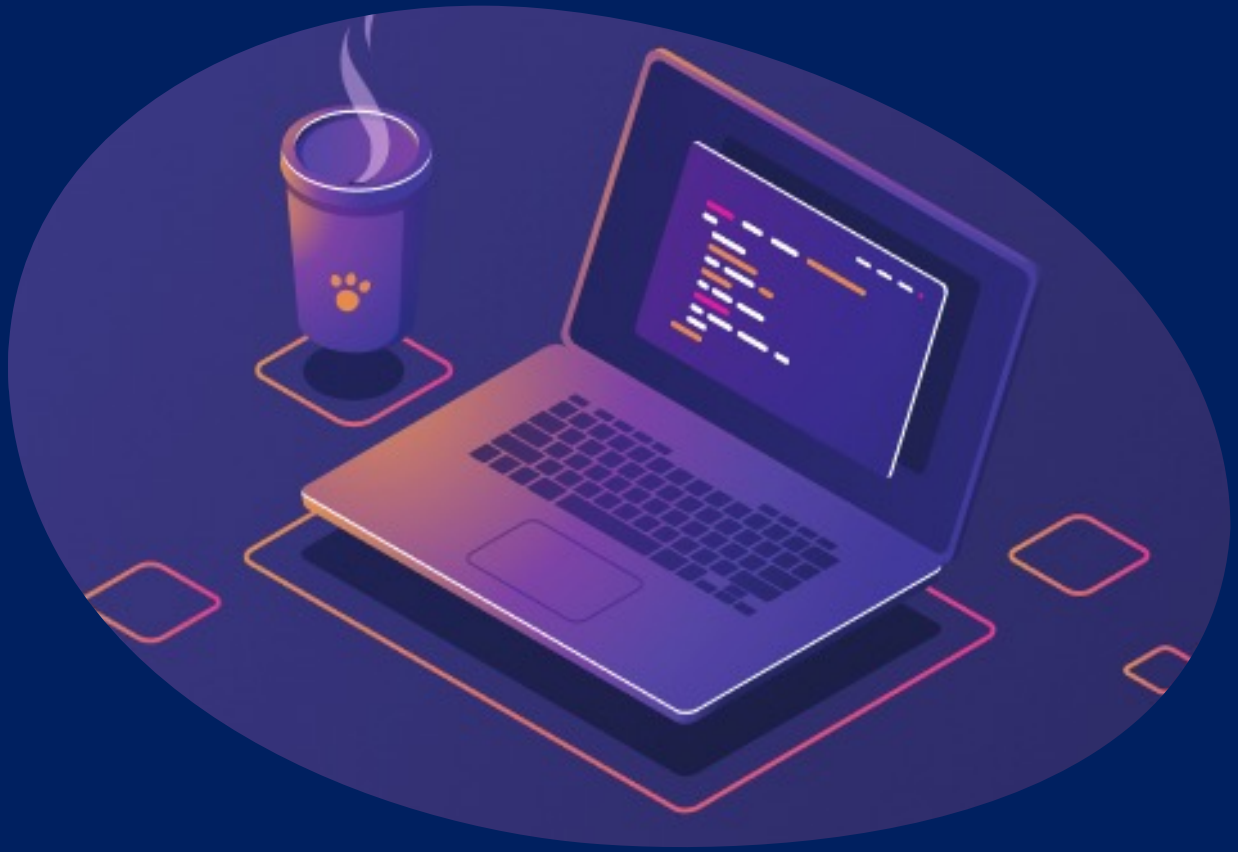


Week 12 : SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

Jenkins เป็นเครื่องมือ automation server แบบโอเพนซอร์สที่ใช้ในการทำงานที่เกี่ยวข้องกับ Continuous Integration (CI) และ Continuous Delivery (CD) ของซอฟต์แวร์ ซึ่งช่วยให้การพัฒนา การทดสอบ และการนำซอฟต์แวร์ไปใช้เป็นไปอย่างต่อเนื่องและอัตโนมัติ

คุณสมบัติหลักของ Jenkins:

1. **Continuous Integration (CI)**: Jenkins ช่วยให้นักพัฒนาสามารถรวมโค้ดจากหลาย ๆ คนเข้าด้วยกันได้อย่างราบรื่น โดย Jenkins จะคอยตรวจสอบโค้ดที่ถูกส่งเข้าไปในระบบ (เช่น Git) และทำการ build, ทดสอบ, หรือดำเนินการที่กำหนดเมื่อมีการเปลี่ยนแปลงโค้ด
2. **Continuous Delivery (CD)**: หลังจากการทดสอบและ build สำเร็จ Jenkins สามารถ deploy ซอฟต์แวร์ไปยังสภาพแวดล้อมต่าง ๆ เช่น staging, production ได้อย่างอัตโนมัติ
3. **Pipeline**: Jenkins มีระบบ **Pipeline** ซึ่งเป็นการจัดการขั้นตอนการทำงานทั้งหมดในการ build, test, และ deploy ซอฟต์แวร์ในรูปแบบของสคริปต์ ทำให้ง่ายต่อการควบคุมและปรับแต่งกระบวนการต่าง ๆ
4. **Plugins**: Jenkins มีระบบปลั๊กอินมากกว่า 1,500 รายการ ซึ่งช่วยขยายความสามารถของ Jenkins ไม่ว่าจะเป็นการเชื่อมต่อกับระบบควบคุมเวอร์ชัน (เช่น Git), การจัดการ build tools (เช่น Maven, Gradle), การจัดการ container (เช่น Docker) หรือระบบ cloud ต่าง ๆ
5. **การจัดการอัตโนมัติ**: Jenkins สามารถทำงานได้ทั้งแบบ on-premise และบน cloud รองรับการตั้งค่าให้ทำงานโดยอัตโนมัติ เช่น การรันสคริปต์ การตรวจสอบการเปลี่ยนแปลงของโค้ด การส่งการแจ้งเตือนเมื่อเกิดข้อผิดพลาด

การทำงานของ Jenkins:

- Jenkins ทำงานโดยการตั้งค่า **Job** หรือ **Pipeline** ซึ่งกำหนดขั้นตอนต่าง ๆ ของการพัฒนา เช่น การ build, การทดสอบ, และการ deploy โค้ด
- Jenkins จะตรวจสอบการเปลี่ยนแปลงใน repository เช่น Git เมื่อมีการ commit โค้ด Jenkins จะเริ่มกระบวนการอัตโนมัติ เช่น การ build หรือทดสอบโค้ดนั้น
- Jenkins สามารถกำหนดเงื่อนไขการทำงาน (เช่น ทำงานทุกครั้งที่มีการ commit หรือทำงานในช่วงเวลาที่กำหนด)

ตั้งweb hook on github

Traditional Deployment

Developer

Operations

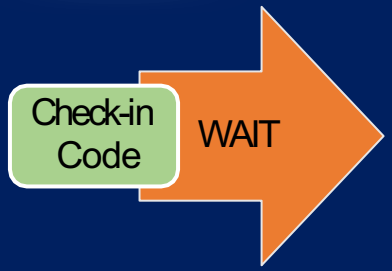
จำลองปัญหา

ขั้นตอน deployment sw ไปที่ customer

Traditional Deployment

Developer

Operations

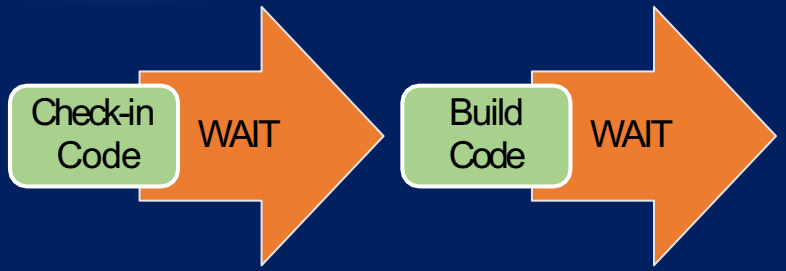


Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

Developer

Operations

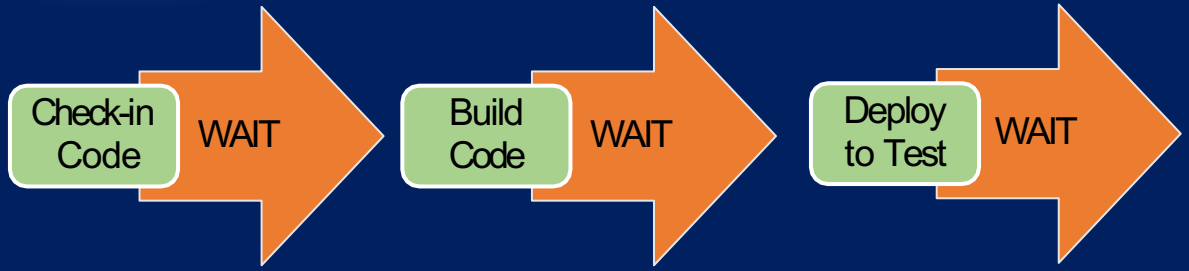


Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

Developer

Operations

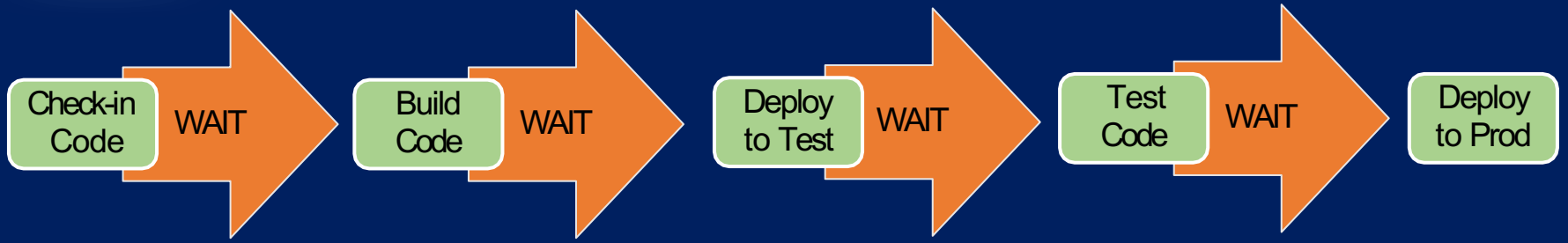


Hours/Days + Lot of Grief for Developer & Operations

Traditional Deployment

Developer

Operations



Hours/Days + Lot of Grief for Developer & Operations

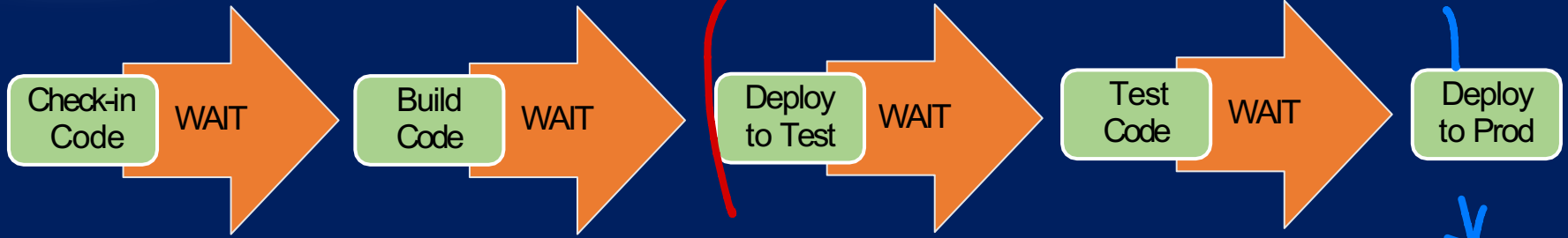
Traditional Deployment

When are you gonna deploy my code?

