Week 12 : SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

Traditional Deployment

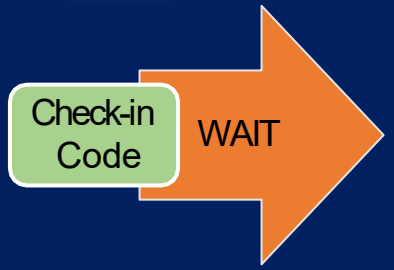
Developer

Operations

Traditional Deployment

Developer

Operations

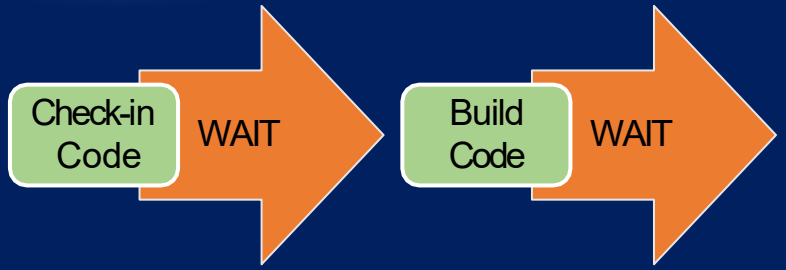


Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

Developer

Operations

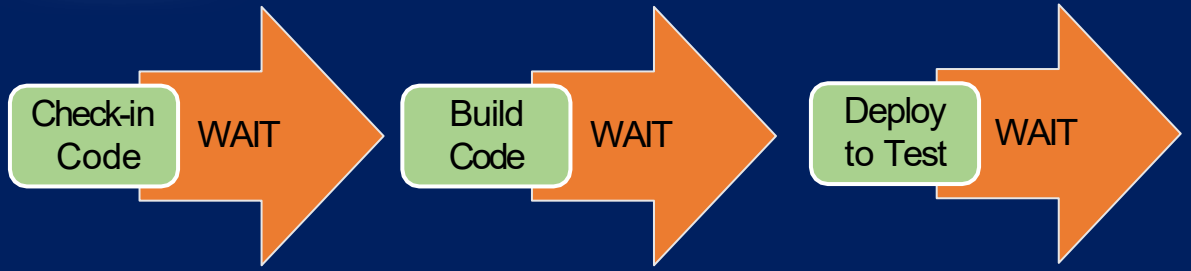


Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

Developer

Operations

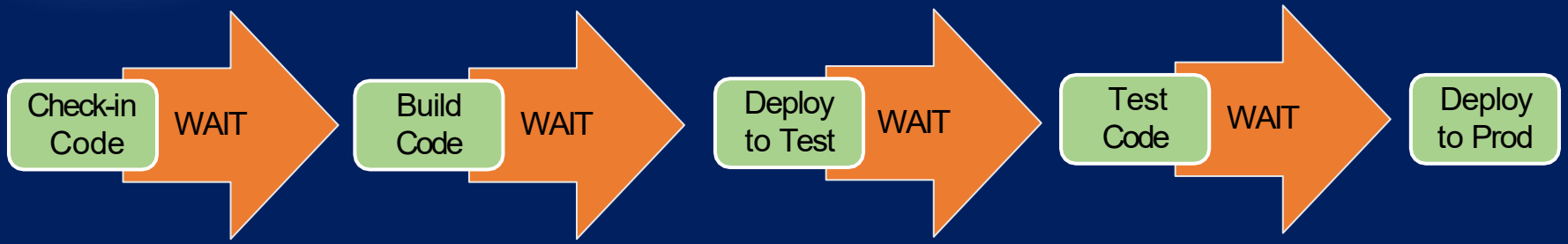


Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

Developer

Operations



Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

When are you gonna deploy my code?

Developer

When you stop breaking my servers

Operations

Check-in Code → WAIT → Build Code → WAIT → Deploy to Test → WAIT → Test Code → WAIT → Deploy to Prod


I just wanna do cool stuff
SYSTEM

What is DevOps?

- Word “DevOps” coined in 2009 by Patrick Debois
- Combination of cultural philosophies, practices, and tools
  - Job market is based on tools!
- Development and Operations teams are no longer “siloed”

ความหมายของแต่ละขั้นในภาพ

ส่วนของ Dev (Development)

1. Plan (**วางแผน**) → กำหนดความต้องการ, คุณสมบัติ, และสิ่งที่จะพัฒนา
2. Code (**เขียนโค้ด**) → นักพัฒนาสร้างซอฟต์แวร์ตามแผนที่วางไว้
3. Build (**สร้างระบบ**) → รวมโค้ด, คอมไพล์, และจัดการ dependencies ให้เป็นแอปพลิเคชัน
4. Test (**ทดสอบ**) → ตรวจสอบความถูกต้อง, คุณภาพ และความปลอดภัยของซอฟต์แวร์

ส่วนของ Ops (Operations)

5. Release (**ปล่อยเวอร์ชัน**) → นำซอฟต์แวร์เข้าสู่สภาพแวดล้อมจริง
6. Deploy (**ติดตั้งใช้งาน**) → ติดตั้งแอปพลิเคชันไปยังเซิร์ฟเวอร์/คลาวด์
7. Operate (**ปฏิบัติการ**) → ดูแลให้ระบบทำงานได้อย่างเสถียร
8. Monitor (**ติดตาม**) → เฝ้าระวัง ประสิทธิภาพ, ปัญหา และ feedback จากผู้ใช้

DevOps

Developer Operations

Same Team

Automated End to End

Check-in Code → Build Code → Deploy to Test → Test Code → Deploy to Prod → Monitor

Rollback to Previous Version

DevOps Phases

AUTHOR → SOURCE → BUILD → TEST → DEPLOY → MONITOR

- Write Code

- Check-in Source Code

- Compile Code
- Create Artifacts
- Unit Testing

- Integration Testing
- Load Testing
- UI Testing
- Penetration Testing

- Deploy Artifacts

- Logs, Metrics, and Traces

Continuous Integration (CI)

Continuous Delivery (CD)

Manual Approval

Continuous Deployment (CD)

🚀 Continuous Integration (CI)

ความหมาย:

- คือกระบวนการที่นักพัฒนาทุกคน **รวมโค้ด (merge) เข้า repository หลักบ่อยครั้ง** (หลายครั้งต่อวัน)
- ทุกครั้งที่ push โค้ดใหม่ ระบบจะ **รันการ build และ test อัตโนมัติ**
- เป้าหมายคือ **จับข้อผิดพลาดให้เร็วที่สุด** ก่อนที่จะสะสมจนแก้ยาก

ประโยชน์:

- ลดปัญหา "works on my machine"
- ทำให้โค้ดของทีมอยู่ในสถานะที่ deploy ได้ตลอดเวลา
- เพิ่มคุณภาพและความมั่นใจในการเปลี่ยนแปลง

ตัวอย่างเครื่องมือ: Jenkins, GitHub Actions, GitLab CI, CircleCI

🚀 Continuous Delivery (CD)

ความหมาย:

- คือการ **ต่อยอดจาก CI** โดยทำให้ซอฟต์แวร์ที่ผ่านการ build และ test แล้ว สามารถ **ถูกปล่อย (release) ได้ตลอดเวลา** ด้วยขั้นตอนอัตโนมัติ
- ทีมสามารถเลือกได้ว่าจะปล่อยจริง (Deploy) หรือยังแค่เตรียม build ที่พร้อม deploy

ประโยชน์:

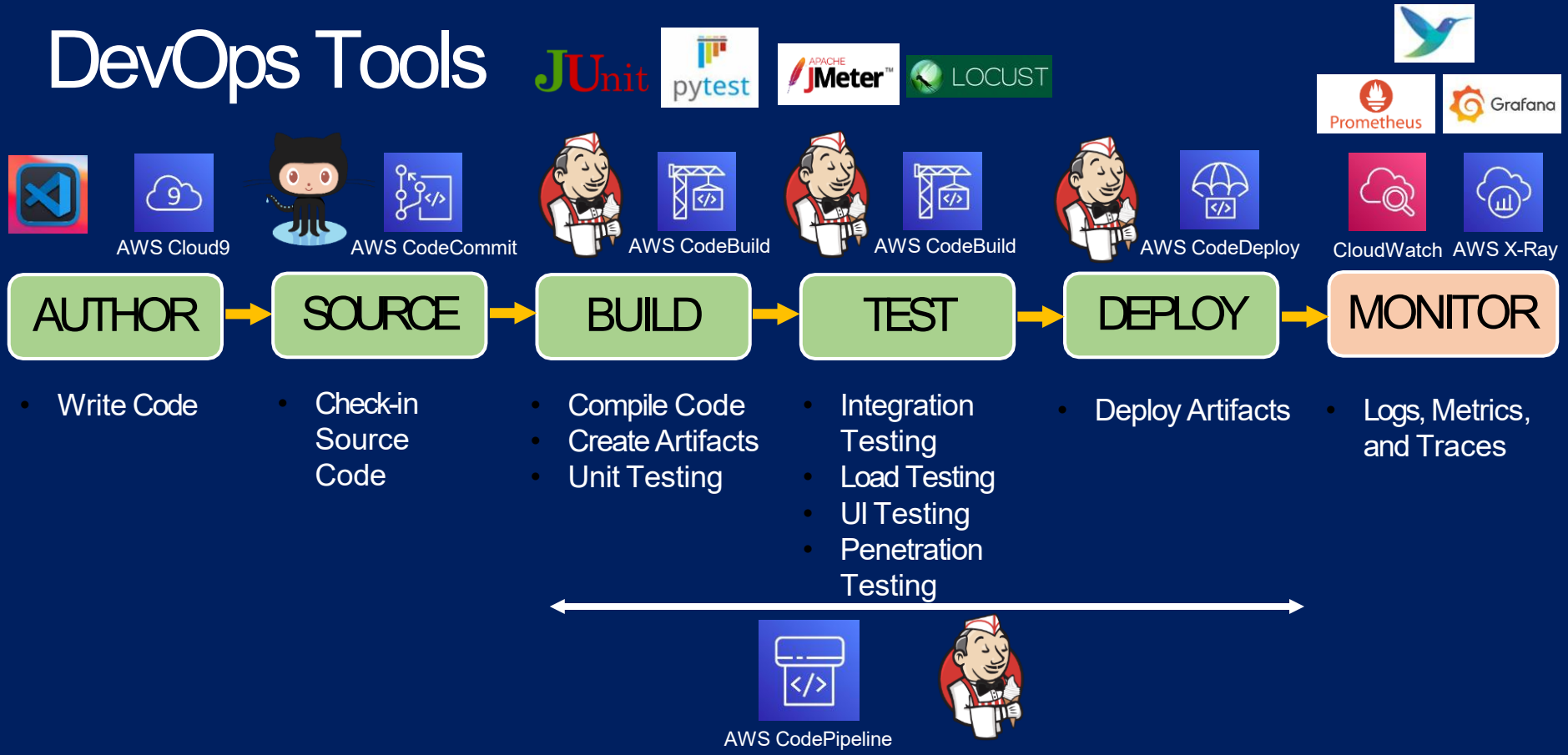
- ส่งมอบซอฟต์แวร์สู่ผู้ใช้ได้อย่างรวดเร็ว
- ลดความเสี่ยงจากการ deploy ครั้งใหญ่ (Big Bang Release)
- ทำงานร่วมกับแนวคิด **Feature Toggle / Canary Release / Blue-Green Deployment**

ตัวอย่างเครื่องมือ: Spinnaker, ArgoCD, Jenkins X

✅ CI vs CD เปรียบเทียบสั้น ๆ

ด้าน	Continuous Integration (CI)	Continuous Delivery (CD)
โฟกัส	รวมโค้ด + ตรวจสอบคุณภาพ	ส่งมอบซอฟต์แวร์พร้อม deploy
Automation	Build + Test	Build + Test + Release Pipeline
ความถี่	หลายครั้งต่อวัน	ตามรอบ release ที่ทีมต้องการ
เป้าหมาย	โค้ดเสถียรและพร้อมใช้งาน	ระบบพร้อมส่งมอบสู่ production ได้ตลอดเวลา

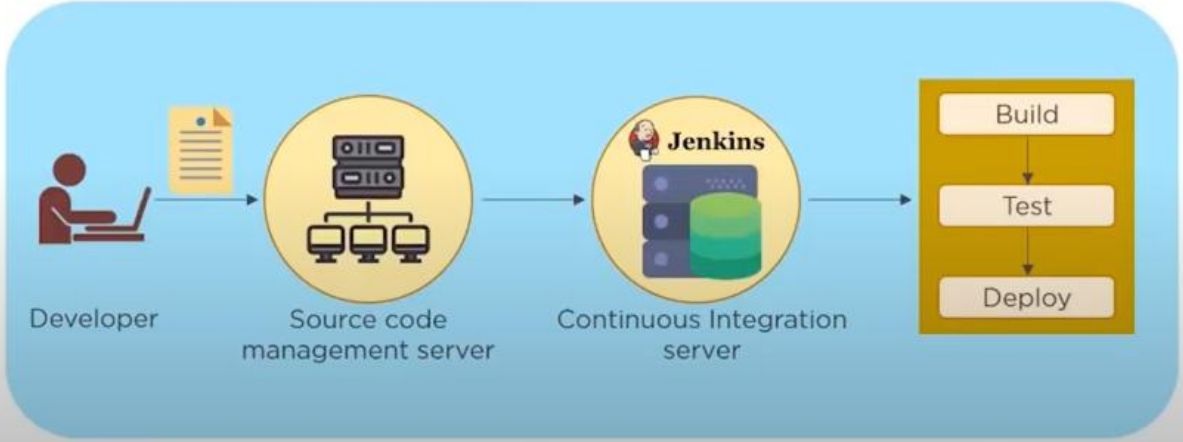
DevOps Tools



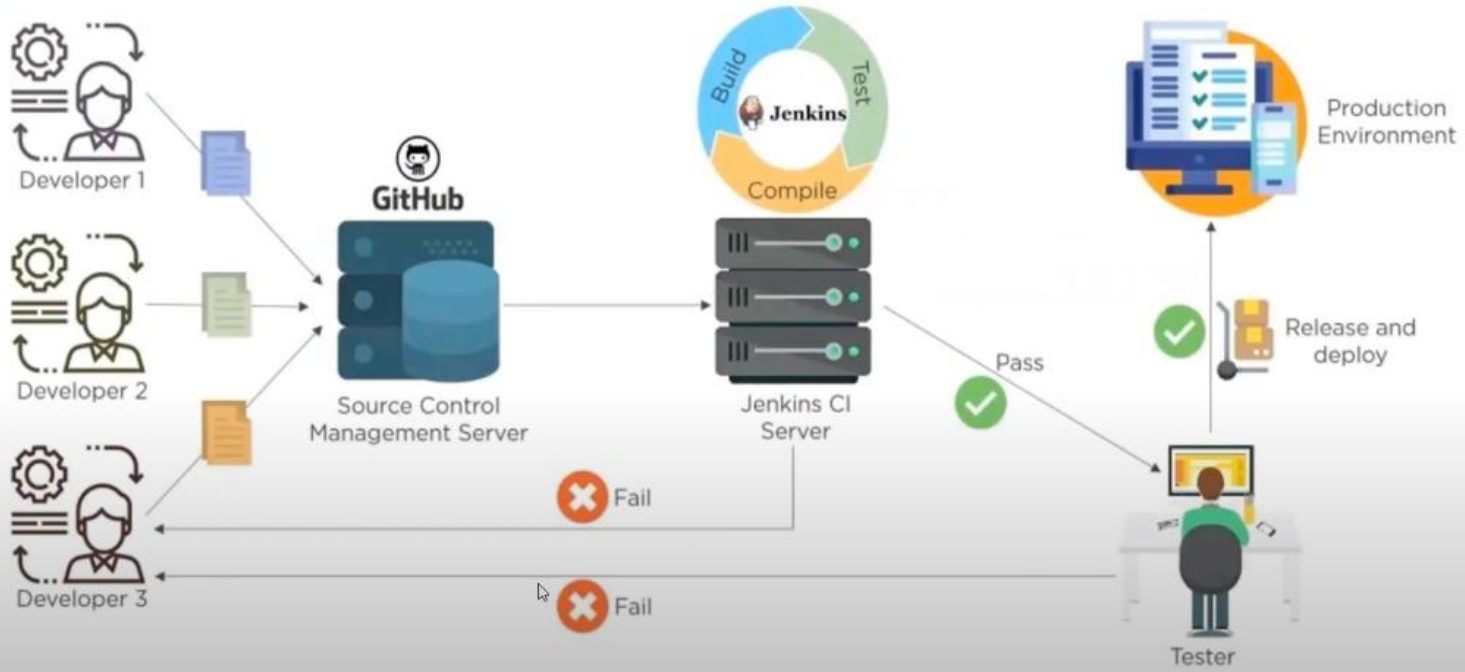
What is Jenkins?

Jenkins

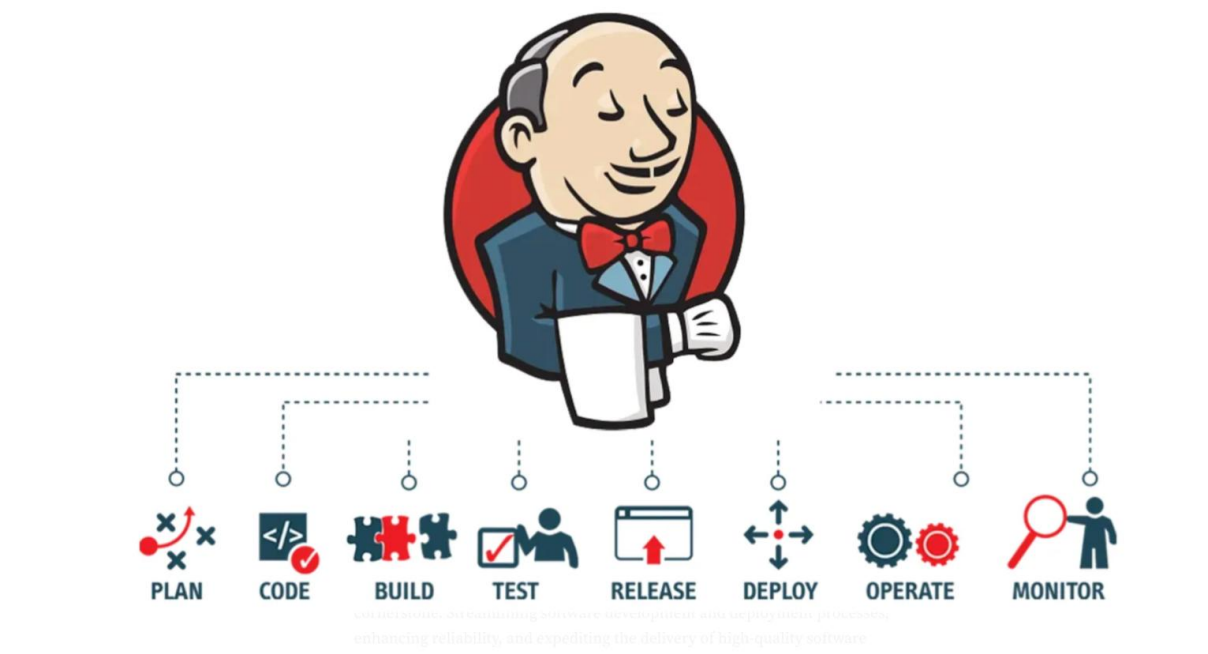
Jenkins is an open source Continuous Integration server written in Java that allows continuous development, test and deployment of codes



