**Triggering Parameterized Builds**

This section will guide you to:

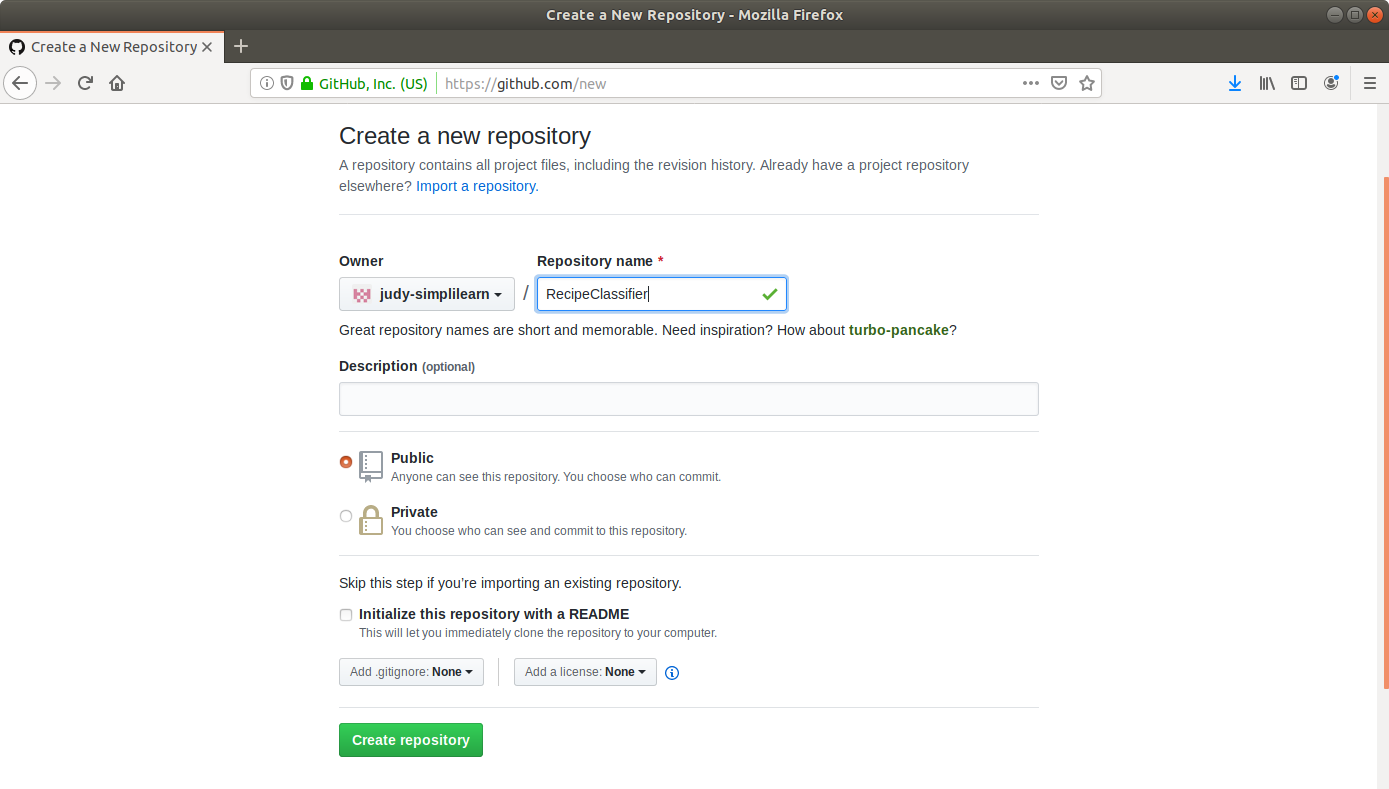
* Configure parameterized builds
* Trigger a parameterized build remotely via curl
* Trigger a build job based on another job’s results

**Step 1:** Creating a Git repository for the recipe categorization program

* Login to your Github account.
* Click on the plus icon next to the profile picture and select *New repository* from the drop down menu.



* Fill the required fields in the create repository form .



* Click on the **Create Repository** button.

**Step 2:** Adding the code for recipe categorization to the repository

* Open the terminal and navigate to an appropriate location.
* Run **mkdir RecipeClassifier** to create a directory.
* Navigate to the *RecipeClassifier* directory.
* Run **nano MostRepeatedWord.java** to open the java file in a text editor.
* Paste the below code into the file. The code will find the most repeated word in the recipe other than articles and that would be the category the recipe belongs to.

**import java.io.BufferedReader;**

**import java.io.FileReader;**

**import java.util.ArrayList;**

**import java.util.Arrays;**

**public class MostRepeatedWord {**

**public static void main(String[] args) throws Exception {**

**String line, word = "";**

**int count = 0, maxCount = 0;**

**ArrayList<String> words = new ArrayList<String>();**

**ArrayList<String> articles = new ArrayList<String>(Arrays.asList("the", "a", "an"));**

**//Opens file in read mode**

**FileReader file = new FileReader(args[0]);**

**BufferedReader br = new BufferedReader(file);**

**//Reads each line**

**while((line = br.readLine()) != null) {**

**String string[] = line.toLowerCase().split("([,.\\s]+)");**

**//Adding all words generated in previous step into words**

**for(String s : string){**

**if (articles.contains(s))**

**continue;**

**else**

**words.add(s);**

**}**

**}**

**//Determine the most repeated word in a file**

**for(int i = 0; i < words.size(); i++){**

**count = 1;**

**//Count each word in the file and store it in variable count**

**for(int j = i+1; j < words.size(); j++){**

**if(words.get(i).equals(words.get(j))){**

**count++;**

**}**

**}**

**//If maxCount is less than count then store value of count in maxCount**

**//and corresponding word to variable word**

**if(count > maxCount){**

**maxCount = count;**

**word = words.get(i);**

**}**

**}**

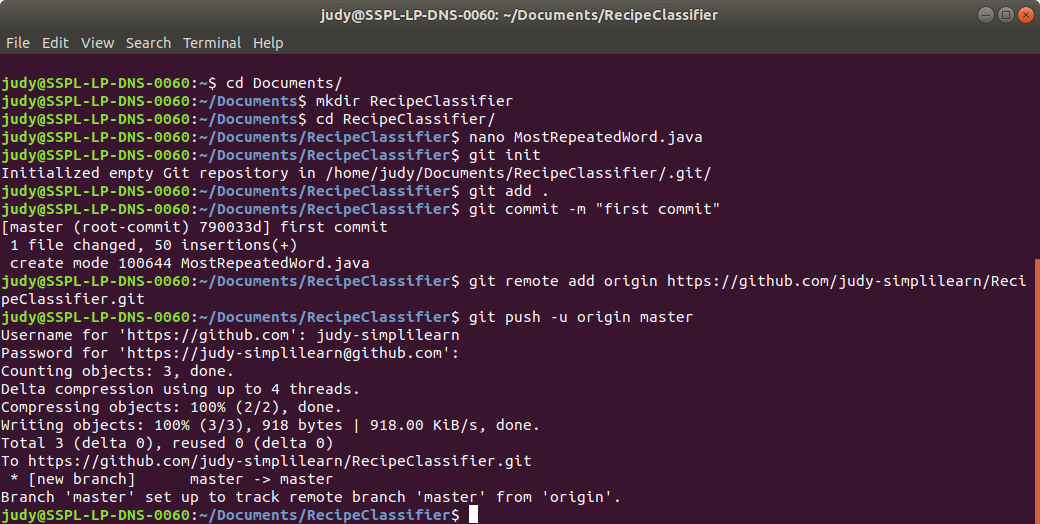
**System.out.println("Most repeated word: " + word);**

