

{K}ODE{K}LOUD



What is Jenkins

Jenkins is one of the most popular automation tool used worldwide for continuous integration and continuous delivery.

Jenkins is a free and open-source automation server that enables developers to build, integrate, and test code automatically as soon as it is committed to the source repository.

Why Jenkins?

Challenges

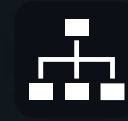
When working on a project with different teams, developers often face issues with different teams using different CI tools, version management, and other tools.

Setting up a CI/CD toolchain for each new project will lead to certain challenges like:

- Slower Releases
- Manual Builds
- Non-repeatable processes
- No Automations



Slower Releases



Manual Builds



Non-repeatable
processes



No automations

Why Jenkins?

Solution

Jenkins is the solution to those challenges.
It provides:

- Automated builds
- Automated Tests
- Automated CI/CD pipelines
- Automated deployments
- Ability to install Jenkins locally
- Jenkins support and Plugins



Automated builds



Automated Tests

CI

CD

Automated CI/CD
pipelines



Automated
deployments



Ability to install
Jenkins locally



Jenkins support
and Plugins

Why Jenkins?



Open-source



1000+ plugins



Free



Paid, Enterprise

Free!

Jenkins is free and you don't have to pay for anything.
Jenkins can be hosted on a Virtual Machine, a container. Or
even locally for development purposes.



Plugins

Jenkins is well tested and provide several integrations with 1800+ plugins to support build, deployment and automation for the project



The banner features the Jenkins robot mascot on the left, a large 'Plugins Index' title in the center, and a search bar below it. The search bar has a 'Browse' button and a magnifying glass icon.

Plugins Index

Discover the 1800+ community contributed Jenkins plugins to support building, deploying and automating any project.

[Browse](#) Find plugins...

Browse categories	New Plugins	Recently updated	Trending
Platforms	CloudBees Feature Management	MSTestRunner	Localization: Chinese (Simplified)
User interface	Artifactz.io	Deployed On Column	Pipeline: GitHub Groovy Libraries
Administration	Jakarta Activation API	InfluxDB	Timestamper
Build management	Jakarta Mail API	MATLAB	Infrastructure for Publish Over X
Source Code Management	JavaMail API	Cloudify	Gradle
	JavaBeans Activation Framework (JAF) API	Allure	Role-based Authorization Strategy
	Jersey 2 API	InsightVM Container Image Scanner	GitLab
	Shutdown Queue	Build Monitor View	Build Timeout
	BMC DevOps for CFA	URLTrigger	NodeJS
	XTrigger API	XTrigger API	Workspace Cleanup

Enterprise Options



Support



Managed
Service

The screenshot displays the CloudBees website with a navigation bar containing links to Platform, Use Cases, Customer Stories, Resources, Developers, and Why CloudBees. A purple banner below the navigation bar states 'CloudBees Ranked #11 in G2 Top 50 Enterprise Software P'. The main content area has a blue background with colorful hexagonal shapes and features the heading 'Continuous Integration for the Enterprise'. Below the heading, a paragraph reads: 'Built on the most widely used automation server in the world Jenkins™ - CloudBees Continuous Integration provides flexible, scalable and governed CI/CD you can trust.' At the bottom of the section are two buttons: 'Schedule a Demo' and 'Contact Sales'.

CloudBees. Platform Use Cases Customer Stories Resources Developers Why CloudBees

CloudBees Ranked #11 in G2 Top 50 Enterprise Software P

Continuous Integration for the Enterprise

Built on the most widely used automation server in the world Jenkins™ - CloudBees Continuous Integration provides flexible, scalable and governed CI/CD you can trust.

[Schedule a Demo](#) [Contact Sales](#)

Continuous Integration

Continuous Integration is a process in which the code is merged from multiple contributors and added to a single repository.

In simple words, CI is a process to take the code package it and send it to the CD for further processing.

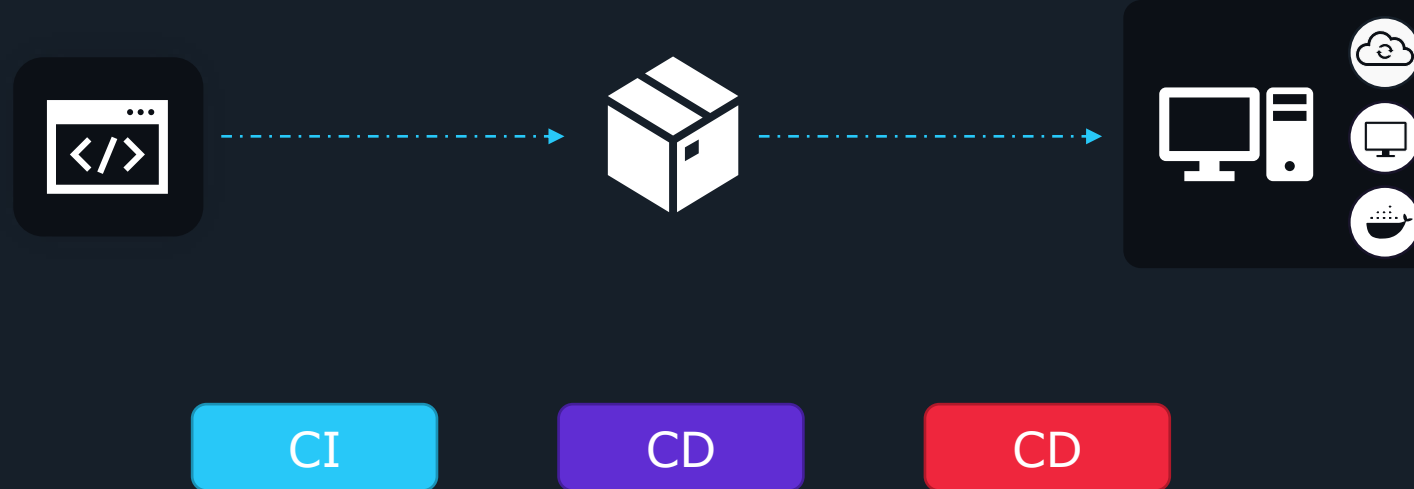
Continuous Deployment

Continuous Deployment is an automated process in which the code is taken from the repository and deployed to the system.

Continuous Integration and Continuous Delivery/Deployment (CI/CD)

CI/CD in simple words is a process to take a code, package it up and deploy it to a system that can be serverless, a VM, or a container.
CI/CD can be broken down into 3 steps:

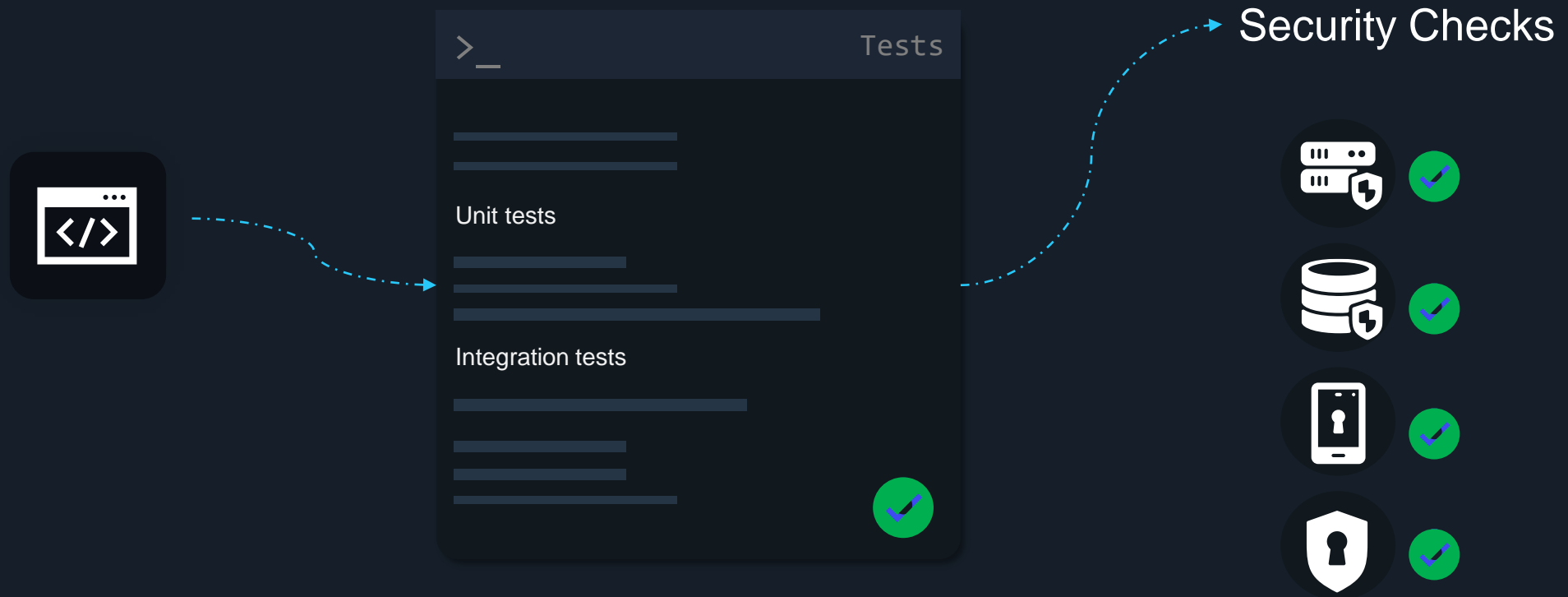
- CI – Continuous Integration
- CD – Continuous Delivery
- CD – Continuous Deployment



The Key Pieces of CI

Key Processes of Continuous Integration

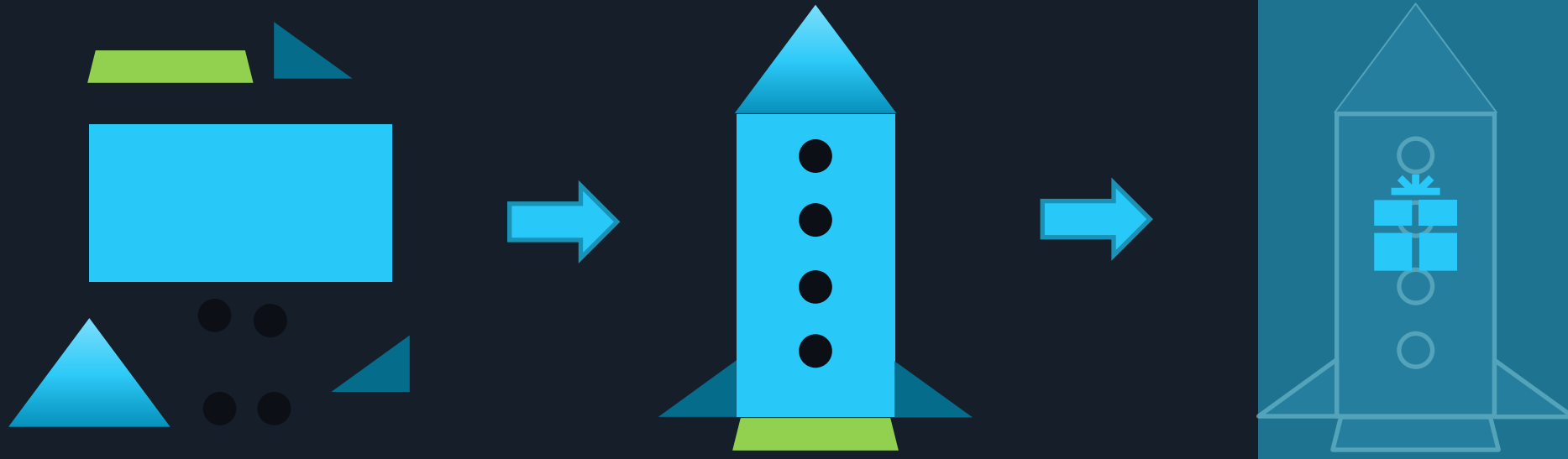
- Package up the code
- Test the code (run unit tests, integration tests, etc)
- Run security checks against the code



Continuous Integration (CI)

Think of the Continuous Integration process like a gift you're wrapping

- The gift comes in pieces
- You put the gift together (maybe a toy chest/box)
- The gift gets wrapped in wrapping paper
- You put it in the car and deliver it to the person.

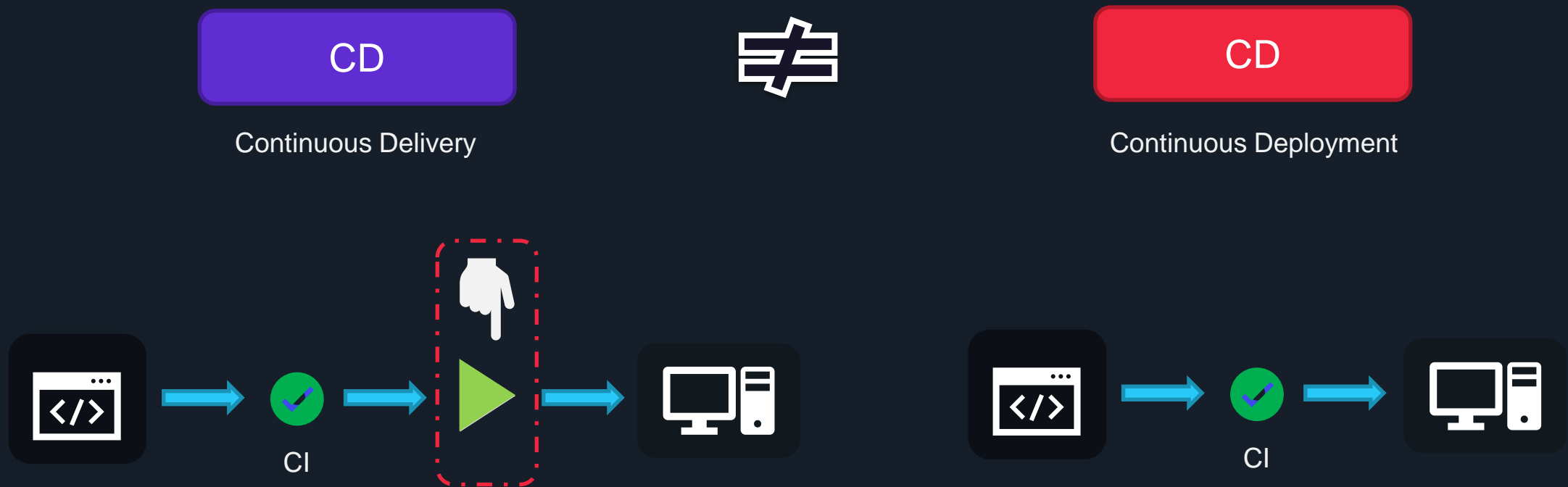


Continuous Integration (CI)



