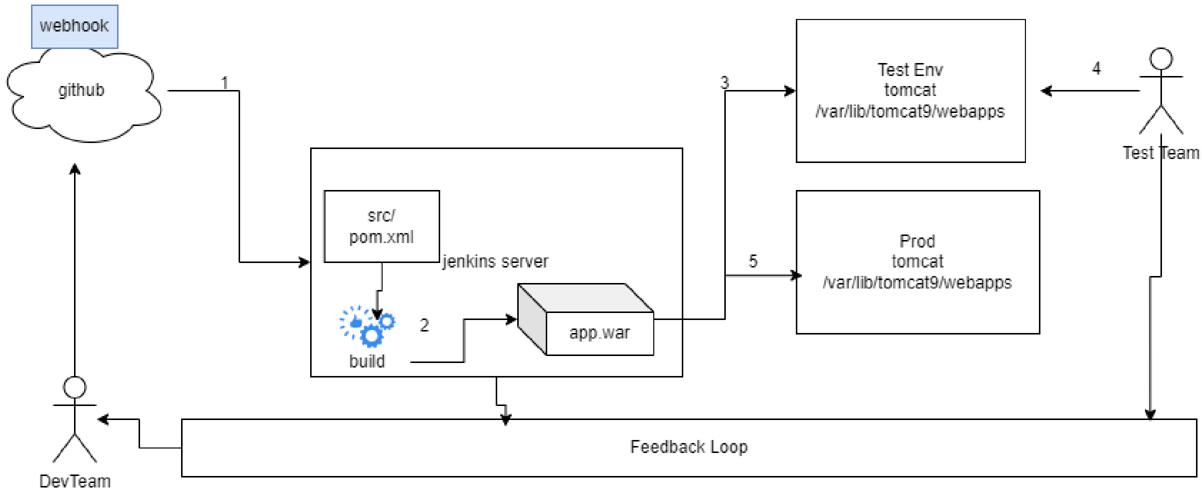
DevOps CICD Project 1 with tools
[Git - github - jenkins - maven - tomcat - AWS Cloud]



1. Cont download
2. Cont build
3. Cont deployment
4. Cont test
5. Cont delivery

Infrastructure Creation:
==================

1. Create Jenkins server
   t2.micro - ubuntu 22.04 and ports open 22 and 8080
   Included the below shell script in user data section

```
#!/bin/bash
sudo apt update
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]"
https://pkg.jenkins.io/debian-stable binary/ | sudo tee
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install fontconfig openjdk-17-jre -y
sudo apt-get install jenkins -y
sudo apt-get install -y maven
```

2. Create QA and Prod Servers
   t2.micro - ubuntu 22.04 and ports open 22 and 8080
   Included the below shell script in user data section

```
#!/bin/bash
sudo apt-get update
sudo apt-get install -y tomcat9 tomcat9-admin
```

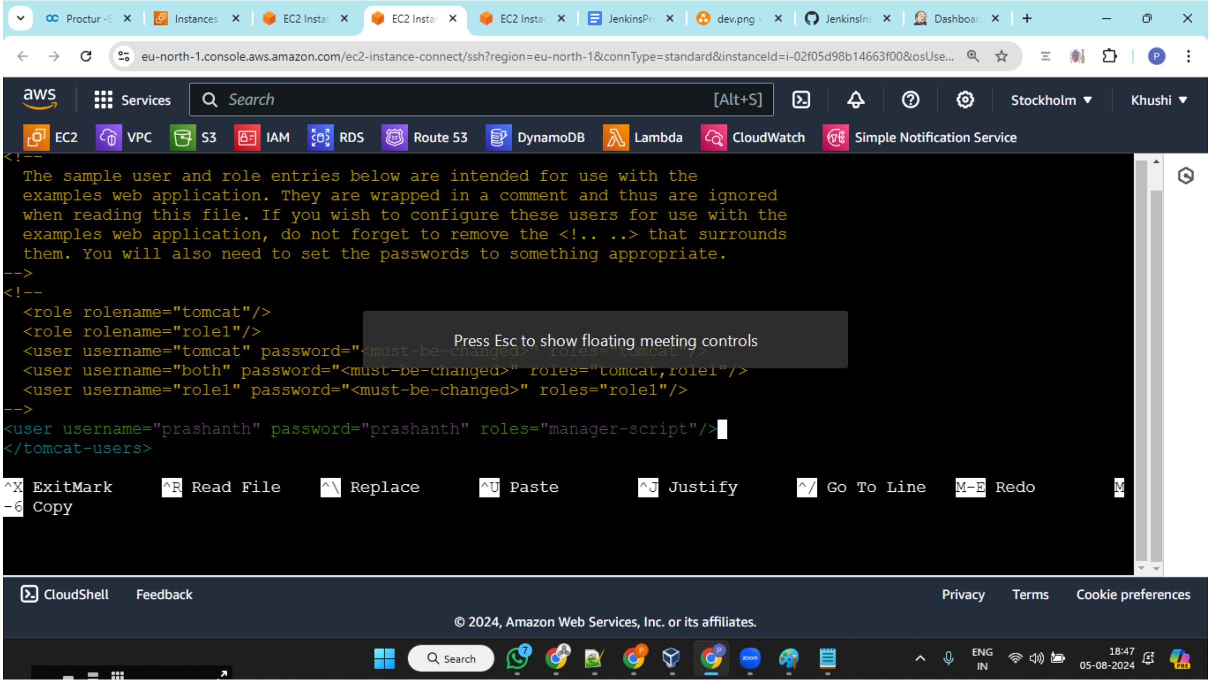
[!Optional]
To change the hostnames of servers
Jenkins
sudo hostnamectl set-hostname jenkins
exec bash

QAserver
sudo hostnamectl set-hostname qaserver
exec bash

ProdServer
sudo hostnamectl set-hostname prodserver
exec bash

Connect to Jenkins server and unlock jenkins
Install suggested plugins
Create a first admin user
Set up Jenkins and reach to the Dashboard of Jenkins