CI & CD


Developer 1
Developer 2
Developer 3
GitHub
Source Control
Management Server
Build
Test
Jenkins
Compile
Jenkins CI
Server
Pass
Fail
Fail
Tester
Release and
deploy
Production
Environment

Lab 01:
01_1_Jenkins Installation
01_2_Jenkins Plugin Installation



01_1_Jenkins Installation

Installation Steps

1. **Update System Packages** Update the list of available packages and their versions.

```
sudo apt-get update
```

2. Installation of Java

```
sudo apt update
sudo apt install fontconfig openjdk-17-jre
java -version
openjdk version "17.0.8" 2023-07-18
OpenJDK Runtime Environment (build 17.0.8+7-Debian-1deb12u1)
OpenJDK 64-Bit Server VM (build 17.0.8+7-Debian-1deb12u1, mixed mode, sharing)
```

3. Long Term Support release

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

4. Start Jenkins

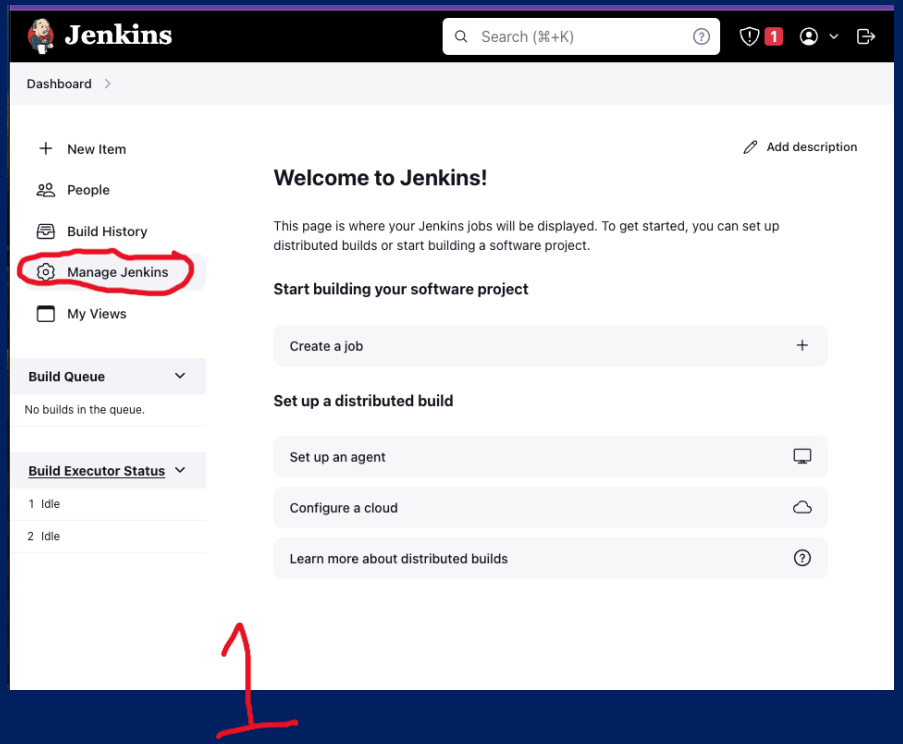
```
sudo systemctl enable jenkins
```

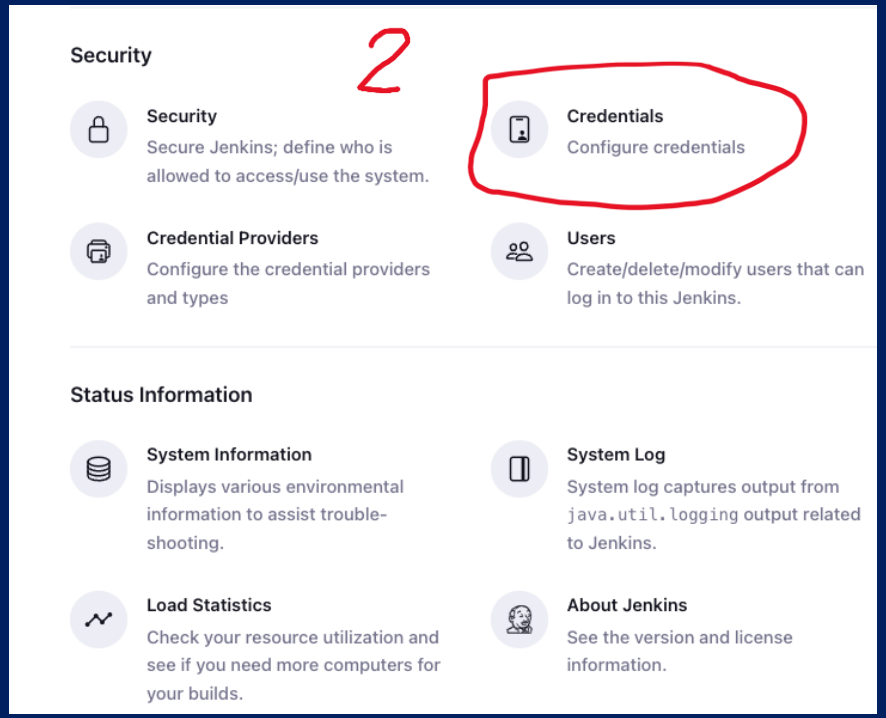
```
sudo systemctl start jenkins
```

```
sudo systemctl status jenkins
```

LAB 02 : Jenkins Add Credentials

1.Add SSH-Credentials





Jenkins

Search (⌘+K)

Dashboard > Manage Jenkins > Credentials

Credentials

T	P	Store ↓	Domain	ID	Name

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

Icon: S M L

3

Jenkins

Search (⌘+K)

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
This credential domain is empty. How about adding some credentials?			

4

Icon: S M L

For SSH

Jenkins

Search (⌘+K)

Tucl ysuwan

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

SSH Username with private key

Scope ?

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

5

ID ?

ssh-prod_instance

Description ?

Username

tuchsanai

Treat username as secret ?

Private Key

Enter directly

Key

Enter New Secret Below

[redacted]
-----END OPENSSH PRIVATE KEY-----

Passphrase

Create

Jenkins

Search (⌘+K)

Tuchsanai Ploysuwan

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
ssh-prod_instance	ssh-prod_instance	SSH Username with private key	

Icon: S M L

6

Compute Engine

Virtual machines

VM instances

Instance templates

Sole-tenant nodes

Machine images

TPUs

Committed use discounts

Reservations

Migrate to Virtual Machin...

Storage

Disks

VM instances

CREATE INSTANCE IMPORT VM REFRESH

INSTANCES OBSERVABILITY INSTANCE SCHEDULES

VM instances

Filter Enter property name or value

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	jenkins	asia-southeast1-a			10.148.0.15 (nic0)	34.126.119.107 (nic0)	SSH
✓	prod-instance	asia-southeast1-b			10.148.0.16 (nic0)	34.143.151.212 (nic0)	SSH

Related actions

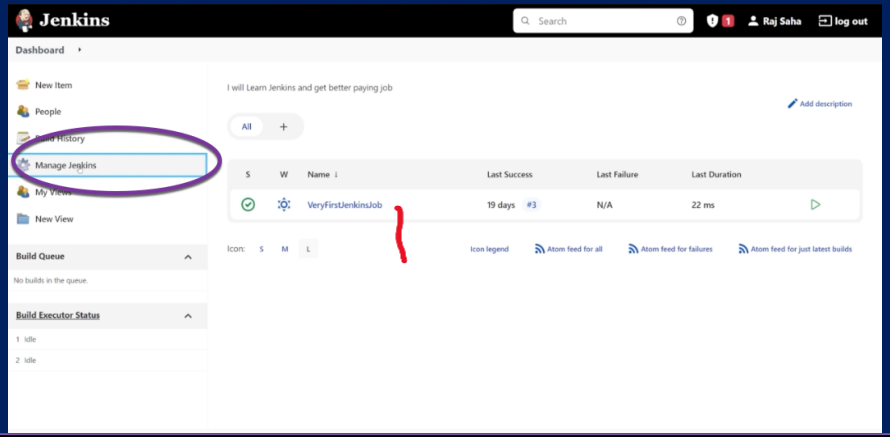
Explore Backup and DR NEW
Back up your VMs and set up disaster recovery