Developer

When you stop breaking my servers

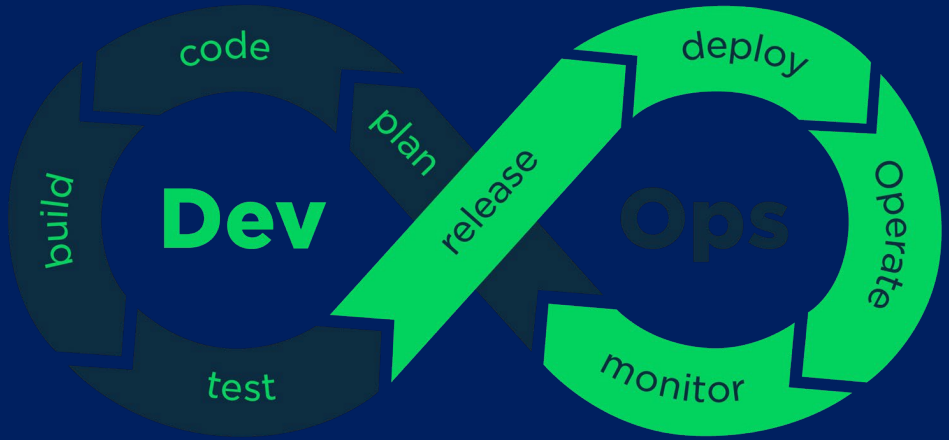
Operations




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”



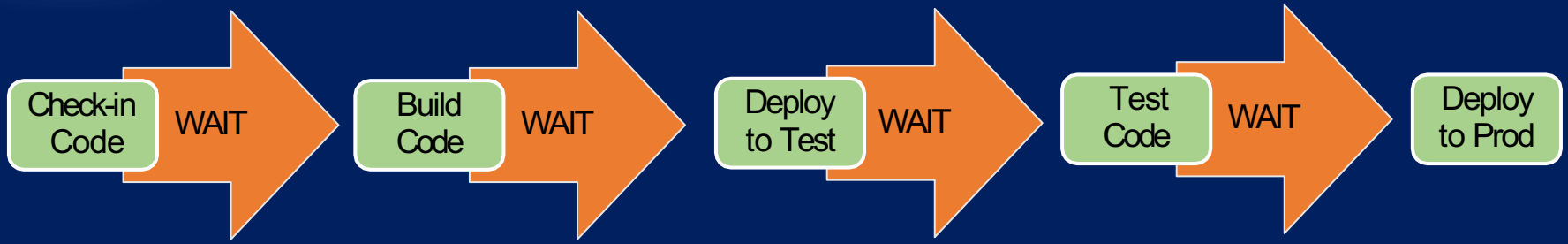
Traditional Deployment

When are you gonna deploy my code?

Developer

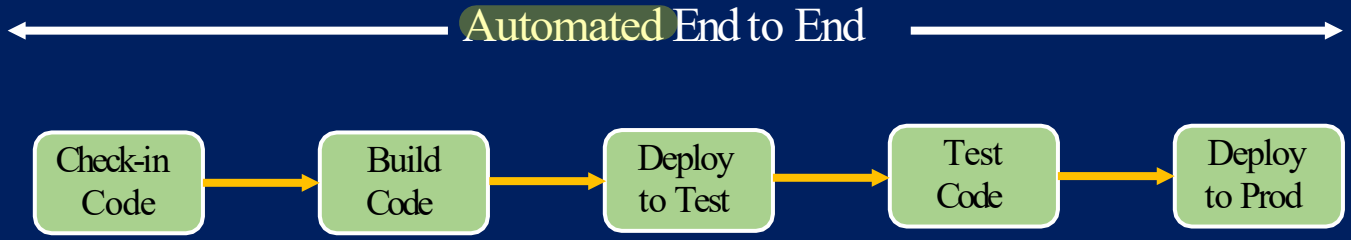
When you stop breaking my servers

Operations



Devops

Same Team



- Whole flow done in seconds!
- Easy to rollback in case of errors

DevOps

Same Team

Automated End to End

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

Rollback to Previous Version

General DevOps Practices

- Automate everything!
- Deploy frequently rather than one mega deployment in months
- Codify every step - infrastructure, application and more
- Rome was not built in a day!

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)

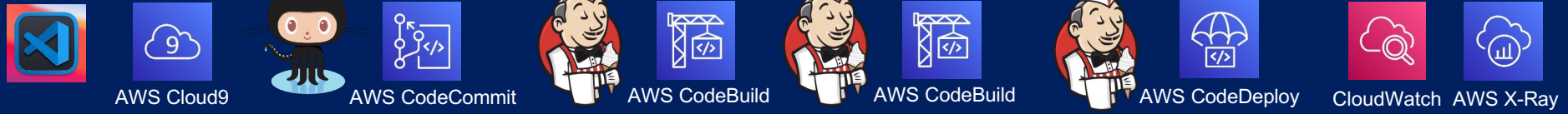
ภาพนี้แบ่งกระบวนการ DevOps ออกเป็น 6 ขั้นตอนหลัก ได้แก่:

1. **Author:** เป็นขั้นตอนที่นักพัฒนาเขียนโค้ด
2. **Source:** นักพัฒนาทำการนำโค้ดที่เขียนขึ้นไปเก็บไว้ใน repository เช่น Git
3. **Build:** Jenkins จะทำการคอมไพล์โค้ด สร้าง artifact (ไฟล์ที่สร้างขึ้นจากการคอมไพล์) และทำการทดสอบหน่วย (unit test)
4. **Test:** ในขั้นตอนนี้จะทำการทดสอบซอฟต์แวร์ในรูปแบบต่างๆ เช่น การทดสอบการทำงานร่วมกัน (integration testing), การทดสอบโหลด (load testing), การทดสอบ UI (UI testing), และการทดสอบการเจาะระบบ (penetration testing)
5. **Deploy:** เมื่อผ่านการทดสอบแล้ว ซอฟต์แวร์จะถูกนำไป deploy ไปยังสภาพแวดล้อมต่างๆ เช่น การ deploy ไปยัง server หรือ cloud platform
6. **Monitor:** ในขั้นตอนนี้จะทำการติดตามสถานะของซอฟต์แวร์ที่ถูก deploy ไปแล้ว เช่น การตรวจสอบ log, การวัดค่าต่างๆ (metrics), และการติดตาม trace

นอกจากนี้ ภาพยังแสดงให้เห็นถึงแนวคิดของ Continuous Integration (CI), Continuous Delivery (CD), และ Continuous Deployment (CD) ซึ่งเป็นส่วนหนึ่งของกระบวนการ DevOps:

- **Continuous Integration (CI):** เป็นกระบวนการที่นักพัฒนาจะทำการรวมโค้ดที่เขียนขึ้นเข้ากับโค้ดหลักบ่อยครั้ง ทำให้สามารถตรวจพบปัญหาได้เร็วขึ้น
- **Continuous Delivery (CD):** เป็นกระบวนการที่ซอฟต์แวร์ที่ผ่านการทดสอบแล้วจะถูกเตรียมพร้อมสำหรับการ deploy ไปยังสภาพแวดล้อมจริง แต่ยังต้องมีการอนุมัติจากผู้เกี่ยวข้องก่อน
- **Continuous Deployment (CD):** เป็นกระบวนการที่ซอฟต์แวร์ที่ผ่านการทดสอบแล้วจะถูก deploy ไปยังสภาพแวดล้อมจริงโดยอัตโนมัติ

DevOps Tools



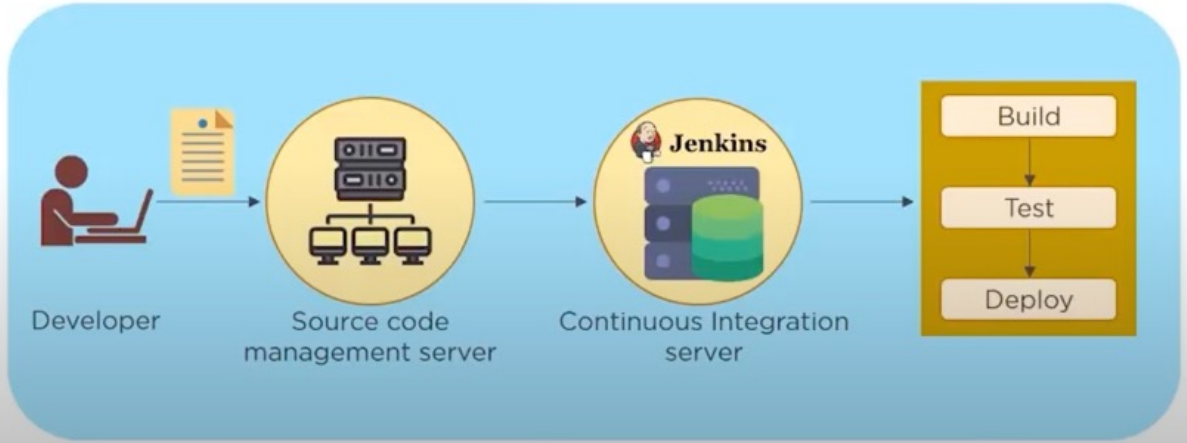
AUTHOR → SOURCE → BUILD → TEST → DEPLOY → MONITOR

- 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

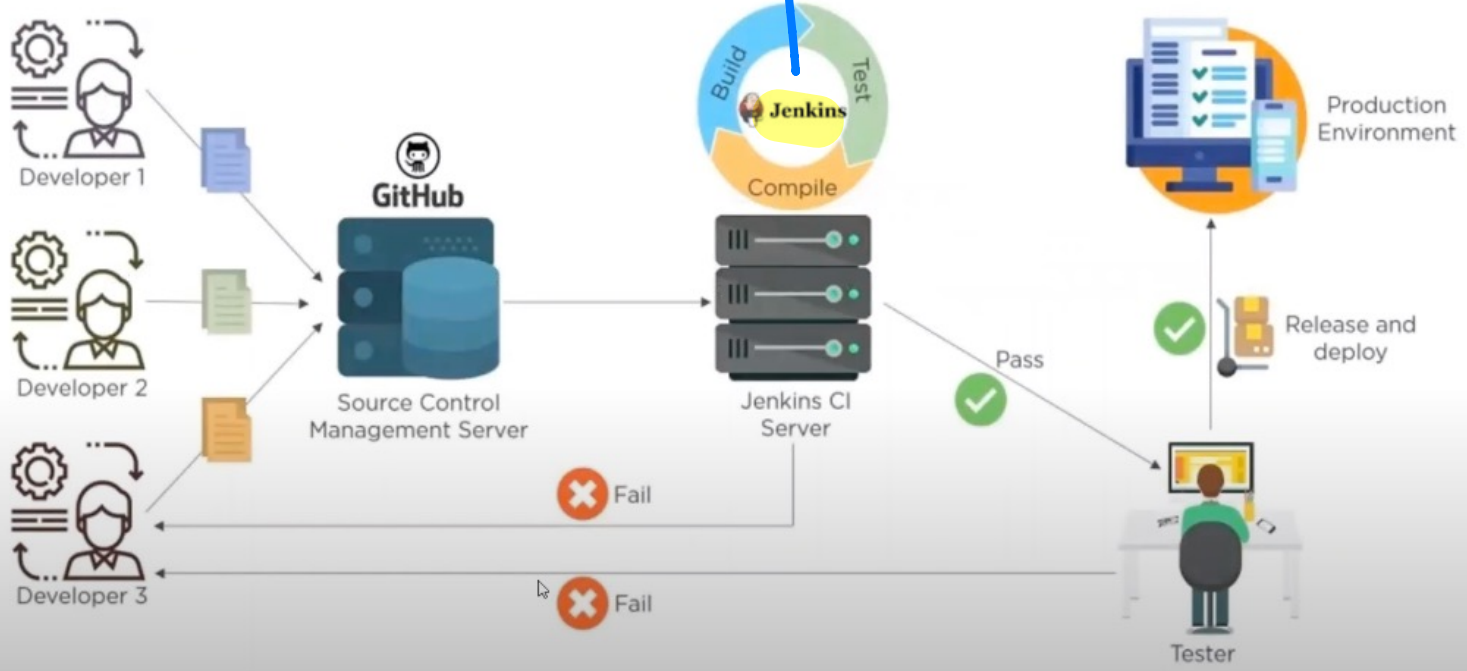
What is Jenkins?

