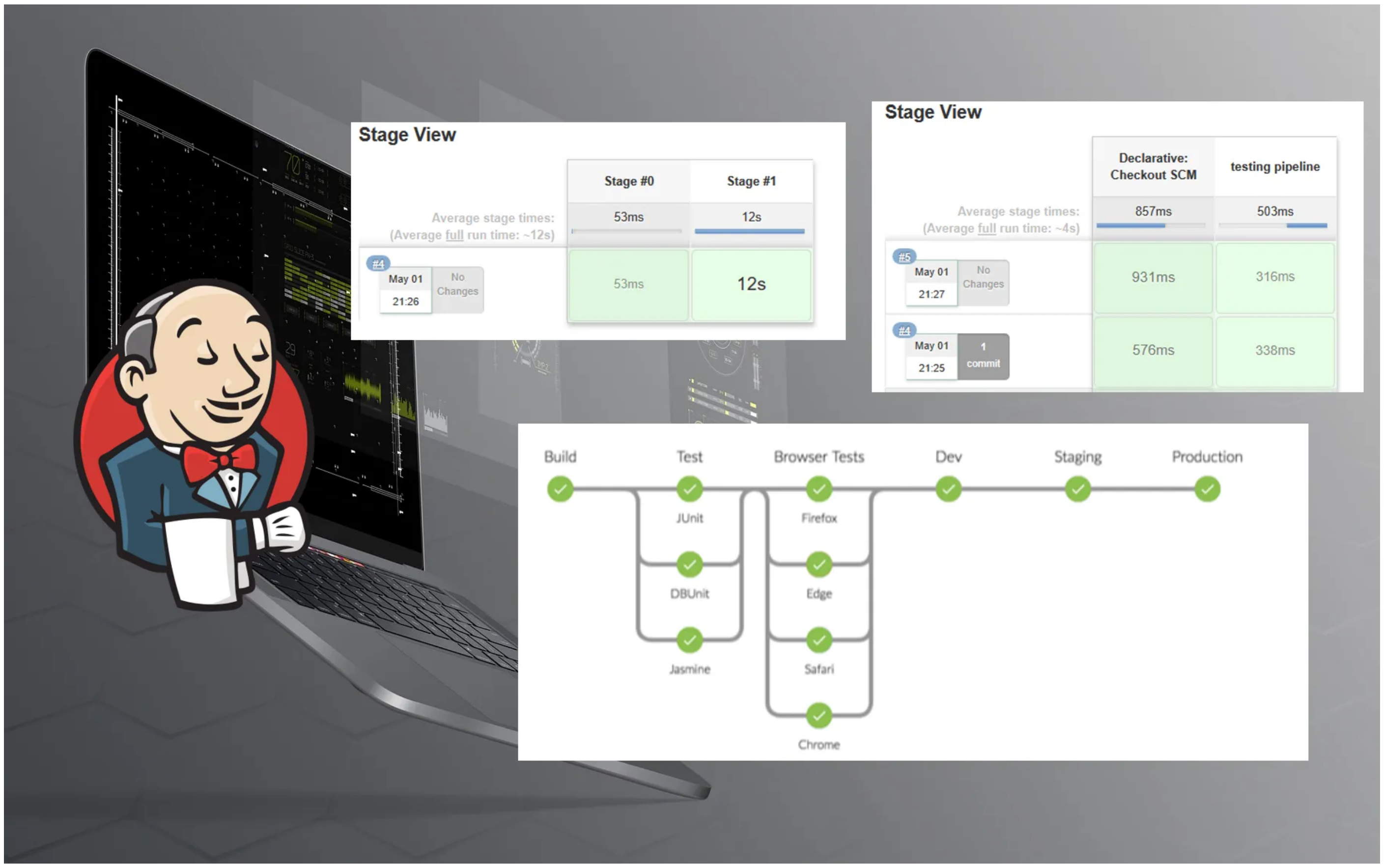
Digital Varys



Continuous Delivery with Jenkins Pipeline.

