Jenkins :Build Pipeline job,Create Docker image ,Push to AWS

**Goal:**

1. Build Service
2. Run BDD & TDD including Unit Tests
3. SonarQube Analysis (Static code analysis)
4. Build Container Image (Docker based)
5. Register Container Image in Amazon ECR (Push Docker Image)
6. Deploy Container Image in Amazon ECS/Fargate

**STEP 0. Pre-Requisites:**

For the next Steps:

1. Create an AWS instance
2. Have a Jenkins Set up inside the aws linux box( 2.132)
3. Have java installed
4. Have maven integrated in Jenkins(used installed automatically in Jenkins:3.5.4)
5. Have SonarQube installed
6. Have docker installed in the server.

**STEP 1. Install Jenkins and Jenkins Plugin**

Open your favorite browser and navigate to Jenkins. Log in and select “**Manage Jenkins**” followed by “**Manage Plugins**”. Select the “**Available**” tab, locate the “[**Deploy to container**](https://wiki.jenkins-ci.org/display/JENKINS/Deploy+Plugin)” plugin and install it.

**Pre requisite plugins:**

1. Git plugin

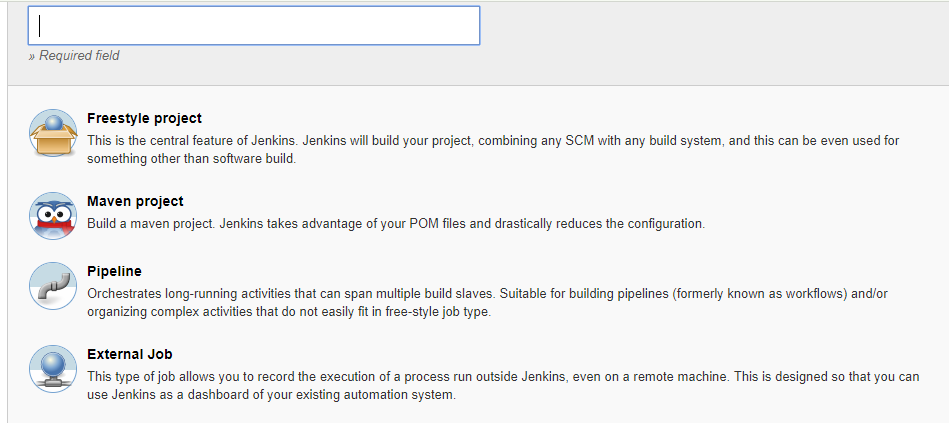
**Extra Plugins to be installed along with custom plugins:**

1. Amazon EC2 plugin
2. Amazon ECR plugin
3. Build Pipeline Plugin
4. Docker Build and Publish plugin

**STEP 2. Create Jobs for each module.**

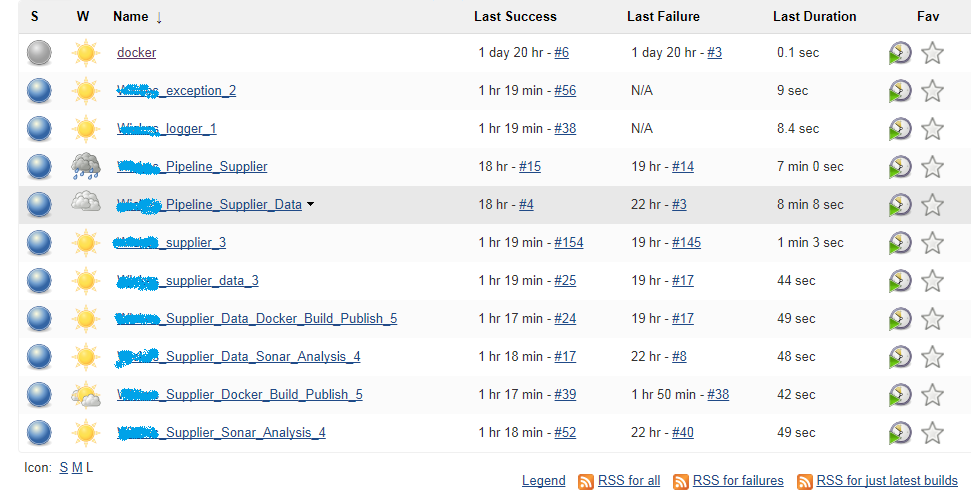
* Create jobs for the projects.(The projects are dependent to each other)

Jenkins, go to your job and select “**New Item**”. Select the Type of job to be built.Eg:The project here is a maven project.