Jenkins

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



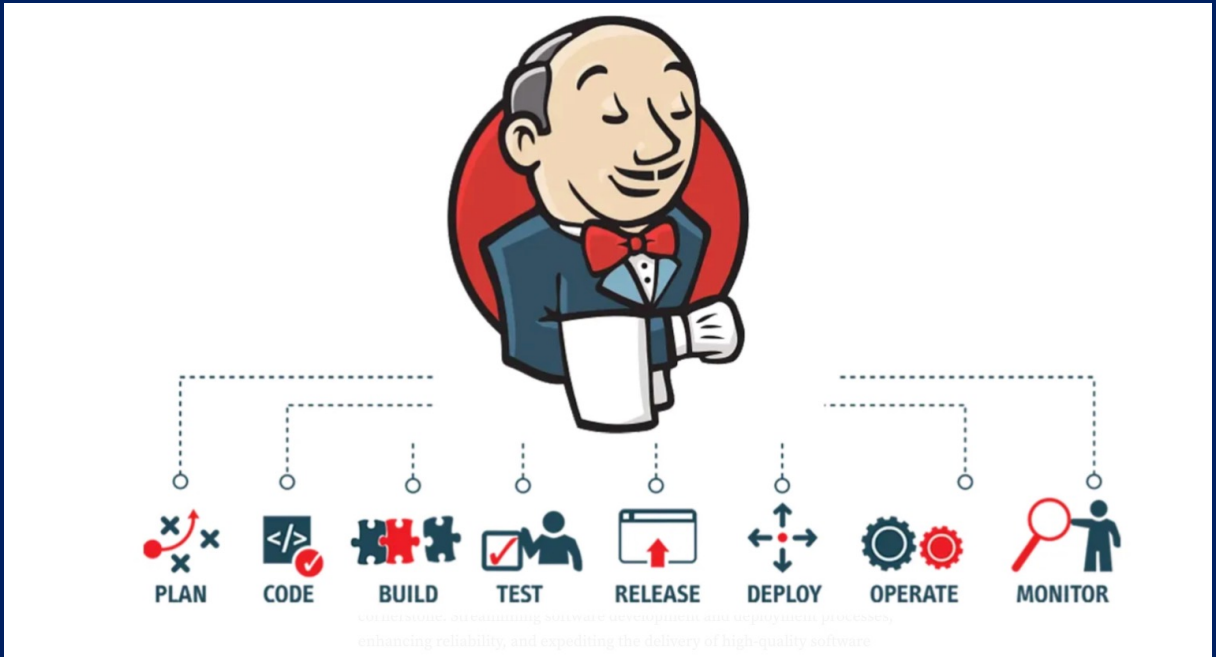
CI & CD


Developer 1
Developer 2
Developer 3
GitHub
Source Control Management Server
Build
Test
Jenkins
Compile
Jenkins CI Server
Fail
Fail
Pass
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
```

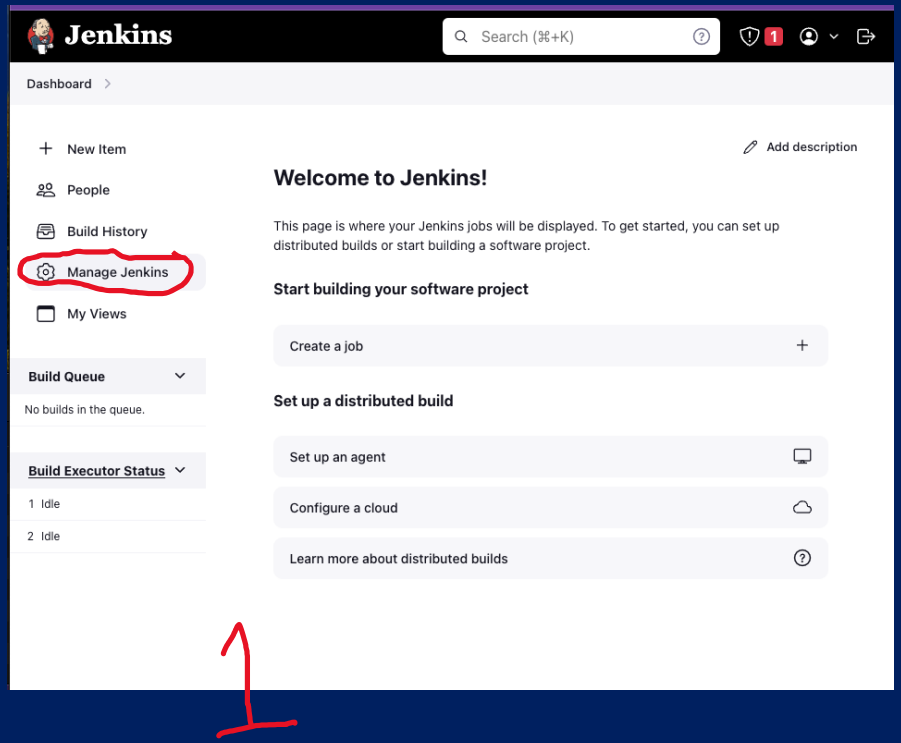
```
sudo systemctl start jenkins
```

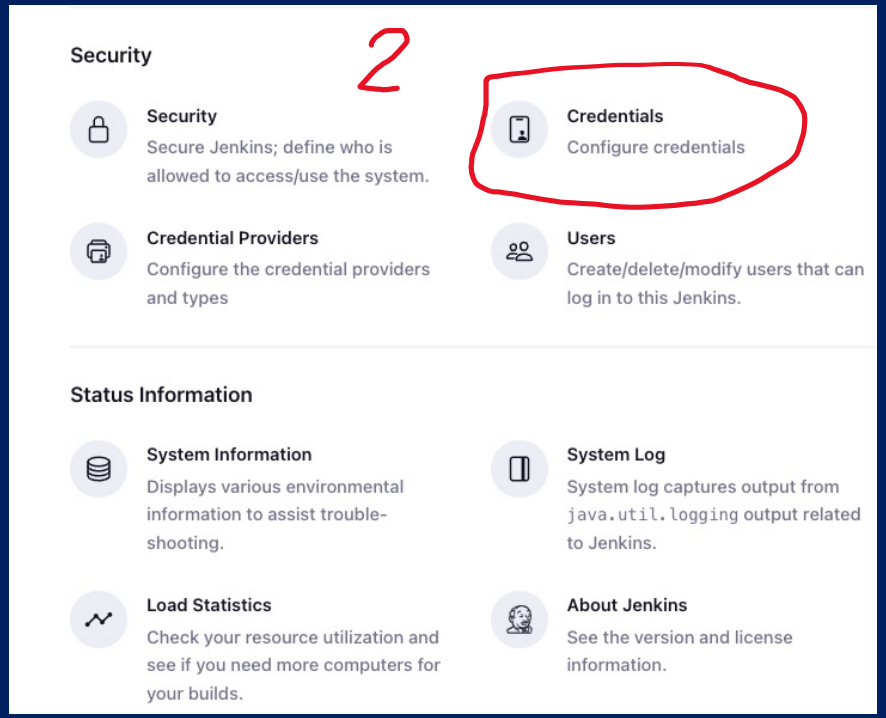
```
sudo systemctl status jenkins
```

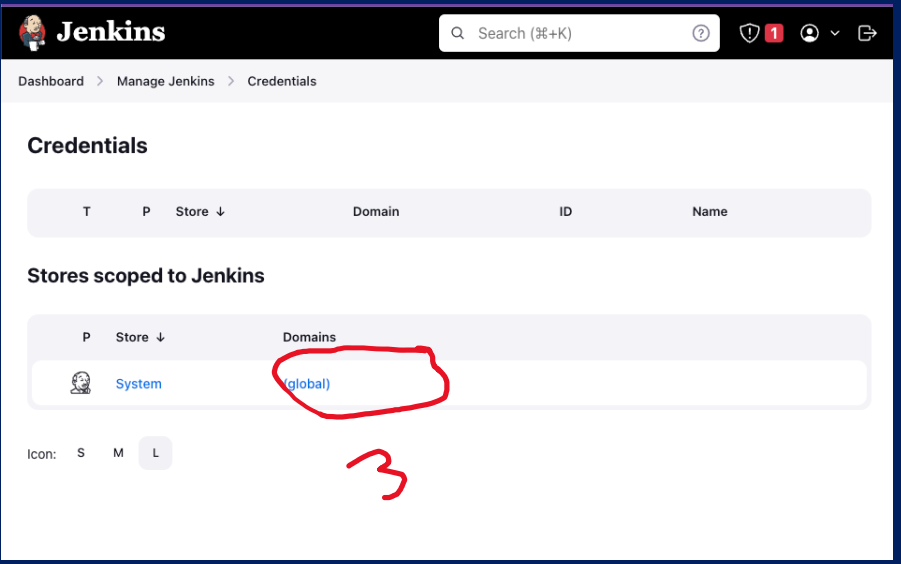
LAB 02 : Jenkins Add Credentials

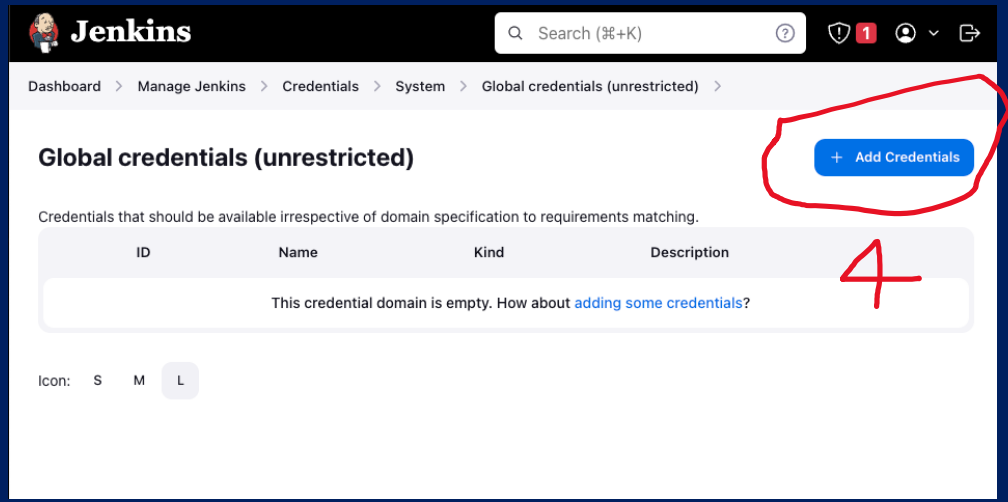
- เพิ่ม private key ไปที่ Jenkins
  เพื่อ ใช้ GitHub, Docker

1.Add SSH-Credentials

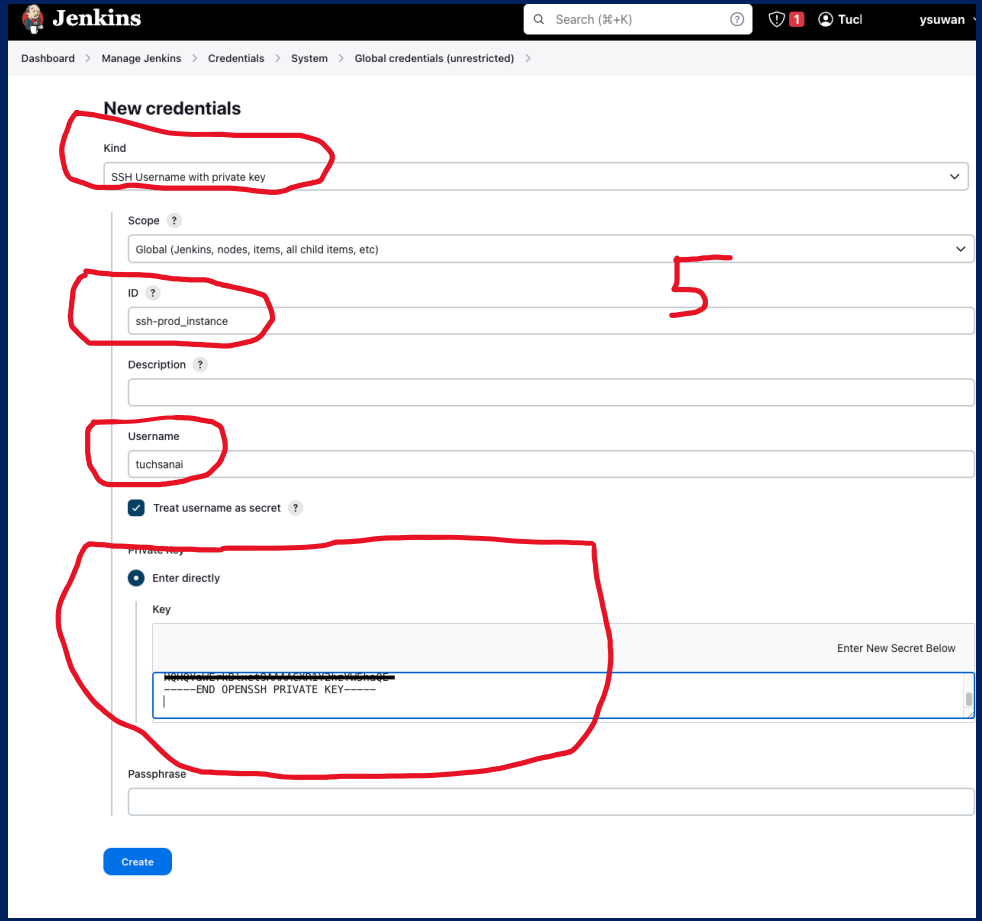


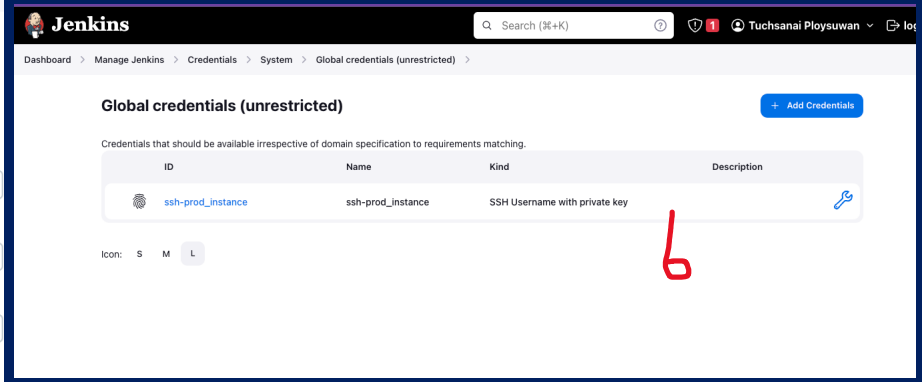


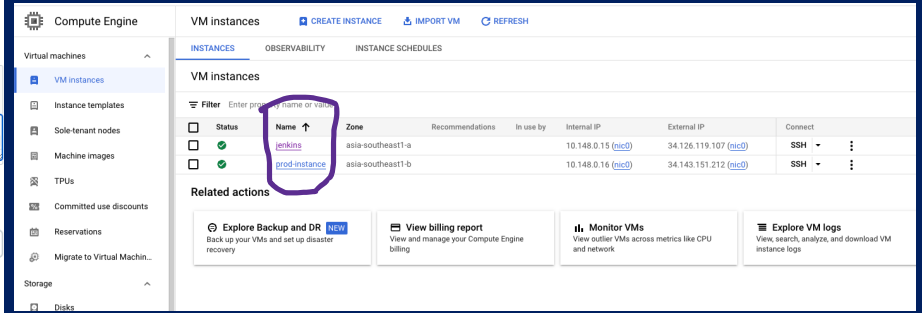

Jenkins
Search (⌘+K)
1
Dashboard
Manage Jenkins
Credentials
Credentials
T
P
Store ↓
Domain
ID
Name
Stores scoped to Jenkins
P
Store ↓
Domains
System
(global)
Icon:
S
M
L
3


Jenkins
Search (⌘+K)
1
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

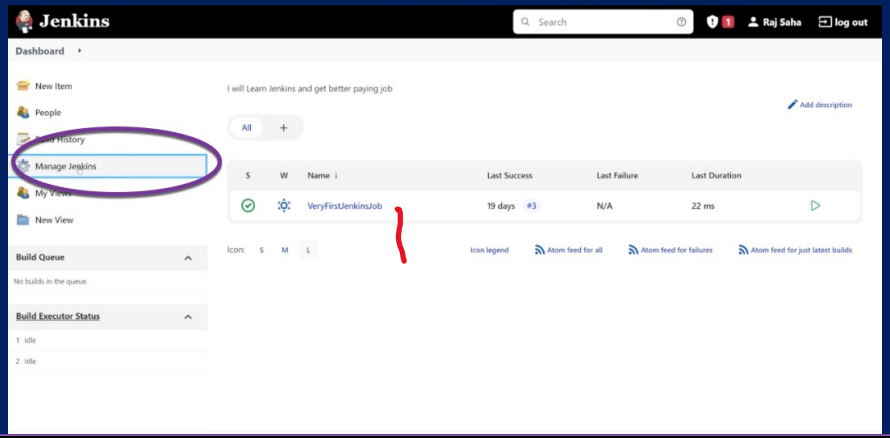
For SSH

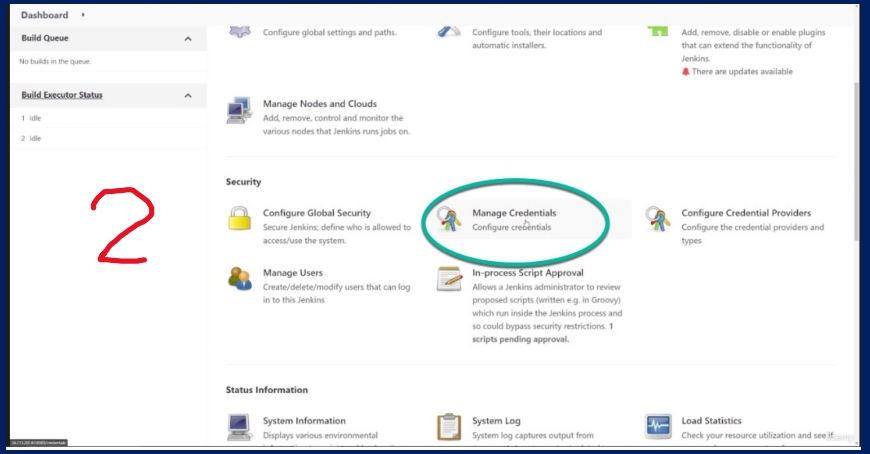

Jenkins
Search (⌘+K)
Tucl
ysuwan
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
New credentials
Kind
