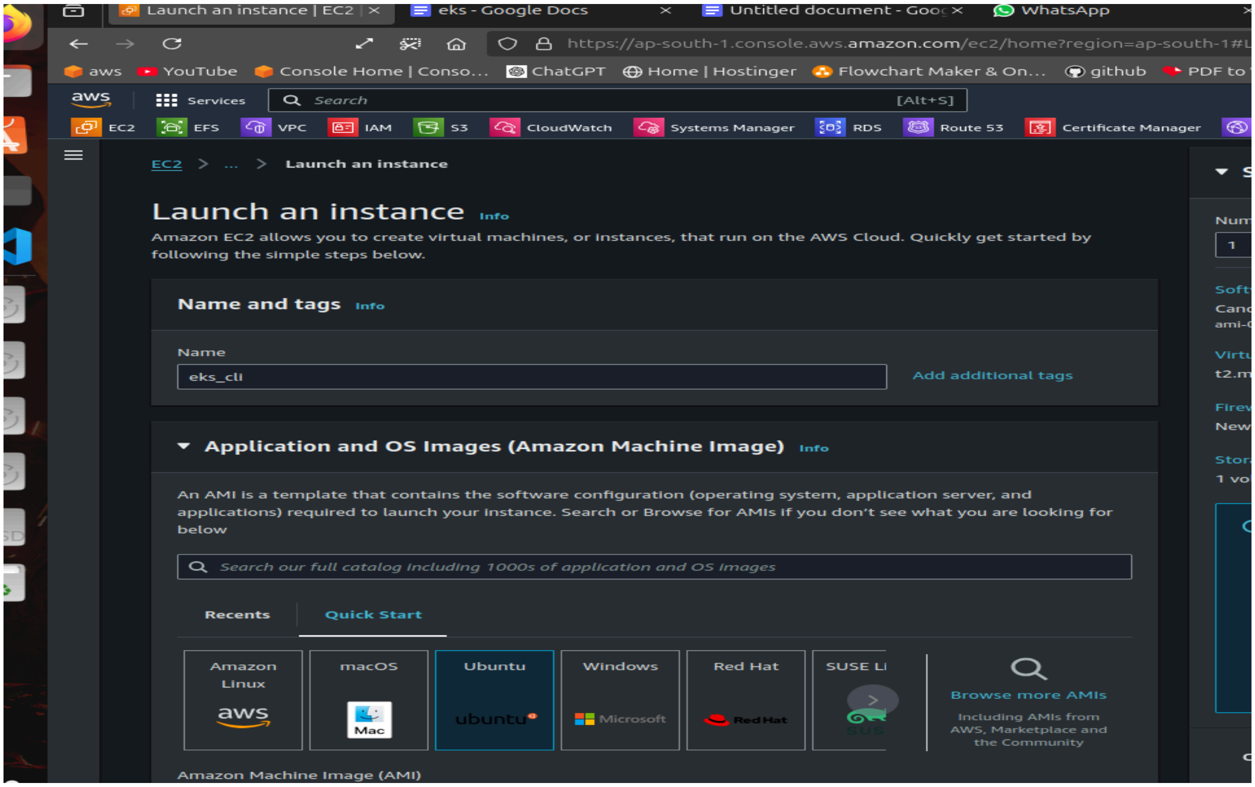
Task : Install and Setup EKS CLI for managing Kubernetes clusters on AWS.

Step 1:Create an ec2 instance and Launch it.



