

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.

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 freestyle jobs 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]-----+
|  ..0   . . . . |
|  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
```

```
jenkins@ip-172-31-40-129:~$ cd .ssh/
jenkins@ip-172-31-40-129:~/.ssh$ ls -l
total 8
-rw----- 1 jenkins jenkins 1831 Oct 17 09:21 id_rsa
-rw-r--r-- 1 jenkins jenkins 406 Oct 17 09:21 id_rsa.pub
jenkins@ip-172-31-40-129:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCKGld4rBPXvvk53sYm4is59wTPK6zbh8bjso88mN2t7z/
NPBp0S73v0QevjacXBXKK3lEv2ysfeL3X8rXFuALWTRdxe1djzZGLR2aochx1VTbGs0XpRl1zw2DhK7vwWNZ
```

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-172-31-1-175:~$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCKGld4rBPXvvk53sYm4is59wTPK6zbh8bjso88mN2t7z/NPBp0S73v0QevjacXBXKK3l
XpRl1zw2DhK7vwWNZ5D8ujNYBG60o+fcXCe0WBMwk5fQqNLW54npBbtzKhLJ0g5pViMLwkWyOxAgw038k8oKpSYvYtptyNqcFYIIcsb241
6g2UtdUkBL5C2zultLHsa1701dI6o96mimUCUjH5DBSzzhe4DJuL6G6CyXFSpXpa43CaBoDV1uPunxHHv jenkins@ip-172-31-40-129

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCK3mJDAm4fNlNPr/iEJXV12XMA3jhMw2qxoK0a14uQmdQkMmfjz/QXbGjiuN2MSgEnmt
uGHsNgvixiPwBpij/mCpS4Ahr9fmR1ciYkc1sUi51uItKmTnzIz+vJ15aBEe3DrpgRhPJ6mLzkMc7GealXUFjDd8mPPlpoT78m+8Nx02hE
AZUM4efUW09m09q30TVoDigw9Mn8obV5XiB4edA8HYSIMv5U/DxJuhFc03mNCGD0EGQXo8HDuWwH0rCJ backend-key-pair
ubuntu@ip-172-31-1-175:~$
```

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.

Dashboard > All > New Item

New Item

Enter an item name

Select an item type

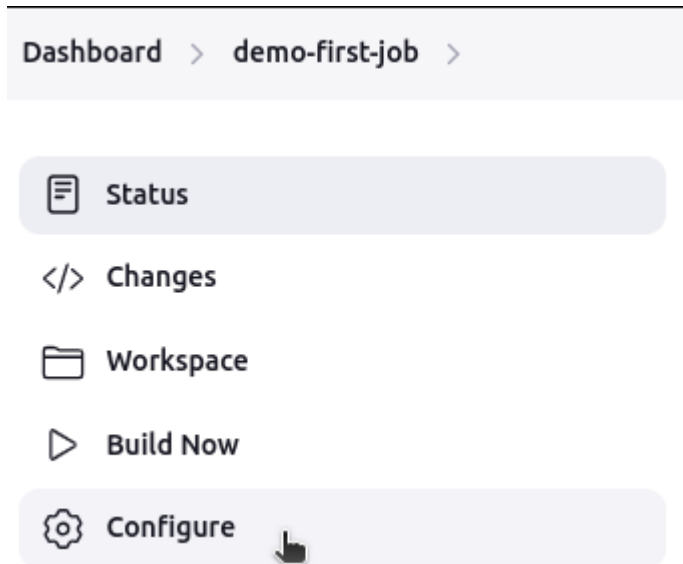


Freestyle project







Classic, general-purpose job type that checks out from up to one SCM, build steps like archiving artifacts and sending email notifications.

Configure Project

- In the job configuration page, scroll to General




Configure

-  General
-  Source Code Management
-  Build Triggers
-  Build Environment
-  Build Steps
-  Post-build Actions

Description

first project demo

Plain text [Preview](#)

☒ Discard old builds 

Strategy

Log Rotation

Days to keep builds

if not empty, build records are only kept

5

Max # of builds to keep

if not empty, only up to this number of

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](https://github.com:username/repo.git)).
- If needed, configure credentials by clicking on Add next to Credentials.

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

☐ None

☒ Git ?

Repositories ?

Repository URL ?

<https://github.com/ayush-prajapati01/fundoo-notes-copy.git>

Credentials ?

- none -

Configure Build Steps

Here we configure our build

Select Discard Old builds

Build Environment

☒ Delete workspace before build starts



Advanced ▾

☐ Use secret text(s) or file(s) ?

☐ Add timestamps to the Console Output

☐ Inspect build log for published build scans

☐ Terminate a build if it's stuck

☐ With Ant ?

Build Steps

Build Steps

≡ Execute shell ?

Command

See [the list of available environment variables](#)

```
echo "Syncing the application to backend/app server"
rsync -avz $WORKSPACE ubuntu@172.31.1.175:/tmp
ssh ubuntu@172.31.1.175 'cp -rfv /tmp/demo-first-job/* /home/ubuntu/fundoo-notes-copy/'
ssh ubuntu@172.31.1.175 'source /home/ubuntu/myenv/bin/activate && pip3 install -r /home/ubuntu/fundoo-notes-copy/requirements.txt'
ssh ubuntu@172.31.1.175 'source /home/ubuntu/myenv/bin/activate && python3 /home/ubuntu/fundoo-notes-copy/main.py'
echo "Starting Django server on port 8000"
ssh ubuntu@172.31.1.175 'sudo systemctl restart fundoo-notes.service'
ssh ubuntu@172.31.1.175 'sudo systemctl status fundoo-notes.service'
```

Save and Build

- Click on the Save button at the bottom of the configuration page.

