

1. Git clone chapter-7

```
Windows PowerShell
PS C:\Users\Ashis\desktop\SIT722> git clone https://github.com/bootstrapping-microservices-2nd-edition/chapter-7
Cloning into 'chapter-7'...
remote: Enumerating objects: 241, done.
remote: Counting objects: 100% (241/241), done.
remote: Compressing objects: 100% (123/123), done.
remote: Total 241 (delta 105), reused 237 (delta 101), pack-reused 0 (from 0)
Receiving objects: 100% (241/241), 1.04 MiB | 2.06 MiB/s, done.
Resolving deltas: 100% (105/105), done.
PS C:\Users\Ashis\desktop\SIT722>
```

2. Az --version , terraform --version and kubectl --version

```
Windows PowerShell
PS C:\Users\Ashis\desktop\SIT722> az --version
azure-cli                2.62.0 *
core                     2.62.0 *
telemetry                1.1.0

Dependencies:
msal                     1.28.1
azure-mgmt-resource     23.1.1

Python location 'C:\Program Files\Microsoft SDKs\Azure\CLI2\python.exe'
Extensions directory 'C:\Users\Ashis\.azure\cliextensions'

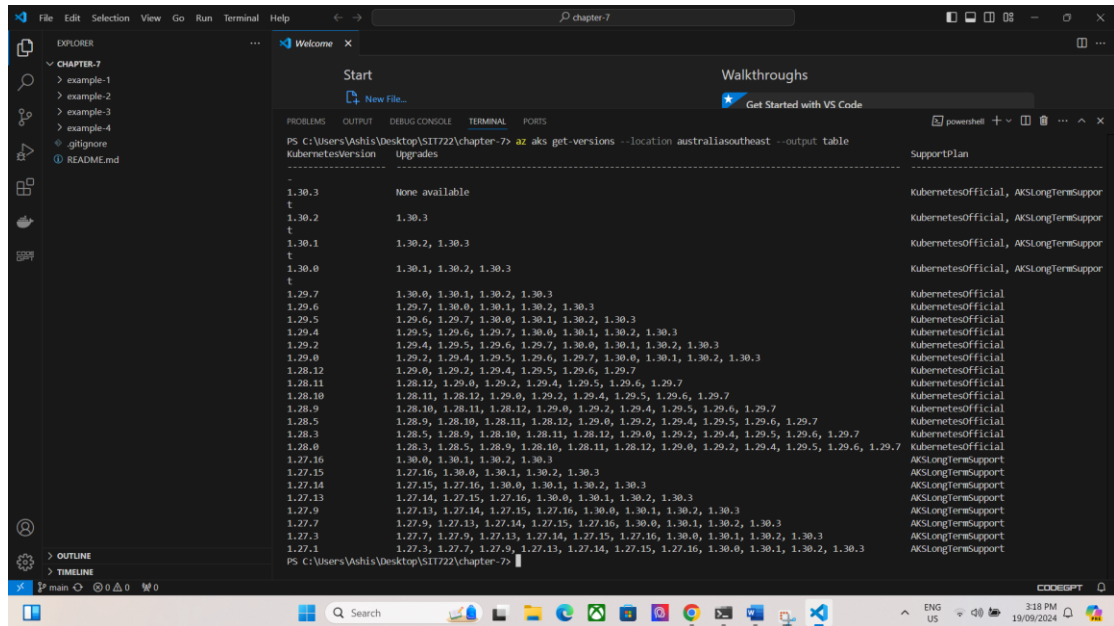
Python (Windows) 3.11.8 (tags/v3.11.8:db85d51, Feb  6 2024, 22:03:32) [MSC v.1937 64 bit (AMD64)]

Legal docs and information: aka.ms/AzureCliLegal

You have 2 update(s) available. Consider updating your CLI installation with 'az upgrade'
PS C:\Users\Ashis\desktop\SIT722> terraform --version
Terraform v1.5.7
on windows_amd64

Your version of Terraform is out of date! The latest version
is 1.9.5. You can update by downloading from https://www.terraform.io/downloads.html
PS C:\Users\Ashis\desktop\SIT722> kubectl version
Client Version: v1.30.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Unable to connect to the server: dial tcp: lookup flitubepr-flitubeprod-6c3473-11bt107k.hcp.australiacentral.azmk8s.io: no such host
PS C:\Users\Ashis\desktop\SIT722>
```

