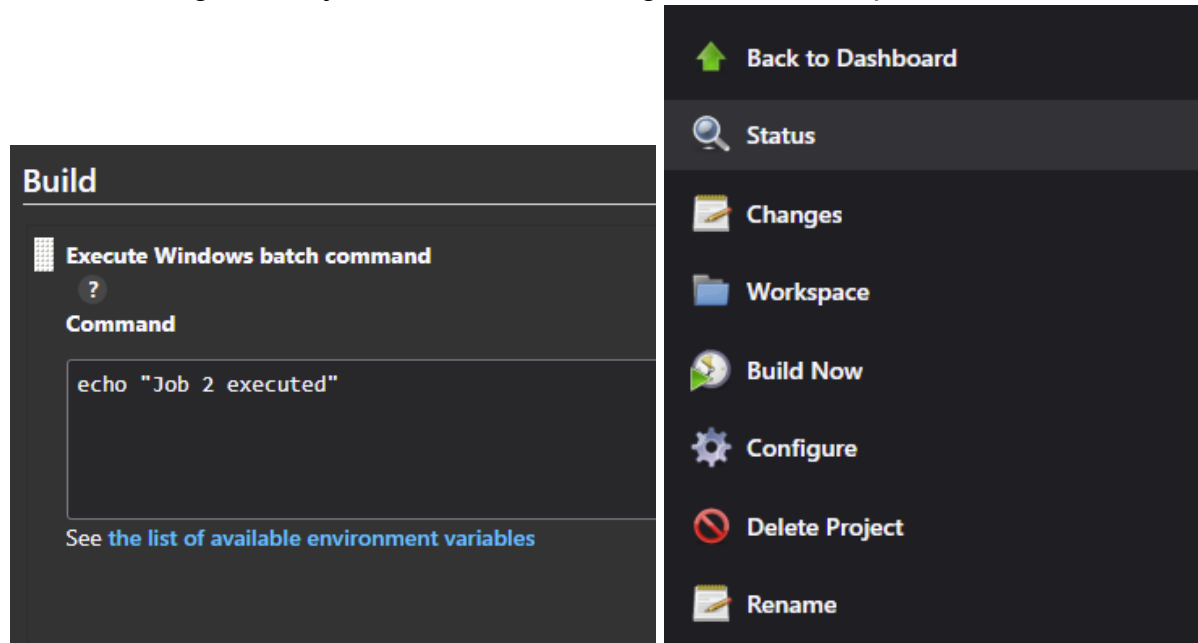


Jenkins-Learning

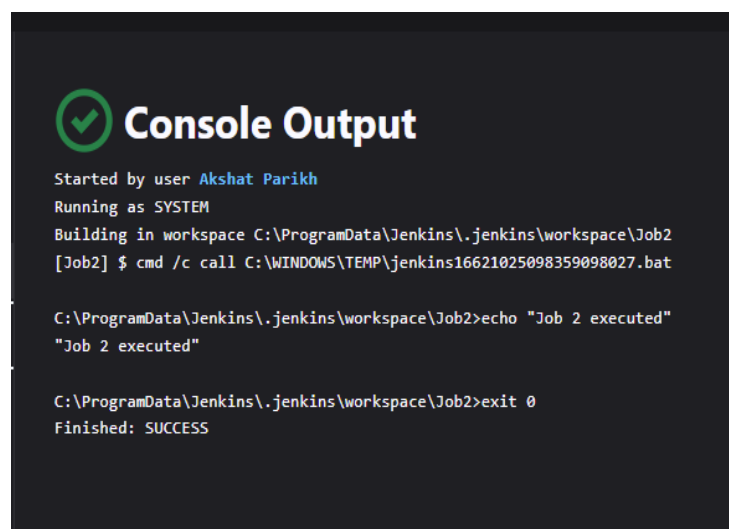
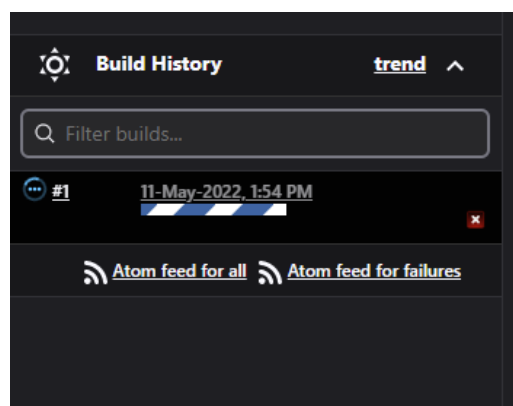
In this repo, I have demonstrated my understanding of Jenkins.

