

Tony Shen

AWS EKS

Contents

Introduction	3
Overview	3
Environment and naming convention	4
Preparing to Create EKS Cluster.....	5
Creating EKS Cluster.....	13
Running Apps in EKS Cluster	23
Namespace in Cluster	23
Pod details.....	25
Accessing Pod.....	29
Cluster networking.....	30
Removing EKS Cluster	30
Summary	31
References	42

Revision History
Revision v1, December, 2020
Revision v2, February, 2021

Glossary

Term	Description
AWS	Amazon Web Services
Kubernetes	An open-source system for automating deployment, scaling, and management of containerized applications
K8	Abbreviation for Kubernetes
Control Plane	K8 cluster management software system that manages nodes and pods
Node	A Node is a worker machine in Kubernetes and may be either a virtual or a physical machine, depending on the cluster. Also referred to as work nodes. K8 cluster has at least one node. Nodes run containerized application workloads. Nodes are managed by Control Plane
Control Plane components	There are five (5) control plane components, i.e., Cluster Controller (c-m), Cloud Controller Manager (c-c-m), api / api server, sched (Scheduler), etcd (agent that communicates with nodes), and node controller
Node components	There are three node components, i.e., kubelet, kube-proxy, container runtime
Addons	DNS, Web UI (dashboard), container resource monitoring, cluster-level logging, where DNS is required, and others are optional
Pod	Logical host that runs on a node. Logical resource grouping that runs containerized workloads on a node. One node can run multiple pods when the node capacity allows. When a node fails, K8 control plane recreates pods on surviving nodes.
Container runtime	Docker, containerd, CRI-O, and any implementation of the Kubernetes CRI (Container Runtime Interface)
Job	Job is a Kubernetes resource that runs a <u>Pod</u> , or perhaps several Pods, to carry out a task and then stop.
Cluster state	Current state or desired state
K8 objects	Persistent entities in K8 system. K8 uses these entities to represent the state of your cluster. Objects are expressed in YAML format
Kubernetes API	Based on RESTful API
RESTful API	
Prometheus	an open-source systems monitoring and alerting toolkit
CNCF	Cloud Native Computing Foundation

Introduction

Kubernetes is a software system that manages and orchestrates Docker containerized apps to run in a fairly comprehensive frame that includes a cluster taking care of workloads with auto-scaling and load balancing capabilities built-in, and a control plane that manages the cluster. Kubernetes, or simply K8 for short, integrates well with container development systems such as Docker Community Edition (CE) or Docker Enterprise Edition (EE). In addition, it fully supports popular software repos and registries such as Docker Hub, Git/GitHub/GitLab/GitOps for distributing and version controlling docker images and source code. K8 is a power tool for organizations large and small to use in pursuing DevOps/DevSecOps in seek of greater IT efficiency, cost-effectiveness, security, and agility.

AWS offers K8 as a service in AWS cloud, which is called EKS, standing for Elastic Kubernetes Service. AWS Management Console (Console) and AWS CLI (CLI) are two primary interfaces for user to use in interacting with EKS.

This write-up walks you through a process of how to set up an EKS cluster and how to run containerized apps in the cluster using both Console and CLI in your AWS cloud environment.

Overview

Before creating a cluster, you need to meet a number of prerequisites.

1. Set up AWS CLI on a local workstation (WS) with access to your AWS account
2. Install kubectl utility on WS
3. Install eksctl on WS
4. Create EKS cluster role
5. Create EKS cluster node group role
6. Create Key Pair for access to EKS cluster nodes
7. Create EKS VPC (VPC) to deploy EKS cluster, if that VPC does not already exists

How-to details are provided in [Preparing to Create EKS Cluster](#) section below

After you have done all your preparation work, you are ready to create a cluster and verify it all works by deploying a container app in the cluster. A few steps in doing so are listed below.

1. Create EKS cluster (Cluster) using AWS Management Console (Console)
2. Check authentication
3. Set up or update kubeconfig for kubectl to use in managing EKS cluster created in Step 8
4. Create KS cluster node group (NG) using Console, and check NG nodes status before deploying a container app to the cluster
5. Deploy a sample app

AWS EKS

How-to details are provided in [Creating EKS Cluster](#) section

To minimize your cost for using EKS, you may remove a cluster that you don't intend to keep. Unlike in Azure where you can simply remove your resource group that contains all your cluster resources and nothing else to accomplish your cluster removal by one click in Azure Portal, equivalent to AWS Management Console, or by one command in Azure CLI, similar to AWS CLI, it takes a quite few steps for you to remove a cluster cleanly from your AWS cloud space. These steps are listed below.

1. Check sample app running status
2. Remove the sample app
3. Remove NG
4. Remove Cluster
5. Remove VPC

How-to details are provided in [Removing EKS Cluster](#) section

To recap, first of all, there are 7 steps for you to take to get ready before creating your EKS cluster. Secondly, 5 steps for you to take in creating and verifying your cluster. Thirdly, there are another 5 steps for you to take in removing your cluster cleanly.

Now that you are clear of your roadmap. You are ready to move on.

Environment and naming convention

It is highly advisable that you devise a naming standard (NS), if you don't have one already, to guide you in naming EKS cluster resources that you are going to create.

In this write-up, a simple NS was adopted. The NS is based on three elements.

1. Account alias
2. Selected Region
3. Service type

Table 1 below illustrates how this NS works in resource naming

Table-1 Naming Convention

Item	Name	Comment
Account alias	dls1	
Region	usw2	Oregon (us-west-2)
Cluster	dls1-usw2-eks-cluster-201206	201206 was the cluster creation date
Cluster Node Group	dls1-usw2-eks-cluster-node-grp-201206	
Cluster role	dls1-eks-cluster-role	This is an IAM role, global, having no region scope
Cluster NG role	<u>dls1-eks-cluster-node-grp-role</u>	This is an IAM role, global, having no

AWS EKS

		region scope
Cluster NG Key Pair	dls1-usw2-eks-cluster-node-grp-keypair	
VPC	dls1-usw2-eks-cluster-vpc-201206	201206 was the VPC creation date

Note:

An IAM admin user was used in the following examples. The user was granted with full permissions to EKS. The user has programmatic access to the AWS account. When interacting with EKS using AWS CLI, the user accesses from a Linux workstation, and not from AWS Cloud Shell. The user has AWS CLI for Linux installed on his workstation.

Preparing to Create EKS Cluster

Prep work steps 1 through 7

1. Install AWS CLI version 2

```
awsdls1admin@ubun1802:~$ aws --version
aws-cli/2.1.2 Python/3.7.3 Linux/4.15.0-123-generic exe/x86_64.ubuntu.18
awsdls1admin@ubun1802:~$
```

2. Install kubectl (installed previously)

```
awsdls1admin@ubun1802:~$ kubectl version --short --client
Client Version: v1.17.2
awsdls1admin@ubun1802:~$
```

3. Install eksctl

```
awsdls1admin@ubun1802:~$ curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -
s)_amd64.tar.gz" | tar xz -C /tmp
awsdls1admin@ubun1802:~$ sudo mv /tmp/eksctl /usr/local/bin
awsdls1admin@ubun1802:~$ eksctl version
0.32.0
awsdls1admin@ubun1802:~$ which eksctl
/usr/local/bin/eksctl
awsdls1admin@ubun1802:~$
```

4. Create EKS cluster role (that includes AmazonEKSClusterPolicy)

Use AWS IAM Console to create the required role Cluster Role **dls1-eks-cluster-role**
Screen 1 – Select EKS

AWS EKS

shladmin @ 9936-3688-3118

Create role

1234

Select type of trusted entity

**AWS service**
EC2, Lambda and others

**Another AWS account**
Belonging to you or 3rd party

**Web identity**
Cognito or any OpenID provider

**SAML 2.0 federation**
Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

API Gateway	CodeBuild	EKS	Lambda	SMS
AWS Backup	CodeDeploy	EMR	Lex	SNS
AWS Support	Config	ElastiCache	License Manager	SWF
Amplify	Connect	Elastic Beanstalk	Machine Learning	SageMaker
AppSync	DMS	Elastic Container Service	Macie	Security Hub
Application Auto Scaling	Data Lifecycle Manager	Elastic Transcoder	MediaConvert	Service Catalog
Application Discovery Service	Data Pipeline	ElasticLoadBalancing	OpsWorks	Step Functions
Auto Scaling	DataSync	Glue	RAM	Storage Gateway
Batch	DeepLens	Greengrass	RDS	Transfer
CloudFormation	Directory Service	GuardDuty	Redshift	Trusted Advisor
CloudHSM	DynamoDB	Inspector	Rekognition	VPC
CloudTrail	EC2	IoT	S3	WorkLink
	EC2 - Fleet	Kinesis		

