**Assessment 20 – Jenkins CI/CD**

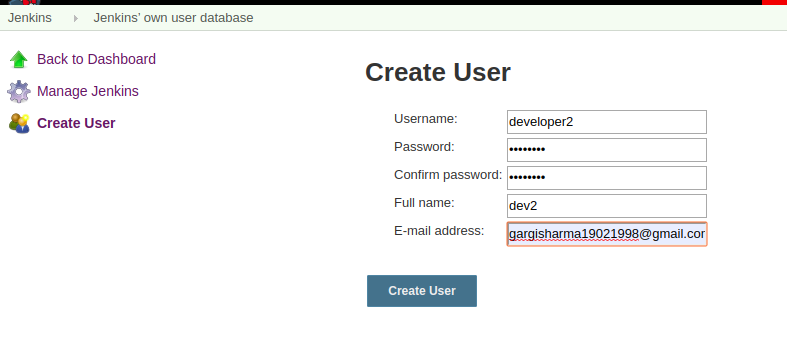
**Trainee Name : Gargi Sharma**

**Mentor Name : Mr. Akansh Gupta**

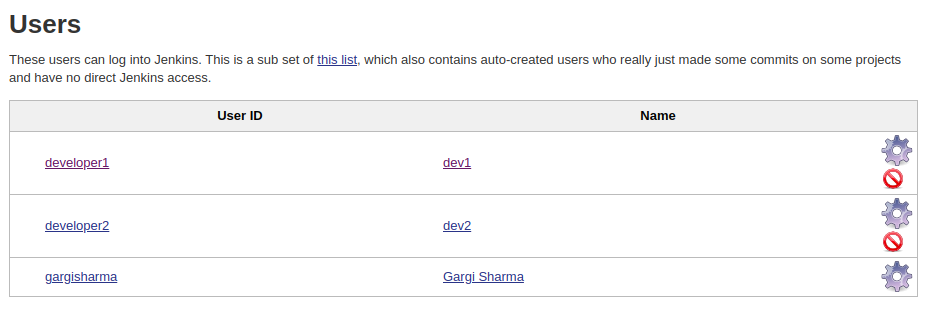
**College Name : UPES**

1. Create 2 users: developer1, developer2. The developer1 should be able to build job1 only and can’t change the job configuration. The developer2 can configure and build the job2, also he is able to view job1 but can’t build/configure it.

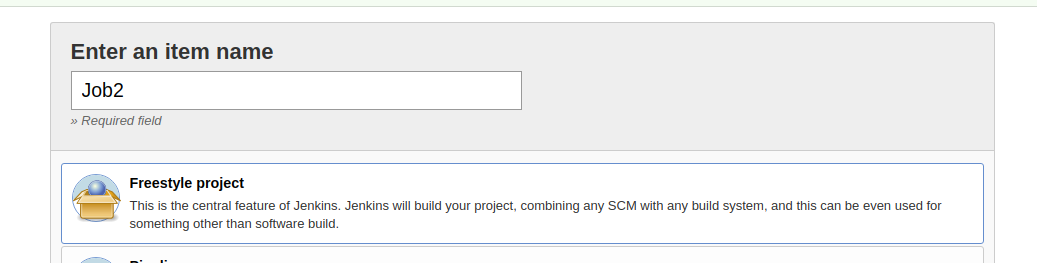
Go to manage jenkins -> manage users-> create user

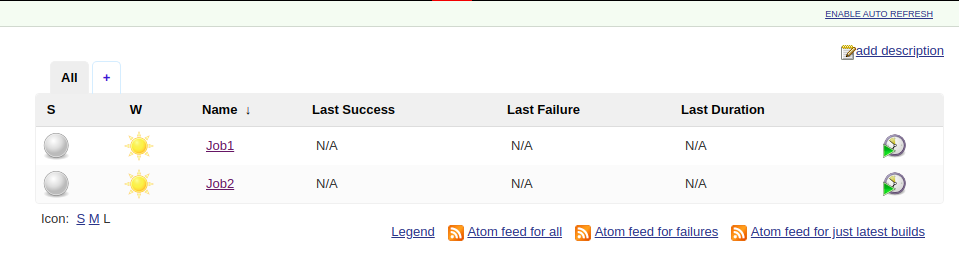


2 users developer 1 and developer2 have been created. By default these user do not have any permissions. So we will have to create global roles to give this user some permission.

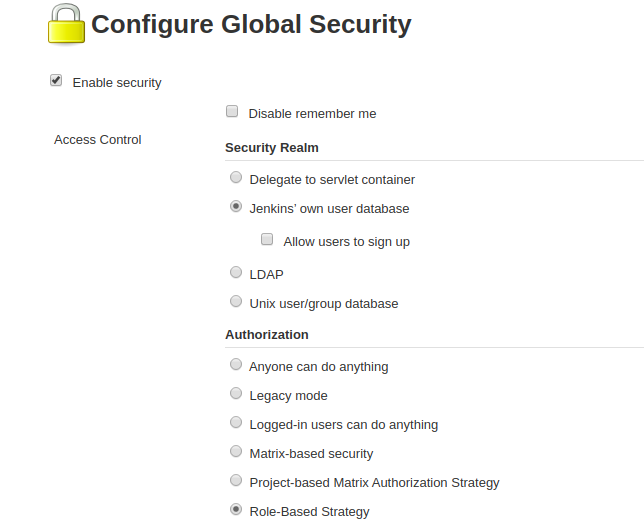


Create 2 freestyle jobs : job 1 and job 2.



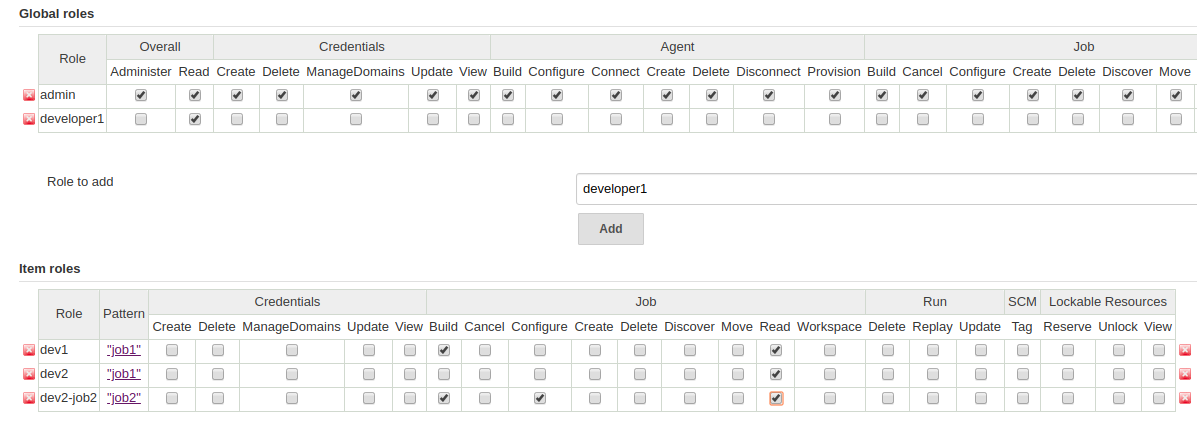


Now go to manage jenkins -> global security and enable security and allow role based access.

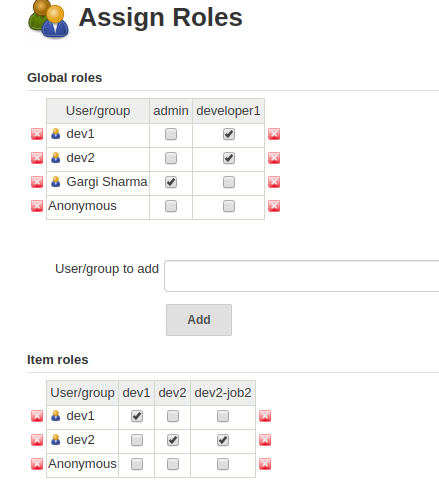


Now create one global role: developer1 that will be attached to both the new users giving them read permission.

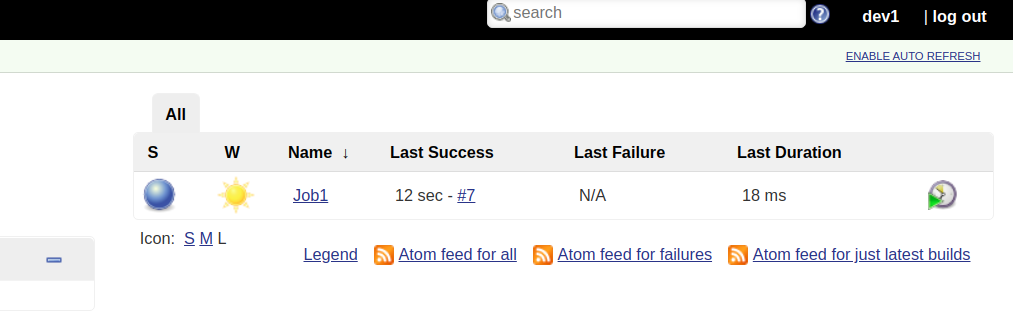
Also , create 3 item roles that will be job specific: dev1, dev2, dev2-job2.



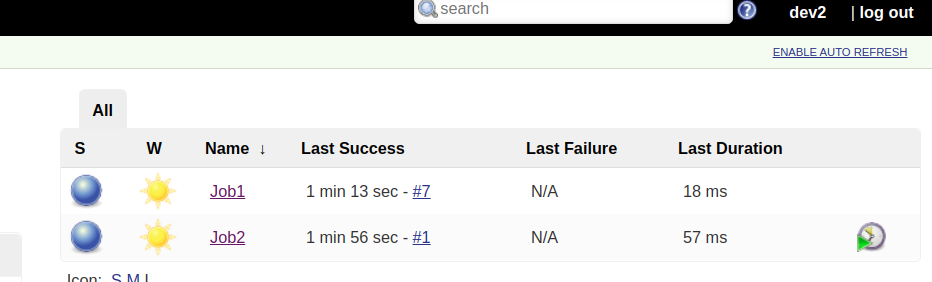
Now assign these roles to the users.

