

# FULL STACK



## CI/CD Pipeline with Jenkins

# FULL STACK

## Automated Deployment





# Learning Objectives

By the end of this lesson, you'll be able to:

- 🕒 Implement automated and continuous deployment
- 🕒 Deploy a Python application to an application server
- 🕒 Launch a simple java web application using Tomcat
- 🕒 Deploy scripting-based applications like Ruby and PHP



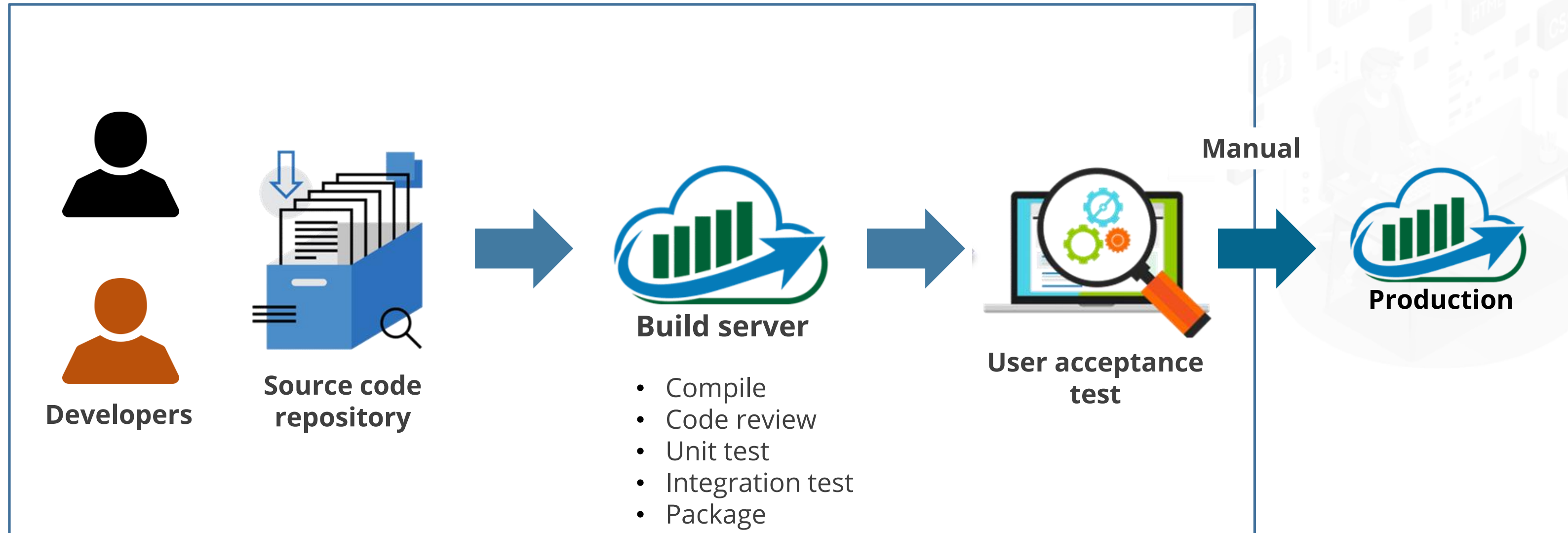
# FULL STACK

## Introduction to Automated Deployment and Continuous Delivery

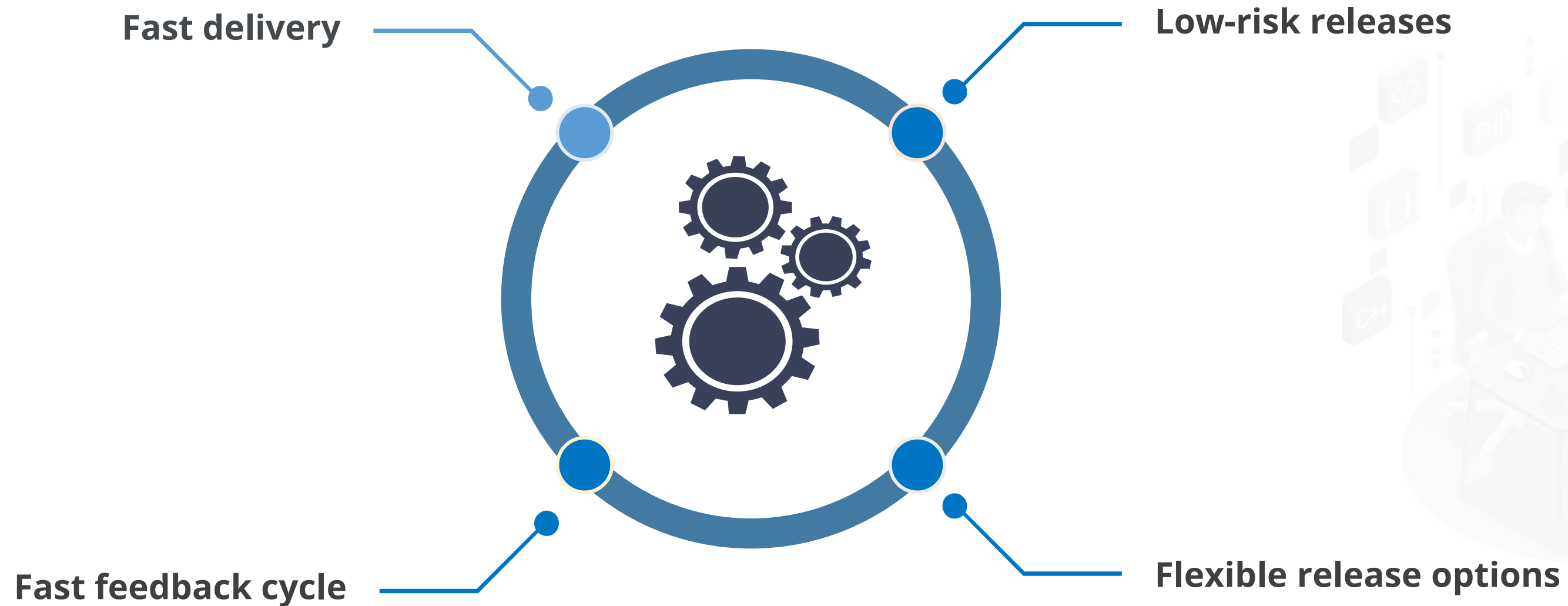
# Continuous Delivery

Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes, and experiments—into production, or into the hands of users, safely and quickly in a sustainable way.

