Installing HPCC Systems on AWS Platform

- By Skanda P R

**Steps to install**:

1. Create an AWS account.
2. Create the IAM user.
3. Setting up the prerequisites.
4. Create Virtual Private Cloud (VPC).
5. Create Subnets within the VPC.
6. Deploy an EKS cluster in the VPC.
7. Install EFS CSI Driver.
8. Authorizing inbound access to the security group.
9. Deploy the HPCC Systems cluster on EKS.

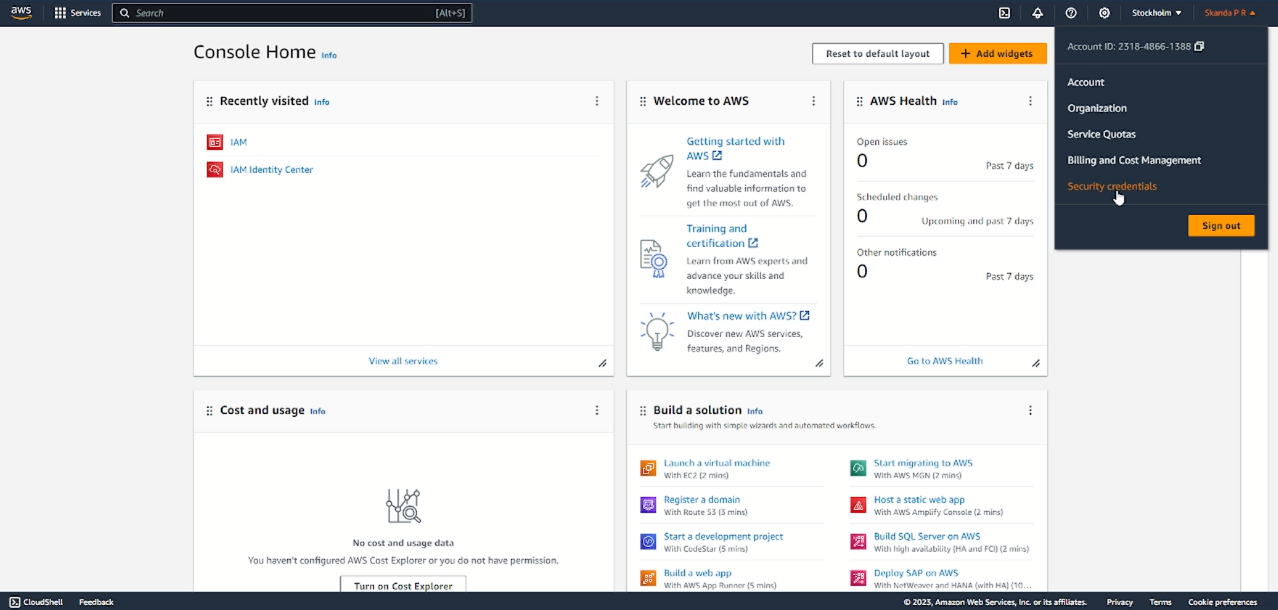