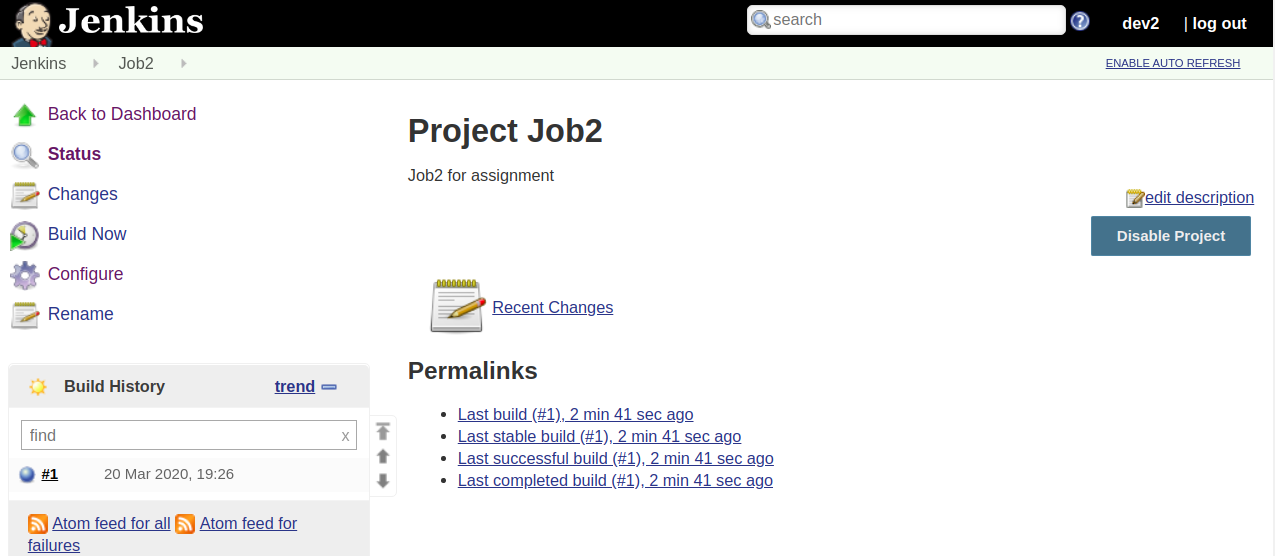


Now when we log in to dev1 user, we can see job1 and build it .



And when we login to dev2, we can read both jobs but can configure only job2.

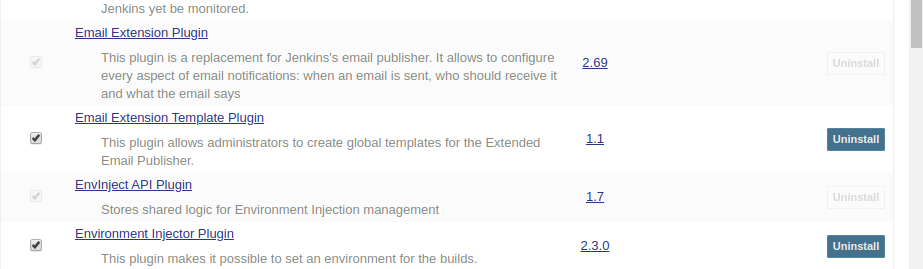




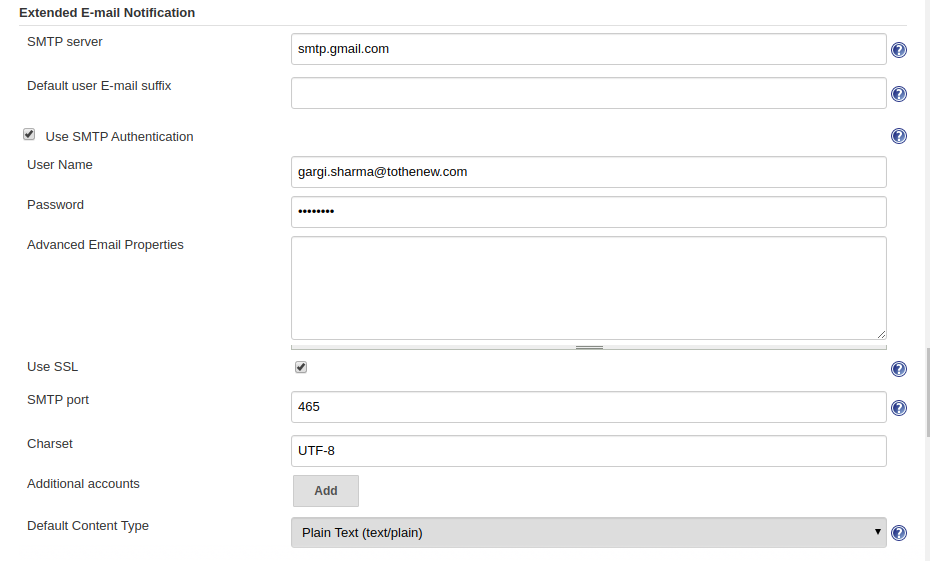
2. Create a Jenkins Job to create a calculator. It should give you a dropdown to ask Addition, Subtraction, Multiplication or Division and email the results.

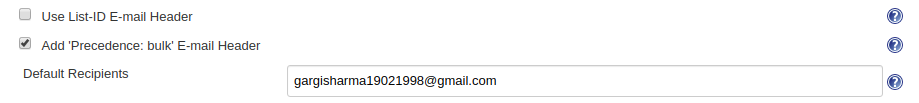
*Hint: Plugin Used (Environment Injector, Extended Email Notification)*

Install the two plugins: Environment Injector and Extended email Notification



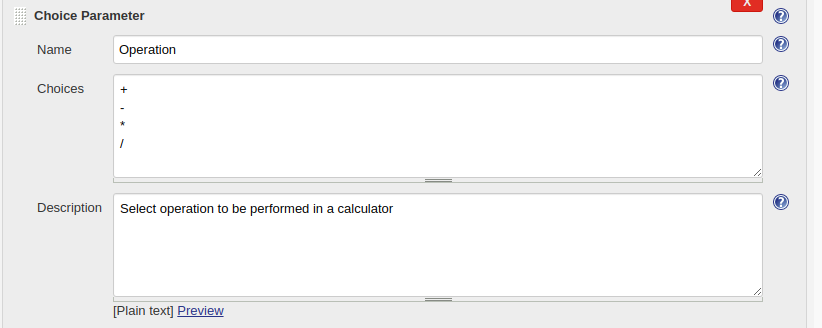
Now configure the extended email notification. Go to manage jenkins-> configure systems->Extended Email Notification. Also enable debug mode.



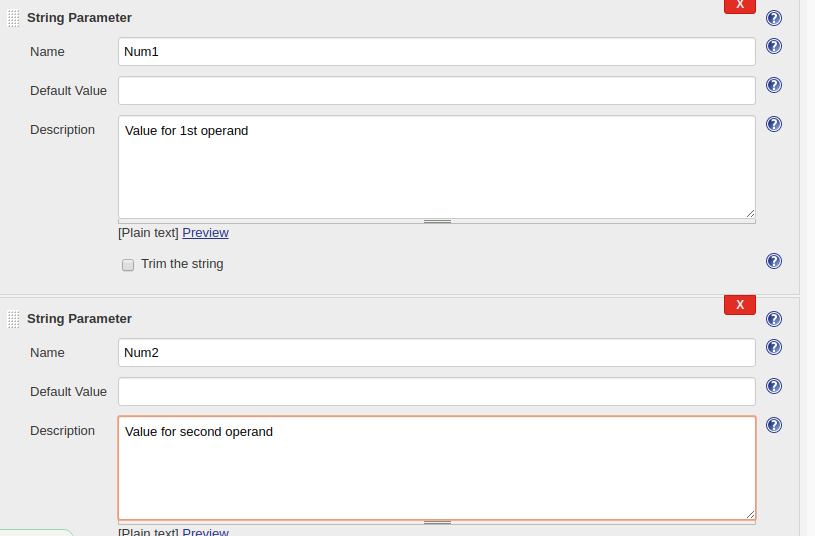


Now we will create a job for a calculator. It will be a parameterized job.

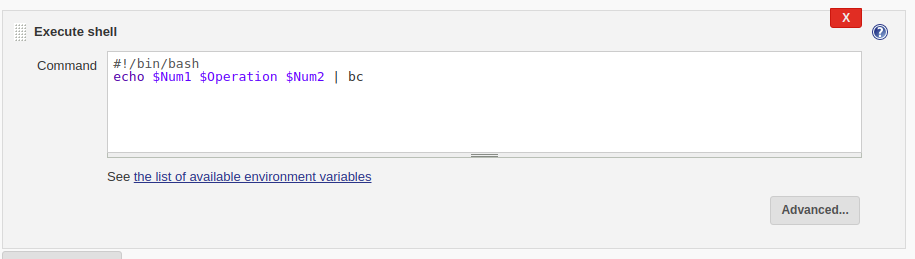
Create one choice parameter for “operations” in a calculator.