Continuous Deployment vs Continuous Delivery

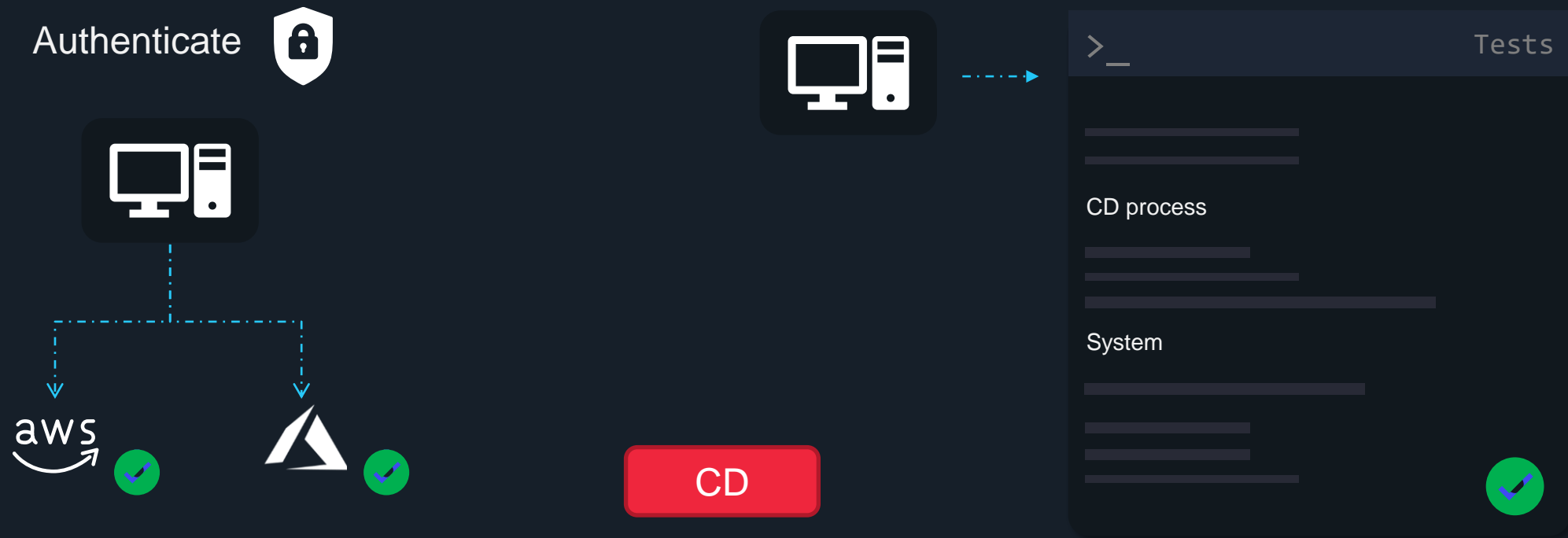
The basic difference between Continuous Delivery and Continuous Deployment is that in Continuous Delivery to deploy the code after the CI process you have to manually trigger it via some button to deploy on the system whereas in Continuous Deployment this process is automatic.



Key Pieces of CD

Key Pieces of CD:

- Ensure you're authenticated to the system or wherever you're deploying
- Ensure that the code that's being deployed is working as expected once it's deployed



Installing Jenkins



Ubuntu 20.04



<https://www.jenkins.io/doc/book/installing/linux/>

Install Jenkins on Ubuntu

Step 1: Install Java on Ubuntu

```
$ sudo apt update  
$ sudo apt install openjdk-8-jdk
```

Alternatively, install version 11:

```
$ sudo apt install openjdk-11-jdk
```

Confirm the download by pressing Y and Enter

Step 2: Add the repository key to the system:

```
$ sudo apt wget -q -O -  
https://pkg.jenkins.io/debian-  
stable/jenkins.io.key | sudo apt-key add -
```

Step 3: Once the key is added with no errors, append the Debian package repository address

```
$ sudo sh -c 'echo deb  
http://pkg.jenkins.io/debian-stable  
binary/ >  
/etc/apt/sources.list.d/jenkins.list'
```

Step 4: Run update to use new repository

```
$ sudo apt update
```

Step 5: Install Jenkins

```
$ sudo apt install jenkins
```

Start Jenkins on Ubuntu

After successful installation let us start Jenkins

```
$ sudo systemctl start jenkins
```

The above command will not display any output

To check the running status of Jenkins use the below command which should show active status on run

```
$ sudo systemctl status jenkins
```

Jenkins Plugins

Plugins are used in Jenkins to enhance Jenkins functionality and cater to user-specific needs. Just like how Gmail, Facebook and LinkedIn help you connect your one service to another, plugins also work the same way and allow us to connect one service to other services and work with other products.



User



Login with Google



Login with Facebook



Login with LinkedIn



Website

Plugins

For example, you want to connect to Azure from Jenkins you would need to download Azure Plugin which will allow you to connect to Azure at a programmatic level.

Similarly, we can have other integrations with AWS, GitHub, etc using plugins.



Azure Plugin



AWS Plugin



Github Actions Plugin



Install Plugins

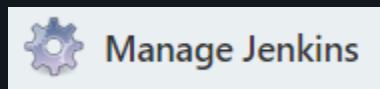
To install a new plugin in Jenkins


- 1) Go to Manage Jenkins -> Manager Plugins
- 2) Click Available and search for the desired plugin.
- 3) Select the desired plugin and Install.

Note: Few plugins may need a restart

To restart Jenkins


```
$ sudo systemctl restart jenkins
```





Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

 There are updates available



Updates

Available

Installed

Advanced

Install ↑

Name

☐

Amazon Web Services SDK :: All

api-plugin

aws

This plugin provides all AWS SDK for Java modu



Install without restart

OR

Download now and install after restart

Update Plugins

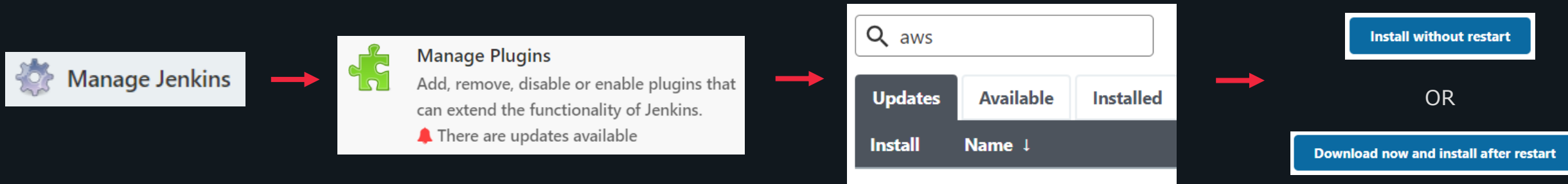
To update any existing plugin in Jenkins

- 1) Go to Manage Jenkins -> Manager Plugins
- 2) Click Updates and search for the desired plugin.
- 3) Select the desired plugin and Install.

