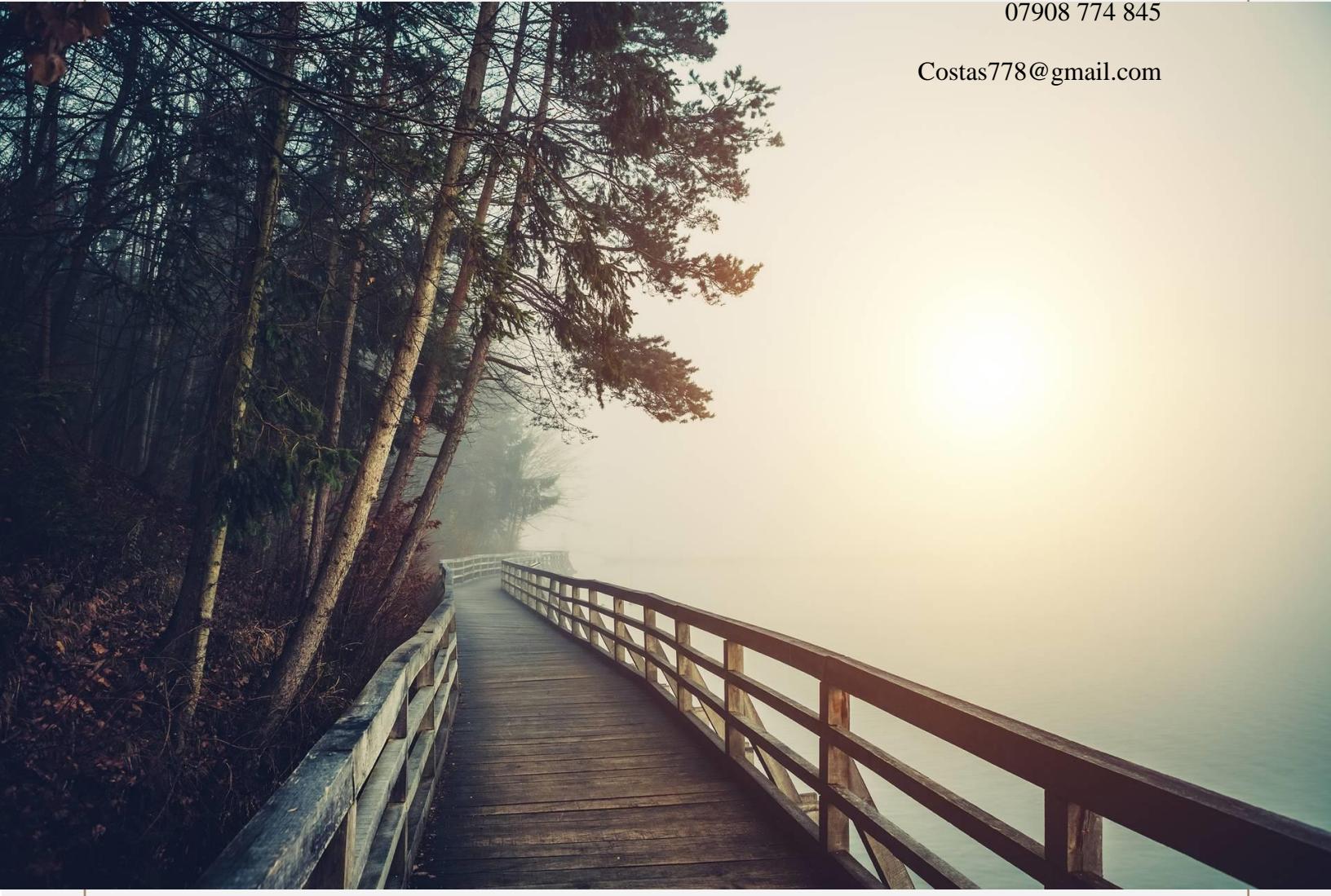


DEVOPS Projects

Costas Constantinou MSc

Version 2.5.5

A photograph of a wooden boardwalk with railings, leading through a dense forest of tall evergreen trees towards a misty body of water. The scene is bathed in soft, warm light, possibly from the rising or setting sun.

07908 774 845

Costas778@gmail.com

DEVOPS PROJECTS 2.5.5

What is this document for?

Having spent over 10 years in the IT Industry, I have noticed that there are, from time to time, new “flavors of the month”. This leads to shortages of skilled professionals. DEVOPS is no exception.

What never changes are the dilemma of sifting through hundreds (and even thousands) of applications claiming they can add value to either your organisation or corporate entity.

HR professionals and recruiters must sift through many applications that have all the ‘right words’ on their CV or covering letter. However, there are more than a few applicants that slip through that turn out to be “economical with the truth”. In other words, they cannot support, either by knowledge or experience, what is written by them in their documents.

Now it is true that many of these people are screened and rejected by a technical test. However, even this is no guarantee that they, genuinely, know their subject matter considering all the resources on the internet to help you prep in various technical disciplines!

As a result, I decided to come up with this DEVOPS projects document that demonstrates, practically, what I can do with the various DEVOPS tools. Within this document you will find various projects I have done. It will be a growing list and one that will be updated as technology never sleeps.

If you are a HR or recruiter simply attach this to my application and pass it on to others that are best placed to make an assessment. If you’re a fellow IT professional wanting to get up to speed on DEVOPS then this document may help you too. After all, the most effective learning is doing!

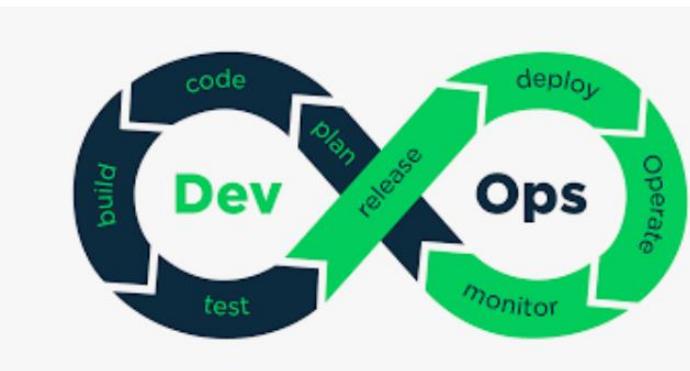


Table of Contents

[EKS Projects]

[Use Terraform to Create an EKS Deployment](#)

[Deploying a Kubernetes Cluster Using Elastic Kubernetes Service](#)

[Deploy Springboot Microservices App into Amazon EKS Cluster using Jenkins Pipeline and](#)

[Kubectl CLI Plug-in | Containerize Springboot App and Deploy into EKS Cluster using Jenkins](#)

[Pipeline](#)

[AKS Projects].....

[Create an AKS Cluster in Azure with Terraform](#)

[Deploying and Accessing an Application to an AKS Cluster](#)

[Jenkins Projects].....

[Deploying Jenkins Master and Worker Nodes in AWS Behind an ALB Using Terraform and Ansible](#)

[Deploying a Docker Container with Jenkins Pipelines](#)

[Deploy Python App into Kubernetes Cluster using Jenkins Pipeline | Containerize Python App and Deploy into AKS Cluster](#)

[Azure Pipeline].....

[Create and Test an ASP.NET Core App in Azure Pipelines](#)

[Deploy a Python App to an AKS Cluster Using Azure Pipelines](#)

[Docker].....

Load Balancing Containers (Docker Swarm and Docker Compose)

[Prometheus].....

Setting Up Prometheus and Adding Endpoints

Monitoring in Kubernetes with Prometheus and Grafana

How to setup monitoring on an EKS Kubernetes Cluster using both Prometheus and Grafana

[Ansible].....

Getting Started with Ansible

Ansible Playbooks: The Basics

Writing Your First Ansible Playbook

Deploying Services Using Ansible

Ad-Hoc Ansible Commands

Advanced Features in Ansible Playbooks

Working with Ansible Roles

[Git].....

Using Git for Source Control and Programming

Use Git to Make Changes to Code

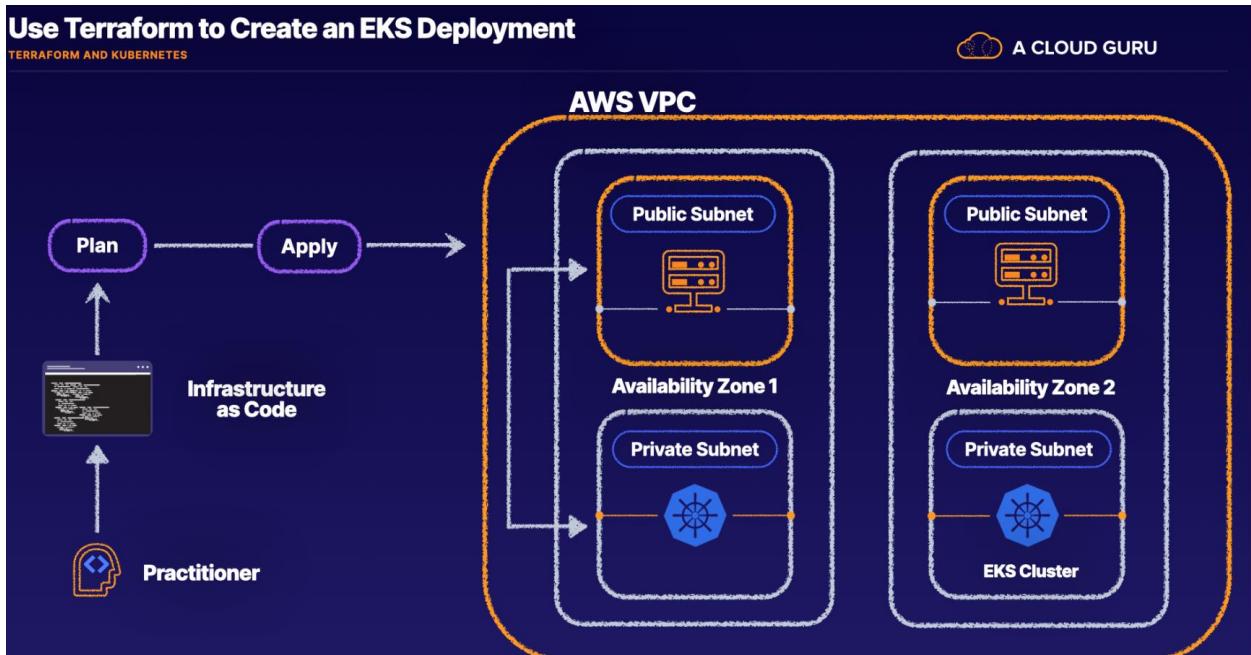
[AWS].....

Working with CloudFormation Nested Stacks

Create a DynamoDB Table Using CloudFormation

Deploy an Amazon RDS Multi-AZ and Read Replica in AWS

[EKS Projects] — Use Terraform to Create an EKS Deployment



Solution:

Log in to the server using the credentials provided:

```
Host '3.80.152.91' is not in the trusted hosts file.  
(ssh-ed25519 fingerprint shall! 58:fd:99:ca:9a:f3:7a:6e:2a:32:9b:2c:37:0e:35:8c:63:0c:e8:f2)  
Do you want to continue connecting? (y/n) y  
cloud_user@3.80.152.91's password:  
Last login: Fri Sep 22 09:45:08 2023
```

```
Amazon Linux 2 AMI  
https://aws.amazon.com/amazon-linux-2/
```

ssh cloud_user@<PUBLIC_IP_ADDRESS>

Configure your AWS CLI:

aws configure

When prompted for your AWS Access Key ID, copy, and paste in the **Access Key**.

When prompted for your AWS Secret Access Key, copy, and paste in the **Secret Access Key**.

Press **Enter** to accept the **default** region.

Press **Enter** to accept the **default** output.

NOTE: if you don't have the above details, go to **IAM**, and generate the above.

```
[cloud_user@ip-10-0-1-201 ~]$ aws configure
AWS Access Key ID [None]: AKIAZXMQHZEGXNLZ67HY
AWS Secret Access Key [None]: 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
Default region name [us-east-1]:
Default output format [None]:
[cloud_user@ip-10-0-1-201 ~]$ █
```

Review the Contents of the Configuration Files in the lab-terraform-eks Directory:

View the contents of the directory:

Ls

Go into the lab-terraform-eks-2 directory:

cd lab-terraform-eks-2/

View the contents of the lab-terraform-eks-2 directory:

Ls

```
[cloud_user@ip-10-0-1-201 ~]$ ls
lab-terraform-eks-2  lab-terraform-eks-2.zip  resource_ids.txt
[cloud_user@ip-10-0-1-201 ~]$ cd lab-terraform-eks-2/
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ ls
main.tf  outputs.tf  terraform.tf  variables.tf
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ vi terraform.tf █
```

Open the terraform.tf file:

vim terraform.tf

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 4.47.0"
    }
    kubernetes = {
      source = "hashicorp/kubernetes"
      version = ">= 2.16.1"
    }
    random = {
      source = "hashicorp/random"
      version = "~> 3.4.3"
    }
    tls = {
```

```

    source = "hashicorp/tls"
    version = "~> 4.0.4"
}
}
required_version = "~> 1.3"
}

```

Review the contents of the file, and then enter :q! to exit the file.

Open the main.tf file:

vim main.tf

```

provider "aws" {
  region = var.region
}
data "aws_availability_zones" "available" {}
locals {
  cluster_name = "guru-eks-${random_string.suffix.result}"
}
resource "random_string" "suffix" {
  length = 8
  special = false
}
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"
  version = "3.19.0"
  name = "guru-vpc"
  cidr = "10.0.0.0/16"
  azs = slice(data.aws_availability_zones.available.names, 0, 3)
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets = ["10.0.4.0/24", "10.0.5.0/24", "10.0.6.0/24"]
  enable_nat_gateway = true
  single_nat_gateway = true
  enable_dns_hostnames = true
  public_subnet_tags = {
    "kubernetes.io/cluster/${local.cluster_name}" = "shared"
    "kubernetes.io/role/elb" = 1
  }
  private_subnet_tags = {
    "kubernetes.io/cluster/${local.cluster_name}" = "shared"
    "kubernetes.io/role/internal-elb" = 1
  }
}

```

Review the contents of the file, and then enter :q! to exit the file.

Open the variables.tf file:

vim variables.tf

```
variable "region" {
  description = "AWS region"
  type       = string
  default    = "us-east-1"
}
```

Review the contents of the file, and then enter :q! to exit the file.

Open the outputs.tf file:

vim outputs.tf

```
output "cluster_endpoint" {
  description = "Endpoint for EKS control plane"
  value      = module.eks.cluster_endpoint
}
output "cluster_security_group_id" {
  description = "Security group ids attached to the cluster control plane"
  value      = module.eks.cluster_security_group_id
}
output "region" {
  description = "AWS region"
  value      = var.region
}
output "cluster_name" {
  description = "Kubernetes Cluster Name"
  value      = module.eks.cluster_name
}
```

Review the contents of the file, and then enter :q! to exit the file.

NOTE: to recreate this Terraform project use the above code to create the files required.

```
[cloud_user@ip-10-0-1-201 ~]$ ls
lab-terraform-eks-2  lab-terraform-eks-2.zip  resource_ids.txt
[cloud_user@ip-10-0-1-201 ~]$ cd lab-terraform-eks-2/
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ ls
main.tf  outputs.tf  terraform.tf  variables.tf
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ vi terraform.tf
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ vi main.tf
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ vi variables.tf
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ vi outputs.tf
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ 
```

Deploy the EKS Cluster

Initialize the working directory (and download the appropriate plugins):

terraform init

Review the actions that will be performed and check for any potential errors:

```
terraform has been successfully initialized!
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
run this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ 
```

Check to see if the provisioning plan of what you wish to deploy is acceptable to Terraform with the following command.

terraform plan

Now apply the configuration and deploy your cluster:

terraform apply

Type "**yes**" and hit **Enter** to confirm. Allow the configuration to complete successfully and create your resources, including the EKS cluster.

```
aws_eksAddon.ebs-csi: Still creating... [14m40s elapsed]
aws_eksAddon.ebs-csi: Still creating... [14m50s elapsed]
aws_eksAddon.ebs-csi: Still creating... [15m0s elapsed]
aws_eksAddon.ebs-csi: Creation complete after 15m8s [id=guru-eks-jxyiEf3V:aws-ebs-csi-driver]

Apply complete! Resources: 58 added, 0 changed, 0 destroyed.

Outputs:

cluster_endpoint = "https://3830E9EF67E279F402DEEAAD9AC100DA.gr7.us-east-1.eks.amazonaws.com"
cluster_name = "guru-eks-jxyiEf3V"
cluster_security_group_id = "sg-0657edeb6b12b8238"
region = "us-east-1"
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ 
```

Note: This can take around 10 minutes or more to complete so please be patient.

The **Output.tf** gives us the following:

```
cluster_endpoint = "https://3830E9EF67E279F402DEEAAD9AC100DA.gr7.us-east-1.eks.amazonaws.com"
cluster_name = "guru-eks-jxyiEf3V"
cluster_security_group_id = "sg-0657edeb6b12b8238"
region = "us-east-1"
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ 
```

Configure kubectl to interact with the cluster:

```
aws eks --region ${terraform output -raw region} update-kubeconfig --name ${terraform output -raw cluster_name}
```

```
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ aws eks --region ${terraform output -raw region} update-kubeconfig --name ${terraform output -raw cluster_name}
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$
```

Confirm that kubectl was configured properly and that the cluster was successfully deployed:

```
kubectl get cs
```

All the components should be up and running with a status of Healthy as below.

```
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ kubectl get cs
Warning: v1 ComponentStatus is deprecated in v1.19+
NAME      STATUS  MESSAGE           ERROR
etcd-1    Healthy {"health": "true", "reason": ""}
etcd-0    Healthy {"health": "true", "reason": ""}
etcd-2    Healthy {"health": "true", "reason": ""}
controller-manager  Healthy   ok
scheduler  Healthy   ok
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$
```

Deploy the NGINX Pods

Now download the **nginx.tf** file that contains the configuration for the NGINX deployments from the GitHub repo:

```
wget https://raw.githubusercontent.com/ACloudGuru-Resources/content-terraform-2021/main/nginx.tf
```

Confirm that the file was downloaded to the directory successfully:

```
ls
```

Open the nginx.tf file:

```
vim nginx.tf
```

```
data "terraform_remote_state" "eks" {
  backend = "local"

  config = {
    path = "../lab-terraform-eks-2/terraform.tfstate"
  }
}
```

```

data "aws_eks_cluster" "cluster" {
  name = data.terraform_remote_state.eks.outputs.cluster_name
}

provider "kubernetes" {
  host           = data.aws_eks_cluster.cluster.endpoint
  cluster_ca_certificate = base64decode(data.aws_eks_cluster.cluster.certificate_authority.0.data)
  exec {
    api_version = "client.authentication.k8s.io/v1beta1"
    command     = "aws"
    args = [
      "eks",
      "get-token",
      "--cluster-name",
      data.aws_eks_cluster.cluster.name
    ]
  }
}

resource "kubernetes_deployment" "nginx" {
  metadata {
    name = "long-live-the-bat"
    labels = {
      App = "LongLiveTheBat"
    }
  }

  spec {
    replicas = 2
    selector {
      match_labels = {
        App = "LongLiveTheBat"
      }
    }
    template {
      metadata {
        labels = {
          App = "LongLiveTheBat"
        }
      }
      spec {
        container {
          image = "nginx:1.7.8"
          name = "batman"

          port {
            container_port = 80

```

```
resources {
    limits = {
        cpu   = "0.5"
        memory = "512Mi"
    }
    requests = {
        cpu   = "250m"
        memory = "50Mi"
    }
}
}
}
}
}
```

Review the contents of the file, and then enter :q! to exit the file.

```
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ wget https://raw.githubusercontent.com/ACloudGuru-Resources/content-terraform-2021/main/nginx.tf
--2023-09-22 11:09:19-- https://raw.githubusercontent.com/ACloudGuru-Resources/content-terraform-2021/main/nginx.tf
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1401 (1.4K) [text/plain]
Saving to: 'nginx.tf'

100%[=====] 1,401

2023-09-22 11:09:19 (22.5 MB/s) - 'nginx.tf' saved [1401/1401]

[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ ls
main.tf  nginx.tf  outputs.tf  terraform.tf  terraform.tfstate  variables.tf
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ vi nginx.tf
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ terraform plan
```

Review the actions that will be performed and check for any potential errors:

Since we are updating the infrastructure, we need to run the same commands again.
terraform plan

Apply the configuration and deploy your NGINX pods to your EKS cluster:

terraform apply

Type "yes" and hit Enter to confirm. Allow the configuration to complete successfully and create your resources, including the NGINX pods.

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

cluster_endpoint = "https://3830E9EF67E279F402DEEAAD9AC100DA.gr7.us-east-1.eks.amazonaws.com"
cluster_name = "guru-eks-jxyiEf3V"
cluster_security_group_id = "sg-0657edeb6b12b8238"
region = "us-east-1"
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ █
```

Confirm that the NGINX pods were successfully deployed:

kubectl get deployments

The two long-live-the-bat pods should both be up and running.

```
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ kubectl get deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
long-live-the-bat   2/2      2          2           111s
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ █
```

Destroy Your Cluster

Before you destroy the infrastructure, view the resources Terraform is managing:

terraform state list

A whole list of resources will appear.

Destroy all of the resources you just created:

terraform destroy

Type "yes" and hit Enter to confirm.

```
random_string.suffix: Destroying... [id=jxyiEf3V]
random_string.suffix: Destruction complete after 0s
module.vpc.aws_vpc.this[0]: Destroying... [id=vpc-0
module.vpc.aws_vpc.this[0]: Destruction complete af

Destroy complete! Resources: 59 destroyed.
[cloud_user@ip-10-0-1-201 lab-terraform-eks-2]$ █
```

Confirm the resources were all destroyed:

terraform state list

Nothing should be returned, confirming that all resources were properly destroyed.

Addendum

To recreate the project we went through we will need to take the following pre requisite steps beforehand.

Create and connect to a new instance on AWS (hence we will need an AWS account).

Create an Admin User Role within IAM

Upgrade AWS CLI to version 2 (for AMI distros)

Install the AWS CLI into that directory

Install and configure Terraform on an EC2 instance

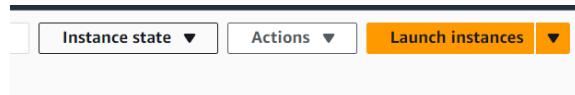
Install Kubectl

Create and Connect to New Instance

In the search bar on top, enter **EC2** and select EC2 from the search results to access EC2.

Select **Instances (running)**.

Click the **Launch instances** button.



On the Launch an instance button, set the following parameters:

Name and tags: Enter **eks-admin**.

Application and OS Images (Amazon Machine Image): Select **Amazon Linux**.

Key pair name: Select **Create new key pair**.

On the Create key pair page, under Key pair name, enter **eks-key**.

Click the **Create key pair** button to create the key pair.

Create key pair

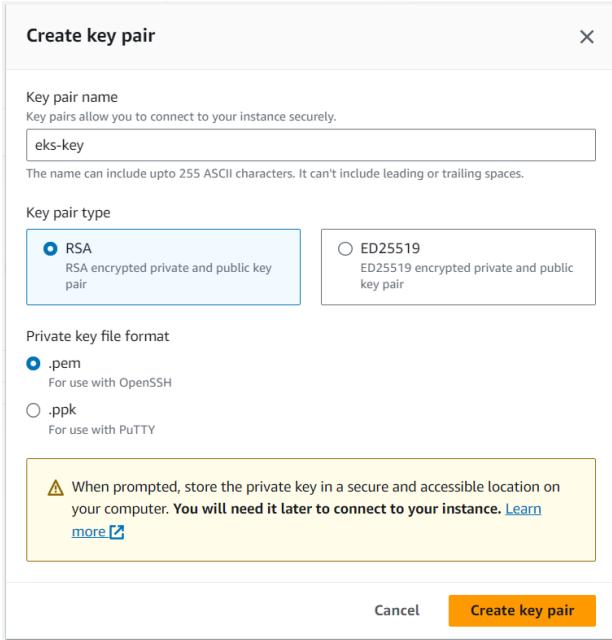
Key pair name
Key pairs allow you to connect to your instance securely.
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type
 RSA RSA encrypted private and public key pair
 ED25519 ED25519 encrypted private and public key pair

Private key file format
 .pem For use with OpenSSH
 .ppk For use with PuTTY

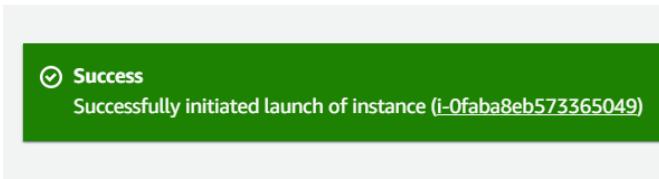
⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)

[Cancel](#) [Create key pair](#)



Click the **Launch instance** button.

Under **Success**, select the link to the instance to see its status. It can take a little while for the instance to launch successfully and for the Instance state to change from Pending to Running.

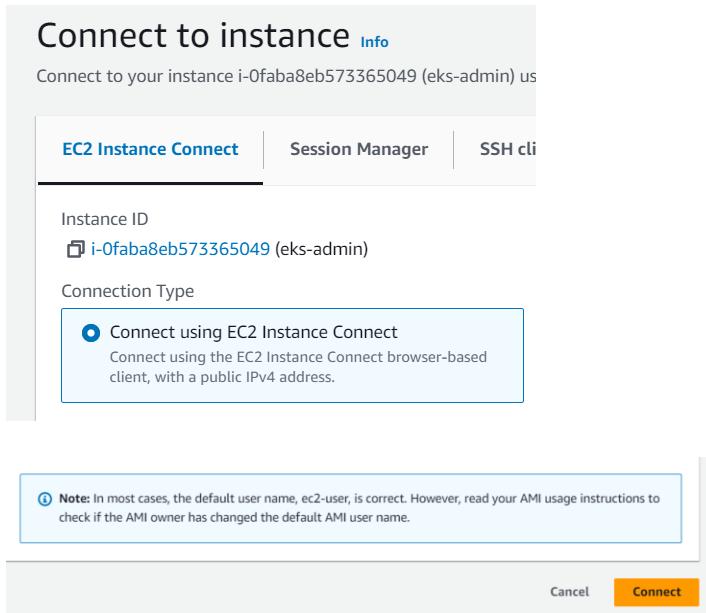


Click the checkbox next to the eks-admin instance to select it.

Click the **Connect** button.

Instances (1/1) Info		C	Connect
<input type="text"/> Find instance by attribute or tag (case-sensitive)			
<input type="text" value="Instance ID = i-0faba8eb573365049"/> X	Clear filters		
Name	Instance ID	Instance state	
<input checked="" type="checkbox"/> eks-admin	i-0faba8eb573365049	Running 	

On the Connect to instance page, click the **Connect** button.



Create an Admin User Role within IAM

Under Favourites, select **IAM**. If you don't see a Favourites section, you can enter **IAM** in the search bar on top of the console and select **IAM** from the search results to access the **IAM** dashboard.

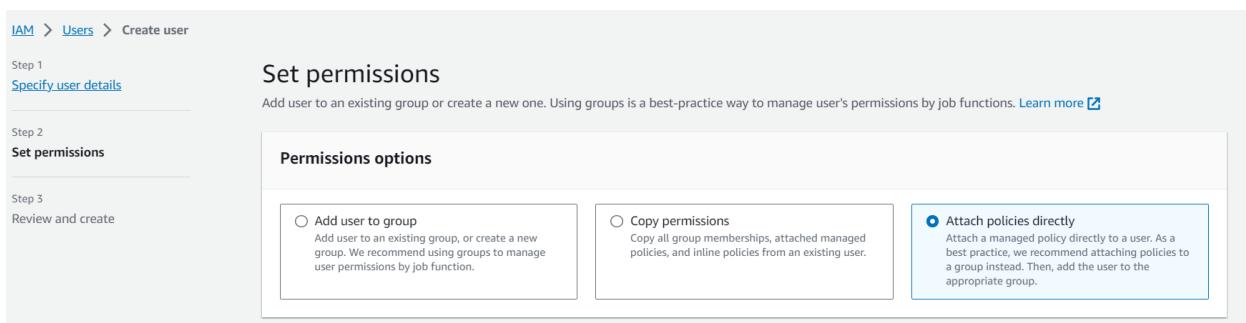
In the lefthand navigation menu, select **Users**.

Click the **Add Users** button.

On the Specify user details page, under Username, enter **eks-user**.

Click the **Next** button.

On the Set permissions page, under Permissions options, select **Attach policies directly**.



Under Permissions policies, click the checkbox next to **AdministratorAccess** to grant admin access to the role being created.

Policy name	Type
<input type="checkbox"/> <input checked="" type="checkbox"/>  AccessAnalyzerServiceRolePolicy	AWS managed
<input checked="" type="checkbox"/> <input type="checkbox"/>  AdministratorAccess	AWS managed - job function
<input type="checkbox"/> <input type="checkbox"/>  AdministratorAccess	AWS managed

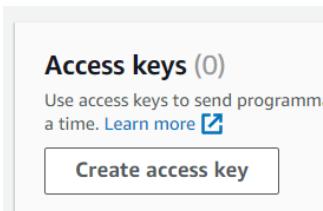
Click the **Next** button.

On the Review and create page, click the **Create user** button.

Once the user is created, select **eks-user**.

Select the **Security credentials** tab.

Under **Access keys**, click the **Create access key** button.



On the Access key best practices & alternatives page, select **Command Line Interface (CLI)**.

Use case

- Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

Click the checkbox next to **I understand the recommendation and want to proceed to create an access key**.

Click the **Next** button.

On the Set description tag - optional page, click the **Create access key** button.

Click the **Download .csv** file.

[Upgrade AWS CLI to version 2](#)

[Download AWS CLI 2](#)

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"
```

[Unzip the downloaded CLI 2 file:](#)

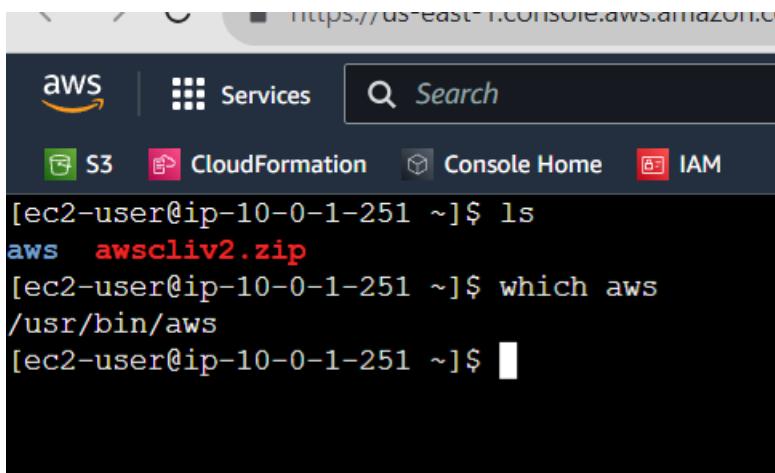
unzip awscli2.zip

This can take a few minutes to unzip and create everything needed.

Check the location of our AWS files:

which aws

You should see they are located in **usr/bin/aws**.



```
aws      | Services   | Search
S3     CloudFormation  Console Home  IAM
[ec2-user@ip-10-0-1-251 ~]$ ls
aws  awscli2.zip
[ec2-user@ip-10-0-1-251 ~]$ which aws
/usr/bin/aws
[ec2-user@ip-10-0-1-251 ~]$
```

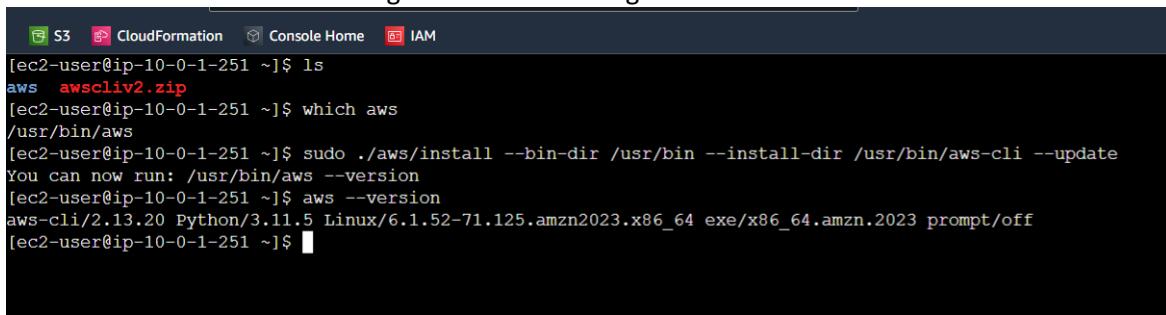
Install the AWS CLI into that directory:

sudo ./aws/install --bin-dir /usr/bin --install-dir /usr/bin/aws-cli --update

Re-check which version of AWS is being used:

aws --version

You should see we are now using AWS CLI 2.11 or higher



```
S3     CloudFormation  Console Home  IAM
[ec2-user@ip-10-0-1-251 ~]$ ls
aws  awscli2.zip
[ec2-user@ip-10-0-1-251 ~]$ which aws
/usr/bin/aws
[ec2-user@ip-10-0-1-251 ~]$ sudo ./aws/install --bin-dir /usr/bin --install-dir /usr/bin/aws-cli --update
You can now run: /usr/bin/aws --version
[ec2-user@ip-10-0-1-251 ~]$ aws --version
aws-cli/2.13.20 Python/3.11.5 Linux/6.1.52-71.125.amzn2023.x86_64 exe/x86_64.amzn.2023 prompt/off
[ec2-user@ip-10-0-1-251 ~]$
```

Install and configure Terraform on an EC2 instance:

```
[cloud_user@ip-10-0-1-201 ~]$ terraform version  
Terraform v1.5.7  
on linux_amd64  
[cloud_user@ip-10-0-1-201 ~]$ █
```

sudo yum update -y

wget https://releases.hashicorp.com/terraform/1.5.7/terraform_1.5.7_linux_amd64.zip

unzip terraform_1.5.7_linux_amd64.zip

sudo mv terraform /usr/local/bin/

terraform --version

rm terraform_1.5.7_linux_amd64.zip

Install Kubectl

curl -LO "<https://dl.k8s.io/release/v1.21.2/bin/linux/amd64/kubectl>"

chmod +x kubectl

sudo mv kubectl /usr/local/bin/

kubectl version --client

rm kubectl

[cloud_user@ip-10-0-1-201 ~]\$ kubectl version

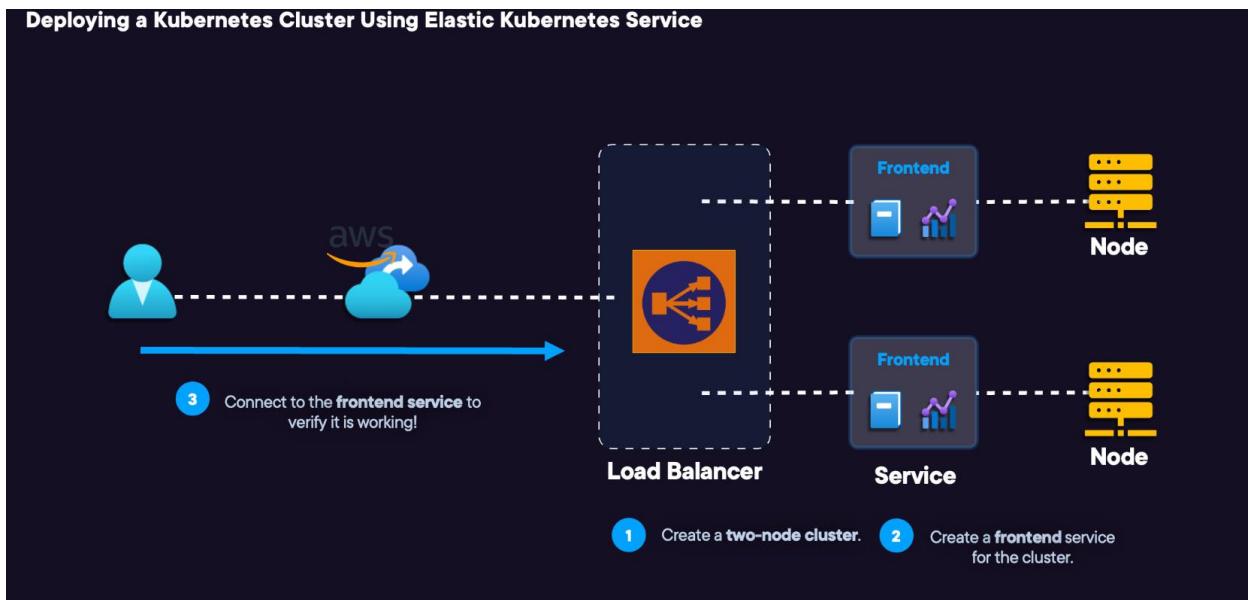
Client Version: version.Info{Major:"1", Minor:"21+", GitVersion:"v1.21.2-13+d2965f0db10712",
GitCommit:"d2965f0db1071203c6f5bc662c2827c71fc8b20d", GitTreeState:"clean", BuildDate:"2021-06-
26T01:02:11Z", GoVersion:"go1.16.5", Compiler:"gc", Platform:"linux/amd64"}

The connection to the server localhost:8080 was refused -

did you specify the right host or port?

[cloud_user@ip-10-0-1-201 ~]\$

Deploying a Kubernetes Cluster Using Elastic Kubernetes Service



Solution

Log into the AWS Management Console.

Note: you will need to have an AWS account prior to starting.

Note: Make sure you're using the us-east-1 Region.

Create an EKS Cluster in the us-east-1 Region:

Create an Admin User Role:

Under Favourites, select **IAM**. If you don't see a Favourites section, you can enter **IAM** in the search bar on top of the console and select **IAM** from the search results to access the **IAM** dashboard.

In the lefthand navigation menu, select **Users**.

Click the **Add Users** button.

On the Specify user details page, under Username, enter **eks-user**.

Click the **Next** button.

On the **Set permissions** page, under Permissions options, select **Attach policies directly**.

IAM > Users > Create user

Step 1 Specify user details

Step 2 Set permissions

Step 3 Review and create

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Under Permissions policies, click the checkbox next to **AdministratorAccess** to grant admin access to the role being created.

Policy name	Type
<input type="checkbox"/> AccessAnalyzerServiceRolePolicy	AWS managed
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function
<input type="checkbox"/> AdministratorAccess - Analytics	AWS managed

Click the **Next** button.

On the Review and create page, click the **Create user** button.

Once the user is created, select **eks-user**.

Select the **Security credentials** tab.

Under **Access keys**, click the **Create access key** button.

Access keys (0)

Use access keys to send programmatic requests at any time. [Learn more](#)

Create access key

On the Access key best practices & alternatives page, select **Command Line Interface (CLI)**.

Use case

- Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

Click the checkbox next to **I understand the recommendation and want to proceed to create an access key**.

Click the **Next** button.

On the Set description tag - optional page, click the **Create access key** button.

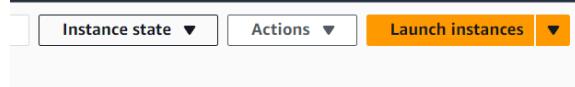
Click the **Download .csv** file.

Create and Connect to New Instance:

In the search bar on top, enter **EC2** and select **EC2** from the search results to access EC2.

Select **Instances (running)**.

Click the **Launch instances** button.



On the Launch an instance button, set the following parameters:

Name and tags: Enter **eks-admin**.

Application and OS Images (Amazon Machine Image): Select **Amazon Linux**.

Key pair name: Select **Create new key pair**.

On the Create key pair page, under Key pair name, enter **eks-key**.

Click the **Create key pair** button to create the key pair.

A screenshot of the 'Create key pair' dialog box. It has a title bar 'Create key pair' with a close button 'X'.

- Key pair name:** A text input field containing 'eks-key'. A note below says: 'Key pairs allow you to connect to your instance securely.' and 'The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.'
- Key pair type:** A radio button group. The 'RSA' option is selected, with the note: 'RSA encrypted private and public key pair'. The 'ED25519' option is also available, with the note: 'ED25519 encrypted private and public key pair'.
- Private key file format:** A radio button group. The '.pem' option is selected, with the note: 'For use with OpenSSH'. The '.ppk' option is also available, with the note: 'For use with PuTTY'.
- Warning message:** A yellow-bordered box contains a warning: '⚠ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)'.
- Buttons:** 'Cancel' and 'Create key pair' (highlighted in orange).

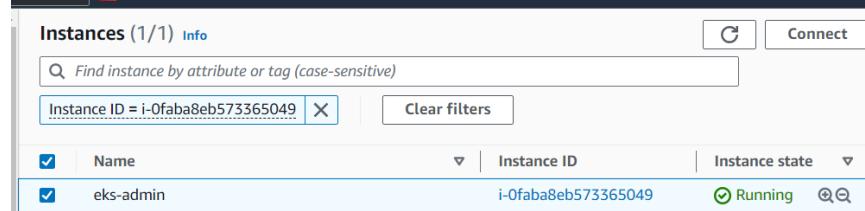
Click the **Launch instance** button.

Under **Success**, select the link to the instance to see its status. It can take a little while for the instance to launch successfully and for the Instance state to change from Pending to Running.

 **Success**
Successfully initiated launch of instance ([i-0faba8eb573365049](#))

Click the checkbox next to the eks-admin instance to select it.

Click the **Connect** button.



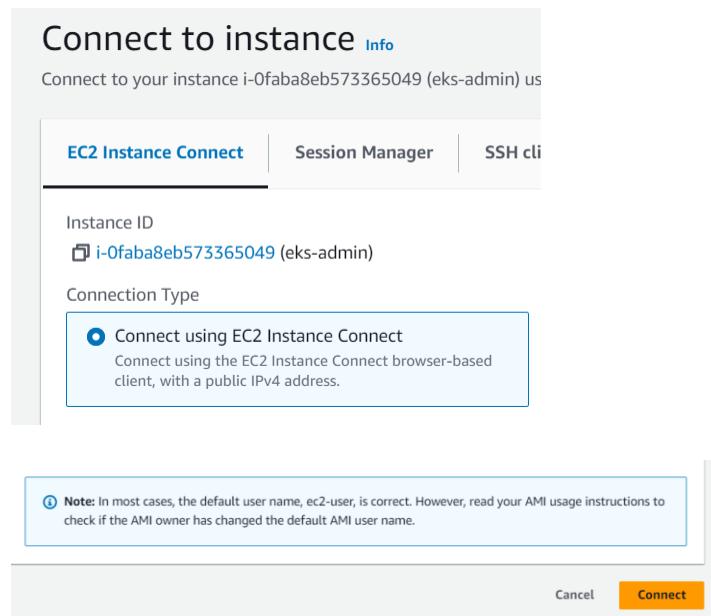
Instances (1/1) [Info](#)

Find instance by attribute or tag (case-sensitive)

Instance ID = i-0faba8eb573365049

Name	Instance ID	Instance state
<input checked="" type="checkbox"/> eks-admin	i-0faba8eb573365049	 Running 

On the Connect to instance page, click the **Connect** button.



Connect to instance [Info](#)

Connect to your instance i-0faba8eb573365049 (eks-admin) us

EC2 Instance Connect Session Manager SSH cli

Instance ID [i-0faba8eb573365049 \(eks-admin\)](#)

Connection Type

Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

Note: In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel

Create Cluster

Check the version of the CLI being used to verify whether it is NOT AWS CLI 2:

aws --version

You should see that the version being used is AWS CLI 1.18 or above.

The screenshot shows a terminal window within the AWS CloudShell interface. The title bar indicates the URL is <https://us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0faba8eb573365049&osUser>. The terminal displays the following text:

```
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-10-0-1-251 ~]$ aws --version
aws-cli/2.9.19 Python/3.9.16 Linux/6.1.52-71.125.amzn2023.x86_64 source/x86_64.amzn.2023 prompt/off
[ec2-user@ip-10-0-1-251 ~]$
```

For this project we will need to upgrade our AWS CLI to version 2.

Note: if AWS CLI does NOT exist at all then follow these steps as well.

Download AWS CLI 2:

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

Unzip the downloaded CLI 2 file:

```
unzip awscliv2.zip
```

This can take a few minutes to unzip and create everything needed.

Check the location of our AWS files:

```
which aws
```

You should see they are located in **usr/bin/aws**.

The screenshot shows a terminal window within the AWS CloudShell interface. The title bar indicates the URL is <https://us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0faba8eb573365049&osUser>. The terminal displays the following text:

```
[ec2-user@ip-10-0-1-251 ~]$ ls
aws  awscliv2.zip
[ec2-user@ip-10-0-1-251 ~]$ which aws
/usr/bin/aws
[ec2-user@ip-10-0-1-251 ~]$
```

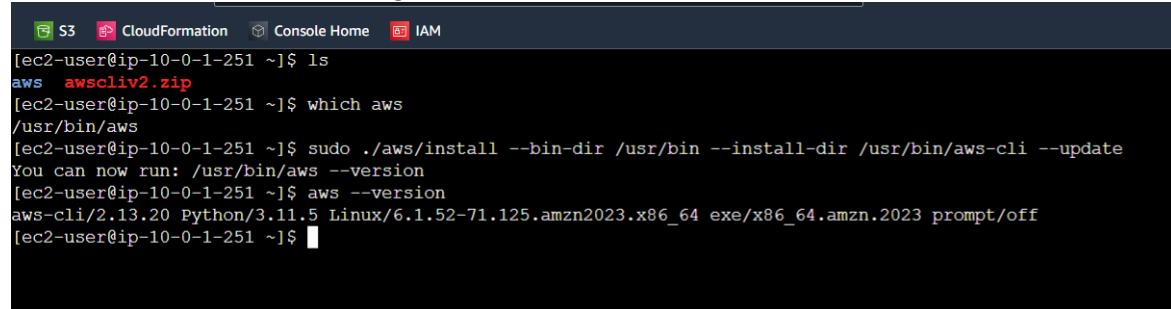
Install the AWS CLI into the destination directory:

```
sudo ./aws/install --bin-dir /usr/bin --install-dir /usr/bin/aws-cli --update
```

Re-check which version of AWS is being used:

aws –version

You should see we are now using AWS CLI 2.11 or above.



```
S3 CloudFormation Console Home IAM
[ec2-user@ip-10-0-1-251 ~]$ ls
aws awscli2.zip
[ec2-user@ip-10-0-1-251 ~]$ which aws
/usr/bin/aws
[ec2-user@ip-10-0-1-251 ~]$ sudo ./aws/install --bin-dir /usr/bin --install-dir /usr/bin/aws-cli --update
You can now run: /usr/bin/aws --version
[ec2-user@ip-10-0-1-251 ~]$ aws --version
aws-cli/2.13.20 Python/3.11.5 Linux/6.1.52-71.125.amzn2023.x86_64 exe/x86_64.amzn.2023 prompt/off
[ec2-user@ip-10-0-1-251 ~]$
```

Log into our AWS environment:

aws configure

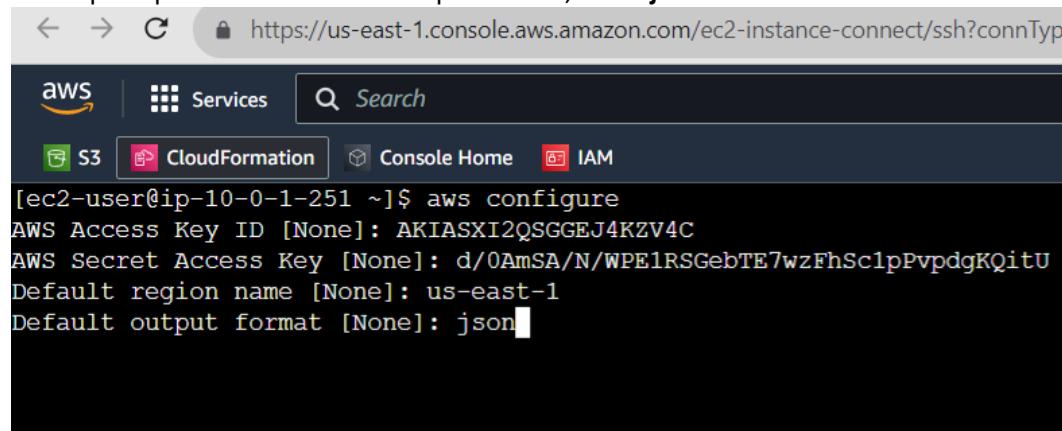
From the CSV file created earlier, copy and paste the **Access Key ID** when prompted.

From the same CSV file, copy and paste the **Secret Access Key** when prompted.

A	B	C	D	E	F	G
Access key ID	Secret access key					
AKIASXI2QSGGEJ4KZV4C	d/0AmSA/N/WPE1RSGebTE7wzFhSc1pPvpdgKQitU					

When prompted for the default region name, enter **us-east-1**.

When prompted for the default output format, enter **json**.



```
← → C https://us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connTyp
aws Services Search
S3 CloudFormation Console Home IAM
[ec2-user@ip-10-0-1-251 ~]$ aws configure
AWS Access Key ID [None]: AKIASXI2QSGGEJ4KZV4C
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
```

Note: In a few hours this infrastructure will be gone with the key pair credentials!

Download kubectl:

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.6/2023-01-30/bin/linux/amd64/kubectl
```

Add execution permissions to the file:

```
chmod +x ./kubectl
```

Create a new bin directory and copy kubectl to the bin directory and home directory:

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
```

Download eksctl:

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

This will install the CLI tool you will use to create a cluster on Amazon EKS.

Move the files to usr/local/bin:

```
sudo mv /tmp/eksctl /usr/local/bin
```

Check the eksctl version:

```
eksctl version
```

This should show the version as 0.133.0 or higher!

```
[ec2-user@ip-10-0-1-251 ~]$ curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.6/2023-01-30/bin/linux/amd64/kubectl
$ Total    % Received % Xferd  Average Speed   Time   Time Current
          Dload Upload   Total Spent   Left Speed
100 42.9M  100 42.9M    0     0  17.2M  0:00:02  0:00:02  --:-- 17.2M
[ec2-user@ip-10-0-1-251 ~]$ chmod +x ./kubectl
[ec2-user@ip-10-0-1-251 ~]$ mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
[ec2-user@ip-10-0-1-251 ~]$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
[ec2-user@ip-10-0-1-251 ~]$ sudo mv /tmp/eksctl /usr/local/bin
[ec2-user@ip-10-0-1-251 ~]$ eksctl version
0.157.0
```

Create the cluster:

```
eksctl create cluster --name myeks --region us-east-1 --zones=us-east-1a,us-east-1b --nodegroup-name eks-workers --node-type t3.medium --nodes-min 2 --nodes-max 4 --managed
```

This may take several minutes to finish creating the cluster.

```
[ec2-user@ip-10-0-1-251 ~]$ eksctl create cluster --name myeks --region us-east-1 --zones=us-east-1a,us-east-1b --nodegroup-name eks-workers --node-type t3.medium --nodes-min 2 --nodes-max 4 --managed
2023-09-21 20:29:49 [i] eksctl version 0.157.0
2023-09-21 20:29:49 [i] using region us-east-1
2023-09-21 20:29:49 [i] subnets for us-east-1a - public:192.168.0.0/19 private:192.168.64.0/19
2023-09-21 20:29:49 [i] subnets for us-east-1b - public:192.168.32.0/19 private:192.168.96.0/19
2023-09-21 20:29:50 [i] nodegroup "eks-workers" will use "" (AmazonLinux/1.25)
2023-09-21 20:29:50 [i] using Kubernetes version 1.25
2023-09-21 20:29:50 [i] creating EKS cluster "myeks" in "us-east-1" region with managed nodes
2023-09-21 20:29:50 [i] Will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2023-09-21 20:29:50 [i] If you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=myeks'
2023-09-21 20:29:50 [i] Kubernetes API endpoint access will use default of [publicAccess=true, privateAccess=false] for cluster "myeks" in "us-east-1"
2023-09-21 20:29:50 [i] CloudWatch logging will not be enabled for cluster "myeks" in "us-east-1"
2023-09-21 20:29:50 [i] You can enable it with 'eksctl utils update-cluster-logging --enable-types=[SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)] --region=us-east-1 --cluster=myeks'
2023-09-21 20:29:50 [i] sequential tasks: ( create cluster control plane "myeks",
2023-09-21 20:29:50 [i]   2 sequential sub-tasks: (
2023-09-21 20:29:50 [i]     wait for control plane to become ready,
2023-09-21 20:29:50 [i]     create managed nodegroup "eks-workers",
2023-09-21 20:29:50 [i]   )
2023-09-21 20:29:50 [i]   building cluster stack "eksctl-myeks-cluster"
2023-09-21 20:29:50 [i]   deploying stack "eksctl-myeks-cluster"
```

Stack name	Status	Created time	Description
eksctl-myeks-cluster	CREATE_COMPLETE	2023-09-21 23:29:50 UTC+0300	EKS cluster (dedicated VPC: true, dedicated IAM: true) [created and managed by eksctl]
cfst-4038-af246102f34a0759f74dd7d062a0fe4b	CREATE_COMPLETE	2023-09-21 21:00:40 UTC+0300	Creates a VPC with (3) public subnets allowing ssh

Once the cluster is created, connect to the cluster:

aws eks update-kubeconfig --name myeks --region us-east-1

Verify that it is running eksctl:

eksctl get cluster

You should see that it is running.

Deploy a Test Application to Mimic the Application

Install the Git package:

sudo yum install git -y

Clone a repository from the EKS Basics course:

git clone https://github.com/costas778/Course_EKS-Basics

Change into the Course_EKS-Basics directory:

cd Course_EKS-Basics/

View the contents of the directory:

_ls

You should see a license file, an nginx-deployment.yaml file, and an nginx-svc.yaml file.

```
[ec2-user@ip-10-0-1-251 ~]$ git clone https://github.com/costas778/Course_EKS-Basics
Cloning into 'Course_EKS-Basics'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 21 (delta 8), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (21/21), 5.37 KiB | 1.34 MiB/s, done.
Resolving deltas: 100% (8/8), done.
[ec2-user@ip-10-0-1-251 ~]$ cd Course_EKS-Basics/
[ec2-user@ip-10-0-1-251 Course_EKS-Basics]$ ls
LICENSE  nginx-deployment.yaml  nginx-svc.yaml
[ec2-user@ip-10-0-1-251 Course_EKS-Basics]$ █
```

View the nginx-deployment.yaml file:

cat nginx-deployment.yaml

View the nginx-svc.yaml file:

cat nginx-svc.yaml

Apply the nginx-svc.yaml file:

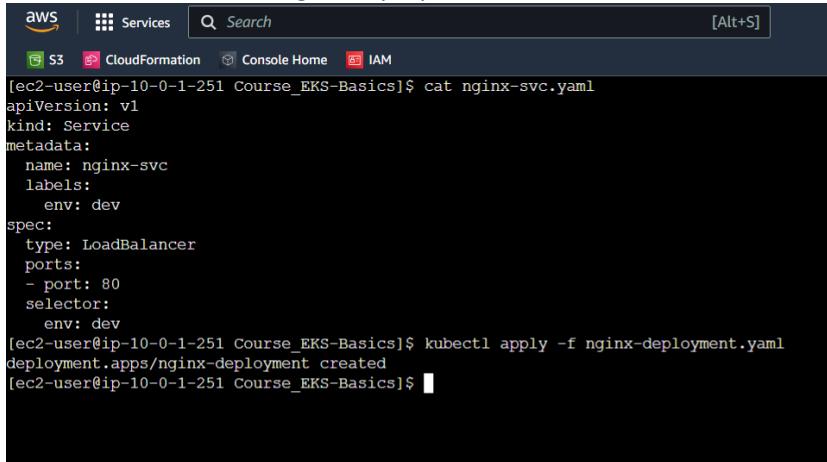
kubectl apply -f nginx-svc.yaml

This should create the Nginx service.

Apply the nginx-deployment.yaml file:

kubectl apply -f nginx-deployment.yaml

This should create the Nginx deployments.



```
aws | Services | Search [Alt+S]
S3 CloudFormation Console Home IAM

[ec2-user@ip-10-0-1-251 Course_EKS-Basics]$ cat nginx-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
  labels:
    env: dev
spec:
  type: LoadBalancer
  ports:
  - port: 80
    selector:
      env: dev
[ec2-user@ip-10-0-1-251 Course_EKS-Basics]$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
[ec2-user@ip-10-0-1-251 Course_EKS-Basics]$
```

Verify that the deployments are available:

kubectl get deployment

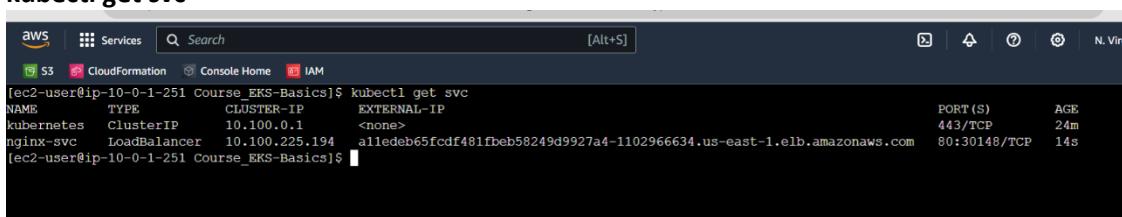
You should see that all three deployments are available.

Verify that all Pods are running:

kubectl get pods

Verify that the service is running:

kubectl get svc



```
aws | Services | Search [Alt+S] N. Virg
S3 CloudFormation Console Home IAM

[ec2-user@ip-10-0-1-251 Course_EKS-Basics]$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes   ClusterIP   10.100.0.1      <none>           443/TCP     24m
nginx-svc   LoadBalancer 10.100.225.194  a11edeb65fcdf481fbeb58249d9927a4-1102966634.us-east-1.elb.amazonaws.com  80:30148/TCP  14s
[ec2-user@ip-10-0-1-251 Course_EKS-Basics]$
```

You should see above that the load balancer has a DNS name.

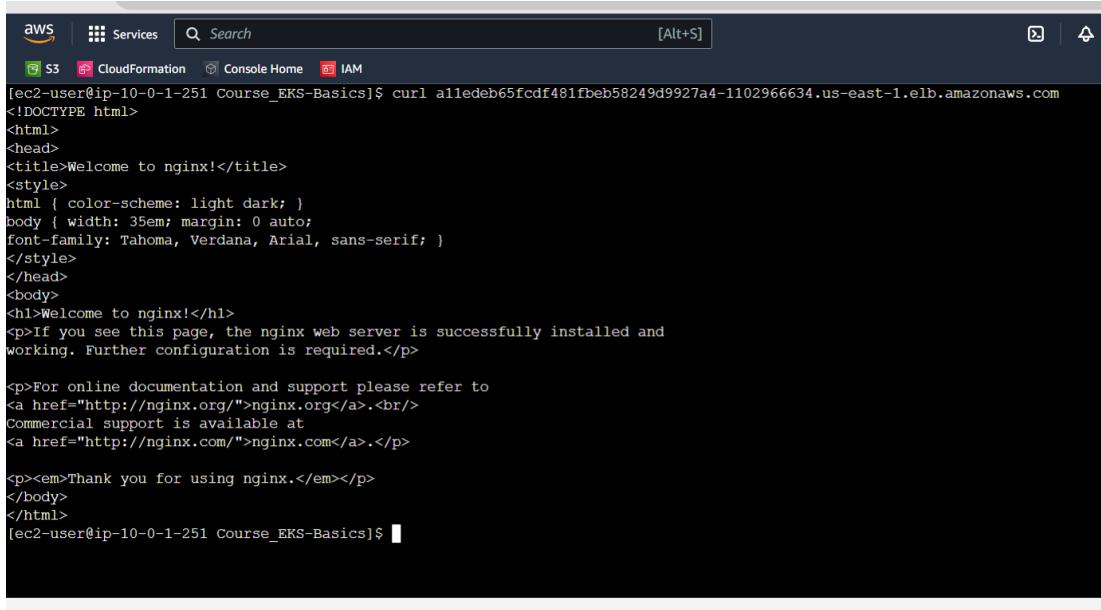
Use the DNS Name of the Load Balancer to Test the Cluster:

Copy the DNS name of the load balancer.

Use the DNS name to test that the cluster is working:

curl <PASTE DNS NAME HERE>

curl a11edeb65fcdf481fbef58249d9927a4-1102966634.us-east-1.elb.amazonaws.com



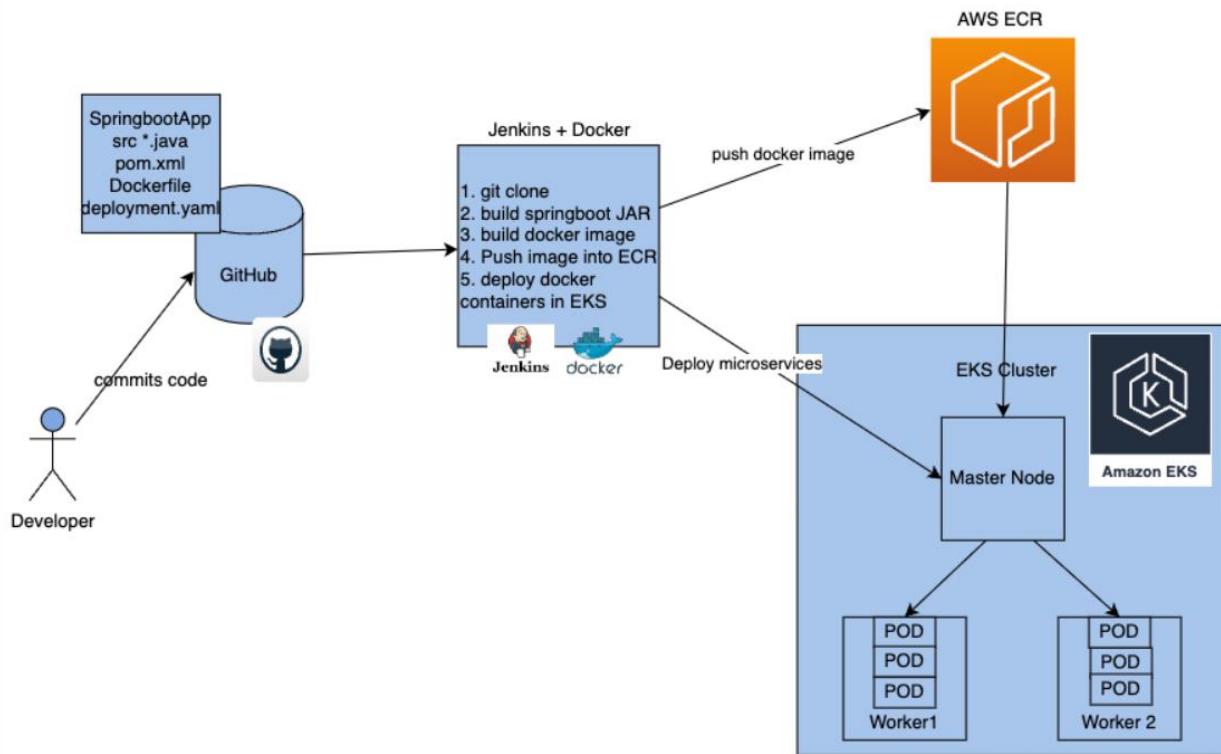
The screenshot shows a terminal window within the AWS CloudWatch interface. The terminal prompt is [ec2-user@ip-10-0-1-251 Course_EKS-Basics]\$. The user has run the command curl a11edeb65fcdf481fbef58249d9927a4-1102966634.us-east-1.elb.amazonaws.com. The terminal output displays the Nginx welcome page, which includes HTML code for the title, style, head, body, h1, p, and em tags, along with links to nginx.org and nginx.com. The output ends with a closing bracket and a cursor.

```
[ec2-user@ip-10-0-1-251 Course_EKS-Basics]$ curl a11edeb65fcdf481fbef58249d9927a4-1102966634.us-east-1.elb.amazonaws.com
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[ec2-user@ip-10-0-1-251 Course_EKS-Basics]$
```

You should see the test page for the Nginx service.

Deploy Springboot Microservices App into Amazon EKS Cluster using Jenkins Pipeline and Kubectl CLI Plug-in | Containerize Springboot App and Deploy into EKS Cluster using Jenkins Pipeline

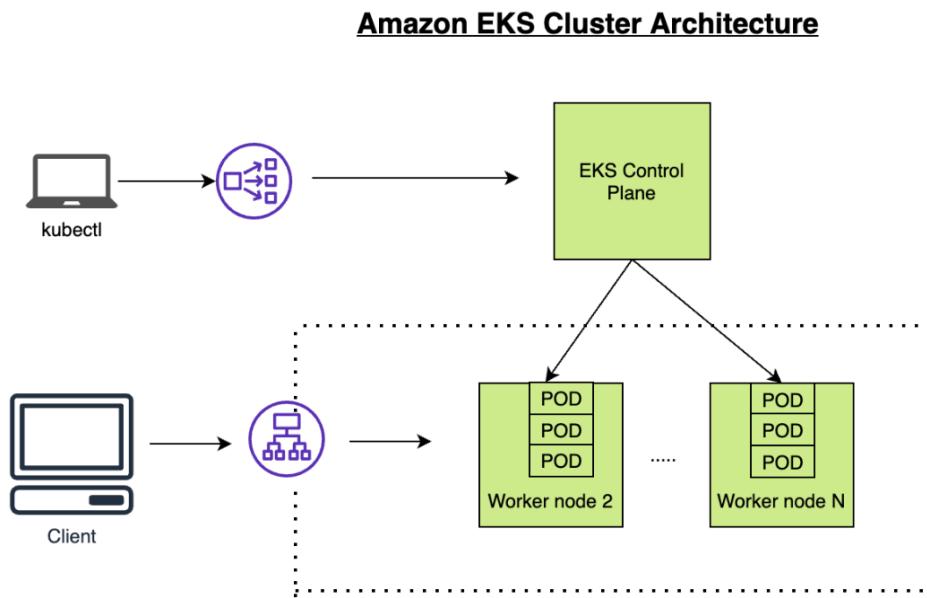
Springboot Microservices Deployment into EKS Cluster using Jenkins Pipeline



Pre-requisites:

1. Amazon EKS Cluster is setup and running.
2. Create ECR repo in AWS
3. Jenkins Master is up and running
4. Docker installed on Jenkins instance
5. Docker, Docker pipeline and Kubernetes CLI plug-ins are installed in Jenkins

Setup Amazon EKS Cluster



To help achieve this we need to first setup a Jenkins **EC2 instance**. This **EC2 instance**, that will be an **Ubuntu 22.4 LTS** server, needs to have following both installed and configured:

- Install AWS CLI** – Command line tools for working with AWS services, including Amazon EKS.
- Install eksctl** – A command line tool for working with EKS clusters that automates many individual tasks.
- Install kubectl** – A command line tool for working with Kubernetes clusters.

Create the Ubuntu instance.

Go to EC2 and launch an instance.

The screenshot shows the AWS EC2 'Create New Instance' wizard. In the 'Name and tags' step, the name 'Jenkins' is entered in the 'Name' field. Below it, there's a section for 'Application and OS Images (Amazon Machine Image)'. A search bar allows you to search for AMIs. Under 'Quick Start', several OS options are listed: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and SUSI. A 'Browse more AMIs' link is available. The selected AMI is 'Ubuntu Server 22.04 LTS (HVM), SSD Volume Type' (ami-053b0d53c279acc90). The instance is marked as 'Free tier eligible'. The 'Description' section notes it's Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-05-16. The 'Architecture' is set to '64-bit (x86)'. The 'AMI ID' is 'ami-053b0d53c279acc90' and it has a 'Verified provider' badge. In the 'Instance type' step, the 't2.medium' instance type is selected. It has a family of t2, 2 vCPU, 4 GiB Memory, and is current generation. On-Demand Linux base pricing is 0.0464 USD per Hour. The 'All generations' checkbox is checked. A note says 'Additional costs apply for AMIs with pre-installed software'.

Give it the name **Jenkins**, choose Ubuntu under **Quick Start** and select a **t2.medium** instance type.

Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.

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

Key pair type

RSA
RSA encrypted private and public key pair

ED25519
ED25519 encrypted private and public key pair

Private key file format

.pem
For use with OpenSSH

.ppk
For use with PuTTY

⚠ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)

[Cancel](#) [Create key pair](#)

On the same page choose to create a key pair and click **Create key pair**.

Create the instance, then once created navigate to EC2 and locate the Jenkins instance you just created security group.

EC2 > Security Groups > sg-085e8f18246ebc44b8 - launch-wizard-1 > Edit inbound rules

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
sgr-0654a505485c1a4e2	SSH	TCP	22	Custom	<input type="text" value="0.0.0.0/0"/> Delete
-	Custom TCP	TCP	8080	Any... Info	<input type="text" value="0.0.0.0/0"/> Delete

[Add rule](#) [Cancel](#) [Preview changes](#) [Save rules](#)

Create a new inbound rule of port **8080** with **0.0.0.0/0** to give access to the Jenkins server from anywhere. Click **Save Rules**.

Click the **Connect** button within **EC2**.

Use the key pair to login into the Ubuntu instance:

 **Command copied**

 `ssh -i "Jenkins.pem" ubuntu@ec2-54-146-82-200.compute-1.amazonaws.com`

Copy the above link into a command prompt on your local workstation that has the **Jenkins.pem** key pair file.

Note: Ensure that you run the above command in the same directory as the **Jenkins.pem** file. Otherwise, you will have to alter the above command to reflect the path of the **Jenkins.pem** file.

Please follow the steps to install **Java**, **Jenkins**, and **Maven** on Ubuntu 22.0.4 instance. **Jenkins** and, **Maven** are Java based applications, so we need to install Java first.

Change Host Name to Jenkins

sudo hostnamectl set-hostname Jenkins

Note: if you have already named the server Jenkins you can skip this step.

Perform update first

sudo apt update

Install Java 11

sudo apt install default-jdk -y

```
ubuntu@ip-172-31-44-89:~$ sudo apt install default-jdk -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  at-spi2-core default-jdk-headless default-jre fonts-dejavu-extra libatk-bridge2.0-0
  libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0
  libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontenc1 libgif7 libgl1
  libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libice-dev libice6
  liblomm10 libpciaccess0 libpthread-stubs0-dev libsensors4 libsm-dev libsm6 libx11-dev
  libx11-doc libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0
  libxcb-present0 libxcb-shape0 libxcb-sync1 libxcb1-dev libcomposite1 libxdamage1
```

Once install java, enter the below command

Verify Java Version

java -version

```
ubuntu@ip-172-31-19-22:~/opt$ java -version
openjdk version "11.0.6" 2020-01-14
OpenJDK Runtime Environment (Build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1)
OpenJDK 64-Bit Server VM (Build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1, mixed mode, sharing)
ubuntu@ip-172-31-19-22:~/opt$
```

Maven Installation

You can install Maven by executing below command:

```
sudo apt install maven -y
```

you can type **mvn -version**

you should see the below output.

```
ubuntu@ip-172-31-34-93:~$ mvn -version
Apache Maven 3.6.0
Maven home: /usr/share/maven
Java version: 11.0.7, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-1865-aws", arch: "amd64", family: "unix"
ubuntu@ip-172-31-34-93:~$
```

Note: you can skip the Java step and just install Maven. This is because Maven will do the job of installing Java for you.

Now let's start Jenkins installation.

Jenkins Setup

Add Repository key to the system

```
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

```
ubuntu@ip-172-31-28-138:~$ curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

Append debian package repo address to the system

```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
ubuntu@ip-172-31-28-138:~$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

Update Ubuntu package

sudo apt update

```
ubuntu@ip-172-31-37-211:~$ sudo apt-get update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu xenial-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:5 http://ppa.launchpad.net/webupd8team/java/ubuntu xenial InRelease
Ign:6 http://pkg.jenkins.io/debian-stable binary/ InRelease
Get:7 http://pkg.jenkins.io/debian-stable binary/ Release [2,042 B]
Get:8 http://pkg.jenkins.io/debian-stable binary/ Release.gpg [181 B]
Get:9 http://pkg.jenkins.io/debian-stable binary/ Packages [13.2 kB]
Fetched 15.5 kB in 0s (17.8 kB/s)
Reading package lists... Done
```

Install Jenkins

sudo apt install jenkins -y

```
ubuntu@ip-172-31-37-211:~$ sudo apt-get install jenkins -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  daemon
The following NEW packages will be installed:
  daemon jenkins
0 upgraded, 2 newly installed, 0 to remove and 8 not upgraded.
Need to get 72.0 MB of additional disk space will be used.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu xenial/universe amd64 daemon [71.9 MB]
Get:2 http://pkg.jenkins.io/debian-stable binary/ jenkins 2.121.3 [71.9 MB]
Fetched 72.0 MB in 5s (12.5 MB/s)
Selecting previously unselected package daemon.
(Reading database ... 51843 files and directories currently installed.)
Preparing to unpack .../daemon_0.6.4-1_amd64.deb ...
Unpacking daemon (0.6.4-1) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.121.3_all.deb ...
Unpacking jenkins (2.121.3) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for systemd (229-4ubuntu21.4) ...
Processing triggers for ureadahead (0.100.0-19) ...
Setting up daemon (0.6.4-1) ...
Setting up Jenkins (2.121.3) ...
Processing triggers for systemd (229-4ubuntu21.4) ...
Processing triggers for ureadahead (0.100.0-19) ...
ubuntu@ip-172-31-37-211:~$
```

The above screenshot should confirm that Jenkins is successfully installed.

Access Jenkins in web browser

Now Go to AWS console. Click on EC2 then click on the running instances link.
Connect to your instance using its Public IP Address:

The screenshot shows the AWS EC2 Instances page. A specific instance, 'i-0fbfb784941fcf59b5 (Jenkins)', is selected. The instance summary panel displays the following details:

- Instance ID:** i-0fbfb784941fcf59b5 (Jenkins)
- IPv6 address:** -
- Hostname type:** IP name: ip-172-31-51-115.ec2.internal
- Answer private resource DNS name:** IPv4 (A)
- Auto-assigned IP address:** 54.146.82.200 [Public IP]

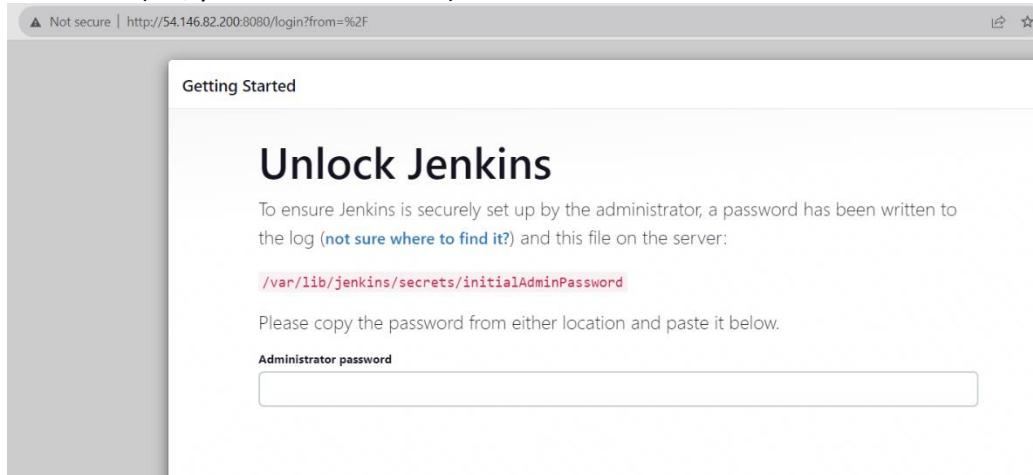
Now go to a browser and enter the **Auto-assigned (public) IP address** with port no **8080**.

http://Public IP Address:8080

It should load without issues after adding the inbound rule with the port 8080.

Unlock Jenkins

Since this is a fresh build, you will get the following screen. Enter the **cat** command below into the bash shell. (i.e., your Ubuntu console).



Get the initial password from the below file

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

```
ubuntu@ip-172-31-37-211:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Copy the password and paste it into the browser's text box under **Administrator password**. Move to the next page.

Click on **Install suggested plugins**.

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.



Also create an admin user.

We choose for our non-production server to use both the username **admin** and password **admin**.

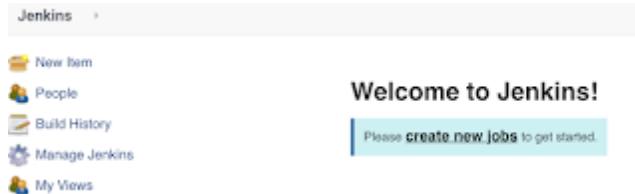
Create User

Username:	admin
Password:
Confirm password:
Full name:	admin
E-mail address:	admin@admin.com

Create User

Click **Create User**.

Click on **Save and Finish**. Click on **start using Jenkins**. Now you should see a screen like below:



Install the AWS CLI version 2 on Linux.

Follow these steps from the command line to install AWS CLI on Linux.

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

it should display the below output.

```
ubuntu@ip-172-31-24-217:~$ aws --version
aws-cli/2.2.32 Python/3.8.8 Linux/5.4.0-1045-aws exe/x86_64.ubuntu.18 prompt/off
ubuntu@ip-172-31-24-217:~$
```

Install eksctl on a Ubuntu Instance

eksctl is a command line tool for working with EKS clusters that automates many individual tasks. The **eksctl** tool uses **CloudFormation** under the hood, creating one stack for the EKS master control plane and another stack for the worker nodes.

In Amazon **Elastic Kubernetes Service (EKS)**, a **stack** typically refers to a collection of AWS **CloudFormation** resources that are used to provision and manage an EKS cluster and its associated infrastructure

Download and extract the latest release of **eksctl** with the following command.

```
curl --silent --location
```

```
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -
```

```
s)_amd64.tar.gz" | tar xz -C /tmp
```

Move the extracted binary to /usr/local/bin.

```
sudo mv /tmp/eksctl /usr/local/bin  
eksctl version
```

```
ubuntu@ip-172-31-17-55:~$ eksctl version  
0.92.0  
ubuntu@ip-172-31-17-55:~$
```

Install kubectl on Ubuntu Instance

Kubernetes uses a command line utility called **kubectl** for communicating with the cluster API server. It is tool for controlling **Kubernetes** clusters. **kubectl** looks for a file named config in the \$HOME directory.

Install kubectl

```
sudo curl --silent --location -o /usr/local/bin/kubectl https://s3.us-west-  
2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl
```

```
sudo chmod +x /usr/local/bin/kubectl
```

Verify if kubectl got installed

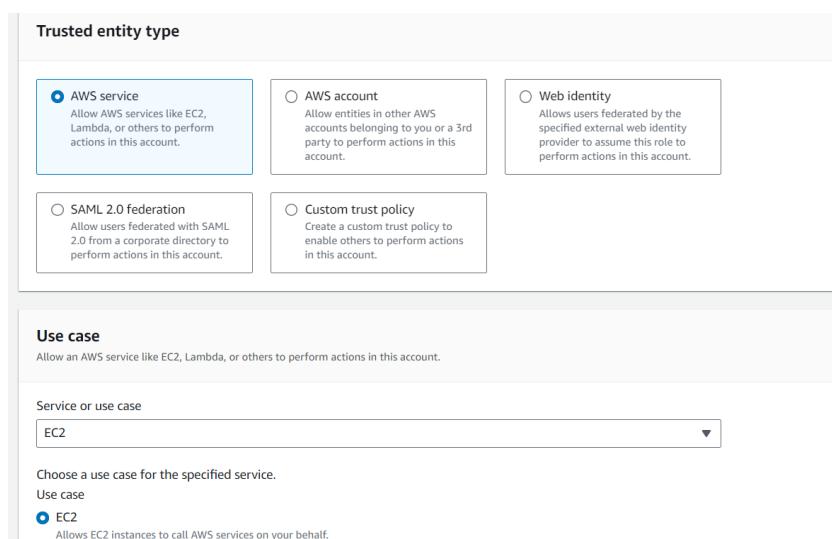
```
kubectl version --short --client
```

NOTE: You may get an error with Ubuntu with the above command. It may complain of deprecated command. If you find this, drop the **-short** switch and just use **-client**.

Its fine on AMI as of writing this!

Create an IAM Role with Administrator Access

You need to create an **IAM** role with **AdministratorAccess** policy.
Go to the AWS console, **IAM**, click on **Roles**. create a role.



Select AWS service.

Under **Use case**, select **EC2**.

Click Next

Add permissions [Info](#)

Permissions policies (1/879) [Info](#)

Choose one or more policies to attach to your new role.

Filter by Type

Search All types ▾

Policy name	Type
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function

Search for **AdministratorAccess**, select it and choose **Next**.

IAM > Roles > Create role

Step 1 Select trusted entity

Step 2 Add permissions

Step 3 Name, review, and create

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+-=,.@-_ ' characters.

Give it a name such as **Cluster-Admin-Role** and click **Create Role**.

Assign the role to EC2 instance

Go to AWS console, click on **EC2**, select **EC2 instance**, Choose **Security**.

Click on **Modify IAM Role**

Actions	Launch insta
Connect	
Instance state ▾	
Instance type ▾	
Stopped	t2.small
Stopped	t2.medium
Stopped	Change security groups
Stopped	Get Windows password
Stopped	Modify IAM role
Stopped	t2.small

Actions ▾ Launch insta

Connect

View details

Manage instance state

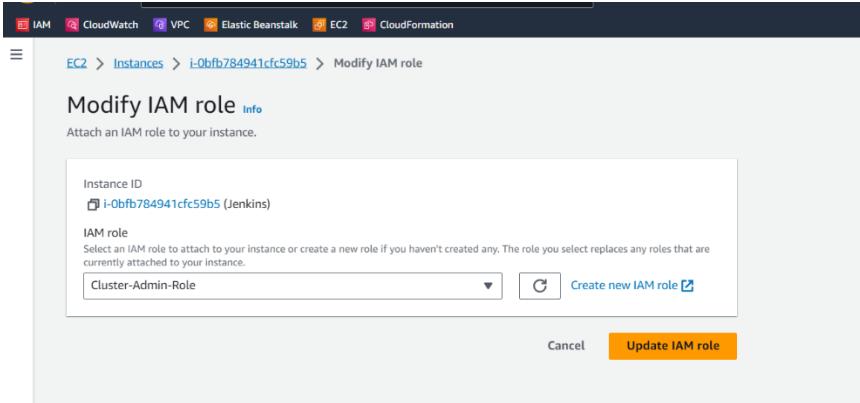
Instance settings ►

Networking ►

Security ►

Image and templates ►

Monitor and troubleshoot ►



Choose the role you have created from the dropdown.

Click on **Update IAM role**.

Instances (1/1) [Info](#)
Find instance by attribute or tag (case-sensitive)
Name Jenkins
Instance ID i-0fb784941cf59b5
Instance: i-0fb784941cf59b5 (Jenkins)
Security [Details](#) Networking Storage Status checks Monitoring
▼ Security details
IAM Role Cluster-Admin-Role Owner ID 642660919026

You can see from above that the IAM role is attached to the Jenkins instance.
Let's get back to the Ubuntu Instance.

Switch to Jenkins user

sudo su - jenkins

Create EKS Cluster with two worker nodes using eksctl

eksctl create cluster --name demo-eks --region us-east-1 --nodegroup-name my-nodes --node-type t3.small --managed --nodes 2

Note: if we didn't explicitly declare the -nodes 2 it will create two nodes by default.

The above command should create an EKS cluster in AWS, it might take 15 to 20 mins.

