AWS EKS Cluster Setup

Introduction

This document helps you to create all the required resources to get started with Amazon EKS. We will be creating one managed node in a regular availability zone and one self – managed node in a Wavelength Zone.

Deployment Playbook

Below steps outline the deployment instructions to provision and configure the cluster resources

Prerequisites:

The following tools and resources need to be installed and configured to create and manage an Amazon EKS cluster. (Note you must go through At Your Service to request resource installation of each. Once installed you must contact At Your Service to complete execution of install in order for it to properly work)

1. **AWS CLIv2** – A command line tool for working with AWS services. Note, make sure the latest version is installed..
2. **kubectl** – A command line tool for working with Kubernetes Clusters
3. eksctl - a simple CLI tool for creating clusters on Amazon EKS. (Windows). Chocolatey must also be installed.
4. **Cluster IAM Role** – This role allows kubernetes clusters managed by Amazon EKS to make calls to other AWS services on your behalf to manage the resources that you use with the service.

Note: By clicking on the above resources, they will be directed to the respective pages which helps you to install and configure them based on your system specifications.

Create your Amazon EKS cluster IAM role (ie. aifiTpnA-EKSClusterRole)

1. Open the **IAM console** at https://console.aws.amazon.com/iam/.

verizon

2. Choose **Roles** and then **Create role**.
3. Enter or select from dropdown **EKS** Use cases for other AWS
4. Select **EKS - Cluste**r
5. Select **Next**:
6. On **Add Permissions** page review policies added or add new policies if necessary
7. Select **Next**
8. On Name, Review, and Create page
9. Role Details: **Enter Unique Role Name**(ie. Alpha-EKSClusterRole)
10. Add Tag (optional) Add metadata to the role by attaching tags as key–value pairs.
11. Select: **Create Role**

Create your Amazon EKS Node IAM role

1. Open the **IAM console** at https://console.aws.amazon.com/iam/.
2. Choose **Roles**, then **Create role**.
3. Choose **EC2** from the list of Common use cases under
4. Choose **Next**: On Add Permissions page select required Permission Policies.
5. In the Filter Permissions policies box, input **AmazonEKSWorkerNodePolicy** then enter**,**. **Check the box to the left** of AmazonEKSWorkerNodePolicy.
6. In the Filter policies box, input **AmazonEC2ContainerRegistryReadOnly,** then enter. **Check the box to the left** of AmazonEC2ContainerRegistryReadOnly.
7. Repeat the filter and select steps in add **AmazonEKS_CNI_Policy** policy to this role.
8. Choose **Next**. On the Name,review, and Create page
9. Input Role Name: For Role name, enter a unique name for your role, such as **NodeInstanceRole**
10. On **Step2 : Add permissions** section (Review permissions previously added or Edit)
11. Add Tags(Optional) Add metadata to the role by attaching tags as key–value pairs.
12. Choose **Create Role**
13. Review.
14. In the Roles Search box input the name of Role just created.

Create the Cluster

1. Navigate to EKS -> Click on **Create Clusters**
2. Configure Cluster: Enter a **Unique name** for the cluster (ie. abc-EKSCluster)
3. Kubernetes version: **Select** the latest version (default)
4. Cluster Service Role: **Select** from the dropdown the **IAM EKSClusterRole** previously created. This allows the kubernetes control plane to manage AWS resources on your behalf.
5. Secrets encryption and Tags are **optional**
6. Click on **NEXT**
7. Specify the networking properties.

   Note: These cannot be changed after the cluster is created.
8. **VPC**: Select the VPC in which you want to create the cluster. To create a new VPC, please follow VPC Console..

   Note: If you choose to Create VPC via Wizard, it will auto create all associated resources(Subnets, Routing Tables, NATgateway, Elastic IP, Internet gateway, S3 endpoint, NAT Gateway). However, you must create and associate Security Group/s and define inbound/outbound traffic)

   - **Create VPC using VPC Wizard**
     - Navigate to the VPC console: **Select** VPC Dashboard
     - Select **Launch VPC Wizard**
     - On **Create VPC** page, in VPC Setting:
     - Choose **Resources to Create** (VPC only or VPC,subnet,etc)
     - **Name tag auto generation**: Creates and labels all resources with Unique Name(VPC,subnet, route tables, Network Connections(internet gateway, S3)

- Input VPC Name(ie. abc-project) auto generation will create name for all resources
- **IPv4 CIDR block**:Input desired block (ie. 10.0.0.0/16)
- **IPv6 CIDR block**: Select No IPv6 CIDR block or use Amazon provided)
- Tenancy: Default
- Availability Zones(AZs): **Choose number of required AZs** (1, 2, or 3)
- Customize AZs: **Select from dropdown one for each AZ** (ie us-east-1)
- Number of **Public Subnets**: Select available choice (ie. 0 or 2)
- Customize **Public subnets** CIDR blocks: input ie. 10.0.2.0/24
- Number of **Private Subnets**: Select available choice (ie. 0, 1, 2)
- Customize **Private subnets** CIDR blocks: input ie. 10.0.1.0/24
- NAT gateway($): Choose number of AZ to create NAT gateway
  - ie None , **In 1 AZ** or 1 per AZ
- VPC endpoints: Choose None or **S3 Gateway**
- DNS options: Choose your option by Check Mark
  - Enable DNS hostnames
  - **Enable DNS resolution**
- Choose: **Create VPC**
- **On the Create VPC Workflow page:** wait for VPC Resources to be created Successfully. Note you will be able to see and select links to all resources created.
- Note: You will need to **Enable auto-assign public IPv4 address on your subnets**. Goto your subnet>actions>edit Subnet settings> check mark Enable auto-assign public IPv4 address>SAVE. (DO This process for each subnet)

9. **Subnet**: Choose the subnets in your VPC. To add a new subnet, please follow Add a Subnet. **(??? Should one Public or Private Subnet be selected or ALL?????**

   Note:

   1) Do not select a subnet in AWS outputs, AWS Wavelength or an AWS Local Zone when creating your cluster. After cluster creation, you can tag the AWS Wavelength with cluster name, which will then enable you to deploy self managed nodes to the subnet.

   2) **The subnets specified must be in at least two different Availability Zones**.
10. **Security groups**: Select the security group to apply to the EKS-managed Elastic Network Interfaces that are created in your worker node subnets. (Create Security group to add new)
11. Configure Kubernetes Service IP address range: (optional and can leave in disabled state).
12. **Cluster endpoint access**: Select the cluster endpoint access as per your requirement (Public, Public and Private, or Private)
13. Networking add-ons: . **Continue to Next** leave default settings as is..
14. On the Configure logging page, you can optionally choose which log types that you want to enable.
15. **Select Next** -> Review -> **Create.** Cluster provisioning usually takes between 10 -15 minutes.

