Tunis JAM

September, 14 2018

Welcome

Presented by Ichrak Chakroun

Hosts

- Location
  - Cimpress/Vistaprint

Immeuble Lac 8 les Jardins du Lac, les Berges du Lac 2 · Tunis

- Organizers
  - Dhouha melki
  - Amal turki
  - Ichrak chakroun
  - Ahmed Hosni
  - Dhia Balti
  - Ines Cheikhrouhou
  - Naeil zoueidi
  - Hamza boughraira
  - Khalil badri
  - Melik zribi
  - Marwen blel

Introduction

Speakers

- Ahmed Hosni
  - Ahmedhosni.contact@gmail.com
  - Cloudbees News and events

- Ines Cheikhrouhou
  - cheikhrouhouines94@gmail.com
  - Jenkins Pipeline

- Dhia Balti
  - dhia-balti@outlook.com
  - CI/CD for asp .net core application

Continuous Information

Latest Releases

- Weekly (2.118)
  - Release Notes - jenkins.io/changelog/#v2.118
- LTS (2.107.2)
  - Release Notes - jenkins.io/changelog-stable/#v2.107.2
  - Upgrade Guide - jenkins.io/doc/upgrade-guide/2.107/

What's new in 2.107.2 (2018-04-11)

66 1 5

DevOps World - Jenkins World 2018

- DevOps World - Jenkins World 2018

  (formerly known as Jenkins World)

  - San Francisco, CA | Sept 16-19, 2018
  - Nice, France | October 22-25, 2018

DevOps World

Jenkins World

Topics

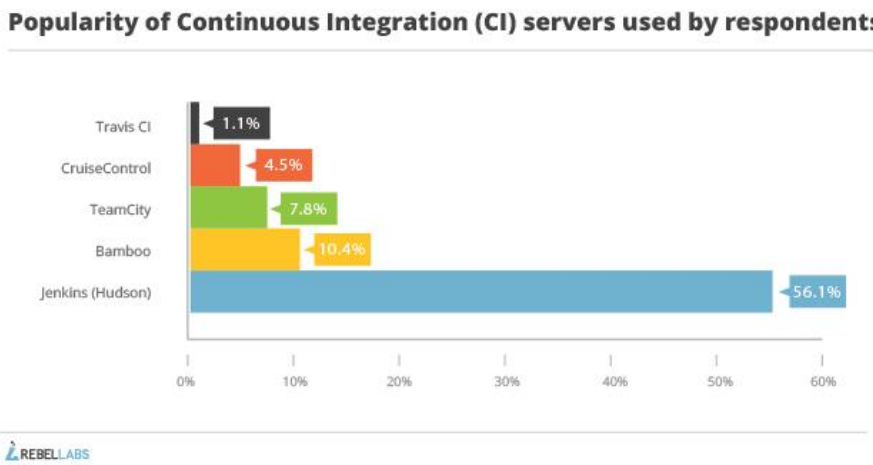
- Title
  - Jenkins Pipeline

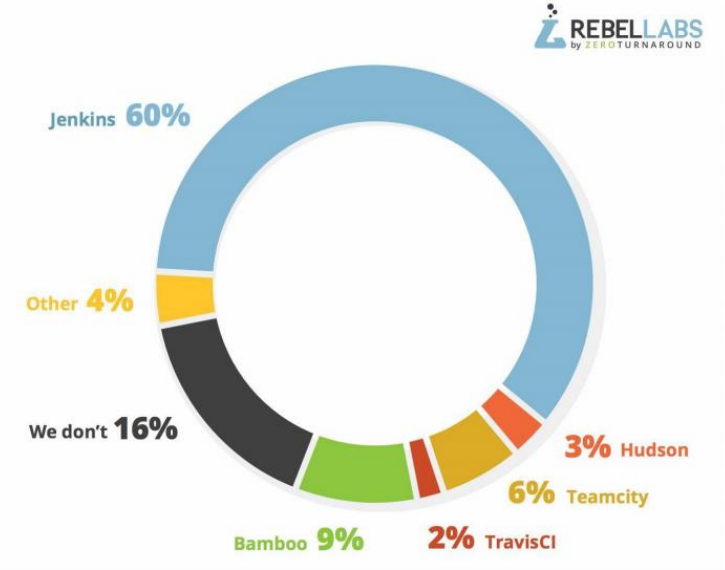
Jenkins pipeline

Jenkins

Jenkins is an open source automation server that enables developers around the world to reliably build, test, and deploy their software.