Here ,**Create jobs** for

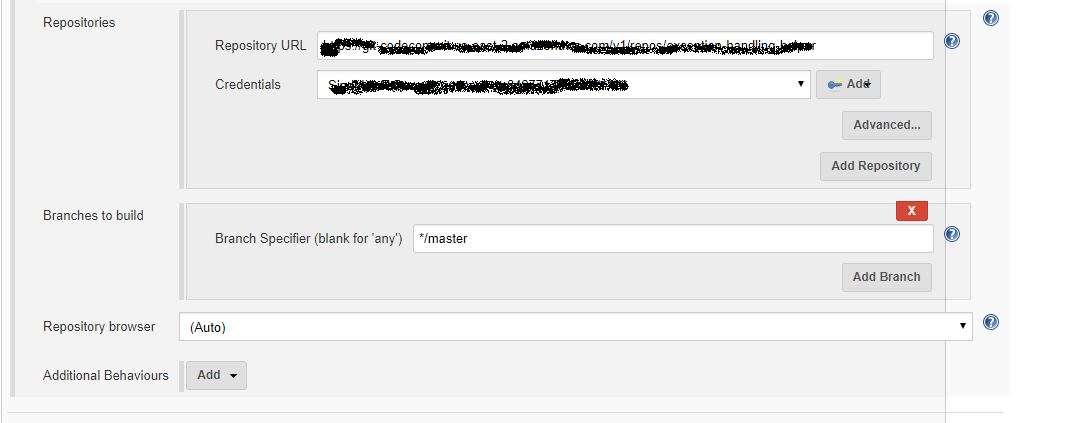
1. Logger\_helper\_11 ,exception\_handler\_helper2
2. Supplier-service-api\_3 and its Sonarqube\_Analysis\_4, Docker\_build\_and\_publish\_5
3. Supplier-data-api\_3 and its Sonarqube Analysis\_4, Docker build and publish\_5



* Back in Jenkins,go to your job and select “**Configure**”.Under General Tab Provide the description of the project.
* Under the “**Source Code management”** ,Select the Required scm. Eg: here Git

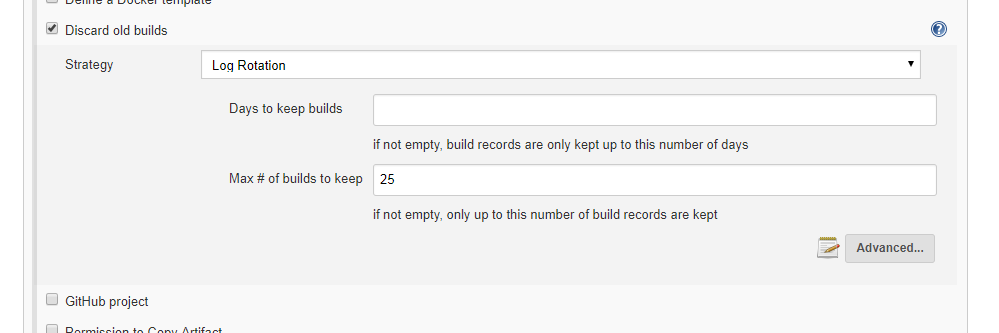
Fill in the Details :

* **Repositoryurl** : [the](URL:the) git repo url of the project to build.
* **Credentials**: Provide the git credentials
* **Branch** **Specifier** :Specify the name of the branch from where thw code needs to be pulled.(**Note** :Specify the same name give in the git since it is case–sensitive (**REF ERROR 1**)



This Pulls the code from the git repo and places it in the workspace inside Jenkins.(Check for cross verification if code is pulled properly).

* Check the “**Discard old builds**” under general tabto have only latest 25 builds**.(optional)**



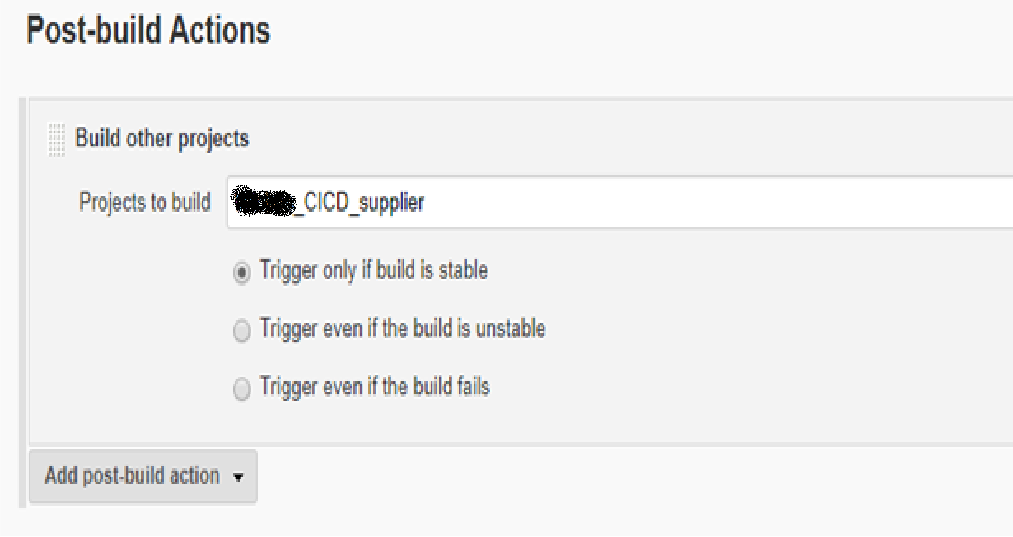
* In “**Configure**”, Provide the goals for the jobs as per the module to be built.This is specified in each of the module steps.
* **To create a pipeline job.**

