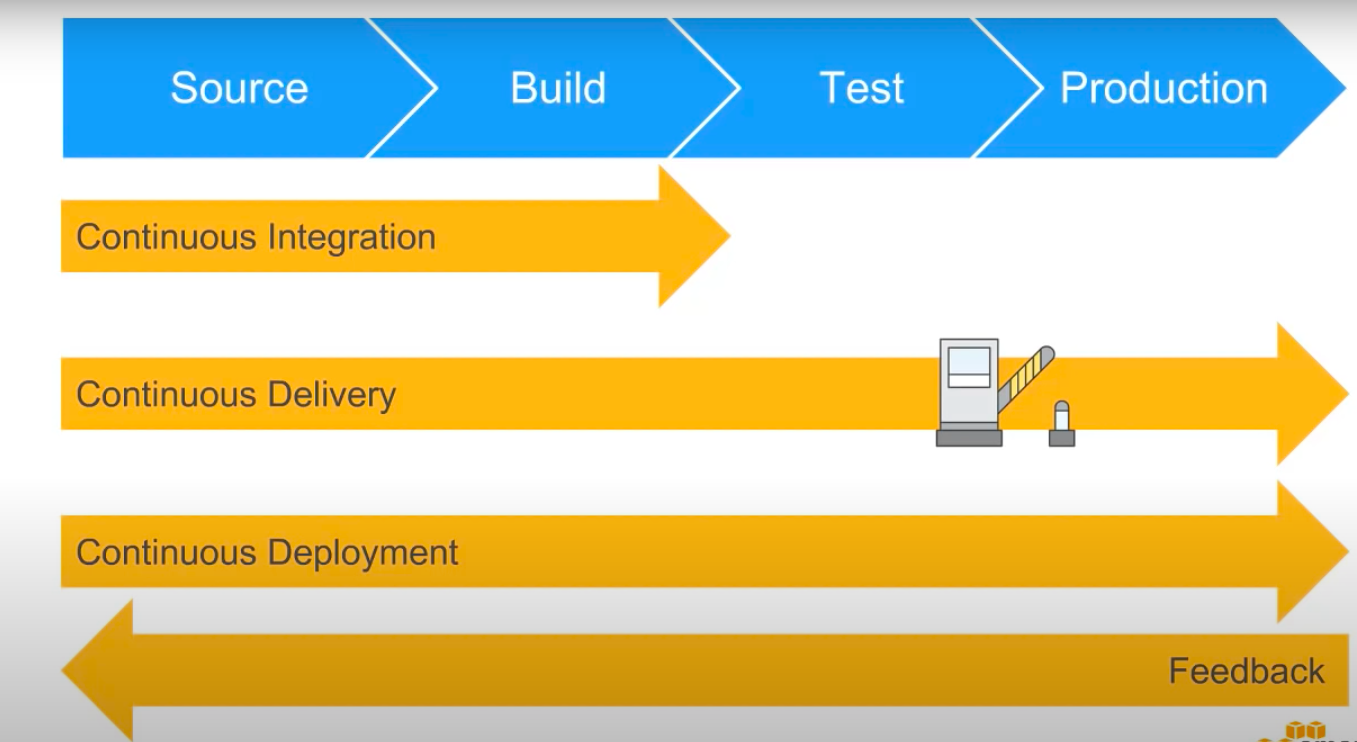
Automated CI/CD with Jenkins

CI-CD Pipeline implementation of the Continuous Integration/Continuous Deployment software is the backbone of the modern DevOps environment. Before moving onto the CI CD pipeline’s working, let’s start by understanding DevOps.

What is CI CD Pipeline?