–Jez Humble

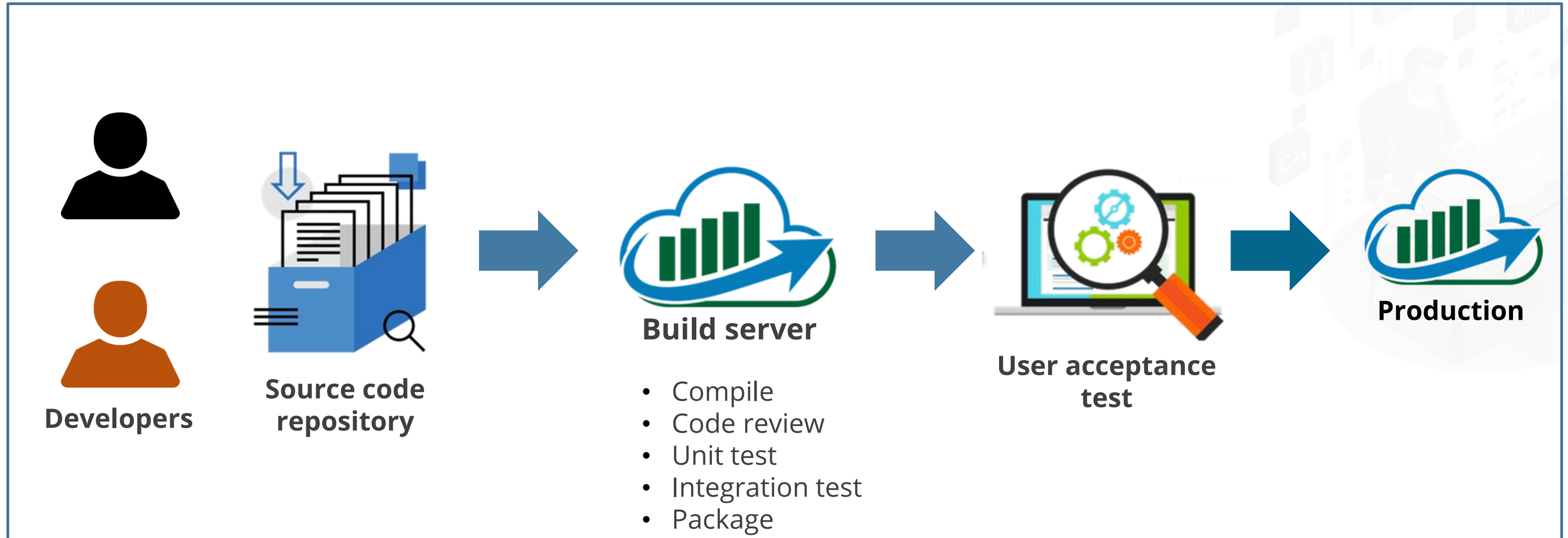


# Benefits of Continuous Delivery

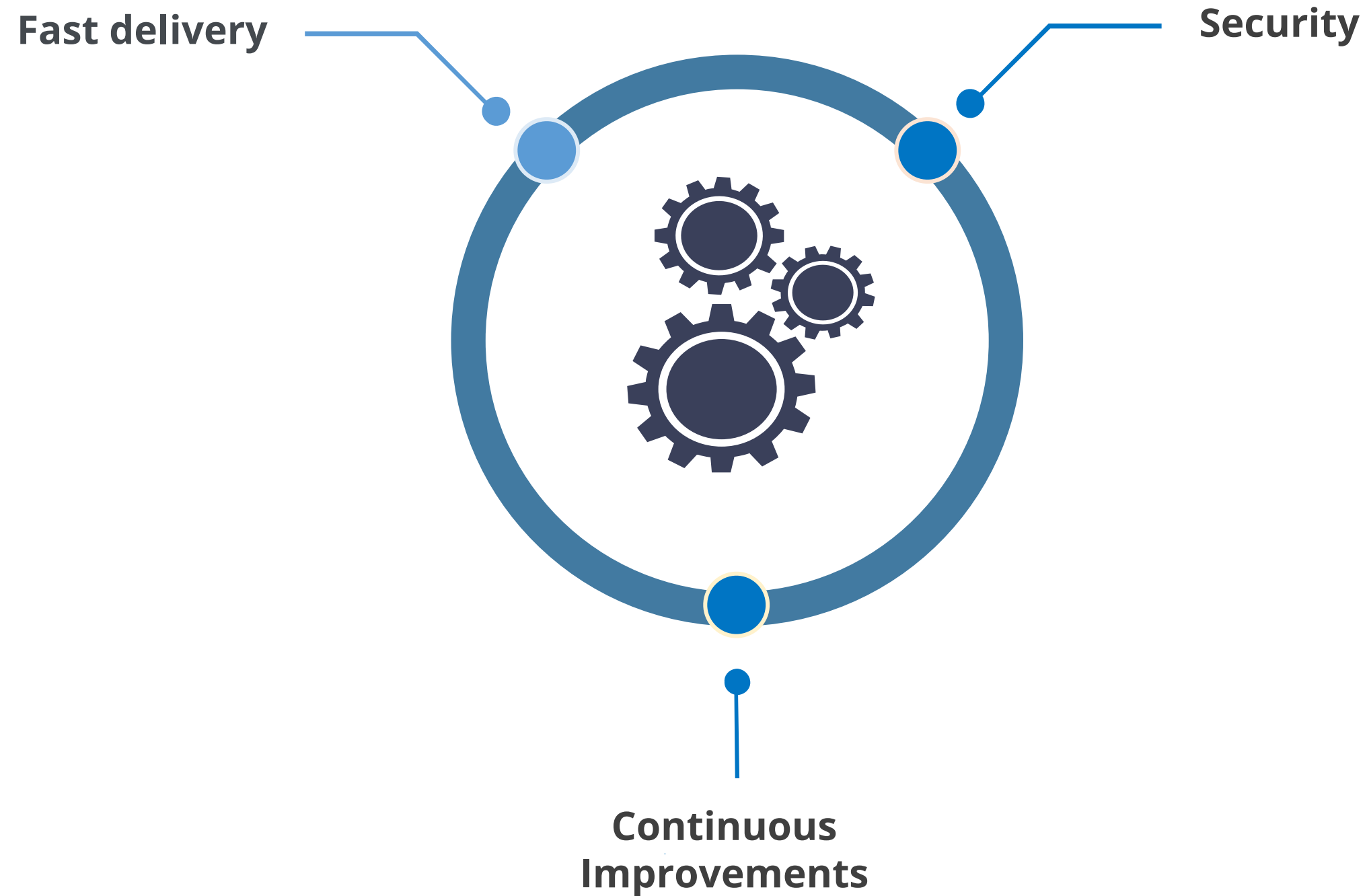


# Automated Deployment

The process wherein any code change subject to automated tests and other appropriate verifications is immediately deployed into production.



# Benefits of Automated Deployment

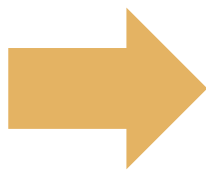




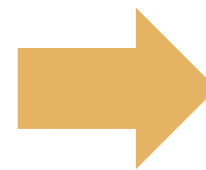
# Automated Deployment Pipeline

The automated pipeline deployment is a sequence of scripts executed following each change of code committed to the repository. If the process succeeds, the deployment ends in the production environment.

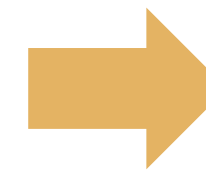
**Code  
Change**



Continuous  
Integration



Automated  
Acceptance  
Testing



Configuration  
Management

# Phases of Automated Deployment Pipeline

---

Each step below corresponds to a phase in the traditional delivery process:

**Continuous Integration:** Makes sure that the code written by different developers integrates together.

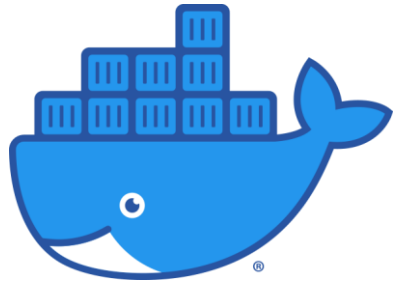
**Automated Acceptance Testing:** Replaces the manual QA phase and checks if the features implemented by developers meet the client's requirements.

**Configuration Management:** Replaces the manual operations phase, configures the environment, and deploys the software.

