

CICD End To End Flow

DevOps - CI/CD

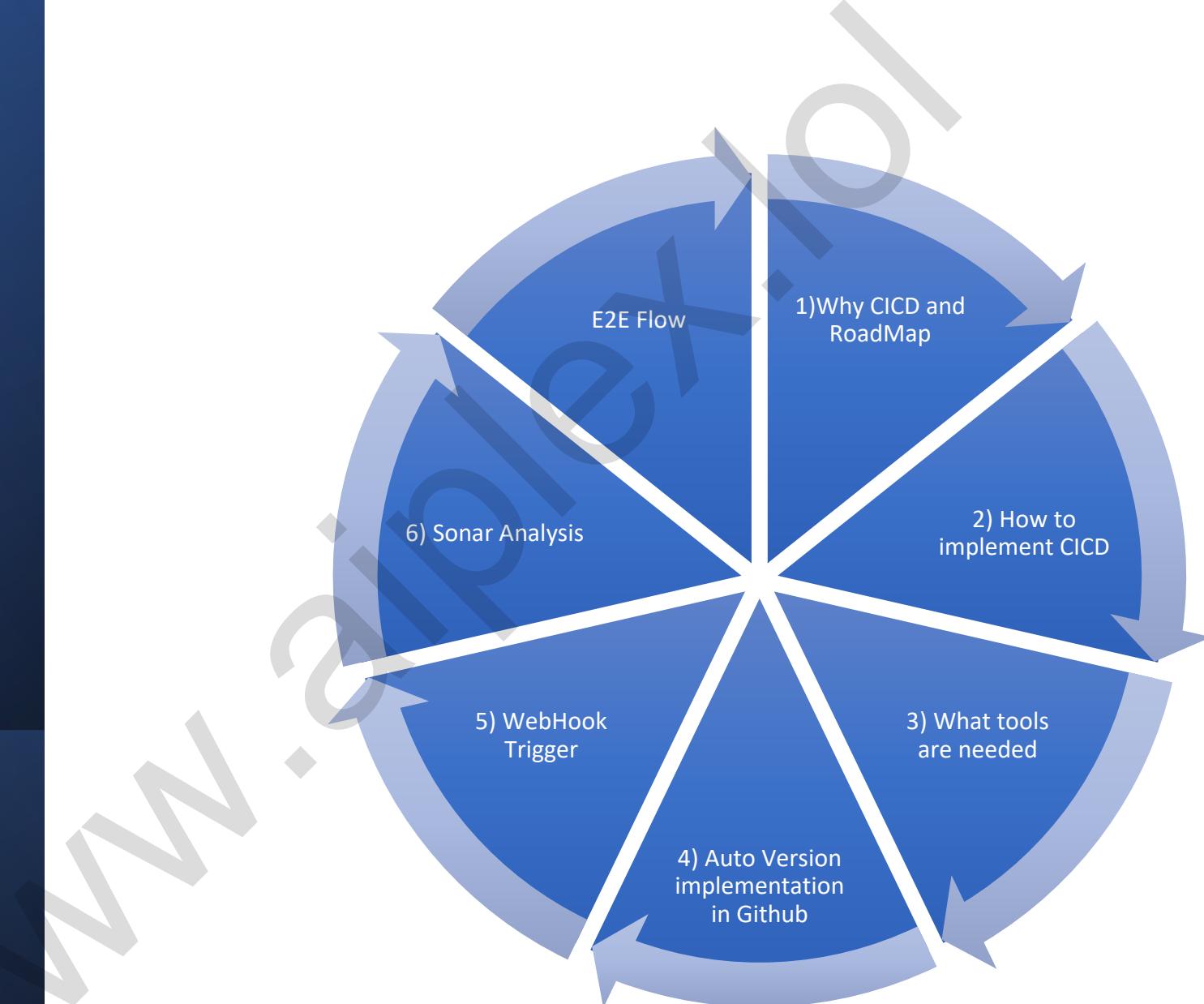


Get more free courses at www.aiplex.lol

By Praveen Singampalli

CICD discussion:

Get more free courses at www.aiplex.lol





KubeSphere Container Platform

splunk®



Deployment
Service/Ingress



Developer



Monitor and Operate



Deploy to Kubernetes



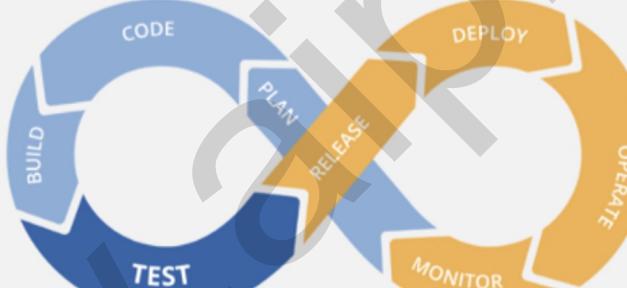
Build and Push Image



docker



JFrog



Source Code



Repository



Unit Test



Static Code Analysis



Build

JUnit

sonarqube

Jenkins

maven

Get more free courses at www.aiplex.lol

Master Slave Setup

- Reduce the load on system
- Have different systems tied up for different code pipelines
- Failover Mechanism

Jenkins > Nodes >

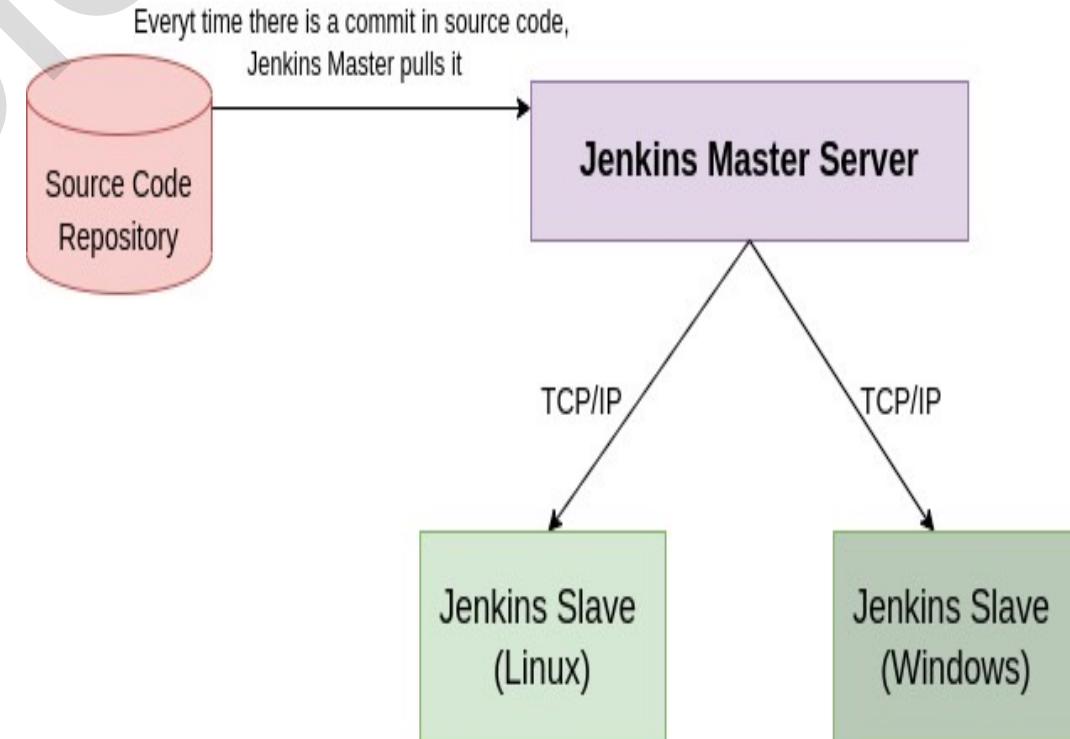
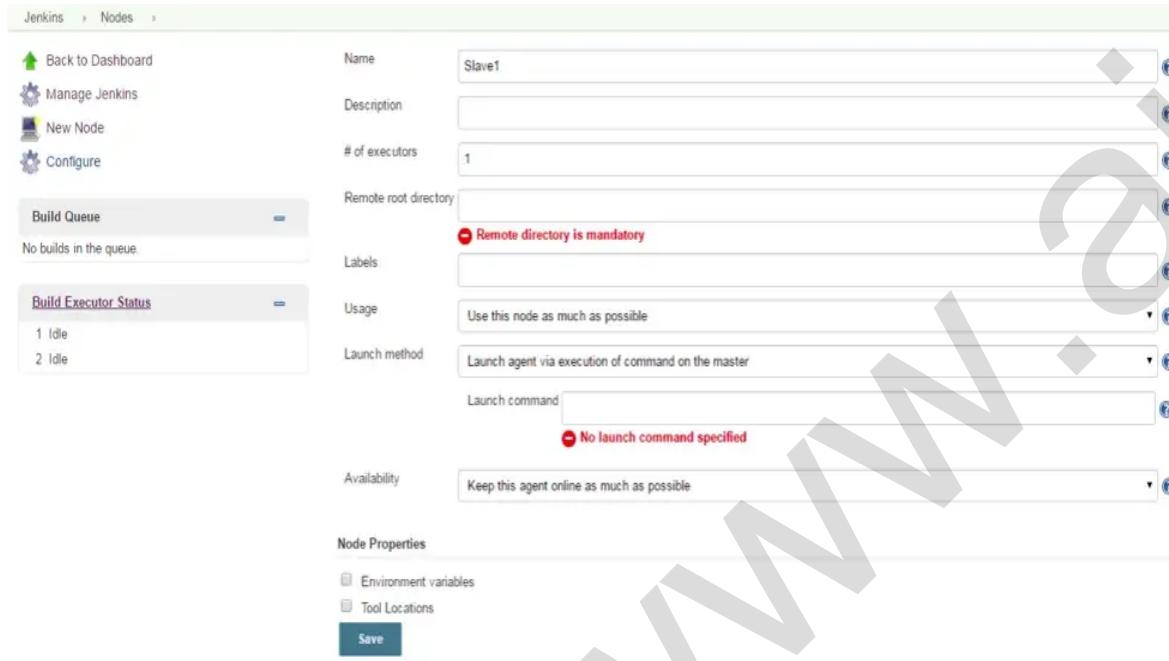
Back to Dashboard Manage Jenkins New Node Configure

Build Queue No builds in the queue.

Build Executor Status 1 Idle 2 Idle

Name Slave1 Description # of executors 1 Remote root directory Remote directory is mandatory Labels Usage Use this node as much as possible Launch method Launch agent via execution of command on the master Launch command No launch command specified Availability Keep this agent online as much as possible

Node Properties Environment variables Tool Locations Save



Jenkins Master slave Setup Steps

1

Welcome to Jenkins!

Please [create new jobs](#) to get started.

ENABLE AUTO REFRESH

New Item
People
Build History
Manage Jenkins **←**
My Views
Credentials

Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle

3

Back to Dashboard
Manage Jenkins
New Node **←**
Configure

Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle

5

Back to List
Status
Delete Agent
Configure
Build History
Load Statistics
Log

Agent Slave1

Connect agent to Jenkins one of these ways:

- Launch** Launch agent from browser
- Run from agent command line:
`java -jar slave.jar -jnlpUrl http://localhost:8080/computer/Slave1/slave-agent.jnlp -secret 19385e54ab9539fd6272df525dc9e7687efea162e44a83d918ca918294eb9f8e`

Mark this node temporarily offline

Save

Build Executor Status
None

2

Load Statistics
Check your resource utilization and see if you need more computers for your builds.

