/kahootali

DevOps - Week 5 - Jenkins

Muhammad Ali Kahoot
Dice Analytics

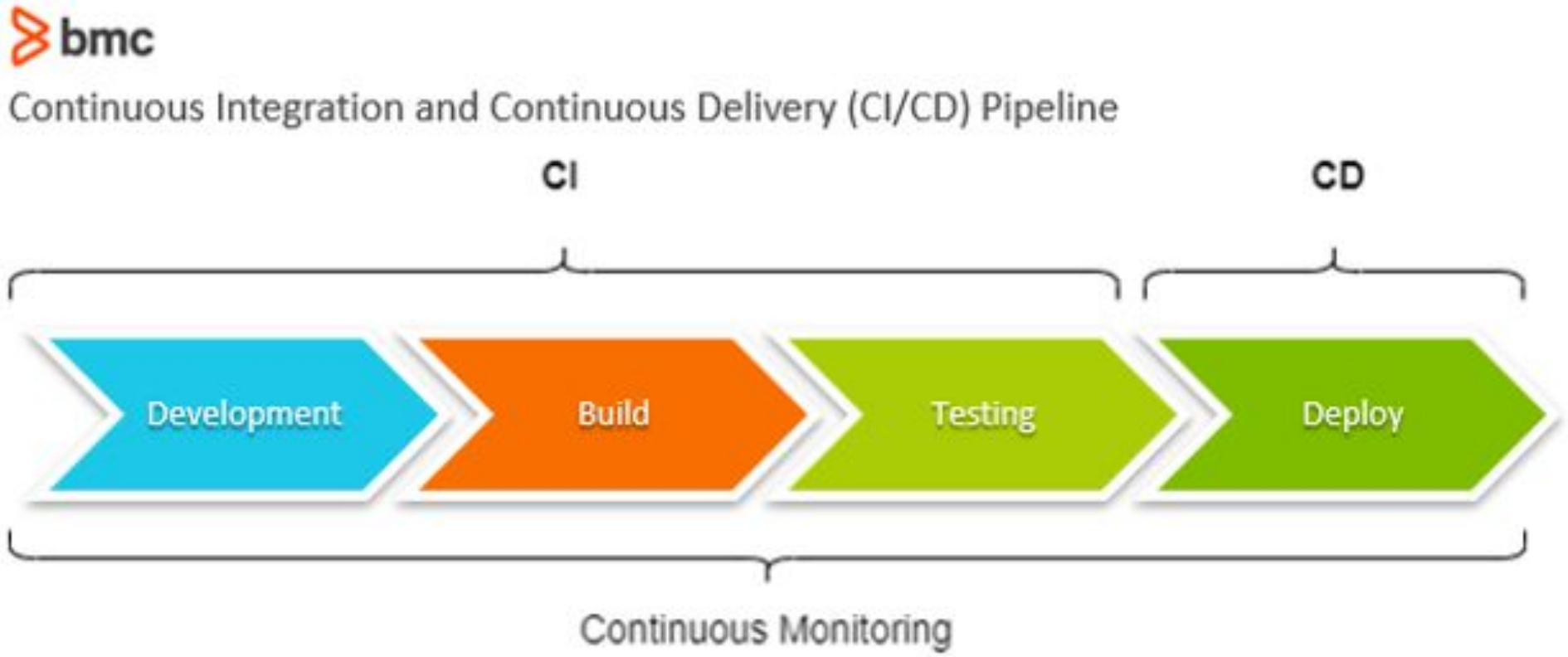
QUIZ - Week 5

- What is CI vs CD
- Container vs VM
- Container vs Image
- What are layers in Docker Images
- RUN vs CMD vs ENTRYPOINT
- Benefits of Docker compose
- Benefits of Docker Swarm

Disclaimer

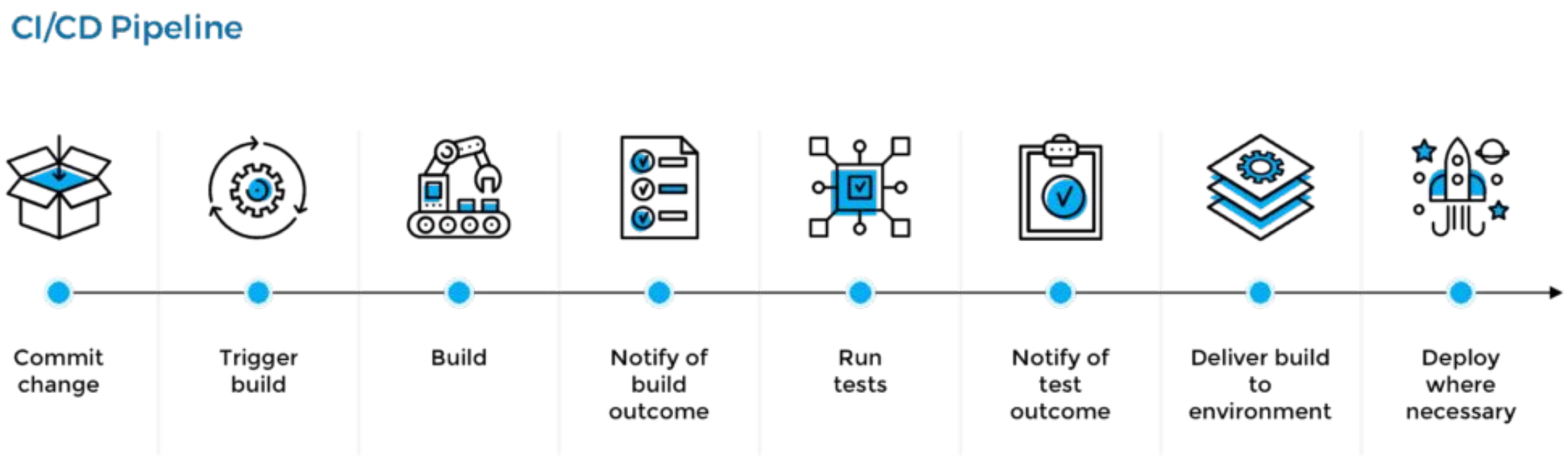
These slides are made with a lot of effort, so it is a humble request not to share it with any one or reproduce in any way.

CI/CD Pipelines



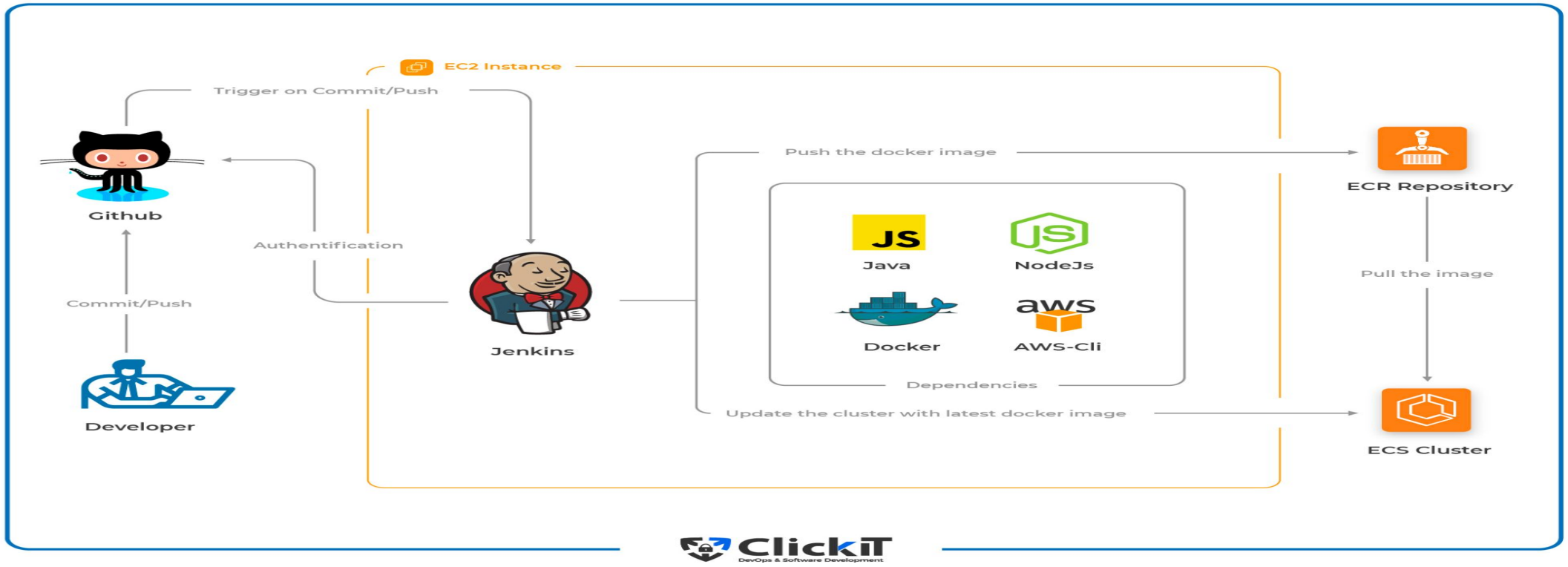
From https://www.bmc.com/blogs/devops-ci-cd-metrics/

