**Deployment #1,2,3**

Name: Biki Gurung

|  |  |
| --- | --- |
| **Version** | **Date** |
| **V 1.0** | **09/05/2022** |
| **V 2.0** | **09/22/2022** |
| **V 3.0** | **10/08/2022** |

**Description:**

* **Create a VPC with Public and Private Subnet**
* **Deploy your Flask app into your VPC using Jenkins Agent**
* **Use NGINX and gUnicorn to Deploy your application**
* **Use Jenkins for CI / CD Pipeline - Build, Test and Deploy application**
* **Use Elastic Beanstalk to deploy application**
* **Use AWS CloudWatch to Monitor**
* **Use Cypress and JUnit Test**

**Pre-requisites:**

* AWS account
* CI tool of choice (Jenkins)
* GitHub repository you’d like to deploy

Tables of Contents:

[Install Jenkins on an EC2: 3](#_Toc116389487)

[Install Virtual Environment 5](#_Toc116389488)

[Activate the Jenkins user on the EC2 6](#_Toc116389489)

[Create a Jenkins user in your AWS account 7](#_Toc116389490)

[Install AWS CLI on the Jenkins EC2 and configure 9](#_Toc116389491)

[Configure AWS CLI for Jenkins (Sub User) 9](#_Toc116389492)

[Install EB CLI in the Jenkins EC2 user 9](#_Toc116389493)

[Connect GitHub to Jenkins Server 10](#_Toc116389494)

[Create an access token from GitHub 11](#_Toc116389495)

[How to setup Jenkins: 12](#_Toc116389496)

[Create a multibranch build 14](#_Toc116389497)

[Configure and connect a Jenkins agent to Jenkins: 19](#_Toc116389498)

[How to Deploy application in Elastic Beanstalk 23](#_Toc116389499)

[Deploy application using Elastic Beanstalk CLI and Jenkins: 23](#_Toc116389500)

[How to Manually Deploy to Elastic Beanstalk 25](#_Toc116389501)

[How to create an Elastic Beanstalk Env 27](#_Toc116389502)

[Continuous Deployment (CD) using Jenkins and Elastic Beanstalk 29](#_Toc116389503)

[How to Setup NGINX with GUNICORN: 31](#_Toc116389504)

[How to start GUnicorn with Jenkins: 32](#_Toc116389505)

[List of Issues and their solution 32](#_Toc116389506)

[Modifications & Improvements to Pipeline 38](#_Toc116389507)

[Add a way to Monitor and Notify you 38](#_Toc116389508)

[Add CloudWatch Agent to Monitor your Jenkins Server 38](#_Toc116389509)

[Use Cypress for testing 42](#_Toc116389510)

[Add a Linter 42](#_Toc116389511)

[Change something on the application front 43](#_Toc116389512)

[What Improvements can be made? 45](#_Toc116389513)

[What stack has been implemented? 46](#_Toc116389514)

[Explaining the Stack, we are using: LEJP 46](#_Toc116389515)

[How is the application running – Overview? 46](#_Toc116389516)

[CI/CD Pipeline Diagram 3: 49](#_Toc116389517)

