Deployment 2

Brayan Molina
September 25th, 2022

Documentation:

Jenkins – Installation, Configuration, and Activation:

Objective: Purpose of this section is to configure an EC2 instance with Jenkins installed. A script can be used to configure the EC2 with Jenkins installed upon start. Activating the Jenkins user will be needed to connect to AWS services.

Jenkins Installation and Configuration

- A network security group needs to be configured to have ports 22,80, and 8080 open to allow SSH, HTTP, and Jenkins services.
- Jenkins account creation can be skipped and proceed as admin. Secret password found in path provided by Jenkins account set-up wizard.
- Include script upon creation of EC2 to have Jenkins installed.

Jenkins User Activation

- In EC2 Terminal:
    - Activate the Jenkins user by creating a password and switching into Jenkins user.
    - Command to switch into Jenkins user requires -s option since Jenkins is originally configured without any shell configuration. Including /bash/shell for Jenkins allows us to use bash commands in the CLI.
- On AWS Identity and Access Management (IAM):
    - Creating a user in AWS allows us to give permissions to that user to interact with AWS services. In this case, the Jenkins user will receive permissions to deploy the application through Elastic beanstalk.
    - Selecting AWS credential type – Programmatic access provides us with the Access Key ID and Secret Access Key used to configure the AWS CLI.
    - Administrator access policy means Jenkins user has full access and can delegate permissions to every service and resource in AWS.

AWS CLI – Installation and Configuration:

Objective: Correctly install and configure AWS CLI in the EC2 Instance.

AWS CLI – Installation:

- The curl command retrieves the installation file and -o option will write the package to the specified file name.
- This portion requires installation of the unzip command.
- Install the AWS CLI by following the commands provided and verify the existing version of the AWS CLI once installation is completed.

AWS CLI – Configuration:

- Configuration of AWS CLI is done in the Jenkins user.
- Run the command AWS configure and include the following parameters to ensure AWS CLI is connected to the user made in the IAM.
  - `Set Access Key ID - This will be from AWS user made`
  - `Set Secret Access Key - This will be from AWS user made`
  - `Set region to: us-east-1`
  - `Set output format: json`

AWS EB CLI – Installation

Objective: Installation of Elastic Beanstalk through the AWS CLI.

- The following commands should be ran to ensure the eb command is properly installed:

```
sudo su - jenkins -s /bin/bash

export PATH="/var/lib/jenkins/.local/bin:$PATH"

pip install awsebcli --upgrade --user

eb --version
```

- Exporting the path of the bin where eb is installed is necessary for the command to be used anywhere in the CLI. Without exporting the command, eb cannot be used.
- Export PATH must be used every time user logs back into Jenkins user.

GitHub and Jenkins Pipeline:

GitHub:

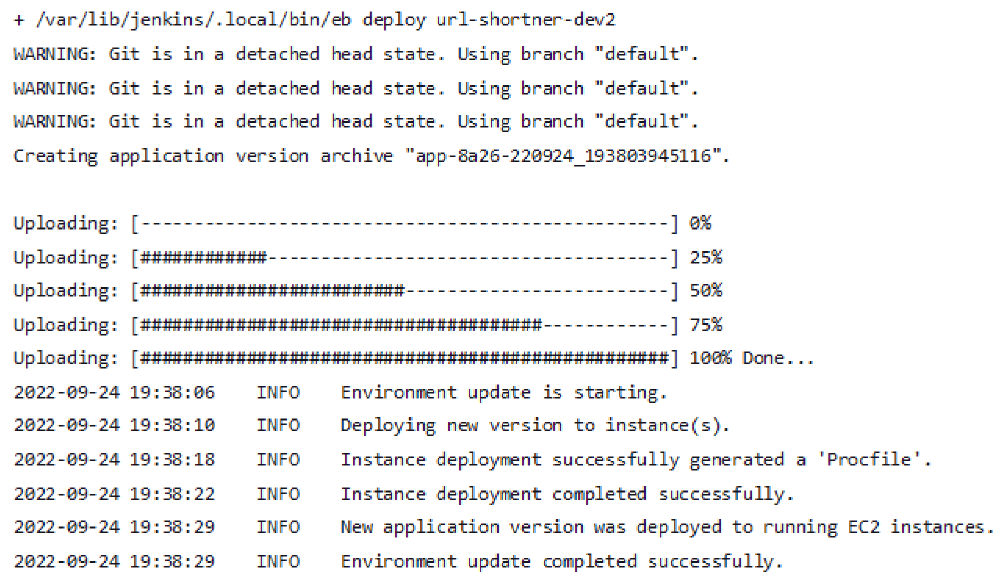
- Fork the deployment repository https://github.com/kura-labs-org/kuralabs_deployment_2.git
- Creating an access token allows Jenkins to be able access the source code repository when providing our credentials.

