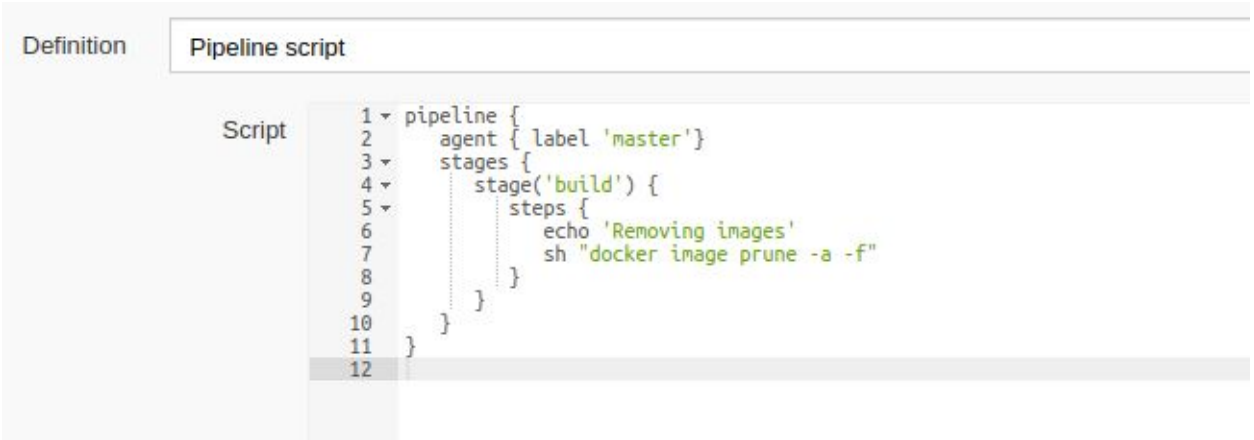
ASSESSMENT ON: JENKINS2

TO THE NEW

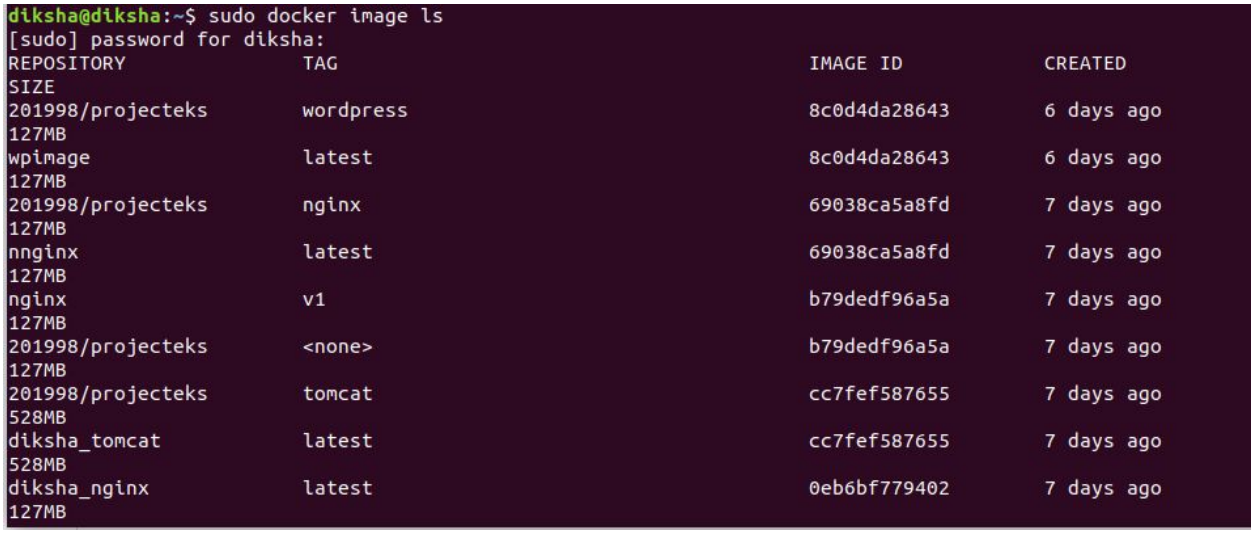
1) Create a jenkins pipeline Job to delete redundant docker images daily at 1 AM UTC.

