Jenkins Freestyle Job

Introduction

A **Jenkins Freestyle Job** is a simple, flexible job type in Jenkins that allows users to define custom build processes without needing specialized configurations. It offers the most basic way to run continuous integration (CI) pipelines, where you can configure various steps like pulling code from a version control system, running shell commands, building the project, and deploying.

Use Cases of Jenkins Freestyle Jobs

1. **Basic Builds**: For compiling code, running unit tests, or packaging artifacts without needing advanced configurations.
2. **Running Custom Scripts**: Freestyle jobs allow running scripts (like shell, Python, batch) as part of the build process.
3. **Manual Builds**: When there's no need for complex CI/CD pipelines, you can use freestyle jobs for manual or simple automated builds.
4. **Initial Project Setup**: It's ideal for smaller projects or teams just starting with Jenkins, as it doesn't require complex configuration or setup.
5. **Advantages of Jenkins Freestyle Jobs**
   1. **Simplicity**: It's easy to configure, making it a good starting point for beginners or for simple automation tasks.
   2. **Flexibility**: You can run any shell script or build command, making it adaptable to many different tasks or environments.
   3. **Quick Setup**: Since it doesn’t need specific plugins or complex setup, you can create a freestyle job very quickly.

Disadvantages of Jenkins Freestyle Jobs

1. **Lack of Structure**: For complex pipelines, freestyle jobs can become difficult to maintain as they lack the structured flow that modern pipelines (like Declarative Pipelines) provide.
2. **Difficult to Scale**: When you need to scale to more complex workflows (e.g., parallel builds, advanced branching logic), freestyle jobs become inadequate
3. **Low Reusability**: Freestyle jobs are less reusable across projects compared to Jenkins Pipelines, where you can modularize stages and steps.
4. **Not Ideal for DevOps Practices**: Modern DevOps practices like Infrastructure as Code and continuous delivery benefit more from pipeline jobs, which can

model complex workflows and environments, while freestylejobs fall short in these areas.

Steps to Run the Freestyle Job

Prerequisites

Two EC2 instances: one with Jenkins installed and another with your Django Application.

Switch to Jenkins User

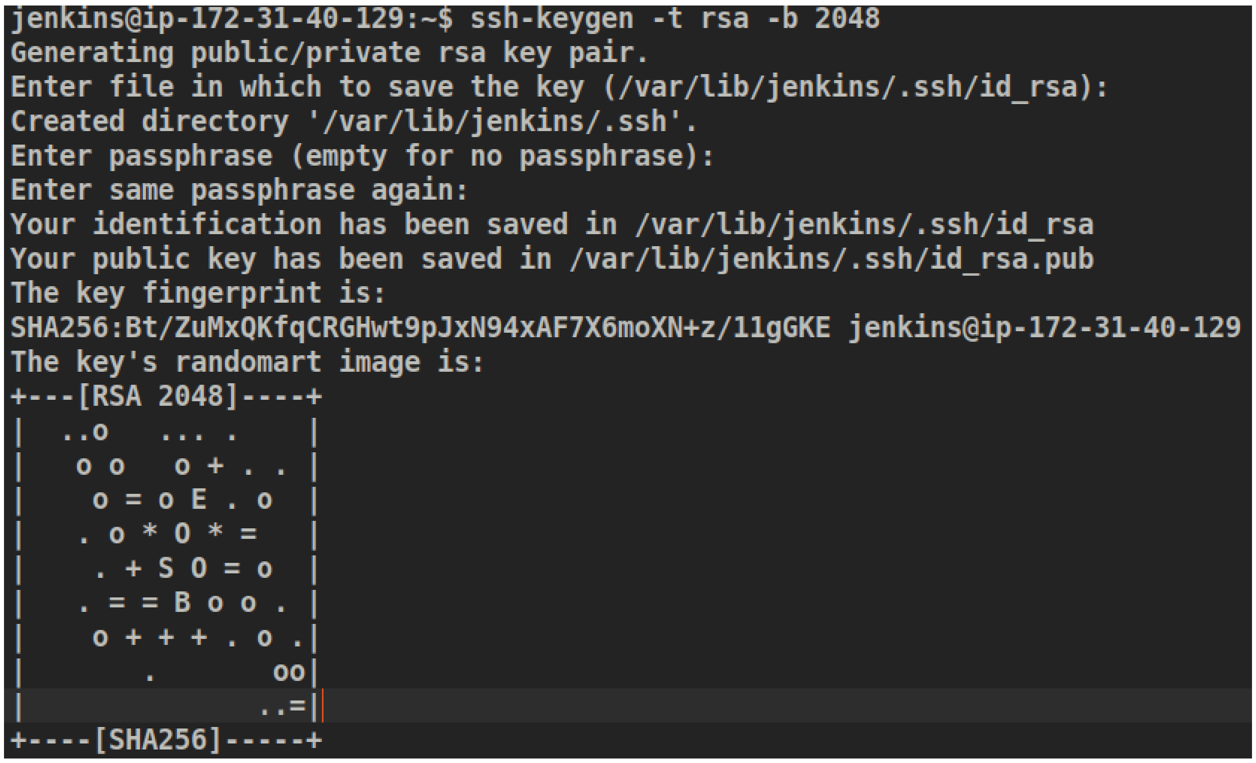
SSH into ec2 instance having jenkins and switch to jenkins user

```
sudo su - jenkins
```