SSH Username with private key
Scope
Global (Jenkins, nodes, items, all child items, etc)
5
ID
ssh-prod_instance
Description
Username
tuchsanai
Treat username as secret
Enter directly
Key
Enter New Secret Below
-----END OPENSSH PRIVATE KEY-----
Passphrase
Create

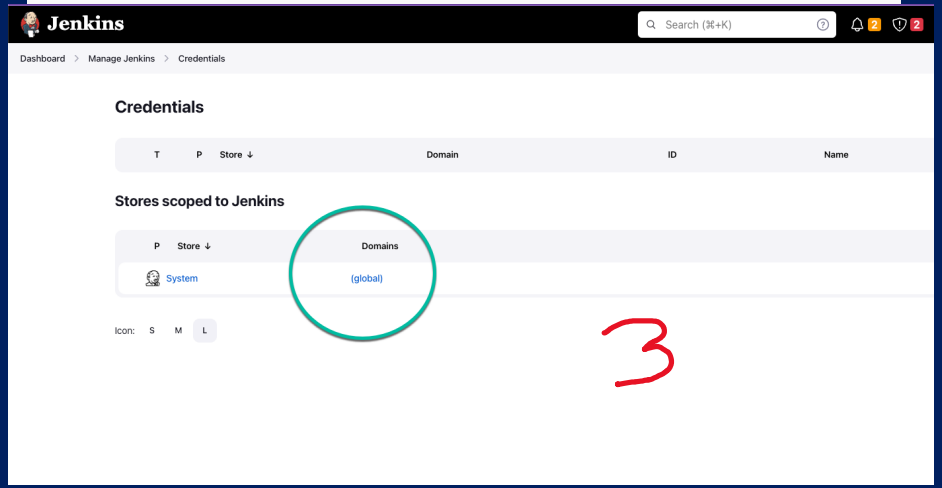

Jenkins
Search (⌘+K)
Tuchsanai Ploysuwan
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Global credentials (unrestricted)
+ Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID
Name
Kind
Description
ssh-prod_instance
ssh-prod_instance
SSH Username with private key
Icon: S M L
6

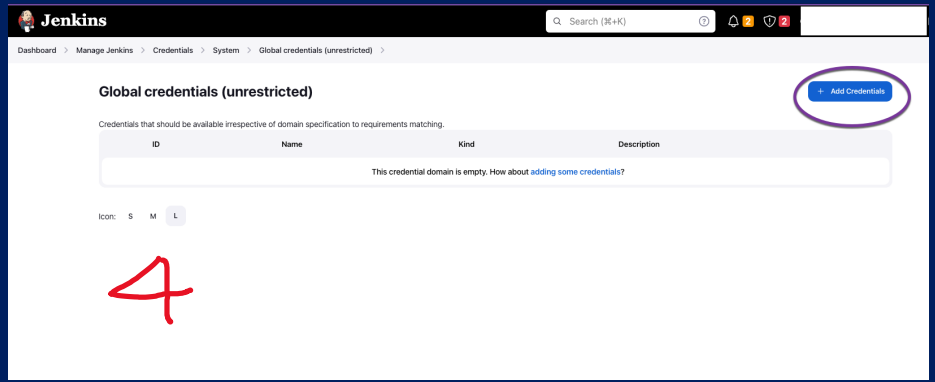

Compute Engine
VM instances
CREATE INSTANCE
IMPORT VM
REFRESH
Virtual machines
VM instances
Instance templates
Sole-tenant nodes
Machine images
TPUs
Committed use discounts
Reservations
Migrate to Virtual Machin...
Storage
Disks
INSTANCES
OBSERVABILITY
INSTANCE SCHEDULES
VM instances
Filter
Status
Name
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

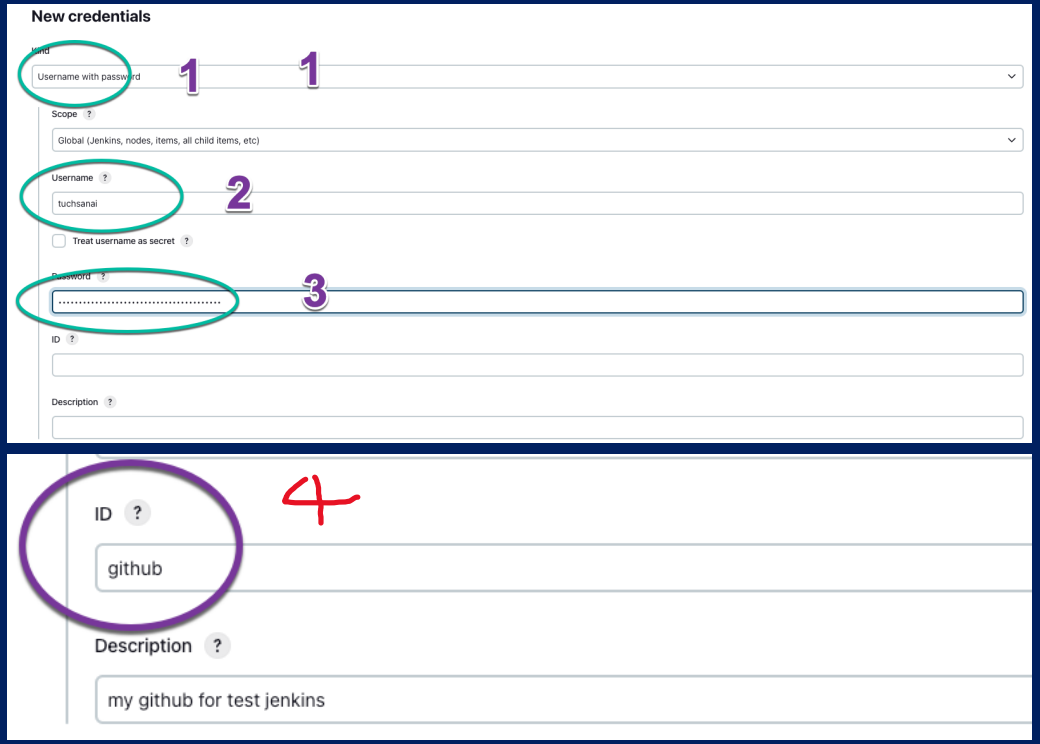




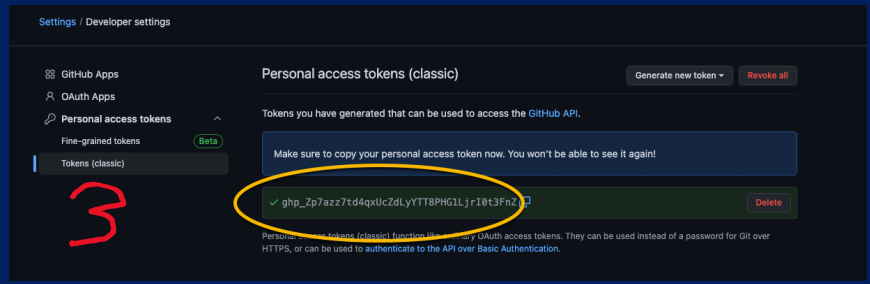


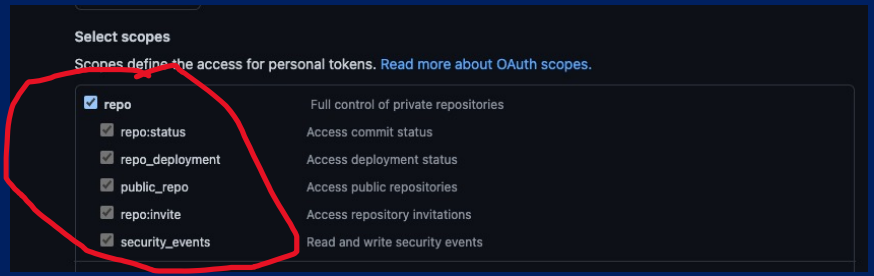


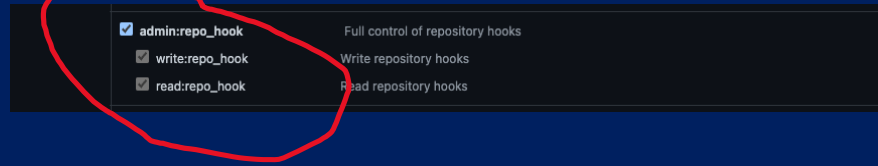
For Github


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

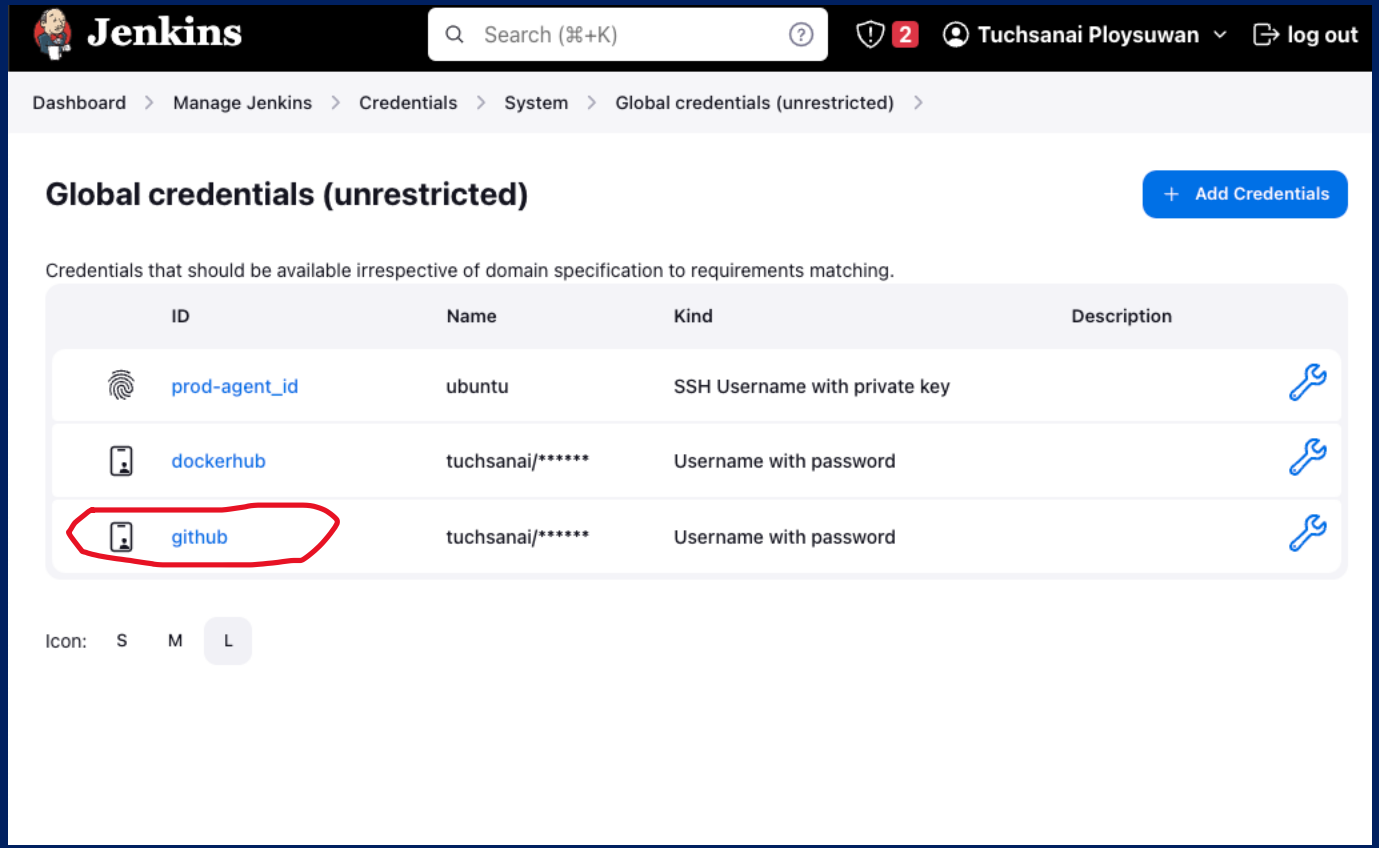
