upGrad

#LifeKoKaroLift

Working With Jenkins

upGrad

Course: Introduction to CI/CD With Jenkins

Lecture on: Working With Jenkins

Instructor: Dr. B Thangaraju

Today’s Agenda

- Creating users and managing privileges
- Configuring role-based strategy plugin
- Integrating Jenkins with Git via Poll scm and web hooks
- Analysing Poll scm vs web hooks
- Understanding Jenkins pipeline and its need
- Creating declarative pipeline and using Jenkins file
- Distributed architecture and its need
- Setting up master slave architecture in Jenkins

Understand How to Manage Users in Jenkins

Importance of User Management

- In a typical project environment, many employees working on a project can access the Jenkins server to run their build or test jobs. This can create security and authorization issues.

- So, you need to give every Jenkins user appropriate permission to enable the Jenkins server's safety and security.

- In Jenkins, different configuration options are available to enable, edit or disable various security features.

- By default, anonymous users have no permissions and logged in users have complete control.

Importance of User Management

- Jenkins admin manages these users based on their roles. Jenkins provides capabilities to add users, edit users and provide different roles to each user. For this, Jenkins provides a role-based authentication plugin.

- The ‘Configure Global Security’ option helps a Jenkins administrator to enable, configure or disable key security features to the Jenkins environment.

Security Realm

upGrad

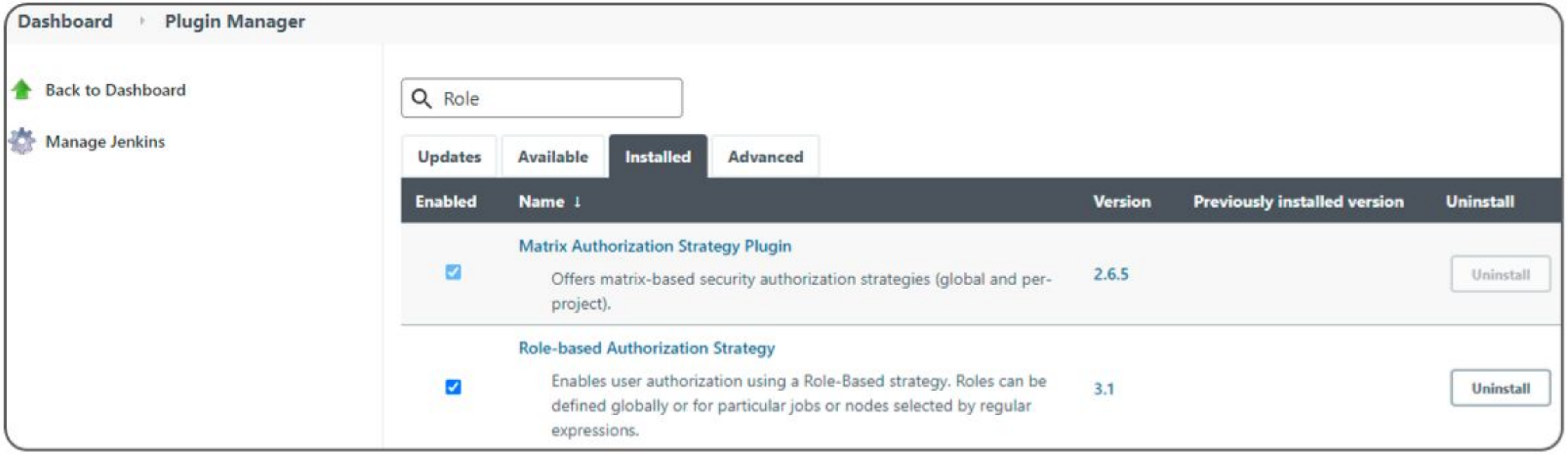
- To configure authentication and authorization schemes in Jenkins, you need to use Security Realm and Authorization configurations.
- Security Realm informs the Jenkins environment how and from where to pull user information.
- Authorization configuration informs the Jenkins environment about which users can access which aspects of Jenkins and to what extent.
- The Security Realm/authentication specifies who can access the Jenkins environment, whereas Authorization specifies what they can access.
- Matrix-based security allows the administrator a granular control over assigning users:
  - Provides the most security and flexibility
  - Recommended for production environments

Role-Based Authentication Strategy

- Used to add a new role-based mechanism to manage users' permission
- Roles can be assigned to users and user groups
- Global Roles: Admin, Developer, Tester, QA and Anonymous
- Allow to set permission: Agent, Job, Run, View and SCM
- Project/Item roles: Allow additional access control for each project separately in the Project configuration screen and give access control to specific user to access only the specified projects
- Agent roles: Set node-related permissions

Required Plugins

upGrad



Dashboard › Plugin Manager

Back to Dashboard

Manage Jenkins

Role

Updates | Available | Installed | Advanced

Enabled	Name ↓	Version	Previously installed version	Uninstall
✓	Matrix Authorization Strategy Plugin Offers matrix-based security authorization strategies (global and per-project).	2.6.5		Uninstall
✓	Role-based Authorization Strategy Enables user authorization using a Role-Based strategy. Roles can be defined globally or for particular jobs or nodes selected by regular expressions.	3.1		Uninstall

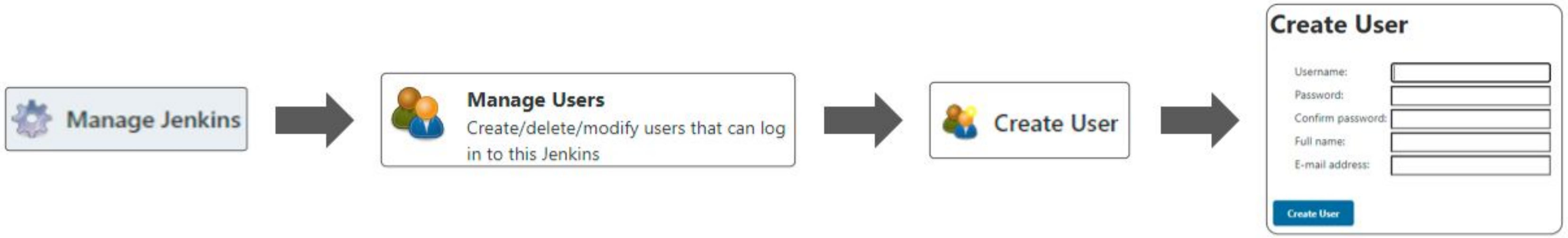
Demo

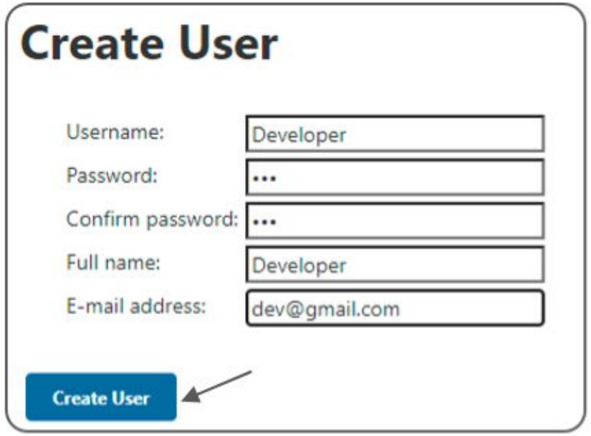
upGrad

- Create two users: Developer and Tester
- Configure Global Security: Enable Role-based Authentication strategy
- Manage and Assign roles:
    - Manage roles: Create a Global Roles - add ProjectMember and enable only required access
    - Item roles: Create two roles - Developer and Tester. Enable all the options.
    - Manage and Assign roles: Add Developer and Tester as ProjectMember
    - Item roles: Developer and Tester users should be assigned to Developer and Tester roles, respectively. Set pattern as Prog.* for Developer and Test.* for Tester
- Create two projects: Program1 and TestProject1
- Login as Developer or Tester and view/build/create/delete the Projects

User Account Creation

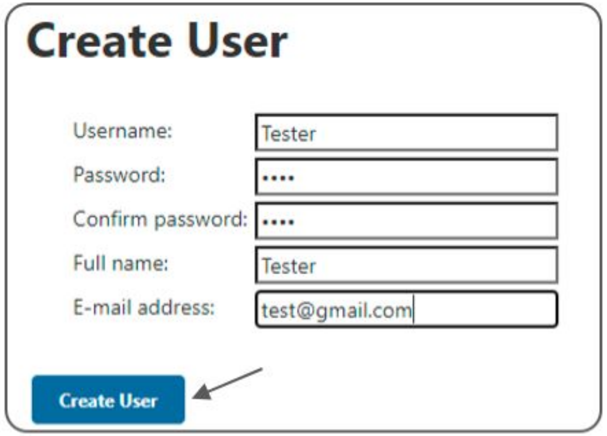
upGrad





Create User

Username:	Developer
Password:	•••
Confirm password:	•••
Full name:	Developer
E-mail address:	dev@gmail.com

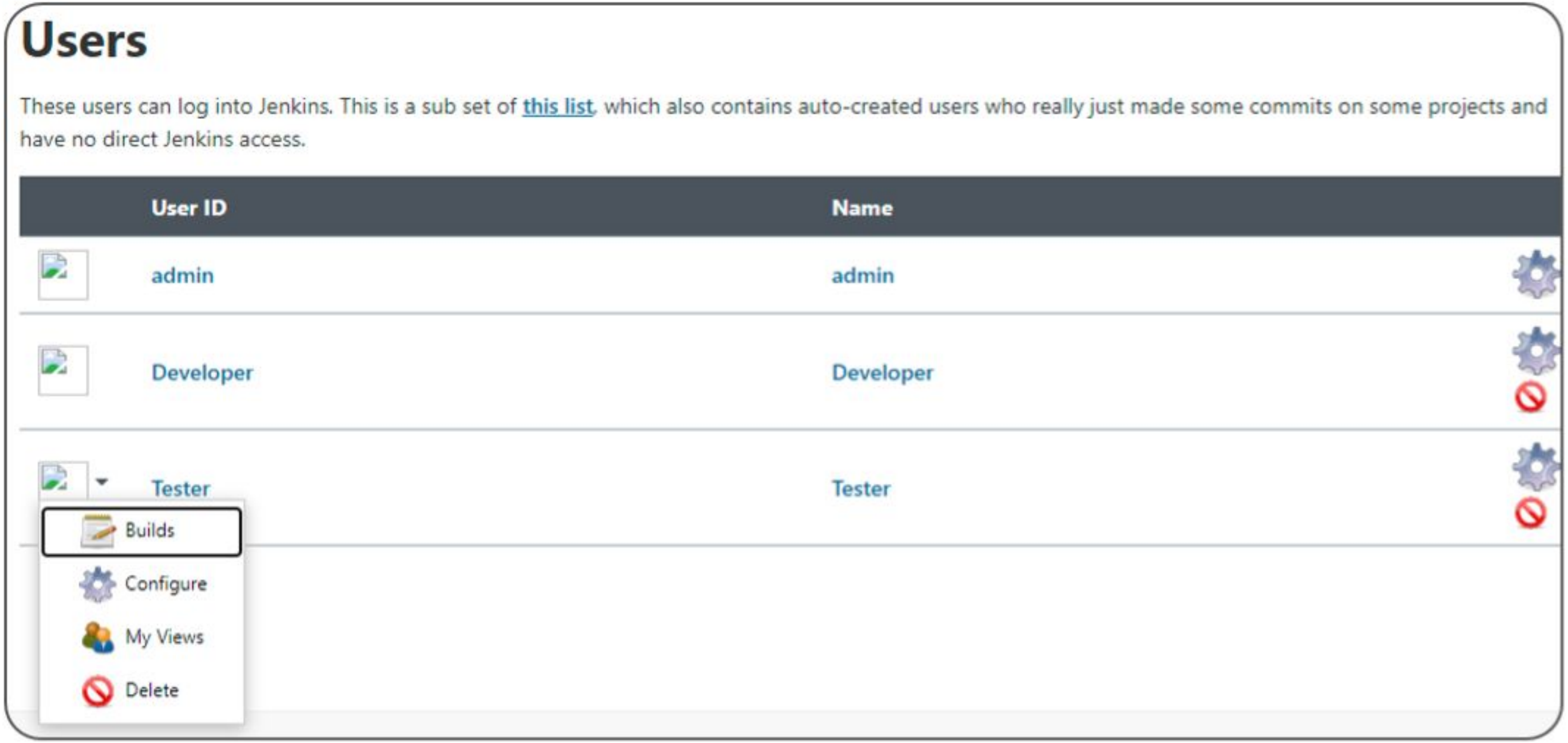


Create User

Username:	Tester
Password:	••••
Confirm password:	••••
Full name:	Tester
E-mail address:	test@gmail.com

List and Delete User

upGrad



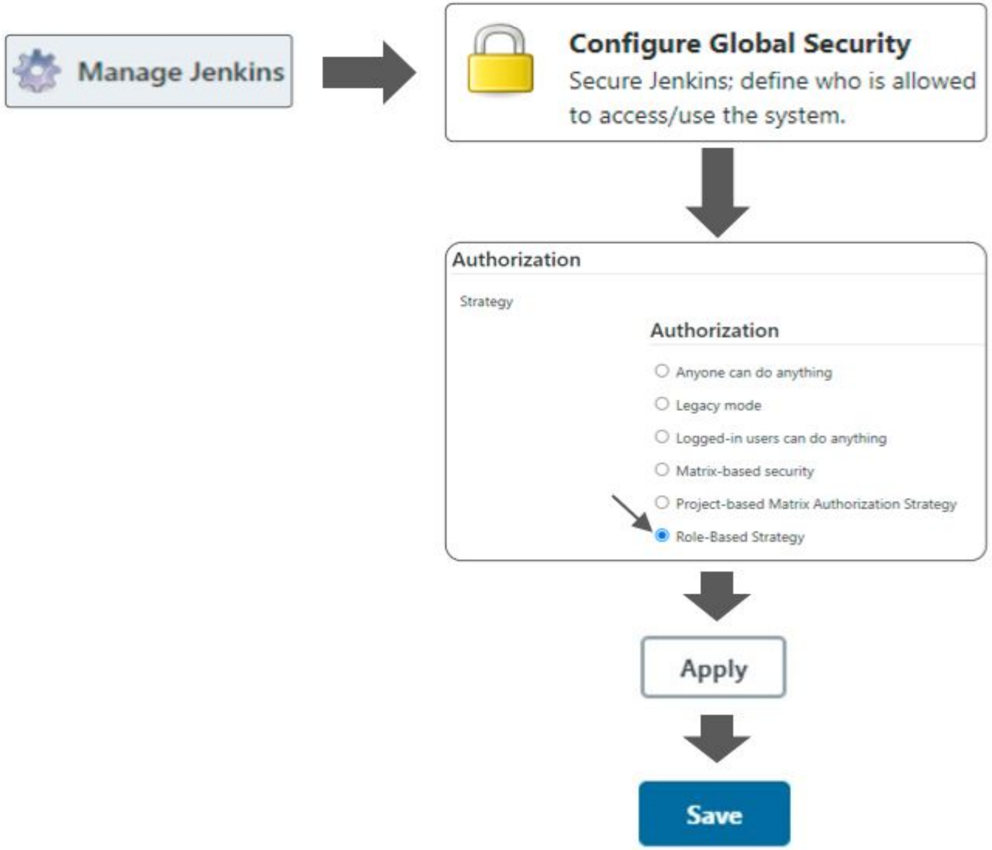
Users

These users can log into Jenkins. This is a sub set of this list, which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

	User ID	Name
	admin	admin
	Developer	Developer
	Tester	Tester

Enable Role-Based Authentication Strategy

upGrad



Manage Jenkins

Configure Global Security

Secure Jenkins; define who is allowed to access/use the system.

Authorization

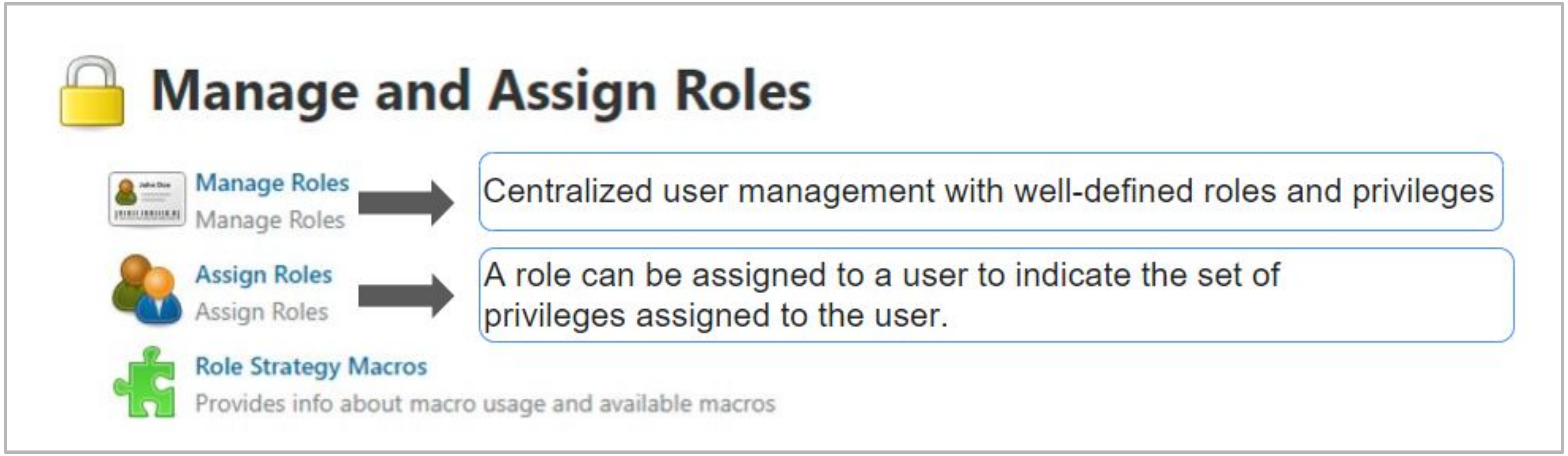
Strategy

Authorization

- Anyone can do anything
- Legacy mode
- Logged-in users can do anything
- Matrix-based security
- Project-based Matrix Authorization Strategy
- [✓ Role-Based Strategy]

Manage and Assign Roles

Handle permissions by creating roles and assigning them to users/groups



Manage and Assign Roles

Manage Roles
Manage Roles

Centralized user management with well-defined roles and privileges

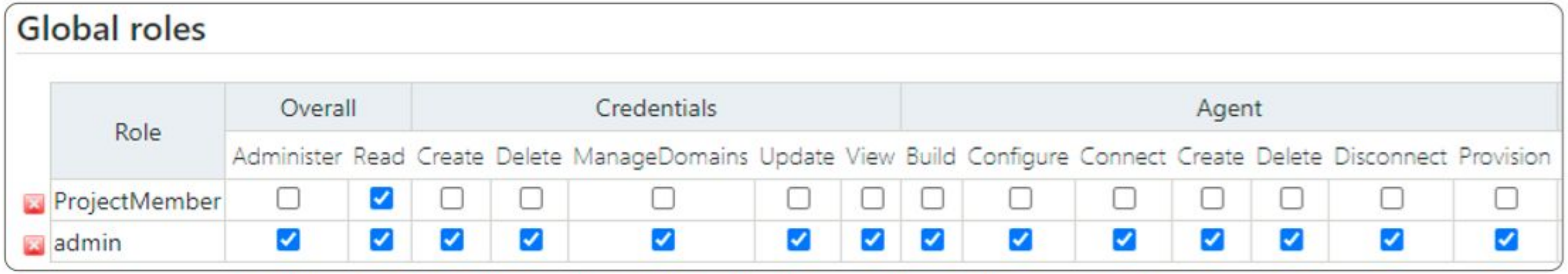
Assign Roles
Assign Roles

A role can be assigned to a user to indicate the set of privileges assigned to the user.

