**Spring boot application deploy on AWS**

**Master CI/CD Pipeline using Jenkins in one shot | Hindi**

[**https://www.youtube.com/watch?v=y4RcDlfYKB8**](https://www.youtube.com/watch?v=y4RcDlfYKB8)

**Build and deployment process**

1. Developer develops the project and push into version control system like github or bitbucket.
2. Build the code – then exe of the application is generated
3. Test the code
4. Deploy the code

Develop –> Build -> Test -> Deploy

By default all the process is done manually.

Since it is continuous process, it is tiresome process

Dev-Ops [Devops]

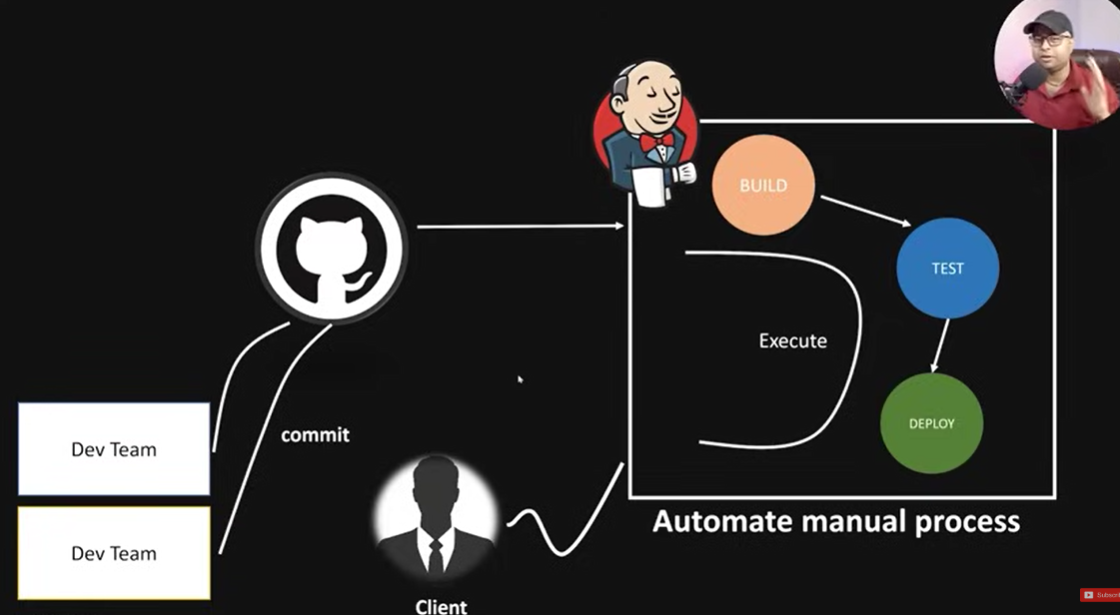
Development + Operation

Project Deve + Build & Deploy

* Challenges in the manual work

1. Dev write the code every day and follow process every day.
2. Multiple environment [dev - prod]
3. Time consuming
4. Error prone

**Jenkins :-** it is tool or software which is used to automate build and deployment process so that developer don’t need to do the process manually after every change.



**Environments in software development :-**

1. **Dev env**
2. **QA – quality assurance env**
3. **UAT – user acceptance testing**
4. **Pilot env – pre-production env :-** performance testing
5. **Production env –** project is live here.

**CI/CD :-**

**Continuous Integration/Continuous delivery or deployment**

**Build+test+deploy / release+build+deploy**

**Lets implement CI/CD pipeline :-**

Create java project -> apache tomcat [server – install - deploy]

Create java project -> CI/CD pipeline through Jenkins

1. Download apache tomcat :- <https://tomcat.apache.org/download-90.cgi>

By default tomcat is running on port port :- 8080 So change the default port from 8080 to 8086 or port of your which is not used in any other service.

Change Tomcat Port to 8086

**Locate :- apache-tomcat-10.1.44\conf folder –** server.xml – change port from 8080 to 8086

Connector port="8080" to Connector port="8086"

[Make sure the JAVA\_HOME must be added to System variables – path\to\ jdk-17.0.12 and added Path %JAVA\_HOME%\bin added to Path ]

Now start tomcat server :-

**Install Jenkins :-**

**Download the jar file of Jenkins**

[**https://www.jenkins.io/download/**](https://www.jenkins.io/download/)

* Always Run the following command to start Jenkins :- it will run on port : 8080

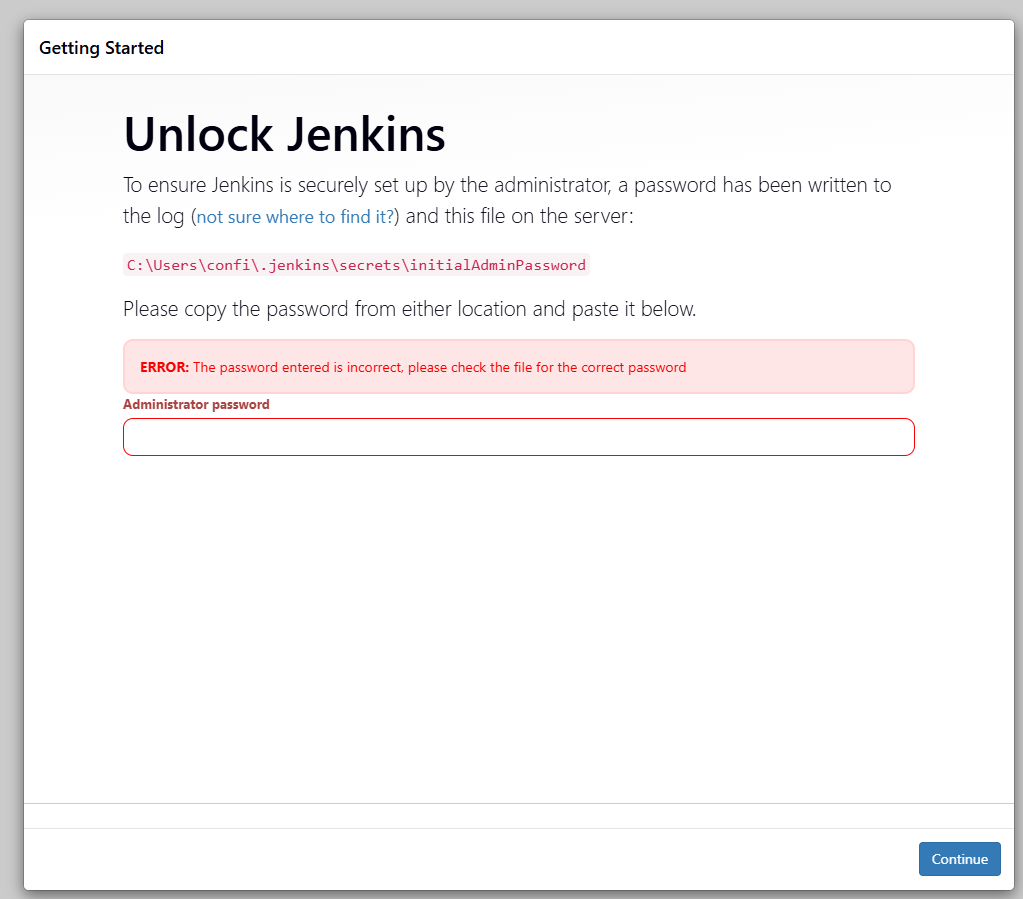
Add Jenkins.war fil in c drive or any location and enter blow command from that path.