|  |  |
| --- | --- |
| Install Jenkins on an EC2: | |
|  | 1. Create EC2 Ubuntu Instance <https://www.cloudbooklet.com/create-an-ec2-instance-on-aws-with-ubuntu-18-04/>      1. Download the .pem file found when creating the EC2 Instance cand store on your local P.      1. SSH into the EC2 using the .pem file and the Public IP Address of the Ec2 instance.   ssh -i "aaa.pem" [ubuntu@ec2------](mailto:ubuntu@ec2-52-90-159-255.compute-1.amazonaws.com) | |
|  | Opening Ports 80,8080 and 22 on the ec2 instance    <https://aws.amazon.com/premiumsupport/knowledge-center/connect-http-https-ec2/> | |
|  | **Login to EC2 Instance:**  Once you’ve created the EC2, log into the EC2 and then  enter the commands below to install Jenkins:    $ ssh to your EC2 | |
|  | $ sudo apt update && sudo apt install default-jre | |
|  | $wget -q -O - <https://pkg.jenkins.io/debian-stable/jenkins.io.key> |sudo gpg --dearmor -o/usr/share/keyrings/jenkins.gpg    $sudo sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg] [http://pkg.jenkins.io/debian-stable binary/](http://pkg.jenkins.io/debian-stable%20binary/) > /etc/apt/sources.list.d/jenkins.list'    $sudo apt update && sudo apt install jenkins -y    $sudo systemctl start jenkins      For assistance:   * [https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkin s-on-AWS/](https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkin%20s-on-AWS/)      * <https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04> | |
|  | To check the current status of Jenkins:    $sudo systemctl status jenkins | |
| Install Virtual Environment | | |
|  | Checking if python3 is intalled    $ python3 --version    Install updates:  $ sudo apt install    Install Python3-pip    Install Python pip $ sudo apt install python3-pip    Upgrade to latest version of pip  $ sudo pip3 install --upgrade pip            For Assistance:   * <https://phoenixnap.com/kb/how-to-install-pip-on-ubuntu> * <https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/> | |
|  | Install python3-10-venv    $sudoapt-get-y installpython3.10-venv    Or    $sudoapt-y installpython3.10-venv        Assistance:   * <https://installati.one/ubuntu/22.04/python3.10-venv/> | |
| Activate the Jenkins user on the EC2 | | |
|  | $sudo passwd jenkins  $sudo su - jenkins -s /bin/bash | |
| Create a Jenkins user in your AWS account | | |
| page2image37831696 | * Navigate to IAM in the AWS console. * Next click on the Users option in the Access management. | |
| page3image38029968 | * Select add user. * Next, the username can be EB-user. * Select Programmatic access and then next. | |
| page3image38029552 | * Select “Attach existing policies directly” and select administrator access. Now select Next for this page and the next page. | |
| Graphical user interface, text, application, email  Description automatically generated | * Finally, create the user | |
| Graphical user interface, application, Teams  Description automatically generated | * Copy and save the “access key ID” and the “secret access key” * Click on Close | |
| Install AWS CLI on the Jenkins EC2 and configure | | |
|  | First, ssh into your Jenkins EC2  $ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"  To download AWS cli package to install on your EC2  sudo apt install unzip  unzip awscliv2.zip  sudo ./aws/install | |
|  | To Make sure the AWS Cli installed properly,  $aws --version | |
| Configure AWS CLI for Jenkins (Sub User) | | |
|  | $ sudo su - jenkins -s /bin/bash  if user Jenkins is unable to sudo add user Jenkins to the sudoer:  $usermod -aG sudo jenkins | |
|  | To configure aws CLI with your user account you will need the Access Key and the Secret we created earlier.  $ aws configure  - Set Access Key ID  - Set Secret Access Key  - Set region to : us-east-1  - Set Output format: json | |
| Install EB CLI in the Jenkins EC2 user | | |
|  | To install elastic Beanstalk(eb) CLI:  $ pip install awsebcli --upgrade –user | |
|  | To check the eb Version  $eb –-version  If eb is not recognized by the system, the Path needs to be added so that the shell recognizes the application eb.  So in order to do so:  $nano /etc/bash.bashrc    export PATH=$PATH:/var/lib/jenkins/.local/bin  <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-install-advanced.html> | |
| Connect GitHub to Jenkins Server | | |
|  | First Fork the Deployment repo: <https://github.com/kura-labs-org/kuralabs_deployment_1>. git          mkdir deployment1  cd deployment1  git init    ssh-keygen -t ed25519 -C "bikigurung8@gmail.com"    ls ~/.ssh    cat id\_ed25519.pub | |
