Mike Major

10/29/2022

Deployment 4

Deployment 4

Install and Configure Jenkins Server on AWS EC2

* Use Search to find “EC2” and then click on EC2
* Click “Launch Instance” -- Example of setup below with image Fig.1
  + Enter name of instance
  + Select AMI image
  + Select Instance Type -- recommended to use t2.mcro for Free Tier usage
  + Select key pair
  + Network setting → Select default VPC → Subnet → Auto Assign IP (Enable)
  + Use a security group with the following configurations
    - SSH port 22 / TCP port 8080 (used for Jenkins)
* Click on Advanced Details
  + To automate your installation, it is recommended you use a script to install Jenkins and python venv. See Fig. 2
* Then, Click 
* After the instance is launched, confirm Jenkins is running by using the command, “systemctl status jenkins”. This should return a “running” message that displays on your screen.

Fig. 1

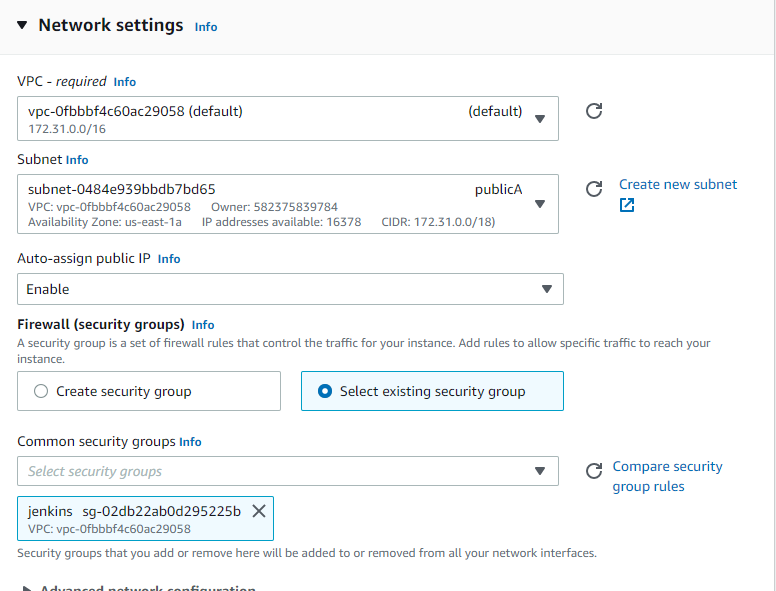
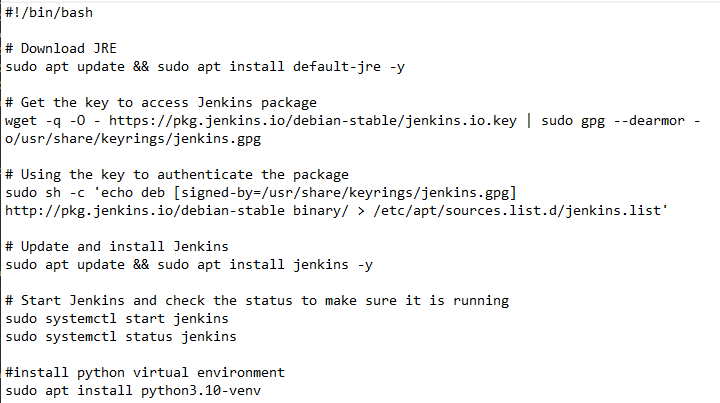


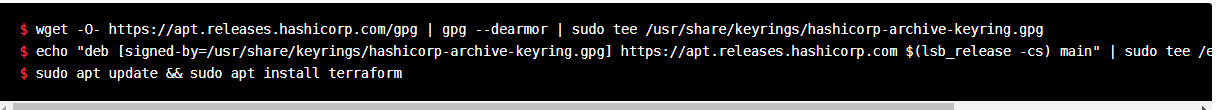
Fig. 2



* Use your public facing ip address alongside port 8080 to navigate to the Jenkins server website.
  + This website will allow you to administer your Jenkins build
* Select the provided path key to locate your secret passcode to allow entry into Jenkins
* Follow steps to complete the initial setup of Jenkins.

Terraform Installation

* Terraform must be installed on the initial EC2 that was created above. Terraform will allow you to create resources within your AWS environment for this deployment.
* Navigate to <https://www.terraform.io/downloads> and select “Ubuntu/Debian” for the commands to install Terraform. The screen should look like this.