Leave a Comment / Build and Package, DevOps, Release and Operation / By DigitalVarys

Table of Contents

About Jenkins

> *Jenkins is an Open source, Java-based automation tool. This tool automates the Software Integration and delivery process called Continuous Integration and Continuous Delivery. Jenkins support various popular Version control system, Software build, and delivery tools. Over the Years, Jenkins become giant in CICD process, especially new features like Jenkins Pipelines (Scripted and Declarative Pipeline) makes the delivery process very easy and help Team to adopt DevOps easily.*

Jenkins pipeline

Jenkins Pipeline is a stack of Jenkins plugins and other tools which helps implementing and continuous integration and delivery pipelines. In Jenkins, Pipelines are written in DSL code which implements this continuous integration and delivery pipeline jobs.

Get the pipeline plugin from the Jenkins plugin market place and install into the Jenkins instance. For the same, Go to Manage Jenkins > Manage Plugins. > Available tab, search for "**build pipeline**".

This Jenkins pipeline can be built using Web UI or Scripted Jenkinsfile. This Jenkinsfile is written with Groovy DSL and updated in configuration page on the Jenkins job. Overall, Jenkinsfile can also be called pipeline as code.

Types of Jenkins pipeline

There are two type of Jenkins pipeline based on which format the Jenkinsfile is written.

1. Declarative pipeline
2. Scripted pipeline

Declarative Pipeline

The Declarative pipeline is a new feature that is added to create the pipeline. This is basically written in a Jenkinsfile which can be stored into a source code management system such as Git. Declarative pipelines is an ideal solution for the simple continuous delivery pipeline as it has very limited and pre-defined structure.

Lets see the structure and syntax of the Declarative pipeline.

The Agent is where the whole pipeline runs. Example, Docker. The Agent has following parameters:

- any – Which mean the whole pipeline will run on any available agent.
- none – Which mean all the stages under the block will have to declared with agent separately.
- label – this is just a label for the Jenkins environment
- docker – this is to run the pipeline in Docker environment.

The Declarative pipeline code will looks like this:

```
pipeline {
  agent { label 'node-1' }
  stages {
    stage('Source') {
      steps {
        git 'https://github.com/digitalvarys/jenkins-tutorials.git''
      }
    }
    stage('Compile') {
      tools {
        gradle 'gradle4'
      }
      steps {
        sh 'gradle clean compileJava test'
      }
    }
  }
}
```

Scripted Pipeline

The scripted pipeline is a traditional way of writing the Jenkins pipeline as code. Ideally, Scripted pipeline is written in Jenkins file on web UI of Jenkins. Unlike Declarative pipeline, the scripted pipeline strictly uses groovy based syntax. Since this, The scripted pipeline provides huge control over the script and can manipulate the flow of script extensively. This helps developers to develop advance and complex pipeline as code.

Let us see the structure and syntax of the scripted pipeline

Node Block:

Node is the part of the Jenkins architecture where Node or agent node will run the part of the workload of the jobs and master node will handle the configuration of the job. So this will be defined in the first place as

```
node {
}
```

Stage Block:

Stage block can be a single stage or multiple as the task goes. And it may have common stages like

- Cloning the code from SCM
- Building the project
- Running the Unit Test cases
- Deploying the code
- Other functional and performance tests.

So the stages can be written as mentioned below:

```
stage {
}
```

So, Together, the scripted pipeline can be written as mentioned below.

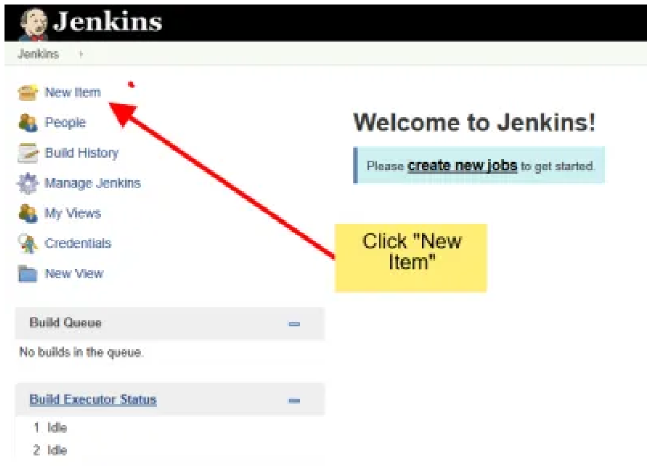
```
stage {
}
So, Together, the scripted pipeline can be written as mentioned below.
node ('node-1') {
  stage('Source') {
    git 'https://github.com/digitalvarys/jenkins-tutorials.git''
  }
  stage('Compile') {
    def gradle_home = tool 'gradle4'
    sh "'${gradle_home}/bin/gradle' clean compileJava test"
  }
}
```