* Required

Cancel

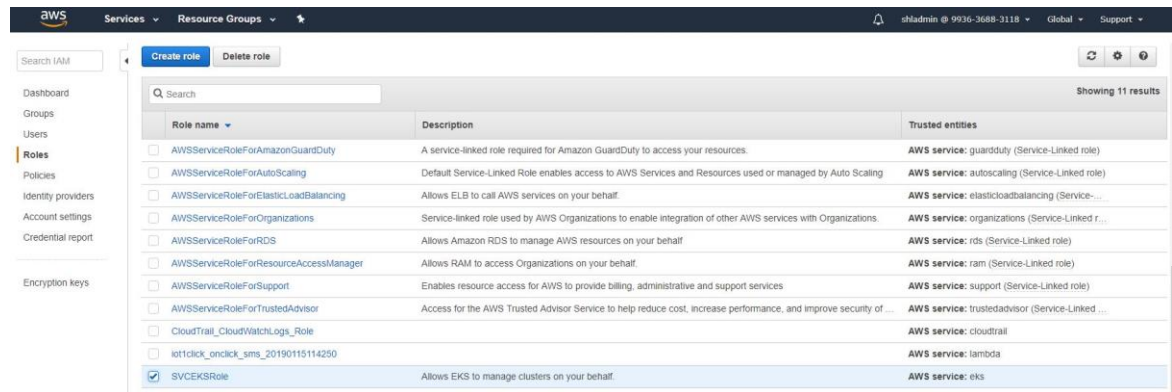
Next: Permissions

Screen 2 – Attach two policies

Screen 3 – Review and Create

V0.1

Screen 4 – Role created



5. Create EKS node (group) role that includes three (3) policies

dls1-eks-cluster-node-grp-role

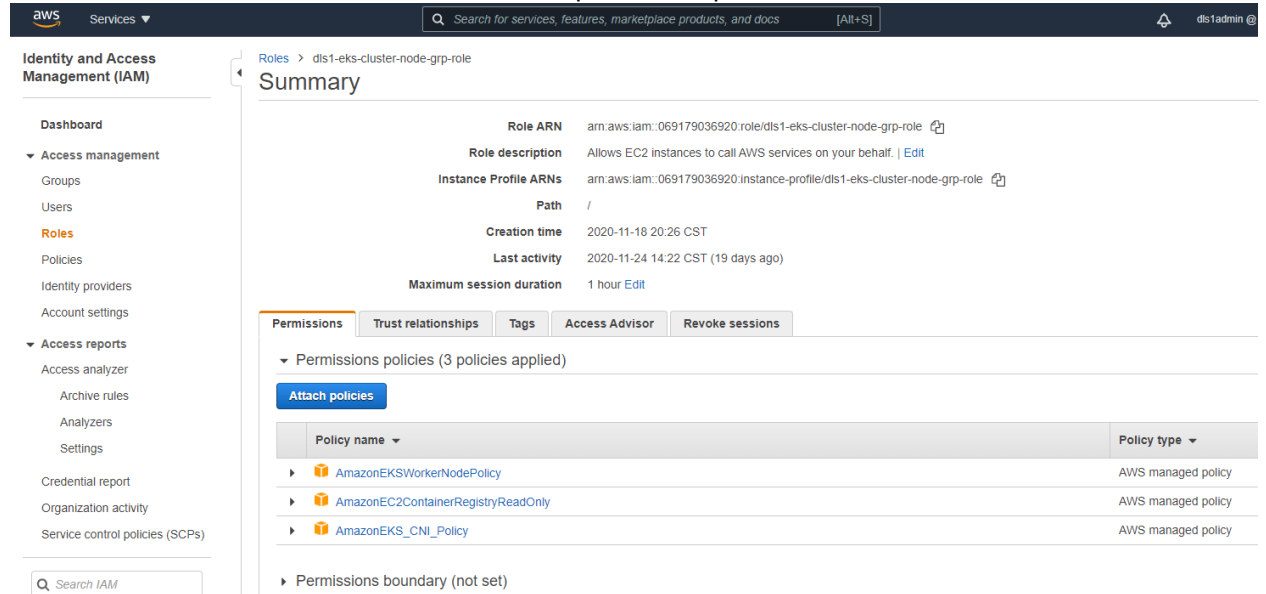
This role needs to include three policies. Those three policies are:

AmazonEKSWorkerNodePolicy

AmazonEC2ContainerRegistryReadOnly

AmazonEKS_CNI_Policy

Screen 1 - The role created that includes three permission policies

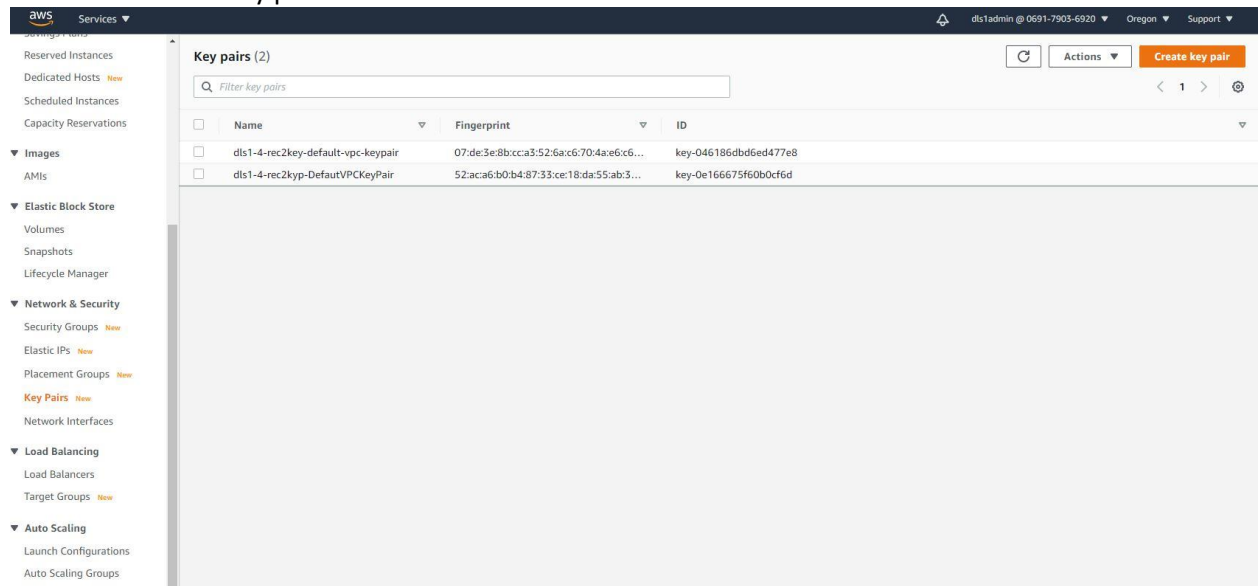


6. Create EKS cluster node group key pair

dls1-usw2-eks-cluster-node-grp-keypair

AWS EKS

Screen 1 – Create key pair EC2 Console



Screen 2 – Name key pair. Specify file format

EC2 > Key pairs > Create key pair

Create key pair

Key pair
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

File format

☐ pem
For use with OpenSSH

☒ ppk
For use with PuTTY

Tags (Optional)
No tags associated with the resource.

You can add 50 more tags

7. Create EKS VPC and subnets using AWS provided CloudFormation template

dls1-use1-eks-cluster-vpc-201120 (two public subnets and two private subnets)

<https://amazon-eks.s3.us-west-2.amazonaws.com/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml>

Screen 1 – Create VPC in CloudFormation (CF) Console, using CF template provided (See in Appendix A the template)

The screenshot shows the 'Create stack' wizard in the AWS CloudFormation console. The left sidebar indicates the current step is 'Step 1: Specify template'. The main content area is titled 'Create stack' and contains two sections: 'Prerequisite - Prepare template' and 'Specify template'. In the 'Prerequisite' section, the 'Template is ready' radio button is selected. In the 'Specify template' section, the 'Amazon S3 URL' radio button is selected, and the 'Amazon S3 URL' text box contains the URL: `https://amazon-eks.s3.us-west-2.amazonaws.com/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml`. Below this, the 'S3 URL' is displayed as `https://amazon-eks.s3.us-west-2.amazonaws.com/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml`. At the bottom right, there are 'Cancel' and 'Next' buttons.

Screen 2 – Specify CF stack details

The screenshot shows the 'Specify stack details' screen in the AWS CloudFormation console. The title is 'Specify stack details'. The first section is 'Stack name', with a text box containing 'dls1-use1-eks-cluster-vpc-201120'. Below this is a note: 'Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-)'. The second section is 'Parameters', with a note: 'Parameters are defined in your template and allow you to input custom values when you create or update a stack.' Below this is the 'Worker Network Configuration' section, which contains five text boxes for CIDR blocks: 'VpcBlock' (192.168.0.0/16), 'PublicSubnet01Block' (192.168.0.0/18), 'PublicSubnet02Block' (192.168.64.0/18), 'PrivateSubnet01Block' (192.168.128.0/18), and 'PrivateSubnet02Block' (192.168.192.0/18). At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons.

Review
dis1-use1-eks-vpc-201120

Step 1: Specify template
6/20

Template

Template URL

<https://amazon-eks-s3-us-west-2.amazonaws.com/latest/manifest/2020-10-26/amazon-eks-vpc-private-subnets.yaml>

Stack description

Amazon VPC Sample VPC - Private and Public subnets

Estimated cost

Step 2: Specify stack details
6/20

Parameters (5)

Key	Value
PrivateSubnet1IDblock	fs2-168-126.0/18
PrivateSubnet2IDblock	fs2-168-182.0/18
PublicSubnet1IDblock	fs2-168.0.0/16
PublicSubnet2IDblock	fs2-168-54.0/18
VPCIDblock	fs2-168.0.0/16

Step 3: Configure stack options
6/20

Tags (0)