Role Strategy Macros
Provides info about macro usage and available macros

Manage Roles

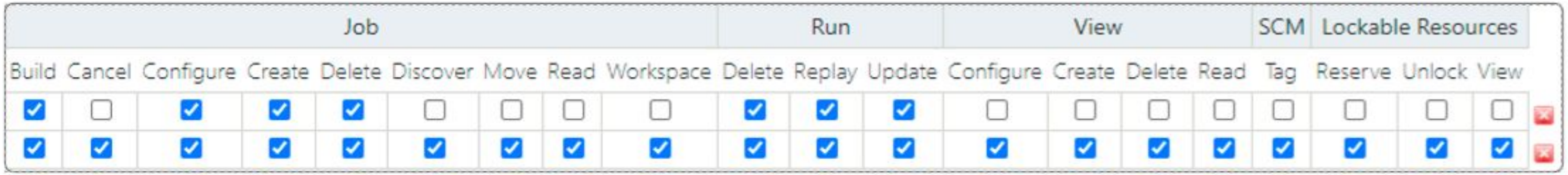
upGrad



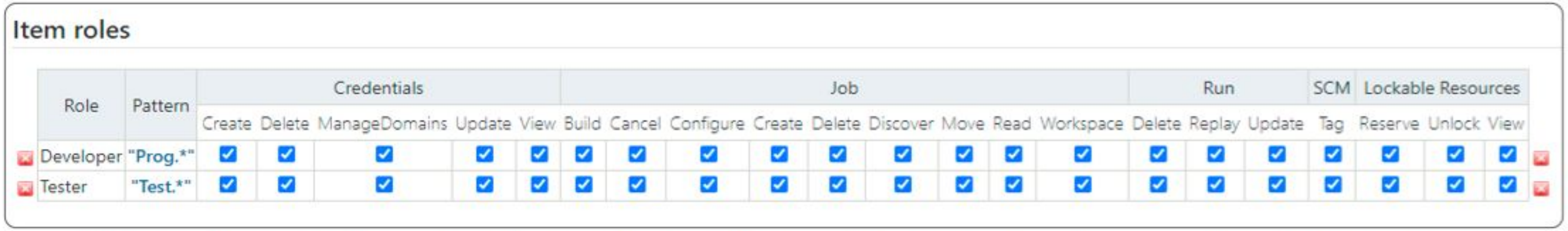
Global roles

Role	Overall		Credentials					Agent						
	Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Provision
ProjectMember	☐	✓	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
admin	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Add ProjectMember role in Global roles and enable required permissions



Job									Run			View				SCM	Lockable Resources		
Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Configure	Create	Delete	Read	Tag	Reserve	Unlock	View
✓	☐	✓	✓	✓	☐	☐	☐	☐	✓	✓	✓	☐	☐	☐	☐	☐	☐	☐	☐
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

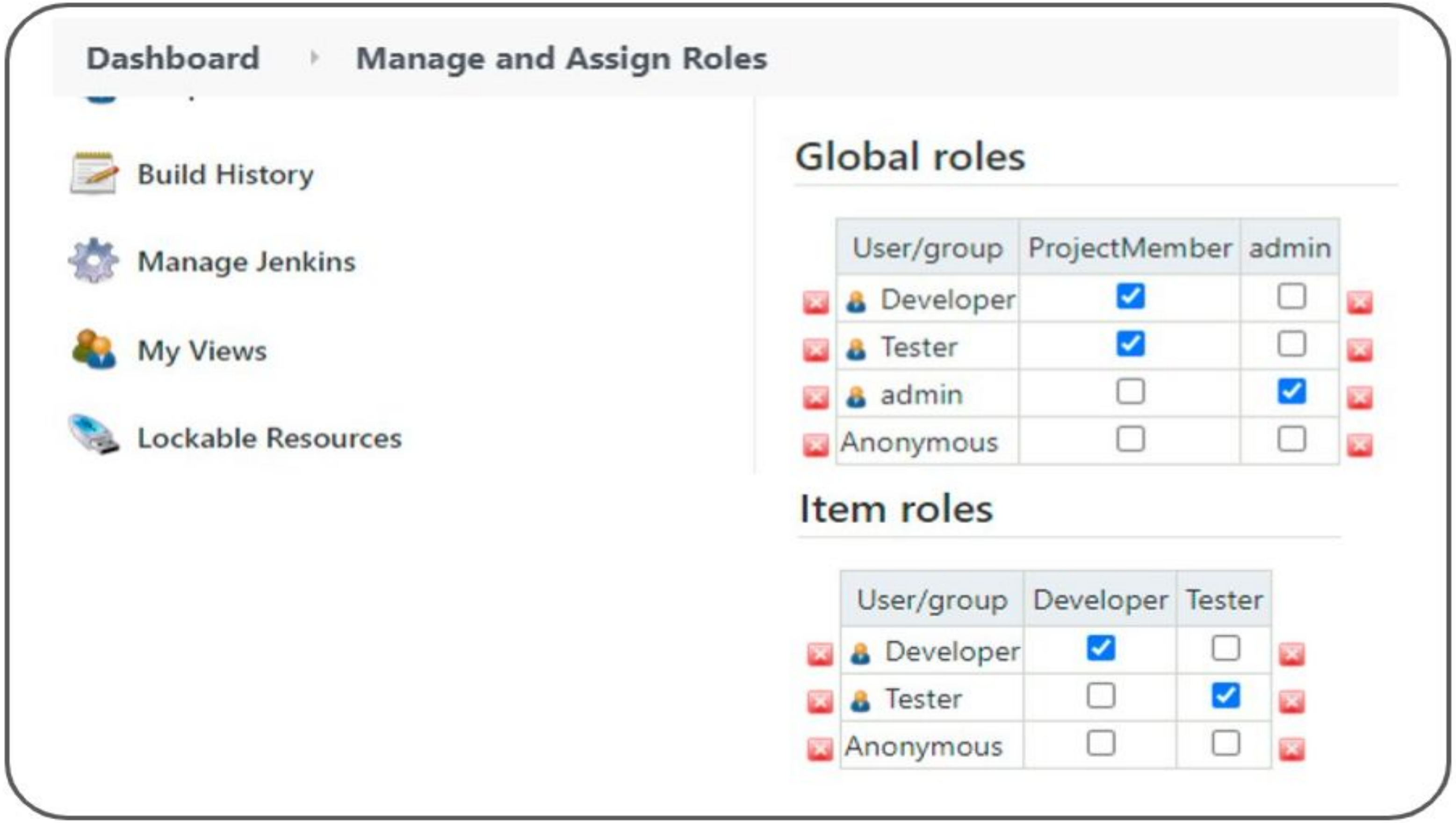


Item roles

Role	Pattern	Credentials					Job									Run			SCM	Lockable Resources		
		Create	Delete	ManageDomains	Update	View	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Tag	Reserve	Unlock	View
Developer	"Prog.*"	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Tester	"Test.*"	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Add Developer and Tester roles and set pattern as Prog.* for Developer and Test.* for Tester.

Assign Roles: Global and Item Roles



Dashboard › Manage and Assign Roles

Build History
Manage Jenkins
My Views
Lockable Resources

Global roles

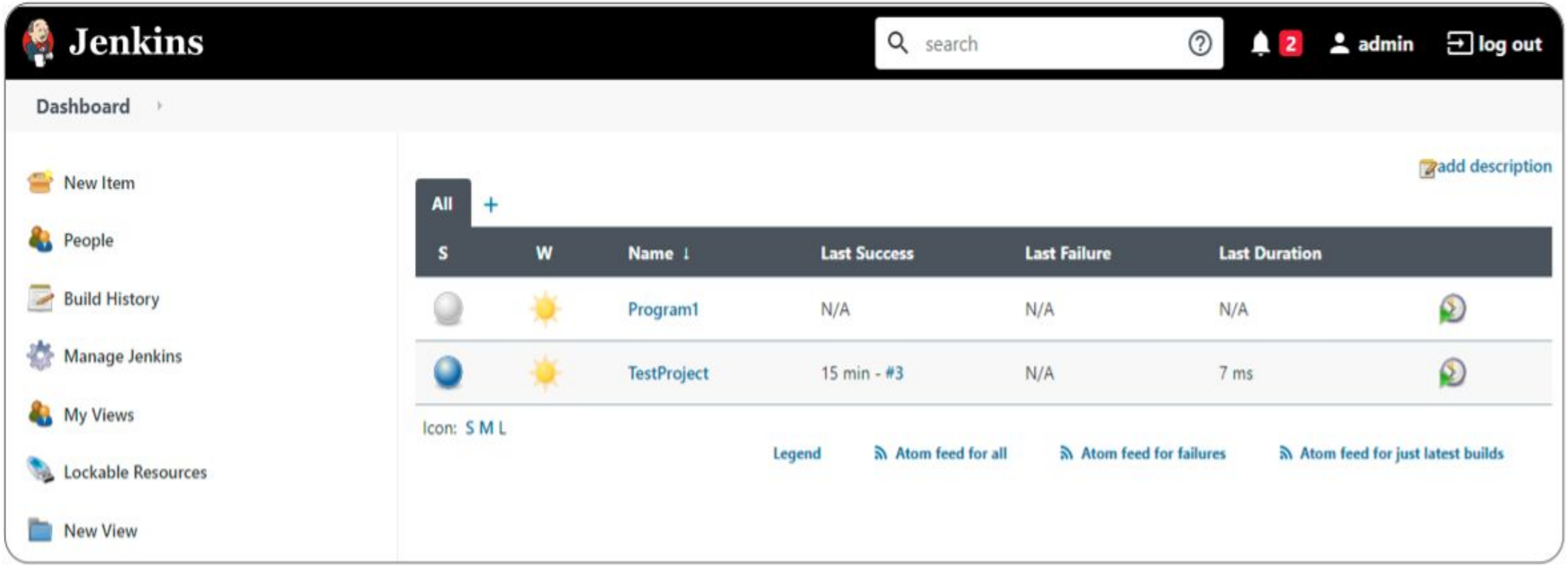
User/group	ProjectMember	admin
Developer	✓	☐
Tester	✓	☐
admin	☐	✓
Anonymous	☐	☐

Item roles

User/group	Developer	Tester
Developer	✓	☐
Tester	☐	✓
Anonymous	☐	☐

List of Projects Created

upGrad



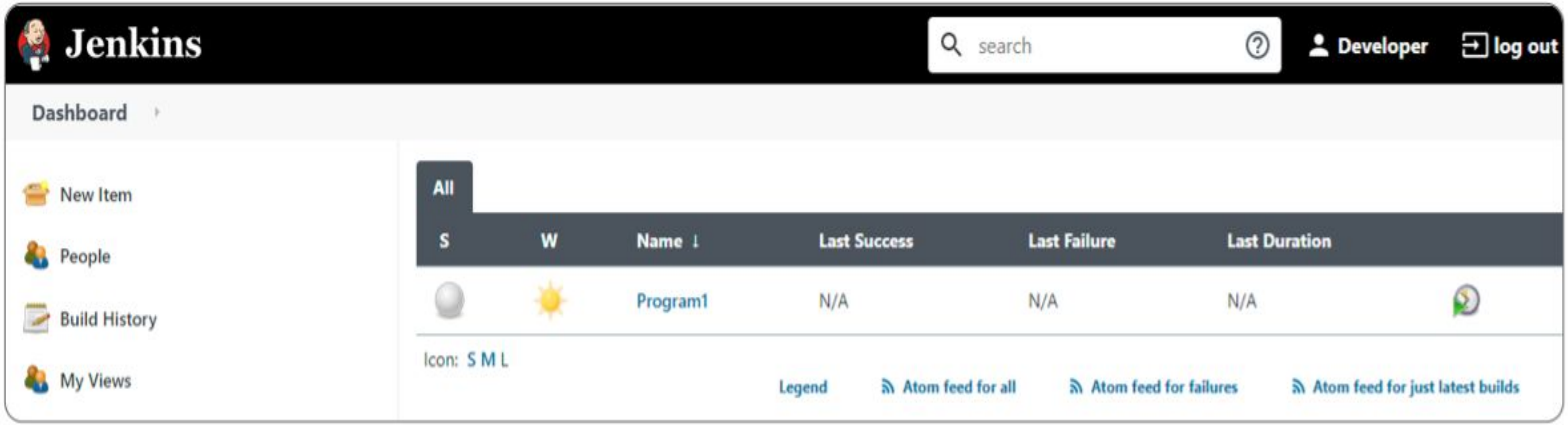
S	W	Name ↓	Last Success	Last Failure	Last Duration	
●	☀	Program1	N/A	N/A	N/A	
●	☀	TestProject	15 min - #3	N/A	7 ms	

Icon: S M L

Legend | Atom feed for all | Atom feed for failures | Atom feed for just latest builds

Login as Developer and View the Dashboard

upGrad

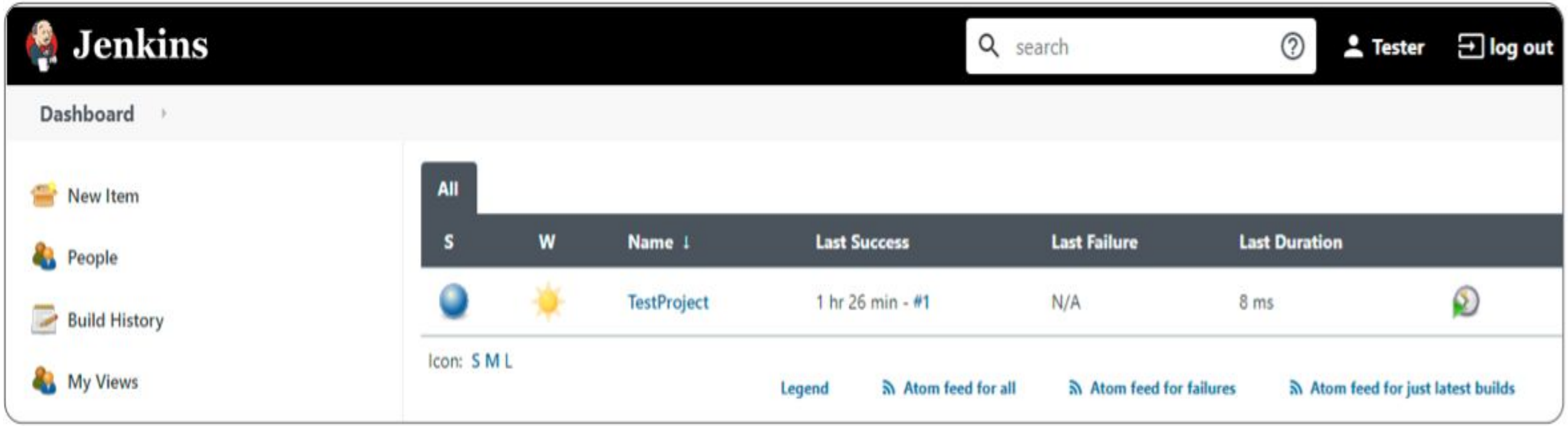


S	W	Name ↓	Last Success	Last Failure	Last Duration
		Program1	N/A	N/A	N/A

- Developer can view only the jobs started with Prog.*
- Developer can build the available Job
- Developer can create a new development job, but the job name should starts with Prog.
- Developer cannot see: Tester projects, configure the system and manage plugin

Login as Tester and View the Dashboard

upGrad


Jenkins

search

Tester log out

Dashboard

New Item

People

Build History

My Views

All

S	W	Name ↓	Last Success	Last Failure	Last Duration
		TestProject	1 hr 26 min - #1	N/A	8 ms

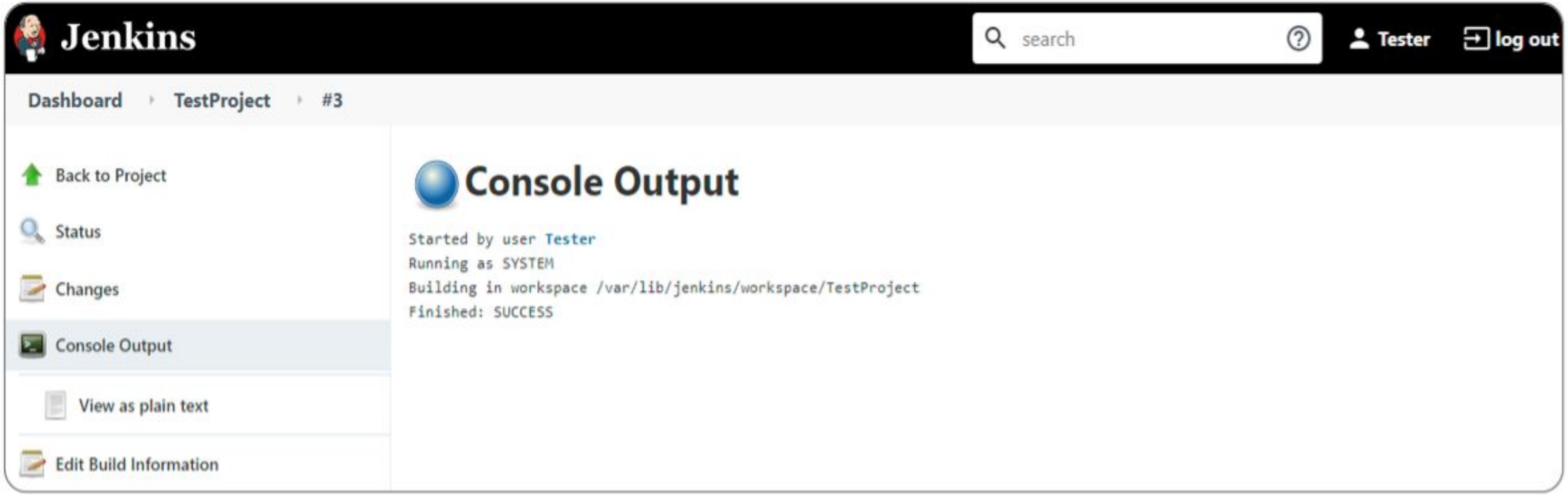
Icon: S M L

Legend Atom feed for all Atom feed for failures Atom feed for just latest builds

- Tester can view only the jobs started with Test.*
- Tester can build the available Job
- Tester can create a new test job, but the job name should starts with Test.
- Tester cannot see: Developer projects, configure the system and manage plugin

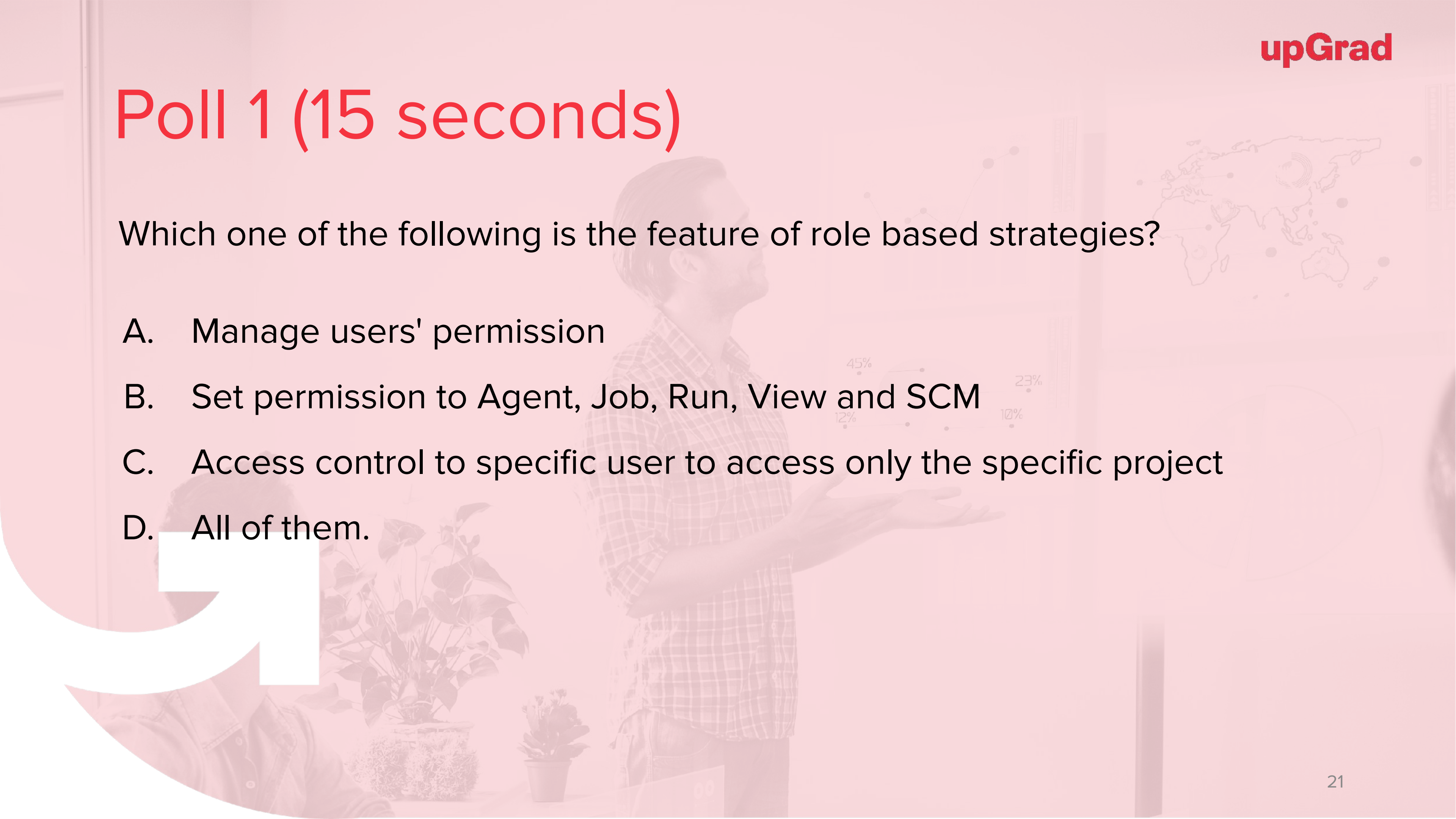
Tester: Build and Verify

upGrad



Console Output

```
Started by user Tester
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/TestProject
Finished: SUCCESS
```



Poll 1 (15 seconds)

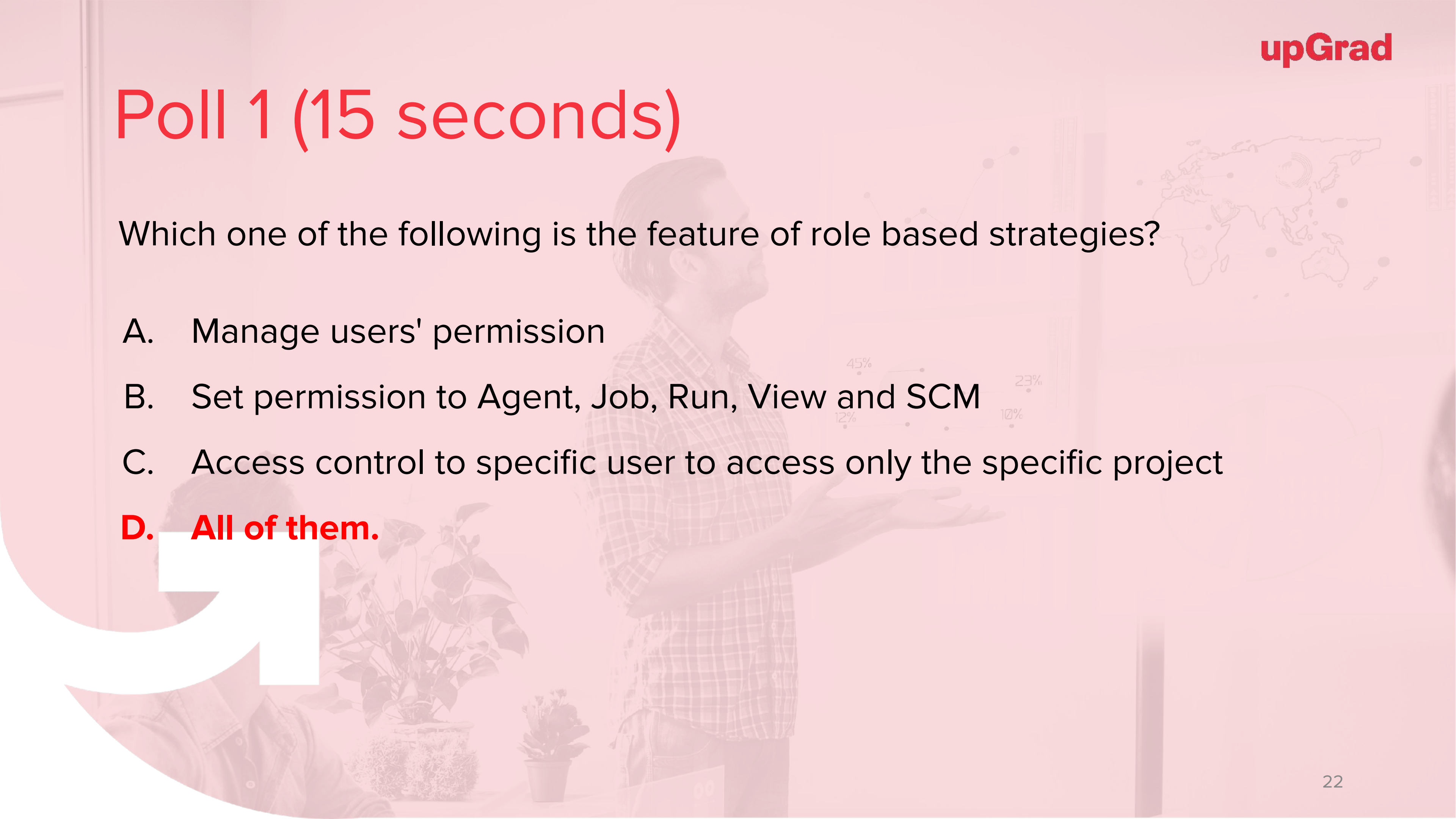
Which one of the following is the feature of role based strategies?