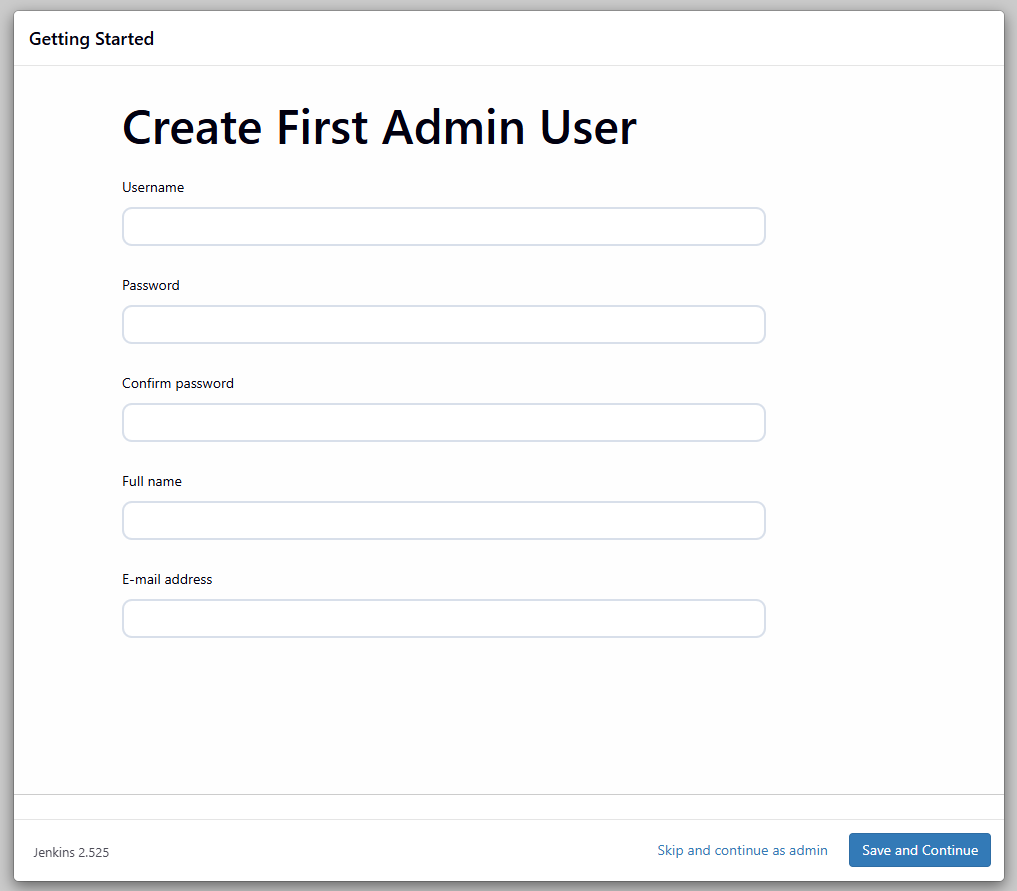
* **Command :- java -jar .\jenkins.war**

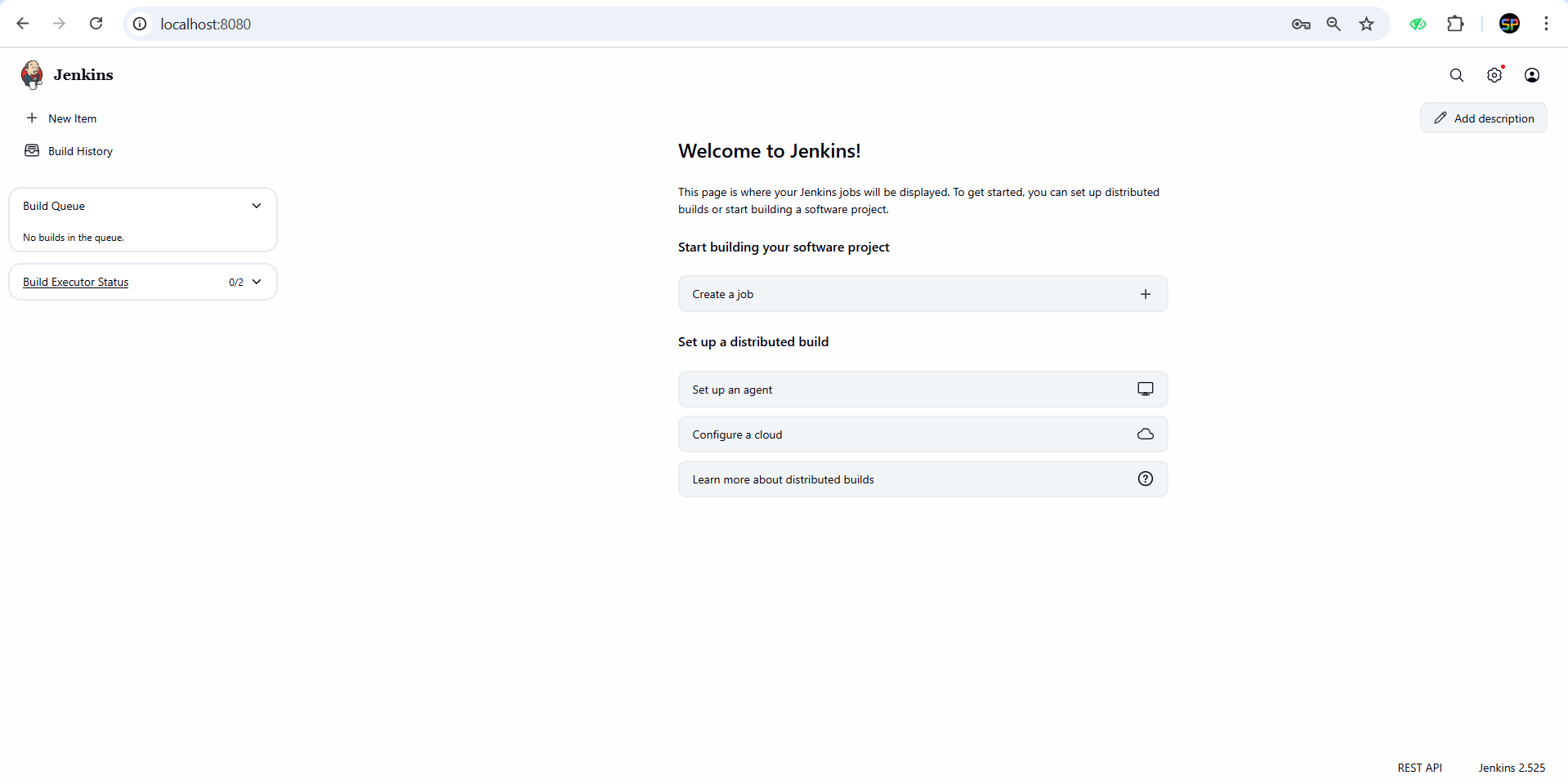
**Now, Jenkins is running on** [**http://localhost:8080/**](http://localhost:8080/)

