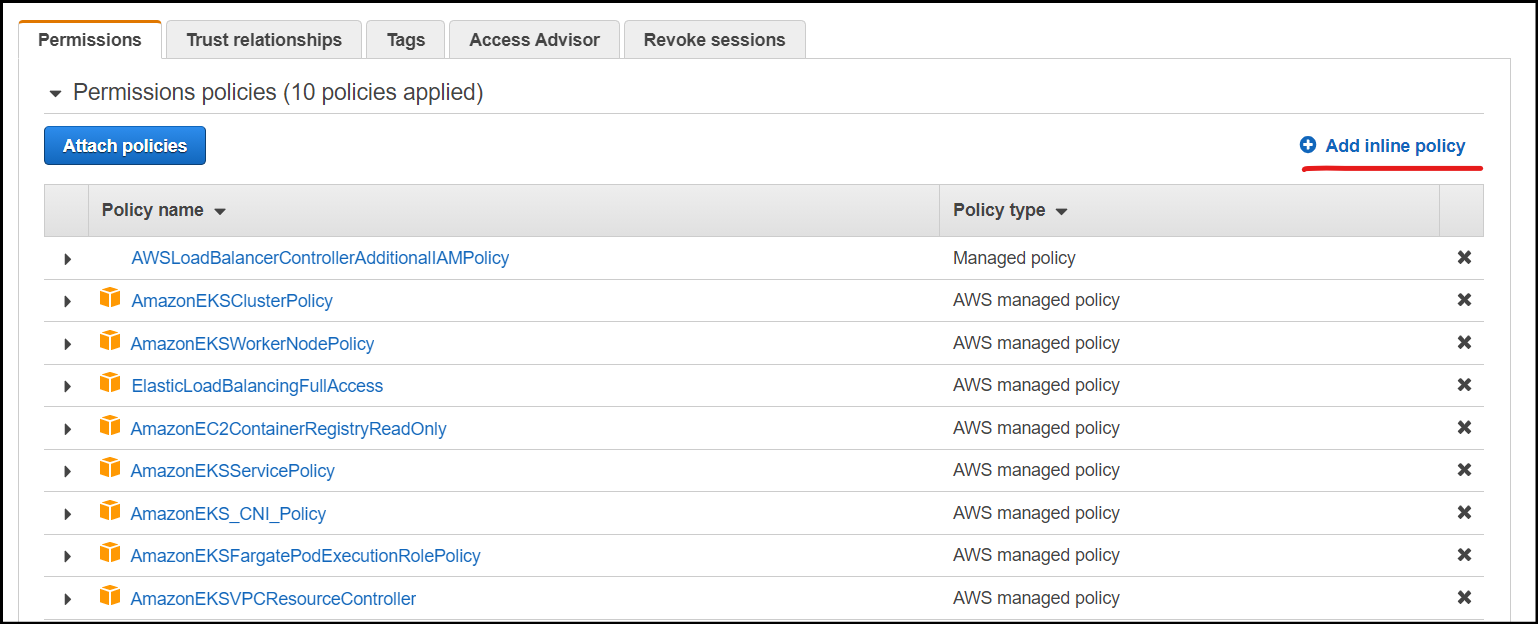
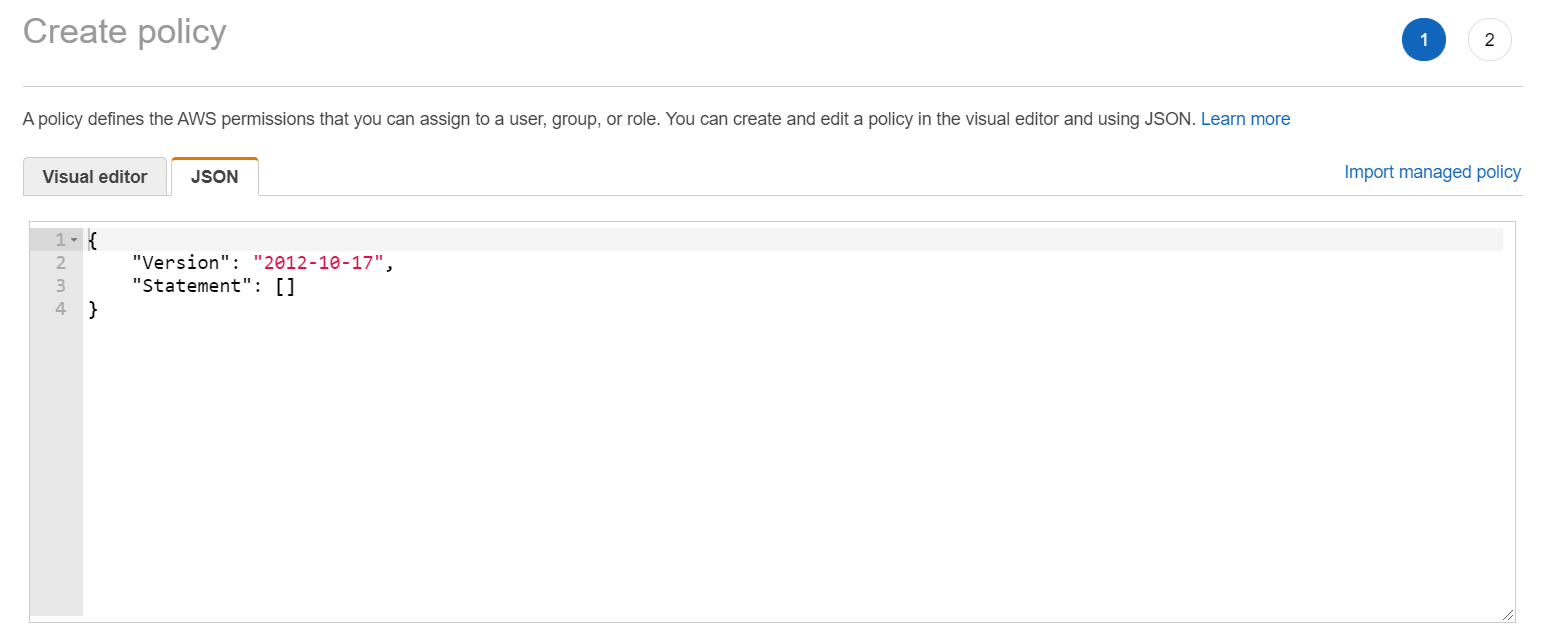
**Create an IAM Policy** **for Worker Node**