ID
4
github
Description
my github for test jenkins


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
HTTPS, or can be used to authenticate to the API over Basic Authentication.

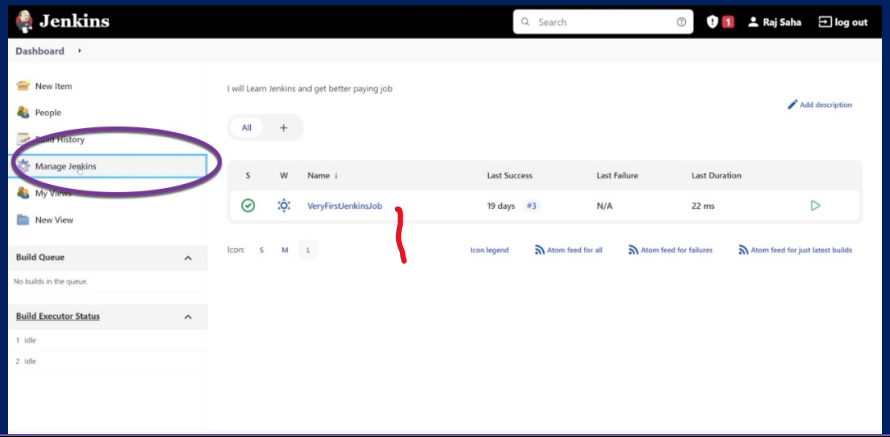

Select scopes
Scopes define the access for personal tokens. Read more about OAuth scopes.
repo
Full control of private repositories
repo:status
Access commit status
repo_deployment
Access deployment status
public_repo
Access public repositories
repo:invite
Access repository invitations
security_events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

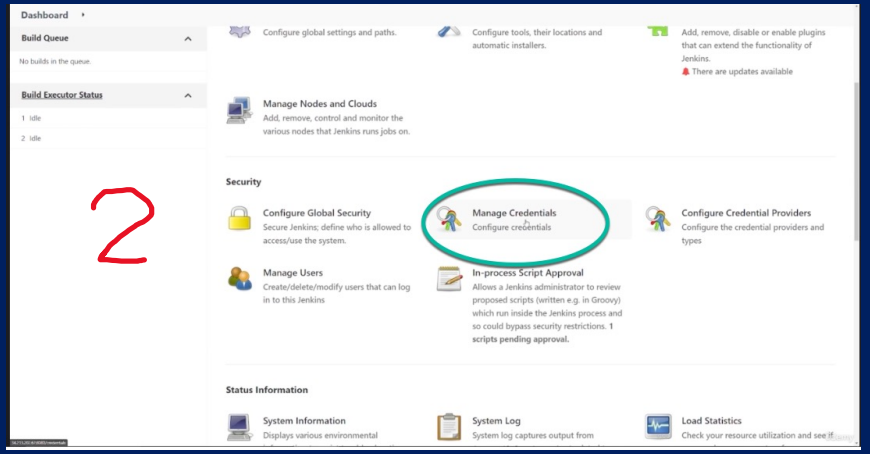

admin:repo_hook
Full control of repository hooks
write:repo_hook
Write repository hooks
read:repo_hook
Read repository hooks

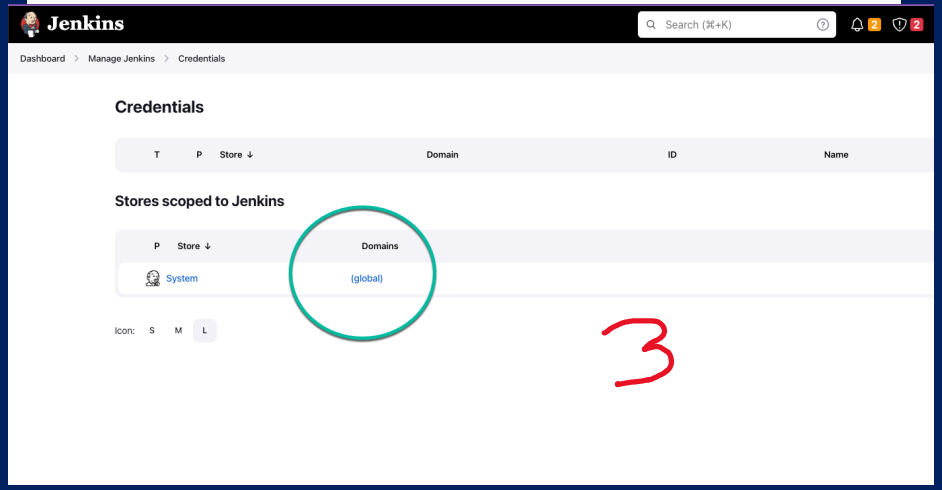
For Github

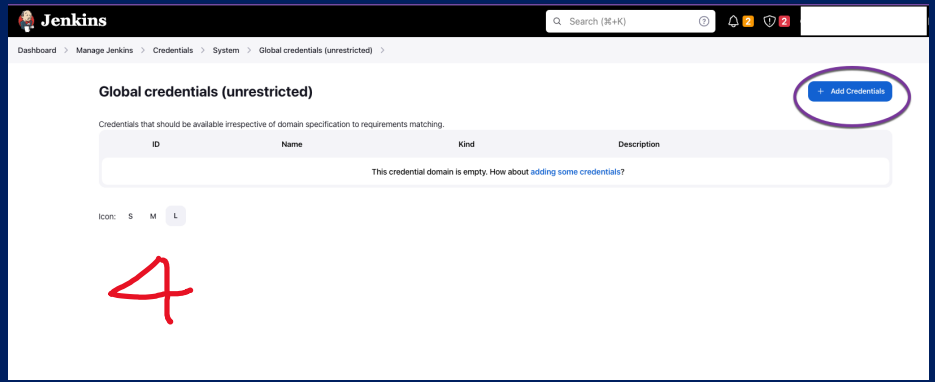

Jenkins
Search (⌘+K)
2
Tuchsanai Ploysuwan
log out
Dashboard
Manage Jenkins
Credentials
System
Global credentials (unrestricted)
Global credentials (unrestricted)
Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID
Name
Kind
Description
prod-agent_id
ubuntu
SSH Username with private key
dockerhub
tuchsanai/******
Username with password
github
tuchsanai/******
Username with password
Icon:
S
M
