# Task 1

NOTE: Ensure that the host device has Ansible, boto and AWS CLI configured using the following commands:

```
sudo apt install ansible
pip3 install boto
aws configure
```

**Steps:**

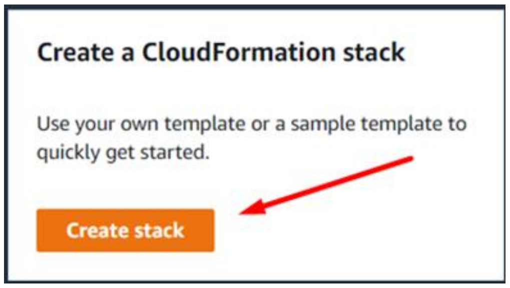
1. Once the set up is completed create a YAML file called Deployment8.yaml using the following:

```
Description:
  This template deploys a VPC, with a pair of public and private subnets spread
  across two Availability Zones. It deploys an internet gateway, with a default
  route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
  and default routes for them in the private subnets.

Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String

  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 192.168.0.0/16

  PublicSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in the first Availability Zone
    Type: String
    Default: 192.168.0.0/18

  PublicSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in the second Availability Zone
    Type: String
    Default: 192.168.64.0/18

  PrivateSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet in the first Availability Zone
    Type: String
    Default: 192.168.128.0/18

  PrivateSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet in the second Availability Zone
    Type: String
    Default: 192.168.192.0/18

  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instance
    Type: AWS::EC2::KeyPair::KeyName

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGatewayAttachment:
```

```
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC

PublicSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [0, !GetAZs ""]
    CidrBlock: !Ref PublicSubnet1CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [1, !GetAZs ""]
    CidrBlock: !Ref PublicSubnet2CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [0, !GetAZs ""]
    CidrBlock: !Ref PrivateSubnet1CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [1, !GetAZs ""]
    CidrBlock: !Ref PrivateSubnet2CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc

NatGateway2EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc

NatGateway1:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway1EIP.AllocationId
    SubnetId: !Ref PublicSubnet1

NatGateway2:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway2EIP.AllocationId
    SubnetId: !Ref PublicSubnet2

PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Routes

DefaultPublicRoute:
  Type: AWS::EC2::Route
  DependsOn: InternetGatewayAttachment
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet2

PrivateRouteTable1:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ1)

DefaultPrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1

PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1

PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ2)

DefaultPrivateRoute2:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2

JenkinsControllerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: "Security group that allows SSH from anywhere"
    GroupName: "JenkinsController"
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 8080
        ToPort: 8080
        CidrIp: 0.0.0.0/0
    VpcId: !Ref VPC

JenkinsControllerEC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: ami-09e67e426f25ce0d7
    InstanceType: t2.micro
    SubnetId: !Ref PublicSubnet1
    KeyName: !Ref KeyName
    SecurityGroupIds:
      - !Ref JenkinsControllerSecurityGroup
    Tags:
      - Key: "Name"
        Value: "Jenkins Controller"

JenkinsAgentSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: "Security group that allows SSH from anywhere"
    GroupName: "JenkinsAgent"
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
    VpcId: !Ref VPC

JenkinsAgentEC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: ami-09e67e426f25ce0d7
    InstanceType: t2.micro
    SubnetId: !Ref PublicSubnet1
    KeyName: !Ref KeyName
    SecurityGroupIds:
      - !Ref JenkinsAgentSecurityGroup
    Tags:
      - Key: "Name"
        Value: "Jenkins Agent"

ProductionSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: "Security group that allows SSH from anywhere"
    GroupName: "Production"
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
    VpcId: !Ref VPC

ProductionEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-09e67e426f25ce0d7
      InstanceType: t2.micro
      SubnetId: !Ref PublicSubnet1
      KeyName: !Ref KeyName
      SecurityGroupIds:
        - !Ref ProductionSecurityGroup
      Tags:
        - Key: "Name"
          Value: "Production"

  Deploy08DBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: "Security group for the RDS MySQL database that allows access from Production/Agent SG only"
      GroupName: "Deploy08-DB"
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 3306
          ToPort: 3306
          SourceSecurityGroupId:
            Fn::GetAtt:
              - ProductionSecurityGroup
              - GroupId
      VpcId: !Ref VPC

Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

  JenkinsControllerEC2Instance:
    Value: !GetAtt JenkinsControllerEC2Instance.PublicIp
    Description: JenkinsController's PublicIp Address
```

```
JenkinsAgentEC2Instance:
  Value: !GetAtt JenkinsAgentEC2Instance.PublicIp
  Description: JenkinsAgentEC2Instance's PublicIp Address

ProductionEC2Instance:
  Value: !GetAtt ProductionEC2Instance.PublicIp
  Description: ProductionEC2Instance's PublicIp Address
```