QAServer:
Edit the file tomcat-users.xml and create a user in tomcat
sudo nano /etc/tomcat9/tomcat-users.xml



```
<!--
  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
-->
<user username="prashanth" password="prashanth" roles="manager-script"/>
</tomcat-users>

^X ExitMark    ^R Read File    ^\ Replace    ^U Paste    ^J Justify    ^/ Go To Line    M-E Redo    M
-6 Copy
```

Press Esc to show floating meeting controls

eu-north-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=eu-north-1&connType=standard&instanceId=i-02f05d98b14663f00&osUse...
Services
Search
[Alt+S]
Stockholm
Khushi
EC2
VPC
S3
IAM
RDS
Route 53
DynamoDB
Lambda
CloudWatch
Simple Notification Service
CloudShell
Feedback
Privacy
Terms
Cookie preferences
© 2024, Amazon Web Services, Inc. or its affiliates.
18:47
05-08-2024

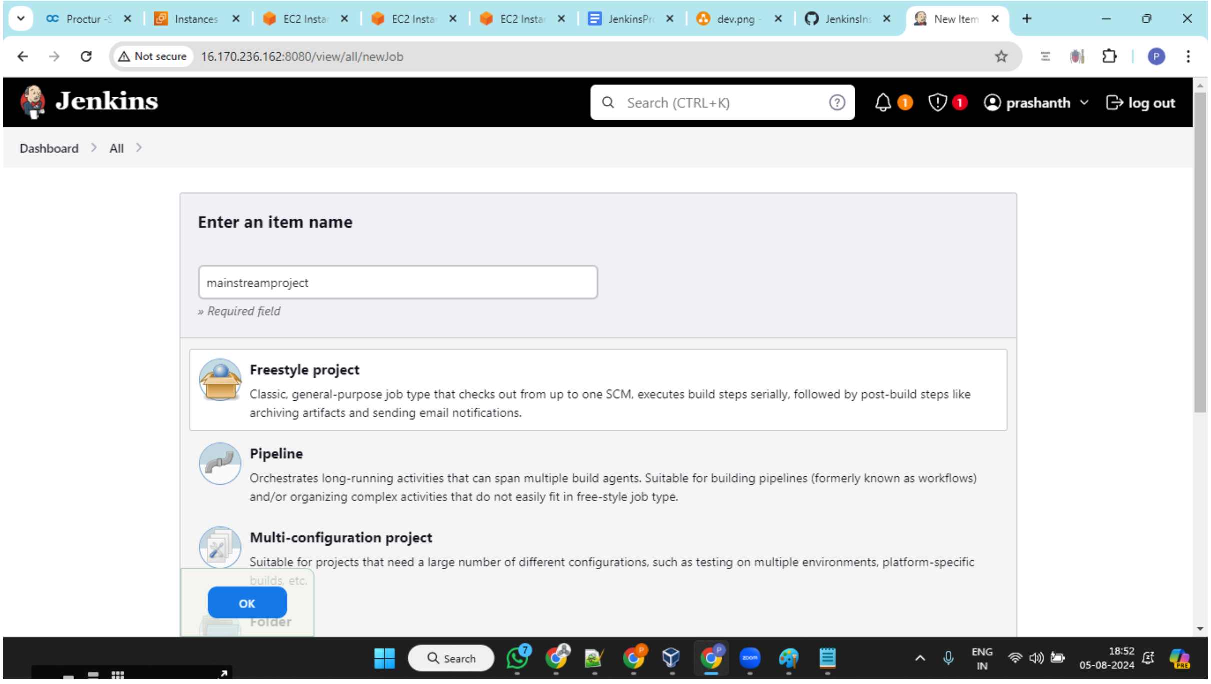
After adding the user, save the file and restart tomcat
sudo systemctl restart tomcat9

ProdServer:
Create a tomcat user as you did for the previous qa server

Connect to Jenkins and install the below plugins

1. Deploy to container plugin
2. Copy artifact plugin

Creating the project:



16.170.236.162:8080/view/all/newJob

Enter an item name

mainstreamproject

» Required field

Freestyle project

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

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

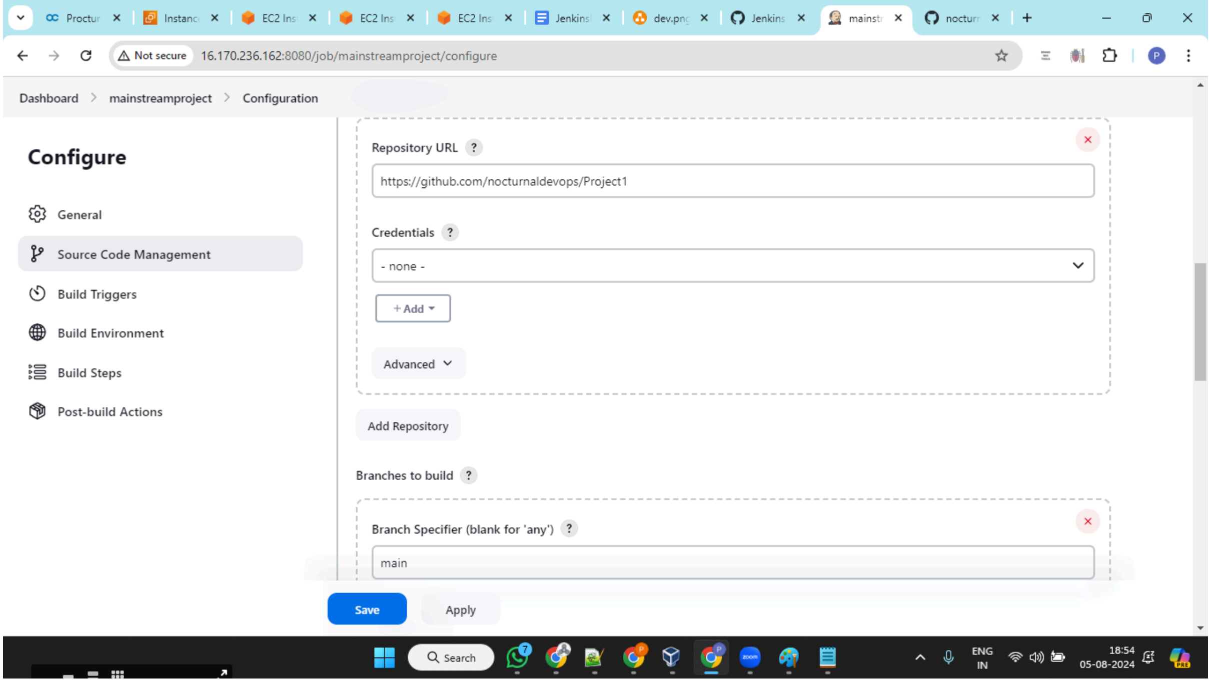
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