Jenkins


Popularity of Continuous Integration (CI) servers used by respondents
Travis CI
1.1%
CruiseControl
4.5%
TeamCity
7.8%
Bamboo
10.4%
Jenkins (Hudson)
56.1%
0%
10%
20%
30%
40%
50%
60%
REBELLABS


REBELLABS
by ZEROTURNAROUND
Jenkins 60%
Other 4%
We don't 16%
3% Hudson
6% Teamcity
Bamboo 9%
2% TravisCI

CLOUDBEES

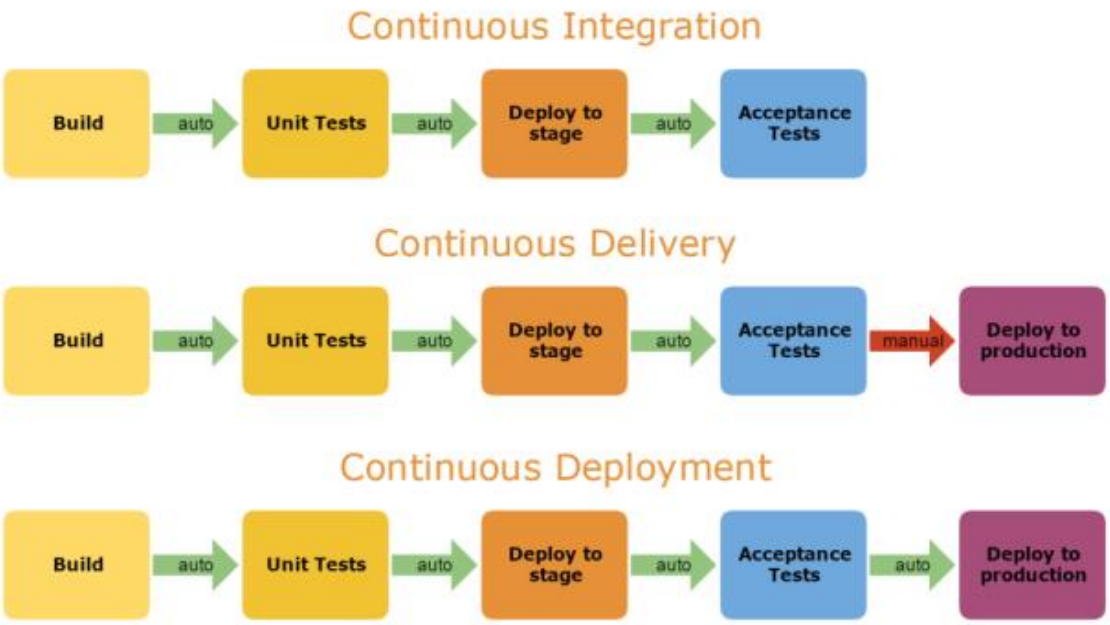
Kohsuke Kawaguchi

Community leader and CTO at CloudBees

- Incorporated in March 9, 2010
- Provide the leading continuous delivery solutions for enterprise DevOps
- We help our customers deliver software at the speed of ideas
- Several hundred people worldwide

CONTINUOUS DEPLOYMENT

Every change is automatically deployed into production



Jenkins Pipeline

- Jenkins Pipeline is a tool for defining your Continuous Delivery/Deployment flow as code