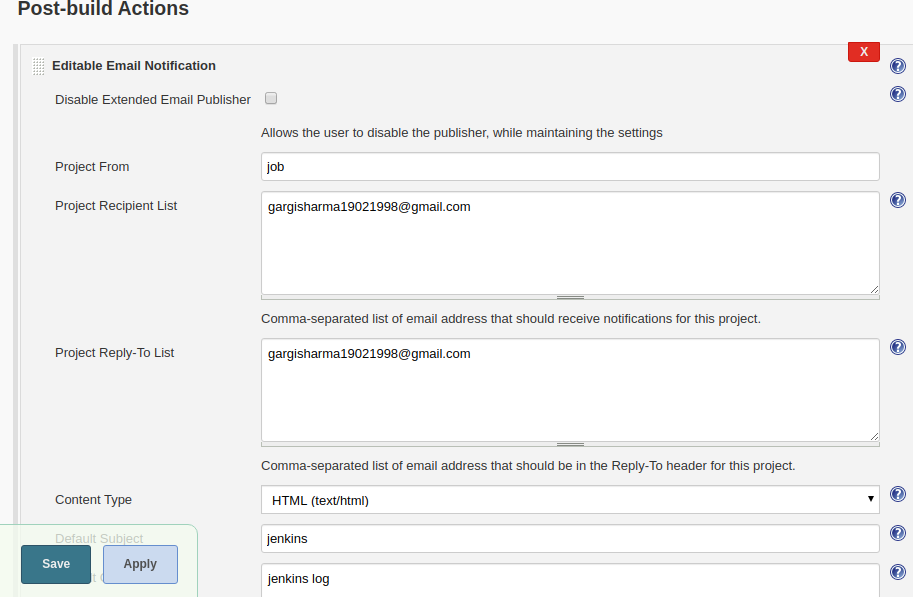
Create two string parameters for taking input as numbers(string)

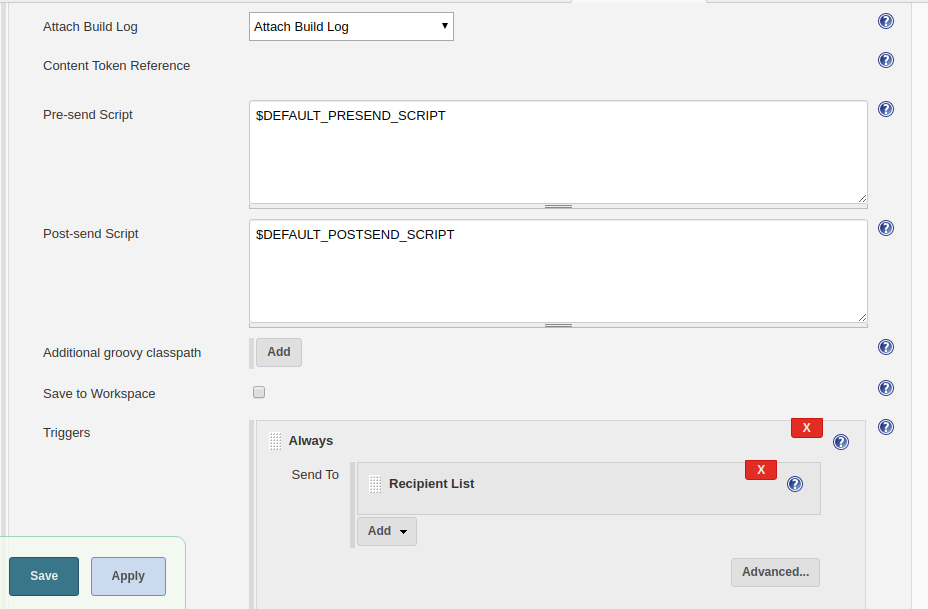


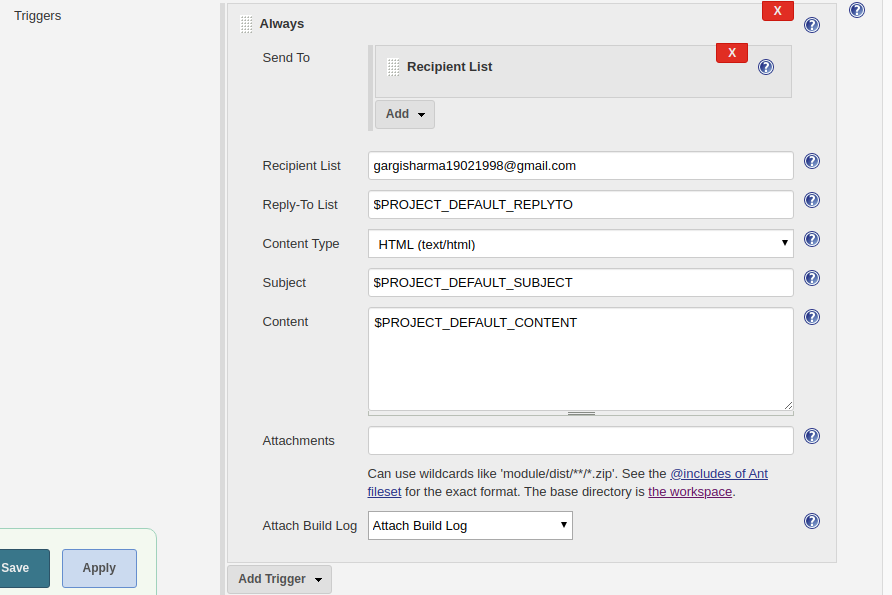
Execute shell in build steps.



In post build actions, edit editable email notification. Edit the advanced options also. Set the triggers to “Always”.

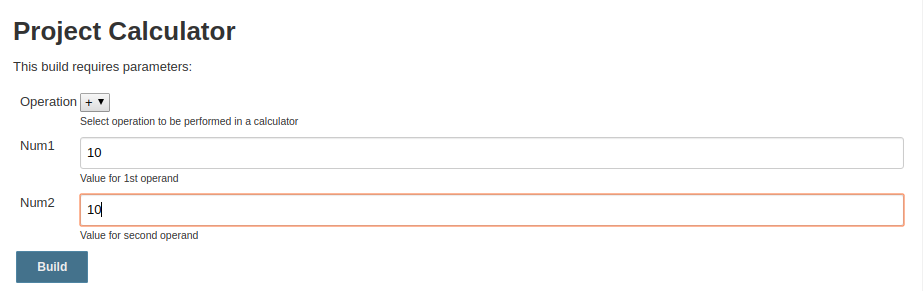


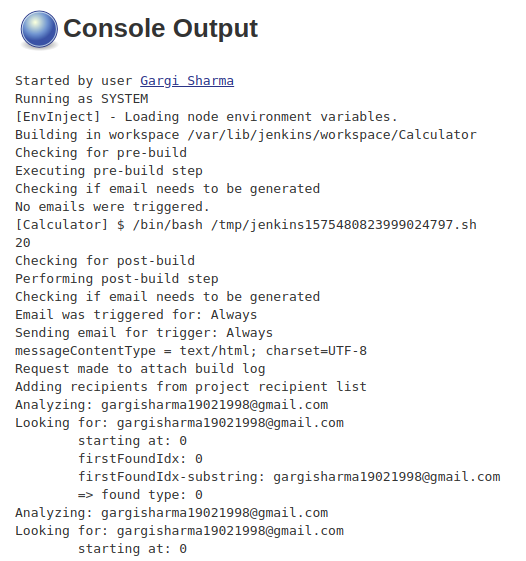




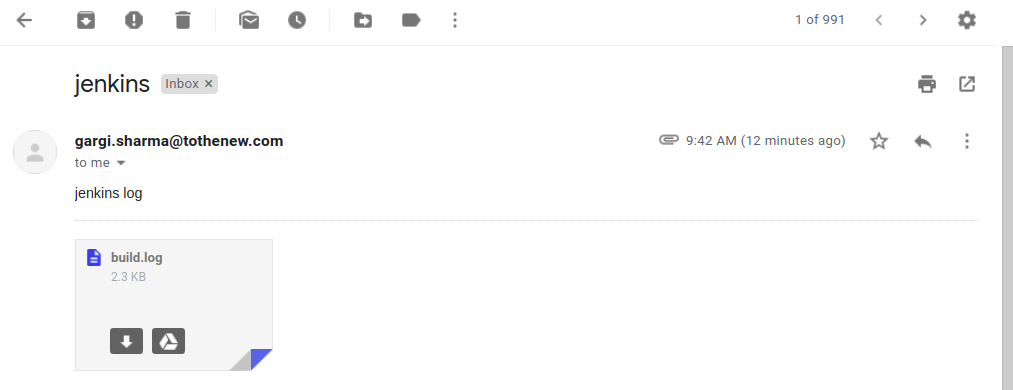
Now go to the sender email IDl ->.manage your account -> security-> turn on less secure app access and turn off two step verification.

Now start the build and enter the required parameters.



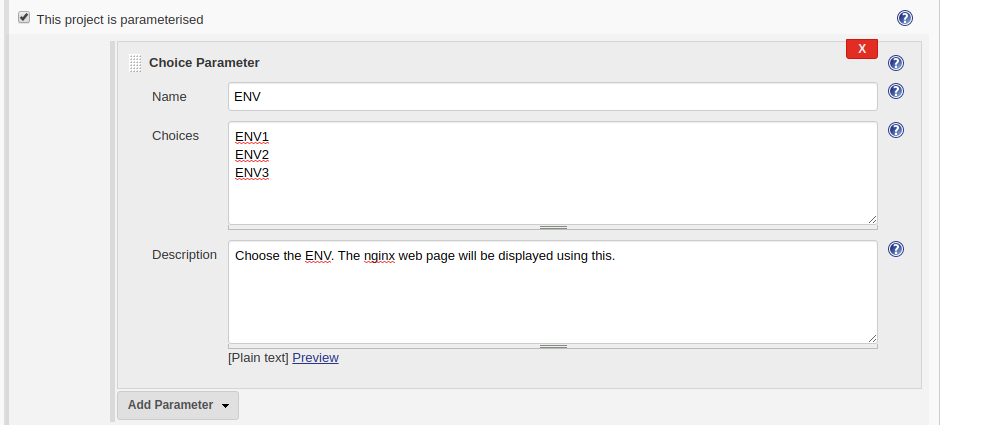


You can now check for the mail in mail IDs entered in the recipient's list.