Run the pipeline

Now lets configure the Jenkins pipeline. Following are the steps to create Jenkins Pipeline.

Step 1:

Select "New Item" from the dashboard of the Jenkins instance



Welcome to Jenkins!

Please create new jobs to get started.

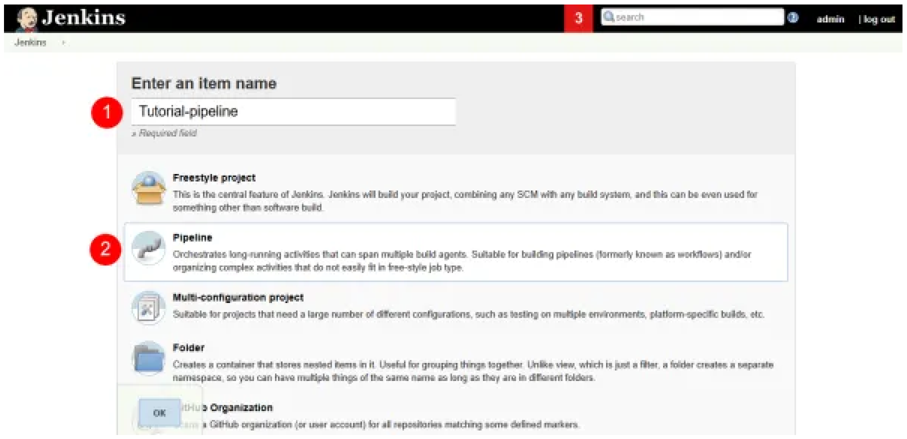
No builds in the queue.

1 Idle

2 Idle

Step 2:

Name the item and select pipeline as the style of the job item.

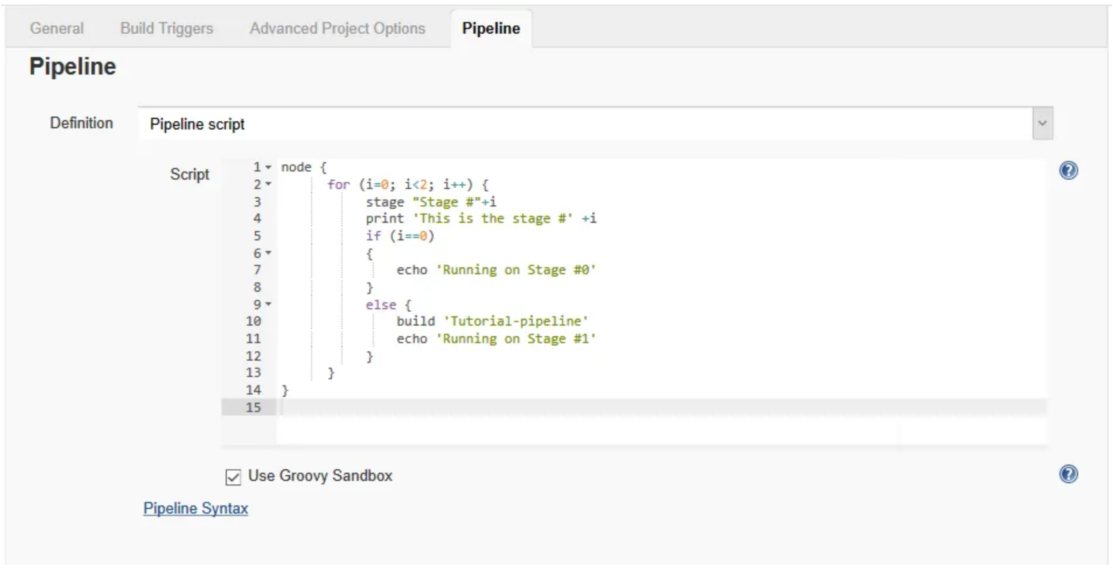

Jenkins
3
search
admin | log out
Jenkins
Enter an item name
1
Tutorial-pipeline
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
2
Pipeline
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.
OK
GitHub Organization
a GitHub organization (or user account) for all repositories matching some defined markers.

