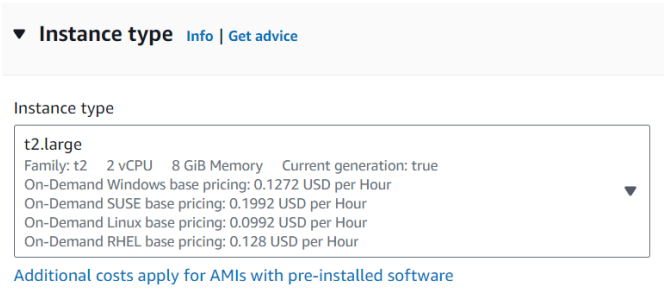
Minikube Installation

Reference website

https://minikube.sigs.k8s.io/docs/start/?arch=%2Flinux%2Fx86-64%2Fstable%2Fbinary+download

Launch Instance with **t3.xlarge** instance type



Install docker with script docker.sh

minikube start --driver=docker

#!/bin/bash

sudo apt-get update

sudo apt-get install ca-certificates curl -y

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

---

Give Executable permission to other & execute

sudo chmod o+x docker.sh

sudo ./docker.sh

```
ubuntu@ip-172-31-34-130:~$ docker info
Client: Docker Engine - Community
 Version:    27.3.1
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.17.1
    Path:     /usr/libexec/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version:  v2.29.7
    Path:     /usr/libexec/docker/cli-plugins/docker-compose
```

To install the latest minikube **stable** release on **x86-64 Linux** using **binary download**:

$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

```
ubuntu@ip-172-31-34-130:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 99.0M  100 99.0M    0     0   9.7M      0  0:00:10  0:00:10 --:--:-- 12.0M
```

$ sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64

$ minikube start

Note – it showing driver error so enter given command

```
ubuntu@ip-172-31-34-130:~$ minikube start
* minikube v1.34.0 on Ubuntu 24.04 (xen/amd64)
* Unable to pick a default driver. Here is what was considered, in preference order:
  - docker: Not healthy: "docker version --format {{.Server.Os}}-{{.Server.Version}}:{{.Server.Platform.Name}}" exit status 1: p
e trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.47/ver
/run/docker.sock: connect: permission denied
  - docker: Suggestion: Add your user to the 'docker' group: 'sudo usermod -aG docker $USER && newgrp docker' <https://docs.dock
/linux-postinstall/>
* Alternatively you could install one of these drivers:
  - kvm2: Not installed: exec: "virsh": executable file not found in $PATH
  - podman: Not installed: exec: "podman": executable file not found in $PATH
  - qemu2: Not installed: exec: "qemu-system-x86_64": executable file not found in $PATH
  - virtualbox: Not installed: unable to find VBoxManage in $PATH

X Exiting due to DRV_NOT_HEALTHY: Found driver(s) but none were healthy. See above for suggestions how to fix installed drivers.
```

