Kubernetes Cluster with Amazon EKS[kubectl]

Tuesday, 25 October, 2022 08:22 PM

What is Amazon EKS?
Amazon EKS (Elastic Container Service for Kubernetes) is a managed Kubernetes service that allows you to run Kubernetes on AWS without the hassle of managing the Kubernetes control plane.

- EKS stands for "Elastic Kubernetes Service"
- EKS is a fully managed AWS service
- EKS is the best place to run K8S applications because of its security, reliability and scalability
- EKS can be integrated with other AWS services such as ELB, CloudWatch, Autoscaling, IAM and VPC
- EKS makes it easy to run K8S on AWS without needing to install, operate and maintain your own k8s control plane.
- Amazon EKS runs the 'K8S control Plane' across three availability zones in order to ensure high availability and it automatically detects and replaces unhealthy masters.
- AWS will have complete control over Control Plane. We don't have control on Control Plane. We need to create Worker Nodes and attach to Control Plane.

Note: We will create Worker Nodes Group using Autoscaling Group

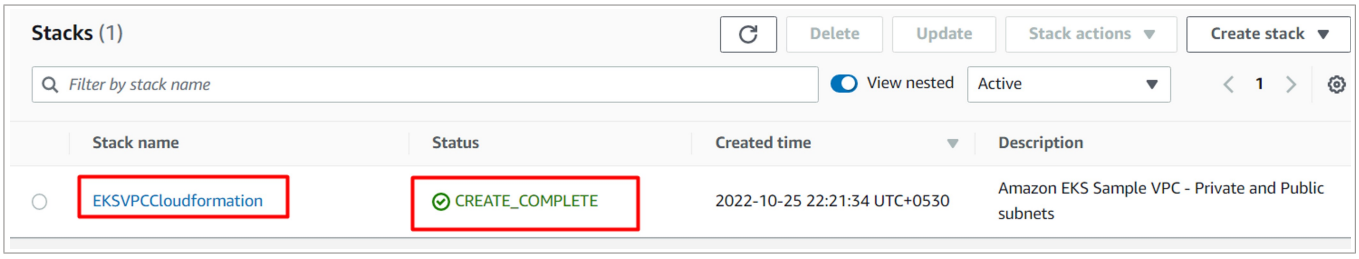
prerequisites:-

1	Aws account with admin privileges
2	Instance to manage/access EKS cluster using kubectl
3	Aws CLI access to use kubectl utility

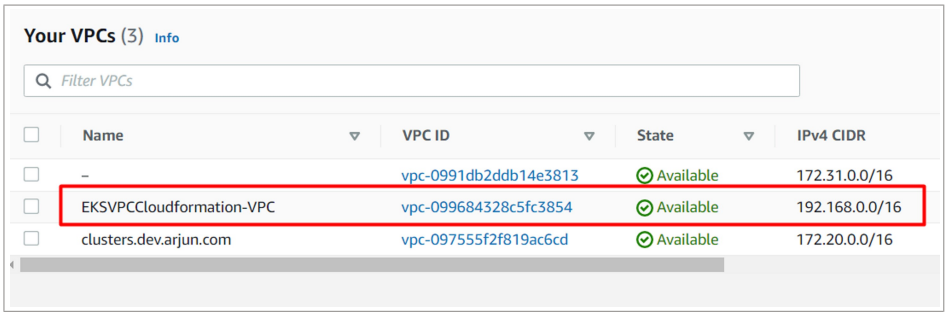
1. **Creating VPC using Cloud Formation**

- Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation
- Choose Create stack [With new resources (standard)]
- Under Prerequisite - Prepare template, Select [template is ready] and then under Specify template, select Amazon S3 URL.
- Paste the given S3 URLs into the text area under Amazon S3 URL and choose Next

