DEVOPS – CI/CD

23-11-2020 to 27-11-2020

Venkatesh Reddy Madduri && ShankarPrasad

Prerequisites

- Familiarity with Linux Commands
- Basic knowledge on networking concepts
- Target Audience
  - System administrators
  - Software developers
  - Anyone who wants learn !!!

About You

- Please tell me about yourself:

  - Your Name
  - Your background
  - What is the purpose of this course?
  - Where and how you will be using this knowledge?
  - What do you currently know about DevOps?

About Us

- **Your Trainers : Venkatesh Reddy Madduri and ShankarPrasad**
- **Experience : 14.5 years and 15 years**

Course Organization

- Hours : 10:00 to 18:00 hrs
- Breaks: As and when required !!

We would be using RedHat 8 as our primary OS and hope every one was able to access their VMs !

Credits

- We would like to thank FIS and Training department for giving us opportunity to learn and share our knowledge.
- Some of the images/materials may be borrowed from the internet and may not be owned by us. We would like to give the original content authors their due credit.
- We extend our gratitude to the original content authors of those images/contents.

Agenda

- Day 1: DevOps Overview
  Git & GitHub
- Day 2: Continuous Integration using Jenkins
  Introduction to Docker
- Day 3: Kubernetes
- Day 4: Ansible
- Day 5: Continuous Monitoring using Nagios
- Day 5: DevOps Project

What to expect from this training ?

- Theory for the first 2 hours on the first day
- Later on heavy hands-on
- Some theoretical section will obviously be there to explain architecture/concepts

Introduction

- For many years, we've been finding better ways to build systems
- Learning from what has come before - adopting new technologies
- Observing how a new wave of technology companies operate in different
- Ways to create IT systems that help make both their customers and their own developers happier, eg:
  - Golang
  - Docker
  - Kubernetes
  - Nginx etc
- Few terminologies - these weren't invented or described before the fact
  - Domain-driven design
  - Continuous delivery
  - On-demand virtualization
  - Infrastructure automation
  - Small autonomous teams
  - Systems at scale

Introduction

- Continuous delivery helped to move software into production more effectively & efficiently
- Understanding how the Web works led us to develop better ways of having machines talk to each other.
- Virtualization allowed us to provision, resize our machines at will (Vagrant, AWS and Azure etc.)
- Infrastructure automation - handle these machines at scale(Terraform, AWS CloudFormation, OpsWorks)
- Successful organizations like Amazon and Google belief the view of small teams owning the full lifecycle of their services
- Netflix has shared with us ways of building antifragile systems at a scale that would have been hard to comprehend just 10 years ago
- Microservices have emerged from this world as a trend, or a pattern, from real-world use and they exist only because of all that has gone before - Our Mistakes embracing fine-grained, microservice architectures helps
  - deliver software faster
  - embrace newer technologies
- Microservices give us significantly
  - more freedom to react and make different decisions
  - allowing us to respond faster to the change that impacts all of us

Monolith Systems

- Monolithic System Architecture - Single Tiered Architecture
- Functionally distinguishable aspects like data input and output, data processing, error handling, and the user interface are all interwoven
- User interface & data access code are combined into a single program from a single platform
- Monolithic applications are
  - self-contained
  - independent from other computing applications

Pros:

- Everything is running through the same app
- Easy to hook up components to those cross-cutting concerns
- Performance advantages, - shared-memory access is faster than inter-process communication
- Over time - better architectural patterns have been introduced to the monolithic model to add significant flexibility into the architecture

Monolith Systems Cons

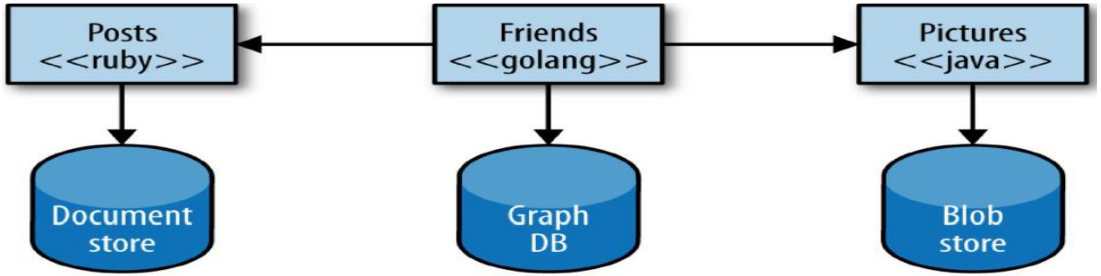
- Majority of logic is deployed and runs in a centralized server - SPOF
- Typically large & built by big/multiple teams with a large code base
- More effort for orchestration between teams for any change/deployment
- Difficult for developers to become familiar with the code
- Especially a new member to the team
- Causes developers to be less productive
- Discourages attempts to make changes
- Update/small change to a component, entire application to be redeployed
- Large effort is required to run regression test against the untouched parts
- Dependency on one technology stack
- Less flexibility for migration to other tools etc.