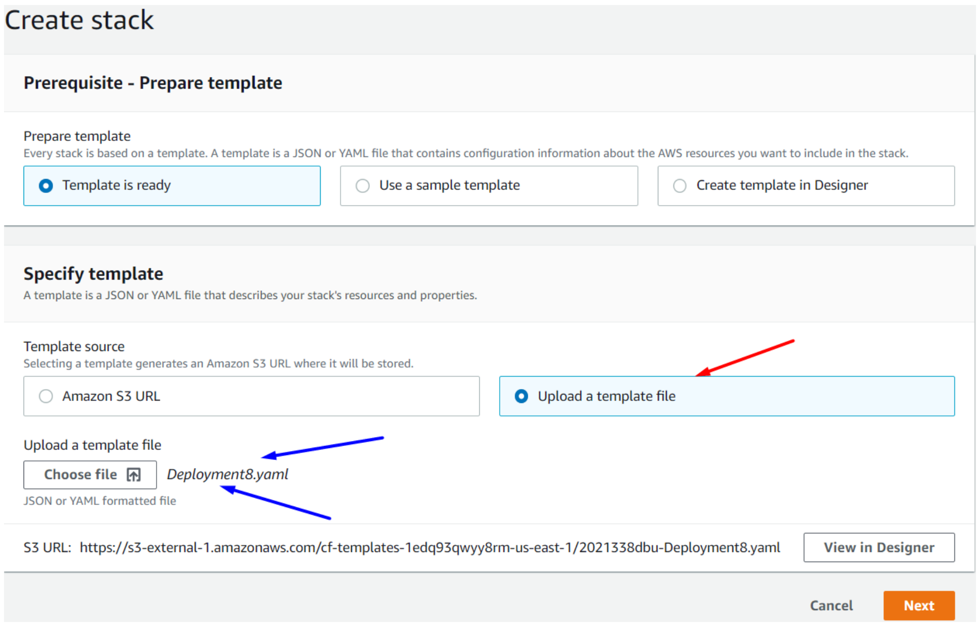
2. Once the file is created, save it and got to AWS CloudFormation to create a stack. Ensure you're in the correct region.



Create a CloudFormation stack

Use your own template or a sample template to quickly get started.

3. Create your stack and add the yaml file as shown below:



Create stack

Prerequisite - Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

[✓ Template is ready] Use a sample template Create template in Designer

Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.

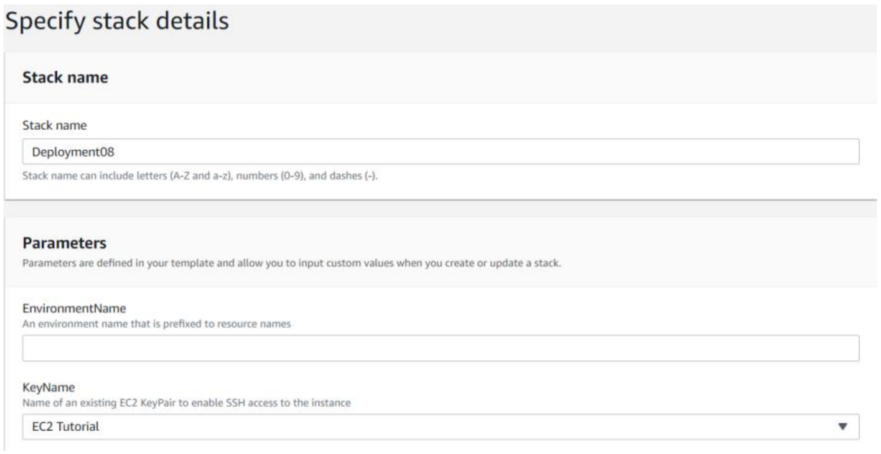
Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL [✓ Upload a template file]

Upload a template file
Choose file Deployment8.yaml
JSON or YAML formatted file

S3 URL: https://s3-external-1.amazonaws.com/cf-templates-1edq93qwyy8rm-us-east-1/2021338dbu-Deployment8.yaml

4. Give the stack a name and include the Key Pair you'll use to SSH into the ec2 instances.



Specify stack details

**Stack name**

Stack name

Deployment08

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.

