

Jenkins Class – 8

We have 3 types of pipeline creation in Jenkins, which is,

1. Freestyle
2. Pipeline
 - a. Declarative pipeline
 - b. Scripted pipeline

PIPELINES:

Jenkins Pipeline is a combination of plugins that supports integration and implementation of continuous delivery pipelines.

A Pipeline is a group of events interlinked with each other in a sequence.

There are two types of Jenkins pipeline syntax used for defining your JenkinsFile

1. Declarative
2. Scripted

NEW ITEM — > NAME — > PIPELINE — > SAVE

DECLARATIVE

```
pipeline {
  agent any

  stages {
    stage('Hello') {
      steps {
        echo 'Hello World'
      }
    }
  }
}
```

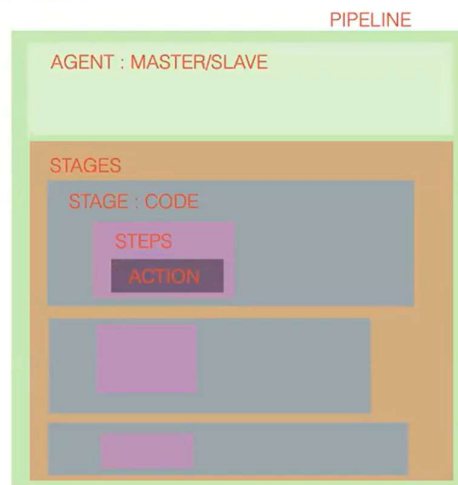
SCRIPTED

```
node {
  stage("stage1"){
    echo "hai"
  }
}
```

The structure of Declarative pipeline.

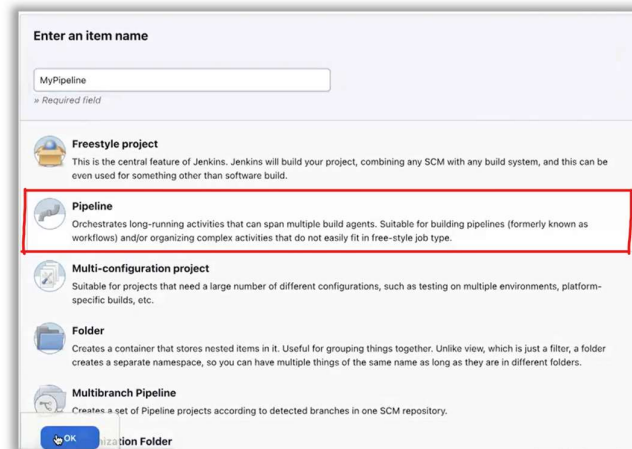
```
pipeline {
  agent any
  stages {
    stage ("Code") {
      steps {
        action
      }
    }
  }
}
```

P = PIPELINE
A = AGENT
S = STAGES
S = STAGE
S = STEPS



The pipeline job is similar to the freestyle but we have additional options in the pipeline jobs.

Launch the Jenkins job as same as freestyle but select the pipeline option while launching the job.



The option is also the same as freestyle but in the pipeline, we don't have the post-build option. Instead of that, we have a scripting console where we need to write the script to deploy our job.

Syntax of the pipeline script is

```
pipeline {
  agent any

  stages {
    stage('Hello') {
      steps {
        echo 'Hello World'
      }
    }
  }
}
```

pipeline – this is initializing the pipeline

agent any – this is agent calling which is master or slave like that.

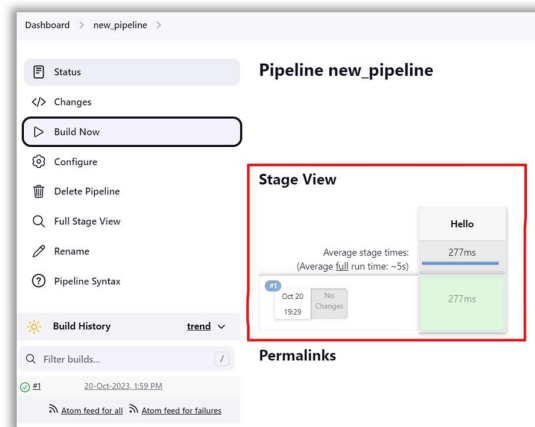
Stages – this is stages of the pipeline. In the freestyle we have a plugin to build the pipeline visualization but in this part we have to declare the pipeline via script. We should declare all the stages inside of the stages part.

Stage – this is stage which should declare inside of the stages. We should give the name of the stage inside of the ("Hello") like this. Every stage should have only one step.

Steps - this is steps to declare what are the actions we have to do on it. **One stage should have only one step. We can't add more than one step in a stage.**

Note: The syntax should follow the proper indentation.

The output of the pipelines will look like the below snap.



