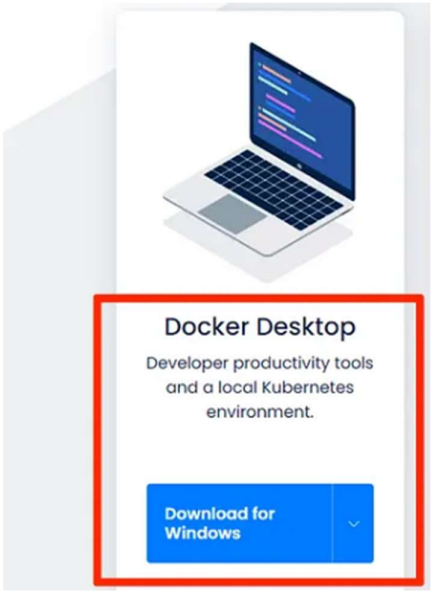
**Install Docker on Windows**

## Step 1: Downloading Docker

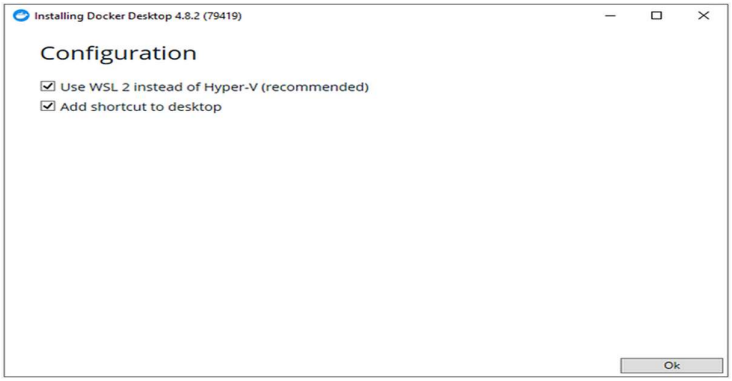

The first place to start is the official Docker website from where we can download Docker Desktop.

Please note that Docker Desktop is intended only for Windows 10/11 and not for Windows Server.



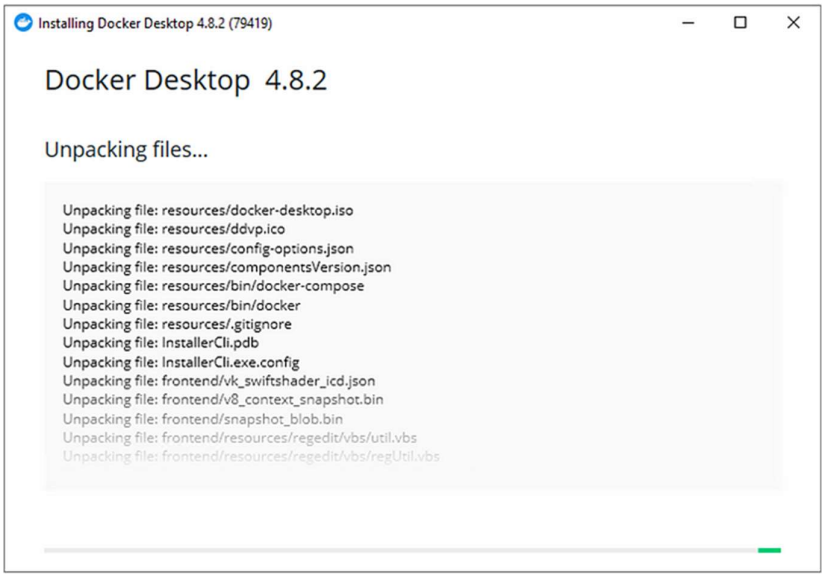
**Step 2: Configuration**

To run Linux on Windows, Docker requires a virtualization engine. Docker recommends using WSL 2.




**Step 3: Running the installation**

Click Ok, and wait a bit…



### Step 4: Restart

For Docker to be able to properly register with Windows, a restart is required at this point.

### Step 5: License agreement

After the restart, Docker will start automatically and you should see the window below:

**Kubernetes Command**

1. Start Minikube

**Syntax:** minikube start

**Explanation**: This command starts a local Kubernetes cluster using Minikube. Minikube is a tool that makes it easy to run Kubernetes clusters locally for development and testing purposes.