STEP 1:

STEP 2:

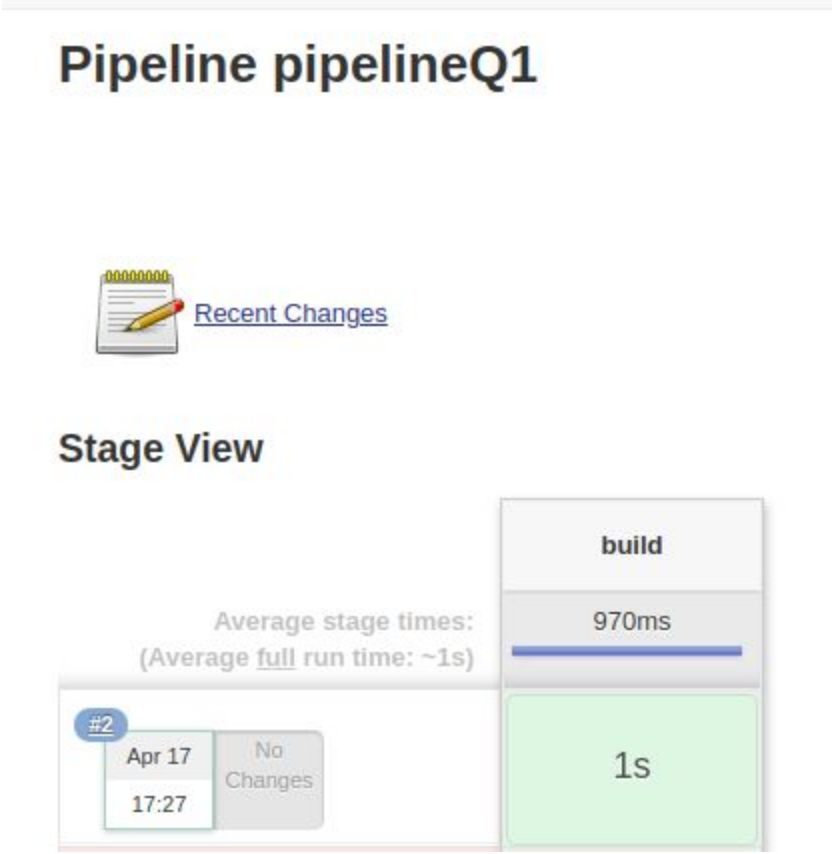


STEP 3:



```
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:



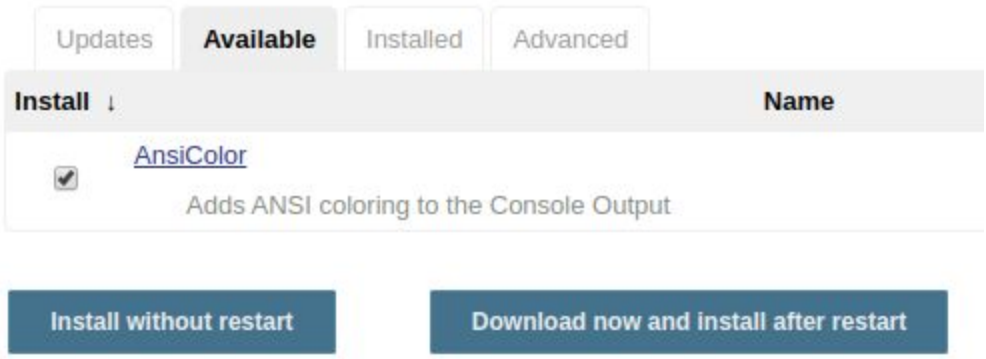
STEP 5:

```
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 colourfull output and use it in the upcoming questions(Hint: use ANSI color).

