What is Minikube?

Minikube is an **open-source tool** that allows running a **single-node Kubernetes cluster** locally on a machine. It is ideal for developers and learners who want to **experiment with Kubernetes** without setting up a complex multi-node cluster on the cloud.

Purpose of Minikube

- Minikube is mainly used for **learning Kubernetes concepts** and **testing applications** locally.
- It helps **developers deploy, manage, and debug** Kubernetes applications in a local environment.

Why Use Minikube?

1. Ease of Setup

Minikube **simplifies** Kubernetes installation by setting up a **lightweight virtual machine or container** that contains all required Kubernetes components.

2. Features of Minikube

- ✅ **Supports Kubernetes add-ons** (e.g., `ingress`, `metrics-server`, and `dashboard`).
- ✅ **Multi-cluster support** for testing multiple Kubernetes clusters at the same time.
- ✅ **Built-in Docker daemon**, eliminating the need for a separate Docker installation.
- ✅ **Configurable resource limits** (CPU and memory) to optimize performance.

3. Cross-Platform Compatibility

Minikube works on:

- **Windows**

- **macOS**
- **Linux**

4. Use Cases

- 📌 Learning **Kubernetes basics** in a local environment.
- 📌 **Testing CI/CD pipelines** and Kubernetes deployments.
- 📌 **Debugging Kubernetes issues** locally before deploying to production.

5. Integration

Minikube integrates with the Kubernetes CLI tool **kubectl**, allowing users to interact with Kubernetes clusters easily.

Minikube Installation Guide

Step 1: Download Minikube

Run the following command to download Minikube for **Linux**:

```
bash
CopyEdit
curl -LO
https://github.com/kubernetes/minikube/releases/latest/download/minikube-linux-amd64
```

📌 **Explanation:**

- `curl -LO`: Downloads the latest Minikube binary from the official **GitHub** repository.

Step 2: Install Minikube

```
bash
```

```
CopyEdit
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm
minikube-linux-amd64
```

📌 **Explanation:**

- `sudo install minikube-linux-amd64 /usr/local/bin/minikube`: Installs Minikube globally.
- `rm minikube-linux-amd64`: Deletes the downloaded file after installation.

🔴 **Common Error:**

- If you see `rm: cannot remove 'minikube-linux-': No such file or directory`, it means the filename is **incorrectly split**.
  ✅ **Fix:** Remove the extra space in `minikube-linux- amd64` and use the correct filename `minikube-linux-amd64`.

Step 3: Start Minikube

```
bash
CopyEdit
minikube start
```

📌 **Explanation:**

- Starts the Minikube **Kubernetes cluster**.

Step 4: Check Minikube Status

```
bash
CopyEdit
minikube status
```

📌 **Expected Output:**

```
plaintext
CopyEdit
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

📌 **Explanation:**

- Ensures **Minikube is running properly**.

Step 5: Get Running Pods

```
bash
CopyEdit
kubectl get pod
```

📌 **Explanation:**

- Displays a list of **all running pods**.
- If no pods exist, the output will be:

```
plaintext
CopyEdit
No resources found in default namespace.
```

Step 6: Get Deployments

```
bash
CopyEdit
```

```
kubectl get deploy
```

📌 **Explanation:**

- Lists all **Kubernetes deployments** in the default namespace.
- If no deployments exist, the output will be:

```
plaintext
CopyEdit
No resources found in default namespace.
```

Step 7: Get ReplicaSets

```
bash
CopyEdit
kubectl get replicaset
```

📌 **Explanation:**

- Lists all **ReplicaSets** that manage pod replicas.
- If no ReplicaSets exist, the output will be:

```
plaintext
CopyEdit
No resources found in default namespace.
```

❌ **Common Error:**