# FULL STACK

## Building the Continuous Delivery Process

## Tools Used



Docker ecosystem



Jenkins



Ansible



Java



GitHub



Gradle



Spring boot

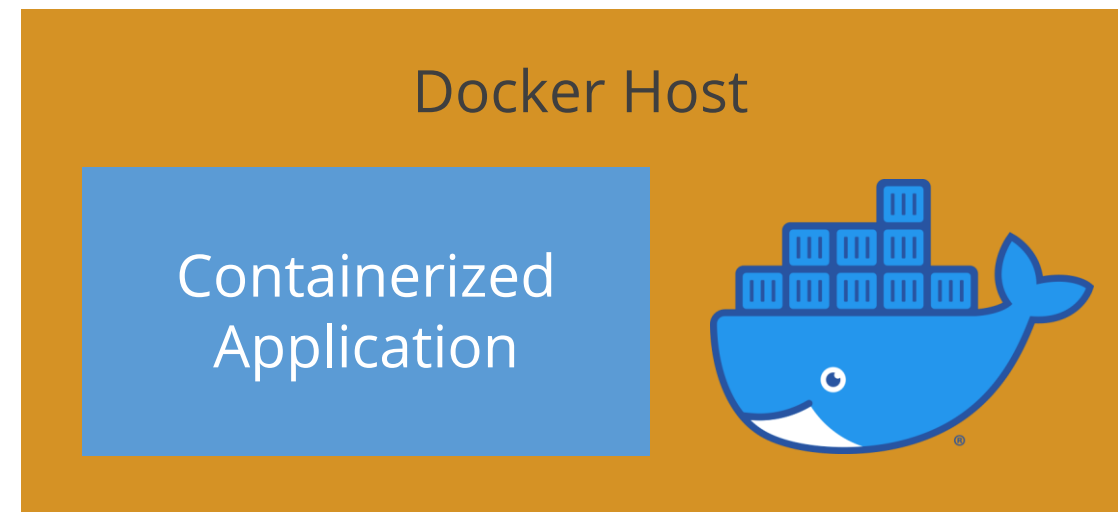


Cucumber



# Automated Deployment System

A dockerized application (web service) is made to run as a container on a Docker Host. This is reachable as it would run directly on the host machine.



Dockerized application

Jenkins configuration

Continuous Integration

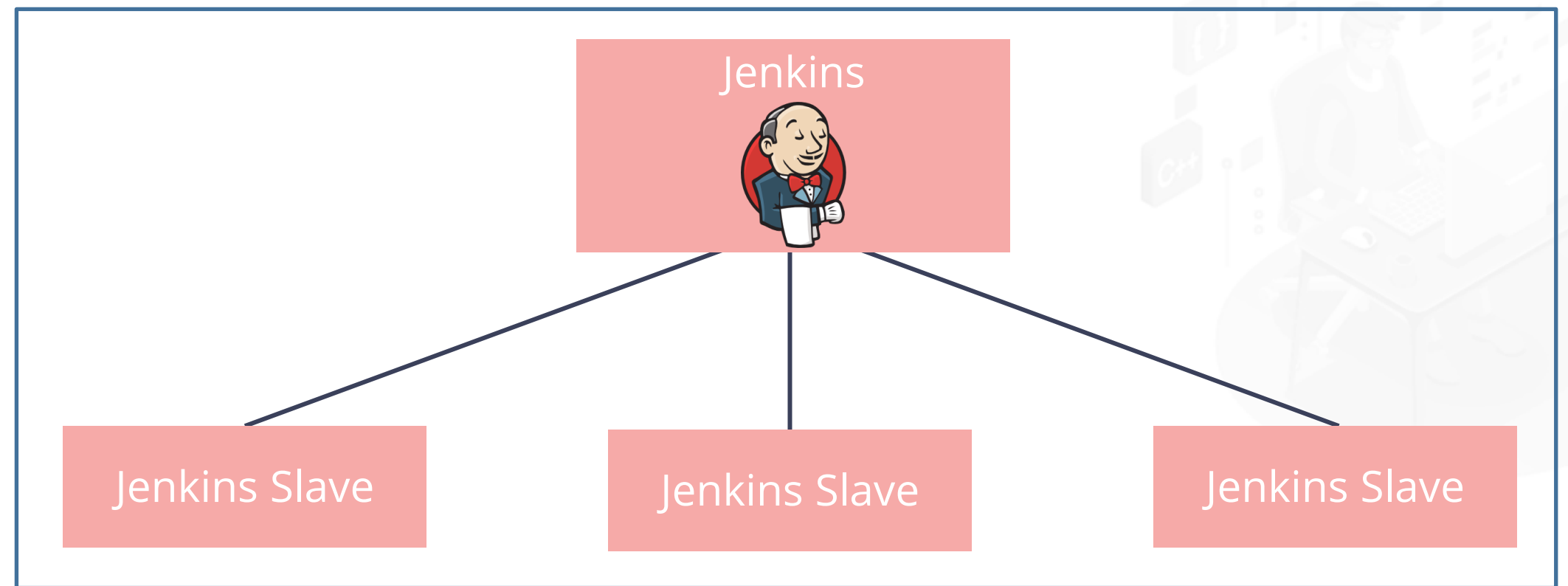
Automated acceptance  
testing

Configuration  
management

Advanced continuous  
delivery

# Automated Deployment System

The Jenkins master accepts a build request, but the execution is started at one of the Jenkins Slave (agent) machines. This provides horizontal scaling of the Jenkins environment.



