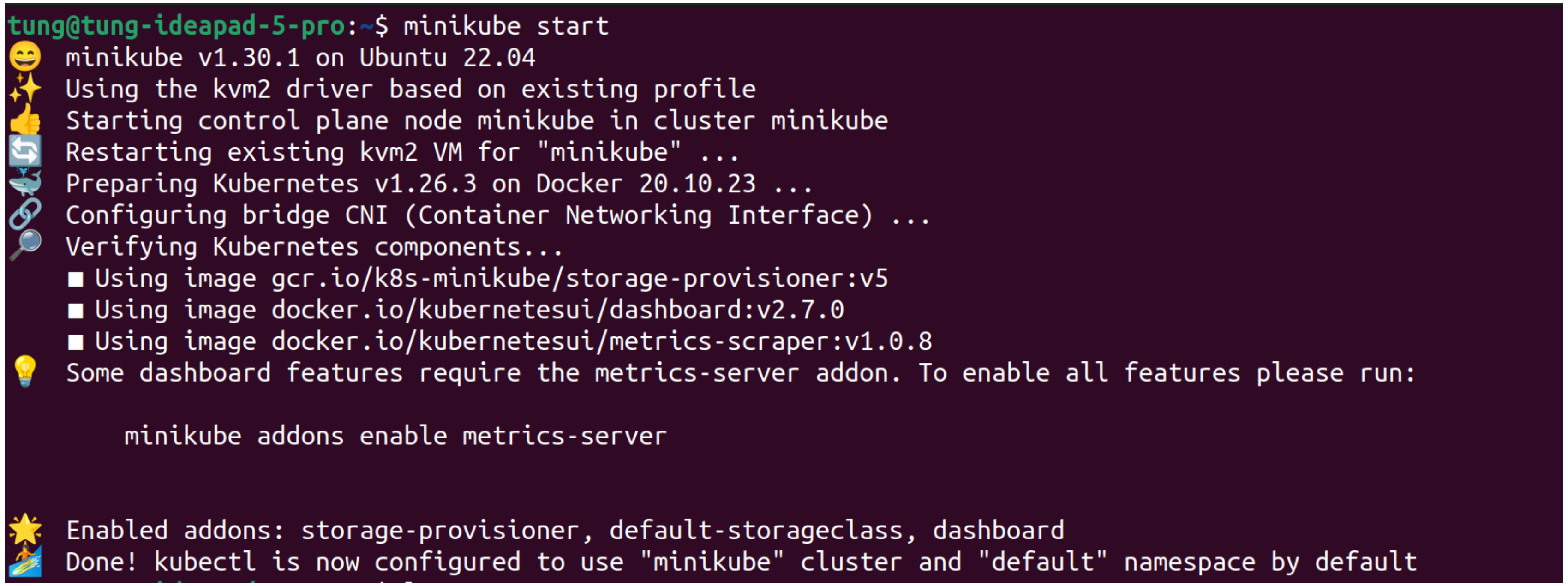
Lesson 2: Setting Up a Local Kubernetes Environment

Lữ Thanh Tùng

Deploy a simple "Hello World" application on your local cluster using a Kubernetes manifest file

