# k8 - Azure Kubernetes Service

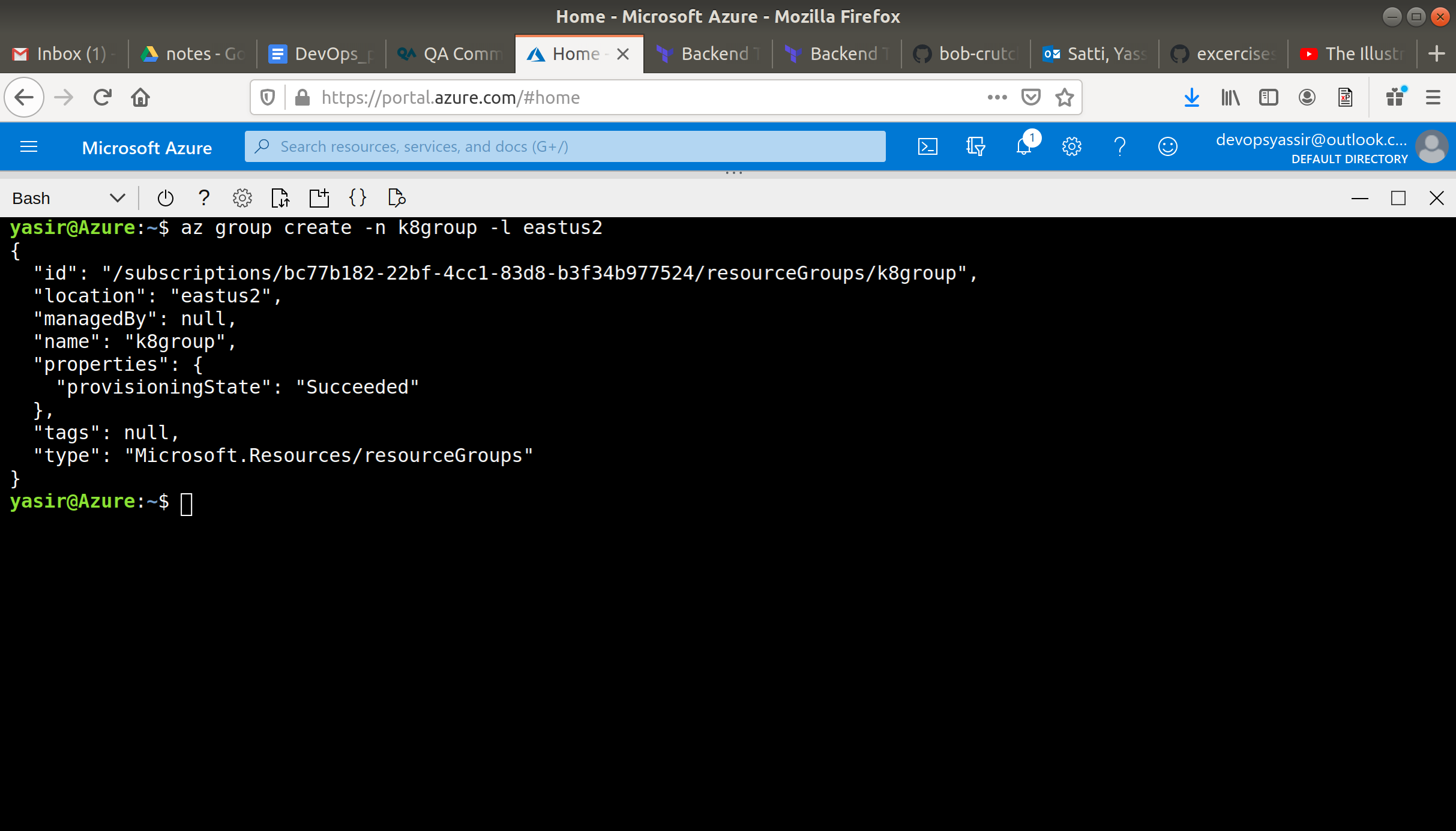
# 

## **Tutorial**

In this tutorial, we will be deploying an AKS cluster using the Azure CLI.