In the project configurations

General section:

Add proper documentation

Get the project from github to jenkins



16.170.236.162:8080/job/mainstreamproject/configure

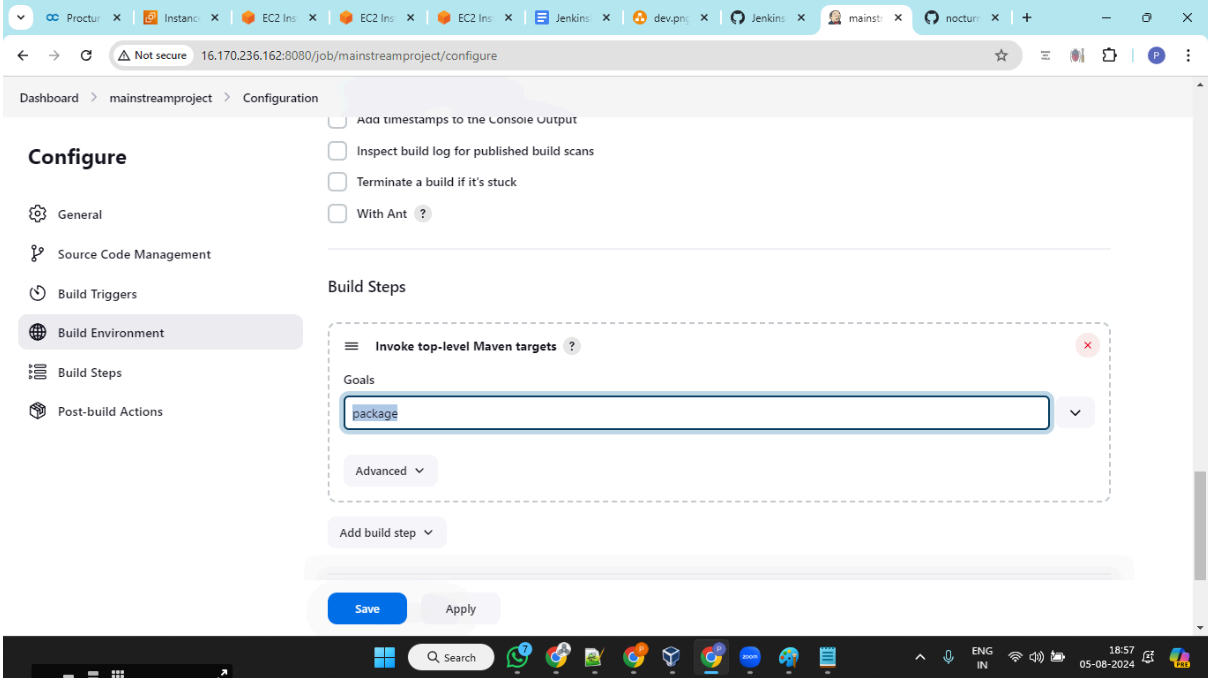
Repository URL: https://github.com/nocturnaldevops/Project1

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): main

Add “package” in the build step to perform build activity


16.170.236.162:8080/job/mainstreamproject/configure

Dashboard > mainstreamproject > Configuration

Configure

□ Add timestamps to the Console Output
□ Inspect build log for published build scans
□ Terminate a build if it's stuck
□ With Ant

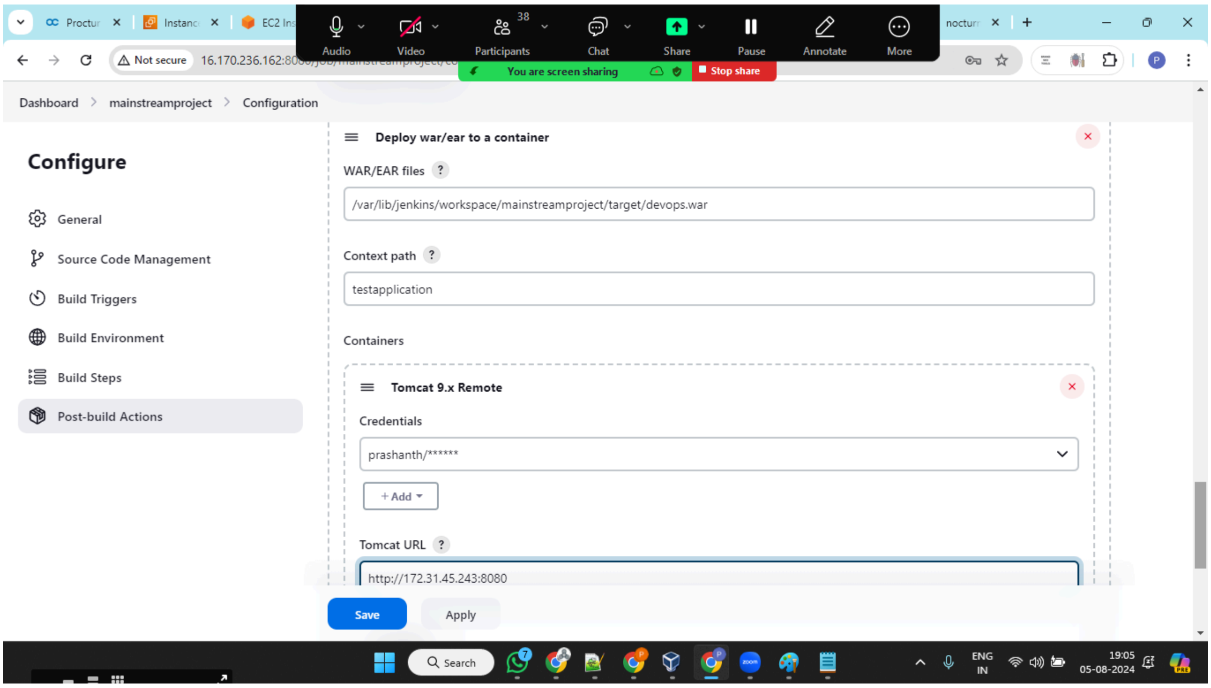
Build Steps

Invoke top-level Maven targets

Goals: package

When you build the project, an artifact gets created in the project workspace
Now copy the artifact to test environment

