# 2. Set Up EKS 1.20 cluster with eksctl in 15 minutes

__NOTE__:
Even if this course touts __Zero to Hero AWS EKS Handson Production Best Practices 2020__, for illustration purpose, we'll use __eksctl__ to spin up AWS VPC and EKS cluster in one command. In production, you should be using __IaC__ such as __Terraform (which could require an entire course for it)__ (or AWS-vendered CloudFormation).

# 2.1 Create AWS IAM user, access key, and EKS cluster IAM role from Console

Create a free account first
```
https://aws.amazon.com/resources/create-account/
```

# 2.2 Install AWS cli
Ref: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
```bash
# ref: mkdir homebrew && curl -L https://github.com/Homebrew/brew/tarball/master | tar xz --strip 1 -C homebrew
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"

# for Mac
brew install awscli

aws --version

# create "default" profile
# aws configure

# create "eks-demo" profile
aws configure --profile eks-demo

aws sts get-caller-identity
```

## Create AWS Profile
In production environment, it's easier to swtich to different AWS IAM user or IAM role identiy by `export AWS_PROFILE=PROFILE_NAME`.

Instead of using `default` profile created by above `aws configure`, you can create a named AWS Profile `eks-demo` in two ways:

1. `aws configure --profile eks-demo`

~~~~markdown
2. create profile entry in `~/.aws/credentials` file


To create profile entry in `~/.aws/credentials` file, do the followings:
```
vim ~/.aws/credentials
```
Enter `i` key and paste below lines into the file
```
[eks-demo]
aws_access_key_id=YOUR_ACCESS_KEY
aws_secret_access_key=YOUR_SECRET_ACCESS_KEY
aws_region = YOUR_REGION
```


Hit `escape`key and type `:wq!` to save and exit out from Vim.

Then check if new profile can be authenticated
```sh
export AWS_PROFILE=esk-demo

# successful output
aws sts get-caller-identity
{
    "UserId": "xxxxxxxxx",
    "Account": "12321313123131",
    "Arn": "arn:aws:iam::1231231231231:user/eks-demo"
}
```




# 2.3 Install aws-iam-authenticator (if aws cli is 1.20.156 or earlier)
```bash
# Mac
brew install aws-iam-authenticator

# Windows
# install chocolatey first: https://chocolatey.org/install
choco install -y aws-iam-authenticator
```


# 2.4 Install kubectl
Ref: https://kubernetes.io/docs/tasks/tools/install-kubectl/
```bash
# Mac
brew install kubectl

# Windows
~~~~

~~~~
choco install kubernetes-cli

kubectl version
```


# 2.5 Install eksctl
Ref: https://docs.aws.amazon.com/eks/latest/userguide/getting-started-
eksctl.html
```bash
# Mac
brew tap weaveworks/tap
brew install weaveworks/tap/eksctl
eksctl version

# Windows: https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html

# install eskctl from chocolatey
chocolatey install -y eksctl

eksctl version

# Windows: https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html
# install chocolatey first
https://chocolatey.org/install

# instakk eskctl from chocolatey
chocolatey install -y eksctl
```


# 2.6 Create ssh key for EKS worker nodes
```bash
ssh-keygen
eks_worker_nodes_demo.pem
```


# 2.7 Setup EKS cluster with eksctl (so you don't need to manually create VPC)
`eksctl` tool will create K8s Control Plane (master nodes, etcd, API server, e
tc), worker nodes, VPC, Security Groups, Subnets, Routes, Internet Gateway, et
c.
```bash
# use official AWS EKS AMI
# dedicated VPC
# EKS not supported in us-west-1

eksctl create cluster \
    --name eks-from-eksctl \
    --version 1.20 \
    --region us-west-2 \
~~~~

