



DevOps series

# DevOps Development Deployment





Somkiat Puisungnoen

Search

Somkiat | Home

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามชานาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata

 DevOps

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

3

somkiat.cc

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc  
@somkiat.cc

Home Posts Videos Photos

Liked Following Share ...

+ Add a Button



# Section 1

1. Introduction to DevOps
2. DevOps workflow
3. DevOps tools
4. DevOps team topologies
5. Version Control System with Git
6. Manage containers with Docker
7. Docker image, container, Dockerfile, compose



# Section 2

1. Introduction to Kubernetes
2. K8s foundation (Pods, Deployment, Service)
3. K8s command with kubectl
4. Writing K8s manifest files
5. Introduction to Helm to manage K8s



# Section 3

1. Continuous Integration and Delivery (CI/CD)
2. CI/CD principles
3. Design your pipeline
4. Working with Jenkins
5. Create your pipeline
6. Pipeline as a Code
7. Scalable Jenkins with Master/slave



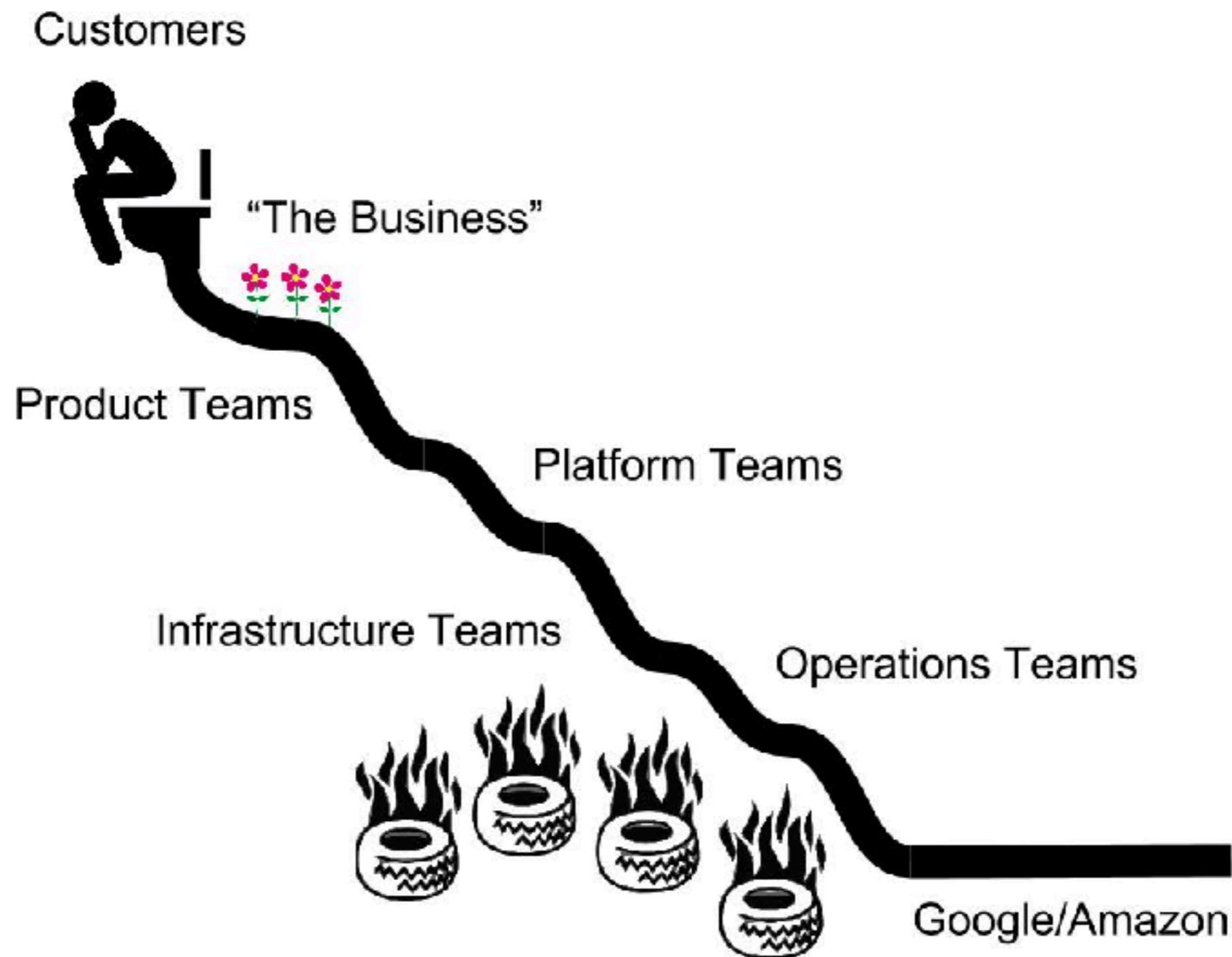
**[https://github.com/up1/  
course-devops-workshop](https://github.com/up1/course-devops-workshop)**



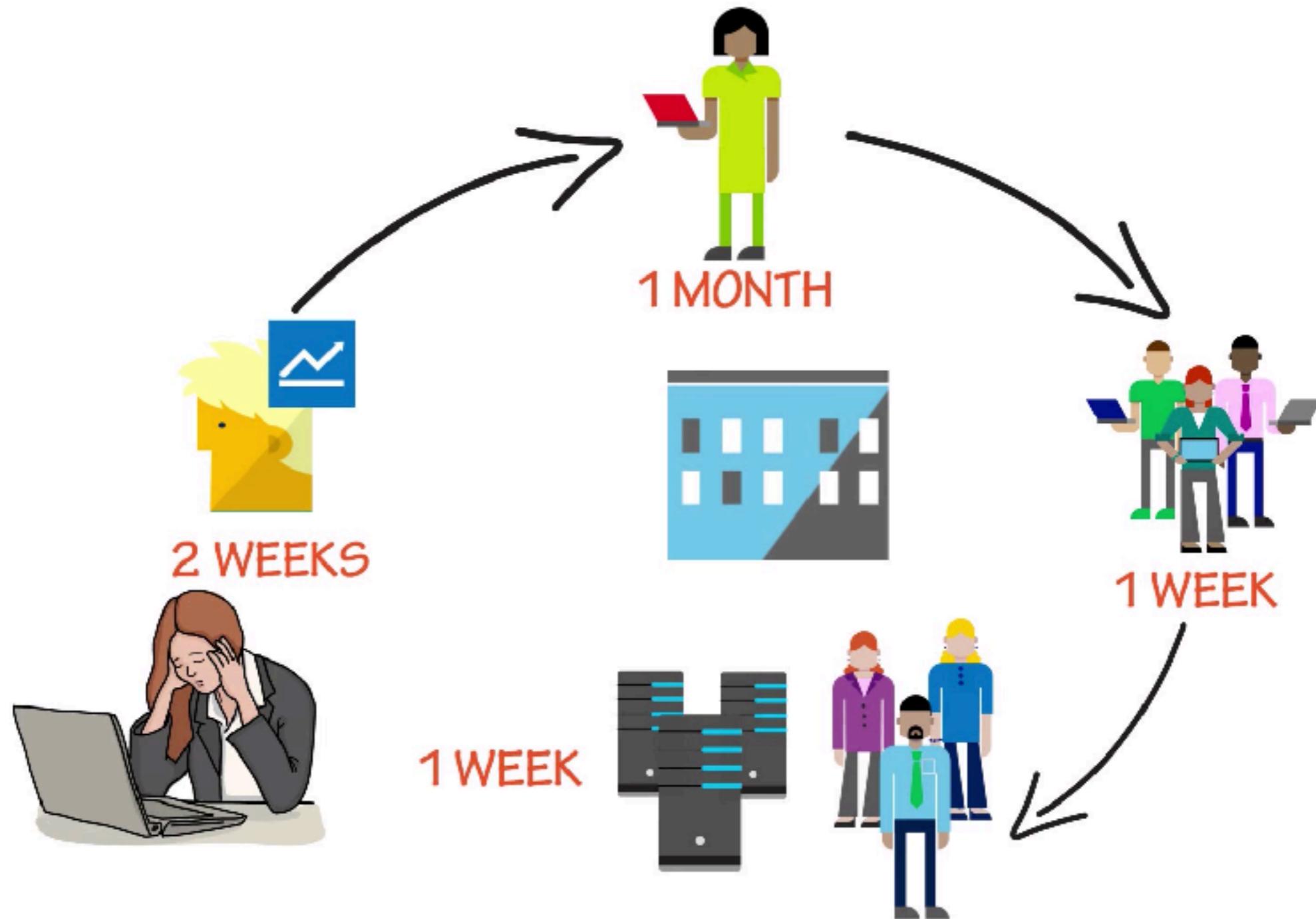
# DevOps foundation



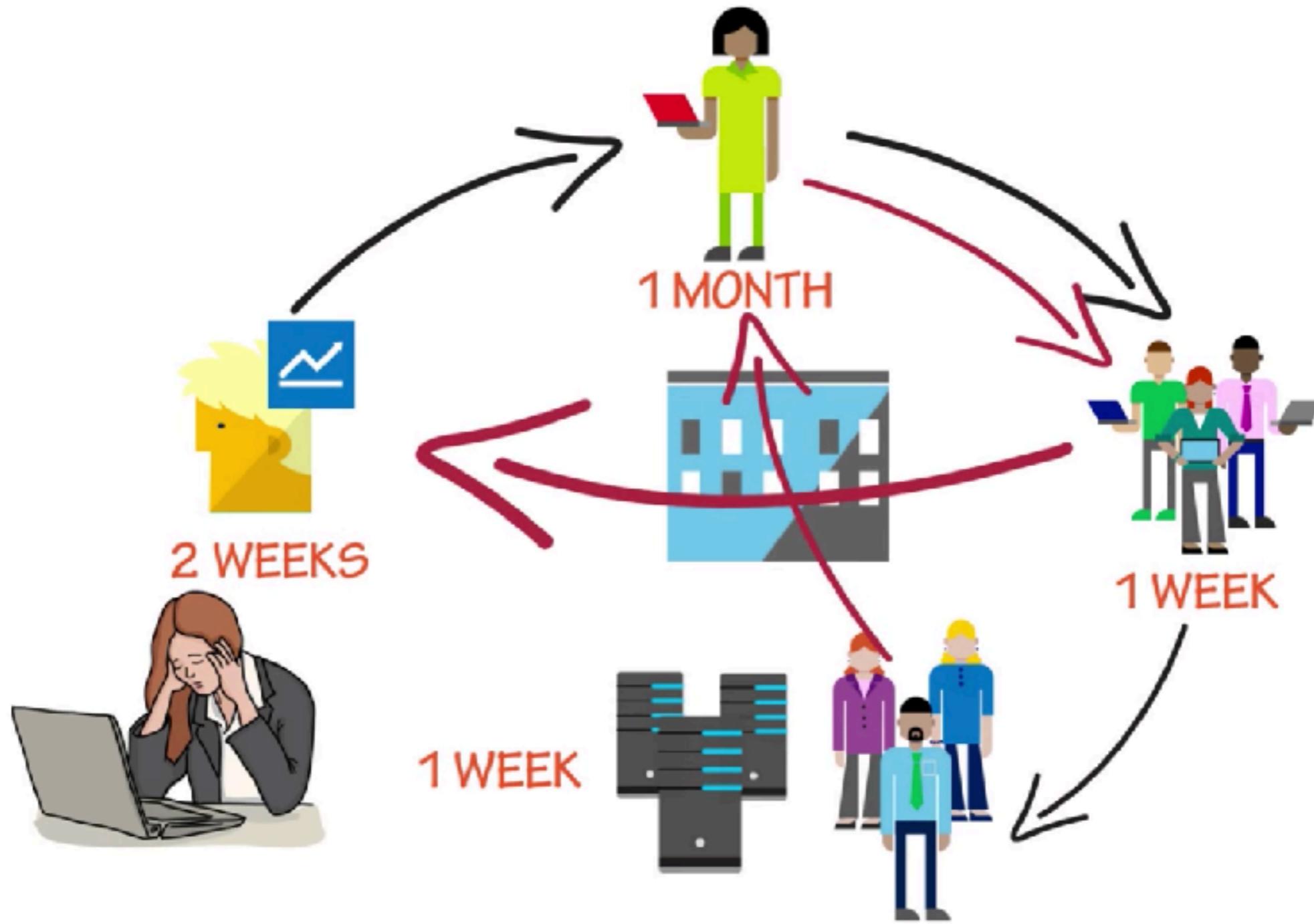
# SDLC



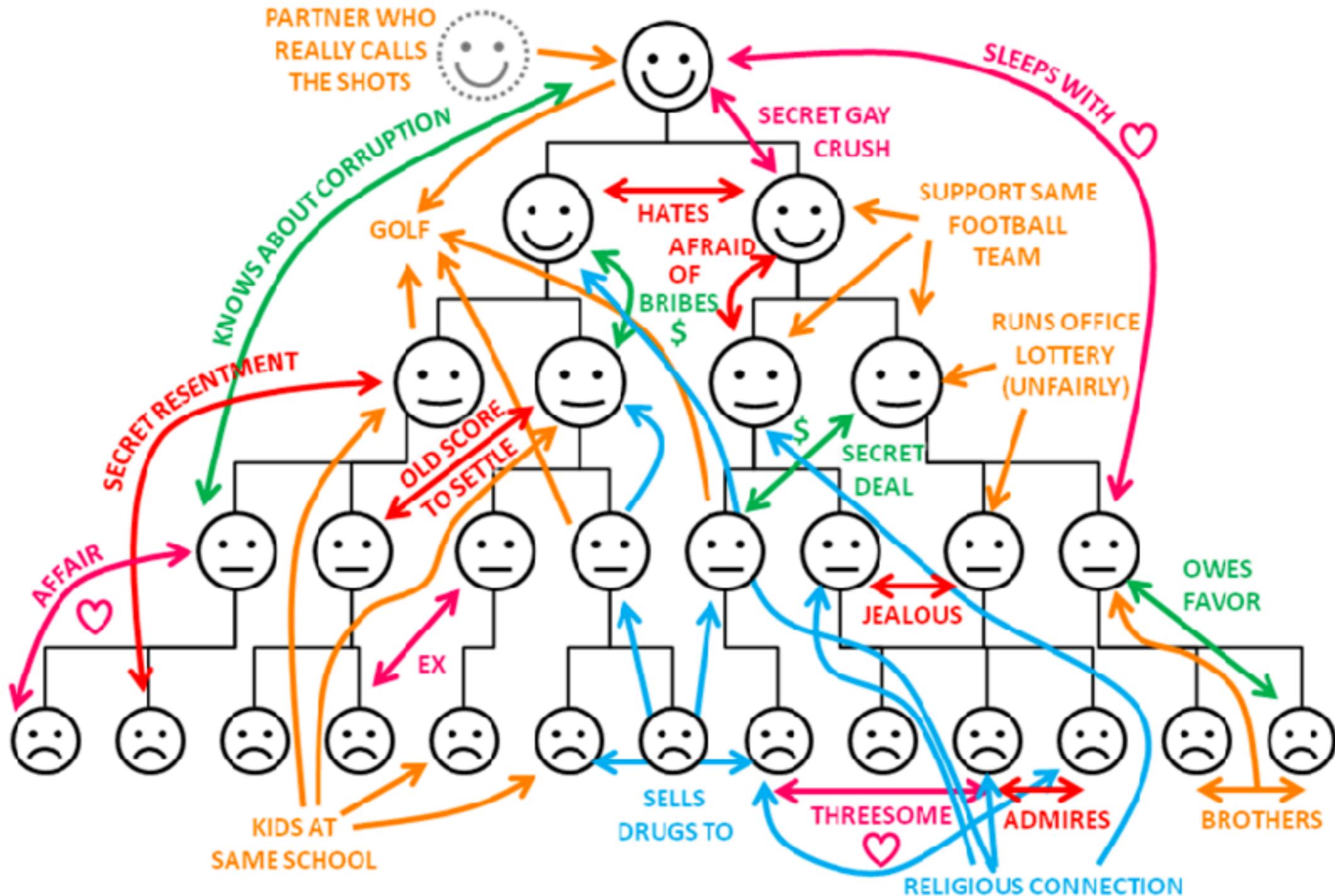
# SDLC



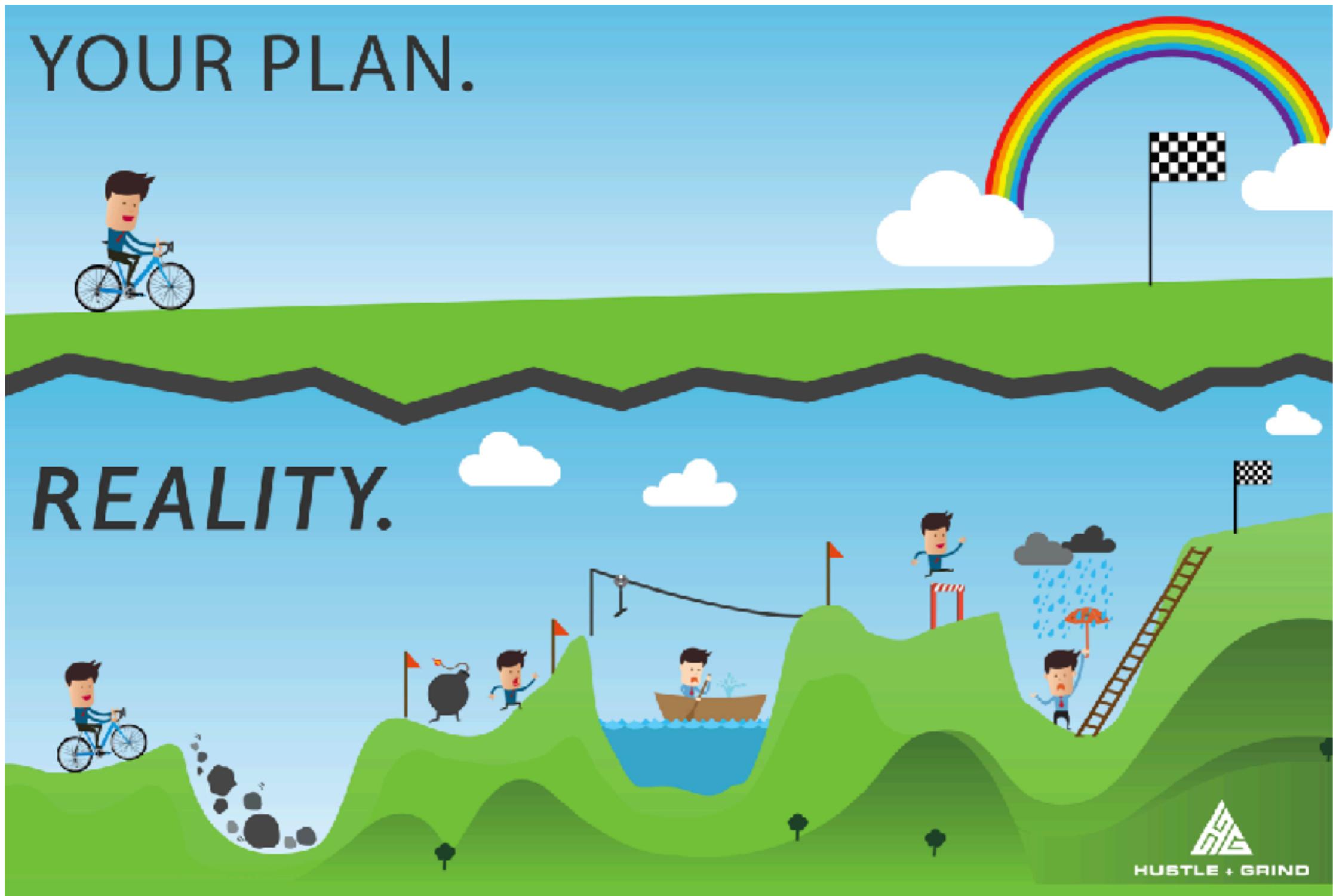
# SDLC



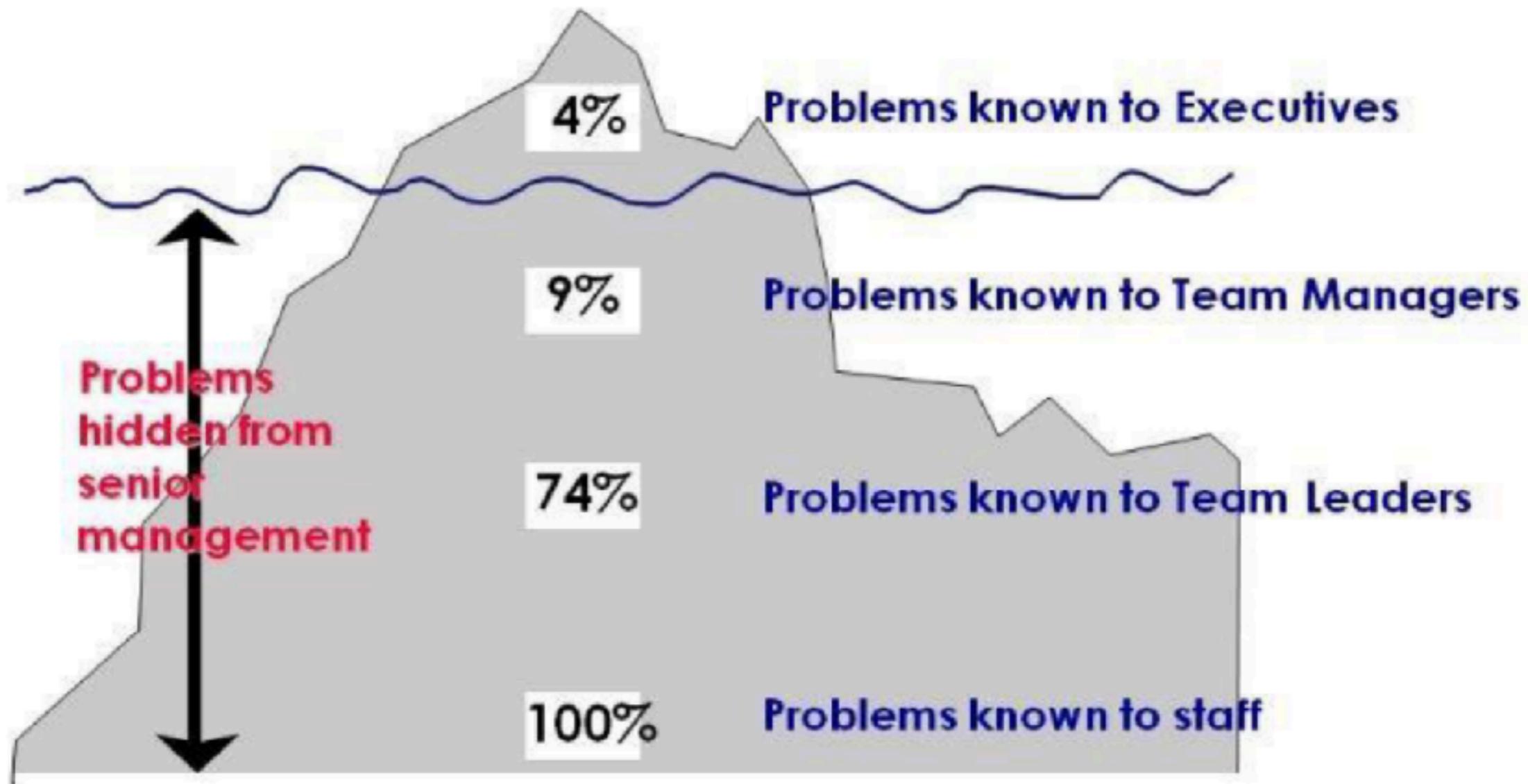
# Organization !!



# Plan vs Reality



# The iceberg of ignorance



# Consequence of inefficient



## Grow Fat

Code base grows. All the things slow down.



## Age

Your code base will become a jurassic park introducing new tech becomes hard



## Ownership

Who is responsible for which part and more important: who has the pager



## Economies of Scale

The bigger the team the more they interrupt each other



# Consequence of inefficient



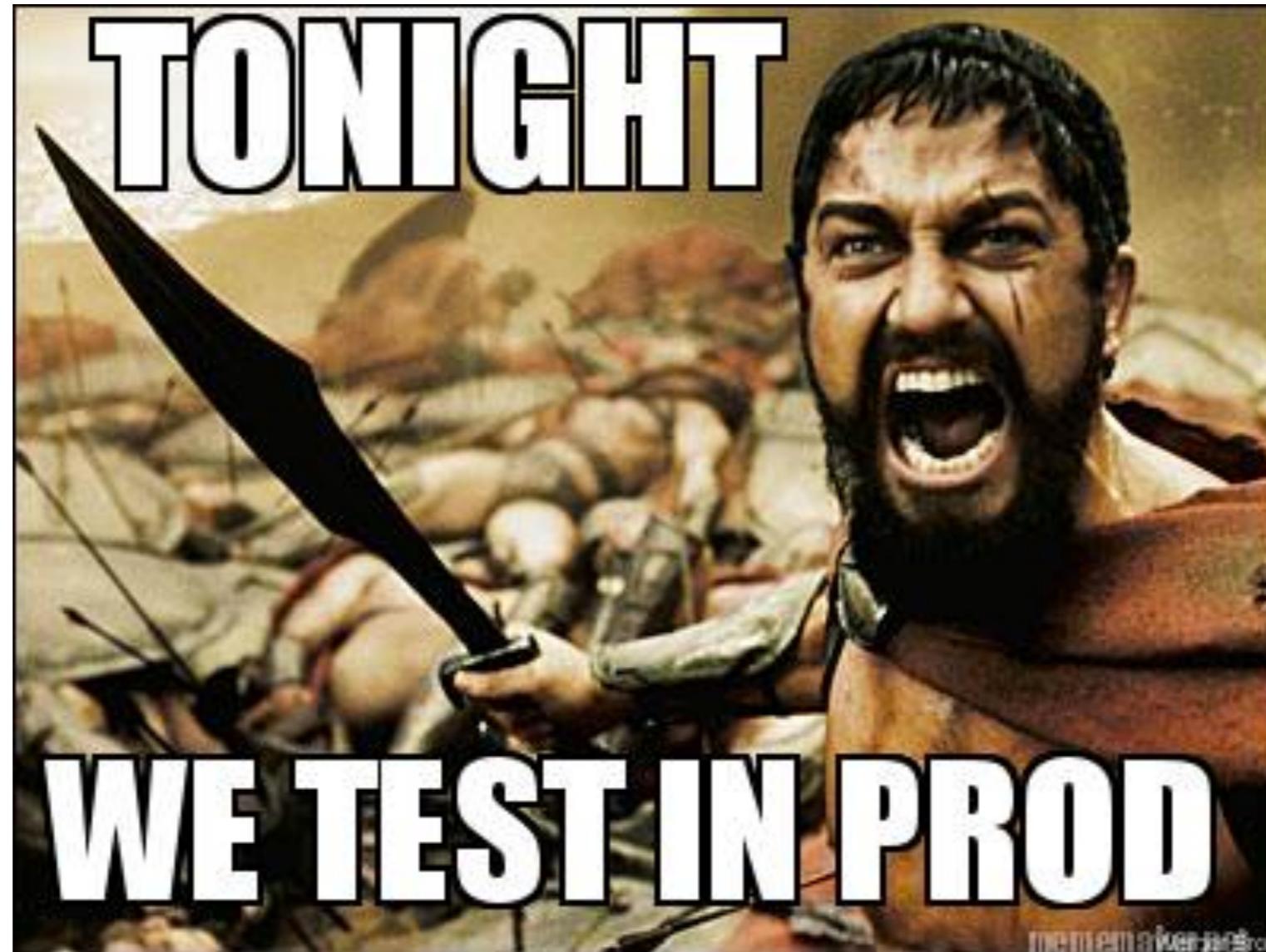
[www.cartoonistshilpa.com](http://www.cartoonistshilpa.com)



# Consequence of inefficient



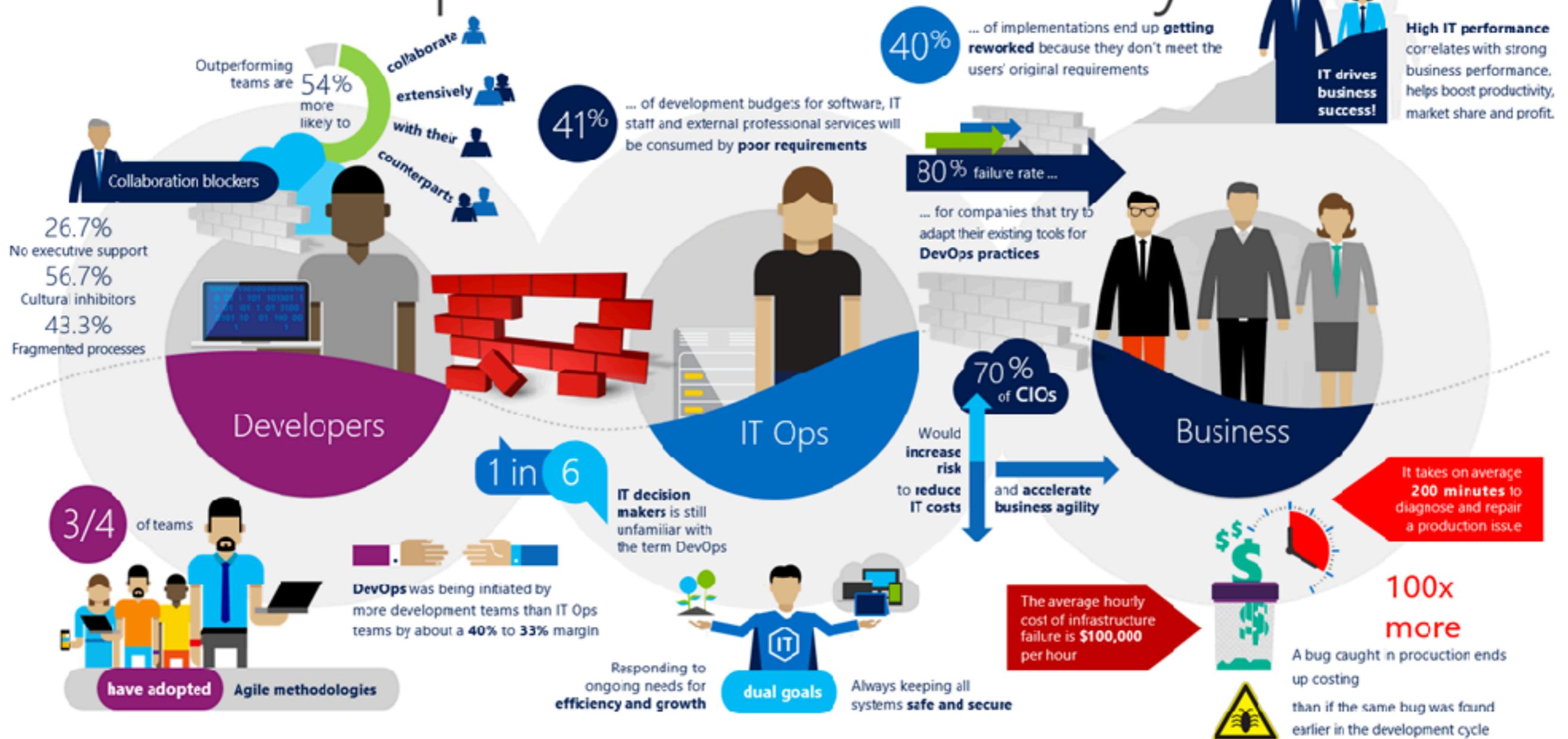
# Consequence of inefficient



# Consequence of inefficient

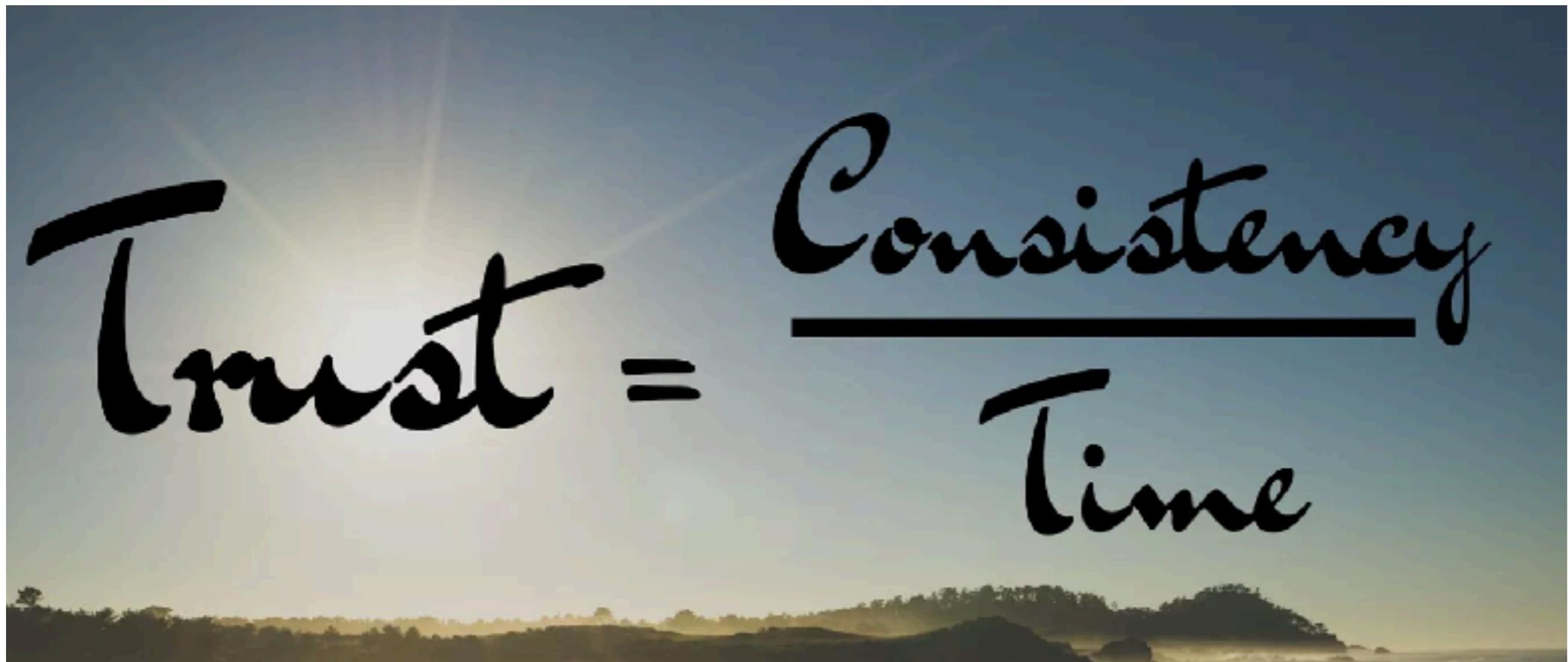


# The consequences of inefficiency



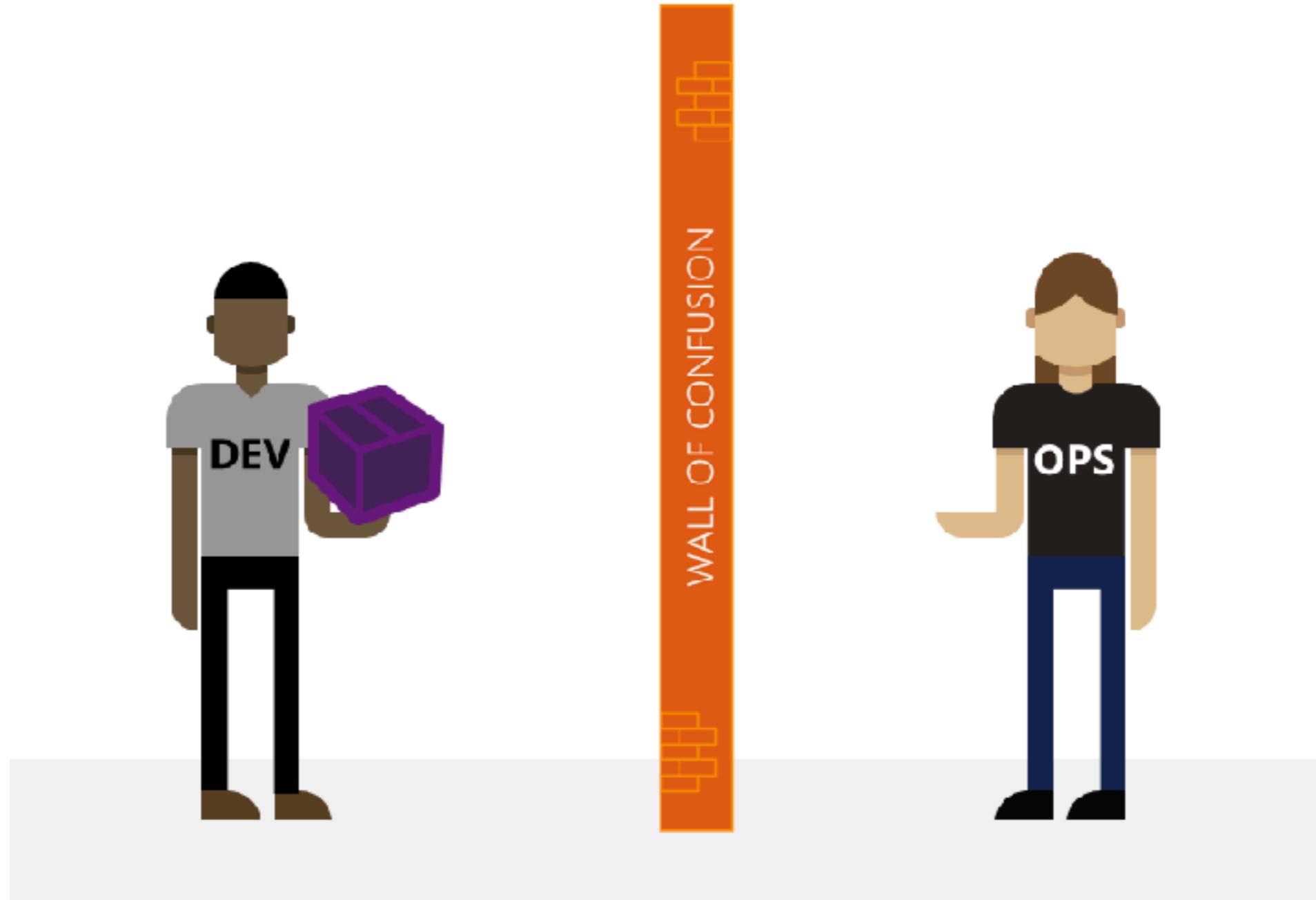
<https://channel9.msdn.com/Series/DevOps-Fundamentals>



