Windows User

[Draw your reader in with an engaging abstract. It is typically a short summary of the document. When you’re ready to add your content, just click here and start typing.]

[Document subtitle]

INTRODUCTION

DEVOPS

DevOps is a set of software development practices that combine [software development](https://en.wikipedia.org/wiki/Software_development) (Dev) and [information technology operations](https://en.wikipedia.org/wiki/Information_technology_operations) (Ops) to shorten the [systems development life cycle](https://en.wikipedia.org/wiki/Systems_development_life_cycle) while [delivering features, fixes, and updates frequently](https://en.wikipedia.org/wiki/Continuous_delivery) in close alignment with business objectives.

Reflective of key aspects of the development and delivery process:

1. Coding – code development and review, [source code management](https://en.wikipedia.org/wiki/Version_control) tools, code merging
2. Building – [continuous integration](https://en.wikipedia.org/wiki/Continuous_integration) tools, build status
3. Testing – [continuous testing](https://en.wikipedia.org/wiki/Continuous_testing) tools that provide quick and timely feedback on business risks
4. Packaging – [artifact repository](https://en.wikipedia.org/wiki/Binary_repository_manager" \o "Binary repository manager), application pre-deployment staging
5. Releasing – change management, release approvals, [release automation](https://en.wikipedia.org/wiki/Application_release_automation)
6. Configuring – infrastructure configuration and management, [infrastructure as code](https://en.wikipedia.org/wiki/Infrastructure_as_code) tools
7. Monitoring – [applications performance monitoring](https://en.wikipedia.org/wiki/Application_performance_management), end-user experience

GIT

Git is a [distributed version-control](https://en.wikipedia.org/wiki/Distributed_version_control) system for tracking changes in [source code](https://en.wikipedia.org/wiki/Source_code) during [software development](https://en.wikipedia.org/wiki/Software_development). It is designed for coordinating work among [programmers](https://en.wikipedia.org/wiki/Programmer), but it can be used to track changes in any set of [files](https://en.wikipedia.org/wiki/Computer_file). Its goals include speed, [data integrity](https://en.wikipedia.org/wiki/Data_integrity), and support for distributed, non-linear workflows. We will do the coding on the git.

JENKINS

Jenkins is an [open source](https://en.wikipedia.org/wiki/Open-source_software) automation server written in [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). Jenkins helps to automate the non-human part of the [software development](https://en.wikipedia.org/wiki/Software_development) process, with [continuous integration](https://en.wikipedia.org/wiki/Continuous_integration) and facilitating technical aspects of [continuous delivery](https://en.wikipedia.org/wiki/Continuous_delivery). It is a server-based system that runs in [servlet containers](https://en.wikipedia.org/wiki/Java_Servlet#Container_servers) such as [Apache Tomcat](https://en.wikipedia.org/wiki/Apache_Tomcat). It supports [version control](https://en.wikipedia.org/wiki/Version_control) tools. Jenkins is [free software](https://en.wikipedia.org/wiki/Free_software). [Builds](https://en.wikipedia.org/wiki/Software_build) can be triggered by various means, for example by [commit](https://en.wikipedia.org/wiki/Commit_(version_control)) in a version control system, by scheduling via a [cron](https://en.wikipedia.org/wiki/Cron" \o "Cron)-like mechanism and by requesting a specific build [URL](https://en.wikipedia.org/wiki/Uniform_Resource_Locator). It can also be triggered after the other builds in the queue have completed. Jenkins functionality can be extended with [plugins](https://en.wikipedia.org/wiki/Plug-in_(computing)). The jobs from Code compiling to Code Packing is done in Jenkins.

CONTINOUS INTEGRATION

Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. The CI process is comprised of automatic tools that assert the new code’s correctness before integration. A source code version control system is the crux of the CI process. The version control system is also supplemented with other checks like automated code quality tests, syntax style review tools, and more.

CONTINUOUS DEPLOYMENT

Continuous deployment (CD) is a [software engineering approach](https://en.wikipedia.org/wiki/Software_development_process) in which software functionalities are delivered frequently through automated [deployments](https://en.wikipedia.org/wiki/Software_deployment).[[1]](https://en.wikipedia.org/wiki/Continuous_deployment#cite_note-1)[[2]](https://en.wikipedia.org/wiki/Continuous_deployment#cite_note-2)[[3]](https://en.wikipedia.org/wiki/Continuous_deployment#cite_note-3)CD contrasts with [continuous delivery](https://en.wikipedia.org/wiki/Continuous_delivery), a similar approach in which software functionalities are also frequently delivered and deemed to be potentially capable of being deployed but are actually not deployed.

DOCKER

Docker is a set of coupled [software-as-a-service](https://en.wikipedia.org/wiki/Software-as-a-service) and [platform-as-a-service](https://en.wikipedia.org/wiki/Platform-as-a-service) products that use [operating-system-level virtualization](https://en.wikipedia.org/wiki/Operating-system-level_virtualization) to develop and deliver software in packages called [containers](https://en.wikipedia.org/wiki/Container_(virtualization)). The software that hosts the containers is called Docker Engine.

PUPPET

Puppet is an [open-core](https://en.wikipedia.org/wiki/Open-core_model) [software configuration management](https://en.wikipedia.org/wiki/Software_configuration_management) tool. It runs on many [Unix-like](https://en.wikipedia.org/wiki/Unix-like) systems as well as on [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), and includes its own [declarative language](https://en.wikipedia.org/wiki/Declarative_language) to describe [system configuration](https://en.wikipedia.org/wiki/System_configuration).

ANSIBLE

Ansible is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) software provisioning, configuration management, and [application-deployment](https://en.wikipedia.org/wiki/Application_deployment) tool.[[2]](https://en.wikipedia.org/wiki/Ansible_(software)#cite_note-2) It runs on many [Unix-like](https://en.wikipedia.org/wiki/Unix-like) systems, and can configure both Unix-like systems as well as [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows). It includes its own [declarative language](https://en.wikipedia.org/wiki/Declarative_language) to describe [system configuration](https://en.wikipedia.org/wiki/System_configuration).

SELENIUM

Selenium is a portable [framework](https://en.wikipedia.org/wiki/Software_framework) for [testing](https://en.wikipedia.org/wiki/Software_testing) [web applications](https://en.wikipedia.org/wiki/Web_application). Selenium provides a playback (formerly also recording) tool for authoring [functional tests](https://en.wikipedia.org/wiki/Functional_testing) without the need to learn a test [scripting language](https://en.wikipedia.org/wiki/Scripting_language) (Selenium IDE).

We need to follow these steps:

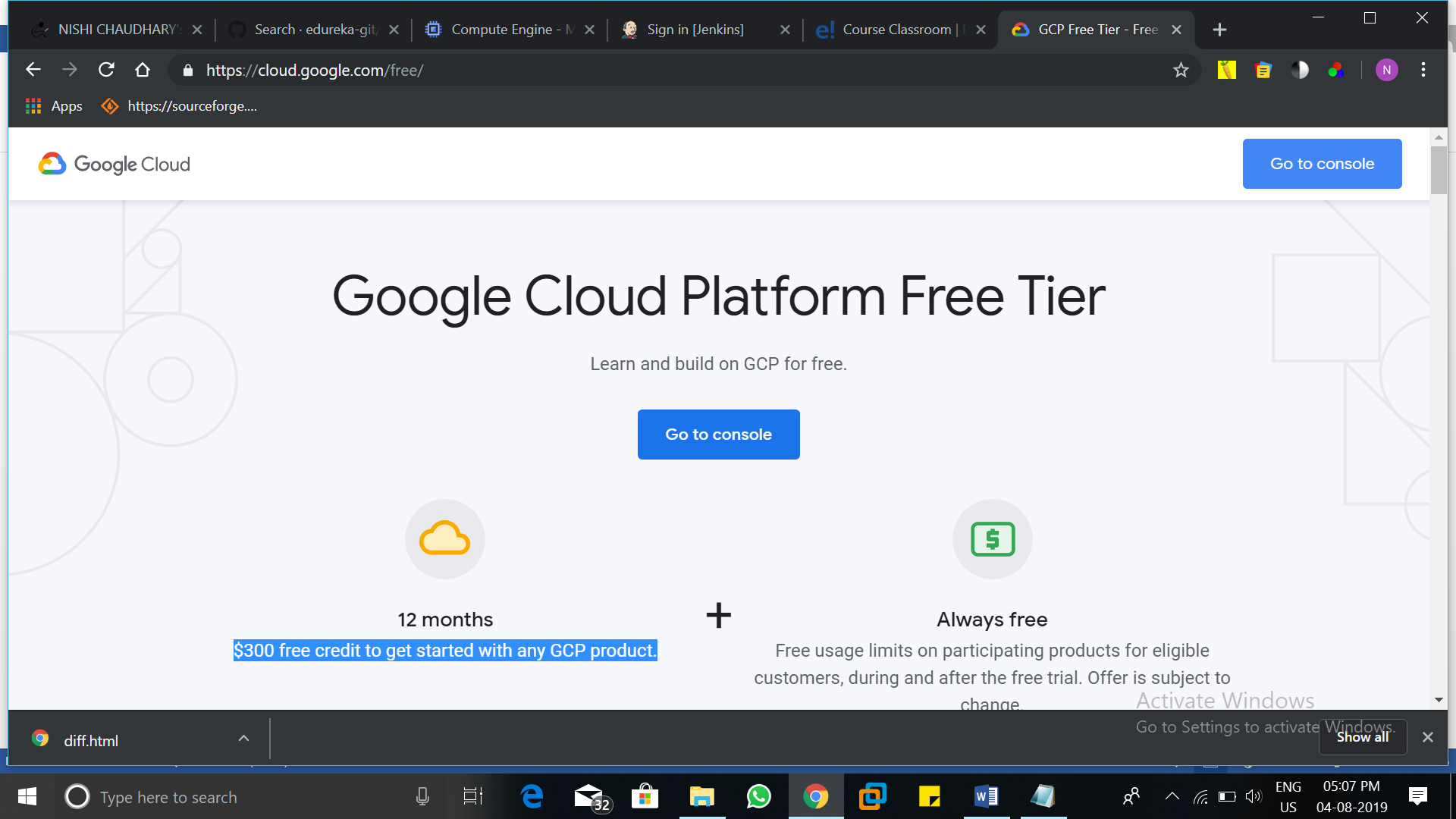
**Creating Account on GIT**

Take a Computer system or a laptop for the operations. Go to github.com, create your account and start using it by creating repositories and committing the code there.

Here we are using a git repository <https://github.com/edureka-git/DevOpsClassCodes>. We can fork this repo. to our own account and can then use it.

So, in this scenario we have a java application i.e. Addessbook. Now we need to install Jenkins.

For this we are using Google Cloud Platform (<http://cloud.google.com/free>). It will provide a free trial of the services for a year with $300 free credit to get started with any GCP product.



Go to your console and click on compute engine 🡪 VM instances.

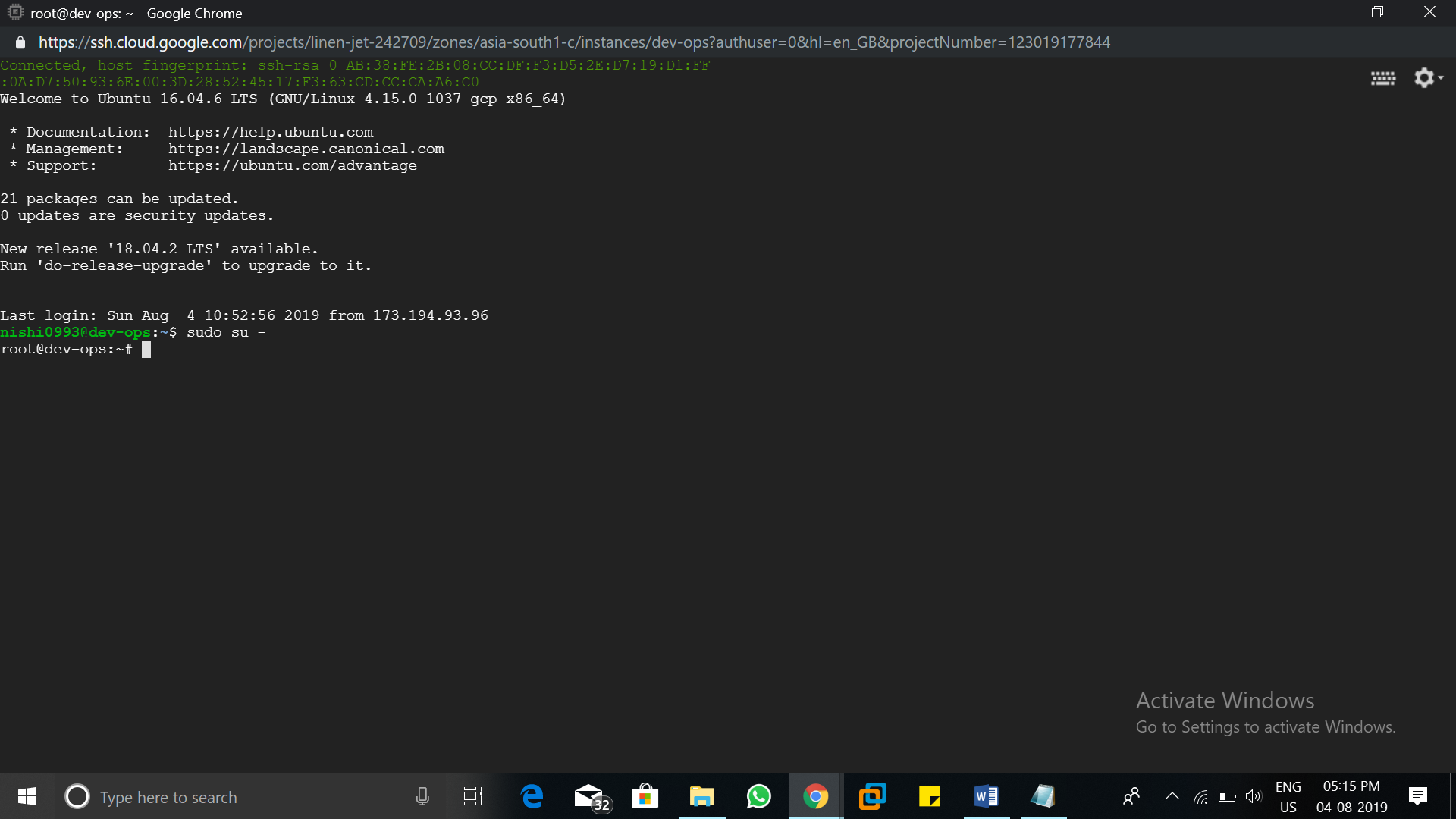
Create a new virtual machine. Add resources according to your needs like, processors, CPUs, RAM etc.