CI/CD Pipelines



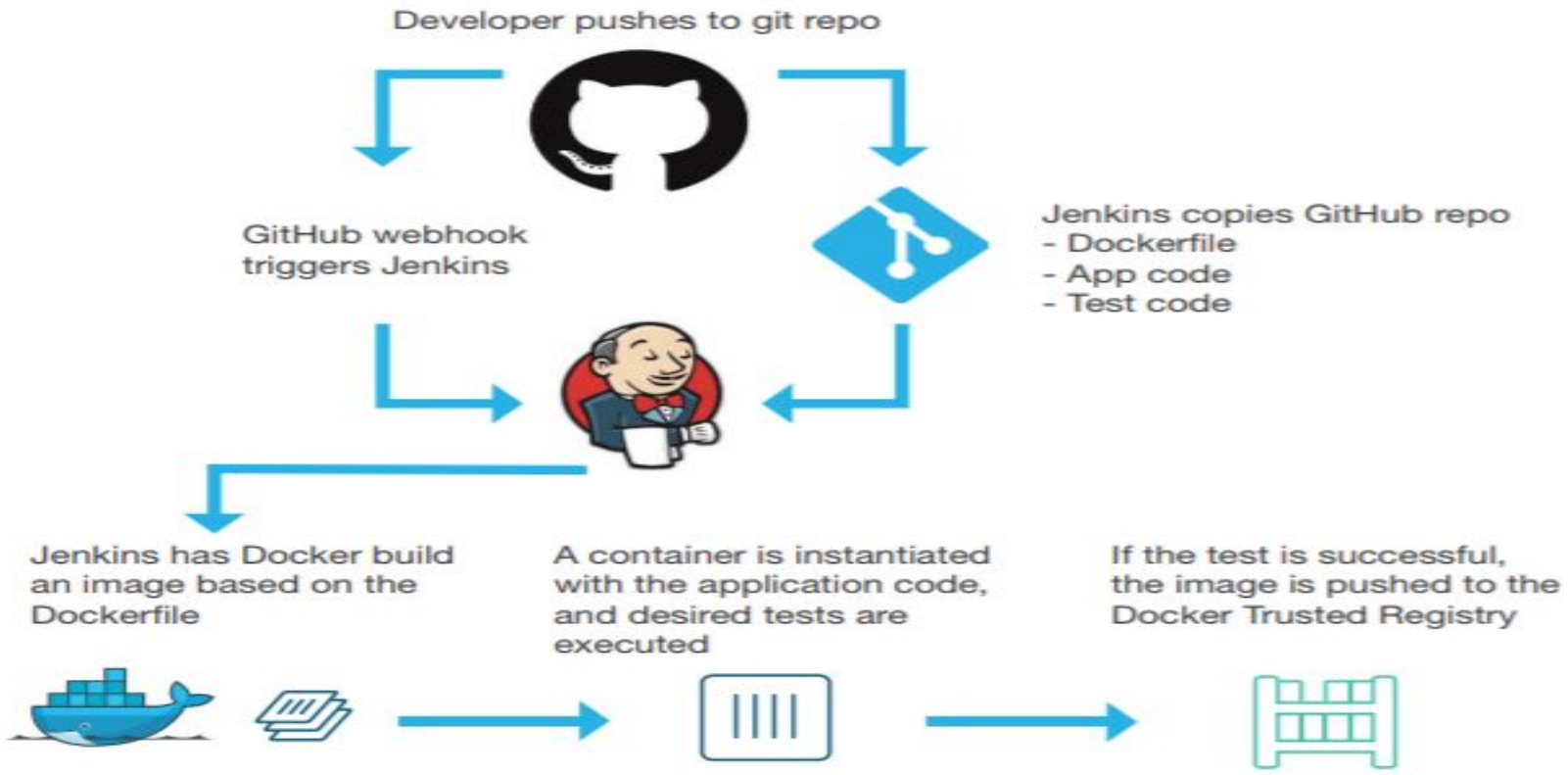
From https://www.plutora.com/blog/understanding-ci-cd-pipeline

CI/CD Pipelines



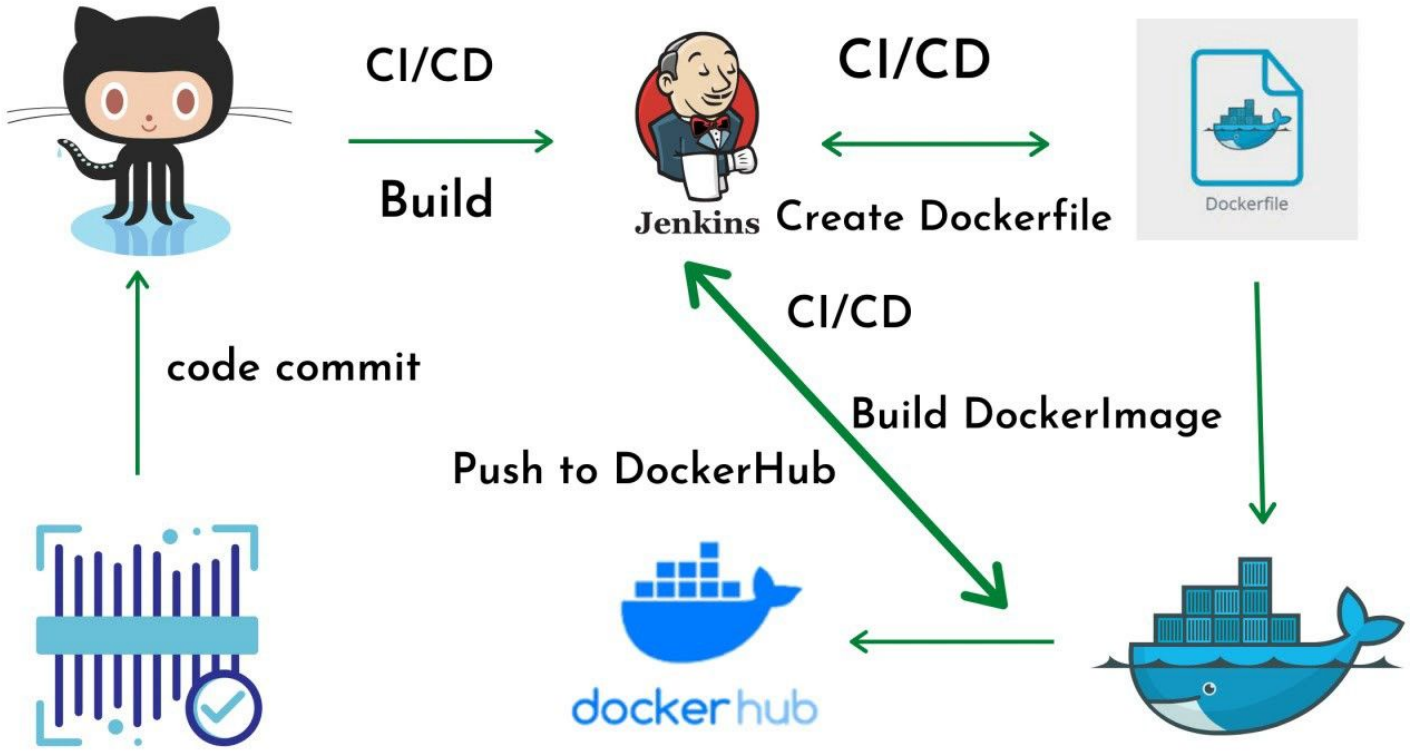
From https://www.clickittech.com/devops/ci-cd-docker/

CI/CD Pipelines



From https://collabnix.com/5-minutes-to-continuous-integration-pipeline

CI/CD Pipelines



From
https://k-adithya21.medium.com/build-a-docker-jenkins-pipeline-to-implement-ci-cd-workflow-e4918bb8bca3

CI/CD Workflows

- Code committed in PR
- Build triggered through Webhook
- Pipeline Steps:
  - Get source code from repo
  - Build & Test the code
  - Build & Push Docker Image
  - Generate Report
  - Deploy in Dev/Stage environment(Optional)
  - Continuous Deployment (Optional)
  - Notify teams especially in Failure

Sample CI/CD Workflows

/kahootali

Feature Branch -> Pull Request	Develop Branch	Main/Master Branch	Git Tag/Release
Build Code	Build Code	Build Code	Build Code
Run Unit Tests (if any)	Run Unit Tests	Run Unit Tests (if any)	Run Unit Tests
Code Quality or Security check	Code Quality or Security check	Code Quality or Security check	Code Quality or Security check
Build & Push Docker Image	Build & Push Docker Image	Build & Push Docker Image	Build & Push Docker Image
Dev can pull image locally & test locally	Deploy to Dev/Staging Environment	Deploy to Sandbox Environment	Deploy to Production
Can deploy to a test or a per-PR based environment	Run Integration or End-to-End Tests or Manual QA on Dev Env	Run any Tests or validations (optional)	

Sample CI/CD Workflows

/kahootali

Feature Branch -> Pull Request	Main/Master Branch	Git Tag/Release
Build Code	Build Code	Build Code
Run Unit Tests (if any)	Run Unit Tests (if any)	Run Unit Tests
Code Quality or Security check	Code Quality or Security check	Code Quality or Security check
Build & Push Docker Image	Build & Push Docker Image	Build & Push Docker Image
Dev can pull image locally & test locally	Deploy to Sandbox Environment	Deploy to Production
Deploy on Dev or Staging Environment	Run any Tests or validations (optional)	
Run Integration or End-to-End Tests or Manual QA on Dev Env		