Generate SSH Key Pair

● Press Enter to accept the default file location.
● Optionally set a passphrase.

```
ssh-keygen -t rsa -b 2048
```



```
jenkins@ip-172-31-40-129:~$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa):
Created directory '/var/lib/jenkins/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Bt/ZuMxQKfqCRGHwt9pJxN94xAF7X6moXN+z/1lgGKE jenkins@ip-172-31-40-129
The key's randomart image is:
+---[RSA 2048]----+
|  ..o   ... .    |
|   o o   o + . . |
|    o = o E . o  |
|   . o * O * =   |
|    . + S O = o  |
|   . = = B o o . |
|    o + + + . o .|
|       .       oo|
|              ..=|
+----[SHA256]-----+
```

Copy the Public Key

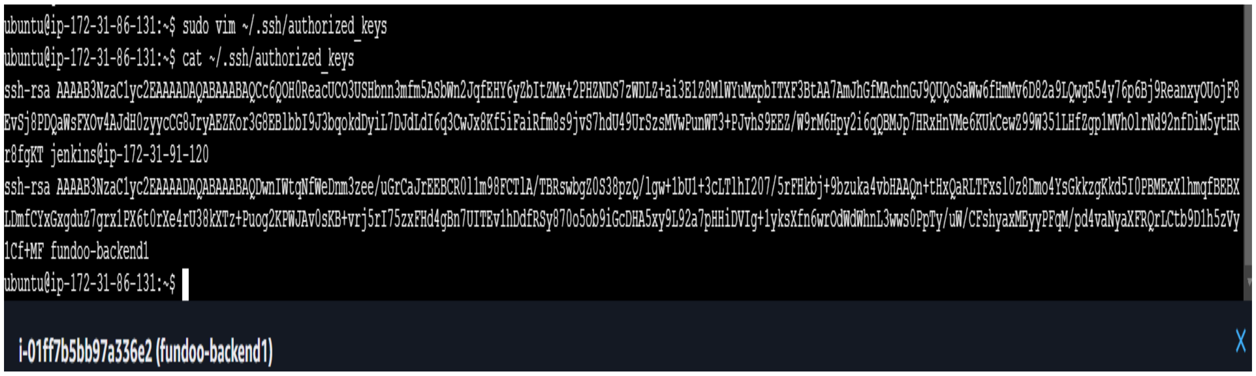
From the generated rsa key pair copy the public key

`cat ~/.ssh/id_rsa.pub`

Copy the public key to the Django instance

Access the Django EC2 instance (backend) and carefully paste the generated public key into the .ssh folder of this instance. This will allow us to SSH into the Django instance from the Jenkins instance without needing to manually add the public key each time.

