axcess.io

Pre-requisite Document

Axcess EKS Standard Governance Model Pre-Requisites

TABLE OF CONTENT

TABLE OF CONTENTS

1 DEPLOYMENT PROCESS

Upon completion of all prerequisites for deploying the Axcess EKS Standard Governance Model, it is imperative to ensure that the **IAM role** with the necessary permissions is attached to the EC2 instance during the deployment process via the AMI. Also make sure that EC2 has the **Internet access** to download the required scripts. This IAM role is essential for launching the CloudFormation stack and executing the automation script.

The shell script embedded within the AMI is responsible for initiating the CloudFormation Nested Stacks, which facilitate the deployment of multiple monitoring solutions. These CloudFormation Stacks will automatically create the required IAM roles for the Lambda functions, Lambda functions themselves, CloudWatch Dashboards, CloudWatch Alerts, EventBridge Rules, CloudWatch Log Groups, and Metric Filters.

Upon successful completion of this deployment, the automation script will initiate the EKS cronjob deployment to the EKS clusters specified in the SSM Parameter Store.

Once the EKS deployment is finalized, the EC2 instance will self-terminate as it is no longer required. This approach not only ensures the efficient execution of the deployment but also helps in reducing infrastructure costs.

2 PRE-REQUISITES

2.1 SSM Parameter store –

Two parameters of type String must be created:

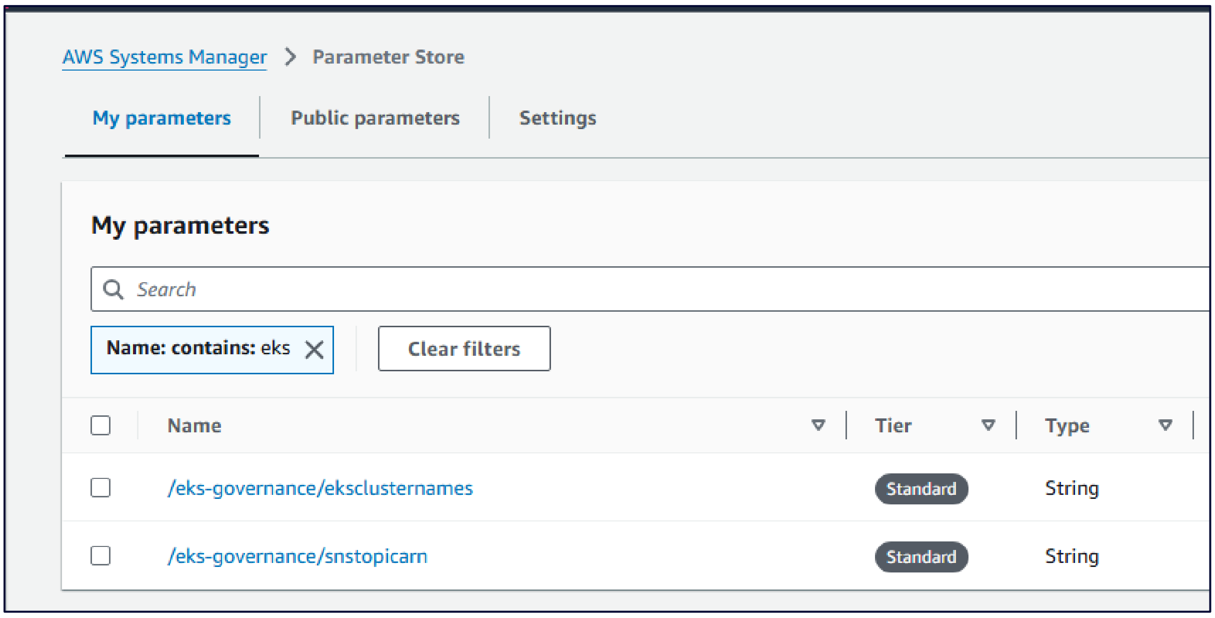
1. **EKS Cluster Name**
2. **SNS Topic ARN**

Refer to the provided image for creating the parameters. When passing values for multiple EKS cluster names, avoid adding any commas between the names. The correct format is: cluster1,cluster2,clusterA.

- Ensure there are no extra blank spaces before or after the names.
- The names of the SSM parameters must be **exactly as specified**, or the automated deployment will fail.