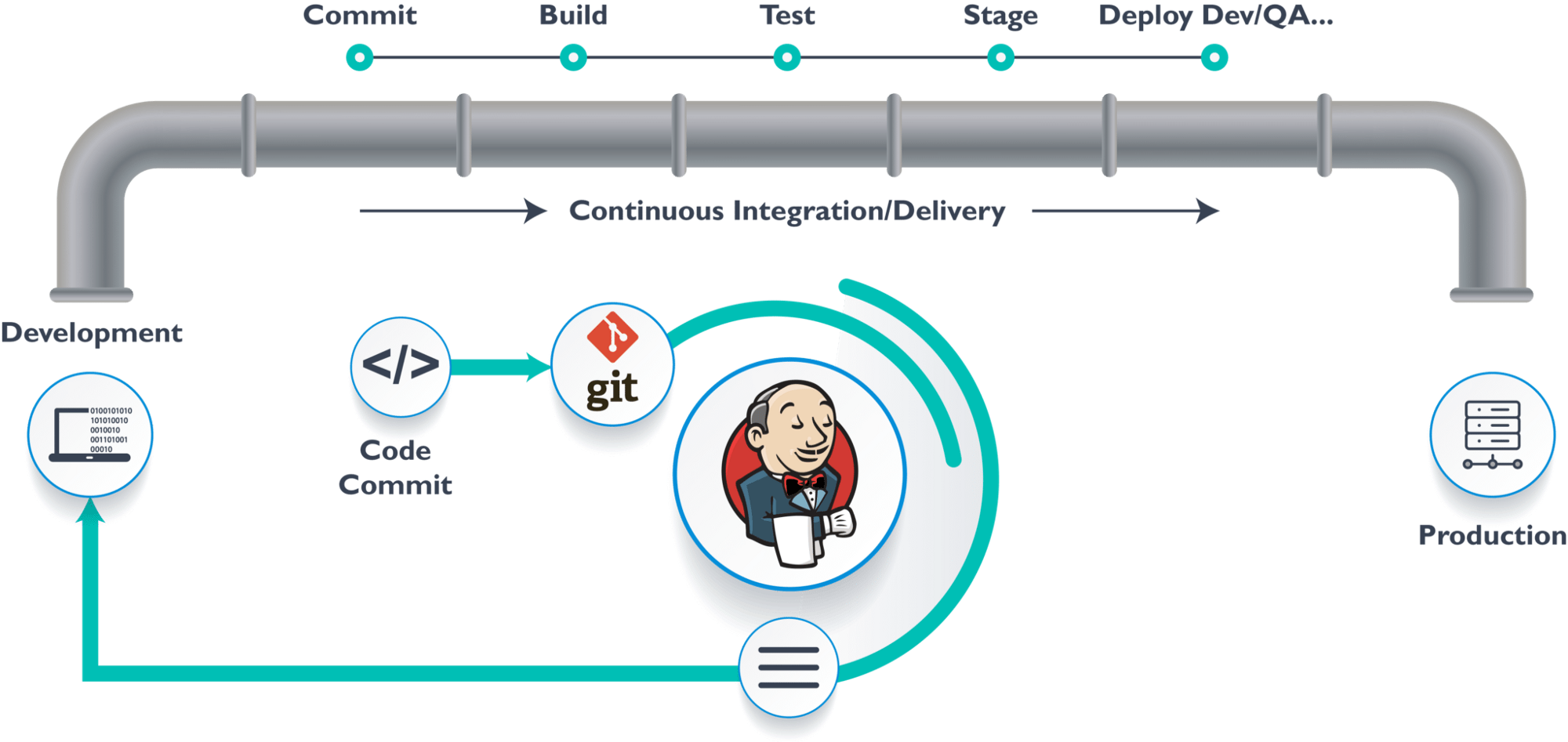
CI means Continuous Integration and CD means Continuous Delivery and Continuous Deployment. This is how SDLC works for dev-ops



Jenkins — The Ultimate CI Tool and Its Importance in CI-CD Pipeline

In this blog, we will try to automate entire Deployments with the help of Jenkins pipeline, Jenkinsfile using dockers

The whole development cycle will be based on the dev-ops model with the help of dev-ops tools as well



Jenkins provides us with various interfaces and tools in order to automate the entire process.

1. We will create a git repository or use an existing one
2. Dev team have the responsibility to commit the code to Dev-Branch
3. Jenkins will fetch the code from Github and will map with the job enabled for a specific task
4. We will make sure that CI and CD is done for the job/task
5. Jenkins will pull the code and will enter the commit phase of the task
6. Jenkins then will compile code and its called build phase of the task
7. The code is deployed by Jenkins after the code is merged to Master branch by DevOps team and the job is started for a specific application
8. The code is ready to be deployed and enter the deployment phase cycle
9. The code after deployed from Jenkins then get deployed to the server using docker container
10. After the code is working fine in staging server with unit testing, Same code then is deployed on the production server

With all the above 10 steps, Jenkins is responsible for the delivery phase with automating the deployment

Docker



Docker is simply sorted of virtual surroundings within which we will produce a server. It takes a couple of seconds to make a whole server and deploy the artifacts that we wish to check.