Select the ISO image of Ubuntu 16.04. Select the zone nearby your area for better performance.

We create a virtual instance named dev-ops.

Start the instance and open the ssh window.

Run the command:- sudo du - , to switch to root user.



**Installing JENKINS**

To install latest version of Jenkins follow the following guide: <https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+on+Ubuntu>

Run the following commands on your Ubuntu server

**🡪wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -**

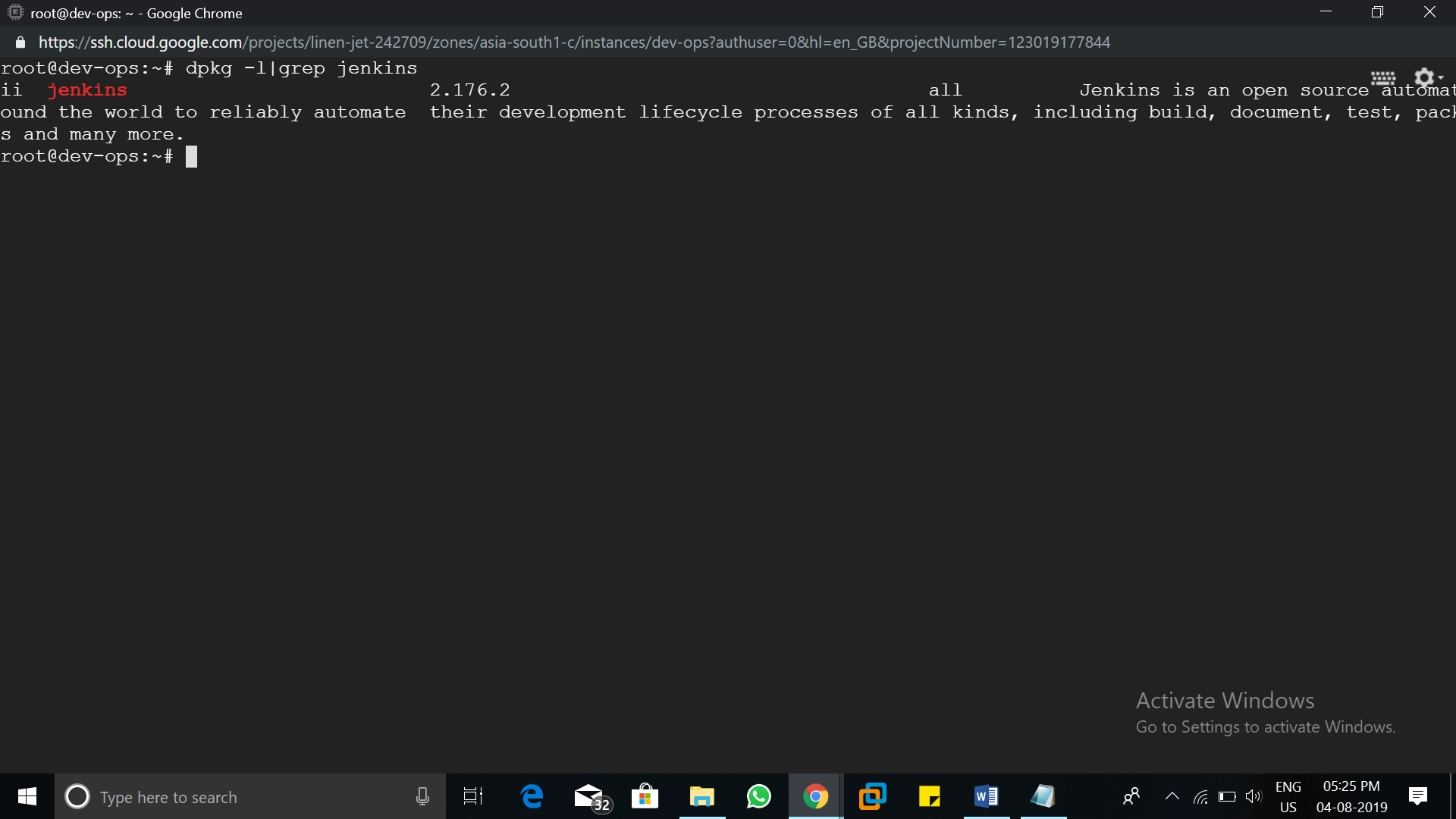
**🡪sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'**

**🡪sudo apt-get update**

**🡪 sudo apt-get install jenkins**

And now if all goes correct, Jenkins will be installed.

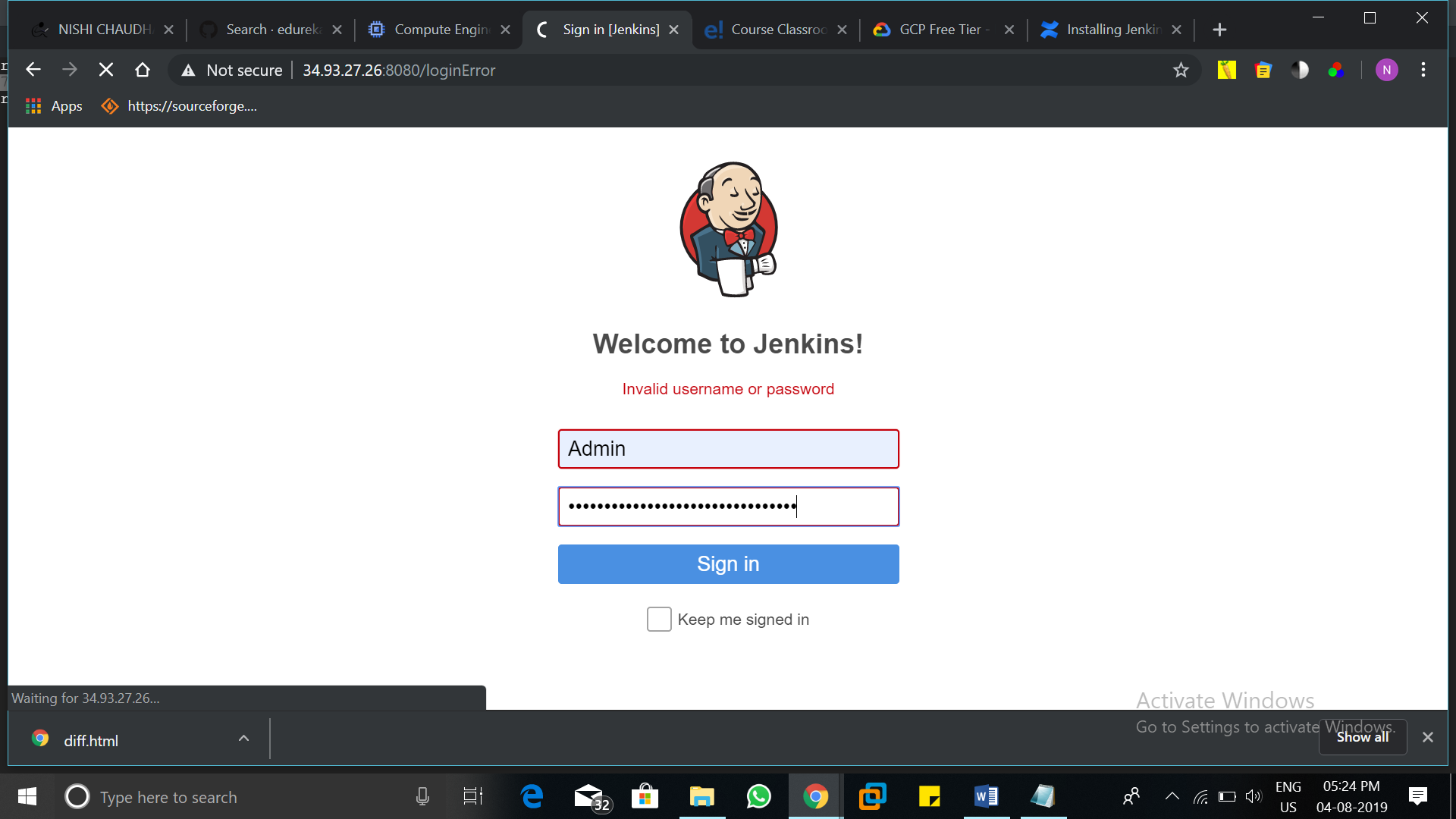
Check the version of Jenkins to confirm the installation



Now copy your google server public IP address and paste it on a new tab with :8080(as Jenkins is a web application which works on 8080 port).

Then, it need a password, just go to your server and type the path say:- cat /var/lib/Jenkins/secrets/initialAdminpassword

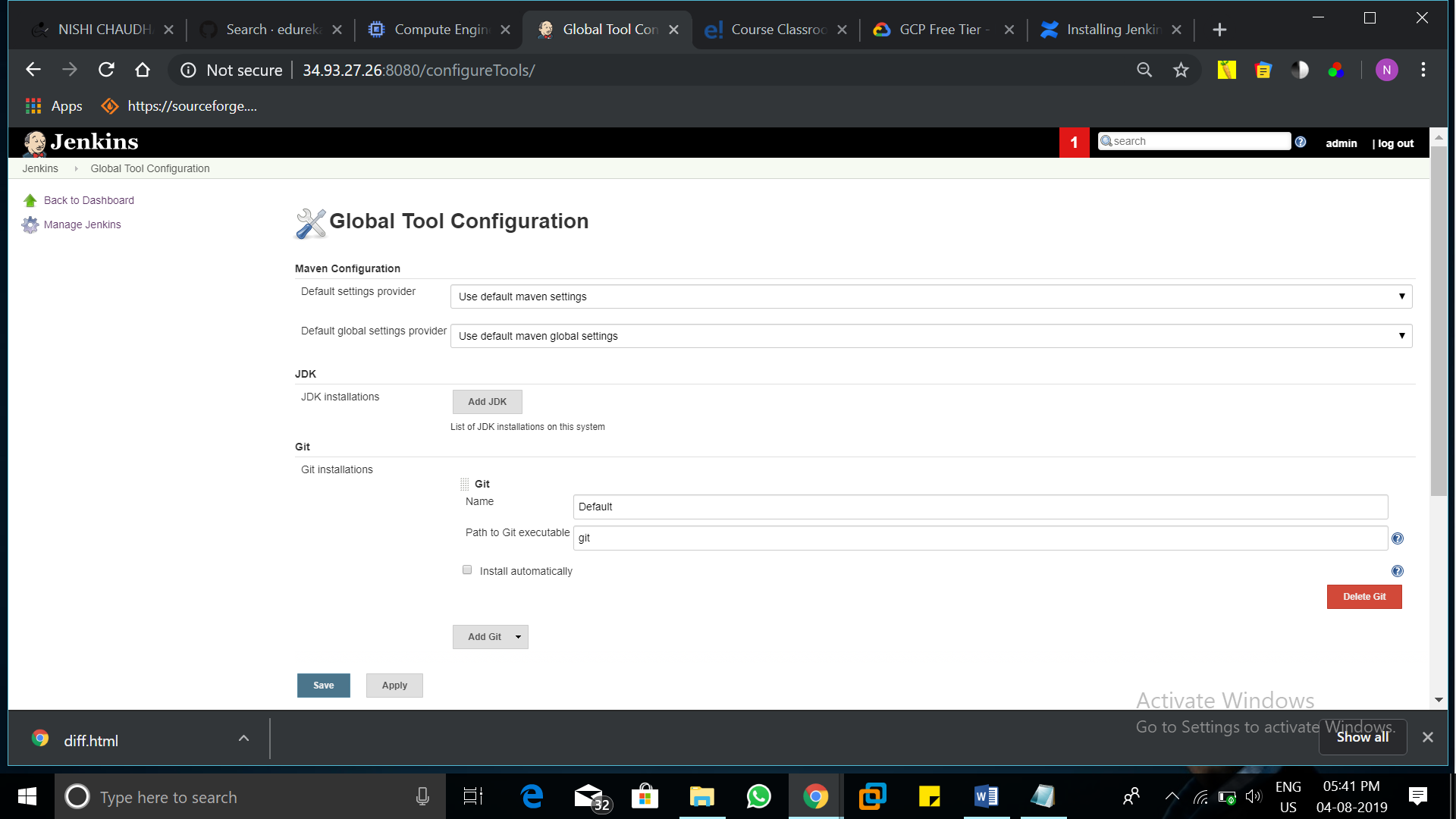
Paste the password on your Jenkins pge and install the Jenkins, login as Admin.



**Starting with JENKINS**

First we need to install maven in the Jenkins because we are going to package a java application.