**Create AWS Account**

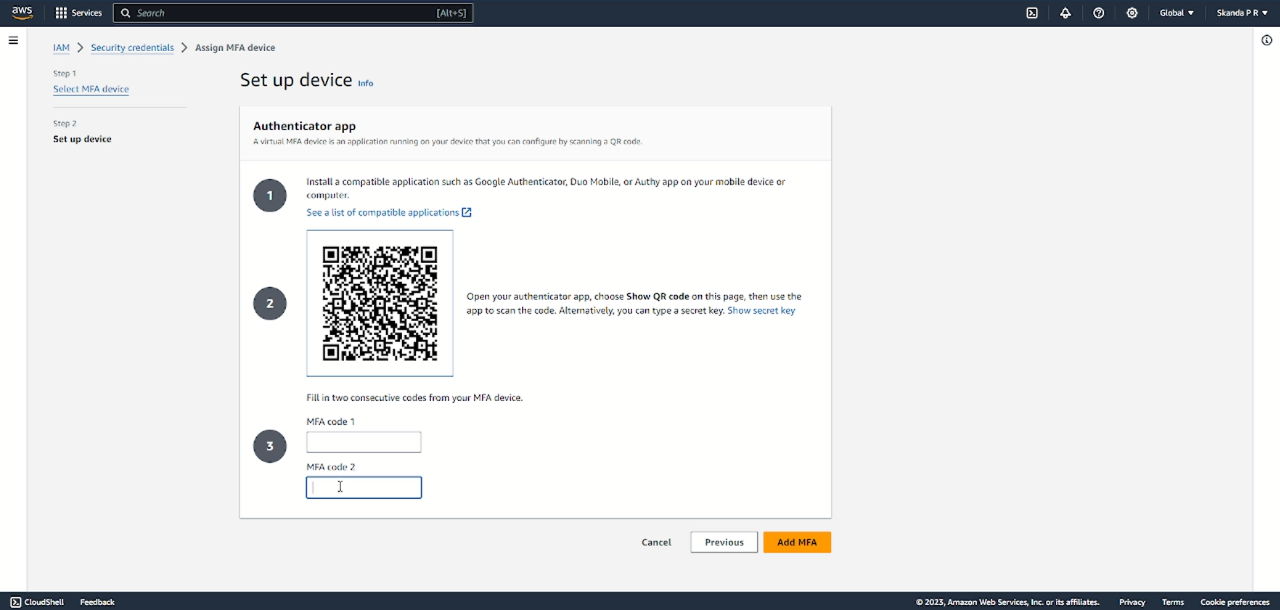
Head over to <https://aws.amazon.com/console/> to create an AWS Account.

**Create the IAM user and add policies**

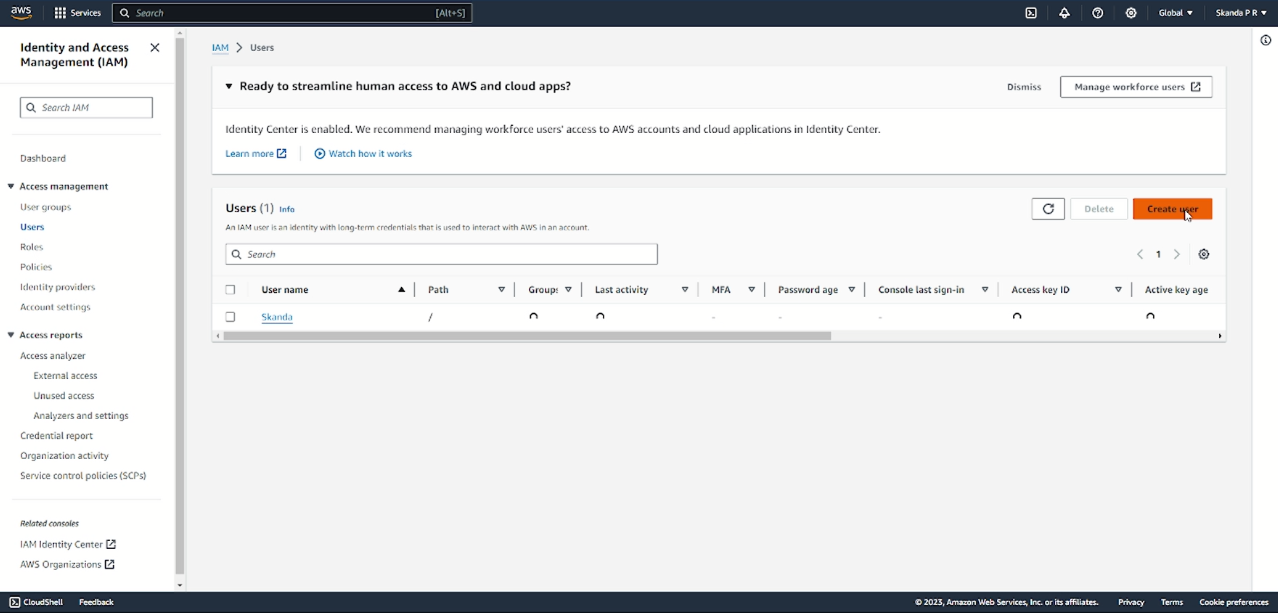
Identity and Access Management (IAM), is a service provided by AWS, which allows us to manage access to certain AWS services and resources securely. This should be added to the user who manages the cluster. It is given to the user in the AWS Console page.