Parameter Names:

- /eks-governance/eksclusternames
- /eks-governance/snstopicarn


AWS Systems Manager > Parameter Store

My parameters | Public parameters | Settings

My parameters

Name: contains: eks | Clear filters

Name	Tier	Type
/eks-governance/eksclusternames	Standard	String
/eks-governance/snstopicarn	Standard	String

2.2 EC2 Instance IAM Role (Deployment Machine)

The IAM Role has to-be created using this CloudFormation Template-
https://eks-governance-marketplace.s3.us-east-1.amazonaws.com/EKS-Governance-EC2-Deploy-Machine-Role.yaml

Download this template and deploy via AWS Console or AWS CLI.

This role has to-be assigned to the EC2 Instance which will be created using the Marketplace AMI, This role gives Instance permissions to deploy the necessary resources in the account.

2.3 EKS OIDC –

For each EKS cluster to be monitored, it is essential to create the OIDC and update the IAM role with the appropriate permissions and trust policy that references the correct service account.

To create the OIDC for the EKS cluster, refer to the official AWS documentation

https://docs.aws.amazon.com/eks/latest/userguide/enable-iam-roles-for-service-accounts.html

Once the IAM OIDC for each EKS cluster is created, proceed to create the IAM role for the service account as the next step

2.4 EKS IRSA (IAM Role For Service Account)–

If you intend to monitor one or multiple EKS clusters, it is necessary to create separate IAM roles for each EKS cluster. You can download the IAM policy and IAM trust policy from the provided URL and manually create the IAM roles. Ensure that the trust policy correctly references the appropriate EKS OIDC ID for each respective EKS cluster.

IRSA IAM Policy JSON - https://eks-governance-marketplace.s3.us-east-1.amazonaws.com/IRSA-Policy.json

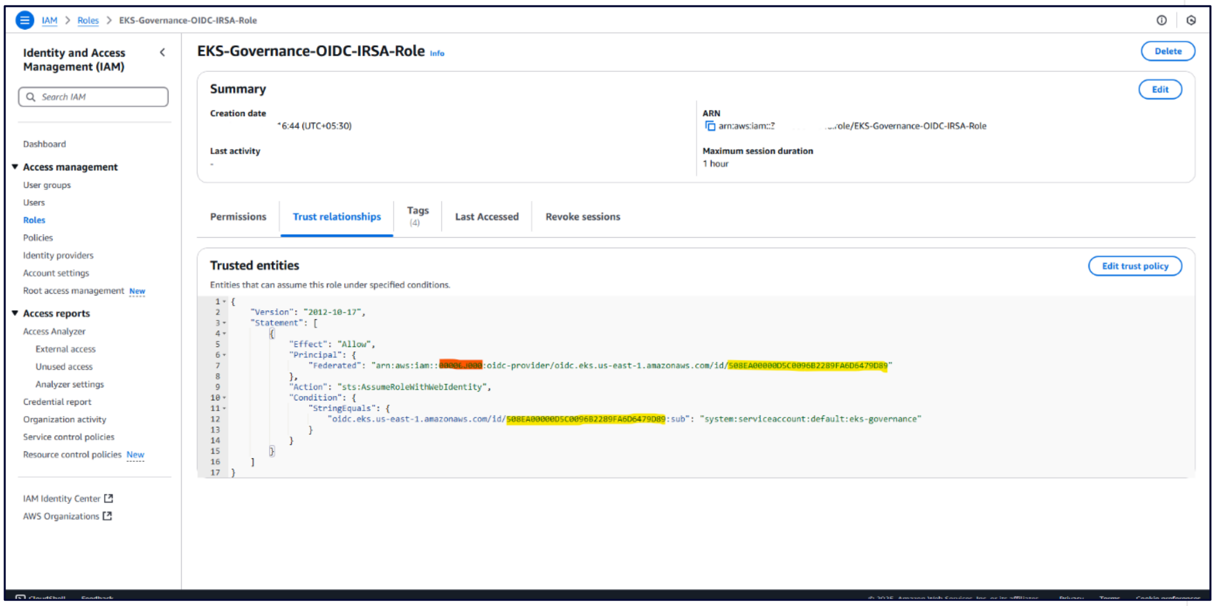
IRSA IAM Trust Relationship Policy JSON - https://eks-governance-marketplace.s3.us-east-1.amazonaws.com/IRSA-trust-policy.json

Once you create the IAM Role, Go to the Role - Trust Relationships tab and modify these entries –

a. AWS Account Number

b. EKS OIDC ID – which can be retrieved from the EKS Console - **OpenID Connect provider URL**

Example –



EKS-Governance-OIDC-IRSA-Role Info

Summary

Creation date
'6:44 (UTC+05:30)

