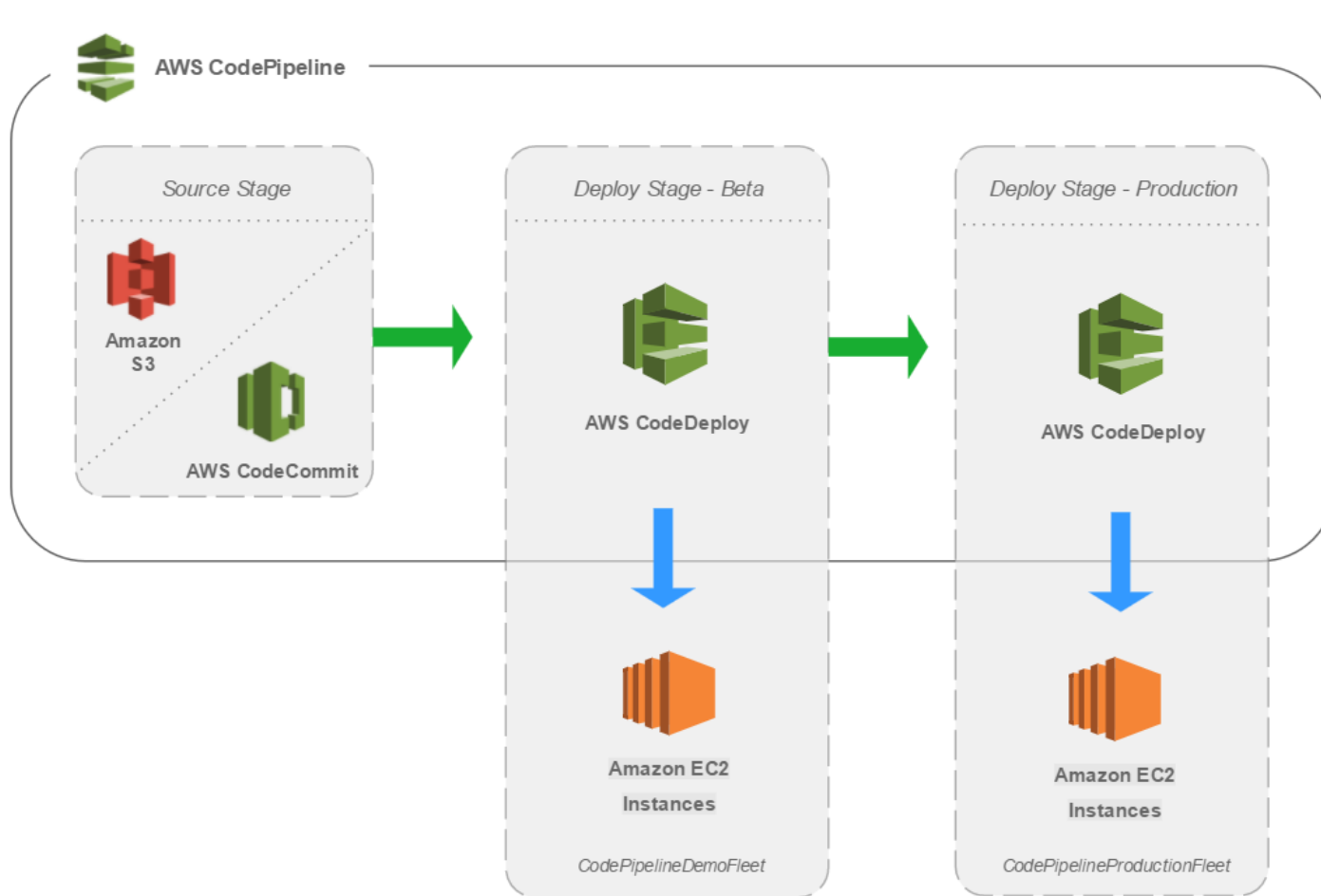


Deploy Django Application at A.W.S Using CI/CD pipelines

Posted by: Anees Rehman Khan

2023-03-26T06:48:49.981030Z

In this blog, you will learn how to set up a continuous integration and continuous delivery (CI/CD) pipeline on AWS. A pipeline helps you automate steps in your software delivery process, such as initiating automatic builds and then deploying to Amazon EC2 instances. You will use AWS CodePipeline, a service that builds, tests, and deploys your code every time there is a code change, based on the release process models you define. Use CodePipeline to orchestrate each step in your release process. As part of your setup, you will plug other AWS services into CodePipeline to complete your software delivery pipeline. This guide will show you how to create a very simple pipeline that pulls code from a source repository and automatically deploys it to an Amazon EC2 instance.



