Day 23 Task: Jenkins Freestyle Project for DevOps Engineers:

What is CI/CD?

- CI or Continuous Integration is the practice of automating the integration of code changes from multiple developers into a single codebase. It is a software development practice where the developers commit their work frequently into the central code repository (Github or Stash). Then there are automated tools that build the newly committed code and do a code review, etc as required upon integration. The key goals of Continuous Integration are to find and address bugs quicker, make the process of integrating code across a team of developers easier, improve software quality and reduce the time it takes to release new feature updates.
- CD or Continuous Delivery is carried out after Continuous Integration to make sure that we can release new changes to our customers quickly in an error-free way. This includes running integration and regression tests in the staging area (similar to the production environment) so that the final release is not broken in production. It ensures to automate the release process so that we have a release-ready product at all times and we can deploy our application at any point in time.

What Is a Build Job?

A Jenkins build job contains the configuration for automating a specific task or step in the application building process. These tasks include gathering dependencies, compiling, archiving, or transforming code, and testing and deploying code in different environments.

Jenkins supports several types of build jobs, such as freestyle projects, pipelines, multi-configuration projects, folders, multibranch pipelines, and organization folders.

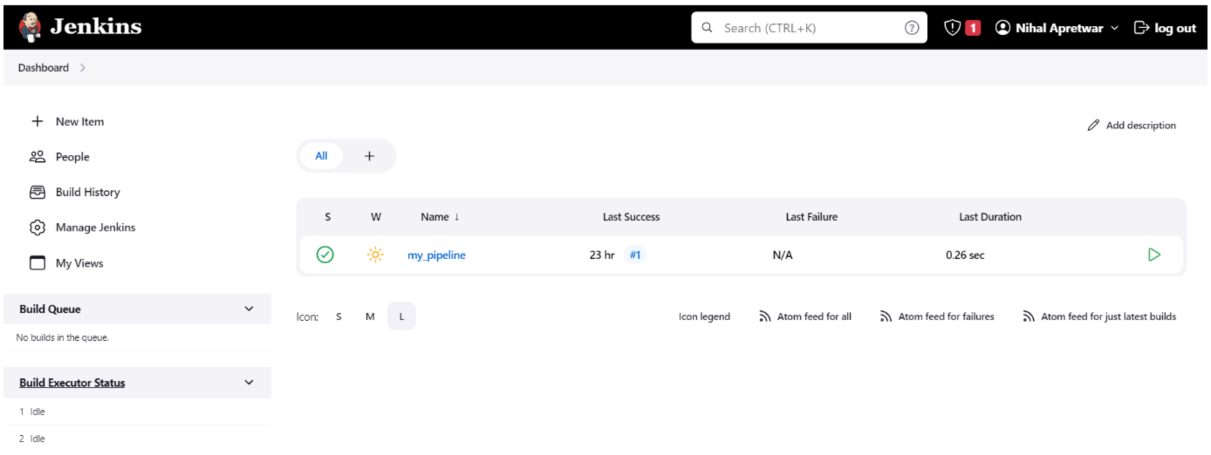
What is Freestyle Projects?

A freestyle project in Jenkins is a type of project that allows you to build, test, and deploy software using a variety of different options and configurations. Here are a few tasks that you could complete when working with a freestyle project in Jenkins.

Task-01:

Create a new Jenkins freestyle project for your app.

- Log in to your Jenkins instance and click on "New Item."


Jenkins — Search (CTRL+K) — Nihal Apretwar — log out

Dashboard

New Item · People · Build History · Manage Jenkins · My Views

Build Queue: No builds in the queue.

