

Devops Project 1

Pre-requisite for CI/CD pipeline project

1. Github account
2. Simple java project
3. AWS account
4. Jenkins Server(linux)
5. Maven and Git installation
6. Tomcat Server (Linux)

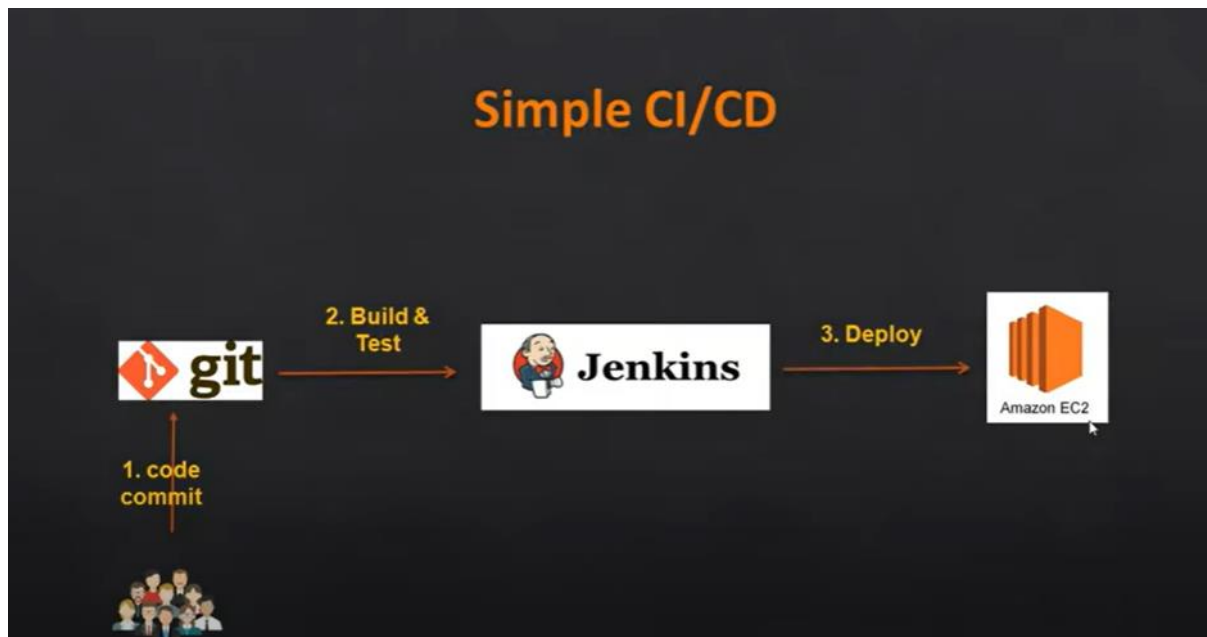
Objective:

To deploy war file on Amazon EC2 instance (tomcat server) using git and Jenkins.

Steps:

1. Developers will commit the changes to the code using git on github account
2. Jenkins will build and test the project for every commit made by developers and create distributable jar or war file.
3. Finally Jenkins will deploy war file on tomcat server residing on ec2 instance.

Project Workflow:



Part-01 : Adding steps for Integration

Steps to create Jenkin job

1. Login to Jenkins console
2. Create *Jenkins job*, Fill the following details,
 - *Source Code Management:*
 - Repository: <https://github.com/ValaxyTech/hello-world.git>
 - Branches to build : */master
 - *Build:*
 - Root POM:pom.xml
 - Goals and options : clean install package

Part-02: Adding Deployment Steps

in this port we are going to install 'deploy to container' plugin. this is need to deploy on tomcat server which we are using.

- Install maven plugin without restart
 - Manage Jenkins > Jenkins Plugins > available > deploy to container

To deploy our build artifacts on tomcat server our Jenkins server need access. For this we should setup credentials. This option is available in Jenkins home page

- setup credentials
 - credentials > jenkins > Global credentials > add credentials
 - Username : deployer
 - Password : XXXXXXXX
 - id : Tomcat_user
 - Description: Tomcat user to deploy on tomcat server

Modify the same job which created in part-01 and add deployment steps.