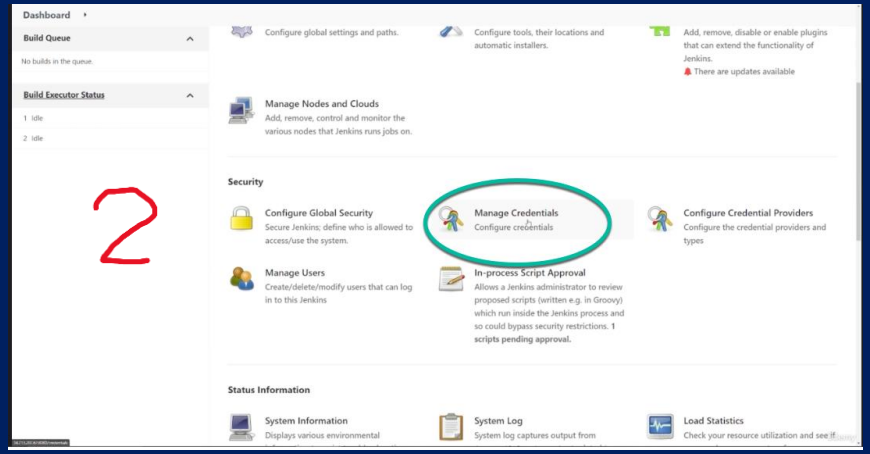
View billing report
View and manage your Compute Engine billing

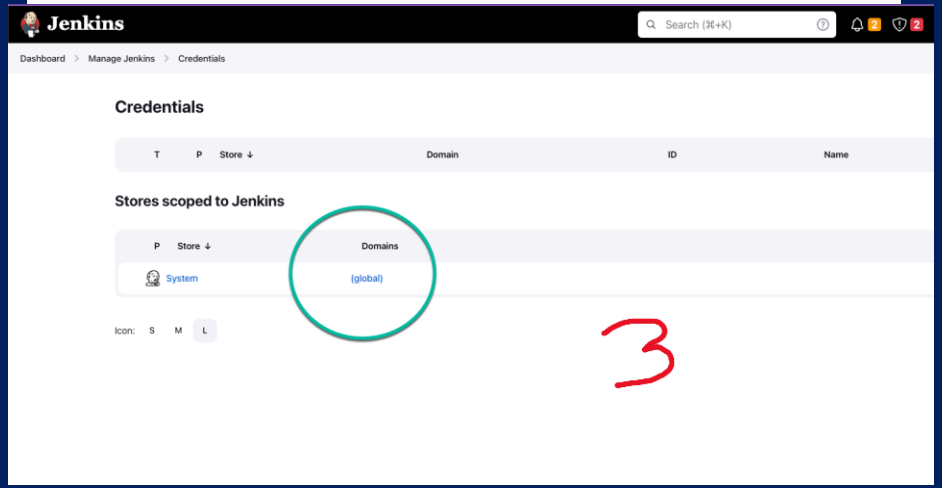
Monitor VMs
View outlier VMs across metrics like CPU and network

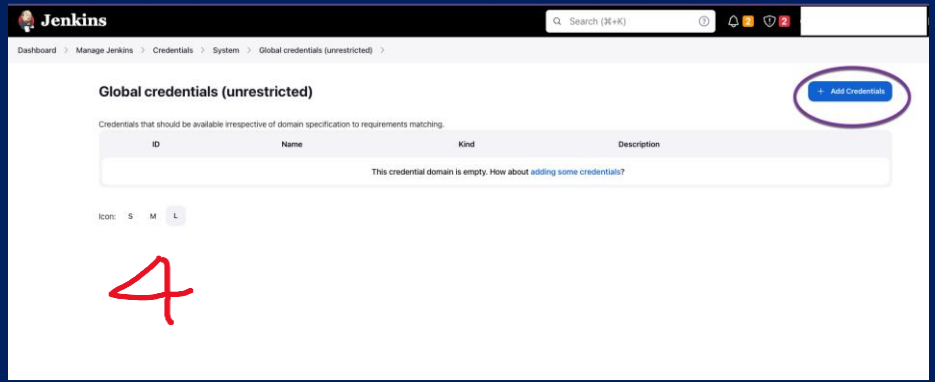
Explore VM logs
View, search, analyze, and download VM instance logs

2.Add Github-Credentials

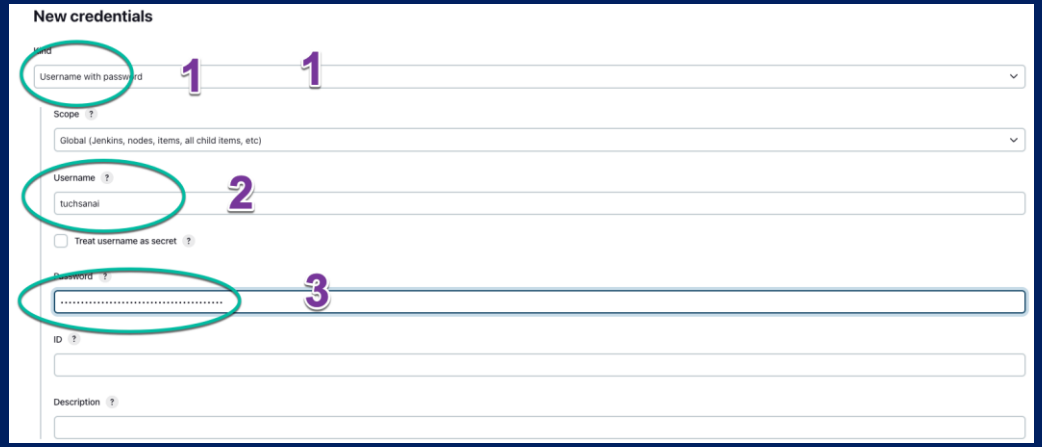




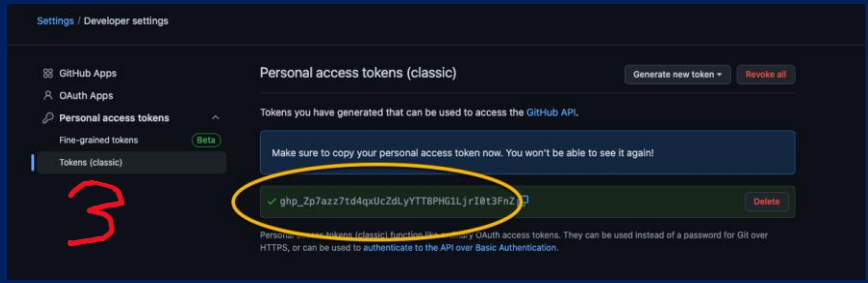


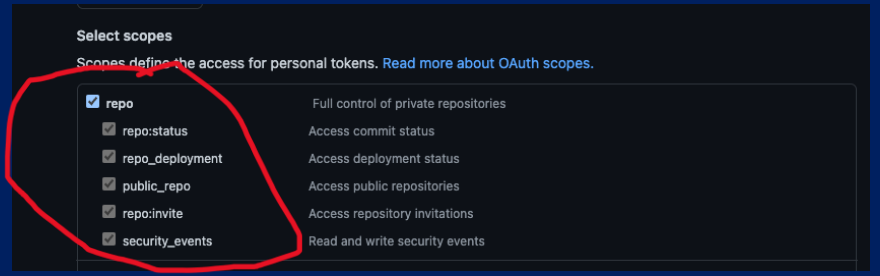


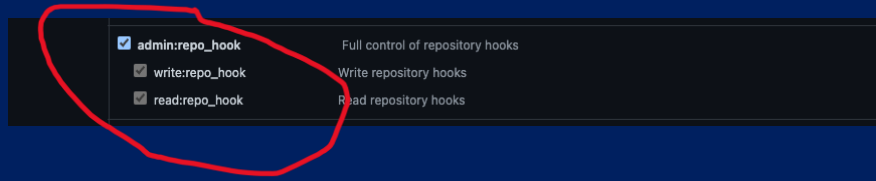
For Github


New credentials
Kind
Username with password
1
1
Scope
Global (Jenkins, nodes, items, all child items, etc)
Username
tuchsanai
2
Treat username as secret
Password
3
ID
Description


4
ID
github
Description
my github for test jenkins


Settings / Developer settings
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Beta
Tokens (classic)
3
Personal access tokens (classic)
Generate new token
Revoke all
Tokens you have generated that can be used to access the GitHub API.
Make sure to copy your personal access token now. You won't be able to see it again!
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
Delete
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.