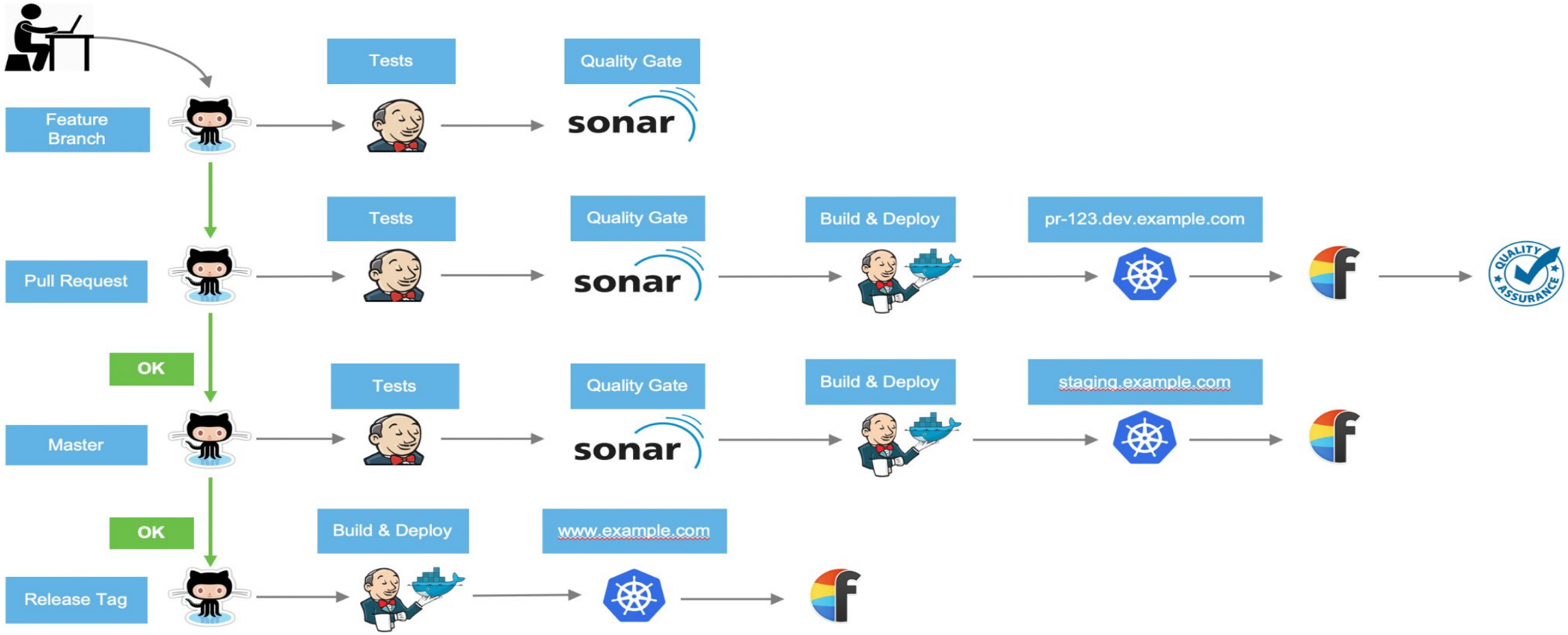
Sample CI/CD Workflows

The above are just sample CI/CD workflows.

Actual workflow can be architected by taking into account many things e.g.:

- Number of environments (Dev, Staging, Sandbox, Prod, etc)
- Environment conventions i.e sandbox contains which data, etc
- Branching Strategy
- Testing flow
- Way of Deployment i.e Docker Compose/Swarm/Kubernetes, etc

CI/CD Pipelines



From: https://docs.cxcloud.com/getting-started-1/setting-up-a-cxcloud-project/configuring-cicd

Github Actions

Sample repos with sample pipelines in Github Actions

- https://github.com/kahootali/golang-sample-app
- https://github.com/kahootali/github-actions-sample

JENKINS

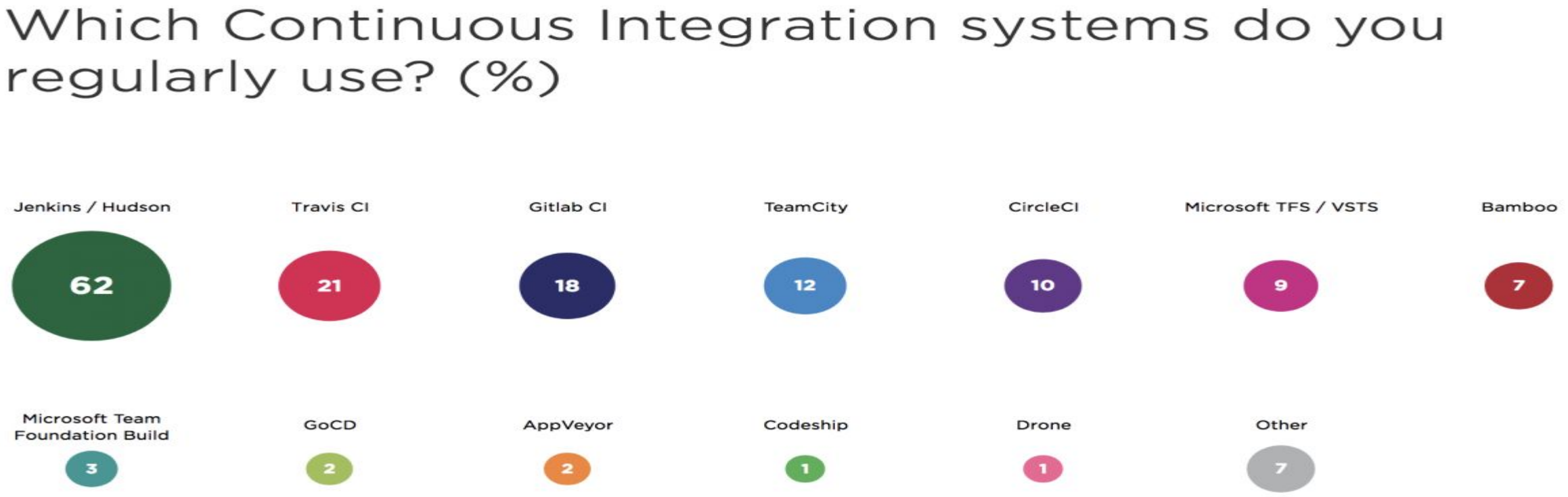
- Self-contained, open source automation server
- Automate all sorts of tasks related to building, testing, and delivering or deploying software
- Continuous integration and continuous delivery tool
- Written in Java
- Forked from Hudson
- Platform independent
- Rich set of plugins(Over 1000 plugins)
- Easily configurable
- Easy to create new Jenkins plugin if one is not available

Benefits

- Developer time is concentrated on work that matters making development faster
- Issues are detected and resolved almost right away which keeps the software in a state - where it can be released at any time safely. Complete history maintained
- Write Pipeline once, use many times
- Deployment made easy
- Improves Software development process
- Almost all sorts of plugins(Github, Gitlab, Bitbucket, etc)

Jenkins Adoption

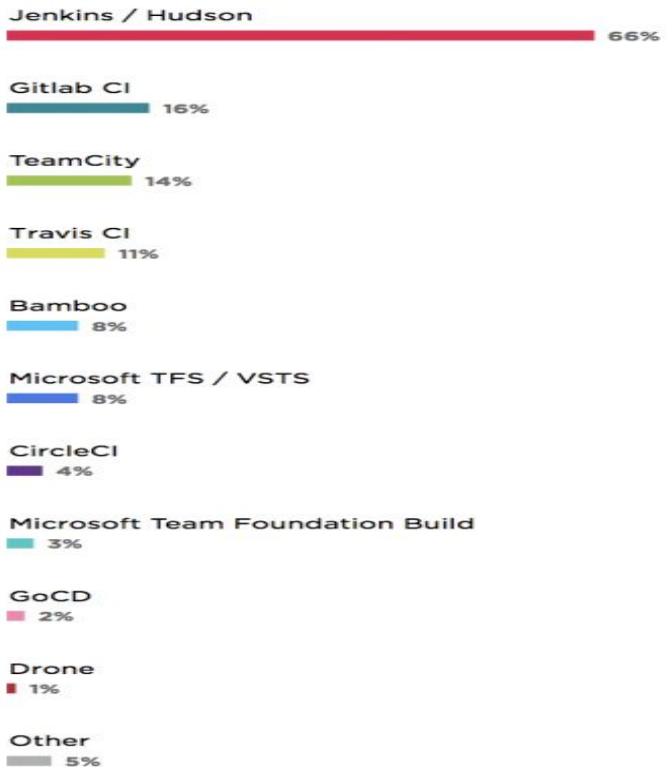
From https://dzone.com/articles/jenkins-is-showing-the-cicd-way