- Post Steps
 - Deploy war/ear to container
 - WAR/EAR files : **/*.war
 - Containers : Tomcat 8.x
 - Credentials: Tomcat_user (which created in above step)
 - Tomcat URL : http://<PUBLIC_IP>:<PORT_NO>

Save and run the job now.

Part-03 : Continuous Integration & Continuous Deployment (CI/CD)

Now job is running fine but to make this as Continuous Integration and Continuous Deployment Tod do that go back and modify job as below.

- Build Triggers
 - Poll SCM
 - schedule */2 * * * *

Save the job and modify the code in GitHub. Then you could see your job get trigger a build without any manual intervention.

Flow :

Step1: Make a simple maven project and push it on github account

- Pom.xml file is used to build maven project

The screenshot shows the GitHub interface for a repository named 'yadneshingole / CI-CD_Project'. The repository is public and has 1 branch (main) and 0 tags. The file list includes:

File/Folder	Commit Message	Commit Hash	Time
server	Add files via upload	f3a1625	4 days ago
webapp	updated		2 days ago
Dockerfile	Add files via upload		4 days ago
README.md	Add files via upload		4 days ago
pom.xml	Add files via upload		4 days ago
webapp.war	war file added		13 hours ago

Below the file list, there is a section for 'README.md' with a link to the wiki page: https://github.com/yadneshingole/CI-CD_Project/wiki.

Step 2: Start Jenkins Server and access it on localhost port 8080

The screenshot shows the Jenkins Dashboard at localhost:8080. The left sidebar contains navigation links: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Lockable Resources, and New View. The main area displays a table of builds for the 'TestDeliveryPipeline' project.

S	W	Name	Last Success	Last Failure	Last Duration
✗	🔗	AutomatedDeploymentTest	N/A	1 mo 0 days #1	2.1 sec
✓	⚙️	maven-java-freestyle-project	27 days #3	N/A	1 min 42 sec
✓	🔗	maven_hello_world	3 min 55 sec #26	6 min 47 sec #25	13 sec
✓	☁️	Maven_java_pipeline_project	27 days #9	27 days #7	2 min 12 sec
✓	🔗	PetClinicCompile	1 mo 0 days #4	1 mo 0 days #1	19 sec
✓	🔗	PetClinicDeclarativePipeline	1 mo 0 days #4	1 mo 0 days #2	2 min 29 sec

Step 3: Make a simple maven Project on Jenkins

- Go to configure project and add scm of git repository
- Go to build options and add goals such as
- Clean – For Cleaning previous maven project
- install – for installing necessary packages and dependency
- Package – for building jar, war file from java project

The screenshot shows the Jenkins job configuration page for 'maven_hello_world' at localhost:8080/job/maven_hello_world/. The left sidebar includes links: Back to Dashboard, Status, Changes, Workspace, Build Now, Configure, Delete Maven project, Modules, Git Polling Log, and Rename. The main area displays the job configuration details.

Maven project maven_hello_world

Test Result Trend

Legend: Passed (green), Skipped (grey), Failed (red)

Workspace

Recent Changes

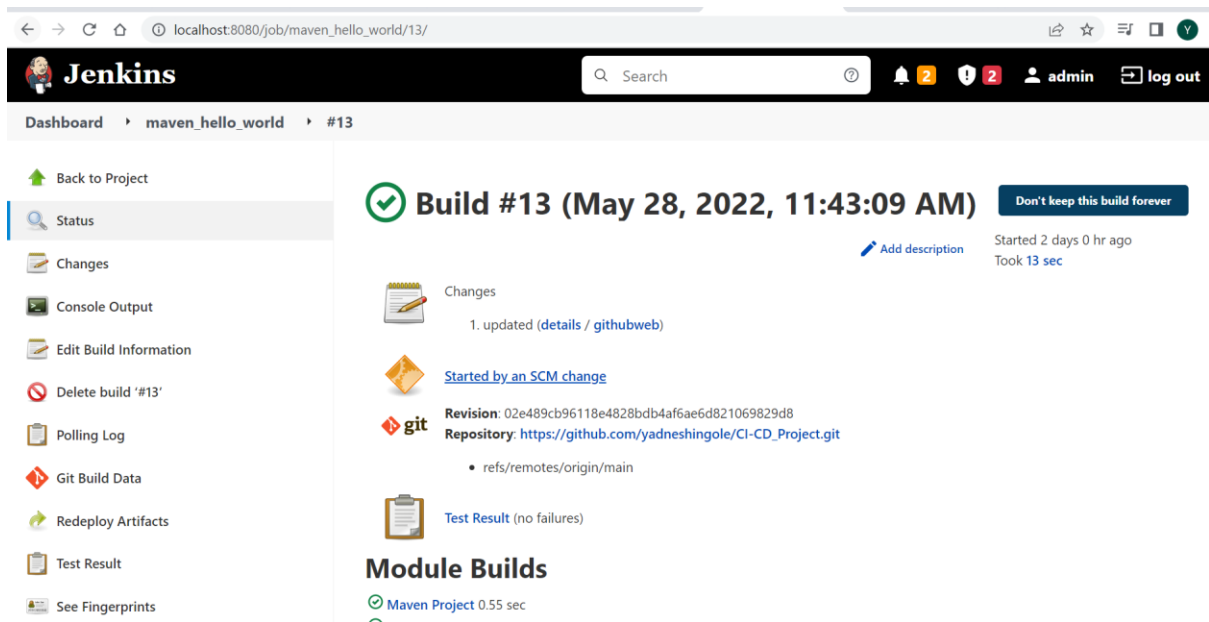
Latest Test Result (no failures)