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

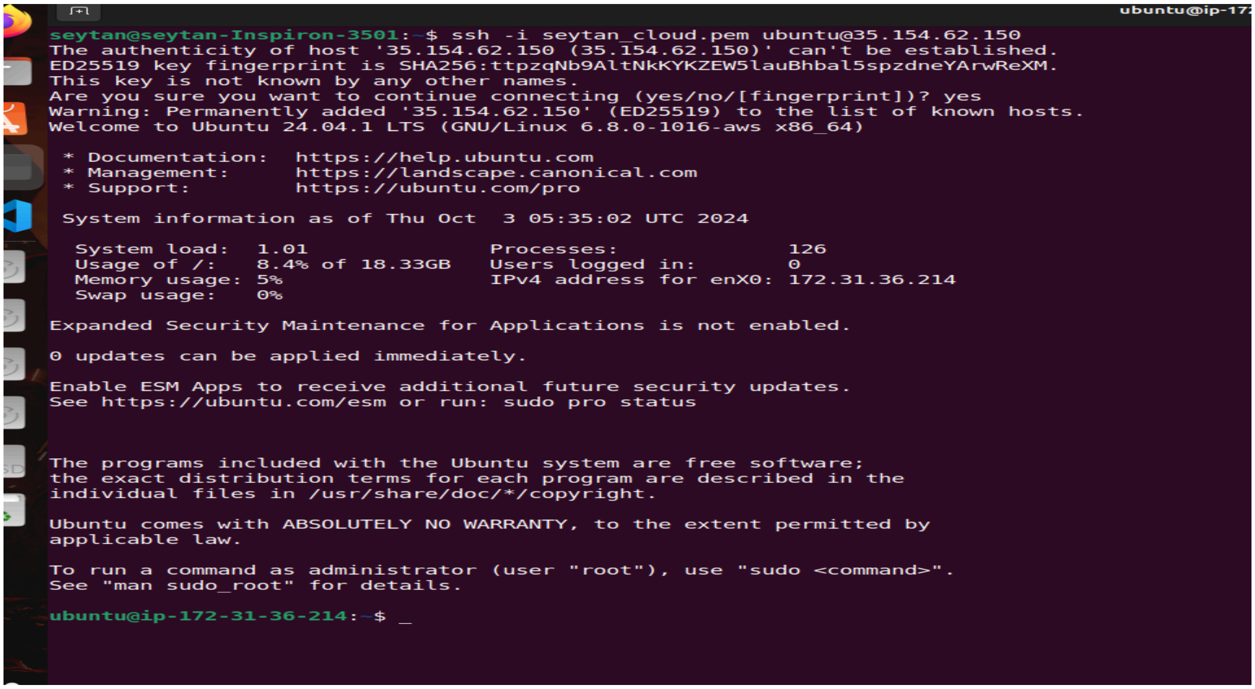
Name: eks_cli

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Li
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)

Step 2: Take ssh of the instance.



```
seytan@seytan-Inspiron-3501:~$ ssh -i seytan_cloud.pem ubuntu@35.154.62.150
The authenticity of host '35.154.62.150 (35.154.62.150)' can't be established.
ED25519 key fingerprint is SHA256:ttpzqNb9AltNkKYKZEW5lauBhbal5spzdneYArwReXM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '35.154.62.150' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Oct  3 05:35:02 UTC 2024

  System load:  1.01               Processes:             126
  Usage of /:   8.4% of 18.33GB    Users logged in:       0
  Memory usage: 5%                 IPv4 address for enX0: 172.31.36.214
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-36-214:~$ _
```

Step 3:Run the following commands to install the kubectl on our system.

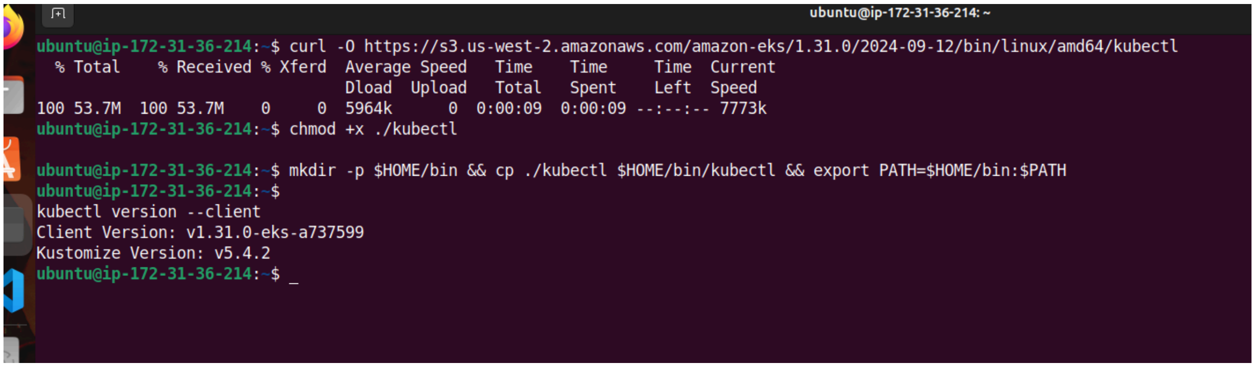
Commands :

curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.0/2024-09-12/bin/linux/amd64/kubectl

chmod +x ./kubectl

mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH

kubectl version --client



```
ubuntu@ip-172-31-36-214:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.0/2024-09-12/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 53.7M  100 53.7M    0     0  5964k      0  0:00:09  0:00:09 --:--:-- 7773k
ubuntu@ip-172-31-36-214:~$ chmod +x ./kubectl

ubuntu@ip-172-31-36-214:~$ mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
ubuntu@ip-172-31-36-214:~$
kubectl version --client
Client Version: v1.31.0-eks-a737599
Kustomize Version: v5.4.2
ubuntu@ip-172-31-36-214:~$ _
```

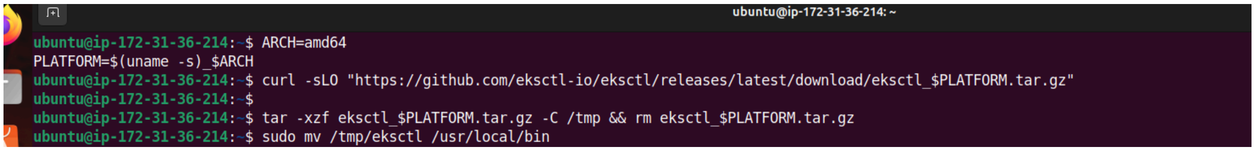
# for ARM systems, set ARCH to: `arm64`, `armv6` or `armv7`
ARCH=amd64
PLATFORM=$(uname -s)_$ARCH

curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_$PLATFORM.tar.gz"

# (Optional) Verify checksum
curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_checksums.txt" | grep $PLATFORM | sha256sum --check

tar -xzf eksctl_$PLATFORM.tar.gz -C /tmp && rm eksctl_$PLATFORM.tar.gz

sudo mv /tmp/eksctl /usr/local/bin



```
ubuntu@ip-172-31-36-214:~$ ARCH=amd64
PLATFORM=$(uname -s)_$ARCH
ubuntu@ip-172-31-36-214:~$ curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_$PLATFORM.tar.gz"
ubuntu@ip-172-31-36-214:~$
ubuntu@ip-172-31-36-214:~$ tar -xzf eksctl_$PLATFORM.tar.gz -C /tmp && rm eksctl_$PLATFORM.tar.gz
ubuntu@ip-172-31-36-214:~$ sudo mv /tmp/eksctl /usr/local/bin
```

What is eks cli ?