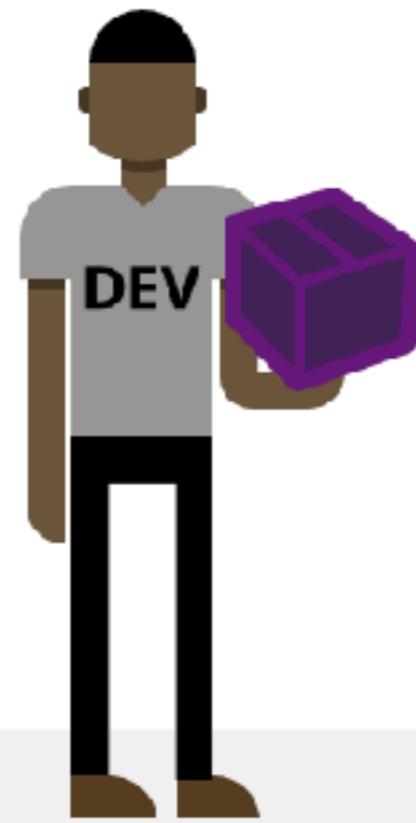


# Development vs Operations



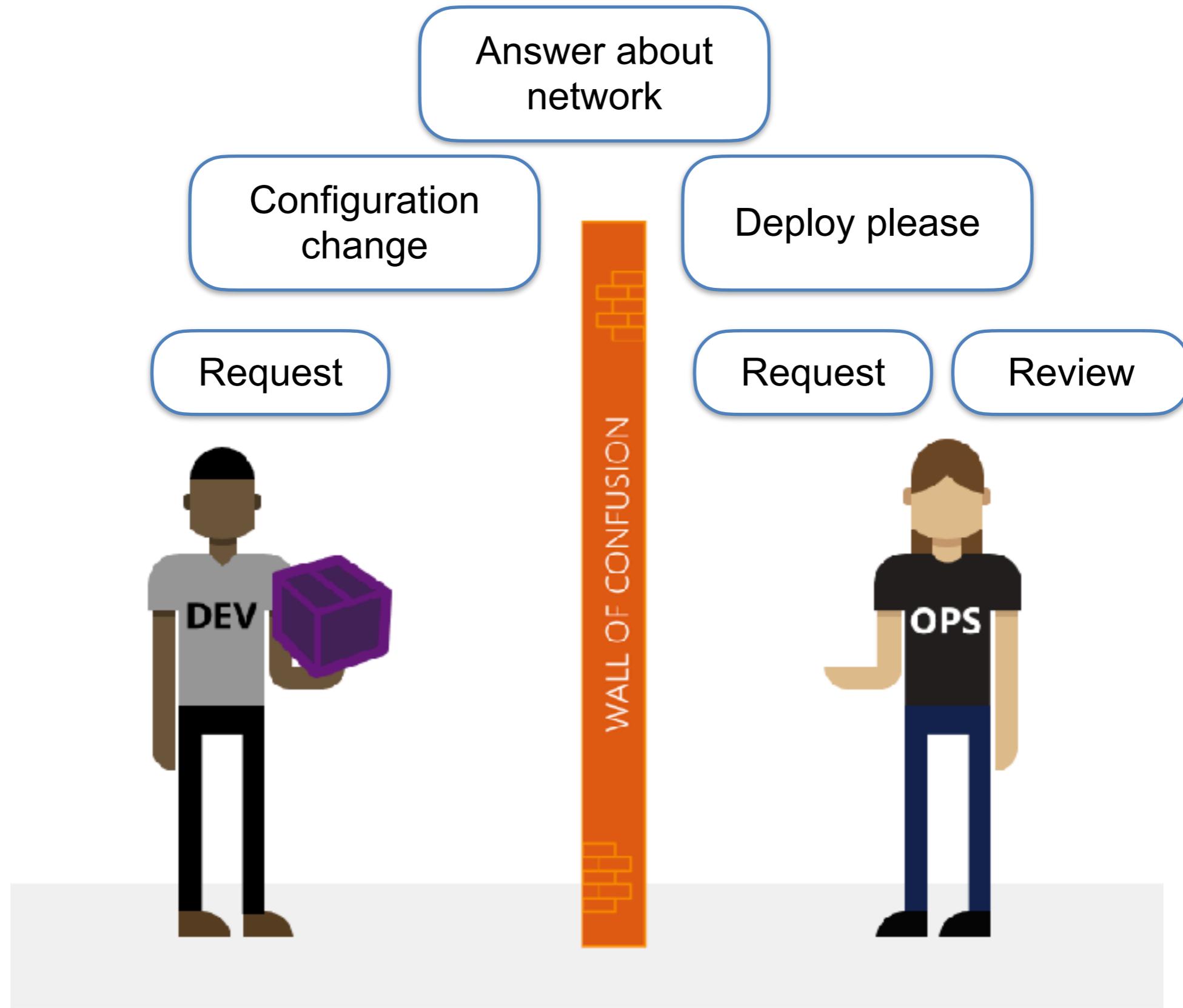


**I want to change !**

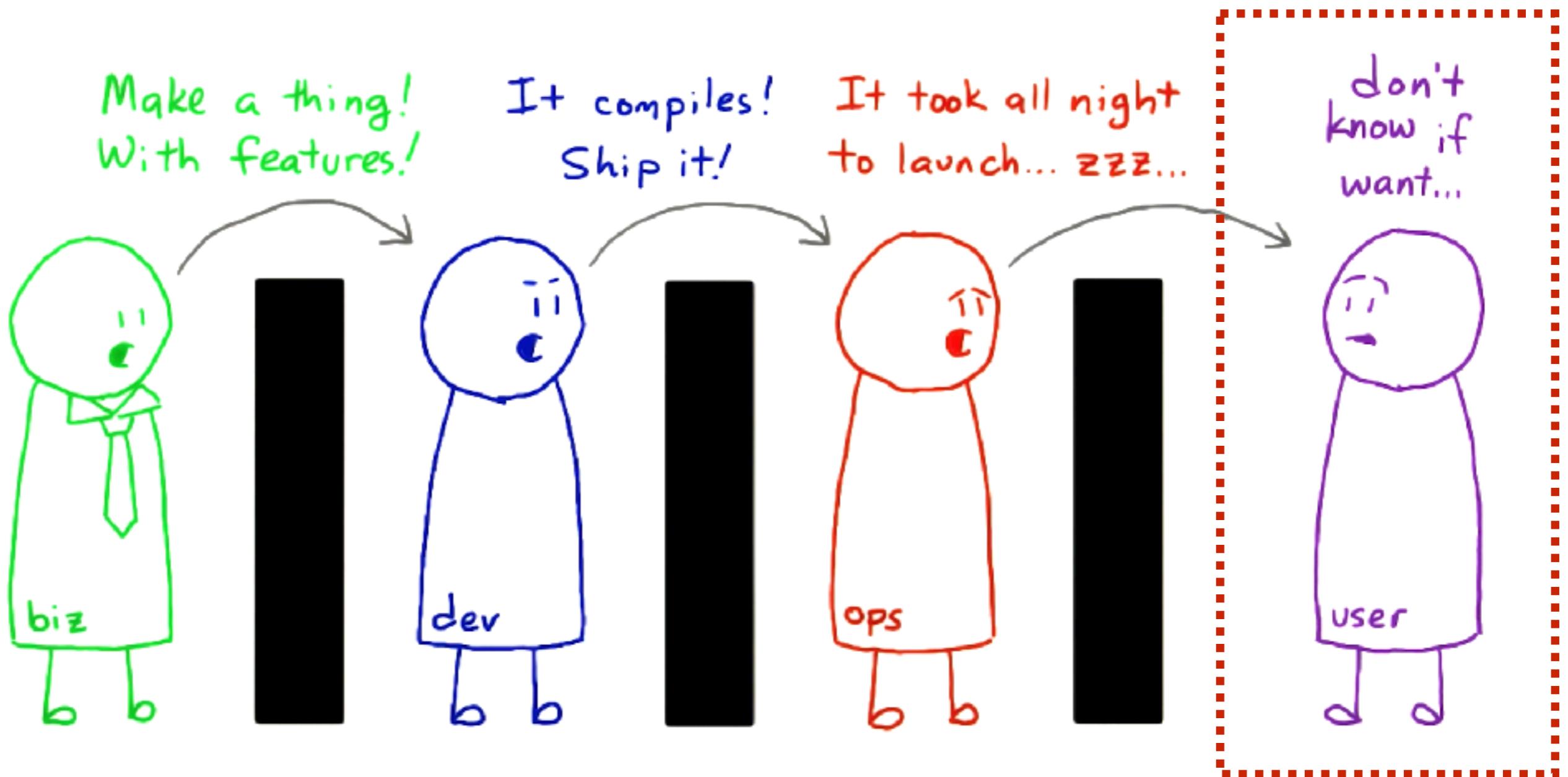


**I want to stability !**





# Result ?



# Problems ?

Deadline Driven Development

Delayed project/product deliver

Bad quality

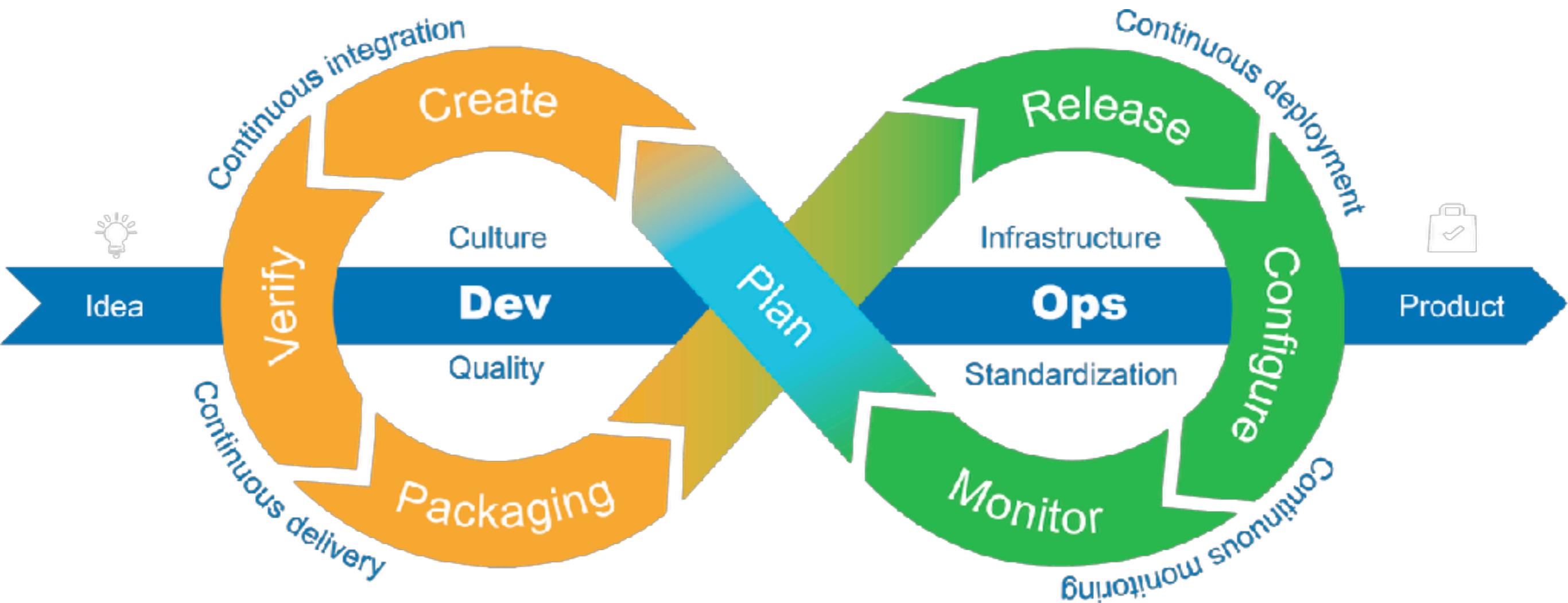
Low customer satisfaction



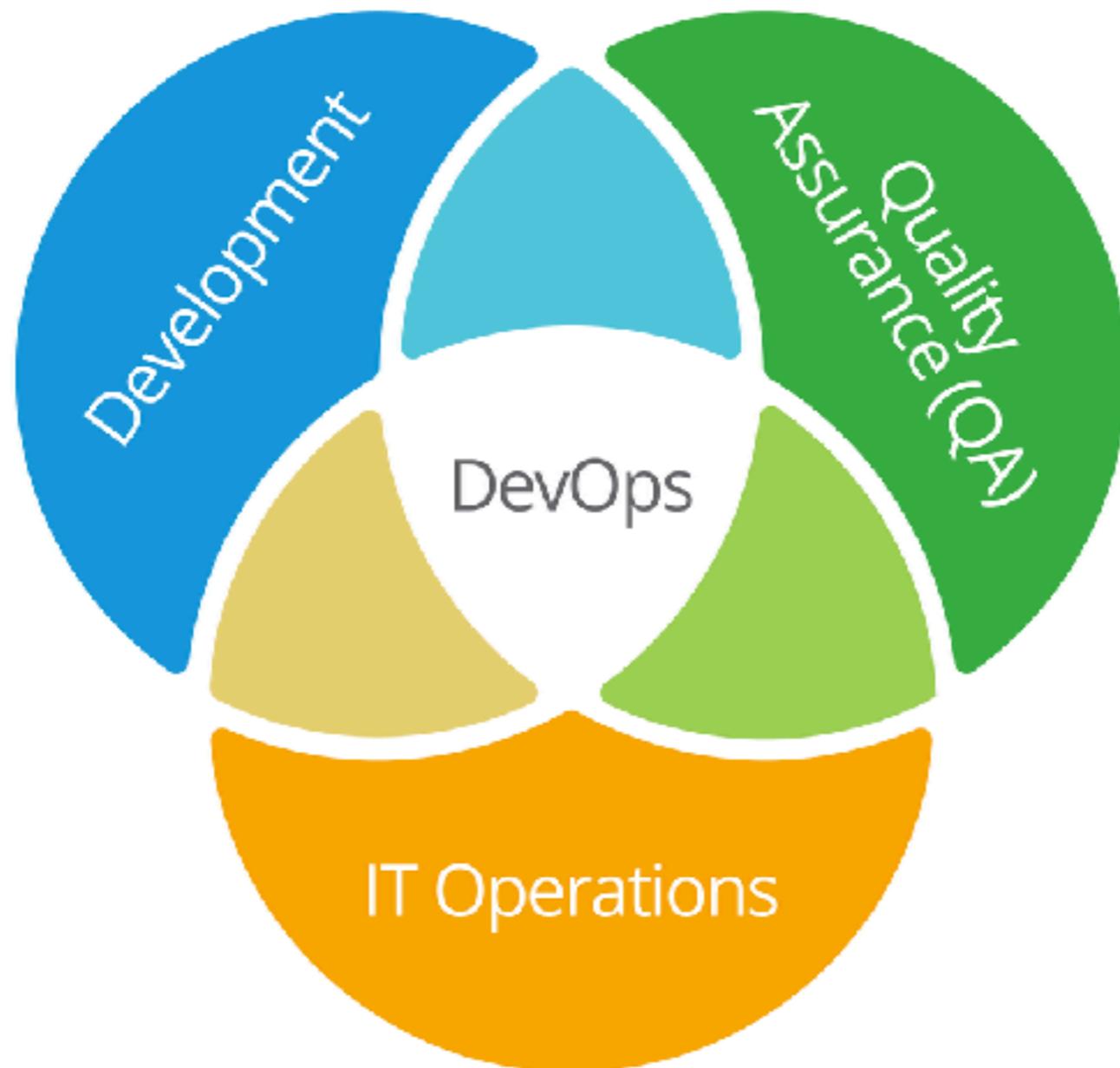
# Rise of DevOps



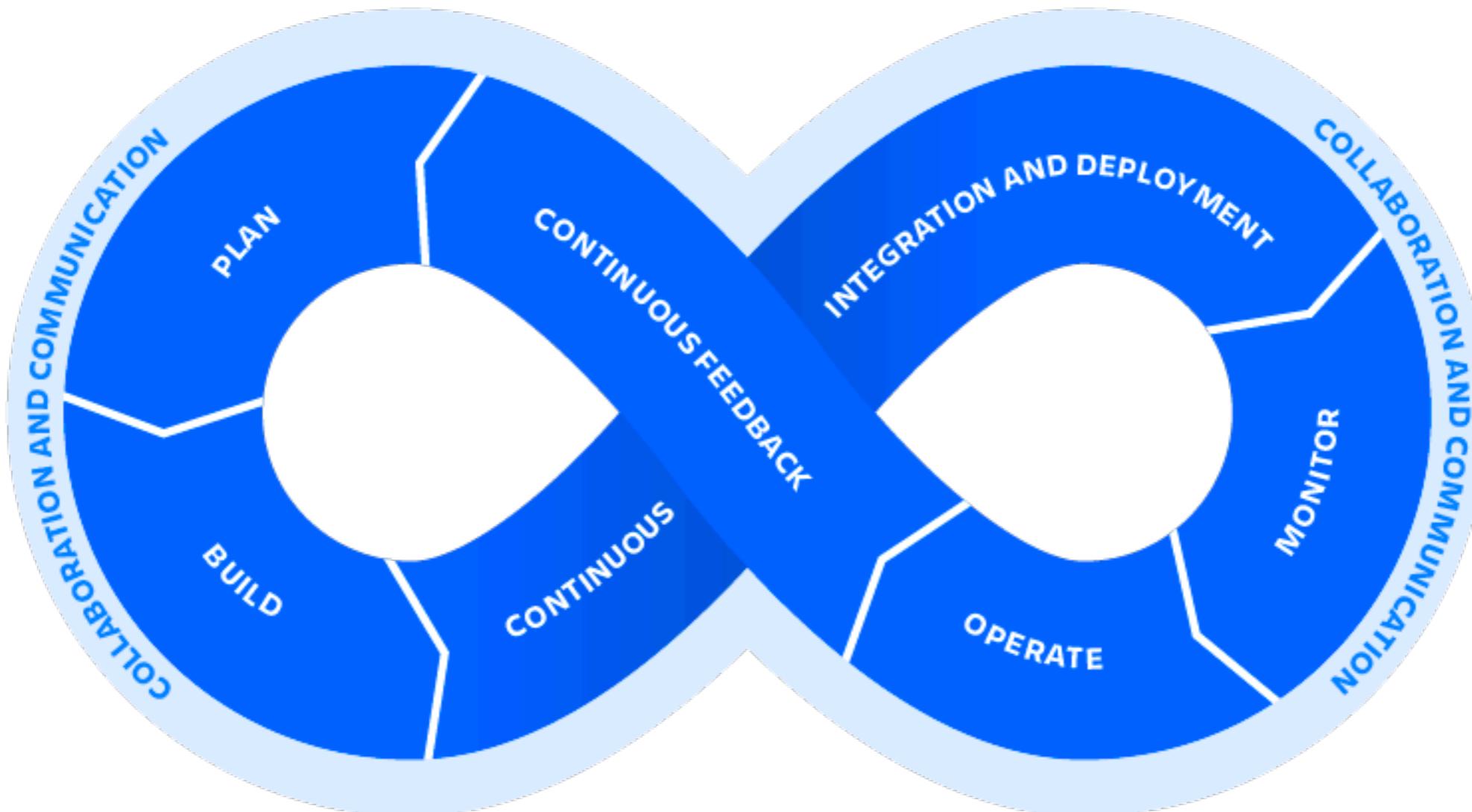
# DevOps



# DevOps



# DevOps

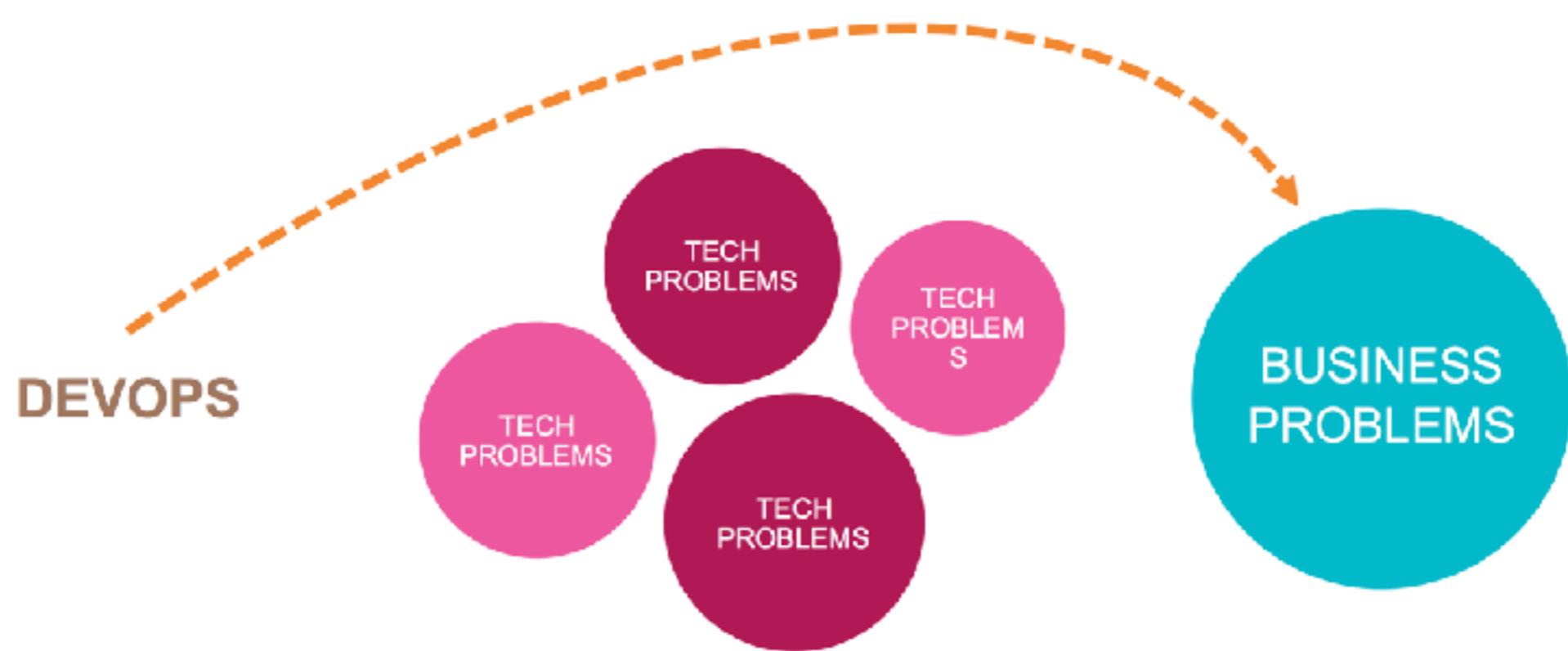


**Devops isn't any single person's job.  
It's everyone's job.**

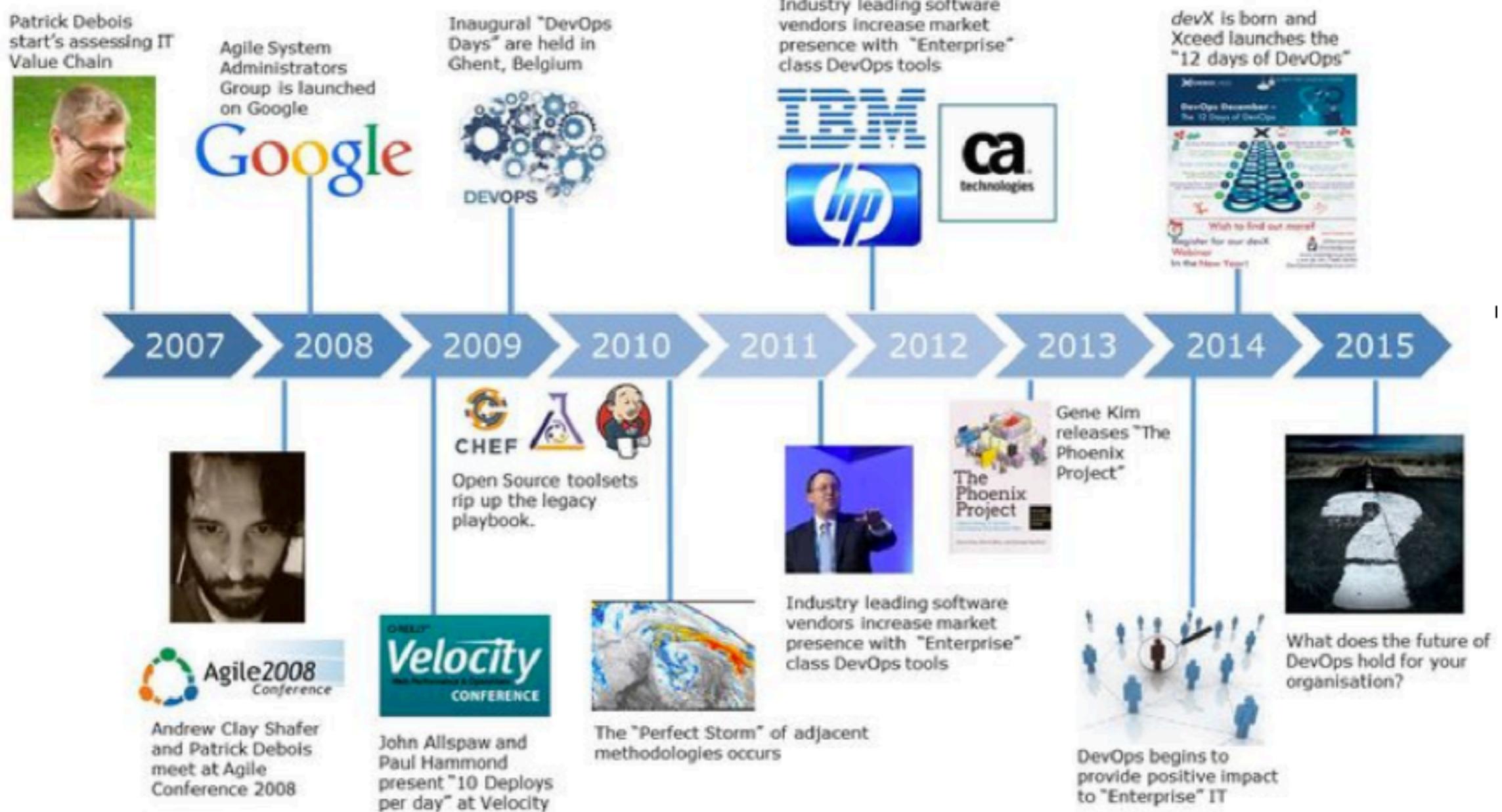


# DevOps

Solve business problem first



# History of DevOps



# State of DevOps Report

**High-performing teams deploy more frequently and have much faster lead times.**



**200x more frequent deployments**



**2,555x shorter lead times**

**They make changes with fewer failures, and recover faster from failures.**



**3x lower change failure rate**



**24x faster recovery from failures**

<https://puppet.com/resources/whitepaper/state-of-devops-report>



# What is DevOps ?



# DevOps is not ...

Certification

Role

Set of tools

Prescriptive process



# What is DevOps ?

**“DevOps is development and operations collaboration”**

**“DevOps is using automation”**

**“DevOps is small deployments”**

**It's DevOps!**

**It's DevOps!**

**It's DevOps!**

**It's DevOps!**

**“DevOps is treating your infrastructure as code”**

**“DevOps is feature switches”**

**“Kanban for Ops?”**



# What is DevOps ?

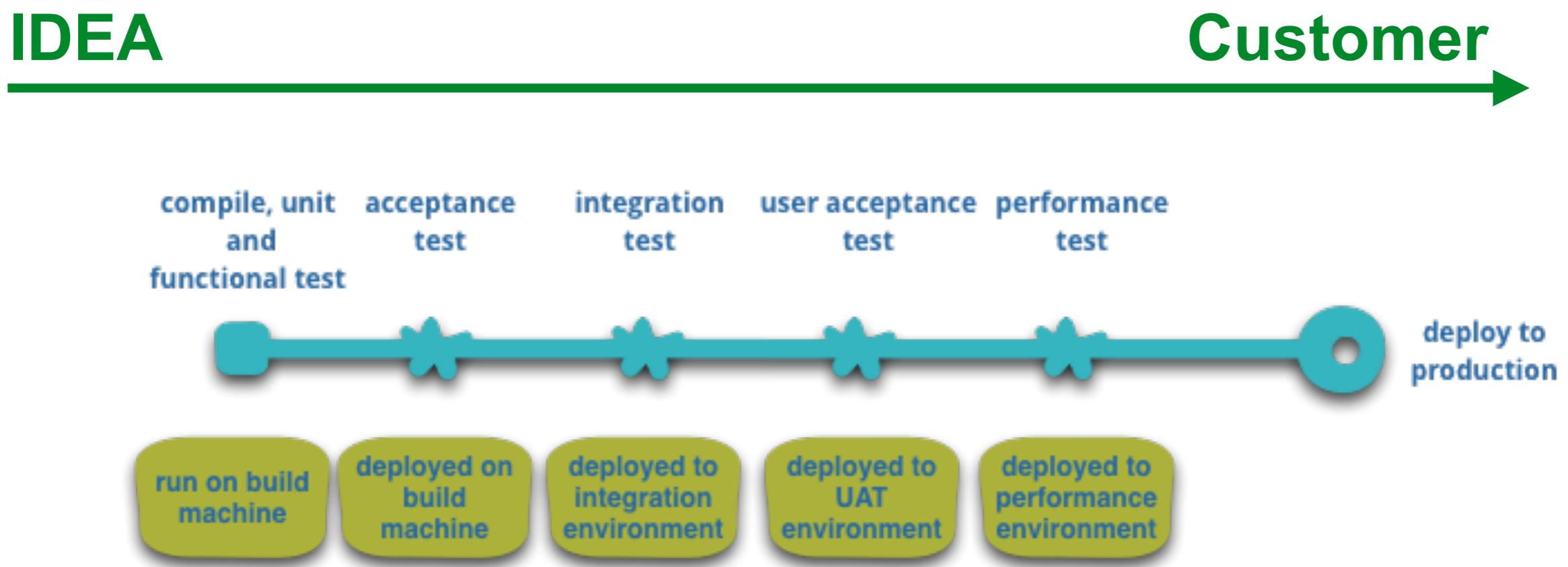
DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality

[https://en.wikipedia.org/wiki/DevOps#Definitions\\_and\\_history](https://en.wikipedia.org/wiki/DevOps#Definitions_and_history)



# Goals of DevOps

**Improve the delivery of value  
for Customer and Business**



# Goals of DevOps

Enable the **continuous delivery** of value to customer and business



# Continuous Delivery

The more often to deploy

Small change

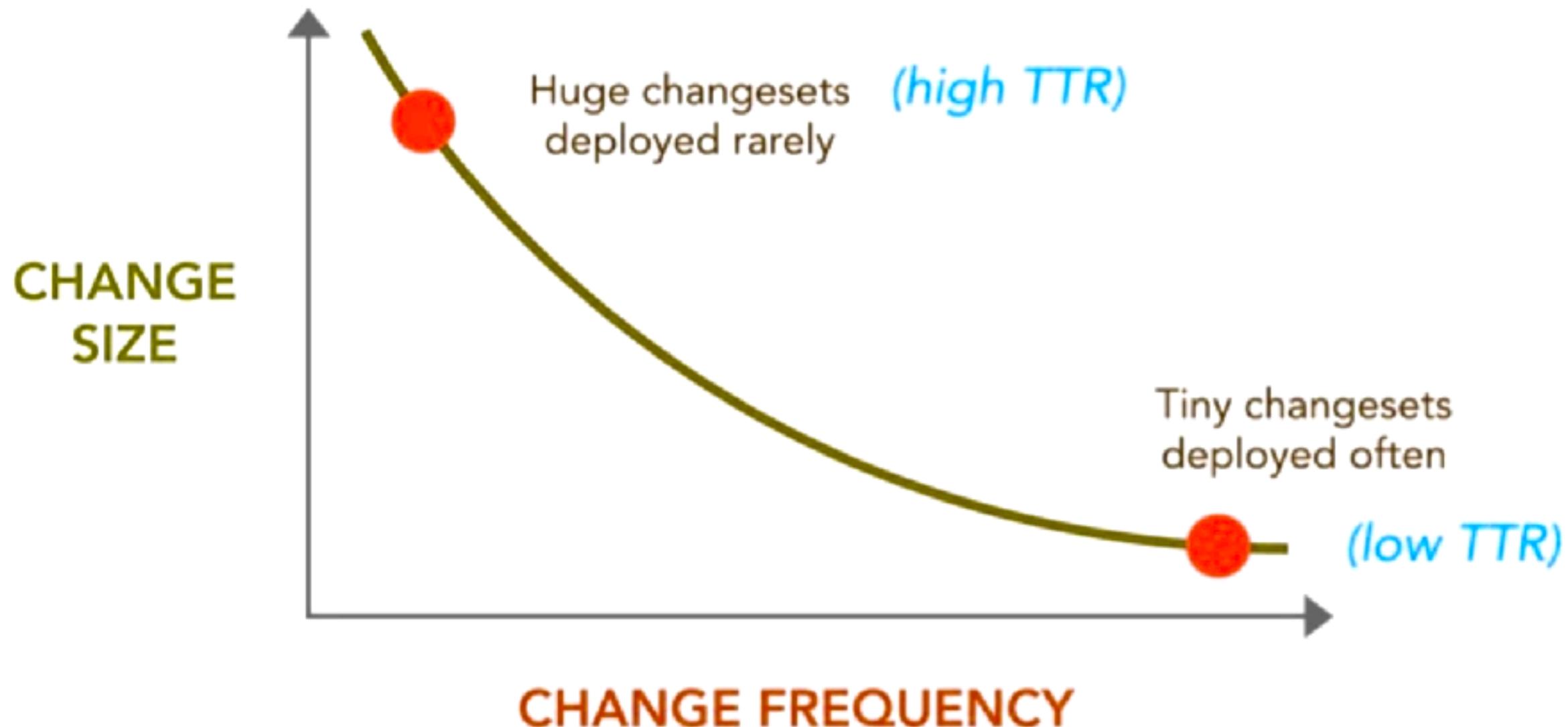
Automate it

Reduce TTR (Time to Repair/Recovery)

Learn and repeat



# Continuous Delivery



# DevOps Three Ways

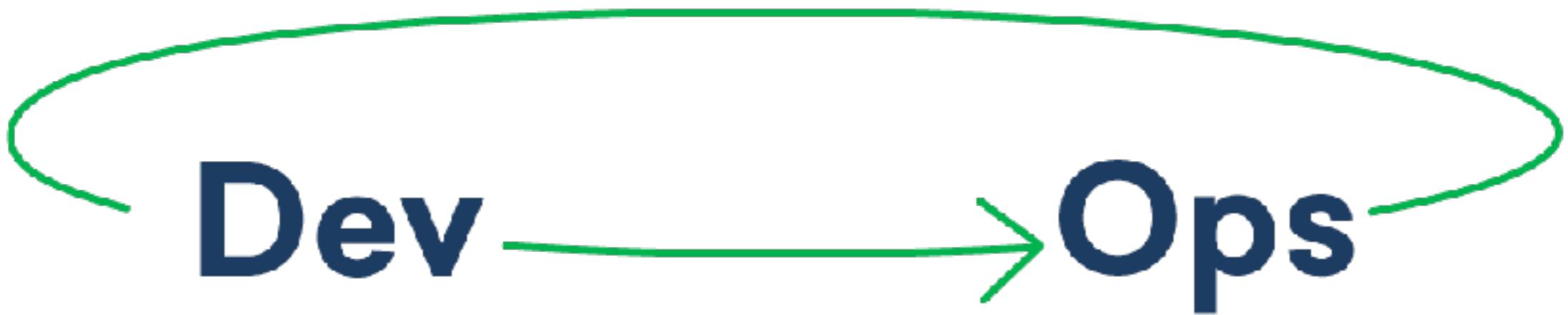
1. Flow of work
2. Feedback process
3. Environment and culture



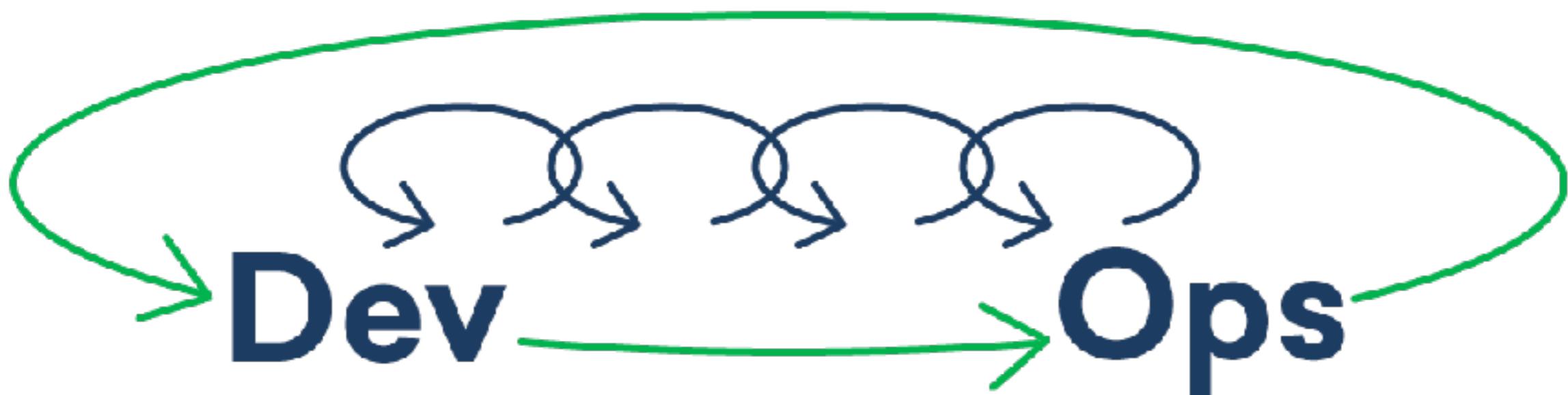
# 1. Flow of work



## 2. Feedback process/loop



# 3. Environment and culture



# CALMS Framework

Culture  
Automation  
Lean  
Measurement  
Sharing



# DevOps is all about **Human** problems



# DevOps Metrics

Lead time for changes

Change failure rate

Deployment frequency

Mean time to recovery (MTTR)

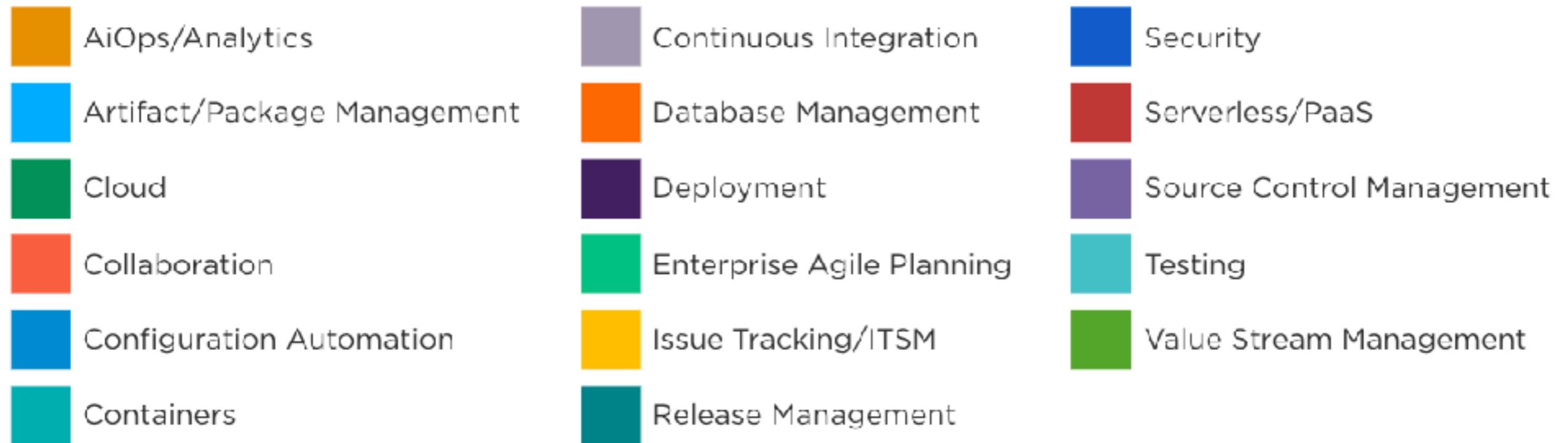


# DevOps Tools



<https://digital.ai/periodic-table-of-devops-tools>





<https://digital.ai/periodic-table-of-devops-tools>



# Workshop

## Design your delivery process



# DevOps with Security

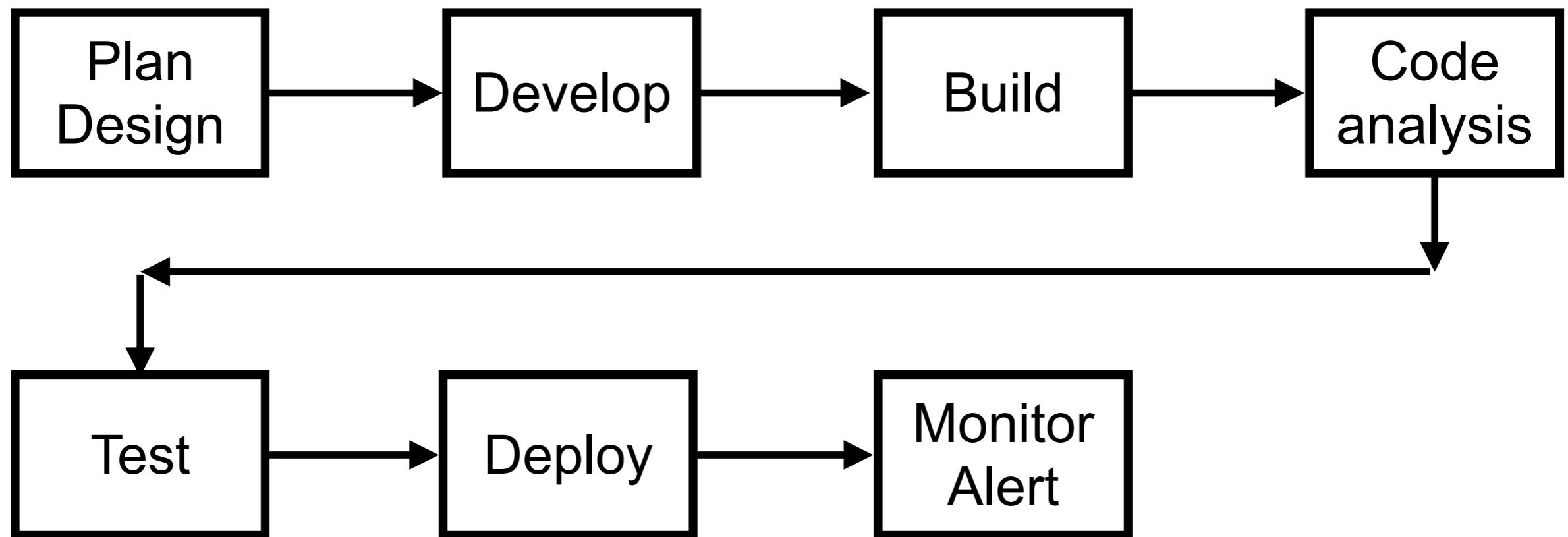


# DevSecOps

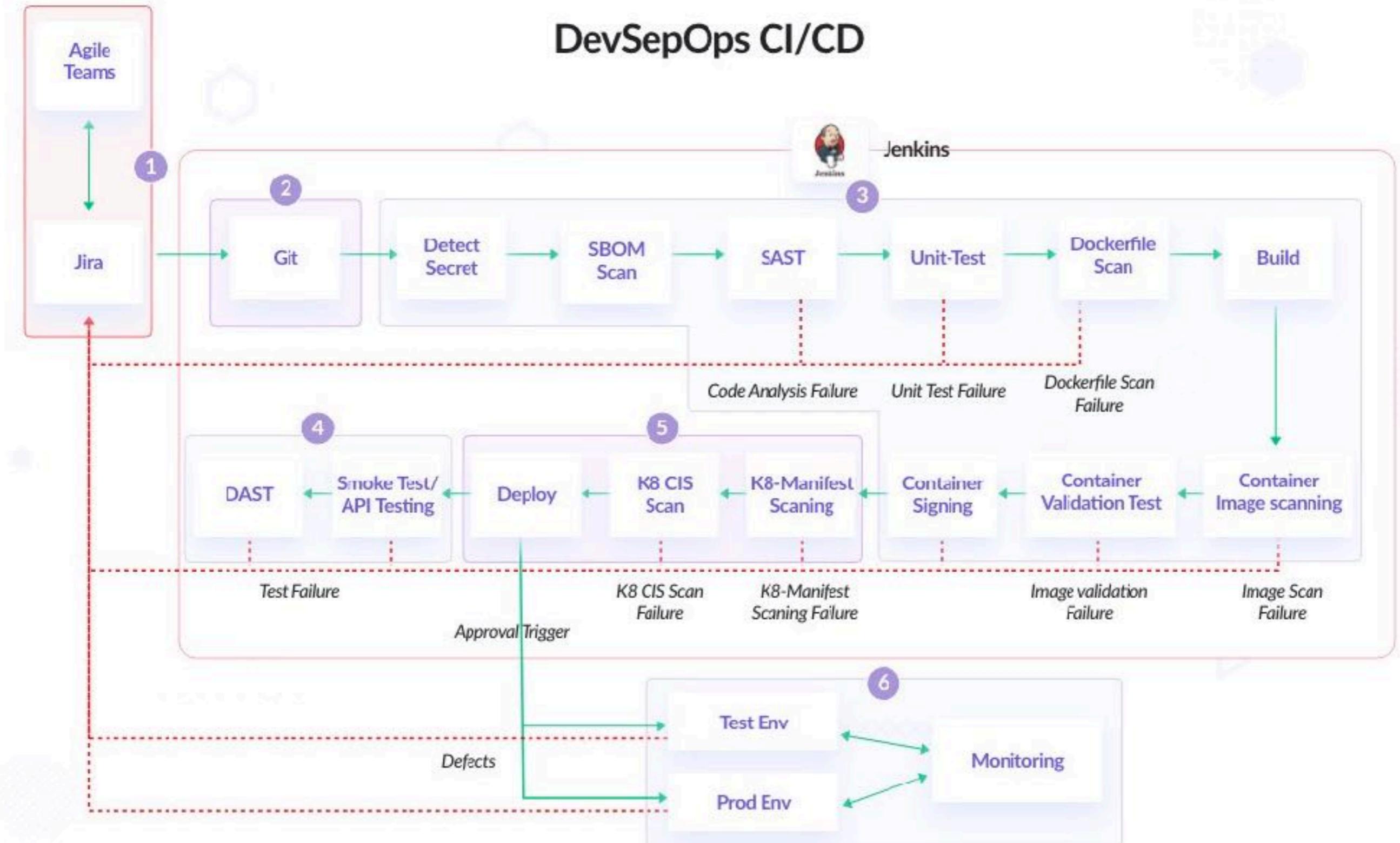
Security in software development life cycle  
Finds vulnerability and bugs at an earlier stage of development



# DevSecOps Pipeline



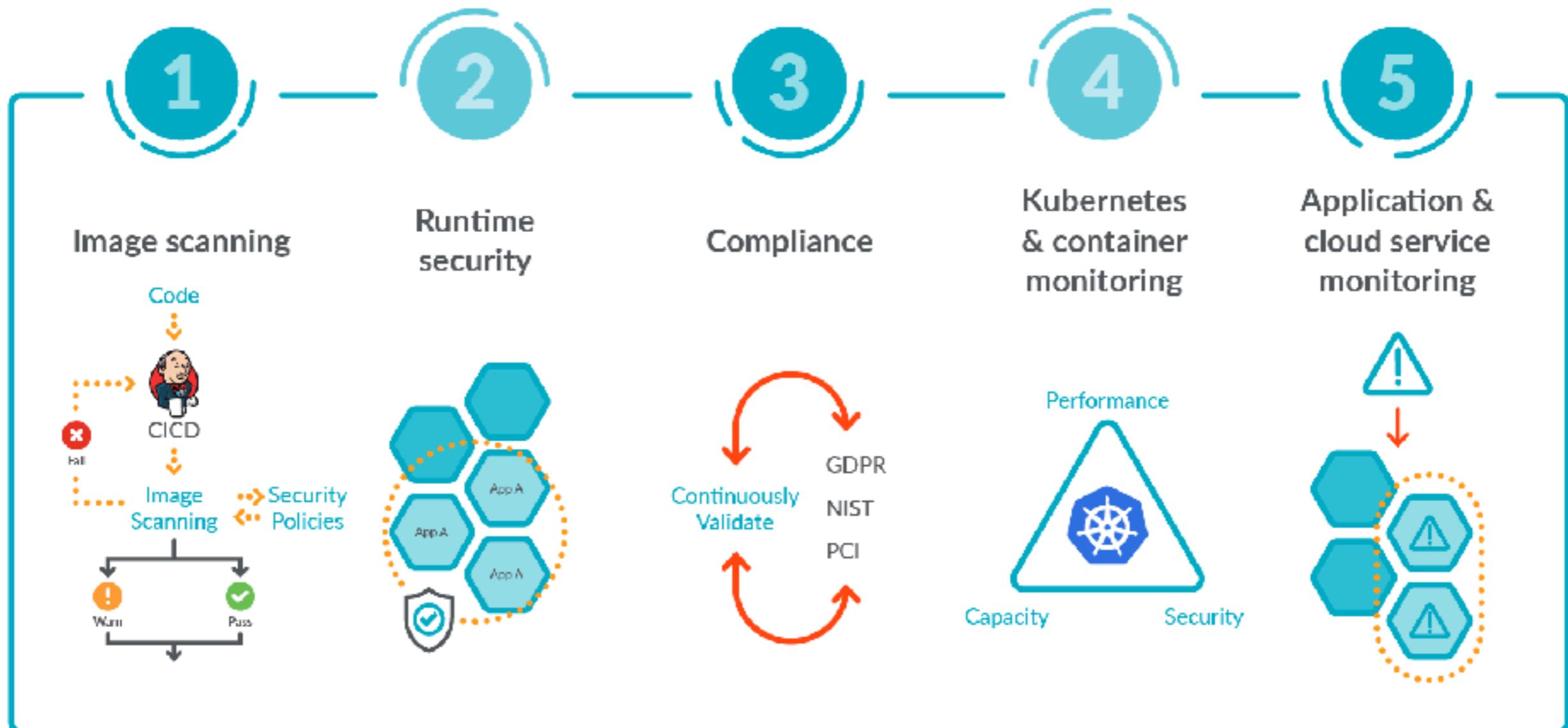
# DevSepOps CI/CD



<https://www.infracloud.io/blogs/implement-devsecops-secure-ci-cd-pipeline/>



# Five Essential Workflows for Secure DevOps



<https://sysdig.com/blog/image-scanning-best-practices/>



# Plan and Design

## Threat modeling

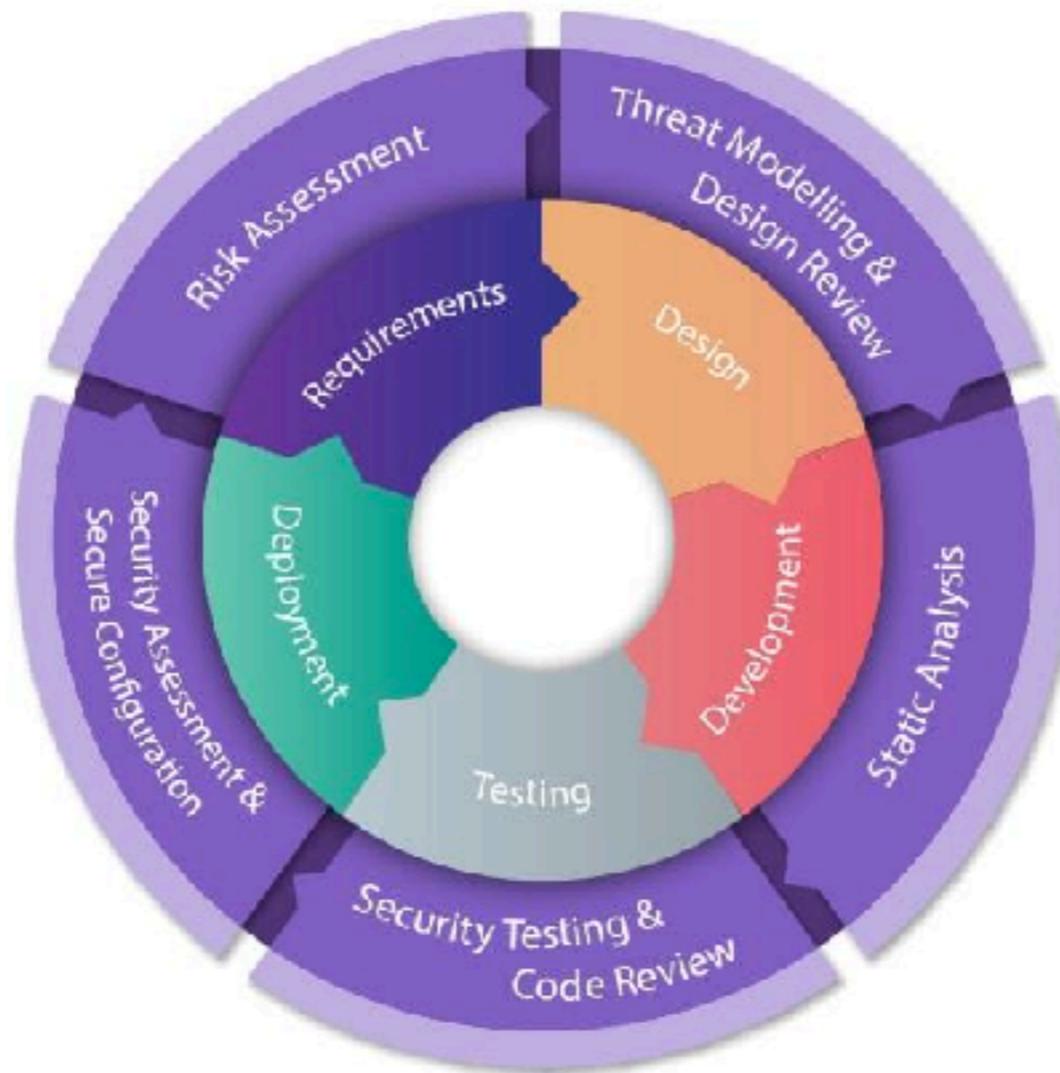


[https://owasp.org/www-community/Threat\\_Modeling](https://owasp.org/www-community/Threat_Modeling)



# Plan and Design

## Secure Software Development Life Cycle (SSDLC)



<https://snyk.io/learn/secure-sdlc/>



# Develop

Shift left security in development phase  
SonarLint  
Pre-commit hooks  
Setup protected branch  
**Signed git commit with GPG key**

[https://docs.gitlab.com/ee/user/project/repository/gpg\\_signed\\_commits/](https://docs.gitlab.com/ee/user/project/repository/gpg_signed_commits/)



# Build and Code analysis

Scan code for vulnerabilities and secrets

SonarQube

Dependency check

Detect code lead to a memory leak

Scan docker image

Signing and verify a docker image



# Container !!

How to identify malware in a container ?



Docker  
scout

<https://snyk.io/learn/docker-security-scanning/>



# Docker Security Top 10

## OWASP/Docker-Security



Getting a handle on container security

81 9

Contributors

0 14

Issues

★ 571

Stars

0 122

Forks



<https://github.com/OWASP/Docker-Security>



# Scan for secret and credential

Detect-secret  
Gitleaks

```
$detect-secrets scan test_data/ --all-files
```

<https://github.com/Yelp/detect-secrets>



# Deployment

Static scan of k8s manifest file

Pre-deploy policy check k8s manifest file

IaC scanning (terrascan, terratest)



# **Continuous Integration Continuous Delivery**



# Why CI/CD ?

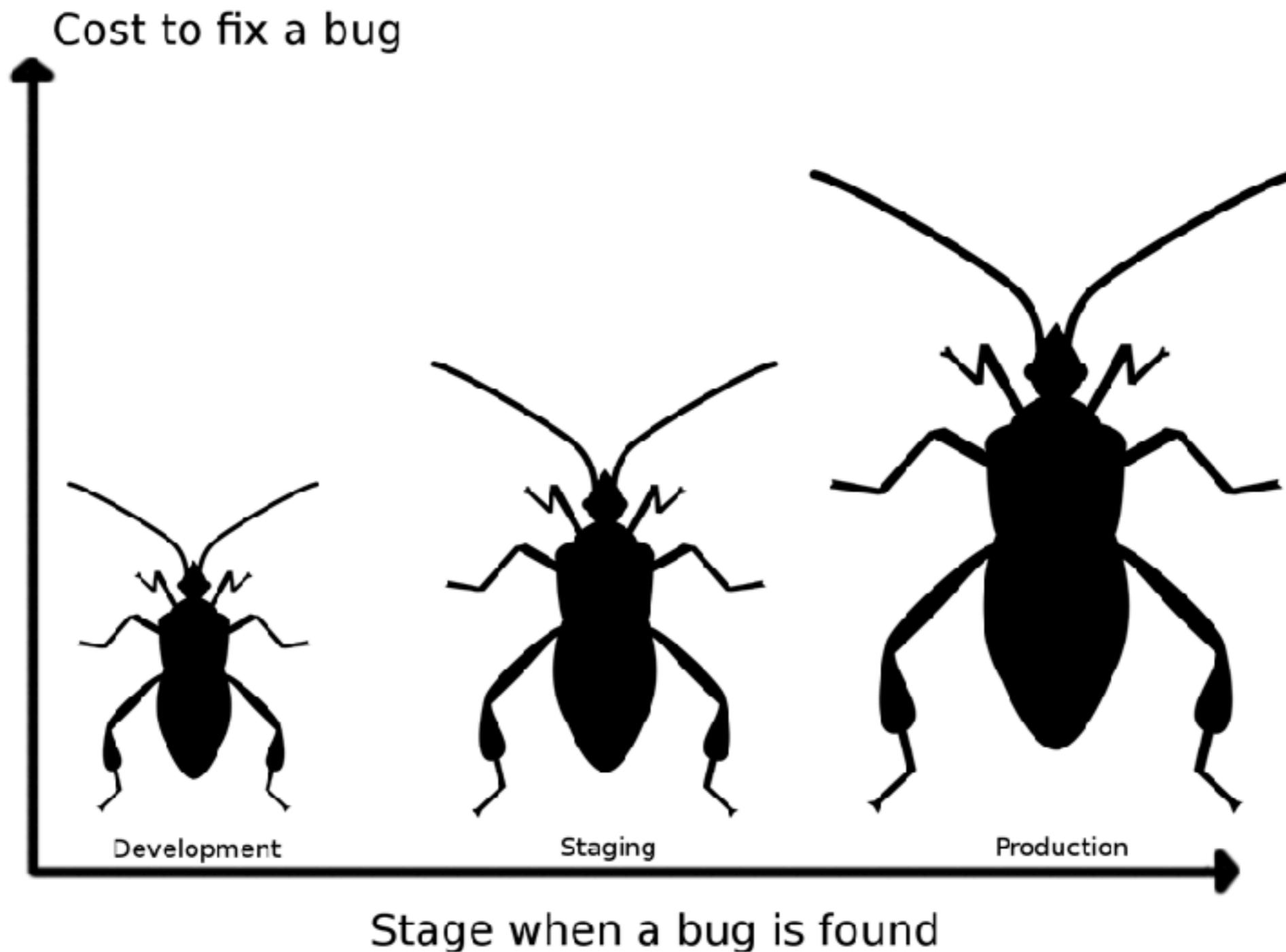


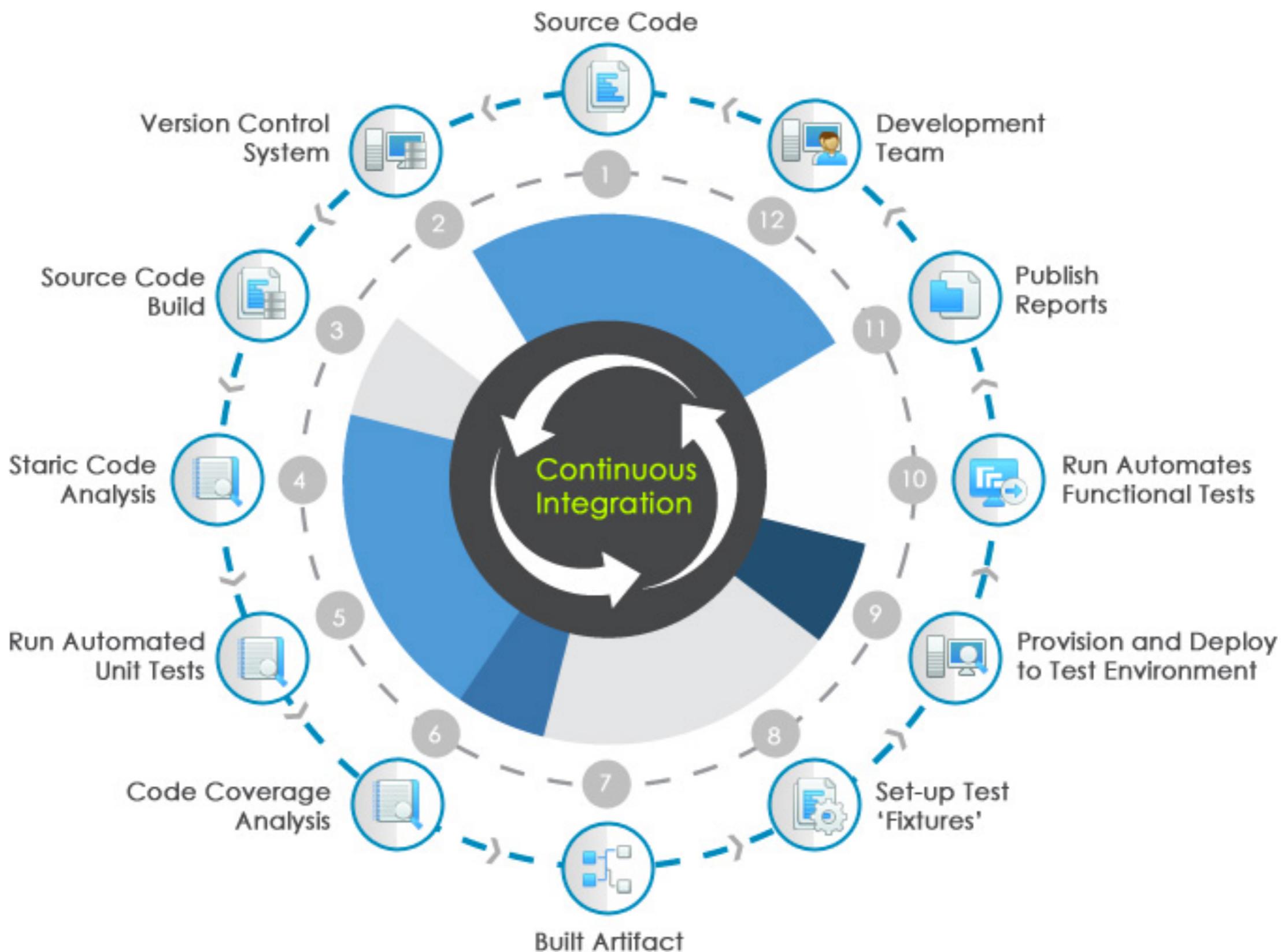
# The cost of integration

1. Merging the code
2. Duplicate changes
3. Test again again !!
4. Fixing bugs
5. Impact on stability



# The cost of integration







Jenkins

Bamboo



TeamCity

> go<sup>TM</sup>



Hudson





Jenkins

Bamboo

CI is about what people do  
not about what tools they use



Visual Studio



Team Foundation Server

Hudson



Travis

wercker

circleci



# Continuous Integration

Discipline to integrate frequently



# Continuous Integration

Strive to make **small change**



# Continuous Integration

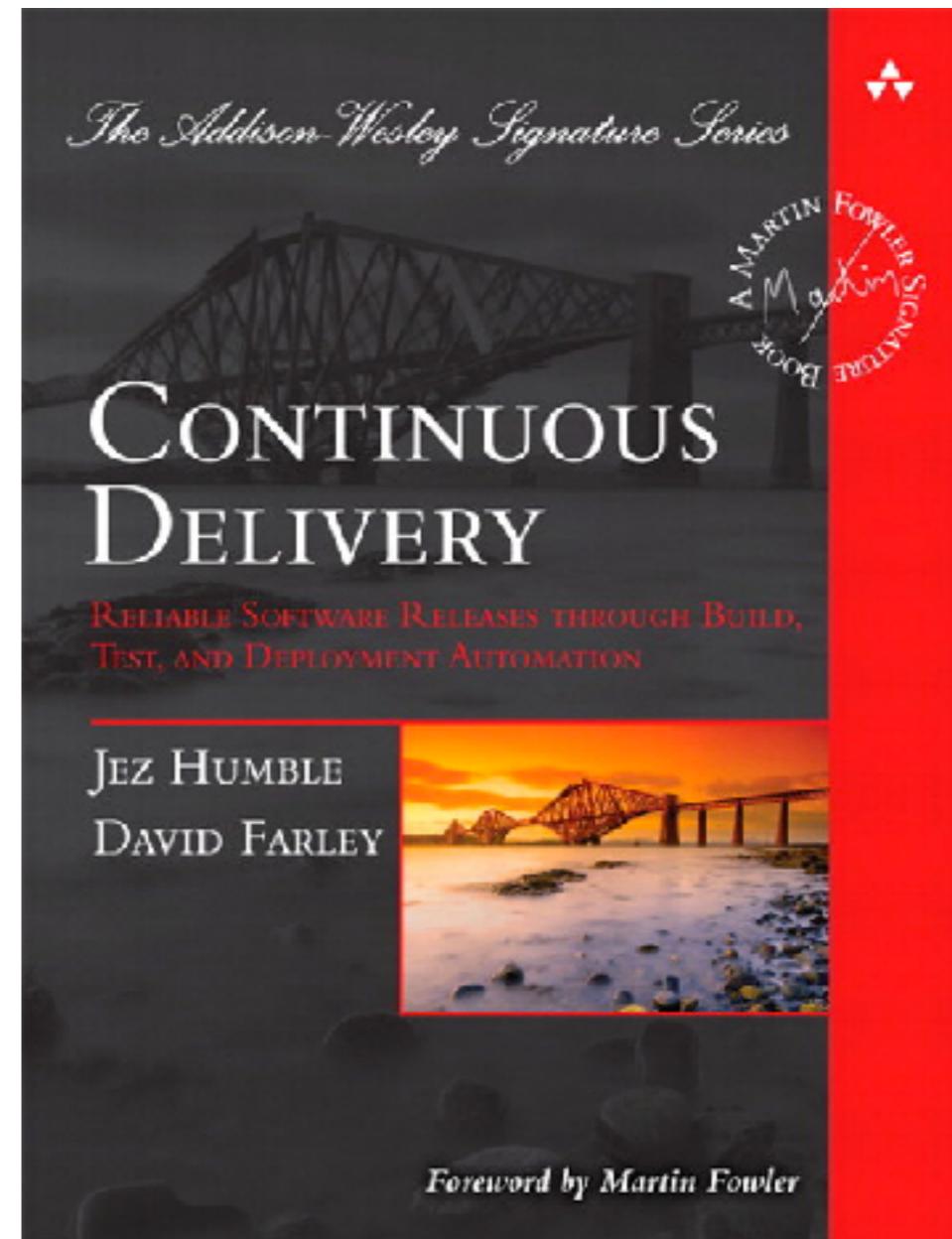
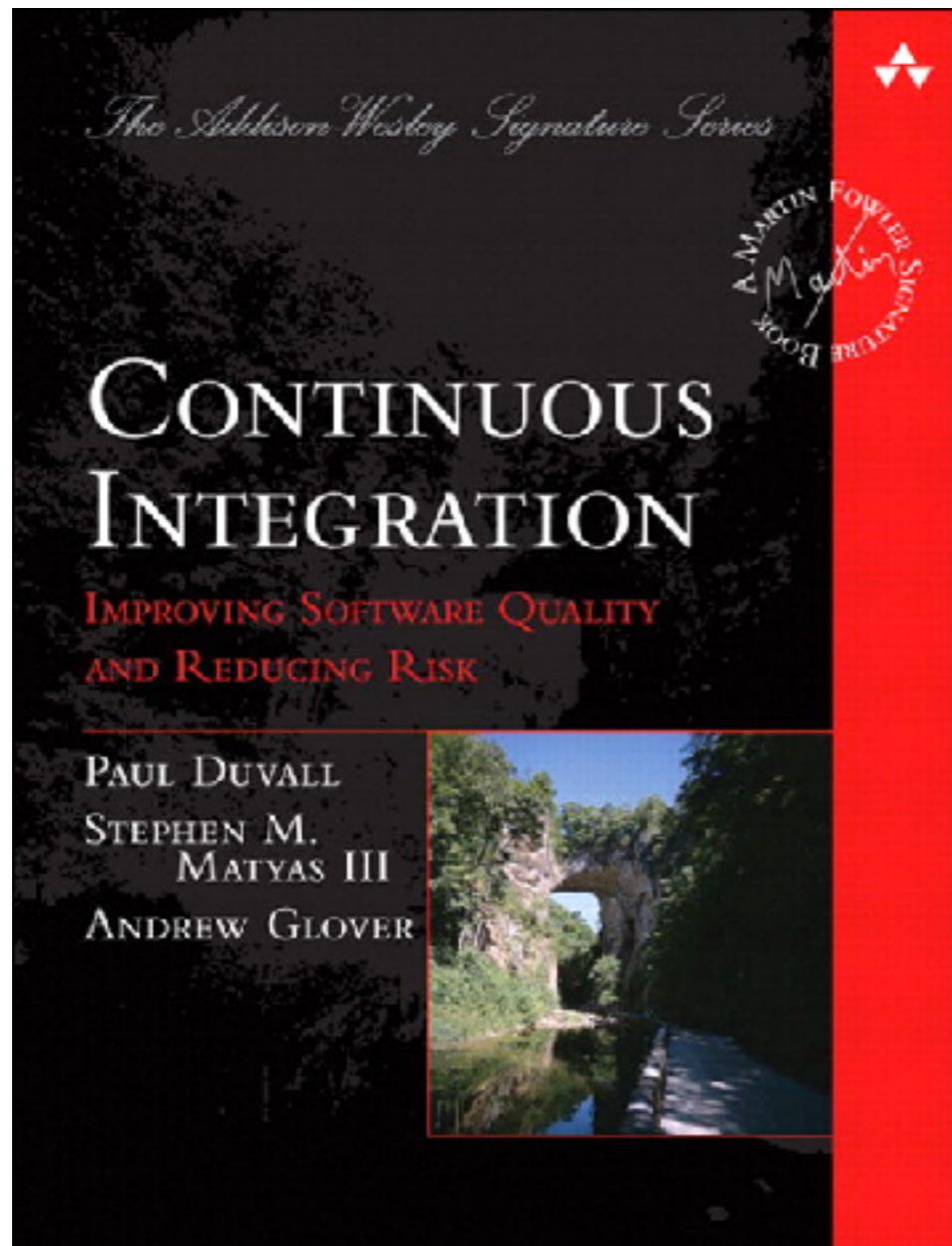
Strive for **fast feedback**



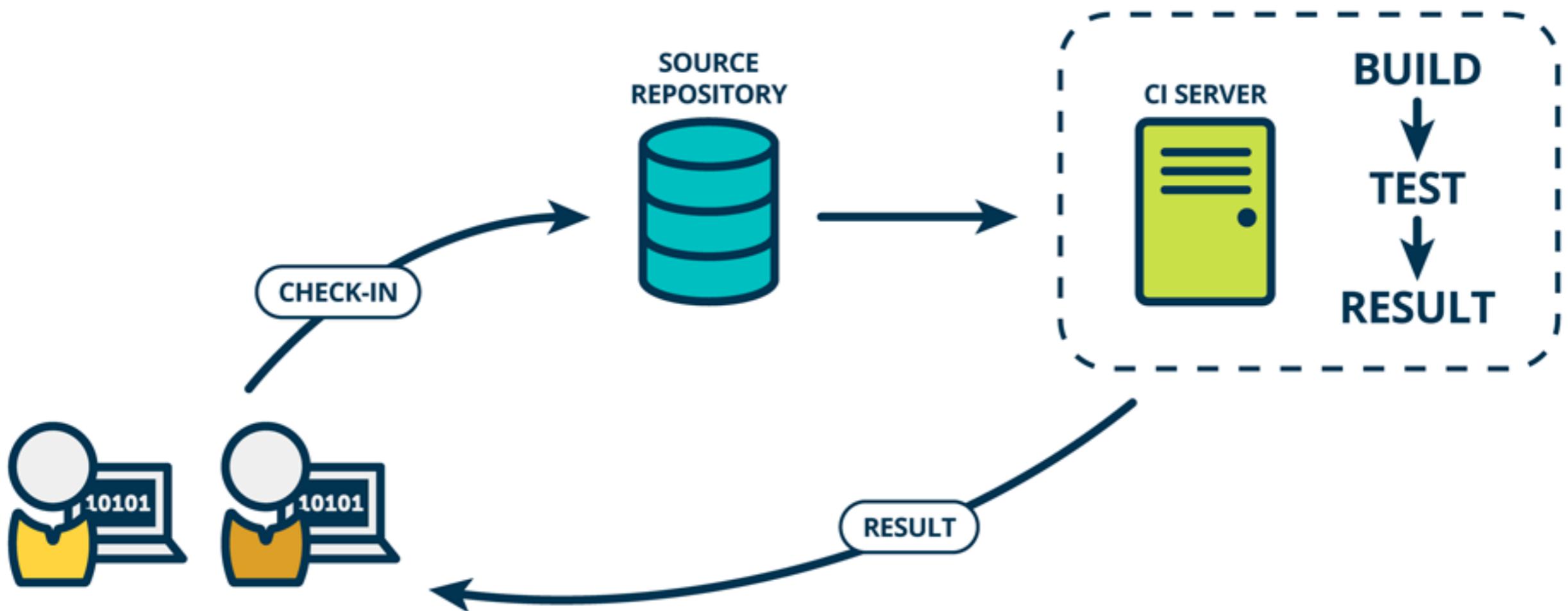
# **Practices of Continuous Integration**