Jenkins Adoption

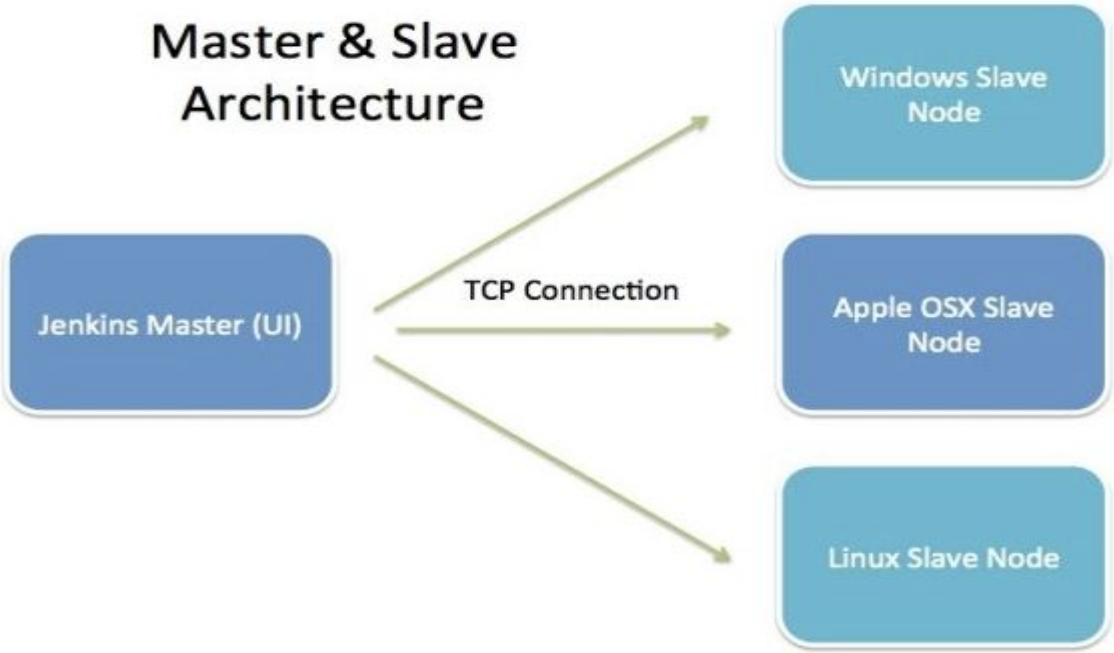
Which Continuous Integration (CI) system(s) do you regularly use?
Shares among people who use **in cloud** CI.

Which Continuous Integration (CI) system(s) do you regularly use?
Shares among people who use **on-premises** CI.



JENKINS ARCHITECTURE

Master and slave architecture
Can have single master and multiple slaves with different environment to test app



JENKINS TERMINOLOGIES

- Job: A unit of work
- Master: central & main unit, Does job scheduling
- Slave: Execute one or more jobs
- Plugin: Common workows merged into one
- View: Different Jobs can be combined in a single view

LAB - INSTALL JENKINS

- Docker

You can create a Jenkins container by running the following command. This will create a container named jenkins.

docker run -d -p 8081:8080 --name jenkins jenkins/jenkins

You can access Jenkins at http://localhost:8081. You can follow the same steps specified in Stand alone application section

LAB - INSTALL JENKINS

- Package

You will install Jenkins through apt by running the following command

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.lis sudo apt-get update
sudo apt-get install jenkins -y
```

Check the status of Jenkins by running the command

```
systemctl status jenkins
```

LAB - INSTALL JENKINS

- Package

If jenkins is not running you can start Jenkins service by running the command

systemctl start jenkins

You can access Jenkins at http://localhost:8080. You can follow the same steps specified in Stand alone application section.

LAB - INSTALL JENKINS

- Stand alone application

Jenkins comes with an embedded Winstone server; therefore enabling it to run as a standalone application. Create directory for jenkins stand alone application

Download Jenkins WAR file from

http://mirrors.jenkins.io/war-stable/latest/jenkins.war

to your local machine on any directory

LAB - INSTALL JENKINS

- Stand alone application

To run Jenkins as stand alone application Java is required. Open terminal and check if Java is installed by run the following command

```
java -version
```

If Java is not installed, you can install Java by running the following command

```
sudo apt-get install openjdk-8-jdk -y
```

LAB - INSTALL JENKINS

Run Jenkins

```
java -jar jenkins.war
```

If your docker runs with `sudo` command, then run

```
sudo java -jar jenkins.war
```

If you want to run Jenkins on any other port, run

```
java -jar jenkins.war --httpPort=8081
```

LAB - INSTALL JENKINS

- Stand alone application

Now you should see the initial setups being run in the command line. The setup will generate a random password for you to use on the initial login. Note this one down.

LAB - INSTALL JENKINS

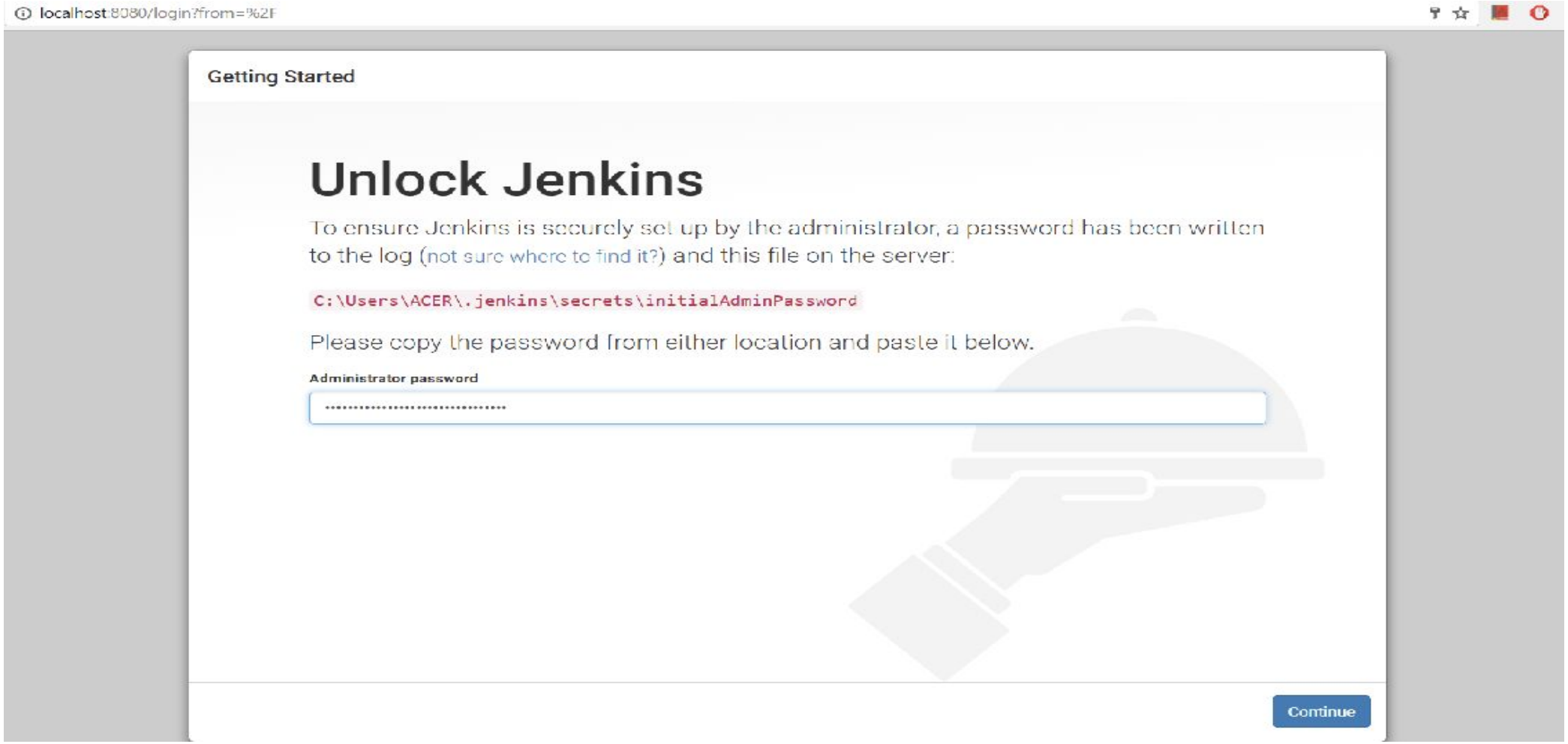
- Stand alone application

```
java -jar jenkins.war
licationContext]; startup date [Fri Apr 14 13:48:31 IST 2017]; root of context hierarchy
Apr 14, 2017 1:48:31 PM org.springframework.context.support.AbstractApplicationContext obtainFreshBeanFactory
INFO: Bean factory for application context [org.springframework.web.context.support.StaticWebApplicationContext@3af9c257
]: org.springframework.beans.factory.support.DefaultListableBeanFactory@30483487
Apr 14, 2017 1:48:31 PM org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiateSingletons
INFO: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@30483487: def
ining beans [filter,legacy]; root of factory hierarchy
Apr 14, 2017 1:48:31 PM jenkins.install.SetupWizard init
INFO:

*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

e415d7bd535f4ba79ac13587fbd10549

This may also be found at: C:\Users\ACER\.jenkins\secrets\initialAdminPassword

*************************************************************
*************************************************************
*************************************************************