In the Console page, select your account and open “Security Credentials”, which will take you to Identity and Access Management Page. Here you enable the Multi Factor Authentication by using an application like Google Authenticator.



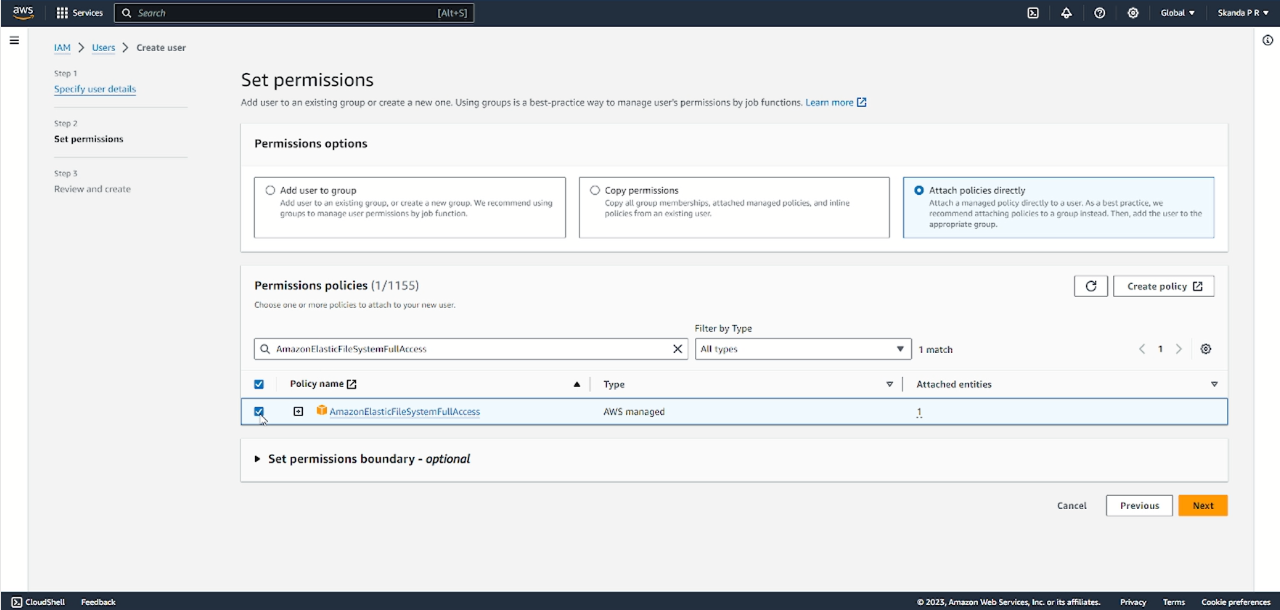


Next, create an IAM user and make a note of the ACCESS\_KEY and the SECRET\_KEY.



While you are creating the user, add the following policies to the user:

* AmazonElasticFileSystemFullAccess
* AWSCloudFormationFullAccess
* AmazonEC2FullAccess
* IAMFullAccess
* AmazonEKSClusterPolicy
* AmazonEKSWorkerNodePolicy
* AmazonS3FullAccess
* CloudFrontFullAccess
* AmazonVPCFullAccess
* AmazonEKSServicePolicy



After successfully creating the user, click on the username which you created and click on “Add Policies” and select “Create Inline Policy”. Now in the “Policy Editor” select “JSON” and paste this text:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"eks:\*"