**br.close();**

**}**

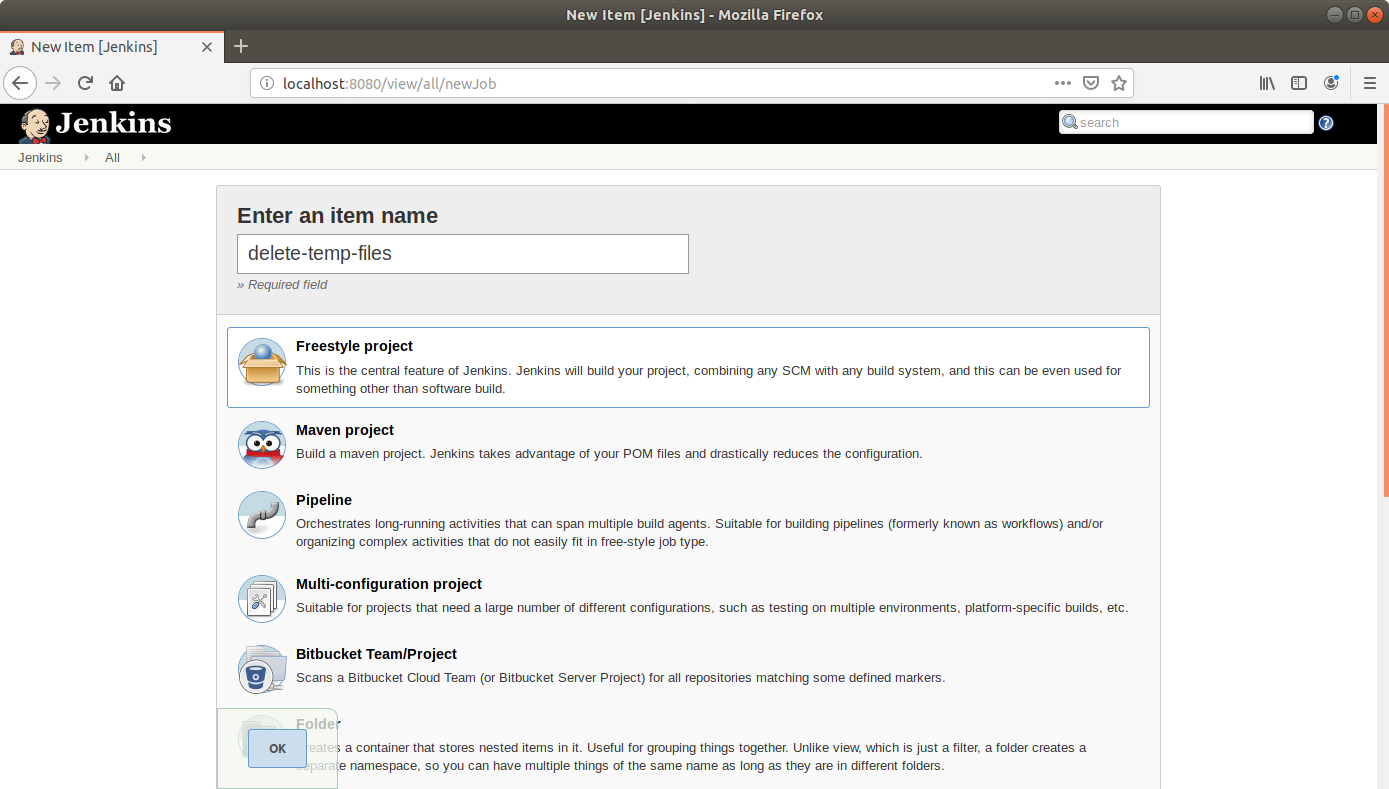
**}**

* Save the file and exit the text editor.
* Run **git init**
* Run **git add .**
* Run **git commit -m “Add new files”**
* Run **git remote add origin [Repository\_URL]**
* Run **git push -u origin master**