1). Creating a freestyle job:

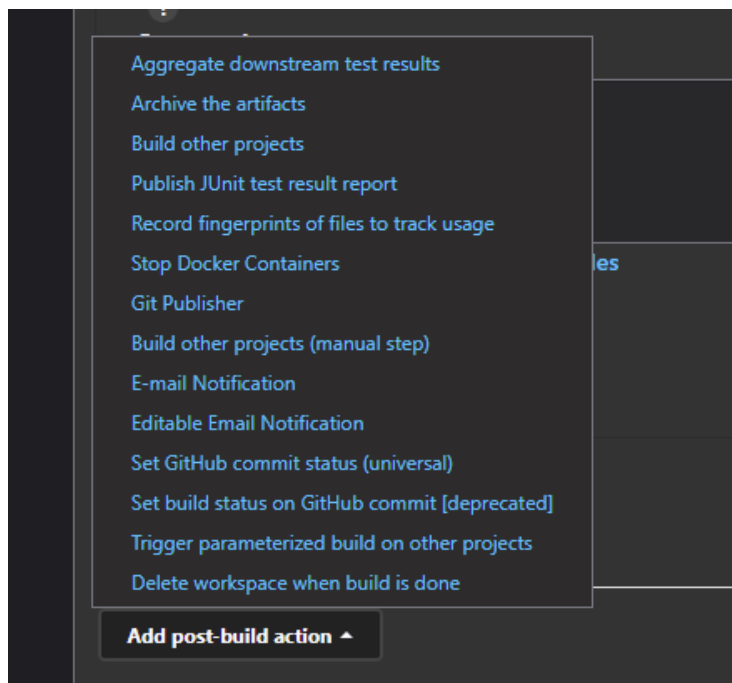
- First, I click on New Item -> Freestyle Project, give the project a name, and click OK.
- Then, go to the job and click on “Configure” on the left panel



- In the “General” tab, go to the “Build” section and type in a command shown in the image above and save:
- In the end, click on “Build now” to execute the job.

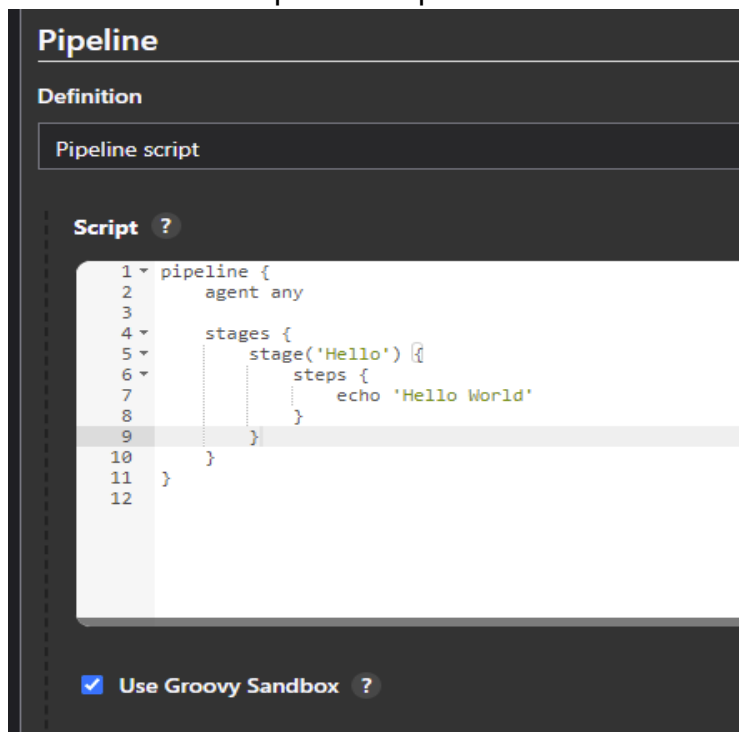


- To check the results, click on “Build History->Console Output” and you will see the output
- We can add post-build actions which are executed once the build is complete and there are many options as per the need.



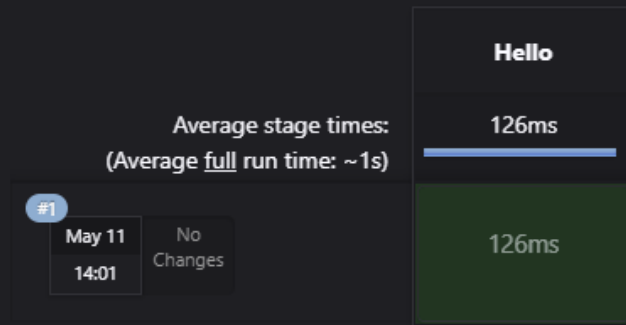
2). Build a pipeline to execute a job:

- From the dashboard, choose “New item -> Pipeline”.
- Go to “pipeline -> Configure” and go to the “Pipeline” tab.
- Choose a Pipeline script and write down a sample pipeline and build the project.