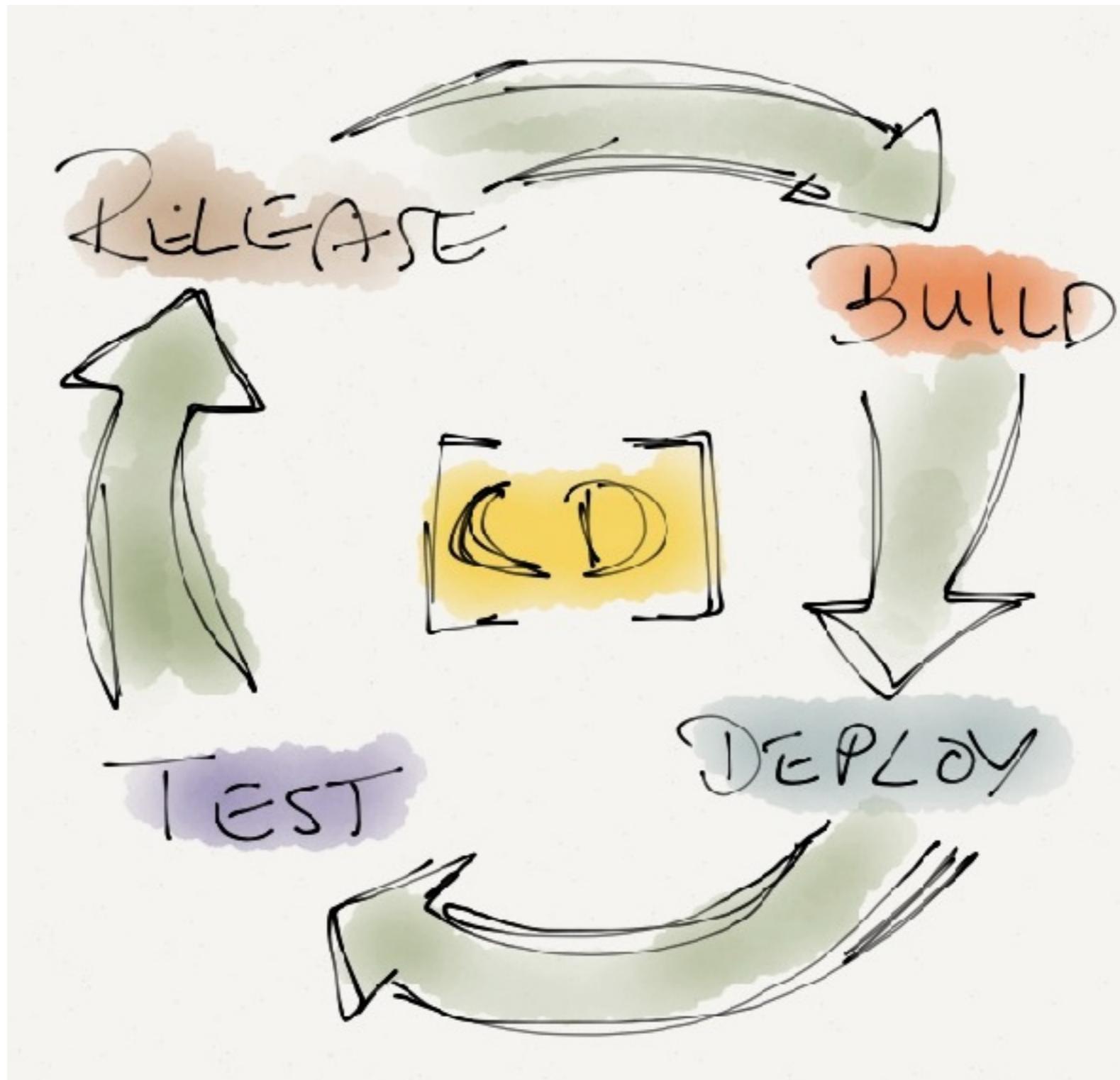
# Improve quality and reduce risk



# Continuous Integration



# CD ?



# CD ?

## CONTINUOUS DELIVERY



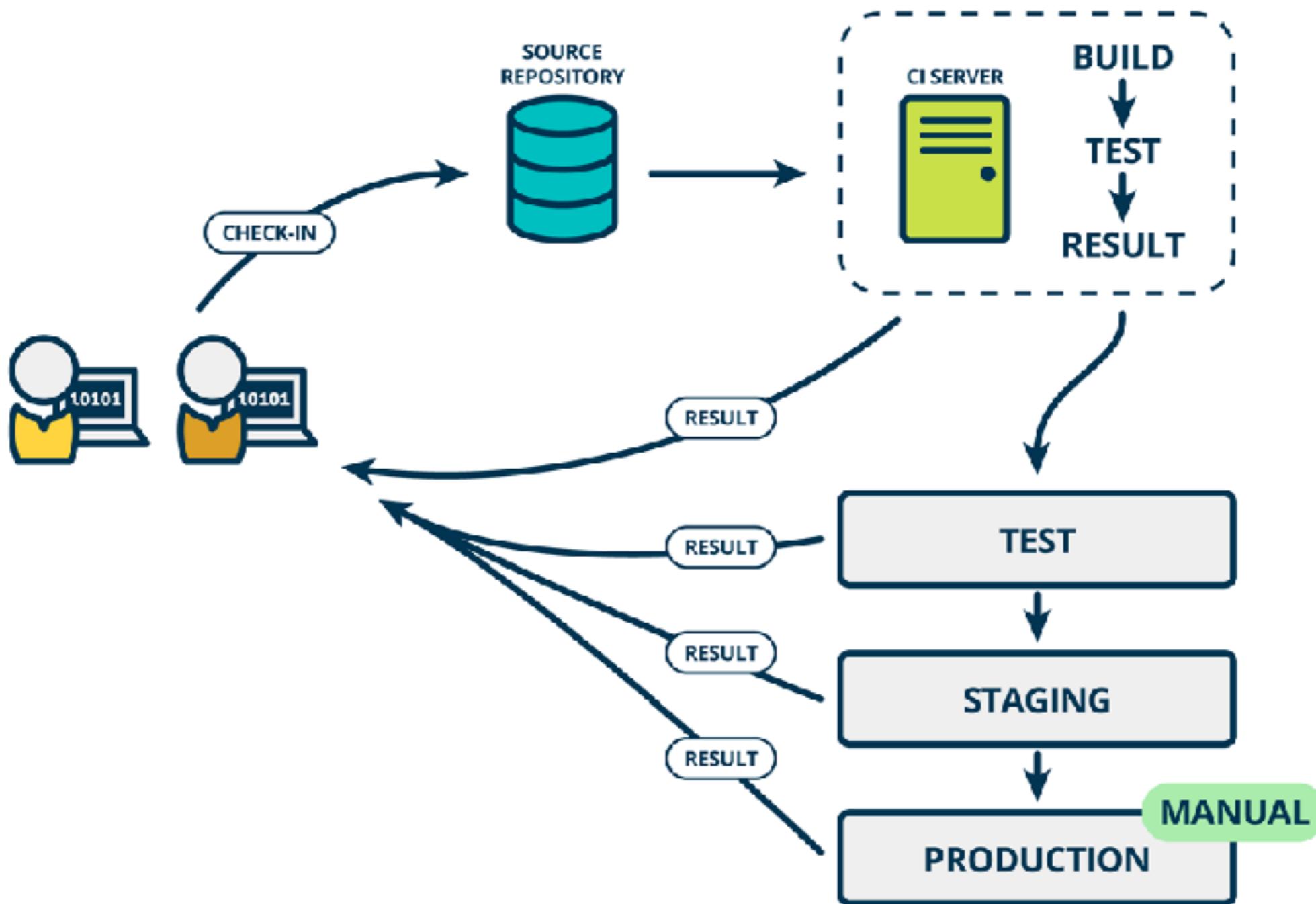
## CONTINUOUS DEPLOYMENT



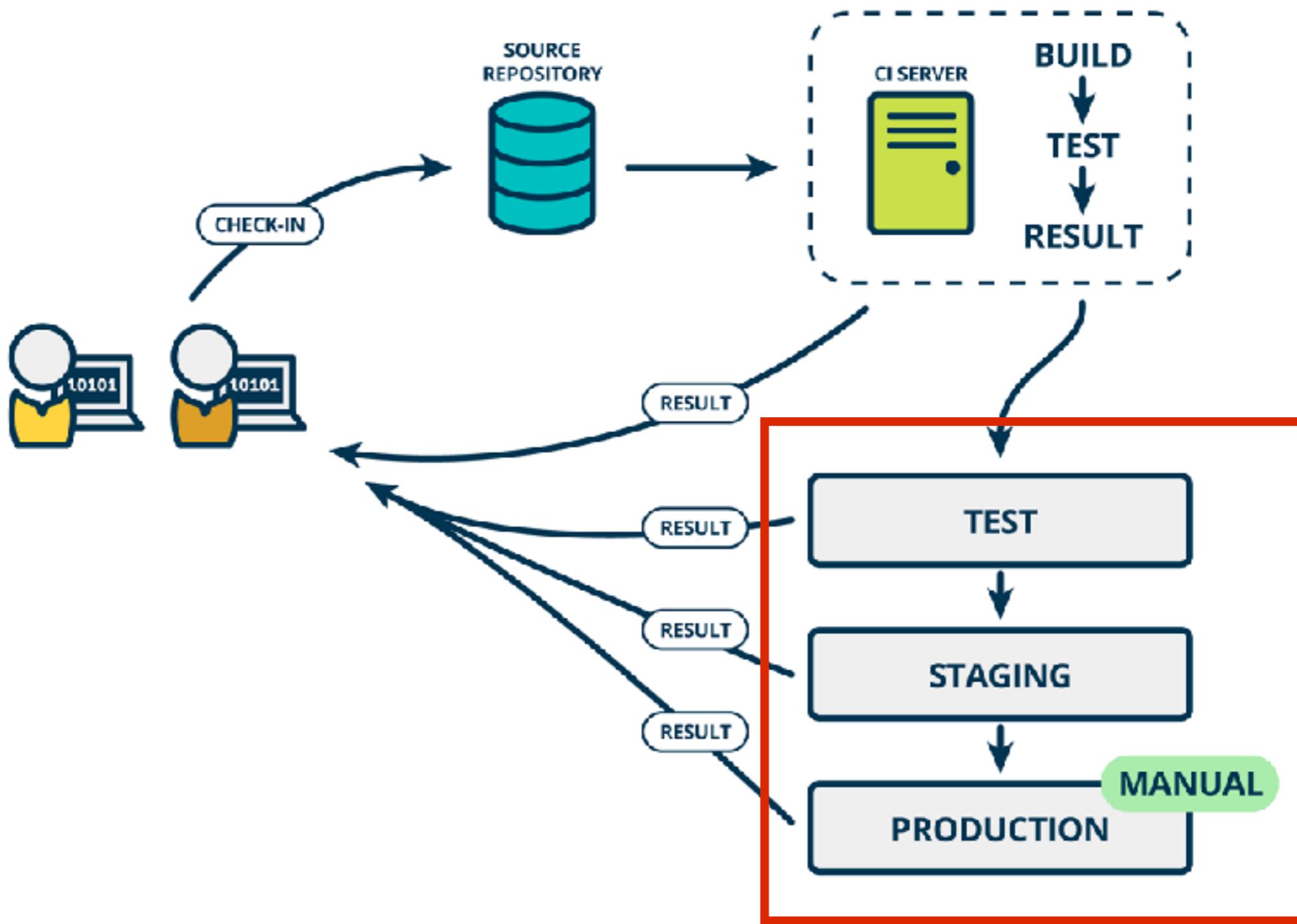
<http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment>



# Continuous Delivery



# Rise of DevOps



# **Continuous Integration**

**is a Software development practices**



# Practice 1

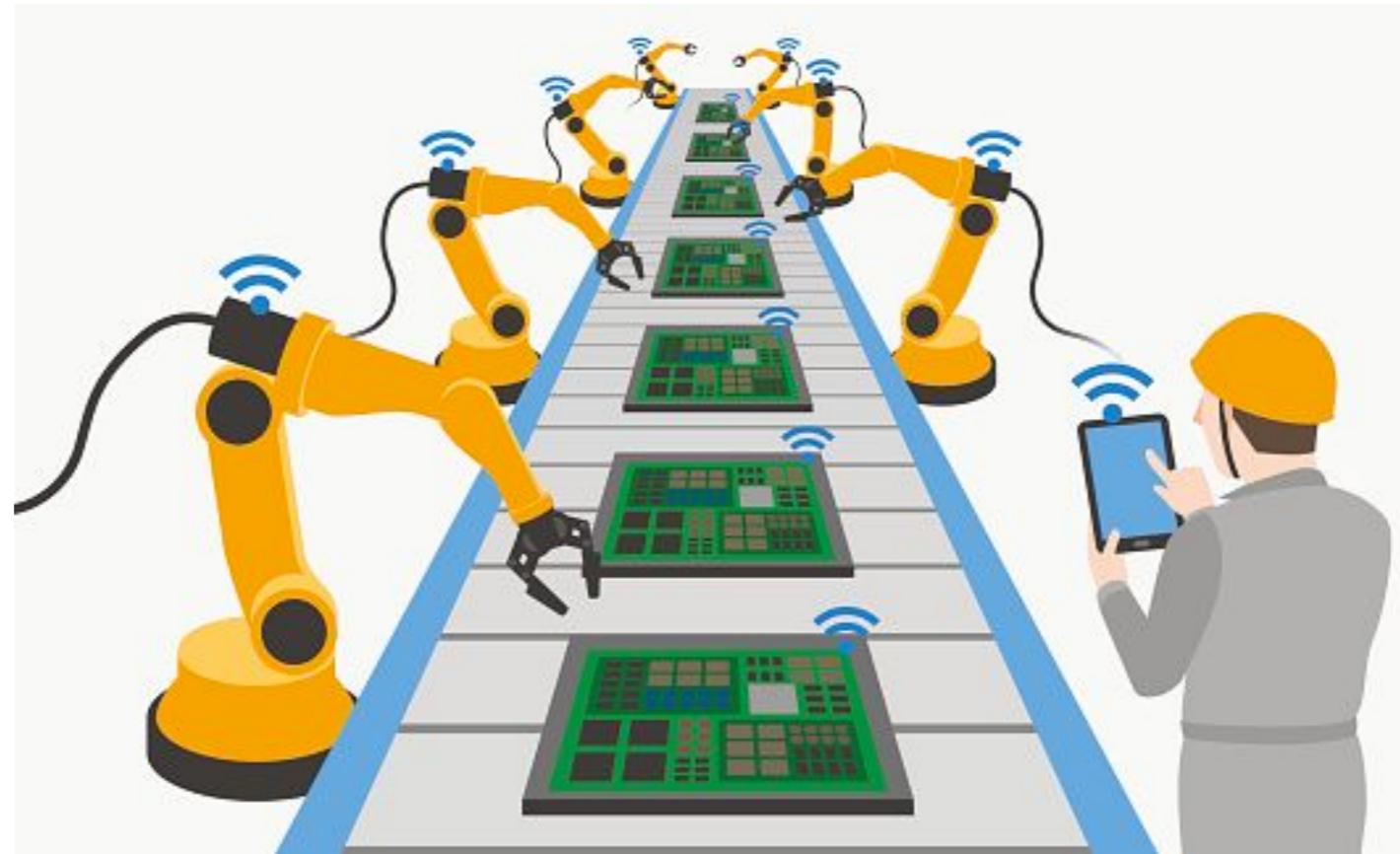
Maintain a single source repository

In general, you should store in source control  
everything you need to build anything



# Practice 2

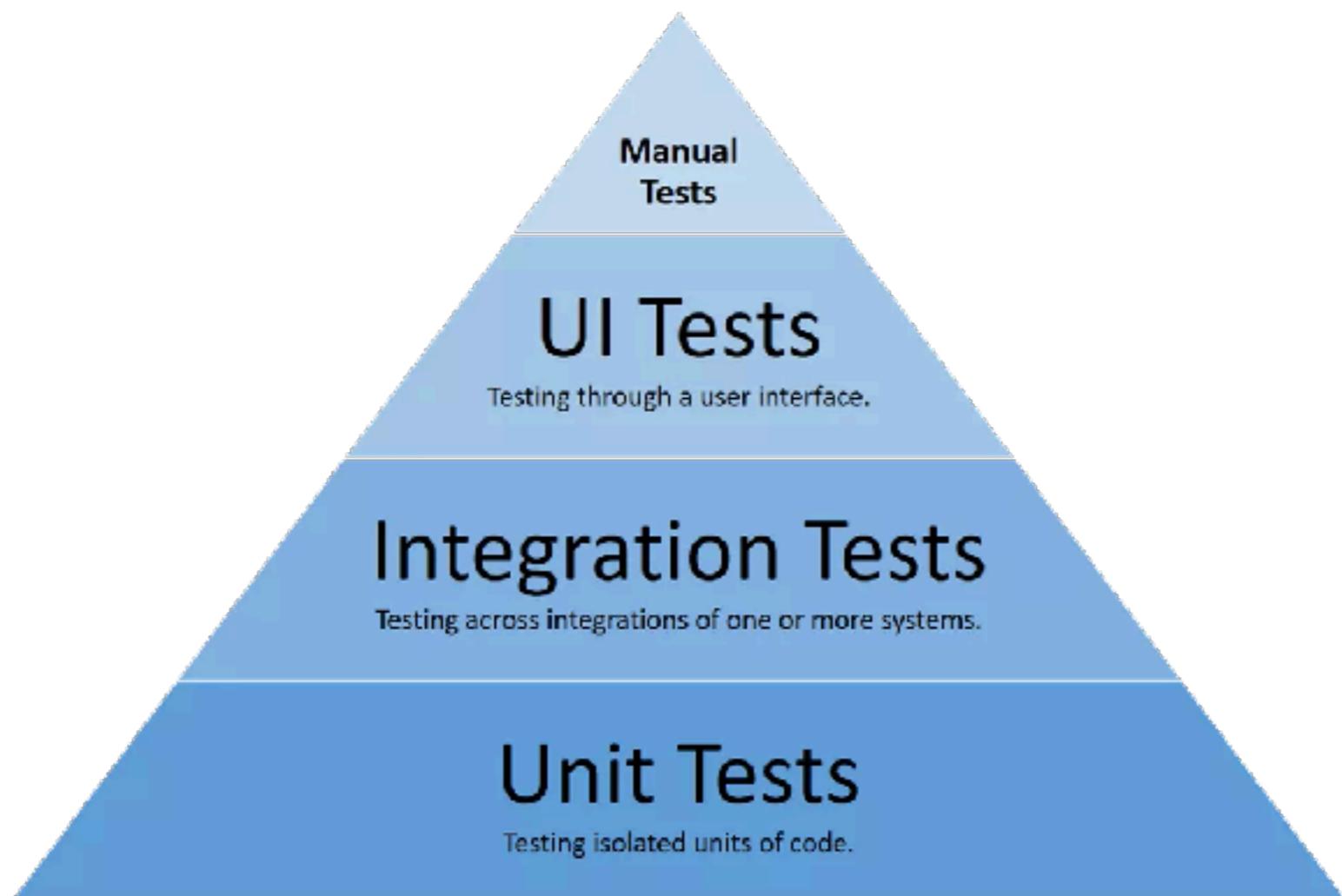
Automated the build  
Automated environment for builds



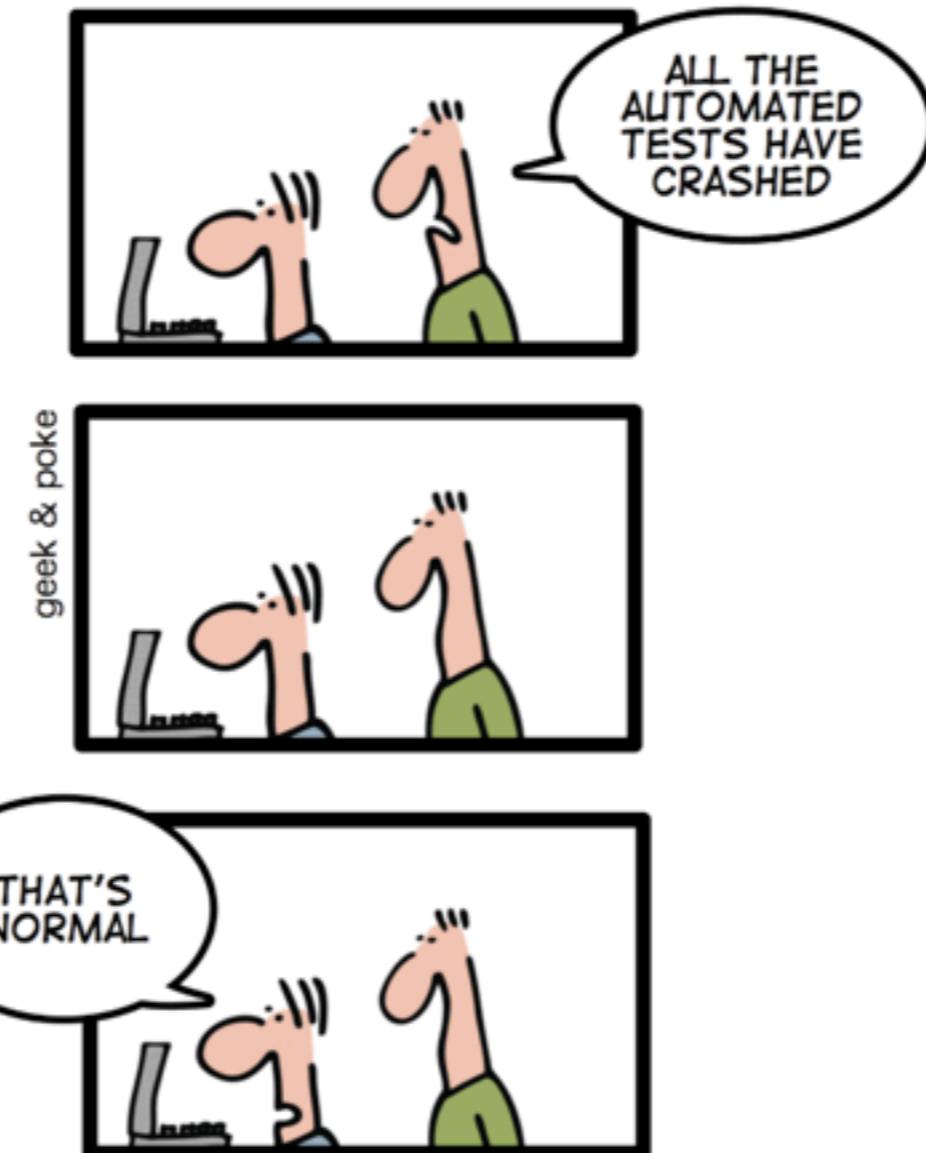
# Practice 3

Make your build **self-testing**

Build process => compile, linking and **testing**



*TODAY: CONTINUOUS INTEGRATION  
GIVES YOU THE COMFORTING  
FEELING TO KNOW THAT  
EVERYTHING IS NORMAL*



<http://geekandpoke.typepad.com/>



DevOps

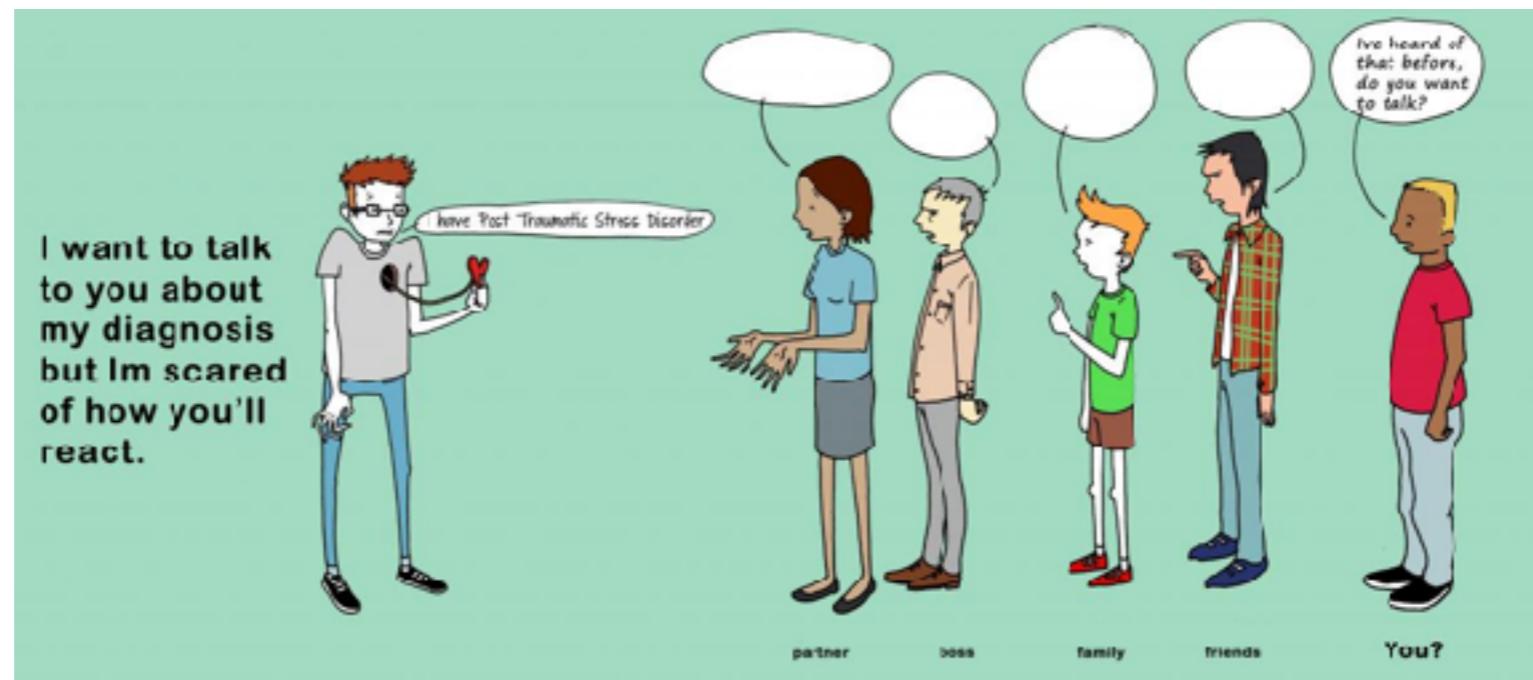
© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

# Practice 4

**Everyone commits to the mainline everyday**

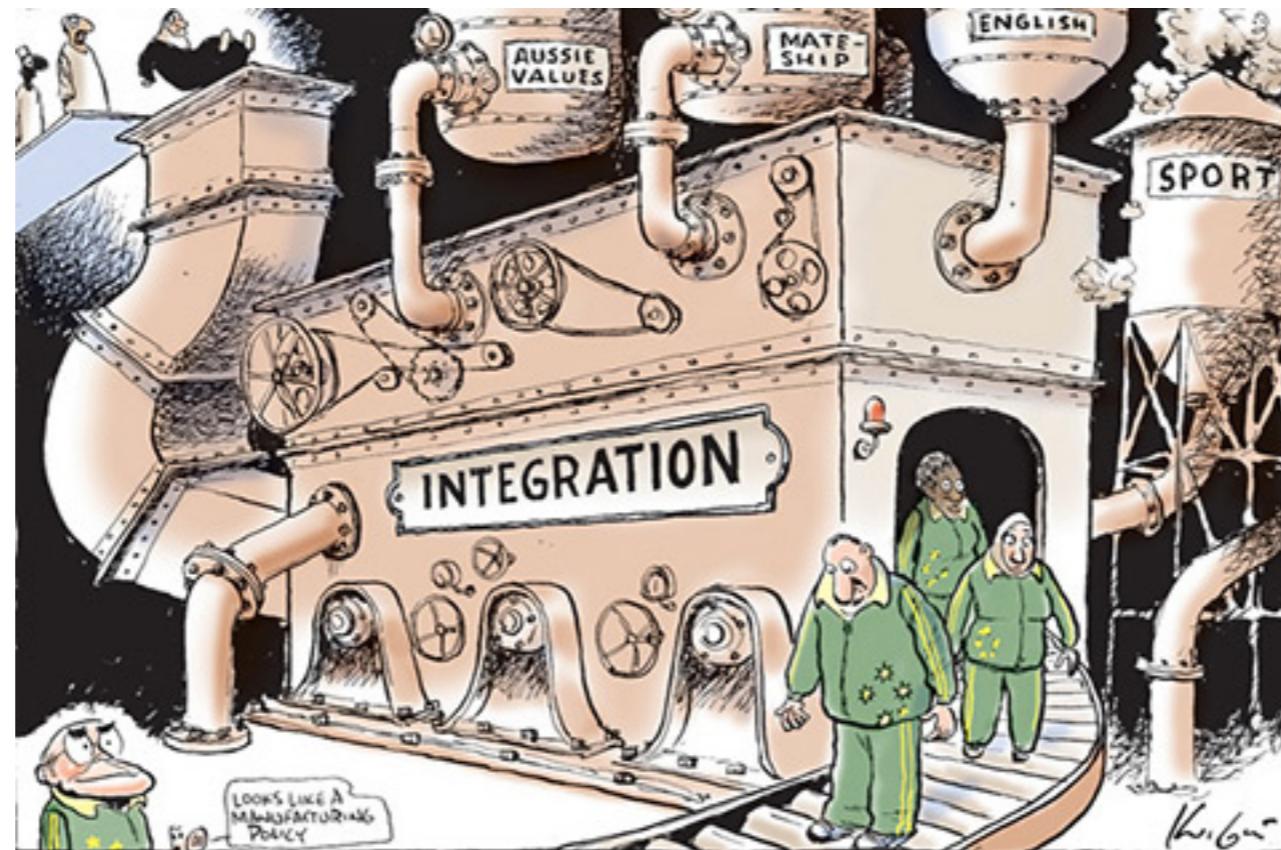
**Integration is about communication**

**Integration allows developers to tell other developers**



# Practice 5

Every commits should build the mainline on an  
**Integration machine**



# Nightly build is not enough for Continuous Integration



# Practice 6

**Fix broken builds immediately**

**“Nobody has a higher priority task than  
fixing the build”**



# Practice 7

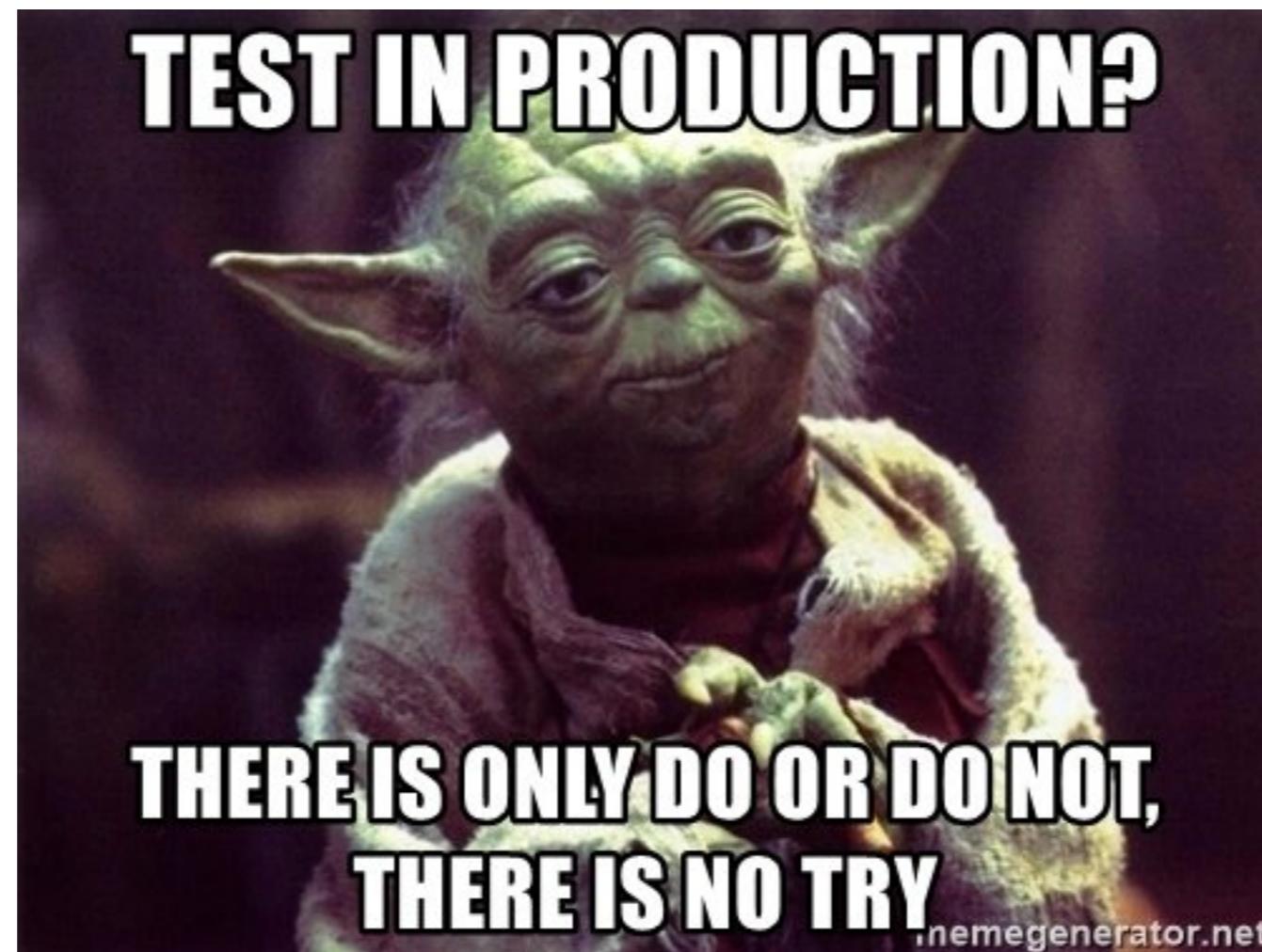
Keep the build **fast**

Continuous Integration is to provide rapid feedback



# Practice 8

Test in clone of the **Production** environment



# Practice 9

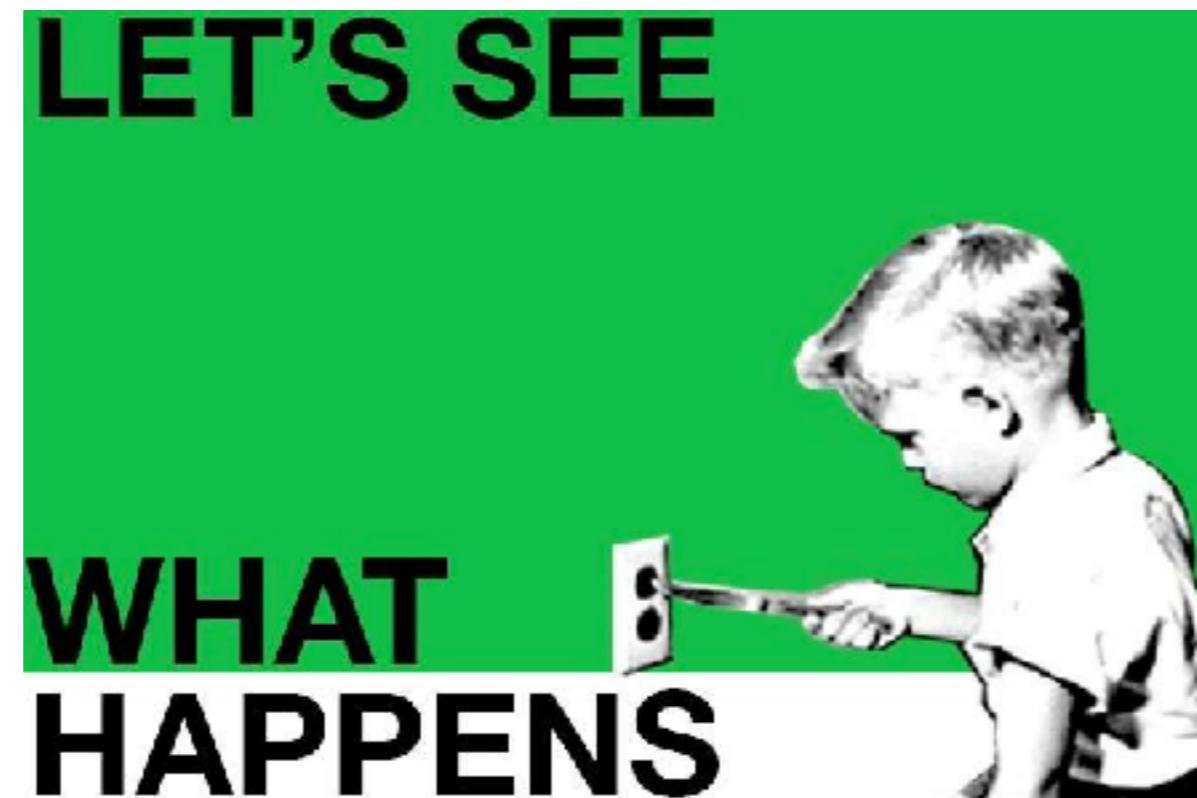
Make it easy for anyone to get  
the latest executable

Make sure well known place where people can find



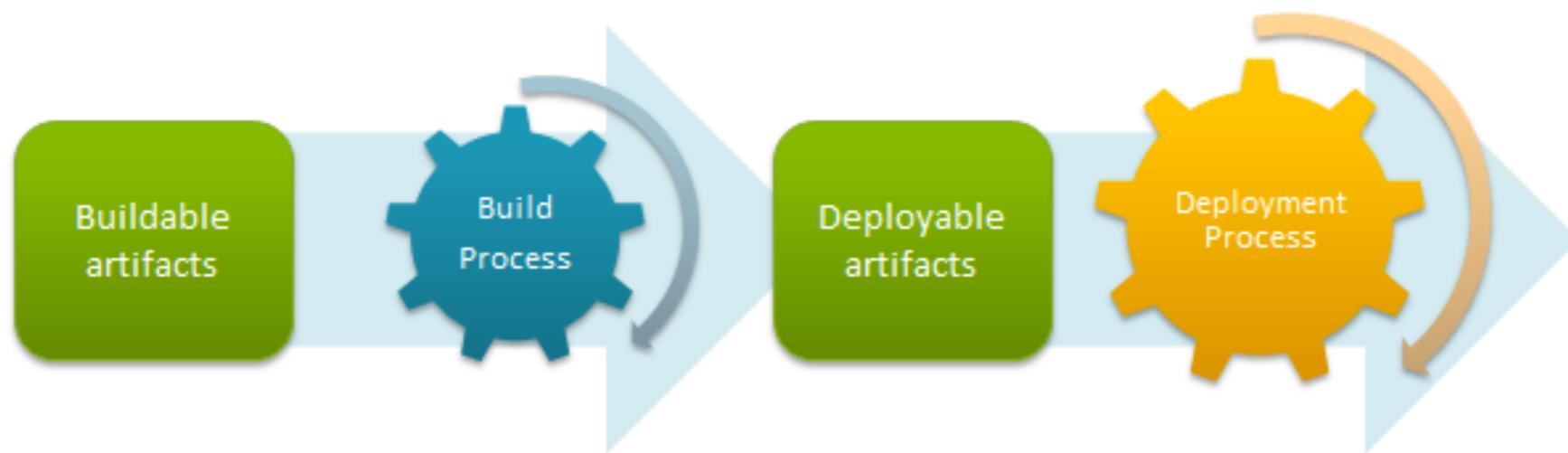
# Practice 10

**Everyone** can see what's happening  
**Easier** to see the state of the system and changes  
Show the good information



# Practice 11

## Automated deployment



# Continuous Delivery



# Continuous Delivery

Use version control for all production artifacts

Automate your deployment process

Implement continuous integration (CI)

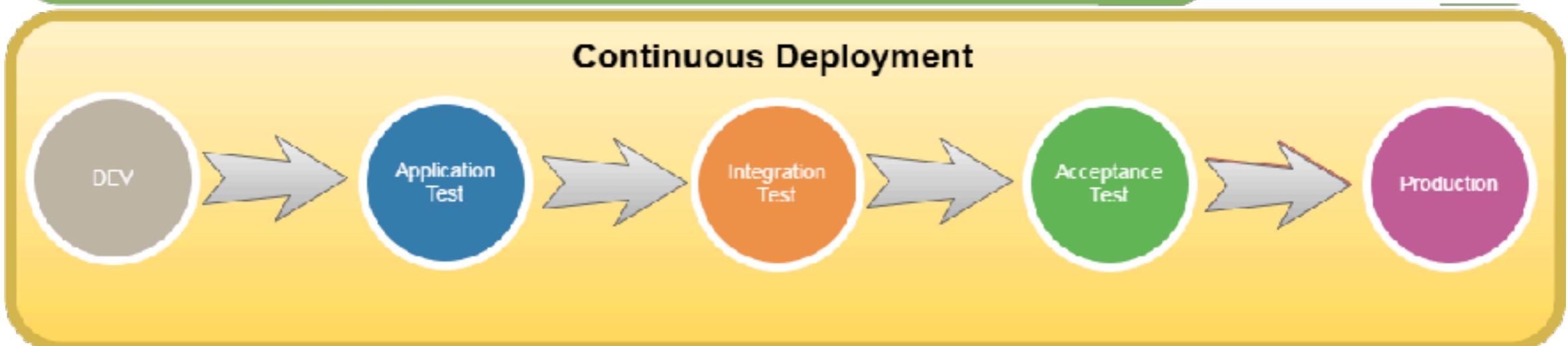
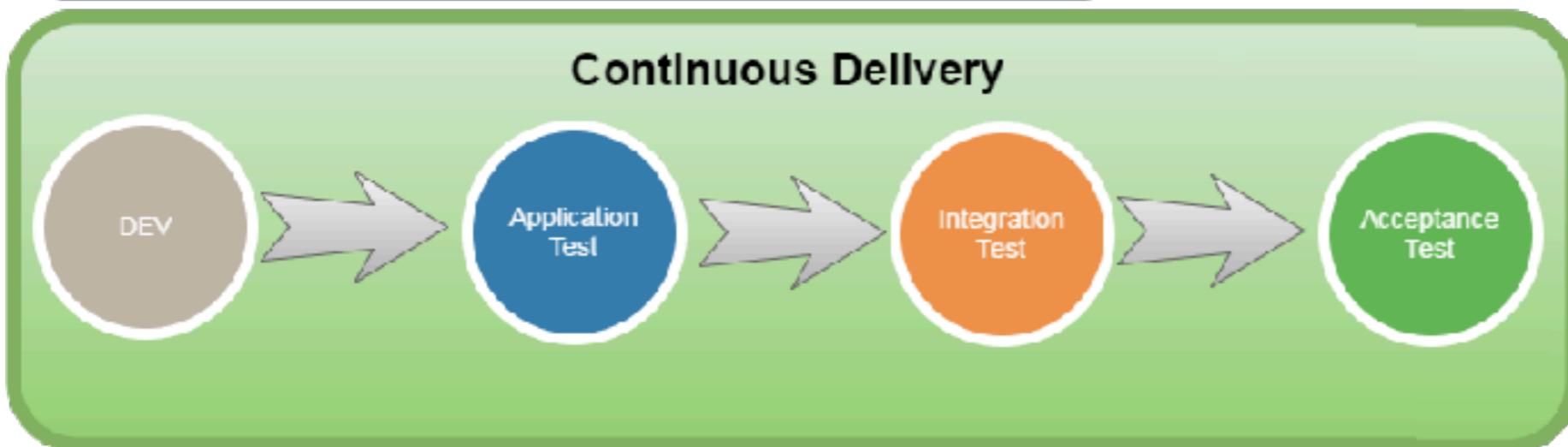
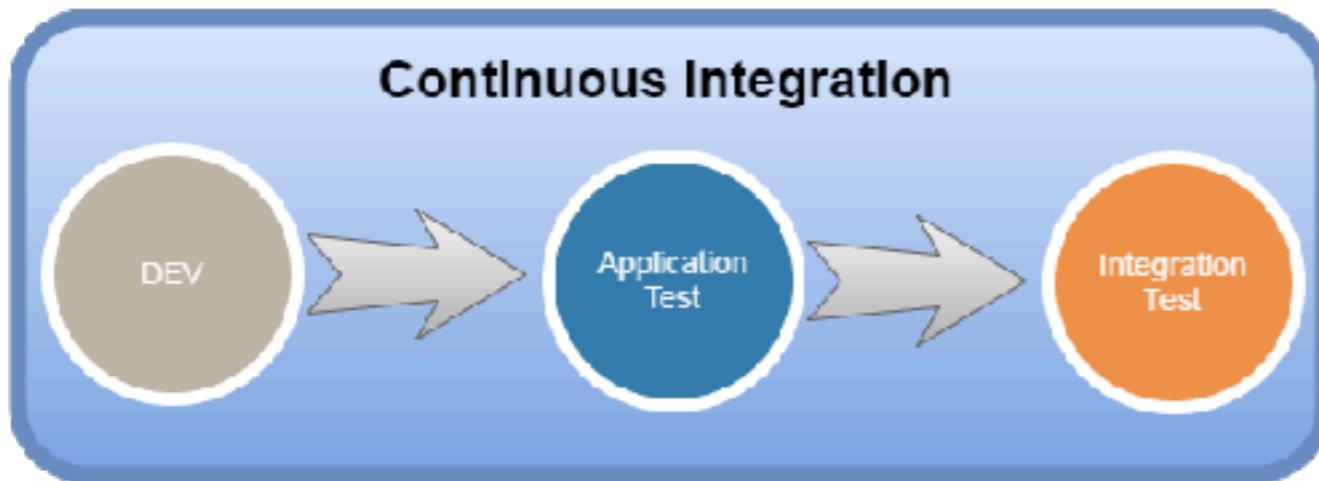
Use trunk-based development methods

Implement test automation

Support test data management

Integrate security into software development process





# How to achieve the CI ?



# 1. Use good version control

Local  
Centralize  
Distributed



VSS = A brown粪便 emoji with three white wavy lines above it, representing a臭屎.

JUST SAY NO!



## 2. Choose Branch strategy

Main only

Development isolation

Feature isolation

Release isolation

Service and Release isolation

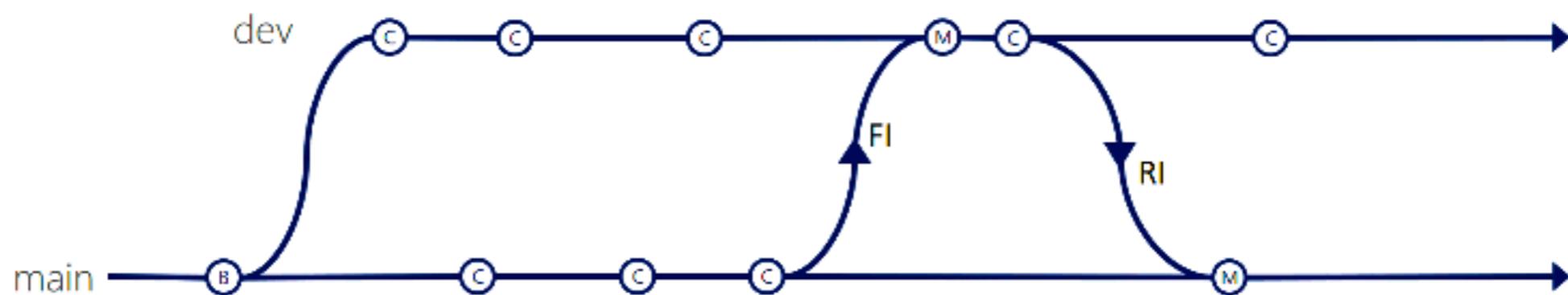
Service, Hotfix and Release isolation



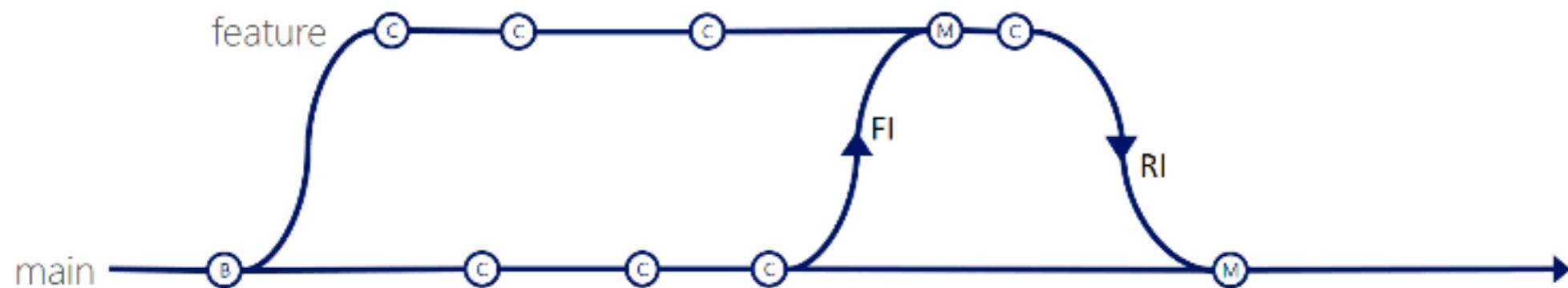
# Main only



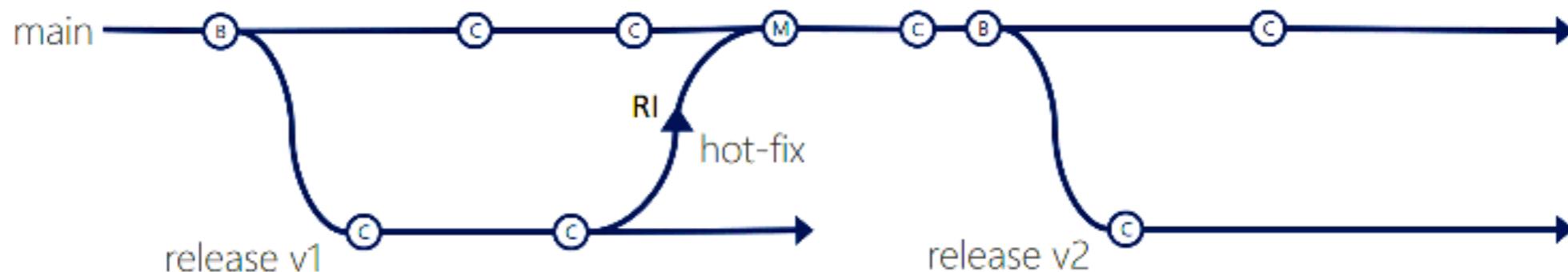
# Development isolation



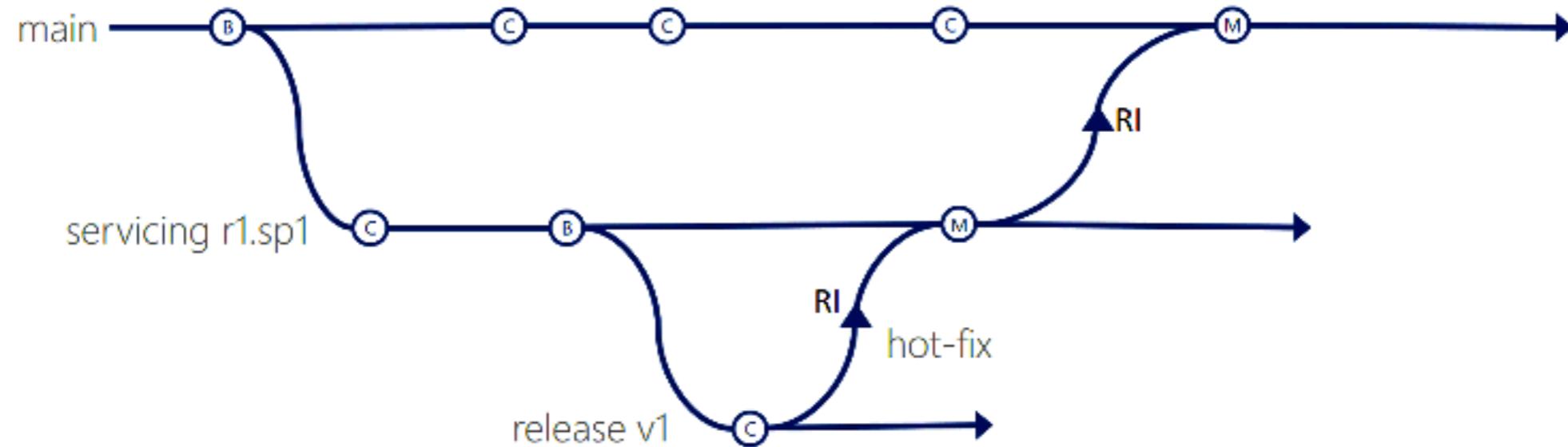
# Feature isolation



# Release isolation



# Service and Release isolation



# Service, Hotfix, Release isolation



# Validate, Validate and Validate

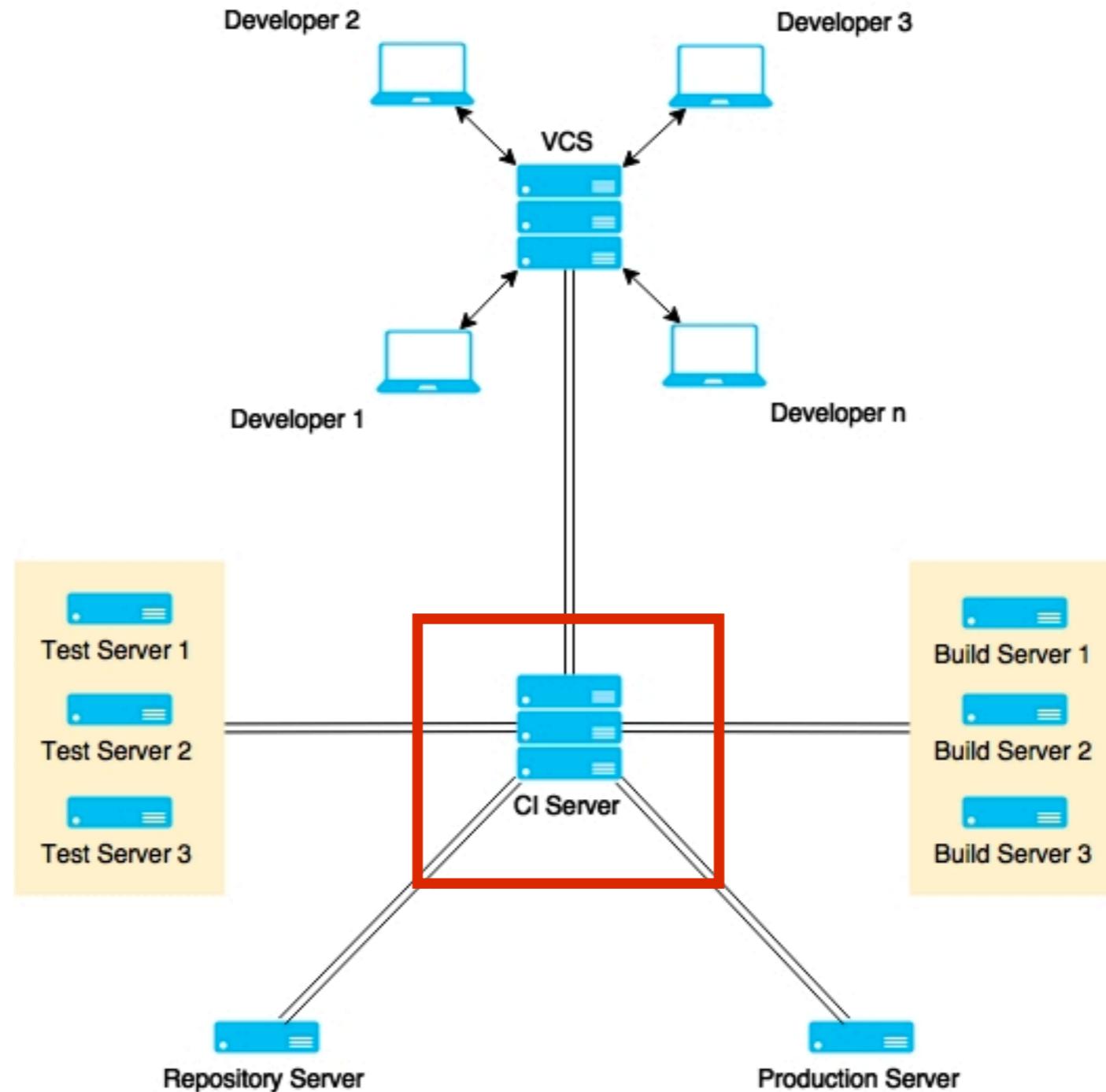


# Suggestion

Keeping branches short-lived, merge conflicts are  
keep to as few as possible



# 3. Use good CI tool





Jenkins

Bamboo



TeamCity

> go<sup>TM</sup>



Hudson



# 4. Use good build tool

- Javascript
  - Gulp, Grunt, Brocolli



- C#/.NET
  - Nant, MSBuild



- Java/JVM
  - Ant, Maven, Gradle, SBT, Leiningen



sbt gradle



# More ...

Use static code analysis  
Automated testing  
Automated deployment  
**People discipline/habit**



**“Behind every successful agile  
project, there is a  
Continuous Integration Server”**



# Anti-pattern for CI Server

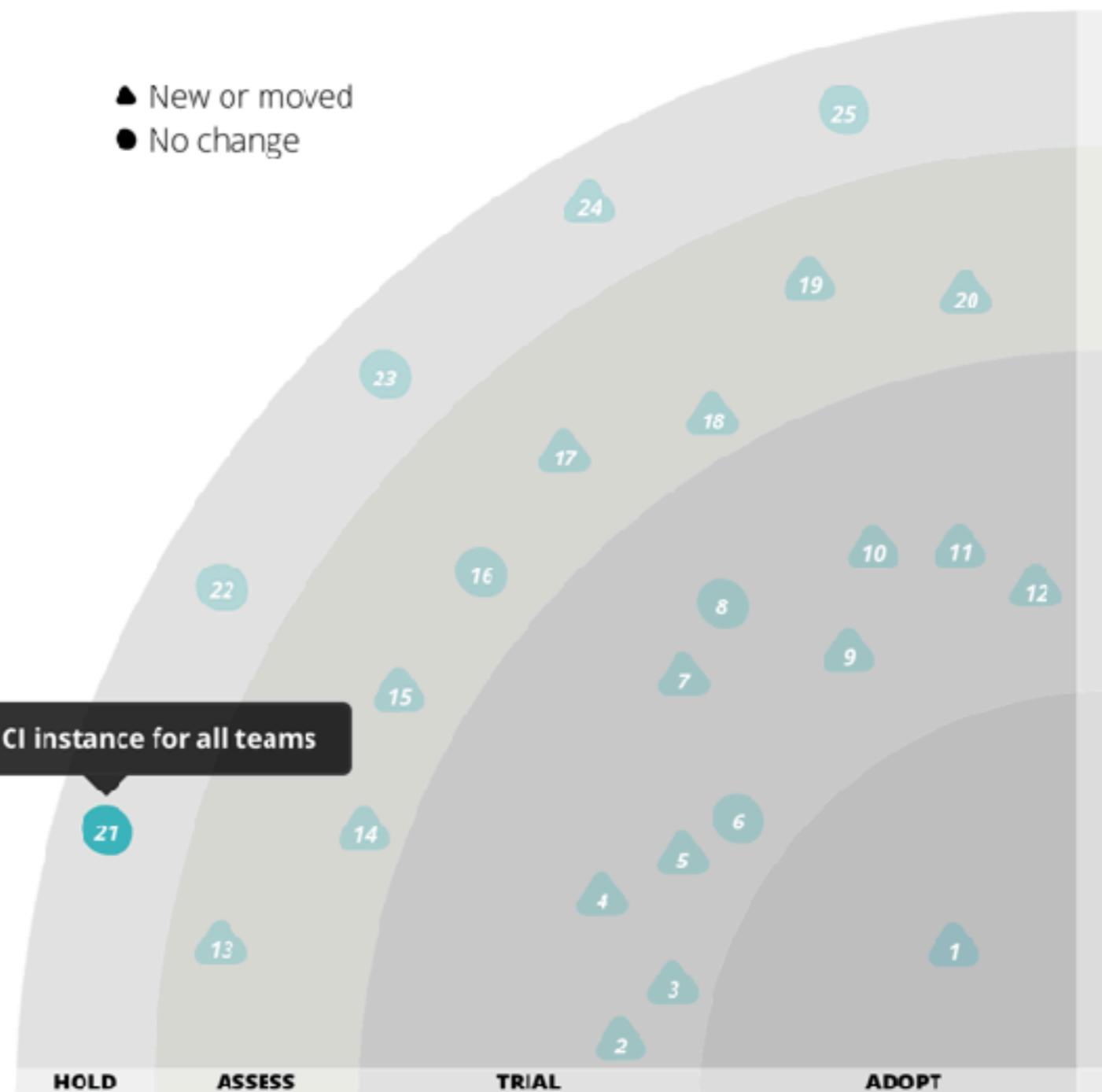
● HOLD ?

## 21. A single CI instance for all teams

We're compelled to caution, again, against creating **a single CI instance for all teams**. While it's a nice idea in theory to consolidate and centralize Continuous Integration (CI) infrastructure, in reality we do not see enough maturity in the tools and products in this space to achieve the desired outcome. Software delivery teams which must use the centralized CI offering regularly have long delays depending on a central team to perform minor configuration tasks, or to troubleshoot problems in the shared infrastructure and tooling. At this stage, we continue to recommend that organizations limit their centralized investment to establishing patterns, guidelines and support for delivery teams to operate their own CI infrastructure.

- ▲ New or moved
- No change

A single CI instance for all teams



<https://www.thoughtworks.com/radar/techniques>



DevOps

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

# Let's start with CI/CD



Application and framework to manage and monitor  
the executable of **repeated tasks**



# Jenkins

<https://jenkins.io/>



# Centralize Continuous Integration Server



# Jenkins

<https://jenkins.io/>



# Why Jenkins ?

Easy !!

Extensible

Scalable

Opensource

Large community

**Lot of plugins**



# Who use Jenkins ?

We thank the following organizations for their major commitments to support the Jenkins project.



Microsoft



redhat.

We thank the following organizations for their support of the Jenkins project through free and/or open source licensing programs.

Atlassian

Datadog

JFrog

Mac Cloud

PagerDuty

XMission

<https://jenkins.io/>



# Download Jenkins



The Jenkins website homepage. At the top is a dark navigation bar with links: Blog, Documentation, Plugins, Use-cases ▾, Participate, Sub-projects ▾, and Resources ▾. The main title "Jenkins" is in large, bold, black font. Below it is the tagline "Build great things at any scale". A descriptive paragraph follows: "The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project." At the bottom right are two buttons: "Documentation" (white background) and "Download" (red background).

<https://jenkins.io/>



# Use Long Term Support (LTS)

## Getting started with Jenkins

The Jenkins project produces two release lines, LTS and weekly. Depending on your organization's needs, one may be preferred over the other.

Both release lines are distributed as `.war` files, native packages, installers, and Docker containers.

### Long-term Support (LTS)

LTS (Long-Term Support) releases are chosen every 12 weeks from the stream of regular releases as the stable release for that time period. [Learn more...](#)

[Changelog](#) | [Upgrade Guide](#) | [Past Releases](#)

[Deploy Jenkins 2.46.3](#)

 [Deploy to Azure](#)

[Download Jenkins 2.46.3 for:](#)

Docker

FreeBSD

### Weekly

A new release is produced weekly to deliver bug fixes and features to users and plugin developers.