```

Mac-Silver-Coach:~ devopscoach$ eksctl create cluster --name demo-eks --region us-east-2 --nodegroup-name my-nodes
--node-type t3.small --managed
[ℹ] ubectl version v0.20.3
[ℹ]  using region us-east-2
[ℹ]  setting availability zones as [us-east-2c us-east-2b us-east-2a]
[ℹ]  subnets for us-east-2c - public:192.168.0.0/19 private:192.168.96.0/19
[ℹ]  subnets for us-east-2b - public:192.168.32.0/19 private:192.168.128.0/19
[ℹ]  subnets for us-east-2a - public:192.168.64.0/19 private:192.168.168.0/19
[ℹ]  using Kubernetes version 1.17
[ℹ]  creating the cluster "demo-eks" in "us-east-2" region with managed nodes
[ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
[ℹ]  if you encounter any issues, check CloudFormation console or try `eksctl utils describe-stacks --region=us-east-2 --cluster=demo-eks`
[ℹ]  CloudWatch Logging will not be enabled for cluster "demo-eks" in "us-east-2"
[ℹ]  you can enable it with `eksctl utils update-cluster-logging --enable-types=[SPECIFY-YOUR-LOG-TYPES-HERE] (e.g. --regions=us-east-2 --cluster=demo-eks)`
[ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "demo-eks" in "us-east-2"
[ℹ]  2 sequential sub-tasks: [ no tasks, create managed nodegroup "my-nodes" ] 1
[ℹ]  building cluster stack "eksctl-demo-eks-cluster"
[ℹ]  deploying stack "eksctl-demo-eks-cluster"
[ℹ]  building managed nodegroup stack "eksctl-demo-eks-nodegroup-my-nodes"
[ℹ]  deploying stack "eksctl-demo-eks-nodegroup-my-nodes"
[ℹ]  waiting for the control plane availability...
[ℹ]  saved kubeconfig as "/Users/devopscoach/.kube/config"
[ℹ]  no tasks
[ℹ]  all EKS cluster resources for "demo-eks" have been created
[ℹ]  nodegroup "my-nodes" has 2 nodes(s)
[ℹ]  node "ip-192-168-3-147.us-east-2.compute.internal" is ready
[ℹ]  node "ip-192-168-84-233.us-east-2.compute.internal" is ready
[ℹ]  waiting for at least 2 node(s) to become ready in "my-nodes"
[ℹ]  nodegroup "my-nodes" has 2 nodes(s)
[ℹ]  node "ip-192-168-3-347.us-east-2.compute.internal" is ready
[ℹ]  node "ip-192-168-84-233.us-east-2.compute.internal" is ready
[ℹ]  kubeconfig should work with `aws/eksctl/eksctl.kube/config` , try "eksctl get nodes"
[ℹ]  the cluster "demo-eks" is managed. Please be ready

```

After the cluster build run the following command:
eksctl get cluster --name demo-eks --region us-east-1

This should confirm that EKS cluster is up and running.

Update Kube config by entering below command:

aws eks update-kubeconfig --name demo-eks --region us-east-1

```

jenkins@ip-172-31-18-53:~ aws eks update-kubeconfig --name demo-eks --region us-east-2
added new context: arn:aws:eks:us-east-2:231223701595:cluster/demo-eks to /var/lib/jenkins/.kube/config

```

kubeconfig file be updated under the **/var/lib/jenkins/.kube** folder.
You can view the kubeconfig file by entering the below command:

cat /var/lib/jenkins/.kube/config

Create ECR repo in AWS

Amazon Elastic Container Registry (ECR) uses Amazon **S3** for storage to make your container images highly available and accessible, allowing you to reliably deploy new containers for your applications. Amazon ECR transfers your container images over HTTPS and automatically encrypts your images at rest. Amazon **ECR** is integrated with Amazon **Elastic Container Service (ECS)**, simplifying your development to production workflow.

Create Docker image & Push into Amazon ECR



Pre-requisites:

- **EC2** instance up and running with **Docker** installed.
- Make sure you open port **8081**.
- Install **aws cli**

Create a repo in ECR

Go to AWS console and search for ECR.

A screenshot of the AWS ECR service page. At the top, there's a navigation bar with 'ECR' selected. Below it, there are three main service links: 'Secrets Manager', 'Elastic Container Service', and 'Elastic Container Registry'. 'Elastic Container Registry' is highlighted with a light gray background. Underneath these links, there's a section titled 'IAM' with a single link. The entire interface is set against a white background with blue and black text.

Click on **Create Repository**



Enter a name for your repo - all lower case and click create repository.
We chose my-docker-repo

ECR > Repositories > Create repository

Create repository

Repository access and tags

Repository name
287301745136.dkr.ecr.us-east-1.amazonaws.com/**my-docker-repo**
A namespace can be included with your repository name (e.g., namespace/repository-name).

Tag immutability
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

Disabled

Image scan settings

Scan on push
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.

Disabled

[Cancel](#) [Create repository](#)

Once the repo is created, choose the repo and click on **View push commands**.

Note down the **account ID**

my-docker-repo

Successfully created repository my-docker-repo

ECR > Repositories

Repositories (1 of 1)			Create repository	View push commands
Repository name	URI	Created at		
my-docker-repo	287301745136.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo	05/21/20 AM		

Note the URL below, as this will be used for tagging and pushing docker images into ECR.

642660919026.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo

642660919026.dkr.ecr.us-east-1.amazonaws.com/my-python-repo

Push commands for mywebapp

[macOS / Linux](#) [Windows](#)

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry.
Use the AWS CLI:

```
aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
docker build -t mywebapp .
```
3. After the build completes, tag your image so you can push the image to this repository:

```
docker tag mywebapp:latest 642660919026.dkr.ecr.us-east-2.amazonaws.com/mywebapp
```
4. Run the following command to push this image to your newly created AWS repository:

Copy the above into a location for later reference.

Make sure that you have the latest version of AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry.
Use the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 642660919026.dkr.ecr.us-east-1.amazonaws.com
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
docker build -t my-docker-repo .
```

3. After the build completes, tag your image so you can push the image to this repository:

```
docker tag my-docker-repo:latest 642660919026.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
docker push 642660919026.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo:latest
```

That's it, you have created repo successfully. Let us now create docker images and push it to the above repo in ECR.

Docker installation steps using default repository from Ubuntu

Update local packages by executing below command:

sudo apt update

Install the below packages

sudo apt install gnupg2 pass -y

gnupg2 is tool for secure communication and data storage. It can be used to encrypt data and to create digital signatures.

Install docker

sudo apt install docker.io -y

Add Ubuntu user to Docker group

sudo usermod -aG docker \$USER

We need to reload shell to have new group settings applied. Now you need to logout and log back in command line or execute the below command:

newgrp docker

The Docker service needs to be set up to run at startup. To do so, type in each command followed by enter:

sudo systemctl start docker

sudo systemctl enable docker

sudo systemctl status docker

```
ubuntu@ip-172-31-24-229:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-07-23 02:34:51 UTC; 12s ago
     Docs: https://docs.docker.com
      Main PID: 4299 (dockerd)
        Tasks: 8
       CGroup: /system.slice/docker.service
               └─4299 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jul 23 02:34:51 ip-172-31-24-229 dockerd[4299]: time="2020-07-23T02:34:51.427980999Z" level=
Jul 23 02:34:51 ip-172-31-24-229 dockerd[4299]: time="2020-07-23T02:34:51.428872171Z" level=
Jul 23 02:34:51 ip-172-31-24-229 dockerd[4299]: time="2020-07-23T02:34:51.428256519Z" level=
Jul 23 02:34:51 ip-172-31-24-229 dockerd[4299]: time="2020-07-23T02:34:51.428660089Z" level=
Jul 23 02:34:51 ip-172-31-24-229 dockerd[4299]: time="2020-07-23T02:34:51.630279701Z" level=
Jul 23 02:34:51 ip-172-31-24-229 dockerd[4299]: time="2020-07-23T02:34:51.730563362Z" level=
Jul 23 02:34:51 ip-172-31-24-229 dockerd[4299]: time="2020-07-23T02:34:51.850890001Z" level=
Jul 23 02:34:51 ip-172-31-24-229 dockerd[4299]: time="2020-07-23T02:34:51.850213536Z" level=
Jul 23 02:34:51 ip-172-31-24-229 dockerd[4299]: time= 2020-07-23T02:34:51.935841521Z" level=
```

The above screenshot should confirm that Docker daemon is up and running.

Docker, Docker pipeline and Kubernetes CLI plug-ins are installed in Jenkins.

Add jenkins user to Docker group

sudo usermod -a -G docker jenkins

Restart Jenkins service

sudo service jenkins restart

Reload system daemon files

sudo systemctl daemon-reload

Restart Docker service as well

sudo service docker stop

sudo service docker start

Go to the Jenkins portal
Manage Jenkins > Plugins
Available plugins
Select the following:

Plugins

Updates Available plugins **Installed plugins** Advanced settings

Q kubernetes

Install Name ↴

Plugin	Version	Category	Description
Docker	1.5	Cloud Providers Cluster Management docker	This plugin integrates Jenkins with Docker
Docker Pipeline	572.v950f58993843	pipeline DevOps Deployment docker	Build and use Docker containers from pipelines.
Kubernetes	4029.v5712230ccb_f8	Cloud Providers Cluster Management kubernetes Agent Management	This plugin integrates Jenkins with Kubernetes
Kubernetes Credentials	0.11	kubernetes credentials	Common classes for Kubernetes credentials
Kubernetes Client API	6.8.1-224.vd388fca_4db_3b_	kubernetes Library plugins (for use by other plugins)	Kubernetes Client API plugin for use by other Jenkins plugins.
Kubernetes CLI	1.12.1	kubernetes	Configure kubectl for Kubernetes

Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress

Download progress

Preparation	
Cloud Statistics	Success
Authentication Tokens API	Success
Docker Commons	Success
Apache HttpComponents Client 5.x API	Success
Docker API	Success
Docker	Success
Docker Pipeline	Success
Kubernetes Client API	Success
Kubernetes Credentials	Success
Kubernetes CLI	Success
Loading plugin extensions	Running

→ [Go back to the top page](#)
(you can start using the installed plugins right away)

→ Restart Jenkins when installation is complete and no jobs are running

Note: do not opt to restart Jenkins

Step # 1 - Create Maven3 variable under Global tool configuration in Jenkins.
Manage Jenkins > Global tool configuration.

Make sure you create a **Maven3** variable under **Global tool configuration**.

The screenshot shows the 'Add Maven' configuration dialog. It includes fields for 'Name' (Maven3) and 'MAVEN_HOME' (/usr/share/maven). A checkbox for 'Install automatically' is present but unchecked. There are 'Add Maven' buttons at the top and bottom of the form.

Click Save

Step #2 - Create Credentials for connecting to Kubernetes Cluster using kubeconfig

Manage Jenkins > Credentials

Click on Add Credentials, use Kubernetes configuration from drop down.

The screenshot shows the Jenkins Global credentials (unrestricted) page. At the top, there is a breadcrumb navigation: Jenkins > Credentials > System > Global credentials (unrestricted). Below the navigation, there are two buttons: "Back to credential domains" and "Add Credentials". The "Add Credentials" button is circled in pink. On the right, there is a section titled "Global cre" with a house icon. It lists "Credentials that should be available" with two items: "fa32f95a-2d3e-4c7b" and "0c5c8205-0068-4aa". Below this, there is a "Icon:" dropdown menu with options S, M, L.

Use secret file from drop down.

The screenshot shows the Jenkins Global credentials (unrestricted) page. The breadcrumb navigation is the same as the previous screenshot. On the left, there is a sidebar with a "Kind" dropdown. The dropdown menu lists several options: "Username with password" (checked), "GitHub App", "OpenShift Username and Password", "SSH Username with private key", "Secret file" (which is highlighted with a blue selection bar), "Secret text", "X.509 Client Certificate", and "Certificate".

Let's now go back to our Jenkins server to generate the Secret file.

Execute the below command to login as jenkins user.

sudo su - jenkins

You should see the nodes running in EKS cluster.

kubectl get nodes

```
ubuntu@ip-172-31-32-245:~$ kubectl get nodes
NAME           STATUS   ROLES      AGE     VERSION
ip-192-168-25-216.us-east-2.compute.internal   Ready    <none>    49m    v1.17.11-eks-cfdcc40
ip-192-168-51-213.us-east-2.compute.internal   Ready    <none>    49m    v1.17.11-eks-cfdcc40
```

Execute the below command to get kubeconfig info, copy the entire content of the file:

cat /var/lib/jenkins/.kube/config

Open your text editor or notepad, copy and paste the entire content and save in a file. We will upload this file.

Kind	Secret file
Scope	Global (Jenkins, nodes, items, all ch...)
File	<input type="button" value="Choose File"/> No file chosen
ID	K8S
Description	

Enter ID as **K8S** and Choose File.

Kind

Secret file

Scope

Global (Jenkins, nodes, items, all children)

File

kuebconfig-jan19.txt

ID

K8S

Description

K8S description



Upload the file and click OK to save.
Make note of the ID **K8S** for the pipeline script later.

Step # 3 - Create a pipeline in Jenkins.

Choose New Item > Pipeline

Enter an item name

Mykube » Required field

 **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with a build step for something other than software build.

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building and/or organizing complex activities that do not easily fit in free-style job type.

 **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple platforms or builds, etc.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike a folder in a file system, Jenkins items have their own separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Call the item **Mykube** and click **OK**.

Configure

Pipeline

Definition
Pipeline script

Script ?

```
1 * pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 echo 'Hello World'  
8             }  
9         }  
10    }  
11}  
12
```

Hello World

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

This screenshot shows the Jenkins Pipeline configuration interface. On the left, there's a sidebar with 'General' and 'Advanced Project Options' sections, and a 'Pipeline' section which is currently selected and highlighted in grey. The main area is titled 'Pipeline' and contains a 'Definition' dropdown set to 'Pipeline script'. Below it is a 'Script' editor with syntax highlighting for Groovy. The script defines a pipeline with one stage named 'Hello' that echoes 'Hello World'. A 'Hello World' dropdown menu is open next to the script. At the bottom of the editor are 'Save' and 'Apply' buttons. A checkbox for 'Use Groovy Sandbox' is checked. There's also a 'Pipeline Syntax' link.

Select Pipeline > Hello World

Script ?

```
1 * pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Checkout') {  
6             steps {  
7                 echo 'Hello World'  
8             }  
9         }  
10    }  
11}
```

This screenshot shows a Jenkins Pipeline script editor. The script defines a pipeline with a single stage named 'Checkout'. Inside the stage, there is a step that echoes 'Hello World'. The code is numbered from 1 to 12. The 'Checkout' stage is currently selected, indicated by a grey background. The 'echo' command is highlighted in green. The editor has a light grey background and a vertical scroll bar on the right.

Click Pipeline Syntax

Sample Step

checkout: Check out from version control

checkout ?

SCM

Git

Repositories ?

Repository URL ?

<https://github.com/costas778/springboot-app>

Credentials ?

Select the above options. Ensure that the Repository URL has the URL pointing to the springboot-app as above.

Branch Specifier (blank for 'any') ?

*/main

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add ▾

Include in polling? ?

Include in changelog? ?

Generate Pipeline Script

```
checkout scmGit(branches: [[name: '/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/costas778/springboot-app']])
```

Click the **Generate Pipeline Script**.

Copy the string back into the pipeline script.

Script ?

```
1 * pipeline {
2     agent any
3
4     stages {
5         stage('Checkout') {
6             steps {
7                 checkout scmGit(branches: [[name: '/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/costas778/springbo
8
9             }
10        }
11    }
12 }
```

Hello World ▾

Save the script.

The screenshot shows the Jenkins Pipeline interface for the 'Mykube' pipeline. The top navigation bar includes 'Dashboard > Mykube >'. On the left, there's a sidebar with various actions: Status (selected), Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. Below this is the 'Build History' section, which is currently set to 'trend'. It shows two builds: #2 (19 Sep 2023, 09:13) and #1 (19 Sep 2023, 09:13). At the bottom of the history section are links for 'Atom feed for all' and 'Atom feed for failures'. The main content area is titled 'Stage View' and displays two stages: 'Checkout' (976ms) and 'Stage View' (577ms). Each stage has a table showing the execution time and any changes made.

Step # 4 - Copy the pipeline code from below.

Make sure you change yellow highlighted values below as per your settings:

Your docker user id should be updated with the following:

642660919026.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo

Your registry credentials ID from Jenkins we created earlier when we created the Secret file is as follows:

K8S

Finally, change the following **springboot-app** file. For myself this is located as follows:

<https://github.com/costas778/springboot-app>

Make changes in **eks-deploy-k8s.yaml** to pull Docker image from your AWS ECR repo.

main > springboot-app / eks-deploy-k8s.yaml

akannan1087 changed port no mapping

1 contributor

42 lines (42 sloc) | 806 Bytes

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   labels:
5     app: springboot-app
6     name: springboot-app
7   spec:
8     replicas: 2
9     selector:
10       matchLabels:
11         app: springboot-app
12     template:
13       metadata:
14         labels:
15           app: springboot-app
16       spec:
17         containers:
18           - name: my-springboot-app
19             image: [REDACTED].dkr.ecr.us-east-2.amazonaws.com/my-docker-repo
20             imagePullPolicy: Always
21             ports:
22               - containerPort: 8085
23
```

main > springboot-app / eks-deploy-k8s.yaml

akannan1087 changed port no mapping

1 contributor

42 lines (42 sloc) | 806 Bytes

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   labels:
5     app: springboot-app
6     name: springboot-app
7   spec:
8     replicas: 2
9     selector:
10       matchLabels:
11         app: springboot-app
12     template:
13       metadata:
14         labels:
15           app: springboot-app
16       spec:
17         containers:
18           - name: my-springboot-app
19             image: 211223789150.dkr.ecr.us-east-2.amazonaws.com/my-docker-repo
20             imagePullPolicy: Always
21             ports:
```

Go to Spec. In image, update it with [642660919026.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo](#)

```
ubuntu@Jenkins: ~
jenkins@Jenkins:~$ docker images
REPOSITORY                                     TAG      IMAGE ID      CREATED       SIZE
642660919026.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo    latest   2c042c242c2d  3 minutes ago  285MB
lolhens/baseimage-openjre                      latest   7990e5390b8d  3 years ago   264MB
jenkins@Jenkins:~$
```

```
pipeline {
  tools {
    maven 'Maven3'
```

```

        }
    }
    agent any
    environment{
        registry = "account_id.dkr.ecr.us-east-2.amazonaws.com/my-docker-repo"
    }
}

stages {
    stage('Cloning Git') {
        steps {
            checkout([$class: 'GitSCM', branches: [[name: '*/main']]],
doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs:
[[credentialsId: '', url: 'https://github.com/<your GITHUB username>/springboot-app']]])
        }
    }
    stage ('Build') {
        steps {
            sh 'mvn clean install'
        }
    }
}
// Building Docker images
stage('Building image') {
    steps{
        script {
            dockerImage = docker.build registry
        }
    }
}

// Uploading Docker images into AWS ECR
stage('Pushing to ECR') {
    steps{
        script {
            sh 'aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-
stdin account_id.dkr.ecr.us-east-2.amazonaws.com'
            sh 'docker push account_id.dkr.ecr.us-east-2.amazonaws.com/my-docker-repo:latest'
        }
    }
}

stage('K8S Deploy') {
    steps{
        script {
            withKubeConfig([credentialsId: 'K8S', serverUrl: ""]){
                sh ('kubectl apply -f eks-deploy-k8s.yaml')
            }
        }
    }
}

```

Once the above script is updated you can overwrite the whole pipeline script.

Step # 5 - Build the pipeline

Once you create the pipeline and changes values per your configuration, click on **Build now:**



Step # 6 - Verify deployments to K8S

kubectl get pods

```
AK-DevOps-Coach:Downloads devopscoachings$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
my-dk-deployment-67984f8cd9-2vl47  1/1     Running   0          28m
```

kubectl get deployments

```
AK-DevOps-Coach:Downloads devopscoachings$ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
my-dk-deployment  1/1      1          1          27m
```

kubectl get services

```
AK-DevOps-Coach:Downloads devopscoachings$ kubectl get svc
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP
kubernetes   ClusterIP   10.100.0.1      <none>
springboot-app   LoadBalancer  10.100.197.254  a7898d8a8449f48afb356e0d205069c2-41286874.us-east-2.elb.amazonaws.com
  PORT(S)        AGE
  443/TCP      21m
springboot-app   LoadBalancer  10.100.197.254  a7898d8a8449f48afb356e0d205069c2-41286874.us-east-2.elb.amazonaws.com
  80:32179/TCP  39s
```

If you see any errors after deploying the pods, you can check the pod logs.

kubectl logs <pod_name>

```
curl: (6) Could not resolve host: a7898d8a8449f48afb356e0d205069c2-41286874.us-east-2.elb.amazonaws.com
jenkins@Jenkins:~$ kubectl logs springboot-app-5cff7b788f-6sk7f
```

Steps # 7 - Access SpringBoot App in K8S cluster

```
Jenkins@Jenkins:~$ kubectl get deployment
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
springboot-app  2/2      2          2          22s
jenkins@Jenkins:~$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
springboot-app-5cff7b788f-6sk7f  1/1     Running   0          30s
springboot-app-5cff7b788f-9cp79  1/1     Running   0          30s
jenkins@Jenkins:~$ kubectl get svc
NAME        TYPE        CLUSTER-IP       EXTERNAL-IP
  PORT(S)        AGE
kubernetes   ClusterIP   10.100.0.1      <none>
  443/TCP      21m
springboot-app   LoadBalancer  10.100.197.254  a7898d8a8449f48afb356e0d205069c2-41286874.us-east-2.elb.amazonaws.com
  80:32179/TCP  39s
jenkins@Jenkins:~$ 
jenkins@Jenkins:~$ curl a7898d8a8449f48afb356e0d205069c2-41286874.us-east-2.elb.amazonaws.com
curl: (6) Could not resolve host: a7898d8a8449f48afb356e0d205069c2-41286874.us-east-2.elb.amazonaws.com
jenkins@Jenkins:~$ kubectl
```

Once build is successful, go to browser and enter master or worker node public IP address along with port number mentioned above

http://loadbalancer_ip_address

You should see page like below:

My Awesome SpringBoot + Docker App

FirstName:

Enter FirstName

LastName:

Enter LastName

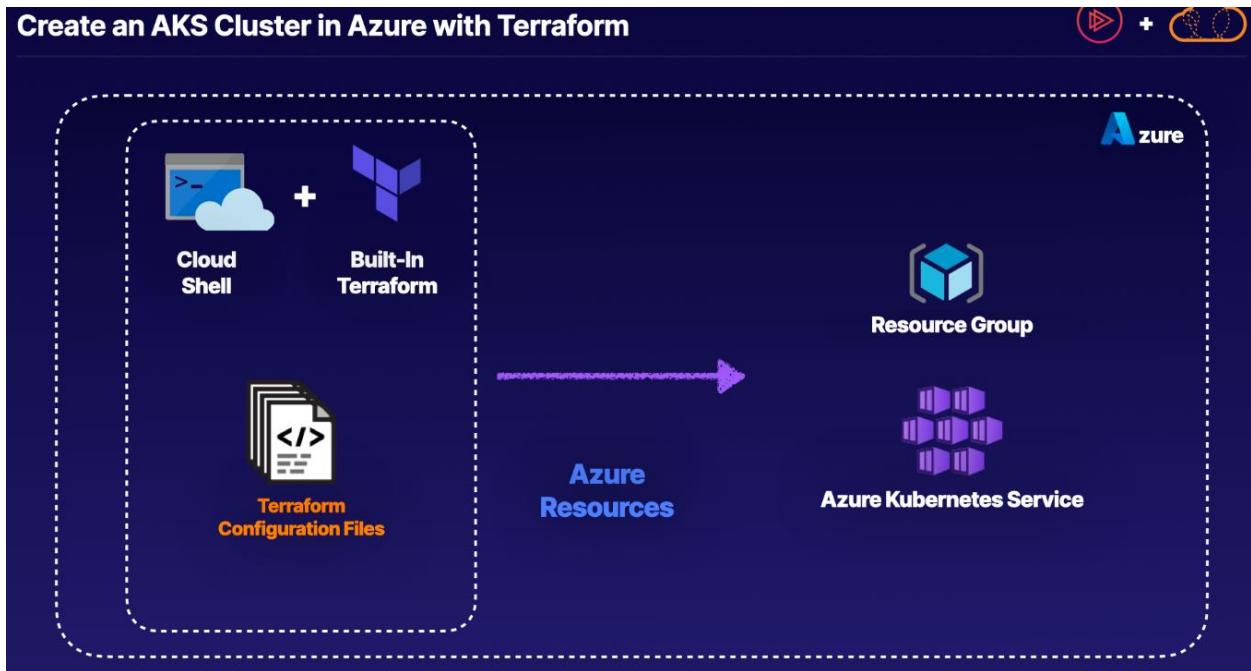
Submit

`{{postResultMessage}}`

`Get All Customers`

`• {{cust.firstname + " " + cust.lastname}}`

Create an AKS Cluster in Azure with Terraform



Solution

A screenshot of the Azure portal showing a Resource Group named "664-3f7d28bd-create-an-aks-cluster-in-azure-with-t". The portal interface includes a search bar, navigation links for Overview, Activity log, Access control (IAM), and Tags, and a toolbar with Create, Manage view, Delete resource group, Refresh, Export to CSV, Open query, and Assign tags. Below the toolbar, detailed resource information is shown: Subscription (move) : P8-Real Hands-On Labs, Deployment count : 1 Succeeded, Subscription ID : 9734ed68-621d-47ed-babd-269110dbacb1, Location : East US, and Tags : Add tags.

Set Up the Cloud Shell and Lab Environment

In the Azure portal, click on the Cloud Shell icon (`>_`) at the top of the page, to the right of the search bar.

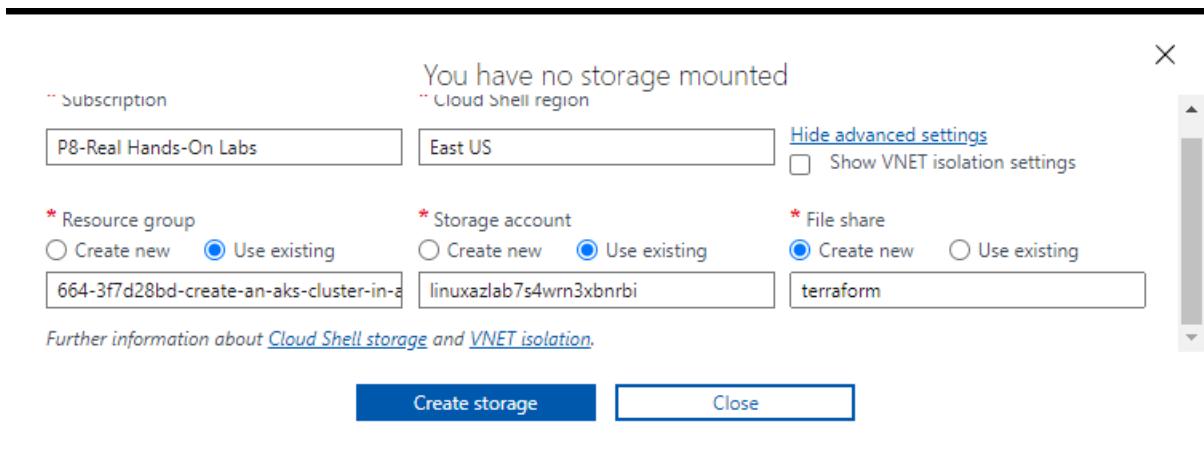
Select **Bash**.

Click **Show advanced settings**.

For the Cloud Shell region, select the **same region** as your **resource group location**.

For **Storage account**, choose **Use existing**.

Under **File share**, select **Create new** and type in the name of **terraform**.



Click **Create storage**. Your Cloud Shell should begin to configure.

Run the following command to pull down the lab setup script from the GitHub repo:

```
 wget https://raw.githubusercontent.com/ACloudGuru/advanced-terraform-with-  
azure/main/lab_aks_cluster/lab_7/setup.sh
```

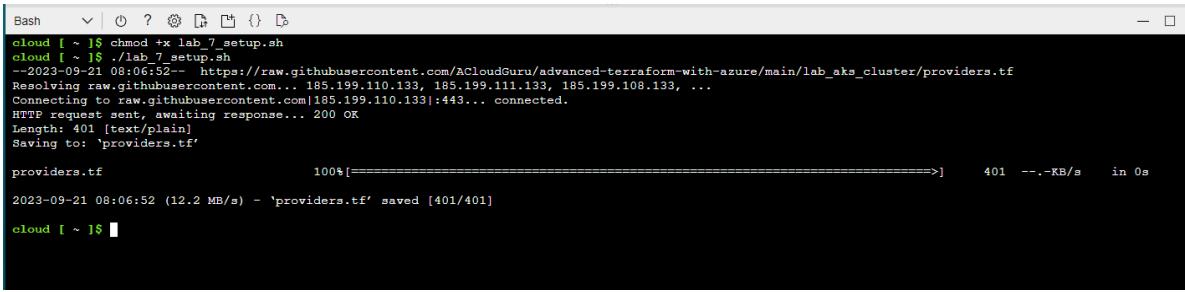
Run **ls** to list the contents.

```
cloud [ ~ ]$ ls  
clouddrive lab_7_setup.sh  
cloud [ ~ ]$ █
```

You should see the lab setup script listed, **lab_7_setup.sh**.

Run **chmod +x lab_7_setup.sh** to make it executable. . .

Run the script, `./lab 7 setup.sh`.



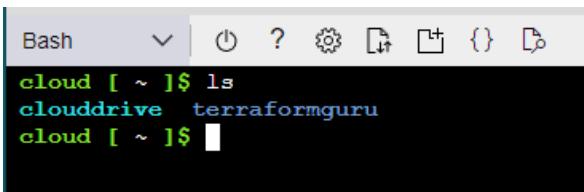
```
Bash      v | ⌁ ? ⚙️ 📁 ⓘ 🔍
cloud [ ~ ]$ chmod +x lab_7_setup.sh
cloud [ ~ ]$ ./lab_7_setup.sh
--2023-09-21 08:06:52-- https://raw.githubusercontent.com/ACloudGuru/advanced-terraform-with-azure/main/lab_aks_cluster/providers.tf
Resolving raw.githubusercontent.com... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 401 [text/plain]
Saving to: 'providers.tf'

providers.tf          100%[=====]   401 --.-KB/s    in 0s

2023-09-21 08:06:52 (12.2 MB/s) - 'providers.tf' saved [401/401]

cloud [ ~ ]$
```

Run **ls** to list the contents.

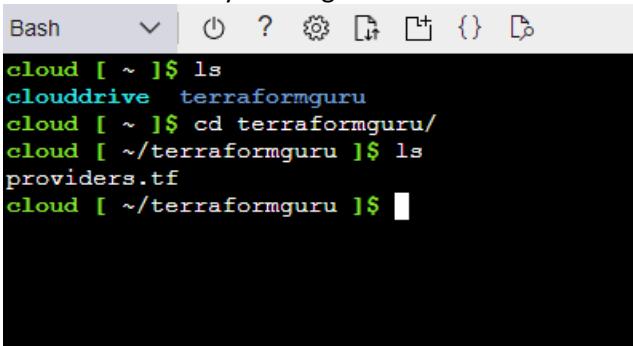


```
Bash      v | ⌁ ? ⚙️ 📁 ⓘ 🔍
cloud [ ~ ]$ ls
clouddrive  terraformguru
cloud [ ~ ]$
```

You should see a **terraformguru** directory listed.

Run **cd terraformguru/** to change into that directory.

List the contents by running **ls**.



```
Bash      v | ⌁ ? ⚙️ 📁 ⓘ 🔍
cloud [ ~ ]$ ls
clouddrive  terraformguru
cloud [ ~ ]$ cd terraformguru/
cloud [ ~/terraformguru ]$ ls
providers.tf
cloud [ ~/terraformguru ]$
```

You should see one configuration file listed: **providers.tf**.

Run **vim providers.tf** to look at the file.

Type **Esc** then **:q** to quit out of the file.

Import the Resource Group

Run **terraform init** to initialize the working directory.

```
Bash ▾ | ⌂ ? ⌘ ⌚ ⌛ ⌘ ⌚

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/azurerm versions matching "~> 2.0"...
- Installing hashicorp/azurerm v2.99.0...
- Installed hashicorp/azurerm v2.99.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
cloud [ ~/terraformguru ]$
```

Run `az group list` to look up the subscription ID.

```
Bash [ ~ /terraformguru ]$ az group list
[ {
    "id": "/subscriptions/9734ed68-621d-47ed-babd-269110dbacb1/resourceGroups/664-3f7d28bd-create-an-aks-cluster-in-azure-with-t",
    "location": "eastus",
    "managedBy": null,
    "name": "664-3f7d28bd-create-an-aks-cluster-in-azure-with-t",
    "properties": {
        "provisioningState": "Succeeded"
    },
    "tags": null,
    "type": "Microsoft.Resources/resourceGroups"
}
]
cloud [ ~/terraformguru ]$
```

Copy the **subscription ID** to your clipboard. It should be located on the top line after "id":
Make sure to copy all of the characters in between the quotation marks.

[/subscriptions/9734ed68-621d-47ed-babd-269110dbacb1/resourceGroups/664-3f7d28bd-create-an-aks-cluster-in-azure-with-t](#)

Run the following command, making sure to paste in your copied subscription ID to replace <SUBSCRIPTION_ID>:

```
terraform import azurerm_resource_group.k8s <SUBSCRIPTION_ID>
terraform import azurerm_resource_group.k8s /subscriptions/9734ed68-621d-47ed-babd-269110dbacb1/resourceGroups/664-3f7d28bd-create-an-aks-cluster-in-azure-with-t
```

Note: It may take a minute to import your resource.

```
Bash [ ~ /terraformguru ]$ terraform import azurerm_resource_group.k8s /subscriptions/9734ed68-621d-47ed-babd-269110dbacb1/resourceGroups/664-3f7d28bd-create-an-aks-cluster-in-azure-with-t
azurerm_resource_group.k8s: Importing from ID "/subscriptions/9734ed68-621d-47ed-babd-269110dbacb1/resourceGroups/664-3f7d28bd-create-an-aks-cluster-in-azure-with-t"...
azurerm_resource_group.k8s: Import prepared!
  Prepared azurerm_resource_group for import
azurerm_resource_group.k8s: Refreshing state... [id=/subscriptions/9734ed68-621d-47ed-babd-269110dbacb1/resourceGroups/664-3f7d28bd-create-an-aks-cluster-in-azure-with-t]

Import successful!

The resources that were imported are shown above. These resources are now in
your Terraform state and will henceforth be managed by Terraform.

cloud [ ~ /terraformguru ]$ 
```

Run **vim providers.tf** to edit the file.

Delete the comment hashes (#) in front of name and location.

Replace the placeholder <RESOURCE GROUP NAME> next to name.

Copy the resource group name located at the top left of the Azure portal, under Home.

Paste it into the file, to replace <RESOURCE_GROUP_NAME> making sure not to replace the quotation marks.

Replace the placeholder <RESOURCE GROUP LOCATION> next to location.

Copy the resource group location listed to the right of Location in the Azure portal.

(If you hover over it, a copy icon should appear that you can click to copy it to your clipboard.)

Paste it into the file, to replace <RESOURCE_GROUP_LOCATION> making sure not to replace the quotation marks.

Type Esc followed by :wq to save and quit the file.

Run the following command to create an SSH key:

```
ssh-keygen -m PEM -t rsa -b 4096
```

Hit Enter to keep the defaults.

Hit **Enter** to leave the passphrase empty.

Hit **Enter** again to create your key pair.

Add the AKS Config, Variables, and Outputs to the Configuration:

Run **vim aks.tf** to create your first configuration file.

Enter the following configuration:

```
resource "azurerm_kubernetes_cluster" "k8s" {
    name          = var.cluster_name
    location      = azurerm_resource_group.k8s.location
    resource_group_name = azurerm_resource_group.k8s.name
    dns_prefix    = var.dns_prefix

    linux_profile {
        admin_username = "ubuntu"
        ssh_key {
            key_data = file(var.ssh_public_key)
        }
    }

    default_node_pool {
```

```

name      = "agentpool"
node_count = var.agent_count
vm_size    = "Standard_D2s_v3"
os_disk_size_gb = 30
}

service_principal {
  client_id  = var.aks_service_principal_app_id
  client_secret = var.aks_service_principal_client_secret
}

network_profile {
  load_balancer_sku = "Standard"
  network_plugin = "kubenet"
}

tags = {
  Environment = "Development"
}
}

```

Type **Esc** followed by **:wq** to save and quit the file.

Run **vim variables.tf** to create your next configuration file.

Enter the following configuration. Be sure to replace **<YOUR_RESOURCE_GROUP_LOCATION>** with the location of your resource group, and replace **<SERVICE_PRINCIPAL_APP_ID>** and **<SERVICE_PRINCIPAL_CLIENT_SECRET>** with the service principal IDs generated for this lab, which can be found in the lab credentials section.

```

variable "resource_group_location" {
  default = "<YOUR_RESOURCE_GROUP_LOCATION>"
}

variable "agent_count" {
  default = 3
}

variable "ssh_public_key" {
  default = "~/.ssh/id_rsa.pub"
}

variable "dns_prefix" {
  default = "k8sguru"
}

variable cluster_name {
  default = "k8sguru"
}

```

```
}
```

```
variable aks_service_principal_app_id {
  default = "<SERVICE_PRINCIPAL_APP_ID>"
}
```

```
variable aks_service_principal_client_secret {
  default = "<SERVICE_PRINCIPAL_CLIENT_SECRET>"
}
```

Type **Esc** followed by **:wq** to save and quit the file.

Run **vim output.tf** to create your final configuration file.

Enter the following configuration:

```
output "resource_group_name" {
  value = azurerm_resource_group.k8s.name
}

output "client_key" {
  value = azurerm_kubernetes_cluster.k8s.kube_config.0.client_key
}

output "client_certificate" {
  value = azurerm_kubernetes_cluster.k8s.kube_config.0.client_certificate
}

output "cluster_ca_certificate" {
  value = azurerm_kubernetes_cluster.k8s.kube_config.0.cluster_ca_certificate
}

output "cluster_username" {
  value = azurerm_kubernetes_cluster.k8s.kube_config.0.username
}

output "cluster_password" {
  value = azurerm_kubernetes_cluster.k8s.kube_config.0.password
}
output "kube_config" {
  value = azurerm_kubernetes_cluster.k8s.kube_config_raw
  sensitive = true
}
output "host" {
  value = azurerm_kubernetes_cluster.k8s.kube_config.0.host
}
```

Type **Esc** followed by **:wq** to save and quit the file.

Deploy and Verify the Kubernetes Cluster is Running

Run **terraform fmt** to check the formatting of your configuration files. Your aks.tf, output.tf, providers.tf, and variables.tf files should be listed.

```
Bash      ▾ | ⌁ ? ⌂ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋

cloud [ ~/terraformguru ]$ ls
providers.tf  terraform.tfstate
cloud [ ~/terraformguru ]$ vi terraform.tfstate
cloud [ ~/terraformguru ]$ aks.tf
bash: aks.tf: command not found
cloud [ ~/terraformguru ]$ vi aks.tf
cloud [ ~/terraformguru ]$ vi variables.tf
cloud [ ~/terraformguru ]$ vi output.tf
cloud [ ~/terraformguru ]$ terraform fmt
aks.tf
output.tf
variables.tf
cloud [ ~/terraformguru ]$ █
```

Run **terraform validate** to validate the code in your configuration files.

You should see a message confirming that your configuration is valid.

```
Bash └─| ⌂ ? ⚙ ⌂ ⌂ ⌂ { } ⌂
cloud [ ~/terraformguru ]$ terraform validate
Success! The configuration is valid.

cloud [ ~/terraformguru ]$ █
```

Run **terraform plan -out aks.tfplan** to create your execution plan.

```
Saved the plan to: aks.tfplan

To perform exactly these actions, run the following command to apply:
  terraform apply "aks.tfplan"
cloud [ ~/terraformguru ]$ █
```

Run **terraform apply aks.tfplan** to execute your execution plan.

Note: It may take a couple of minutes to deploy your resources.

You will see a big block of text appear, which should mean that your cluster deployed successfully. You can scroll up to view the Apply complete message in green to confirm.

Scrolling down from the Apply complete message, you can view the client_certificate, client_key, cluster_ca_certificate, cluster_password, and cluster_username. Lastly, you should see the host address, kube_config, and resource_group_name.

```
azurerm_kubernetes_cluster.k8s: still creating... [3m0s elapsed]
azurerm_kubernetes_cluster.k8s: Creation complete after 3m8s [id=/subscriptions/  
-cluster-in-azure-with-t/providers/Microsoft.ContainerService/managedClus

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:
```

Run the following command to move your kube_config to a different file:

```
echo "$(terraform output kube_config)" > ./azurek8s
```

Run `cat ./azurek8s` to check the file.

You should see EOT at the beginning and end of the file, which will need to be removed.

Run **vim ./azurek8s** to edit the file.

Delete the **<<EOT** at the beginning and the **EOT** at the end of the file.

Type **Esc** followed by **:wq** to save and quit the file.

Run `export KUBECONFIG=./azurek8s` to create your environment variable.

Run `kubectl get nodes` to check if your nodes are running and healthy.

You should see your 3 nodes returned with a **STATUS** of Ready.

Go to the Kubernetes Services within the Azure portal to see the

as well.

Microsoft Azure Search resources, services, and docs (G+)

Home > Kubernetes services > 664-3f7d28bd-create-an-aks-cluster-in-azure-with-t > k8sguru

k8sguru | Node pools

Kubernetes service

Search Add node pool Refresh Start Stop Upgrade Kubernetes Update image Scale node pool Delete Troubleshoot Give feedback

Node pools Nodes

Node pools provide space for applications to run. Node pools of different types can be added to the cluster to handle a variety of workloads, existing node pools can be scaled and upgraded, or node pools that are no longer needed can be deleted. Each node pool will contain nodes backed by virtual machines. [Learn more about node pools](#)

Node pool	Provisioning state	Power state	Node count	Mode	Kubernetes version	Node size	Operating system
agentpool	Succeeded	Running	3/3 ready	System	1.26.6	Standard_D2s_v3	Linux

Home > Kubernetes services > 664-3f7d28bd-create-an-aks-cluster-in-azure-with-t > k8sguru

k8sguru | Node pools

Kubernetes service

Search Add node pool Refresh Show taints Troubleshoot Give feedback

about node pools

Filter by node pool name Enter the full node pool name

Node	Status	CPU	Memory	Disk	Pods	Node pool	Kubernetes
aks-agentpool-29165910-vmss000000	Ready	10%	20%	72%	9	agentpool	1.26.6
aks-agentpool-29165910-vmss000001	Ready	11%	17%	70%	5	agentpool	1.26.6
aks-agentpool-29165910-vmss000002	Ready	9%	20%	73%	8	agentpool	1.26.6

Search Create Delete Refresh Show labels Give feedback

Kubernetes resources

Namespaces Workloads Services and ingresses Storage Configuration Custom resources Run command

Filter by namespace name Enter the full namespace name Add label filter

Name	Status	Age
kube-node-lease	Active	16 minutes
kube-public	Active	16 minutes
kube-system	Active	16 minutes
default	Active	16 minutes

Addendum

```
cloud [ ~/terraformguru ]$ kubectl get namespaces
NAME      STATUS  AGE
default   Active  13m
kube-node-lease  Active  13m
kube-public  Active  13m
kube-system  Active  13m
cloud [ ~/terraformguru ]$ kubectl get pods
No resources found in default namespace.
cloud [ ~/terraformguru ]$
```

To check the current default namespace, run the following command:

kubectl config view | grep namespace

To set a new default namespace, run the following command:

kubectl config set-context --current --namespace=NAMESPACE_NAME
kubectl config set-context --current --namespace=kube-node-lease

To verify that the new default namespace has been set, run the following command:

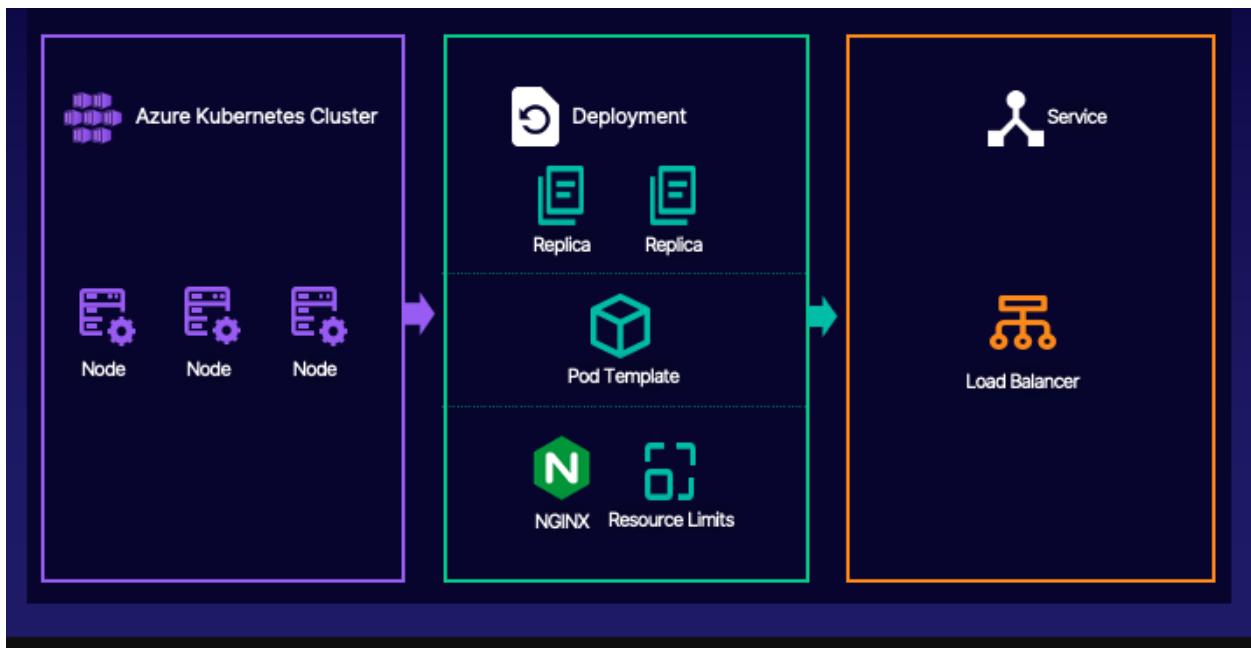
kubectl config view | grep namespace

Change Namespace for a Single Command:

If you want to use a different namespace for a single command, you can specify it using the "--namespace" flag. For example, to list all the pods in a different namespace, run the following command:

kubectl get pods --namespace=NAMESPACE_NAME
kubectl get pods --namespace=KUBE-NODE-LEASE

Deploying and Accessing an Application to an AKS Cluster



Introduction

We will work with the Azure Command Line Interface (CLI) to create an AKS cluster. We will then use the `kubectl` command line utility to deploy an application to that cluster, and then add a kubernetes service, so that the application is accessible over the internet.

Solution

Log in to the Azure portal using the credentials provided on the lab instructions page.

Create AKS Cluster

In the Azure Portal, click the Cloud Shell icon (`>_`) in the upper right.

Select **Bash**.

Click **Show advanced settings**.

Set the **Cloud Shell region** to the same location as the existing resource group.

For **Storage account**, select **Create new** and give it a globally unique name (e.g., "cloudshell" with a series of numbers at the end).

For **File share**, select **Create new** and give it a name of "**fileshare1**".

Click **Create storage**.

Microsoft Azure Search resources, services, and docs (G+ /)

Home > **491-cdc3bb61-deploying-and-accessing-an-applicatio** ⚡ ⚪ ...

Resource group

Search + Create Manage view Delete resource group Refresh Export to CSV Open query Assign tags

Overview Activity log Access control (IAM) Tags Resource visualizer Events

Essentials

Resources Recommendations

Filter for any field... Type equals all × Location equals all × Add filter

Showing 0 to 0 of 0 records. Show hidden types ⓘ No grouping

Name ↑↓	Type ↑↓	Locat ↑↓
---------	---------	----------

You have no storage mounted

Subscription: P4-Real Hands-On Labs Cloud Shell region: East US Hide advanced settings Show VNET isolation settings

* Resource group: Create new Use existing **491-cdc3bb61-deploying-and-accessing-an-applicatio**

* Storage account: Create new Use existing **cloudshell**

* File share: Create new Use existing **fileshare1**

Further information about [Cloud Shell storage](#) and [VNET isolation](#).

Create storage Close

Locate the resource group name at the top of the webpage and copy it for later use:

491-cdc3bb61-deploying-and-accessing-an-applicatio

az group list

```
cloud [ ~ ]$ az group list
[
  {
    "id": "/subscriptions/0fce2870-d256-4119-b0a3-16293ac11bdc/resourceGroups/491-cdc3bb61-deploying-and-accessing-an-applicatio",
    "location": "eastus",
    "managedBy": null,
    "name": "491-cdc3bb61-deploying-and-accessing-an-applicatio",
    "properties": {
      "provisioningState": "Succeeded"
    },
    "tags": null,
    "type": "Microsoft.Resources/resourceGroups"
  }
]
cloud [ ~ ]$
```

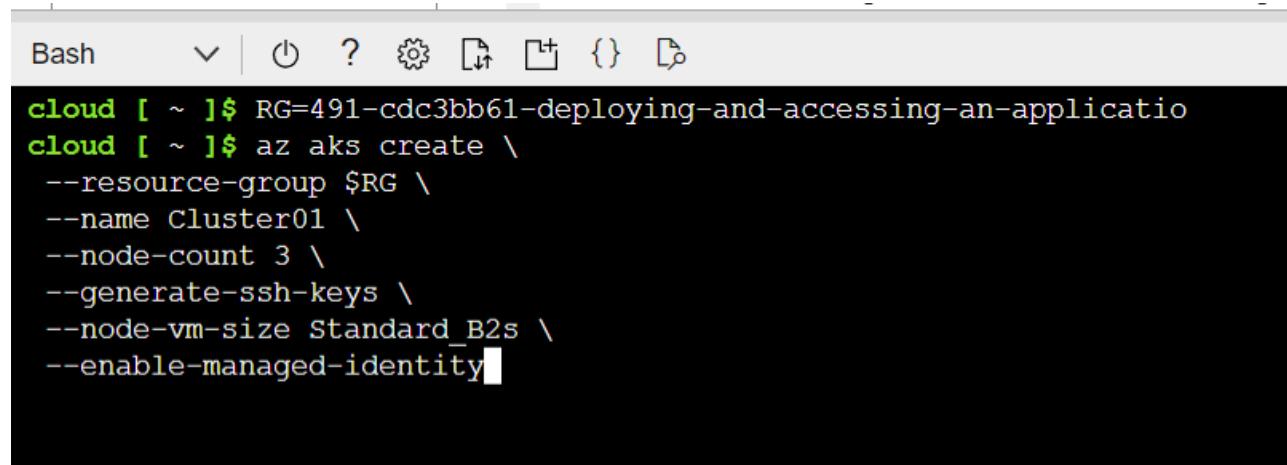
Create a variable for the resource group name:

```
RG=<RESOURCE_GROUP_NAME>
```

RG=491-cdc3bb61-deploying-and-accessing-an-applicatio

Create an AKS cluster:

```
az aks create \
--resource-group $RG \
--name Cluster01 \
--node-count 3 \
--generate-ssh-keys \
--node-vm-size Standard_B2s \
--enable-managed-identity
```



A screenshot of the Azure Cloud Shell interface. The title bar says "Bash". Below it is a command-line area with the following text:

```
cloud [ ~ ]$ RG=491-cdc3bb61-deploying-and-accessing-an-applicatio
cloud [ ~ ]$ az aks create \
--resource-group $RG \
--name Cluster01 \
--node-count 3 \
--generate-ssh-keys \
--node-vm-size Standard_B2s \
--enable-managed-identity
```

```
--enable-managed-identity
SSH key files '/home/cloud/.ssh/id_rsa' and '/home/cloud/.ssh/id_rsa.pub' have been generated under ~/.ssh to allow SSH access to the VM. If using machines without permanent storage like Azure Cloud Shell without an attached file share, back up your keys to a safe location
[]\ Running ..
```

In the Azure Portal, under **All services**, navigate to **Kubernetes services**, and verify that the cluster was deployed successfully.

Back in the Cloud Shell, configure kubectl so that commands can be run against the cluster:

```
az aks get-credentials --name Cluster01 --resource-group $RG
```

```
Read more about the command in reference docs
cloud [ ~ ]$ RG=491-cdc3bb61-deploying-and-accessing-an-applicatio
cloud [ ~ ]$ az aks get-credentials --name Cluster01 --resource-group $RG
Merged "Cluster01" as current context in /home/cloud/.kube/config
cloud [ ~ ]$ []
```

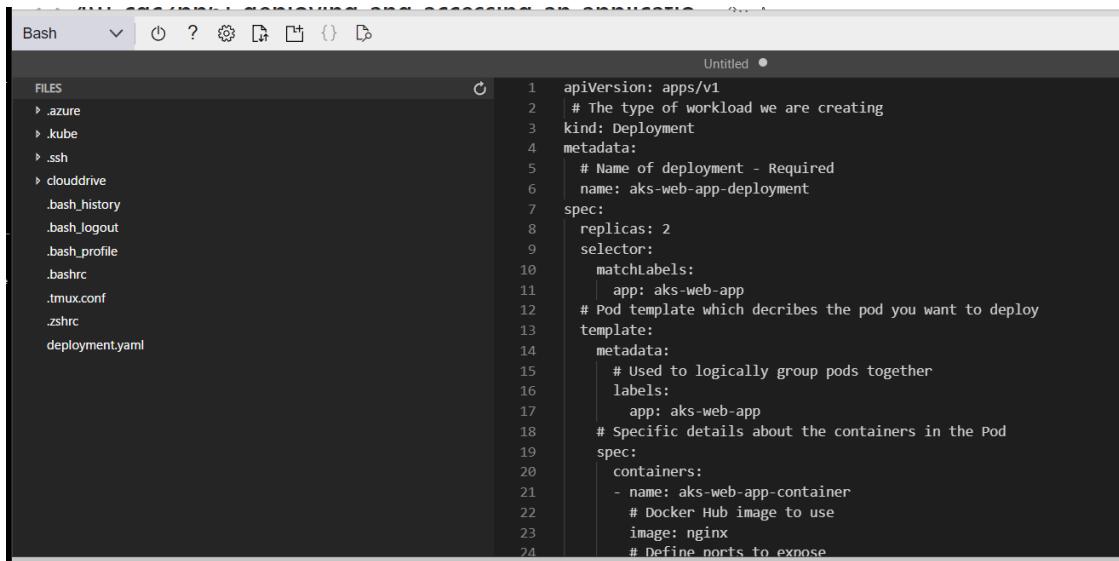
Deploy the Application

Create a new deployment manifest file:

touch deployment.yaml

Open the editor by clicking the curly braces {}, and open the **deployment.yaml** file.

Copy and paste the contents of the **deployment.yaml** file from the **GitHub** repo, into the new file.



```
apiVersion: apps/v1
# The type of workload we are creating
kind: Deployment
metadata:
  # Name of deployment - Required
  name: aks-web-app-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: aks-web-app
  # Pod template which describes the pod you want to deploy
  template:
    metadata:
      # Used to logically group pods together
      labels:
        app: aks-web-app
    # Specific details about the containers in the Pod
    spec:
      containers:
        - name: aks-web-app-container
          # Docker Hub image to use
          image: nginx
          # Define ports to expose
```

Click the menu icon in the top right (...), and click Close Editor.

Make sure to save the file. Use **ctrl + S**

Name the file **deployment.yaml**.

Deploy the application:

kubectl apply -f ./deployment.yaml

In the Azure Portal, under Kubernetes services, open **Cluster01**.

In the left-hand menu, under **Kubernetes Services**, click **Workloads**.

	Name	Namespace	Ready	Up-to-date	Available
<input type="checkbox"/>	coredns	kube-system	✓ 2/2	2	2
<input type="checkbox"/>	coredns-autoscaler	kube-system	✓ 1/1	1	1
<input type="checkbox"/>	konnectivity-agent	kube-system	✓ 2/2	2	2
<input type="checkbox"/>	metrics-server	kube-system	✓ 2/2	2	2
<input type="checkbox"/>	azure-policy	kube-system	✓ 1/1	1	1
<input type="checkbox"/>	azure-policy-webhook	kube-system	✓ 1/1	1	1
<input type="checkbox"/>	gatekeeper-audit	gatekeeper-system	✓ 1/1	1	1
<input type="checkbox"/>	gatekeeper-controller	gatekeeper-system	✓ 2/2	2	2
<input type="checkbox"/>	aks-web-app-deployment	default	✓ 2/2	2	2

In the **Deployments** tab, open the new deployment, and verify that the pods are running.

Access the Application Externally

Create a new file named **service.yaml**:

touch service.yaml

Open the editor by clicking the curly braces { }, and open the **service.yaml** file.

Copy and paste the contents of the **service.yaml** file from the **GitHub** repo, into the new file.

Click the menu icon in the top right (...), and click Close Editor.

Make sure to save the file. Use **ctrl + S**

Name the file **service.yaml**

Deploy the LoadBalancer service:

kubectl apply -f service.yaml

```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

cloud [ ~ ]$ kubectl apply -f service.yaml
service/aks-web-app-service created
cloud [ ~ ]$ █
```

Back in the **Azure Portal**, go to back to **Kubernetes services**. In **Cluster01**, click **Services and ingresses**.

Open the newly deployed **LoadBalancer**.

Name	Namespace	Status	Type	Cluster IP	External IP	Ports
kubernetes	default	Ok	ClusterIP	10.0.0.1	443/TCP	
kube-dns	kube-system	Ok	ClusterIP	10.0.0.10	53/UDP,53/TCP	
metrics-server	kube-system	Ok	ClusterIP	10.0.193.47	443/TCP	
azure-policy-webhook-s...	kube-system	Ok	ClusterIP	10.0.218.42	443/TCP	
gatekeeper-webhook-se...	gatekeeper-system	Ok	ClusterIP	10.0.199.195	443/TCP	
aks-web-app-service	default	Ok	LoadBalancer	10.0.201.164	20.185.101.46	80:32656/TCP

Click the External IP and verify access to the application.

Not secure | http://20.185.101.46

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

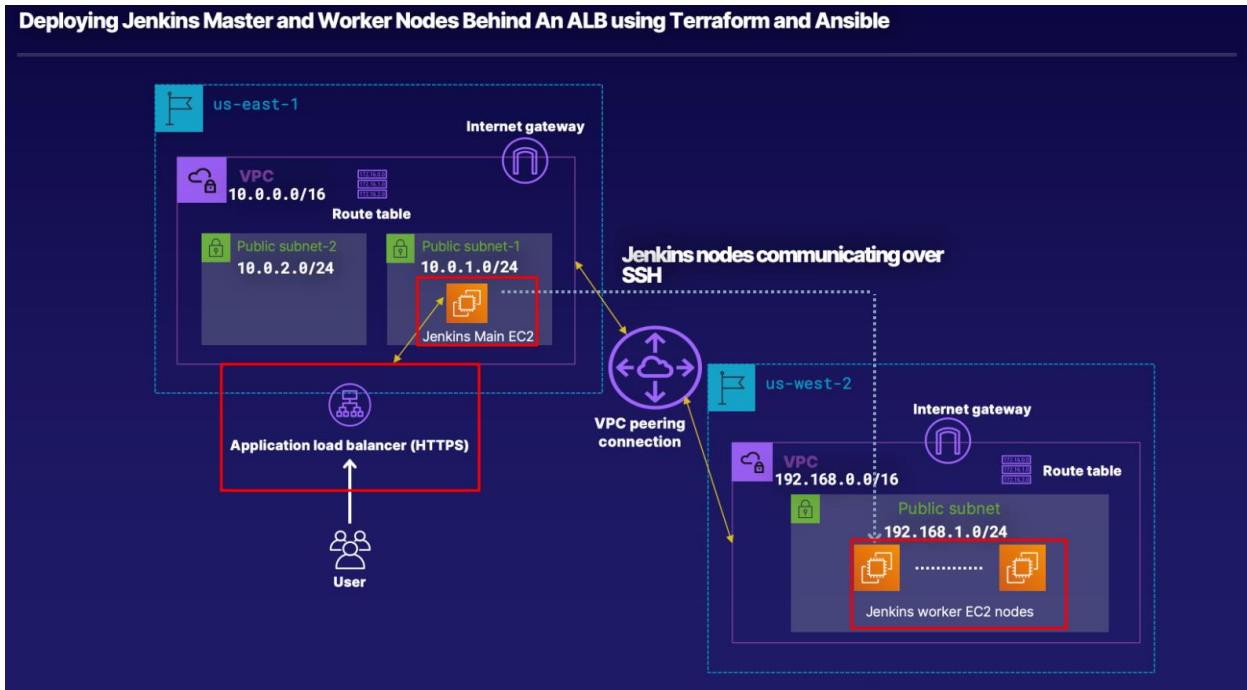
Thank you for using nginx.

Addendum

GitHub Repo files are located as follows:

<https://github.com/linuxacademy/content-aks-basics/tree/master/Lab-Deploying-and-accessing-an-application-to-an-AKS-cluster>

Deploying Jenkins Master and Worker Nodes in AWS Behind an ALB Using Terraform and Ansible



Solution

Pre-Steps:

1. Go and setup within an AWS environment (SANDBOXES are fine) a new EC2 instance within EC2:

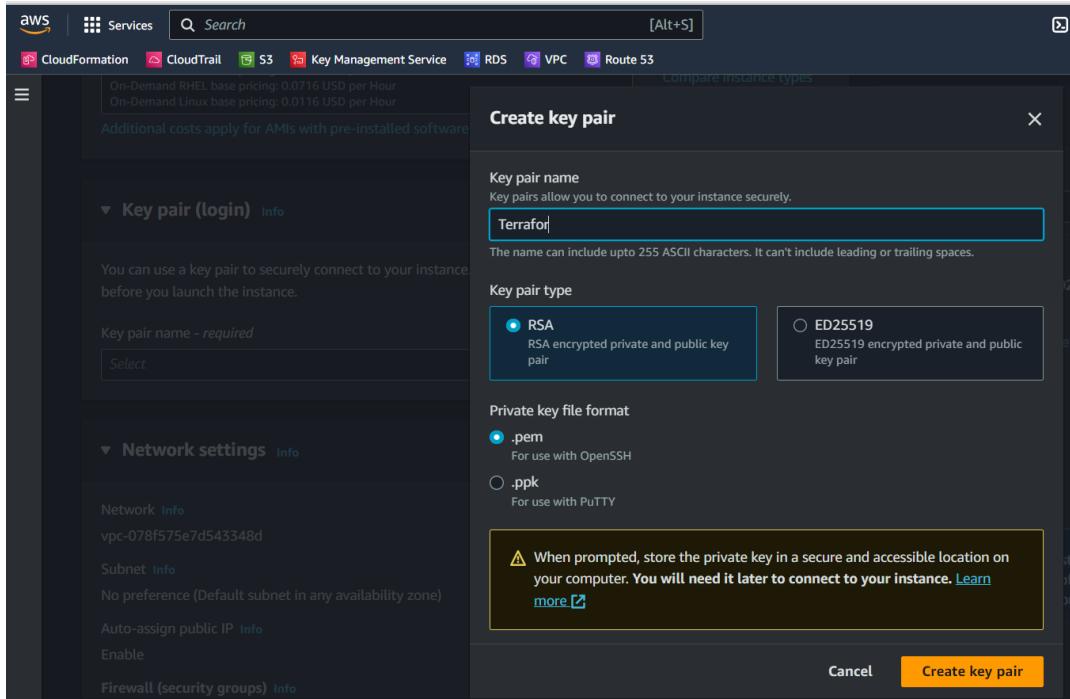
Click on **Launch Instance**

Choose one of the free tier Amazon Linux machines.

Ensure you also create a key pair. Call it "**Terraform**".

Click on **Create Key Pair**

Scroll to the bottom of the webpage and create the instance.



2.Launch a command prompt from your local system and login to the EC2 instance using the command below.

```

[ec2-user@ip-172-31-43-92 ~]
27/08/2023 10:19          1,674 pipelineex.pem
05/03/2023  09:17          1,678 SSHv2.pem
08/09/2023  23:49          1,678 Terraform.pem
29/03/2023  11:09          1,674 WebKey.pem
28/03/2023  11:51          1,678 WhizKey.pem
28/03/2023  13:15          1,674 WhizKey2.pem
30/03/2023  10:49          1,674 WhizKey30.pem
30/03/2023  11:30          1,674 WhizKey30a.pem
30/03/2023  11:50          1,678 WhizKey30b.pem
      21 File(s)           35,198 bytes
          0 Dir(s)   10,389,254,144 bytes free

C:\Users\User\Downloads>ssh ec2-user@100.24.48.169
ec2-user@100.24.48.169: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).

C:\Users\User\Downloads>Terraform.pem ssh ec2-user@100.24.48.169

C:\Users\User\Downloads>ssh -i Terraform.pem ec2-user@100.24.48.169
,
#_
~\_ ####_      Amazon Linux 2023
~~ \####\_
~~ \###|
~~  \#/ __  https://aws.amazon.com/linux/amazon-linux-2023
~~ V~' '-'>
~~~ /
~~. _/ /
~~ /_/
~/m/'

Last login: Fri Sep  8 20:50:51 2023 from 18.206.107.27
[ec2-user@ip-172-31-43-92 ~]$
```

```
ssh -i /path/to/your/Terraform.pem ec2-user@<Public IP of the EC2 Instance you just created>
```

Run the following **cat** command:

```
cat /etc/os-release
[cloud_user@ip-10-0-1-79 ~]$ cat /etc/os-release
NAME="Amazon Linux"
VERSION="2"
ID="amzn"
ID_LIKE="centos rhel fedora"
VERSION_ID="2"
PRETTY_NAME="Amazon Linux 2"
ANSI_COLOR="0;33"
CPE_NAME="cpe:2.3:o:amazon:amazon_linux:2"
HOME_URL="https://amazonlinux.com/"
[cloud_user@ip-10-0-1-79 ~]$
```

3. Check if Python is installed

Use the following command: **python3 –version**

NOTE: It is important to ensure that both **PIP** and **botocore boto3** libraries are installed on this system.

Please follow these steps:

```
sudo yum update
sudo yum install python3
sudo yum install python3-pip
pip install botocore boto3
python3 –version
```

NOTE: if **botocore boto3** libraries are missing than Jenkins will result in a Gateway 502 error!

4. Install Ansible using PIP

```
sudo pip3 install ansible
```

You may get the following warning:

WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead:

<https://pip.pypa.io/warnings/venv>

Ignore the above warning.

Issue the following to confirm the installation of ansible:

```
ansible –version
```

5. Install Git

```
sudo yum install git
```

Y

6. Authenticate to AWS using aws configure

Type **aws configure** and press <enter>.

Provide both the **Access Key ID** and **Secret Access key ID** went prompted.

Accept the **defaults** for the other prompts.

AWS Sandbox
Dig into Amazon Web Services console in a sandbox, so you can get your hands cloudy.

Username	cloud_user
Password	K&\$0U^yM3RX8Swek5%
URL	https://949221891277.siginin.aws.amazon.com/cc
Access Key Id	AKIA52AQITTGW3GW2CFV
Secret Access Key	CteAU4DHV+JRTV9bvYy8hA+FdZ5HLsTru7KMyV

NOTE: if you don't have both the Access key ID and Secret Access Key you can go to IAM within the AWS console and generate them.

Please review earlier projects for the process.

7. Install Terraform

Check if Terraform is installed on the system with the follow command:

terraform version

Note: The Terraform version should be **12.13**. If it's a different version there will be issues when processing the Terraform files. There will be different errors that will require editing of the code. To avoid such headaches simply ensure the version above is installed (even if you have to remove another version).

```
# Uninstall any existing Terraform version.  
sudo rm /usr/local/bin/terraform  
# Download Terraform v0.12.13 for Linux (adjust for your OS)  
wget https://releases.hashicorp.com/terraform/0.12.13/terraform_0.12.13_linux_amd64.zip  
# Unzip the downloaded file  
unzip terraform_0.12.13_linux_amd64.zip  
# Move the Terraform binary to /usr/local/bin to make it globally accessible  
sudo mv terraform /usr/local/bin/  
# Verify the installation  
terraform version
```

8. Download git files and deploy the Terraform code

Clone the GitHub Repo for Terraform Code

a. Use the git command to clone the GitHub repo which has the Terraform code for deploying the solution.

Execute the following command:

```
git clone https://github.com/ACloudGuru-Resources/content-deploying-to-aws-ansible-terraform.git
```

b. Generate a key pair using Ansible

```
cd content-deploying-to-aws-ansible-terraform/lab_jenkins_master_worker
```

Examine the contents of the directory you're in:

```
ls
```

Run the `gen_ssh_key.yaml` Ansible Playbook to generate a SSH Key Pair

```
ansible-playbook ansible_templates/gen_ssh_key.yaml
```

This Ansible Playbook will generate an SSH key pair for your user **cloud_user** which is required for deploying EC2 key pairs in our code.

Note: Alternatively, you may also run the following Linux command to do the same:

```
ssh-keygen -t rsa
```

Note: When this above command prompts for input, keep pressing enter until you're returned to the prompt.

Do not enter a passphrase.

c. Deploy the Terraform Code

```
terraform init
```

```
terraform fmt
```

```
terraform validate
```

```
terraform plan
```

```
terraform apply
```

Enter **yes** when prompted.

After terraform apply has run successfully, you can either use the AWS CLI on the Controller node to list and describe created resources or you can log in to the AWS Console to verify and investigate created resources.

After a successful terraform apply, you will get the DNS URL of the ALB.

Test it out to see if you can reach your Jenkins deployment.

Apply complete! Resources: 3 added, 0 changed, 1 destroyed.

Outputs:

```
Jenkins-Master-AMI-ID = ami-0e1c5d8c23330dee3
```

```
Jenkins-Master-Private-IP = 10.0.1.217
```

```
Jenkins-Worker-AMI-ID = ami-04288abc8d2000768
```

```
Jenkins-Worker-Private-IPs = {
```

```
    "i-0cd8f79141ddd30c4" = "192.168.1.137"
```

```
}
```

```
Jenkins-Worker-Public-IPs = {
```

```
"i-0cd8f79141ddd30c4" = "18.237.211.240"
}
Loadbalancer-DNS-URL = jenkins-lb-22632709.us-east-1.elb.amazonaws.com
```

d. Use the following Jenkins credentials to login:

username: admin

password: password

The screenshot shows the Jenkins dashboard at <http://jenkins-lb-1662211725.us-east-1.elb.amazonaws.com>. The page title is "Welcome to Jenkins!". A search bar at the top right contains the placeholder "Search (CTRL+K)". On the left, there is a sidebar with various links: "New Item", "People", "Build History", "Project Relationship", "Check File Fingerprint", "Manage Jenkins", "My Views", "Job Config History", "Open Blue Ocean Homepage", and "Lockable Resources". Below the sidebar, a "Build Queue" section is shown with the message "No builds in the queue.". The main content area features a "Start building your software project" button and a "Create a job" button.

e. Finally, on the Terraform Controller node CLI, delete all resources which were created and ensure that it runs through successfully.

terraform destroy

Addendum

1. If you fail to install the Python libraries above using PIP you will get the following errors:

```
aws_instance.jenkins-worker-oregon[0] (local-exec): [WARNING]: * Failed to parse /home/ec2-user/content-deploying-to-aws-ansible-t
aws_instance.jenkins-worker-oregon[0] (local-exec):
erraform/lab_jenkins_master_worker/ansible_templates/inventory_aws/tf_aws_ec2.y
aws_instance.jenkins-worker-oregon[0] (local-exec): ml with auto plugin: Failed to import the required Python library (botocore and
aws_instance.jenkins-worker-oregon[0] (local-exec): boto3) on ip-172-31-43-130.ec2.internal's Python /usr/bin/python3. Please read
aws_instance.jenkins-worker-oregon[0] (local-exec): the module documentation and install it in the appropriate location. If the
aws_instance.jenkins-worker-oregon[0] (local-exec): required library is installed, but Ansible is using the wrong Python
aws_instance.jenkins-worker-oregon[0] (local-exec): interpreter, please consult the documentation on ansible_python_interpreter
aws_instance.jenkins-worker-oregon[0] (local-exec): [WARNING]: * Failed to parse /home/ec2-user/content-deploying-to-aws-ansible-t
aws_instance.jenkins-worker-oregon[0] (local-exec):
erraform/lab_jenkins_master_worker/ansible_templates/inventory_aws/tf_aws_ec2.y
aws_instance.jenkins-worker-oregon[0] (local-exec): ml with yaml plugin: Plugin configuration YAML file, not YAML inventory
aws_instance.jenkins-worker-oregon[0] (local-exec): [WARNING]: * Failed to parse /home/ec2-user/content-deploying-to-aws-ansible-t
aws_instance.jenkins-worker-oregon[0] (local-exec):
erraform/lab_jenkins_master_worker/ansible_templates/inventory_aws/tf_aws_ec2.y
aws_instance.jenkins-worker-oregon[0] (local-exec): ml with ini plugin: Invalid host pattern '---' supplied, '---' is normally a
aws_instance.jenkins-worker-oregon[0] (local-exec): sign this is a YAML file.
aws_instance.jenkins-worker-oregon[0] (local-exec): [WARNING]: Unable to parse /home/ec2-user/content-deploying-to-aws-ansible-terr
aws_instance.jenkins-worker-oregon[0] (local-exec):
aform/lab_jenkins_master_worker/ansible_templates/inventory_aws/tf_aws_ec2.yml
aws_instance.jenkins-worker-oregon[0] (local-exec): as an inventory source
aws_instance.jenkins-worker-oregon[0] (local-exec): [WARNING]: No inventory was parsed, only implicit localhost is available
aws_instance.jenkins-worker-oregon[0] (local-exec): [WARNING]: provided hosts list is empty, only localhost is available. Note that
aws_instance.jenkins-worker-oregon[0] (local-exec): the implicit localhost does not match 'all'
aws_instance.jenkins-worker-oregon[0] (local-exec): [WARNING]: Could not match supplied host pattern, ignoring:
aws_instance.jenkins-worker-oregon[0] (local-exec): tag_Name_jenkins_worker_tf_1
```

The above was the Output at the end of **Terraform Apply**

It will generate a load balance string similar to the following.

DNS-URL = **jenkins-lb-22632709.us-east-1.elb.amazonaws.com**

However, when you load it into a web browser it will generate a Gateway 502 error!

2. If you fail to login with aws configure you will get the following during the Terraform Plan command.