Key	Value
No tags There is no tag default for this stack	

Permissions

No permissions

There is no IAM role associated with this stack

Stack policy

No stack policy

There is no stack policy defined

Rollback configuration

Rollback timeout

Rollback timeout: 300 seconds

Rollback maximum attempts

Rollback maximum attempts: 3

Notification options

No notification options

There are no notification options defined

Stack creation options

Rollback on failure

Enabled

Timeout

300 seconds

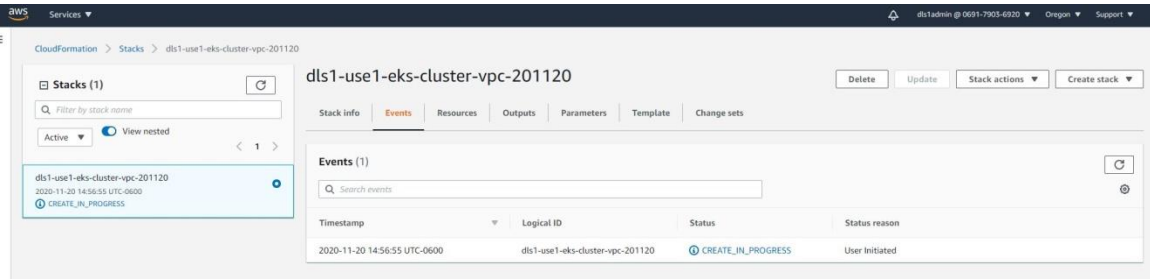
Termination protection

Disabled

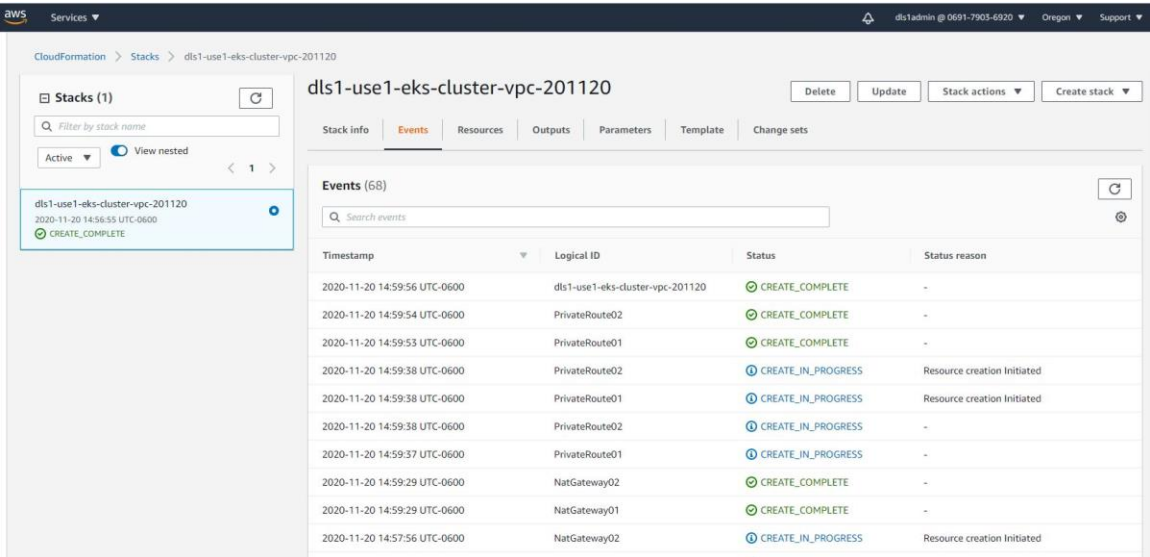
Quick-create link

Cancel
Preview
Create change set
Create stack

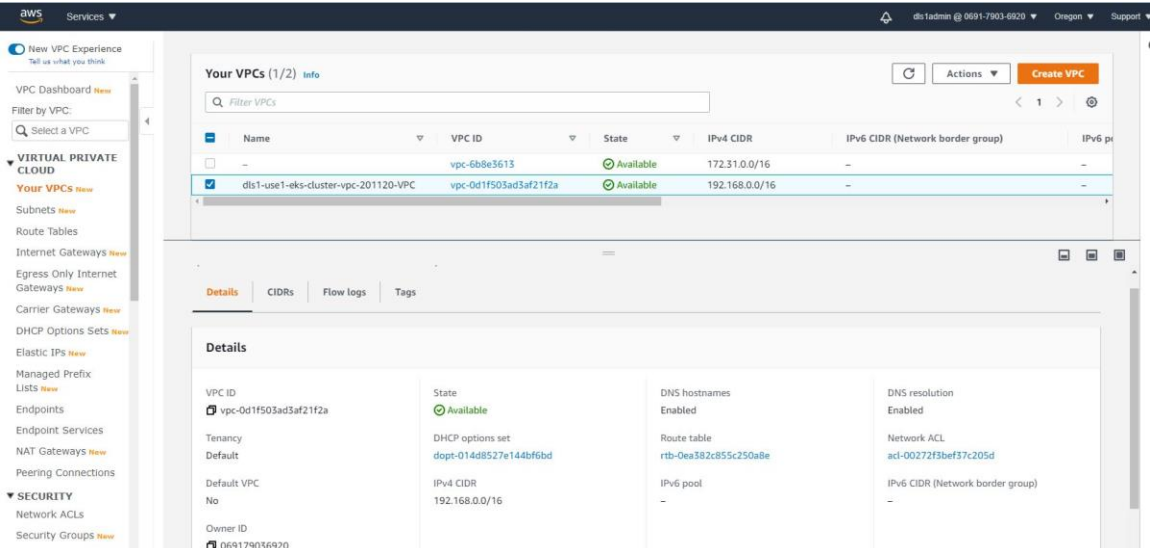
AWS EKS



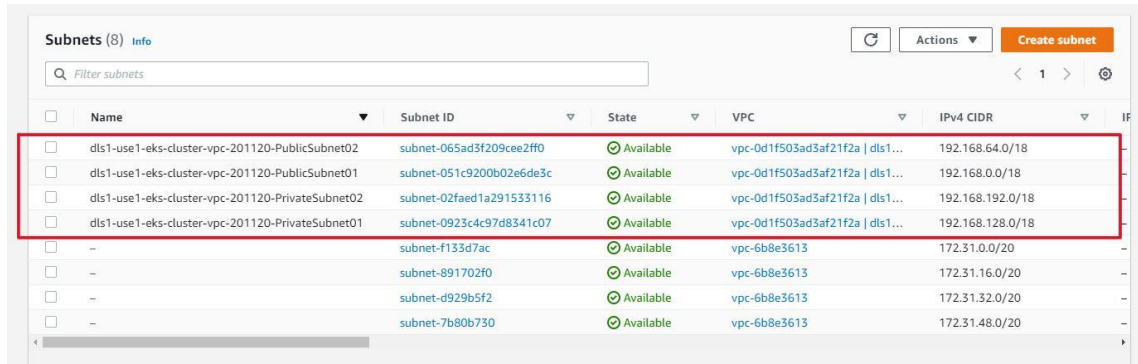
Screen 5 – Events occurred in stack creation process



Screen 6 – Stack created and available



Screen 7 – In VPC Console four subnets showed up

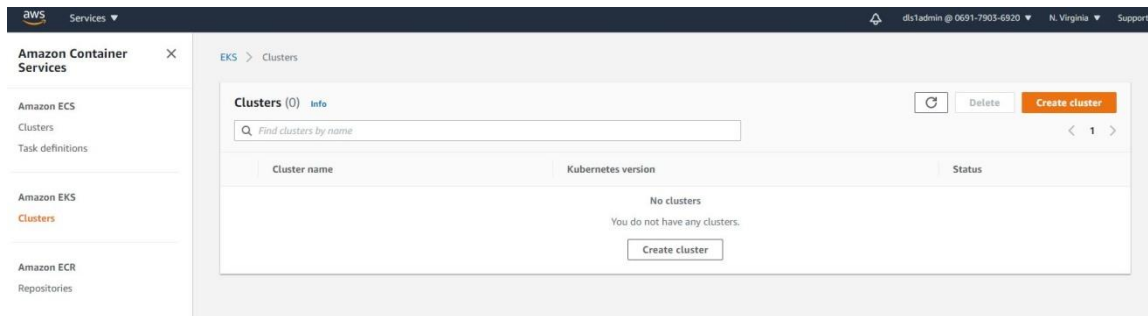


☐	Name	Subnet ID	State	VPC	IPv4 CIDR	IF
☐	dls1-use1-eks-cluster-vpc-201120-PublicSubnet02	subnet-065ad3f209cee2ff0	Available	vpc-0d1f503ad3af21f2a dls1...	192.168.64.0/18	
☐	dls1-use1-eks-cluster-vpc-201120-PublicSubnet01	subnet-051c9200b02e6de3c	Available	vpc-0d1f503ad3af21f2a dls1...	192.168.0.0/18	
☐	dls1-use1-eks-cluster-vpc-201120-PrivateSubnet02	subnet-02faed1a291533116	Available	vpc-0d1f503ad3af21f2a dls1...	192.168.192.0/18	
☐	dls1-use1-eks-cluster-vpc-201120-PrivateSubnet01	subnet-0923c4c97d8341c07	Available	vpc-0d1f503ad3af21f2a dls1...	192.168.128.0/18	
☐	-	subnet-f133d7ac	Available	vpc-6b8e3613	172.31.0.0/20	
☐	-	subnet-891702f0	Available	vpc-6b8e3613	172.31.16.0/20	
☐	-	subnet-d929b5f2	Available	vpc-6b8e3613	172.31.32.0/20	
☐	-	subnet-7b80b730	Available	vpc-6b8e3613	172.31.48.0/20	

Creating EKS Cluster

8. Create EKS cluster using console dls1-usw2-eks-cluster-201123

Screen 1 – In EKS Console create cluster



Screen 2 – Configure cluster

AWS EKS

aws

Services

Amazon Container Services

Amazon ECS

Clusters

Task definitions

Amazon EKS

Clusters

Amazon ECR

Repositories

EKS > Clusters > Create EKS cluster

Step 1
Configure cluster

Step 2
Specify networking

Step 3
Configure logging

Step 4
Review and create

Configure cluster

Cluster configuration

Name

Not editable after creation.
Enter a unique name for this cluster.

dls1-usw2-eks-cluster-201120

Kubernetes version

Info

Select the Kubernetes version for this cluster.

1.17

Cluster Service Role

Info

Not editable after creation.
Select the IAM Role to allow the Kubernetes control plane to manage AWS resources on your behalf.
To create a new role, go to the [IAM console](#).

dls1-eks-cluster-role

Secrets encryption

Info

These properties cannot be changed after the cluster is created.

☐

Enable envelope encryption of Kubernetes secrets using KMS

Enable envelope encryption to provide an additional layer of encryption for your Kubernetes secrets.