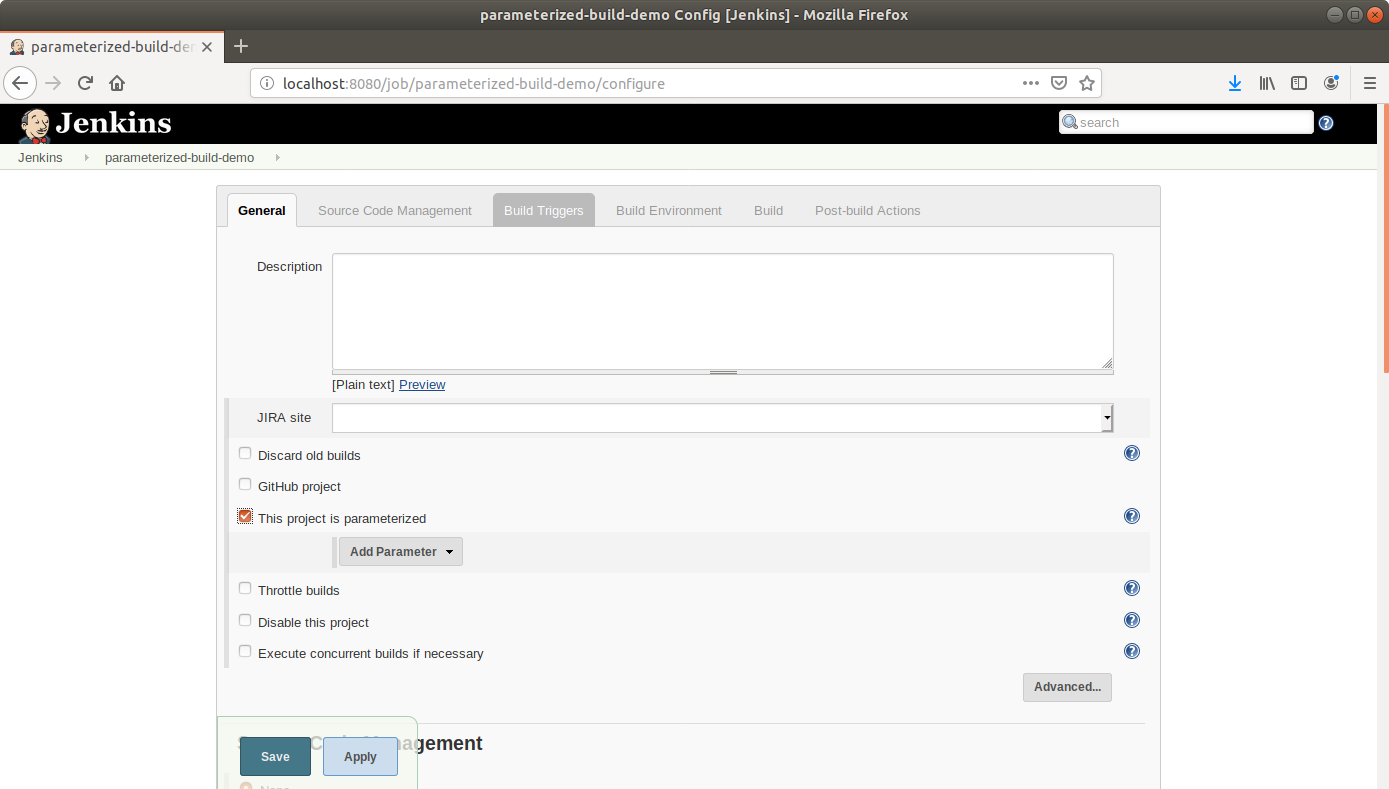


**Step 3:** Creating a parameterized build job in Jenkins to delete temporary files

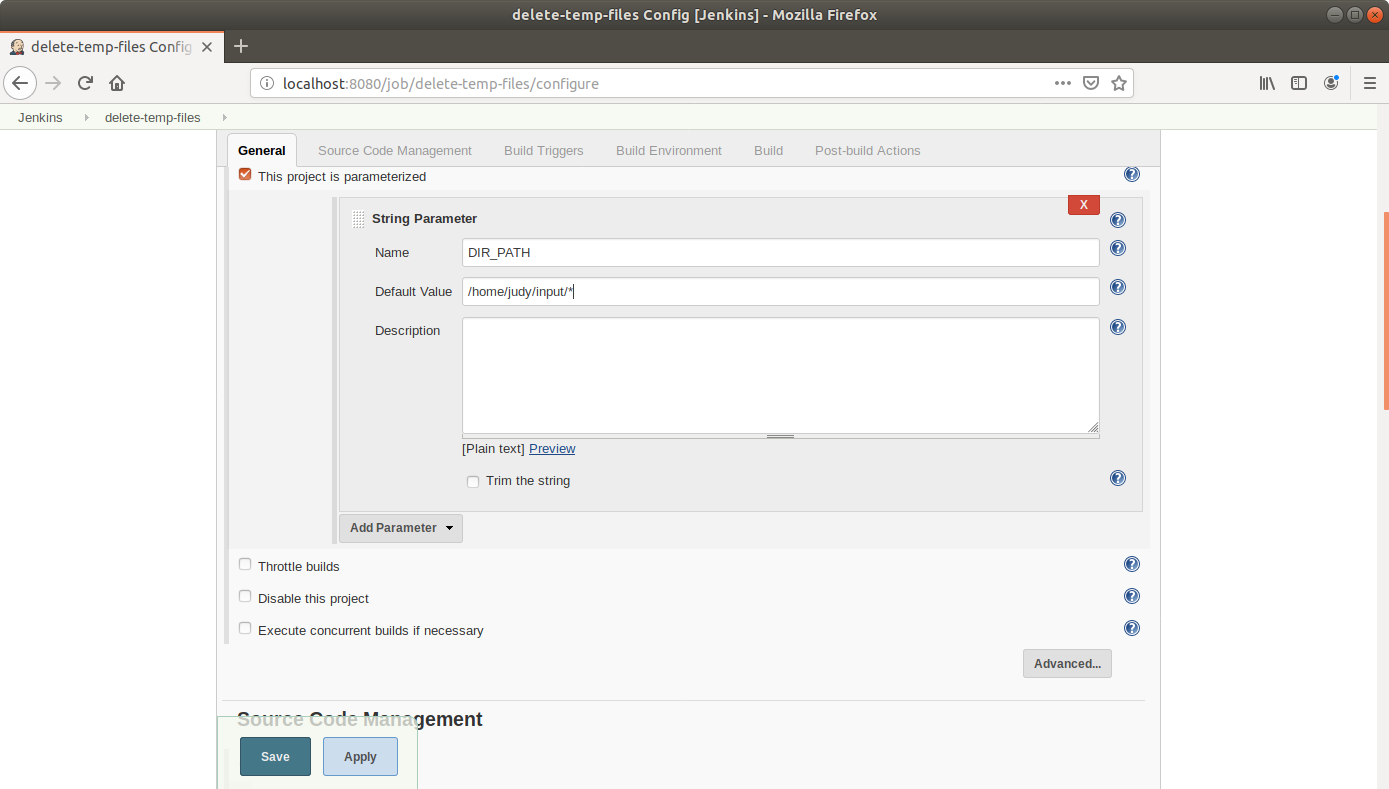
* Go to Jenkins dashboard.
* Click on *New Item*.
* Enter a name for your build job.
* Select *Freestyle project* as the build job type.



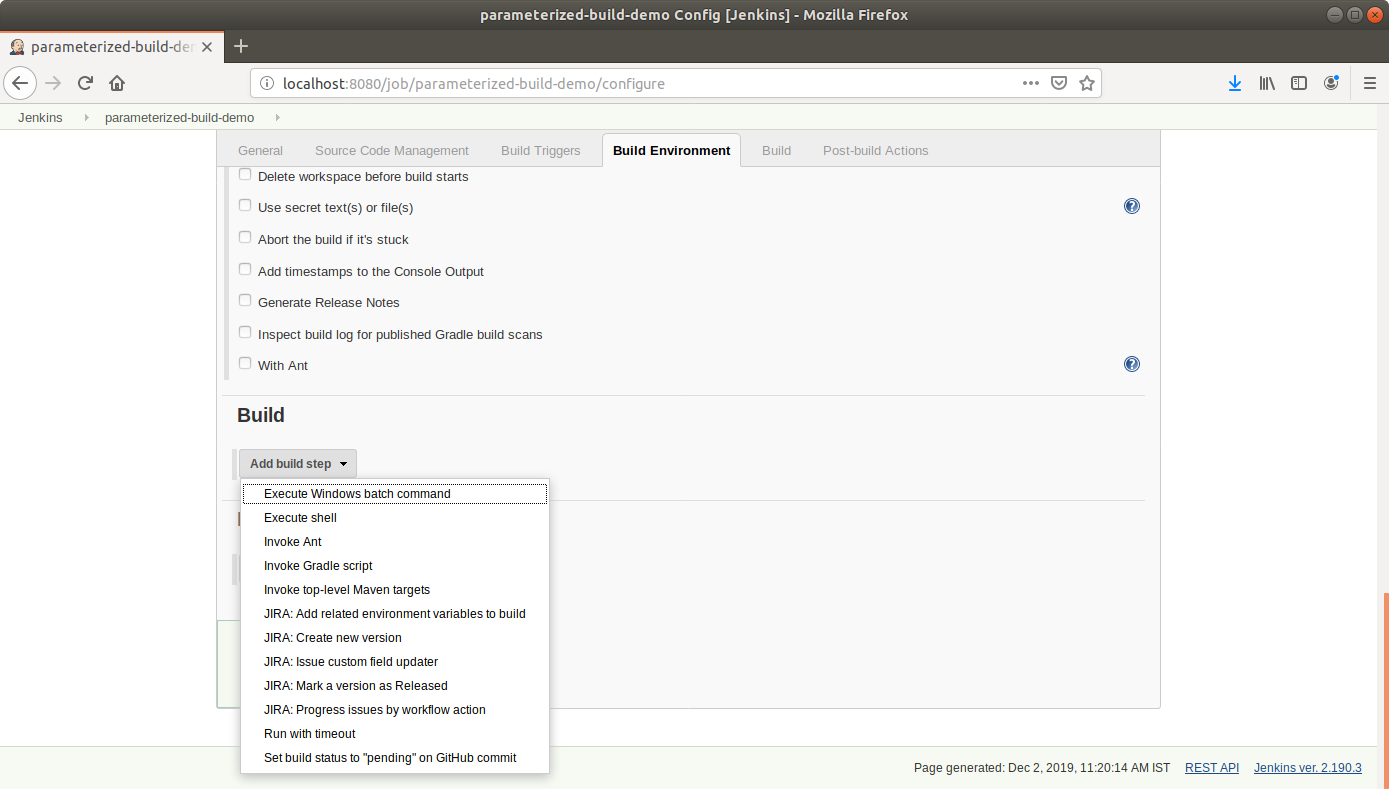
* Click OK.
* On the configuration page, select the checkbox next to text that states, *This project is parameterized*.
* Click on the *Add parameter* button.



* From the drop-down menu, select *String parameter*.
* Name the parameter as *DIR\_PATH* and enter the *Default Value*, the path to the input files.

