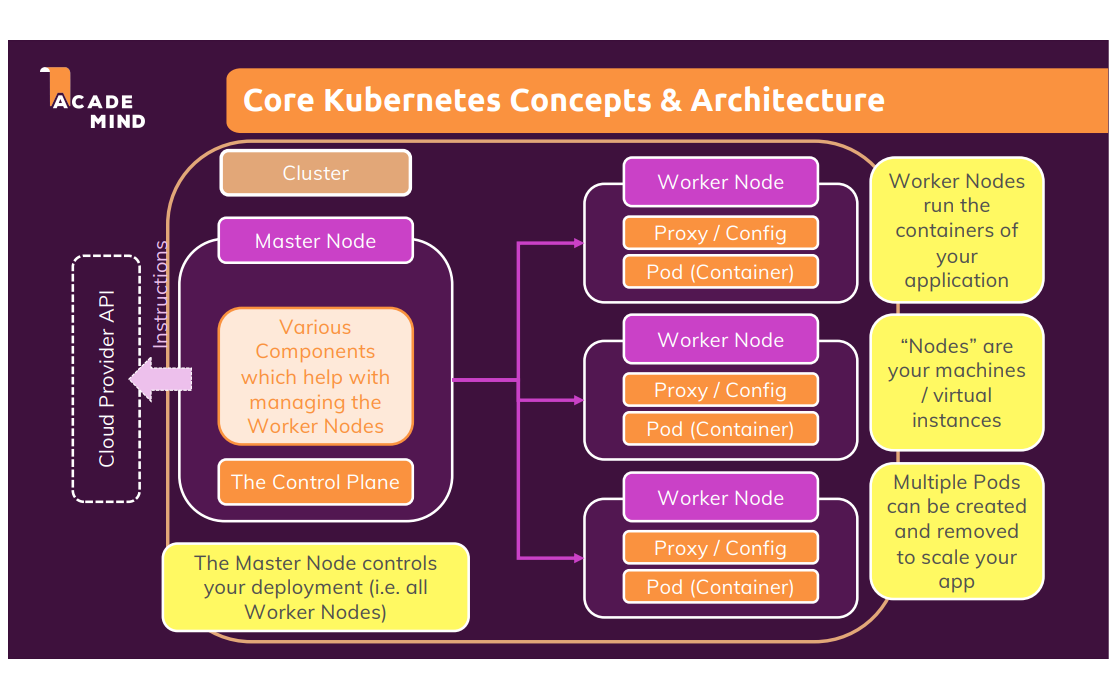
**KUBERNETES:**

**SECTION 1 : INTRODUCTION**

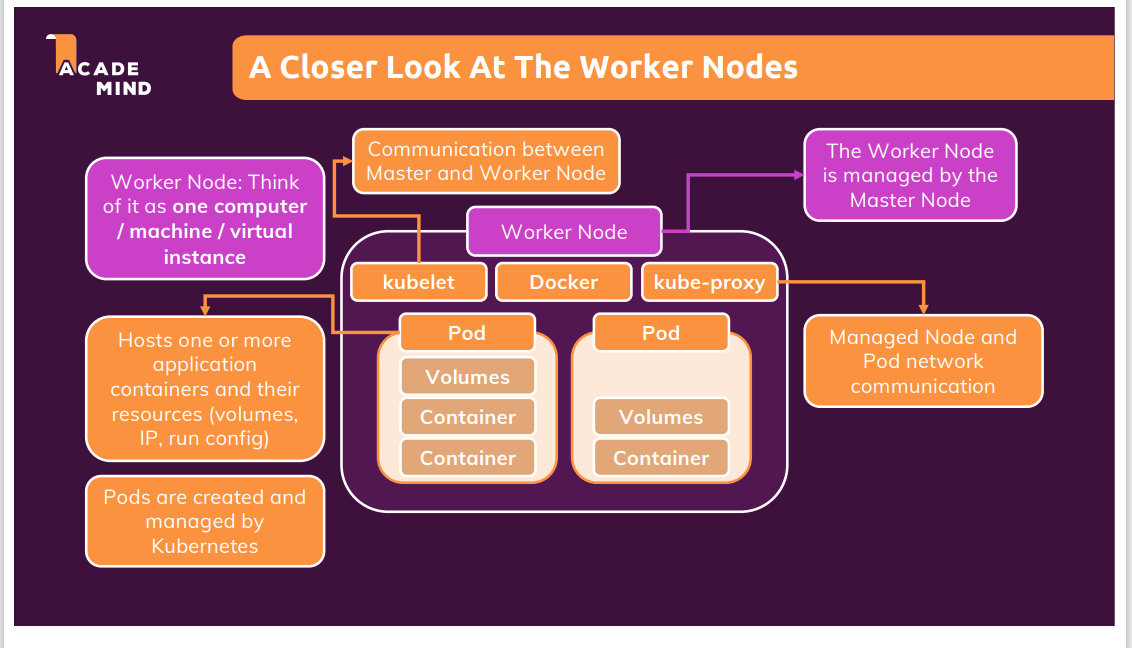
* An Open-source system for orchestrating container deployments