1. **Create a Resource Group**:

az group create -n k8group -l uksouth



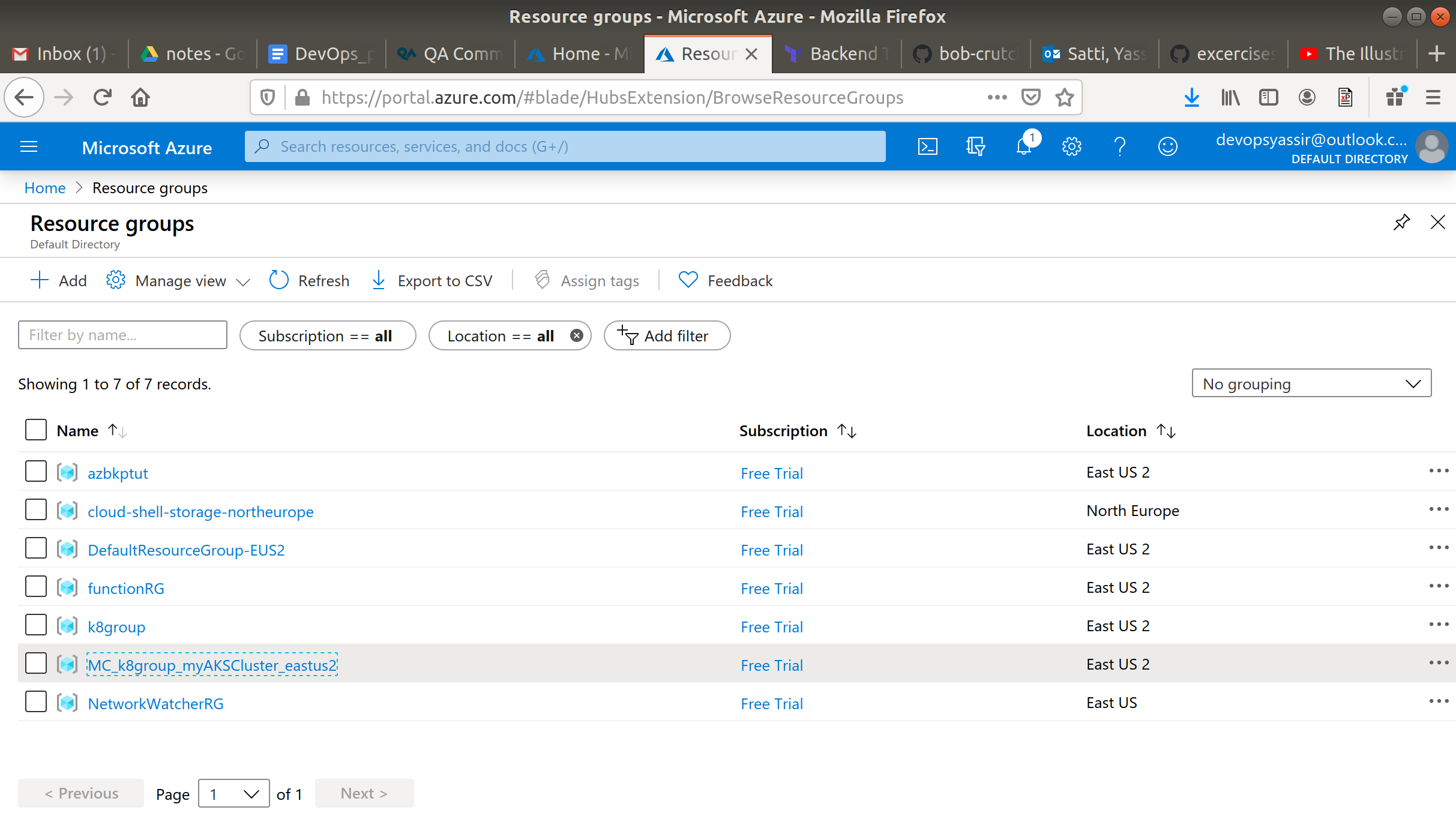
1. **Create AKS Cluster**:

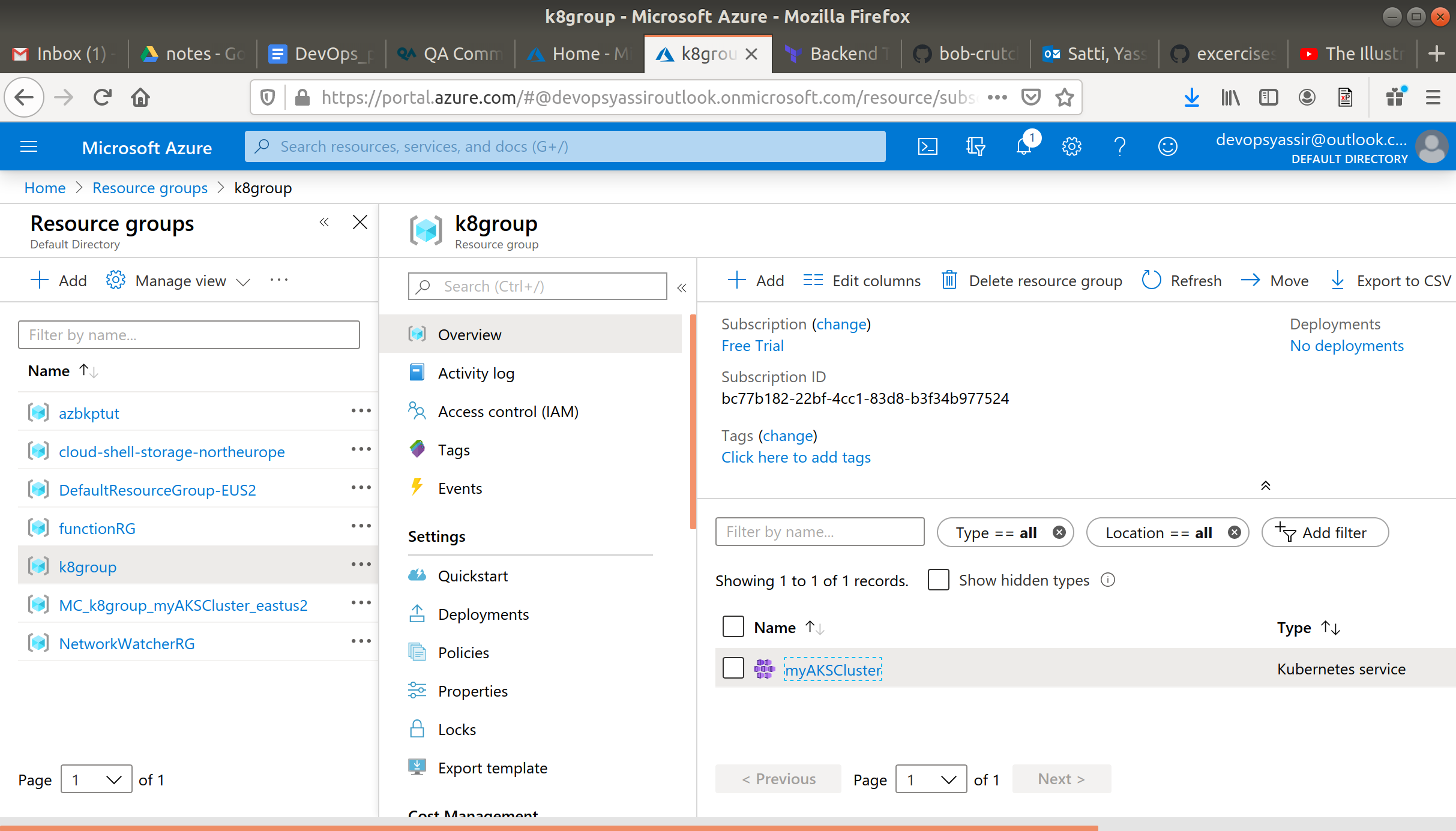
az aks create --resource-group k8group --name myAKSCluster --node-count 1 --enable-addons monitoring --generate-ssh-keys