[Changelog](#) | [Past Releases](#)

[Download Jenkins 2.65 for:](#)

Arch Linux

Docker

FreeBSD

Gentoo



# Start Jenkins

\$java -jar jenkins.war

Default port of server is **8080**

```
org.eclipse.jetty.server.AbstractConnector doStart
ector@3e2fc448{HTTP/1.1,[http/1.1]}{0.0.0.0:8080}
org.eclipse.jetty.server.Server doStart
```

```
winstone.Logger logInternal
Engine v4.0 running: controlPort=disabled
jenkins.InitReactorRunner$1 onAttained
:ion
jenkins.InitReactorRunner$1 onAttained
jenkins.InitReactorRunner$1 onAttained
```



# Change port of Jenkins

```
$java -jar jenkins.war --httpPort=<port>
```



# Open in browser

<http://localhost:8080>

Getting Started

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/Users/somkiat/data/slide/ci-cd/swpark/software/keep/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue



# Copy password from console

```
*****  
*****  
*****
```

Jenkins initial setup is required. An admin user has been created.

Please use the following password to proceed to installation:

a4b3a5231b8048419192d0c5afd3fce8

This may also be found at: /Users/somkiat/data/slide/ci-cd/swp/initialAdminPassword

```
*****  
*****  
*****
```



# Customize plugins

Getting Started



## Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

### Install suggested plugins

Install plugins the Jenkins community finds most useful.

### Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.46.3



DevOps

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

# Waiting

## Getting Started

## Getting Started

<input type="radio"/> Folders Plugin	<input type="radio"/> OWASP Markup Formatter Plugin	<input type="radio"/> build timeout plugin	<input type="radio"/> Credentials Binding Plugin
<input type="radio"/> Timestamper	<input type="radio"/> Workspace Cleanup Plugin	<input type="radio"/> Ant Plugin	<input type="radio"/> Gradle Plugin
<input type="radio"/> Pipeline	<input type="radio"/> GitHub Organization Folder Plugin	<input type="radio"/> Pipeline: Stage View Plugin	<input type="radio"/> Git plugin
<input type="radio"/> Subversion Plug-in	<input type="radio"/> SSH Slaves plugin	<input type="radio"/> Matrix Authorization Strategy Plugin	<input type="radio"/> PAM Authentication plugin
<input type="radio"/> LDAP Plugin	<input type="radio"/> Email Extension Plugin	<input type="radio"/> Mailer Plugin	

\*\* - required dependency

Jenkins 2.46.3



# Success

Getting Started

## Installation Failures

Some plugins failed to install properly, you may retry installing them or continue with

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	✓ build timeout plugin	✓ Credentials Binding Plugin
✓ Timestamper	✓ Workspace Cleanup Plugin	✓ Ant Plugin	✓ Gradle Plugin
✓ Pipeline	✓ GitHub Organization Folder Plugin	✓ Pipeline: Stage View Plugin	✓ Git plugin
✓ Subversion Plug-in	✓ SSH Slaves plugin	✓ Matrix Authorization Strategy Plugin	✓ PAM Authentication plugin
✓ LDAP Plugin	✓ Email Extension Plugin	✓ Mailer Plugin	

Jenkins 2.46.3

[Continue](#)

[Retry](#)



# Create a new user

Getting Started

## Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.46.3

[Continue as admin](#)

[Save and Finish](#)



# Ready to use

Getting Started

## Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

---

Jenkins 2.46.3



# Welcome to Jenkins

Jenkins  somkiat | log out

ENABLE AUTO REFRESH

New Item  add description 

People 

Build History 

Manage Jenkins 

My Views 

Credentials 

Build Queue  -  
No builds in the queue.

Build Executor Status  -  
1 Idle  
2 Idle

Please [create new jobs](#) to get started.

Page generated: Jun 14, 2017 2:08:57 PM ICT [REST API](#) [Jenkins ver. 2.46.3](#)



# About Jenkins's HOME

Default of **JENKINS\_HOME** is  
`<path of user>/jenkins`



# About Jenkins's HOME

## Data in JENKINS\_HOME

```
/Users/somkiat/.jenkins
├── config.xml
├── failed-boot-attempts.txt
├── hudson.model.UpdateCenter.xml
├── jenkins.CLI.xml
├── jenkins.install.UpgradeWizard.state
├── jobs
├── logs
├── nodeMonitors.xml
├── nodes
├── plugins
├── queue.xml
├── queue.xml.bak
├── secret.key
├── secret.key.not-so-secret
├── secrets
├── updates
├── userContent
├── users
└── war
```



# About Jenkins's HOME

File and Folder name	Description
config.xml	All about configuration
jobs	Keep all jobs/project
plugins	Keep all plugins
nodes	Keep all nodes
logs	Keep all logs



# Change Jenkins's HOME

## For Windows

```
set JENKINS_HOME=<your path>
```

## For Linux/Mac

```
export JENKINS_HOME=<your path>
```

try to restart Jenkins ...



# **Disable Jenkins's security**



# Set useSecurity=false

# <JENKINS HOME>/config.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<hudson>
    <disabledAdministrativeMonitors/>
    <version>2.89.3</version>
    <numExecutors>2</numExecutors>
    <mode>NORMAL</mode>
    <useSecurity>false</useSecurity>
    <authorizationStrategy class="hudson.security.LoggedInAuthorizationStrategy">
        <denyAnonymousReadAccess>true</denyAnonymousReadAccess>
    </authorizationStrategy>
```



# **Learn to use Jenkins in the right way**



# Manage Jenkins

For Administrator to config anything in Jenkins

Jenkins

New Item

People

Build History

Manage Jenkins

My views

Credentials

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

search

## Manage Jenkins

- [Configure System](#)  
Configure global settings and paths.
- [Configure Global Security](#)  
Secure Jenkins; define who is allowed to access/use the system.
- [Configure Credentials](#)  
Configure the credential providers and types
- [Global Tool Configuration](#)  
Configure tools, their locations and automatic installers.
- [Reload Configuration from Disk](#)  
Discard all the loaded data in memory and reload everything from file system. Useful when you modify configuration files.
- [Manage Plugins](#)  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- [System Information](#)  
Displays various environmental information to assist trouble-shooting.
- [System Log](#)  
System log captures output from java.util.logging output related to Jenkins.
- [Load Statistics](#)  
Check your resource utilization and see if you need more computers for your builds.



# Configure System

## Global setting and paths

Jenkins  

New Item 

People 

Build History 

Manage Jenkins 

My Views 

Credentials 

New View 

**Build Queue**   
No builds in the queue.

**Build Executor Status**   
1 Idle  
2 Idle

### Manage Jenkins

-  [Configure System](#)  
Configure global settings and paths.
-  [Configure Global Security](#)  
Secure Jenkins; define who is allowed to access/use the system.
-  [Configure Credentials](#)  
Configure the credential providers and types
-  [Global Tool Configuration](#)  
Configure tools, their locations and automatic installers.
-  [Reload Configuration from Disk](#)  
Discard all the loaded data in memory and reload everything from file system. Useful when you modify configuration files.
-  [Manage Plugins](#)  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
-  [System Information](#)  
Displays various environmental information to assist trouble-shooting.
-  [System Log](#)  
System log captures output from `java.util.logging` related to Jenkins.
-  [Load Statistics](#)  
Check your resource utilization and see if you need more computers for your builds.



# Configure System

JENKINS Home

# of executors

Label name of node

Environment variables

Email notification



# Configure Global Security

## Setting for secure Jenkins

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Credentials', and 'New View'. Below that are two sections: 'Build Queue' (empty) and 'Build Executor Status' (2 Idle). The main area is titled 'Manage Jenkins' and contains several configuration options: 'Configure System' (Configure global settings and paths), 'Configure Global Security' (Secure Jenkins; define who is allowed to access/use the system, this link is highlighted with a red box), 'Configure Credentials' (Configure the credential providers and types), 'Global Tool Configuration' (Configure tools, their locations and automatic installers), 'Reload Configuration from Disk' (Discard all the loaded data in memory and reload everything from file system. Useful when you modify configuration files), 'Manage Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), 'System Information' (Displays various environmental information to assist trouble-shooting), 'System Log' (System log captures output from java.util.logging output related to Jenkins), and 'Load Statistics' (Check your resource utilization and see if you need more computers for your builds).



# Global Tool Configuration

Config tools, location and automatic installers

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like New Item, People, Build History, Manage Jenkins (which is selected), My Views, Credentials, and New View. Below that are sections for Build Queue (empty) and Build Executor Status (2 Idle). The main content area is titled "Manage Jenkins" and lists several configuration options: Configure System, Configure Global Security, Configure Credentials, Global Tool Configuration (which is highlighted with a red box), Reload Configuration from Disk, Manage Plugins, System Information, System Log, and Load Statistics.

Link	Description
<a href="#">Configure System</a>	Configure global settings and paths.
<a href="#">Configure Global Security</a>	Secure Jenkins; define who is allowed to access/use the system.
<a href="#">Configure Credentials</a>	Configure the credential providers and types
<a href="#">Global Tool Configuration</a>	Configure tools, their locations and automatic installers.
<a href="#">Reload Configuration from Disk</a>	Discard all the loaded data in memory and reload everything from file system. Useful when you modify configuration files.
<a href="#">Manage Plugins</a>	Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
<a href="#">System Information</a>	Displays various environmental information to assist trouble-shooting.
<a href="#">System Log</a>	System log captures output from <code>java.util.logging</code> related to Jenkins.
<a href="#">Load Statistics</a>	Check your resource utilization and see if you need more computers for your builds.



# Global Tool Configuration

Apache Maven  
JDK (Java Development Kit)  
Git  
Gradle  
Docker



# Manage Plugins

Add, remove, enable/disable plugins

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like New Item, People, Build History, Manage Jenkins (which is selected and highlighted in blue), My Views, Credentials, and New View. Below that are sections for Build Queue (empty) and Build Executor Status (2 Idle). The main content area is titled "Manage Jenkins" and lists several configuration options: Configure System, Configure Global Security, Configure Credentials, Global Tool Configuration, Reload Configuration from Disk, Manage Plugins (which is highlighted with a red box), System Information, System Log, and Load Statistics.

**Manage Jenkins**

- [Configure System](#)  
Configure global settings and paths.
- [Configure Global Security](#)  
Secure Jenkins: define who is allowed to access/use the system.
- [Configure Credentials](#)  
Configure the credential providers and types
- [Global Tool Configuration](#)  
Configure tools, their locations and automatic installers.
- [Reload Configuration from Disk](#)  
Discard all the loaded data in memory and reload everything from file system. Useful when you modi
- [Manage Plugins](#)  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- [System Information](#)  
Displays various environmental information to assist trouble-shooting.
- [System Log](#)  
System log captures output from `java.util.logging` related to Jenkins.
- [Load Statistics](#)  
Check your resource utilization and see if you need more computers for your builds.



# Manage Plugins

Add, remove, enable/disable plugins

				Filter: <input type="text"/>
Updates	Available	Installed	Advanced	
Install	Name ↓	Version	Installed	
No updates				

Update information obtained: 14 hr ago

[Check now](#)

Select: [All](#), [None](#)

This page lists updates to the plugins you currently use.



# Manage Plugins

## Filter plugins

Filter:

**Updates** Available Installed Advanced

Install	Name ↓	Version	Installed
No updates			

Update information obtained: 14 hr ago [Check now](#)

Select: [All](#), [None](#)

This page lists updates to the plugins you currently use.



# Finds Jenkins's plugin

Jenkins

Blog Documentation Plugins Use-cases Participate Sub-projects Resources

# Plugins Index

Discover the 1000+ community contributed Jenkins plugins to support building, deploying and automating any project.

Browse Find plugins...

<https://plugins.jenkins.io/>



# Try to install a first plugin

Choose Available tab and select a plugin

The screenshot shows a plugin management interface with a top navigation bar featuring tabs: Updates, Available (highlighted with a red box), Installed, and Advanced. Below the tabs is a search bar labeled "Filter:" with a magnifying glass icon. The main content area displays a table of available plugins under the ".NET Development" category. The table has columns for Name and Version. Each plugin entry includes a checkbox, a brief description, and the version number. At the bottom of the page, there are three buttons: "Install without restart", "Download now and install after restart" (highlighted with a red box), and "Check now". A status message "Update information obtained: 9 hr 37 min ago" is also present.

Name	Version
<a href="#">CCM Plug-in</a> This plug-in generates the trend report for CCM, an open source static code analysis program.	3.1
<a href="#">FxCop Runner plugin</a>	1.1
<a href="#">MSBuild Plugin</a>	1.27
<a href="#">MSTest plugin</a> Generates test reports for MSTest.	0.19
<a href="#">MSTestRunner plugin</a>	1.3.0
<a href="#">NAnt Plugin</a>	1.4.3
<a href="#">NCover.plugin</a>	0.3
<a href="#">PowerShell plugin</a>	1.3
<a href="#">Violation Comments to Bitbucket Server Plugin</a> Finds violations reported by code analyzers and comments Bitbucket Server (or Stash) pull requests (or commits) with them.	1.50
<a href="#">Violations plugin</a>	0.7.11

Install without restart   Download now and install after restart   Check now

Update information obtained: 9 hr 37 min ago



# Manage Nodes

Add, remove, status of nodes

Jenkins

Jenkins > Nodes >

ENABLE AUTO REFRESH

Back to Dashboard

Manage Jenkins

New Node

Configure

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

search

S Name ↓ Architecture Clock Difference Free Disk Space Free Swap Space Free Temp Space Response Time

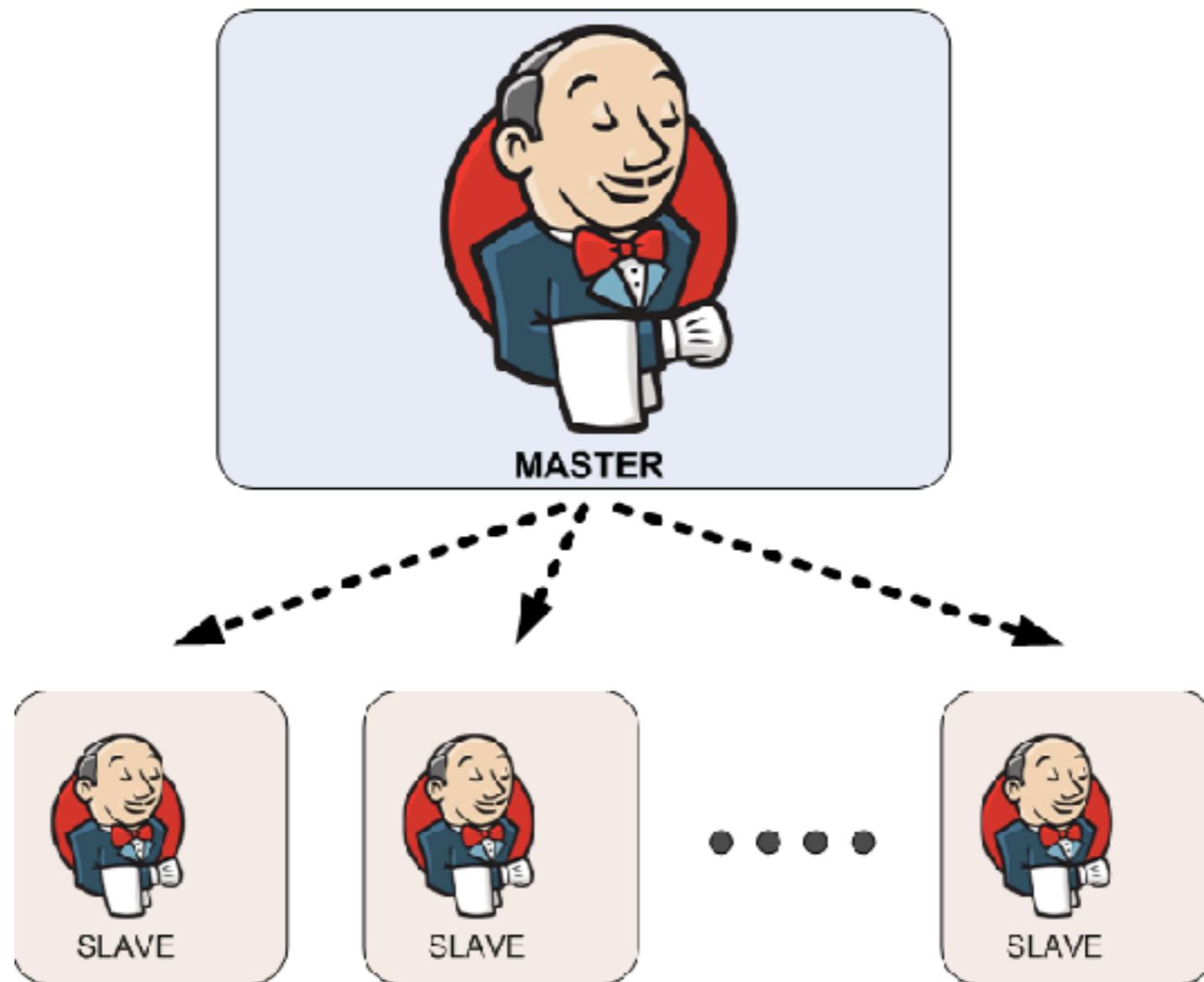
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Mac OS X (x86_64)	In sync	5.73 GB	463.00 MB	5.73 GB	0ms
	Data obtained	36 min	36 min	36 min	36 min	36 min	36 min

Refresh status



# Manage Nodes

Master-slave concept to scale Jenkins



# Create a first Job



# 1. Create a new job

The screenshot shows the Jenkins dashboard. At the top left is the Jenkins logo. Below it is a navigation bar with the word "Jenkins" and a right-pointing arrow. The main content area has a "Welcome to Jenkins!" message in large bold letters. Below it is a teal box containing the text "Please create new jobs to get started." A red box highlights the "New Item" button in the sidebar, which has a icon of a briefcase with a dollar sign. The sidebar also lists "People", "Build History", "Manage Jenkins", "My Views", and "Credentials". At the bottom left is a "Build Queue" section with the message "No builds in the queue.".



# 2. Fill in a job name

**Enter an item name**

hello

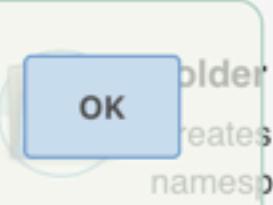
» Required field

 **Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Pipeline**  
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **External Job**  
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

 **Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

 **Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



# 3. Choose type of job

Enter an item name

hello

» Required field



## Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.



## Pipeline

Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



## External Job

This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.



## Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



# 3. Choose type of job

**Enter an item name**

hello

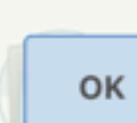
» Required field

**Freestyle project**  
 This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**  
 Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**External Job**  
 This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

**Multi-configuration project**  
 Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**  
 creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



# 4. General section

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

Project name: hello

Description:

[Plain text] [Preview](#)

Discard old builds    ?  
 GitHub project    ?  
 This project is parameterized    ?  
 Throttle builds    ?  
 Disable this project    ?  
 Execute concurrent builds if necessary    ?

[Advanced...](#)

Source Code Management

Save    Apply

None



# 4.1 Advanced options

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

Project name: hello

Description:

[Plain text] [Preview](#)

Discard old builds      [?](#)

GitHub project      [?](#)

This project is parameterized      [?](#)

Throttle builds      [?](#)

Disable this project      [?](#)

Execute concurrent builds if necessary      [?](#)

[Advanced...](#)

Source Code Management

Save      Apply

None



# 4.1 Advanced options

The screenshot shows the 'General' tab selected in a Jenkins build configuration. The tab bar also includes 'Source Code Management', 'Build Triggers', and 'Build Environment'. The 'General' tab contains the following configuration:

- Quiet period
- Retry Count
- Block build when upstream project is building
- Block build when downstream project is building
- Use custom workspace

Below these options is a 'Display Name' field with an empty input box. At the bottom of the section is another checkbox:

- Keep the build logs of dependencies



# 5. Source code management

By default is Git and Subversion

The screenshot shows a software interface for managing build configurations. At the top, there is a navigation bar with tabs: General, Source Code Management (which is currently selected), Build Triggers, Build Environment, Build, and Post-build Actions. The 'Source Code Management' tab is highlighted with a dark background and white text.

**Source Code Management**

None

Git

Subversion

**Build Triggers**

Trigger builds remotely (e.g., from scripts)

Build after other projects are built

Build periodically

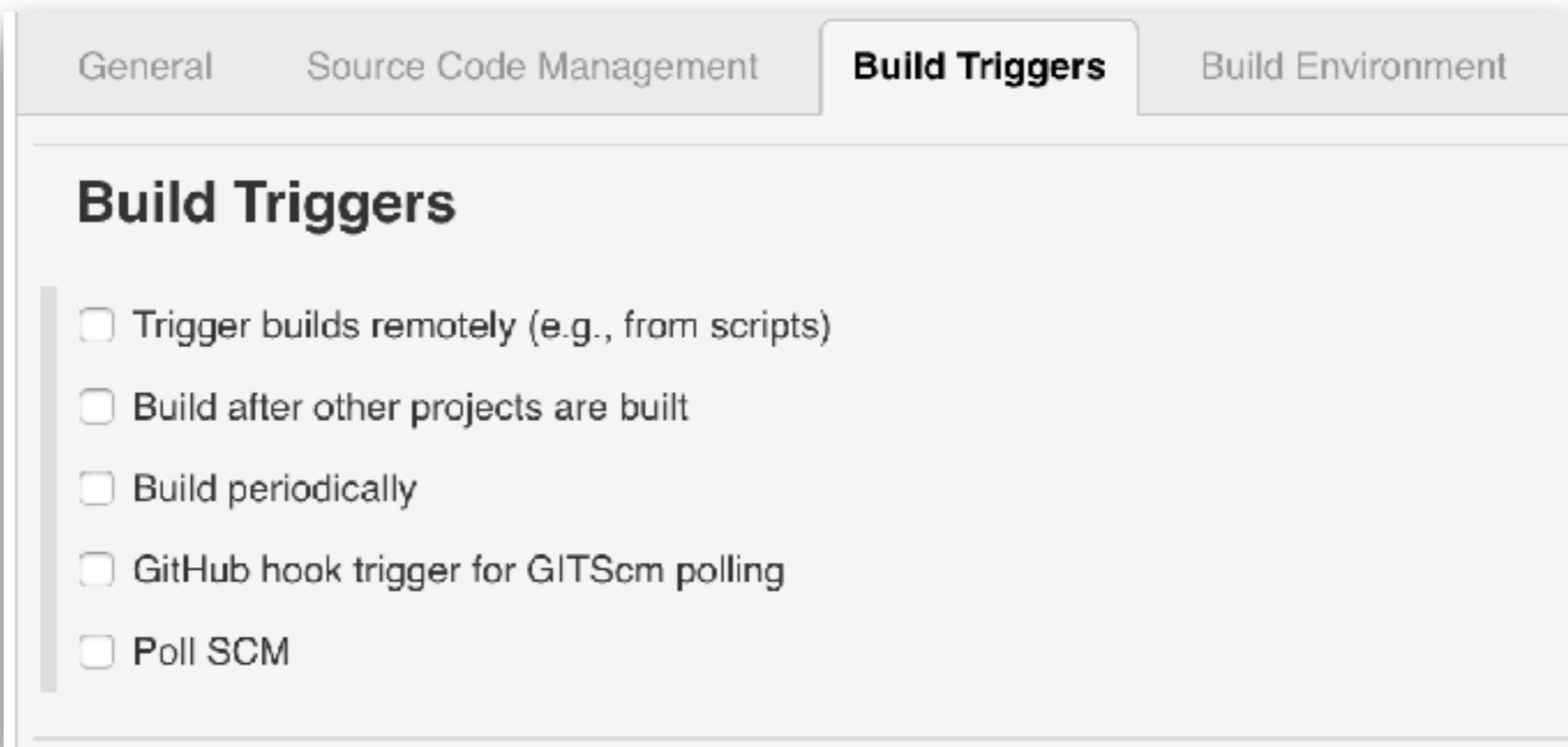
GitHub hook trigger for GITScm polling

Poll SCM



# 6. Build trigger

When to run this job



The screenshot shows a software interface for managing build triggers. At the top, there are four tabs: 'General', 'Source Code Management', 'Build Triggers', and 'Build Environment'. The 'Build Triggers' tab is currently active, indicated by a dark background and bold text. Below the tabs, the title 'Build Triggers' is displayed in a large, bold font. To the left of the main content area, there is a vertical gray sidebar. The main content area contains a list of five options, each preceded by an empty checkbox:

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



# 6.1 Periodically

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

## Build Triggers

Trigger builds remotely (e.g., from scripts) ?  
 Build after other projects are built ?  
 Build periodically ?

Schedule ?

**H 23 \* \* \***

⚠ **No schedules so will never run**

GitHub hook trigger for GITScm polling ?  
 Poll SCM ?



# 6.2 Poll SCM

Poll SCM ?

Schedule ?

No schedules so will only run due to SCM changes if triggered by a post-commit hook

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DOM MONTH DOW

MINUTE Minutes within the hour (0–59)  
HOUR The hour of the day (0–23)  
DOM The day of the month (1–31)  
MONTH The month (1–12)  
DOW The day of the week (0–7) where 0 and 7 are Sunday.

To specify multiple values for one field, the following operators are available. In the order of precedence,

- \* specifies all valid values
- M–N specifies a range of values
- M–N/X or \*/X steps by intervals of X through the specified range or whole valid range
- A,B,...,Z enumerates multiple values



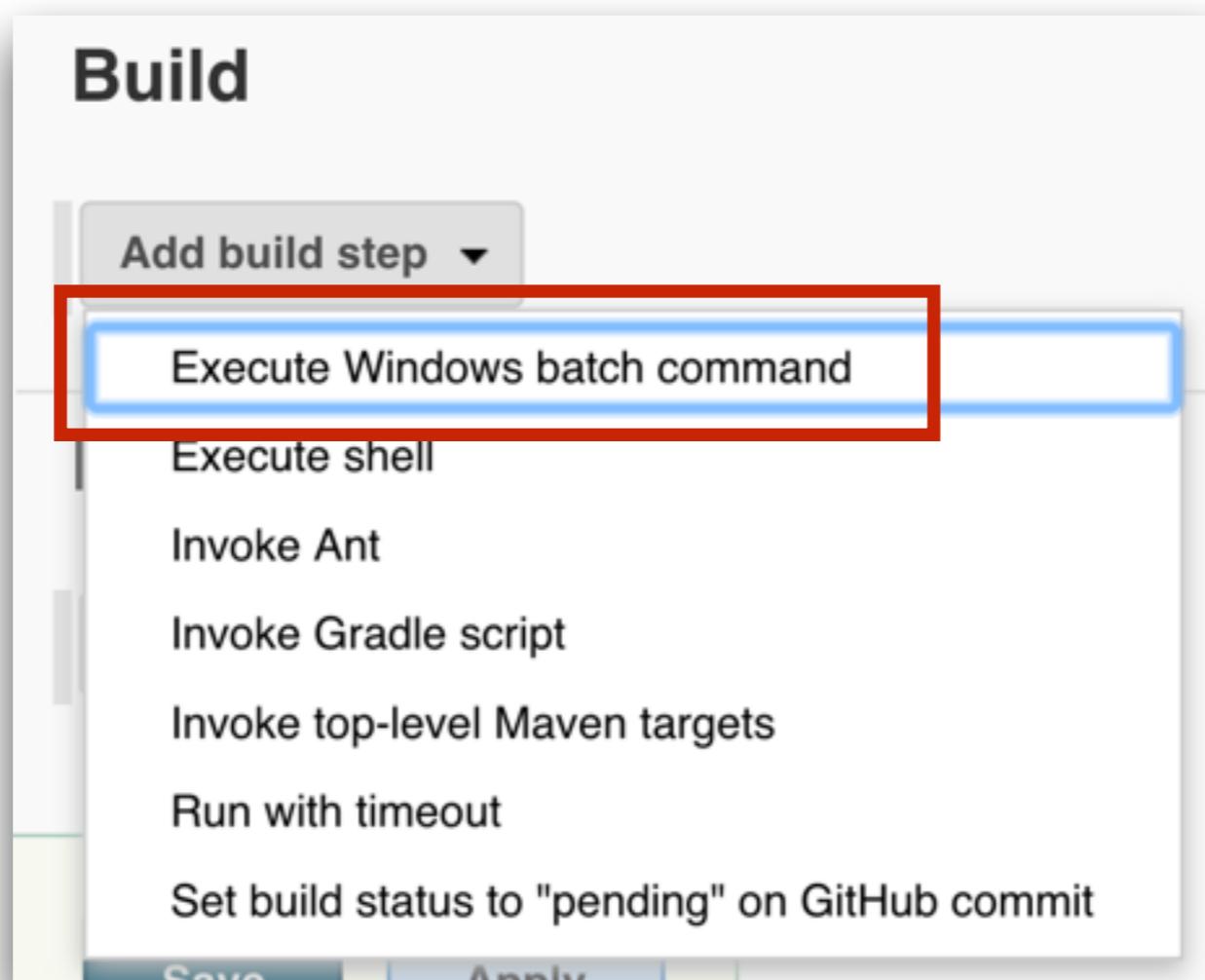
# 7. Add build step

What to run this job

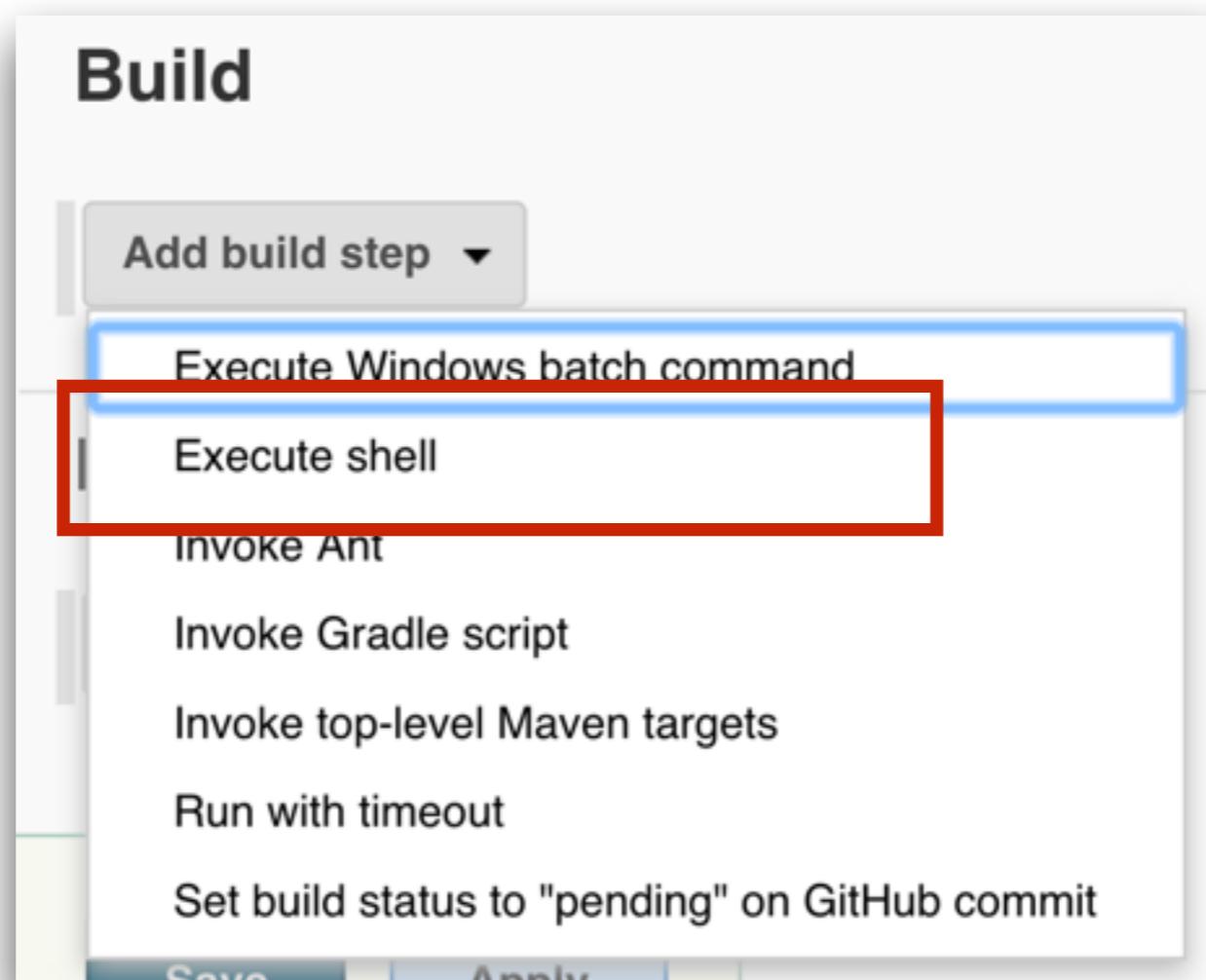
The screenshot shows a Jenkins job configuration interface. The top navigation bar includes tabs for General, Source Code Management, Build Triggers, Build Environment, **Build**, and Post-build Actions. The **Build** tab is active. Below the tabs is a section titled **BUILD ENVIRONMENT** containing four checkboxes: Delete workspace before build starts, Abort the build if it's stuck, Add timestamps to the Console Output, and Use secret text(s) or file(s). Under the **Build** tab, there is a sub-section with a dropdown menu labeled "Add build step". The menu lists several options: Execute Windows batch command (which is highlighted with a blue border), Execute shell, Invoke Ant, Invoke Gradle script, Invoke top-level Maven targets, Run with timeout, and Set build status to "pending" on GitHub commit. At the bottom of the "Add build step" panel are two buttons: "Save" and "Apply".



# 7.1 For Windows



# 7.2 For Linux/Mac



# 8. Post build actions

Generate reports

Send email

Run other jobs/projects



# 8. Post build actions

The screenshot shows a Jenkins configuration interface for a job. The top navigation bar includes tabs for General, Source Code Management, Build Triggers, Build Environment, Build, and Post-build Actions. The Post-build Actions tab is selected. Below the tabs, a section titled "BUILD ENVIRONMENT" is visible. A dropdown menu is open under "Post-build Actions", listing several options: Delete workspace before build starts, Aggregate downstream test results, Archive the artifacts, Build other projects, Publish JUnit test result report, Record fingerprints of files to track usage, Git Publisher, E-mail Notification (which is highlighted with a blue selection bar), Editable Email Notification, Set GitHub commit status (universal), Set build status on GitHub commit [deprecated], and Delete workspace when build is done. At the bottom left of the main area, there are "Save" and "Apply" buttons.

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

**BUILD ENVIRONMENT**

Delete workspace before build starts

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Record fingerprints of files to track usage
- Git Publisher
- E-mail Notification**
- Editable Email Notification
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Delete workspace when build is done

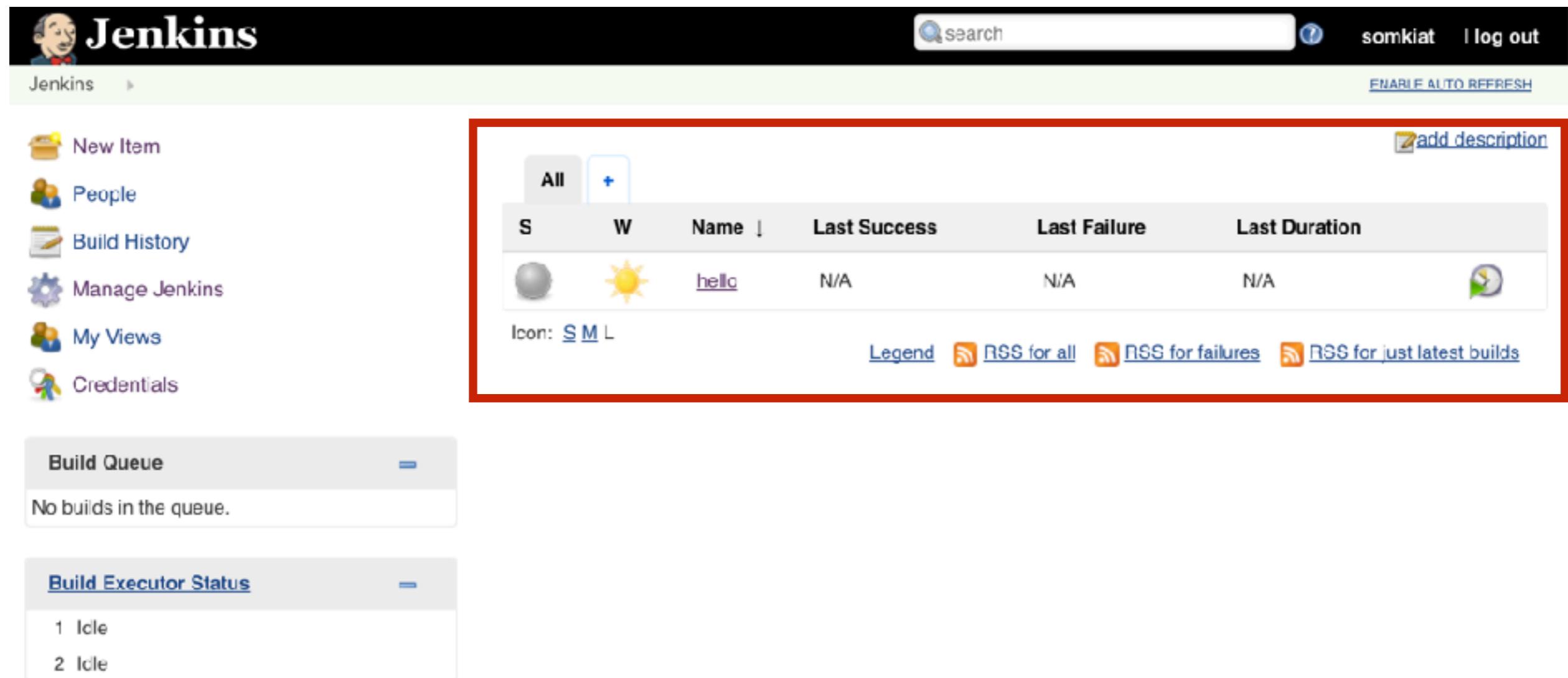
Add post-build action ▾

Save    Apply



# 9. Run your job

## Manual and Scheduler run



The screenshot shows the Jenkins dashboard with a red box highlighting the main job list area. The job 'hello' is listed with a yellow sun icon, indicating it is successful. The dashboard also includes sections for Build Queue and Build Executor Status.

S	W	Name	Last Success	Last Failure	Last Duration
		<a href="#">hello</a>	N/A	N/A	

Icon: [S](#) [M](#) [L](#)

Legend: [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

**Build Queue**  
No builds in the queue.

**Build Executor Status**  
1 Idle  
2 Idle



# 9. Run your job

Start build your job

The screenshot shows the Jenkins dashboard with the following details:

- Top Bar:** Includes the Jenkins logo, a search bar, and user information for "somkiat".
- Left Sidebar:** Lists navigation options: New Item, People, Build History, Manage Jenkins, My Views, and Credentials.
- Job Overview:** A table displays a single job entry:

S	W	Name	Last Success	Last Failure	Last Duration
		<a href="#">hello</a>	N/A	N/A	

Icon: [S](#) [M](#) [L](#)

RSS links: [RSS for all](#), [RSS for failures](#), [RSS for just latest builds](#)
- Build Queue:** Shows "No builds in the queue."
- Build Executor Status:** Shows "1 Idle" and "2 Idle".



# 9. Run your job

Start build your job

The screenshot shows the Jenkins interface for a project named "hello". The top navigation bar shows "Jenkins" and "hello". The main content area has a title "Project hello". On the left, there is a sidebar with several options: "Back to Dashboard", "Status", "Changes", "Workspace", "Build Now", "Delete Project", and "Configure". The "Build Now" button is highlighted with a red box. To the right of the sidebar, there are two links: "Workspace" and "Recent Changes".



# 9. Run your job

See your job's output

Jenkins > hello > #1

Back to Project  
 Status  
 Changes  
**Console Output** View as plain text  
 Edit Build Information  
 Delete Build  
 Next Build

**Console Output**

Started by user [Somkiat Puisungnoen](#)  
Building in workspace /Users/somkiat/Downloads/set/workspace/hello  
Finished: SUCCESS



# 10. See job's status

By default is **Blue**, **Red** and **Gray**

The screenshot shows a Jenkins dashboard with a single job listed:

S	W	Name ↓	Last Success	Last Failure	Last Duration
		<a href="#">hello</a>	N/A	N/A	

Icon: [S](#) [M](#) [L](#)

Legend: [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

**Blue = build success**

**Red = build failure**

**Gray = disabled/never executed**



# 11. See job's health

All[+](#)  [add description](#)

S	W	Name ↓	Last Success	Last Failure	Last Duration
		<a href="#">hello</a>	N/A	N/A	

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

Sunny = 100% success rate

Cloudy = 60% success rate

Raining = 40% success rate



# Let's workshop



# Jenkins driven by Plugins !!



# **Working with Pipeline as a Code**



# Pipeline as a Code

Declarative Pipeline

Scripted Pipeline

<https://jenkins.io/doc/book/pipeline/>



# Create a new job with pipeline

Enter an item name  
project01 » Required field

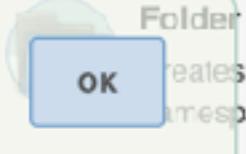
**Freestyle project**  
 This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Maven project**  
 Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

**Pipeline**  
 Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**External Job**  
 This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

**Multi-configuration project**  
 Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**  
 Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



# Write your pipeline

Pipeline script or from .Jenkinsfile

**Pipeline**

Definition

- Pipeline script
- Pipeline script from SCM

Script 1 try sample Pipeline... ?

Use Groovy Sandbox ?

[Pipeline Syntax](#)



# Pipeline syntax

Jenkins

search ?

Jenkins > xxxx > Pipeline Syntax

Back

**Snippet Generator**

Declarative Directive Generator

Declarative Online Documentation

Steps Reference

Global Variables Reference

Online Documentation

IntelliJ IDEA GDSL

Overview

This Snippet Generator will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click Generate Pipeline Script, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step archiveArtifacts: Archive the artifacts

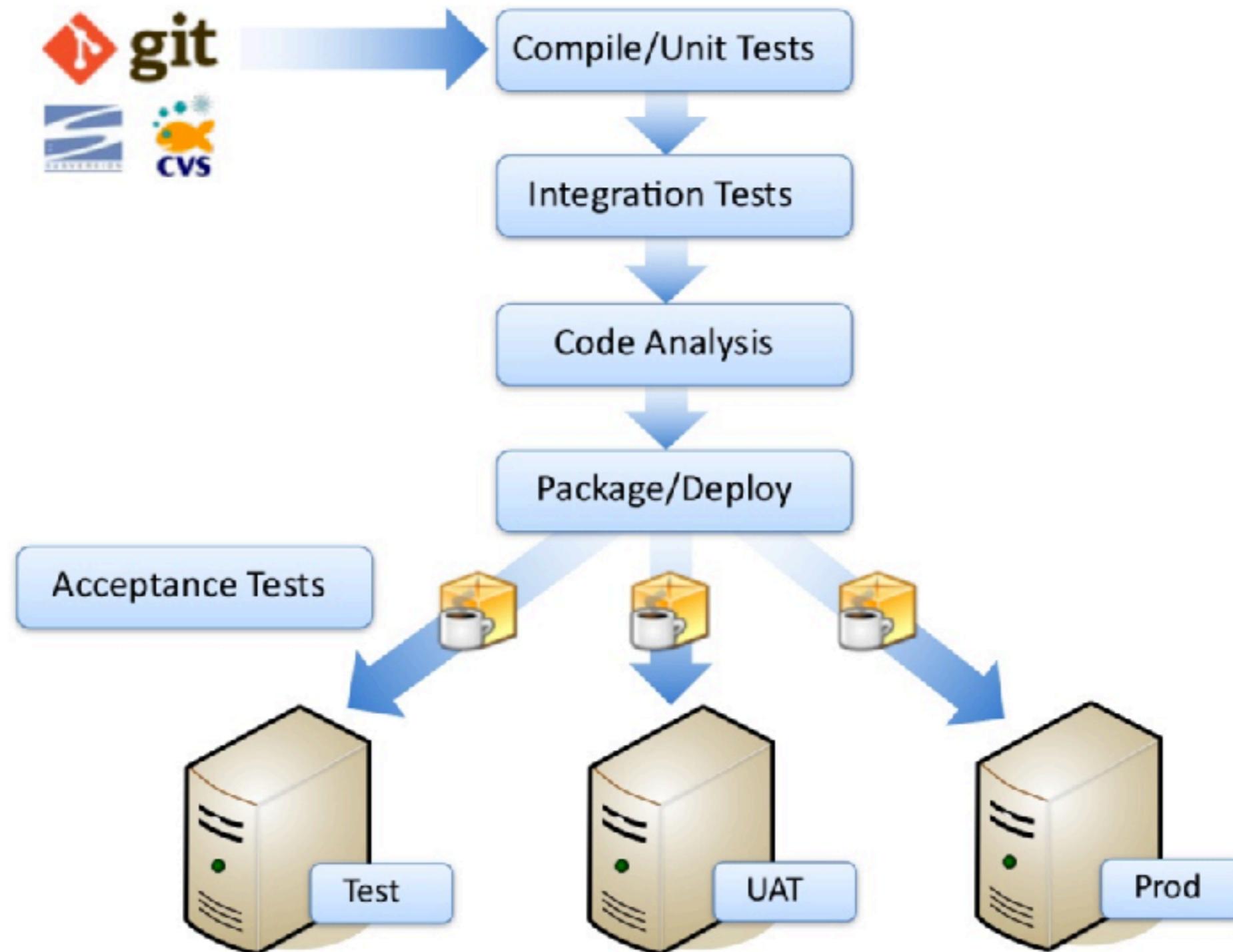
Files to archive

Advanced...

Generate Pipeline Script



# Build pipeline

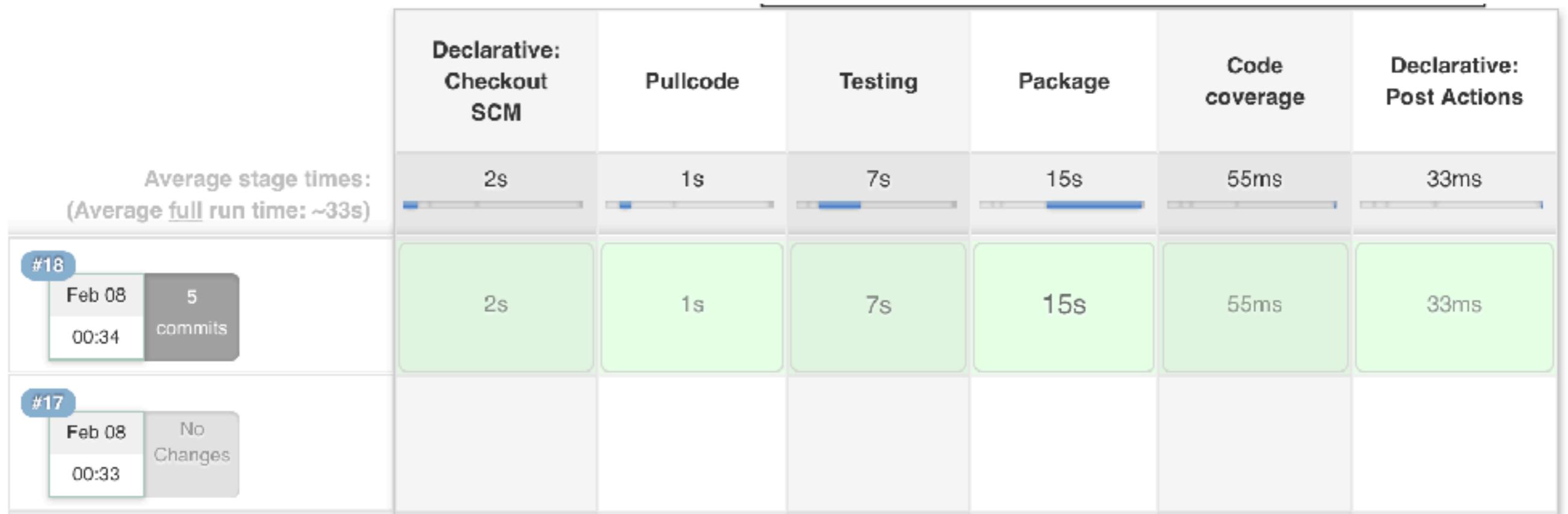


# Create pipeline as code

```
node {  
    stage('Pullcode') {  
        git 'https://github.com/up1/workshop-java-web-tdd.git'  
    }  
    stage('Testing') {  
        sh "mvn clean test"  
        junit 'target/surefire-reports/*.xml'  
    }  
    stage('Package') {  
        sh "mvn package"  
    }  
    stage('Code coverage') {  
        cobertura autoUpdateHealth: false, autoUpdateStability: false  
    }  
}
```



# Result



# Jenkinsfile

```
pipeline {  
    agent any  
    stages {  
        stage('Pullcode') {  
            steps {  
                git 'https://github.com/up1/workshop-java-web-tdd.git'  
            }  
        }  
        stage('Testing') {  
            steps {  
                sh "mvn clean test"  
                junit 'target/surefire-reports/*.xml'  
            }  
        }  
    }  
}
```



# Jenkinsfile