Kubernetes Architecture

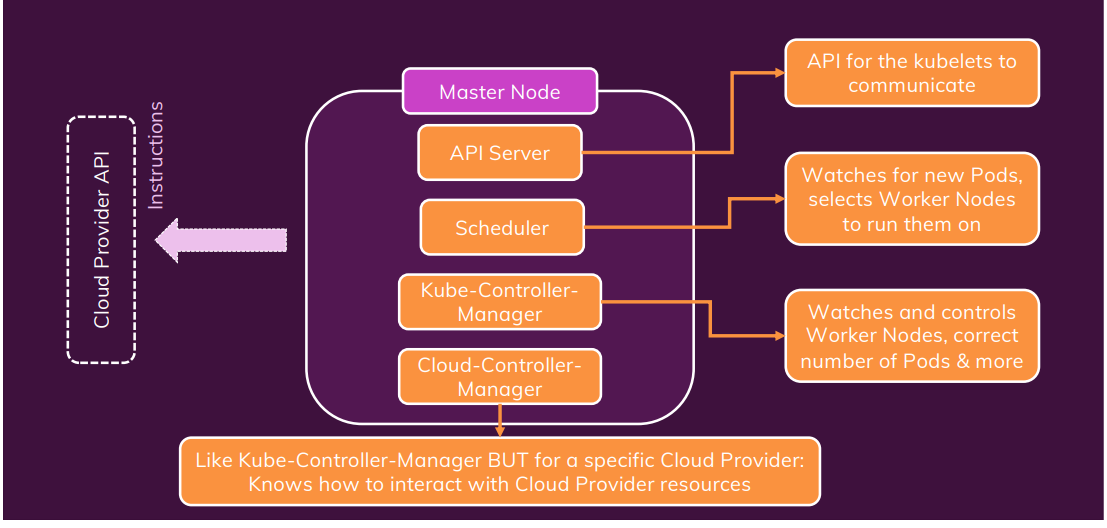


* Pods : Basic unit in Kubernetes

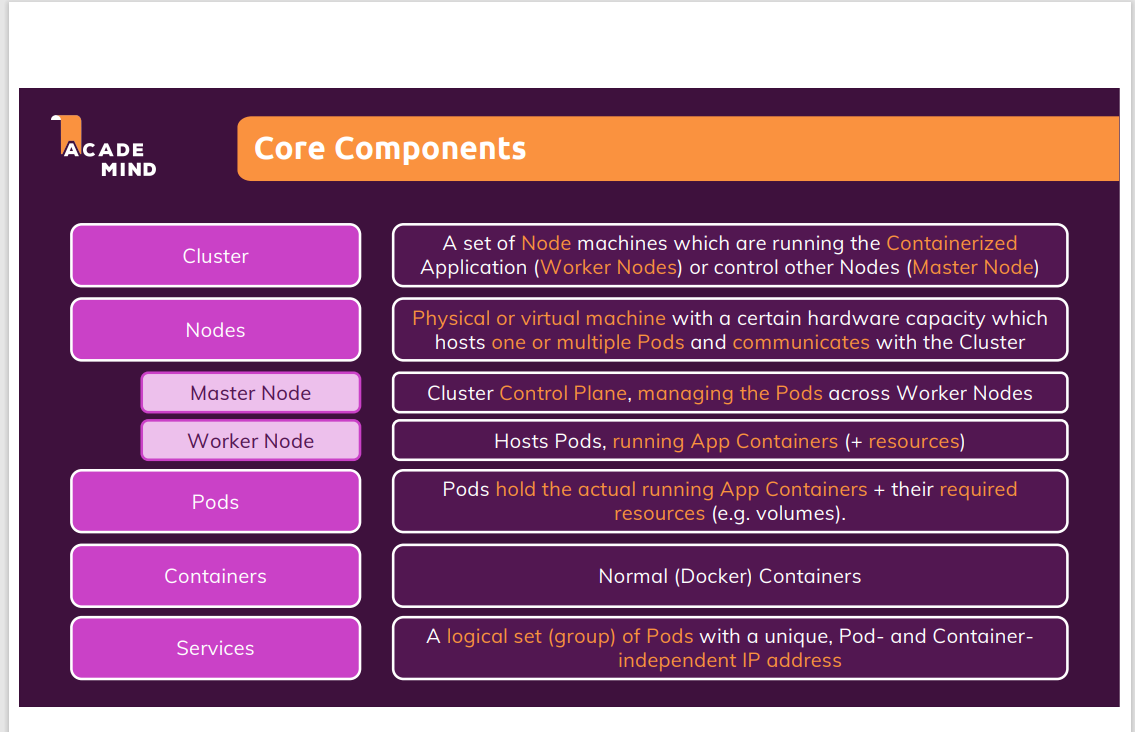
Worker node in details



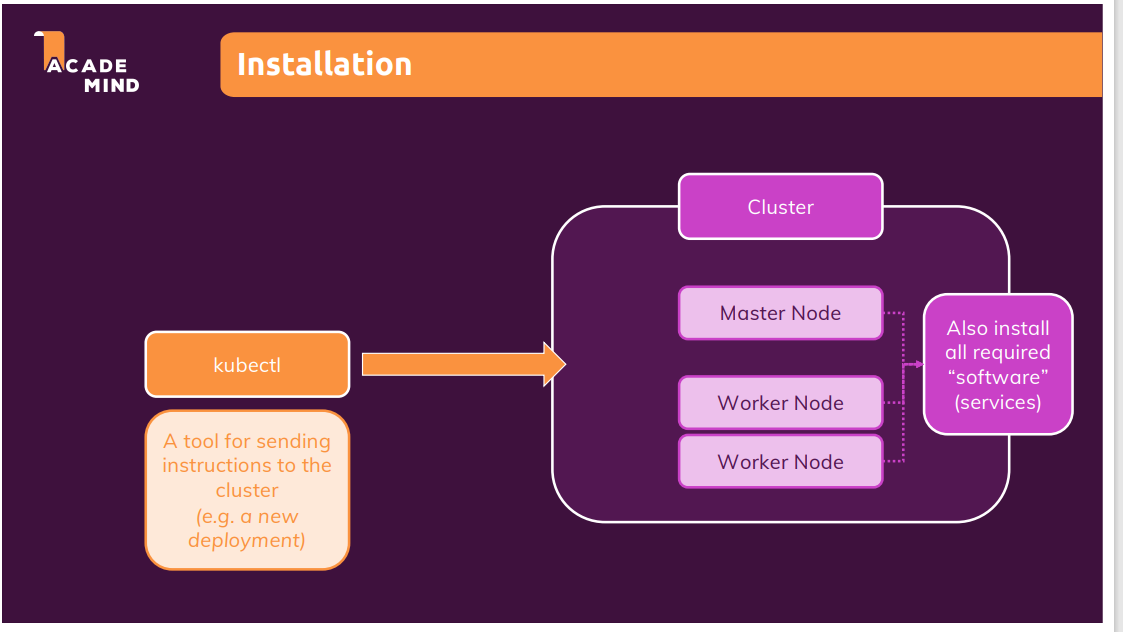
Master node in details



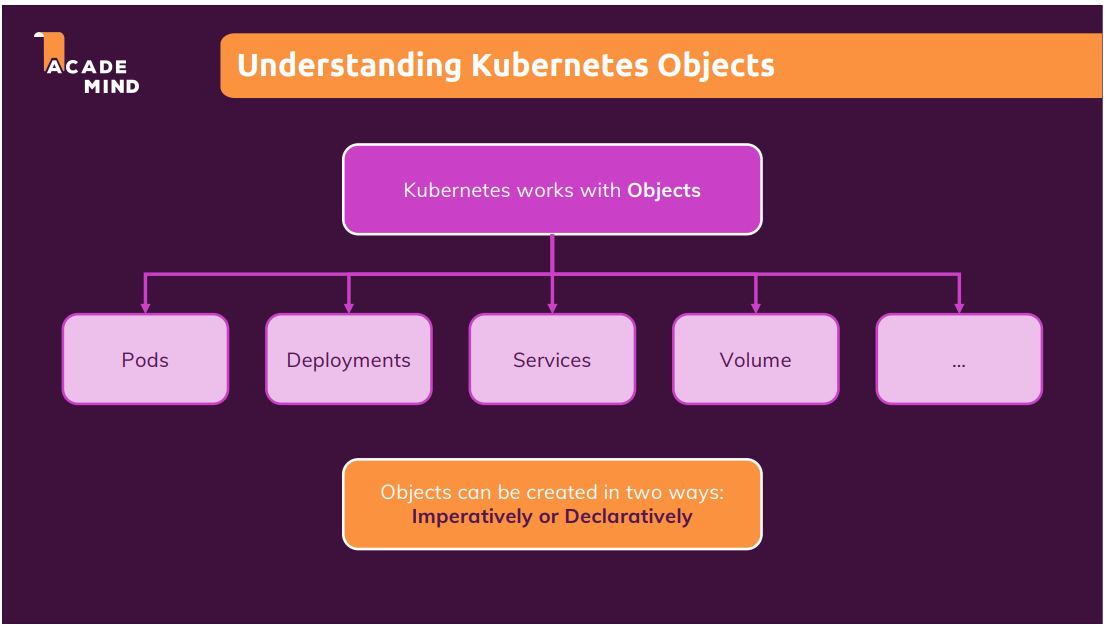
Core Concepts



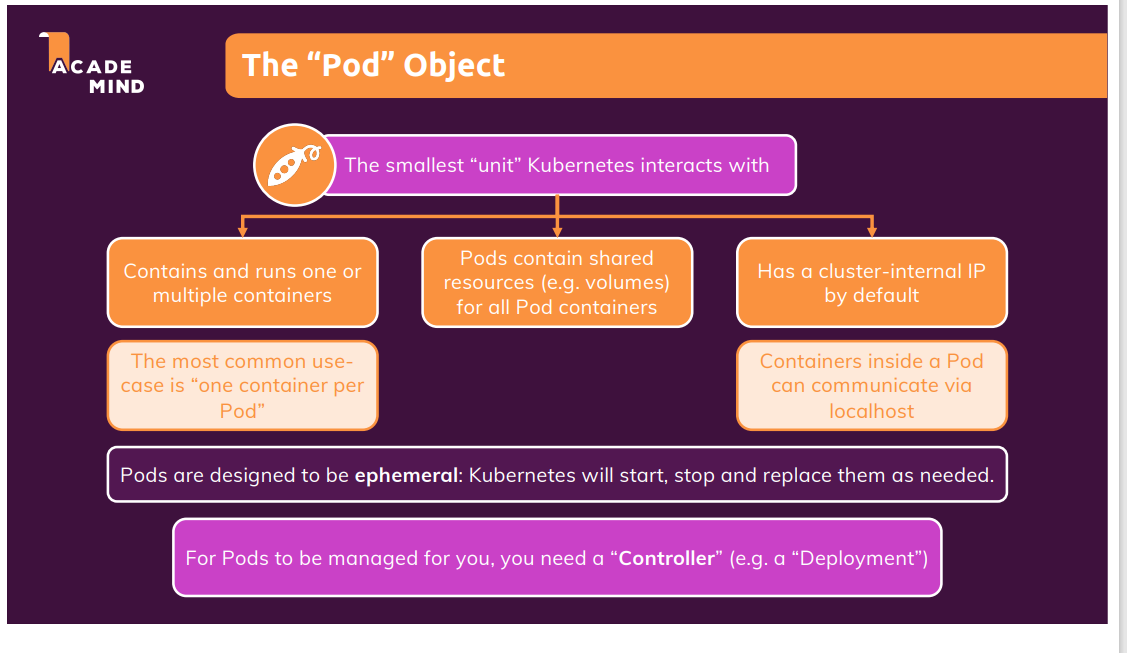
**SECTION 2: CORE CONCEPTS EXPLAINED**



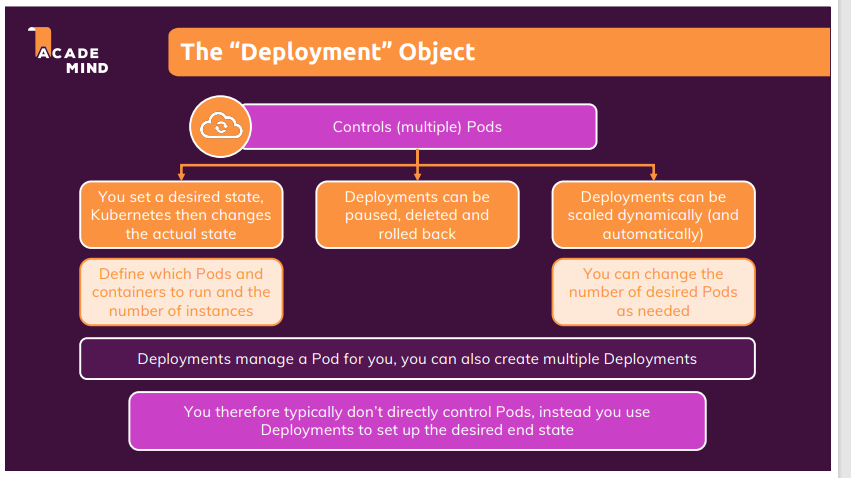
Understanding Kubernetes objects



POD

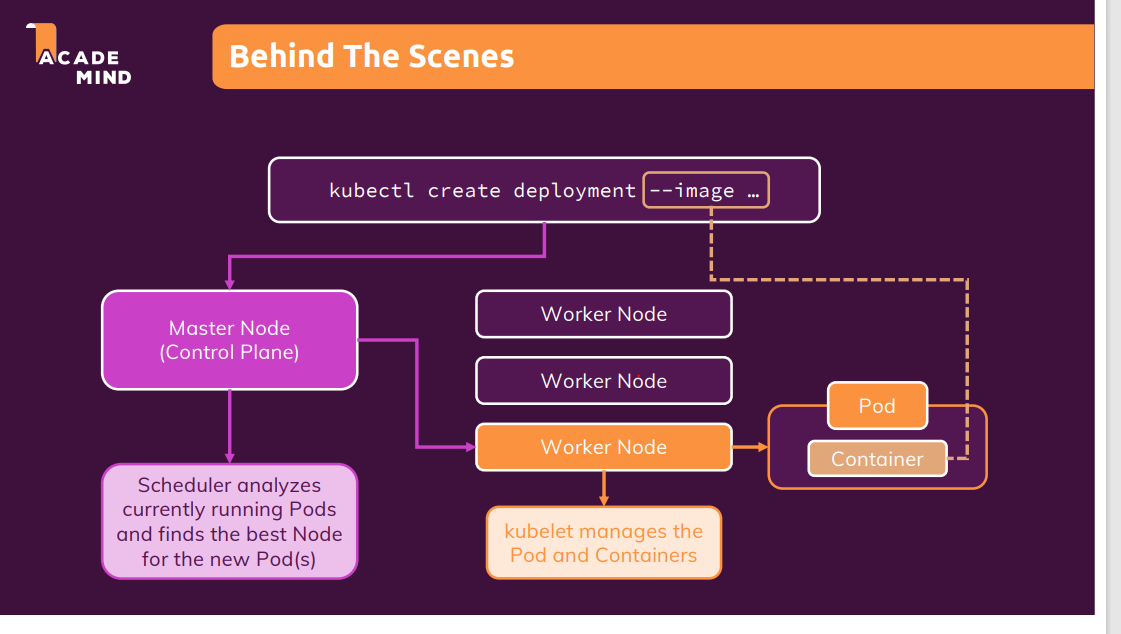


Deployment



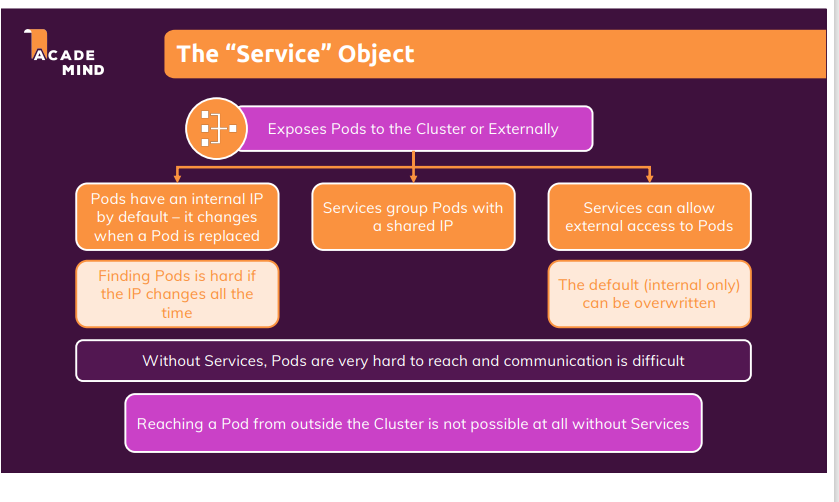
Deployment Example

* Kubectl create deployment deployment-name –image=image-name # the image should be in any repository like docker hub
* Kubectl get deployments: for getting all available deployments
* Kubectl get pods: for getting all available pods
* Kubectl delete deployment deployment-name : for delete the deployment
* minikube status : for getting minikube status
* minikube dashboard : for opening minikube dashboard



…………………………………………………………………………………………………………………………………………………………….

Services



Exposing a deployment with a service

* Kubectl expose deployment deployment-name –type=LoadBalancer –port=8080 :Command to expose our deployment . # the type can be Cluster IP ,NodePort,LoadBalancer etc..
* Kubectl get services : command to get all available services

Scaling the deployment

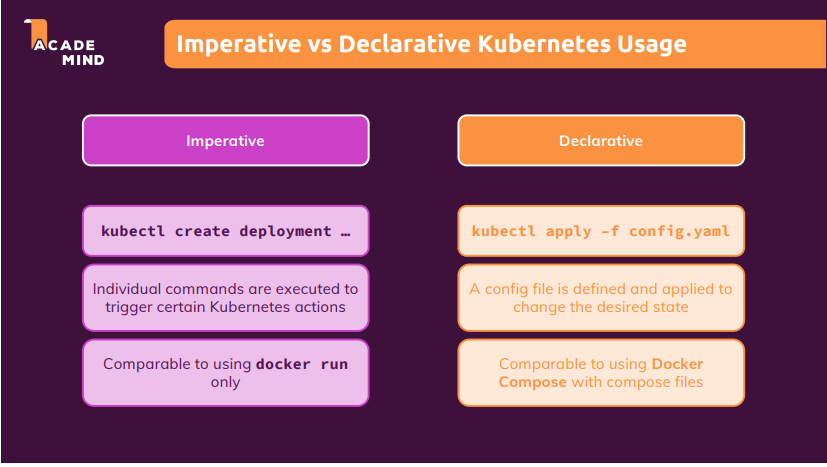
* Kubectl scale deployment/deployment-name –replicas=3 #replicas mean number of pods we need to run