Step 3:

Scroll down and select the definition of the Pipeline. Here we get two options.

1. One is the Pipeline script from SCM which is for Declarative Pipeline
2. Another is the Pipeline script which is for Scripted Pipeline that we will write directly on UI.

If it is the Pipeline Script, which will be written directly on UI, Need to write it on Script place (Scripted pipeline)

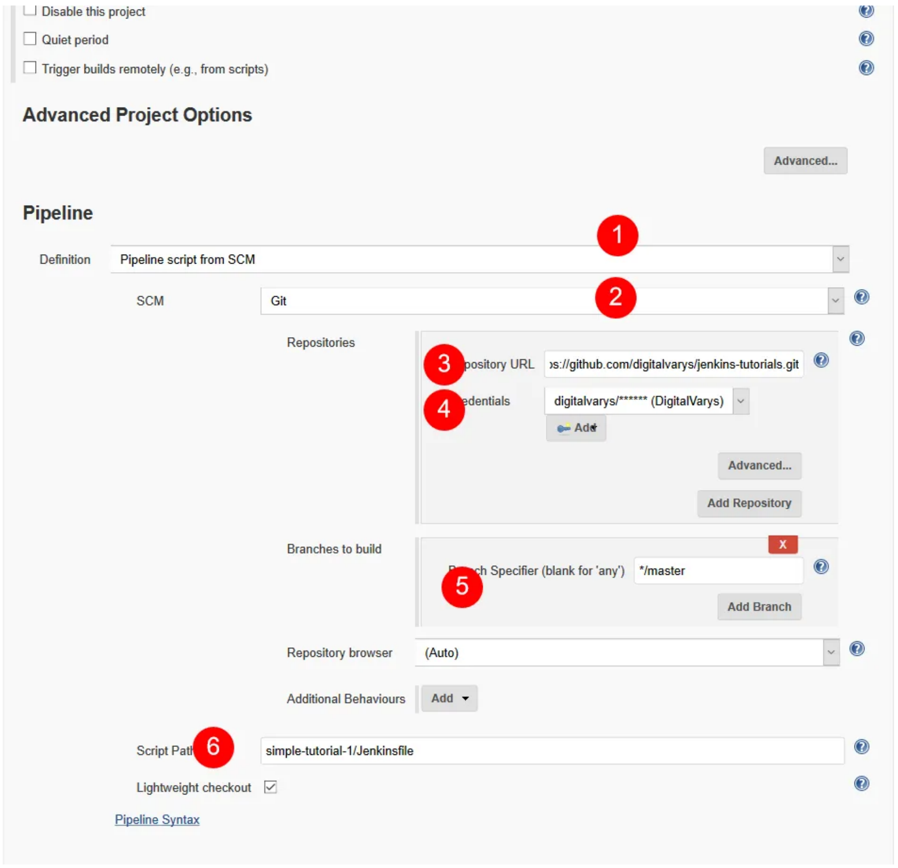


Definition: Pipeline script

Script:

```
node {
    for (i=0; i<2; i++) {
        stage "Stage #"+i
        print 'This is the stage #' +i
        if (i==0)
        {
            echo 'Running on Stage #0'
        }
        else {
            build 'Tutorial-pipeline'
            echo 'Running on Stage #1'
        }
    }
}
```

If we are selecting the Pipeline script from SCM definition, then we need to select the Git > SCM Repository URL > Credentials. (Declarative Pipeline).