- We can now see these 2 views that suggest that the pipeline has run successfully

Stage View



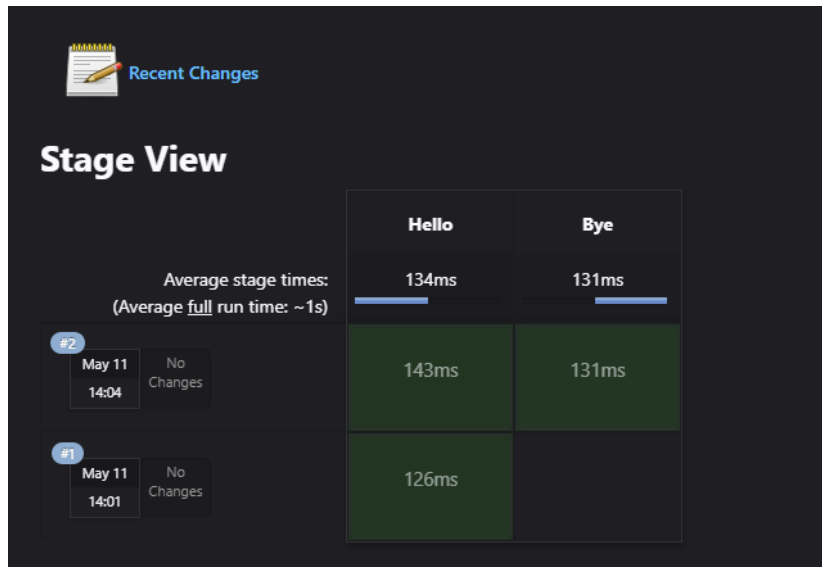
✓ Console Output

```
Started by user Akshat Parikh
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\FirstPipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] echo
Hello World
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

We can add multiple stages to the pipeline as per the requirement.


Script ?

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Hello') {
6       steps {
7         echo 'Hello World'
8       }
9     }
10    stage('Bye') {
11      steps {
12        echo 'Bye Bye World'
13      }
14    }
15  }
16 }
17
```



3). Deploy a sample nodejs application using the scripted pipeline

As per the image below, I have set up a sample nodejs application on my Github account.

	akshatparikh Update index.js	e78ad21 23 hours ago	🕒 16 commits
📁	.vscode	tweaks	2 years ago
📄	.gitignore	ignore	4 years ago
📄	.prettierrc	pretty	2 years ago
📄	README.md	run	4 years ago
📄	index.js	Update index.js	23 hours ago
📄	package-lock.json	tweaks	2 years ago
📄	package.json	tweaks	2 years ago

Now, I have created a new pipeline and configure the pipeline script like this:

```
Script ?
1 pipeline {
2   agent any
3
4   stages{
5     stage ('Install Dependencies'){
6       steps{
7         git 'https://github.com/akshatparikh/node-hello-world'
8         bat 'npm install'
9       }
10    }
11
12    stage('Build') {
13      steps {
14        bat 'npm start'
15      }
16    }
17  }
18 }
```

Now, I will build the pipeline and the project will startup on localhost:3000.

```
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\DemoPipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Install Dependencies)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\DemoPipeline\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/akshatparikh/node-hello-world # timeout=10
Fetching upstream changes from https://github.com/akshatparikh/node-hello-world
> git.exe --version # timeout=10
> git --version # 'git version 2.35.1.windows.2'
> git.exe fetch --tags --force --progress -- https://github.com/akshatparikh/node-hello-world +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision e78ad2157096215ca68dc5002f31cf73e2283c6a (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f e78ad2157096215ca68dc5002f31cf73e2283c6a # timeout=10
> git.exe branch -a -v --no-abbrev # timeout=10
> git.exe branch -D master # timeout=10
> git.exe checkout -b master e78ad2157096215ca68dc5002f31cf73e2283c6a # timeout=10
Commit message: "Update index.js"
> git.exe rev-list --no-walk e78ad2157096215ca68dc5002f31cf73e2283c6a # timeout=10
[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\DemoPipeline>npm install
npm WARN ancient lockfile
npm WARN ancient lockfile The package-lock.json file was created with an old version of npm,
npm WARN ancient lockfile so supplemental metadata must be fetched from the registry.
npm WARN ancient lockfile
npm WARN ancient lockfile This is a one-time fix-up, please be patient...
npm WARN ancient lockfile

removed 298 packages, and audited 1 package in 3s

found 0 vulnerabilities
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\DemoPipeline>npm start
> node-hello@1.0.0 start
> node index.js

Server running on http://localhost:3000/
```

A browser address bar with navigation icons (back, forward, refresh, home) and an information icon on the left, followed by the text "localhost:3000".