Dockerized application

Jenkins configuration

Continuous Integration

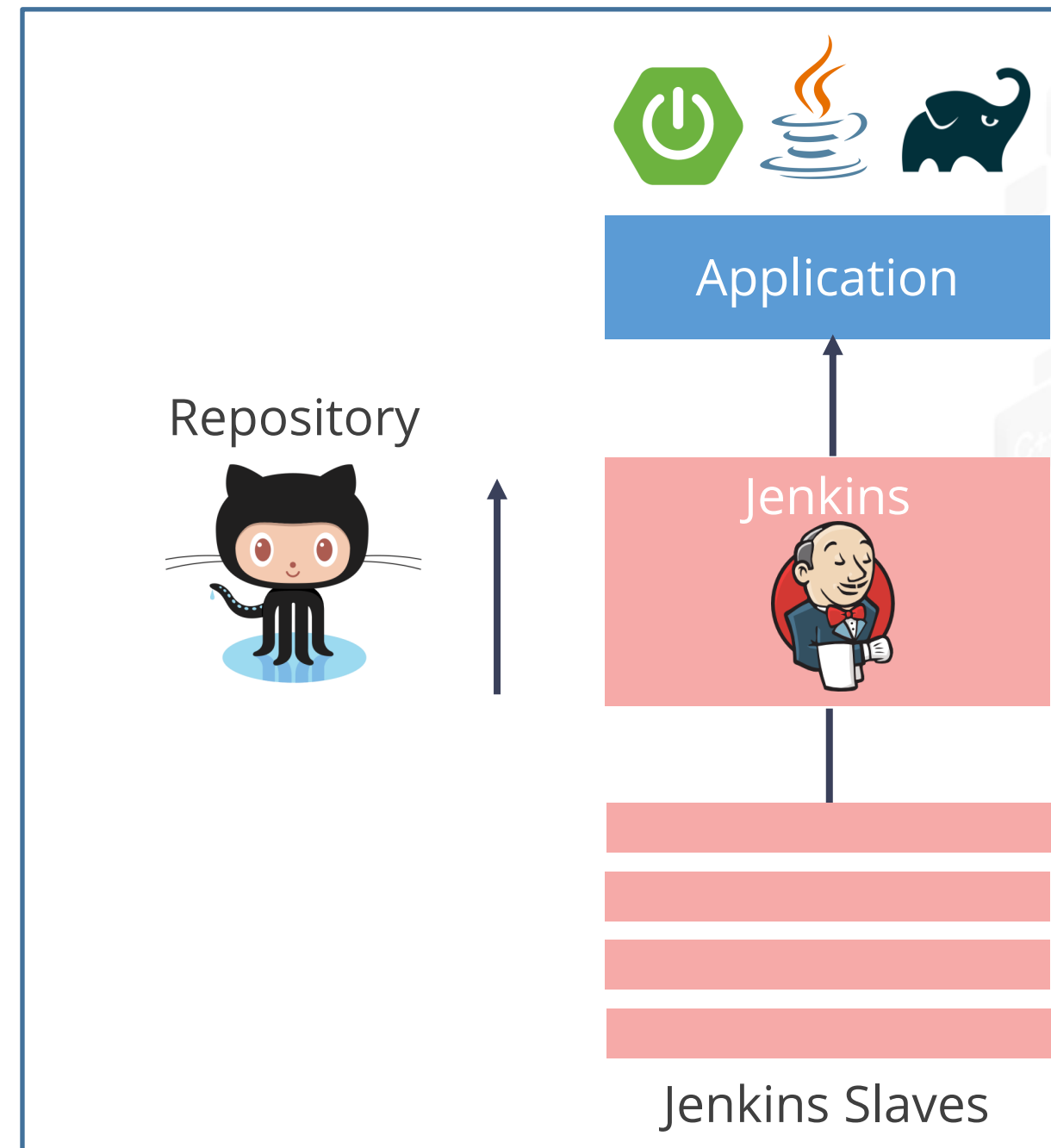
Automated acceptance testing

Configuration management

Advanced continuous delivery

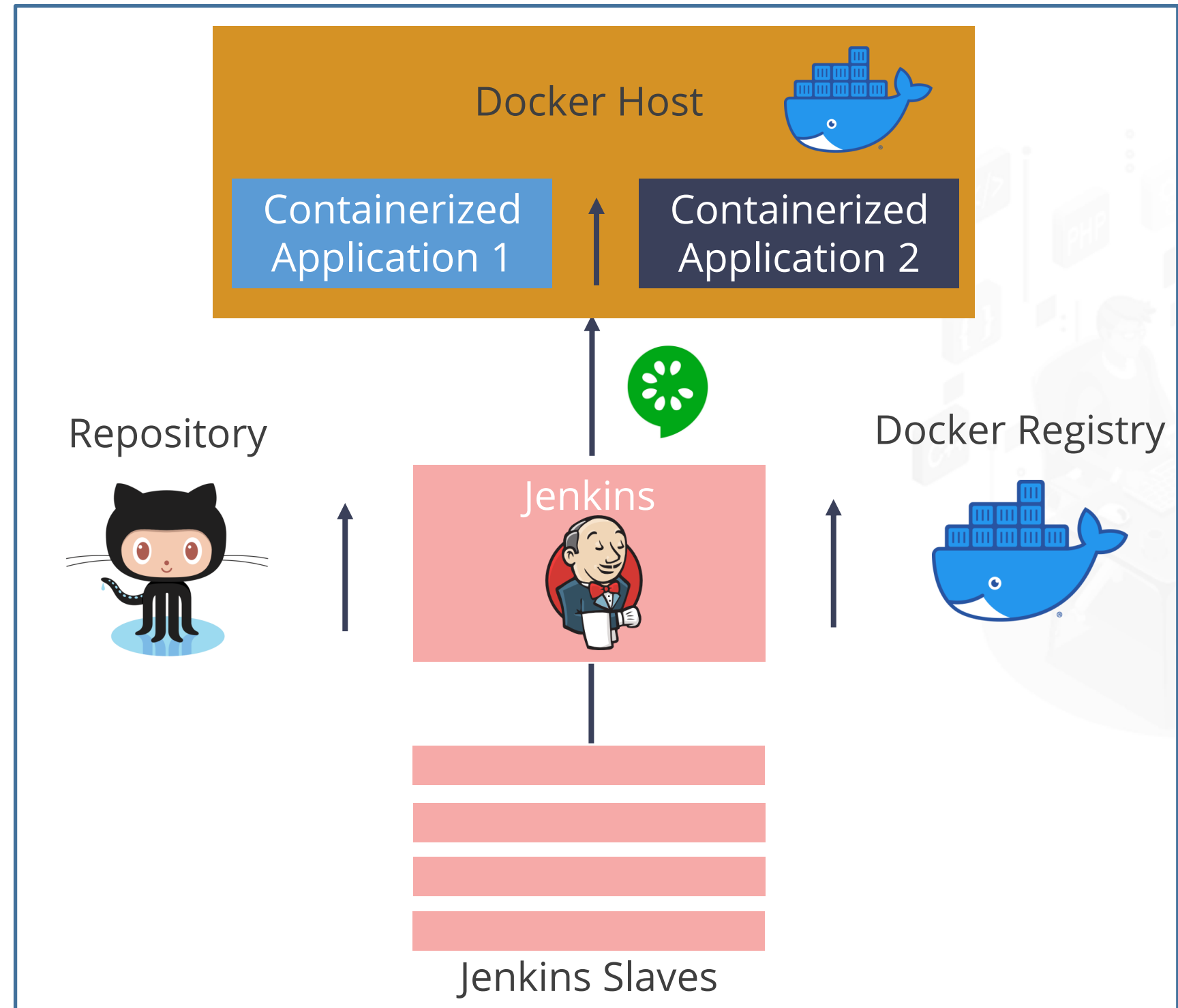
# Automated Deployment System

The application is a simple web service written in Java with the Spring Boot framework. Gradle is used as a build tool and GitHub as the source code repository.



# Automated Deployment System

After the application is started on the Docker Host, Jenkins runs a suite of acceptance tests written in the Cucumber framework.