1. After the creation of EKS, The Cluster Autoscaler requires the following IAM permissions to make calls to AWS APIs on your behalf.
2. Open the Node policy which was created during EKS setup and click on “***Add inline policy***”

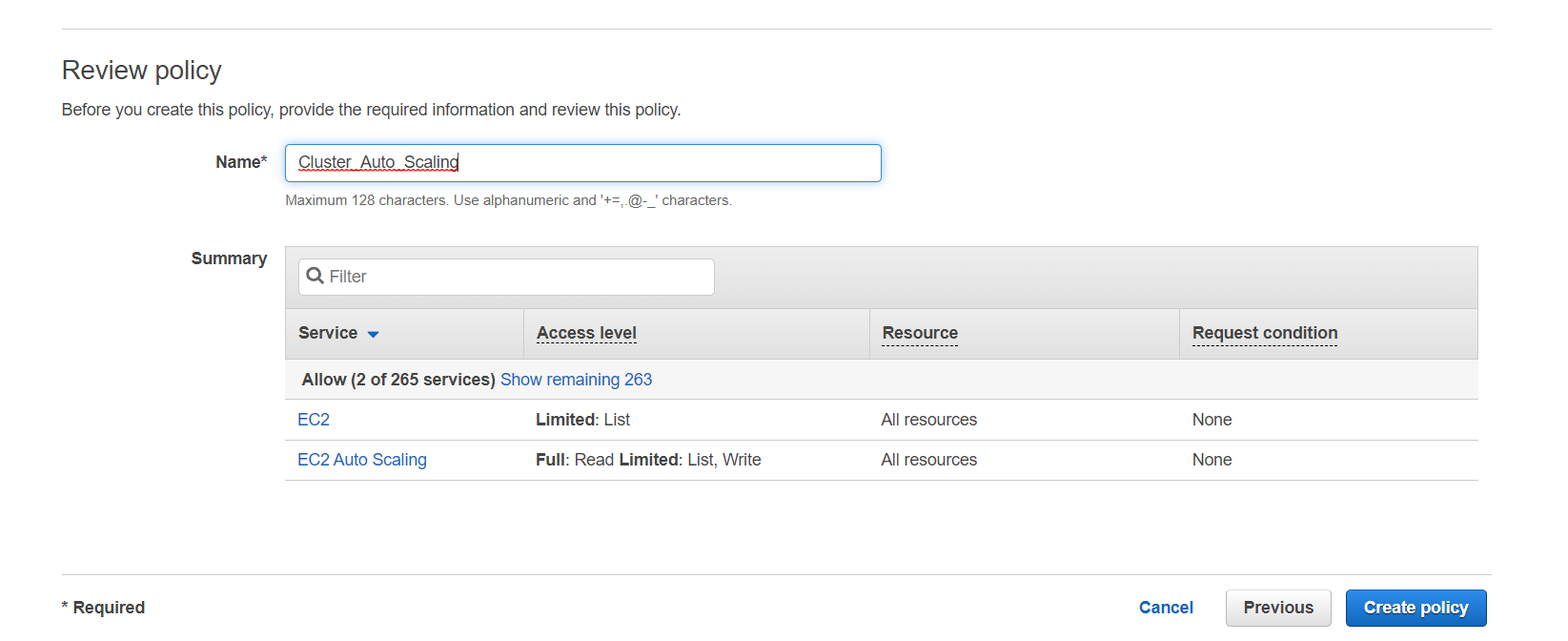




Add below json data into the inline policy field and click on review policy. Give a name for your policy and create the same.

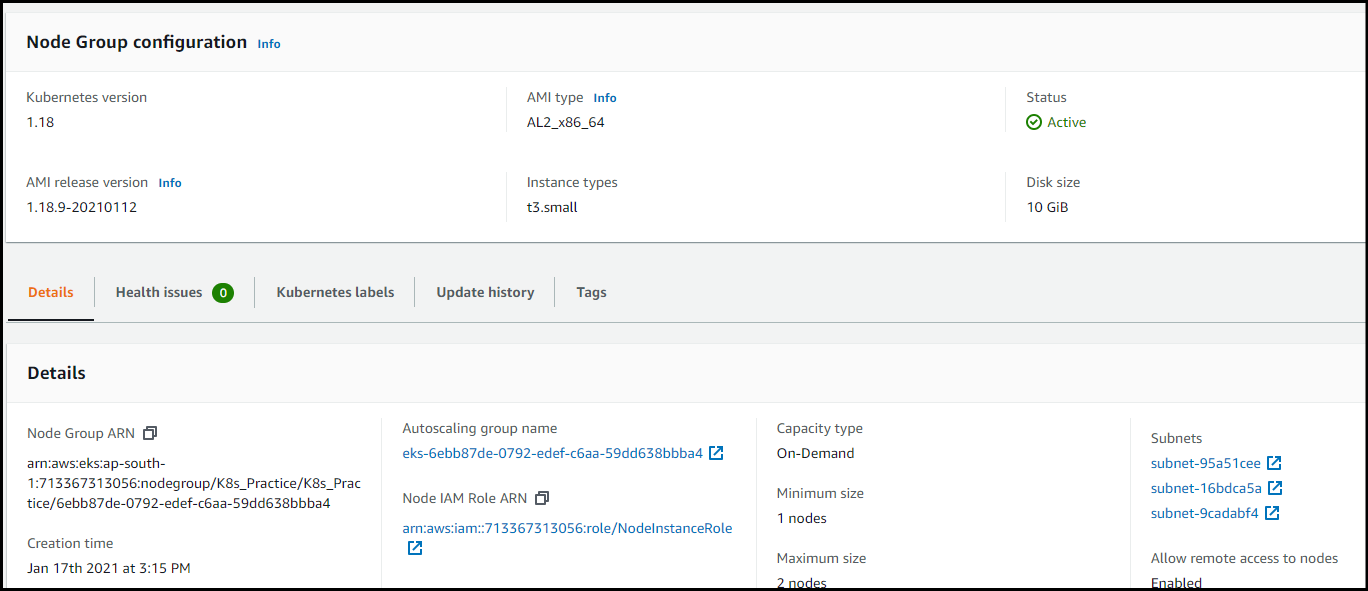
Note: This is the policy which will enable our EKS to access ASG and create instances based on the ASG template.