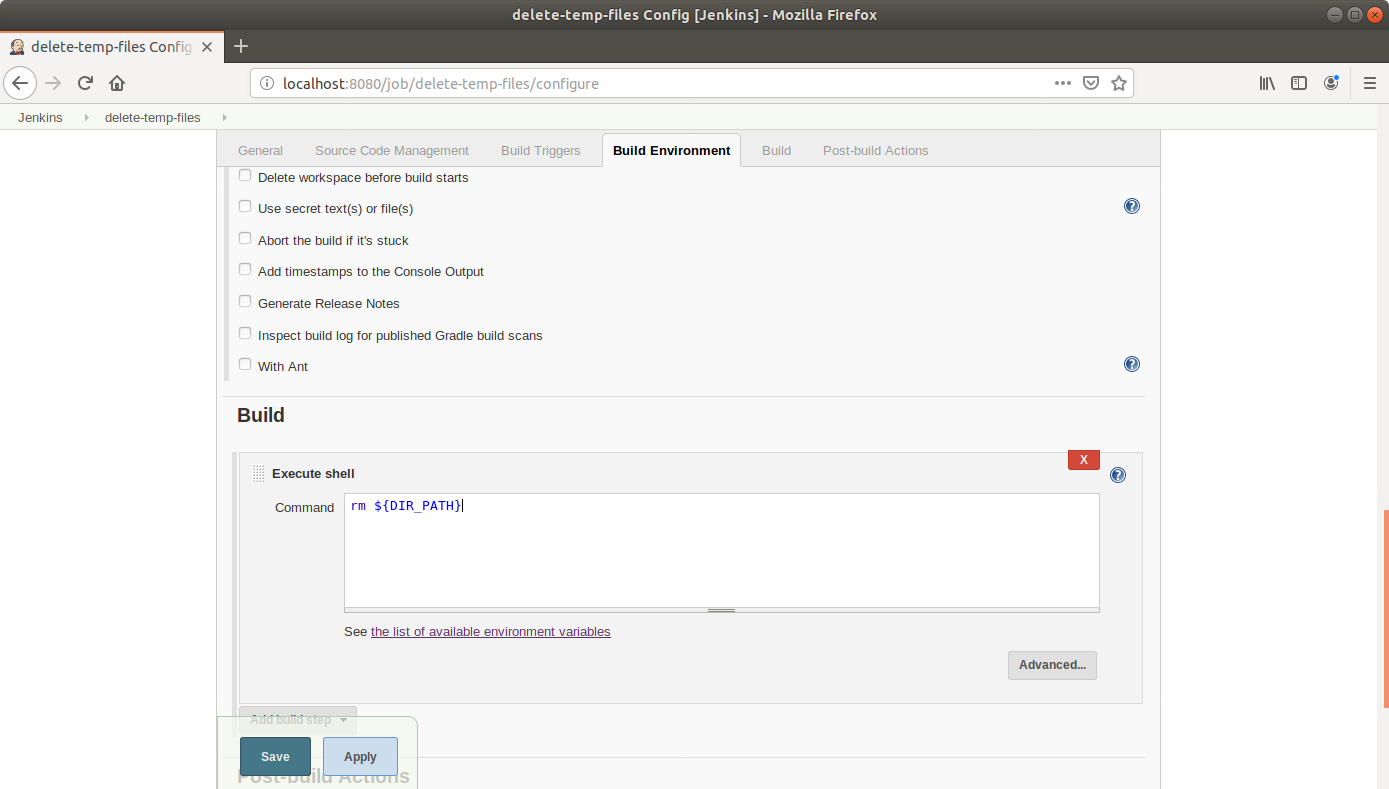


* Scroll down to the *Build* section and click on *Add build step*.
* Select *Execute Shell Command* from the drop down.



* In the text input area, enter the below text:

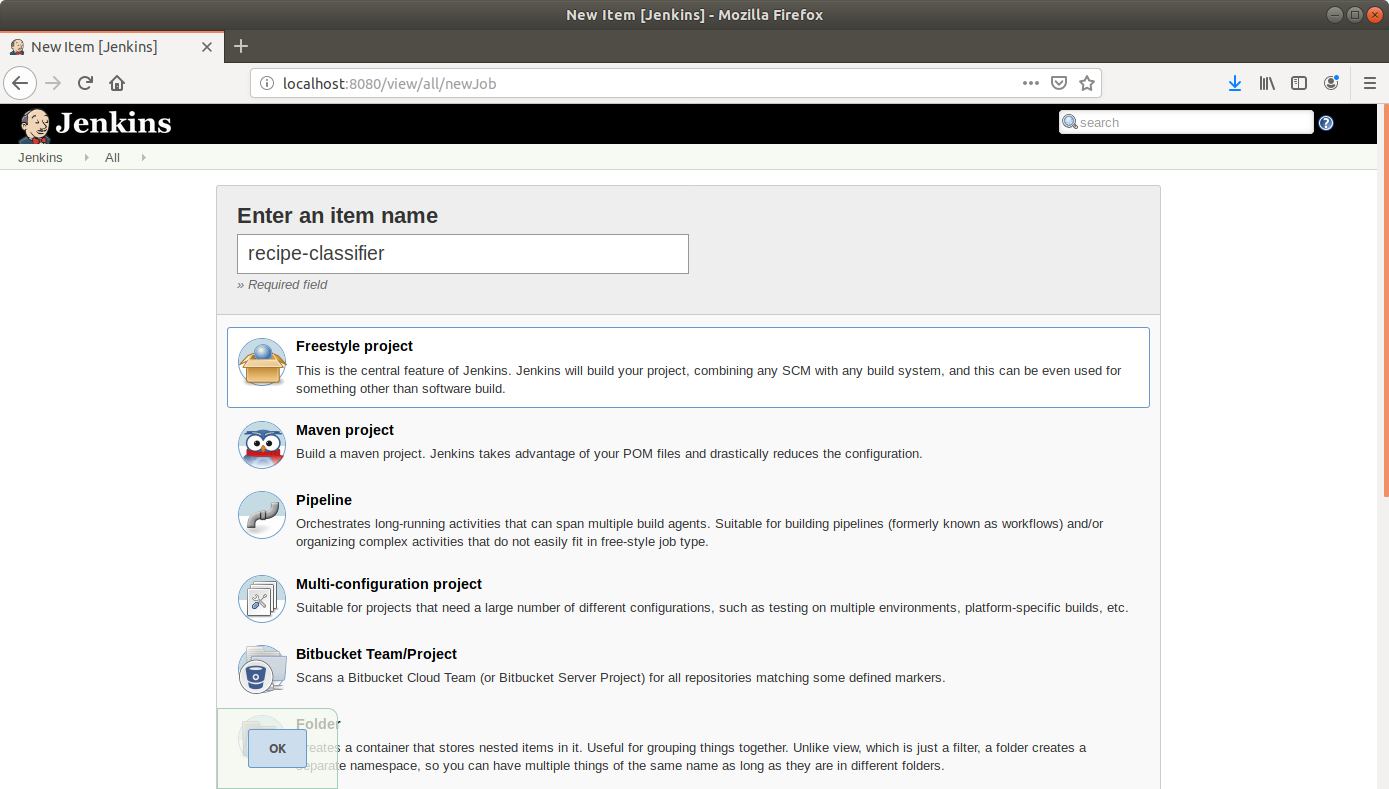
**rm ${DIR\_PATH}**



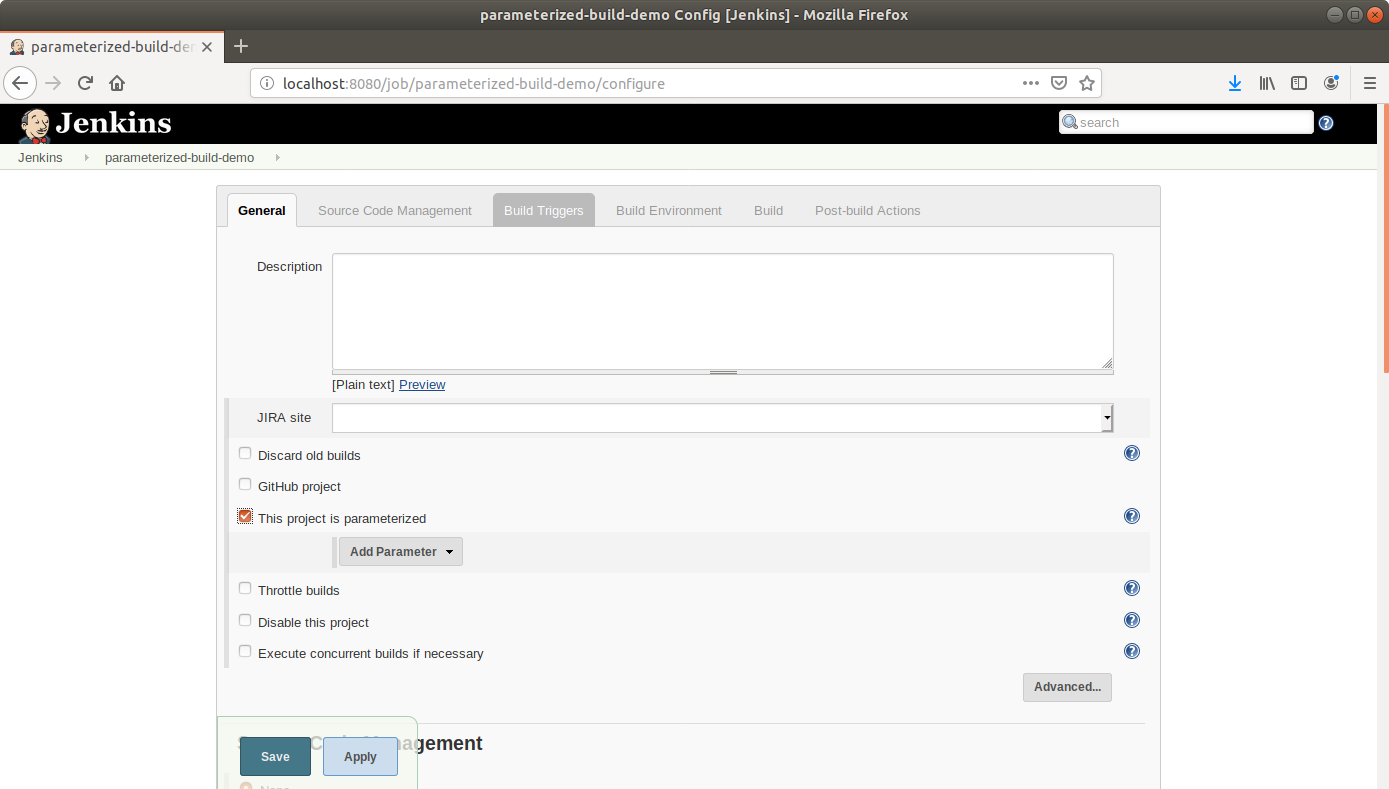
* Click Save.