*Why do we use docker?*

As the same earlier, you’ll run the whole cluster in a very few seconds. we have a storage GitHub registry for images wherever you build your image and store it forever. you’ll use it anytime in any setting which may replicate itself.

Hands-On Lab:

Building CI CD Pipeline Using Jenkins and dockers

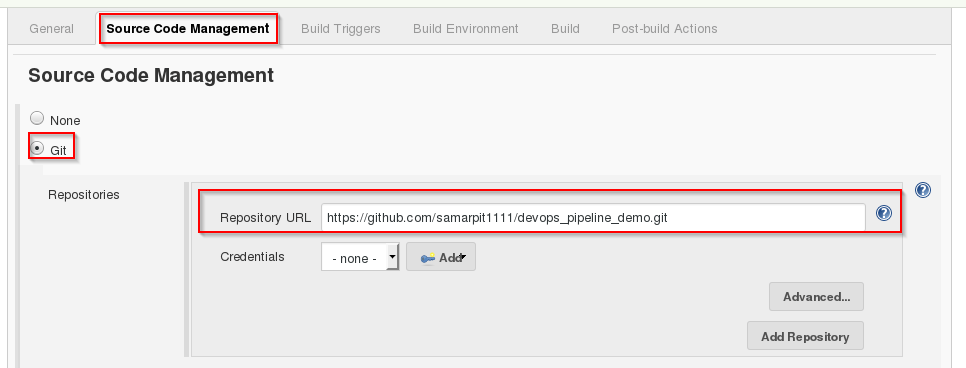
Step 1: In your Terminal or CLI, Start and enable Jenkins and docker

systemctl start jenkins  
systemctl enable jenkins  
systemctl start docker

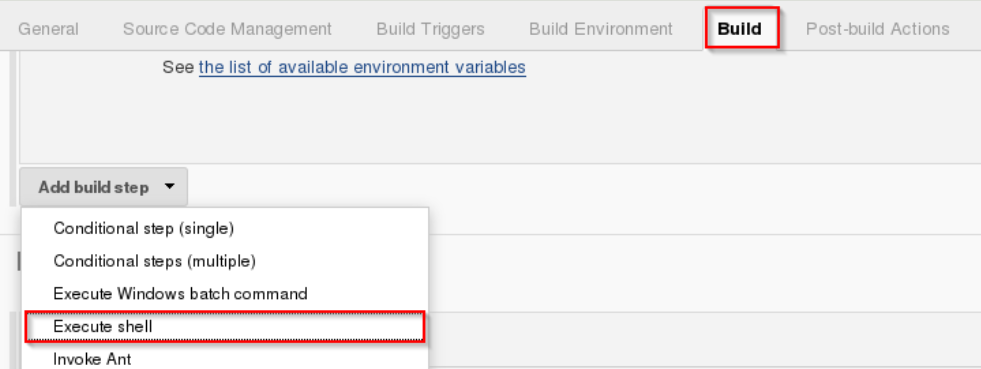
**Step 2:** In your Jenkins console click on **New Item** from where you will create your first job.

**Step 3:** After you click on New Item , You need to choose an option **freestyle** project with name and save

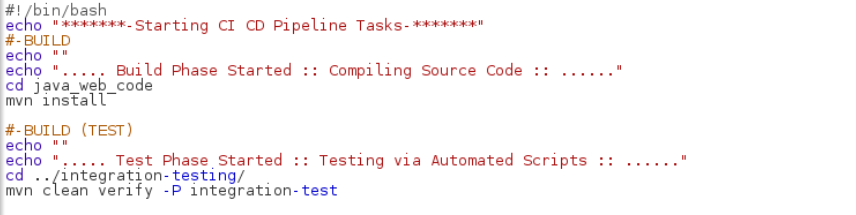
**Step 4:** In the configuration section select **SCM** and you will add the **git repo** link and save it.



