Jenkins Setup Using AWS Autoscaling Group, Load Balancer & EFS

Version History

#	Date	Version	Author	Reviewer
1	11/5/2023	Initial Version	P. Umapoornima	

Overview

The aim of this POC is to demonstrate how to deploy a reliable and scalable Jenkins environment on AWS. The POC aims to provide a solution that can handle a large amount of traffic and requests, ensuring high availability, scalability, and fault tolerance. The POC provides a cost-effective solution to deploy Jenkins on AWS by automatically scaling the nodes based on the load and traffic. This approach can be useful for organizations that require a reliable and scalable CI/CD solution for their software development and deployment process on AWS.

Pre-requisites

- ubuntu 22.04 server
- Git
- Ansible
- Terraform
- Hashi Corp packer
- AWS CLI, default VPC, us-west-2 with three subnets (2a,2b, and 2c)
- Access key & Secret Key with Aws account (us-west-2 region)
- Key Pair with Aws account Under (us-west-2 region)

Tools & Services

DevOps Tools	AWS Services
Git	IAM
Ansible	EFS
Terraform	Aws Parameter Store
Hashi Corp Packer	Auto Scaling Group
	Application Load Balancer

Procedure

- To Perform this POC I used Aws console. In aws console account first we have to login our console and we must change our region **{Oregon Us-west-2}**
- We have to launch one Ec2 Linux or Ubuntu server I'm taking Ubuntu Server 22.04 and connect to that instance
- We must install what are all the packages we required to start that poc we have to install Git, ansible, Terraform, Packer & Aws CLI
- First, we must create one access key & secret key to configure our aws account to create that one we have to go to Security Credentials under security credentials we have to create one access key & secret key & download and save that key
- Configure your account details in your server by using **"Aws configure"** command provide your access key & Secret key, region
- These are the following steps to execute this poc

Step1: Add SSH key Pair to AWS Parameter Store

- First, we are going to create Aws parameter store in your Aws console. We will use SSH authentication for both the Jenkins controller and agent, storing the SSH keys securely in the AWS Parameter Store
- First, you need to create a ssh key pair using the following command to generate the keys, run **"ssh-keygen"** command in your server. It will create a private (id_rsa) and public key (id_rsa.pub). To get that private & public keys we have to navigate "**cd .ssh"** folder
- Run **"ls"** command keys will be displayed (id_rsa) & (id_rsa.pub) to copy that keys we have to run following commands **"cat id_rsa", "cat id_rsa.pub"**

- Go to parameter store service in your aws console, now you need to create two parameters of type **"Secure String"** in the following paths
  /devops-tools/jenkins/id_rsa
  /devops-tools/jenkins/id_rsa.pub
- Parameter Store will be created for both private and public key

Step2: Create Jenkins IAM Role Using Terraform

- Clone the Git Repository "git clone https://github.com/techiescamp/devops-projects" & run "**ls"** command and navigate to the **"cd devops-projects"** folder All the automation script for this project is present it 01-jenkins-setup folder **"cd 01-jenkins-setup"** in that navigate **"cd /terraform/iam"**.
- Using Terraform create IAM Role and Policy for accessing the Aws instance
- Initialize terraform and apply the configuration using the following commands
  terraform init
  terraform plan
  terraform apply --auto-approve
- It creates a role named jenkins-role. You can verify the IAM policy and role from the AWS console.

Step3: Create EFS Using Terraform

- Using Terraform, Create EFS storage for persisting the Jenkins data. This will help maintain the state of Jenkins controller. EFS storage will be created in all 3 availability zones under the region.
- Navigate to the "cd **terraform/efs**" folder
- You will find a **main.tf** file go to that file **Vim main.tf**, with the following content, Replace the **VPC** and **subnet IDs** highlighted in bold with your **VPC** and **subnet IDs from the us-west-2 region** (Oregon) after changing the changes save the file (:wq)

```
provider "aws" {
 region = "us-west-2"
}
module "efs_module" {
 source = "../modules/efs"
 vpc_id     = "vpc-0a5ca4a92c2e10163"
 subnet_ids = ["subnet-058a7514ba8adbb07", "subnet-0dbcd1ac168414927", "subnet-032f5077729435858"]
}
```

- To provision the EFS storage run following commands
  terraform init
  terraform plan
  terraform apply --auto-approve
- In the next step, when building the Jenkins controller image, we will use the DNS endpoint of EFS and Go to the EFS in your console & copy the DNS name of EF **"fs0e3fc4345eeb399fa.efs.us-west-2.amazonaws.com"**

Step4: Build Jenkins Controller AMI

- Now we can build the Jenkins controller AMI using Packer and Ansible.

- Ensure your current directory is **01-jenkins-setup** in that directory we have jenkins-controller.pkr.hcl go to that file
- Vim jenkins-controller.pkr.hcl in this file change **"AMI ID"** add AMI ID of ubuntu 22.04 server & in that default section add **"DNS Name of EFS"** & save the file

```
variable "ami_id" {
 type    = string
 default = "ami-0735c191cf914754d"
}
variable "efs_mount_point" {
 type    = string
 default = ""
}
```

- After changing the changes run the command **"packer build jenkins-controller.pkr.hcl"** to build Jenkins Controller AMI
- Upon successful execution, you will see the registered Jenkins-controller AMI ID in your Aws Console under AMIs.

