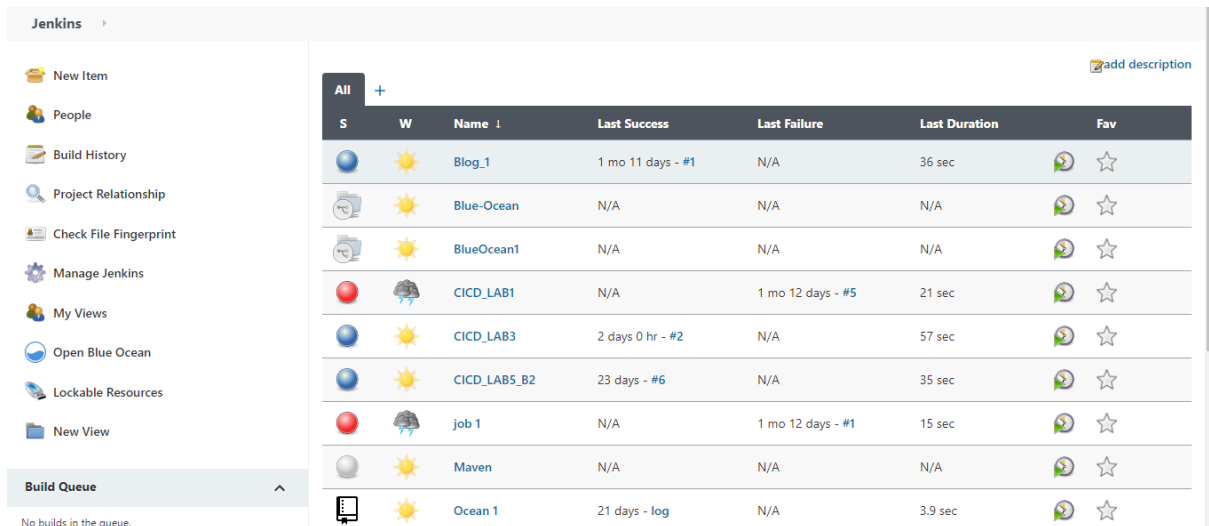


Experiment-10

Jenkins integration with SonarQube

Click on the new item option



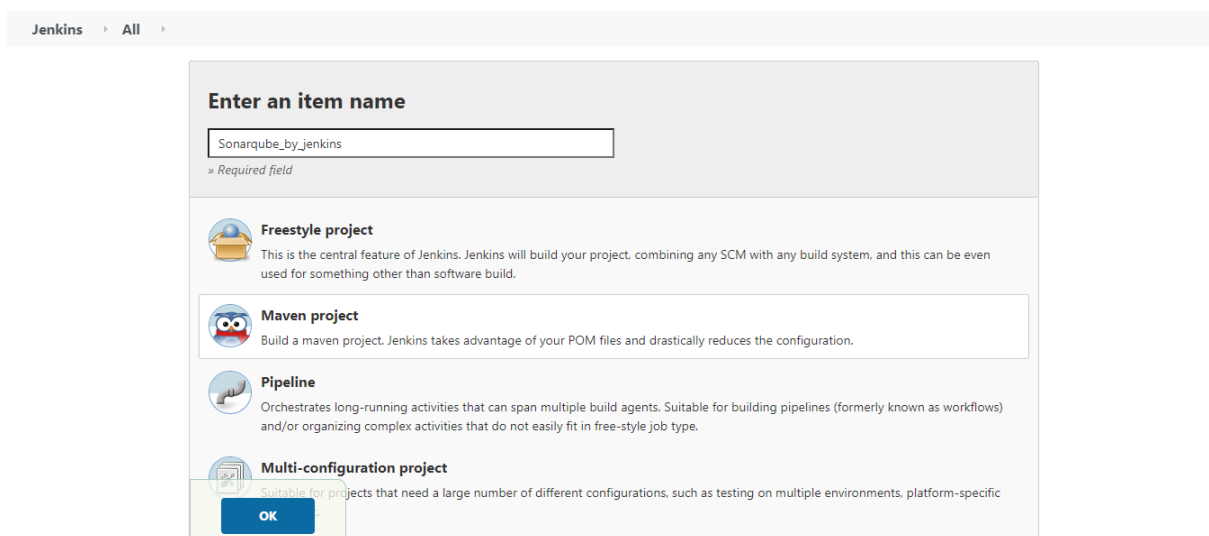
The screenshot shows the Jenkins 'New Item' dialog. On the left is a sidebar with navigation options: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Open Blue Ocean, Lockable Resources, and New View. The main area displays a table of existing items. At the top right of the table is a '+ add description' link. Below the table is a 'Build Queue' section indicating 'No builds in the queue.'

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav
		Blog_1	1 mo 11 days - #1	N/A	36 sec	
		Blue-Ocean	N/A	N/A	N/A	
		BlueOcean1	N/A	N/A	N/A	
		CICD_LAB1	N/A	1 mo 12 days - #5	21 sec	
		CICD_LAB3	2 days 0 hr - #2	N/A	57 sec	
		CICD_LAB5_B2	23 days - #6	N/A	35 sec	
		job 1	N/A	1 mo 12 days - #1	15 sec	
		Maven	N/A	N/A	N/A	
		Ocean 1	21 days - log	N/A	3.9 sec	

Enter the name for the project

Select project type as maven project

Click on OK



The screenshot shows the 'Enter an item name' dialog in Jenkins. It has a text input field containing 'Sonarqube_by_jenkins' and a 'Required field' message. Below the input field are four project type options, each with an icon and a description:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific

At the bottom left is an 'OK' button.

In the dashboard that appears, go the source code management and provide the location of the github repository that hold the Jenkinsfile for the maven command invocation.

Provide the complete path of the pom.xml file.

The screenshot shows the Jenkins configuration interface for a job named 'SonarQube_by_Jenkins'. The 'Build Environment' tab is selected, showing options for post-build actions, pre-build steps, and build settings. The 'Root POM' field is filled with 'Test-Maven-1/pom.xml'. The 'Run regardless of build result' option is selected under 'Post Steps'. A tooltip is visible at the bottom left, stating 'Should the post-build steps run only for successful builds, etc.'.

Jenkins > SonarQube_by_Jenkins >

General Source Code Management Build Triggers **Build Environment** Pre Steps Build Post Steps Build Settings

Post-build Actions

- ☐ Generate test reports
- ☐ Inspect build log for published Gradle build scans
- ☐ With Ant

Pre Steps

Add pre-build step

Build

Root POM: Test-Maven-1/pom.xml

Goals and options

Advanced...

Post Steps

☐ Run only if build succeeds ☐ Run only if build succeeds or is unstable ☒ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Save Apply

```

32     }
33
34
35     stage('Test phase Starts'){
36         steps{
37             echo 'Start the Test phase'
38         }
39     }
40     stage('Test'){
41         steps{
42             bat 'mvn test'
43         }
44     }
45
46
47
48     stage('Sonar phase Starts'){
49         steps{
50             echo 'Start the Sonar phase'
51         }
52     }
53     stage('Sonar'){
54         steps{
55             bat 'mvn sonar:sonar -Dsonar.host.url=http://localhost:9000 -Dsonar.analysis.mode=publish'
56         }
57     }
58 }
59 }

```

The project should lead you to a dashboard similar to this.

Click on the build now option

Click the option of console output to check the build process

The build starts to happen

- **BUILD SUCCESS**