`sudo vim .ssh/authorized_keys`



```
ubuntu@ip-172-31-86-131:~$ sudo vim ~/.ssh/authorized_keys
ubuntu@ip-172-31-86-131:~$ cat ~/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCc6QOH0ReacUCO3USHbnn3mfm5ASbWn2JqfEHY6yZbItZMx+2PHZNDS7zWDLZ+ai3E1Z8MlWYuMxpbITXF3BtAA7AmJhGfMAchnGJ9QUQoSaWw6fHmMv6D82a9LQwgR54y76p6Bj9ReanxyOUojF8
EvSj8PDQaWsFXOv4AJdH0zyycCG8JryAEZKor3G8EBlbbI9J3bqokdDyiL7DJdLdI6q3CwJx8Kf5iFaiRfm8s9jvS7hdU49UrSzsMVwPunWT3+PJvhS9EEZ/W9rM6Hpy2i6qQBMJp7HRxHnVMe6KUkCewZ99W351LHfZgp1MVhOlrNd92nfDiM5ytHR
r8fgKT jenkins@ip-172-31-91-120
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDwnIWtqNfWeDnm3zee/uGrCaJrEEBCR0l1m98FCTlA/TBRswbgZ0S38pzQ/lgw+1bU1+3cLTlhI207/5rFHkbj+9bzuka4vbHAAQn+tHxQaRLTFxsl0z8Dmo4YsGkkzgKkd5I0PBMExXlhmqfBEBX
LDmfCYxGxgduZ7grx1PX6t0rXe4rU38kXTz+Puog2KPWJAv0sKB+vrj5rI75zxFHd4gBn7UITEv1hDdfRSy870o5ob9iGcDHA5xy9L92a7pHHiDVIg+1yksXfn6wrOdWdWhnL3wws0PpTy/uW/CFshyaxMEyyPFqM/pd4vaNyaXFRQrLCtb9D1h5zVy
1Cf+MF fundoo-backend1
ubuntu@ip-172-31-86-131:~$
```

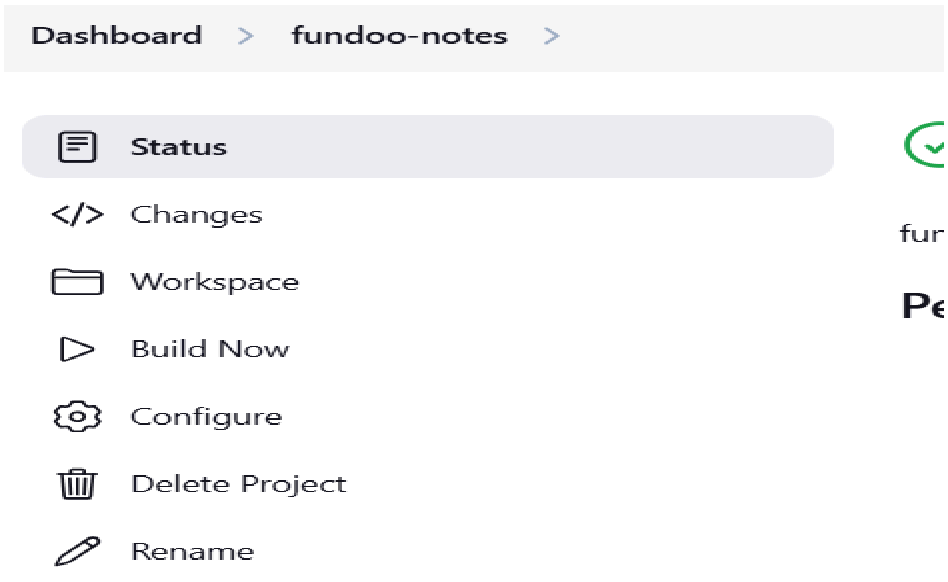
i-01ff7b5bb97a336e2 (fundoo-backend1)

Create a New Freestyle Project in Jenkins

- From the Jenkins dashboard, click on New Item.
- Enter a name for your job (e.g., DjangoAppBuild) and Select Freestyle project.

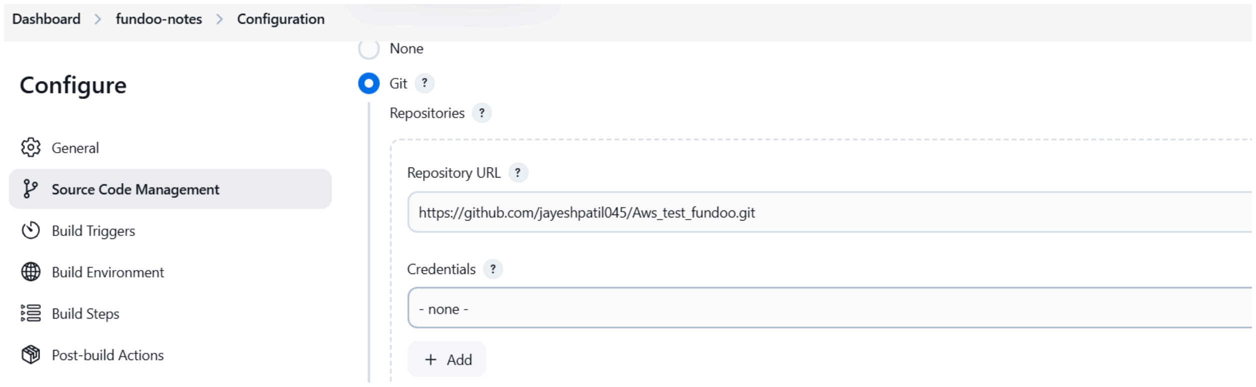
Configure Project

- In the job configuration page, scroll to General



Configure Git repository

- In the job configuration page, scroll down to Source Code Management.
- Select Git.
- Enter your repository URL (e.g., git@github.com:username/repo.git).
- If needed, configure credentials by clicking on Add next to Credentials