A. Manage users' permission

B. Set permission to Agent, Job, Run, View and SCM

C. Access control to specific user to access only the specific project

D. All of them.



Poll 1 (15 seconds)

Which one of the following is the feature of role based strategies?

A. Manage users' permission

B. Set permission to Agent, Job, Run, View and SCM

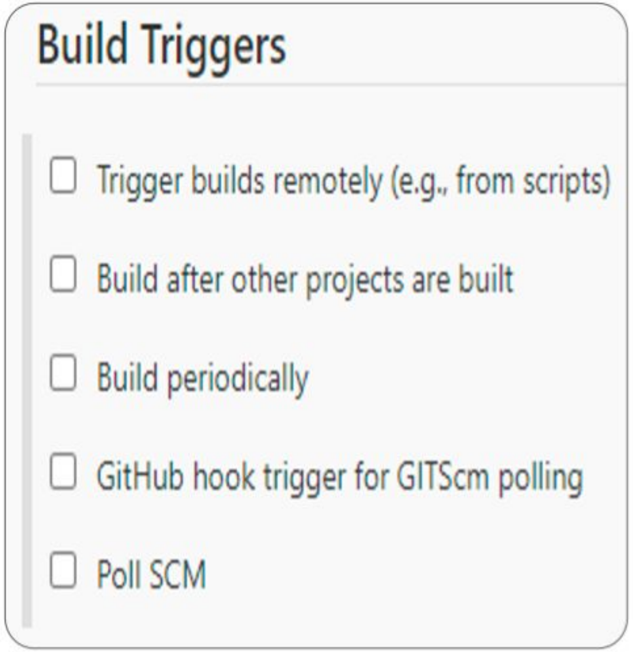
C. Access control to specific user to access only the specific project

D. All of them.

Integrating Jenkins With Git

Introduction

- So far, you have seen how to Trigger build remotely and build after other projects are built (chains of Jenkins job).
- Build periodically – You can trigger the jobs periodically with crontab time format.
- In this section, let's see how to GitHub hook trigger for GitScm polling and Poll SCM.
- For GitHub hook trigger for GITScm polling and Poll SCM, you will first need to integrate with GitHub repository.
- Next, Install Git Plugin.

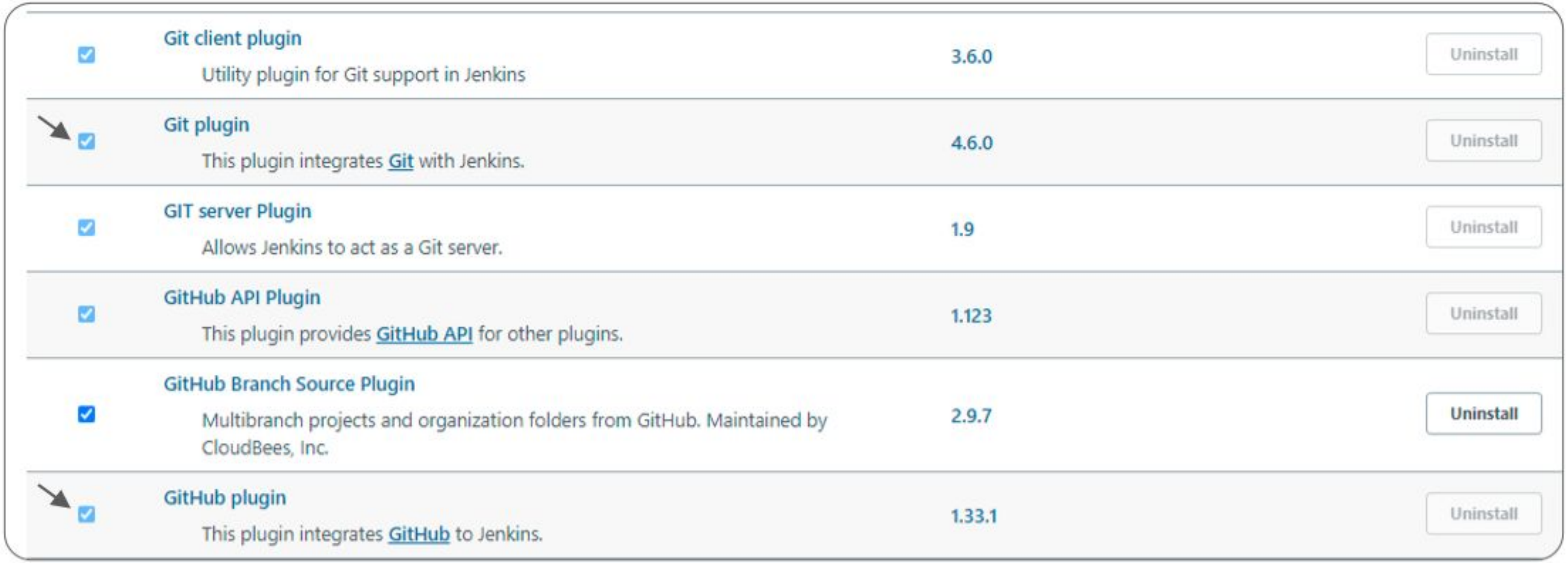


Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

Install Git Plugin

upGrad



✓	**Git client plugin** Utility plugin for Git support in Jenkins	3.6.0	Uninstall
✓	**Git plugin** This plugin integrates Git with Jenkins.	4.6.0	Uninstall
✓	**GIT server Plugin** Allows Jenkins to act as a Git server.	1.9	Uninstall
✓	**GitHub API Plugin** This plugin provides GitHub API for other plugins.	1.123	Uninstall
✓	**GitHub Branch Source Plugin** Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc.	2.9.7	Uninstall
✓	**GitHub plugin** This plugin integrates GitHub to Jenkins.	1.33.1	Uninstall

Poll SCM

Poll Source Code Management (SCM) vs Build Periodically

- **Build Periodically**: Jenkins builds periodically even if there are no changes in the project.

- **Poll SCM**: Jenkins builds periodically only if any new changes are made in the project.

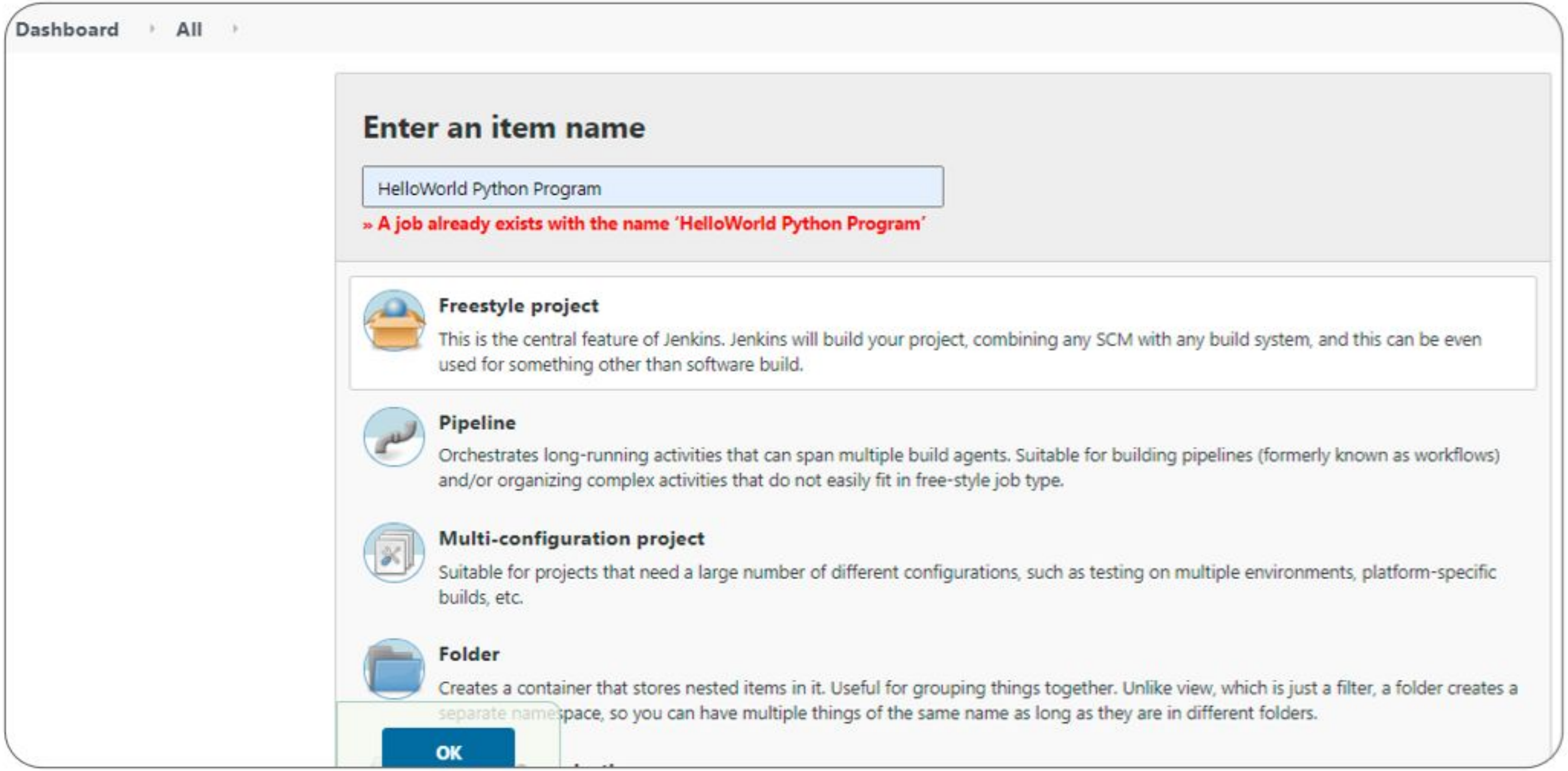
Poll SCM

- * * * * * - for every minute, Jenkins polls periodically the GitHub to check whether any new commits were made.

- If there are any changes pushed since the last build, then Jenkins automatically builds the project.

Create a New Freestyle Project

upGrad



Dashboard › All ›

Enter an item name

HelloWorld Python Program

» A job already exists with the name 'HelloWorld Python Program'

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.

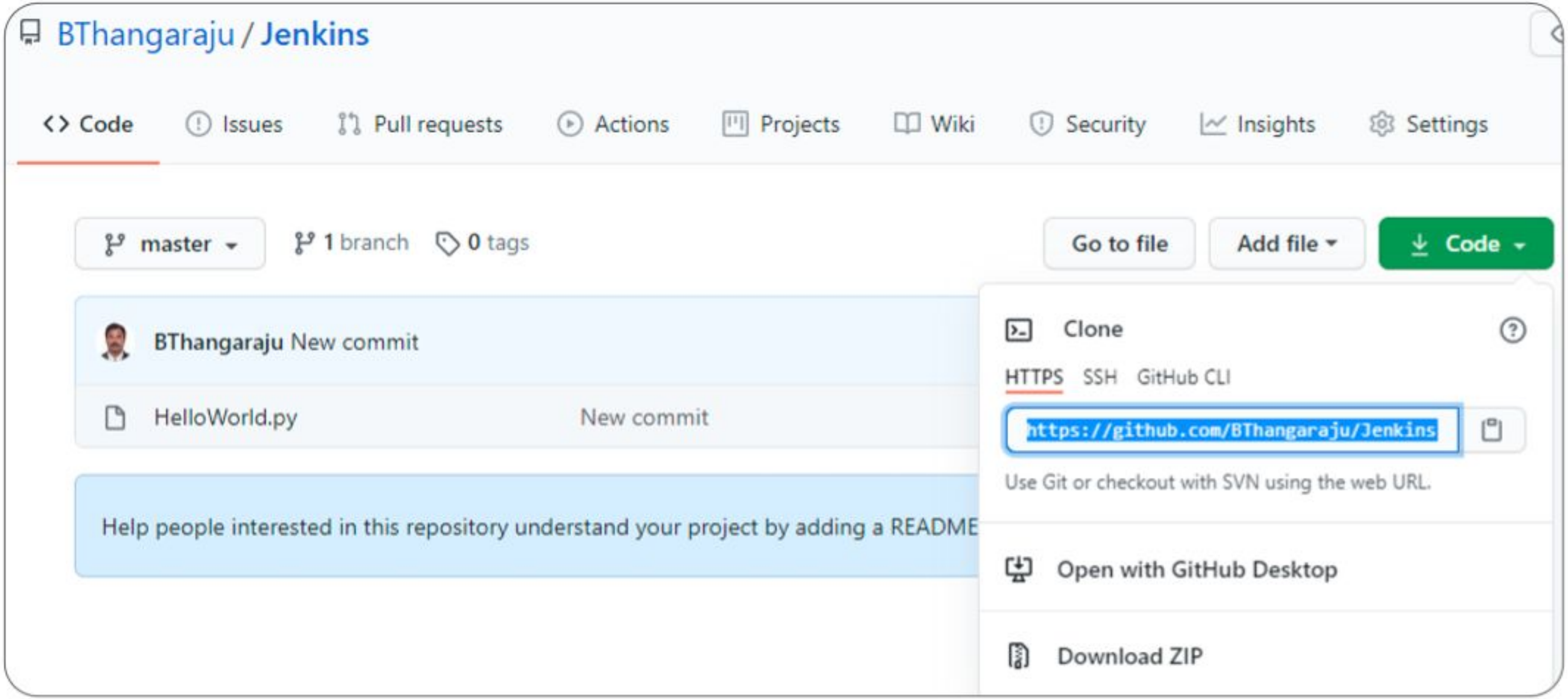
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Copy Your Project GitHub URL

upGrad



BThangaraju / Jenkins

BThangaraju New commit

HelloWorld.py New commit

Help people interested in this repository understand your project by adding a README

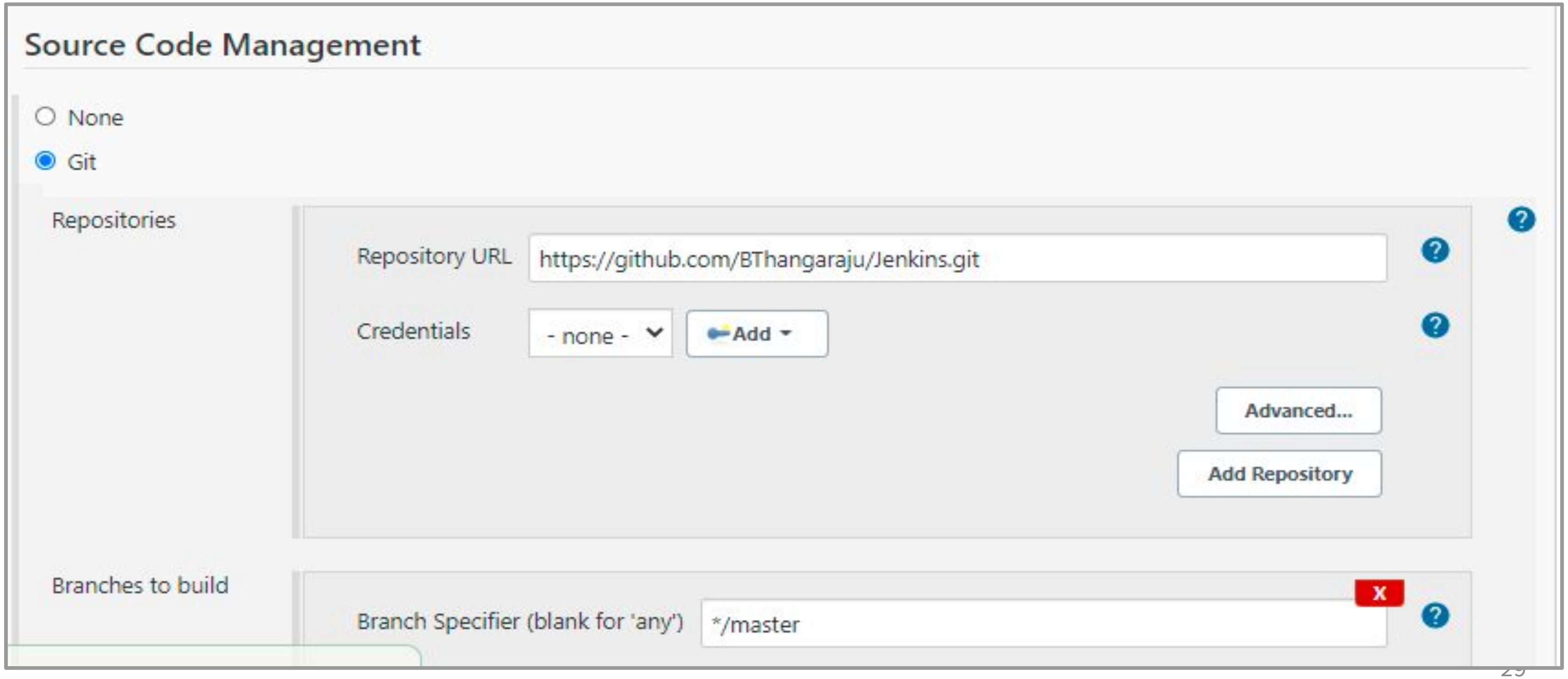
https://github.com/BThangaraju/Jenkins

Use Git or checkout with SVN using the web URL.

Enable Git

upGrad

- Select Git in the Source Code Management and enter GitHub URL in the given Repository URL option.