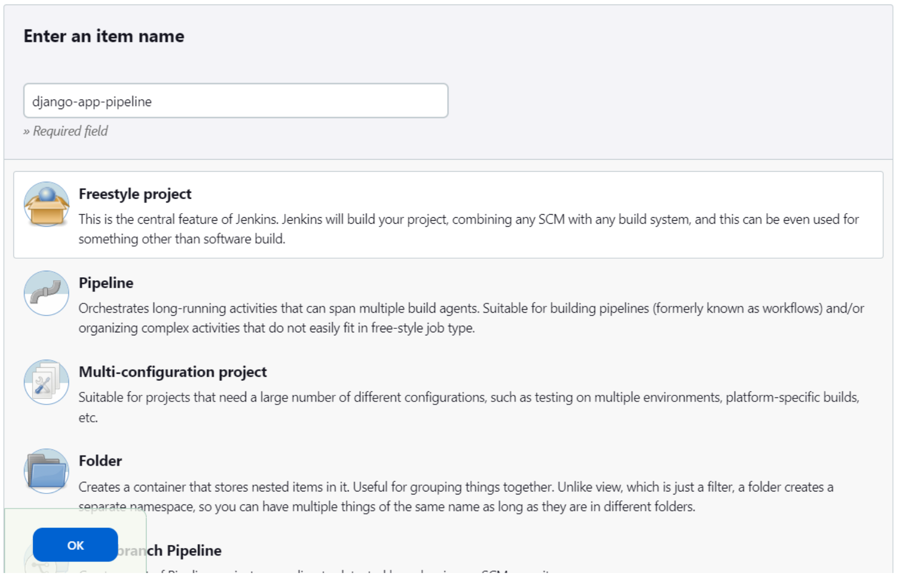
Build Executor Status: 1 Idle, 2 Idle

Add description

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	my_pipeline	23 hr #1	N/A	0.26 sec

Icon: S M L · Icon legend · Atom feed for all · Atom feed for failures · Atom feed for just latest builds

- Choose "Freestyle project" as the type of project.
- Provide a name for your project and click "OK."


Enter an item name

django-app-pipeline

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

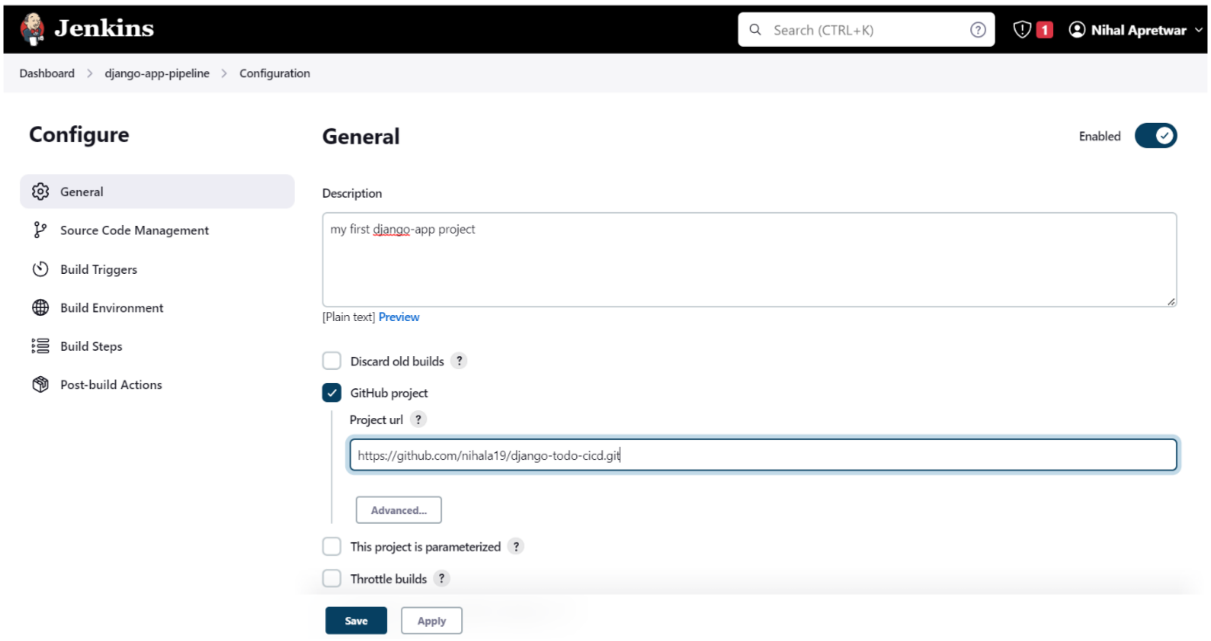
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK ranch Pipeline

- In the project configuration page, you can specify the details of the project, such as the source code management system, build triggers, and build actions. In GitHub project write your GitHub project repository URL.