Last activity
-

ARN
arn:aws:iam::2 ...role/EKS-Governance-OIDC-IRSA-Role

Maximum session duration
1 hour

Trusted entities

Entities that can assume this role under specified conditions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::[redacted]:oidc-provider/oidc.eks.us-east-1.amazonaws.com/id/508EA00000D5C0096B2289FA6D6479D89"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-1.amazonaws.com/id/508EA00000D5C0096B2289FA6D6479D89:sub": "system:serviceaccount:default:eks-governance"
                }
            }
        }
    ]
}
```

Note - In the above screenshot highlighted Red is the AWS account ID and Yellow is the EKS OIDC ID

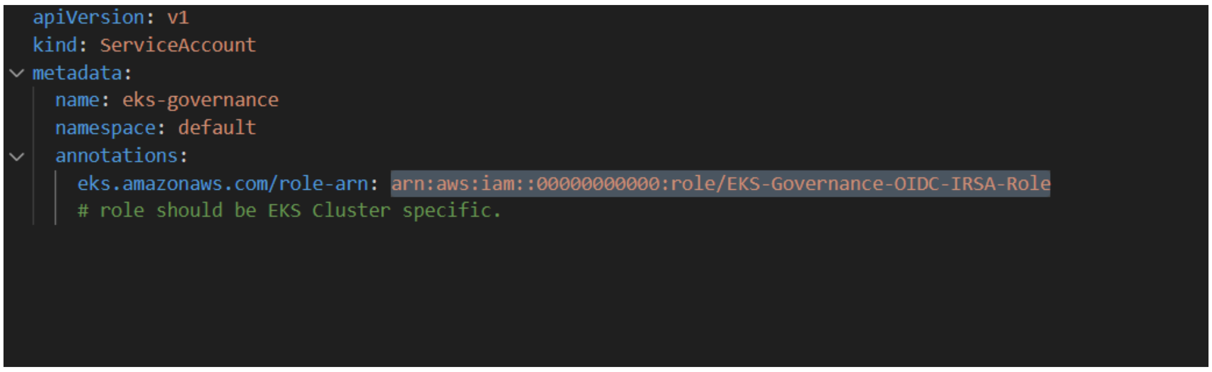
2.5 EKS Deployments –

2.5.1 Service Account

2.5.1 Service Account – Service account has to-be created in each EKS Cluster that has to-be monitored. Use the following URL to download the yaml file.

curl -O https://eks-governance-marketplace.s3.us-east-1.amazonaws.com/Service-account.yaml

then edit the file and change these IAM Role (For Service Account) ARN, save the file and apply it



