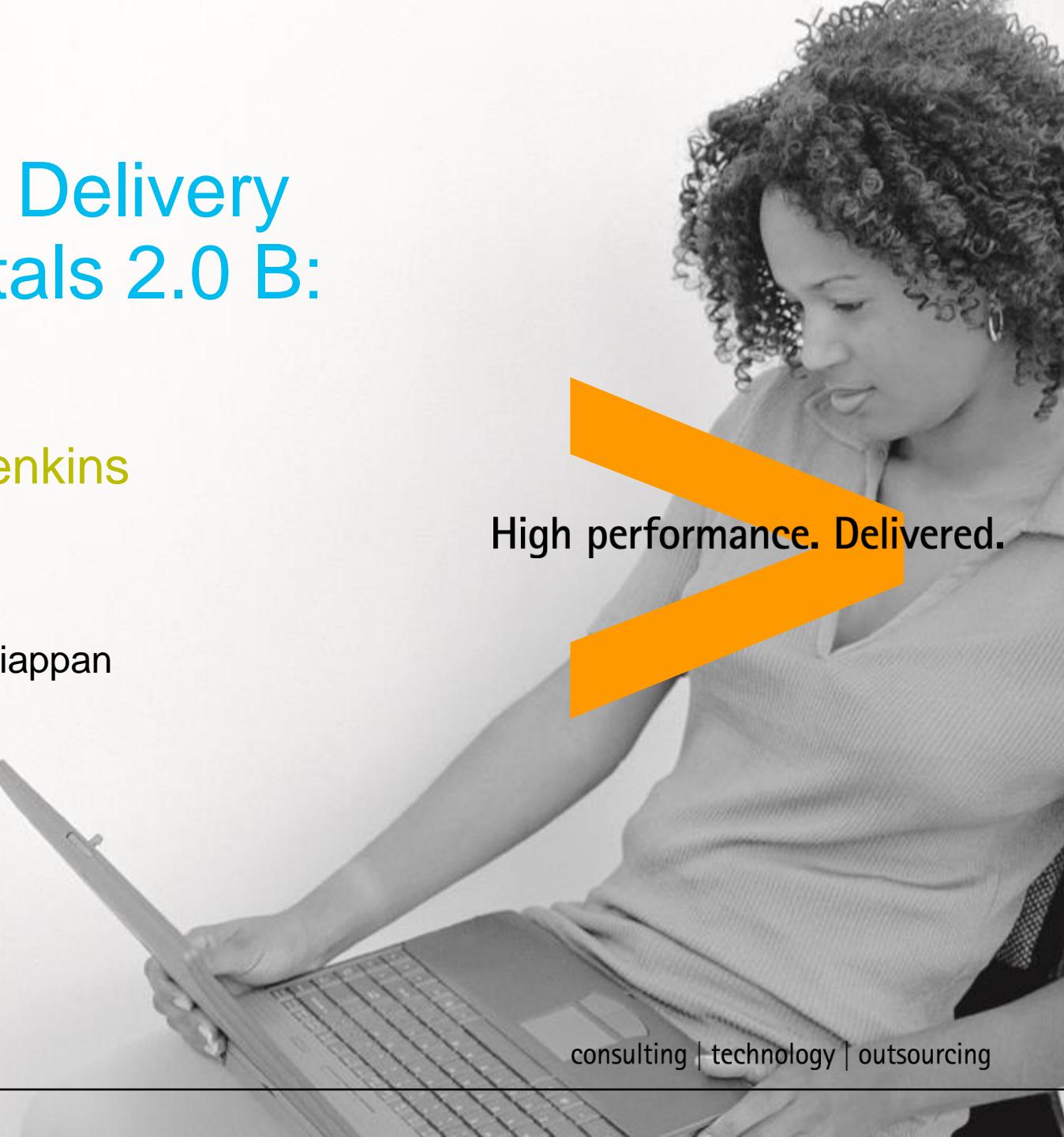


Application Delivery Fundamentals 2.0 B: Java

Introduction to Jenkins

Parameswari Ettiappan

A black and white photograph of a woman with curly hair, wearing a light-colored shirt, sitting at a desk and working on a laptop. She is looking down at the screen. The background is plain.

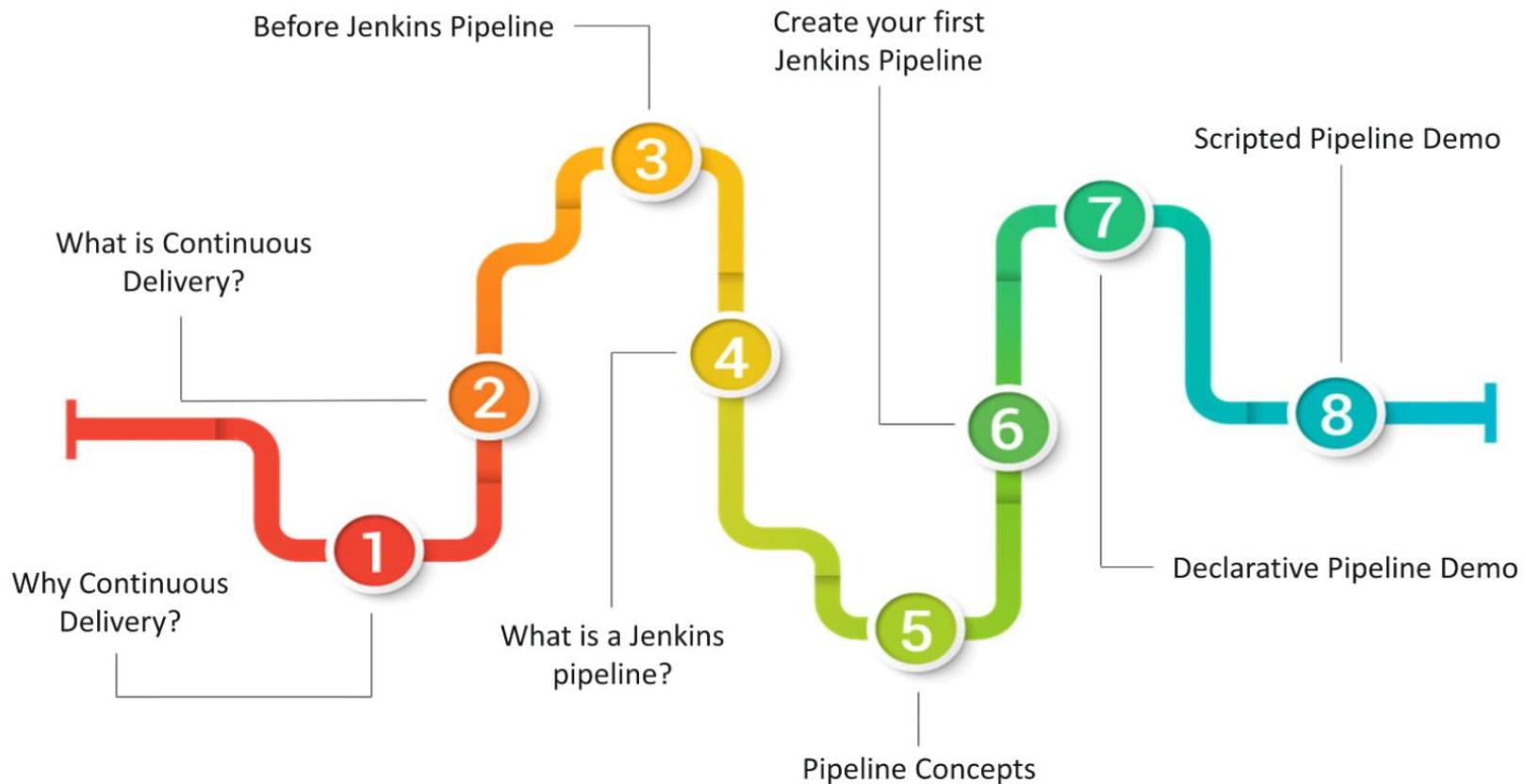
High performance. Delivered.

Goals

- Devops
- CI/CD Pipeline
- Maven Build
- Docker Build
- Docker Run
- Jenkins Pipeline
- Sonar Qube

Goals

What Will I Learn Today?



Software Requirements

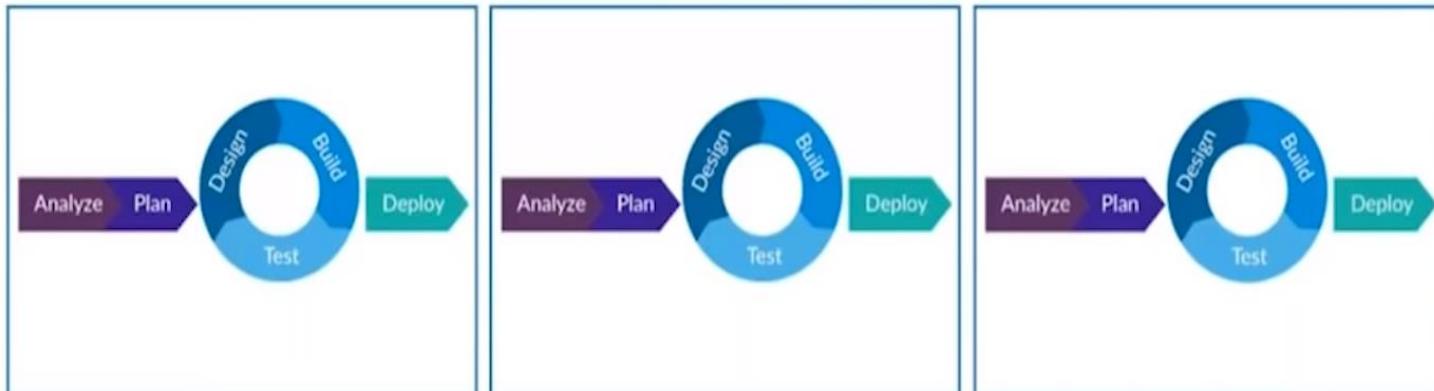
- Minimum java 8
- STS 4
- maven 3.6
- jenkins 2 or more
- sonar qube 6.6
- github/gitlab
- Docker desktop wsl2

Water Fall Model vs Agile

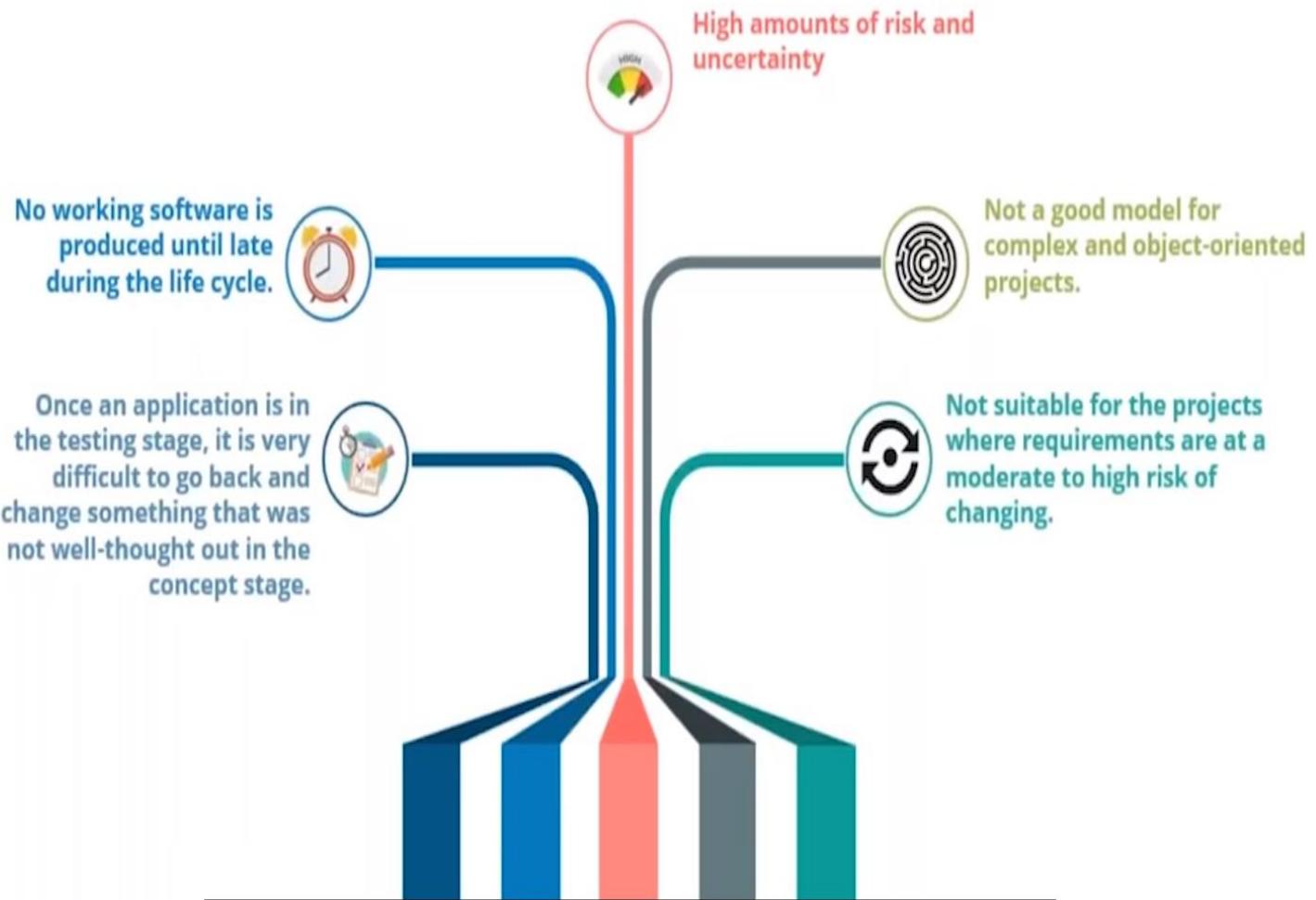
Waterfall



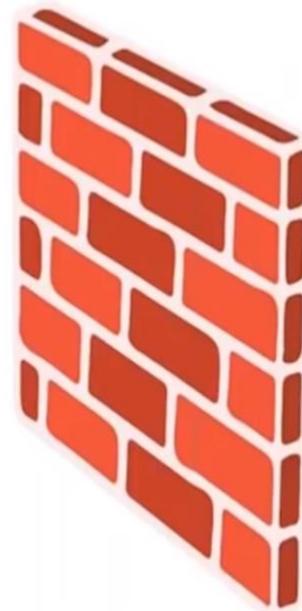
Agile



Limitations of Waterfall Model



Limitations of Agile



Wants Change

Wants Stability

HP case study



HP case study

How did they achieve this?

They implemented a CD pipeline
with two key features

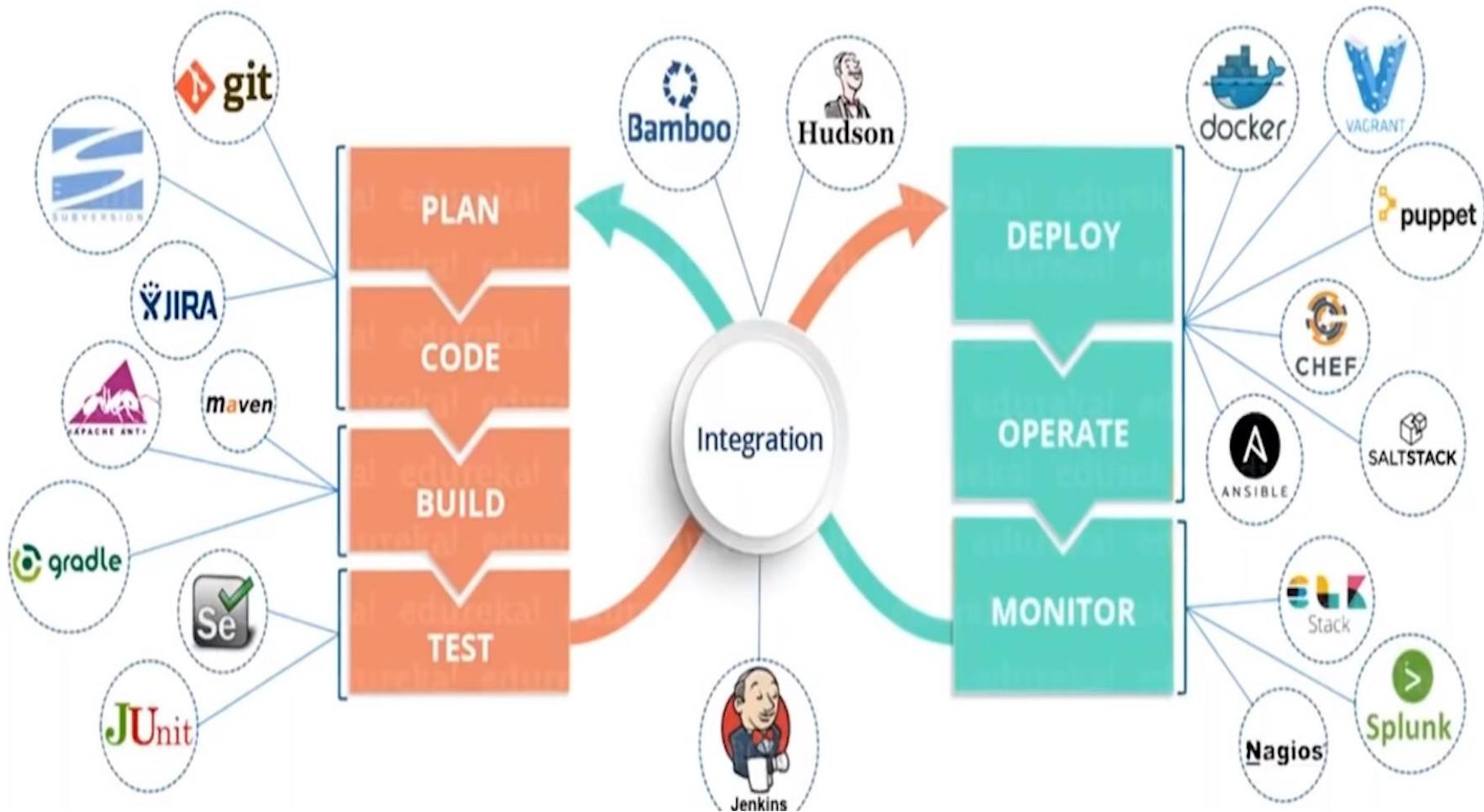


- Practice CI
- Automation at every step

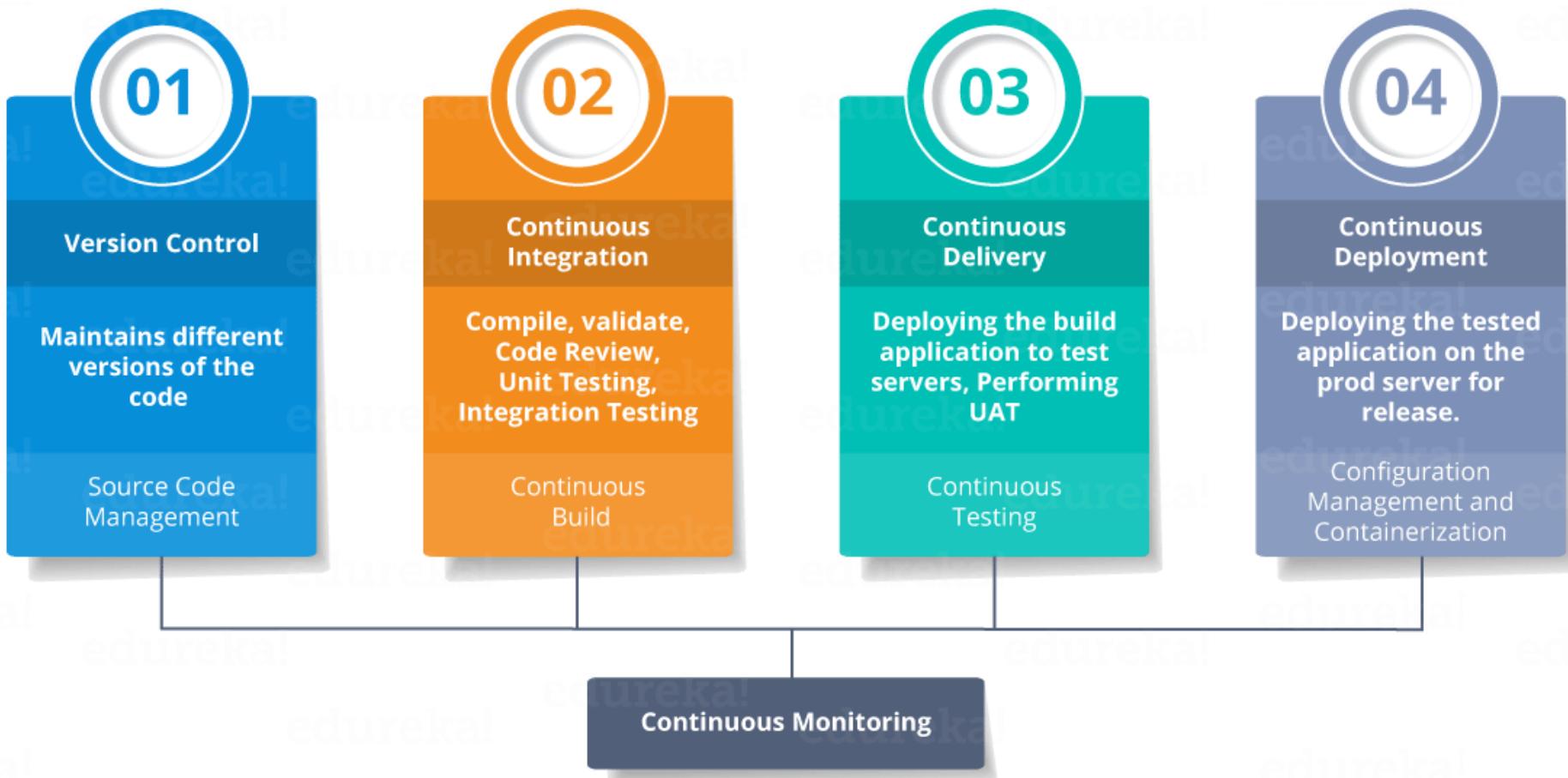


- Overall development cost reduced ~ 40%
- Programs under development increased ~ 140%
- Development cost per program went down ~ 78%

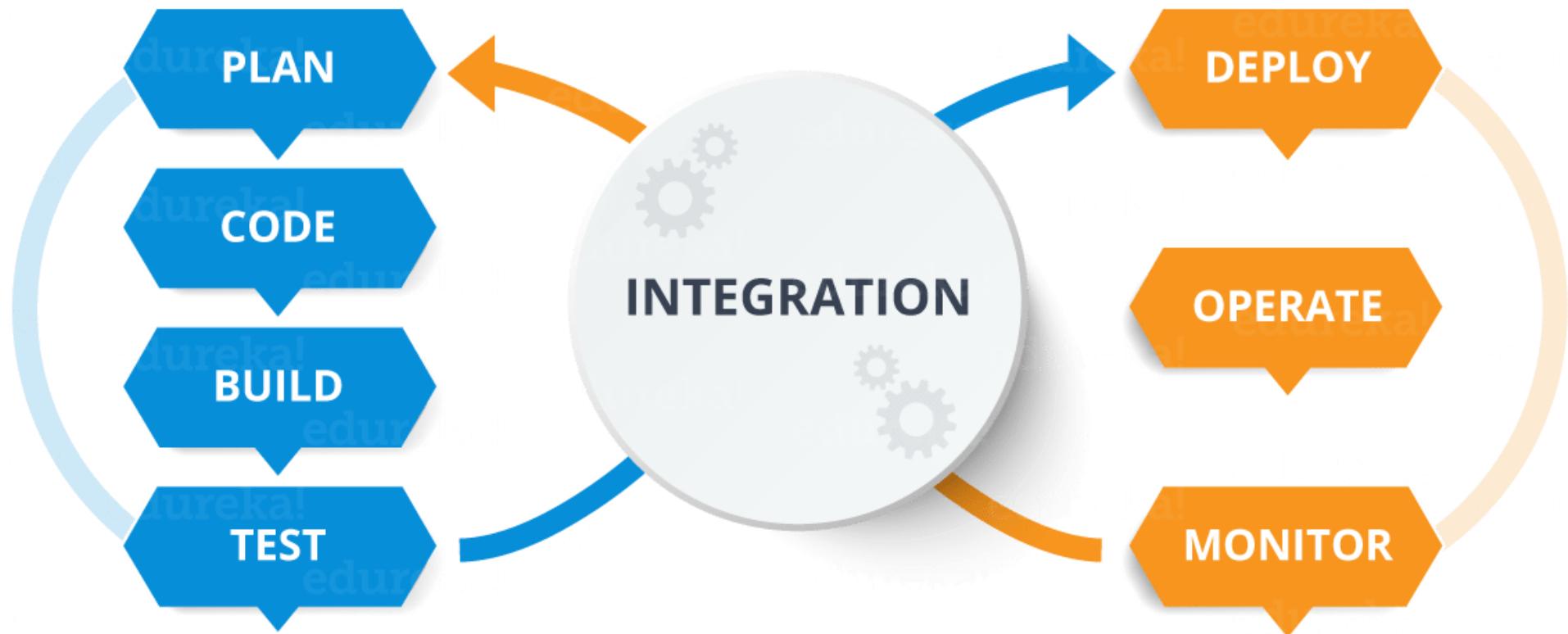
What is Devops



CI/CD Pipeline



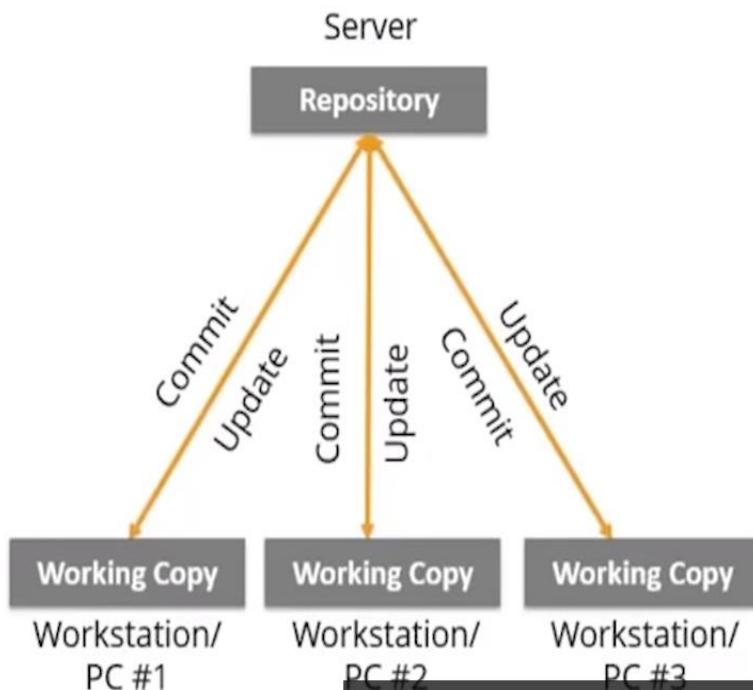
CI/CD Pipeline



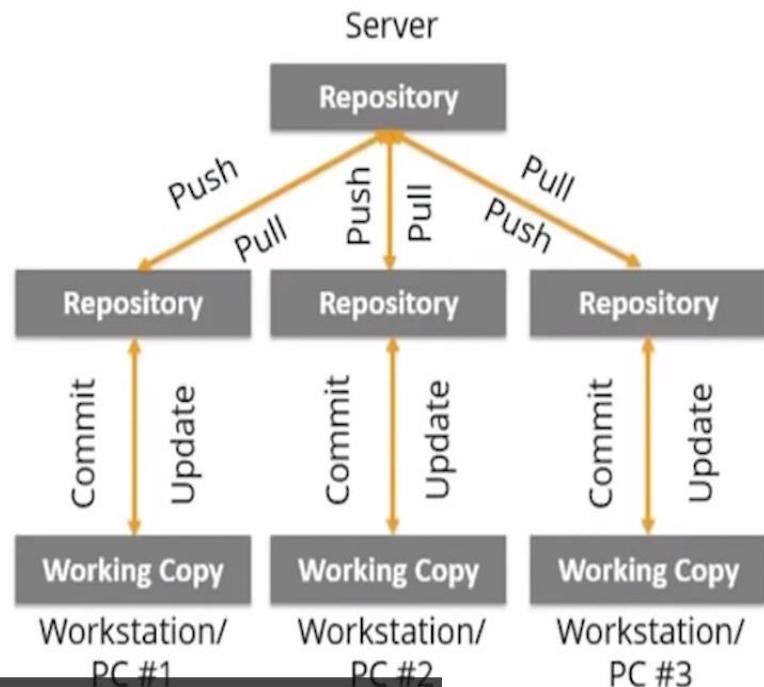
Source Code Management

The management of changes to documents, computer programs, large websites and other collection of information