Here are the following steps we need to follow for deployment by building cicd pipeline

1. Create your AWS Account
2. Setup IAM Role

3. EC2 Instance Creation
 4. CodeDeploy Installation
 5. Setting Up Code Structure
 6. Create Configuration Files
 7. Code Pipelines
-

IAM Role setup

1.create first role

Search for IAM in search bar > click on Roles > Create Role
> keep default aws services > select Use case as EC2
>next>

ADD Permissions

search for the role "amazonec2roleforawscodedeploy"
select the starting checkbox > next > give role name and
scroll down create the role for ec2

2. create second role

click on Roles > Create Role > keep default aws services >
select Use case dropdown select CodeDeploy
>CodeDeploy>Next>next>give role name aws code deploy
role and create the role

2. Search for EC2

Ec2 home page

1.Instance > click on launch instance > give instance name
<django-server>

2. Select operation system as ubuntu 20.04 and keep
default config

3. Key pair login (create new key pair) > give key pair
name (django-server keep rsa,.pem

4. Enable fire walls > Allow HTTP traffic from internet and
all Allow SSH traffice from internet keep checked

5. Rest keep all default and click on launch instance

In home page click our instance and click on actions on
header > security > modify iam role > chose iam role which
we created and update iam role

now we need to reebot select> instance state >reboot
instance

3. CODE DEPLOY INSTALLATION

Login to the server how to see credentials

1. click on to the instance > connect to instance > SSH client

we have downloaded the .pem filer django-server.pem we have to create a directory add that file and open same folder in terminal and run the those two commands first run chmod command and then ssh -i command and give yes access we will login into ece machine

run the commands

1. sudo apt update
2. sudo apt install ruby-full
3. sudo apt install wget

we need to enter this command:

wget

```
https://bucket-name.s3.region-identifier.amazonaws.com/  
latest/install
```

For Example:

Bucket-name for the **US-East-1-region(N.Virginia)** is
aws-codedeploy-us-east-1

Region-identifier is **us-east-1**

```
wget https://aws-codedeploy-us-  
east-1.s3.us-east-1.amazonaws.com  
/latest/install
```

in userguides we can see region names
and bucketname

ext up, we need to change the
permission on the install file we will
get after running the command above.

```
$ chmod +x ./install
```

Finally, to install the codedeploy-agent,
run this command:

```
$ sudo ./install auto > /tmp/logfile
```

Here we are logging the output of the installation to the /tmp/logfile file. To check if the codedeploy-agent is running, enter this command:








```
$ sudo service codedeploy-agent status
```

If it is not running, enter this command to start the codedeploy-agent service:














```
$ sudo service codedeploy-agent status
```

4. Project Structure Configuration

1. Add Configuration files and rename the foldername and project name accordingly in all the configuration files

 .git	26-03-2023 12:10	File folder	
 gunicorn	26-03-2023 12:10	File folder	
 nginx	26-03-2023 12:10	File folder	
 scripts	26-03-2023 12:10	File folder	
 appspec.yml	26-03-2023 12:10	Yaml Source File	1 KB
 buildspec.yml	26-03-2023 12:10	Yaml Source File	1 KB
 README.md	26-03-2023 12:10	MD File	1 KB

2. Add your django project

 app	26-03-2023 11:32	File folder	
 blog	26-03-2023 11:35	File folder	
 gunicorn	28-02-2023 22:07	File folder	
 media	26-03-2023 11:32	File folder	
 nginx	28-02-2023 22:07	File folder	
 scripts	28-02-2023 22:07	File folder	
 static	26-03-2023 11:32	File folder	
 templates	26-03-2023 11:32	File folder	
 appspec.yml	28-02-2023 22:07	Yaml Source File	1 KB
 buildspec.yml	28-02-2023 22:07	Yaml Source File	1 KB
 db.sqlite3	26-03-2023 11:35	SQLITE3 File	144 KB
 manage.py	26-03-2023 11:32	Python File	1 KB
 requirements.txt	26-03-2023 11:28	Text Document	1 KB

Add public ip adress from ec2 instance and add in
nginx.conf file at server_name

5. Codepipeline Deploy

Go to console of aws search for Codepipeline > Build >
Create build Project > give project name > Source > source
provider is github > connect your github add the repository
link (the above image project structure just push
everything into the repository and provide the link of it) >
Operation system > ubuntu > runn time standard > image
<aws/codebuild/standard:6.0> we will go with new service
role > build spec file keep default > create build project

Select menu left as Deploy > Applications > create
application > give application name > choose platform as

EC2/On-premises > create it

Create a deployment group > enter deployment group
name > Service role search iam role and chooses >
deployment type same >

Environment configuration > enable Amazon EC2 instances
key as Name Value as django_server

Keep other setting default disable the load balancer

create it...

6. Create a pipeline

Developer Tools > CodePipeline > Pipelines > Create new
pipeline

give pipeline name > chose new service role

advance setting both default

Add source stage > select as github version 1 > connect to
github > choose the github repository

Build Provider > aws codeBuild select project name

Add deploy stage > Aws Codedeploy > select application
name we created and deployment group select it which we
created

Review the steps and create the pipeline

it will trigger and start build all the changes so whenever we push the code it automatically works and get deployed