|  | Cat and copy the contents from the id\_ed25519.pub  Go to Github Settings > Add new key | |
|  | Fork the repo to your repository  And then git pull the repo from your github    $ git pull git@github.com:bikigrg11/kuralabs\_deployment\_1.git | |
| Create an access token from GitHub | | |
|  | Navigate to your GitHub settings, select developer settings    Select the settings you see below for access token permissions. | |
|  | Select personal access token and create a new token.    Select the settings you see below for access token permissions.    New token has been generated. | |
| How to setup Jenkins: | | |
|  | After you verified that the Jenkins has been started on your AWS EC2 Machine. | |
|  | Connect to http://<your\_server\_public\_DNS>:8080 from your browser. You will be able to access Jenkins through its management interface:  As prompted, enter the password found in /var/lib/jenkins/secrets/initialAdminPassword.  Use the following command to display this password:  $ sudo cat /var/lib/jenkins/secrets/initialAdminPassword | |
|  | Let the environment creation be finished | |
|  | Enter Username  Enter Password  Enter Confirm Password  Enter Full Name  Enter Email Password | |
|  | You can use this Jenkins URL to login and view your Jenkins application. | |
| Create a multibranch build | | |
|  | ● Log back into Jenkins and select “New item” | |
|  | ● Select multibranch pipeline  ● Enter a display name and brief description | |
|  | ● Add a Branch source by selecting Add source and select GitHub | |
|  | ● Select the Add button and select GitHub | |
|  | ● Click on Add and then select Jenkins | |
|  | Under username enter your GitHub username  ● Under password enter your token  ● (Optional) under ID and Description enter GitHub repo | |
|  | ● Enter your URL to the repository and you can validate by selecting validate. | |
|  | Make sure this says Jenkinsfile | |
|  | ● Select Apply and then Save | |
|  | Build Flask > Status > click On Main | |
|  | You should see a build happening. If you don’t, select Scan Repository.  The build has been completed which will run on your EC2 by pulling the Jenkinsfile from the github repo.  If you see green check box next to the Build No in this case #1 this means the build was completed successfully.  We can see that there was 2 Stages that were created.   * 1. Build   2. Test   More info: If you hover through the time (11s for the Build Stage) on how long each stage took to build you can see the logs and the results of the tests that took place in detail. | |
| Configure and connect a Jenkins agent to Jenkins: | | |
| page2image57780176 | Enter your Jenkins server  Select the Build Executor Status | |
| page3image57721040 | - Next Select “+ New Node” to configure and add the agent. | |
| page4image57690768 | - Enter the node name “awsDeploy” and select “Permanent Agent” and then create. | |
| page6image41184752 | Now enter the configurations below:  ○  Name: awsDeploy  ○  Description: Deployment server  ○  Number of executors: 1  ○  Remote root directory: /home/ubuntu/agent  ○  Labels: aweDeploy  ○  Usage: only build jobs with label....  ○  Launch method: launch agents via ssh  ○  Host: {Enter the public IP of your EC2 in the Public subnet and not this text}  ○  Credentials: see below  ○  Host key verification strategy: non verifying verification strategy  Availability: keep this agent online as much as possible | |
| page7image57759424 | Credential steps:  ○  Select “Add” => “Jenkins”=>Kind:”SSH username  with private key”  ○  Enter the ID, Description, username  ○  To add the key, select “Enter Directly” => select  “add” => paste the private key into the white box and save. | |
| page9image57750528 | Save the configurations and wait for Jenkins to connect to the agent. It should look like what you see below: | |
| page11image57672304 | | |
| How to Deploy application in Elastic Beanstalk | | |
| Deploy application using Elastic Beanstalk CLI and Jenkins: | | |
|  | Switch to System User Jenkins  $sudo su - jenkins -s /bin/bash | |
|  | Go to our workspace created by Jenkins which is called url\_shortener\_main  $cd /var/lib/jenkins/workspace/url-shortener\_main | |
|  | Initialize Elastic Beanstalk:  $ eb init  Select: us-east-1  ● Press enter  ● Select: Python  ● Select: (The latest version of python available)  ● Select: N (for CodeCommit) | |
|  | $ eb create  Take the default for the next 3 questions by hitting enter (remember the environment name)  Spot Fleet: No | |
|  | Wait for the environment to be made!! And then check it | |