EnvironmentName

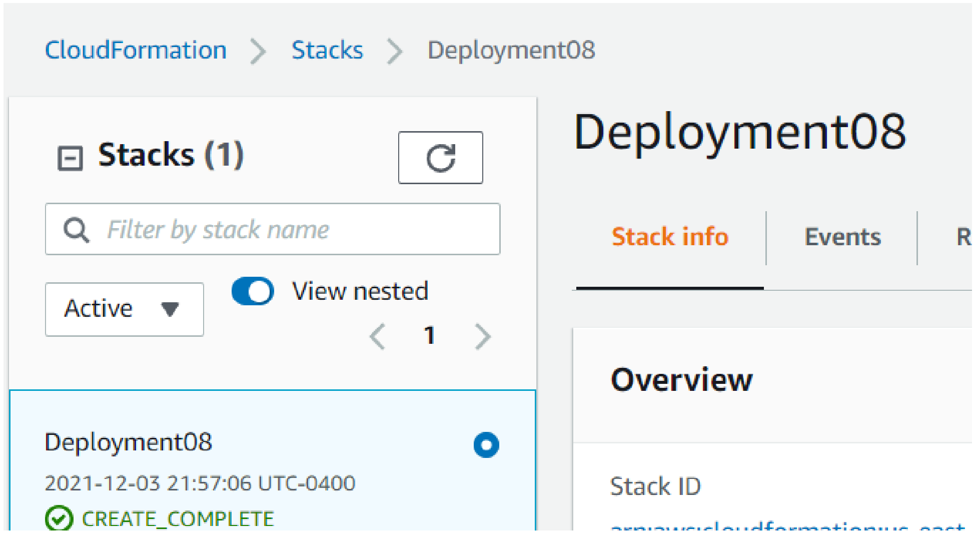
An environment name that is prefixed to resource names

KeyName

Name of an existing EC2 KeyPair to enable SSH access to the instance

EC2 Tutorial

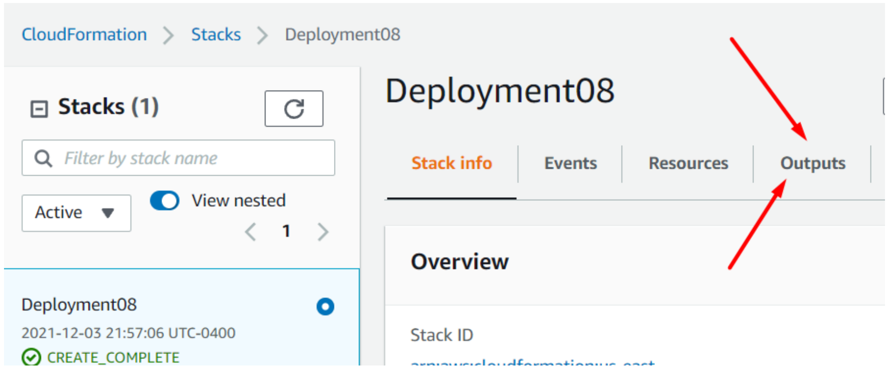
5. Keep all the default settings and create the stack. Then wait for it to be completed



CloudFormation > Stacks > Deployment08

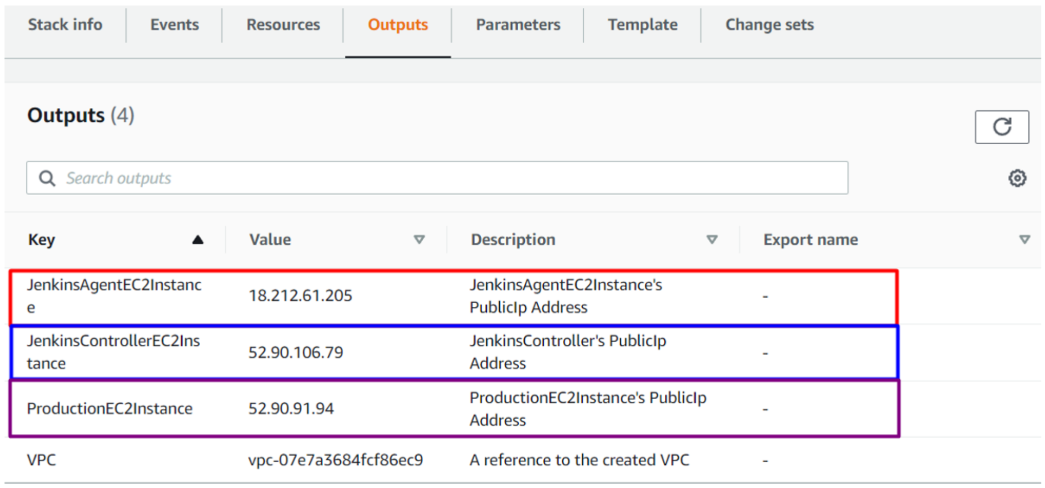
Deployment08
2021-12-03 21:57:06 UTC-0400
CREATE_COMPLETE