Jenkins — Search (CTRL+K) — Nihal Apretwar

Dashboard > django-app-pipeline > Configuration

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

General — Enabled

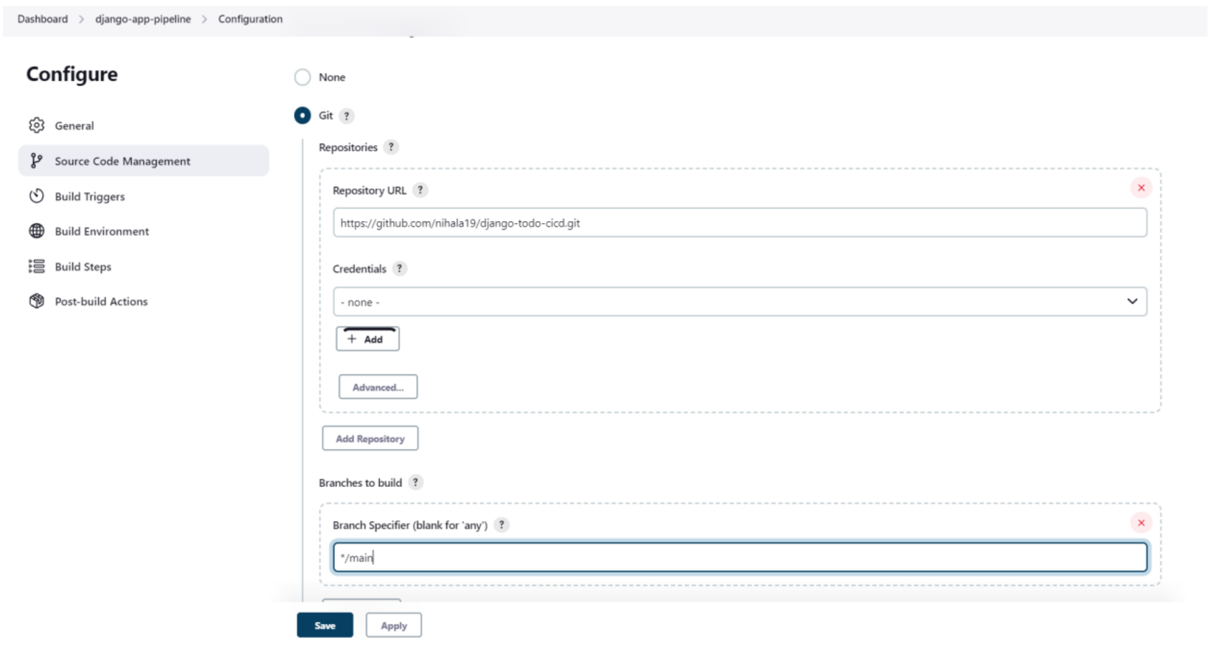
Description: my first django-app project

[Plain text] Preview

- [ ] Discard old builds
- [x] GitHub project
  - Project url: https://github.com/nihala19/django-todo-cicd.git
  - Advanced...
- [ ] This project is parameterized
- [ ] Throttle builds

Save · Apply

- In the "Source Code Management" section, you can specify the repository location, such as Git and Select branch name.



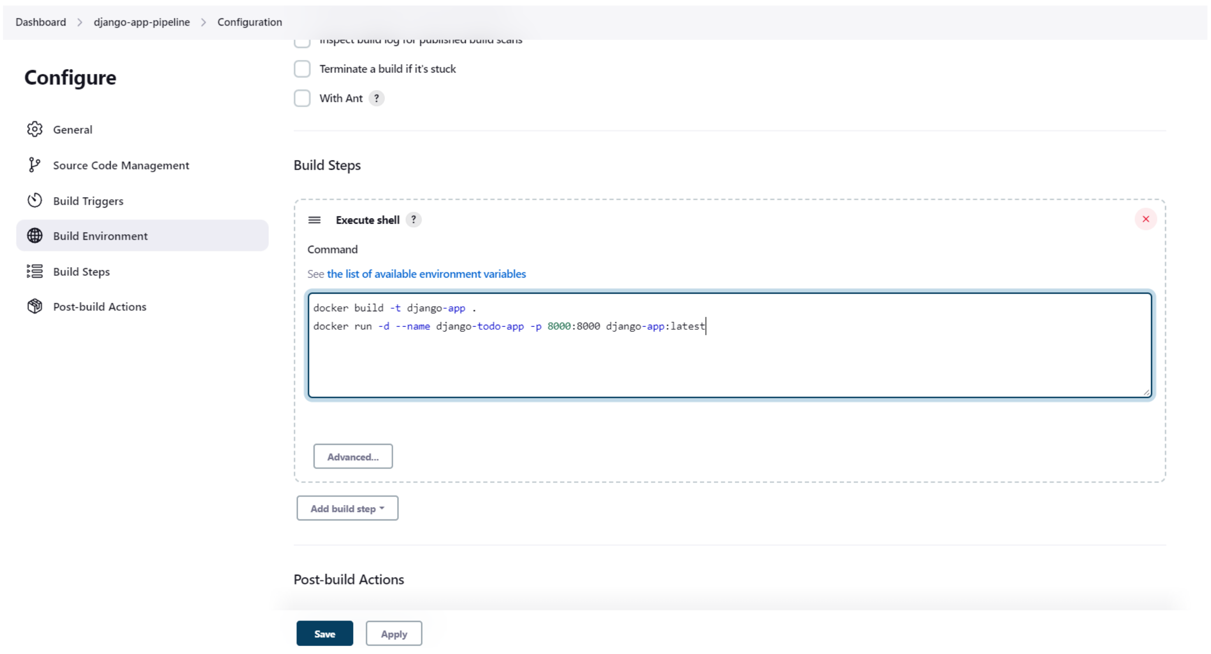
Repository URL: https://github.com/nihala19/django-todo-cicd.git