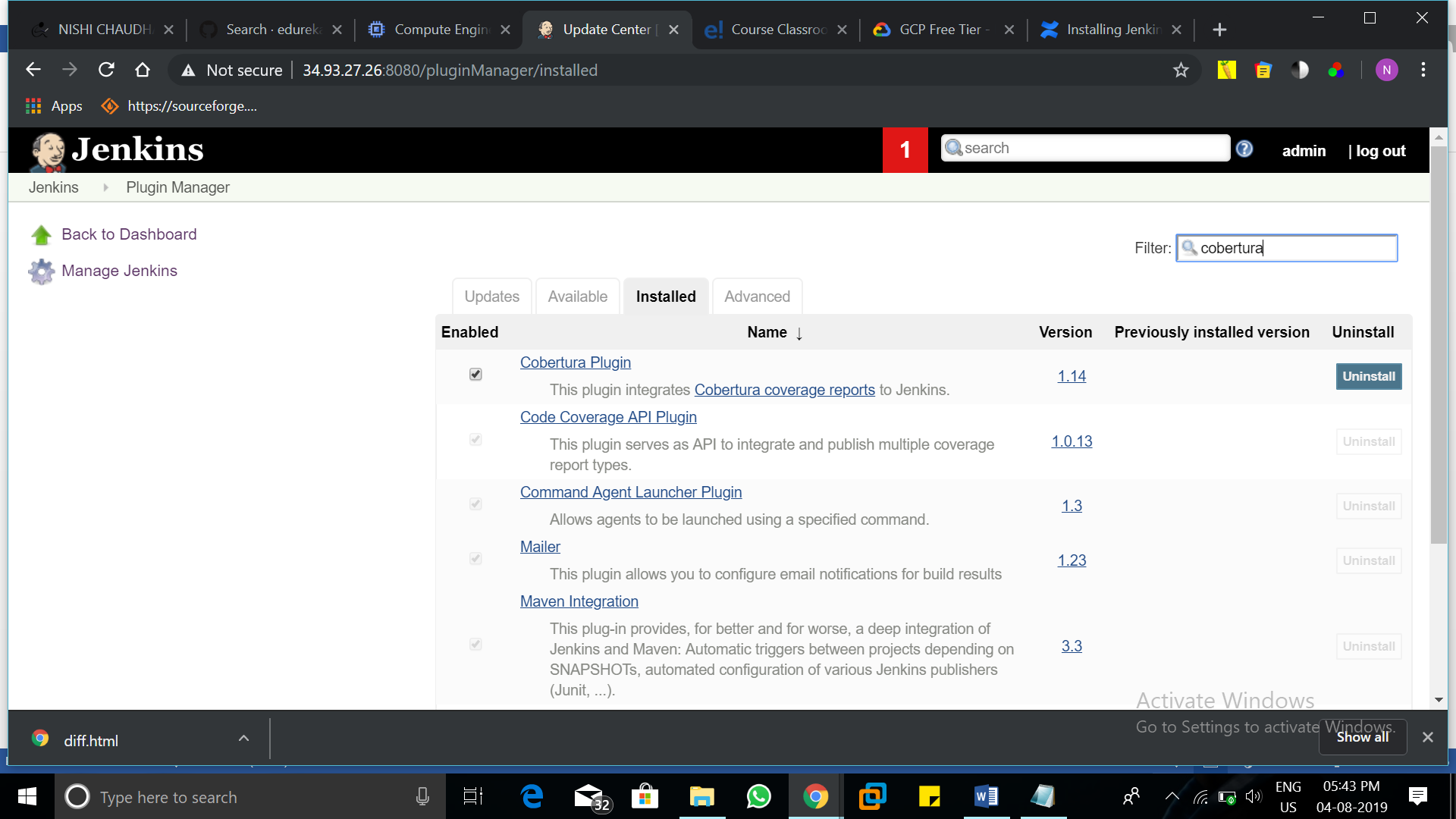
So, go to manage Jenkins🡪Global Tool Configuration



Click on maven installations and give a name to your maven say, MyMaven and click on save.



Also in Manage Jenkins, go to Manage Plugins and install plugin named Cobertura and PMD.



**Creating the jobs**

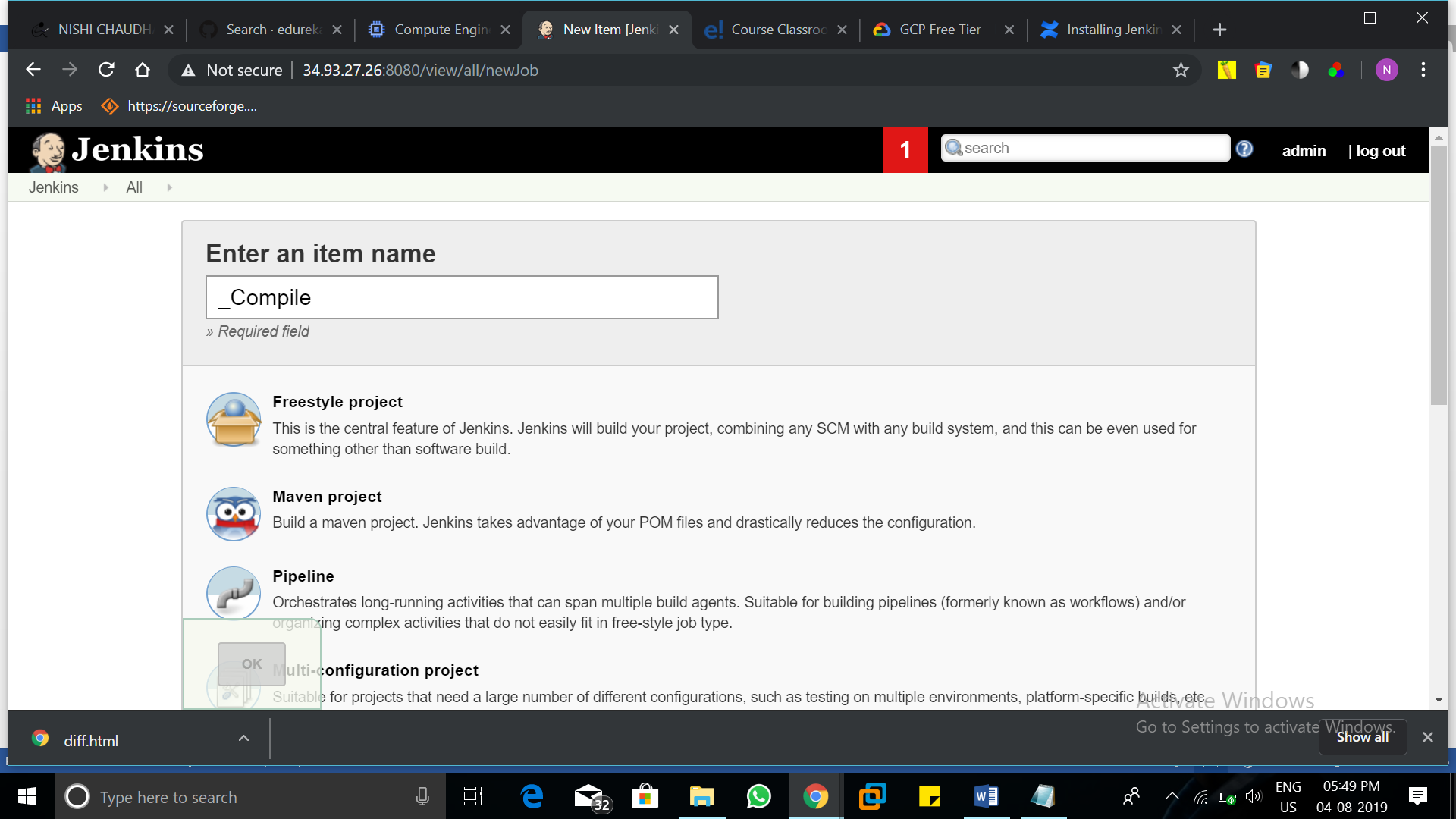
Giving an overview about what we need to di in Jenkins now,

1. Compile the java code🡪 2.Review the code 🡪 3.Test the code 🡪 4.Generate a coverage report 🡪 5.Package the application.

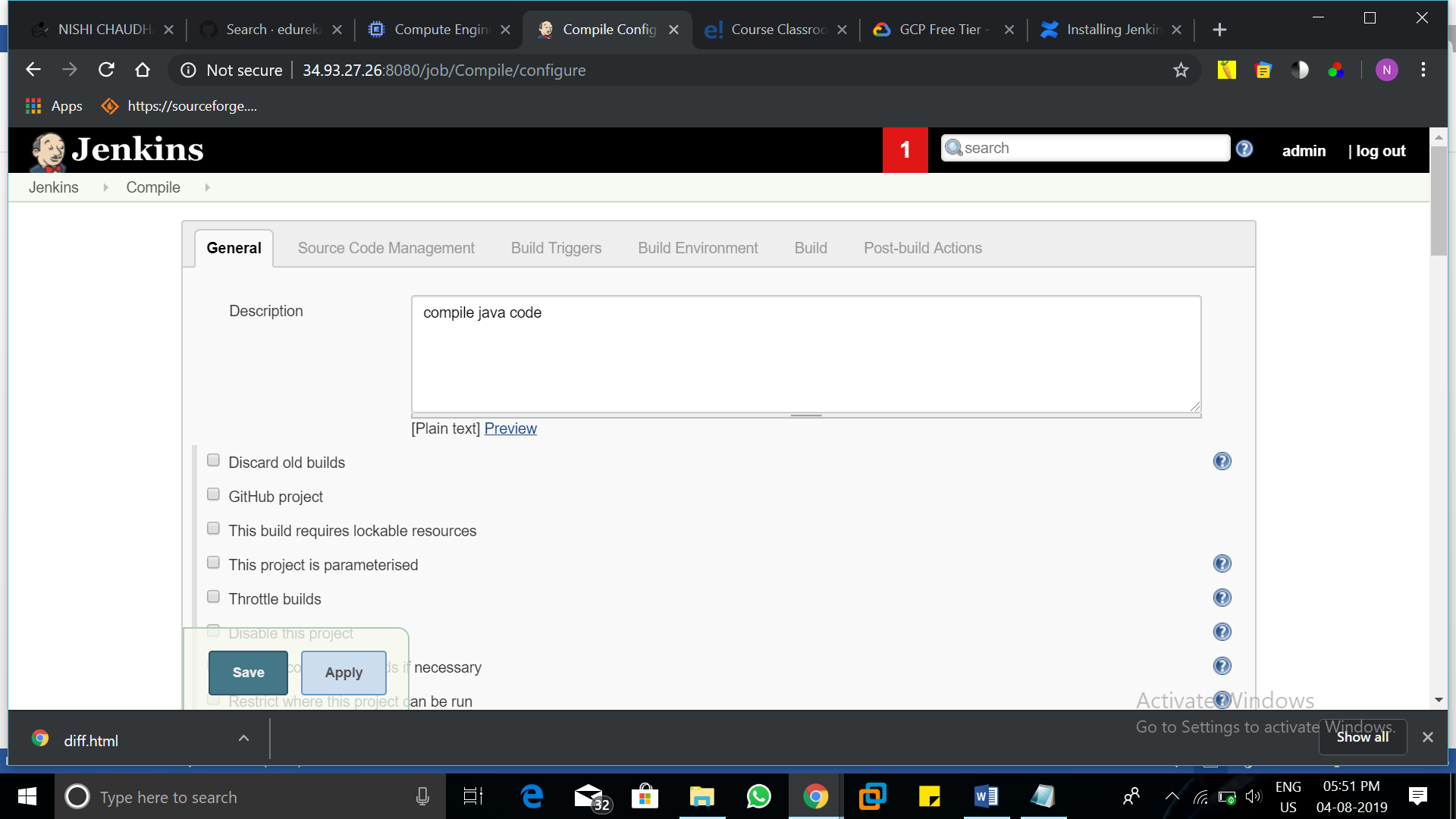
**1. Compile the java code**

First, come to the Jenkins dashboard, and click on new item

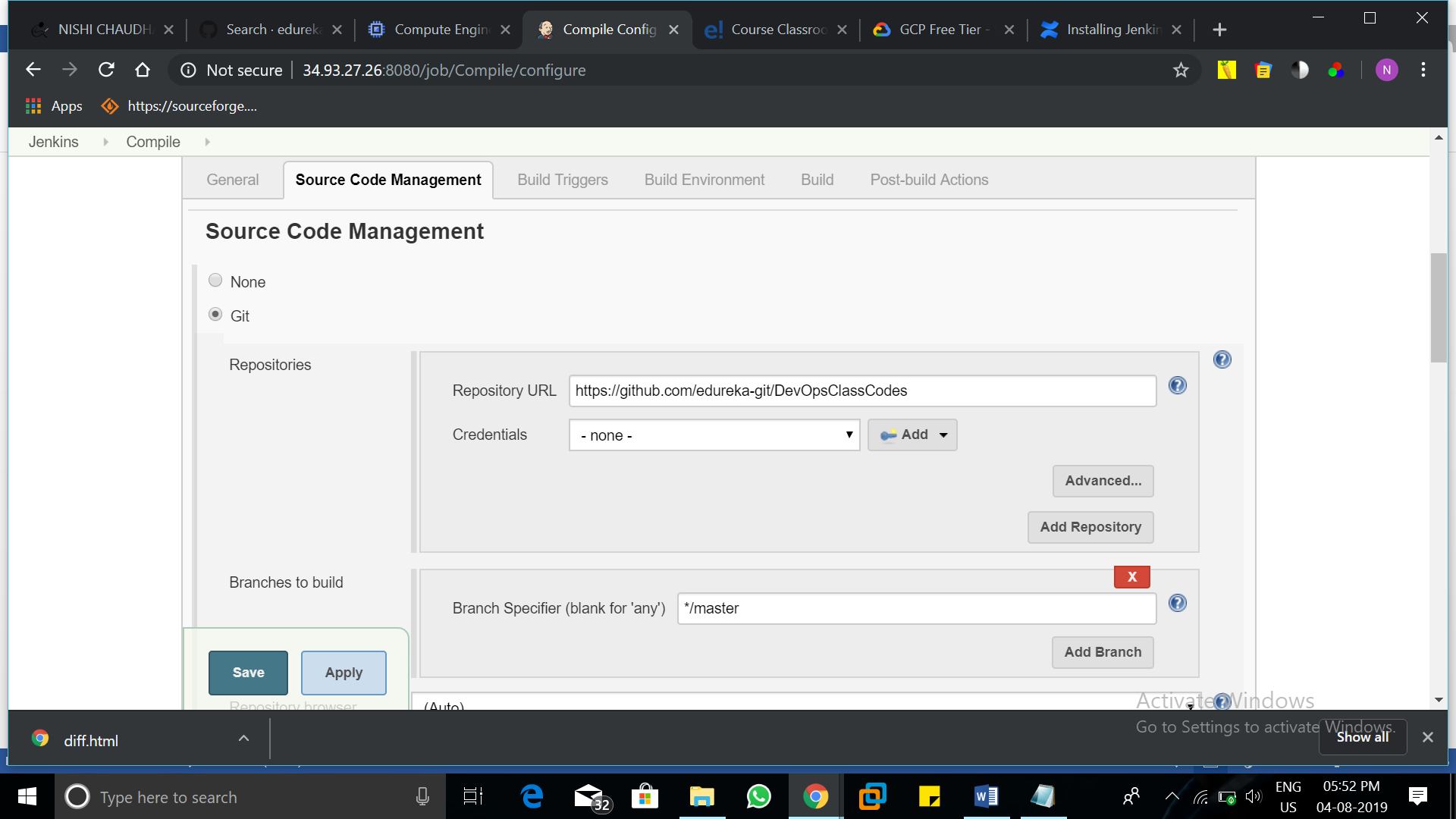
Name your job and select freestyle project and click on OK