Apr 14, 2017 1:48:37 PM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Apr 14, 2017 1:48:37 PM hudson.WebAppMain$3 run
INFO: Jenkins is fully up and running
```

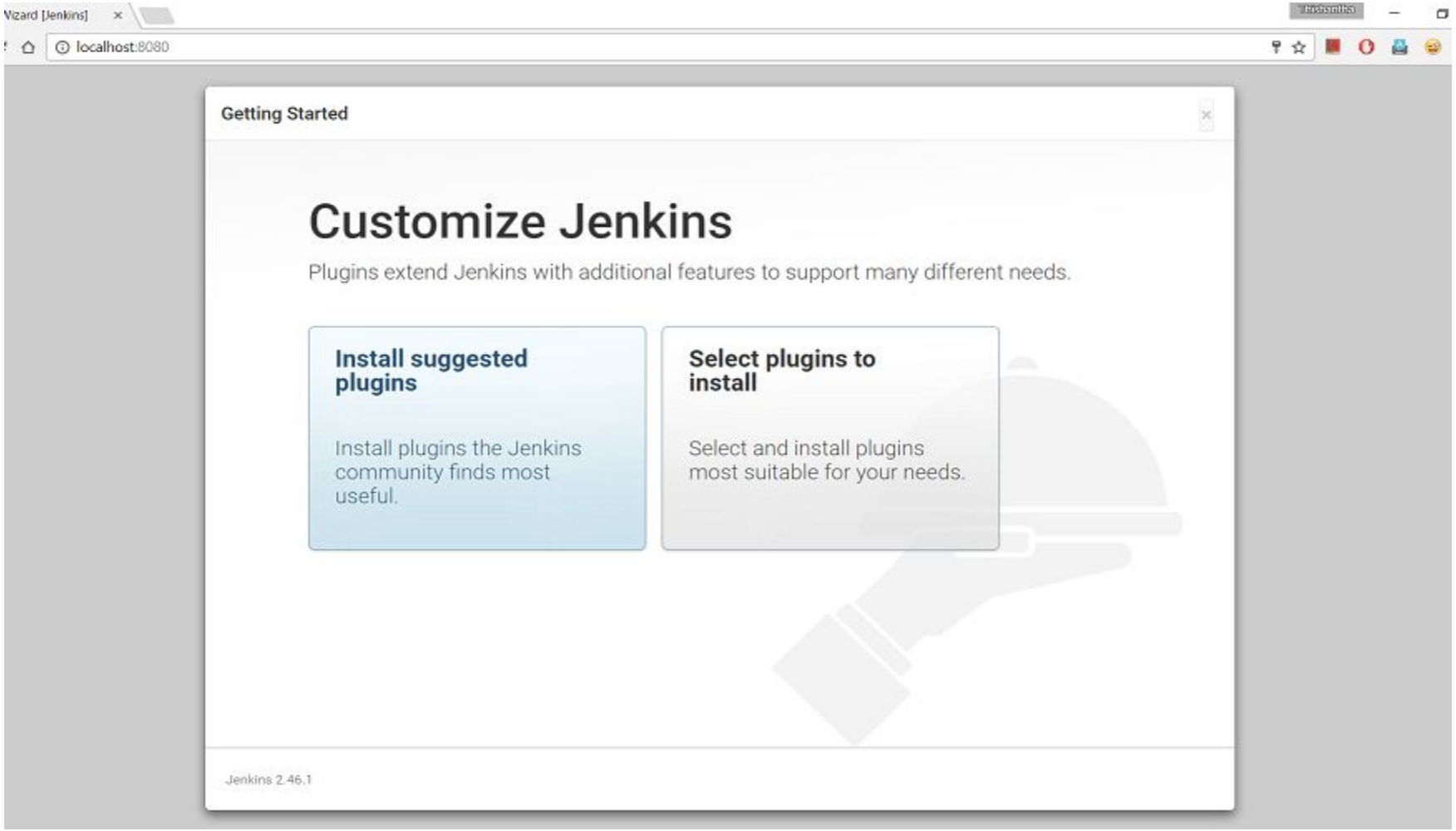
LAB - INSTALL JENKINS

Wait for the console output to print 'Jenkins is fully up and running'. Now go to your browser and type the url: http://localhost:8080/ and you should see a screen as shown below. Enter the initial administrator password that was generated in the previous step. Input the password and click 'Continue'.

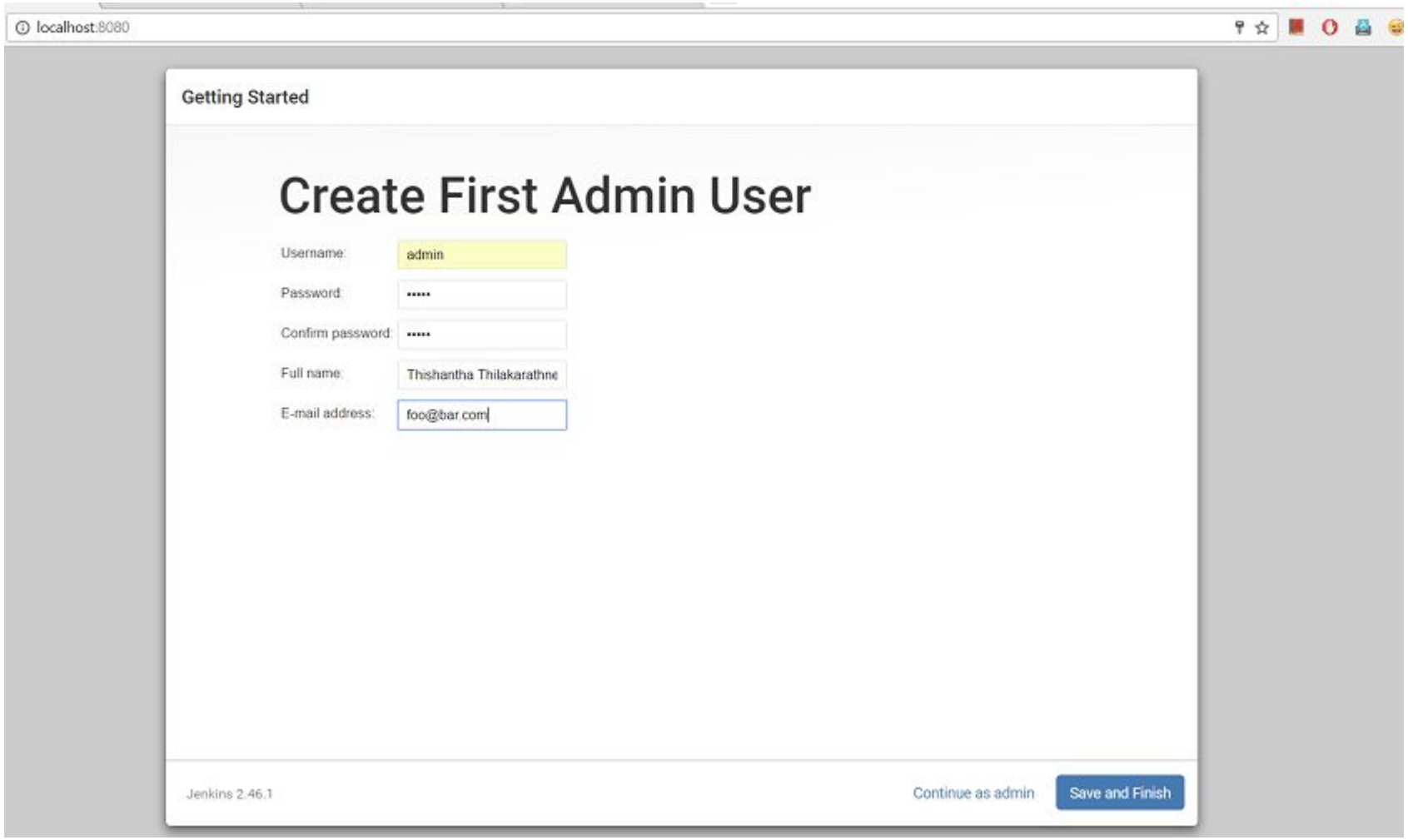
LAB - INSTALL JENKINS



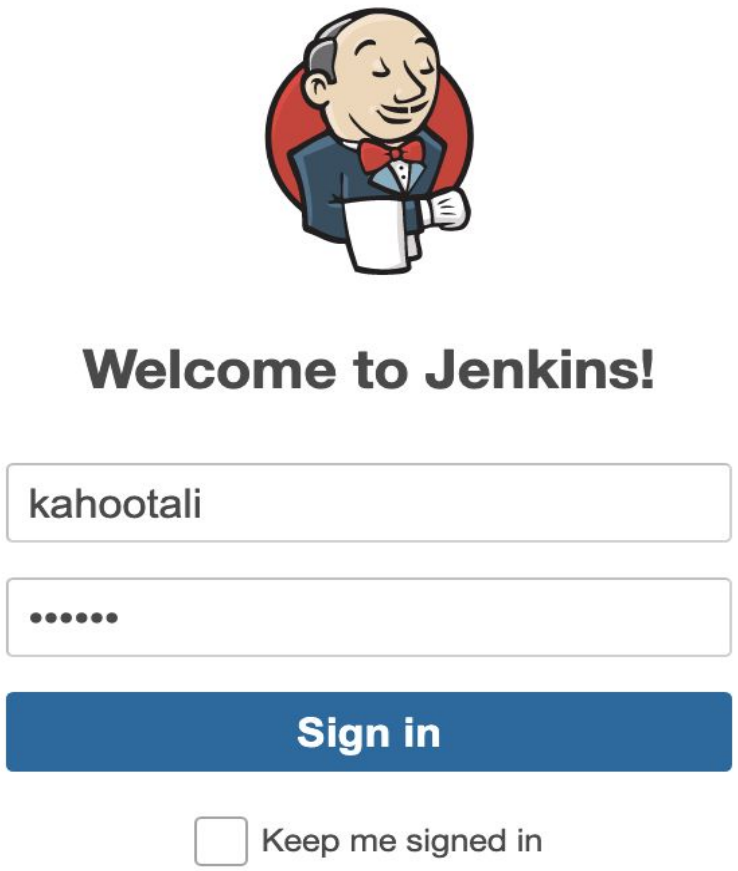
LAB - INSTALL JENKINS



LAB - INSTALL JENKINS



LAB - INSTALL JENKINS



LAB - INSTALL JENKINS

Jenkins

search

1 Ali Kahoot log out

Dashboard

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

add description

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

JENKINS UI

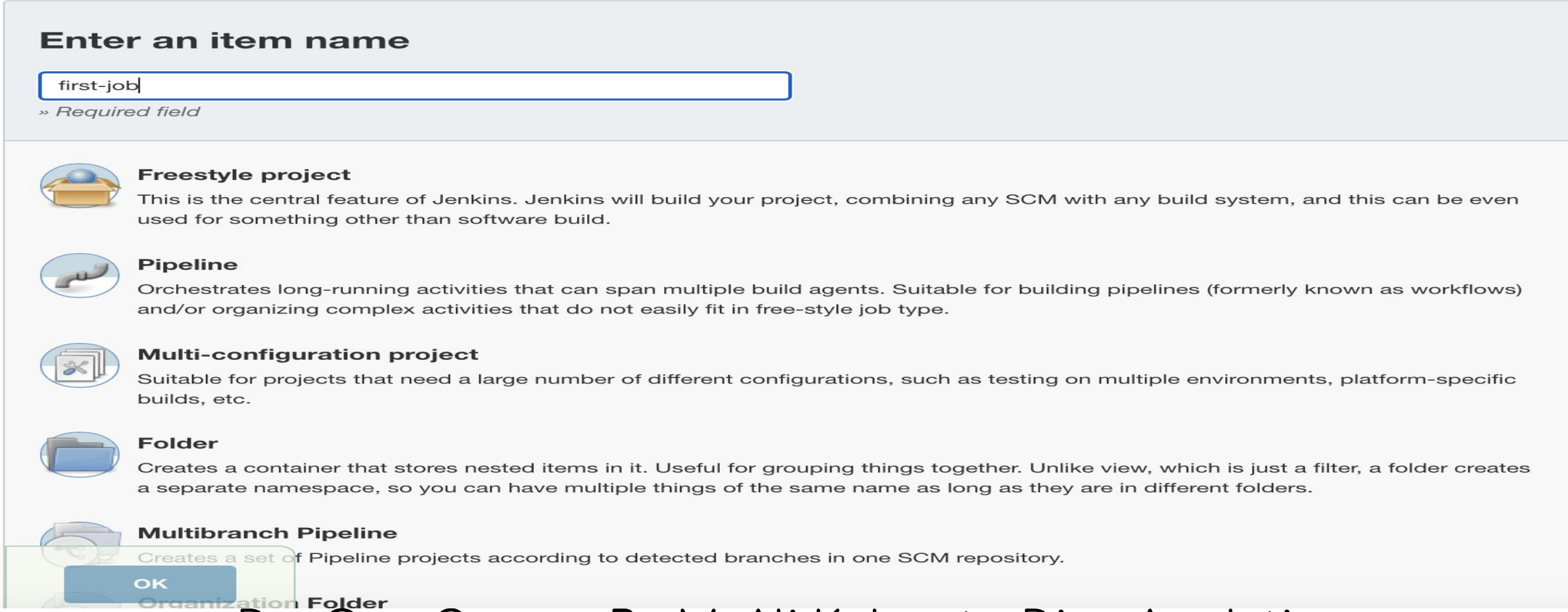
- New Item: To create a new job
- People: Manages users within Jenkins
- Build History: Shows history of builds
- Manage Jenkins: Used to configure Jenkins
- My Views: Private view for ltering Jenkins jobs

LAB - FIRST JENKINS JOB

- Create First Job
- Edit Existing Job
- Run Shell Script
- Parameterized Job

LAB - FIRST JENKINS JOB

- Create First Job: Enter name of your first job i.e. first-job as shown below, select Freestyle Project and click OK



LAB - FIRST JENKINS JOB

General | Source Code Management | **Build Triggers** | Build Environment | Build | Post-build Actions

- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

Build Environment

- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s)
- [ ] Abort the build if it's stuck
- [ ] Add timestamps to the Console Output
- [ ] Inspect build log for published Gradle build scans
- [ ] With Ant

Build

Add build step

- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit

LAB - FIRST JENKINS JOB

General | Source Code Management | Build Triggers | Build Environment | **Build** | Post-build Actions

With Ant

Build

Execute shell

Command