Select scopes
Scopes define the access for personal tokens. Read more about OAuth scopes.
repo
Full control of private repositories
repo:status
Access commit status
repo_deployment
Access deployment status
public_repo
Access public repositories
repo:invite
Access repository invitations
security_events
Read and write security events


admin:repo_hook
Full control of repository hooks
write:repo_hook
Write repository hooks
read:repo_hook
Read repository hooks

For Github

Jenkins

Search (⌘+K)

2 Tuchsanai Ploysuwan log out

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

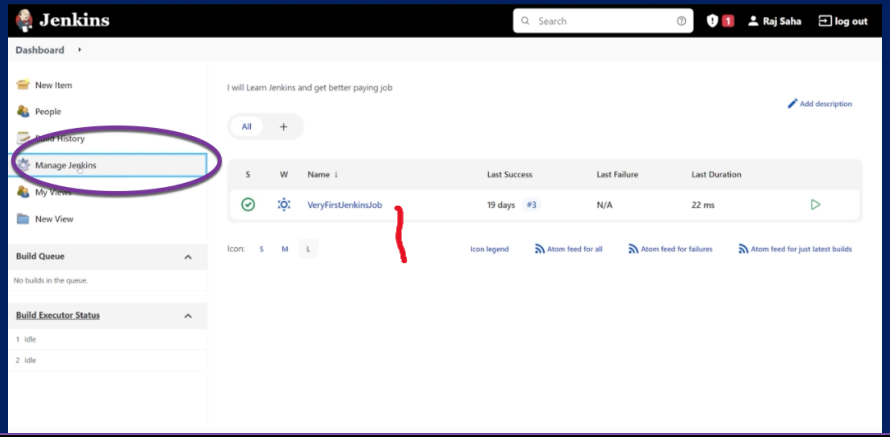
+ Add Credentials

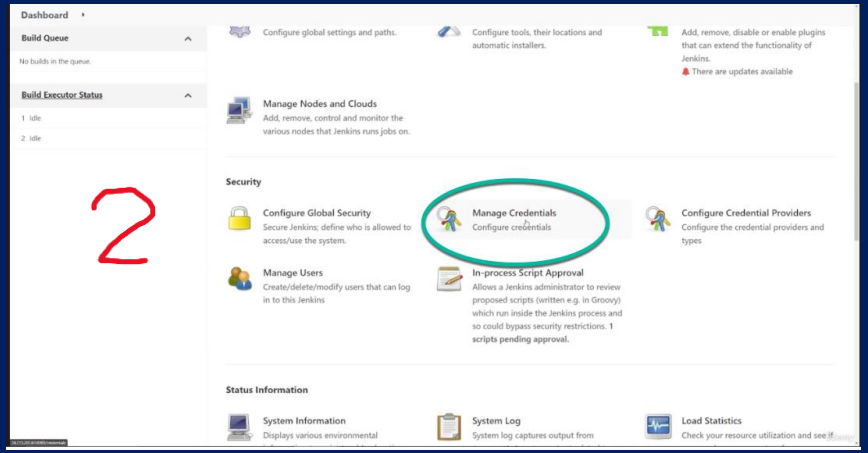
Credentials that should be available irrespective of domain specification to requirements matching.

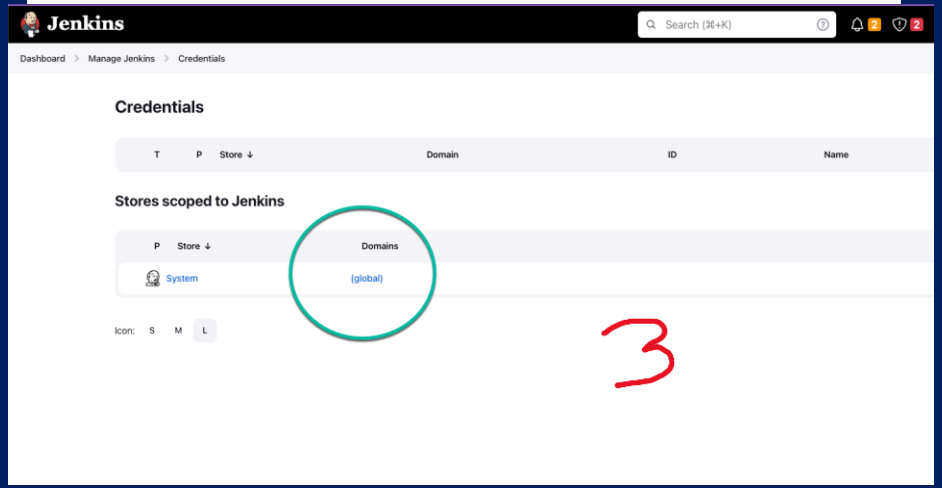
ID	Name	Kind	Description
prod-agent_id	ubuntu	SSH Username with private key	
dockerhub	tuchsanai/******	Username with password	
github	tuchsanai/******	Username with password	

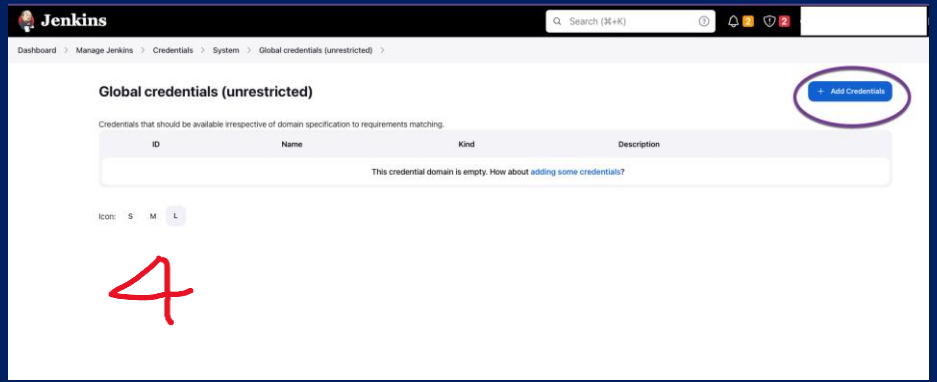
Icon: S M L

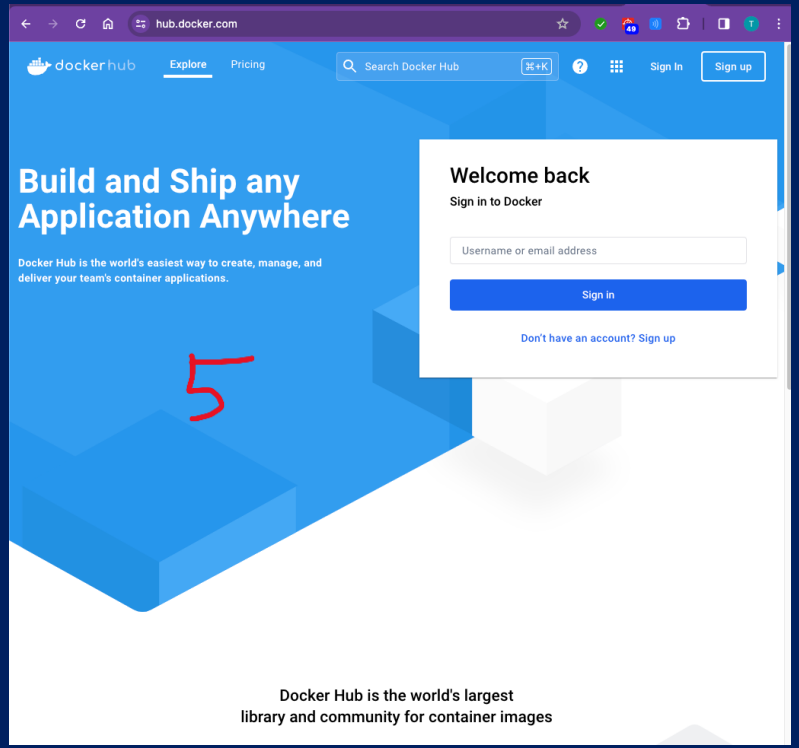
3.Add docker hub Credentials

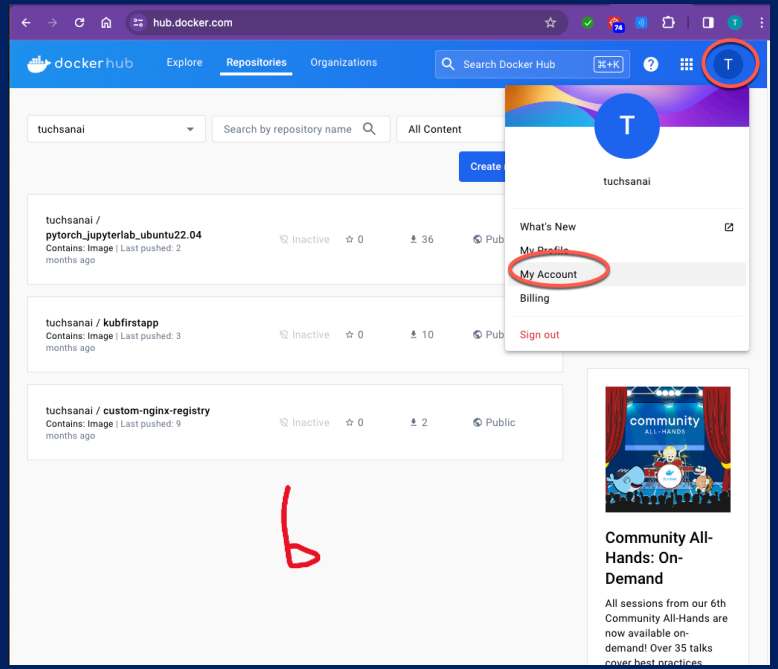









hub.docker.com
dockerhub
Explore
Pricing
Search Docker Hub
Sign In
Sign up
Build and Ship any Application Anywhere
Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications.
Welcome back
Sign in to Docker
Username or email address
Sign in
Don't have an account? Sign up
5
Docker Hub is the world's largest library and community for container images


hub.docker.com
dockerhub
Explore
Repositories
Organizations
Search Docker Hub
tuchsanai
Search by repository name
All Content
Create
tuchsanai / pytorch_jupyterlab_ubuntu22.04
Contains: Image | Last pushed: 2 months ago
Inactive
0
36
tuchsanai / kubfirstapp
Contains: Image | Last pushed: 3 months ago
Inactive
0
10
tuchsanai / custom-nginx-registry
Contains: Image | Last pushed: 9 months ago
Inactive
0
2
Public
T
tuchsanai
What's New
My Account
Billing
Sign out
6
community
ALL-HANDS
Community All-Hands: On-Demand
All sessions from our 6th Community All-Hands are now available on-demand! Over 35 talks cover best practices

hub.docker.com/settings/security

dockerhub Explore Repositories Organizations Search Docker Hub ⌘+K

Account Settings / Security

tuchsanai

User Joined April 4, 2022

7

General
Security
Default Privacy
Notifications
Convert Account
Deactivate Account