Dockerized application

Jenkins configuration

Continuous Integration

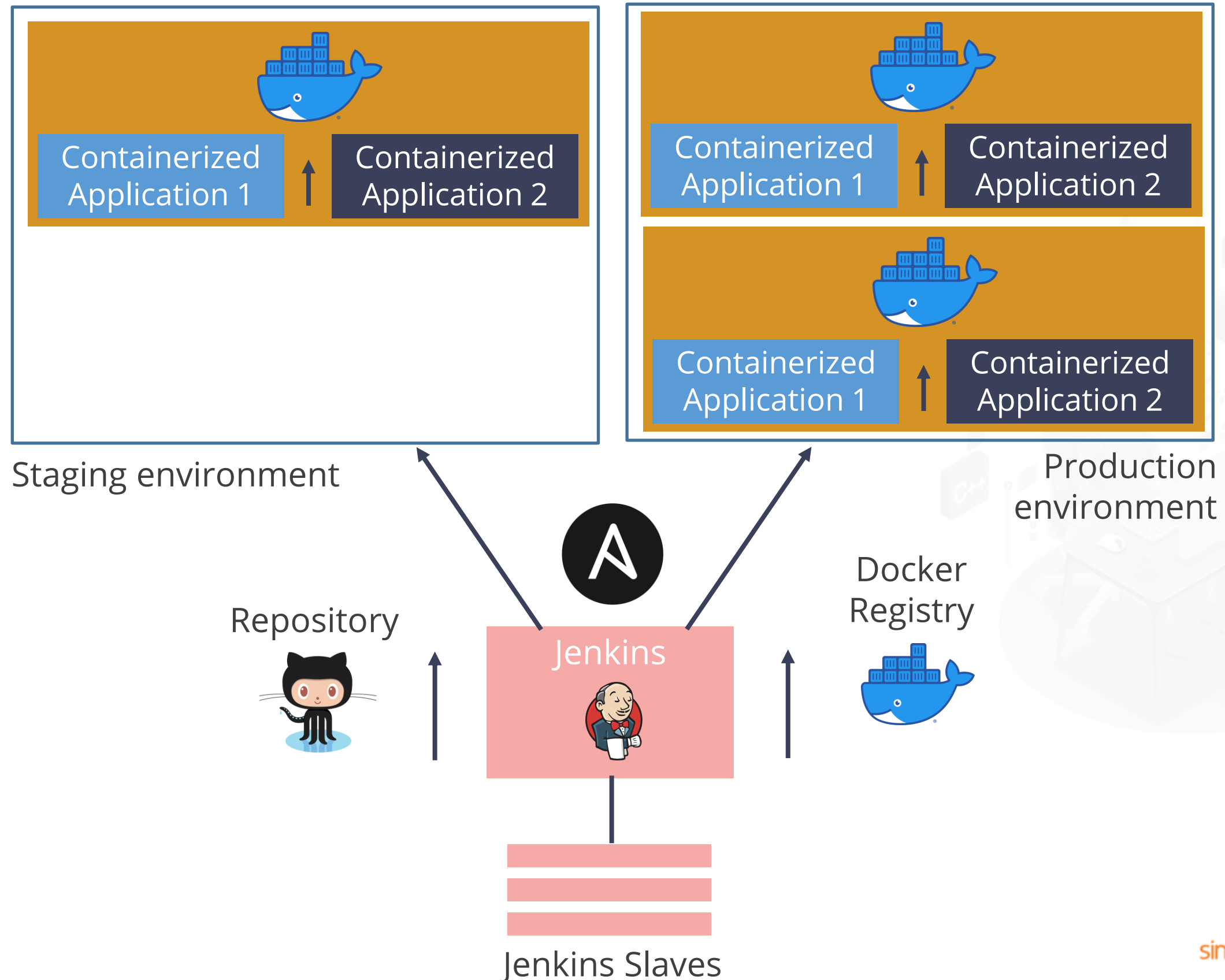
Automated acceptance testing

Configuration management

Advanced continuous delivery



# Automated Deployment System



Dockerized application

Jenkins configuration

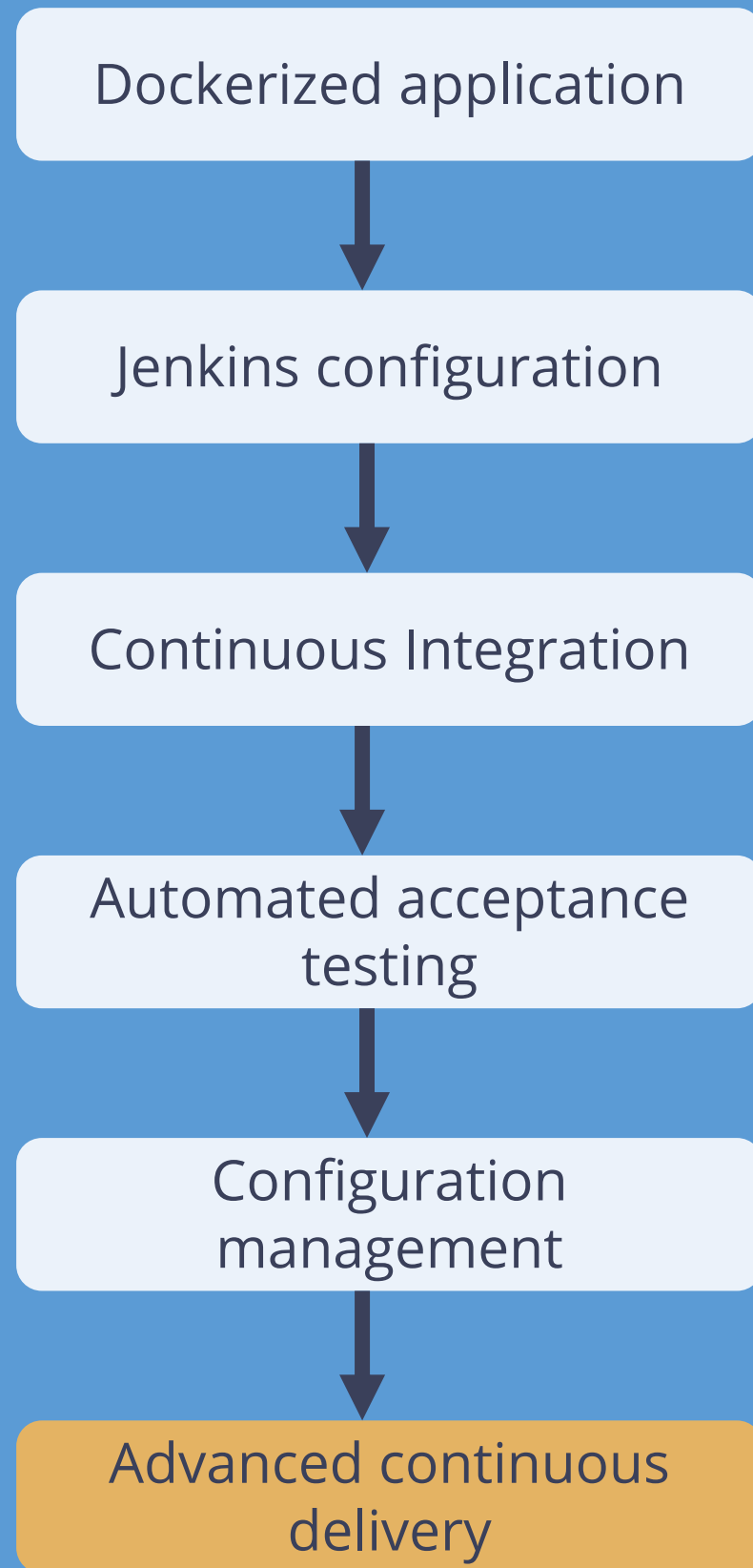
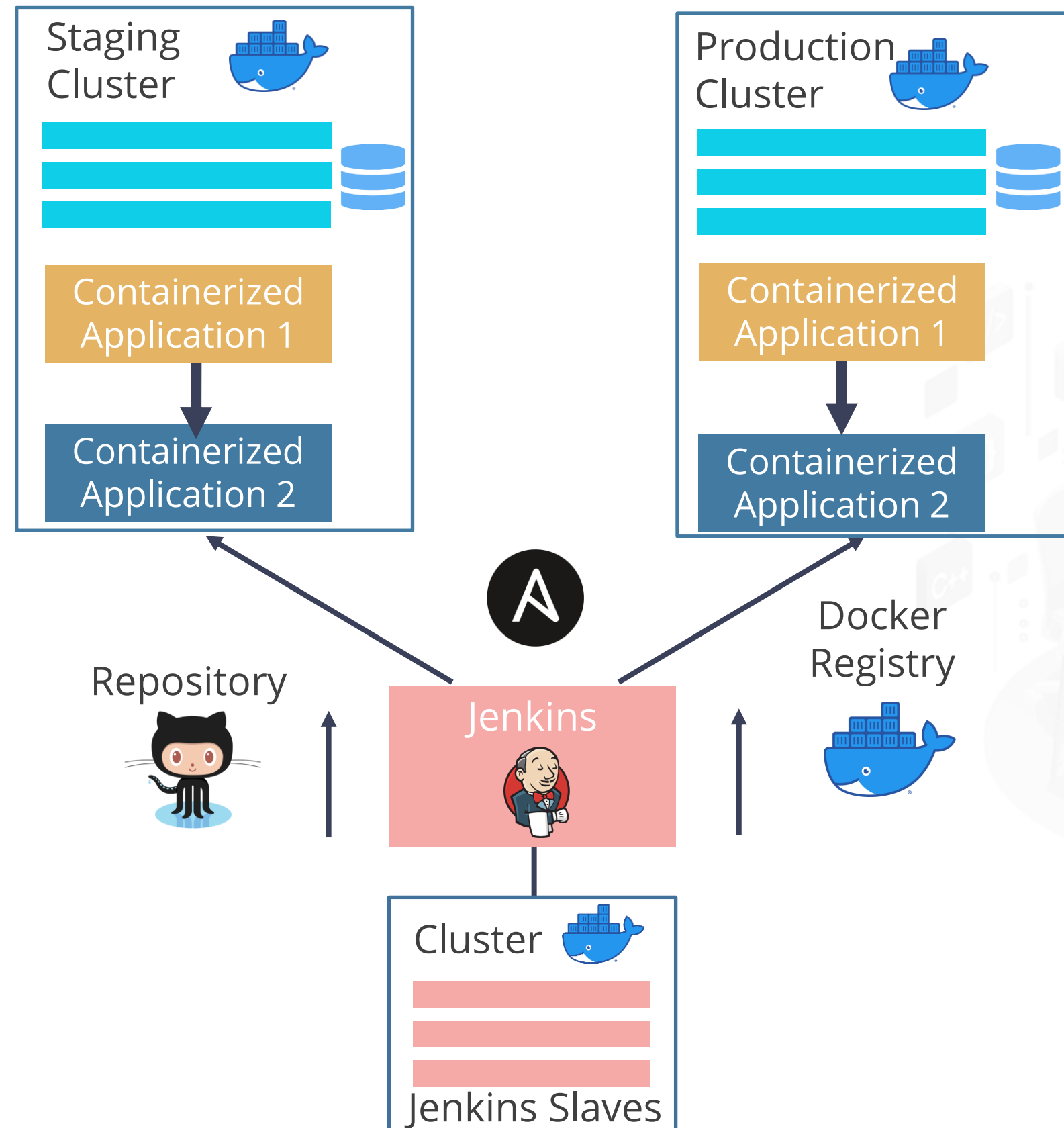
Continuous Integration