|  | If you go AWS Console you can also see Elastic Beanstalk env being created. | |
|  | When the Elastic Beanstalk has been created you can see: | |
| How to Manually Deploy to Elastic Beanstalk | | |
|  | Use git clone to copy deployment1 repo files of the flask application to your local computer.    $ git clone git@github.com:bikigrg11/kuralabs\_deployment\_1.git | |
|  | When you use the AWS Elastic Beanstalk console to deploy a new application or an application version, you'll need to upload a source bundle. Your source bundle must   * Consist of a single ZIP file or WAR file (you can include multiple WAR files inside your ZIP file) * Not exceed 512 MB * Not include a parent folder or top-level directory (subdirectories are fine)     Documentation:  To compress the files from the repo, follow the how to from aws documentation: [https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/ applications-sourcebundle.html](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/%20applications-sourcebundle.html) | |
|  | In order to zip the file first install zip  $ sudo apt install zip  $ zip kuralabs\_deployment\_1.zip kuralabs\_deployment\_1 | |
| How to create an Elastic Beanstalk Env | | |
|  | After you have compressed/zipped your files, head over to AWS Elastic Beanstalk     * Go to AWS Elastic Beanstalk      * Select the Web server Environment | |
|  | Next type in the following information  Application name: url-shortner  ○ Environment name: Urlshortner-env  ○ Platform: Python  ○ Platform branch: 3.8  ○ Platform version: 3.3.16  ○ Application code: Upload your code, local file      Resource:  <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/GettingStarted.CreateApp.html> | |
|  | Click on locale file    Click on Choose file    Select the zip file downloaded from the server | |
| A picture containing graphical user interface  Description automatically generated | Click to Create Environment | |
|  | Wait for few minutes until the Environment is created and started. | |
| Continuous Deployment (CD) using Jenkins and Elastic Beanstalk | | |
| pipeline {  agent any  stages {  stage ('Build') {  steps {  sh '''#!/bin/bash  python3 -m venv test3  source test3/bin/activate  pip install pip --upgrade  pip install -r requirements.txt  export FLASK\_APP=application  flask run &  '''  } }  stage ('test') {  steps {  sh '''#!/bin/bash  source test3/bin/activate  py.test --verbose --junit-xml test-reports/results.xml  '''  }  post{  always {  junit 'test-reports/results.xml'  }  } }  stage ('Deploy') {  steps {  sh '/var/lib/jenkins/.local/bin/eb deploy url-shortener-dev '  } }  } } | Update the Jenkins file and add the Stage: Deploy  $ sh '/var/lib/jenkins/.local/bin/eb deploy url-shortener-dev '  Will use the eb cli to create an artifact and then deploy to eb that has been created earlier name “url-shortener-dev” | |
| Calendar  Description automatically generated | After the Jenkinsfile has been updated and GitHub has been updated.  Build Jenkins again.  If all the stages have passed OK then go to AWS Console and elastic beanstalk | |
| Graphical user interface, text, application, chat or text message  Description automatically generated  Graphical user interface, text, application, website  Description automatically generated | EB Env Health is OK  If you click on the link below you can go to the application page. | |
| Graphical user interface, application, website  Description automatically generated | | |
| How to Setup NGINX with GUNICORN: | | |
| Before you build your pipeline, SSH into the EC2 in your VPC and then nano into the “/etc/nginx/sites-enabled/default” file.  ● First change the port from 80 to 5000:  server {  listen 5000;  ● Scroll down to where you see “location” and replace it with the text below:  Location / {  proxy\_pass http://127.0.0.1:8000;  proxy\_set\_header Host $host;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  }  Additional Resources:  <https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-gunicorn-and-nginx-on-ubuntu-18-04>  <https://realpython.com/django-nginx-gunicorn/>  Serve Python App on Nginx. Hello Readers, In this blog, I will… | by Krupa  Bhimani | Dev Genius | | |
| How to start GUnicorn with Jenkins: | | |
| Add Deploy stage to your JenkinsFile  stage ('Deploy') {  agent{label 'awsDeploy'}  steps {  sh '''#!/bin/bash  git clone https://github.com/kura-labs-org/kuralabs\_deployment\_2.git  cd ./kuralabs\_deployment\_2  python3 -m venv test3  source test3/bin/activate  pip install -r requirements.txt  pip install gunicorn  gunicorn -w 4 application:app -b 0.0.0.0 --daemon  '''  } } | | |