In the “**Configure**” of the jobs provide the job to be built next in the **“Post build actions”** and **“Build Triggers”.**

**Post-build Actions:**

Here We can specify the job to be triggered in **“Projects to build”** for build after current project is built.

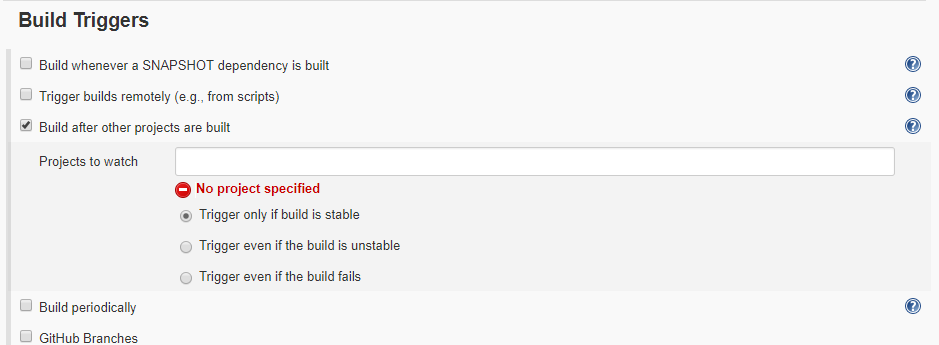
It provides 3 options to trigger a build :checks if the current build is stable, unstable, or failed.



**Build Triggers:**

This triggers current job only after the specified jobs in the “**Project to watch**” is built.

It provides 3 options to trigger a build :checks if the current build is stable, unstable, or failed.



Here , **Link the jobs as**

**For supplier\_api\_service:**

logger\_helper\_1🡪Exception\_handling\_helper\_2🡪Supplier\_api\_service\_3🡪Sonarqube\_Analyis🡪Docker\_build\_and\_publish\_5

**For supplier\_data\_service:**

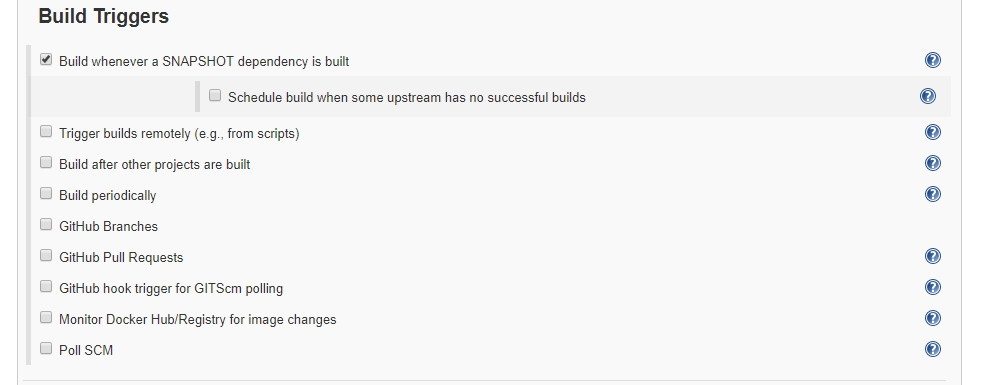
logger\_helper\_1🡪Exception\_handling\_helper\_2🡪Supplier\_data\_service\_3🡪Sonarqube\_Analyis🡪Docker\_build\_and\_publish\_5

* **To Avoid issues occurring in chain job**

In the “**Build Triggers tab**” inside configure,except for

1. Supplier\_api\_service
2. Supplier\_data\_Service

Uncheck the **“Build whenever a SNAPSHOT dependency is built”**.This is mainly to chain the jobs properly without any dependency issue since the logger-helper and exception-handling-helper are added ans dependency in pom.xml of Supplier\_api\_service and Supplier\_data\_Service.