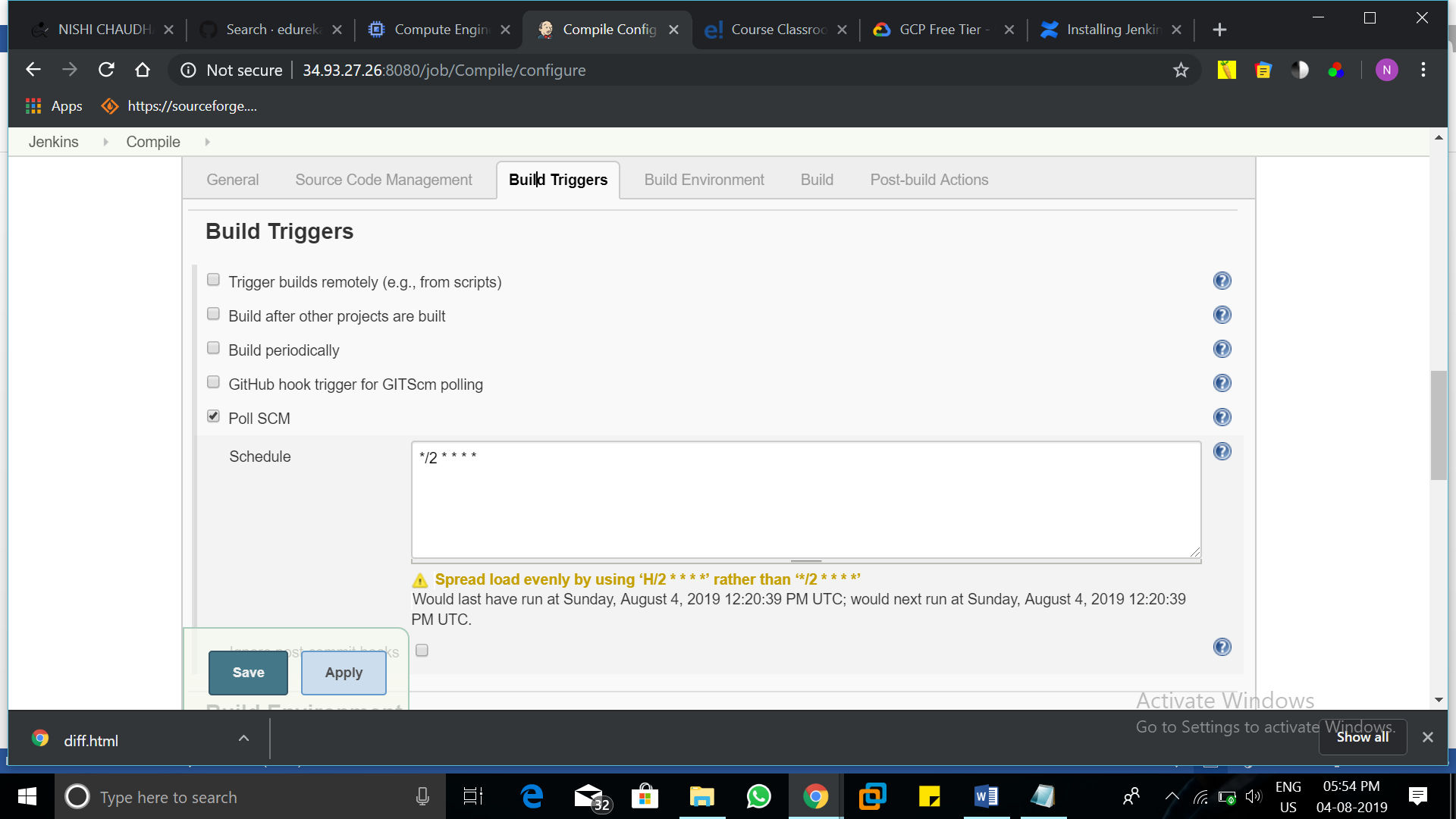
In General, give a proper description about the project.



In Source Code Management, select the option git and copy the git repo we mentioned above in the repository URL.

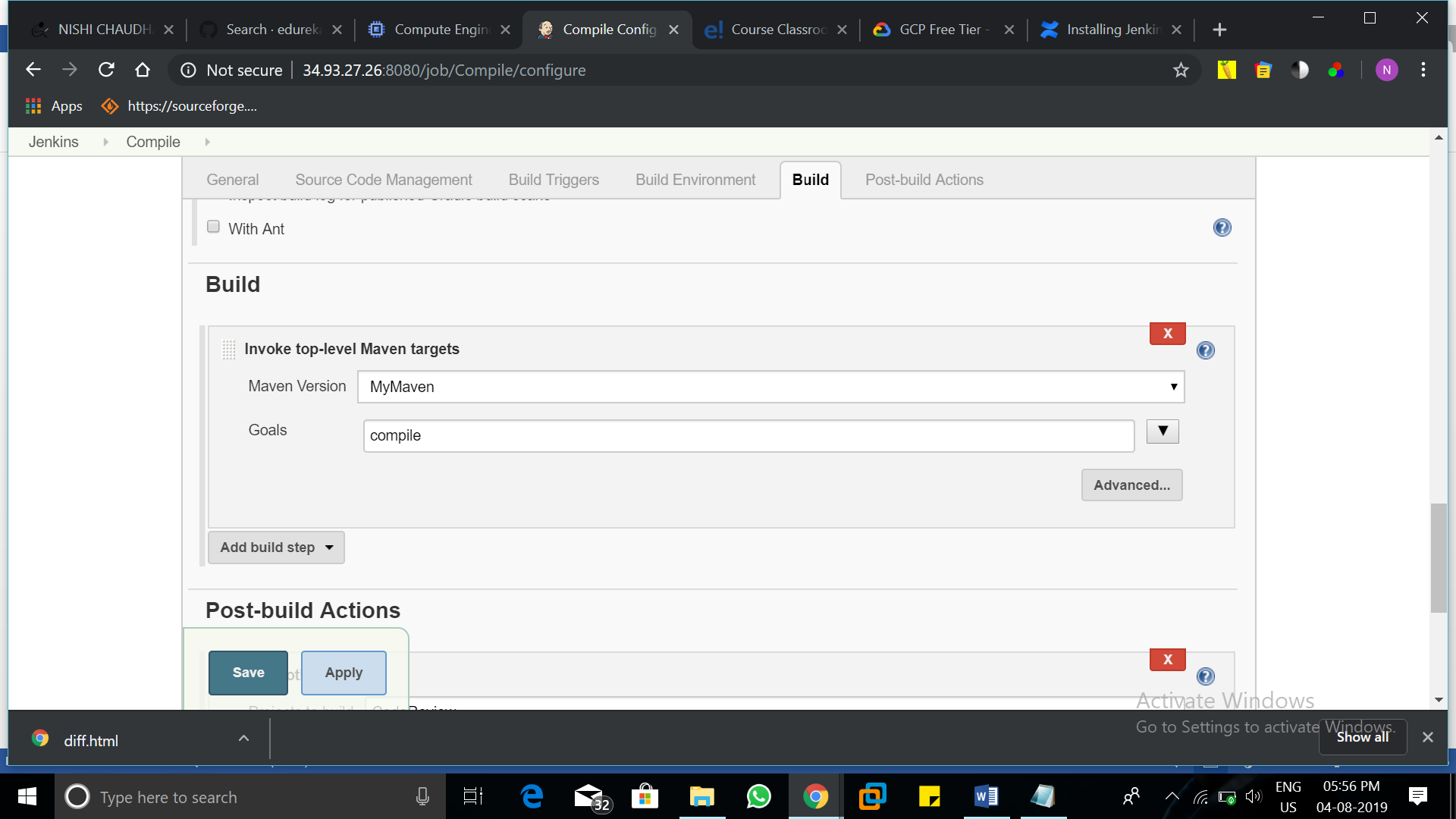


In Build Triggers, select Poll SCM for triggering the job whenever there is a anew commit on the git repository.



Skip build environments for now. In Build, select the top level maven command.

Select the maven you created and in goals type compile.



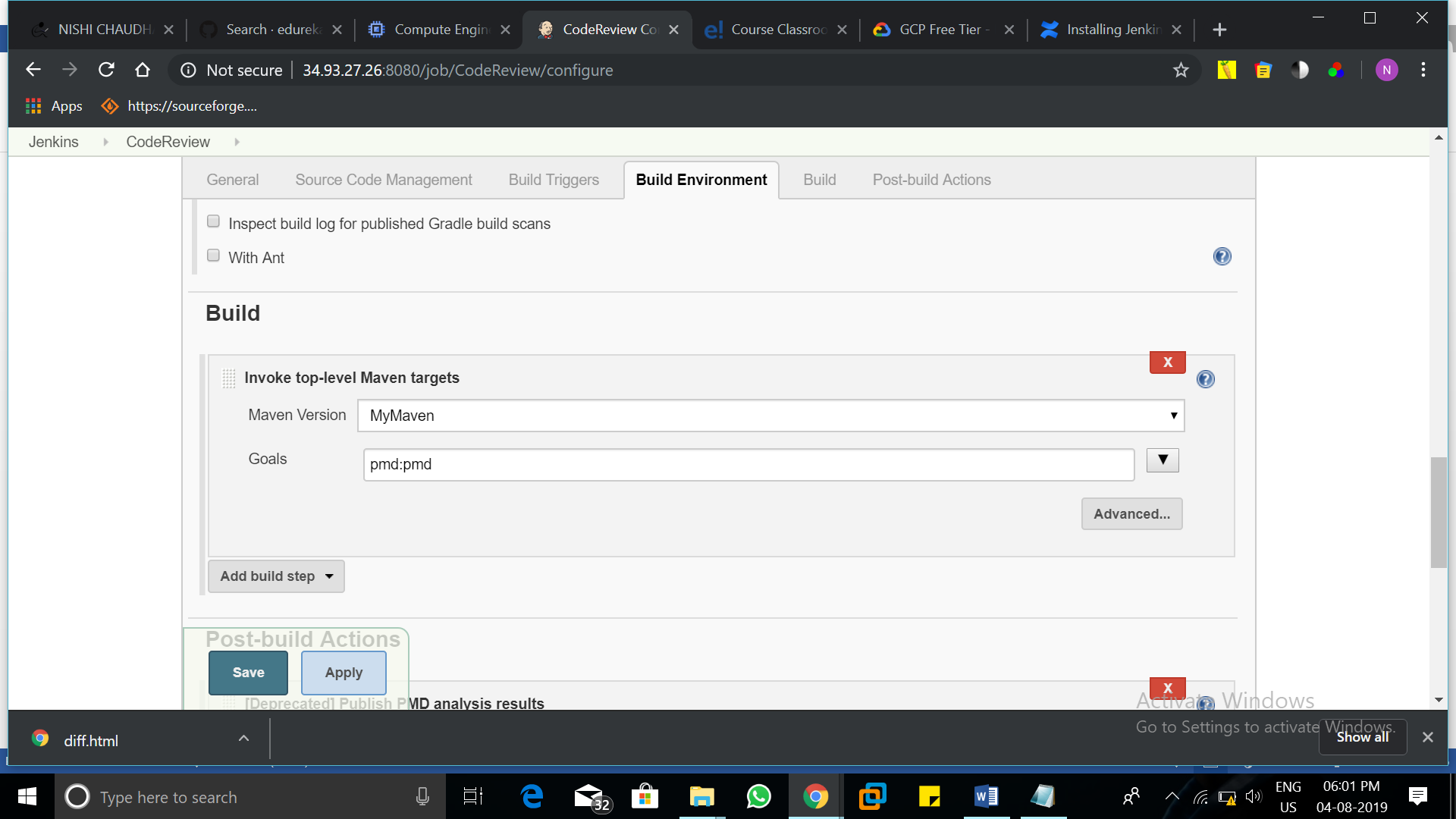
Then, click save and click on build now,