Access Tokens

It looks like you have not created any access tokens.
Docker Hub lets you create tokens to authenticate access. Treat personal access tokens as alternatives to your password. Learn more

New Access Token

Two-Factor Authentication

Two-factor authentication is not enabled yet.
Two-factor authentication adds an extra layer of security to your account by requiring more than just a password to sign in. Learn more

Enable Two-Factor Authentication

Copy Access Token

When logging in from your Docker CLI client, use this token as a password. Learn more

ACCESS TOKEN DESCRIPTION

a

ACCESS PERMISSIONS

Read, Write, Delete

8

To use the access token from your Docker CLI client:

1. Run `docker login -u` [redacted]
2. At the password prompt, enter the personal access token.

```
dckr_pat_tgAISGZbNSnN5LuR4ckSI6Zb6c8
```

WARNING: This access token will only be displayed once. It will not be stored and cannot be retrieved. Please be sure to save it now.

Copy and Close

dockerhub Explore Repositories Organizations Search Docker Hub ⌘+K

Account Settings / Security

9

tuchsanai

User Joined April 4, 2022

General
Security
Default Privacy
Notifications
Convert Account
Deactivate Account

Access Tokens

New Access Token

Description	Source	Scope	Last Used	Created
mydockerhub	MANUAL	Read, Write, Delete	Never	Jan 04, 2024 20:48:55

Two-Factor Authentication

Two-factor authentication is not enabled yet.
Two-factor authentication adds an extra layer of security to your account by requiring more than just a password to sign in. Learn more

Enable Two-Factor Authentication

Jenkins | Search (⌘+K) | 2 | Tuchsanai Ploysuwan | log out

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

Username with password

Scope ?

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

Username ?

tuchsanai

Treat username as secret ?

Password ?

••••••••••••••••••••••••••••••••••

ID ?

dockerhub

Description ?

Create

Jenkins | Search (⌘+K) | 2 | Tuchsanai Ploysuwan | log out

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

	ID	Name	Kind	Description	
	prod-agent_id	ubuntu	SSH Username with private key		
	dockerhub	tuchsanai/******	Username with password		

Icon: S M L

LAB 3 : First Jenkins

Jenkins

Search (⌘+K)

1 1 e log out

Dashboard >

1

- New Item
- People
- Build History
- Manage Jenkins
- My Views

Build Queue

No builds in the queue.

Build Executor Status

Built-In Node (offline)

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

Search (⌘+K)

1 Tuchsanai Ploysuwa

Dashboard > All >

Enter an item name

first_project

» Required field

2

Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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.

3

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.

Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder

Creates a set of multibranch project subfolders by scanning for repositories.

Dashboard > first_project > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Throttle builds ?

Build Triggers

- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?
- Quiet period ?
- Trigger builds remotely (e.g., from scripts) ?

Advanced Project Options

Advanced

4

Pipeline

Definition

Pipeline script

Script ?

try sample Pipeline...

```
pipeline {
    agent any  // Execute on any available Jenkins agent

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

Use Groovy Sandbox ?

https://github.com/Tuchsanai/DevTools/tree/main/03_Jenkins/03_First_Jenkins

Name	Last commit message	Last commit date
..		
readme.md	ss	28 minutes ago

readme.md

First Jenkins Pipeline

```
pipeline {
    agent any  // Execute on any available Jenkins agent

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

Jenkins

Search (⌘+K)

1 Tuchsanai Ploysuwan log out

Dashboard > first_project >

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

5

Build History trend

No builds

Atom feed for all Atom feed for failures

first_project

Add description

Disable Project

Stage View

No data available. This Pipeline has not yet run.

Permalinks

Jenkins

Search (⌘+K)

1 Tuchsanai Ploysuwan log out

Dashboard > first_project >

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

Build History trend

Filter...

#1 23 ก.พ. 2024 13:05

Atom feed for all Atom feed for failures

first_project

Add description

Disable Project

Stage View

	Hello World
Average stage times: (Average full run time: ~2s)	191ms
#1 ก.พ. 23 20:05 No Changes	191ms

Permalinks

6

Jenkins

Search (⌘+K)

1 Tuchsanai Ploysuwan log out

Dashboard > first_project > #1

- Status
- Changes
- Console Output
  - View as plain text
- Edit Build Information
- Delete build '#1'
- Restart from Stage
- Replay
- Pipeline Steps
- Workspaces

7

Console Output

```
Started by user Tuchsanai Ploysuwan
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/first_project
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello World)
[Pipeline] echo
Hello World!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

LAB 4 : Jenkins Pipeline and Environment

Click: Go to LAB

Declarative Pipeline fundamentals

In Declarative Pipeline syntax, the `pipeline` block defines all the work done throughout your entire Pipeline.

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent any ❶
    stages {
        stage('Build') { ❷
            steps {
                // ❸
            }
        }
        stage('Test') { ❹
            steps {
                // ❺
            }
        }
        stage('Deploy') { ❻
            steps {
                // ❼
            }
        }
    }
}
```

1. Execute this Pipeline or any of its stages, on any available agent.
2. Defines the "Build" stage.
3. Perform some steps related to the "Build" stage.
4. Defines the "Test" stage.
5. Perform some steps related to the "Test" stage.
6. Defines the "Deploy" stage.
7. Perform some steps related to the "Deploy" stage.

Jenkins pipeline with write in Groovy language

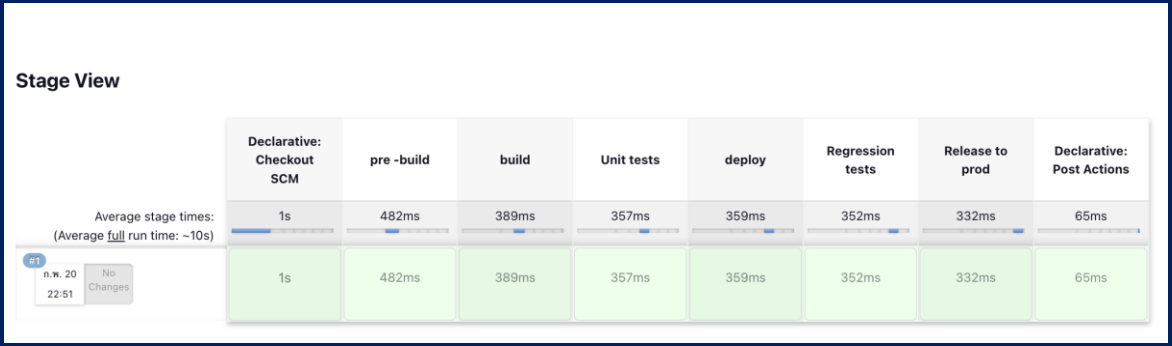
https://www.jenkins.io/doc/book/pipeline/

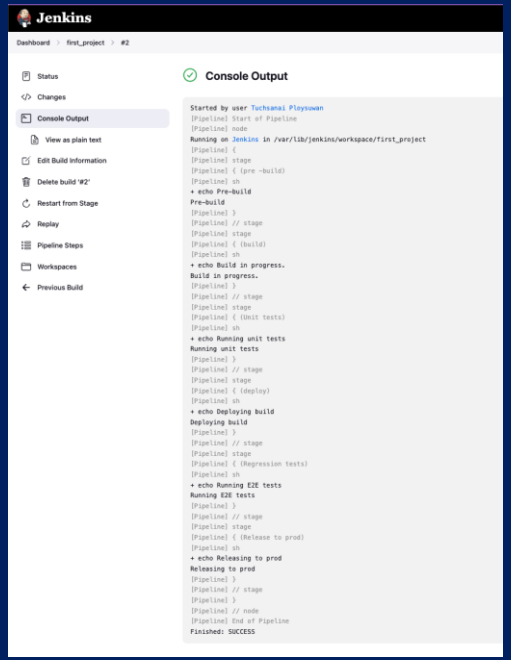
https://www.jenkins.io/doc/pipeline/steps/workflow-cps/#pipeline-groovy