],

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "iam:PassRole",

"Resource": "\*",

"Condition": {

"StringEquals": {

"iam:PassedToService": "eks.amazonaws.com"

}

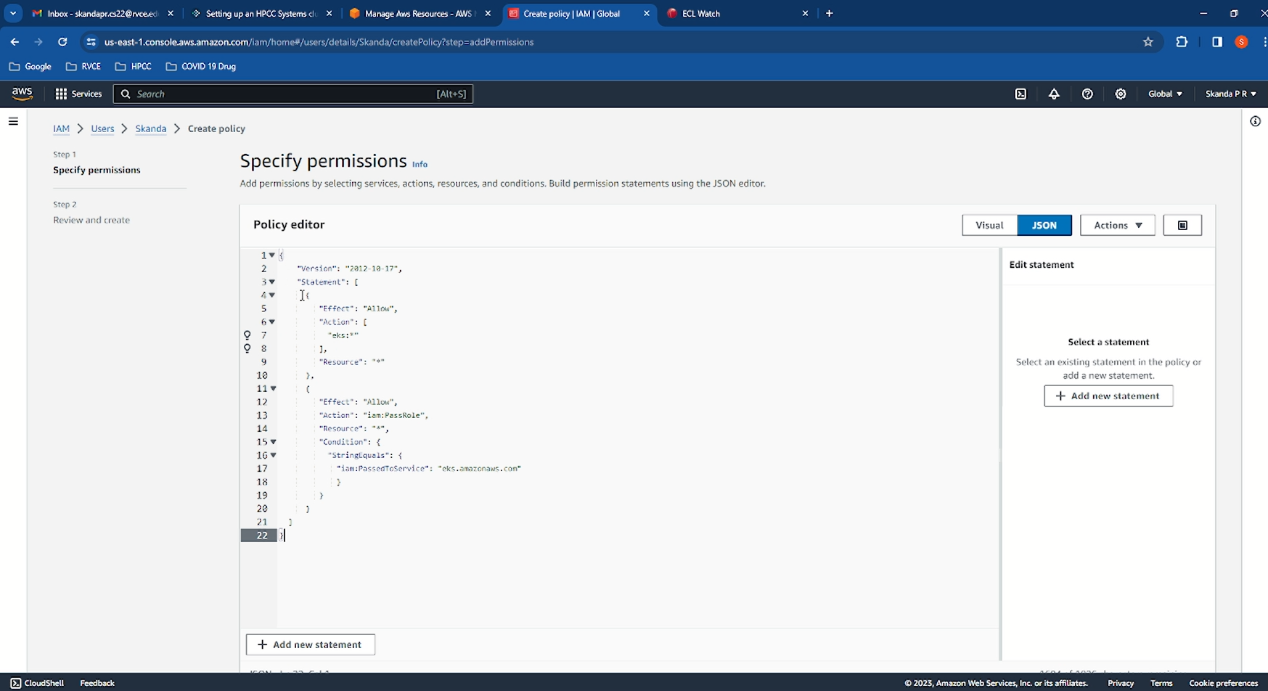
}

}

]

}

Give any name for the policy and click on “Create Policy”.



**Setting up the prerequisites**

These tools are prerequisites to install the HPCC System, hence install them by clicking on the provided links and make sure to add all the files related to the tools to the Environment PATH variables.

* AWS CLI v2: <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html>
* Kubectl: <https://kubernetes.io/docs/tasks/tools/install-kubectl/>
* EKSCTL: <https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html>
* Helm: <https://helm.sh/docs/intro/install/>

Next open your command prompt and paste the below command and provide the ACCESS KEY and the SECRET ACCESS KEY.:

aws configure

**Deploy a Virtual Private Cloud (VPC) and Subnets**

VPC is a logically isolated section where we can launch AWS resources.

Subnets are subdivisions of a larger network like VPC. Subnetting allows us to divide these larger networks into smaller and manageable segments. A minimum of two subnets must be created within the VPC.

