**PHASE FIVE PROJECT- CI/CD and EC2**

**Introduction**:

This*is* a simple Application which displays welcoming message for the user by rendering using s3 bucket and deployed using amazon EC2 instance. It only posts the given message based on URL slinked.

**Developer Details**:

**Name**: Abhishek Jana

**Tools Used:**

* Eclipse/Spring tool Suite: An IDE to code for the Spring applications
* Java: A programming language to develop the prototype
* MySQL: It is an open-source relational database management system
* Git: To connect and push files from the local system to GitHub
* AWS: Amazon web services has been used to set up an EC2 instance where the Spring Boot Application has been deployed\*
* Jenkins: It is used for scheduling the tasks. In our case, on every commit/ change in the github repo the build happens.
* WinSCM: To transfer the files securely from the local computer to the remote computer. In our case the jar file is transferred using this.

(\* The AWS EC2 instance is used in the labs provided by Simplilearn. )

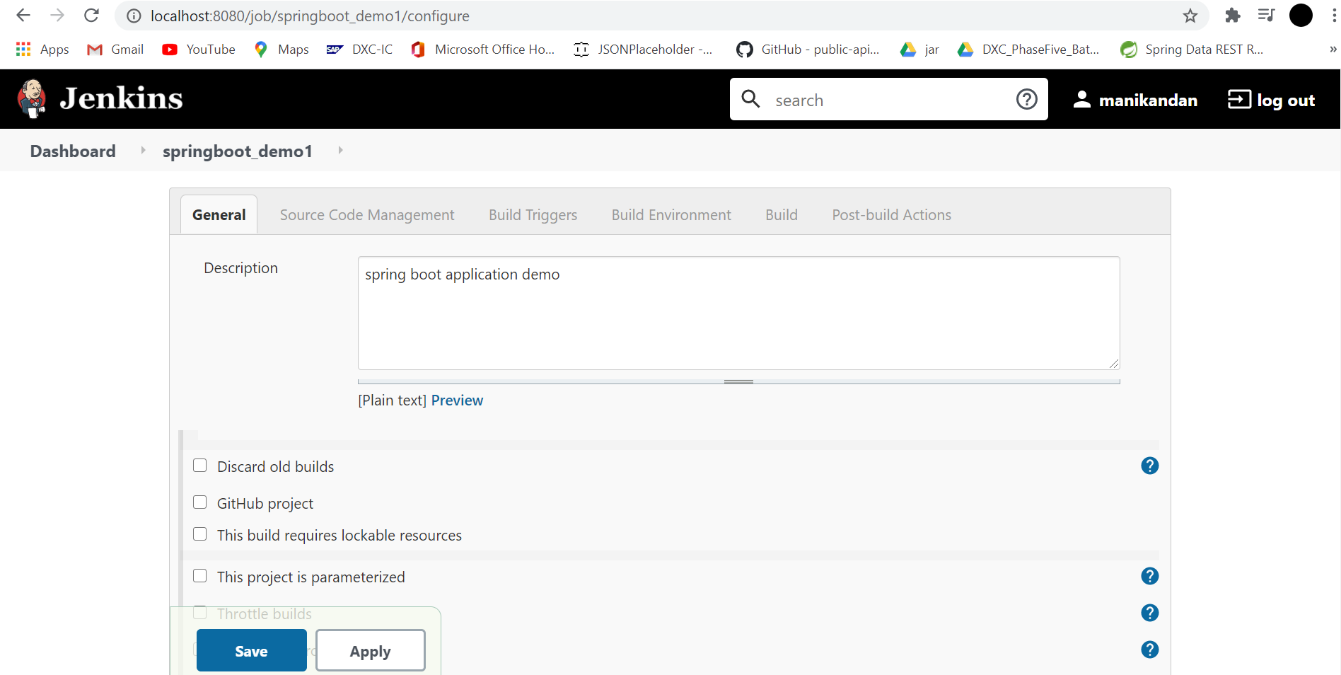
**Procedure:**

Jenkins is used to achieve CI/CD using pipelines and Poll SCM is used for further processes.

AWS EC2 instance is used for deployment of SpringBoot Application.