Search in chat gpt :-

1. Change Tomcat Port to 8086
2. Steps to Auto-Deploy Spring Boot WAR from Jenkins to Tomcat

****

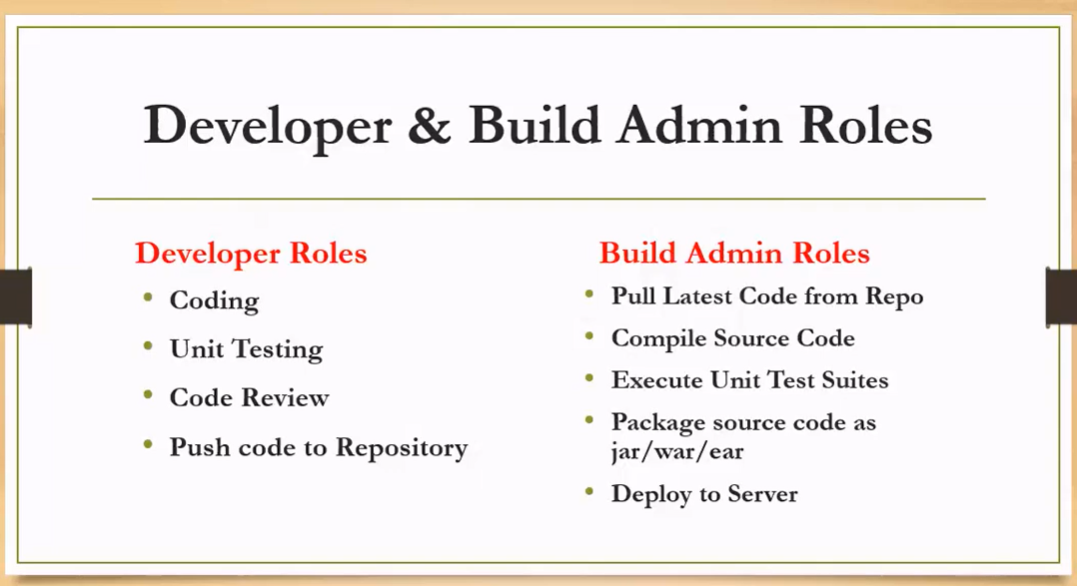


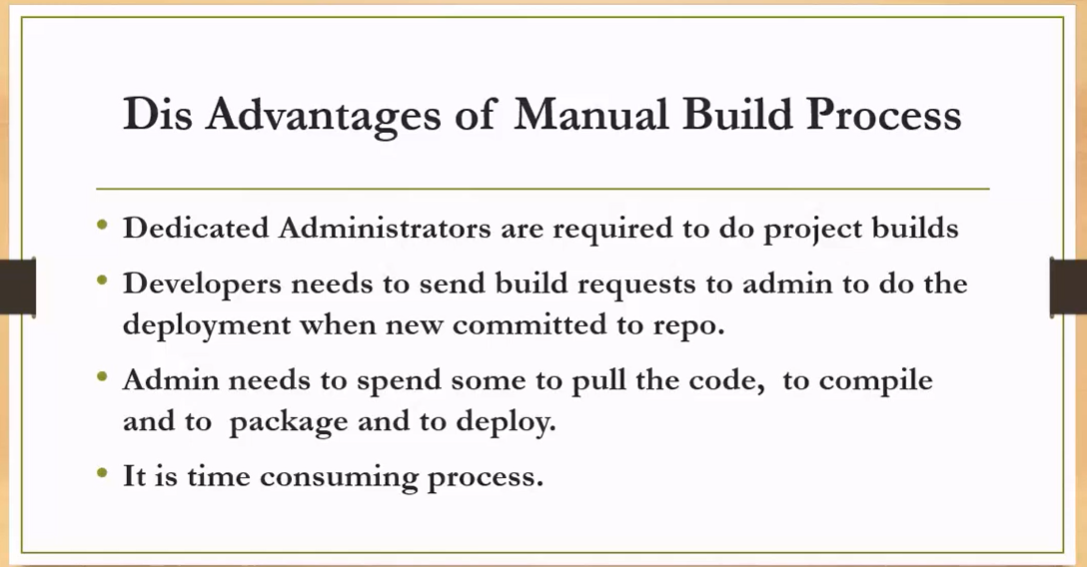


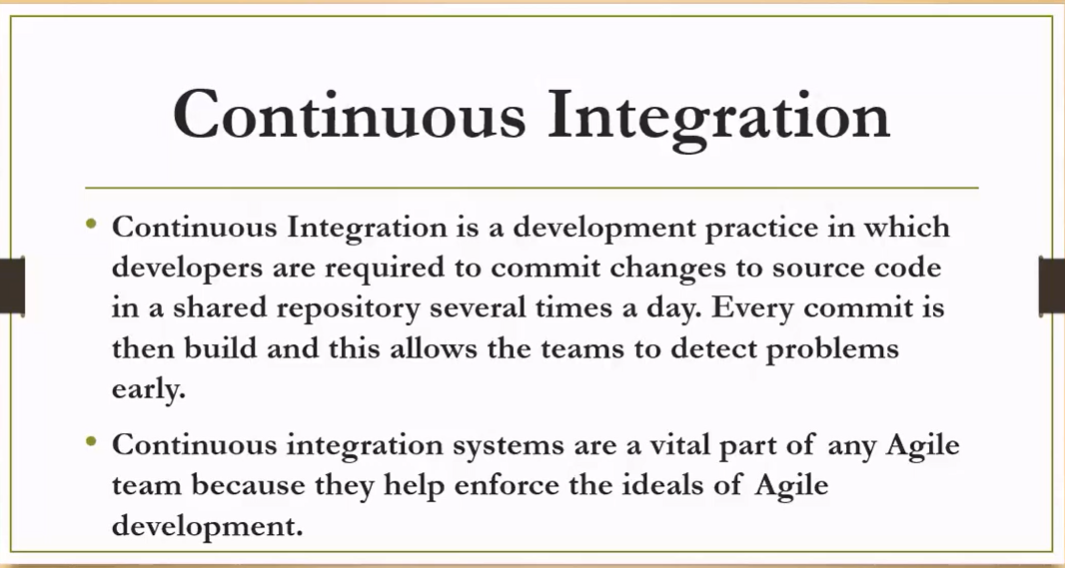
**Password is display in the cmd use that password.**

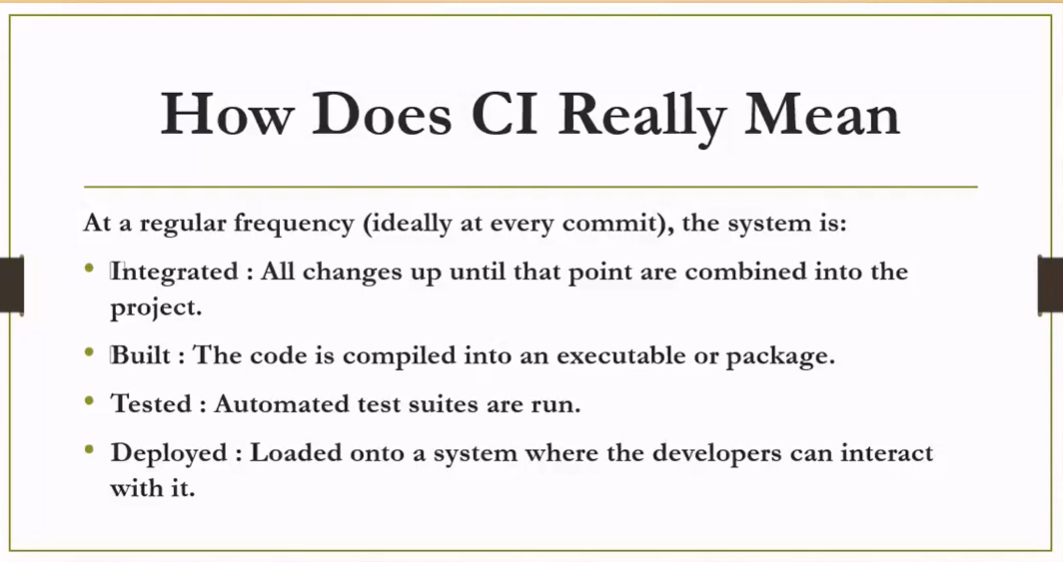
**Plugins are most powerful in jenkins**

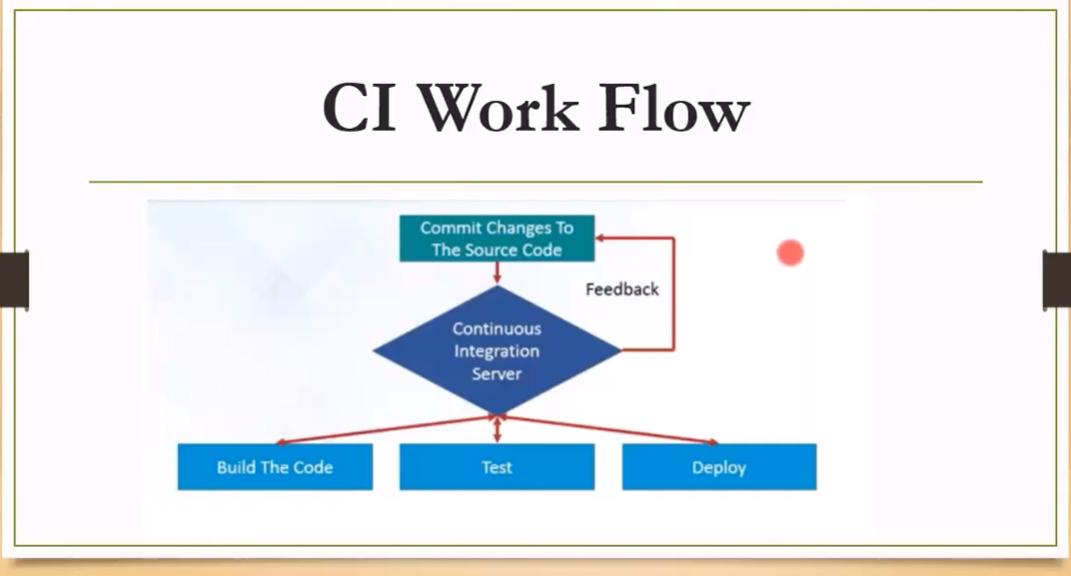
* Setting -> plugins -> Add require plugins like maven integration, deploy to container.