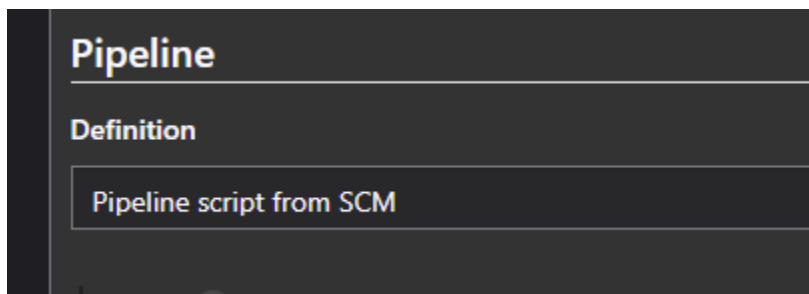
Hello Node!

This is the sample nodejs app listening on port 3000

4). Build a docker image using a pipeline from SCM

akshatparikh updated jenkinsfile			✓ cb571df 5 hours ago	🕒 166 commits
📁 azure-vote	reverting			2 years ago
📁 jenkins-tutorial	Fix incorrect comments in Jenkins deployment script			3 years ago
📁 scripts	Adding test_container.sh			2 years ago
📁 tests	adding tests			2 years ago
📄 .gitignore	Initial commit			5 years ago
📄 Jenkinsfile	updated jenkinsfile			5 hours ago
📄 LICENSE	Initial commit			5 years ago
📄 README.md	Update README.md			2 years ago
📄 azure-vote-all-in-one-redis.yaml	addign deploy stages			2 years ago
📄 docker-compose.yml	addign deploy stages			2 years ago

- I have set up a repository on GitHub for this purpose as shown below:



- Now, I will create a new pipeline and this time, I will choose the option “Pipeline Script from SCM”

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/akshatparikh/azure-voting-app-redis

Credentials ?

- none - Add

Advanced...

Here I have selected my GitHub repository to get “Jenkinsfile” from.

15 lines (13 sloc) | 259 Bytes

```
1 pipeline {
2   agent any
3
4   stages {
5
6     stage('Docker Build') {
7       steps {
8         dir('azure-vote/') {
9           bat 'docker images -a'
10          bat 'docker build -t jenkins-pipeline .'
11        }
12      }
13    }
14  }
15 }
```

- This is how the Jenkinsfile is scripted inside the repo

- This is how the Jenkinsfile is scripted inside the repo

- Upon building the pipeline, you can see the output below with the “SUCCESS” message.

```
C:\ProgramData\Jenkins\.jenkins\workspace\ScriptedPipeline\azure-vote>docker build -t jenkins-pipeline .
#1 [internal] load build definition from Dockerfile
#1 sha256:f0187e479e125d2d2597f8bbbd54ba78f26b3cf89117cdddb65974464b55ffe
#1 transferring dockerfile: 125B 0.0s done
#1 DONE 0.1s

#2 [internal] load .dockerignore
#2 sha256:6e203306cba32acc35962e00b6ba694134e230b1a0ea045e1dbe703d19df31e5
#2 transferring context: 2B done
#2 DONE 0.1s

#3 [internal] load metadata for docker.io/tiangolo/uwsgi-nginx-flask:python3.6
#3 sha256:0c2fa23bcb11452f95b5b29eb37f2d2bf90f1bc8956bd39d202248798f27ffd9
#3 DONE 0.9s

#6 [1/3] FROM docker.io/tiangolo/uwsgi-nginx-flask:python3.6@sha256:695c26b418dad47121ed71b1216e2d3b96733832d9b010ef87a10300a32fc558
#6 sha256:00a1cd8ef37ec90d64cead664a4b8b733ec131b486f3c90d040b1ec28bd09bf1
#6 DONE 0.0s

#7 [internal] load build context
#7 sha256:ce741eac3ae5482891adf57567fe894056b275a28e2f76d07725b1edbed9dfd6
#7 transferring context: 5.89kB done
#7 DONE 0.0s

#4 [2/3] RUN pip install redis
#4 sha256:0d1fd0fd6308456bba40aa303e015c44411c934d8deea47539d0ccefa79cc713
#4 CACHED

#5 [3/3] ADD /azure-vote /app
#5 sha256:4a7eaba2a80ed33de4ca55e537cfff62f8159dc2a1e341c0503445ec314b05beb
#5 CACHED

#8 exporting to image
#8 sha256:e8c613e07b0b7ff33893b694f7759a10d42e180f2b4dc349fb57dc6b71dcab00
#8 exporting layers done
#8 writing image sha256:12431997af0fbcfe76437247a94a5da9aff58e0cb0c2d9513979d51a90d75c02 done
#8 naming to docker.io/library/jenkins-pipeline done
#8 DONE 0.2s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Now, I have added a post-build step to the Jenkinsfile.