Credentials: - none -

Branch Specifier (blank for 'any'): */main

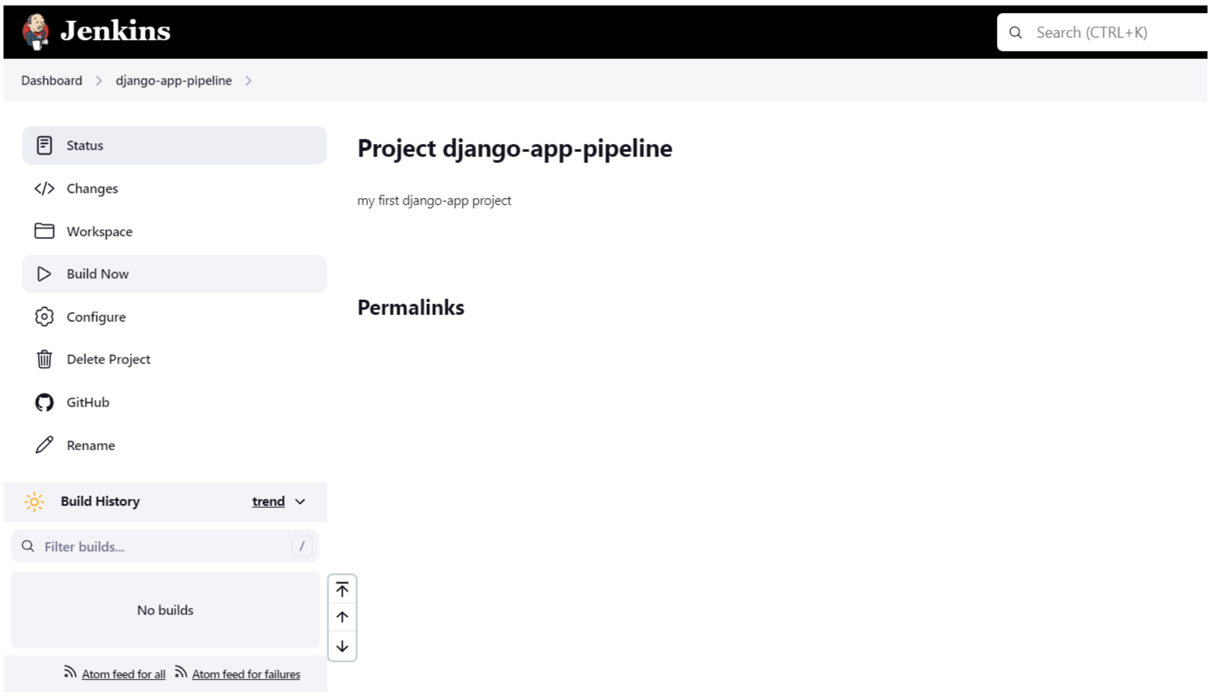
In the "Build" section of the project, add a build step to run the "docker build" command to build the image for the container.

Add a second step to run the "docker run" command to start a container using the image created in step 3.



```
docker build -t django-app .
docker run -d --name django-todo-app -p 8000:8000 django-app:latest
```

- Once you have completed these steps, you can save and build the project.