**2.Review the code**

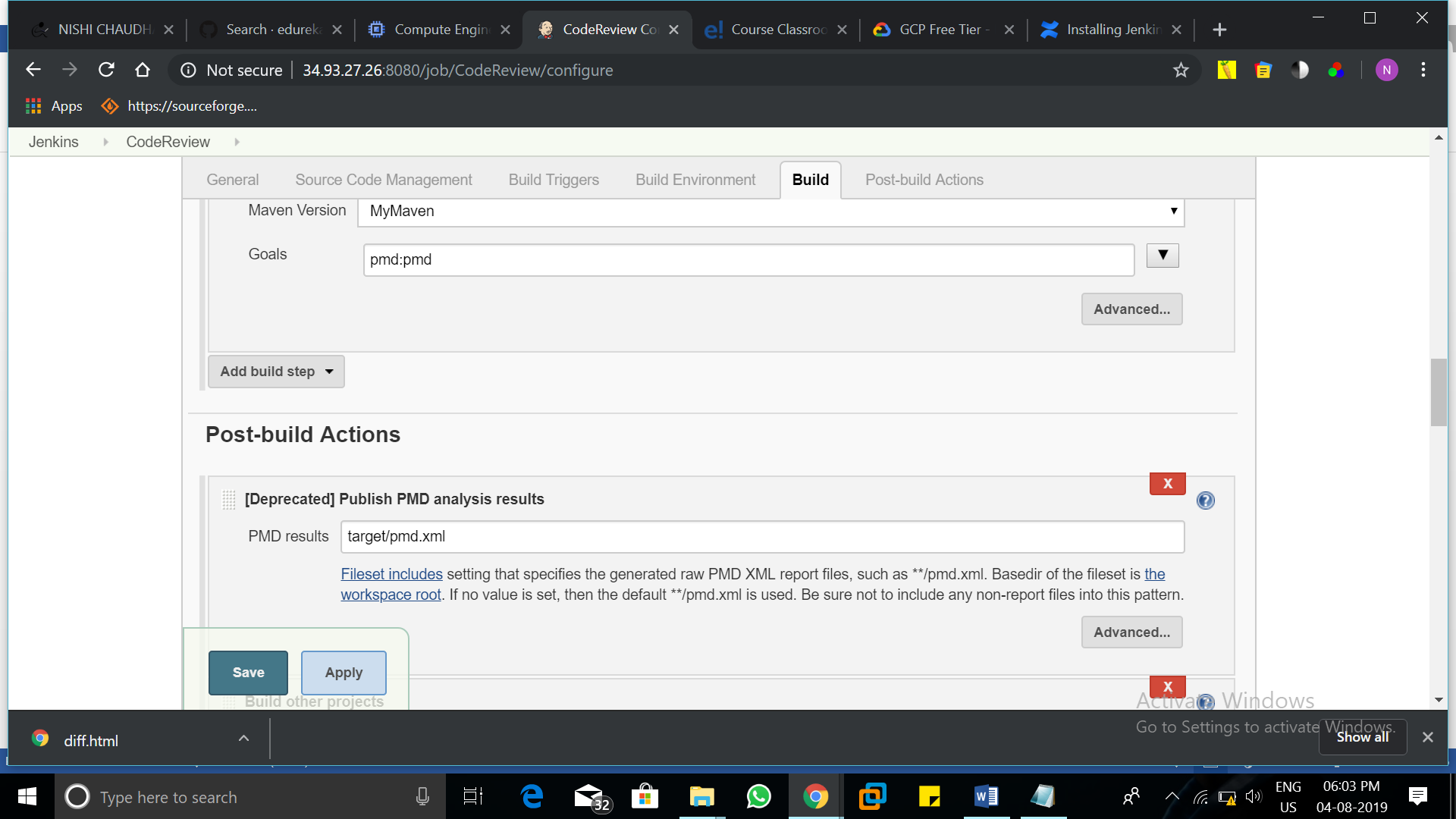
To create the second job again create the new item give a name and select freestyle project.

Give Description, and select git in Source Code Management, and copy the same link.

In Build, select the top level maven command, select the maven you created, and in goals type pmd:pmd



In Post-build Actions, select PMD analysis report and type the path: target/pmd.xml.



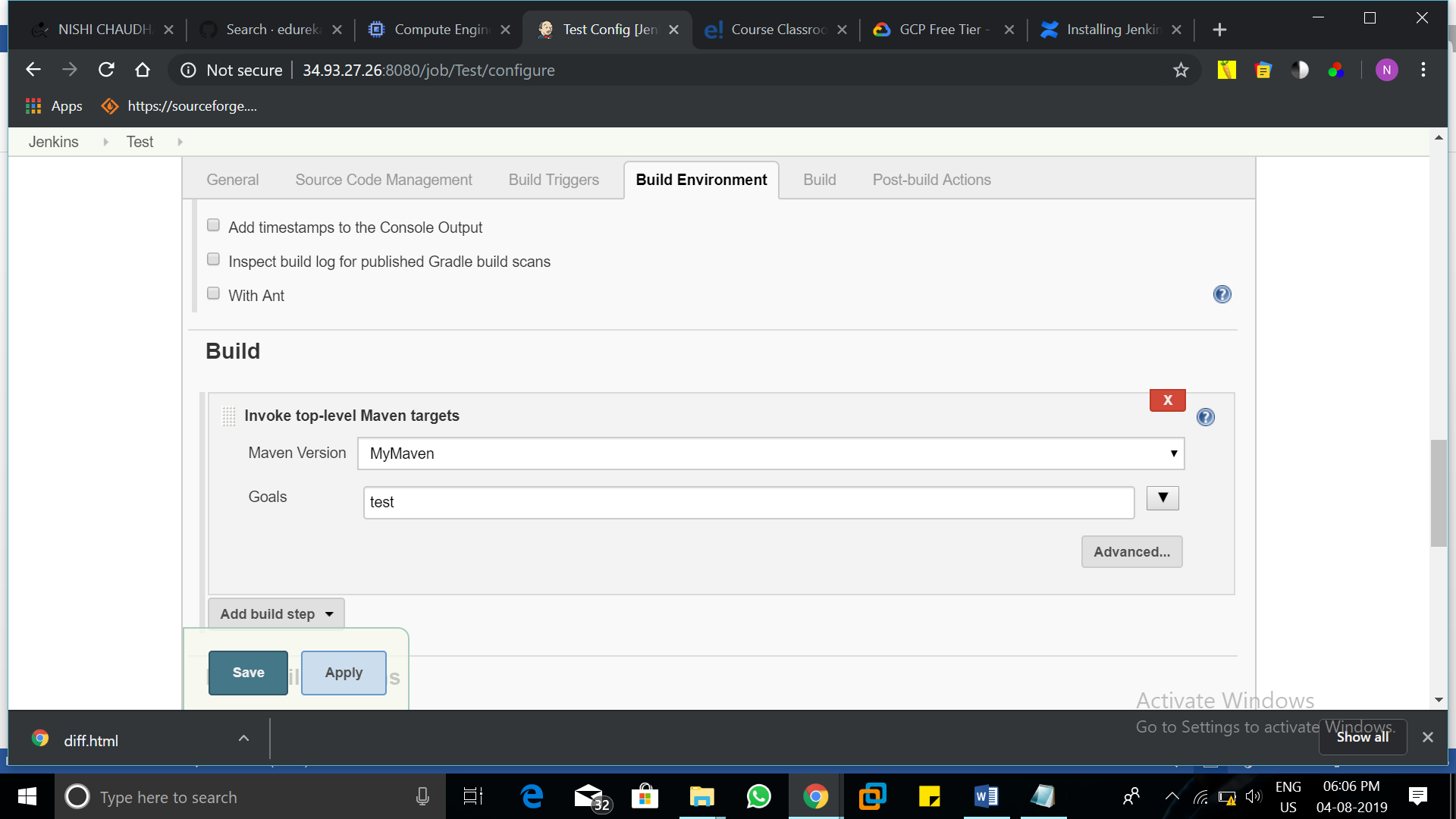
Click on save and build the job.

**3.Test the code**

To create the third job, again create the new item give a name and select freestyle project.

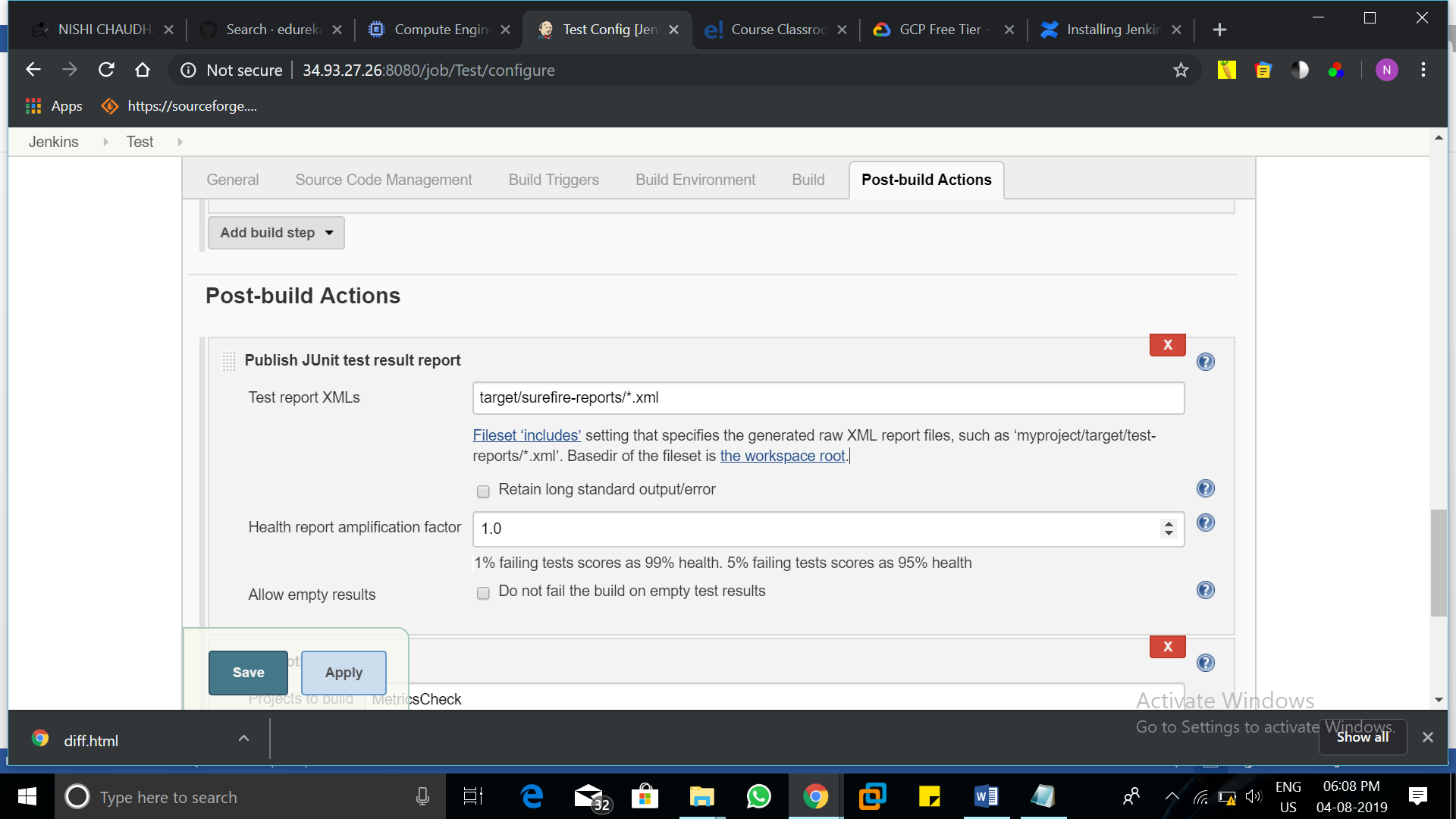
Give Description, and select git in Source Code Management, and copy the same link.

In Build, select the top level maven command, select the maven you created, and in goals type test.



In Post-build Actions, select JUnit test result report.

And give the path: target/surefire-reports/\*.xml



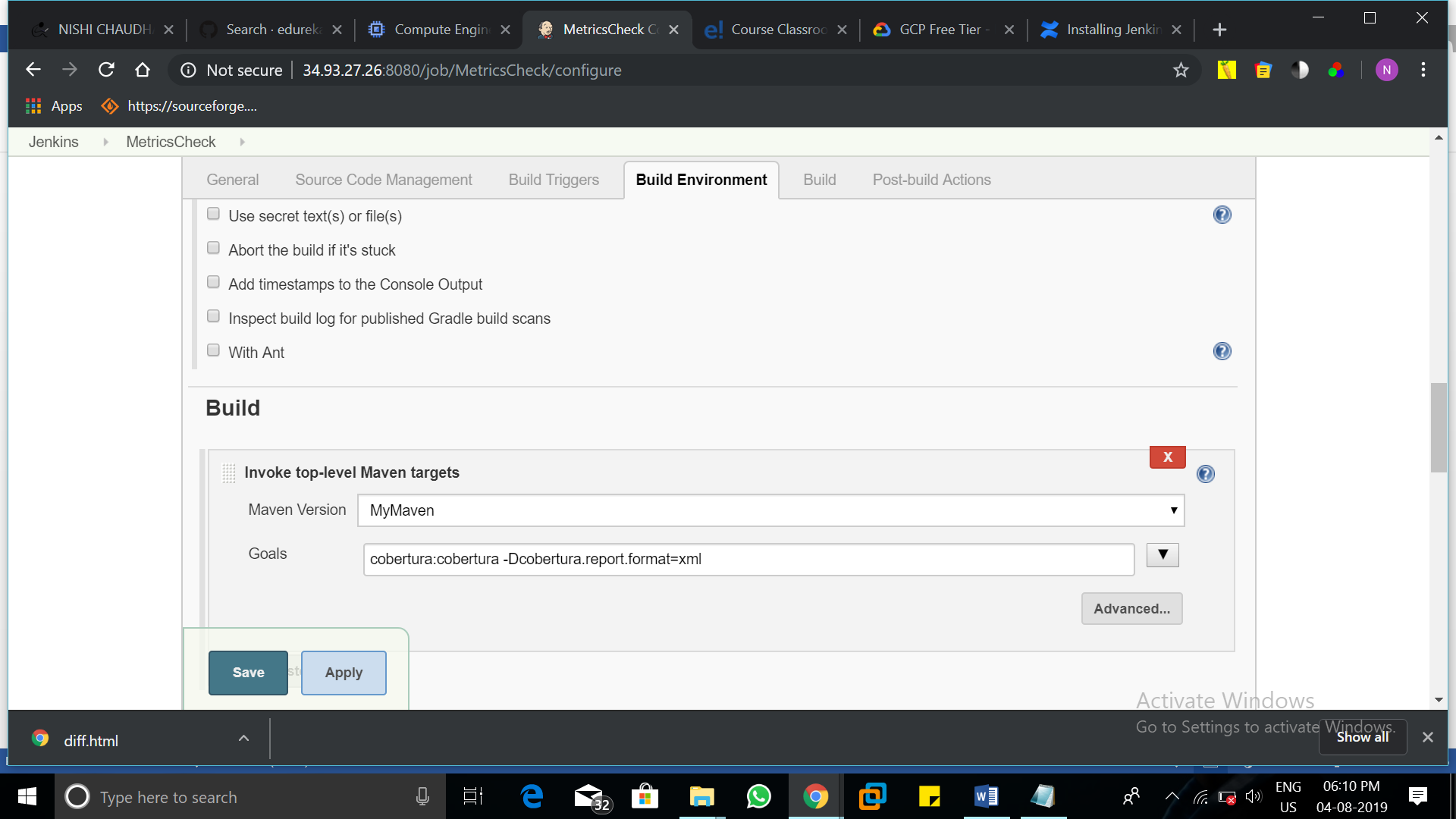
Save the settings, and then build the job.