Source Code Management

None [✓ Git]

Repositories

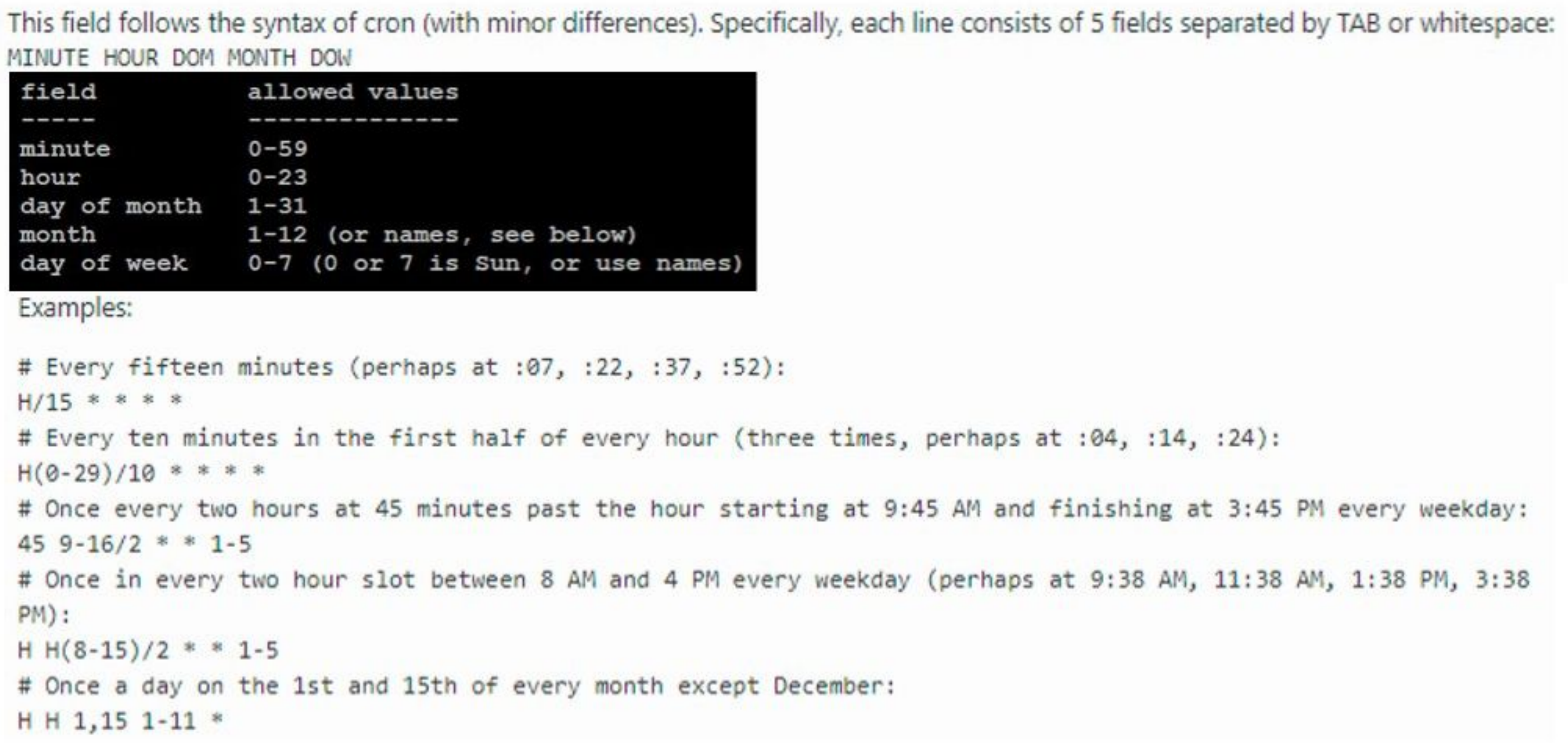
Repository URL: https://github.com/BThangaraju/Jenkins.git

Credentials: - none -

Branches to build

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

Poll SCM Fields and Format



This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

```
MINUTE HOUR DOM MONTH DOW
```

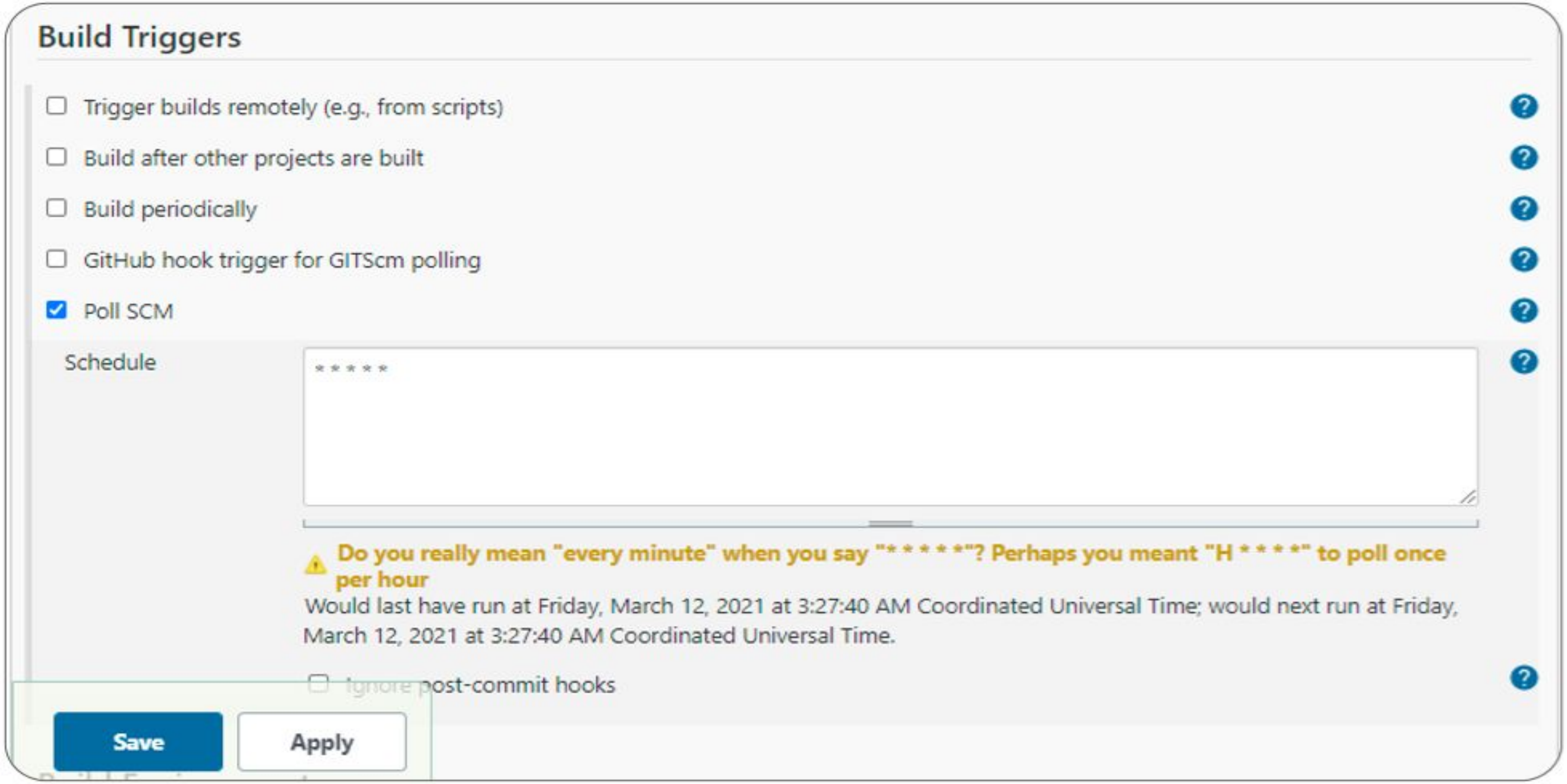
```
field          allowed values
-----          --------------
minute         0-59
hour           0-23
day of month   1-31
month          1-12 (or names, see below)
day of week    0-7 (0 or 7 is Sun, or use names)
```

Examples:

```
# Every fifteen minutes (perhaps at :07, :22, :37, :52):
H/15 * * * *
# Every ten minutes in the first half of every hour (three times, perhaps at :04, :14, :24):
H(0-29)/10 * * * *
# Once every two hours at 45 minutes past the hour starting at 9:45 AM and finishing at 3:45 PM every weekday:
45 9-16/2 * * 1-5
# Once in every two hour slot between 8 AM and 4 PM every weekday (perhaps at 9:38 AM, 11:38 AM, 1:38 PM, 3:38
PM):
H H(8-15)/2 * * 1-5
# Once a day on the 1st and 15th of every month except December:
H H 1,15 1-11 *
```

Select Poll SCM and Enter the Value

upGrad



Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [x] Poll SCM

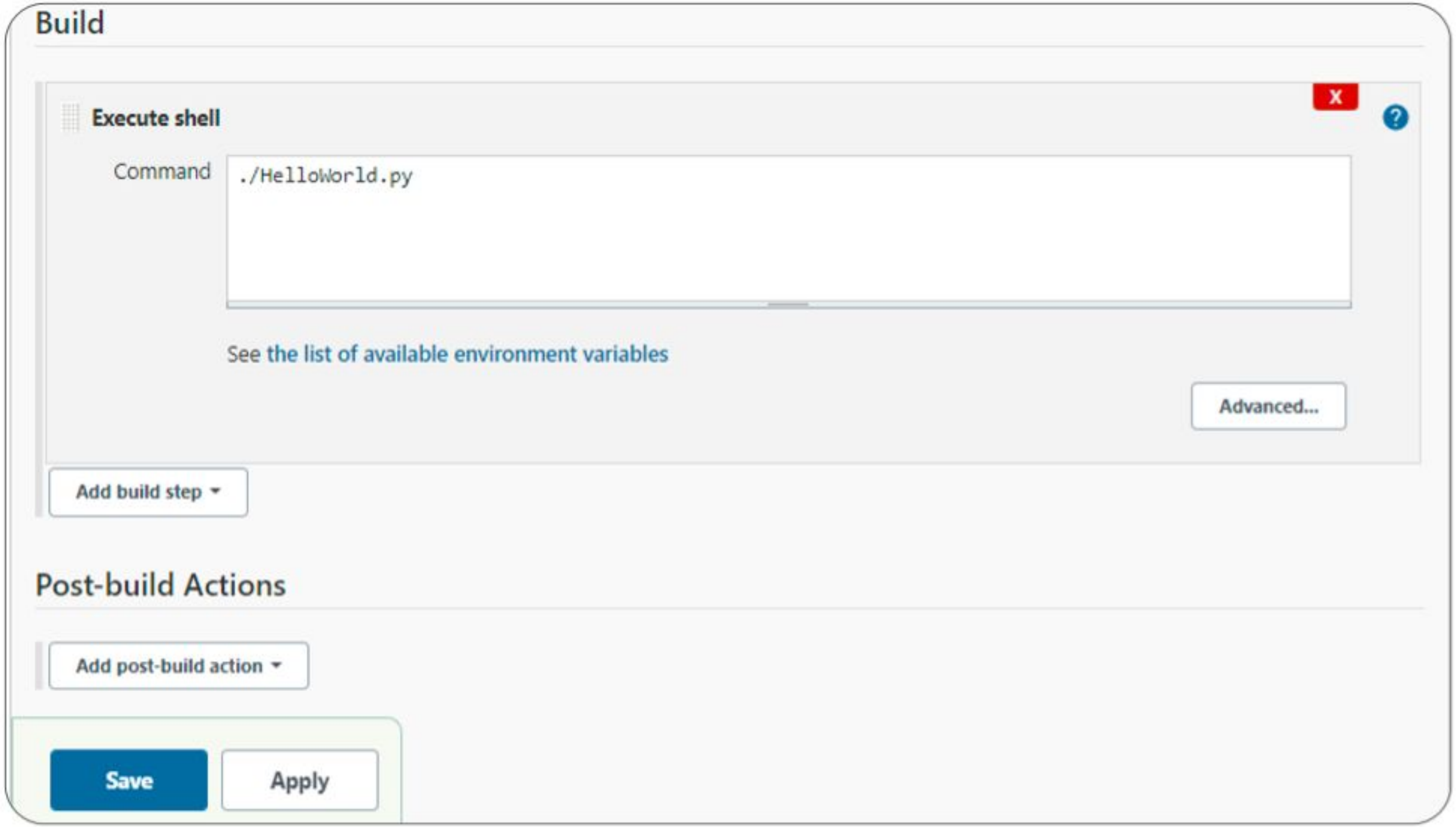
Schedule: `* * * * *`

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour

Would last have run at Friday, March 12, 2021 at 3:27:40 AM Coordinated Universal Time; would next run at Friday, March 12, 2021 at 3:27:40 AM Coordinated Universal Time.

- [ ] Ignore post-commit hooks

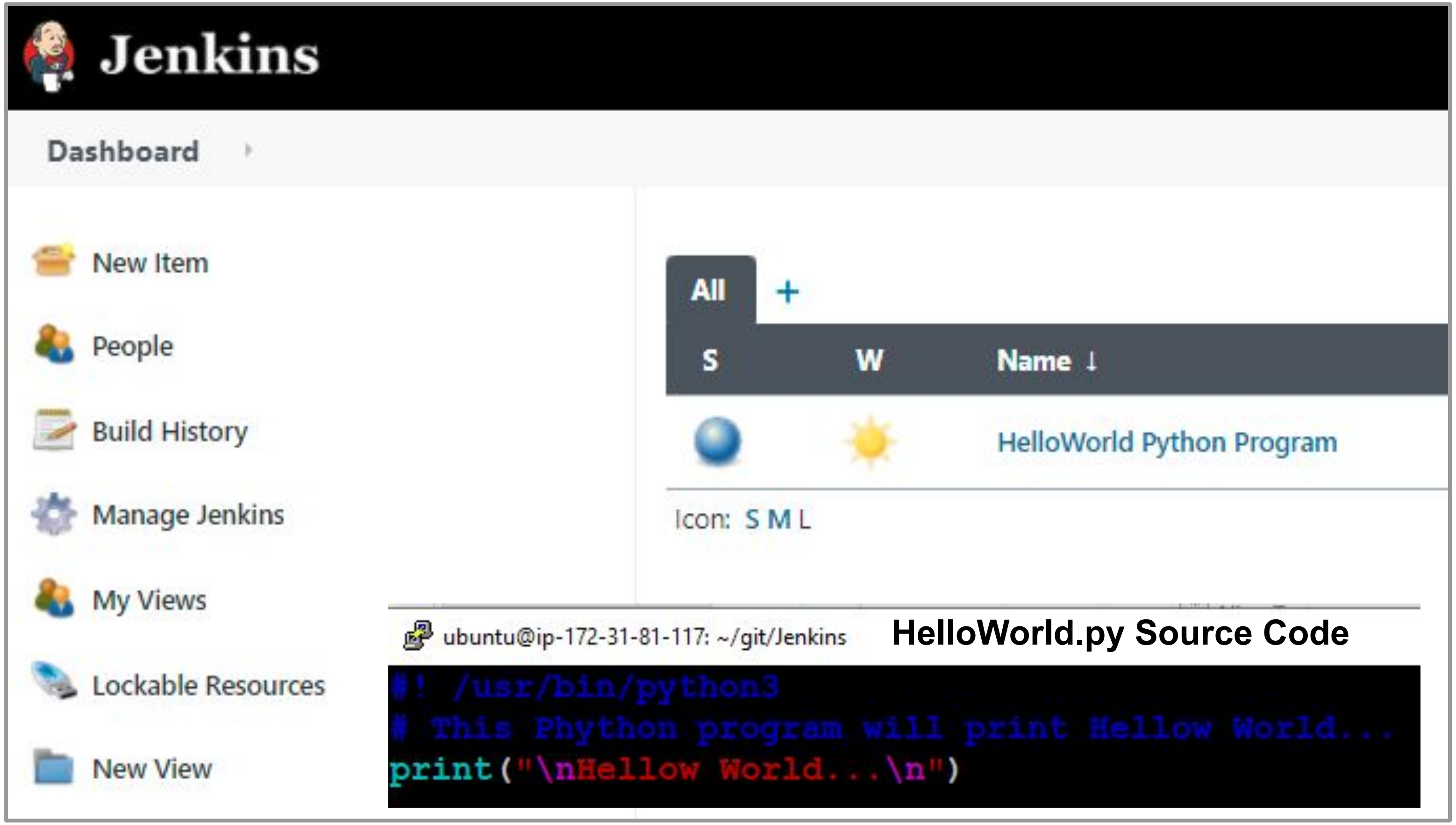
Enter Command to Execute the Project



```
./HelloWorld.py
```

Jenkins Dashboard

upGrad



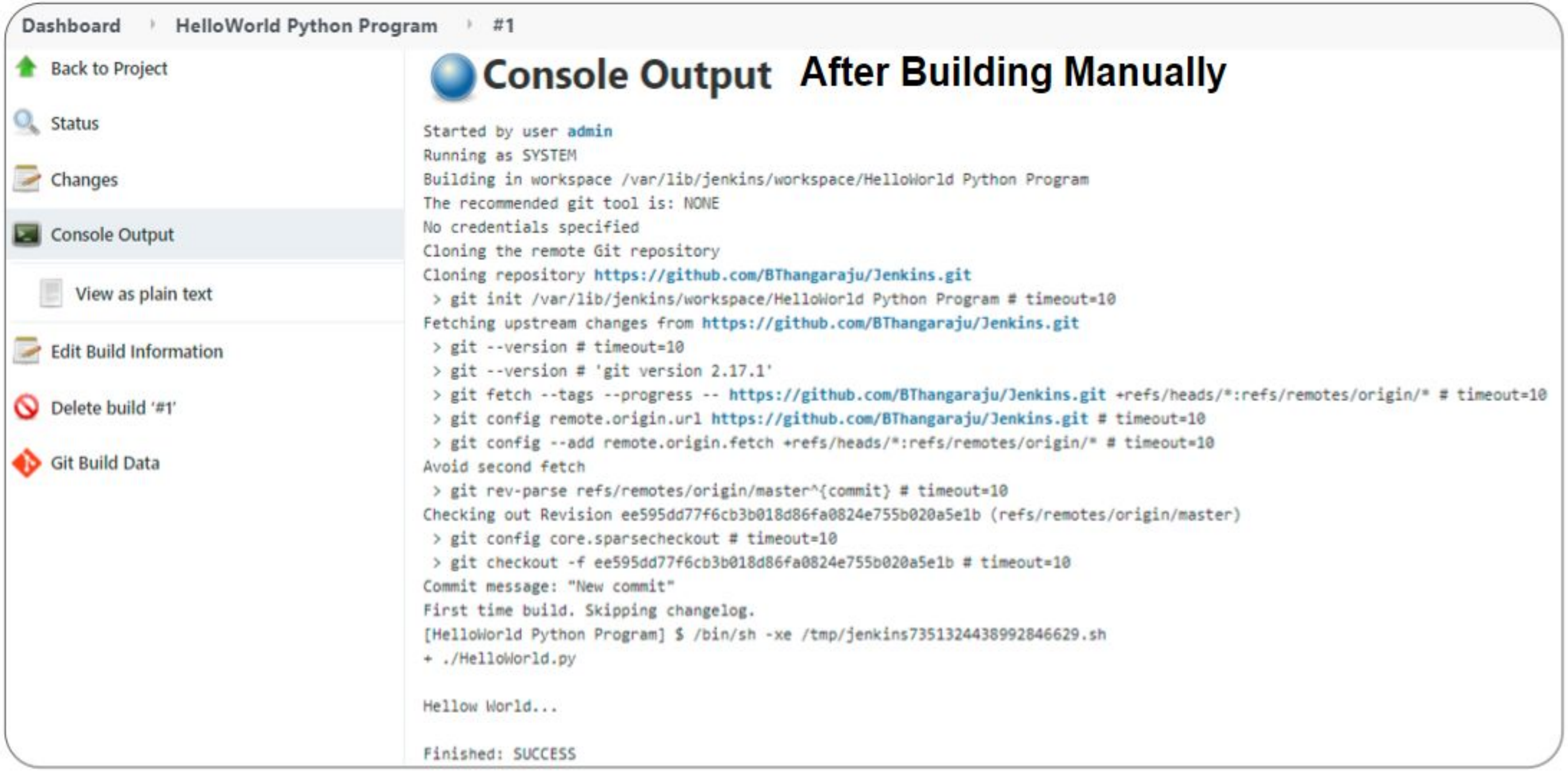
HelloWorld Python Program

HelloWorld.py Source Code

```
#! /usr/bin/python3
# This Phython program will print Hellow World...
print("\nHellow World...\n")
```

Console Output

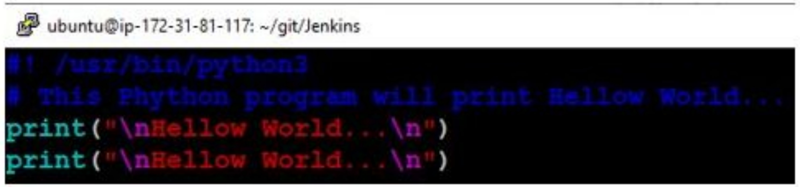
upGrad



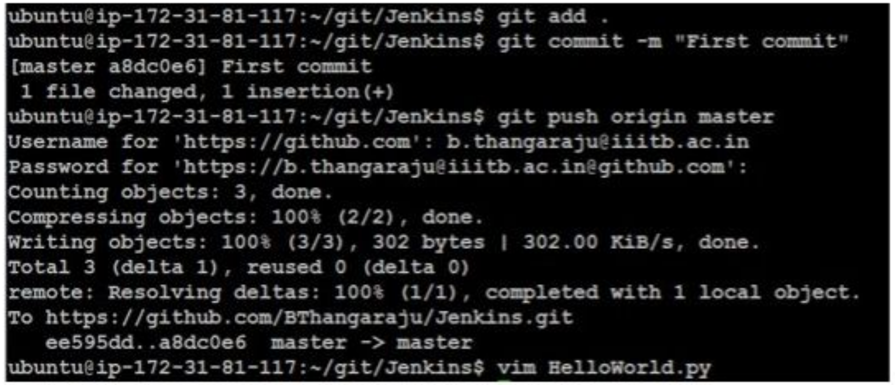
After Building Manually

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorld Python Program
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/BThangaraju/Jenkins.git
 > git init /var/lib/jenkins/workspace/HelloWorld Python Program # timeout=10
Fetching upstream changes from https://github.com/BThangaraju/Jenkins.git
 > git --version # timeout=10
 > git --version # 'git version 2.17.1'
 > git fetch --tags --progress -- https://github.com/BThangaraju/Jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/BThangaraju/Jenkins.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision ee595dd77f6cb3b018d86fa0824e755b020a5e1b (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f ee595dd77f6cb3b018d86fa0824e755b020a5e1b # timeout=10
Commit message: "New commit"
First time build. Skipping changelog.
[HelloWorld Python Program] $ /bin/sh -xe /tmp/jenkins7351324438992846629.sh
+ ./HelloWorld.py

Hellow World...