Jenkins CLI
Access/manage Jenkins from your shell, or from your script.

Script Console
Executes arbitrary script for administration/trouble-shooting/diagnostics.

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

About Jenkins
See the version and license information.

4

Back to Dashboard
Manage Jenkins
New Node
Configure

Build Queue
No builds in the queue

Build Executor Status
1 Idle
2 Idle

Name: Slave1
Description:
of executors: 1
Remote root directory: **Remote directory is mandatory**
Labels:
Usage: Use this node as much as possible
Launch method: Launch agent via execution of command on the master
Launch command: **No launch command specified**
Availability: Keep this agent online as much as possible

Node Properties
 Environment variables
 Tool Locations

Save

Different types of Jenkins Pipeline

- **Declarative Pipeline Syntax**

The declarative syntax is a new feature that uses code for the pipeline. It provides a limited pre-defined structure. Thereby, it offers an easy & simple continuous delivery pipeline. Moreover, it uses a *pipeline block*.

- **Scripted Pipeline Syntax**

Unlike declarative syntax, the *scripted pipeline syntax* is the old traditional way to write the *Jenkinsfile* on Jenkins web UI. Moreover, it strictly follows the groovy syntax and helps to develop a complex pipeline as code.

```
pipeline {  
    agent any  
    stages {  
        stage('One') {  
            steps {  
                echo 'Hi, welcome to pipeline demo...'  
            }  
        }  
        stage('Two') {  
            steps {  
                echo('Sample testing of Stage 2')  
            }  
        }  
        stage('Three') {  
            steps {  
                echo 'Thanks for using Jenkins Pipeline'  
            }  
        }  
    }  
}
```

Declarative Pipeline

```
pipeline {  
    agent {  
        // executes on an executor with the label 'some-label' or 'docker'  
        label "some-label || docker"  
    }  
  
    stages {  
        stage("foo") {  
            steps {  
                // variable assignment (other than environment variables) can only  
                // complex global variables (with properties or methods) can only  
                // env variables can also be set within a script block  
                script {  
                    foo = docker.image('ubuntu')  
                    env.bar = "${foo.imageName()}"  
                    echo "foo: ${foo.imageName()}"  
                }  
            }  
        }  
        stage("bar") {  
            steps{  
                echo "bar: ${env.bar}"  
                echo "foo: ${foo.imageName()}"  
            }  
        }  
    }  
}
```

Scripted Pipeline

```
node {  
  
    git url: 'https://github.com/jfrogdev/project-examples.git'  
  
    // Get Artifactory server instance, defined in the Artifactory Plugin  
    def server = Artifactory.server "SERVER_ID"  
  
    // Read the upload spec and upload files to Artifactory.  
    def downloadSpec =  
        '''{  
        "files": [  
            {  
                "pattern": "libs-snapshot-local/*.zip",  
                "target": "dependencies/",  
                "props": "p1=v1;p2=v2"  
            }  
        ]  
    }'''  
  
    def buildInfo1 = server.download spec: downloadSpec  
  
    // Read the upload spec which was downloaded from github.  
    def uploadSpec =  
        '''{  
        "files": [  
            {  
                "pattern": "resources/Kermit.*",  
                "target": "libs-snapshot-local",  
                "props": "p1=v1;p2=v2"  
            },  
            {  
                "pattern": "resources/Frogger.*",  
                "target": "libs-snapshot-local"  
            }  
        ]  
    }'''
```