**Step 5:** Then you will select **Build**option and choose to**Execute shell**



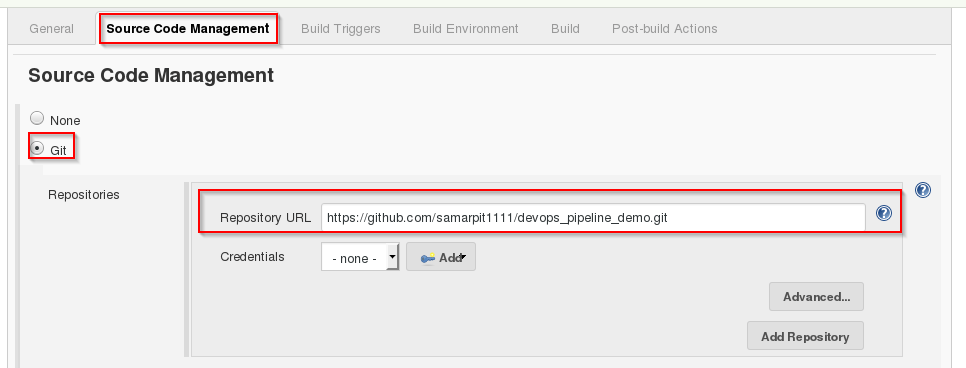
**Step 6:** Provide the shell commands. Here it’ll build the archive file to induce a war file. After that, it’ll get the code that already forces then it uses wiz to put in the package. So, it merely installs the dependencies and compiles the applying.



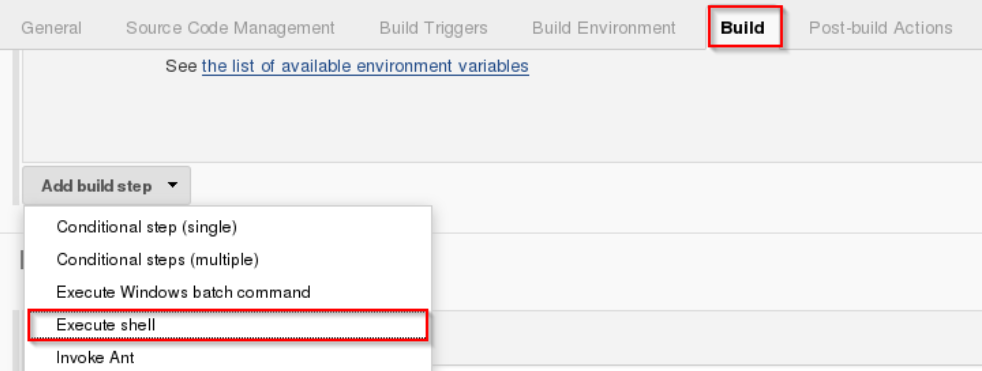
**Step 7:** Similarly you will create a new job as before.

**Step 8:** Click on the .**freestyle** project and save it with the proper name.

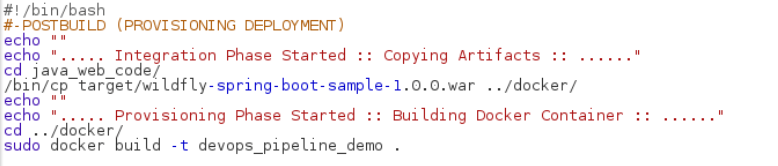
**Step 9:** Again repeat step 4, In the configuration section select **SCM** and you will add the **git repo** link and save it.