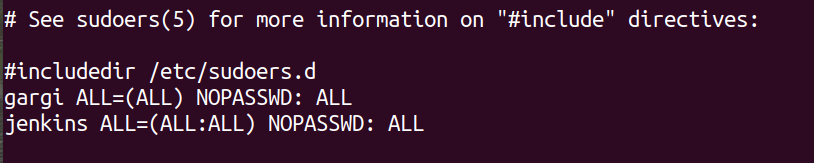


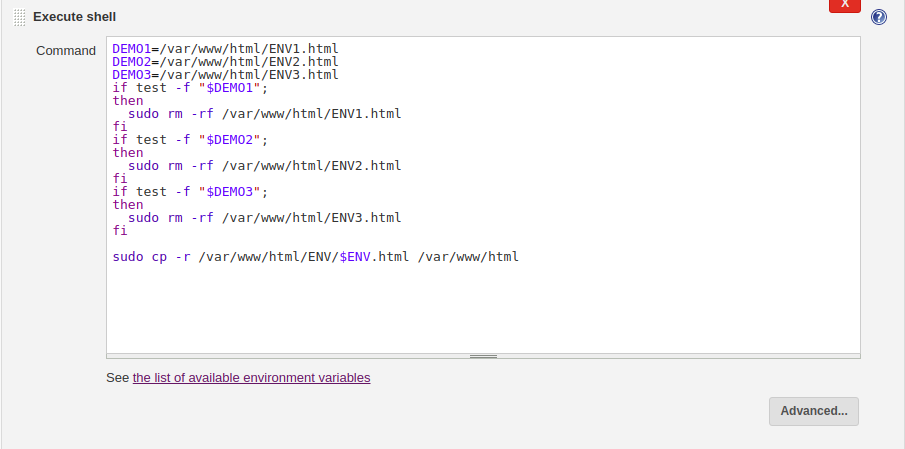
3. Create jenkins parameterized job which on selecting the different Env. will display different web pages by nginx.

Create a freestyle job with parameters and choose choice parameters.

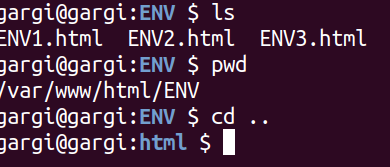


Now execute shell in build steps.

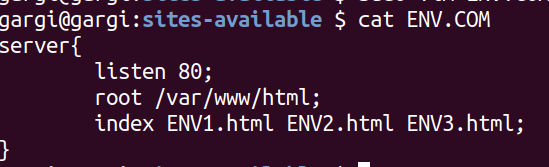


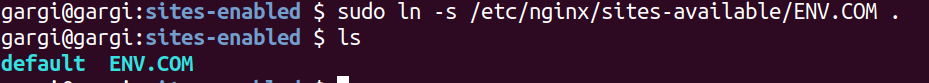


Now we create a directory in /var/www/html/ named ENV and create the 3 index files in it.

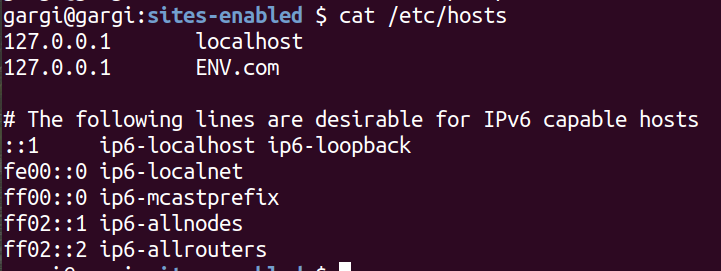


Now create a ENV.com in /etc/nginx/sites-available and create its soft link in sites-enabled which will contain the server and location block.



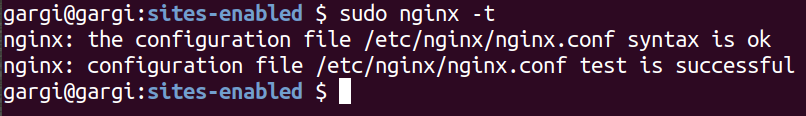


Now edit /etc/hosts file for ENV.com

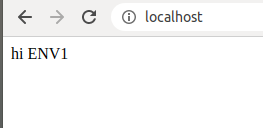


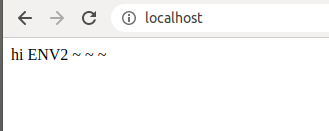
Now edit /etc/sudoers list and give jenkins full permission with no password. Then add jenkins in roo usergroup. Once done edit the file permissions(give 777) of all the index files and ENV directory.

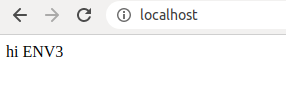
Now check if nginx configuration file syntax is ok.



Now restart nginx . Restart jenkins and build the job with parameters. When we choose ENV1 as the parameter, the nginx web page picks up the index file of ENV1 and displays it on the browser.

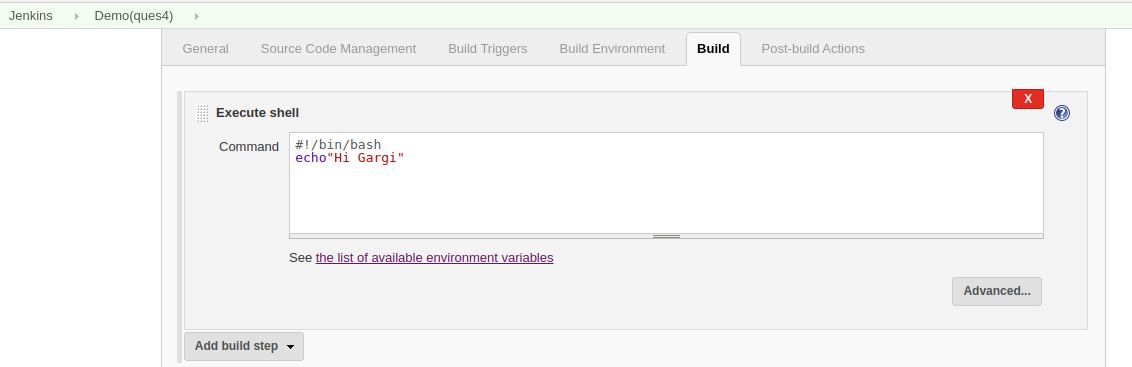




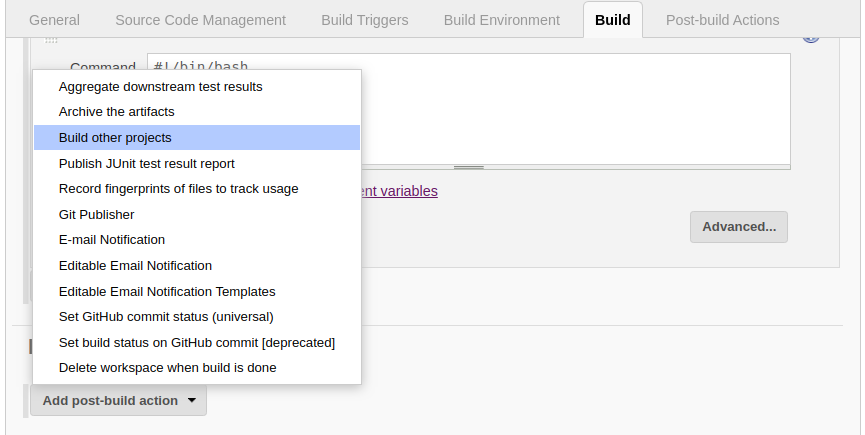


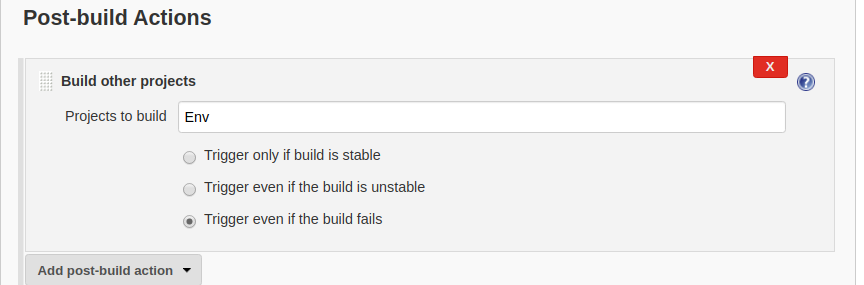
4. Create a job which on its failure will trigger another job.

Create a new job

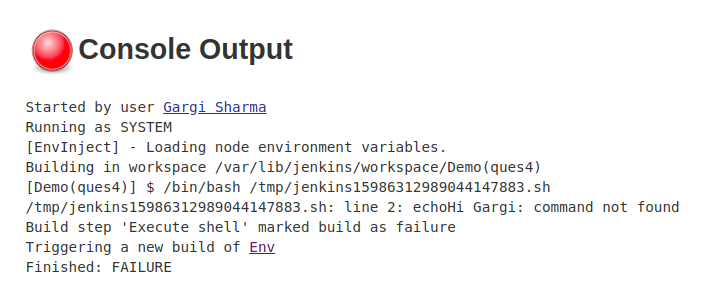


Add post build actions -> Build other projects.