Centralized Version Control System



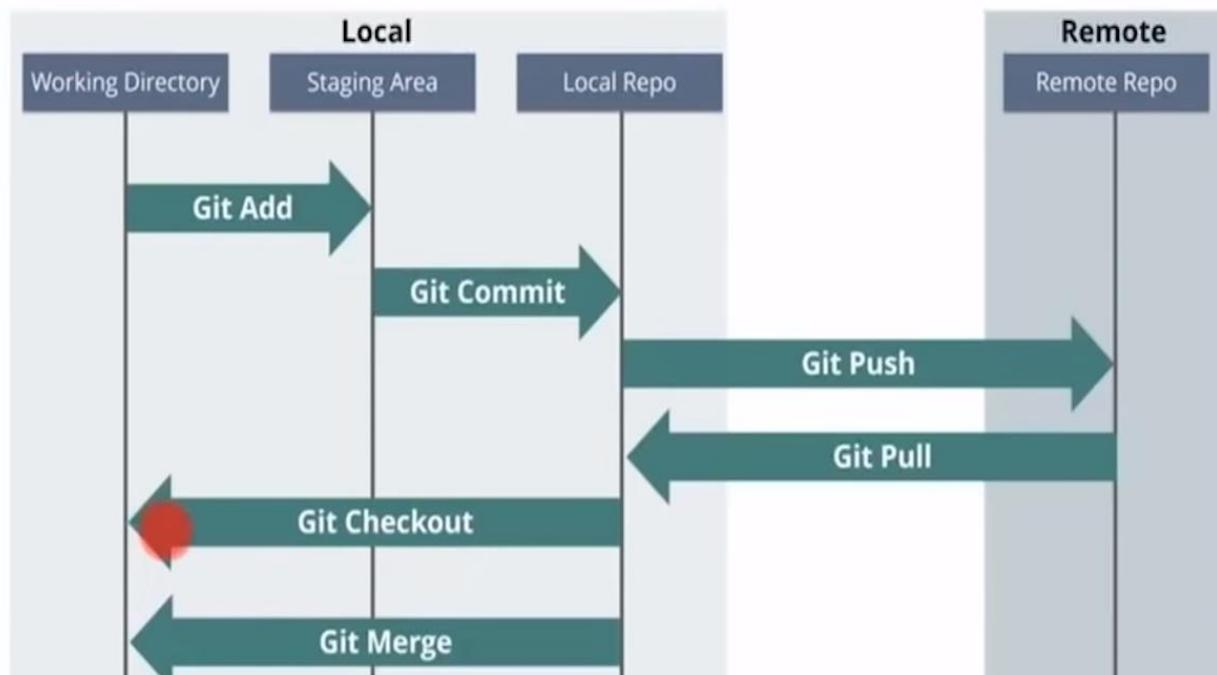
Distributed Version Control System



Source Code Management



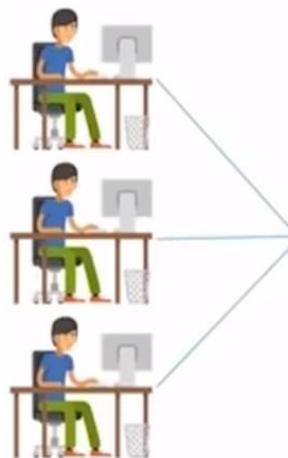
Git is a Distributed Version Control tool that supports distributed non-linear workflows by providing data assurance for developing quality software



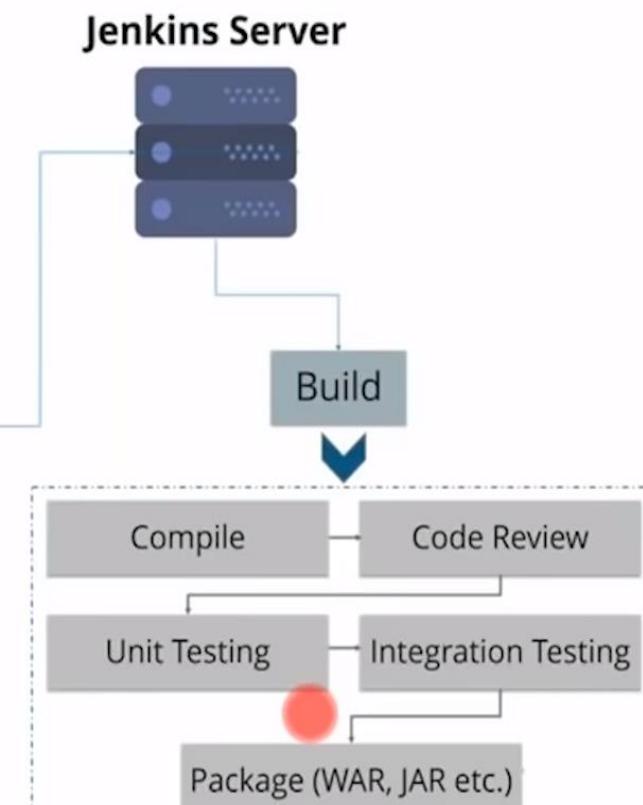
Continuous Integration



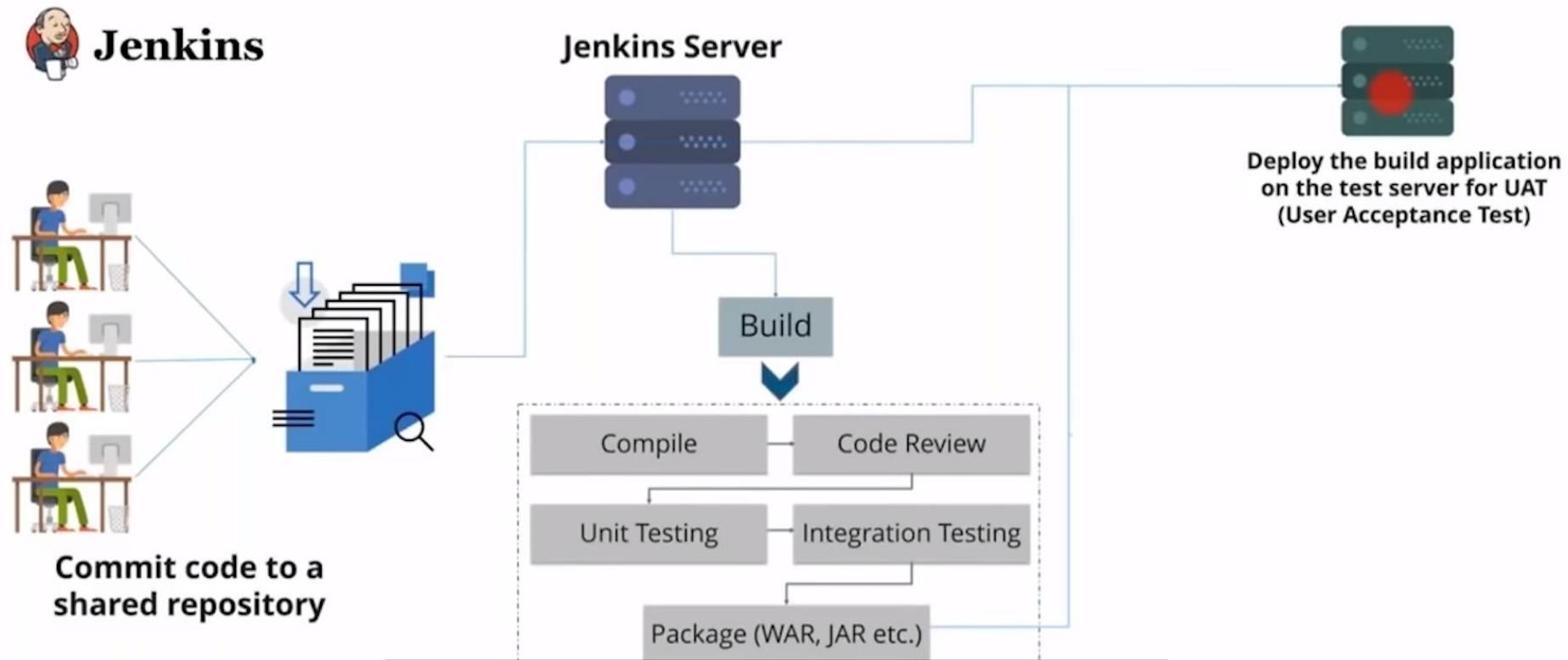
Jenkins



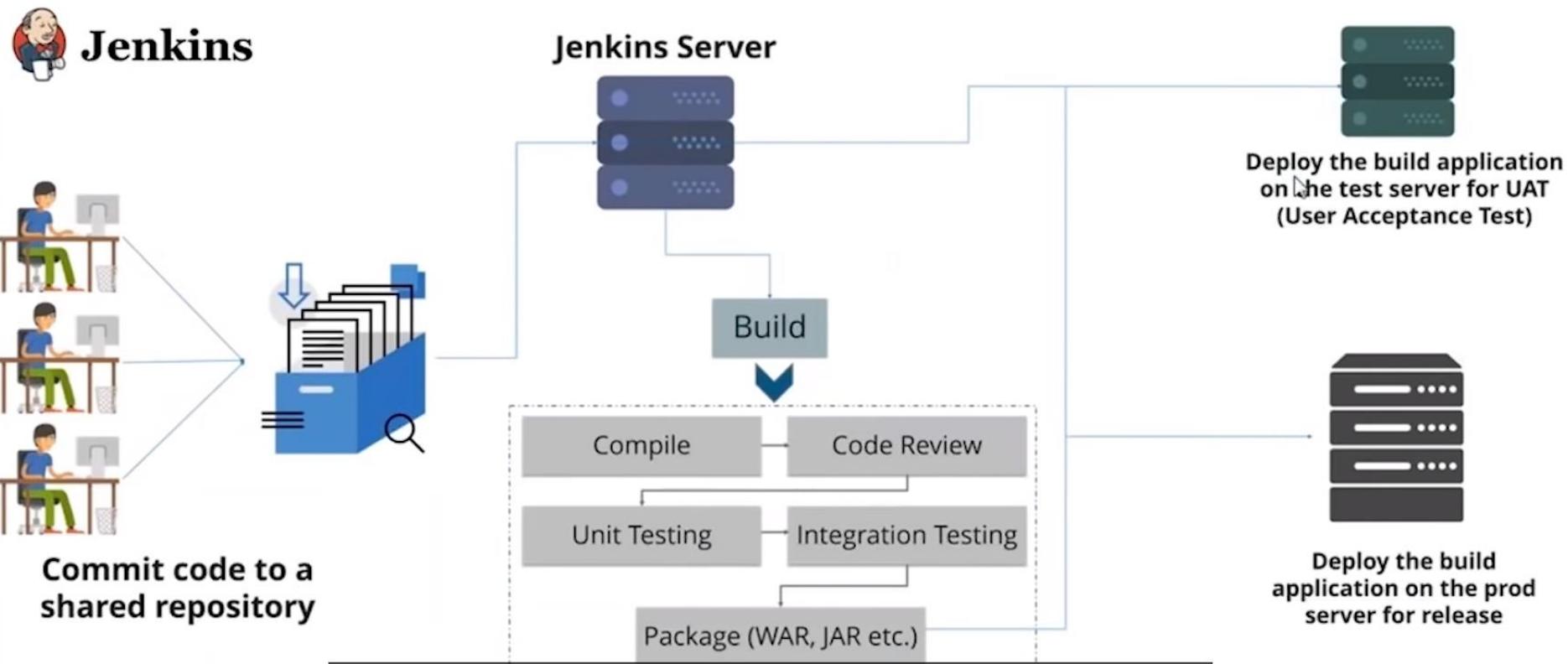
Commit code to a shared repository



Continuous Delivery



Continuous Deployment

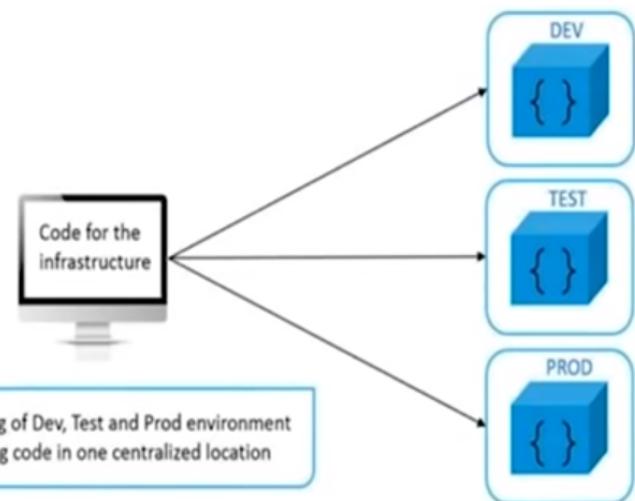


Configuration Management

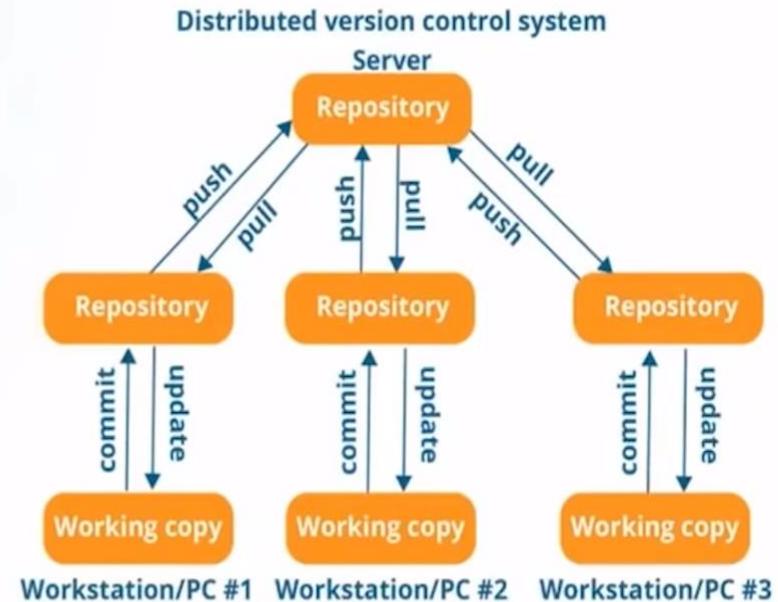
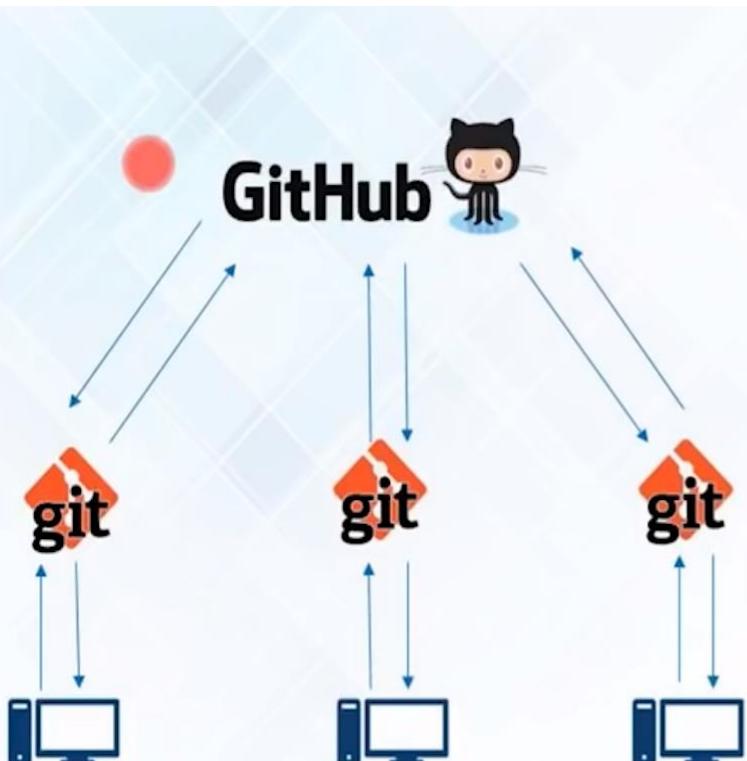
- Configuration Management is the practice of handling changes systematically so that a system maintains its integrity over time

- Configuration Management (CM) ensures that the current design and build state of the system is known, good & trusted

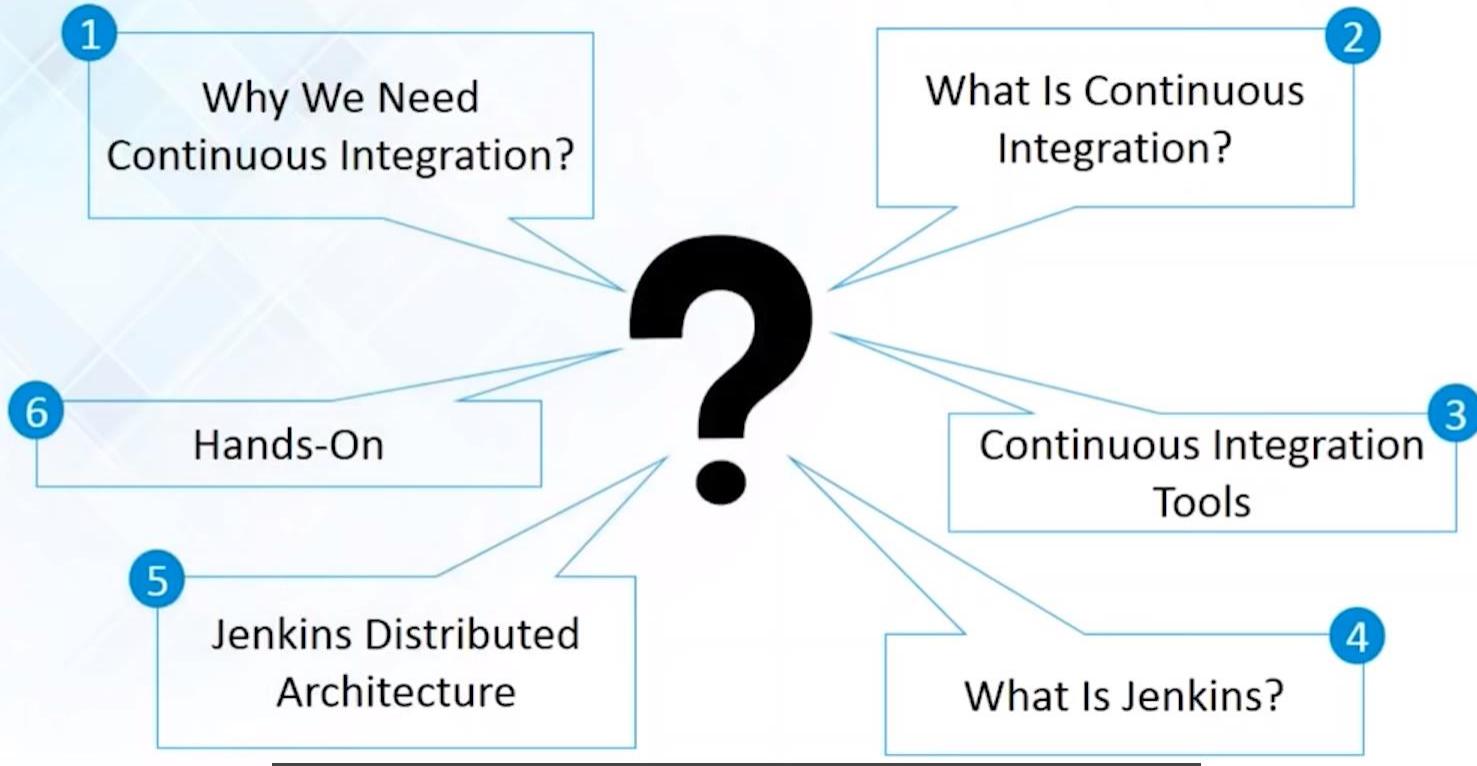
- It doesn't rely on the tacit knowledge of the development team



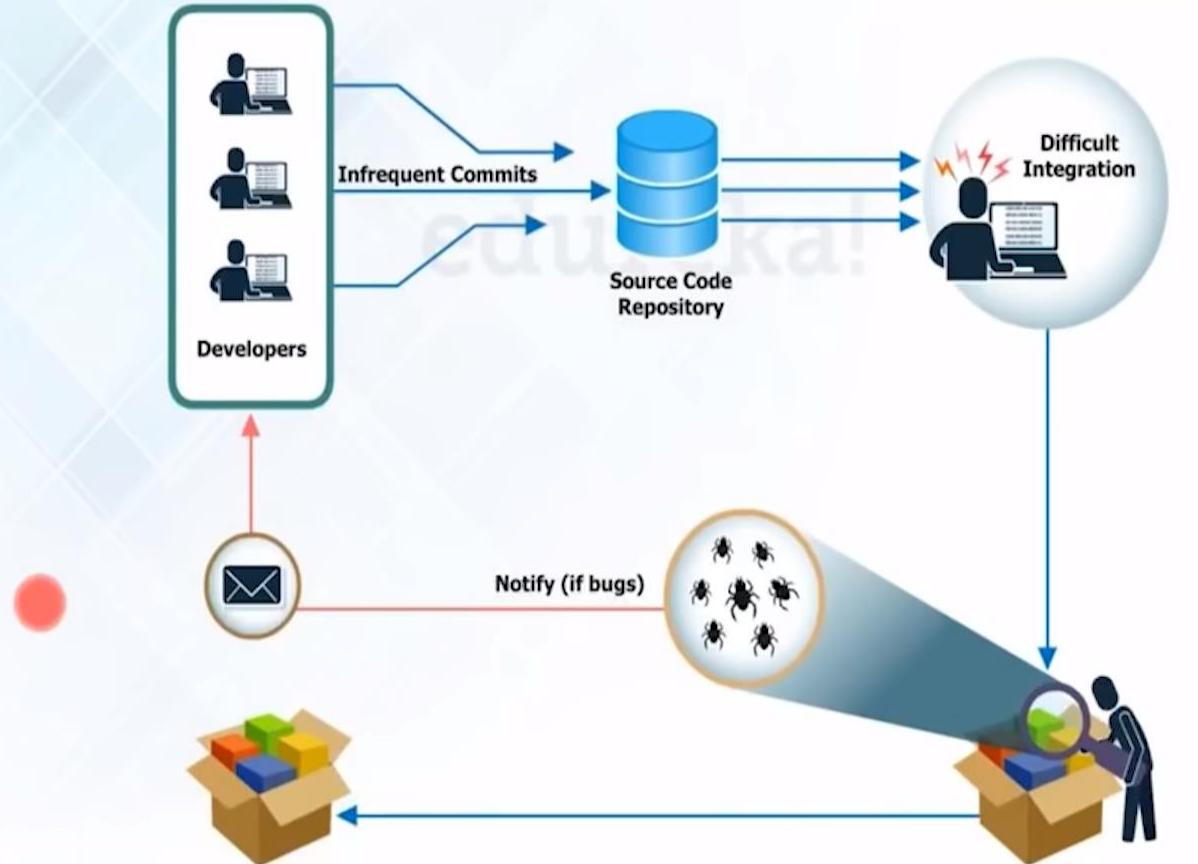
Git vs Github



Why Continuous Integration?



Process Before Continuous Integration?



Process Before Continuous Integration?

Developers have to wait till the complete software is developed for the test results.

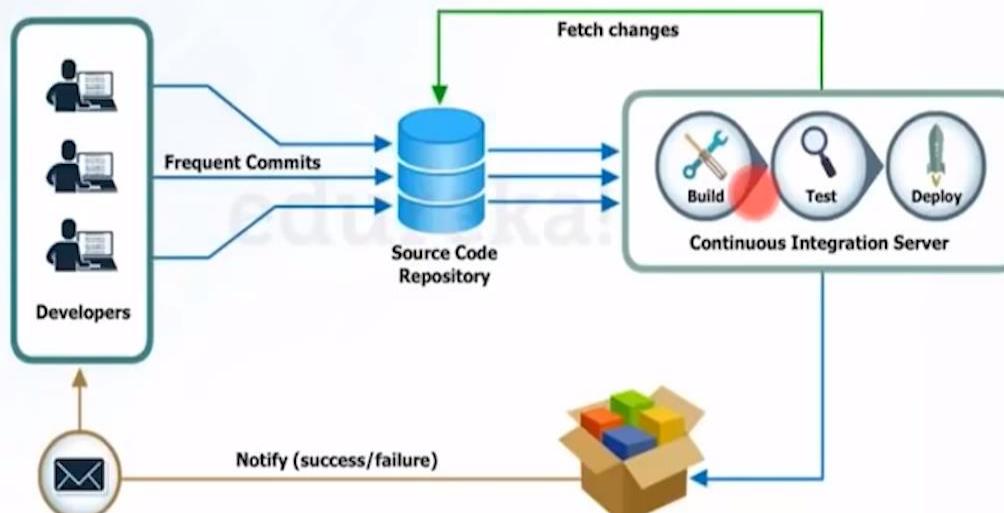


If the test fails then locating and fixing bugs is very difficult. Developers have to check the entire source code of the software.