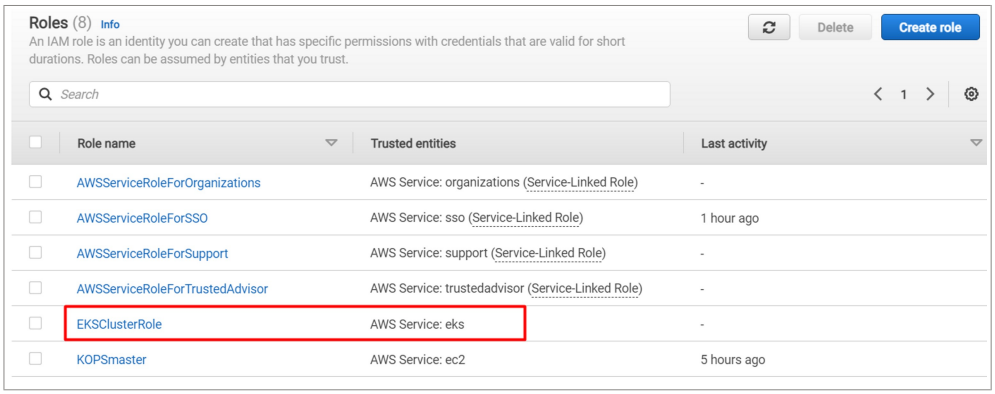
  https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
- Specify stack details, Stack Name [EKSVPCCloudformation], choose next.
- Configure stack options [optional], choose next.
- Review EKSVPCCloudformation and click on submit to create Stack.



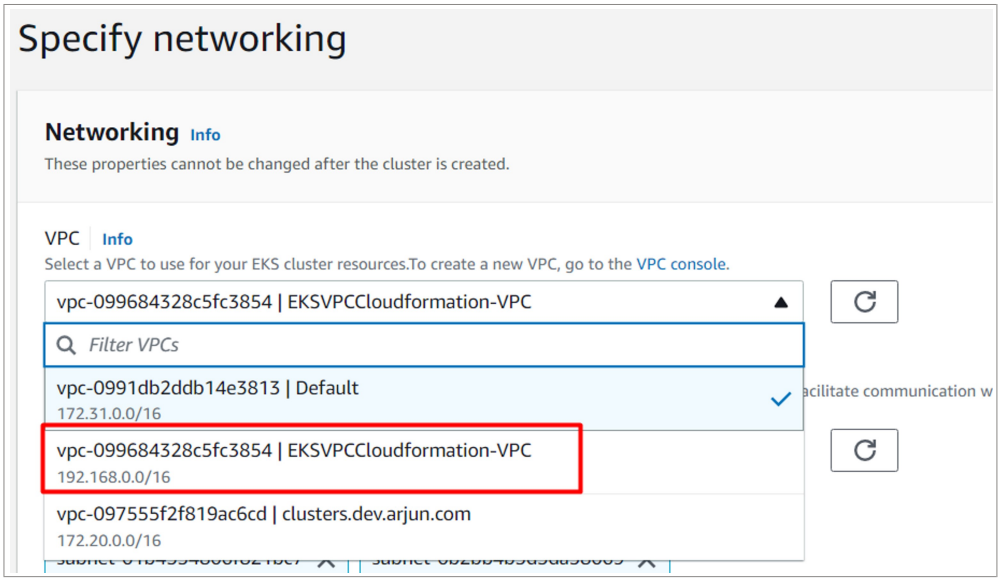
[Note]:by using cloud formation template I'm creating custom VPC in my current region



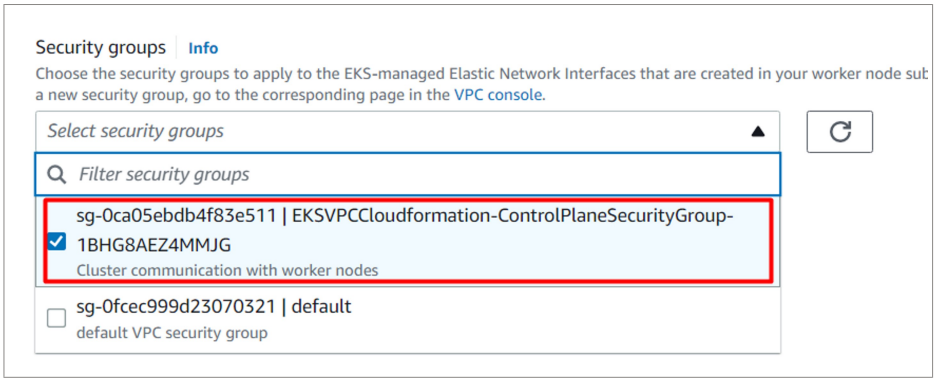
2. Creating IAM role in AWS
- Under access management, select roles and choose create role.
- Select trusted entity type as AWS service
- Use case as EKS - EKS Cluster and choose next.
- Add permissions [optional], choose next.
- Name your role [EKSClusterRole],review other details, and click on create role.