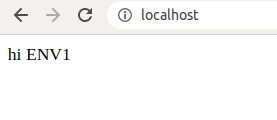




Job Fails:

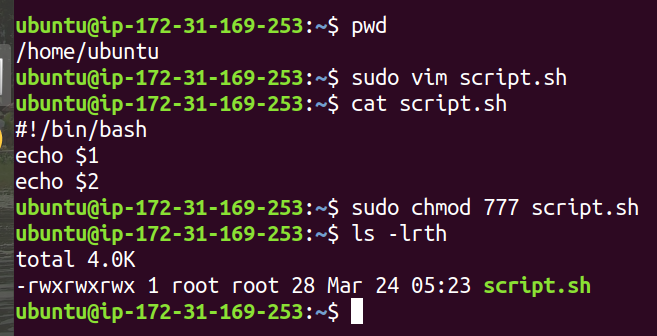


Job “Env” is triggered and jenkins takes ENV1 as default parameter and localhost shows ENV1 page.

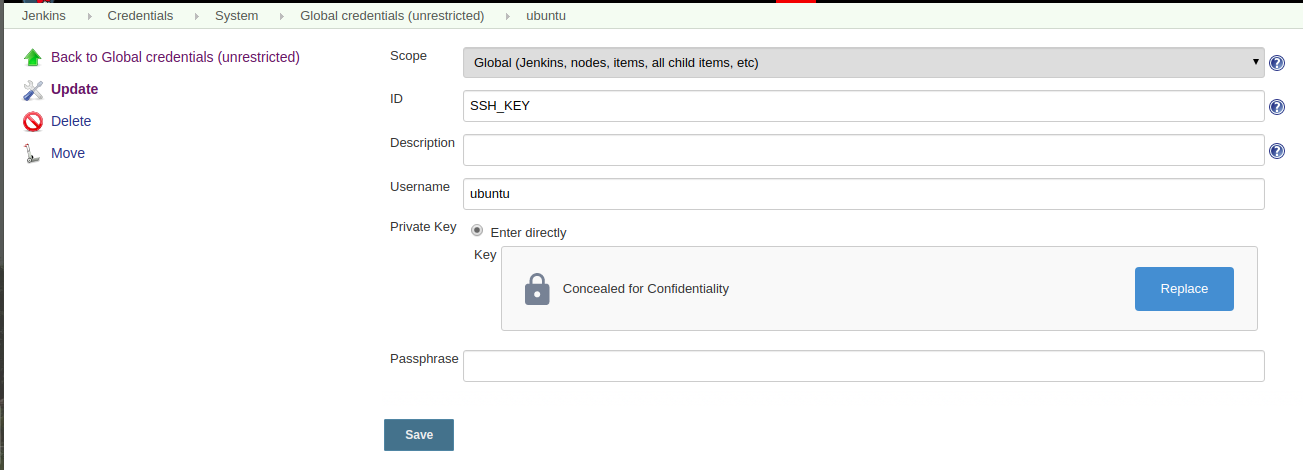


5. Create a job which can set a cron job on another server. This server contains a script on its home directory and the script will print the two string parameters which will be given by the jenkins job.

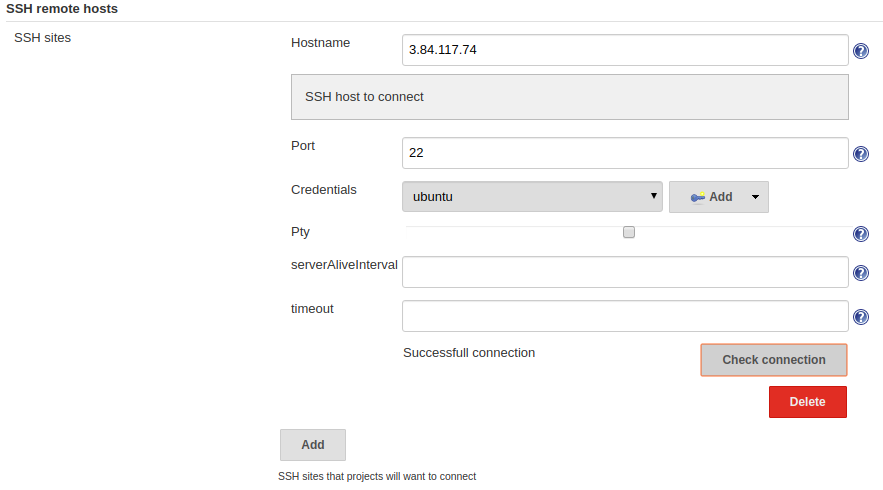
Launch an EC2 instance and ssh into it. Then Write a shell script in its home directory Also grant the script all permissions.



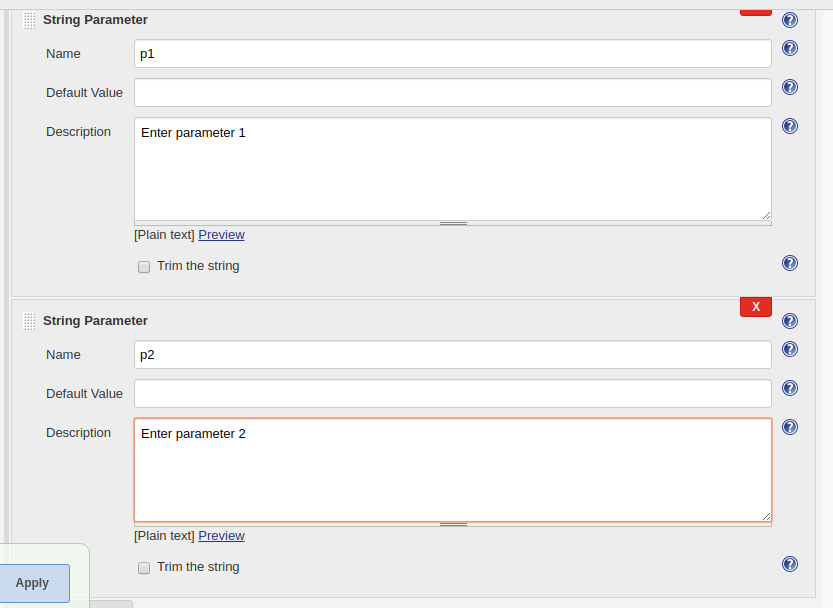
Now go to credentials and select SSH with a private key. Save the required details.



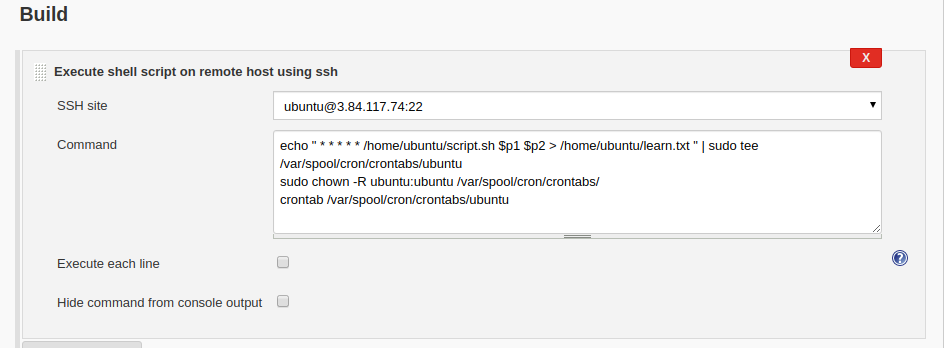
Now add remote server details in system configuration.



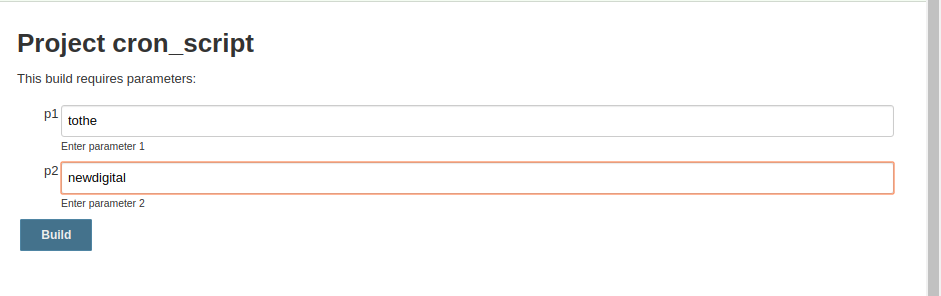
Now create a freestyle job with string parameters.

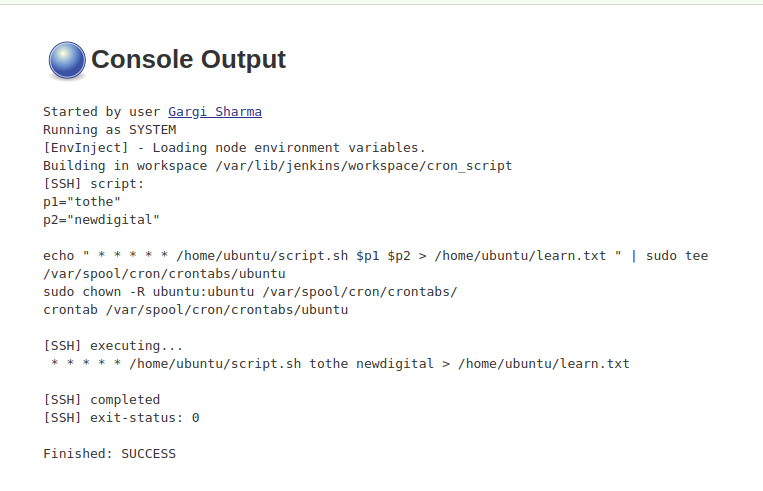


Go to build and execute shell on remote host using ssh..

