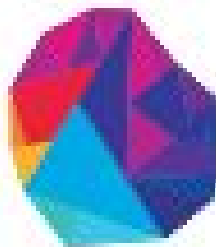


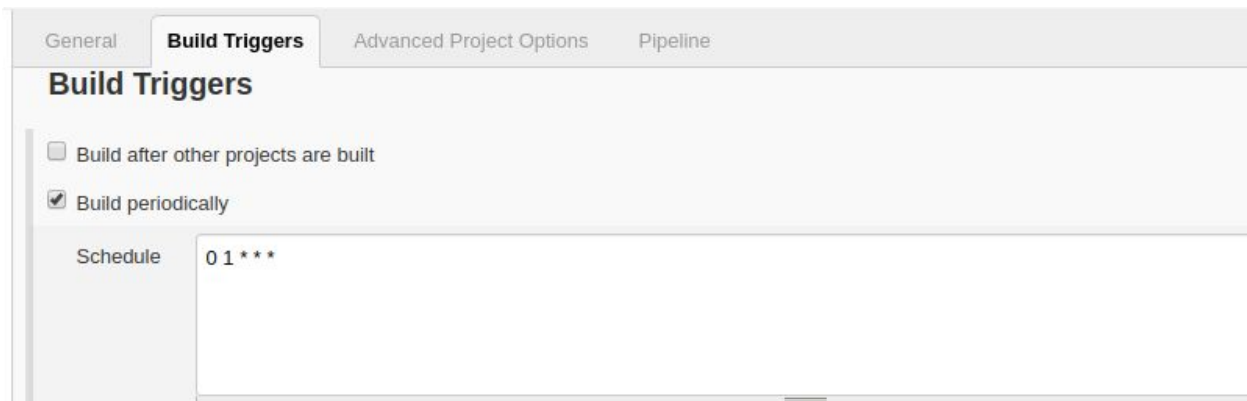
ASSESSMENT ON: JENKINS2

**TO
THE
NEW**



1) Create a Jenkins pipeline Job to delete redundant docker images daily at 1 AM UTC.

STEP 1:



The image shows the Jenkins configuration page for a job, specifically the 'Build Triggers' tab. The 'Build after other projects are built' checkbox is unchecked. The 'Build periodically' checkbox is checked. The 'Schedule' field contains the text '0 1 ***'.

General | **Build Triggers** | Advanced Project Options | Pipeline

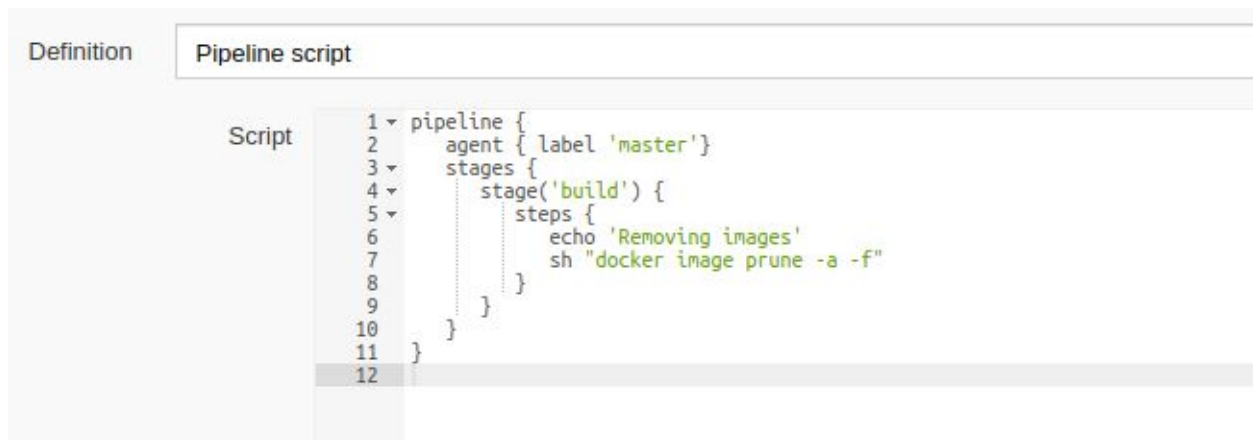
Build Triggers

☐ Build after other projects are built

☒ Build periodically

Schedule: 0 1 ***

STEP 2:



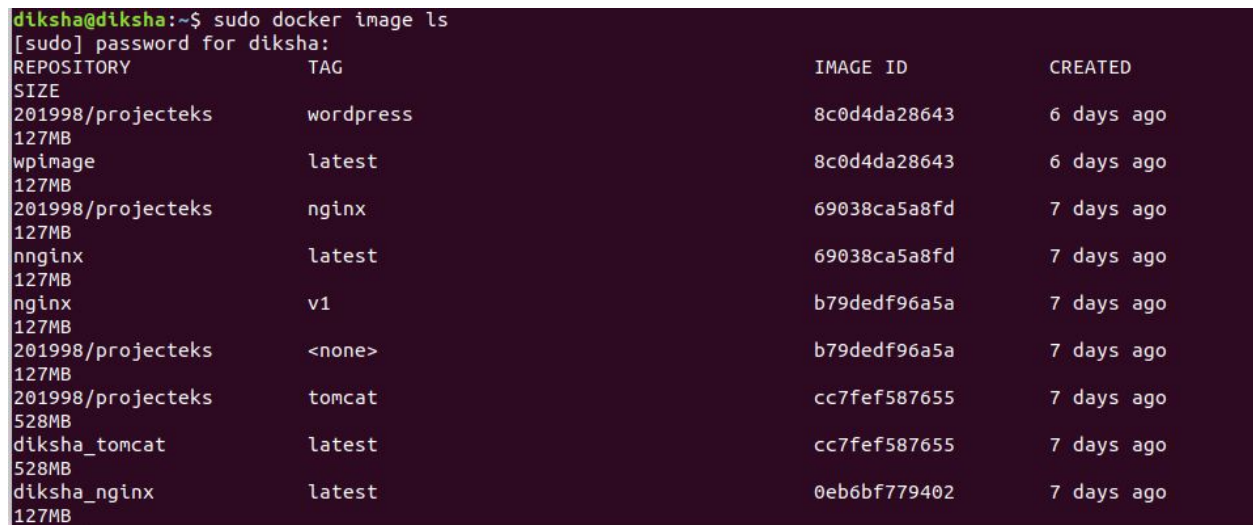
The image shows the Jenkins configuration page for a job, specifically the 'Pipeline script' tab. The 'Script' field contains a Jenkins pipeline script.

Definition | **Pipeline script**

Script

```
1 pipeline {
2   agent { label 'master' }
3   stages {
4     stage('build') {
5       steps {
6         echo 'Removing images'
7         sh "docker image prune -a -f"
8       }
9     }
10  }
11 }
12
```

STEP 3:



The image shows a terminal window with the command 'docker image ls' executed. The output is a table with columns: REPOSITORY, TAG, IMAGE ID, and CREATED.

```
diksha@diksha:~$ sudo docker image ls
[sudo] password for diksha:
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
201998/projecteks   wordpress           8c0d4da28643        6 days ago
127MB
wpimage             latest             8c0d4da28643        6 days ago
127MB
201998/projecteks   nginx              69038ca5a8fd        7 days ago
127MB
nnginx             latest             69038ca5a8fd        7 days ago
127MB
nginx              v1                 b79dedf96a5a        7 days ago
127MB
201998/projecteks   <none>             b79dedf96a5a        7 days ago
127MB
201998/projecteks   tomcat             cc7fef587655        7 days ago
528MB
diksha_tomcat       latest             cc7fef587655        7 days ago
528MB
diksha_nginx        latest             0eb6bf779402        7 days ago
127MB
```

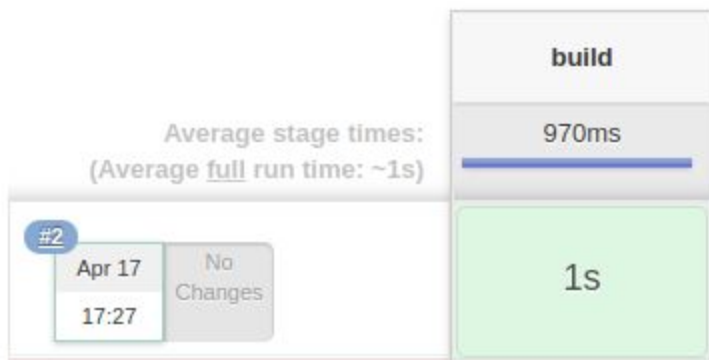
STEP 4:

Pipeline pipelineQ1



[Recent Changes](#)

Stage View



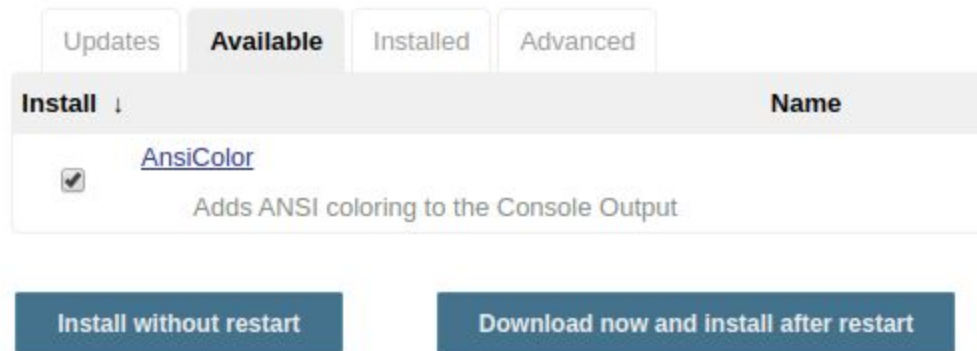
STEP 5:

```
diksha@diksha:~$ sudo docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
test          tomcat    d37b07e9ad0f   7 days ago    147MB
diksha@diksha:~$
```

2) Create a shared library function to convert error and success output into a colourful output and use it in the upcoming questions(Hint: use ANSI color).

** shared libraries – are loaded when a program is launched and loaded into memory and binding occurs at run time.*

STEP 1: Install AnsiColor plugin Manage Jenkins> Manage Plugins



STEP 2: Setting up a shared library { The use of Shared Libraries provides the ability to share and reuse code across jobs for CI and CD processes.}

- Navigate to Dashboard > select Manage Jenkins > select Configure System > scroll down to Global Pipeline Libraries > select Add
- Enter “my-shared_lib” in the Name field
- Enter master in Default Version
 - This tells jenkins which branch of our Shared Library we plan to use by default.

Global Pipeline Libraries

Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without “sandbox” restrictions. May use @Grab.

Library Name	<input type="text" value="my_shared_lib"/>
Default version	<input type="text" value="master"/> <small>Cannot validate default version until after saving and reconfiguring.</small>
Load implicitly	<input type="checkbox"/>
Allow default version to be overridden	<input checked="" type="checkbox"/>
Include @Library changes in job recent changes	<input checked="" type="checkbox"/>

STEP 3: Create a git repository named “jenkins2” with a folder named “vars”. Then In the vars folder we shall write the groovy scripts that’ll be pulled into the Jenkinsfile at the runtime.

Branch: **master** ▾

Jenkins2 / Jenkinsfile

 **DikshaTomar101** Create Jenkinsfile

1 contributor

23 lines (23 sloc) | 599 Bytes

```
1  #!groovy
2  library identifier: 'Jenkins2@master', retriever: modernSCM(
3      [$class: 'GitSCMSource' ,
4          remote: 'https://github.com/DikshaTomar101/Jenkins2'])
5  pipeline {
6      agent any
7      options{
8          timestamps()
9          ansiColor( 'xterm' )
10     }
11     stages {
12         stage( 'Coloured Outputs with git commit id' ) {
13             steps {
14                 script{
15                     logs.info "SUCCESS"
16                     logs.warn "WARNING"
17                     def gitId=sh(script: 'git rev-parse HEAD' , returnStdout: true)
18                     logs.gitCommitId(gitId)
19                 }
20             }
21         }
22     }
23 }
```

