**Developer steps for developing an application with Minikube**

# MAC OS

## Installing Docker:

* Check the system requirement and select the MAC chip which is required
  + MAC with Intel chip
  + MAC with Apple Silicon
  + And other requirements
* Download Docker.dmg image
* Double-click Docker.dmg to open the installer, then drag the Docker icon to the Applications folder.
* Double-click Docker.app in the Applications folder to start Docker. In the example below, the Applications folder is in “grid” view mode.

<https://docs.docker.com/desktop/install/mac-install/>

## Installing Kubectl :

* Install with Homebrew on macOS
  + brew install kubectl
  + kubectl version --client

<https://kubernetes.io/docs/tasks/tools/install-kubectl-macos/#install-with-homebrew-on-macos>

## Installing Minikube:

* Select the system requirement for Docker:
  + Operating system: Mac OS
  + Architecture: ARM64
  + Release type: Stable
  + Installer type:Homebrew
* brew install minikube
* Check if minikube is installed by running “which minikube” output

# Windows OS

## Installing kubectl:

* For the easy installation of kubectl in Windows, use the given link to download kubectl.exe - <https://storage.googleapis.com/kubernetes-release/release/v1.12.0/bin/windows/amd64/kubectl.exe>
* After this, you will be able to see kubectl.exe in Downloads folder. It is better if you keep the kubectl.exe file in C drive where all the program files are stored.
* For doing the same, create a new folder in C drive, name is “kube” and move the kubectl.exe file here.
* Now, we need to set the path. Open Control Panel > System Properties > Advanced System Settings.
* Open Environment Variables, set the variable name as “Path” and variable value as the path of kubectl file in C drive.
* Lastly, we will verify that kubectl is installed correctly or not. So, open the command prompt and run kubectl. As it shows the details, it is confirmed that kubectl is successfully installed.

## Installing Docker:

For installing Docker on Windows, we need 1 thing to keep in mind that docker only supports the Docker Desktop version in Windows and it has some prerequisites.

**Prerequisites -**

* The operating system should be -Windows 10 with 64 bit
* RAM has to be minimum 4 GB.
* BIOS settings should have hardware virtualization support enabled.
* The Hyper V feature, WSL-2 feature and the Container feature should be enabled in windows.
* Windows supported by Microsoft need to be updated, if computers have older versions of Windows 10. Docker will not work on older versions of Windows which are not within Microsoft's servicing timeline.

**Installation Procedure**:

* Download the docker file from <https://docs.docker.com/docker-for-windows/install/>
* As the Docker Desktop Installer.exe gets downloaded, double click on the file and start the installation.
* Enable Hyper-V windows feature on the configuration page.
* Now, follow the installation and wait till the process is completed.
* After the installation process is complete, click Close and Restart.

You should now be able to successfully run Docker Desktop on Windows.

Check the Docker installation version on your system by going to Docker CLI and running docker version

With this, docker is successfully installed and running on Windows.

## Installing Minikube:

* Check the windows architecture if it is X86 or X64
* Install the latest minikube stable release on x86-64 Windows using .exe download
* Open PowerShell, and enter this command:

New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force

Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/download/minikube-windows-amd64.exe' -UseBasicParsing

* Add the minikube.exe binary to your PATH.

Make sure to run PowerShell as Administrator and use the following command

$oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine)

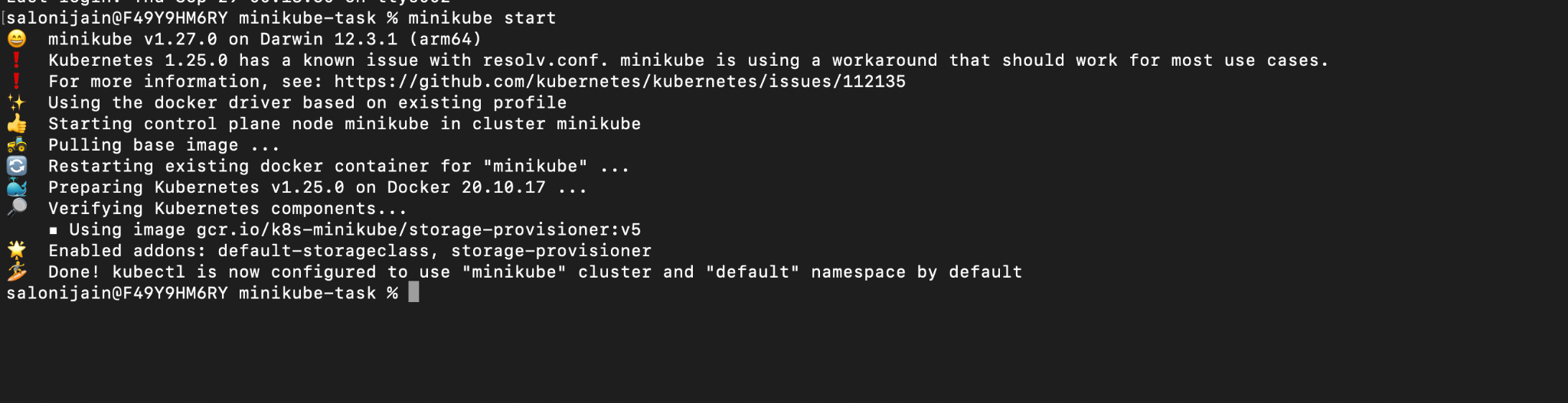
if ($oldPath.Split(';') -inotcontains 'C:\minikube'){ `

[Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machine) `

}

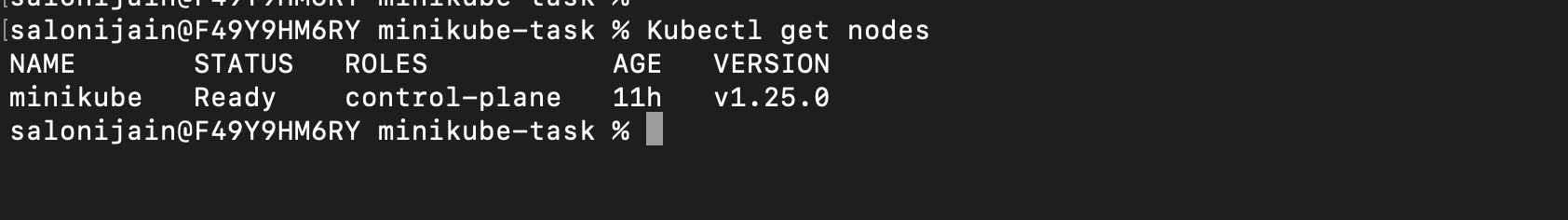
# Minikube Demo (Nginx and Hello World):

* Start you cluster by running: minikube start



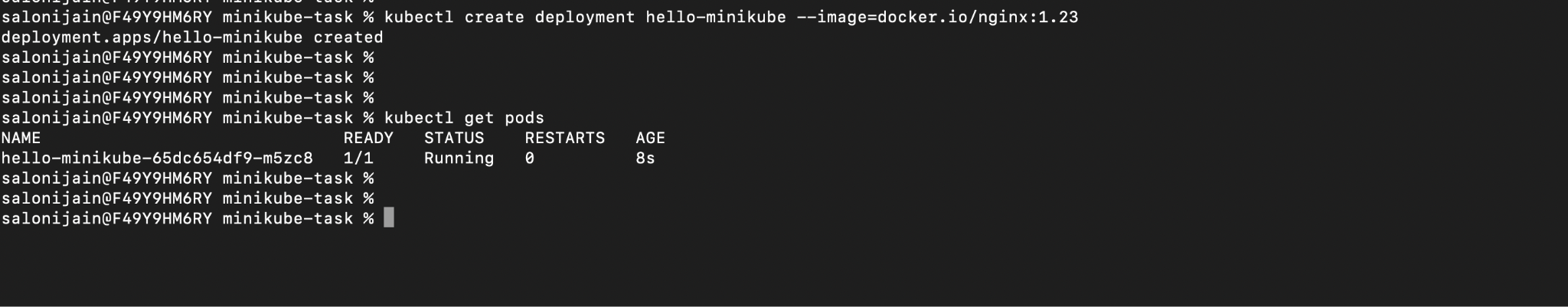


* Run a command to see nodes: “kubectl get nodes”