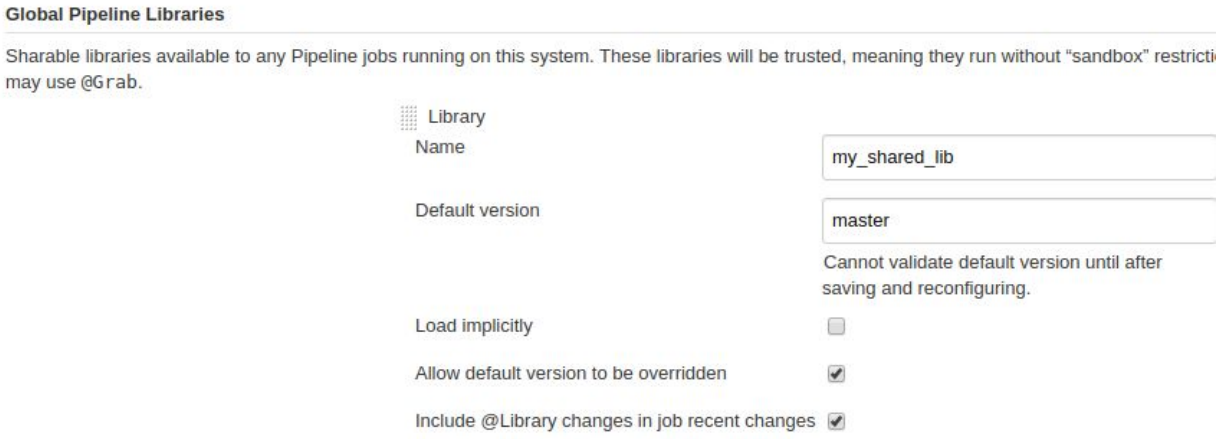
** shared libraries – are loaded when a program is launched and loaded into memory and binding occurs at run time.*

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.



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

```
#!groovy
library identifier: 'Jenkins2@master', retriever: modernSCM(
        [$class: 'GitSCMSource' ,
         remote: 'https://github.com/DikshaTomar101/Jenkins2'])
pipeline {
   agent any
    options{
       timestamps()
       ansiColor( 'xterm' )
   }
   stages {
      stage( 'Coloured Outputs with git commit id' ) {
         steps {
             script{
                 logs.info "SUCCESS"
                 logs.warn "WARNING"
                 def gitId=sh(script: 'git rev-parse HEAD' , returnStdout: true)
                 logs.gitCommitId(gitId)
             }
         }
      }
  }
}
```