```
ec2-user@ip-172-31-43-92 lab_jenkins_master_worker]$ terraform plan
```

Refreshing Terraform state in-memory prior to plan...

The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.

Error: error configuring Terraform AWS Provider: **no valid credential sources** for Terraform AWS Provider found.

Please see <https://registry.terraform.io/providers/hashicorp/aws> for more information about providing credentials.

Error: NoCredentialProviders: no valid providers in chain. Deprecated.

For verbose messaging see aws.Config.CredentialsChainVerboseErrors on network_setup.tf line 1, in provider "aws":

```
1: provider "aws" {
```

Error: error configuring Terraform AWS Provider: no valid credential sources for Terraform AWS Provider found.

Please see <https://registry.terraform.io/providers/hashicorp/aws> for more information about providing credentials.

Error: NoCredentialProviders: no valid providers in chain. Deprecated.

For verbose messaging see aws.Config.CredentialsChainVerboseErrors

on network_setup.tf line 7, in provider "aws":

```
7: provider "aws" {
```

3. If you have another version of Terraform than the one mentioned above you will get the following errors when you run Terraform Init:

Error: Invalid reference from destroy provisioner

```
|  
| on instances.tf line 60, in resource "aws_instance" "jenkins-worker-oregon":  
| 60: inline = [  
| 61:   "java -jar /home/ec2-user/jenkins-cli.jar -auth @/home/ec2-user/jenkins_auth -s  
http://${aws_instance.jenkins-master.private_ip}:8080 -auth @/home/ec2-user/jenkins_auth delete-  
node ${self.private_ip}"  
| 62: ]  
|  
| Destroy-time provisioners and their connection configurations may only reference attributes of the  
related resource, via 'self',
```

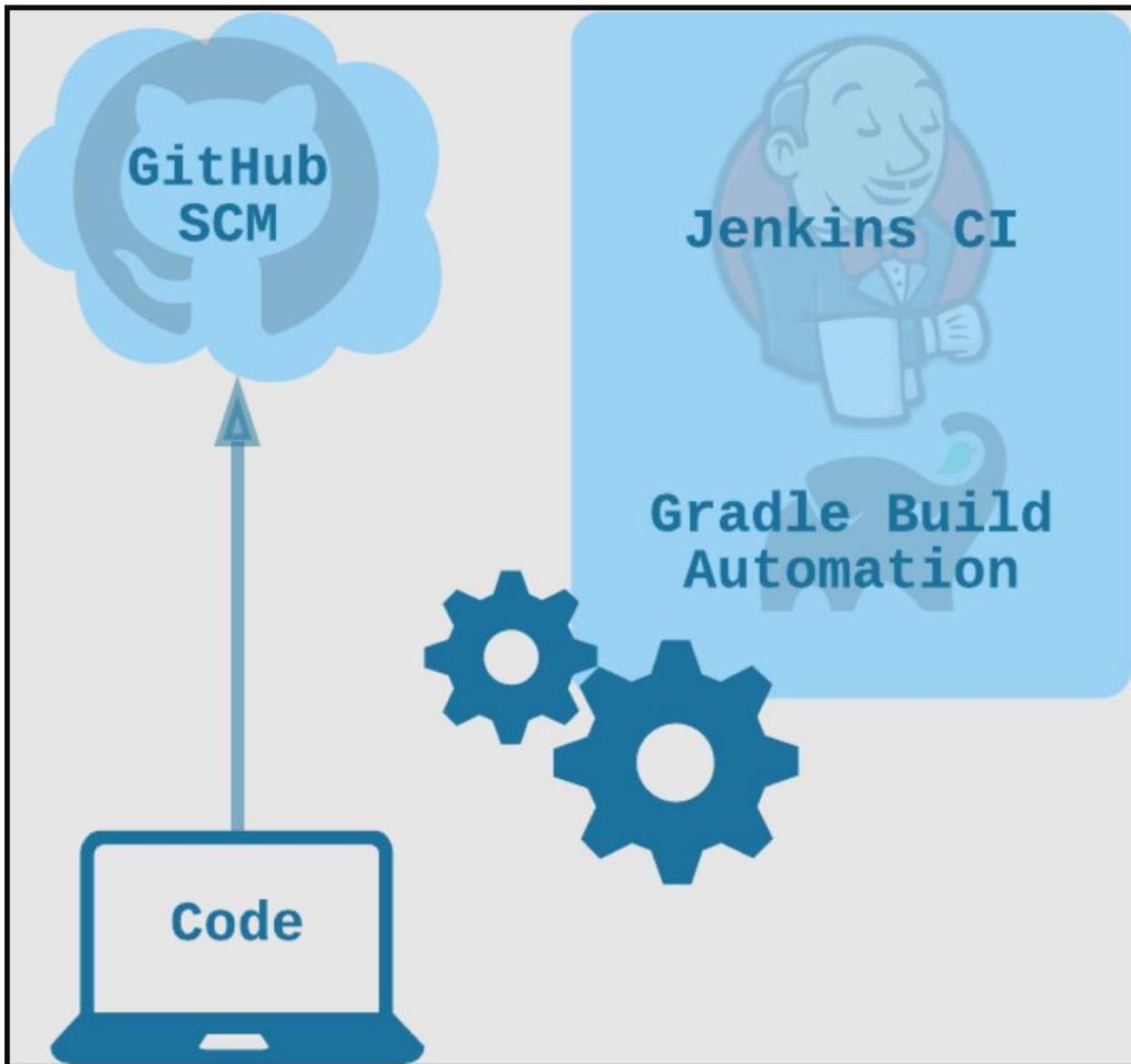
```
| 'count.index', or 'each.key'.  
|
```

| References to other resources during the destroy phase can cause dependency cycles and interact poorly with create_before_destroy.

|

[cloud_user@9c438d91c5a44174a4a8c7a5cd9beb884c lab_jenkins_master_worker]\$

Deploying a Docker Container with Jenkins Pipelines



Pre-Steps:

1. Setup two (FREE) Centos 7.5 servers on the Azure portal.



On one we will Install Jenkins.

On the other we will setup our production server that we will deploy the Docker Container.

Microsoft Azure Search resources, services, and docs (G+)

Home > 1-ad54ef4b-playground-sandbox > Marketplace > CentOS 7.5 Free >

Create a virtual machine ...

Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * P5-Real Hands-On Labs

Resource group * 1-ad54ef4b-playground-sandbox [Create new](#)

Instance details

Virtual machine name * Jenkins

Region * (US) East US

Availability options No infrastructure redundancy required

Security type Standard

Image * CentOS 7.5 Free - x64 Gen1 [See all images](#) | [Configure VM generation](#)

VM architecture Arm64 x64

Give the Jenkins server a name (e.g., Jenkins)

Run with Azure Spot discount

Size Standard_DS3_v2 - 4 vcpus, 14 GiB memory (US\$213.89/month)

[See all sizes](#)

Info Item(s) availability based on policy assignment(s) for the selected scope.
vadergen2labs/Microsoft.Authorization/8ab9bdb77b304947a1347e88 ([Policy details](#))

[Request quota](#)
[Refresh quota](#)

Administrator account

Authentication type SSH public key Password

Info Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.

Username *

SSH public key source

Key pair name *

The options for both servers will likely be as above. It depends, in part, on your subscription. The key option to ensure you select **SSH public key source** radio button as the **authentication type**. This is because we intent to remote into the virtual machines from our local workstation.

Microsoft Azure

Home > 1-49cd24dc-playground-sandbox > Marketplace >

Create a virtual machine ...

Validation passed

Basics Disks Networking Management Monitoring Advanced Tags [Review + create](#)

Info Cost given below is an estimate and not the final price. Please use [Pricing calculator](#) for all your pricing needs.

Price

CentOS 7.5 Free by Cognosys Terms of use Privacy policy	Not covered by credits <input type="checkbox"/> 0.0000 USD/hr
1 X Standard DS3 v2 by Microsoft Terms of use Privacy policy	Subscription credits apply <input type="checkbox"/> 0.2930 USD/hr Pricing for other VM sizes

TERMS

When the Validation passes as above click **create**.

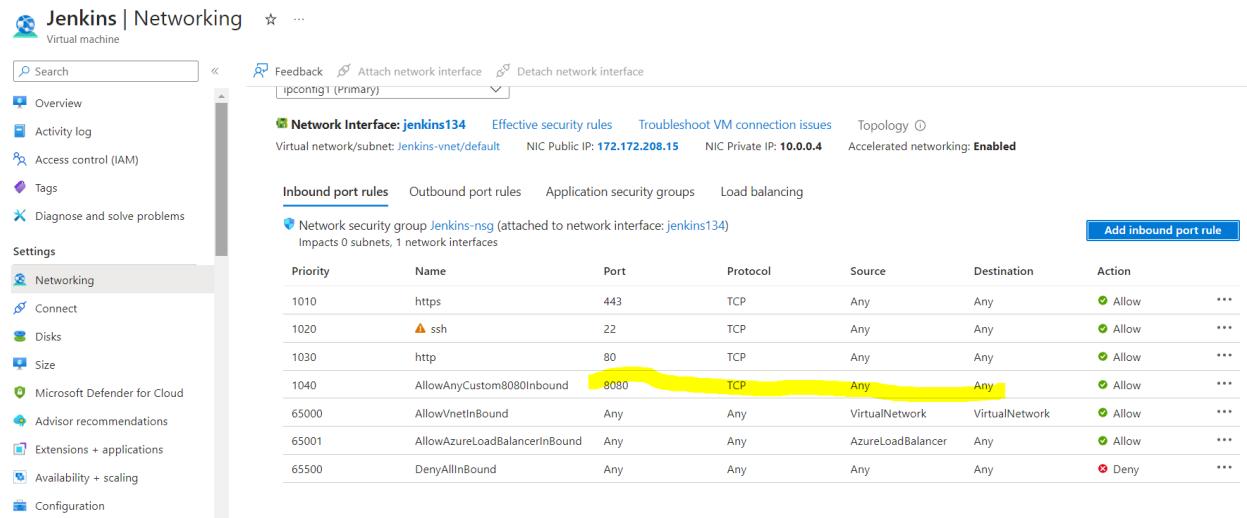
Generate new key pair

Info An SSH key pair contains both a public key and a private key. **Azure doesn't store the private key**. After the SSH key resource is created, you won't be able to download the private key again. [Learn more](#)

When you select the **Download private key and create resource** button this will generate a new key pair to help you securely login to the server remotely!

When the virtual machines are generated select each, one at a time, and create a Network interface inbound rule. This function is located by going to **Networking** under **Settings**.

The port rule you will create for each server is to enable port 8080 from any source are as shown below.



Jenkins | Networking

Network Interface: jenkins134 Effective security rules Troubleshoot VM connection issues Topology

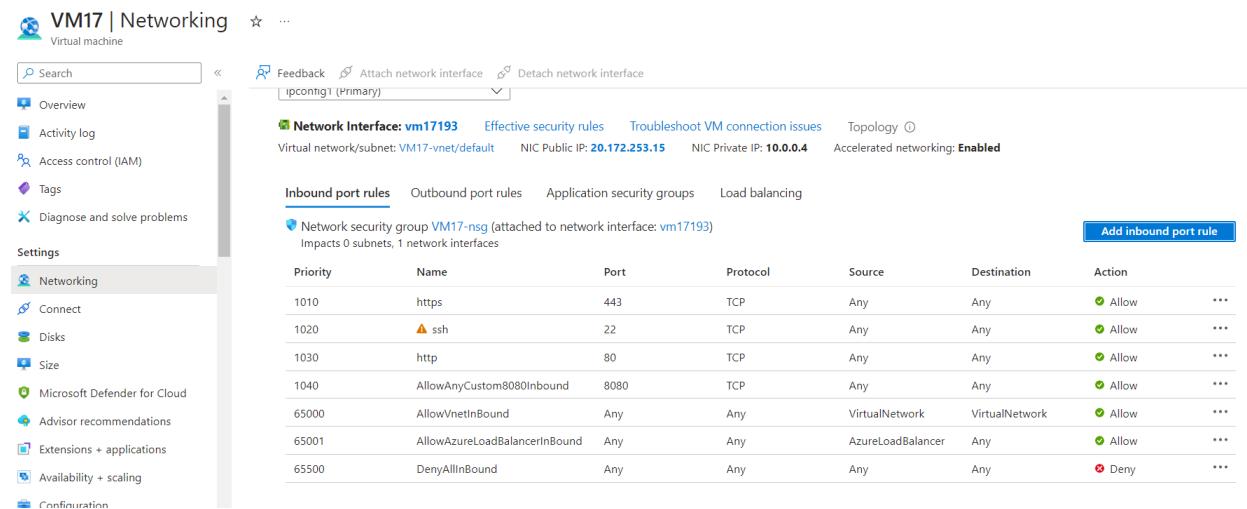
Virtual network/subnet: Jenkins-vnet/default NIC Public IP: 172.172.208.15 NIC Private IP: 10.0.0.4 Accelerated networking: Enabled

Inbound port rules Outbound port rules Application security groups Load balancing

Priority	Name	Port	Protocol	Source	Destination	Action
1010	https	443	TCP	Any	Any	Allow
1020	ssh	22	TCP	Any	Any	Allow
1030	http	80	TCP	Any	Any	Allow
1040	AllowAnyCustom8080Inbound	8080	TCP	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

The port 8080 rule in the case of the Jenkins server is to allow it to communicate externally. Failure to do so will prevent you from loading the Jenkins web portal.

<Jenkins server public IP address>:8080



VM17 | Networking

Network Interface: vm17193 Effective security rules Troubleshoot VM connection issues Topology

Virtual network/subnet: VM17-vnet/default NIC Public IP: 20.172.253.15 NIC Private IP: 10.0.0.4 Accelerated networking: Enabled

Inbound port rules Outbound port rules Application security groups Load balancing

Priority	Name	Port	Protocol	Source	Destination	Action
1010	https	443	TCP	Any	Any	Allow
1020	ssh	22	TCP	Any	Any	Allow
1030	http	80	TCP	Any	Any	Allow
1040	AllowAnyCustom8080Inbound	8080	TCP	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

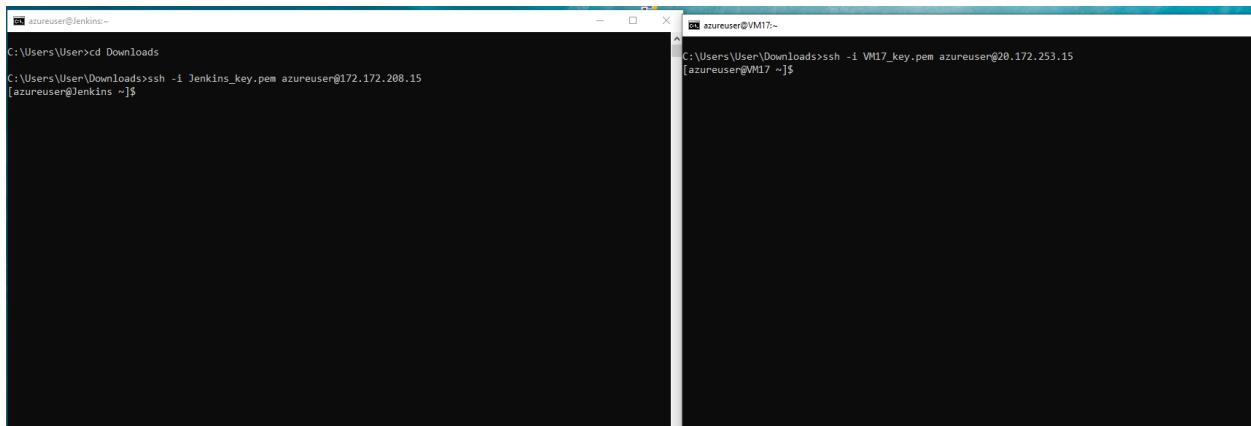
Follow the same process to create similar rule for the Production server. Failure, to do so will prevent the Train Schedule application loading.

<Production server public IP address>:8080

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Networking, Connect, Disks, Size, Microsoft Defender for Cloud, Advisor recommendations, Extensions + applications, Availability + scaling, and Configuration. Under the 'Connect' section, 'Native SSH' is highlighted. The main content area shows a 'Native SSH' configuration dialog. It has sections for 'Connecting using public IP address' (IP: 172.172.208.15), 'Admin username' (azureuser), 'Port (change)' (22), 'Just-in-time policy' (Not configured for port 22), and 'Recommended' and 'Most common' connection methods. The 'Native SSH' method is selected. To the right, there are sections for 'Public IP address' (checked), 'Open a local shell (on Windows)', 'Copy and execute SSH command' (with a text input field containing 'ssh -i ~/ssh/id_rsa.pem azureuser@172.172.208.15'), and 'Other Information'.

Once the networking rule is completed, go to Connect under Settings. Locate the **Native SSH** option and click **Select**.

Follow the instructions to launch a SSH connection with the server.



Note: please ensure that the command above points to where the .PEM key pair file resides.

```
Host '44.212.64.255' is not in the trusted hosts file.  
(ssh-ed25519 fingerprint sha1!! 18:50:bb:a7:54:fb:af:fe:44:10:bb:22:57:c8:26:4a:16:e1:8a:e9)  
Do you want to continue connecting? (y/n) y  
cloud_user@44.212.64.255's password:  
[cloud_user@ip-10-0-1-162 ~]$ cat /etc/os-release  
NAME="CentOS Linux"  
VERSION="7 (Core)"  
ID="centos"  
ID_LIKE="rhel fedora"  
VERSION_ID="7"  
PRETTY_NAME="CentOS Linux 7 (Core)"  
ANSI_COLOR="0;31"  
CPE_NAME="cpe:/o:centos:centos:7"  
HOME_URL="https://www.centos.org/"  
BUG_REPORT_URL="https://bugs.centos.org/"  
  
CENTOS_MANTISBT_PROJECT="CentOS-7"  
CENTOS_MANTISBT_PROJECT_VERSION="7"  
REDHAT_SUPPORT_PRODUCT="centos"  
REDHAT_SUPPORT_PRODUCT_VERSION="7"  
  
[cloud_user@ip-10-0-1-162 ~]$ █
```

The **cat /etc/os-release** commands should confirm the Linux OS version.

2. Commands to run on the Jenkins server using SSH

Install Jenkins

Update the server:

sudo yum update

Install Java and the epel-release package:

sudo yum install -y epel-release java-11-openjdk

Configure the Jenkins YUM repository:

sudo wget -O /etc/yum.repos.d/jenkins.repo <http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo>

Install the Jenkins key:

sudo rpm --import <https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key>

Install Jenkins:

sudo yum -y install jenkins

Enable Jenkins:

sudo systemctl enable jenkins

Start Jenkins:

sudo systemctl start Jenkins

Check the status of Jenkins:

sudo systemctl status jenkins

Post Jenkins Install

In a browser, navigate to the Jenkins IP address:

JENKINS_PUBLIC_IP:**8080**

On the server, find the temporary password:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Copy the temporary password.

In the browser, paste in the temporary admin password into the box provided.

Click **Install** suggested plugins.

Enter details for an "**admin**" user.

Click **Save and Continue**.

Click **Save and Finish**.

Click Start using Jenkins.

Install Docker Plugins

Manage Jenkins > Plugins

Go to **available** plugs. Search for Docker plugs.

Select all the Docker plugins.

Click on **Install**.

Select the option to reboot Jenkins after the install.

When the installation finishes go back to the SSH command prompt.

Install Git

```
sudo yum update
```

```
sudo yum install git
```

```
git --version
```

Install Docker

```
sudo yum update
```

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
sudo yum install -y docker-ce
```

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

```
sudo docker --version
```

Set permissions for Docker

```
docker run hello-world
```

```
docker: Got permission denied while trying to connect to the Docker daemon socket at  
unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.38/containers/create: dial  
unix /var/run/docker.sock: connect: permission denied.
```

See 'docker run --help'.

```
sudo chmod 666 /var/run/docker.sock
```

```
docker run hello-world
```

```
Unable to find image 'hello-world:latest' locally
```

```
latest: Pulling from library/hello-world
```

```
1b930d010525: Pull complete
```

```
Digest: sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
```

```
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

<https://hub.docker.com/>

For more examples and ideas, visit:

<https://docs.docker.com/get-started/>

Install sshpass

Install sshpass

```
sudo yum install sshpass
```

3.Commands to run on the Production server using SSH

Update the server

sudo yum update

Install Java and the epel-release package

```
sudo yum install -y epel-release java-11-openjdk
```

Install Git

```
sudo yum update
```

```
sudo yum install git
```

```
git --version
```

Install Docker

```
sudo yum update
```

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
sudo yum install -y docker-ce
```

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

```
sudo docker --version
```

Set permissions for Docker

When you run the following command on docker, by default, you will get the a permission denied error!

docker run hello-world

```
docker: Got permission denied while trying to connect to the Docker daemon socket at
unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.38/containers/create: dial
unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
```

sudo chmod 666 /var/run/docker.sock

docker run hello-world

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:c3b4ada4687bbba170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

<https://hub.docker.com/>

For more examples and ideas, visit:

<https://docs.docker.com/get-started/>

[Setup the Deploy account for Jenkins on the Production server.](#)

```
[azureuser@vm16 ~]$ whoami
```

azureuser

```
[azureuser@vm16 ~]$
```

Create a User:

```
sudo useradd -m deploy
```

Set a password:

sudo passwd deploy

Enter password

Confirm password

Note: we will use this later on the Jenkins portal.

List all user accounts

cat /etc/passwd

Host '54.210.24.6' is not in the trusted hosts file.

(ssh-ed25519 fingerprint sha1!! 5c:11:16:0b:b6:77:19:f9:aa:7c:64:01:02:c3:4d:97:e2:03:50:78)

Do you want to continue connecting? (y/n) y

cloud_user@54.210.24.6's password:

[cloud_user@ip-10-0-1-89 ~]\$ cat /etc/passwd

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

halt:x:7:0:halt:/sbin:/sbin/halt

mail:x:8:12:mail:/var/spool/mail:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin

games:x:12:100:games:/usr/games:/sbin/nologin

ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin

nobody:x:99:99:Nobody:/sbin/nologin

systemd-bus-proxy:x:999:998:systemd Bus Proxy:/sbin/nologin

systemd-network:x:192:192:systemd Network Management:/sbin/nologin

dbus:x:81:81:System message bus:/sbin/nologin

polkitd:x:998:997:User for polkitd:/sbin/nologin

tss:x:59:59:Account used by the trousers package to sandbox the tcscd daemon:/dev/null:/sbin/nologin

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

postfix:x:89:89::/var/spool/postfix:/sbin/nologin

chrony:x:997:995::/var/lib/chrony:/sbin/nologin

cloud_user:x:1000:1000:Cloud Assessment User:/home/cloud_user:/bin/bash

centos:x:1001:1001:Cloud User:/home/centos:/bin/bash

jenkins:x:996:993:Jenkins Automation Server:/var/lib/jenkins:/bin/false

deploy:x:1002:1002::/home/deploy:/bin/bash

[cloud_user@ip-10-0-1-89 ~]\$

Setup permissions for the Jenkins server so that it can SSH to the Production server.

When you run the following command you will get, by default, a permissions denied error as below

sshpass -p '<password for the deploy account on the production server>' -v ssh -o

StrictHostKeyChecking=no deploy@<production server Public IP Address>

```
[azureuser@Jenkins ~]$ sudo sshpass -p 'Harvee777$' -v ssh -o StrictHostKeyChecking=no  
deploy@20.172.253.15  
Warning: Permanently added '20.172.253.15' (ECDSA) to the list of known hosts.  
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).  
[azureuser@Jenkins ~]$
```

The solution to this is to enable password authentication.

Go to the Production server and use the following commands:

sudo vi /etc/ssh/sshd_config

In the file, find the **PasswordAuthentication** line and make sure it ends with **yes**.
Find the **ChallengeResponseAuthentication** option and disable it by adding **no**.
If lines are commented out, remove the hash sign # to uncomment them.

Switch to command mode by pressing the Esc key.

Type : (colon). This will open the prompt bar in the bottom left corner of the window.
Type wq after the colon and hit Enter. This will save in Vim the changes made to the file
and exit you out of the editor.

Now restart SSH with the following command:

```
sudo systemctl restart sshd
```

Now return to the Jenkins server and issue the same command as before.

sshpass -p '<password for the deploy account on the production server>' -v ssh -o

StrictHostKeyChecking=no deploy@<production server Public IP Address>

```
[azureuser@Jenkins ~]$ sudo sshpass -p 'Harvee777$' -v ssh -o StrictHostKeyChecking=no deploy@172.172.155.55  
SSHPASS searching for password prompt using match "assword"  
SSHPASS read: deploy@172.172.155.55's password:  
SSHPASS detected prompt. Sending password.  
SSHPASS read:  
  
[deploy@vm777 ~]$ logout  
Connection to 172.172.155.55 closed.  
[azureuser@Jenkins ~]$
```

It has the effect of creating an SSH session from the Jenkins server to the production server. Then logout to close the session.

The reason for the Jenkins to have these permissions is because it uses the above command to remote to the Production server then issue a docker command to pull the built docker container locally to the production server.

sshpass -p 'your_password_here' -v ssh -o StrictHostKeyChecking=no deploy@<production server public IP address> 'docker pull costas778/train-schedule'

NOTE: the password above is NOT one I will use for production purposes.

Deploying a Docker Container with a Jenkins Pipeline

We are, finally, in a position to deploy the Docker container using a Jenkins pipeline!

Go to **Manage Jenkins > Credentials**, then click **(global)**.

Click + Add Credentials

Create a global Jenkins credential called **deploy**

Kind: Username with password

Username: **deploy**

Password:

ID: **webserver_ID**

Description: **Webserver ID**

Use the same deploy password that you used for the production server.

Click **Create**

Create a global Jenkins credential for the Docker image registry (Docker Hub).

Click + Add Credentials:

Kind: Username with password

Username: <**DOCKER_HUB_USERNAME**>

Password: <**DOCKER_HUB_PASSWORD**>

ID: **docker_hub_login**

Description: **Docker Hub Login**

Click **Create**

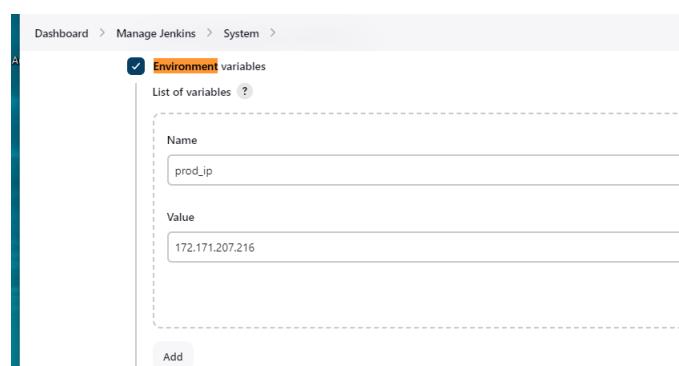
Note: You will need a Docker hub account to use Docker Hub as an image registry.

Configure a global property in Jenkins to store the production server IP by navigating to **Manage Jenkins > System** and adding an **environment variable** under **Global properties**.

Name: **prod_ip**

Value: <**PRODUCTION_SERVER_PUBLIC_IP_ADDRESS**>

Click **Save**



Go to GitHub

Make a personal fork from the following GitHub repository:

<https://github.com/ACloudGuru-Resources/cicd-pipeline-train-schedule-dockerdeploy>

https://github.com/<YOUR_GITHUB_USER>/cicd-pipeline-train-schedule-dockerdeploy

Generate a new **GitHub API key** to allow Jenkins to access the forked repo by navigating to **Profile > Settings > Developer Settings > Personal Access Tokens > Tokens (classic) > Generate new token (classic)**.

Note: **Jenkinsapp**

Permissions: **admin:repo_hook**

Copy the GitHub token.

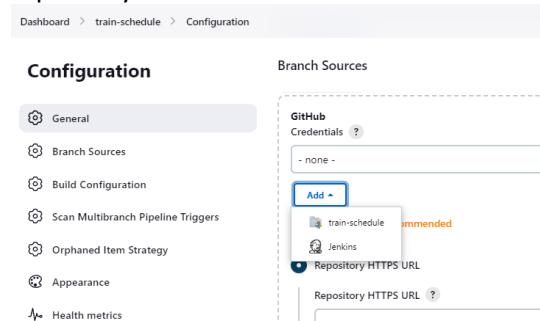
Go back to the Jenkins portal.

Click **New Items**

In Jenkins, create a **Multibranch Pipeline** project named **train-schedule**.

Click **OK**

Under **Branch Sources**, click **Add** below **GitHub**, then choose **Jenkins** to configure your forked repository.



The screenshot shows the Jenkins configuration interface for a Multibranch Pipeline project named 'train-schedule'. On the left, there's a sidebar with links like General, Branch Sources, Build Configuration, Scan Multibranch Pipeline Triggers, Orphaned Item Strategy, Appearance, and Health metrics. The 'Branch Sources' section is currently active. It displays a 'GitHub' source configuration with a 'Credentials' dropdown set to 'none'. Below it, there's an 'Add' button with a dropdown menu open, showing two options: 'train-schedule' (recommended) and 'Jenkins'. There's also a 'Repository HTTPS URL' field with a question mark icon.

Kind: Username with password

Scope: Global

Username: <GITHUB_USERNAME>

Password: <YOUR_GITHUB_PERSONAL_ACCESS_TOKEN>

ID: **github_key**

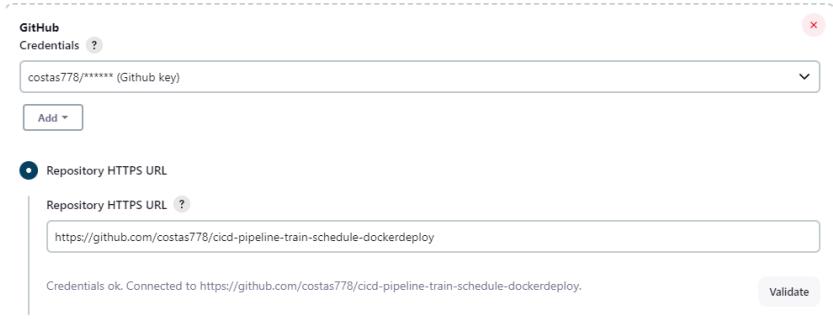
Description: **GitHub Key**

Click **Add**

Select the **GitHub Key** and paste your forked repository under **Repository HTTPS URL**:

https://github.com/<YOUR_GITHUB_USER>/cicd-pipeline-train-schedule-dockerdeploy

Click **Validate** to test your credentials.



Click **Save**.

Successfully deploy the train-schedule app to production as a Docker container using the Jenkins Pipeline.

Modify the **Jenkinsfile** in GitHub to build and push the Docker image to Docker Hub and commit the changes. Replace `willbla` from the example file with your own Docker Hub username.

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo 'Running build automation'
                sh './gradlew build --no-daemon'
                archiveArtifacts artifacts: 'dist/trainSchedule.zip'
            }
        }
        stage('Build Docker Image') {
            when {
                branch 'master'
            }
            steps {
                script {
                    app = docker.build("<DOCKER_HUB_USERNAME>/train-schedule")
                    app.inside {
                        sh 'echo $(curl localhost:8080)'
                    }
                }
            }
        }
        stage('Push Docker Image') {
            when {
                branch 'master'
            }
            steps {
                script {
```

```
        docker.withRegistry('https://registry.hub.docker.com', 'docker_hub_login') {
            app.push("${env.BUILD_NUMBER}")
            app.push("latest")
        }
    }
}
}
}
```

In Jenkins, click **Build Now**.

Note: The initial build may take several minutes to complete.

In **Docker Hub**, under **Repositories**, select the **train-schedule app**.

Click the **Tags tab** to verify that the build was pushed successfully.

In GitHub, modify the **Jenkinsfile** to include a stage that pushes the build to the production server, and commit the changes.

```
stage ('DeployToProduction') {

when {
    branch 'master'
}

steps {
    input 'Deploy to Production'
    milestone(1)
    withCredentials ([usernamePassword(credentialsId: 'webserver_login', usernameVariable:
'USERNAME', passwordVariable: 'USERPASS')]) {
        script {
```

```

sh "sshpass -p '$USERPASS' -v ssh -o StrictHostKeyChecking=no $USERNAME@${env.prod_ip}
\"docker pull <DOCKER_HUB_USERNAME>/train-schedule:${env.BUILD_NUMBER}\"
try {
    sh "sshpass -p '$USERPASS' -v ssh -o StrictHostKeyChecking=no $USERNAME@${env.prod_ip}
    \"docker stop train-schedule\"
    sh "sshpass -p '$USERPASS' -v ssh -o StrictHostKeyChecking=no $USERNAME@${env.prod_ip}
    \"docker rm train-schedule\"
} catch (err) {
    echo: 'caught error: $err'
}
sh "sshpass -p '$USERPASS' -v ssh -o StrictHostKeyChecking=no $USERNAME@${env.prod_ip}
\"docker run --restart always --name train-schedule -p 8080:8080 -d <DOCKER_HUB_USERNAME>/train-
schedule:${env.BUILD_NUMBER}\"
}
}
}

```

Place your Docker hub username above and **commit** the changes.
In Jenkins, click **Build Now**.

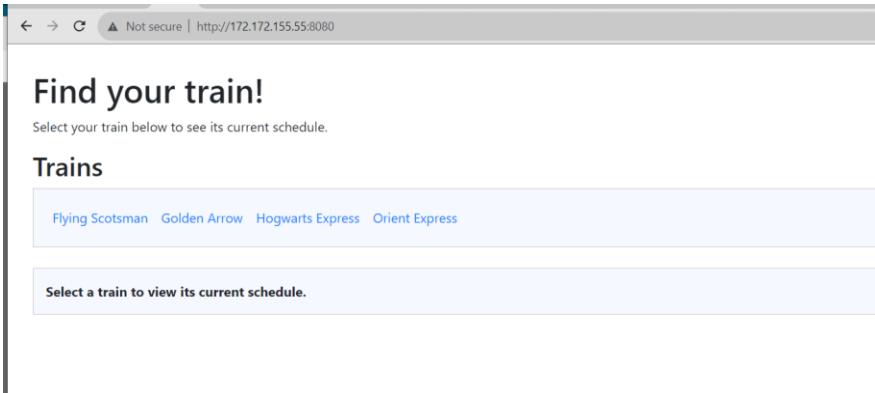
The screenshot shows the Jenkins Pipeline Stage View for the 'Branch master' branch of the 'train-schedule' project. The pipeline consists of five stages: Declarative: Checkout SCM, Build, Build Docker Image, Push Docker Image, and DeployToProduction. The Push Docker Image stage is currently active, indicated by a green background. The Build stage has a duration of 1min 9s. The Push Docker Image stage has a duration of 7s. The DeployToProduction stage has a duration of 26s. The pipeline was last updated on Sept 15 at 11:50 with no changes.

Note: to proceed from Push Docker Image to **DeployToProduction** you will need to hover over Push Docker Image within the green area and allow it to proceed further.

Once the build is complete, using a web browser, verify that the application has been deployed successfully.

The screenshot shows the Jenkins Pipeline Stage View for the 'Branch master' branch of the 'train-schedule' project. The pipeline stages are now all highlighted in green, indicating they have been completed. The Build stage took 1min 9s, the Push Docker Image stage took 7s, and the DeployToProduction stage took 1min 38s. The pipeline was last updated on Sept 15 at 11:50 with no changes.

Place the following in a web browser <PRODUCTION_SERVER_PUBLIC_IP_ADDRESS>:8080



If successful, you will see the above!

Addendum

1) Build Failure

Started by user Costas constantinou

08:44:29 Connecting to <https://api.github.com> using costas778/******** (Github key)

Obtained Jenkinsfile from 248d9bbd2a5c031c08db8c6b21a28f6336cc4173

[Pipeline] Start of Pipeline

[Pipeline] node

Running on Jenkins in /var/lib/jenkins/workspace/train-schedule_master

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Declarative: Checkout SCM)

[Pipeline] checkout

The recommended git tool is: NONE

using credential github_key

> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/train-schedule_master/.git # timeout=10

Fetching changes from the remote Git repository

> git config remote.origin.url <https://github.com/costas778/cicd-pipeline-train-schedule-dockerdeploy.git> # timeout=10

Fetching without tags

Fetching upstream changes from <https://github.com/costas778/cicd-pipeline-train-schedule-dockerdeploy.git>

> git --version # timeout=10

> git --version # 'git version 1.8.3.1'

using GIT_ASKPASS to set credentials Github key

> git fetch --no-tags --progress <https://github.com/costas778/cicd-pipeline-train-schedule-dockerdeploy.git> +refs/heads/master:refs/remotes/origin/master # timeout=10

Checking out Revision 248d9bbd2a5c031c08db8c6b21a28f6336cc4173 (master)

> git config core.sparsecheckout # timeout=10

> git checkout -f 248d9bbd2a5c031c08db8c6b21a28f6336cc4173 # timeout=10

Commit message: "Update Jenkinsfile"

> git rev-list --no-walk 33c78dcc8603870d855e4caa9653c89b7848b739 # timeout=10

[Pipeline] }

```
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Running build automation
[Pipeline] sh
+ ./gradlew build --no-daemon
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.vmplugin.v7.Java7$1
(file:/var/lib/jenkins/.gradle/wrapper/dists/gradle-6.0.1-bin/1lxlpkiy24sb18odw96cp4ojv/gradle-6.0.1/lib/groovy-all-1.3-2.5.8.jar) to constructor
java.lang.invoke.MethodHandles$Lookup(java.lang.Class,int)
WARNING: Please consider reporting this to the maintainers of
org.codehaus.groovy.vmplugin.v7.Java7$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
> Task :nodeSetup UP-TO-DATE
> Task :npmSetup UP-TO-DATE
> Task :npmInstall UP-TO-DATE
> Task :npm_test
> cicd-pipeline-train-schedule-git@0.0.0 test /var/lib/jenkins/workspace/train-schedule_master
> mocha
Index Page
[0mGET / [32m200 [0m186.268 ms - 829[0m
    ✓ renders successfully (205ms)
```

Trains API

```
[0mGET /trains [32m200 [0m2.163 ms - 1093[0m
    ✓ returns data successfully
2 passing (227ms)
> Task :npm_build
> Task :zip
> Task :build
```

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.

See

https://docs.gradle.org/6.0.1/userguide/command_line_interface.html#sec:command_line_warnings

```
BUILD SUCCESSFUL in 7s
6 actionable tasks: 3 executed, 3 up-to-date
[Pipeline] archiveArtifacts
Archiving artifacts
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
```

```
[Pipeline] { (Build Docker Image)
[Pipeline] script
[Pipeline] {
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Push Docker Image)
Stage "Push Docker Image" skipped due to earlier failure(s)
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (DeployToProduction)
Stage "DeployToProduction" skipped due to earlier failure(s)
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Also: org.jenkinsci.plugins.workflow.actions.ErrorAction$ErrorId: 78a67a41-c4f8-405c-a52c-f0f33958a0bf
groovy.lang.MissingPropertyException: No such property: docker for class: groovy.lang.Binding
    at groovy.lang.Binding.getVariable(Binding.java:63)
    at
    org.jenkinsci.plugins.scriptsecurity.sandbox.groovy.SandboxInterceptor.onGetProperty(SandboxIn
terceptor.java:285)
    at org.kohsuke.groovy.sandbox.impl.Checker$7.call(Checker.java:375)
    at org.kohsuke.groovy.sandbox.impl.Checker.checkedGetProperty(Checker.java:379)
    at org.kohsuke.groovy.sandbox.impl.Checker.checkedGetProperty(Checker.java:355)
    at org.kohsuke.groovy.sandbox.impl.Checker.checkedGetProperty(Checker.java:355)
    at org.kohsuke.groovy.sandbox.impl.Checker.checkedGetProperty(Checker.java:355)
    at com.cloudbees.groovy.cps.sandbox.SandboxInvoker.getProperty(SandboxInvoker.java:29)
    at org.jenkinsci.plugins.workflow.cps.LoggingInvoker.getProperty(LoggingInvoker.java:121)
    at com.cloudbees.groovy.cps.impl.PropertyAccessBlock.rawGet(PropertyAccessBlock.java:20)
    at WorkflowScript.run(WorkflowScript:17)
    at __cps.transform__(Native Method)
    at
    com.cloudbees.groovy.cps.impl.PropertyishBlock$ContinuationImpl.get(PropertyishBlock.java:73)
    at com.cloudbees.groovy.cps.LValueBlock$GetAdapter.receive(LValueBlock.java:30)
    at
    com.cloudbees.groovy.cps.impl.PropertyishBlock$ContinuationImpl.fixName(PropertyishBlock.jav
a:65)
    at jdk.internal.reflect.GeneratedMethodAccessor386.invoke(Unknown Source)
```

```
at
java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.base/java.lang.reflect.Method.invoke(Method.java:566)
at
com.cloudbees.groovy.cps.impl.ContinuationPtr$ContinuationImpl.receive(ContinuationPtr.java:72)
at com.cloudbees.groovy.cps.impl.ConstantBlock.eval(ConstantBlock.java:21)
at com.cloudbees.groovy.cps.Next.step(Next.java:83)
at com.cloudbees.groovy.cps.Continuable$1.call(Continuable.java:152)
at com.cloudbees.groovy.cps.Continuable$1.call(Continuable.java:146)
at
org.codehaus.groovy.runtime.GroovyCategorySupport$ThreadCategoryInfo.use(GroovyCategorySupport.java:136)
at org.codehaus.groovy.runtime.GroovyCategorySupport.use(GroovyCategorySupport.java:275)
at com.cloudbees.groovy.cps.Continuable.run0(Continuable.java:146)
at
org.jenkinsci.plugins.workflow.cps.SandboxContinuable.access$001(SandboxContinuable.java:18)
at org.jenkinsci.plugins.workflow.cps.SandboxContinuable.run0(SandboxContinuable.java:51)
at org.jenkinsci.plugins.workflow.cps.CpsThread.runNextChunk(CpsThread.java:187)
at org.jenkinsci.plugins.workflow.cps.CpsThreadGroup.run(CpsThreadGroup.java:422)
at org.jenkinsci.plugins.workflow.cps.CpsThreadGroup$2.call(CpsThreadGroup.java:330)
at org.jenkinsci.plugins.workflow.cps.CpsThreadGroup$2.call(CpsThreadGroup.java:294)
at
org.jenkinsci.plugins.workflow.cps.CpsVmExecutorService$2.call(CpsVmExecutorService.java:97)
at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
at hudson.remoting.SingleLaneExecutorService$1.run(SingleLaneExecutorService.java:139)
at jenkins.util.ContextResettingExecutorService$1.run(ContextResettingExecutorService.java:28)
at jenkins.security.ImpersonatingExecutorService$1.run(ImpersonatingExecutorService.java:68)
at jenkins.util.ErrorLoggingExecutorService.lambda$wrap$0(ErrorLoggingExecutorService.java:51)
at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515)
at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128)
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)
at java.base/java.lang.Thread.run(Thread.java:829)
```

Could not update commit status, please check if your scan credentials belong to a member of the organization or a collaborator of the repository and repo:status scope is selected
GitHub has been notified of this commit's build result.

Finished: FAILURE

Solution:

Make sure that the **Docker plugins** are installed and configured correctly on your Jenkins server.

2) Failure to Deploy to Production (Stage)

```
Shell Script -- sshpass -p '${USERPASS}' -v ssh -o StrictHostKeyChecking=no  
jenkinsad@172.171.207.216 "docker pull costas778/train-schedule:3" (self time 288ms)
```

Warning: A secret was passed to "sh" using Groovy String interpolation, which is insecure.

Affected argument(s) used the following variable(s): [USERPASS]

See <https://jenkins.io/redirect/groovy-string-interpolation> for details.

```
+ sshpass -p **** -v ssh -o StrictHostKeyChecking=no jenkinsad@172.171.207.216 'docker pull  
costas778/train-schedule:3'  
/var/lib/jenkins/workspace/train-schedule_master@tmp/durable-b344b4a1/script.sh: line 1: sshpass:  
command not found
```

Solution:

Install SSH with the following command.

```
sudo yum install sshpass
```

3) Failure to Deploy To Production (Stage) after installing sshpass

Warning: A secret was passed to "sh" using Groovy String interpolation, which is insecure.

Affected argument(s) used the following variable(s): [USERPASS]

See <https://jenkins.io/redirect/groovy-string-interpolation> for details.

```
+ sshpass -p **** -v ssh -o StrictHostKeyChecking=no jenkinsad@172.171.207.216 'docker pull  
costas778/train-schedule:4'
```

Warning: Permanently added '172.171.207.216' (ECDSA) to the list of known hosts.

Permission denied (publickey,gssapi-keyex,gssapi-with-mic).

Solution:

Go to sshd_config file on the production server and make the following changes:

```
sudo vi /etc/ssh/sshd_config
```

In the file, find the **PasswordAuthentication** line and make sure it ends with yes.

Find the **ChallengeResponseAuthentication** option and disable it by adding no.

If lines are commented out, remove the hash sign # to uncomment them.

Save the changes in the file then restart the SSH service for the changes to take effect.

```
sudo systemctl restart sshd
```

4) Build error - Failure to download Node with Gradle.

```
+ ./gradlew build --no-daemon
```

```
Downloading https://services.gradle.org/distributions/gradle-6.0.1-bin.zip
```

```
.....  
Unzipping /var/lib/jenkins/.gradle/wrapper/dists/gradle-6.0.1-bin/1lxlpkiy24sb18odw96cp4ojv/gradle-  
6.0.1-bin.zip to /var/lib/jenkins/.gradle/wrapper/dists/gradle-6.0.1-bin/1lxlpkiy24sb18odw96cp4ojv
```

```
Set executable permissions for: /var/lib/jenkins/.gradle/wrapper/dists/gradle-6.0.1-  
bin/1lxlpkiy24sb18odw96cp4ojv/gradle-6.0.1/bin/gradle
```

Welcome to Gradle 6.0.1!

Here are the highlights of this release:

- Substantial improvements in dependency management, including
 - Publishing Gradle Module Metadata in addition to pom.xml
 - Advanced control of transitive versions
 - Support for optional features and dependencies
 - Rules to tweak published metadata
- Support for Java 13
- Faster incremental Java and Groovy compilation
- New Zinc compiler for Scala
- VS2019 support
- Support for Gradle Enterprise plugin 3.0

For more details see <https://docs.gradle.org/6.0.1/release-notes.html>

WARNING: An illegal reflective access operation has occurred

WARNING: Illegal reflective access by org.codehaus.groovy.vmplugin.v7.Java7\$1
(file:/var/lib/jenkins/.gradle/wrapper/dists/gradle-6.0.1-bin/1lxlpkiy24sb18odw96cp4ojv/gradle-6.0.1/lib/groovy-all-1.3-2.5.8.jar) to constructor

java.lang.invoke.MethodHandles\$Lookup(java.lang.Class,int)

WARNING: Please consider reporting this to the maintainers of

org.codehaus.groovy.vmplugin.v7.Java7\$1

WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations

WARNING: All illegal access operations will be denied in a future release

> Task :nodeSetup

> Task :nodeSetup FAILED

FAILURE: Build failed with an exception.

* What went wrong:

Execution failed for task ':nodeSetup'.

> Could not resolve all files for configuration ':detachedConfiguration1'.

 > Could not download node-9.11.1-linux-x64.tar.gz (org.nodejs:node:9.11.1)

 > Could not get resource '<https://nodejs.org/dist/v9.11.1/node-v9.11.1-linux-x64.tar.gz>'.

 > Read timed out

* Try:

Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output. Run with --scan to get full insights.

* Get more help at <https://help.gradle.org>

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.

Use '--warning-mode all' to show the individual deprecation warnings.

See

https://docs.gradle.org/6.0.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD FAILED in 10m 17s

1 actionable task: 1 executed

Solution:

This is, likely, down to internet issues.

Select **Run Build** again.

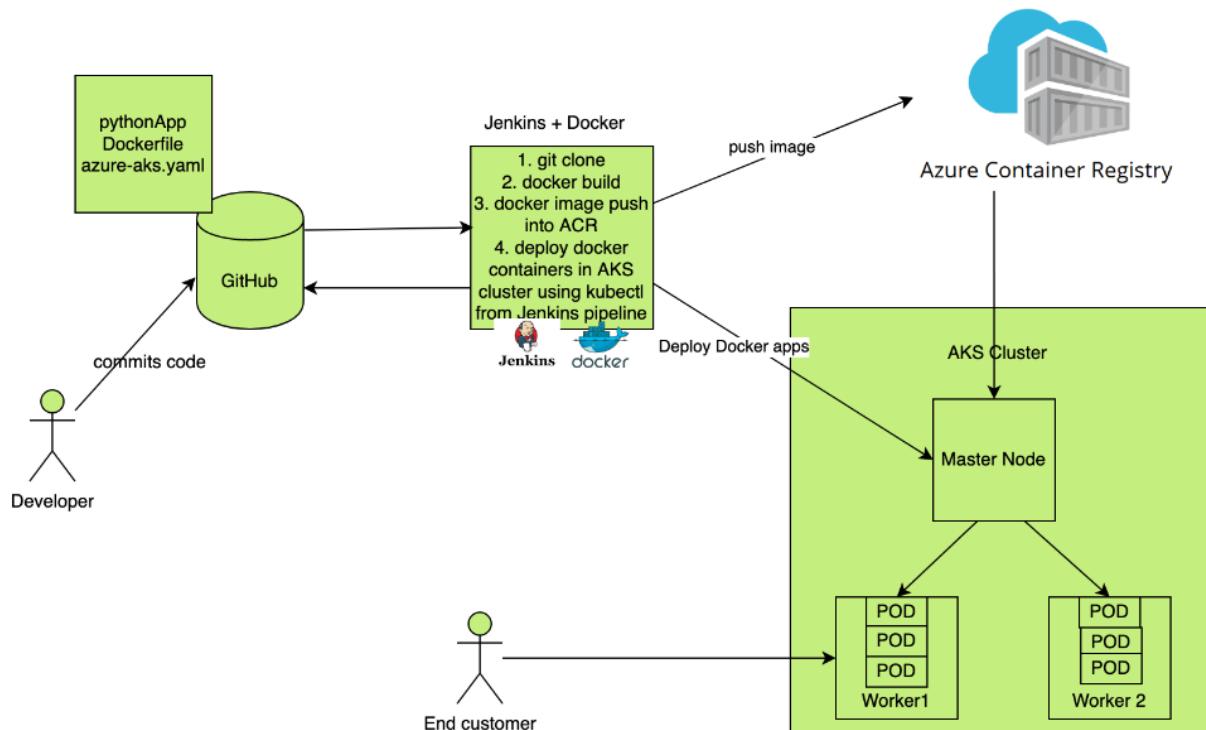
If this still fails then perform troubleshooting on your internet connection!

Alternatively, download the following manually and install it.

```
sudo wget https://nodejs.org/dist/v9.11.1/node-v9.11.1-linux-x64.tar.gz
https://nodejs.org/en/download
```

Deploy Python App into Kubernetes Cluster using Jenkins Pipeline | Containerize Python App and Deploy into AKS Cluster

Kubernetes(AKS) Deployment from Jenkins pipeline using Kubectl CLI



Pre-requisites:

Create an Ubuntu 22.0.4 LTS machine on the Azure Portal.

Remote into the Ubuntu machine using a keypair from your command prompt.

1. AKS Cluster is setup and running.

First check to see if Azure CLI is installed on your local machine.

Use the command, **az version** to confirm.

If you do not see a version of **azure-cli** then proceed to the following instructions:

Run the update first

sudo apt update

Second install with one command

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

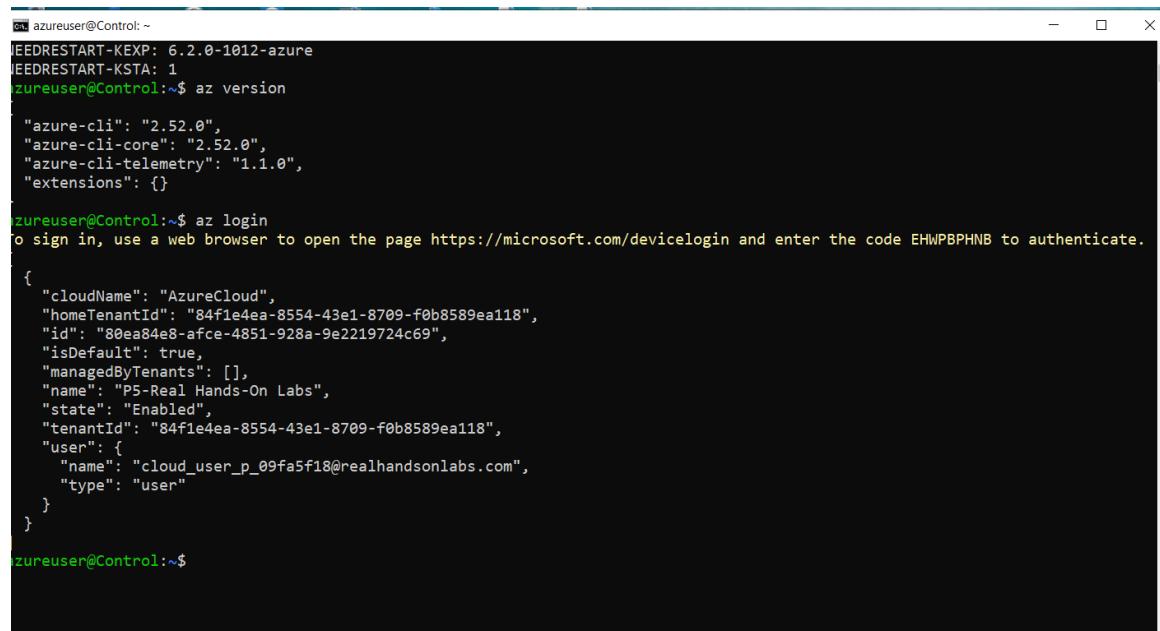
Check again to see if the system has a version of Azure CLI

az version

Second, insure you have an account setup in Azure cloud.

To sign in, use the **az login** command and follow the instructions in the command prompt.

az login

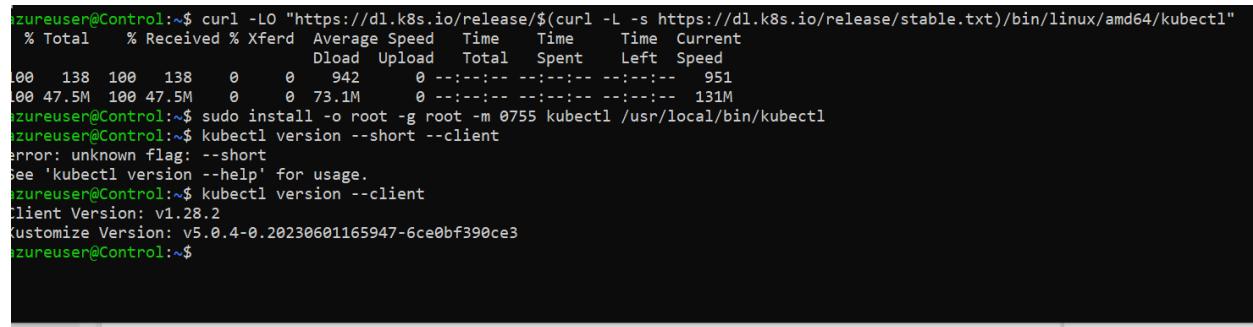


```
azureuser@Control:~$ az login
o sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code EHWPBPHNB to authenticate.

{
  "cloudName": "AzureCloud",
  "homeTenantId": "84f1e4ea-8554-43e1-8709-f0b8589ea118",
  "id": "80ea84e8-afce-4851-928a-e2219724c69",
  "isDefault": true,
  "managedByTenants": [],
  "name": "P5-Real Hands-On Labs",
  "state": "Enabled",
  "tenantId": "84f1e4ea-8554-43e1-8709-f0b8589ea118",
  "user": {
    "name": "cloud_user_p_09fa5f18@realhandsonlabs.com",
    "type": "user"
  }
}

azureuser@Control:~$
```

Third, Install kubectl



```
azureuser@Control:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100 138 100 138  0     0  942  0 --:--:-- --:--:-- --:--:--  951
100 47.5M 100 47.5M  0     0  73.1M  0 --:--:-- --:--:-- --:--:-- 131M
azureuser@Control:~$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
azureuser@Control:~$ kubectl version --short --client
error: unknown flag: --short
See 'kubectl version --help' for usage.
azureuser@Control:~$ kubectl version --client
Client Version: v1.28.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
azureuser@Control:~$
```

Install kubectl binary with curl on Linux

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

Install kubectl

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Verify if kubectl got installed

```
kubectl version --short --client
```

```
ubuntu@ip-172-31-28-83:~$ kubectl version --short --client
Flag --short has been deprecated, and will be removed in the
become the default.
Client Version: v1.24.0
Kustomize Version: v4.5.4
ubuntu@ip-172-31-28-83:~$
```

Now let's setup the AKS cluster

Step 1

Make sure, once again, you are logged into the Azure portal first with
The following command:

```
az account show -o jsonc
```

```
azureuser@Control:~$ az account show -o jsonc
{
  "environmentName": "AzureCloud",
  "homeTenantId": "84f1e4ea-8554-43e1-8709-f0b8589ea118",
  "id": "80ea84e8-afce-4851-928a-9e2219724c69",
  "isDefault": true,
  "managedByTenants": [],
  "name": "PS-Real Hands-On Labs",
  "state": "Enabled",
  "tenantId": "84f1e4ea-8554-43e1-8709-f0b8589ea118",
  "user": {
    "name": "cloud_user_p_09fa5f18@realhandsonlabs.com",
    "type": "user"
  }
}azureuser@Control:~$
```

Step 2 - create a resource group first

```
az group create --name myResourceGroup --location southcentralus
```

Step 3 Create AKS cluster with 2 worker nodes

```
az aks create --resource-group myResourceGroup --name myAKSCluster --node-count 2
```

```
az aks create --resource-group 1-07ee85d4-playground-sandbox --name myAKSCluster --node-count 2
```

NOTE: if you do not include the switch or flag, **--node-count 2** it will, by default, generate 2 nodes.

```
}
```

```
azureuser@Control:~$ az aks create --resource-group 1-07ee85d4-playground-sandbox --name myAKSCluster --node-count 2
■| Running ..
```

```
E
```

Output:

```
azureuser@Control:~$ az aks create --resource-group 1-07ee85d4-playground-sandbox --name
myAKSCluster --node-count 2
{
```

```
"aadProfile": null,
"addonProfiles": null,
"agentPoolProfiles": [
{
"availabilityZones": null,
"count": 2,
"creationData": null,
"currentOrchestratorVersion": "1.26.6",
"enableAutoScaling": false,
"enableEncryptionAtHost": false,
"enableFips": false,
"enableNodePublicIp": false,
"enableUltraSsd": false,
"gpuInstanceProfile": null,
"hostGroupId": null,
"kubeletConfig": null,
"kubeletDiskType": "OS",
"linuxOsConfig": null,
"maxCount": null,
"maxPods": 110,
"minCount": null,
"mode": "System",
"name": "nodepool1",
"nodeImageVersion": "AKSUbuntu-2204gen2containerd-202308.22.0",
"nodeLabels": null,
"nodePublicIpPrefixId": null,
"nodeTaints": null,
"orchestratorVersion": "1.26.6",
"osDiskSizeGb": 128,
"osDiskType": "Managed",
"osSku": "Ubuntu",
"osType": "Linux",
"podSubnetId": null,
"powerState": {
  "code": "Running"
},
"provisioningState": "Succeeded",
"proximityPlacementGroupId": null,
"scaleDownMode": null,
"scaleSetEvictionPolicy": null,
"scaleSetPriority": null,
"spotMaxPrice": null,
"tags": null,
"type": "VirtualMachineScaleSets",
"upgradeSettings": {
  "drainTimeoutInMinutes": null,
  "maxSurge": null
},
},
```

```
"vmSize": "Standard_DS2_v2",
"vnetSubnetId": null,
"workloadRuntime": null
},
],
"apiServerAccessProfile": null,
"autoScalerProfile": null,
"autoUpgradeProfile": {
"nodeOsUpgradeChannel": "NodeImage",
"upgradeChannel": null
},
"azureMonitorProfile": null,
"azurePortalFqdn": "myaksclust-1-07ee85d4-playg-80ea84-qkr1nwg3.portal.hcp.westus.azmk8s.io",
"currentKubernetesVersion": "1.26.6",
"disableLocalAccounts": false,
"diskEncryptionSetId": null,
"dnsPrefix": "myAKSClust-1-07ee85d4-playg-80ea84",
"enablePodSecurityPolicy": null,
"enableRbac": true,
"extendedLocation": null,
"fqdn": "myaksclust-1-07ee85d4-playg-80ea84-qkr1nwg3.hcp.westus.azmk8s.io",
"fqdnSubdomain": null,
"httpProxyConfig": null,
"id": "/subscriptions/80ea84e8-afce-4851-928a-9e2219724c69/resourcegroups/1-07ee85d4-
playground-sandbox/providers/Microsoft.ContainerService/managedClusters/myAKSCluster",
"identity": {
"delegatedResources": null,
"principalId": "26942b51-f827-4b6e-b0b0-6da6368b5e56",
"tenantId": "84f1e4ea-8554-43e1-8709-f0b8589ea118",
"type": "SystemAssigned",
"userAssignedIdentities": null
},
"identityProfile": {
"kubeletIdentity": {
"clientId": "71c5b1ff-8c95-4cd8-b461-372eee4e2ffe",
"objectId": "3cc16e6d-7e41-4097-a312-aadb5ae2adc0",
"resourceId": "/subscriptions/80ea84e8-afce-4851-928a-9e2219724c69/resourcegroups/MC_1-
07ee85d4-playground-
sandbox_myAKSCluster_westus/providers/Microsoft.ManagedIdentity/userAssignedIdentities/myAKSCL
uster-agentpool"
}
},
"kubernetesVersion": "1.26.6",
"linuxProfile": {
"adminUsername": "azureuser",
"ssh": {
"publicKeys": [
{

```

```
        "keyData": "ssh-rsa
AAAAAB3NzaC1yc2EAAAQABAAQGrNWX7252YzD3Rr95Yq1NRommZLti9mmjcMEQYeHujWmy4
VyPV0eUdXd25rzzXk5CxG7ZJyTqvvmz+lubS/6L40blYll/Z4ckBrDRt9CEbJGDEYwb7qM9pNyoXtW0W+XfV4
Xpf/XIOG1OS/ei7DUs8mo8kNLwEyZXKr2ZvTpX51o9N7Ahvr81aiDLX6fcxyIL/x6zu57Xe1Zy2zqxhI/GfjdDQ4
HooejXndqqy21X1B8hoMT3jFWTfwTvQr5ItJmJswQpNmFa+WDV7166hQHroe7XBp6/GT9yECoP7Q331e
MKzUJ7X64JwEAUqmKhlzvmN6lzk+7OBau8X0cAnhMx"
    }
]
}
},
"location": "westus",
"maxAgentPools": 100,
"name": "myAKSCluster",
"networkProfile": {
"dnsServiceIp": "10.0.0.10",
"ipFamilies": [
"IPv4"
],
"loadBalancerProfile": {
"allocatedOutboundPorts": null,
"effectiveOutboundIPs": [
{
"id": "/subscriptions/80ea84e8-afce-4851-928a-9e2219724c69/resourceGroups/MC_1-07ee85d4-
playground-
sandbox_myAKSCluster_westus/providers/Microsoft.Network/publicIPAddresses/e8443d68-4b32-48fa-
bd74-71c057458ab5",
"resourceGroup": "MC_1-07ee85d4-playground-sandbox_myAKSCluster_westus"
}
],
"enableMultipleStandardLoadBalancers": null,
"idleTimeoutInMinutes": null,
"managedOutboundIPs": {
"count": 1,
"countIpv6": null
},
"outboundIPs": null,
"outboundIpPrefixes": null
},
"loadBalancerSku": "Standard",
"natGatewayProfile": null,
"networkDataplane": null,
"networkMode": null,
"networkPlugin": "kubernetes",
"networkPluginMode": null,
"networkPolicy": null,
"outboundType": "loadBalancer",
"podCidr": "10.244.0.0/16",
"podCidrs": [
```

```
"10.244.0.0/16"
],
"serviceCidr": "10.0.0.0/16",
"serviceCidrs": [
  "10.0.0.0/16"
]
},
"nodeResourceGroup": "MC_1-07ee85d4-playground-sandbox_myAKSCluster_westus",
"oidcIssuerProfile": {
  "enabled": false,
  "issuerUrl": null
},
"podIdentityProfile": null,
"powerState": {
  "code": "Running"
},
"privateFqdn": null,
"privateLinkResources": null,
"provisioningState": "Succeeded",
"publicNetworkAccess": null,
"resourceGroup": "1-07ee85d4-playground-sandbox",
"securityProfile": {
  "azureKeyVaultKms": null,
  "defender": null,
  "imageCleaner": null,
  "workloadIdentity": null
},
"servicePrincipalProfile": {
  "clientId": "msi",
  "secret": null
},
"sku": {
  "name": "Base",
  "tier": "Free"
},
"storageProfile": {
  "blobCsiDriver": null,
  "diskCsiDriver": {
    "enabled": true
  },
  "fileCsiDriver": {
    "enabled": true
  },
  "snapshotController": {
    "enabled": true
  }
},
"supportPlan": "KubernetesOfficial",
```

```

"systemData": null,
"tags": null,
"type": "Microsoft.ContainerService/ManagedClusters",
"upgradeSettings": null,
"windowsProfile": null,
"workloadAutoScalerProfile": {
    "keda": null,
    "verticalPodAutoscaler": null
}
}
}

```

Display Details of Cluster

```

az aks show --name myAKSCluster --resource-group myResourceGroup
az aks show --name myAKSCluster --resource-group 1-07ee85d4-playground-sandbox

```

The above command will display Cluster details.

Node	Status	CPU	Memory	Disk	Pods	Kubernetes version
aks-nodepool1-35897272-vmss000000	Ready	10%	28%	16%	10	1.26.6
aks-nodepool1-35897272-vmss000001	Ready	10%	30%	16%	12	1.26.6

The two nodes that we created can be seen above!

2. Jenkins Master and Slave is up and running.

Jenkins Master

Change Host Name to Jenkins

```
sudo hostnamectl set-hostname Jenkins
```

Perform update first

```
sudo apt update
```

Install Java 11

```
sudo apt install default-jdk -y
```

```
ubuntu@ip-172-31-44-89:~$ sudo apt install default-jdk -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
at-spi2-core default-jdk-headless default-jre fonts-dejavu-extra libatk-bridge2.0-0
libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0
libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontenc1 libgif7 libgl1
libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libice-dev libice6
libllvm10 libpciaccess0 libpthread-stubs0-dev libsensors4 libsm-dev libsm6 libx11-dev
libx11-doc libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0
libxcb-present0 libxcb-shape0 libxcb-sync1 libxcb1-dev libxcomposite1 libxdamage1
```

Once install java, enter the below command.

Verify Java Version

java -version

```
ubuntu@ip-172-31-19-22:/opt$ java -version
openjdk version "11.0.6" 2020-01-14
OpenJDK Runtime Environment (Build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1)
OpenJDK 64-Bit Server VM (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1, mixed mode, sharing)
ubuntu@ip-172-31-19-22:/opt$
```

Maven Installation

You can install Maven by executing below command:

sudo apt install maven -y

Now type the following:

mvn --version

```
ubuntu@ip-172-31-34-93:~$ mvn --version
Apache Maven 3.6.0
Maven home: /usr/share/maven
Java version: 11.0.7, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-1065-aws", arch: "amd64", family: "unix"
ubuntu@ip-172-31-34-93:~$
```

Now let's start the Jenkins installation.

Jenkins Setup

Add a Repository key to the system

```
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

```
ubuntu@ip-172-31-28-134:~$ curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

Append debian package repo address to the system

```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
ubuntu@ip-172-31-28-138:~$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \n https://pkg.jenkins.io/debian binary/ | sudo tee \n /etc/apt/sources.list.d/jenkins.list > /dev/null
```

Update Ubuntu package

sudo apt update

```
ubuntu@ip-172-31-211:~$ sudo apt-get update\nReading package lists... Done\nBuilding dependency tree\nReading state information... Done\nFetched 15.5 kB in 0s (17.8 kB/s)\nReading package lists... Done
```

Install Jenkins

sudo apt install jenkins -y

```
ubuntu@ip-172-31-211:~$ sudo apt-get install jenkins -y\nReading package lists... Done\nBuilding dependency tree\nReading state information... Done\nThe following additional packages will be installed:\n  daemon\nThe following NEW packages will be installed:\n  daemon jenkins\n0 upgraded, 2 newly installed, 0 to remove and 8 not upgraded.\nNeed to get 72.0 MB of archives.\nAfter this operation, 75.1 MB of additional disk space will be used.\nGet:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu xenial/universe amd64 daemon\nGet:2 http://pkg.jenkins.io/debian-stable binary/ jenkins 2.121.3 [71.9 MB]\nFetched 72.0 MB in 5s (12.5 MB/s)\nSelecting previously unselected package daemon.\n(Reading database ... 51843 files and directories currently installed.)\nPreparing to unpack .../daemon_0.6.4-1_amd64.deb ...\nUnpacking daemon (0.6.4-1) ...\nSelecting previously unselected package jenkins.\nPreparing to unpack .../jenkins_2.121.3_all.deb ...\nUnpacking jenkins (2.121.3) ...\nProcessing triggers for man-db (2.7.5-1) ...\nProcessing triggers for systemd (229-4ubuntu21.4) ...\nProcessing triggers for ureadahead (0.100.0-19) ...\nSetting up daemon (0.6.4-1) ...\nSetting up jenkins (2.121.3) ...\nProcessing triggers for systemd (229-4ubuntu21.4) ...\nProcessing triggers for ureadahead (0.100.0-19) ...\nubuntu@ip-172-31-211:~$
```

The above screenshot should confirm that Jenkins is successfully installed.

Allow Access in Jenkins in a web browser

Create an inbound rule within **Settings > Networking** for port **8080**.

Now go to browser and enter the virtual machines public IP address with port no **8080**.

http://Public IP address:8080

Unlock Jenkins

When this page loads, enter the below command in Git bash (Ubuntu console)

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Get the initial password from the below file

`sudo cat /var/lib/jenkins/secrets/initialAdminPassword`

```
ubuntu@ip-172-31-37-211:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
df43c4644ce491298acefbac929bc5F
```

Now copy the password and paste it into the browser text box.

Then click on install suggested plug-ins.

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.



Also create a username and password.

Enter everything as admin. At least username as **admin** password as **admin**

Create User

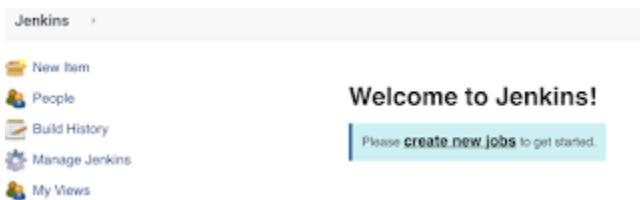
Username:	admin
Password:
Confirm password:
Full name:	admin
E-mail address:	admin@admin.com

Create User

Click on **Create User**.

Click on **Save and Finish**.

Click on **start using Jenkins**. Now you should see a screen like below:



Setup Jenkins slave. Then ensure it can run Docker builds.

Create a similar Ubuntu 22.4 LTS to be a slave.

Note: You need to create at least a t2.small Ubuntu 22.0.4 instance for this slave.
Only port 22 needs to be open

Configure it with the following:

Perform update first

sudo apt update

Install Java 11

sudo apt install default-jdk -y

```
ubuntu@ip-172-31-44-89:~$ sudo apt install default-jdk -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  at-spi2-core default-jdk-headless default-jre fonts-dejavu-extra libatk-bridge2.0-0
  libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0
  libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontenc1 libgif7 libgl1
  libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libice-dev libice6
  libllvm10 libpciauthentication0 libpthread-stubs0-dev libsensors4 libsm-dev libsm6 libx11-dev
  libx11-doc libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0
  libxcb-present0 libxcb-shape0 libxcb-sync1 libxcb1-dev libxcomposite1 libxdamage1
```

Once install java, enter the below command

Verify Java Version

java -version

```
ubuntu@ip-172-31-19-22:/opt$ java -version
openjdk version "11.0.6" 2020-01-14
OpenJDK Runtime Environment (Build 11.0.6+0-Ubuntu-1ubuntu118.04.1)
OpenJDK 64-Bit Server VM (Build 11.0.6+0-Ubuntu-1ubuntu118.04.1, mixed mode, sharing)
ubuntu@ip-172-31-19-22:/opt$
```

Maven Installation

You can install Maven by executing below command:

```
sudo apt install maven -y
```

you can type **mvn --version**

you should see the below output.

```
ubuntu@ip-172-31-34-93:~$ mvn -version
Apache Maven 3.6.0
Maven home: /usr/share/maven
Java version: 11.0.7, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-1065-aws", arch: "amd64", family: "unix"
ubuntu@ip-172-31-34-93:~$
```

Steps for installing Docker

```
sudo apt-get update && sudo apt install docker.io -y
```

Add Jenkins to Docker Group

```
sudo usermod -aG docker jenkins
```

```
sudo newgrp docker
```

```
sudo systemctl daemon-reload
```

Restart Docker service

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

```
sudo systemctl restart docker
```

```
sudo systemctl status docker
```

Test Permissions on Docker

```
docker run hello-world
```

permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:

Post "<http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/create?fromImage=tiangolo%2Fuwsgi-nginx-flask&tag=python3.6>": dial unix /var/run/docker.sock: connect: permission denied

Run the following:

```
sudo chmod 666 /var/run/docker.sock
```

Now run the following hello-world command again.

```
docker run hello-world
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
azureuser@VM12:~$
```

Slave node configuration

Change Host Name to Slave

```
sudo hostnamectl set-hostname Slave
```

Create User as Jenkins

```
sudo useradd -m jenkins
```

```
sudo -u jenkins mkdir /home/jenkins/.ssh
```

```
ubuntu@ip-172-31-11-223:~$ sudo useradd -m jenkins
ubuntu@ip-172-31-11-223:~$ sudo -u jenkins mkdir /home/jenkins/.ssh
ubuntu@ip-172-31-11-223:~$
```

Now Login to Jenkins Master

Create SSH keys by executing below command:

```
ssh-keygen -t rsa -m PEM
```

Please overwrite your existing keys.

```

azureuser@Control:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
02fbe195b2b0433681388effdc07abce
azureuser@Control:~$ ssh-keygen -t rsa -m PEM
Generating public/private rsa key pair.
Enter file in which to save the key (/home/azureuser/.ssh/id_rsa):
/home/azureuser/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/azureuser/.ssh/id_rsa
Your public key has been saved in /home/azureuser/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:/TbxQmoycKBn7Wkjzz1NkhG9MVRKkSVWTRee48GRmU azureuser@Control
The key's randomart image is:
+---[RSA 3072]---+
|       .oooB=.oE |
|     .. =o.o.. |
|    . . . . o. . |
|   .... .. . . |
|  .. .S.....o . |
| oo.= +. +... |
| o += = + ..o |
| .oo+o . . . |
| ++o.          |
+---[SHA256]---+
azureuser@Control:~$
```

Copy SSH Keys from Master to Slave

Execute the below command in Jenkins master EC2.

sudo cat ~/.ssh/id_rsa.pub

Copy the output of the above command:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADQABAAABgQDcKK7OR/+aMijfkUkqGXCOEqZMBY+069Dna38HIDtzAiKOYYqDuElcpbzOHSxjyrD6CPbaL6SC+Bx3zW2JIIWUVJZ3qsELjCzgLfKQLyvd9aWARzQcbcJGIOtMEx/kL2IUsb+sIFyDfI592nk0QpG5fYBC6Y8LrG9+FgQQrue8a7BC8aHvQuZ6o1c4k+aefc2CqQexonF9cDF4KMLnzYB+h3RRHUGMJ/ECd6OxmbprzQ4p1AeMcIV55OaJ7y34c/Pqq6lV2CF2ZpEd9dcROYW+hsNAZRHGEGdxmXIA5pmFuLh3DBVzUkVV0FL/+h5NUxfecY13BVTEfsQehCEdc1IE69oVUD3tsfxlt61ZrEdtkAcoJ9iLPY3WF8zLS1vMI7eeHDiYdSHSgFuZe2z/2D4peuY9/dKwc2nLqHSraD5vir6Xcz79YTx8+tA5r9aPpW9+hKV0FT9GiuzmyMjk8j/383wWcLqVTM2ZAkg44o+Y9bsK6Z4YJ1FY6qIY0= azureuser@Control
```

Now Login to Slave node and execute the below command

sudo -u jenkins vi /home/jenkins/.ssh/authorized_keys

This will be an empty file, now copy the public keys from **master** into above file.

Once you pasted the public keys in the above file in Slave, come out of the file by entering :wq!

Now go into master node

ssh jenkins@slave_node_ip

ssh jenkins@40.86.171.169

```
ubuntu@ip-172-31-44-147:~$ ssh jenkins@172.31.11.223
The authenticity of host '172.31.11.223 (172.31.11.223)' can't be established.
ECDSA key fingerprint is SHA256:z3tpstU79lwVTGNSbPgXfkScDvI6j8S1vJI3Zwi5o/k.
Are you sure you want to continue connecting (yes/no)? yes
```

```
jenkins@Slave: ~
The authenticity of host '40.86.171.169 (40.86.171.169)' can't be established.
ED25519 key fingerprint is SHA256:OVLBA39AeEpXaall1yDE3SWpKQsjHutofNoGaOKMf/Bw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '40.86.171.169' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1012-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Wed Sep 20 10:56:48 UTC 2023

System load:  0.0615234375      Processes:           116
Usage of /:   6.5% of 28.89GB   Users logged in:    1
Memory usage: 4%                  IPv4 address for eth0: 10.0.0.5
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

jenkins@Slave:~$
```

This is to make sure master is able to connect slave node. once you are successfully logged into slave, type **exit** to come out of slave.

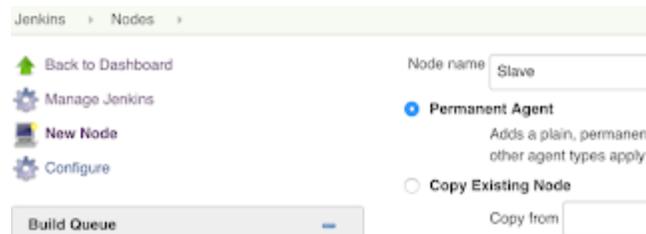
```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

$ exit
```

Register slave node in Jenkins:

Now go to Jenkins Master.

Manage Jenkins > Manage nodes.



Click on **New Node**.

Give name and check **permanent agent**.

Set no of executors as **1**.

Enter **/home/jenkins** as remote directory.

Select launch method as **Launch slaves nodes via SSH**.

Enter Slave node IP address as Host.

Click on **Manage Jenkins > credentials**.

Enter user name as **jenkins**.

Note: Make **jenkins** as lowercase as it is shown.

Kind as **SSH username with private key**.

Enter private key of master node directly by executing below command:

Add Credentials

Domain: Global credentials (unrestricted)

Kind: SSH Username with private key

Scope: Global (Jenkins, nodes, items, all child items, etc.)

Username: jenkins

Private Key: Enter directly

```
-----BEGIN RSA PRIVATE KEY-----
MIIEsgIBAAKCAQEAMWn0EggRBeU9qfutjZ1sMDAb0QcICWymyWNI9dL0kPGq2iR
qDv6OrEtKw3t4oTW8nhW1zxcC1Tu9tpQmwn0d4YsVX08GdJUWkq1+NZGoIm5+k
Pi5SojA3YEHgBAOY+OTUQg2EphsV26hEsPfsndzy9uPy64FMIEZEqwOsnp
DHdZMJA8M0xpjHAYG+8MGBIMViil68J5znViFEG7Gg9OsNPmmcozjNn+
-----
```

sudo cat ~/.ssh/id_rsa

(Make sure you copy the whole key including the below without missing anything)

-----BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----

```
azurereader@Control:~$ sudo cat ~/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIG5AIBAAKCAYEA3ciuzkfV2jIiX5FJKh1wtBKmTAwPt0vQ52t/B5Q7cwIitGGK
g7HCKYWh0sY8qw+gj4Wi+kgvgcd81tiSCF1FSWd6rBC4ws4C37SkC8r3fwlgEc
0Am3CRpTk5fH+c9iFLG/rCbgvBZefdp5NEKRuX2AQuPC6xvfhYEEK7nvK0wQv
Gh70FM+qNQXOPmn3NgqkHsaJxfXAxeCjC582Afod0UR1BjCfxAnejsZm6a80Kd
QHjHCFeeTmie8t+HPz6qupSL9ghdmaRHFXETmFvobDQGUrxhIHcZ15QoAzbh4d
wwC13FVdBS/v4eTVMX3nGndwVUx7EHoQhHNZNROvaFVA97bH8ZbetSGaxHbZAH
KCFYiz2N1hfMy0tbzJe3nhw4mNu0oBbmxts/9g+KXrmnPf3SsHnPy6h0q2g+b4q
+13M+/WE8FPrQ0a/Wj6VvfoS1dBu/RorkWzsjCZP/9/N8FnC6lUzNmQjIOOKPmP
W7CumeGCdRW0qpWNAGmBAAECggGASgP/aOh0ccXjG0AMuf0dmVyfvfUssCWLsRHJ
vjG0nwa7xir413n+N+DW6T/6s2svdHeKAJEnaJBHn+Dr/NVfSs0CTSZ4Fkq1ItQGg
dwE2rKM6yGmNVz+84nmQwZGbDuWJYth5gEy06DPeGPWP5nOz0aLGNOCLde34zUwu
5p5XX4AMqDs5Twp80PUAe5a2LngMrYV8Ucbtzr82UnBZvnawVq0t2p/i/Atra
etR9sn+nAQ7+0RsJxvs2BKRVs+D2BVdt0XZh5S1377u+PFQlwEZKGJ5U4m74wq
UNefkLtl+SShyggn6zSua11k8n0EdEBaQ/dEWox1zwL3H0wRmENaUN1aKisQ20Xp
bxHRWPNorwKRSKYtqCs0574Bs2tR0VCTdipp6NPJPSufeZIdKJ30zeGopbSu51
f+4iwMyb03PFWJniUkfMp0G+Y2fppA637yd14yPs+oX3+AIlmfTs5kLL461RsF
rdJ7a2HeSw0X276YH9Te10teaKdAoHBA02bsHe2ZfVFx8i3R/sLvwWix1/himSL
mbB7ouozBpig5anQNb1+0heCja3D+w3mFyH5oXM8R4pmXHYD8ZPrEQ/ENBwgjqQ
bpizs1tin6Mqt2/x274f82voxiXnseJRLCPw44KzljM2tYSneS+Hod8/HvKG+ZRG
9BcbaNlwRYUthPfgXS4j2za6W2HTsak6h03tp3uSmli2VY7qyWcPr/Uipy3I0mgj
b/y8VuBeEkhiYsj01BHQBeb+wx2Hy01T/owKbwDtMzqhowu7ofZt/teTna0+yL6e
QctnCyufhGvqiHx2tJ5DrZjqqPK0rLhwUeznc1hc9kVpZR0D77Z78/f1Hb15ow
0SjBjDAcLuUn+swnswQ4dv+tnd5C/30Tt0f0HYDkDPP8heaEIoDoeLygZ8PFrC7b+
XmNm6ku5Sz0sBjQ04c2CgqHz+qDm496AChvTmzPN8P7VAnET7sIAaSS5OOffHi0
/T3BuKwGD+vA6NdKbpj+2le7D+1TOdyk01svqQ8CgcEAi5V+oZUBaI1JyFwDaTOJ
UG1EBcuVZW0tMcQv/K11HpzGzm+GyN/yI-VNBtUvecRHsqo/H0C6FSITBpIJqohH
r/VufIRQ3KS1k/ddUj4p0cy1ReorURGsCqR/cowo+/HooWM4edDAfq2zhskUgt6
r++A2rD4Rw5VanAepw9+vJ5pjTpjtj2E66CEzuWjuhxlwbrk0iay6qxshelQvox
4DBhV/A+95Z531t1HjGkT4cdAF4gs6Z90Z+msfts8dJaoHAYCpm20VLens9ffbV
26kPdebZMt+3TVhgBKjMz+uzKkUr7PBSkLU4KPkvi1as9StBw9s15ekpk4tCqsP
q2so9jseG10dQSQJTr1+PF7F5VCXsoP+omIdM00hmunt+eM5b0zCpVPljoXhNe
VLYUinujYxinUAalUKgpCkr/U31AKXdcCS2eLF/X1nR98DH00IUhVaHKm+ZfQ22
3I//Qx1/mpEZgpo3ArYttHxy4qFYMeq/q1YUgb6XvMzkwcNAoHBAKvhqwCst+bM
TZ2pMbBy2swGxoJVOAReyJ7HTWRhWvc3y8dQh0FhXMOmePmUSGwhYJD+VSbpqId
gn+cSsNdAqAIRCz3jfdwjVd1xJd2u3GZNVzViHzd7P+oPNINVtvvPn1taaLuwGE
Xn0+qgyZP4mBmMoP/LMQODizzxbpAJinzpvVJgWjCG8VVGEDwJ3c6D0/LA9cA1wG
ipy+WuAKGRGIep8DH1Rtt28h9cWRtxS1EkppR3YRI2CD/eHe9wT14Q==
```

-----END RSA PRIVATE KEY-----

azurereader@Control:~\$

Click **Save**.