|  |  |
| --- | --- |
| List of Issues and their solution | |
| Issue #1 – Couldn’t download zip file using SCP from the Server | |
|  | When trying to download the zip file from the Ec2 server, I kept on getting permission denied error message. Although I was using the correct .pem file and using scp -I “.pem file” I was unable to download from the server.    so I downloaded and used FileZilla on my local machine to connect to the server    <https://dearsikandarkhan.medium.com/files-copying-between-aws-ec2-and-local-d07ed205eefa>  <https://filezilla-project.org/download.php?type=client> |
|  | Go to Settings > General    Enter the Host Machine Information    Provide the path to the .pem key file    And connect |
|  | If you are connected successfully then you can see the files from the remote server |
|  | Select the folder on your local machine where you want the files to be downloaded |
|  | On your Remote machine, select the files and right click on it  And click on Download    When the file is downloaded you can see the status. |
| Issue #2: Elastic beanstalk kept on failing with HTTP 502 | |
|  | After creating the Env for the Elastic Beanstalk the Health of my Elastic Beanstalk kept on getting Severe. |
|  | I downloaded the log files and noticed that the artifact (Zip file) that was uploaded was not loading properly.  Because I was using MacBook to zip and upload to the elastic beanstalk env it was giving me this error message. |
| Files selected in Mac OS X Finder | So, I followed what was written here:  Zipping files in Mac OS X Finder or Windows explorer:  When you create a ZIP file in Mac OS X Finder or Windows Explorer, make sure you zip the files and subfolders themselves, rather than zipping the parent folder. |
| Compressing files in Mac OS X Finder  Icon  Description automatically generated | Once I followed this step and uploaded the new zip file to the elastic beanstalk the Health Turned to “OK” |
| A picture containing timeline  Description automatically generated | If you click on the link you will be able to open the application that has started on the Elastic Beanstalk env. |
|  | |
| Issue #3, Do not create new Jenkins User | |
| When installing Jenkins, a Jenkins user is already created for you. So if you sudo passwd jenkins, you should see a jenkins systems user already being created.  For More info check: System Users Vs Sub User  Would say a major difference between rep:User and rep:SystemUser is that one doesn't explicitly have a password set, nor is one really needed... The system user is configured in such a way that it can only be used by sub-services that have been mapped with the org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl OSGi Config Factory. This allows you to use a resource resolver that isn't an individual end user/author, nor is it the full-access admin resolver.  Resource: <https://experienceleaguecommunities.adobe.com/t5/adobe-experience-manager/main-differences-between-the-system-user-and-the-normal-user/td-p/212618> | |
| Issue #4 gUnicorn Daemon not starting using Jenkins | |
| $ gunicorn --workers=4 application:app --bind=0.0.0.0 --daemon  Gunicron not starting with daemon using Jenkins inorder to start the application. I had to ssh into the Jenkins Agent server and then start the gunicorn –daemon to run the application. | |