Note: Few plugins may need a restart

To restart Jenkins

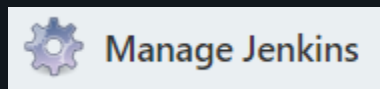
```
$ sudo systemctl restart jenkins
```




Delete Plugins

To delete any plugin in Jenkins


- 1) Go to Manage Jenkins -> Manager Plugins
 - 2) Go to Installed and search for the desired plugin.
 - 3) Click on uninstall button for the plugin you want to delete.
- Click yes to proceed with the deletion.





Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

 There are updates available



filter				
Updates Available Installed Advanced				
Enabled	Name	Version	Previously installed version	Uninstall
<input checked="" type="checkbox"/>	Git client plugin Utility plugin for Git support in Jenkins	3.11.0		Uninstall
<input type="checkbox"/>	Git plugin This plugin integrates Git with Jenkins.	4.10.3		Uninstall

Jenkins Visuals



Visuals

Jenkins
Menu

Build

User
account

Navigate

Jenkins Jobs

Different types of jobs that can be created in Jenkins:

1) Freestyle project

This is a central feature of Jenkins. It will build the project, combine SCM with the build system. It can also be used for things other than building applications.

2) Pipeline

This is used to create a pipeline

3) Multi-configuration project

This is great if you need a large number of Jenkins configurations if you need multiple environments like Dev/ UAT.

4) Folder

This creates containers and stores nested items. It is useful in grouping, creating a namespace, etc.

5) Organisation folder

Creates a multibranch project for all different subfolders that are available.

6) Multibranch Pipeline

It sets up pipeline projects for different repositories.

Administering Jenkins



Backup



Restore



Monitor



Scale



Manage

Backup and Restore



Backup

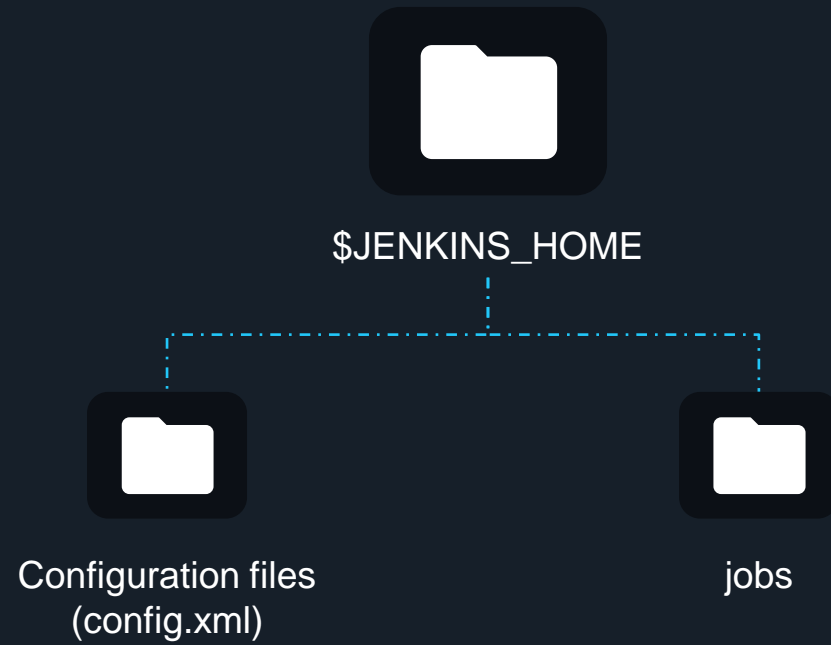


Full



Snapshots

Which Files To Backup?



Backup Jenkins

It is crucial to have adequate backups of your Jenkins instance. Backups are used to recover from accidental configuration changes. Recovering a file that has been mistakenly erased or has been corrupted. Or just to recover a previous setup.

There are two ways we can backup Jenkins:

- 1) Using Plugins
- 2) Using custom shell script

Backup Jenkins

To backup Jenkins using a plugin, you will first need to install a backup plugin. Some of the most commonly used plugins are ThinBackup, Periodic Backup, Google cloud Backup.

For backing up using any of these plugins there are a few general steps that must be followed:

- 1) Creating a backup directory with read and write access
- 2) Selecting files that need backup

Backing up using shell script

Please check out these popular repositories for your reference:

- 1) repository: <https://github.com/sue445/jenkins-backup-script>
- 2) gist: <https://gist.github.com/abayer/527063a4519f205efc74>

Jenkinsfile

Jenkinsfile is a text file that contains definitions. This could be templates or instructions. It tells pipelines what they should be doing and what services and plugins they should be interacting with.

Components of Jenkinsfile:

- 1) Pipeline – The task you are trying to accomplish
- 2) Build Agent –The place where you run your pipeline
- 3) Stages – Staging/Production/UAT
- 4) Steps –Work done in the pipeline

What Is A Jenkinsfile?

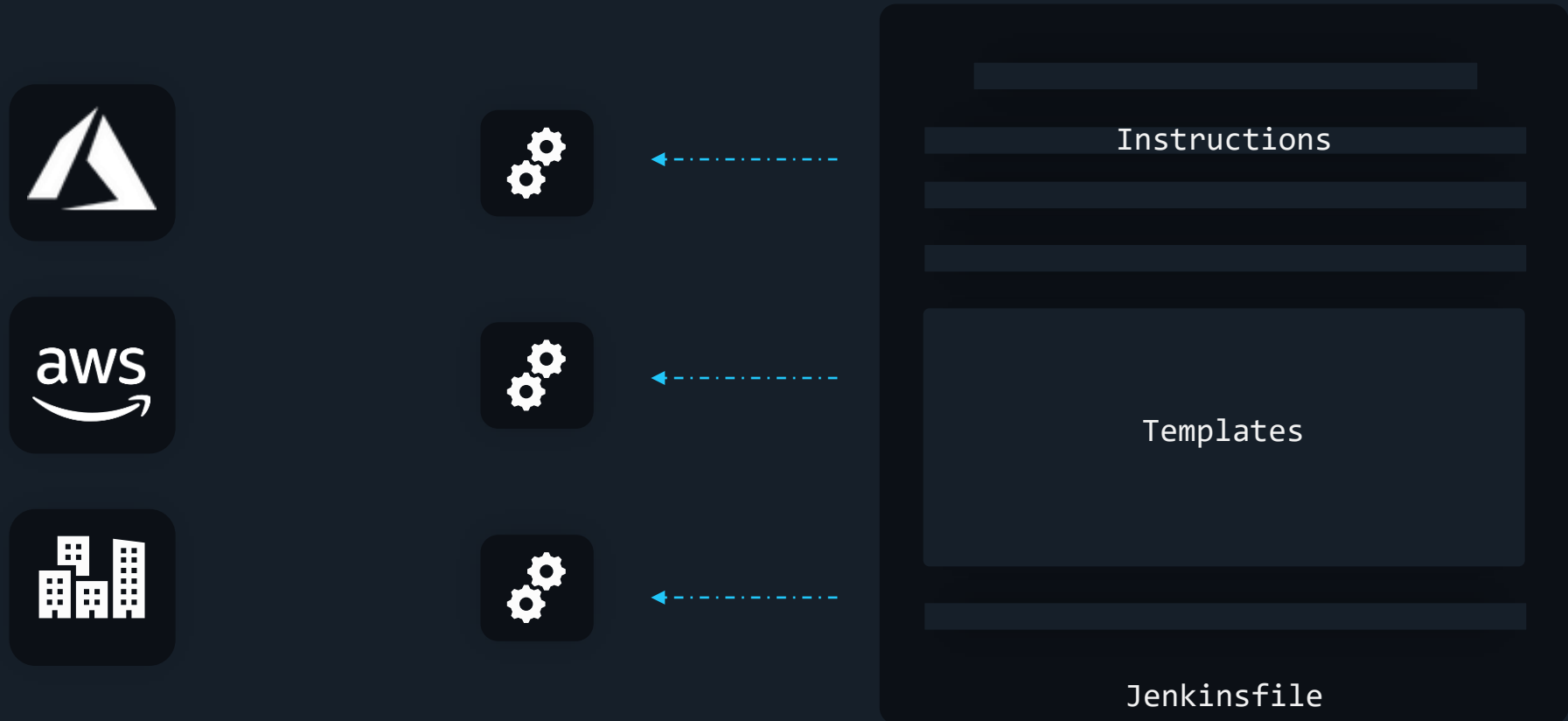


Instructions

Templates

Jenkinsfile

What Is A Jenkinsfile?