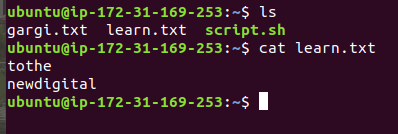


Start build with parameters.



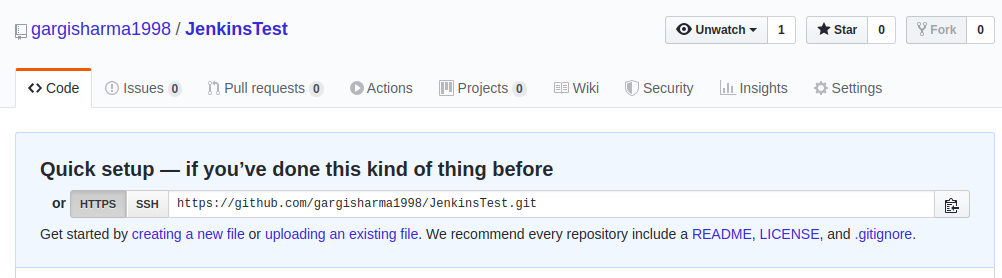


Execute crontab -l to list the cronjob.Now check after one minute on the remote server.

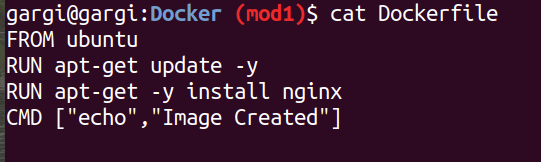


6. Create a job in which: Pull Dockerfile from GitHub, build it and push to Dockerhub. The docker image should have the tag: git commit id.

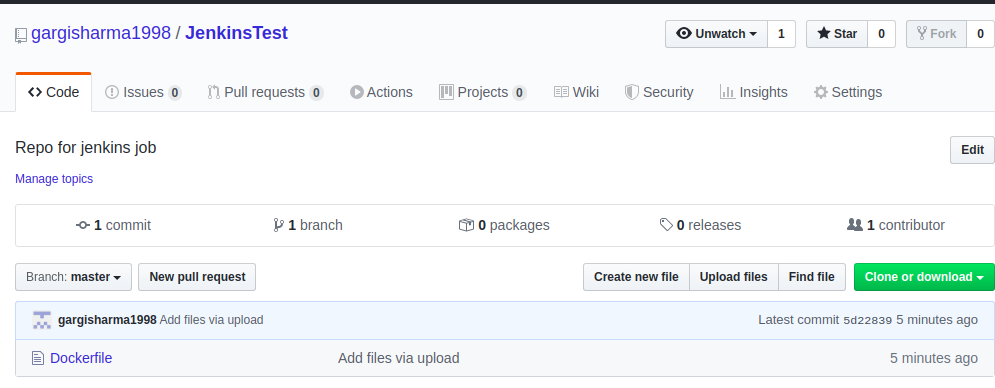
Create a new public github repository.



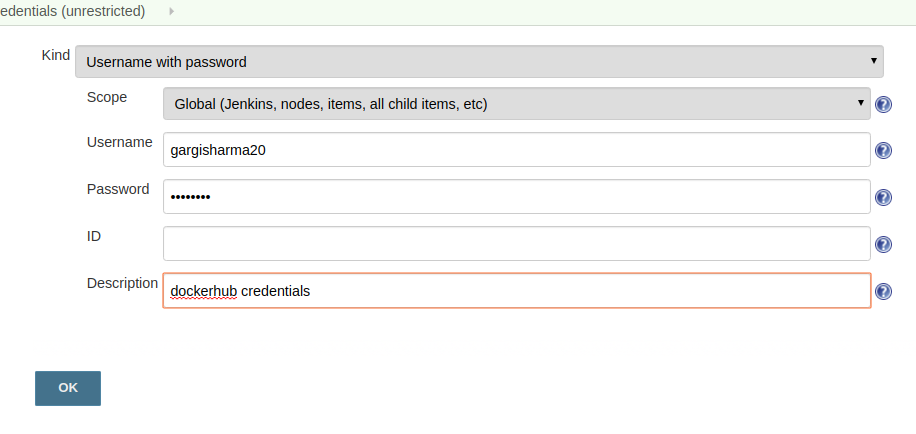
Create a Dockerfile



Push Dockerfile to Github repository.

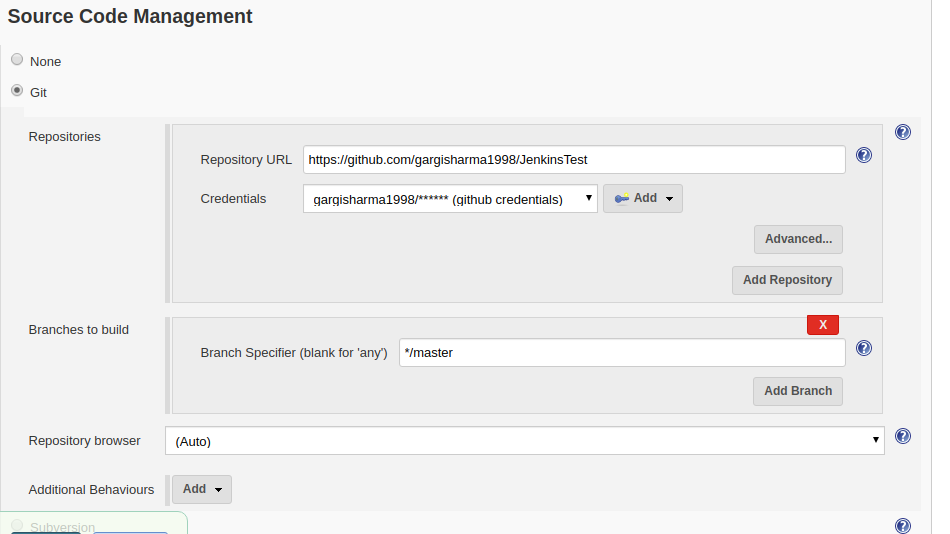


Go to jenkins-> credentials-> global -> create new credentials. Save the dockerhub credentials here.



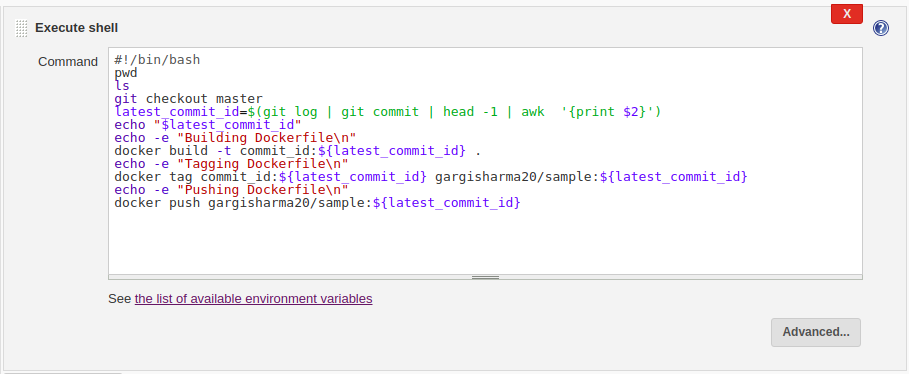
Create a new job. Execute shell

Ste SCM to git.

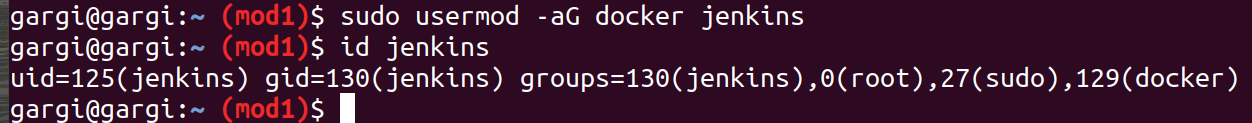


Tick the Delete workspace option before the build option.

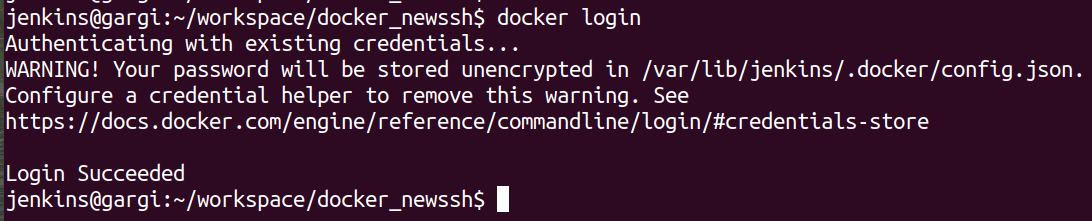
Then execute shell.



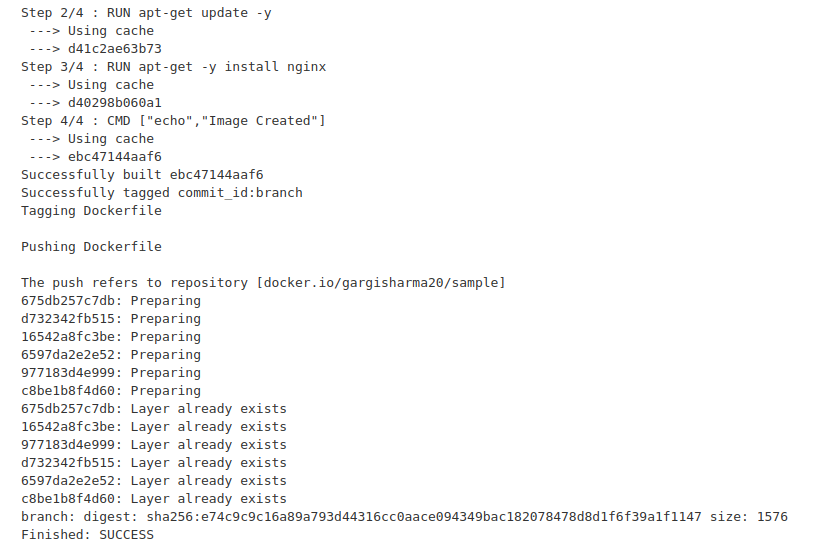
Now for jenkins to use docker, we add jenkins in the docker group.