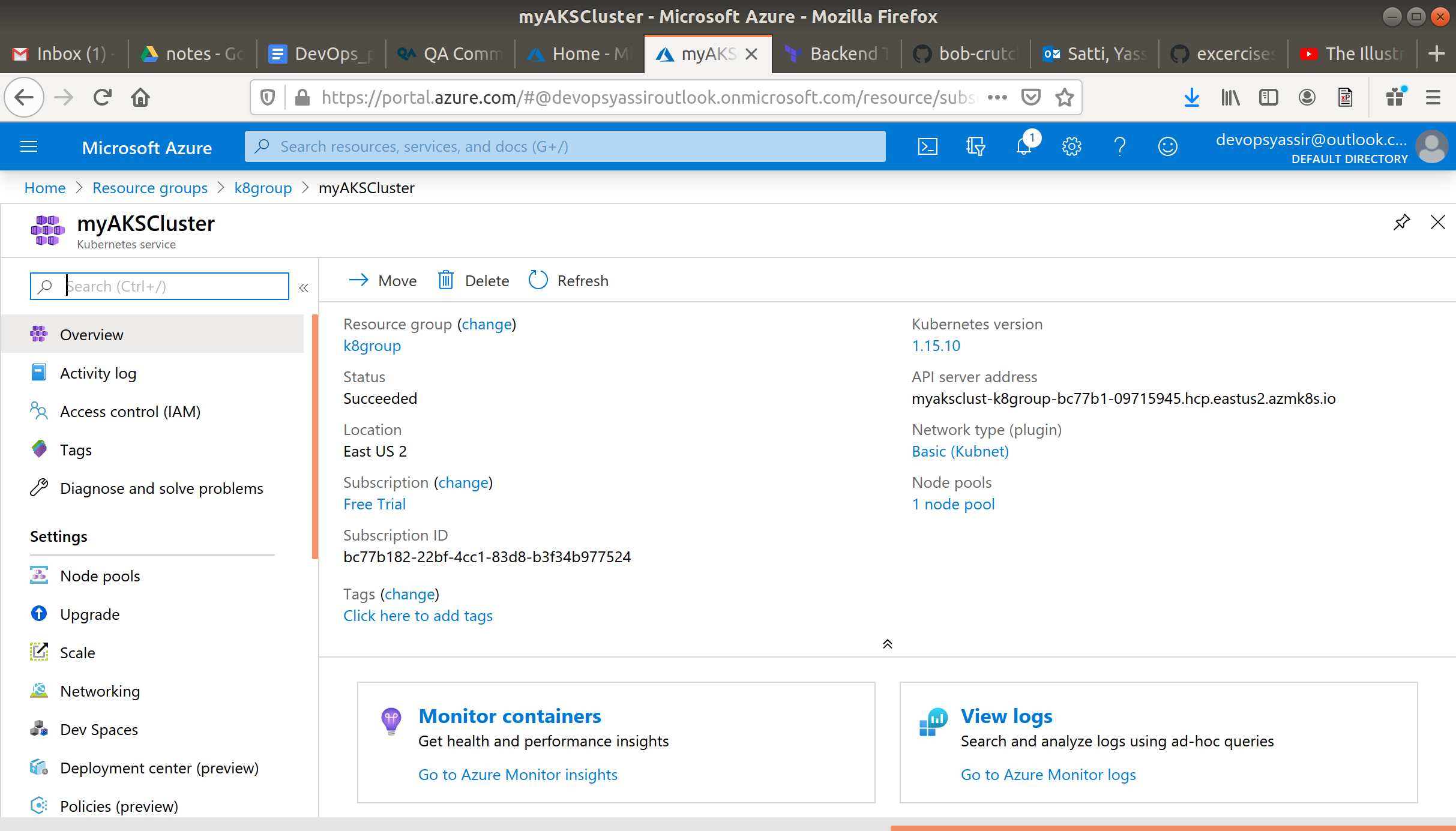
Here, we have created an AKS cluster called myAKSCluster with **one** node.

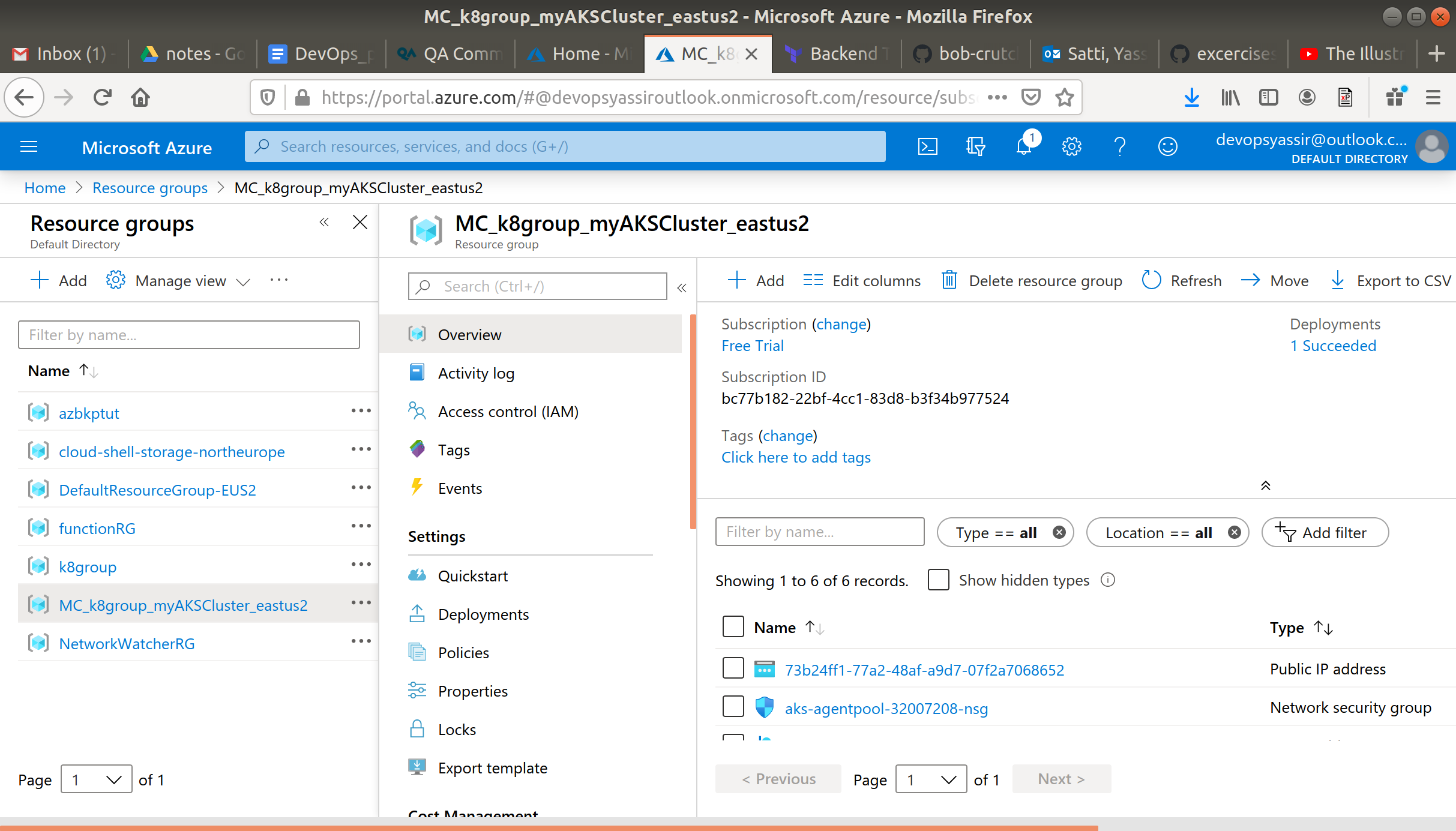
The --enable-addons monitoring command ensures that **Azure Monitor for containers** is enabled.