```
echo Hello World
```

See the list of available environment variables

Advanced...

Add build step

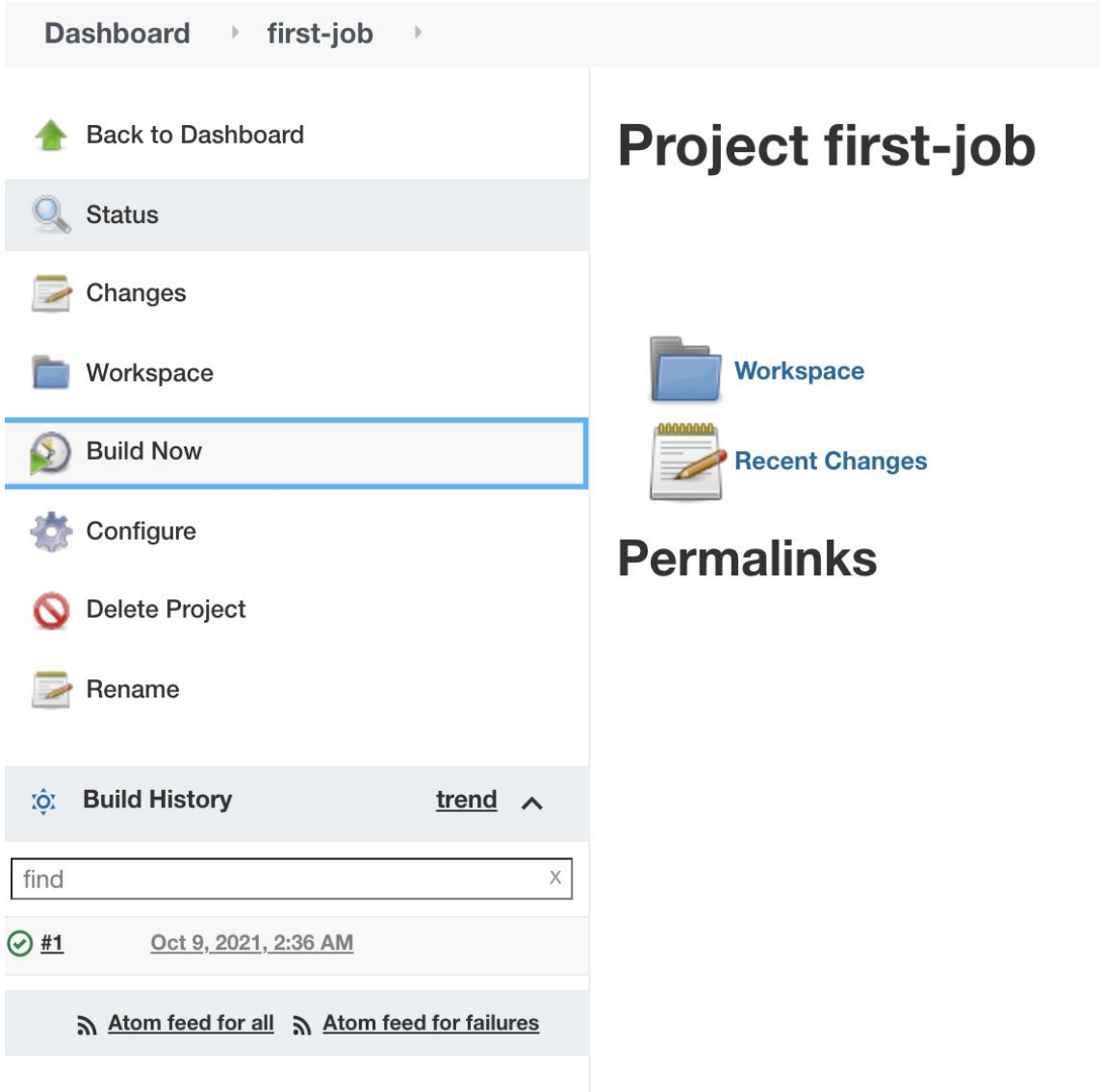
Post-build Actions

Add post-build action

Save | Apply

LAB - FIRST JENKINS JOB

Save the pipeline and run Build Now
Shortly, you will see the build history of your pipeline



/kahootali

LAB - FIRST JENKINS JOB

Click on the history and see Console Output

Dashboard › first-job › #1

- Back to Project
- Status
- Changes
- Console Output
  - View as plain text
- Edit Build Information
- Delete build '#1'

Console Output

```
Started by user Ali Kahoot
Running as SYSTEM
Building in workspace /Users/kahoot/.jenkins/workspace/first-job
[first-job] $ /bin/sh -xe /var/folders/vj/bhztqdtj1176w22njzvfj8k40000gn/T/jenkins10485528840773153113.sh
+ echo Hello World
Hello World
Finished: SUCCESS
```

LAB - EDIT JENKINS JOB

Click on Job -> Configure -> Edit the Job and can change command to any other command and save.

Now Again click Build Now and you will see the Console Output of the 2nd run

LAB - RUN A SCRIPT

You are going to create a job to run a shell script. First created a shell script by running the command