Finished: SUCCESS
```

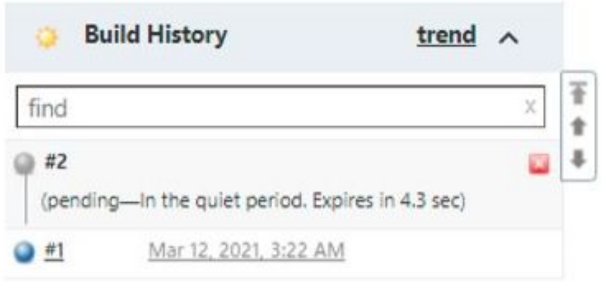
Commit Changes and Build Automatically

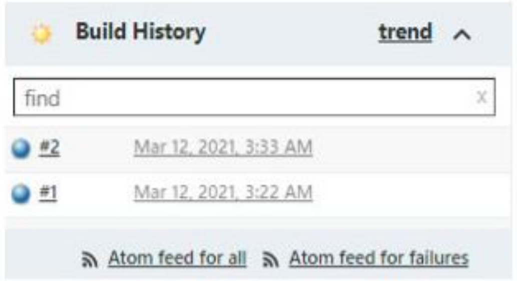

ubuntu@ip-172-31-81-117: ~/git/Jenkins

```
#! /usr/bin/python3
# This Phython program will print Hellow World...
print("\nHellow World...\n")
print("\nHellow World...\n")
```



```
ubuntu@ip-172-31-81-117:~/git/Jenkins$ git add .
ubuntu@ip-172-31-81-117:~/git/Jenkins$ git commit -m "First commit"
[master a8dc0e6] First commit
 1 file changed, 1 insertion(+)
ubuntu@ip-172-31-81-117:~/git/Jenkins$ git push origin master
Username for 'https://github.com': b.thangaraju@iiitb.ac.in
Password for 'https://b.thangaraju@iiitb.ac.in@github.com':
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 302 bytes | 302.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/BThangaraju/Jenkins.git
   ee595dd..a8dc0e6  master -> master
ubuntu@ip-172-31-81-117:~/git/Jenkins$ vim HelloWorld.py
```

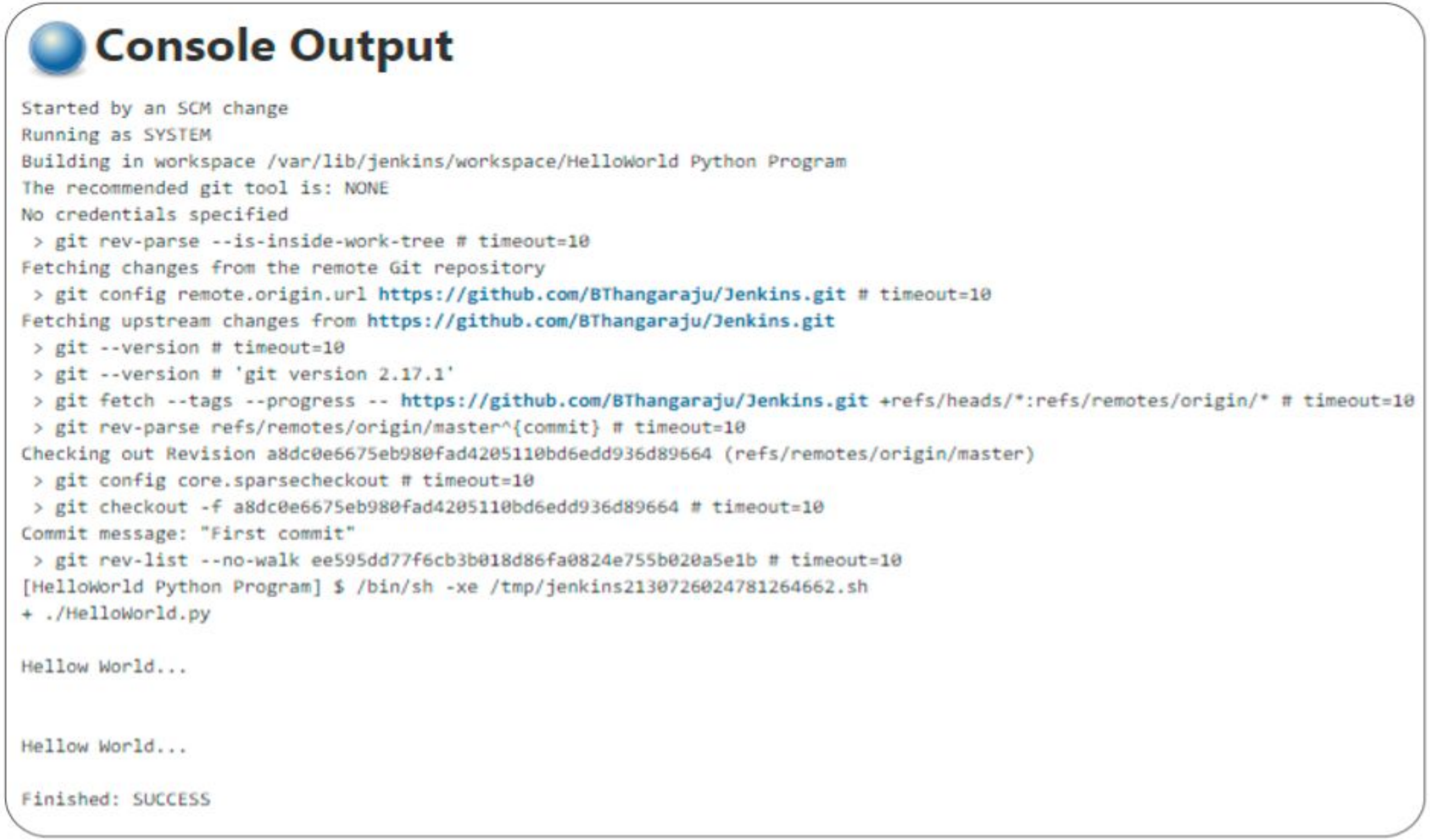




- After Committing changes in the local Git repo, we need to push the changes into our GitHub repository.
- Then Jenkins check the GitHub repository periodically and automatically build the Job.

Console Output

upGrad

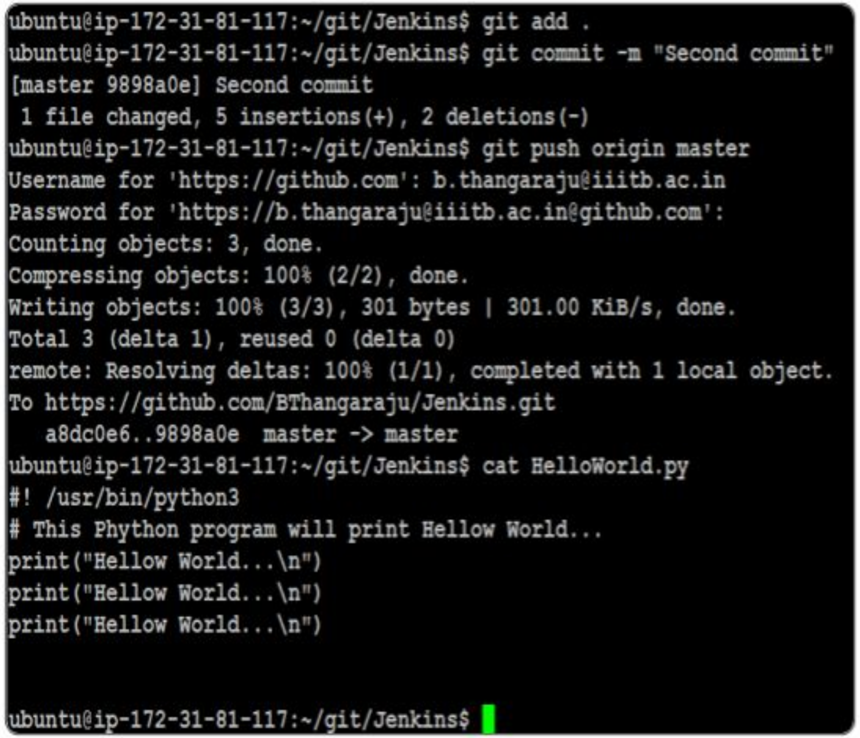


Console Output

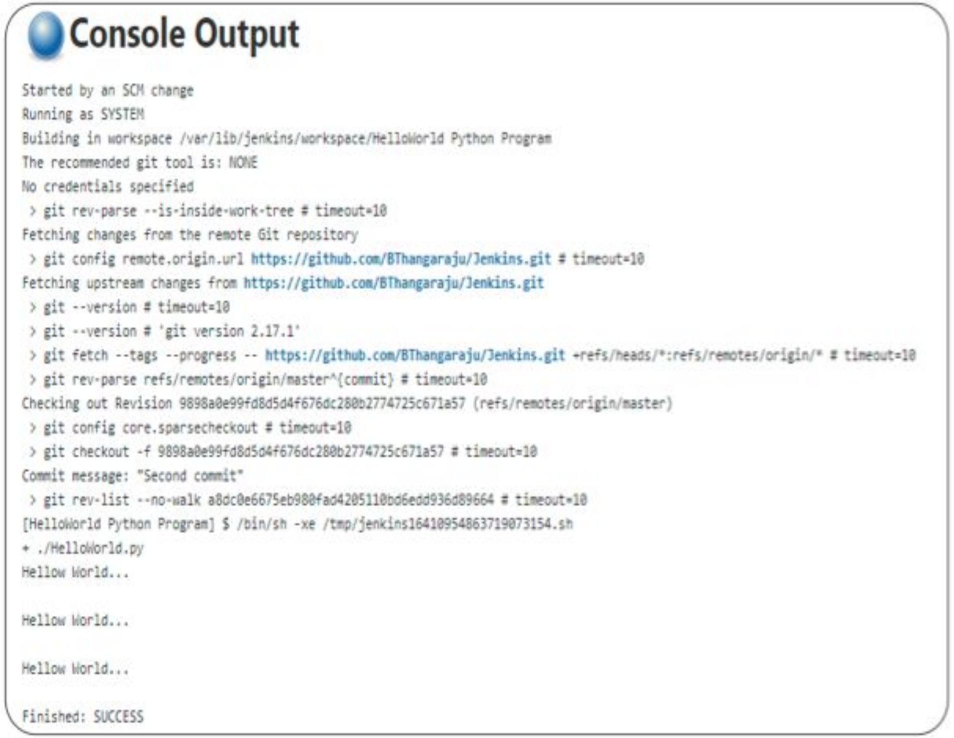
```
Started by an SCM change
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorld Python Program
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/BThangaraju/Jenkins.git # timeout=10
Fetching upstream changes from https://github.com/BThangaraju/Jenkins.git
 > git --version # timeout=10
 > git --version # 'git version 2.17.1'
 > git fetch --tags --progress -- https://github.com/BThangaraju/Jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision a8dc0e6675eb980fad4205110bd6edd936d89664 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f a8dc0e6675eb980fad4205110bd6edd936d89664 # timeout=10
Commit message: "First commit"
 > git rev-list --no-walk ee595dd77f6cb3b018d86fa0824e755b020a5e1b # timeout=10
[HelloWorld Python Program] $ /bin/sh -xe /tmp/jenkins2130726024781264662.sh
+ ./HelloWorld.py

Hellow World...


Hellow World...

Finished: SUCCESS
```

Commit One More Changes

upGrad



```
ubuntu@ip-172-31-81-117:~/git/Jenkins$ git add .
ubuntu@ip-172-31-81-117:~/git/Jenkins$ git commit -m "Second commit"
[master 9898a0e] Second commit
 1 file changed, 5 insertions(+), 2 deletions(-)
ubuntu@ip-172-31-81-117:~/git/Jenkins$ git push origin master
Username for 'https://github.com': b.thangaraju@iiitb.ac.in
Password for 'https://b.thangaraju@iiitb.ac.in@github.com':
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes | 301.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/BThangaraju/Jenkins.git
   a8dc0e6..9898a0e  master -> master
ubuntu@ip-172-31-81-117:~/git/Jenkins$ cat HelloWorld.py
#! /usr/bin/python3
# This Phython program will print Hellow World...
print("Hellow World...\n")
print("Hellow World...\n")
print("Hellow World...\n")


ubuntu@ip-172-31-81-117:~/git/Jenkins$
```



Console Output

```
Started by an SCM change
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/HelloWorld Python Program
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/BThangaraju/Jenkins.git # timeout=10
Fetching upstream changes from https://github.com/BThangaraju/Jenkins.git
 > git --version # timeout=10
 > git --version # 'git version 2.17.1'
 > git fetch --tags --progress -- https://github.com/BThangaraju/Jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 9898a0e99fd8d5d4f676dc280b2774725c671a57 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 9898a0e99fd8d5d4f676dc280b2774725c671a57 # timeout=10
Commit message: "Second commit"
 > git rev-list --no-walk a8dc0e6675eb980fad4205110bd6edd936d89664 # timeout=10
[HelloWorld Python Program] $ /bin/sh -xe /tmp/jenkins1641095486371907315 4.sh
+ ./HelloWorld.py
Hellow World...


Hellow World...


Hellow World...


Finished: SUCCESS
```

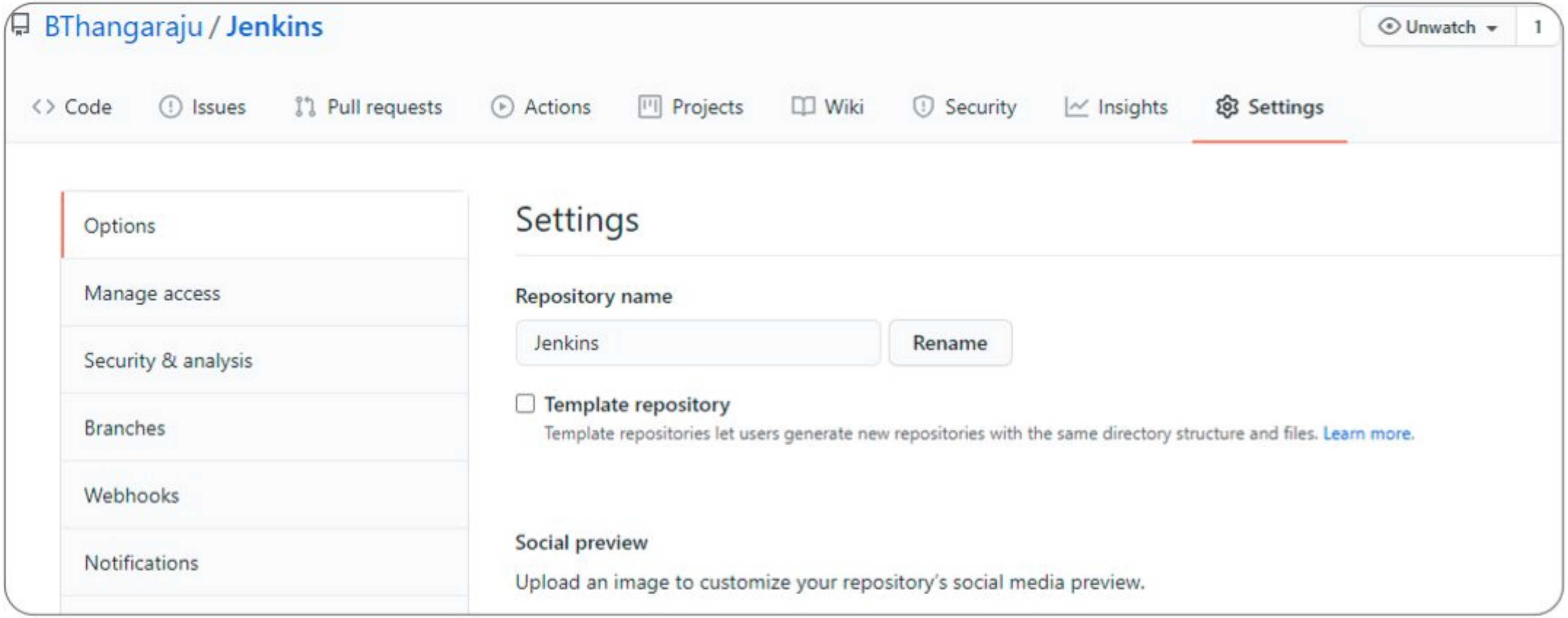
GitHub Hook Trigger for GITScm Polling

upGrad

- For GitHub hook trigger for GITScm polling, Jenkins will not build the job periodically; instead, when you commit new changes in the GitHub repository, only then will Jenkins trigger the job to build.

Working With Webhooks

upGrad



Settings

Repository name

Jenkins

Template repository

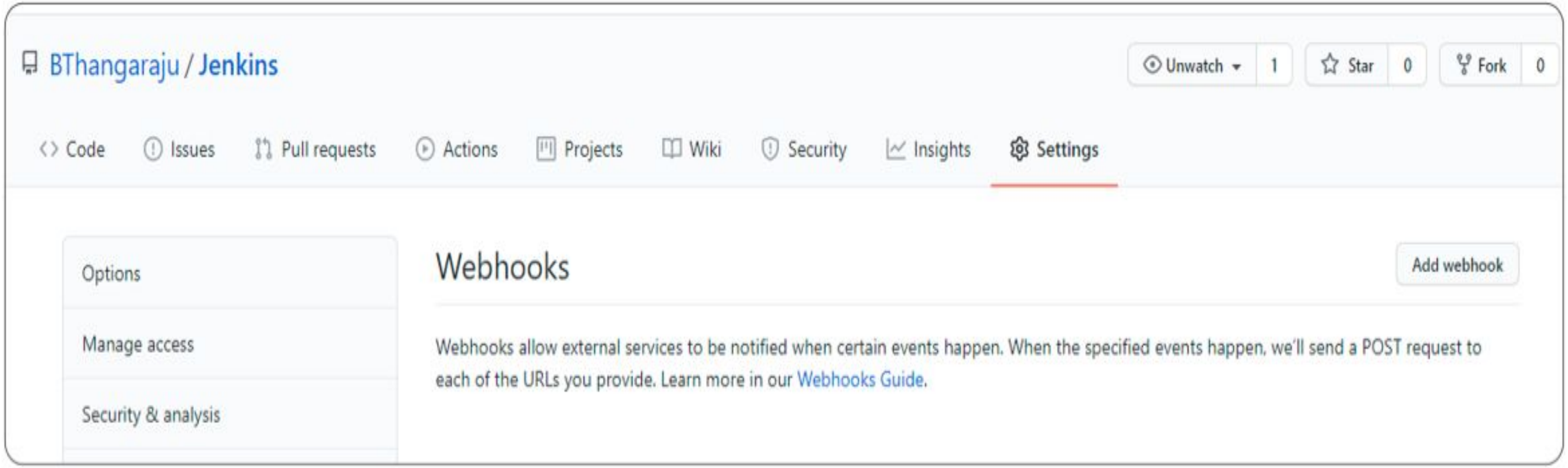
Template repositories let users generate new repositories with the same directory structure and files. Learn more.

Social preview

Upload an image to customize your repository's social media preview.

Working With Webhooks

upGrad

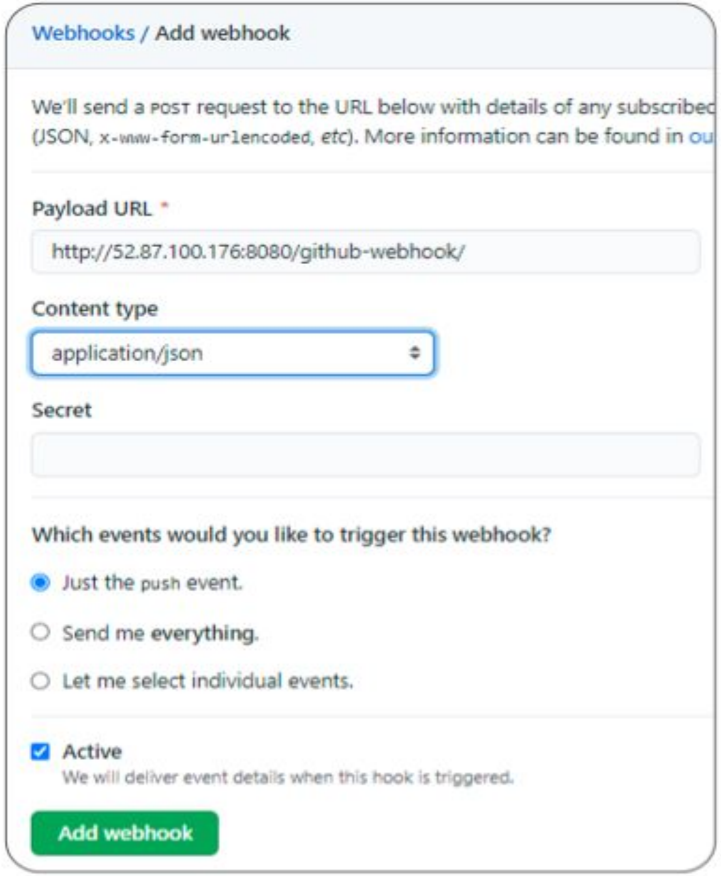


Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

Working With Webhooks

upGrad



Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed
(JSON, x-www-form-urlencoded, *etc*). More information can be found in ou

Payload URL *

http://52.87.100.176:8080/github-webhook/

Content type

application/json

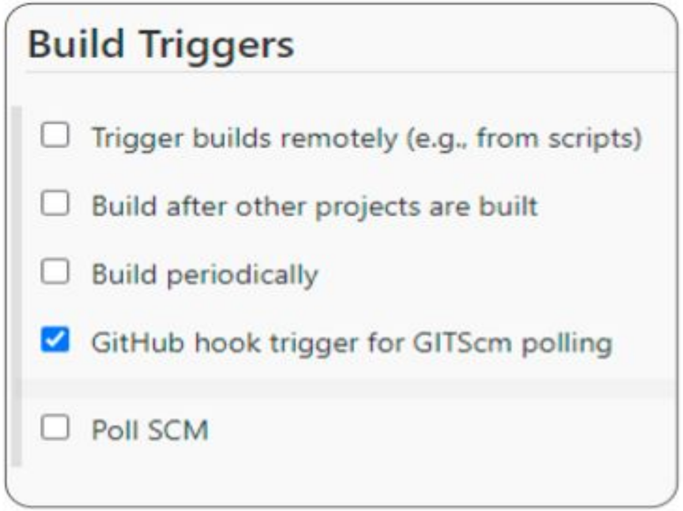
Secret

Which events would you like to trigger this webhook?

- [✓ Just the push event.]
- Send me everything.
- Let me select individual events.

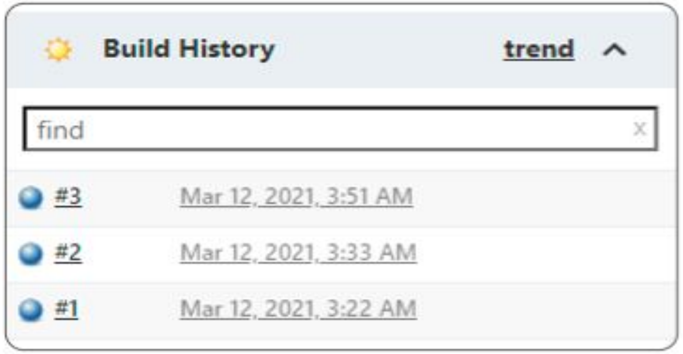
- [✓] Active
  We will deliver event details when this hook is triggered.

Add webhook



Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [✓] GitHub hook trigger for GITScm polling
- [ ] Poll SCM



Build History trend

find

#3	Mar 12, 2021, 3:51 AM
#2	Mar 12, 2021, 3:33 AM
#1	Mar 12, 2021, 3:22 AM

Poll 2 (15 seconds)

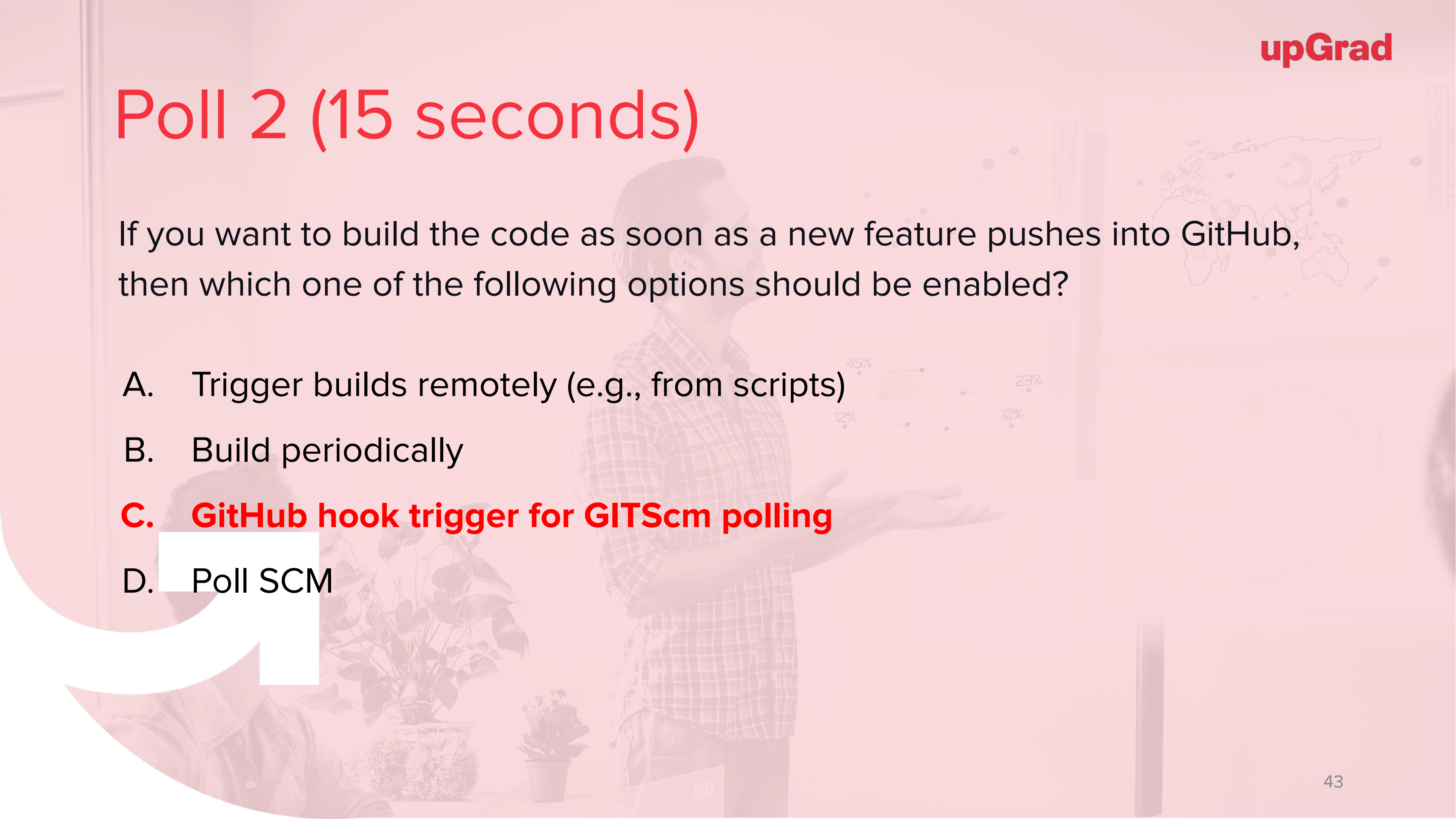
If you want to build the code as soon as a new feature pushes into GitHub, then which one of the following options should be enabled?

A. Trigger builds remotely (e.g., from scripts)

B. Build periodically

C. GitHub hook trigger for GITScm polling

D. Poll SCM



Poll 2 (15 seconds)

If you want to build the code as soon as a new feature pushes into GitHub, then which one of the following options should be enabled?

A. Trigger builds remotely (e.g., from scripts)

B. Build periodically

C. GitHub hook trigger for GITScm polling

D. Poll SCM

Introduction to Jenkins Pipeline

Jenkins Pipeline Introduction

Pipeline:

- It can be specified as code, so you can write pipeline script and maintain version control in the Git repository.

- It will provide continuous release of the application.

- Sequence of stages to perform the given tasks such as pulling code from the Git repository, static code analysis, building project, executing unit test, automated tests, performance tests and deploying application.

Types of Pipeline

- **Declarative**
  - New method
  - Easy to use for beginners
  - Groovy language skill is desirable

- **Scripted**
  - Traditional
  - Based on Groovy Domain Specific Language (DSL)
  - Multiple features - very expressive and flexible tool
  - Difficult to use for beginners
  - Should have working experience on Groovy language

Declarative vs Scripted Pipeline Script Template

upGrad