```
post {
    success{
        echo ('The run is a success')
    }
    failure{
        echo ('The run is a failure')
    }
}
```

```
[Pipeline] }
[Pipeline] // dir
Post stage
[Pipeline] echo
The run is a success
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

5). Creating a multi-branch pipeline

- From Dashboard, create a new item and choose 'Multibranch Pipeline'.
- To get the repository from GitHub, I have set up the following and saved the pipeline

Branch Sources

GitHub
Credentials ?

akshatparikh/***** Add

User akshatparikh


Repository HTTPS URL

Repository HTTPS URL ?

https://github.com/akshatparikh/azure-voting-app-redis

Credentials ok. Connected to https://github.com/akshatparikh/azure-voting-app-redis.

Validate

 **Scan Repository Log**

Started

[Wed May 11 13:23:34 GMT-04:00 2022] Starting branch indexing...

13:23:34 Connecting to https://api.github.com using akshatparikh/*****

Examining akshatparikh/azure-voting-app-redis

Checking branches...

Getting remote branches...

Checking branch master

Getting remote pull requests...

'Jenkinsfile' found

Met criteria

Scheduled build for branch: master

Checking branch add-tests

'Jenkinsfile' found

Met criteria

Scheduled build for branch: add-tests

Checking branch declarative-pipeline

'Jenkinsfile' found

Met criteria

Scheduled build for branch: declarative-pipeline

- Now, it will run a scan of the repository to find Jenkinsfile and return the list of branches

Branches (8)		Pull Requests (0)				
S	W	Name		Last Success	Last Failure	Last Duration
❌	🔧	add-tests		N/A	19 hr #1	18 sec
✅	🔧	declarative-pipeline		19 hr #1	N/A	27 sec
❌	🔧	docker-step		N/A	19 hr #1	1 min 46 sec
✅	🔧	hello-args		19 hr #1	N/A	39 sec
✅	🔧	hello-library		19 hr #1	N/A	46 sec
✅	🔧	master		19 hr #1	N/A	45 sec
✅	🔧	pipeline-library		19 hr #1	N/A	53 sec
✅	🔧	scripted-pipeline		19 hr #1	N/A	24 sec

- This is the list of branches in the repository.

6). Pushing an image to the container registry

In this module, I have pushed an image to DockerHub.

First, I added the credential of my DockerHub account by traversing to 'Manage Jenkins -> Manage Credentials -> Global'

Credentials						
T	P	Store	Domain	ID	Name	
🔑	🏠	Jenkins	(global)	Github_Connection	Github_Connection	
🔑	🏠	Jenkins	(global)	Secret text	Secret text	
🔑	🏠	Jenkins	(global)	DockerHub	akshatparikh/*****	

Then, I created the script as follows to push the image to my DockerHub account.

```
stage('Push a container'){
    steps {
        dir('azure-vote/') {
            script{
                docker.withRegistry('https://index.docker.io/v1/', 'DockerHub'){
                    def image = docker.build('akshatparikh/jenkins_demo:latest')
                    image.push()
                }
            }
        }
    }
}
```

```
C:\ProgramData\Jenkins\.jenkins\workspace\ScriptedPipeline\azure-vote>docker tag "akshatparikh/jenkins_demo:latest" "index.docker.io/akshatparikh/jenkins_demo:latest"
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] isUnix
[Pipeline] withEnv
[Pipeline] {
[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\ScriptedPipeline\azure-vote>docker push "index.docker.io/akshatparikh/jenkins_demo:latest"
The push refers to repository [docker.io/akshatparikh/jenkins_demo]
5b6562fbba3b: Preparing
956c098c2724: Preparing
587ea72f0bca: Preparing
254f56a032fd: Preparing
5dd448b3c6e1: Preparing
83508bdcc446: Preparing
```