If we want to add the new stage we have to add the line where the current stage has ends.

```
pipeline {
  agent any
  stages {
    stage ("Code") {
      steps {
        echo "This is code stage"
      }
    }
  }
}
```

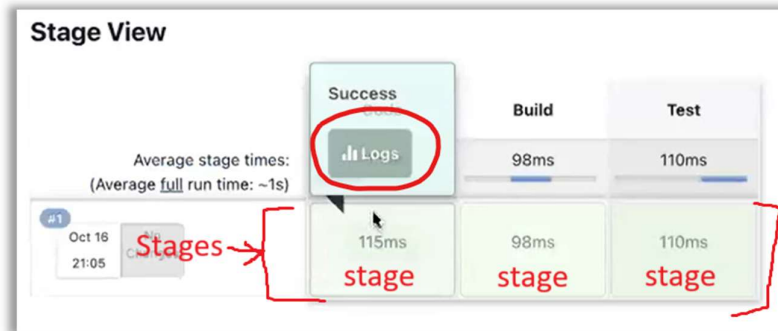
If you feel confused you can minimize the line by click on the arrow.

```
1 pipeline {
2   agent any
3   stages {
4     stage ("Code") {
5       steps {
6         echo "This is code stage"
7       }
8     }
9   }
10 }
```

We can add multiple stage inside the stages.

```
1 pipeline {
2   agent any
3   stages {
4     stage ("Code") {}
5     stage ("Build") {}
6     stage ("Test") {
7       steps {
8         echo "This is Testing stage"
9       }
10    }
11  }
12 }
```

If you want to see the logs, you have to place your cursor on the stage. If you want to see overall logs you can go to the console output.



To execute the command, use the following syntax : **sh "command"**

```
1 pipeline {
2   agent any
3   stages {
4     stage ("Commands") {
5       steps {
6         sh 'touch Mustafa.pdf'
7       }
8     }
9   }
10 }
```

If you have to create multiple stage, make sure that the stage name should be unique.

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Hello') {
6       steps {
7         echo 'Hello World'
8       }
9     }
10    stage('file'){
11      step{
12        sh 'touch file{1..3}.txt'
13      }
14    }
15    stage('folder'){
16      step{
17        sh 'mkdir new_folder'
18      }
19    }
20    stage('move'){
21      step{
22        sh 'mv *.txt new_folder/'
23      }
24    }
25  }
26 }
```

We can use multiple commands in one step.

```
1 pipeline {
2   agent any
3   stages {
4     stage ("mycommands") {
5       steps {
6         sh 'mkdir krishna'
7         sh 'touch sravanth'
8         sh 'date'
9       }
10    }
11  }
12 }
```

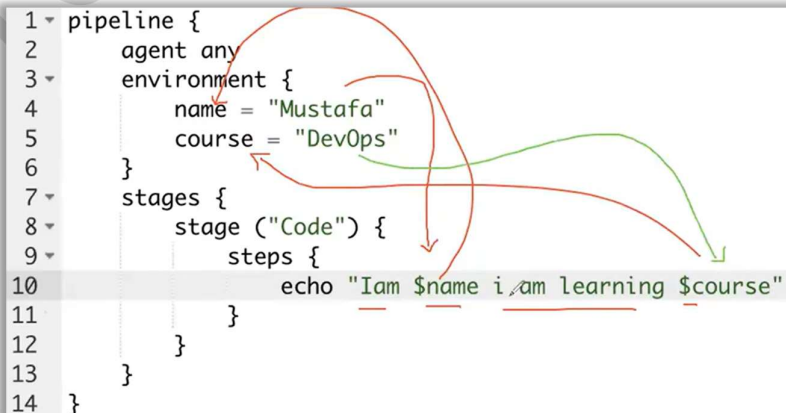
If we want to add multiple action in single step. We can but all the actions will be performed based on the type of actions.

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Hello') {
6       steps {
7         echo 'Hello World'
8         sh '''
9           touch file{1..3}.txt
10          mkdir new_folder
11          mv *.txt new_folder/
12          ...
13        '''
14      }
15    }
16  }
17 }
```

Environmental Variable

To declare the variable, we should declare the variable in environment. The environment should be under the agent. We need to use dollar symbol (\$) to call the variable.

```
1 pipeline {
2   agent any
3   environment {
4     name = "Mustafa"
5     course = "DevOps"
6   }
7   stages {
8     stage ("Code") {
9       steps {
10        echo "Iam $name i am learning $course"
11      }
12    }
13  }
14 }
```



Global variable vs Local variable

The global variable is used to call a variable to the entire code. But the local variable only valid within the current block. Outside of the block we can't call it.

Global variable should be placed under agent. Local variable should be placed inside of the stage where you have to call that variable as a local variable.

```
1 pipeline {
2   agent any
3   environment{
4     x = 10
5   }
6
7   stages {
8     stage('Global') {
9       steps {
10        echo 'Hello World. i have Rs. $x '
11      }
12    }
13    stage('Local') {
14      environment{
15        x=50
16      }
17      steps {
18        echo 'Hello World. i have Rs. $x'
19      }
20    }
21  }
22 }
23
24 }
```

The diagram shows a Jenkins pipeline configuration. A green arrow labeled 'Global variable' points to the `x = 10` assignment in the `environment` block at line 4. A red arrow originates from this assignment and points to the `$x` variable in the `echo` command at line 10. Another green arrow labeled 'Local Variable' points to the `x=50` assignment in the `environment` block of the 'Local' stage at line 15. A red arrow originates from this assignment and points to the `$x` variable in the `echo` command at line 18. This illustrates that the global variable `x` is accessible throughout the pipeline, while the local variable `x` is only accessible within the 'Local' stage.