery.k8s.io/v1 EndpointSlice
NAME            ENDPOINTS           AGE
kubernetes      192.168.49.2:8443   16h
microservice-1  10.244.0.8:80       34m
microservice-2  10.244.0.9:80       34m
microservice-3  10.244.0.10:80      34m
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$ kubectl describe svc microservice-1
Name:                     microservice-1
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=microservice-1
Type:                     ClusterIP
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.103.237.229
IPs:                      10.103.237.229
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
Endpoints:                10.244.0.8:80
Session Affinity:         None
Internal Traffic Policy:  Cluster
Events:                   <none>
ubuntu@ip-172-31-15-140:~/Container-Orchestration-Practice-Tasks/k8s-microservices$
```

Check ingress details

Describe ingress to see routing rules
kubectl describe ingress microservices-ingress

Check ingress controller logs
kubectl get pods -n ingress-nginx
kubectl logs -n ingress-nginx <ingress-controller-pod-name>



Quick Diagnostic Commands

```
# One-liner to check everything
kubectl get all

# Check events (useful for troubleshooting)
kubectl get events --sort-by='.lastTimestamp'

# Check resource usage
kubectl top pods
kubectl top nodes

# Check if services have endpoints
kubectl get endpointslices

# Check pod resource limits
kubectl get pods -o jsonpath='{range
.items[*]}{.metadata.name}{"\t"}{.spec.containers[*].resources}{"\n"}{end}'
```

Testing with Minikube Dashboard

```
# Open minikube dashboard in browser
minikube dashboard

# Or get dashboard URL
minikube dashboard --url
```

Debugging Tips

```
# Check if pods are actually running the application
kubectl exec microservice-1-d94466577-cnrhc -- ps aux

# Check container environment
kubectl exec microservice-1-d94466577-cnrhc -- env

# Test from inside the pod
kubectl exec microservice-1-d94466577-cnrhc -- curl -s localhost

# Check network policies
kubectl get networkpolicies
```