***JENKINS:***

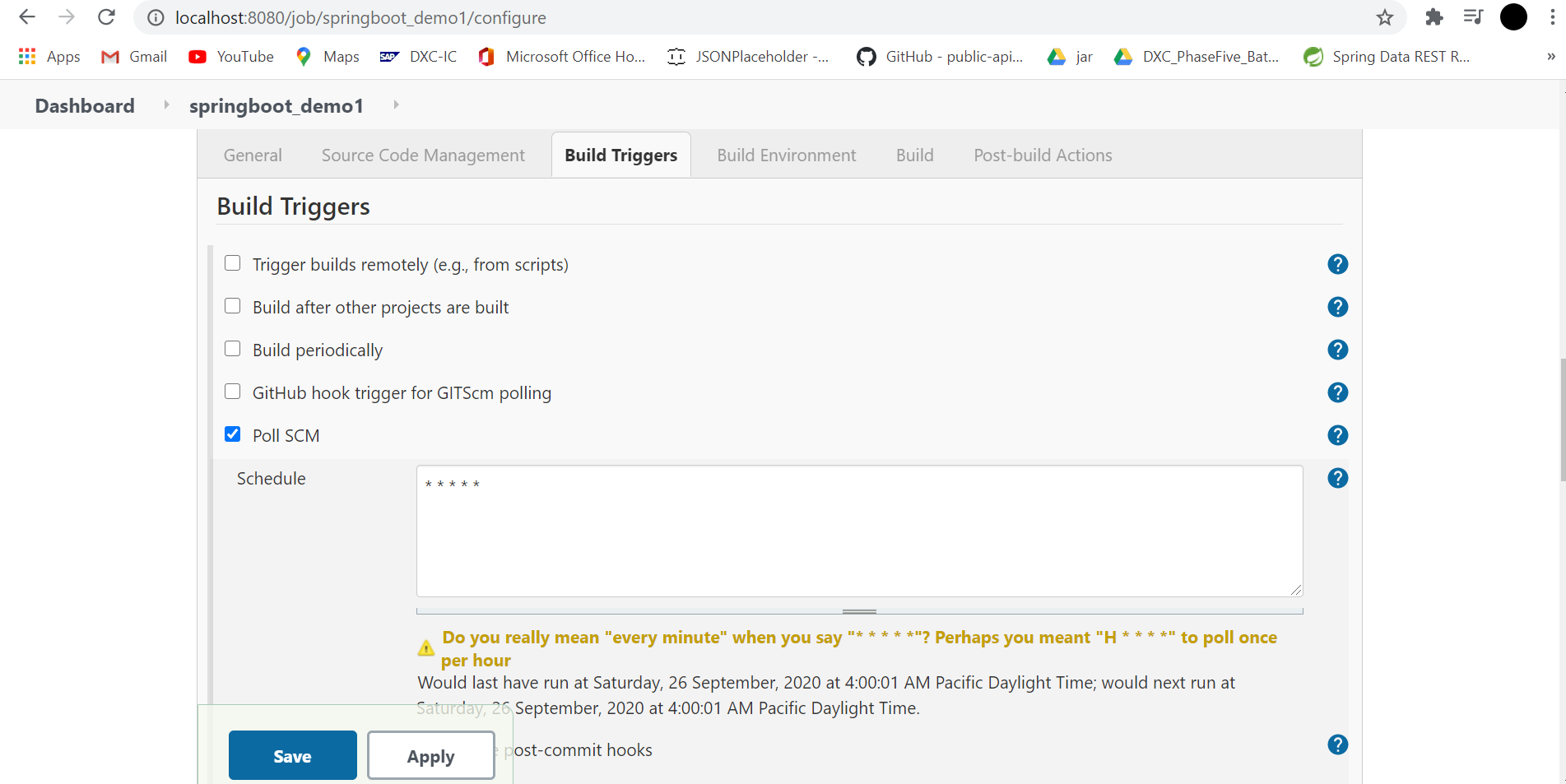
* Create a new freestyle project
* Add the plugin *deploy to container* in the manage extensions
* Select Git in the Source Code Management Tab and provide the repository URL and credentials if any