This command will take a few minutes to run.





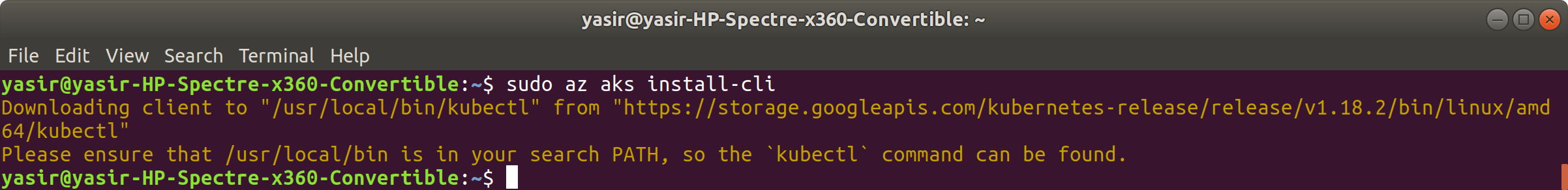




1. **Connect to the Cluster**:

To manage a Kubernetes cluster, you use kubectl, the Kubernetes command-line client:

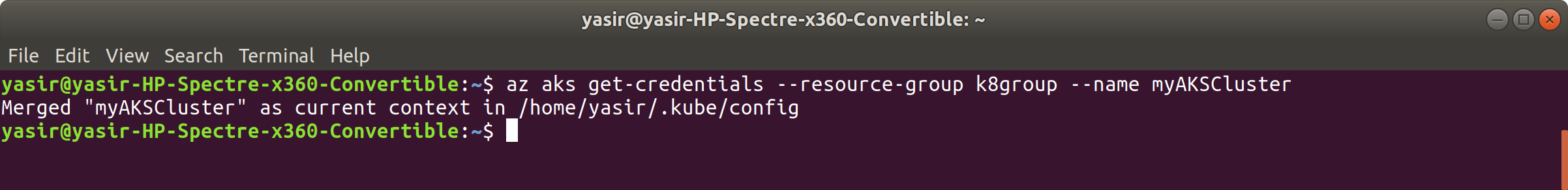
**$ az aks install-cli**



To configure kubectl to connect to your Kubernetes cluster, use the az aks get-credentials command.