```
stage('Package') {
    steps {
        sh "mvn package"
    }
}
stage('Code coverage') {
    steps {
        cobertura autoUpdateHealth: false, autoUpdateStability: fa
    }
}
post {
    always {
        junit 'target/surefire-reports/*.xml'
    }
}
```



# Jenkins Best Practices

<https://wiki.jenkins.io/display/JENKINS/Jenkins+Best+Practices>



# Jenkins Best Practices

**Always secure Jenkins**

In large system, don't build on the master

**Backup JENKINS\_HOME regularly**

Limit project name to the sane character set

<https://wiki.jenkins.io/display/JENKINS/Administering+Jenkins>



# Jenkins Best Practices

Always config your job to generate report

Archive unused jobs before removing them

Setting difference job for each branch

Prevent resource collisions in job (parallel)

Fail fast



# DevOps Toolchain



# Build an Open-Source DevOps Toolchain

**Repository:**

- JFrog Artifactory
- Sonatype Nexus OSS
- Git (on-premises)
- GitLab CE
- Bit Component Repo

**IDE:**

- Eclipse
- CircleCI
- Electric Cloud

**Testing:**

- Appium
- Jmeter
- Protractor
- Selenium
- Visual Studio Test Platform
- Watir

**Security:**

- ChaoSlingr
- Hashicorp Vault
- OWASP ZAP
- OSSEC

**Build/Integration:**

- Jenkins
- CloudBees (Jenkins)
- Apache Maven
- Gradle
- CruiseControl
- Travis CI
- XebiaLabs (Hudson)

**Bug Tracking:**

- Mantis
- Bugzilla
- Redmine
- Trac

**Code Review:**

- Gerrit
- Review Board
- Sonarqube

**Compliance Verification:**

- Inspec
- Compliance Masonry

**Task Management:**

- Taiga
- TaskBoard
- Pencil
- Inkscape
- OpenProject
- XWiki

**BDD:**

- Behat
- Concordian
- Cucumber
- Jbehave
- Jasmine
- SpecFlow

**Release, Deploy and Coordination:**

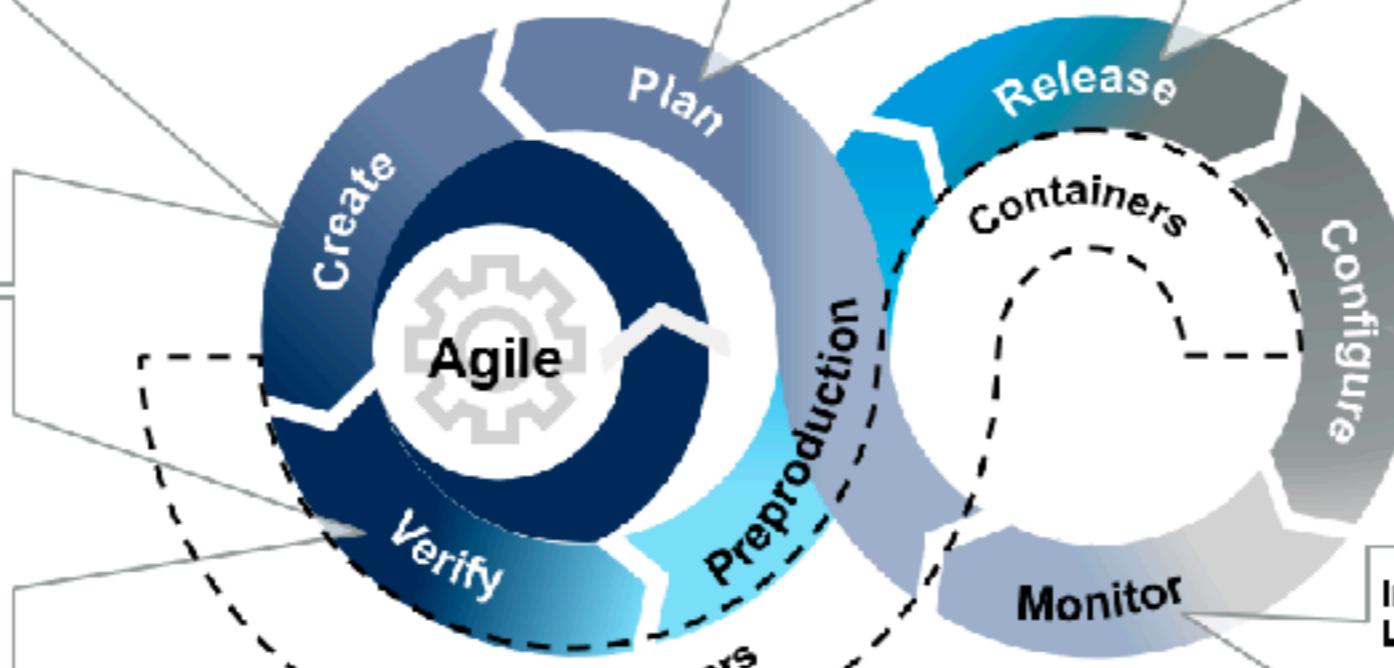
- JFrog Bintray
- OpenMake DeployHub
- Hashicorp TerraForm
- ElectricFlow
- Capistrano
- Spinnaker
- GoCD

**Continuous Configuration Automation**

- CFEngine
- Chef
- Inedo Otter
- Puppet Labs
- Red Hat (Ansible Tower)
- SaltStack
- StackStorm

**Infrastructure, APM and Analytics, and Log Management:**

<ul style="list-style-type: none"> <li>Cacti</li> <li>Caliper</li> <li>ELK Stack</li> <li>Ganglia</li> <li>Graphite</li> <li>Hyperic</li> <li>Hawtio</li> </ul>	<ul style="list-style-type: none"> <li>Kubernetes</li> <li>Vagrant</li> <li>Mesos</li> <li>Nomad</li> </ul>	<ul style="list-style-type: none"> <li>Heapster</li> <li>InfluxData</li> <li>Nagios</li> <li>Prometheus</li> <li>Wireshark</li> <li>Zabbix</li> <li>Zenoss Core</li> </ul>	<ul style="list-style-type: none"> <li>Fluentd</li> <li>Collectd</li> <li>Semantext Logagent</li> <li>Apache ELK</li> </ul>
---	---	--	---



- Docker
- Canonical LXD
- CoreOS
- CentOS
- Kubernetes
- Vagrant
- Mesos
- Nomad

© 2019 Gartner, Inc.

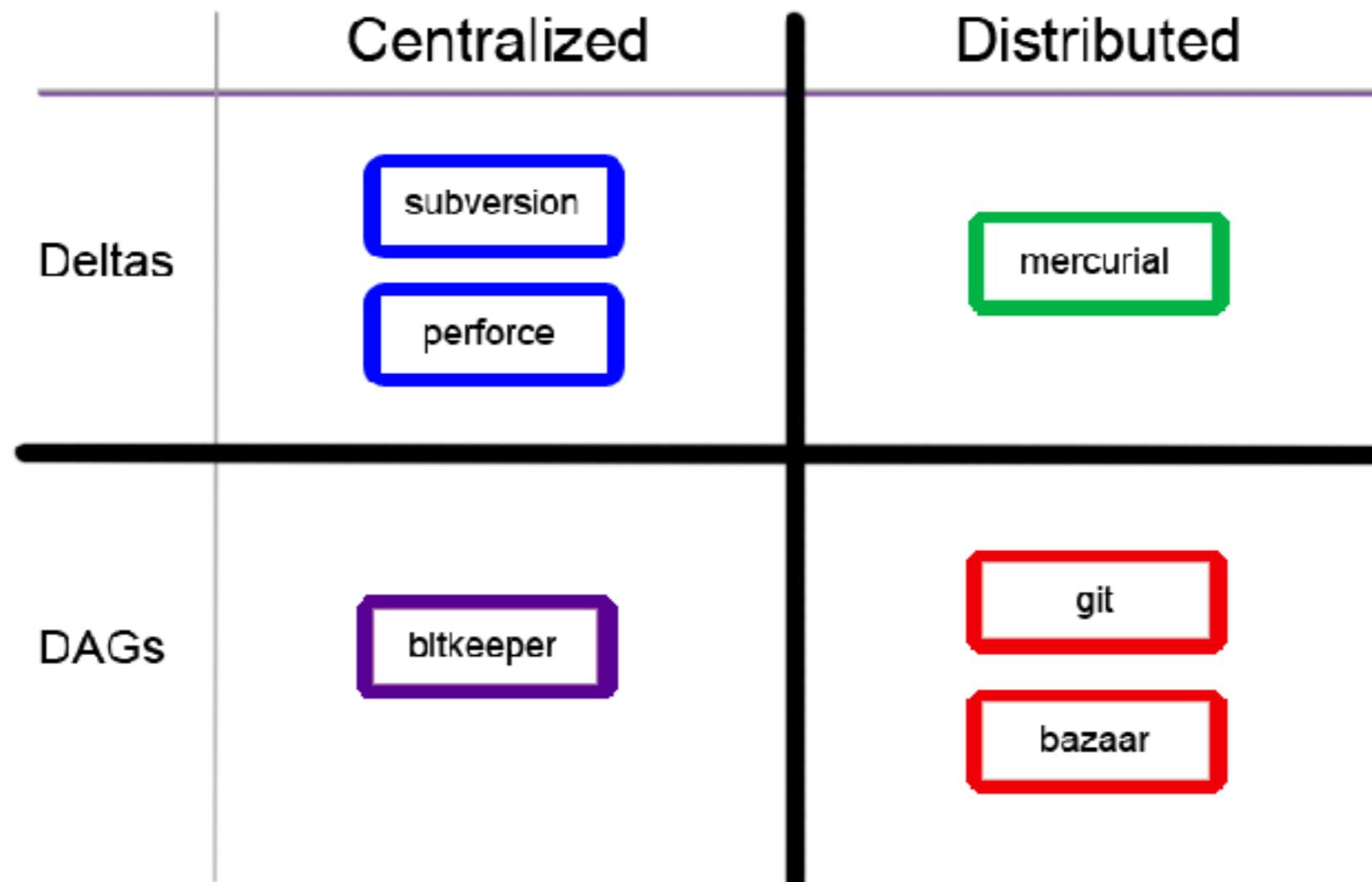
IID: 378544



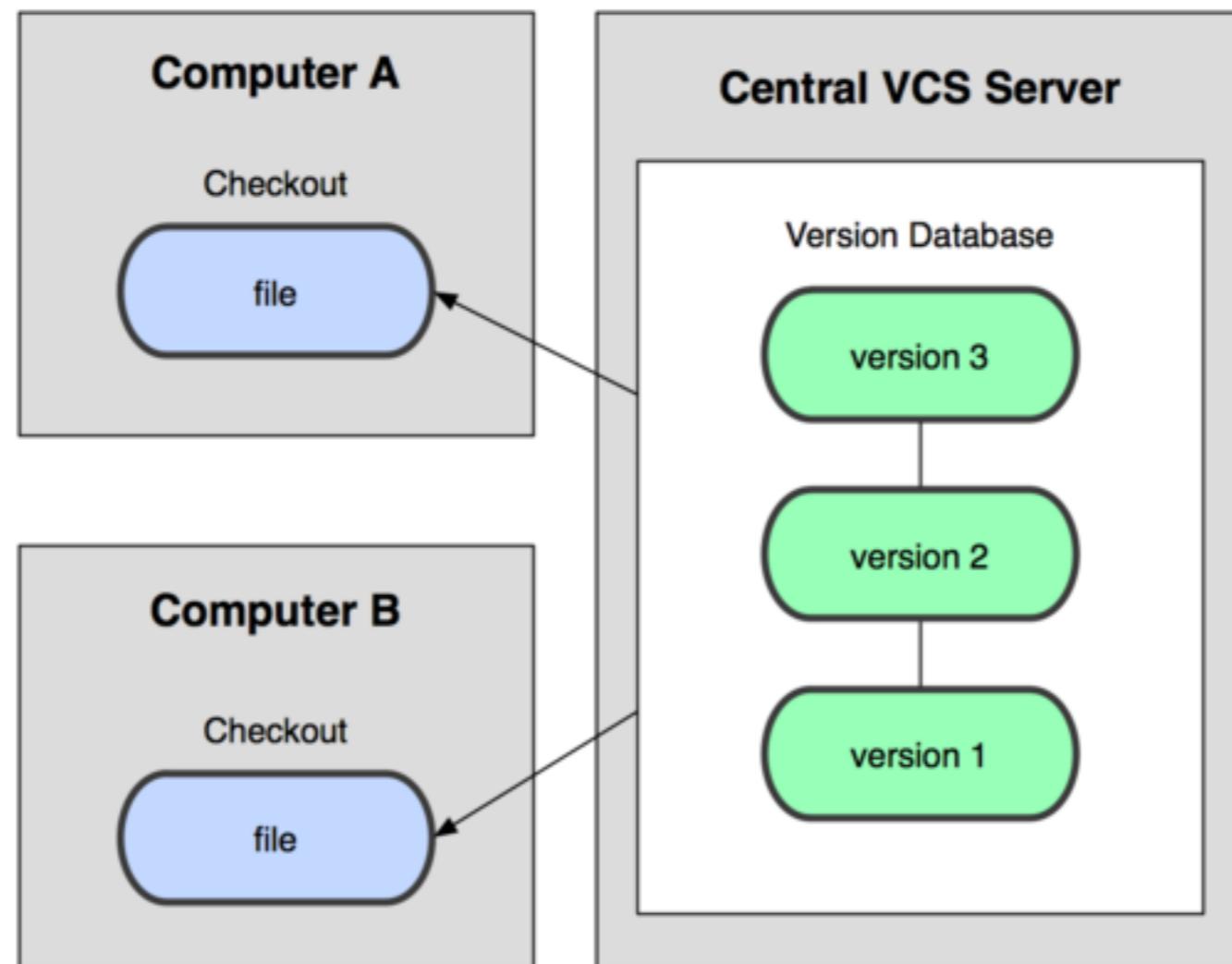
# Version Control System



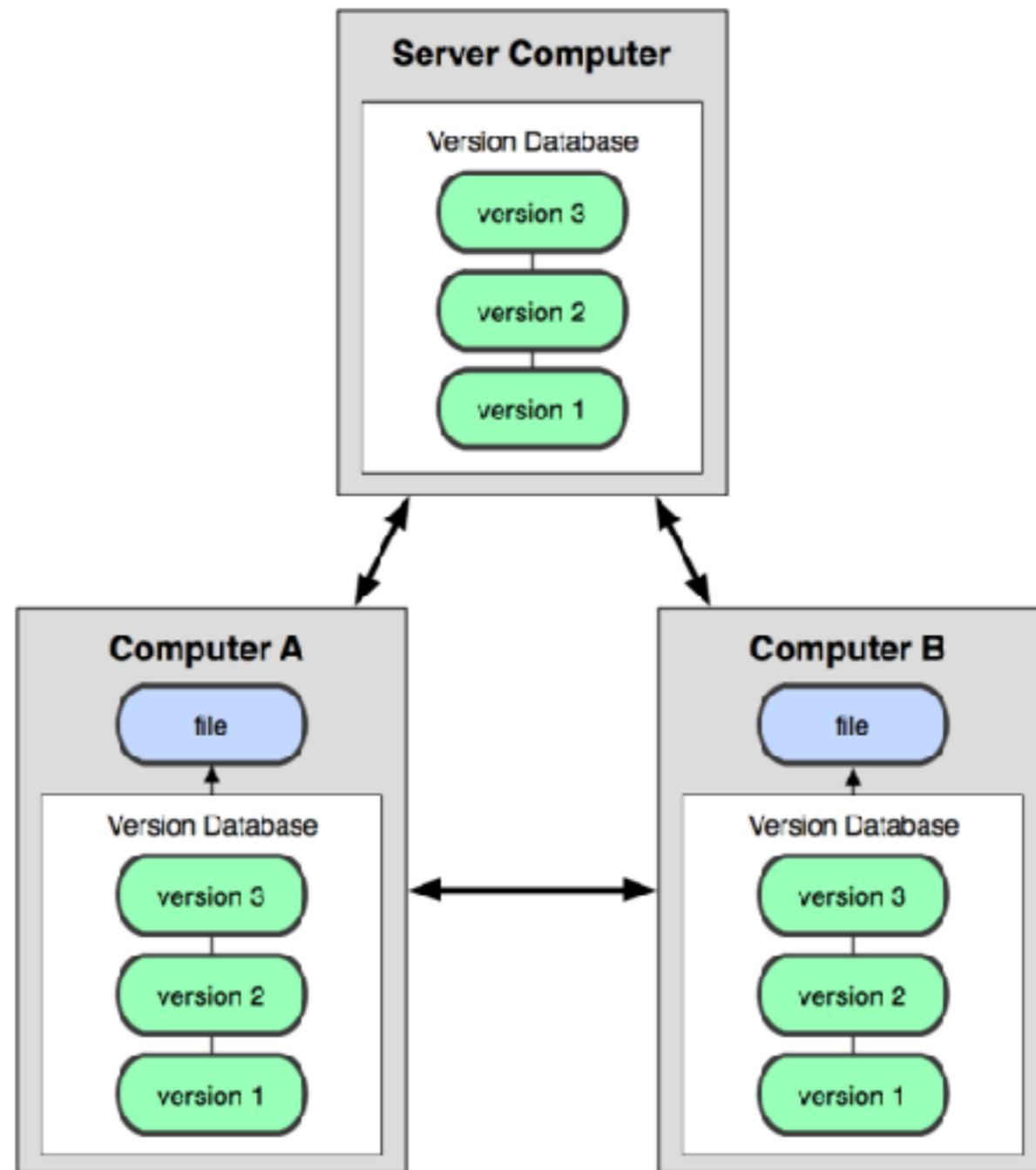
# Version Control System



# Centralize



# Decentralize



# Goals of Git

Speed

Simple

Support many parallel branches

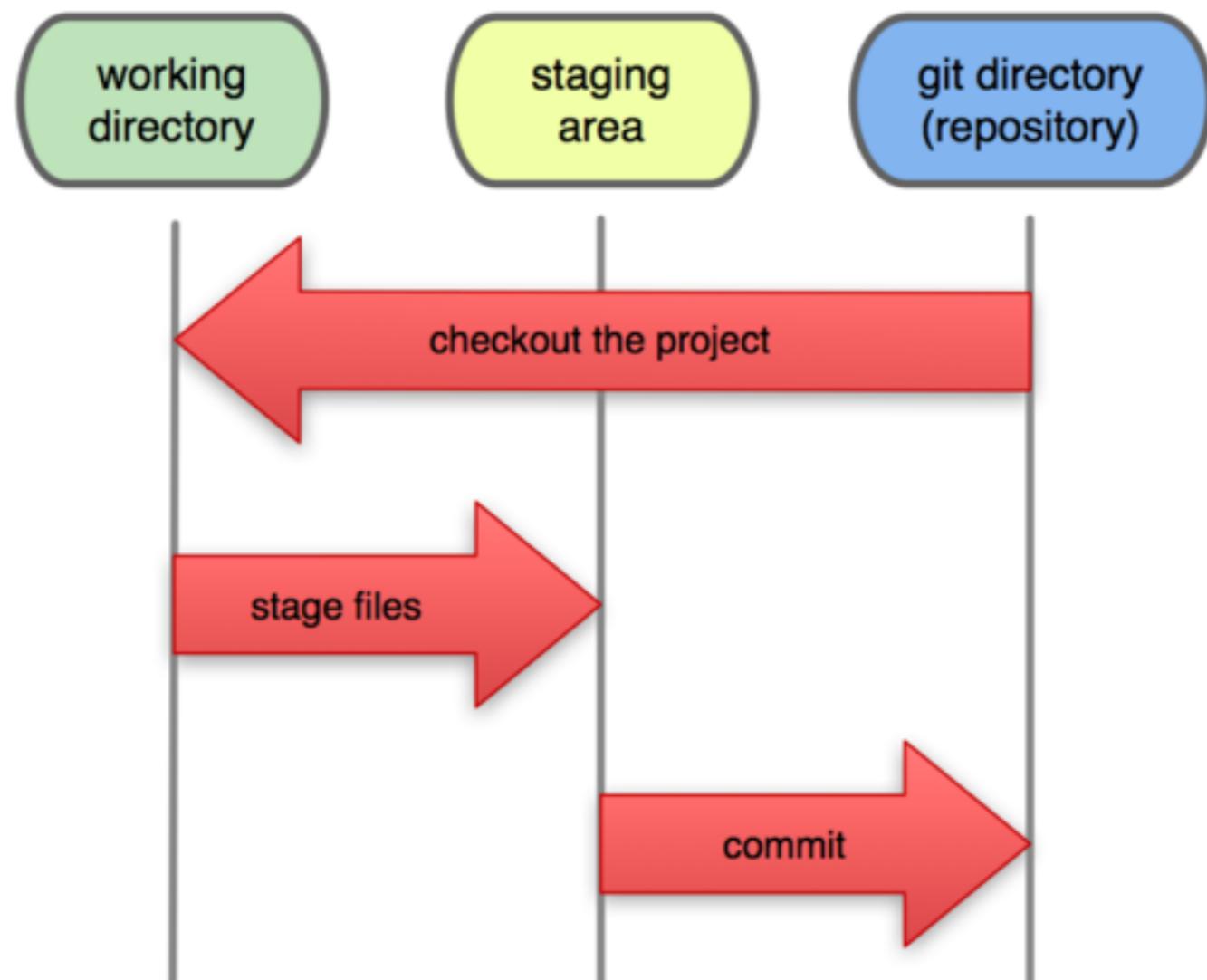
Fully distributed

Handle large project



# Git Workflow

## Local Operations



# Installation

The screenshot shows the official Git website (<https://git-scm.com/>). At the top left is the Git logo with the tagline "git --everything-is-local". A search bar at the top right contains the placeholder "Search entire site...". Below the header, there's a brief introduction: "Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency." To the right of this text is a 3D diagram illustrating the distributed nature of Git, showing multiple repositories connected by bidirectional arrows. Below the introduction, there are five main navigation links: "About", "Documentation", "Downloads", "Community", and a large "Download" button for the latest release. Each link has a corresponding icon: a gear for "About", an open book for "Documentation", a downward arrow for "Downloads", and a speech bubble for "Community". The "Downloads" section also includes a note about GUI clients and binary releases for major platforms.

git --everything-is-local

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

**About**  
The advantages of Git compared to other source control systems.

**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

**Downloads**  
GUI clients and binary releases for all major platforms.

**Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release  
**2.30.0**  
Release Notes (2020-12-27)  
Download 2.27.0 for Mac

<https://git-scm.com/>

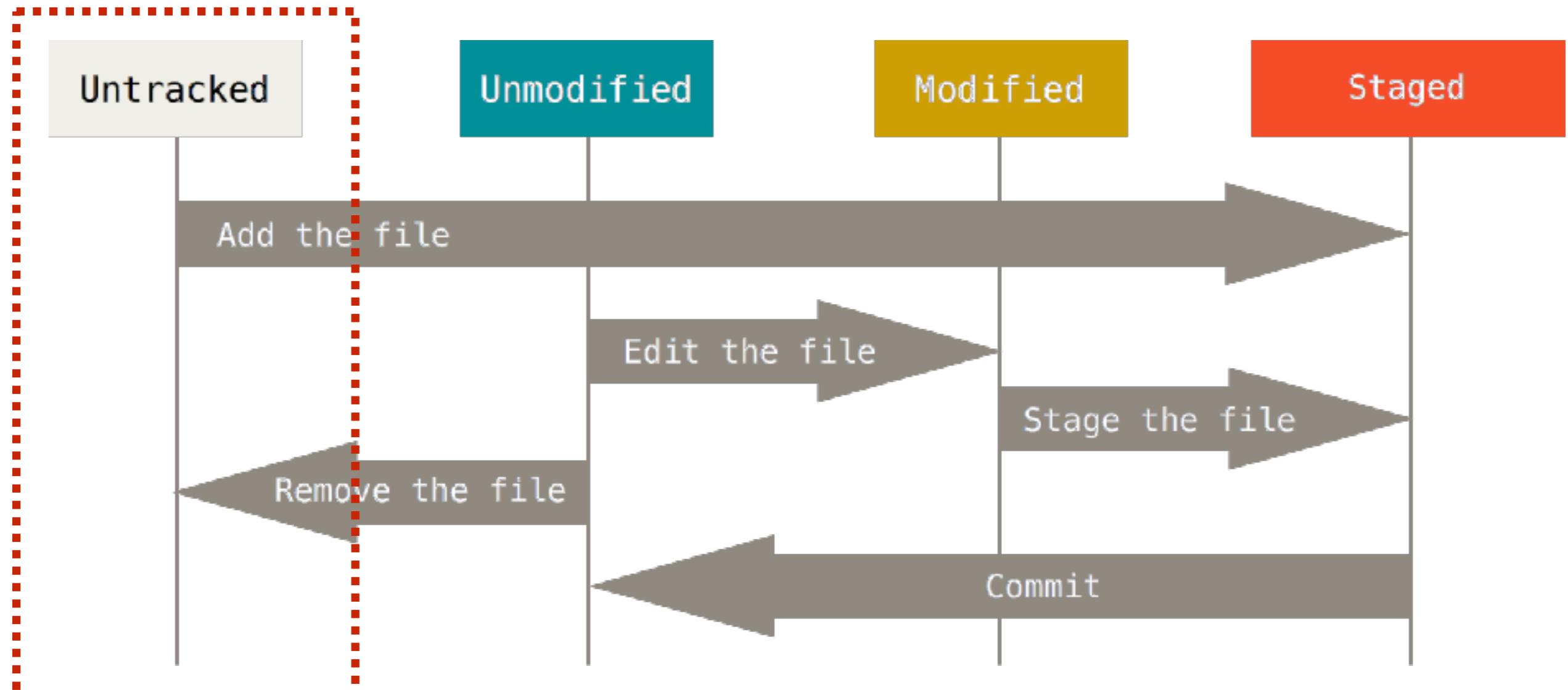


# Life cycle of file status



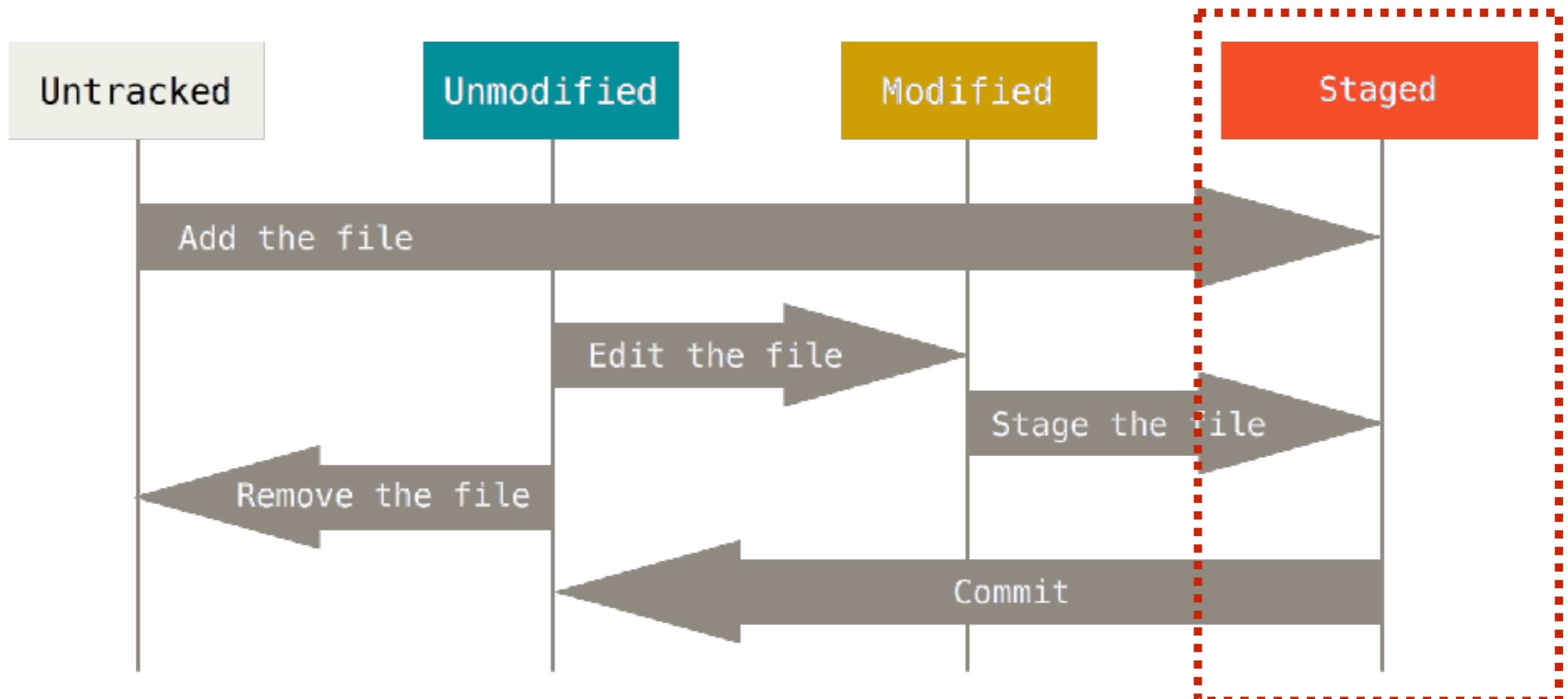
# Create new file

\$touch *README*



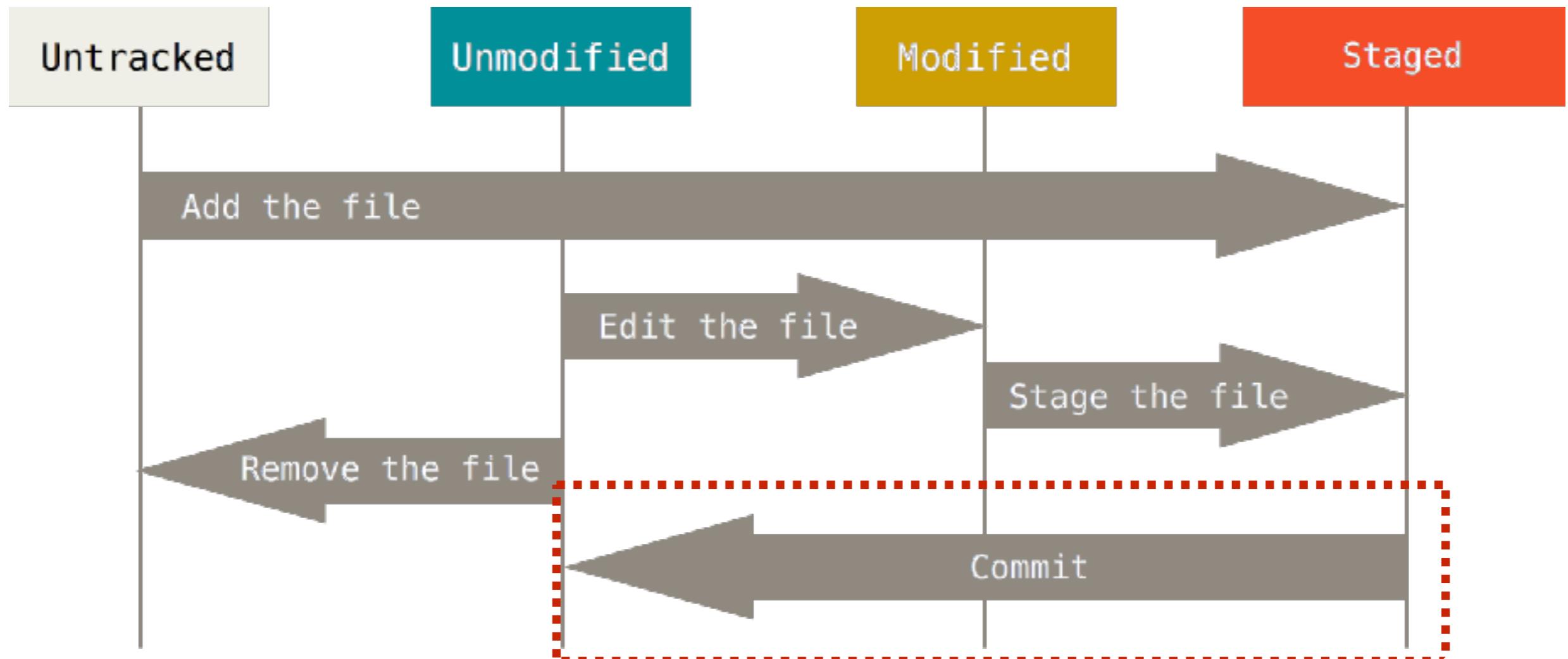
# Add file to staged

\$git add *README*



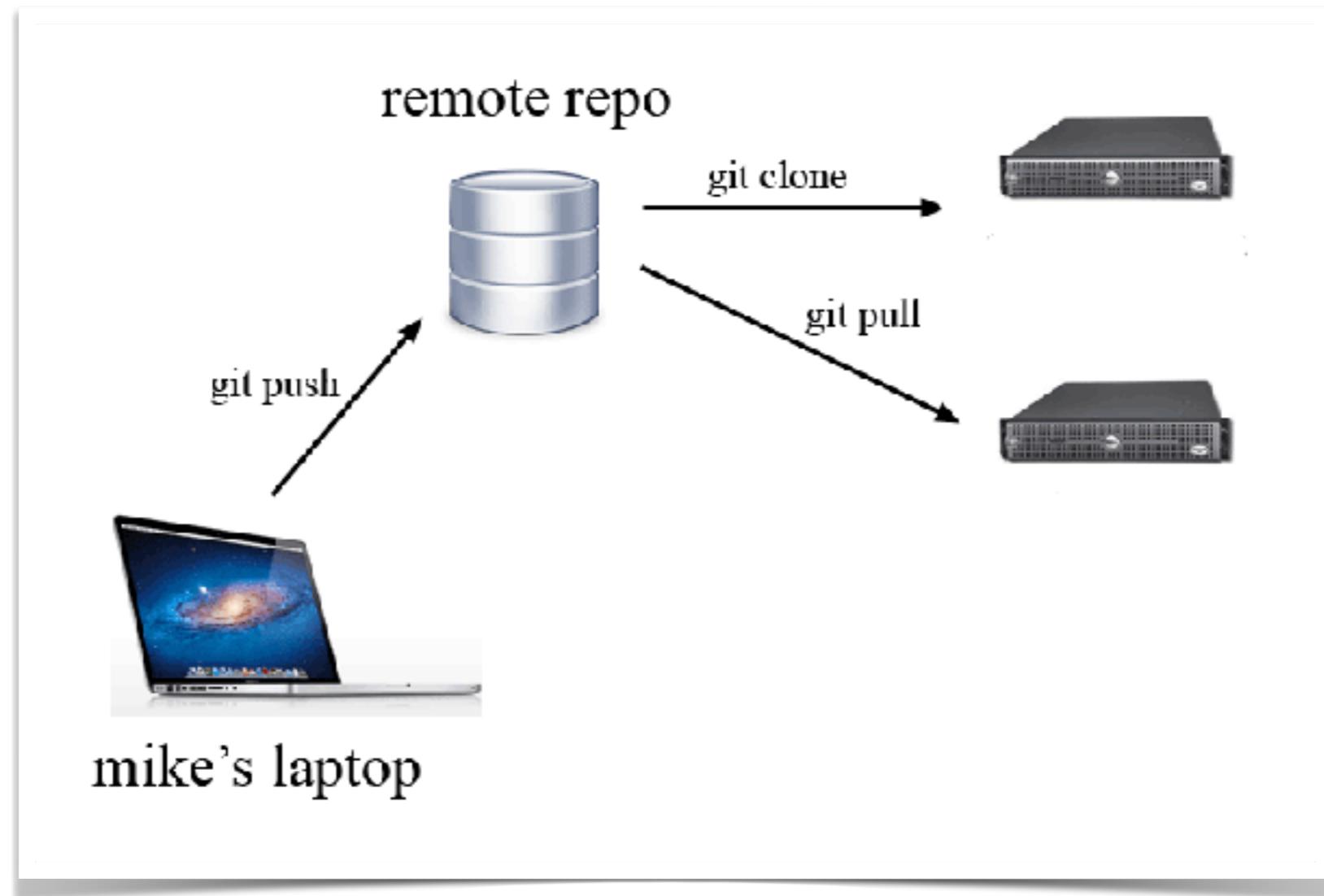
# Commit your changes ...

`$git commit -m “your message”`



# Push local's changes to remote

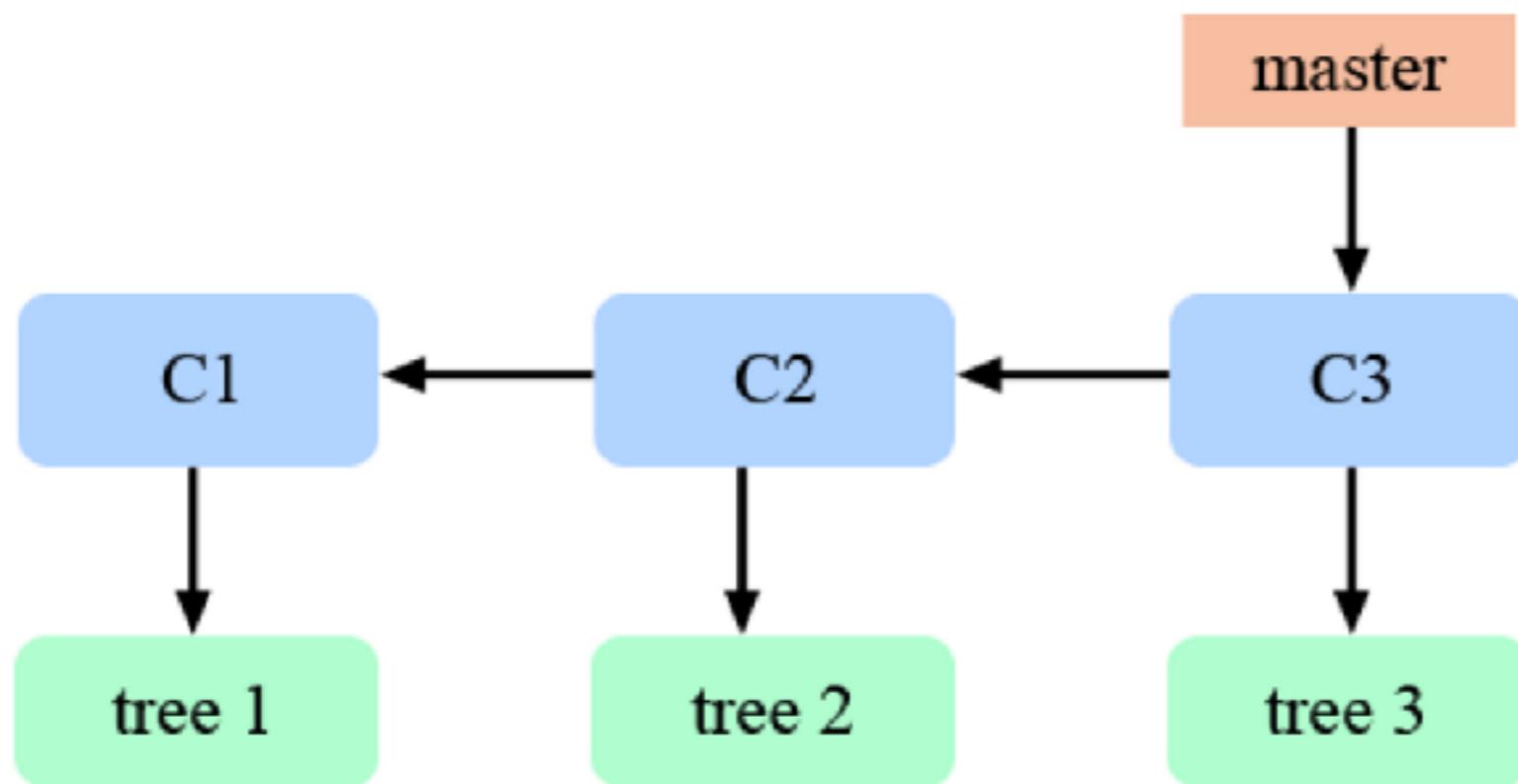
\$git push



# Working with branch

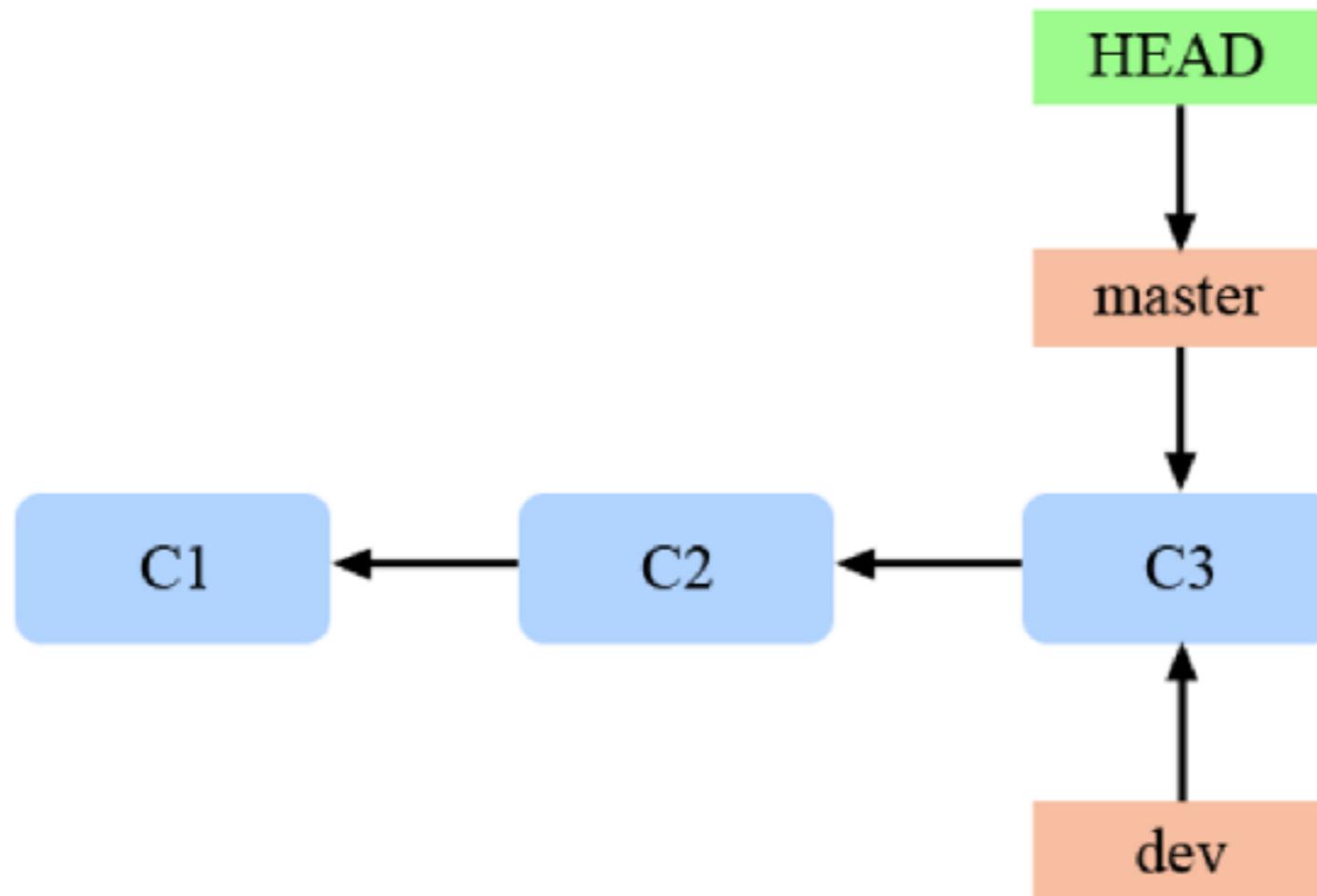


# Each branch points to a commit



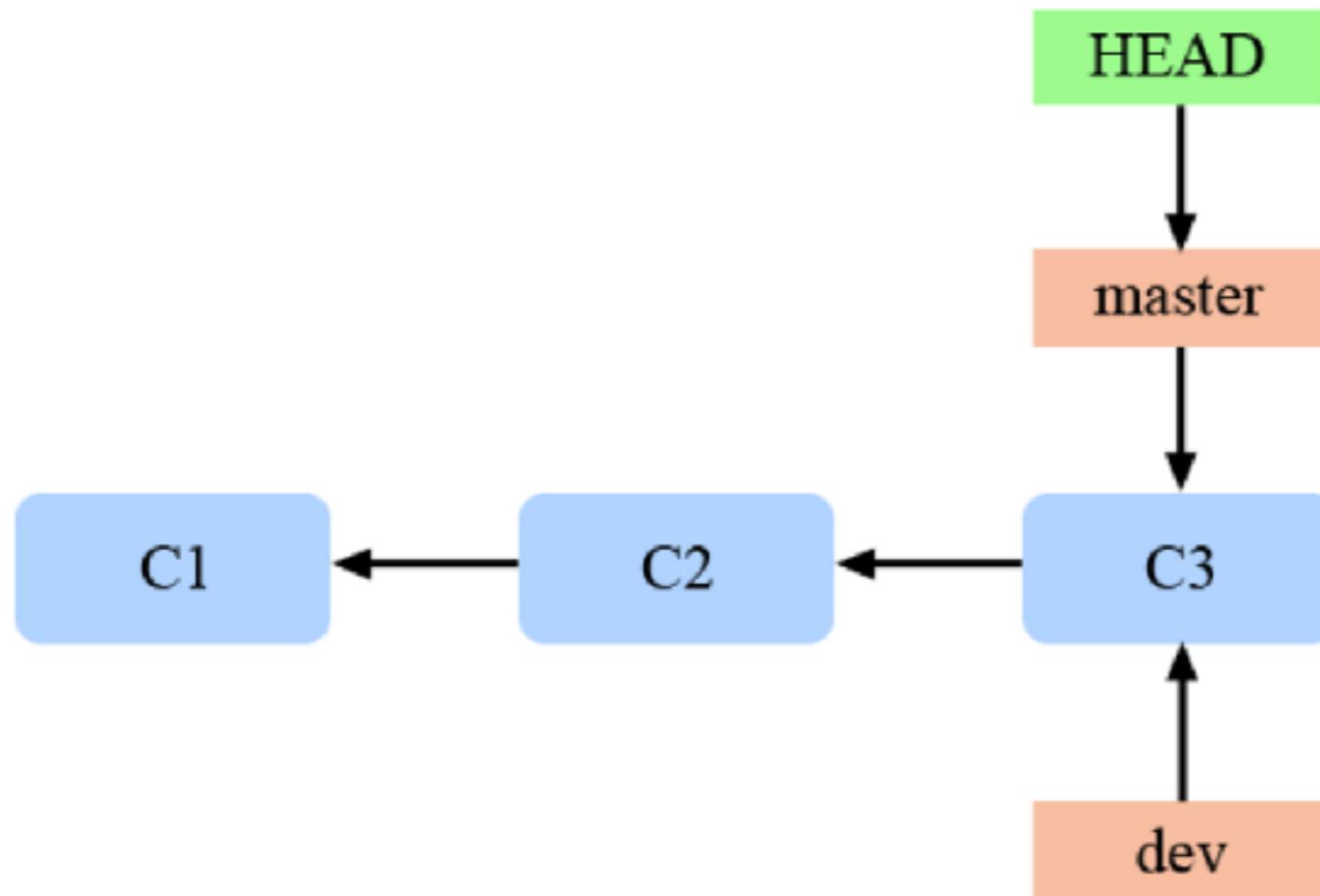
# Create new branch

\$git branch <branch name>



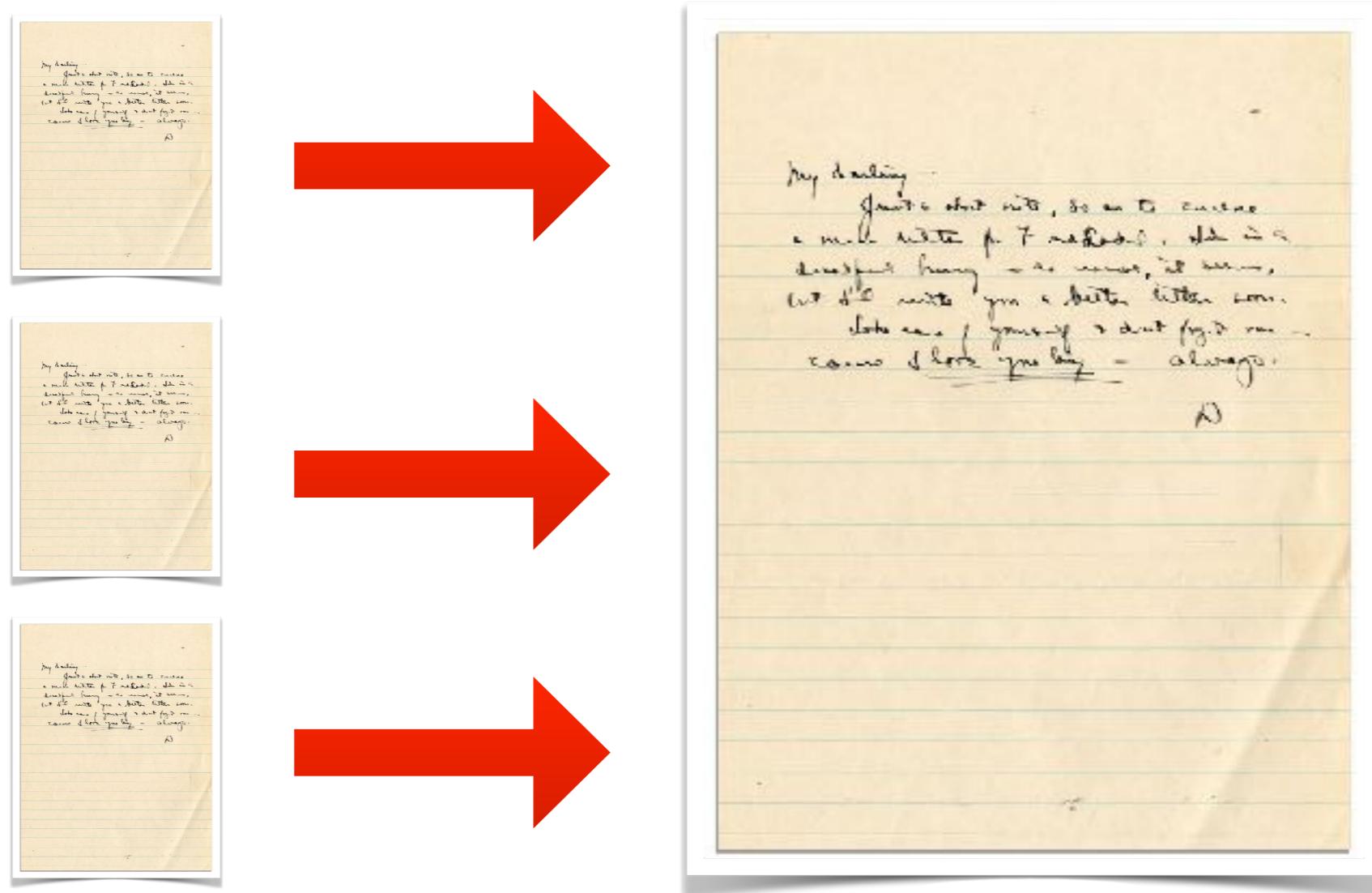
# Switch branch

\$git switch <branch name>

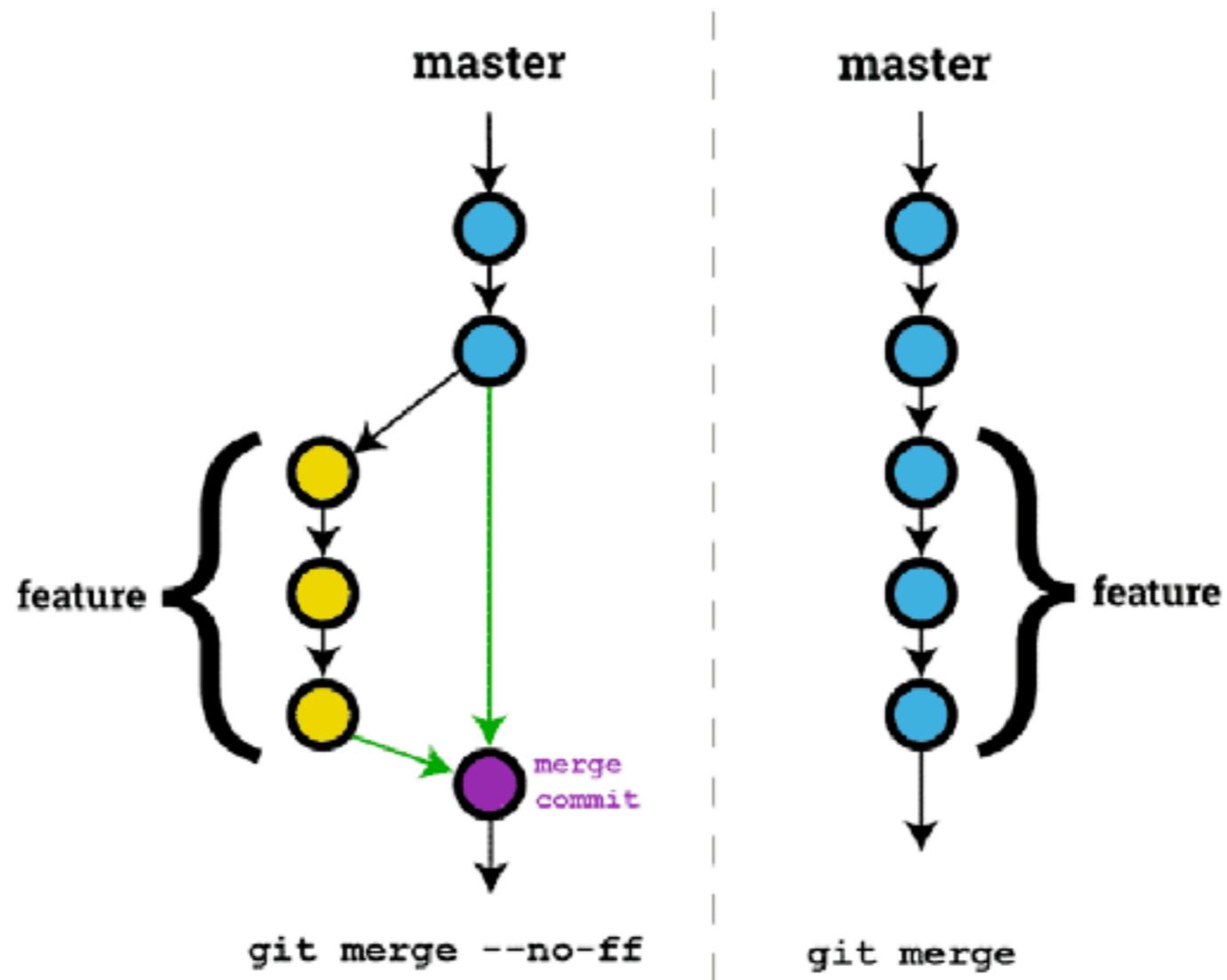


# All branch become one

\$git merge



# Merge vs Rebase



**BRANCHED FROM MASTER 3  
WEEKS AGO**



**MERGED BACK WITHOUT  
ANY CONFLICTS**



# Avoid merge conflict !!

Small change and commit

Early merge

Single Responsibility Principle

**Communication is a key**



# Commit message



Update  
TODO  
fixed bug  
add feature



“Added a user object to the database.  
currently only has a name and email.  
no authentication yet ”



“Fixed the bug that would add something to everything. Turn to not add something to everything”



# Workshop

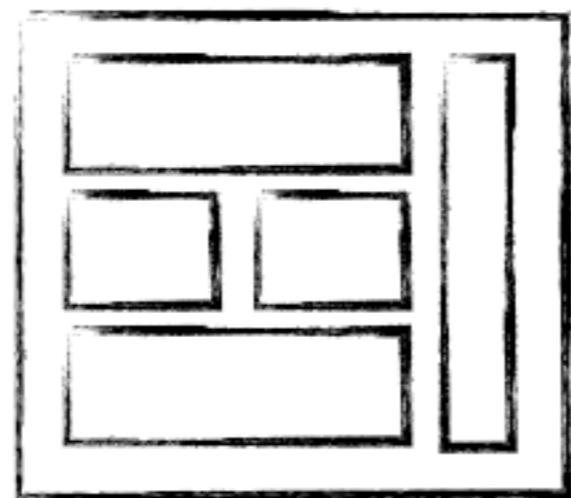


# Manage containers



# Why docker ?

Software industry has changed !!



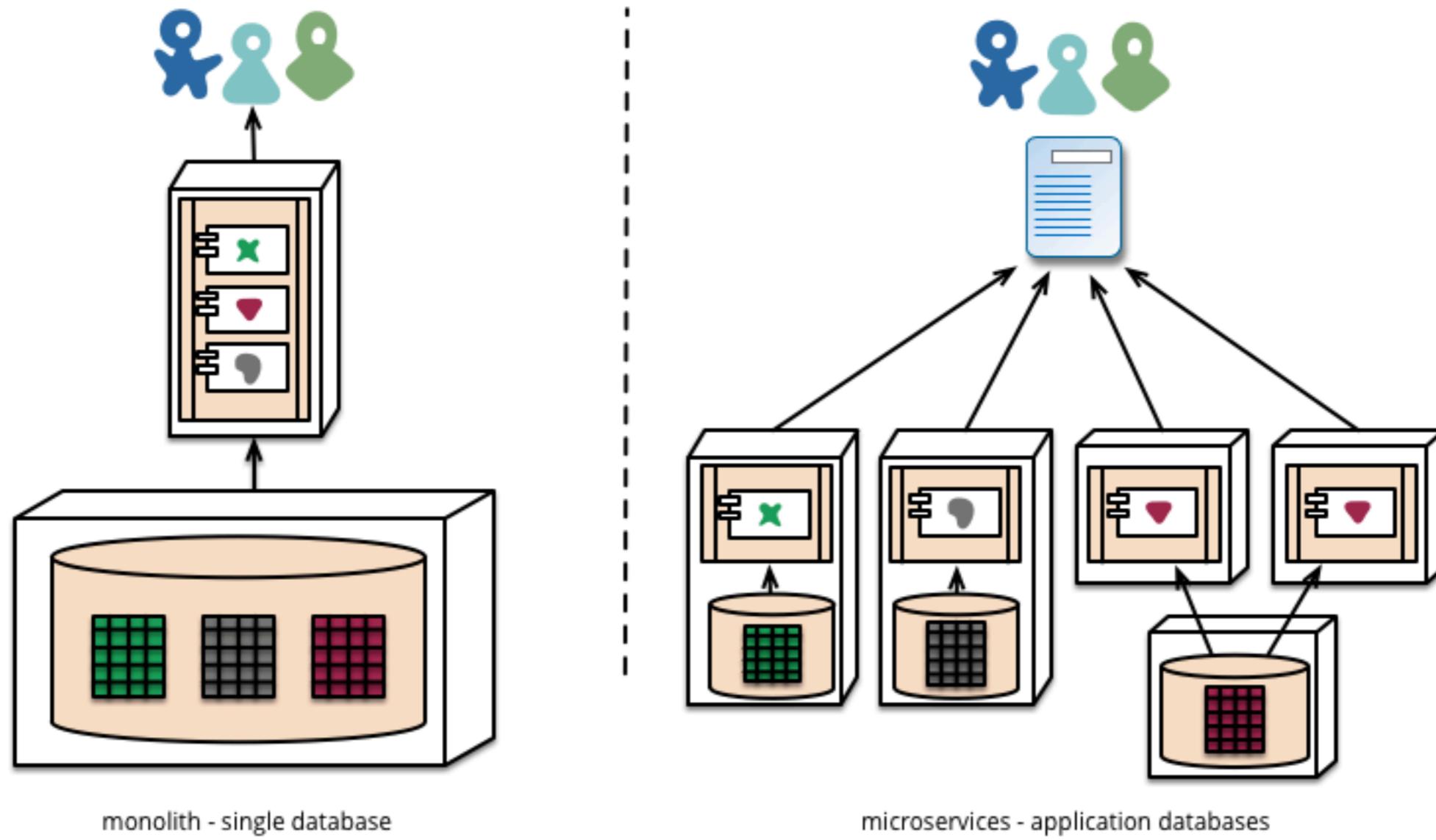
MONOLITHIC/LAYERED



MICRO SERVICES



# Microservice architecture

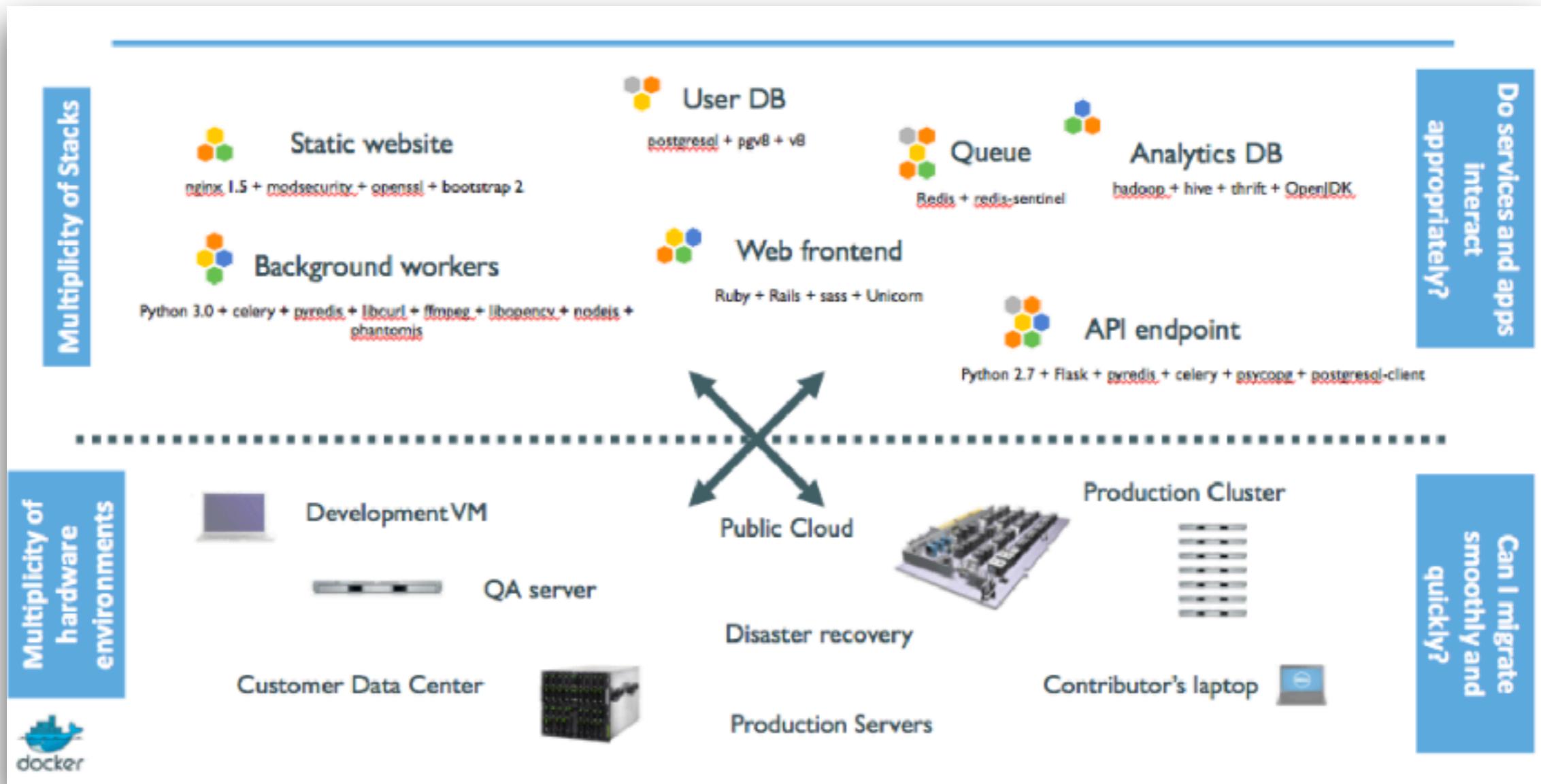


<https://martinfowler.com/articles/microservices.html>



# Why docker ?

We have problem !!

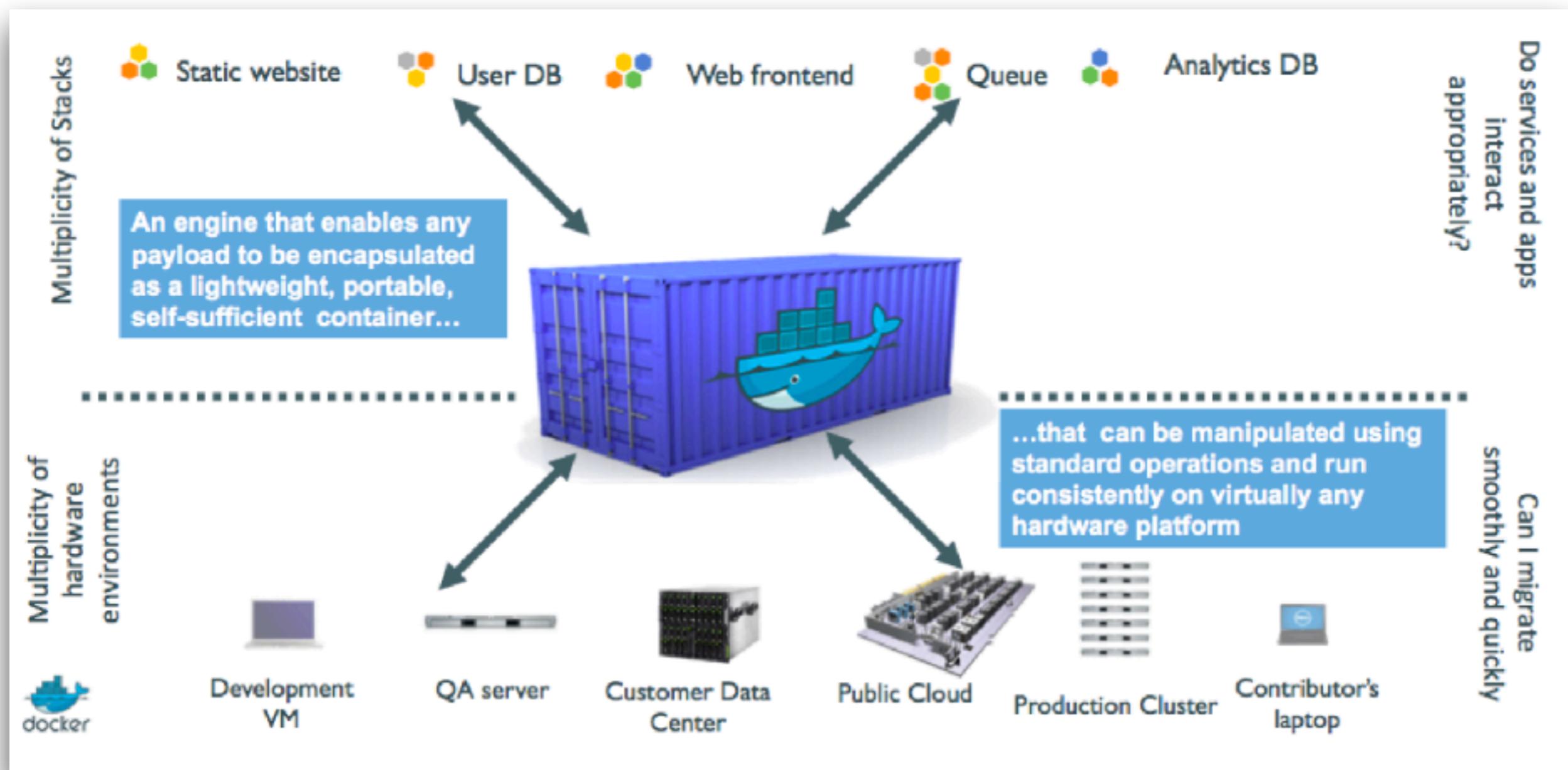


# Problem ?

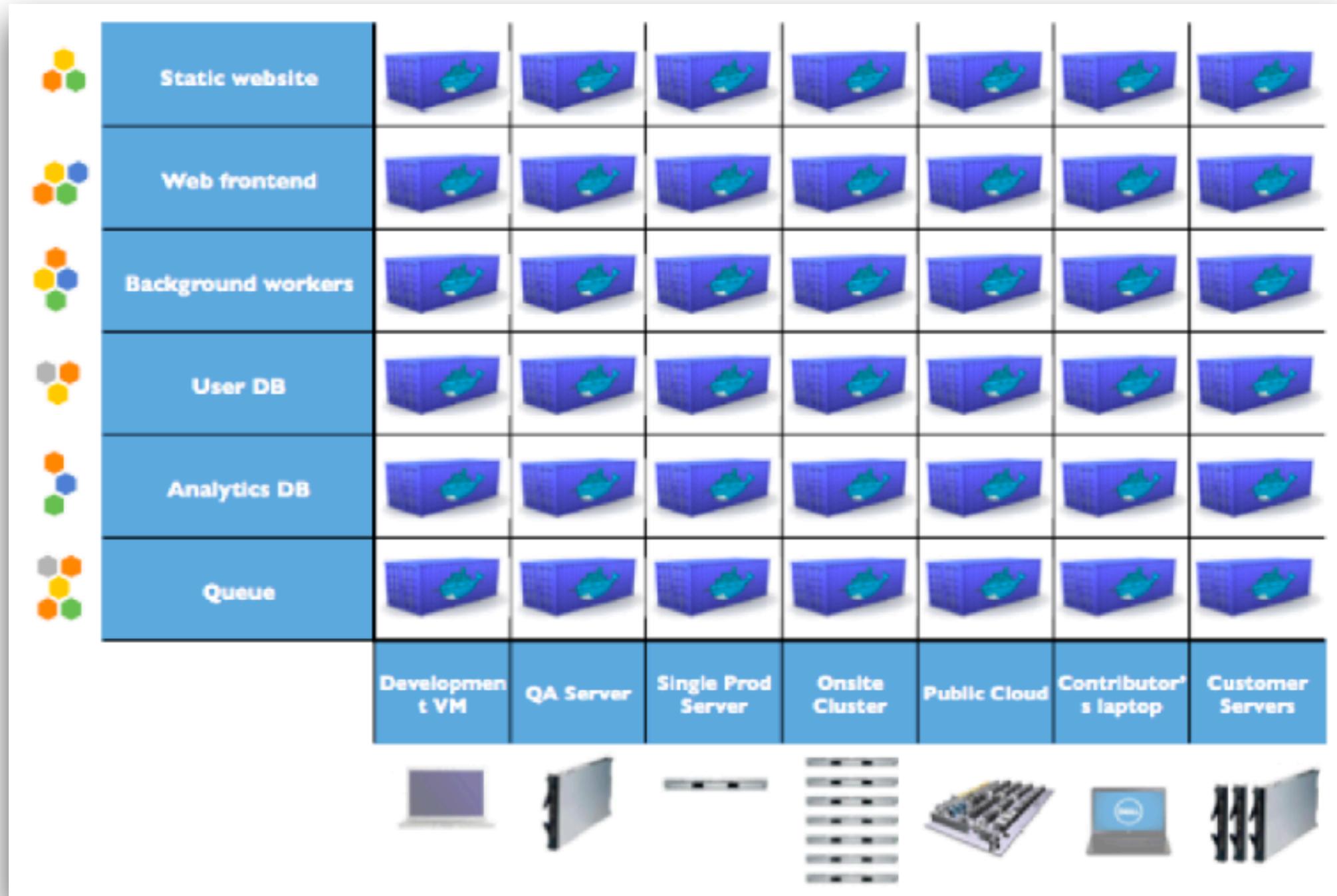
	Static website	?	?	?	?	?	?	?
	Web frontend	?	?	?	?	?	?	?
	Background workers	?	?	?	?	?	?	?
	User DB	?	?	?	?	?	?	?
	Analytics DB	?	?	?	?	?	?	?
	Queue	?	?	?	?	?	?	?
	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers	
								



# Solution ?

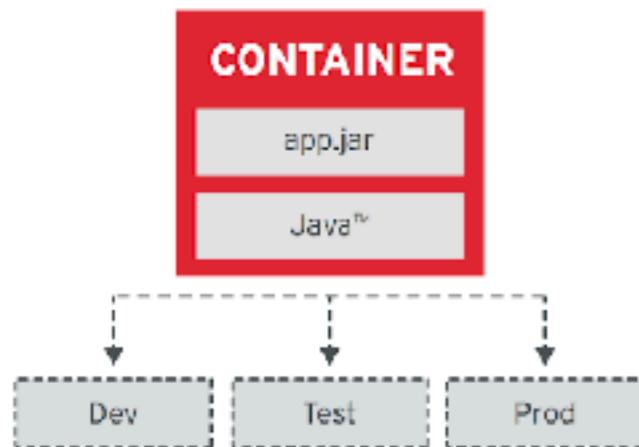


# Solution ?



# Container Design Principles

Image Immutability Principle



High Observability Principle



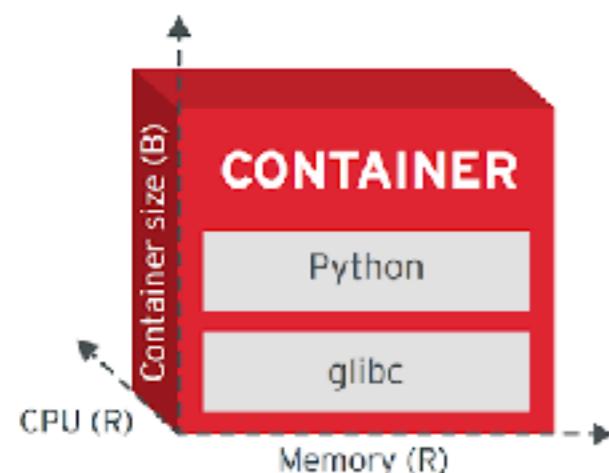
Process Disposability Principle



Lifecycle Conformance Principle



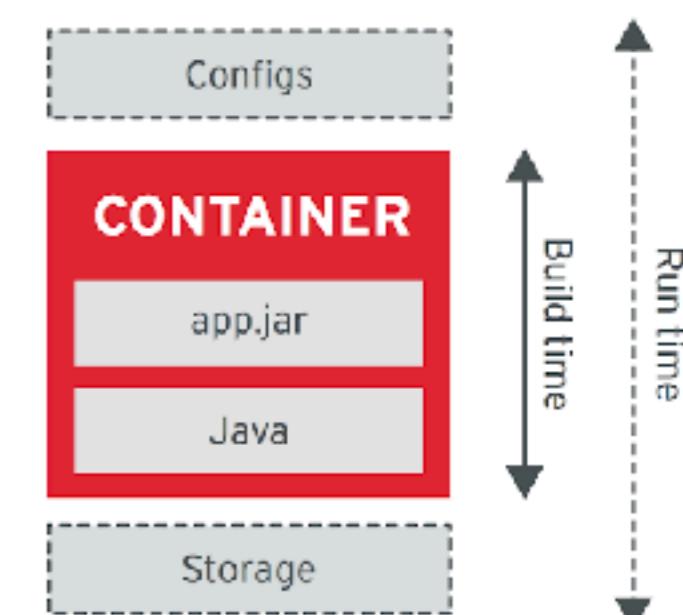
Runtime Confinement Principle



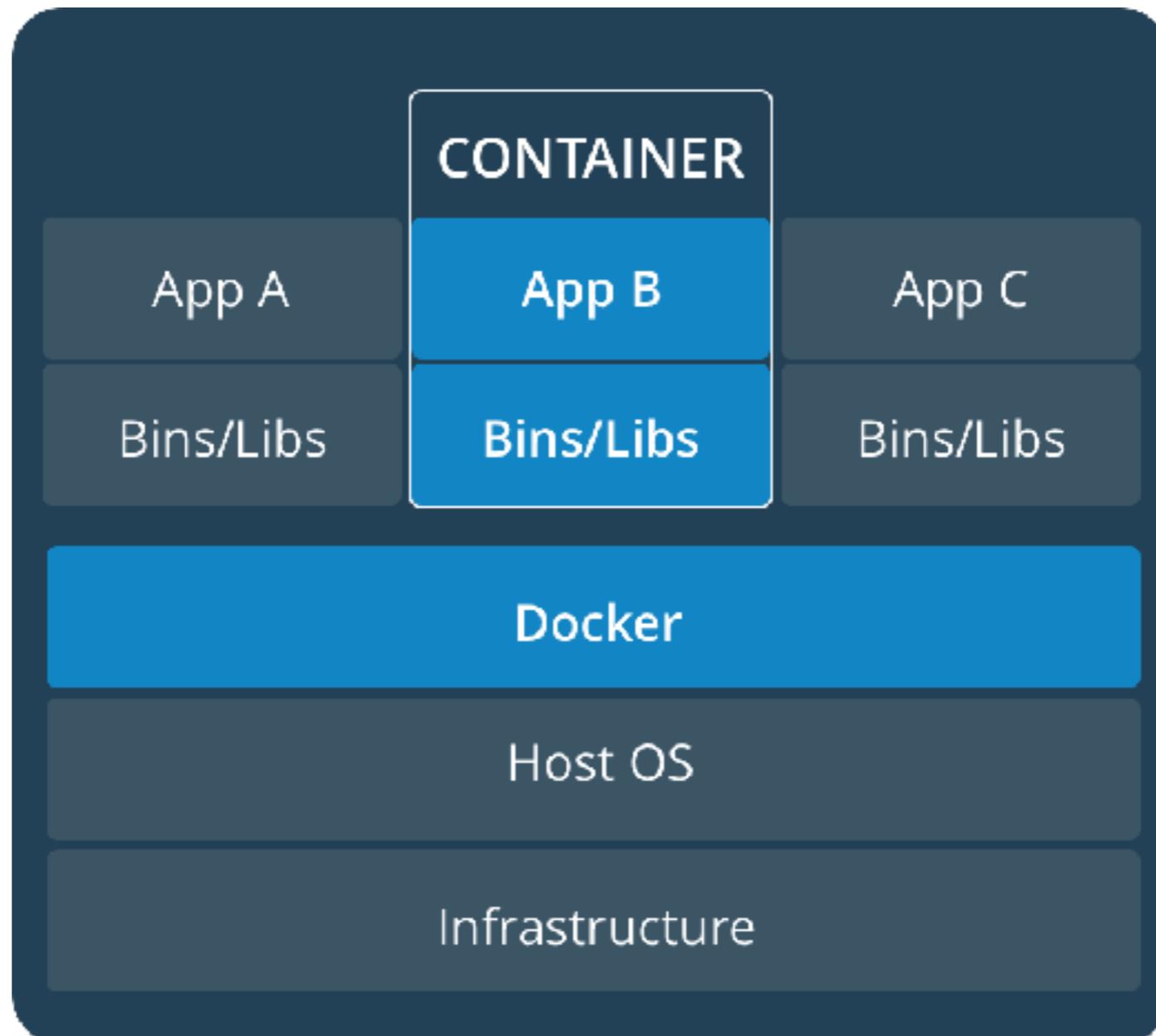
Single Concern Principle



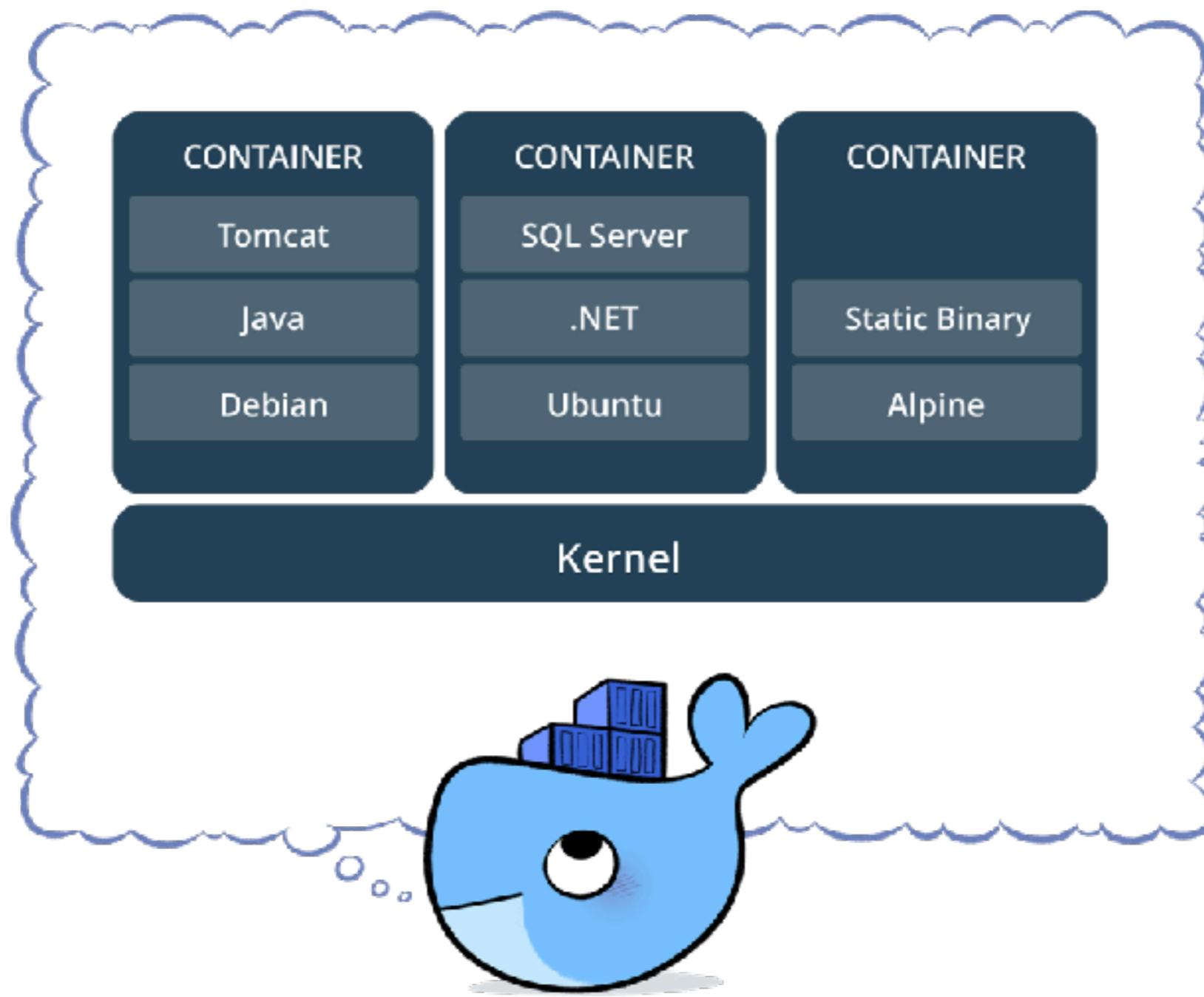
Self-Containment Principle



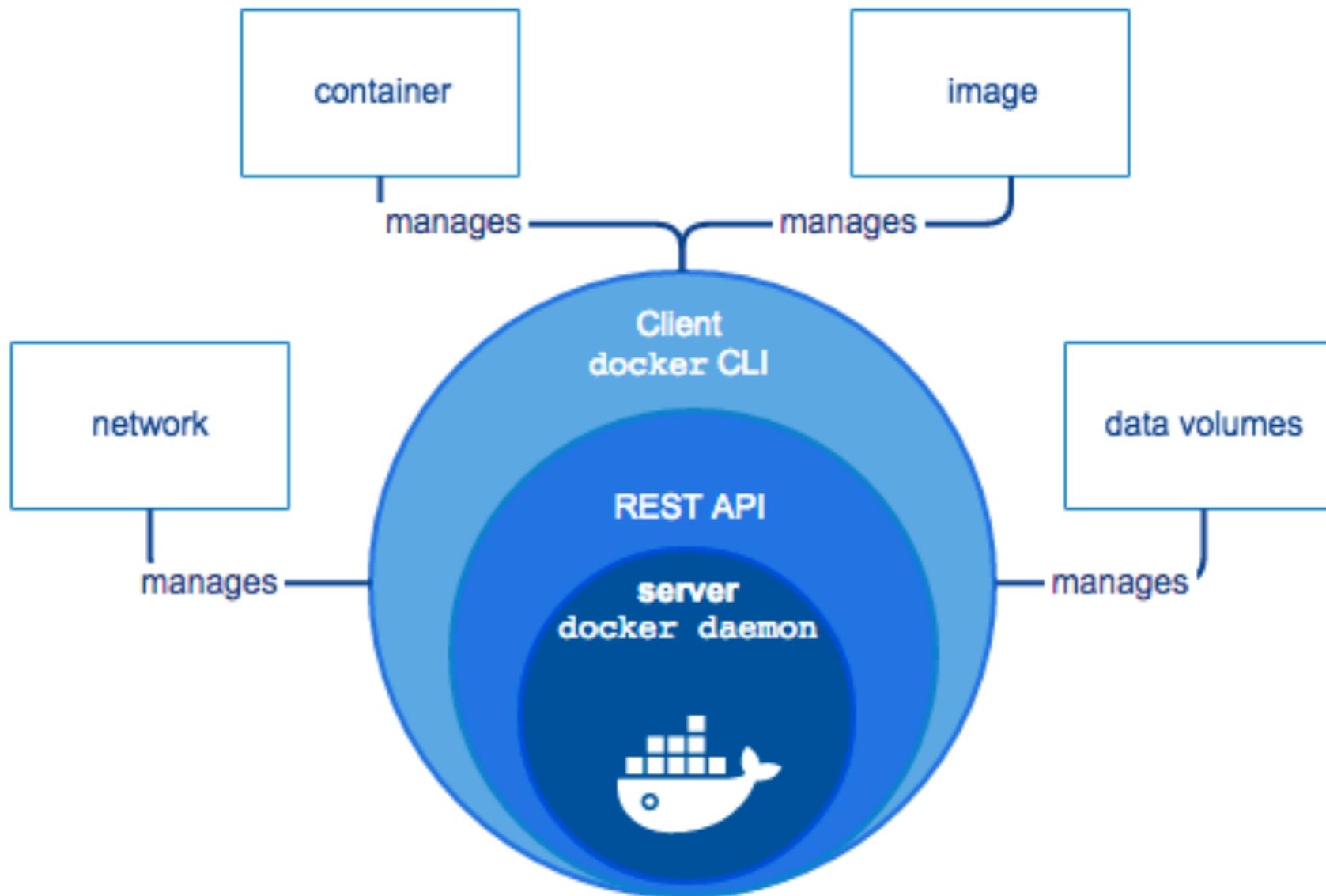
# Container with Docker



# Container with Docker



# Basic of Docker

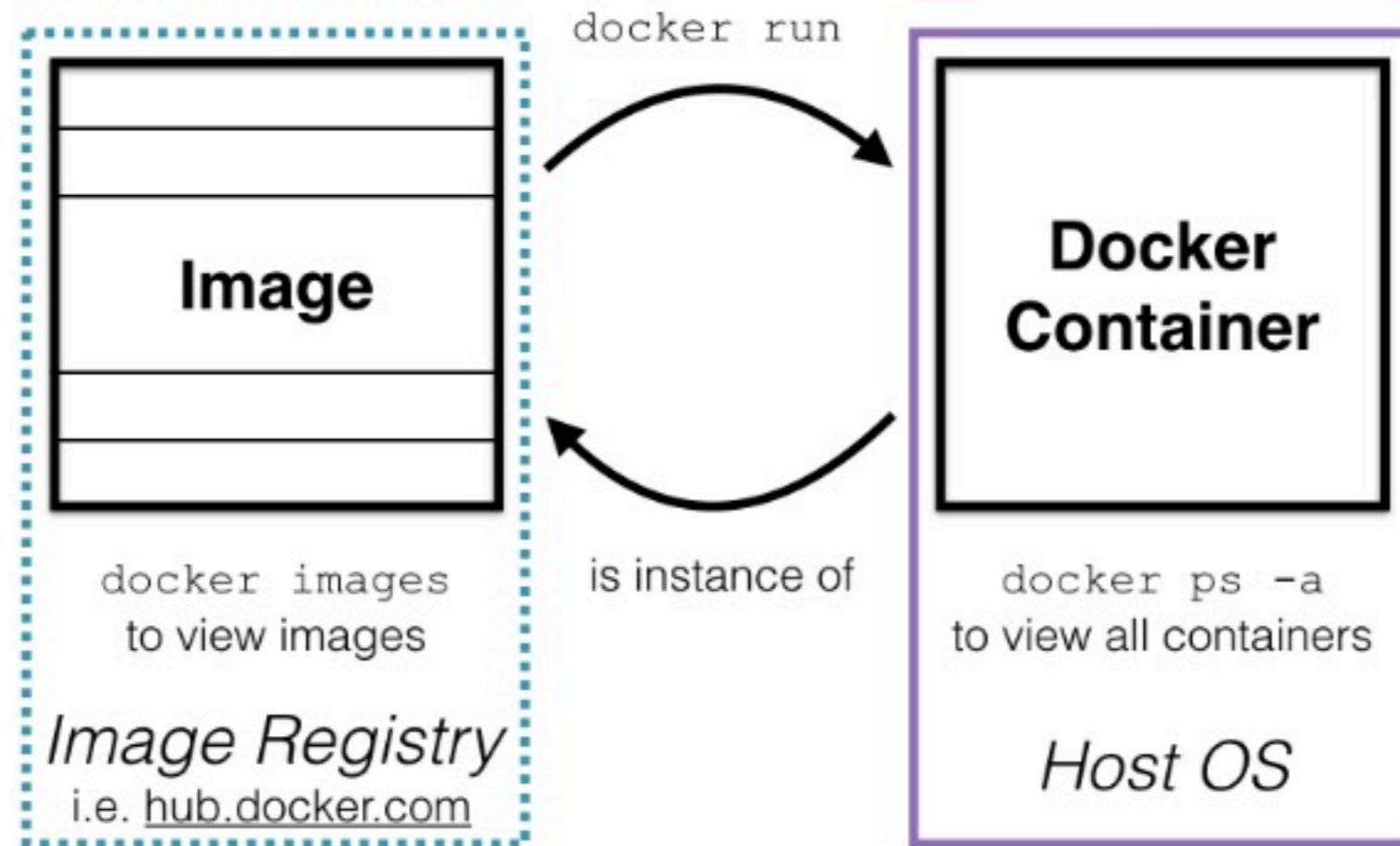


# **Basic of Docker**

**Image  
Container  
Dockerfile  
Registry  
Volume and network**



# Image vs Container



# Image vs Container

**Image** like template/blueprint  
**Containers** are created from image



# Docker image

Collection of files and some meta data

Made of **layers**

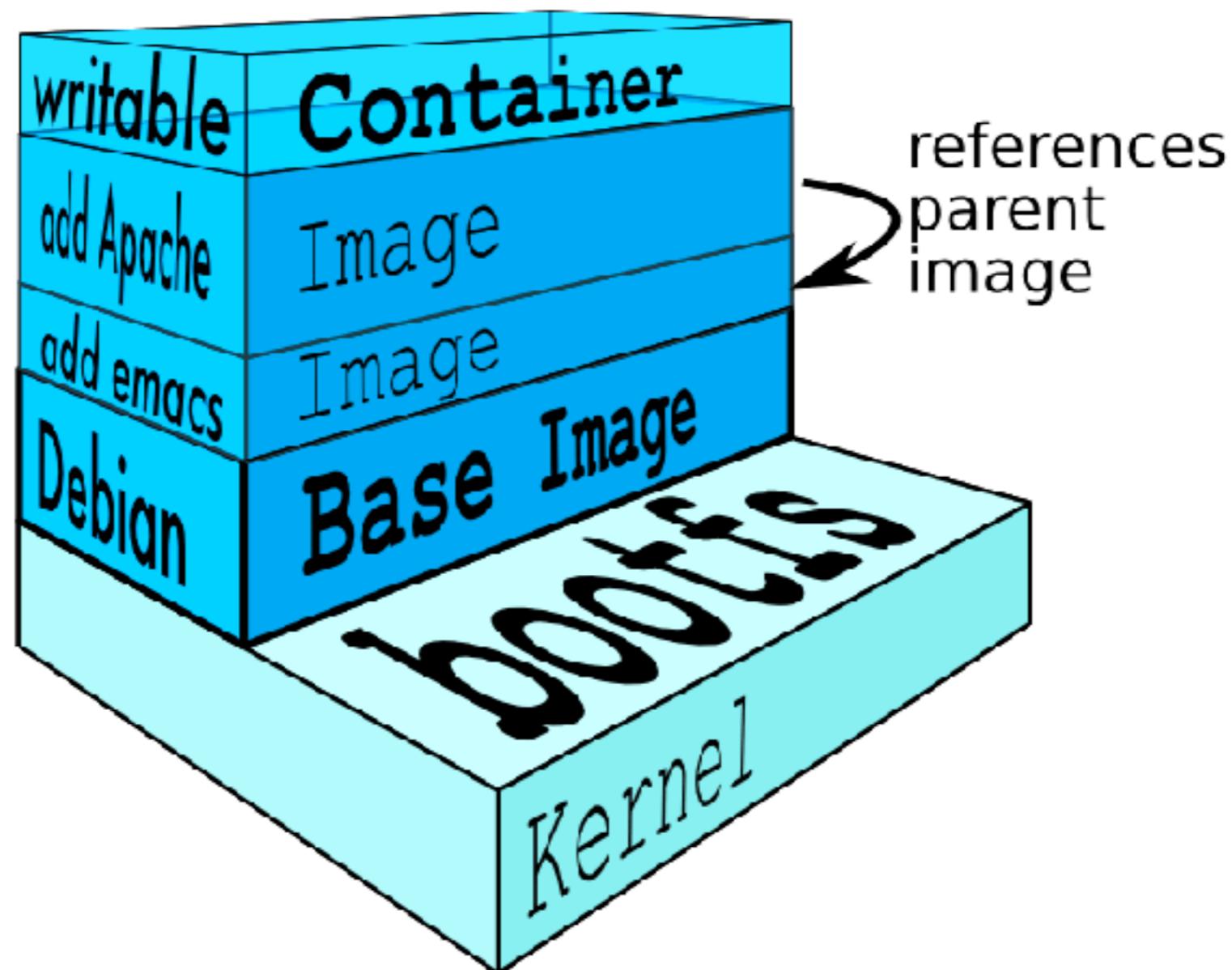
Each layer can add/change/remove files

Image can share layers

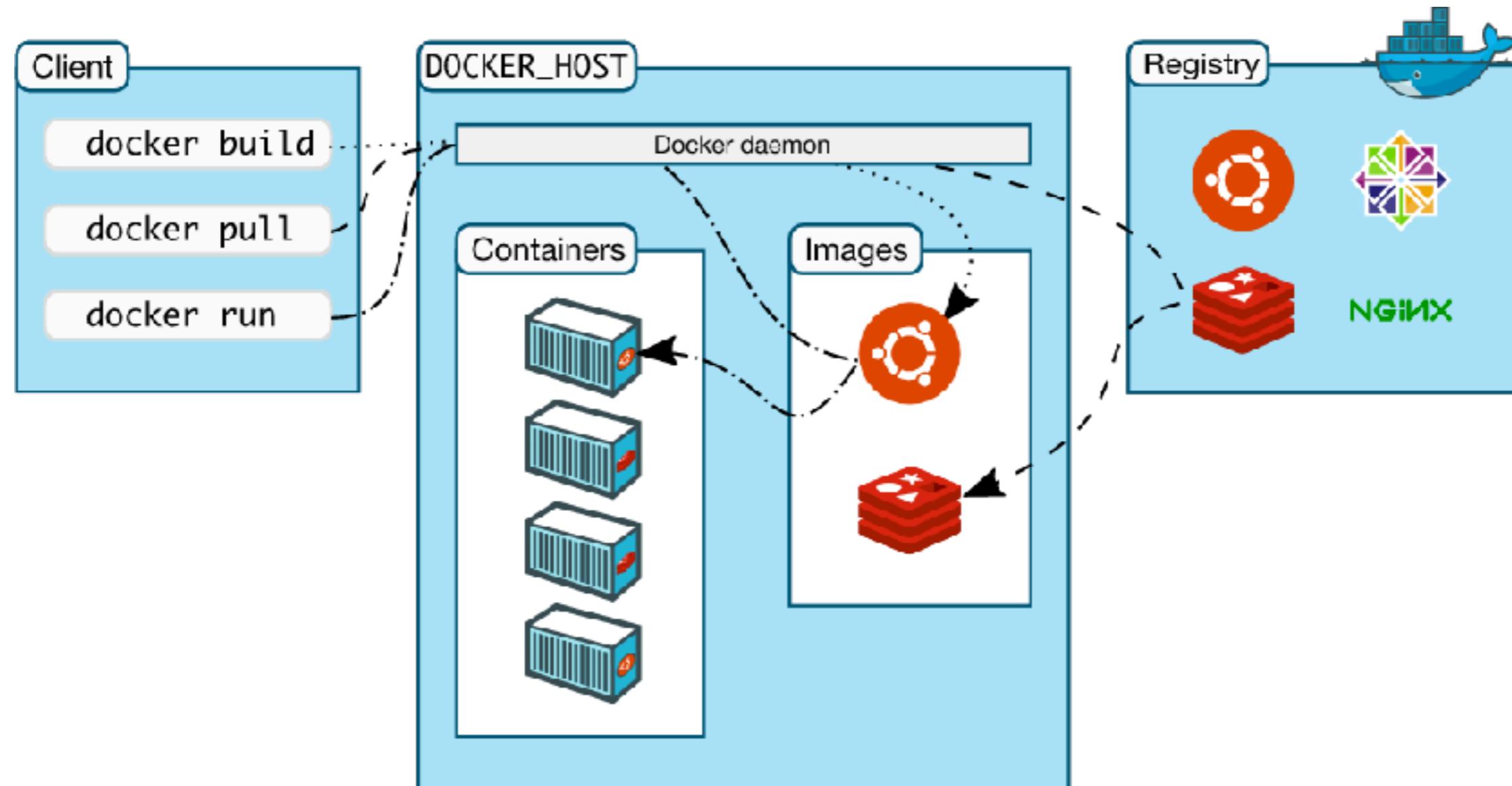
**Read-only** file system



# Docker image layer



# How docker works ?



<https://docs.docker.com/get-started/overview/>



# Docker commands

Manage image  
Manage container  
Manage system



# Management commands