Components of Jenkinsfile

The task you are trying to accomplish

Build Agent

Stages

Steps

```
Jenkinsfile

pipeline {
  agent any

  stages {
    stage('Build') {
      steps {
        echo 'Building..'
      }
    }
    stage('Test') {
      steps {
        echo 'Testing..'
      }
    }
    stage('Deploy') {
      steps {
        echo 'Deploying....'
      }
    }
  }
}
```



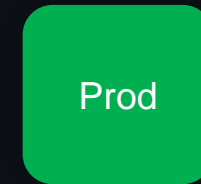
Jenkinsfile



Jenkinsfile

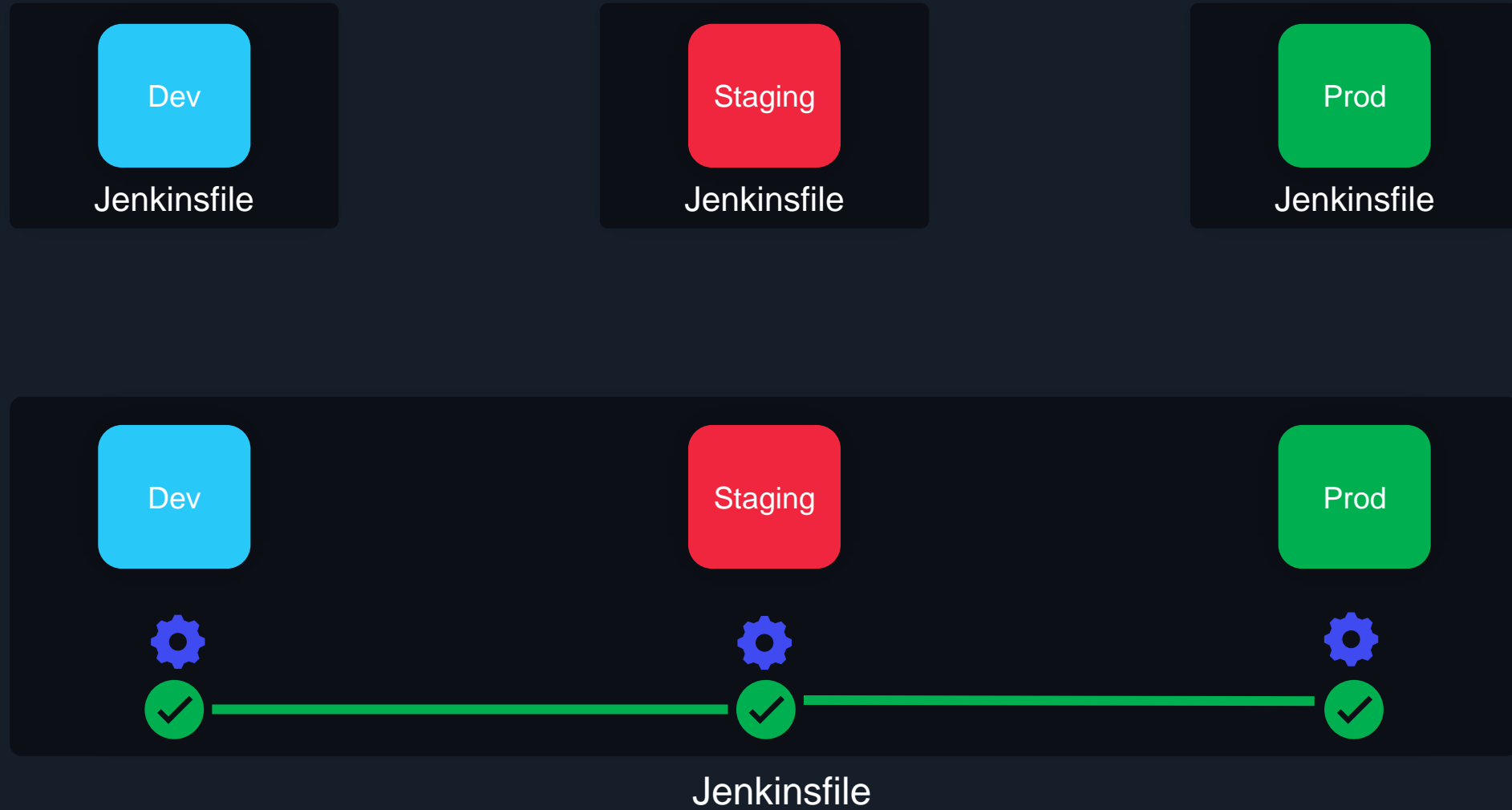


Jenkinsfile



Jenkinsfile

Multi-Stage Pipelines



Build Agents



Windows



Linux



MacOS

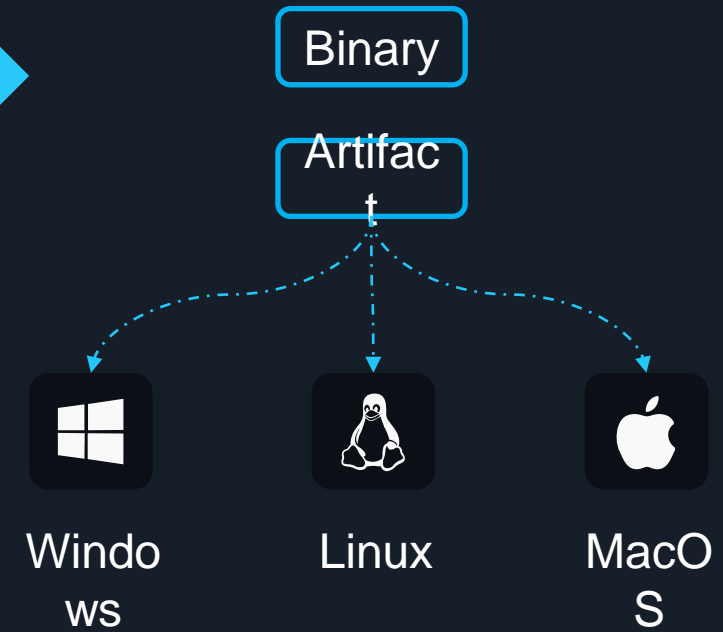
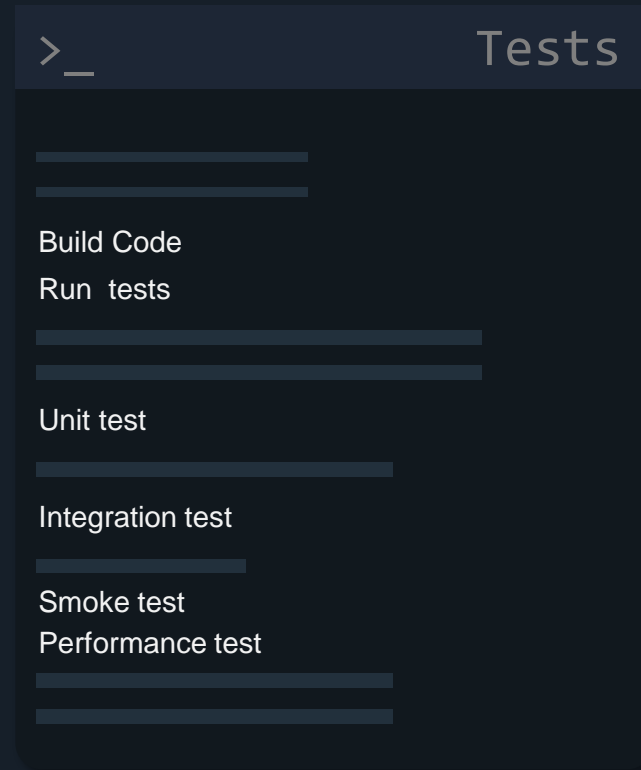