{  
"Version": "2012-10-17",  
"Statement": [  
{  
"Effect": "Allow",  
"Action": [  
"autoscaling:DescribeAutoScalingGroups",  
"autoscaling:DescribeAutoScalingInstances",  
"autoscaling:DescribeLaunchConfigurations",  
"autoscaling:DescribeTags",  
"autoscaling:SetDesiredCapacity",  
"autoscaling:TerminateInstanceInAutoScalingGroup",  
"ec2:DescribeLaunchTemplateVersions"  
],  
"Resource": "\*"  
}  
]  
}

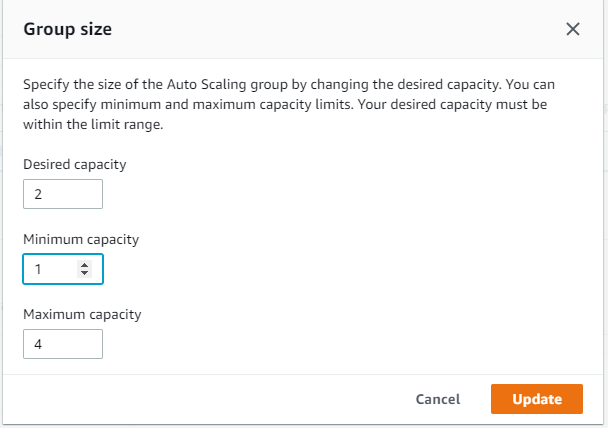


# Worker node ASG

1. Go to the EKS console and open the Auto Scaling group by clicking on “Autoscaling group name”



1. Update the ASG config of how many nodes we requires for the Node Group



# Deploy the Cluster Autoscaler

1. Deploy the Cluster Autoscaler to your cluster with the following command.

***kubectl apply -f*** [*https://raw.githubusercontent.com/kubernetes/autoscaler/master/cluster-autoscaler/cloudprovider/aws/examples/cluster-autoscaler-autodiscover.yaml*](https://raw.githubusercontent.com/kubernetes/autoscaler/master/cluster-autoscaler/cloudprovider/aws/examples/cluster-autoscaler-autodiscover.yaml)

1. Add the cluster-autoscaler.kubernetes.io/safe-to-evict annotation to the deployment with the following command.

***kubectl -n kube-system annotate deployment.apps/cluster-autoscaler cluster-autoscaler.kubernetes.io/safe-to-evict="false"***

1. Edit the Cluster Autoscaler deployment with the following command.

***kubectl -n kube-system edit deployment.apps/cluster-autoscaler***

1. **And once the yaml file is opened, edit the file by adding/editing highlighted data**

***spec:***

***containers:***

***- command:***

***- ./cluster-autoscaler***

***- --v=4***

***- --stderrthreshold=info***

***- --cloud-provider=aws***

***- --skip-nodes-with-local-storage=false***

***- --expander=least-waste***

***- --balance-similar-node-groups***

***- --skip-nodes-with-system-pods=false***

***- --node-group-auto-discovery=asg:tag=k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/K8s\_Practice***

1. **Set the Cluster Autoscaler image tag to the version of your EKS cluster.**

kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-autoscaler=us.gcr.io/k8s-artifacts-prod/autoscaling/cluster-autoscaler:v1.18.1