Continuous Integration To the Rescue

- ❑ Since after every commit to the source code an auto build is triggered and then it is automatically deployed on the test server
- ❑ If the test results shows that there is a bug in the code then the developers only have to check the last commit made to the source code
- ❑ This also increases the frequency of new software releases
- ❑ The concerned teams are always provided with the relevant feedback



Continuous Integration To the Rescue

Before Continuous Integration

The entire source code was built and then tested.

Developers have to wait for test results

No Feedback

After Continuous Integration

Every commit made in the source code is built and tested.

Developers know the test result of every commit made in the source code on the run

Feedback is present

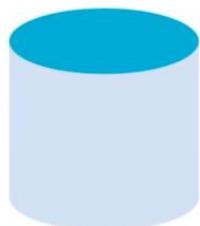
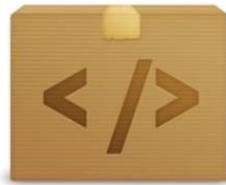


Why do we need Continuous Delivery?

Developers



Application directly deployed to prod



Code repo



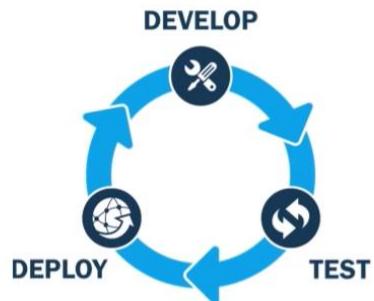
Manual testing
(UT, IT)

POSSIBLE REASONS

- Different environments (servers)
- Different libraries & packages (dependencies)
- End user load (traffic)
- App not accessible to intended audience

Why do we need Continuous Delivery?

Automates software release



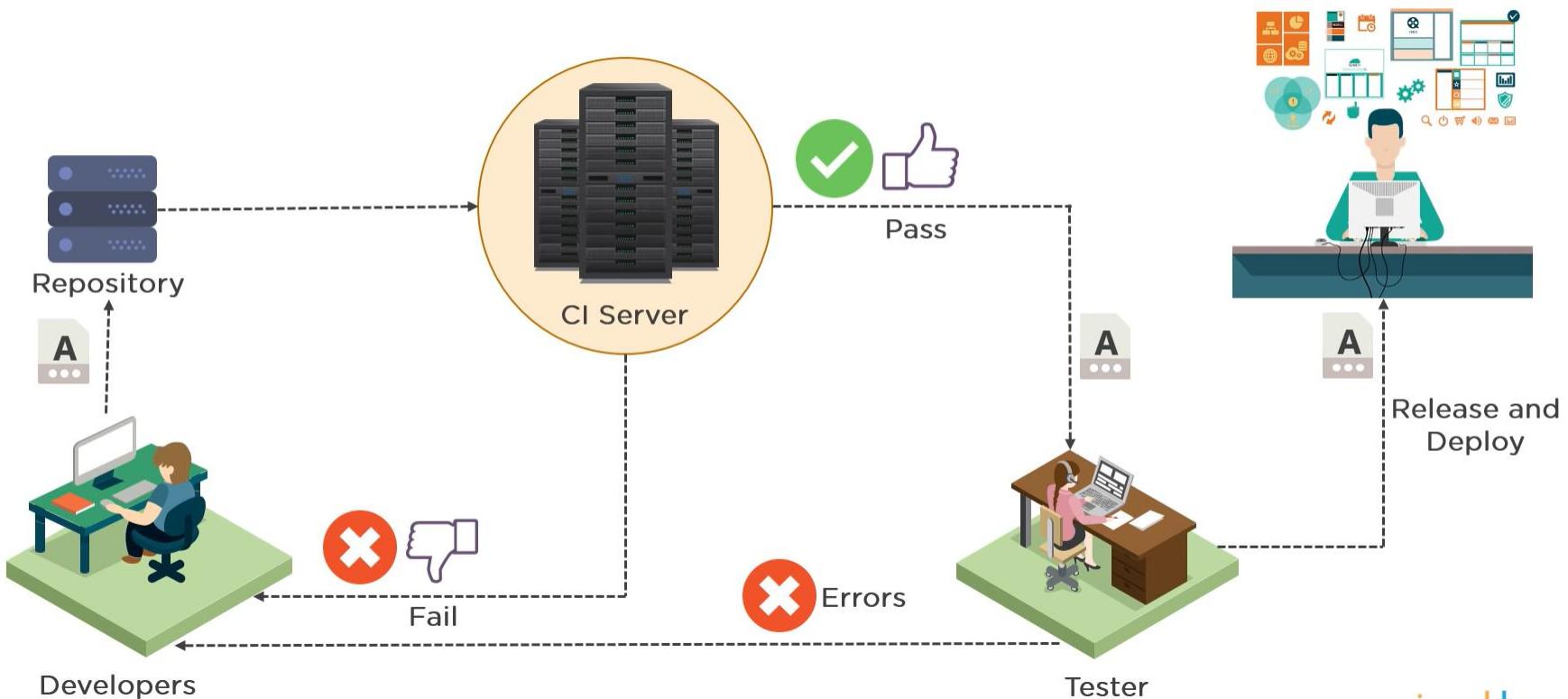
Increases developer productivity



Locates and addresses bugs quicker

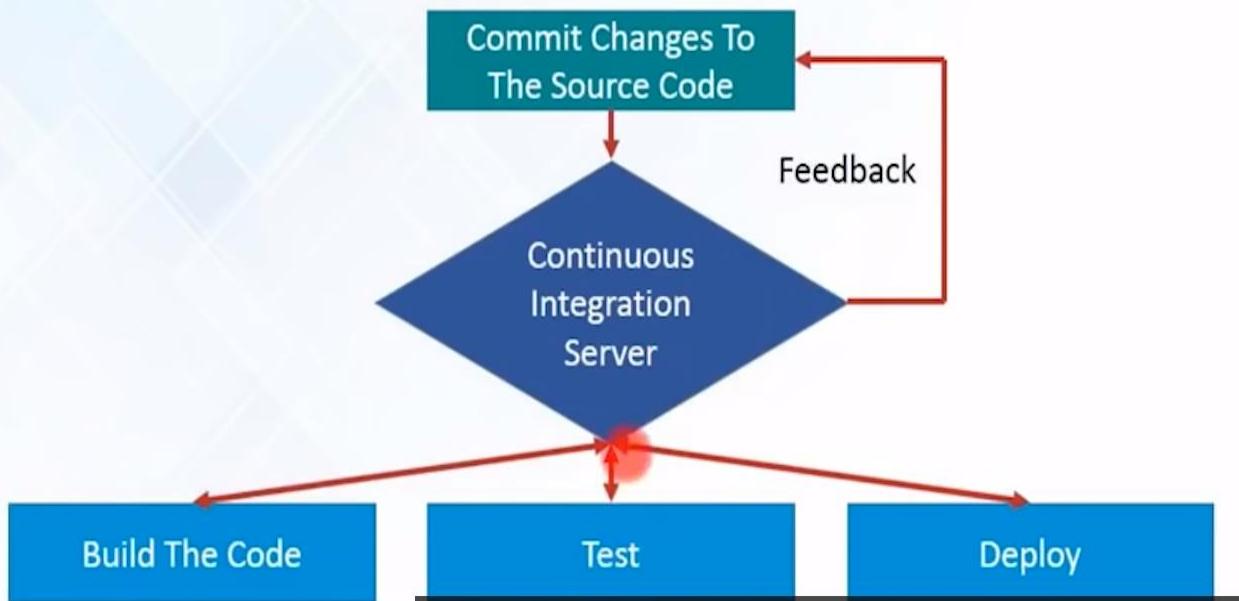


What is Continuous Integration



What is Continuous Integration

- ❑ Continuous Integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently.
- ❑ Every commit made in the repository is then built. This allows the teams to detect the problems early.

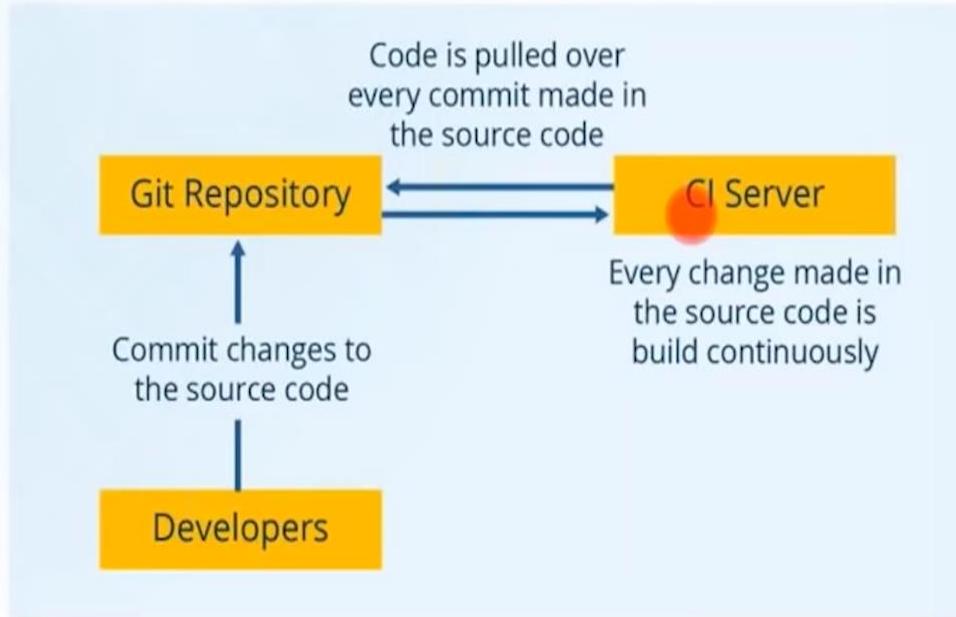


What is Continuous Integration Case Study : Nokia

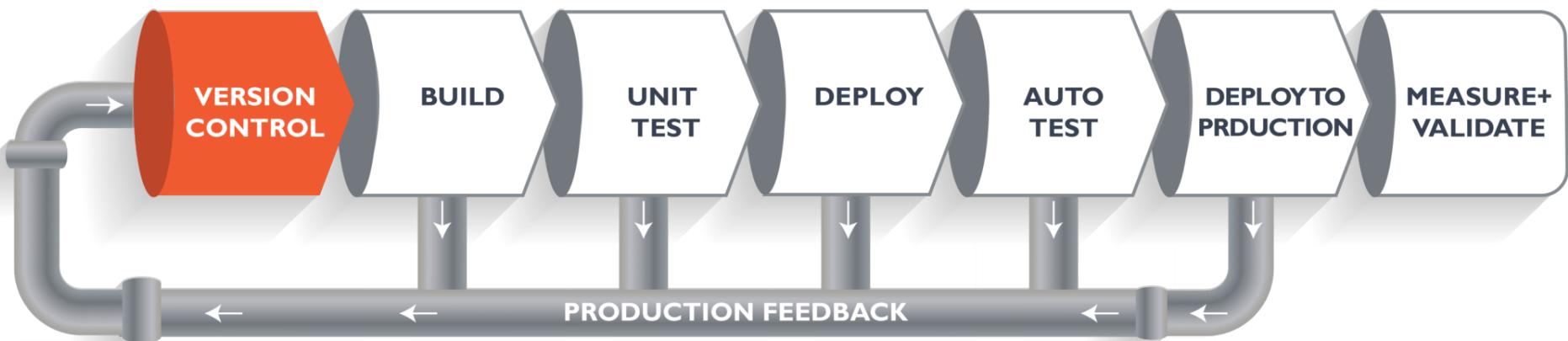
Solution:



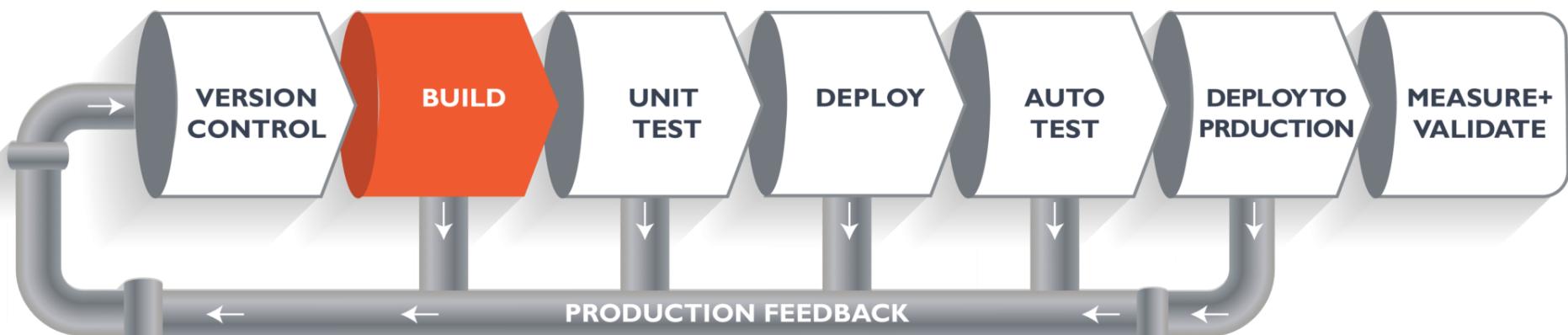
Continuous Integration



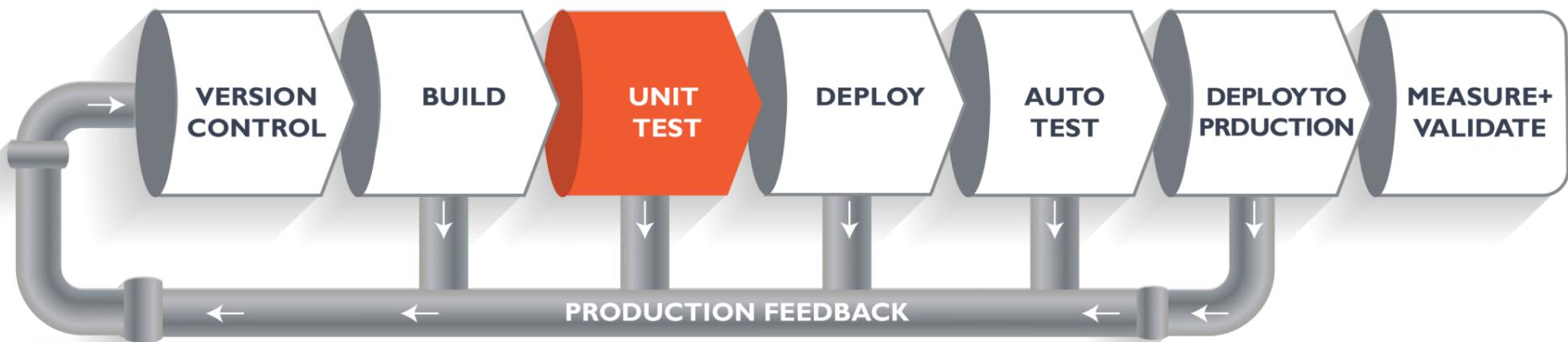
CI/CD Pipeline



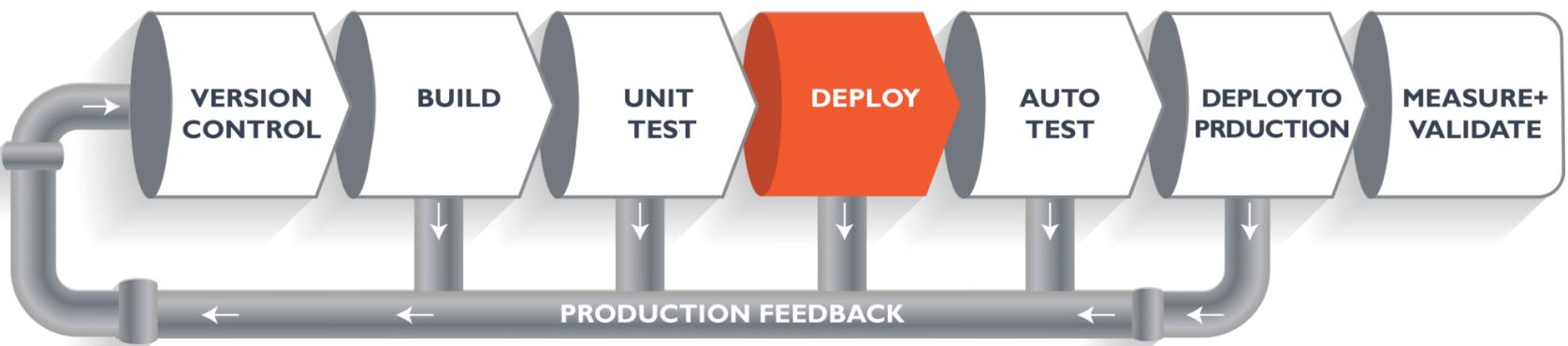
CI/CD Pipeline



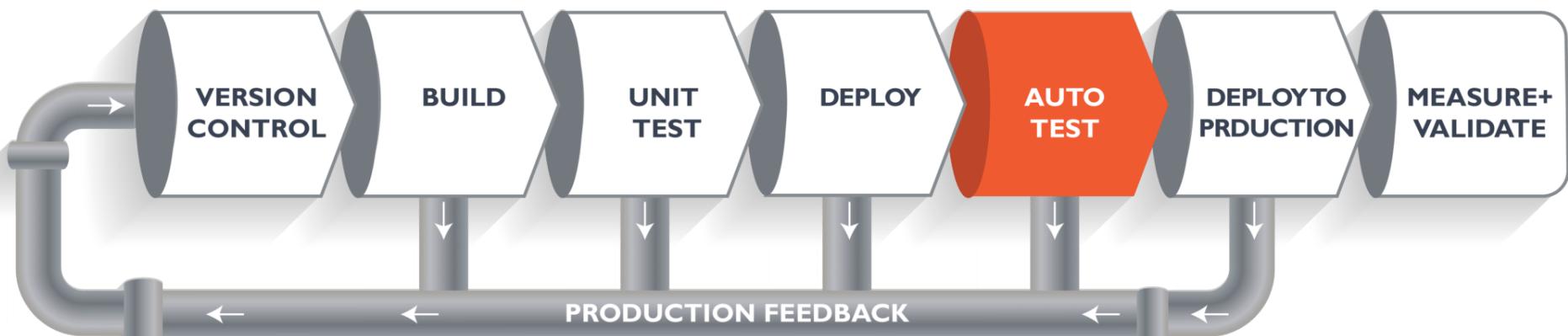
CI/CD Pipeline



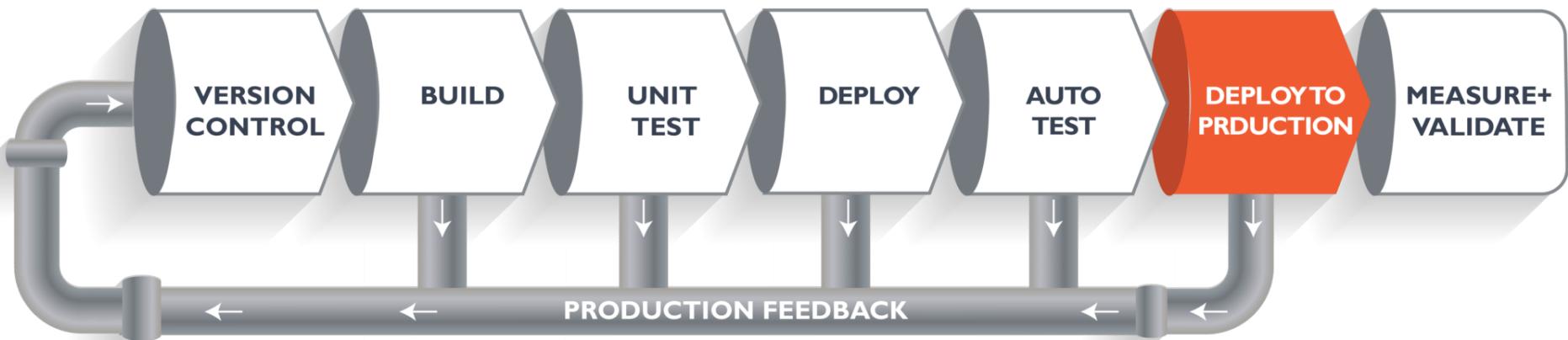
CI/CD Pipeline



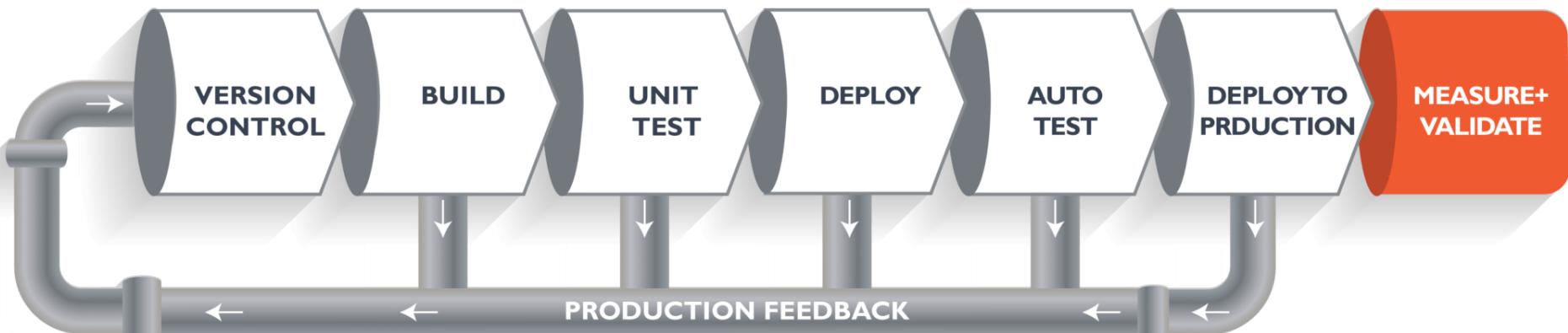
CI/CD Pipeline



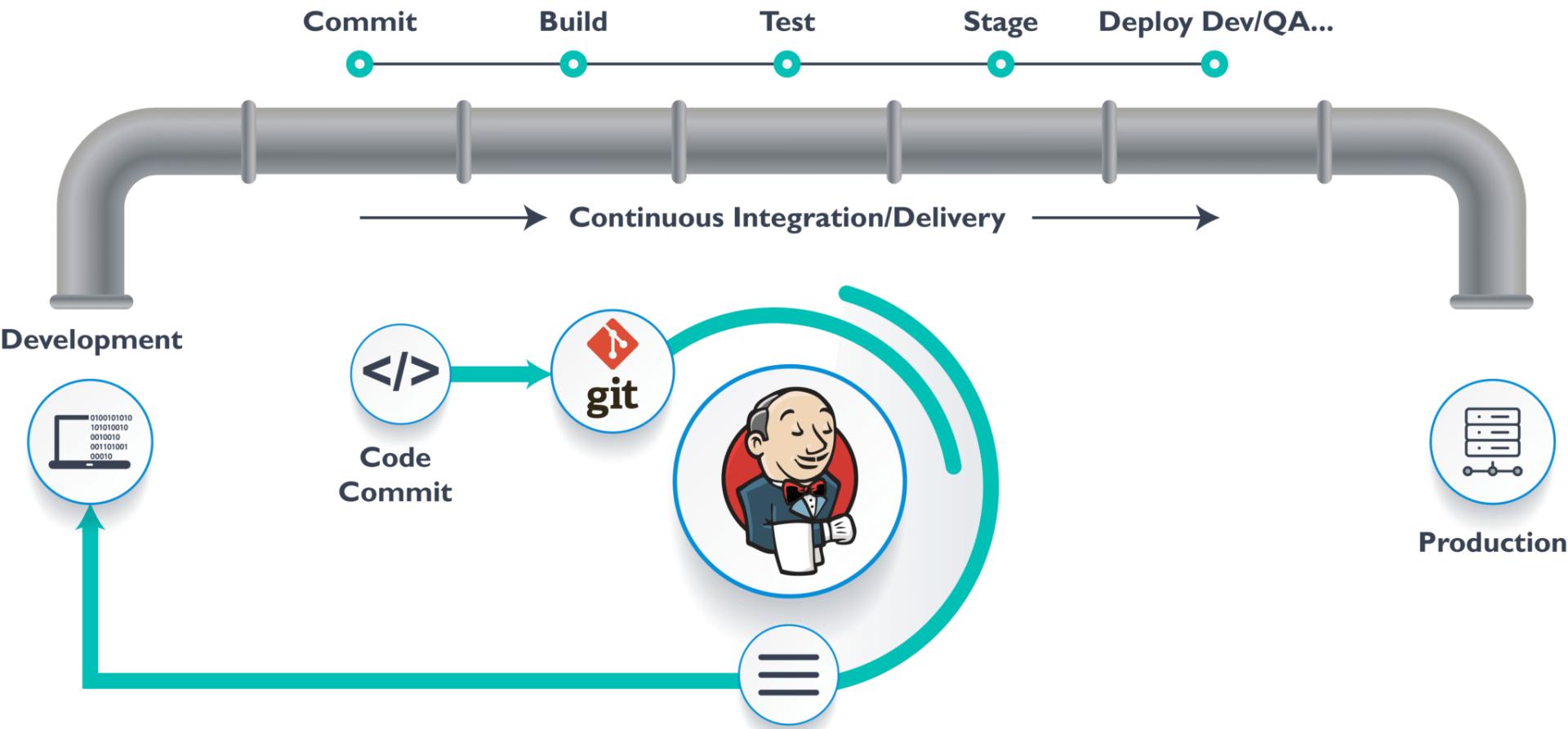
CI/CD Pipeline



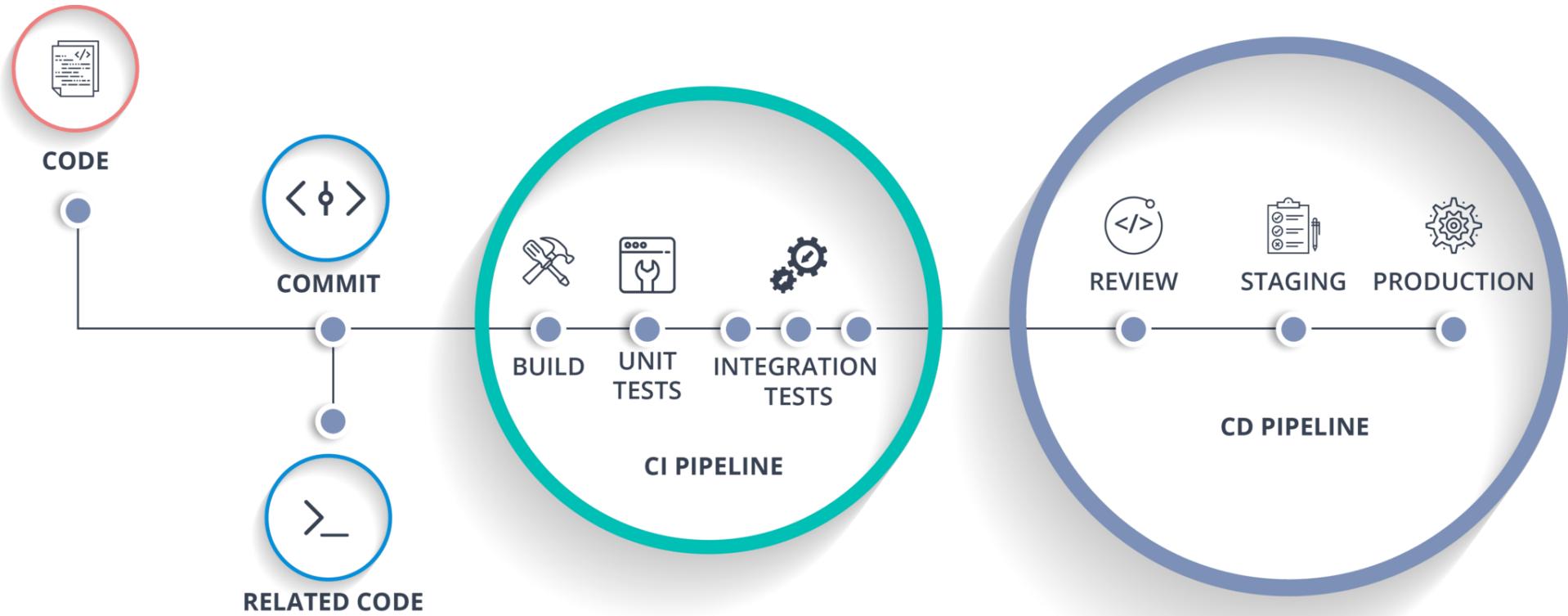
CI/CD Pipeline



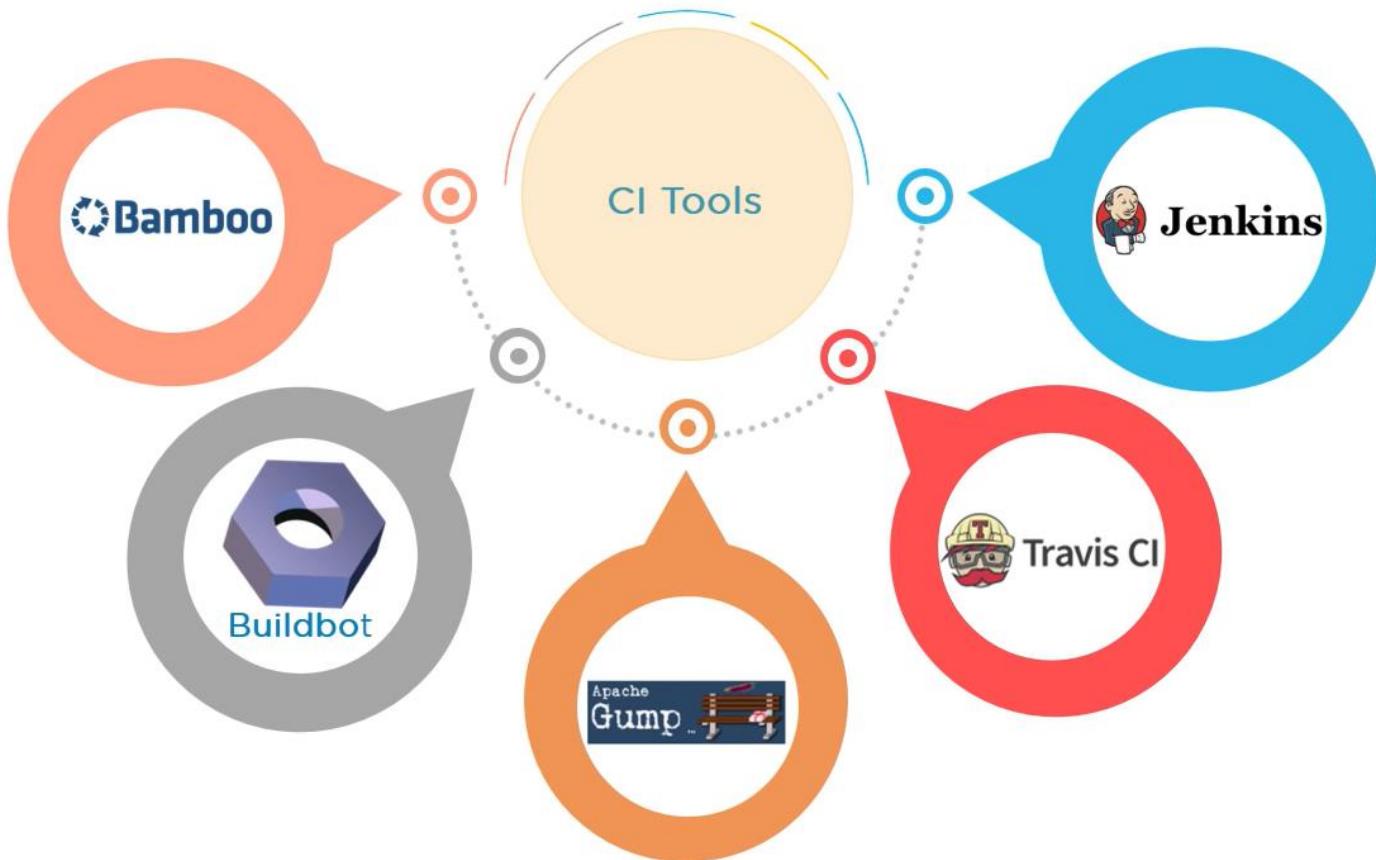
CI/CD Pipeline



CI/CD Pipeline

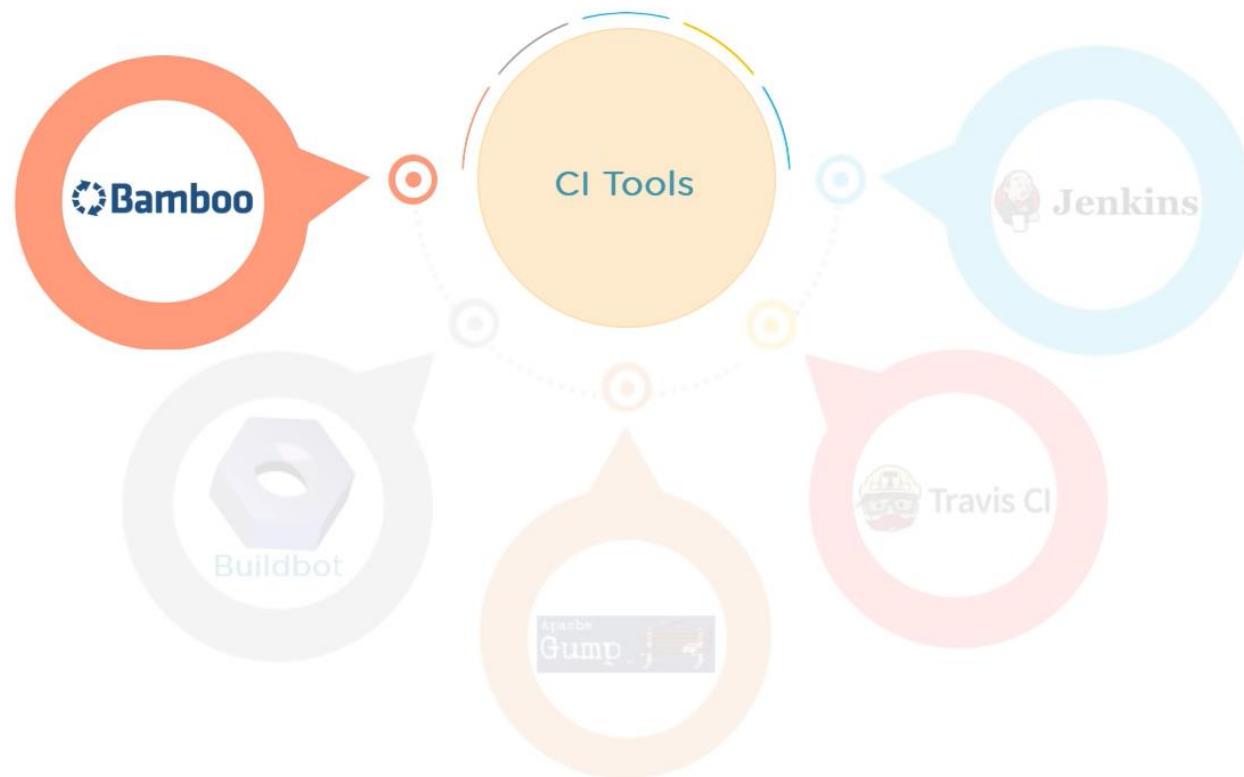


Continuous Integration Tools



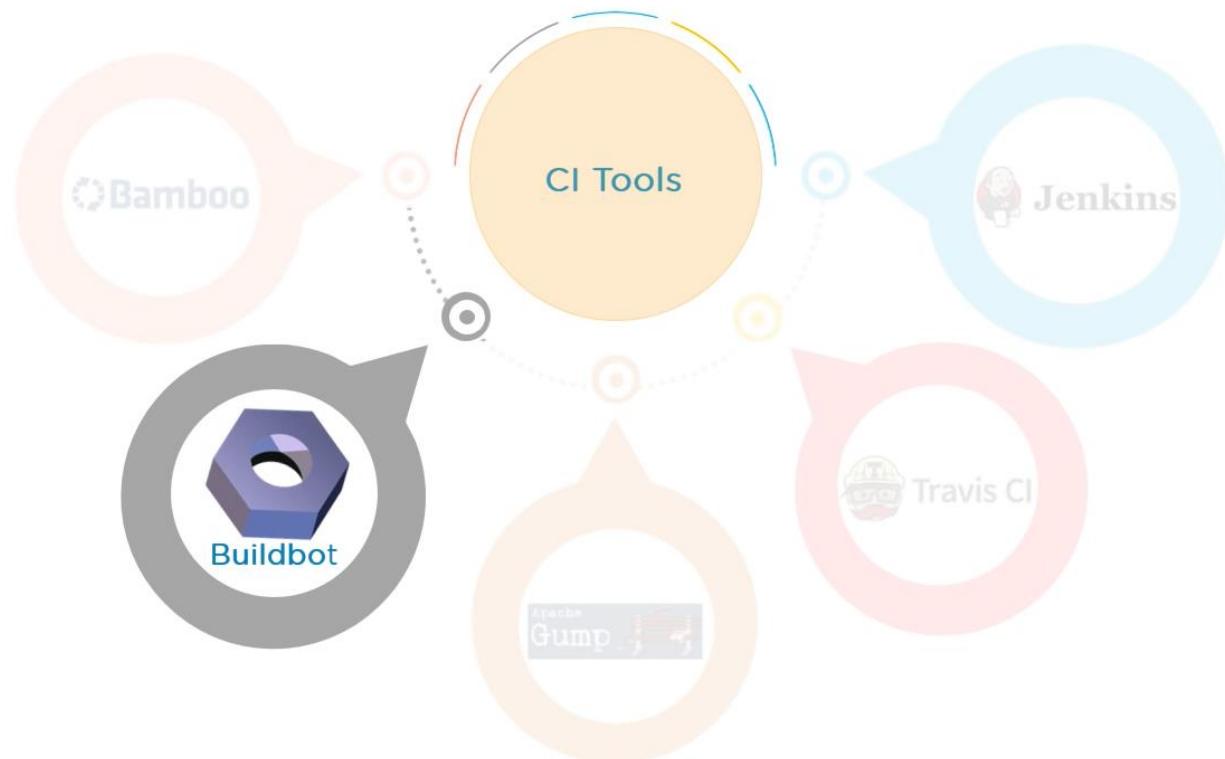
Continuous Integration Tools

Bamboo is a CI tool that can run multiple builds in parallel for faster compilation. It has built in functionality to connect with repositories and has build tasks for Ant, Maven, etc.



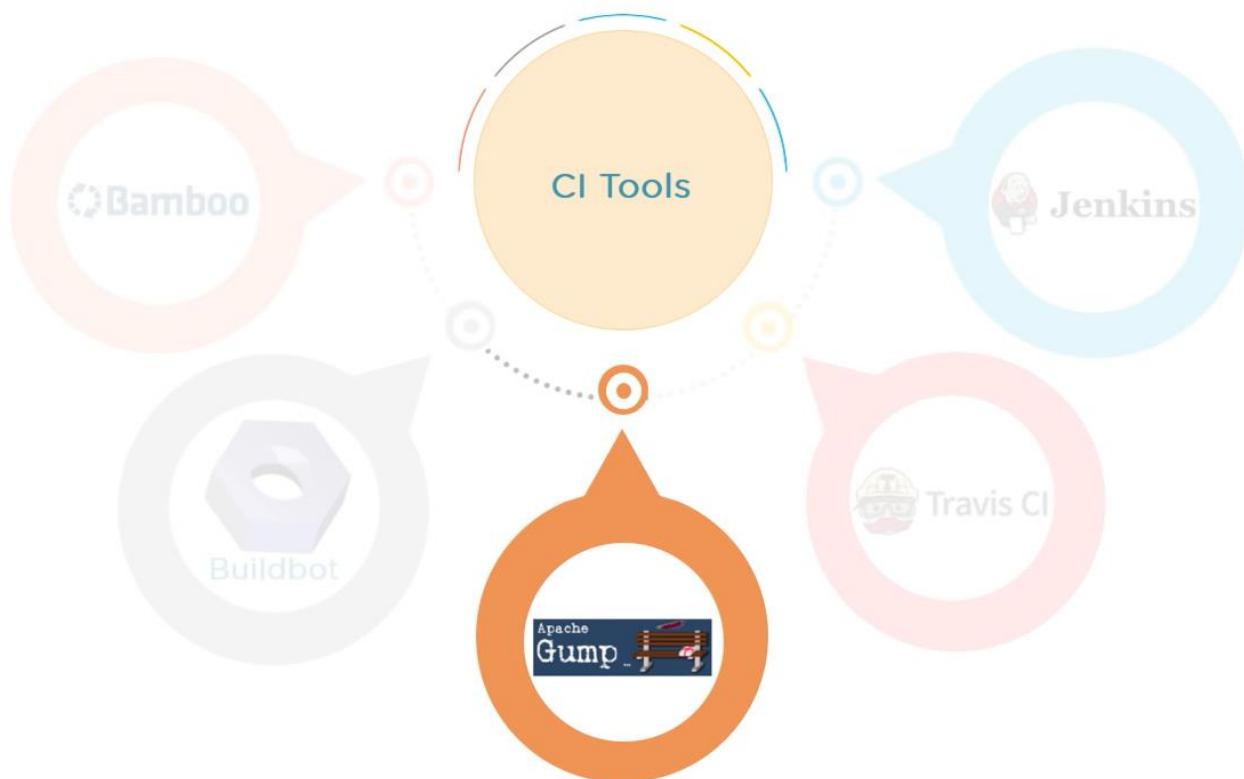
Continuous Integration Tools

Buildbot is an open-source framework for automating software build, test and release processes. It is written in Python and supports distributed, parallel execution of jobs across multiple platforms.



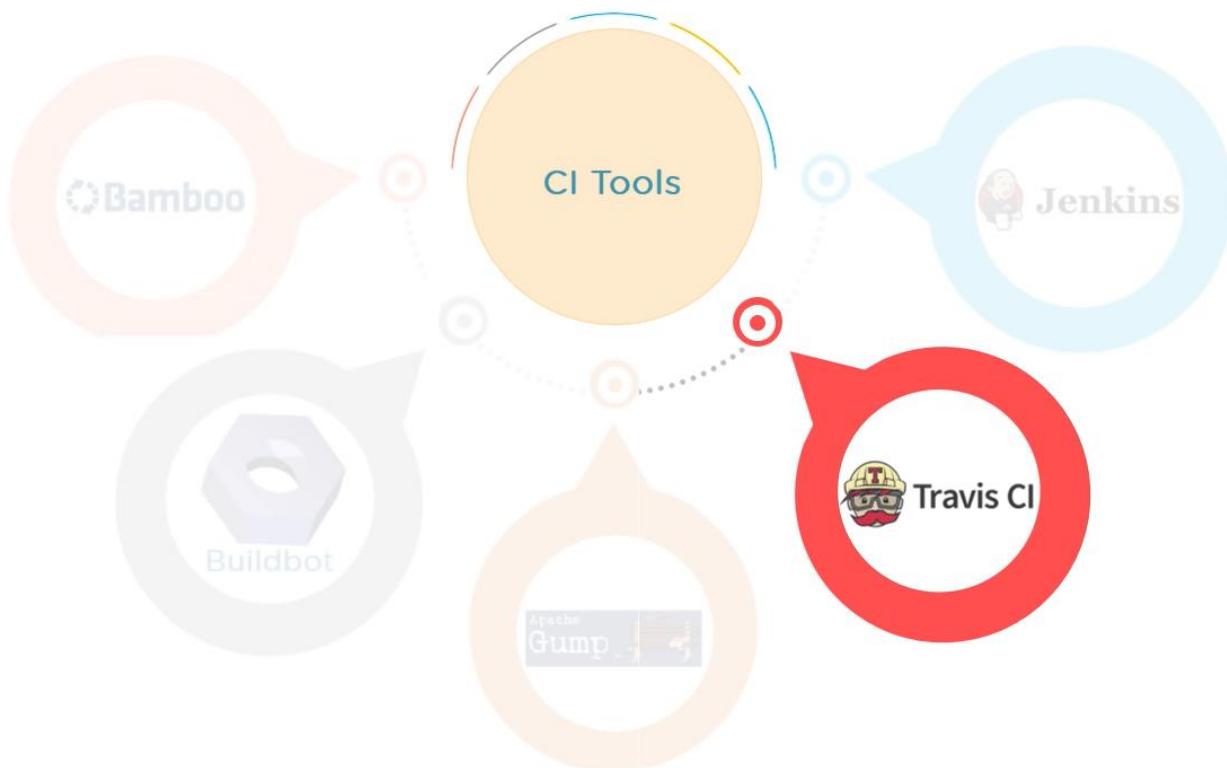
Continuous Integration Tools

Apache Gump is designed with the aim to build and test all the open source Java projects, every night. It makes sure that all the projects are compatible at both API level and functionality level.



Continuous Integration Tools

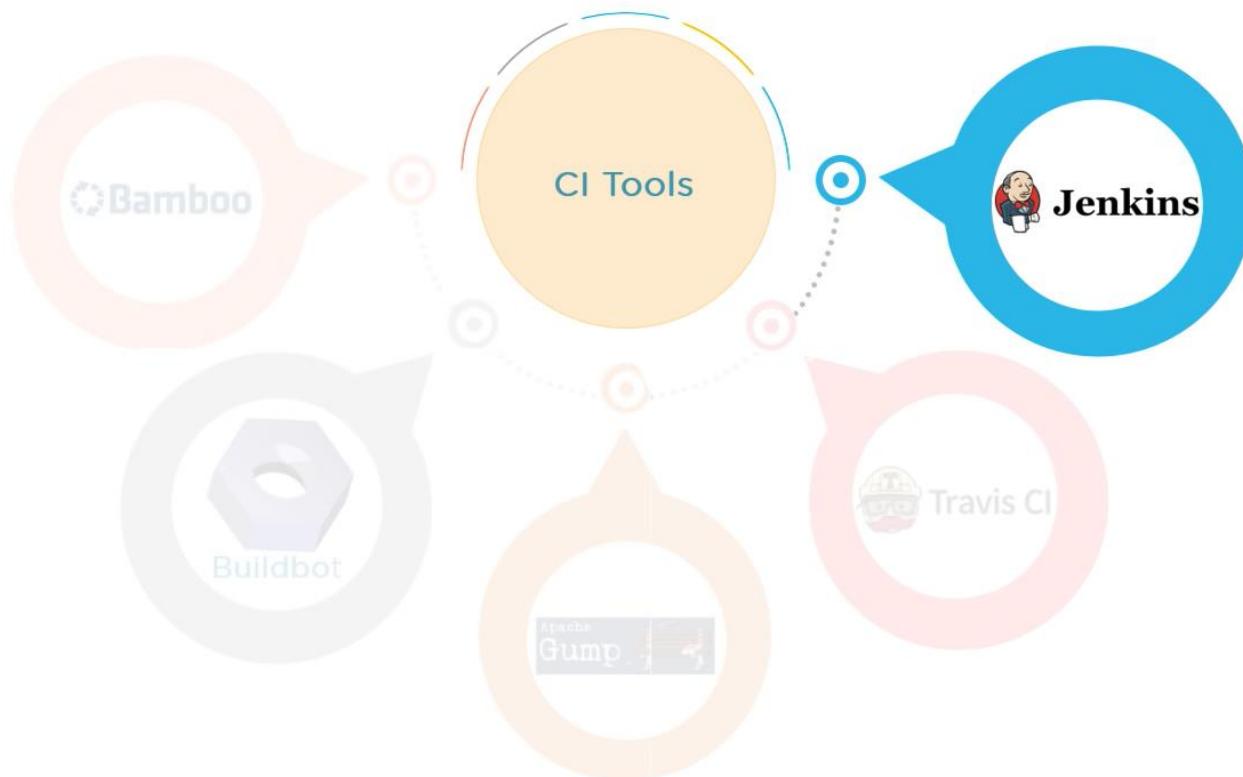
Travis CI is a hosted, distributed continuous integration service used to build and test software projects hosted at GitHub. It's built for projects and team of all sizes and supports over 20 different languages.



simpl

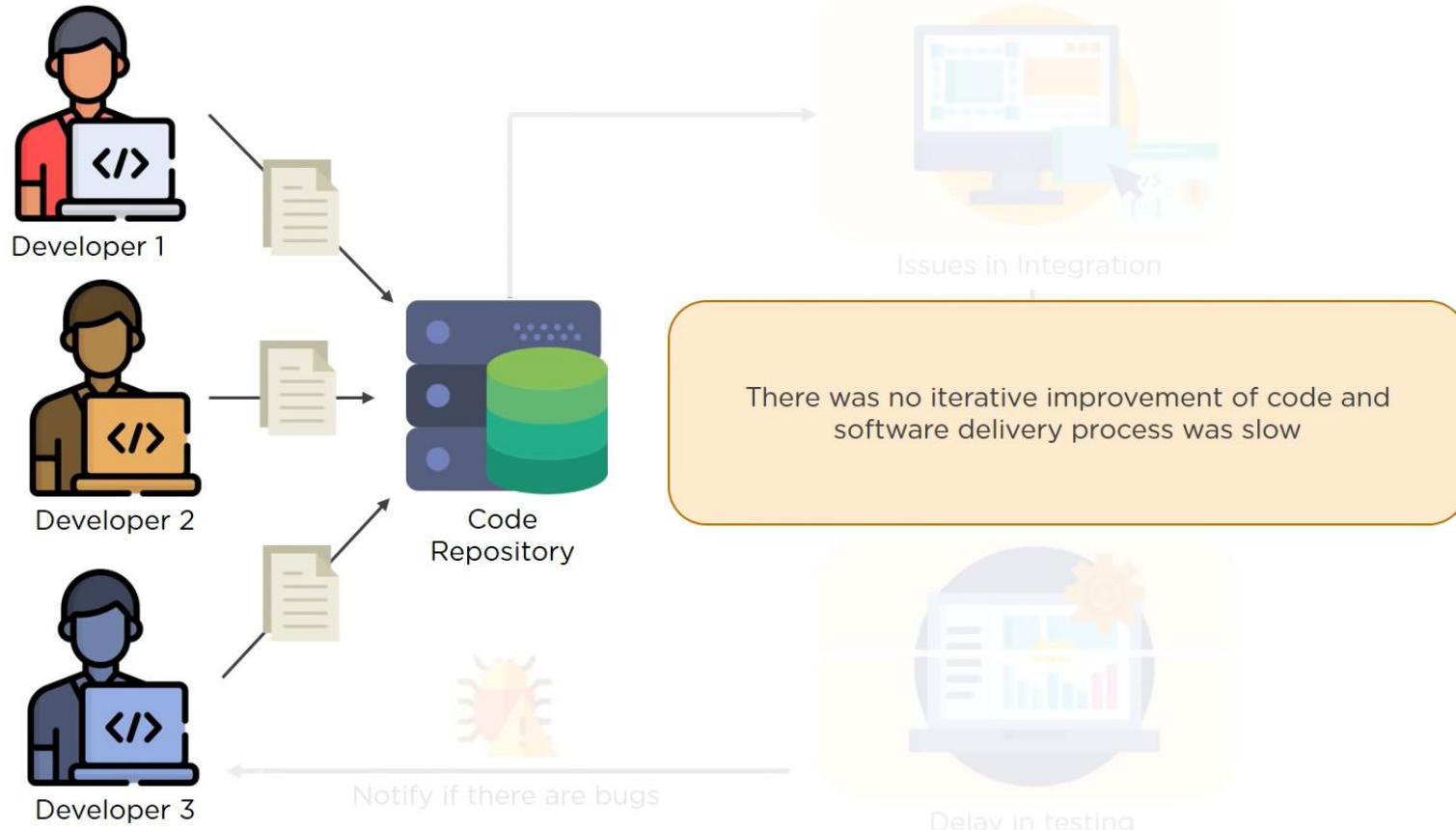
Continuous Integration Tools

Jenkins is an open source automation server written in Java. It is used to automate software development process via continuous integration and facilitates continuous delivery.