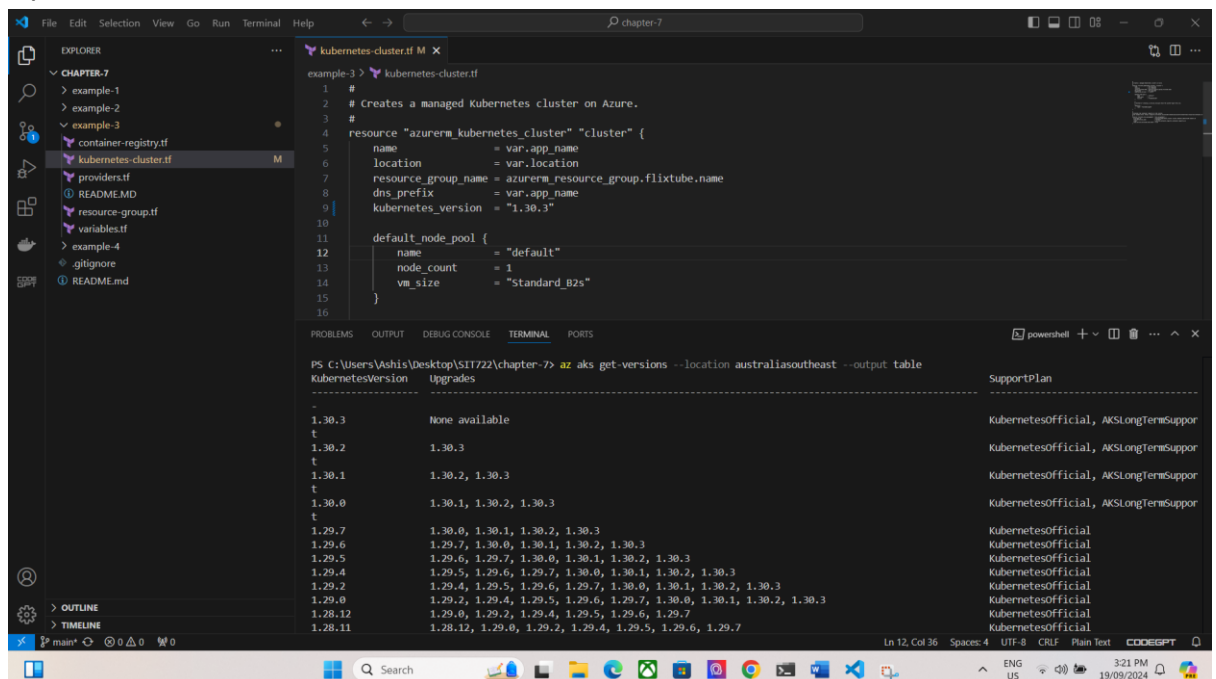
3. Get Kubernetes version by region



The screenshot shows a VS Code window with a terminal running the command `az aks get-versions --location australia-southeast --output table`. The output is a table listing available Kubernetes versions and their support plans.