**4.Generate a coverage report**

To create the fourth job, again create the new item give a name and select freestyle project.

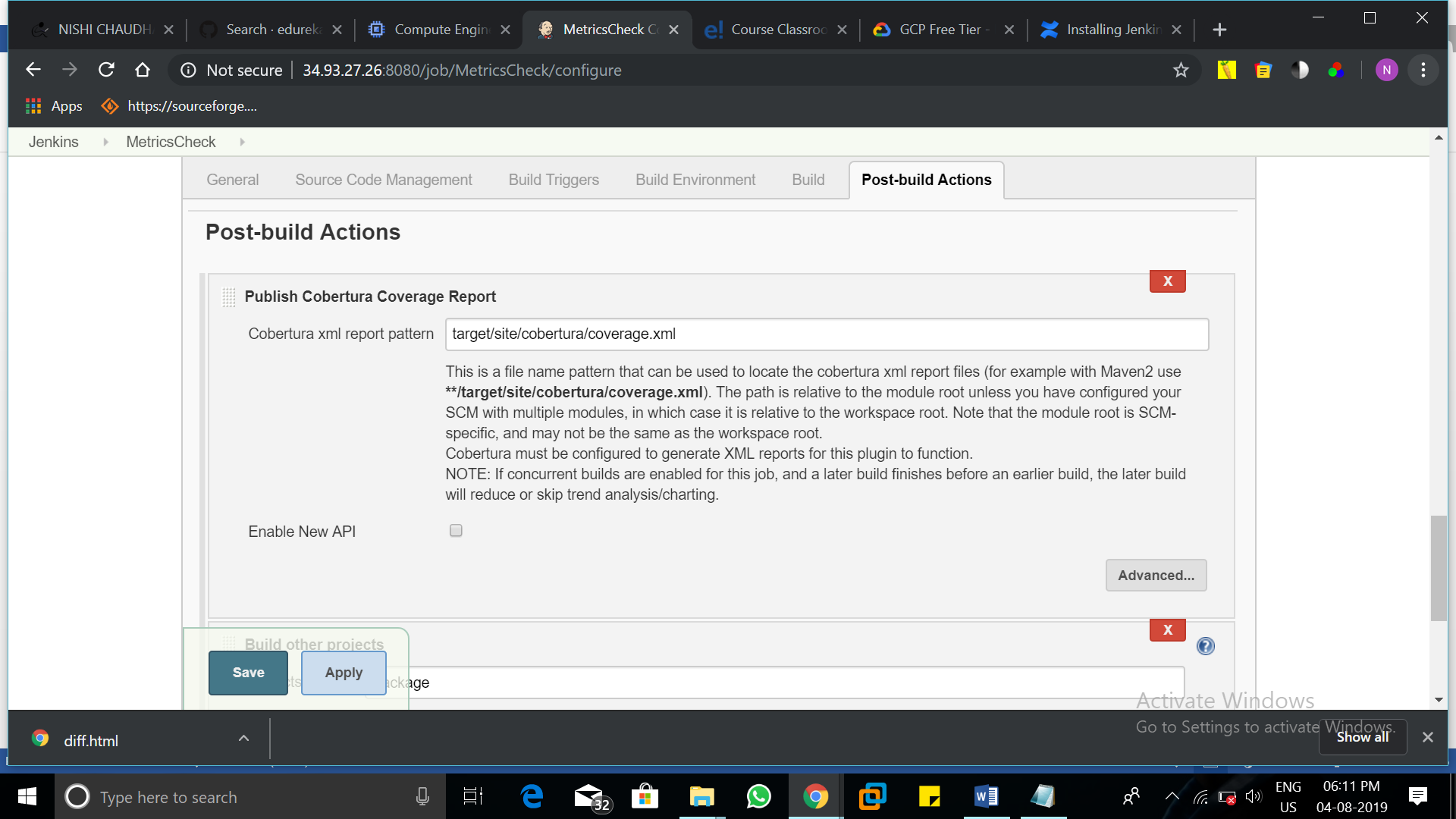
Give Description, and select git in Source Code Management, and copy the same link.

In Build, select the top level maven command, select the maven you created, and in goals type cobertura:cobertura -Dcobertura.report.format=xml.



In Post-build Actions, select Cobertura Coverage Report.

Give the required path mentioned below.



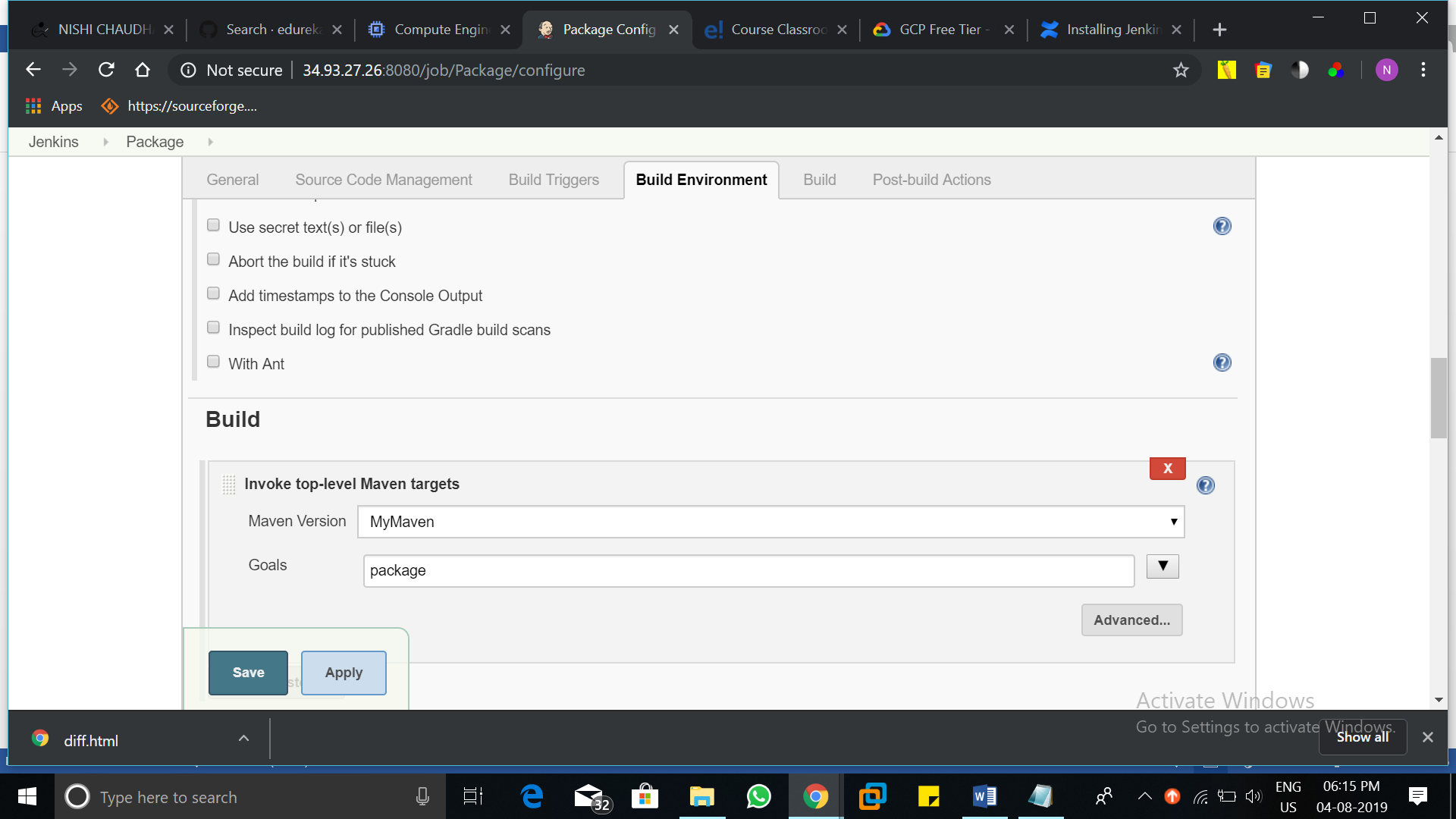
Save the settings and build the job.

**5.Package the application**

To create the fifth job, again create the new item give a name and select freestyle project.

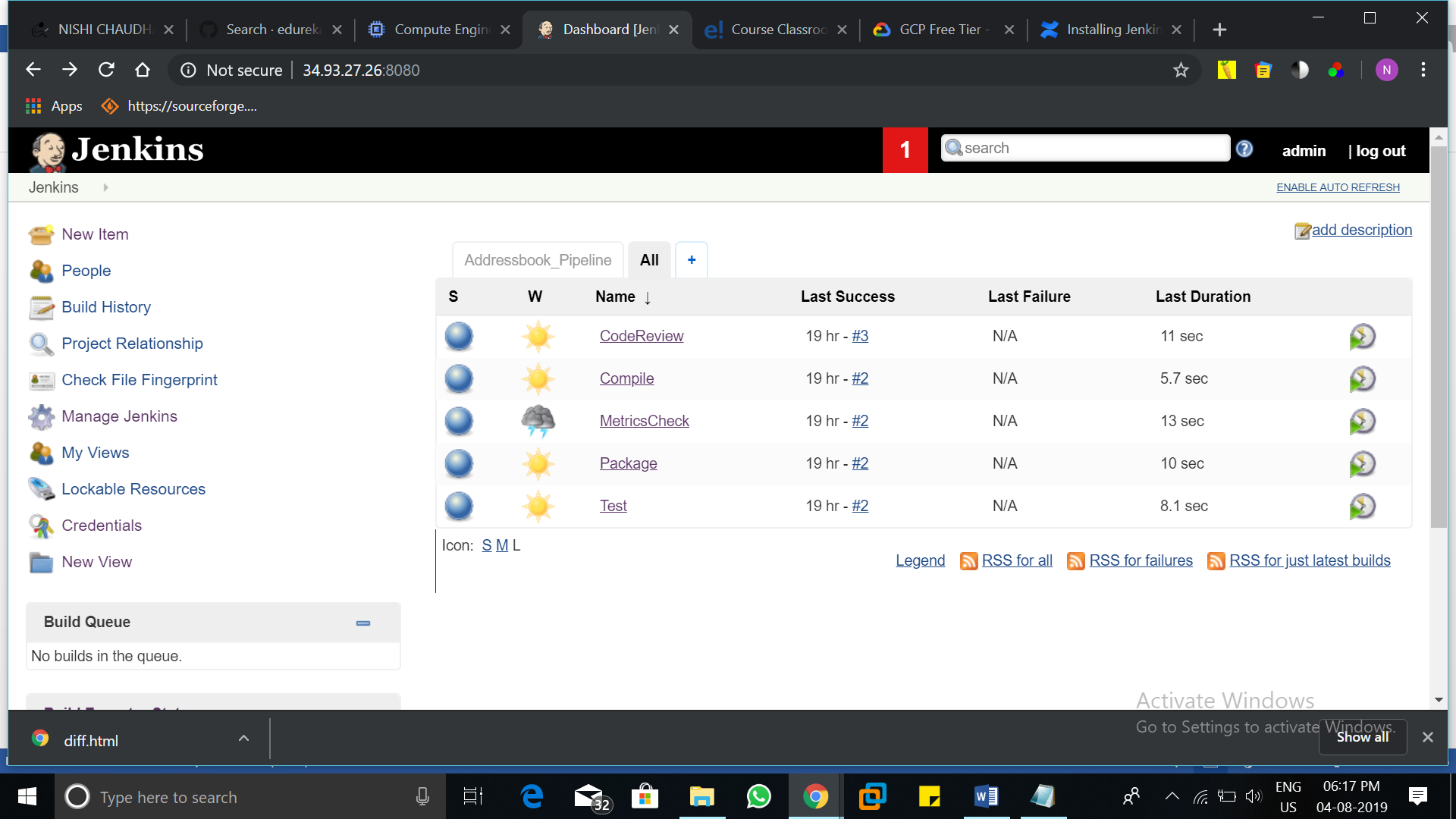
Give Description, and select git in Source Code Management, and copy the same link.

In Build, select the top level maven command, select the maven you created, and in goals type package.



Save the configuration and build the job.