Tags (0)

Info

This cluster does not have any tags.

Add tag

Remaining tags available to add: 50

Cancel

Next

Screen 2 – Specify networking

aws

Services

Amazon Container Services

Amazon ECS

Clusters

Task definitions

Amazon EKS

Clusters

Amazon ECR

Repositories

Step 1
Configure cluster

Step 2
Specify networking

Step 3
Configure logging

Step 4
Review and create

Specify networking

Networking

Info

These properties cannot be changed after the cluster is created.

VPC

Info

Select a VPC to use for your EKS Cluster resources.
To create a new VPC, go to the [VPC console](#).

vpc-0d1f503ad3af21f2a | dls1-us-east-1-eks-cluster-vpc-201120-VPC

Subnets

Info

Choose the subnets in your VPC where the control plane may place elastic network interfaces (ENIs) to facilitate communication with your cluster.
To create a new subnet, go to the corresponding page in the [VPC console](#).

Select subnets

subnet-051c9200b02e6de3c

subnet-065ad3f209cee2ff0

subnet-0923c4c97d8341c07

subnet-02faed1a291533116

Security groups

Info

Choose the security groups to apply to the EKS-managed Elastic Network Interfaces that are created in your worker node subnets.
To create a new security group, go to the corresponding page in the [VPC console](#).

Select security groups

sg-03cfa5ba8fe5f9ca4

☒

Configure Kubernetes Service IP address range

Info

Specify the range from which cluster services will receive IP addresses.

Cluster endpoint access

Info

Configure access to the Kubernetes API server endpoint.

☐Public

The cluster endpoint is accessible from outside of your VPC. Worker node traffic will leave your VPC to connect to the endpoint.

☒Public and private

The cluster endpoint is accessible from outside of your VPC. Worker node traffic to the endpoint will stay within your VPC.

☐Private

The cluster endpoint is only accessible through your VPC. Worker node traffic to the endpoint will stay within your VPC.

Advanced Settings

Cancel

Previous

Next

Screen 3 – Configure logging

EKS > Clusters > Create EKS cluster

Step 1
Configure cluster

Step 2
Specify networking

Step 3
Configure logging

Step 4
Review and create

Configure logging

Control Plane Logging [Info](#)

CloudWatch log group
Send audit and diagnostic logs from the Amazon EKS control plane to CloudWatch Logs.

API server
Logs pertaining to API requests to the cluster.
☐ Disabled

Audit
Logs pertaining to cluster access via the Kubernetes API.
☐ Disabled

Authenticator
Logs pertaining to authentication requests into the cluster.
☐ Disabled

Controller manager
Logs pertaining to state of cluster controllers.
☐ Disabled

Scheduler
Logs pertaining to scheduling decisions.
☐ Disabled

Cancel Previous **Next**

Screen 4 – Review and create

AWS EKS

Step 1
Configure cluster

Step 2
Specify networking

Step 3
Configure logging

Step 4
Review and create

Review and create

Step 1: Configure cluster

Cluster configuration

Name - *Not editable after creation.*
dls1-usw2-eks-cluster-201120

Kubernetes version
1.17

Cluster Service Role - *Not editable after creation.*
arn:aws:iam::069179036920:role/dls1-eks-cluster-role

Tags (0)

Key	Value
No tags	

This cluster does not have any tags.

Step 2: Specify networking

Networking

These properties cannot be changed after the cluster is created.

VPC
vpc-0d1f503ad3af21f2a

Subnets
subnet-051c9200b02e6de3c
subnet-065ad3f209cee2ff0
subnet-0923c4c97d8341c07
subnet-02faed1a291533116

Security groups
sg-03cfa5ba8fe5f9ca4

Cluster endpoint access

API server endpoint access
Public and private

Step 3: Configure logging

Control Plane Logging

API server
Disabled

Audit
Disabled

Authenticator
Disabled

Controller manager
Disabled

Scheduler
Disabled

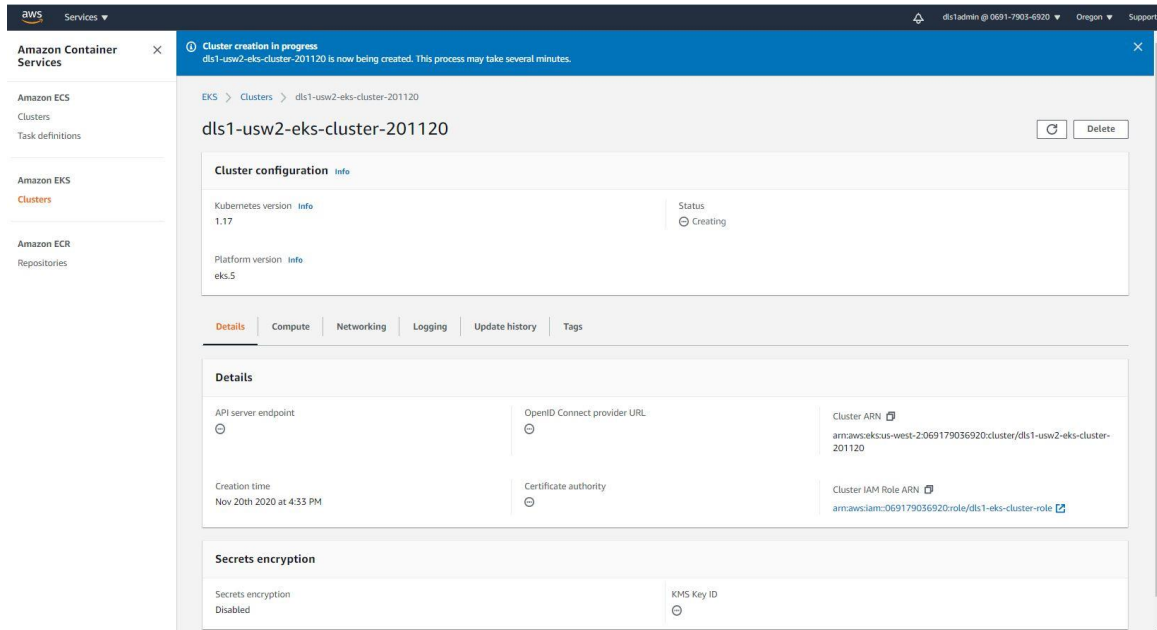
Cancel

Previous

Create

Screen 5 – Cluster being created

AWS EKS



9. Check authentication

```
awsdls1admin@ubun1802:~$ aws sts get-caller-identity
{
  "UserId": "AIDARAG3EKT4MVCKMWGJ5",
  "Account": "069179036920",
  "Arn": "arn:aws:iam::069179036920:user/dls1admin"
}
awsdls1admin@ubun1802:~$
```

10. Create / update configuration kubeconfig, and verify

```
awsdls1admin@ubun1802:~$ aws eks --region us-west-2 update-kubeconfig --name dls1-usw2-eks-cluster-201123
Added new context arn:aws:eks:us-west-2:069179036920:cluster/dls1-usw2-eks-cluster-201123
to /home/awsdls1admin/.kube/config
awsdls1admin@ubun1802:~$
```

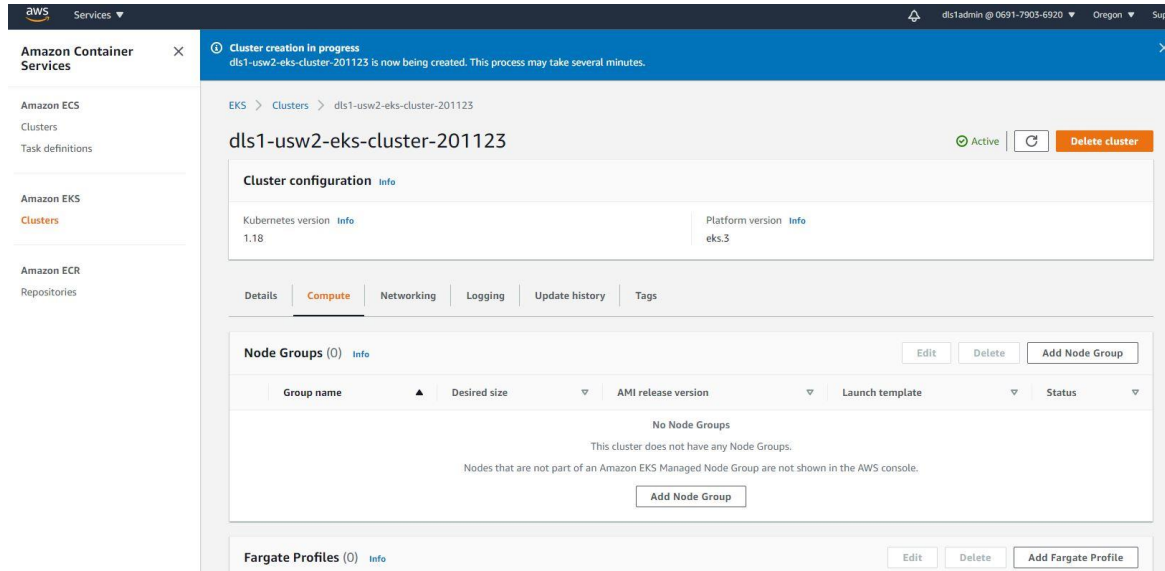
```
awsdls1admin@ubun1802:~$ kubectl get svc
NAME      TYPE      CLUSTER-IP  EXTERNAL-IP  PORT(S)  AGE
kubernetes ClusterIP  10.100.0.1   <none>       443/TCP  52m
awsdls1admin@ubun1802:~$
```