Create the Managed Node (Reg Availability Zone)

Note: this is an **example of a Node created manually**

1. Open the Amazon EKS console
2. Clusters: **Choose your cluster** where you want to create a managed node group.
3. On the "Your Cluster" page
4. Select the **Configuration tab.**
5. Select the **Compute tab**, and then Choose **Add Node Group**.
6. On Configure Node Group page

7. **Name**: Assign a Unique name for your Managed Node Group
   (ie. abcManagedNodeGroup)
8. **Node IAM Role**: Select from dropdown **your NodeInstanceRole** created previously to use with node group.
   Note: The selected role must not be used by a self-managed node group as this could lead to a service interruption upon Managed Node Group deletion.
9. Launch template, Kubernetes labels , Kubernetes taints, and tags (**optional**).
10. Choose **NEXT**
11. **Set compute and scaling configuration** of the node group:
    - AMI type – Choose
      - Amazon Linux 2 (AL2_x86_64) for **non-GPU** instances,
      - Amazon Linux 2 GPU Enabled (AL2_x86_64_GPU) for GPU instances,
      - or Amazon Linux 2 (AL2_ARM_64) for Arm.
    - **Note:** For RegAZ, you can go with regular AMI whereas for wavelength zone choose
    - AMI with GPU enabled.
    - Capacity Type – Select **On-Demand** type
    - Instance Type – **Select t3.medium/t3.large;** Change as per the requirement
    - Size – **80;** Change as per requirement
    - Node group scaling – Specify the maximum and minimum number of nodes that the managed node group can scale.(**For test projects max, min, desired preferred to be 1 node**)
    - Node Group update configuration: Select Number or Percentage and input and corresponding **Value:** ie. 1 node

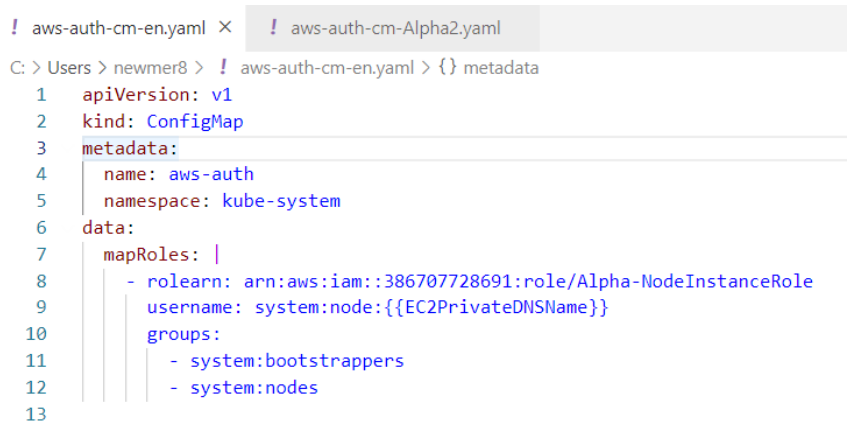
12. Choose **Next**
13. Node Group Network Configuration
14. **Subnets**:Select from dropdown **your subnets** to launch your managed nodes(**select Private Subnets**)
15. Enable "**Configure SSH access to nodes**"
    Note: Warning
    When enabling this option, managed node groups will create a security group on your behalf with port 22 inbound access. If launching your worker in a public subnet, it's strongly recommended to restrict the source IP address ranges. (**Why is another Security group created at this point?**)
16. **Choose the SSH pair**. . If you don't have one, create a new keypair on the EC2 console.

17. **Allow remote access from**, Select your preferred method
    a. **Selected security groups:** Specify security groups to restrict which source IPs can remotely access nodes.
    b. **All** : Do not restrict source IPs that can remotely access nodes.
    If you want to limit access to specific instances, then select the security groups that are associated to those instances. If you don't select specific security groups, then SSH access is allowed from anywhere on the internet
18. Security Groups: **Select the security group generated by "Configure SSH Access to Node" click the refresh button to see newly generated Security group**
    EKS created a security group applied to ENI that is attached to EKS Control Plane master nodes, as well as any managed workloads.
19. Choose: NEXT
20. **Review and Create** the node group.

21. Test SSH Connection to network (**Managed Node Connection to Cluster**)
    a. Enable Nodes to Join the Cluster instruction section below.
    b. Download the configuration map and save to your local C:drive.

curl -o aws-auth-cm.yaml
https://s3.**us-west-2**.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-auth-cm.yaml

    c. Log into your Text Editor: ie Visual Studio Code or CMD
    d. Open your created **KeyPair script** and update with ARN from your **NodeInstanceRole.**
    e. Map Roles: Goto your IAM Role and find the Managed Role ARN. Copy and paste your arn NodeInstanceRole into script.. Locate in your EKS Cluster Managed **Node Group: Node IAM Role ARN** (NodeInstanceRole)



    f. In your text editor terminal: **configure aws** with access key, secret key, region, and output.
        i. aws configure
            1. AWS Access Key ID:
            2. AWS Secret Access Key:
            3. Default region name [us-east-1]:
            4. Default output format [json]:
        ii. aws sts get-session-token --serial-number arn-of-the-mfa-device --token-code code-from-token
            1. (**arn of mfa device**-get from IAM Users Security Credentials-MFA i.e(arn:aws:iam::386707728691:mfa/jane.doe@verizon.com)
            2. (**token code** - get Verizon MFA code from your phone or other device i.e 581 031)

Response:(Temporary Bearer Token Credentials)

{
"AccessKeyId": "xxxxxxxxxxxxxx"",
"SecretAccessKey": "xxxxxxxxxxxxxxxxxx",
"SessionToken": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
"Expiration": "2022-04-20T04:21:54Z" (good for 24hrs only)
}

            3. Store your Temporary credentials in your AWS configure file in your Windows file explorer. Goto C:drive: ie(username8/ .aws/ credentials. Open file using NOTEPAD , Add [mfa] credentials and save and close file..

Example Notepad File:

[default] (note: your AWS credentials)
aws_access_key_id = xxxxxxx
aws_secret_access_key = xxxxxxxxxx

[mfa] (note: your Temporary TOKEN AWS Credentials)
aws_access_key_id = xxxxxxxxxx
aws_secret_access_key = xxxxxxxxx
aws_session_token = xxxxxxxxxxx

iii. Update EKS Configuration: (everything in blue needs to be updated with your information)
   1. aws eks update-kubeconfig --region region code --name EKSCluster name --profile mfa

g. **Run command** to apply and watch the status of the nodes and wait for them to reach the "Ready" status
   1. kubectl apply -f aws-auth-cm.yaml (your KeyPair file)

   2. Validate SSH Connect to Instances
      kubectl get nodes -o wide or kubectl get nodes --watch

```
PS C:\Users\newmer8> kubectl get nodes
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-0-138-239.ec2.internal   Ready    <none>   22h   v1.22.6-eks-7d68063
```

```
PS C:\Users\newmer8> kubectl get nodes -o wide
  OS-IMAGE         KERNEL-VERSION                CONTAINER-RUNTIME
ip-10-0-138-239.ec2.internal   Ready    <none>   21h   v1.22.6-eks-7d68063   10.0.138.239   18.232.134.68
```

```
PS C:\Users\newmer8> kubectl get nodes --watch
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-0-138-239.ec2.internal   Ready    <none>   21h   v1.22.6-eks-7d68063
ip-10-0-138-239.ec2.internal   Ready    <none>   21h   v1.22.6-eks-7d68063
ip-10-0-138-239.ec2.internal   Ready    <none>   21h   v1.22.6-eks-7d68063
```

   3. If you are not able to connect try the following commands to see if you have authority:
      a. aws sts get-caller-identity (returns current UserID, Account, Arn)

   4. Add NVIDIA plugin for Kubernetes as a DaemonSet on your cluster. Required for GPU instance type because of the Amazon EKS optimized accelerated AMI.

      kubectl apply -f
      https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.6.0/nvidia-device-plugin.yml

```
PS C:\Users\newmer8> kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.6.0/nvidia-device-plugin.yml
Warning: spec.template.metadata.annotations[scheduler.alpha.kubernetes.io/critical-pod]: non-functional in v1.16+; use the "prior
ityClassName" field instead
daemonset.apps/nvidia-device-plugin-daemonset created
```

   5. kubectl get ds -n kube-system

```
PS C:\Users\newmer8>  kubectl get ds -n kube-system
NAME                             DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
aws-node                         1         1         1       1            1           <none>          22h
kube-proxy                       1         1         1       1            1           <none>          22h
nvidia-device-plugin-daemonset   1         1         1       1            1           <none>          2m47s
```

   6. aws configure list (displays your configuration settings)

```
PS C:\Users\newmer8> aws configure list
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************T5MA shared-credentials-file
    region                us-east-1      config-file    ~/.aws/config
```

Create the Self-Managed Node (Wavelength Zone)

Note: this is an example of a Node created using CloudFormation Template

Prerequisites:

An existing Amazon EKS cluster that uses a VPC and security group that meet the requirements of an Amazon EKS cluster. (NOTE: it is recommended that you create your Subnet and Carrier Gateway prior to creating Cloudformation Stack. If you do not create them before starting Stack, you will be required to restart stack creation.)