6. Then go to the Outputs tab to make sure everything was created successfully.



CloudFormation > Stacks > Deployment08

Deployment08
2021-12-03 21:57:06 UTC-0400
CREATE_COMPLETE



Outputs (4)

| Key | Value | Description | Export name |
|---|---|---|---|
| JenkinsAgentEC2Instance | 18.212.61.205 | JenkinsAgentEC2Instance's PublicIp Address | - |
| JenkinsControllerEC2Instance | 52.90.106.79 | JenkinsController's PublicIp Address | - |
| ProductionEC2Instance | 52.90.91.94 | ProductionEC2Instance's PublicIp Address | - |
| VPC | vpc-07e7a3684fcf86ec9 | A reference to the created VPC | - |

7. Record by taking **note** of the Public IPv4 address of all the EC2 instances created in the following format since we'll need to edit our host file later on.

[Controller]

**52.90.106.79** ansible_user=ubuntu ansible_ssh_private_key_file=~/.ssh/**EC2Tutorial.pem**

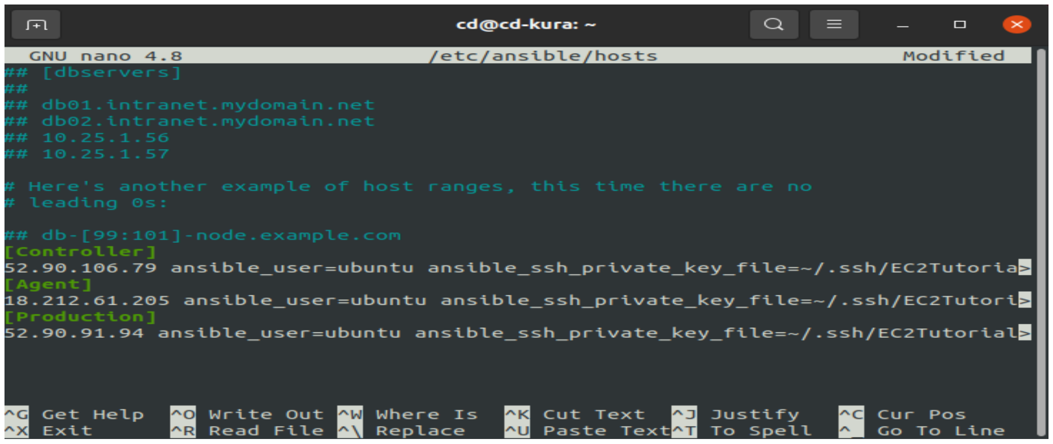
[Agent]

**18.212.61.205** ansible_user=ubuntu ansible_ssh_private_key_file=~/.ssh/**EC2Tutorial.pem**

[Production]