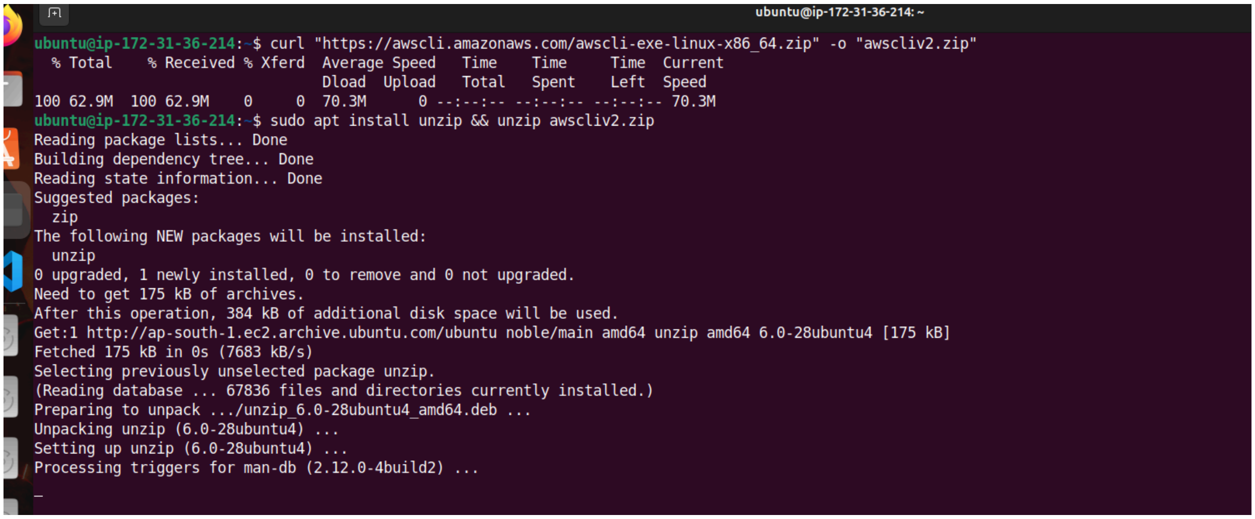
→ The **Amazon EKS CLI (`eksctl`)** is a simple command-line tool used to create and manage Amazon EKS clusters. It simplifies the process of creating Kubernetes clusters on AWS, managing worker nodes, and handling cluster updates.

What is kubectl ?

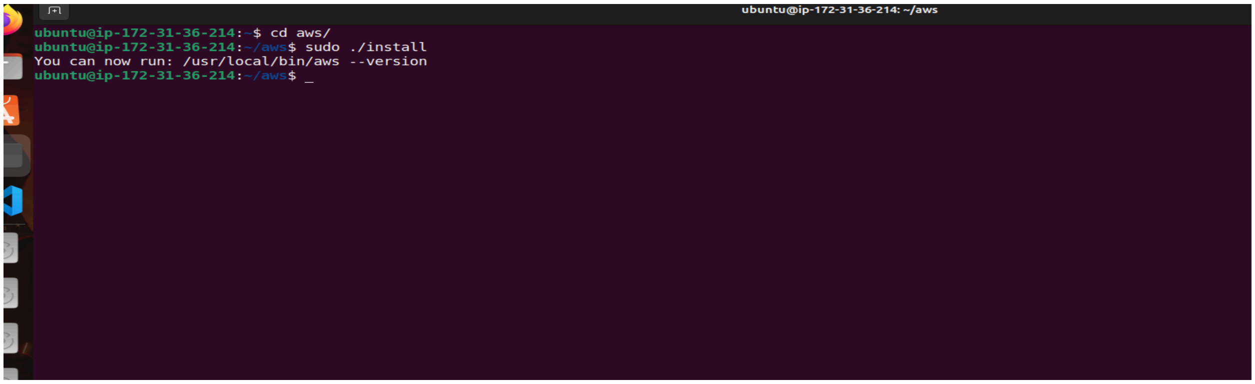
→ **kubectl** is the command-line tool used to interact with Kubernetes clusters. It allows you to manage and deploy applications, inspect resources, and troubleshoot issues within the cluster.

kubectl = Managing Kubernetes resources and workloads.
eksctl = Creating and managing EKS clusters/nodes.

Step 4: Install the awscli and setup it.