Updating the Deployment

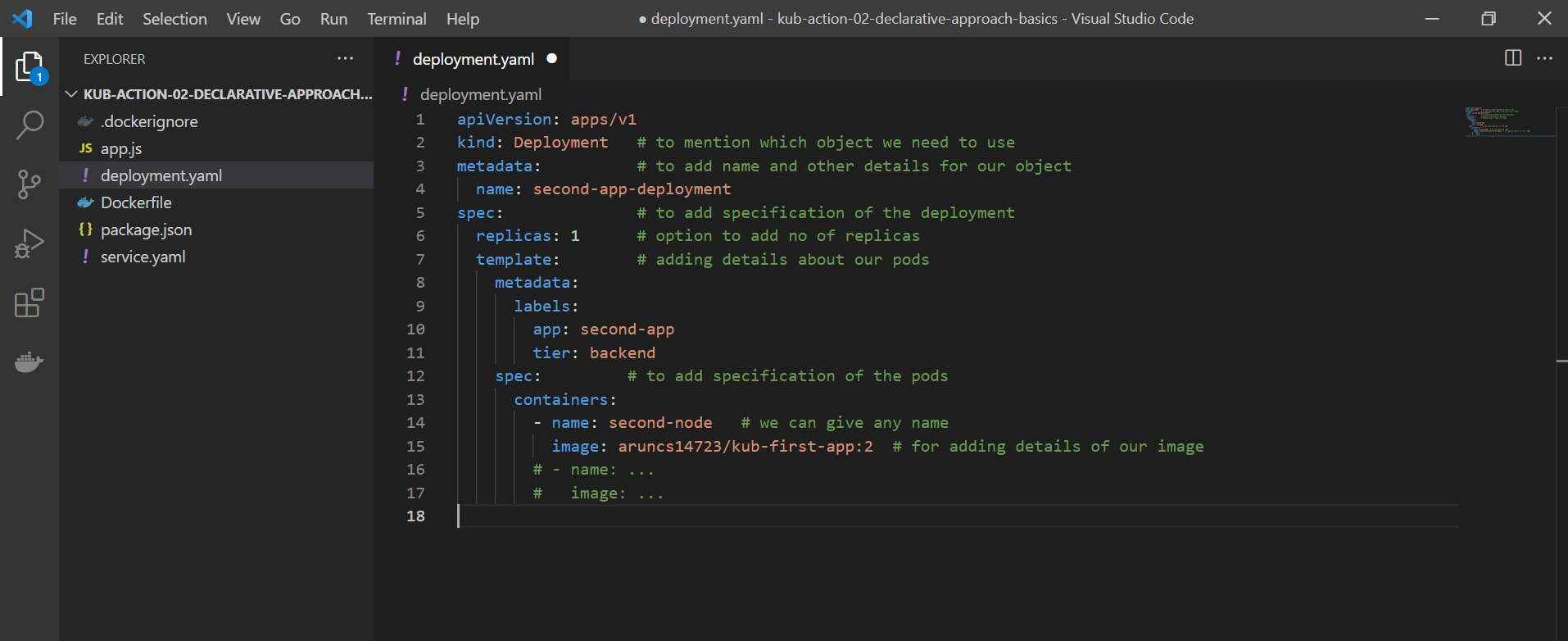
* Kubectl set image deployments/deployment-name previous created container-name=aruncs1472/kub-first-app : command to update the deployment #we should change the tag of the image before pushing into docker hub , otherwise it wont use latest image.
* Kubectl rollout status deployment/deployment-name : to view the updating status of our deployment
* Kubectl rollout undo deployment/deployment-name : Command for undo the latest deployment
* Kubectl rollout history deployment/deployment-name : command to get the deployment history.
* Kubectl rollout undo deployment/deployment-name –to-revision=1 : Command to go back to particular revision

………………………………………………………………………………………………………………………………………………………

Imperative vs Declarative approach



Creating a Deployment configuration file



apiVersion: apps/v1

kind: Deployment   # to mention which object we need to use

metadata:          # to add name and other details for our object

  name: second-app-deployment

spec:              # to add specification of the deployment

  replicas: 1      # option to add no of replicas

  template:        # adding details about our pods

    metadata:

      labels:

        app: second-app

        tier: backend

    spec:         # to add specification of the pods

      containers:

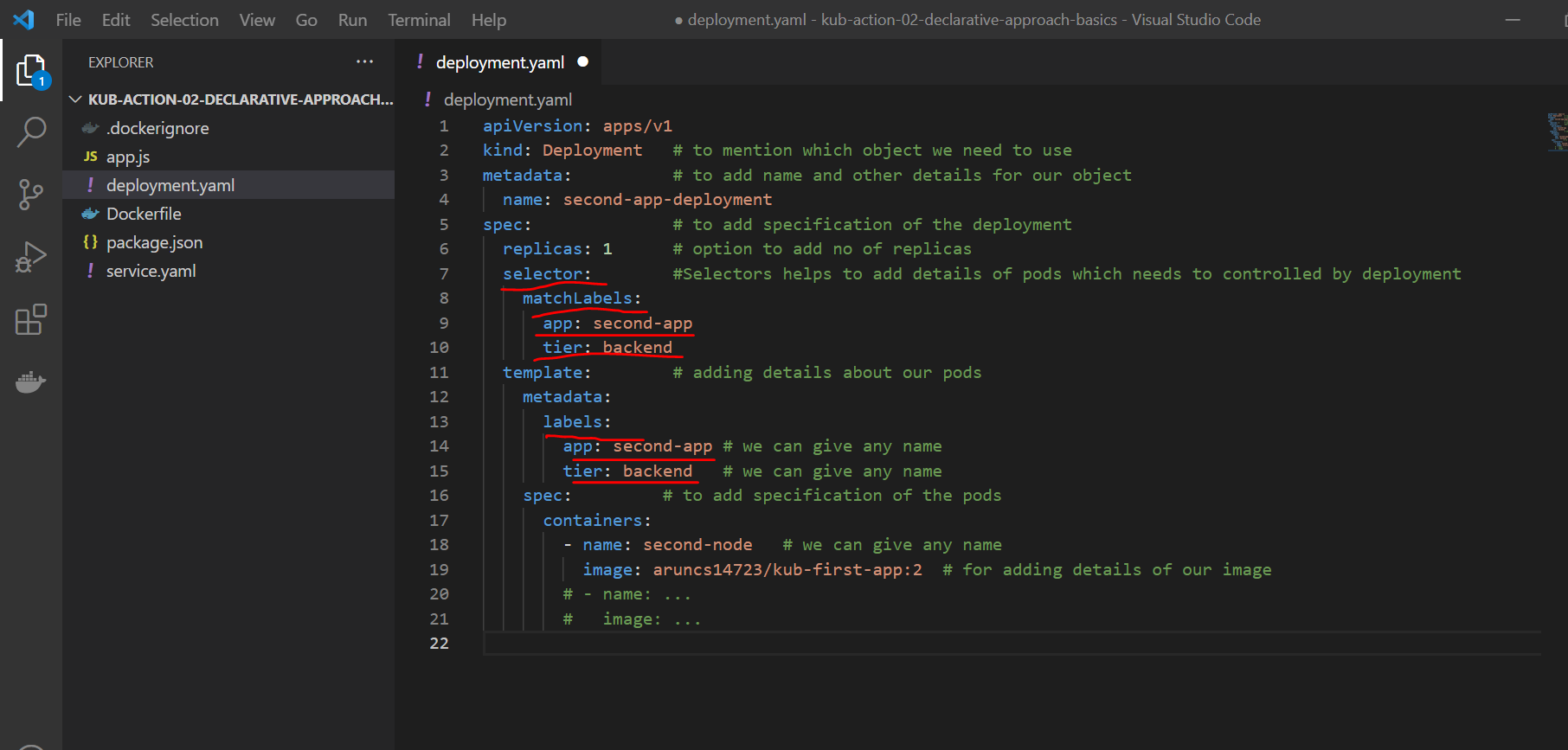
        - name: second-node   # we can give any name

          image: aruncs14723/kub-first-app:2  # for adding details of our image

        # - name: ...

        #   image: ...

Working with labels and selectors



apiVersion: apps/v1

kind: Deployment   # to mention which object we need to use

metadata:          # to add name and other details for our object

  name: second-app-deployment

spec:              # to add specification of the deployment

  replicas: 1      # option to add no of replicas

  selector:        #Selectors helps to add details of pods which needs to controlled by deployment

    matchLabels:

      app: second-app

      tier: backend

  template:        # adding details about our pods

    metadata:

      labels:

        app: second-app # we can give any name

        tier: backend   # we can give any name

    spec:         # to add specification of the pods

      containers:

        - name: second-node   # we can give any name

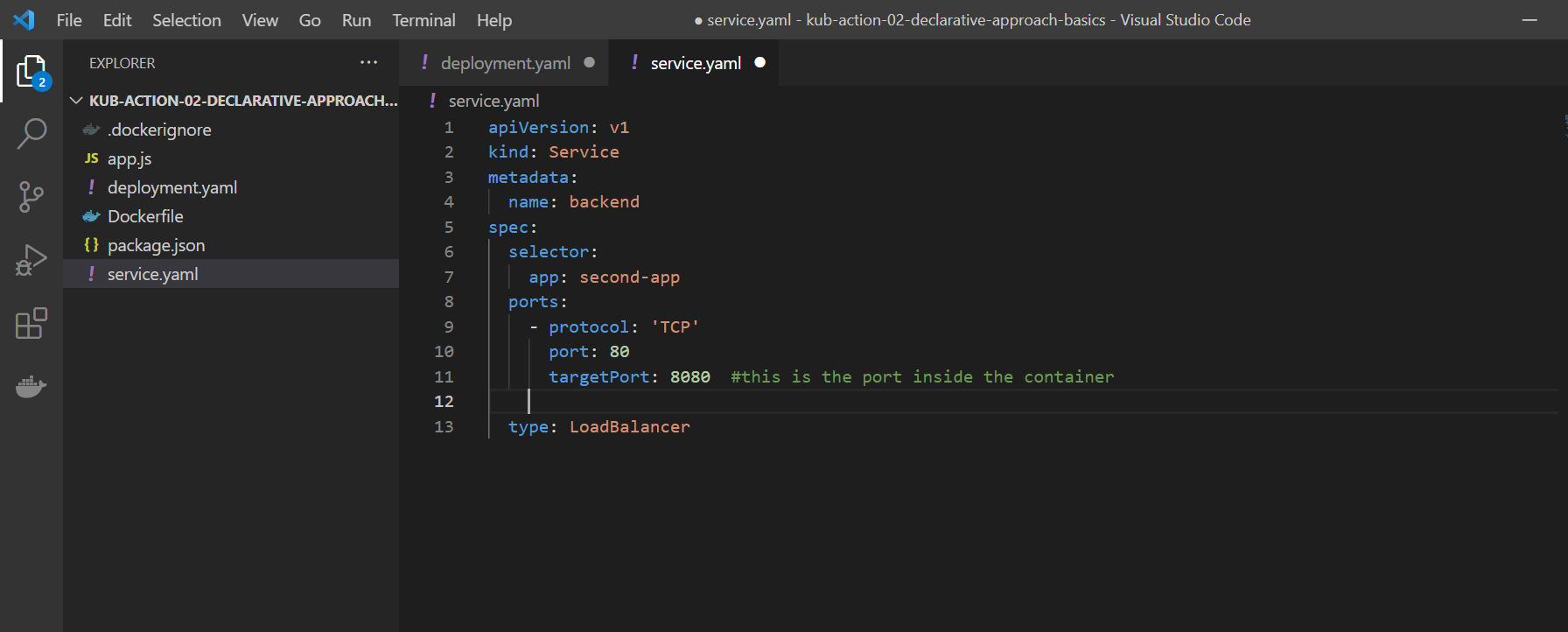
          image: aruncs14723/kub-first-app:2  # for adding details of our image

        # - name: ...

        #   image: ...

* In the above example selectors helps to tell deployment to manage pods which is having Labels , app:second-app ,tier:backend.
* Kubectl apply -f deployment.yaml : command to run the deployment file
* Kubectl delete -f deployment.yaml : Command to delete resources which is created using deployment file

Creating a service Declaratively



apiVersion: v1

kind: Service

metadata:

  name: backend

spec:

  selector:

    app: second-app

  ports:

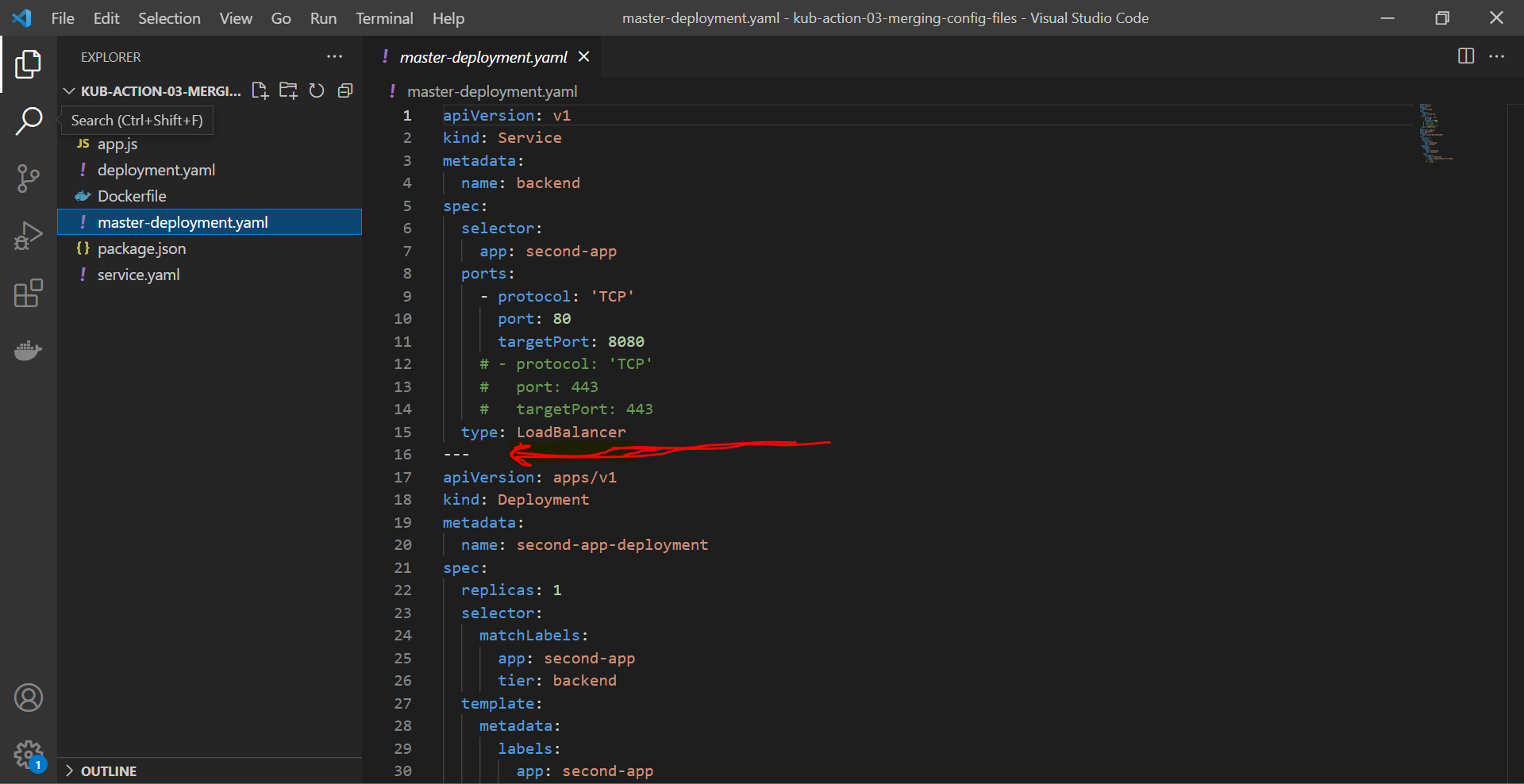
    - protocol: 'TCP'

      port: 80

      targetPort: 8080  #this is the port inside the container

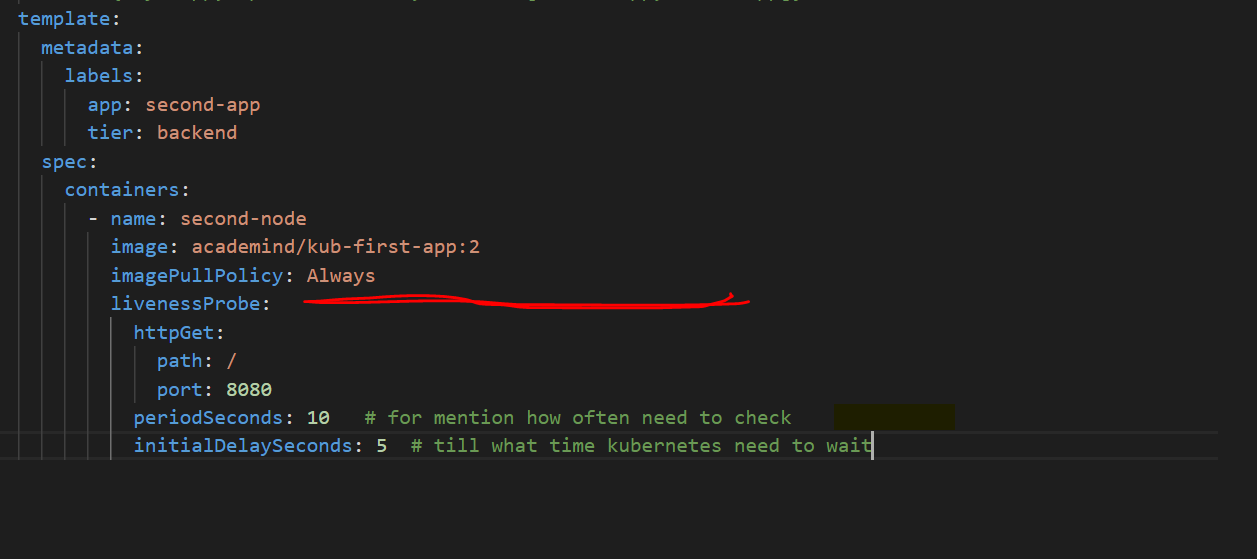
  type: LoadBalancer

Merging the configuration files



Liveness Probes Option

* This option helps to check whether our Pods and containers are up and running.



  template:

    metadata:

      labels:

        app: second-app

        tier: backend

    spec:

      containers:

        - name: second-node

          image: academind/kub-first-app:2

          imagePullPolicy: Always

          livenessProbe:

            httpGet:

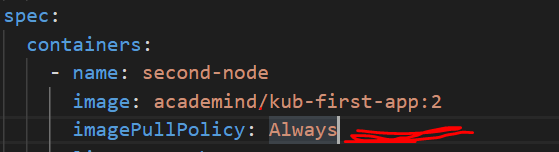
              path: /

              port: 8080

            periodSeconds: 10   # for mention how often need to check

            initialDelaySeconds: 5  # till what time kubernetes need to wait

………………………………………………………………………………………………………………………………………………………..

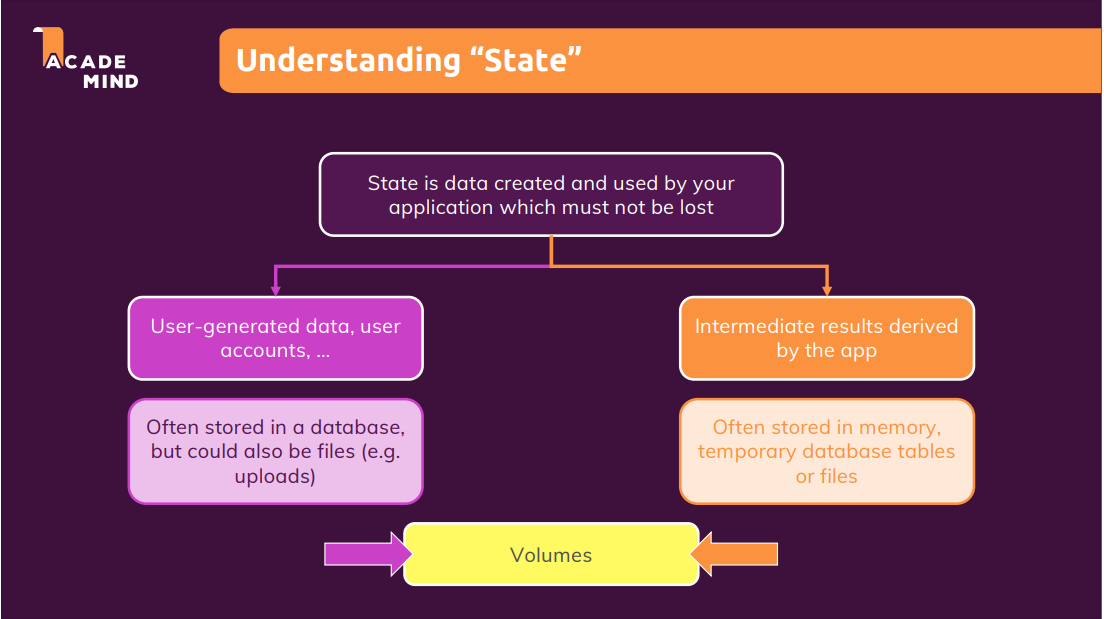


* This ImagePullpolicy option helps to pull updated images. That is once we pushing the latest image to docker hub , we don’t need to give different tag.

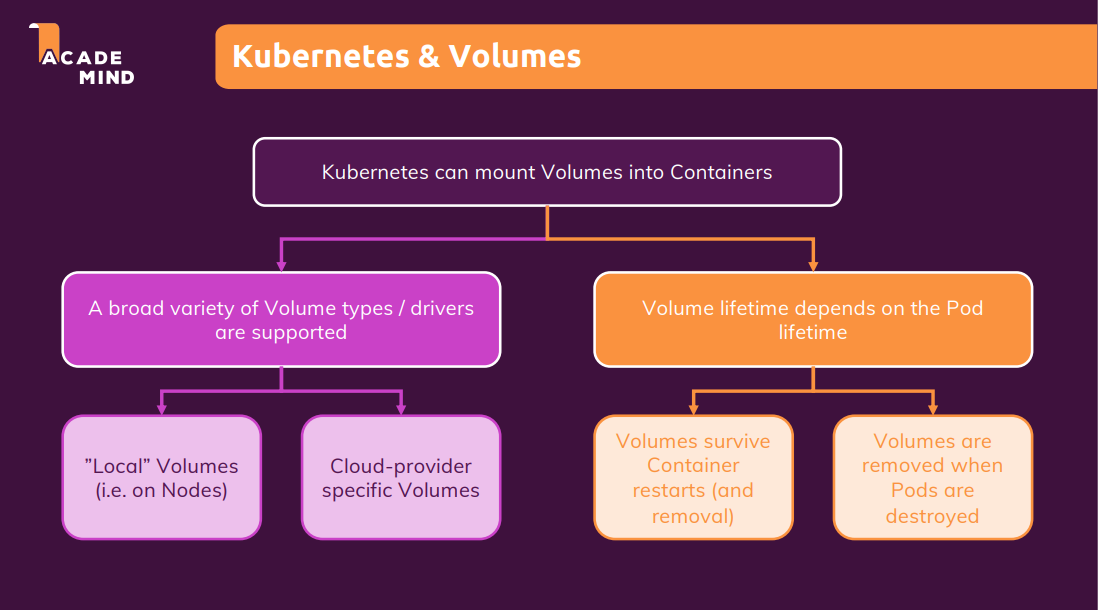
**SECTION 3: Managing Data and Volumes with Kubernetes**