L

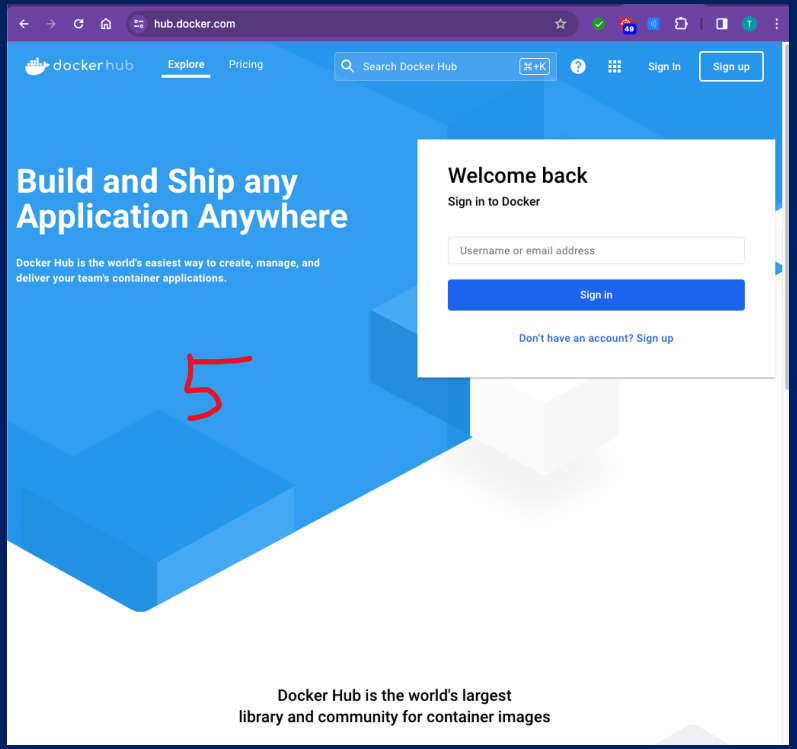
3.Add docker hub Credentials

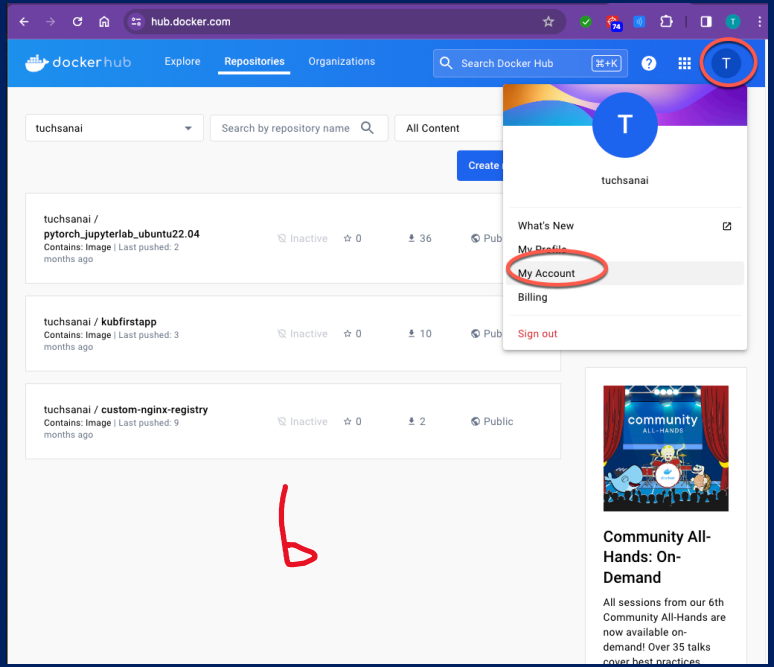


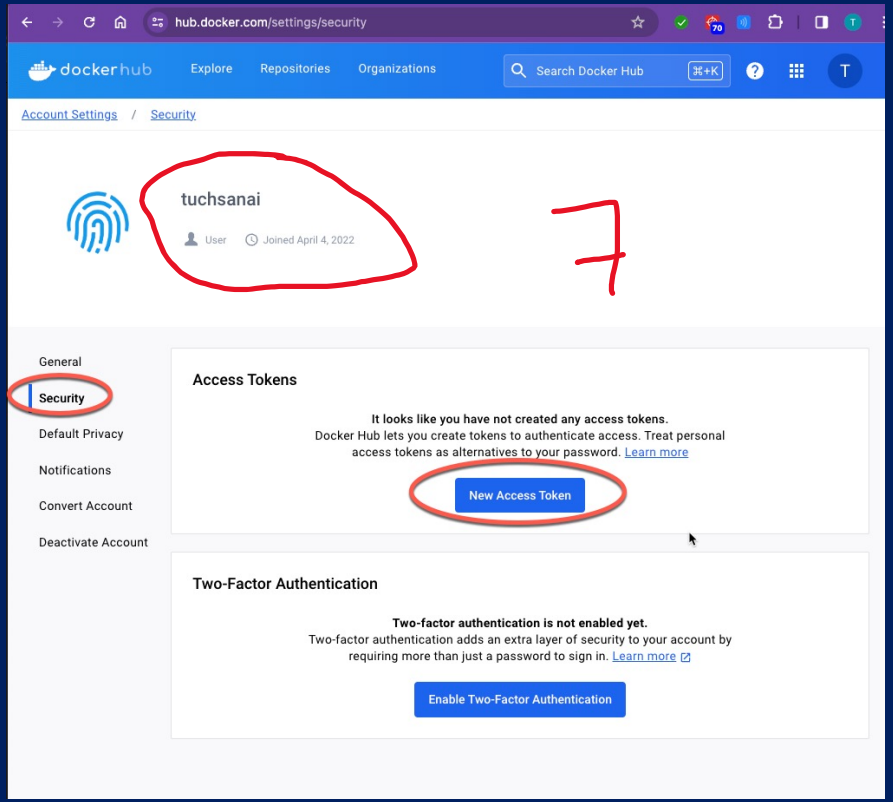


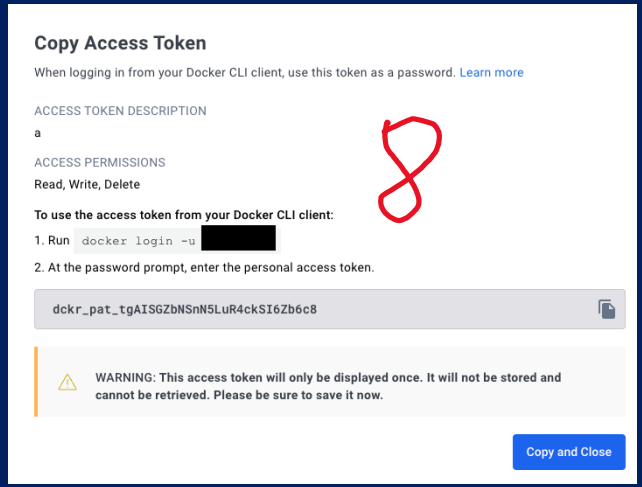


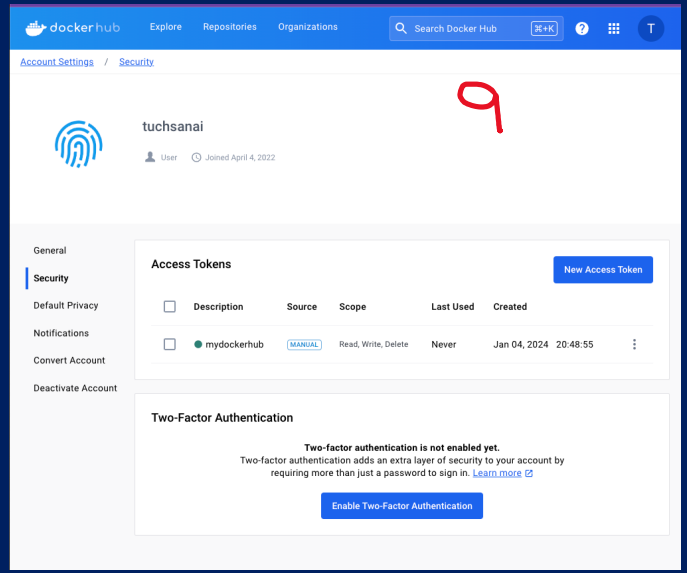


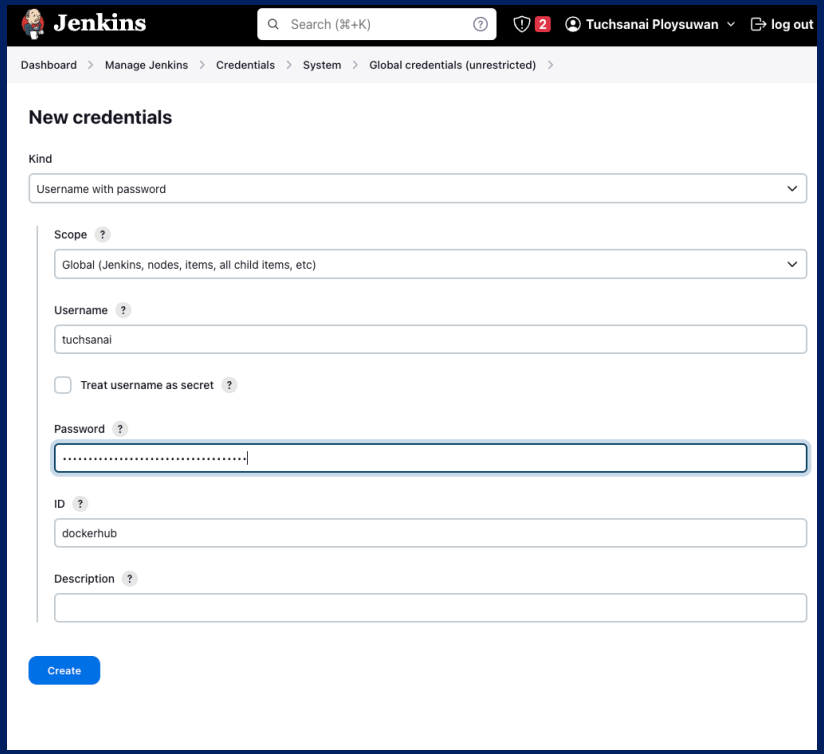

hub.docker.com
dockerhub
Explore
Pricing
Search Docker Hub
⌘+K
Sign In
Sign up
Build and Ship any Application Anywhere
Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications.
Welcome back
Sign in to Docker
Username or email address
Sign in
Don't have an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

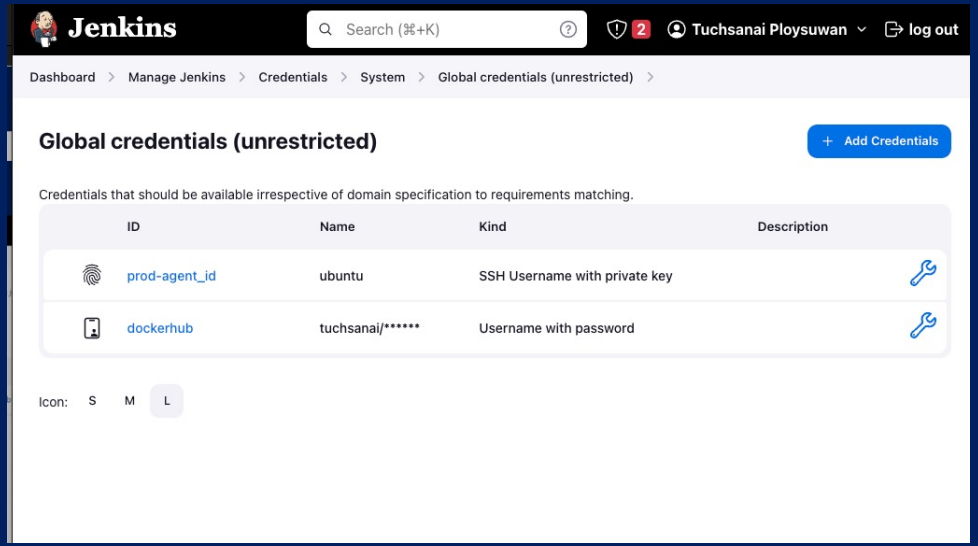

hub.docker.com
dockerhub
Explore
Repositories
Organizations
Search Docker Hub
⌘+K
T
tuchsanai
Search by repository name
All Content
Create
tuchsanai / pytorch_jupyterlab_ubuntu22.04
Contains: Image | Last pushed: 2 months ago
Inactive
0
36
tuchsanai / kubfirstapp
Contains: Image | Last pushed: 3 months ago
Inactive
0
10
tuchsanai / custom-nginx-registry
Contains: Image | Last pushed: 9 months ago
Inactive
0
2
Public
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

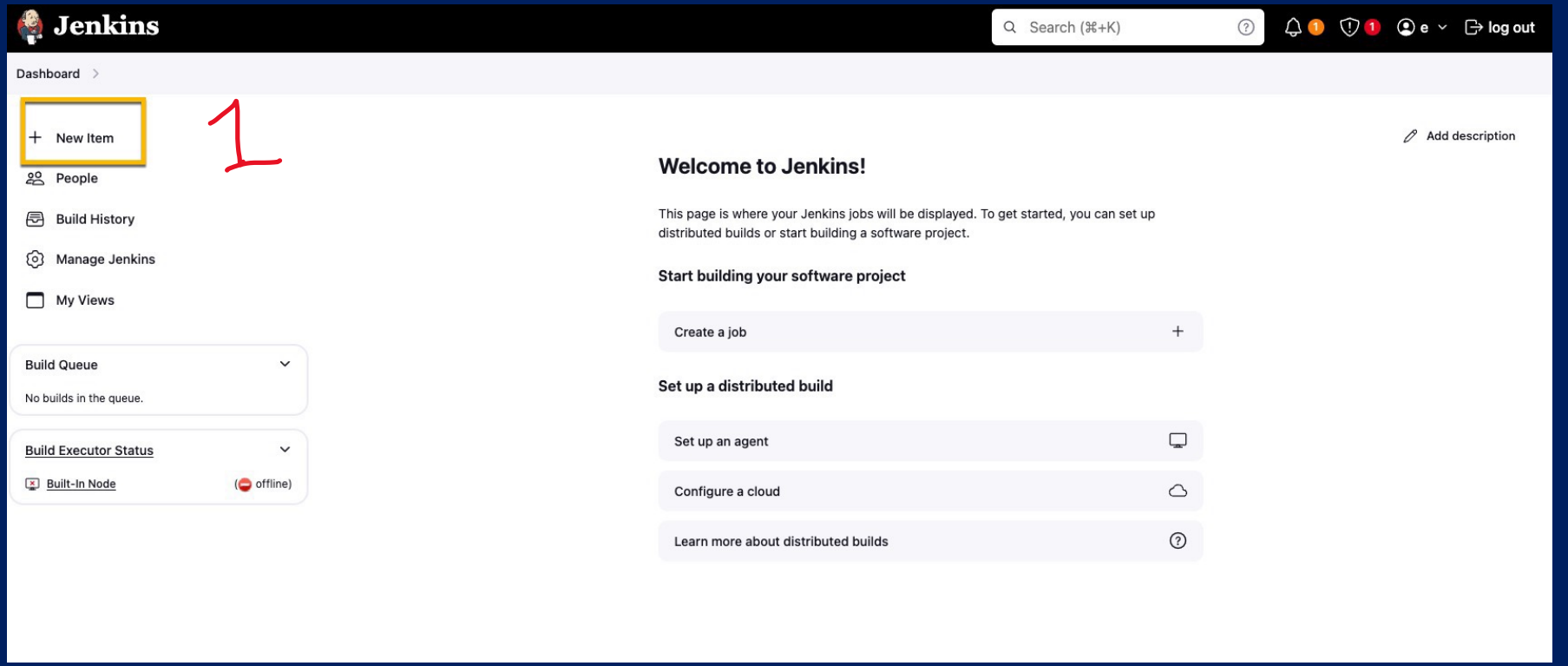

Copy Access Token
When logging in from your Docker CLI client, use this token as a password. Learn more