**Step 10:** Repeat step 5, You will select **Build**option and choose to**Execute shell**

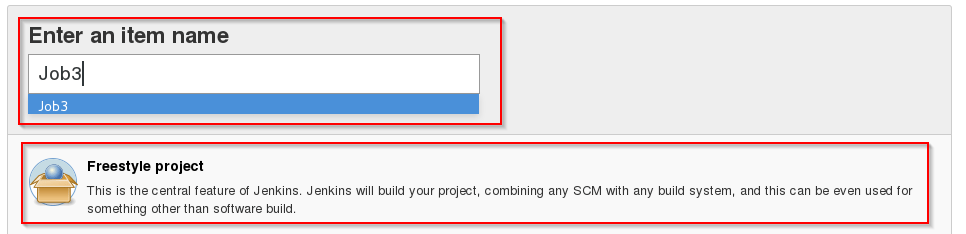


**Step 11:** You will now write the shell module commands as for int phase and build the container.

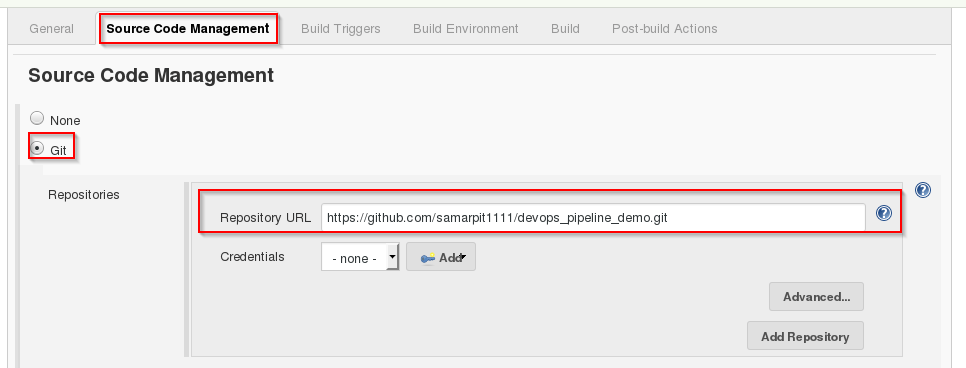


**Step 12:** Again you will create a new job as before in previous steps.

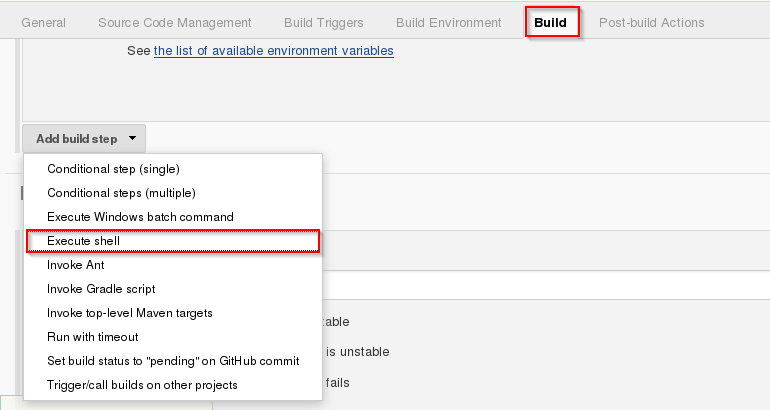
**Step 13:** Select **freestyle** project and provide the item name (here I have given Job3) and click on OK.



**Step 14:** Again repeat step 4, In the configuration section select **SCM** and you will add the **git repo** link and save it.



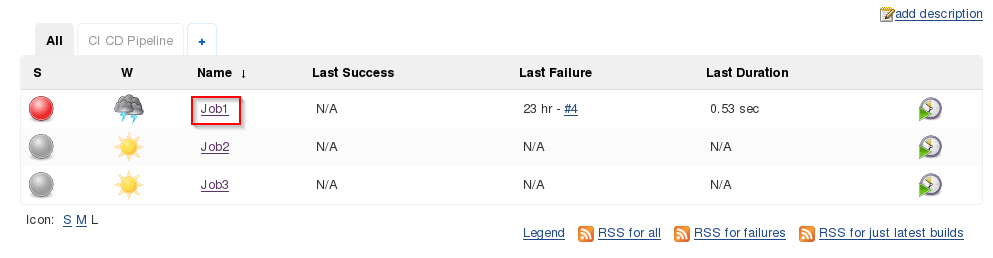
**Step 15:** Repeat step 10, You will select **Build**option and choose to**Execute shell.**