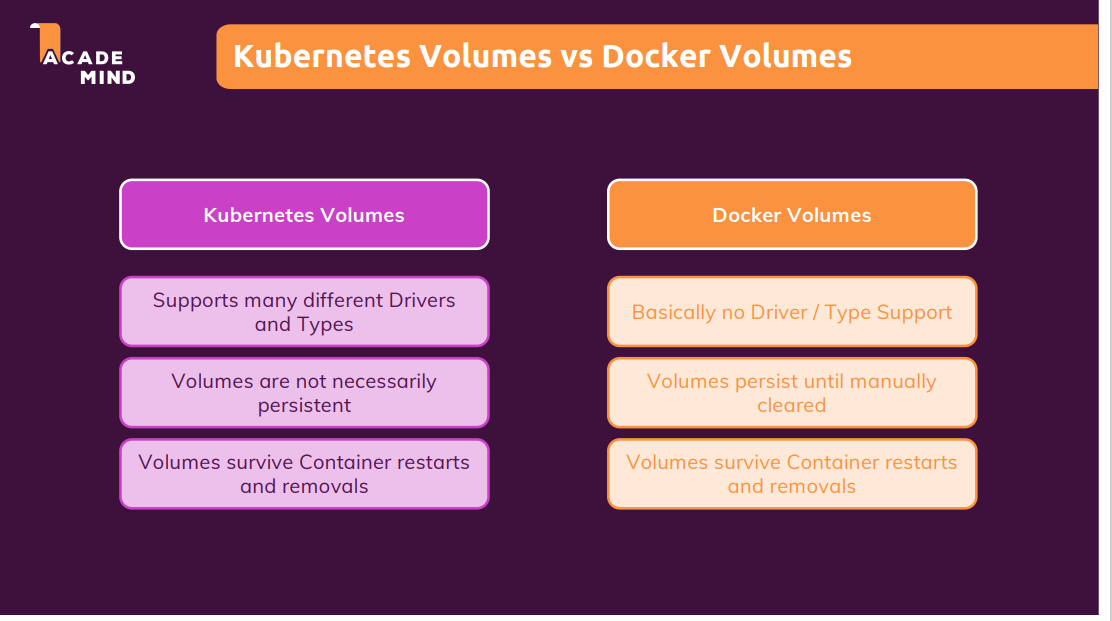
**State:**

* State is data created and used by your application which must not be lost

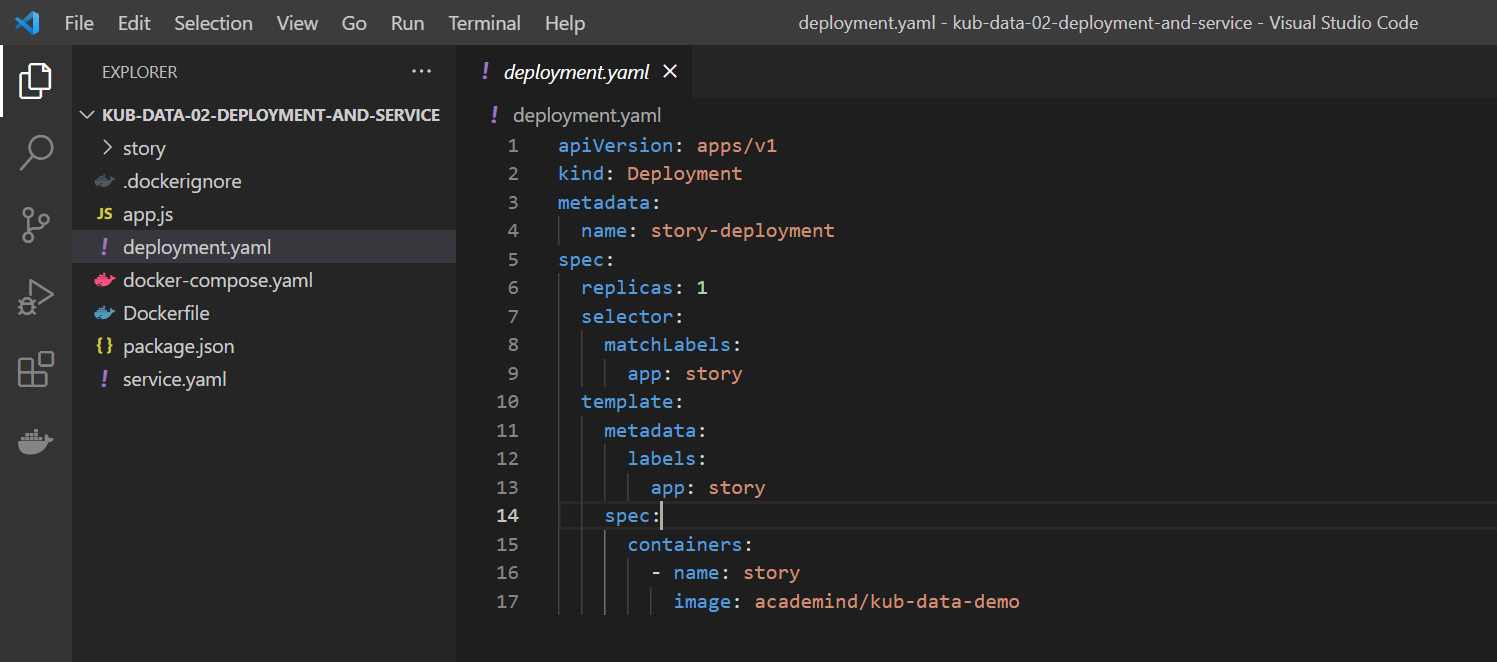


Kubernetes Volumes VS Docker Volume





Creating New Deployment and service



apiVersion: apps/v1

kind: Deployment

metadata:

  name: story-deployment

spec:

  replicas: 1

  selector:

    matchLabels:

      app: story

  template:

    metadata:

      labels:

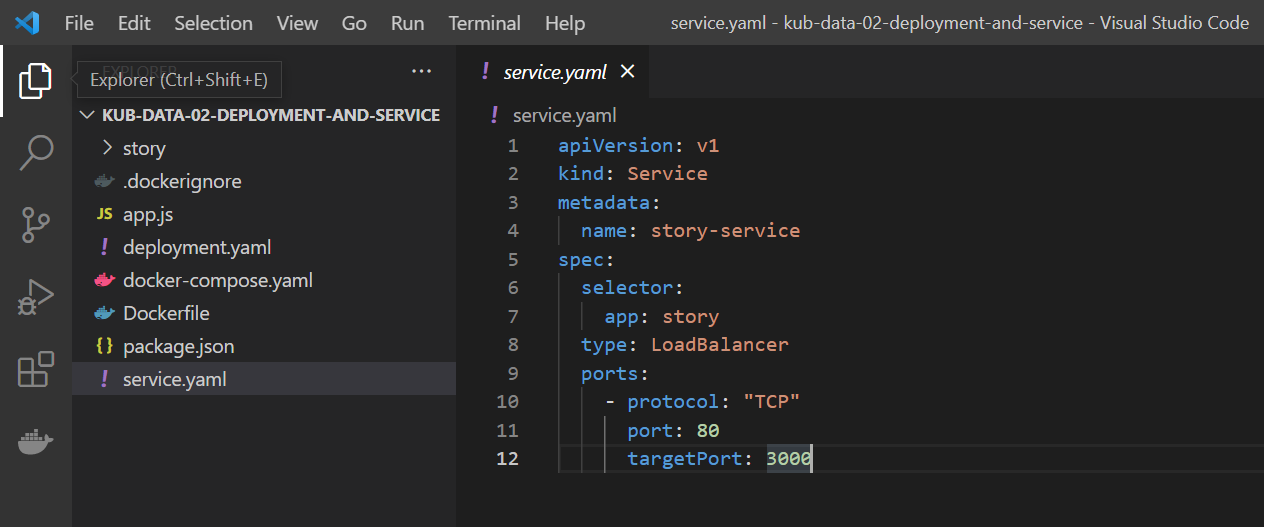
        app: story

    spec:

      containers:

        - name: story

          image: academind/kub-data-demo



apiVersion: v1

kind: Service

metadata:

  name: story-service

spec:

  selector:

    app: story

  type: LoadBalancer

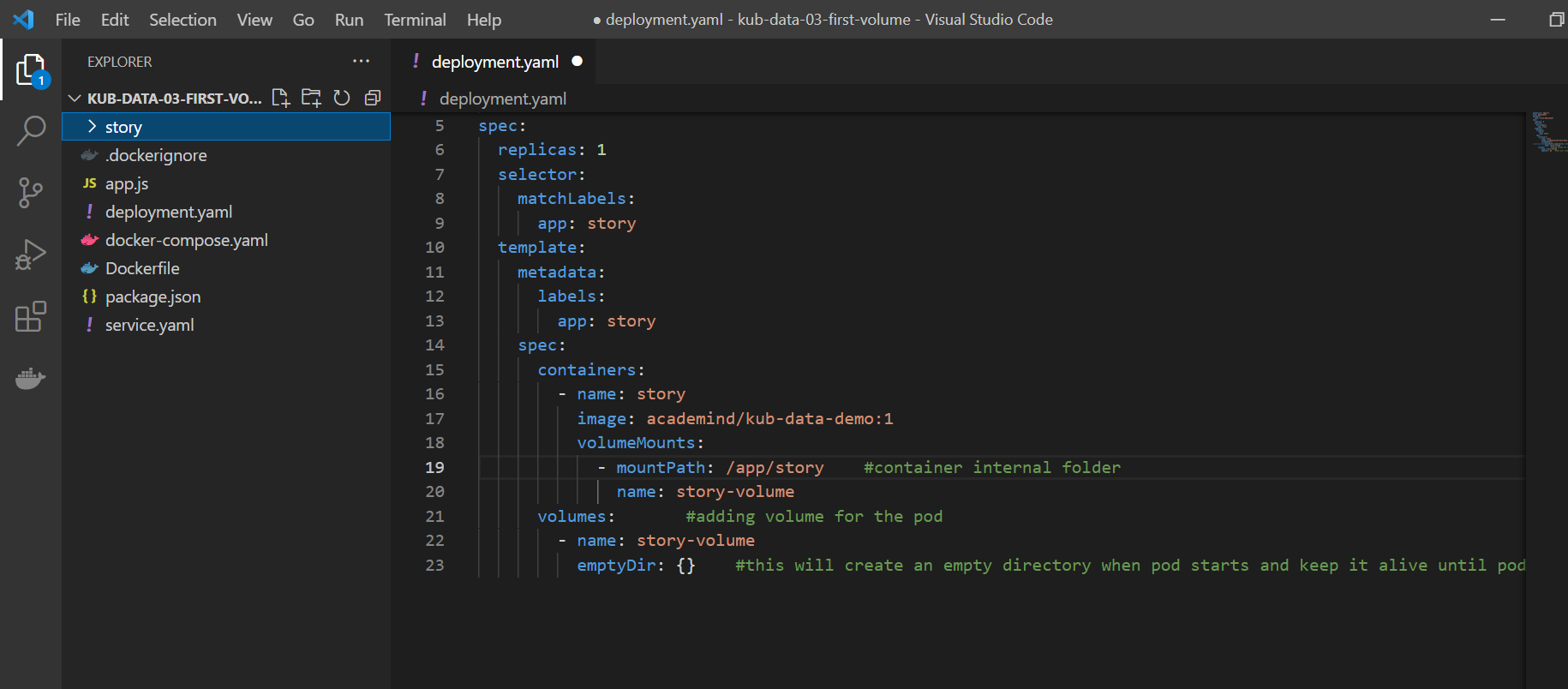
  ports:

    - protocol: "TCP"

      port: 80

      targetPort: 3000

Creating Volume: emptyDir type



spec:

      containers:

        - name: story

          image: academind/kub-data-demo:1

          volumeMounts:

            - mountPath: /app/story    #container internal folder

              name: story-volume

      volumes:       #adding volume for the pod

        - name: story-volume

          emptyDir: {}    #this will create an empty directory when pod starts

and keep it alive until pod is active

* emptyDir approach is only good if we are using single replica.

Creating Another volume type : hostPath type

* It will help with multiple replicas , it’s using local host machine.
* It will be only used by pods within same node ,not other nodes.

spec:

      containers:

        - name: story

          image: academind/kub-data-demo:1

          volumeMounts:

            - mountPath: /app/story    #container internal folder

              name: story-volume

      volumes:       #adding volume for the pod

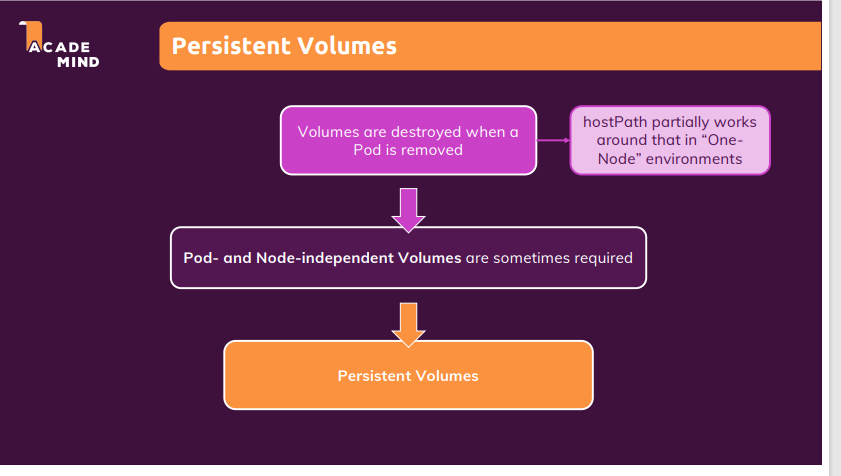
        - name: story-volume

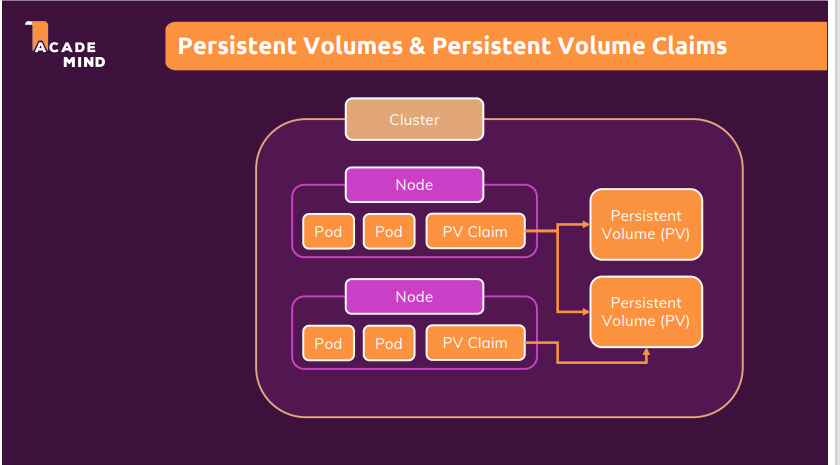
          hostPath:

            path: /data

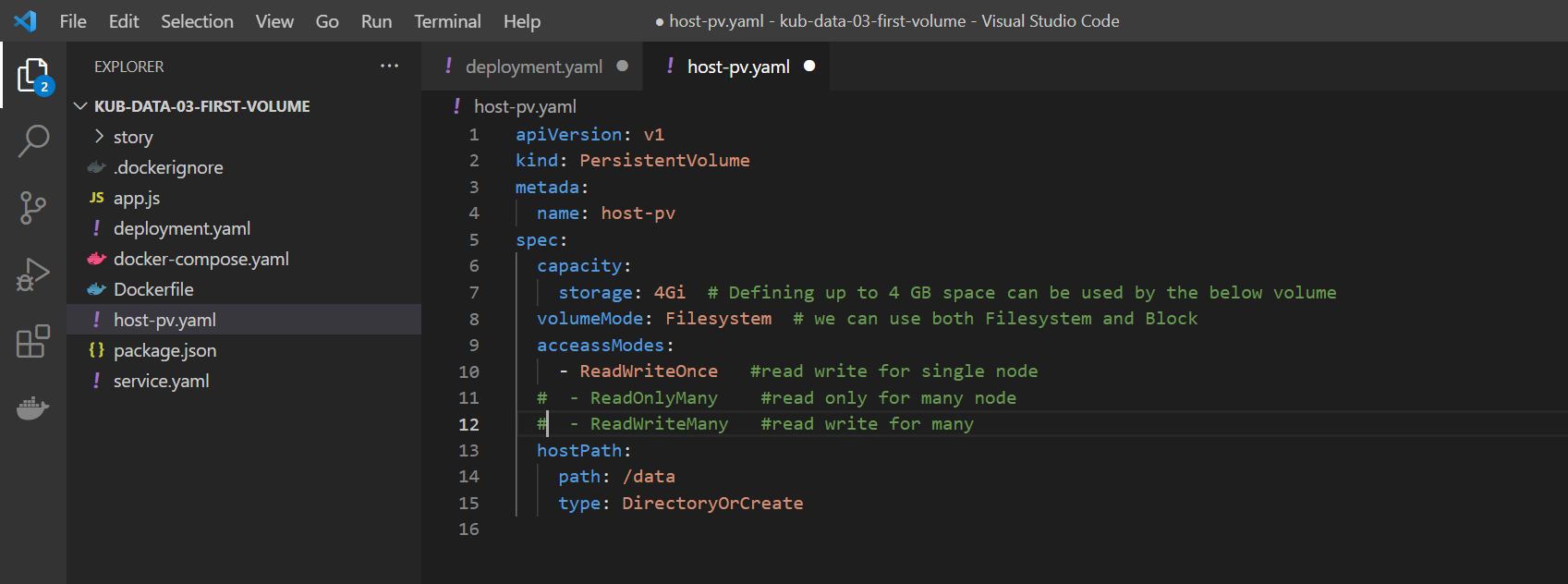
            type: DirectoryOrCreate

Persistent Volume





Defining a Persistent volume



apiVersion: v1

kind: PersistentVolume

metada:

  name: host-pv

spec:

  capacity:

    storage: 4Gi  # Defining up to 4 GB space can be used by the below volume

  volumeMode: Filesystem  # we can use both Filesystem and Block

  acceassModes:

    - ReadWriteOnce   #read write for single node

  #  - ReadOnlyMany    #read only for many node

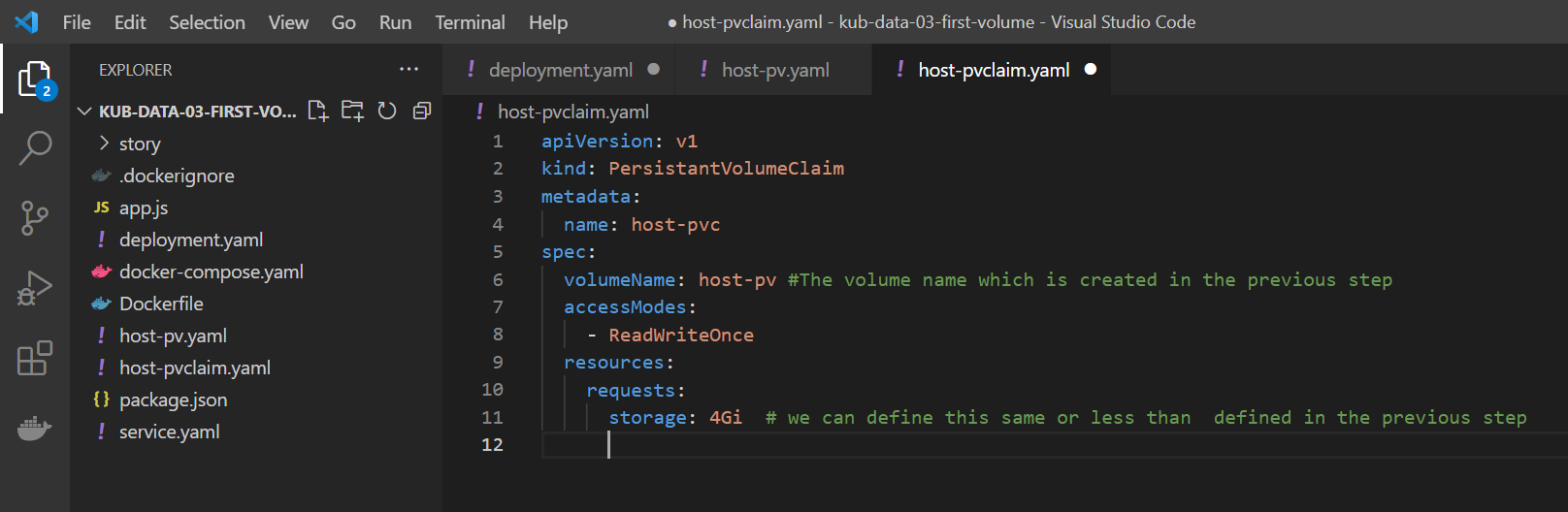
  #  - ReadWriteMany   #read write for many

  hostPath:

    path: /data

    type: DirectoryOrCreate

Creating a Persistent Volume (Creating a Persistent volume claim)



apiVersion: v1

kind: PersistantVolumeClaim

metadata:

  name: host-pvc

spec:

  volumeName: host-pv #The volume name which is created in the previous step

  accessModes:

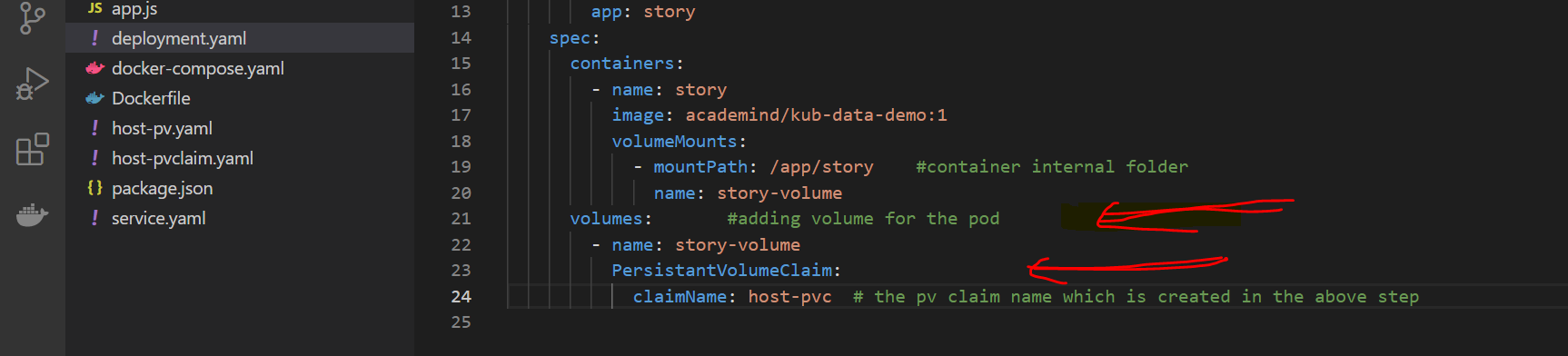
    - ReadWriteOnce

  resources:

    requests:

      storage: 4Gi  # we can define this same or less than  defined in the previous step

* now we need to edit the volume section in the deployment file



spec:

      containers:

        - name: story

          image: academind/kub-data-demo:1

          volumeMounts:

            - mountPath: /app/story    #container internal folder

              name: story-volume

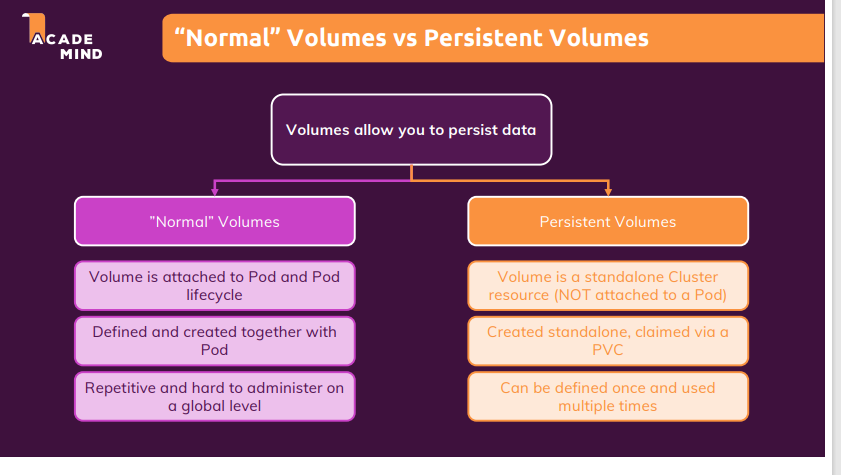
      volumes:       #adding volume for the pod

        - name: story-volume

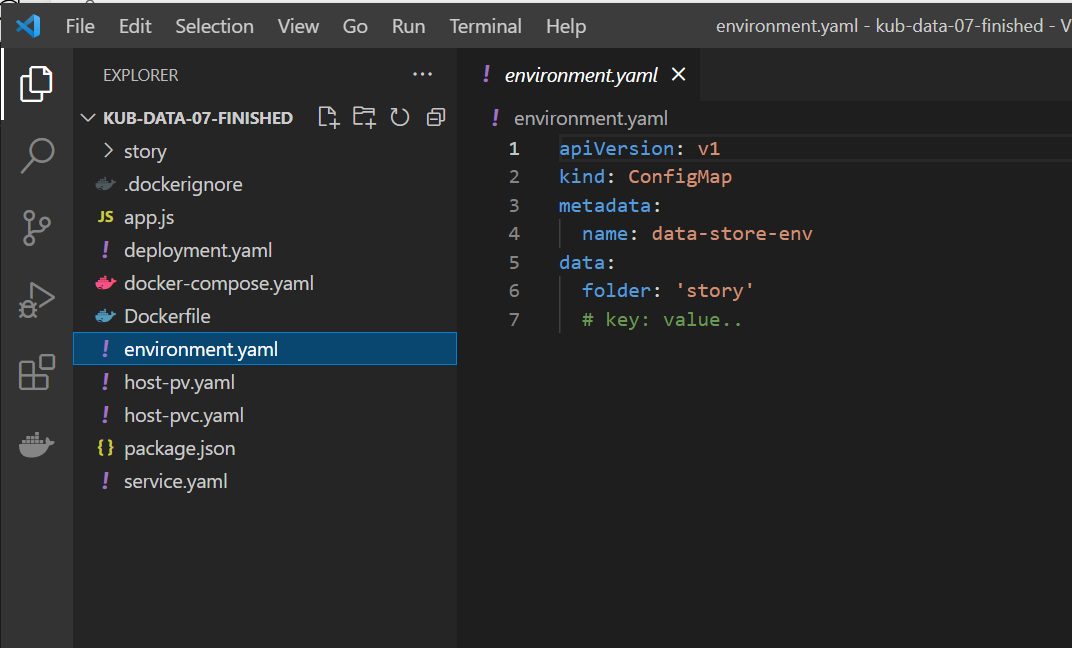
          PersistantVolumeClaim:

            claimName: host-pvc  # the pv claim name which is created in the above step

Normal Volumes VS persistent Volume



Environment Variables and Config map



apiVersion: v1

kind: ConfigMap

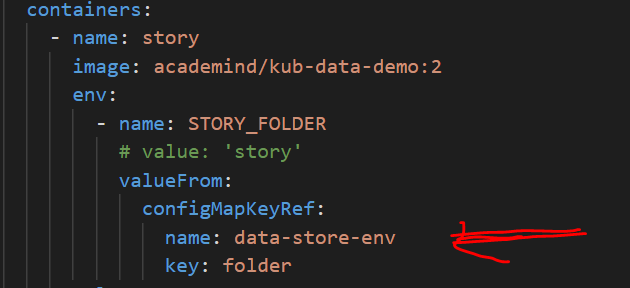
metadata:

  name: data-store-env

data:

  folder: 'story'