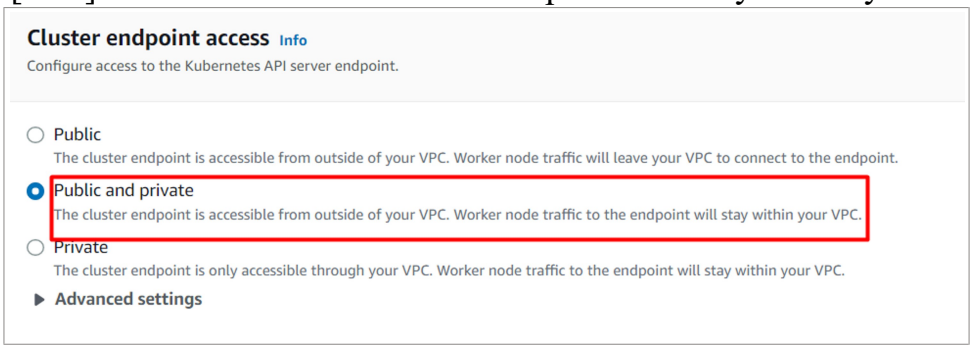
3. **Creating EKS cluster using newly created VPC and IAMrole**
- Visit eks page from search box
- Click on clusters from side menu, click on create cluster[create]
- In cluster configuration page, name your cluster[EKS-cluster] and choose next.
- In networking tab choose newly created VPC



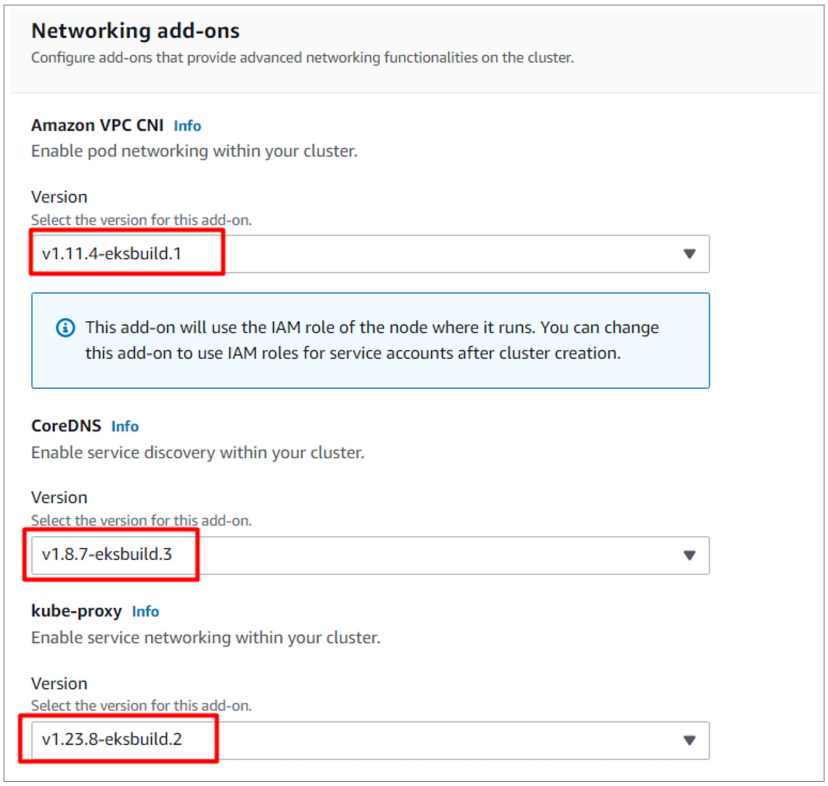
- In security group, select CloudFormation created security group [controlplane-securitygroup]



- In cluster end point access tab, I'm selecting public and private endpoint
  [note] The cluster endpoint is accessible from outside of your VPC.
  [note] Worker node traffic to the endpoint will stay within your VPC.