**STEP 2. Build Service (Build a chain job)**

* Back in Jenkins,go to your job and select “**Configure**”.Under General Tab Provide the description of the project.
* Goto Build tab, In the “**Goals and options**” provide **clean install** .This creates a jar for the job being built.

This is provided in :

1. logger-helper\_1
2. exception\_handler\_helper\_2
3. Supplier\_api\_service 3
4. Supplier\_data\_service\_3



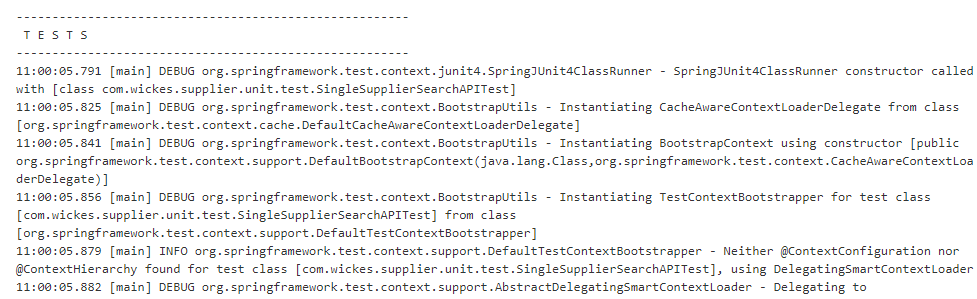
**STEP 3. Run BDD & TDD including Unit Tests**

**BDD- Behavior-Driven Development**

**TDD-Test Driven Development**

1. Start by writing a test
2. Run the test and any other tests. At this point, your newly added test should fail. If it doesn’t fail here, it might not be testing the right thing and thus has a bug in it.
3. Write the minimum amount of code required to make the test pass
4. Run the tests to check the new test passes
5. Optionally refactor your code

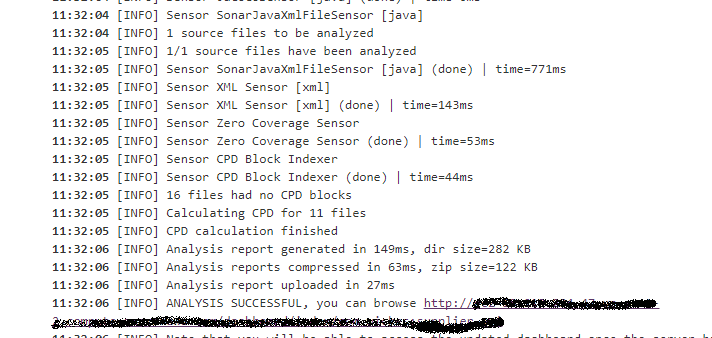
Hence Tests are integrated inside the test folder of the code.To run the test during a build DON’T PROVIDE : -**DskipTests=True**

****

**STEP 4. SonarQube Analysis (Static code analysis)**

Back in Jenkins, go to your job and select “Configure”. Goto Build tab, In the “Goals and options” provide the command:

**sonar:sonar -Dsonar.host.url=http://ec2-xx-xxx-xxx-xx.us-east-2.compute.amazonaws.com -Dsonar.login=xxxxxxxxxx**

****

**STEP 5. Build Container Image (Docker based)**

1. **Create a Docker file and place it inside the project where pom.xml is present.**

FROM java:8

VOLUME /tmp

ARG JAR\_FILE=target/supplier-api-0.0.1-SNAPSHOT.jar

ADD ${JAR\_FILE} app.jar

EXPOSE 8080

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]

1. **Add the Docker plugin in the pom .xml of the project .**

<plugin>

<groupId>com.spotify</groupId>

<artifactId>dockerfile-maven-plugin</artifactId>

<version>${docker.plugin.version}</version>

<configuration>

<repository>${docker.image.prefix}/${project.artifactId}</repository>

<buildArgs>

<JAR\_FILE>target/${project.build.finalName}.jar</JAR\_FILE>

</buildArgs>

</configuration>

</plugin>

1. **METHOD 1**: Goto Build tab, In the “Goals and options” provide: **package dockerfile:build**

****

**Or**

**METHOD 2:**

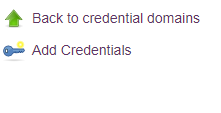
Use the **Docker build and publish plugin** configuration as specified below in step 6 to build and push the image.