|  |  |  |
| --- | --- | --- |
| Modifications & Improvements to Pipeline | | |
| Add another test to Pytest unit testing | | |
| def test\_urls\_json():  url = {'url': 'http://google.com'}  bool = checkURL(url)  assert bool == True | | Checking if the URLs have been stored in the urls.json file with the function.  Type: $ py.test to run the test  Make sure the function name is satisfied with the naming convention.  \* File name should start with the test along with the name of the file.We need to mention the underscore symbol “\_” to separate terms test and filename for visualization purpose.  Below is the example of the file name which is actually looks like.  "test\_FileName.py" |
| Add a way to Monitor and Notify you | | |
| Add CloudWatch Agent to Monitor your Jenkins Server | | |
|  | | To download the Agent package:  $wget <https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb>    To install the Package:  $sudo dpkg -i -E ./amazon-cloudwatch-agent.deb    Once the installation is done:  $cd /opt/aws/amazon-cloudwatch-agent/bin    To start the configuration wizard  $sudo ./amazon-cloudwatch-agent-config-wizard |
| Text  Description automatically generated | | To start the CloudWatch Agent  $ sudo amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -c ssm:AmazonCloudWatch-jenkins -s  To see if the agent is running:  $ systemctl status amazon-cloudwatch-agent.service |
| Graphical user interface, application  Description automatically generated | | After the Agent is running:    Go to AWS Console> Cloudwatch > Metrics > all Metrics > CWAgent |
|  | | If you choose the instance name and the parameters, you want the data on you can see the usage on the metrics. |
| Add a way to notify you: | |
| Icon  Description automatically generated  A picture containing rectangle  Description automatically generated | | On your AWS console  Go to Cloud watch > Alarms  Click on Create Alarm |
| Graphical user interface, text, application, email  Description automatically generated | | Click on Select Metric |
| Graphical user interface, text, application, email  Description automatically generated | | Select the Metric you want the alarm to be created on. In our case since we have AWS Cloudwatch agent applied to the Jenkins server, we will use CWAgent |
| Shape  Description automatically generated with low confidence | | We will choose CPU usage system as the metric.  Then click select metric |
| Graphical user interface, application  Description automatically generated | | Choose your conditions and in our case we will send alarm if the threshold is above the 1 value. |
| Graphical user interface, text, application, email  Description automatically generated | | You can create new topic but since there is already one created, we will use BillingAlert SNS for our case. |
| Graphical user interface, application  Description automatically generated | | Create a name for the alarm and then click next.  Click on Create alarm |
|  | | |
| Use Cypress for testing | | |
|  | | $ sudo apt install  The following:  default-jre, git, nodejs, npm |
| Graphical user interface, application  Description automatically generated with medium confidence | | then go to the application dir and  $ npm install  $ npm install cypress --save-dev  add a Cypress test file inside the cypress folder created.  Run the application, and then run the  Cypress test:  $ npx cypress run --spec ./cypress/e2e/test.cy.js |
| Graphical user interface, application  Description automatically generated | | Test Results. |
| Add a Linter | | |
|  | | To add a linter:  Go to Extension in VS Code > search Linter > Install |
|  | | Linter is installed |
|  | | Check your code after installing Linter |
|  | | Quick Fix for your Linter. |
| Change something on the application front | | |
|  | | After changing the front end,   * Kura Labs Logo on top * Added background color on back   We will have to do the build again:  $ git push the updates to GitHub  Go to Jenkins > main > do a new build.  build should automatically deploy to Elastic beanstalk. |
|  | | |