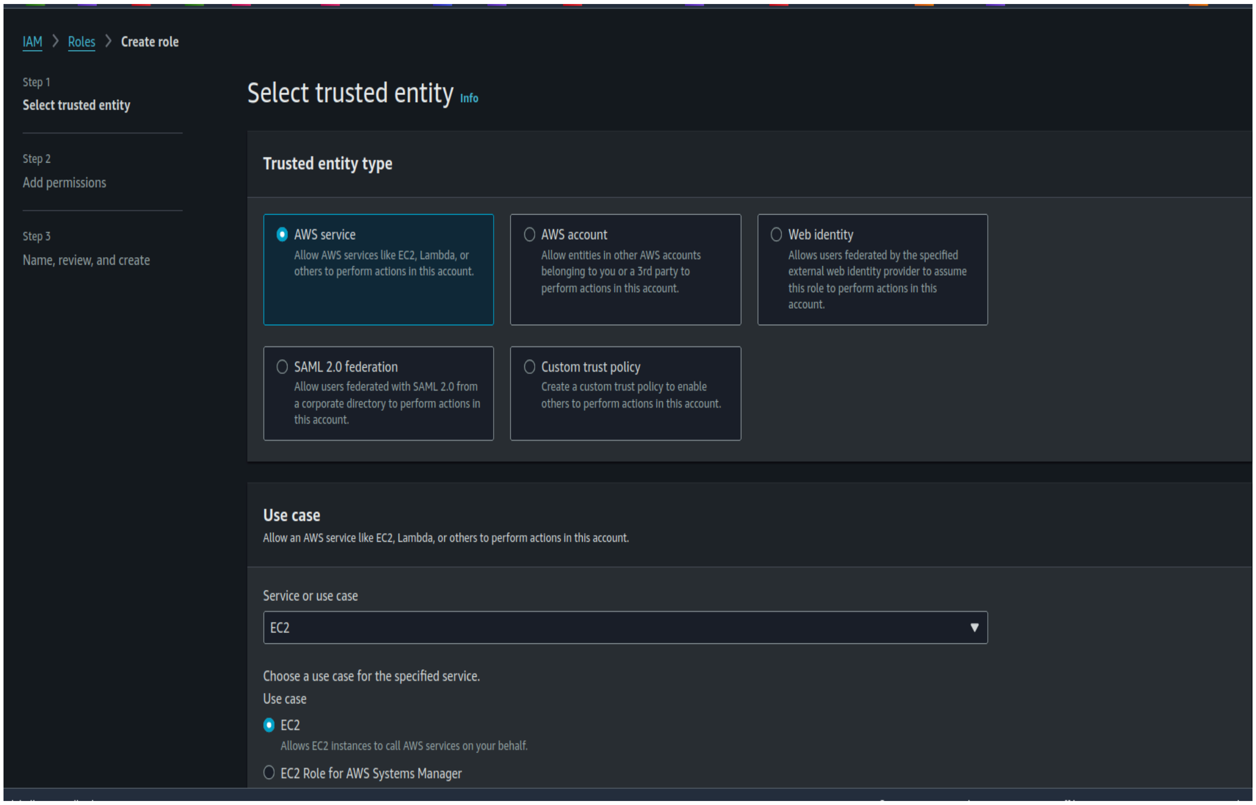


```
ubuntu@ip-172-31-36-214:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 62.9M  100 62.9M    0     0  70.3M      0 --:--:-- --:--:-- --:--:-- 70.3M
ubuntu@ip-172-31-36-214:~$ sudo apt install unzip && unzip awscliv2.zip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  zip
The following NEW packages will be installed:
  unzip
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 175 kB of archives.
After this operation, 384 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 unzip amd64 6.0-28ubuntu4 [175 kB]
Fetched 175 kB in 0s (7683 kB/s)
Selecting previously unselected package unzip.
(Reading database ... 67836 files and directories currently installed.)
Preparing to unpack .../unzip_6.0-28ubuntu4_amd64.deb ...
Unpacking unzip (6.0-28ubuntu4) ...
Setting up unzip (6.0-28ubuntu4) ...
Processing triggers for man-db (2.12.0-4build2) ...
_
```



```
ubuntu@ip-172-31-36-214:~$ cd aws/
ubuntu@ip-172-31-36-214:~/aws$ sudo ./install
You can now run: /usr/local/bin/aws --version
ubuntu@ip-172-31-36-214:~/aws$ _
```

Step 5: Now create an IAM Role for creating and managing the eks cluster from ec2 instance and then attach the role to the instance .



IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Select trusted entity Info

Trusted entity type

● AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

○ AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

○ Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

○ SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

○ Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

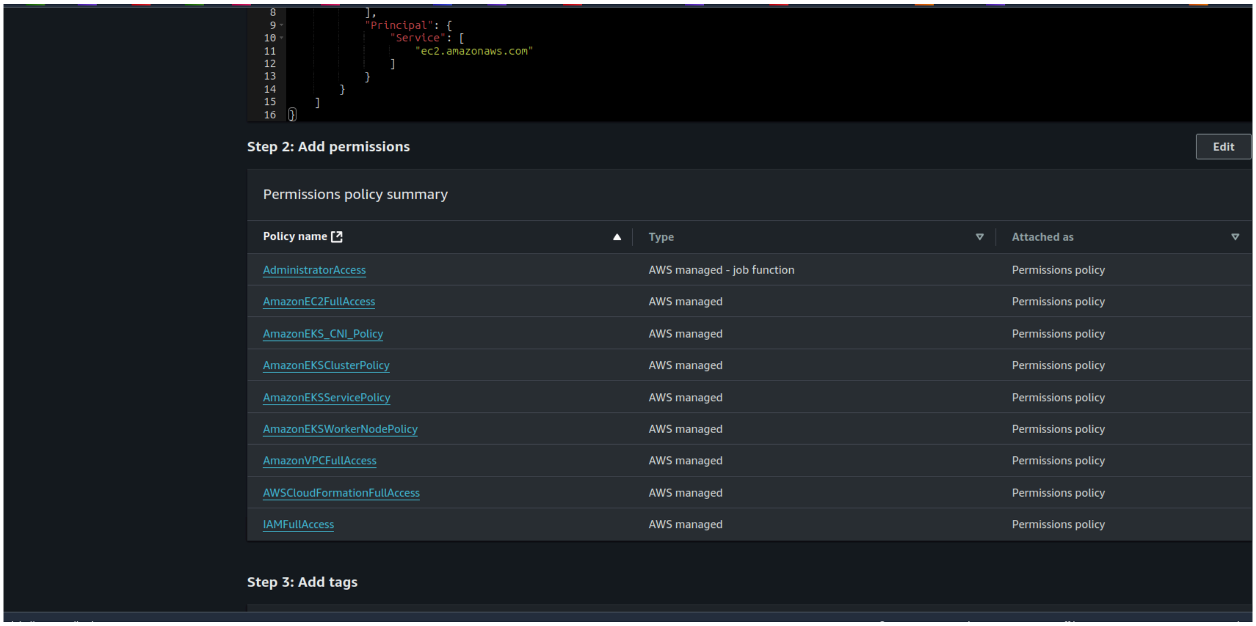
Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
EC2 ▼

Choose a use case for the specified service.
Use case

● EC2
Allows EC2 instances to call AWS services on your behalf.

○ EC2 Role for AWS Systems Manager



```
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

Step 2: Add permissions

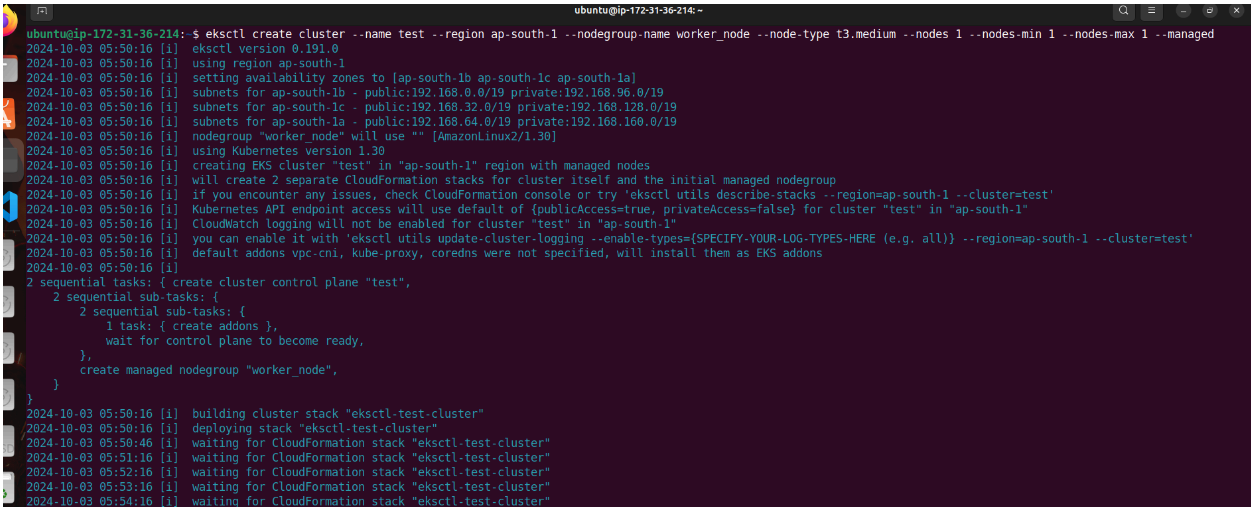
Permissions policy summary

Policy name	Type	Attached as
AdministratorAccess	AWS managed - job function	Permissions policy
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonEKS_CNI_Policy	AWS managed	Permissions policy
AmazonEKSClusterPolicy	AWS managed	Permissions policy
AmazonEKSServicePolicy	AWS managed	Permissions policy
AmazonEKSWorkerNodePolicy	AWS managed	Permissions policy
AmazonVPCFullAccess	AWS managed	Permissions policy
AWSCloudFormationFullAccess	AWS managed	Permissions policy
IAMFullAccess	AWS managed	Permissions policy

Step 3: Add tags

Step 6: Now Create a cluster using cli command .

Command : eksctl create cluster --name my-cluster --region <region> --nodegroup-name <worker-node-name> --node-type <instance-type> --nodes <number-of-nodes> --nodes-min <min-nodes> --nodes-max <max-nodes> --managed