LAB 4.1: Pipeline with Multiple Stages

Example 1. Multiple Stages

```
pipeline {
    agent any
    stages {
        stage('pre -build') {
            steps {
                sh 'echo Pre-build'
            }
        }
        stage('build') {
            steps {
                sh 'echo Build in progress.'
            }
        }
        stage('Unit tests') {
            steps {
                sh 'echo Running unit tests'
            }
        }
        stage('deploy') {
            steps {
                sh 'echo Deploying build'
            }
        }
        stage('Regression tests') {
            steps {
                sh 'echo Running E2E tests'
            }
        }
        stage('Release to prod') {
            steps {
                sh 'echo Releasing to prod'
            }
        }
    }

}
```





[Click: Go to LAB]

LAB 4.2: Pipeline with post actions

Pipeline with post actions

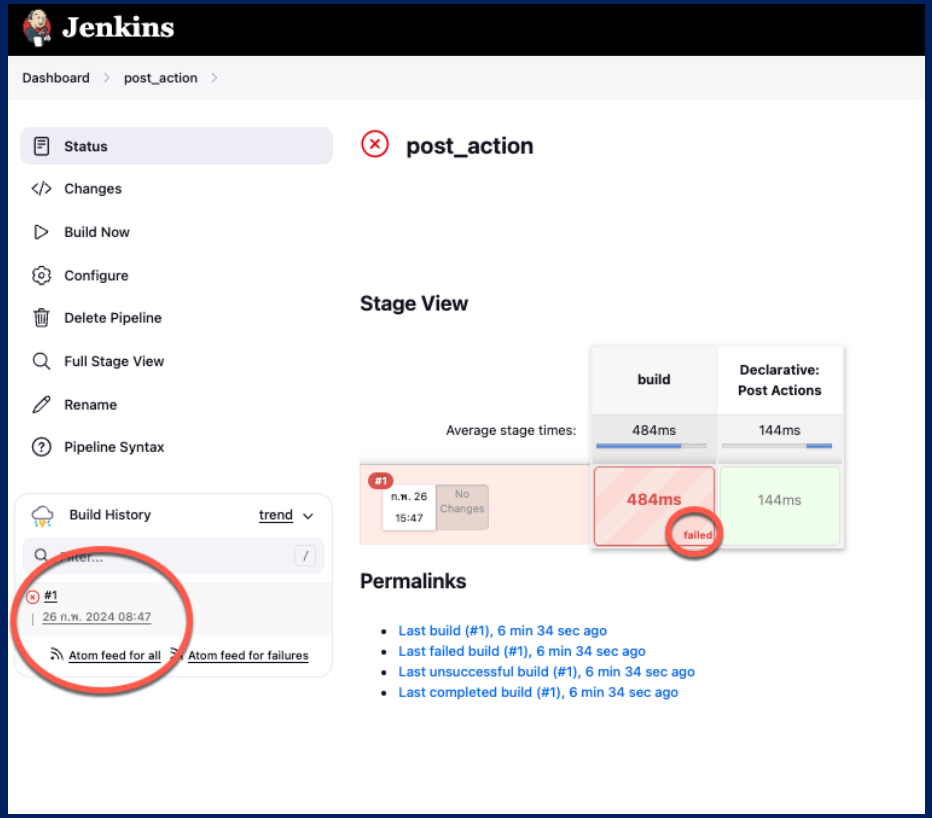
- Run additional steps at the end of pipeline or stage
- Not a required field
- Used to handle failure conditions
- Supports different conditions

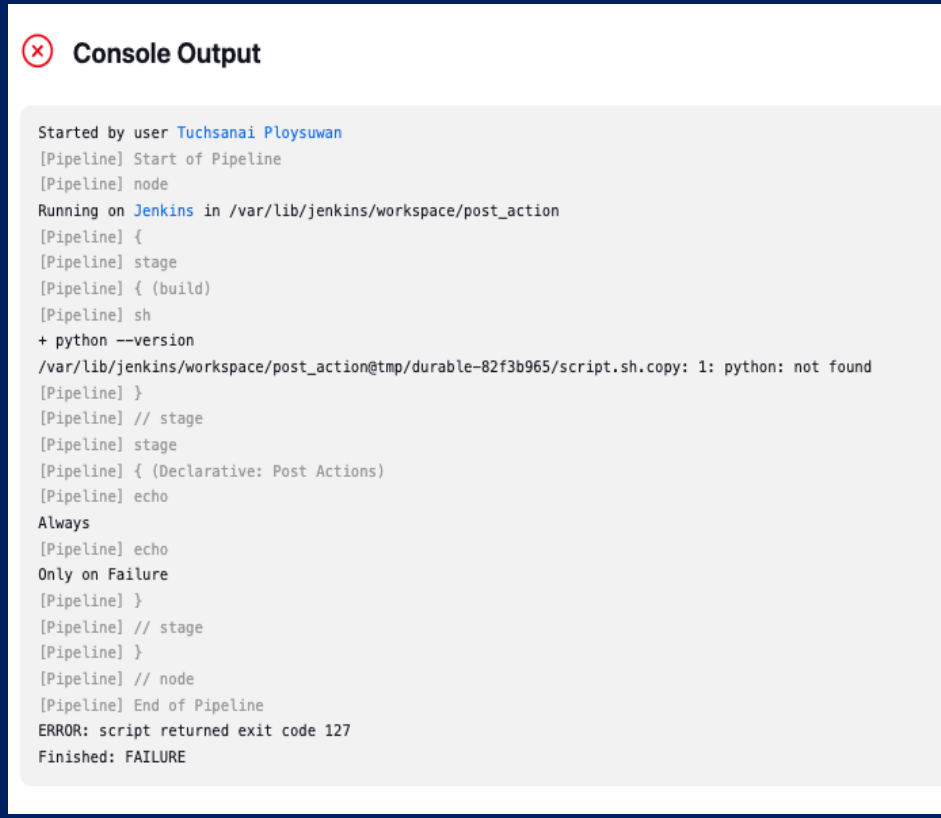
Example 2. Post actions and Conditions

```
pipeline {
    agent any
    stages {
        stage('build') {
            steps {
                sh 'python --version'
            }
        }
    }
 post {
        always {
            echo 'Always'
        }
        success {
            echo 'Only on SUCCESS'
        }
        failure {
            echo 'Only on Failure'
        }
        unstable {
            echo 'Only if run is unstable'
        }
        changed {
            echo 'Only if status changed from Success to Failure or vice versa w.r.t. last run.'
        }
    }
}
```

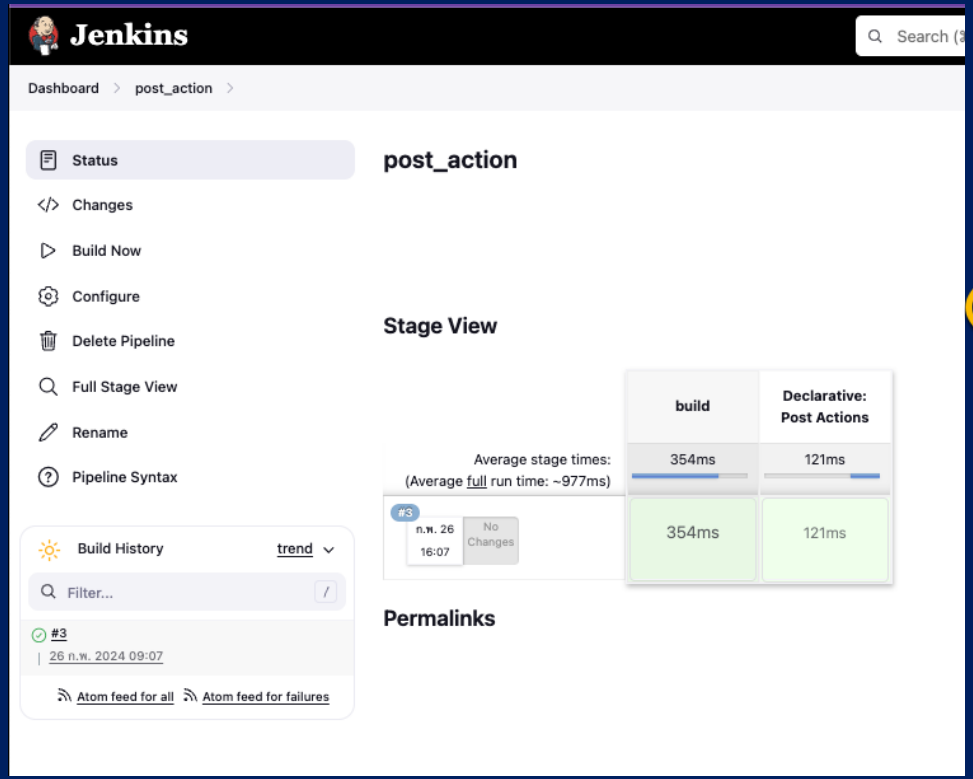
Click: Go to LAB

If there are not python in your system





If there are python in your system



`always`
Run the steps in the `post` section regardless of the completion status of the Pipeline's or stage's run.

`changed`
Only run the steps in `post` if the current Pipeline's or stage's run has a different completion status from its previous run.

`fixed`
Only run the steps in `post` if the current Pipeline's or stage's run is successful and the previous run failed or was unstable.