```
pipeline {
        agent any
        stages {
stage('build code') {
        steps {
              /*write steps */
             }
        }
        stage ('test') {
        steps {
               /*write steps */
             }
        }
      }
}
```

Declarative Pipeline Script

```
node {
        stage ('build code' {
                /*write steps */
            }
        stage ('test') {
              /*write steps */
            }
}
```

Scripted Pipeline Script

Pipeline – contains all the script content
Agent and Node – defines the agent where the pipeline will run
Stages – contains all the stages
Steps – way to execute various jobs

Pipeline Directives

upGrad

- **Environment** – defined as environment variables
- **Input** – prompt for input
- **Options** – configure pipeline-specific options like retry, timeout, etc.
- **Parallel** – list of nested stages to be run in parallel
- **Parameters** – list of parameters to provide when triggering the Pipeline (e.g., string, password)
- **Post** – run at the end of a Pipeline's execution (e.g., add some notification or other end of Pipeline tasks)

Pipeline Directives

upGrad

- **Tools** – defining tools or packages to auto-install and put on the PATH (e.g., maven. Jdk, gradle)
- **Triggers** – determines how pipelines should be triggered (e.g., cron, poll SCM)
- **When** – determine executing stage depending on the given condition

Jenkins Pipeline

Pipeline:

- Defined as a suite of plugins that helps you orchestrate simple or complex automation

Jenkins Pipeline:

- Provides tool for modeling delivery pipelines, “as Code” (Pipeline as Code)
- Implement and integrate CD pipeline.

CI/CD Pipeline:

- Integrates SDLC stages, steps to execute tasks in each stage, trigger jobs for a given order and show pipeline status with logs
- Automation from Continuous build to Continuous Monitoring (build, test, staging, deploy and monitor)

Demo – Jenkins Pipeline CI/CD

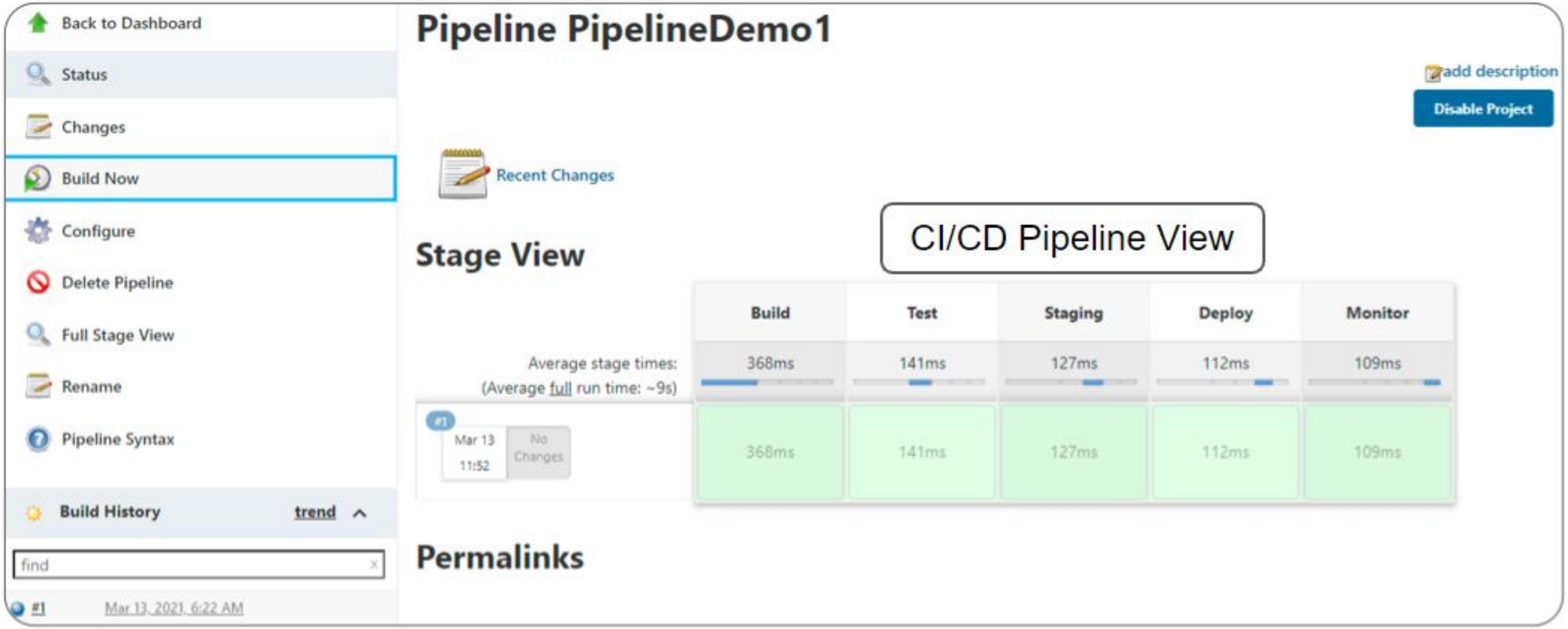
upGrad

Declarative Pipeline Script

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo 'This is job building stage'
            }
        }
 stage('Test') {
            steps {
                echo 'This is Testing stage'
            }
        }
stage('Staging') {
            steps {
                echo 'This is Staging environment'
            }
        }
stage('Deploy') {
            steps {
                echo 'This is Deploying stage'
            }
        }
 stage('Monitor') {
            steps {
                echo 'This is Monitoring stage'
            }
        }
    }
}
```

Demo – Jenkins Pipeline CI/CD

upGrad



Back to Dashboard
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax

Build History trend

find

#1 Mar 13, 2021, 6:22 AM

Pipeline PipelineDemo1

add description

Disable Project

Recent Changes

CI/CD Pipeline View

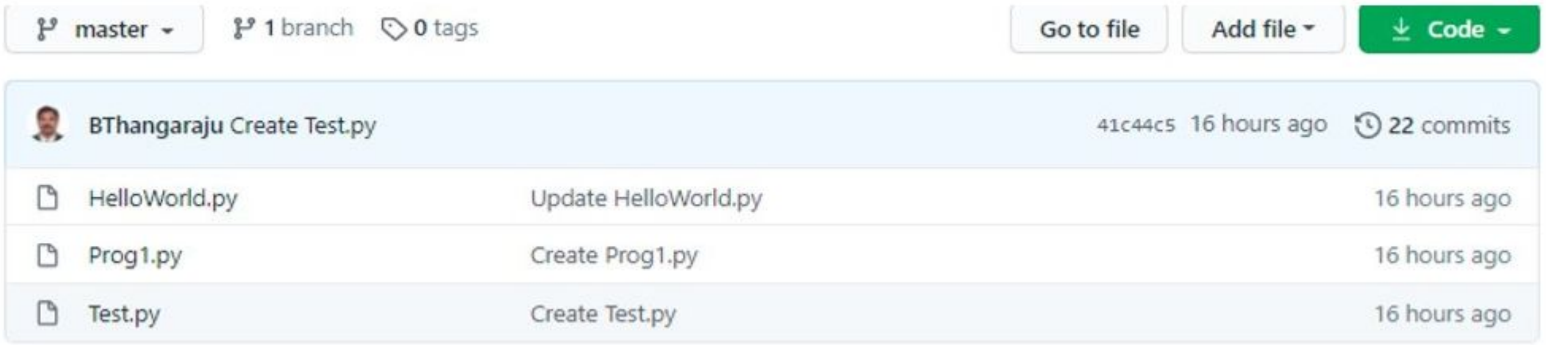
Stage View

	Build	Test	Staging	Deploy	Monitor
Average stage times: (Average full run time: ~9s)	368ms	141ms	127ms	112ms	109ms
#1 Mar 13 11:52 No Changes	368ms	141ms	127ms	112ms	109ms

Permalinks

Demo Programs

- Git Repository



master · 1 branch · 0 tags · Go to file · Add file · Code

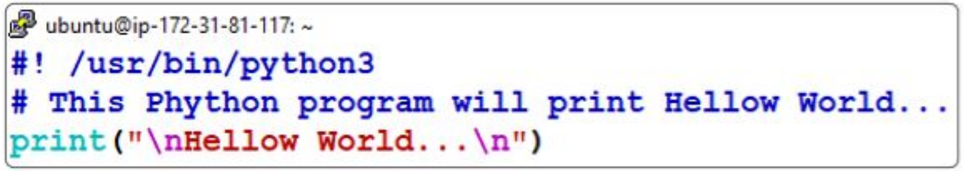
BThangaraju Create Test.py · 41c44c5 16 hours ago · 22 commits

HelloWorld.py	Update HelloWorld.py	16 hours ago
Prog1.py	Create Prog1.py	16 hours ago
Test.py	Create Test.py	16 hours ago

- HelloWorld.py –program to git clone example.
- Prog1.py – for building the code.
- Test.py –for testing the Prog1.py source code.

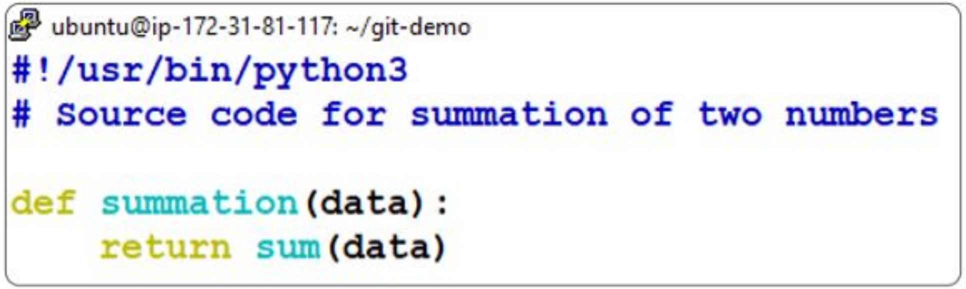
Demo Programs

upGrad


ubuntu@ip-172-31-81-117: ~

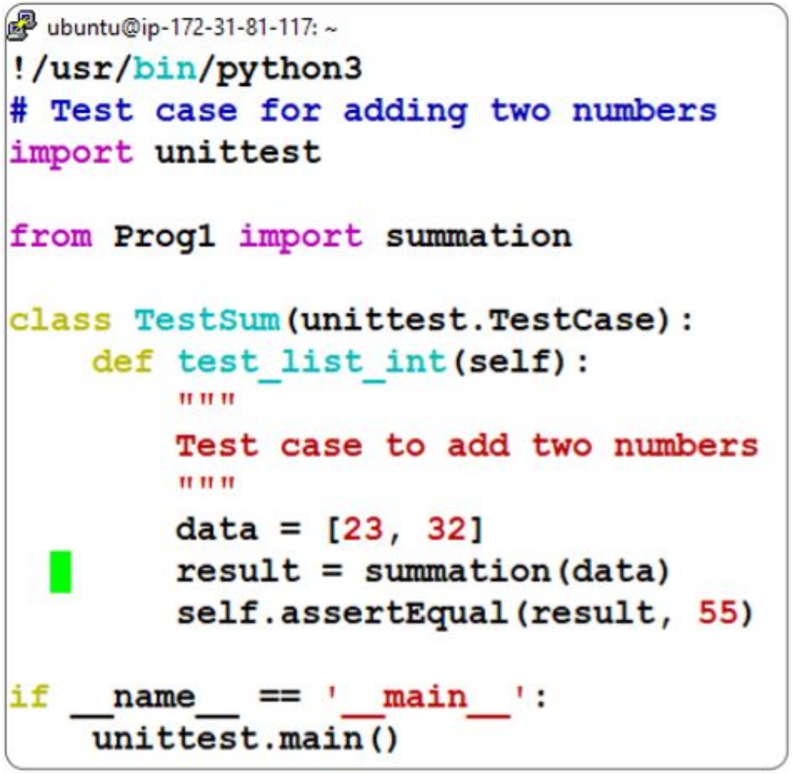
```
#! /usr/bin/python3
# This Phython program will print Hellow World...
print("\nHellow World...\n")
```

HelloWorld.py


ubuntu@ip-172-31-81-117: ~/git-demo

```
#!/usr/bin/python3
# Source code for summation of two numbers

def summation(data):
    return sum(data)
```

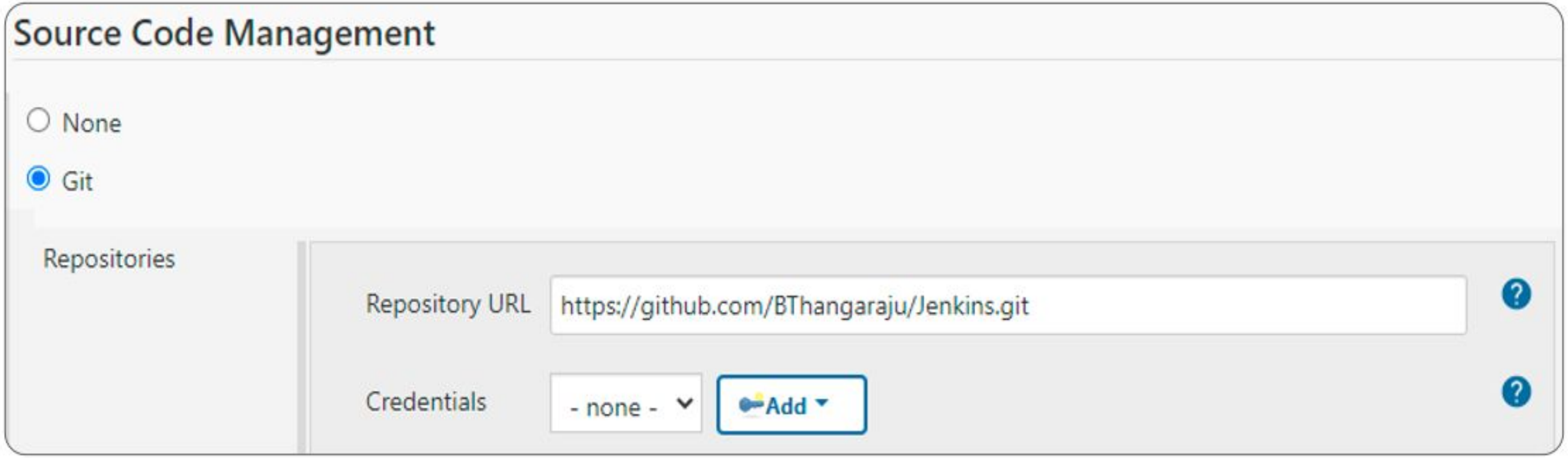
Prog1.py


ubuntu@ip-172-31-81-117: ~

```
!/usr/bin/python3
# Test case for adding two numbers
import unittest

from Prog1 import summation

class TestSum(unittest.TestCase):
    def test_list_int(self):
        """
        Test case to add two numbers
        """
        data = [23, 32]
        result = summation(data)
        self.assertEqual(result, 55)

if __name__ == '__main__':
    unittest.main()
```