**STEP 6. Register Container Image in Amazon ECR (Push Docker Image)**

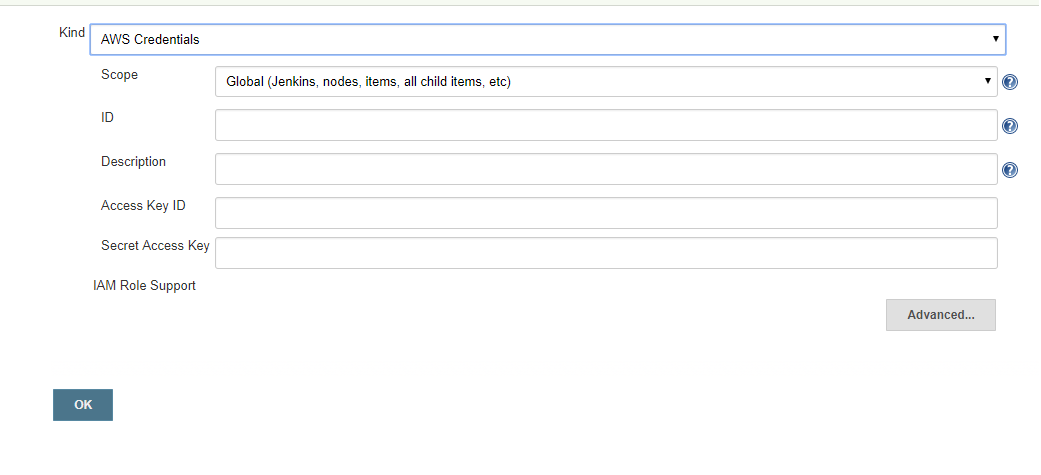
**Pre-Requisite:**

1. Create the **repository** in the ECR
2. Add credentials to jenkins

* Add the credentials of AWS:
* From the Jenkins home page goto credentials. Select global credentials and add a new credential.

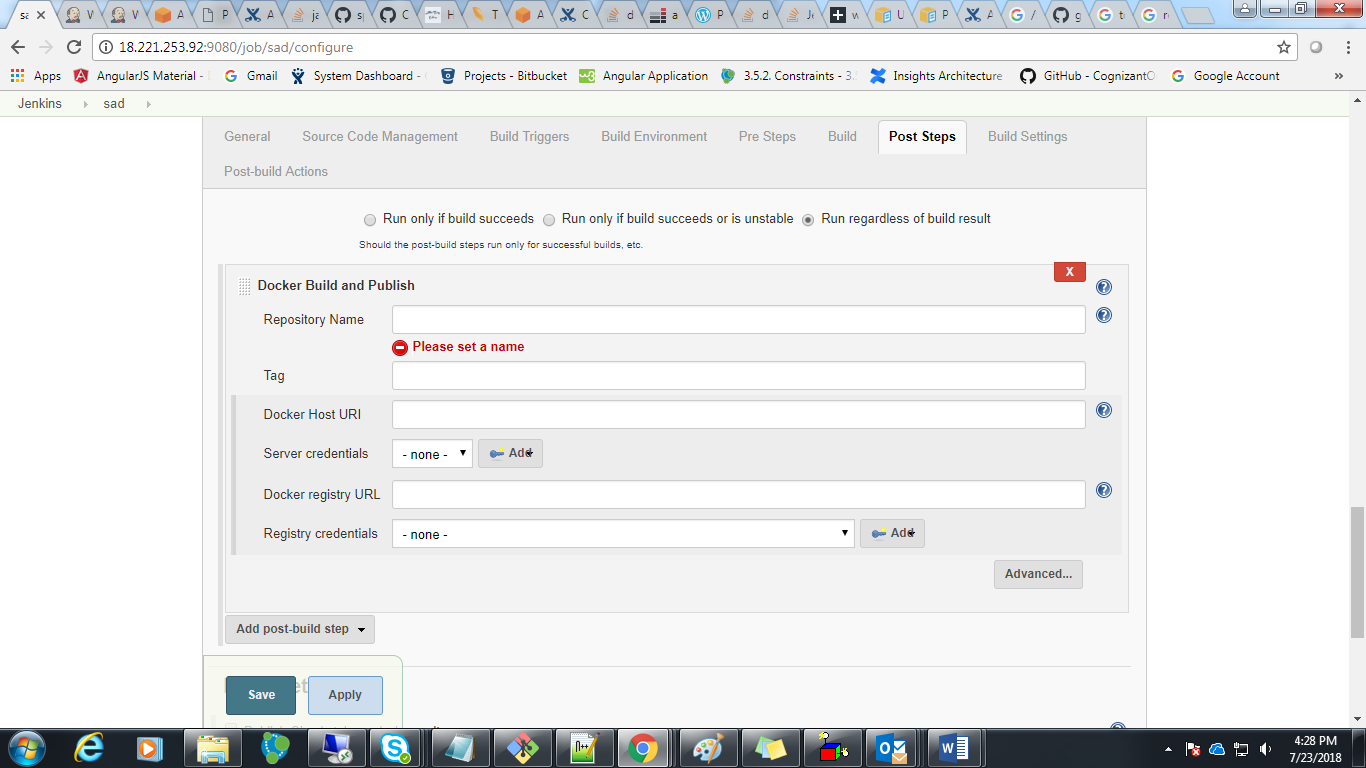


* Provide the kind and AWS Credentials .Fill in the Access Key ID,Secret Access Key of the AWS instance.