~~~
    --nodegroup-name workers \
    --node-type t3.medium \
    --nodes 2 \
    --nodes-min 1 \
    --nodes-max 4 \
    --ssh-access \
    --ssh-public-key ~/.ssh/eks_worker_nodes_demo.pem.pub \
    --managed
```

Output
```bash
[ℹ]  eksctl version 0.21.0
[ℹ]  using region us-west-2
[ℹ]  setting availability zones to [us-west-2b us-west-2a us-west-2c]
[ℹ]  subnets for us-west-2b - public:192.168.0.0/19 private:192.168.96.0/19
[ℹ]  subnets for us-west-
2a - public:192.168.32.0/19 private:192.168.128.0/19
[ℹ]  subnets for us-west-
2c - public:192.168.64.0/19 private:192.168.160.0/19
[ℹ]  using SSH public key "/Users/USERNAME/.ssh/eks_worker_nodes_demo.pem.pu
b" as "eksctl-eks-from-eksctl-nodegroup-workers-
51:34:9d:9e:0f:87:a5:dc:0c:9f:b9:0c:29:5a:0b:51"
[ℹ]  using Kubernetes version 1.20
[ℹ]  creating EKS cluster "eks-from-eksctl" in "us-west-
2" region with managed nodes
[ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the
 initial managed nodegroup
[ℹ]  if you encounter any issues, check CloudFormation console or try 'eksct
l utils describe-stacks --region=us-west-2 --cluster=eks-from-eksctl'
[ℹ]  CloudWatch logging will not be enabled for cluster "eks-from-
eksctl" in "us-west-2"
[ℹ]  you can enable it with 'eksctl utils update-cluster-logging --
region=us-west-2 --cluster=eks-from-eksctl'
[ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true,
privateAccess=false} for cluster "eks-from-eksctl" in "us-west-2"
[ℹ]  2 sequential tasks: { create cluster control plane "eks-from-
eksctl", 2 sequential sub-
tasks: { no tasks, create managed nodegroup "workers" } }
[ℹ]  building cluster stack "eksctl-eks-from-eksctl-cluster"
[ℹ]  deploying stack "eksctl-eks-from-eksctl-cluster"
[ℹ]  building managed nodegroup stack "eksctl-eks-from-eksctl-nodegroup-
workers"
[ℹ]  deploying stack "eksctl-eks-from-eksctl-nodegroup-workers"
[ℹ]  waiting for the control plane availability...
[✔]  saved kubeconfig as "/Users/USERNAME/.kube/config"
[ℹ]  no tasks
[✔]  all EKS cluster resources for "eks-from-eksctl" have been created
~~~

~~~
[ℹ]  nodegroup "workers" has 2 node(s)
[ℹ]  node "ip-192-168-20-213.us-west-2.compute.internal" is ready
[ℹ]  node "ip-192-168-39-97.us-west-2.compute.internal" is ready
[ℹ]  waiting for at least 1 node(s) to become ready in "workers"
[ℹ]  nodegroup "workers" has 2 node(s)
[ℹ]  node "ip-192-168-20-213.us-west-2.compute.internal" is ready
[ℹ]  node "ip-192-168-39-97.us-west-2.compute.internal" is ready
[ℹ]  kubectl command should work with "/Users/USERNAME/.kube/config", try 'k
ubectl get nodes'
[✔]  EKS cluster "eks-from-eksctl" in "us-west-2" region is ready
```

Once you have created a cluster, you will find that cluster credentials were a
dded in ~/.kube/config

```bash
# get info about cluster resources
aws eks describe-cluster --name eks-from-eksctl --region us-west-2
```

Output
```json
{
    "cluster": {
        "name": "eks-from-eksctl",
        "arn": "arn:aws:eks:us-west-2:202536423779:cluster/eks-from-eksctl",
        "createdAt": "2020-06-13T18:48:18.244000+07:00",
        "version": "1.20",
        "endpoint": "https://242F02260C230DA3D2C46D5C9035E46E.sk1.us-west-
2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::202536423779:role/eksctl-eks-from-eksctl-
cluster-ServiceRole-NHR5AAVMYKBY",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0820f91de866118c6",
                "subnet-033da8b1a4e094fd0",
                "subnet-0b2142f44f04cf336",
                "subnet-0cd3179fbb2403217",
                "subnet-079e58ed09df36c91",
                "subnet-0e8ff49f41d33141b"
            ],
            "securityGroupIds": [
                "sg-05e9063cc2cabd063"
            ],
            "clusterSecurityGroupId": "sg-0cf04559e421786da",
            "vpcId": "vpc-07f3adc9189a6baab",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": false,
~~~