In Project configurations
Jump to post build actions
Select the option (deploy war/ear to container) and furnish the details


Dashboard > mainstreamproject > Configuration

Configure

Deploy war/ear to a container

WAR/EAR files: /var/lib/jenkins/workspace/mainstreamproject/target/devops.war

Context path: testapplication

Containers

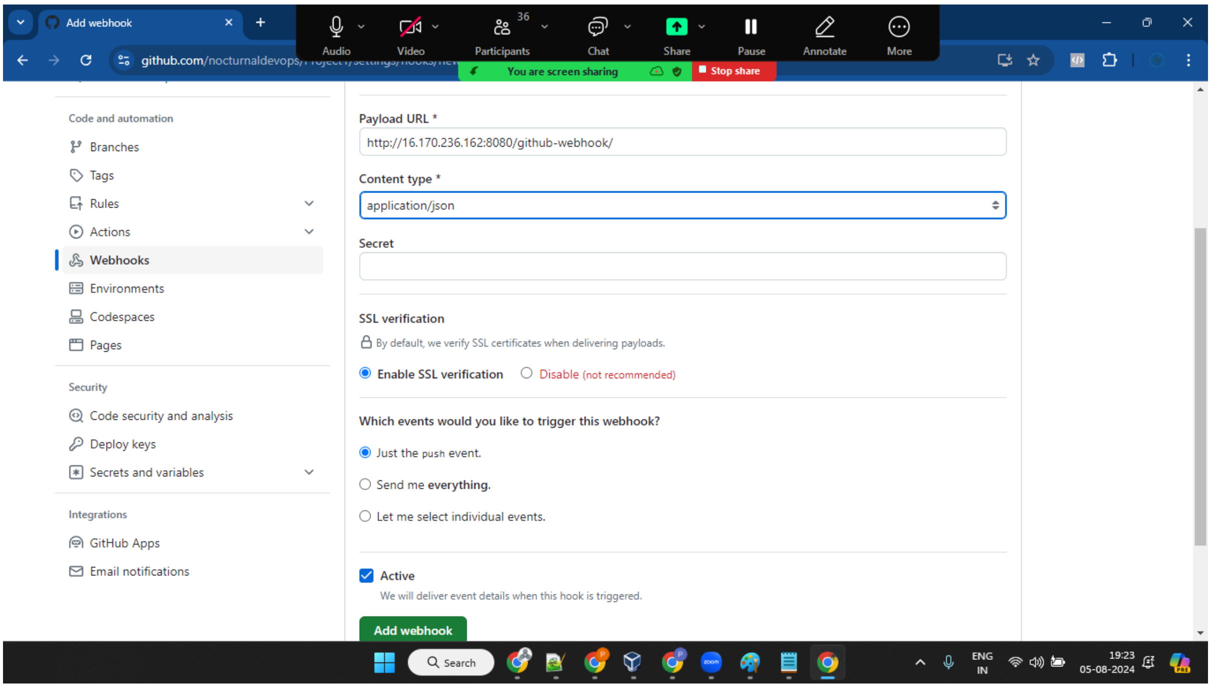
Tomcat 9.x Remote

Credentials: prashanth/******

Tomcat URL: http://172.31.45.243:8080

!!war/ear files - please make sure you add **/*.war (dont include a complete path)

Adding webhook to github



Payload URL *
http://16.170.236.162:8080/github-webhook/

Content type *
application/json

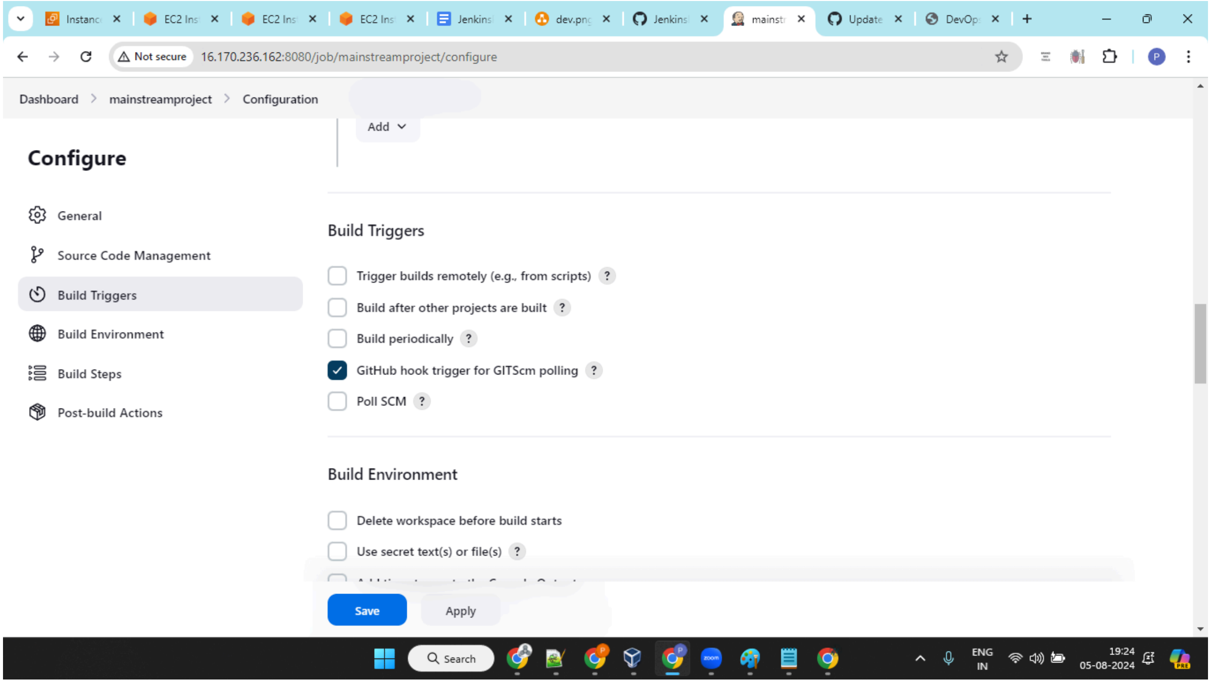
Secret

SSL verification
By default, we verify SSL certificates when delivering payloads.
● Enable SSL verification ○ Disable (not recommended)