Automated acceptance testing

Configuration management

Advanced continuous delivery

# Automated Deployment System



# FULL STACK

## Implementing Automated and Continuous Deployment

# The Deployment Script

Scriptable deployment process is an important part in any automated deployment. Essentially, the deployment stage executes only after the successful completion of the **Build** and **Test** stages.

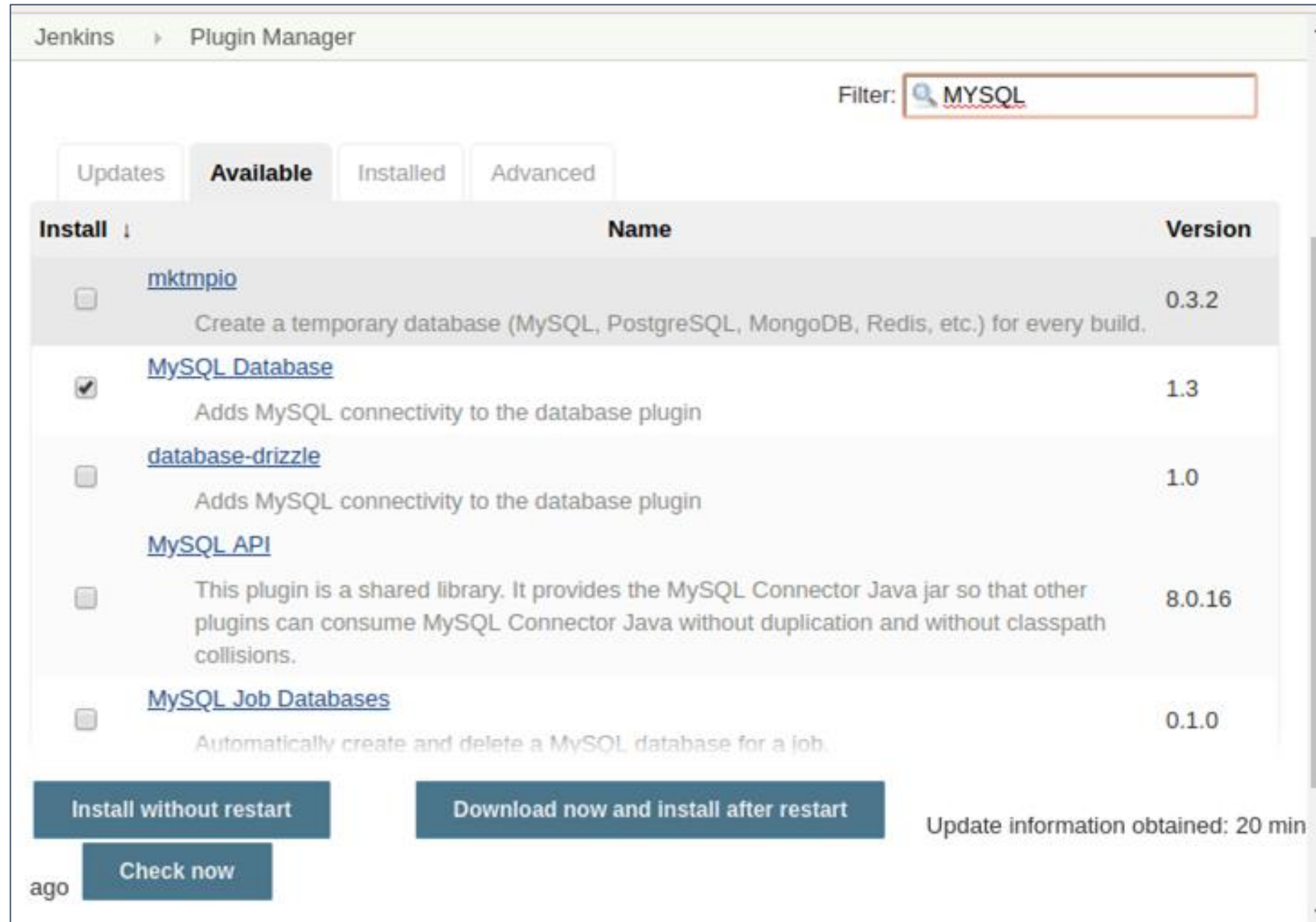
```
Jenkinsfile (Declarative Pipeline)
pipeline {
    agent any

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



# Database Updates: MySQL Plugin

Database plugin adds a system configuration entry to let the administrator configure the database used by Jenkins. MySQL Database plugin is a driver plugin for Database plugin that adds MySQL database driver.



The screenshot shows the Jenkins Plugin Manager interface. The 'Filter' box contains 'MYSQL'. The 'Available' tab is selected. A table lists several plugins, with 'MySQL Database' checked for installation. The table has columns for 'Name' and 'Version'. Below the table are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'. The text 'Update information obtained: 20 min ago' is also visible.

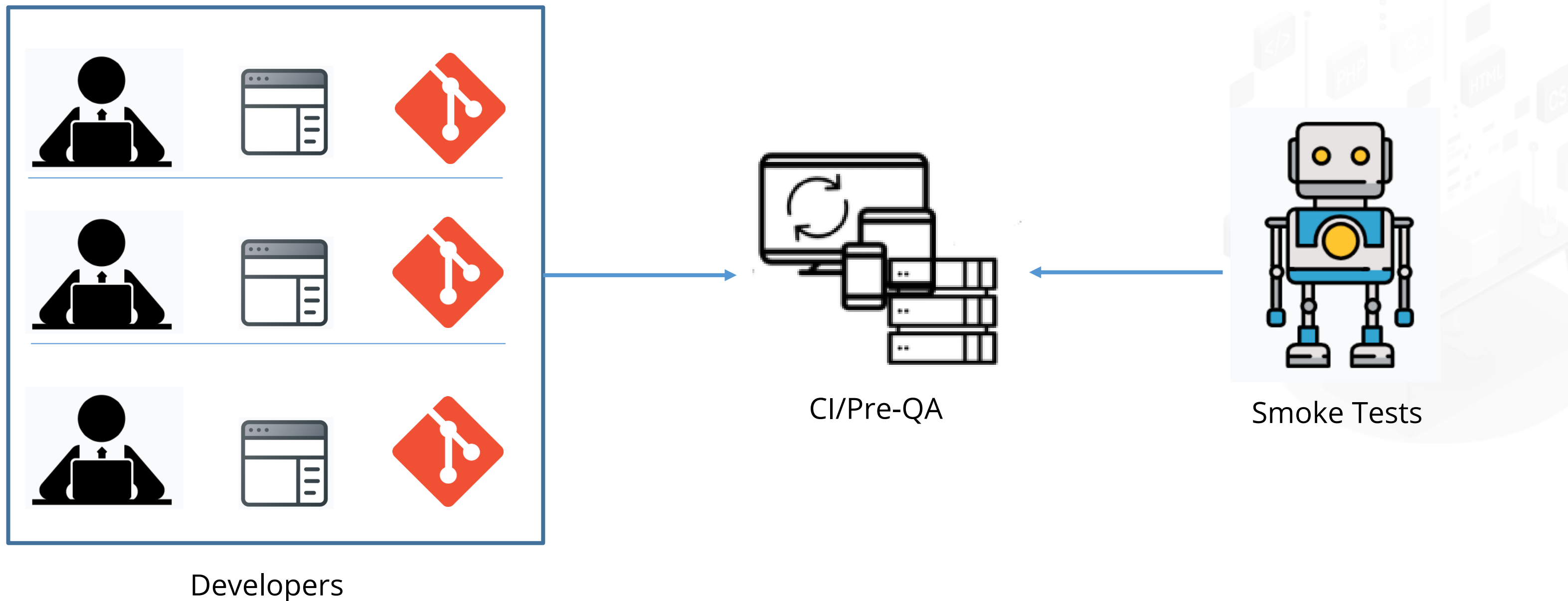
Install	Name	Version
<input type="checkbox"/>	<a href="#">mktmpio</a> Create a temporary database (MySQL, PostgreSQL, MongoDB, Redis, etc.) for every build.	0.3.2
<input checked="" type="checkbox"/>	<a href="#">MySQL Database</a> Adds MySQL connectivity to the database plugin	1.3
<input type="checkbox"/>	<a href="#">database-drizzle</a> Adds MySQL connectivity to the database plugin	1.0
<input type="checkbox"/>	<a href="#">MySQL API</a> This plugin is a shared library. It provides the MySQL Connector Java jar so that other plugins can consume MySQL Connector Java without duplication and without classpath collisions.	8.0.16
<input type="checkbox"/>	<a href="#">MySQL Job Databases</a> Automatically create and delete a MySQL database for a job.	0.1.0

Install without restart    Download now and install after restart    Update information obtained: 20 min ago    Check now



# Smoke Tests

A series of automated smoke checks must be executed for any significant automated implementation.  
An automated acceptance test subset can be a good smoke test candidate.

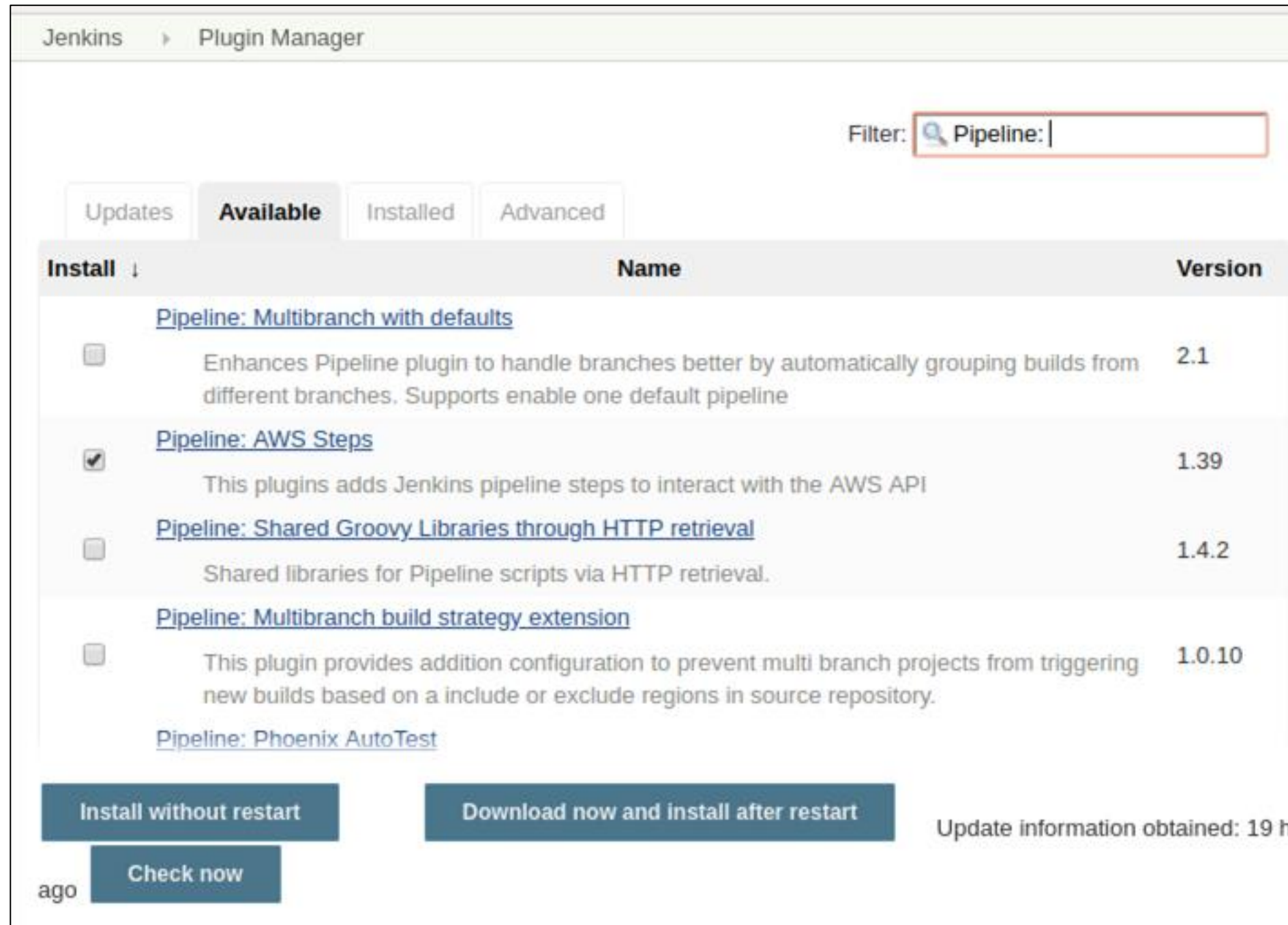


# Hot Deploy

Hot deployment is the process of adding new components (such as WAR files, EJB Jar files, enterprise Java beans, servlets, and JSP files) to a running server without having to stop the application server process and start it again.

# AWS Pipeline

AWS pipeline plugin adds Jenkins pipeline steps to interact with the AWS API.