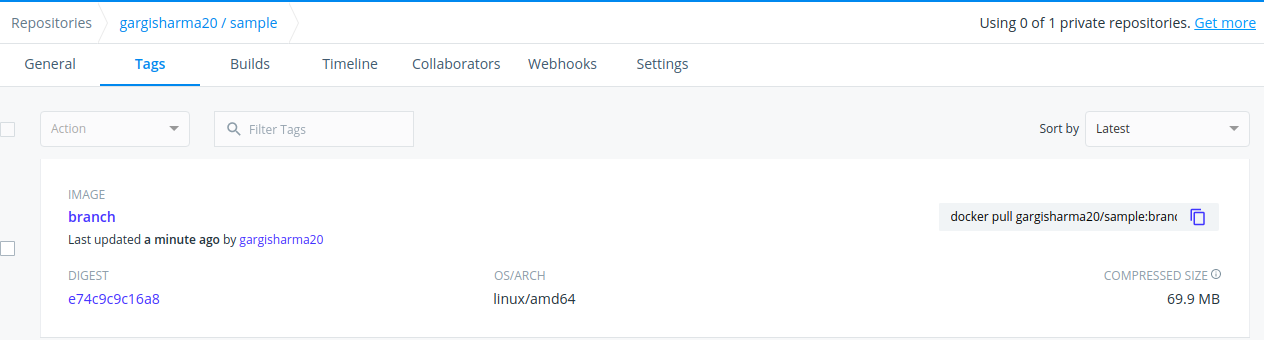
Now login docker in the jenkins user.



Once done, start the build.



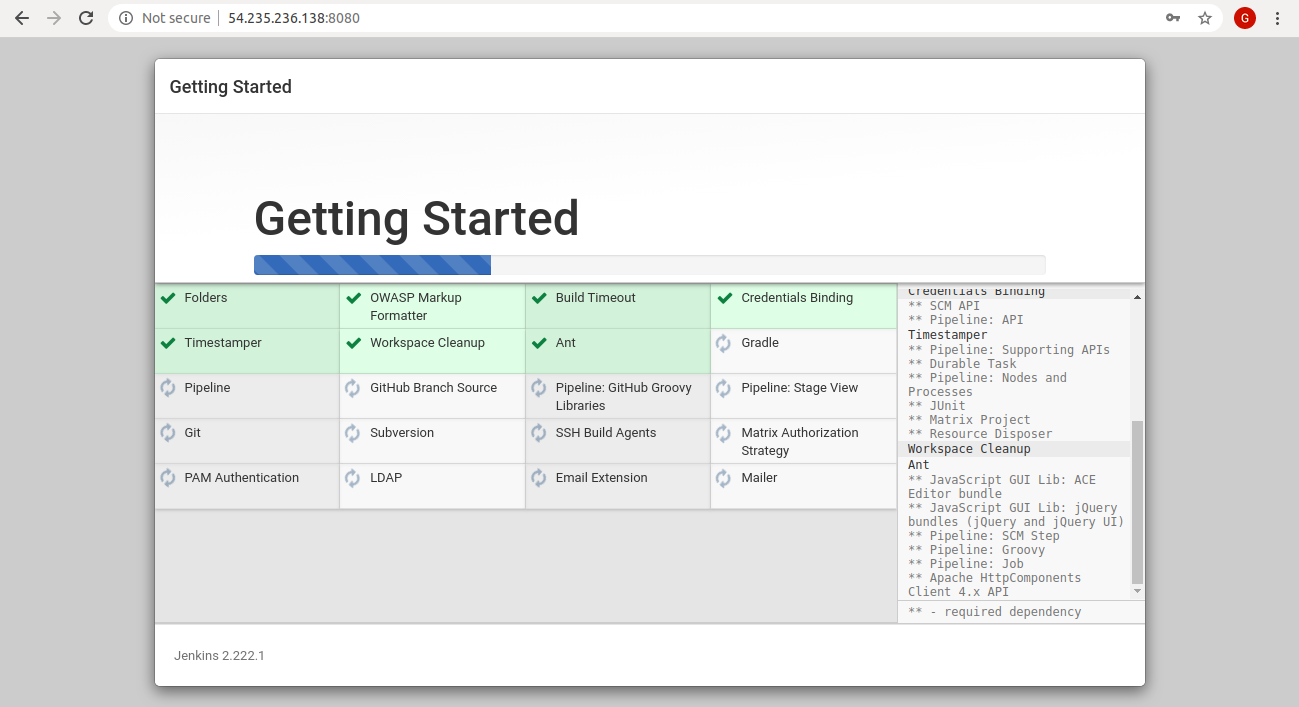
Now go to dockerhub and check for the image.



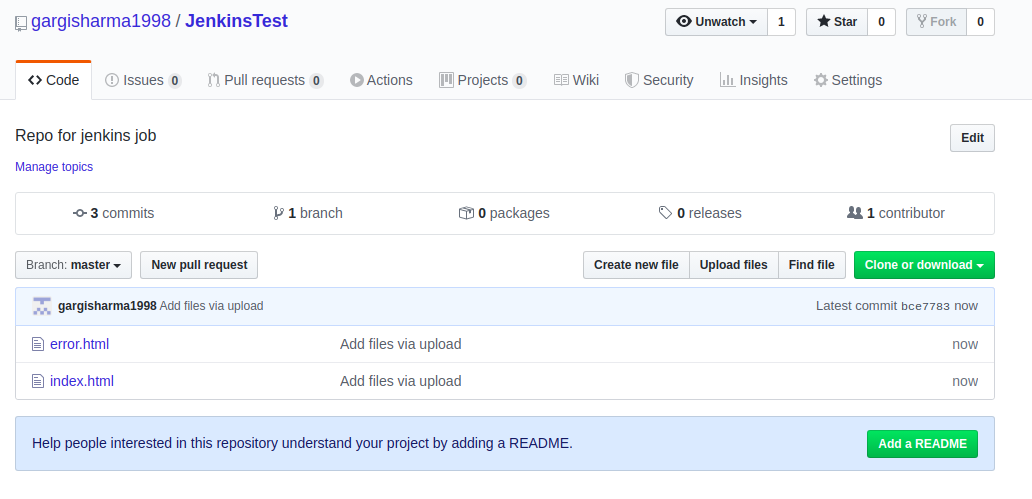
7. Host a static website on s3.

Its static content should be in git repo. When a person commits any change in the repo, the job should automatically reflect the changes in the s3 website.

Install jenkins on EC2 server.



Create two files index.html and error.html and upload it to a new git repo.



Create a public s3 bucket and host a static website there. Make the objects public and enable static website hosting. To make the bucket pubic, edit the bucket policy:

{

"Version": "2008-10-17",

"Statement": [

{

"Sid": "AllowPublicRead",

"Effect": "Allow",

"Principal": {

"AWS": "\*"

},

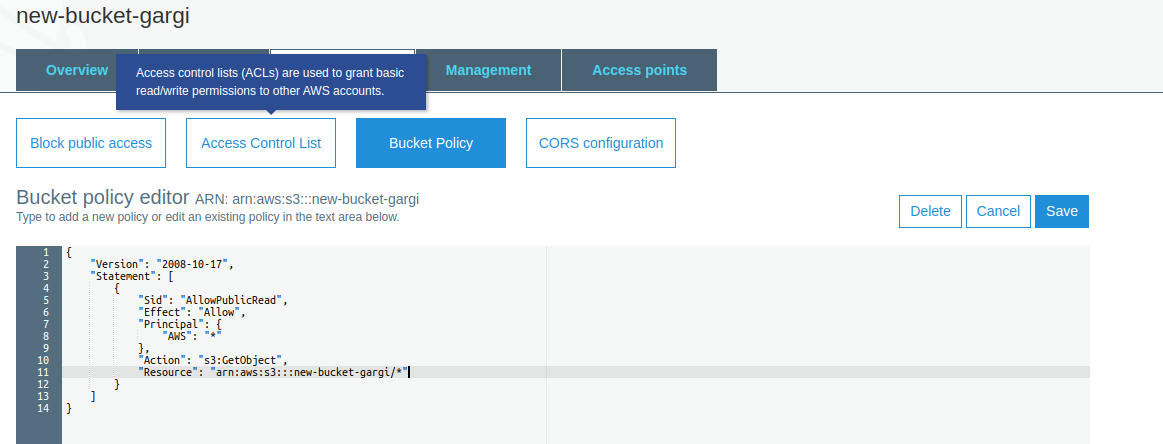
"Action": "s3:GetObject",

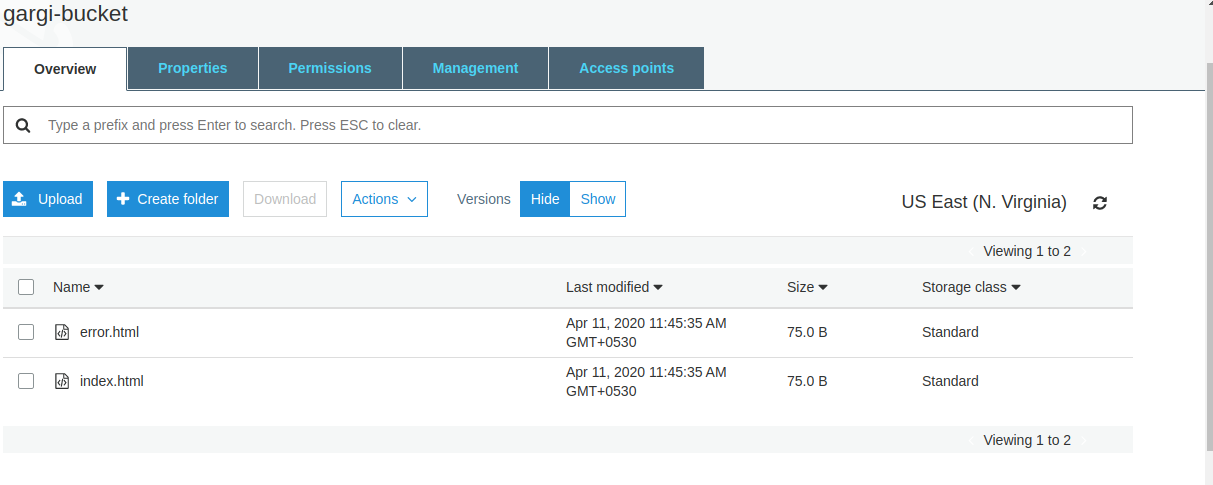
"Resource": "arn:aws:s3:::new-bucket-gargi/\*"

