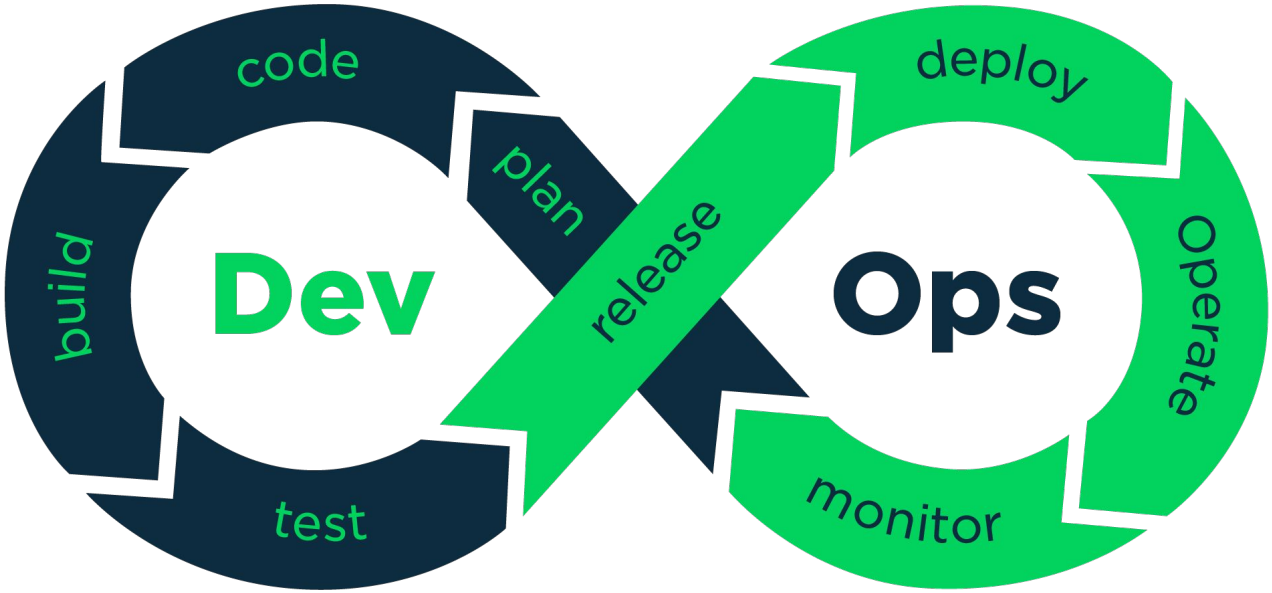
# DevOps Essential Training v6.1




Trainer: Man Guo Chang

TINFOTECH

Website:www.tertiarycourses.com.sg
Email: enquiry@tertiaryinfotech.com

# About the Trainer

Mr. Man Guo Chang graduated from Nanyang Technological University, School of Electrical and Electronic Engineering, major in Computer Engineering.

Mr. Man has more than 25 years of working experience in the Semiconductor field, specialized in IC Testing, Product Engineering, Data Analysis, and Software Development.

Mr. Man is an ACTA certified trainer. His skill set includes Website Development, Software Development, Machine Vision,  Internet of Things, ROS, Cyber Security, etc.

# Let's Know Each Other...

Say a bit about yourself

- Name
- What Industry you are from?
- Do you have any prior knowledge in Linux and DevOps?
- Why do you want to learn DevOps?
- What do you want to learn from this course?

# Ground Rules

- Set your mobile phone to silent mode
- Actively participate in the class. No question is stupid.
- Respect each other view
- Exit the class silently if you need to step out for phone call, toilet break

# Ground Rules for Virtual Training

- Upon entering, mute your mic and turn on the video. Use a headset if you can
- Use the 'raise hand' function to indicate when you want to speak
- Participant actively. Feel free to ask questions on the chat whenever.
- Facilitators can use breakout rooms for private sessions.

# Guidelines for Facilitators

1. Once all the participants are in and introduce themselves
2. Goto gallery mode, take a snapshot of the class photo - makes sure capture the date and time
3. Start the video recording (only for WSQ courses)
4. Continue the class
5. Before the class end on that day, take another snapshot of the class photo - makes sure capture the date and time
6. For NRIC verification, facilitator to create breakout room for individual participant to check (only for WSQ courses)
7. Before the assessment start, take another snapshot of the class photo - makes sure capture the date and time (only for WSQ courses)
8. For Oral Questioning assessment, facilitator to create breakout room for individual participant to OQ (only for WSQ courses)
9. End the video recording and upload to cloud (only for WSQ courses)
10. Assessor to send all the assessment records, assessment plan and photo and video to the staff (only for WSQ courses).

# Prerequisite

This is a beginner course. No prior programming prerequisite is assumed.

# Google Classroom

- The resources can be found on the Google classroom
- Goto https://classroom.google.com and enter the class code below
- If you have problem to access the Google Classroom, please inform trainer or the staff

z6zgzzf

# Agenda

## Topic 1  Overview of DevOps

- Enterprise Software Delivery
- What is DevOps for Software Delivery
- DevOps Success Stories
- Build a Effective DevOps Team/Culture
- Build a Continuous Integration/Continuous Delivery System

## Topic 2 Continuous Integration/Delivery

- What is Integration
- Why Integration
- Benefits CI/CD
- DevOps Lifecycle
- What is CI/CD
- DevOps CI/CD Tools
- Activity: Understanding of Devops
- Life cycle with CI/CD Pipeline

# Agenda

## Topic 3 MicroServices

- What is Microservices
- Key Benefits of Microservices
- Companies adopting Microservices
- Microservice Components
- Container LifeCycle

## Topic 4 Infrastructure as Code

- Infrastructure as Code
- IaC Configuration Management Tools
- Container Based Infrastructure
- Examples of IaC
- Building IaC using Docker File
- Dockerfile Example
- Compose Multi-Container Application

# Agenda

## Topic 5 Monitoring and Logging

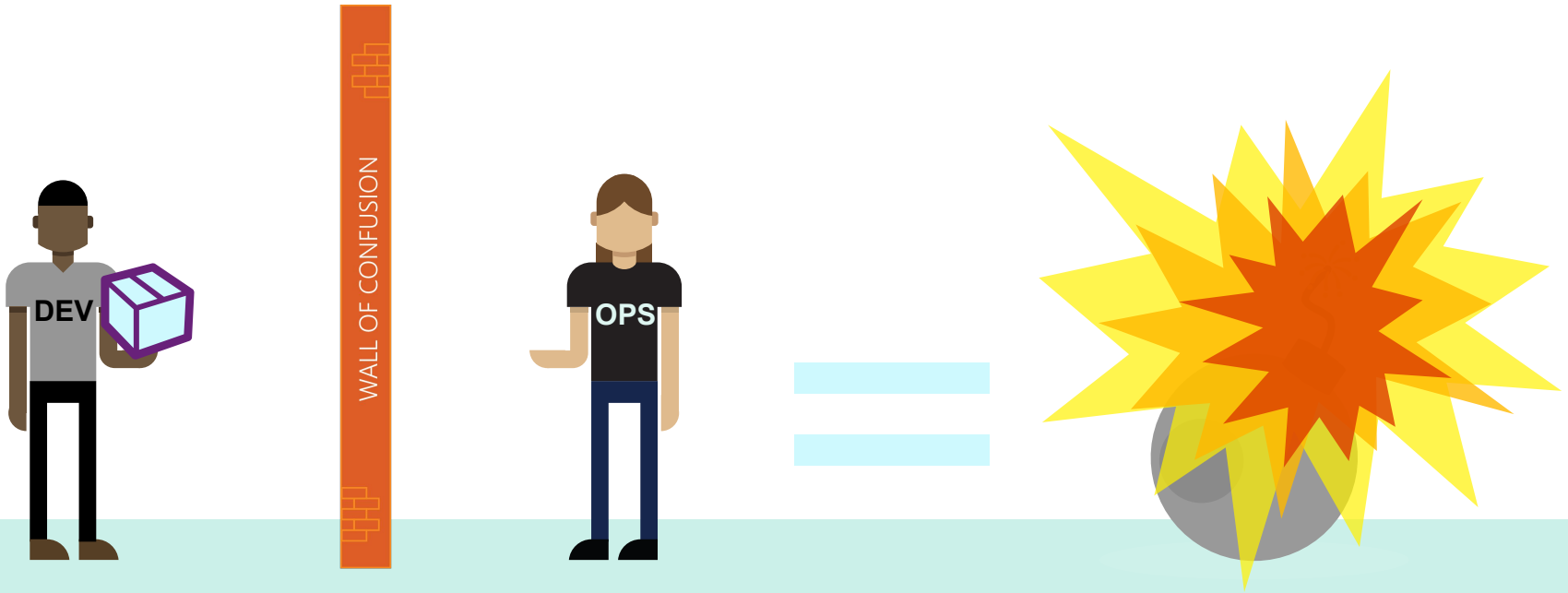
- What is Monitoring
- What is Log Management
- The 3 Pillars of Observability
- External Monitoring
- Metrics and Distributed Tracing
- Events and Logs
- Best Practices
- Logging and monitoring Tools

## Topic 6: Communication and Collaboration

- Devops Collaboration and Communication
- Benefit of Communication and Collaboration Tools in Devops
- Communication and Collaboration Tools
- DevOps Collaboration and Communication
- Activity: GIT and Slack Integration

# Topic 1
# Overview of DevOps

# Traditional Development and Operations



# Dev & Ops Typical conversation

Put the current release live, NOW!

It works on my machine

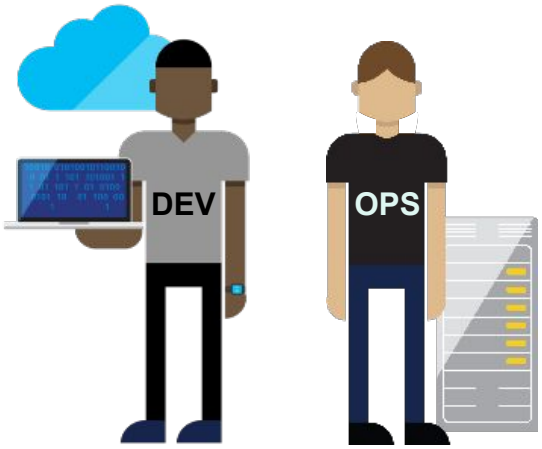
We need this Yesterday

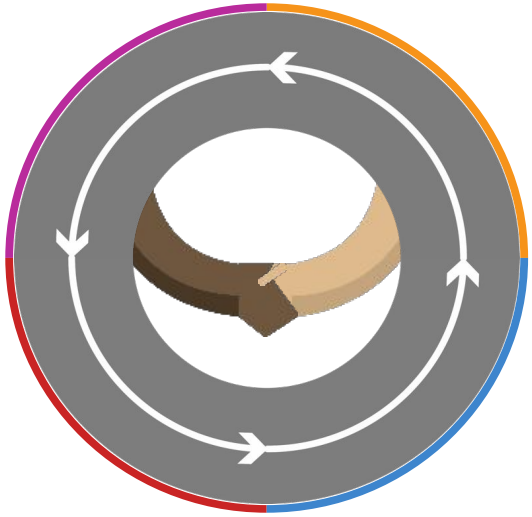
You are using the wrong version


What are the dependencies?
No machines available...
Which DB?
High Availability?
Scalability?

# Balance between Dev & Ops

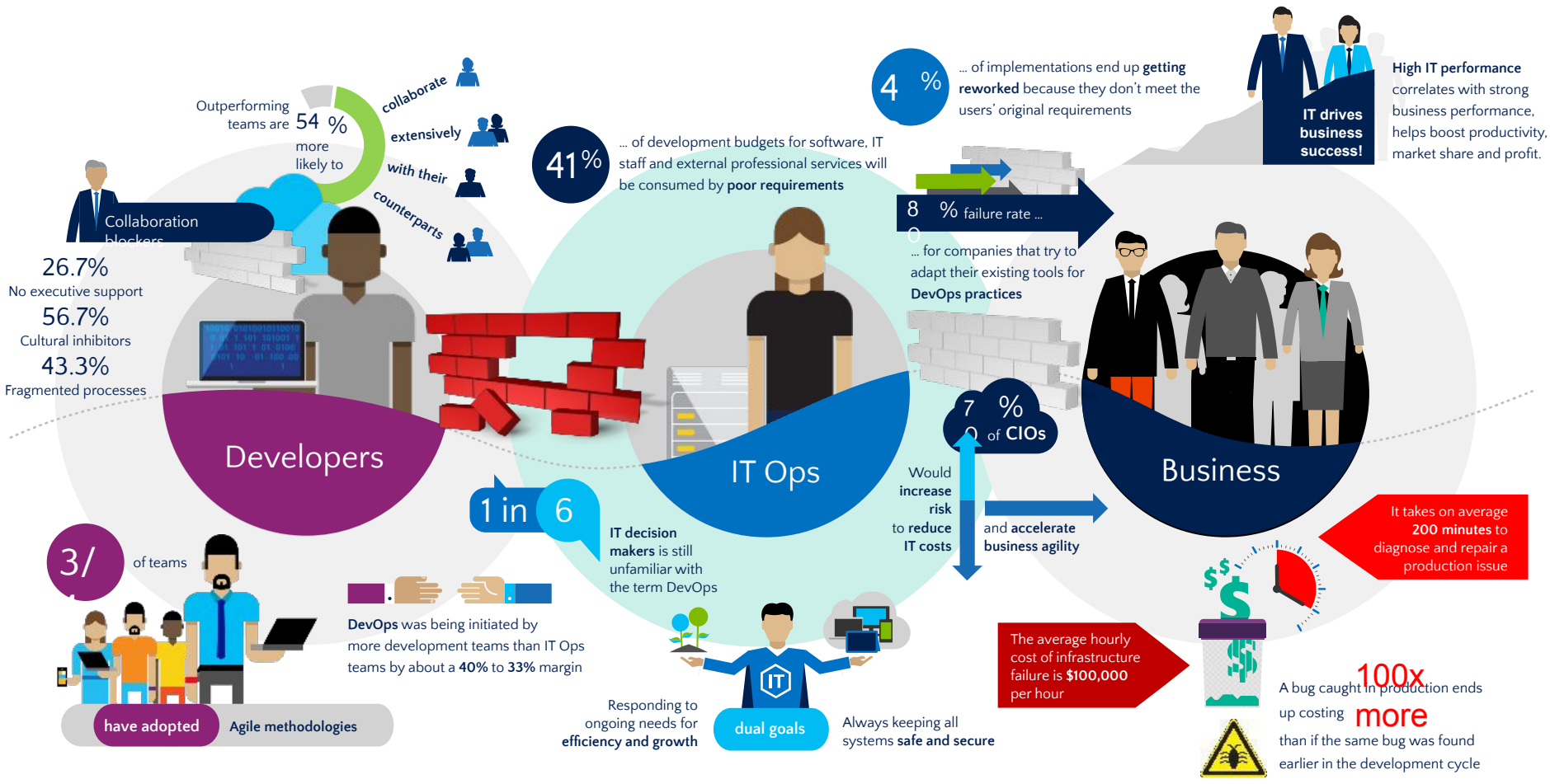
# DevOps: the three stage conversation



| 1 | People | 2 | Process | 3 | Products |
|---|---|---|---|---|---|

# The consequences of inefficiency

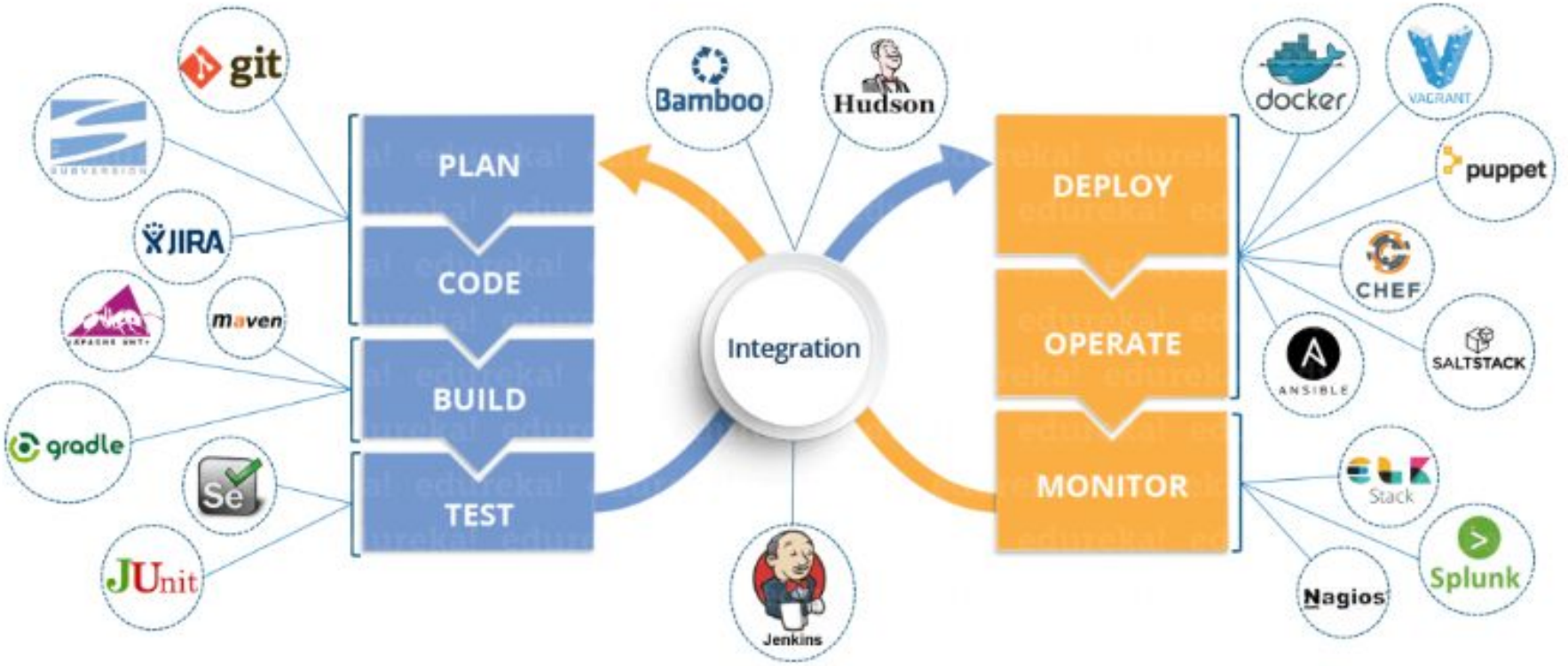


# Introduction to Devops

## What is DevOps?

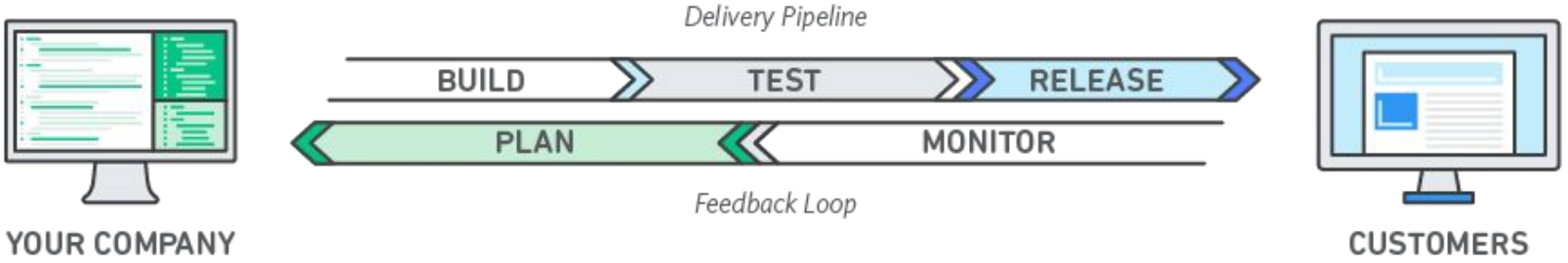
# Introduction to Devops

DevOps (development and operations) is an enterprise software development phrase used to mean a type of agile relationship between development and IT operations.
The goal of DevOps is to change and improve the relationship by advocating better communication and collaboration between these two business units.



# DevOps Model Defined

DevOps (development and operations) is an enterprise software development phrase
used to mean a type of agile relationship between development and IT operations.
The goal of DevOps is to change and improve the relationship by advocating
better communication and collaboration between these two business units.