Select **Host key verification strategy** as "manually trusted key verification strategy".

Launch agents via SSH

Host ?
40.86.171.169

Credentials ?
jenkins

Add ▾

Host Key Verification Strategy ?
Manually trusted key Verification Strategy

| Require manual verification of initial connection ?

Advanced ▾ Edited

Availability ?
Keep this agent online as much as possible

Save

Click **Save**.

Click on launch agent..make sure it connects to agent node.

```
INFO: Both error and output logs will be printed to /home/jenkins/remoting
<===[JENKINS REMOTING CAPACITY]==>channel started
Remoting version: 3131.vf2b_b_798b_ce99
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by jenkins.slaves.StandardOutputSwapper$ChannelSwapper to constructor java.io.FileDescriptor(int)
WARNING: Please consider reporting this to the maintainers of jenkins.slaves.StandardOutputSwapper$Chan
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Evacuated stdout
Agent successfully connected and online
```

S	Name ↓	Architecture	C
	master	Linux (amd64)	
	Slave	Linux (amd64)	

The screenshot shows the Jenkins 'Nodes' page. On the left, there are navigation links for 'Nodes', 'Clouds', 'Node Monitoring', 'Build Queue' (with a note: '1 builds in the queue'), and 'Build Executor Status'. The main area is titled 'Nodes' and contains a table with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, Response Time, and a status icon. Two nodes are listed: 'Built-In Node' (Linux amd64) and 'Slave' (Linux amd64). Both nodes show 'In sync' under Clock Difference, and their Free Disk Space is 24.75 GB and 26.99 GB respectively. The 'Slave' node has a warning icon indicating 0 B free swap space.

Now you can kick start building the jobs, you will see Jenkins master runs jobs in slave nodes.

3. Docker, Docker pipeline and Kubectl CLI plug-ins are installed in Jenkins.

Managing Jenkins > Plugins

Available Plugins

Select and install the following:

Docker
Docker Pipeline
Kubectl CLI

The screenshot shows the Jenkins 'Plugin Manager' page. A search bar at the top has 'kubernetes' typed into it. Below the search bar, there is a button labeled 'Install'. Under the search results, there are two checked checkboxes: 'Docker' and 'Docker Pipeline'. The 'Docker' plugin is described as integrating Jenkins with Docker, and the 'Docker Pipeline' plugin is described as building and using Docker containers from pipelines.

Plugin	Description
Docker	This plugin integrates Jenkins with Docker
Docker Pipeline	Build and use Docker containers from pipelines.

The screenshot shows the Jenkins 'Plugin Manager' page with 'kubernetes' searched. The results list several 'kubernetes' related plugins, each with a checkbox, a link, and a timestamp. All checkboxes are checked.

Plugin	Description	Last Updated
Kubernetes	build and use Docker containers from pipelines.	21 days ago
Kubernetes Credentials	This plugin integrates Jenkins with Kubernetes	21 days ago
Kubernetes Client API	Kubernetes Client API plugin for use by other Jenkins plugins.	21 days ago
Kubernetes CLI	Configure kubectl for Kubernetes	21 days ago
Kubernetes Credentials Provider	Provides a read only credentials store backed by Kubernetes.	15 days ago
Kubernetes :: Pipeline :: DevOps Steps	pipeline kubernetes	4 yr 7 mo ago

→ [Go back to the top page](#)

(you can start using the installed plugins right away)

→ Restart Jenkins when installation is complete and no jobs are running

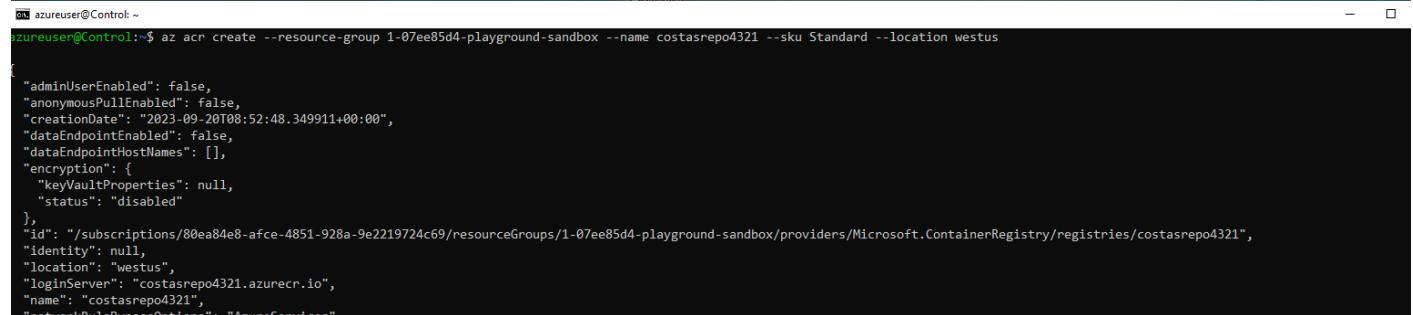
4. Azure container registry is setup in <https://portal.azure.com>

Run the below command to create your own private container registry using Azure Container Registry (ACR).

Make sure below green marked registry name is unique.

```
az acr create --resource-group myResourceGroup --name myacrrepo4321 --sku Standard --location southcentralus
```

```
az acr create --resource-group 1-07ee85d4-playground-sandbox --name costasrepo4321 --sku Standard --location westus
```



```
azureuser@Control ~
azureuser@Control:~$ az acr create --resource-group 1-07ee85d4-playground-sandbox --name costasrepo4321 --sku Standard --location westus

{
  "adminUserEnabled": false,
  "anonymousPullEnabled": false,
  "creationDate": "2023-09-20T08:52:48.349911+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
    "status": "disabled"
  },
  "id": "/subscriptions/80ea84e8-acfe-4851-928a-9e2219724c69/resourceGroups/1-07ee85d4-playground-sandbox/providers/Microsoft.ContainerRegistry/registries/costasrepo4321",
  "identity": null,
  "location": "westus",
  "loginServer": "costasrepo4321.azurecr.io",
  "name": "costasrepo4321",
  "networkRuleResourceGroup": "AzureServices"
}
```

Connect to the cluster

```
az aks get-credentials --resource-group myResourceGroup --name myAKSCluster --overwrite-existing
```

```
az aks get-credentials --resource-group 1-07ee85d4-playground-sandbox --name myAKSCluster --overwrite-existing
```



```
devopscoaching@DevOps-Coach-MacBook-Pro:~/Downloads$ az aks get-credentials --resource-group myResourceGroup --name myAKSCluster --overwrite-existing
Merged "myAKSCluster" as current context in /Users/devopscoaching/.kube/config
```

To verify the connection to your cluster, use the kubectl get command to return a list of the cluster nodes.

kubectl get nodes

```
AK-DevOps-01:~ devopscoaching$ kubectl get nodes
NAME           STATUS  ROLES   AGE    VERSION
aks-nodepool1-26195414-0  Ready   agent   13m    v1.17.11
aks-nodepool1-26195414-1  Ready   agent   13m    v1.17.11
```

```
# List all deployments in a specific namespace  
kubectl get deployments --all-namespaces=true
```

AK-DevOps-81:- devopscoaching\$ kubectl get deployments --all-namespaces=true					
NAMESPACE	NAME	READY	UP-TO-DATE	AVAILABLE	AGE
kube-system	coredns	2/2	2	2	6m25s
kube-system	coredns-autoscaler	1/1	1	1	6m25s
kube-system	metrics-server	1/1	1	1	6m25s
kube-system	omsagent-rs	1/1	1	1	6m25s
kube-system	tunnelfront	1/1	1	1	6m24s

```
az azureuser@Control:~  
azureuser@Control:~$ az aks get-credentials --resource-group 1-07ee85d4-playground-sandbox --name myAKSCluster --overwrite-existing  
Merged "myAKSCluster" as current context in /home/azureuser/.kube/config  
azureuser@Control:~$ kubectl get nodes  
NAME STATUS ROLES AGE VERSION  
aks-nodepool1-35897272-vmss00000 Ready agent 15m v1.26.6  
aks-nodepool1-35897272-vmss00001 Ready agent 15m v1.26.6  
azureuser@Control:~$  
azureuser@Control:~$ kubectl get deployments --all-namespaces=true  
NAMESPACE NAME READY UP-TO-DATE AVAILABLE AGE  
gatekeeper-system gatekeeper-audit 1/1 1 1 7m25s  
gatekeeper-system gatekeeper-controller 2/2 2 2 7m25s  
kube-system azure-policy 1/1 1 1 7m26s  
kube-system azure-policy-webhook 1/1 1 1 7m25s  
kube-system coredns 2/2 2 2 16m  
kube-system coredns-autoscaler 1/1 1 1 16m  
kube-system konnectivity-agent 2/2 2 2 16m  
kube-system metrics-server 2/2 2 2 16m  
azureuser@Control:~$  
azureuser@Control:~$
```

```
ubuntu@ip-172-31-2-128:~$ sudo cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZBQFURVFRSLN9OKL7UNSRERDQ0QDFJSUWZB1CQ...
  servers:
  - server: https://172.31.2.128:6443
  users:
  - user:
    client-certificate-data: MIGfMA0GCSqGSIb3DQEBAQUAAQCH...
    client-key-data: MIIEvQIBAAKCAQ...
    name: default-client
```

Enter ID as K8S and choose enter directly and paste the above file content and save

Global credentials (unrestricted)

Kind	Kubernetes configuration (kubeconfig)
Scope	Global (Jenkins, nodes, items, all child items, etc)
ID	K8S
Description	kube configuration
Kubeconfig	<input checked="" type="radio"/> Enter directly Content: <pre>apiVersion: v1 clusters: - cluster: certificate-authority-data: LSQLS1CRUDUTIBDRVJUSUZ/JQFURS0tL80tCk1JSUN5RENDQWJD</pre>
	<input type="radio"/> From a file on the Jenkins master <input type="radio"/> From a file on the Kubernetes master node

OK

Step # 3 - Create a pipeline in Jenkins

Create a new pipeline job.

Enter an item name

myKubernetsDeployJob
» Required field

Freestyle project

This is the central feature of Jenkins for building something other than software builds.

Pipeline

Orchestrates long-running activities by organizing complex activities that

Step # 4 - Copy the pipeline code from below

Make sure you change red highlighted values below:

Your docker user id should be updated.

your registry credentials ID from Jenkins from step # 1 should be copied

```
pipeline {
    agent {
        label 'mystave'
    }
    environment {
        //once you sign up for Docker hub, use that user_id here
        registry = "your_docker_hub_user_id/mypython-app"
        // update your credentials ID after creating credentials for connecting to Docker Hub
        registryCredential = 'dockerhub'
```

```

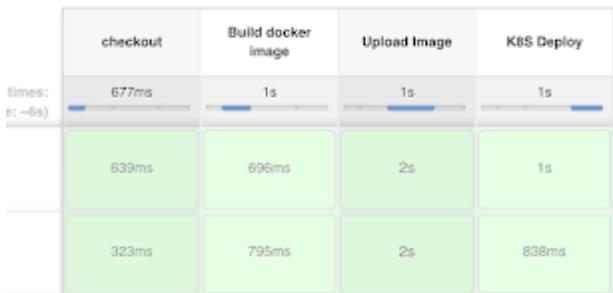
        dockerImage = "
    }
    stages {
        stage ('checkout') {
            steps {
                checkout([$class: 'GitSCM', branches: [[name: '/master']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[url: 'https://github.com/akannan1087/myPythonDockerRepo']]])
            }
        }
        stage ('Build docker image') {
            steps {
                script {
                    dockerImage = docker.build registry
                }
            }
        }
    }
    // Uploading Docker images into Docker Hub
    stage('Upload Image') {
        steps{
            script {
                docker.withRegistry( "", registryCredential ) {
                    dockerImage.push()
                }
            }
        }
    }
}

stage ('K8S Deploy') {
    steps {
        script {
            kubernetesDeploy(
                configs: 'k8s-deployment.yaml',
                kubeconfigId: 'K8S',
                enableConfigSubstitution: true
            )
        }
    }
}
}

```

Step # 5 - Build the pipeline

Once you create the pipeline and changes values per your Docker user id and credentials ID, click on



Step # 6 - Verify deployments to K8S

`kubectl get pods`

```
ubuntu@ip-172-31-2-128:~$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
ok1-controller-h8dck   1/1     Running   3          23h
my-php-deployment-8cd555cd6-b25bv   1/1     Running   0          4m2s
my-php-deployment-8cd555cd6-wbh1k   1/1     Running   0          4m2s
```

`kubectl get deployments`

```
ubuntu@ip-172-31-2-128:~$ kubectl get deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
my-php-deployment   2/2     2           2           7m27s
ubuntu@ip-172-31-2-128:~$
```

`kubectl get services`

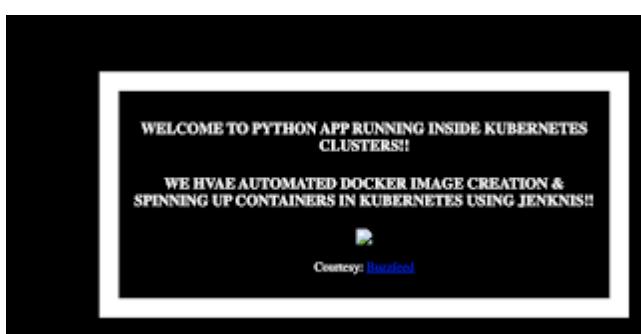
```
ubuntu@ip-172-31-2-128:~$ kubectl get services
NAME            TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
ok-angular-app   NodePort   10.103.50.84 <none>       8885:32004/TCP   23h
kubernetes       ClusterIP  10.96.0.1    <none>       443/TCP       26h
php-app-svc     LoadBalancer 10.97.39.217 <pending>    5000:30814/TCP   3m31s
ubuntu@ip-172-31-2-128:~$
```

Steps # 7 - Access Python App in K8S cluster

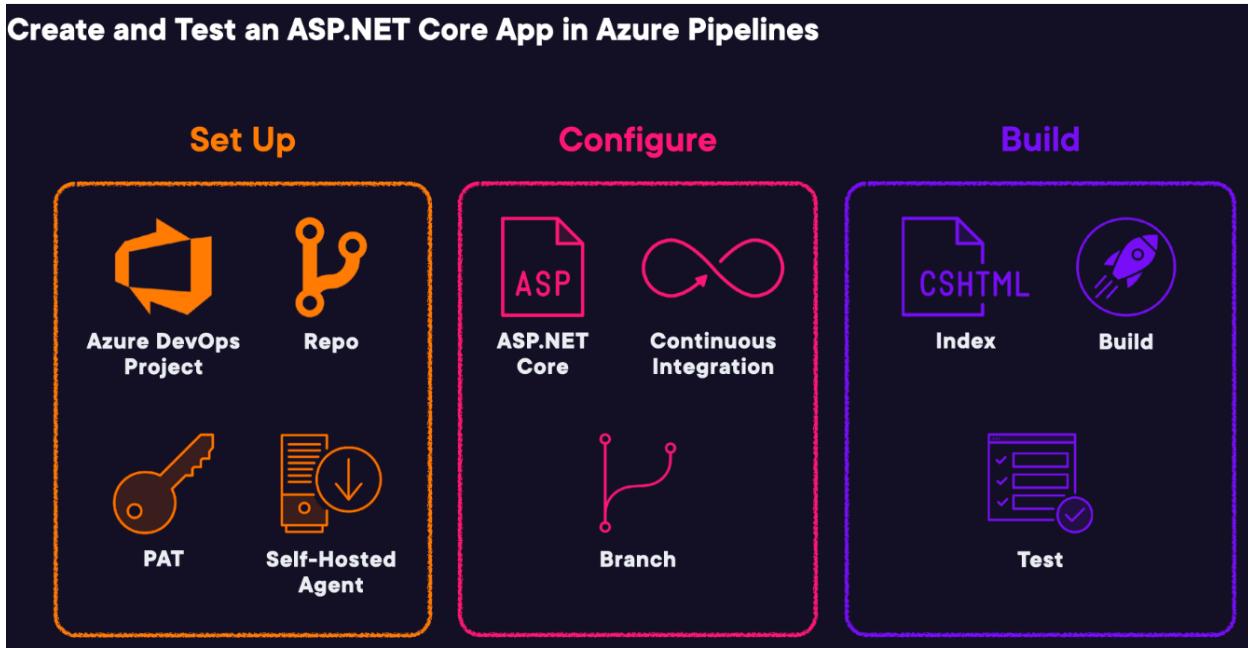
Once build is successful, go to browser and enter master or worker node public ip address along with port number mentioned above

`http://master_or_worker_node_public_ipaddress:port_no_from_above`

You should see page like below:



Create and Test an ASP.NET Core App in Azure Pipelines



Create the VM (and other resources using ARM)

The screenshot shows the Azure portal interface for a resource group named "1-b2909d3f-playground-sandbox". The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Resource visualizer, Events, Deployments, Security, Deployment stacks, Policies, Properties, and Locks. The main pane displays the "Essentials" section with details about the subscription (move), ID, and location (West US). Below this is the "Resources" section, which is currently empty, showing 0 records. Filter options and sorting tools are available at the bottom of the resources table.

Go to the **Market Place** and locate an Ubuntu image

Microsoft Azure Search resources, services, and docs (G+)

Home > Marketplace > Create a virtual machine

Region * (US) West US

Availability options No infrastructure redundancy required

Security type Trusted launch virtual machines [Configure security features](#)

Image * Ubuntu Server 20.04 LTS - x64 Gen2 [See all images](#) | [Configure VM generation](#)

VM architecture x64

Run with Azure Spot discount

Size * Standard_D2s_v3 - 2 vcpus, 8 GiB memory (US\$85.41/month) [See all sizes](#)

Administrator account

Authentication type SSH public key

Note: Item(s) availability based on policy assignment(s) for the selected scope.
vadergen2labs/Microsoft.Authorization/8ab9bdb77b304947a1347e88 ([Policy details](#))

Username * azureuser

SSH public key source Generate new key pair

Key pair name * linVM_key

Ensure that the **Region** is the same as the Resource group.

NOTE: Ubuntu 20.04 LTS was the only option.

Choose the **Size** Standard D2 with 2 CPUs as shown above.

For **Authentication** type, choose **SSH Public Key** and download the key pair locally to your system. Go to end and create the ubuntu virtual machine.

NOTE: When you complete creating the VM all the other associated resources will also be created.

Navigate to the Virtual machine **linVM** within the resource group and select the link.
On the left hand menu, navigate to Settings and select **connect > Native SSH**

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation menu is open, showing various options under 'Settings' such as Networking, Connect, Disks, Size, Microsoft Defender for Cloud, Advisor recommendations, Extensions + applications, Availability + scaling, Configuration, Identity, Properties, Locks, and Operations. The 'Connect' option is selected. In the center, there is a 'Native SSH' dialog box. At the top of the dialog, it says 'Connecting using public IP address 20.253.229.27'. Below this, there are two sections: 'Recommended' and 'Most common'. Under 'Recommended', there is an option 'SSH using Azure CLI' with a 'Select' button. Under 'Most common', there is an option 'Native SSH' with a 'Select' button. At the bottom of the dialog, there is a note: 'More ways to connect (3)'. To the right of the dialog, there is a sidebar titled 'Native SSH' with the sub-section 'Configure prerequisites for Native SSH'. It lists three steps: 1. Ready for configuration (with a note about Port 22 access), 2. Open a local shell (on Windows), and 3. Copy and execute SSH command. The 'Copy and execute SSH command' section contains a command line: `ssh -i ~/ssh/id_rsa.pem azureuser@20.253.229.27`.

In a command prompt on your system type the following:
ssh -i <path to key pair>/<key pair name>.pem <username created earlier>@<public IP address of VM>

Once logged in, proceed to create the Pipeline within Azure DEVOPS.

Create the Azure DEVOPS Pipeline

In the main search bar, search for and select **Azure DevOps organizations**.

Scroll down and click **My Azure DevOps Organizations**.

Select your country/region from the dropdown list.

Click **Continue**.

Select **Create new organization > Continue > Continue** (complete the required CAPTCHA).

Set the following values to create a project:

Project name: **ASP_.NET**

Visibility: **Private**

Click + **Create project**.

Obtain and Push Project Code

In the Linux VM, pull code from the unit-test branch using the following URL:

git clone -b unit-test <https://github.com/ACloudGuru-Resources/content-az400-lab-resources.git>

```
azures and docs (G+) 
az user@linVM: ~
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

azureuser@linVM:~$ git clone -b unit-test https://github.com/ACloudGuru-Resources/content-az400-lab-resources
Cloning into 'content-az400-lab-resources'...
remote: Enumerating objects: 4353, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (104/104), done.
remote: Total 4353 (delta 29), reused 17 (delta 6), pack-reused 4242
Receiving objects: 100% (4353/4353), 356.10 MiB | 29.57 MiB/s, done.
Resolving deltas: 100% (1508/1508), done.
azureuser@linVM:~$
```

List the contents of the directory with **ls**

Observe the folder is cloned to the VM.

Change into the directory shown:

cd content-az400-lab-resources/

View the remote origin:

git remote -v

It is currently the ACG GitHub repo.

Remove the remote origin:

git remote remove origin

```
az user@linVM:~/content-az400-lab-resources
az user@linVM:~/content-az400-lab-resources$ git clone -b unit-test https://github.com/ACloudGuru-Resources/content-az400-lab-resources.git
Cloning into 'content-az400-lab-resources'...
remote: Enumerating objects: 4353, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (104/104), done.
remote: Total 4353 (delta 29), reused 17 (delta 6), pack-reused 4242
Receiving objects: 100% (4353/4353), 356.10 MiB | 29.57 MiB/s, done.
Resolving deltas: 100% (1508/1508), done.
az user@linVM:~/content-az400-lab-resources$ ls
content-az400-lab-resources
az user@linVM:~/content-az400-lab-resources$ cd content-az400-lab-resources/
az user@linVM:~/content-az400-lab-resources$ git remote -v
origin https://github.com/ACloudGuru-Resources/content-az400-lab-resources.git (fetch)
origin https://github.com/ACloudGuru-Resources/content-az400-lab-resources.git (push)
az user@linVM:~/content-az400-lab-resources$ git remote remove origin
az user@linVM:~/content-az400-lab-resources$ git remote -v
az user@linVM:~/content-az400-lab-resources$ git remote add origin https://clouduserp4b842ee7@dev.azure.com/clouduserp4b842ee7/ASP_.NET/_git/ASP_.NET
az user@linVM:~/content-az400-lab-resources$ git push -u origin --all
Password for 'https://clouduserp4b842ee7@dev.azure.com':
```

If you use the **git remote -v** command again you will find that nothing comes up!
Go back to **Azure DevOps**. From the left menu, click **Repos**.
Under **Push an existing repository** from command line, copy the commands.

The screenshot shows the Azure DevOps interface. On the left, there's a sidebar with icons for Repos, Files, Commits, Pushes, Branches, Tags, Pull requests, Pipelines, and Test Plans. The 'Repos' icon is highlighted. The main area has tabs for 'HTTPS' and 'SSH'. The URL in the address bar is https://clouduserp9d19bec0@dev.azure.com/clouduserp9d19bec0/MyFirstProject/_git/MyFirstProject. Below the tabs, there's a 'Generate Git Credentials' button and a note about authentication. A large text box contains the command: 'git remote add origin https://clouduserp9d19bec0@dev.azure.com/clouduserp9d19bec0/MyFirstProject/_git/MyFirstProject'. At the bottom, a terminal window shows the command being run on a Linux VM: 'git remote -v', 'git remote add origin https://clouduserp9d19bec0@dev.azure.com/clouduserp9d19bec0/MyFirstProject/_git/MyFirstProject', 'git push -u origin -all', and a password prompt for the URL.

```
cloud_user@linVM:~/content-az400-lab-resources$ git remote -v
cloud_user@linVM:~/content-az400-lab-resources$ git remote add origin https://clouduserp9d19bec0@dev.azure.com/clouduserp9d19bec0/MyFirstProject/_git/MyFirstProject
cloud_user@linVM:~/content-az400-lab-resources$ git push -u origin -all
Password for 'https://clouduserp9d19bec0@dev.azure.com':
```

Return to the Linux VM and paste in the commands. You will be asked for a personal access token.

Create Personal Access Token

Back in Azure DevOps, in the upper right-hand corner, click the **User settings icon > Personal access tokens**.

Click **+ New Token** and set the following values:

Name: **linuxvm**

Scopes: **Full access**

Click **Create**.

Create a new personal access token

Name: linuxvm

Organization: clouduserp9d19bec0

Expiration (UTC): 30 days, 10/9/2023

Scopes: Full access (selected)

It will create a string like below:

das5yco2kl474qngkvp3fikjev6vtgxgymaverwa7kasm5nq3xfkq

Copy the **linuxvm** token and save it in a safe location.

Click **Close**.

```
azureuser@linVM: ~/content-az400-lab-resources
emote: Compressing objects: 100% (104/104), done.
emote: Total 4353 (delta 29), reused 17 (delta 6), pack-reused 4242
receiving objects: 100% (4353/4353), 356.10 MiB | 29.57 MiB/s, done.
resolving deltas: 100% (1508/1508), done.
azureuser@linVM:~$ ls
content-az400-lab-resources
azureuser@linVM:~/content-az400-lab-resources$ cd content-az400-lab-resources/
azureuser@linVM:~/content-az400-lab-resources$ git remote -v
origin https://github.com/ACloudGuru-Resources/content-az400-lab-resources.git (fetch)
origin https://github.com/ACloudGuru-Resources/content-az400-lab-resources.git (push)
azureuser@linVM:~/content-az400-lab-resources$ git remote remove origin
azureuser@linVM:~/content-az400-lab-resources$ git remote -v
azureuser@linVM:~/content-az400-lab-resources$ git remote add origin https://clouduserp4b842ee7@dev.azure.com/clouduserp4b842ee7/ASP_.NET/_git/ASP_.NET
azureuser@linVM:~/content-az400-lab-resources$ git push -u origin --all
assword for 'https://clouduserp4b842ee7@dev.azure.com':
numerating objects: 184, done.
ounting objects: 100% (184/184), done.
elta compression using up to 2 threads
ompressing objects: 100% (146/146), done.
riting objects: 100% (184/184), 1.20 MiB | 7.96 MiB/s, done.
otal 184 (delta 39), reused 150 (delta 25)
emote: Analyzing objects... (184/184) (212 ms)
emote: Validating commits... (8/8) done (1 ms)
emote: Storing packfile... done (97 ms)
emote: Storing index... done (44 ms)
o https://dev.azure.com/clouduserp4b842ee7/ASP_.NET/_git/ASP_.NET
* [new branch]    unit-test -> unit-test
ranch 'unit-test' set up to track remote branch 'unit-test' from 'origin'.
azureuser@linVM:~/content-az400-lab-resources$
```

Return to the Linux VM and paste in the access token as above.

If you now use **git remote -v** you will see the git hub code again.

In Azure DevOps, click the **Azure DevOps** icon in the upper-left corner.

Select your project.

From the left-hand navigation menu, click **Repos**. Verify that the code was pushed to Azure Repos.

The screenshot shows the 'Repos' section of the Azure DevOps interface. On the left, there's a sidebar with icons for Repos, Files, Commits, Pushes, Branches, Tags, and Pull requests. The 'Repos' icon is highlighted. The main area displays a table of files with columns for Name, Last change, and Commits. The table includes entries for 'AspNetCoreWebApplication', 'AspNetCoreWebApplicationTest', '.gitignore', 'AspNetCoreWebApplication.sln', 'azure-pipelines.yml', and 'README.md'. Each entry shows the date of the last change and a link to the commit details.

Name ↑	Last change	Commits
AspNetCoreWebApplication	Apr 11, 2022	ad3a4cd8 Updated package references Peter Ksiazek
AspNetCoreWebApplicationTest	Apr 11, 2022	ad3a4cd8 Updated package references Peter Ksiazek
.gitignore	Oct 9, 2020	39c1b922 added .gitignore chadmcrowell
AspNetCoreWebApplication.sln	Oct 9, 2020	92a3ef16 unit test added Chad Crowell
azure-pipelines.yml	Oct 9, 2020	92a3ef16 unit test added Chad Crowell
README.md	May 31	a540a49d Update README.md alexanderpotasnick

Create a New Azure DevOps Pipeline

At the top of the page, click the **Azure DevOps** icon, then select the project.

On the bottom left, click **Project settings**.

Scroll down the left-hand pane and under **Pipelines**, select **Agent pools**.

Select the **Default** pool.

The screenshot shows the 'Project Settings' page for the 'MyFirstProject'. The left sidebar has sections for General, Overview, Teams, Permissions, Notifications, Service hooks, Dashboards, Boards, Project configuration, Team configuration, GitHub connections, Pipelines, and Agent pools. The 'Agent pools' section is currently selected. The main area is titled 'Agent pools' and shows a list of existing agent pools. One pool, 'Azure Pipelines', is listed under 'Name'. Another pool, 'Default', is also listed and is highlighted with a yellow background, indicating it is the selected pool.

Get the agent

X

Windows

macOS

Linux

x64

ARM

ARM64

RHEL6

System prerequisites

Configure your account

Configure your account by following the steps outlined [here](#).

Download the agent

[Download](#)



Create the agent

```
~/$ mkdir myagent && cd myagent  
~/myagent$ tar zxvf ~/Downloads/vsts-agent-linux-x64-3.225.0.tar.gz
```

Configure the agent [Detailed instructions ↗](#)

```
~/myagent$ ./config.sh
```

Optionally run the agent interactively

If you didn't run as a service above:

```
~/myagent$ ./run.sh
```

Click **New agent**.

Select the **Linux** tab.

Click the Copy icon, next to Download to copy the agent URL to your clipboard.

Paste it to a text file to use in the following steps.

<https://vstsagentpackage.azureedge.net/agent/3.225.0/vsts-agent-linux-x64-3.225.0.tar.gz>

Keep the **Linux** tab and popup open.

Go back to the Linux VM, and change directory to the root: **cd ~**

Make and navigate to a new directory called **Downloads**:

mkdir Downloads && cd Downloads

Run **wget** against the agent URL you just copied:

wget <AGENT_URL>

wget https://vstsagentpackage.azureedge.net/agent/3.225.0/vsts-agent-linux-x64-3.225.0.tar.gz

This will download the agent.

Navigate back to the root again by using **cd ~**.

Create and navigate to another directory called myagent:

mkdir myagent && cd myagent

Now copy the file downloaded from **downloads** to the **myagent** folder

example:

```
cp /home/cloud_user/Downloads/vsts-agent-linux-x64-3.225.0.tar.gz /home/cloud_user/myagent
```

NOTE: To work out the paths to use above use the command **pwd**.

Unzip the package you just downloaded. You can copy this same command from the popup in Azure DevOps to get the full package and version number:

```
tar zxvf <PACKAGE>
```

example:

```
tar zxvf vsts-agent-linux-x64-3.225.0.tar.gz
```

```
cloud_user@linVM:~/myagent$ ls
bin config.sh env.sh externals license.html run-docker.sh run.sh vsts-agent-linux-x64-3.225.0.tar.gz
cloud_user@linVM:~/myagent$
```

Review the contents of the directory (**ls**). Observe all of the files and folders for the agent.

Configure the Agent

Run the configuration script with the **config.sh** file:

```
azureuser@linVM: ~/myagent
azureuser@linVM:~/myagent$ azureuser@linVM:~/myagent$ ./config.sh

agent v3.225.0
          (commit ef0cd4d)

>> End User License Agreements:

Building sources from a TFVC repository requires accepting the Team Explorer Everywhere End User License Agreement. This step is not required for building sources from Git repositories.

A copy of the Team Explorer Everywhere license agreement can be found at:
  /home/azureuser/myagent/license.html

Enter (Y/N) Accept the Team Explorer Everywhere license agreement now? (press enter for N) > y

>> Connect:

Enter server URL > https://dev.azure.com/clouduserp4b842ee7
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) >
Enter agent name (press enter for linVM) >
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2023-09-09 18:35:05Z: Settings Saved.
azureuser@linVM:~/myagent$
```

./config.sh

Type **y** to accept the license agreement.

When prompted for your server URL, go back to Azure DevOps and grab the URL from your browser. It will look something like <https://dev.azure.com/clouduser...>, followed by some numbers (i.e., copy everything up to and not including /ASP_.NET).

Paste this in the terminal.

Example:

<https://dev.azure.com/clouduserp4b842ee7>

NOTE: even adding / after the above URL will cause a failure and you will have start again!

Press **Enter** to accept the default authentication type of **PAT**.

For the personal access token option, copy and paste the personal access token you created and saved earlier in this lab.

example:

das5yco2kl474qngkvp3fikjev6vtgxgymaverwa7kasm5nq3xfkq

NOTE: this sandbox environment was deleted after creating this lab. Hence the above PAT is useless.

Press **Enter** to accept the default agent pool.

Press **Enter** to accept the default name.

Press **Enter** to accept the default work folder.

Run the agent interactively so that it is always listening for any jobs by using the following bash script.

./run.sh

```
azureuser@linVM:~/myagent$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2023-09-09 18:38:03Z: Listening for Jobs
```

Create the New Pipeline

Go back to Azure DevOps, and in the left-hand navigation menu, click **Pipelines**.

Click **Create Pipeline**.

The screenshot shows the Azure DevOps Pipelines interface. On the left, there's a sidebar with a tree view of project components: Overview, Boards, Repos, Pipelines (selected), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area has tabs at the top: Connect (selected), Select, Configure, and Review. Below the tabs, it says "New pipeline" and features a large heading "Where is your code?". A list of source options is shown:

- Azure Repos Git (YAML) - Free private Git repositories, pull requests, and code search
- Bitbucket Cloud (YAML) - Hosted by Atlassian
- GitHub (YAML) - Home to the world's largest community of developers
- GitHub Enterprise Server (YAML) - The self-hosted version of GitHub Enterprise
- Other Git - Any generic Git repository
- Subversion - Centralized version control by Apache

At the bottom, a note says "Use the classic editor to create a pipeline without YAML."

Click **Use the classic editor**.

Under **Select a source**, select **Azure Repos Git** and make the following selections:

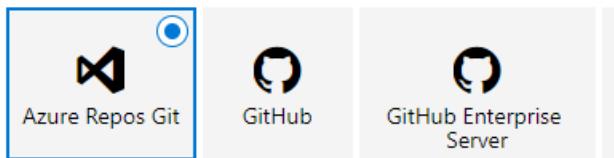
Team project: **ASP_.NET**

Repository: **ASP_.NET**

Default branch: **unit-test**

Click **Continue**.

Select a source



Team project

 ▼

Repository

 ▼

Default branch for manual and scheduled builds

 ▼

Continue

Scroll down and select the **ASP.NET Core** template.

Click **Apply**.

On the right panel, for **Agent pool**, select **Default**.

For Project(s) to test, enter: ****/*[Tt]est/*.csproj**.

NOTE: Please avoid typos as this will cause the job to fail!

Next to Agent job 1, click the + icon to add a new task.

Pipeline

Build pipeline

Get sources

ASP_.NET unit-test

Agent job 1

Run on agent

+

Restore .NET Core

Build .NET Core

Test .NET Core

Publish .NET Core

Publish Artifact

Publish build artifacts

Name *

ASP_.NET-ASP.NET Core-CI

Agent pool | Pool information | Manage

Default

Parameters | Unlink all

Project(s) to restore and build

**/*.csproj

Project(s) to test

**/*[Tt]est/*.*proj

Search for and click Add for the **Use .NET Core** task.

Add tasks | Refresh

Use .NET Core

dotnet

Use .NET Core

Acquires a specific version of the .NET Core SDK from the internet or the local cache and adds it to the PATH. Use this task to change the version of .NET Core used in subsequent tasks. Additionally provides proxy support.

NOTE: Please ensure you use this legacy version.

In the task list for Agent job 1, drag and move the **Use .NET Core task** to the top.

With the **Use .NET Core** task still selected, change the Version on the right to **3.1.x**.

At the top, click **Save & queue > Save > Save**.

The screenshot shows the Azure DevOps Pipeline Editor. The pipeline consists of a 'Get sources' task for 'ASP.NET' and an 'Agent job 1' which runs on an agent. Inside 'Agent job 1', there are several tasks: 'Use .NET Core sdk 3.1.x', 'Restore', 'Build', 'Test', 'Publish', and 'Publish Artifact'. The 'Use .NET Core' task is currently selected and expanded, showing its configuration. The 'Task version' is set to '2.*'. The 'Display name' is 'Use .NET Core sdk 3.1.x'. Under 'Package to install', 'SDK (contains runtime)' is selected. The 'Version' is set to '3.1.x'. In the 'Advanced' section, there are fields for 'Compatible Visual Studio version' (empty) and 'Path To Install .Net Core' (set to '\$(Agent.ToolsDirectory)/dotnet'). There is also an unchecked checkbox for 'Perform Multi Level Lookup'.

Select the **Triggers** tab and check the box for **Enable continuous integration** on the right panel.
At the top, click **Save & queue** > **Save** > **Save**.

The screenshot shows the Azure DevOps Pipeline Editor with the 'Triggers' tab selected. Under 'Continuous integration', the 'ASP.NET' trigger is listed with the status 'Enabled'. On the right side, there are configuration options for this trigger. The 'Enable continuous integration' checkbox is checked. There is also an unchecked checkbox for 'Batch changes while a build is in progress'. Below these are sections for 'Branch filters' and 'Path filters'. The 'Branch filters' section has a 'Type' dropdown set to 'Include' and a 'Branch specification' field containing 'unit-test'. The 'Path filters' section has a '+ Add' button.

Update the Branch

On the left menu, select **Repos**.

Select the **AspNetCoreWebApplication** folder on the right.

Navigate through the folders to **Views > Home**.

Open the **Index.cshtml** file.

To the right of the code pane, click **Edit**.

Make a minor change in the **<h1> Congratulations! After Congratulations! </h1>**, add **You Did It!**.

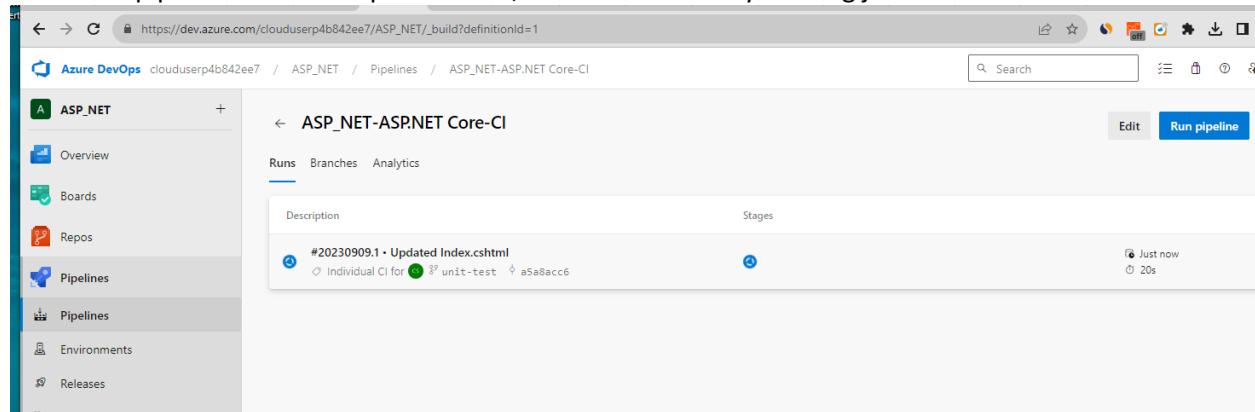
<h1> Congratulations! You Did It!</h1>

At the top, click **Commit > Commit**.

Run the Pipeline and Verify Success

From the left navigation menu, select **Pipelines**.

Select the pipeline shown to open it. Then, select the currently running job.



The screenshot shows the Azure DevOps Pipelines interface. On the left, there's a sidebar with options like Overview, Boards, Repos, and Pipelines. The Pipelines option is selected and highlighted in grey. The main area shows a pipeline named "ASP.NET-ASP.NET Core-CI". A single run is listed under the "Runs" tab, with the status "Just now" and a duration of "20s". The run description is "#20230909.1 • Updated Index.cshtml". Below this, there's a "Recently run pipelines" section showing a single entry for "MyFirstProject-ASP.NET Core-CI" with the same run details.

Pipelines

Recent All Runs

Recently run pipelines

Pipeline

Last run



MyFirstProject-ASP.NET Core-CI

#20230909.1 • Updated Index.cshtml

Individual CI for 63 39 unit-test

Select **Agent job 1** to view the progress.

Azure DevOps Pipeline details:

- Project: ASP.NET
- Pipeline: ASP.NET-ASP.NET Core-CLI
- Job: Agent job 1 (Test)
- Task: .NET Core
- Description: Build, test, package, or publish a dotnet application, or run a custom dotnet command
- Version: 2.221.0
- Author: Microsoft Corporation
- Help: <https://docs.microsoft.com/azure/devops/pipelines/tasks/build/dotnet-core-cli>

Test logs:

```
Starting: Test
=====
1 Task : .NET Core
2 Description : Build, test, package, or publish a dotnet application, or run a custom dotnet command
3 Version : 2.221.0
4 Author : Microsoft Corporation
5 Help : https://docs.microsoft.com/azure/devops/pipelines/tasks/build/dotnet-core-cli
6 =====
7 Info: .NET Core SDK/runtime 2.2 and 3.0 are now End of Life(EOL) and have been removed from all hosted agents. If you're using these
8 /home/azureuser/.myagent/_work/_tool/dotnet/dotnet test /home/azureuser/.myagent/_work/1/s/AspNetCoreWebApplicationTest/AspNetCoreWebAp
9 Test run for /home/azureuser/.myagent/_work/1/s/AspNetCoreWebApplicationTest/bin/Release/netcoreapp3.1/AspNetCoreWebApplicationTest.d
10 Microsoft (R) Test Execution Command Line Tool Version 16.7.1
11 Copyright (c) Microsoft Corporation. All rights reserved.
12
13 Starting test execution, please wait...
14
15 A total of 1 test files matched the specified pattern.
16
17 Results File: /home/azureuser/.myagent/_work/_temp/_linM_2023-09-09_18_51_41.trx
18
19
20 Test Run Successful.
21 Total tests: 2
22 Passed: 2
23 Total time: 2.3819 Seconds
```

It will take a few minutes for the job to complete.
Once complete, click the back arrow above the job progress.

#20230909.1 • Updated Index.cshtml

ASP.NET-ASP.NET Core-CLI

This run is being retained as one of 3 recent runs by unit-test (Branch).

View retention leases

Summary Tests

Triggered by Cloud Student 4b842ee7

View 9 changes

Repository and version	Time started and elapsed	Related	Tests and coverage
ASP.NET unit-test a5a8acc6	Today at 9:50 PM 1m 26s	0 work items 1 published: 1 consumed	100% passed Setup code coverage

Select the **Tests** tab to verify success.

#20230909.1 • Updated Index.cshtml
ASP.NET-ASP.NET Core-CI

This run is being retained as one of 3 recent runs by unit-test (Branch). [View retention leases](#)

[Run new](#) [⋮](#)

[Summary](#) [Tests](#)

Summary

1 Run(s) Completed (1 Passed, 0 Failed)

Total tests	Passed	Failed	Others
2	2	0	0

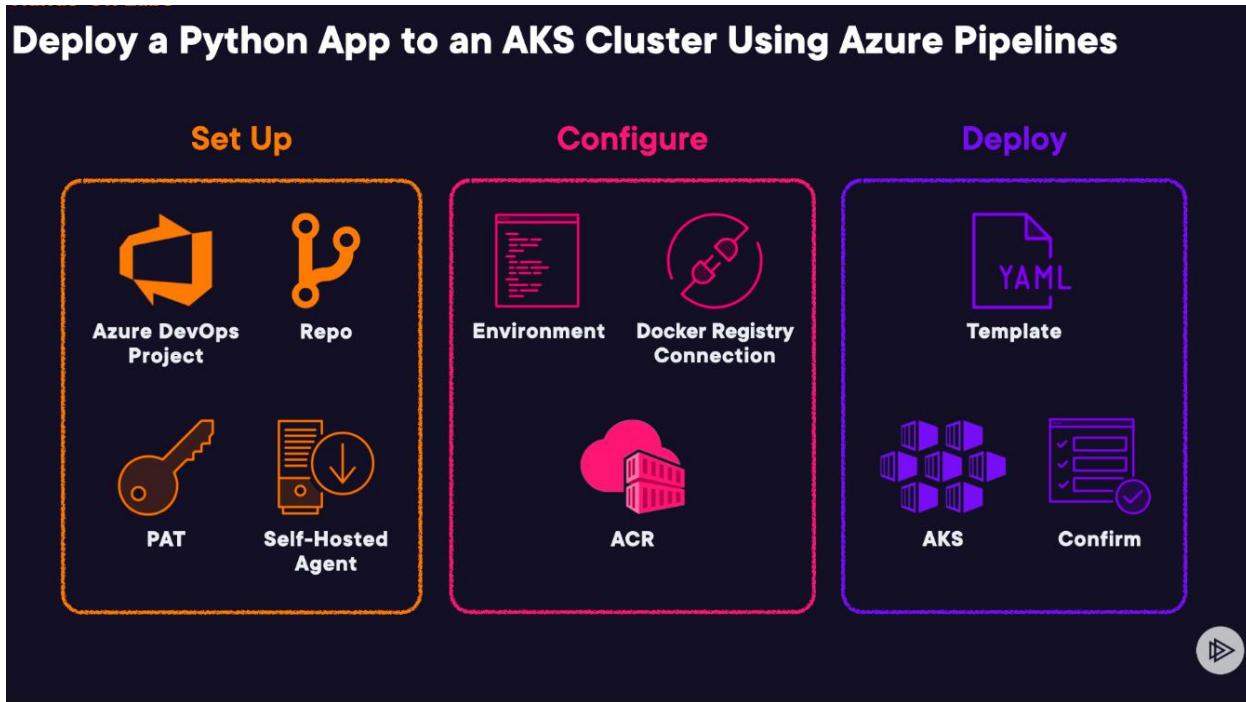
Pass percentage: 100% | Run duration: 2s 403ms | Tests not reported: +2s 403ms

[Bug](#) [Link](#) [Test run](#) [Column Options](#)

Filter by test or run name: Tags Test file Owner Aborted (+1)

Hooray! There are no test failures.

Deploy a Python App to an AKS Cluster Using Azure Pipelines



Pre steps

1. Create a Linux virtual machine (Ubuntu)
2. Create a Container Registry
3. Create a Kubernetes service.
4. Setting up and configuring Docker on the Ubuntu server.

The screenshot shows the Azure portal interface with the following details:

- Resource Group:** 1-c7cc6262-playground-sandbox
- Essentials:** Shows 13 records of resources.
- Resources:**

Type	Location
Container registry	East US
Metric alert rule	Global
Container registry replication	East US
Kubernetes service	East US
Metric alert rule	Global
Action group	Global
Virtual machine	East US
Public IP address	East US

Login to the Azure portal sandbox and create the following resources we see above:

1. Create a Linux virtual machine (Ubuntu)

The screenshot shows the Microsoft Azure Marketplace interface. On the left, there's a sidebar with links like 'Get Started', 'Service Providers', 'Management' (which is expanded), 'Private Marketplace', 'Private Offer Management', 'My Marketplace' (which is expanded), 'Favorites', 'My solutions', 'Recently created', and 'Private plans'. The main area has a search bar at the top with the placeholder 'Search resources, services, and ...'. Below the search bar, there's a 'Create' dropdown menu. Underneath it, there are several cards for different offerings. One card for 'Ubuntu Server 20.04 LTS' by Canonical is visible, showing its logo and a 'Create' button. Other cards include 'Virtual Machine' by Canonical, 'Linux For The Cloud' by Canonical, and another 'Ubuntu Server' card.

2. Create a Container Registry

This screenshot shows the Microsoft Azure Marketplace interface again. The sidebar on the left is identical to the previous one. In the main area, there's a search bar with the placeholder 'Search resources, services, and ...'. Below the search bar, there's a 'Create' dropdown menu. A search bar above the results shows the query 'container registry'. There's also a checkbox labeled 'Azure services only' which is unchecked. The results list shows one item: 'Container Registry' by Microsoft, under the 'Azure Service' category. The description for this item is 'Manage a Docker private registry as a first-class Azure resource.' There are also 'Create' and 'Edit' buttons next to the result card.

3. Create a Kubernetes service.

Azure Kubernetes Service (AKS)

Azure services only

Showing 1 to 20 of 52 results for 'Azu'

Azure Kubernetes Service (AKS)

Microsoft

Azure Service

A managed cluster with a Kubernetes orchestrator for container deployments.

Create

Now, SSH into the virtual machine with the keypair

VM12 | Connect

Virtual machine

Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Networking

Connect

Disks

Size

Microsoft Defender for Cloud

Advisor recommendations

Extensions + applications

Availability + scaling

Configuration

Identity

Properties

Connecting using public IP address
52.146.88.8

Admin username : azuser

Port (change) : 22 [Check access](#)

Just-in-time policy : Not configured for port 22 [Configure for this port](#)

Recommended

SSH using Azure CLI

Quickly connect in browser. Supports Azure AD authentication. Private key not required.

Using public IP address (52.146.88.8)

Select

Most common

Native SSH

No additional software needed. Private key connection. Best for those with existing SSH clients.

Using public IP address (52.146.88.8)

Select

Native SSH

Connect from your local machine (Windows)

Change the port for connecting to this virtual machine on the Connect page of the virtual machine.

Public IP address

A public IP address is required to connect via this connection method.

I understand just-in-time policy on the virtual machine may be re-configured to allow local machine IP (82.102.41.146) to request just-in-time access to port 22.

Configure

Open a local shell (on Windows)

Open Terminal (Windows 11), PowerShell (Windows 10 or less), or a shell of your choice. Or switch the local machine OS above to view more instructions.

Copy and execute SSH command

Provide a path to your SSH private key file on your local machine.

~/ssh/id_rsa.pem

Can't find your private key? [Reset your SSH private key](#)

SSH to VM with specified private key.

ssh -i ~/ssh/id_rsa.pem azuser@52.146.88.8

Other Information

Using a Linux subsystem like WSL or Ubuntu on Terminal?

Copy your private key path to the Linux subsystem and ensure it has the correct read-only access.

Close Troubleshooting Give feedback

ssh -i <key_pair>@<Public IP Address>

cat /etc/os-release

```
azureuser@VM12:~$ cat /etc/os-release
NAME="Ubuntu"
VERSION="20.04.6 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04.6 LTS"
VERSION_ID="20.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=focal
UBUNTU_CODENAME=focal
azureuser@VM12:~$
```

4. Setting up and configuring Docker on the Ubuntu server.

```
sudo apt update
```

```
sudo apt install -y \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg
```

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt update
```

```
sudo apt install docker-ce
```

```
docker --version
```

```
sudo systemctl enable docker
sudo systemctl start docker
```

```
whoami
azureuser
```

```
sudo usermod -aG docker azureuser
sudo systemctl restart docker
```

```
docker run hello-world
```

permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:

Post "<http://var/run/docker.sock/v1.24/images/create?fromImage=tiangolo%2Fuwsgi-nginx-flask&tag=python3.6>": dial unix /var/run/docker.sock: connect: permission denied

Run the following:

```
sudo chmod 666 /var/run/docker.sock
```

Now run the following hello-world command again.

```
docker run hello-world
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
azureuser@VM12:~$
```

Now we have completed the pre steps lets now proceed to setting up the Azure pipeline.

Create an Azure DevOps Organization

The screenshot shows the Azure portal's search interface. In the top search bar, the text "Azure DevOps organizations" is typed. Below the search bar, there are several navigation tabs: "All" (which is selected), "Services (99)", "Documentation (99+)", "Resources (0)", "Resource Groups (0)", and "Marketplace (0)". Under the "Services" section, there is a list of services, with "Azure DevOps organizations" highlighted. Other listed services include Azure Active Directory, Azure Cosmos DB, Azure Database for MySQL servers, Azure Arc, Azure Databricks, Azure Lighthouse, and Azure Migrate. At the bottom of the screen, there is a "Documentation" section with a "See" link.

In the main search bar, search for and select **Azure DevOps organizations**.

Scroll down and click **My Azure DevOps Organizations**.

The screenshot shows the Microsoft Azure portal interface. At the top, there's a header bar with a back arrow, forward arrow, and a refresh icon. The URL in the address bar is <https://portal.azure.com/#view/AzureTfsExtension/OrganizationsTemp>. Below the address bar is a blue navigation bar with the Microsoft Azure logo and a search bar that says "Search resources, services, and docs (G+/)". Underneath the navigation bar, the breadcrumb trail shows "Home > Azure DevOps ...". A prominent orange callout box contains the text: "We've made it easier to manage Azure DevOps billing and subscriptions. You can [set up billing](#) or [switch organizations](#)". The main content area features a large blue heading "Azure DevOps" followed by a dark gray subtext: "Plan smarter, collaborate better, and ship faster with a set of modern dev services". Below this, there are three blue links: "My Azure DevOps Organizations", "Get started using Azure DevOps", and "Billing management for Azure DevOps".

Select your **country/region** from the dropdown list.

Select Create **new organization** > **Continue** > **Continue** (complete the required **CAPTCHA**).



Get started with Azure DevOps

Plan better, code together, ship faster with Azure DevOps

[Create new organization](#)

Set the following values to create a project:

Project name: **Dev**

Visibility: **Private**

Click + **Create project**.

Import Code and Set Up the Environment:

In the sidebar menu, select **Repos**.

In the Import a repository section, click **Import**.

In the Clone URL field, enter the following repository:

<https://github.com/ACloudGuru-Resources/content-az400-lab-resources.git>

Click **Import**.

After the code import completes successfully, the Files page should open automatically.

The screenshot shows the Azure DevOps interface for a repository named 'DEV'. The left sidebar has 'D DEV' selected. The main area shows a list of files under the 'aks' branch. At the top right, there's a green checkmark icon labeled 'succeeded' and a 'Clone' button. A 'Create a pull request' button is also visible. The file list includes 'azure-vote', 'manifests', '.gitignore', 'azure-pipelines.yml', 'Dockerfile', and 'README.md'.

At the top of the page, use the dropdown to switch the branch from **DSC** to **aks**.

You should now see the files you need for the lab as above.

Create Personal Access Token

In the upper right-hand corner, click the **User settings icon > Personal access tokens**.

Click **+ New Token** and set the following values:

Name: **pat1**

Scopes: **Full access**

Click **Create**.

A modal dialog box titled 'Success!' displays a message: 'You have successfully added a new personal access token. Copy the token now!'. Below this, a text input field contains the token value: 'fjtp7hxp6wu3hcyybqsgtzc'. A warning message in a yellow box says: 'Warning - Make sure you copy the above token now. We don't store it and you will not be able to see it again.' At the bottom right of the dialog is a blue 'Close' button.

Copy the **pat1** token and save it in a safe location.

Click **Close**.

Download Self-Hosted Agent

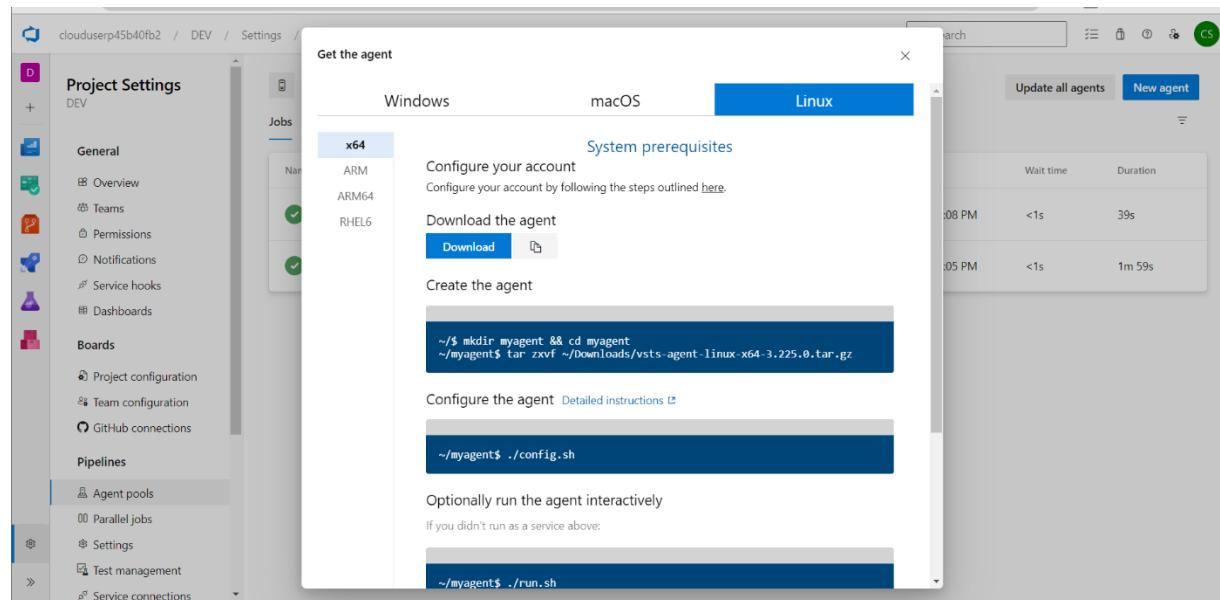
At the top of the page, click the **Azure DevOps icon**, then select the **project**.

On the bottom left, click **Project settings**.

Scroll down the left-hand pane and under **Pipelines**, select **Agent pools**.

Select the **Default pool**.

Click **New agent**.



Select the **Linux** tab.

Click the **Copy** icon, next to **Download** to copy the agent URL to your clipboard.
Paste it to a text file to use in the following steps. Keep this tab and popup open.

For example:

<https://vstsagentpackage.azureedge.net/agent/3.225.0/vsts-agent-linux-x64-3.225.0.tar.gz>

Setup the agent on the virtual machine

Ensure the directory is set to the root:

`cd ~`

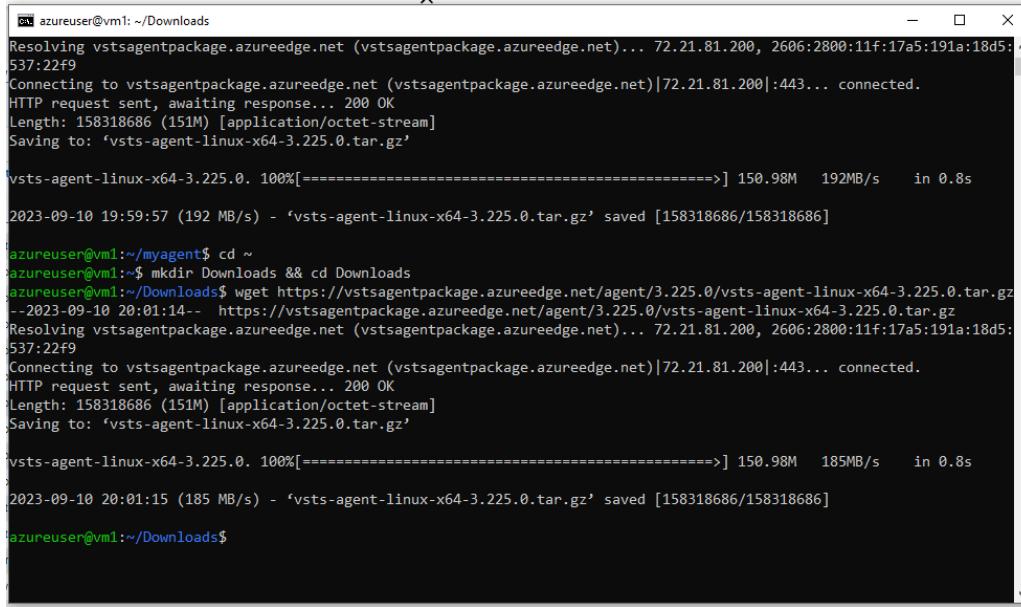
Create and navigate to another directory called `myagent`:

`mkdir myagent && cd myagent`

Download the tar file within the agent directory.

```
wget <AGENT_URL>
```

```
wget https://vstsagentpackage.azureedge.net/agent/3.225.0/vsts-agent-linux-x64-3.225.0.tar.gz
```



The screenshot shows a terminal window with two separate sessions of the wget command. In the first session, the user downloads the agent package from the Azure DevOps URL. The second session shows the user navigating to the Downloads directory and running wget again, likely to verify the download or to download it again. The terminal output includes details like the connection, file length, and download speed.

```
azureuser@vm1:~/Downloads
Resolving vstsagentpackage.azureedge.net (vstsagentpackage.azureedge.net)... 72.21.81.200, 2606:2800:11f:17a5:191a:18d5:537:22f9
Connecting to vstsagentpackage.azureedge.net (vstsagentpackage.azureedge.net)|72.21.81.200|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 158318686 (151M) [application/octet-stream]
Saving to: 'vsts-agent-linux-x64-3.225.0.tar.gz'

vsts-agent-linux-x64-3.225.0. 100%[=====] 150.98M 192MB/s in 0.8s

2023-09-10 19:59:57 (192 MB/s) - 'vsts-agent-linux-x64-3.225.0.tar.gz' saved [158318686/158318686]

azureuser@vm1:~/myagent$ cd ~
azureuser@vm1:~$ mkdir Downloads && cd Downloads
azureuser@vm1:~/Downloads$ wget https://vstsagentpackage.azureedge.net/agent/3.225.0/vsts-agent-linux-x64-3.225.0.tar.gz
--2023-09-10 20:01:14- https://vstsagentpackage.azureedge.net/agent/3.225.0/vsts-agent-linux-x64-3.225.0.tar.gz
Resolving vstsagentpackage.azureedge.net (vstsagentpackage.azureedge.net)... 72.21.81.200, 2606:2800:11f:17a5:191a:18d5:537:22f9
Connecting to vstsagentpackage.azureedge.net (vstsagentpackage.azureedge.net)|72.21.81.200|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 158318686 (151M) [application/octet-stream]
Saving to: 'vsts-agent-linux-x64-3.225.0.tar.gz'

vsts-agent-linux-x64-3.225.0. 100%[=====] 150.98M 185MB/s in 0.8s

2023-09-10 20:01:15 (185 MB/s) - 'vsts-agent-linux-x64-3.225.0.tar.gz' saved [158318686/158318686]

azureuser@vm1:~/Downloads$
```

Unzip the package you just downloaded with the following command:

```
azureuser@VM12:~$ cd ./myagent/
azureuser@VM12:~/myagent$ ls
_diag bin env.sh license.html run.sh vsts-agent-linux-x64-3.225.0.tar.gz
_work config.sh externals run-docker.sh svc.sh
azureuser@VM12:~/myagent$
```

```
tar zxvf <PACKAGE>
```

You can also copy this same command from the popup in Azure DevOps to get the full package and version number:

```
azureuser@VM12:~$ cd ./myagent/
azureuser@VM12:~/myagent$ ls
_diag bin env.sh license.html run.sh vsts-agent-linux-x64-3.225.0.tar.gz
_work config.sh externals run-docker.sh svc.sh
azureuser@VM12:~/myagent$ tar zxvf vsts-agent-linux-x64-3.225.0.tar.gz
```

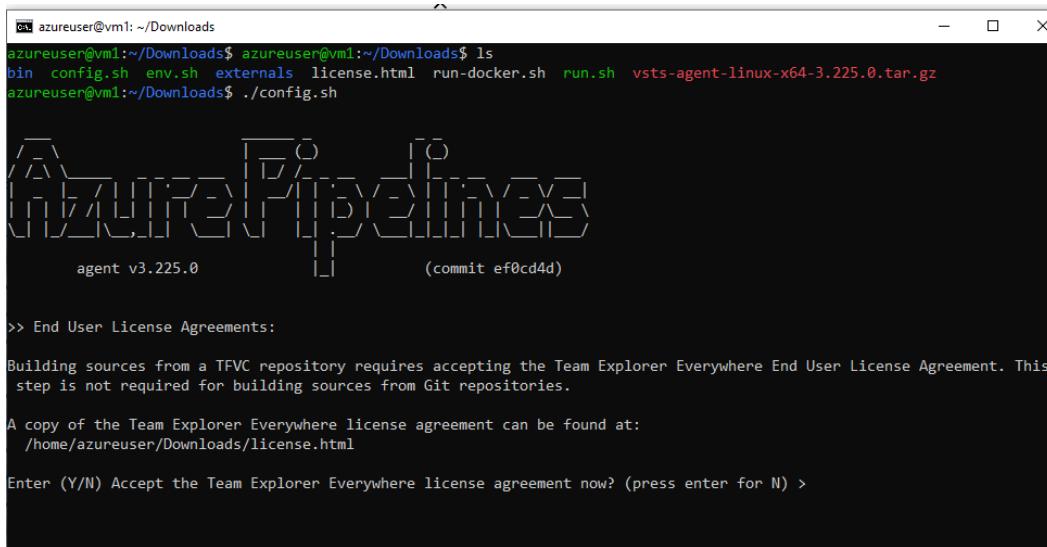
Review the contents of the directory (**ls**). Observe all the files and folders for the agent.

Configure the Agent

Run the configuration script with the following command:

```
./config.sh
```

Type **y** to accept the license agreement.



The screenshot shows a terminal window with the following content:

```
azureuser@vm1:~/Downloads$ ./config.sh
agent v3.225.0          (commit ef0cd4d)

>> End User License Agreements:
Building sources from a TFVC repository requires accepting the Team Explorer Everywhere End User License Agreement. This step is not required for building sources from Git repositories.

A copy of the Team Explorer Everywhere license agreement can be found at:
/home/azureuser/Downloads/license.html

Enter (Y/N) Accept the Team Explorer Everywhere license agreement now? (press enter for N) >
```

When prompted for your server URL, go back to Azure DevOps and grab the URL from your browser. It will look something like <https://dev.azure.com/clouduser...>, followed by some characters (i.e., copy everything up to but not including /dev).

For example:

<https://dev.azure.com/clouduserp45b40fb2>

Paste this in the terminal

Press **Enter** to accept the default authentication type of **PAT**.

For personal access token, copy and paste the personal access token you created and saved earlier.

```
az user@vm1:~/Downloads$ azureuser@vm1:~/Downloads$ ls
bin config.sh env.sh externals license.html run-docker.sh run.sh vsts-agent-linux-x64-3.225.0.tar.gz
azureuser@vm1:~/Downloads$ ./config.sh


agent v3.225.0          |          (commit ef0cd4d)

>> End User License Agreements:

Building sources from a TFVC repository requires accepting the Team Explorer Everywhere End User License Agreement. This step is not required for building sources from Git repositories.

A copy of the Team Explorer Everywhere license agreement can be found at:
/home/azureuser/Downloads/license.html

Enter (Y/N) Accept the Team Explorer Everywhere license agreement now? (press enter for N) > y

>> Connect:

Enter server URL > https://dev.azure.com/clouduserpd2bb1751
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) >
Enter agent name (press enter for vm1) >
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2023-09-10 20:06:19Z: Settings Saved.
azureuser@vm1:~/Downloads$
```

Press Enter to accept the default **agent pool**.

Press Enter to accept the default name.

Press Enter to accept the default **work folder**.

Now run the agent interactively so that it is always listening for any jobs:

./run.sh

```
azureuser@vm1:~/Downloads$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2023-09-10 20:07:13Z: Listening for Job
```

Now let's go back to the DevOps portal

Create Pipeline Environment

In the sidebar menu, select **Pipelines**, then select **Environments**.

Click Create environment.

Fill in the environment details:

Name: Enter dev

Resource: Select **Kubernetes**.

Click **Next**.

Ensure the resource details auto-populate, then click **Validate and create**.

The screenshot shows the Azure DevOps Pipelines Environments page. On the left, there's a sidebar with options like Overview, Boards, Repos, Pipelines, Environments (which is selected), Library, Test Plans, and Artifacts. The main area displays an 'Environments' table with one row named 'Dev'. To the right, a modal window titled 'New environment' is open. It has fields for 'Name' (containing 'Dev') and 'Description' (with the placeholder 'Describe the environment'). Under the 'Resource' section, there are three options: 'None' (selected), 'Kubernetes' (with the sub-note 'Add Kubernetes namespace'), and 'Virtual machines'.

Create a Service Connection and an ACR

In the bottom-left corner, click **Project Settings**.

This opens the settings in a new tab.

In the sidebar menu, select **Service connections**.

In the top right, click **New service connection**.

In the pane on the right, select **Docker Registry**, then click **Next**.

You'll need to gather some information from an **Azure Container Registry (ACR)** before you finish creating your service connection.

The screenshot shows the 'Service connections' page in Azure DevOps. On the left, there's a sidebar with 'Project Settings' (DEV), 'General' (Overview, Teams, Permissions, Notifications, Service hooks, Dashboards), 'Boards' (Project configuration, Team configuration, GitHub connections), and 'Pipelines' (Agent pools, Parallel jobs, Settings, Test management). The main area is titled 'Service connections' with a search bar 'Filter by keywords'. It lists two connections: 'ACR' and 'Dev-K8-default-1694508846439'. To the right, a modal window titled 'New Docker Registry service connection' is open. It contains fields for 'Docker Registry' (https://index.docker.io/v1/), 'Docker ID' (acrqnqworsxpnyk), 'Docker Password' (redacted), 'Email (optional)' (empty), 'Service connection name' (ACR), and 'Description (optional)' (empty). Under 'Security', there's a checked checkbox 'Grant access permission to all pipelines'. At the bottom right of the modal are 'Back' and 'Save' buttons.

Navigate back to your Azure DevOps tab.

Navigate to **All resources** using the hamburger menu in the top-left corner.

Select the link for the **Azure container registry (ACR)**.

In the ACR's sidebar menu, select **Access keys** (under **Settings**).
You'll use these details for your service connection.

The screenshot shows the 'Access keys' page for a container registry named 'con1234567'. The left sidebar includes 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Quick start', 'Events', 'Settings' (selected), 'Access keys' (selected), 'Encryption', 'Identity', 'Networking', and 'Microsoft Defender for Cloud'. The main area displays two sets of access key pairs. The first pair is for 'password': 'Name' is 'password' and 'Password' is '9iT+eTuXkeH1jc+247LSWF9yy2dARzI5V5xWkVC5oq+ACR...'. The second pair is for 'password2': 'Name' is 'password2' and 'Password' is '7VcfYP3tSpQIKtfA7cklbiVrB5TgCe7r3mn/Yaxxy+ACRA0Jyj'. Each password field has a 'Regenerate' button to its right.

Add the New Docker Registry service connection details:

Fill in the Docker Registry:

From the Access keys page, copy the **Login server**.

Navigate to the service connections tab and paste the login server into the **Docker Registry** field using the format https://<LOGIN_SERVER>.

Fill in the Docker ID:

Navigate to the ACR tab and copy the **Username**.

Navigate to the service connections tab and paste the username into the **Docker ID** field.

Fill in the Docker Password:

Navigate to the ACR tab and copy either **password**.

Navigate to the service connections tab and paste the password into the **Docker Password** field.

In the Service connection name field, enter **ACR**.

Below Security, check the **Grant access permission to all pipelines** checkbox.

Click **Save**.

The screenshot shows the 'ACR' service connection in the Azure DevOps interface. The 'Overview' tab is selected. The 'Service connection type' is listed as 'Docker Registry using basic authentication'. There is also a 'Details' section which is currently collapsed.

Create the CI/CD Pipeline

Update the Deployment Image

From the service connections tab, use the sidebar menu to select **Repos**.

Select the **manifests** folder and open the **vote.yml** file.

In the top right, click **Edit**.

Update the deployment image:

Navigate to the **ACR** tab and copy the **Login server** again.

Navigate to the **vote.yaml** tab and replace the container image on line 59 with the Azure ACR DNS registry name, following the format `<LOGIN_SERVER>/azure-vote-front:v1`.

The screenshot shows the Azure DevOps interface for the 'DEV' project. The left sidebar is open, showing options like Overview, Boards, Repos, Files, Pipelines, Test Plans, and Artifacts. The 'Files' option is selected. In the main area, the path 'DEV / Repos / Files / aks / manifests / vote.yml' is displayed. The file 'vote.yml' is shown with its contents. A yellow box highlights the 'image' field in the YAML code, which is set to 'con1234567.azurecr.io/azure-vote-front:v1'. The code is as follows:

```
40 ---  
41 apiVersion: apps/v1  
42 kind: Deployment  
43 metadata:  
44 | name: azure-vote-front  
45 spec:  
46 | replicas: 1  
47 | selector:  
48 |   matchLabels:  
49 |     app: azure-vote-front  
50 | template:  
51 |   metadata:  
52 |     labels:  
53 |       app: azure-vote-front  
54 |     spec:  
55 |       nodeSelector:  
56 |         "beta.kubernetes.io/os": linux  
57 |       containers:  
58 |         - name: azure-vote-front  
59 |           image: con1234567.azurecr.io/azure-vote-front:v1  
60 |           resources:  
61 |             requests:  
62 |               cpu: 100m  
63 |               memory: 128Mi  
64 |             limits:  
65 |               cpu: 250m  
66 |               memory: 256Mi
```

Note: The ACR DNS name ends with azurecr.io.

In the top right, click **Commit**.

In the Commit pane, ensure the Branch name is set to **aks**, then click **Commit**.

Create the Pipeline

In the sidebar menu, select **Pipelines**.

Click **Create Pipeline**.

For the code location, select **Azure Repos Git**.

For the repository, select the dev repo.

Select **Existing Azure Pipelines YAML file**.

Fill in the YAML file details:

Branch: Select **aks**.

Path: Select **/azure-pipelines.yml**.

Click **Continue**.

Update the **azure-pipelines.yml** code:

Navigate to the **ACR** tab and copy the **Login server** again.

Navigate to the **pipeline** tab and replace the value for container Registry with your copied Azure ACR DNS registry name.

```

trigger:
- aks

variables:
  dockerRegistryServiceConnection: 'ACR'
  imageRepository: 'azure-uvote-front'
  containerRegistry: 'ccr234567.azurecr.io'
  dockerfilePath: '**/Dockerfile'
  tag: '${{build.buildId}}'
  imagePullSecret: 'csacr5d93-auth'

stages:
- stage: Build
  displayName: Build stage
  jobs:
  - job: Build
    displayName: Build
  
```

In the top-right corner, click **Save and run**.

In the pane on the right, leave the default settings and click **Save and run**.

After the pipeline is triggered, select **Build stage** to monitor the build progress.

Note the banner in the terminal indicating that the pipeline needs additional permissions to continue the build.

Click **View** to the right of the banner.

Click **Permit** to the right of the requested permissions and confirm the selection.

The screenshot shows the Azure DevOps interface for a project named 'DEV'. The left sidebar has 'Pipelines' selected. The main area displays a 'Build stage' for run #20230912.1. The stage includes tasks like 'Initialize job', 'Checkout DEV@aks to s', 'Build and push ...', 'PublishPipelineArtifact', 'Post-job: Checkout DEV@aks to s', and 'Finalize Job'. A terminal window on the right shows the log for the 'Post-job: Checkout DEV@aks to s' task, which includes commands for starting, getting sources, and finishing the checkout process.

```
1 Starting: Checkout DEV@aks to s
2 =====
3 Task      : Get sources
4 Description: Get sources from a repository. Supports Git, Tfvc, and SVN repositories.
5 Version   : 1.0.0
6 Author    : Microsoft
7 Help      : [More Information](https://go.microsoft.com/fwlink/?LinkId=798199)
8 =====
9 Cleaning any cached credential from repository: DEV (Git)
10 Finishing: Checkout DEV@aks to s
```

After the build has completed, select **Deploy to dev**.

The screenshot shows the Azure DevOps interface for the same project 'DEV'. The left sidebar has 'Pipelines' selected. The main area displays a 'Build stage' for run #20230912.1. Below it, a 'Deploy to dev' stage is shown. This stage includes tasks like 'Initialize job', 'Download Artifact', 'Create imagePullSecret', 'Deploy to Kubernetes ...', and 'Finalize Job'. A terminal window on the right shows the log for the 'Deploy to dev' task, which details the deployment process on agent VM12, including live console data and command logs for DetectDockerContainer.

```
1 Agent: VM12
2 Started: Just now
3 Duration: 35s
4
5 Job live console data:
6 Starting: Deploy to dev
7 Async Command Start: DetectDockerContainer
8 Async Command End: DetectDockerContainer
9 Async Command Start: DetectDockerContainer
10 Async Command End: DetectDockerContainer
11 Finishing: Deploy to dev
```

Note the banner in the terminal indicating that the pipeline needs additional permissions to continue the deployment. If the banner does not display. Click the back arrow above the deployment to go back to the job summary. Then, click View in the yellow notification box.

Like before, allow all permissions. The deployment will take a few minutes to complete.

Access the AKS Cluster

Navigate to the ACR tab.

At the top, click **All resources**.

Select the link for the **Kubernetes service** cluster. (You can ignore any errors that appear as they are related to restrictions).

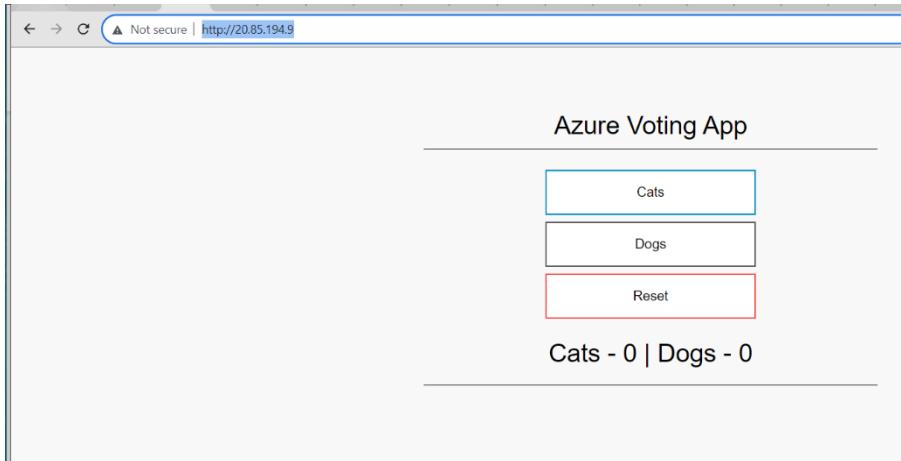
On the left menu, under **Kubernetes resources**, select **Services and ingresses**.

All services in the cluster are displayed. Observe the azure-vote-front service has an **External IP**

Click the link for the IP.

Name	Namespace	Status	Type	Cluster IP	External IP	Ports	Age
kubernetes	default	Ok	ClusterIP	10.0.0.1	443/TCP	4 hours	...
kube-dns	kube-system	Ok	ClusterIP	10.0.0.10	53/UDP,53/TCP	4 hours	...
metrics-server	kube-system	Ok	ClusterIP	10.0.159.21	443/TCP	4 hours	...
azure-vote-back	default	Ok	ClusterIP	10.0.50.102	6379/TCP	3 hours	...
azure-vote-front	default	Ok	LoadBalancer	10.0.161.109	20.85.194.9	80:32219/TCP	3 hours

The app opens in a new tab where you can click to vote on cats or dogs.



Addendum

Errors to avoid

1) The pipeline gives this error!

A screenshot of an Azure Pipeline run titled "#20230910.1 • Set up CI with Azure Pipelines" under the "Python_App" project. The run status is failed. A message says "This run will be cleaned up after 1 month based on your project settings." The "Summary" section shows the run was triggered by "Cloud Student d2bb1751" and started "Just now" ago. It lists repository details ("Python_App", "aks", "93ca53c3") and related items (0 work items, 0 artifacts). The "Tests and coverage" section has a "Get started" link. The "Errors" section shows one error: "Job Deploy: Resource default does not exist in environment dev" from 20230910.1. A link to documentation for troubleshooting failed runs is at the bottom.

Solution: Is your Docker installed and properly configured on your virtual machine?
Please follow the instructions above in the pre steps!

2) The pipeline build gives this error!

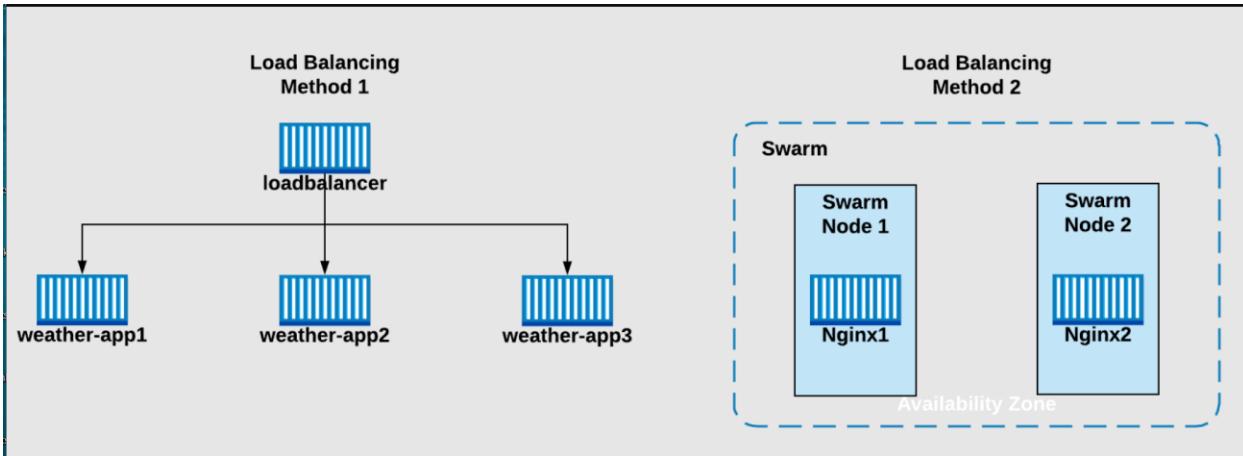
```
##[error]ERROR: permission denied while trying to connect to the Docker daemon  
socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/_ping":  
dial unix /var/run/docker.sock: connect: permission denied  
##[error]The process '/usr/bin/docker' failed with exit code 1
```

Solution: As indicated above, this is a permission issue related to your Docker on your virtual machine. Instructions for assigning permissions so that the DevOps pipeline can be

deployed to the virtual machine are above in pre steps.

Note: if the hello world test above fails with a similar message. Please do not proceed further with the Pre steps until this permission error is resolved. Otherwise, the pipeline Will fail!

Load Balancing Containers (Docker Swarm and Docker Compose)



Solution

This is as follows:

Prerequisites.

Before you begin, make sure you have:

Two CentOS 7 servers; one to be the master and the other to be the worker.

Master and Worker

```
[root@ip-10-0-1-4 ~]# cat /etc/os-release
```

```
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"
```

You will need to have the following inbound port rules set.

Port **22** and **8080** need to be open.

SSH access to both servers with sudo privileges.

When setting up both server instances in the cloud, you will need to generate two .PEM key pairs to gain remote access to.

Installation and Configuration on the Master Node

Step 1: Update the System

SSH into the master server and update the system packages:

sudo yum update -y

Step 2: Install Docker

Install Docker on the master server:

sudo yum install -y docker

Step 3: Start and Enable Docker

Start and enable the Docker service:

sudo systemctl start docker

sudo systemctl enable docker

Step 4: Install Docker Compose

Install Docker Compose on the master node:

sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-\$(uname -s)-\$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

Installation and Configuration on the Worker Node

Step 1: Update the System

SSH into the master server and update the system packages:

sudo yum update -y

Step 2: Install Docker

Install Docker on the master server:

sudo yum install -y docker

Step 3: Start and Enable Docker

Start and enable the Docker service:

```
sudo systemctl start docker  
sudo systemctl enable docker
```

Step 4: Install Docker Compose

Install Docker Compose on the master node:

```
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-  
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

Work on the Master node

```
ssh cloud_user@PUBLIC_IP_ADDRESS
```

Become the root user:

```
sudo su -
```

Create a Docker Compose file

Look at the directories available to you.

```
ls
```

```
[root@ip-10-0-1-103 ~]# ls  
anaconda-ks.cfg  lb-challenge  swarm-token.txt  
[root@ip-10-0-1-103 ~]# pwd  
/root
```

Change to the lb-challenge directory:

```
cd lb-challenge
```

Use the VIM editor to create the **docker-compose.yml** file.

```
vi docker-compose.yml
```

Use **Ctrl + :**

type **set paste**

Use the letter I for insert

```
version: '3.2'
services:
  weather-app1:
    build: ./weather-app
    tty: true
    networks:
      - frontend
  weather-app2:
    build: ./weather-app
    tty: true
    networks:
      - frontend
  weather-app3:
    build: ./weather-app
    tty: true
    networks:
      - frontend

  loadbalancer:
    build: ./load-balancer
    image: nginx
    tty: true
    ports:
      - '80:80'
    networks:
      - frontend

networks:
  frontend:
```

Now copy paste or type the above in the format as above.