Test.py

Integration With Git



Source Code Management

( ) None

(●) Git

Repositories

Repository URL: https://github.com/BThangaraju/Jenkins.git

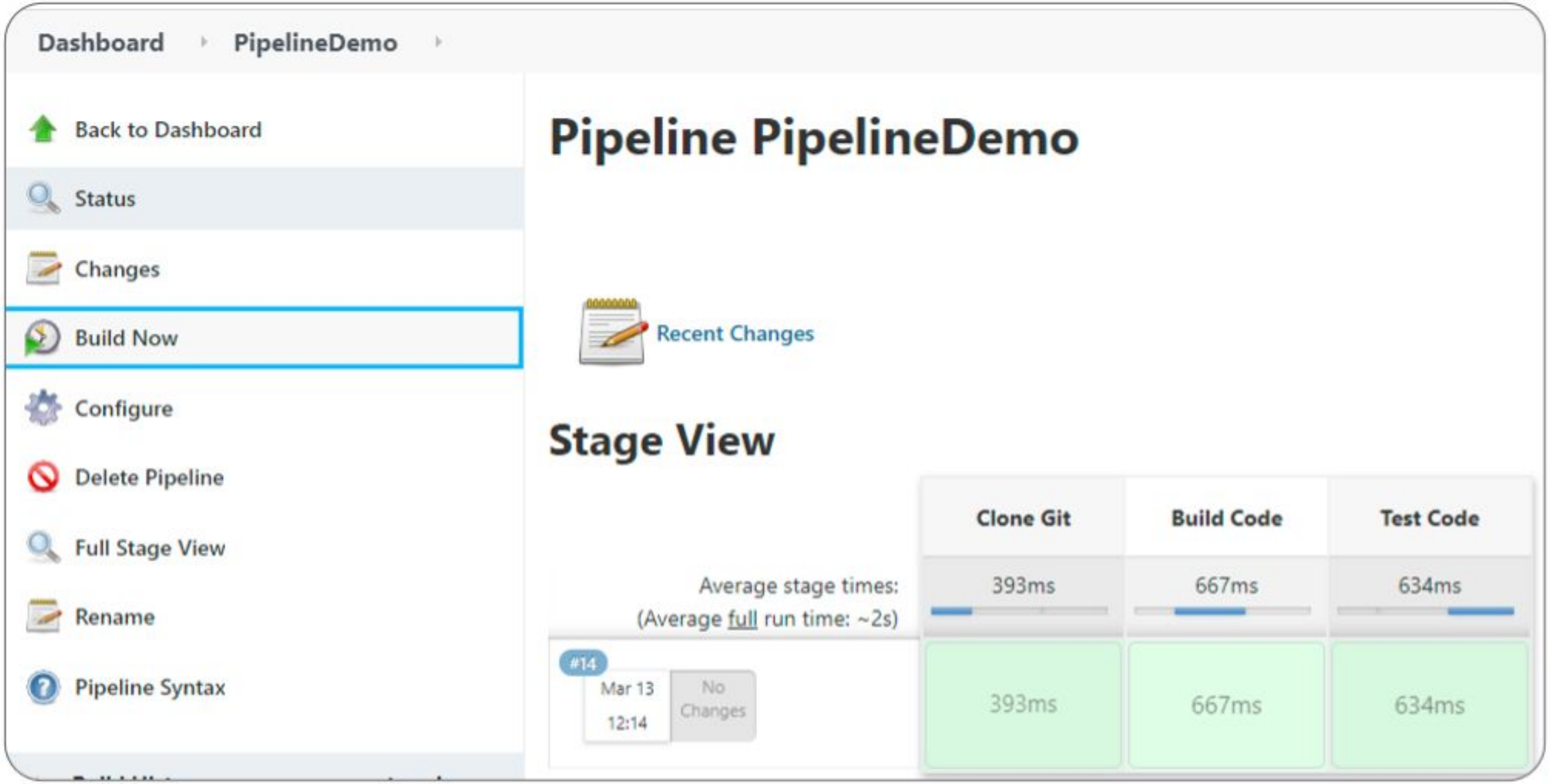
Credentials: - none -

Git Clone, Build and Test a Python Script

```
pipeline {
agent any
    stages {
        stage('Clone Git') {
            steps {
                git 'https://github.com/BThangaraju/Jenkins.git'
            }
        }
        stage('Build Code') {
            steps {
                sh "chmod u+x Prog1.py"
                sh "./Prog1.py"
             }
          }
stage('Test Code') {
            steps {
                sh "chmod u+x Test.py"
                sh "./Test.py"
            }
        }
    }
}
```

Pipeline View

upGrad



Dashboard › PipelineDemo ›

Pipeline PipelineDemo

Recent Changes

Stage View

	Clone Git	Build Code	Test Code
Average stage times: (Average full run time: ~2s)	393ms	667ms	634ms
#14 Mar 13 12:14 No Changes	393ms	667ms	634ms

Console Output

```
Started by user admin
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/PipelineDemo
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Clone Git)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/BThangaraju/Jenkins.git # timeout=10
Fetching upstream changes from https://github.com/BThangaraju/Jenkins.git
 > git --version # timeout=10
 > git --version # 'git version 2.17.1'
 > git fetch --tags --progress -- https://github.com/BThangaraju/Jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 41c44c51691fc2f9ca2697c48cbbaad6d5760f2f (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 41c44c51691fc2f9ca2697c48cbbaad6d5760f2f # timeout=10
 > git branch -a -v --no-abbrev # timeout=10
 > git branch -D master # timeout=10
 > git checkout -b master 41c44c51691fc2f9ca2697c48cbbaad6d5760f2f # timeout=10
Commit message: "Create Test.py"
```

Console Output

upGrad

```
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build Code)
[Pipeline] sh
+ chmod u+x Prog1.py
[Pipeline] sh
+ ./Prog1.py
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test Code)
[Pipeline] sh
+ chmod u+x Test.py
[Pipeline] sh
+ ./Test.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Jenkins Distributed Architecture

Scaling Jenkins

- Initially, when you start working with Jenkins, you have a single server to carry out all the tasks. The single Jenkins server is also called as Master node or Jenkins controller.

- Single Jenkins server is not enough to meet certain requirements like:
  - When you configure more jobs
  - When you orchestrate more frequent builds
  - When more developers depend on one controller
  - When you add incremental features in large and complex projects frequently
  - When you need different environments (diff OS) to test the build

Scaling Jenkins

upGrad

- Instead of adding new team members or new projects to an existing single Jenkins controller, you can create additional Jenkins controllers to accommodate new teams or projects.

- The Jenkins distributed architecture enables us to use various environments for each build project, dividing the workload across multiple agents running jobs concurrently.

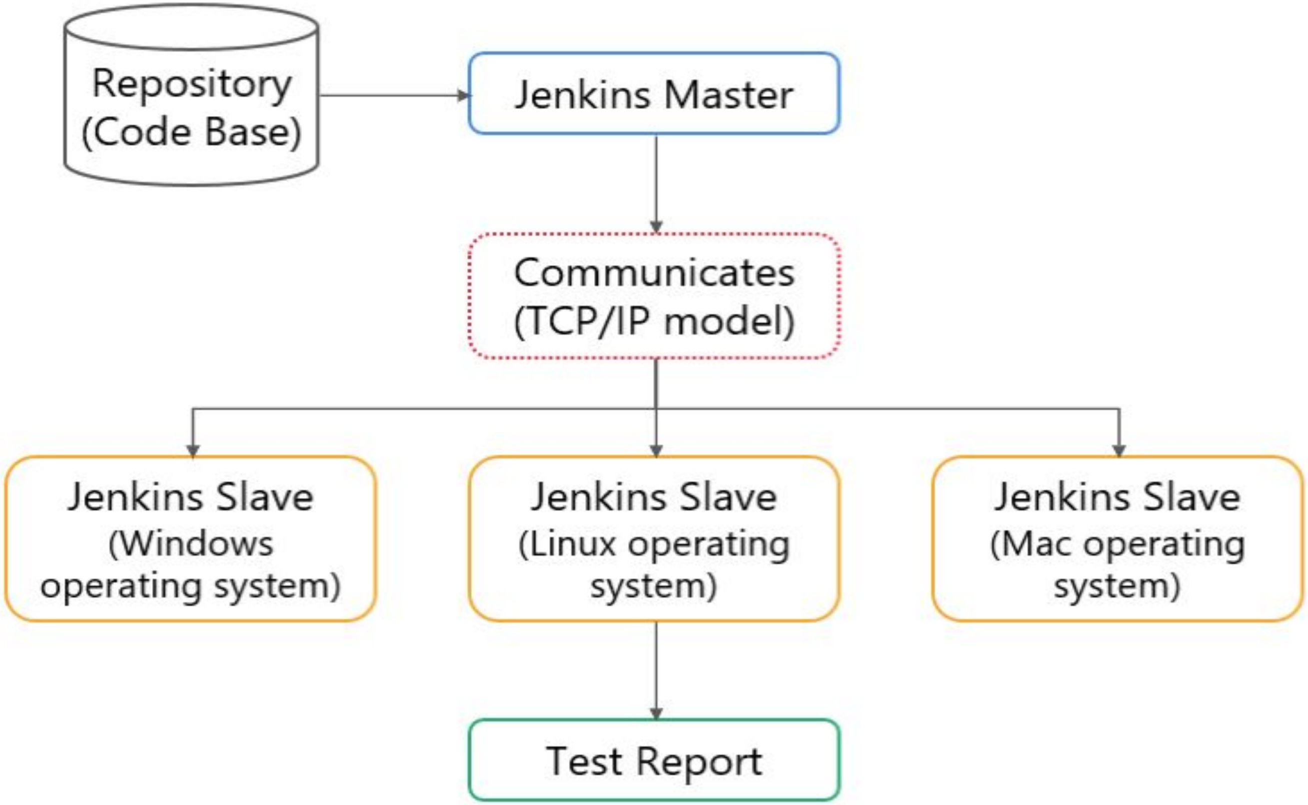
- Jenkins' distributed architecture is based on the idea of ‘Master + Agent’. The master is responsible for coordination and providing the GUI and API endpoints, while the Agents perform the work.

- The Jenkins master manages the Jenkins agents and orchestrates their work by scheduling jobs on agents and monitoring them.

- Agents can link to the Jenkins controller via local or cloud computers.

Jenkins Distributed Architecture

upGrad


Repository (Code Base)
Jenkins Master
Communicates (TCP/IP model)
Jenkins Slave (Windows operating system)
Jenkins Slave (Linux operating system)
Jenkins Slave (Mac operating system)
Test Report

Host Setup

upGrad

- sudo su - jenkins
- sudo apt-get install docker.io
- systemctl status docker.service
- sudo docker pull ubuntu
- sudo docker run –it –name Jenkins_Agent ubuntu /bin/bash
- ssh-keygen
- ls .ssh/; **id_rsa id_rsa.pub** - two files were created.
- cat id_rsa; copy and paste in Jenkins Global Credentials
- ssh-copy-id jenkins@172.17.0.2

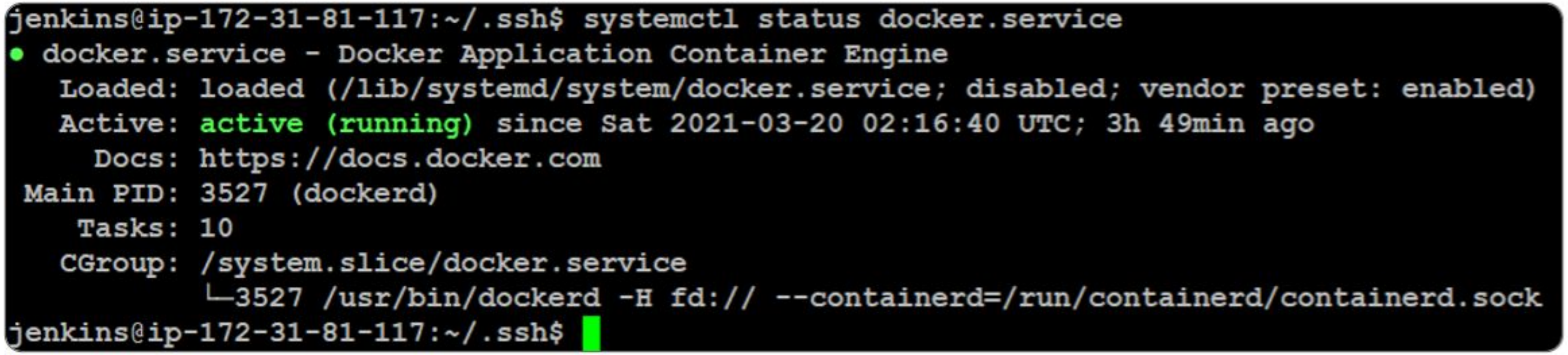
Docker Setup

upGrad

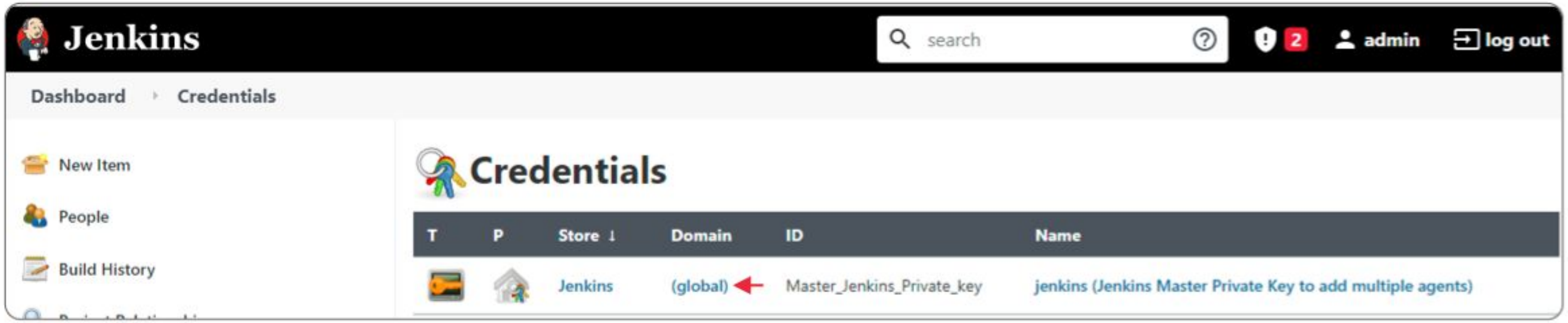
1. adduser jenkins
2. usermod -aG sudo jenkins
3. apt-get install sudo
4. su - jenkins
5. sudo apt-get update
6. sudo apt-get install openssh-server
8. sudo service ssh restart
9. service ssh status
10. sudo apt install openjdk-11-jdk
11. java --version

Check the docker.service in Host

upGrad



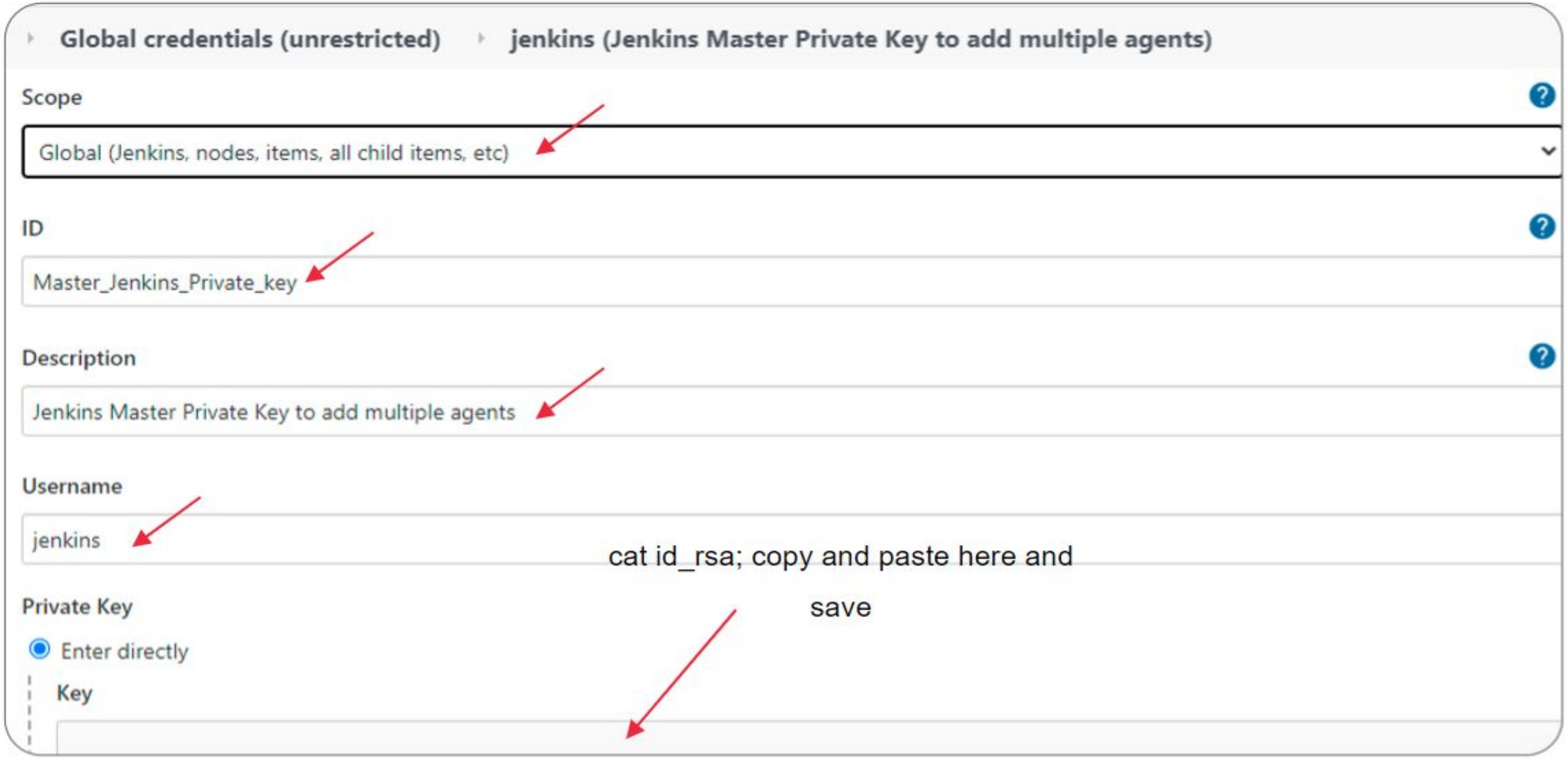
```
jenkins@ip-172-31-81-117:~/.ssh$ systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; disabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-03-20 02:16:40 UTC; 3h 49min ago
     Docs: https://docs.docker.com
 Main PID: 3527 (dockerd)
    Tasks: 10
   CGroup: /system.slice/docker.service
           └─3527 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
jenkins@ip-172-31-81-117:~/.ssh$ 
```



Credentials

T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	Master_Jenkins_Private_key	jenkins (Jenkins Master Private Key to add multiple agents)

Configure Jenkins Global Credentials



Global credentials (unrestricted) › jenkins (Jenkins Master Private Key to add multiple agents)

Scope

Global (Jenkins, nodes, items, all child items, etc)

ID

Master_Jenkins_Private_key

Description

Jenkins Master Private Key to add multiple agents

Username

jenkins

Private Key

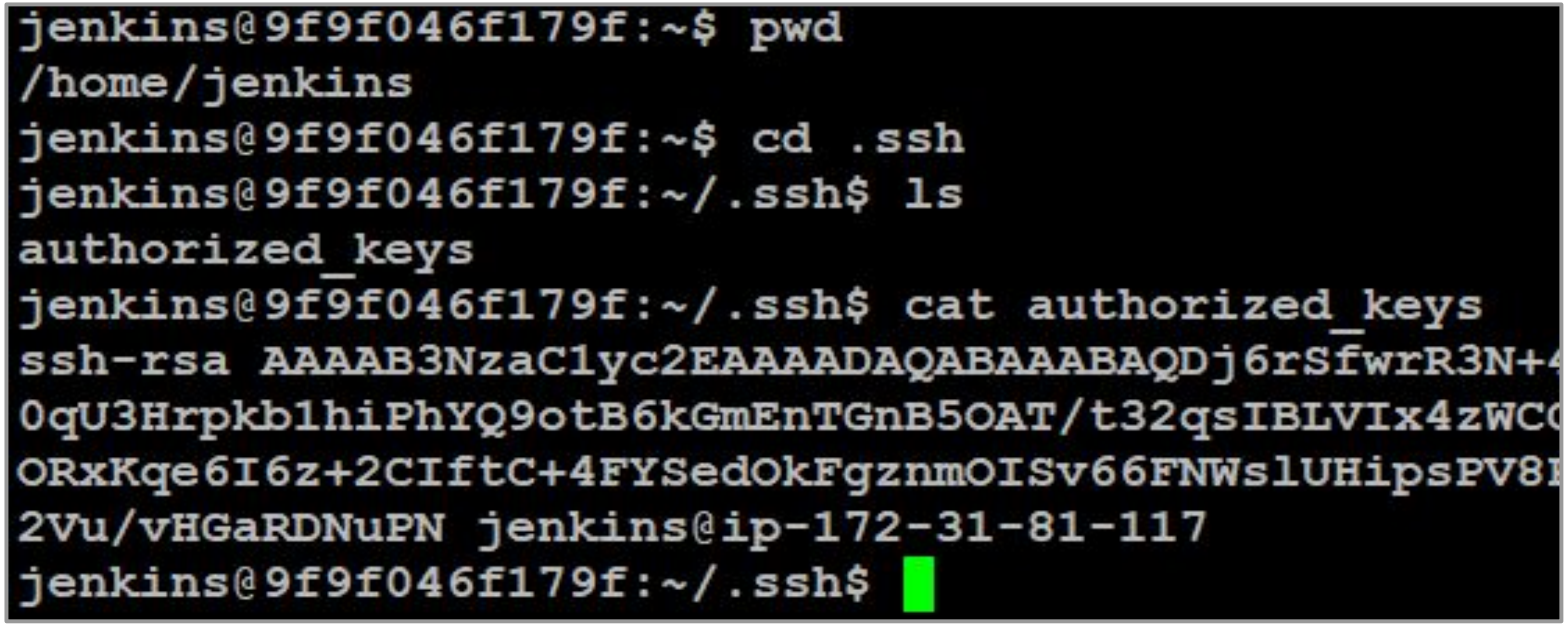
(●) Enter directly

Key

cat id_rsa; copy and paste here and save

Copy Authorized Keys From Host to Container

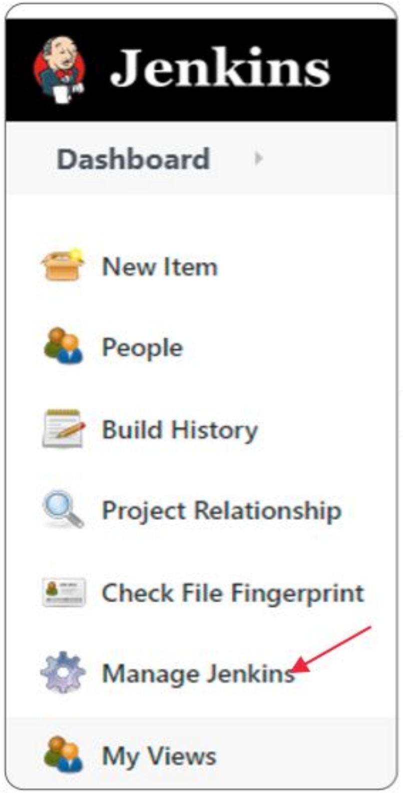
- In the Host: ssh-copy-id jenkins@172.17.0.2
- Check the copied authorized_keys in the container

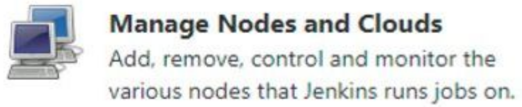


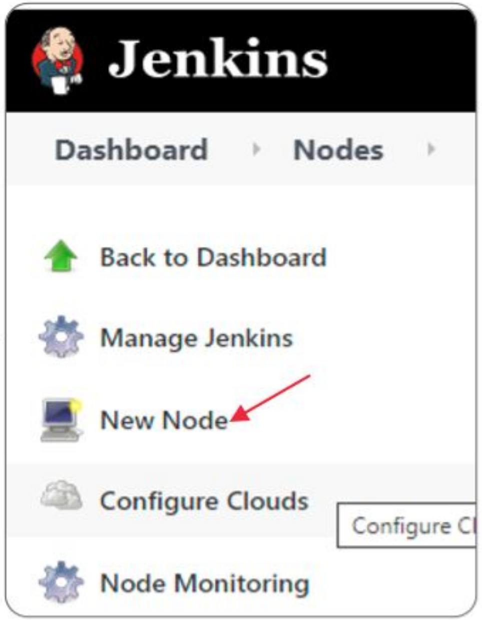
```
jenkins@9f9f046f179f:~$ pwd
/home/jenkins
jenkins@9f9f046f179f:~$ cd .ssh
jenkins@9f9f046f179f:~/.ssh$ ls
authorized_keys
jenkins@9f9f046f179f:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDj6rSfwrR3N+
0qU3Hrpkb1hiPhYQ9otB6kGmEnTGnB5OAT/t32qsIBLVIx4zWC
ORxKqe6I6z+2CIftC+4FYSedOkFgznmOISv66FNWs1UHipsPV8
2Vu/vHGaRDNuPN jenkins@ip-172-31-81-117
jenkins@9f9f046f179f:~/.ssh$
```

Configure Agent in Jenkins

upGrad

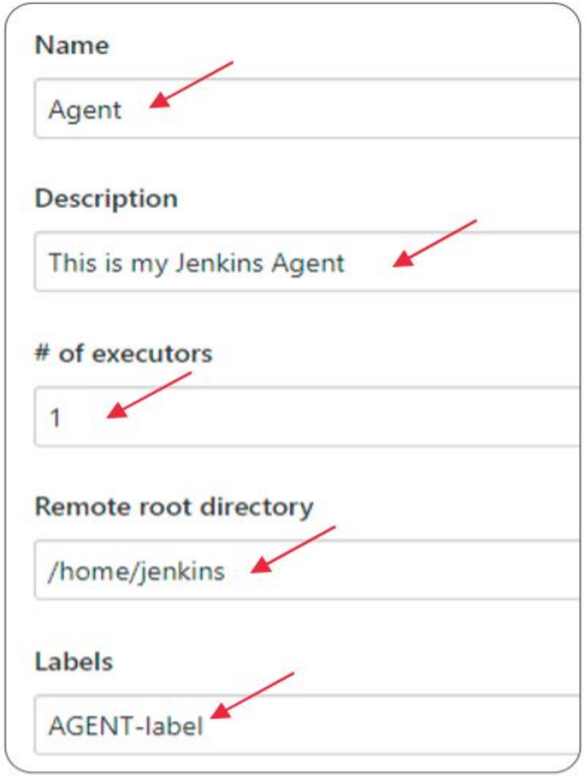



Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.



Configure Agent in Jenkins

upGrad



Name

Agent

Description

This is my Jenkins Agent

# of executors

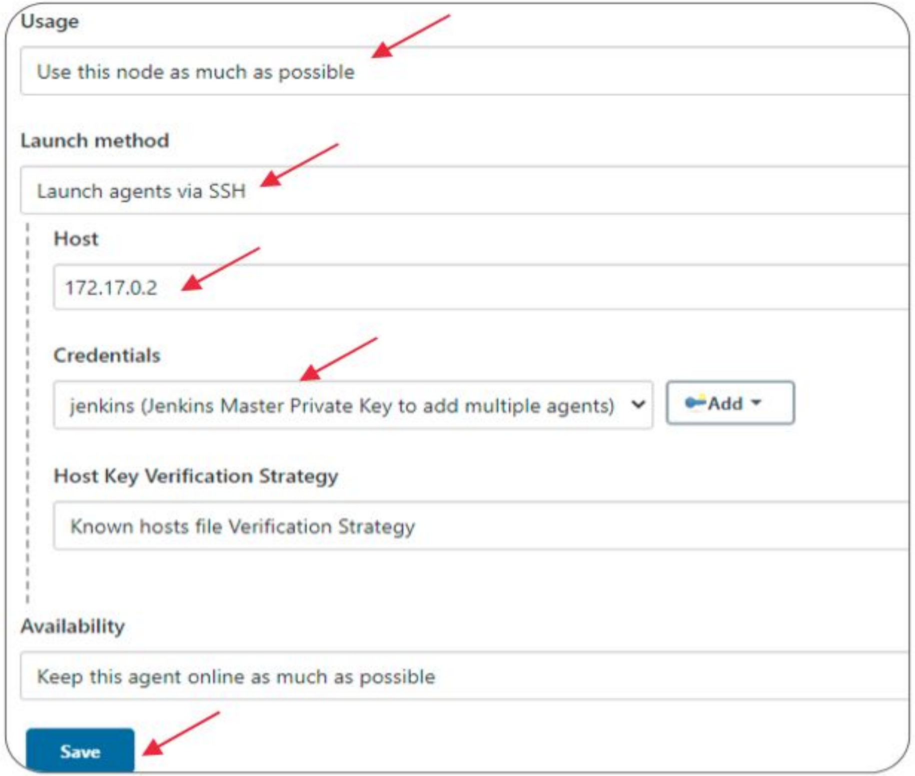
1

Remote root directory

/home/jenkins

Labels

AGENT-label



Usage

Use this node as much as possible

Launch method

Launch agents via SSH

Host

172.17.0.2

Credentials

jenkins (Jenkins Master Private Key to add multiple agents)

Host Key Verification Strategy

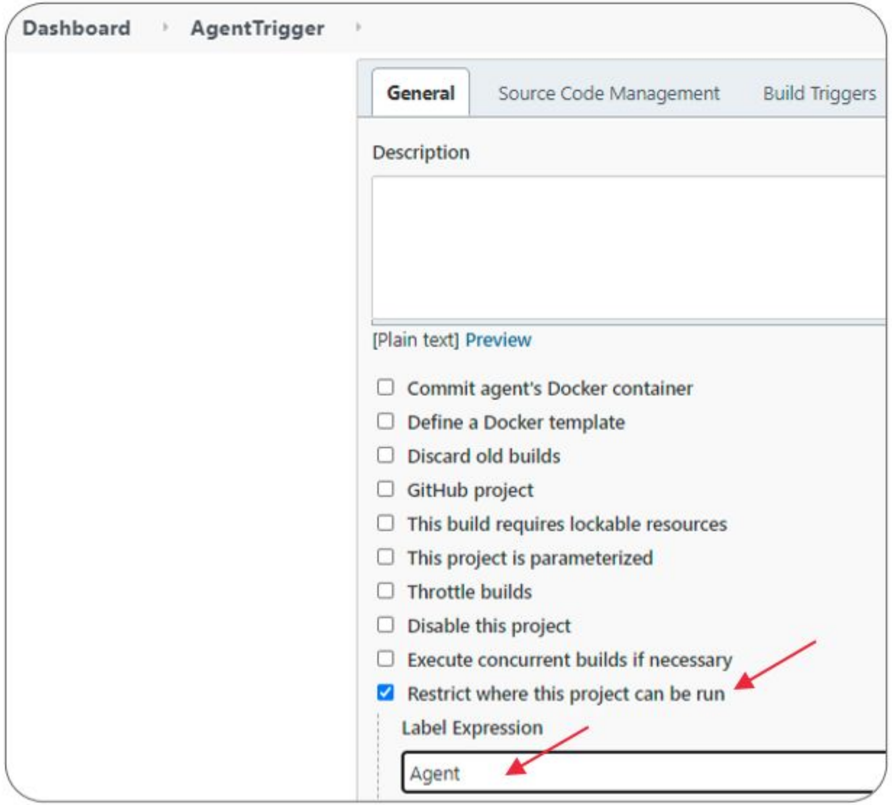
Known hosts file Verification Strategy

Availability

Keep this agent online as much as possible

Create a New Project as AgentTrigger

upGrad


Dashboard › AgentTrigger ›

General | Source Code Management | Build Triggers

Description

[Plain text] Preview

- [ ] Commit agent's Docker container
- [ ] Define a Docker template
- [ ] Discard old builds
- [ ] GitHub project
- [ ] This build requires lockable resources
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Disable this project
- [ ] Execute concurrent builds if necessary
- [✓] Restrict where this project can be run

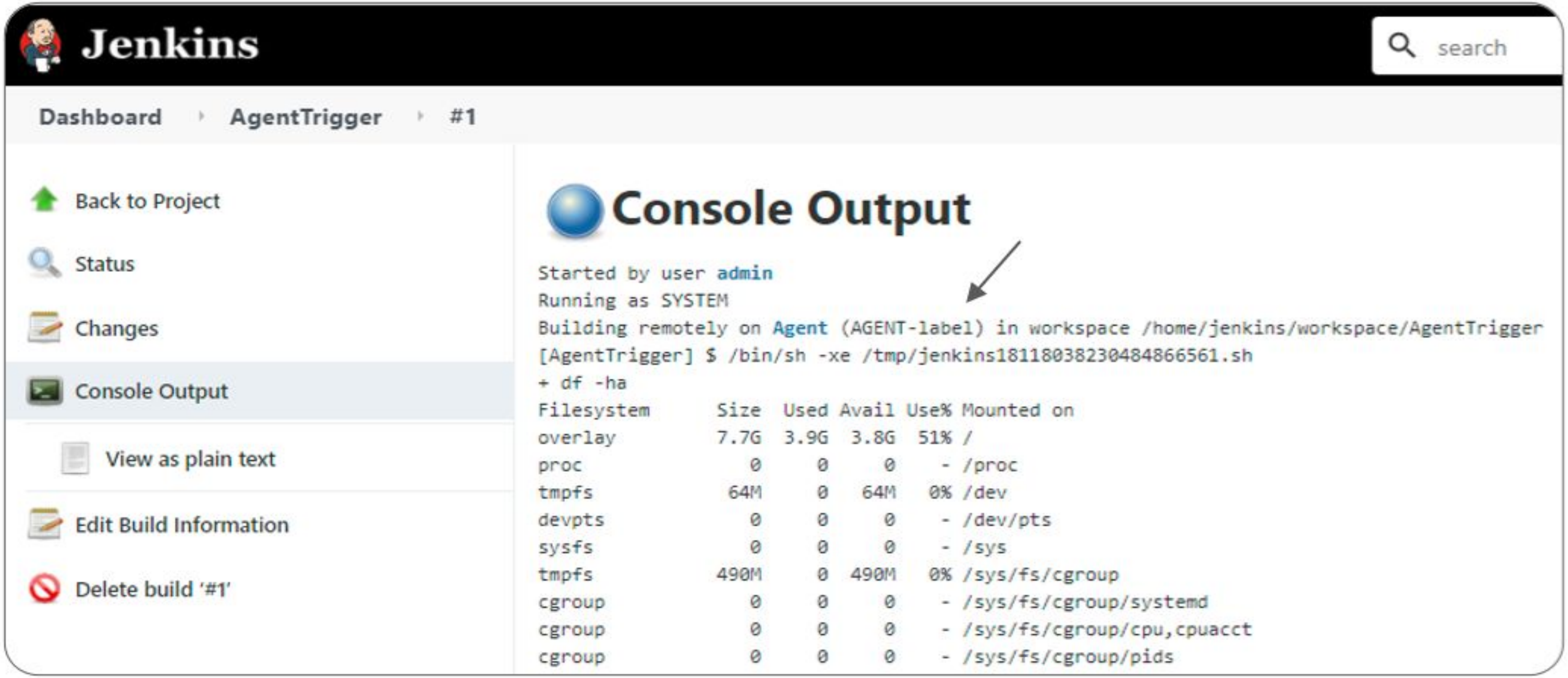
Label Expression

Agent

- Add build script as: df –ha
- Save and build manually

Console Output

upGrad



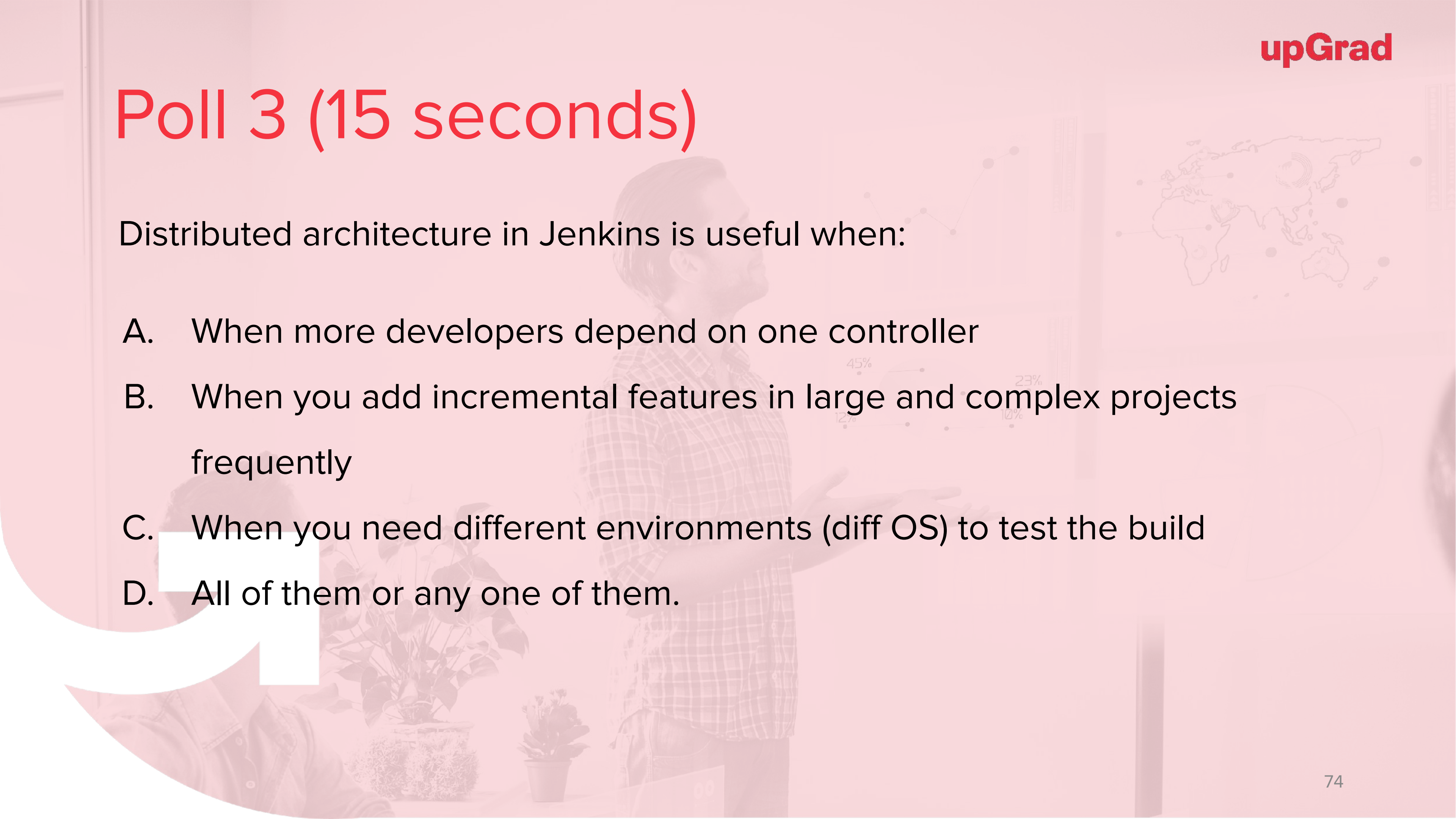
Console Output