Branch: master Jenkins2 / vars / logs.groovy

 DikshaTomar101 Create logs.groovy

1 contributor

21 lines (17 sloc) 409 Bytes


```
1 def loadColors() {
2     RED='\033[0;31m'
3     GREEN='\033[0;32m'
4     NC='\033[0m'
5 }
6
7 def info(message){
8     loadColors()
9     sh ""set +x;echo -e "${GREEN}[INFO] - ${message} ${NC}" ""
10 }
11
12 def warn(message){
13     loadColors()
14     sh ""set +x;echo -e "${RED}[WARN] - ${message} ${NC}" ""
15 }
16
17 def gitCommitId(message){
18     loadColors()
19     sh ""set +x;echo -e "${GREEN}[GIT COMMIT ID] - ${message} ${NC}" ""
20 }
21
```

STEP 4: Create a new job


Enter an item name

Q2


» Required field

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, or something other than software build.

**Pipeline**


Orchestrates long-running activities that can span multiple build agents, 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

**Folder**

Creates a container that stores nested items in it. Useful for grouping namespaces, so you can have multiple things of the same name as long as they are in the same folder

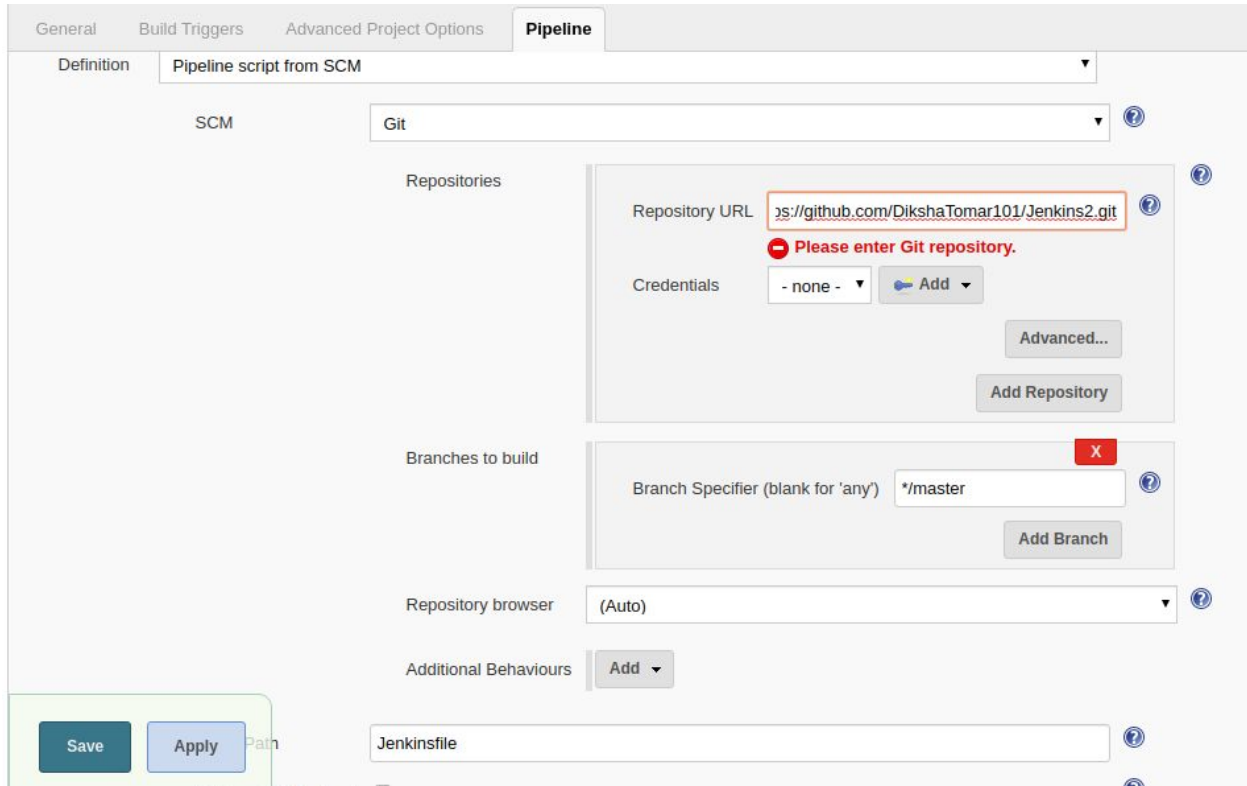
**GitHub Organization**

Scans a GitHub organization (or user account) for all repositories managed by it

OK

Cancel

STEP 5: Provide the git repository Url



The screenshot shows the Jenkins Pipeline configuration page. The 'Definition' tab is selected, and 'Pipeline script from SCM' is chosen. The 'SCM' dropdown is set to 'Git'. Under 'Repositories', the 'Repository URL' is entered as 'https://github.com/DikshaTomar101/Jenkins2.git', which is highlighted with a red box and a red error message: 'Please enter Git repository.' The 'Credentials' dropdown is set to '- none -'. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' set to '*/master'. The 'Repository browser' is set to '(Auto)'. The 'Additional Behaviours' section has an 'Add' button. At the bottom, there are 'Save' and 'Apply' buttons, and a 'Jenkinsfile' field.

General Build Triggers Advanced Project Options **Pipeline**

Definition Pipeline script from SCM

SCM Git

Repositories

Repository URL `https://github.com/DikshaTomar101/Jenkins2.git`

⊖ Please enter Git repository.

Credentials - none - Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any') */master

Add Branch

Repository browser (Auto)

Additional Behaviours Add

Save Apply Path Jenkinsfile

STEP 6: Build the job

Console Output

Started by user [Diksha Tomar](#)

Obtained Jenkinsfile from git <https://github.com/DikshaTomar101/Jenkins2.git>

```
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] timestamps
[Pipeline] {
[Pipeline] ansiColor