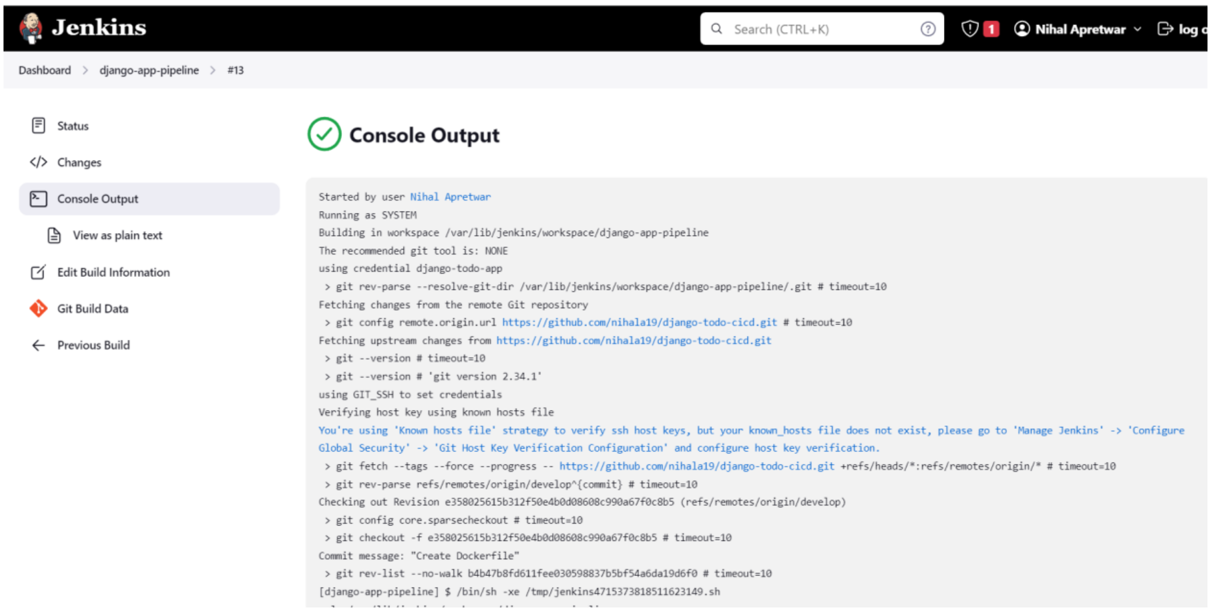


Project django-app-pipeline

my first django-app project

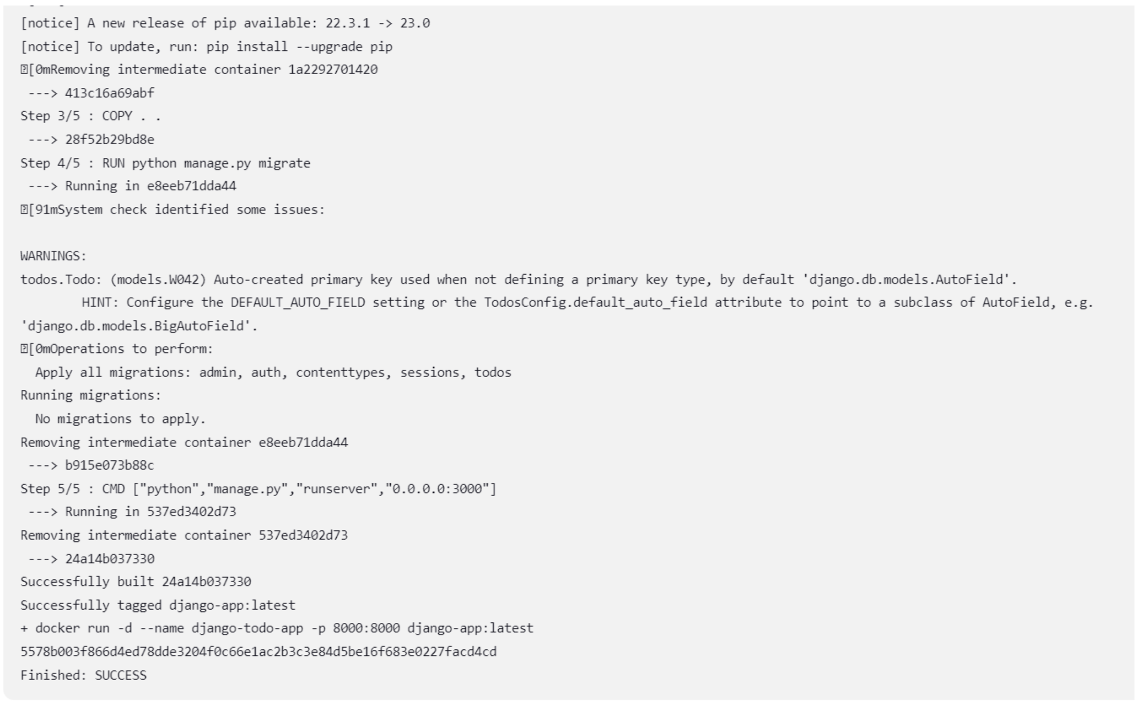
Permalinks

- After a build is completed, you can view the console output by clicking on the "Console Output" link in the build page.