# List of DevOps Activities

- Infrastructure as Code (IaC)
- Continuous Integration
- Automated Testing
- Continuous Deployment
- Release Management
- App Performance Monitoring
- Load Testing & Auto-Scale

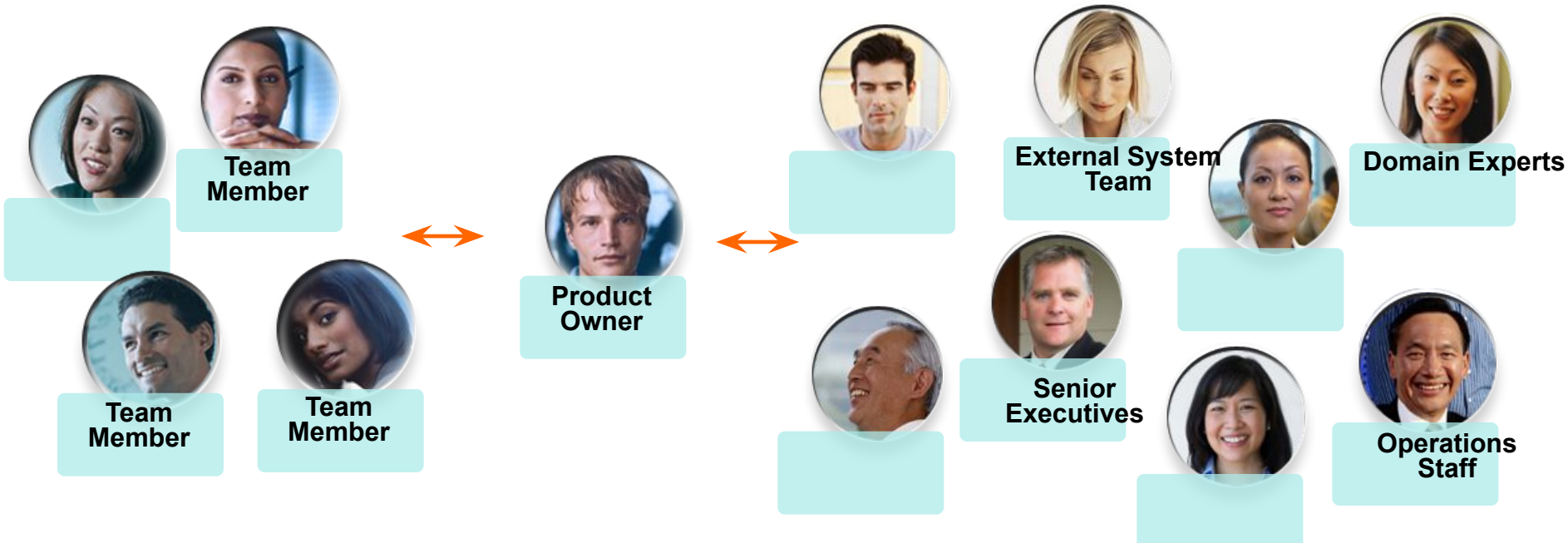
*http://www.itproguy.com/devops-practices/*

- Availability Monitoring
- Change/Configuration Management
- Feature Flags
- Automated Environment De-Provisioning
- Self Service Environments
- Automated Recovery (Rollback & Roll-Forward)
- Hypothesis Driven Development
  - –Testing in Production
  - –Fault Injection
  - –Usage Monitoring/User Telemetry

# List of DevOps Practices

# Adopting DevOps in the Enterprise: People/Culture

- Common Business Objectives
  - Common measures of Success
- Building a DevOps Culture
  - There is no Silver Bullet
  - *Right* People are needed
  - *Everyone is responsible for Delivery*



# DevOps Benefits

## Strong IT Performance is a competitive advantage

Firms with high-performing IT organizations were 2x as likely to exceed their profitability, market share, and productivity goals

## DevOps Practices improve IT performance


## Deploy code 30x faster

and with 200x shorter lead time as compared to their lower-performing peers

## Have 60x fewer failures

and recover from failure 168x faster as compared to their lower-performing peers

*Source:*
*https://puppetlabs.com/*

# Benefits of DevOps

## Speed

Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables your developers and operations teams to achieve these results. For example, microservices and continuous delivery let teams take ownership of services and then release updates to them quicker.

## Rapid Delivery

Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers' needs and build competitive advantage. Continuous integration and continuous delivery are practices that automate the software release process, from build to deploy.

# Benefits of DevOps

## Reliability

Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users. Use practices like continuous integration and continuous delivery to test that each change is functional and safe. Monitoring and logging practices help you stay informed of performance in real-time.

## Scale

Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner

# Benefits of DevOps

## Improved Collaboration

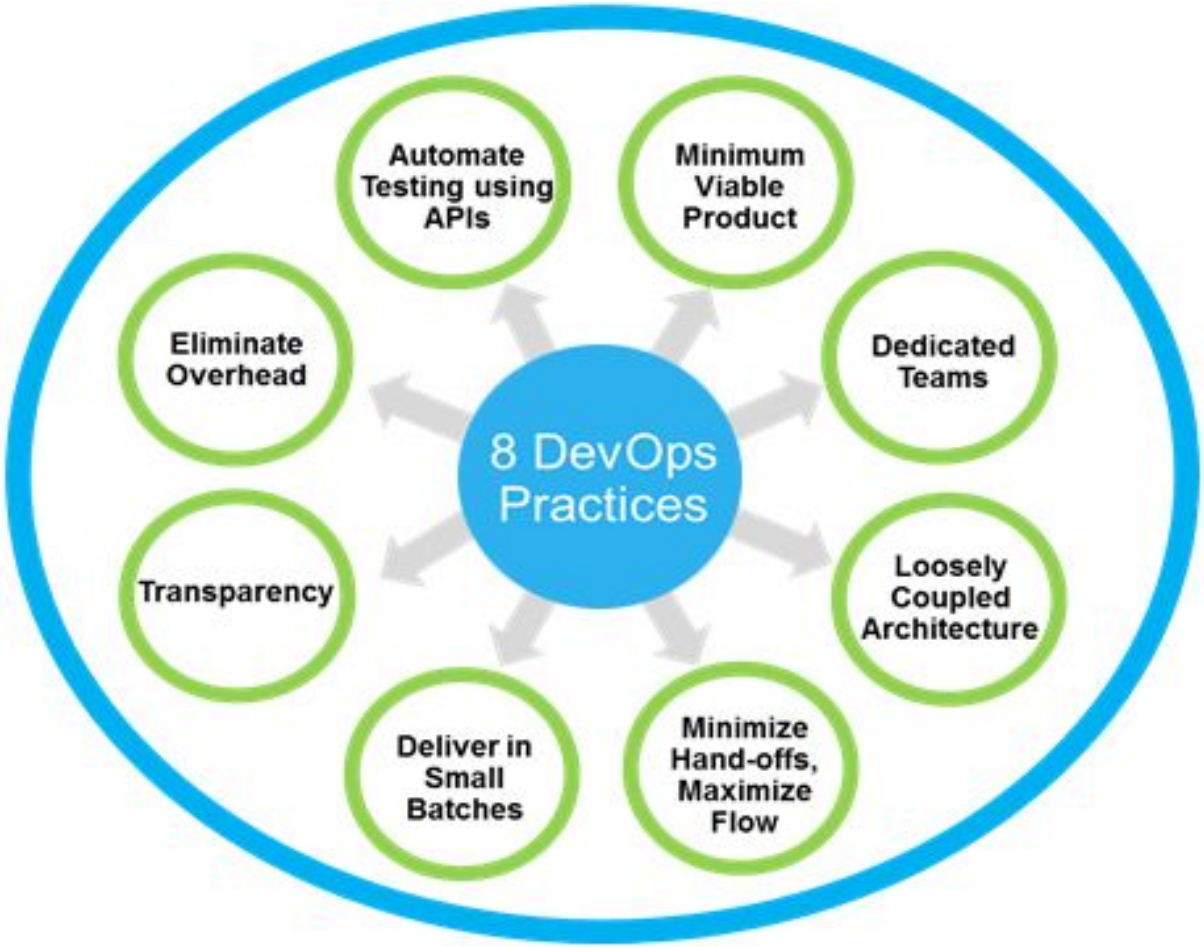
Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).

## Security

Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and policy as code, you can define and then track compliance at scale.

# DevOps Improvement on Organization

There are many specific practices that individuals and organizations have found to help their devOps.



# How to Adopt a DevOps Model

## DevOps Cultural Philosophy

- Transitioning to DevOps requires a change in culture and mindset. At its simplest, DevOps is about removing the barriers between two traditionally siloed teams, development and operations. In some organizations, there may not even be separate development and operations teams; engineers may do both.
- With DevOps, the two teams work together to optimize both the productivity of developers and the reliability of operations. They strive to communicate frequently, increase efficiencies, and improve the quality of services they provide to customers. They take full ownership for their services, often beyond where their stated roles or titles have traditionally been scoped by thinking about the end customer's needs and how they can contribute to solving those needs.

# How to Adopt a DevOps Model

## DevOps Cultural Philosophy

- Quality assurance and security teams may also become tightly integrated with these teams. Organizations using a DevOps model, regardless of their organizational structure, have teams that view the entire development and infrastructure lifecycle as part of their responsibilities.

# Success Stories with DevOps

laminarmedica
insulated shipping systems

**Laminar Medica** reduced new product development time and costs by 25%, contributing to 10% increase in competitive wins

SIBRA
Ingenieurgesellschaft für Datentechnik mbH
Germany

**SIBRA GmbH** keeps a vital project within budget and on time

Nationwide®
*On Your Side™*

**Nationwide** improved code quality 50%, reduced end-user downtime by 70%, and increased on-time delivery 90%

A healthcare information provider cuts deployment time down to minutes

sandhata™
we integrate your business

**Sandhata** increased productivity 100%, and added tens of millions in new revenue

IBM

**IBM CICS development team** simplifies software builds and helps support agile development, improve collaboration


**Sky Bet** monitors the online customer experience to increase overall revenues

招商銀行
CHINA MERCHANTS BANK

**China Merchants Bank** profits from a unified collaboration platform

**IBM Watson** team is reducing delivery cycles from 9 weeks to 3 weeks, & has achieved zero maintenance window downtime

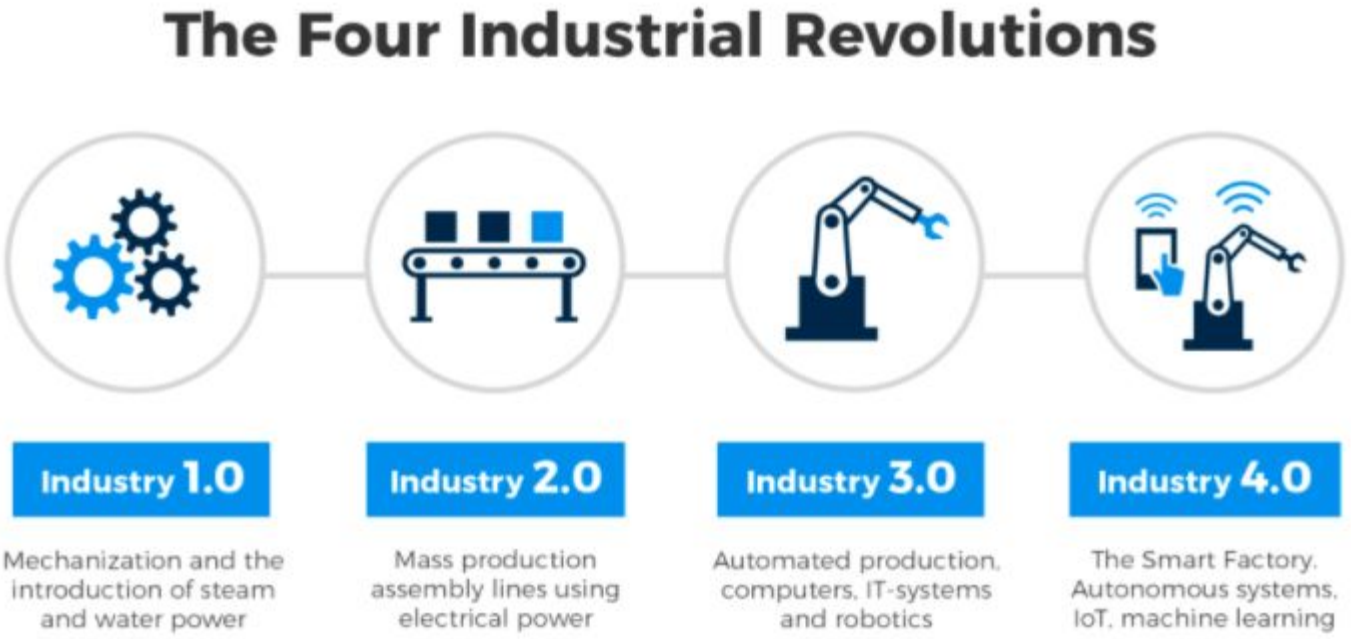
inter
VERSICHERUNGSGRUPPE

**INTER Versicherungsgruppe** increases productivity in application development

AON Integramark

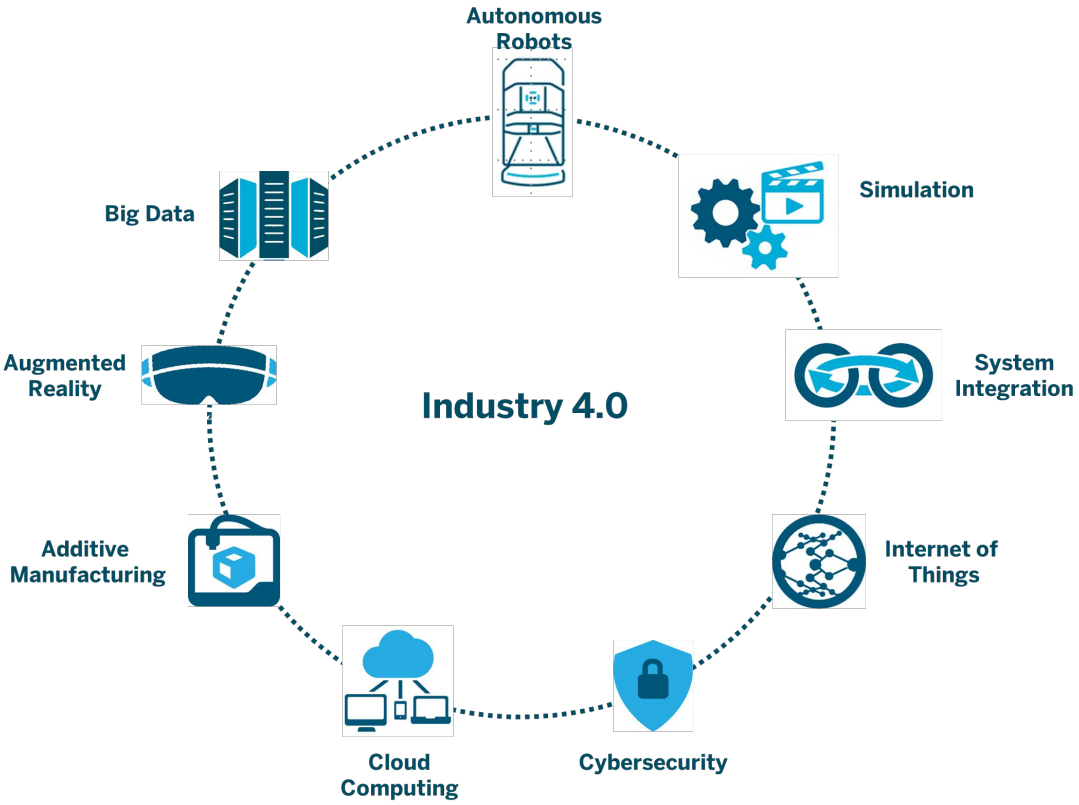
**Aon Integramark** establishes a dynamic SOA environment that automates data synchronization

# Revolutions of Industry



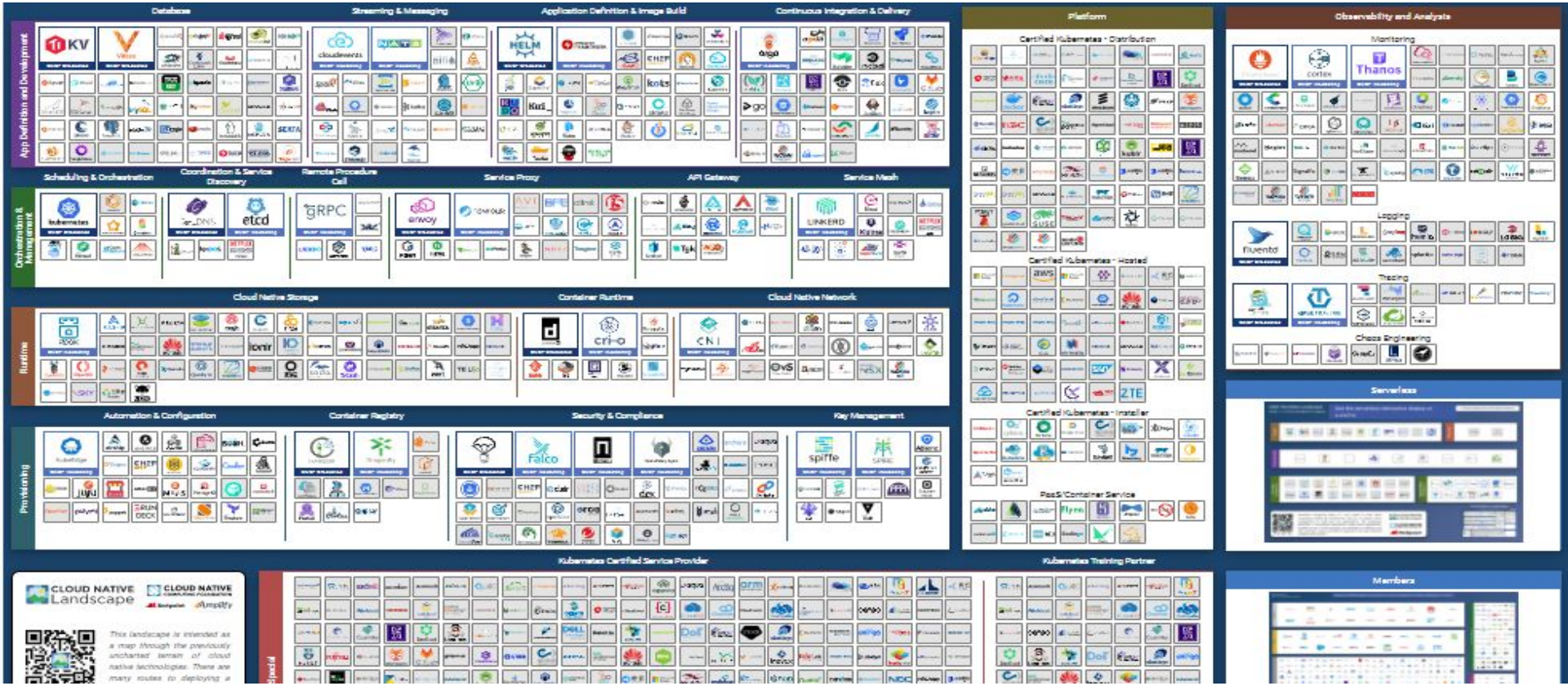
# Different Pillars of Industrial 4.0

- Revolution of Industrial 4.0 and different pillars



# Revolution of Cloud v3.0 – Cloud Native Devops Tools

- The Devops is working closely with Cloud Native Application(Open Source)
- Below is the landscape of Cloud Native Application ( https://landscape.cncf.io/ )



# DevOps Practices

The following are DevOps best practices:

- Continuous Integration
- Continuous Delivery
- Microservices
- Infrastructure as Code
- Monitoring and Logging
- Communication and Collaboration

# Continous Integration(CI)

- *Continuous integration are  the point of entry for many new DevOps  organizations. As an extension of agile, CI  servers enable automated build and test, along with varying levels of notifications fundamental to keeping agile efforts on track.*

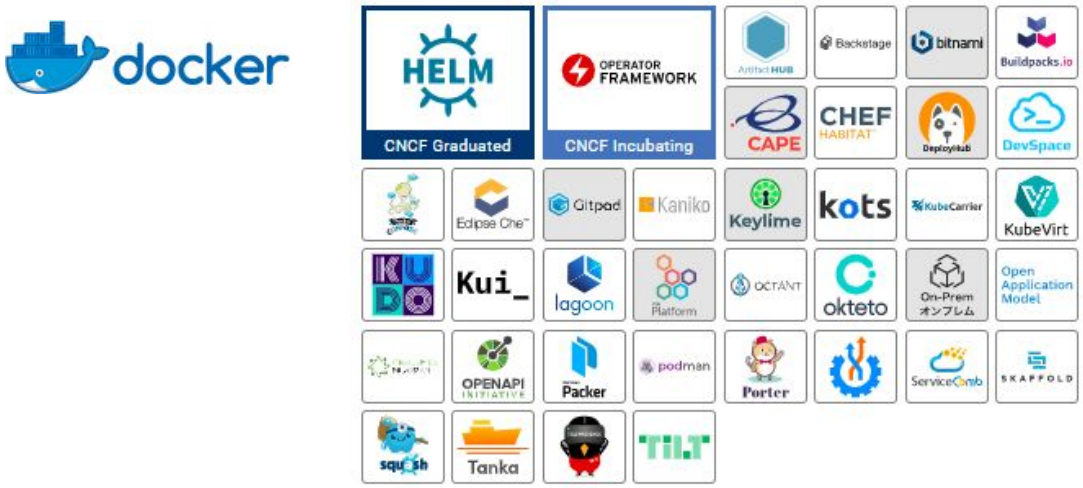
# Continuous Delivery(CD)

- *Source Code Management logical place to start a review of DevOps tools is with code repositories, as they are storing the code that makes up all of the applications deployed*



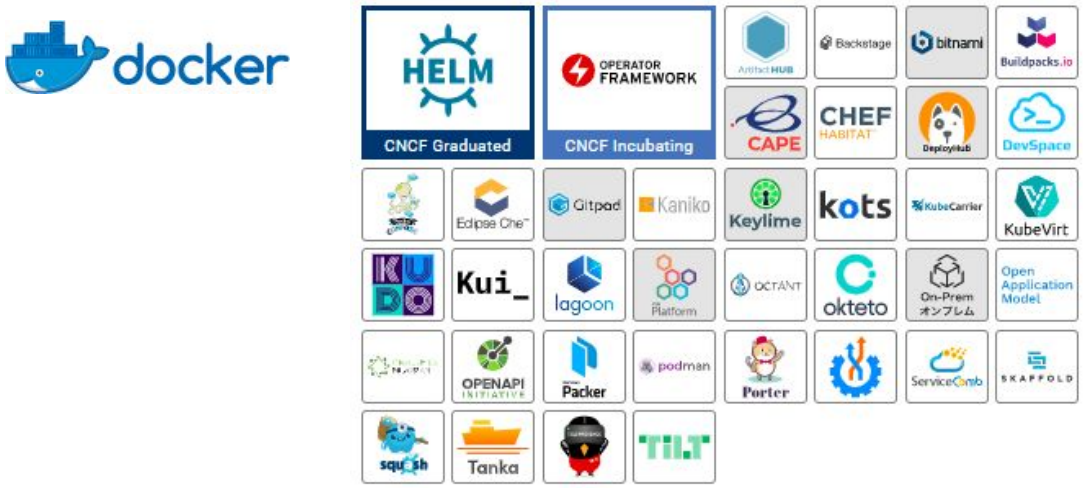
# Microservices

- Cloud native application definition and image building are important parts of automating application delivery. Images and application definitions provide you with a source to deploy and scale cloud native microservices.
- Images are binaries that provide an authoritative source for container and virtual appliance deployments. Application definitions define all the resources required by your application. As a result, developers benefit from programmatically repeatable deployments.