Latest Test Result (no failures)

Permalinks

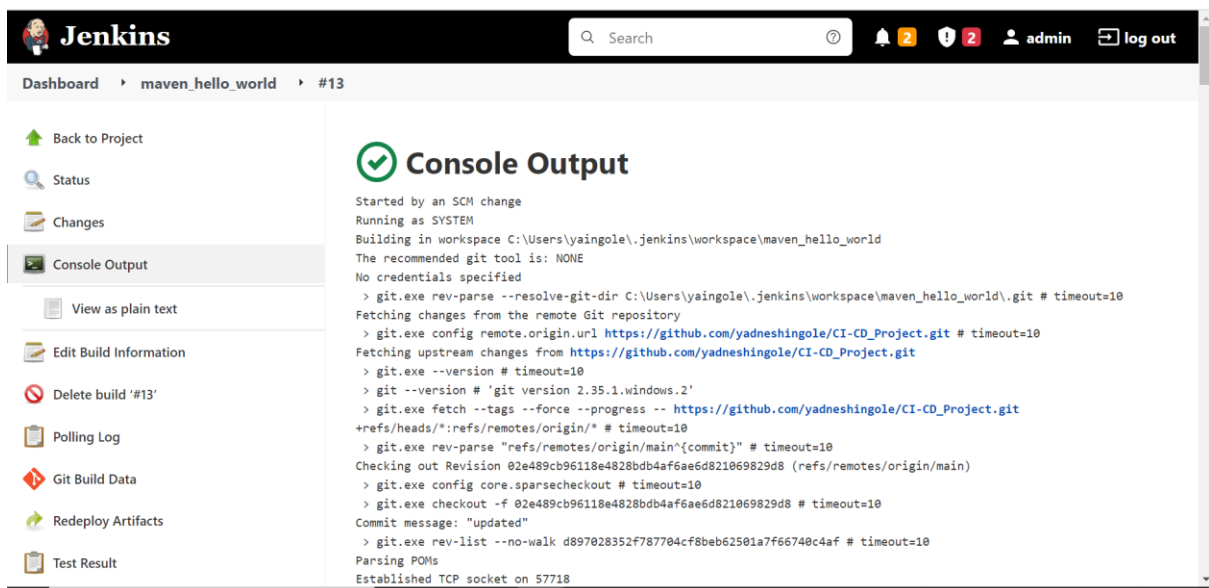
- Last build (#26), 28 min ago
- Last stable build (#26), 28 min ago
- Last successful build (#26), 28 min ago