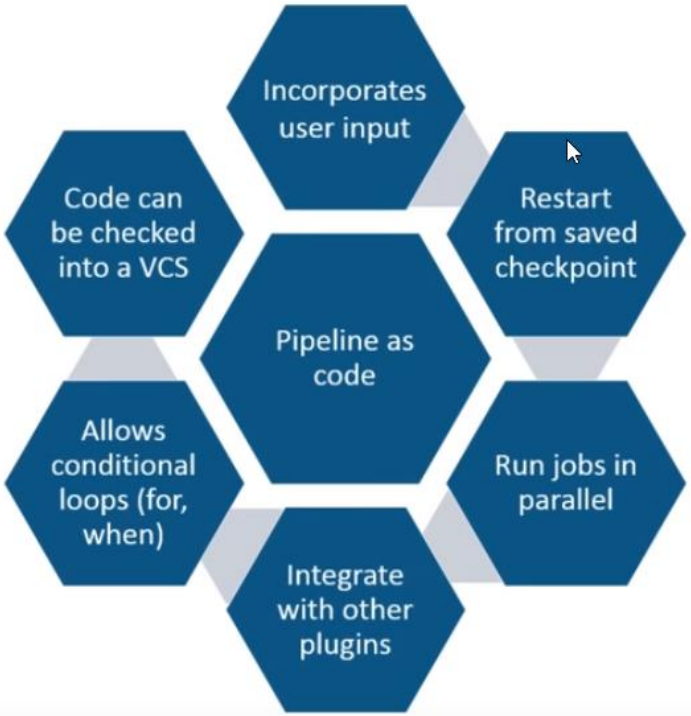
Code

Pausable

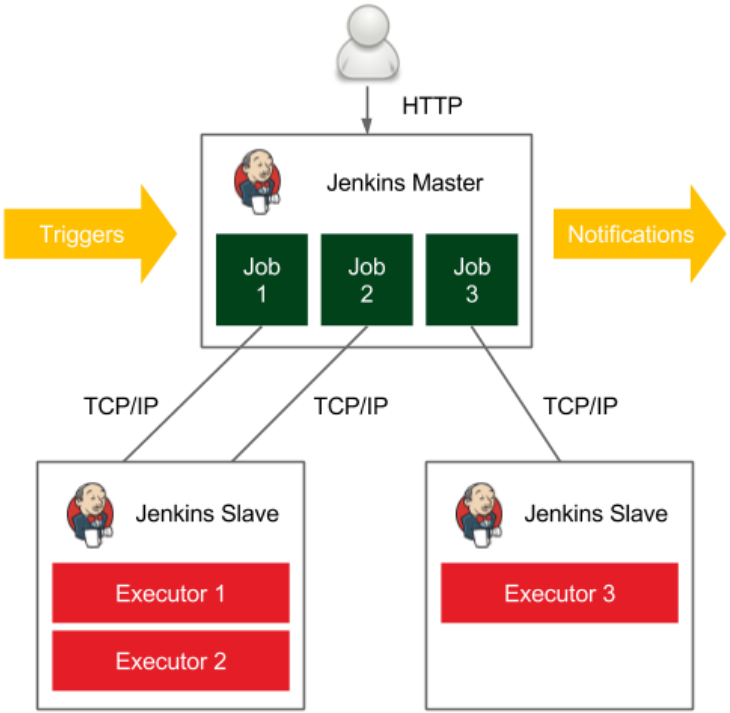
Versatile

Extensible

Jenkins Pipeline



Jenkins Vocabulary



Declarative VS Scripted

```
//Declarative Pipeline

pipeline {
    agent none
    stages {
        stage('Clone') {
            agent { label 'manhnv1_slave' }
            // TODO
        }
        stage('Build') {
            agent { label 'manhnv2_slave' }
            // TODO
        }
        stage('Test') {
            agent { label 'master' }
            // TODO
        }
    }
}
```

```
//Scripted Pipeline

stage('Clone') {
    node('manhnv1_slave') {
        // TODO
    }
}
stage('Build') {
    node('manhnv2_slave') {
        // TODO
    }
}
stage('Test') {
    node('master') {
        // TODO
    }
}
```

Declarative VS Scripted

Similarities:

- The implementors of Jenkins Pipeline found Groovy to be a solid foundation upon which to build what is now referred to as the "Scripted Pipeline" DSL
- Scripted Pipeline offers a tremendous amount of flexibility and extensibility to Jenkins users
- The Groovy learning-curve isn’t typically desirable for all members of a given team, so Declarative Pipeline was created to offer a simpler and more opinionated syntax for authoring Jenkins Pipeline
- They are both durable implementations of “Pipeline as code”

Declarative VS Scripted

Differences:

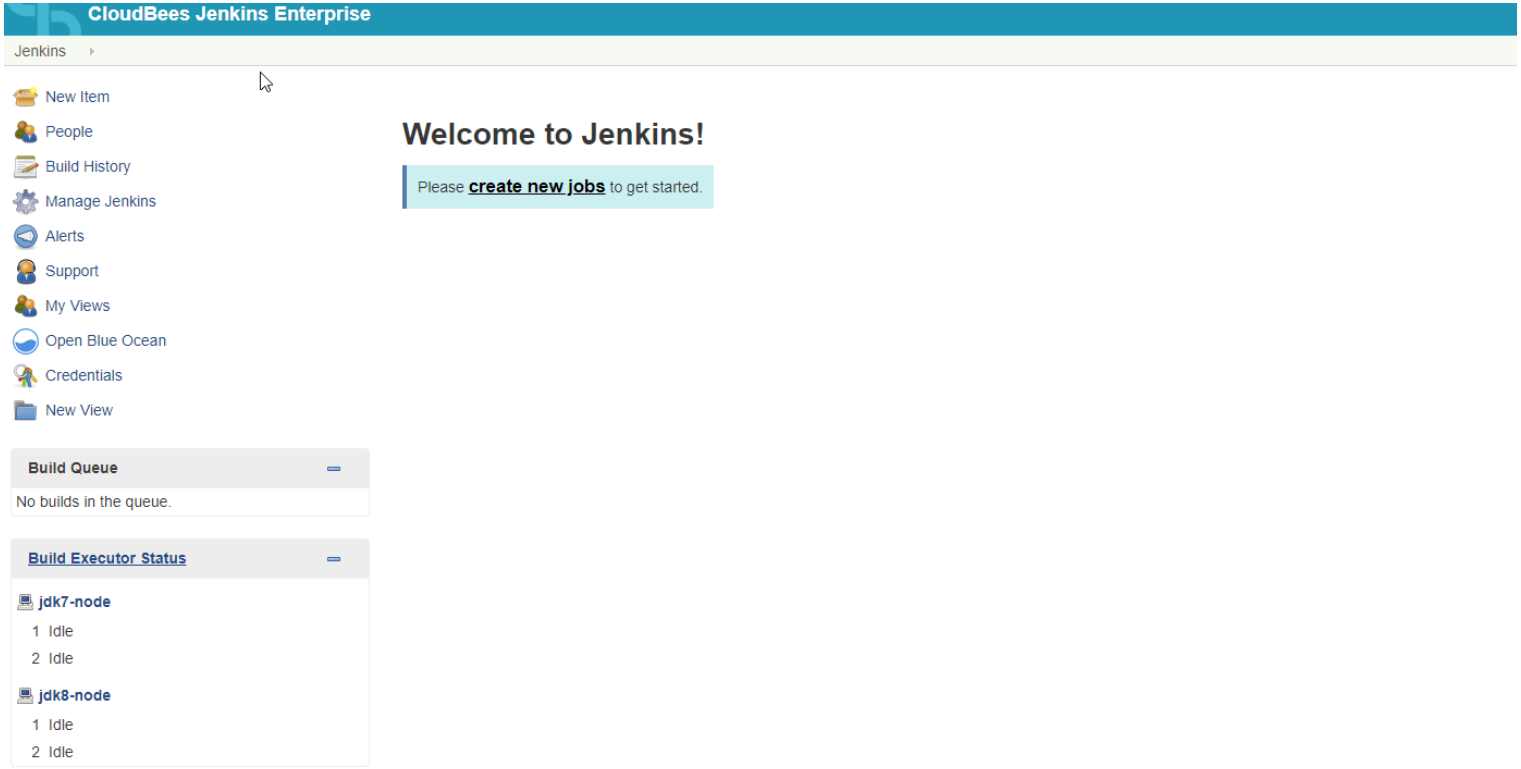
- They differ in syntax and flexibility
- Declarative limits what is available to the user with a more strict and pre-defined structure
- Declarative Pipeline encourages a declarative programming model. Whereas Scripted Pipelines follow a more imperative programming model.
- Declarative Pipeline is written locally in a file and checked into a SCM however, Scripted is written on the Jenkins UI instance

Declarative Pipeline using Blue Ocean Editor

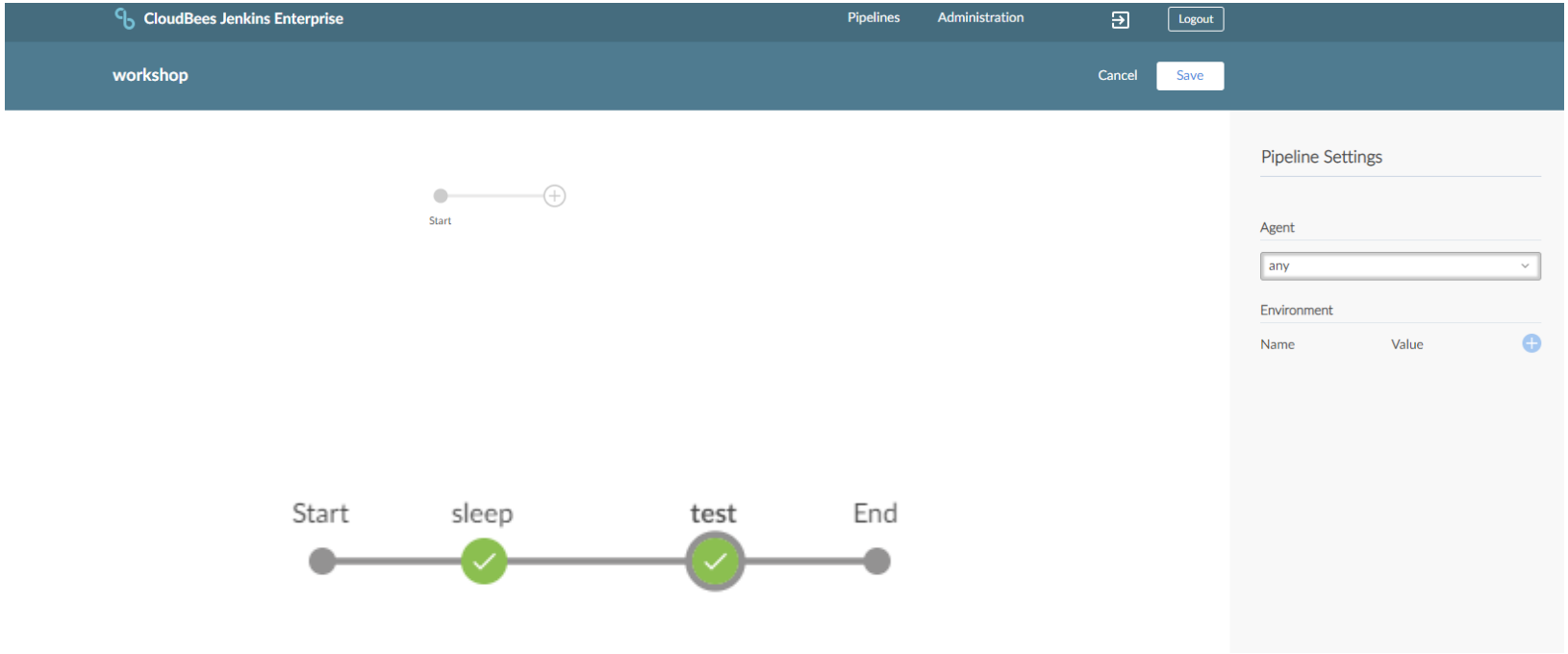
- Basic Declarative Pipeline structure
  - A stage groups tasks to be done
  - A step defines an actual task, such as execute a script or program

```
pipeline {
    agent any

    stages {

        stage('Build') {

            steps {
                echo 'Your building steps...'
            }
        }

        stage('Test') {

            steps {
                echo 'Run your tests...'
            }
        }

        stage('Deploy') {

            steps {
                echo 'Deploy your application...'
            }
        }
    }
}
```