`regression`
Only run the steps in `post` if the current Pipeline's or stage's run's status is failure, unstable, or aborted and the previous run was successful.

`aborted`
Only run the steps in `post` if the current Pipeline's or stage's run has an "aborted" status, usually due to the Pipeline being manually aborted. This is typically denoted by gray in the web UI.

`failure`
Only run the steps in `post` if the current Pipeline's or stage's run has a "failed" status, typically denoted by red in the web UI.

`success`
Only run the steps in `post` if the current Pipeline's or stage's run has a "success" status, typically denoted by blue or green in the web UI.

`unstable`
Only run the steps in `post` if the current Pipeline's or stage's run has an "unstable" status, usually caused by test failures, code violations, etc. This is typically denoted by yellow in the web UI.

`unsuccessful`
Only run the steps in `post` if the current Pipeline's or stage's run has not a "success" status. This is typically denoted in the web UI depending on the status previously mentioned.

`cleanup`
Run the steps in this `post` condition after every other `post` condition has been evaluated, regardless of the Pipeline or stage's status.

LAB 4.3: Pipeline Environment Variables

- Basic environment variables 1

```
pipeline {
    agent any

    environment {
        // Define environment variables
        MY_ENV_VAR = 'Hello, Jenkins Environment Variables!'
        ANOTHER_VAR = 'This is another environment variable.'
    }

    stages {
        stage('Demo') {
            steps {

                // Use the environment variables
                echo "Using environment variable: ${env.MY_ENV_VAR}"
                echo "Using another environment variable: ${env.ANOTHER_VAR}"

                // Set a new environment variable or modify an existing one
                script {
                    env.NEW_VAR = 'This is a new environment variable set during runtime.'
                }

                echo "Using a newly set environment variable: ${env.NEW_VAR}"
                sh 'printenv'
            }
        }
    }
}
```

Click: Go to LAB

Console Output

```
Started by user Tuchsanai Ploysuwan
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/env_variable
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Demo)
[Pipeline] echo
Using environment variable: Hello, Jenkins Environment Variables!
[Pipeline] echo
Using another environment variable: This is another environment variable.
[Pipeline] script
[Pipeline] {
[Pipeline] }
[Pipeline] // script
[Pipeline] echo
Using a newly set environment variable: This is a new environment variable set during runtime.
[Pipeline] sh
+ printenv
JENKINS_HOME=/var/lib/jenkins
USER=jenkins
CI=true
RUN_CHANGES_DISPLAY_URL=http://54.251.180.144:8080/job/env_variable/4/display/redirect?page=changes
NODE_LABELS=built-in
HUDSON_URL=http://54.251.180.144:8080/
HOME=/var/lib/jenkins
BUILD_URL=http://54.251.180.144:8080/job/env_variable/4/
HUDSON_COOKIE=506258a1-9ba2-4618-b4f0-becad5c27368
JENKINS_SERVER_COOKIE=durable-bb56faeff47a3330ae9e322e09b2ce9ed8d56fa52e54949893ce25db758a8fe5
NOTIFY_SOCKET=/run/systemd/notify
SYSTEMD_EXEC_PID=8226
WORKSPACE=/var/lib/jenkins/workspace/env_variable
LOGNAME=jenkins
NODE_NAME=built-in
JOURNAL_STREAM=8:46337
NEW_VAR=This is a new environment variable set during runtime.
RUN_ARTIFACTS_DISPLAY_URL=http://54.251.180.144:8080/job/env_variable/4/display/redirect?page=artifacts
STAGE_NAME=Demo
EXECUTOR_NUMBER=0
RUN_TESTS_DISPLAY_URL=http://54.251.180.144:8080/job/env_variable/4/display/redirect?page=tests
BUILD_DISPLAY_NAME=#4
HUDSON_HOME=/var/lib/jenkins
JOB_BASE_NAME=env_variable
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
INVOCATION_ID=3dd1b667c6f2440d98512e0c5fa349ed
BUILD_ID=4
BUILD_TAG=jenkins-env_variable-4
LANG=C.UTF-8
JENKINS_URL=http://54.251.180.144:8080/
JOB_URL=http://54.251.180.144:8080/job/env_variable/
ANOTHER_VAR=This is another environment variable.
MY_ENV_VAR=Hello, Jenkins Environment Variables!
BUILD_NUMBER=4
JENKINS_NODE_COOKIE=38051b77-e40e-43eb-a269-1c5f0bae68b5
SHELL=/bin/bash
RUN_DISPLAY_URL=http://54.251.180.144:8080/job/env_variable/4/display/redirect
HUDSON_SERVER_COOKIE=25d1a75683dc7e3e
JOB_DISPLAY_URL=http://54.251.180.144:8080/job/env_variable/display/redirect
JOB_NAME=env_variable
PWD=/var/lib/jenkins/workspace/env_variable
WORKSPACE_TMP=/var/lib/jenkins/workspace/env_variable@tmp
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
```

- 3.2 Basic environment variables 2

```
pipeline {
    agent any

    environment {
        DISABLE_AUTH = 'true'
        DB_ENGINE    = 'sqlite'
    }

    stages {
        stage('Build') {
            steps {
                echo "Database engine is ${DB_ENGINE}"
                echo "DISABLE_AUTH is ${DISABLE_AUTH}"
                sh 'printenv'
            }
        }
    }
}
```

Console Output

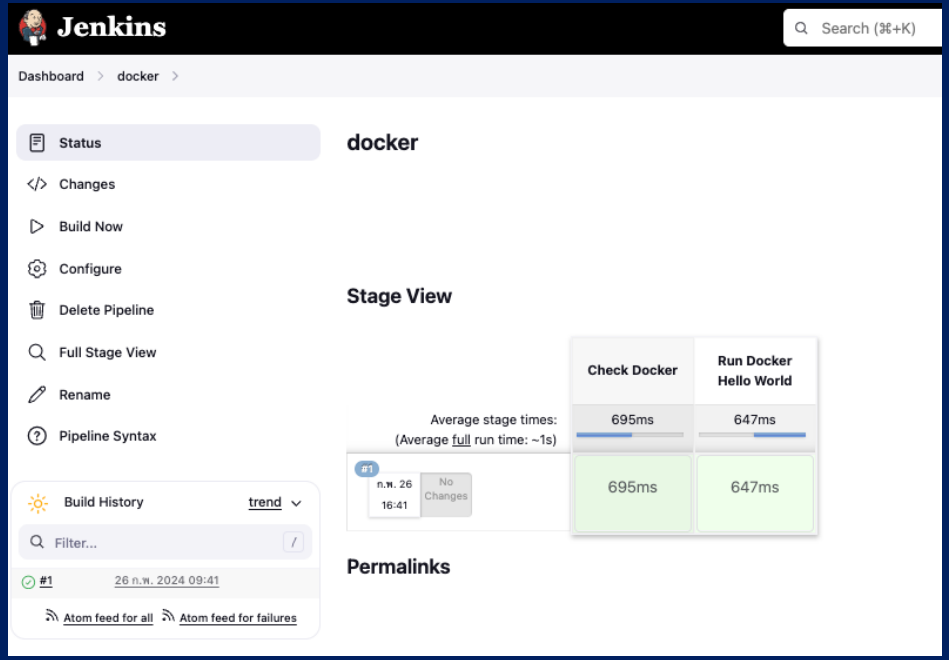
```
Started by user Tuchsanai Ploysuwan
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/env_variable
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Database engine is sqlite
[Pipeline] echo
DISABLE_AUTH is true
[Pipeline] sh
+ printenv
DISABLE_AUTH=true
JENKINS_HOME=/var/lib/jenkins
DB_ENGINE=sqlite
USER=jenkins
CI=true
RUN_CHANGES_DISPLAY_URL=http://54.251.180.144:8080/job/env_variable/7/display/redirect?page=changes
NODE_LABELS=built-in
HUDSON_URL=http://54.251.180.144:8080/
HOME=/var/lib/jenkins
```

LAB 4.4: Docker and Environment variables

Example 4. Docker and Environment variables

```
pipeline {
    agent any  // Execute on any available Jenkins agent

    environment {
        // Define an environment variable
        DOCKER_VERSION = ''
    }

    stages {
        stage('Check Docker') {
            steps {
                script {
                    // Try to get Docker version
                    def dockerCheck = sh(script: 'docker --version', returnStdout: true).trim()

                    sh 'echo dockerCheck = ' + dockerCheck

                    // Check if Docker is available and set environment variable accordingly
                    if (dockerCheck.contains('Docker version')) {
                        env.DOCKER_VERSION = dockerCheck
                    } else {
                        env.DOCKER_VERSION = 'No Docker'
                    }
                }
            }
        }

        stage('Run Docker Hello World') {
            steps {
                script {
                    // Check if Docker version was found and run hello-world image
                    if (env.DOCKER_VERSION != 'No Docker') {
                        sh 'docker run hello-world'
                    } else {
                        echo 'No Docker available on this machine'
                    }
                }
            }
        }
    }
}
```

Click: Go to LAB



Console Output

```
Started by user Tuchsanai Ploysuwan
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/docker
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Check Docker)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ docker --version
[Pipeline] sh
+ echo dockerCheck = Docker version 25.0.3, build 4debf41
dockerCheck = Docker version 25.0.3, build 4debf41
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Run Docker Hello World)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```