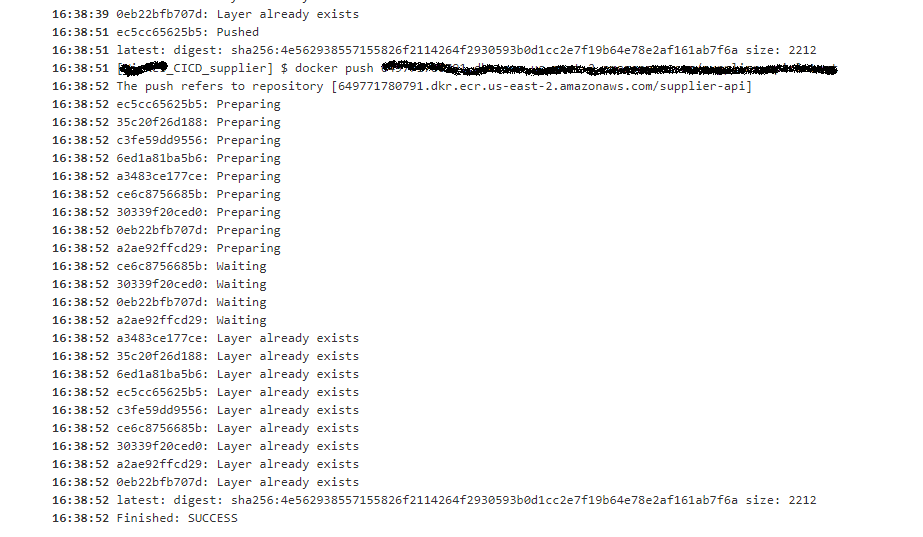
**Steps in Configuration:**

* In the “**Configure**” of the jobs Select “**Post build actions**”
* Select the task “**Docker Build and Publish** “
* Provide the repository Name to which the image needs to be pushed
* Tag the image
* Provide the Repository URL in the Docker registry URL
* Select the credentials from the one we added in Credentials.



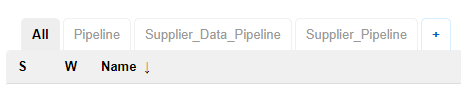
(Note in Advanced Tab check the Skip Decorate to avoid the build number appearing with the name of the repository url)