Step 4: Click on save and build the project



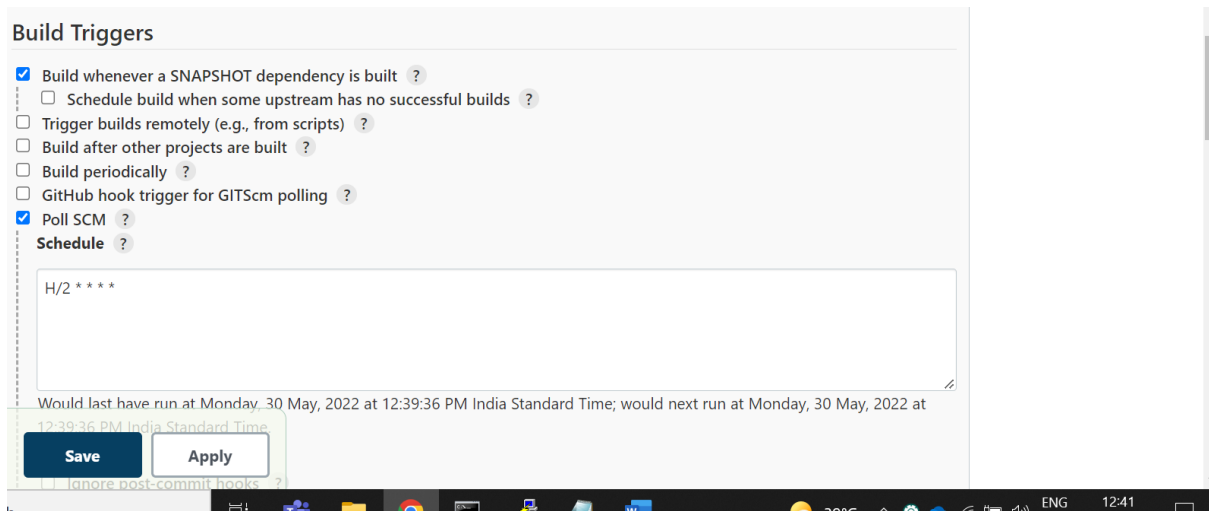
The screenshot shows the Jenkins web interface for a build named 'Build #13' on May 28, 2022, at 11:43:09 AM. The left sidebar contains a 'Console Output' link. The main content area shows the build status as 'Success' with a green checkmark. It lists changes, SCM information (GitHub repository), and a test result showing 'Maven Project' completed in 0.55 seconds. A 'Don't keep this build forever' button is visible in the top right.

Step 5: Click on Console Ouput to see the Output



The screenshot shows the Jenkins 'Console Output' page for Build #13. The left sidebar has 'Console Output' selected. The main area displays the build's execution log, starting with 'Started by an SCM change' and 'Running as SYSTEM'. It shows the workspace path, git tool configuration, and the execution of git commands to fetch and checkout the latest code from the repository. The log ends with 'Established TCP socket on 57718'.

Step 6: In build trigger section checkbox on poll scm and add H/2 * * * * means it check git repository every 2 minutes if any change is there it automatically build the project again.



Now Continuous Integration is complete

Step 7: Now Make ec2 instance on aws and install tomcat8 onto it

- Launch as ec2-user with the help of putty

Step 8: Go back to configure project in Jenkins and add post build action

- Enter the url of tomcat server and credentials
- Apply and save the project
- Build the project again

General Source Code Management Build Triggers Build Environment Pre Steps Build Post Steps Build Settings

Post-build Actions

Context path ?

Containers

Tomcat 8.x Remote Credentials

deployer/***** (Tomcat Credentials) Add

Tomcat URL ?

http://54.242.129.231:8090/

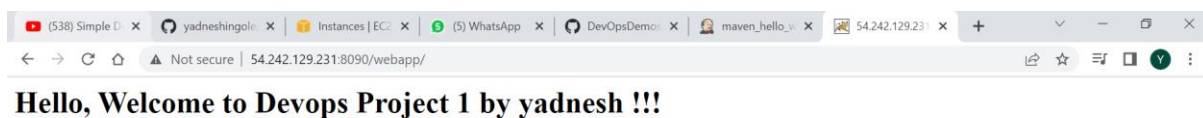
Advanced...

Add Container

Save Apply

You can access it in web browser

<http://54.242.129.231:8090/webapp/>



Now we have successfully completed the continuous deployment step.

so whenever push is done or file is updated on github, jenkins uses poll scm to trigger the build and deploy it on tomcat server so that client can see the changes made to application for any new features or bugs simultaneously and make changes accordingly.

we don't need to worry about compiling the code, creating distributable jar and war file, managing the project structure and deploying it on tomcat server.