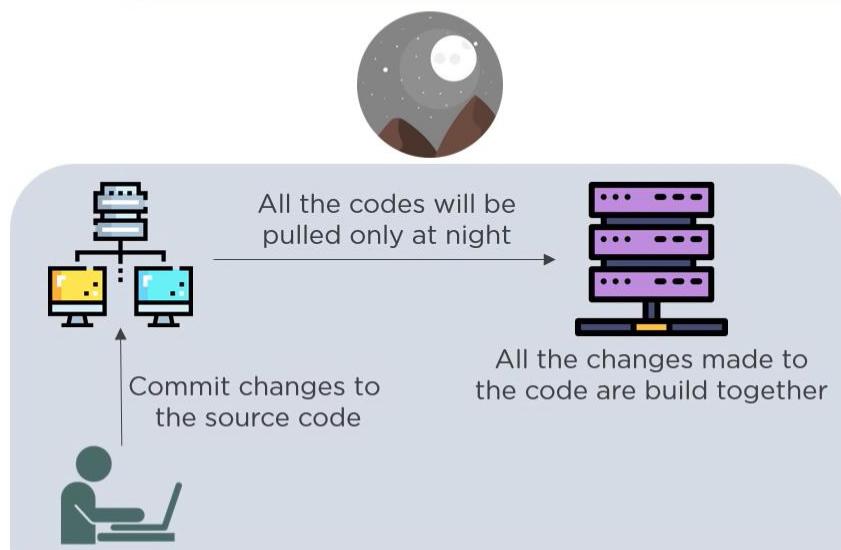
simp

Problems Before jenkins

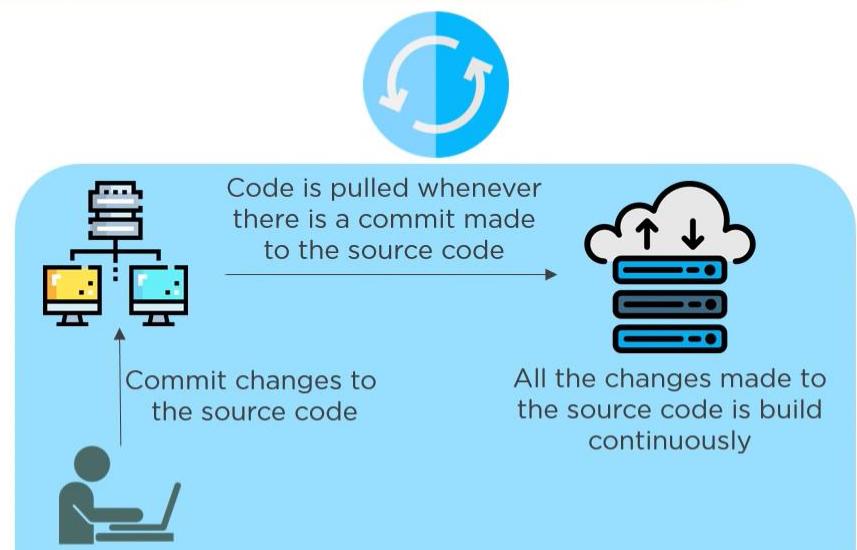


What is Jenkins

Jenkins is a Continuous Integration tool that allows continuous development, test and deployment of newly created codes



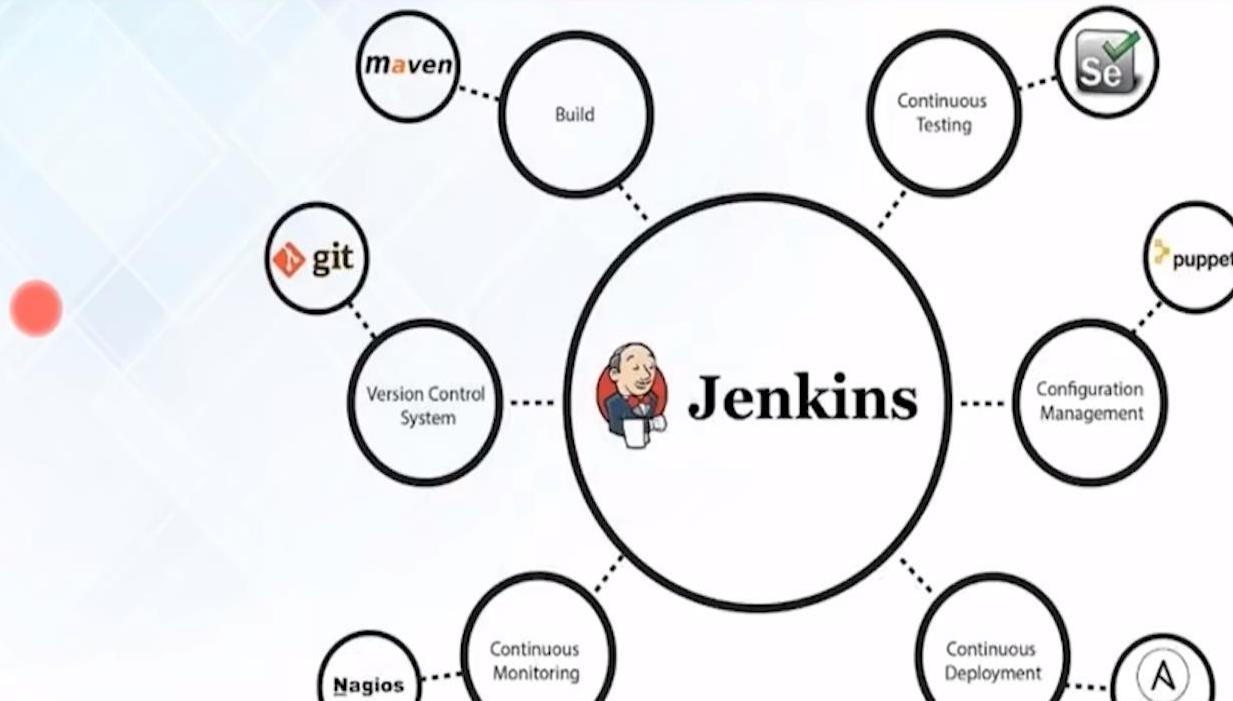
Nightly build and integration



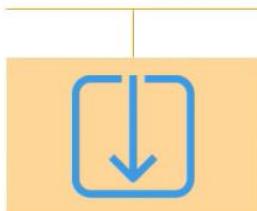
Continuous build and Integration

What is Jenkins

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Plugins allows integration of various DevOps stages.



Jenkins Features



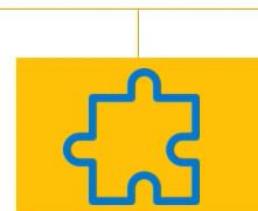
Easy
Installation



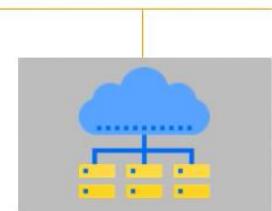
Easy
Configuration



Plugins



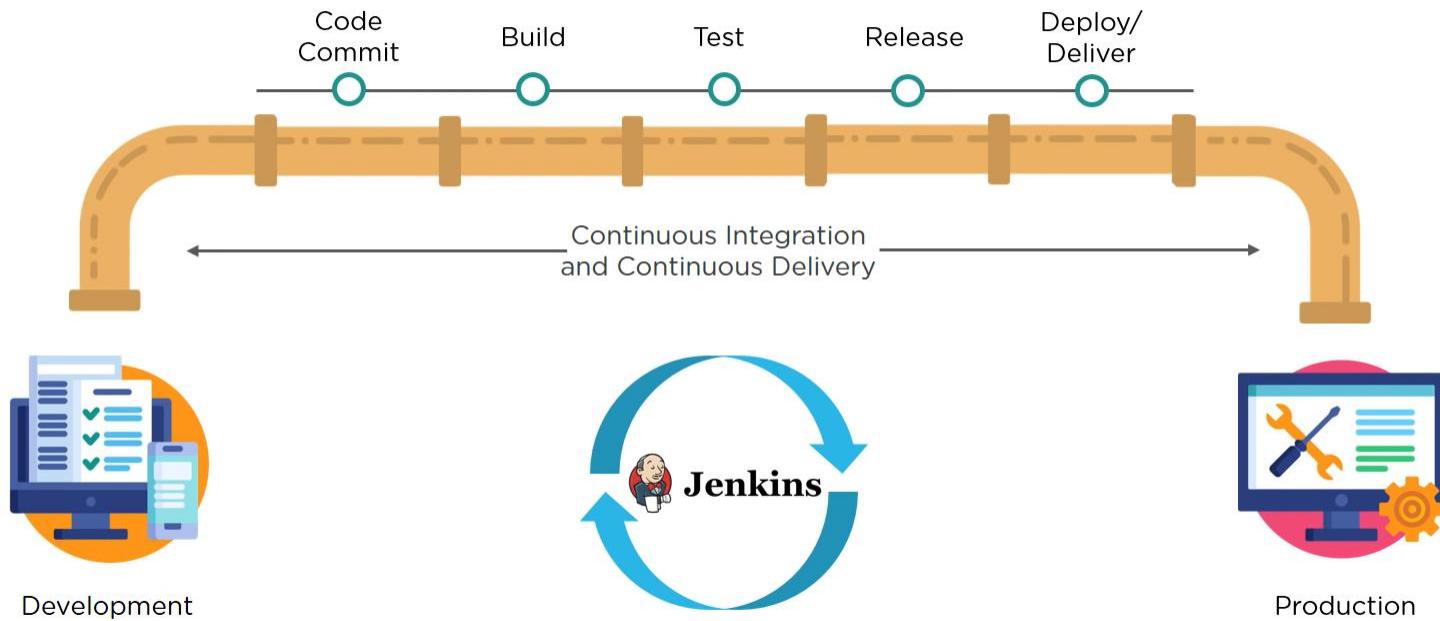
Extensible



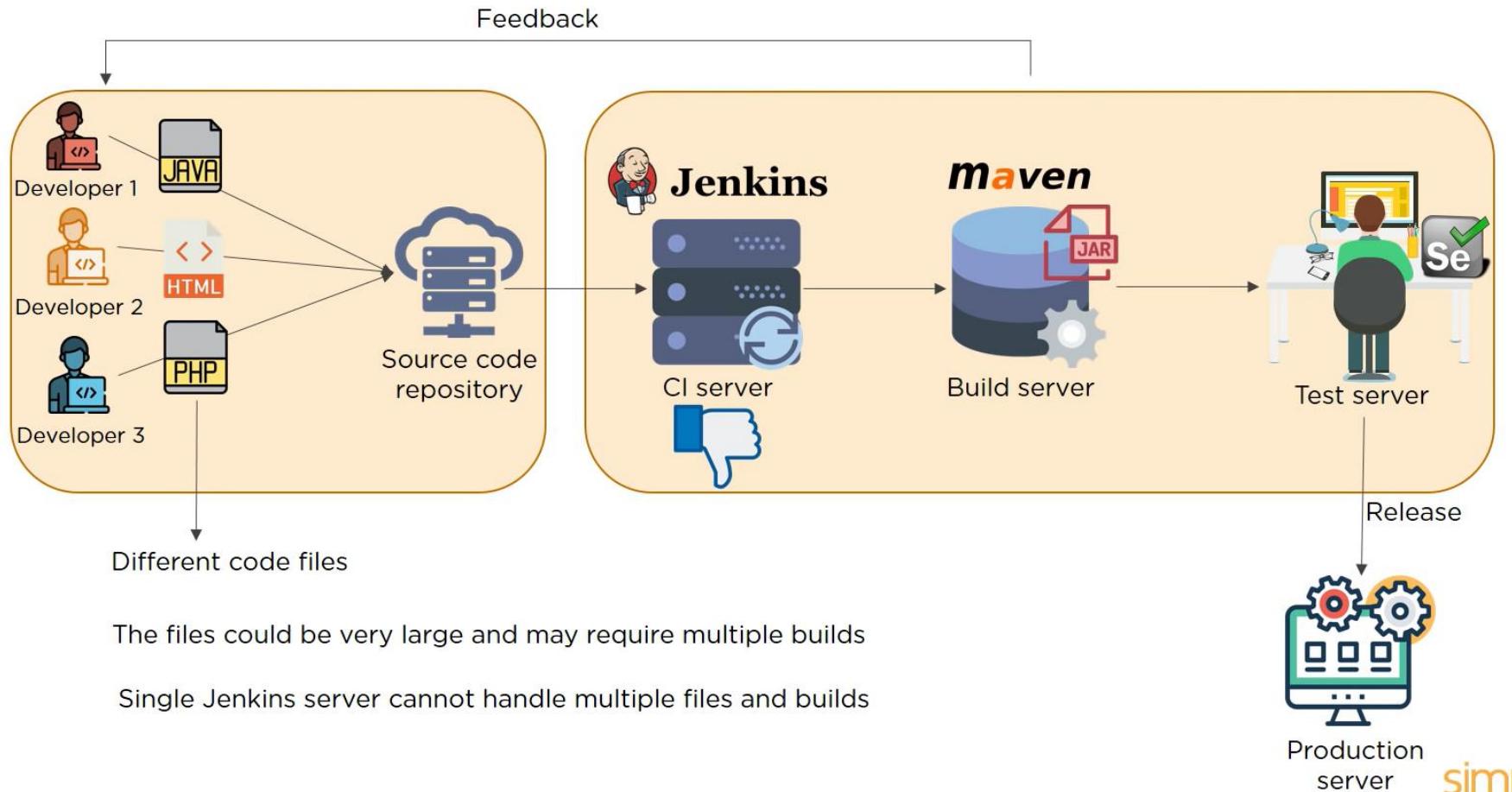
Distributed

Jenkins is a self contained Java-based program, ready to run with packages for Windows, Mac OS X and Unix-like OS

Jenkins Pipeline



Jenkins Architecture



Shortcomings of Single Jenkins Server



If you need to run web tests using Internet Explorer, you need to use a Windows machine.

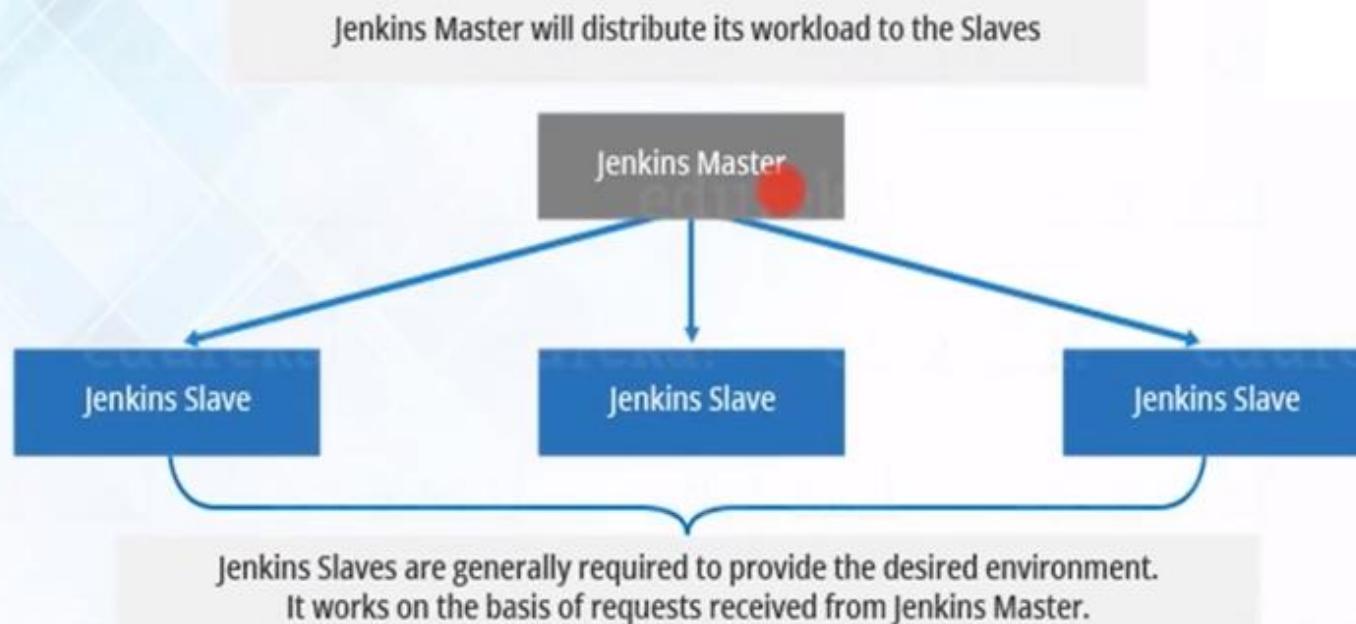


Another build job may require Linux box.

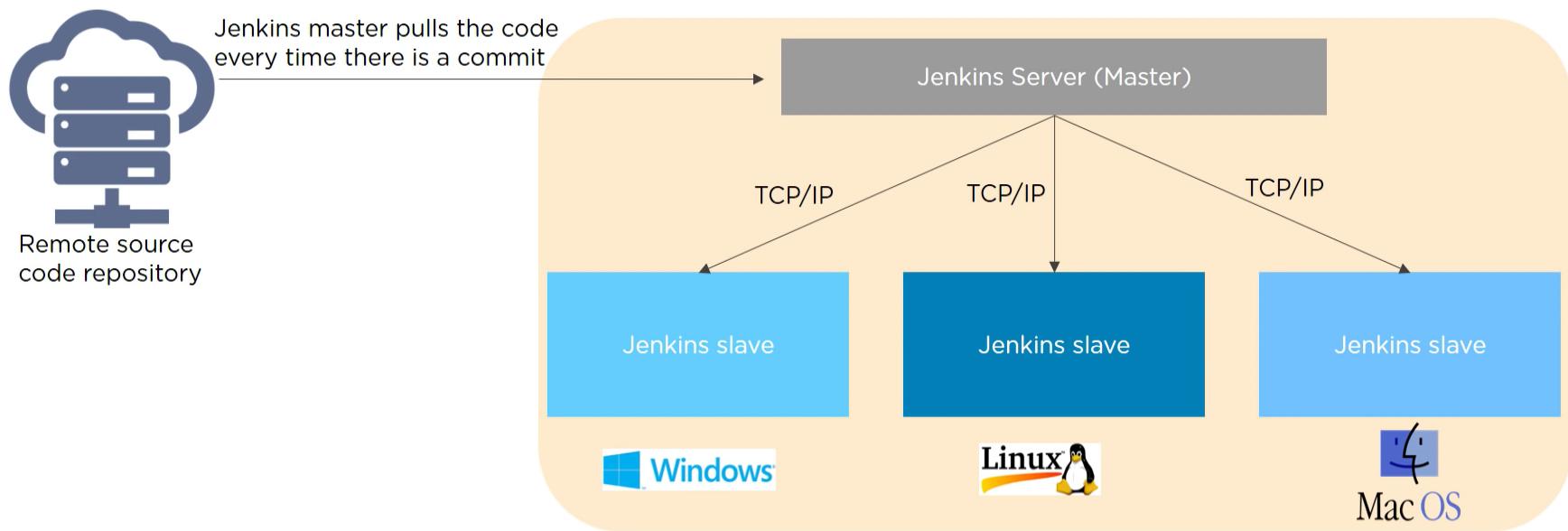


How to manage spikes in build activity

Jenkins Distributed Architecture



Jenkins Master Slave Architecture



- Jenkins master distributes its workload to all the slaves
- On request from Jenkins master, the slaves carry out builds and tests and produce test reports

Jenkins Case study



The illustration features the Bosch logo at the top right. Below it, a man in a suit sits at a desk with a laptop, looking thoughtful. To his left is a red car icon. In the bottom left corner, there's a robotic arm icon with gears. The background shows a blurred factory or warehouse setting with shipping containers.

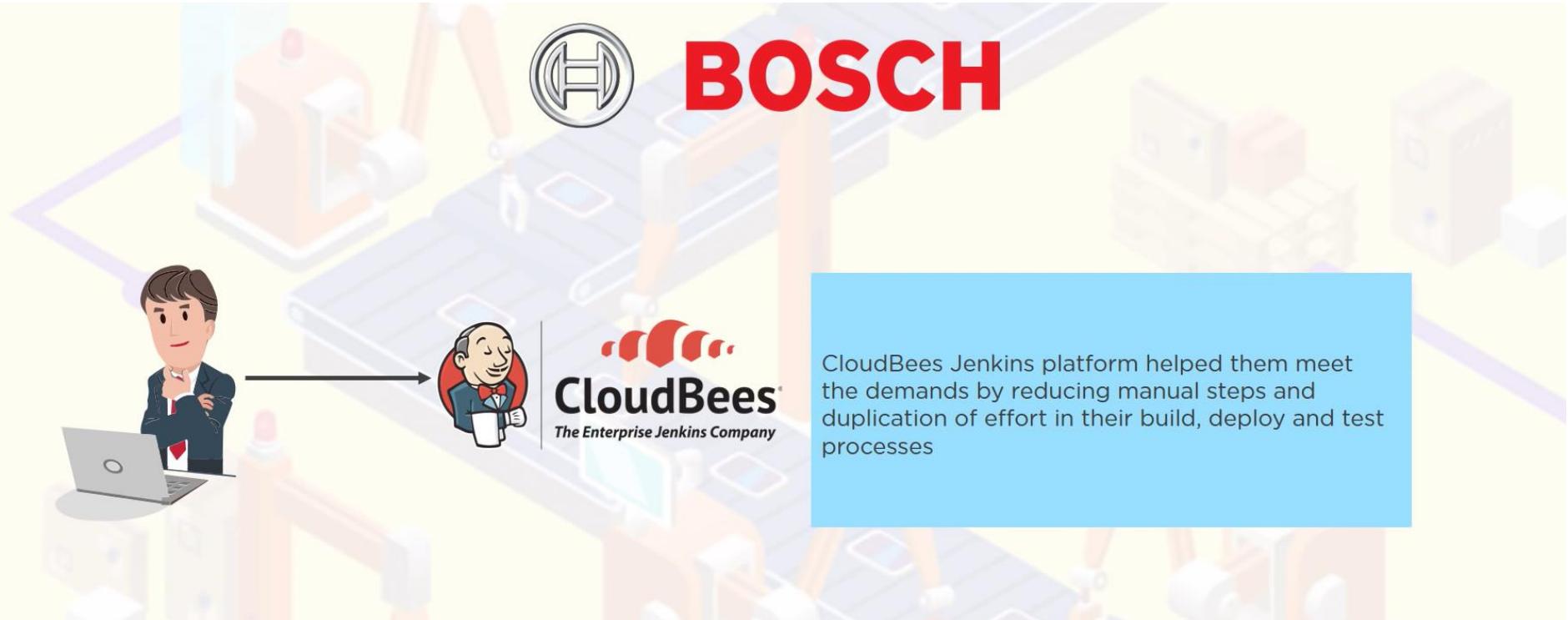
BOSCH

BOSCH found a growing need to help its software engineers produce and deliver higher quality software faster

CHALLANGE

Manage and streamline the development of increasingly complex automotive software by adopting CI and CD practices to shorten the entire development and delivery process

Jenkins Case study



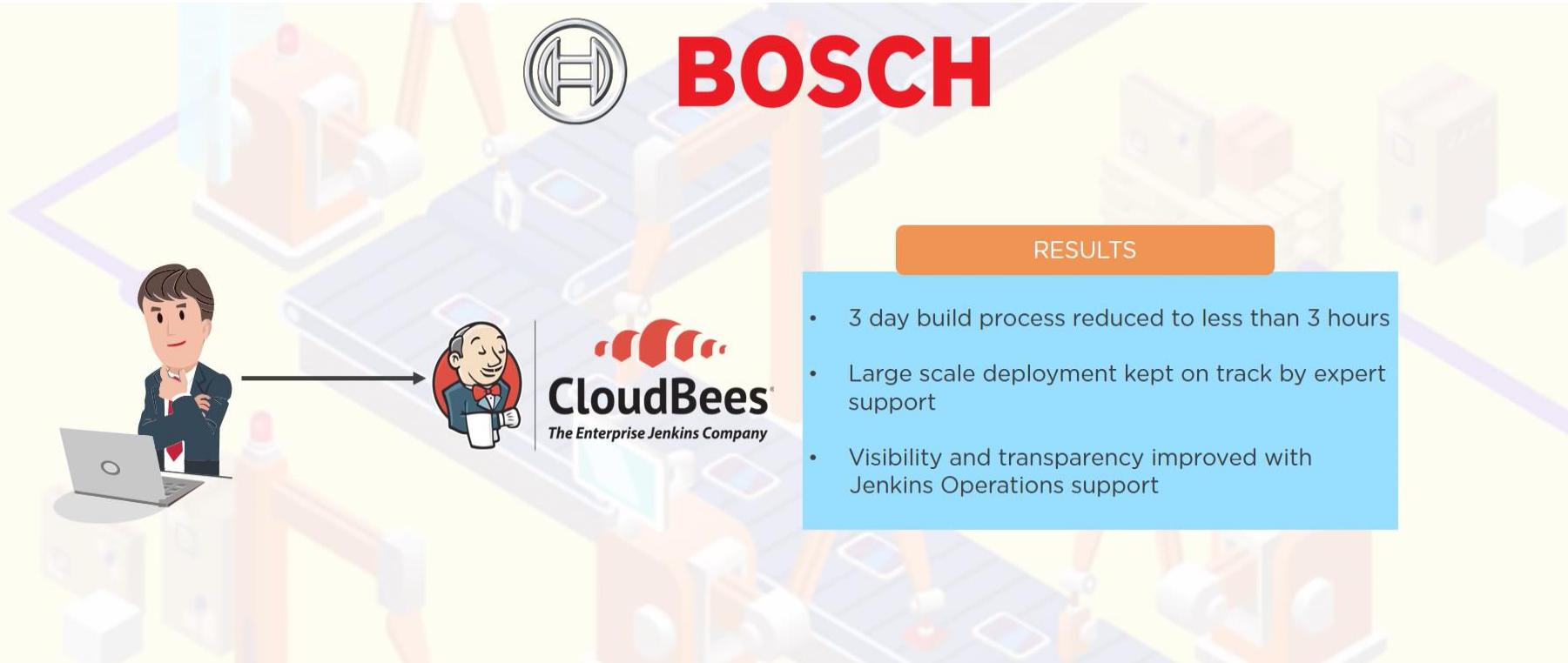
The background of the slide features the Bosch logo (a stylized 'H' inside a circle) and the CloudBees logo ('The Enterprise Jenkins Company'). The scene is set against a backdrop of several industrial robots in a factory environment.

BOSCH

CloudBees
The Enterprise Jenkins Company

CloudBees Jenkins platform helped them meet the demands by reducing manual steps and duplication of effort in their build, deploy and test processes

Jenkins Case study



The background of the slide features a stylized illustration of a factory or industrial setting. It shows various pieces of machinery, conveyor belts, and a worker in the distance. In the foreground, there's a man in a suit sitting at a desk with a laptop, looking towards the right. An arrow points from him to a logo for CloudBees, which is described as "The Enterprise Jenkins Company". The CloudBees logo consists of a red bee icon above the word "CloudBees" and the tagline "The Enterprise Jenkins Company" below it. To the right of the CloudBees logo, there's a large Bosch logo, featuring a circular emblem with a stylized letter 'H' and the word "BOSCH" in red capital letters.