When you have completed the above do the following to save and exit the file:

Ctrl + :
wq <enter>

Update nginx.conf

Change to the load-balancer directory:

cd load-balancer

Edit the nginx.conf file:

vi nginx.conf

The contents of your **nginx.conf** file should look like the following:

```
events { worker_connections 1024; }

http {
    upstream localhost {
        server weather-app1:3000;
        server weather-app2:3000;
        server weather-app3:3000;
    }
    server {
        listen 80;
        server_name localhost;
        location / {
            proxy_pass http://localhost;
            proxy_set_header Host $host;
        }
    }
}
```

Note: Please ensure that the format is as above. Otherwise the **nginx** file will not be processed!
Once done, save and exit with **wq <enter>**

Execute docker-compose up

Change directories:

```
cd ../
```

Execute the docker-compose up command:

```
docker-compose up --build -d
```

```
Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.  
To deploy your application across the swarm, use `docker stack deploy`.  
  
Creating network "lb-challenge_frontend" with the default driver  
Building weather-app1  
Step 1/10 : FROM node  
latest: Pulling from library/node  
167b8a53ca45: Download complete  
b47a222d28fa: Download complete  
debece5f9f3a9: Download complete  
1d7ca7cd2e06: Downloading [=====] 29.14MB/211MB  
94c7791033e8: Download complete  
72ab0dfaf5cb: Downloading [=====] 7.796MB/47.86MB  
3316ed2852d4: Download complete  
ef5505406bea: Waiting  
[  
]
```

```
Step 10/10 : CMD ./bin/www  
--> Using cache  
--> 746f221ee7ca  
Successfully built 746f221ee7ca  
Successfully tagged lb-challenge_weather-app3:latest  
Building loadbalancer  
Step 1/4 : FROM nginx  
latest: Pulling from library/nginx  
a803e7c4b030: Pull complete  
8b625c47d697: Pull complete  
4d3239651a63: Pull complete  
0f816efa513d: Pull complete  
01d159b8db2f: Pull complete  
5fb9a81470f3: Pull complete  
9b1e1e7164db: Pull complete  
Digest: sha256:32da30332506740a2f7c34d5dc70467b7f14ec67d912703568daff790ab3f  
Status: Downloaded newer image for nginx:latest  
--> 61395b4c586d  
Step 2/4 : COPY nginx.conf /etc/nginx/nginx.conf  
--> e9a8196303dc  
Step 3/4 : EXPOSE 80  
--> Running in 065c58de4883  
Removing intermediate container 065c58de4883  
--> 4ceaad870e19  
Step 4/4 : CMD ["nginx", "-g", "daemon off;"]  
--> Running in ad2713a8c142  
Removing intermediate container ad2713a8c142  
--> b72755fd3d7c  
Successfully built b72755fd3d7c  
Successfully tagged nginx:latest  
Creating lb-challenge_weather-app3_1 ... done  
Creating lb-challenge_weather-app1_1 ... done  
Creating lb-challenge_loadbalancer_1 ... done  
Creating lb-challenge_weather-app2_1 ... done  
[root@ip-10-0-1-4 lb-challenge]# [  
]
```

Now let's check our work:

docker ps

Copy the public IP address of the Swarm master and paste it into a new tab in your browser.

Create a Docker service using Docker Swarm

Change to the root directory:

cd ~/

View the contents of swarm-token.txt:

cat swarm-token.txt

```
[root@ip-10-0-1-4 lb-challenge]# cd ~/
[root@ip-10-0-1-4 ~]# cat swarm-token.txt
Swarm initialized: current node (yt05f9gtvto20vp2s209mjy5i) is now a manager.

To add a worker to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-17j7h3wn62fkvit6zevjebaarb2o5imtrmdux7dwbonvnid4h1-3if43dsktfc7re2c8mezyl0i1 10.0.1.4:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[root@ip-10-0-1-4 ~]#
```

Copy the docker swarm join command from the previous step.

Note: You can also initiate this process external to the script by running the following on the Master:

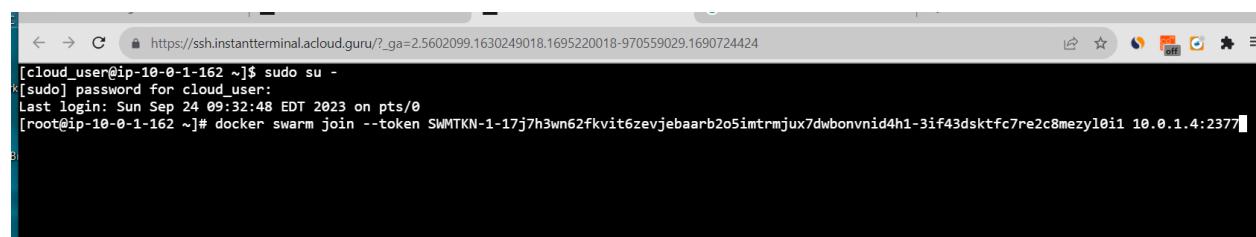
sudo docker swarm init

On Swarm worker node

Execute the command that was copied from the previous step.

i.e., the string docker swarm join –token <token> <IP address>:<port>

docker swarm join --token SWMTKN-1-17j7h3wn62fkvit6zevjebaarb2o5imtrmdux7dwbonvnid4h1-3if43dsktfc7re2c8mezyl0i1 10.0.1.4:2377



```
[cloud_user@ip-10-0-1-162 ~]$ sudo su -
[sudo] password for cloud_user:
Last login: Sun Sep 24 09:32:48 EDT 2023 on pts/0
[root@ip-10-0-1-162 ~]# docker swarm join --token SWMTKN-1-17j7h3wn62fkvit6zevjebaarb2o5imtrmjux7dwbonnid4h1-3if43dsktfc7re2c8mezy1oi1 10.0.1.4:2377
This node joined a swarm as a worker.
[root@ip-10-0-1-162 ~]#
```

On Swarm master

Create a Docker service by executing the following command:

```
docker service create --name nginx-app --publish published=8080,target=80 --replicas=2 nginx
```

```
[root@ip-10-0-1-4 ~]# docker service create --name nginx-app --publish published=8080,target=80 --replicas=2 nginx
[cp0sutcgldlw8mpb5ivjb4y2
overall progress: 2 out of 2 tasks
1/2: running [=====>]
2/2: running [=====>]
Verify: Service converged
[root@ip-10-0-1-4 ~]#
```

Note: the port 8080 used here. Please ensure that its open.

Verify this completed successfully by running the following command on both servers:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
601698d167e5	nginx:latest	"/docker-entrypoint..."	About a minute ago	Up About a minute	80/tcp	nginx-app.1.1beqr
b2kp29b94bqzqyrlrg3q						
7e3cd322282	nginx	"/docker-entrypoint..."	8 minutes ago	Up 8 minutes	0.0.0.0:80->80/tcp, :::80->80/tcp	lb-challenge_load
balancer_1						
98301bc56291	lb-challenge_weather-app2	"docker-entrypoint.s..."	8 minutes ago	Up 8 minutes	3000/tcp	lb-challenge_weat
her-app2_1						
3ad6e3a29f83	lb-challenge_weather-app3	"docker-entrypoint.s..."	8 minutes ago	Up 8 minutes	3000/tcp	lb-challenge_weat
her-app3_1						
d3a92f585e7	lb-challenge_weather-app1	"docker-entrypoint.s..."	8 minutes ago	Up 8 minutes	3000/tcp	lb-challenge_weat
her-app1_1						

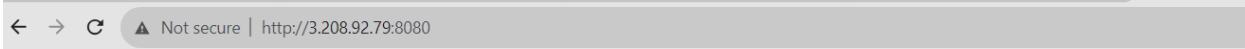
```
[root@ip-10-0-1-4 ~]#
```

docker ps

Note: If you wish to see all containers (whether running or shutdown) use the **docker ps -a** command.

Verify that the default nginx page loads in the browser:

PUBLIC_IP_ADDRESS:8080



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Addendum

How to download files used for this project between your local machine and Centos.

```
[root@ip-10-0-1-103 ~]# ls  
anaconda-ks.cfg  lb-challenge  swarm-token.txt  
[root@ip-10-0-1-103 ~]# pwd  
/root
```

The files, due to the permissions, can only be worked on by someone with root account access. Hence you can only see the above with the root account.

You may, however, wish to download the files and folders to work on them from your own local machine.

To make matter more complicated, when using **WinSCP**, you are unable to login as a root user. Hence, you will not be able to download the files!

Rather than playing with both the permissions and commands like **sudo visudo** that can weaken security you can instead copy the files from the root profile to a non-root profile as follows:

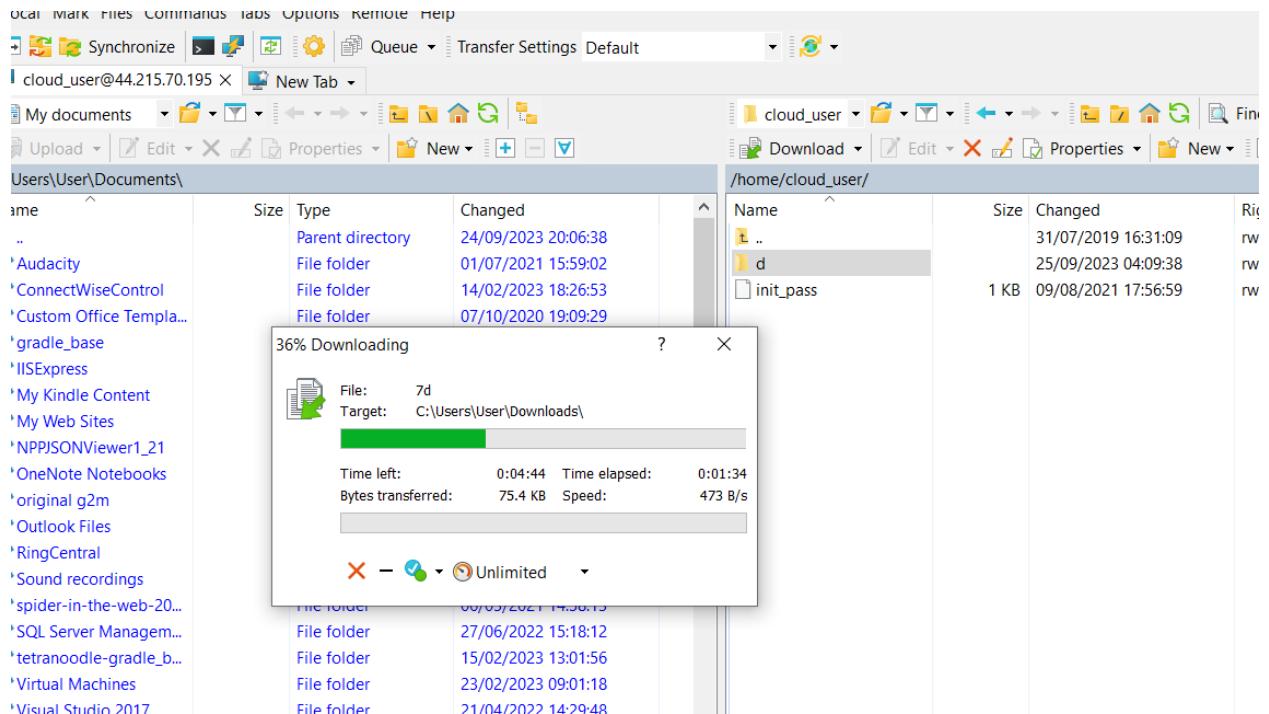
```
[root@ip-10-0-1-103 ~]# cp -r /root/lb-challenge/ /home/cloud_user/d  
[root@ip-10-0-1-103 ~]# cp -r /root/anaconda-ks.cfg /home/cloud_user/d  
[root@ip-10-0-1-103 ~]# cp -r /root/swarm-token.txt /home/cloud_user/d  
[root@ip-10-0-1-103 ~]# █
```

Note: **pwd** is helpful in constructing the folder paths.

cp == copy

-r == recursive command. Good when you have subfolders.

/path/source/ /path/destination

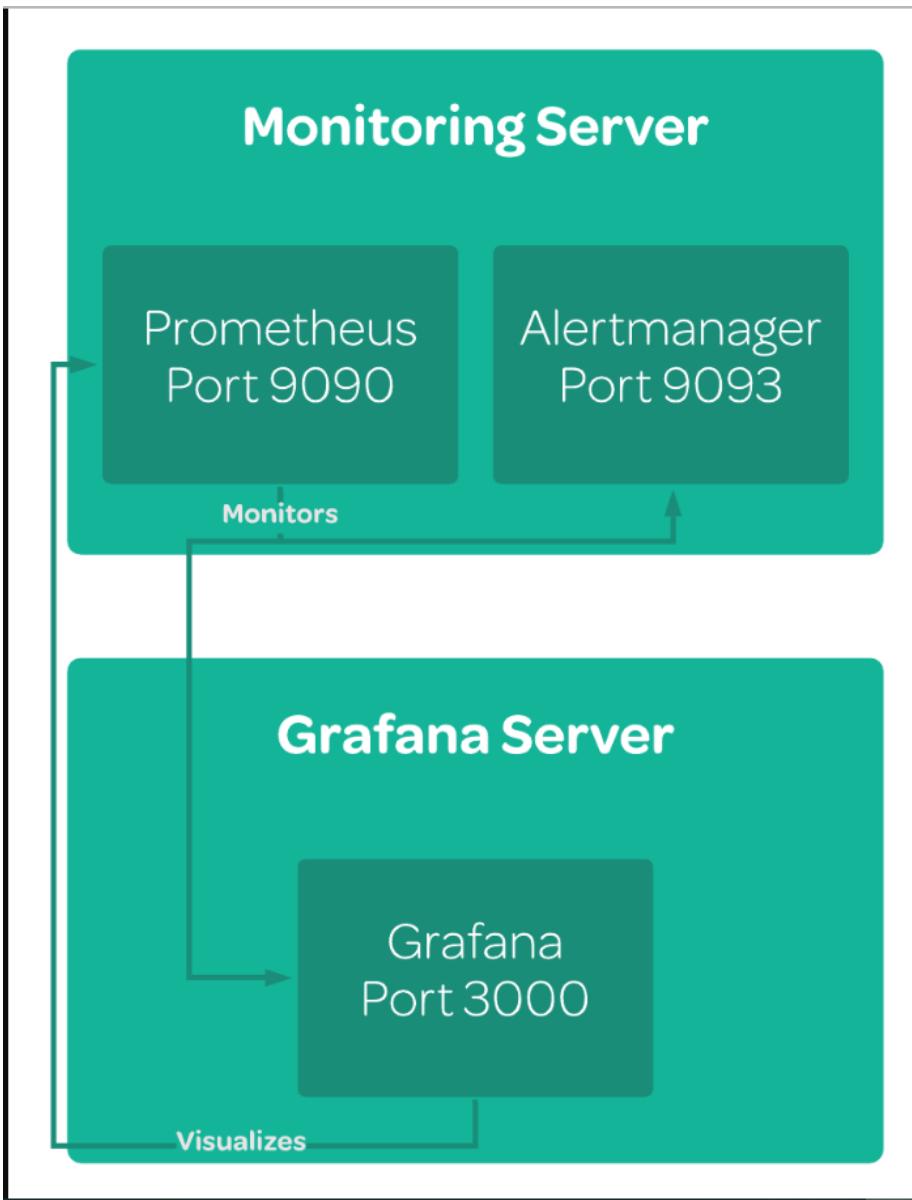


If you have any 'stubborn' files that won't download. No problem!

```
[root@ip-10-0-1-103 ~]# cat anaconda-ks.cfg
```

Use the cat command! Copy and paste the text locally.

Setting Up Prometheus and Adding Endpoints



Solution

Prerequisites:

You need two Linux servers.

One for installing **Prometheus** and the **Alert Manager**
The other for **Grafana**.

Call one **Monitoring** and the other **Grafana**.

As the following commands will, likely, run on any distro I am agnostic on which one you may choose.

For this project I will be using Centos 7

Logging In

NOTE: It's probably a good idea to have two or three terminals open, one for each node.

Install Docker and Kubernetes on All Servers

Most of these commands need to be run on each of the nodes. Pay attention though. Down at Step 10, we are going to do a little bit on just the master, and down at Step 15 we'll run something on just the nodes. There are notes down there, just be watching for them.

1 - Once we have logged in, we need to elevate privileges using sudo:

```
sudo su
```

2 - Disable SELinux:

```
setenforce 0
```

```
sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
```

3 - Enable the br_netfilter module for cluster communication:

```
modprobe br_netfilter
```

```
echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

4 - Ensure that the Docker dependencies are satisfied:

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

5 - Add the Docker repo and install Docker:

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
yum install -y docker-ce
```

6 - Set the cgroup driver for Docker to systemd, reload systemd, then enable and start Docker:

```
sed -i '/^ExecStart/ s/$/ --exec-opt native.cgroupdriver=systemd/'
```

```
/usr/lib/systemd/system/docker.service
```

```
systemctl daemon-reload
```

```
systemctl enable docker --now
```

7 - Add the Kubernetes repo:

```
cat << EOF > /etc/yum.repos.d/kubernetes.repo
```

```
[kubernetes]
```

```
name=Kubernetes
```

```
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64
```

```
enabled=1
```

```
gpgcheck=0
```

```
repo_gpgcheck=0
```

```
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
```

```
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
```

EOF

8 - Install Kubernetes v1.14.0:

```
yum install -y kubelet-1.14.0-0 kubeadm-1.14.0-0 kubectl-1.14.0-0 kubernetes-cni-0.7.5
```

9 - Enable the kubelet service. The kubelet service will fail to start until the cluster is initialized, this is expected:

```
systemctl enable kubelet
```

Note: Complete the following section on the **MASTER ONLY!**

10 - Initialize the cluster using the IP range for Flannel:

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

11 - Copy the kubeadmn join command that is in the output. We will need this later.

NOTE: see **Addendum** for an example of the whole output from using the **kubeadm init** command.

12 - Exit sudo, copy the admin.conf to your home directory, and take ownership.

```
mkdir -p $HOME/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

13 - Deploy Flannel:

```
kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel-old.yaml
```

14 - Check the cluster state:

```
kubectl get pods --all-namespaces
```

Note: Complete the following steps on the **NODES ONLY!**

15 - Run the join command that you copied earlier, this requires running the command prefaced with sudo on the nodes (if we hadn't run sudo su to begin with).

Then we'll check the nodes from the **master**.

```
kubectl get nodes
```

Create and Scale a Deployment Using kubectl

Note: These commands will only be run on the **master node**.

16 - Create a simple deployment:

```
kubectl create deployment nginx --image=nginx
```

17 - Inspect the pod:

```
kubectl get pods
```

18 - Scale the deployment:

```
kubectl scale deployment nginx --replicas=4
```

19 - Inspect the pods. We should have four now:

```
kubectl get pods
```

Configure permissions for Docker on both Master and node/s

```
sudo chmod 666 /var/run/docker.sock
```

```
docker run hello-world
```

Also install the following on both Master and node/s

Install the Git package:

```
sudo yum install git -y
```

Install Python3 on both Master and node/s:

NOTE: It is important to ensure that both **PIP** and **botocore boto3** libraries are installed on this system.
Please follow these steps:

```
sudo yum update
```

```
sudo yum install python3
```

```
sudo yum install python3-pip
```

```
pip install botocore boto3
```

```
python3 –version
```

Install Java 11 on both Master and node/s

```
sudo apt install default-jdk -y
```

Please, ensure that the above ports are open externally and between each other.

Log into the monitoring server

```
ssh cloud_user@<MONITORING_SERVER_PUBLIC_IP_ADDRESS>
```

Note: When copying and pasting code into Vim as you will later, first enter

:set paste (and then i to enter insert mode) to avoid adding unnecessary spaces and hashes.

Set Up Prometheus

On the monitoring server, create a user for Prometheus:

```
sudo useradd --no-create-home --shell /bin/false prometheus
```

```
cloud_user@mon:~$ sudo useradd --no-create-home --shell /bin/false prometheus  
[sudo] password for cloud_user:
```

Create the needed directories:

```
sudo mkdir /etc/prometheus  
sudo mkdir /var/lib/Prometheus
```

```
cloud_user@mon:~$ sudo mkdir /etc/prometheus  
cloud_user@mon:~$ sudo mkdir /var/lib/prometheus
```

Set the ownership of the /var/lib/prometheus directory:

```
sudo chown prometheus:prometheus /var/lib/Prometheus
```

```
cloud_user@mon:~$ sudo chown prometheus:prometheus /var/lib/prometheus
```

Download Prometheus:

```
cd /tmp/  
wget https://github.com/prometheus/prometheus/releases/download/v2.7.1/prometheus-2.7.1.linux-amd64.tar.gz
```

2.7.1.linux-amd64.tar.gz

```
cloud_user@mon:/tmp$ wget https://github.com/prometheus/prometheus/releases/download/v2.7.1/prometheus-2.7.1.linux-amd64.tar.gz  
--2023-09-26 07:58:37-- https://github.com/prometheus/prometheus/releases/download/v2.7.1/prometheus-2.7.1.linux-amd64.tar.gz  
Resolving github.com (github.com)... 140.82.113.3  
Connecting to github.com (github.com)|140.82.113.3|:443... connected.  
HTTP request sent, awaiting response... 302 Found  
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/926b3400-2563-11e9-8088-63bcd1a7b246?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53AX2F20230926X2fus-east-1%2Fs3%2Faws4\_request&X-Amz-Date=20230926T075838Z&X-Amz-Expires=300&X-Amz-Signature=id39eda3807ab9947a7b0249eea36747ce68187a46c14988f18337b500d87edb8&X-Amz-SignedHeaders=host&actor\_id=0&key\_id=0&repo\_id=6838921&response-content-disposition=attachment%3B%20filename%3Dprometheus-2.7.1.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream  
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.109.133, 185.199.110.133, 185.199.108.133, ...  
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.109.133|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 38342763 (37M) [application/octet-stream]  
Saving to: 'prometheus-2.7.1.linux-amd64.tar.gz'  
  
prometheus-2.7.1.linux-amd64.tar.gz 100%[=====] 36.57M 181MB/s in 0.2s  
2023-09-26 07:58:37 (181 MB/s) - 'prometheus-2.7.1.linux-amd64.tar.gz' saved [38342763/38342763]  
cloud_user@mon:/tmp$
```

Extract the files:

```
tar -xvf prometheus-2.7.1.linux-amd64.tar.gz
```

```
cloud_user@mon:~$ ls
cloud_user@mon:~$ cd /tmp/
cloud_user@mon:/tmp$ ls
prometheus-2.7.1.linux-amd64.tar.gz  systemd-private-0e2b4d0243044376a327d441faa9ad51-systemd-resolved.service-Crrxcz
snap-private-tmp                      systemd-private-0e2b4d0243044376a327d441faa9ad51-systemd-timesyncd.service-kQt5eu
cloud_user@mon:/tmp$ tar -xvf prometheus-2.7.1.linux-amd64.tar.gz
```

```
cloud_user@mon:~$ ls
cloud_user@mon:~$ cd /tmp/
cloud_user@mon:/tmp$ ls
prometheus-2.7.1.linux-amd64.tar.gz  systemd-private-0e2b4d0243044376a327d441faa9ad51-systemd-resolved.service-Crrxcz
snap-private-tmp                      systemd-private-0e2b4d0243044376a327d441faa9ad51-systemd-timesyncd.service-kQt5eu
cloud_user@mon:/tmp$ tar -xvf prometheus-2.7.1.linux-amd64.tar.gz
prometheus-2.7.1.linux-amd64/
prometheus-2.7.1.linux-amd64/LICENSE
prometheus-2.7.1.linux-amd64/prometheus.yml
prometheus-2.7.1.linux-amd64/consoles/
prometheus-2.7.1.linux-amd64/consoles/node-cpu.html
prometheus-2.7.1.linux-amd64/consoles/node.html
prometheus-2.7.1.linux-amd64/consoles/prometheus.html
prometheus-2.7.1.linux-amd64/consoles/prometheus-overview.html
prometheus-2.7.1.linux-amd64/consoles/node-overview.html
prometheus-2.7.1.linux-amd64/consoles/node-disk.html
prometheus-2.7.1.linux-amd64/consoles/index.html.example
prometheus-2.7.1.linux-amd64/console_libraries/
prometheus-2.7.1.linux-amd64/console_libraries/prom.lib
prometheus-2.7.1.linux-amd64/console_libraries/menu.lib
prometheus-2.7.1.linux-amd64/NOTICE
prometheus-2.7.1.linux-amd64/promtool
prometheus-2.7.1.linux-amd64/prometheus
cloud_user@mon:/tmp$
```

Move the configuration file and set the owner to the prometheus user:

```
cd prometheus-2.7.1.linux-amd64/
sudo mv console* /etc/prometheus
sudo mv prometheus.yml /etc/prometheus
sudo chown -R prometheus:prometheus /etc/prometheus
```

```
cloud_user@mon:/tmp$ cd prometheus-2.7.1.linux-amd64/
cloud_user@mon:/tmp/prometheus-2.7.1.linux-amd64$ sudo mv console* /etc/prometheus
[sudo] password for cloud_user:
cloud_user@mon:/tmp/prometheus-2.7.1.linux-amd64$ sudo mv prometheus.yml /etc/prometheus
cloud_user@mon:/tmp/prometheus-2.7.1.linux-amd64$ sudo chown -R prometheus:prometheus /etc/prometheus
cloud_user@mon:/tmp/prometheus-2.7.1.linux-amd64$
```

Move the binaries and set the owner:

```
sudo mv prometheus /usr/local/bin/
sudo mv promtool /usr/local/bin/
sudo chown prometheus:prometheus /usr/local/bin/prometheus
```

```
sudo chown prometheus:prometheus /usr/local/bin/promtool
```

Change back to your main directory, and then create the service file:

```
cloud_user@mon:/tmp/prometheus-2.7.1.linux-amd64$ sudo mv prometheus /usr/local/bin/
cloud_user@mon:/tmp/prometheus-2.7.1.linux-amd64$ sudo mv promtool /usr/local/bin/
cloud_user@mon:/tmp/prometheus-2.7.1.linux-amd64$ sudo chown prometheus:prometheus /usr/local/bin/prometheus
cloud_user@mon:/tmp/prometheus-2.7.1.linux-amd64$ sudo chown prometheus:prometheus /usr/local/bin/promtool
cloud_user@mon:/tmp/prometheus-2.7.1.linux-amd64$ █
```

```
cd
```

```
sudo vim /etc/systemd/system/prometheus.service
```

```
cloud_user@mon:/tmp/prometheus-2.7.1.linux-amd64$ cd
cloud_user@mon:~$ sudo vim /etc/systemd/system/prometheus.service
cloud_user@mon:~$ █
```

Add the following to the file:

```
[Unit]
```

```
Description=Prometheus
Wants=network-online.target
After=network-online.target
```

```
[Service]
```

```
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
    --config.file /etc/prometheus/prometheus.yml \
    --storage.tsdb.path /var/lib/prometheus/ \
    --web.console.templates=/etc/prometheus/consoles \
    --web.console.libraries=/etc/prometheus/console_libraries
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Save and exit the file by pressing Escape followed by :wq.

```

[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
--config.file /etc/prometheus/prometheus.yml \
--storage.tsdb.path /var/lib/prometheus/ \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target
~
~

```

Start Prometheus and make sure it automatically starts on boot:

```

sudo systemctl start prometheus
sudo systemctl enable Prometheus

```

```

cloud_user@mon:~$ sudo systemctl start prometheus
cloud_user@mon:~$ sudo systemctl enable prometheus
Created symlink /etc/systemd/system/multi-user.target.wants/prometheus.service → /etc/systemd/system/prometheus.service.
cloud_user@mon:~$ 

```

Check Prometheus's status to ensure everything is working correctly:

```

sudo systemctl status prometheus

```

```

cloud_user@mon:~$ sudo systemctl status prometheus
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-09-26 08:16:54 UTC; 1min 5s ago
       Main PID: 31758 (prometheus)
          Tasks: 9 (limit: 1104)
         CGroup: /system.slice/prometheus.service
             └─ 31758 /usr/local/bin/prometheus --config.file /etc/prometheus/prometheus.yml --storage.tsdb.path /var/lib/prometheus/ --web.console.templates=/etc/pr

Sep 26 08:16:55 mon prometheus[31758]: level=info ts=2023-09-26T08:16:55.052864138Z caller=main.go:303 build_context="(go=go1.11.5, user=root@f9f82868fc43, date=2023-09-26T08:16:55+00:00)"
Sep 26 08:16:55 mon prometheus[31758]: level=info ts=2023-09-26T08:16:55.05292744Z caller=main.go:304 host_details="(Linux 5.4.0-1092-aws #100-18.04.2-Ubuntu SMP PREEMPT_DYNAMIC 5.4.0-1092-aws 18.04.2-Ubuntu 5.4.0-1092-aws #100-18.04.2-Ubuntu SMP PREEMPT_DYNAMIC 2023-09-26T08:16:55+00:00)"
Sep 26 08:16:55 mon prometheus[31758]: level=info ts=2023-09-26T08:16:55.05296933Z caller=main.go:305 fd_limits="(soft=1024, hard=4996)"
Sep 26 08:16:55 mon prometheus[31758]: level=info ts=2023-09-26T08:16:55.05300037Z caller=main.go:306 vm_limits="(soft=unlimited, hard=unlimited)"
Sep 26 08:16:55 mon prometheus[31758]: level=info ts=2023-09-26T08:16:55.05456719Z caller=web.go:416 component=web msg="Start listening for connections" address="0.0.0.0:9090"
Sep 26 08:16:55 mon prometheus[31758]: level=info ts=2023-09-26T08:16:55.05451770Z caller=main.go:620 msg="Starting TSDB ..."
Sep 26 08:16:55 mon prometheus[31758]: level=info ts=2023-09-26T08:16:55.08004556Z caller=main.go:635 msg="TSDB started"
Sep 26 08:16:55 mon prometheus[31758]: level=info ts=2023-09-26T08:16:55.08013416Z caller=main.go:695 msg="Loading configuration file" filename=/etc/prometheus/prometheus.yml
Sep 26 08:16:55 mon prometheus[31758]: level=info ts=2023-09-26T08:16:55.08121309Z caller=main.go:722 msg="Completed loading of configuration file" filename=/etc/prometheus/prometheus.yml
Sep 26 08:16:55 mon prometheus[31758]: level=info ts=2023-09-26T08:16:55.08124299Z caller=main.go:589 msg="Server is ready to receive web requests."
Lines 1-18 (END)

```

Set Up Alertmanager:

Still in the monitoring server, create the alertmanager system user:

```
sudo useradd --no-create-home --shell /bin/false alertmanager
```

Create the needed directories:

```
sudo mkdir /etc/alertmanager
```

```
cloud_user@mon:~$ sudo useradd --no-create-home --shell /bin/false alertmanager
cloud_user@mon:~$ sudo mkdir /etc/alertmanager
cloud_user@mon:~$ █
```

Download Alertmanager:

```
cd /tmp/
wget "https://github.com/prometheus/alertmanager/releases/download/v0.16.1/alertmanager-0.16.1.linux-amd64.tar.gz"
```

```
cloud_user@mon:~$ cd /tmp/
cloud_user@mon:/tmp$ wget "https://github.com/prometheus/alertmanager/releases/download/v0.16.1/alertmanager-0.16.1.linux-amd64.tar.gz"
--2023-09-26 08:23:36-- https://github.com/prometheus/alertmanager/releases/download/v0.16.1/alertmanager-0.16.1.linux-amd64.tar.gz
Resolving github.com (github.com)... 140.82.113.4
Connecting to github.com (github.com)|140.82.113.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/11452538/5abdb580-257e-11e9-8be2-57c655a5e02d?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAINWJYAX4CSVEH53A%2F20230926%2Fus-east-1%2Fs%2Faws4_request&X-Amz-Date=20230926T082337Z&X-Amz-Expires=300&X-Amz-Signature=db8f876fa7cc7a10f5e6fd09cf2e2b998b56900cd121658712eac2a542ece47c1&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=11452538&response-content-disposition=attachment%20filename%20alertmanager-0.16.1.linux-amd64.tar.gz&response-content-type=application%20octet-stream [following]
--2023-09-26 08:23:36-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/11452538/5abdb580-257e-11e9-8be2-57c655a5e02d?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAINWJYAX4CSVEH53A%2F20230926%2Fus-east-1%2Fs%2Faws4_request&X-Amz-Date=20230926T082337Z&X-Amz-Expires=300&X-Amz-Signature=db8f876fe7cc7a10f5e6fd09cf2e2b998b56900cd121658712eac2a542ece47c1&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=11452538&response-content-disposition=attachment%20filename%20alertmanager-0.16.1.linux-amd64.tar.gz&response-content-type=application%20octet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19070056 (18M) [application/octet-stream]
Saving to: 'alertmanager-0.16.1.linux-amd64.tar.gz'

alertmanager-0.16.1.linux-amd64.tar.gz 100%[=====] 18.19M ---KB/s in 0.1s
2023-09-26 08:23:36 (169 MB/s) - 'alertmanager-0.16.1.linux-amd64.tar.gz' saved [19070056/19070056]
cloud_user@mon:/tmp$ █
```

Extract the files:

```
tar -xvf alertmanager-0.16.1.linux-amd64.tar.gz
```

```
cloud_user@mon:/tmp$ tar -xvf alertmanager-0.16.1.linux-amd64.tar.gz █
```

```
cloud_user@mon:/tmp$ tar -xvf alertmanager-0.16.1.linux-amd64.tar.gz
alertmanager-0.16.1.linux-amd64/
alertmanager-0.16.1.linux-amd64/NOTICE
alertmanager-0.16.1.linux-amd64/amtool
alertmanager-0.16.1.linux-amd64/LICENSE
alertmanager-0.16.1.linux-amd64/alertmanager
alertmanager-0.16.1.linux-amd64/alertmanager.yml
cloud_user@mon:/tmp$
```

`cd alertmanager-0.16.1.linux-amd64/`

```
cloud_user@mon:/tmp$ cd alertmanager-0.16.1.linux-amd64/
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$
```

Move the binaries:

```
sudo mv alertmanager /usr/local/bin/
sudo mv amtool /usr/local/bin/
```

```
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$ sudo mv alertmanager /usr/local/bin/
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$ sudo mv amtool /usr/local/bin/
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$
```

Set the ownership of the binaries:

```
sudo chown -R alertmanager:alertmanager /usr/local/bin/alertmanager
sudo chown -R alertmanager:alertmanager /usr/local/bin/amtool
```

```
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$ sudo chown -R alertmanager:alertmanager /usr/local/bin/alertmanager
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$ sudo chown -R alertmanager:alertmanager /usr/local/bin/amtool
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$
```

Move the configuration file into the /etc/alertmanager directory:

```
sudo mv alertmanager.yml /etc/alertmanager/
```

```
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$ sudo mv alertmanager.yml /etc/alertmanager/
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$
```

Set the ownership of the /etc/alertmanager directory:

sudo chown -R alertmanager:alertmanager /etc/alertmanager/

```
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$ sudo chown -R alertmanager:alertmanager /etc/alertmanager/
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$
```

Create the alertmanager.service file for systemd:

sudo vim /etc/systemd/system/alertmanager.service

Add the following to the file:

```
[Unit]
Description=Alertmanager
Wants=network-online.target
After=network-online.target
```

```
[Service]
User=alertmanager
Group=alertmanager
Type=simple
WorkingDirectory=/etc/alertmanager/
ExecStart=/usr/local/bin/alertmanager \
--config.file=/etc/alertmanager/alertmanager.yml
```

```
[Install]
WantedBy=multi-user.target
```

Save and exit the file by pressing Escape followed by :wq.

```
[Unit]
Description=Alertmanager
Wants=network-online.target
After=network-online.target

[Service]
User=alertmanager
Group=alertmanager
Type=simple
WorkingDirectory=/etc/alertmanager/
ExecStart=/usr/local/bin/alertmanager \
--config.file=/etc/alertmanager/alertmanager.yml

[Install]
WantedBy=multi-user.target
~
~
~
```

Stop Prometheus, change back to your main directory, and then update the Prometheus configuration file to use Alertmanager:

```
sudo systemctl stop prometheus
cd
sudo vim /etc/prometheus/prometheus.yml
```

```
cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$ sudo systemctl stop prometheus
:cloud_user@mon:/tmp/alertmanager-0.16.1.linux-amd64$ cd
cloud_user@mon:~$ sudo vim /etc/prometheus/prometheus.yml
cloud_user@mon:~$ █
```

Un-comment alertmanager, and change alertmanager to localhost (so it will look like the following):

```
alerting:
  alertmanagers:
    - static_configs:
      - targets:
        - localhost:9093
```

```
# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
      - targets:
          # - alertmanager:9093
```

```
# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
      - targets:
          - alertmanager:9093
```

Save and exit the file by pressing Escape followed by :wq.

Reload systemd, and then start the prometheus and alertmanager services:

```
sudo systemctl daemon-reload
sudo systemctl start prometheus
sudo systemctl start alertmanager
```

```
cloud_user@mon:~$ sudo systemctl daemon-reload
cloud_user@mon:~$ sudo systemctl start prometheus
cloud_user@mon:~$ sudo systemctl start alertmanager
cloud_user@mon:~$ █
```

Make sure alertmanager starts on boot:

```
sudo systemctl enable alertmanager
```

```
cloud_user@mon:~$ sudo systemctl enable alertmanager
Created symlink /etc/systemd/system/multi-user.target.wants/alertmanager.service → /etc/systemd/system/alertmanager.service.
cloud_user@mon:~$ █
```

Set Up Grafana

In a new terminal window, log in to the grafana server via SSH using the credentials provided on the lab page:

```
ssh cloud_user@<GRAFANA_SERVER_PUBLIC_IP_ADDRESS>
```

Install the prerequisite package:

```
sudo apt-get install libfontconfig
```

Y

```
cloud_user@grafana:~$ sudo apt-get install libfontconfig
[sudo] password for cloud_user:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libfontconfig1' instead of 'libfontconfig'
The following packages were automatically installed and are no longer required:
  linux-aws-5.4.0-1060 linux-headers-5.4.0-1060-aws linux-image-5.4.0-1060-aws linux-modules-5.4.0-1060-aws
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core
The following NEW packages will be installed:
  fontconfig-config fonts-dejavu-core libfontconfig1
0 upgraded, 3 newly installed, 0 to remove and 26 not upgraded.
Need to get 1233 kB of archives.
After this operation, 4015 kB of additional disk space will be used.
Do you want to continue? [Y/n] ■
```

Download and install Grafana using the .deb package provided on the Grafana download page:

```
cd /tmp/
wget https://dl.grafana.com/oss/release/grafana\_5.4.3\_amd64.deb
```

```
cloud_user@grafana:~$ cd /tmp/
[cloud_user@grafana:tmp]$ wget https://dl.grafana.com/oss/release/grafana_5.4.3_amd64.deb
--2023-09-26 08:45:29-- https://dl.grafana.com/oss/release/grafana_5.4.3_amd64.deb
Resolving dl.grafana.com (dl.grafana.com)... 146.75.38.217, 2a04:4e42:78::729
Connecting to dl.grafana.com (dl.grafana.com)|146.75.38.217|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 55405690 (53M) [application/x-debian-package]
Saving to: 'grafana_5.4.3_amd64.deb'

grafana_5.4.3_amd64.deb          100%[=====] 52.84M  166MB/s   in 0.3s

2023-09-26 08:45:30 (166 MB/s) - 'grafana_5.4.3_amd64.deb' saved [55405690/55405690]
cloud_user@grafana:/tmp$ sudo dpkg -i grafana_5.4.3_amd64.deb■
```

```
sudo dpkg -i grafana_5.4.3_amd64.deb
```

```
cloud_user@grafana:/tmp$ sudo dpkg -i grafana_5.4.3_amd64.deb
Selecting previously unselected package grafana.
(Reading database ... 149179 files and directories currently installed.)
Preparing to unpack grafana_5.4.3_amd64.deb ...
Unpacking grafana (5.4.3) ...
Setting up grafana (5.4.3) ...
Adding system user `grafana' (UID 111) ...
Adding new user `grafana' (UID 111) with group `grafana' ...
Not creating home directory `/usr/share/grafana'.
### NOT starting on installation, please execute the following statements to configure grafana to start automatically using systemd
  sudo /bin/systemctl daemon-reload
  sudo /bin/systemctl enable grafana-server
### You can start grafana-server by executing
  sudo /bin/systemctl start grafana-server
Processing triggers for systemd (237-3ubuntu10.57) ...
Processing triggers for ureadahead (0.100.0-21) ...
cloud_user@grafana:/tmp$ ■
```

Start Grafana:

sudo systemctl start grafana-server

Ensure Grafana starts at boot:

sudo systemctl enable grafana-server

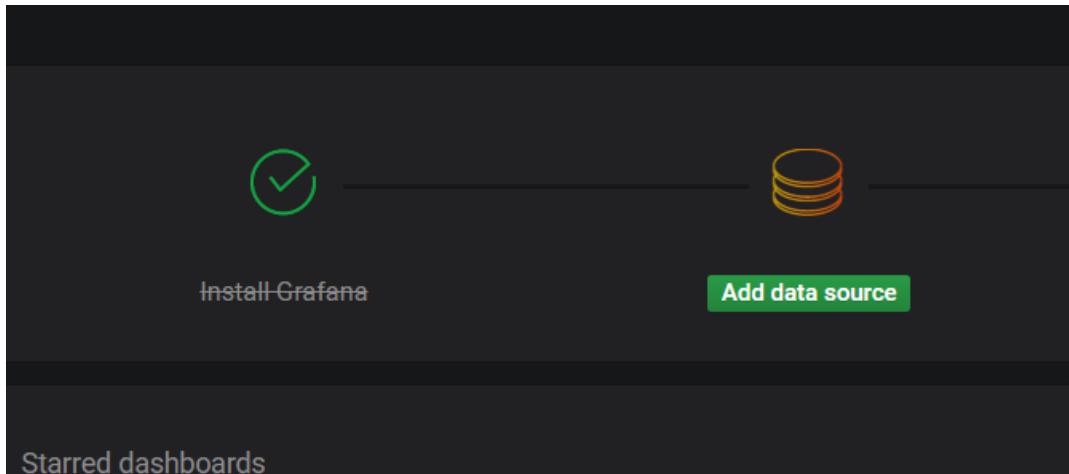
```
cloud_user@grafana:/tmp$ sudo systemctl start grafana-server
cloud_user@grafana:/tmp$ sudo systemctl enable grafana-server
Synchronizing state of grafana-server.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.service → /usr/lib/systemd/system/grafana-server.service.
cloud_user@grafana:/tmp$
```

Using the Grafana IP address listed on the lab page, access Grafana's web UI by navigating to **<GRAFANA_IP_ADDRESS>:3000** in a new browser tab.



Log in with the username **admin** and the password **admin**. Reset the password when prompted.

Add a Data Source



Click **Add data source** on the home page.

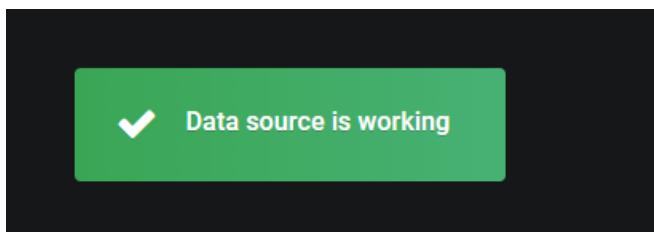
The image shows the Grafana Configuration page. At the top left is a gear icon followed by the word 'Configuration'. Below it says 'Organization: Main Org.'. The top navigation bar includes tabs for 'Data Sources' (which is red), 'Users', 'Teams', 'Plugins', 'Preferences', and 'API Keys'. The main content area is titled 'Choose data source type' and features a search bar with a magnifying glass icon and the placeholder 'Filter by name or type'. Below the search bar is a 3x3 grid of data source icons and names: CloudWatch, Elasticsearch, Grafana Logging; Graphite, InfluxDB, Microsoft SQL Server; MySQL, OpenTSDB, PostgreSQL; Prometheus, Stackdriver, TestData DB.

Select **Prometheus**.

In the HTTP section, set the URL to http://<MONITORING_IP_ADDRESS>:9090 (using the public IP of the monitoring server provided on the lab page).

The screenshot shows the 'Settings' page for a Prometheus data source in Grafana. On the left is a sidebar with icons for Home, Create, Dashboard, Alerting, Metrics, and Settings. The main area has a dark background with light-colored input fields. At the top, there's a row for 'Name' (Prometheus), 'Default' (checkbox checked), and 'Help'. Below that is the 'HTTP' section with fields for 'URL' (http://34.227.148.26:9090), 'Access' (Server (Default)), and 'Whitelisted Cookies' (Add Name). Under 'Auth', there are three options: 'Basic Auth' (unchecked), 'With Credentials' (unchecked), 'TLS Client Auth' (unchecked), 'With CA Cert' (unchecked), and 'Skip TLS Verify' (unchecked). At the bottom of the configuration are fields for 'Scrape interval' (15s), 'Query timeout' (60s), and 'HTTP Method' (GET). At the very bottom are three buttons: 'Save & Test' (green), 'Delete' (red), and 'Back'.

Click **Save & Test**.



Add Endpoints

Back in the **monitoring server** terminal, open the Prometheus configuration file:

```
sudo vim /etc/prometheus/prometheus.yml
```

At the end of the file, at the bottom of the scrape_configs section, add the Alertmanager endpoint (make sure it aligns with the - job_name: 'prometheus' line):

```
- job_name: 'alertmanager'  
  static_configs:
```

```
- targets: ['localhost:9093']
```

Beneath what you just added, add the Grafana endpoint (using the public IP address of the Grafana server):

```
- job_name: 'grafana'  
  static_configs:  
    - targets: ['<GRAFANA_IP_ADDRESS>:3000']
```

Save and exit the file by pressing Escape followed by :wq.

```
# The job name is added as a label `job=<  
- job_name: 'prometheus'  
  
  # metrics_path defaults to '/metrics'  
  # scheme defaults to 'http'.  
  
  static_configs:  
    - targets: ['localhost:9090']  
- job_name: 'alertmanager'  
  static_configs:  
    - targets: ['localhost:9093']  
- job_name: 'grafana'  
  static_configs:  
    - targets: ['44.208.31.119:3000']
```

Restart Prometheus:

```
sudo systemctl restart Prometheus
```

Check its status:

```
sudo systemctl status prometheus
```

```

cloud_user@mon:~$ sudo systemctl restart prometheus
cloud_user@mon:~$ sudo systemctl status prometheus
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-09-26 09:12:31 UTC; 22s ago
       Main PID: 32261 (prometheus)
         Tasks: 10 (limit: 1104)
        CGroup: /system.slice/prometheus.service
                  └─32261 /usr/local/bin/prometheus --config.file /etc/prometheus/prometheus.yml --storage.tsdb.path /var/lib/prometheus/ --web.console.templates=/etc/prome

Sep 26 09:12:31 mon prometheus[32261]: level=info ts=2023-09-26T09:12:31.631910232Z caller=main.go:304 host_details="(Linux 5.4.0-1092-aws #100~18.04.2-Ubuntu S
Sep 26 09:12:31 mon prometheus[32261]: level=info ts=2023-09-26T09:12:31.631952175Z caller=main.go:305 fd_limits="(soft=1024, hard=4096)"
Sep 26 09:12:31 mon prometheus[32261]: level=info ts=2023-09-26T09:12:31.631983667Z caller=main.go:306 vm_limits="(soft=unlimited, hard=unlimited)"
Sep 26 09:12:31 mon prometheus[32261]: level=info ts=2023-09-26T09:12:31.636602338Z caller=main.go:620 msg="Starting TSDB ..."
Sep 26 09:12:31 mon prometheus[32261]: level=info ts=2023-09-26T09:12:31.636671328Z caller=web.go:416 component=web msg="Start listening for connections" address=":9090"
Sep 26 09:12:31 mon prometheus[32261]: level=info ts=2023-09-26T09:12:31.661077185Z caller=main.go:635 msg="TSDB started"
Sep 26 09:12:31 mon prometheus[32261]: level=info ts=2023-09-26T09:12:31.661155465Z caller=main.go:695 msg="Loading configuration file" filename=/etc/prometheus
Sep 26 09:12:31 mon prometheus[32261]: level=info ts=2023-09-26T09:12:31.6621346Z caller=main.go:722 msg="Completed loading of configuration file" filename=/etc/prome
Sep 26 09:12:31 mon prometheus[32261]: level=info ts=2023-09-26T09:12:31.662173397Z caller=main.go:589 msg="Server is ready to receive web requests."
Sep 26 09:12:36 mon prometheus[32261]: level=error ts=2023-09-26T09:12:36.640107365Z caller=notifier.go:481 component=notifier alertmanager=http://alertmanager:9093
lines 1-18 (END)

```

Using the public IP address of the monitoring server, navigate to the Prometheus web UI in a new browser tab:

http://<MONITORING_IP_ADDRESS>:9090

The screenshot shows the Prometheus web interface. At the top, there is a navigation bar with links for 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below the navigation bar, there is a section for 'Enable query history' with a checkbox. A large input field labeled 'Expression (press Shift+Enter for newlines)' is present, with a dropdown menu showing '- insert metric at cursor -'. Below the input field, there are two tabs: 'Graph' (which is selected) and 'Console'. Under the 'Graph' tab, there is a time range selector with arrows and the word 'Moment'. Below the time range, there is a section for 'Element' with the message 'no data'. At the bottom left, there is a blue button labeled 'Add Graph'.

Click **Status > Targets**.

Prometheus Alerts Graph Status ▾ Help

Targets

All Unhealthy

alertmanager (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9093/metrics	UP	instance="localhost:9093" job="alertmanager"	10.458s ago	3.023ms	

grafana (1/1 up) show less

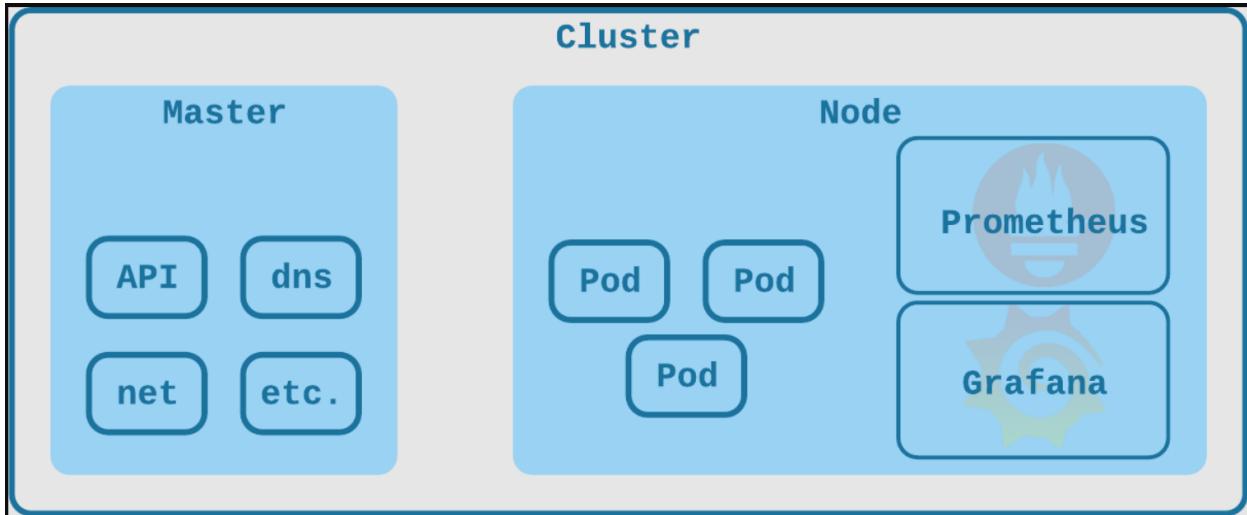
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://44.208.31.119:3000/metrics	UP	instance="44.208.31.119:3000" job="grafana"	7.493s ago	5.42ms	

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	3.753s ago	4.961ms	

Ensure all three endpoints are listed on the **Targets** page.

Monitoring in Kubernetes with Prometheus and Grafana



Creating a Kubernetes Cluster

Introduction

We will install and configure a Kubernetes cluster consisting of 1 master and 1 or 2 nodes. Once the installation and configuration are complete, we will have a 2-node Kubernetes cluster that uses Flannel as the network overlay.

We will be using Centos 7.x for all virtual machines.

Logging In

NOTE: It's probably a good idea to have two or three terminals open, one for each node.

Install Docker and Kubernetes on All Servers

Most of these commands need to be run on each of the nodes. Pay attention though. Down at Step 10, we are going to do a little bit on just the master, and down at Step 15 we'll run something on just the nodes. There are notes down there, just be watching for them.

1 - Once we have logged in, we need to elevate privileges using sudo:

```
sudo su
```

2 - Disable SELinux:

```
setenforce 0
```

```
sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
```

3 - Enable the br_netfilter module for cluster communication:

```
modprobe br_netfilter
```

```
echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

4 - Ensure that the Docker dependencies are satisfied:

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

5 - Add the Docker repo and install Docker:

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
yum install -y docker-ce
```

6 - Set the cgroup driver for Docker to systemd, reload systemd, then enable and start Docker:

```
sed -i '/^ExecStart/ s/$/ --exec-opt native.cgroupdriver=systemd/'
/usr/lib/systemd/system/docker.service
systemctl daemon-reload
systemctl enable docker --now
```

7 - Add the Kubernetes repo:

```
cat << EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

8 - Install Kubernetes v1.14.0:

```
yum install -y kubelet-1.14.0-0 kubeadm-1.14.0-0 kubectl-1.14.0-0 kubernetes-cni-0.7.5
```

9 - Enable the kubelet service. The kubelet service will fail to start until the cluster is initialized, this is expected:

```
systemctl enable kubelet
```

Note: Complete the following section on the **MASTER ONLY!**

10 - Initialize the cluster using the IP range for Flannel:

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

11 - Copy the kubeadm join command that is in the output. We will need this later.

NOTE: see **Addendum** for an example of the whole output from using the **kubeadm init** command.

12 - Exit sudo, copy the admin.conf to your home directory, and take ownership.

```
mkdir -p $HOME/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

13 - Deploy Flannel:

```
kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel-old.yaml
```

14 - Check the cluster state:

```
kubectl get pods --all-namespaces
```

Note: Complete the following steps on the **NODES ONLY!**

15 - Run the join command that you copied earlier, this requires running the command prefaced with sudo on the nodes (if we hadn't run sudo su to begin with).

Then we'll check the nodes from the **master**.

kubectl get nodes

Create and Scale a Deployment Using kubectl

Note: These commands will only be run on the **master node**.

16 - Create a simple deployment:

kubectl create deployment nginx --image=nginx

17 - Inspect the pod:

kubectl get pods

18 - Scale the deployment:

kubectl scale deployment nginx --replicas=4

19 - Inspect the pods. We should have four now:

kubectl get pods

Configure permissions for Docker on both Master and node/s

sudo chmod 666 /var/run/docker.sock

docker run hello-world

Also install the following on both Master and node/s

Install the Git package:

sudo yum install git -y

Install Python3 on both Master and node/s:

NOTE: It is important to ensure that both **PIP** and **botocore boto3** libraries are installed on this system.

Please follow these steps:

sudo yum update

sudo yum install python3

sudo yum install python3-pip

pip install botocore boto3

python3 –version

Install Java 11 on both Master and node/s

```
sudo apt install default-jdk -y
```

Install components to use Gradle on both Master and node/s

```
sudo yum update
```

```
curl -o https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash  
nvm install node
```

A task to perform on the Node

```
git clone https://github.com/ACloudGuru-Resources/cicd-pipeline-train-schedule-pipelines.git
```

```
cd
```

```
./gradlew build
```

```
./gradlew npm_start
```

Install Helm v2.8.2 on the Master

Helm is a package manager for Kubernetes!

Helm 3 can be installed many ways. We will install Helm 2.8.2 using the scripts option.

1.Download the script:

Download Helm v2.8.2 for your operating system from the Helm GitHub releases page.

Replace the URL with the appropriate one for your OS:

```
wget https://get.helm.sh/helm-v2.8.2-linux-amd64.tar.gz  
tar -zxf helm-v2.8.2-linux-amd64.tar.gz
```

If you're using a different OS, adjust the URL accordingly.

2.Move Helm Binary to a Directory in Your PATH:

```
sudo mv linux-amd64/helm /usr/local/bin/
```

Move the Helm binary to a directory in your PATH so you can use it conveniently.

Initialize Helm with Tiller and the Specific Image:

```
helm init --stable-repo-url=https://charts.helm.sh/stable --wait --tiller-image  
ghcr.io/helm/tiller:v2.8.2
```

```
cloud_user@ip-10-0-1-101:~$ helm init --stable-repo-url=https://charts.helm.sh/stable --wait --tiller-image ghcr.io/helm/tiller:v2.8.2
Creating /home/cloud_user/.helm
Creating /home/cloud_user/.helm/repository
Creating /home/cloud_user/.helm/repository/cache
Creating /home/cloud_user/.helm/repository/local
Creating /home/cloud_user/.helm/plugins
Creating /home/cloud_user/.helm/starters
Creating /home/cloud_user/.helm/cache/archive
Creating /home/cloud_user/.helm/repository/repositories.yaml
Adding stable repo with URL: https://charts.helm.sh/stable
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /home/cloud_user/.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated users' policy.
For more information on securing your installation see: https://docs.helm.sh/using_helm/#securing-your-helm-installation
Happy Helming!
cloud_user@ip-10-0-1-101:~$
```

Install Prometheus in the Kubernetes Cluster

To do this, make sure you have cloned the Kubernetes charts repo:

```
cd ~/
git clone https://github.com/kubernetes/charts
```

```
cloud_user@ip-10-0-1-101:~$ git clone https://github.com/kubernetes/charts
Cloning into 'charts'...
remote: Enumerating objects: 129385, done.
remote: Total 129385 (delta 0), reused 0 (delta 0), pack-reused 129385
Receiving objects: 100% (129385/129385), 284.57 MiB | 59.74 MiB/s, done.
Resolving deltas: 100% (87191/87191), done.
Checking connectivity... done.
cloud_user@ip-10-0-1-101:~$
```

```
cd charts
git checkout efdfcffe0b6973111ec6e5e83136ea74cdbe6527d
```

```
cloud_user@ip-10-0-1-101:~/charts$ git checkout efdfcffe0b6973111ec6e5e83136ea74cdbe6527d
Note: checking out 'efdfcffe0b6973111ec6e5e83136ea74cdbe6527d'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at efdfcffe... [stable/prometheus] Ability to enable admin API (#5570)
cloud_user@ip-10-0-1-101:~/charts$
```

cd ..

Create a prometheus-values.yml file:

vi prometheus-values.yml

And paste in this content:

```
alertmanager:
  persistentVolume:
    enabled: false
server:
  persistentVolume:
    enabled: false
```

```
alertmanager:
  persistentVolume:
    enabled: false
server:
  persistentVolume:
    enabled: false
~
~
```

Save and close the file:

:wq

Use helm to install Prometheus with prometheus-values.yml:

```
helm install -f ~/prometheus-values.yml ~/charts/stable/prometheus --name prometheus --namespace prometheus
```

```
cloud_user@ip-10-0-1-101:~$ vi prometheus-values.yml
cloud_user@ip-10-0-1-101:~$ helm install -f ~/prometheus-values.yml ~/charts/stable/prometheus --name prometheus --namespace prometheus
```

NOTES:

The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.prometheus.svc.cluster.local

Get the Prometheus server URL by running these commands in the same shell:
export POD_NAME=\$(kubectl get pods --namespace prometheus -l "app=prometheus,component=server" -o jsonpath=".items[0].metadata.name")
kubectl --namespace prometheus port-forward \$POD_NAME 9090

WARNING: Persistence is disabled!!! You will lose your data when #####
the Server pod is terminated. #####
#####

The Prometheus alertmanager can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-alertmanager.prometheus.svc.cluster.local

Get the Alertmanager URL by running these commands in the same shell:
export POD_NAME=\$(kubectl get pods --namespace prometheus -l "app=prometheus,component=alertmanager" -o jsonpath=".items[0].metadata.name")
kubectl --namespace prometheus port-forward \$POD_NAME 9093

WARNING: Persistence is disabled!!! You will lose your data when #####
the AlertManager pod is terminated. #####
#####

The Prometheus PushGateway can be accessed via port 9091 on the following DNS name from within your cluster:
prometheus-pushgateway.prometheus.svc.cluster.local

Get the PushGateway URL by running these commands in the same shell:
export POD_NAME=\$(kubectl get pods --namespace prometheus -l "app=prometheus,component=pushgateway" -o jsonpath=".items[0].metadata.name")
kubectl --namespace prometheus port-forward \$POD_NAME 9091

For more information on running Prometheus, visit:
<https://prometheus.io/>

```
cloud_user@ip-10-0-1-101:~$
```

We can see which pods are running in this new namespace with the command:

kubectl get pods -n prometheus

```
cloud_user@ip-10-0-1-101:~$ kubectl get pods -n prometheus
NAME                               READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-7b47d55bdf-7h4wd   2/2     Running   0          1m
prometheus-kube-state-metrics-6584885ccf-twdb2   1/1     Running   0          1m
prometheus-node-exporter-tk2td                1/1     Running   0          1m
prometheus-pushgateway-66c9fdb48f-d4xch      1/1     Running   0          1m
prometheus-server-65d5cc8544-b224f        2/2     Running   0          1m
cloud_user@ip-10-0-1-101:~$
```

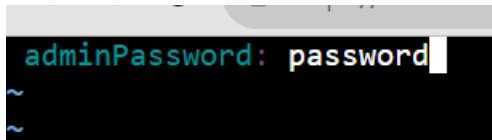
Install Grafana in the Kubernetes Cluster

Create a grafana-values.yml:

```
vi grafana-values.yml
```

Paste in this content (you will use this password to log in to Grafana):

```
adminPassword: password
```



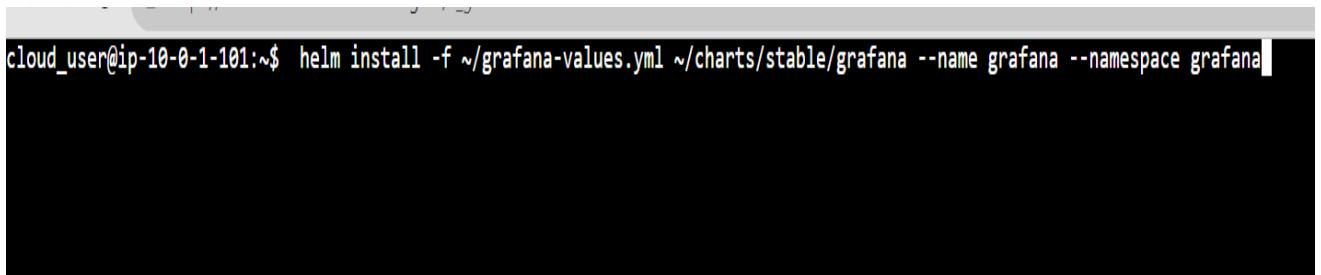
A screenshot of a terminal window. The text "adminPassword: password" is visible in the terminal area. The terminal prompt shows two tilde symbols (~).

Save and close the file:

```
:wq
```

Use helm to install Grafana with grafana-values.yml:

```
helm install -f ~/grafana-values.yml ~/charts/stable/grafana --name grafana --namespace grafana
```



A screenshot of a terminal window. The command "helm install -f ~/grafana-values.yml ~/charts/stable/grafana --name grafana --namespace grafana" is visible in the terminal area. The terminal prompt shows "cloud_user@ip-10-0-1-101:~\$".

```

==> v1/ConfigMap
NAME          DATA AGE
grafana       1    0s
grafana-dashboards-json 0    0s

==> v1/Service
NAME      TYPE      CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
grafana  ClusterIP  10.104.74.80 <none>        80/TCP    0s

==> v1beta2/Deployment
NAME   DESIRED CURRENT UP-TO-DATE AVAILABLE AGE
grafana 1       1       1       0       0s

==> v1/Pod(related)
NAME          READY STATUS      RESTARTS AGE
grafana-d65c89cdc-778vg  0/1  ContainerCreating  0       0s

```

NOTES:

1. Get your 'admin' user password by running:

```
kubectl get secret --namespace grafana grafana -o jsonpath=".data.admin-password" | base64 --decode ; echo
```

2. The Grafana server can be accessed via port 80 on the following DNS name from within your cluster:

```
grafana.grafana.svc.cluster.local
```

Get the Grafana URL to visit by running these commands in the same shell:

```
export POD_NAME=$(kubectl get pods --namespace grafana -l "app=grafana,component=" -o jsonpath=".items[0].metadata.name")
kubectl --namespace grafana port-forward $POD_NAME 3000
```

3. Login with the password from step 1 and the username: admin

```
#####
##### WARNING: Persistence is disabled!!! You will lose your data when #####
#####           the Grafana pod is terminated.                         #####
#####
```

```
cloud_user@ip-10-0-1-101:~$ 
```

We can see which pods are running in this new namespace with the command:

```
kubectl get pods -n grafana
```

```
cloud_user@ip-10-0-1-101:~$ kubectl get pods -n grafana
NAME                  READY   STATUS    RESTARTS   AGE
grafana-d65c89cdc-778vg   0/1     Running   0          1m
cloud_user@ip-10-0-1-101:~$ █
```

Deploy a NodePort Service to Provide External Access to Grafana

Make a file called grafana-ext.yml:

```
vi grafana-ext.yml
```

Paste in this content:

```
kind: Service
apiVersion: v1
metadata:
  namespace: grafana
  name: grafana-ext
spec:
  type: NodePort
  selector:
    app: grafana
  ports:
  - protocol: TCP
    port: 3000
    nodePort: 8081
```

```
kind: Service
apiVersion: v1
metadata:
  namespace: grafana
  name: grafana-ext
spec:
  type: NodePort
  selector:
    app: grafana
  ports:
  - protocol: TCP
    port: 3000
    nodePort: 8081█
```

```
~  
~  
~
```

Save and close the file:

:wq

Deploy the service:

kubectl apply -f ~/grafana-ext.yml

```
cloud_user@ip-10-0-1-101:~$ vi grafana-ext.yml
cloud_user@ip-10-0-1-101:~$ kubectl apply -f ~/grafana-ext.yml
service "grafana-ext" created
cloud_user@ip-10-0-1-101:~$ █
```

Log in to Grafana using the Kubernetes Node Public IP

<Grafana Public IP address>:8081



Log in using the following credentials that were set earlier:

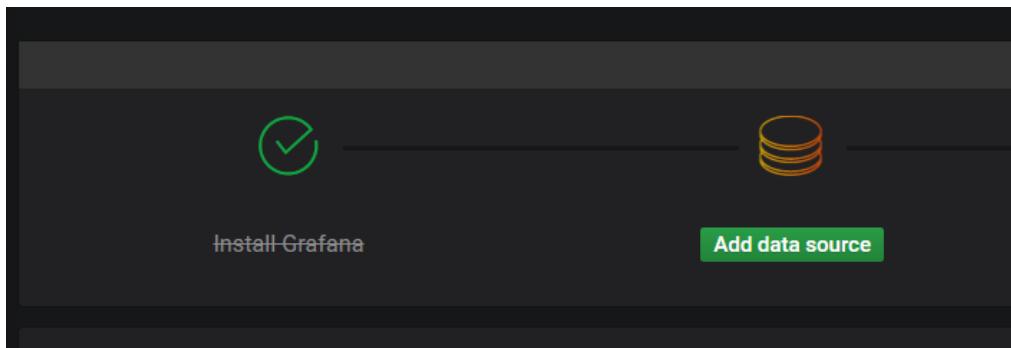
Username: admin

Password: password

Create the Monitoring Dashboards

Add a Datasource for Prometheus

Click on **Add data source**



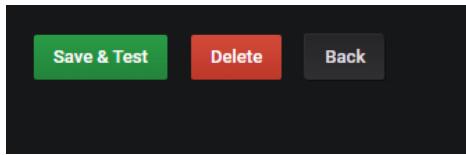
Name: **Kubernetes**

Type: **Prometheus**

URL: <http://prometheus-server.prometheus.svc.cluster.local>

Click **Save & Test**

The screenshot shows the 'Data Sources / Kubernetes' configuration page. The 'Settings' tab is selected. The data source is named 'Kubernetes' and is of type 'Prometheus'. It is set as the 'Default' source. The 'HTTP' section includes fields for 'URL' (set to 'http://localhost:9090') and 'Access' (set to 'proxy'). The 'Auth' section contains options for 'Basic Auth' and 'TLS Client Auth', both with their respective checkboxes unchecked. There is also a checkbox for 'Skip TLS Verification (Insecure)'. The 'Advanced HTTP Settings' section includes 'Whitelisted Cookies' and 'Add Name' buttons, and a 'Scrape interval' field set to '15s'.



Add the Kubernetes All Nodes Community Dashboard

Hover your mouse over the + in the left sidebar and click on **Import**.

In the Grafana.com Dashboard field, provide the ID **3131**.

Click outside of the field to load information about the dashboard.

In the Options section:

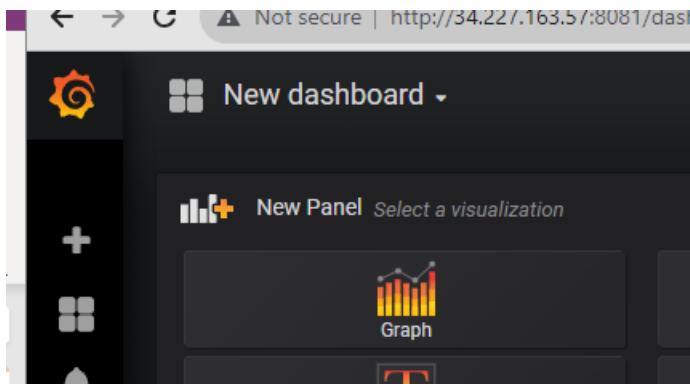
prometheus: **Kubernetes**

Click **Import**.

The screenshot shows the 'Import' dialog box. At the top, there's a logo with an orange arrow pointing up and the word 'Import'. Below it, the text 'Import dashboard from file or Grafana.com' is displayed. Underneath, a section titled 'Importing Dashboard from [Grafana.com](#)' shows details: 'Published by' (Bart Van Bos) and 'Updated on' (2017-09-06 14:14:13). A 'Options' section contains two rows: 'Name' (Kubernetes All Nodes) and 'prometheus' (Kubernetes). Both rows have a checked checkbox at the end. At the bottom, there are 'Import' and 'Cancel' buttons.

Hover your mouse over the + in the left sidebar and then click **Dashboard**.

Select the Graph panel.



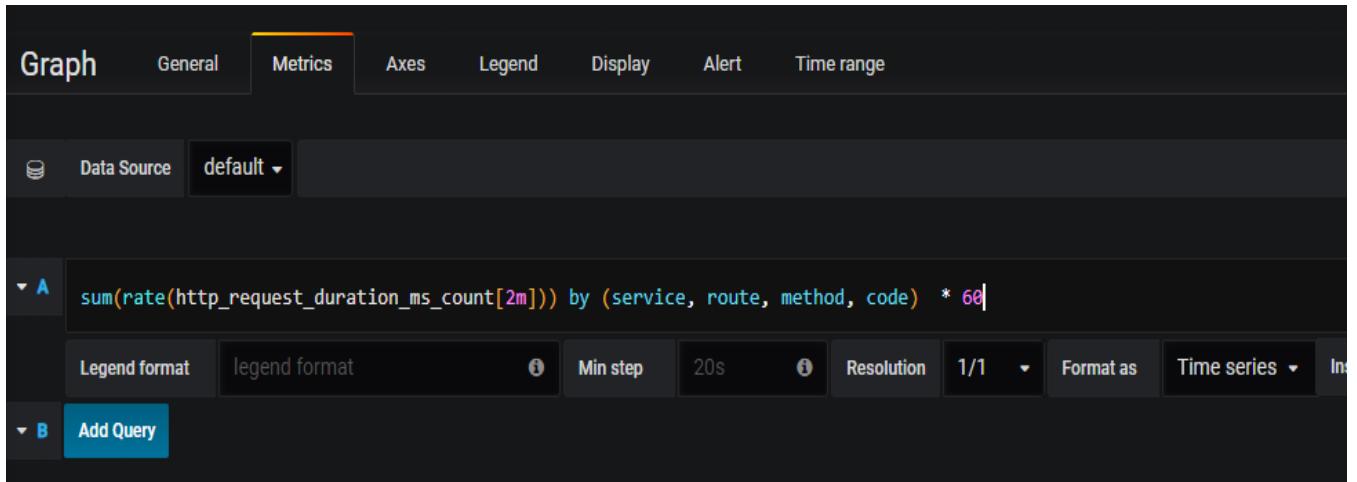
Hover over the Panel Title and click Edit.

In the General tab at the bottom of the screen, set the Title to Requests Per Minute.

A screenshot of the Grafana panel configuration interface. At the top, tabs are labeled: Graph (selected), General, Metrics, Axes, Legend, Display, and Alerts. The General tab shows a table under the "Info" section. The "Title" row has a text input field containing "Requests Per Minute". The "Description" row has a larger text area containing "Panel description, supports markdown & links". A "Transparent" checkbox is checked. To the right of the table, there are buttons for "Repeat" and "For each item".

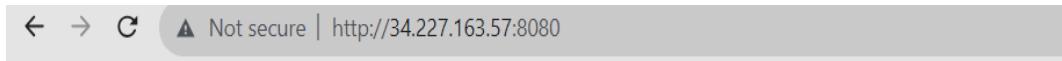
In the Metrics tab, paste in the following query:

```
sum(rate(http_request_duration_ms_count[2m])) by (service, route, method, code) * 60
```



With that in place, let's load our train-schedule app to give our graph some data:

<Grafana Public IP Address>:8080



Find your train!

Select your train below to see its current schedule.

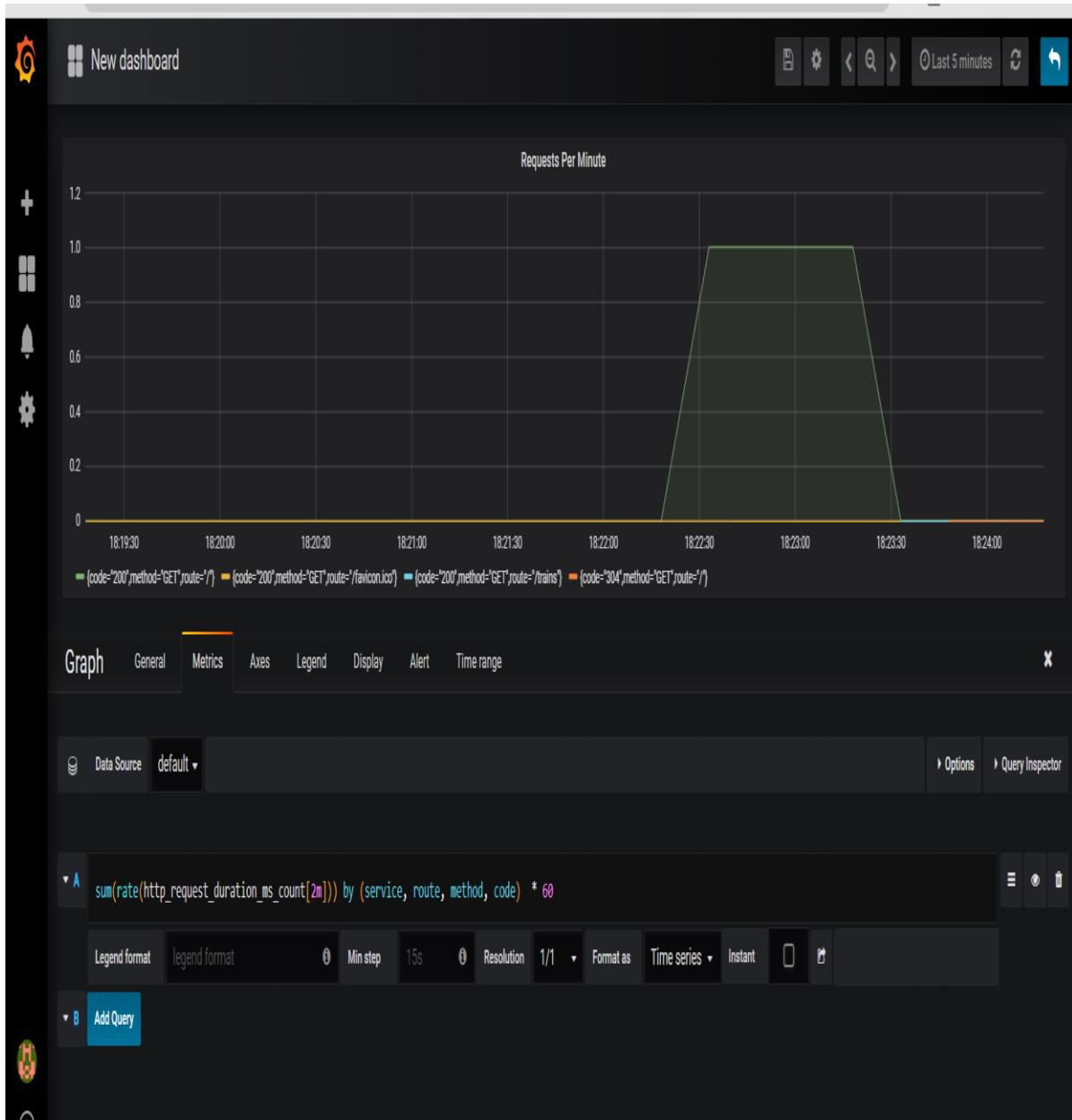
Trains

Flying Scotsman Golden Arrow Hogwarts Express Orient Express

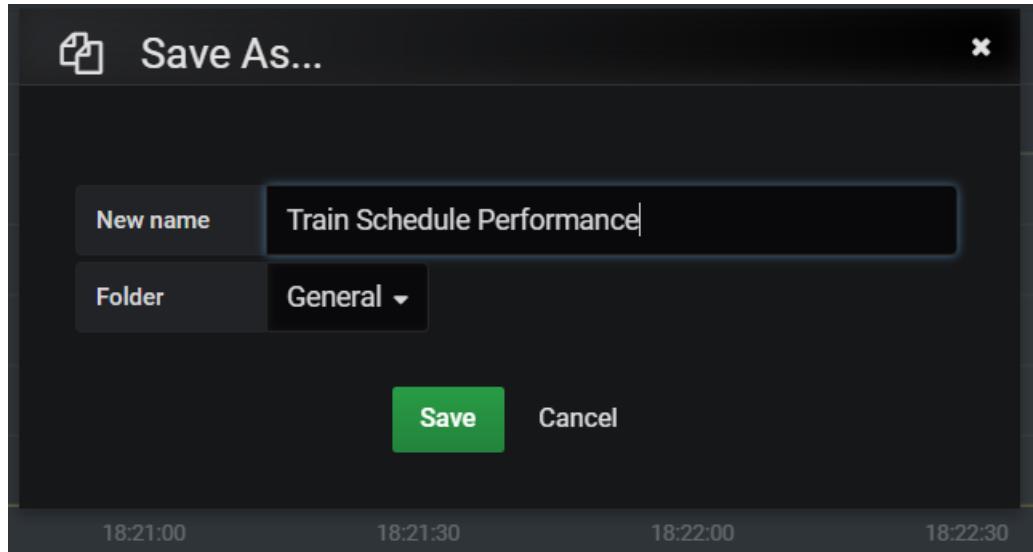
Select a train to view its current schedule.