**Step 4:** Creating a parameterized build job in Jenkins to categorize recipes and trigger delete job

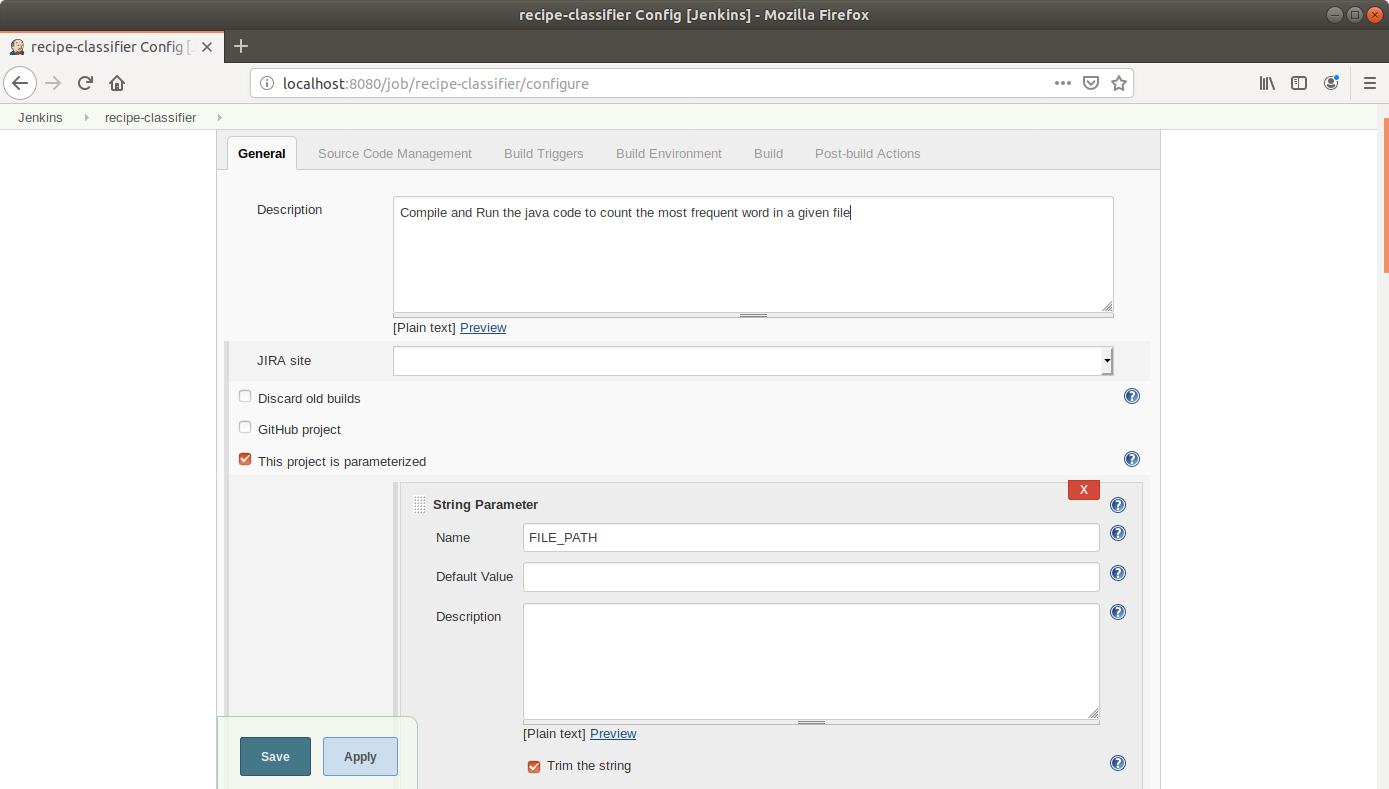
* Go to Jenkins dashboard.
* Click on *New Item*.
* Enter a name for your build job.
* Select *Freestyle project* as the build job type.



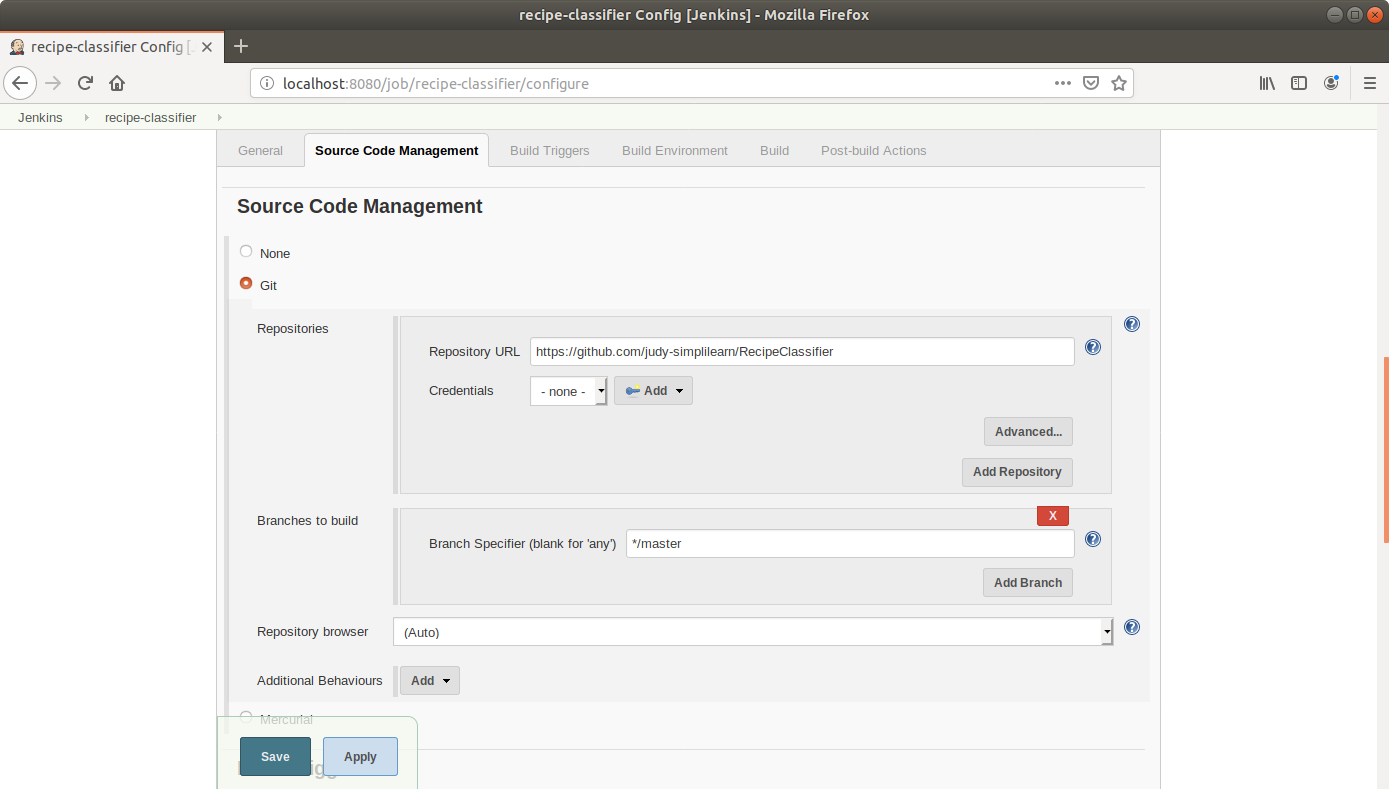
* Click OK.
* On the configuration page, select the checkbox next to text that states, *This project is parameterized*.
* Click on the *Add parameter* button.