☐ Disable this project
☐ Quiet period
☐ Trigger builds remotely (e.g., from scripts)

Advanced Project Options

Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories
pository URL: ɔs://github.com/digitalvarys/jenkins-tutorials.git
edentials: digitalvarys/****** (DigitalVarys)

Branches to build
ch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

Script Path: simple-tutorial-1/Jenkinsfile

Lightweight checkout ☑

Declarative Pipeline from SCM

In the Declarative pipeline, We need to select the repository of the Jenkinsfile, Credentials, Branch and the path of the Jenkinsfile as shown in the above screenshot.

Now let us see how these jobs are executed.

Scripted Pipeline Execution

For testing, we are going to use the following code for the pipeline.

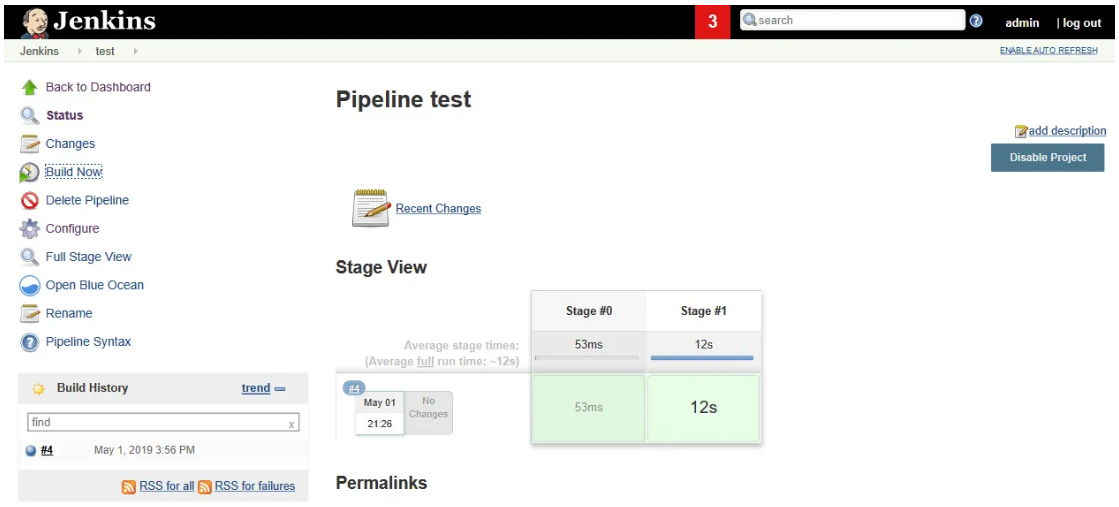
```
node {
      for (i=0; i<2; i++) {
           stage "Stage #"+i
           print 'This is the stage #' +i
           if (i==0)
           {
               echo 'Running on Stage #0'
           }
```

```
                else {
                    build 'Tutorial-pipeline'
                    echo 'Running on Stage #1'
                }
            }
    }
```

Basically,

- This Script will run two stages as we mentioned in the “For loop”.
- Next, the Condition will execute the stages mentioned in the block when i’ = 0.
- Next condition will execute another job from the Jenkins when ‘i’ not equal to 0.

So once we run the job we will get output in the dashboard (As mentioned in the screenshot)