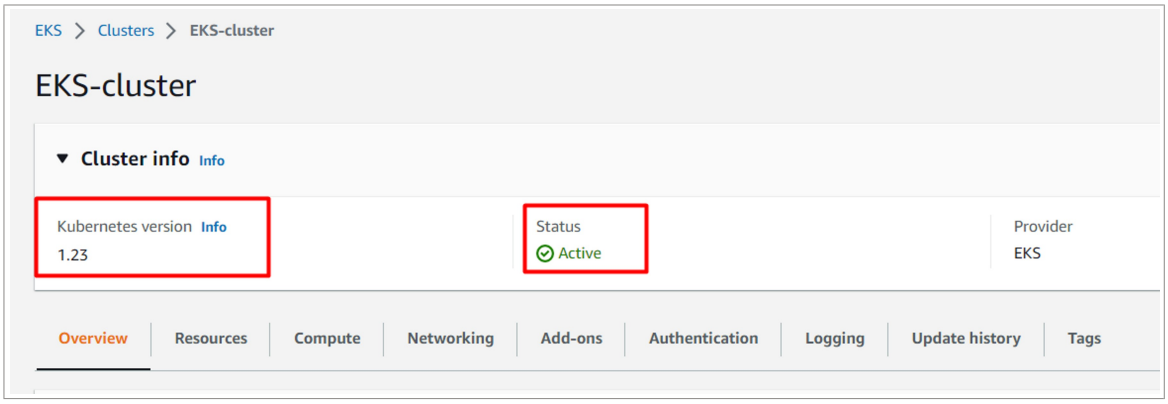


- Networking add-ons tab, select all latest versions which are available.



- Control plane logging [optional], choose next

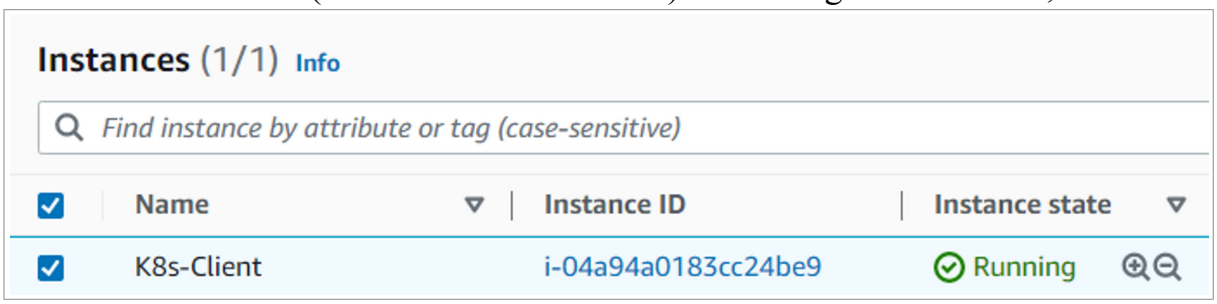
- Review and click on create
  Note: it will take around 5-10 mins to create Cluster



- From all the above steps we have created control-plane Cluster using VPC and IAMrole.
- In order to manage and access the control plane cluster, we are creating a ec2 instance and we will install kubectl and configure AWS credentials on it.

4. Installing and configuring kubectl for EKS Cluster in AWS ec2 instance.

- We will launch a [t2.micro] instance and will use it as a Kubernetes client.
- Launch the instance (Amazon AMI Preferred) and configure AWS CLI, kubectl on it.



- Connect to the newly created instance [k8s-Client]
- download kubectl binary with curl on Linux and run the following commands.

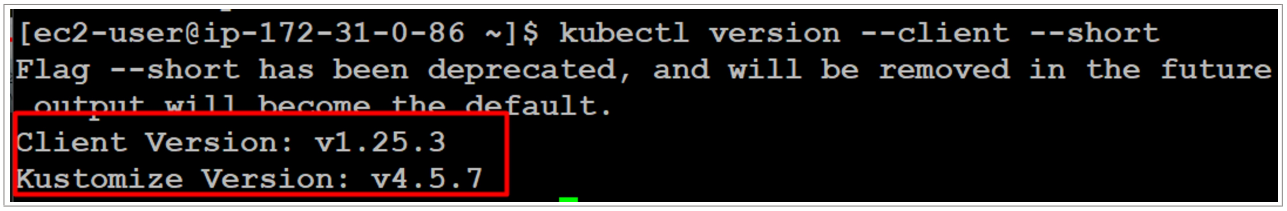
```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.23.6/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl
```