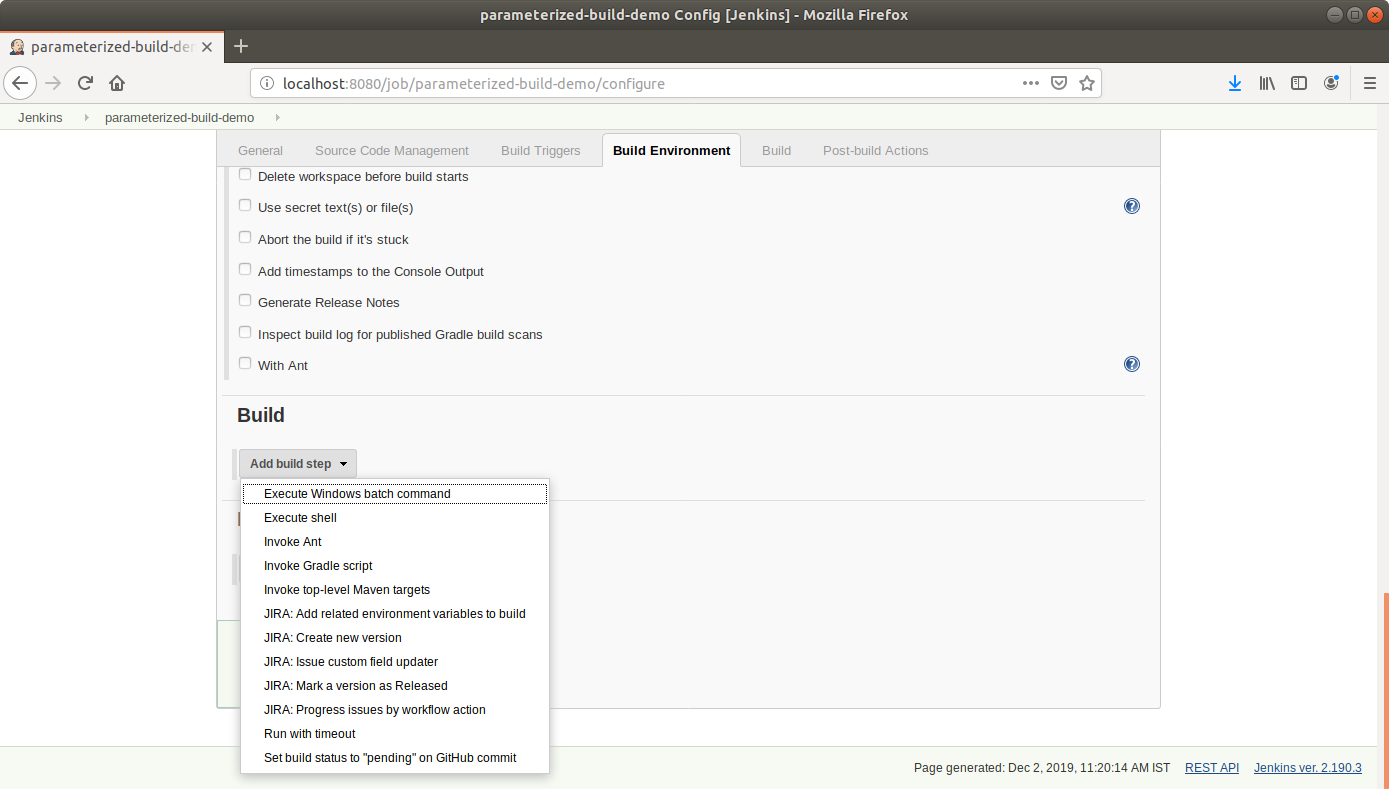
* From the drop-down menu, select *String parameter*.
* Name the parameter as *FILE\_PATH* and select the checkbox that says, *Trim the string*.



* Scroll down to the Source Code Management section and select *Git*.
* Enter the link to the repository in the field that appears.



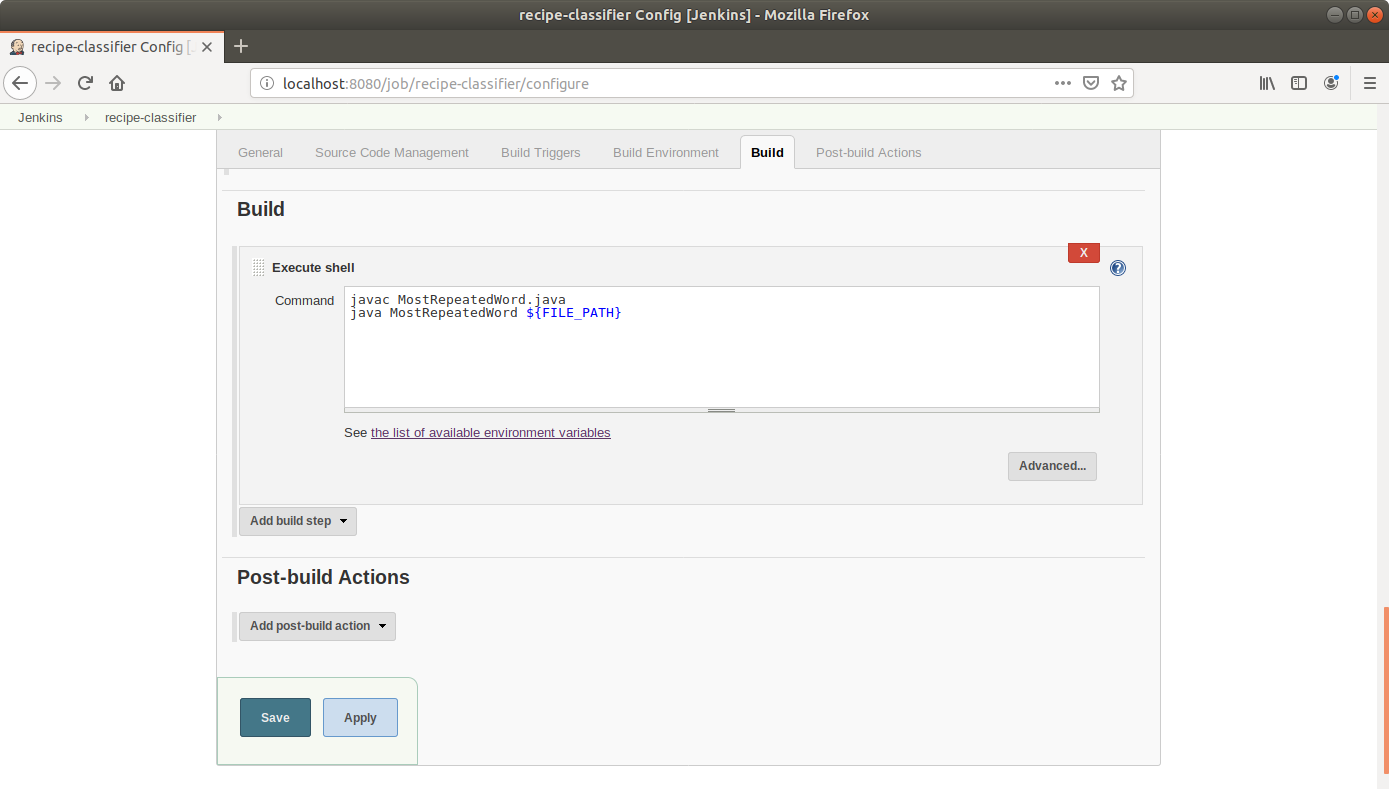
* Scroll down to the *Build* section and click on *Add build step*.
* Select *Execute Shell Command* from the drop down.