```
$docker image <command>  
$docker container <command>  
$docker system <command>
```



# Working with Nginx

OFFICIAL REPOSITORY

[nginx](#) 

Last pushed: 20 days ago

[Repo Info](#) [Tags](#)

**Short Description**

Official build of Nginx.

**Docker Pull Command** 

```
docker pull nginx
```

**Full Description**

Supported tags and respective [Dockerfile](#) links

- `1.11.10, mainline, 1, 1.11, latest` ([mainline/jessie/Dockerfile](#))
- `1.11.10-alpine, mainline-alpine, 1-alpine, 1.11-alpine, alpine` ([mainline/alpine/Dockerfile](#))
- `1.10.3, stable, 1.10` ([stable/jessie/Dockerfile](#))
- `1.10.3-alpine, stable-alpine, 1.10-alpine` ([stable/alpine/Dockerfile](#))

[https://hub.docker.com/\\_/nginx/](https://hub.docker.com/_/nginx/)



# Pull image

\$ docker image pull nginx:latest

```
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
693502eb7dfb: Pull complete
6decb850d2bc: Pull complete
c3e19f087ed6: Pull complete
Digest: sha256:52a189e49c0c797cf5cbfe578c68c225d160fb13a42954144b29af3fe4fe335
Status: Downloaded newer image for nginx:latest
```



# Pull image

```
$docker image pull nginx:latest
```



Tag



# Image and Tag

Image can have **tags**

Tags define image variants

Default of tag is **:latest**

***:latest tag can be updated frequently !!***



# Docker container

Foreground  
Background  
Interactive



# Run with foreground

```
$ docker container run nginx
```



# Run with background

```
$ docker container run -d nginx
```

*-d = --detach*

*Run container in background and print container ID*



# Run with interactive

\$ docker container run -it nginx bash

*-i = --interactive*

*-t = --tty*



# Access to container

\$docker container **run -it**

\$docker container **exec -it**



# Create Image from Dockerfile



# Dockerfile

Build recipe for a Docker image

Contain series of instructions

Use **\$docker image build** command

<https://docs.docker.com/engine/reference/builder/>



# Example

```
FROM ubuntu
RUN apt-get update
RUN apt-get install -y wget
```



# Build image from Dockerfile

```
$ docker image build -t first_image .
```

```
Sending build context to Docker daemon 2.048 kB
Step 1/3 : FROM ubuntu
    ----> 0ef2e08ed3fa
Step 2/3 : RUN apt-get update
    ----> Running in 6c598d2946b7
Get:1 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://archive.ubuntu.com/ubuntu xenial-updates InRelease
Get:3 http://archive.ubuntu.com/ubuntu xenial-security InRelease
Get:4 http://archive.ubuntu.com/ubuntu xenial/main Sources
Get:5 http://archive.ubuntu.com/ubuntu xenial/restricted Sources
```



# History of image

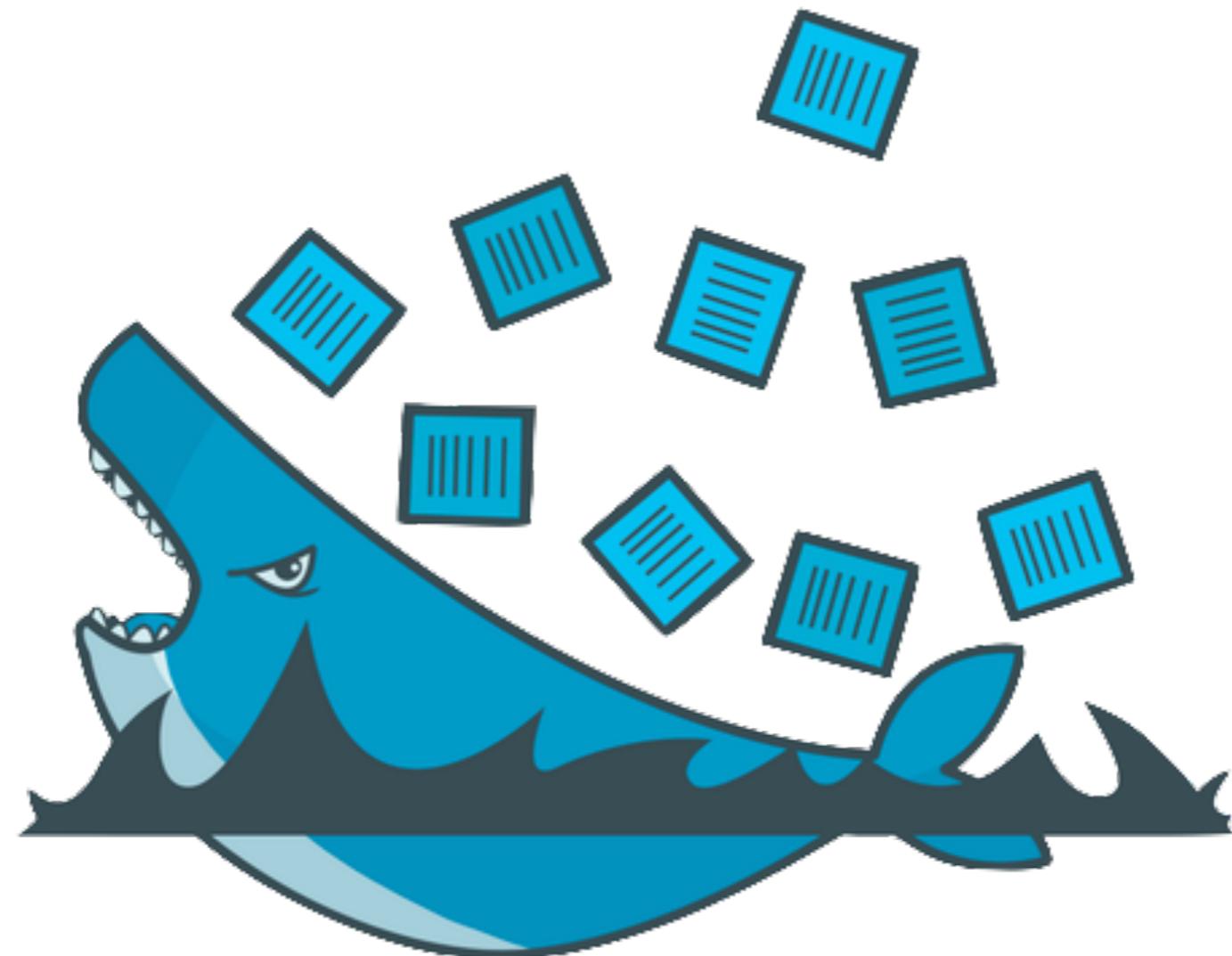
Show all layers of image

\$docker image history <image name>

IMAGE	CREATED	CREATED BY	SIZE
1813d5ecf658	4 minutes ago	/bin/sh -c apt-get install -y wget	7.35 MB
2930e9a322d6	5 minutes ago	/bin/sh -c apt-get update	40.1 MB
0ef2e08ed3fa	3 weeks ago	/bin/sh -c #(nop) CMD ["/bin/bash"]	0 B
<missing>	3 weeks ago	/bin/sh -c mkdir -p /run/systemd && echo '...'	7 B
<missing>	3 weeks ago	/bin/sh -c sed -i 's/^#\s*\(\deb.*universe\ ...)/.../g'	1.9 kB
<missing>	3 weeks ago	/bin/sh -c rm -rf /var/lib/apt/lists/*	0 B
<missing>	3 weeks ago	/bin/sh -c set -xe && echo '#!/bin/sh' >.../etc/init.d/docker	745 B
<missing>	3 weeks ago	/bin/sh -c #(nop) ADD file:efb254bc677d66d.../etc/init.d/docker	130 MB



# Workshop



# Working with Docker compose

<https://docs.docker.com/compose/>



# Mutilple containers app!!

Difficult to create and manage !!



# Mutilple containers app!!

Build images from Dockerfile

Pull images from Hub/private/cache

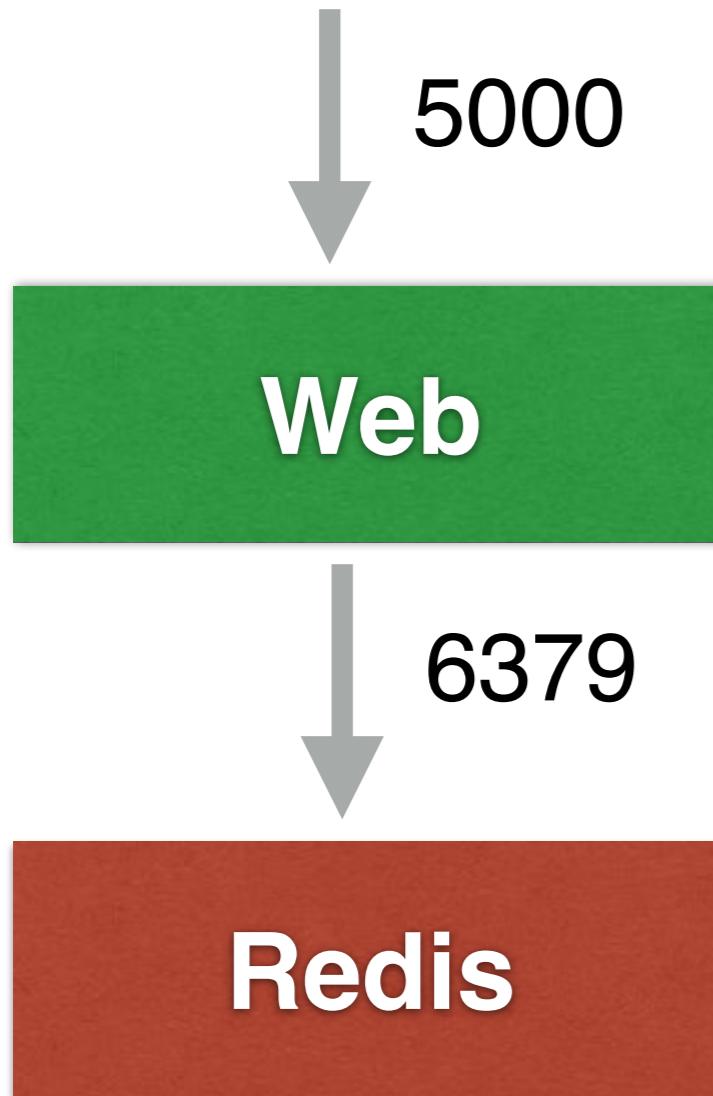
Configuration and create containers

Start and stop containers

Stream their logs



# Mutilple containers app!!



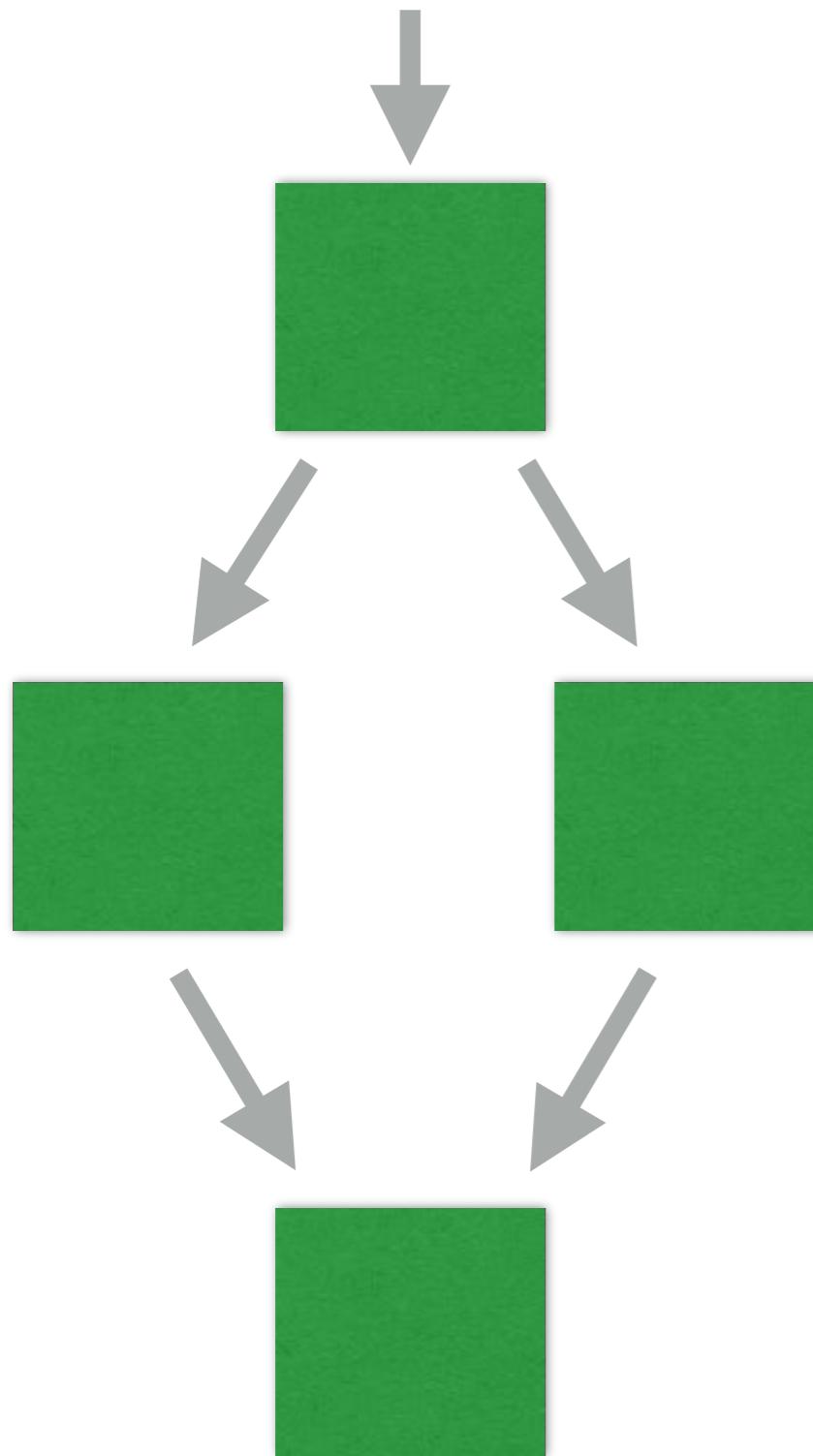
```
$docker pull redis:latest  
$docker build -t web .
```

```
$docker run -d --name db redis  
redis-server --appendonly yes
```

```
$docker run -d --name web --link  
db:db -p 5000:5000 -e  
REDIS_HOST=db -v $(pwd):/code  
web
```



# Mutilple containers app!!



\$docker pull

\$docker build

\$docker build

\$docker build

\$docker run

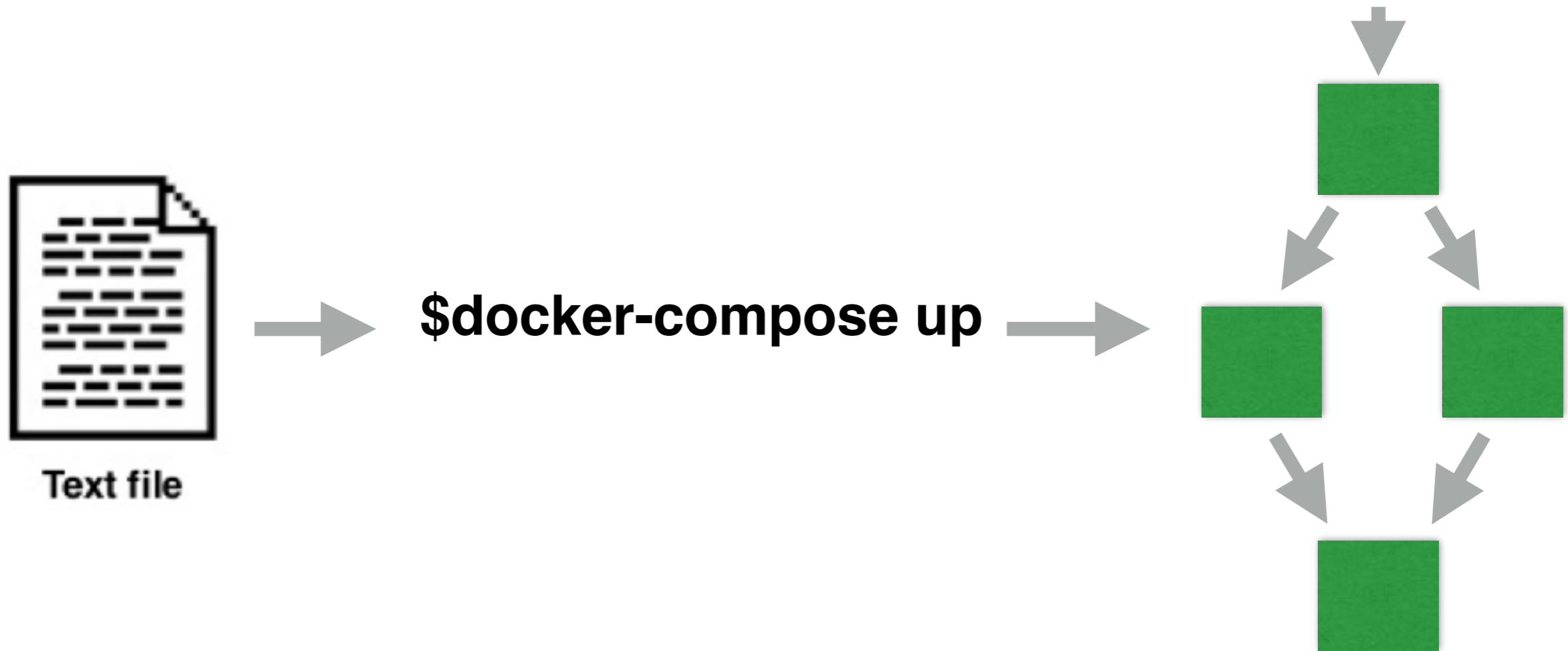
\$docker run

\$docker run



# Docker compose

Running app in one command-line  
Configuration in YAML file



# docker-compose.yml

```
version: '3'  
services:  
  web:  
    container_name: web  
    build: .  
    ports:  
      - 80:5000  
    environment:  
      - REDIS_HOST=redis  
  