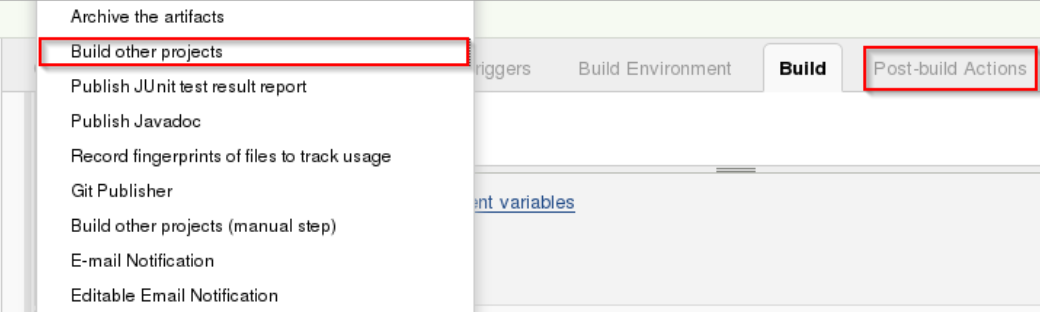
**Step 16:** Write the shell commands, Now it will verify the container files and the deployment will be doe on port 8180, save it



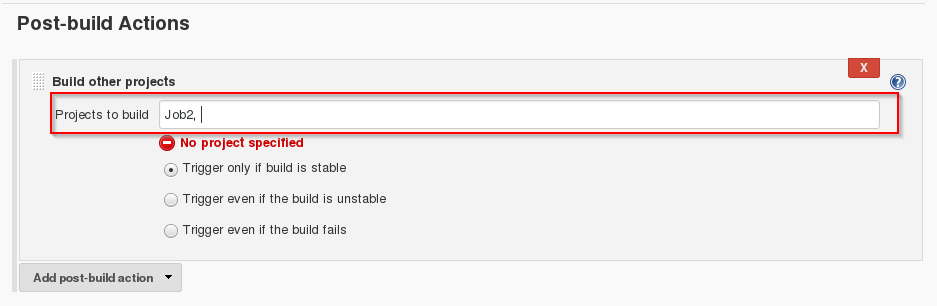
**Step 17:** Now, you will choose job 1 and click to configure.



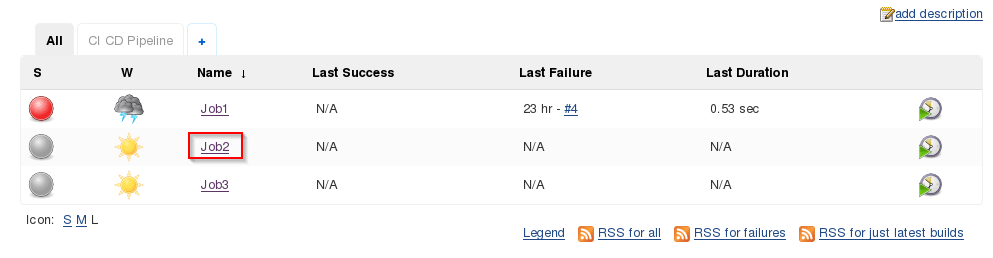
**Step 18:**From the build actions, You will choose **post-build**and click on build other projects