Repository URL: https://github.com/jayeshpatil045/Aws_test_fundoo.git

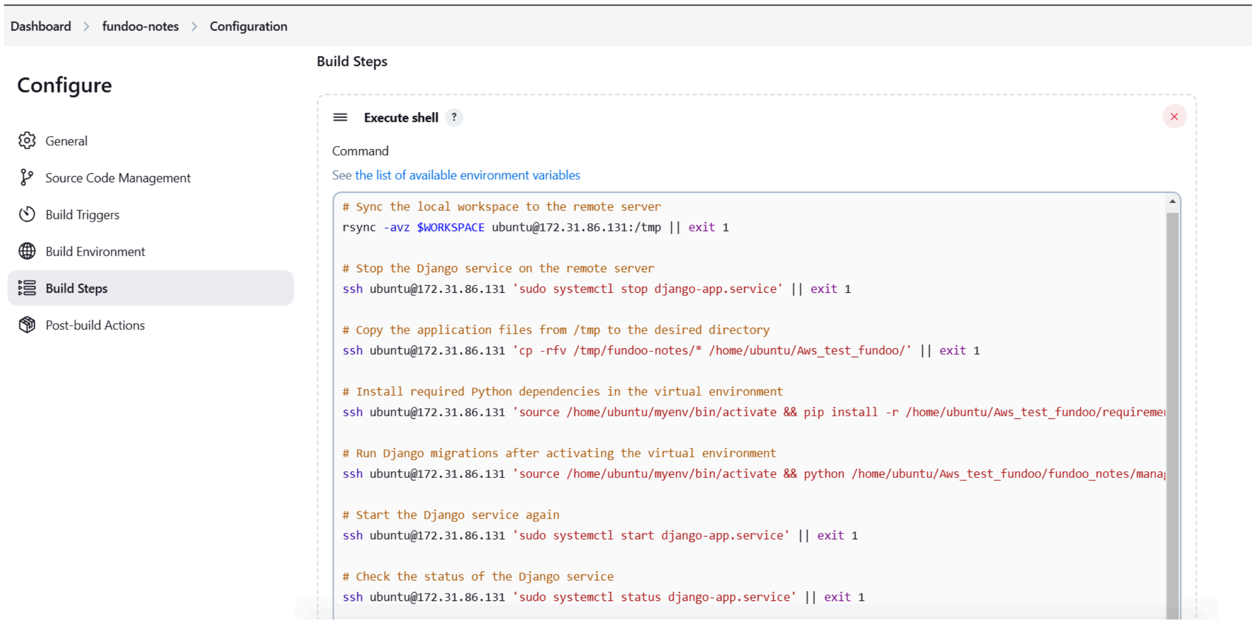
Credentials: - none -

Configure Build Steps

Here we configure our build

Select Discard Old builds

Build Steps



```
# Sync the local workspace to the remote server
rsync -avz $WORKSPACE ubuntu@172.31.86.131:/tmp || exit 1

# Stop the Django service on the remote server
ssh ubuntu@172.31.86.131 'sudo systemctl stop django-app.service' || exit 1

# Copy the application files from /tmp to the desired directory
ssh ubuntu@172.31.86.131 'cp -rfv /tmp/fundoo-notes/* /home/ubuntu/Aws_test_fundoo/' || exit 1

# Install required Python dependencies in the virtual environment
ssh ubuntu@172.31.86.131 'source /home/ubuntu/myenv/bin/activate && pip install -r /home/ubuntu/Aws_test_fundoo/requireme

# Run Django migrations after activating the virtual environment
ssh ubuntu@172.31.86.131 'source /home/ubuntu/myenv/bin/activate && python /home/ubuntu/Aws_test_fundoo/fundoo_notes/mana

# Start the Django service again
ssh ubuntu@172.31.86.131 'sudo systemctl start django-app.service' || exit 1

# Check the status of the Django service
ssh ubuntu@172.31.86.131 'sudo systemctl status django-app.service' || exit 1
```