KubernetesVersion	Upgrades	SupportPlan
-	-	-
1.30.3	None available	KubernetesOfficial, AKSLongTermSupport
t	-	-
1.30.2	1.30.3	KubernetesOfficial, AKSLongTermSupport
t	-	-
1.30.1	1.30.2, 1.30.3	KubernetesOfficial, AKSLongTermSupport
t	-	-
1.30.0	1.30.1, 1.30.2, 1.30.3	KubernetesOfficial, AKSLongTermSupport
t	-	-
1.29.7	1.30.0, 1.30.1, 1.30.2, 1.30.3	KubernetesOfficial
1.29.6	1.29.7, 1.30.0, 1.30.1, 1.30.2, 1.30.3	KubernetesOfficial
1.29.5	1.29.6, 1.29.7, 1.30.0, 1.30.1, 1.30.2, 1.30.3	KubernetesOfficial
1.29.4	1.29.5, 1.29.6, 1.29.7, 1.30.0, 1.30.1, 1.30.2, 1.30.3	KubernetesOfficial
1.29.2	1.29.4, 1.29.5, 1.29.6, 1.29.7, 1.30.0, 1.30.1, 1.30.2, 1.30.3	KubernetesOfficial
1.29.0	1.29.2, 1.29.4, 1.29.5, 1.29.6, 1.29.7, 1.30.0, 1.30.1, 1.30.2, 1.30.3	KubernetesOfficial
1.28.12	1.29.0, 1.29.2, 1.29.4, 1.29.5, 1.29.6, 1.29.7	KubernetesOfficial
1.28.11	1.28.12, 1.29.0, 1.29.2, 1.29.4, 1.29.5, 1.29.6, 1.29.7	KubernetesOfficial
1.28.10	1.28.11, 1.28.12, 1.29.0, 1.29.2, 1.29.4, 1.29.5, 1.29.6, 1.29.7	KubernetesOfficial
1.28.9	1.28.10, 1.28.11, 1.28.12, 1.29.0, 1.29.2, 1.29.4, 1.29.5, 1.29.6, 1.29.7	KubernetesOfficial
1.28.5	1.28.9, 1.28.10, 1.28.11, 1.28.12, 1.29.0, 1.29.2, 1.29.4, 1.29.5, 1.29.6, 1.29.7	KubernetesOfficial
1.28.3	1.28.5, 1.28.9, 1.28.10, 1.28.11, 1.28.12, 1.29.0, 1.29.2, 1.29.4, 1.29.5, 1.29.6, 1.29.7	KubernetesOfficial
1.28.0	1.28.3, 1.28.5, 1.28.9, 1.28.10, 1.28.11, 1.28.12, 1.29.0, 1.29.2, 1.29.4, 1.29.5, 1.29.6, 1.29.7	KubernetesOfficial
1.27.16	1.30.0, 1.30.1, 1.30.2, 1.30.3	AKSLongTermSupport
1.27.15	1.27.16, 1.30.0, 1.30.1, 1.30.2, 1.30.3	AKSLongTermSupport
1.27.14	1.27.15, 1.27.16, 1.30.0, 1.30.1, 1.30.2, 1.30.3	AKSLongTermSupport
1.27.13	1.27.14, 1.27.15, 1.27.16, 1.30.0, 1.30.1, 1.30.2, 1.30.3	AKSLongTermSupport
1.27.9	1.27.13, 1.27.14, 1.27.15, 1.27.16, 1.30.0, 1.30.1, 1.30.2, 1.30.3	AKSLongTermSupport
1.27.7	1.27.9, 1.27.13, 1.27.14, 1.27.15, 1.27.16, 1.30.0, 1.30.1, 1.30.2, 1.30.3	AKSLongTermSupport
1.27.3	1.27.7, 1.27.9, 1.27.13, 1.27.14, 1.27.15, 1.27.16, 1.30.0, 1.30.1, 1.30.2, 1.30.3	AKSLongTermSupport
1.27.1	1.27.3, 1.27.7, 1.27.9, 1.27.13, 1.27.14, 1.27.15, 1.27.16, 1.30.0, 1.30.1, 1.30.2, 1.30.3	AKSLongTermSupport