Cài và khởi động minikube



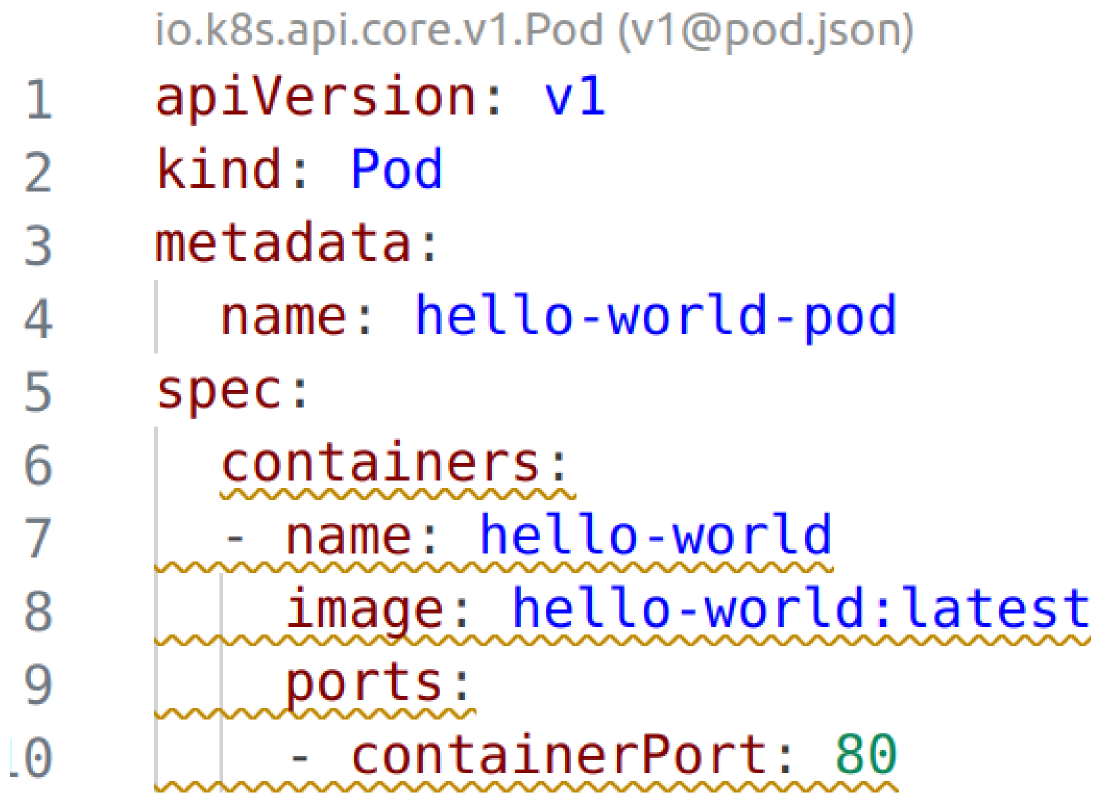
```
tung@tung-ideapad-5-pro:~$ minikube start
😄  minikube v1.30.1 on Ubuntu 22.04
✨  Using the kvm2 driver based on existing profile
👍  Starting control plane node minikube in cluster minikube
🔄  Restarting existing kvm2 VM for "minikube" ...
🐳  Preparing Kubernetes v1.26.3 on Docker 20.10.23 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server


🌟  Enabled addons: storage-provisioner, default-storageclass, dashboard
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Deploy a simple "Hello World" application on your local cluster using a Kubernetes manifest file

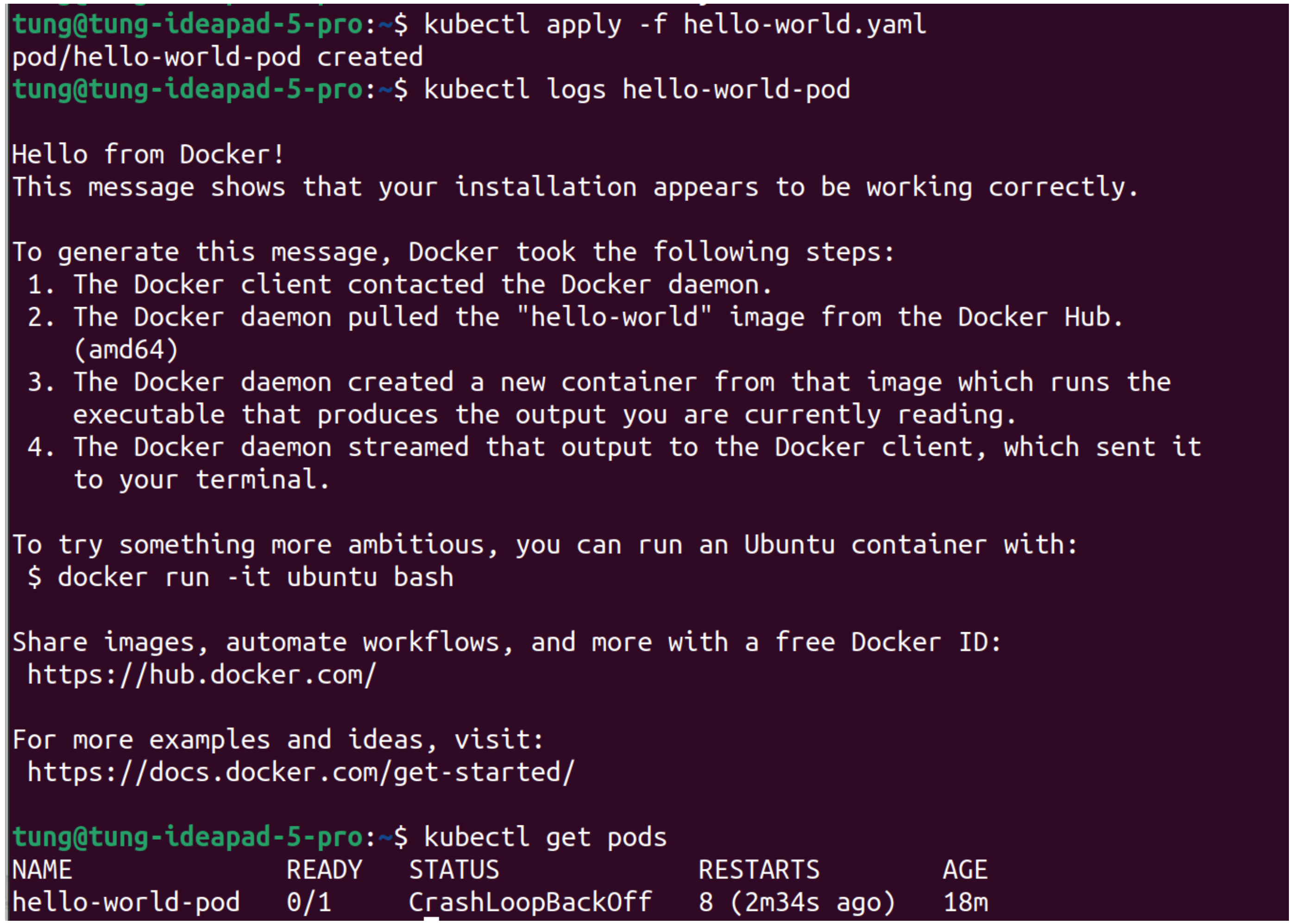
- Tạo file yaml


io.k8s.api.core.v1.Pod (v1@pod.json)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-world-pod
spec:
  containers:
  - name: hello-world
    image: hello-world:latest
    ports:
    - containerPort: 80
```

Deploy a simple "Hello World" application on your local cluster using a Kubernetes manifest file

- Tạo pod đơn giản với file yaml đã tạo và xem thông tin



```
tung@tung-ideapad-5-pro:~$ kubectl apply -f hello-world.yaml
pod/hello-world-pod created
tung@tung-ideapad-5-pro:~$ kubectl logs hello-world-pod

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

tung@tung-ideapad-5-pro:~$ kubectl get pods
NAME              READY   STATUS             RESTARTS        AGE
hello-world-pod   0/1     CrashLoopBackOff   8 (2m34s ago)   18m
```