```
ubuntu@ip-172-31-36-214:~$ eksctl create cluster --name test --region ap-south-1 --nodegroup-name worker_node --node-type t3.medium --nodes 1 --nodes-min 1 --nodes-max 1 --managed
2024-10-03 05:50:16 [i]  eksctl version 0.191.0
2024-10-03 05:50:16 [i]  using region ap-south-1
2024-10-03 05:50:16 [i]  setting availability zones to [ap-south-1b ap-south-1c ap-south-1a]
2024-10-03 05:50:16 [i]  subnets for ap-south-1b - public:192.168.0.0/19 private:192.168.96.0/19
2024-10-03 05:50:16 [i]  subnets for ap-south-1c - public:192.168.32.0/19 private:192.168.128.0/19
2024-10-03 05:50:16 [i]  subnets for ap-south-1a - public:192.168.64.0/19 private:192.168.160.0/19
2024-10-03 05:50:16 [i]  nodegroup "worker_node" will use "" [AmazonLinux2/1.30]
2024-10-03 05:50:16 [i]  using Kubernetes version 1.30
2024-10-03 05:50:16 [i]  creating EKS cluster "test" in "ap-south-1" region with managed nodes
2024-10-03 05:50:16 [i]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2024-10-03 05:50:16 [i]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=test'
2024-10-03 05:50:16 [i]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "test" in "ap-south-1"
2024-10-03 05:50:16 [i]  CloudWatch logging will not be enabled for cluster "test" in "ap-south-1"
2024-10-03 05:50:16 [i]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-south-1 --cluster=test'
2024-10-03 05:50:16 [i]  default addons vpc-cni, kube-proxy, coredns were not specified, will install them as EKS addons
2024-10-03 05:50:16 [i]
2 sequential tasks: { create cluster control plane "test",
    2 sequential sub-tasks: {
        2 sequential sub-tasks: {
            1 task: { create addons },
            wait for control plane to become ready,
        },
        create managed nodegroup "worker_node",
    }
}
2024-10-03 05:50:16 [i]  building cluster stack "eksctl-test-cluster"
2024-10-03 05:50:16 [i]  deploying stack "eksctl-test-cluster"
2024-10-03 05:50:46 [i]  waiting for CloudFormation stack "eksctl-test-cluster"
2024-10-03 05:51:16 [i]  waiting for CloudFormation stack "eksctl-test-cluster"
2024-10-03 05:52:16 [i]  waiting for CloudFormation stack "eksctl-test-cluster"
2024-10-03 05:53:16 [i]  waiting for CloudFormation stack "eksctl-test-cluster"
2024-10-03 05:54:16 [i]  waiting for CloudFormation stack "eksctl-test-cluster"
```

Types of Object in kubernets :

Kubernetes Objects Explained

1. Pods
Dependency: None directly. Pods are the fundamental units in Kubernetes.
Use: Pods run containerized applications. They can hold multiple containers that share resources
like storage and network.
Example: A Pod might contain a web server container and a logging container that share the same
network and storage.

2. Nodes
Dependency: Kubernetes cluster.
Use: Nodes are the worker machines that run Pods. They can be physical or virtual, and multiple
nodes make up a Kubernetes cluster.
Example: A Kubernetes cluster might have three virtual machines (nodes) that each run multiple
Pods.

3. Service

Dependency: Pods.

Use: A Service allows Pods to communicate within or outside the cluster.

- **ClusterIP: Internal communication within the cluster.**
- **NodePort: Exposes the application to external traffic through a static port on each node.**
- **LoadBalancer: Exposes the application externally via a cloud provider's load balancer.**