Step5: Build Jenkins Agent AMI

- In the Jenkins Agent Ansible role, we have tasks to install tools like Terraform, Ansible, boto3, Java, etc.
- To Create this Jenkins Agent AMI, go to Jenkins-agent.pkr.hcl file "**vim jenkins-agent.pkr.hcl"** and change the **AMI ID** & save the file, which AMI Id you used in the Jenkins controller same AMI Id we have to use for Jenkins Agent also
- To Build Jenkins Agent AMI run the command **"packer build Jenkins-agent.pkr.hcl"**
- Upon successful execution, you will see the registered Jenkins-Agent AMI ID in your aws console under AMIs.

Step6: Deploy Jenkins Autoscaling Group with Load Balancer

- To deploy the autoscaling group with the load balancer, cd into the **terraform/lb-asg** folder
- You will find a **main.tf** file in that main.tf file, replace the following values
  - Replace **subnet IDs** with your subnet IDs
  - Replace **techiescamp** with your ssh key pair name.
  - Replace AMI Id **ami-011269e155315bf83** with the AMI id of your Jenkins controller AMI Id
  - Replace **vpc-0a5ca4a92c2e10163** VPC Id with your VPC Id & save the file
- Execute the following commands to deploy autoscaling group with load balancer

```
terraform init
terraform apply –auto-approve
```

- After the Terraform, execution is complete, you can verify the auto-scaling group and load balancer from the AWS console & under target group it should bring up the one instance of Jenkins controller & healthy it should become 1 after that check One new Jenkins controller instance was up and running in your console.

- If you open the **jenkins-alb** load balancer, you can get the DNS name of the load balancer & copy the DNS name of load balancer and hit in the browser you should see the Unlock Jenkins page will open
- To unlock the Jenkins, connect to the Jenkins controller instance and run the command " **sudo cat /data/jenkins/secrets/initialAdminPassword**" one key will be generated copy the key and paste in the Jenkins dashboard
- Jenkins dashboard is ready & create one sample job
- Terminate the controller AWS instance and note that a new controller instance is scaled up automatically due to the auto scaling group

Step7: Provision Jenkins Agent

- To provision Jenkins agent, **cd into terraform/agent** folder & in the main.tf file, replace the following values
    - Replace **subnet IDs** with your subnet IDs
    - Replace **techiescamp** with your ssh key pair name.
    - Replace AMI Id **ami-047fe714e6e0ac977** with the AMI id of your Jenkins agent AMI Id
    - If you want more than one Jenkins agent, you can replace the **instance_count** number with the required number of agents.
- Execute the following commands to provision the agent node

```
terraform init
terraform plan
terraform apply --auto-approve
```

References

- Poc Link: https://devopscube.com/jenkins-autoscaling-setup/
- Git Installation: https://www.digitalocean.com/community/tutorials/how-to-install-git-on-ubuntu-22-04
- Ansible Installation: https://linux.how2shout.com/install-and-configure-ansible-on-ubuntu-22-04-linux/
- Packer Installation: https://developer.hashicorp.com/packer/downloads?product_intent=packer
- Terraform Installation: https://developer.hashicorp.com/terraform/cli/install/apt

Errors Occurred

- **Step2:**

  Error: creating IAM Instance Profile (jenkins-instance-profile): EntityAlreadyExists: Instance Profile jenkins-instance-profile already exists

  Reason: This error is occurring because jenkins role& policy was already exists in the same region

  Solution: we have to remove the existed Jenkins role, policy and Jenkins-Instance-profile by using following commands

  Run "terraform destroy" in your terminal under this path "01-jenkins-setup/terraform/iam " & to remove jenkins instance profile use following command

“ aws --region=us-east-1 iam delete-instance-profile --instance-profile-name ocp4-925gm-master-profile ”

Refer Link: https://stackoverflow.com/questions/62621268/aws-install-fails-iam-instance-profile-already-exists

- **Step4:**

  Error: TASK [jenkins-controller: Install Java JDK 17] ********** amazon-ebs.jenkins: fatal: [default]: FAILED! => {"cache_update_time": 1683719597, "cache_updated": false, "changed": false, "msg": "'/usr/bin/apt-get -y -o \"Dpkg::Options:=--force-confdef\" -o \"Dpkg::Options::=--force-confold\"

  Reason: installation of Java JDK 17 failed because there are some broken dependencies while installing the Java JDK 17.

  Solution: Change the Script openjdk-17-jdk to openjdk-17-jre under 01-jenkins-setup/ansible/roles - for both Jenkins Controller and Jenkins Agent **(OR)** Try to Try installing Java JDK 17 again by running those commands

  sudo apt-get update && sudo apt-get install -f

  sudo apt-get install openjdk-17-jdk

- **Step5:**

  Error: amazon-ebs.jenkins: ERROR! We were unable to read either as JSON or YAML, these are the errors we got from each:

  amazon-ebs.jenkins: JSON: Expecting value: line 1 column 1 (char 0)

  Reason: syntax problem

  Solution: check the syntax and update the script under devops-projects/01-jenkins-setup/ansible/roles/jenkins-agent/tasks/java.yaml': line 3, column 1