  redis:  
    container_name: redis  
    image: redis
```



# Build and run

\$docker-compose build

\$docker-compose up -d

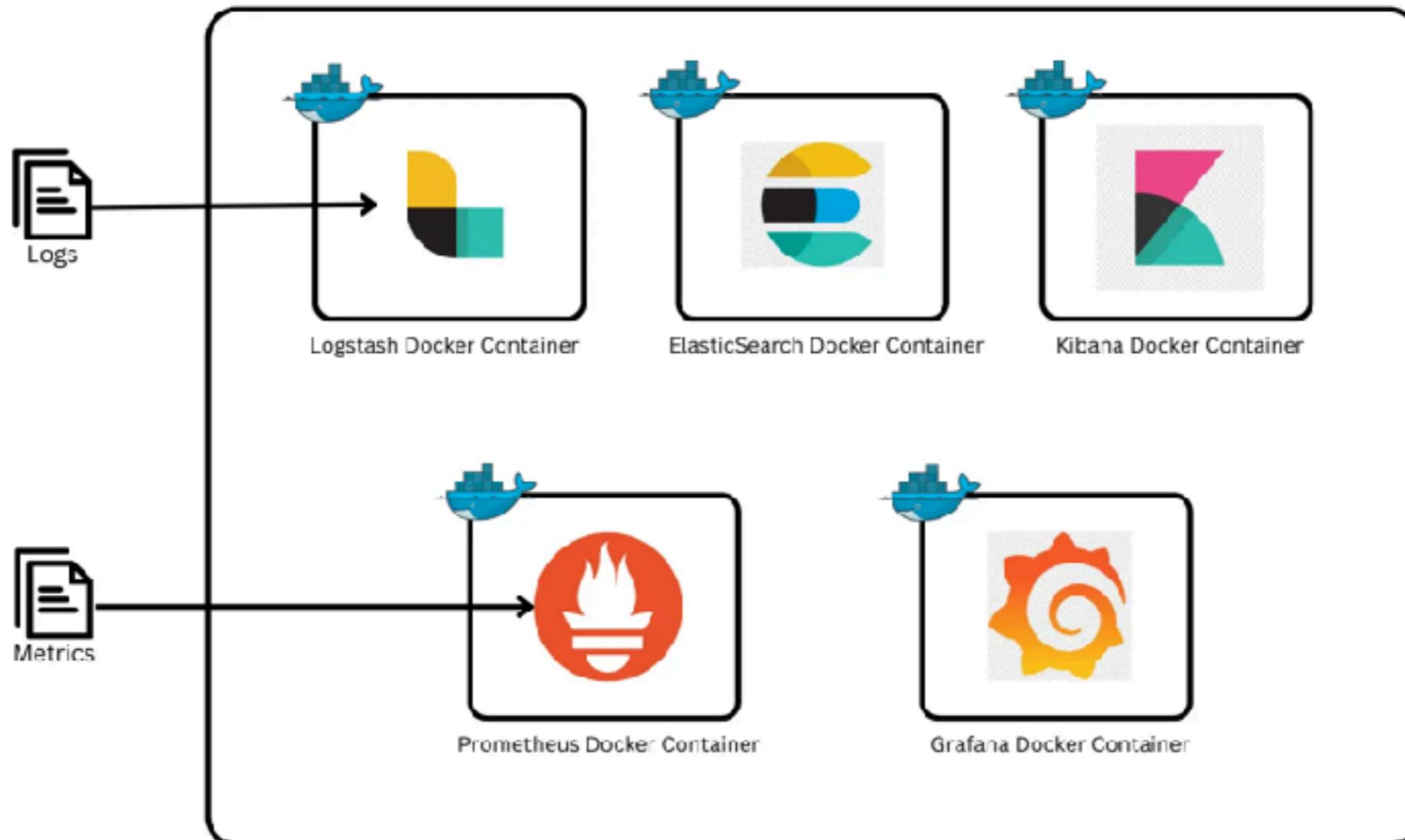
\$docker-compose logs --follow

\$docker-compose ps

\$docker-compose down



# Monitoring with Docker



# Workshop



# Introduction to k8s



**kubernetes**



# Why we need Kubernetes ?

Managing containers for production is challenging

Need something to manage beyond a container engine !!





# Key capabilities was missing

Using multiple containers with shared resources

- Monitoring running containers

- Handling dead containers

- Moving containers so utilization improves



# Key capabilities was missing

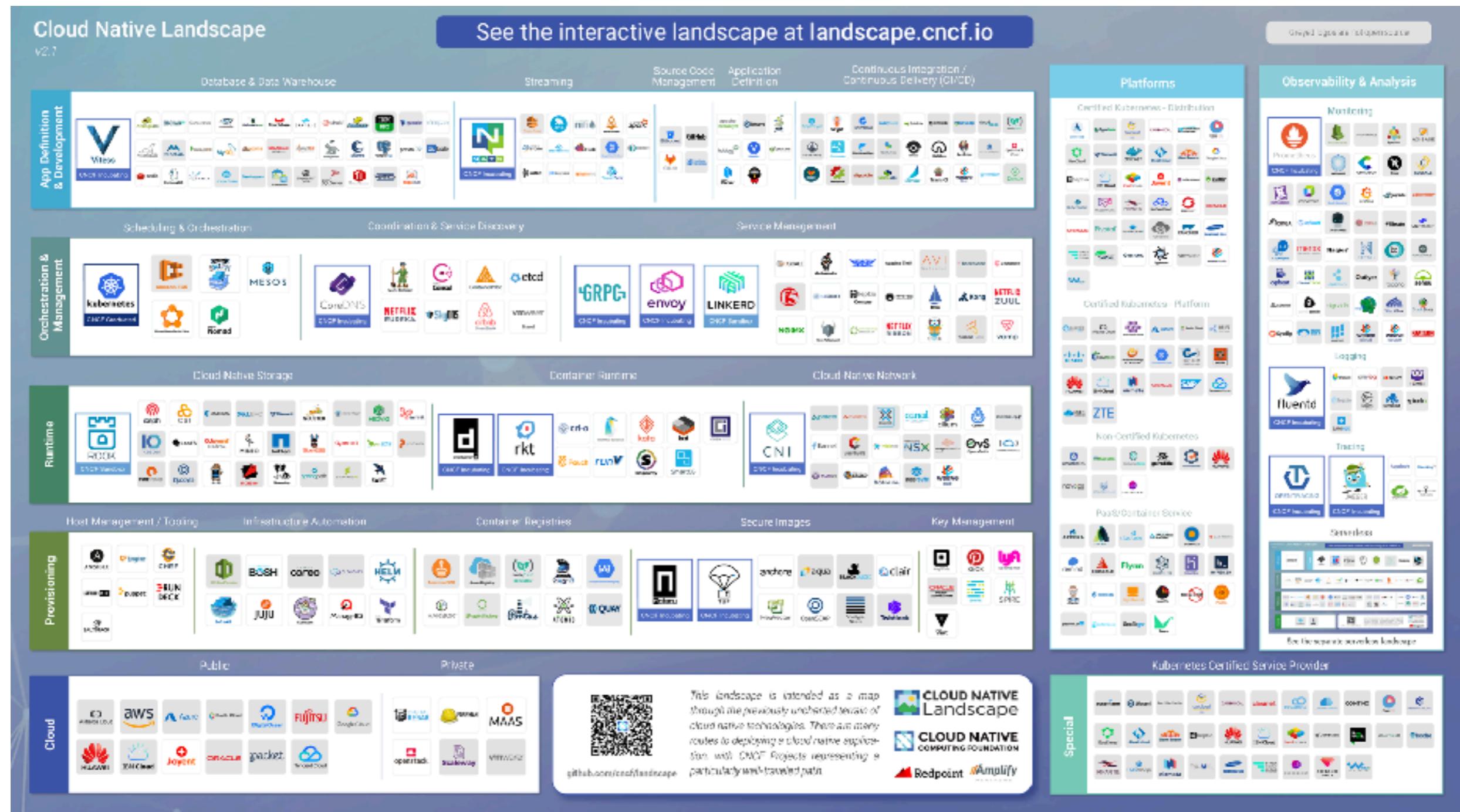
Autoscaling container instances to handle load

Making the container services easily accessible

Handling dead Connecting containers to a variety of  
external data sources



# Cloud Native Landscape



<https://github.com/cncf/landscape>

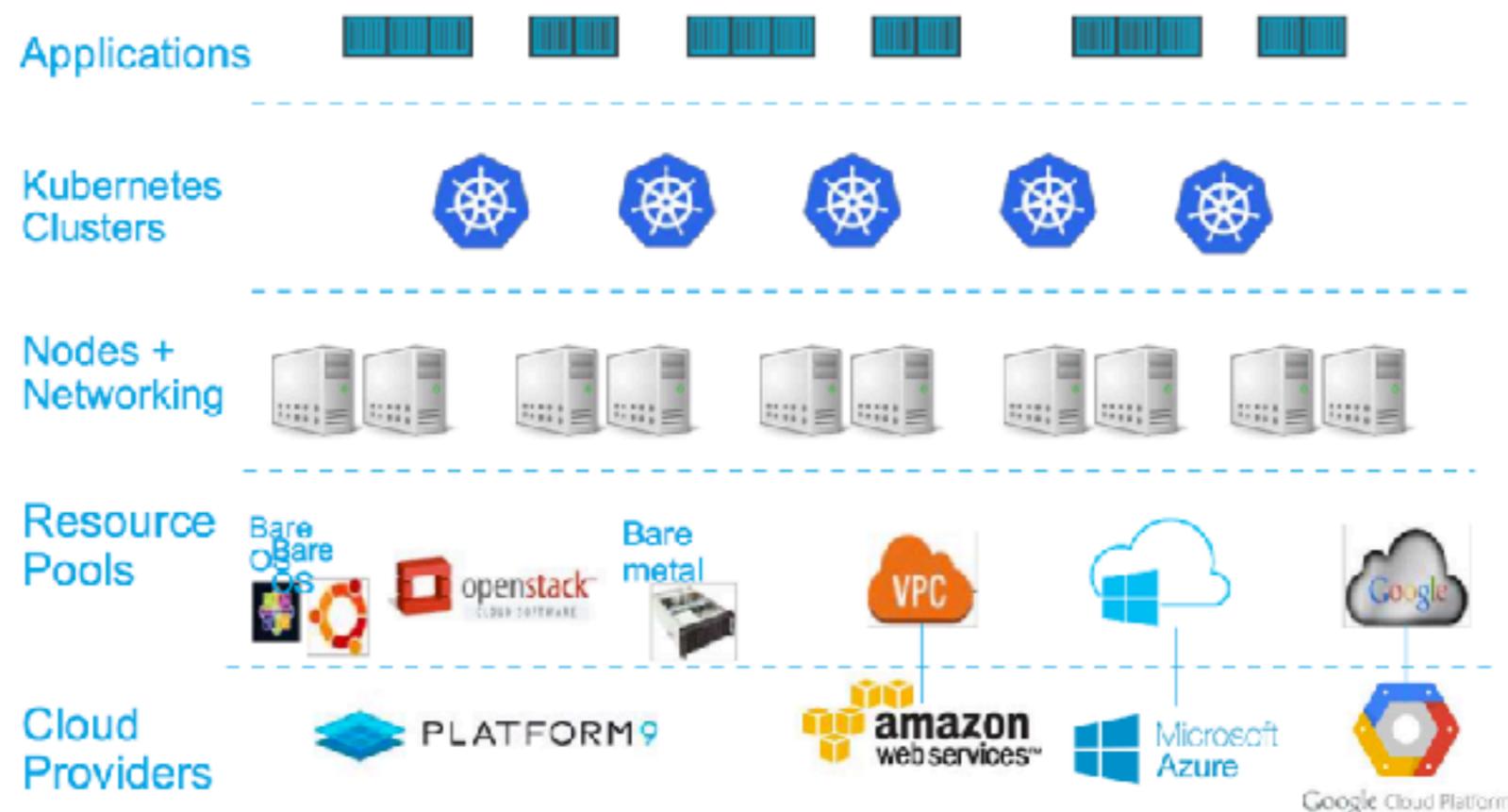


# Write once, run anywhere

Eliminate infrastructure lock-in

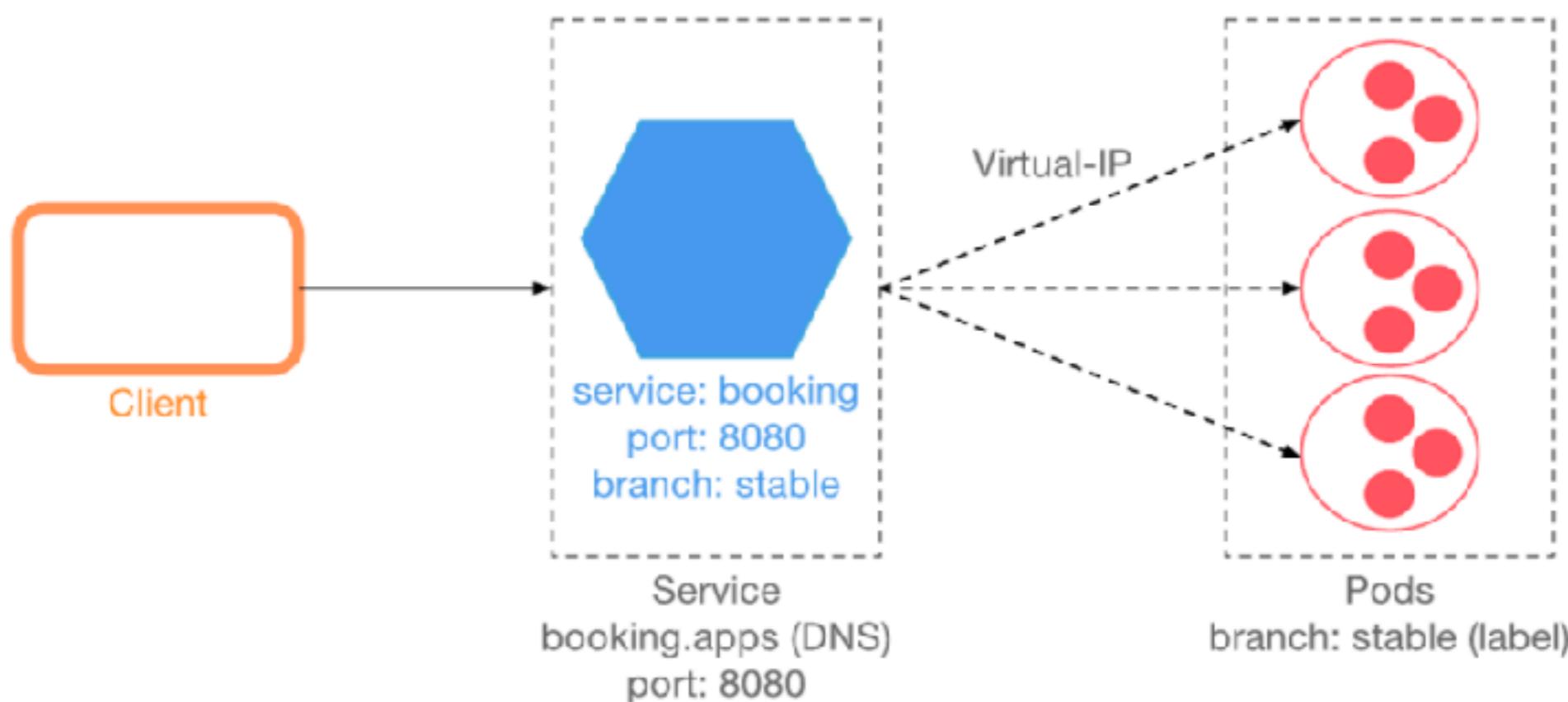
Use containers

Provides management for containers



# Modular app design

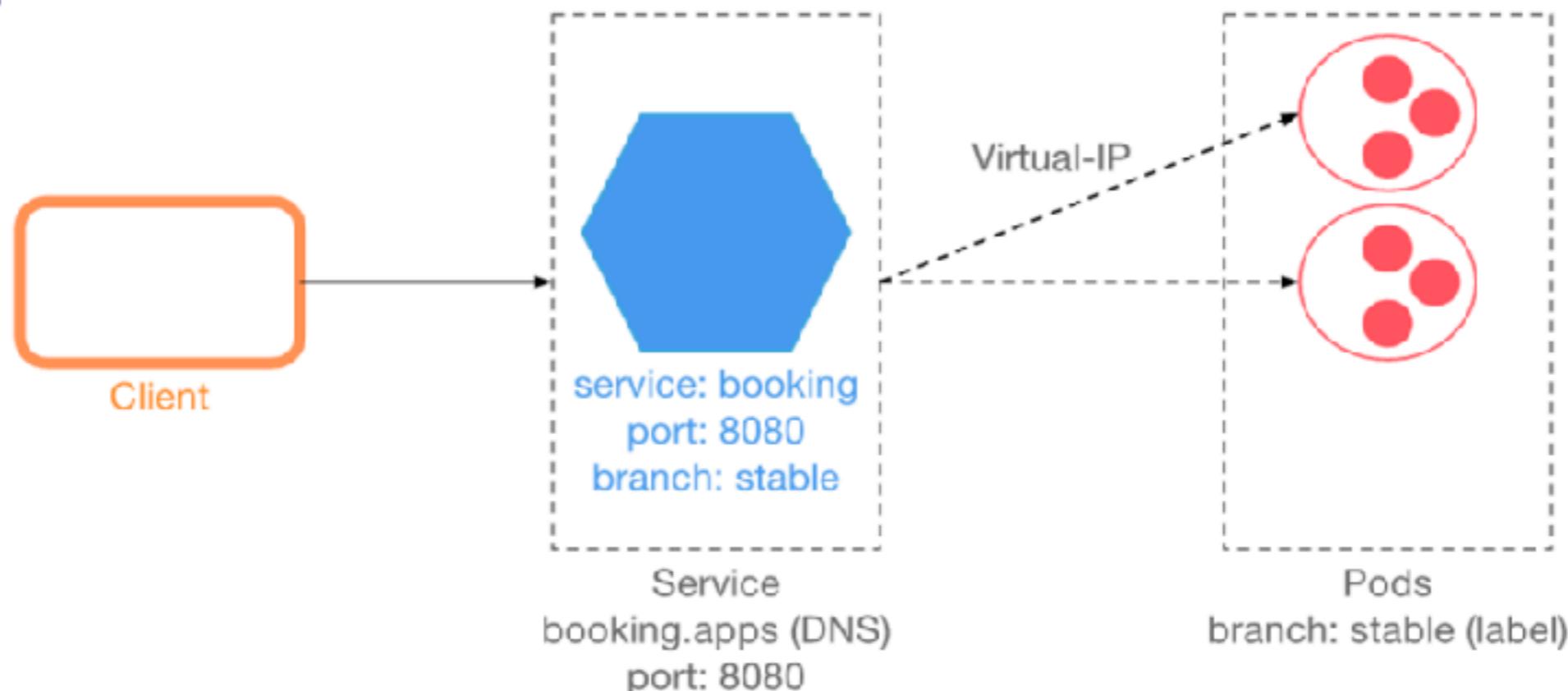
Container/Image boundary similar to class  
How to use/manage a collection of container ?



# Fault-tolerant by design

## Design for failure

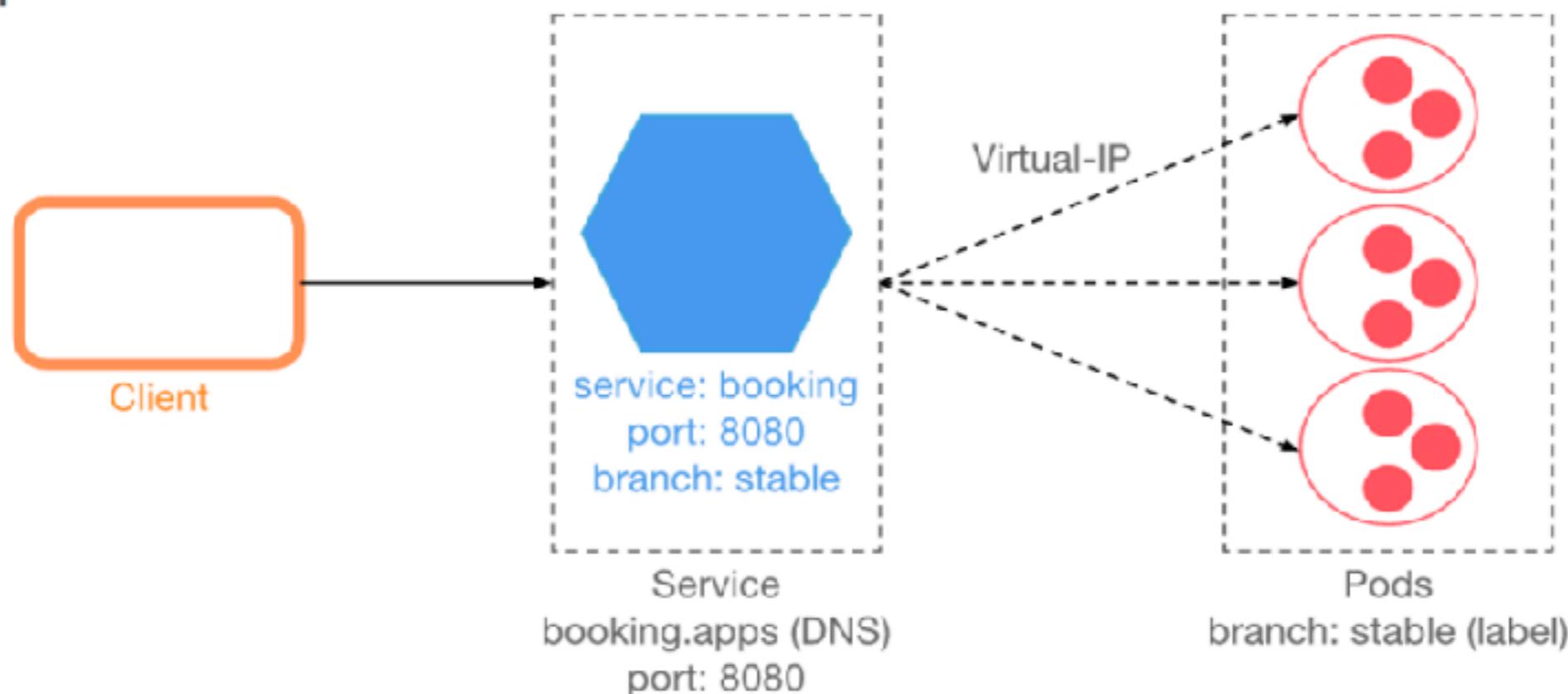
replicas = 2



# Fault-tolerant by design

## Design for failure

replicas = 3



# Deployment, not infrastructure

Software deployment is hard

Infrastructure provisioning/re-provisioning

Configuration networking and load balance

Redundancy (scale-out)

Lifecycle management (Software update)



# K8s support for deployment

Scale-out service

Rolling update for new version

Rollback to a previous version

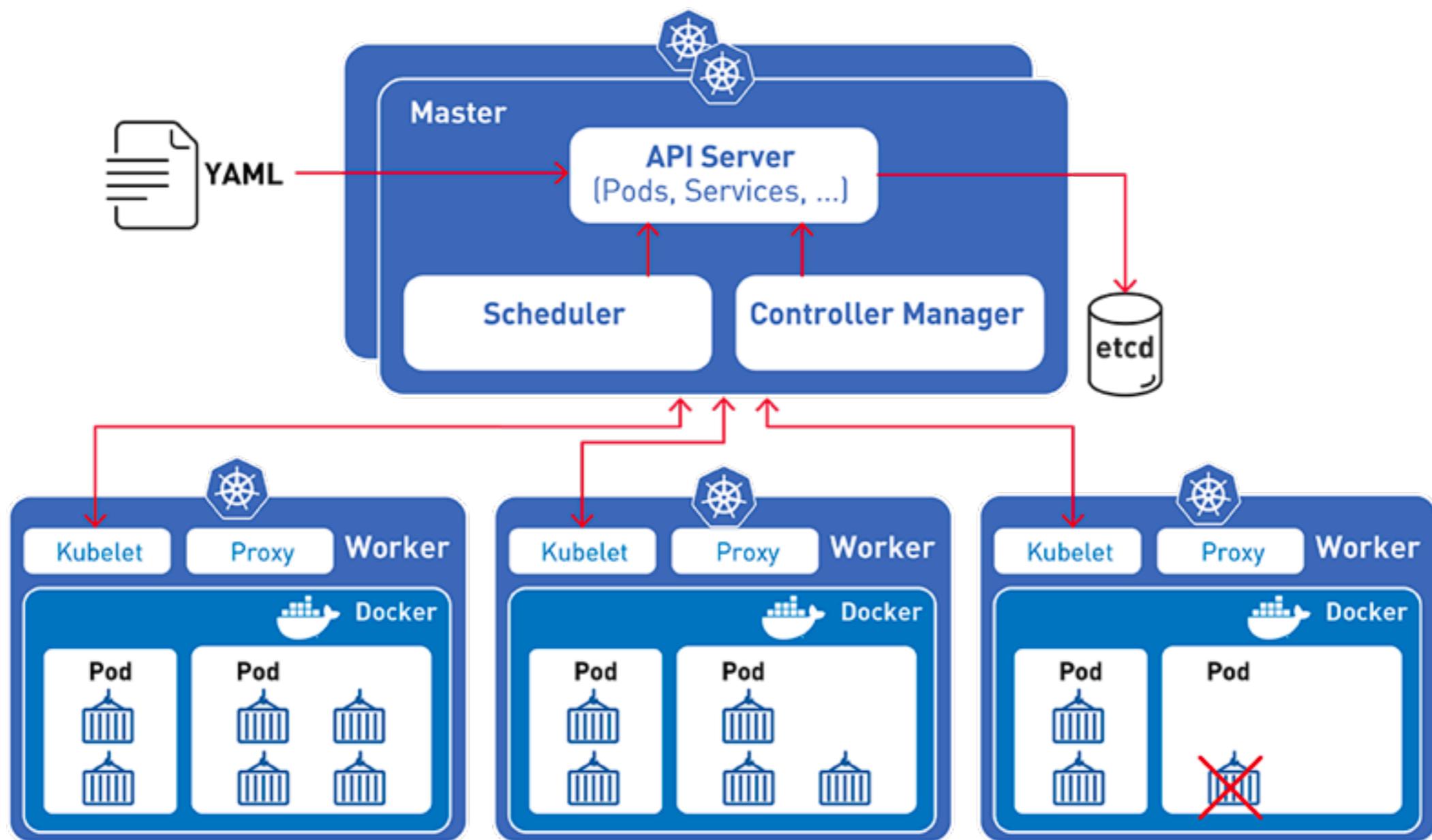
Pause and resume a deployment

Horizontal auto-scaling

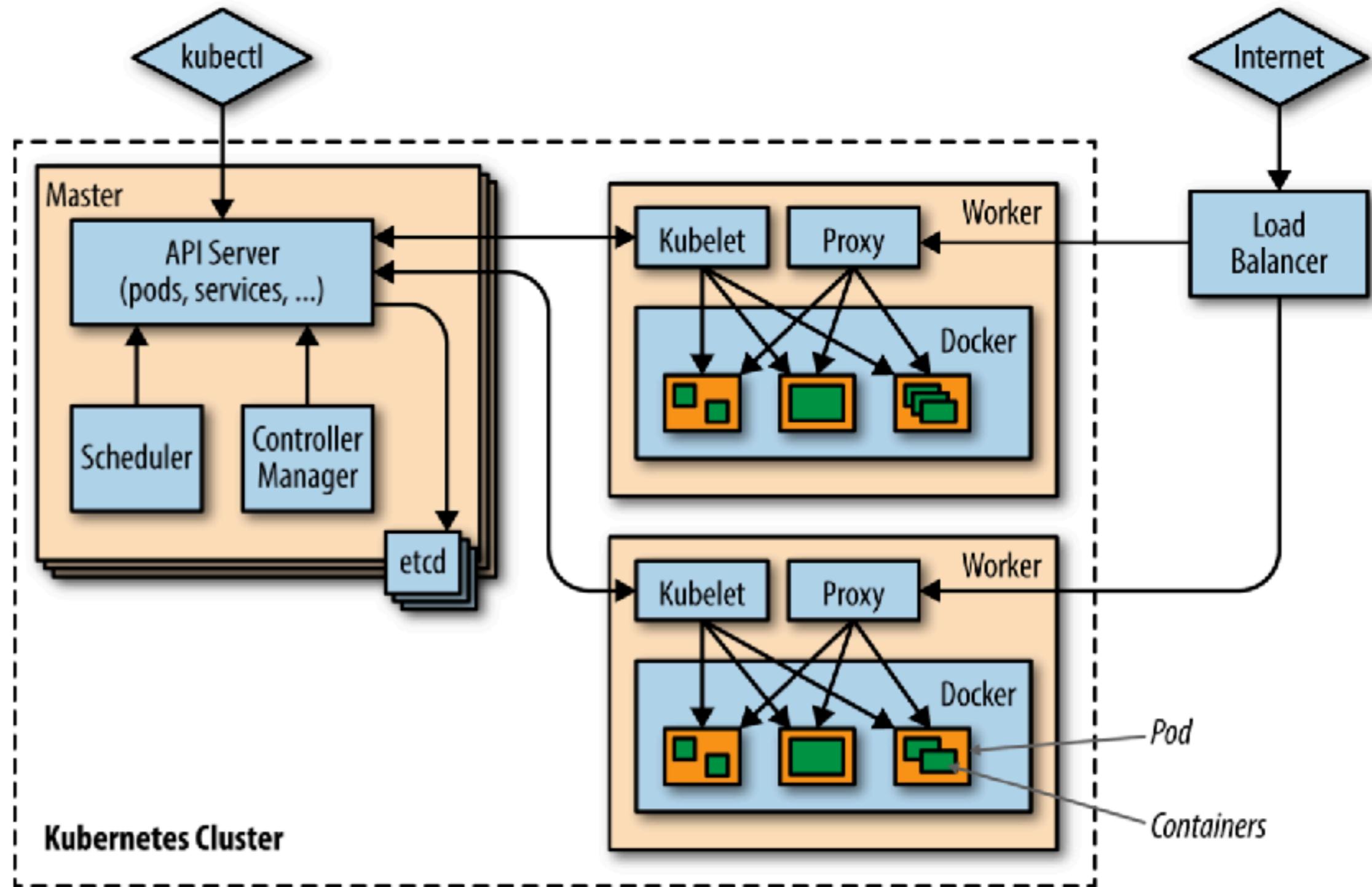
Canary deployment



# Kubernetes

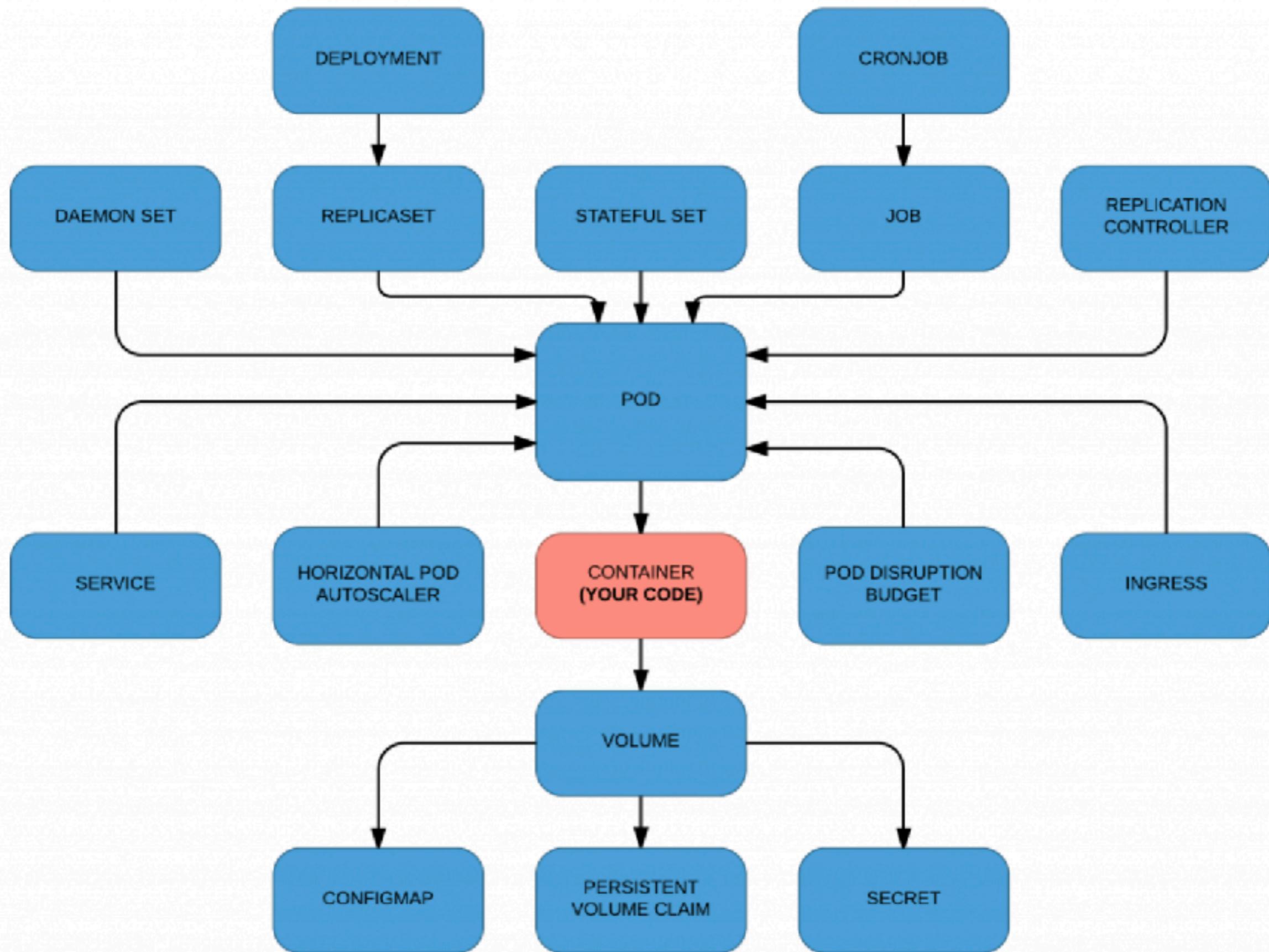


# Kubernetes Architecture



# K8s components





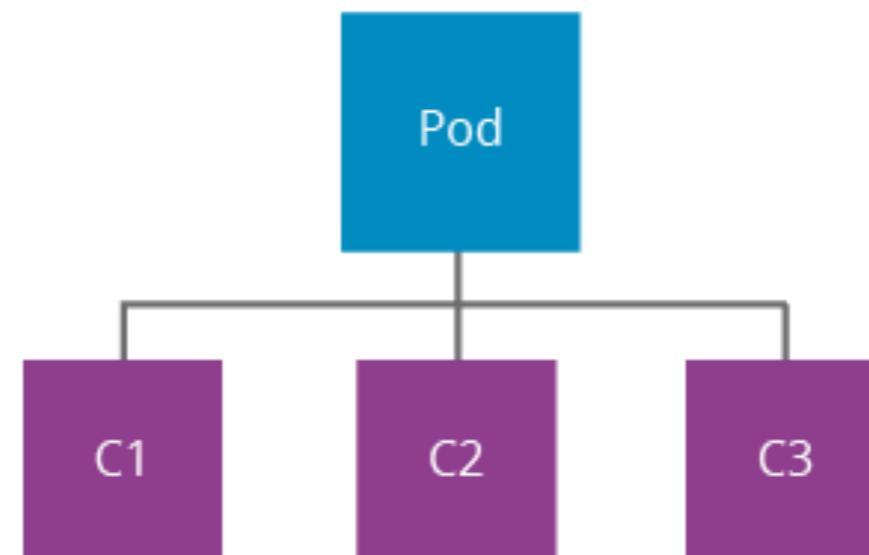
# K8s components

Pods  
Services  
Deployments  
ConfigMap and Secret  
Volumes  
StatefulSets



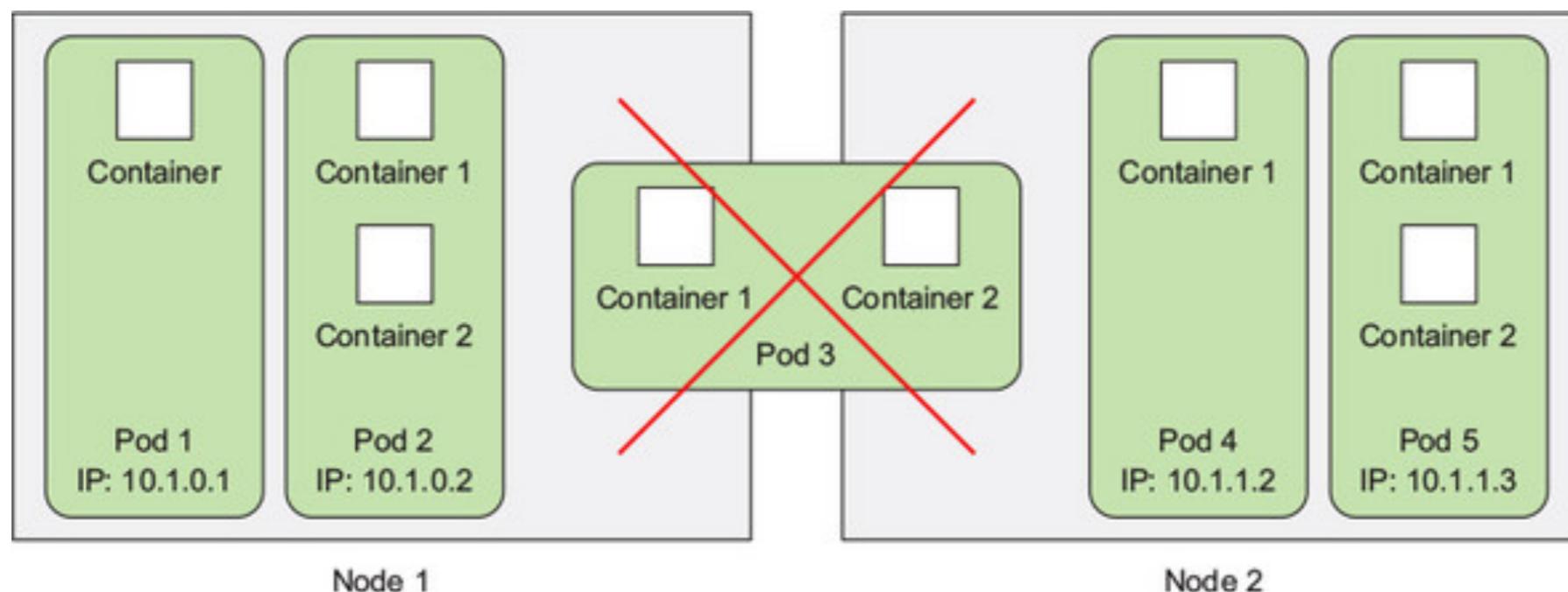
# Pods

Small group of co-located containers  
Optionally shared volume between containers  
Basic deployment unit in Kubernetes



# Pods

1 pods = 1 container  
1 pods = N containers



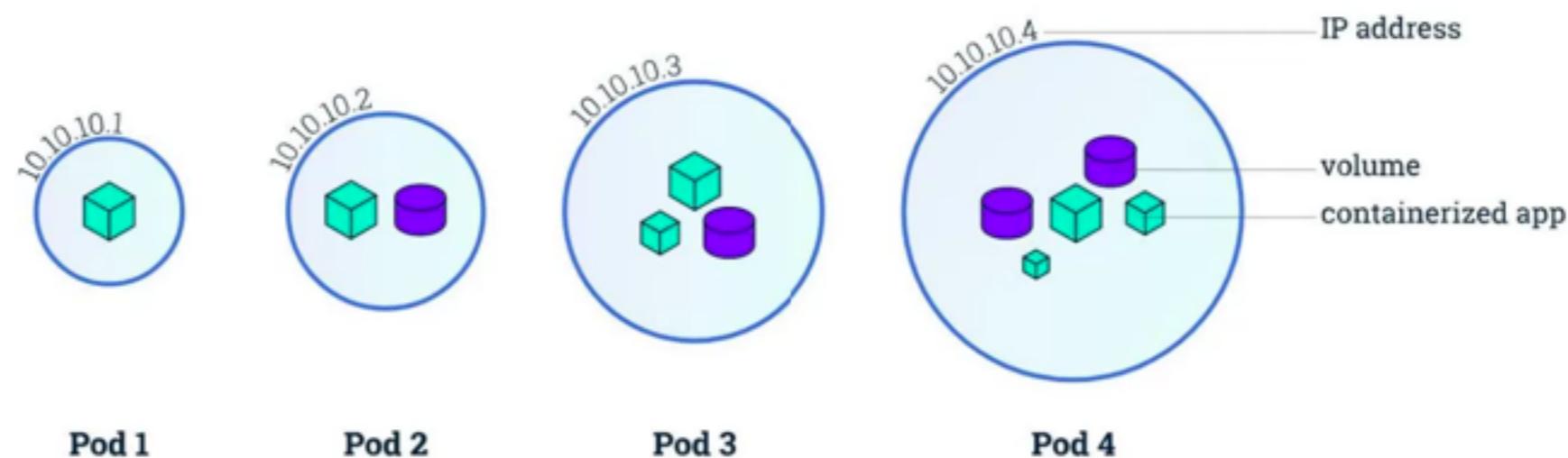
# All containers in same Pods

Share process ID

Share network interface

Share Hostname/IP/Port

Share Unix Time Sharing (UTS)



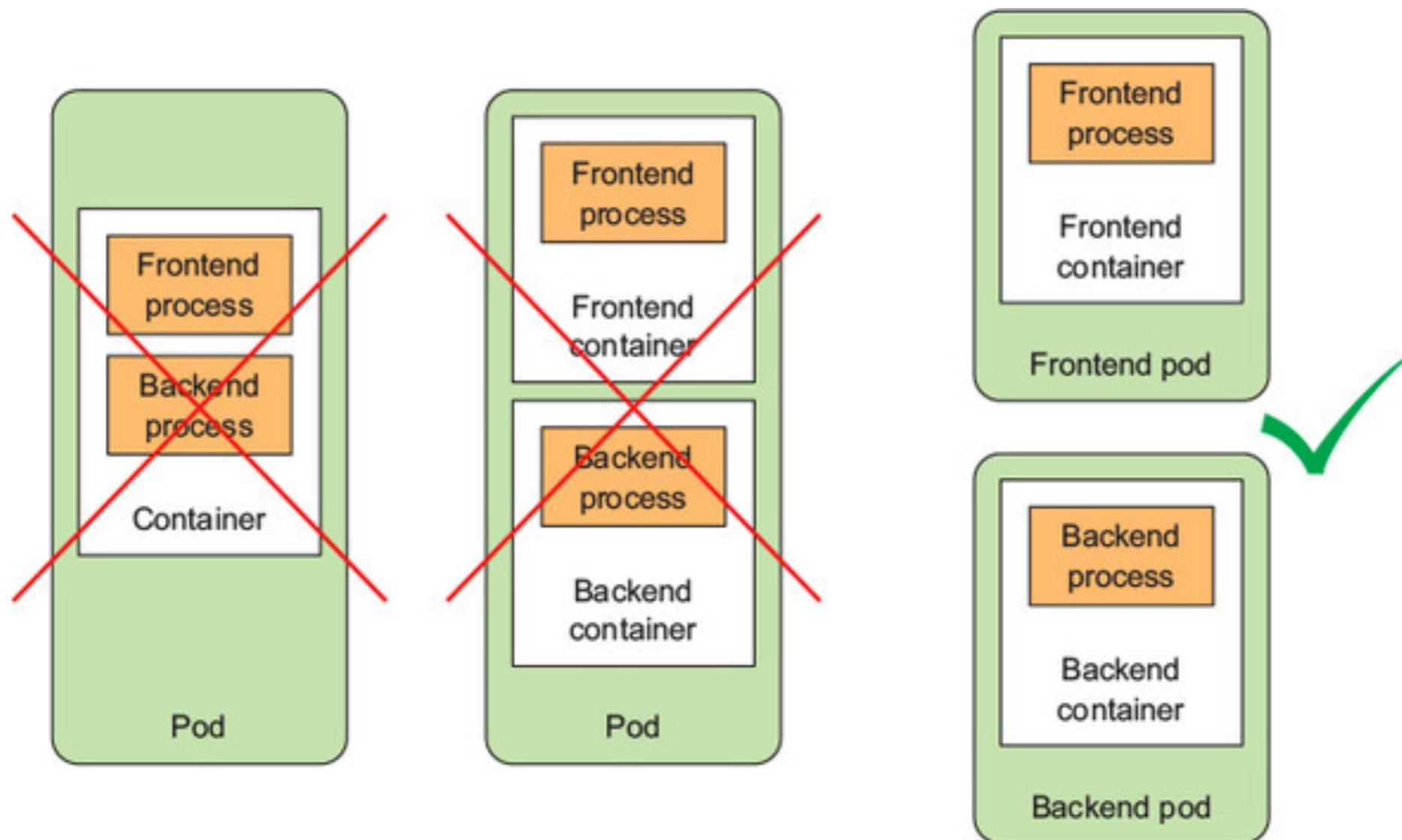
# Pods Life Cycle

Phase	Description
Pending	Accepted by kubernetes but container not created yet
Running	Pods bound to the node, all containers created and at least one container is running/start/restarting
Successed	Containers exited with status 0
Failed	All containers exit and at least one exited with non-zero status
Unknow	State of Pods can't be determined due to communication issues with its node



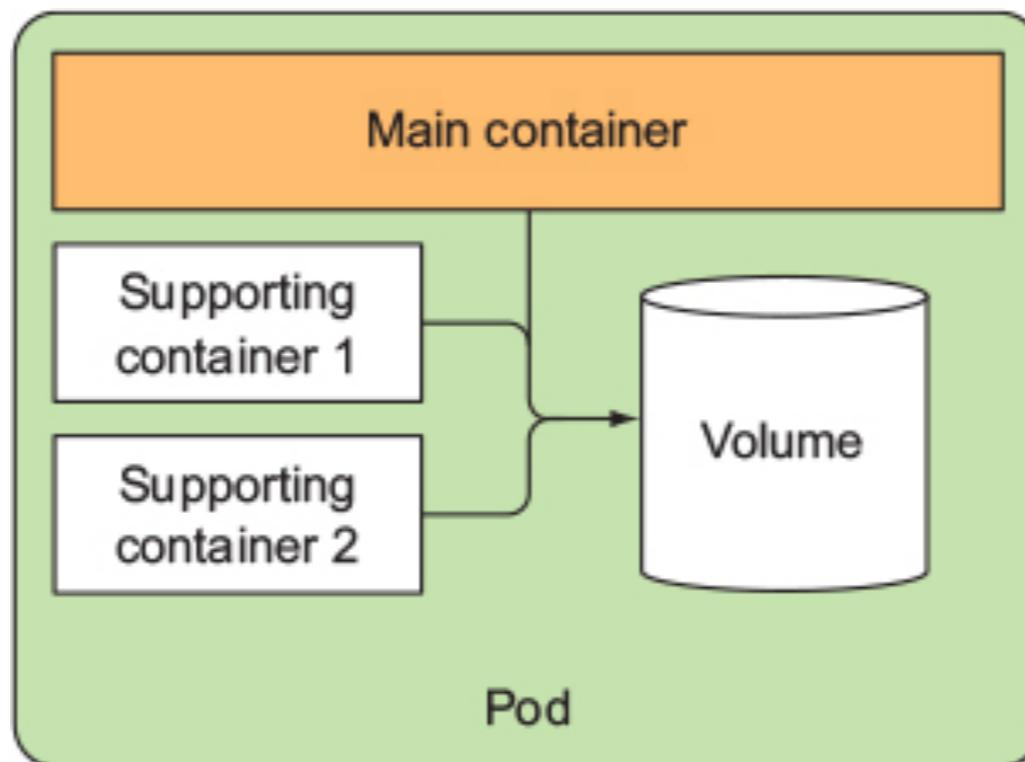
# Organize container across Pods

Split multi-tiers app into multiple pods

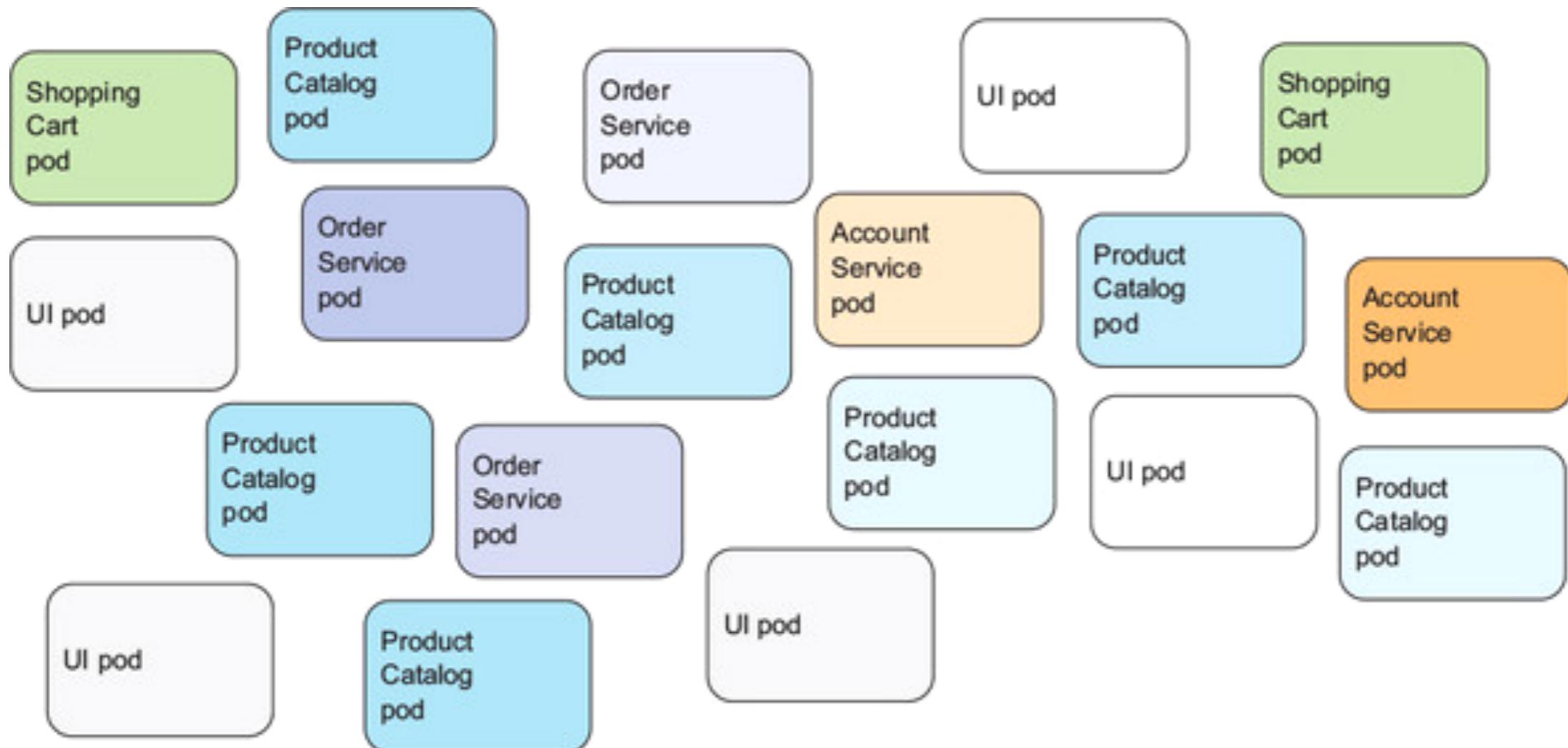


# Organize container across Pods

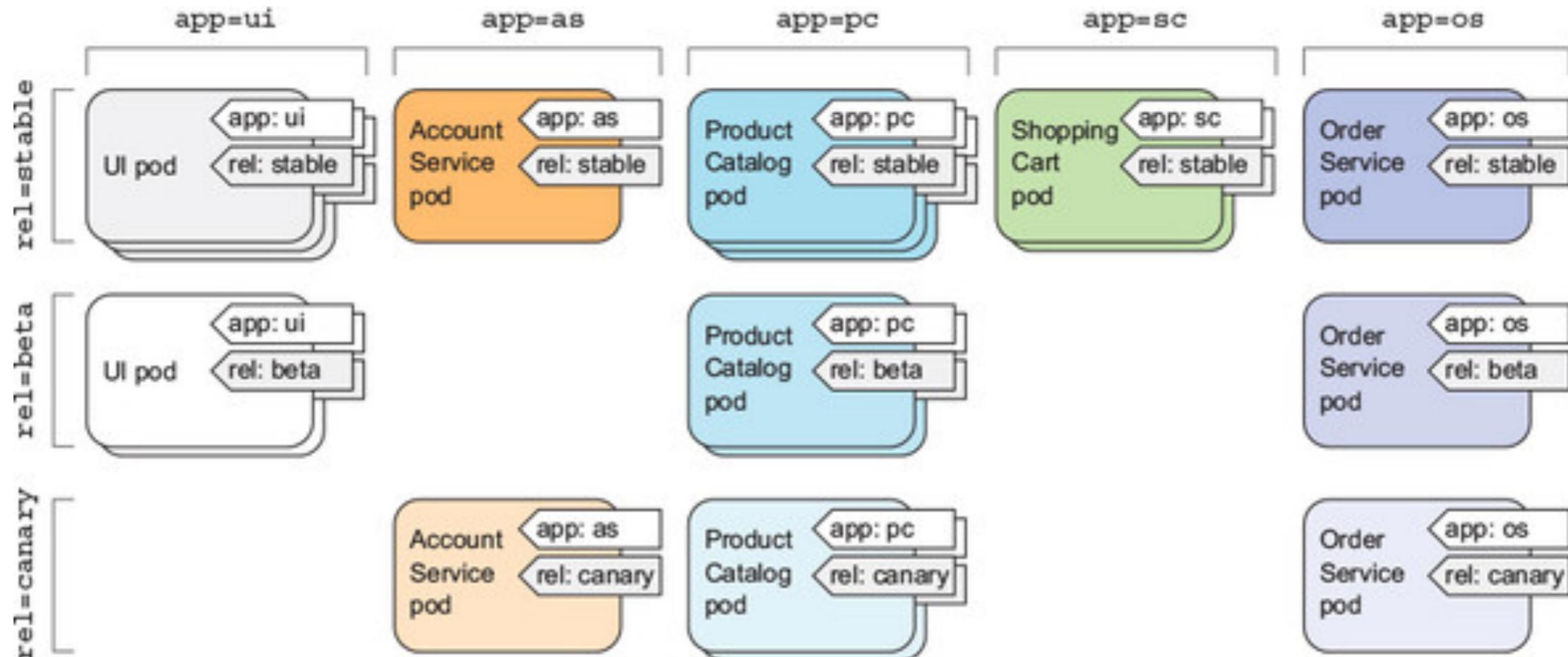
When to use multiple containers in a pods



# Organize pods with Labels

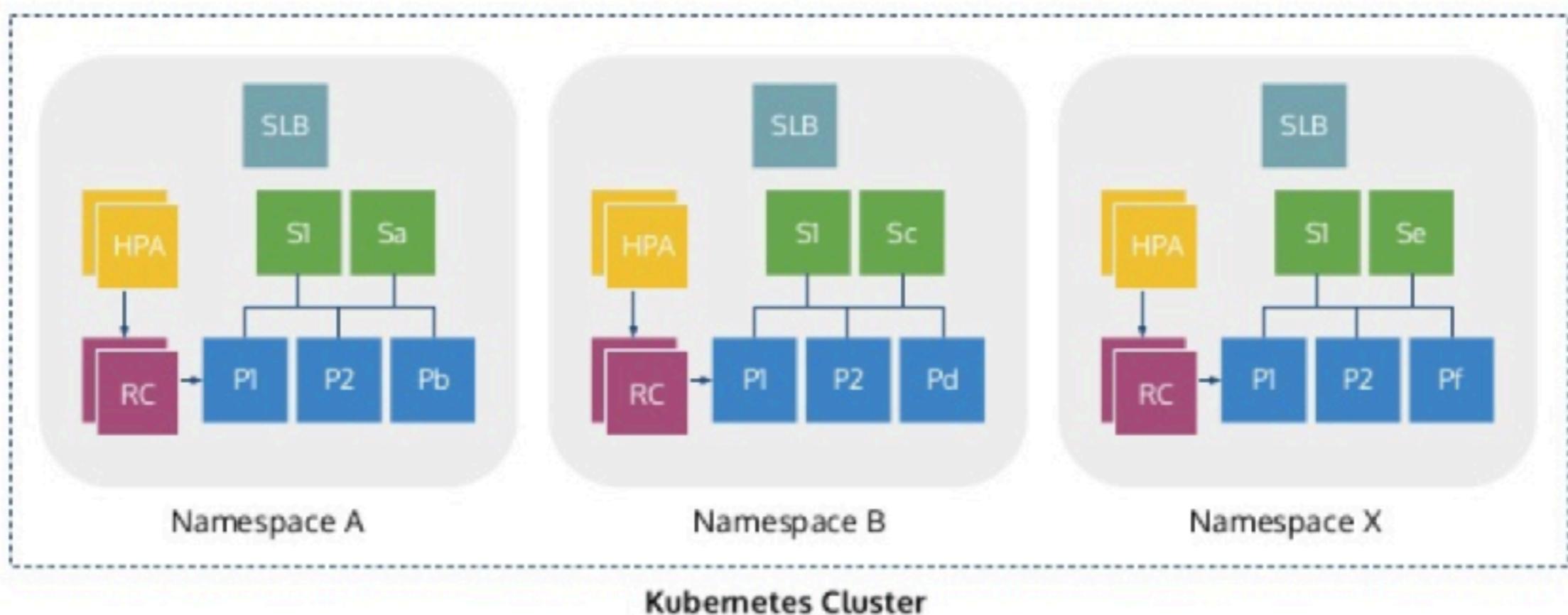


# Organize pods with Labels



# Pods Namespaces

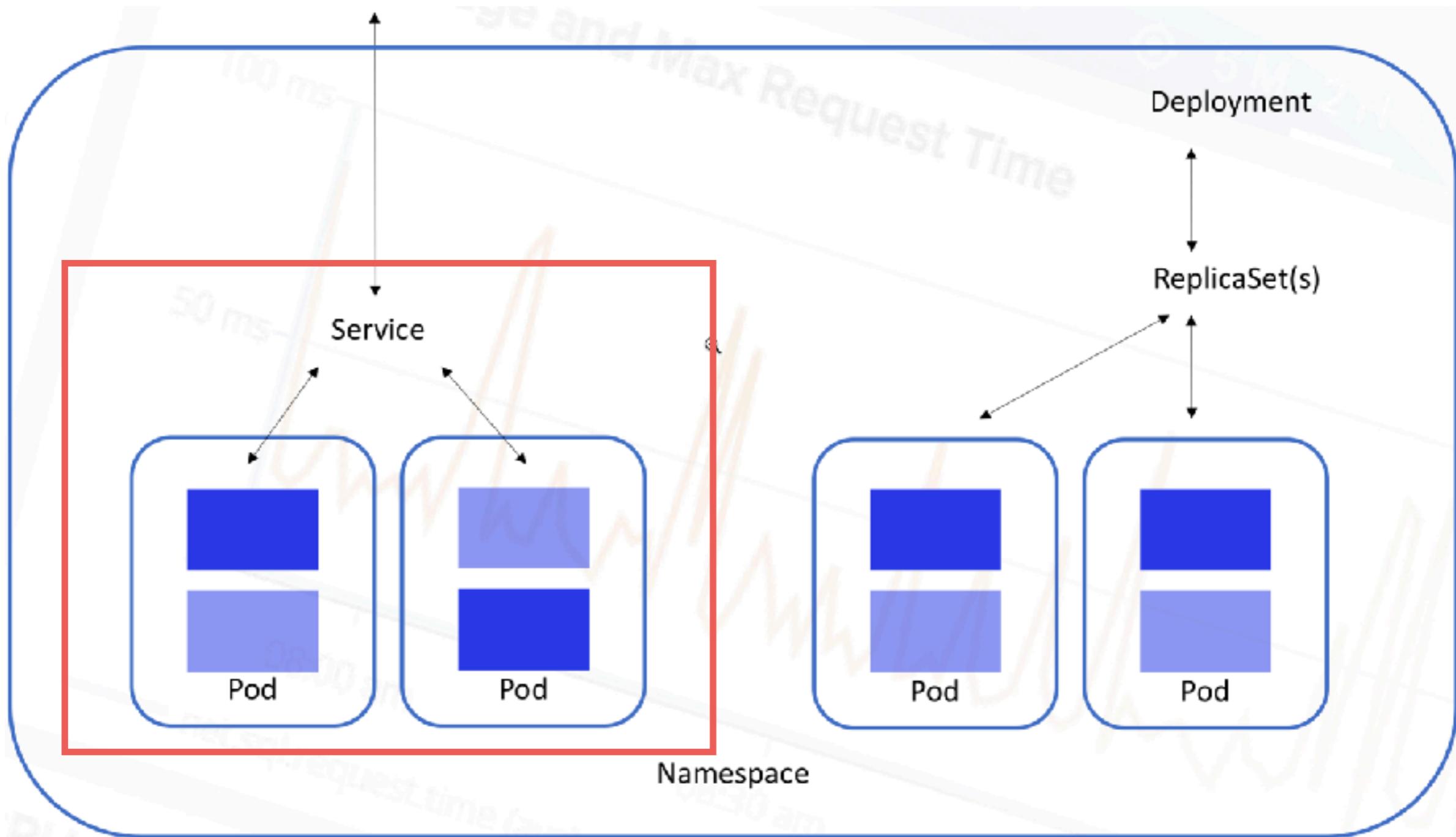
Allow different teams to use the same cluster



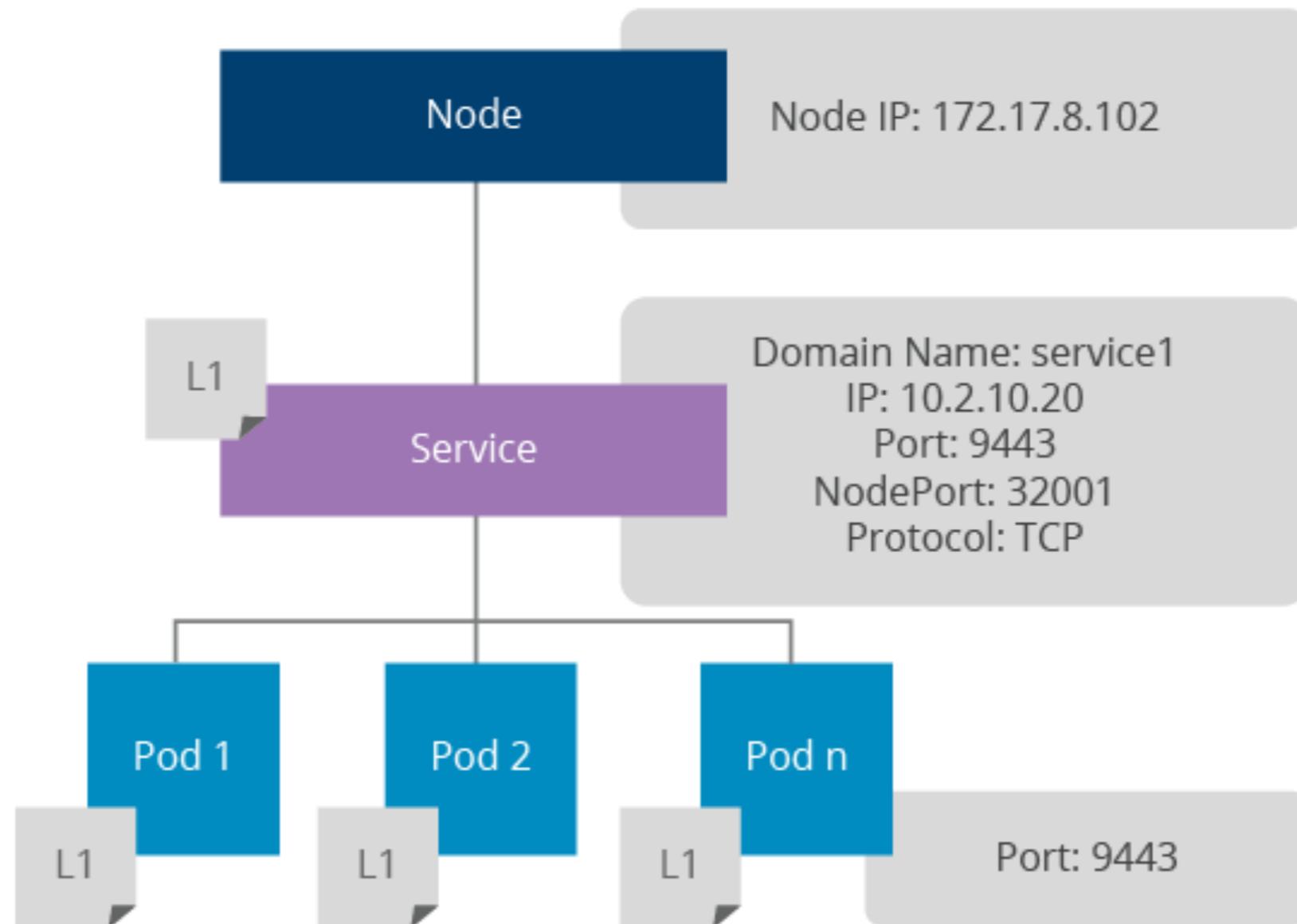
# **Services discovery and Load balancing**



# Services



# Services

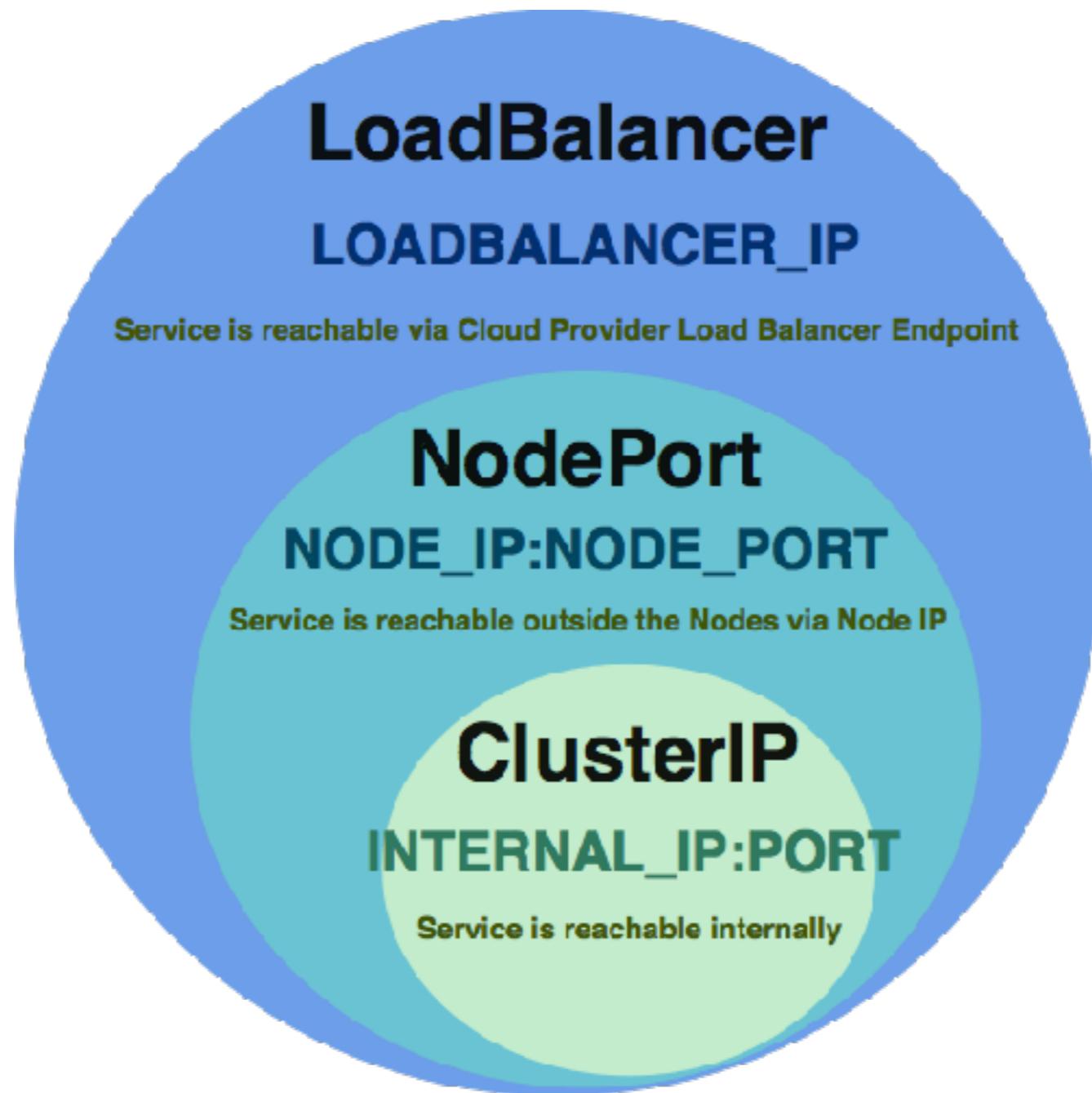


# Services

- Independent from Pods
- Abstraction layer of Pods
- Provide load balance
- Expose access Pods/Load balance
- Find Pods by label selector



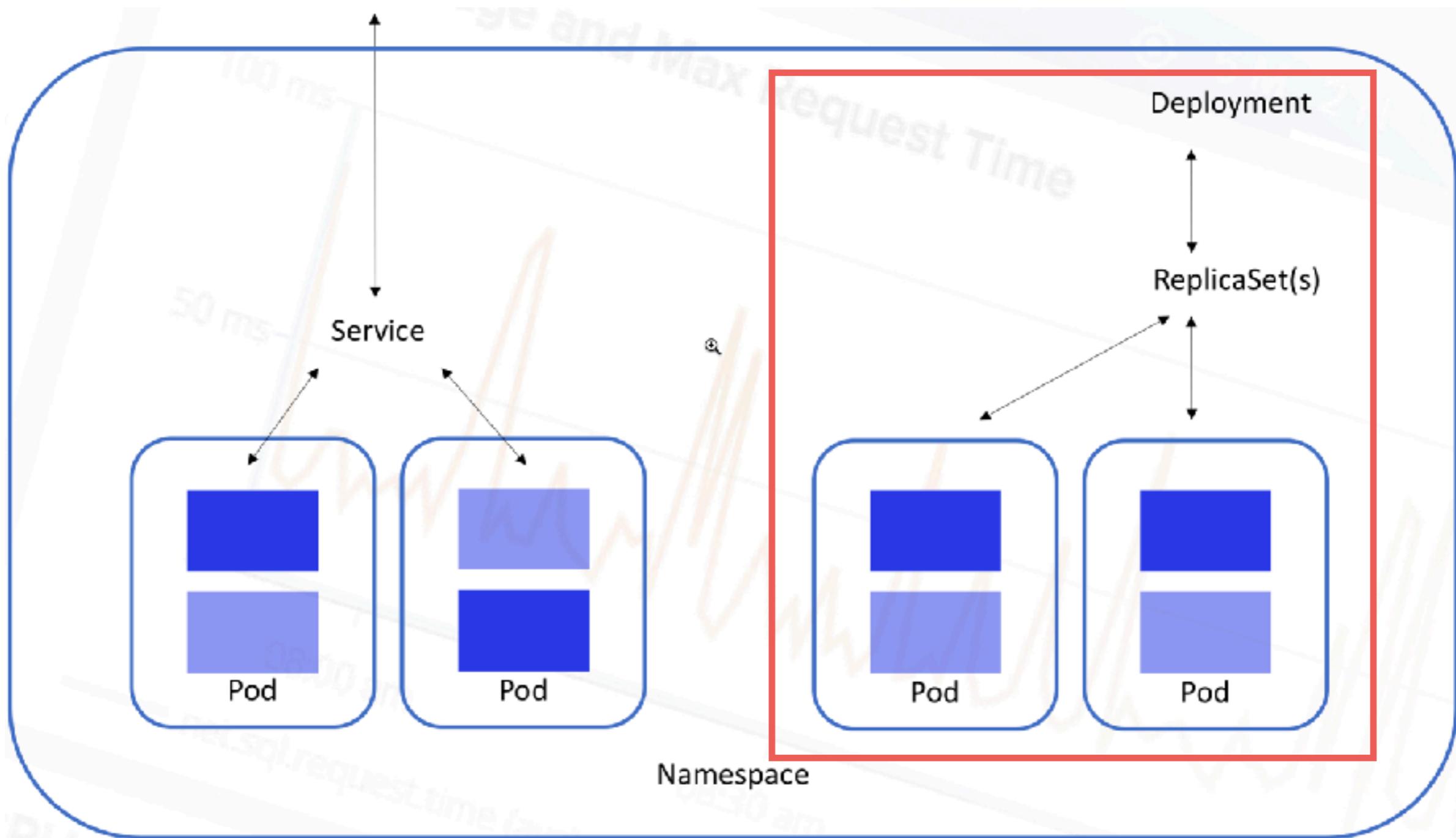
# 3 types of services



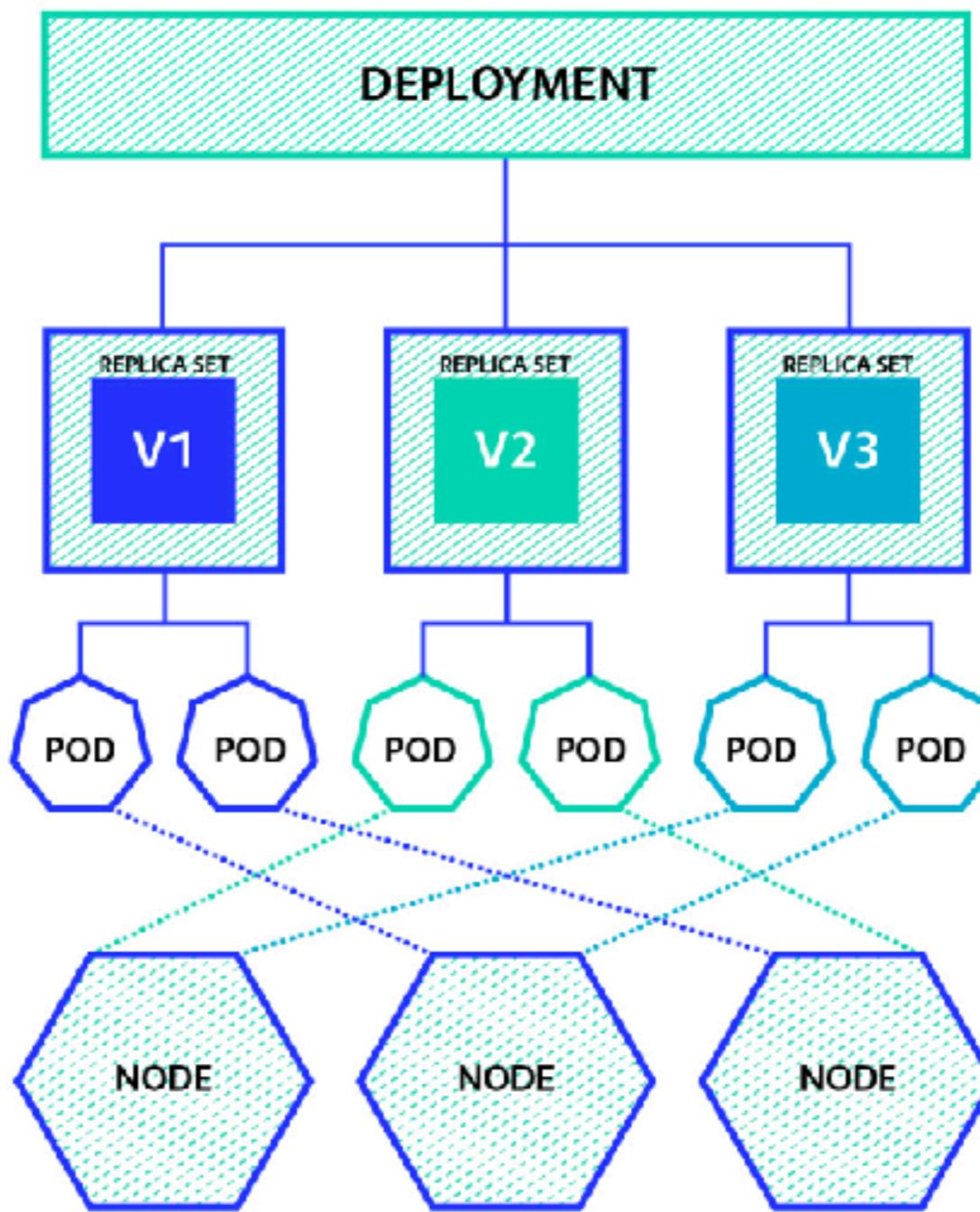
# **Deployment and ReplicaSet (RS)**



# Deployment and ReplicaSet



# Deployment and ReplicaSet



# Deployment and ReplicaSet

Next-generation of Replication Controller  
Provide function to maintain versioning of Pods

- Update new version (Rollout)
- Revert to old version (Rollback)
- Scale a deployment
- Pause/Resume process

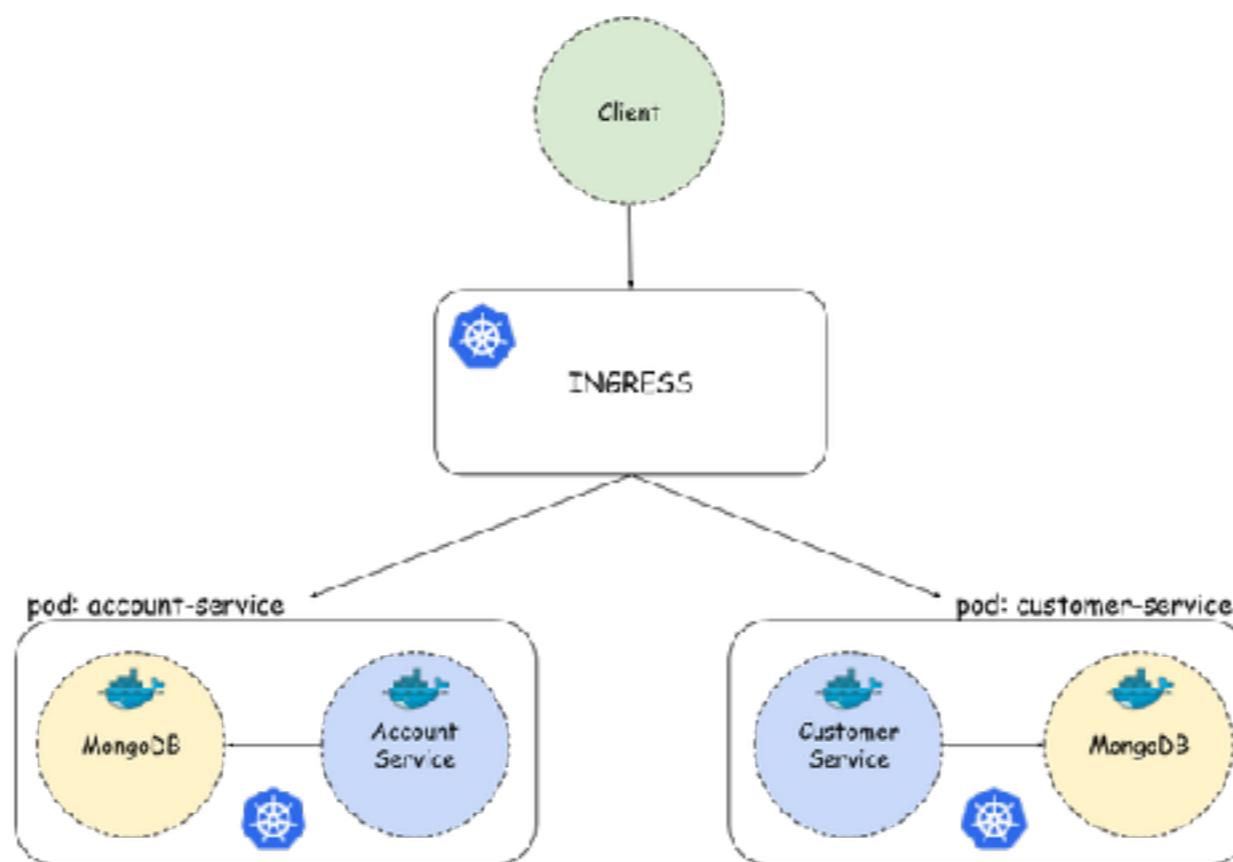


# Ingress network



# Ingress Network

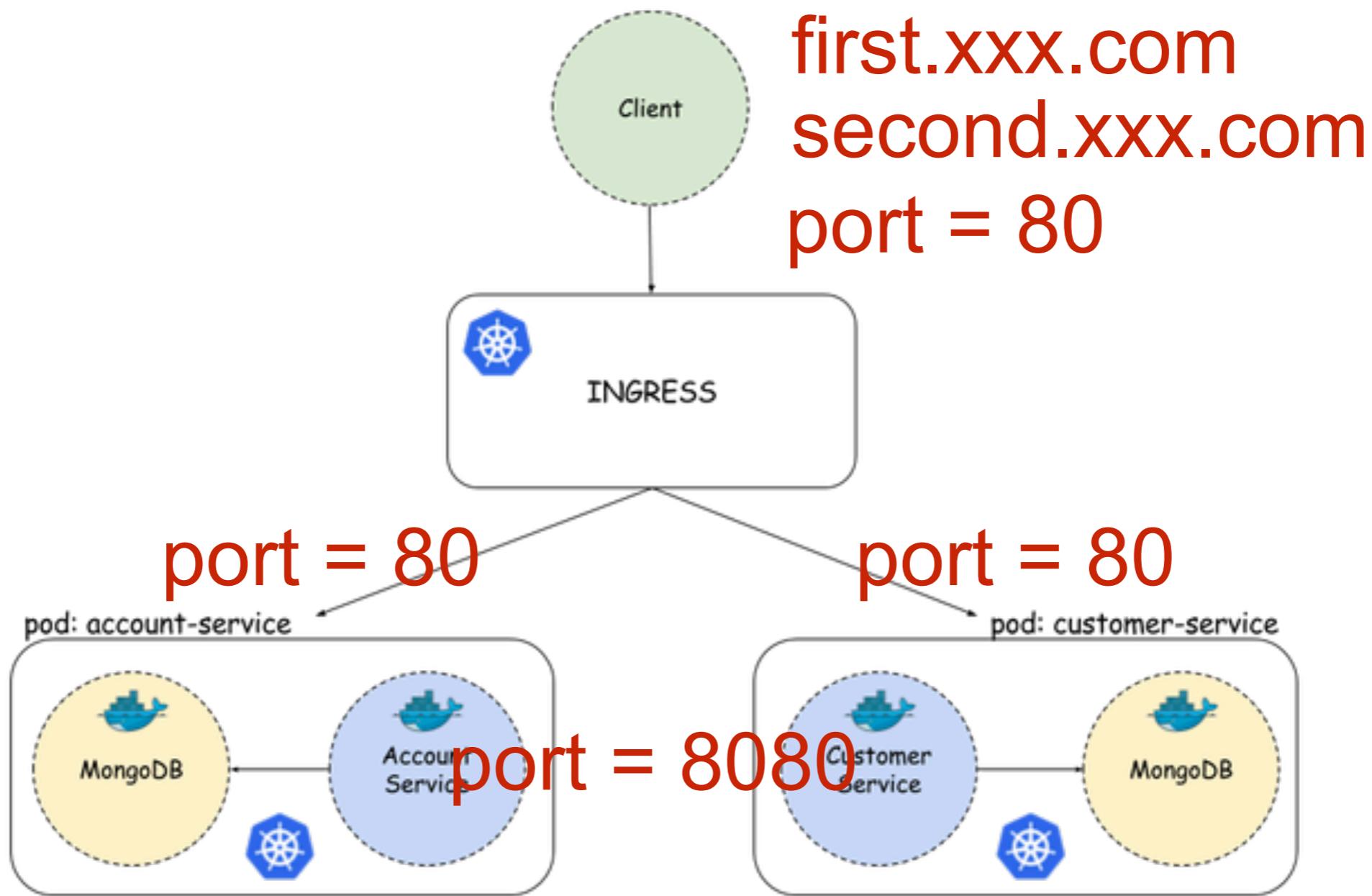
How to handle multiple services in same port ?  
How to limit protocol to access ?



<https://kubernetes.io/docs/concepts/services-networking/ingress/>



# Ingress Network



# StatefulSet



# StatefulSet

Bringing the concept of ReplicaSets to **stateful** Pods  
Enable running Pods in **cluster** mode  
Ideal for deploy **highly** available database **workload**



# StatefulSet

Stable, unique network identifiers

Stable, persistent storage

Ordered, graceful deployment and scaling

Ordered, graceful deletion and termination



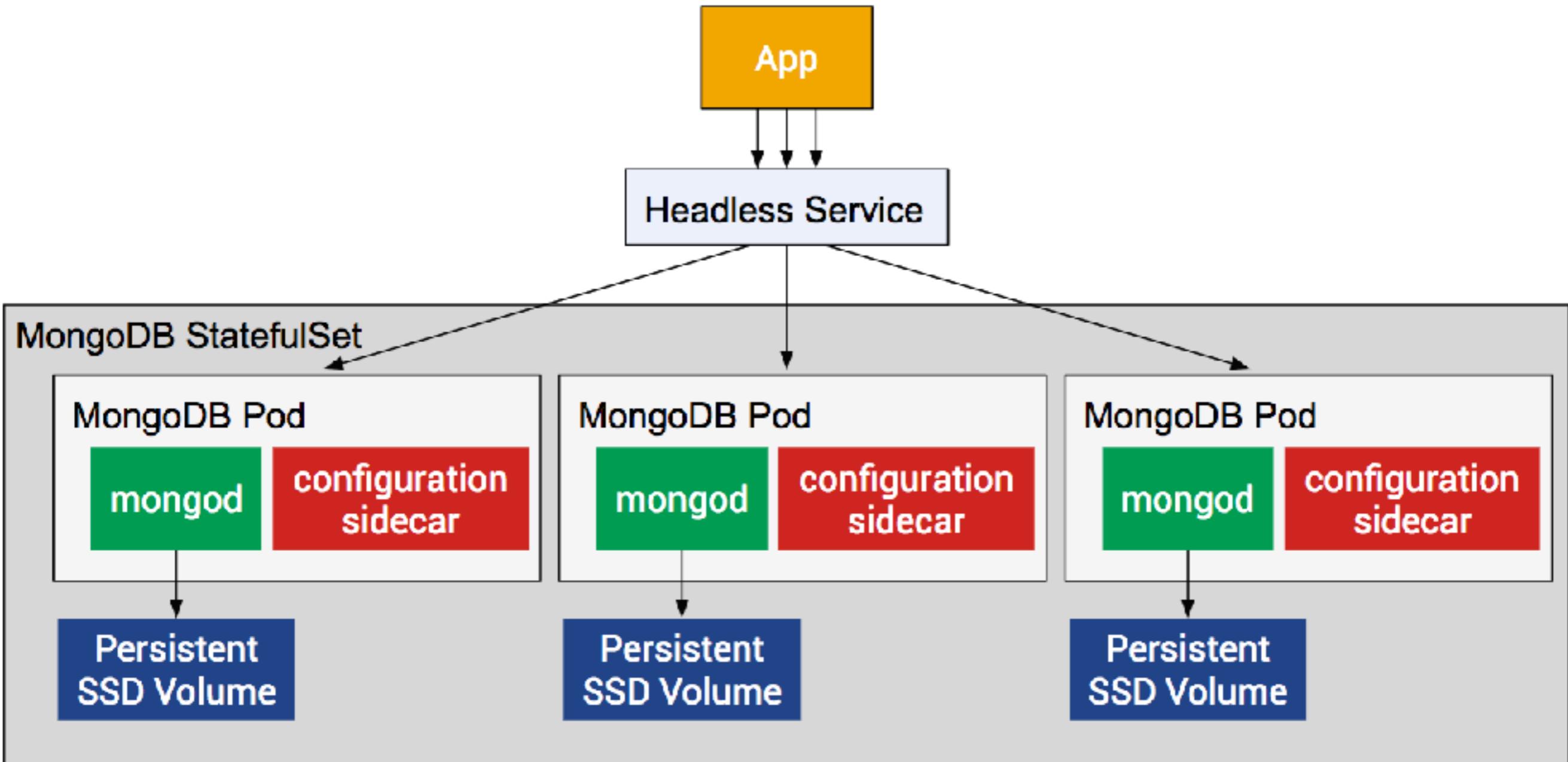
# Key concepts

Pods are created sequentially

Pods are terminated in LiFo (Last in, First out)



# StatefulSet



# Workshop



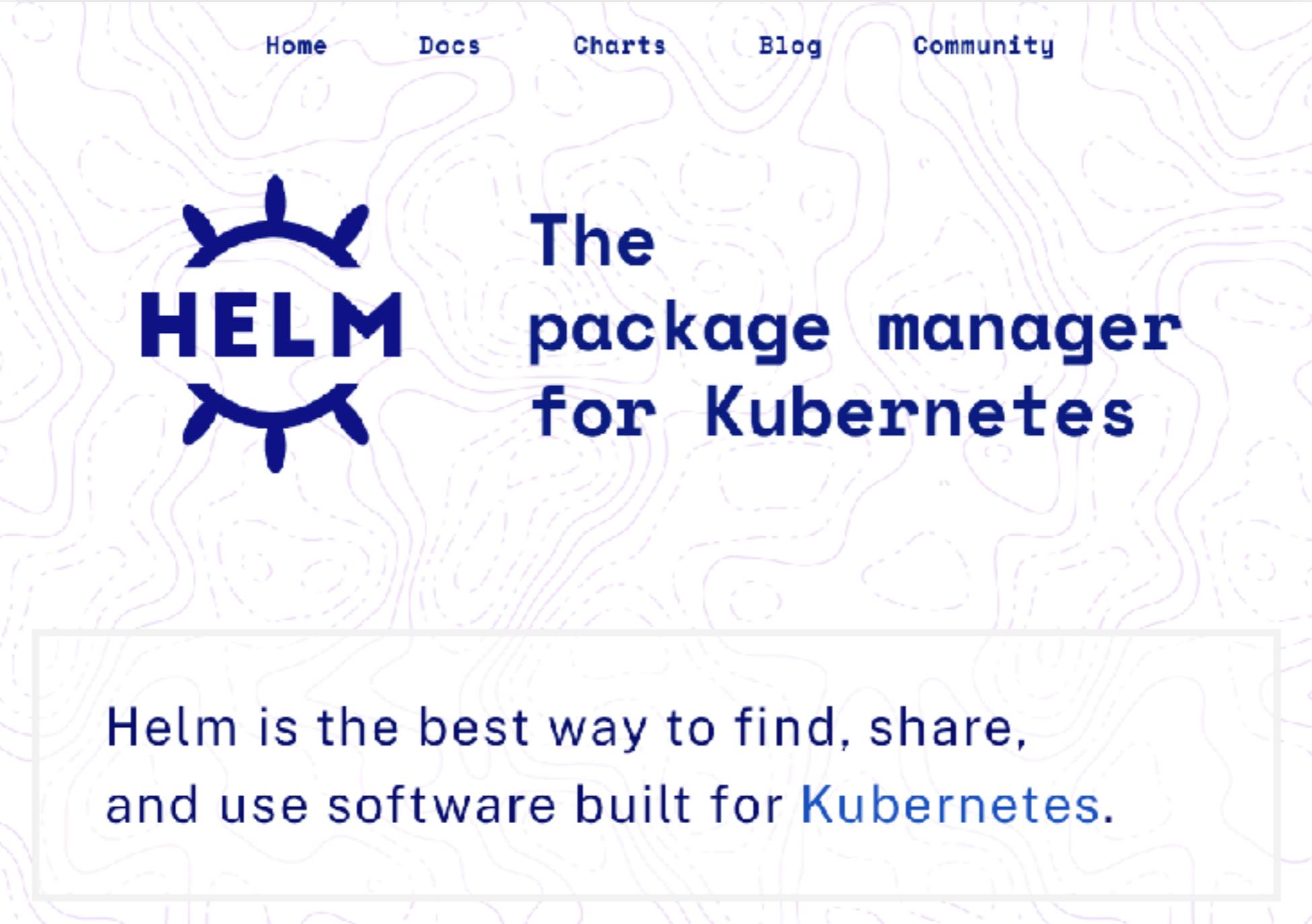
kubernetes



# Working with Helm



# Helm ?

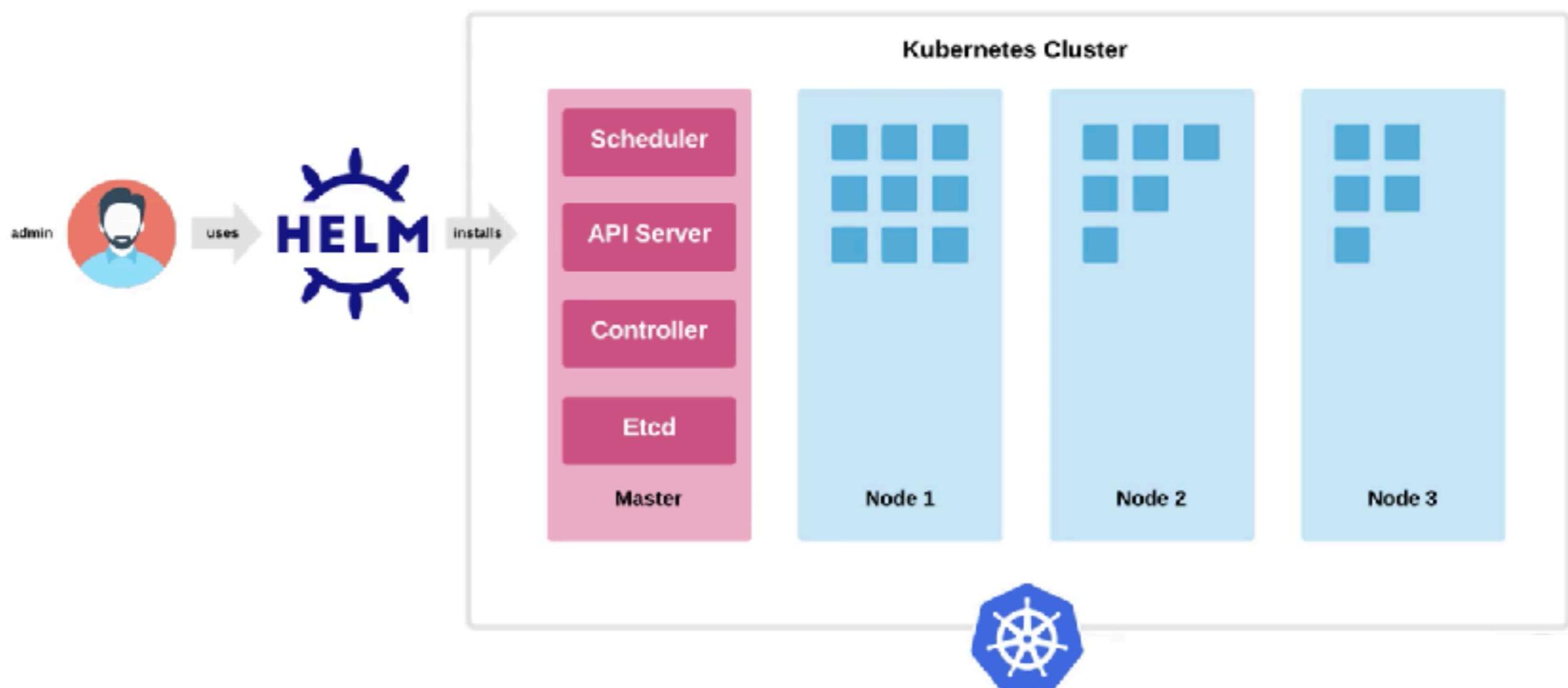


The image shows the official Helm website landing page. The background features a light purple and white abstract wavy pattern. At the top, there is a navigation bar with links: Home, Docs, Charts, Blog, and Community. Below the navigation, the Helm logo is displayed, consisting of a blue circular icon with three arrows pointing clockwise, surrounding the word "HELM" in a bold, blue, sans-serif font. To the right of the logo, the text "The package manager for Kubernetes" is written in a large, bold, blue font. A large rectangular callout box with a thin gray border is positioned in the lower-left area of the page. Inside this box, the text "Helm is the best way to find, share, and use software built for Kubernetes." is written in a large, bold, dark blue font.

<https://helm.sh/>



# Package manager for K8s

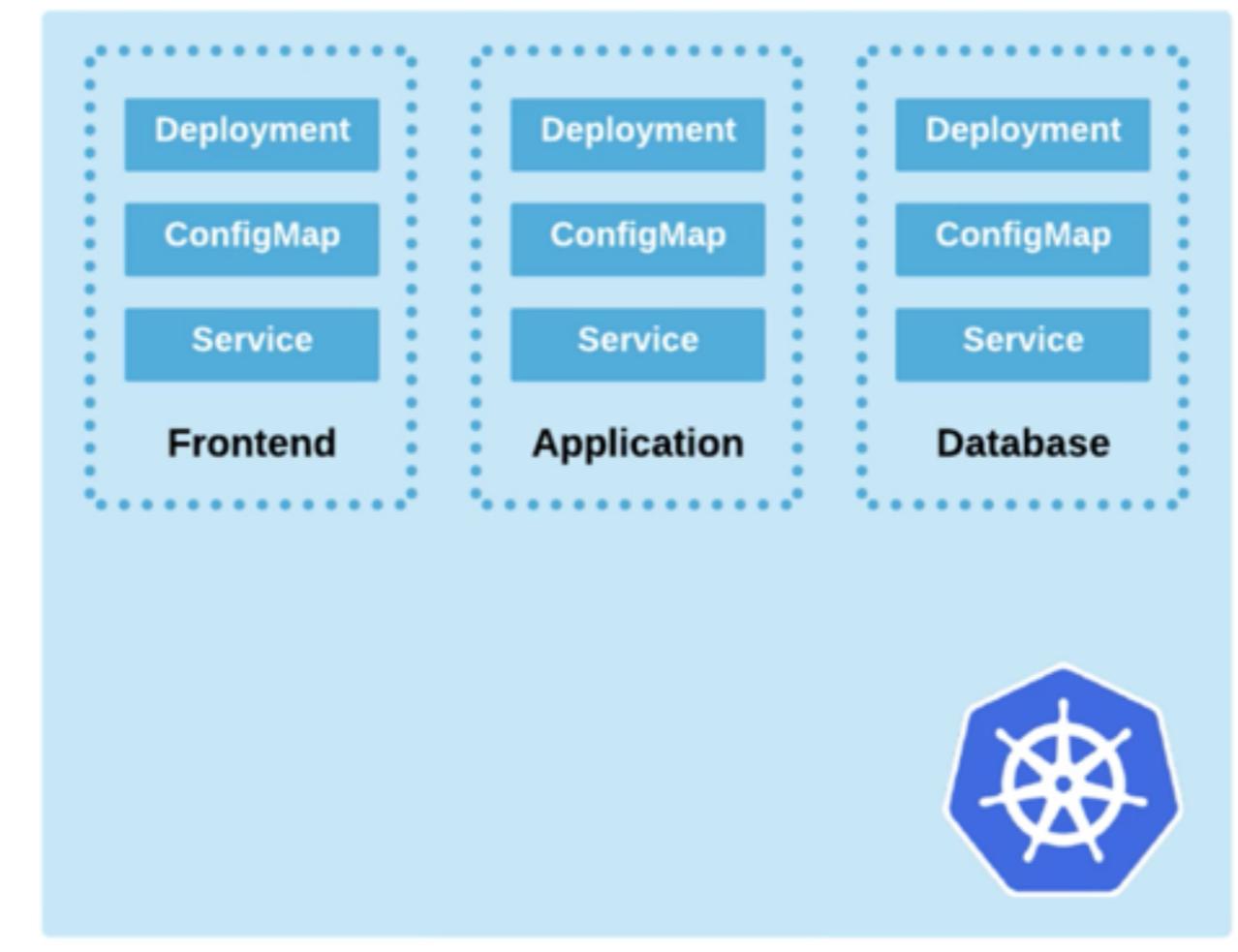


# Why Helm ?

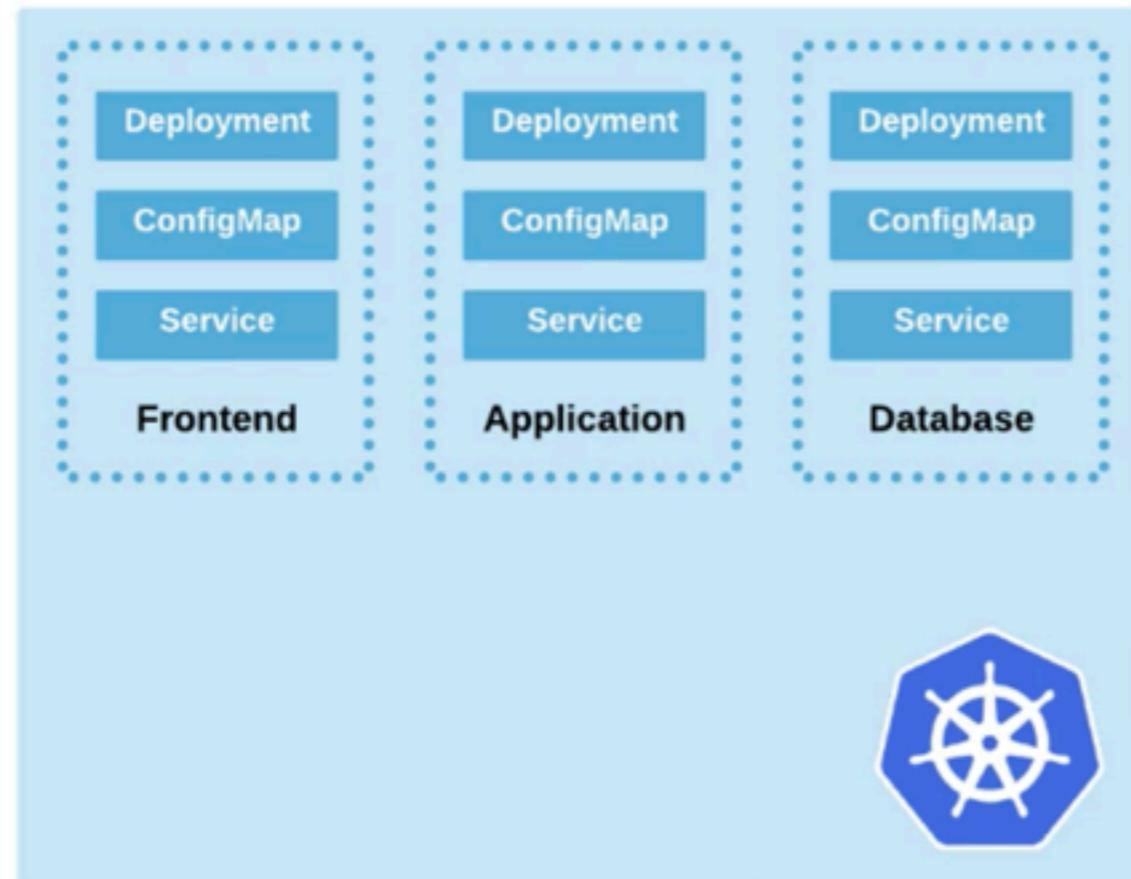
Hard to manage multiple K8s configurations  
Hard to deploy multiple K8s configurations  
Hard to share and reuse K8s configurations  
Hard to parameterise and support multiple environments



# K8s resources



# K8s resources