```
# create shell script
    vi jenkins_script.sh
```

Copy following lines

```
#!/bin/sh
echo Hello $1 $2
```

Now make the script executable

```
    chmod u+x jenkins_script.sh
```

LAB - RUN A SCRIPT

Now again create a New Job -> New Item -> Freestyle Project, name it shell-job, and in Build -> Execute Shell
Add

```
FirstName=Ali
LastName=KAhoot
PATH_TO_SCRIPT/jenkins_script.sh $FirstName $LastName
```

LAB - RUN A SCRIPT

Build

Execute shell

Command

```
FirstName=Ali
LastName=KAhoot
/Users/kahoot/Desktop/Training/jenkins_script.sh $FirstName $LastName
```

See **the list of available environment variables**

LAB - RUN A SCRIPT

Build

Execute shell

Command

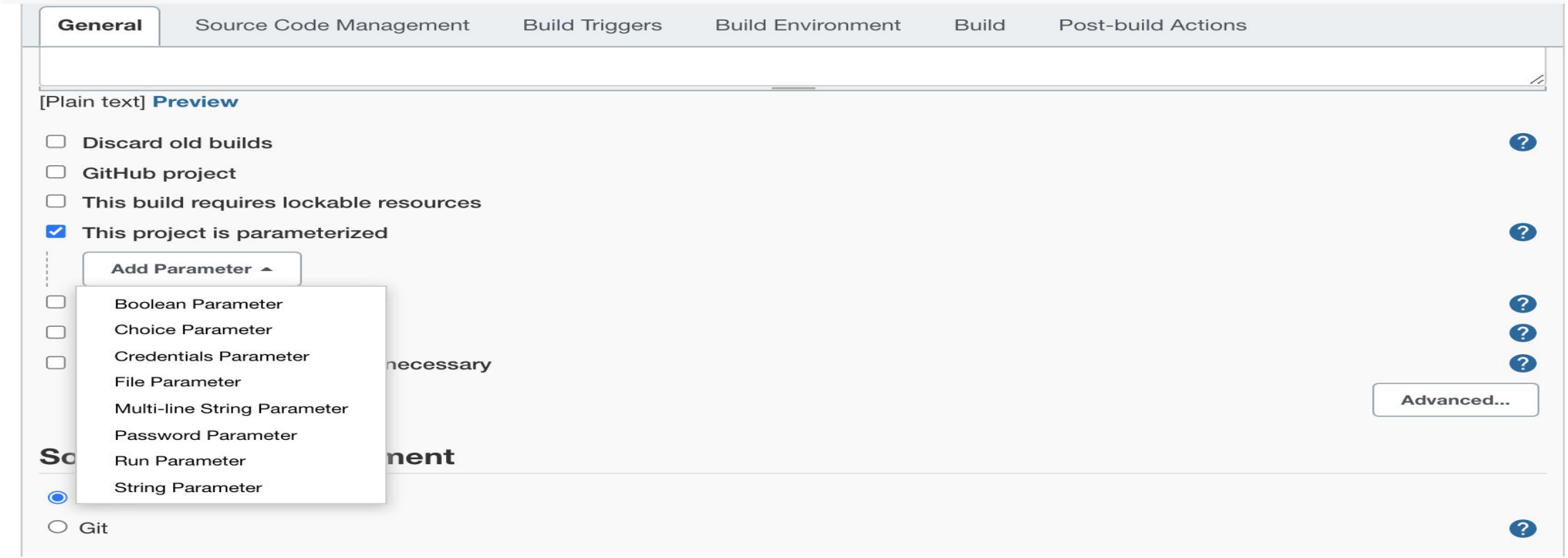
```
FirstName=Ali
LastName=KAhoot
/Users/kahoot/Desktop/Training/jenkins_script.sh $FirstName $LastName
```

See the list of available environment variables

Run the job, and see the Console Output

Passing Parameters to Job

Edit existing Shell Job, Go to Job -> Configure -> Enable This Project is Parameterized



Passing Parameters to Job

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

String Parameter

Name

FIRST_NAME

Default Value

Ali

Description

[Plain text] Preview

☑ Trim the string

String Parameter

Name

LAST_NAME

Default Value

Kahoot

Description

Passing Parameters to Job

And edit the shell to use these Parameters

PATH_TO_SCRIPT/jenkins_script.sh $FIRST_NAME $LAST_NAME

Now Build With Parameters, and you can update the Parameters if needed at runtime

GITHUB INTEGRATION

- Integrate Jenkins with GitHub
- Integration built into the default installation
- Number of plugins for integrating with GitHub
- Trigger jobs
  - Polling
  - Service WebHook
- Status can be recorded back to GitHub
- GitHub Branch Source Plugin already installed through Suggested Plugins

LAB-GITHUB INTEGRATION

Create new Job -> New Item -> Freestyle Project -> Name it **first-github-job**

Under Source Code Management -> Select Git -> Add a Repo URL

https://github.com/kahootali/jenkins-labs.git

We will use this as Repo URL, Its a public repo, so no need of Credentials but if you add to a private repo, you need credentials

Select Branch Specifier

LAB-GITHUB INTEGRATION

Source Code Management

- ○ None
- ◉ Git

Repositories

Repository URL

https://github.com/kahootali/jenkins-labs.git

Credentials

- none - | Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/main

Add Branch

LAB-GITHUB INTEGRATION

Select Build Triggers -> Poll SCM which means Jenkins is going to poll GitHub as per the specified schedule. For this job we want Jenkins to poll GitHub to check for any commits every five minutes

H/5 * * * *

/kahootali

LAB-GITHUB INTEGRATION

Dashboard › first-github-job › #2

- Back to Project
- Status
- Changes
- Console Output
  - View as plain text
- Edit Build Information
- Delete build '#2'
- Polling Log
- Git Build Data
- Previous Build

Console Output

```
Started by an SCM change
Running as SYSTEM
Building in workspace /Users/kahoot/.jenkins/workspace/first-github-job
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /Users/kahoot/.jenkins/workspace/first-github-job/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/kahootali/jenkins-labs.git # timeout=10
Fetching upstream changes from https://github.com/kahootali/jenkins-labs.git
 > git --version # timeout=10
 > git --version # 'git version 2.24.3 (Apple Git-128)'
 > git fetch --tags --force --progress -- https://github.com/kahootali/jenkins-labs.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision aaca7422b6062275904d8d86e40ce2eb27082085 (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f aaca7422b6062275904d8d86e40ce2eb27082085 # timeout=10
Commit message: "Create test-trigger.txt"
 > git rev-list --no-walk 6255d152521d3e6dff61430399a32ead64de7d9d # timeout=10
[first-github-job] $ /bin/sh -xe /var/folders/vj/bhztqdtj1176w22njzvfj8k40000gn/T/jenkins6785342863518537193.sh
+ ls
README.md
test-trigger.txt
Finished: SUCCESS
```

JENKINS JOB DSL

- Plugin that allows you to define jobs in programmatic form
- DSL stands for Domain Specific Language
  - Groovy as scripting language
- Easier to manage jobs
  - Use UI if you have less jobs to maintain
  - When number of jobs grow use DSL
- Easier to restore

JENKINS JOB DSL

- Can use with Version Control
- Job Structure
  - Parameters
  - SCM
  - Triggers
  - Build Steps
  - Post-build actions

JOB DSL

First we are going to install Job DSL plugin. Go to the landing page and Click Manage Jenkins -> Manage Plugins -> Available -> search DSL -> Select Job DSL

- Download Now and Install after Restart
- Select Restart

JOB DSL

Once restarted and create new item -> Freestyle -> Name: seed-job

Build -> Process Job DSLs -> Use Provided DSL Script

Can copy the script from
https://github.com/kahootali/jenkins-labs/blob/main/job-dsl/job-dsl

Save & Build.
You can see a new job will be created “test-job”

JENKINS PIPELINE

- Set of plugins which support implementing and integrating continuous delivery pipelines into Jenkins
- Is a job type to perform sequence of steps
  - Build
  - Test
  - Deploy
- A Pipeline can be created
  - Through UI
  - In SCM
- Types
  - Scripted Pipeline
  - Declarative Pipeline

SCRIPTED PIPELINE

Follows a more imperative programming model built with Groovy.

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

DECLARATIVE PIPELINE

Presents a more simplified and opinionated syntax on top of the Pipeline sub-systems

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps { //..
            }
        }
        stage ('Test') {
            steps {
            //..
            }
        }
    }
}
```

Difference

https://www.edureka.co/community/54705/difference-between-declarative-pipeline-scripted-pipeliine

LAB - FIRST PIPELINE

In this Lab, you are going to create a Jenkins Pipeline Job without any SCM integration. Go to Jenkins landing page by clicking on Jenkins on the left top corner and click New Item -> Pipeline -> set name to first-pipeline-job

Go to Pipeline -> Pipeline Script

LAB - FIRST PIPELINE

```
pipeline {
    agent any

    stages {
        stage('build') {
            steps {
                echo 'Build'
            }
        }
        stage('test') {
            steps {
                echo 'Test'
            }
        }
    }
}
```

LAB - FIRST PIPELINE

Save & Build the Pipeline, and you can see the outputs for each stage, you can see logs for each stage and console output