```
Started by user admin
Running as SYSTEM
Building remotely on Agent (AGENT-label) in workspace /home/jenkins/workspace/AgentTrigger
[AgentTrigger] $ /bin/sh -xe /tmp/jenkins1811803823048486561.sh
+ df -ha
Filesystem      Size  Used Avail Use% Mounted on
overlay         7.7G  3.9G  3.8G  51% /
proc               0     0     0    - /proc
tmpfs            64M     0   64M   0% /dev
devpts             0     0     0    - /dev/pts
sysfs              0     0     0    - /sys
tmpfs           490M     0  490M   0% /sys/fs/cgroup
cgroup             0     0     0    - /sys/fs/cgroup/systemd
cgroup             0     0     0    - /sys/fs/cgroup/cpu,cpuacct
cgroup             0     0     0    - /sys/fs/cgroup/pids
```

Benefits of Jenkins Distributed Architecture

upGrad

- Higher performance
- High availability
- Failover mechanism
- Enhanced security
- Rollback mechanism from machine failure



Poll 3 (15 seconds)

Distributed architecture in Jenkins is useful when:

A. When more developers depend on one controller

B. When you add incremental features in large and complex projects frequently

C. When you need different environments (diff OS) to test the build

D. All of them or any one of them.

upGrad

Poll 3 (15 seconds)

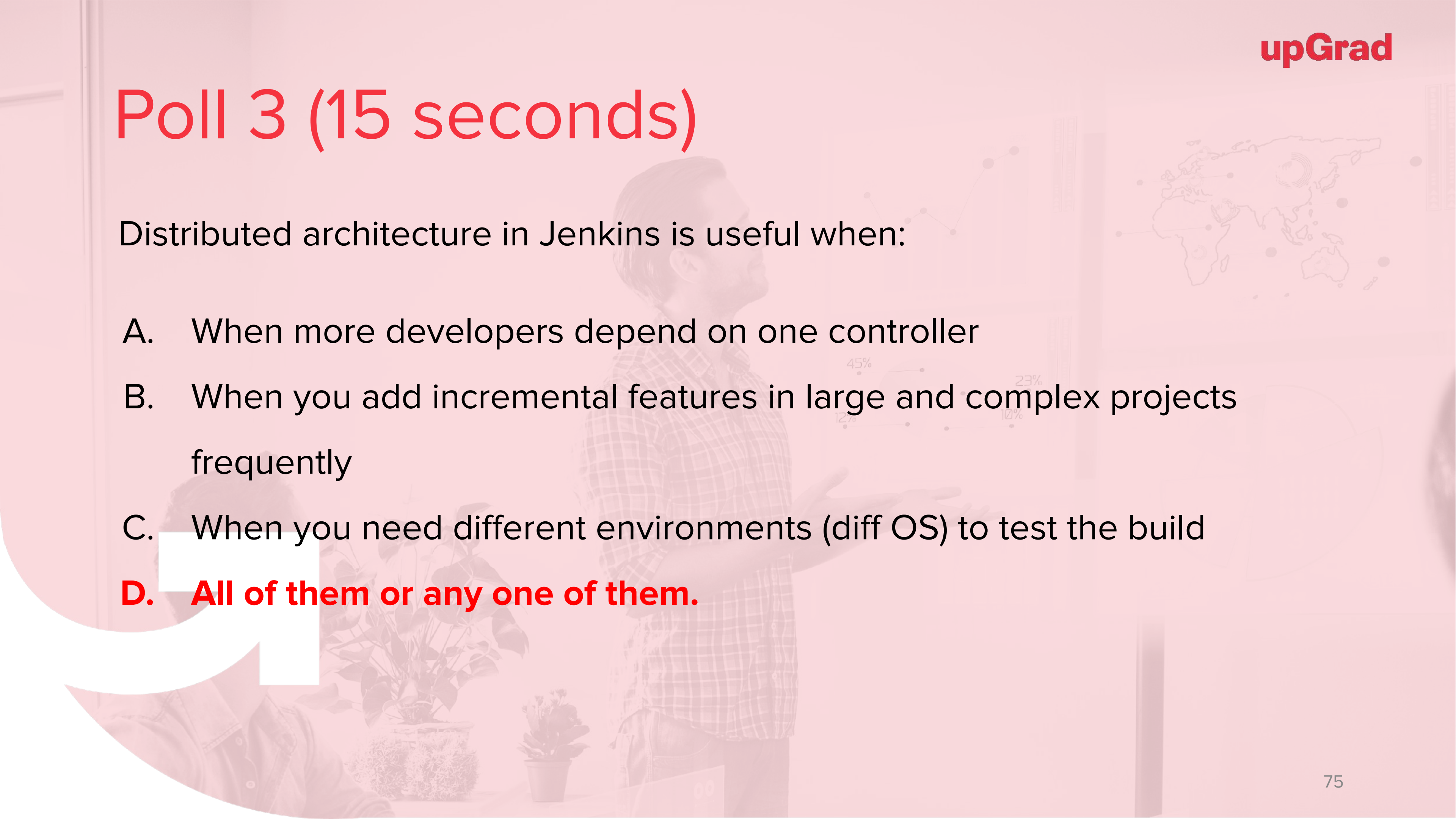
Distributed architecture in Jenkins is useful when:

A. When more developers depend on one controller

B. When you add incremental features in large and complex projects frequently

C. When you need different environments (diff OS) to test the build

D. All of them or any one of them.



upGrad
#RahoAmbitious

Thank You!