1. **View your Cluster Autoscaler logs with the following command.**

kubectl -n kube-system logs -f deployment.apps/cluster-autoscaler

**If u see any permission related issue then check if u have attached inline policy of ASG in your node role.**

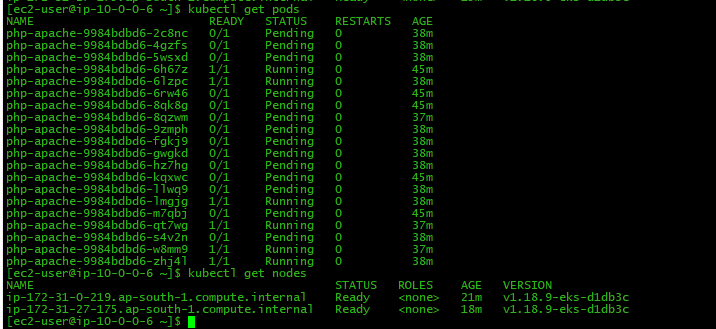
# Checking Cluster Autoscaler Working

1. **Deploy a 1 sample application.**

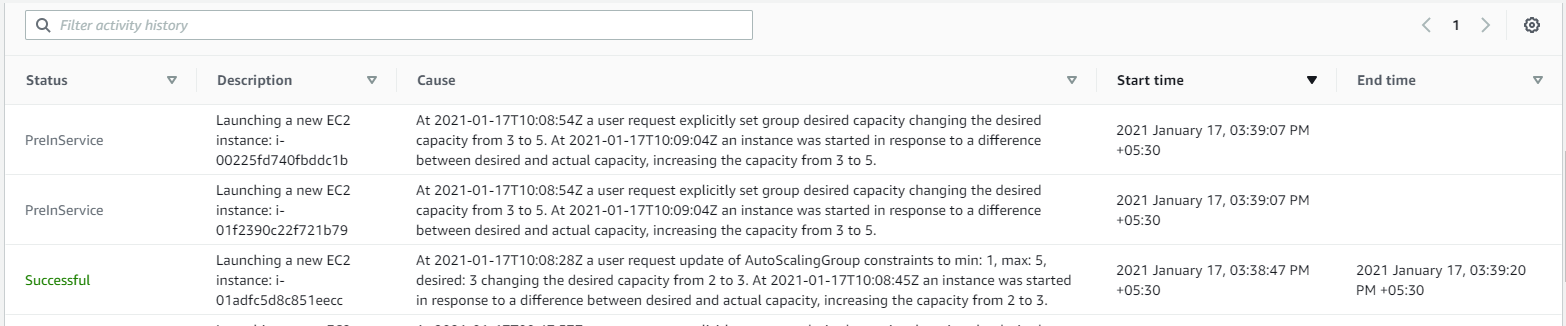


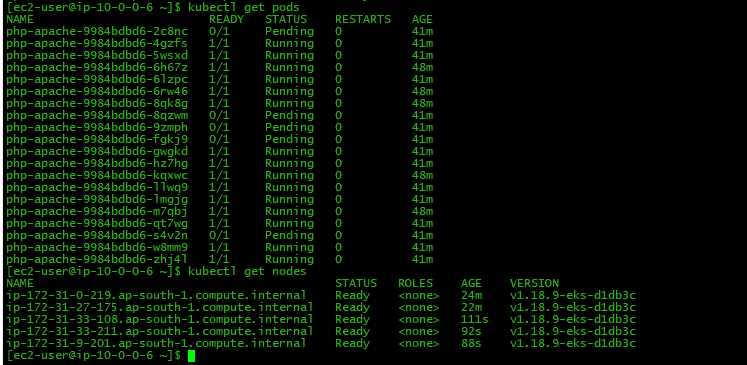
1. **Once the above deployment is deployed, EKS will check if it can schedule pods on current node with above resource specs, and if it can’t it will auto scale new nodes and schedule pods on them.**

**Before ASG kicking in!**

****

**After ASG kicking in!**

****

****