- Verify the status of kubectl using kubectl version command

```
kubectl version -- client
```



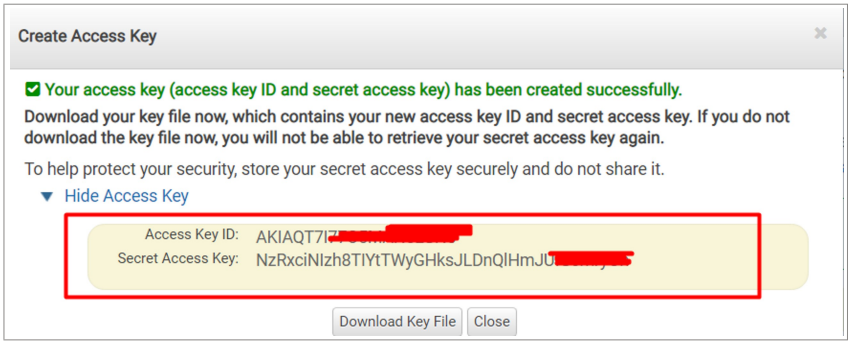
- Install AWS-CLI in client machine using below commands

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo yum install unzip
unzip awscliv2.zip

run above command as one single command
sudo ./aws/install
sudo ./aws/install --update [use this if you are using amazon Linux instance]
aws --version [verify the installation status]
```

5. **Configuring AWS CLI with Credentials [im using root credentials]**

- In the security credentials tab of our account, we can generate access key for current user.
- Click on new access key and make a note of [access key ID] and [secret access key]



- Now in k8s client machine we need to enter above aws credentials

```
aws configure
```

- Configure our cluster data with the kubectl in client machine[k8s-client]