# Microservices

- These descriptions of images and application definitions may be a little abstract, so let's take a closer look at each.
- Application Definitions have 2 parts: application components and service resources.
- Application components are chunks of code that enable specific functionality in your cloud native application.
- Service resources, on the other hand, abstract away module dependencies and enable data transport between nodes within an app.



# Microservices

- Containers have enjoyed rapid adoption across enterprises and industries. Containers are quick and easy to spin up, use fewer resources than a fully virtualized machine and, when combined with DevOps automation, can be scaled quickly to meet the needs of the application. Still, when a given application or service grows in complexity, more than just containers is needed. Infrastructure management, scalability and the ability to manage groups of containers uniformly are all necessary so the application can be created with dependencies and the container environment is useful. Container management tools address these needs.

# Infrastructure as Code

- Infrastructure as code is a practice in which infrastructure is provisioned and managed using code and software development techniques, such as version control and continuous integration. The cloud's API-driven model enables developers and system administrators to interact with infrastructure programmatically, and at scale, instead of needing to manually set up and configure resources.
- Thus, engineers can interface with infrastructure using code-based tools and treat infrastructure in a manner similar to how they treat application code. Because they are defined by code, infrastructure and servers can quickly be deployed using standardized patterns, updated with the latest patches and versions, or duplicated in repeatable ways.

# Infrastructure as Code

## Policy as Code

With infrastructure and its configuration codified with the cloud, organizations can monitor and enforce compliance dynamically and at scale. Infrastructure that is described by code can thus be tracked, validated, and reconfigured in an automated way. This makes it easier for organizations to govern changes over resources and ensure that security measures are properly enforced in a distributed manner (e.g. information security or compliance with PCI-DSS or HIPAA). This allows teams within an organization to move at higher velocity since non-compliant resources can be automatically flagged for further investigation or even automatically brought back into compliance.

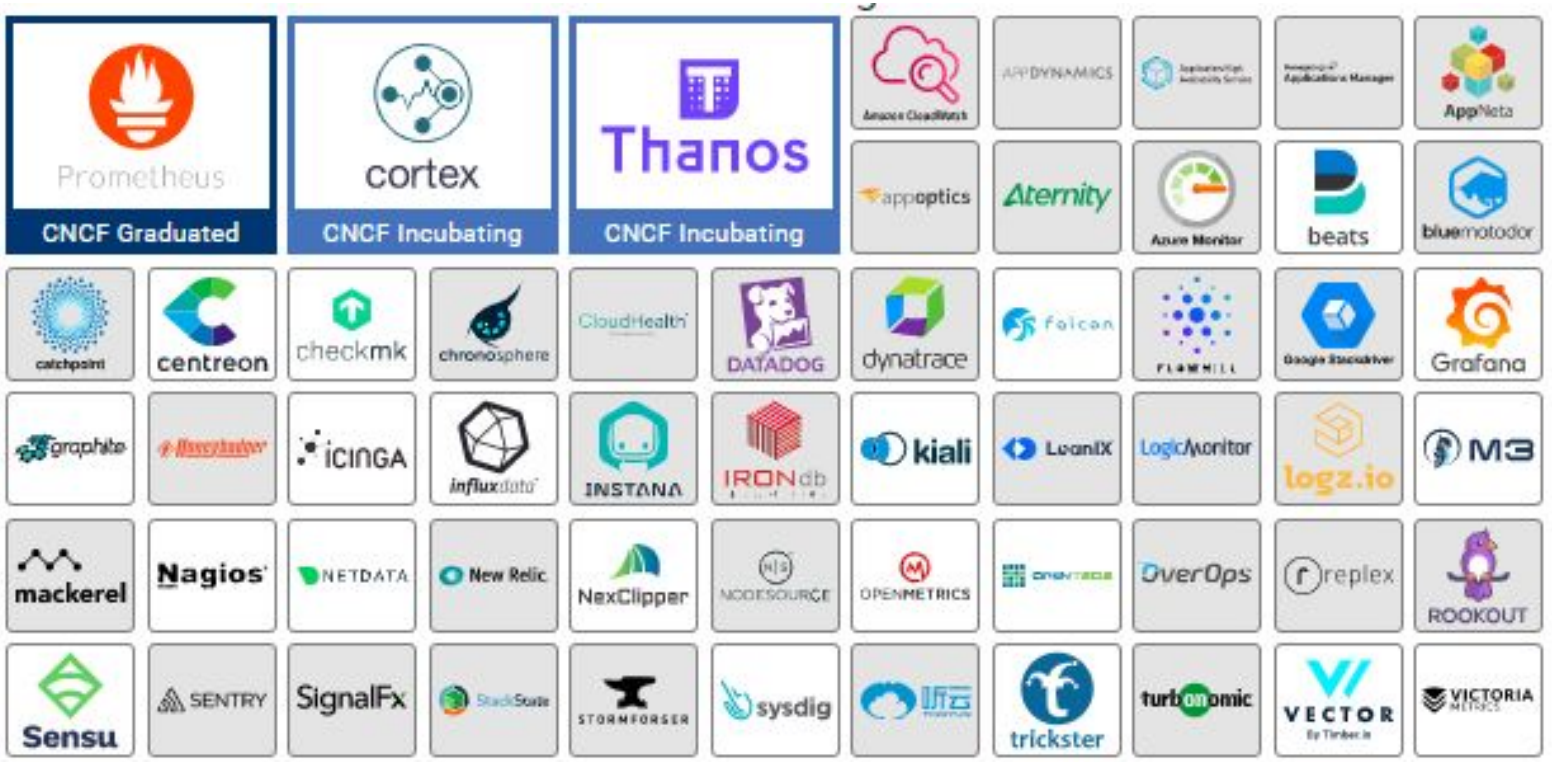
# Infrastructure as Code

## Configuration Management

Developers and system administrators use code to automate operating system and host configuration, operational tasks, and more. The use of code makes configuration changes repeatable and standardized. It frees developers and systems administrators from manually configuring operating systems, system applications, or server software.

# Monitoring and Logging

***Monitoring*** *from a DevOps perspective encompasses not only application performance but also can include user interaction, infrastructure capacity and delivery. Leading tools in this category have stretched the envelope in UX design to give DevOps teams visibility into a constantly scaling environment. The best of the best in this category can give a birds-eye view of expansive deployments, while allowing for quick drill-down to the gritty details.*

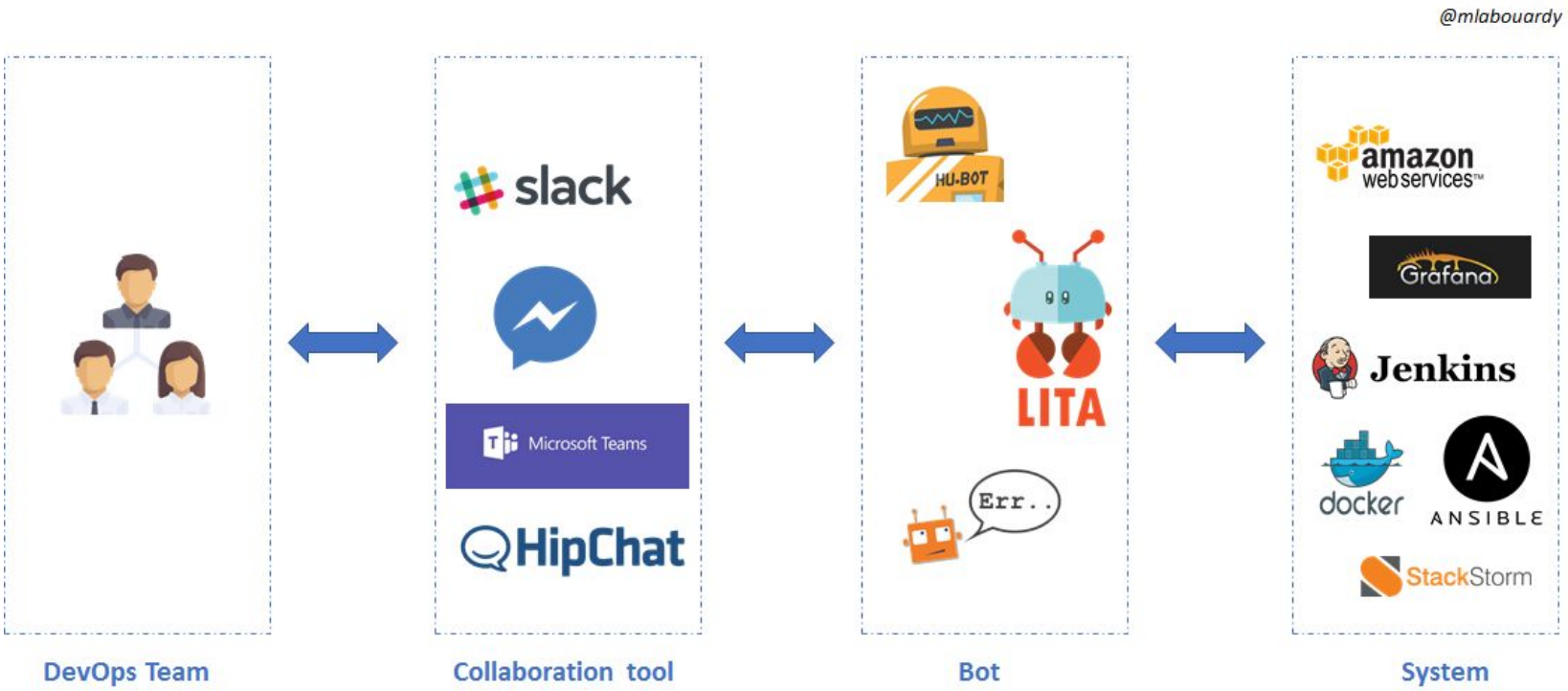


# Monitoring and Logging

- DevOps teams rely on log management to better run their environments and organizations. Without a log management tool, it can be difficult — if not almost impossible — to manage and analyze the volume of logs in most environments.
- Discovering and identifying trouble spots through either automation or search is essential.

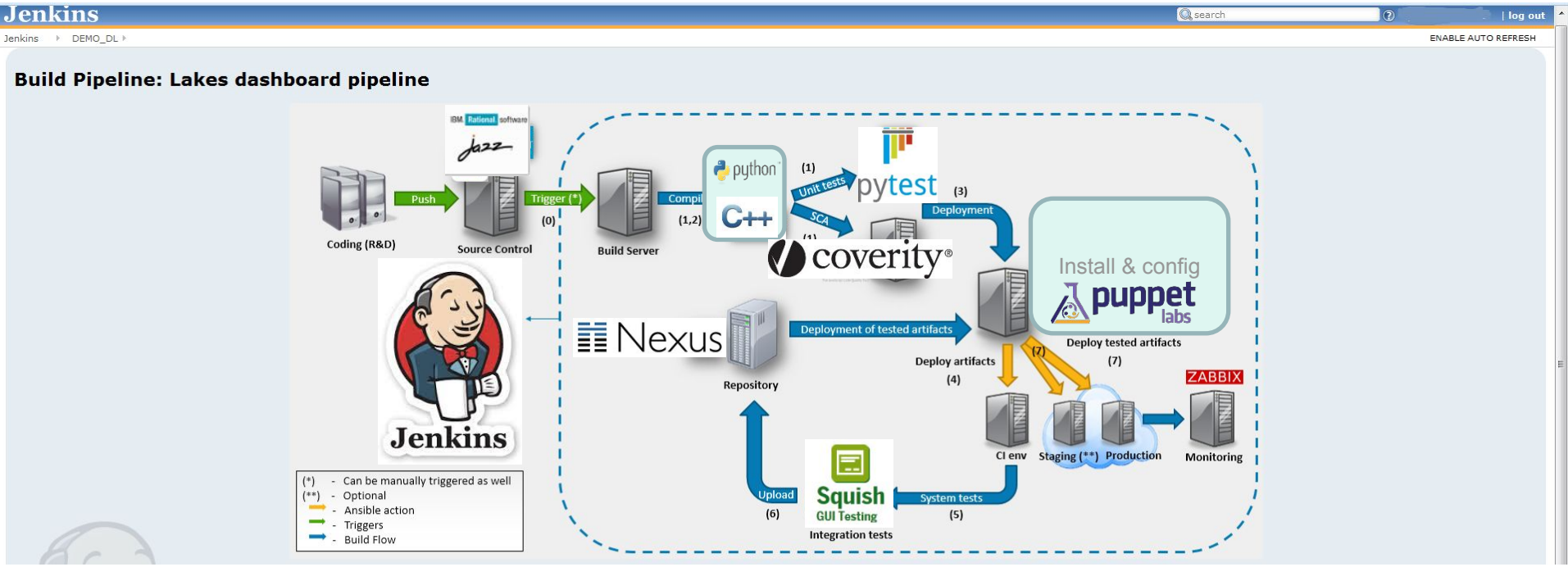
# Communication and Collaboration

- ChatOps originally described the use of APIs to embed chat tools into systems alerting. Vendors identified customer need and some started integrating chat capabilities into the DevOps toolchain to facilitate systems management and troubleshooting. Today, using ChatOps, it is possible to check in code changes and inform teams of the results, helping resolve problems quickly.



# DevOps Implementation Example

- A typical DevOps adoption example is shown as below:



# Lab: Understanding Devops Landscape and Tools

- Access to URL https://landscape.cncf.io/ and understand the Cloud Native Tools that aligning with the Devops Practices

# Topic 2
# Continuous Integration & Delivery (CI/CD)

# What is Integration

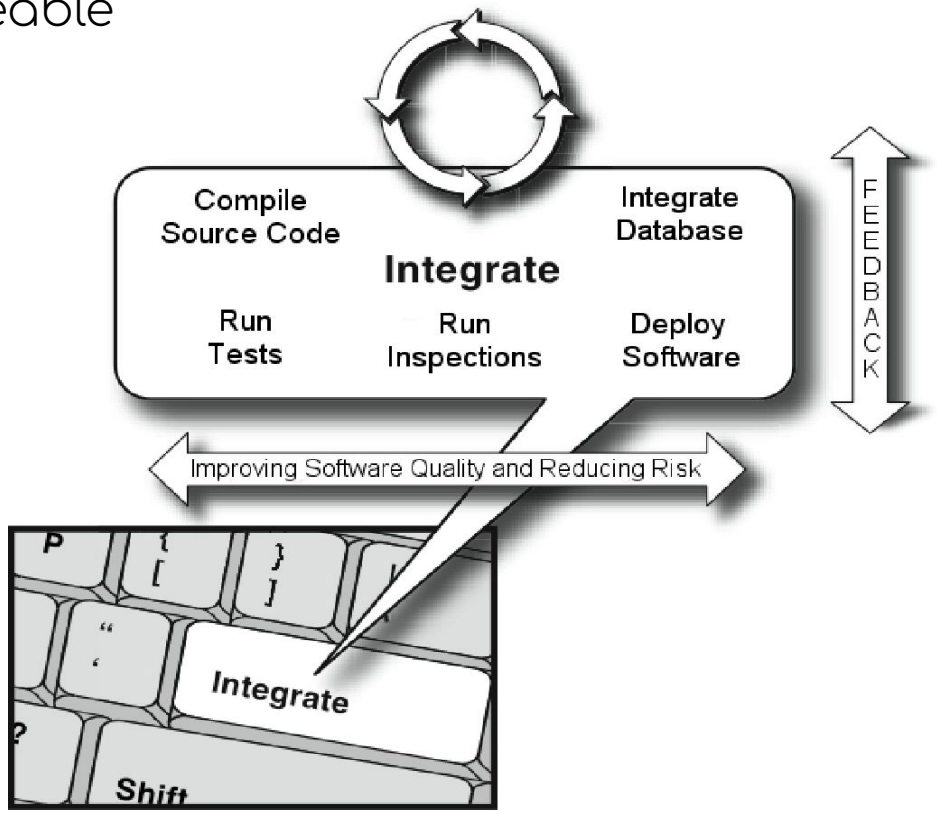
## Integration is …
## Making different modules work together

Modularization

1. enables team development
2. makes complex systems manageable
3. Modules have to work together
   i.e. they must be integrated

Integrated Modules do successfully

- compile
- run
- pass test
- deploy



# Why Integration

You have a broken integration when:

- Source code server does not build successfully
- Shared component works in one system, but breaks others
- Unit tests fail
- Code quality fails (coding conventions, quality metrics)
- Deployment fails

*The earlier you can detect problems, the easier it is to resolve them*

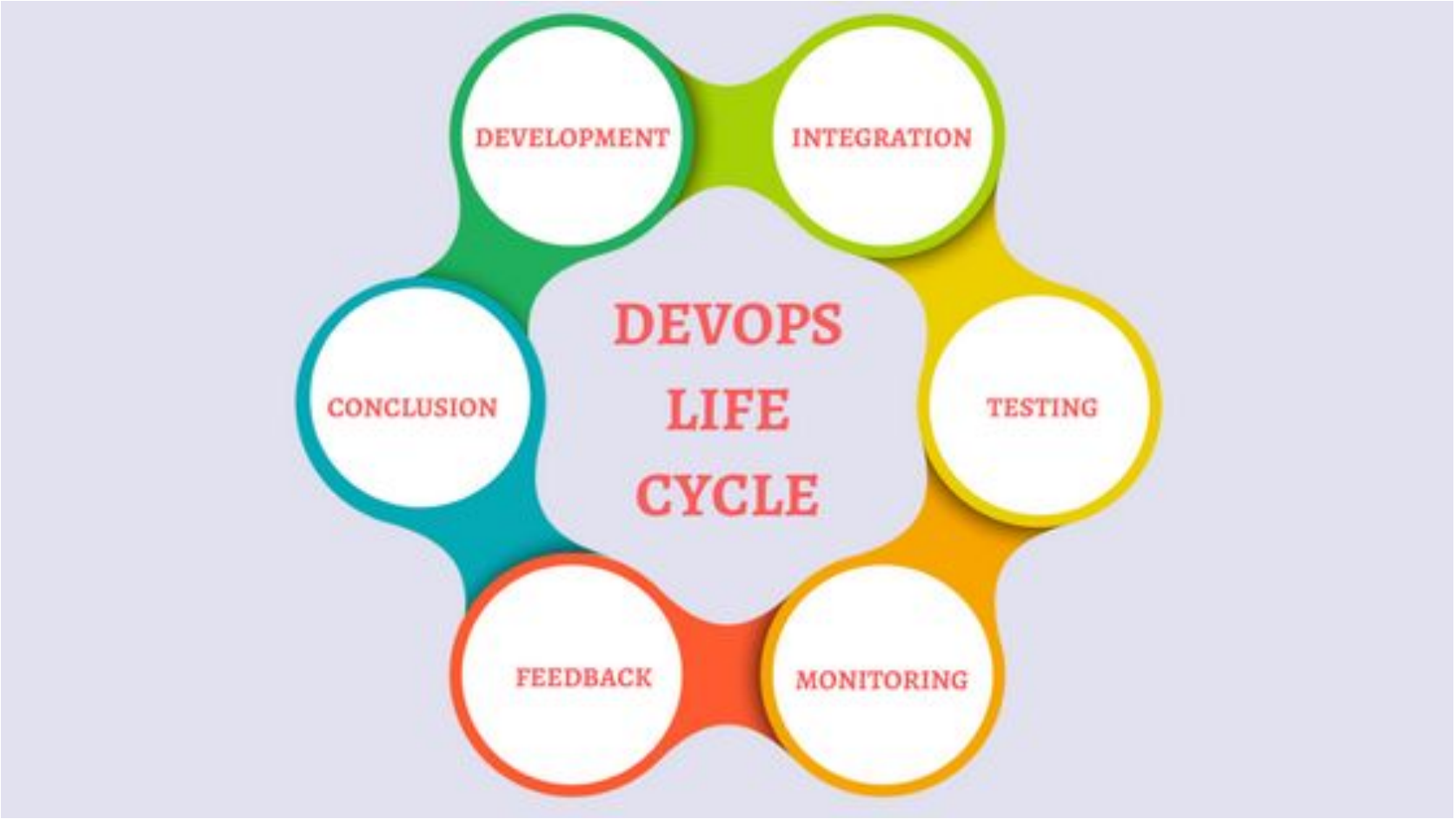
# Benefits CI/CD

Small (changes) + Fast (delivery) = Better (quality)

- Time to Market goes down
- Quality improves
- CD limits your work in progress
- Shorten lead time to changes
- Improve mean time to recover

# DevOps Lifecycle

To understand the DevOps, it is essential to understand the stages of the DevOps lifecycle. The continuous DevOps lifecycle incorporates seven stages as discussed below:



# DevOps Lifecycle

## Continuous Development

The first phase of the DevOps lifecycle, Continuous Development involves the phase in which all the planning and coding is done for the software application.

This is the phase where the team members sit down and visualize the outcome or, in other words, how the software application will turn out to be.

Once the vision is in place, the developers start writing and maintaining the code.

There are several tools used for maintaining the code – the process is referred to as Source Code Management – and some of these tools are SVN, Git, Jira and CVS.

# DevOps Lifecycle

## Continuous Integration

After the phase of Continuous Development comes Continuous Integration, and the first phase automatically translates into the second one.

This phase involves different steps which include planning of test execution to be carried out in the testing phase, as well as understanding the project requirements provided by the client to incorporate the necessary features in the software product.

The stage also involves changes made into the source code frequently, and often on a daily or weekly basis.

Many refer to the Continuous Integration stage as the heart of the overall DevOps lifecycle, as this is the phase in which problems in the code can be detected early on.

# DevOps Lifecycle

## Continuous Testing

The Continuous Testing phase involves the phase in which the developed code is continuously tested to look for bugs, defects and flaws.