Docker



What Are Build Agents?

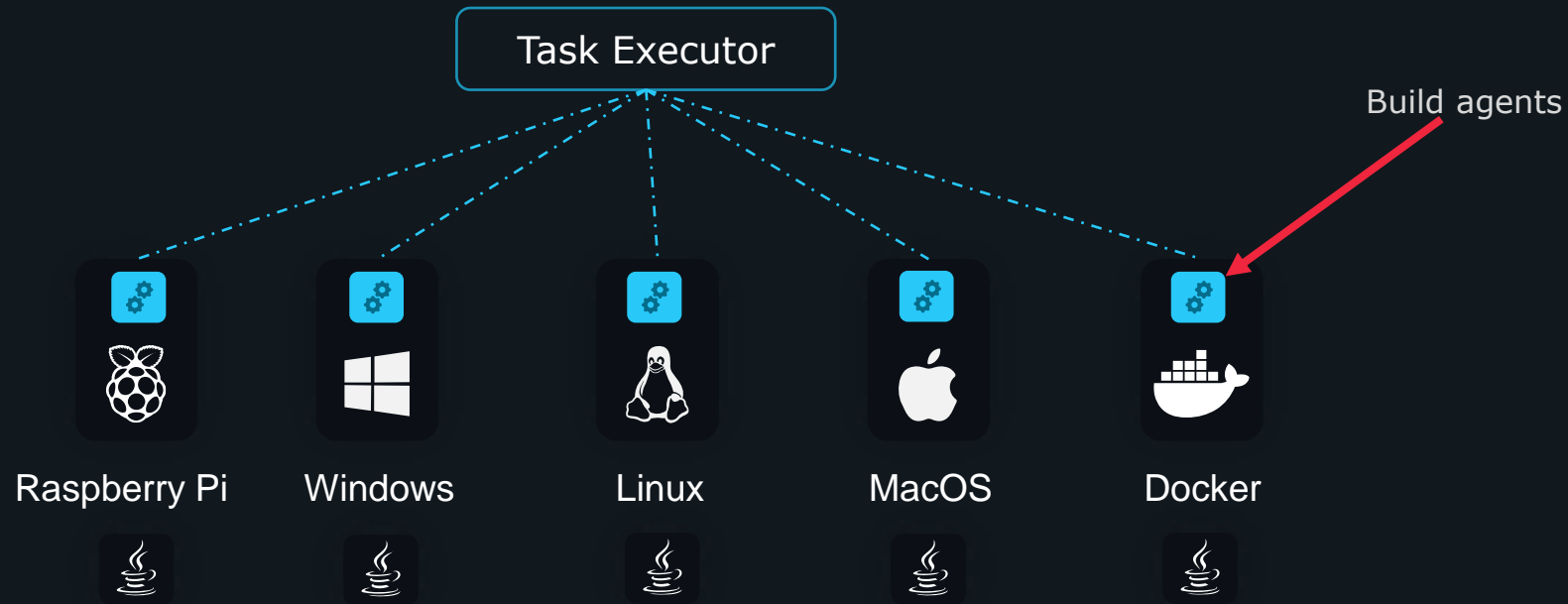
CI Pipeline



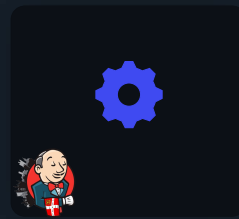
Build Agents

Build Agents are systems that run the processes throughout the pipelines.

Build agents help in building codes, deploying, and running automated tests. It is a system that runs the entire workload.



