**CI/CD pipeline deployment using Jenkins, maven & tomcat:**

* **Continuous integration (CI):**
* Continuous integration is the practice of automatically building and testing your code changes frequently, usually after each commit.
* **Continuous deployment (CD):**
* Automate the deployment of code changes to production after they pass automated testing.
* **Meaning plugin**
* Plugins gives add extra features and functionality to the software.
* **Meaning of Jenkins:**
* Jenkins is an open-source automation server that enables developers to build, test, and deploy software applications. It's a popular tool for implementing CI/CD.
* **Meaning of Maven:**
* Maven is a build automation tool for java-based projects. Its primarily used for building, packaging, and deploying java applications.
* **Meaning of Tomcat:**
* Tomcat is an open-source web server and servlet container developed by the Apache software foundation. It's a popular choice for deploying java-based web applications.
* **Workflow of github,maven,jenkins and tomcat:**
* Developers push their code into git hub repository.
* Devops engineer take the code from the GitHub.
* Using maven compiles the code, run tests, and packages the application into a deployable artifact (like a WAR file).
* After a successful build, Jenkins can deploy the artifact to a tomcat server.
* Tomcat can be monitored for performance and logs to ensure the application is running smoothly.
* **Prerequisites for Project deployment using Jenkins and tomcat:**

1. Aws account
2. GitHub account
3. Java and git
4. Maven
5. Tomcat
6. Jenkins

* **Steps:**
* Create 2 ec2 instances one for the Jenkins platform and one for the tomcat server.
* Setup Jenkins and tomcat
* Create Jenkins pipeline job which consists of 4 stages

1. CleanWs
2. Get the code from the GitHub
3. Build the code using Maven Tool (which compile+test+package the code to generate artifact file inside the target folder)
4. Deploy to the tomcat server.

* Let's explore each stage:
* CleanWs- which is used to clean the workspace after every build in declarative pipeline.
* Get the code and implemented webhooks for build trigger automatically when code commit occurs.
* Integrated maven with Jenkins (manage system-tools-add maven) and use “clean package” for generating the WAR artifact file.
* Integrated tomcat with Jenkins using the deploy to container plugin.

**Process:**

* Select Jenkins server –connect-open the terminal login to the slave machine.
* Install maven in Jenkins machine
* Install java in machine
* Install git in machine
* Install jenkins
* Login to the Jenkins dashboard
* Click on new node
* Enter Jenkins-slave
* Click on permanent agent
* Create
* Enter some information (name,description,root directory,labels,via ssh)
* Click on save.
* Go to manage plugins and installed some plugins.
* Couple of methods to create CI/CD projects in Jenkins:
* Freestyle method
* Declarative method
* Go to freestyle method-click on new item-enter an item name-click freestyle project-click ok
* Select restrict where this project can be run
* Give label expression (jenkins-slave)
* Under scm select repo URL
* Give branch name
* Click on invoke top level maven targets
* Enter “clean install”
* Select deploy war/ear to a container
* Enter \*\*./\*.war
* Select tomcat 9
* To authenticate jenkins on tomcat(give some username and password and id)
* Click add.
* Go to tomcat server
* Copy private ip(http://private ip)]
* Click save.
* Click on build now
* Go to the tomcat web app
* We see left side /freestyle web app
* Click that we received output

2. declarative pipeline:

Selct new item

Enter item name

Select pipeline

Click ok

Select pipeline script from scm

Select git

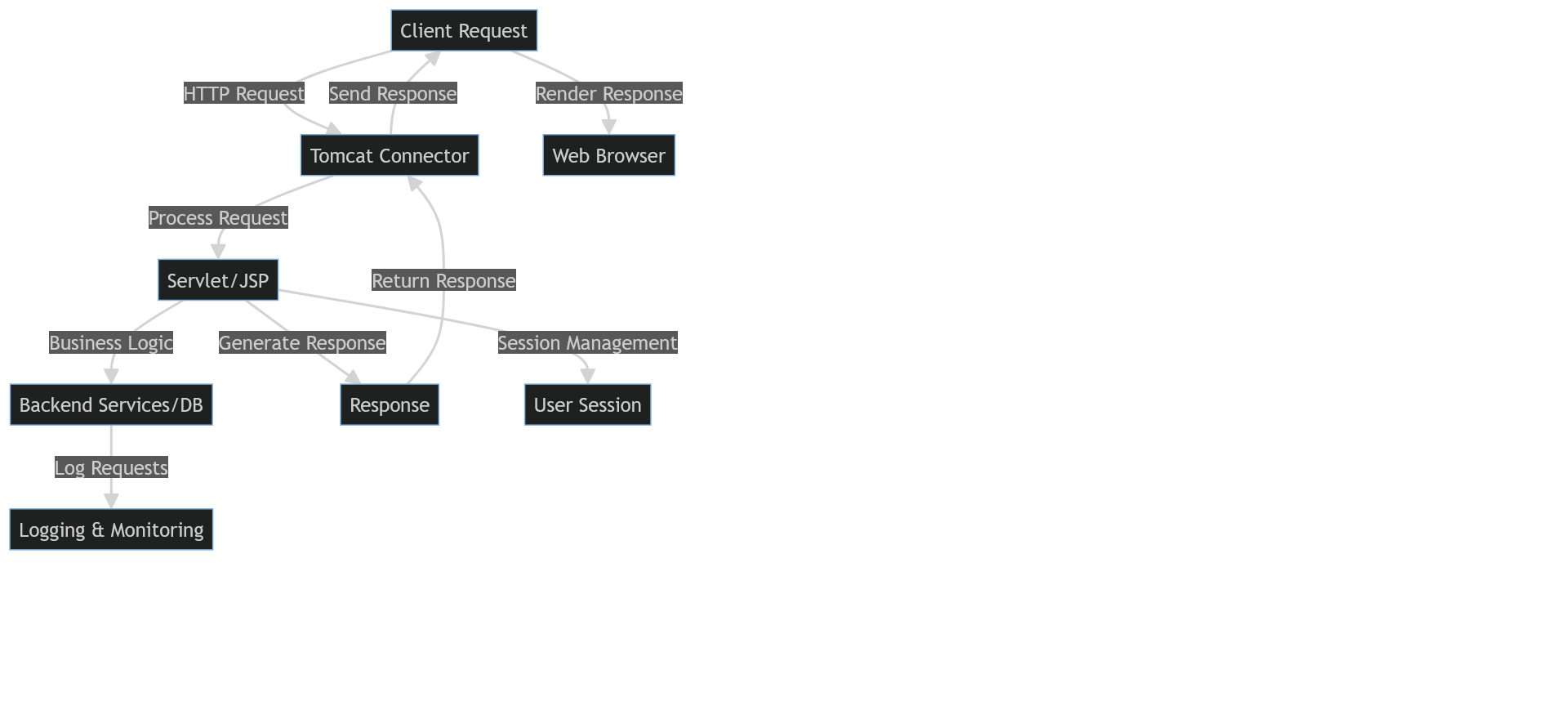
Go to github – copy the repo url and paste it

Go to the pipeline and see the stage view

Go to the tomcat see pipeline app

Click on the pipeline app

See the output.

* Tomcat workflow:
* 
* Users send HTTP requests to the tomcat server via a web browser or other HTP clients.
* Tomcat uses connectors (like HTTP, AJP) to hnadle incoming requests..
* The http connector listens for requests on a specified port.
* Once a request is received, tomcat processes it and determines which servlet or JSP should handle the request.
* The response is sent back through the connector to the client.
* Tomcat logs requests, errors, and performance metrics, which can be monitored for troubleshooting and optimization.