Which events would you like to trigger this webhook?
● Just the push event.
○ Send me everything.
○ Let me select individual events.

☑ Active
We will deliver event details when this hook is triggered.

Activate the webhook by enabling in the project at jenkins level



Build Triggers

□ Trigger builds remotely (e.g., from scripts) ?
□ Build after other projects are built ?
□ Build periodically ?
☑ GitHub hook trigger for GITScm polling ?
□ Poll SCM ?

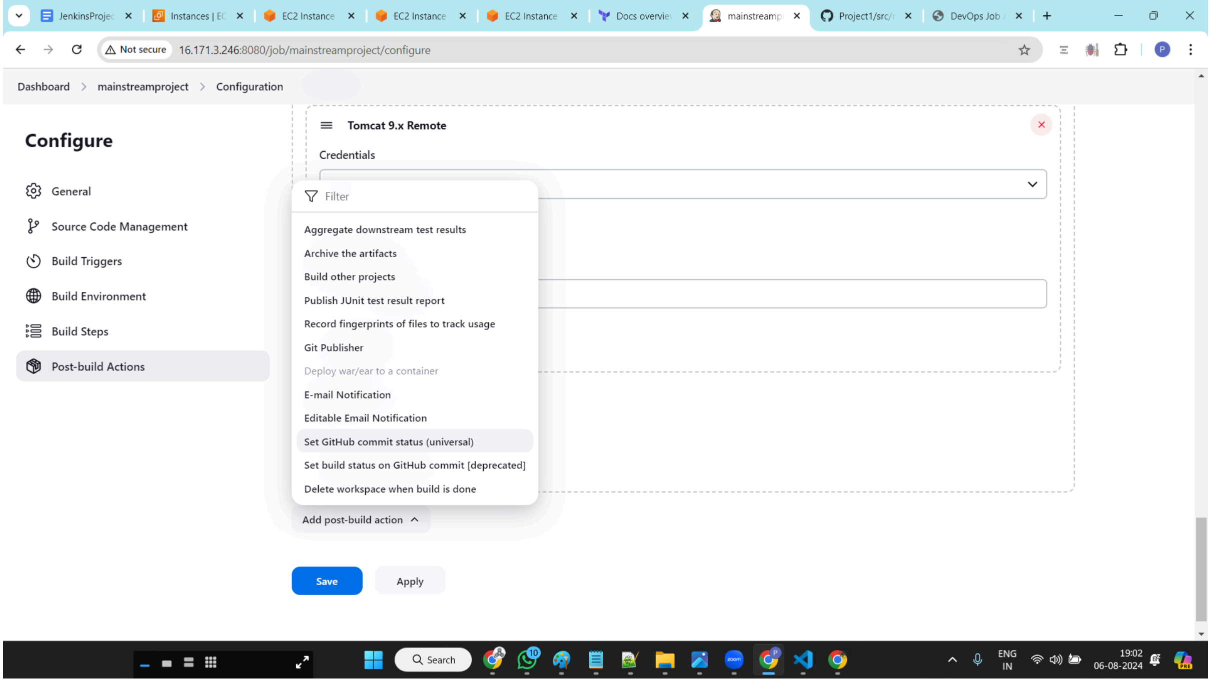
Build Environment

□ Delete workspace before build starts
□ Use secret text(s) or file(s) ?

The application gets deployed to test environment as soon as a developer commits a change.

Manual testing

Let’s assume the application tested by the test team is “approved” and should be delivered to the prod environment, what steps do we need to follow?