Running Builds on Same Server



Jenkins Server

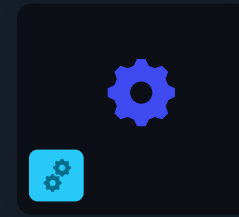


Not Recommended

Separate Build Server



Jenkins Server

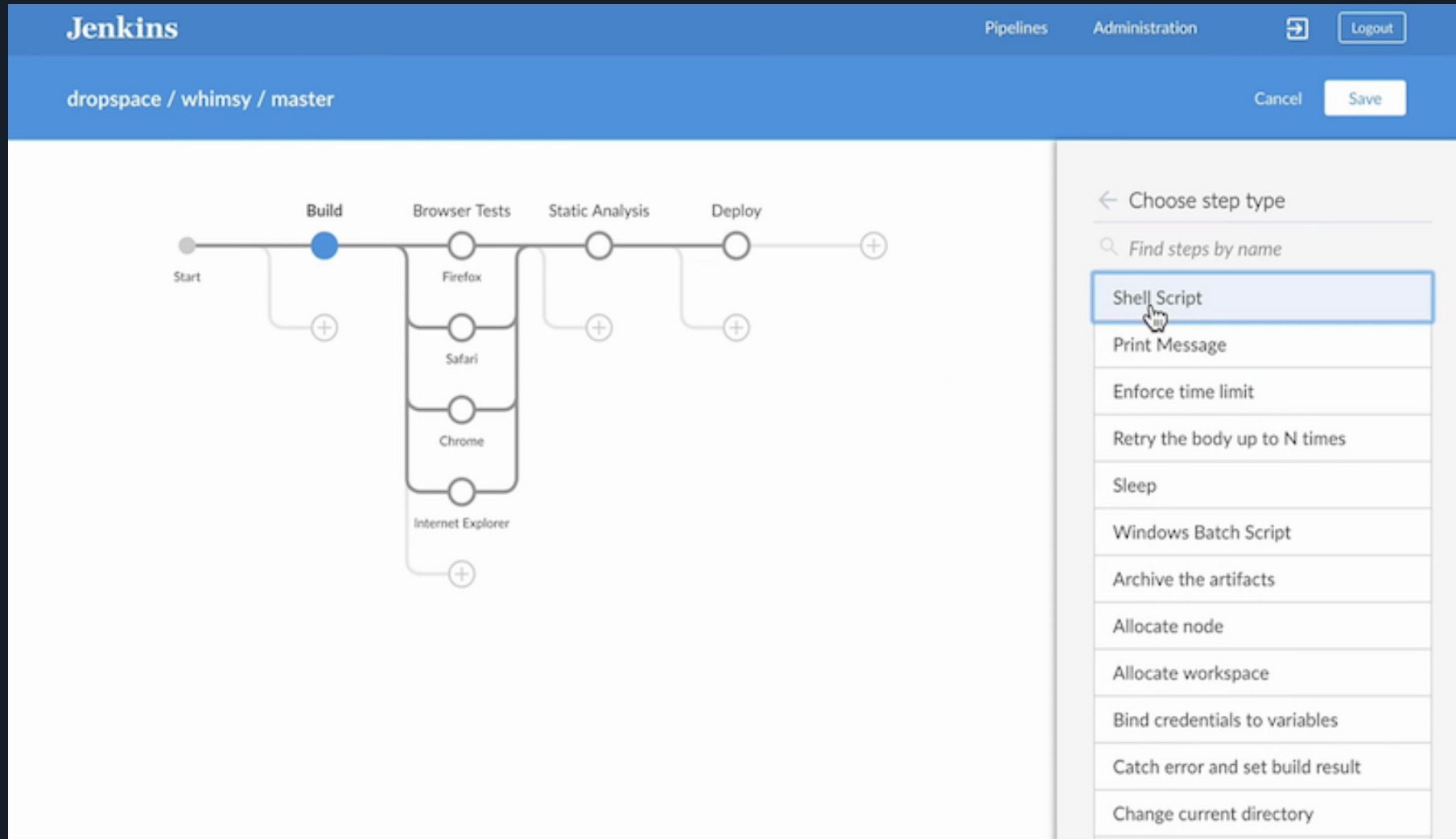


Build Server



Recommended

Blue Ocean CICD



BlueOcean; New & Improved CICD!


The whole idea of BlueOcean is a new UI experience for CICD in Jenkins

- Jenkins was definitely falling behind from a UI standpoint
- There were a ton of other CICD tools that felt much easier to use from a UI perspective
- BlueOcean is meant to changed that narrative

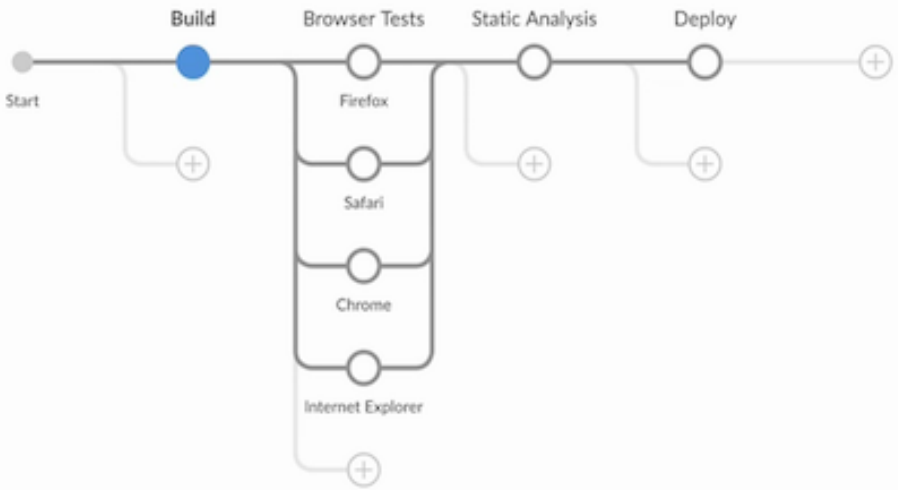
What Are We Getting Out Of BlueOcean?

- Sophisticated visualizations of continuous delivery (CD) Pipelines, allowing for fast and intuitive comprehension of your Pipeline's status.
- Pipeline editor - makes the creation of Pipelines approachable by guiding the user through an intuitive and visual process to create a Pipeline.
- Personalization to suit the role-based needs of each member of the team.
- Pinpoint precision when intervention is needed and/or issues arise. Blue Ocean shows where in the pipeline attention is needed, facilitating exception handling and increasing productivity.
- Native integration for branch and pull requests, enables maximum developer productivity when collaborating on code with others in GitHub and Bitbucket.

Blue Ocean

Jenkins Pipelines Administration  Logout

dropspace / whimsy / master Cancel Save



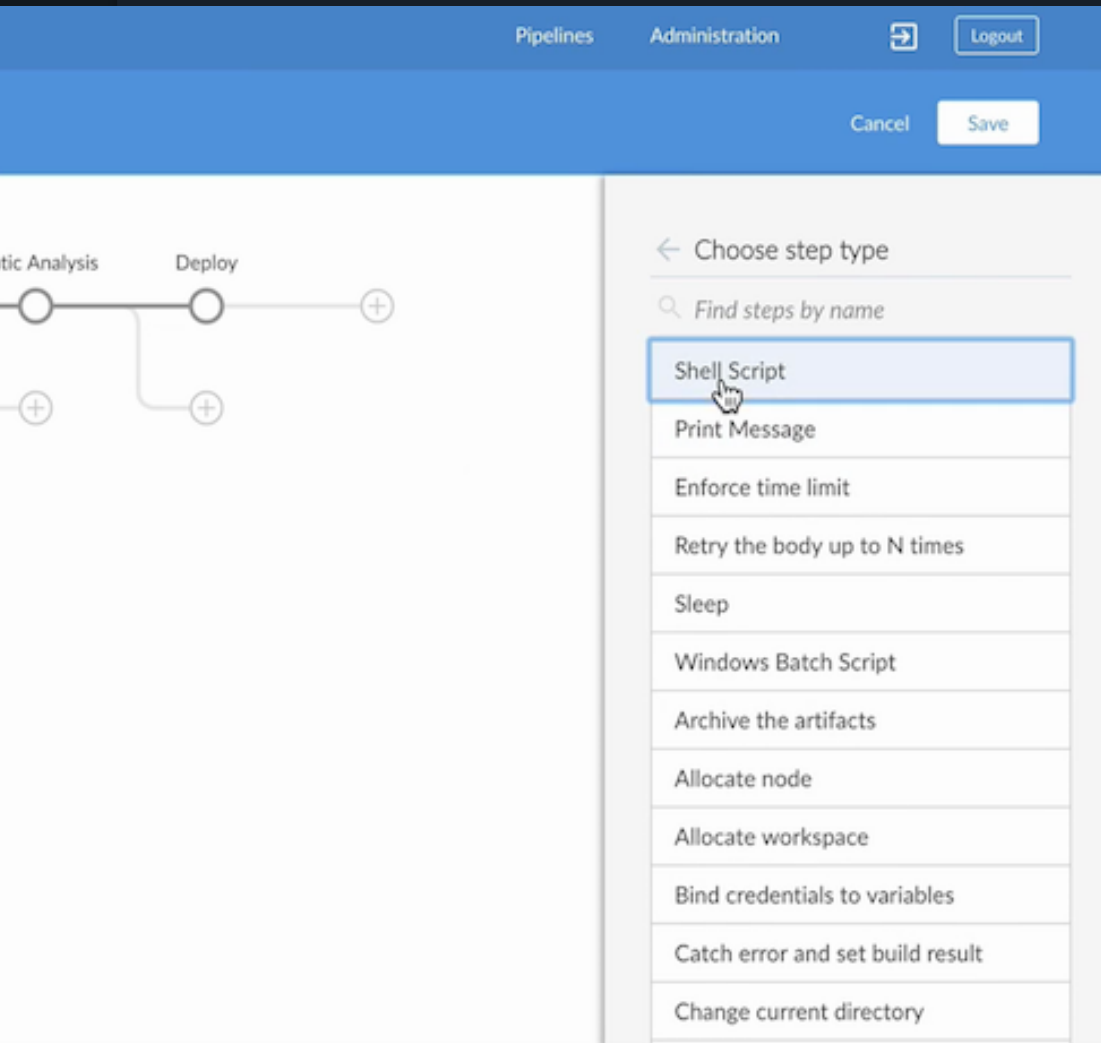
```
graph LR; Start((Start)) --> Build((Build)); Build --> BrowserTests[Browser Tests]; BrowserTests --> StaticAnalysis[Static Analysis]; StaticAnalysis --> Deploy((Deploy)); Deploy --> End((+)); BrowserTests --> Firefox((Firefox)); BrowserTests --> Safari((Safari)); BrowserTests --> Chrome((Chrome)); BrowserTests --> IE((Internet Explorer)); Build --> Plus1((+)); StaticAnalysis --> Plus2((+)); Deploy --> Plus3((+)); IE --> Plus4((+));
```