This is also the phase where the usability of the software is tested using the set of best practices for QA, and it is determined whether the software meets the specifications defined by the client.

Continuous testing is carried out using automation testing tools, which can be open source tools like Selenium or advanced test management tools like QARA Enterprise and TestNG.

These tools allow the QA team to execute multiple tests in parallel in order to increase test coverage and increase the efficiency of the testing process.

The outcome from the testing is shared with the developers to make changes in the product.

# DevOps Lifecycle

## Continuous Monitoring

The Continuous Testing phase involves the phase in which the developed code is continuously tested to look for bugs, defects and flaws.

This is also the phase where the usability of the software is tested using the set of best practices for QA, and it is determined whether the software meets the specifications defined by the client.

Continuous testing is carried out using automation testing tools, which can be open source tools like Selenium or advanced test management tools like QARA Enterprise and TestNG.

These tools allow the QA team to execute multiple tests in parallel in order to increase test coverage and increase the efficiency of the testing process.

The outcome from the testing is shared with the developers to make changes in the product.

# DevOps Lifecycle

## Continuous Feedback

Feedback is important to computerized releases.

Continuous feedback provides information to all stakeholders with respect to what is being sent and what issues are being occurred.

In DevOps, feedback is considered as information that is gathered from the client. It is basically a contribution by the client to complete planning and development.

The gathered information contains significant data about the performance and any issue that is experienced by the end-user.

# DevOps Lifecycle

## Conclusion

DevOps is certainly an excellent way of carrying out application development.

It significantly improves the business performance of applications and allows the end users to directly contribute to the application development process, by sharing relevant feedback from the operational phase.

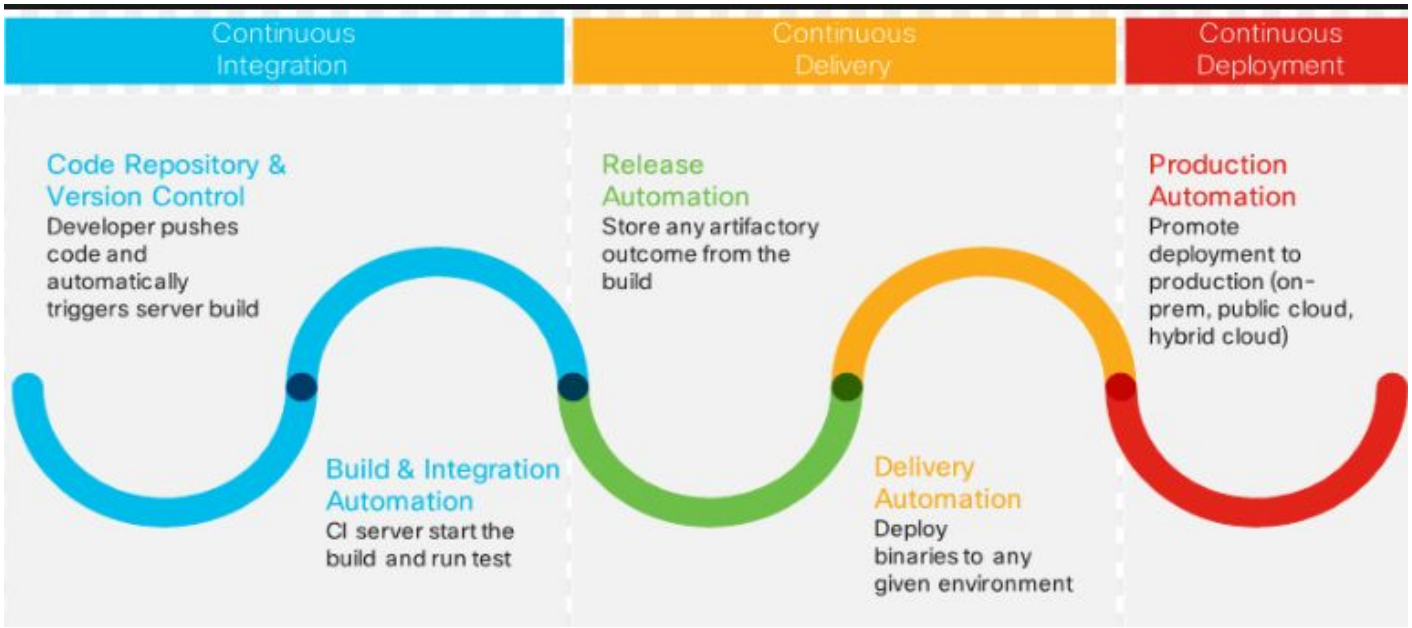
It is a method that we will continue to see in the future as a top option for developing dynamic applications that constantly evolve to meet new challenges.

# What is CI/CD

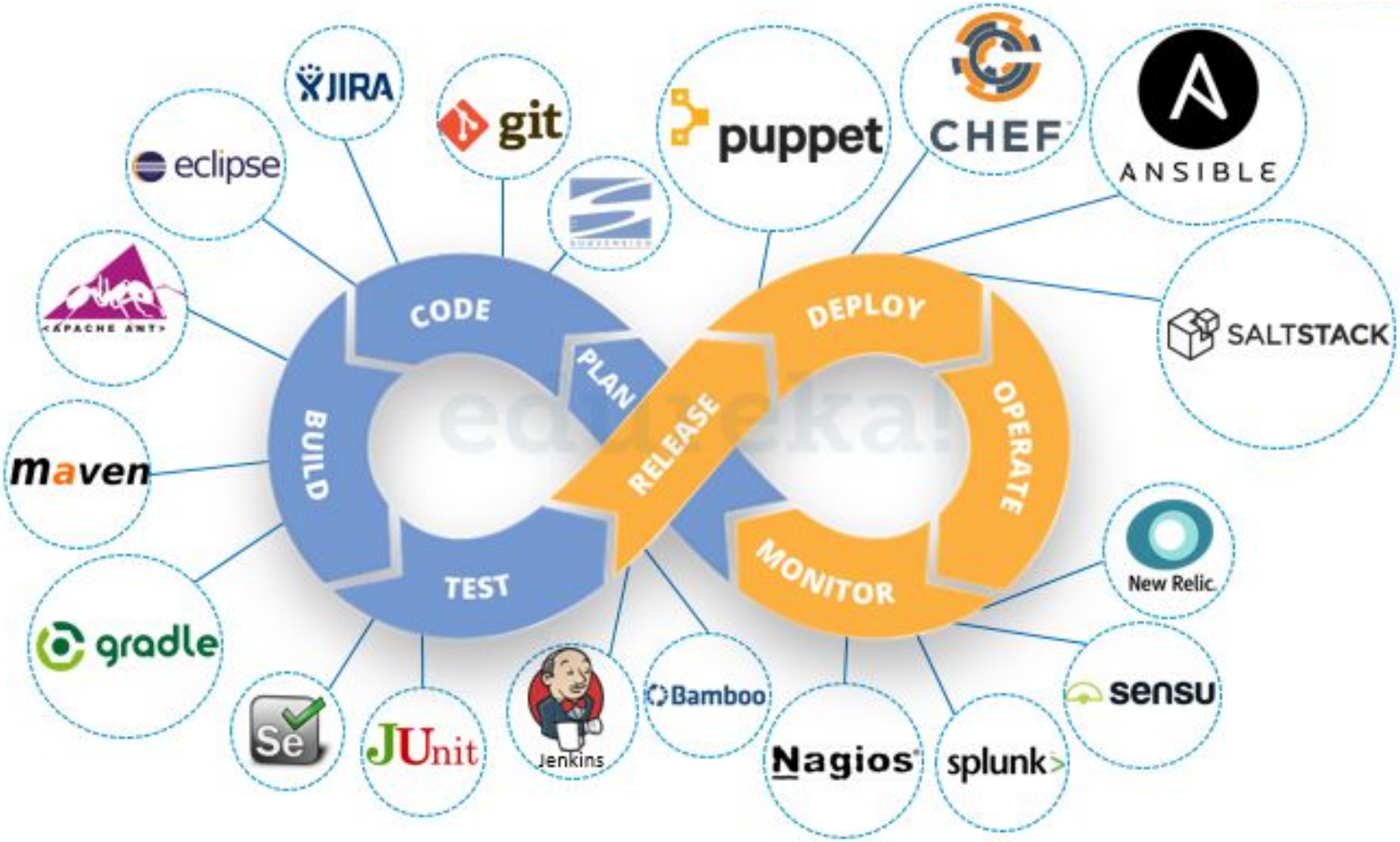
Continuous Integration is the most important part of DevOps that is used to integrate various *DevOps stages*

Continuous delivery(CD) is to packaging and deployment what CI is to build and test.

Software is built, configured, and packaged and its deployment orchestrated in such a way that it can be released to production in a software-defined manner (low cost, high automation) at any time.

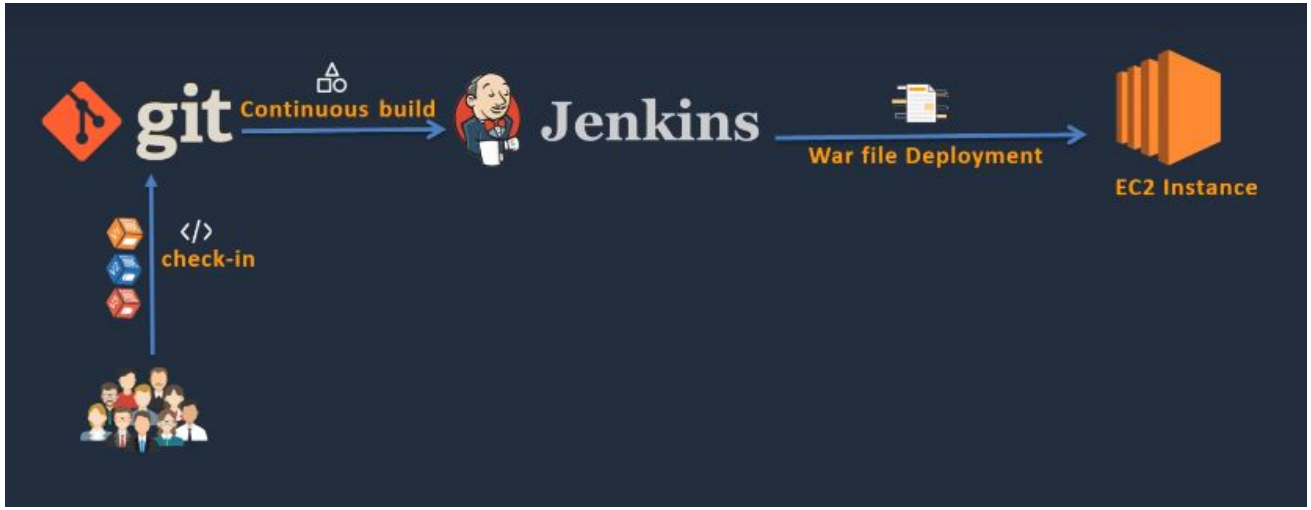


# DevOps CI/CD Tools



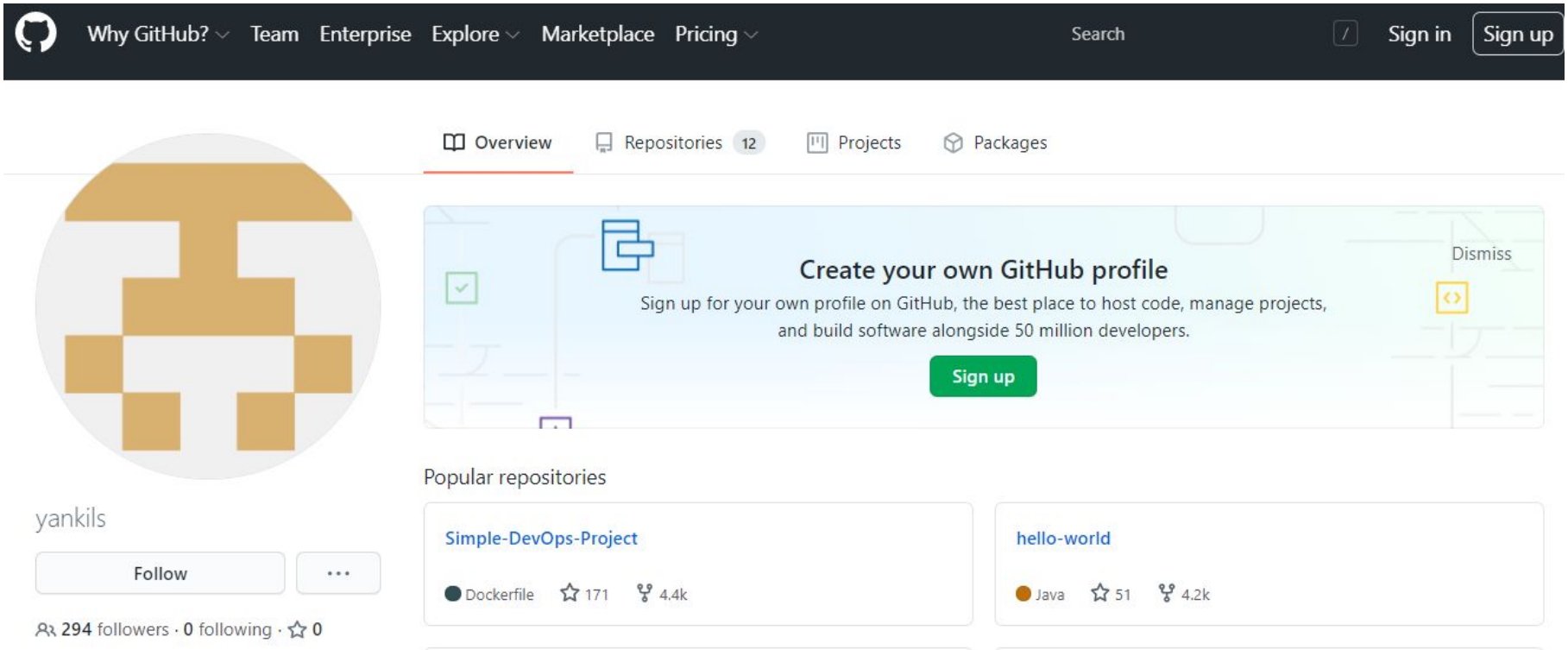
# Activity: Understanding of Devops life cycle with CI/CD Pipeline

- In this lab, we are going to understand Devops lifecycle of CI/CD Pipeline
- Development Phase: New Code release and uploaded to GIT
- Integration Phase: Integration between GIT(Code Versioning Control) and Jenkins for Build and Integration Automation
- Testing Phase: Observation of the Built Code output
- Monitoring: Observation the Output
- Feedback: Verified the Result
- Conclusion



# Development Phase: Understand GitHub Repository

- GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.
- Upload the new version of Source Code to GIT



# Integration Phase: Installation of Jenkins

Get the latest version of jenkins from https://pkg.jenkins.io/redhat-stable/ and install

yum -y install wget
sudo wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
yum -y install jenkins

Start Jenkins

# Start jenkins service
service jenkins start

# Setup Jenkins to start at boot,
chkconfig jenkins on

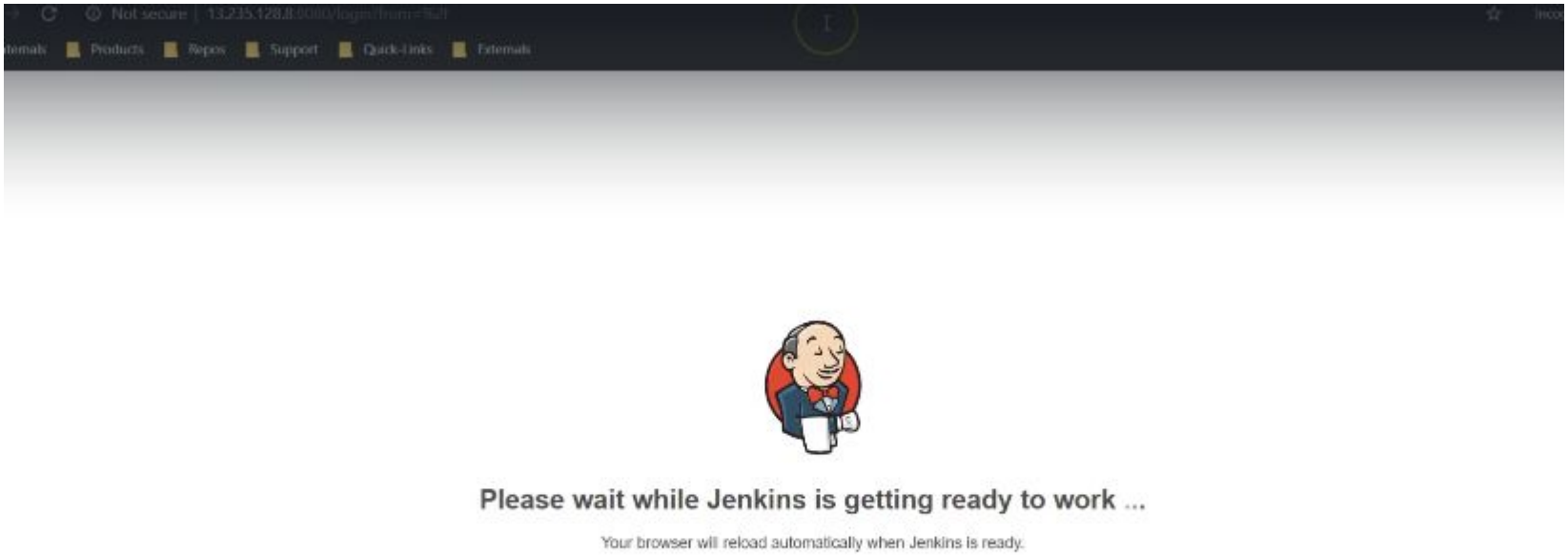
Accessing Jenkins
# Start jenkins service
service jenkins start

service jenkins status

# Integration Phase: Installation of Jenkins

Accessing Jenkins
By default jenkins runs at port 8080, You can access jenkins at

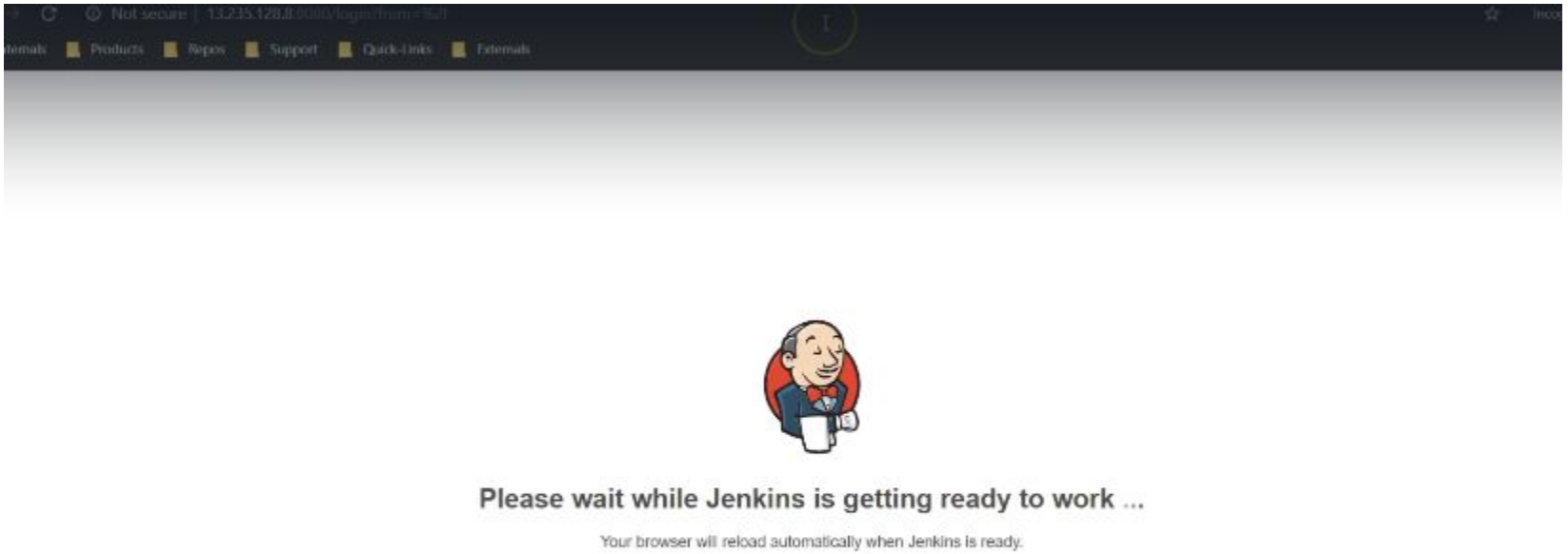
http://YOUR-SERVER-PUBLIC-IP:8080



# Integration Phase: Installation of Jenkins

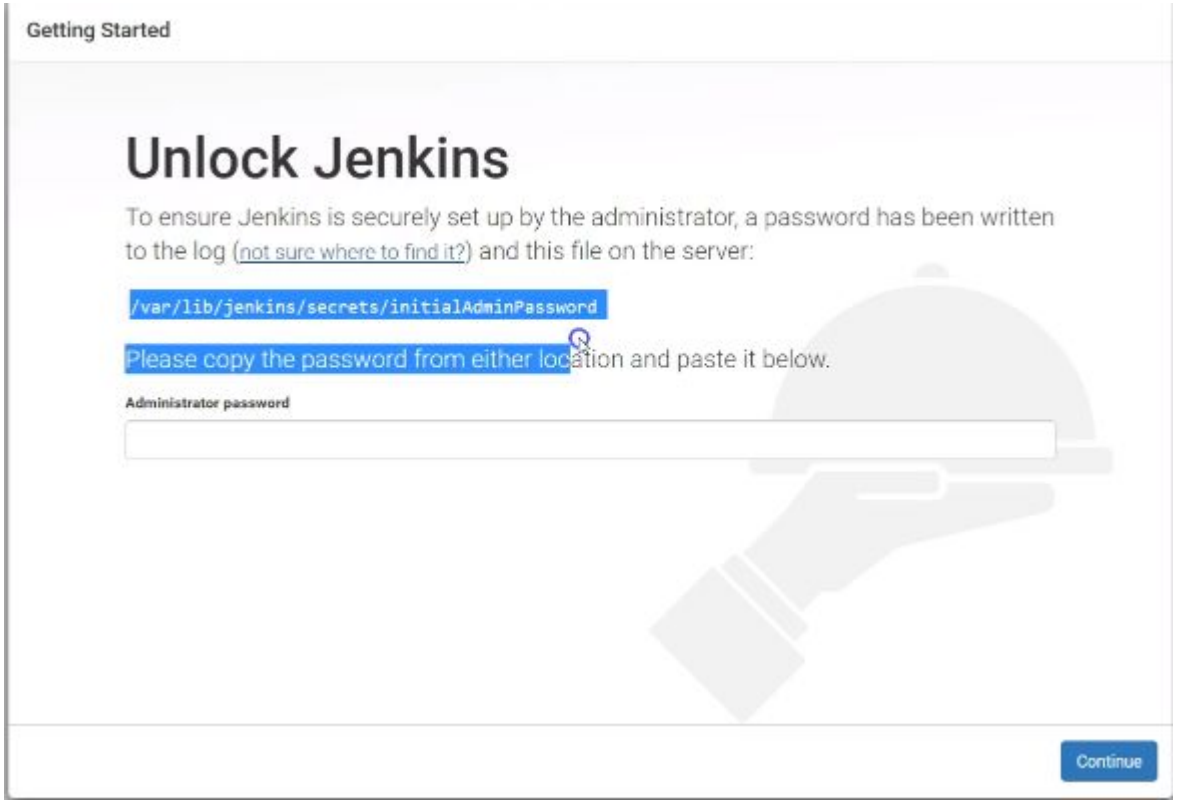
Accessing Jenkins
By default jenkins runs at port 8080, You can access jenkins at