ACCESS TOKEN DESCRIPTION
a
ACCESS PERMISSIONS
Read, Write, Delete
8
To use the access token from your Docker CLI client:
1. Run docker login -u [redacted]
2. At the password prompt, enter the personal 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)
log out
Dashboard
New Item
1
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

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

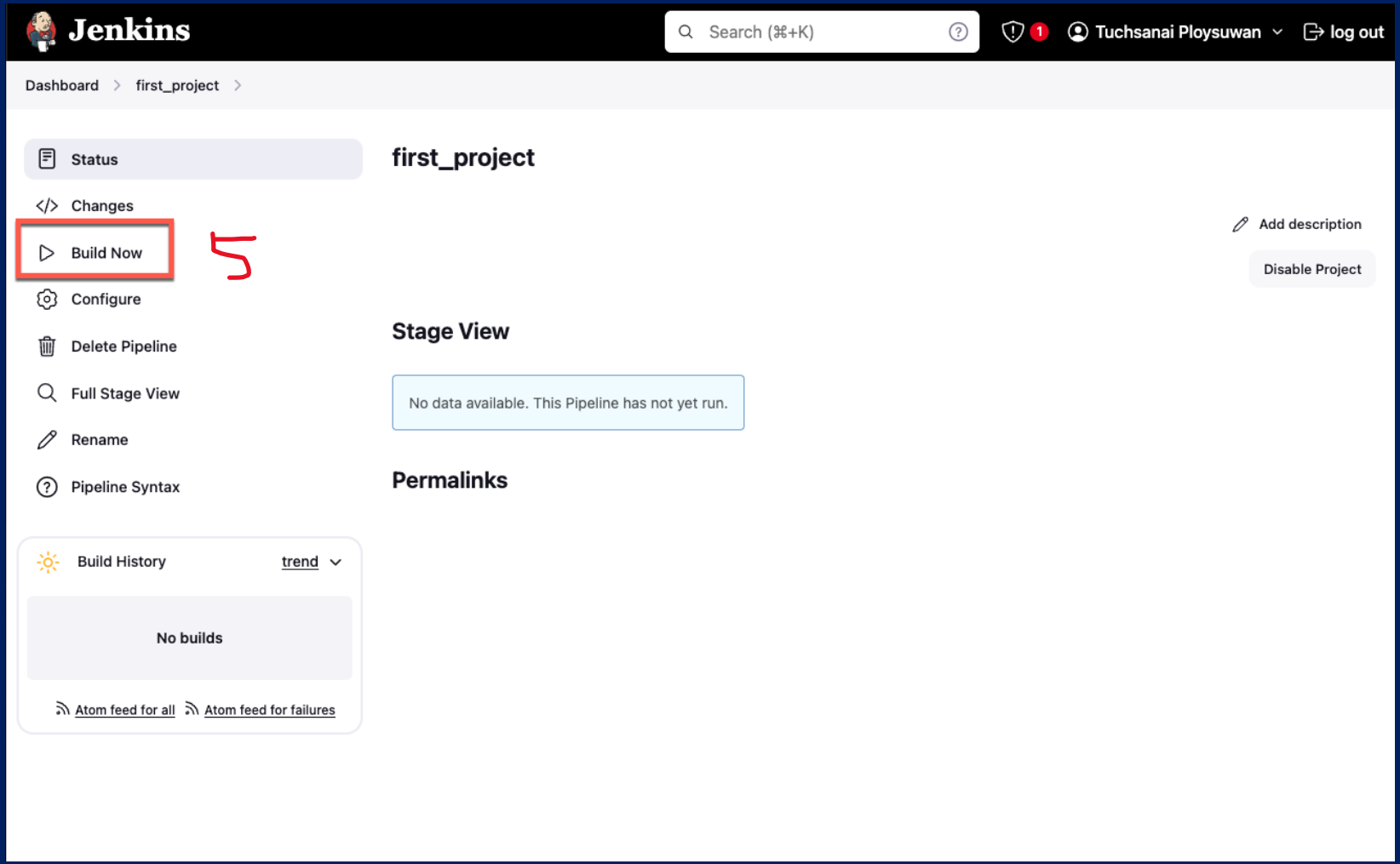
https://github.com/Tuchsanai/DevTools/tree/main/03_Jenkins/03_First_Jenkins

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
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 n.w. 2024 13:05

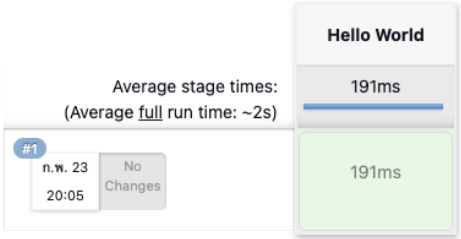
Atom feed for all Atom feed for failures

first_project

Add description

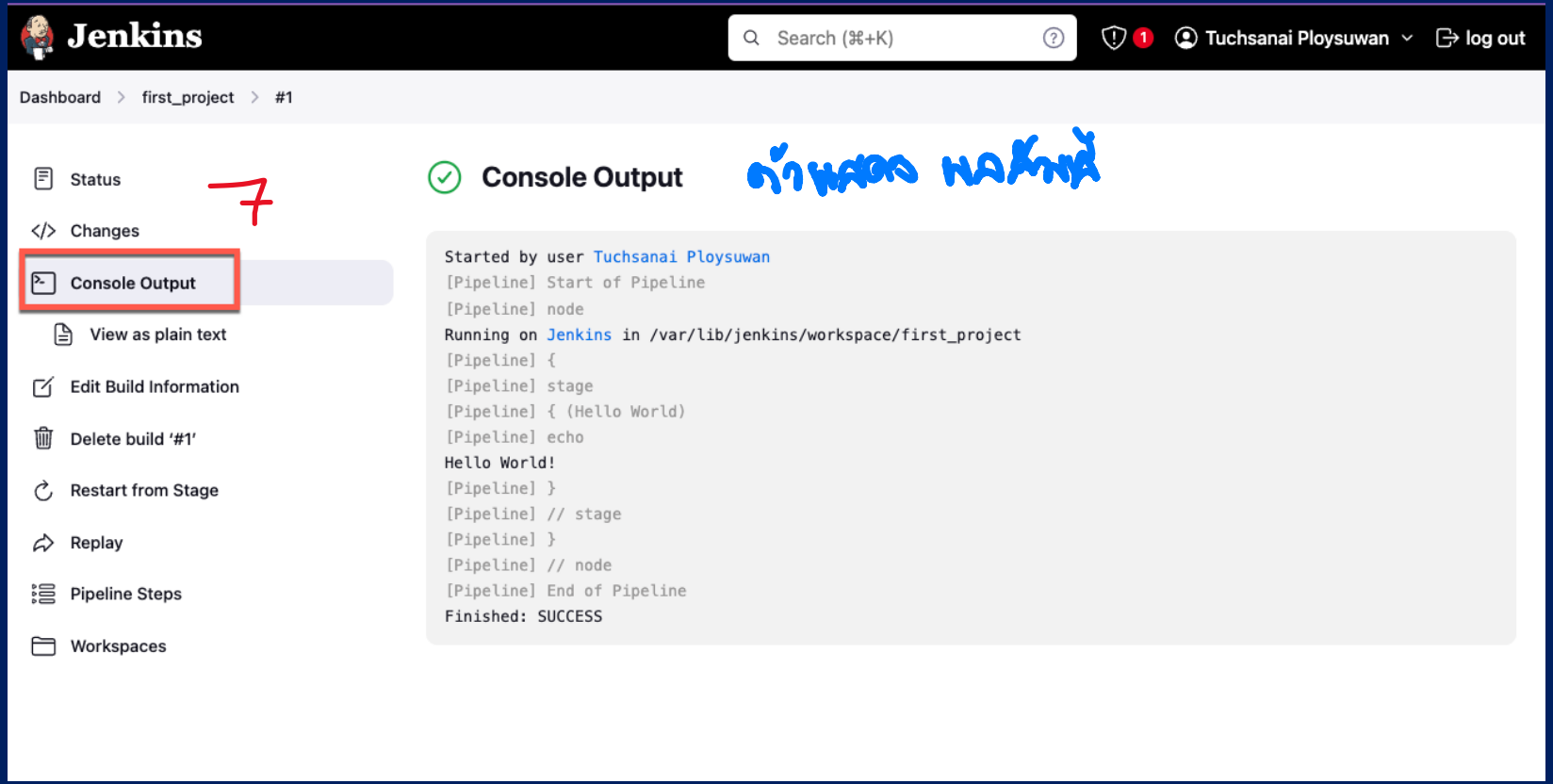
Disable Project

Stage View



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
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 (1)
    stages {
        stage('Build') { (2)
            steps {
                // (3)
            }
        }
        stage('Test') { (4)
            steps {
                // (5)
            }
        }
        stage('Deploy') { (6)
            steps {
                // (7)
            }
        }
    }
}
```