Choose step type

Find steps by name


- Shell Script
- Print Message
- Enforce time limit
- Retry the body up to N times
- Sleep
- Windows Batch Script
- Archive the artifacts
- Allocate node
- Allocate workspace
- Bind credentials to variables
- Catch error and set build result
- Change current directory

Blue Ocean



- **Easier to use**
- **Sophisticated visualizations**
- **Fast and intuitive Pipeline status**
- **Pipeline editor**
- **Personalization**
- **Pinpoint precision**
- **Native integration for branch and pull requests**

Jenkins Security


 **Jenkins**

search ?

?

?

Dashboard > Configure Global Security

 **Configure Global Security**

Authentication

☐ Disable remember me

Security Realm

☐ Delegate to servlet container

☒ Jenkins' own user database

☐ Allow users to sign up

☐ None

?

?

?

Authorization

☐ Anyone can do anything

☐ Legacy mode

☒ Logged-in users can do anything

☐ Allow anonymous read access

?

?

?

?

Markup Formatter

Markup Formatter

Plain text

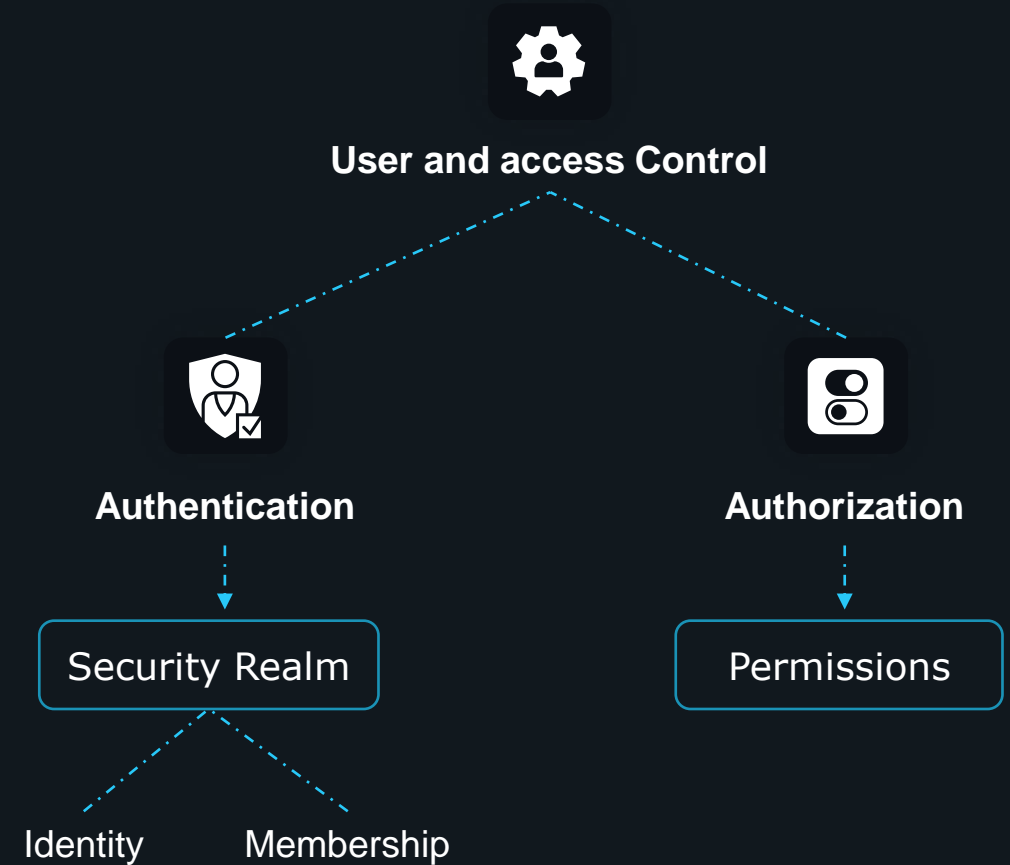
Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

Save Apply

Jenkins Security

Jenkins access control is split into two parts:

- 1) Authentication** (users prove who they are) is done using a security realm. The security realm determines user identity and group memberships.
- 2) Authorization** (users are permitted to do something) is done by an authorization strategy. This controls whether a user (directly or through group memberships) has a permission



Common Jenkins Security Mistakes

Anyone can do anything

This authorization strategy is very rarely a good choice, as it allows even anonymous users to administer Jenkins. As a rule of thumb, it should not be used. Never rely on the Jenkins URL to not be known outside your team or organization alone for security.

Logged-in users can do anything

This authorization strategy can be a sensible choice as long as only fully trusted users have accounts to access Jenkins. This is the default with Jenkins's single admin user when setting up Jenkins with the setup wizard.

Switching to an authentication realm that allows untrusted users to have an account later will result in those users getting administrative access to Jenkins if you keep this authorization strategy. Examples include enabling account signup for *Jenkins' own user database*, or various other authorization realms, many of which (GitHub, Google, GitLab, etc.) allow anyone to sign up for an account.

Anonymous and authenticated users

Similar to the previous items, you should generally not grant significant permissions to anonymous (the anonymous user) or authenticated (any authenticated user) when using an authorization strategy that allows finer-grained control (like [Matrix Authorization Strategy](#)).

Granting Overall/Administer permission to *anonymous* is similar to *Anyone can do anything*, while granting that permission to *authenticated* is essentially the same as *Logged-in users can do anything*.

Built-in node

Users with limited permissions must not be able to configure jobs that run on the built-in node. When setting up a new Jenkins instance, adding users and switching authorization strategies, it is important to also set up distributed builds and limit what jobs are able to run on the built-in node.

ext

Jenkins Security

Common security mistakes that often happens when using Jenkins



- **Anyone can do anything**
- **Logged-in users can do anything**
- **Anonymous and authenticated users**
- **Built-in node**

References

- 1) <https://github.com/AdminTurnedDevOps/go-webapp-sample>
- 2) <https://github.com/AdminTurnedDevOps/Go-Demo-App>

{K}ODE{K}LOUD