1. Open the **Cloudformation** console.
2. Choose **Create Stack**.
3. Prerequisite - **Prepare template**: Choose your desired template
4. Specify Template Source: Amazon S3 URL or Upload a template file
5. If Amazon S3 URL:
6. Select the below Amazon S3 URL and paste into the URL section
   https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-nodegroup.yaml
7. Choose **NEXT**
8. Specify Stack Details
   a. **Stack name**: Enter Unique Name (ie abc-SelfManagedNode-Wavelength)
   b. **EKS Cluster:** Enter your EKS Cluster name previously created
   c. Provide the **ClusterControlPlaneSecurityGroup**.
      i. Select the **Security group** that is generated while creating the cluster. You can get this information from cluster -> configuration -> Networking -> Additional Security Groups
      ii. You can check the control plane security group for your cluster in the AWS Management Console under the cluster's **Networking** section (listed as **Additional security groups**), or with the following AWS CLI command in your terminal: aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.securityGroupIds
   d. Provide the **Node Group name(should be the same node group created for Reg Node)**.
   e. Node group scaling – Specify the maximum and minimum number of nodes that the managed node group can scale.
   f. **NodeInstance** Type – **g4dn.2xlarge**
   g. **NodeImageIdSSMParam** - /aws/service/eks/optimized-ami/1.18/amazon-linux-2-gpu/recommended/image_id
   h. NodeImageId: Optional
   i. NodeVolume Size: For Wavelength, **80 – 100 GB** (Min 80GB)
   j. Provide the **KeyName**
   k. **Bootstrap Arguments**: Specify any optional arguments to pass to the node bootstrap script, such as extra kubelet arguments.
      i. Copy and paste the link below: --apiserver-endpoint <cluster-endpoint> --b64-cluster-ca <cluster-certificate-authority>

ii. replace <cluster-endpoint> and <cluster-certificate-authority with details from your Cluster Configuration Details tab.
iii. **Goto: Cluster Configuration Details Tab .** Copy Certificate Authority first and paste then copy API server endpoint and paste
iv. **Replace cluster endpoint and cluster certificate authority** (You will find this information in **cluster details**)

l. Select **False** for DisableIMDSv1
Note: Each node supports the Instance Metadata Service Version 1 (IMDSv1) and IMDSv2 by default, but you can disable IMDSv1. Select true if you don't want any nodes in the node group, or any pods scheduled on the nodes in the node group to use IMDSv1.

m. Enter the **VPCID** for the VPC that you created.
n. Choose your **Wavelength subnet(s)**.
i. IF you already have a wavelength subnet select it, otherwise,
ii. Create a Wavelength subnet. (IF you have to create subnet at this point, you will have to restart Cloudformation Stack Creation to associate subnet created)

1. **Create a Wavelength subnet with Wavelength AZ**. (i.e. us-east-1-**wl1**-bos-wlz-1)
a. Goto VPC console, select Subnet.
b. Create a Subnet. Select your VPC previously created.
c. Define Wavelength subnet name(make sure to include word **wavelength** in name)
d. Select AZ(Note: Wavelength AZs are defined with "**-wl1** extension"
e. Input IPv4 CIDR block based on the range selected for your previously created subnets associated with your VPC. (ie. If the non-wavelength subnet IPv4 CIDR is 10.0.128.0/20, you must define your new wavelength subnet within the range (ie. 10.0.192.0**/20**).
f. Select **Create Subnet**.(record the wavelength subnet number for reference)
Note: To route subnet traffic to a carrier network in a Wavelength Zone, a carrier gateway in VPC is required. Create a carrier gateway.

2. Carrier Gateway go to:Create Carrier Gateway:
a. **Name:** Create Name tag (ie. abc-cagw-wl1)
b. **VPC**: Attach carrier gateway to your VPC
c. Route Subnet traffic: Checkbox only if**:**
i. IF you have **NOT** already created a Wavelength subnet, you must **Create One now**; otherwise go to the next step (ii).
ii. Select Check Mark to: Route subnet traffic to the carrier gateway. Define unique subnet name>select wavelength zon> Ipv4 CIDR block
iii. Otherwise leave blank and you can manually add route association after you create the carrier gateway.
d. Tags: Optional
e. **Choose**: Create Carrier Gateway

9. Choose **NEXT**:
10. Configure Stack Options> Tag (optional)>Permissions(optional)>
Stack failure options > Choose **Roll Back** or Preserve > Select Def
Advanced options(SKIP -use default
11. Choose NEXT >: REVIEW
12. **Check**: Acknowledge that the stack might create IAM resources, and then choose Create stack.> **Create Stack**

13. When your stack has finished creating, select it in the console and choose **Outputs**.
14. Record the **NodeInstanceRole** for the node group that was created. You need this when you configure your Amazon EKS nodes.

Join Nodes to Your EKSCluster:

Note: Once the instance is created in a wavelength zone, make sure it has the internet access by attaching the carrier IP to the instance. Assign the cluster security group to the instance so the kubectl can talk to the pods in the node.

1. Use the command that corresponds to the **Region that your cluster is in** to download the configuration map.

   curl -o aws-auth-cm.yaml
   https://s3.**us-west-2**.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-auth-cm.yaml

2. This command **Opens the file with your text editor (ie Visual Studio Code)**
3. Map Roles: Goto your IAM Role and find the SelfManaged Role ARN
4. Replace the <ARN of **NodInstanceRole** (not instance profile)> snippet with the **NodeInstanceRole**. NOTE: **IF** there are two nodes and roles are different for both the nodes then **Both ARN's must be added.**

   a.

C: > Users > newmer8 > ! aws-auth-cm-en.yaml > {} data > mapRoles

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: arn:aws:iam::386707728691:role/Alpha-NodeInstanceRole
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws:iam::386707728691:role/Beta-SelfManagedNode-Wavelength-NodeInstanceRole-179JD0U5YU0HN
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

5. Get MFA Security Token(Verizon requires MFA to access AWS CLI in VSCode Terminal):(Note: everything in **blue** must be updated with your information)
   a. input: **aws sts get-session-token --serial-number arn-of-the-mfa-device --token-code code-from-token**
      i. (**arn of mfa device**-get from IAM Users Security Credentials- MFA i.e(arn:aws:iam::386707728691:mfa/jane.doe@verizon.com)
      ii. (**token code** - get Verizon MFA code from your phone or other device i.e 581 031)
   b. Update "your [mfa] credentials in your .aws/ credentials file on your C:Drive and save..
   c. Update Configuration Input: **aws eks update-kubeconfig --region region code --name EKSCluster name --profile mfa**
6. Apply the configuration. This command may take a few minutes to finish.
   input: **kubectl apply -f aws-auth-cm.yaml**
7. Watch the status of your nodes and wait for them to reach the Ready status.
   a. input: **kubectl get nodes**
   b. kubectl get nodes -- watch
8. (**Note: Only add if not previously added during Regular Node Creation**)

verizon

Add NVIDIA plugin for Kubernetes as a DaemonSet on your cluster. Required for GPU instance type because of the Amazon EKS optimized accelerated AMI.