http://YOUR-SERVER-PUBLIC-IP:8080



# Integration Phase: Installation of Jenkins

Unlock Jenkins with Password:
/var/lib/jenkins/secrets/initialAdminPassword

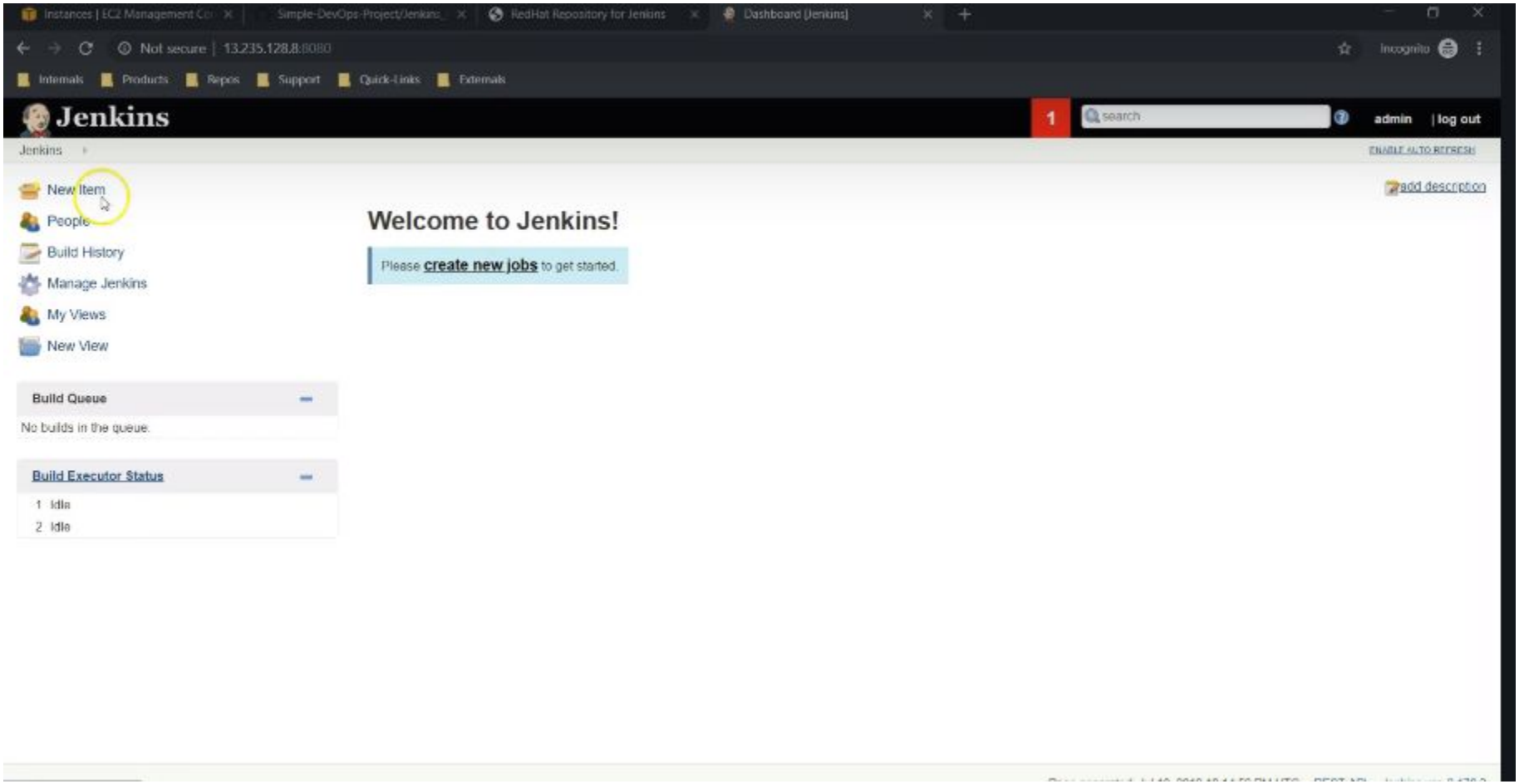


# Integration Phase: Installation of Jenkins

Successfully Login to Jenkins

# Testing Phase: Run Jenkins Job

Create “new item”

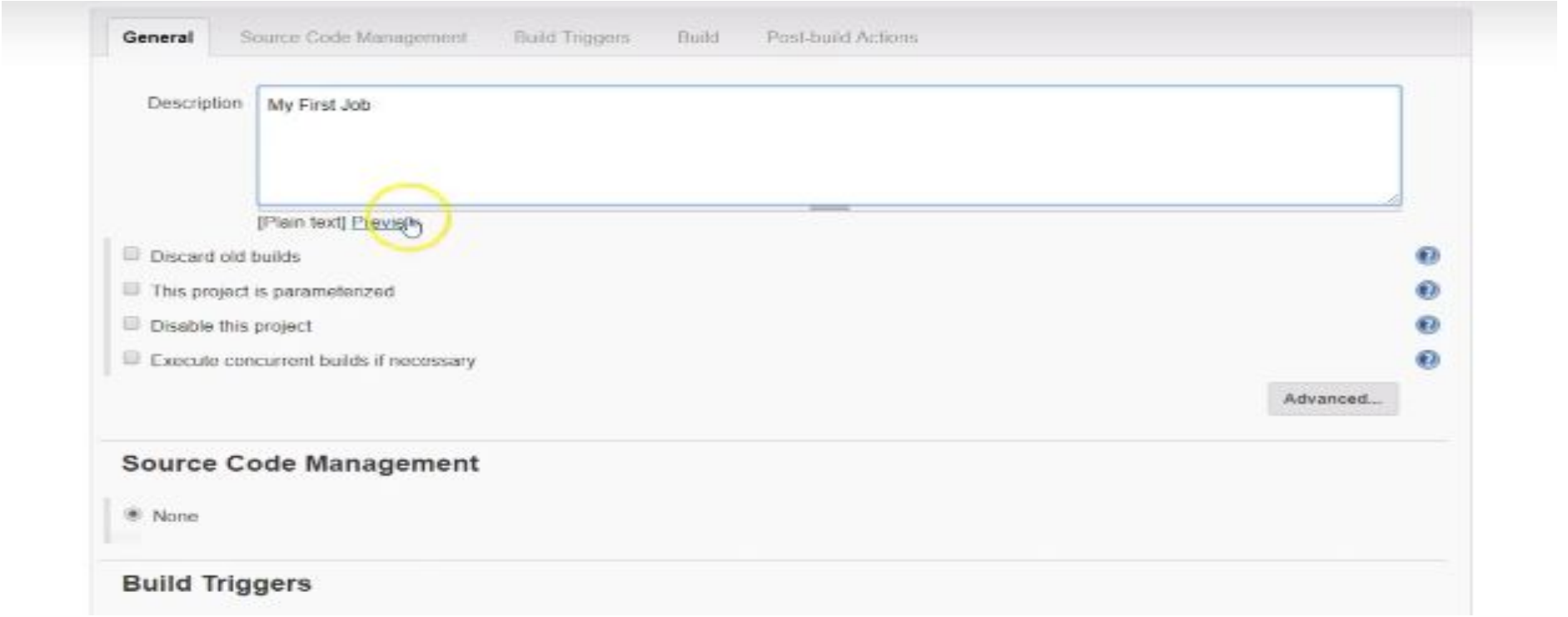


# Testing Phase: Run Jenkins Job

Enter an item name – My-First-Project
Chose Freestyle project

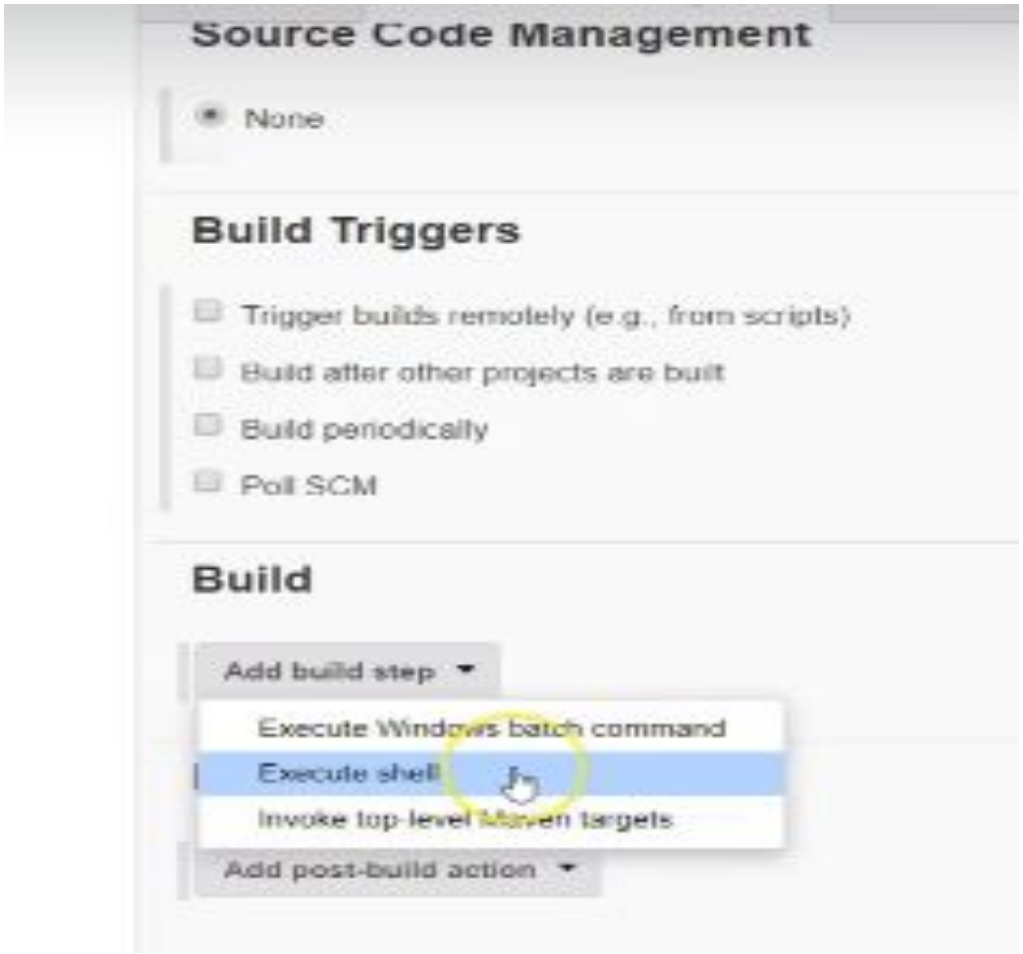
# Testing Phase: Run Jenkins Job

Under the Build section Execute shell: echo "Welcome to Jenkins Demo"



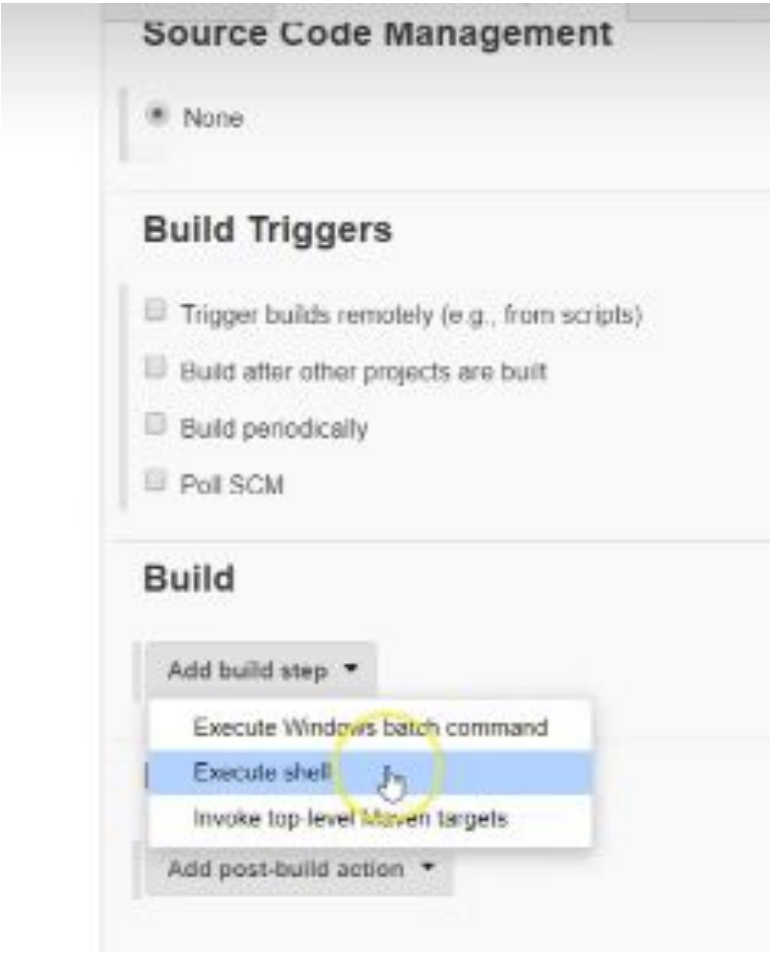
# Testing Phase: Run Jenkins Job

Under the Build section Execute shell:



# Testing Phase: Run Jenkins Job

echo "Welcome to Jenkins Demo"



# Testing Phase: Run Jenkins Job

Under the Build section Execute shell: echo "Welcome to Jenkins Demo"

# Monitoring Phase: Console Output

Save your job
Build job
Check "console output"

# Integration Phase: Install Git Packages on Jenkins Server

Install git packages on jenkins server
yum install git –y

Setup Git on jenkins console
Install git plugin without restart

Manage Jenkins > Jenkins Plugins > available > github
Configure git path

Manage Jenkins > Global Tool Configuration > git

# Integration Phase: Install Git Packages on Jenkins Server

Install git packages on jenkins server
yum install git –y

Setup Git on jenkins console
Install git plugin without restart

Manage Jenkins > Jenkins Plugins > available > github
Configure git path

Manage Jenkins > Global Tool Configuration > git

# Integration Phase: Install Maven Packages on Jenkins Server

- Maven is a build automation tool used primarily for Java projects.

Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.

Create maven directory under /opt
mkdir /opt/maven
cd /opt/maven

Downloading maven
wget
http://mirrors.estointernet.in/apache/maven/maven-3/3.6.1/binaries/apache-maven-3.6.1-bin.tar.gz
tar -xvzf apache-maven-3.6.1-bin.tar.gz

Rename the Folder
mv apache-maven-3.6.1 maven
cd maven

# Integration Phase: Install Maven Packages on Jenkins Server

Setup M2_HOME and M2 paths in .bash_profile of the user
and add these to the path variable
vi ~/.bash_profile
M2_HOME=/opt/maven/apache-maven-3.6.1
# M2_HOME=/opt/maven
M2=$M2_HOME/bin
PATH=<Existing_PATH>:$M2_HOME:$M2

logoff and login to check maven version
mvn --version

# Integration Phase: Install Maven Packages on Jenkins Server

Install maven plugin without restart
Manage Jenkins > Jenkins Plugins > available > Maven Invoker
Manage Jenkins > Jenkins Plugins > available > Maven Integration

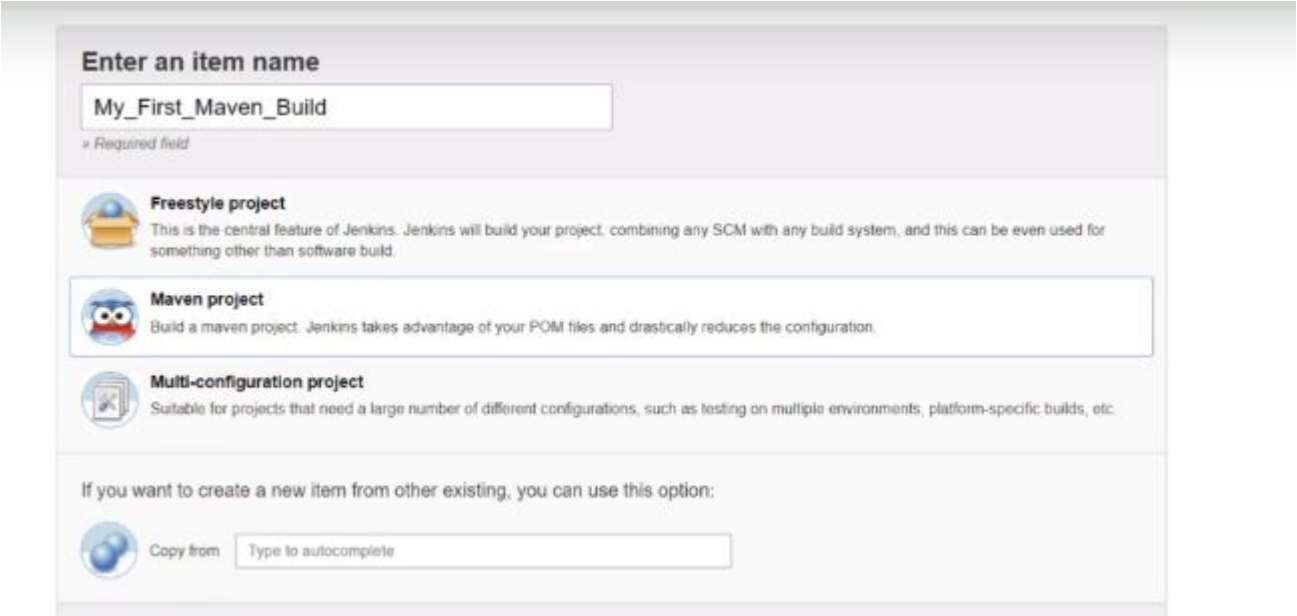
Configure maven path
Manage Jenkins > Global Tool Configuration > Maven

# Integration Phase: Install Maven Packages on Jenkins Server

Install maven plugin without restart
Manage Jenkins > Jenkins Plugins > available > Maven Invoker
Manage Jenkins > Jenkins Plugins > available > Maven Integration

Configure maven path
Manage Jenkins > Global Tool Configuration > Maven

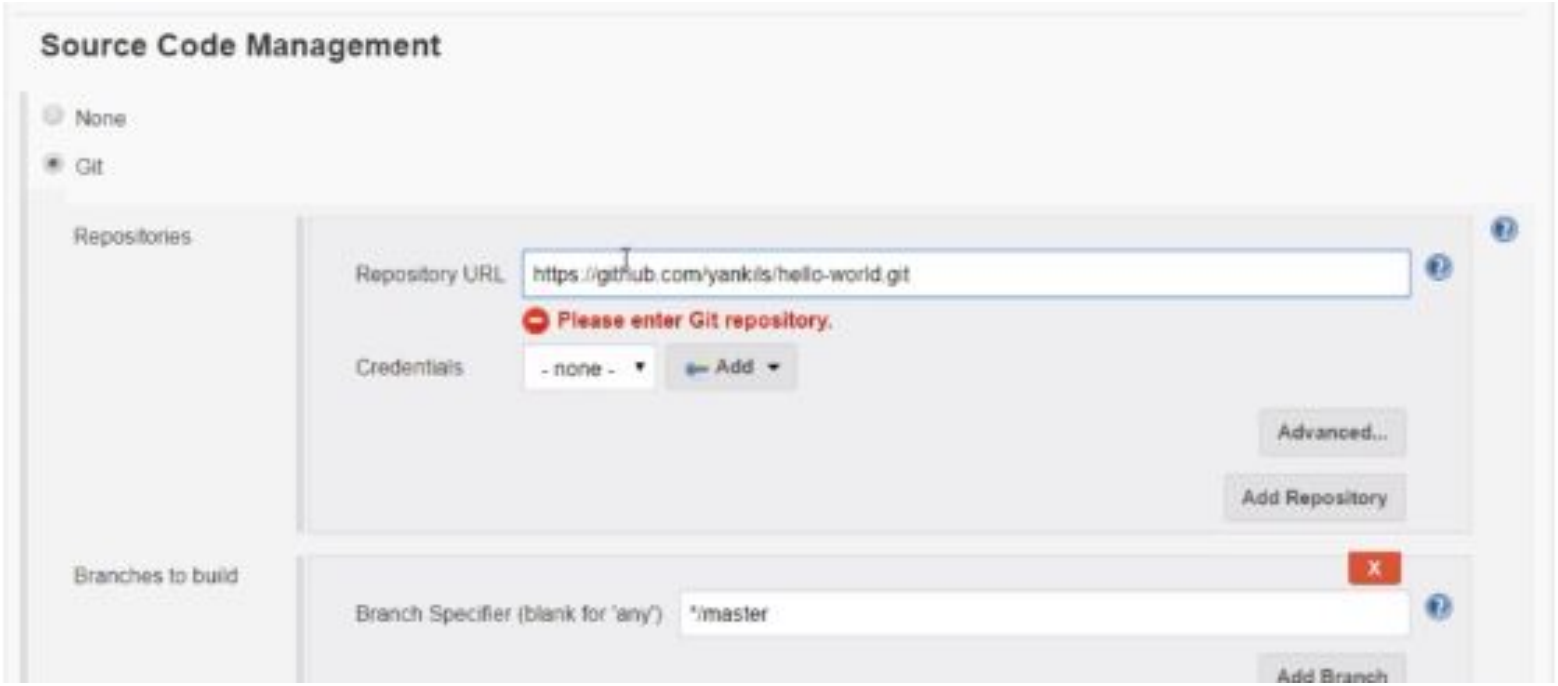
# Integration Phase: Install Maven Packages on Jenkins Server