2. Stop Minikube

**Syntax:** minikube stop

**Explanation:** This command stops the running Minikube cluster.

3. Delete Minikube

**Syntax:** minikube delete

**Explanation:** This command deletes the Minikube cluster, removing all associated resources.

4. Get Minikube Status

**Syntax:** minikube status

**Explanation:** Displays the current status of the Minikube cluster, including whether it is running.

5. kubectl get service

**Syntax:** kubectl get service [NAME] [-n NAMESPACE] [-o wide]

**Example**: kubectl get service my-service -n my-namespace

This command would retrieve information about the service named "my-service" in the namespace "my-namespace."

**Explanation**: The kubectl get service command is used to retrieve information about services in a Kubernetes cluster. Services in Kubernetes provide a stable endpoint (usually an IP address and a port) that can be used to access the pods that belong to the service. Here's an explanation of the command:

6. Get Nodes:

   **Syntax:** kubectl get nodes
   **Explanation:** Lists the nodes in the Kubernetes cluster, along with their status.

7. Get Pods:

   **Syntax:** kubectl get pods
   **Explanation:** Lists all the pods running in the cluster.

8. Describe pods:

   **Syntax:** kubectl describe pod <pod-name>
   **Explanation :** Provides detailed information about a specific pod, including its containers, volumes, and events.

9. Kubectl cluster infokub

   **Syntax:** kubectl cluster info
   **Explanation :** Provides detailed information cluster.

10. Kubectl get all

    **Syntax:** kubectl get all
    **Explanation:** It will display all the pods which is currently running.

11. kubectl delete node <node_name>:

    **Syntax:** kubectl delete node my-node
    **Explanation:** Deletes a specific node from the cluster, resulting in the rescheduling of the affected pods to other available nodes.

12. Kubectl top node

    **Syntax**: kubectl top node
    **Explanation:** Displays resource usage statistics—such as CPU and memory consumption—for each node in the cluster.

13. kubectl create pod <pod_name>

    **Syntax:** `kubectl create pod my-pod --image=my-container-image`
    **Explanation:** Creates a new pod using a YAML or JSON file that describes the pod's configuration and specifications.

14. Kubectl delete service <service_name>

    **Syntax:** `kubectl delete service my-service`
    **Explanation** Delete a specific service in the cluster.

15. Kubectl create deployment <deployment_name>

    Syntax: `kubectl create deployment my-deployment`
    Explanation : Create a new deployment in Kubernetes with a specified name.

16. Kubectl logs <pod_name>

    Syntax: kubectl logs my_pod
    Explanation: Retrieve the logs from a specific pods.

**Working of Kubernetes with Java display Hello World**

**Step 1: Create HelloWorld.java**

```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello, World!");
   }
}
```

**File Name: HelloWorld.java**

**Step 2: Compile the Java Application:**

Open a terminal and navigate to the directory containing HelloWorld.java. Compile the Java code using the javac command.

**Step 3: Create a Dockerfile:**

Create a Dockerfile in the same directory as your Java source code. This file specifies how the Docker image should be built.

```
FROM openjdk:11

COPY HelloWorld.class /app/HelloWorld.class

WORKDIR /app

CMD ["java", "HelloWorld"]
```

**Step 4: Build the Docker Image:**

Build the Docker image using the following command:

docker build -t helloworld-java .

This command creates a Docker image with the tag helloworld-java

**Step 5: Run the Docker Container:**

Run a Docker container based on the image you just built:

docker run helloworld-java

You should see the "Hello, World!" message printed in the console.

**Step 6: Create a Kubernetes Deployment:**

Create a Kubernetes Deployment YAML file, e.g., deployment.yaml, specifying the Docker image to use.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld-java-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: helloworld-java
  template:
```

```
  metadata:
    labels:
      app: helloworld-java
  spec:
    containers:
    - name: helloworld-java-container
      image: <your-docker-username>/helloworld-java
```

**Replace <your-docker-username> with your Docker Hub username.**

**Step 7: Apply the Deployment to Kubernetes:**

Apply the Deployment YAML file to create the deployment in the Kubernetes cluster.

kubectl apply -f deployment.yaml

**Step 8: Expose the Deployment as a Service:**

Create a Service YAML file, e.g., service.yaml, to expose the deployment.

```
apiVersion: v1
kind: Service
metadata:
  name: helloworld-java-service
spec:
  selector:
    app: helloworld-java
  ports:
    - protocol: TCP
```

```
    port: 80
    targetPort: 8080
  type: LoadBalancer
```

Apply the Service YAML file to create the service:

```
kubectl apply -f service.yaml
```

**Step 9 Access the Application:**

Once the service is created, use the following command to open the application in a web browser or retrieve the external IP:

```
kubectl get svc helloworld-java-service
```