```
            "publicAccessCidrs": [
                "0.0.0.0/0"
            ]
        },
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": false
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-west-
2.amazonaws.com/id/242F02260C230DA3D2C46D5C9035E46E"
            }
        },
        "status": "ACTIVE",
        "certificateAuthority": {
            "data": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5RENDQWJDZ0F3S
UJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnp
NQjRYRFRJd01EWXhNekV4TlRZek9Wb1hEVE13TURZeE1URXhOVFl6T1Zvd0ZURVRNQkVHQTFVRQpBe
E1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJ
BSlJmCkdKaHFSekhYbkNVVDRNNlQxZmNLKzNRVlVpZDhuMDFxV2RvSlIyRHJvUm9KTTFWUy9Iekc5Y
TVaUlhYNklLaTcKZUsyeVhzMkxVajErVXl0bGFRaVh5Q1N1Ykxlc3Q2Q2xhRXFBQ2FZNE5DVUNjc2J
1WFhlY2JnVEI4cGZlZ2FIUgovMGJFNUhkY1hiSEpzZ0lodmdjMFYxMHhDM2ZhV3lDbDdUTGQ2dkg0Y
m5RbktxTjdvU0pDTmtsbVZ4Z3hsajRNCnE1aWV6bW5LakRxUnEyN003bUw2YXNhNlBDWUN2QUZlL05
oaFNYcjVaWDRyYjcybUtoQW9Qb0FadFJPMFN1VFgKV1NUVGFkNFpCeXZMZkZIU1FJNzV1QnBoYUtLZ
TRBUWFpVGxPRHZhMUkyQmc1ejJQeS9yaDRMZXB1RjlMNzNoRApBQjF6R3AvQmdkS2ltMHBIOTVNQ0F
3RUFBYU1qTUNFd0RnWURWUjBQQVFIL0JBUURBZ0trTUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0RRW
UpLb1pJaHZjTkFRRUxCUUFEZ2dFQkFJTzhrQlBHQTYyK2ZJYzhGcjNEUE96Nk9VclEKYjc1dmR4VS9
xQUxJSGJxRk1SZ3NIOHkvWERmK2Y3TEI3QndiU1BUWTZraExPT2xPV3ByRzI3cFdnVGgrK2E4NApie
UJIU0xMbWhFZFAzZHFuVzVyS3piNVovdGF0S2lJZGFyQTZwQlhOR2lLWFZWaStMTmJLeVNhVGdVbXh
IVUpoCjZuSno5TXA2MFBMcDk4ZkJzVnROejhxZXFkelZrUFd4d1FRUWFLZ1hqdUNDakU2cjhvQmZmS
nVkVGpMUmxwWkoKT292cGFPRU9RRlcvRzBGM1lOQ2FiRFpnSmhUcGZSSnVmRkd4MkdoV1pFOGQ4TWJ
yUko2aExPZGswOTFFSTIwLwpJWUtHYklvZFhPMUIydGQ4eWRDOWh5M1RHOEhaekxMMUNxK3QxdGNQV
msxL0tMRXVwc1NLcGFocE5HWT0KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo="
        },
        "platformVersion": "eks.1",
        "tags": {}
    }
```

~~~
}
```


```bash
# get services
kubectl get svc
```


Output shows the default `kubernetes` service, which is the API server in mast
er node
```bash
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   38m
```


# 2.8 AWS Networking Basics Overview - Region, AZ, VPC and Subnet

![alt text](../imgs/eks_aws_architecture.png "K8s Architecture")
Master (AWS manages this, hence master nodes not visible in Console):
- three master nodes for HA
- security group for masters
- IAM role and instance profile for master nodes

Worker:
- arbitrary # of worker nodes
- auto scaling group (ASG)
- launch config for ASG (launch config is a template for ASG)
- security group for workers
- IAM role and instance profile for workers

AWS VPC:
- VPC
- Subnets for three availability zones (AZ) for us-west-2 region
- Route tables with routes
- Internet Gateway
- NAT gateway

Shared responsibility model for EKS
![alt text](../imgs/eks_shared_responsibility.png "K8s Architecture")


# 2.9 EKS Console Walkthrough
![alt text](../imgs/eks_console.png "K8s Architecture")
~~~