$ sudo usermod -aG docker $USER && newgrp docker

$ minikube start

Finally check minikube status-

$ minikube status

```
ubuntu@ip-172-31-34-130:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

Task – Kubernetes using CLI

Step 1 - Install kubectl curl -LO h ps://storage.googleapis.com/kubernetes

release/release/${KUBECTL_VERSION}/bin/linux/amd64/kubectl

Step 2 - Make kubectl executable and move to /usr/local/bin chmod

+x ./kubectl sudo mv ./kubectl /usr/local/bin/kubectl

Step 3 - Verify kubectl version kubectl

version –client

Step 4 - Install eksctl

curl --silent --loca on "h ps://github.com/weaveworks/eksctl/releases/latest/download/eksctl_Linux_amd64.tar.gz" | tar xz -C /tmp

Step 5 - Move eksctl to /usr/local/bin and make it executable sudo mv

/tmp/eksctl /usr/local/bin sudo chmod +x /usr/local/bin/eksctl

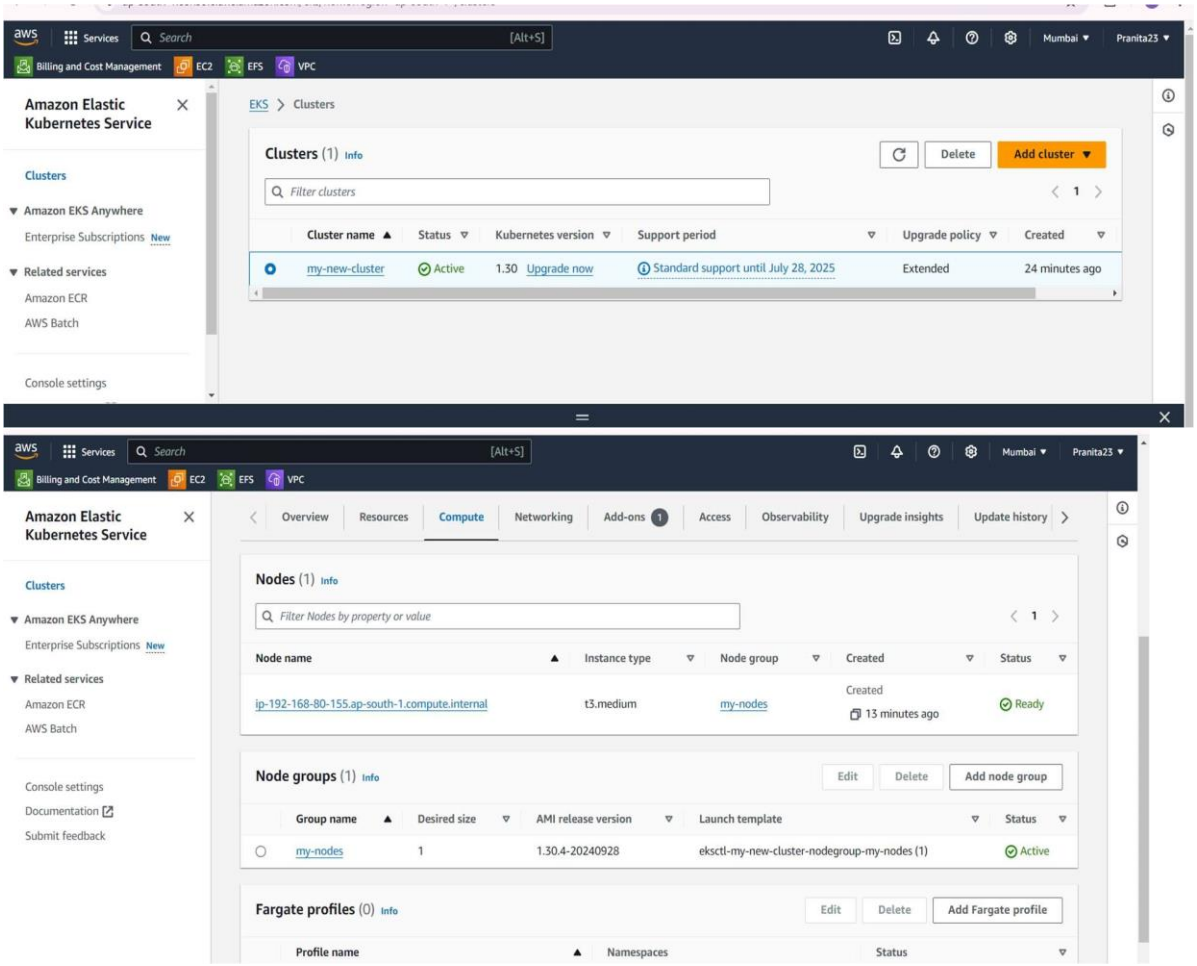
Step 6 - Create a Kubernetes cluster with eksctl

eksctl create cluster --name my-new-cluster --region ap-south-1 --nodegroup-name my-nodes -node-type t3.medium --nodes 1 --nodes-min 1 --nodes-max 1 –managed

Step 7 - Delete the node group and cluster aws eks delete-nodegroup --cluster-name my-new-cluster --

nodegroup-name my-nodes --region ap-south-1

aws eks delete-cluster --name my-new-cluster --region ap-south-1



Kubernetes objects :

- **Pod**: The smallest deployable unit in Kubernetes, representing a single instance of a running process in a cluster.

- **Namespace**: A way to divide cluster resources between multiple users or applications within the same cluster.

- **ReplicationController**: Ensures that a specified number of pod replicas are running at any given time.

- **ReplicaSet**: A newer version of ReplicationController that maintains a stable set of replica pods.

- **Deployment**: Manages updates to applications by creating, updating, and scaling ReplicaSets.

- **Service**: Exposes a set of pods as a network service, allowing stable networking for the pods.

- **ConfigMap**: Stores configuration data in key-value pairs, which can be consumed by pods.
- **Secret**: Stores sensitive information, like passwords, securely and can be used in pods.
- **PersistentVolume (PV)**: Represents storage resources in the cluster that can be used by pods.
- **PersistentVolumeClaim (PVC)**: A request for storage by a user, linked to a PersistentVolume.
- **Ingress**: Manages external access to services, usually HTTP/HTTPS routing.

Deployemnet strategies :

1. **Recreate**:
    - **How it works**: Stops all old pods before starting new ones.
    - **Use case**: Suitable for non-critical applications where downtime is acceptable.

2. **Rolling Update**:
    - **How it works**: Updates pods gradually, replacing old ones with new ones one at a time.
    - **Use case**: Default strategy for Kubernetes deployments; minimizes downtime while ensuring smooth transitions.

3. **Blue-Green Deployment**:
    - **How it works**: A new environment (green) is created alongside the existing one (blue), and traffic is switched to the new version once verified.
    - **Use case**: Reduces the risk of issues, as rollback to the old version is easy.

4. **Canary Deployment**:
    - **How it works**: New updates are rolled out to a small subset of users before a full deployment.
    - **Use case**: Ideal for testing changes in production with minimal risk.

5. **A/B Testing**:

- **How it works**: Different versions of the application run simultaneously, and specific traffic is routed to each version for comparison.
- **Use case**: Used for experiments and comparison to see which version performs better.

6. **Shadow Deployment** is a deployment strategy where the new version of an application runs alongside the current production version but doesn't serve live user traffic.