Lab Environment



Lab Time


CloudBees Jenkins Enterprise
Pipelines
Administration
Logout
workshop
Cancel
Save
Start
Start
sleep
test
End
Pipeline Settings
Agent
any
Environment
Name
Value

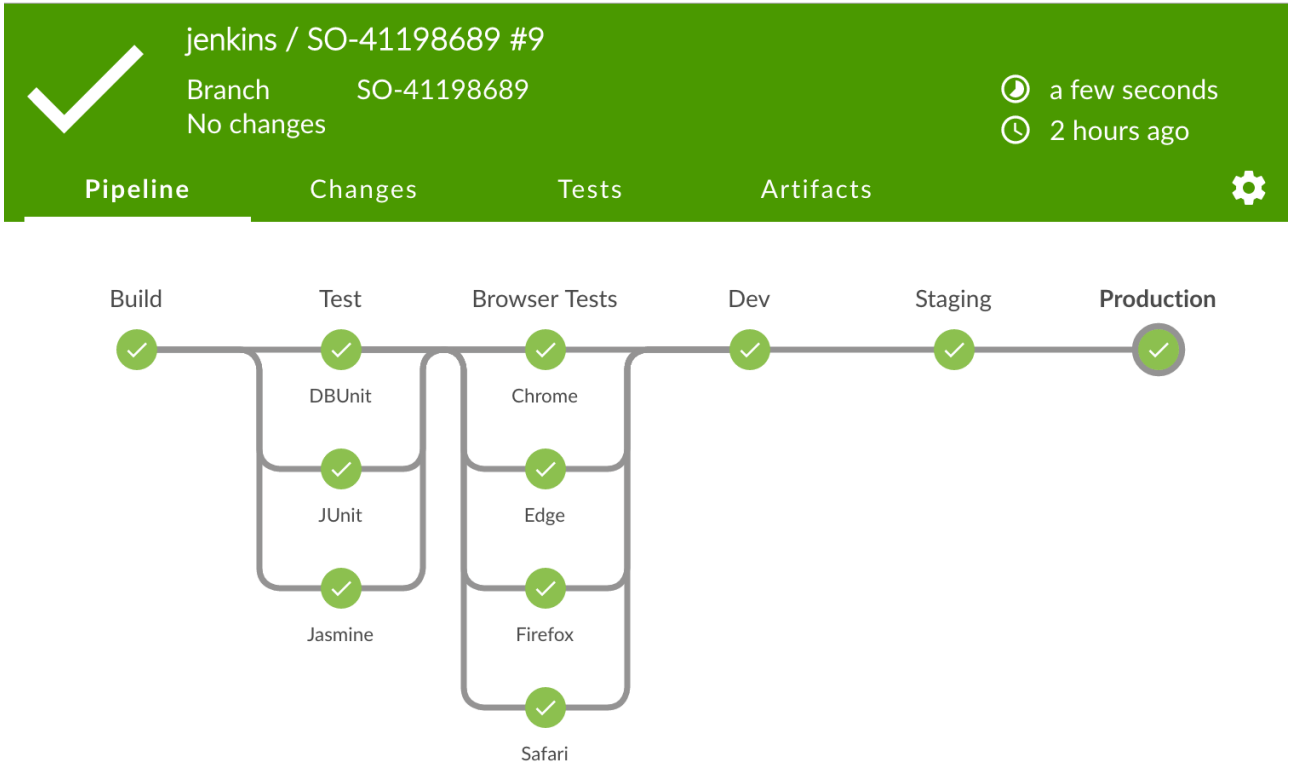
Branches & Artifacts

- Branches : run pipelines in feature branches

- An artifact is a file produced as a result of a Jenkins build
- Archived artifacts are kept forever unless a retention policy is applied to builds to delete them periodically
- Archiving keeps those files in ${JENKINS_HOME}