Refresh the page a few times.

Now, navigate back to the Grafana dashboard tab in your browser.
In the top-right of the page, click Last 6 hours and change the time selection
to Last 5 minutes.



Click Back to dashboard in the top-right of the page.
Next, click Save dashboard, also in the top-right of the page.



Name the dashboard "**Train Schedule Performance**" and click **Save**.

||||||||||||||||||||||||||||||||||||||||||||||||
Addendum

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

```
mkdir -p $HOME/.kube  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

You should now deploy a pod network to the cluster.

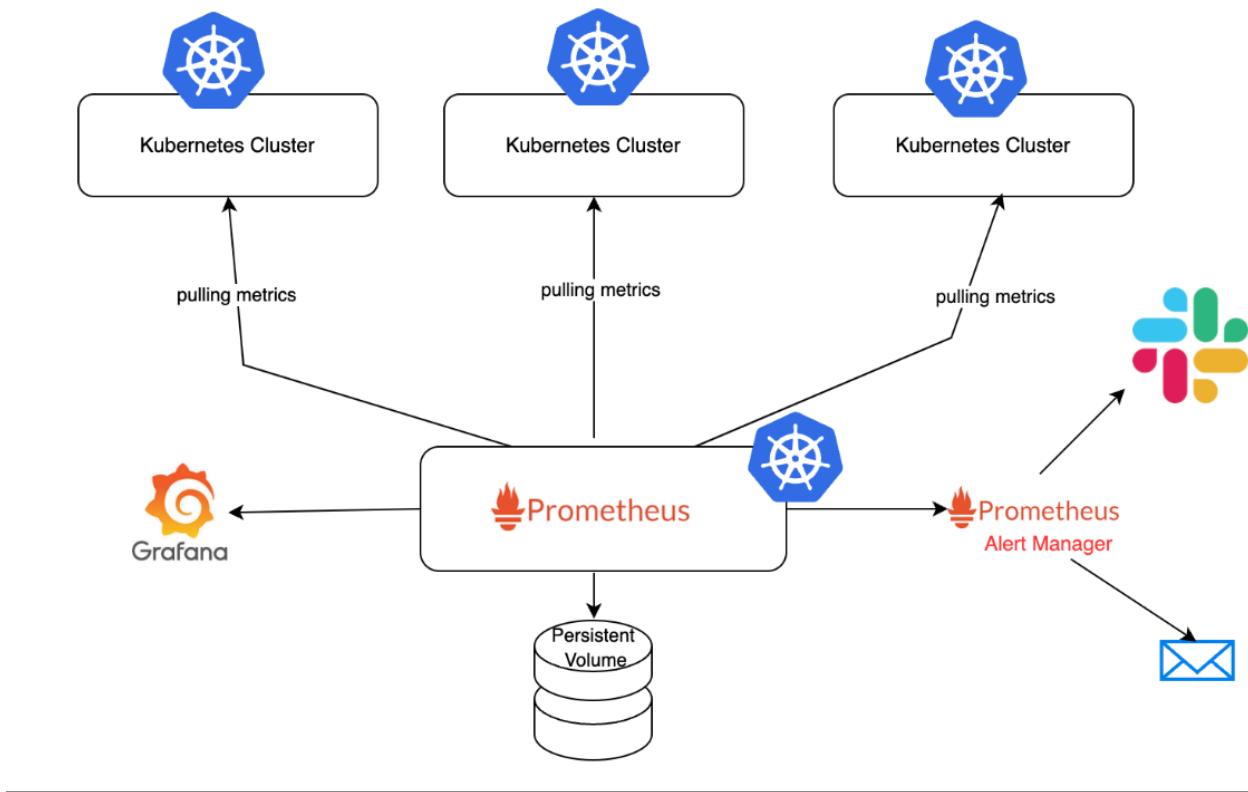
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
<https://kubernetes.io/docs/concepts/cluster-administration/addons/>

Then you can join any number of worker nodes by running the following on each as root:

```
kubeadm join 172.31.27.197:6443 --token 2kla1r.ncp3uo7myontulj7 \  
--discovery-token-ca-cert-hash  
sha256:228f75560859d6389ec4b6ac2b2b463f44bf42aa5f03c396596c8ec835e871ca  
[root@9c438d91c5a44174a4a8c7a5cd9beb882c cloud_user]#  
||||||||||||||||||||||||||||||||||||||||||||||||||||
```

How to setup monitoring on an EKS Kubernetes Cluster using Prometheus and Grafana

Prometheus and Grafana Setup for Monitoring Kubernetes Clusters



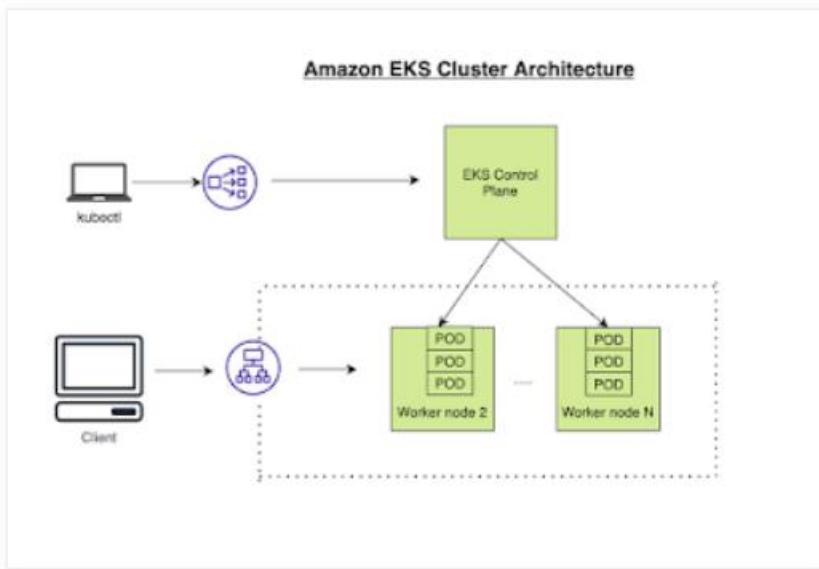
Pre-requisites:

- EC2 instance to access EKS cluster.
- EKS Cluster needs to be up and running.
- Install Helm3

How to create EKS cluster in AWS cloud using eksctl.

What is Amazon EKS

Amazon EKS is a fully managed container orchestration service. EKS allows you to quickly deploy a production ready Kubernetes cluster in AWS, deploy and manage containerized applications more easily with a fully managed Kubernetes service.



Pre-requisites:

An EC2 instance (Ubuntu 22.4 LTS)

Login to the AWS portal.

Go to **EC2**

Launch Instance

Launch instance

Name and tags

Name: VM

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux	macOS	Ubuntu	Windows	Red Hat	SUSE Linux	Browse more AMIs Including AMIs from AWS, Marketplace and the Community
--------------	-------	--------	---------	---------	------------	--

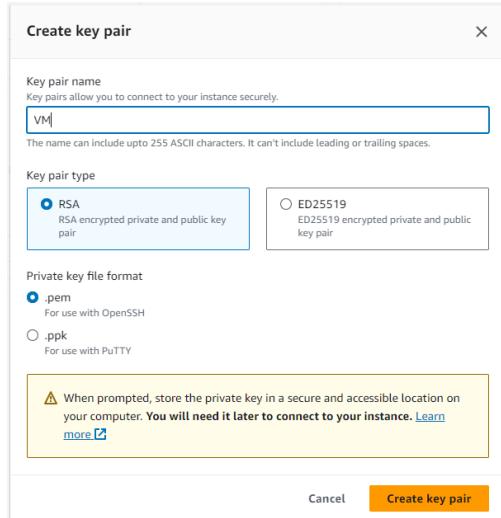
Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
ami-053b0d53c279ac90 (64-bit (x86)) / ami-0a0cbeebcd6dcdb0 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Select Ubuntu

Create a new key pair.



Click **Create key pair**.

Create the Instance.

Now select **View all instances**.

The screenshot shows the 'Instances (1) [Info](#)' page. An orange button labeled 'View all instances' is visible. The table lists one instance: 'VM' (Instance ID: i-098a01231c18f85ab, State: Running, Type: t2.micro, Status: Initializing). Action buttons include 'Connect', 'Actions', and 'Launch instances'.

Select the instance on the left.

The screenshot shows the 'Instance summary for i-098a01231c18f85ab (VM) [Info](#)' page. It displays the instance's state as 'Running' and provides a 'Connect' button.

Select the **Connect** button on the right.

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
 i-098a01231c18f85ab (VM)

Connection Type
 Connect using EC2 Instance Connect
 Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

Connect using EC2 Instance Connect Endpoint
 Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

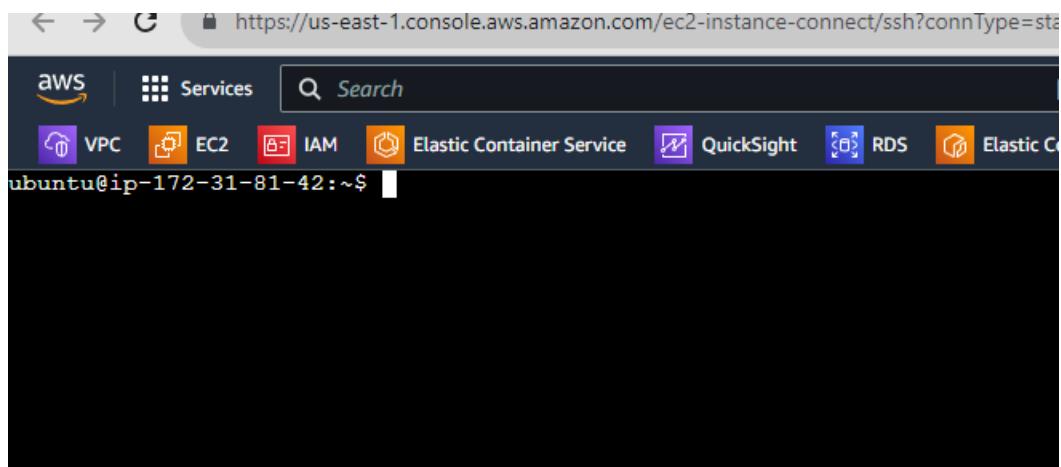
Public IP address
 3.95.217.226

User name
 Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ubuntu.

Note: In most cases, the default user name, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel **Connect**

Click the **Connect** button.



Install AWS CLI – Command line tools for working with AWS services, including Amazon EKS.

[Download the AWS CLI](#)

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

```
ubuntu@ip-172-31-81-42:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
% Total    % Received % Xferd  Average Speed   Time     Time   Current
                                         Dload  Upload Total Spent   Left Speed
100 55.8M  100 55.8M    0     0  110M      0 --:--:-- --:--:-- 110M
```

Install the unzip tool

sudo apt install unzip

```
ubuntu@ip-172-31-81-42:~$ sudo apt install unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  zip
The following NEW packages will be installed:
  unzip
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 174 kB of archives.
After this operation, 385 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 unzip amd64 6.0-26ubuntu3.1 [174 kB]
Fetched 174 kB in 0s (6351 kB/s)
Selecting previously unselected package unzip.
(Reading database ... 64295 files and directories currently installed.)
Preparing to unpack .../unzip_6.0-26ubuntu3.1_amd64.deb ...
Unpacking unzip (6.0-26ubuntu3.1) ...
Setting up unzip (6.0-26ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

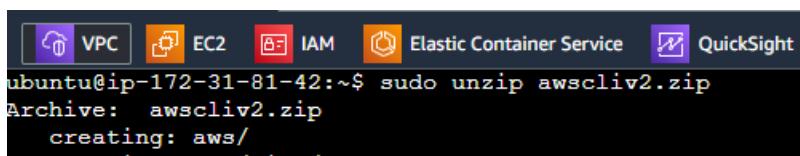
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-81-42:~$
```

Unzip the AWS CLI

sudo unzip awscliv2.zip



Install the AWS CLI tool

sudo ./aws/install

```
ubuntu@ip-172-31-81-42:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
ubuntu@ip-172-31-81-42:~$ █
```

Check if the AWS CLI tool is installed

aws --version

it should display the below output.

```
ubuntu@ip-172-31-81-42:~$ aws --version
aws-cli/2.13.22 Python/3.11.5 Linux/5.19.0-1025-aws exe/x86_64.ubuntu.22 prompt/off
ubuntu@ip-172-31-81-42:~$ █
```

Install eksctl – This is a command line tool for working with EKS clusters that automates many individual tasks.

The **eksctl** tool uses **CloudFormation** under the hood, creating one stack for the **EKS master control plane** and another stack for the **worker nodes**.

Download and extract the latest release of **eksctl** with the following command.

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

```
ubuntu@ip-172-31-81-42:~$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
ubuntu@ip-172-31-81-42:~$ █
```

Move the extracted binary to /usr/local/bin.

```
sudo mv /tmp/eksctl /usr/local/bin
```

```
ubuntu@ip-172-31-81-42:~$ sudo mv /tmp/eksctl /usr/local/bin
ubuntu@ip-172-31-81-42:~$ █
```

Check if the eksctl is running.

```
eksctl version
```

```
ubuntu@ip-172-31-81-42:~$ eksctl version
0.159.0
ubuntu@ip-172-31-81-42:~$ █
```

Install kubectl – This is a command line tool for working with Kubernetes clusters.

Kubernetes uses a command line utility called kubectl for communicating with the cluster API server. It is tool for controlling Kubernetes clusters. kubectl looks for a file named config in the \$HOME directory.

Downloading kubectl and placing it within a specific location.

```
sudo curl --silent --location -o /usr/local/bin/kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl
```

```
ubuntu@ip-172-31-81-42:~$ sudo curl --silent --location -o /usr/local/bin/kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kub  
ectl  
ubuntu@ip-172-31-81-42:~$ █
```

Making the kubectl executable.

```
sudo chmod +x /usr/local/bin/kubectl
```

```
ubuntu@ip-172-31-81-42:~$ sudo chmod +x /usr/local/bin/kubectl  
ubuntu@ip-172-31-81-42:~$ █
```

Verifying if kubectl got installed

```
kubectl version --short --client
```

```
ubuntu@ip-172-31-81-42:~$ kubectl version --short --client  
Client Version: v1.22.6-eks-7d68063  
ubuntu@ip-172-31-81-42:~$ █
```

NOTE: The **-short** flag doesn't work on all Linux distributions.

Install Helm 3

What is Helm and How to install Helm version 3?

Helm is a package manager for Kubernetes. Helm is the K8s equivalent of yum or apt. It accomplishes the same goals as Linux system package managers like APT or YUM: managing the installation of applications and dependencies behind the scenes and hiding the complexity from the user.

Why use Helm?

As the Kubernetes platform and ecosystem continued to expand, deploying one and only one Kubernetes configuration file (ie: a single YAML) was not the norm anymore. As number of K8S deployment files increased, how to manage those files? Helm solves that problem.

Helm Charts

Helm uses a packaging format called Charts. A Helm Chart is a collection of files that describe a set of Kubernetes resources. Helm Charts help you define, install, and upgrade even the most complex Kubernetes application. Charts are easy to create, version, share, and publish.

```
devopescoaching@DevOpsCoachMBP springboot % tree mychart
mychart
└── Chart.yaml
    └── Charts
        └── templates
            ├── NOTES.txt
            ├── _helpers.tpl
            ├── deployment.yaml
            ├── hpa.yaml
            ├── ingress.yaml
            ├── service.yaml
            └── serviceaccount.yaml
            └── tests
                └── test-connection.yaml
    └── values.yaml

4 directories, 10 files
```

Helm Kubernetes Integration



In Helm 3 there is no tiller component (a feature of earlier versions of Helm). The Helm client directly interacts with the Kubernetes API for the helm chart deployment.

Helm 3 can be installed in many ways. We will, however, install Helm 3 using scripts option.

Download scripts

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
```

```
ubuntu@ip-172-31-81-42:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
ubuntu@ip-172-31-81-42:~$
```

Provide permission

```
sudo chmod 700 get_helm.sh
```

```
ubuntu@ip-172-31-81-42:~$ sudo chmod 700 get_helm.sh
ubuntu@ip-172-31-81-42:~$
```

Execute script to install

```
sudo ./get_helm.sh
```

```
ubuntu@ip-172-31-81-42:~$ sudo ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.13.0-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
ubuntu@ip-172-31-81-42:~$
```

Verify installation

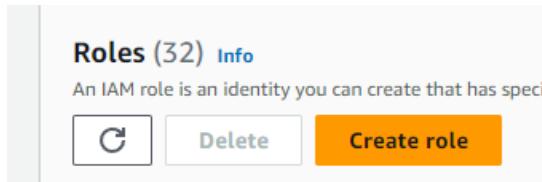
helm version --client

```
ubuntu@ip-172-31-81-42:~$ helm version --client
version.BuildInfo{Version:"v3.13.0", GitCommit:"825e86f6a7a38cef1112bfa606e4127a706749b1", GitTreeState:"clean", GoVersion:"go1.20.8"}
ubuntu@ip-172-31-81-42:~$
```

Create an IAM Role with Administrator Access

You need to create an **IAM** role with **AdministratorAccess** policy.

Go to AWS console, **IAM**, click on **Roles** and click **create role**



Select **AWS services**, Click **EC2**, Click on **Next permissions**.

Trusted entity type

<input checked="" type="radio"/> AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.	<input type="radio"/> AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.	<input type="radio"/> Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
<input type="radio"/> SAML 2.0 federation Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.	<input type="radio"/> Custom trust policy Create a custom trust policy to enable others to perform actions in this account.	

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

EC2

Choose a use case for the specified service.

Use case

EC2

Allows EC2 instances to call AWS services on your behalf.

Now search for **AdministratorAccess** policy and click **Next**

<input type="checkbox"/>	<input checked="" type="checkbox"/>  AdministratorAccess	AWS m... Grants full access permissions necessary for resource
<input checked="" type="checkbox"/>	 AdministratorAccess	AWS m... Provides full access to AWS services and resources.

Now give a role name and click **Create role**.

IAM > Roles > Create role

Step 1
[Select trusted entity](#)

Step 2
[Add permissions](#)

Step 3
Name, review, and create

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+,-,_' characters.

Description

Create role

Assign the role to EC2 instance

Go to **AWS** console, click on **EC2**, select the EC2 instance, Choose **Security**. Click on **Modify IAM role**.

The screenshot shows the AWS EC2 Instances page. There is one instance listed: 'VM' (Instance ID: i-098a01231c18f85ab, State: Running, Type: t2.micro). A context menu is open for this instance, with the 'Security' option highlighted. Other options in the menu include 'Change security groups', 'Get Windows password', 'Modify IAM role', and 'Monitor and troubleshoot'.

Choose the role you have created from the dropdown.

Select the role and click on **Apply**.

The screenshot shows the 'Modify IAM role' dialog box. At the top, it shows the path: EC2 > Instances > i-098a01231c18f85ab > Modify IAM role. The main section is titled 'Modify IAM role' with an 'Info' link. It says 'Attach an IAM role to your instance.' Below that, the 'Instance ID' is listed as 'i-098a01231c18f85ab (VM)'. Under 'IAM role', it says 'Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.' A dropdown menu shows 'eks-admin-role' selected. To the right of the dropdown is a 'Create new IAM role' button. At the bottom, there are 'Cancel' and 'Update IAM role' buttons, with 'Update IAM role' being highlighted.

Instances > Attach/Replace IAM Role

Attach/Replace IAM Role

 IAM role operation succeeded

Create EKS Cluster with two worker nodes using eksctl

```
eksctl create cluster --name demo-eks --region us-east-1 --nodegroup-name my-nodes --node-type t3.small --managed --nodes 2
```

the above command should create a EKS cluster in AWS, it might take 15 to 20 mins. The **eksctl** tool uses CloudFormation under the hood, creating one stack for the EKS master control plane and another stack for the worker nodes.

```
ubuntu@ip-172-31-81-42:~$ eksctl create cluster --name demo-eks --region us-east-1 --nodegroup-name my-nodes --node-type t3.small --managed --nodes 2
2023-09-28 08:53:06 [i] eksctl version 0.159.0
2023-09-28 08:53:06 [i] using region us-east-1
2023-09-28 08:53:06 [i] skipping us-east-1e from selection because it doesn't support the following instance type(s): t3.small
2023-09-28 08:53:06 [i] setting availability zones to [us-east-1d us-east-1c]
2023-09-28 08:53:06 [i] subnets for us-east-1d - public:192.168.0.0/19 private:192.168.64.0/19
2023-09-28 08:53:06 [i] subnets for us-east-1c - public:192.168.32.0/19 private:192.168.96.0/19
2023-09-28 08:53:06 [i] nodegroup "my-nodes" will use "" [AmazonLinux2/1.25]
2023-09-28 08:53:06 [i] using Kubernetes version 1.25
2023-09-28 08:53:06 [i] creating EKS cluster "demo-eks" in "us-east-1" region with managed nodes
2023-09-28 08:53:06 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2023-09-28 08:53:06 [i] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=demo-eks'
2023-09-28 08:53:06 [i] Kubernetes API endpoint access will use default of (publicAccess=true, privateAccess=false) for cluster "demo-eks" in "us-east-1"
2023-09-28 08:53:06 [i] CloudWatch logging will not be enabled for cluster "demo-eks" in "us-east-1"
2023-09-28 08:53:06 [i] you can enable it with 'eksctl utils update-cluster-logging --enable-types=[SPECIFY-YOUR-LOG-TYPES-HERE] (e.g. all) --region=us-east-1 --cluster=demo-eks'
2023-09-28 08:53:06 [i]
2 sequential tasks: ( create cluster control plane "demo-eks",
  2 sequential sub-tasks: (
    wait for control plane to become ready,
    create managed nodegroup "my-nodes",
  )
)
2023-09-28 08:53:06 [i] building cluster stack "eksctl-demo-eks-cluster"
2023-09-28 08:53:06 [i] deploying stack "eksctl-demo-eks-cluster"
```

CloudFormation X

[CloudFormation](#) > Stacks

Stacks (2)

C Delete Update Stack actions ▾ Create stack ▾

Filter status Active View nested < 1 > ⚙

Stack name Status Created time Description

eksctl-demo-eks-cluster CREATE_IN_PROGRESS 2023-09-28 11:53:06 UTC+0300 EKS cluster [dedicated VPC: true, dedicated IAM: true] [created and managed by eksctl]

cfst-1449-3026bde225e07a2c93b07e789959fa23 CREATE_COMPLETE 2023-09-28 10:44:09 UTC+0300 -

Stacks

StackSets

Exports

Designer

Registry

Public extensions

Activated extensions

CloudFormation > Stacks

Stacks (2)

C Delete Update Stack actions ▾ Create stack ▾

Filter status Active View nested < 1 > ⚙

Stack name Status Created time Description

eksctl-demo-eks-cluster CREATE_COMPLETE 2023-09-28 11:53:06 UTC+0300 EKS cluster [dedicated VPC: true, dedicated IAM: true] [created and managed by eksctl]

cfst-1449-3026bde225e07a2c93b07e789959fa23 CREATE_COMPLETE 2023-09-28 10:44:09 UTC+0300 -

The EKS cluster has been created.

```
eksctl get cluster --name demo-eks --region us-east-1
```

The above should confirm that EKS cluster is up and running.

```
ubuntu@ip-172-31-81-42:~$ eksctl get cluster --name demo-eks --region us-east-1
NAME      VERSION STATUS CREATED          VPC           SUBNETS
SECURITYGROUPS      PROVIDER
demo-eks      1.25   ACTIVE 2023-09-28T08:53:30Z  vpc-05fee4f22b2eb7f92  subnet-028e3389dd7a28274,subnet-059cb0cd9161466ae,subnet-0f5c03b32882d1e8f,subnet-0fd5d
41a07fbfc0de  sg-05c8b0d8016035fc7  EKS
ubuntu@ip-172-31-81-42:~$
```

Update Kube config by entering command below:

```
- aws eks update-kubeconfig --name demo-eks --region us-east-1
```

```
ubuntu@ip-172-31-81-42:~$ aws eks update-kubeconfig --name demo-eks --region us-east-1
Added new context arn:aws:eks:us-east-1:594182463744:cluster/demo-eks to /home/ubuntu/.kube/config
ubuntu@ip-172-31-81-42:~$
```

Connect to EKS cluster using kubectl commands

To view the list of worker nodes as part of EKS cluster.

```
kubectl get nodes
```

```
ubuntu@ip-172-31-81-42:~$ kubectl get nodes
NAME           STATUS    ROLES      AGE    VERSION
ip-192-168-28-73.ec2.internal  Ready    <none>    51s    v1.25.13-eks-43840fb
ip-192-168-52-210.ec2.internal  Ready    <none>    44s    v1.25.13-eks-43840fb
ubuntu@ip-172-31-81-42:~$
```

```
kubectl get ns
```

```
ubuntu@ip-172-31-81-42:~$ kubectl get ns
NAME        STATUS   AGE
default     Active   9m14s
kube-node-lease  Active   9m16s
kube-public   Active   9m16s
kube-system   Active   9m16s
ubuntu@ip-172-31-81-42:~$
```

Deploy Nginx on a Kubernetes Cluster

Let us run some apps to make sure they are deployed to Kubernetes cluster. The below command will create deployment:

```
kubectl create deployment nginx --image=nginx
```

```
ubuntu@ip-172-31-81-42:~$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
ubuntu@ip-172-31-81-42:~$
```

View Deployments

```
kubectl get deployments
```

```
ubuntu@ip-172-31-81-42:~$ kubectl get deployments
NAME      READY     UP-TO-DATE   AVAILABLE   AGE
nginx    1/1       1           1           45s
ubuntu@ip-172-31-81-42:~$
```

Errors during Cluster creation

If you are having issues when creating a cluster, try to delete the cluster by executing the below command and re-create it.

```
eksctl delete cluster --name demo-eks --region us-east-1
```

```
ubuntu@ip-172-31-81-42:~$ eksctl delete cluster --name demo-eks --region us-east-1
[...]
I overrode external and cluster hasn't been created properly, you may wish to check CloudFormation console
D-29 23:23:00 [4]  to cleanup resources, run "eksctl delete cluster --region us-east-1 --remove-when-dt"
D-29 23:23:00 [4]  creating CloudFormation stack "eksctl-demo-eks-cluster"; AlreadyByTimestamp: Stack [eksctl-demo-eks-cluster] already exists
StackARN: arn:aws:cloudformation:us-east-1:60439514709:stack/eksctl-demo-eks-cluster/20190929145052
[4] 1 failed to cleanup cluster "demo-eks".
```

or Login to **AWS console** --> **AWS Cloud formation** --> **delete the stack manually**.

You can also delete the cluster under AWS console --> Elastic Kubernetes Service --> Clusters

Click on Delete cluster

Implementation steps

We need to add the Helm Stable Charts for your local client. Execute the below command:

```
helm repo add stable https://charts.helm.sh/stable
```

```
ubuntu@ip-172-31-81-42:~$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
ubuntu@ip-172-31-81-42:~$
```

Add prometheus Helm repo

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

```
ubuntu@ip-172-31-81-42:~$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
ubuntu@ip-172-31-81-42:~$
```

helm search repo prometheus-community

Prometheus and grafana helm chart moved to kube prometheus stack

NAME	CHART VERSION	APP VERSION	DESCRIPTION
prometheus-community/alertmanager	1.7.0	v0.26.0	The Alertmanager handles alerts sent by client ...
prometheus-community/alertmanager-snmp-notifier	0.1.2	v1.4.0	The SNMP Notifier handles alerts coming from Pr...
prometheus-community/jiralert	1.6.0	v1.3.0	A Helm chart for Kubernetes to install jiralert...
prometheus-community/kube-prometheus-stack	51.2.0	v0.68.0	kube-prometheus-stack collects Kubernetes manif...
prometheus-community/kube-state-metrics	5.13.0	v2.10.0	Install kube-state-metrics to generate and expo...
prometheus-community/prom-label-proxy	0.6.0	v0.7.0	A proxy that enforces a given label in a given ...
prometheus-community/prometheus	25.0.0	v2.47.0	Prometheus is a monitoring system and time seri...
prometheus-community/prometheus-adapter	4.5.0	v0.11.1	A Helm chart for k8s prometheus adapter
prometheus-community/prometheus-blackbox-exporter	8.4.0	v0.24.0	Prometheus Blackbox Exporter
prometheus-community/prometheus-cloudwatch-expo...	0.25.2	0.15.4	A Helm chart for prometheus cloudwatch-exporter
prometheus-community/prometheus-contrack-stats...	0.5.7	v0.4.15	A Helm chart for contrack-stats-exporter
prometheus-community/prometheus-consul-exporter	1.0.0	0.4.0	A Helm chart for the Prometheus Consul Exporter
prometheus-community/prometheus-couchdb-exporter	1.0.0	1.0	A Helm chart to export the metrics from couchdb...
prometheus-community/prometheus-druid-exporter	1.1.0	v0.11.0	Druid exporter to monitor druid metrics with Pr...
prometheus-community/prometheus-elasticsearch-e...	5.3.0	v1.6.0	Elasticsearch stats exporter for Prometheus
prometheus-community/prometheus-fastly-exporter	0.2.0	7.2.4	A Helm chart for the Prometheus Fastly Exporter
prometheus-community/prometheus-ipmi-exporter	0.1.0	1.6.1	This is an IPMI exporter for Prometheus.
prometheus-community/prometheus-json-exporter	0.7.1	v0.5.0	Install prometheus-json-exporter
prometheus-community/prometheus-kafka-exporter	2.6.0	v1.7.0	A Helm chart to export the metrics from Kafka i...
prometheus-community/prometheus-modbus-exporter	0.1.0	0.4.0	A Helm chart for prometheus-modbus-exporter
prometheus-community/prometheus-mongodb-exporter	3.4.0	0.39.0	A Prometheus exporter for MongoDB metrics
prometheus-community/prometheus-mysql-exporter	2.0.0	v0.15.0	A Helm chart for prometheus mysql exporter with...
prometheus-community/prometheus-nats-exporter	2.13.0	0.12.0	A Helm chart for prometheus-nats-exporter
prometheus-community/prometheus-nginx-exporter	0.1.1	0.11.0	A Helm chart for the Prometheus NGINX Exporter
prometheus-community/prometheus-node-exporter	4.23.1	1.6.1	A Helm chart for prometheus node-exporter
prometheus-community/prometheus-operator	9.3.2	0.38.1	DEPRECATED - This chart will be renamed. See ht...
prometheus-community/prometheus-operator-admiss...	0.7.0	0.68.0	Prometheus Operator Admission Webhook
prometheus-community/prometheus-operator-crds	6.0.0	v0.68.0	A Helm chart that collects custom resource defi...
prometheus-community/prometheus-pgbouncer-exporter	0.1.1	1.18.0	A Helm chart for prometheus pgbouncer-exporter
prometheus-community/prometheus-pingdom-exporter	2.5.0	20190610-1	A Helm chart for Prometheus Pingdom Exporter
prometheus-community/prometheus-pingmesh-exporter	0.3.0	v1.1.0	Prometheus Pingmesh Exporter
prometheus-community/prometheus-postgres-exporter	5.1.0	v0.14.0	A Helm chart for prometheus postgres-exporter
prometheus-community/prometheus-pushgateway	2.4.1	v1.6.1	A Helm chart for prometheus pushgateway
prometheus-community/prometheus-rabbitmq-exporter	1.8.1	v0.29.0	Rabbitmq metrics exporter for prometheus
prometheus-community/prometheus-redis-exporter	6.0.0	v1.54.0	Prometheus exporter for Redis metrics
prometheus-community/prometheus-smartctl-exporter	0.6.0	v0.11.0	A Helm chart for Kubernetes

Now create the Prometheus namespace

kubectl create namespace prometheus

```
ubuntu@ip-172-31-81-42:~$ kubectl create namespace prometheus
namespace/prometheus created
ubuntu@ip-172-31-81-42:~$
```

Install kube-prometheus-stack

Below is helm command to install kube-prometheus-stack. The helm repo kube-stack-prometheus (formerly prometheus-operator) comes with a grafana deployment embedded.

helm install stable prometheus-community/kube-prometheus-stack -n prometheus

```
ubuntu@ip-172-31-81-42:~$ helm install stable prometheus-community/kube-prometheus-stack -n prometheus

NAME: stable
LAST DEPLOYED: Thu Sep 28 09:15:30 2023
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace prometheus get pods -l "release=stable"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
ubuntu@ip-172-31-81-42:~$
```

Lets check if prometheus and grafana pods are running already

kubectl get pods -n prometheus

```
ubuntu@ip-172-31-81-42:~$ kubectl get pods -n prometheus
NAME                               READY   STATUS    RESTARTS   AGE
alertmanager-stable-kube-prometheus-sta-alertmanager-0   2/2     Running   0          49s
prometheus-stable-kube-prometheus-sta-prometheus-0        2/2     Running   0          49s
stable-grafana-6c8fd88f56-zrcpr                         3/3     Running   0          54s
stable-kube-prometheus-sta-operator-75947cf97-xjmmv       1/1     Running   0          54s
stable-kube-state-metrics-7ccdc6767c-xj26t               1/1     Running   0          54s
stable-prometheus-node-exporter-w646x                    1/1     Running   0          54s
stable-prometheus-node-exporter-x6w9w                     1/1     Running   0          54s
ubuntu@ip-172-31-81-42:~$
```

```
kubectl get svc -n prometheus
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
alertmanager-operated	ClusterIP	None	<none>	9093/TCP,9094/TCP,9094/UDP	90s
prometheus-operated	ClusterIP	None	<none>	9090/TCP	90s
stable-grafana	ClusterIP	10.100.186.94	<none>	80/TCP	95s
stable-kube-prometheus-sta-alertmanager	ClusterIP	10.100.254.197	<none>	9093/TCP,8080/TCP	95s
stable-kube-prometheus-sta-operator	ClusterIP	10.100.85.234	<none>	443/TCP	95s
stable-kube-prometheus-sta-prometheus	ClusterIP	10.100.229.18	<none>	9090/TCP,8080/TCP	95s
stable-kube-state-metrics	ClusterIP	10.100.245.2	<none>	8080/TCP	95s
stable-prometheus-node-exporter	ClusterIP	10.100.247.199	<none>	9100/TCP	95s

This confirms that prometheus and grafana have been installed successfully using Helm.

In order to make prometheus and grafana available outside the cluster, we need to use **LoadBalancer** or **NodePort** instead of **ClusterIP** within the service files.

Edit Prometheus Service

```
kubectl edit svc stable-kube-prometheus-sta-prometheus -n prometheus
```

```
ipFamilyPolicy: SingleStack
ports:
- name: http-web
  port: 9090
  protocol: TCP
  targetPort: 9090
- appProtocol: http
  name: reloader-web
  port: 8080
  protocol: TCP
  targetPort: reloader-web
selector:
  app.kubernetes.io/name: prometheus
  operator.prometheus.io/name: stable-kube-prometheus-sta-prometheus
sessionAffinity: None
type: LoadBalancer
status:
  loadBalancer: {}
```

Edit Grafana Service

```
kubectl edit svc stable-grafana -n prometheus
```

```
ipFamilyPolicy: SingleStack
ports:
- name: http-web
  port: 9090
  protocol: TCP
  targetPort: 9090
- appProtocol: http
  name: reloader-web
  port: 8080
  protocol: TCP
  targetPort: reloader-web
selector:
  app.kubernetes.io/name: prometheus
  operator.prometheus.io/name: stable-kube-prometheus-sta-prometheus
sessionAffinity: None
type: LoadBalancer
status:
  loadBalancer: {}
```

Verify if service is changed to **LoadBalancer** and also to get the **Load Balancer URL** for each application.

```
kubectl get svc -n prometheus
```

```
ubuntu@ip-172-31-81-42:~$ kubectl edit svc stable-grafana -n prometheus
service/stable-grafana edited
ubuntu@ip-172-31-81-42:~$ kubectl get svc -n prometheus
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP          PORT(S)
AGE
alertmanager-operated   ClusterIP  None         <none>            9093/TCP,9094/TCP,90
94/UDP   13m
prometheus-operated    ClusterIP  None         <none>            9090/TCP
13m
stable-grafana        LoadBalancer 10.100.186.94  ad1a1ea7902e441d797edacd585fa999-733838404.us-east-1.elb.amazonaws.com  80:30905/TCP
13m
stable-kube-prometheus-sta-alertmanager ClusterIP 10.100.254.197  <none>            9093/TCP,8080/TCP
13m
stable-kube-prometheus-sta-operator       ClusterIP 10.100.85.234  <none>            443/TCP
13m
stable-kube-prometheus-sta-prometheus   LoadBalancer 10.100.229.18   a8ee684ef50a04fa99b6fa68a6a549e7-907360140.us-east-1.elb.amazonaws.com  9090:30489/TCP,8080:
30104/TCP  13m
stable-kube-state-metrics     ClusterIP 10.100.245.2   <none>            8080/TCP
13m
stable-prometheus-node-exporter   ClusterIP 10.100.247.199  <none>            9100/TCP
13m
ubuntu@ip-172-31-81-42:~$ []
```

The screenshot shows the Prometheus web interface. At the top, there's a navigation bar with links for Prometheus, Alerts, Graph, Status, and Help. Below the navigation, there are several configuration checkboxes: 'Use local time' (unchecked), 'Enable query history' (unchecked), 'Enable autocomplete' (checked), 'Enable highlighting' (checked), and 'Enable linter' (checked). A search bar labeled 'Expression (press Shift+Enter for newlines)' is present. Below the search bar, there are two tabs: 'Table' (selected) and 'Graph'. Underneath the tabs is a dropdown menu for 'Evaluation time' with arrows for navigating through time intervals. A message 'No data queried yet' is displayed. At the bottom left, there's a blue button labeled 'Add Panel'.

Access Grafana UI in the browser

Get the URL after you run **kubectl get svc -n Prometheus** and place it in the browser.

The screenshot shows the Grafana login page. The background is dark with a purple-to-red gradient. In the center, there's a large orange and yellow sun-like logo. Below the logo, the text 'Welcome to Grafana' is displayed in a white sans-serif font. There are two input fields: 'Email or username' containing 'email or username' and 'Password' containing 'password'. To the right of the password field is a small eye icon for password visibility. At the bottom is a large blue 'Log in' button. Below the 'Log in' button, there's a link 'Forgot your password?'.

You should see the above.

You can now login with the following details:

UserName: admin

Password: prom-operator

The screenshot shows the Grafana home page. At the top, it says "Welcome to Grafana". Below that, there's a "Basic" section with a "TUTORIAL" card titled "DATA SOURCE AND DASHBOARDS" and "Grafana fundamentals". The card describes setting up Grafana and covers "Data source" and "Dashboards" steps. To the right, there are two "COMPLETE" cards: "Add your first data source" (with a database icon) and "Create your first dashboard" (with a grid icon). At the bottom left, there are links for "Dashboards", "Starred dashboards", and "Recently viewed dashboards". On the right, there's a "Latest from the blog" section with a post about "client_java 1.0.0" dated Sep 27.

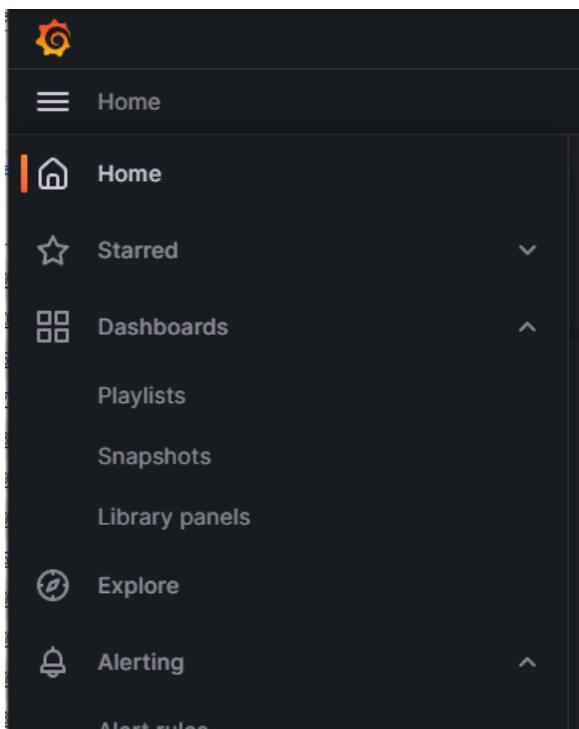
Create Dashboard in Grafana

In Grafana, we can create of various kinds of dashboards according to our needs.

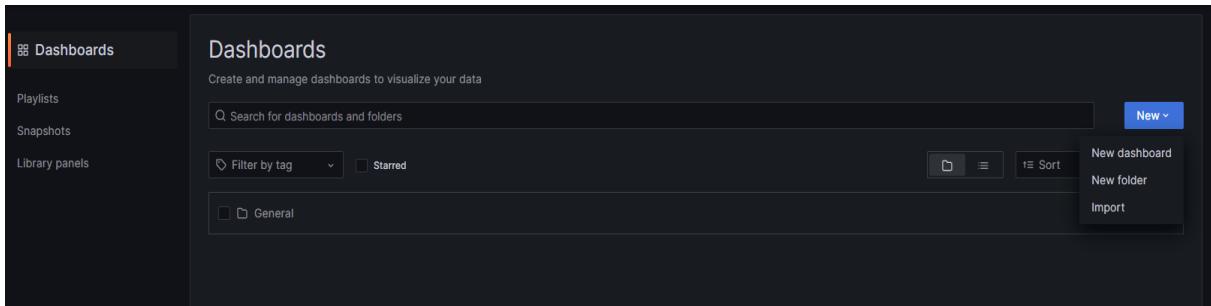
How to Create Kubernetes Monitoring Dashboard?

For creating a dashboard to monitor the cluster:

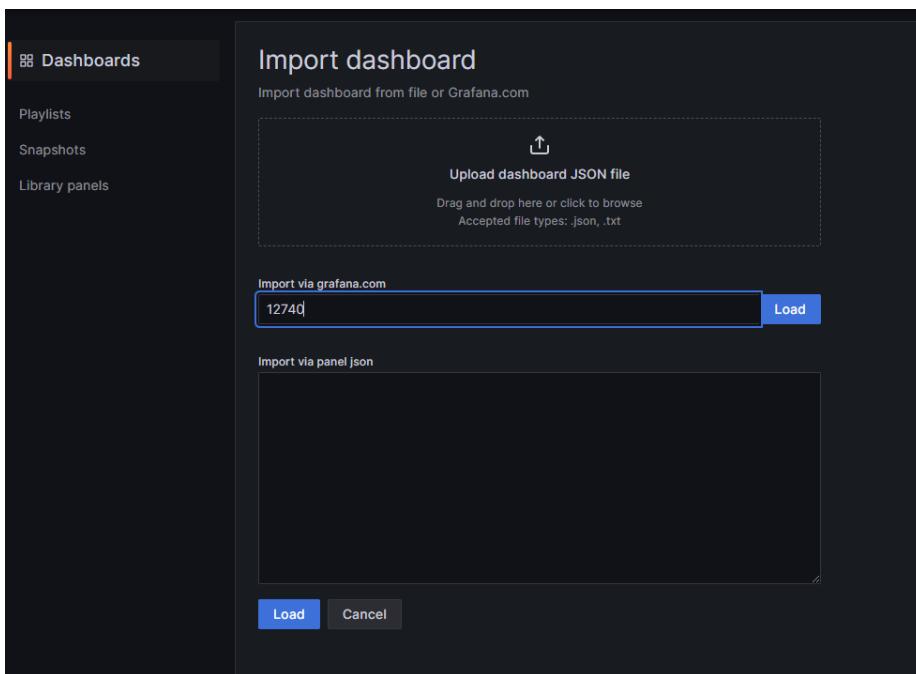
Click the hamburger at the top left than select **Dashboard**



Then on the right select **New > 'Import'**.

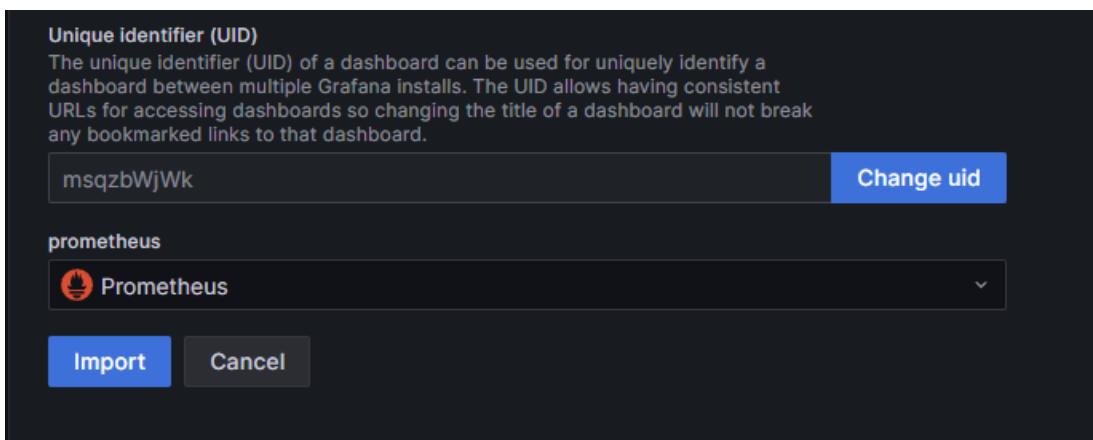


Enter **12740** dashboard id under [Grafana.com Dashboard](#).



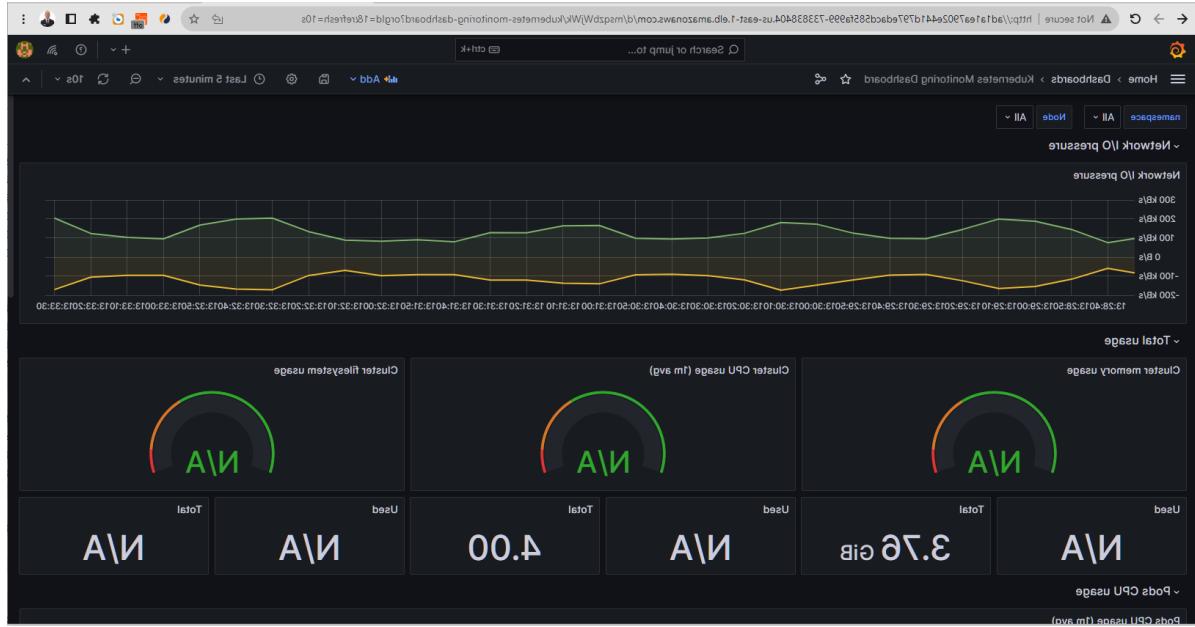
Click 'Load'.

Select 'Prometheus' as the endpoint under Prometheus data sources drop down.



Click 'Import'.

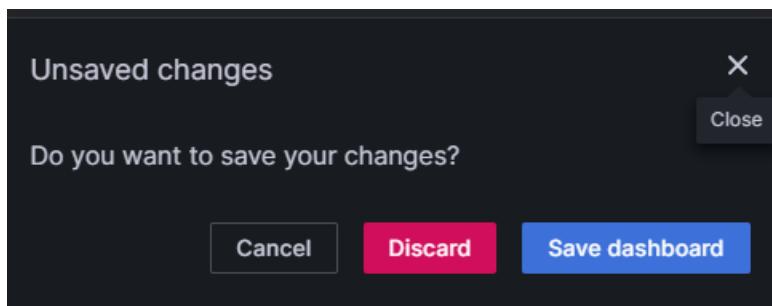
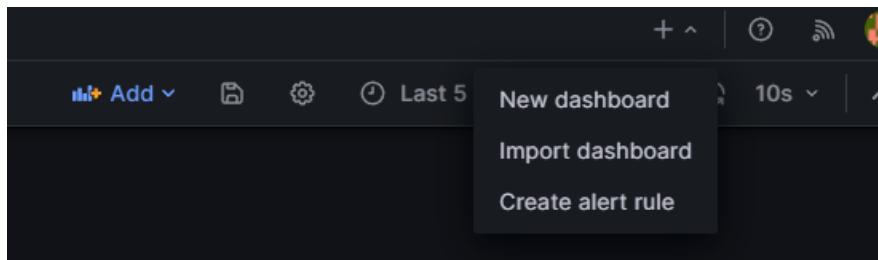
This will show the monitoring dashboard for all cluster nodes.



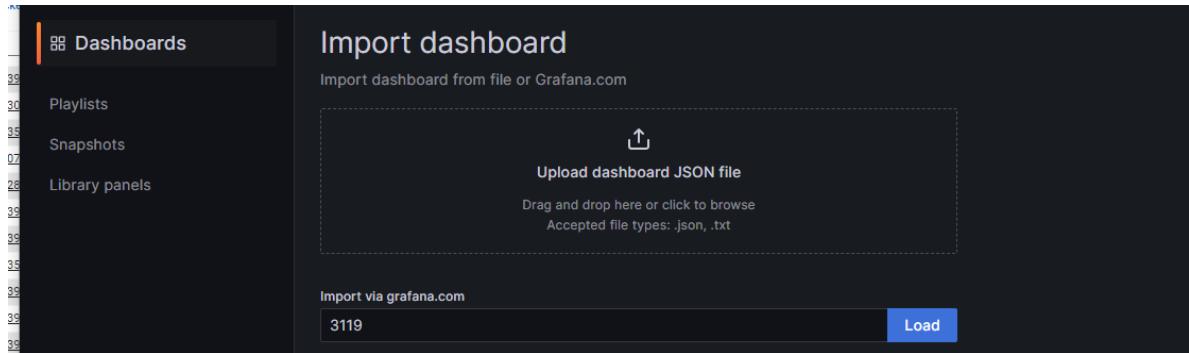
How to Create Kubernetes Cluster Monitoring Dashboard?

For creating a dashboard to monitor the cluster:

Click '+' button on the top right and select 'Import dashboard'.

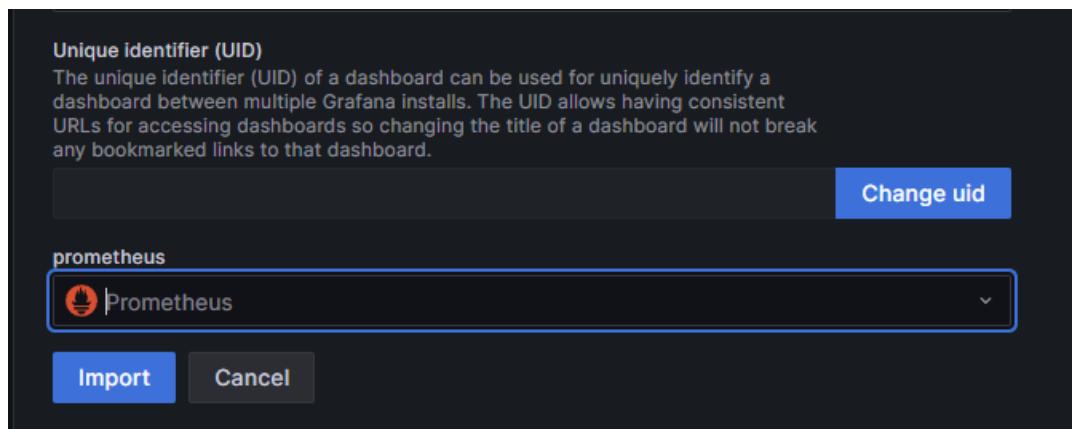


Enter **3119** dashboard id under [Grafana.com Dashboard](#).



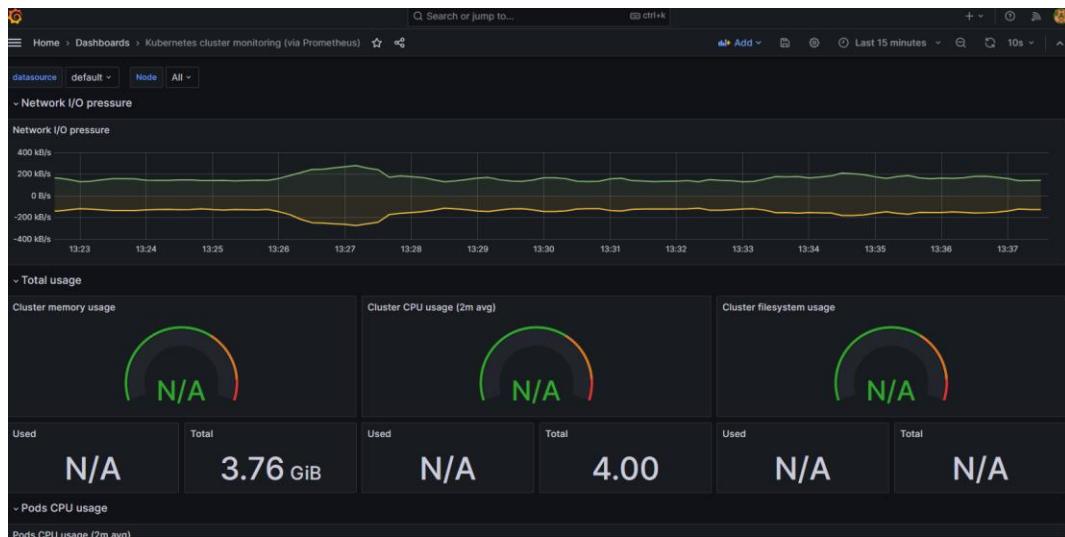
Click 'Load'.

Select 'Prometheus' as the endpoint under Prometheus data sources drop down.



Click 'Import'.

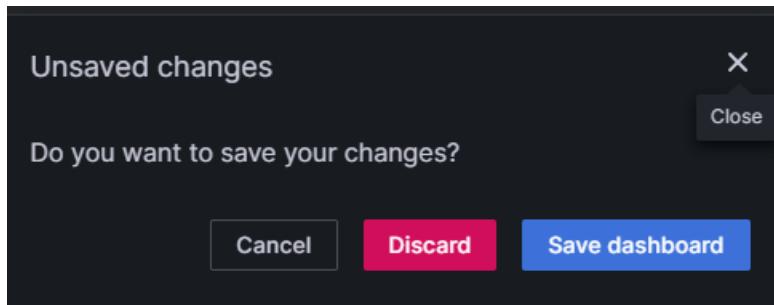
This will show a monitoring dashboard for all cluster nodes.



How to create a POD Monitoring Dashboard

For creating a dashboard to monitor the cluster:

Click '+' button on the top right panel and select 'Import dashboard'.

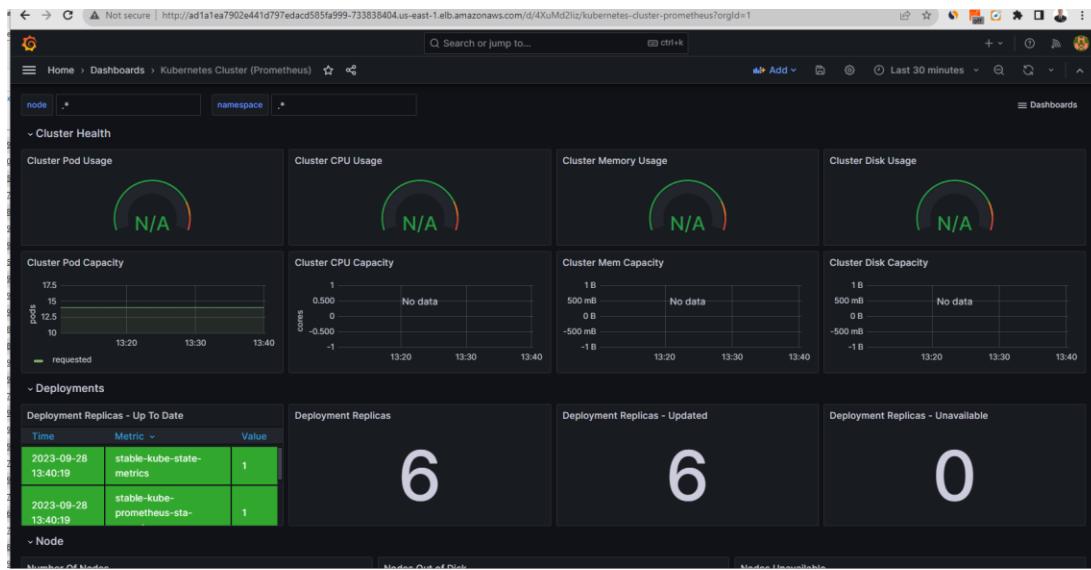


Enter **6417** dashboard id under Grafana.com **Dashboard**.

Click 'Load'.

Select '**Prometheus**' as the endpoint under Prometheus data sources drop down.
Click 'Import'.

This will show a monitoring dashboard for all cluster nodes.



Cleanup EKS Cluster

Use the below command to delete EKS cluster to avoid being charged by AWS.

eksctl delete cluster --name demo-eks --region us-east-1

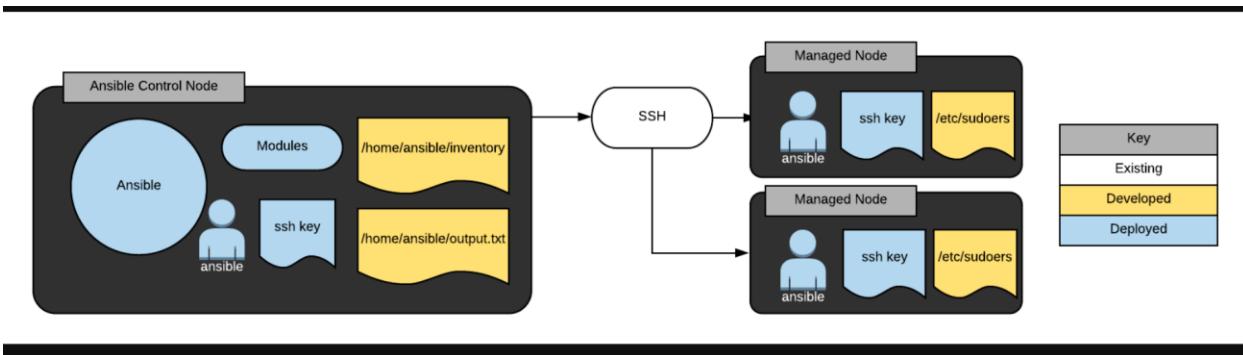
Alternatively, login to **AWS console --> AWS Cloudformation --> delete the stack manually**.

The screenshot shows the AWS CloudFormation console with the URL <https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks?filteringText=&filteringStatus=active&viewNested=true>. The page displays three stacks:

Stack name	Status	Created time	Description
eksctl-demo-eks-nodegroup-my-nodes	CREATE_COMPLETE	2023-09-28 12:05:08 UTC+0300	EKS Managed Nodes (SSH access: false) [created by eksctl]
eksctl-demo-eks-cluster	CREATE_COMPLETE	2023-09-28 11:53:06 UTC+0300	EKS cluster (dedicated VPC: true, dedicated IAM: true) [created and managed by eksctl]
cfst-1449-3026hde225e07a2c93b07e789959fa23	CREATE_COMPLETE	2023-09-28 10:44:09 UTC+0300	-

You can also delete the cluster under **AWS console --> Elastic Kubernetes Service --> Clusters**
Click on Delete cluster.

Getting Started with Ansible



Install Ansible on the Control Node:

Log in to the control node using ssh, cloud_user, and the provided public IP address and password:

```
ssh cloud_user@<PUBLIC IP>
```

To install Ansible on the control node:

```
sudo yum install ansible
```

```
[cloud_user@control1 ~]$ sudo yum install ansible
```

```
Installed:
  ansible.noarch 0:2.9.27-1.el7

Dependency Installed:
  python-cffi.x86_64 0:1.6.0-5.el7      python-enum34.noarch 0:1.0.4-1.el7      python-idna.noarch 0:2.4-1.el7      python-paramiko.noarch 0:2.1.1-9.el7
  python-ply.noarch 0:3.4-11.el7        python-pycparser.noarch 0:2.14-1.el7      python2-cryptography.x86_64 0:1.7.2-2.el7    python2-httplib2.noarch 0:0.18.1-3.el7
  python2-jmespath.noarch 0:0.9.4-2.el7  python2-pyasn1.noarch 0:0.1.9-7.el7      sshpass.x86_64 0:1.06-1.el7

Complete!
[cloud_user@control1 ~]$
```

Configure the ansible User on the Control Node:

Next, we'll configure the ansible user on the control node for ssh shared key access to managed nodes.

```
sudo su - ansible
```

```
[cloud_user@control1 ~]$ sudo su - ansible
[sudo] password for cloud_user:
[ansible@control1 ~]$ █
```

```
ssh-keygen
```

```
[ansible@control1 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):
Created directory '/home/ansible/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansible/.ssh/id_rsa.
Your public key has been saved in /home/ansible/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:hxr0J7gxHByy4BkPu3/DRKwQUNsioR6LNGEwgkGI2+Q ansible@control1
The key's randomart image is:
+---[RSA 2048]---+
|&B* . .
|B+o@ = .
|oOB + *
|=E+ = + .
|.o. . B S o
| . o * +
| . *
| ..
+---[SHA256]---+
[ansible@control1 ~]$ █
```

Note: Do not use a passphrase for the key pair.

Create a key pair for the ansible user on the control host, accepting the defaults when prompted:

Copy the public key to both node1 and node2:

```
ssh-copy-id node1
```

```
[ansible@control1 ~]$ ssh-copy-id node1
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa.pub"
The authenticity of host 'node1 (10.0.1.85)' can't be established.
ECDSA key fingerprint is SHA256:kk5TX0feYjKZFmj56uyfNhkXTKQjMhVGgC4Lpsoek2k.
ECDSA key fingerprint is MD5:40:f0:88:b0:cb:e1:4d:66:62:ee:9f:97:fd:8a:1d:2c.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': y
Please type 'yes' or 'no': yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
Password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'node1'"
and check to make sure that only the key(s) you wanted were added.

[ansible@control1 ~]$ ssh node1
[ansible@node1 ~]$ exit
logout
Connection to node1 closed.
[ansible@control1 ~]$
```

ssh-copy-id node2

```
[ansible@control1 ~]$ ssh-copy-id node2
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa.pub"
The authenticity of host 'node2 (10.0.1.16)' can't be established.
ECDSA key fingerprint is SHA256:U4aqdBHcqJtTEsvVgUMsXI+hi4Z22Fkzp+1mUKuby4w.
ECDSA key fingerprint is MD5:ed:c2:60:f5:20:83:58:1b:76:86:15:29:3a:82:f1:eb.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
Password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'node2'"
and check to make sure that only the key(s) you wanted were added.

[ansible@control1 ~]$ ssh node2
[ansible@node2 ~]$ exit
logout
Connection to node2 closed.
[ansible@control1 ~]$
```

Create a Simple Ansible Inventory

Next, we'll create a simple Ansible inventory on the control node in /home/ansible/inventory containing node1 and node2.

On the control host:

```
sudo su - ansible
touch /home/ansible/inventory
echo "node1" >> /home/ansible/inventory
```

```
echo "node2" >> /home/ansible/inventory
```

```
[ansible@control1 ~]$ touch /home/ansible/inventory
[ansible@control1 ~]$ echo "node1" >> /home/ansible/inventory
[ansible@control1 ~]$ echo "node2" >> /home/ansible/inventory
[ansible@control1 ~]$ █
```

Configure sudo Access for Ansible:

Now, we'll configure sudo access for Ansible on node1 and node2 such that Ansible may use sudo for any command with no password prompt.

Log in to node1 as cloud_user and edit the sudoers file to contain appropriate access for the ansible user:

```
[ansible@control1 ~]$ ssh cloud_user@node1
Password:
[cloud_user@node1 ~]$ sudo visudo
[sudo] password for cloud_user: █
```

ssh cloud_user@node1

sudo visudo

Add the following line to the file and save:

ansible ALL=(ALL) NOPASSWD: ALL

```
## Same thing without a password
# %wheel      ALL=(ALL)      NOPASSWD: ALL
ansible    ALL=(ALL)      NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users  ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

## Allows members of the users group to shutdown this system
# %users  localhost=/sbin/shutdown -h now

## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
-- INSERT --
```

```
[ansible@control1 ~]$ ssh cloud_user@node1
Password:
[cloud_user@node1 ~]$ sudo visudo
[sudo] password for cloud_user:
[cloud_user@node1 ~]$ exit
logout
Connection to node1 closed.
[ansible@control1 ~]$ ssh cloud_user@node2
Password:
[cloud_user@node2 ~]$ sudo visudo
[sudo] password for cloud_user:
[cloud_user@node2 ~]$ logout
Connection to node2 closed.
[ansible@control1 ~]$
```

Enter: **Logout**

Repeat these steps for node2, and then back out to the control node.

Verify Each Managed Node Is Accessible

Finally, we'll verify each managed node is able to be accessed by Ansible from the control node using the ping module.

Redirect the output of a successful command to /home/ansible/output.

To verify each node, run the following as the ansible user from the control host:

```
ansible -i /home/ansible/inventory node1 -m ping
```

```
[ansible@control1 ~]$ ansible -i /home/ansible/inventory node1 -m ping
node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[ansible@control1 ~]$ █
```

```
ansible -i /home/ansible/inventory node2 -m ping
```

```
[ansible@control1 ~]$ ansible -i /home/ansible/inventory node2 -m ping
node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[ansible@control1 ~]$ █
```

To redirect output of a successful command to /home/ansible/output:

```
ansible -i /home/ansible/inventory node1 -m ping > /home/ansible/output
```

```
[ansible@control1 ~]$ ansible -i /home/ansible/inventory node1 -m ping > /home/ansible/output
[ansible@control1 ~]$ █
```

```
[ansible@control1 ~]$ cat /home/ansible/output
node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[ansible@control1 ~]$
```

cat /home/ansible/output

Addendum

cat /etc/os-release

```
[cloud_user@control1 ~]$ cat /etc/os-release
NAME="Red Hat Enterprise Linux Server"
VERSION="7.9 (Maipo)"
ID="rhel"
ID_LIKE="fedora"
VARIANT="Server"
VARIANT_ID="server"
VERSION_ID="7.9"
PRETTY_NAME="Red Hat Enterprise Linux Server 7.9 (Maipo)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:redhat:enterprise_linux:7.9:GA:server"
HOME_URL="https://www.redhat.com/"
BUG_REPORT_URL="https://bugzilla.redhat.com/"

REDHAT_BUGZILLA_PRODUCT="Red Hat Enterprise Linux 7"
REDHAT_BUGZILLA_PRODUCT_VERSION=7.9
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux"
REDHAT_SUPPORT_PRODUCT_VERSION="7.9"
[cloud_user@control1 ~]$
```

```
cat /etc/hosts
```

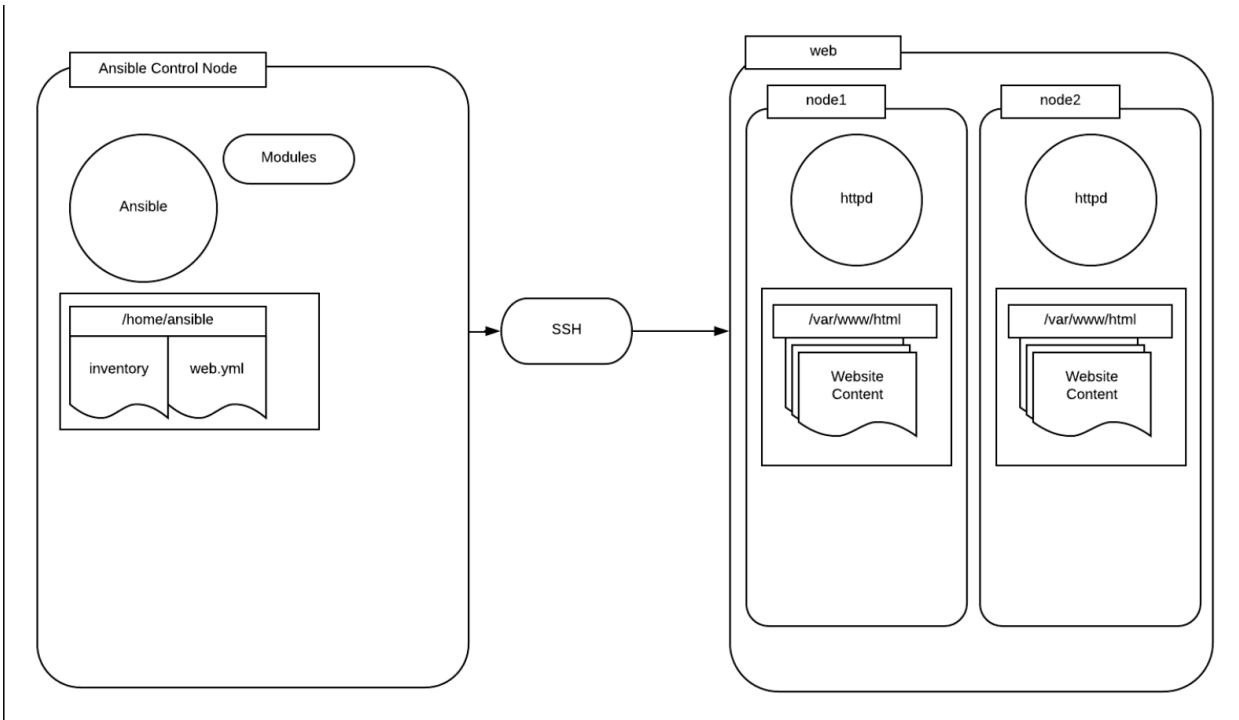
```
[cloud_user@control1 ~]$ cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6

10.0.1.85 node1
10.0.1.16 node2
[cloud_user@control1 ~]$ █
```

```
getent passwd | awk -F: '{print $1}'
```

```
[cloud_user@control1 ~]$ getent passwd | awk -F: '{print $1}'
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
operator
games
ftp
nobody
systemd-network
dbus
polkitd
sshd
postfix
chrony
ec2-user
linuxacademy
cloud_user
ansible
[cloud_user@control1 ~]$ █
```

Ansible Playbooks: The Basics



The Scenario

Our company has been increasing the deployment of small brochure-style websites for clients. The head of IT has decided that each client should have their own web server, for better client isolation, and has tasked us with creating concept automation to quickly deploy web-nodes with simple static website content.

We have been provided an Ansible control node (control1) and 2 test lab servers (node1 and node2) that have been preconfigured with the ansible user and key (see below).

We must create an Ansible inventory in **/home/ansible/inventory** containing a host group named **web**.

The **web** group should contain **node1** and **node2**.

Then we've got to design an Ansible playbook that will execute the following tasks on your configured inventory:

Install httpd

Start and enable the httpd service

Install a simple website provided on a repository server.

The playbook will be in /home/ansible/web.yml. The simple website may be accessed from <http://repo.example.com/website.tgz>.

Get Logged In

Once we're logged into the control1 server, become the ansible user (su - ansible) and we can get going.