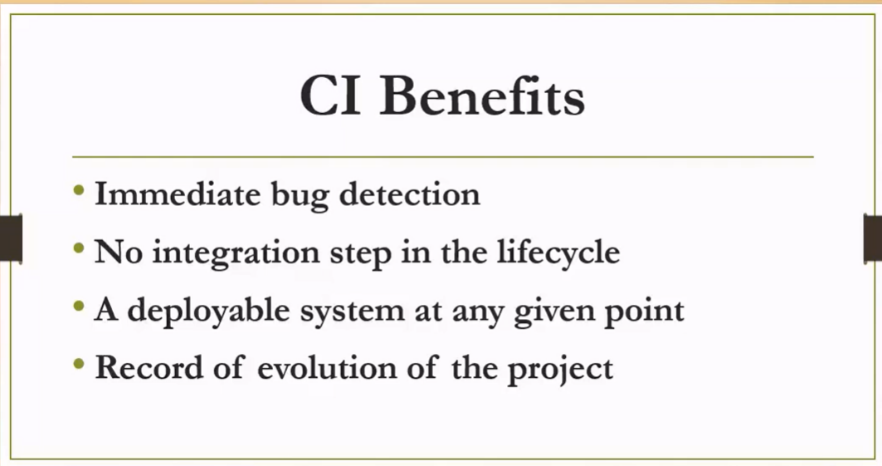


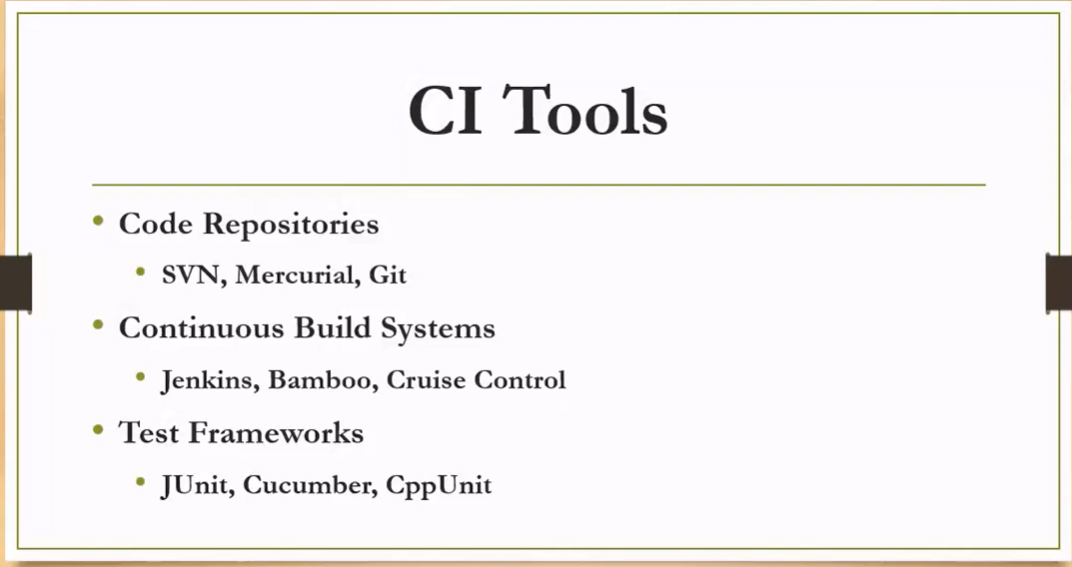


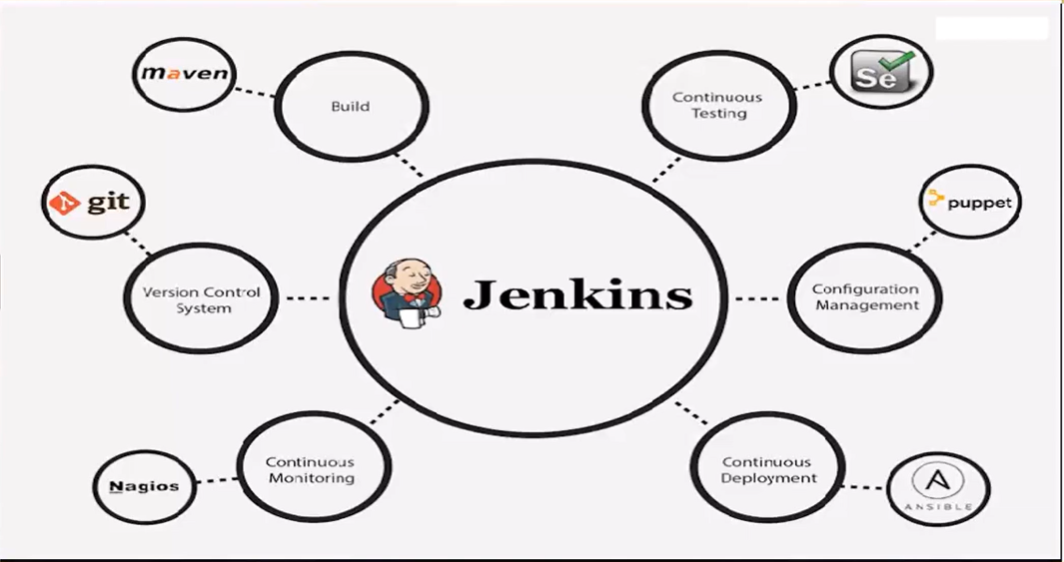


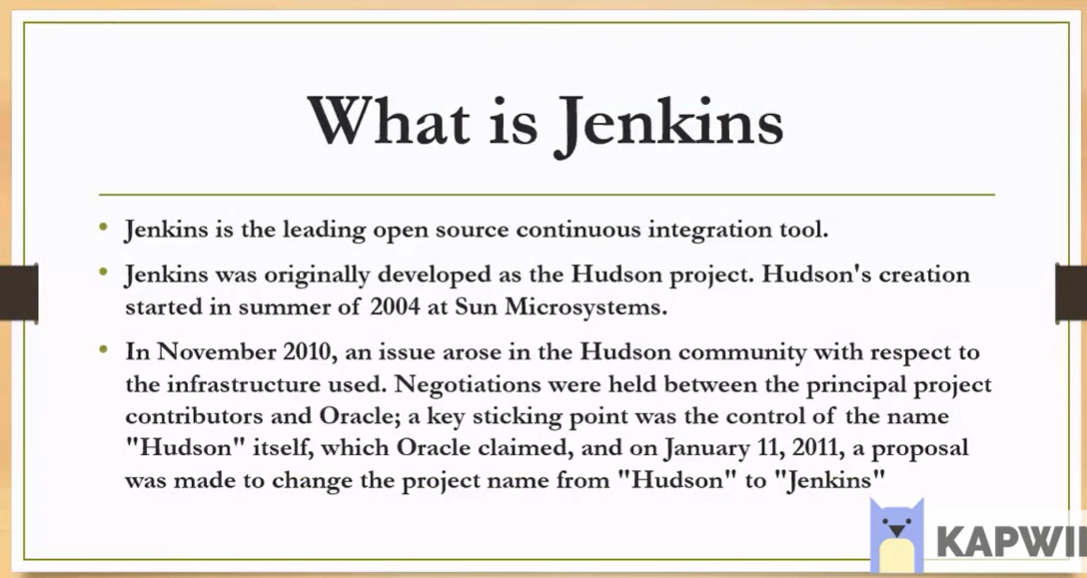


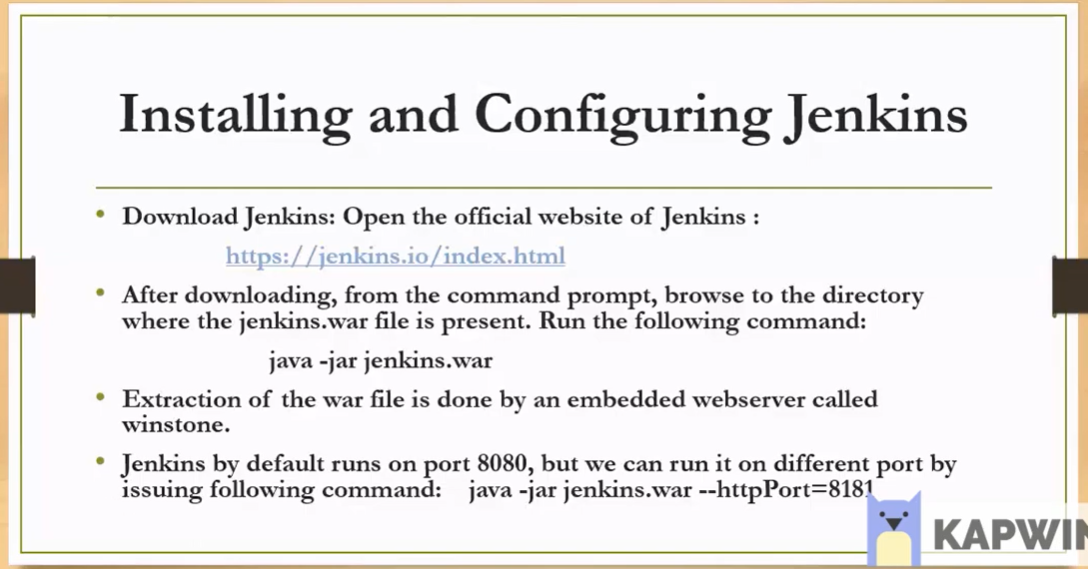