- A fingerprint is the MD5 checksum of an artifact

Parallel


jenkins / SO-41198689 #9
Branch SO-41198689
No changes
a few seconds
2 hours ago
Pipeline
Changes
Tests
Artifacts
Build
Test
Browser Tests
Dev
Staging
Production
DBUnit
JUnit
Jasmine
Chrome
Edge
Firefox
Safari

Agents

The agent section specifies where the entire Pipeline, or a specific stage, will execute in the Jenkins environment depending on where the agent section is placed. The section must be defined at the top-level inside the pipeline block, but stage-level usage is optional.

- ❑ None
- ❑ Any
- ❑ Docker
- ❑ Label

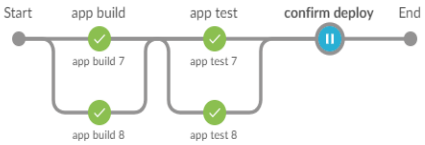
Stash/unstash

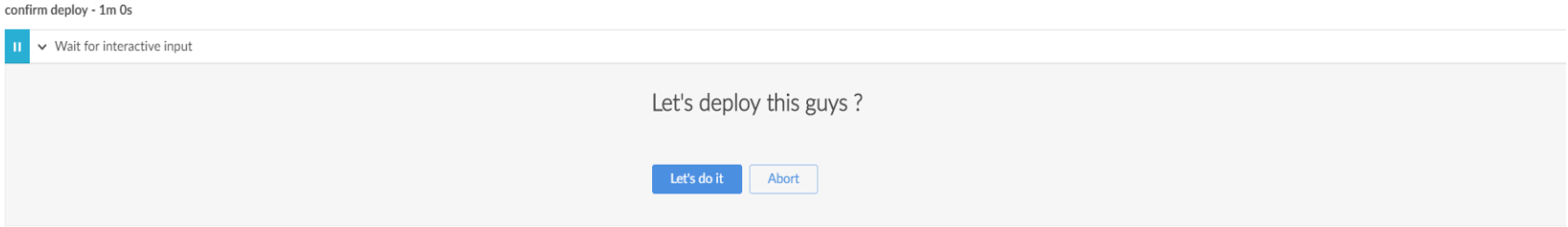
- Use stash to save a set of files for use later in the same build, but in another stage that executes on another node/workspace
- Use unstash to restore the stashed files into the current workspace of the other stage

```
// Run on a node with the "first-node" label.
node('first-node') {
    // Stash that directory and file.
    // Note that the includes could be "output/", "output/*" as below, or even
    // "output/**/*" - it all works out basically the same.
    stash name: "first-stash", includes: "output/*"
}

// Run on a node with the "second-node" label.
node('second-node') {
    // Run the unstash from within that directory!
    dir("first-stash") {
        unstash "first-stash"
    }
}
```

Interactive input

- Jenkins provides the ability to pause the pipeline to wait for input from a human.
- The input step should run on agent none
- Use timeout to avoid waiting for an infinite amount of time





Post section

- Post section contains steps to be executed at the end of a Pipeline run or stage
- Post section is divided into conditions such as always, success, failure

```
pipeline {
    agent any
    stages {
        stage('No-op') {
            steps {
                sh 'ls'
            }
        }
    }
    post {
        always {
            echo 'I have finished'
            deleteDir() // clean up workspace
        }
        success {
            echo 'I succeeded!'
        }
        unstable {
            echo 'I am unstable :/'
        }
        failure {
            echo 'I failed :('
        }
        changed {
            echo 'Things are different...'
        }
    }
}
```

When directive

- Specifies conditions that must be met for Pipeline to execute the stage

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent any

    stages {
        stage('Deploy') {
            when {
              expression {
                currentBuild.result == null || currentBuild.result == 'SUCCESS' (1)
              }
            }
            steps {
                sh 'make publish'
            }
        }
    }
}
```

Parameters

- Parameters directive provides a list of parameters a user should provide when triggering the Pipeline

```
pipeline {
  agent none
  parameters {
    string(name: 'DEPLOY_ENV', defaultValue: 'staging', description: '')
  }
  stages {
    stage ('Deploy') {
      echo "Deploying to ${DEPLOY_ENV}"
    }
  }
}
```

Notifications

Email

```
post {
    failure {
        mail to: 'team@example.com',
            subject: 'Failed Pipeline',
            body: "Something is wrong"
    }
}
```

Hipchat