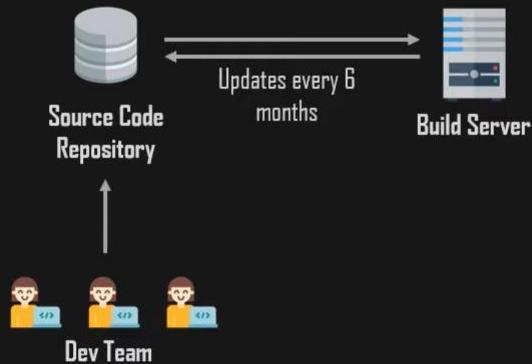
RESULTS

- 3 day build process reduced to less than 3 hours
- Large scale deployment kept on track by expert support
- Visibility and transparency improved with Jenkins Operations support

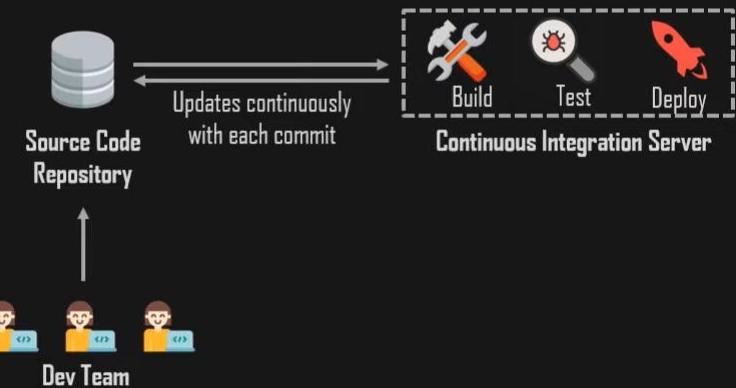
Jenkins Case study

CASE STUDY: ADOBE

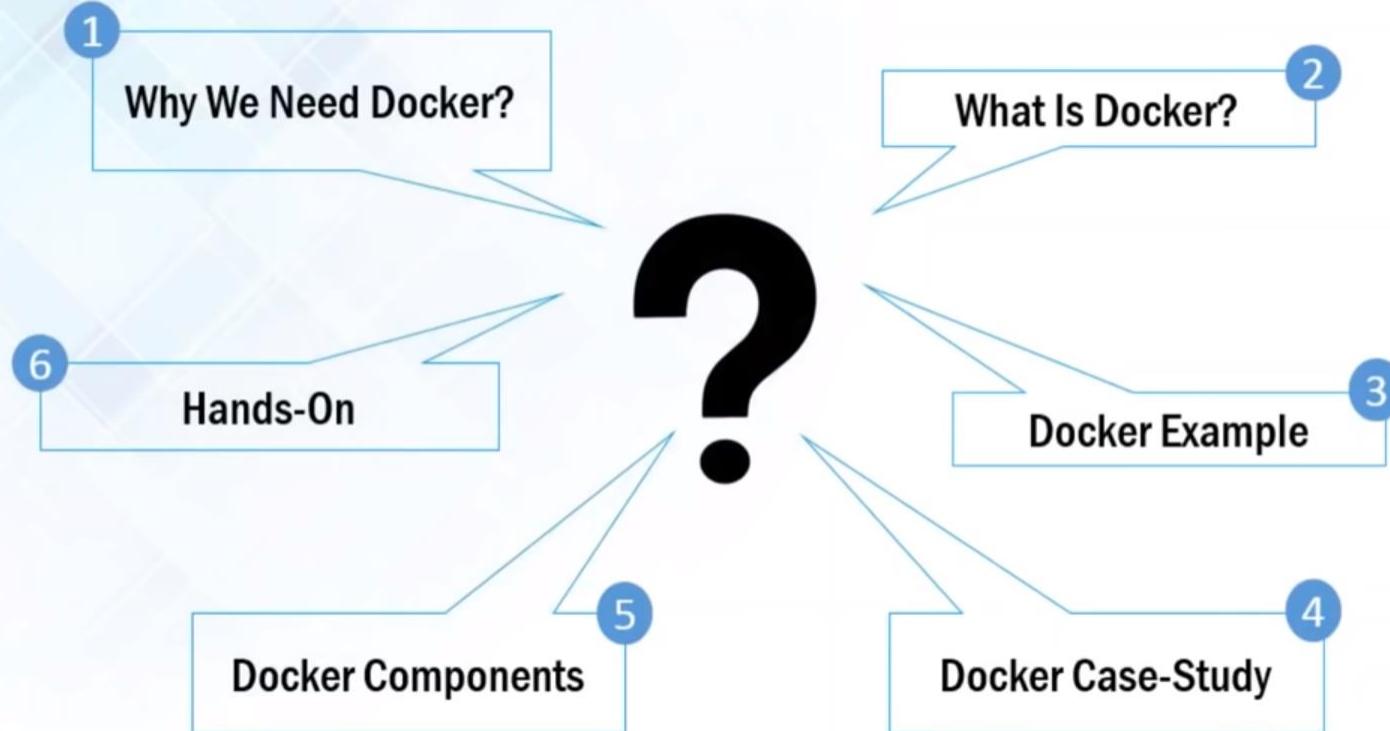
Adobe before CI: Semi Annual Releases



Adobe after CI: 60% Faster Releases

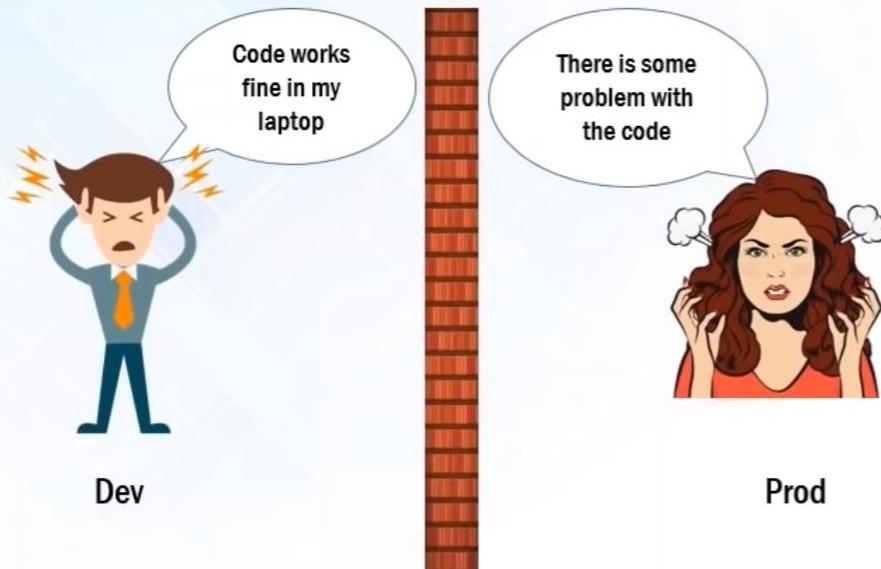


Do you know the following?



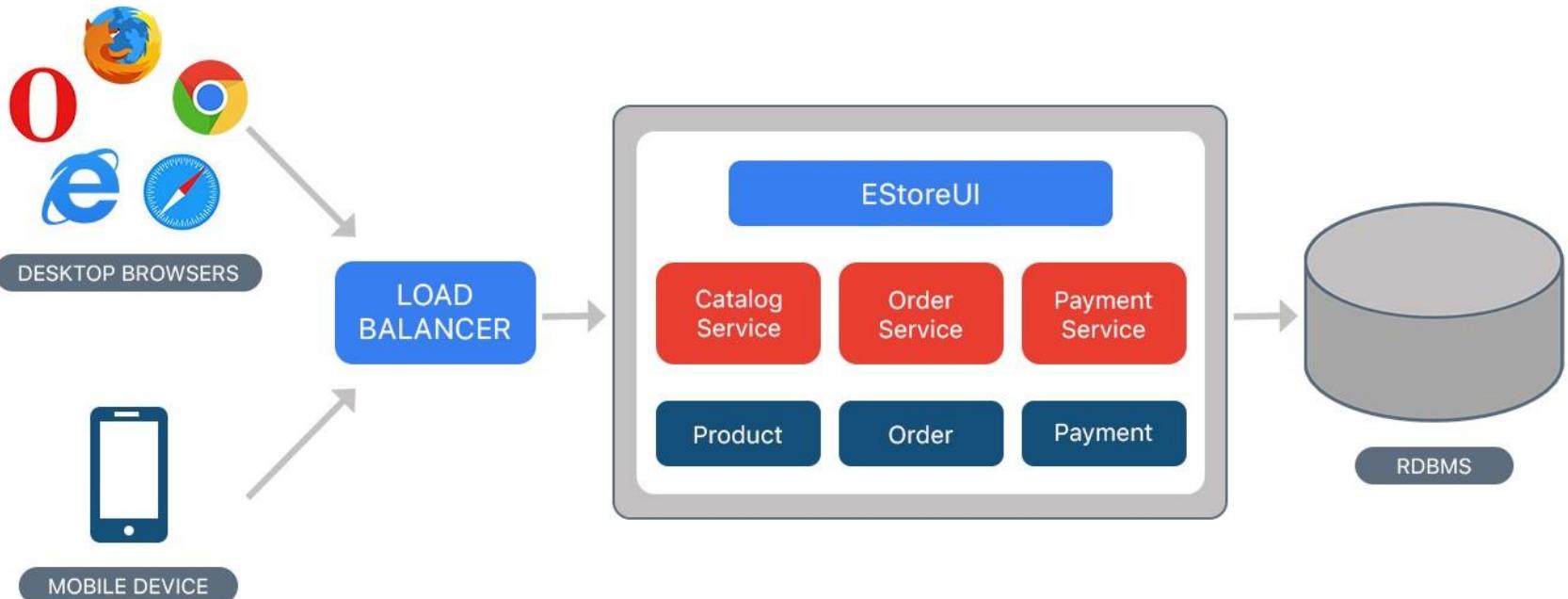
Problems Before Docker

An application works in developer's laptop but not in testing or production. This is due to difference in computing environment between Dev, Test and Prod.

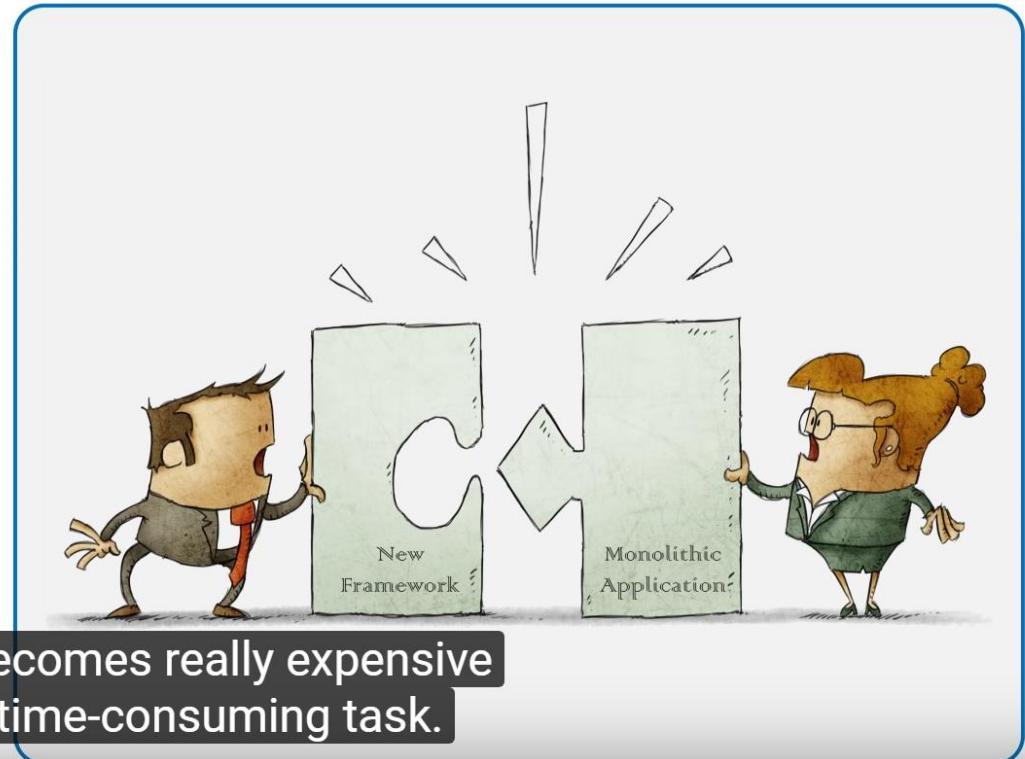


In Dev there can be a software that is upgraded and in Prod the old version of software might be present

Example of Monolithic Approach

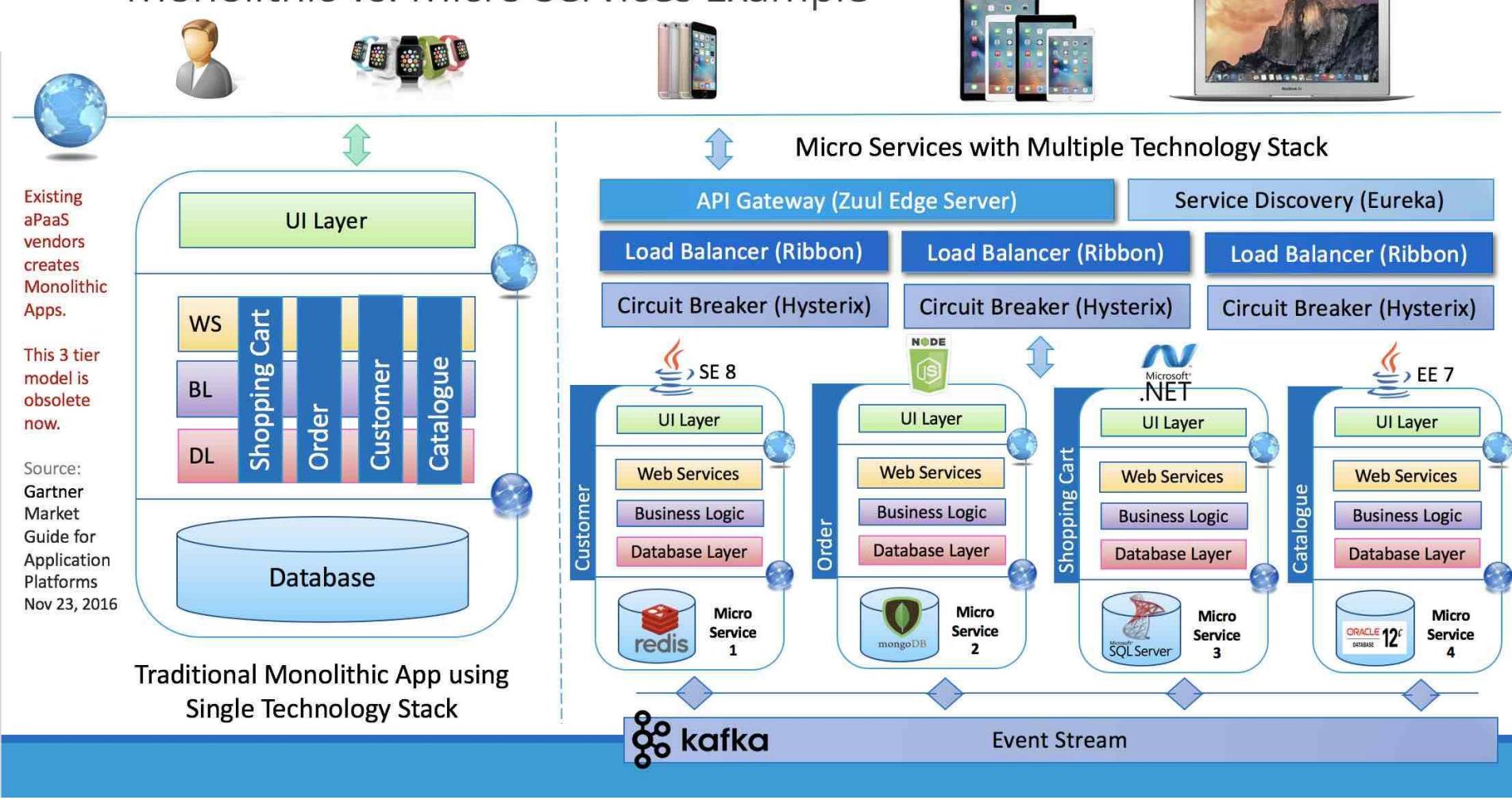


Monolithic Architecture



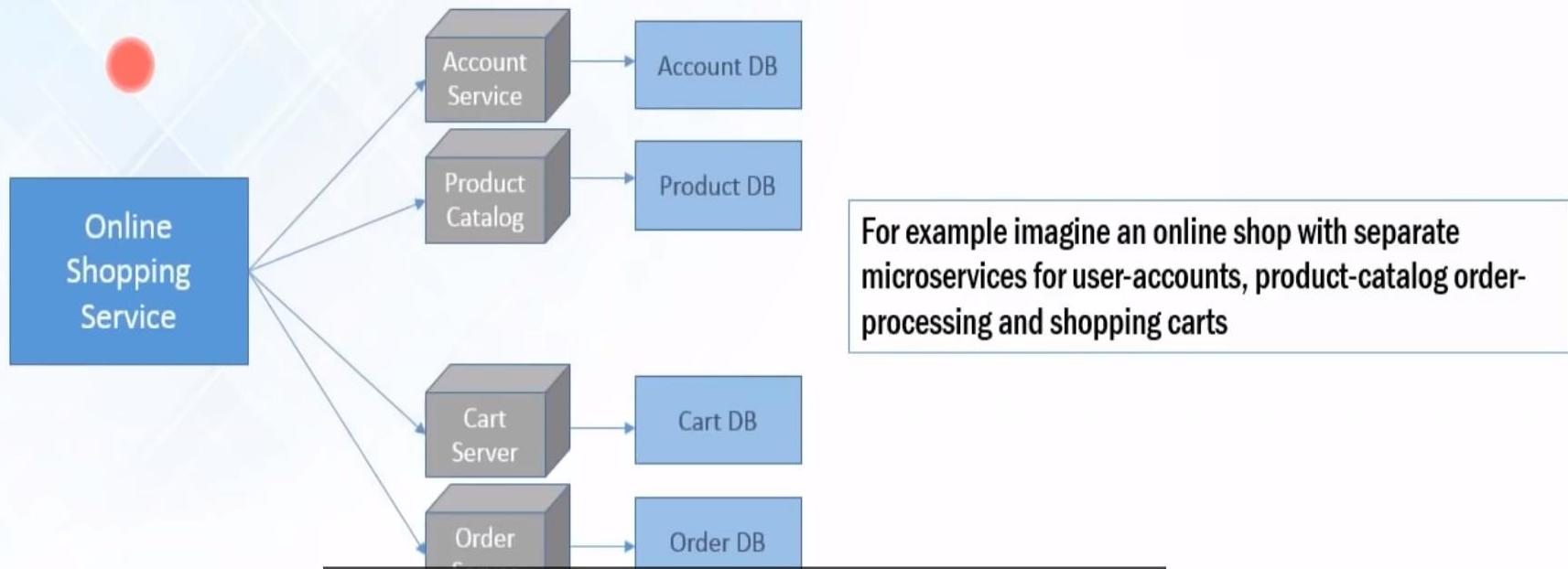
So it becomes really expensive
and time-consuming task.

Monolithic vs. Micro Services Example



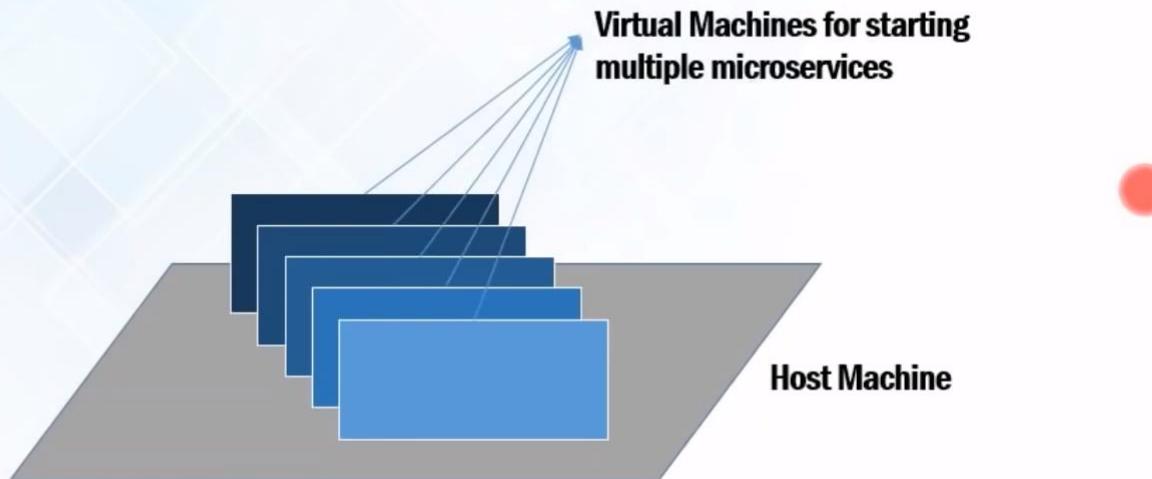
Problems Before Docker

The idea behind microservices is that some types of applications become easier to build and maintain when they are broken down into smaller, composable pieces which work together. Each component is developed separately, and the application is then simply the sum of its constituent components.



Problems Before Docker

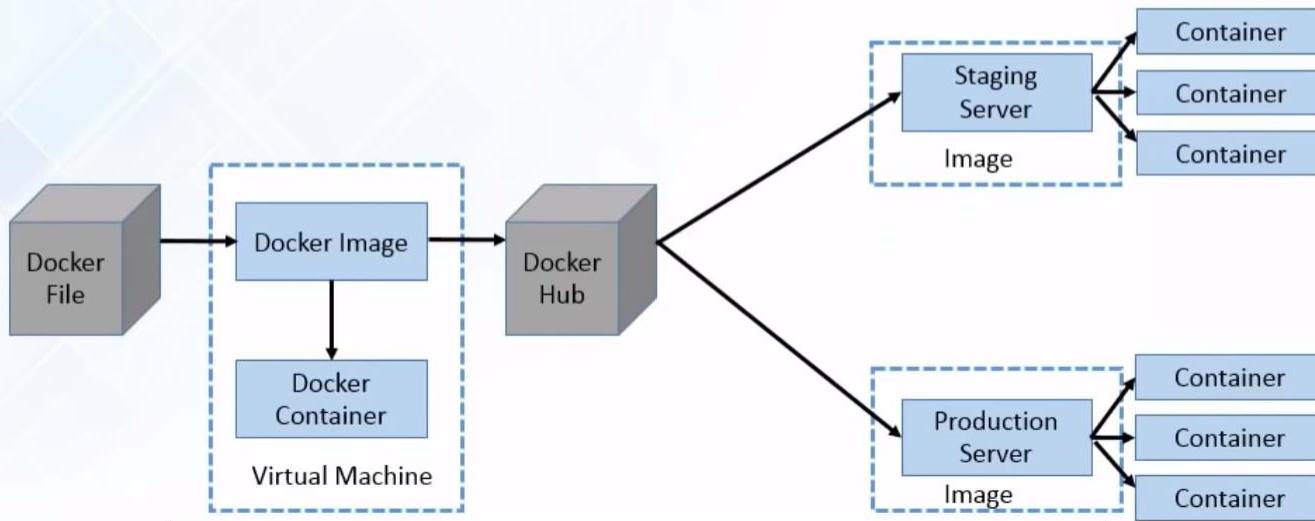
Developing an application requires starting several of microservices in one machine. So if you are starting five of those services you require five VMs on that machine.



Docker in a Nutshell

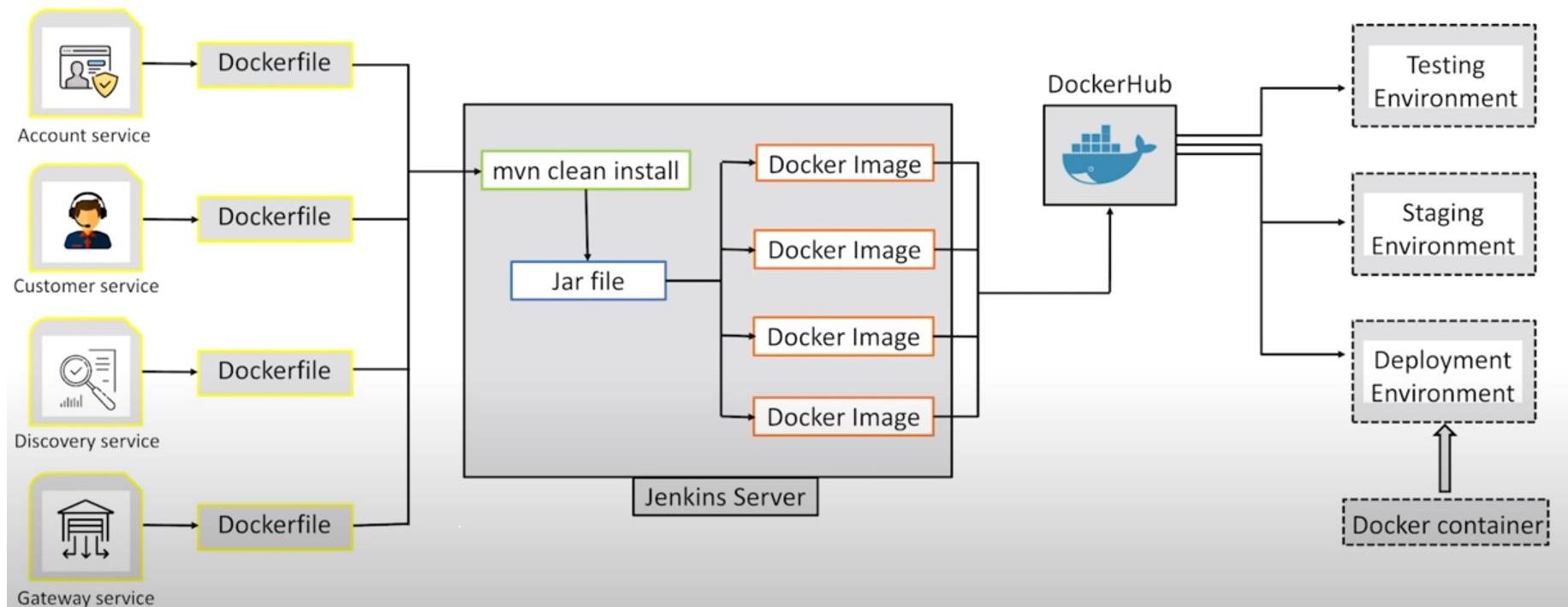


- Docker file builds a Docker image and that image contains all the project's code
- You can run that image to create as many Docker containers as you want
- Then this Image can be uploaded on Docker hub, from Docker hub any one can pull the image and build a container



Continuous delivery through DevOps Integration

Solution



Docker jenkins

- docker jenkins
- docker pull jenkins/jenkins
- docker run -p 8080:8080 --name=jenkins-master jenkins/jenkins
- docker stop jenkins-master
- docker rm jenkins-master
- docker run -p 8080:8080 --name=jenkins-master -d --env JAVA_OPTS="-Xmx8192m" --env JENKINS_OPTS="--handlerCountMax=300" jenkins/jenkins