The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected, and a search filter 'Pipeline:' is applied. The following table lists the available plugins:

Install	Name	Version
<input type="checkbox"/>	<a href="#">Pipeline: Multibranch with defaults</a> Enhances Pipeline plugin to handle branches better by automatically grouping builds from different branches. Supports enable one default pipeline	2.1
<input checked="" type="checkbox"/>	<a href="#">Pipeline: AWS Steps</a> This plugins adds Jenkins pipeline steps to interact with the AWS API	1.39
<input type="checkbox"/>	<a href="#">Pipeline: Shared Groovy Libraries through HTTP retrieval</a> Shared libraries for Pipeline scripts via HTTP retrieval.	1.4.2
<input type="checkbox"/>	<a href="#">Pipeline: Multibranch build strategy extension</a> This plugin provides addition configuration to prevent multi branch projects from triggering new builds based on a include or exclude regions in source repository.	1.0.10
	<a href="#">Pipeline: Phoenix AutoTest</a>	

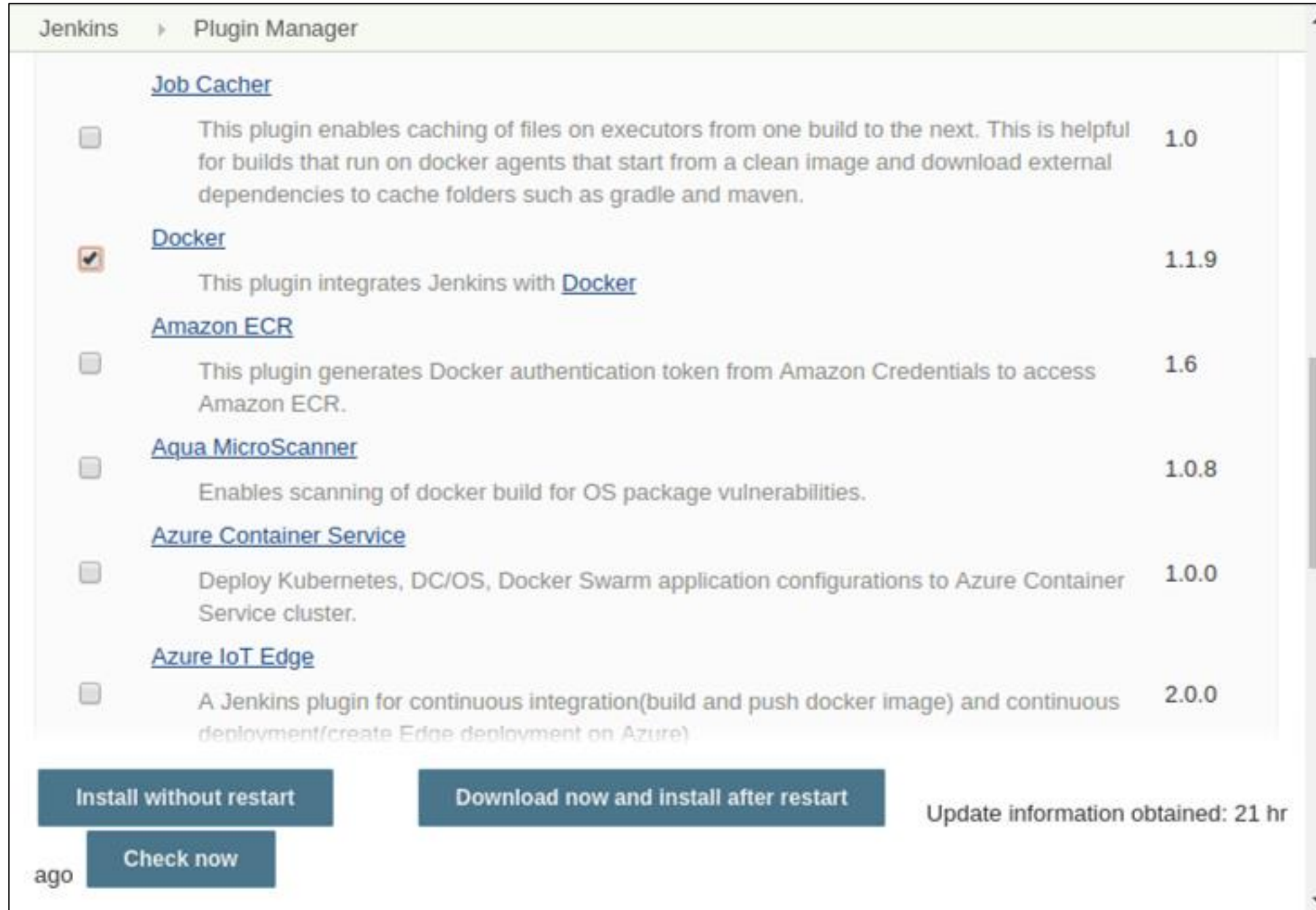
At the bottom, there are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'. The update information is noted as 'Update information obtained: 19 hr ago'.





# Jenkins and Docker

Docker plugin allows to use a docker host dynamically to provision build agents, run single build, tear-down agent. This plugin integrates Jenkins with Docker.



The screenshot shows the Jenkins Plugin Manager interface. The title bar indicates 'Jenkins' and 'Plugin Manager'. The main content area lists several plugins with their descriptions and versions. The 'Docker' plugin is highlighted with a checkmark in the checkbox column, indicating it is installed. Below the list, there are two buttons: 'Install without restart' and 'Download now and install after restart'. To the right of these buttons, it says 'Update information obtained: 21 hr ago'. At the bottom left, there is a 'Check now' button.

Plugin Name	Description	Version
<a href="#">Job Cacher</a>	This plugin enables caching of files on executors from one build to the next. This is helpful for builds that run on docker agents that start from a clean image and download external dependencies to cache folders such as gradle and maven.	1.0
<input checked="" type="checkbox"/> <a href="#">Docker</a>	This plugin integrates Jenkins with <a href="#">Docker</a>	1.1.9
<a href="#">Amazon ECR</a>	This plugin generates Docker authentication token from Amazon Credentials to access Amazon ECR.	1.6
<input type="checkbox"/> <a href="#">Aqua MicroScanner</a>	Enables scanning of docker build for OS package vulnerabilities.	1.0.8