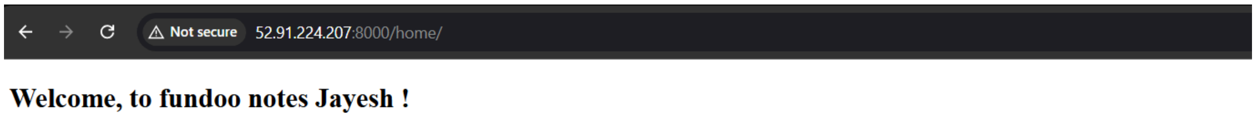
Save and Build

- Click on the Save button at the bottom of the configuration page.
- To run your job, go back to the job page and click on Build Now

Monitor the build process by clicking on the build number in the build history.

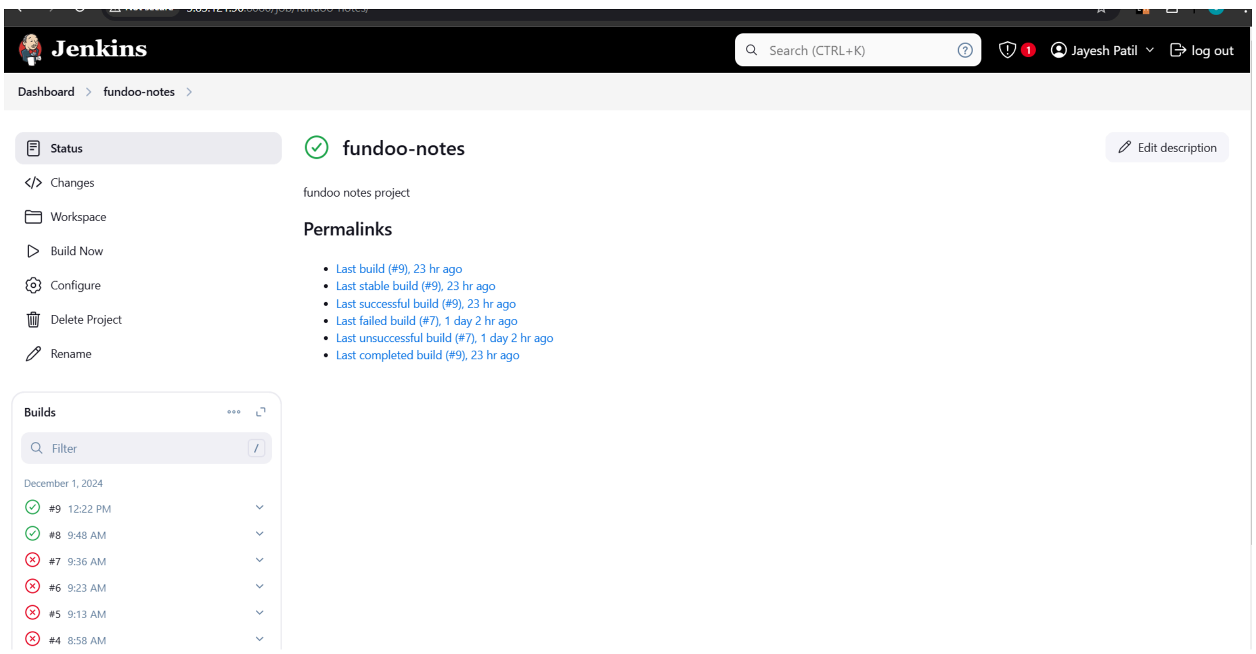
Verify Deployment

Once the build is complete, verify that your Django application is running correctly by accessing it via its public IP address or domain name.



Not secure 52.91.224.207:8000/home/

Welcome, to fundoo notes Jayesh !



fundoo-notes

fundoo notes project

Permalinks

- Last build (#9), 23 hr ago
- Last stable build (#9), 23 hr ago
- Last successful build (#9), 23 hr ago
- Last failed build (#7), 1 day 2 hr ago
- Last unsuccessful build (#7), 1 day 2 hr ago
- Last completed build (#9), 23 hr ago

Builds

December 1, 2024

- #9 12:22 PM
- #8 9:48 AM
- #7 9:36 AM
- #6 9:23 AM
- #5 9:13 AM
- #4 8:58 AM

Conclusion

By following these steps, you should be able to successfully configure and run a freestyle job in Jenkins that pulls your Django application code from a Git repository and executes necessary commands on your EC2 instance. This setup allows for continuous integration and deployment of your application as changes are made in your codebase.