Jenkins:

- Creating a multibranch pipeline allows Jenkins to gain access to all branches from our source code repository. This deployment consists of one main branch, but generally other repositories may include multiple branches.
- Use GitHub credentials and deployment repository to properly configure Jenkins job.
- Jenkins will use the Jenkinsfile provided in the repository to run the commands defined in the build and test stage.

Deployment of application from Elastic Beanstalk CLI:

- Switch into Jenkins user and initialize elastic beanstalk through eb init. Create the virtual environment for your application through eb create.
- Once eb create is entered, Elastic beanstalk will deploy the application and we can verify using the URL provided.
- Deployment stage added to Jenkinsfile uses eb deploy. This command will be used to deploy any changes made to our application or pipeline. If successful, the command returns the status of the deploy operation.
- Result of deployment step from Jenkins can be seen below.



```
+ /var/lib/jenkins/.local/bin/eb deploy url-shortner-dev2
WARNING: Git is in a detached head state. Using branch "default".
WARNING: Git is in a detached head state. Using branch "default".
WARNING: Git is in a detached head state. Using branch "default".
Creating application version archive "app-8a26-220924_193803945116".

Uploading: [--------------------------------------------------] 0%
Uploading: [############--------------------------------------] 25%
Uploading: [#########################-------------------------] 50%
Uploading: [######################################------------] 75%
Uploading: [##################################################] 100% Done...
2022-09-24 19:38:06    INFO    Environment update is starting.
2022-09-24 19:38:10    INFO    Deploying new version to instance(s).
2022-09-24 19:38:18    INFO    Instance deployment successfully generated a 'Procfile'.
2022-09-24 19:38:22    INFO    Instance deployment completed successfully.
2022-09-24 19:38:29    INFO    New application version was deployed to running EC2 instances.
2022-09-24 19:38:29    INFO    Environment update completed successfully.
```

Modification to pipeline:

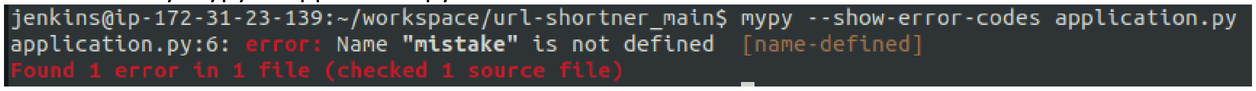
Linter: A linter provides a way for developers to check syntax prior to pushing to GitHub repository. Additionally, linter can be used to verify syntax for any modification made to the jenkinsfile and confirm there are no syntax errors. Below are two uses of linters for python and groovy: mypy and groovy-lint.

Instructions – Mypy for Python applications:

Installation of mypy can be done through apt package manager or using the python3 module option.