Console Output

```
Started by user Nihal Apretwar
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/django-app-pipeline
The recommended git tool is: NONE
using credential django-todo-app
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/django-app-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/nihala19/django-todo-cicd.git # timeout=10
Fetching upstream changes from https://github.com/nihala19/django-todo-cicd.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
using GIT_SSH to set credentials
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist, please go to 'Manage Jenkins' -> 'Configure
Global Security' -> 'Git Host Key Verification Configuration' and configure host key verification.
 > git fetch --tags --force --progress -- https://github.com/nihala19/django-todo-cicd.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/develop^{commit} # timeout=10
Checking out Revision e358025615b312f50e4b0d08608c990a67f0c8b5 (refs/remotes/origin/develop)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f e358025615b312f50e4b0d08608c990a67f0c8b5 # timeout=10
Commit message: "Create Dockerfile"
 > git rev-list --no-walk b4b47b8fd611fee030598837b5bf54a6da19d6f0 # timeout=10
[django-app-pipeline] $ /bin/sh -xe /tmp/jenkins4715373818511623149.sh
```



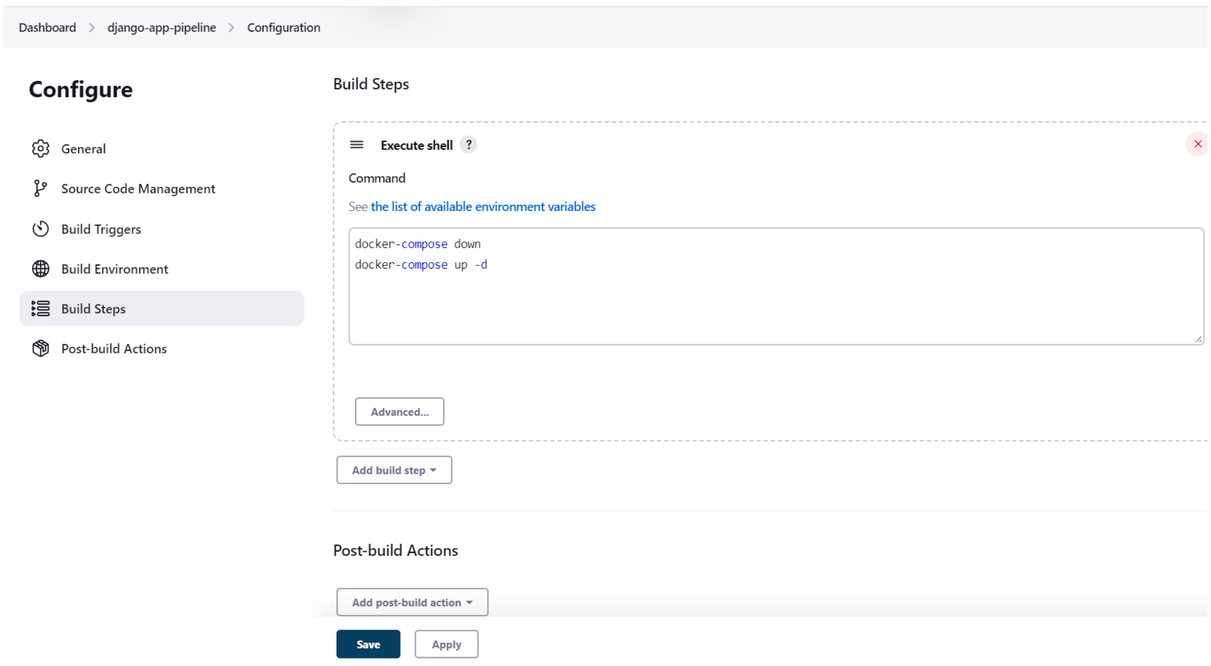
```
[notice] A new release of pip available: 22.3.1 -> 23.0
[notice] To update, run: pip install --upgrade pip
[0mRemoving intermediate container 1a2292701420
 ---> 413c16a69abf
Step 3/5 : COPY . .
 ---> 28f52b29bd8e
Step 4/5 : RUN python manage.py migrate
 ---> Running in e8eeb71dda44
[91mSystem check identified some issues:

WARNINGS:
todos.Todo: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the TodosConfig.default_auto_field attribute to point to a subclass of AutoField, e.g.
'django.db.models.BigAutoField'.
[0mOperations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, todos
Running migrations:
  No migrations to apply.
Removing intermediate container e8eeb71dda44
 ---> b915e073b88c
Step 5/5 : CMD ["python","manage.py","runserver","0.0.0.0:3000"]
 ---> Running in 537ed3402d73
Removing intermediate container 537ed3402d73
 ---> 24a14b037330
Successfully built 24a14b037330
Successfully tagged django-app:latest
+ docker run -d --name django-todo-app -p 8000:8000 django-app:latest
5578b003f866d4ed78dde3204f0c66e1ac2b3c3e84d5be16f683e0227facd4cd
Finished: SUCCESS
```