a. input command:

kubectl apply -f
https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.6.0/nvidia-device-plugin.yml

9. Validate SSH Connect to Instances

Test SSH Connection:

Wavelength (Carrier Gateway / with Private Subnet)

1. Since the Carrier Gateway is private and is connected via a Private Subnet a Test Connection to an EndPoint EC2 needs to be created.
   a. **Create** a **BastionHOST EC2** endpoint
   b. Goto and Select EC2
   c. Select Instances and Launch an EC2 Instance
   d. **Name**: your label-Bastion HOST (i.e. Beta-BastionHOST)
   e. **Select** AMI - ie. Amazon Linux
   f. **Instance Type**: Default t2.micro
   g. **Keypair**: select your KeyPair from dropdown
   h. **Network Setting**:
      1. Edit VPC: Select your EKS VPC
      2. Default - Allow SSH Traffic from Anywhere 0.0.0.0/0
   i. **Configure Storage** : Default no selection necessary
   j. Launch Instance

2. Goto EC2 Instances and Find your newly created BastionHOST (wait for status check of "Running")
3. **Select InstanceID**
4. **Select Connect** located in the top right corner of screen
5. Select **EC2 Instance Connect** TAB
6. **Select Orange Connect button**
   a. Connection to TEST BastionHOST validated:

```
Last login: Wed Apr 20 22:20:56 2022 from ec2-18-206-107-26.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-0-14-100 ~]$
```

7. Input : **sudo su -** to access bastion host as root user
   a.

```
Last login: Wed Apr 20 22:20:56 2022 from ec2-18-206-107-26.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-0-14-100 ~]$ sudo su -
[root@ip-10-0-14-100 ~]#
```

8. Get Your KeyPair details:
   a. In your VSCode terminal input (**cat** "yourKeyPair.pem")

b. **Copy** the KeyPair details generated in terminal

9. Go back to BastionHost Terminal
   a. Input (**vim** "yourKeyPair.pem")
   b. input (" i ") in the BastionHost terminal to open VIM
   c. Go back to VSCode Terminal to Copy yourKeyPair details –all details beginning to ending
   d. Paste into BastionHost terminal
   e. Enter " **:wq"** to exit vim.
   f. Enter "chmod 400 "yourKeyPair.pem"
   g. Enter the SelfManageNode EC2 SSH Client info into your BastionHost Terminal::
      1. (ie ssh -i "yourKeyPair.pem"**root@**10.0.202.83)(remove the word "**root@**")
      2. Select enter and Answer "YES" To Connect
      3.

```
[ec2-user@ip-10-0-14-100 ~]$ ssh -i "Erica-MasterKeyPair.pem" 10.0.202.83
The authenticity of host '10.0.202.83 (10.0.202.83)' can't be established.
ECDSA key fingerprint is SHA256:6100xtj3KOrXuI7sc5bkG6t0FVujoFK1gM0QL62lJiE.
ECDSA key fingerprint is MD5:b5:e2:22:5f:e9:0d:88:5c:b5:de:72:05:d7:1f:35:f2.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.202.83' (ECDSA) to the list of known hosts.
Last login: Wed Apr  6 00:56:41 2022 from 205.251.233.52

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-202-83 ~]$
```

10. Test Kubelet status
    a. Input : **sudo su -** to access bastion host as root user
    b.

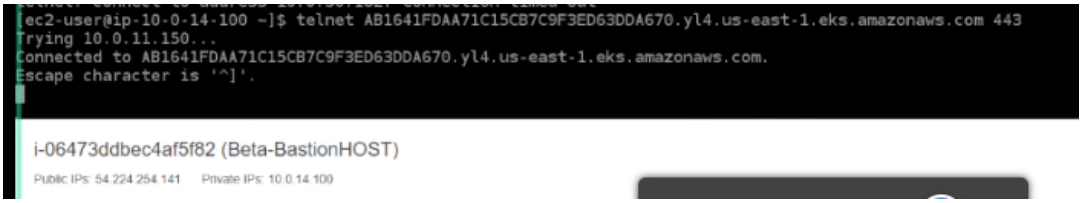
```
Last login: Wed Apr 20 22:20:56 2022 from ec2-18-206-107-26.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-0-14-100 ~]$ sudo su -
[root@ip-10-0-14-100 ~]#
```

    c. Input: **systemctl status kublet**
       1. If kublet does not run properly input : (to identify error)
       2. try pinging a web page: input : **ping google.com**
       3. if still not working check your VPC Endpoints **journal -fu kubelet**

11. **To Connect your SelfMangedNODE directly to BastionHost**
    a. Create a BastionHost Role with Admin Access
       1. Goto IAM and create BastionHostRole for your EC2 - Add Administrator Access
    b. Goto your BastionHost EC2
       1. Select Actions , Security, Modify IAMRole
       2. select BastionHostRole , SAVE
    c. Goto BastionHost Terminal to validate BastionHost is able to Reach Endpoint of SelfMangedNode EC2
    d. Input: aws sts get-caller-identity to verify your access to kublet
    e. Input: sudo yum install telnet -y to install telnet (no need to install if you have already done so)

verizon

f. Input: telnet AB1641FDAA71C15CB7C9F3ED63DDA670.yl4.us-east-1.eks.amazonaws.com 443
g. If the connection fails, check the security group of your EKS Cluster.
h. Goto the **your EKSCluster** and get your Additional Security Group id
    1. **Edit Inbound Rules** for **your EKSCluster , Networking,** additional Security Group id " **sg-0a1c412736c325754"**
    2. input: HTTPS
    3. CUSTOM: input **yourBastionHostEC2** Security Group "sg-0bd51e8cc50958f00" (this will enable inbound communication between yourEKSCluster and the yourBastionHostEC2 in the bastion host terminal)
    4. SAVE
i. Goto BastionHost Terminal to test connection again
    1. Input: telnet AB1641FDAA71C15CB7C9F3ED63DDA670.yl4.us-east-1.eks.amazonaws.com 443
    2. Validates connection made to internet



Join Carrier Gateway IP to EC2 instance in Wavelength Zone

12. Allocate and associate a Carrier IP address with the instance in the Wavelength Zone:
    a. prerequisites: must have already provisioned **a VPC, subnets, EC2 instances resources** prior to allocation and association
13. Goto VSCode Terminal
    a. (make sure you have already updated MFA authorization updated kube configuration) (make sure you are using AWSCLIv2 or commands may not work)
14. Input Describe command to identify profile: (verify your wavelength zone)
    a. aws ec2 describe-availability-zones --profile mfa --region us-east-1

```
PS C:\Users\newmer8> aws ec2 describe-availability-zones --profile mfa --region us-east-1
{
    "AvailabilityZones": [
        {
            "State": "available",
            "Messages": [],
            "RegionName": "us-east-1",
            "ZoneName": "us-east-1a",
            "ZoneId": "use1-az6"
        },
```

    b.

```
        {
            "State": "available",
            "Messages": [],
            "RegionName": "us-east-1",
            "ZoneName": "us-east-1-wl1-bos-wlz-1",
            "ZoneId": "use1-wl1-bos-wlz1"
        },
```

    c.
15. Get Allocation Id :
    a. In the VSCode terminal:

b. Input: aws ec2 --region us-east-1 allocate-address --domain vpc --profile mfa (**AWSCLIv1**only)

```
PS C:\Users\newmer8> aws ec2 --region us-east-1 allocate-address --domain vpc --profile mfa
{
    "PublicIp": "54.164.200.40",
    "AllocationId": "eipalloc-02615fd1fe30dcf45",
    "PublicIpv4Pool": "amazon",
    "Domain": "vpc"
}
PS C:\Users\newmer8>
```

c. Or, Input: aws ec2 --region us-east-1 allocate-address --domain vpc --network-border-group us-east-1-wl1-bos-wlz-1 --profile mfa (**AWSCLIv2** only) provides Allocation Id and notice CarrierIP Association

```
C:\Users\newmer8>aws ec2 --region us-east-1 allocate-address --domain vpc --network-border-group us-east-1-wl1-bos-wlz-1 --profile mfa
{
    "AllocationId": "eipalloc-08d239524a501e04c",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-east-1-wl1-bos-wlz-1",
    "Domain": "vpc",
    "CarrierIp": "155.146.12.49"
}
```

d. Allocation Id: "**eipalloc-02615fd1fe30dcf45**"