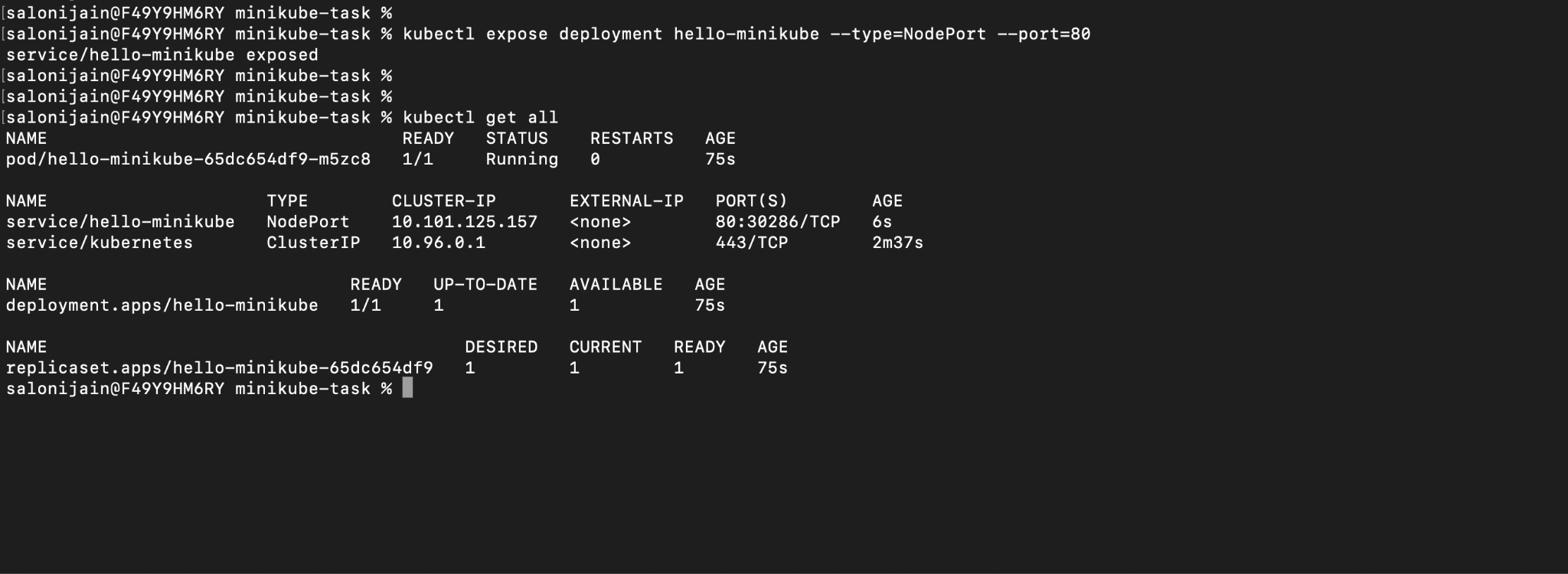


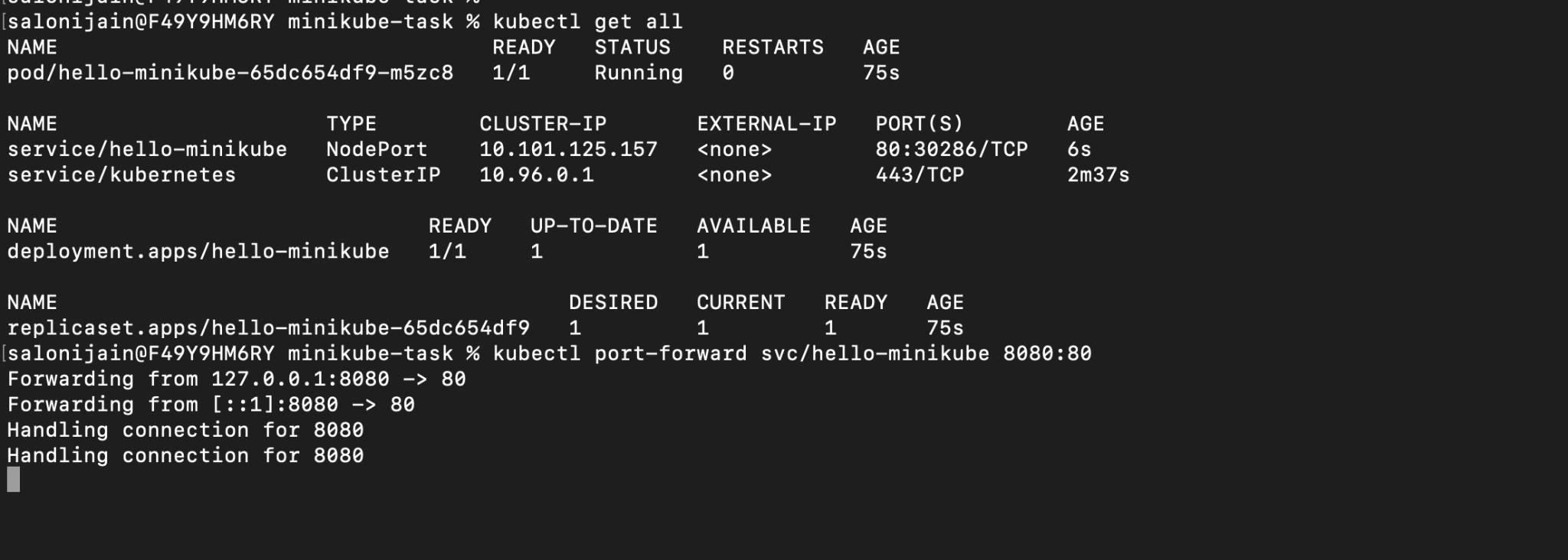
* Deploying an application on minikube
  + Run following command : “kubectl create deployment hello-minikube --image=docker.io/nginx:1.23”