Task-02

Create Jenkins project to run "docker-compose up -d" command to start the multiple containers defined in the compose file (Hint- use day-19 Application & Database docker-compose file)

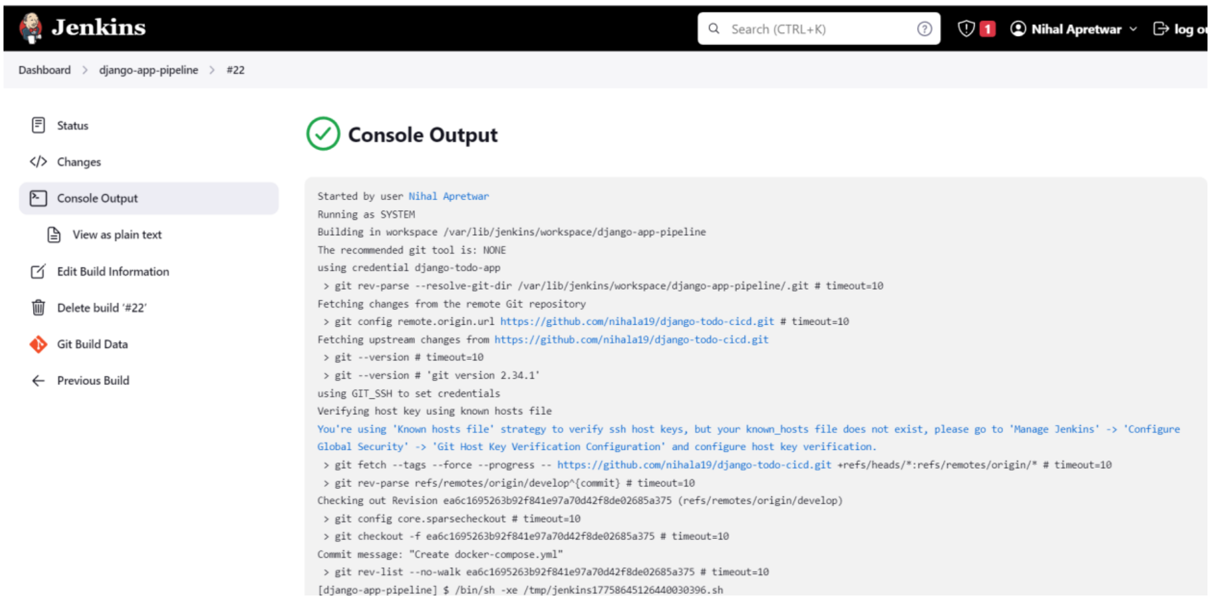
Set up a cleanup step in the Jenkins project to run "docker-compose down" command to stop and remove the containers defined in the compose file.

- In the "Build" section of the project, add a build step "docker-compose down" command to stop and remove the containers defined in the compose file. then add "docker-compose up -d" command.

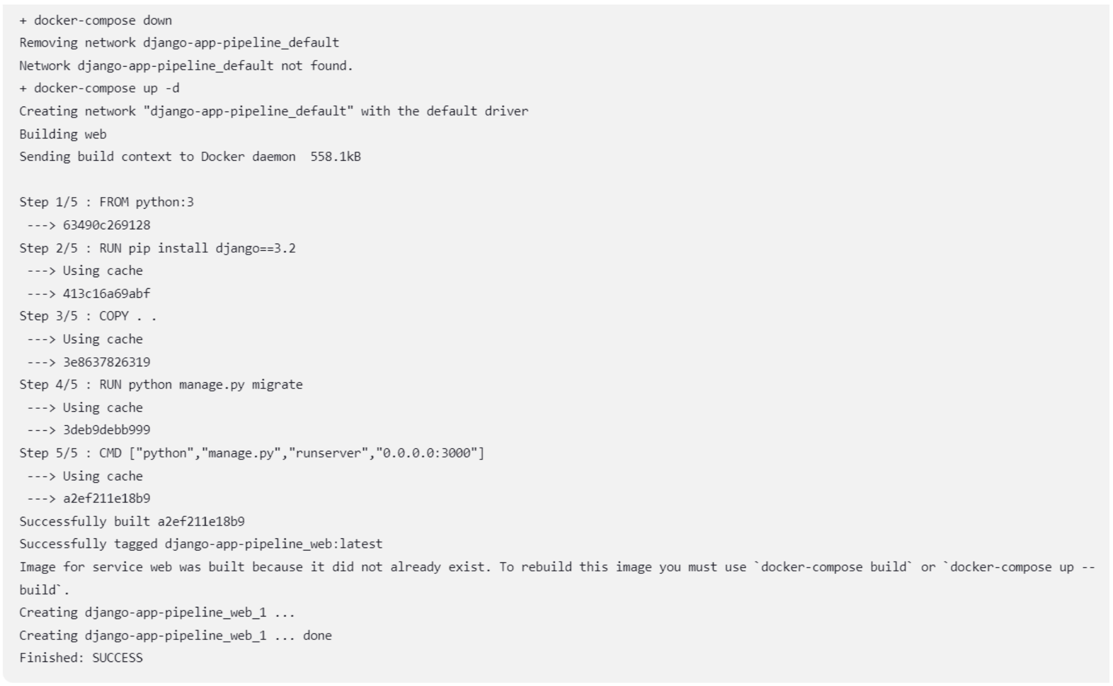


```
docker-compose down
docker-compose up -d
```