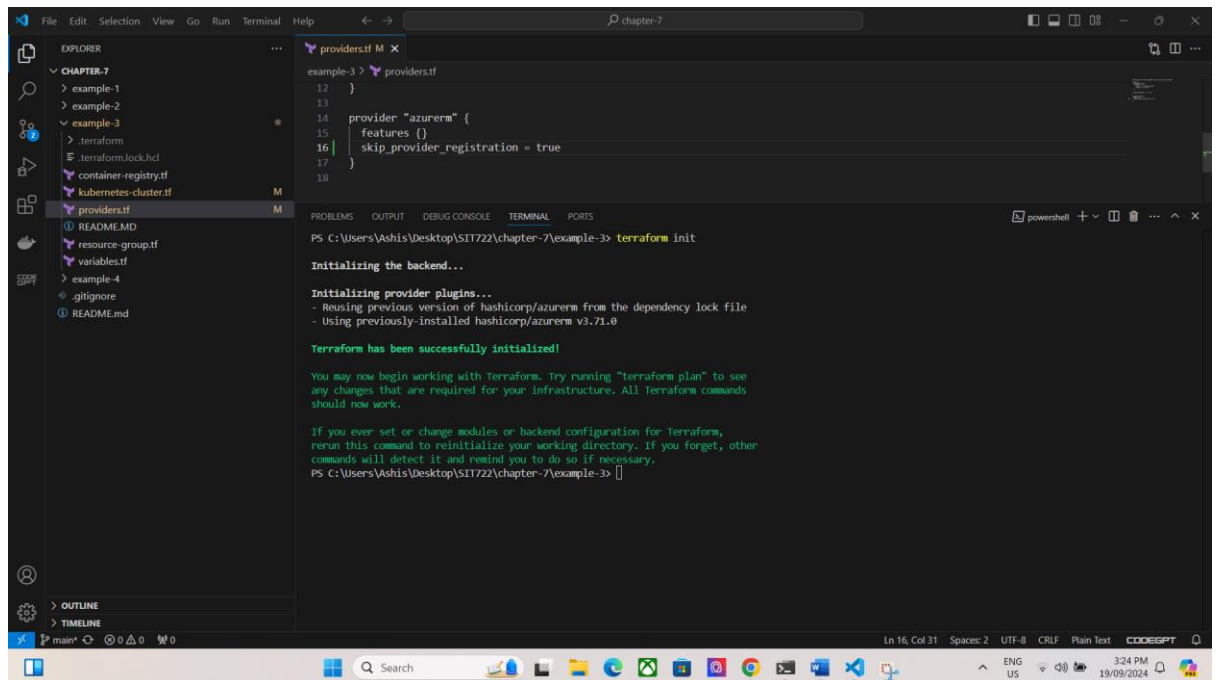
4. Update cluster.tf file



The screenshot shows a VS Code window with the file `kubernetes-cluster.tf` open. The file content is as follows:

```
1 #
2 # Creates a managed Kubernetes cluster on Azure.
3 #
4 resource "azurerm_kubernetes_cluster" "cluster" {
5   name                = var.app_name
6   location             = var.location
7   resource_group_name = azurerm_resource_group.flixtube.name
8   dns_prefix           = var.app_name
9   kubernetes_version  = "1.30.3"
10
11   default_node_pool {
12     name     = "default"
13     node_count = 1
14     vm_size  = "Standard_B2s"
15   }
16 }
```

5. Terraform commands



The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project structure for 'chapter-7' with files like 'example-1', 'example-2', 'example-3', 'terraform.lock.hcl', 'container-registry.tf', 'kubernetes-cluster.tf', 'provider.tf', 'README.md', 'resource-group.tf', 'variables.tf', 'example-4', '.gitignore', and 'README.md'. The code editor displays the 'provider.tf' file with the following content:

```
example-3 > provider.tf
12 }
13
14 provider "azurerm" {
15   features {}
16   skip_provider_registration = true
17 }
18
```