Dashboard > mainstreamproject > Configuration

Configure

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

Tomcat 9.x Remote

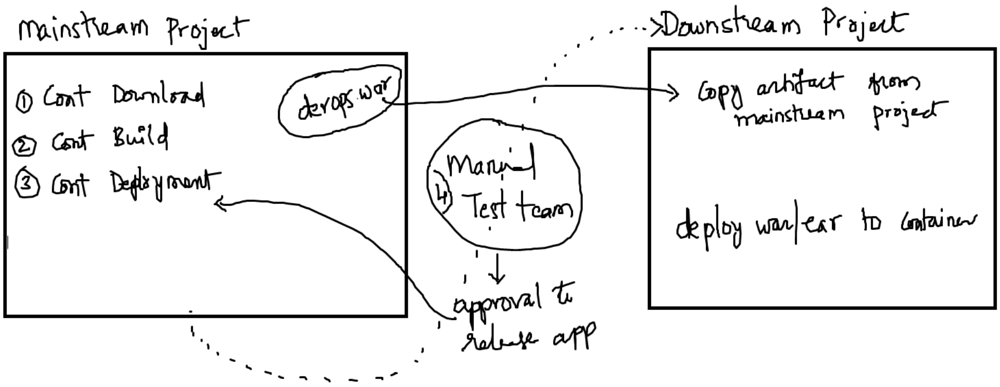
Credentials

Filter

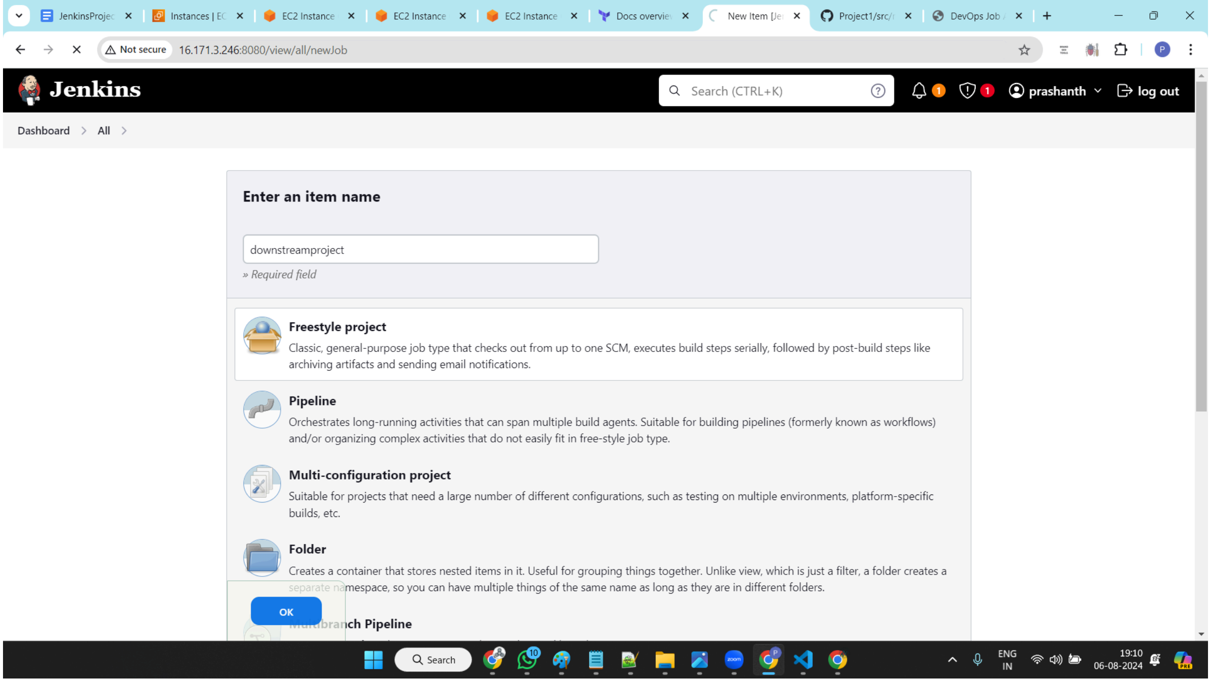
- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Record fingerprints of files to track usage
- Git Publisher
- Deploy war/ear to a container
- E-mail Notification
- Editable Email Notification
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Delete workspace when build is done

Add post-build action

Post build actions → *deploy war/ear to container* option is ***disabled*** in order to deliver the application to prod environment



Create a new Downstream Project



Enter an item name

downstreamproject

» Required field

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

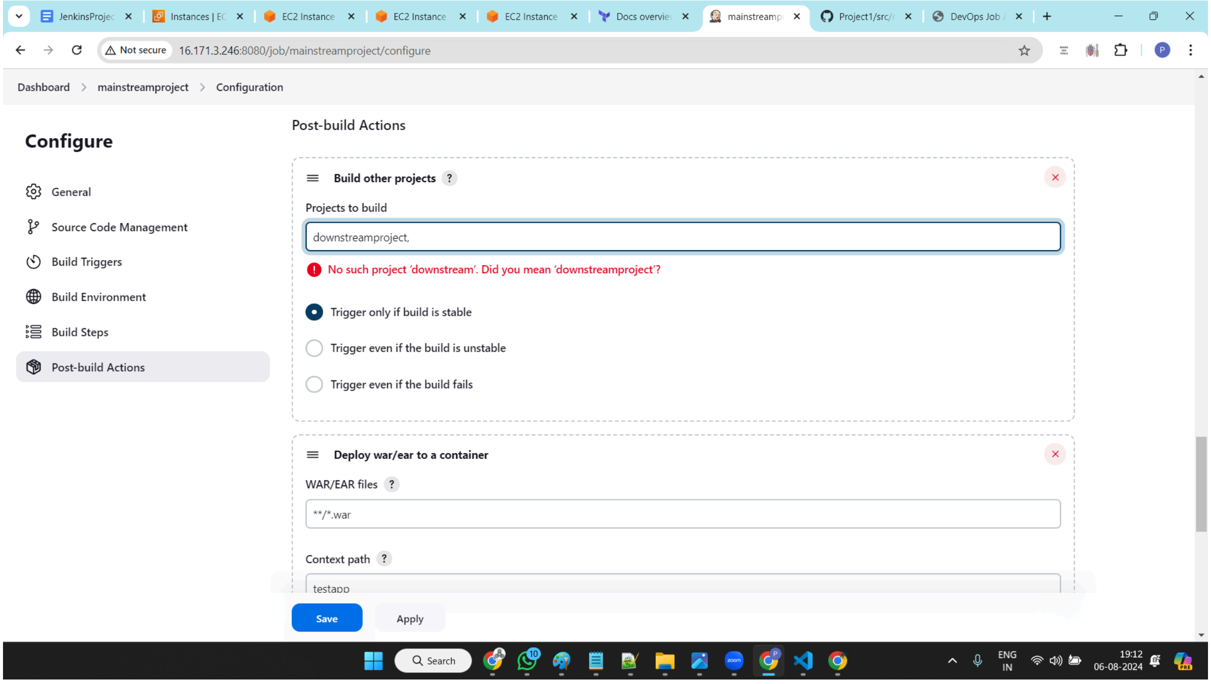
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

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.

Multibranch Pipeline

Create a downstream project without any configuration
Now switch to the mainstream project configuration and
Post build actions —> build other projects —> and provide downstreamproject name



Post-build Actions

Build other projects

Projects to build: downstreamproject,

No such project 'downstream'. Did you mean 'downstreamproject'?