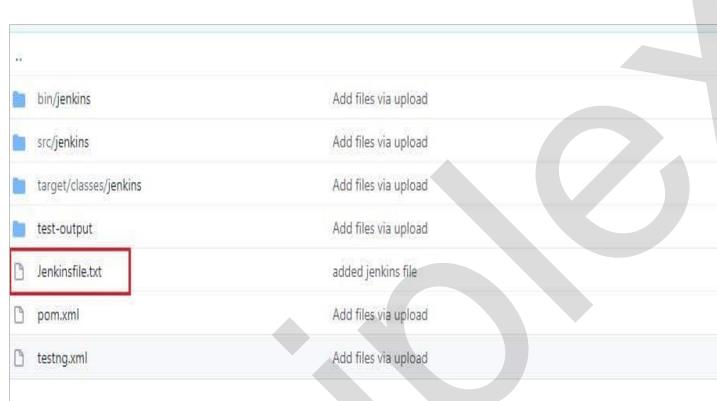
Scripted Pipeline

The screenshot shows the Jenkins Pipeline configuration page for a scripted pipeline. The main area displays a Groovy script:

```
4 *: stage('One') {  
5:     steps {  
6:         echo 'Hi, welcome to pipeline demo...'  
7:     }  
8: }  
9 *: stage('Two') {  
10:    steps {  
11:        echo('Sample testing of Stage 2')  
12:    }  
13: }  
14 *: stage('Three') {  
15:    steps {  
16:        echo 'Thanks for using Jenkins Pipeline'  
17:    }  
18: }  
19:  
20:  
21: }
```

Below the script, there are two buttons: "Save" and "Apply".

Declarative Pipeline(Jenkinsfile add to git repo)



The screenshots show the Jenkins Pipeline configuration page for a declarative pipeline. The main area displays the SCM configuration:

1. Pipeline Definition: Pipeline script from SCM, SCM, None, None, Git, Jenkinsfile, Lightweight checkout.

2. Pipeline Syntax: Pipeline script from SCM, SCM, Git, Repositories, Repository URL: https://github.com/gunjanikaushik/Jenkins.git, Credentials: /*****, Add.

3. Now, you will get an option to input your **Repository URL** and **credentials**.

4. Next, you may set the **branch** or let it be blank for any branch. In the script path, you need to write the **Jenkinsfile** name that exists in your repository. Click on **Save**, and there you go, your declarative pipeline is ready for use.

JenkinsFile Data

Get more free courses at www.aiplex.lol

Key concepts of Jenkinsfile

- **Pipeline** - A pipeline is a set of instructions that includes the processes of continuous delivery. For example, creating an application, testing it, and deploying the same. Moreover, it is a critical element in declarative pipeline syntax, which is a collection of all stages in a Jenkinsfile. We declare different stages and steps in this block.
- **pipeline{ } Node** - A node is a key element in scripted pipeline syntax. Moreover, it acts as a machine in Jenkins that executes the Pipeline.
- **node{ } Stage** - A stage consists of a set of processes that the Pipeline executes. Additionally, the tasks are divided in each stage, implying that there can be multiple stages within a Pipeline. The below snippet shows the different stages that one can define in a Pipeline.
- **Steps** - A step in Jenkins defines what we have to do at a particular step in the process. There can be a series of steps within the stage. Moreover, whatever step we define in a stage would be executed within it.
- **Agent** - An agent is a directive that enables the users to execute multiple projects within the same Jenkins instance by distributing the load. Moreover, we assign an executor to the build through an agent. You can either use a single agent for the entire pipeline or use a distinct agent for the different stages of the pipeline. Subsequently, some of the parameters used with agents are -
 - **Any**- Any of the available agents execute the pipeline.
 - **None**- It is used at the pipeline root and implies no global agent, but each stage must specify its own agent.
 - **Label**- The labeled agent is used to execute the pipeline or the specific stage.
 - **Docker**- One can use the Docker images as the execution environment & specifying the agent as docker.