Branch: **master** Jenkins2 / vars / **logs.groovy**

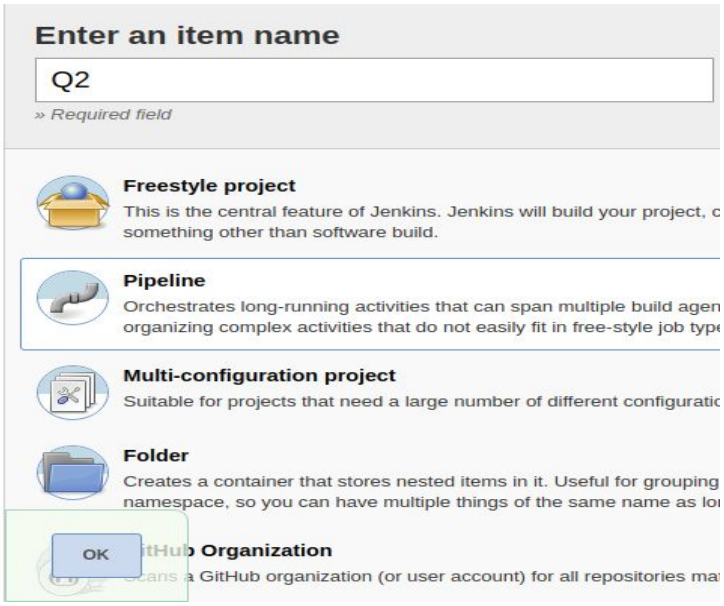
DikshaTomar101 Create logs.groovy

1 contributor

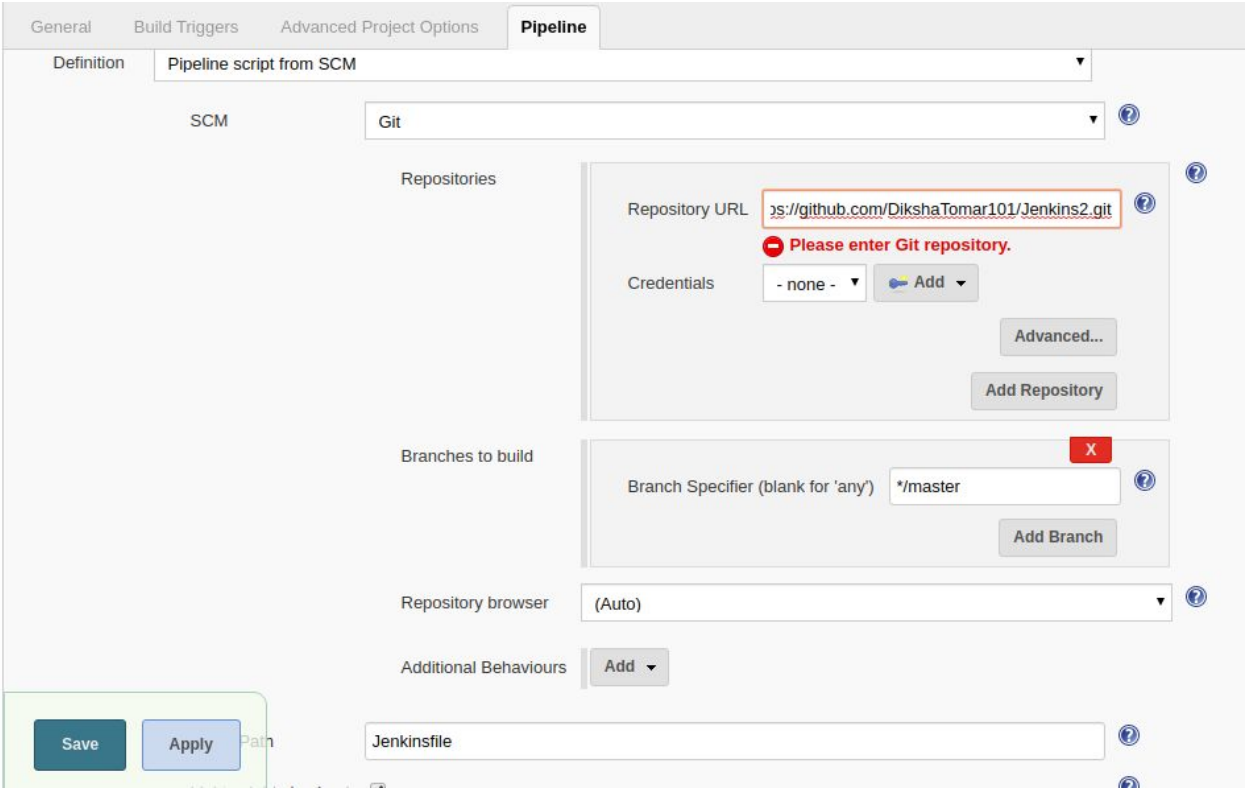
21 lines (17 sloc) | 409 Bytes

```
def loadColors() {
    RED='\033[0;31m'
    GREEN='\033[0;32m'
    NC='\033[0m'
}

def info(message){
    loadColors()
    sh """set +x;echo -e "${GREEN}[INFO] - ${message} ${NC}" """
}

def warn(message){
    loadColors()
    sh """set +x;echo -e "${RED}[WARN] - ${message} ${NC}" """
}

def gitCommitId(message){
    loadColors()
    sh """set +x;echo -e "${GREEN}[GIT COMMIT ID] - ${message} ${NC}" """
}

```

STEP 4: Create a new job



STEP 5: Provide the git repository Url



STEP 6: Build the job

```
[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

```
#!groovy
library identifier: 'Jenkins2@master', retriever: modernSCM(
        [$class: 'GitSCMSource' ,
         remote: 'https://github.com/DikshaTomar101/Jenkins2'])
pipeline {
   agent any
    options{
       timestamps()
       ansiColor( 'xterm' )
   }
   stages {
      stage( 'Coloured Outputs with git commit id' ) {
         steps {
             script{
                 logs.info "SUCCESS"
                 logs.warn "WARNING"
                 def gitId=sh(script: 'git rev-parse HEAD' , returnStdout: true)
                 logs.gitCommitId(gitId)
             }
         }
      }
  }
}
```

```
[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.