Scripted Pipeline Execution Result

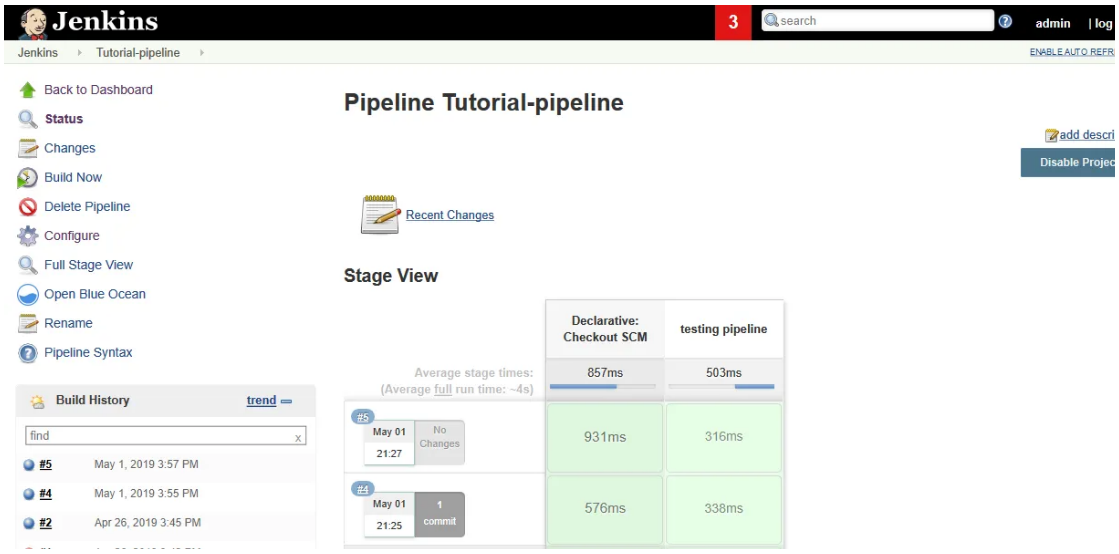
Declarative Pipeline Execution

Let us take the Jenkinsfile from the SCM. In our case, we will take this from the Github. The script inside the Jenkinsfile will look like this

```
pipeline {
          agent any
                  stages {
```

```
                stage('One') {
                        steps {
                                echo 'I am executing stage 1'
                        }
                }
                stage('Two') {
                        steps {
                                input('In pipeline, We take decision. Can we proce
                        }
                }
                stage('Three') {
                        steps {
                                echo "Running Stage Three"
                        }
                }
                stage('Four') {
                        steps {
                        echo "Running another test job"
                        }
                }
        }
}
```

So basically, this script is having 4 stages. And each stage is running in any defined agent. For the Demo purpose, I made it so simple. Once we execute this, we can see the result in the dashboard as shown in the screenshot.



Pipeline Tutorial-pipeline

Stage View

	Declarative: Checkout SCM	testing pipeline
Average stage times: (Average full run time: ~4s)	857ms	503ms
#5 May 01 21:27 No Changes	931ms	316ms
#4 May 01 21:25 1 commit	576ms	338ms

#5	May 1, 2019 3:57 PM
#4	May 1, 2019 3:55 PM
#2	Apr 26, 2019 3:45 PM

Declarative Pipeline Execution Result

Ideally, we can connect any Build job with these scripts to upstream or downstream the pipelines. So this pipelines will provide continuous delivery right from the small changes from the major changes that are happening from the source code. Especially the pipeline is very useful when we connect multiple environments together and deliver the software. Like, Build – Test – Deploy on every environment. say Dev environment to QA or UAT to Stage to Prod.

Stay tuned. We will see the detailed tutorial of the Scripted and Declarative pipelines and tools like Blue oceans in upcoming articles.

0

Article Rating

★★★★★

← Previous Post Next Post →

Subscribe Connect with

Be the First to Comment!

B I U S {} [+]

0 COMMENTS

Deal Of the Day

FREE

How To Crack Job Interview?

Learn to market yourself in an #interview through the "How To #CrackJobInterview" course available for #free for 3 days using coupon code "GETFREEINT"

GETFREEINT

Valid until May 11, 2020

GO TO THE DEAL

Recent Posts

Complete Terraform Tutorial Part – 4 – Terraform State

Complete Terraform Tutorial Part – 3 – Terraform CLI

Complete Terraform Tutorial Part – 2 – Terraform Configuration File.

Complete Terraform Tutorial Part – 1 – Introduction

How to Configure and Running Jenkins Behind Apache Reverse Proxy?

Categories

- App Development (17)
- Cloud (9)
- DevOps (48)
  - Build and Package (11)
  - DevOps and Business (1)
  - Monitoring and Analytics (1)
  - Planning and Development (4)
  - Release and Operation (15)
  - Testing and QA (3)
- DevSecOps (4)
- How To's (12)

Project Management (2)

Whispers of DigitalVarys (7)

Services

Advertise

Partnership

Showcase

Deals

Offers and Deals

Learning Materials

Contact Us

Contact Services