Copy of EKS limited access to IAM users using role

NOTE: To perform most of the action we have to run commands using that user which have admin access on eks kubeconfig.

STEP-1: Create cluster role and cluster role binding. here we have defined limited access on eks.

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: reader
rules:
- apiGroups: ["*"]
  resources: ["deployments", "configmaps", "pods", "secrets",
"services", "namespaces", "pods/log"]
  verbs: ["get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: reader
subjects:
- kind: Group
  name: reader
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: reader
  apiGroup: rbac.authorization.k8s.io
```

STEP-2: Create AWS policy

name: eks-limited-access-policy-2

```
{
    "Statement": [
        {
            "Action": [
                "eks:AccessKubernetesApi",
                "eks:Describe*",
                "eks:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

STEP-3: Create AWS role to which we have to attache limited access policy.

- choose: aws-account [same account id]
- name: limited-access-role
- Attach step-1 created policy to the role.

Check Trust-relationship should look like below.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::68498XXX5XX:root"
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

STEP-4: Create another policy to establish trust between user and role.

- name: eks-assume-policy
- add policy json for created role on step-2

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": "arn:aws:iam::68498XXXX:role/limited-access-
role"
        }
    ]
}
```

STEP-5: Create user to which we have to provide restricted access.

- name: test-developer
- attach policy which we have created on step-3.
- create user with access-key and secret key this will be used on local-system.

STEP-6: Run aws configure command to create profile for step-4 user.

command: aws configure --profile developer

NOTE: Pass access-key and secret-key

STEP-7: {OPTIONAL} It is to verify that we have access to role.

command:

```
aws sts assume-role --role-arn arn:aws:iam::68498XXXXX:role/limited-access-role --role-session-name test --profile developer
```

STEP-8: we need to update the kubernetes config file because we need to add a role on kubeconfig. To do this we have to run this using user which have admin access to eks environment.

command:

```
aws eks update-kubeconfig --profile dev-eks-ankit --name project-dev --region eu-west-2
```

Now we need to edit the kubernetes configmap to provide access to role

```
kubectl edit -n kube-system configmap/aws-auth
```

we have added 13-to-17 line on config-map. where we are passing group name and role ARN and username which is same name of role ARN.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::68498XXXX:role/eks_node_group_role
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - reader
      rolearn: arn:aws:iam::68498XXXXX:role/limited-access-role
      username: limited-access-role
kind: ConfigMap
metadata:
  creationTimestamp: "2022-05-27T04:51:53Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "3389653"
  uid: f94ede82-c87e-4a0d-a1fb-27a7XfbbXXXX
```

STEP-9: We need to create additional profile for role.

vi .aws/config

and add 4-to-6 line. here source_profile would be the profile name define in .aws/credentials for IAM User which we have created above.

```
  [default]
region = ap-south-1

[profile eks-limited-access]
role_arn = arn:aws:iam::68498XXXXX:role/limited-access-role
source_profile = developer
```

STEP-10: we need to update kubeconfig last time using profile eks-limited-access

```
aws eks update-kubeconfig --profile eks-limited-access --name project-
dev --region eu-west-2
```

verify using below command it will have added the profile on kubeconfig

```
kubectl config view --minify
```

STEP-11: Run commands to check what permission we have as a user/role on eks environment.

```
kubectl auth can-i create pod
kubectl auth can-i delete pods
kubectl auth can-i get pods
```