Deployment #8 Documentation

First we create our first EC2 which will be acting as our Jenkins master

Configuration:

TCP custom 80 0.0.0.0/0

Http 8080 0.0.0.0/0

Next we create the second EC2 to act as the Jenkins Agent

Configuration:

TCP default security group only

We will be able to shh into this from the master

Task 1:

We will use a yaml file to create 3 EC2s via ansible

In our AWS account we are going to open cloud formation and create a stack (*with new resources)*

We are going to use nano to create a deployment 8.yaml file and paste the following inside

|  |
| --- |
| Resources: |
|  | Ec2OneSG: |
|  | Type: AWS::EC2::SecurityGroup |
|  | Properties: |
|  | GroupDescription: String |
|  | GroupName: "Ec2OneSG" |
|  | 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 |
|  | Ec2TwoSG: |
|  | Type: AWS::EC2::SecurityGroup |
|  | Properties: |
|  | GroupDescription: "Security group that allows SSH from anywhere" |
|  | GroupName: "Ec2TwoSG" |
|  | SecurityGroupIngress: |
|  | - IpProtocol: tcp |
|  | FromPort: 22 |
|  | ToPort: 22 |
|  | CidrIp: 0.0.0.0/0 |
|  | - IpProtocol: tcp |
|  | FromPort: 3000 |
|  | ToPort: 3000 |
|  | CidrIp: 0.0.0.0/0 |
|  | Ec2ThreeSG: |
|  | Type: AWS::EC2::SecurityGroup |
|  | Properties: |
|  | GroupDescription: "Security group that allows SSH from anywhere" |
|  | GroupName: "Ec2ThreeSG" |
|  | SecurityGroupIngress: |
|  | - IpProtocol: tcp |
|  | FromPort: 22 |
|  | ToPort: 22 |
|  | CidrIp: 0.0.0.0/0 |
|  | Ec2One: |
|  | Type: AWS::EC2::Instance |
|  | Properties: |
|  | ImageId: ami-09e67e426f25ce0d7 |
|  | InstanceType: t2.micro |
|  | KeyName: "your key" |
|  | SecurityGroupIds: |
|  | - !Ref Ec2OneSG |
|  | Tags: |
|  | - Key: "Name" |
|  | Value: "Ec2One" |
|  | Ec2Two: |
|  | Type: AWS::EC2::Instance |
|  | Properties: |
|  | ImageId: ami-09e67e426f25ce0d7 |
|  | InstanceType: t2.micro |
|  | KeyName: "your key" |
|  | SecurityGroupIds: |
|  | - !Ref Ec2TwoSG |
|  | Tags: |
|  | - Key: "Name" |
|  | Value: "Ec2Two" |
|  | Ec2Three: |
|  | Type: AWS::EC2::Instance |
|  | Properties: |
|  | ImageId: ami-09e67e426f25ce0d7 |
|  | InstanceType: t2.micro |
|  | KeyName: "your key" |
|  | SecurityGroupIds: |
|  | - !Ref Ec2ThreeSG |
|  | Tags: |
|  | - Key: "Name" |
|  | Value: "Ec2Three" |

Next we are going to use the terminal to cd into the .ssh folder

cd /etc/ansible

sudo ansible-playbook key\_copy.yaml

Now we will use ansible to install Jenkins and Docker, for some reason when trying to run the ansible playbook it will fail on the production ec2 due to docker engine not being installed

To get around this we should ssh into our third EC2 and manually install docker engine.

Run the following command once inside the third EC2

1. **Update the apt package index and install packages to allow apt to use a repository over HTTPS:**

*sudo apt-get update*

*sudo apt-get install \*

*ca-certificates \*

*curl \*

*gnupg \*

*lsb-release*

1. **Add Docker’s Official GPG Key**

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

1. Use the following command to set up the **stable** repository. To add the **nightly** or **test** repository, add the word nightly or test (or both) after the word stable in the commands below.

*echo \ "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \*

*$(lsb\_release -cs) stable test" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null*

1. **Update the apt package index, and install the *latest version* of Docker Engine and containerd**

*sudo apt-get update*

*sudo apt-get install docker-ce docker-ce-cli containerd.io*

you need to run apt-get update before you can install any other packages. After running the updates you can then apt-get install -y ca-certificates and this is the package that contains the command update-ca-certificates

you need to run apt-get update before you can install any other packages. After running the updates you can then apt-get install -y ca-certificates and this is the package that contains the command update-ca-certificates

Now exit this EC2 and back on your local machine sudo into the playbook.yaml file and erase the last section that says install modules and well as install docker engine because we just did that manually.