```
[cloud_user@control1 ~]$ sudo su - ansible  
[sudo] password for cloud_user:  
[ansible@control1 ~]$ █
```

Create an inventory in /home/ansible/inventory That Contains a Host Group Named web. The web Group Should Contain node1 and node2

Use Vim to create the inventory file:

```
[ansible@control1]$ echo "[web]" >> /home/ansible/inventory  
[ansible@control1]$ echo "node1" >> /home/ansible/inventory  
[ansible@control1]$ echo "node2" >> /home/ansible/inventory
```

```
[ansible@control1 ~]$ echo "[web]" >> /home/ansible/inventory  
[ansible@control1 ~]$ echo "node1" >> /home/ansible/inventory  
[ansible@control1 ~]$ echo "node2" >> /home/ansible/inventory  
[ansible@control1 ~]$ █
```

It should contain this when we're done:

```
[web]  
node1  
node2
```

```
[ansible@control1 ~]$ cat /home/ansible/inventory  
[web]  
node1  
node2  
[ansible@control1 ~]$ █
```

Create a Playbook in /home/ansible/web.yml

```
[ansible@control1 ~]$ vi /home/ansible/web.yml  
[ansible@control1 ~]$ █
```

Using Vim, we'll create our web.yml file with these contents:

```
---  
- hosts: web  
  become: yes  
  tasks:  
    - name: install httpd  
      yum: name=httpd state=latest  
    - name: start and enable httpd  
      service: name=httpd state=started enabled=yes  
    - name: retrieve website from repo  
      get_url: url=http://repo.example.com/website.tgz dest=/tmp/website.tgz  
    - name: install website  
      unarchive: remote_src=yes src=/tmp/website.tgz dest=/var/www/html/
```

```
---  
- hosts: web  
  become: yes  
  tasks:  
    - name: install httpd  
      yum: name=httpd state=latest  
    - name: start and enable httpd  
      service: name=httpd state=started enabled=yes  
    - name: retrieve website from repo  
      get_url: url=http://repo.example.com/website.tgz dest=/tmp/website.tgz  
    - name: install website  
      unarchive: remote_src=yes src=/tmp/website.tgz dest=/var/www/html/  
~  
~  
~  
~  
~
```

Verify the Work by Executing the Playbook Using the Inventory

```
[ansible@control1]$ansible-playbook -i /home/ansible/inventory /home/ansible/web.yml
```

```
[ansible@control1 ~]$ ansible-playbook -i /home/ansible/inventory /home/ansible/web.yml

PLAY [web] ****
TASK [Gathering Facts] ****
ok: [node1]
ok: [node2]

TASK [install httpd] ****
changed: [node2]
changed: [node1]

TASK [start and enable httpd] ****
changed: [node2]
changed: [node1]

TASK [retrieve website from repo] ****
changed: [node2]
changed: [node1]

TASK [install website] ****
changed: [node2]
changed: [node1]

PLAY RECAP ****
node1                  : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node2                  : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansible@control1 ~]$
```

Addendum

cat /etc/ansible/hosts

```
[ansible@control1 ~]$ cat /etc/ansible/hosts

## db-[99:101]-node.example.com

[nodes]
node1
node2

[ansible@control1 ~]$
```

Cat /etc/hosts

```
[ansible@control1 ~]$ cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6

10.0.1.201 node1
10.0.1.45 node2
127.0.0.1      repo.example.com
[ansible@control1 ~]$ █
```

sudo visudo

```
## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
ec2-user      ALL=(ALL)      NOPASSWD: ALL
ansible       ALL=(ALL)      NOPASSWD: ALL
```

ssh node1

```
[ansible@control1 ~]$ ssh node1
Last login: Sat Sep 30 14:02:37 2023 from ip-10-0-1-120.ec2.internal
[ansible@node1 ~]$ sudo visudo
visudo: /etc/sudoers.tmp unchanged
[ansible@node1 ~]$ logout
Connection to node1 closed.
[ansible@control1 ~]$ █
```

getent passwd | awk -F: '{print \$1}'

```
[ansible@control1 ~]$ getent passwd | awk -F: '{print $1}'  
root  
bin  
daemon  
adm  
lp  
sync  
shutdown  
halt  
mail  
operator  
games  
ftp  
nobody  
systemd-network  
dbus  
polkitd  
sshd  
postfix  
chrony  
ec2-user  
linuxacademy  
cloud_user  
ansible  
apache  
[ansible@control1 ~]$ █
```

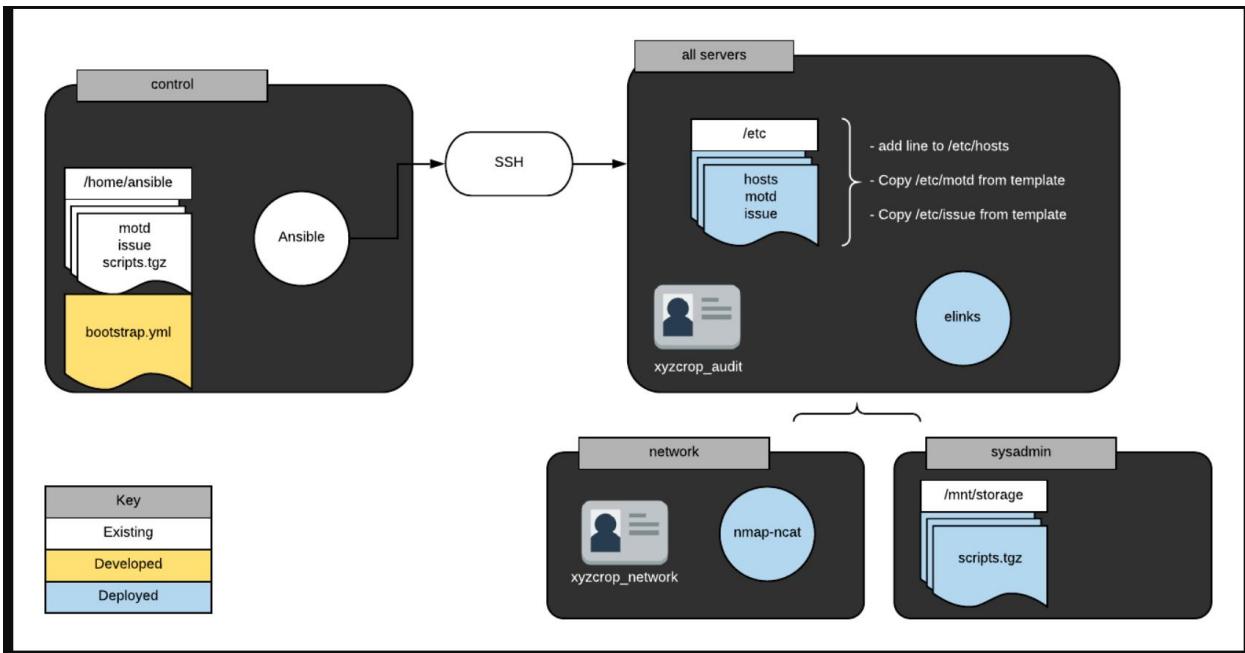
cat /etc/os-release

```
[ansible@control1 ~]$ cat /etc/os-release  
NAME="Red Hat Enterprise Linux Server"  
VERSION="7.9 (Maipo)"  
ID="rhel"  
ID_LIKE="fedora"  
VARIANT="Server"  
VARIANT_ID="server"  
VERSION_ID="7.9"  
PRETTY_NAME="Red Hat Enterprise Linux Server 7.9 (Maipo)"  
ANSI_COLOR="0;31"  
CPE_NAME="cpe:/o:redhat:enterprise_linux:7.9:GA:server"  
HOME_URL="https://www.redhat.com/"  
BUG_REPORT_URL="https://bugzilla.redhat.com/"  
  
REDHAT_BUGZILLA_PRODUCT="Red Hat Enterprise Linux 7"  
REDHAT_BUGZILLA_PRODUCT_VERSION=7.9  
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux"  
REDHAT_SUPPORT_PRODUCT_VERSION="7.9"  
[ansible@control1 ~]$ █
```

Cat ~/.ssh/id_rsa.pub | cat ~/.ssh/id_rsa

```
[ansible@control1 ~]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCe6HRoHqo1lBrIFys71sZJlgHufKghub1xUAnCEYemcYwoWT60FgDEuBaqCg4WQqU2k0DqEjdCtop9vbrhlcY50NbhwFxFsf2WH2066kJqgh+TFQhRZh+KRWlkwBPxG+IaP
8qYhoWM4XVxxTwxINExaFpmWwt/8SVAXN0rU/qBa0LtREq3ehz4rtsFltFZi7nU5IkklkjsofiuQq9GDBztxRrs12HKg19WEVUOy2+W4Tb2yqmK6IP+Jue187MueFcgrFR9Shm0ZxMw3x37Q/Z14CDMyPvE0/1/bt
m7JOGMr2rZYLoQxb05uKAETZaX8r+m1GfLm01 ansible@control1
[ansible@control1 ~]$ cat ~/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAnneh0bkR6B6qNZQay8cr09bGSZYB7nyoIbm9cVAJwhGHpnGM
KFk+tBYAxLgwqgo0Fk1NpNA6hI3QraKfb264ZXG0dDW4Vn10hd1h9tBupCaoIfk
xIUWYfikVpMAT8RviGj/KmIaFvj0F1ccU1sSDRF2hW6ZlsLf/E1QFzdK1P6gWtC
7URkt3oc+C7bBzbRWy510SiXCJJXibKH4rkKvRgwwc7cua7NdyoC/VhFVmIt
v1uE29sqpiuID/ibnpQezLnhXIIKxJfUoZjmctFt8d+0PZzeAgojMj7xdv5f27R5
uyThrjK9q2NC6EMW60VMCgB62W1/K/ptrNy5jpQIDAQABAoIBAQCYf0/+XzdYRYpP
HUPYoj1M8tqcDUZRySQuTyIcCcJ+zXuNHzaWvfpLVAweir/Wn1R2RVn1fV730g6s1
TEB1wR9qz5N/N/ZRzxy9HT7byqggPdzP/fatzbV/00Dp2Beka2l1qlwXblhjeh1
6YJ4lqVJpuRBvx8mWR9r9EiA89zzP83CqAD8gXb8LwDtGObwnwb6x0KP7kRKpzA
TSYeI6JTgJE8kCgH1IGxWjt+N9BFx+z5xFHYwurc1noL/eSY3y6pvWqT7g9dtHe
NnLx7P+ZKrjtlawQwUZW31wsaNKzG3TK42+MTLU1MY38qHJ8laotakC8UBRA5c71
eYukGEdRAoGBANC8spBriVA/5p1dzbMygVK9hsM0NvkErPNL+knUhFaKZZ2GfZ
YpD047a1uMAu0JgD+Ws1EfGa9NzmqW86Us+x2fnM16jA3yvaKZqjWqYPfff9vz0d
IUBGX2A70jfz6asKy9VNSOdC4QKc+pPpZAEjSKCavtH6NxgI1Fh/U5bAoGBAMKQ
GldupCTrmalkw3s5aJhv3+ZBQzd/f5tnzFzx2qXqroVdtHnmG9+T/2eORicgocVT
QTqdq3MF7Xn3PhnWvb8ZG7g9M1F4IdHNotMPE9kuJuYdkLOB4YaFx1QaeHaR/HES
T0i+PtcL0NTG4MzPuLo+TU+GvzSnnntBujaFbX/AoGBAMv983BMzgnYm7n8hzPw
C7495Q72L0jVL18Is1UT1TgBwC7w087uW7Z/Qz30aXsPCZnZRna7vCd76VLCb2
+FAKDaFUCm2YYvL70hDugP/OjuIvOYibFuSDDJ1Mac7y5dwNn/KBFKoxHXEjoOCe
58YNdd+zZONPKT1i0WuHyQF9AoGAQChf1ge8E8Wa68rnH6xQreV4kyRi0Lr5Wdu
2S4VimUisojV1QHThmekahVcgTQyRVggQFFfcmcMHRXxj/nmt5fjMw5ViEm1dqec
MGwi05g3e7xkJF7u65wlpokMg4w73IT+IxteUrdaqV4xErpKVh1pa5mFNuM0mMsy
5SX01ScCfxw4cvdfNkql5b0N1+VCmfQqDbA8SK0PC60QYAO+vNXx1GqcJojF1XX
4rkvRi0FxFil/WDXdr2ljxwJ1/3EJBjhhlhw3YHB9qSL4MtZUDBZOpm6dTRWSLa
Sqkgx2kdy0+uM0r0oSGrV7k1sovTIht6PjVcvde0BvkS3u8IqY0=
-----END RSA PRIVATE KEY-----
[ansible@control1 ~]$
```

Writing Your First Ansible Playbook



Playbooks are the heart of Ansible. They provide a means of executing numerous tasks

across any number of systems.

This exercise sets you on the path to Ansible mastery by allowing you to craft and execute an Ansible playbook.

Your supervisor has asked you to find a way to automate and audit basic system configuration for new servers in your environment. Given that Ansible is already configured at a basic level in your environment, the simplest solution would be to just write a playbook for bootstrapping your new hosts.

Create a playbook called **/home/ansible/bootstrap.yml** to fulfill the following bootstrap requirements:

All servers:

Edit /etc/hosts to include the following entry:

ansible.xyzcorp.com 169.168.0.1

Install elinks

Create the user xyzcorp_audit

Copy the files /home/ansible/motd and /home/ansible/issue to /etc/

Network servers:

Install nmap-ncat

Create the user xyzcorp_network

SysAdmin servers:

Copy /home/ansible/scripts.tgz from the control node to /mnt/storage

The Ansible control node has been configured for you and each testing server has already been configured for use with Ansible. The default inventory has been configured to include the groups network and sysadmin.

Each group includes a sample host.

Solution

Begin by logging in to the Ansible control node:

ssh cloud_user@PUBLIC_IP_ADDRESS

Become the ansible user:

sudo -i -u ansible

```
[cloud_user@control ~]$ sudo -i -u ansible  
[sudo] password for cloud_user:
```

Create a basic playbook for all servers in the Ansible inventory:

Create a basic playbook in **/home/ansible/bootstrap.yml** that completes the noted task required for all servers in the Ansible inventory.

Create the file **/home/ansible/bootstrap.yml** and add the following content:

vim /home/ansible/bootstrap.yml

```
---  
- hosts: all  
become: yes  
tasks:  
  - name: edit host file  
    lineinfile:  
      path: /etc/hosts  
      line: "ansible.xyzcorp.com 169.168.0.1"  
  - name: install elinks  
    package:  
      name: elinks  
      state: latest  
  - name: create audit user  
    user:  
      name: xyzcorp_audit  
      state: present  
  - name: update motd  
    copy:  
      src: /home/ansible/motd  
      dest: /etc/motd
```

```
- name: update issue
copy:
  src: /home/ansible/issue
  dest: /etc/issue
```

```
---
- hosts: all
  become: yes
  tasks:
    - name: edit host file
      lineinfile:
        path: /etc/hosts
        line: "ansible.xyzcorp.com 169.168.0.1"
    - name: install elinks
      package:
        name: elinks
        state: latest
    - name: create audit user
      user:
        name: xyzcorp_audit
        state: present
    - name: update motd
      copy:
        src: /home/ansible/motd
        dest: /etc/motd
    - name: update issue
      copy:
        src: /home/ansible/issue
        dest: /etc/issue
```

```
~  
~  
~  
~
```

Add a section to the playbook for the Network servers in the Ansible inventory.

Add a section to the playbook in **/home/ansible/bootstrap.yml** that completes the noted task required for the network servers in the Ansible inventory.

Create a play in the file **/home/ansible/bootstrap.yml** similar to the following:

```

---
- hosts: all
  become: yes
  tasks:
    - name: edit host file
      lineinfile:
        path: /etc/hosts
        line: "ansible.xyzcorp.com 169.168.0.1"
    - name: install elinks
      package:
        name: elinks
        state: latest
    - name: create audit user
      user:
        name: xyzcorp_audit
        state: present
    - name: update motd
      copy:
        src: /home/ansible/motd
        dest: /etc/motd
    - name: update issue
      copy:
        src: /home/ansible/issue
        dest: /etc/issue

- hosts: network
  become: yes
  tasks:
    - name: install netcat
      yum:
        name: nmap-ncat
        state: latest
    - name: create network user
      user:
        name: xyzcorp_network
        state: present
~
```

```

- hosts: network
  become: yes
  tasks:
    - name: install netcat
      yum:
        name: nmap-ncat
        state: latest
    - name: create network user
      user:
        name: xyzcorp_network
        state: present
```

Add a section to the playbook for the SysAdmin servers in the Ansible inventory.

Add a section to the playbook in **/home/ansible/bootstrap.yml** that completes the noted task required for the sysadmin servers in the Ansible inventory.

Create a play in the file **/home/ansible/bootstrap.yml** similar to the following:

```
- hosts: network
  become: yes
  tasks:
    - name: install netcat
      yum:
        name: nmap-ncat
        state: latest
    - name: create network user
      user:
        name: xyzcorp_network
        state: present

- hosts: sysadmin
  become: yes
  tasks:
    - name: copy tarball
      copy:
        src: /home/ansible/scripts.tgz
        dest: /mnt/storage/
```

-- INSERT (paste) --

```
- hosts: sysadmin
  become: yes
  tasks:
    - name: copy tarball
      copy:
        src: /home/ansible/scripts.tgz
        dest: /mnt/storage/
```

Execute playbook to verify your playbook works correctly.

Execute playbook **/home/ansible/bootstrap.yml** to verify your playbook works correctly.

Run **ansible-playbook /home/ansible/bootstrap.yml** from the control node.

```
[ansible@control ~]$ vim /home/ansible/bootstrap.yml
[ansible@control ~]$ ansible-playbook /home/ansible/bootstrap.yml
```

```
[ansible@control ~]$ ansible-playbook /home/ansible/bootstrap.yml

PLAY [all] ****
TASK [Gathering Facts] ****
ok: [node1]
ok: [localhost]
ok: [node2]

TASK [edit host file] ****
changed: [localhost]
changed: [node1]
changed: [node2]

TASK [install elinks] ****
changed: [localhost]
changed: [node1]
changed: [node2]

TASK [create audit user] ****
changed: [node1]
changed: [node2]
changed: [localhost]

TASK [update motd] ****
changed: [localhost]
changed: [node1]
changed: [node2]

TASK [update issue] ****
changed: [node1]
changed: [localhost]
changed: [node2]

PLAY [network] ****
TASK [Gathering Facts] ****

ok: [node1]

TASK [install netcat] ****
changed: [node1]

TASK [create network user] ****
changed: [node1]

PLAY [sysadmin] ****
TASK [Gathering Facts] ****
ok: [node2]

TASK [copy tarball] ****
changed: [node2]

PLAY RECAP ****
localhost          : ok=6    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node1              : ok=9    changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node2              : ok=8    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansible@control ~]$
```

Addendum

cat /etc/hosts

```
[ansible@control ~]$ cat /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.1.92 node1
10.0.1.185 node2
[ansible@control ~]$ █
```

cat /etc/os-release

```
[ansible@control ~]$ cat /etc/os-release
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"
[ansible@control ~]$ █
```

sudo visudo

```
## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
cloud_user ALL=(ALL)    ALL
ansible      ALL=(ALL)      NOPASSWD: ALL
```

sudo whoami

```
[ansible@control ~]$ sudo whoami
root
[ansible@control ~]$ █
```

cat /etc/ansible/hosts

or / and

cat /home/ansible/inventory

```
[ansible@control ~]$ vim /home/ansible/bootstrap.yml
[ansible@control ~]$ ansible-playbook /home/ansible/bootstrap.yml
ERROR! We were unable to read either as JSON nor YAML, these are the errors we got from each:
JSON: No JSON object could be decoded

Syntax Error while loading YAML.
  mapping values are not allowed in this context

The error appears to be in '/home/ansible/bootstrap.yml': line 2, column 9, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

---
- hosts: all
  ^ here
[ansible@control ~]$ █
```

```
---  
- hosts: all  
  become: yes  
  tasks:  
    - name: edit host file
```

How did I troubleshoot the above

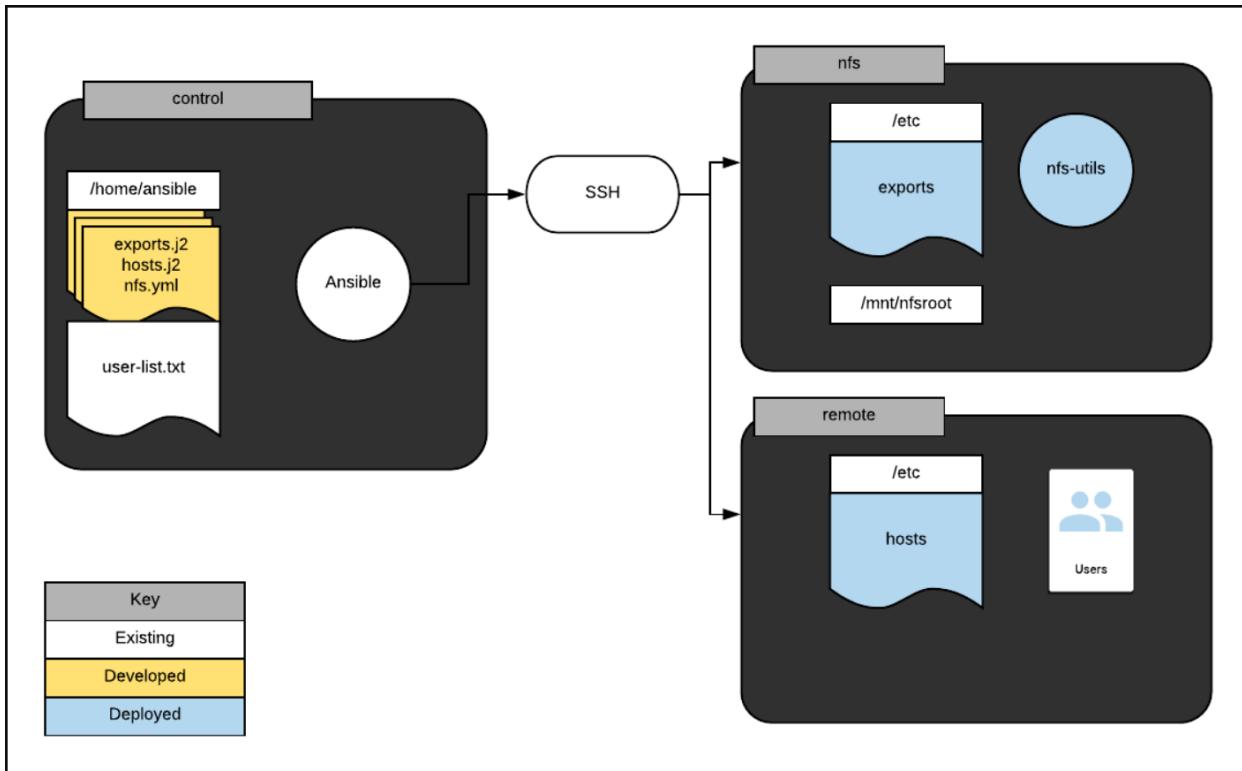
How did I solve the above?

First, I deleted the `- hosts: all` line and copy and pasted it again!

When that failed, I deleted the 3 dashes above `---` and copy and pasted them again!

One key indicator for the `---` having the wrong syntax was the colour (It was light white Rather than a dull yellow).

Deploying Services Using Ansible



Introduction

The team has been working on automating the builds for common services using Ansible. You have been tasked with creating the build for a common NFS server. Write a playbook that satisfies the following requirements:

On nfs:

Make sure `nfs-utils` is installed.

Configure `/etc/exports` via an Ansible template stored in `/home/ansible/exports.j2`.

Deploy the template so that `/mnt/nfsroot` is exported with read and write to all hosts.

Note: Your template file should contain the following with the variable `{{ share_path }}` being defined within your playbook:

`{{ share_path }} *(rw)`

Note: The file `/etc/exports` on nfs should have the following content once deployed:

/mnt/nfsroot *(rw)

Create a handler that runs the command `export fs -a` if the file **/etc/exports** is modified in a playbook task.

You can assume all necessary firewall rules have been deployed.

On remote:

Configure **/etc/hosts** from a template file stored on control at **/home/ansible/etc.hosts.j2** with the following entries:

```
127.0.0.1 localhost,{{ ansible_hostname }}  
{{ nfs_ip }} {{ nfs_hostname }}
```

Note: You should populate the variables `{{ nfs_ip }}` and `{{ nfs_hostname }}` using magic variables in your playbook.

Create users from file stored on control at **/home/ansible/user-list.txt** only if the remote host has the file **/opt/user-agreement.txt**.

The Ansible control node has been configured for you and all other servers have already been configured for use with Ansible. The default inventory has been configured to include the group `remote` and server `nfs`.

Solution

Begin in by logging in to the Ansible control node as the `ansible` user.

ssh ansible@IP_ADDRESS

Create the necessary template files on the Ansible control node.

Create the files **exports.j2** in **/home/ansible** with the noted content:

vim /home/ansible/exports.j2

Provide the following content for this file:

{{ share_path }} *(rw)

Create the file **etc.hosts.j2** in **/home/ansible** with the noted content:

vim /home/ansible/etc.hosts.j2

Provide the following content for this file:

```
127.0.0.1      localhost {{ ansible_hostname }}  
{{ nfs_ip }}      {{ nfs_hostname }}
```

Create a playbook for the server 'nfs' in the Ansible inventory.

Create a playbook in **/home/ansible/nfs.yml** that meets the provided specification for the server nfs in the ansible inventory.

Edit the /home/ansible/nfs.yml file:

```
vim /home/ansible/nfs.yml
```

Add a playbook in **/home/ansible/nfs.yml** with the following content:

```
- hosts: nfs
  become: yes
  vars:
    share_path: /mnt/nfsroot
  tasks:
    - name: install nfs
      yum:
        name: nfs-utils
        state: latest
    - name: start and enable nfs-server
      service:
        name: nfs-server
        state: started
        enabled: yes
    - name: configure exports
      template:
        src: /home/ansible/exports.j2
        dest: /etc/exports
        notify: update nfs
  handlers:
    - name: update nfs exports
      command: exportfs -a
      listen: update nfs
~
```

```
- hosts: nfs
  become: yes
  vars:
    share_path: /mnt/nfsroot
  tasks:
    - name: install nfs
      yum:
        name: nfs-utils
        state: latest
    - name: start and enable nfs-server
      service:
        name: nfs-server
        state: started
        enabled: yes
    - name: configure exports
      template:
        src: /home/ansible/exports.j2
        dest: /etc/exports
        notify: update nfs
  handlers:
    - name: update nfs exports
      command: exportfs -a
      listen: update nfs
```

```
- hosts: nfs
  become: yes
  vars:
    share_path: /mnt/nfsroot
  tasks:
    - name: install nfs
      yum:
        name: nfs-utils
        state: latest
    - name: start and enable nfs-server
      service:
        name: nfs-server
        state: started
        enabled: yes
    - name: configure exports
      template:
        src: /home/ansible/exports.j2
        dest: /etc/exports
        notify: update nfs
  handlers:
    - name: update nfs exports
      command: exportfs -a
      listen: update nfs
```

Add a play for the remote host group.

Edit **/home/ansible/nfs.yml** such that changes are executed on the remote host group to meet the provided requirements.

Open the **/home/ansible/nfs.yml** file:

```
vim /home/ansible/nfs.yml
```

Edit the **/home/ansible/nfs.yml** file to contain the following content after the nfs play:

```
- hosts: remote
  become: yes
  vars:
    nfs_ip: "{{ hostvars['nfs']['ansible_default_ipv4']['address'] }}"
    nfs_hostname: "{{ hostvars['nfs']['ansible_hostname'] }}"
  vars_files:
    - /home/ansible/user-list.txt
  tasks:
    - name: configure hostsfile
      template:
        src: /home/ansible/etc.hosts.j2
        dest: /etc/hosts
    - name: get file status
      stat:
        path: /opt/user-agreement.txt
      register: filestat
    - name: debug info
      debug:
        var: filestat
    - name: create users
      user:
        name: "{{ item }}"
      when: filestat.stat.exists
      loop: "{{ users }}"
```

```
- hosts: remote
  become: yes
  vars:
    nfs_ip: "{{ hostvars['nfs']['ansible_default_ipv4']['address'] }}"
    nfs_hostname: "{{ hostvars['nfs']['ansible_hostname'] }}"
  vars_files:
    - /home/ansible/user-list.txt
  tasks:
    - name: configure hostsfile
      template:
        src: /home/ansible/etc.hosts.j2
        dest: /etc/hosts
    - name: get file status
      stat:
        path: /opt/user-agreement.txt
      register: filestat
    - name: debug info
      debug:
```

```
var: filestat
- name: create users
  user:
    name: "{{ item }}"
  when: filestat.stat.exists
  loop: "{{ users }}"
```

```
- hosts: remote
  become: yes
  vars:
    nfs_ip: "{{ hostvars['nfs']['ansible_default_ipv4']['address'] }}"
    nfs_hostname: "{{ hostvars['nfs']['ansible_hostname'] }}"
  vars_files:
    - /home/ansible/user-list.txt
  tasks:
    - name: configure hostsfile
      template:
        src: /home/ansible/etc.hosts.j2
        dest: /etc/hosts
    - name: get file status
      stat:
        path: /opt/user-agreement.txt
      register: filestat
    - name: debug info
      debug:
        var: filestat
    - name: create users
      user:
        name: "{{ item }}"
      when: filestat.stat.exists
      loop: "{{ users }}"
-- INSERT (paste) --
```

Execute playbook to verify your playbook works correctly.

Execute playbook /home/ansible/nfs.yml to verify your playbook works correctly.

Run the command below from the control node.

ansible-playbook /home/ansible/nfs.yml

```

        "mimetype": "text/plain",
        "mode": "0644",
        "mtime": 1696046537.6733913,
        "nlink": 1,
        "path": "/opt/user-agreement.txt",
        "pw_name": "root",
        "readable": true,
        "rgrp": true,
        "roth": true,
        "rusr": true,
        "size": 34,
        "uid": 0,
        "version": "18446744071586397626",
        "wgrp": false,
        "woth": false,
        "writeable": true,
        "wusr": true,
        "xgrp": false,
        "xoth": false,
        "xusr": false
    }
}
}

TASK [create users] ****
changed: [node2] => (item=frank)
changed: [node2] => (item=judy)
changed: [node2] => (item=joe)
changed: [node2] => (item=sarah)
changed: [node2] => (item=sam)
changed: [node2] => (item=carry)

PLAY RECAP ****
nfs : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node2 : ok=5    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[ansible@control ~]$ 

```

Cat /home/ansible/user-list.txt

```

[ansible@control ~]$ cat /home/ansible/user-list.txt
users:
- frank
- judy
- joe
- sarah
- sam
- carry
[ansible@control ~]$ 

```

Addendum

```
[ansible@control ~]$ vim /home/ansible/nfs.yml
[ansible@control ~]$ ansible-playbook /home/ansible/nfs.yml
ERROR! We were unable to read either as JSON nor YAML, these are the errors we got from each:
JSON: No JSON object could be decoded

Syntax Error while loading YAML.
  did not find expected '-' indicator

The error appears to be in '/home/ansible/nfs.yml': line 10, column 6, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

    state: latest
  - name: start and enable nfs-server
    ^ here
[ansible@control ~]$ █
```

```
[ansible@control ~]$ vim /home/ansible/nfs.yml
[ansible@control ~]$ ansible-playbook /home/ansible/nfs.yml
ERROR! We were unable to read either as JSON nor YAML, these are the errors we got from each:
JSON: No JSON object could be decoded

Syntax Error while loading YAML.
  did not find expected '-' indicator

The error appears to be in '/home/ansible/nfs.yml': line 20, column 5, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

    notify: update nfs
  handlers:
    ^ here
[ansible@control ~]$ █
```

```
[ansible@control ~]$ ansible-playbook /home/ansible/nfs.yml
ERROR! We were unable to read either as JSON nor YAML, these are the errors we got from each:
JSON: No JSON object could be decoded

Syntax Error while loading YAML.
  did not find expected '-' indicator

The error appears to be in '/home/ansible/nfs.yml': line 25, column 2, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

- hosts: remote
^ here
[ansible@control ~]$
```

```
- hosts: nfs
  become: yes
  vars:
    share_path: /mnt/nfsroot
  tasks:
    - name: install nfs
      yum:
        name: nfs-utils
        state: latest
    - name: start and enable nfs-server
      service:
        name: nfs-server
        state: started
        enabled: yes
    - name: configure exports
      template:
        src: /home/ansible/exports.j2
        dest: /etc/exports
        notify: update nfs
  handlers:
    - name: update nfs exports
      command: exportfs -a
      listen: update nfs
```

```

- hosts: nfs
  become: yes
  vars:
    share_path: /mnt/nfsroot
  tasks:
    - name: install nfs
      yum:
        name: nfs-utils
        state: latest
    - name: start and enable nfs-server
      service:
        name: nfs-server
        state: started
        enabled: yes
    - name: configure exports
      template:
        src: /home/ansible/exports.j2
        dest: /etc/exports
      notify: update nfs
  handlers:
    - name: update nfs exports
      command: exportfs -a
      listen: update nfs

- hosts: remote
  become: yes
  vars:
    nfs_ip: "{{ hostvars['nfs']['ansible_default_ipv4']['address'] }}"
    nfs_hostname: "{{ hostvars['nfs']['ansible_hostname'] }}"
  vars_files:
    - /home/ansible/user-list.txt
  tasks:
    - name: configure hostsfile
      template:
        src: /home/ansible/etc.hosts.j2
        dest: /etc/hosts
-- INSERT --

```

The errors above were generated by syntax errors. Unfortunately, YAML is not agnostic about text not aligning like Terraform!

cat /etc/os-release

```

[ansible@control ~]$ cat /etc/os-release
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"

[ansible@control ~]$ 

```

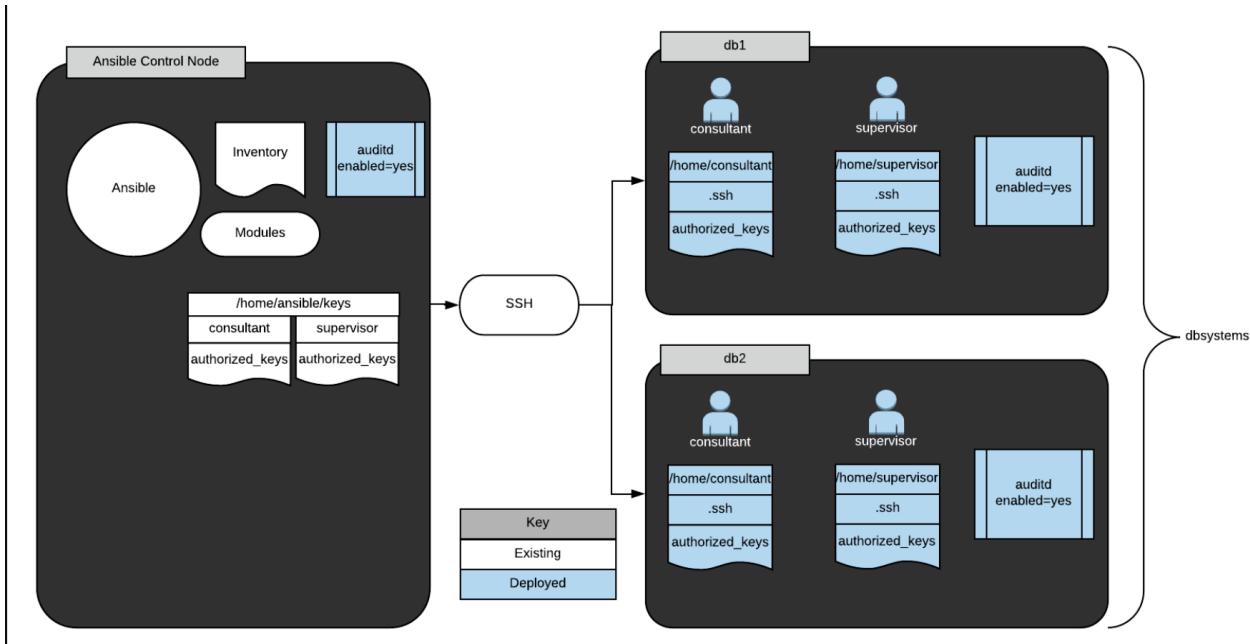
Sudo cat /etc/hosts (hosts is similar to the host.ini in windows)

```
[ansible@control ~]$ sudo cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.1.53 nfs node1
10.0.1.211 node2
[ansible@control ~]$
```

cat /etc/ansible/hosts

```
localhost
nfs
[remote]
node2
[ansible@control ~]$
```

Ad-Hoc Ansible Commands



The Scenario

Some consultants will be performing audits on a number of systems in our company's environment. We've got to create the user accounts listed in **/home/ansible/userlist.txt** and set up the provided public keys for their accounts. The security team has built a jump host for the consultants to access production systems and provided us with the full key-pair so we can set up and test the connection.

All hosts in dbsystems will need that public key installed so the consultants may use key-pair authentication to access the systems. We must also ensure the **auditd service** is enabled and running on all systems.

Important notes:

Ansible is already on the control node. If we connect to the server by clicking on the Public IP address in a web browser, we need to make sure we change to the ansible user, with the **sudo su - ansible** command.

The user ansible is present on all servers with appropriate shared keys for access to managed servers from the control node. We need to make sure we use this user to complete the commands.

The ansible user has the same password as cloud_user.

The default Ansible inventory has already been configured with the appropriate hosts and groups.

/etc/hosts entries are present on the control1 host for the managed servers.

Get Logged In

Once we're logged into the control1 server, become the ansible user (**su - ansible**) and we can get going.

```
[cloud_user@control1 ~]$ sudo su - ansible  
[sudo] password for cloud_user:  
[ansible@control1 ~]$ █
```

Create the User Accounts Noted in /home/ansible/userlist.txt

If we read the **userlist.txt** file in our home directory, we'll see consultant and supervisor.

```
[ansible@control1 ~]$ cat /home/ansible/userlist.txt  
# Please create the the below users on all hosts in dbsystems  
# You will find a pregenerated key pair and an authorized key file for each user in /home/ansible/keys  
  
# User accounts to create:  
consultant  
supervisor  
[ansible@control1 ~]$ █
```

Those are the two new user accounts we have to create:

```
[ansible@control1]$ ansible dbsystems -b -m user -a "name=consultant"
```

```
[ansible@control1 ~]$ ansible dbsystems -b -m user -a "name=consultant"  
db1 | CHANGED => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/bin/python"  
    },  
    "changed": true,  
    "comment": "",  
    "create_home": true,  
    "group": 1004,  
    "home": "/home/consultant",  
    "name": "consultant",  
    "shell": "/bin/bash",  
    "state": "present",  
    "system": false,  
    "uid": 1004  
}  
db2 | CHANGED => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/bin/python"  
    },  
    "changed": true,  
    "comment": "",  
    "create_home": true,  
    "group": 1004,  
    "home": "/home/consultant",  
    "name": "consultant",  
    "shell": "/bin/bash",  
    "state": "present",  
    "system": false,  
    "uid": 1004  
}  
[ansible@control1 ~]$ █
```

```
[ansible@control1] $ ansible dbsystems -b -m user -a "name=supervisor"
```

```
[ansible@control1 ~]$ ansible dbsystems -b -m user -a "name=supervisor"
db1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 1005,
    "home": "/home/supervisor",
    "name": "supervisor",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1005
}
db2 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 1005,
    "home": "/home/supervisor",
    "name": "supervisor",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1005
}
[ansible@control1 ~]$ 
[ansible@control1 ~]$ █
```

Place Key Files in the Correct Location, /home/\$USER/.ssh/authorized_keys, on Hosts in dbsystems

```
[ansible@control1] $ ansible dbsystems -b -m file -a "path=/home/consultant/.ssh state=directory owner=consultant group=consultant mode=0755"
```

```
[ansible@control1 ~]$ ansible dbsystems -b -m file -a "path=/home/consultant/.ssh state=directory owner=consultant group=consultant mode=0755"
db1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "gid": 1004,
    "group": "consultant",
    "mode": "0755",
    "owner": "consultant",
    "path": "/home/consultant/.ssh",
    "secontext": "unconfined_u:object_r:ssh_home_t:s0",
    "size": 6,
    "state": "directory",
    "uid": 1004
}
db2 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "gid": 1004,
    "group": "consultant",
    "mode": "0755",
    "owner": "consultant",
    "path": "/home/consultant/.ssh",
    "secontext": "unconfined_u:object_r:ssh_home_t:s0",
    "size": 6,
    "state": "directory",
    "uid": 1004
}
[ansible@control1 ~]$ █
```

```
[ansible@control1]$ ansible dbsystems -b -m copy -a
"src=/home/ansible/keys/consultant/authorized_keys dest=/home/consultant/.ssh/authorized_keys
mode=0600 owner=consultant group=consultant"
```

```
[ansible@control1 ~]$ ansible dbsystems -b -m copy -a "src=/home/ansible/keys/consultant/authorized_keys dest=/home/consultant/.ssh/authorized_keys mode=0600 owner=co
nsultant group=consultant"
db1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "checksum": "a249d6a58d39b67f331b37c2ae8c24af42faab7",
    "dest": "/home/consultant/.ssh/authorized_keys",
    "gid": 1004,
    "group": "consultant",
    "md5sum": "6478a0a7abb698adbb0b9c6cb81bcb4f",
    "mode": "0600",
    "owner": "consultant",
    "secontext": "unconfined u:object_r:ssh_home_t:s0",
    "size": 395,
    "src": "/home/ansible/.ansible/tmp/ansible-tmp-1696086040.98-1764-124376452151507/source",
    "state": "file",
    "uid": 1004
}
db2 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "checksum": "a249d6a58d39b67f331b37c2ae8c24af42faab7",
    "dest": "/home/consultant/.ssh/authorized_keys",
    "gid": 1004,
    "group": "consultant",
    "md5sum": "6478a0a7abb698adbb0b9c6cb81bcb4f",
    "mode": "0600",
    "owner": "consultant",
    "secontext": "unconfined u:object_r:ssh_home_t:s0",
    "size": 395,
    "src": "/home/ansible/.ansible/tmp/ansible-tmp-1696086041.03-1766-231417013313828/source",
    "state": "file",
    "uid": 1004
}
[ansible@control1 ~]$
```

```
[ansible@control1]$ ansible dbsystems -b -m file -a "path=/home/supervisor/.ssh state=directory
owner=supervisor group=supervisor mode=0755"
```

```
[ansible@control1]$ ansible dbsystems -b -m copy -a
"src=/home/ansible/keys/supervisor/authorized_keys dest=/home/supervisor/.ssh/authorized_keys
mode=0600 owner=supervisor group=supervisor"
```

Ensure auditd Is Enabled and Running on All Hosts

```
[ansible@control1]$ ansible all -b -m service -a "name=auditd state=started enabled=yes"
```

```
[ansible@control1 ~]$ ansible all -b -m service -a "name=auditd state=started enabled=yes"
```

Addendum

getent passwd | awk -F: '{print \$1}'

```
[ansible@control1 ~]$ getent passwd | awk -F: '{print $1}'  
root  
bin  
daemon  
adm  
lp  
sync  
shutdown  
halt  
mail  
operator  
games  
ftp  
nobody  
systemd-network  
dbus  
polkitd  
sshd  
postfix  
chrony  
ec2-user  
linuxacademy  
cloud_user  
ansible  
[ansible@control1 ~]$ █
```

cat /etc/hosts

```
[ansible@control1 ~]$ cat /etc/hosts  
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
  
10.0.1.224 db1 node1  
10.0.1.229 db2 node2  
[ansible@control1 ~]$ █
```

sudo visudo

```
## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)  
#includedir /etc/sudoers.d  
ec2-user    ALL=(ALL)      NOPASSWD: ALL  
ansible     ALL=(ALL)      NOPASSWD: ALL
```

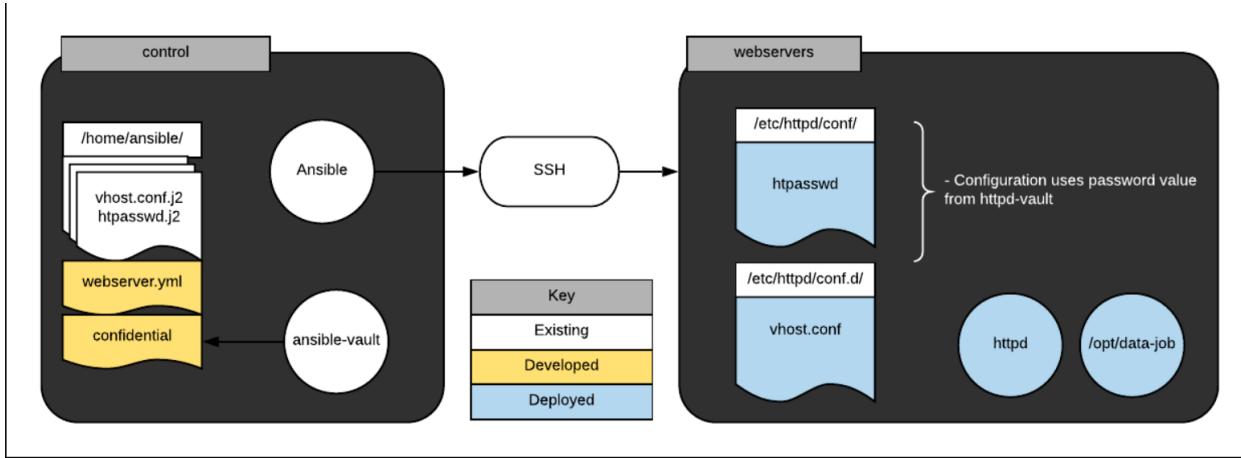
ssh node1

```
[ansible@control1 ~]$ ssh node1
Warning: Permanently added 'node1' (ECDSA) to the list of known hosts.
[ansible@db1 ~]$ ls
[ansible@db1 ~]$ sudo visudo
visudo: /etc/sudoers.tmp unchanged
[ansible@db1 ~]$ logout
Connection to node1 closed.
[ansible@control1 ~]$
```

cat ~/ssd/id_rsa.pub | cat ~/ssh/id_rsa

```
[ansible@control1 ~]$ cat ~/ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQABAAAQAB0Q1BuUhXqZjZgPv0+QwD/GNzOCB8mWGS/OTc1bPVVmQC6kZ+e8KXsx1dAZPUyPMySQCX+ZEQmJxZEMUsK3SFHpjLEQKhZjH8R9u0DxOodJ+IIRFKz/SY2AKPe/60KF
+QlXSiysb507iMh3TDrx5zXU53Xcu7Gyz13SE7+Y10g+ppDIVt3ZV0mn91bt4lh5st1mrfXT2x07LGi19cU4u9/4UZAhlba/QttxKxhcY5URn0G5QHmw7BoYoEtKyQoX5SCvih9xIfaZF/zvXtGv3tSmIH3AaFD
adun1HE3XTMbggDCgv0AlFuthiti6w6waWJpWf5C4K9yAb ansible@control1
[ansible@control1 ~]$ cat ~/ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpqIBAAKCAQEa5QVIVVmSWYD70PwUMA/xjczzggfJ1hfzk3JWz1V2kAupGfn
vC17MZXQGT1MqTM-KgI/mREj3iV3GRDPLrCt0nx6YyxECh8yR/EfbtA8TqHsfIJUR
ss+HNc3jv+j1hfkC1+crG7N051jd07JQ68ec11Ef7DMXLixess4t0h0/MjdGo
PqaQyFBsWVdjzpPZW7eJyebLdzk109sd0xyxpXF0Lvf+FGBQ822v/0LbbyxYG0
VEZ9BrEB5s0waGKBLs5kkF0kgr40fcSH2mRF87Rv97Up1b9wABQ2nbp9RxN10z
CWAIAwoK1dA3RVLYZbyusOr2liT8H+QuCvcgGwIDAQABAoIBAQCKGLtia05ds/Pr
bMusja0nB5Y5jDSUHsgeoGIQcFV0iLMXU8ggf9DXP3zyGzKEvGQEvyr/fzPQxLqi
X65en-P1vJmWY3TLnbAMDWFK0o/q4k0o3cvvuuueh7u7sU1plwoWtrKEYy+Ubpb39
epiZEZfxmHlevAay7PHgwvlw/mBeX53XXYK810k8Lk479yhK110ZHU/UBeyp1RP
ekxD53egm3fCndpYdTITCj1VWhWkEPwMMyY2qQZy2C2DhbNlWQBFevL7SgT
X80EcDx0p10iyM/hIMwdtYRLB8sL2Tu0hhHK1PHzYRY093jZfi0MpuesCsTp395sG
FtN2VU3hAoGAP5SvhIPoj0QunXltuOnZ21a4zvH9U/C4QRaTaZirRNap0kmlCxZ
wT7ddQa05v5MSME614ggz9neC0xbhjqE9N4c33xUjeFaAFz6K1c1DHMu4+sf
ou9pJHkQfcg86970ocZbVF12uiujoKfbvUQydTQlmn0M4o6ppQvV34JAoGBAOZr
vzwz5rXgVRmOfgcDCRs0g01b16PPq41XGcmr50STPskrUv1PSV1t8VQEKY1JzRz
175Y-BF+yX1WtjhPH2NBd0Z5c1Wk85rBwEtswFnwFD1/19w0+5G5sK6WEFFXii
h1DvjhCFCyrlrDScAXtsdk3msAkX+RIPoI69yJYDAgGACb+z558/uRU9/6xogp73
PTOMvdBhMjUL0k42oydW5gvHw89y0k2FTf0h1h1LTf1NnAaMPQ5Yz4X1/WMc
gkXCP0CdPDmtsdsnAi7JhdKqtbzC8Ux1dryge0Npc+LnjfjE6nMCfprqioJ0Xirqw
Enx9LzE6JXbb1ly6BEHKCgYEaju1SpnGvrg1EAYjPDNDMj5C31gmaxMh
6442hsKpBhwS23A6g6YLyLxsLMs4bGTFU9wvTfNcCX+KdAQTxvaQ909pfWr
+ob1Su00l+rX0to+uEMZf2+D4DYtjHe60ZMZVsRkt1G5xhLw3uMVLjBhaN2F1
7C35bZEcgYE8fL1atxR729scH8e/cpJFPknRE1tRnt/Q1udVrHqfn/fSptb
ndKC/tyc+ZnAwBx1czEzhi6V5r+S49iYqhs9j0H0p2Y1TF8GKEn0x1ATA5ky5k
u0Z6LaKTdSzGaRM1khbeBiCmLovaIipqh5ialzz8Fr0jxPGuVIP0BGhk=
-----END RSA PRIVATE KEY-----
[ansible@control1 ~]$
```

Advanced Features in Ansible Playbooks



Introduction

There are a number of features unique to Ansible playbooks which provide robust functionality. This exercise explores many of these features in a practical scenario of deploying a web server. Most notably, this exercise deals with confidential data in an Ansible vault and working with tags in Ansible playbooks.

You must create a modular playbook used for webserver management.

Create a playbook called **/home/ansible/webserver.yml** that meets the following requirements:

On the host group webservers:

Deploy `httpd`.

You can assume all necessary firewall rules have been deployed.

Start and enable `httpd`.

Configure virtual host using the provided templates in **/home/ansible/vhost.conf.j2** and **/home/ansible/htpasswd.j2**.

Note: The template references a variable defined in `/home/ansible/confidential` which must be included as a variable file in your playbook.

Use ansible vault to secure `/home/ansible/confidential` with password "**I love ansible**".

Run the data job stored in **/opt/data-job.sh** on each node in `webservers` asynchronously, without polling for status.

Create tags for the following tasks:

base-install for `httpd` installation and service configuration.

vhost for virtual host deployment.

data-job to execute the asynchronous data job.

Solution

Begin by logging in to the Ansible control node.

```
ssh cloud_user@PUBLIC_IP_ADDRESS
```

Become the ansible user:

```
sudo -i -u ansible
```

```
[cloud_user@control ~]$ sudo -i -u ansible  
[sudo] password for cloud_user:  
[ansible@control ~]$ █
```

Use ansible-vault to protect the confidential information

Use ansible-vault to encrypt /home/ansible/confidential to protect the confidential information stored within using the password "**I love ansible**".

Run **ansible-vault encrypt /home/ansible/confidential** and supply the password "**I love ansible**".

```
[ansible@control ~]$ ansible-vault encrypt /home/ansible/confidential  
New Vault password:  
Confirm New Vault password:  
Encryption successful  
[ansible@control ~]$ █
```

Create a playbook that deploys httpd on webservers.

Create a playbook in **/home/ansible/webserver.yml** that deploys httpd on webservers. It should be tagged with base-install and contain a handler that restarts the httpd daemon that is flagged by both installation and service manipulation for httpd.

Create the file **/home/ansible/webserver.yml** and add the following content:

```
vim /home/ansible/webserver.yml
```

```
---  
- hosts: webservers  
become: yes  
vars_files:  
  - /home/ansible/confidential  
tasks:  
  - name: install httpd  
    yum:
```

```

name: httpd
state: latest
notify: httpd service
tags:
  - base-install
handlers:
  - name: Restart and enable httpd
    service:
      name: httpd
      state: restarted
      enabled: yes
      listen: httpd service

```

```

---  

- hosts: webservers  

become: yes  

vars_files:  

  - /home/ansible/confidential  

tasks:  

  - name: install httpd  

    yum:  

      name: httpd  

      state: latest  

      notify: httpd service  

      tags:  

        - base-install  

handlers:  

  - name: Restart and enable httpd  

    service:  

      name: httpd  

      state: restarted  

      enabled: yes  

      listen: httpd service

```

Deploy the templates stored on the control node to the webservers group.

Configure /home/ansible/webserver.yml to deploy the templates **/home/ansible/vhost.conf.j2** and **/home/ansible/htpasswd.j2** stored on the control node to the webservers group. httpd must restart on config change. The tasks should be tagged vhost.

Add the following text to /home/ansible/webserver.yml just before the handler section:
vim /home/ansible/webserver.yml

```

- name: configure virtual host
  template:
    src: /home/ansible/vhost.conf.j2
    dest: /etc/httpd/conf.d/vhost.conf
  notify: httpd service
  tags:

```

```

- vhost
- name: configure site auth
  template:
    src: /home/ansible/htpasswd.j2
    dest: /etc/httpd/conf/htpasswd
  notify: httpd service
  tags:
    - vhost

```

```

---
- hosts: webservers
  become: yes
  vars_files:
    - /home/ansible/confidential
  tasks:
    - name: install httpd
      yum:
        name: httpd
        state: latest
      notify: httpd service
      tags:
        - base-install
    - name: configure virtual host
      template:
        src: /home/ansible/vhost.conf.j2
        dest: /etc/httpd/conf.d/vhost.conf
      notify: httpd service
      tags:
        - vhost
    - name: configure site auth
      template:
        src: /home/ansible/htpasswd.j2
        dest: /etc/httpd/conf/htpasswd
      notify: httpd service
      tags:
        - vhost
  handlers:
    - name: Restart and enable httpd
      service:
        name: httpd
        state: restarted
        enabled: yes
      listen: httpd service
~
~
-- INSERT (paste) --

```

Asynchronously execute data-job on webservers

Configure **/home/ansible/webserver.yml** to asynchronously execute **/opt/data-job.sh** located on webservers with a timeout of 600 seconds and no polling. The task should be tagged with **data-job**.

Add the following text to **/home/ansible/webserver.yml** just before the handler section:
vim /home/ansible/webserver.yml

```
- name: run data job
  command: /opt/data-job.sh
  async: 600
  poll: 0
  tags:
    - data-job
```

Your complete file should look similar to the below:

```
---
- hosts: webservers
  become: yes
  vars_files:
    - /home/ansible/confidential
  tasks:
    - name: install httpd
      yum:
        name: httpd
        state: latest
      notify: httpd service
    tags:
      - base-install
    - name: configure virtual host
      template:
        src: /home/ansible/vhost.conf.j2
        dest: /etc/httpd/conf.d/vhost.conf
      notify: httpd service
    tags:
      - vhost
    - name: configure site auth
      template:
        src: /home/ansible/htpasswd.j2
        dest: /etc/httpd/conf/htpasswd
      notify: httpd service
    tags:
      - vhost
    - name: run data job
      command: /opt/data-job.sh
      async: 600
      poll: 0
    tags:
      - data-job
  handlers:
    - name: Restart and enable httpd
      service:
        name: httpd
        state: restarted
        enabled: yes
```

listen: httpd service

```
---
- hosts: webservers
  become: yes
  vars_files:
- hosts: webservers
  become: yes
  vars_files:
    - /home/ansible/confidential
  tasks:
    - name: install httpd
      yum:
        name: httpd
        state: latest
        notify: httpd service
      tags:
        - base-install
    - name: configure virtual host
      template:
        src: /home/ansible/vhost.conf.j2
        dest: /etc/httpd/conf.d/vhost.conf
        notify: httpd service
      tags:
        - vhost
    - name: configure site auth
      template:
        src: /home/ansible/htpasswd.j2
        dest: /etc/httpd/conf/htpasswd
        notify: httpd service
      tags:
        - vhost
    - name: run data job
      command: /opt/data-job.sh
      async: 600
      poll: 0
      tags:
        - data-job
  handlers:
    - name: Restart and enable httpd
      service:
        name: httpd
        state: restarted
        enabled: yes
      listen: httpd service
~
~
```

[Execute playbook to verify your playbook works correctly](#)

Execute playbook /home/ansible/webserver.yml to verify your playbook works correctly.

Run `ansible-playbook --ask-vault-pass /home/ansible/webserver.yml` from the control node and provide the vault password "I love ansible".

```
[ansible@control ~]$ ansible-playbook --ask-vault-pass /home/ansible/webserver.yml
Vault password:

PLAY [webservers] *****
TASK [Gathering Facts] *****
ok: [node2]
ok: [node1]

TASK [install httpd] *****
changed: [node2]
changed: [node1]

TASK [configure virtual host] *****
changed: [node1]
changed: [node2]

TASK [configure site auth] *****
changed: [node2]
changed: [node1]

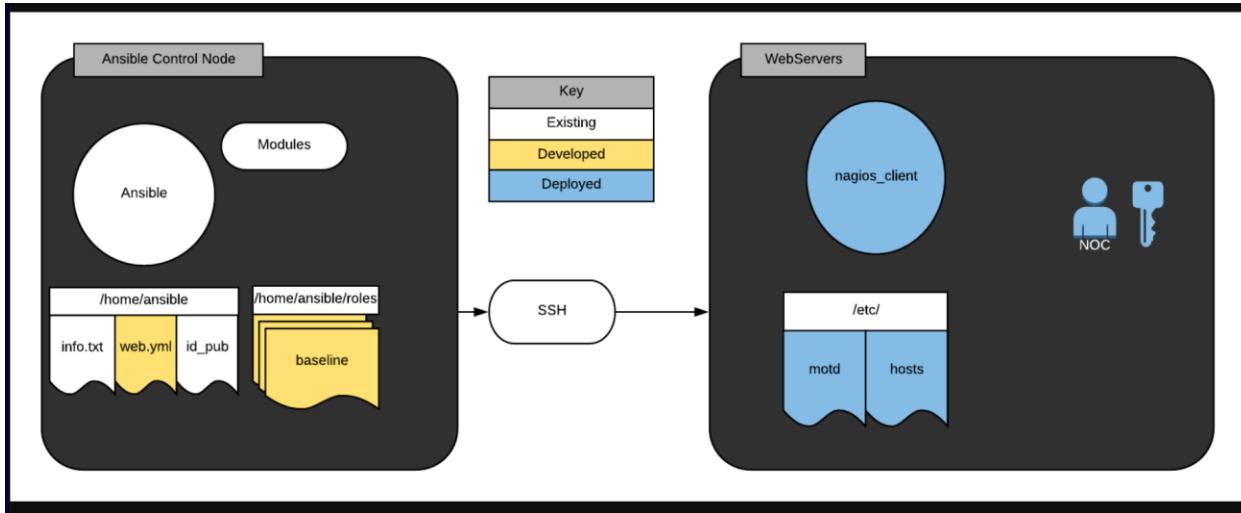
TASK [run data job] *****
changed: [node1]
changed: [node2]

RUNNING HANDLER [Restart and enable httpd] *****
changed: [node2]
changed: [node1]

PLAY RECAP *****
node1 : ok=6    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node2 : ok=6    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansible@control ~]$
```

Working with Ansible Roles



Introduction

Working with Ansible roles is a key concept covered on the Red Hat Certified Ansible Specialist Exam (EX407). This should not be a surprise, considering how much functionality roles provide.

Solution

Log in to the Ansible control node as ansible using the credentials provided:

```
ssh ansible@<PUBLIC_IP_ADDRESS>
```

```
Host '3.210.185.145' is not in the trusted hosts file.  
(ssh-ed25519 fingerprint sha1!! 84:f7:73:f8:11:d8:3e:76:d0:71:f0:f0:40:11:fb:96:d5:43:c9:6d)  
Do you want to continue connecting? (y/n) y  
ansible@3.210.185.145's password:  
[ansible@control1 ~]$ █
```

Note: When copying and pasting code into Vim from the lab guide, first enter: **set paste** (and then **i** to enter insert mode) to avoid adding unnecessary spaces and hashes.

Create a Role Called baseline in /etc/ansible/roles

Create the structure needed for the role:

Change to the roles directory:

```
cd /etc/ansible/roles/
```

Create a directory called baseline and set ownership permissions for both the user account and group ansible:

```
sudo mkdir baseline && sudo chown ansible.ansible /etc/ansible/roles/baseline
```

Create the three subdirectories under the baseline directory:

mkdir /etc/ansible/roles/baseline/{templates,tasks,files}

Create a YAML file called main.yml and place the code "--- inside:

echo "---" > baseline/tasks/main.yml

Note: "---" is the first three characters within each Ansible script!

```
[ansible@control1 ~]$ ls
[ansible@control1 ~]$ cd /etc/ansible/roles/
[ansible@control1 roles]$ sudo mkdir baseline && sudo chown ansible.ansible /etc/ansible/roles/baseline
[ansible@control1 roles]$ mkdir /etc/ansible/roles/baseline/{templates,tasks,files}
[ansible@control1 roles]$ echo "---" > baseline/tasks/main.yml
[ansible@control1 roles]$
```

Configure the Role to Deploy the /etc/motd Template:

Copy the file:

cp /home/ansible/resources/motd.j2 baseline/templates

```
[ansible@control1 roles]$ cp /home/ansible/resources/motd.j2 baseline/templates
[ansible@control1 roles]$
```

Create a file called deploy_motd.yml using the VIM editor:

vim baseline/tasks/deploy_motd.yml

Add the following content:

```
---
- template:
  src: motd.j2
  dest: /etc/motd
```

```
---
- template:
  src: motd.j2
  dest: /etc/motd
~
```

Save and exit with Escape followed by: **wq**.

Open main.yml:

```
vim baseline/tasks/main.yml
```

```
[ansible@control1 roles]$ vim baseline/tasks/main.yml
```

Add the following lines to the file:

```
- name: configure motd
  import_tasks: deploy_motd.yml
```

```
-->
  - name: configure motd
    import_tasks: deploy_motd.yml
~
```

Save and exit with Escape followed by: wq.

Configure the Role to Install the Latest Nagios Client

Find the package we need to install by reading a text file in our home directory:

```
cat /home/ansible/resources/nagios_info.txt
```

```
[ansible@control1 roles]$ cat /home/ansible/resources/nagios_info.txt
```

That file tells us the package we need to install is **nrpe.x86_64**.

Copy the IP of the Nagios server that's in the file and paste it into a text file, as we'll need it later.

```
[ansible@control1 roles]$ cat /home/ansible/resources/nagios_info.txt
## For the sake of the exercise, you only need to install nrpe and
## add a local host entry for "nagios.example.com" pointing to the
## specified IP. There is no need to do any further nagios configuration.

## Install the nrpe package "nrpe.x86_64" in the baseline role.
## You will find it in the EPEL repo which has been configured for you.

## The nagios Server IP should be set in the host file as follows:

10.0.1.89 nagios.example.com
[ansible@control1 roles]$
```

10.0.1.89 nagios.example.com

Create a file, which will install the package, called `deploy_nagios.yml`:

```
vim baseline/tasks/deploy_nagios.yml
```

Add the following content:

```
---
```

```
- yum: name=nrpe state=latest
```

```
[root@control1 ~]# vim baseline/tasks/main.yml
```

Save and exit with Escape followed by: wq.

Open main.yml:

vim baseline/tasks/main.yml

Add the following lines to the bottom of the file:

```
- name: deploy nagios client
  import_tasks: deploy_nagios.yml
```

```
[root@control1 ~]# vim baseline/tasks/main.yml
[...]
- name: configure motd
  import_tasks: deploy_motd.yml
- name: deploy nagios client
  import_tasks: deploy_nagios.yml
```

Save and exit with Escape followed by: wq.

Configure the Role to Add an Entry to /etc/hosts for the Nagios Server

Create a file called edit_hosts.yml:

vim baseline/tasks/edit_hosts.yml

Add the following content, substituting <IP_ADDRESS> with the Nagios server IP you copied earlier

```
[ansible@control1 roles]$ vim baseline/tasks/edit_hosts.yml
```

```
---
- lineinfile:
  line: "<IP_ADDRESS> nagios.example.com"
  path: /etc/hosts
```

```
[root@control1 ~]# vim baseline/tasks/edit_hosts.yml
[...]
- lineinfile:
  line: "10.0.1.89 nagios.example.com"
  path: /etc/hosts
```

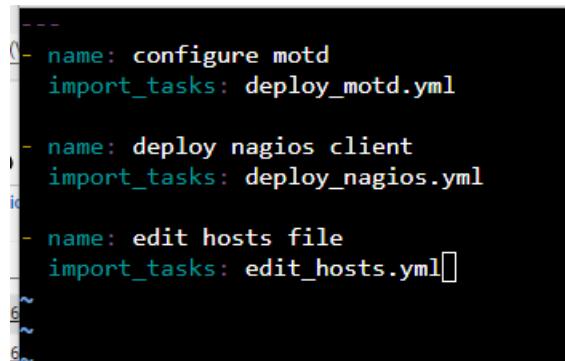
Save and exit with Escape followed by: wq.

Open main.yml:

vim baseline/tasks/main.yml

Add the following lines to the bottom of the file:

```
- name: edit hosts file
  import_tasks: edit_hosts.yml
```



```
---
- name: configure motd
  import_tasks: deploy_motd.yml

- name: deploy nagios client
  import_tasks: deploy_nagios.yml

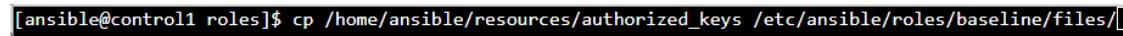
- name: edit hosts file
  import_tasks: edit_hosts.yml
```

Save and exit with Escape followed by: wq.

Configure the Role to Create the noc User and Deploy the Provided Public Key for the noc User on Target Systems

Copy the provided authorized_keys file to our files directory:

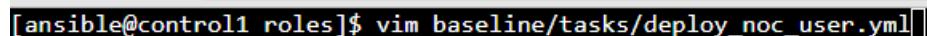
cp /home/ansible/resources/authorized_keys /etc/ansible/roles/baseline/files/



```
[ansible@control1 roles]$ cp /home/ansible/resources/authorized_keys /etc/ansible/roles/baseline/files/
```

Create a file called deploy_noc_user.yml:

vim baseline/tasks/deploy_noc_user.yml



```
[ansible@control1 roles]$ vim baseline/tasks/deploy_noc_user.yml
```

Add the following content:

```
---
- user: name=noc
- file:
  state: directory
  path: /home/noc/.ssh
```

```
mode: 0600
owner: noc
group: noc
- copy:
  src: authorized_keys
  dest: /home/noc/.ssh/authorized_keys
  mode: 0600
  owner: noc
  group: noc
```

```
---
- user: name=noc
- file:
  state: directory
  path: /home/noc/.ssh
  mode: 0600
  owner: noc
  group: noc
- copy:
  src: authorized_keys
  dest: /home/noc/.ssh/authorized_keys
  mode: 0600
  owner: noc
  group: noc
~
~
~
```

Save and exit with Escape followed by :wq.

Open main.yml:

vim baseline/tasks/main.yml

```
[ansible@control1 roles]$ vim baseline/tasks/main.yml
```

Add the following lines to the bottom of the file:

```
- name: set up noc user and key
import_tasks: deploy_noc_user.yml
```

```
---
- name: configure motd
  import_tasks: deploy_motd.yml

- name: deploy nagios client
  import_tasks: deploy_nagios.yml

- name: edit hosts file
  import_tasks: edit_hosts.yml

- name: set up noc user and key
  import_tasks: deploy_noc_user.yml
~
```

Save and exit with Escape followed by: wq.

Edit web.yml to Deploy the baseline Role

Change back to the home directory:

cd /home/ansible/

Open web.yml:

vim resources/web.yml

```
[ansible@control1 roles]$ cd /home/ansible/
[ansible@control1 ~]$ vim resources/web.yml
```

Edit it to match the following:

```
---
- hosts: webservers
  become: yes
  roles:
    - baseline
  tasks:
    - name: install httpd
      yum: name=httpd state=latest
    - name: start and enable httpd
      service: name=httpd state=started enabled=yes
```

```

---
- hosts: webservers
  become: yes
  roles:
    - baseline
  tasks:
    - name: install httpd
      yum: name=httpd state=latest
    - name: start and enable httpd
      service: name=httpd state=started enabled=yes

```

Save and exit with Escape followed by: wq.

Run Your Playbook Using the Default Inventory

Deploy the playbook:

ansible-playbook resources/web.yml

```
[ansible@control1 ~]$ ansible-playbook resources/web.yml
```

```
[ansible@control1 ~]$ ansible-playbook resources/web.yml
PLAY [webservers] ****
TASK [Gathering Facts] ****
ok: [node1]
ok: [node2]

TASK [baseline : template] ****
changed: [node2]
changed: [node1]

TASK [baseline : yum] ****
changed: [node1]
changed: [node2]

TASK [baseline : lineinfile] ****
changed: [node2]
changed: [node1]

TASK [baseline : user] ****
changed: [node1]
changed: [node2]

TASK [baseline : file] ****
changed: [node1]
changed: [node2]

TASK [baseline : copy] ****
changed: [node1]
changed: [node2]

TASK [install httpd] ****
changed: [node1]
changed: [node2]

TASK [start and enable httpd] ****
changed: [node1]
changed: [node2]

PLAY RECAP ****
node1                  : ok=9    changed=8    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node2                  : ok=9    changed=8    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansible@control1 ~]$
```

Check Our Work

Log in to one of the nodes (the IP addresses are on the hands-on lab overview page):

ssh node1

We should see a new MOTD, so we know that play worked.

```
[ansible@control1 ~]$ ssh node1
```

```
[ansible@control1 ~]$ ssh node1
Last login: Tue Oct  3 20:47:34 2023 from ip-10-0-1-234.ec2.internal
#####
node1 is....
```



See if the noc user was set up:

id noc

```
[ansible@node1 ~]$ id noc  
uid=1004(noc) gid=1004(noc) groups=1004(noc)  
[ansible@node1 ~]$ █
```

Check to see if the nrpe package was installed:

sudo yum list nrpe

Addendum

The Linux distribution used

```
[ansible@control1 ~]$ cat /etc/os-release
NAME="Red Hat Enterprise Linux Server"
VERSION="7.9 (Maipo)"
ID="rhel"
ID_LIKE="fedora"
VARIANT="Server"
VARIANT_ID="server"
VERSION_ID="7.9"
PRETTY_NAME="Red Hat Enterprise Linux Server 7.9 (Maipo)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:redhat:enterprise_linux:7.9:GA:server"
HOME_URL="https://www.redhat.com/"
BUG_REPORT_URL="https://bugzilla.redhat.com/"

REDHAT_BUGZILLA_PRODUCT="Red Hat Enterprise Linux 7"
REDHAT_BUGZILLA_PRODUCT_VERSION=7.9
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux"
REDHAT_SUPPORT_PRODUCT_VERSION="7.9"
[ansible@control1 ~]$ █
```

The Hosts file entries:

```
[ansible@control1 ~]$ sudo cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6

10.0.1.89 node1
10.0.1.102 node2
[ansible@control1 ~]$ █
```

Note: this is the Hosts.ini file in Windows

Ansible's default inventory file:

cat /etc/ansible/hosts

```
## db-[99:101]-node.example.com  
localhost  
[webservers]  
node1  
node2  
[ansible@control1 ~]$ []
```

How to create the ansible user:

```
[cloud_user@master ~]$ sudo adduser ansible  
[sudo] password for cloud_user:  
[cloud_user@master ~]$ sudo passwd ansible  
Changing password for user ansible.  
New password:  
Retype new password:  
passwd: all authentication tokens updated successfully.  
[cloud_user@master ~]$ []
```

The visudo file

vi sudo visudo

```
# Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)  
#includedir /etc/sudoers.d  
ec2-user      ALL=(ALL)      NOPASSWD: ALL  
ansible       ALL=(ALL)      NOPASSWD: ALL
```

A list of all the users on a server:

```
[ansible@control1 ~]$ getent passwd | awk -F: '{print $1}'  
root  
bin  
daemon  
adm  
lp  
sync  
shutdown  
halt  
mail  
operator  
games  
ftp  
nobody  
systemd-network  
dbus  
polkitd  
sshd  
postfix  
chrony  
ec2-user  
linuxacademy  
cloud_user  
ansible  
[ansible@control1 ~]$ []
```

How to reveal both the public and private keypair on the server:

```
[ansible@control1 ~]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQACQC8N9UgTWhdmejoMsW4N9T3dLDb5zY9xmXMe/zMwqrMMTH1LE4yoG4jTThqV6s+j8WDq2mmB/Ix+noCQVCZAnbu64+iaCmRJS9dvks+boayEz1GV7wpSfnxxjaGJN4v4VlJa69t5aDRfZMDQwrgKcnpyJGt1SUlwD05sA6174soq5ebmw/Buv1gZuE7iaME8Q8Ksf/rYR9i0bZTBuUAT7dBZmZL1UTHz086BEtGmhR6Zla++Z3vKXlx5cFFoxdfvXbmdaHJHmooTCsqUS9mbe+CygA4Zulp6ePLcC08ETV1
UGt3KLm7G4ZaogV8zkYDm06AVgW2k+5GU7IoPxN ansible@control1
[ansible@control1 ~]$ cat ~/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEavbFVIE1oXZno6DLFuDFU93Sw+c2PcZ1zHv8zMKG5jdDLR95x0
MqDuI084a1er1/Fg67ppgfYHfp6AkFQnQJ27uuK4mpk5uVXZLPm6GshHM5R1e1q
Un58CY2h1eL+rZSWuverbglgOXZTA0MKxpJwqc1rZULFW8IQ9v0ou+KKUxM5sP
wbr-9YGV3x04mjbPEPCo7H/8kfytG2UwVMA1o3QWmzNSVExe89POgrLrpolemzWvn
d7y14sxEBraxXX7125nhyR5qKEwka1Evz23vgmIAOGhpaenjy3A3AtPBE1ZVBr
dy15uxuGkqIfd5GASqOpFYTtpUr1G0yKD8TQ1DAQABoIAuLKhv795Ua
WpDn017drgzckR14eBy79meUCYyDnNLGSD1rc94ccMT/+0NTy3LizwzRcfAmDp
SoccaeGbzt4evhxdjwgpFEBzg1t07oi/crgWmB5ayDxBGPJyq1gZoM0Lqkee1PB
pdEp08yCEAfus87370NWYfzEss5iGF710i13oi670BV+0gCv3ad8j5xxDkqN8sNY/
qzTRfp036+XR9fYL1/P7bxG3CD1F0adbn/1cfkwMrnxw11TBewYOnzuDq5mWm
mqfMRNGJcdsAswFjyFx1lAPsxFvf519B0Z6Z4ohNL1QFGkrTCdK3RFJPdYd
iuymQuEcgyFA8EB3i28gvQr/2/6B6gEmrTxgATNLsZu+4jAWLFsoB4VB1yxlwo3
TAVz29CqDW7INU8kCEFvEdbkzTkn1qhusw0iu0s9vFwf7U319ajp033XrqdDb
XTpSY2nmYo86Csuk12PnC81CX8pJGhbC4Gx5WtRNwAtevTj1jXkcrgyEay143
YVgp+qrWwCkdRbM82a1VL7/dGeVkrbtix4E5F5LFe2T9YLVz5oykLUghh
h2ChFy6463a/v7n6joi+j+zQvhIM3nXug8GNteczlmpimSc59Grdh0LyYjdZk
sYGECPDDqk0w6K8gmnJ2ygGyMn-1jNWE1hk4Q9HUCgYBxPKha3b+mMTsnMBhUmIR+
VouGpVIZfosd/JyHvt1v6gv1ySP2QyVhcb534u69D+mDvyy+J6qBhw7L9gOMt8
muHPG55FOt/wgRjnVFTP/FgyRekj5PHdtUQnGrRkjGNT3//7MM2entqSFwC1/wX
PMfQnxC5xUs59PrZyy/KQK8gjh3dkzAkjr>8fa7/1ftCBY0e0D/omschQyCpn13R
qhfN0C/A9g8jzC08H6xpTs62fft/EocAL112yobcJe75zdvXOQ0qlRxXaAEtwh+
eNvy8tYdV0s86mQmDqjw1IPxCPFD7+Esgxofg9wxmMh8BFy1u00P+j+Qf1b
AdlJ4oGAJJR185CfyAFl1xb2ugjgsFxabDp033zMF+8mu61SbzoxVxqsMjBMAO
I4j9GvMi/aHDWQt3sDFD075ww5qjycY4zTBgJQYc/bUpyY014Ma0JppfhWMcmez
b99ksawVzLqQ1aG/Yi3e7qqU33N11Ub8xa8gFmmIFvxBSv7V1k=
-----END RSA PRIVATE KEY-----
[ansible@control ~]$
```

How to generate the keypair on the server

NOTE: This is key for the ansible control plane or master server to communicate with nodes

```
[ansible@master cloud_user]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):
Created directory '/home/ansible/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansible/.ssh/id_rsa.
Your public key has been saved in /home/ansible/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Pm4NkxPNPSE/vDR+fV7pqeofCo4DJNZdiwOC3ylFT9U ansible@master
The key's randomart image is:
-----[RSA 2048]-----
. . . .
. . oo ..E.
. = +.oo.= .
= = +..o 0
. + So o = ...
. .= o ..+
. o* o..+
o+.o . .o.
oo...oo.
-----[SHA256]-----
[ansible@master cloud_user]$
```

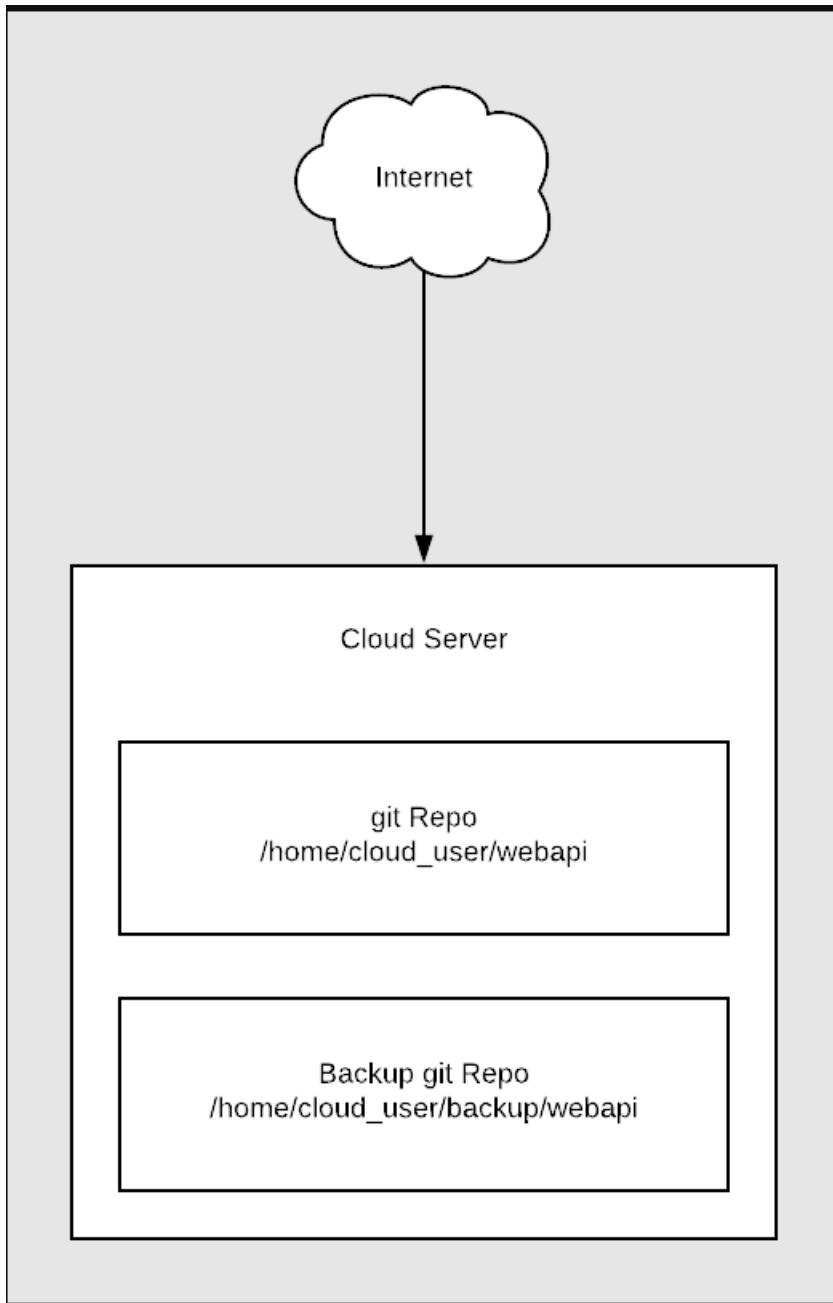
How to copy the keys to the nodes:

```
ansible@control1 ~]$ ssh-copy-id node1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
(if you think this is a mistake, you may want to use -f option)

ansible@control1 ~]$ ssh-copy-id node2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
(if you think this is a mistake, you may want to use -f option)

ansible@control1 ~]$
```

Using Git for Source Control and Programming



Introduction

Once this activity is complete, you will understand how to install, configure, and check files into and out of the Git server using the command line.

Solution

Log in to the server

ssh cloud_user@<PUBLIC_IP_ADDRESS>

```
/ $ ssh cloud_user@52.23.252.67
Host '52.23.252.67' is not in the trusted hosts file.
(ssh-ed25519 fingerprint sha1!! 37:2c:22:29:d0:3c:32:11:7b:23:11:0d:a3:f5:9a:6c:05:43:3b:be)
Do you want to continue connecting? (y/n) y
cloud_user@52.23.252.67's password:
[cloud_user@ip-10-0-1-100 ~]$
```

Install the git Tool

Install Git:

sudo yum install git -y

```
Installed:
  git.x86_64 0:1.8.3.1-25.el7_9

Dependency Installed:
  perl-Error.noarch 1:0.17020-2.el7          perl-Git.noarch 0:1.8.3.1-25.el7_9          perl-TermReadKey.x86_64 0:2.30-20.el7          rsync.x86_64 0:3.1.2-12.el7_9

Complete!
[cloud_user@ip-10-0-1-100 ~]$
```

Initialize a webapi Repository

Initialize a new webapi Git repository in the /home/cloud_user directory:

git init ~/webapi

```
[cloud_user@ip-10-0-1-100 ~]$ git init ~/webapi
Initialized empty Git repository in /home/cloud_user/webapi/.git/
[cloud_user@ip-10-0-1-100 ~]$
```

Change to the webapi directory:

cd webapi/

```
[cloud_user@ip-10-0-1-100 ~]$ cd webapi/
[cloud_user@ip-10-0-1-100 webapi]$
```

Create a README file that only contains the text webapi:

```
echo "webapi" > README
```

```
[cloud_user@ip-10-0-1-100 webapi]$ echo "webapi" > README
[cloud_user@ip-10-0-1-100 webapi]$ █
```

Create a current version file called currver.txt that only contains the text v0.1:

```
echo "v0.1" > currver.txt
```

```
[cloud_user@ip-10-0-1-100 webapi]$ echo "webapi" > README
[cloud_user@ip-10-0-1-100 webapi]$ █
```

Add the files to the repo:

```
git add *
```

```
[cloud_user@ip-10-0-1-100 webapi]$ git add *
[cloud_user@ip-10-0-1-100 webapi]$ █
```

Verify Git knows about the files:

```
git status
```

```
[cloud_user@ip-10-0-1-100 webapi]$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   README
#
[cloud_user@ip-10-0-1-100 webapi]$ █
```

You should see the files listed.

Commit the files with a comment:

```
git commit -m "First commit"
```

```
[cloud_user@ip-10-0-1-100 webapi]$ git commit -m "First commit"
[master (root-commit) 329669f] First commit
Committer: Cloud Assessment User <cloud_user@ip-10-0-1-100.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:
```

```
git config --global user.name "Your Name"
git config --global user.email you@example.com
```

```
After doing this, you may fix the identity used for this commit with:
```

```
git commit --amend --reset-author
```

```
1 file changed, 1 insertion(+)
create mode 100644 README
[cloud_user@ip-10-0-1-100 webapi]$ █
```

Set your username, replacing <YOUR_NAME> with yours:

```
git config --global user.name "<YOUR_NAME>"
```

```
[cloud_user@ip-10-0-1-100 webapi]$ git config --global user.name costas constantinou█
```

Set your email address, replacing <YOUR_EMAIL_ADDRESS> with yours:

```
git config --global user.email "<YOUR_EMAIL_ADDRESS>"
```

```
[cloud_user@ip-10-0-1-100 webapi]$ git config --global user.email costas778@gmail.com
[cloud_user@ip-10-0-1-100 webapi]$ █
```

Amend the commit to change the author:

```
git commit --amend --reset-author
```

```
[cloud_user@ip-10-0-1-100 webapi]$ git commit --amend --reset-author
[master b680ac5] First commit
1 file changed, 1 insertion(+)
create mode 100644 README
[cloud_user@ip-10-0-1-100 webapi]$ █
```

Leave the commit message as-is and press Escape followed by: **wq**.

View the history:

git log

```
[cloud_user@ip-10-0-1-100 webapi]$ git log
commit 78da7cad6da7cafdbb2829899bcf76a7bcbeacef
Author: costas <costas778@gmail.com>
Date:   Thu Oct 5 00:27:12 2023 -0400

    First commit
[cloud_user@ip-10-0-1-100 webapi]$
```

You should see your author's name, email address, the date, and "First commit".

Clone the webapi Repository After Configuration into a Backup Directory

Change back to the root directory:

cd ~

Create a backup directory that contains a clone of the original webapi repo you previously set up:

git clone /home/cloud_user/webapi/ /home/cloud_user/backup/webapi