11. Create EKS node group (Managed Node Group) using console

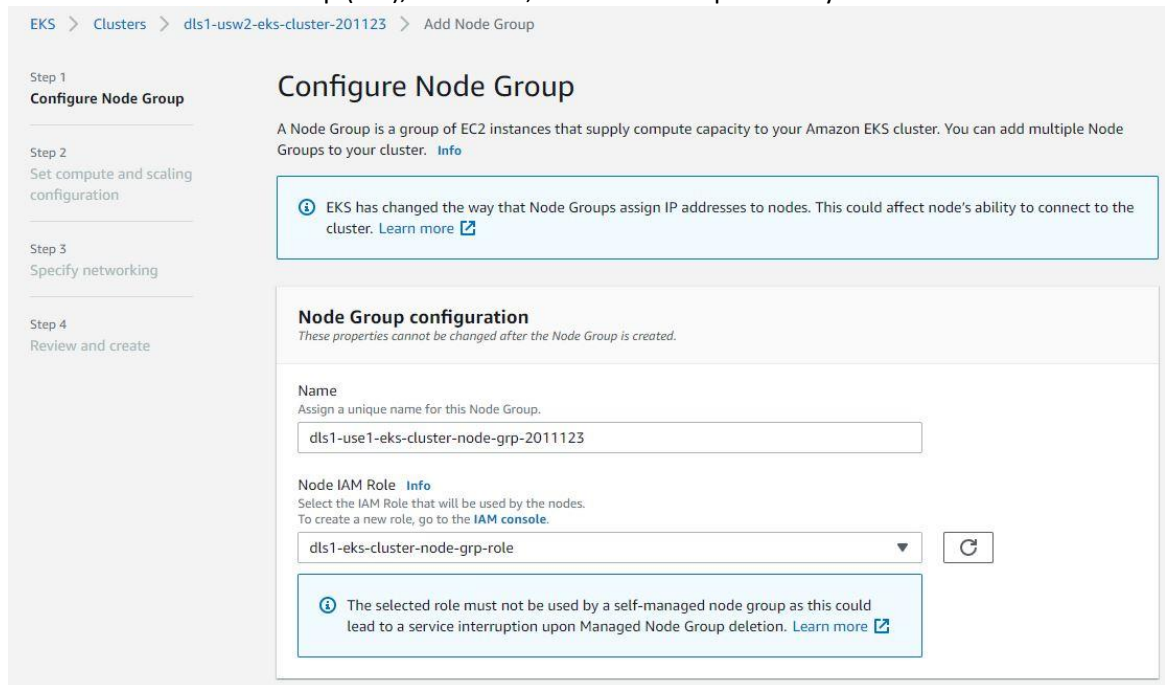
dls1-usw2-eks-cluster-node-grp-2011123

Screen 1 – in EKS Console, select Cluster just created. Select Compute tab

AWS EKS



Screen 2 – Add Node Group (NG), name NG, select NG role previously created



Screen 3 – Use Default, go to Next

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. [Learn more](#)

Launch template [Info](#)

These properties cannot be changed after the Node Group is created.

☒ Use launch template
 Configure this Node Group using an EC2 launch template.

Kubernetes labels [Info](#)

This Node Group does not have any labels.

Remaining labels available to add: 50

Tags (0) [Info](#)

This Node Group does not have any tags.

Remaining tags available to add: 50

Cancel

Screen 4 – Set compute and scaling configuration

EKS

>

Clusters

>

dls1-usw2-eks-cluster-201123

>

Add Node Group

Step 1

Configure Node Group

Step 2

Set compute and scaling configuration

Step 3

Specify networking

Step 4

Review and create

Set compute and scaling configuration

Node Group compute configuration

These properties cannot be changed after the Node Group is created.

AMI type [Info](#)
 Select the EKS-optimized Amazon Machine Image for nodes.

Instance type [Info](#)
 Select the EC2 instance type for nodes.

Disk size
 Select the size of the attached EBS volume for each node.
 GiB

Node Group scaling configuration

Minimum size
 Set the minimum number of nodes that the group can scale in to.
 nodes

Maximum size
 Set the maximum number of nodes that the group can scale out to.
 nodes

Desired size
 Set the desired number of nodes that the group should launch with initially.
 nodes

Cancel

Screen 5 – Specify networking

EKS > Clusters > dls1-usw2-eks-cluster-201123 > Add Node Group

Step 1
Configure Node Group

Step 2
Set compute and scaling configuration

Step 3
Specify networking

Step 4
Review and create

Specify networking

Node Group network configuration
These properties cannot be changed after the Node Group is created.

Subnets [Info](#)
Specify the subnets in your VPC where your nodes will run.
To create a new subnet, go to the corresponding page in the [VPC console](#).

Select subnets

subnet-f133d7ac X subnet-891702f0 X subnet-d929b5f2 X
subnet-7b80b730 X

☒ **Allow remote access to nodes** [Info](#)
Without remote access enabled you will not be able to directly connect to nodes after they are created.

SSH key pair
Select an SSH key pair to allow secure remote access to your nodes.
To create a new SSH key pair, go to the corresponding page in the [EC2 console](#).

dls1-usw2-eks-cluster-node-grp-keypair

Allow remote access from
Configure the source IP ranges that can remotely access nodes.

☒ **All**
Do not restrict source IPs that can remotely access nodes.

☐ **Selected security groups**
Specify security groups to restrict which source IPs can remotely access nodes.

Cancel Previous **Next**

Screen 6 – Review and create

EKS > Clusters > dls1-usw2-eks-cluster-201123 > Add Node Group

Step 1
Configure Node Group

Step 2
Set compute and scaling configuration

Step 3
Specify networking

Step 4
Review and create

Review and create

Step 1: Configure Node Group [Edit](#)

Node Group configuration

Name	Node IAM Role
dls1-use1-eks-cluster-node-grp-2011123	arn:aws:iam::069179036920:role/dls1-eks-cluster-node-grp-role

Kubernetes labels (0)

Key	Value
No labels	

This Node Group does not have any Kubernetes labels.

Tags (0)

Key	Value
No tags	

This Node Group does not have any tags.

Step 2: Set compute and scaling configuration [Edit](#)

Screen 7 – Review and create (Continued)

Key	Value
No tags	
This Node Group does not have any tags.	

Step 2: Set compute and scaling configuration

Edit

Node Group compute configuration

AMI type	Instance type	Disk size
Amazon Linux 2 (AL2_x86_64)	t3.small	20

Node Group scaling configuration

Minimum size	Maximum size	Desired size
2 nodes	4 nodes	2 nodes

Step 3: Specify networking

Edit

Node Group network configuration

Subnets	Allow remote access to nodes	Allow remote access from
subnet-f133d7ac subnet-891702f0 subnet-d929b5f2 subnet-7b80b730	Enabled	All
	SSH key pair	
	dls1-usw2-eks-cluster-node-grp-keypair	

Cancel

Previous

Create

Screen 8 – NG being created

AWS EKS

dls1-use1-eks-cluster-node-grp-2011123 Refresh Edit Delete

Node Group configuration [Info](#)

Kubernetes version 1.18	AMI type Info AL2_x86_64	Status Creating
AMI release version Info 1.18.9-20201117	Instance type t3.small	Disk size 20 GiB

[Details](#) | [Health issues](#) 0 | [Kubernetes labels](#) | [Update history](#) | [Tags](#)

Details

Node Group ARN Info arn:aws:eks:us-west-2:069179036920:nodegroup/dls1-usw2-eks-cluster-2011123/dls1-use1-eks-cluster-node-grp-2011123/60bafbc0-385c-94b8-74d7-9ccd453327a3 Creation time Nov 23rd 2020 at 5:46 PM	Autoscaling group name Node IAM Role ARN Info arn:aws:iam::069179036920:role/dls1-eks-cluster-node-grp-role Info	Minimum size 2 nodes Maximum size 4 nodes Desired size 2 nodes	Subnets subnet-f133d7ac Info subnet-891702f0 Info subnet-d929b5f2 Info subnet-7b80b730 Info Allow remote access to nodes Enabled SSH key pair dls1-usw2-eks-cluster-node-grp-keypair Allow remote access from All
--	--	---	---

Screen 9 – NG created and active

EKS > Clusters > dls1-usw2-eks-cluster-201123 > Node Group: dls1-use1-eks-cluster-node-grp-2011123

dls1-use1-eks-cluster-node-grp-2011123 Refresh

Node Group configuration [Info](#)

Kubernetes version 1.18	AMI type Info AL2_x86_64	Status Active
AMI release version Info 1.18.9-20201117	Instance type t3.small	Disk size 20 GiB

[Details](#) | [Health issues](#) 0 | [Kubernetes labels](#) | [Update history](#) | [Tags](#)

Verify status of nodes:

```
awsdls1admin@ubun1802:~$ kubectl get nodes --watch
```

```
NAME                                STATUS ROLES  AGE  VERSION
ip-192-168-108-148.ec2.internal    Ready <none> 63m  v1.17.12-eks-7684af
ip-192-168-138-102.ec2.internal    Ready <none> 63m  v1.17.12-eks-7684af
```

```
awsdls1admin@ubun1802:~$ kubectl get nodes --watch
```

```
NAME                                STATUS ROLES  AGE  VERSION
ip-172-31-12-3.us-west-2.compute.internal    Ready <none> 8m55s v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal    Ready <none> 8m57s v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal    Ready <none> 10m  v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal    Ready <none> 10m  v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal    Ready <none> 15m  v1.18.9-eks-d1db3c
```

```
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 15m v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal Ready <none> 20m v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 20m v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal Ready <none> 25m v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 25m v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal Ready <none> 30m v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 30m v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal Ready <none> 35m v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 35m v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal Ready <none> 40m v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 40m v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal Ready <none> 45m v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 45m v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal Ready <none> 50m v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 50m v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal Ready <none> 55m v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 55m v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal Ready <none> 60m v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 60m v1.18.9-eks-d1db3c
ip-172-31-41-3.us-west-2.compute.internal Ready <none> 65m v1.18.9-eks-d1db3c
ip-172-31-12-3.us-west-2.compute.internal Ready <none> 65m v1.18.9-eks-d1db3c
awsdl1admin@ubun1802:~$
```