1. **Note: IF** error msg or no access message your Windows environment is not reading your AWSCLIv2 version. It is defaulted to your AWSCLIv1. Apply Work Around - Otherwise Go to Next Step
2. **Work Around**:
    1. Open your CMD Prompt.
    2. As a temporary fix, set your path to read AWSCLIv2.
3.

```
C:\Users\newmer8>aws --version
aws-cli/1.16.81 Python/3.6.0 Windows/10 botocore/1.12.71

C:\Users\newmer8>where aws
C:\Users\newmer8\aws
C:\Program Files\Amazon\AWSCLI\bin\aws.cmd
C:\Program Files\Amazon\AWSCLIV2\aws.exe

C:\Users\newmer8>set PATH="C:\Program Files\Amazon\AWSCLIV2\";%PATH%
```

4. Find out where AWS is on your Windows machine.
   Input command: **where aws**
5. Change path in CMD Prompt
   input : **set PATH-"C:\Program Files\Amazon\AWSCLIV2\aws.exe";%PATH%**

16. Associate the Carrier IP address with the EKS SelfManaged Node EC2 instance(update blue highlighted items with your information)
    a. Input: aws ec2 associate-address --allocation-id eipalloc-02615fd1fe30dcf45 --network-interface-id eni-0f22f0c2f416fd63d -- profile mfa --region us-east-1
        1. allocation id: (**get from your allocation previously ran above**)
        2. network interface id:i - #**(get from SelfMangedNode EC2 - Network Tab-Network Interfaces**)
        3.

```
C:\Users\newmer8>aws ec2 associate-address --allocation-id eipalloc-08d239524a501e04c --network-interface-id eni-0f22f0c2f416fd63d --region us-east-1 --profile mfa
{
    "AssociationId": "eipassoc-0461be8ed2cfc9c08"
}
```

    b. Result is the Association Id: eipassoc-0461be8ed2cfc9c08 which connects and allows communication between your Carrier Gateway IP to your SelfManagedNode EC2

17. Test Communication in your EC2 Bastion over the internet
    a. Open EC2 Bastion Host
        1. Select EC2 Instance and Connect
    b. Open your SelfManagedNode EC2
        1. Go to SSH Client tab
        2. Copy and paste into Bastion Host command line

    SSH Keypair.pem (ie ssh -i "yourKeyPair.pem"**root**@10.0.202.83)(**replace the word "root" with (ec2-user**)
    c. Paste into BastionHOST command line

d. Input : curl http://www.amazon.com (internet connection validated below)

```
Last login: Mon Apr 25 23:27:37 2022 from ec2-18-206-107-24.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-14-100 ~]$ ssh -i "Erica-MasterKeyPair.pem" ec2-user@10.0.202.83
Last login: Mon Apr 25 18:46:54 2022 from ip-10-0-14-100.ec2.internal

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-202-83 ~]$ curl http://www.amazon.com
<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>CloudFront</center>
</body>
</html>
```