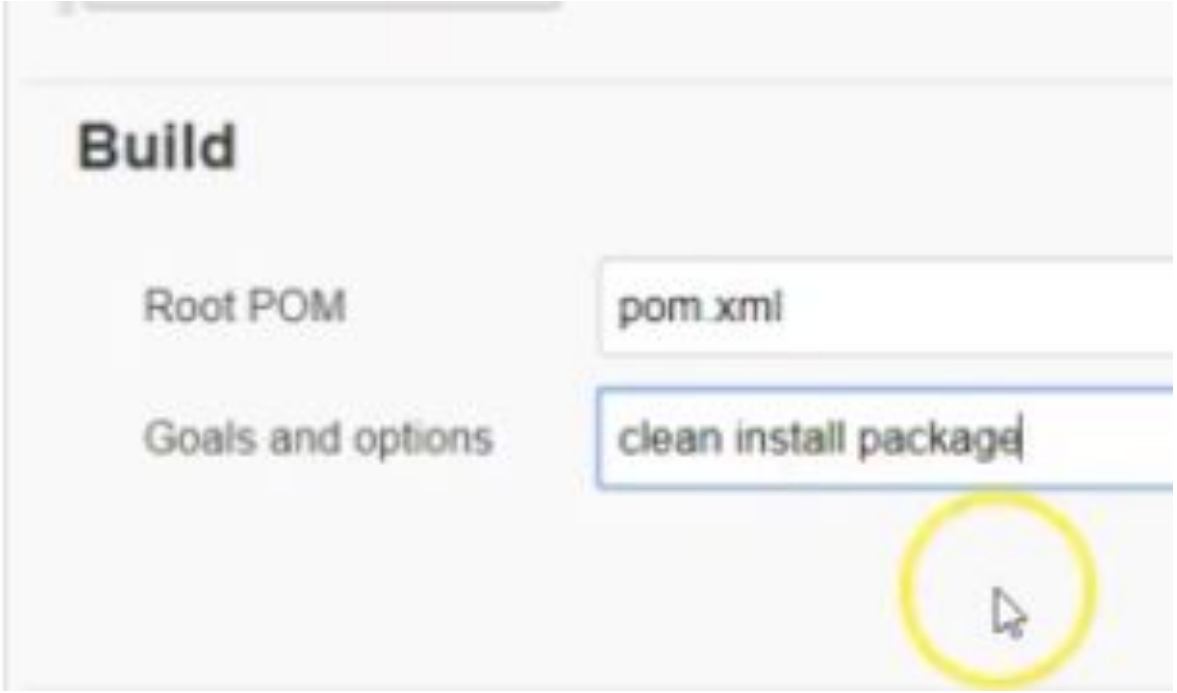
# Integration Phase: Install Maven Packages on Jenkins Server

# Integration Phase: Install Maven Packages on Jenkins Server

# Integration Phase: Install Maven Packages on Jenkins Server

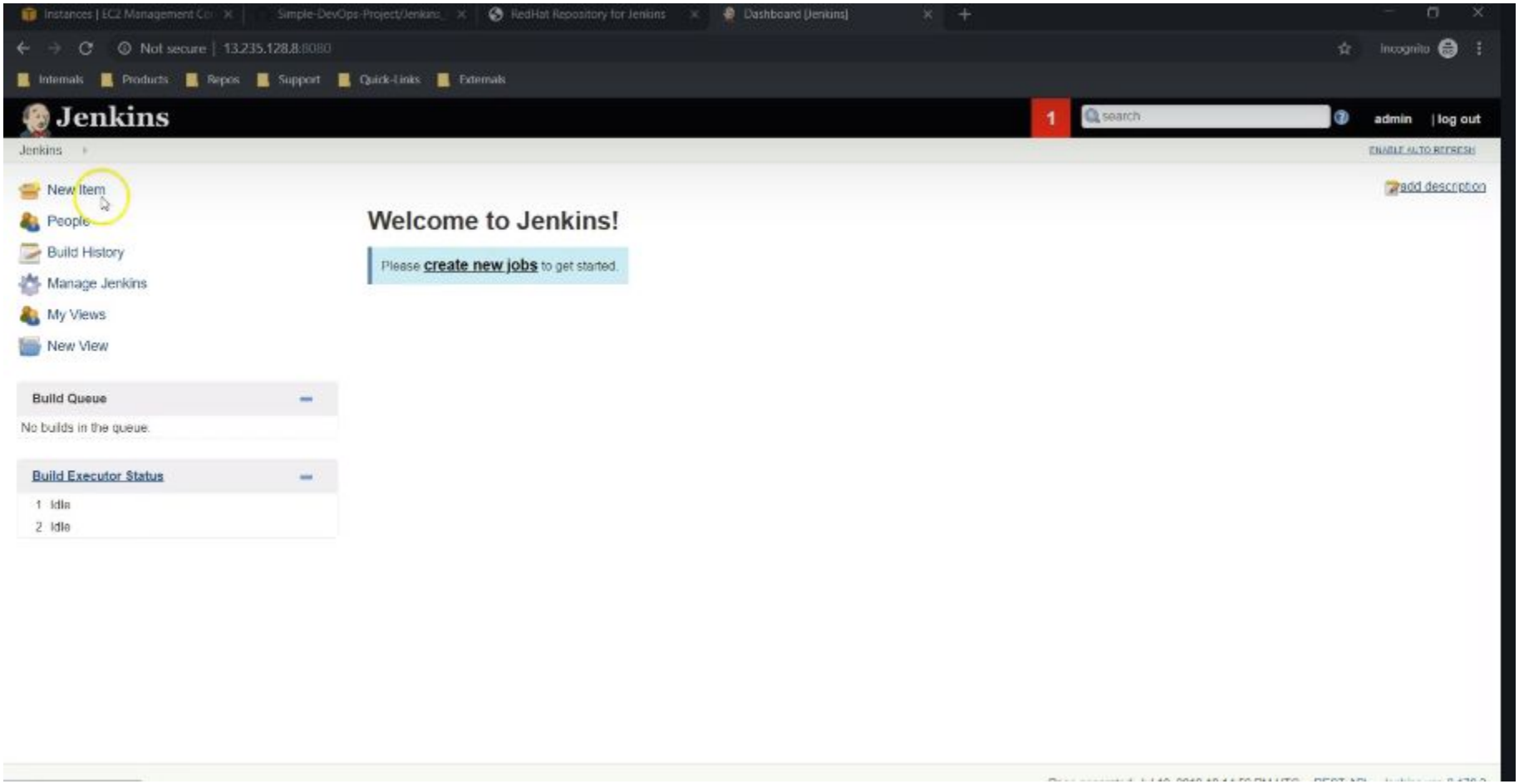


# Integration Phase: Install Maven Packages on Jenkins Server



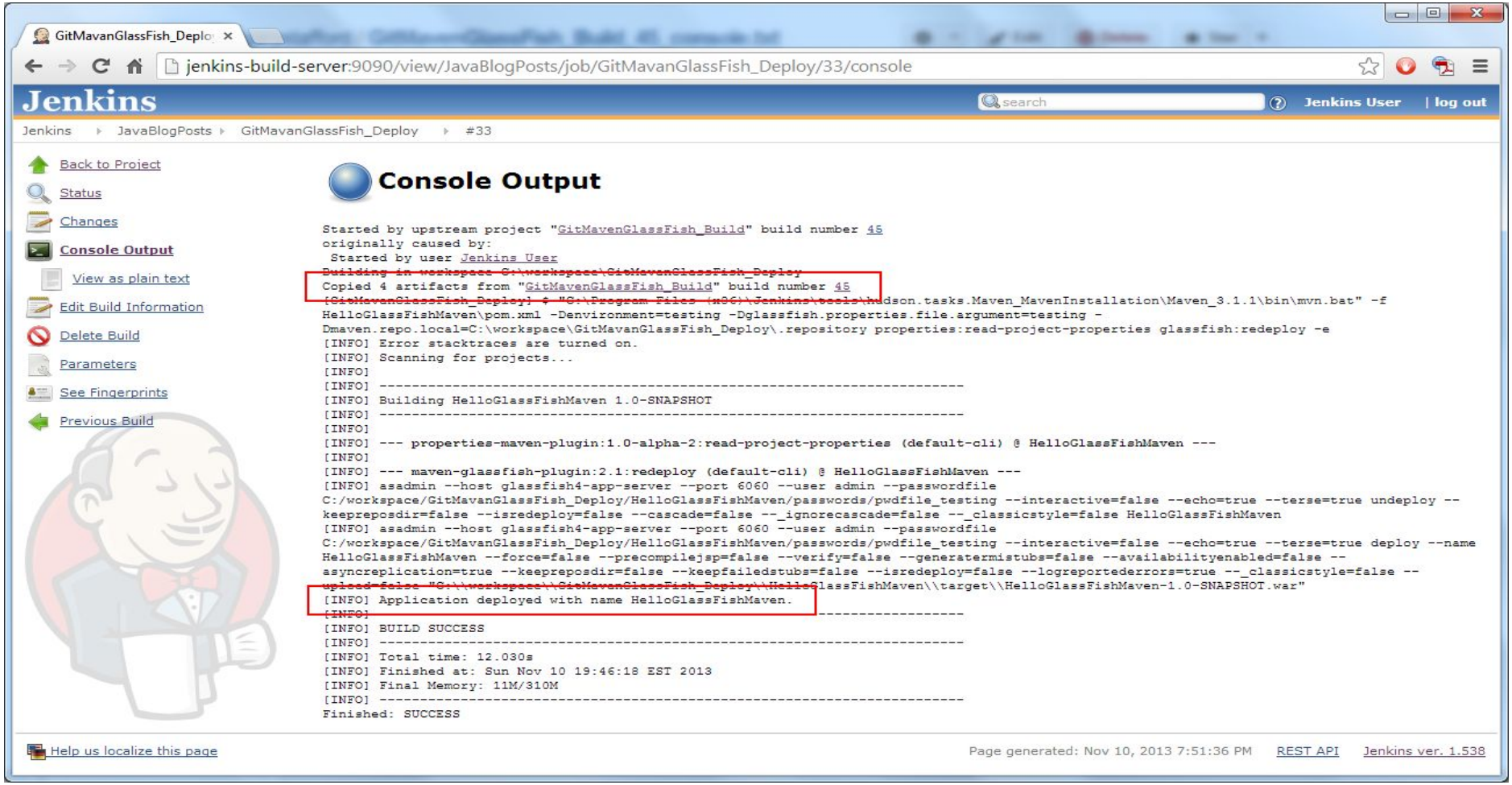
# Testing Phase: Run Jenkins Job

Create “new item”



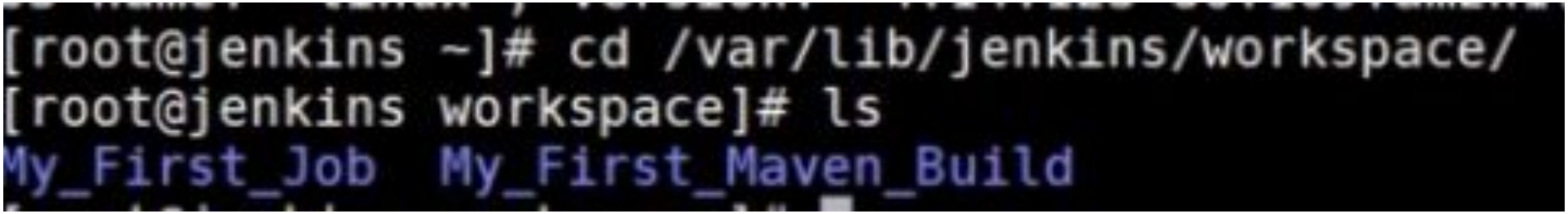
# Monitoring Phase: Console Output

## Build Now> Verify the Folder



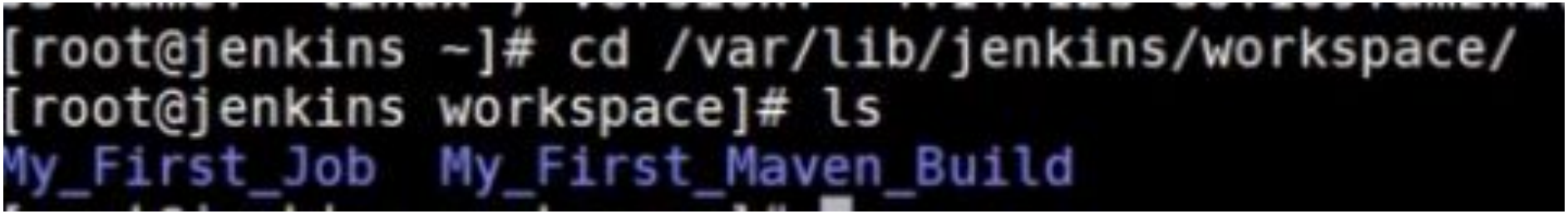
# Feedback Phase: Observe the Built Result

Verify the Folder



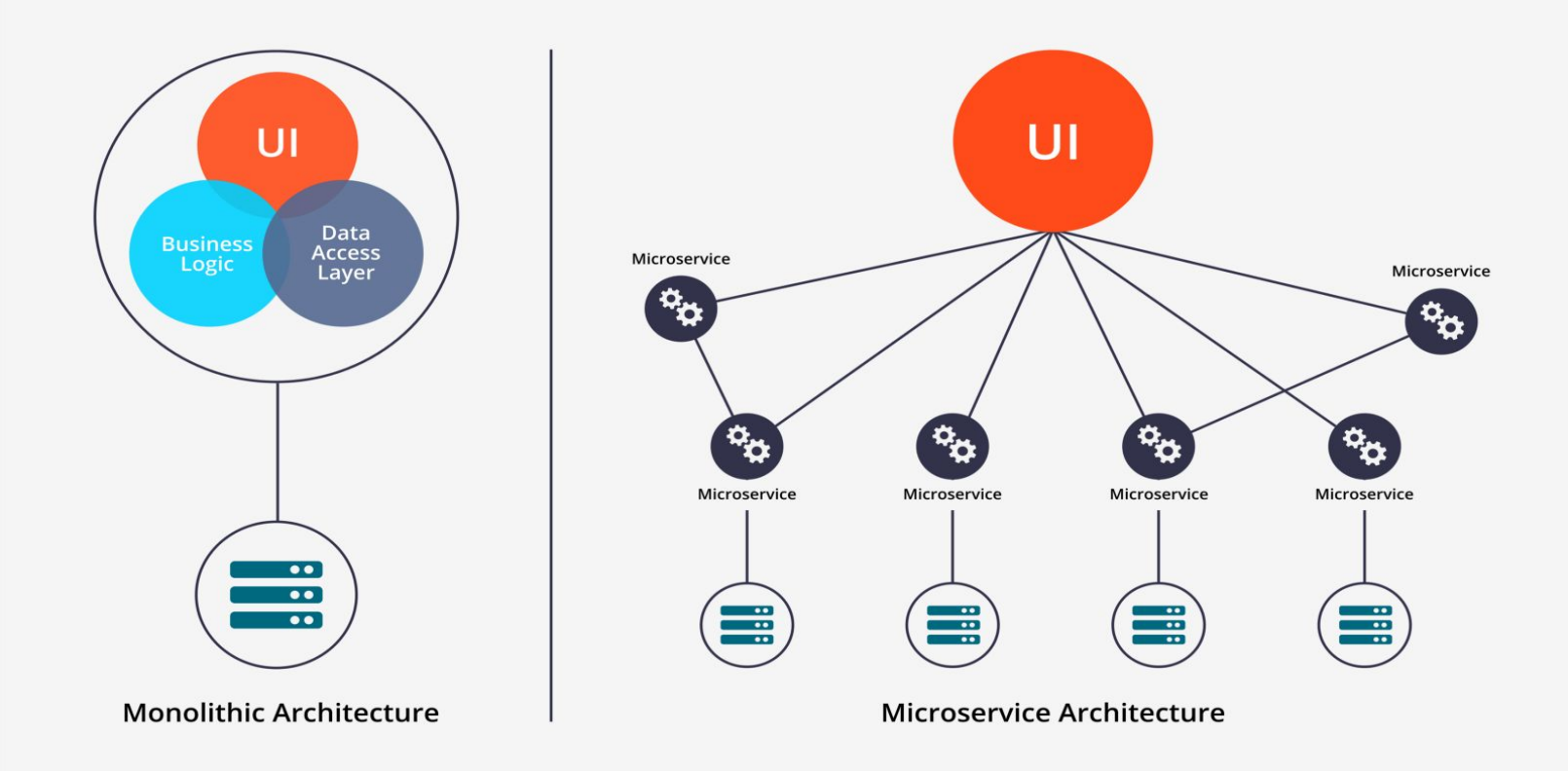
# Conclusion Phase: Conclude the whole pipeline result

Verify the Folder



# Topic 3
# MicroServices

# What is Microservices




Microservice architecture – a variant of the service-oriented architecture structural style – arranges an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight.

It is loosely coupled services which can be developed, deployed, and maintained independently. Each of these services is responsible for discrete task and can communicate with other services through simple APIs to solve a larger complex business problem.

# Key Benefits of Microservices

**Easier to Build and Maintain Apps**

The key principle of microservices is simplicity.

Applications become easier to build and maintain when they're split into a set of smaller, composable fragments.

Managing the code also becomes less painful because each microservice is, in fact, a separate chunk of code.

Services can be implemented using different programming languages, databases and software environments.

This allows each service to be deployed, rebuilt, re-deployed and managed independently.

For example, if a microservice allocates too much memory or puts a heavy load on the processor, it will only affect this service.

Generally speaking, any problem with a microservice will not influence the entire system and the failure of individual microservices can be compensated relatively quickly.

Plus, it allows putting each microservice into production one by one easily.

# Key Benefits of Microservices

**Organized Around Business Capabilities**

Martin Fowler highlights that microservices allow building products instead of projects.

Indeed, microservice architectures invite teams to focus on building business functionality instead of writing glue code.

In other words, development teams are organized around business capabilities and not technologies.

This means that services are adaptable for use in multiple contexts.

The same service can be reused in more than one business process or over different business channels depending on the need.

Each team member is responsible for a particular service which results in building a smart, cross-functional team.

# Key Benefits of Microservices

**Improved Productivity and Speed**

Microservices architecture tackles the problem of productivity and speed by decomposing applications into manageable services that are faster to develop.

Different teams can be working on different components simultaneously without having to wait for one team to finish a chunk of work before starting theirs.

And, as we've mentioned earlier, separate microservices are easier to locate and modify.

This type of architecture is also very handy for speeding up quality assurance since each microservice can be tested individually and you can test the components that have already been developed while the programmers are working on the other ones. Nifty.

# Key Benefits of Microservices

**Flexibility in Using Technologies and Scalability**

You do know by now that each microservice can be written using a different technology.

This simplifies the selection of the most appropriate tech stack for the specific needs of your service.

The microservice architecture allows decoupled services written in different programming languages to peacefully coexist with other fragments.

This is also good news if you're looking to scale your solution in the future.

With microservices, you can add new components to the system painlessly or scale services separately from one another.

# Key Benefits of Microservices

**Autonomous, Cross-functional Teams**

Microservices are a blessing for distributed teams. Carrying out the development of a massive monolith system can be complicated and messy

if you're working with divisions around the globe or extended teams.

Microservices grant the developers more independence to work autonomously and make technical decisions quickly in smaller groups.

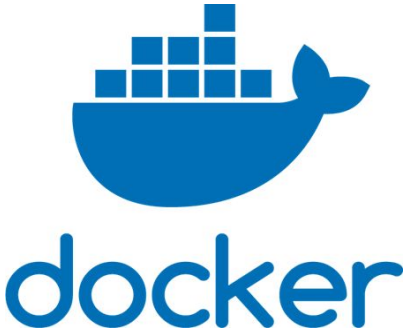
So, if the solution you're developing is expected to be large, be sure to consider the microservice architecture.