<input type="checkbox"/> <a href="#">Azure Container Service</a>	Deploy Kubernetes, DC/OS, Docker Swarm application configurations to Azure Container Service cluster.	1.0.0
<input type="checkbox"/> <a href="#">Azure IoT Edge</a>	A Jenkins plugin for continuous integration(build and push docker image) and continuous deployment/create Edge deployment on Azure)	2.0.0

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 21 hr ago [Check now](#)



# Deploying a Python Application



**Problem statement:** You have been asked to connect and deploy a python application to an application server.

## Steps to perform:

1. Push code to GitHub repositories
2. Login to Jenkins
3. Create Jenkins job for Python
4. Run the job and deploy the app

ASSISTED PRACTICE

# Tomcat and Jenkins



**Problem statement:** You have been asked to connect and deploy a Java application on Tomcat using Jenkins.

## Steps to perform:

1. Push code to GitHub repositories
2. Configure Tomcat
3. Login to Jenkins
4. Create Jenkins job for deployment
5. Run the job and deploy the app

ASSISTED PRACTICE

# PHP and Jenkins



**Problem statement:** You have been asked to connect and deploy a PHP application on Ant using Jenkins.

## Steps to perform:

1. Push code to GitHub repositories
2. Login to Jenkins
3. Create Jenkins job for PHP
4. Run the job and deploy the app

ASSISTED PRACTICE

## Key Takeaways

- Continuous Deployment is a development practice of releasing software on production servers continuously in an automated manner.
- Continuous delivery is a process, where code changes are automatically built, tested, and prepared for production.
- Smoke testing is a software testing method that determines whether the employed build is stable or not.





# Deploying Maven App to Tomcat Server



## Problem Statement:

You're a DevOps engineer at Hooli, a computer vision company that started out as an image search platform. The company is celebrating its 25th anniversary and the CEO wants to celebrate the occasion by replacing the Hooli website with the original website that the company started out with for a day. You're required to build a Maven web app that says "Welcome to Hooli", set up a Tomcat server on an EC2 instance, set up a Jenkins pipeline to compile the app, and deploy it to Tomcat. The Jenkins job has to compile the app, generate a war file, and deploy the war file to the Tomcat server.

## Requirements:

- The app should be built with Maven.
- The Tomcat server should allow remote deployment.
- The pipeline should be built with the Maven Project plugin.
- The build job should compile the app and do the deployment as a post-build action.