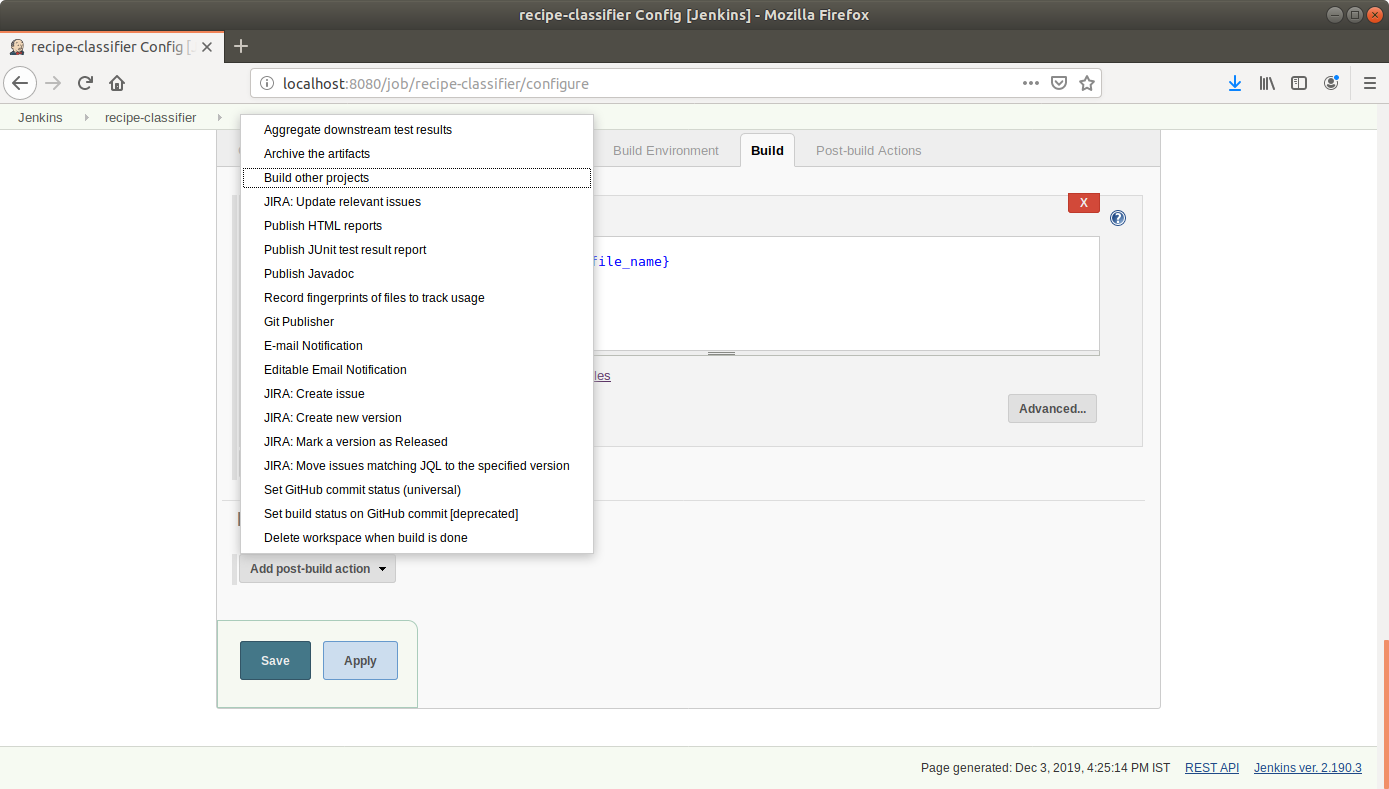
* In the text input area, enter the below text:

**javac MostRepeatedWord.java**

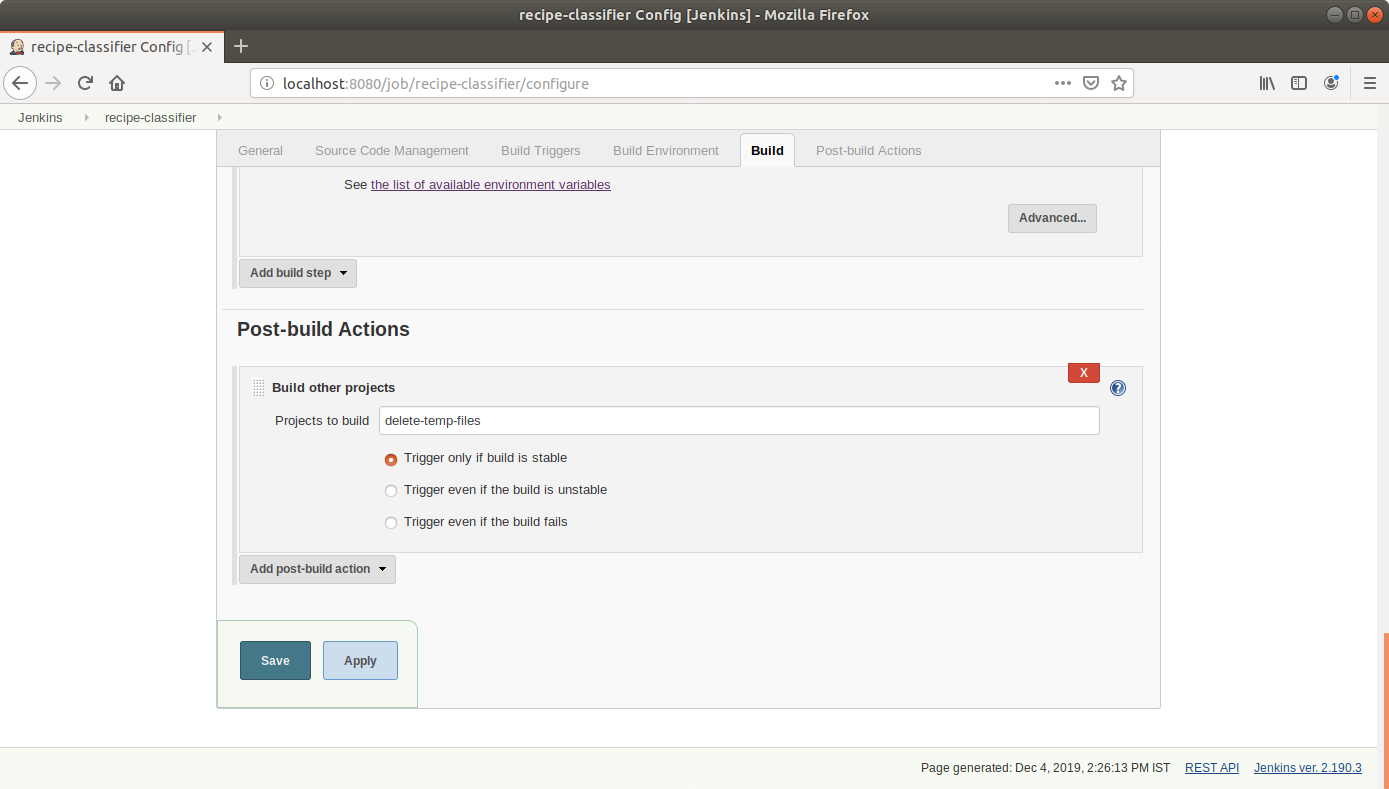
**java MostRepeatedWord ${FILE\_PATH}**

****

* Scroll down to the *Post-build Actions* tab and click on the *Add post-build action* button.
* From the drop down, select *Build other projects.*