LAB - FIRST PIPELINE

Dashboard ▸ first-pipeline-job ▸

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

Build History trend

find

#1
Oct 9, 2021, 3:46 AM

Atom feed for all Atom feed for failures

Pipeline first-pipeline-job

Recent Changes

Stage View

	build	test
Average stage times: (Average full run time: ~3s)	364ms	205ms
#1 Oct 09 03:46 No Changes	364ms	205ms

Permalinks

- Last build (#1), 1 min 40 sec ago
- Last stable build (#1), 1 min 40 sec ago
- Last successful build (#1), 1 min 40 sec ago
- Last completed build (#1), 1 min 40 sec ago

JENKINSFILE

- File that contains the definition of Jenkins Pipeline and checked into source control
- Pipeline as code
- Benefits
  - Version control
  - Code review on the Pipeline
  - Audit trail for the Pipeline
  - Single source of truth for the Pipeline

LAB - Jenkinsfile

Now edit the Previous Pipeline -> Click on Pipeline -> Configure -> Pipeline -> Chose Pipeline Script from SCM

In Build Trigger, chose Poll SCM -> schedule: **H/5 * * * ***
In SCM, chose Git, in Repo URL, add

https://github.com/kahootali/jenkins-labs.git

Update Branch specifier: */main
Script path would be the file name i.e. Jenkinsfile in our case

LAB - Jenkinsfile

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

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/kahootali/jenkins-labs.git

Credentials

- none - | Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/main

Add Branch

CONTINUOUS INTEGRATION

- Merge code changes from different developers into a central repository
- Automated builds and tests are run
- Key goals are to fail fast and find and address bugs quicker
- Benefits Developers most, Less merge conflicts

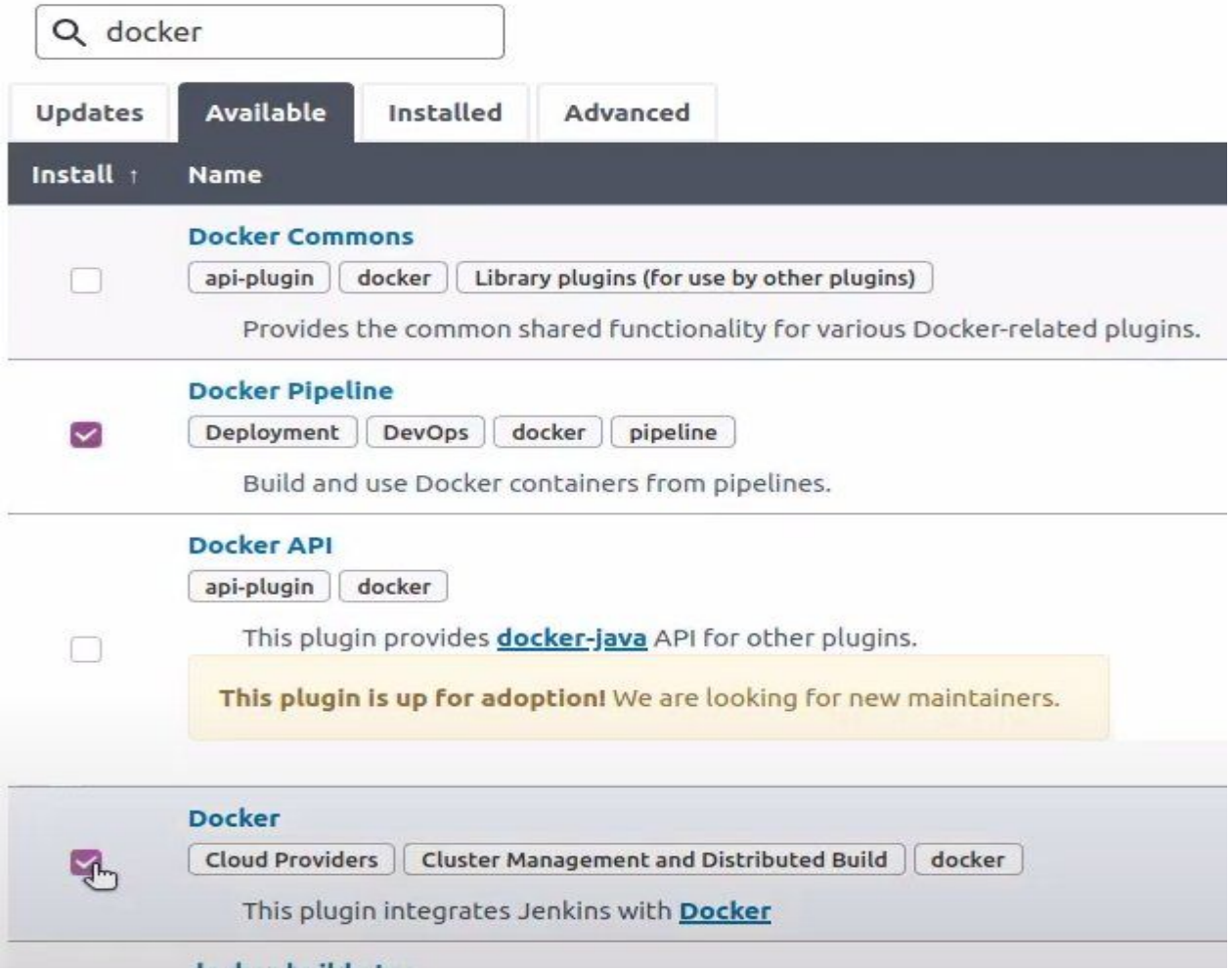
LAB: CONTINUOUS INTEGRATION

In this Lab, we will use the java app and add pipeline to build and test it.
But before that we will need docker plugin to build using docker containers, we will be using containers as runtime agents.

Dashboard -> Manage Jenkins -> Manage Plugins -> Available -> Search Docker -> Select

- Docker Pipeline
- Docker

LAB: CONTINUOUS INTEGRATION



LAB: CONTINUOUS INTEGRATION

New Pipeline -> java-pipeline -> Poll SCM: H/5 * * * *
Pipeline -> Pipeline Script from SCM

Repo URL: https://github.com/kahootali/jenkins-labs.git
Branch Specifier: */main
Script Path: hello-world-java/Jenkinsfile as our file is there

Save & build the pipeline

CONTINUOUS INSPECTION

- Can be used for Continuous Inspection
  - Static Code Analysis
  - Measure Code Quality
  - Identify non-compliant code
  - Fix code quality issue
  - Used to generate reports
- Plugins
  - Warnings Next Generation
  - SonarQube

CONTINUOUS DELIVERY

- CI stage is approved
- A small build cycle for short sprints for releasing small features Code changes are automatically built & tested
- Can be deployed to a test environment
- Can use branching strategy (other than master)
- Mindset to always have a deployment-ready build artifact

LAB - CONTINUOUS DELIVERY

Now we will build Docker Image, test it by running and push the image to Dockerhub

For that we need to add Credentials for Dockerhub

Dashboard -> Manage Jenkins -> Manage Credentials -> Jenkins -> Global Credentials -> Add Credentials
Username:
Password:
ID: dockerhub (This ID we will be specifying in Jenkinsfile)
Description:

/kahootali

LAB - CONTINUOUS DELIVERY

Dashboard › Credentials › System › Global credentials (unrestricted) ›

Back to credential domains

Add Credentials

Kind

Username with password

Scope

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

Username

kahootali

☐ Treat username as secret

Password

••••••••••••••••••••••••••••••••••

ID

dockerhub

Description

Dockerhub Credentials

OK

LAB - CONTINUOUS DELIVERY

Create New Pipeline -> New Item -> Pipeline -> node-pipeline -> Copy properties from previous job (java-pipeline), just need to change Jenkinsfile path

Now it will be “hello-world-nodejs-docker/Jenkinsfile”

LAB - ARTIFACTS

Create New Pipeline -> New Item -> Pipeline -> python-pipeline -> Copy properties from previous job (java-pipeline), just need to change Jenkinsfile path

Now it will be “hello-world-python/Jenkinsfile”

Check the Jenkinsfile, we are creating an artifact for our test results

Post Build Action

Go to Job -> Configure

You can see multiple post build actions

To send status back to Github
https://stackoverflow.com/a/51003334/5113666

To send Slack notification
https://medium.com/appgambit/integrating-jenkins-with-slack-notifications-4f14d1ce9c7a

Github Webhooks

kahootali / jenkins-labs Public

Unwatch 1 | Star 0 | Fork 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

- Options
- Manage access
- Repository roles
- Security & analysis
- Branches
- Webhooks
- Notifications
- Integrations
- Deploy keys

Webhooks

Add webhook

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.

Github Webhooks

We don't have a public url for our Jenkins so we can't add Github Webhooks, but if we had, we could have added as https://JENKINS_URL/github-webhook/

We can use ssh to expose our local port app Jenkins
https://andrewlock.net/using-ssh-and-localhost-run-to-test-github-webhooks-locally/
And then can use Github Webhooks to trigger pipeline
https://www.blazemeter.com/blog/how-to-integrate-your-github-repository-to-your-jenkins-project

Before Next Class

Before next class, install minikube on your host OS from
https://minikube.sigs.k8s.io/docs/start/
and after downloading run
minikube start --driver=virtualbox if you used Virtual Box for Virtualization
minikube start --driver=hyperv if you used Hyper V for Virtualization
minikube start --driver=vmware if you used VMWare for Virtualization

or see your respective driver if you have any other software for Virtualization

https://minikube.sigs.k8s.io/docs/drivers/