Docker jenkins

```
C:\WINDOWS\system32>docker run -p 8080:8080 --name=jenkins-master jenkins/jenkins
docker: Error response from daemon: Conflict. The container name "/jenkins-master" is already in use by container "f43c2222247e094a75a6aca8b1a9a4544baff7145dc299d49ca0d6f2e55361ef5". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.

C:\WINDOWS\system32>docker container ls -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
STATUS            NAMES
f43c2222247e      jenkins/jenkins      "/sbin/tini -- /usr..."   18 minutes ago    Up 18 minutes      0.0.0.0:8080->8080/tcp, 50000/tcp   jenkins-master
e6f0ed404d77      barathece91/spring-cloud-sidecar-polygot-wstore-node-service:v2.3.0   "docker-entrypoint.s..."  27 hours ago     Exited (0) 26 hours ago      wsstore-service
54dee9e0e76b      barathece91/spring-cloud-sidecar-polygot-zuul-proxy:v2.3.0      "java -jar app.jar"    27 hours ago     Exited (143) 26 hours ago     zuul-proxy
a85871df1cb4      barathece91/spring-cloud-sidecar-polygot-sidecar:v2.3.0      "java -jar app.jar"    27 hours ago     Exited (143) 26 hours ago     sidecar
fa7736867ee7      barathece91/spring-cloud-sidecar-polygot-csstore-microservice:v2.3.0  "java -jar app.jar"    27 hours ago     Exited (143) 26 hours ago     csstore-service
5490e027ce4d      barathece91/spring-cloud-sidecar-polygot-msstore-microservice:v2.3.0  "java -jar app.jar"    27 hours ago     Exited (143) 26 hours ago     msstore-service
cd4d202cdf59      barathece91/spring-cloud-sidecar-polygot-eureka-server:v2.3.0      "java -jar app.jar"    27 hours ago     Exited (143) 26 hours ago     eureka-server
5b129b2c9196      barathece91/spring-cloud-sidecar-polygot-config-server:v2.3.0      "java -jar app.jar"    27 hours ago     Exited (143) 26 hours ago     config-server
e3e7a9df79a4      wurstmeister/zookeeper          "/bin/sh -c '/usr/sb..."  9 days ago       Exited (137) 8 days ago
55265d9dbc70      wurstmeister/kafka           "start-kafka.sh"       9 days ago       Exited (1) 8 days ago       kafka-spring-boot-example-master_zookeeper_1
                                            kafka-spring-boot-example-master_kafka_1

C:\WINDOWS\system32>docker container start f43c2222247e
f43c2222247e
```

Jenkins using war

```
D:\Program Files\Jenkins>java -jar jenkins.war
Running from: D:\Program Files\Jenkins\Jenkins.war
webroot: $user.home/.jenkins
2020-11-08 13:02:57.113+0000 [id=1]      INFO  org.eclipse.jetty.util.log.Log#initialized: Logging initialized @1481ms to org.eclipse.jetty.util.log.JavaUtilLog
2020-11-08 13:02:57.419+0000 [id=1]      INFO  winstone.Logger#logInternal: Beginning extraction from war file
2020-11-08 13:03:05.177+0000 [id=1]      WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2020-11-08 13:03:05.306+0000 [id=1]      INFO  org.eclipse.jetty.server.Server#doStart: jetty-9.4.30.v20200611; built: 2020-06-11T12:34:51.929Z; git: 271836e4c1f4612f12b7bb13ef5a92a927634b0d; jvm 1.8.0_261-b12
2020-11-08 13:03:06.749+0000 [id=1]      INFO  o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find org.eclipse.jetty.jsp.JettyJspServlet
2020-11-08 13:03:06.845+0000 [id=1]      INFO  o.e.j.s.s.DefaultSessionIdManager#doStart: DefaultSessionIdManager workerName=node0
2020-11-08 13:03:06.846+0000 [id=1]      INFO  o.e.j.s.s.DefaultSessionIdManager#doStart: No SessionScavenger set, using defaults
2020-11-08 13:03:06.859+0000 [id=1]      INFO  o.e.j.server.session.HouseKeeper#startScavenging: node0 Scavenging every 600000ms
2020-11-08 13:03:07.499+0000 [id=1]      INFO  hudson.WebAppMain#contextInitialized: Jenkins home directory: C:\Users\Balasubramaniam\.jenkins found at: $user.home/.jenkins
2020-11-08 13:03:07.943+0000 [id=1]      INFO  o.e.j.s.handler.ContextHandler#doStart: Started w.@2c5d601e{Jenkins v2.249.1,/,file:///C:/Users/Balasubramaniam/.jenkins/war/,AVAILABLE}{C:\Users\Balasubramaniam\.jenkins\war}
2020-11-08 13:03:08.090+0000 [id=1]      INFO  o.e.j.server.AbstractConnector#doStart: Started ServerConnector@3e0e1046{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
2020-11-08 13:03:08.091+0000 [id=1]      INFO  org.eclipse.jetty.server.Server#doStart: Started @12462ms
2020-11-08 13:03:08.099+0000 [id=23]     INFO  winstone.Logger#logInternal: Winstone Servlet Engine running: controlPort=disabled
2020-11-08 13:03:10.092+0000 [id=29]     INFO  jenkins.InitReactorRunner$1#onAttained: Started initialization
2020-11-08 13:04:02.247+0000 [id=28]     INFO  jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2020-11-08 13:04:15.389+0000 [id=37]     INFO  jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
2020-11-08 13:04:15.404+0000 [id=32]     INFO  jenkins.InitReactorRunner$1#onAttained: Started all plugins
2020-11-08 13:04:19.082+0000 [id=40]     INFO  jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2020-11-08 13:04:19.155+0000 [id=43]     INFO  jenkins.InitReactorRunner$1#onAttained: System config loaded
2020-11-08 13:04:19.158+0000 [id=43]     INFO  jenkins.InitReactorRunner$1#onAttained: System config adapted
2020-11-08 13:04:19.698+0000 [id=42]     INFO  jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2020-11-08 13:04:19.699+0000 [id=32]     INFO  jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2020-11-08 13:04:19.791+0000 [id=62]     INFO  hudson.model.AsyncPeriodicWork$lambda$doRun$0: Started Download metadata
```

Large Repos are unhealthy

- Large git repos (2 GB+) often are a sign of git misuse (large binaries, etc.)
- Yes, there are cases where a large git repo is not a sign of misuse
 - Linux kernel repo is over 1 GB
 - Linux kernel repo is one of the oldest git repos
- ○ Git is able to handle large repos, some operations are slower on some systems
- Git misuse doesn't matter - we need to run our Jenkins jobs anyway
 - The 18 GB+ repository at my work isn't pretty, but it is business critical...

Jenkins tricks for large repos

- Use command line git rather than JGit
 - Command line git explicitly manages memory and handles large repos
 - JGit large repo support has improved, will likely always lag behind command line git
- Reduce data transfer & disc space for history
 - ○ Reference repositories - only clone what's new
 - ○ Shallow clone - only clone recent history
 - ○ Don't fetch tags
 - ○ Narrow refs - only clone specific branches
- Reduce disc space for working directory
 - ○ Sparse checkout

Jenkins free style project sparse checkout

Only specific project from repo

The screenshot shows the Jenkins Source Code Management configuration page for a free-style project. A blue arrow points from the text "Only specific project from repo" to the "Sparse Checkout paths" section.

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

Branch Specifier (blank for 'any') x ?
*/main Add Branch

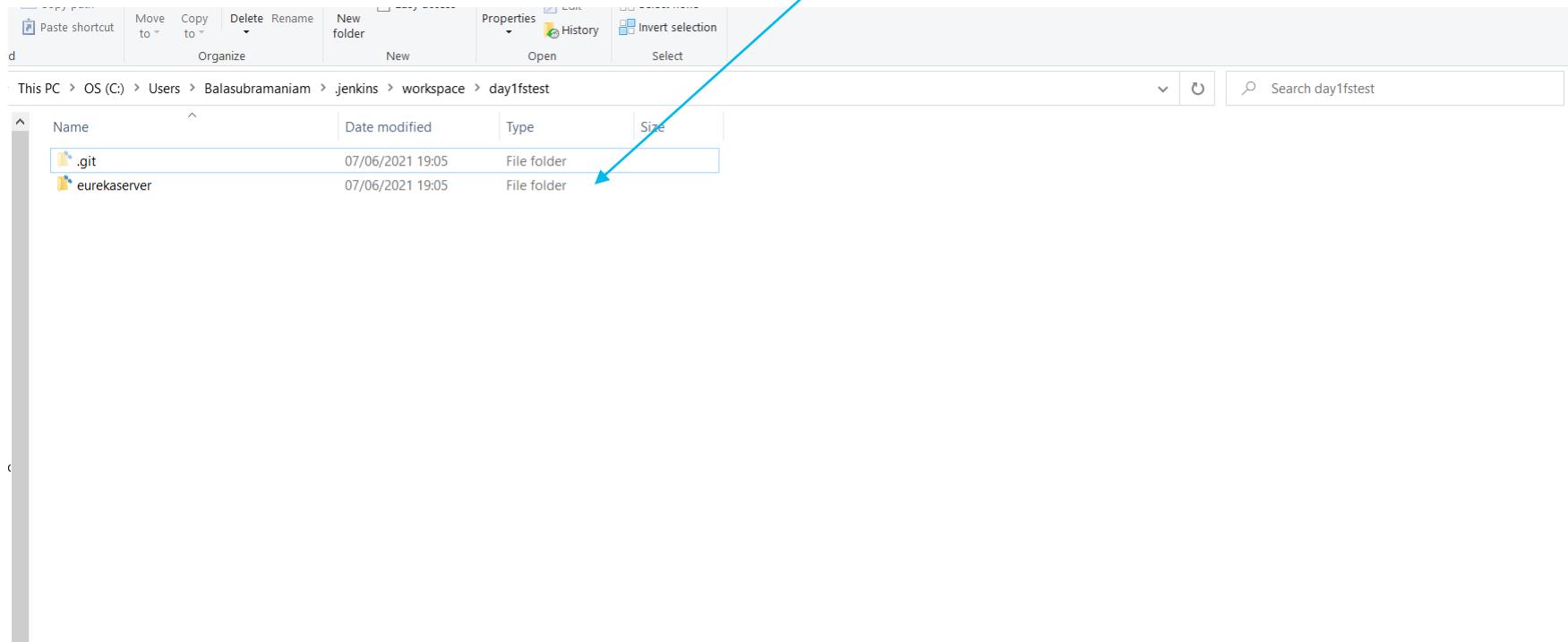
Repository browser ? ▾
(Auto)

Additional Behaviours

Sparse Checkout paths x ?
Path x
/eurekaserver Add

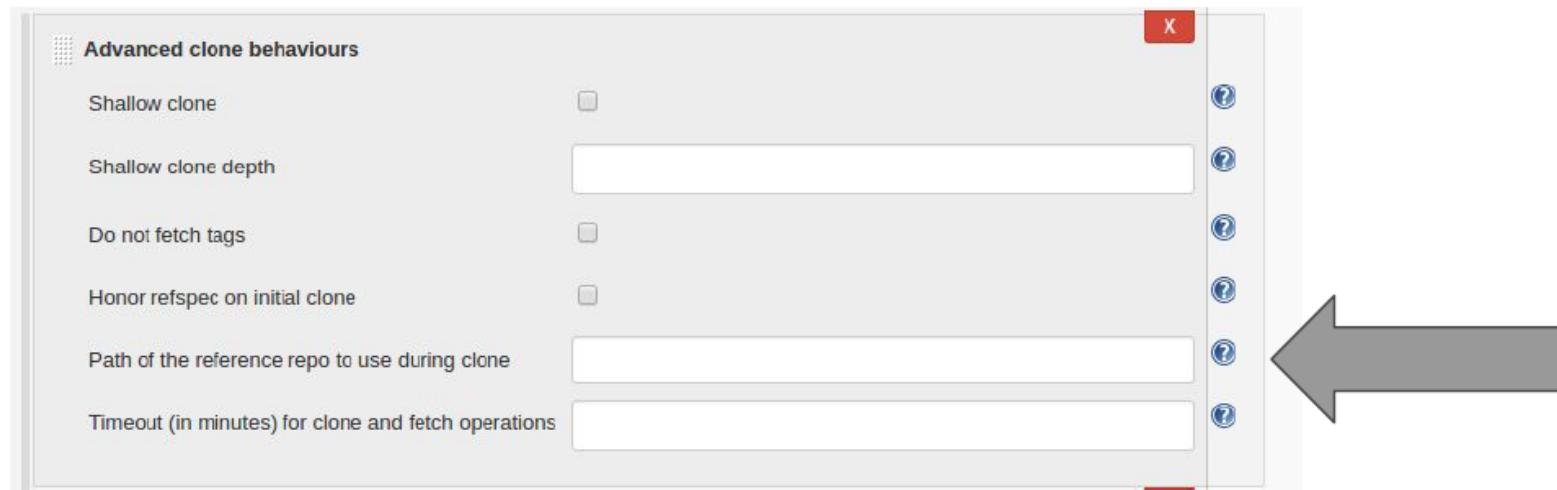
Jenkins free style project sparse checkout

Only specific project from repo



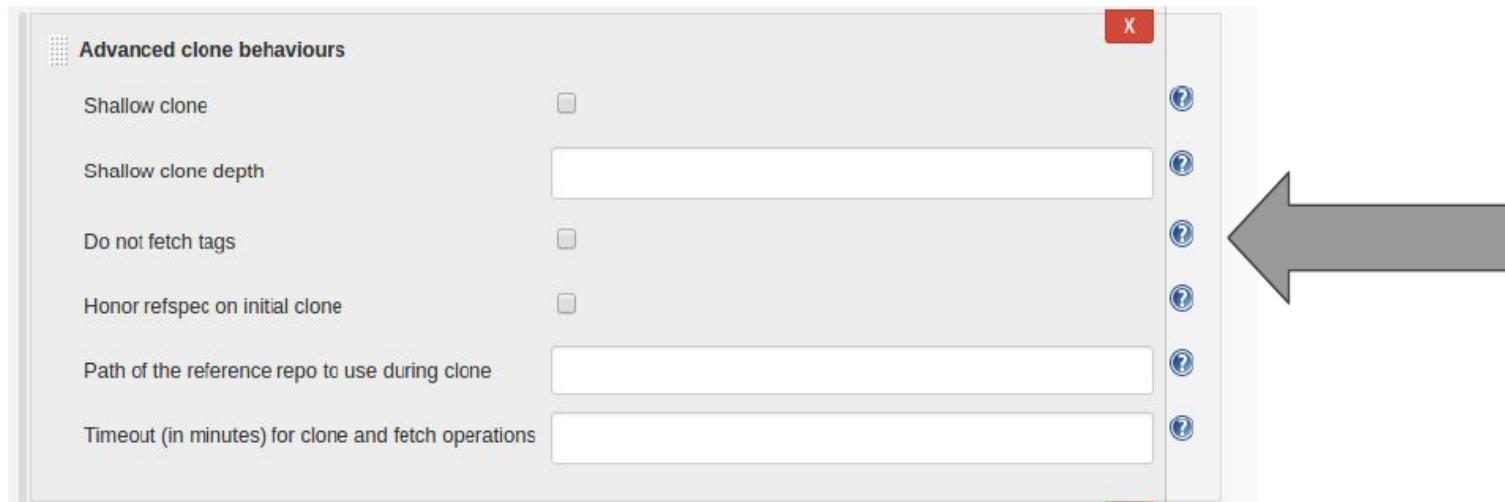
Reference repository - pointers, not copies

- A bare git repository can be used as a “reference repository” for other repositories
- Git creates pointers to the reference repo
 - Saves space by not copying data to disc
 - Saves time by not transferring over the wire



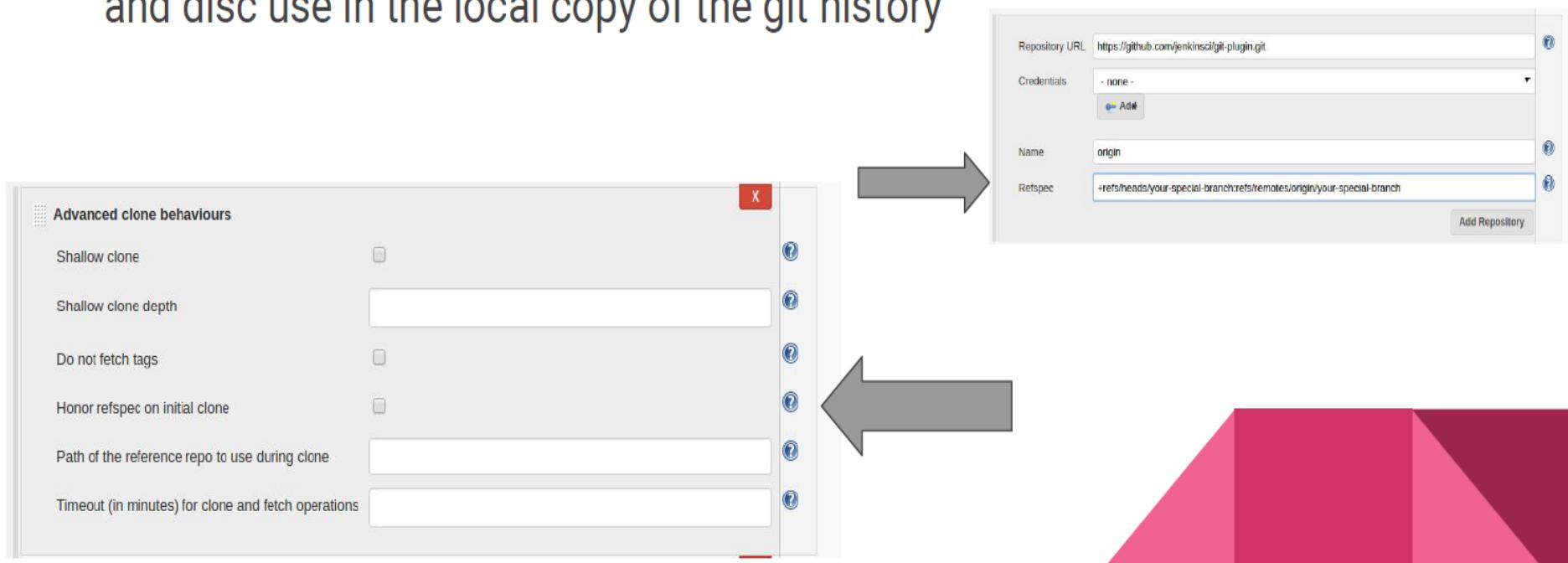
Don't Fetch Tags

- Most Jenkins jobs don't need tags
- Skipping tags can reduce data transfer and local disc use



Narrow refsspecs - clone only what you need

- Git refspec describes what to copy from remote
- Reducing the refspec can reduce the data transfer and disc use in the local copy of the git history



Jenkins Freestyle Project for Docker Container Images with SonarQube

SonarQube Scanner

Name
sonarqube

Install automatically [?](#)

Install from Maven Central

Version
SonarQube Scanner 4.6.2.2472 ▾ [Delete Installer](#)

Install from Maven Central

Version
SonarQube Scanner 4.6.2.2472 ▾ [Delete Installer](#)

[Add Installer ▾](#) [Delete SonarQube Scanner](#)

Add SonarQube Scanner

[Save](#) [Apply](#)

Jenkins Freestyle Project for Docker Container Images with SonarQube

The screenshot shows the configuration page for a Jenkins Freestyle Project. The project is titled "Execute SonarQube Scanner".

Task to run: (empty field)

JDK: java8

Path to project properties: (empty field)

Analysis properties:

```
sonar.projectKey=uploaddocker
sonar.projectName=C:/Users/Balasubramaniam/jenkins/workspace/day1fsdockercproject/uploaddocker
sonar.projectVersion=1.0
sonar.login=admin
sonar.password=admin
sonar.language=java
sonar.sources=C:/Users/Balasubramaniam/jenkins/workspace/$JOB_NAME
sonar.java.binaries=C:/Users/Balasubramaniam/jenkins/workspace/$JOB_NAME/uploaddocker/target/classes
sonar.sourceEncoding=UTF-8
```

Buttons at the bottom:

- Save
- Apply

Jenkins Freestyle Project for Docker Container Images with SonarQube

The screenshot shows the SonarQube web interface with the following details:

- Header:** http://localhost:9000/projects?sort=-analysis_date
- Top Navigation:** sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, Search bar, Log in.
- Filters:** Perspective: Overall Status, Sort by: Last analysis date, Search by project name or key, 4 projects.
- Left Sidebar (Filters):**
 - Quality Gate:** Passed (4), Warning (0), Failed (0).
 - Reliability (Bug count):** A (3), B and worse (1), C and worse (1), D and worse (0), E (0).
 - Security (Vulnerability count):** A (4), B and worse (0), C and worse (0), D and worse (0), E (0).
 - Maintainability (Code Smell count):** A (4), B and worse (0), C and worse (0), D and worse (0), E (0).
- Project Analysis Cards:**
 - C:/Users/Balasubramaniam/jenkins/workspace/day1fsdockerproject/uploaddocker**: Passed. Last analysis: June 7, 2021, 9:57 PM. Issues: 0 Bugs, 0 Vulnerabilities, 16 Code Smells. Coverage: 0.0%. Duplications: 0.0%. Languages: Java. Issues count: 573.
 - C:/Users/Balasubramaniam/jenkins/workspace/NetabankingMavenProject**: Passed. Last analysis: June 6, 2021, 7:57 PM. Issues: 0 Bugs, 0 Vulnerabilities, 1 Code Smell. Coverage: 0.0%. Duplications: 0.0%. Languages: Java. Issues count: 18.
 - netbanking**: Passed. Last analysis: September 30, 2020, 9:45 AM. Issues: 0 Bugs, 0 Vulnerabilities, 0 Code Smells. Coverage: 0.0%. Duplications: 0.0%. Languages: XML, Java. Issues count: 115.
 - day14springmvcjpasshopping**: Passed. Last analysis: October 3, 2019, 3:09 PM. Issues: 1 Bug (C), 0 Vulnerabilities, 50 Code Smells. Coverage: 0.0%. Duplications: 0.0%. Languages: Java, XML. Issues count: 867.

Jenkins

Dashboard [Jenkins] x +

localhost:8080

search

?

1

Parameswari log out

Jenkins

New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Lockable Resources

New View

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
		NetabankingMavenProject	1 mo 8 days - #14	1 mo 9 days - #11	52 sec

Icon: S M L

Legend Atom feed for all Atom feed for failures Atom feed for just latest builds

Build Queue ^

SonarQube

```
jvm 1 | 2020.09.30 09:04:16 INFO app[] [o.s.a.AppFileSystem] Cleaning or creating temp directory E:\software\A08\file\sonarqube-6.6\temp
jvm 1 | 2020.09.30 09:04:16 INFO app[] [o.s.a.es.EsSettings] Elasticsearch listening on /127.0.0.1:9001
jvm 1 | 2020.09.30 09:04:16 INFO app[] [o.s.a.p.ProcessLauncherImpl] Launch process[[key='es', ipcIndex=1, logFilenamePrefix=es]] from [E:\software\A08\file\sonarqube-6.6\elasticsearch]: C:\Program Files\Java\jdk1.8.0_152\jre\bin\java -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -XX:+AlwaysPreTouch -server -Xss1m -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djna.nosys=true -Djdk.io.permissionsUseCanonicalPath=true -Dio.netty.noUnsafe=true -Dio.netty.noKeySetOptimization=true -Dio.netty.recycler.maxCapacityPerThread=0 -Dlog4j.shutdownHookEnabled=false -Dlog4j2.disable.jmx=true -Dlog4j.skipJansi=true -Xms512m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError -Delasticsearch -Des.path.home=E:\software\A08\file\sonarqube-6.6\elasticsearch -cp lib/* org.elasticsearch.bootstrap.Elasticsearch -Epath.conf=E:\software\A08\file\sonarqube-6.6\temp\conf\es
jvm 1 | 2020.09.30 09:04:16 INFO app[] [o.s.a.SchedulerImpl] Waiting for Elasticsearch to be up and running
jvm 1 | 2020.09.30 09:04:17 INFO app[] [o.e.p.PluginsService] no modules loaded
jvm 1 | 2020.09.30 09:04:17 INFO app[] [o.e.p.PluginsService] loaded plugin [org.elasticsearch.transport.Netty4Plugin]
jvm 1 | 2020.09.30 09:04:37 INFO app[] [o.s.a.SchedulerImpl] Process[es] is up
jvm 1 | 2020.09.30 09:04:37 INFO app[] [o.s.a.p.ProcessLauncherImpl] Launch process[[key='web', ipcIndex=2, logFilenamePrefix=web]] from [E:\software\A08\file\sonarqube-6.6]: C:\Program Files\Java\jdk1.8.0_152\jre\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=E:\software\A08\file\sonarqube-6.6\temp -Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError -cp ./lib/common/*;./lib/server/*;E:\software\A08\file\sonarqube-6.6\lib\jdbc\h2\h2-1.3.176.jar org.sonar.server.app.WebServer E:\software\A08\file\sonarqube-6.6\temp\sq-process7163985333166906569properties
jvm 1 | 2020.09.30 09:04:59 INFO app[] [o.s.a.SchedulerImpl] Process[web] is up
jvm 1 | 2020.09.30 09:04:59 INFO app[] [o.s.a.p.ProcessLauncherImpl] Launch process[[key='ce', ipcIndex=3, logFilenamePrefix=ce]] from [E:\software\A08\file\sonarqube-6.6]: C:\Program Files\Java\jdk1.8.0_152\jre\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=E:\software\A08\file\sonarqube-6.6\temp -Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError -cp ./lib/common/*;./lib/server/*;./lib/ce/*;E:\software\A08\file\sonarqube-6.6\lib\jdbc\h2\h2-1.3.176.jar org.sonar.ce.app.CeServer E:\software\A08\file\sonarqube-6.6\temp\sq-process2633312515576990903properties
jvm 1 | 2020.09.30 09:05:06 INFO app[] [o.s.a.SchedulerImpl] Process[ce] is up
jvm 1 | 2020.09.30 09:05:06 INFO app[] [o.s.a.SchedulerImpl] SonarQube is up
```

SonarQube Maven

sonar:sonar -
Dsonar.host.url=http://localhost:9000 -
Dsonar.login=b56042da35208eb303feb4b1ab
afe76f9ae572ac

hsbcfresher2020 - netbanking/pom.xml - Spring Tool Suite 4

File Edit Navigate Search Project Run Design Window Help

Project Explorer Overview Dependencies Dependency Hierarchy Effective POM pom.xml

netbanking/pom.xml

```
<!-- activation -->
<properties>
    <db.driverClassName>oracle.jdbc.driver.OracleDriver</db.driverClassName>
    <db.url>jdbc:oracle:thin:@localhost:1521:XE</db.url>
    <db.username>system</db.username>
    <db.password>vignesh</db.password>
```

Markers Properties Servers Data Source Explorer Snippets Console Git Repositories Git Staging

<terminated> netbanking (2) [Maven Build] C:\Program Files\Java\jdk1.8.0_152\bin\javaw.exe (30-Sep-2020, 9:40:15 am)

```
[INFO] Sensor JaCoCoSensor [java]
[INFO] Sensor JaCoCoSensor [java] (done) | time=1ms
[INFO] Sensor SonarJavaXmlFileSensor [java]
[INFO] 1 source files to be analyzed
[INFO] Sensor SonarJavaXmlFileSensor [java] (done) | time=1539ms
[INFO] 1/1 source files have been analyzed
[INFO] Sensor XML Sensor [xml]
[INFO] Sensor XML Sensor [xml] (done) | time=1687ms
[INFO] Sensor Analyzer for "php.ini" files [php]
[INFO] Sensor Analyzer for "php.ini" files [php] (done) | time=4ms
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=19ms
[INFO] Sensor CPD Block Indexer
[INFO] Sensor CPD Block Indexer (done) | time=17ms
[INFO] SCM provider for this project is: git
[INFO] 3 files to be analyzed
[INFO] 2/3 files analyzed
[WARNING] Missing blame information for the following files:
[WARNING] * pom.xml
[WARNING] This may lead to missing/broken features in SonarQube
[INFO] 1 file had no CPD blocks
[INFO] Calculating CPD for 0 files
[INFO] CPD calculation finished
```