```
aws eks update-kubeconfig --region us-west-2 --name EKS-cluster
```

Note: this command will update kubeconfig file to give access to the cluster.

From all above steps, we have created control plane and client machine to interact, manage the control plane.

6. **Creating IAMrole For EKS worker Nodes**

- Under access management, select roles and choose create role.
- Select trusted entity type as AWS service
- Use case as EC2 and choose next.
- Add permissions policies and choose next.
  - AmazonEC2ContainerRegistryReadOnly

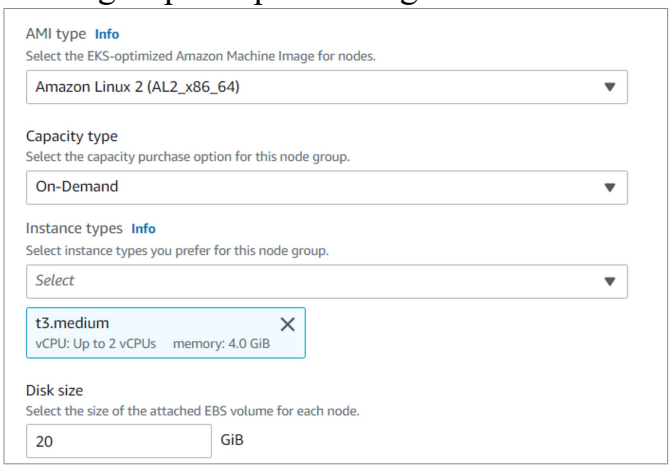
- AmazonEKS_CNI_Policy AWS managed
- AmazonEKSWorkerNodePolicy

- Name your role [EKSWorkerNodeRole],review other details, and click on create role.

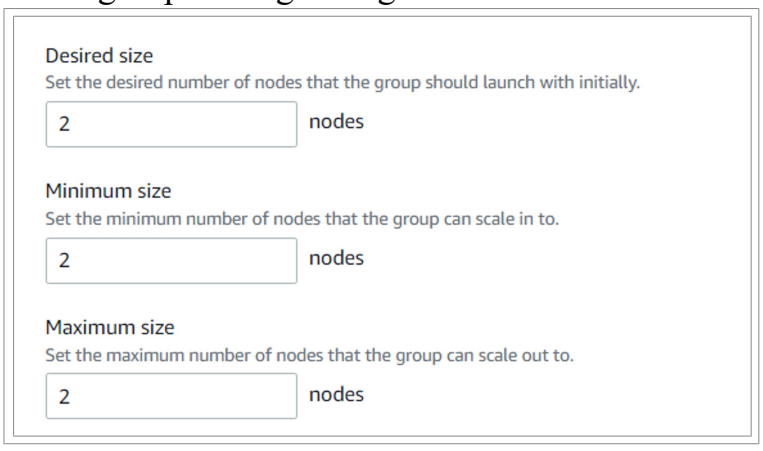
7. Creating worker node group under EKS-Cluster

- Go to Amazon Elastic Kubernetes Service page and select newly created EKS-cluster
- Under eks-cluster, choose compute tab and scroll down until you find node groups
- Click on add node group
- name your node group [k8s-node-group], select newly created [EKSworkernodeRole] in Node IAMrole and choose next
- By default required details are auto filled by eks with minimum requirements. Click next

Node group compute configuration



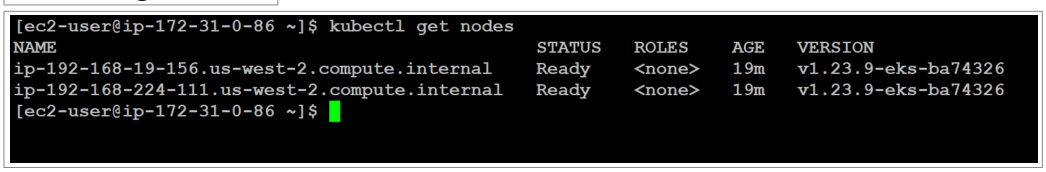
Node group scaling configuration



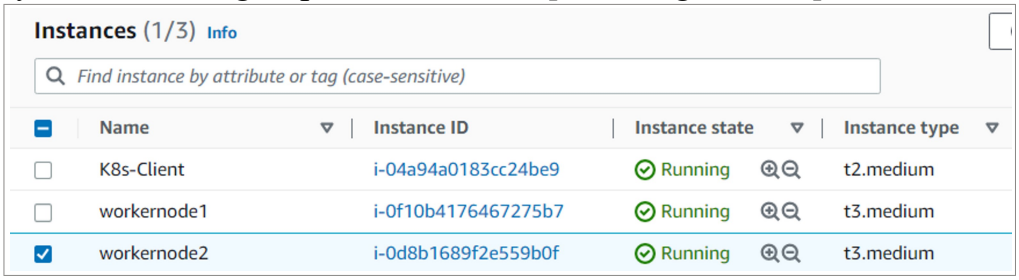
- by default eks will auto select subnets in Node group network configuration, click next

8. Once the node group is created, check node details in k8s-client instance

Kubectl get nodes



Note: We will not be able to see Master node of EKS as it is managed by AWS CloudFormation. Only worker node group is visible from [kubectl get nodes] and aws ec2 dashboard.



- kubectl get all resources in namespace

```
kubectl get pod --all-namespace
```

```
[ec2-user@ip-172-31-0-86 ~]$ kubectl get pod --all-namespaces
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE
kube-system   aws-node-c8zw6             1/1     Running   0          21m
kube-system   aws-node-19dr2             1/1     Running   0          21m
kube-system   coredns-8467d5b874-2g6bf   1/1     Running   0          82m
kube-system   coredns-8467d5b874-tvh25   1/1     Running   0          82m
kube-system   kube-proxy-bpgrz           1/1     Running   0          21m
kube-system   kube-proxy-n29qh           1/1     Running   0          21m
```

==========

Accessing cluster and cluster nodes from windows client [powershell]

==========

Prerequisites:-

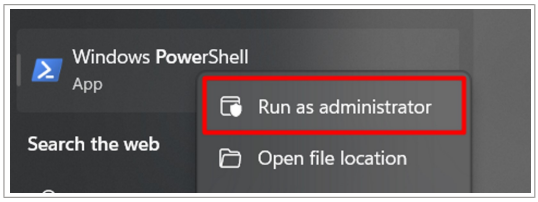
❖ Chocolatey package Manager

What is Chocolatey?

Chocolatey is a command line application installer for Windows based on a developer-centric package manager called NuGet. Unlike manual installations, Chocolatey adds, updates, and uninstalls programs in the background requiring very little user interaction.

1. **Installing Chocolatey package manager in windows machine[Individual Use]**

- First, ensure that you are using an administrative powershell[run terminal as administrator]



- Now execute the following command in powershell:

```
Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

- Wait a few seconds for the command to complete.

```
Installing Chocolatey on the local machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
  Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
  before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
  (i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
  and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.
```

- If you don't see any errors, you are ready to use Chocolatey.