Running Apps in EKS Cluster

Namespace in Cluster

12. Deploy a sample app (sample-service.yaml)

```
awsdl1admin@ubun1802:~$ kubectl create namespace my-namespace
namespace/my-namespace created
awsdl1admin@ubun1802:~$ kubectl apply -f /tmp/sample-service.yaml
service/my-service created
deployment.apps/my-deployment created
awsdl1admin@ubun1802:~$ kubectl get all -n my-namespace
NAME                                READY STATUS RESTARTS AGE
pod/my-deployment-778b477dd5-bnkk4 1/1   Running 0      36s
pod/my-deployment-778b477dd5-htldg 1/1   Running 0      36s
pod/my-deployment-778b477dd5-stgnp 1/1   Running 0      36s

NAME          TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)  AGE
service/my-service ClusterIP  10.100.17.218 <none>      80/TCP    36s

NAME          READY UP-TO-DATE AVAILABLE AGE
deployment.apps/my-deployment 3/3   3         3      36s
```



```
NAME                                DESIRED  CURRENT  READY  AGE
replicaset.apps/my-deployment-778b477dd5 3        3        3      37s
awsdl1admin@ubun1802:~$ kubectl -n my-namespace describe service my-service
Name:      my-service
Namespace:  my-namespace
Labels:     app=my-app
Annotations: kubectl.kubernetes.io/last-applied-configuration:
  {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{"labels":{"app":"my-app"},"name":"my-service","namespace":"my-namespace"}}...
Selector:   app=my-app
Type:       ClusterIP
IP:         10.100.17.218
Port:       <unset> 80/TCP
TargetPort: 80/TCP
Endpoints:   172.31.11.65:80,172.31.13.32:80,172.31.46.245:80
Session Affinity: None
Events:      <none>
awsdl1admin@ubun1802:~$
```

In this step, a name space “my-service” was create before deploying a sample app to it After the sample app was deployed, K8 runs it in three pods per “replica=3” in deployment YAML specification. See **Appendix B** for details.

Each pod has its own endpoint. Three pods, therefore, yielded three endpoints, via which pods interface with outside world. Each endpoint includes an IP address and a port number. K8 resolves the pod name to its endpoint by DNS

`kubectl get all -n my-namespace` displays pod names, sorted by name. See below

```
awsdl1admin@ubun1802:~$ kubectl get all -n my-namespace
NAME                                READY  STATUS   RESTARTS  AGE
pod/my-deployment-778b477dd5-bnkk4 1/1    Running   0          36s
pod/my-deployment-778b477dd5-htldg 1/1    Running   0          36s
pod/my-deployment-778b477dd5-stgnp 1/1    Running   0          36s
...
awsdl1admin@ubun1802:~$
```

`kubectl -n my-namespace describe service my-service | grep Endpoints` displays pod endpoints in one line, ordered by IP address. See below

```
awsdl1admin@ubun1802:~$ kubectl -n my-namespace describe service my-service | grep
Endpoints
Endpoints: 172.31.11.65:80,172.31.13.32:80,172.31.46.245:80
awsdl1admin@ubun1802:~$
```

Pod names and their endpoints do not match in their respective display order, though both are in ascending order, because Pod IP was assigned randomly. See Table 2 below

Table 2 – Sample app pod names and endpoints

POD Name	Endpoint	Node
my-deployment-778b477dd5-bnkk4	172.31.46.245	172.31.41.3
my-deployment-778b477dd5-htldg	172.31.11.65	172.31.12.3
my-deployment-778b477dd5-stgnp	172.31.13.32	172.31.12.3

K8 runs two (2) work nodes per Cluster NG specification. See Step 11 Screen 4 Desired nodes = 2. Three pods are distributed to run on the two nodes, two pods ended up running on one same node with IP of **172.31.12.3**. It indicates that K8 load balances pods among work nodes by round-robin algorithm.

Pod details

To show pod details, use command `kubectl -n <name space> describe pod <pod name>`. See three pod examples below. For easy identification, the pods are color-coded in red, green, and blue.

Pod in red

```
awsdl1admin@ubun1802:~$ kubectl -n my-namespace describe pod my-deployment-778b477dd5-bnkk4
Name:      my-deployment-778b477dd5-bnkk4
Namespace: my-namespace
Priority:   0
Node:      ip-172-31-41-3.us-west-2.compute.internal/172.31.41.3
Start Time: Tue, 24 Nov 2020 01:56:48 +0000
Labels:    app=my-app
           pod-template-hash=778b477dd5
Annotations: kubernetes.io/psp: eks.privileged
Status:     Running
IP:         172.31.46.245
IPs:
  IP:      172.31.46.245
Controlled By: ReplicaSet/my-deployment-778b477dd5
Containers:
  nginx:
    Container ID:
docker://0568f8a86074015b5d18590b8a1734311a83c430b0eb8863759f3053ae4217ca
    Image:      nginx:1.19.2
    Image ID:   docker-
pullable://nginx@sha256:c628b67d21744fce822d22fdcc0389f6bd763daac23a6b77147d0712ea7102d0
    Port:      80/TCP
    Host Port: 0/TCP
    State:     Running
      Started: Tue, 24 Nov 2020 01:56:54 +0000
    Ready:     True
```

```
Restart Count: 0
Environment: <none>
Mounts:
  /var/run/secrets/kubernetes.io/serviceaccount from default-token-hthcw (ro)
Conditions:
  Type           Status
  Initialized     True
  Ready           True
  ContainersReady True
  PodScheduled    True
Volumes:
  default-token-hthcw:
    Type:      Secret (a volume populated by a Secret)
    SecretName: default-token-hthcw
    Optional:  false
QoS Class:     BestEffort
Node-Selectors: <none>
Tolerations:   node.kubernetes.io/not-ready:NoExecute for 300s
                node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason      Age   From          Message
  ----    -
  Normal  Scheduled   2m24s default-scheduler   Successfully assigned my-namespace/my-deployment-778b477dd5-bnkk4 to ip-172-31-41-3.us-west-2.compute.internal
  Normal  Pulling     2m23s kubelet, ip-172-31-41-3.us-west-2.compute.internal   Pulling image "nginx:1.19.2"
  Normal  Pulled      2m18s kubelet, ip-172-31-41-3.us-west-2.compute.internal   Successfully pulled image "nginx:1.19.2"
  Normal  Created     2m18s kubelet, ip-172-31-41-3.us-west-2.compute.internal   Created container nginx
  Normal  Started     2m18s kubelet, ip-172-31-41-3.us-west-2.compute.internal   Started container nginx
```

Pod in green

```
awsdls1admin@ubun1802:~$ kubectl -n my-namespace describe pod my-deployment-778b477dd5-htldg
Name:         my-deployment-778b477dd5-htldg
Namespace:    my-namespace
Priority:      0
Node:         ip-172-31-12-3.us-west-2.compute.internal/172.31.12.3
Start Time:   Tue, 24 Nov 2020 01:56:48 +0000
Labels:       app=my-app
              pod-template-hash=778b477dd5
Annotations:  kubernetes.io/psp: eks.privileged
Status:       Running
IP:           172.31.11.65
IPs:
  IP:         172.31.11.65
```

AWS EKS

Controlled By: ReplicaSet/my-deployment-778b477dd5

Containers:

nginx:

Container ID:

docker://6e67e40d5a72c8e1a080767473decc43eedfb604e5d9d9aea98d137a1127f301

Image: nginx:1.19.2

Image ID: docker-

pullable://nginx@sha256:c628b67d21744fce822d22fdcc0389f6bd763daac23a6b77147d0712ea7102d0

Port: 80/TCP

Host Port: 0/TCP

State: Running

Started: Tue, 24 Nov 2020 01:56:55 +0000

Ready: True

Restart Count: 0

Environment: <none>

Mounts:

/var/run/secrets/kubernetes.io/serviceaccount from default-token-hthcw (ro)

Conditions:

Type Status

Initialized True

Ready True

ContainersReady True

PodScheduled True

Volumes:

default-token-hthcw:

Type: Secret (a volume populated by a Secret)

SecretName: default-token-hthcw

Optional: false

QoS Class: BestEffort

Node-Selectors: <none>

Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s

node.kubernetes.io/unreachable:NoExecute for 300s

Events:

Type	Reason	Age	From	Message
Normal	Scheduled	3m4s	default-scheduler	Successfully assigned my-namespace/my-deployment-778b477dd5-htldg to ip-172-31-12-3.us-west-2.compute.internal
Normal	Pulling	3m2s	kubelet, ip-172-31-12-3.us-west-2.compute.internal	Pulling image "nginx:1.19.2"
Normal	Pulled	2m58s	kubelet, ip-172-31-12-3.us-west-2.compute.internal	Successfully pulled image "nginx:1.19.2"
Normal	Created	2m57s	kubelet, ip-172-31-12-3.us-west-2.compute.internal	Created container nginx
Normal	Started	2m57s	kubelet, ip-172-31-12-3.us-west-2.compute.internal	Started container nginx