18. Or, Ping Google.com Input: ping google.com

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-202-83 ~]$ ping google.com
PING google.com (172.217.2.110) 56(84) bytes of data.
64 bytes from iad23s72-in-f14.1e100.net (172.217.2.110): icmp_seq=1 ttl=252 time=13.3 ms
64 bytes from iad23s72-in-f14.1e100.net (172.217.2.110): icmp_seq=2 ttl=252 time=13.3 ms
64 bytes from iad23s72-in-f14.1e100.net (172.217.2.110): icmp_seq=3 ttl=252 time=13.3 ms
```

Persistent Storage:

To set up the persistent storage in Amazon EKS we can use EBS CSI driver or EFS CSI driver. This allows the cluster to dynamically provision EBS volumes. Documentation to install EBS CSI driver plugin - https://docs.aws.amazon.com/eks/latest/userguide/ebs-csi.html

The Amazon EBS CSI driver isn't installed when you first create a cluster. To use the driver, you must add it as an Amazon EKS add-on or as a self-managed add-on.

For instructions on how to add it as an Amazon EKS add-on, see Managing the Amazon EBS CSI driver as an Amazon EKS add-on.

For instructions on how to add it as a self-managed add-on, see Managing the Amazon EBS CSI self-managed add-on.

Amazon EBS CSI driver

1. Create an IAM OIDC(OpenID Connect issuer URL) provider for your cluster
   A. To create an IAM OIDC identity provider for your cluster with the **AWS Management Console**
   B. Open the Amazon EKS console at https://console.aws.amazon.com/eks/home#/clusters .
   C. **Select** the name of **your cluster** and then select the **Configuration tab**.
   D. In the **Details sectio**n, note the value of the **OpenID Connect provider URL**.
   E. Copy:
      a. **OpenID Connect provider URL**
      https://oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
      b. Note you may also view via Command Line using:
      aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --output text --region us-east-1 --profile mfa
   F. Open the IAM console at https://console.aws.amazon.com/iam/ .
   G. In the left navigation pane, choose Identity Providers under Access management.
   If a Provider is listed that matches the URL for your cluster, then you already have a provider for your cluster.
   If one is NOT listed and matches the URL for your cluster, then you must **create one**.
   H. **To create a provide**r, choose **Add Provider**.
      a. For Provider Type, choose **OpenID Connec**t.
      b. For Provider URL, **paste the OIDC issuer URL for your cluster**, and then choose Get thumbprint.
      c. For Audience, enter sts.amazonaws.com and choose Add provider.
   I. Note: you may also create and associate Provider to your EKSCluster via Command Line using:
   input: eksctl utils associate-iam-oidc-provider --cluster your-cluster --approve

2. Create the Amazon **EBS CSI driver** IAM role for service accounts (click link for instructions)
    a. Prerequisites:
        i. An existing cluster that's version 1.18 or later.
        ii. An existing (IAM) OpenID Connect (OIDC) provider for your cluster.

3. Manage the Amazon EBS CSI driver as an **Amazon EKS** add-on (click link for instructions)
    a. Prerequisites