2. Installing aws-cli V2 using Chocolatey package manager.

- To install AWS Command Line Interface v2 (Install), run the following command from the command line or from PowerShell:

```
choco install awscli
```

```
PS C:\Users\Arjun> aws --version
aws-cli/2.8.6 Python/3.9.11 Windows/10 exe/AMD64 prompt/off
PS C:\Users\Arjun>
```

3. Installing kubectl [1.25.3]using Chocolatey package manager

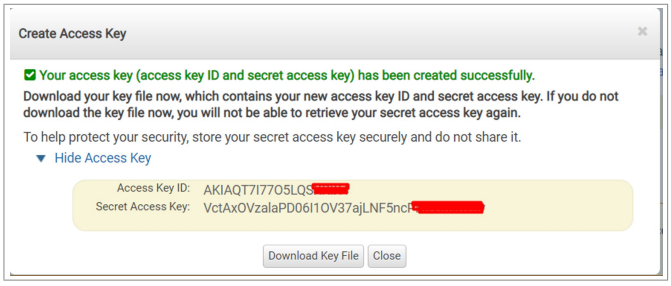
- To install Kubernetes (CLI), run the following command from the command line or from PowerShell:

```
choco install kubernetes-cli
```

```
PS C:\Users\Arjun> kubectl version --client
WARNING: This version information is deprecated and will be replaced with the output from kubectl version --short.  Use
--output=yaml|json to get the full version.
Client Version: version.Info{Major:"1", Minor:"25", GitVersion:"v1.25.3", GitCommit:"434bfd82814af038ad94d62ebe59b133fcb
50506", GitTreeState:"clean", BuildDate:"2022-10-12T10:57:26Z", GoVersion:"go1.19.2", Compiler:"gc", Platform:"windows/a
md64"}
Kustomize Version: v4.5.7
PS C:\Users\Arjun>
```

4. Generating AWS CLI credentials

- First, log into the AWS management console and Now go to the Security credentials tab.
- Scroll down to the Access key section and click on the create access key button to generate the AWS CLI credentials for the user account.



5. Configure AWS Credentials on Windows client

- To configure AWS CLI credentials. Just run the following command

```
aws configure
```

- When you run this command, the CLI will prompt you to provide the following four attributes
  AWS access key ID
  AWS secret access key
  Default region

Default output format

- Enter generated access key and secret key in commandline and set your default region.

```
PS C:\Users\Arjun> aws configure
AWS Access Key ID [****************PBXY]:
AWS Secret Access Key [****************rghE]
Default region name [us-west-2]:
Default output format [json]:
```

6. **Configuring cluster details on commandline**

- run the following command to find the cluster details available in specific region
  aws eks list-clusters --region [region-code] ex: us-west-2

```
PS C:\WINDOWS\system32> aws eks list-clusters
{
    "clusters": [
        "EKS-Cluster"
    ]
}
```

- Above command will list all clusters which are available in us-west-2

7. **Configuring kubectl for EKS**

- we need to configure our kubeconfig file to use with EKS. Just run the following command
  To update cluster and region details with kubectl

  `aws eks update-kubeconfig --region [regioncode] --name [cluster name]`

  `Ex: aws eks update-kubeconfig --region us-west-2 --name EKS-Cluster`

  Above command will create .kube folder and config file in our windows machine

8. **We have successfully updated our cluster details**.

- To view the cluster service details, run the following command:

  `kubectl get services`

```
PS C:\Users\Arjun> kubectl get service
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   157m
```

- To view the status of worker nodes, run the following command

  `kubectl get nodes`

```
[ec2-user@ip-172-31-0-86 ~]$ kubectl get nodes
NAME                                            STATUS   ROLES    AGE   VERSION
ip-192-168-19-156.us-west-2.compute.internal    Ready    <none>   19m   v1.23.9-eks-ba74326
ip-192-168-224-111.us-west-2.compute.internal   Ready    <none>   19m   v1.23.9-eks-ba74326
[ec2-user@ip-172-31-0-86 ~]$
```