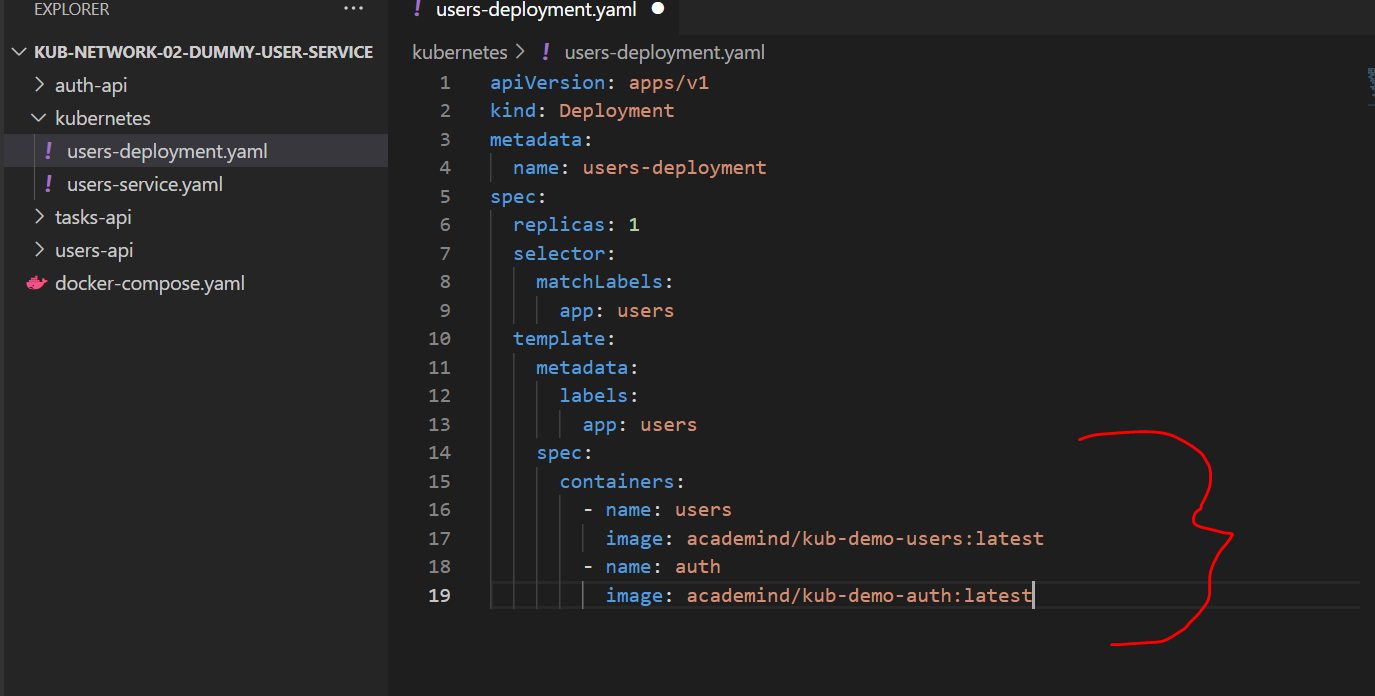
  # key: value..



* above is the example of configuring ConfigMapKey inside Deployment.yaml file
* env:
* - name: STORY\_FOLDER
* # value: 'story'
* valueFrom:
* configMapKeyRef:
* name: data-store-env
* key: folder

**SECTION 4: Kubernetes Networking**

1. Multiple containers in one pod



spec:

  replicas: 1

  selector:

    matchLabels:

      app: users

  template:

    metadata:

      labels:

        app: users

    spec:

      containers:

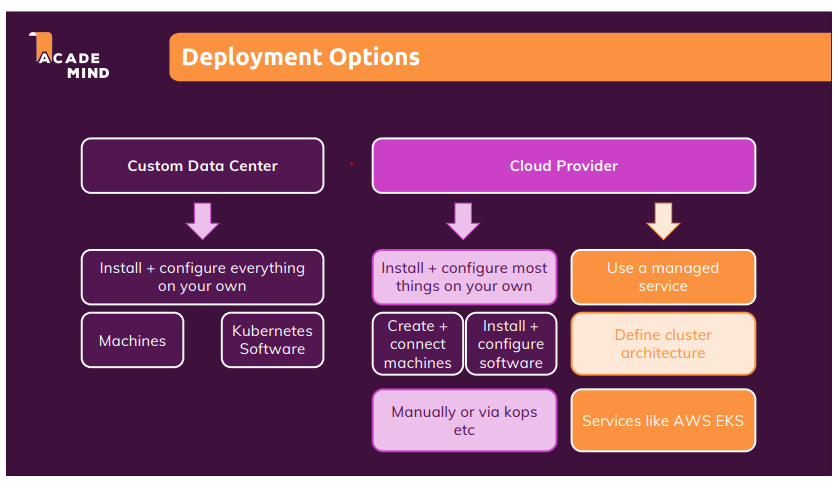
        - name: users

          image: academind/kub-demo-users:latest

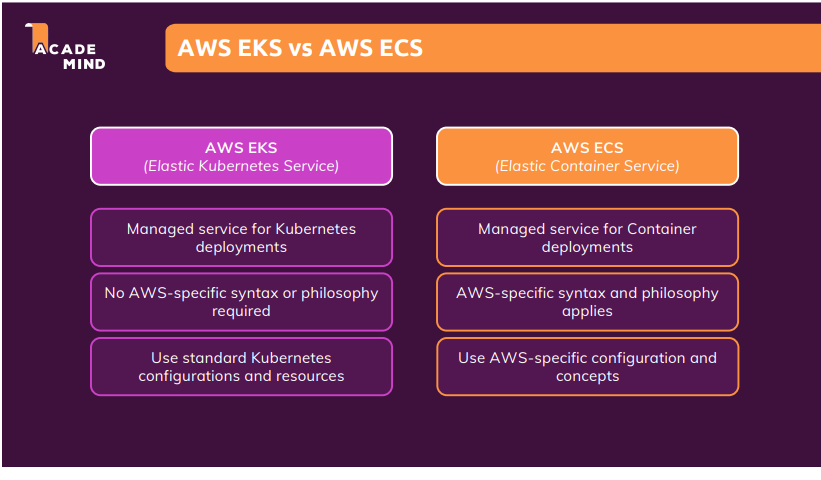
        - name: auth

          image: academind/kub-demo-auth:latest

**SECTION 5: Kubernetes Deployment options**



AWS ECS vs EKS



Project YAML files

apiVersion: v1

kind: Service

metadata:

  name: auth-service

spec:

  selector:

    app: auth

  type: ClusterIP

  ports:

    - protocol: TCP

      port: 3000

      targetPort: 3000

---

apiVersion: apps/v1

kind: Deployment

metadata:

  name: auth-deployment

spec:

  replicas: 1

  selector:

    matchLabels:

      app: auth

  template:

    metadata:

      labels:

        app: auth

    spec:

      containers:

        - name: auth-api

          image: academind/kub-dep-auth:latest

          env:

            - name: TOKEN\_KEY

              value: 'shouldbeverysecure'

apiVersion: v1

kind: Service

metadata:

  name: users-service

spec:

  selector:

    app: users

  type: LoadBalancer

  ports:

    - protocol: TCP

      port: 80

      targetPort: 3000

---

apiVersion: apps/v1

kind: Deployment

metadata:

  name: users-deployment

spec:

  replicas: 1

  selector:

    matchLabels:

      app: users

  template:

    metadata:

      labels:

        app: users

    spec:

      containers:

        - name: users-api

          image: academind/kub-dep-users:latest

          env:

            - name: MONGODB\_CONNECTION\_URI

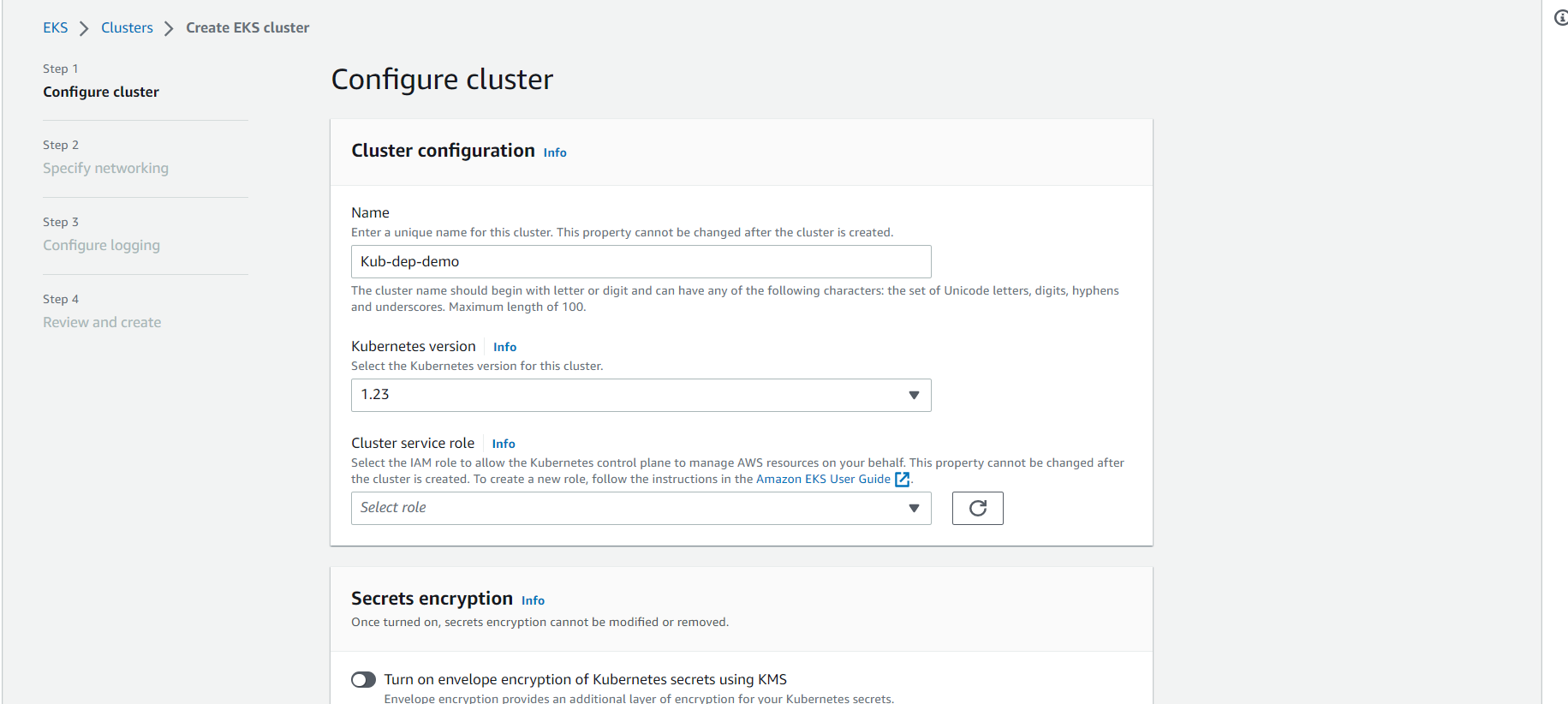
              value: 'mongodb+srv://maximilian:wk4nFupsbntPbB3l@cluster0.ntrwp.mongodb.net/users?retryWrites=true&w=majority'