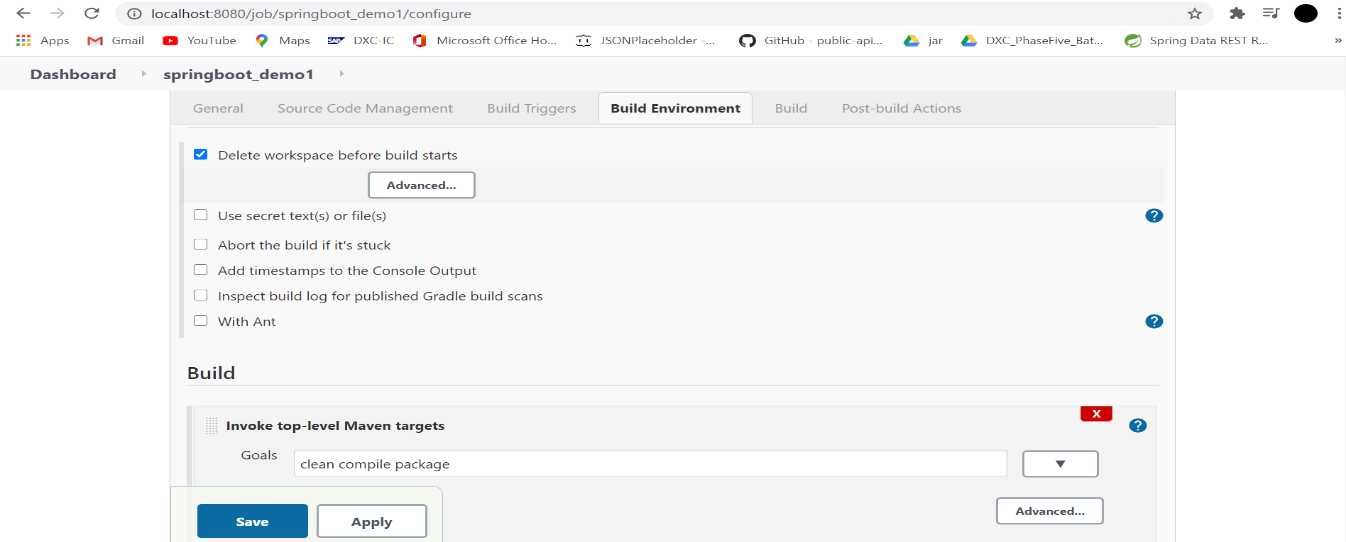
Graphical user interface, text

Description automatically generated

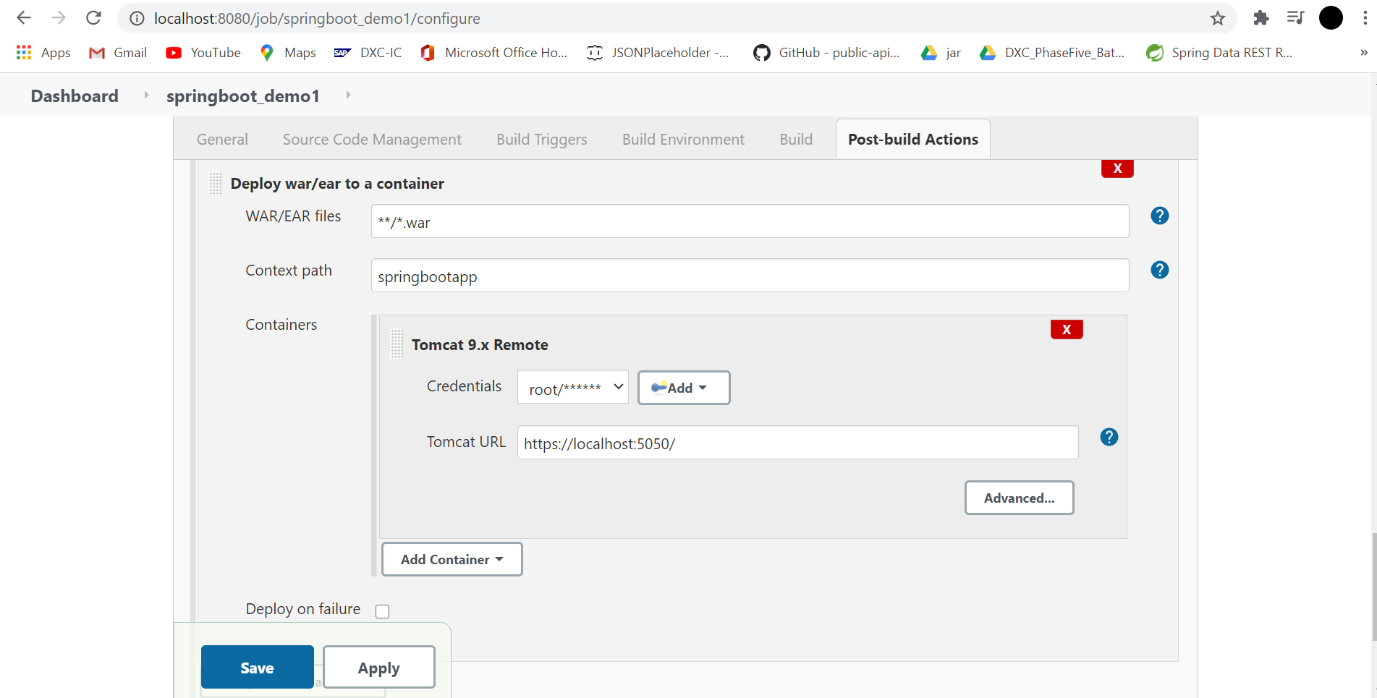
* In the Build triggers section select the build periodically or the Poll SCM option based on the requirement. Here we’ve selected Poll SCM with the value of \*\*\*\*\* (every minute) which denotes that it would run every minute to check for updates if any.



* In the build Environment tab, select Delete workspace before build starts thus deleting and starting a fresh workspace.