**52.90.91.94** ansible_user=ubuntu ansible_ssh_private_key_file=~/.ssh/**EC2Tutorial.pem**

8. Once everything is set up proceed to set up Ansible so that it configures the EC2s. Change into the SSH directory and make sure the Key that is being used to SSH into the EC2 instance is present. Use the following commands ($ cd ~/.ssh) followed by ($ cd ls).
9. Now change the host file for Ansible so that we can SSH into our EC2s with Ansible and run the commands below. Use the command ($ sudo nano /etc/ansible/hosts)



```
  GNU nano 4.8                  /etc/ansible/hosts                  Modified
## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com
[Controller]
52.90.106.79 ansible_user=ubuntu ansible_ssh_private_key_file=~/.ssh/EC2Tutoria>
[Agent]
18.212.61.205 ansible_user=ubuntu ansible_ssh_private_key_file=~/.ssh/EC2Tutori>
[Production]
52.90.91.94 ansible_user=ubuntu ansible_ssh_private_key_file=~/.ssh/EC2Tutorial>



^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

**NOTE:**

i. **We can now use ansible to configure our EC2 instances. For the first EC2, we will need to install Java and Jenkins. Once we install Jenkins, we need to obtain the password to set up jenkins on our browser. We will be able to access our Jenkins application using the Public IPv4 of the Jenkins Controller EC2 followed by port 8080. The second EC2 instance needs Java, npm, and nodejs. The third EC2 needs openjdk and docker.**

ii. **We will need to create a set of YAML files with specific commands. There will be 3 ansible playbooks created to install dependencies in each EC2 instance. Once all the separated ansible playbooks are created, there will be one main ansible playbook that calls upon the other 3 and run them.**

10. In the first Ansible Playbook, create a YAML file called “configure_controller.yaml” with the following:

```
---
- name: "Configuring the Controller EC2 instance"
  hosts: Controller
  gather_facts: false
  connection: ssh

  tasks:
    - name: updating the ec2 instance
      shell: sudo apt-get update && sudo apt-get upgrade -y

    - name: installing java
      shell: sudo apt install openjdk-11-jre-headless -y

    # https://www.jenkins.io/doc/book/installing/linux/
    - name: getting the long term support release of jenkins
      shell: curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee \ /usr/share/keyrings/jenkins-keyring.asc > /dev/null

    - name: signing the downloaded jenkins application and adding it to the repository
      shell: echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \ https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null

    - name: upgrading the repository
      shell: sudo apt-get update && sudo apt-get upgrade -y

    - name: installing the jenkins application.
      shell: sudo apt-get install jenkins -y

    - name: installing git tool
      shell: sudo apt install git -y

    - name: checking if jenkins is active
      shell: sudo systemctl status jenkins | head -n 3
      register: command_output
    - debug:
        var: command_output.stdout_lines

    - name: outputting jenkins password
      shell: echo "The Jenkins password is $(sudo cat /var/lib/jenkins/secrets/initialAdminPassword)"
      register: command_output
    - debug:
        var: command_output.stdout_lines
```

11. In the second Ansible Playbook, create a YAML file called “configure_agent.yaml” with the following:

```
---
- name: "Configuring the Agent EC2 instance"
  hosts: Agent
  gather_facts: false
  connection: ssh

  tasks:
    - name: updating the ec2 instance
      shell: sudo apt-get update && sudo apt-get upgrade -y

    - name: installing java
      shell: sudo apt install openjdk-11-jre-headless -y

    - name: installing nodejs
      shell: sudo apt install nodejs -y

    - name: installing npm
      shell: sudo apt install npm -y
```

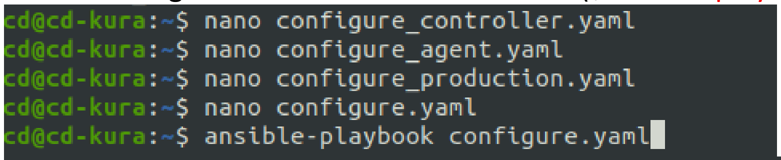
12. In the second Ansible Playbook, create a YAML file called “configure_production.yaml” with the following:

```
---
- name: "Configuring the Production EC2 instance"
  hosts: Production
  gather_facts: false
  connection: ssh

  tasks:
    - name: updating the ec2 instance
      shell: sudo apt-get update && sudo apt-get upgrade -y

    - name: installing java
      shell: sudo apt install openjdk-11-jre-headless -y

    - name: downloading modules
      shell: sudo apt-get install \ ca-certificates \ curl \ gnupg \ lsb-release -y

    - name: adding docker keys
      shell: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-
keyring.gpg

    - name: upgrading the repository
      shell: sudo apt-get update && sudo apt-get upgrade -y

    - name: installing latest version of docker
      shell: sudo apt-get install docker-ce docker-ce-cli containerd.io -y

    - name: starting docker
      shell: sudo systemctl start docker

    - name: configure docker to start on boot
      shell: sudo systemctl enable docker