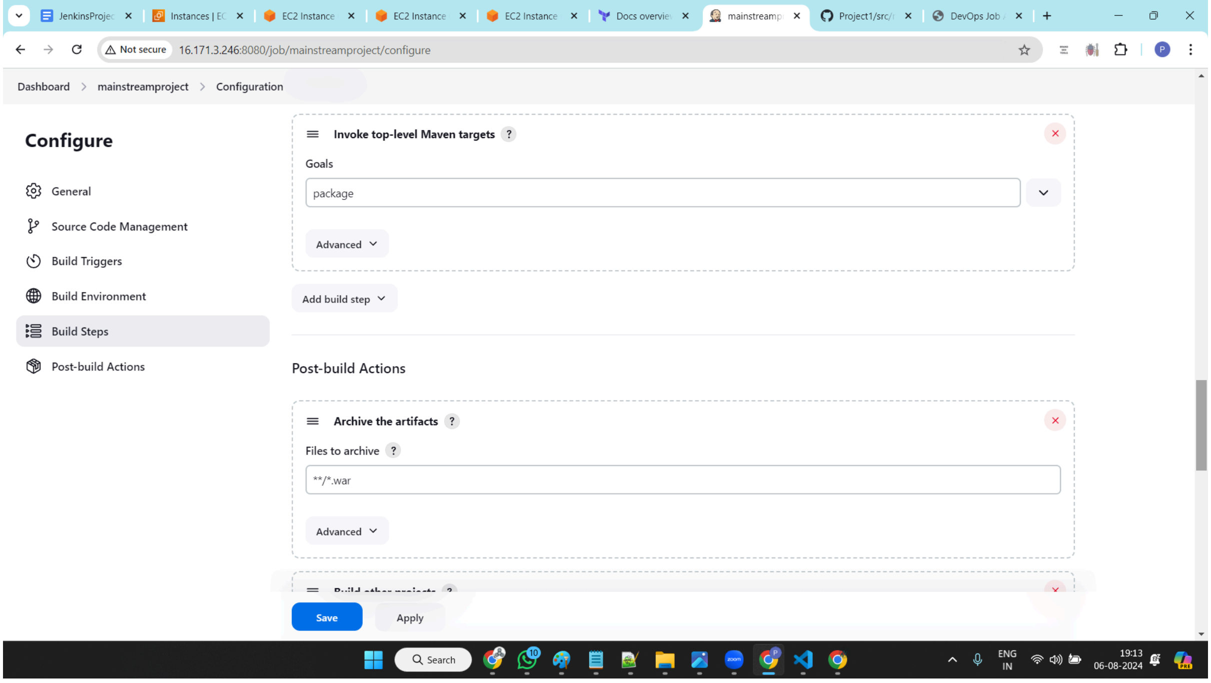
● Trigger only if build is stable
○ Trigger even if the build is unstable
○ Trigger even if the build fails

Deploy war/ear to a container

WAR/EAR files: **/*.war

Context path: testapp

In mainstream project configuration
Post build actions → archive the artifacts —> **/*.war


16.171.3.246:8080/job/mainstreamproject/configure

Dashboard > mainstreamproject > Configuration

Configure

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

Invoke top-level Maven targets ?

Goals: package

Advanced

Add build step

Post-build Actions

Archive the artifacts ?

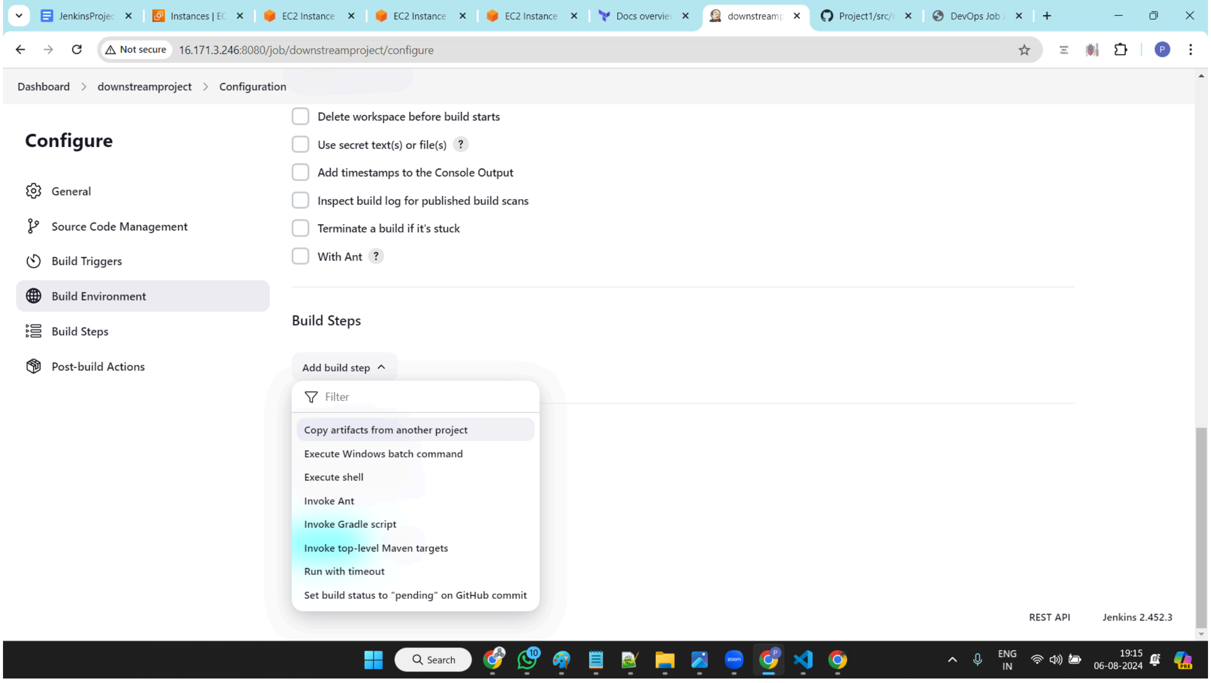
Files to archive ?: **/*.war

Advanced

Build other projects ?

Save Apply

Now to the downstream project configuration,
Select add build step → copy artifact from other project


16.171.3.246:8080/job/downstreamproject/configure

Dashboard > downstreamproject > Configuration

Configure

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

☐ Delete workspace before build starts
☐ 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 ?