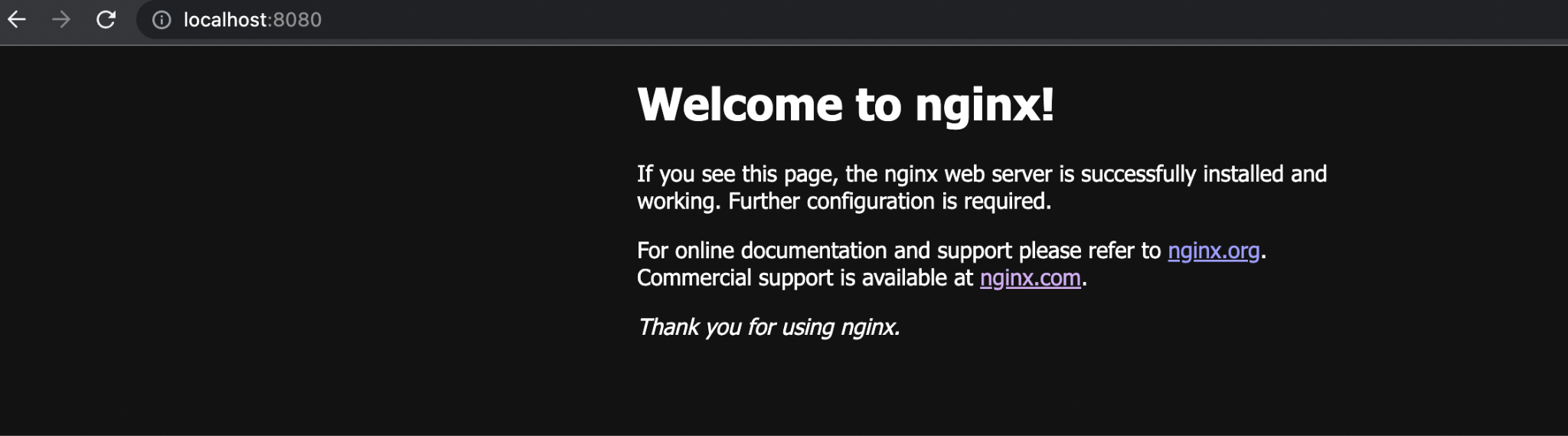




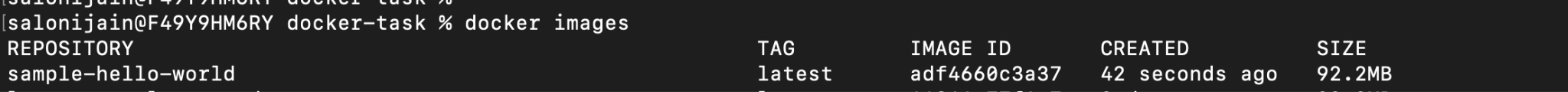
* Exposing the application and port forwarding to the local browser on 8080







* Creating a hello-world python application and deploying it to Minikube
  + For building docker image:
    - Set the environment variable as : eval $(minikube docker-env). If building image locally and testing it
    - docker build --platform -t sample-hello-world . (if using Linux as a local system)
    - docker buildx build --platform linux/amd64 -t sample-hello-world . (if using mac as a local system)



Building an image and listing out that docker image

* To build an image, I have used these following files :

|  |
| --- |
| #Dockerfile  FROM python:alpine3.7 COPY . /app WORKDIR /app RUN pip install -r requirements.txt EXPOSE 5001 ENTRYPOINT [ "python" ] CMD [ "main.py" ] |

|  |
| --- |
| #main.py  from flask import Flask app = Flask(\_\_name\_\_)   @app.route('/') def hello():  return "Hello World"     if \_\_name\_\_ == "\_\_main\_\_":  app.run(host='0.0.0.0', port=5001) |

|  |
| --- |
| #requirements.txt  flask |

Now deploy it to Minikube with simple deployment files:

|  |
| --- |
| apiVersion: apps/v1 kind: Deployment metadata:  name: sample-deployment  labels:  app: hello-world spec:  replicas: 2  selector:  matchLabels:  app: hello-world  template:  metadata:  labels:  app: hello-world  spec:  containers:  - name: hello-world  image: sample-hello-world  imagePullPolicy: Never  ports:  - containerPort: 5001 --- kind: Service apiVersion: v1 metadata:  name: sample-service  labels:  app: hello-world spec:  selector:  app: hello-world  ports:  - port: 5001  protocol: TCP  type: NodePort |



* Port- forward the following service to view it on local browser