        i. An existing cluster that's version 1.18 or later. To see the required platform version, run the following command.
        aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
        ii. An existing (IAM) OpenID Connect (OIDC) provider for your cluster.
        iii. An Amazon EBS CSI driver IAM role.
4.

Load Balancing:

Note: Wavelength Zone doesn't support Network load balancing whereas it allows application load balancing.

Below are the Troubleshootings and findings we found while deploying AIFI application:

1. Loadbalancer for **nginx** ingress controller is not being created.
   First, try to check the cloud trail for the logs and see the error message why it is failed. We noticed that CreateLoadBalancer call has failed with below error messages.

"eventTime": "2021-01-22T20:52:06Z",
"eventSource": "elasticloadbalancing.amazonaws.com",
"eventName": "CreateLoadBalancer",
"awsRegion": "us-east-1",
"sourceIPAddress": "eks.amazonaws.com",
"userAgent": "eks.amazonaws.com",
"errorCode": "AccessDenied",
"errorMessage": "An unknown error occurred",

However, later on we noticed that a new load balancer was created once we tried to create NLB manually. Check the target groups of the loadbalancer if they are healthy. If the status is an unused state try adding the annotation to the subnets so the kubernetes service will pick up the subnets.

SSH Connection and Ping :
(Public Node):

1. Open your Eks Cluster Under overview tab you will see your Nodes
2. Click on your **public Node**
3. Then Click on Instance (It should take you to the Instance summary page)
4. Then Upper right hand corner Select Connect tab.
5. Then copy SSH Path (Example ssh -i "mk-keypair1.pem" root@54.81.150.186) > Go to you VS Code > Paste SSH Path > Do not Press Enter > Change root to ec2-user > Then press Enter. (FYI–Before Pressing Enter, make sure you are disconnected from VPN).

6. Once you have Successful connection now you want to Test ping > Type ping 8.8.8.8 > Press Enter > you will see multiple icmp connections that means you are able to successfully connect.

Keypoints for the cluster:

1. Make sure to assign the cluster security group to the instance so the kubectl can talk to the pods in the node.
2. For PVC's, CSI driver plugin must be installed.
3. Annotations must be added for the subnets so that kubernetes service can pick up the subnets and create the load balancer.
   https://aws.amazon.com/premiumsupport/knowledge-center/eks-load-balancers-troubleshooting/
4. Make sure the wavelength zone instance has kubernetes.io/cluster/aifi-eks- cluster tag to only cluster security groups.

   Code dive for load balancer:

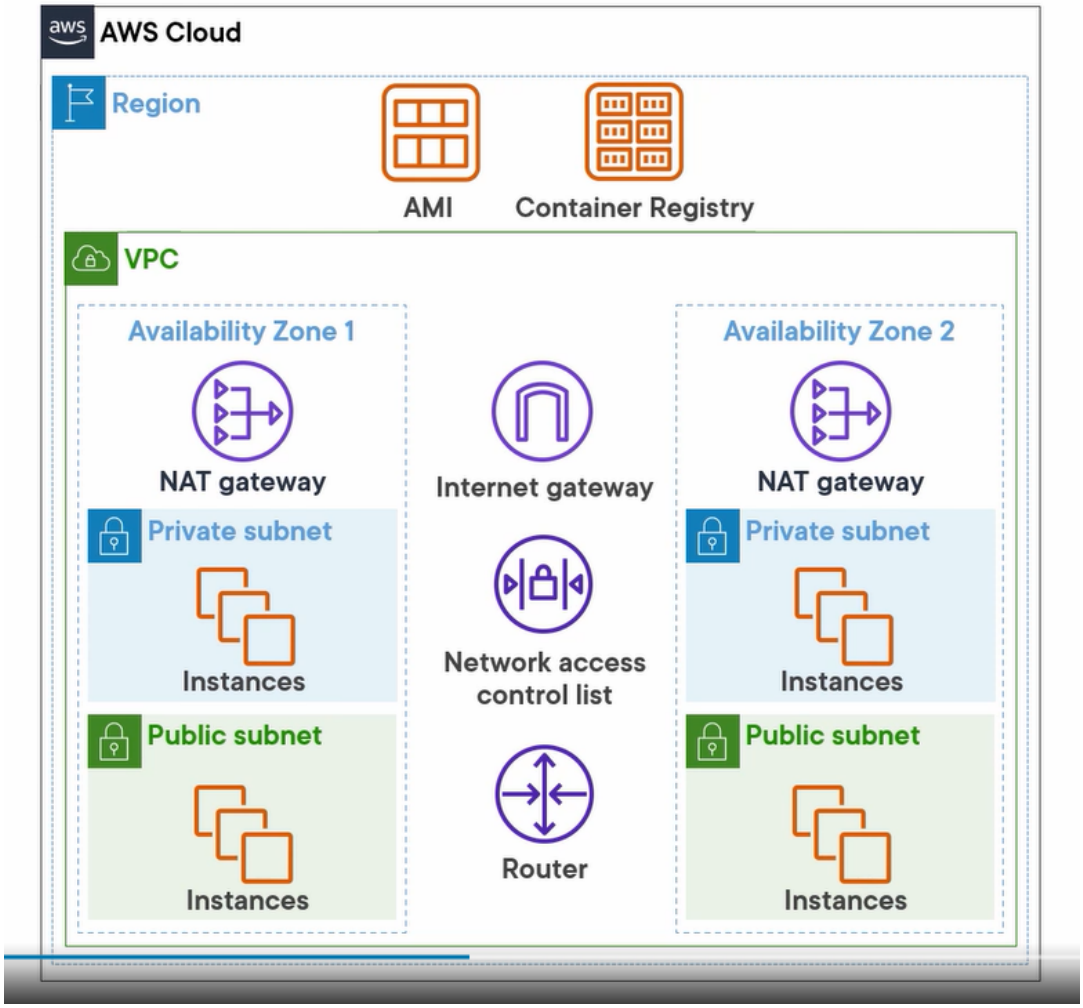
   function ensure load balancer
   - creates load balancer
   - creates target group
   - creates listeners
   - updates instance sg using function updateInstanceSecurityGroupsForLoadBalancer
   - Will get the security groups attached to the worker node instances
   - This functions responsibility is to update the node SG to allow Loadbalancer to communicate with the node
   - function findSecurityGroupForInstance() is called
   - its is checking if the SG has Cluster tag which is "key: kubernetes.io/cluster/aifi-eks-cluster Value: owned or shared."