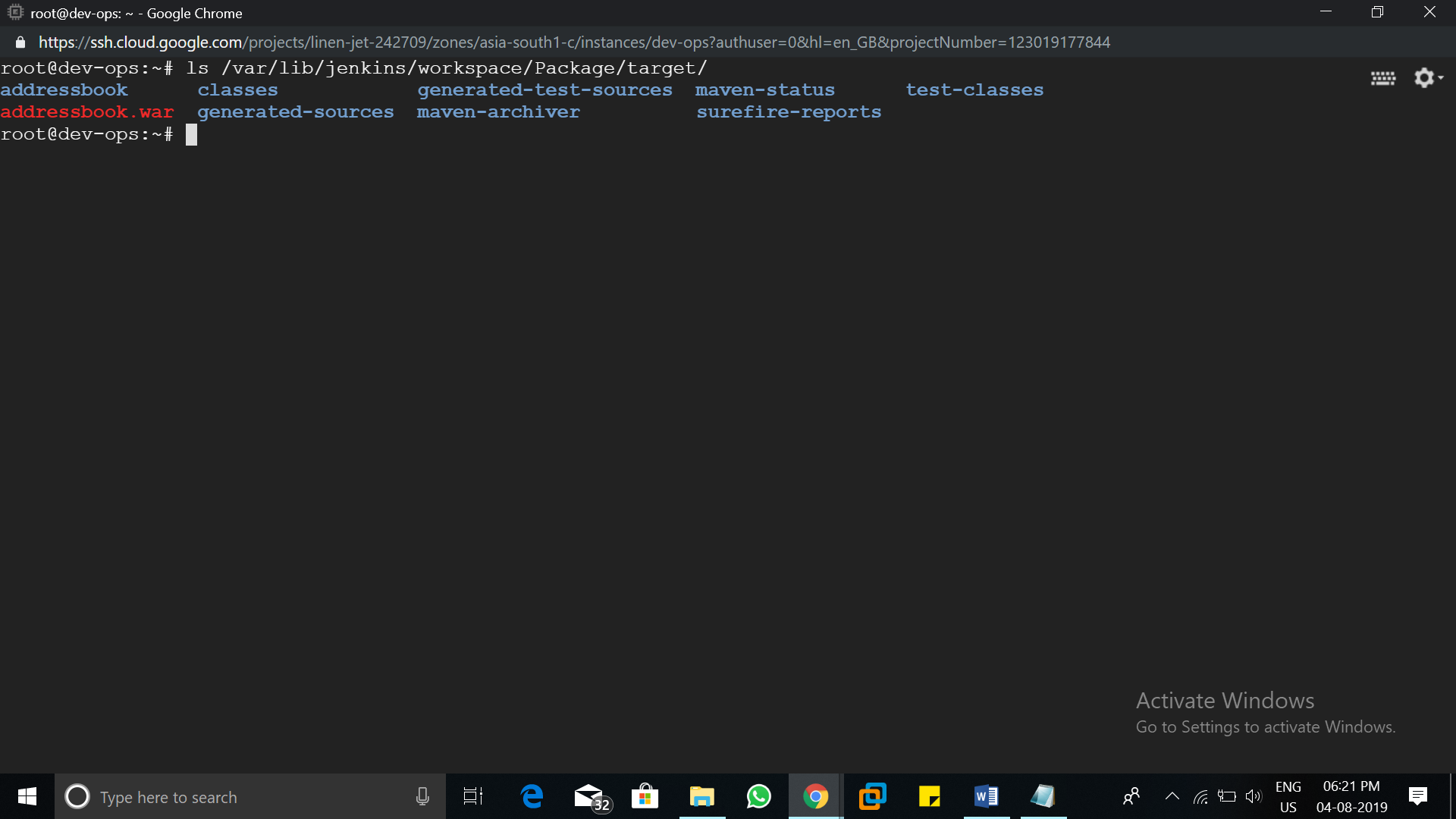
If all the builds are successful then, we are ready to build the pipeline for continuous integration.



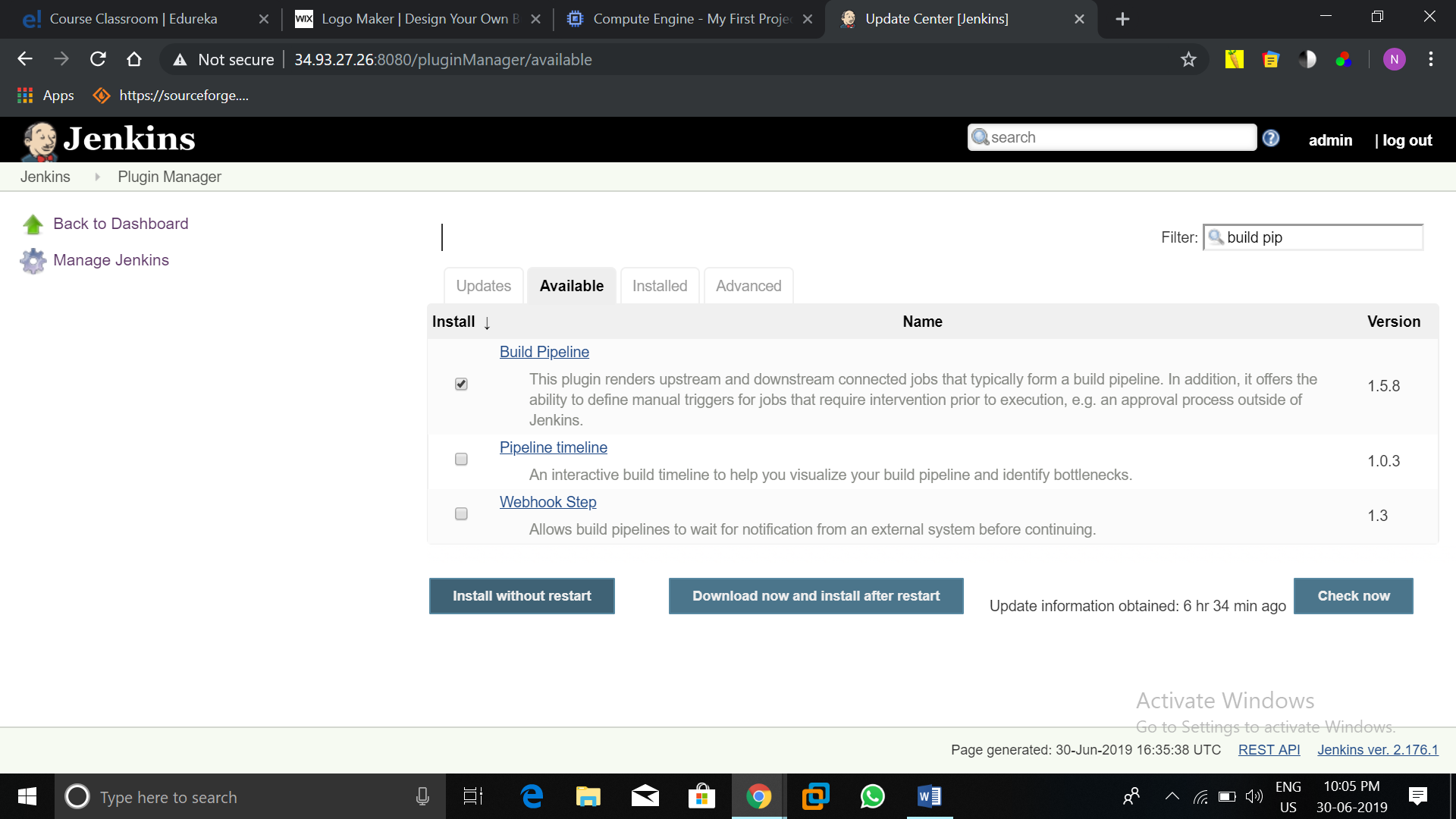
**Build Pipeline**

Go to the Ubuntu server and type ls /var/lib/Jenkins/workspace/Package/target/

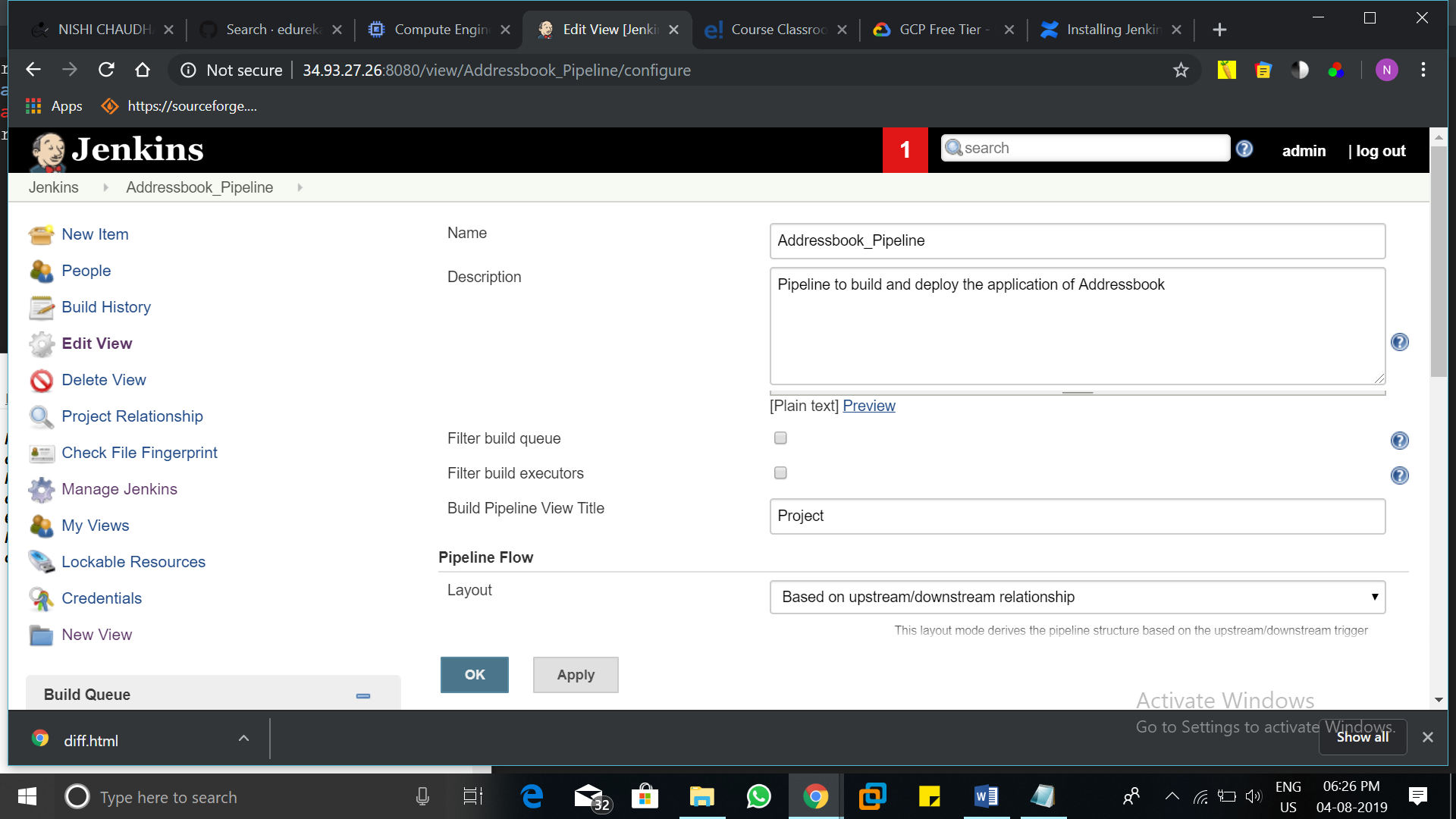
There you will find the war file of the application.



Now go to Jenkins dashboard, and click on manage Jenkins, go to manage plugins and in available section search for Build Pipeline Plugin, Install it without start.

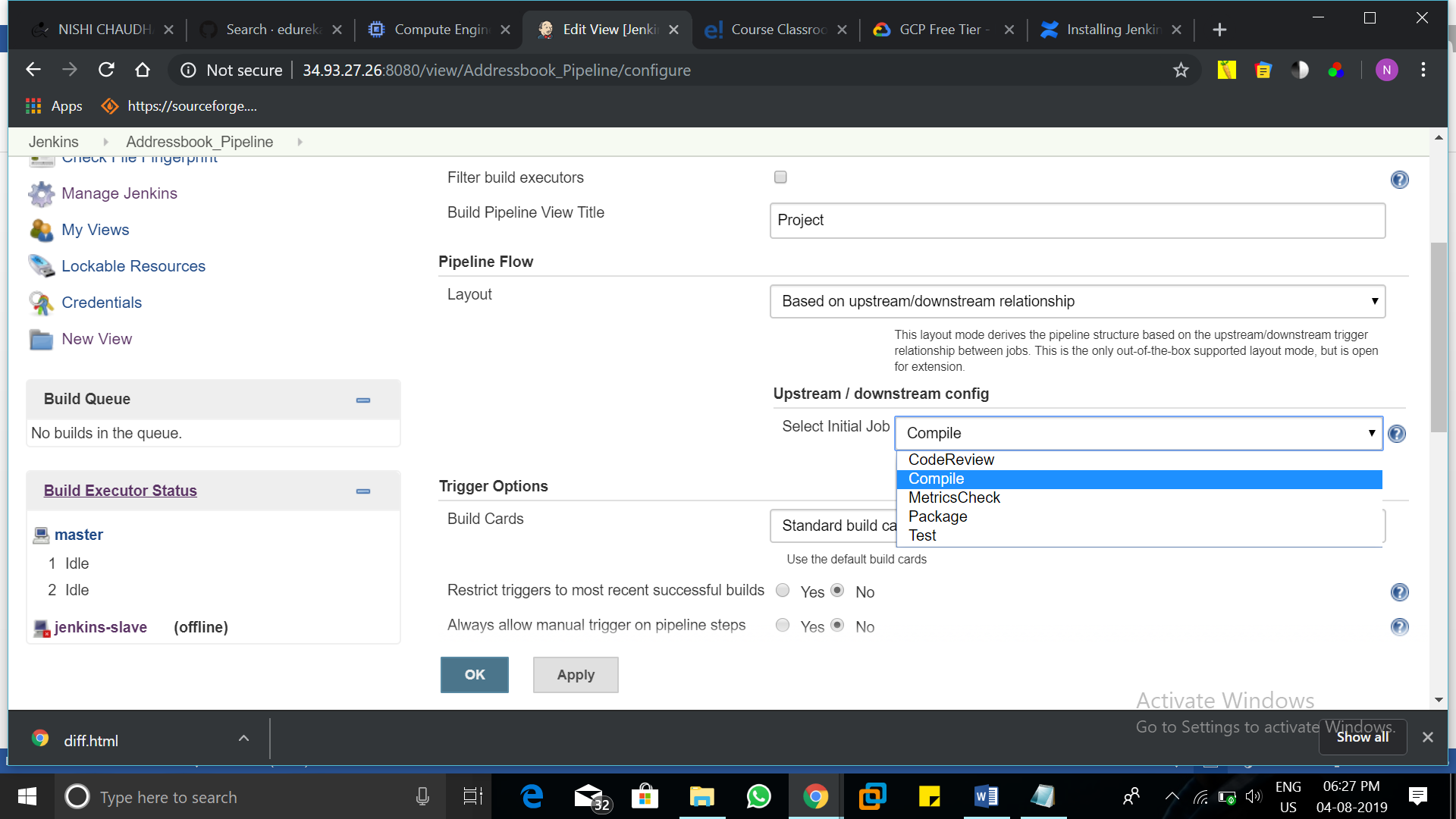


After installation is completed click on the ‘+’ Button in the Jenkins dashboard, there you find the option of Build Pipeline, select it and give a name to your pipeline, we are giving Addressbook Pipeline.