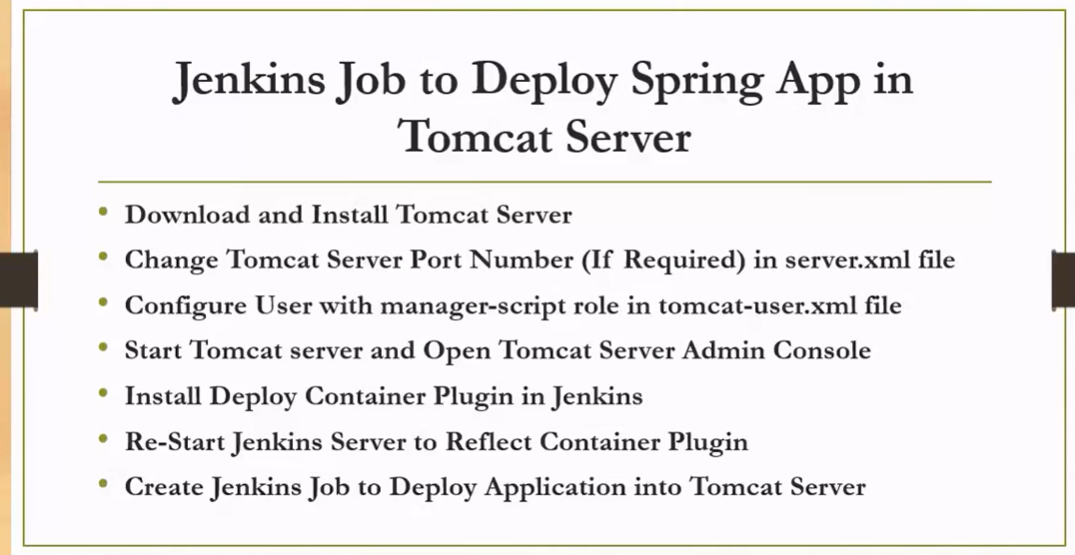


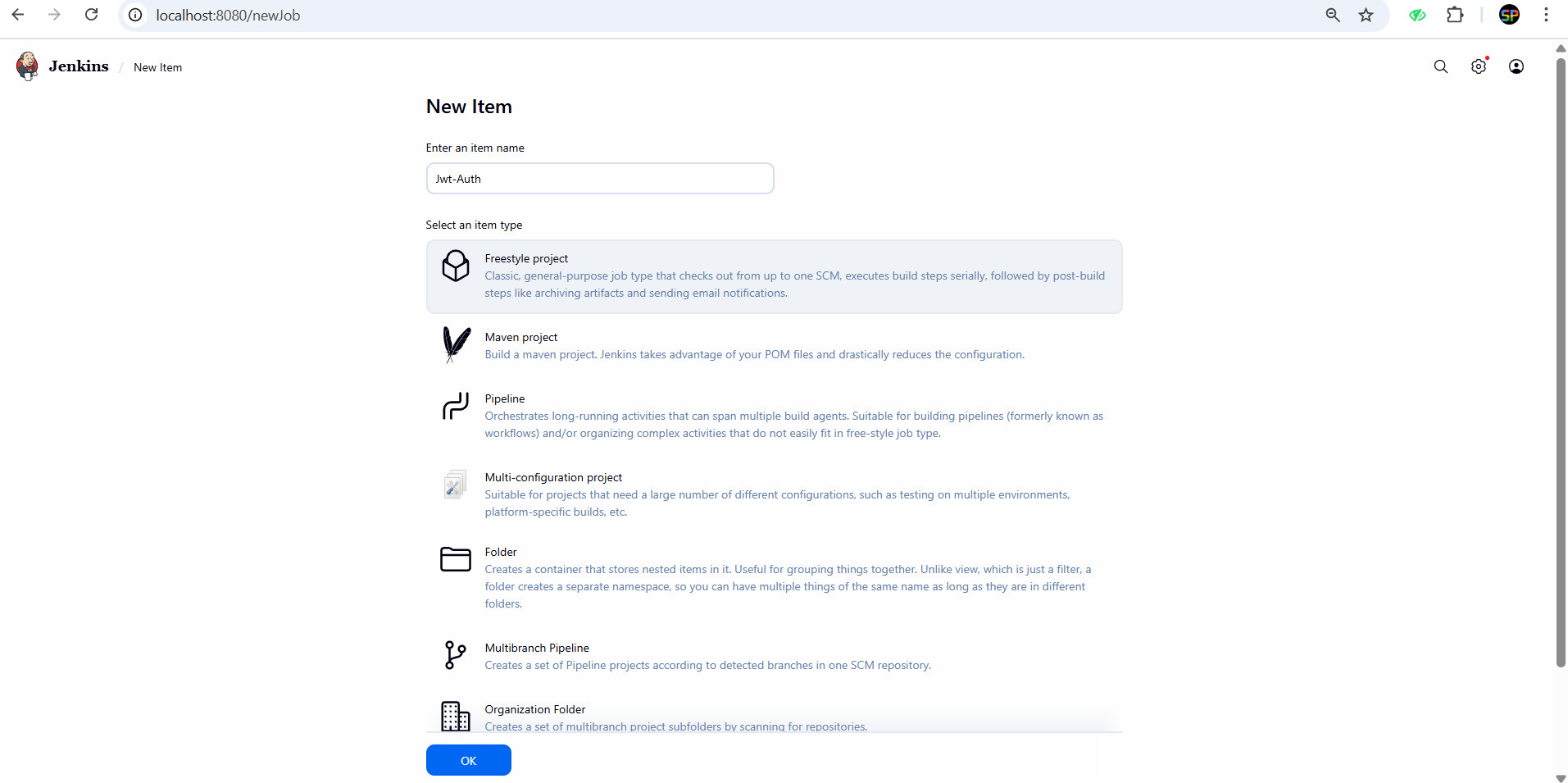


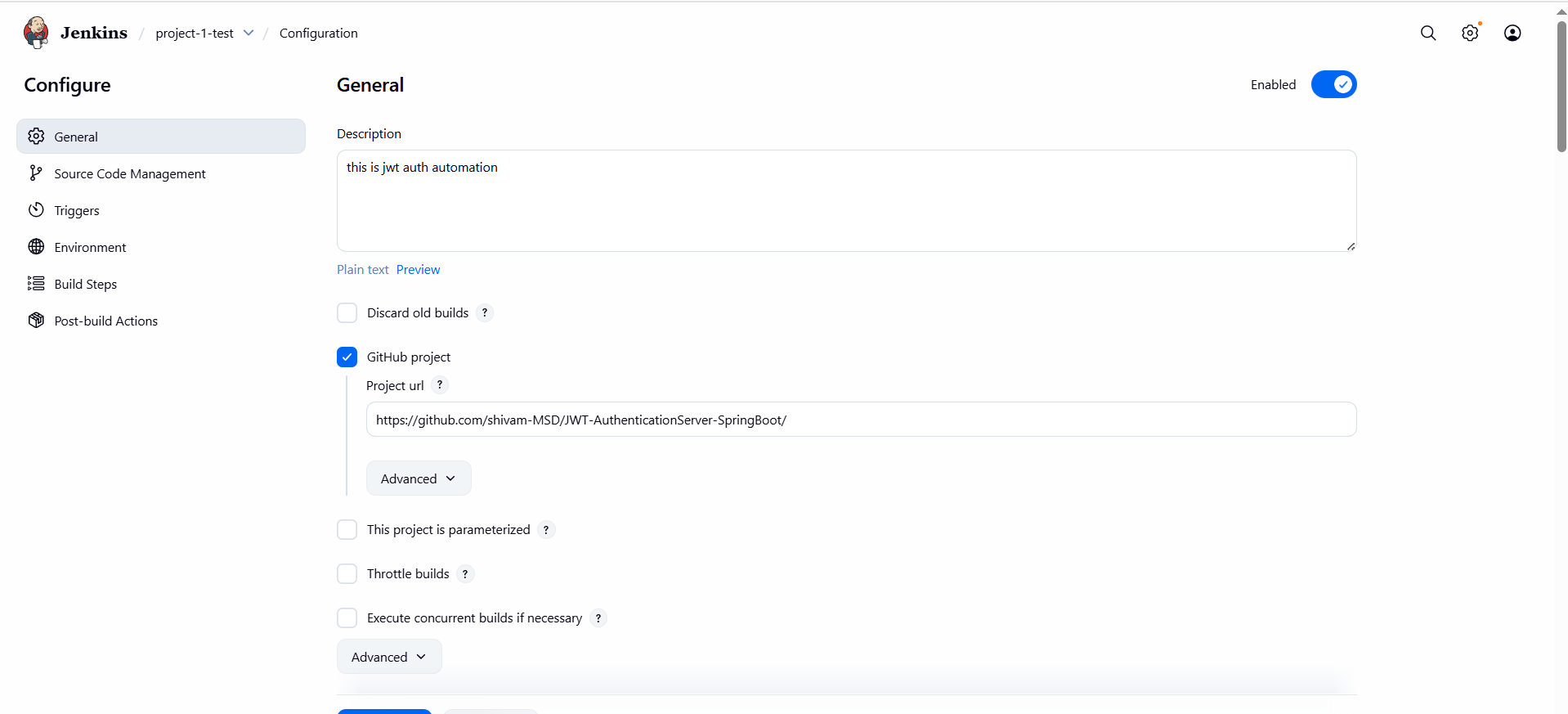


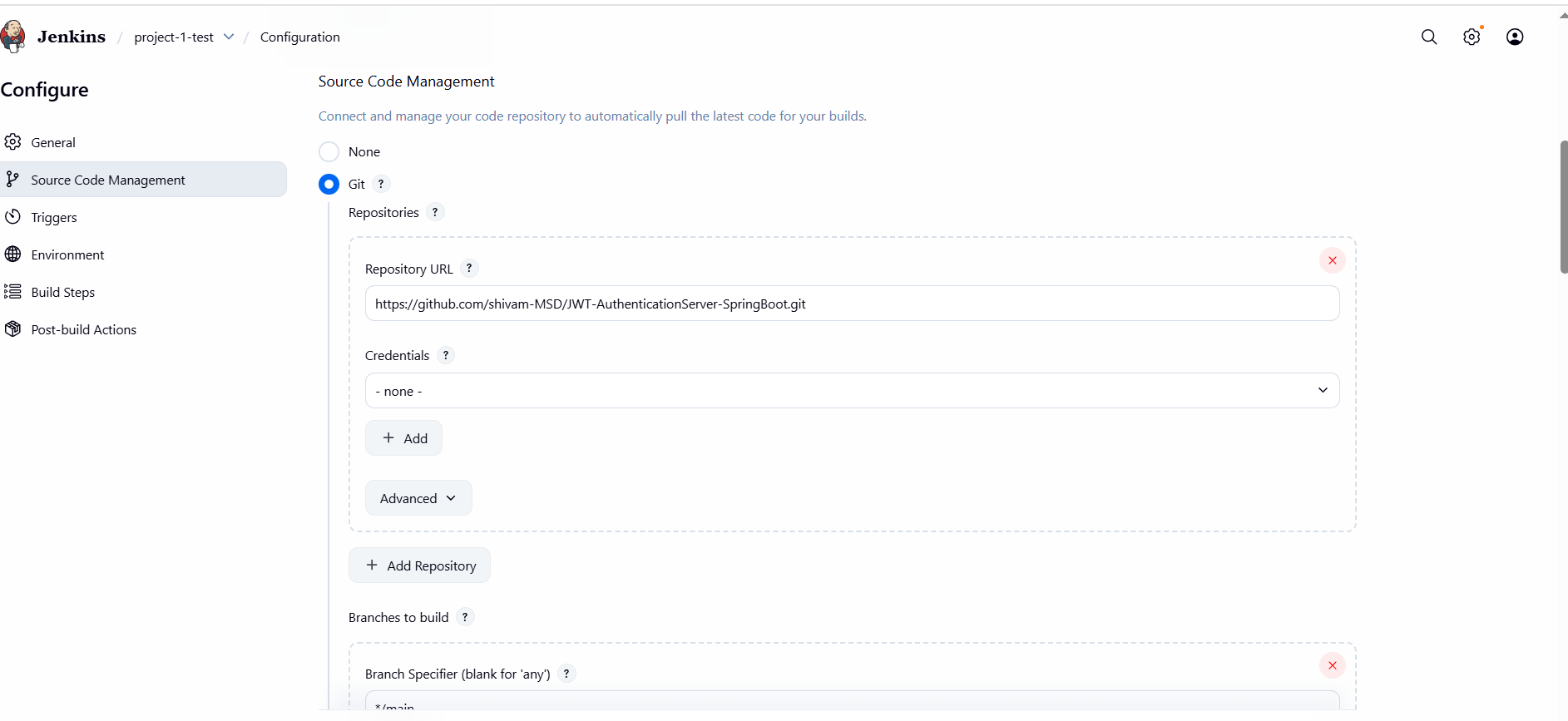




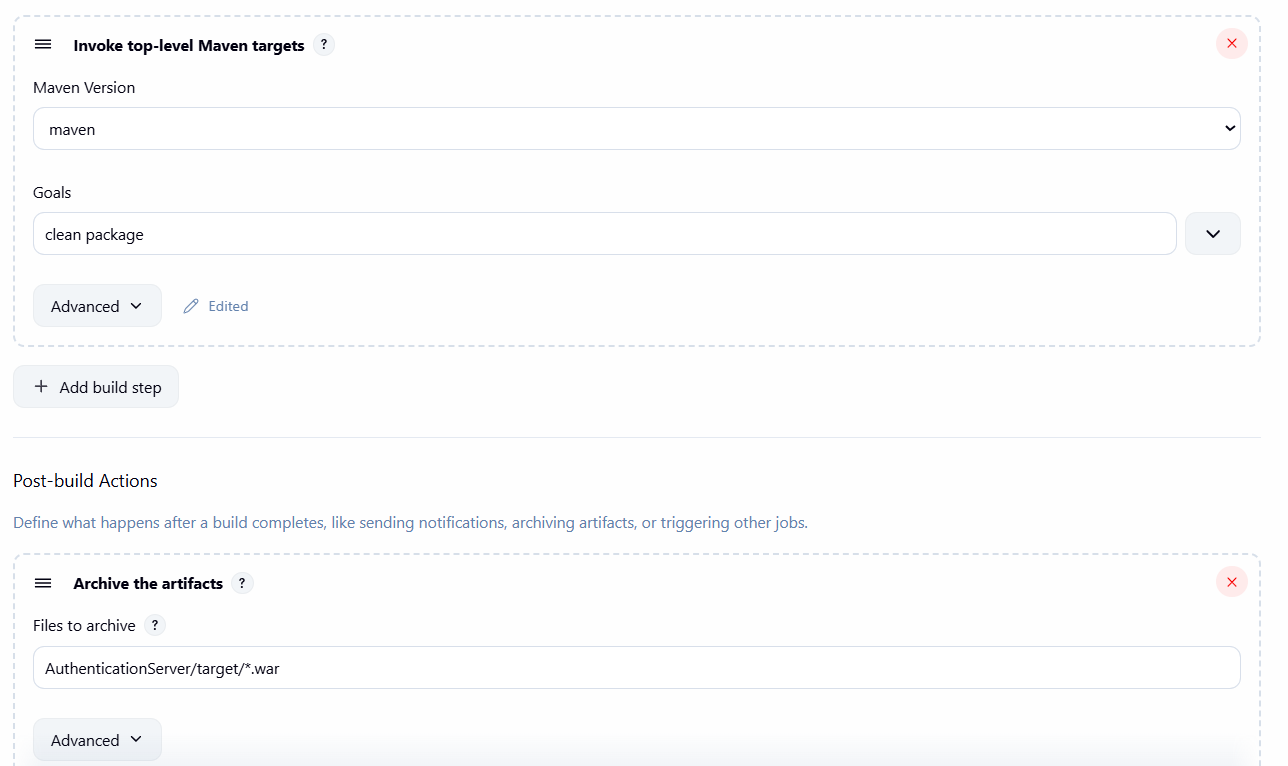


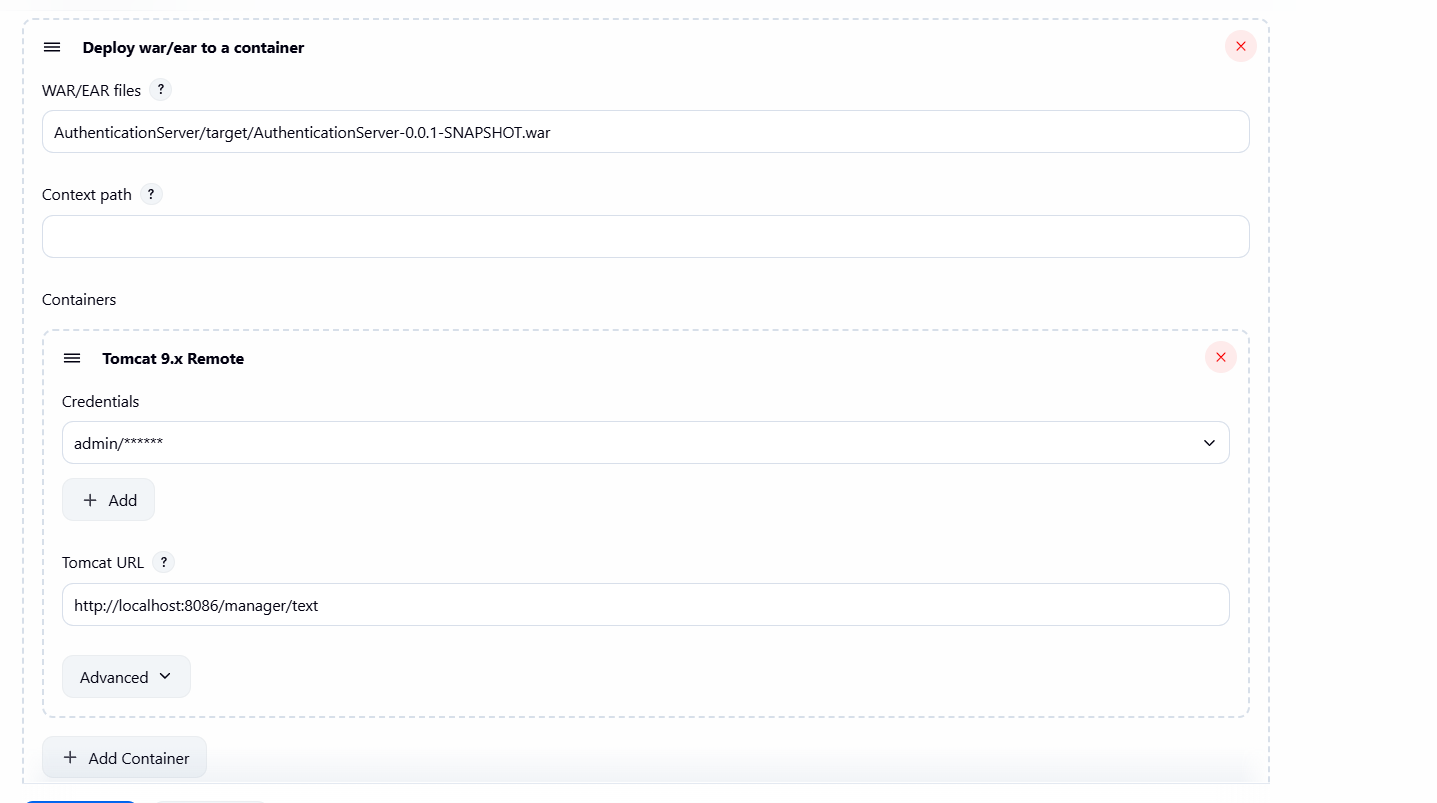