|  |  |
| --- | --- |
| What Improvements can be made? | |
| **Idea #1)** Rapid Deployment with Jenkins (this has been implemented in this 2nd Deployment)  Upon sucessful Jenkins build and test. Use jenkins to compress the github repo and deploy to the Elasticbean automatically. | |
|  | <https://www.serverkaka.com/2019/04/deploy-java-web-app-elastic-beanstalk-jenkins.html>  This will make the process seamless and without needs to zip download and then upload to the elastic beanstalk env. |
| **Idea #2)** Create a separate Staging and Production Env so you can test your codes before you sent the developers code to production. | |
| **Idea #3)** Scripts need to be checked against known attacks and malicious scripts databases to make sure that bad lines of code never get to the production environment. | |
| **Idea #4)** Instead of hosting the Jenkins Server on a different VPC we can host the Jenkins Server on Private Subnet in a VPC and the agent on a public Subnet. | |

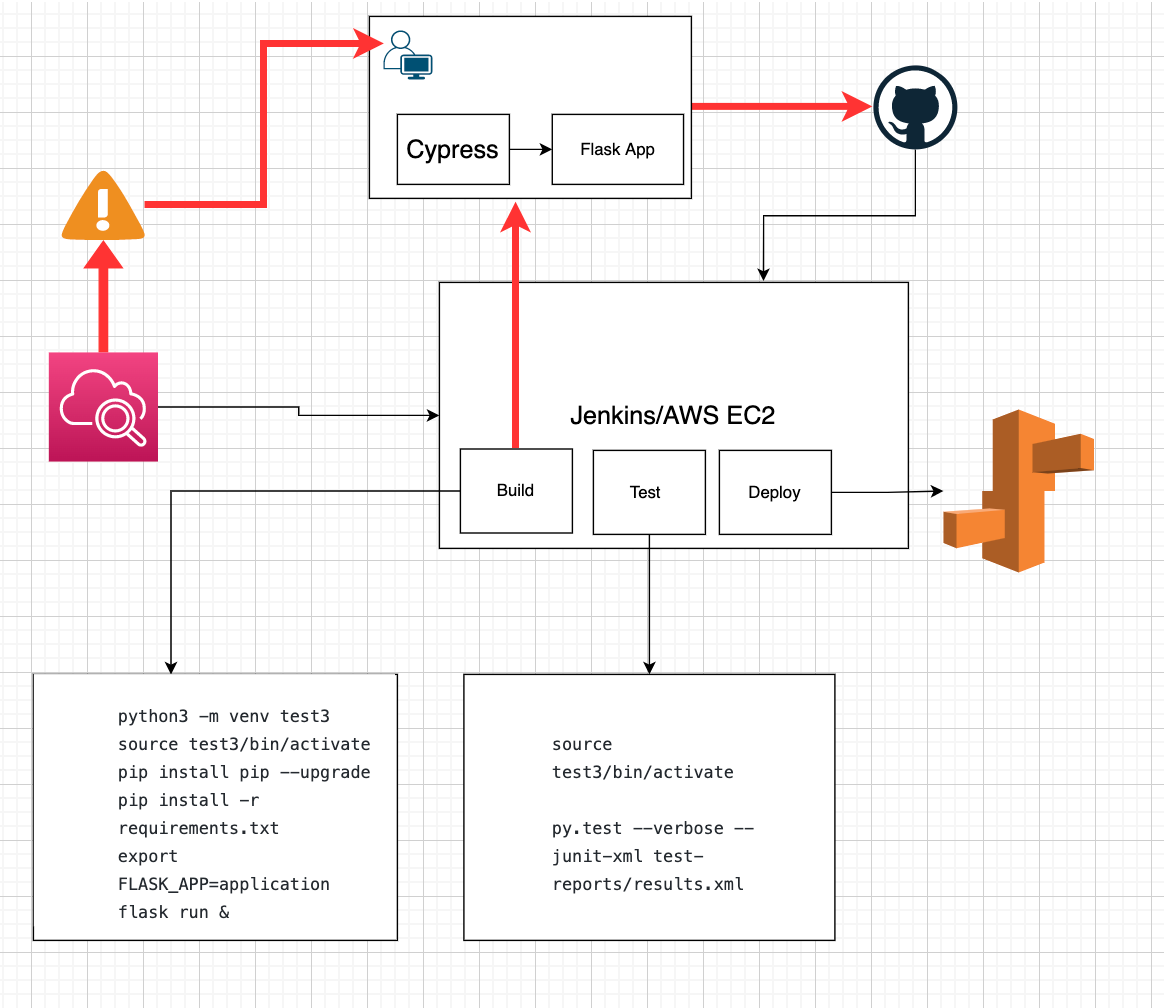
|  |
| --- |
| What stack has been implemented? |
| Explaining the Stack, we are using: LEJP **L** - **Linux** (Ubuntu) as our Operating system to hose the application  **E** – **(Engine – X) Nginx** as our Web server that will be used as Load Balancer + Reverse proxy  **J** – **JSON** file to hold our data  **P** – Python as our scripting Language |
| How is the application running – Overview? |
| Step 1) After the developer commits his code to GitHub, first we will fork the REPO from the developer  Step 2) Create a new EC2 in a VPC with Public subnet, this will be our server for Jenkins Master  Step 3) Start Jenkins and then link your GitHub Repo information to the Jenkins  Step 4) Create another EC2 in a different VPC with Public subnet, this will be our server for Jenkins Agent  Step 5) Install Nginx, gUnicorn and other necessary application on this server. And then Start Nginx.  Step 6) Add the Jenkins Agent IP and the SSH key to the Build of the Jenkins Master  Step 7) Build your Application from Jenkins and you should see the application running on the Agent Server. |

 CI/CD Pipeline Diagram 1:

Diagram

Description automatically generated

CI/CD Pipeline Diagram 2:



# CI/CD Pipeline Diagram 3:

Diagram

Description automatically generated