3 stages

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

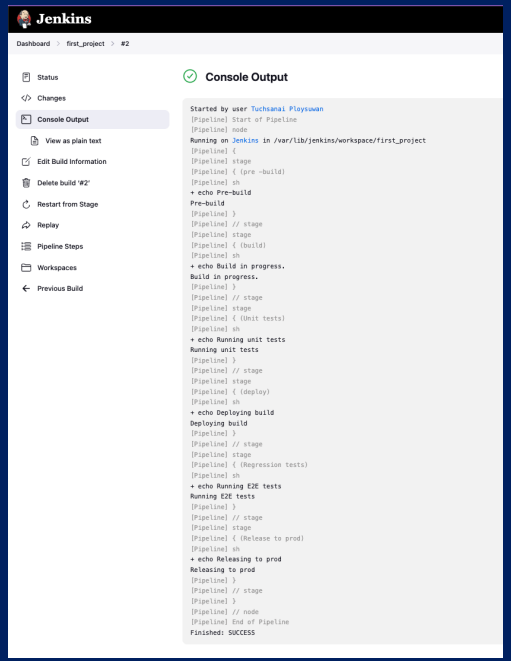
Example 1. Multiple Stages

```
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'
            }
        }
    }

}
```

sh = shell command

Stage View

	Declarative: Checkout SCM	pre -build	build	Unit tests	deploy	Regression tests	Release to prod	Declarative: Post Actions
Average stage times: (Average full run time: ~10s)	1s	482ms	389ms	357ms	359ms	352ms	332ms	65ms
#1 n.ย. 20 22:51 No Changes	1s	482ms	389ms	357ms	359ms	352ms	332ms	65ms



[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

we log out our docker account

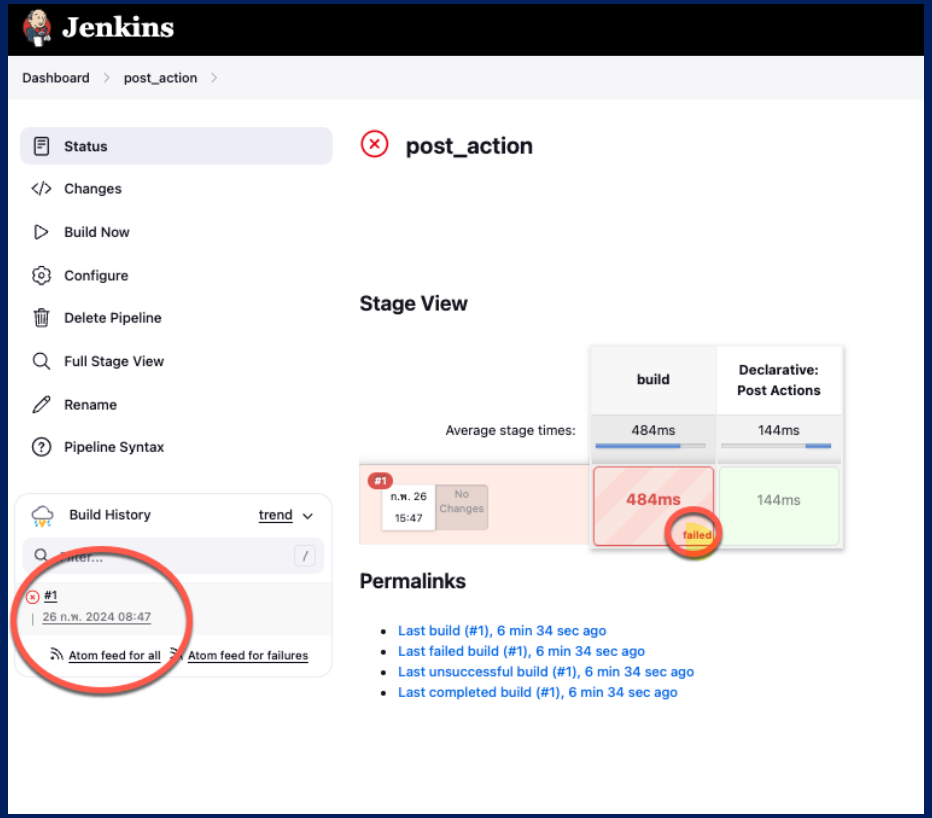
Example 2. Post actions and Conditions

```
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.'
        }
    }
}
```

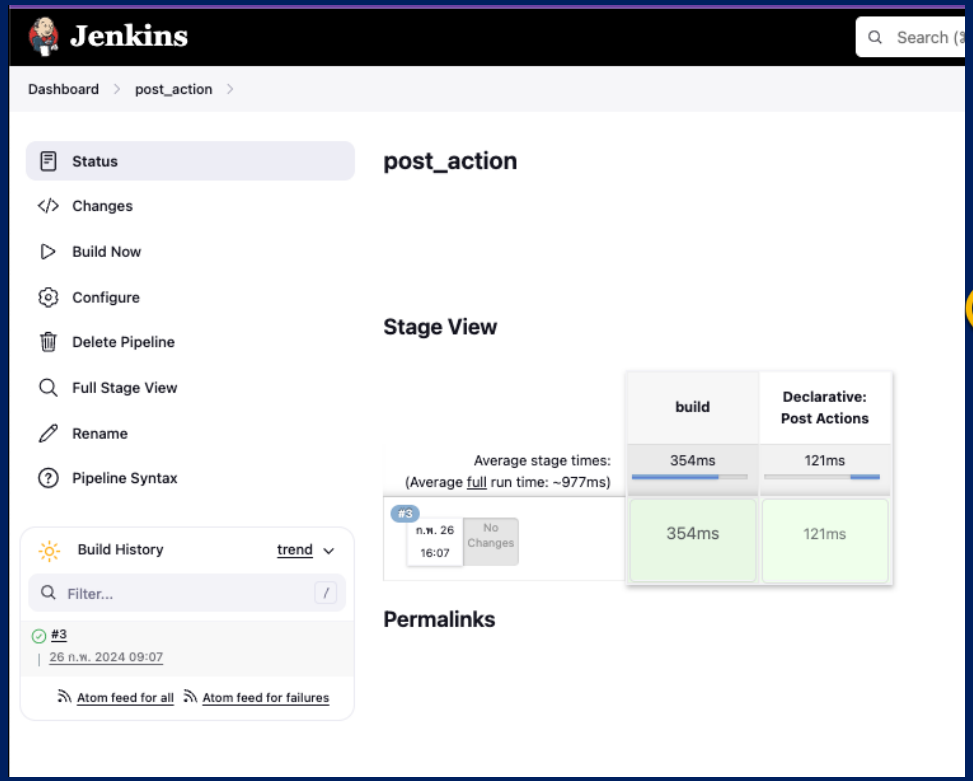
→ เช็ค python เวอร์ชันครับ

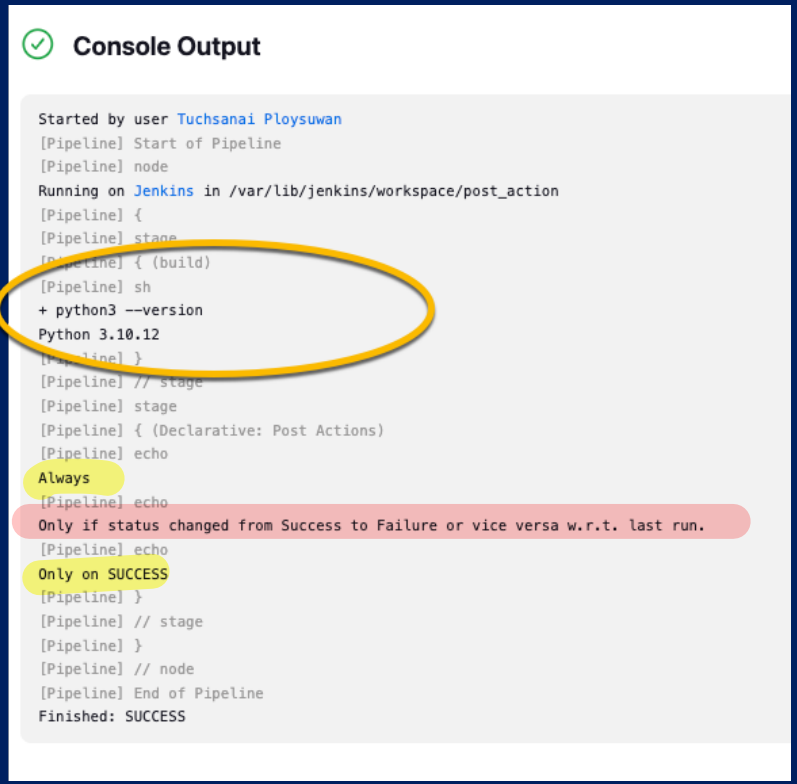
Click: Go to LAB

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'
            }
        }
    }
}
```

→ กำหนดตัวแปร environment

ถ้าเป็น ตัวใหม่ env. ไม่ต้องมี

→ ที่เหลือ ค่าตัวแปร environment var~

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
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!
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

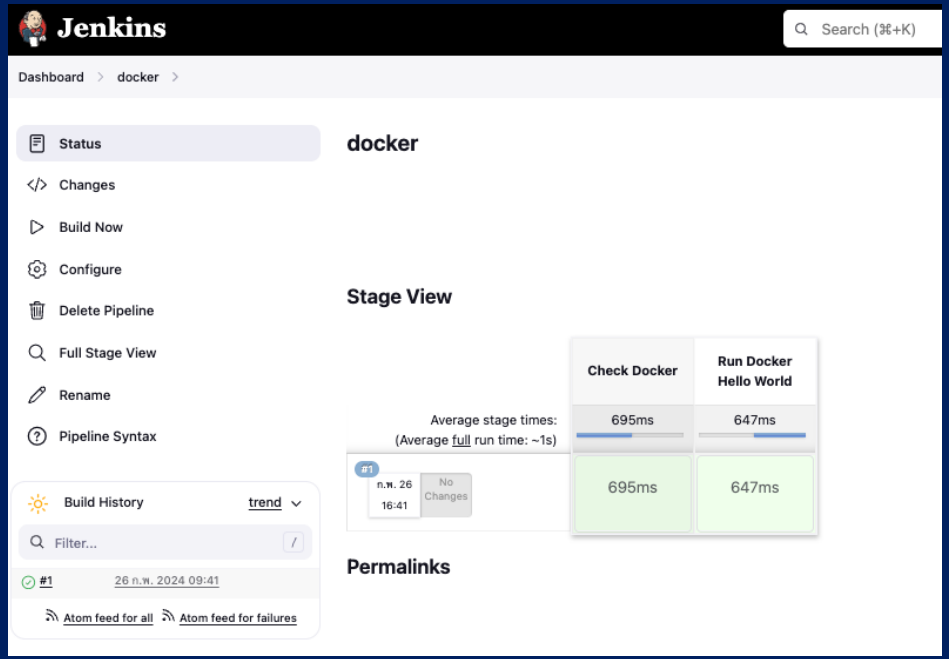
```
Started by user Tuchsanai Ploysuwan
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/env_variable
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Database engine is sqlite
[Pipeline] echo
DISABLE_AUTH 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
```