```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eks-governance
  namespace: default
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::00000000000:role/EKS-Governance-OIDC-IRSA-Role
    # role should be EKS Cluster specific.
```

kubectl apply -f Service-account.yaml

Note - If you are deploying a monitoring solution on multiple EKS clusters and have created a separate IAM role for the service account with the respective EKS cluster's OIDC ID in the trust policy, you must use the corresponding IAM role ARN in the service account annotations.

2.5.2 Cluster Role –

For every EKS Cluster to-be monitored, apply this yaml file using following command

curl -O https://eks-governance-marketplace.s3.us-east-1.amazonaws.com/cluster-role.yaml

kubectl apply -f cluster-role.yaml

2.5.3 Cluster Role Binding –

For every EKS Cluster to-be monitored, apply this yaml file using following command

curl -O https://eks-governance-marketplace.s3.us-east-1.amazonaws.com/cluster-role-binding.yaml

kubectl apply -f cluster-role-binding.yaml

2.5.4 aws-auth ConfigMap –

For each EKS cluster that requires monitoring, it is necessary to grant permissions in the **aws-auth ConfigMap** to enable the EC2 deployment machine to deploy the monitoring solution as a **cronjob** within the EKS cluster.

To achieve this, the following three components must be created:

1. **Cluster Role**
2. **Cluster Role Binding**
3. **aws-auth ConfigMap** modifications

Create Cluster Role and Cluster Role binding using the URL based yaml file.

curl -O https://eks-governance-marketplace.s3.us-east-1.amazonaws.com/Cluster-Permissions-for-aws-auth-configmap-for-ec2.yaml

kubectl apply -f Cluster-Permissions-for-aws-auth-configmap-for-ec2.yaml

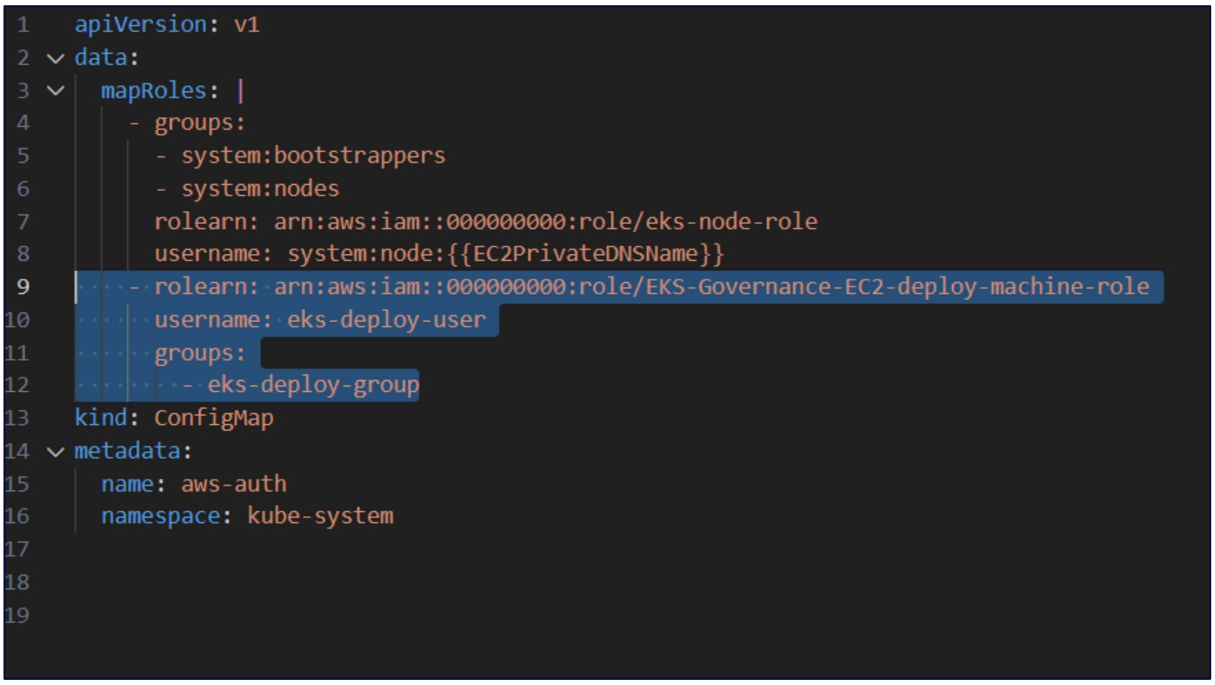
next is you can download the following file just to refer and make the aws-auth configmap changes accordingly in each EKS cluster

sample aws-auth configmap file - https://eks-governance-marketplace.s3.us-east-1.amazonaws.com/aws-auth-configmap.yaml

command to edit aws-auth configmap as follows -

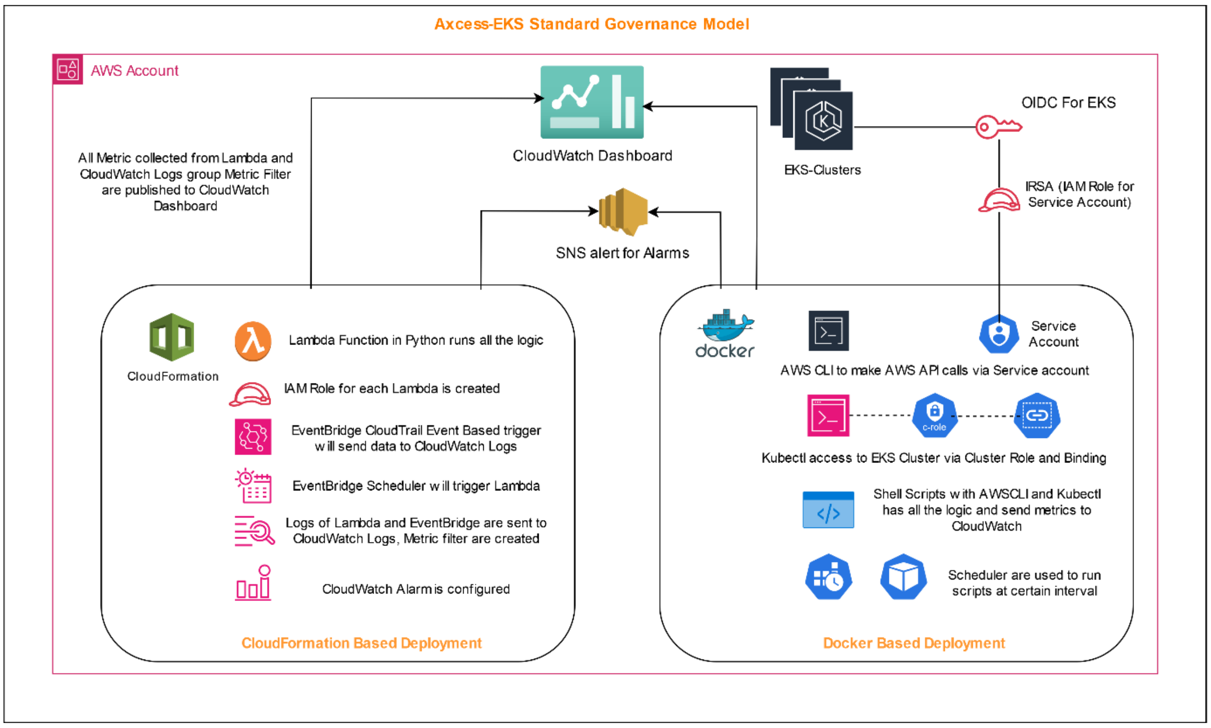
kubectl edit configmap aws-auth -n kube-system

sample aws-auth configmap change as follows – the highlighted lines to-be added, make sure to provice the IAM Role created for EC2 Deploy machine and username and groups has to-be exactly SAME.