            - name: AUTH\_API\_ADDRESSS

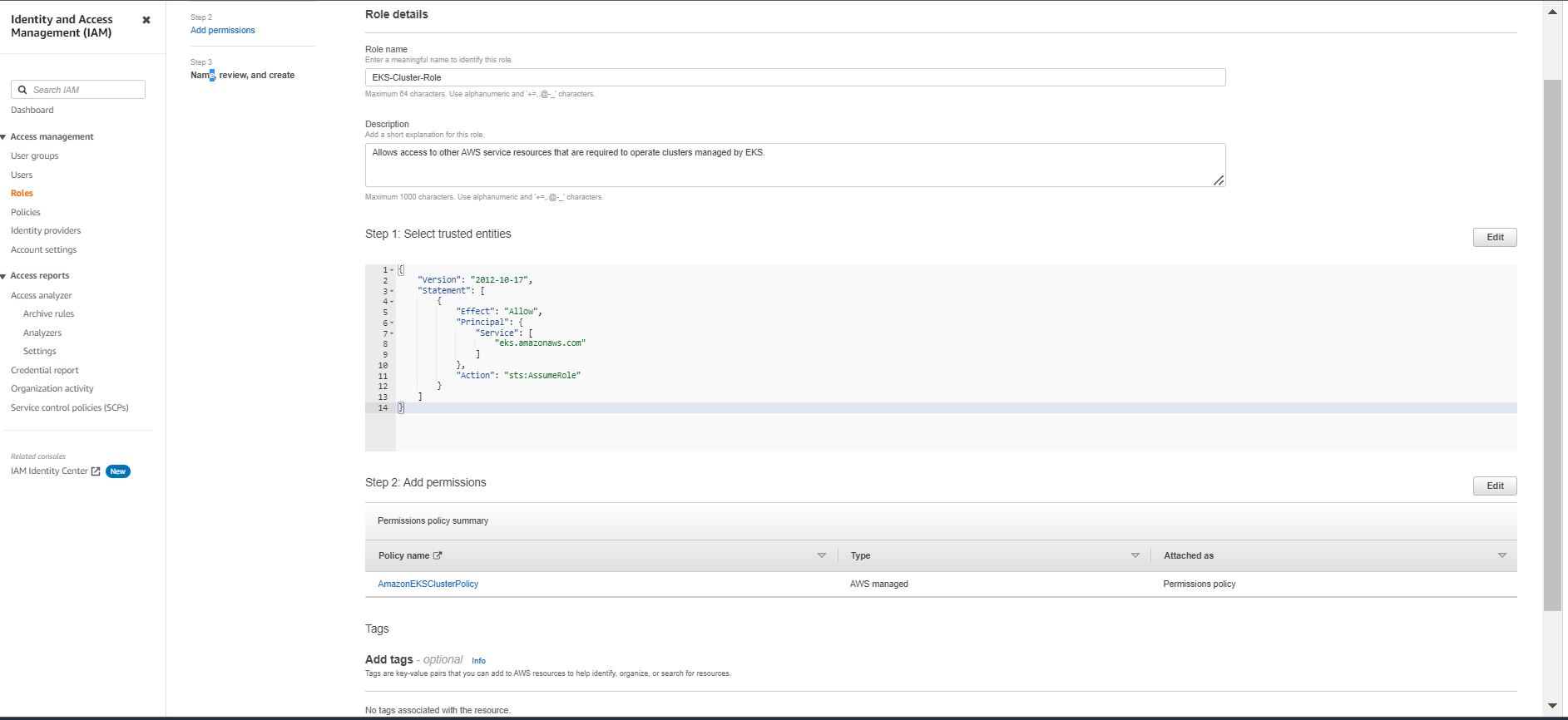
              value: 'auth-service.default:3000'

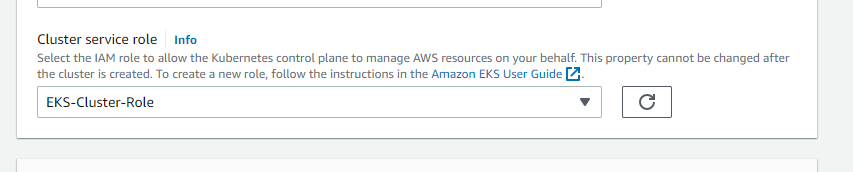
Creating and Configuring Kubernetes Cluster with EKS

* Create a new cluster with following details

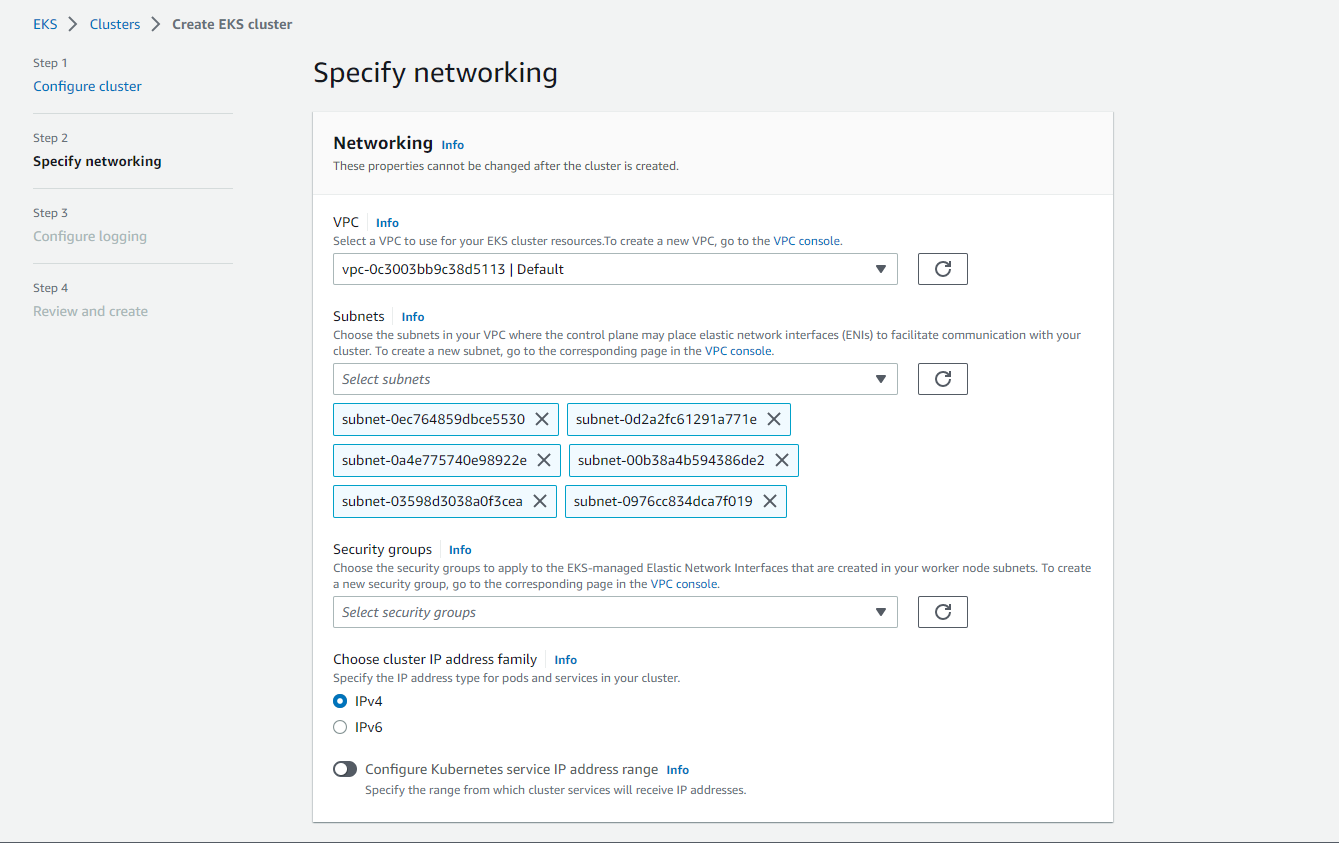


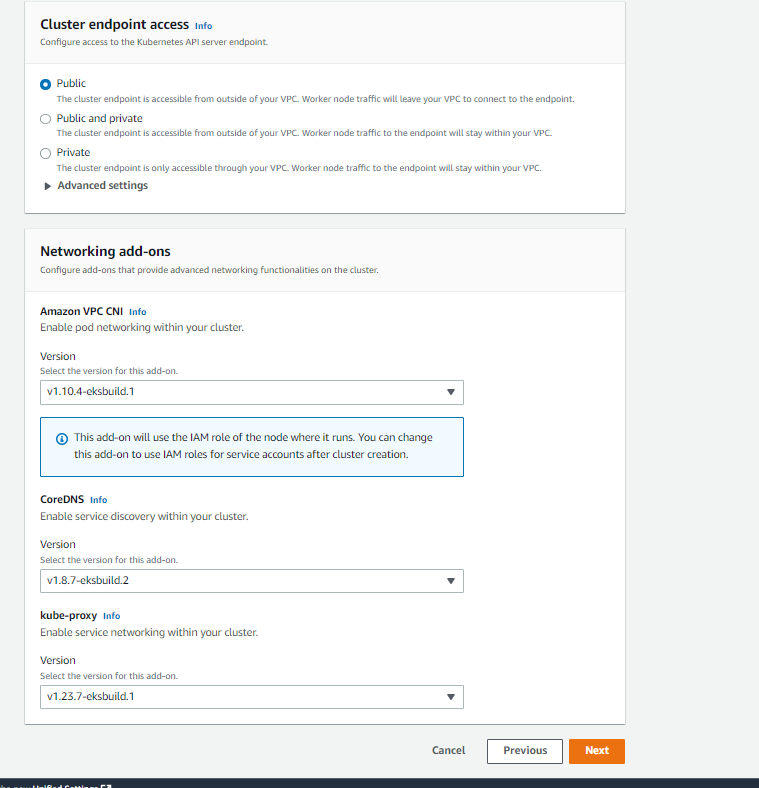
* We need to create a IAM Role for giving EKS necessary permission

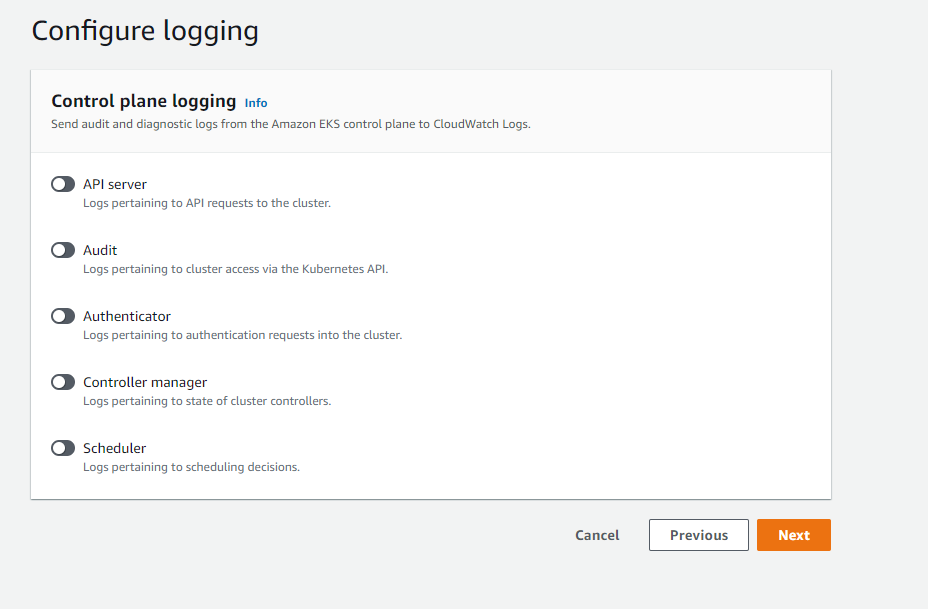




Creating VPC for EKS Cluster documentation : [Creating a VPC for your Amazon EKS cluster - Amazon EKS](https://docs.aws.amazon.com/eks/latest/userguide/creating-a-vpc.html#create-vpc).







Then review and Create the cluster.

* We need to Install AWS CLI for create contact with EKS also we need to change config file inside .Kube Folder(it will be available in user) inside our local machine.

After installing and Configuring AWS CLI We need to run command:

aws eks -- region region\_name update-kubeconfig -- name eks-cluster\_name

* This Command will update the Config file inside .Kube Folder

Adding Nodes for the cluster

* Inside the cluster we created goto compute session -> add NodeGroup
* Then Configure our Node Details
* We should create an IAM role For EC2 instance add one permission for EC2 instance : Permission name : Amazon EKSWorkerNode Policy ,EKSCNIpolicy ,AmazonEC2ContanierRegistryreadonly