# Companies adopting Microservices

Here are list of articles published by companies about their experiences using microservices:

- Comcast Cable
- Uber
- Netflix
- Amazon
- Ebay
- Sound Cloud
- Karma
- Groupon
- Hailo
- Gilt
- Zalando
- Capital One Why Capital One is at Re:Invent and Keynote
- Lending Club
- AutoScout24

# Microservice Components





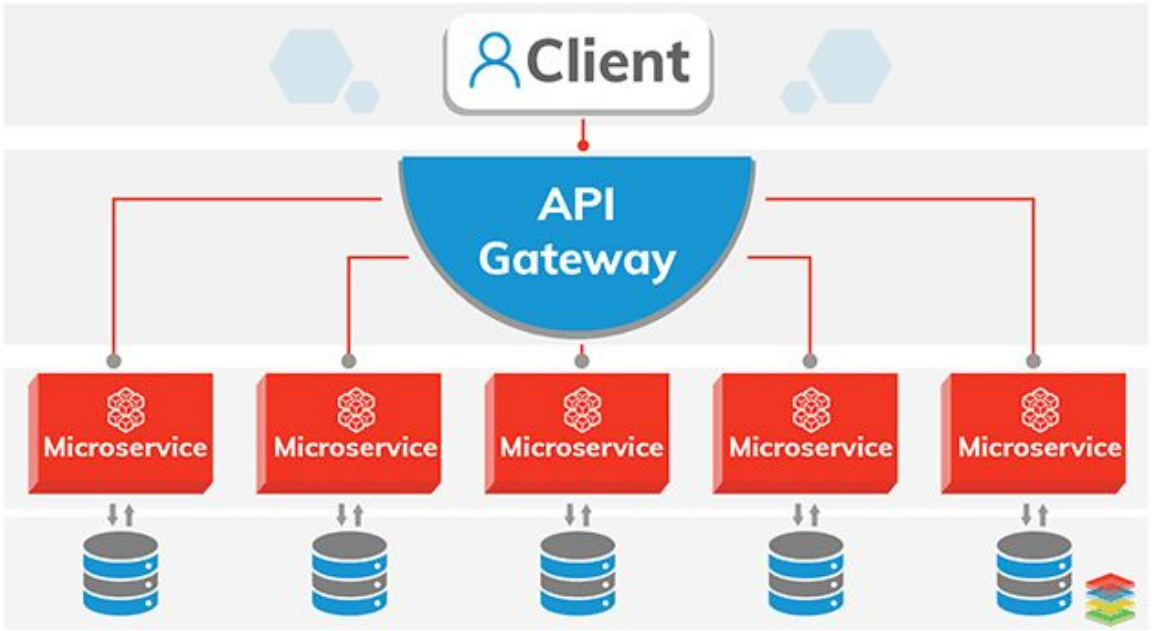
Containers:

You've most likely heard of Docker by now. It's the container standard on the web. It's actually a play on the meaning of the word container. You can think of docker containers and small shipping containers that are running your application in the cloud with an orchestrator moving them around and optimizing them. They also use linux containers directly under the hood.

Orchestration:

Kubernetes is the most popular orchestrator and it works extremely well with Docker. It essentially takes your docker container and looks after them to make sure they're all working in an efficient manner.

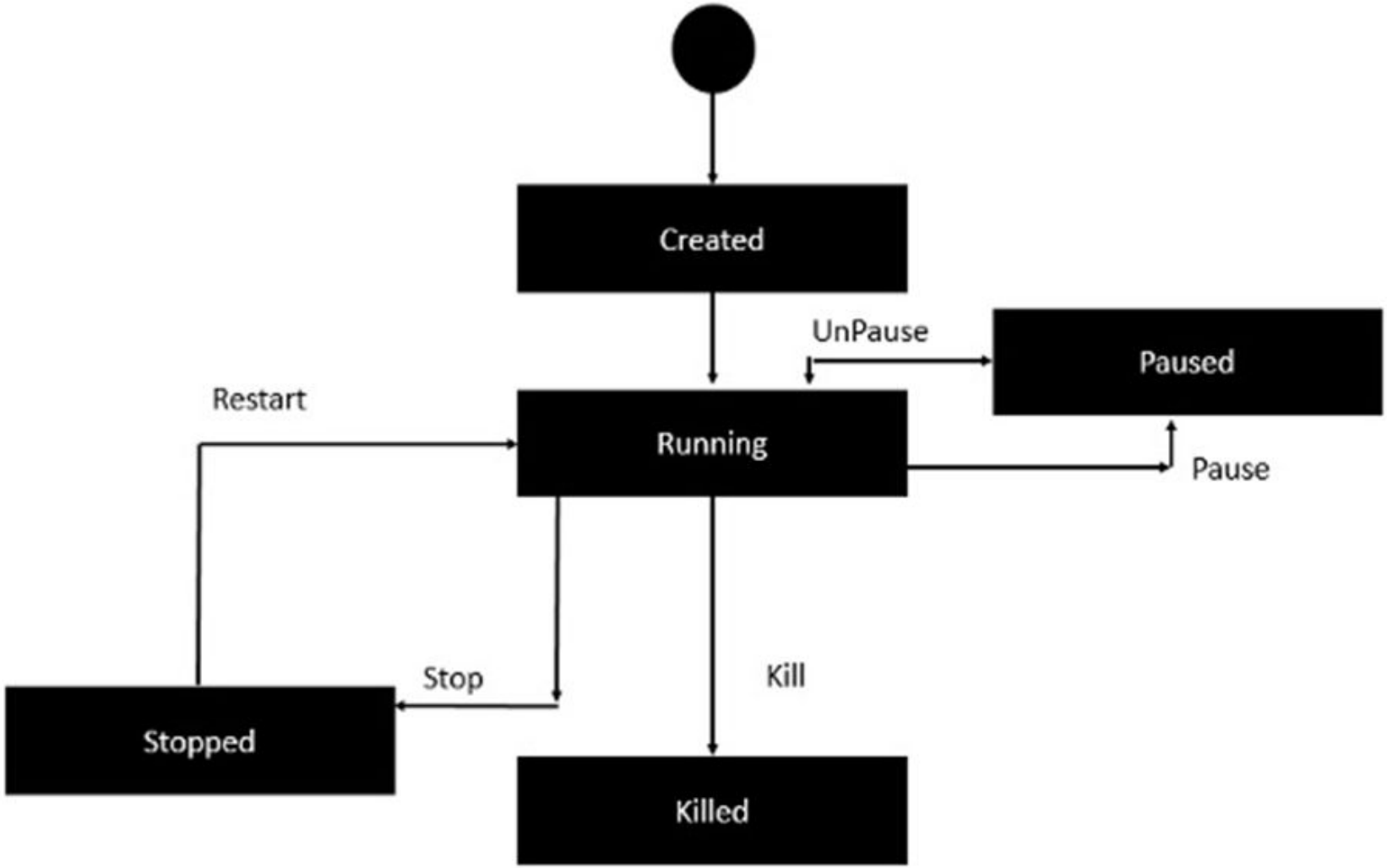
# Components in Microservice Ecosystems




The architecture of Microservices includes

- Containers(Docker)
- Orchestration (Kubernetes)
- Management (Forge)
- Api Gateway / Canary (Ambassador)
- Edge Proxy (Envoy)
- Monitoring (Prometheus)
- Local Testing (Telepresence)

# Container LifeCycle



# Lab: MicroServices Deployment

**Refer to Document:**
https://docs.docker.com/
https://hub.docker.com

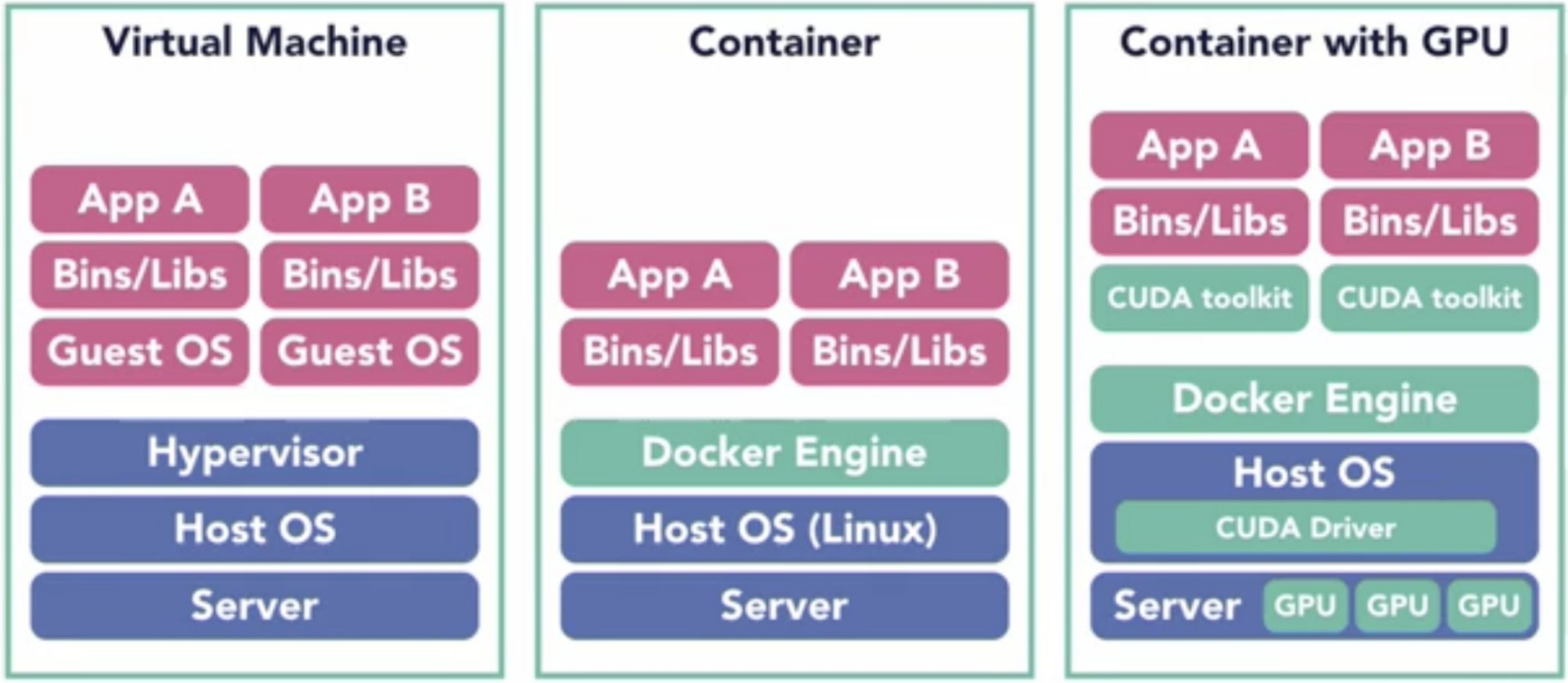
- Open a new terminal and create a container by typing
- docker run --name demo bash -c "echo hello"

- Check the output of the container
- Kill the container
- Remove the container

**Deploy a simple MicroService**
- docker run --name=php1 -p=3003:80 docker/php
- Open the firewall rule(Inbound) at the AWS

# Container vs Virtual Machine

- A container runs natively on Linux and shares the kernel of the host machine with other containers
- A virtual machine (VM) runs a full-blown "guest" operating system with virtual access to host resources through a hypervisor



# Topic 4
# Infrastructure as Code (IaC)

# Infrastructure as Code

- Infrastructure as code, also referred to as IaC, is a type of IT setup wherein developers or operations teams automatically manage and provision the technology stack for an application through software, rather than using a manual process to configure discrete hardware devices and operating systems.
- Infrastructure as code is sometimes referred to as programmable or software-defined infrastructure
- The code-based infrastructure automation process closely resembles software design practices in which developers carefully control code versions, test iterations, and limit deployment until the software is proven and approved for production.

# Infrastructure as Code

Keeping configuration as code in a repository gives many benefits:

- DRY – extracting common parts of the configuration.
- Pull requests – more eyes to check your work.
- Visibility – team members will see what have changed.
- Mobility – migration is not a problem anymore.
- Backup – in case a server has died.
- Testing – you can even write unit tests

# IaC Configuration Management Tools

Using tools like Chef, Puppet, Ansible, Salt, can dynamically configure machines running in their environment.



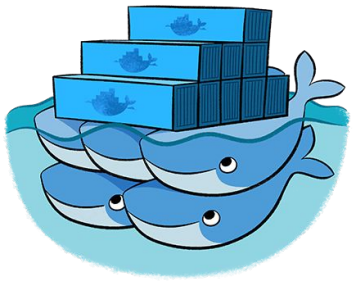

# IaC Service Directory Tools




- etcd, ZooKeeper and Consul are a few common tools to perform service discovery and state tracking across your infrastructure.
- They keep key value information and service health metrics and enable more real time orchestration of your systems

# Container Based Infrastructure

- Docker Swarm, Google's Kubernetes, and Mesos are three popular Docker platforms that do orchestration of containers.
- They allow you to bring in multiple hosts and run your container workload across all of them.
- They handle the deployment, the orchestration and scaling.



Docker

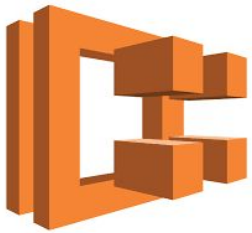
Kubernetes

Apache MESOS™

# Private Container Services


Some private container services, like Rancher, Google's Cloud Platform, or Amazon's ECS take care of running hosts for your containers so you can focus just on your applications

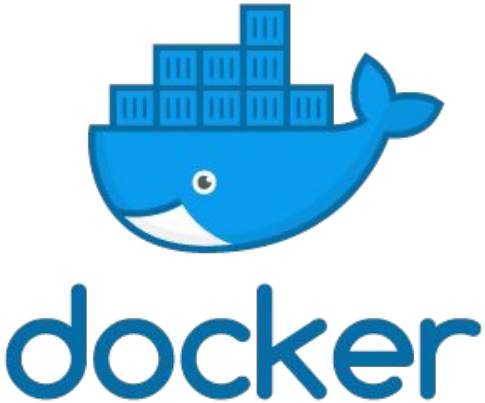
Google Cloud



Amazon ECS

# Examples of IaC

- One example of IaC is to use Ansible, an IT management and configuration tool, can install MySQL server, verify that MySQL is running properly, create a user account and password, set up a new database, and remove unneeded databases.
- An organization may choose to combine infrastructure as code with Docker containers, which abstract the application from the infrastructure at the operating-system level

# Implementation of IaC

You can implement IaC on AWS Cloudformation where templates can be constructed as a set of layers of resources which makes a stack, for example:

```
{
    "Resources":{
        "WebServer":{
            "Type":"AWS::EC2::Instance",
            "Properties":{
                "ImageId":"ami-6869aa05",
                "InstanceType": "t2.micro"
                "KeyName": "mykey"
            }
        }
    }
}
```

# Building IaC using Docker File

## What is Docker File?

- A **Dockerfile** is a text document that contains all the commands a user could call on the command line to assemble an image.
- Using **docker build** to create an automated build that executes several command-line instructions in succession.

# Building IaC using Docker File

## Syntax of Building a Docker File


### Web Application

**Layer 9 : CMD** *Start Pintail.ai website*
**Layer 8 : RUN** *Commands to set up Pintail.ai Website*
**Layer 7 : ADD** *Pintail.ai binaries*
**Layer 6 : FROM** *Node.js - Apline Linux*

### Application Framework

**Layer 5 : RUN** *Commands to set up Node.js*
**Layer 4 : ADD** *Node.js binaries*
**Layer 3 : FROM** *Alpine Linux*

### Operating System

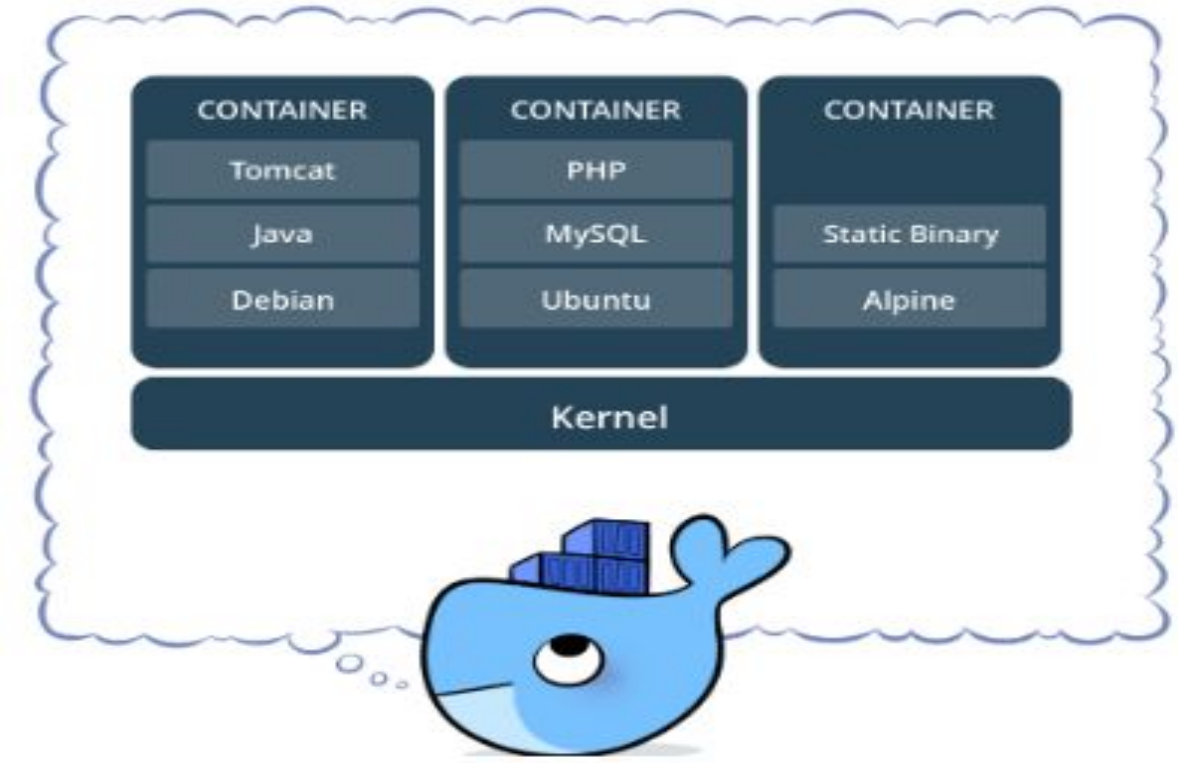
**Layer 2 : RUN** *Commands to set up OS*
**Layer 1 : ADD** *Operating System binaries*
**Layer 0 : FROM** *Scratch*

# Dockerfile Example:

```
FROM node:9.3.0-alpine
RUN npm install -g @angular/cli@1.5.5 \
    && mkdir -p /usr/src/pintail-whoami
ADD . /usr/src/pintail-whoami
RUN npm install && ng build
EXPOSE 80
CMD node server.js $HOSTNAME
```

# Compose Multi-Container Application

- The docker-compose.yml file tells Docker Compose how to build your services and settings your services use like volume on your machine.



# Compose Two Container

- **docker-compose.yml**

## Version

- Current version is 3. So at the top of the file, specify: version: '3'

## Services with builds

- Have a services key in the file. List out services one indent at a time.

## Dependencies

- Use the depends_on key and specify dependencies with a list. Each container dependency is marked by a dash, such as: -backend

# YAML File Example

```yaml
# Filename: docker-compose.yml
version: '3'
services:
  mysqldb:
    image: mysql/mysql-server:5.6
    environment:
      - MYSQL_ALLOW_EMPTY_PASSWORD=true
      - MYSQL_LOG_CONSOLE=true
    ports:
      - "3306:3306"
    volumes:
      - mysqldata:/var/lib/mysql
  redis:
    image: redis:3.0
    ports:
      - "6379:6379"
    volumes:
      - redisdata:/data
volumes:
  mysqldata:
    driver: local
  redisdata:
    driver: local
```

# Docker Compose Commands

- **Docker-compose up:** Turn on Multi Container Service

```
(~/wombat)$ docker-compose up -d
...
  a bunch of docker data pulling your images
...
Creating network "wombat_default" with the default driver
Creating wombat_redis_1    ... done
Creating wombat_mysqldb_1 ... done
```

- **Docker-compose down:** Shut down your services

```
(~/wombat)$ docker-compose down
Stopping wombat_redis_1    ... done
Stopping wombat_mysqldb_1 ... done
Removing wombat_redis_1    ... done
Removing wombat_mysqldb_1 ... done
Removing network wombat_default
```

# Lab: Build Cloud Infra as Code

In this lab, we are going to build an Cloud Infrastructure using Code:

- Build a Dockerfile
  #Dockerfile
  FROM php:latest
  COPY ./index.php ./
  EXPOSE 80
  CMD ["php", "-S", "0.0.0.0:80"]

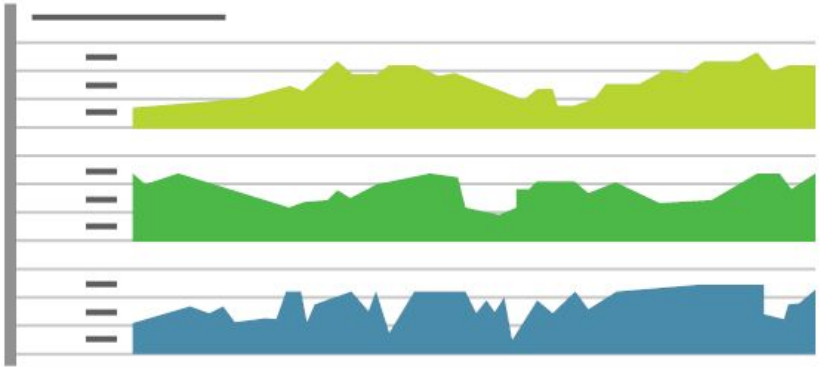
- Build an Website file:
  #Index.php
  <html>
  <?php echo "Hello world from a php container" ?>
  </html>

- Build Docker Image
  docker build -t docker/php:1.0 .

- Build a MicroService and Network Infrastructure
  docker run --name=php1 -p=3003:80 docker/php:1.0

# Topic 5
# Monitoring and Logging

# What is Monitoring



One goal of monitoring is to achieve high availability by minimizing time to detect and time to mitigate (TTD, TTM).

In other words, as soon as performance and other issues arise, rich diagnostic data about the issues are fed back to development teams via automated monitoring. That's TTD.

DevOps teams act on the information to mitigate the issues as quickly as possible so that users are no longer affected. That's TTM. Resolution times are measured, and teams work to improve over time. After mitigation, teams work on how to remediate problems at root cause so that they do not recur. That time is measured as TTR.