```
~/demo: kubectl apply -f
```



# Why Helm ?

Hard to manage app releases  
(rollout, rollback, history)

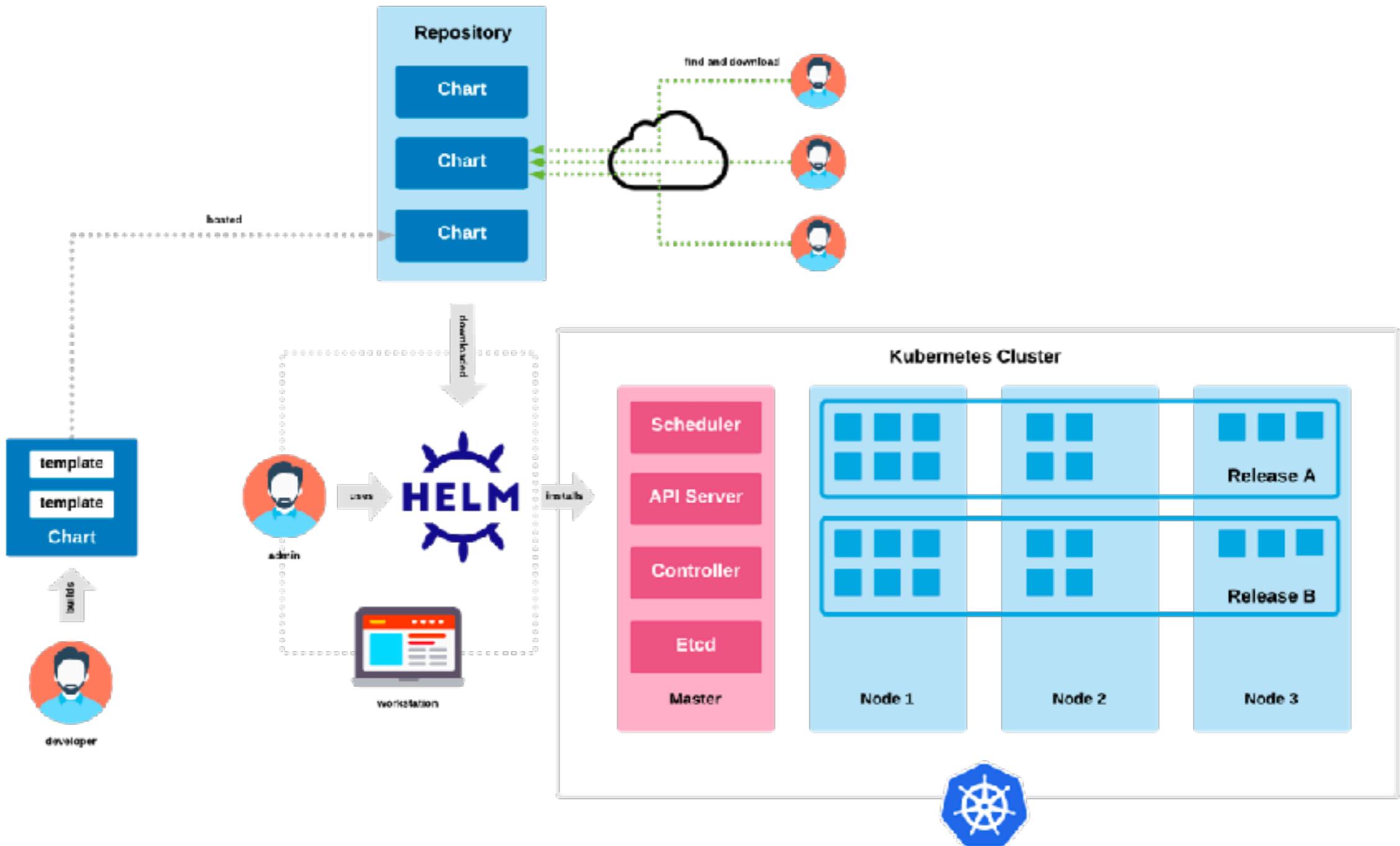
Hard to define deployment life cycle  
Hard to validate release state after deployment



# **Helm make it easy to start Using K8s with real application**



# Welcome to Helm



# What Helm ?

Helm is a **Package Manager** for Kubernetes

Package multiple K8s resources into a single logical deployment unit

Called “**Chart**”



# What Helm ?

Helm is a **Deployment management** for Kubernetes

Repeat deployment

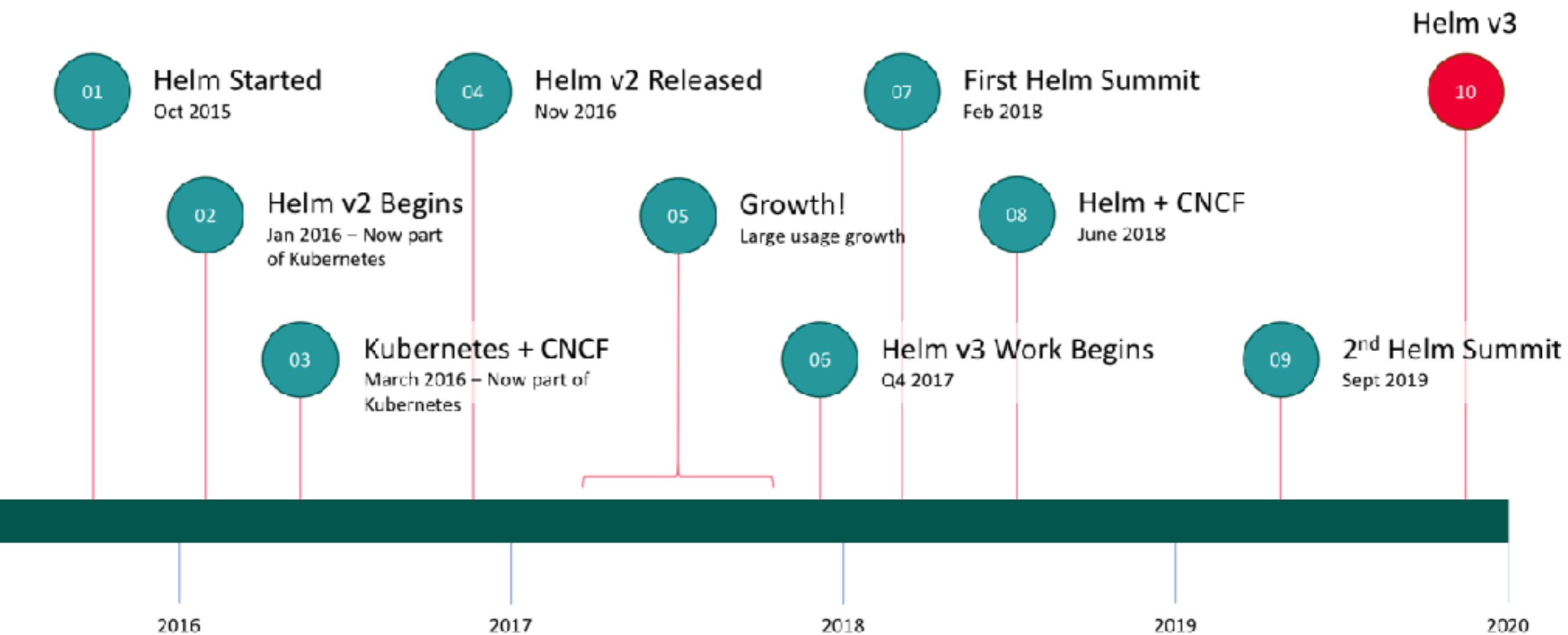
Management dependencies (reuse/share)

Manage multiple configurations

Update/rollback and test application deployments

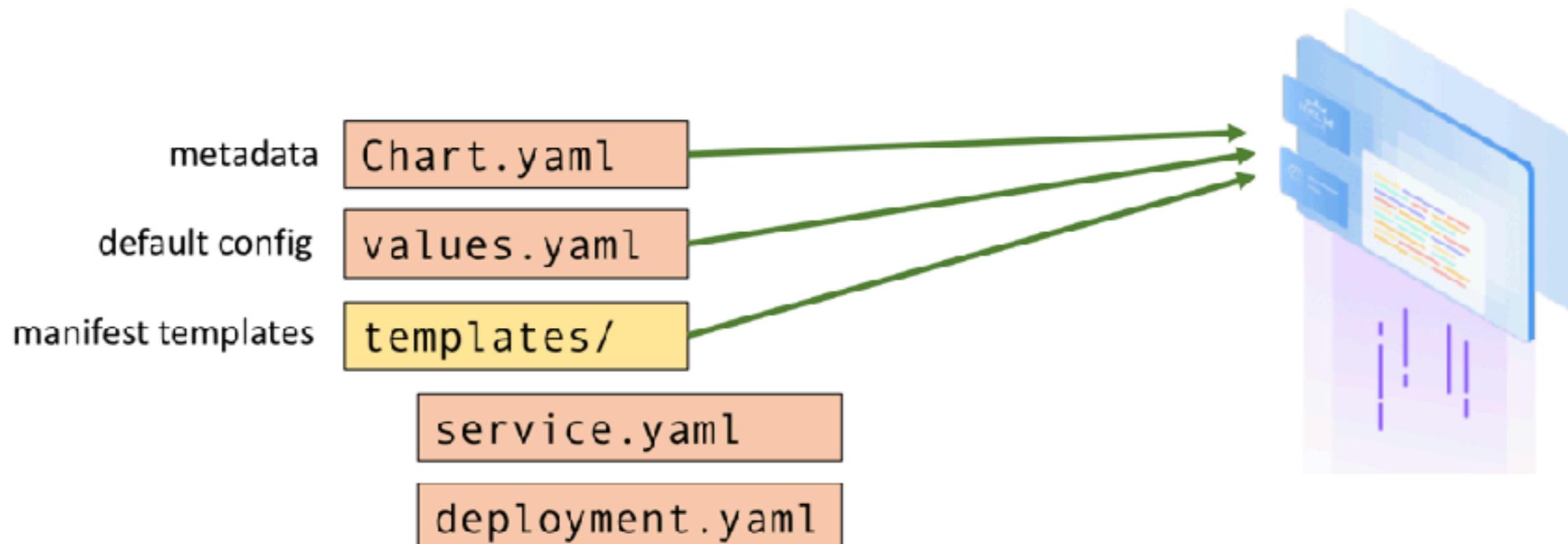


# Helm History



# Helm Charts

Helm packages are referred to as “Charts”  
Collections of files at well-known locations



# Helm Vocabulary

## Chart

A package; bundle of K8s resources

## Release

A chart instance is loaded into K8s

Same chart can be installed several times into the same cluster; each will have it's own Release



# Helm Vocabulary

## Repository

A repository of published Charts

## Template

K8s configuration file mixed with Go/Sprig template



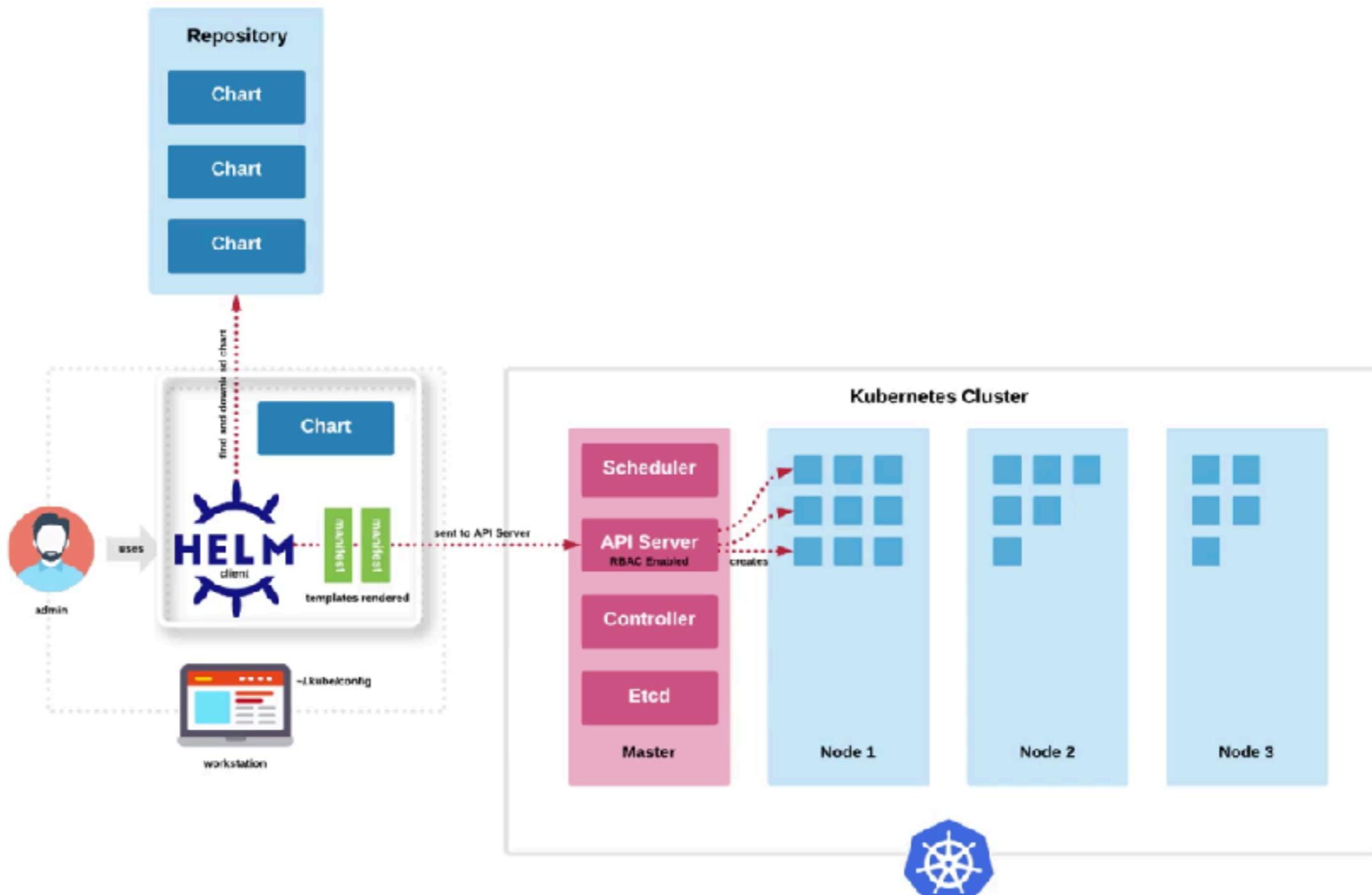
# Artifact Hub

The screenshot shows the homepage of Artifact Hub. At the top left is the logo "Artifact HUB" with a beta badge. Top right features "SIGN UP" and "SIGN IN" buttons, and a user profile icon. The main heading "Find, install and publish Kubernetes packages" is centered. Below it is a search bar with placeholder "Search packages" and a help icon. A tip below the search bar says "Tip: Use multiple words to refine your search. Example: kafka operator". Below the search area, there's a section for sample queries with links like "browse all packages", "Helm plugins", "OLM operators in the monitoring category", "Tekton Tasks", "QPA policies with MIT license", and "False rules for CVE". At the bottom, two large numbers are displayed: "2674 PACKAGES" and "35576 RELEASES".

<https://artifacthub.io/>



# Helm Architecture



# Helm's Core Values

Install resource in K8s should be easy

Teams should be able to collaborate

Release should be reproducible

Packages should be sharable



# Helm Tips

Create Chart for each (micro)service;  
keep it in same Git repository

Learn and practice Go Template language  
(and Sprig template library)

Use Helm hooks to control release flow



# Helm Tips

Use helm test to validate releases

Manage environments with multiple Values files

Do not commit secrets into GitHub

Follow community Helm best practices and conventions



# Helm commands



# Helm chart management

\$helm search hub [KEYWORD]

\$helm search repo [KEYWORD]

\$helm pull [CHART]

\$helm install [NAME] [CHART]

\$helm upgrade [RELEASE] [CHART]

\$helm rollback [RELEASE] [REVISION]

\$helm uninstall [RELEASE]



# Helm repository management

\$helm repo add [NAME] [URL]

\$helm repo list

\$helm repo remove [NAME]

\$helm repo update

\$helm repo index [DIR]



# Helm release management

\$helm status [RELEASE]

\$helm list

\$helm history [RELEASE]

\$helm get manifest [RELEASE]



# Helm chart management

\$helm create [NAME]

\$helm template [NAME] [CHART]

\$helm package [CHART]

\$helm lint [CHART]

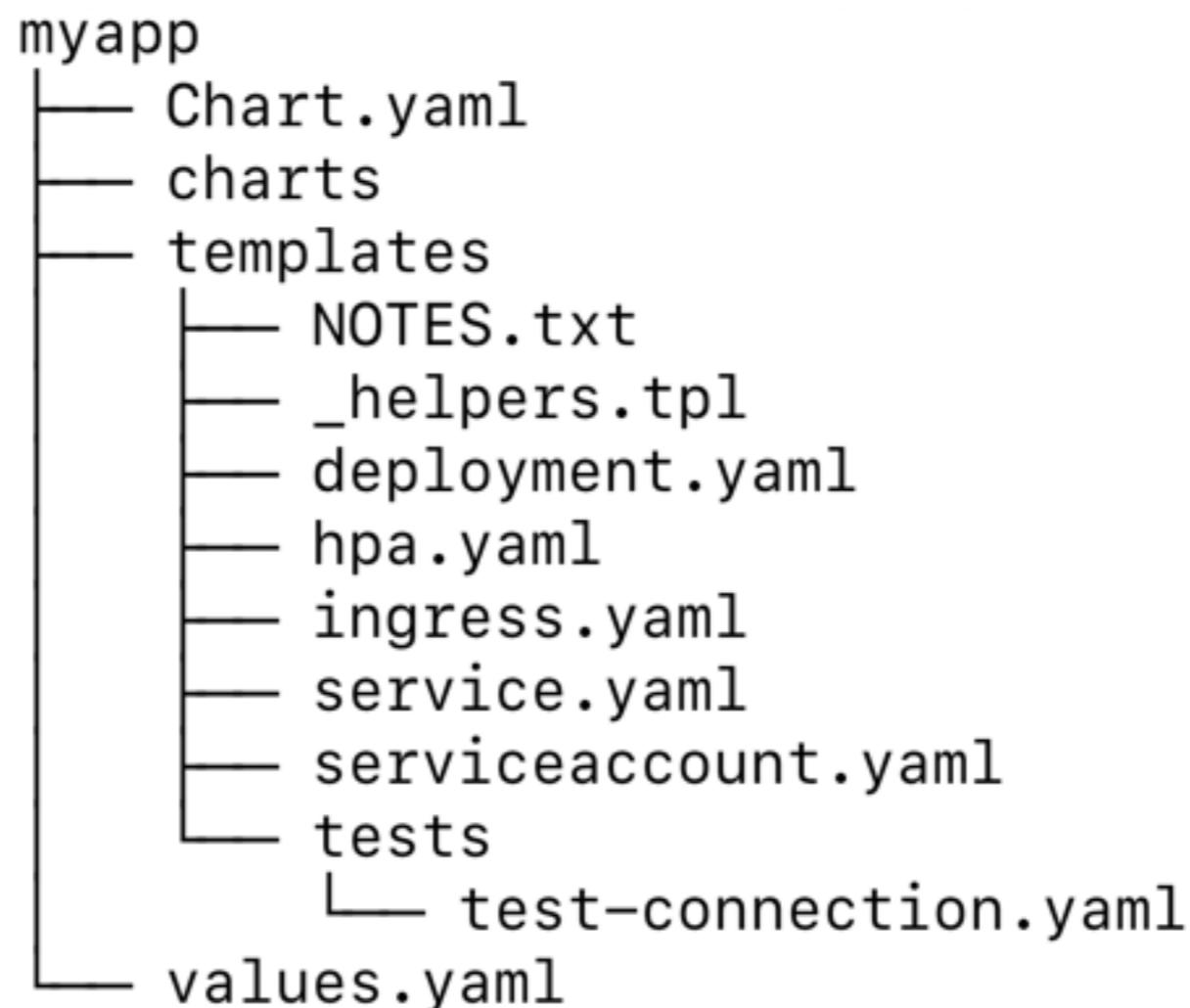


# Hello Helm



# Create a chart

```
$ helm create myapp
```



# Install a chart

`$helm install [NAME] [CHART]`

**From a chart directory**

`$helm install demo ./myapp`

**From a remote chart directory**

`$helm install demo myrepo/myapp`



# Custom values

You can pass the values files or key-value pairs from command line

## Use a values files

```
$helm install demo ./myapp -f custom.yaml
```

## Use a key-value pairs

```
$helm install demo ./myapp --set key=value
```



# Check status

\$helm status demo



# Upgrade a release

Create a new revision of your release  
Update template sources or config values

```
$helm upgrade demo ./myapp --set image.tag=1.1.1
```



# Rollback a release

Helm tracks every revision made on release  
You can revert back to a working version

**\$helm rollback demo <revision>**

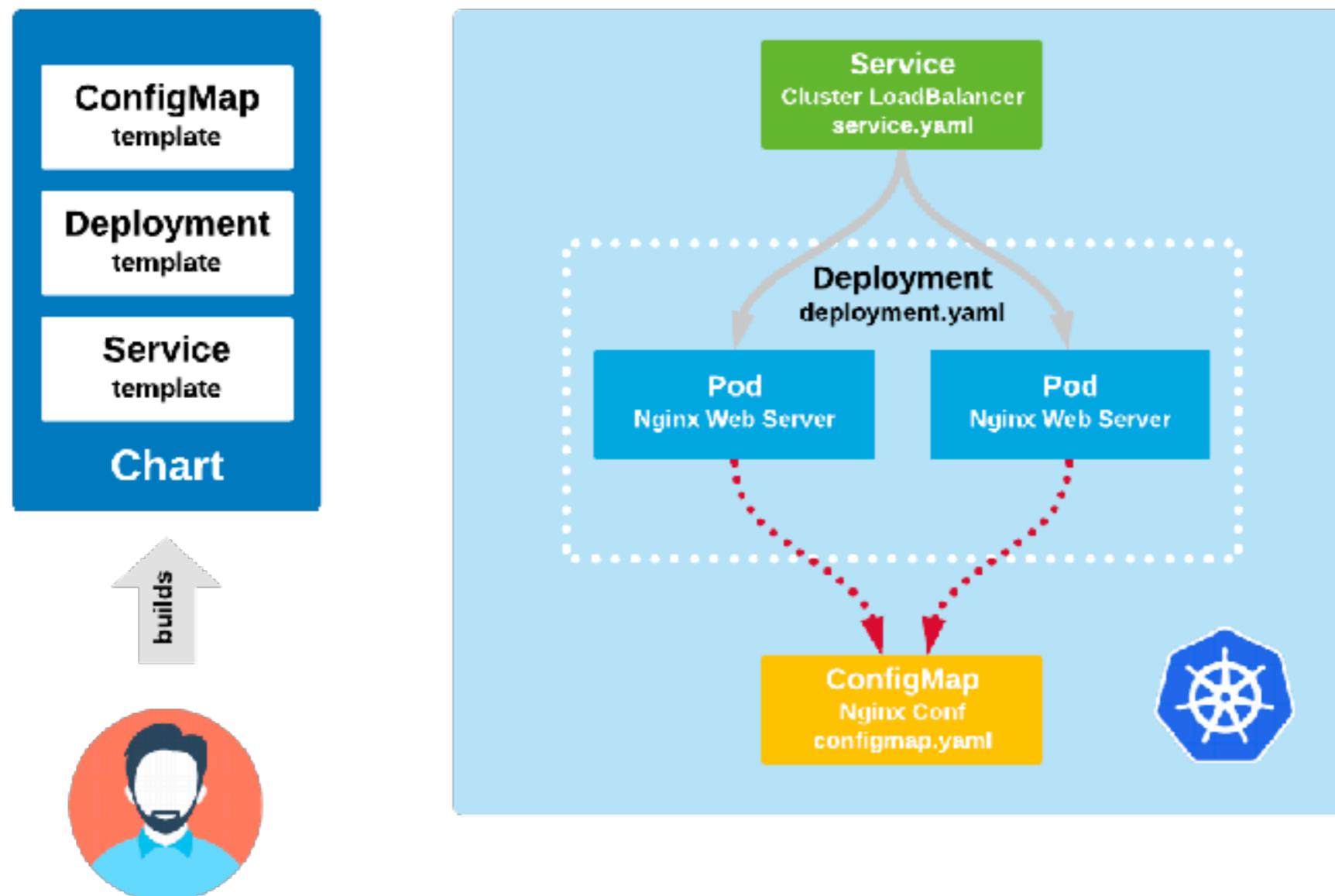


# Remove a release

```
$helm delete demo
```



# Helm Demo



# Workshop