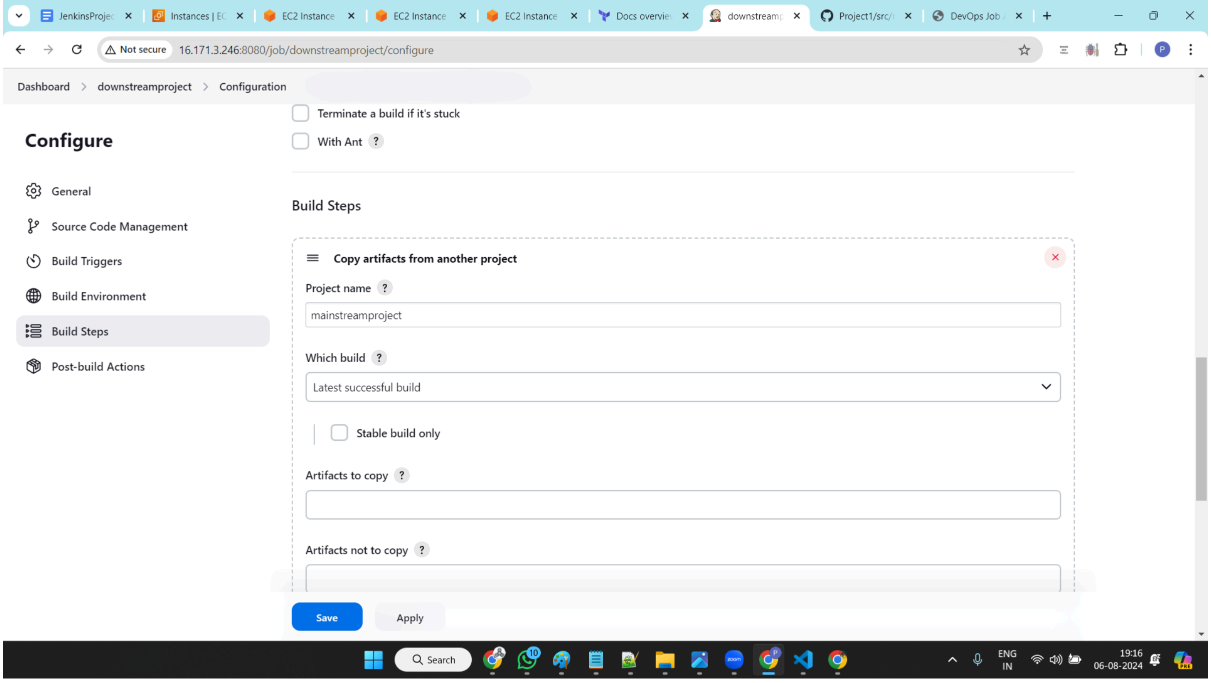
Build Steps

Add build step

Filter
- Copy artifacts from another project
- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit

REST API Jenkins 2.452.3

Type mainstreamproject in project name



□ Terminate a build if it's stuck

□ With Ant

Build Steps

Copy artifacts from another project

Project name: mainstreamproject

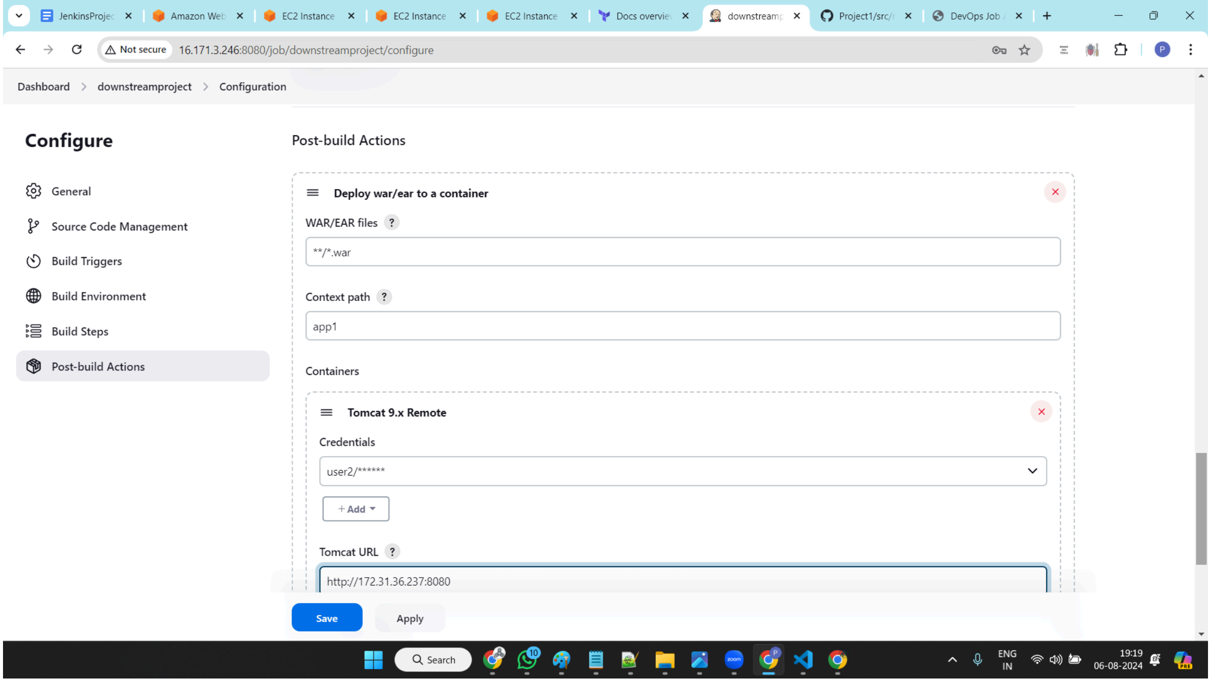
Which build: Latest successful build

□ Stable build only

Artifacts to copy:

Artifacts not to copy:

In the downstream project, jump to post build actions and add the details of prod server's tomcat configuration



Post-build Actions

Deploy war/ear to a container

WAR/EAR files: **/*.war

Context path: app1

Containers

Tomcat 9.x Remote

Credentials: user2/******

Tomcat URL: http://172.31.36.237:8080

Always build the mainstream project only!