- Build the project.
- After a build is completed, you can view the console output.

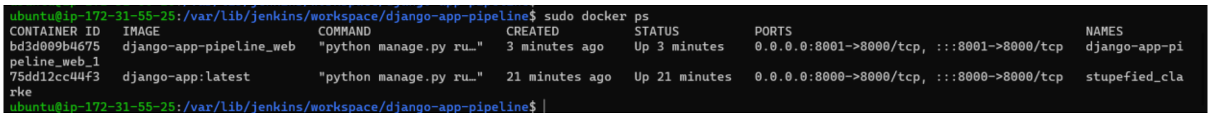


```
Started by user Nihal Apretwar
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/django-app-pipeline
The recommended git tool is: NONE
using credential django-todo-app
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/django-app-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/nihala19/django-todo-cicd.git # timeout=10
Fetching upstream changes from https://github.com/nihala19/django-todo-cicd.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
using GIT_SSH to set credentials
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist, please go to 'Manage Jenkins' -> 'Configure
Global Security' -> 'Git Host Key Verification Configuration' and configure host key verification.
 > git fetch --tags --force --progress -- https://github.com/nihala19/django-todo-cicd.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/develop^{commit} # timeout=10
Checking out Revision ea6c1695263b92f841e97a70d42f8de02685a375 (refs/remotes/origin/develop)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f ea6c1695263b92f841e97a70d42f8de02685a375 # timeout=10
Commit message: "Create docker-compose.yml"
 > git rev-list --no-walk ea6c1695263b92f841e97a70d42f8de02685a375 # timeout=10
[django-app-pipeline] $ /bin/sh -xe /tmp/jenkins17758645126440030396.sh
```



```
+ docker-compose down
Removing network django-app-pipeline_default
Network django-app-pipeline_default not found.
+ docker-compose up -d
Creating network "django-app-pipeline_default" with the default driver
Building web
Sending build context to Docker daemon  558.1kB

Step 1/5 : FROM python:3
 ---> 63490c269128
Step 2/5 : RUN pip install django==3.2
 ---> Using cache
 ---> 413c16a69abf
Step 3/5 : COPY . .
 ---> Using cache
 ---> 3e8637826319
Step 4/5 : RUN python manage.py migrate
 ---> Using cache
 ---> 3deb9debb999
Step 5/5 : CMD ["python","manage.py","runserver","0.0.0.0:3000"]
 ---> Using cache
 ---> a2ef211e18b9
Successfully built a2ef211e18b9
Successfully tagged django-app-pipeline_web:latest
Image for service web was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compose up --
build`.
Creating django-app-pipeline_web_1 ...
Creating django-app-pipeline_web_1 ... done
Finished: SUCCESS
```

- You can see container is created.



```
ubuntu@ip-172-31-55-25:/var/lib/jenkins/workspace/django-app-pipeline$ sudo docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS          PORTS                                       NAMES
bd3d009b4675   django-app-pipeline_web   "python manage.py ru…"   3 minutes ago    Up 3 minutes    0.0.0.0:8001->8000/tcp, :::8001->8000/tcp   django-app-pi
peline_web_1
75dd12cc44f3   django-app:latest         "python manage.py ru…"   21 minutes ago   Up 21 minutes   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   stupefied_cla
rke
ubuntu@ip-172-31-55-25:/var/lib/jenkins/workspace/django-app-pipeline$ |
```

Thank you for reading!

😊😊