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Coloured Outputs with git commit id)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
18:24:20 + set +x
18:24:20 -e [INFO] - SUCCESS
[Pipeline] sh
18:24:20 + set +x
18:24:20 -e [WARN] - WARNING
[Pipeline] sh
18:24:21 + git rev-parse HEAD
[Pipeline] sh
18:24:21 + set +x
18:24:21 -e [GIT COMMIT ID] - fad5f7f2c33e1c12d56d6db762a90025654f67f8
18:24:21
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // ansiColor
[Pipeline] }
[Pipeline] // timestamps
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```


3) Create a function in the same shared library to output git commitID.

Branch: **master**  **Jenkins2 / Jenkinsfile**

 **DikshaTomar101** Create Jenkinsfile

1 contributor

23 lines (23 sloc) | 599 Bytes

```
1  #!groovy
2  library identifier: 'Jenkins2@master', retriever: modernSCM(
3      [$class: 'GitSCMSource' ,
4          remote: 'https://github.com/DikshaTomar101/Jenkins2'])
5  pipeline {
6      agent any
7      options{
8          timestamps()
9          ansiColor( 'xterm' )
10     }
11     stages {
12         stage( 'Coloured Outputs with git commit id' ) {
13             steps {
14                 script{
15                     logs.info "SUCCESS"
16                     logs.warn "WARNING"
17                     def gitId=sh(script: 'git rev-parse HEAD' , returnStdout: true)
18                     logs.gitCommitId(gitId)
19                 }
20             }
21         }
22     }
23 }
```

```

[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] timestamps
[Pipeline] {
[Pipeline] ansiColor
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Coloured Outputs with git commit id)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
18:24:20 + set +x
18:24:20 -e [INFO] - SUCCESS
[Pipeline] sh
18:24:20 + set +x
18:24:20 -e [WARN] - WARNING
[Pipeline] sh
18:24:21 + git rev-parse HEAD
[Pipeline] sh
18:24:21 + set +x
18:24:21 -e [GIT COMMIT ID] - fad5f7f2c33e1c12d56d6db762a90025654f67f8
18:24:21
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // ansiColor
[Pipeline] }
[Pipeline] // timestamps
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

4) Take a sample react application and deploy it on EKS

a. You can use this repo or any other sample

(<https://github.com/gothinkster/react-redux-realworld-example-app>).

b. Create a Dockerfile for react application

c. Build and publish image to ECR (create ECR repo of your name) and image must have the git commit id in its name.

d. Deploy this image on EKS.

e. Send Slack notification/Mail/google chat notification for build pass, abort and fail.