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

```
ssh ubuntu@172.31.1.175
```

Advanced ▾

Add build step ▾

Post-build Actions

Add post-build action ▾

Save

Apply

- To run your job, go back to the job page and click on Build Now.

Dashboard > demo-first-job >

Status

</> Changes

Workspace

Build Now

Configure

Delete Project

demo-first-job

first project demo

Permalinks

- [Last build \(#17\), 2 days 20 hr ago](#)
- [Last stable build \(#17\), 2 days 20 hr ago](#)
- [Last successful build \(#17\), 2 days 20 hr ago](#)
- [Last completed build \(#17\), 2 days 20 hr ago](#)

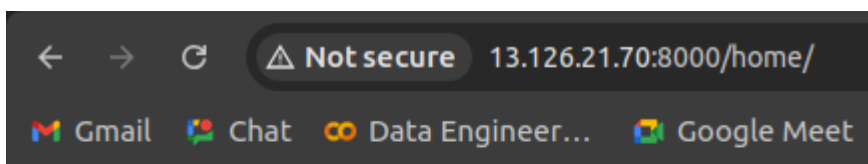
- Monitor the build process by clicking on the build number in the build history.

Build History trend ▼

#17	18 Oct 2024, 09:54
#15	18 Oct 2024, 09:21

Verify Deployment

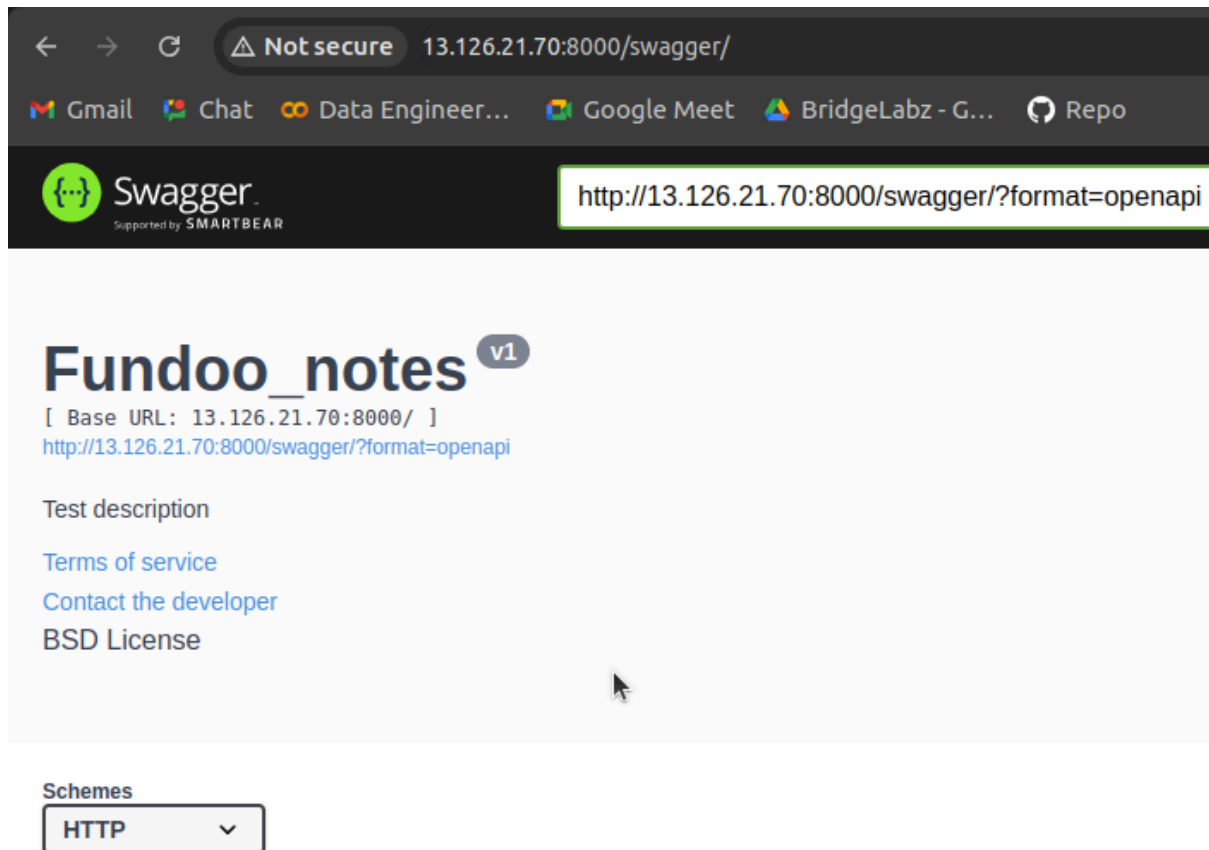
Once the build is complete, verify that your Django application is running correctly by accessing it via its public IP address or domain name.



Welcome, to Fundoo notes ayush !

Perform API testing

We can perform api testing using swagger to confirm our applications is running perfectly



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.