To create a VPC, run the following commands:

aws efs create-file-system --throughput-mode bursting --tags "Key=Name,Value=<EFS NAME>" --region <REGION>

The following command outputs the EFS File System Id, which will be used in further steps.

aws efs describe-file-systems --region <REGION>

aws ec2 describe-vpcs --region <REGION>

To deploy subnets in the VPC, run these following commands:

aws ec2 describe-subnets --region <REGION> --filters "Name=vpc-id,Values=<VPC ID>"

The following creates a mount target with minimum of two subnets.

aws efs create-mount-target --region <REGION> --file-system-id <EFS ID> --subnet-id <Subnet id1>

aws efs create-mount-target --region <REGION> --file-system-id <EFS ID> --subnet-id <Subnet id2>

aws efs describe-mount-targets --region <REGION> --file-system-id <EFS ID>

**Please store all the below information, as they will be required in the subsequent steps**:

* File System ID / EFS ID
* Region
* Subnet ID
* Mount target ID

**Deploy an EKS Cluster**

An Amazon Elastic Kubernetes Service (EKS), which is a managed Kubernetes service on AWS, is setup with the specified subnets, security groups and other configurations. This can be done with the following command:

eksctl create cluster --name <EKS Cluster name> --region <REGION> --nodegroup-name <Node group name> --node-type t3.medium --nodes 3 --nodes-min 1 --nodes-max 4 --managed --vpc-public-subnets <Subnet id1> --vpc-public-subnets <Subnet id2>

For this demonstration, Node type “t3.medium” with an initial 3 nodes and a maximum 4 nodes is deployed.

To confirm the deployment, we can run the below command:

eksctl get clusters

**Install AWS EFS CSI driver**

When we create an EKS Cluster, we can add the Elastic File System (EFS) Container Storage Interface (CSI) Driver, which provisions the Amazon EFS storage volumes, as a Persistent Volumes, for the use by the EKS cluster.

We need to follow these steps to install the EFS CSI driver:

kubectl apply -k "github.com/kubernetes-sigs/aws-efs-csi-driver/deploy/kubernetes/overlays/stable/ecr/?ref=release-1.3"

Next run this command and wait till all the services' status shows as 'Running', NOT as 'ContainerCreating':

kubectl get pods -n kube-system

**Authorizing inbound access to the security group for the EFS mount target**

When we run the below command, we get the security group ID of the EKS cluster, make a note of it:

aws eks describe-cluster --name <Cluster name> --query cluster.resourcesVpcConfig.clusterSecurityGroupId

Next run this to get Security group ID of the mount target created earlier:

aws efs describe-mount-target-security-groups --mount-target-id <MOUNT TARGET ID>

Now authorize inbound access to the security group for the EFS mount target, using the below command:

aws ec2 authorize-security-group-ingress --group-id <Security group ID of Mount Target> --protocol tcp --port 2049 --source-group <Security group ID of the EKS cluster> --region <Region>

Now, we have to deploy EFS Provisioner:

git clone <https://github.com/kubernetes-incubator/external-storage>

cd external-storage/aws/efs/deploy/

The below command is used to set up Role-Based Access Control (RBAC) configurations for the AWS Elastic File System as part of the external storage for Kubernetes:

kubectl apply -f rbac.yaml

Now open manifest.yaml:

nano manifest.yaml

Now, in 7th and 8th line, you will see something like this:

file.system.id: yourEFSsystemid

aws.region: regionyourEFSisin

Here, replace "yourEFSsystemid" with your EFS ID, and "regionyourEFSisin" with the Region where EFS is deployed.

Also in line 56, you will see something like this:

“server: yourEFSsystemID.efs.yourEFSregion.amazonaws.com”

Here, enter your EFS ID and Region in the same format. For the demonstration purpose, I would enter it as follows:

“server: fs-0ebe036d9c20f60b4.efs.eu-north-1.amazonaws.com”

This manifest deploys an External Storage Provisioner for AWS EFS using a ConfigMap for configuration, a Deployment to manage the provisioner pods, a StorageClass to define the storage class for dynamic provisioning, and a PersistentVolumeClaim to request storage from the provisioner.