522M of 919M

Type here to search

09:40 30/09/2020

SonarQube

Jenkins Installation In Windows | Jenkins download and deployment | NetabankingMavenProject [Jenkins] | Integrating SonarQube and Jenkins | Projects

localhost:9000/projects?sort=-analysis_date

Projects Issues Rules Quality Profiles Quality Gates Search for projects, sub-projects and files... Log in

sonarqube Projects Issues Rules Quality Profiles Quality Gates Search for projects, sub-projects and files... Log in

Perspective: Overall Status Sort by: Last analysis date 1 projects

Filters

Quality Gate

Status	Count
Passed	1
Warning	0
Failed	0

Last analysis: October 3, 2019, 3:09 PM

day14springmvcjpasshopping Passed

1 C Bugs 0 A Vulnerabilities 50 A Code Smells 0.0% Coverage 0.0% Duplications

XS 867 Java, XML

1 of 1 shown

Reliability (Bug)

Grade	Count
A	0
B and worse	1
C and worse	1
D and worse	0
E	0

Security (Vulnerabilities)

Grade	Count
A	1
B and worse	0
C and worse	0
D and worse	0
E	0

Maintainability (Code Smells)

Grade	Count
A	1
B and worse	0

Embedded database should be used for evaluation purpose only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA
Version 6.6 (build 32724) - [LGPL v3](#) - [Community](#) - [Documentation](#) - [Get Support](#) - [Plugins](#) - [Web API](#) - [About](#)

Type here to search

9:06 30/09/2020

SonarQube Properties

- sonar.projectKey=netbanking
- sonar.projectName=C:/Users/Balasubramaniam/.jenkins/workspace/NetbankingMavenProject
- sonar.projectVersion=1.0
- sonar.login=admin
- sonar.password=admin
- sonar.exclusions=vendor/**, storage/**, resources/**
- sonar.language=java
- sonar.sources=C:/Users/Balasubramaniam/.jenkins/workspace/\$JOB_NAME
- sonar.java.binaries=C:/Users/Balasubramaniam/.jenkins/workspace/\$JOB_NAME/netbanking/target/classes
- sonar.sourceEncoding=UTF-8

Jenkins Build

localhost:8989/job/NetabankingMavenProject/1/console

```
[INFO]
[INFO] Recording test results
[INFO]
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ netbanking ---
[INFO] Building jar: C:\Users\Balasubramaniam\.jenkins\workspace\NetabankingMavenProject\Netbanking\target\netbanking-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.3.4.RELEASE:repackage (repackage) @ netbanking ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ netbanking ---
[INFO] Installing C:\Users\Balasubramaniam\.jenkins\workspace\NetabankingMavenProject\Netbanking\target\netbanking-0.0.1-SNAPSHOT.jar to C:\Users\Balasubramaniam\.m2\repository\com\hsbc\banking\netbanking\0.0.1-SNAPSHOT\netbanking-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\Users\Balasubramaniam\.jenkins\workspace\NetabankingMavenProject\Netbanking\pom.xml to C:\Users\Balasubramaniam\.m2\repository\com\hsbc\banking\netbanking\0.0.1-SNAPSHOT\netbanking-0.0.1-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  22.561 s
[INFO] Finished at: 2020-09-30T08:48:32+05:30
[INFO] -----
[WARNING] The requested profile "test" could not be activated because it does not exist.
Waiting for Jenkins to finish collecting data
[INFO] Archiving C:\Users\Balasubramaniam\.jenkins\workspace\NetabankingMavenProject\Netbanking\pom.xml to com.hsbc.bankng/netbanking/0.0.1-SNAPSHOT/netbanking-0.0.1-SNAPSHOT.pom
[INFO] Archiving C:\Users\Balasubramaniam\.jenkins\workspace\NetabankingMavenProject\Netbanking\target\netbanking-0.0.1-SNAPSHOT.jar to com.hsbc.bankng/netbanking/0.0.1-SNAPSHOT/netbanking-0.0.1-SNAPSHOT.jar
channel stopped
Finished: SUCCESS
```

Type here to search



08:49
ENG
30/09/2020
6

Role Based Strategy

Day1 project

Day 1 project

localhost:8989/role-strategy/manage-roles

Manage and Assign Roles

Role	Overall				Credentials					Agent									
	Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Provision	Build	Cancel	Configure	Create	
admin	<input checked="" type="checkbox"/>																		
developer	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
qa	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
test	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Role to add

Open Blue Ocean

Add

Item roles

Role	Pattern	Credentials					Job					Run		SCM	Loc					
		Create	Delete	ManageDomains	Update	View	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Tag	Res
test	"day1.*"	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>													

Role to add

Pattern

Role Based Strategy

Limited job access

The screenshot shows the Jenkins dashboard at localhost:8989. A blue arrow points from the text "Limited job access" to the "Name" column header of the main job list table.

Jenkins Dashboard

Jobs

All	S	W	Name	Last Success	Last Failure	Last Duration	Fav
	●	●	day1fsandroidproject	20 hr - #4	20 hr - #3	1 min 7 sec	
	●	●	day1fsdockerproject	8 hr 0 min - #21	22 hr - #19	53 sec	
	●	●	day1fstest	1 day 0 hr - #20	1 day 3 hr - #7	25 sec	

Icon: S M L

Legend: Atom feed for all Atom feed for failures Atom feed for just latest builds

Build Queue

No builds in the queue.

Build Executor Status

master
1 Idle
2 Idle

Jenkins Build log Regex

- Way 1 –
 - Put following lines as part of Default Content text box
 - \${BUILD_LOG, maxLines=9999, escapeHtml=false}
 - this works for free style projects
- Way 2 – If you want send only specific line using regex
 - \${BUILD_LOG_REGEX, regex="\b(Buildfile|BUILD)\b", linesAfter=1}
 - \${BUILD_LOG_REGEX, regex="^.+?BUILD FAILED.+?\$", linesBefore=0, linesAfter=10, maxMatches=5, showTruncatedLines=false, escapeHtml=true}
 - \${BUILD_LOG_REGEX, regex="\b(FC0)\b", linesAfter=1}
 - \${BUILD_LOG_REGEX, regex="^.+?BUILD FAILED.+?\$", linesBefore=0, linesAfter=10, maxMatches=5, showTruncatedLines=false, escapeHtml=true}
 - \${BUILD_LOG_REGEX, regex="^.+?FC0.+?\$", maxMatches=95, showTruncatedLines=false, escapeHtml=false, matchedLineHtmlStyle=true}
- This will show exactly which is expected lines as part of the regEx.

Jenkins Build log Regex

- List of other variables as part of email body
- \${PROJECT_URL}/ws/
- \${BUILD_URL}/artifact/
- \${BUILD_LOG, maxLines=1000}
- More ref
- Content Token Reference
- In string arguments, escape “, \, and line terminators (\n or \r\n) with a \, e.g. arg1="\"quoted\""; arg2="c:\\path"; and arg3="one\\ntwo". The brackets may be omitted if there are no arguments.
- New Lines – Use
 or “\”

Jenkins Build log regex

day1fstest ➔

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

Default Content ?

```
 ${BUILD_LOG, maxLines=1, escapeHtml=false}  
 ${BUILD_LOG_REGEX, regex="\b(FC0)\b", linesAfter=1}  
 ${BUILD_LOG_REGEX, regex="^.+?BUILD FAILED.*$?", linesBefore=0, linesAfter=10, maxMatches=5,  
 ${PROJECT_URL}/ws/  
 ${BUILD_URL}/artifact/  
 ${BUILD_LOG, maxLines=10}
```

Attachments ?

Can use wildcards like 'module/dist/**/*.zip'. See the [@includes of Ant fileset](#) for the exact format. The base directory is [the workspace](#).

Attach Build Log ?

Do Not Attach Build Log ▾

Content Token Reference ?

Advanced Settings...

Jenkins Build log regex

A screenshot of a Gmail inbox interface. The main message is titled "day1fstest - Build # 23 - Successful!" and is from "parameswaribala@gmail.com". The message body contains a truncated log file and several Jenkins log entries. At the bottom, there are "Reply" and "Forward" buttons.

Search mail ② ⚙️ ...

day1fstest - Build # 23 - Successful! Inbox ×

parameswaribala@gmail.com
to me, vebconsulting2021 ▾
[...truncated 5.09 KB...]

8:34 PM (3 minutes ago) ☆ ↶ ⋮

```
$[BUILD_LOG_REGEX, regex="\b(FC0)\b", linesAfter=1]
$[BUILD_LOG_REGEX, regex="^.*?BUILD FAILED.*?$", linesBefore=0, linesAfter=10, maxMatches=5,
http://localhost:8989/job/day1fstest/ws/
http://localhost:8989/job/day1fstest/23//artifact/
[...truncated 4.43 KB...]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 12.822 s
[INFO] Finished at: 2021-06-08T20:34:39+05:30
[INFO] -----
[WARNING] The requested profile "test" could not be activated because it does not exist.
Email was triggered for: Always
Sending email for trigger: Always
```

↶ Reply ➡ Forward

Android Jenkins Project

Administrator: Command Prompt - sdkmanager --update

Directory of C:\Users\Balasubramaniam\AppData\Local\Android\Sdk\tools\bin

```
05/08/2018  02:26 PM    <DIR>        .
05/08/2018  02:26 PM    <DIR>        ..
05/08/2018  02:26 PM            6,742 apk analyzer
05/08/2018  02:26 PM            2,227 archquery.bat
05/08/2018  02:26 PM            3,035 avdmanager.bat
05/08/2018  02:26 PM            2,215 jobb.bat
05/08/2018  02:26 PM            3,839 lint.bat
05/08/2018  02:26 PM            2,053 monkeyrunner.bat
05/08/2018  02:26 PM            3,042 sdkmanager.bat
05/08/2018  02:26 PM            2,189 uiautomatorviewer.bat
               8 File(s)       25,342 bytes
               2 Dir(s)  26,846,961,664 bytes free
```

C:\Users\Balasubramaniam\AppData\Local\Android\Sdk\tools\bin> sdkmanager --update
[] 3% Fetch remote repository...

Android home

Android Jenkins Project

Android home

The screenshot shows the Jenkins configuration page for an 'Android' job. The URL in the browser is `http://localhost:8989/configure`. The page has a header with tabs like 'Dashboard', 'configuration', and 'Advanced...'. A blue arrow points from the text 'Android home' at the top to the 'Android' section in the configuration. The 'Android' section contains fields for 'Android SDK root' (set to `C:\Users\Balasubramaniam\AppData\Local\Android\Sdk`) and checkboxes for 'Automatically install Android components when required' (checked) and 'Keep emulators in the job workspace, in the .android subdirectory, to isolate them as much as possible' (unchecked). At the bottom, there are 'Save' and 'Apply' buttons.

Test configuration by sending test e-mail

SCM Polling

Max # of concurrent polling
10

Android

Android SDK root
`C:\Users\Balasubramaniam\AppData\Local\Android\Sdk`

Enter the path to the root of an Android SDK installation

Automatically install Android components when required

Keep emulators in the job workspace, in the .android subdirectory, to isolate them as much as possible

Cloud

The cloud configuration has moved to a [separate configuration page](#).

Save **Apply**

Before Jenkins Pipeline

Over the years, there have been multiple Jenkins pipeline releases including, **Jenkins Build flow**, **Jenkins Build Pipeline plugin**, **Jenkins Workflow**, etc.

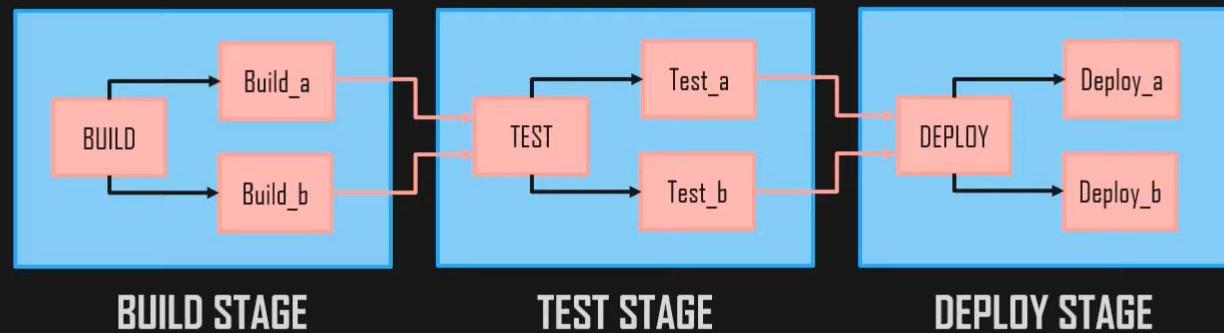


What are the key features of these plugins?

- Represent multiple Jenkins jobs as one pipeline
- What do these pipelines do?

These pipelines are a collection of Jenkins jobs which trigger each other in a specified sequence

Jenkins Pipeline

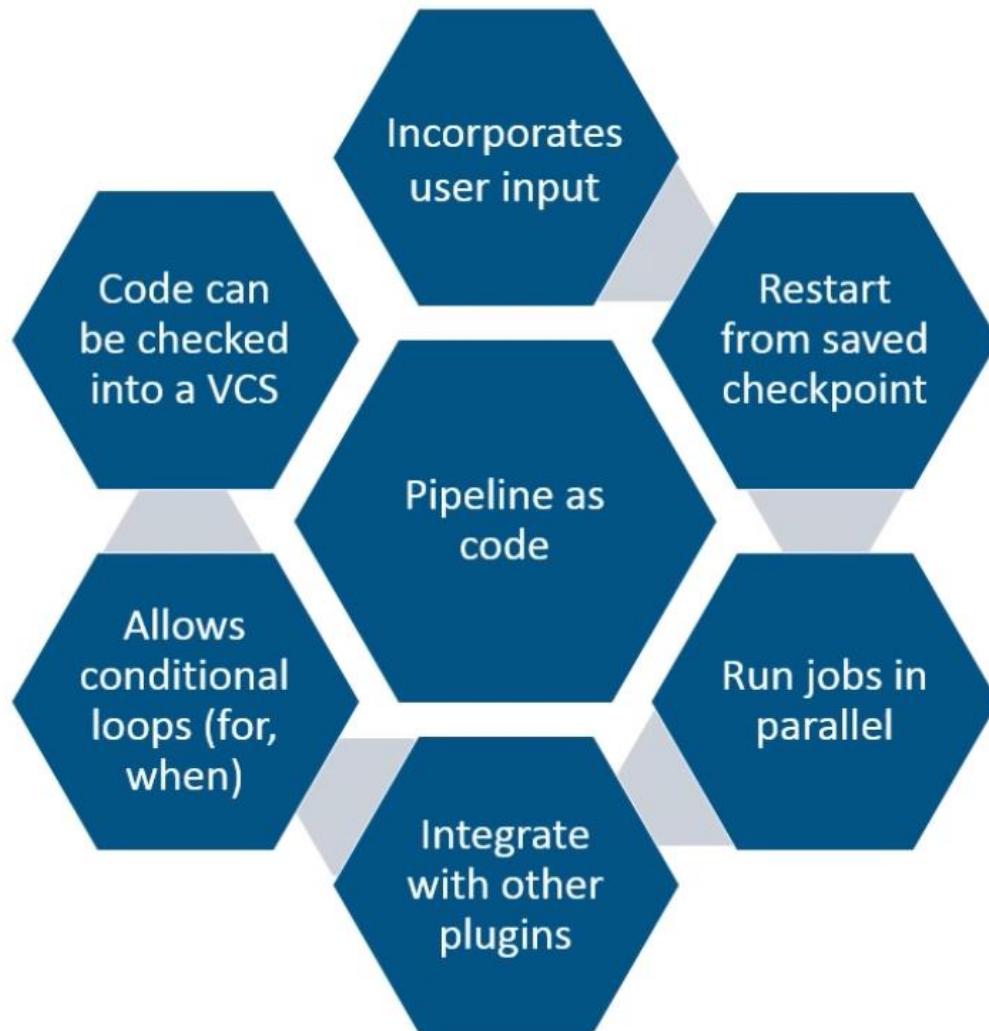


Jenkins Pipeline

- Jenkins pipeline is a single platform that runs the entire *pipeline as code*
- All the standard jobs defined by Jenkins are manually written in one script and they can be stored in a VCS
- Instead of building several jobs for each phase, you can now code the entire workflow and put it in a Jenkinsfile



Jenkins Pipeline key Features



Jenkins File

A text file that stores the pipeline as code

It can be checked into a SCM on your local system

Enables the developers to access, edit and check the code at all times

It is written using the Groovy DSL

Written based on two syntaxes

Jenkins Pipeline

```
// Example of Jenkins pipeline script

pipeline {
    stages {
        stage("Build") {
            steps {
                // Just print a Hello, Pipeline to the console
                echo "Hello, Pipeline!"
                // Compile a Java file. This requires JDKconfiguration from Jenkins
                javac HelloWorld.java
                // Execute the compiled Java binary called HelloWorld. This requires JDK configuration from Jenkins
                java HelloWorld
                // Executes the Apache Maven commands, clean then package. This requires Apache Maven configuration from
                Jenkins
                mvn clean package ./HelloPackage
                // List the files in current directory path by executing a default shell command
                sh "ls -ltr"
            }
        }
        // And next stages if you want to define further...
    } // End of stages
} // End of pipeline
```

then we're using me one as the build

Jenkins Pipeline

- Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins.
- A continuous delivery (CD) pipeline is an automated expression of your process for getting software from version control right through to your users and customers.
- Every change to your software (committed in source control) goes through a complex process on its way to being released.
- This process involves building the software in a reliable and repeatable manner, as well as progressing the built software (called a "build") through multiple stages of testing and deployment.
- Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the Pipeline domain-specific language (DSL) syntax.

Jenkins Pipeline

- The definition of a Jenkins Pipeline is written into a text file (called a `Jenkinsfile`) which in turn can be committed to a project's source control repository.
- This is the foundation of "Pipeline-as-code"; treating the CD pipeline a part of the application to be versioned and reviewed like any other code.

Jenkins Pipeline

- Creating a Jenkinsfile and committing it to source control provides a number of immediate benefits:
 - Automatically creates a Pipeline build process for all branches and pull requests.
 - Code review/iteration on the Pipeline (along with the remaining source code).
 - Audit trail for the Pipeline.
 - Single source of truth for the Pipeline, which can be viewed and edited by multiple members of the project.

Declarative versus Scripted Pipeline syntax

- Declarative:
 - Declarative pipeline syntax offers an easy way to create pipelines.
 - It contains a predefined hierarchy to create Jenkins pipelines.
 - It gives you the ability to control all aspects of a pipeline execution in a simple, straight-forward manner.

Declarative versus Scripted Pipeline syntax

- Scripted:
 - Scripted Jenkins pipeline runs on the Jenkins master with the help of a lightweight executor.
 - It uses very few resources to translate the pipeline into atomic commands.
 - Both declarative and scripted syntax are different from each other and are defined totally differently.

Declarative versus Scripted Pipeline syntax

SCRIPTED PIPELINE

- ✓ Strict & Traditional Syntax
- ✓ Code is written on a Jenkins UI Instance
- ✓ Code is defined within a "Node Block"

DECLARATIVE PIPELINE

- ✓ Simpler Groovy Syntax
- ✓ Code is written into a file & checked into SCM
- ✓ Code is defined within a "Pipeline Block"

jenkins file from a virtual constant

Declarative versus Scripted Pipeline syntax

- A Jenkinsfile can be written using two types of syntax - Declarative and Scripted.
- Declarative and Scripted Pipelines are constructed fundamentally differently.
- Declarative Pipeline is a more recent feature of Jenkins Pipeline which:
 - provides richer syntactical features over Scripted Pipeline syntax, and
 - is designed to make writing and reading Pipeline code easier.

Declarative Syntax

- A valid Declarative pipeline must be defined with the “pipeline” sentence and include the next required sections:
 - agent
 - stages
 - stage
 - steps

Declarative Syntax

- Also, these are the available directives:
 - environment (Defined at stage or pipeline level)
 - input (Defined at stage level)
 - options (Defined at stage or pipeline level)
 - parallel
 - parameters
 - post
 - script
 - tools
 - triggers
 - when

Declarative Syntax

- Agents should be labeled so they can be easily identified from each other. For example, nodes can be labeled by their platform (Linux, Windows, etc), by their versions or by their location, among others.
- The “agent” section configures on which nodes the pipeline can be run. Specifying “agent any” means that Jenkins will run the job on any of the available nodes.

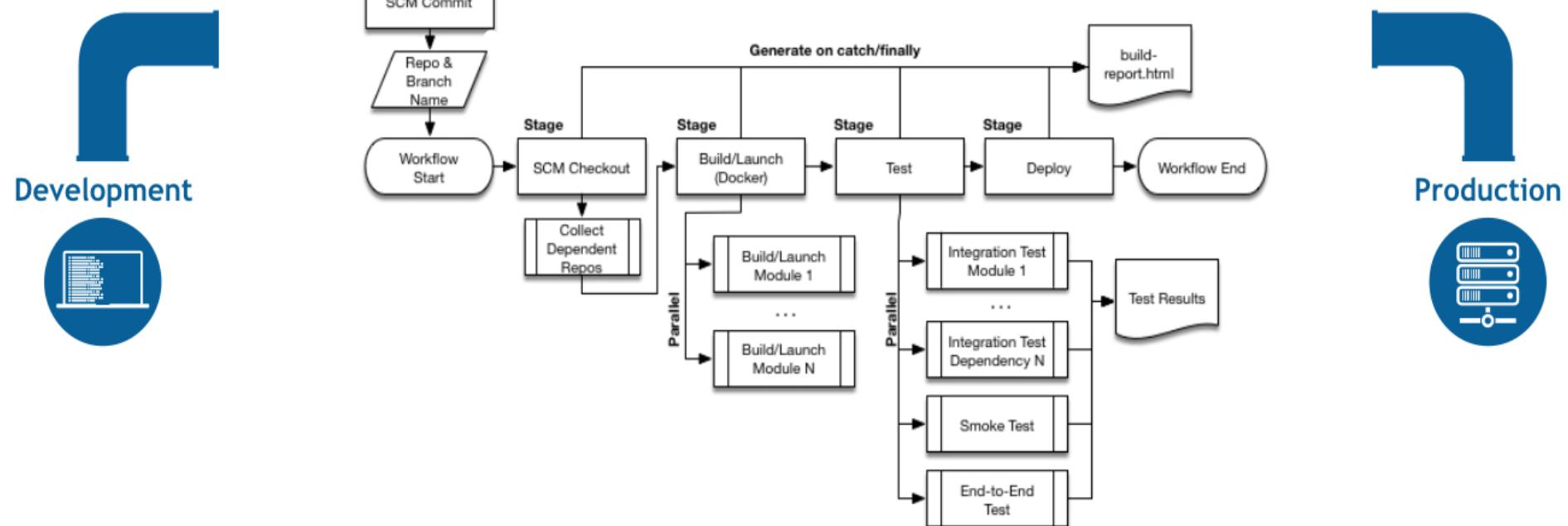
Why Pipeline?

- Jenkins is, fundamentally, an automation engine which supports a number of automation patterns.
- Pipeline adds a powerful set of automation tools onto Jenkins, supporting use cases that span from simple continuous integration to comprehensive CD pipelines.
- By modeling a series of related tasks, users can take advantage of the many features of Pipeline:

Why Pipeline?

- **Code:** Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
- **Durable:** Pipelines can survive both planned and unplanned restarts of the Jenkins controller.
- **Pausable:** Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
- **Versatile:** Pipelines support complex real-world CD requirements, including the ability to fork/join, loop, and perform work in parallel.
- **Extensible:** The Pipeline plugin supports custom extensions to its DSL footnote:dsl:[] and multiple options for integration with other plugins.

Why Pipeline?



Pipeline Concepts

Agent: instructs Jenkins to allocate an executor for the builds. It is defined for an entire pipeline or a specific stage.

It has the following parameters:

- *Any*: Runs pipeline/ stage on any available agent
- *None*: applied at the root of the pipeline, it indicates that there is no global agent for the entire pipeline & each stage must specify its own agent
- *Label*: Executes the pipeline/stage on the labelled agent.
- *Docker*: Uses docker container as an execution environment for the pipeline or a specific stage.

```
pipeline {  
    agent {  
        docker {  
            image 'ubuntu'  
        }  
    }  
}
```

Pipeline Concepts

Stages: It contains all the work, each stage performs a specific task.

```
pipeline {  
    agent any  
    stages {  
        stage ('Build') {  
            ...  
        }  
        stage ('Test') {  
            ...  
        }  
        stage ('QA') {  
            ...  
        }  
        stage ('Deploy') {  
            ...  
        }  
        stage ('Monitor') {  
            ...  
        }  
    }  
}
```

Steps: steps are carried out in sequence to execute a stage

```
pipeline {  
    agent any  
    stages {  
        stage ('Build') {  
            steps {  
                echo 'Running build phase...'  
            }  
        }  
    }  
}
```

Why Pipeline?

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
            }
        }
    }
}
```

Why Pipeline?

- 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.

Why Pipeline?

- In Scripted Pipeline syntax, one or more node blocks do the core work throughout the entire Pipeline.
- Although this is not a mandatory requirement of Scripted Pipeline syntax, confining your Pipeline's work inside of a node block does two things:
- Schedules the steps contained within the block to run by adding an item to the Jenkins queue.
- As soon as an executor is free on a node, the steps will run.
- Creates a workspace (a directory specific to that particular Pipeline) where work can be done on files checked out from source control.
- Caution: Depending on your Jenkins configuration, some workspaces may not get automatically cleaned up after a period of inactivity.

Why Pipeline?

Jenkinsfile (Scripted Pipeline)

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

- ➊ Execute this Pipeline or any of its stages, on any available agent.
- ➋ Defines the "Build" stage. `stage` blocks are optional in Scripted Pipeline syntax. However, implementing `stage` blocks in a Scripted Pipeline provides clearer visualization of each `stage's subset of tasks/steps in the Jenkins UI.
- ➌ Perform some steps related to the "Build" stage.
- ➍ Defines the "Test" stage.
- ➎ Perform some steps related to the "Test" stage.
- ➏ Defines the "Deploy" stage.
- ➐ Perform some steps related to the "Deploy" stage.

Questions



Module Summary

- In this module we discussed
 - Overview of Maven
 - Maven archetypes
 - Maven life cycle phases
 - The pom.xml file
 - Creation of Java projects using Maven
 - Creation of war files