Merge Request from dev to master Branch

Child Pipeline -

DevTeam ↳ Code Commit ↳ To GitHub ↳ Triggers Child Pipeline (WebHook) ↳ Sonar/Blackduck/Fortify(Stage 1 - In Parallel) ↳ Creates the Merge Request(Stage 2 – Sends Email to Approver)

Master Pipeline -

Merge request gets approved ↳ Master Pipeline Trigger(WebHook) ↳ Sonar/Blackduck/Fortify(Stage 1 - In Parallel) ↳ Build Code (Maven) ↳ Artifact Push (Jfrog) ↳ Pull Artifact(Ansible) ↳ Deployment to QA(OnPrem/Cloud) ↳ Performance testing/Integration testing ↳ Deployment to Syage ↳ Send Notification via Email

Webhook Setup

- In your project or group, on the left sidebar, select Settings > Webhooks.
- In URL, enter the URL of the webhook endpoint.
- In Secret token, enter the secret token to validate payloads.
- In the Trigger section, select the events to trigger the webhook

The screenshot shows two panels of the GitLab settings interface for a project named "Kevin / Gitlab Webhook Test".

Left Panel: Webhook Configuration

- URL:** https://www.pushsafer.com/gitlab?k=XXXXXXXXXXXXXXXXXXXX&i=3&s=12&v=1
- Secret Token:** (Empty field)
- Trigger:** A red box highlights the "Push events" section:
 - Push events
 - Tag push events
 - Comments
 - Issues events
 - Confidential Issues events
 - Merge Request events
 - Jobs events
 - Pipeline events
 - Wiki Page events
- SSL verification:** Enable SSL verification
- Add Webhook** button (highlighted with a red box)

Right Panel: Build Triggers

- Build Triggers:**
 - Trigger builds remotely (e.g., from scripts)
 - Build after other projects are built
 - Build periodically
 - Build when a change is pushed to GitHub
 - Build when a change is pushed to GitLab. GitLab CI Service URL: http:// /project/Example%20Job
- Enabled GitLab triggers:**
 - Push Events (checked)
 - Merge Request Events (checked)
 - Rebuild open Merge Requests: Never
 - Comments (checked)
 - Comment for triggering a build: Jenkins please retry a build
- Advanced...** button
- Poll SCM** checkbox (unchecked)

Trigger a pipeline from JenkinsFile

```
pipeline {  
    agent {  
        node {  
            label 'master'  
            customWorkspace "${env.JobPath}"  
        }  
    }  
  
    stages {  
        stage('Start') {  
            steps {  
                sh 'ls'  
            }  
        }  
  
        stage ('Invoke_pipeline') {  
            steps {  
                build job: 'pipeline1', parameters: [  
                    string(name: 'param1', value: "value1")  
                ]  
            }  
        }  
  
        stage('End') {  
            steps {  
                sh 'ls'  
            }  
        }  
    }  
}
```

```
pipeline {  
    agent {  
        docker {  
            image 'node:6-alpine'  
            args '-p 3000:3000 -p 5000:5000'  
        }  
    }  
    environment {  
        CI = 'true'  
    }  
    stages {  
        stage('Build') {  
            steps {  
                sh 'npm install'  
            }  
        }  
        stage('Test') {  
            steps {  
                sh './jenkins/scripts/test.sh'  
            }  
        }  
        stage('Deliver for development') {  
            when {  
                branch 'development' ①  
            }  
            steps {  
                sh './jenkins/scripts/deliver-for-development.sh'  
                input message: 'Finished using the web site? (Click "Proceed" to continue)'  
                sh './jenkins/scripts/kill.sh'  
            }  
        }  
        stage('Deploy for production') {  
            when {  
                branch 'production' ①  
            }  
            steps {  
                sh './jenkins/scripts/deploy-for-production.sh'  
                input message: 'Finished using the web site? (Click "Proceed" to continue)'  
                sh './jenkins/scripts/kill.sh'  
            }  
        }  
    }  
}
```

**PROJECT SETUP IS IN HANDSON
DOCUMENT**

Get more free courses at www.aiplex.lol