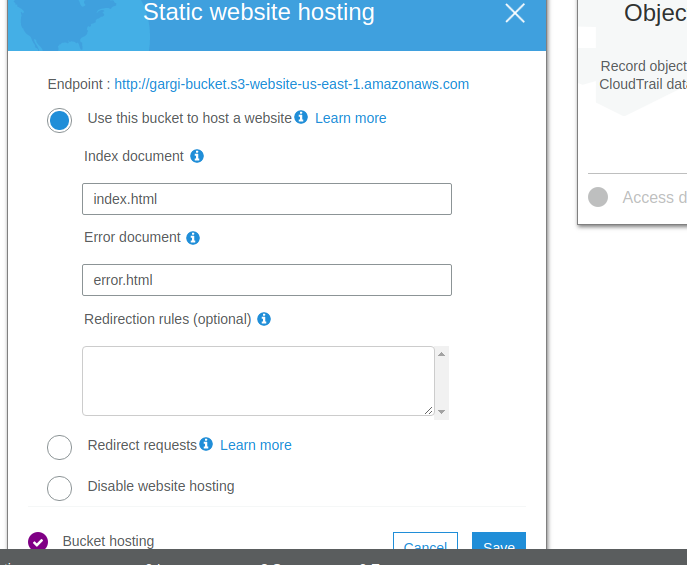
}

]

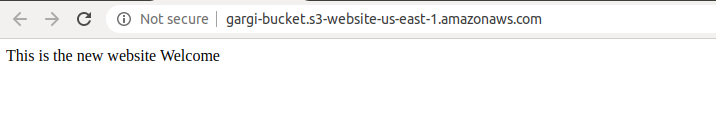
}



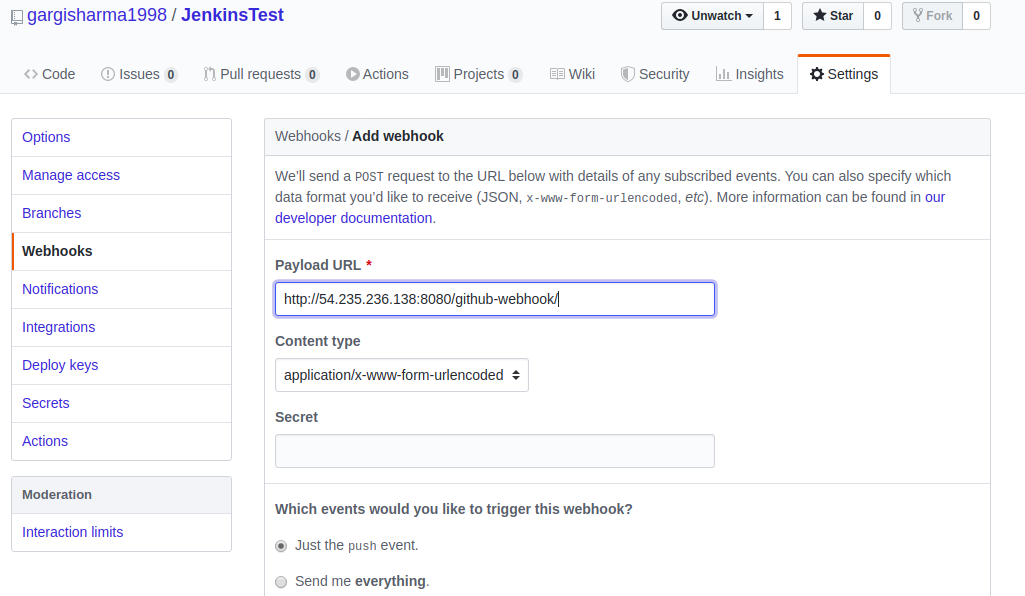


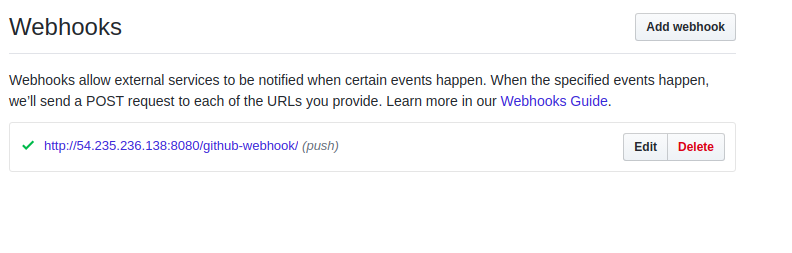


Check if the static website URL is working.

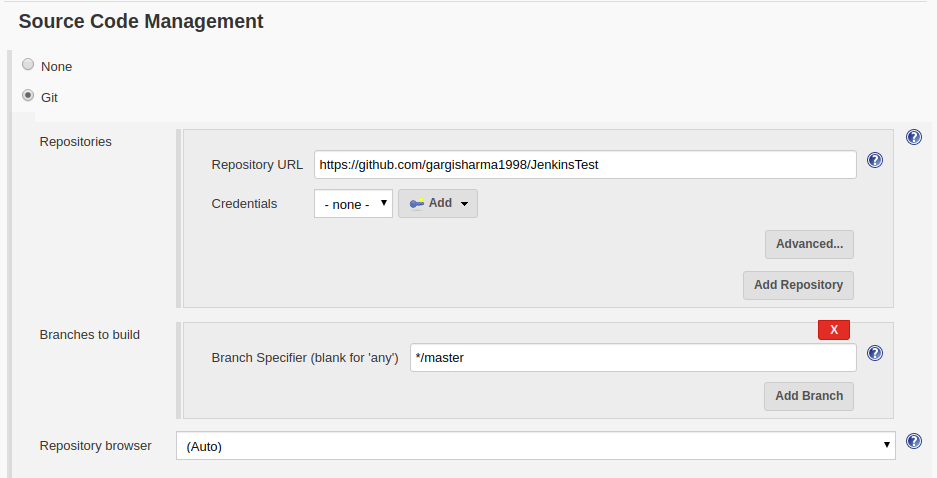


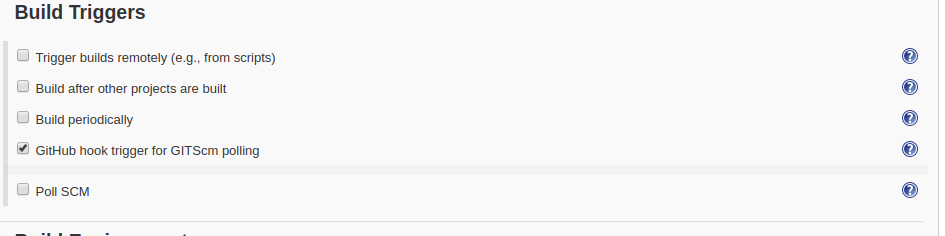
Activate webhook in the newly created git repository.

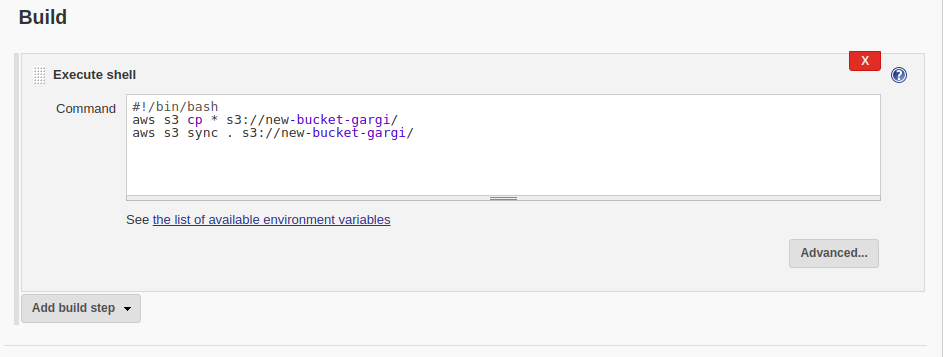




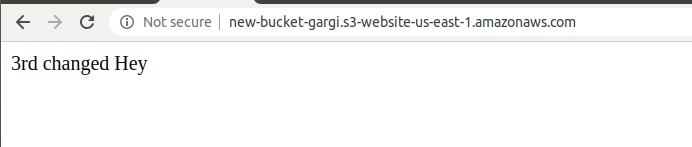
Create a jenkins job(freestyle)







Now click on the static website endpoint to see the initial website.



Now make some changes in index.html and push it again to the git repository. Jenkins job will start as soon as any push is made and will copy the new file to the s3 bucket.

Now check the static website endpoint again after few minutes.