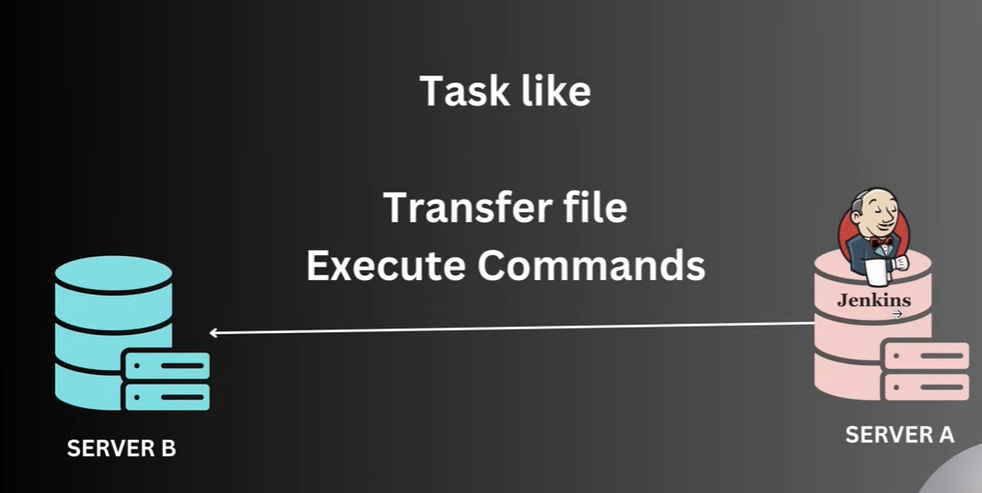
Poll SCM :- \* \* \* \* \*



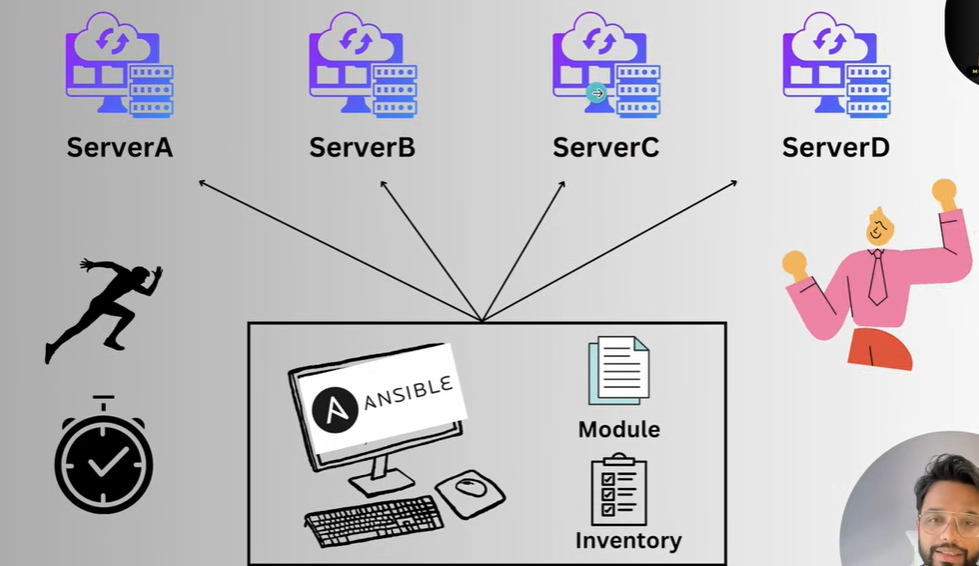


Email Notification :-

Manage Jenkins -> System -> scroll to last -> there is email section and enter your email.