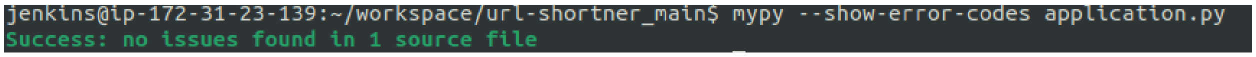
```
python3 -m pip install mypy or sudo apt install mypy

mypy --show-error-codes {{application.py}}
```

Error found by mypy in application.py:



```
jenkins@ip-172-31-23-139:~/workspace/url-shortner_main$ mypy --show-error-codes application.py
application.py:6: error: Name "mistake" is not defined  [name-defined]
Found 1 error in 1 file (checked 1 source file)
```

No errors found in application.py:



```
jenkins@ip-172-31-23-139:~/workspace/url-shortner_main$ mypy --show-error-codes application.py
Success: no issues found in 1 source file
```

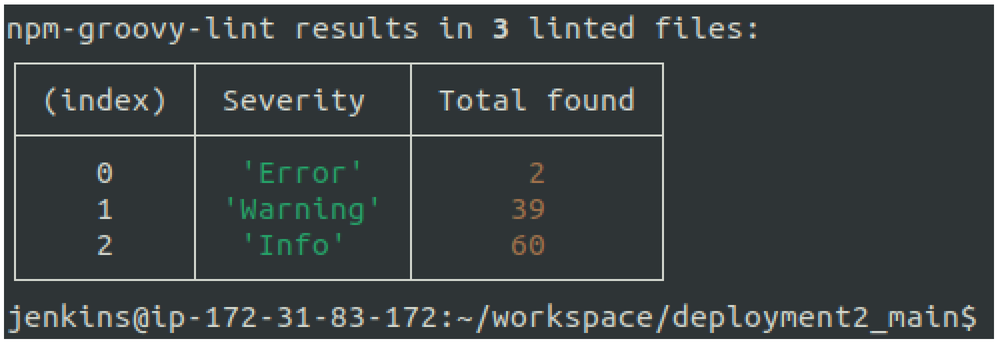
Instructions – Groovy lint for Jenkinsfile syntax:

On EC2:

```
sudo npm install -g npm-groovy-lint
```

On Jenkins User:

```
sudo su - jenkins -s /bin/bash

npm-groovy-lint  // in the root directory of the Jenkinsfile

Results:
```



```
npm-groovy-lint results in 3 linted files:
┌─────────┬───────────┬─────────────┐
│ (index) │ Severity  │ Total found │
├─────────┼───────────┼─────────────┤
│    0    │  'Error'  │      2      │
│    1    │ 'Warning' │     39      │
│    2    │  'Info'   │     60      │
└─────────┴───────────┴─────────────┘
jenkins@ip-172-31-83-172:~/workspace/deployment2_main$
```