* In the Build tab, select Invoke top level maven targets and set the goals in the invoke top level maven targets as *clean compile package.*
* In the post Build Actions tab, add a post-build action *Deploy war/ear to a container*(only on installing the deploy to container).



* Select the format as war and the container as preferred tomcat version. Select the credentials and provide the URL of tomcat server as mentioned in the server configuration file.

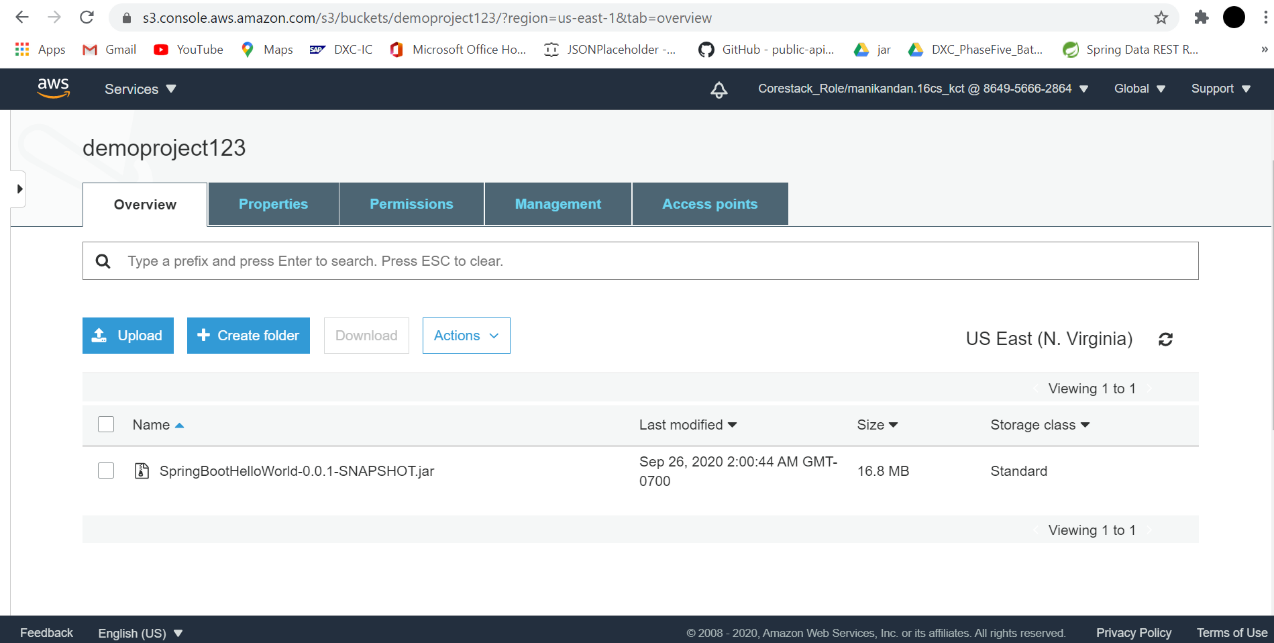
Repository Link:

https://github.com/abhishekjana/Phase5Project.git

***AWS-EC2:***

The EC2 instance was created using the launch instance button in the EC2 dashboard after logging in using the auth-link in the AWS web console tab.

* In the choose AMI tab select Ubuntu server and keep instance type and configure instance default.
* In add storage specify the size required and in the configure security group add a rule, in the port range select **the port of the Spring Boot application** will run with the source selected as **anywhere**
* Select Review and Launch
* In the create new key pair, specify the name and download the key pair for generating the private key.
* Using **PUTTYgen** load the .pem key downloaded and select save private key
* Use the public IP provided in the EC2 dashboard in the hostname or IP address of PUTTY configuration
* In connection->SSH->Auth upload the private key generated using the PUTTYgen
* This opens the console of the ubuntu remote system
* Check for the java version which will not be installed in the ubuntu AMI used, using the command java -version
* Using **sudo apt-get install default-jre**install the default version of the jre.
* In WinSCP paste the public IP of the EC2 instance and upload the private key in the advanced section
* Drag and drop the jar file from the Spring Boot application



**Text

Description automatically generated**

* This creates a jar file in the remote system by default. An S3 bucket does the same, where we create a bucket and upload the jar file inside the bucket. The bucket contents can be used in the remote system using the command **wget**
* Use the command *java* **-jar <name of the jar>** to run the application with the jar file uploaded.
* Copy the Public DNS of the EC2 instance and open the same in a browser followed by the port number of the springboot application

****

**Conclusion:**

Thus, the Jenkins has been used to set up a CI/CD pipeline which builds periodically and, on each push/commit. The spring-boot application has been deployed to an EC2 instance and the procedure has been explained.