* **Once the Build is run:**

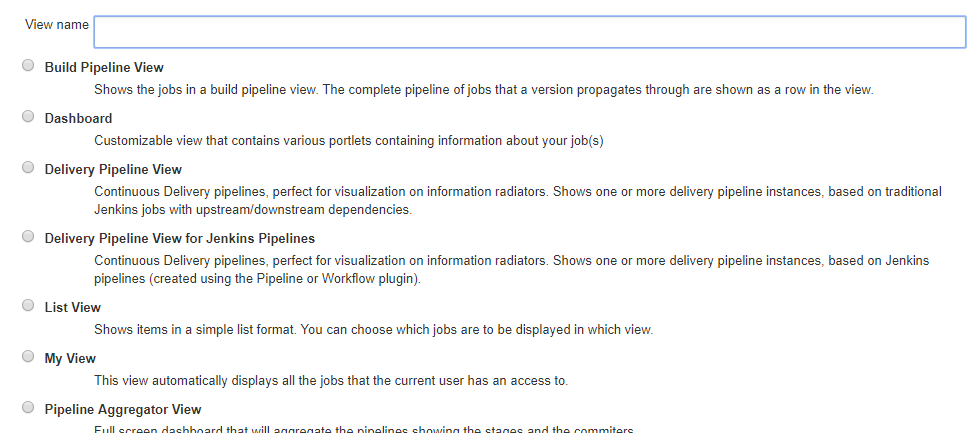


**STEP 7. To create a View Pipeline**

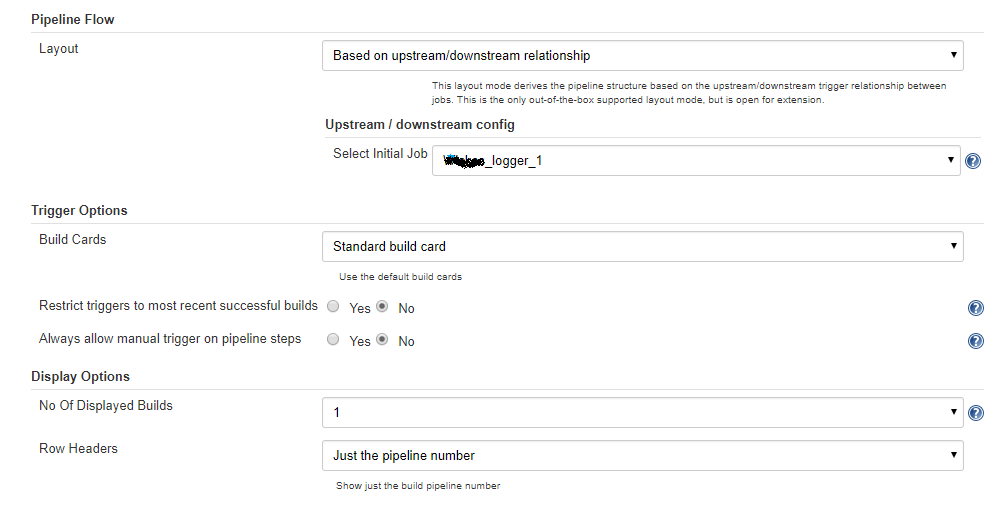
* To create a view, Select the **‘+’ near all tabs:**

****

* Provide the View name and select the option “**Build Pipeline View**”

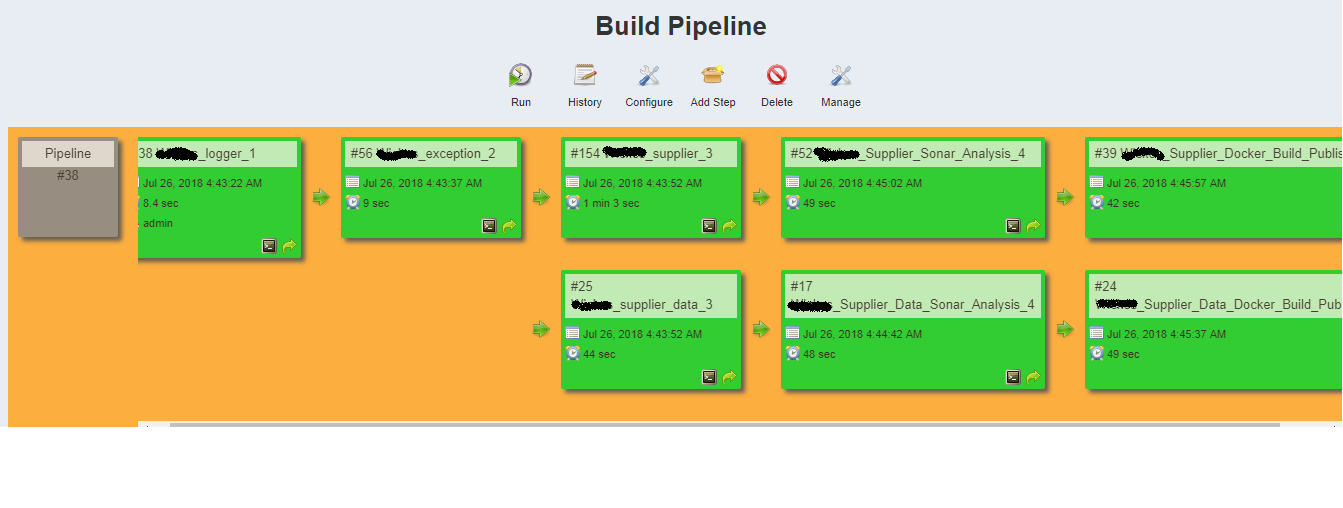
****

* Under “**Pipeline Flow”** ,Fill in the “**Upstram/downstream config”** as the initial job to be triggered in the chain job we created.
* Here **logger\_helper** is provided as it needs to be triggered for all other jobs to be triggered and built.

****

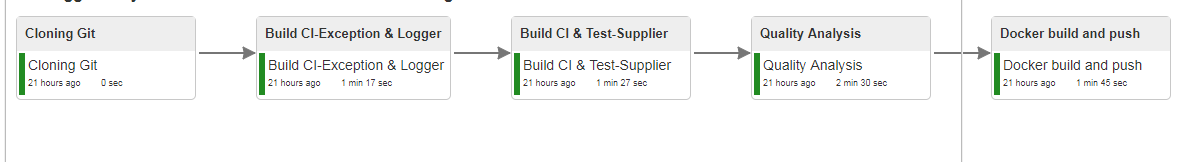
* Finally the Pipeline view is created with the visual view of the chain job built.