Example: A Service might expose a web application running in Pods to the internet using a LoadBalancer.

4. Namespace

Dependency: None.

Use: It organizes and isolates Kubernetes resources, like Pods and Services, within the same cluster for multi-tenant environments.

Example: Different teams can use separate namespaces to avoid resource conflicts in the same cluster.

5. ReplicaSet

Dependency: Pods.

Use: Ensures a specified number of pod replicas are running at all times. It automatically adds or removes Pods to maintain the desired state.

Example: A ReplicaSet might ensure that three instances of a web server Pod are always running.

6. Deployment

Dependency: ReplicaSet and Pods.

Use: Manages the lifecycle of ReplicaSets. It enables rolling updates, scaling, and rollback functionality for applications.

Example: A Deployment might handle updates to a web application by creating new Pods and phasing out the old ones without downtime.

7. DaemonSet

Dependency: Nodes and Pods.

Use: Ensures that a copy of a Pod is running on all (or some specific) Nodes. It is often used for logging, monitoring, or other system services.

Example: A DaemonSet might ensure that a monitoring agent runs on every node in the cluster.

8. StatefulSet

Dependency: Pods and Persistent Volumes (PV).

Use: Manages stateful applications that require stable, persistent storage, ensuring each Pod gets a unique identity and storage.

Example: A StatefulSet might manage a database cluster where each database instance requires its own persistent storage.

9. ConfigMap

Dependency: Pods.

Use: Provides externalized configuration data to Pods in key-value pairs, allowing apps to be configured without hardcoding values.

Example: A ConfigMap might provide configuration settings for a web application, like database connection strings.

10. Secrets

Dependency: Pods.

Use: Stores sensitive information such as passwords or API keys in an encrypted format, which is consumed by Pods securely.

Example: A Secret might store API keys for accessing a third-party service, which are then used by application Pods.

11. Persistent Volume (PV) and Persistent Volume Claim (PVC)
Dependency: Pods and StatefulSets.
Use:
- PV: Provides long-term storage for Pods.
- PVC: A request for storage made by Pods or StatefulSets.
Example: A PV might be a cloud disk, and a PVC would be a claim to use that disk by a database
Pod.

12. RBAC (Role-Based Access Control)
Dependency: Kubernetes resources (e.g., Pods, Services, etc.).
Use: Manages permissions by assigning roles to users and services, controlling access to
Kubernetes resources.
Example: An admin might use RBAC to give read-only access to certain users for specific
namespaces.

13. Ingress Gateway
Dependency: Services and Pods.
Use: Manages external HTTP/HTTPS traffic, routing it to specific Services, which then send traffic to
Pods.
Example: An Ingress Gateway might route incoming web traffic to different Services based on the
URL path.

14. ReplicationController
Dependency: Pods.
Use: Ensures a specified number of Pod replicas are running, similar to a ReplicaSet. It's an older mechanism, with ReplicaSet being more commonly used today.
Example: A ReplicationController might ensure that three instances of an application are running,
but it is generally replaced by ReplicaSet for new deployments.