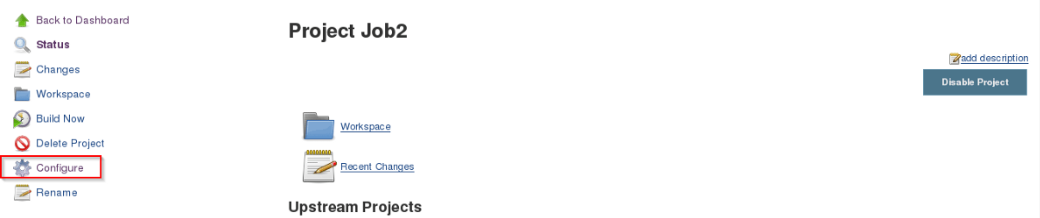


**Step 19:** You will need to provide the name of the project to build after the job 1 and then click **save**

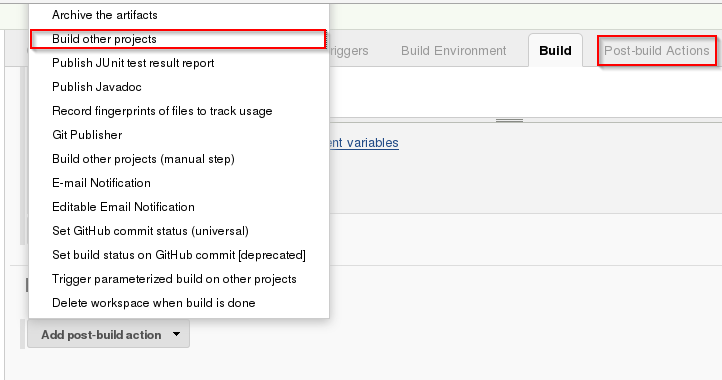


**Step 20:**Now, you will choose job 2 and click to configure.

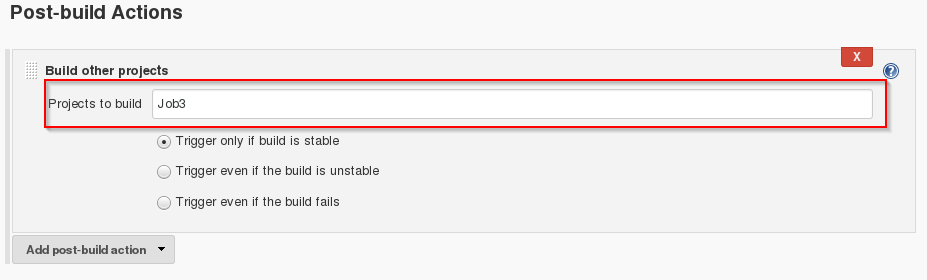




**Step 21:** From the build actions, You will choose **post-build**and click on build other projects



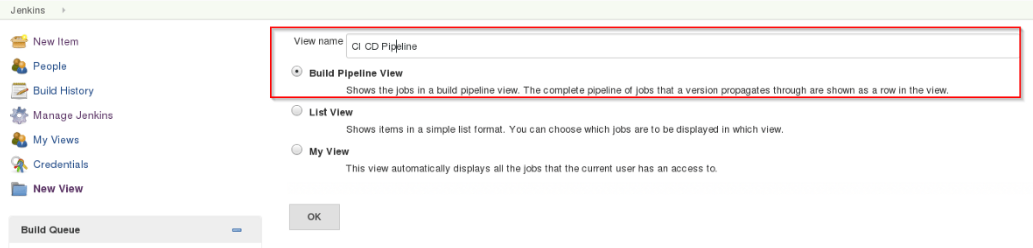
**Step 22:** You will need to provide the name of the project to build after the job 2 and then click **save**



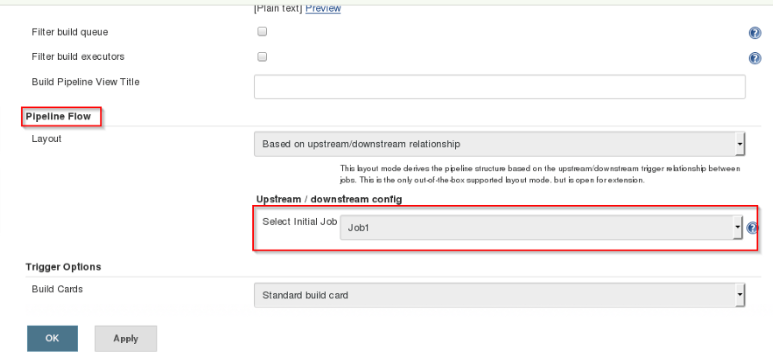
**Step 23:** let's create a pipeline, by adding to + sign



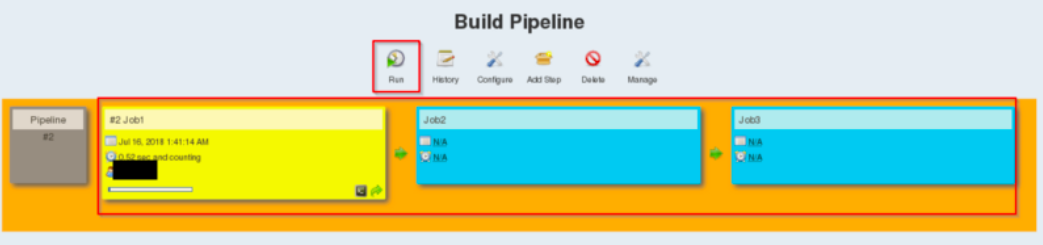
**Step 24:** Now, You will choose and select a build Pipeline view and add the name.



**Step 25:** Choose the Job 1 and save OK



**Step 26:** let's **RUN** it and start the CICD process now



**Step 27:**After you build the job, To verify open the link in your browser localhost:8180/sample.text, This is the port where your app is running