```
plaintext
CopyEdit
error: the server doesn't have a resource type "replica"
```

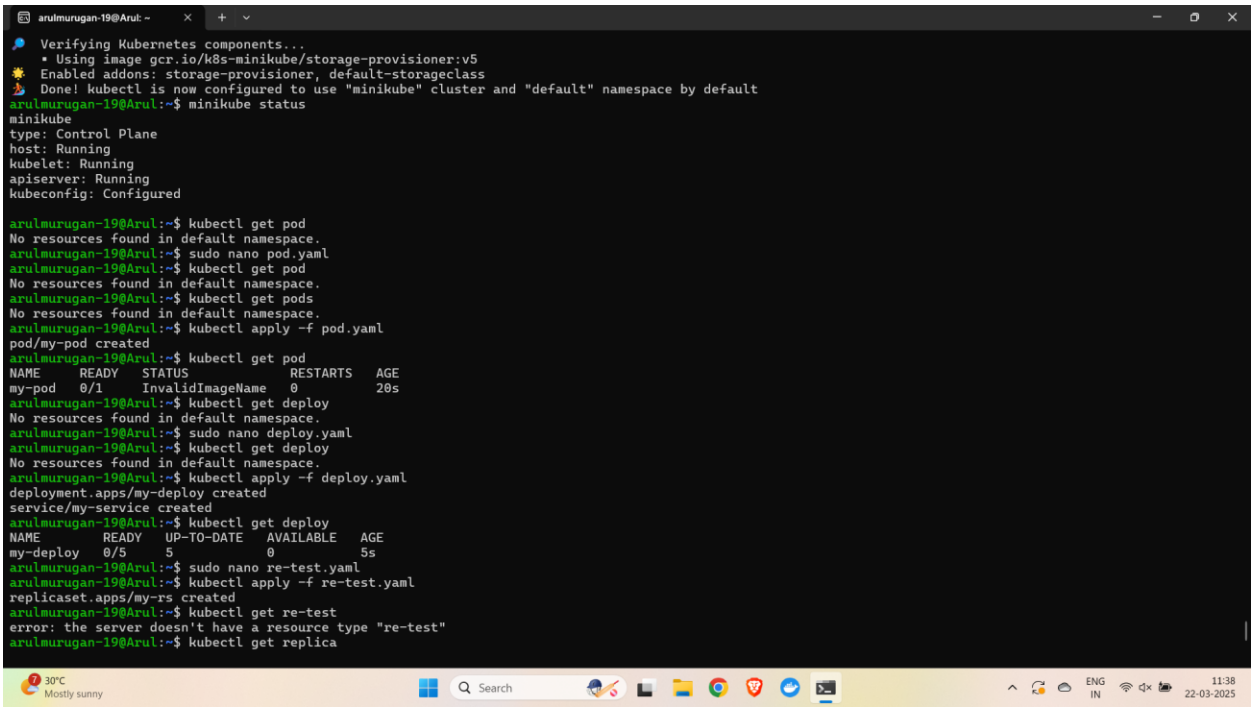
✅ **Fix:** Use `kubectl get replicaset` or the short form `kubectl get rs`.

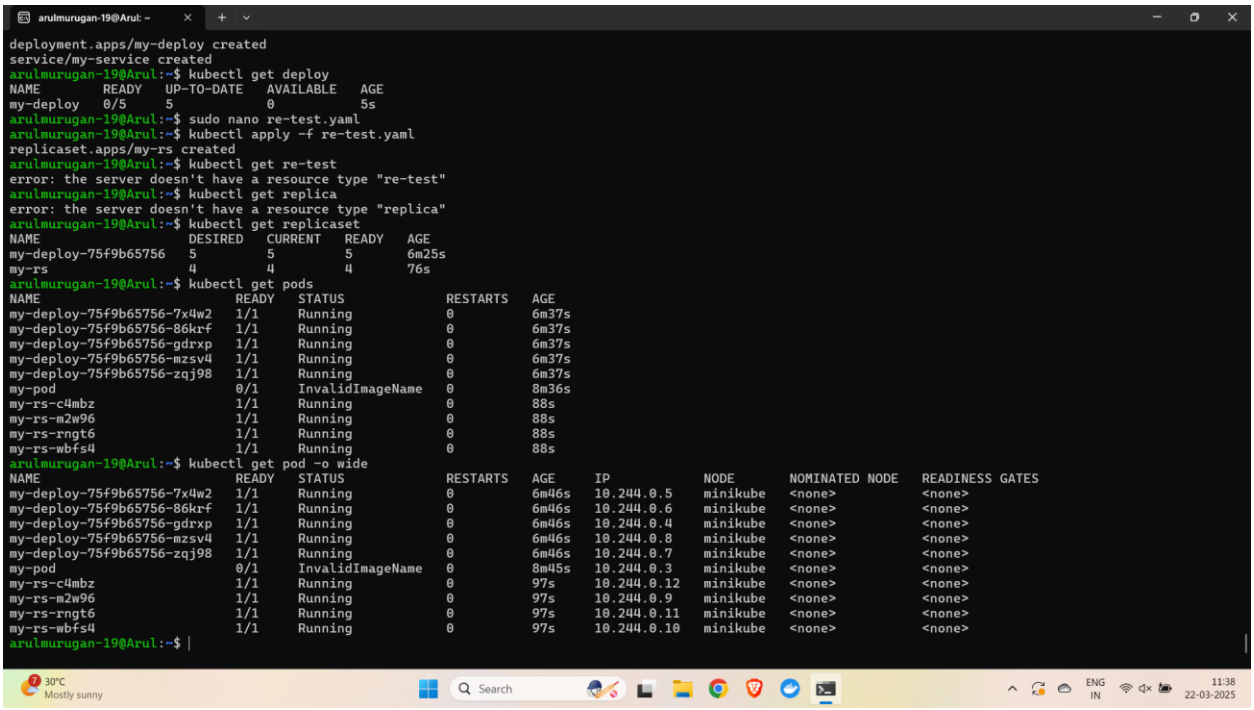
Step 8: Get Detailed Pod Information

```
bash
CopyEdit
kubectl get pod -o wide
```

📌 **Explanation:**

- Shows **detailed information** about running pods, including:
  - Node name
  - IP address
  - Status
  - Restarts





Pod.yaml

apiVersion: v1

kind: Pod

```
metadata:
  name: my-pod
  labels:
     app: my-web-app
     type: backend
spec:
  containers:
   - name: nginx-container
     image: nginx //our image name
     ports:
       - containerPort: 80
```

Deploy.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deploy
  labels:
    name: my-deploy
spec:
  replicas: 5
  selector:
    matchLabels:
      apptype: web-backend
  strategy:
    type: RollingUpdate
```

```
 template:
  metadata:
   labels:
    apptype: web-backend
  spec:
   containers:
   - name: my-app
    image: 22csr019/devops_dockerhub_1
    ports:
    - containerPort: 9000


---


apiVersion: v1
kind: Service
metadata:
 name: my-service
 labels:
  app: my-service
spec:
 type: NodePort
 ports:
  - port: 9000
   targetPort: 8080
   nodePort: 30002
 selector:
```

```
    apptype: web-backend
```

Re-test.yaml

```
apiVersion: apps/v1  # Add this line
kind: ReplicaSet
metadata:
  name: my-rs
  labels:
    name: my-rs
spec:
  replicas: 4
  selector:
    matchLabels:
      apptype: web-backend
  template:
    metadata:
      labels:
        apptype: web-backend
    spec:
      containers:
      - name: my-app
        image: 22csr019/devops_dockerhub_1
        ports:
          - containerPort: 9090
```