   - It expects that this tag is only present for one SG. Since we had SG's with the same tag we ran into the current issue which has halted the workflow.

   - Ideally if updateInstanceSecurityGroupsForLoadBalancer is completed loadbalancer status will be updated and the DNS name of the Loadbalancer will be available when we run "kubectl get svc -n <namespace>"

Troubleshooting:

1. If nodes fail to join the cluster, use the below link for troubleshooting:
   https://docs.aws.amazon.com/eks/latest/userguide/troubleshooting.html#worker-node-fail
2. To change the status of the nodes from Not Ready or Unknown to Ready
   https://aws.amazon.com/premiumsupport/knowledge-center/eks-node-status-ready/

Cloud Infrastructure

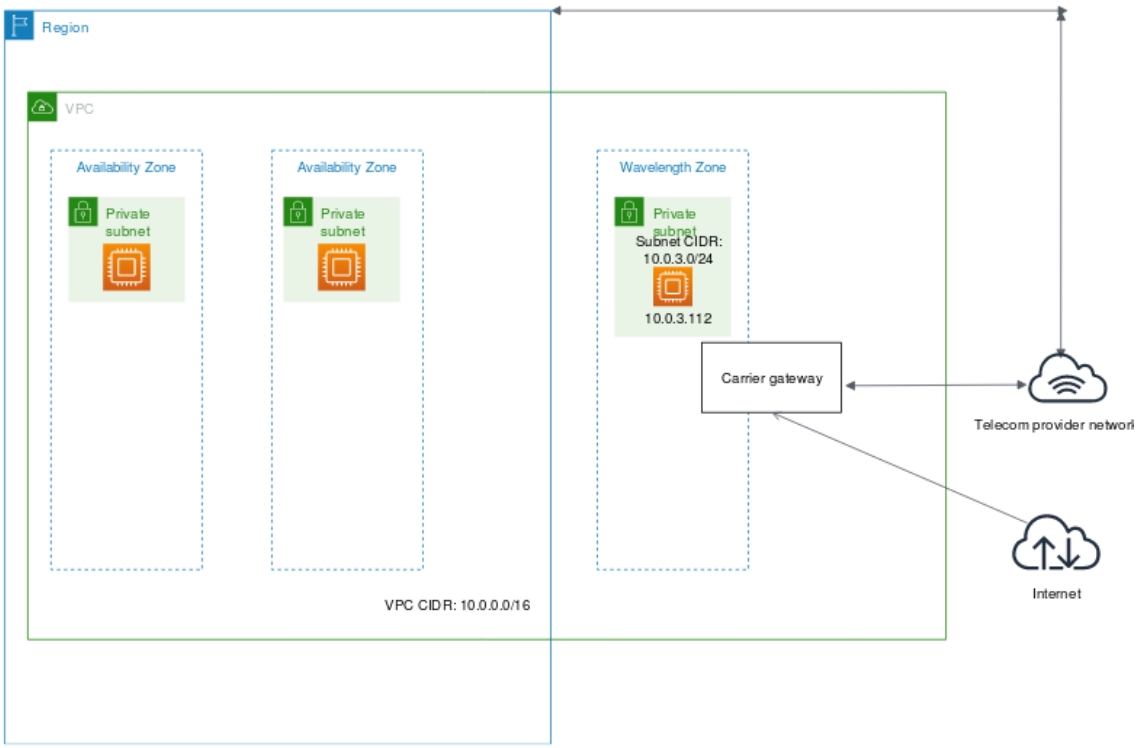


How AWS Wavelength works

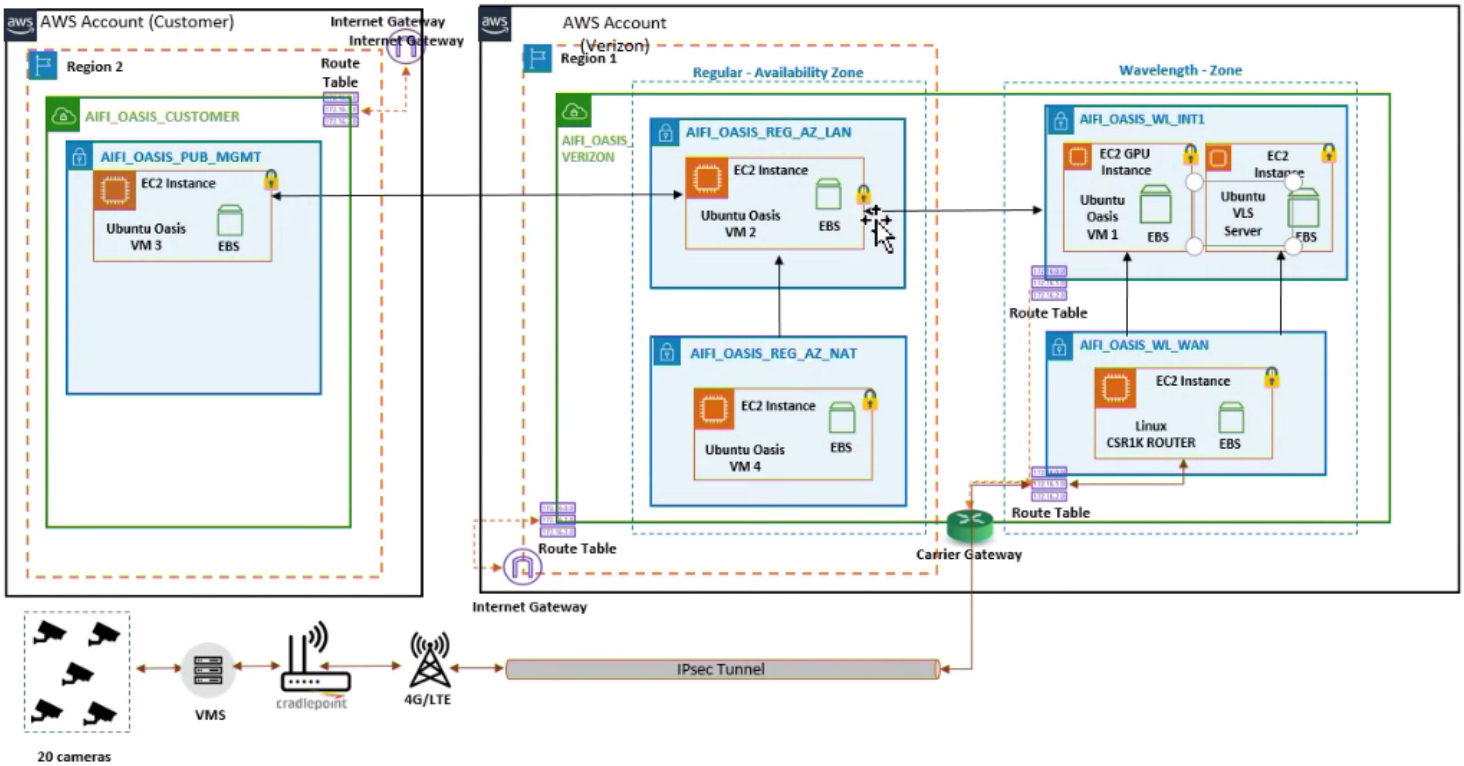
PDF | RSS

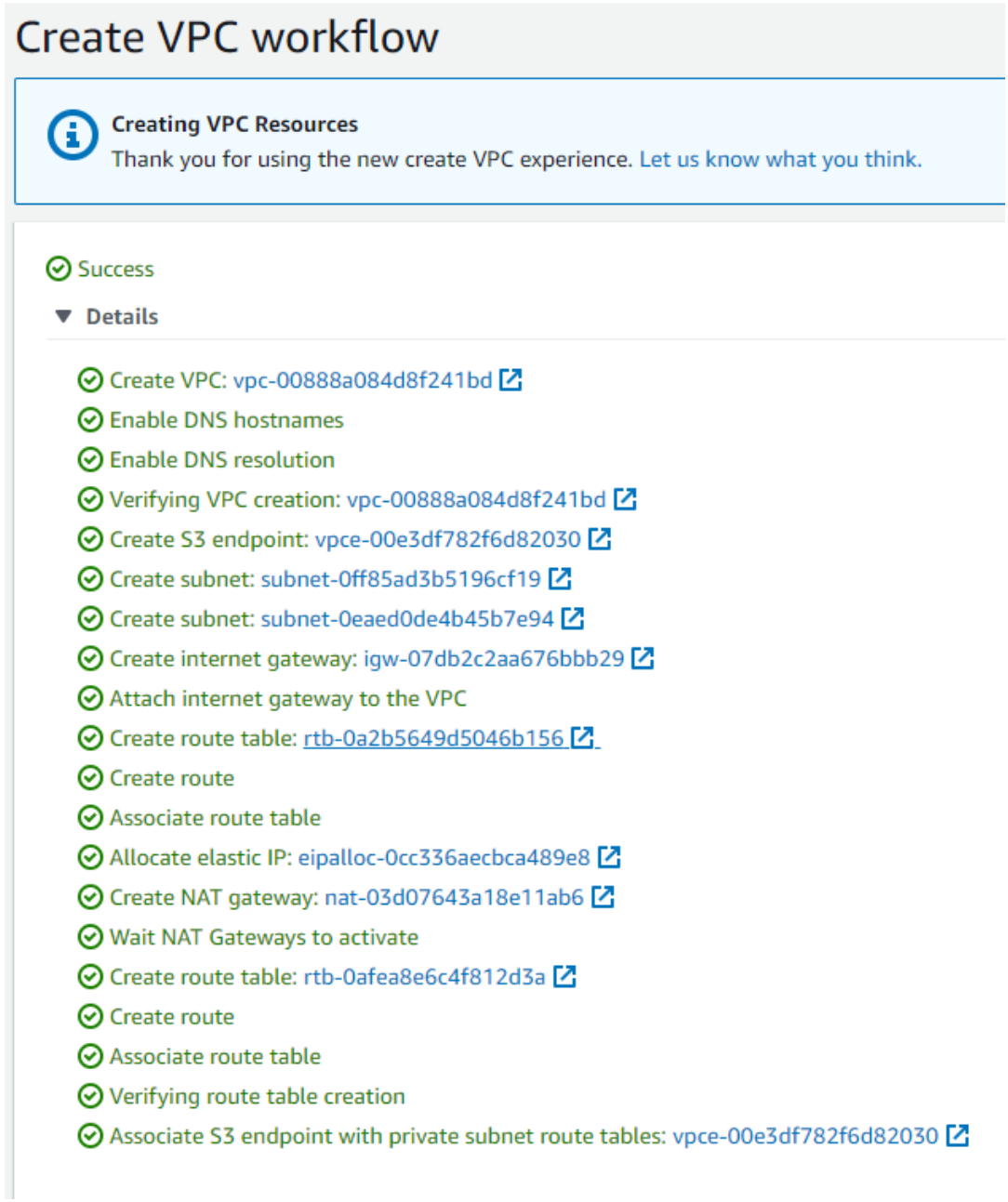
The following diagram demonstrates how you can create a subnet that uses resources in a telecommunication carrier network at a specific location. You create a VPC in the Region. For resources that need to be within the telecommunication carrier network, you opt in to the Wavelength Zone, and then create resources in the Wavelength Zone.

verizon

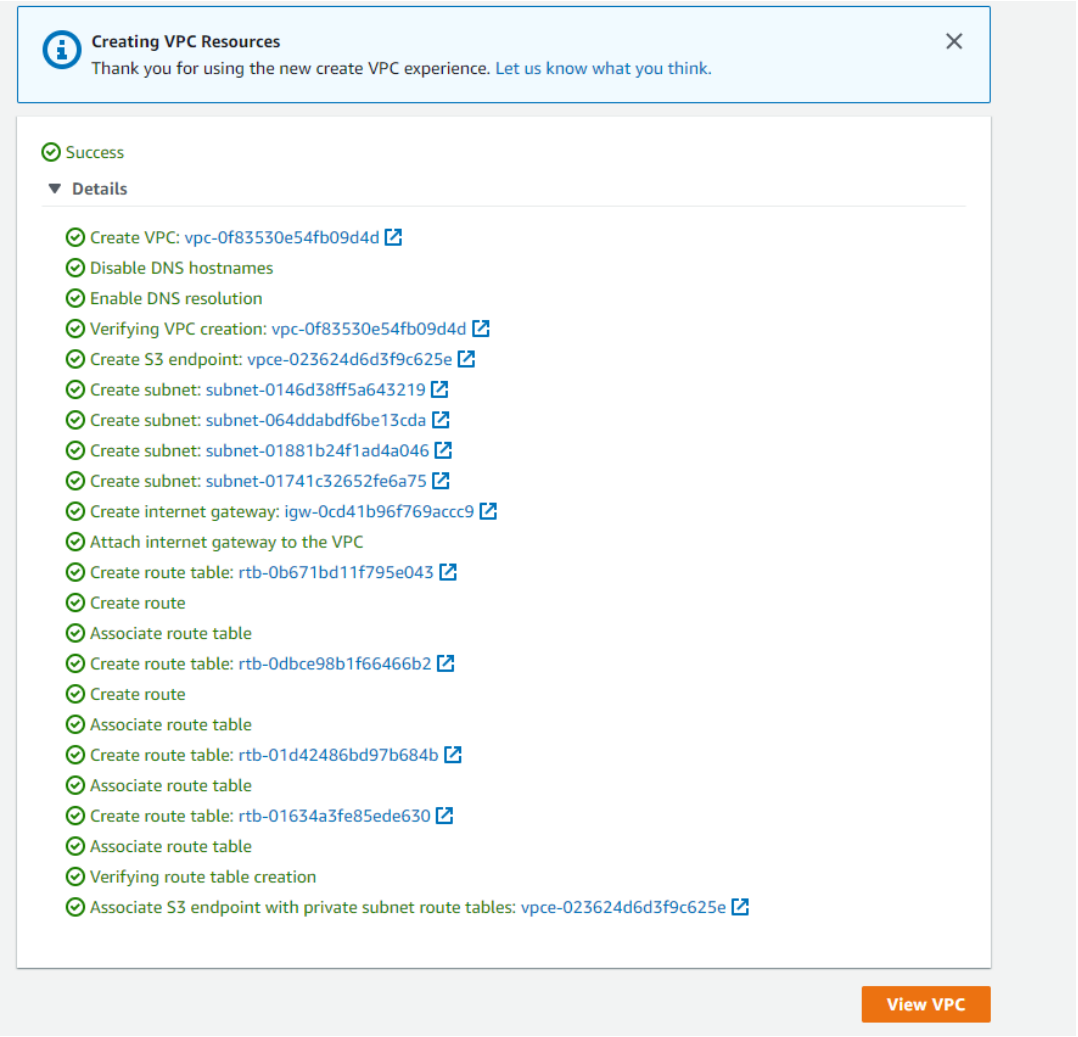


VZ AiFi Touchless payment Network Architecture




Create VPC workflow
Creating VPC Resources
Thank you for using the new create VPC experience. Let us know what you think.
Success
Details
Create VPC: vpc-00888a084d8f241bd
Enable DNS hostnames
Enable DNS resolution
Verifying VPC creation: vpc-00888a084d8f241bd
Create S3 endpoint: vpce-00e3df782f6d82030
Create subnet: subnet-0ff85ad3b5196cf19
Create subnet: subnet-0eaed0de4b45b7e94
Create internet gateway: igw-07db2c2aa676bbb29
Attach internet gateway to the VPC
Create route table: rtb-0a2b5649d5046b156
Create route
Associate route table
Allocate elastic IP: eipalloc-0cc336aecbca489e8
Create NAT gateway: nat-03d07643a18e11ab6
Wait NAT Gateways to activate
Create route table: rtb-0afea8e6c4f812d3a
Create route
Associate route table
Verifying route table creation
Associate S3 endpoint with private subnet route tables: vpce-00e3df782f6d82030

My2nd vpc


Creating VPC Resources
Thank you for using the new create VPC experience. Let us know what you think.
Success
Details
Create VPC: vpc-0f83530e54fb09d4d
Disable DNS hostnames
Enable DNS resolution
Verifying VPC creation: vpc-0f83530e54fb09d4d
Create S3 endpoint: vpce-023624d6d3f9c625e
Create subnet: subnet-0146d38ff5a643219
Create subnet: subnet-064ddabdf6be13cda
Create subnet: subnet-01881b24f1ad4a046
Create subnet: subnet-01741c32652fe6a75
Create internet gateway: igw-0cd41b96f769accc9
Attach internet gateway to the VPC
Create route table: rtb-0b671bd11f795e043
Create route
Associate route table
Create route table: rtb-0dbce98b1f66466b2
Create route
Associate route table
Create route table: rtb-01d42486bd97b684b
Associate route table
Create route table: rtb-01634a3fe85ede630
Associate route table
Verifying route table creation
Associate S3 endpoint with private subnet route tables: vpce-023624d6d3f9c625e
View VPC