```
post {
    failure {
        hipchatSend color: 'RED',
        message:'@here build failed.
    }
}
```

Slack

```
post {
    success {
        slackSend channel:'#ops-room',
            color: 'good',
            message: 'Completed successfully.'
    }
}
```

Notifications

PIPELINE SHARED LIBRARIES

- Allow you to share and reuse Pipeline code
- Scale your Jenkins Pipeline usage
- What is it ?
    - A separate SCM , Reusable functions , Called from Pipelines
    - Configured once per Jenkins instance
    - Loaded and used as code libraries for Jenkins Pipelines
    - Modifications made to a shared library function are applied to all Pipelines that call that function

TRIGGERS DIRECTIVE

- Defines special conditions when the Pipeline should be re-triggered; in other words, used to schedule specialized runs of a Pipeline
  - Cron
  - pollSCM
  - upstream

Final Pipeline

	app build	app build 7	app build 8	app test	app test 7	app test 8	confirm deploy	app deploy
Average stage times: (Average full run time: ~1min 30s)	42ms	7s	8s	53ms	6s	6s	70ms	709ms
#3 Sep 12 02:19 — 2 commits	51ms	8s	10s	58ms	6s	6s	58ms (paused for 35s)	673ms
#2 Sep 11 20:30 — 2 commits	34ms	7s	6s	48ms	7s	6s	82ms (paused for 1min 50s)	745ms

Scripted Pipeline

- The first block to be defined is the “node”
- The next required section is the “stage”

```
node {
        stage ('Build') {
                //...
        }
        stage ('Test') {
                //...
        }
}
```

Lab Time

General | Build Triggers | Advanced Project Options | **Pipeline**

Advanced...

Pipeline

Definition: Pipeline script

Script:

```
node {
    echo 'Hello World'
}
```

Hello World

☑ Use Groovy Sandbox

Pipeline Syntax

Lab Time

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step: checkout: General SCM

SCM: Git

Repositories

Repository URL

Please enter Git repository.

Credentials: - none - | Add

Advanced...

Add Repository

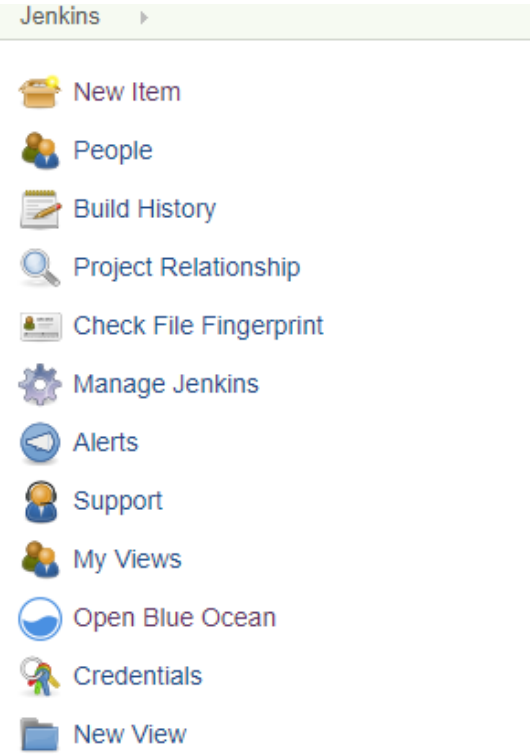
Branches to build

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

Add Branch | Delete Branch

Repository browser: (Auto)

Quick Tour of Cloudbees



https://support.cloudbees.com

Quick Tour of Cloudbees

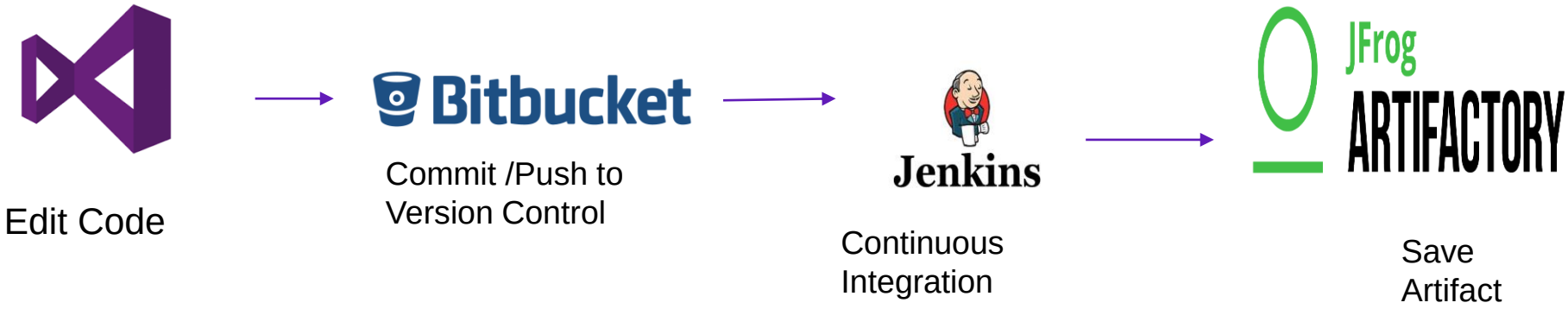
Additional Resources

- CloudBees Network - https://go.cloudbees.com/
- Declarative Pipeline Guided Tour - https://jenkins.io/doc
- Jenkins Pipeline Documentation - https://jenkins.io/doc/book/pipeline
- Jenkins Pipeline Syntax Reference - https://jenkins.io/doc/book/pipeline/syntax
- Jenkins Pipeline Steps Reference - https://jenkins.io/doc/pipeline/steps
- Blue Ocean Documentation - https://jenkins.io/doc/book/blueocean

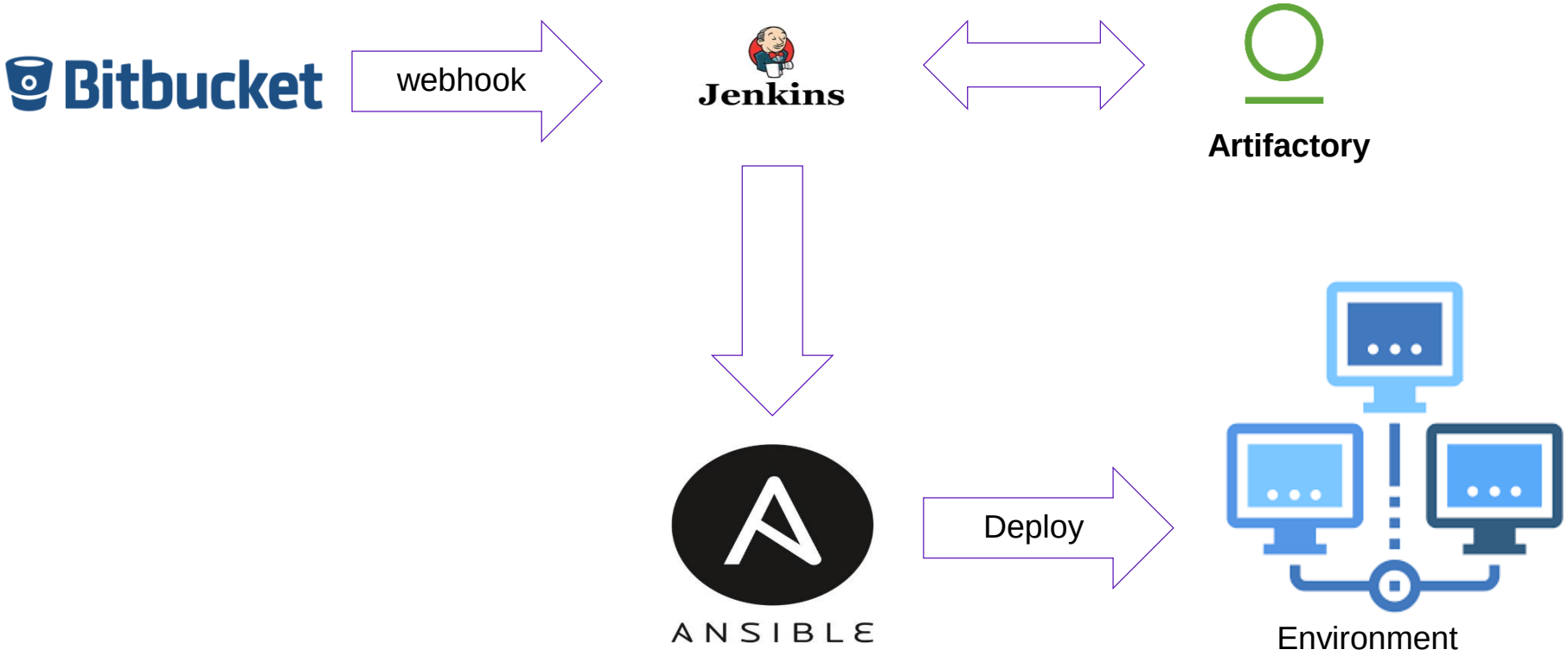
https://dzone.com/refcardz/declarative-pipeline-with-jenkins?chapter=1

CI/CD for Asp.Net Core app using Jenkins

Pipeline


Edit Code
Bitbucket
Commit /Push to
Version Control
Jenkins
Continuous
Integration
JFrog
ARTIFACTORY
Save
Artifact

CI/CD Workflow



Community Time

Supporting A JAM

Thanks!