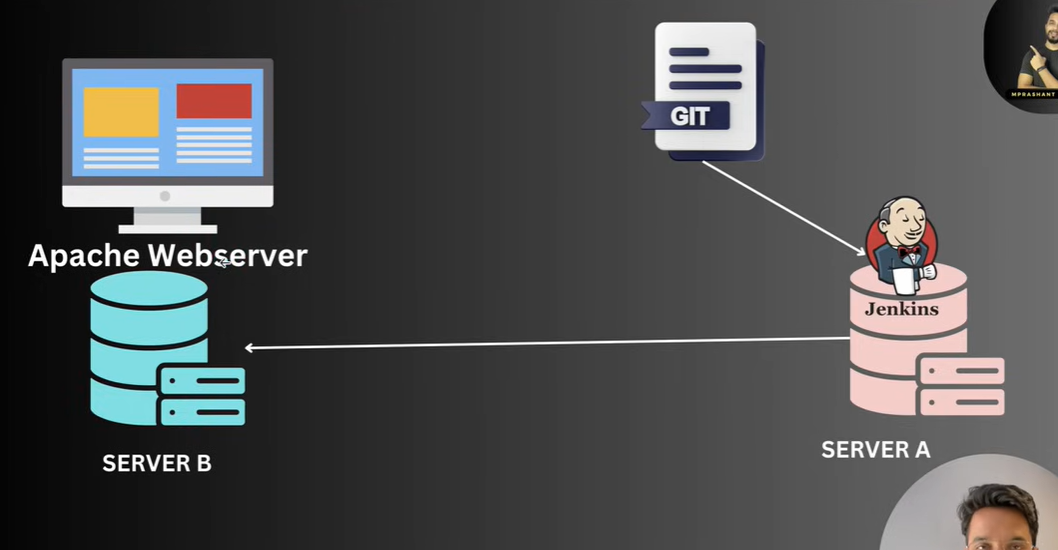


Ansible + Jenkins :-

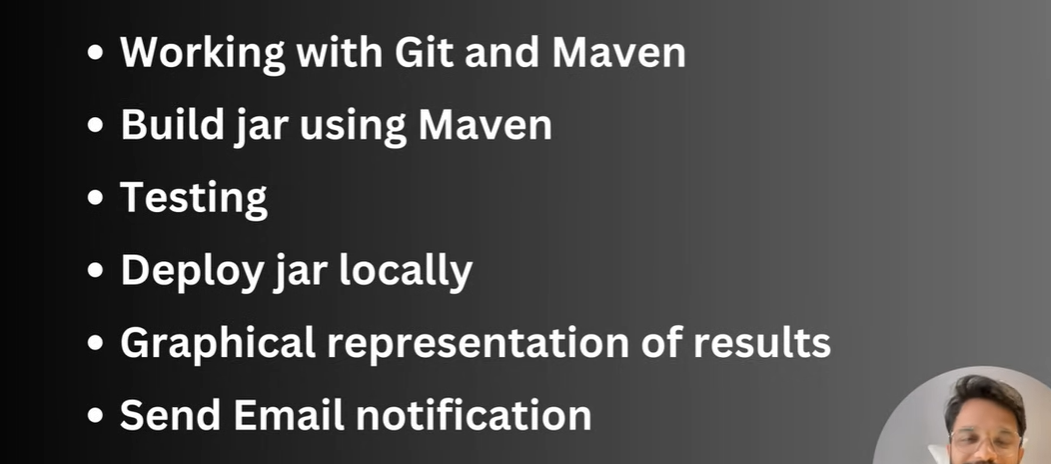


All configuration in one system and you can automat many task from one system instead of doing in all the server.