**Below is the consolidated view of the chain job created for supplier-api-service and supplier-data-service.**

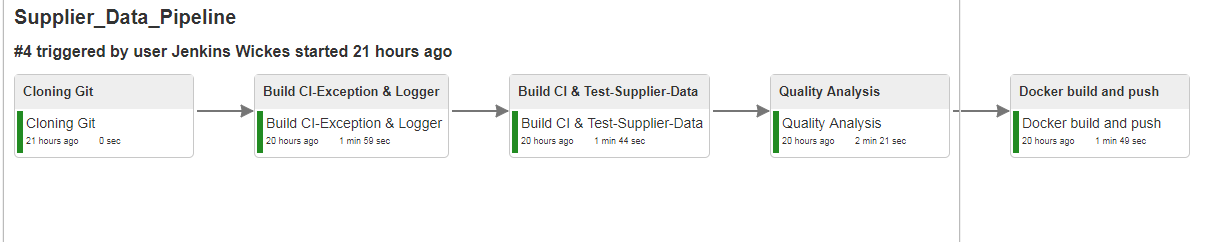
****Here ,to trigger the entire pipeline click on the run icon in the Build Pipeline.

* We have also specified the delivery pipeline for Supplier-api-service and supplier-data-service

1. **Supplier\_api\_service:**



1. **Supplier\_Data\_service**

****

**ERRORS ENCOUNTERED:**

**ERROR 1.**

****

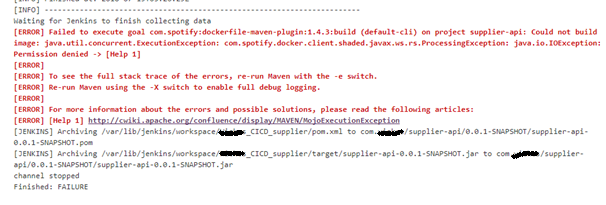
**SOLUTION**: This is because of the conflict that occurred in between the name specified in the branch inside git and the name provided in the Jenkins “**branch specifier**”.Provide the name of the branch as per the name of the branch in the git.

**ERROR 2.**

****

**SOLUTION:** This is because of the test case failure inside the code**.**Check the testcases inside code

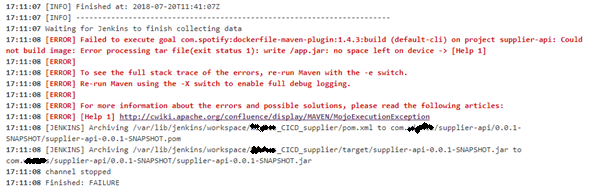
**ERROR 3.**



**SOLUTION**:This is because of the clash in the root user of Jenkins and docker.(Authorisation issue).Add Jenkins to docker user.

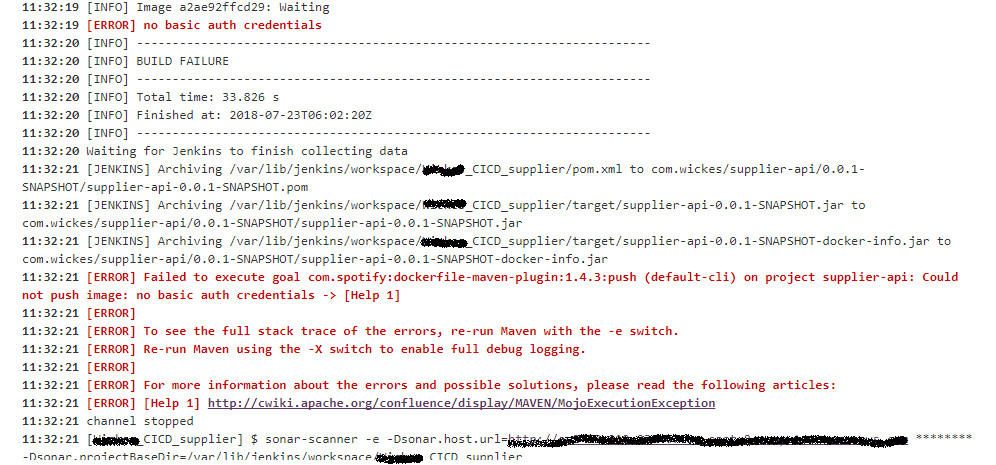
Use command **sudo usermod -aG docker user**

**ERROR 4**.



**SOLUTION**: Try to clean the Add extra memory space to the workspace( linux box )

**ERROR 5.**

****

**SOLUTION:**This is because of the unknown or missing credentials of AWS. Provide the Authentication token.ie add the credentials details as mentioned in Step 6.