# What is Log Management


- Log management is the process of handling log events generated by all software applications and infrastructure on which they run.

- It involves log collection, aggregation, parsing, storage, analysis, search, archiving, and disposal, with the ultimate goal of using the data for troubleshooting and gaining business insights, while also ensuring the compliance and security of applications and infrastructure.

- Logs are typically recorded in one or more log files. Log management allows you to gather the data in one place and look at it as part of a whole instead of separate entities. As such, you can analyze the collected log data, identify issues and patterns so that you can paint a clear and visual picture of how all your systems perform at any given moment.

# The 3 Pillars of Observability

Businesses are in the midst of a shift toward adopting these principles—the three pillars of observability—and using a three-pronged method of monitoring that takes data from several vantage points to depict a more accurate and consumable view of overall health and stability:

1. **External Monitoring:** An example of this is health checks run against your internal and external applications and websites to see the "digital experience" of your users.
2. **Metrics and Distributed Tracing:** This enables you to trace communications between applications distributed across systems/containers to identify errors and exceptions from your applications and resolve latency quickly.
3. **Events and Logs:** Data, which helps provide contextual information about events, enables you to identify issues in the code when combined with information from the first two pillars.

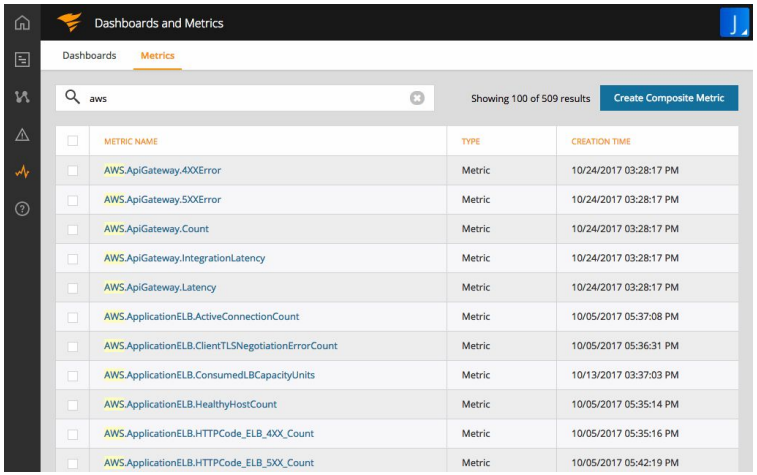
# External Monitoring

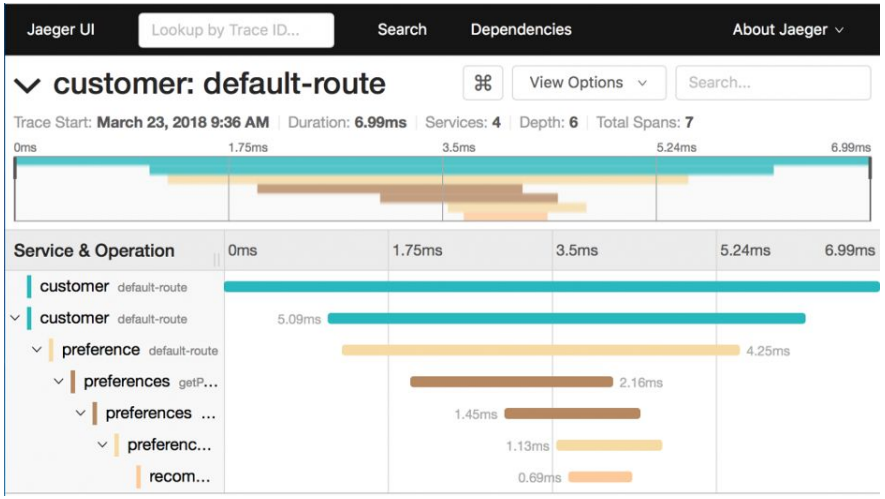
In a general sense, monitoring means regularly checking on something to find out what is happening. It's a way to gather information. In a large network, it means observing and recording data from a wide variety of devices and applications.

# Metrics and Distributed Tracing

A distributed trace is defined as a collection of spans. A span is the smallest unit in a trace and represents a piece of the workflow in a distributed landscape. It can be an HTTP request, call to a database, or execution of a message from a queue.

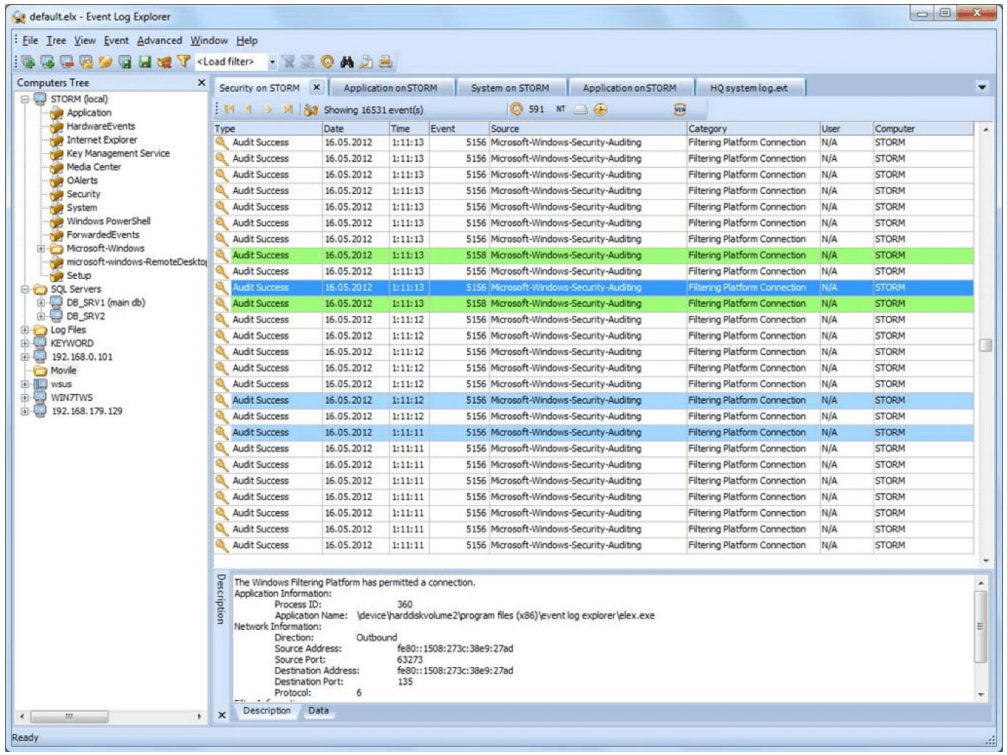
Distributed tracing is a critical component of observability in connected systems and focuses on performance monitoring and troubleshooting. You can use it to know how long a request took to process and identify a slow service in a microservice environment. Or you can track latency issues and gain valuable insights by tracing your call amidst the dependent components in the entire application stack.



# Events and Logs:

**Event** logging provides a standard, centralized way for applications (and the operating system) to record important software and hardware **events**. The **event** logging service records **events** from various sources and stores them in a single collection called an **event log**.



# Best Practices

Given the increasing pace of application development required to support the needs of the business, the spread of containerized applications and virtual environments and the emergence of new disciplines for observability (due to the lowered costs and increased reliability), log and event management has become a critical aspect for all involved in building, supporting and even using mission-critical applications.

It's moving to a defined area where logs should be compartmentalized, and access control should be defined:

1. Compartmentalizing Logs: Critical as a best practice so we can differentiate development, staging and production environments and segregate logs based on practical grouping. Using a log aggregation service provides you with a consistent experience and a common set of capabilities across all your environments.

# Best Practices

2. Access Control: Another best practice that should be implemented more broadly; for example, developers don't need access to all of your production logs if approximately 95 percent of their work happens prior to reaching production environment. In other words, even though tech pros are working to "de-silo" log management, we still want to silo access control and manage visibility. This can also help to ensure personally identifiable data (PII) isn't exposed to anyone it shouldn't be.

# Logging and monitoring Tools

Fluentd Fluentd is an open source data collector for unified logging layer, sponsored by Treasure Data. It structures data as JSON to unify all facets of processing log data: collecting, filtering, buffering, and outputting logs across multiple sources and destinations. Fluentd on GitHub

Heapster Heapster is a container cluster monitoring and performance analysis tool in Kubernetes. It supports Kubernetes and CoreOS natively and can be adapted to run on OpenShift. It also supports a pluggable storage backend: InfluxDB with Grafana, Google Cloud Monitoring, Google Cloud Logging, Hawkular, Riemann and Kafka. Heapster on GitHub

Logstash Logstash is Elastic's open source data pipeline to help process logs and other event data from a variety of systems. Its plugins can connect to a variety of sources and stream data at scale to a central analytics system. LogStash on GitHub

# Logging and monitoring Tools

Prometheus Prometheus is an open source systems monitoring and alerting toolkit, originally built at SoundCloud and now a Cloud-Native Computing Foundation project at The Linux Foundation. It fits both machine-centric and microservices architectures and supports multi-dimensional data collection and querying. Prometheus on GitHub

Weave Scope Weave Scope is Weaveworks' open source tool to monitor distributed applications and their containers in real time. It integrates with Kubernetes and AWS ECS. Weave Scope on GitHub

# Lab: Using Monitoring and logging Tools

Access to the LogStach and Kibana Dashboard demo for log collected and Data Visualisation

https://demo.elastic.co/app/kibana#/dashboard/welcome_dashboard

# Topic 6
# Communication and Collaboration

# Devops Collaboration and Communication

- In an industry where distributed offices full of crucial roles are the norm — and one where even departments within the same buildings tend to distrust one another — any improvement in the way people interact is bound to have some positive results, especially when so many moving parts need to work together for a product to come in on time and under budget.

# Benefit of Communication and Collaboration Tools in Devops

Following are some of the biggest problems that can occur if DevOps teams don't collaborate effectively:

- Redundancy of work due to communication silos that prevent the sharing of ideas.
- Unproductive handoffs among software development and operations teams.
- Confusion among developers about who is responsible for which tasks.
- Negative impact on product stability and release cycles.
- Inefficiencies in operations which will result in late deliverables.
- A massive amount of emails, end-user dissatisfaction and revenue loss.

Undoubtedly, DevOps teams need to streamline the entire lifecycle of software development in a better way through effective collaboration.

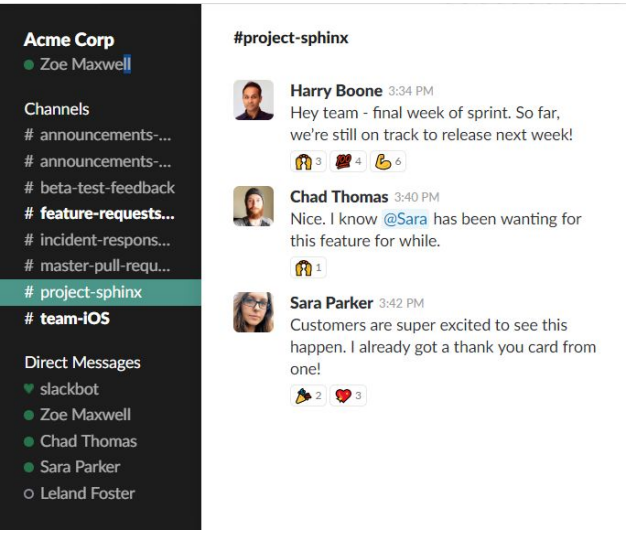
# Communication and Collaboration Tools

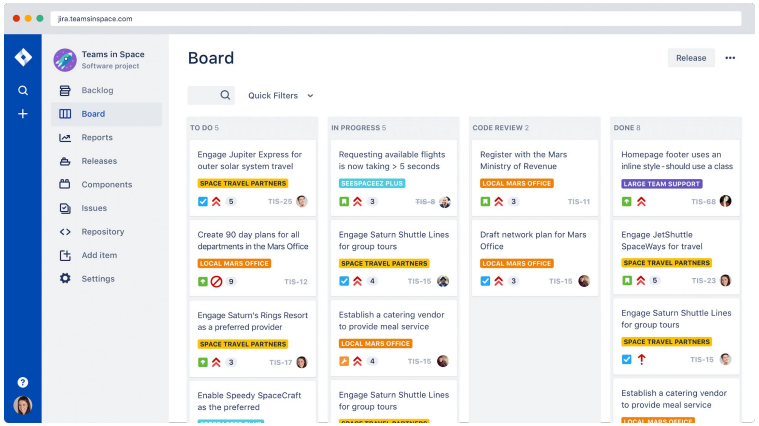
Never has it been more important to be able to collaborate and communicate remotely. The coronavirus outbreak has completely disrupted how we work, and those organisations without the proper tools will suffer. Luckily, there are more tools than ever before to choose from.

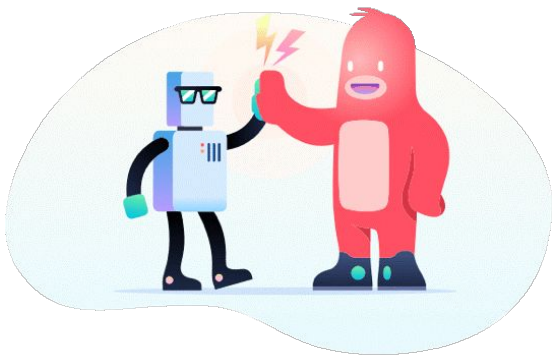
# Communication and Collaboration Tools

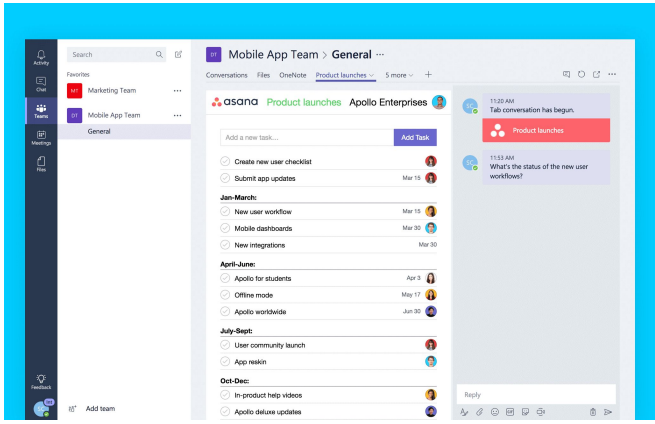
List of Tools for Communication and Collaborations:

1. Slack
2. Jira
3. Trello
4. Asana
5. Glip

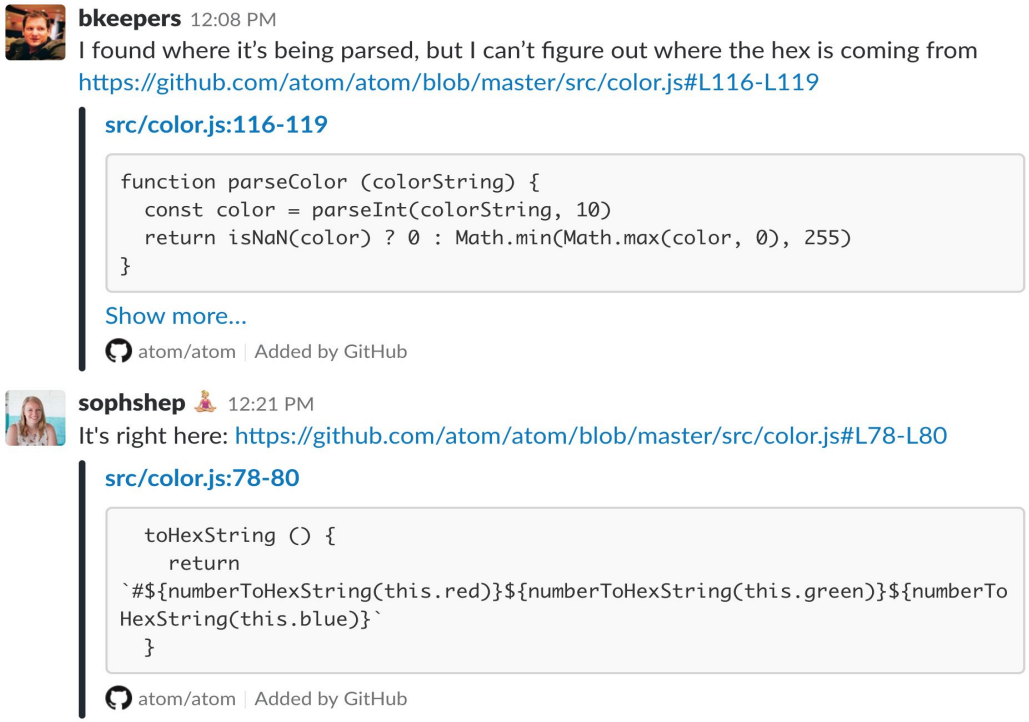




# DevOps Collaboration and Communication




- Team play is so important to DevOps that you could really sum up most of the methodology's goals for improvement with two Cs: collaboration and communication. While it takes more than that to truly become a DevOps workplace, any company that has committed to those two concepts is well on its way.

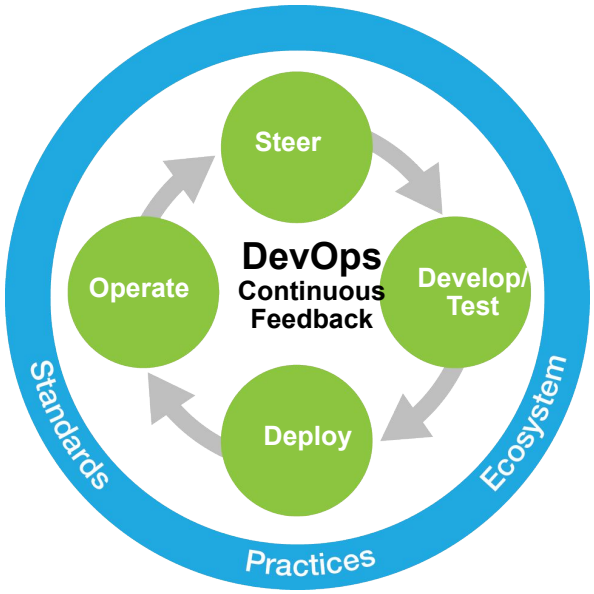
# Lab: GIT and Slack Integration

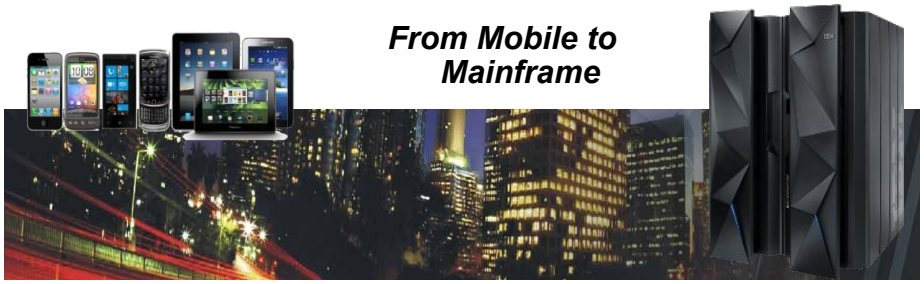
- The GitHub integration for Slack gives you and your teams full visibility into your GitHub projects right in Slack channels, where you can generate ideas, triage issues and collaborate with other teams to move projects forward. This integration is an open source project, built and maintained by GitHub.

Access to the lab https://github.com/integrations/slack and understand on how the collaboration tools can communicate with CI/CD

# Summary

- There are challenges to delivering software-driven innovation
- Disruptive technologies are driving greater need to innovate
- DevOps is critical to your success
- DevOps is just as relevant, if not more so, for the Mainframe as it is for mobile, cloud, and distributed platforms



# References


- Cauldwell, P., Code Leader: Using People, Tools, and Processes to Build Successful Software, Wrox, 2008.
- Duvall, P., Matyas, S., and Glover, A., Continuous Integration: Improving Software Quality and Reducing Risk, Addison-Wesley, 2007.
- Hatcher, E., and Loughran, S., Java Development with Ant, Manning, 2002.
- Smart, J.F., Java Power Tools, O'Reilly, 2008.
- Beton, R., *Software for Continuous Integration - Cruise Control & Hudson Compared*, 2009. http://www.bigbeeconsultants.co.uk/cruise-control-vs-hudson
- Duvall, P., *Automation for the people: Continuous feedback*, 2006.
- Duvall, P., *Automation for the people: Continuous Integration anti-patterns*, 2007. http://www.ibm.com/developerworks/java/library/j-ap11297/
- Graham, M., *Software Defect Prevention Using Orthogonal Defect Classification*, 2005.
- Fleischer, G., *Continuous Integration: What Companies Expect and Solutions Provide*, 2009. http://appl.fontysvenlo.org/results/2008/GF/continuous_integration.pdf
- Fowler, M., Continuous Integration, 2006.http://martinfowler.com/articles/continuousIntegration.html
- Minick, E. and Fredrick, J., Enterprise Continuous Integration Maturity Model, 2009. http://www.anthillpro.com/html/resources/white-papers/view.html?id=1195436
- Muhonen, M., Software Process Improvement: Continuous Integration and Testing for Web Application Development, 2009. http://tutkielmat.uta.fi/haekokoversio.php?id=19979
- Zawadzki, M., Drawing the Line: Continuous Integration and Build Management, 2007. http://www.anthillpro.com/html/resources/white-papers/view.html?id=1114131
- Zawadzki, M., Continuous Integration and Build Management Server Evaluation Guide, 2008. http://www.anthillpro.com/html/resources/white-papers/view.html?id=1132617

# Feedback

https://goo.gl/R2eumq

TINFOTECH

# CERTIFICATE

## *of* ACCOMPLISHMENT

You will receive a digital certificate in your email
after the completion of the class
If you did not receive the digital certificate,
please send your request to
enquiry@tertiaryinfotech.com

# Thank You!

Man Guo Chang
gc.man.sg@gmail.com