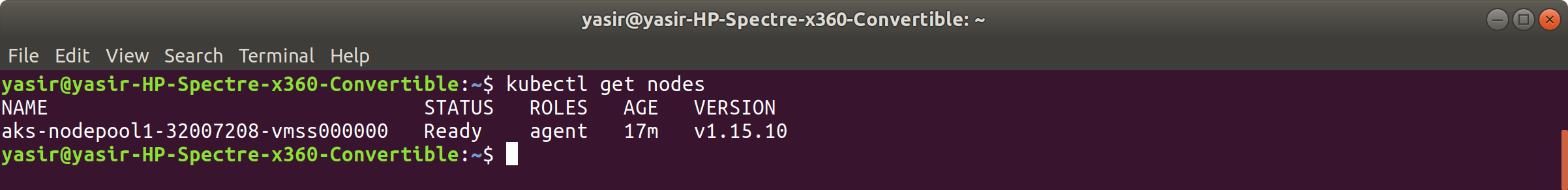
This command downloads credentials and configures the Kubernetes CLI to use them:

**$ az aks get-credentials --resource-group k8group --name myAKSCluster**



Verify your connection to the cluster using:

**$ kubectl get nodes**

****

This should show the single node created in the previous step. The STATUS should say Ready.

1. **Run the Application**:

Create a file called python-http-server.yaml and enter the following into it:

apiVersion: apps/v1

kind: Deployment

metadata:

name: python-http-server

labels:

app: python-http-server

spec:

replicas: 1

selector:

matchLabels:

app: python-http-server

template:

metadata:

labels:

app: python-http-server

spec:

containers:

- name: python-http-server

image: bobcrutchley/python-http-server:latest

ports:

- name: http-port

containerPort: 9000

---

apiVersion: v1

kind: Service

metadata:

name: python-http-server

spec:

type: LoadBalancer

ports:

- port: 80

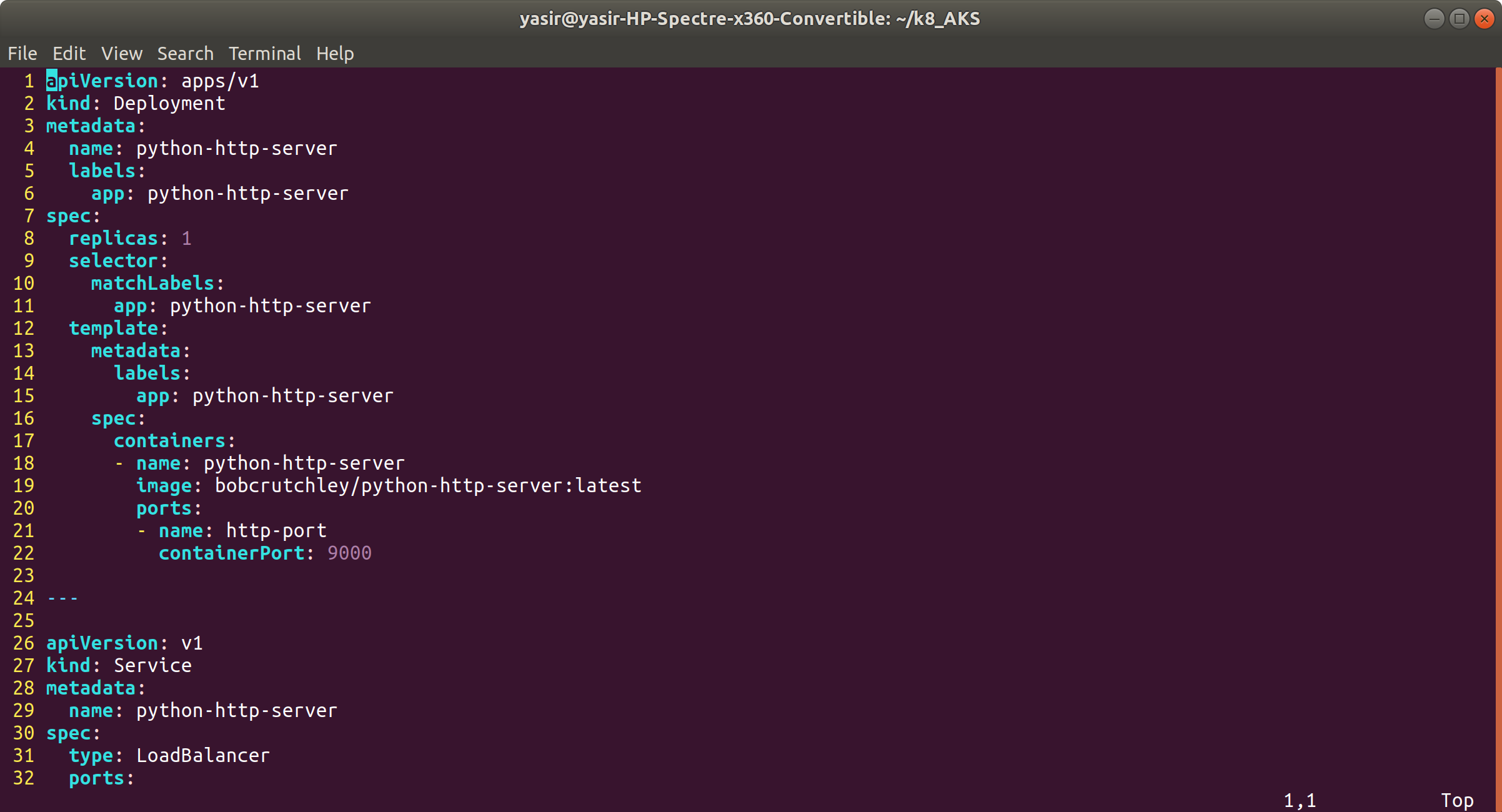
targetPort: 9000

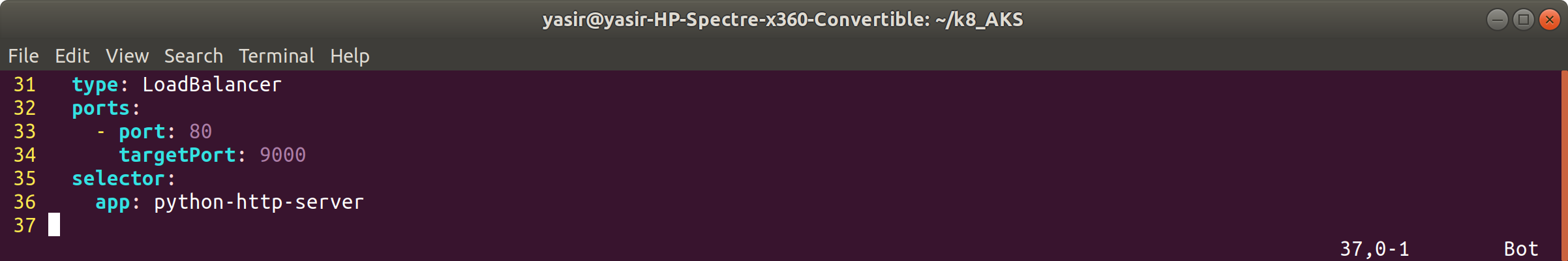
selector:

app: python-http-server

This creates:

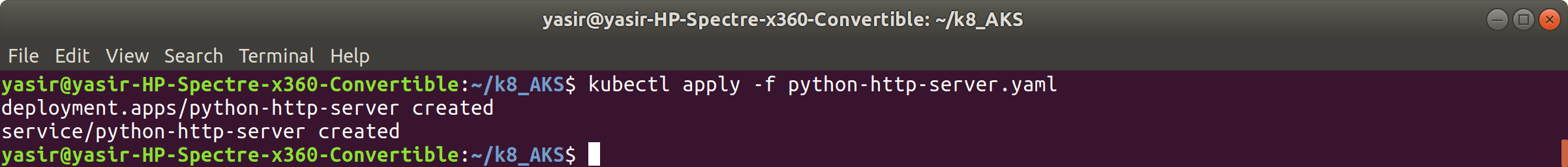
* **1 Deployment** - the **application**
* **1 Service** - the **LoadBalancer**.





To deploy the application use:

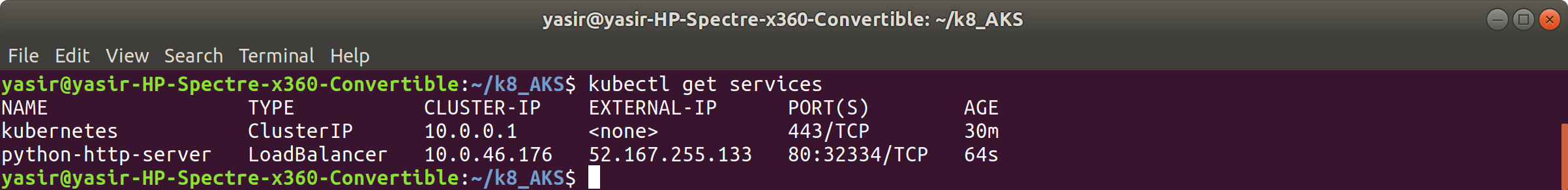
**$ kubectl apply -f python-http-server.yaml**

****

1. **Test the Application**:

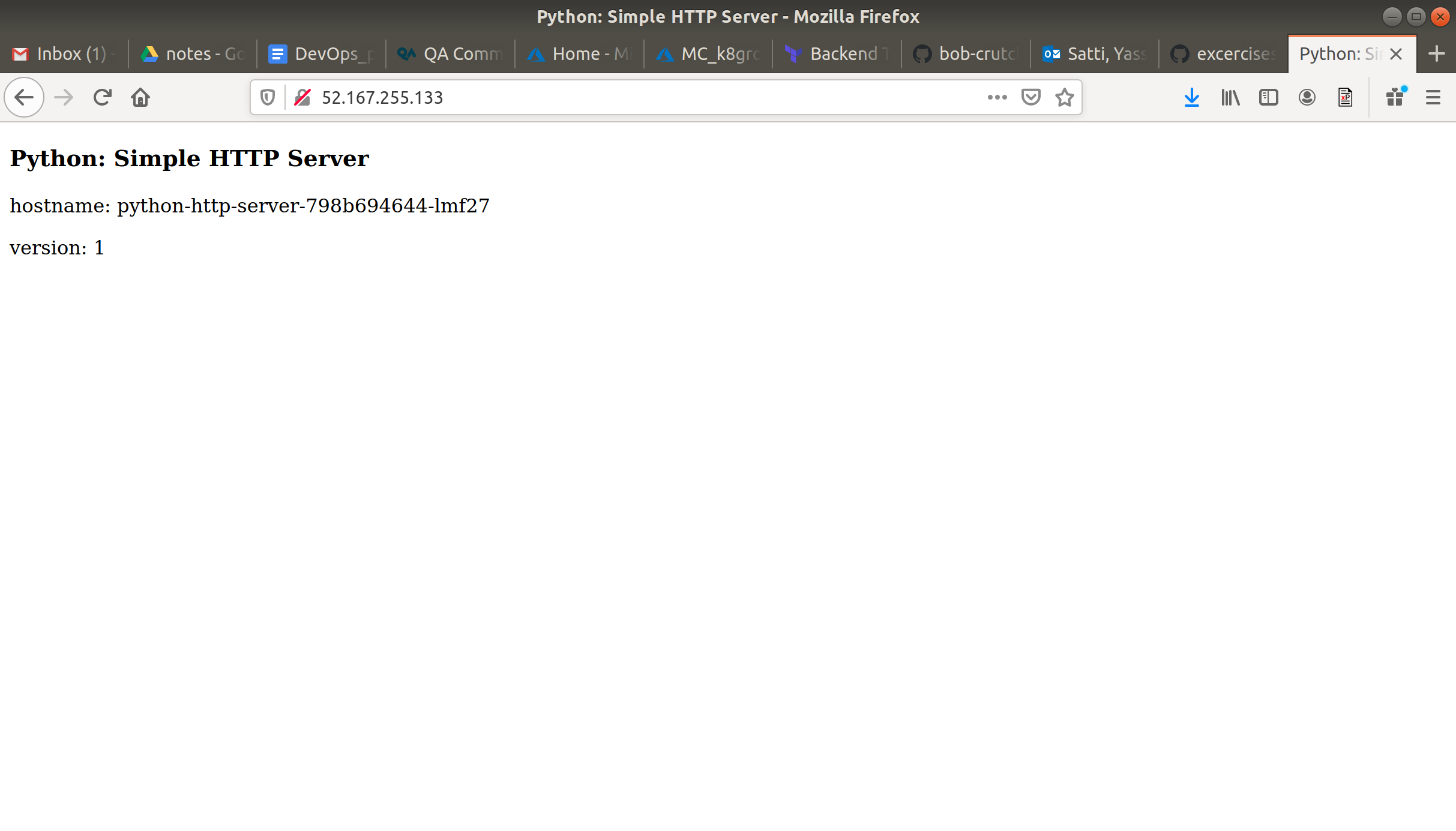
Run the following command:

**$ kubectl get services**

****

This will show you the LoadBalancer's EXTERNAL-IP address.

Go to your browser and enter the External IP address. The application should be running!



## **Exercises**

1. Increase the replicas to **10** and redeploy the python-http-server application  
   * You could curl the IP address a few times at this point and see how the traffic is being distributed between the nodes in the cluster!

### **Python: Simple HTTP Server**

hostname: python-http-server-798b694644-6n55n

hostname: python-http-server-798b694644-wscfg

hostname: python-http-server-798b694644-lmf27

hostname: python-http-server-798b694644-frdl5

hostname: python-http-server-798b694644-7l582

#### **Clean up**

To clean up after the tutorial, and ensure you don't incur any costs, use the following commands:

**$ az group delete -n k8group -y --no-wait**

**$ az group delete -n DefaultResourceGroup-SUK -y --no-wait**