    - name: checking if docker is active
      shell: sudo systemctl status docker | head -n 3
      register: command_output
    - debug:
        var: command_output.stdout_lines
```

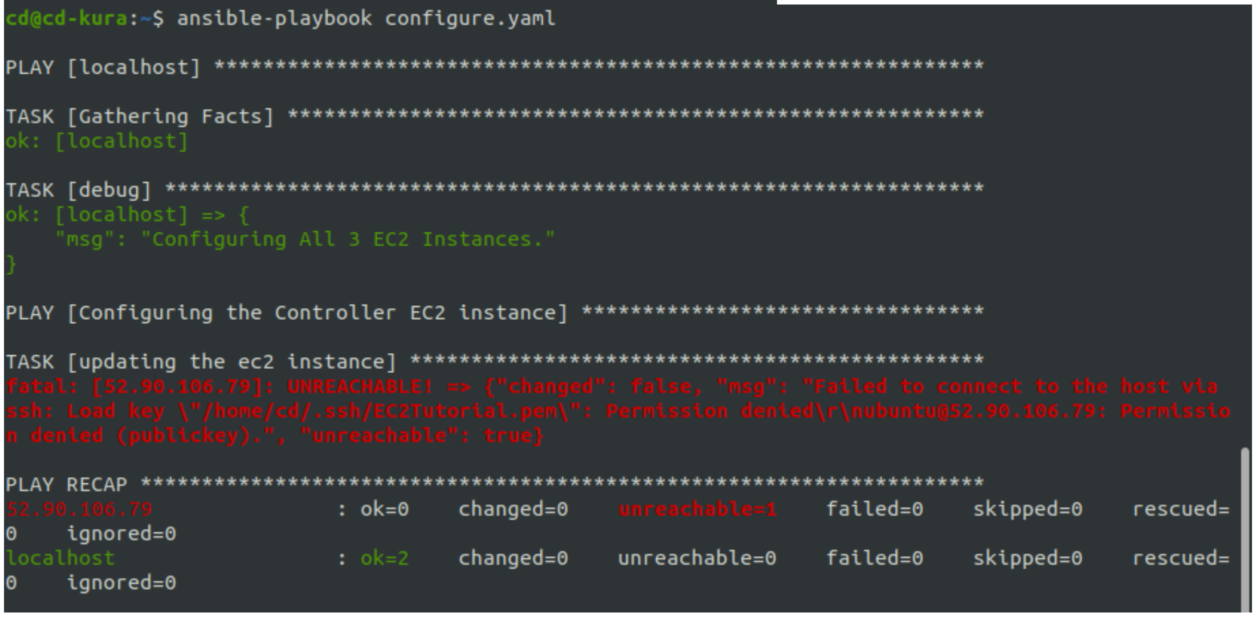
13. Then create the final Ansible playbook called “control.yaml” which will run all 3 of the previous yaml files. Use the following configuration:

```
---
- hosts: localhost
  tasks:
    - debug:
        msg: Configuring All 3 EC2 Instances.

- name: configuring the controller ec2 instance
  import_playbook: configure_controller.yaml

- name: configuring the agent ec2 instance
  import_playbook: configure_agent.yaml

- name: configuring the production ec2 instance
  import_playbook: configure_production.yaml
```

14. After creating all the files use the command ($ ansible-playbook configure.yaml).



```
cd@cd-kura:~$ nano configure_controller.yaml
cd@cd-kura:~$ nano configure_agent.yaml
cd@cd-kura:~$ nano configure_production.yaml
cd@cd-kura:~$ nano configure.yaml
cd@cd-kura:~$ ansible-playbook configure.yaml
```



```
cd@cd-kura:~$ ansible-playbook configure.yaml

PLAY [localhost] ***************************************************************

TASK [Gathering Facts] *********************************************************
ok: [localhost]

TASK [debug] *******************************************************************
ok: [localhost] => {
    "msg": "Configuring All 3 EC2 Instances."
}

PLAY [Configuring the Controller EC2 instance] *********************************

TASK [updating the ec2 instance] ***********************************************
fatal: [52.90.106.79]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via
ssh: Load key \"/home/cd/.ssh/EC2Tutorial.pem\": Permission denied\r\nubuntu@52.90.106.79: Permissio
n denied (publickey).", "unreachable": true}

PLAY RECAP *********************************************************************
52.90.106.79               : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=
0    ignored=0
localhost                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=
0    ignored=0
```

NOTE: Currently working on fixing this error.