```yaml
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::000000000:role/eks-node-role
      username: system:node:{{EC2PrivateDNSName}}
    - rolearn: arn:aws:iam::000000000:role/EKS-Governance-EC2-deploy-machine-role
      username: eks-deploy-user
      groups:
        - eks-deploy-group
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
```

3 ARCHITECTURE DIAGRAM


Axcess-EKS Standard Governance Model
AWS Account
All Metric collected from Lambda and CloudWatch Logs group Metric Filter are published to CloudWatch Dashboard
CloudWatch Dashboard
EKS-Clusters
OIDC For EKS
IRSA (IAM Role for Service Account)
SNS alert for Alarms
CloudFormation
Lambda Function in Python runs all the logic
IAM Role for each Lambda is created
EventBridge CloudTrail Event Based trigger will send data to CloudWatch Logs
EventBridge Scheduler will trigger Lambda
Logs of Lambda and EventBridge are sent to CloudWatch Logs, Metric filter are created
CloudWatch Alarm is configured
CloudFormation Based Deployment
docker
Service Account
AWS CLI to make AWS API calls via Service account
c-role
Kubectl access to EKS Cluster via Cluster Role and Binding
Shell Scripts with AWSCLI and Kubectl has all the logic and send metrics to CloudWatch
Scheduler are used to run scripts at certain interval
Docker Based Deployment

The objective is to monitor **EKS infrastructure-level** and **Kubernetes-level compliance items** through the **CloudWatch Dashboard** and alerts via **CloudWatch Alarms**.

The deployment is executed using **CloudFormation Nested Stacks**, which consist of multiple **CloudFormation templates** for each micro-solution. This deployment process involves creating resources such as:

1. **IAM Role for Lambda Function**
2. **Lambda Function**
3. **EventBridge Rules** (both event-based and scheduled)
4. **CloudWatch Log Groups** and **Metric Filters**
5. **CloudWatch Dashboard** and **Alarms** for each metric

Within the **EKS cluster**, the **cronjob** is deployed using **kubectl**. This cronjob runs every minute, collects the metrics, and sends them to **CloudWatch Metrics** for further analysis through the **CloudWatch Dashboard** and **Alarms**.

The automation leverages **AWS CLI** and **kubectl** in combination to fetch the required data and report metrics to AWS. The **AWS CLI** utilizes a **Service Account**, where the **IAM Role** is specified in the annotation. The same **IRSA (IAM Role for Service Account)** grants the necessary permissions to make API calls to the required AWS

services. Additionally, the **IRSA** must be integrated with **OIDC (OpenID Connect)** for the EKS cluster.
On the other hand, **kubectl** leverages the **Cluster Role** and **Cluster Role Binding** to access Kubernetes resources.

4 4. ROLLBACK PROCESS/ HOW TO HANDLE FAILED DEPLOYMENTS

- Delete the EC2 Deployment machine
- Delete the CloudFormation Parent Stack and all child stacks
- Delete the EKS Cronjob from all Clusters. (Command - Kubectl delete cronjob eks-governance)
- Revert all the pre-requisite configurations