The terminal shows the output of the 'terraform init' command:

```
PS C:\Users\Ashis\Desktop\SIIT722\chapter-7\example-3> terraform init

Initializing the backend...

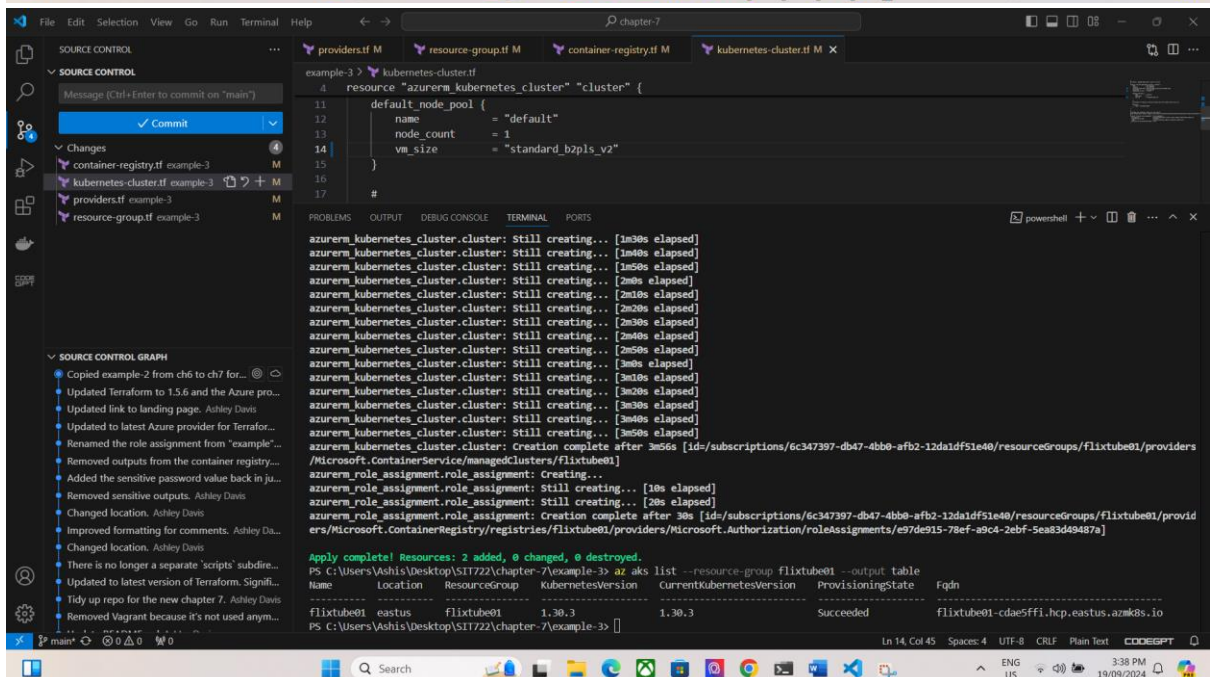
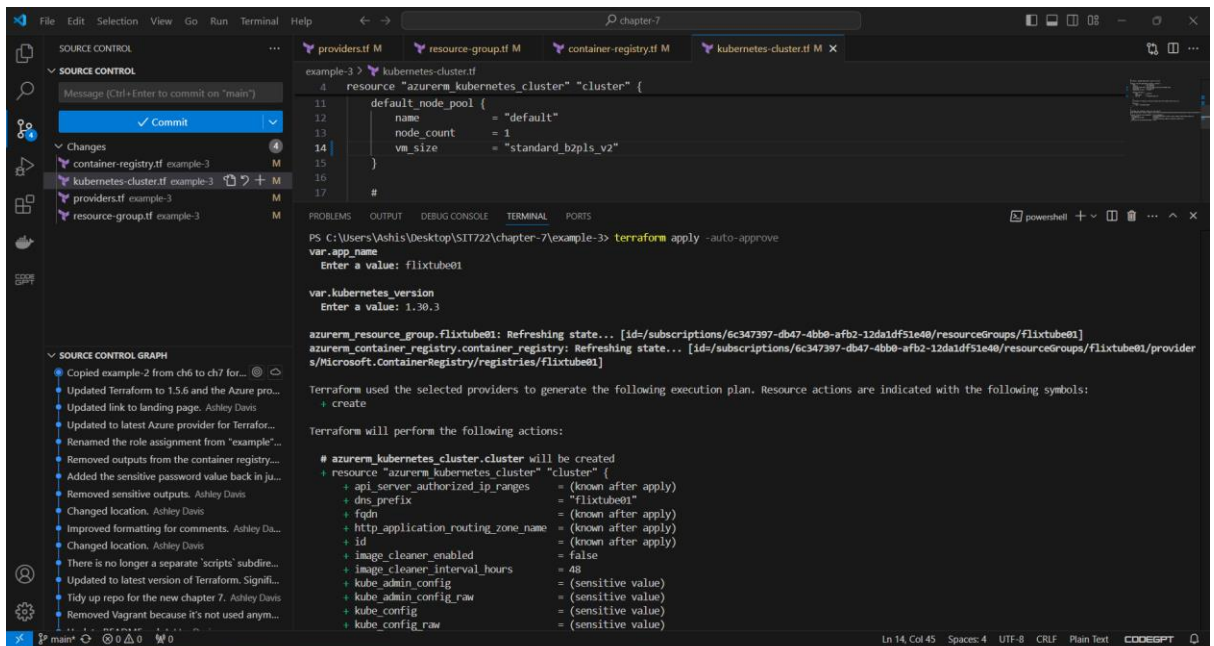
Initializing provider plugins...
- Reusing previous version of hashicorp/azurerm from the dependency lock file
- Using previously-installed hashicorp/azurerm v3.71.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\Ashis\Desktop\SIIT722\chapter-7\example-3>
```

The terminal status bar at the bottom indicates 'Ln 16, Col 31', 'Spaces: 2', 'UTF-8', 'CRLF', 'Plain Text', and 'CODEGPT'.