Pod in blue

```
awsdl1admin@ubun1802:~$ kubectl -n my-namespace describe pod my-deployment-778b477dd5-stgnp
Name:      my-deployment-778b477dd5-stgnp
Namespace: my-namespace
Priority:   0
Node:      ip-172-31-12-3.us-west-2.compute.internal/172.31.12.3
Start Time: Tue, 24 Nov 2020 01:56:48 +0000
Labels:    app=my-app
           pod-template-hash=778b477dd5
Annotations: kubernetes.io/psp: eks.privileged
Status:     Running
IP:         172.31.13.32
IPs:
  IP:      172.31.13.32
Controlled By: ReplicaSet/my-deployment-778b477dd5
Containers:
  nginx:
    Container ID:
      docker://4f8be18e7c20d03f6b5eb39e28daa41a37ebbc67f38f02e7976f68874d9ae4b6
    Image:      nginx:1.19.2
    Image ID:    docker-
    pullable://nginx@sha256:c628b67d21744fce822d22fdcc0389f6bd763daac23a6b77147d0712ea7102d0
    Port:       80/TCP
    Host Port:   0/TCP
    State:      Running
      Started:   Tue, 24 Nov 2020 01:56:55 +0000
    Ready:      True
    Restart Count: 0
    Environment: <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-hthcw (ro)
Conditions:
  Type           Status
  Initialized     True
  Ready           True
  ContainersReady True
  PodScheduled    True
Volumes:
  default-token-hthcw:
    Type:      Secret (a volume populated by a Secret)
    SecretName: default-token-hthcw
    Optional:   false
QoS Class:     BestEffort
Node-Selectors: <none>
Tolerations:   node.kubernetes.io/not-ready:NoExecute for 300s
               node.kubernetes.io/unreachable:NoExecute for 300s
```

Events:

Type	Reason	Age	From	Message
Normal	Scheduled	4m24s	default-scheduler	Successfully assigned my-namespace/my-deployment-778b477dd5-stgnp to ip-172-31-12-3.us-west-2.compute.internal
Normal	Pulling	4m23s	kubelet, ip-172-31-12-3.us-west-2.compute.internal	Pulling image "nginx:1.19.2"
Normal	Pulled	4m18s	kubelet, ip-172-31-12-3.us-west-2.compute.internal	Successfully pulled image "nginx:1.19.2"
Normal	Created	4m17s	kubelet, ip-172-31-12-3.us-west-2.compute.internal	Created container nginx
Normal	Started	4m17s	kubelet, ip-172-31-12-3.us-west-2.compute.internal	Started container nginx

awsdl1admin@ubun1802:~\$

Accessing Pod

Since the sample app is a Linux based docker image, you can access each pod BASH Shell by the following command

```
awsdl1admin@ubun1802:~$ kubectl exec -it my-deployment-778b477dd5-bnkk4 -n my-namespace -- /bin/bash
root@my-deployment-778b477dd5-bnkk4:/# cat /etc/hosts
# Kubernetes-managed hosts file.
127.0.0.1    localhost
::1        localhost ip6-localhost ip6-loopback
fe00::0    ip6-localnet
fe00::0    ip6-mcastprefix
fe00::1    ip6-allnodes
fe00::2    ip6-allrouters
172.31.46.245 my-deployment-778b477dd5-bnkk4
root@my-deployment-778b477dd5-bnkk4:/# exit
exit
command terminated with exit code 127
```

```
awsdl1admin@ubun1802:~$ kubectl exec -it my-deployment-778b477dd5-htldg -n my-namespace -- /bin/bash
root@my-deployment-778b477dd5-htldg:/# cat /etc/hosts
# Kubernetes-managed hosts file.
127.0.0.1    localhost
::1        localhost ip6-localhost ip6-loopback
fe00::0    ip6-localnet
fe00::0    ip6-mcastprefix
fe00::1    ip6-allnodes
fe00::2    ip6-allrouters
172.31.11.65 my-deployment-778b477dd5-htldg
```

```
root@my-deployment-778b477dd5-htldg:/# exit
exit

awsdls1admin@ubun1802:~$ kubectl exec -it my-deployment-778b477dd5-stgnp -n my-
namespace -- /bin/bash
root@my-deployment-778b477dd5-stgnp:/# cat /etc/hosts
# Kubernetes-managed hosts file.
127.0.0.1    localhost
::1        localhost ip6-localhost ip6-loopback
fe00::0    ip6-localnet
fe00::0    ip6-mcastprefix
fe00::1    ip6-allnodes
fe00::2    ip6-allrouters
172.31.13.32 my-deployment-778b477dd5-stgnp
root@my-deployment-778b477dd5-stgnp:/# exit
exit
awsdls1admin@ubun1802:~$
```

Cluster networking

(Details to be added in future revision(s))

Removing EKS Cluster

13. Remove sample app namespace

```
awsdls1admin@ubun1802:~$ kubectl delete namespace my-namespace
namespace "my-namespace" deleted
awsdls1admin@ubun1802:~$
```

14. Delete cluster node group (in cluster)

In EKS Console, select Cluster dls1-usw2-eks-cluster-vpc-201123
In Cluster, select NG dls1-use1-eks-cluster-node-grp-201123
In Console, delete NG

15. Delete cluster

In EKS Console, select Cluster dls1-usw2-eks-cluster-201123
Delete Cluster

16. Delete cluster VPC (by CloudFormation stack)

In CloudFormation Console, select Stack dls1-usw2-eks-cluster-vpc-201123
Delete Stack

Summary

This write-up introduced and described three phases of using AWS EKS.

Phase 1 – Getting ready to create EKS cluster

Phase 2 – Creating and running EKS cluster

Phase 3 – Removing the EKS cluster (optional)

EKS cluster creation requires a VPC (Virtual Private Cloud) for it to be deployed to. The VPC needs to have at least two subnets. Those subnets can be all private or public. In this walk-through an AWS provided CloudFormation template was used to create that VPC with 2 public subnets and 2 private subnets in it. The template is listed in its entirety in Appendix A.

After cluster creation, containerized apps can be deployed to the cluster. The cluster launches and runs apps distributing the workload over the entire cluster automatically for you.

A small Nginx web server app was used in this walk-through. Its deployment was specified the app's YAML configuration file. According the YAML file, EKS cluster will launch three instances of app called pods. Those pods run simultaneously in the cluster. Should one pod fail, the cluster detects it and compensates for it by launching another pod, again, automatically for you. This way your cluster provides you with not only load-balancing but also high availability. The app's YAML file is listed in its entirety in Appendix B.

Details of the following are not discussed in this version of write-up. They will be added in future revisions.

1. How to grant enough permissions to an IAM user to create and manage EKS clusters
2. How to set up a Linux workstation to interact with EKS using AWS CLI
3. How to set up a Windows workstation to interact with EKS using AWS CLI
4. How to set up ECR (AWS Elastic Container Registry) for EKS to use

Author of this write-up always welcomes your feedback.

Please direct your questions and/or comments to Tony Shen at tony.shen@datacommlabs.com

Appendix A - EKS Cluster VPC CloudFormation Template

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Amazon EKS Sample VPC - Private and Public subnets'

Parameters:

  VpcBlock:
    Type: String
    Default: 192.168.0.0/16
    Description: The CIDR range for the VPC. This should be a valid private (RFC 1918) CIDR range.

  PublicSubnet01Block:
    Type: String
    Default: 192.168.0.0/18
    Description: CidrBlock for public subnet 01 within the VPC

  PublicSubnet02Block:
    Type: String
    Default: 192.168.64.0/18
    Description: CidrBlock for public subnet 02 within the VPC

  PrivateSubnet01Block:
    Type: String
    Default: 192.168.128.0/18
    Description: CidrBlock for private subnet 01 within the VPC

  PrivateSubnet02Block:
    Type: String
    Default: 192.168.192.0/18
    Description: CidrBlock for private subnet 02 within the VPC

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      -
        Label:
          default: "Worker Network Configuration"
        Parameters:
          - VpcBlock
          - PublicSubnet01Block
          - PublicSubnet02Block
          - PrivateSubnet01Block
          - PrivateSubnet02Block

Resources:
  VPC:
```

Type: AWS::EC2::VPC

Properties:

CidrBlock: !Ref VpcBlock

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: !Sub '\${AWS::StackName}-VPC'

InternetGateway:

Type: "AWS::EC2::InternetGateway"

VPCGatewayAttachment:

Type: "AWS::EC2::VPCGatewayAttachment"

Properties:

InternetGatewayId: !Ref InternetGateway

VpcId: !Ref VPC

PublicRouteTable:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name

Value: Public Subnets

- Key: Network

Value: Public

PrivateRouteTable01:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name

Value: Private Subnet AZ1

- Key: Network

Value: Private01

PrivateRouteTable02:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name

Value: Private Subnet AZ2

- Key: Network

Value: Private02

PublicRoute:

DependsOn: VPCGatewayAttachment

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PublicRouteTable

DestinationCidrBlock: 0.0.0.0/0

GatewayId: !Ref InternetGateway

PrivateRoute01:

DependsOn:

- VPCGatewayAttachment

- NatGateway01

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PrivateRouteTable01

DestinationCidrBlock: 0.0.0.0/0

NatGatewayId: !Ref NatGateway01

PrivateRoute02:

DependsOn:

- VPCGatewayAttachment

- NatGateway02

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PrivateRouteTable02

DestinationCidrBlock: 0.0.0.0/0

NatGatewayId: !Ref NatGateway02

NatGateway01:

DependsOn:

- NatGatewayEIP1

- PublicSubnet01

- VPCGatewayAttachment

Type: AWS::EC2::NatGateway

Properties:

AllocationId: !GetAtt 'NatGatewayEIP1.AllocationId'

SubnetId: !Ref PublicSubnet01

Tags:

- Key: Name

Value: !Sub '\${AWS::StackName}-NatGatewayAZ1'

NatGateway02:

DependsOn:

- NatGatewayEIP2

- PublicSubnet02

- VPCGatewayAttachment

Type: AWS::EC2::NatGateway

Properties:

AllocationId: !GetAtt 'NatGatewayEIP2.AllocationId'
SubnetId: !Ref PublicSubnet02
Tags:
- Key: Name
Value: !Sub '\${AWS::StackName}-NatGatewayAZ2'

NatGatewayEIP1:
DependsOn:
- VPCGatewayAttachment
Type: 'AWS::EC2::EIP'
Properties:
Domain: vpc

NatGatewayEIP2:
DependsOn:
- VPCGatewayAttachment
Type: 'AWS::EC2::EIP'
Properties:
Domain: vpc

PublicSubnet01:
Type: AWS::EC2::Subnet
Metadata:
Comment: Subnet 01
Properties:
MapPublicIpOnLaunch: true
AvailabilityZone:
Fn::Select:
- '0'
- Fn::GetAZs:
Ref: AWS::Region
CidrBlock:
Ref: PublicSubnet01Block
VpcId:
Ref: VPC
Tags:
- Key: Name
Value: !Sub "\${AWS::StackName}-PublicSubnet01"
- Key: kubernetes.io/role/elb
Value: 1

PublicSubnet02:
Type: AWS::EC2::Subnet
Metadata:
Comment: Subnet 02
Properties:
MapPublicIpOnLaunch: true
AvailabilityZone:

```
Fn::Select:
- '1'
- Fn::GetAZs:
  Ref: AWS::Region
CidrBlock:
  Ref: PublicSubnet02Block
VpcId:
  Ref: VPC
Tags:
- Key: Name
  Value: !Sub "${AWS::StackName}-PublicSubnet02"
- Key: kubernetes.io/role/elb
  Value: 1
```

```
PrivateSubnet01:
  Type: AWS::EC2::Subnet
  Metadata:
    Comment: Subnet 03
  Properties:
    AvailabilityZone:
      Fn::Select:
        - '0'
        - Fn::GetAZs:
            Ref: AWS::Region
    CidrBlock:
      Ref: PrivateSubnet01Block
    VpcId:
      Ref: VPC
    Tags:
      - Key: Name
        Value: !Sub "${AWS::StackName}-PrivateSubnet01"
      - Key: kubernetes.io/role/internal-elb
        Value: 1
```

```
PrivateSubnet02:
  Type: AWS::EC2::Subnet
  Metadata:
    Comment: Private Subnet 02
  Properties:
    AvailabilityZone:
      Fn::Select:
        - '1'
        - Fn::GetAZs:
            Ref: AWS::Region
    CidrBlock:
      Ref: PrivateSubnet02Block
    VpcId:
      Ref: VPC
```

Tags:

- Key: Name
Value: !Sub "\${AWS::StackName}-PrivateSubnet02"
- Key: kubernetes.io/role/internal-elb
Value: 1

PublicSubnet01RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PublicSubnet01
RouteTableId: !Ref PublicRouteTable

PublicSubnet02RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PublicSubnet02
RouteTableId: !Ref PublicRouteTable

PrivateSubnet01RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PrivateSubnet01
RouteTableId: !Ref PrivateRouteTable01

PrivateSubnet02RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PrivateSubnet02
RouteTableId: !Ref PrivateRouteTable02

ControlPlaneSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Cluster communication with worker nodes
VpcId: !Ref VPC

Outputs:

SubnetIds:

Description: Subnets IDs in the VPC

Value: !Join [",", [!Ref PublicSubnet01, !Ref PublicSubnet02, !Ref PrivateSubnet01, !Ref PrivateSubnet02]]

SecurityGroups:

Description: Security group for the cluster control plane communication with worker nodes

Value: !Join [",", [!Ref ControlPlaneSecurityGroup]]

VpcId:

Description: The VPC Id
Value: !Ref VPC

Appendix B – Container deployment YAML

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: beta.kubernetes.io/arch
                    operator: In
```

```
values:
- amd64
containers:
- name: nginx
  image: nginx:1.19.2
ports:
- containerPort: 80
```

Appendix C – Installing helm

```
awsdlsladmin@ubun1802:~$ curl
https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 > get_helm.sh
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
           Dload Upload Total Spent Left  Speed
100 11391  100 11391    0    0 62245    0 --:--:-- --:--:-- --:--:-- 62245
awsdlsladmin@ubun1802:~$ chmod 700 get_helm.sh
awsdlsladmin@ubun1802:~$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.4.1-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
awsdlsladmin@ubun1802:~$ which helm
/usr/local/bin/helm
awsdlsladmin@ubun1802:~$ helm help
The Kubernetes package manager
```

Common actions for Helm:

- helm search: search for charts
- helm pull: download a chart to your local directory to view
- helm install: upload the chart to Kubernetes
- helm list: list releases of charts

Environment variables:

Name	Description
\$HELM_CACHE_HOME	set an alternative location for storing cached files.
\$HELM_CONFIG_HOME	set an alternative location for storing Helm configuration.
\$HELM_DATA_HOME	set an alternative location for storing Helm data.

AWS EKS

\$HELM_DEBUG	indicate whether or not Helm is running in Debug mode	
\$HELM_DRIVER	set the backend storage driver. Values are: configmap, secret, memory, postgres	
\$HELM_DRIVER_SQL_CONNECTION_STRING	set the connection string the SQL storage driver should use.	
\$HELM_MAX_HISTORY	set the maximum number of helm release history.	
\$HELM_NAMESPACE	set the namespace used for the helm operations.	
\$HELM_NO_PLUGINS	disable plugins. Set HELM_NO_PLUGINS=1 to disable plugins.	
\$HELM_PLUGINS	set the path to the plugins directory	
\$HELM_REGISTRY_CONFIG	set the path to the registry config file.	
\$HELM_REPOSITORY_CACHE	set the path to the repository cache directory	
\$HELM_REPOSITORY_CONFIG	set the path to the repositories file.	
\$KUBECONFIG	set an alternative Kubernetes configuration file (default "~/kube/config")	
\$HELM_KUBEAPISERVER	set the Kubernetes API Server Endpoint for authentication	
\$HELM_KUBEASGROUPS	set the Groups to use for impersonation using a comma-separated list.	
\$HELM_KUBEASUSER	set the Username to impersonate for the operation.	
\$HELM_KUBECONTEXT	set the name of the kubeconfig context.	
\$HELM_KUBETOKEN	set the Bearer KubeToken used for authentication.	

Helm stores cache, configuration, and data based on the following configuration order:

- If a HELM_*_HOME environment variable is set, it will be used
- Otherwise, on systems supporting the XDG base directory specification, the XDG variables will be used
- When no other location is set a default location will be used based on the operating system

By default, the default directories depend on the Operating System. The defaults are listed below:

Operating System	Cache Path	Configuration Path	Data Path	
-----	-----	-----	-----	

```
| Linux      | $HOME/.cache/helm      | $HOME/.config/helm      |  
$HOME/.local/share/helm |  
| macOS      | $HOME/Library/Caches/helm | $HOME/Library/Preferences/helm |  
$HOME/Library/helm      |  
| Windows     | %TEMP%\helm             | %APPDATA%\helm           | %APPDATA%\helm  
|
```

Usage:

```
helm [command]
```

Available Commands:

```
completion generate autocompletions script for the specified shell  
create      create a new chart with the given name  
dependency  manage a chart's dependencies  
env         helm client environment information  
get         download extended information of a named release  
help        Help about any command  
history     fetch release history  
install     install a chart  
lint        examine a chart for possible issues  
list        list releases  
package     package a chart directory into a chart archive  
plugin       install, list, or uninstall Helm plugins  
pull        download a chart from a repository and (optionally) unpack it in local directory  
repo        add, list, remove, update, and index chart repositories  
rollback    roll back a release to a previous revision  
search      search for a keyword in charts  
show        show information of a chart  
status      display the status of the named release  
template    locally render templates  
test        run tests for a release  
uninstall   uninstall a release  
upgrade     upgrade a release  
verify      verify that a chart at the given path has been signed and is valid  
version     print the client version information
```

Flags:

```
--debug          enable verbose output  
-h, --help       help for helm  
--kube-apiserver string the address and the port for the Kubernetes API server  
--kube-as-group stringArray Group to impersonate for the operation, this flag can be  
repeated to specify multiple groups.  
--kube-as-user string Username to impersonate for the operation  
--kube-context string name of the kubeconfig context to use
```

```
--kube-token string      bearer token used for authentication
--kubeconfig string      path to the kubeconfig file
-n, --namespace string    namespace scope for this request
--registry-config string  path to the registry config file (default
"/home/awards1admin/.config/helm/registry.json")
--repository-cache string path to the file containing cached repository indexes (default
"/home/awards1admin/.cache/helm/repository")
--repository-config string path to the file containing repository names and URLs (default
"/home/awards1admin/.config/helm/repositories.yaml")
```

Use "helm [command] --help" for more information about a command.
awards1admin@ubun1802:~\$

Appendix D – Installing metrics server for Prometheus

```
awards1admin@ubun1802:~$ kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.3.6/components.yaml
awards1admin@ubun1802:~$ kubectl get deployment metrics-server -n kube-system
awards1admin@ubun1802:~$ kubectl get --raw /metrics
awards1admin@ubun1802:~$ kubectl create namespace Prometheus
awards1admin@ubun1802:~$ helm repo add stable https://charts.helm.sh/stable
awards1admin@ubun1802:~$ helm install prometheus stable/prometheus --namespace
prometheus --set
alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume.storageClass="gp
2"
awards1admin@ubun1802:~$ kubectl get pods -n prometheus
NAME                                READY STATUS  RESTARTS  AGE
prometheus-alertmanager-6b64586d49-csvdr  2/2  Running  0        28m
prometheus-kube-state-metrics-c65b87574-4vrl8  1/1  Running  0        28m
prometheus-node-exporter-8ghb8             1/1  Running  0        28m
prometheus-node-exporter-n589n             1/1  Running  0        28m
prometheus-pushgateway-7d5f5746c7-jl68m    1/1  Running  0        28m
prometheus-server-f8d46859b-grppr          2/2  Running  0        28m
awards1admin@ubun1802:~$ sudo ufw allow 9090
awards1admin@ubun1802:~$ kubectl --namespace=prometheus port-forward
deploy/prometheus-server 9090
```

References

[AWS Documentation Index Page](#)
[Kubernetes Cluster Networking](#)

[RESTful API](#)