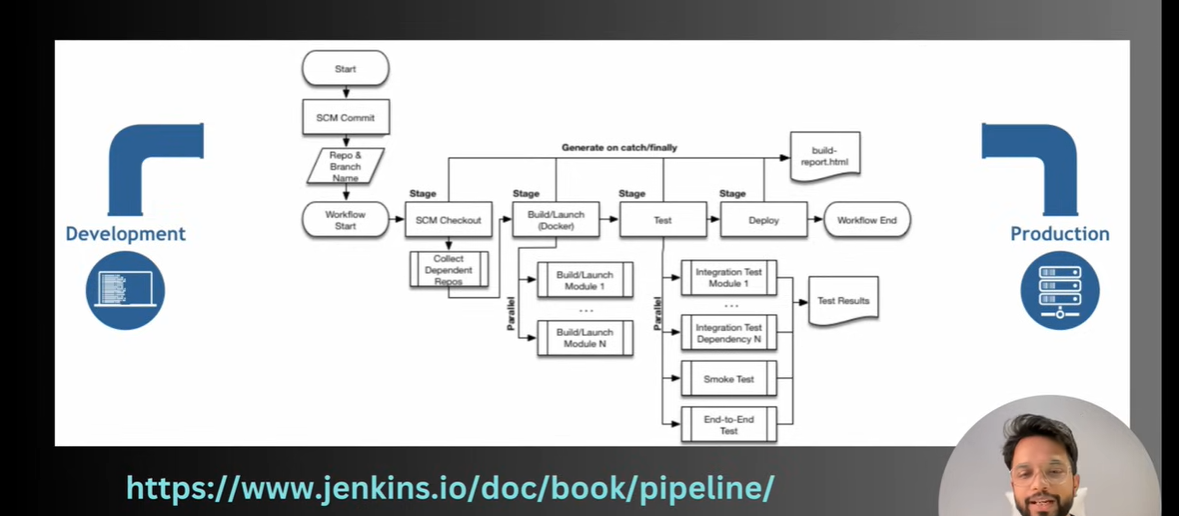
Add plugin :- ansible











**Jenkins from Chatgpt :-**

**Jenkins learning — Copy-paste prompt kit for ChatGPT & Gemini (beginner → advanced, with Spring Boot examples)**

Nice — you want a complete, **actionable** prompt collection you can paste into ChatGPT or Gemini to learn Jenkins end-to-end. Below is a structured kit: a short meta-prompt template you can reuse, then many ready-to-paste prompts broken into **Beginner → Intermediate → Advanced → Production** sections, plus hands-on labs, troubleshooting prompts, cheat-sheets and example Jenkinsfiles for a Java Spring Boot project (Maven). Replace placeholders like <GIT\_REPO\_URL>, <DOCKER\_REGISTRY>, <CREDENTIAL\_ID>, <K8S\_NAMESPACE> before running the prompts.

Tip: for best results prefix prompts with *“Act as a senior DevOps engineer and teacher”* and ask for step-by-step commands + exact files to copy/paste.

Meta-prompt (use this as the base for any Jenkins question)

**Act as a senior DevOps engineer and hands-on teacher. I am a beginner at Jenkins and I want step-by-step guidance I can run on my local machine. Explain WHY we do each step (conceptual), EXACT commands and files to copy/paste, and show expected outputs or how to verify success. Use a Java Spring Boot Maven project hosted at <GIT\_REPO\_URL> as the example. Replace placeholders with real values when applicable. Keep explanations concise and give a final checklist of what I should have at the end. Where applicable provide a Jenkinsfile, shell commands, and configuration snippets (Jenkins UI steps or YAML for Configuration as Code).**

Beginner (core concepts & first pipeline)

### 1) What is Jenkins — simple conceptual explainer + 5-minute demo

**Prompt:**

**Explain what Jenkins is, why teams use it (CI/CD), and the basic architecture (controller, agents, executors, jobs, pipelines). Give a 5-minute hands-on demo plan for absolute beginners: commands to run Jenkins in Docker, first login steps, install minimal plugins (Git, Pipeline, Credentials), and create a simple Declarative pipeline that checks out <GIT\_REPO\_URL> and runs `./mvnw -B -DskipTests package`. Provide copyable commands and the exact Jenkinsfile.**

**What to expect:** conceptual summary + commands:

**# run Jenkins (quick)**

**docker run -d --name jenkins \**

**-p 8080:8080 -p 50000:50000 \**

**-v jenkins\_home:/var/jenkins\_home \**

**jenkins/jenkins:lts**

and a minimal Jenkinsfile (provided by the assistant).

### 2) Install Jenkins (Docker, Ubuntu apt, and Windows) — copy/paste steps

**Prompt:**

**Show step-by-step installation instructions for Jenkins on:**

**1) Docker (Linux/macOS/Windows)**

**2) Ubuntu 22.04 (apt)**

**3) Windows as a service**

**For each: exact commands, how to open initialAdminPassword, how to install recommended plugins, and how to create an admin user. Also show how to install plugins via CLI (jenkins-plugin-cli) for automation.**

### 3) Create your first Declarative pipeline for Spring Boot (Maven)

**Prompt:**

Create a full Declarative Jenkinsfile for a Java Spring Boot project using Maven wrapper (./mvnw). The pipeline should:

- checkout from <GIT\_REPO\_URL>

- build (mvn -B -DskipTests package)

- run unit tests and publish JUnit results

- archive the built jar

- run static code analysis step placeholder

- show environment variables and use credentials id <GIT\_CREDENTIALS\_ID> for private repo access

Provide the complete Jenkinsfile and explain each block.

Intermediate (useful pipelines, plugins, integrations)

### 4) Multibranch pipeline + GitHub webhooks (build PRs & branches)

**Prompt:**

**Explain how to set up a Multibranch Pipeline in Jenkins for a GitHub repo with webhooks so every branch and PR is built. Provide:**

**- Jenkins UI steps for creating Multibranch Pipeline**

**- example GitHub webhook settings**

**- Jenkinsfile adjustments for PR builds (detect PR, report status)**

**- required plugins and how to install them (names only)**

### 5) Jenkinsfile: build → test → SonarQube → docker build → push

**Prompt:**

**Create a Declarative Jenkinsfile that:**

**1. Checks out <GIT\_REPO\_URL>**

**2. Runs mvn test and publishes JUnit results**

**3. Runs SonarQube analysis (use Sonar token credential id <SONAR\_TOKEN\_ID>)**

**4. Builds a Docker image and pushes to <DOCKER\_REGISTRY> with credential id <DOCKER\_CREDS>**

**5. Archives artifact and prints image digest**

**Explain the necessary Jenkins global tool configuration (Maven installation, Sonar, Docker credentials) and include exact pipeline code.**

### 6) Using credentials securely in Pipelines

**Prompt:**

**Explain how Jenkins credentials work and the secure ways to use them in a pipeline. Provide examples:**

**- withCredentials for username/password**

**- string credentials (API tokens)**

**- SSH private key usage for Git**

**Give code snippets showing each usage and explain how to create the credentials in UI and refer to them by ID in the Jenkinsfile.**

### 7) Declarative features: when, post, environment, agent, tools

**Prompt:**

**Explain Declarative pipeline directives (`agent`, `stages`, `environment`, `tools`, `when`, `post`, `options`) with 1–2 real Spring Boot examples each. Provide a single Jenkinsfile demonstrating them all, and short notes on when to prefer Declarative vs Scripted.**

Advanced (scaling, Kubernetes, shared libs, JCasC)

### 8) Jenkins on Kubernetes — ephemeral agents (Pod templates)

**Prompt:**

**Act as an expert DevOps engineer. Teach me how to run Jenkins agents as ephemeral Kubernetes pods using the Kubernetes plugin. Provide:**

**- a sample Pod YAML for a Maven/Docker build agent**

**- Jenkins configuration steps and the Jenkinsfile snippet using `agent { kubernetes { yaml '''...''' } }`**

**- notes on persistent Jenkins home, secrets, and scaling best practices**