6. Docker push

```
example-4 > kubernetes-cluster.tf
4 resource "azurerm_kubernetes_cluster" "cluster" {
11   default_node_pool {
12     name       = "default"
13     node_count = 1
14     vm_size    = "standard_b2pls_v2"
15   }
16 }
17 #

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashis\Desktop\SIIT22\chapter-7\example-4> docker build -t video:1 -file Dockerfile-prod .
[+] building 2.4s (17/12) FINISHED
=> [internal] load build definition from Dockerfile-prod
=> transferring Dockerfile: 200B
=> [internal] load metadata for docker.io/library/node:18.17.1
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> transferring context: 80B
=> [1/6] FROM docker.io/library/node:18.17.1@sha256:933bcfad91e952a02bc29eb5aa29033e542afac4174f9524b79066d97b23c24
=> [internal] load build context
=> transferring context: 200B
=> CACHED [2/6] WORKDIR /usr/src/app
=> CACHED [3/6] COPY package*.json ./
=> CACHED [4/6] RUN npm install --only=production
=> CACHED [5/6] COPY ./src ./src
=> CACHED [6/6] COPY ./videos ./videos
=> exporting to image
=> exporting layers
=> writing image sha256:d2f72a32cc23c213bd14b25f18561c688f65f205a27f8ff80ef9a5182e75b
=> naming to docker.io/library/video:1

1 warning found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 8)

What's next:
View a summary of image vulnerabilities and recommendations > docker scout quickview
PS C:\Users\Ashis\Desktop\SIIT22\chapter-7\example-4>
```

```
example-3 > kubernetes-cluster.tf
4 resource "azurerm_kubernetes_cluster" "cluster" {
11   default_node_pool {
12     name       = "default"
13     node_count = 1
14     vm_size    = "standard_b2pls_v2"
15   }
16 }
17 #
18 # Instead of creating a service principle have the system figure this out.
19 #
20 identity {
21   type = "SystemAssigned"
22 }
23 }
```

```
PS C:\Users\Ashis\Desktop\SIIT22\chapter-7\example-4> az acr login --name flixtube01
login Succeeded
PS C:\Users\Ashis\Desktop\SIIT22\chapter-7\example-4> docker push flixtube01.azurecr.io/video:1
The push refers to repository [flixtube01.azurecr.io/video]
948c5e063807: Pushed
4751b672f4cf: Pushed
239c0bc592c3: Pushed
3e0b7816d719: Pushed
6abdf8584779: Pushed
43f3afe88979: Pushed
4c182359a7e3: Pushed
62fae29abe47: Pushed
5bee294ea097: Pushed
8fe5334a79c9: Pushed
ac4413ce78f8: Pushed
1206fac91f32: Pushed
b8544860ba0b: Pushed
1: digest: sha256:2a15694dfbb9bfaf3dae66d875a8820047228e81acf7bb9c327b314cc3b3fd size: 3049
PS C:\Users\Ashis\Desktop\SIIT22\chapter-7\example-4> az acr repository list --name flixtube01 --output table
Result
-----
video
PS C:\Users\Ashis\Desktop\SIIT22\chapter-7\example-4>
```

```
PS C:\Users\Ashis\Desktop\SIIT22\chapter-7\example-4> kubectl get nodes
NAME STATUS ROLES AGE VERSION
aks-default-21466826-vmss000000 Ready <none> 23m v1.30.3
PS C:\Users\Ashis\Desktop\SIIT22\chapter-7\example-4> kubectl cluster-info
Kubernetes control plane is running at https://flixtubeashish-f93wu0t2.hcp.eastus.azmk8s.io:443
CoreDNS is running at https://flixtubeashish-f93wu0t2.hcp.eastus.azmk8s.io:443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://flixtubeashish-f93wu0t2.hcp.eastus.azmk8s.io:443/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy
```

Describe in your own words what are the four example applications(stages) each demonstrating?

Example 1:- This example demonstrates how to deploy a simple application in Kubernetes. It introduces the basic concepts of creating a Kubernetes deployment running a containerized application and exposing it using a service to make it accessible.

Example 2:- This example showcases how to use Terraform to automate the creation of an Azure Container Registry. The registry can then be used to store Docker images which can later be deployed in Kubernetes.

Example 3:- In this example, along with example 2 steps . This gets the infrastructure ready for running apps in containers.

Example 4:- This example shows how to deploy an app to the Kubernetes cluster made in Example 3. It builds the app using a Dockerfile and then uses a YAML file to deploy and manage the app in the cluster.

In this project, the latest **Kubernetes version** used was **1.30.3**.