```
[cloud_user@ip-10-0-1-100 ~]$ git clone /home/cloud_user/webapi/ /home/cloud_user/backup/webapi
Cloning into '/home/cloud_user/backup/webapi'...
done.
[cloud_user@ip-10-0-1-100 ~]$
```

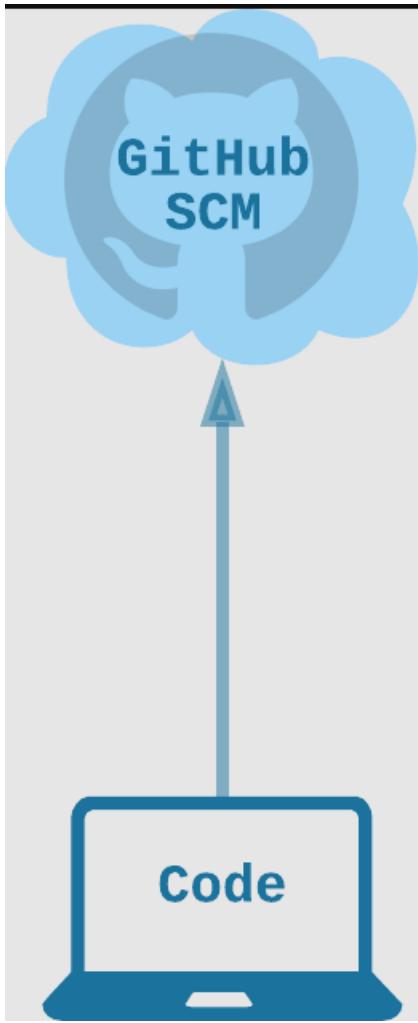
List the contents of home/cloud_user/backup/webapi:

ll /home/cloud_user/backup/webapi/

```
[cloud_user@ip-10-0-1-100 ~]$ ll /home/cloud_user/backup/webapi/
total 4
-rw-rw-r--. 1 cloud_user cloud_user 7 Oct  5 00:32 README
[cloud_user@ip-10-0-1-100 ~]$
```

You should see the files you created and added to the webapi directory.

Use Git to Make Changes to Code



Introduction:

The files for this exercise can be found at the following repository:

<https://github.com/costas778/cicd-pipeline-train-schedule-git>

Please make a fork!

Login to the server:

```
Host '23.22.196.67' is not in the trusted hosts file.  
(ssh-ed25519 fingerprint sha1!! e2:a8:a2:a6:64:b0:ea:4a:1a:06:41:26:3b:df:ae:4a:b9:ed:53:8a)  
Do you want to continue connecting? (y/n) y  
cloud_user@23.22.196.67's password:  
[cloud_user@ip-10-0-1-18 ~]$
```

Install Git:

To push a change to the code, you will need to:

Install git:

sudo yum -y install git

```
Installed:  
git.x86_64 0:1.8.3.1-25.el7_9  
  
Dependency Installed:  
perl-Error.noarch 1:0.17020-2.el7      perl-Git.noarch 0:1.8.3.1-25.el7_9      perl-TermReadKey.x86_64 0:2.30-20.el7      rsync.x86_64 0:3.1.2-12.el7_9  
  
Complete!  
[cloud_user@ip-10-0-1-18 ~]$
```

Configure the name and email address associated with git:

```
git config --global user.name "<YOUR NAME>"  
git config --global user.email "<YOUR EMAIL>"
```

```
[cloud_user@ip-10-0-1-18 ~]$ git config --global user.name costas  
[cloud_user@ip-10-0-1-18 ~]$ git config --global user.email costas778@gmail.com  
[cloud_user@ip-10-0-1-18 ~]$
```

Create an SSH key, using the default settings:

ssh-keygen -t rsa -b 4096

```
[cloud_user@ip-10-0-1-18 ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cloud_user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cloud_user/.ssh/id_rsa.
Your public key has been saved in /home/cloud_user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:vWqwH2+VVd/R6fKQPV5rCG50EcXqvRuGMLQL92rWSLs cloud_user@ip-10-0-1-18.ec2.internal
The key's randomart image is:
+---[RSA 4096]----+
|          .+.o|
|          . +o|
|          . *.+|
|          ..o.*.++|
| S.+=+oB +|
| . o=Bo.* |
| o. +o+o.o.|
| . .+=.....|
| oo.+E. ...|
+---[SHA256]----+
[cloud_user@ip-10-0-1-18 ~]$
```

Display the contents of our new public key (be sure to copy the key to your clipboard):

cat /home/cloud_user/.ssh/id_rsa.pub

```
[cloud_user@ip-10-0-1-18 ~]$ cat /home/cloud_user/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAOQL5r+BksBDjtJi+4hedGP2HeIJBdIK/m0iPi9HRbI5BQB0zCGJx3Q
CY6f0nGtecXyhgOi6Vx4I/yEA6Szq0C9VmS701GQ+gn/Pi2KyHIWX0Y/v8F8PJmSf8ng7I9vb9wgIXXARlhiM
VIO84d/X4Myj3Id3dYq5lg24P8yyfs6Pv0tb9U4SSQb4Qi+K8v/KcCbp9iBTD5cRPz1RbomoaVfGUaBUlftMsS
2zdIXbrsZMzj/YewcaxOYNQIVaHmpHab/toU5phXdxBbld1N6OAwBG3nShuCagToUI/6lxQaf1zfLCsL0q61
yFut1eO1KIOwd2ddNNZZzmmQ0SBI9LMcX6zdjFG5PwJKR508wyxF8q4nPqWflfDNRuX0aq3kYQkNEhXOTUsKuhP
55JsXcW8GzB6ZN+QVO/ZSqLijf3uUXfBvnzuDHbszttW8PUa8iDOVl9r/qqI1q32U5OCEj/P4gQXNsF8dQFXake
pgOOxkJ7IBKrd7VKHGnRcxzdZhkpFT32uJdYtex7TMNzz1RuG2b9/y/sOzbXx2QgYsfBeYp1Tjv9Ulcu4NdB/An
nN55DfwX9aeIS2c7cSF6EDHFUeGWJlUMc0IT8o9dYVvcrccOuOj7+fV NJmRNQc/j6GBQVoE
[cloud_user@ip-10-0-1-18 ~]$
```

Log in to GitHub, navigate to **Settings**, click on SSH and GPG keys, and click on New SSH Key. Paste your public key information into this field and then click Add SSH key.

ssh-rsa

```
AAAAB3NzaC1yc2EAAAADAQABAAACAOQL5r+BksBDjtJi+4hedGP2HeIJBdIK/m0iPi9HRbI5BQB0zCGJx3Q
CY6f0nGtecXyhgOi6Vx4I/yEA6Szq0C9VmS701GQ+gn/Pi2KyHIWX0Y/v8F8PJmSf8ng7I9vb9wgIXXARlhiM
VIO84d/X4Myj3Id3dYq5lg24P8yyfs6Pv0tb9U4SSQb4Qi+K8v/KcCbp9iBTD5cRPz1RbomoaVfGUaBUlftMsS
2zdIXbrsZMzj/YewcaxOYNQIVaHmpHab/toU5phXdxBbld1N6OAwBG3nShuCagToUI/6lxQaf1zfLCsL0q61
yFut1eO1KIOwd2ddNNZZzmmQ0SBI9LMcX6zdjFG5PwJKR508wyxF8q4nPqWflfDNRuX0aq3kYQkNEhXOTUsKuhP
55JsXcW8GzB6ZN+QVO/ZSqLijf3uUXfBvnzuDHbszttW8PUa8iDOVl9r/qqI1q32U5OCEj/P4gQXNsF8dQFXake
pgOOxkJ7IBKrd7VKHGnRcxzdZhkpFT32uJdYtex7TMNzz1RuG2b9/y/sOzbXx2QgYsfBeYp1Tjv9Ulcu4NdB/An
nN55DfwX9aeIS2c7cSF6EDHFUeGWJlUMc0IT8o9dYVvcrccOuOj7+fV NJmRNQc/j6GBQVoE
NsF8dQFXakepgOOxkJ7IBKrd7VKHGnRcxzdZhkpFT32uJdYtex7TMNzz1RuG2b9/y/sOzbXx2QgYsfBeYp1Tjv9
Ulcu4NdB/AnnN55DfwX9aeIS2c7cSF6EDHFUeGWJlUMc0IT8o9dYVvcrccOuOj7+fV NJmRNQc/j6GBQVoE
```

U5dr6QZb06Selg/EP15/ZQB1DtC79P5fNsixw== cloud_user@ip-10-0-1-18.ec2.internal

The screenshot shows the GitHub user settings page for 'costas778'. The left sidebar includes links for Public profile, Account, Appearance, Accessibility, Notifications, Access (Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys, Organizations, Enterprises, Moderation), and Code, planning, and automation. The 'SSH and GPG keys' section is currently selected. The main content area is titled 'Add new SSH Key' with a title input field containing 'GIT_LAB23' and a key type dropdown set to 'Authentication Key'. A large text area contains a long SSH key string starting with 'ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQACQDL5r+BksBDjtJi+4hedGP2HeJBdIK/m0Pi9HRb15BQB0zCGJx3QCY6f0nGtecXyhgO16Vx4l/yEA6ZSq0C9V'. Below this is a green 'Add SSH key' button.

Create a Fork:

Create a personal fork of the sample repository <https://github.com/costas778/cicd-pipeline-train-schedule-git>

Navigate to the URL <https://github.com/costas778/cicd-pipeline-train-schedule-git>

Click Fork in the top-right of the page.

Clone your personal fork from GitHub.

Click the green Clone or download button and copy the string that is displayed

In your terminal, run the following commands:

```
cd ~  
git clone git@github.com:costas778/cicd-pipeline-train-schedule-git.git
```

```
[cloud_user@ip-10-0-1-18 ~]$ cd ~
[cloud_user@ip-10-0-1-18 ~]$ git clone git@github.com:costas778/cicd-pipeline-train-schedule-git.git
Cloning into 'cicd-pipeline-train-schedule-git'...
The authenticity of host 'github.com (140.82.113.3)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
ECDSA key fingerprint is MD5:7b:99:81:1e:4c:91:a5:0d:5a:2e:80:13:3f:24:ca.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added 'github.com,140.82.113.3' (ECDSA) to the list of known hosts.
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 28 (delta 0), reused 1 (delta 0), pack-reused 26
Receiving objects: 100% (28/28), 12.48 KiB | 0 bytes/s, done.
Resolving deltas: 100% (2/2), done.
[cloud_user@ip-10-0-1-18 ~]$ 
```

Change your directory to the following:

```
cd cicd-pipeline-train-schedule-git/
```

```
[cloud_user@ip-10-0-1-18 ~]$ ls
cicd-pipeline-train-schedule-git  init_pass
[cloud_user@ip-10-0-1-18 ~]$ cd cicd-pipeline-train-schedule-git/
[cloud_user@ip-10-0-1-18 cicd-pipeline-train-schedule-git]$ 
```

Create a feature branch to contain the change:

```
git checkout -b myBranch
```

```
[cloud_user@ip-10-0-1-18 cicd-pipeline-train-schedule-git]$ git checkout -b myBranch
Switched to a new branch 'myBranch'
[cloud_user@ip-10-0-1-18 cicd-pipeline-train-schedule-git]$ 
```

Change the header text in views/index.jade from “Train Schedule” to “Find your train Today!”:

```
vim views/index.jade
```

Add the change in views/index.jade to the next commit:

```
git add .
```

```
[cloud_user@ip-10-0-1-18 cicd-pipeline-train-schedule-git]$ git add .
[cloud_user@ip-10-0-1-18 cicd-pipeline-train-schedule-git]$ 
```

Commit the change:

```
git commit -m "<UNIQUE MESSAGE>"
```

```
git commit -m "Made a change to my index.jade"
```

```
[cloud_user@ip-10-0-1-18 cicd-pipeline-train-schedule-git]$ git commit -m "Made a change to my index.jade"
[myBranch 6918aba] Made a change to my index.jade
 1 file changed, 1 insertion(+), 1 deletion(-)
[cloud_user@ip-10-0-1-18 cicd-pipeline-train-schedule-git]$ █
```

Push the Commit

Push the change to the remote SCM repository:

```
git config --global push.default simple
```

```
[cloud_user@ip-10-0-1-18 cicd-pipeline-train-schedule-git]$ git config --global push.default simple█
```

```
git push --set-upstream origin myBranch
```

```
[cloud_user@ip-10-0-1-18 cicd-pipeline-train-schedule-git]$ git push --set-upstream origin myBranch
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 411 bytes | 0 bytes/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'myBranch' on GitHub by visiting:
remote:   https://github.com/costas778/cicd-pipeline-train-schedule-git/pull/new/myBranch
remote:
To git@github.com:costas778/cicd-pipeline-train-schedule-git.git
 * [new branch]      myBranch -> myBranch
Branch myBranch set up to track remote branch myBranch from origin.
[cloud_user@ip-10-0-1-18 cicd-pipeline-train-schedule-git]$ █
```

Create a pull request to merge the feature branch into the master branch:

On the GitHub fork page, click Branches, locate **myBranch** under Your branches, and click **New pull request**.



In the top-left of the **new pull request**, change the base fork dropdown menu to your personal fork.

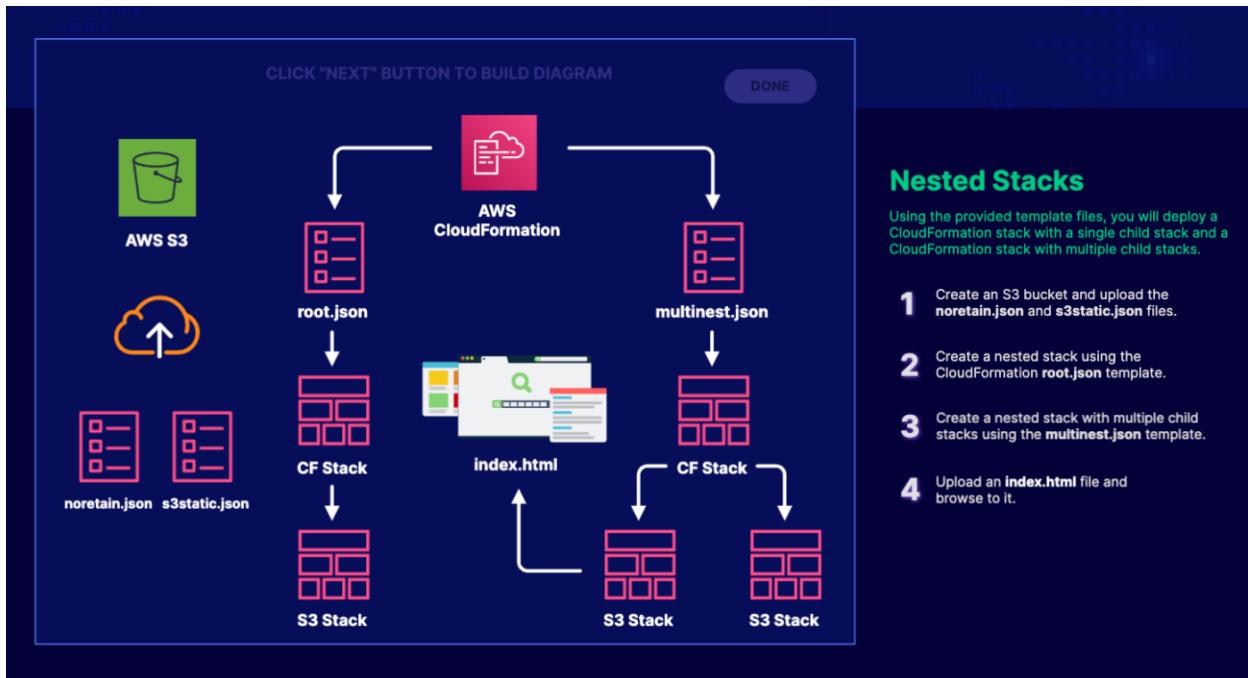
Merge the pull request

After a few moments, click the **Merge pull request** button

Click **Confirm merge**:

The change is pushed.

Working with CloudFormation Nested Stacks



Solution

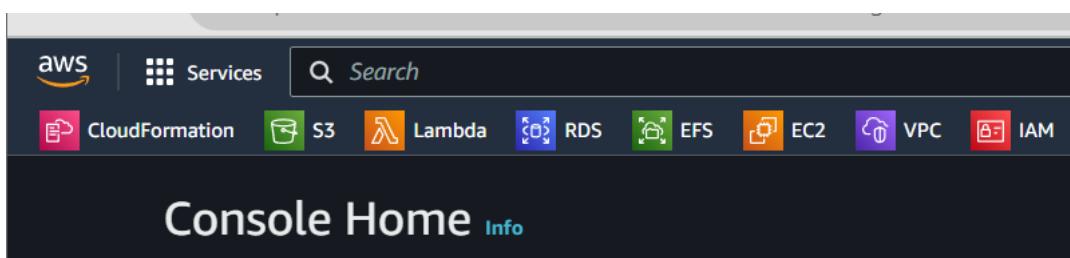
Log in with the credentials provided, and make sure you are in the us-east-1 (N. Virginia) region.

[Download the necessary files from the lab's GitHub repository](#)

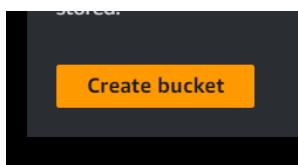
<https://github.com/costas778/Working-with-CloudFormation-Nested-Stacks>

[Create an S3 Bucket and Upload a CloudFormation Template to It](#)

Navigate to S3.



Click + **Create bucket**.



On the Name and region screen, give your bucket a unique name. (**Note:** It must be all lowercase letters and be unique across all AWS accounts.)

A screenshot of the "Create bucket" configuration page. At the top, the navigation path is shown as "Amazon S3 > Buckets > Create bucket". The main title is "Create bucket" with an "Info" link. Below it, a sub-instruction says "Buckets are containers for data stored in S3. [Learn more](#)". A section titled "General configuration" contains fields for "Bucket name" (containing "CCbucket7778") and "AWS Region" (set to "US East (N. Virginia) us-east-1"). There is also a "Copy settings from existing bucket - optional" section with a "Choose bucket" button. The entire form is set against a dark background.

Click **Next**.

A screenshot of a confirmation step. It shows a message in a blue-bordered box: "After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings." Below this, there are two buttons: "Cancel" and a yellow "Create bucket" button.

Accept the default settings on the next screens, and click **Create bucket**.

A screenshot of a success message. It features a green header bar with the text "Success! Successfully created bucket 'ccbucket7778'" and "To upload files and folders, or to configure additional bucket settings, choose View details.". Below this, the navigation path "Amazon S3 > Buckets" is visible.

Click to open the bucket, and click **Upload**.

The screenshot shows the AWS S3 console interface. At the top, the bucket name 'ccbucket7778' is displayed with a 'Info' link. Below the header, there are tabs for 'Objects', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. The 'Objects' tab is selected, showing a sub-header 'Objects (0)'. A note below states: 'Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permission.' Below this are several action buttons: 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', 'Create folder', and 'Upload'. A search bar labeled 'Find objects by prefix' is present. The main table has columns for 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'. A message at the bottom right says 'No objects'.

Click Add files, and select the **s3static.json** CloudFormation template you downloaded earlier.

The screenshot shows a Windows File Explorer window. The path is 'This PC > Downloads > Working-with-CloudFormation-Nested-Stacks-main > Working-with-CloudFormation-Nested-Stacks-main'. The left sidebar shows 'Organise' and 'New folder' options. The main pane lists files and folders. On the left, there are icons for '32-reviews-create', 'dump', 'OneNote Notebooks', 'section3-hol2', 'System32', 'fountainheadequityinvestments.com', 'OneDrive - Personal', and 'This PC'. On the right, a table lists files: 'index.html' (Date modified: 03/05/2023 03:58), 'multinest.json' (03/05/2023 03:58), 'noretain.json' (03/05/2023 03:58), 'root.json' (03/05/2023 03:58), and 's3static.json' (03/05/2023 03:58). The 's3static.json' file is highlighted with a blue selection bar.

The screenshot shows the 'Add files' dialog in the AWS S3 console. At the top, it says 'Files and folders (1 Total, 2.6 KB)' with buttons for 'Remove', 'Add files', and 'Add folder'. Below this, a note says 'All files and folders in this table will be uploaded.' A search bar 'Find by name' is present. The main table has columns for 'Name', 'Folder', 'Type', and 'Size'. One row is shown: 's3static.json' (application/json, 2.6 KB). At the bottom right, there are navigation arrows (< 1 >).

Accept the default settings on the next screens, and click **Upload**.

Click **Upload**.

Click **Add files**, and select the **noretain.json**

Amazon S3 > Buckets > ccbucket7778

ccbucket7778 [Info](#)

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

C Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Find objects by prefix

Name	Type	Last modified	Size	Storage class
noretain.json	json	October 14, 2023, 15:52:50 (UTC+03:00)	2.5 KB	Standard
s3static.json	json	October 14, 2023, 15:51:32 (UTC+03:00)	2.6 KB	Standard

CloudFormation template you downloaded earlier.

Accept the default settings on the next screens, and click **Upload**.

Create a Nested Stack from the CloudFormation Root Template

In a new browser tab, navigate to **CloudFormation**.

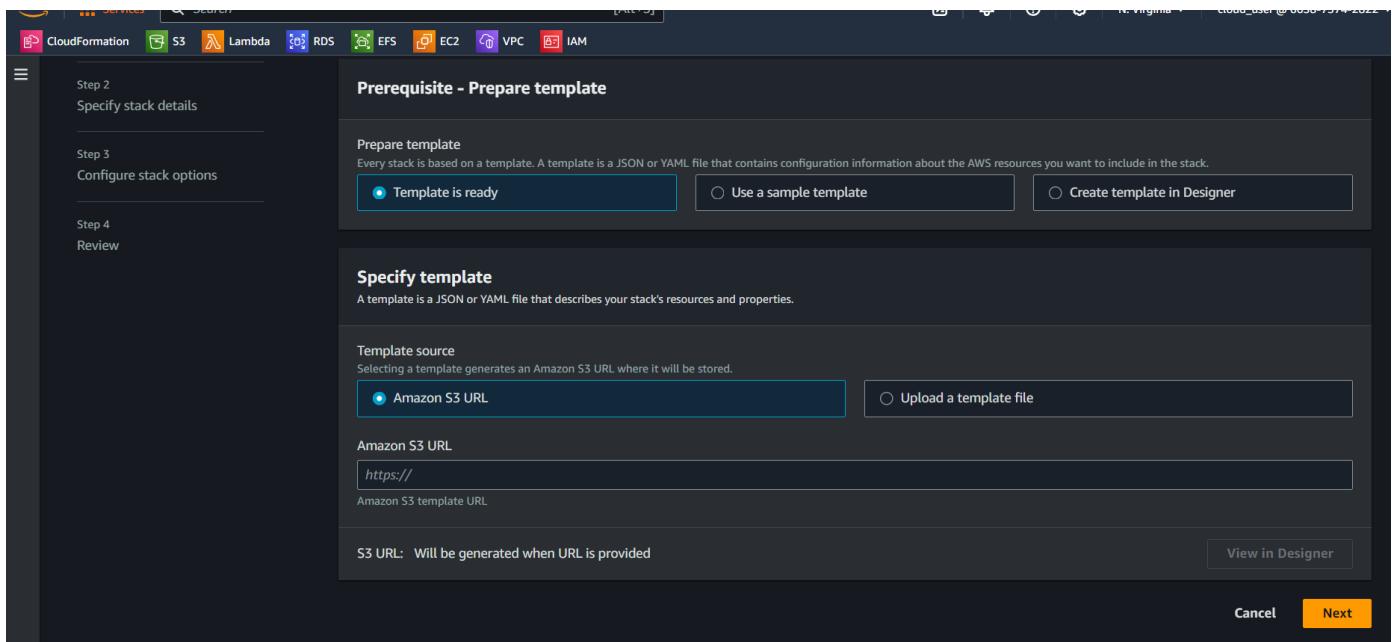
CloudFormation > Stacks

Stacks (1)

Filter status Active View nested

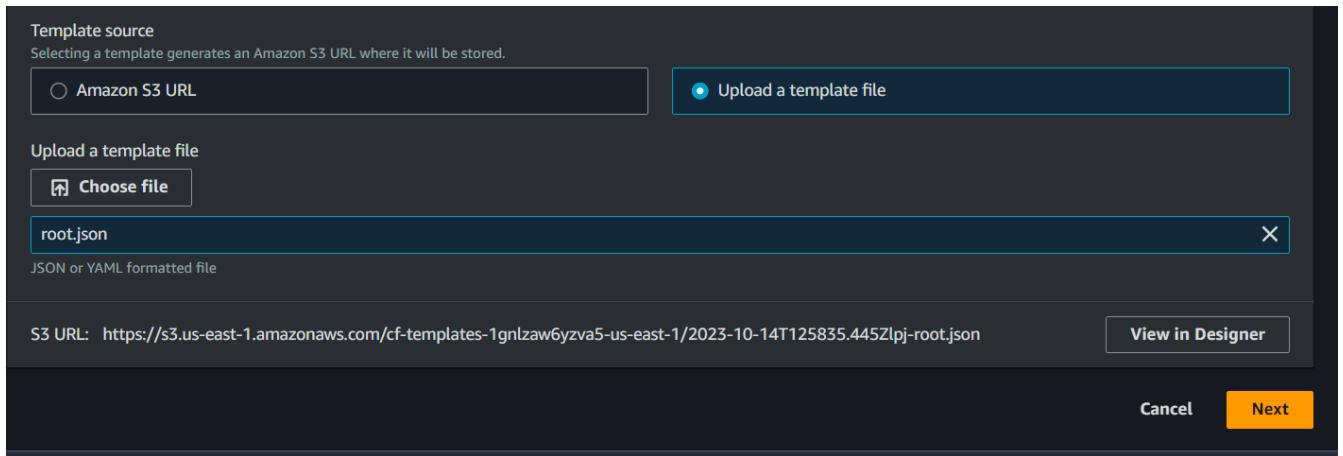
Stack name	Status	Created time	Description
cfst-2095-42ff0b0e257037f3a8b7e263fa4a063b	CREATE_COMPLETE	2023-10-14 14:57:45 UTC+0300	-

Click **Create stack**.



In the Prerequisite - Prepare template section, select **Template is ready**.

Click **Upload a template file**.

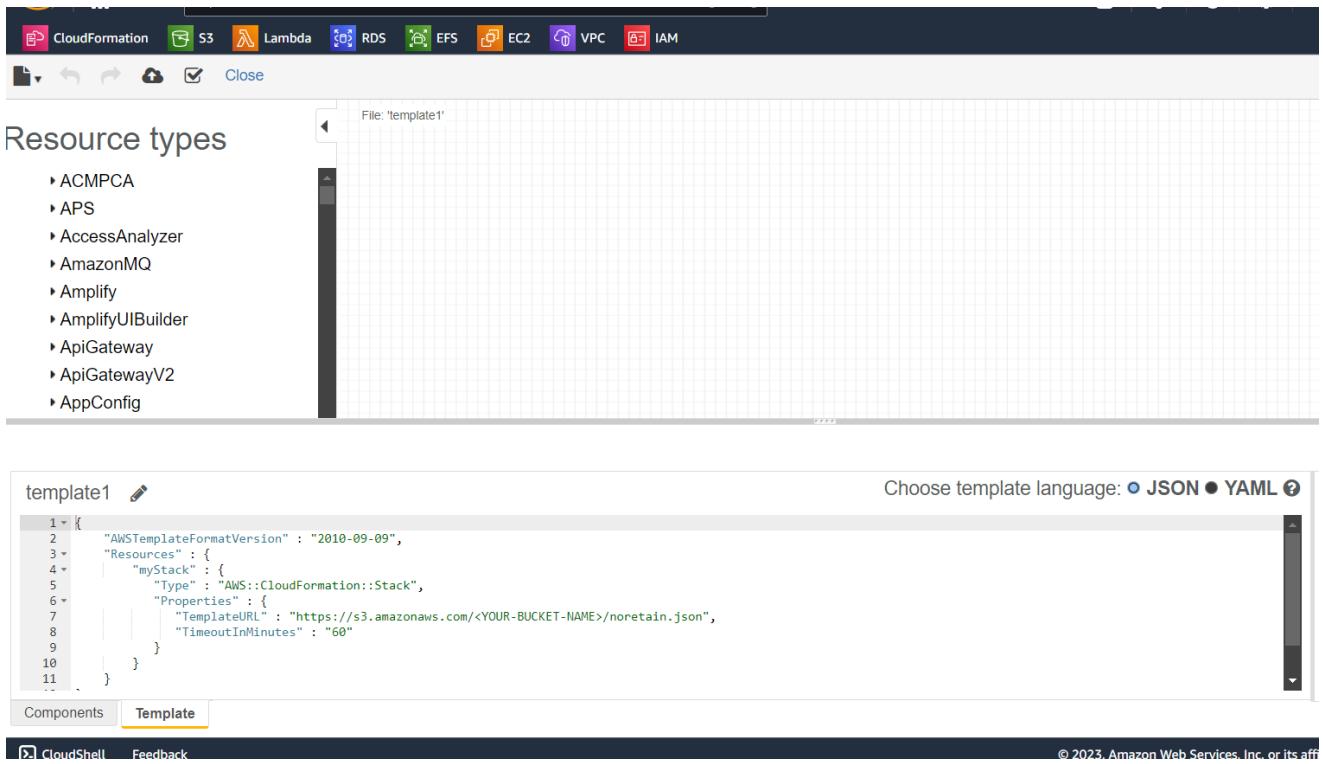


Upload the **root.json** file from your local machine.

Click **View in Designer**.

```
{  
  "AWSTemplateFormatVersion" : "2010-09-09",  
  "Resources" : {  
    "myStack" : {  
      "Type" : "AWS::CloudFormation::Stack",  
      "Properties" : {  
        "TemplateURL" : "https://s3.amazonaws.com/<YOUR-BUCKET-NAME>/noretain.json",  
      }  
    }  
  }  
}
```

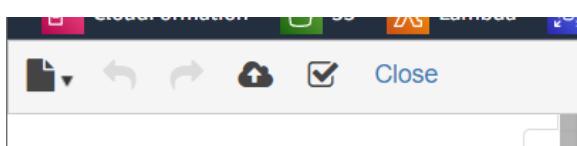
```
        "TimeoutInMinutes" : "60"  
    }  
}  
}
```



In the "TemplateURL" line, replace <YOUR-BUCKET-NAME> with the name of the bucket you just created.



Click the checkbox at the top to **validate** the template, and then click the **cloud icon** with the up arrow to create the stack.



Click **Next**.

On the stack details page, give it a Stack name of **s3webnoretain**.

Specify stack details

Stack name

Stack name

s3webnoretain

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

No parameters

There are no parameters defined in your template

Cancel Previous Next

Click **Next**.

Leave the settings on the stack options page, and click **Next**.

Check the boxes to accept the acknowledgements, and click **Submit**.

Capabilities

The following resource(s) require capabilities: [AWS::CloudFormation::Stack]

This template contains Identity and Access Management (IAM) resources. Check that you want to create each of these resources and that they have the minimum required permissions. In addition, they have custom names. Check that the custom names are unique within your AWS account. [Learn more](#)

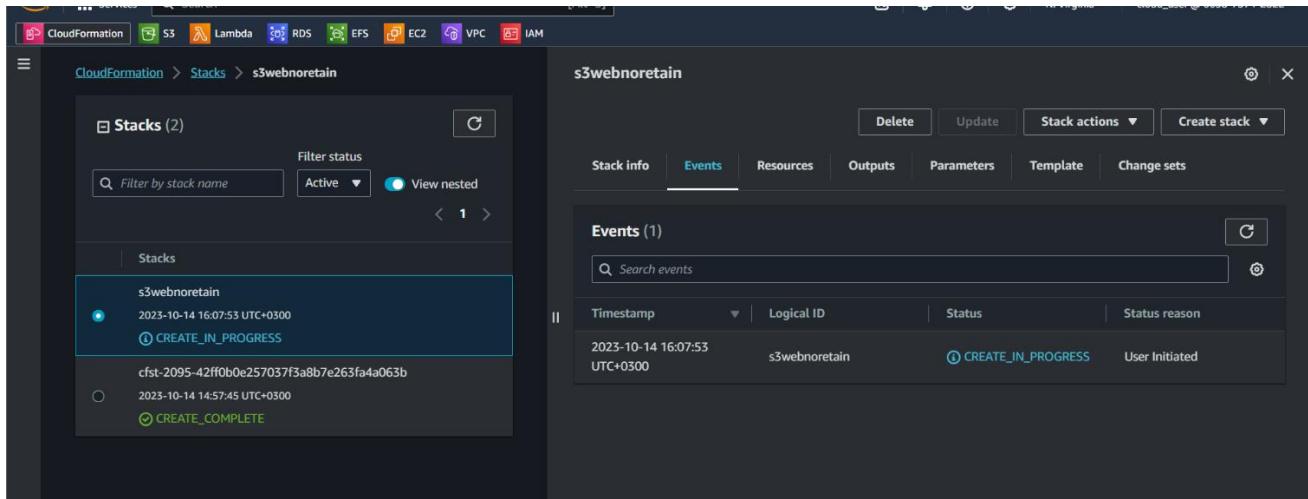
For this template, AWS CloudFormation might require an unrecognized capability: CAPABILITY_AUTO_EXPAND. Check the capabilities of these resources. [Learn more](#)

I acknowledge that AWS CloudFormation might create IAM resources with custom names.

I acknowledge that AWS CloudFormation might require the following capability: CAPABILITY_AUTO_EXPAND

Create change set Cancel Previous Submit

It will take a few minutes for it to fully be created.



Create a Nested Stack with Multiple Child Stacks

In the CloudFormation browser tab, click **Create stack**.

In the Prerequisite - Prepare template section, select **Template is ready**.

Click **Upload** a template file.

Click **Choose file**.

Select the **multinest.json** template, which is included in the lab GitHub repository.

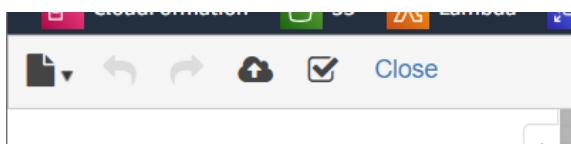
Click **View in Designer**.

```
{  
  "AWSTemplateFormatVersion" : "2010-09-09",  
  "Resources" : {  
    "myStack" : {  
      "Type" : "AWS::CloudFormation::Stack",  
      "Properties" : {  
        "TemplateURL" : "https://s3.amazonaws.com/<YOUR-BUCKET-NAME>/s3static.json",  
        "TimeoutInMinutes" : "60"  
      }  
    },  
    "myStack2" : {  
      "Type" : "AWS::CloudFormation::Stack",  
      "Properties" : {  
        "TemplateURL" : "https://s3.amazonaws.com/<YOUR-BUCKET-NAME>/noretain.json",  
        "TimeoutInMinutes" : "60"  
      }  
    }  
  }  
}
```

```
    }  
}  
}
```

In the "TemplateURL" lines, replace <YOUR-BUCKET-NAME> with the name of your bucket.

Click the checkbox at the top to **validate** the template, and then click the **cloud icon** with the up arrow to create the stack.



Click **Next**.

On the stack details page, give it a Stack name of **multinestdemo**.

Click **Next**.

Leave the settings on the stack options page, and click **Next**.

Check the boxes to accept the acknowledgements, and click **Submit**.

Timestamp	Logical ID	Status	Status reason
2023-10-14 16:17:55 UTC+0300	ccmultinest7778	CREATE_COMPLETE	-
2023-10-14 16:17:54 UTC+0300	myStack	CREATE_COMPLETE	-
2023-10-14 16:17:54 UTC+0300	myStack2	CREATE_COMPLETE	-
2023-10-14 16:17:21 UTC+0300	myStack	CREATE_IN_PROGRESS	Resource creation initiated

It will take a few minutes for it to fully be created.

Upload an **index.html** File and Browse to It

Key	Value	Description	Export name
S3BucketSecureURL	https://ccmultitest7778-myStack-4mxsc...aws.com	Name of S3 bucket to hold website content	-
WebsiteURL	http://ccmultitest7778-myStack-4mxsc...us-east-1.amazonaws.com	URL for website hosted on S3	-

404 Not Found

- Code: NoSuchKey
- Message: The specified key does not exist.
- Key: index.html
- RequestId: E8RX7061SFVD0TVE
- HostId: D0SwLNwdeAT1u1Hb/3sHiYxStg2O6v+HNrWjWAVxsGC4Ecih5QpAi...=

An Error Occurred While Attempting to Retrieve a Custom Error Document

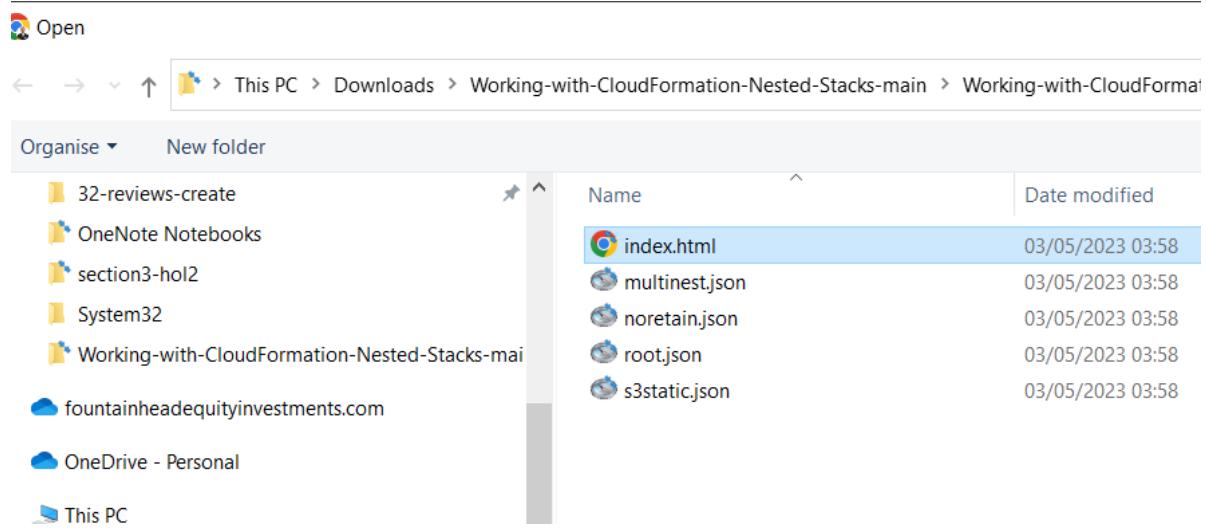
- Code: NoSuchKey
- Message: The specified key does not exist.
- Key: error.html

Name	AWS Region	Access	Creation date
ccbucket7778	US East (N. Virginia) us-east-1	Bucket and objects not public	October 14, 2023, 15:44:45 (UTC+03:00)
ccbucket7778-myStack-12kopqk3bw71p...	US East (N. Virginia) us-east-1	Bucket and objects not public	October 14, 2023, 16:13:39 (UTC+03:00)
cf-templates-1gnlzw6yzva5-us-east-1	US East (N. Virginia) us-east-1	Bucket and objects not public	October 14, 2023, 15:58:36 (UTC+03:00)
ccmultitest7778-myStack-4mxsc...nsnn...	US East (N. Virginia) us-east-1	Public	October 14, 2023, 16:17:25 (UTC+03:00)
ccmultitest7778-myStack-1qp8hkeu8hzcu...	US East (N. Virginia) us-east-1	Public	October 14, 2023, 16:17:24 (UTC+03:00)
s3webnonretain-myStack-1uz2o0tr32kbp...	US East (N. Virginia) us-east-1	Public	October 14, 2023, 16:08:00 (UTC+03:00)

In the S3 browser tab, click to open one of the nested stack buckets.

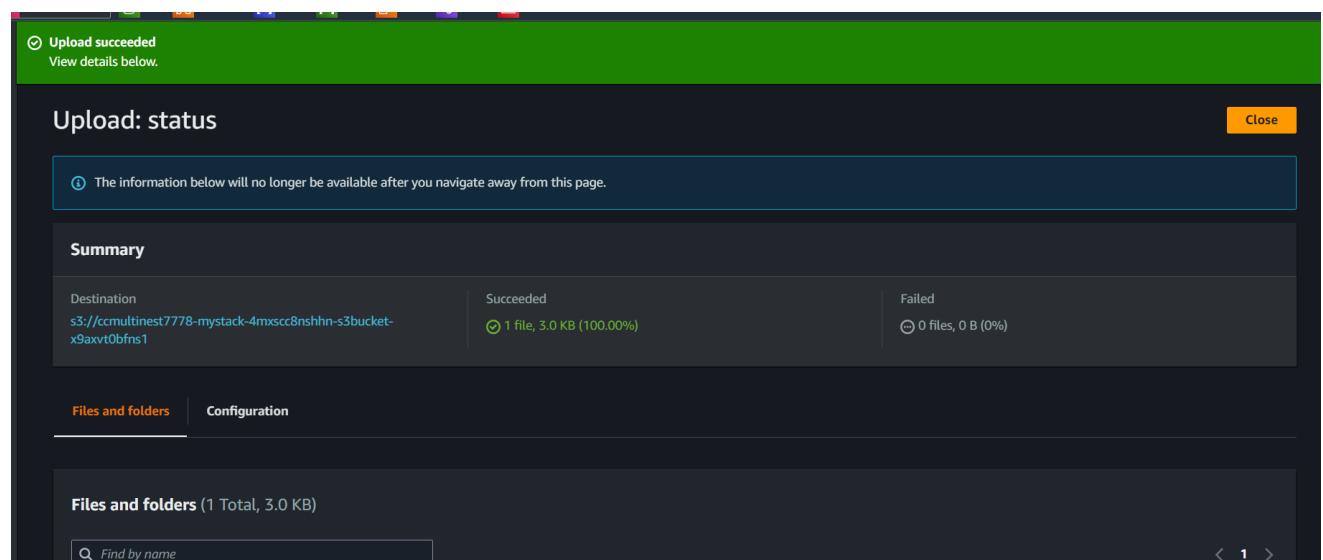
Click **Upload**.

Click Add files, and select the index.html file included in the lab GitHub repository.



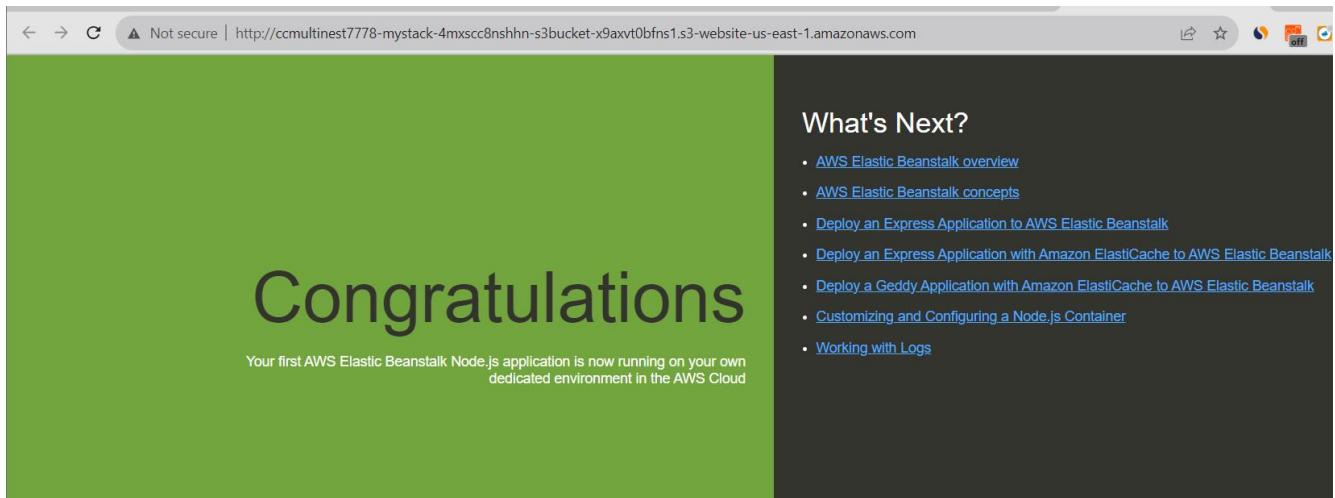
Accept the default settings on the next screens, and click **Upload**.

Click **Upload**.

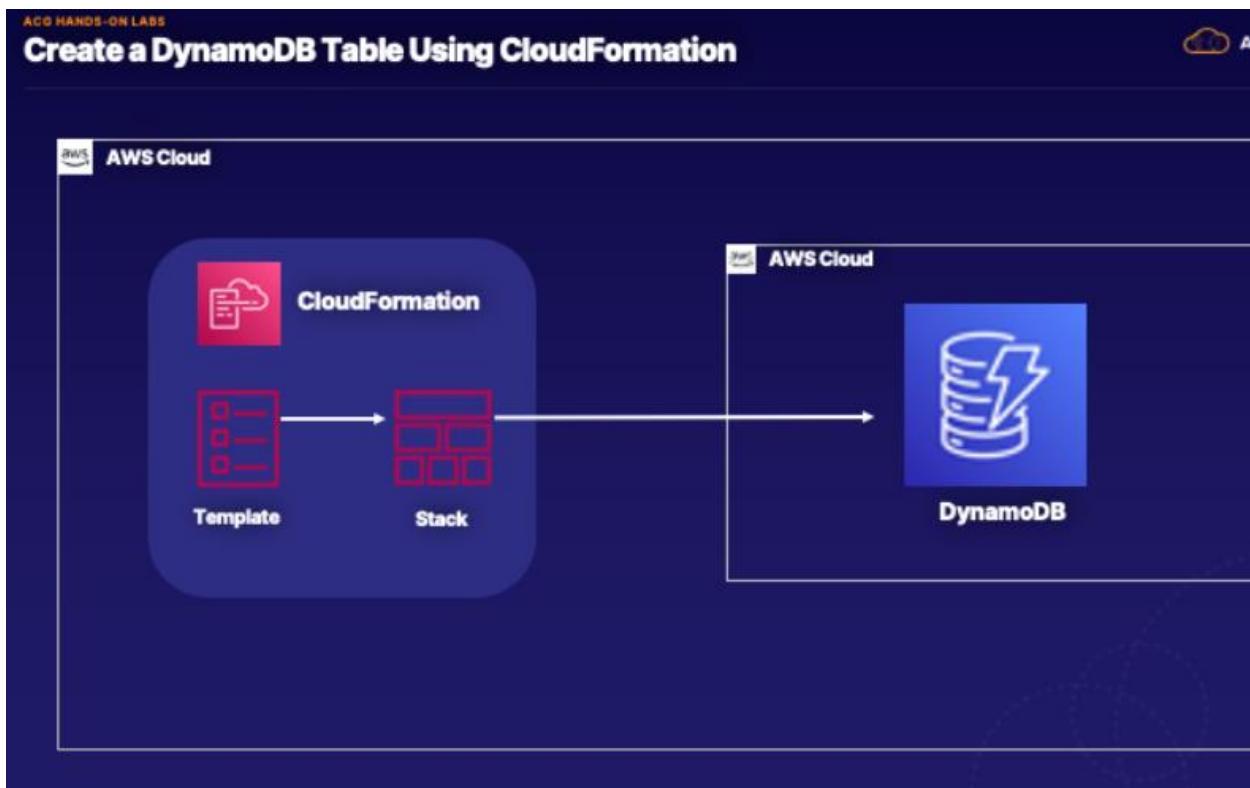


In the **CloudFormation** browser tab, select the nested stack in which you uploaded the **index.html** file.

Click the **Outputs** tab, and browse to the listed WebsiteURL. It should display an Elastic Beanstalk web page, which is fine — all that matters is if it shows up alright.



Create a DynamoDB Table Using CloudFormation



Solution

Log in to the AWS Management Console.

Make sure you're using the us-east-1 Region.

Review and Download the CloudFormation Template

Navigate to the provided CloudFormation template: **acg-dynamodb-template.yaml**.

Click **Raw**.

<https://raw.githubusercontent.com/costas778/content-aws-certified-cloud-practitioner/main/labs/create-dynamodb-table-using-cloudformation/acg-dynamodb-template.yaml>

Right-click and select Save As.

Make sure the Format is YAML, and click **Save**.

Note: You may need to add the .yaml extension to the filename.

Click **Save**.

Launch the CloudFormation Stack

From the top search bar in the AWS Management Console, search for and navigate to **CloudFormation**.

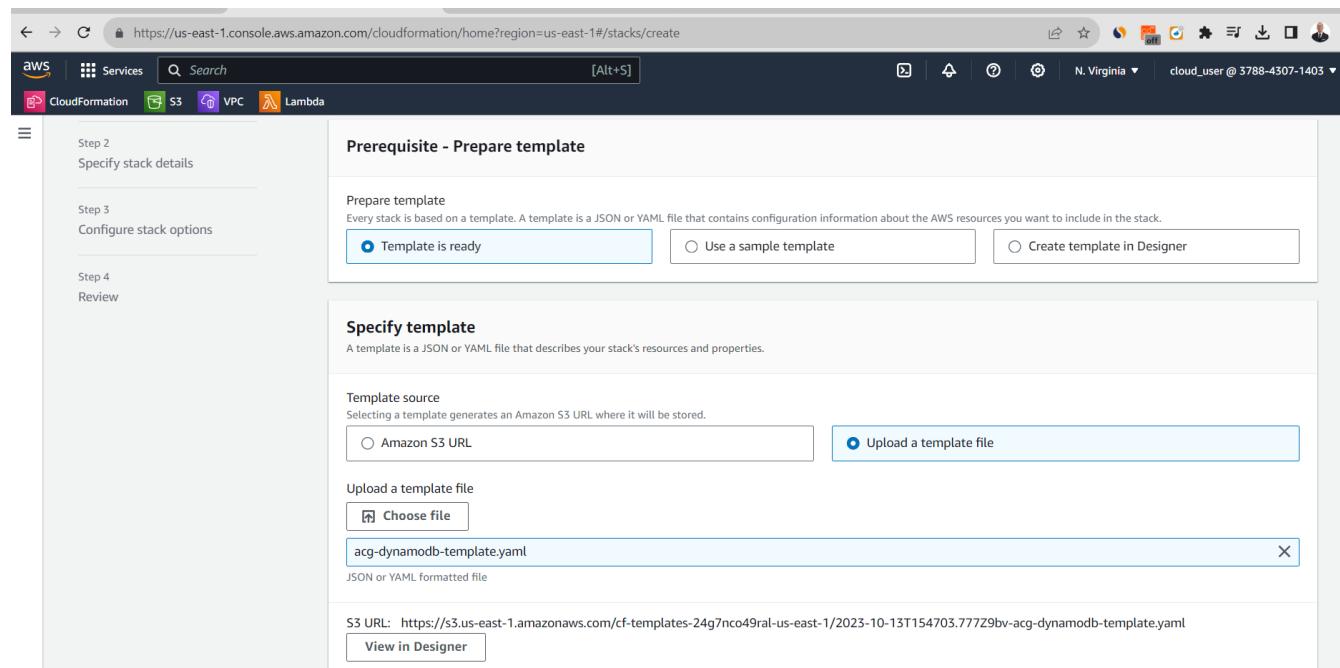
Click **Create stack > With new resources (standard)**.

Select **Template is ready**.

Select **Upload a template file**.

Click **Choose file**.

Select the **YAML** file you downloaded.



Click **Open**.

Click **Next**.

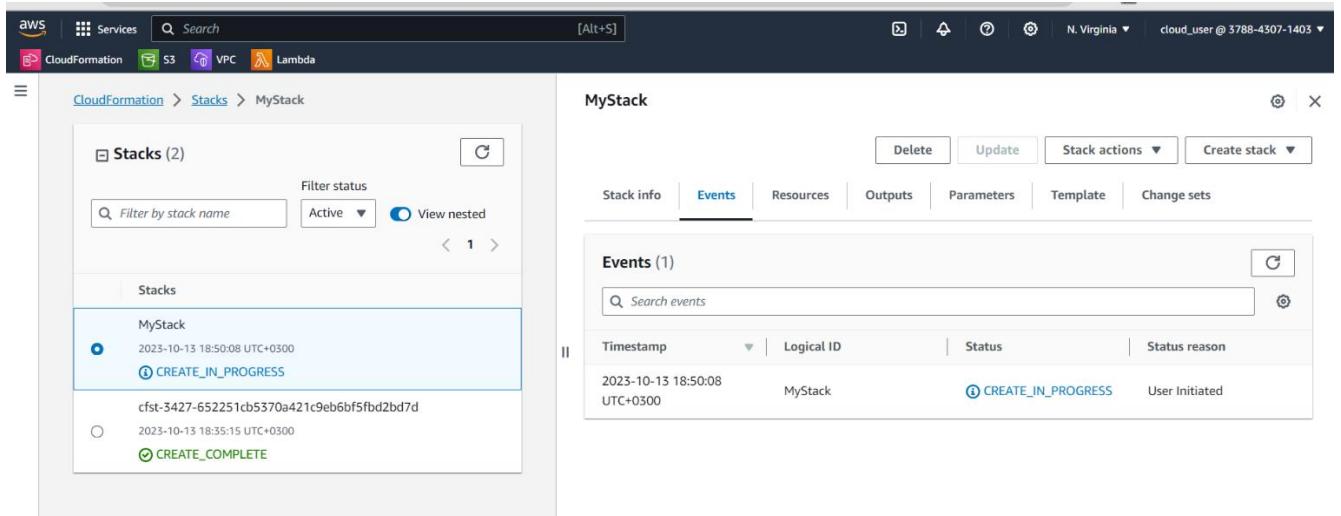
For **Stack name**, enter **MyStack**.

A screenshot of the 'Specify stack details' page. The title is 'Specify stack details'. There is a 'Stack name' field with the value 'MyStack'. Below the field is a note: 'Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes'.

Click **Next**.

On the Configure stack options page leave everything as-is, and click **Next**.

On the Review **MyStack** page, click **Create stack**.

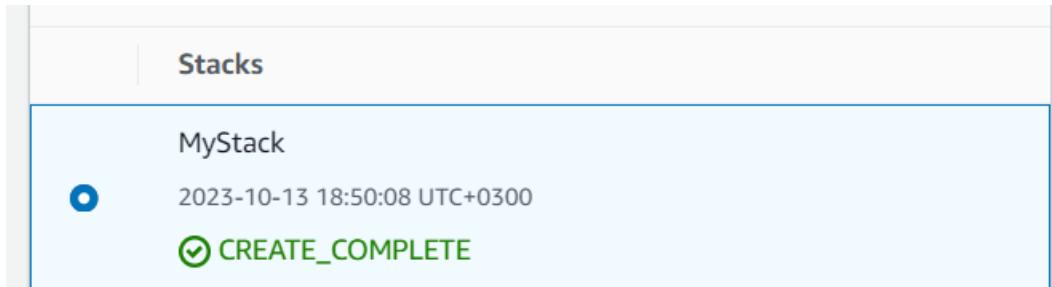


The screenshot shows the AWS CloudFormation console. On the left, a sidebar lists 'Stacks (2)' with 'MyStack' selected. The main area is titled 'MyStack' and contains tabs for 'Stack info', 'Events' (which is selected), 'Resources', 'Outputs', 'Parameters', 'Template', and 'Change sets'. Under the 'Events' tab, there is a table with one row. The table has columns for 'Timestamp', 'Logical ID', 'Status', and 'Status reason'. The data in the table is:

Timestamp	Logical ID	Status	Status reason
2023-10-13 18:50:08 UTC+0300	MyStack	CREATE_IN_PROGRESS	User Initiated

Click the **refresh** icon in the upper-right section of the Events list to check the status of the stack being created.

Note: It will take a few minutes for the stack to create. You can view the progress of the stack creation in the Status column of the Events list.



The screenshot shows the AWS CloudFormation 'Stacks' page. It displays a single stack named 'MyStack' with the following details:

- Status: **CREATE_COMPLETE**
- Timestamp: 2023-10-13 18:50:08 UTC+0300

Verify the DynamoDB Table Was Created

Once the stack is created, use the top search bar within the console to search for and navigate to **DynamoDB**.

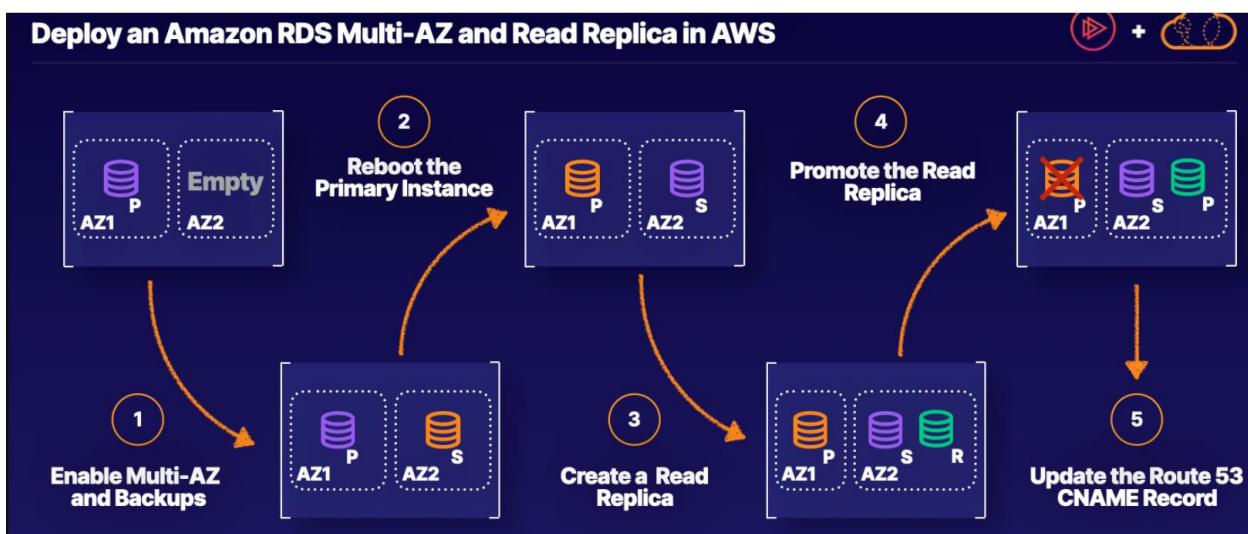
From the left menu, click **Tables**.

The screenshot shows the AWS DynamoDB Tables page. On the left, there's a sidebar with options like Dashboard, Tables, Update settings, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Reserved capacity, and Settings. Under Tables, 'Inventory' is selected. The main area shows a table titled 'Tables (1) Info'. It has columns for Name, Status, Partition key, Sort key, Indexes, Deletion protection, Read capacity mode, Write capacity mode, and Last modified. The 'Inventory' row is listed with 'Active' status, 'InventoryId' as the partition key, and 'Off' as the deletion protection setting. There are buttons for Actions, Delete, and Create table.

Click the listed Inventory table. While there aren't currently any items within the table, it is ready for items to be added.

The screenshot shows the AWS DynamoDB Inventory table details page. The left sidebar is identical to the previous screenshot. The main area has tabs for Overview, Indexes, Monitor, Global tables, Backups, Exports and streams, and a dropdown for sorting. The Overview tab is selected. It shows a summary section with a note about Point-in-time recovery (PITR), a general information table, and an additional info section. The general information table includes fields for Partition key (InventoryId String), Sort key (-), Capacity mode (Provisioned), and Table status (Active). The additional info section has a link to 'Edit PITR'.

Deploy an Amazon RDS Multi-AZ and Read Replica in AWS



Introduction

We will work with Relational Database Service (RDS). We will enable Multi-AZ and backups, create a read replica, promoting a read replica, and updating the RDS endpoint in Route 53.

Multi-AZ and read replicas serve different purposes with RDS. Multi-AZ is strictly for failover, as the standby instances cannot be read from by an application. Read replicas are used for improved performance and migrations.

With read replicas, you can write to the primary database and read from the read replica. Since a read replica can be promoted to be the primary database, it makes for a great tool in disaster recovery and migrations.

Solution

Log in to the AWS Management Console.

Make sure you're in the N. Virginia (us-east-1) Region throughout the lab.

Enable Multi-AZ Deployment

Configure Your Multi-AZ Instance

Note: The target group for the load balancer may need additional time to provision.

Before starting the lab, navigate to **EC2 > Target Groups** and select TG1 to ensure you have at least one instance registered as healthy. If your target group shows no instances are healthy, then wait a few more minutes until one changes.

From the **EC2** service page, scroll down to **Load Balancing** and select **Load Balancers**.

The screenshot shows the AWS Management Console with the EC2 service selected. Under the 'Load balancers' section, there is one entry:

Name	DNS name	State	VPC ID	Availability
load-balancer	load-balancer-1394174152.us-east-1.elb.amazonaws.com	Active	vpc-0817cbcaaefcd280	2 Availability

Below the table, it says '0 load balancers selected'.

Select the load balancer and then copy the DNS name of the load balancer.

load-balancer-1394174152.us-east-1.elb.amazonaws.com

Open a new browser tab, and paste your copied DNS name.

The browser window shows the URL 'http://load-balancer-1394174152.us-east-1.elb.amazonaws.com'. The page content is:

Mindblown: a blog about philosophy.

Hello world!

Welcome to WordPress. This is your first post.
Edit or delete it, then start writing!

October 14, 2023

You will use this web page to test failovers and promotions in this lab.

Go back to the AWS Management Console, and navigate to **RDS** using the Services menu or the unified search bar.

In the Amazon **RDS** sidebar menu, select **Databases**.

Databases (1)										Create database
		Group resources		C	Modify		Actions ▾	Restore from S3		Create database
<input type="text"/> Filter by databases										< 1 >
	DB identifier ▲	Status ▾	Role ▾	Engine ▾	Region & AZ ▾	Size ▾	Actions ▾	CPU ▾	Current ac	
<input checked="" type="radio"/>	wordpress	Available	Instance	MySQL Community	us-east-1b	db.t3.micro	3 Actions	2.98%	1	

Select the radio button to the left of your wordpress database details.

Along the top of the page, click **Modify**.

The database's configuration options may take a moment to load.

Enable Multi-AZ deployment:

In the **Availability & durability** section, select **Create a standby instance (recommended for production usage)**.

Availability & durability

Multi-AZ deployment [Info](#)

Create a standby instance (recommended for production usage)
Creates a standby in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

Do not create a standby instance

Scroll to the bottom of the page, and click **Continue**.

Summary of modifications

You are about to submit the following modifications. Only values that will change are displayed. Carefully verify your changes and click Modify DB Instance.

Attribute	Current value	New value
Multi-AZ deployment	No	Yes

Schedule modifications

When to apply modifications

- Apply during the next scheduled maintenance window

Current maintenance window: October 16, 2023 12:07 - 12:37 UTC+3

- Apply immediately

The modifications in this request and any pending modifications will be asynchronously applied as soon as possible, regardless of the maintenance window setting for this database instance.



Potential performance impact when converting to Multi-AZ

Your DB instance can experience a significant performance impact during and after converting to a Multi-AZ deployment. The impact is greater on DB instances with large amounts of storage and write-intensive workloads. We don't recommend this conversion on a production DB instance.

Cancel

Back

Modify DB instance

In the Schedule modifications section, select **Apply immediately**.

Click **Modify DB instance**.

After a few moments, click **Refresh**.

You should see the Status change to **Modifying**. If you don't see this change, you may need to click Refresh again. The instance may take up to 10 minutes to enable **Multi-AZ**.

Databases (1)		Group resources	Modify	Actions	Restore from S3	Create database
		Filter by databases				
DB identifier	Status	Role	Engine	Region & AZ	Size	CPU
wordpress	Modifying	Instance	MySQL Community	us-east-1b	db.t3.micro	3 Actions 3.31%

Databases (1)			
<input type="checkbox"/> <input checked="" type="radio"/> <input type="checkbox"/>			
<input type="text"/> Filter by databases			
DB identifier	Status	Role	
wordpress	 Available	Instance	

Verify the Multi-AZ Deployment

After the instance is available again, verify that **Multi-AZ** is enabled:

In the Amazon RDS sidebar menu, select **Events**.

Events (8)			
<input type="text"/> Filter by event			
Source	Type	Time	Message
wordpress	Instances	October 14, 2023, 20:09 (UTC+03:00)	Finished applying modification to convert to a Multi-AZ DB Instance
wordpress	Instances	October 14, 2023, 20:03 (UTC+03:00)	Applying modification to convert to a Multi-AZ DB Instance
wordpress	Instances	October 14, 2023, 19:41 (UTC+03:00)	Finished DB Instance backup
rds:wordpress-2023-10-14-16-38	Snapshots	October 14, 2023, 19:41 (UTC+03:00)	Automated snapshot created
wordpress	Instances	October 14, 2023, 19:39 (UTC+03:00)	Backing up DB instance
rds:wordpress-2023-10-14-16-38	Snapshots	October 14, 2023, 19:39 (UTC+03:00)	Creating automated snapshot
wordpress	Instances	October 14, 2023, 19:38 (UTC+03:00)	DB instance created
wordpress	Instances	October 14, 2023, 19:37 (UTC+03:00)	DB instance restarted

Review the events history for your instance.

You should see an event for converting the instance to Multi-AZ.

Reboot the instance to simulate the Multi-AZ redundancy:

In the Amazon RDS sidebar menu, select **Databases**.

Ensure the radio button to the left of your instance is selected.

Use the **Actions** dropdown to select **Reboot**.

The screenshot shows the AWS RDS Databases page. At the top, there's a blue banner with the text "Introducing Aurora I/O-Optimized" and a note about Aurora's I/O-Optimized storage configuration. Below the banner, the page title is "RDS > Databases". A callout box provides information about Blue/Green Deployments. The main table lists one database instance:

DB identifier	Status	Role	Engine	Region & AZ	Size	Actions	CPU	Current ac
wordpress	Available	Instance	MySQL Community	us-east-1b	db.t3.micro	3 Actions	3.51%	0

A vertical menu on the right side includes options like "Convert to Multi-AZ deployment", "Stop temporarily", "Reboot", "Delete", "Set up EC2 connection", "Set up Lambda connection", "Create read replica", "Create Aurora read replica", "Create Blue/Green Deployment - new", and "Create database".

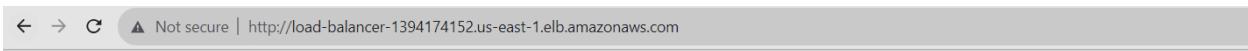
On the Reboot DB Instance page, select **Reboot With Failover?** and click **Confirm**.

The screenshot shows a confirmation dialog titled "Reboot DB Instance". It contains a section for "DB Instances" with the question "Are you sure you want to reboot these DB Instance(s)?". A list of instances is shown, with "wordpress" selected. A checkbox labeled "Reboot With Failover?" is checked. At the bottom, there are "Cancel" and "Confirm" buttons.

This ensures the secondary instance takes over when this instance goes down.
The reboot takes a few moments.

Databases (1)		
<input type="text"/> Filter by databases		
	DB identifier	Status
	wordpress	Rebooting

Navigate to the DNS web page tab and refresh the page.



Wordpress

Mindblown: a blog about philosophy.

Hello world!

Welcome to WordPress. This is your first post.
Edit or delete it, then start writing!

October 14, 2023

The web page will experience only a small amount of downtime while the database is being modified (which causes a temporary 502 error).

After the reboot is complete, the Multi-AZ standby is now the primary.

Navigate back to the AWS Management Console and use the Amazon RDS sidebar to select **Events** (you may need to expand the sidebar menu).

Review the events history for your instance.

Events (12)				
<input type="text"/> Filter by event				
Source	Type	Time	Message	
wordpress	Instances	October 14, 2023, 20:16 (UTC+03:00)	The user requested a failover of the DB instance.	
wordpress	Instances	October 14, 2023, 20:16 (UTC+03:00)	Multi-AZ instance failover completed	
wordpress	Instances	October 14, 2023, 20:16 (UTC+03:00)	DB instance restarted	
wordpress	Instances	October 14, 2023, 20:15 (UTC+03:00)	Multi-AZ instance failover started.	

You should see events for the Multi-AZ failover listed.

Create a Read Replica

In the Amazon RDS sidebar menu, select **Databases** to return to your instance details.

Ensure the radio button to the left of your instance is selected.

Use the **Actions** dropdown to select **Create read replica**.

Settings

Replica source
Source DB instance identifier

wordpress
 Role: Instance

DB instance identifier
DB instance identifier. This is the unique key that identifies a DB instance. This parameter is stored as a lowercase string (for example, mydbinstance).

wordpress-rr

In the Settings section, enter **wordpress-rr** into the DB instance identifier field.

In the AWS Region section, ensure the Destination Region is set to US East (N. Virginia).

Leave the other default settings and click **Create read replica** at the bottom of the page.

Create read replica

Databases (1)										
<input type="checkbox"/> Group resources <input type="button" value="C"/> Actions <input type="button" value="Restore from S3"/> <input type="button" value="Create database"/>										
<input type="text"/> Filter by databases										
DB identifier	Status	Role	Engine	Region & AZ	Size	Actions	CPU	Current activity		
wordpress-rr	Creating	Replica	MySQL Community	-	db.t3.micro	-	-	-		

Databases (2)										
<input type="checkbox"/> Group resources <input type="button" value="C"/> Actions <input type="button" value="Restore from S3"/> <input type="button" value="Create database"/>										
<input type="text"/> Filter by databases										
DB identifier	Status	Role	Engine	Region & AZ	Size	Actions	CPU	Current		
wordpress	Modifying	Primary	MySQL Community	us-east-1a	db.t3.micro	3 Actions	3.94%			
wordpress-rr	Creating	Replica	MySQL Community	-	db.t3.micro	-	-	-		

The read replica will take about **5-10 minutes** to become available.

Databases (2)										
<input type="checkbox"/> Group resources <input type="button" value="C"/> Actions <input type="button" value="Restore from S3"/> <input type="button" value="Create database"/>										
<input type="text"/> Filter by databases										
DB identifier	Status	Role	Engine	Region & AZ	Size	Actions	CPU	Current a		
wordpress	Available	Primary	MySQL Community	us-east-1a	db.t3.micro	2 Actions	3.78%	1		
wordpress-rr	Available	Replica	MySQL Community	us-east-1a	db.t3.micro	2 Actions	3.47%	0		

Navigate to the DNS web page tab and refresh the page.

The web page should stay up because the read replica changes are **asynchronous**.

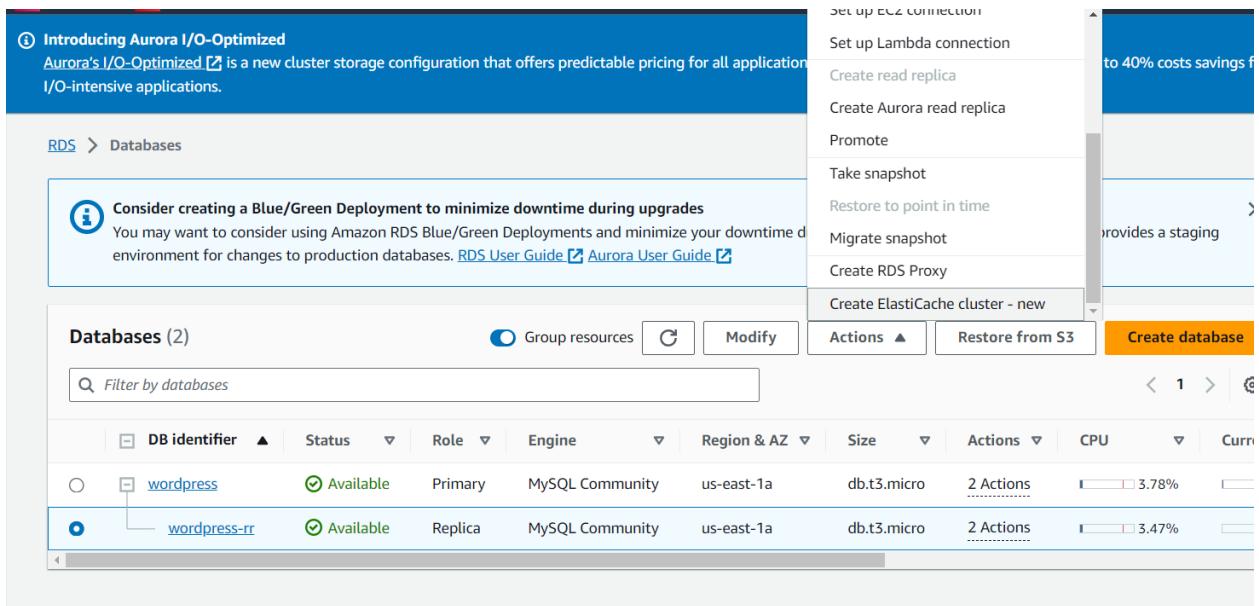
Promote the Read Replica and Change the CNAME Record in Route 53 to Point to the New Endpoint

Promote the Read Replica

Navigate back to the AWS Management Console.

After the read replica is available, select the radio button to the left of it.

Use the Actions dropdown to select **Promote**.



Leave all the defaults and click **Promote read replica** at the bottom of the page.

After a few moments, click Refresh.

You should see that the read replica's Role has changed from **Replica** to **Instance**, and that the Status is now **Modifying**.

Change the CNAME Record to Point to the New Endpoint

After the read replica instance is available, select the `wordpress-rr` instance name.

On the **Connectivity & security tab**, copy the Endpoint to your clipboard.

wordpress-rr.ccdcpvypae41.us-east-1.rds.amazonaws.com

Open **Route 53** in a new tab.

You can disregard any error messages that display.

In the **Route 53** sidebar menu, select **Hosted zones**.

The screenshot shows the AWS Route 53 Hosted zones page. At the top, it says "Route 53 > Hosted zones". Below that, it says "Hosted zones (1/1)". There is a note: "Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings." Below the note are four buttons: "View details", "Edit", "Delete", and a prominent orange "Create hosted zone" button. A search bar with the placeholder "Filter records by property or value" is below the buttons. To the right of the search bar are navigation arrows and a refresh icon. The main table has columns: "Hosted zone name", "Type", "Created by", "Record count", "Description", and "Hosted ...". One row is selected, showing "mydomain.local" under "Hosted zone name", "Private" under "Type", "Route 53" under "Created by", "3" under "Record count", "My private zone f..." under "Description", and "Z0892734..." under "Hosted ...".

Select the **mydomain.local** hosted zone name.

In the hosted zone details, you should see a **CNAME** record that points to your wordpress database endpoint.

Check the **checkbox** to the left of the **CNAME** record.

The screenshot shows the "Hosted zone details" page for the "mydomain.local" zone. At the top, it says "Hosted zone details" and "Edit hosted zone". Below that, there are tabs: "Records (3)" (which is selected) and "Hosted zone tags (1)". The "Records" section has a sub-header "Records (1/3) Info" and a note: "Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings." It includes buttons: "Delete record", "Import zone file", and a prominent orange "Create record" button. A search bar with the placeholder "Filter records by property or value" is followed by dropdown menus for "Type", "Routing policy", and "Alias". Navigation arrows and a refresh icon are to the right. The main table has columns: "Record name", "Type", "Routing...", "Differ...", and "Alias". Three records are listed: 1. "mydomain.local" (NS, Simple, No Alias). 2. "mydomain.local" (SOA, Simple, No Alias). 3. "db.mydomain.local" (CNAME, Simple, No Alias), which has a checked checkbox to its left. To the right of the table is a "Record details" pane. It has a "Edit record" button at the top. Below it, the record details are shown: Record name (db.mydomain.local), Record type (CNAME), Value (wordpress.ccdcpvypae41.us-east-1.rds.amazonaws.com), Alias (No), TTL (seconds) (60), and Routing policy (Simple).

In the Record details pane on the right, click **Edit record**.

Replace the existing Value with your copied read replica endpoint.

Value | [Info](#)

wordpress-rr.ccdcpvypae41.us-east-1.rds.amazonaws.com

Enter multiple values on separate lines.

Click **Save**.

Navigate to the DNS web page tab and refresh the page.

You should see no downtime.

Addendum

The above had certain things in place so the following are provided to help create the pre requests (as well as, pretty much, the whole environment). See it as bonus!

[How to create a Amazon RDS Multi-AZ with load balancer](#)

Using AWS CloudShell:

[Step 1: Access the AWS CloudShell](#)

Log in to your AWS Management Console.

Click on "Services" at the top, and then choose "CloudShell" from the "Developer Tools" section

[Step 2: Configure Your CloudShell Environment](#)

If you're using CloudShell for the first time, it might take a moment to set up the environment.
Make sure you have the AWS CLI (Command Line Interface) configured with the necessary permissions.

[Step 3: Create a Multi-AZ RDS Instance with Read Replicas](#)

```
# Create the RDS Multi-AZ instance
aws rds create-db-instance \
--db-instance-identifier my-multi-az-db \
--allocated-storage 20 \
--db-instance-class db.t2.micro \
--engine mysql \
--master-username your_master_username \
--master-user-password your_master_password \
--availability-zone us-east-1a \
--multi-az
```

```
# Create a Read Replica
aws rds create-db-instance-read-replica \
--db-instance-identifier my-read-replica \
--source-db-instance-identifier my-multi-az-db
```

Step 4: Set Up a DB Proxy

```
# Create a DB Proxy
aws rds create-db-proxy \
```

```
--db-proxy-name my-db-proxy \
--debug-logging \
--engine-family MYSQL \
--role-arn arn:aws:iam::123456789012:role/MyDBProxyRole \
--auth \
  db-user=your_master_username,db-password=your_master_password \
--vpc-security-group-ids sg-01234567890abcdef0
```

Step 5: Configure Load Balancing

```
# Add Target Group
aws rds add-tags-to-resource \
  --resource-name arn:aws:rds:us-east-1:123456789012:cluster:my-db-proxy/your-db-proxy-name \
  --tags Key=Name,Value=my-db-proxy

# Create an Application Load Balancer in the AWS Management Console
# and configure it to route traffic to your RDS DB proxy.
```

Using the AWS Management Console:

Step 1: Create a Multi-AZ RDS Instance

Log in to your AWS Management Console.

Navigate to the RDS service.

Click on "Create Database."

Choose "Standard Create" and then select the database engine you want to use.

In the "Templates" section, select "Dev/Test" or "Production" as per your requirements.

In the "DB details" section, provide the necessary information such as DB instance identifier, master username, and password.

In the "Settings" section, configure your Multi-AZ settings. Select "Enable" for Multi-AZ deployment.

Complete the wizard and click "Create database."

Step 2: Create a Read Replica

After the Multi-AZ instance is created, navigate to the RDS service.

Select the Multi-AZ instance you just created.

Scroll down to the "Read Replicas" section and click "Add read replica."

Configure the replica settings and click "Add read replica."

Step 3: Set Up a DB Proxy

Navigate to the RDS service.

In the left navigation pane, click on "Proxies."

Click "Create proxy."

Provide a name for the proxy, and configure the settings, including the role ARN, debug logging, and security group.

Click "Create proxy."

Step 4: Configure Load Balancing

Create an Application Load Balancer (ALB) in the EC2 service.

Configure the ALB to route traffic to your RDS DB proxy.

How to setup a RDS Instance, with WordPress installed, and a load balancer.

NOTE: The following is closest to what pre requisites to setup the Project we went through.

Step 1: Set up the RDS Instance:

Log in to your AWS Console.

Go to the **RDS** service.

Click on "**Create database.**"

Choose the database engine (e.g., MySQL), and follow the wizard to configure your database instance. Make sure to note down the master username and password that you set.

Choose your database instance class, storage, and configure advanced settings as per your requirements. For simplicity, you can select the default settings for the initial setup.

Create the **RDS** instance.

Step 2: Set up an EC2 Instance:

Go to the EC2 service.

Launch an EC2 instance with the **Amazon Linux 2 AMI** or a similar one. You can use an instance type that suits your needs.

Configure security groups to allow inbound traffic on ports 80 (HTTP) and 443 (HTTPS).

Connect to your EC2 instance using SSH.

Install WordPress on your EC2 instance. You can use the following commands:

```
sudo yum update -y  
sudo yum install httpd php php-mysqlnd php-gd php-xml php-mbstring mysql -y  
sudo systemctl start httpd  
sudo systemctl enable httpd  
sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2  
sudo systemctl restart httpd
```

Download and configure WordPress:

```
wget https://wordpress.org/latest.tar.gz  
tar -xzf latest.tar.gz  
sudo mv wordpress /var/www/html/  
sudo chown -R apache:apache /var/www/html/wordpress  
sudo chmod -R 755 /var/www/html/wordpress
```

Create a WordPress configuration file:

```
cp /var/www/html/wordpress/wp-config-sample.php /var/www/html/wordpress/wp-config.php  
nano /var/www/html/wordpress/wp-config.php
```

Update the database configuration with the RDS endpoint, username, and password you noted earlier.

Configure your WordPress site via the web browser by accessing your EC2 instance's public IP or DNS.

Step 3: Set up an Application Load Balancer:

Go to the EC2 service.

Under "**Load Balancing**," select "**Load Balancers**."

Click on "**Create Load Balancer**."

Choose the "**Application Load Balancer**" type.

Configure the load balancer settings, including security groups and listeners. Make sure to add listeners for HTTP (80) and HTTPS (443).

Configure target groups to route traffic to your EC2 instances.

Add your EC2 instance to the target group.

Complete the load balancer creation process.

Step 4: Update DNS:

Create a Route 53 record set to point to your Application Load Balancer's DNS name.

Update your domain's DNS settings to use the Route 53 name servers.

Step 5: Test the Configuration:

Wait for DNS propagation.

Access your domain name via a web browser. You should see your WordPress site, which is now being served by your EC2 instances behind the load balancer.

This setup creates a basic WordPress website with EC2 instances behind a load balancer, using RDS as the database.