**

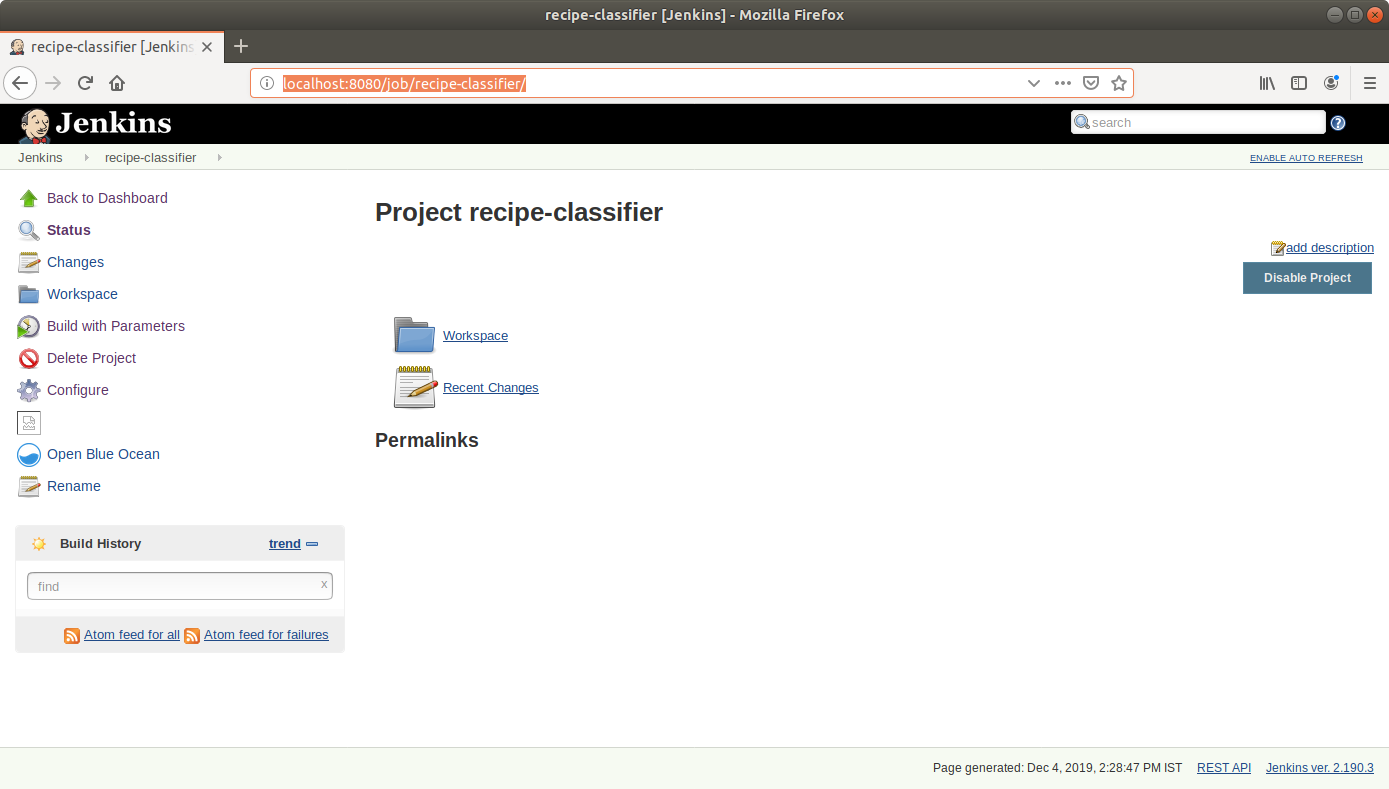
* Enter the name of the project to be triggered post-build in the textbox that appears.



* Click Save.

**Step 5:** Triggering a parameterized build remotely

* Copy the URL to the job from the project window.

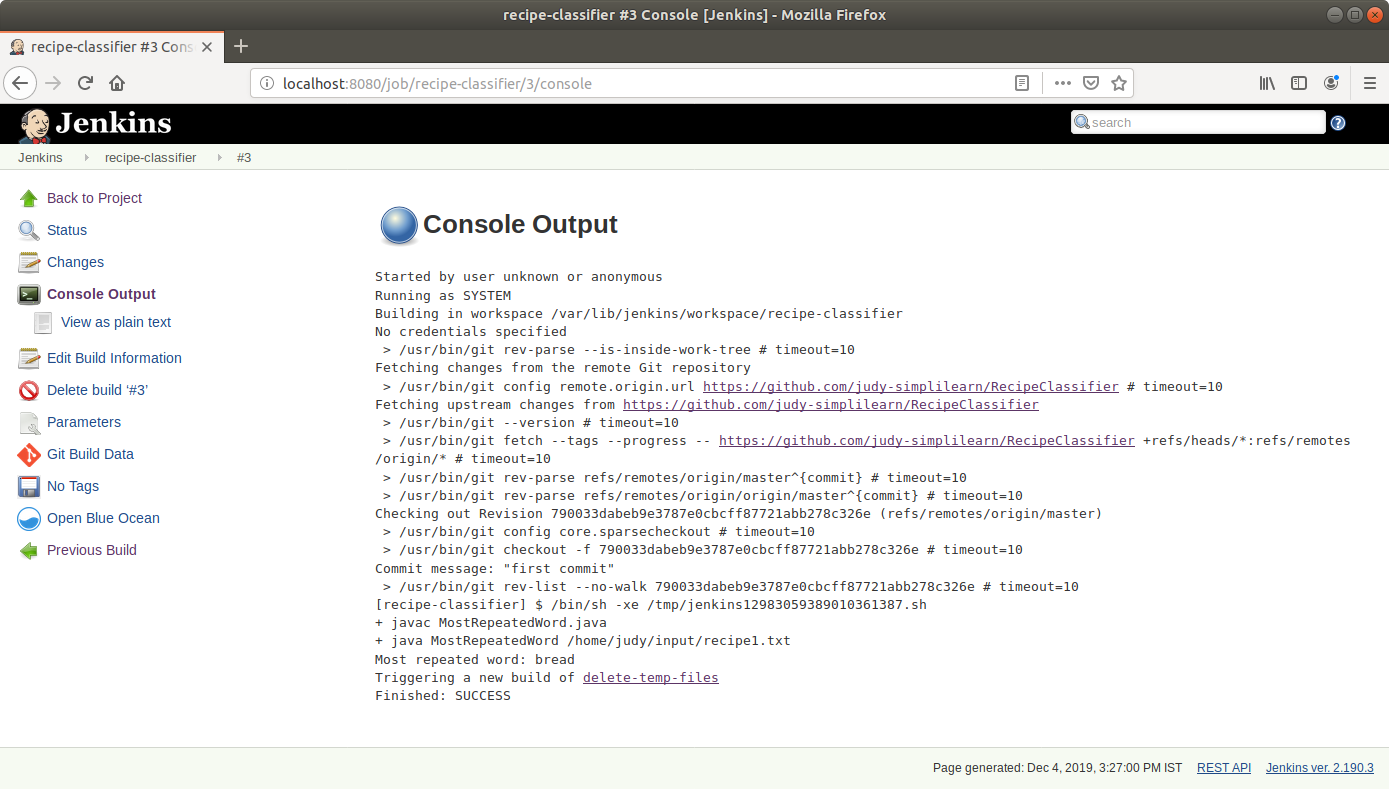


* Open the terminal.
* Suffix the parameter and its value to the URL copied in the previous step and send a *GET* request using *Curl*:

***curl -X GET http://localhost:8080/job/recipe-classifier/buildWithParameters?FILE\_PATH=’/home/judy/input/recipe1.tx***t’



* Open the job in the Jenkins UI.
* Click on the *Build History* to view the build results.
* Click on the *Console Output* to view the build logs.



* Open the newly triggered job name in the logs.
* Click on the *Build History* to view the build results for the post-build job.
* Click on the *Console Output* to view the build logs.