Configure Python Virtual Environment

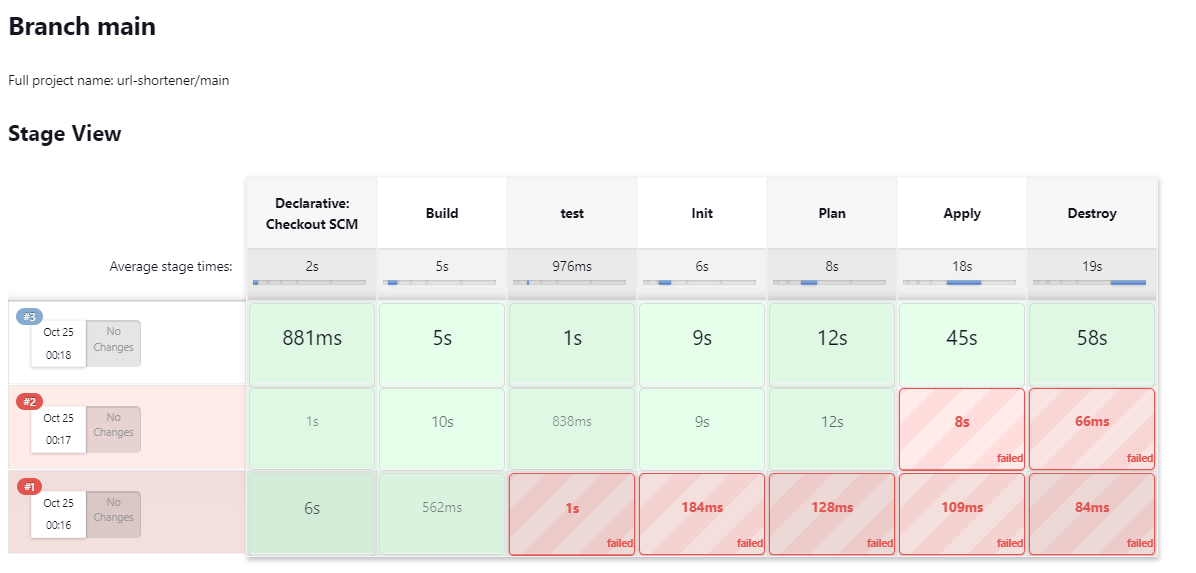
* Use the following commands to install pip and python virtual environments
* Sudo apt install python3.10-venv / sudo apt install python3-pip

Configure AWS Credentials within Jenkins

* This step will allow you to use your programmatic access user to pass authentication into AWS for the purposes of creating resources.
* Within the Jenkins Dashboard → Manage Credentials
* Select Global
* Click Add Credentials
  + Kind: Secret Text
  + Scope: Global
  + Secret: Enter your secret key
  + ID: AWS\_SECRET\_KEY
  + Click Create to finalize adding your secret key
  + Repeat the steps for your Access Key, instead use AWS\_ACCESS\_KEY for ID

Build Pipeline in Jenkins

* This step will create the Jenkins pipeline to begin Building and Testing your code
* Click New Item
* Select Multi Branch Pipeline
* Enter Display Name: Flask App or whichever name you’d prefer
* Enter Description: CI/CD pipeline
* Add Branch Source: Github, then click Add → Jenkins
* Username: Enter your Github username. Example: tman12
* Password: Enter your Github access key token. If not available, please create for future use.
* Click Save
* Enter the repository url for Github
* Click Validate
* Click Apply and Save
* The build process should begin immediately. Monitor to ensure the steps complete successfully with no issues.
* If issues arise, make sure to click the status boxes and review the necessary logs to troubleshoot.
* A successful build should appear like this on your screen.



Add Webhook to Trigger Build

* Go to your Github repository and click on Settings
* Click on webhooks and then click Add webhook
* In the payload url section, enter your jenkins environment url.
* Then select the events you’d like for the webhook to be triggered by. I would recommend using “pull requests” and “pushes”.
* Next, connect with your Jenkins web interface
* Select New Item,
* Select Freestyle Project → Webhook for Deployment 4
* Click Source Code Management
* Enter your Github repository url
* Under Build Triggers → Github Hook Triggers
* Click save

Modify Jenkins File by Adding “Destroy” to Terraform

* It is vitally important to shut down any and all billable resources when not in use in your environment.
* Utilizing “terraform destroy” will ensure that all resources that were created in your Terraform environment will be eliminated.
* Start by going to your Jenkinsfile in your Github build and modifying the Jenkins file.
* Pay special attention to placement of your “destroy” stage and the corresponding bracket and parentheses.

Issues Detected During Deployment

* During the initial build stages in Jenkins, when Jenkins attempted to Test, then initialize Terraform there appeared to be significant lag or freezing with the EC2 instance. This caused multiple failures during the build.
  + Recommended action -- If you suspect network connectivity, log into AWS and restart your instance. After restarting your instance, make sure Jenkins is still running and attempt to create the build again.
* When modifying your Jenkins file, make sure to pay special attention to brackets and parentheses placement. Missing even one can affect your build stages from completion.