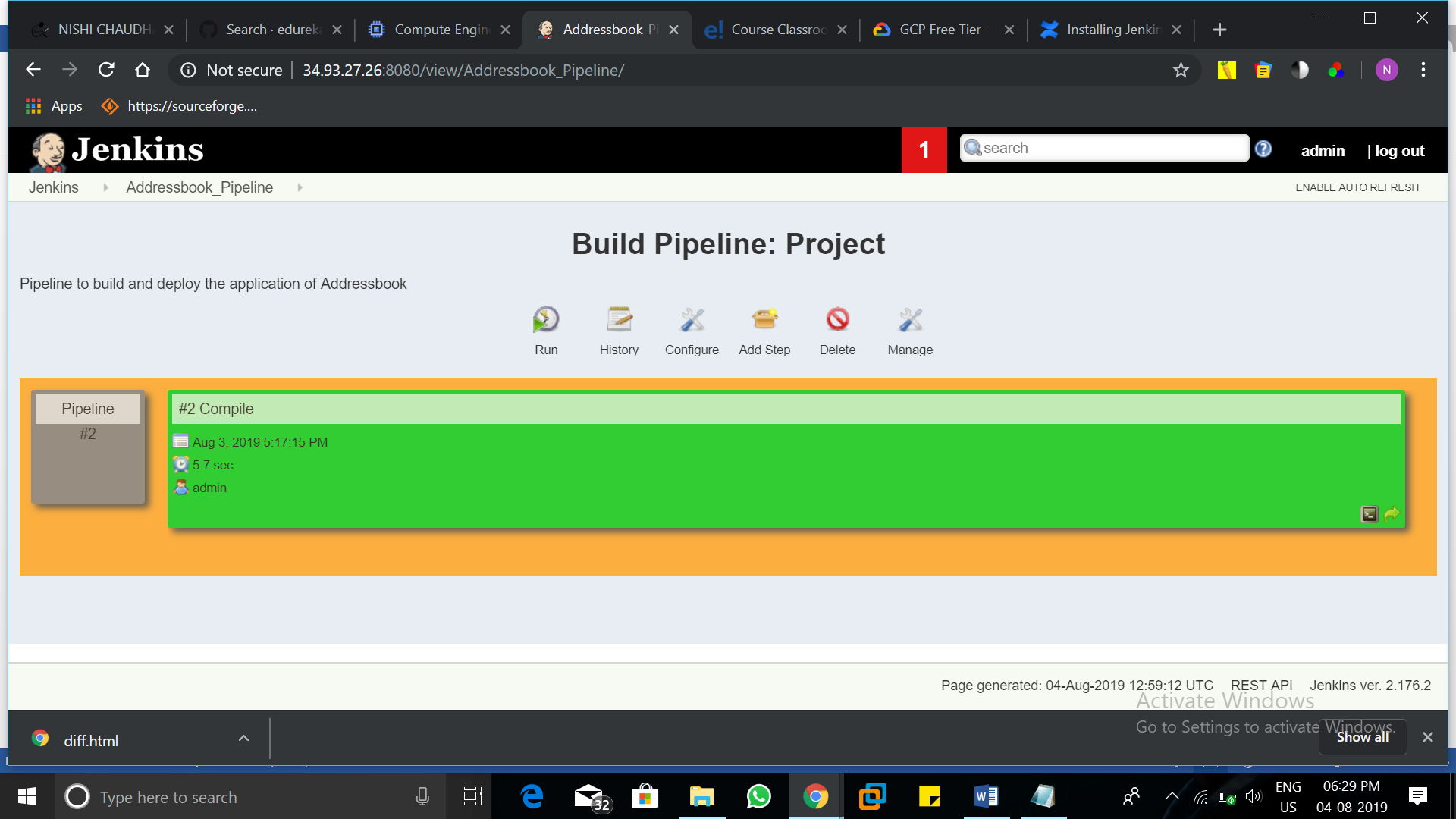
Give the description and title of the pipeline.

Select the first job of the pipeline in Layout option.



We have the first job as Compile, so select it and click on Apply then OK.

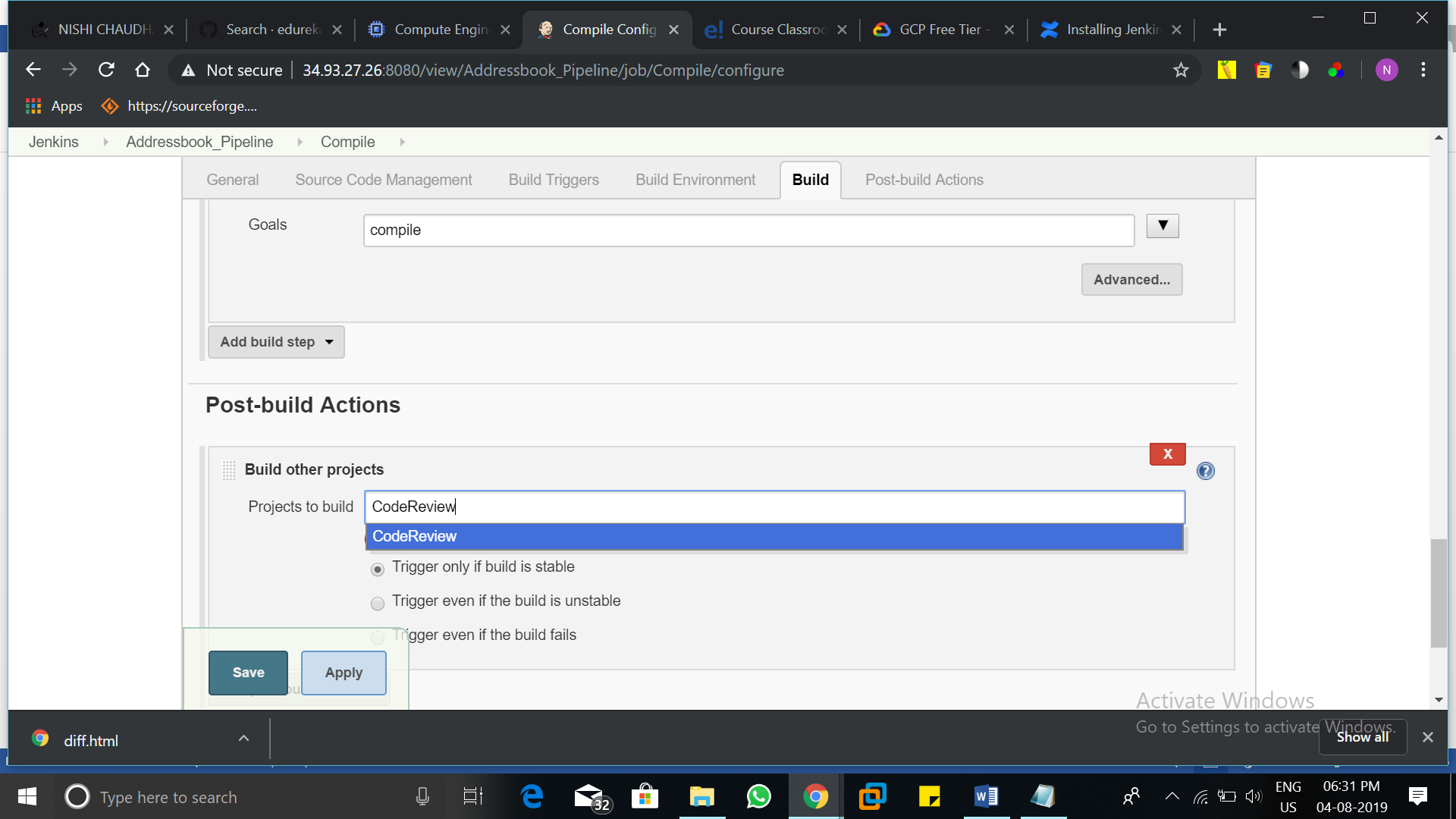
It will looks like the following image.



Now click on compile and configure the job.

In Post-build Actions, select Build Other Projects and select the job CodeReview and select an option of Trigger only if build is stable.

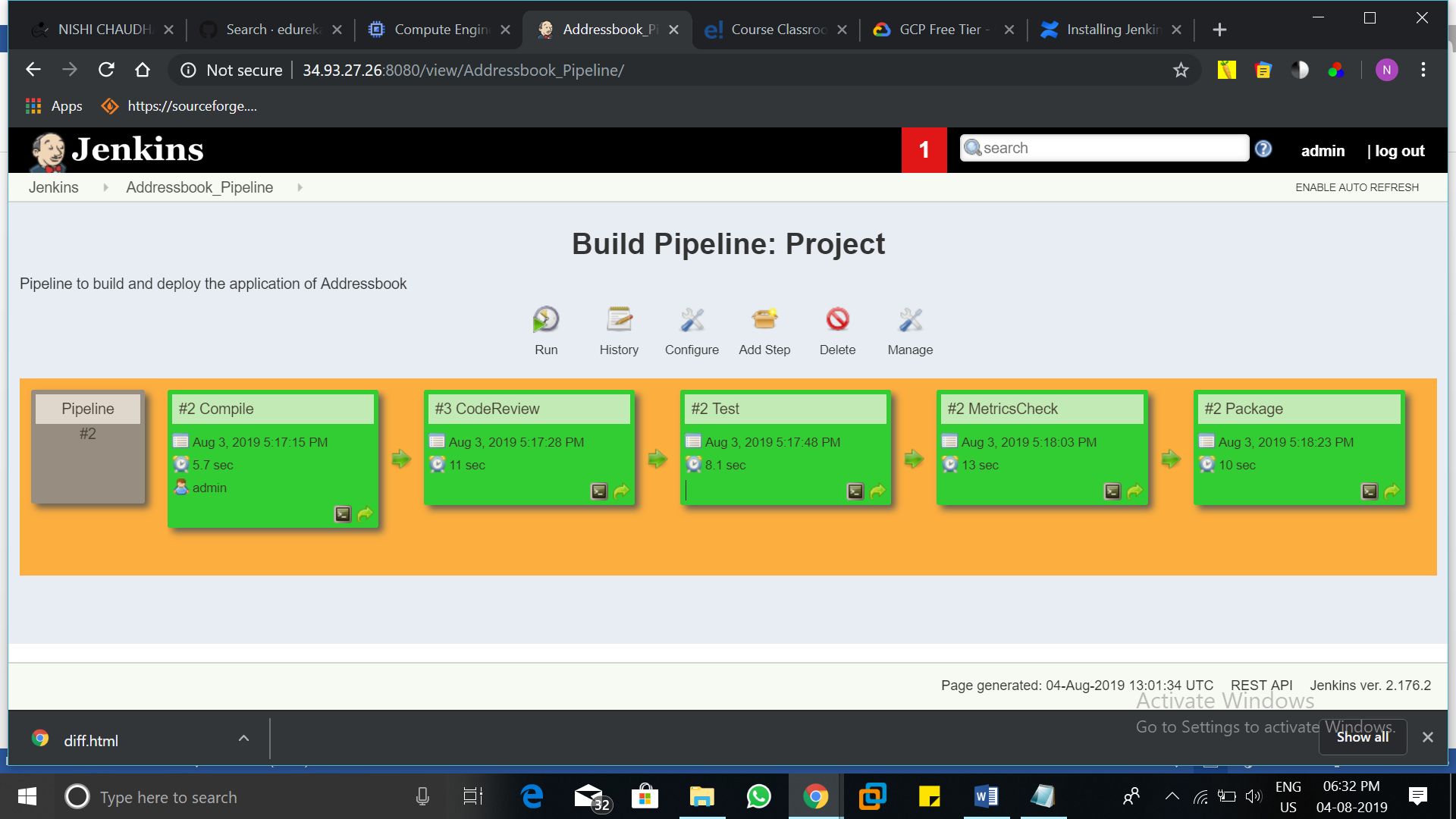
Click on Save.



Similarly add all the jobs after one in the queue.

The final pipeline will looks like this.

Run the pipeline once to confirm its functioning smoothly.



Install python, openssh-server, git on slave node