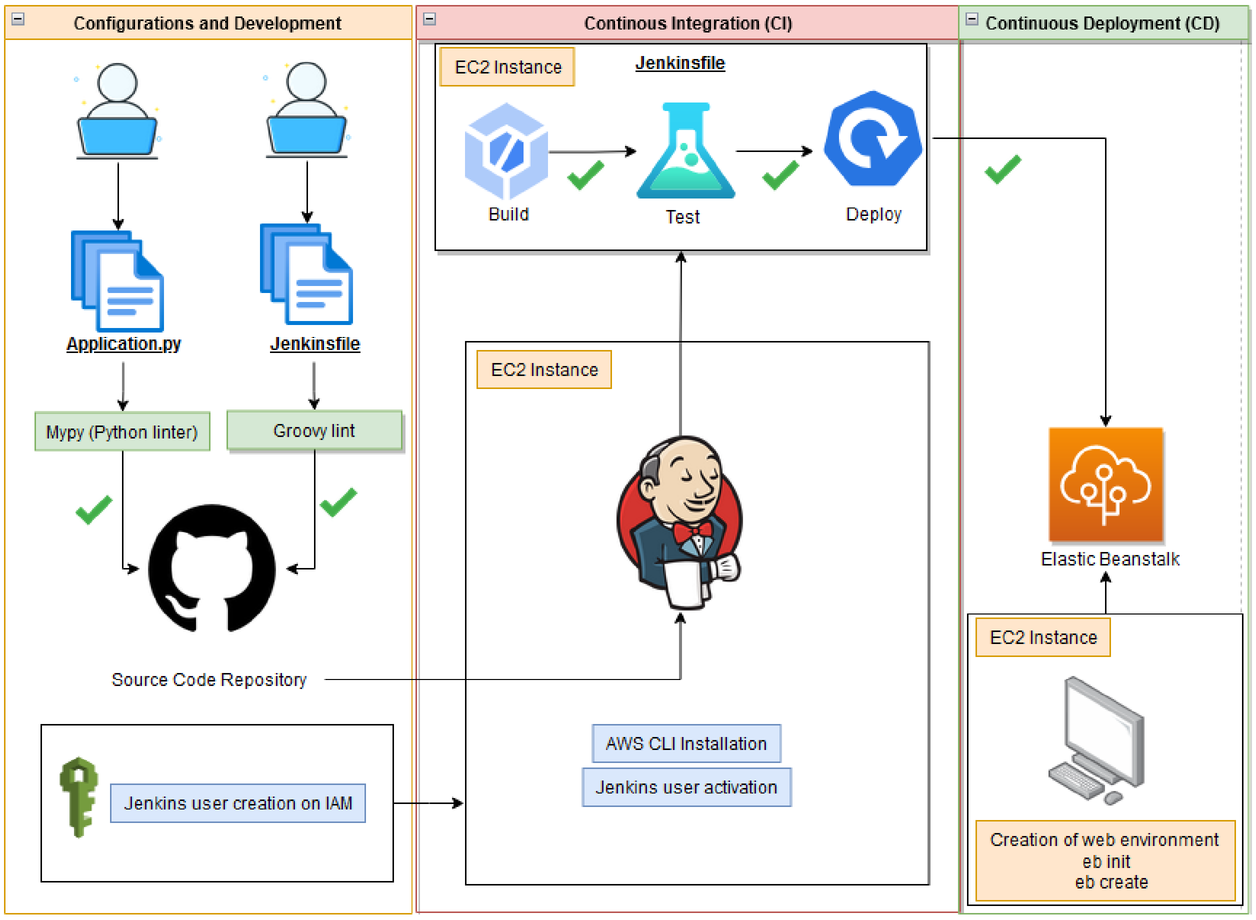
Observations and errors:

- If EC2 is restarted, export PATH="/var/lib/jenkins/.local/bin:$PATH" needs to be ran in the command line.
- /bin/bash necessary for switching into jenkins user since shell is not configured when jenkins user is made: /bin/false
- Remove elastic beanstalk configurations if there are issues when restarting the EC2 server. This will allow you to restart the elastic beanstalk deployment process through eb init and eb create:

```
o  CD into workspace/{{project name}}
o  ls -a
o  rm -rf .elasticbeanstalk
```

Error	*Cause*
The virtual environment was not created successfully because ensurepip is not available. On Debian systems, you need to install the python3-venv package using the following command apt install python3.10-venv	Python 3.10-venv not installed. Run the following to resolve issue: sudo apt install python3.10-venv
Eb version not confirmed – Requires jenkins user bin folder to be added to PATH.	Requires addition of jenkins bin directory to PATH. Run the following to resolve issue: export PATH="/var/lib/jenkins/.local/bin:$PATH"

Diagram:



Key takeaways:

1. Source code application and Jenkinsfile should go through a linter to ensure syntax errors are removed.
2. Jenkins user creation through IAM is necessary to allow permissions for configuration of AWS CLI and elastic beanstalk through the EC2 terminal.
3. Creation of web environment is made through eb init and eb create. eb deploy will be used in Jenkinsfile to deploy the application when changes are made to the source code repository.