By running the below command, Kubernetes will read the contents of the manifest.yaml file and attempt to create or update the specified resources based on the definitions provided in the file.

kubectl apply -f manifest.yaml

kubectl get pods

Now, we will see that efs-provisioner is Running.

**Install HPCC Systems on EKS Cluster using Helm Charts**

We need to clone the HPCC Systems Helm Repository and deploy it as follows:

helm repo update

helm repo add hpcc <https://hpcc-systems.github.io/helm-chart/>

helm install <Helm Cluster name> hpcc/hpcc --version=9.4.0

Here, you can specify which version you have to install, so that whenever you are installing plugins or other modules, you will have an idea in which version you have to install. You can skip this version tag also.

When we run the above command, we will see the output as follows:

NAME: <Helm Cluster Name>

LAST DEPLOYED: Tue Nov 28 17:52:09 2023

NAMESPACE: default

STATUS: deployed

REVISION: 1

TEST SUITE: None

This chart has defined the following HPCC components:

dafilesrv.spray-service

dali.mydali

dfuserver.dfuserver

eclagent.hthor

eclagent.roxie-workunit

eclccserver.myeclccserver

eclscheduler.eclscheduler

esp.eclwatch

esp.eclservices

esp.eclqueries

esp.esdl-sandbox

esp.sql2ecl

esp.dfs

roxie.roxie

thor.thor

dali.sasha.coalescer

sasha.dfurecovery-archiver

sasha.dfuwu-archiver

sasha.file-expiry

sasha.wu-archiver

Now instead of installing the default HPCC Systems platform, we can create a custom configuration YAML file and deploy HPCC System platform using the default configuration plus the customizations. It can be done as follows.

**To create custom configuration chart for two Roxies and two Thors**

We can create a new YAML file named "customroxie.yaml", then we can copy the default values from "myvalues.yaml" file as follows:

helm show values hpcc/hpcc > myvalues.yaml

nano myvalues.yaml

Now copy the entire roxie section from this file, the open "customroxie.yaml" file and paste it. In the second block, rename the name and prefix name to "roxie2"  
Now the "customroxie.yaml" file will have content as follows:

roxie:

- name: roxie

disabled: false

prefix: roxie

services:

- name: roxie

servicePort: 9876

listenQueue: 200

numThreads: 30

visibility: local

replicas: 2

numChannels: 2

serverReplicas: 0

localAgent: false

traceLevel: 1

topoServer:

replicas: 1

- name: roxie2

disabled: false

prefix: roxie2

services:

- name: roxie2

servicePort: 9876

listenQueue: 200

numThreads: 30

visibility: local

replicas: 2

numChannels: 2

serverReplicas: 0

localAgent: false

traceLevel: 1

topoServer:

replicas: 1

The same thing can be done for thor as well. Create a new YAML file "customthor.yaml". Copy the thor content block from "myvalues.yaml", paste it in "customthor.yaml", then rename the second thor name and prefix name to "thor2".

Now the "customthor.yaml" will have contents as follows:

thor:

- name: thor

prefix: thor

numWorkers: 2

maxJobs: 4

maxGraphs: 2

- name: thor2

prefix: thor2

numWorkers: 10

maxJobs: 4

maxGraphs: 2

Now to install HPCC Systems platform with the custom configurations: we can run the following command:

helm install mycluster hpcc/hpcc -f customroxie.yaml -f customthor.yaml

To upgrade the cluster (which is already installed) with this custom configuration, we run the below command:

helm upgrade mycluster hpcc/hpcc -f customroxie.yaml -f customthor.yaml

**To get ECL Watch IP Address**

Now, run this command and wait for all the services' status to turn to 'Running', NOT 'ContainerCreating':

kubectl get pods

Now if we run this command, we can see the IP address of ECL Watch:

kubectl get svc

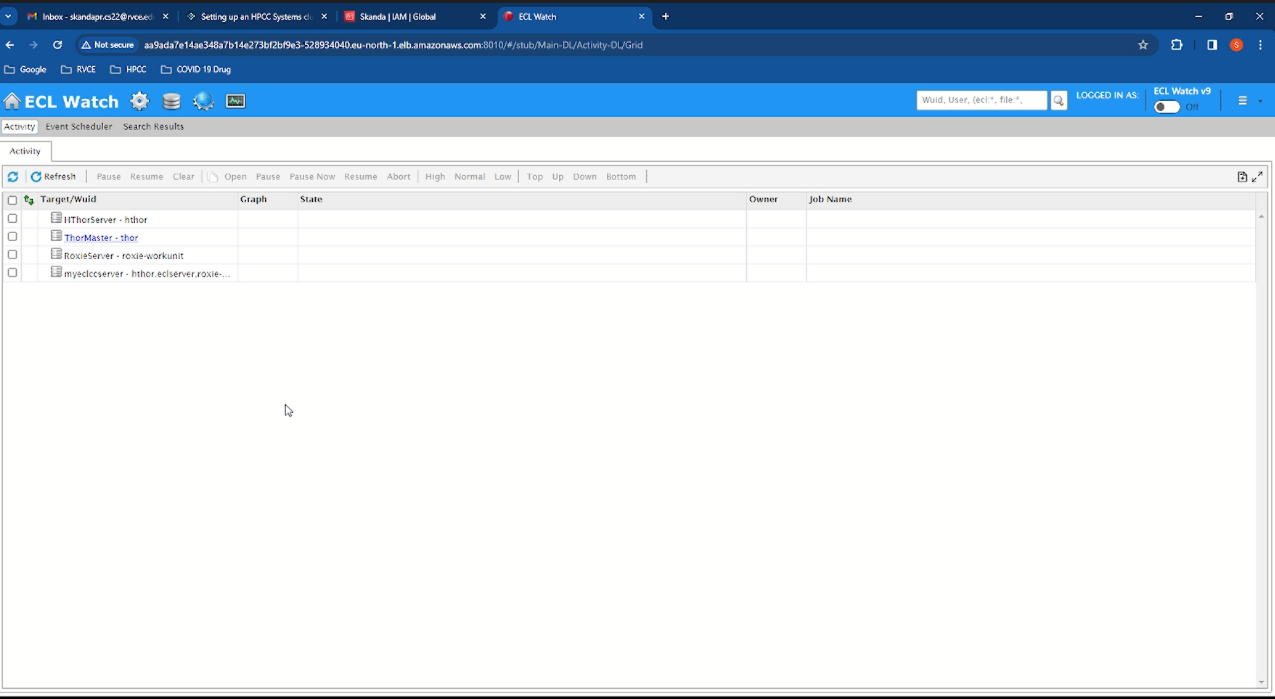
Here copy the External IP Address of ECL Watch which will be similar to this:

aa9ada7e14ae348a7b14e273bf2bf9e3-528934040.eu-north-1.elb.amazonaws.com

Open up a browser and paste this link, along with the port number as follows:

<http://aa9ada7e14ae348a7b14e273bf2bf9e3-528934040.eu-north-1.elb.amazonaws.com:8010>

If everything is working as expected, the ECL Watch page will be displayed as shown:



**Uninstall HPCC Systems and EFS Volumes:**

Follow the below commands to uninstall all the services:

helm uninstall <Helm Cluster name>

kubectl delete pv –all

kubectl delete -k "github.com/kubernetes-sigs/aws-efs-csi-driver/deploy/kubernetes/overlays/stable/ecr/?ref=release-1.3"

eksctl delete cluster <EKS Cluster name>

aws efs delete-mount-target --mount-target-id <mount target ID>

aws efs delete-file-system --file-system-id <EFS ID>

You can also refer to this video for Installation:

<https://youtu.be/tfgxUnJaxVc?si=kHZ3A5blGPx0r3wj>