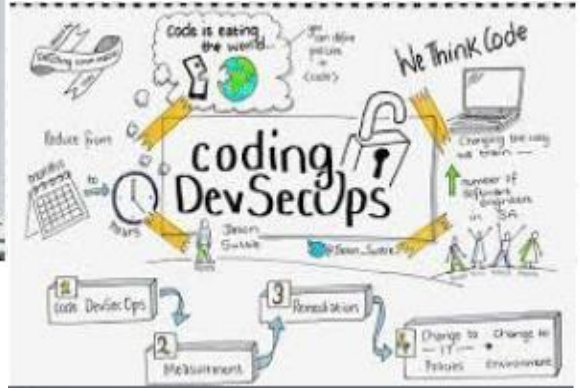
The Solution

- Migration to microservice architecture from monolithic applications:
- Smaller code base
- Easy to develop
- Adding new features quickly
- Fast shipping
- Easy to scale certain parts of application

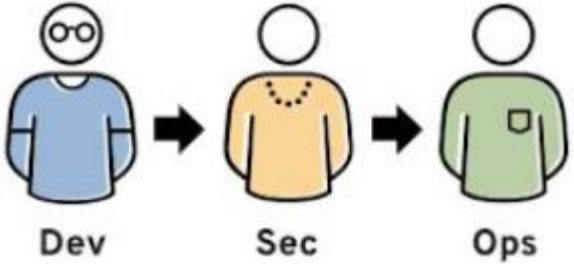
The Solution

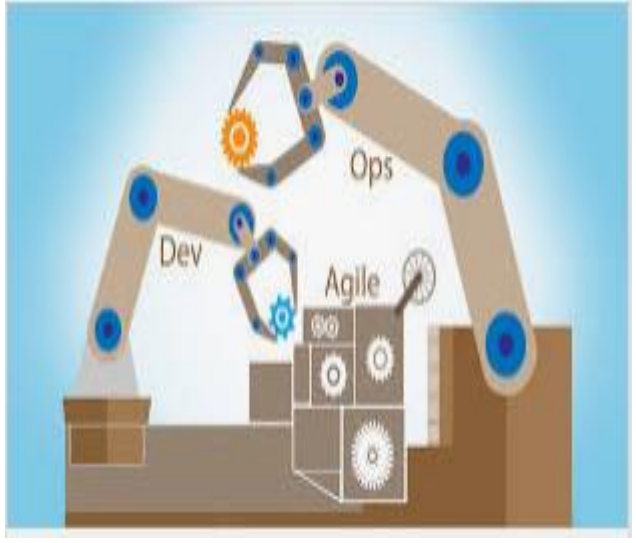
- Microservices also usage of latest tools & technologies to achieve:
  - flexibility in design/architecture
  - Better source code management
  - Achieve High availability of services
  - Ease of Deployments
  - Automation to handle the scale of servers/services
  - Communication between services/components happen via REST API's or MQ solutions.
- Cohesion : related behavior together
- Coupling : change in one product/service should not impact others





DEVSECOPS | SECURITY AS CODE





DEV SOFTWARE
SEC SAFER
OPS SOONER

FIS

DevOps

- We can’t think about microservices without DevOps!!
- Microservices cause an explosion of moving parts and not a good idea to attempt to implement microservices without serious deployment and monitoring automation.
- We should be able to click a button and get our app deployed. In fact, we should not even do anything!!
- Committing code should get our app deployed through the commit hooks that trigger the delivery pipelines in at least development
- We still need some manual checks and balances for deploying into production.

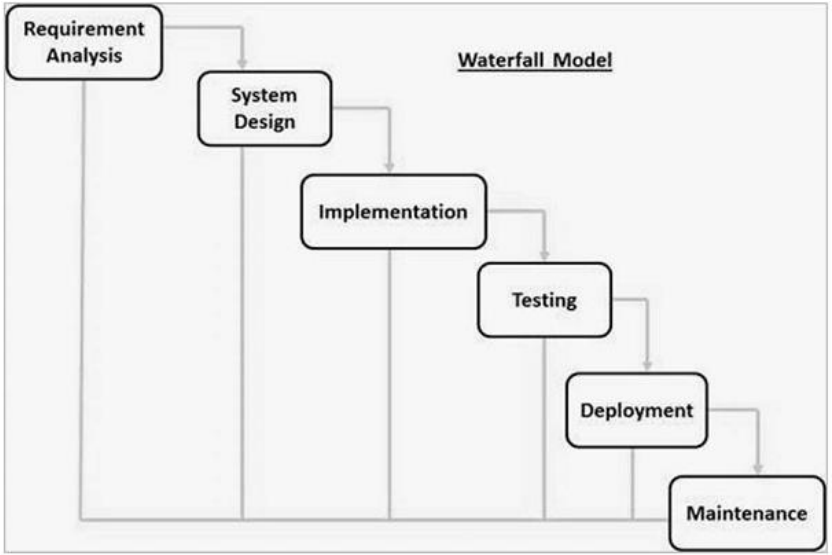
DevOps Overview

- The DevOps is the combination of two words, one is **Development** and other is **Operations.**
- DevOps is a culture which promotes collaboration between Development and Operations team to deploy code to production faster in an automated & repeatable way.
- DevOps helps to increases an organization’s speed to deliver applications and services. It allows organizations to serve their customers better and compete more strongly in the market which leads following questions.

  - Why DevOps?
  - What is DevOps?
  - DevOps Stages
  - DevOps Tools

Waterfall Model

- Waterfall Model – was the first Process Model. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.



Advantages Waterfall Model

- Simple and easy to understand and use.
- Easy to manage as each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Process and results are well documented.

Dis-advantages of Waterfall Model

- No working software is produced until late during the life cycle.
- High amounts of risk uncertainty.
- Not a good model for complex and object-oriented projects.
- Cannot accommodate changing requirements.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

What is Agile Methodology ?

The meaning of Agile is swift or versatile. “Agile process model” refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning

In the Agile Methodology each project is broken up into several ‘Iterations’

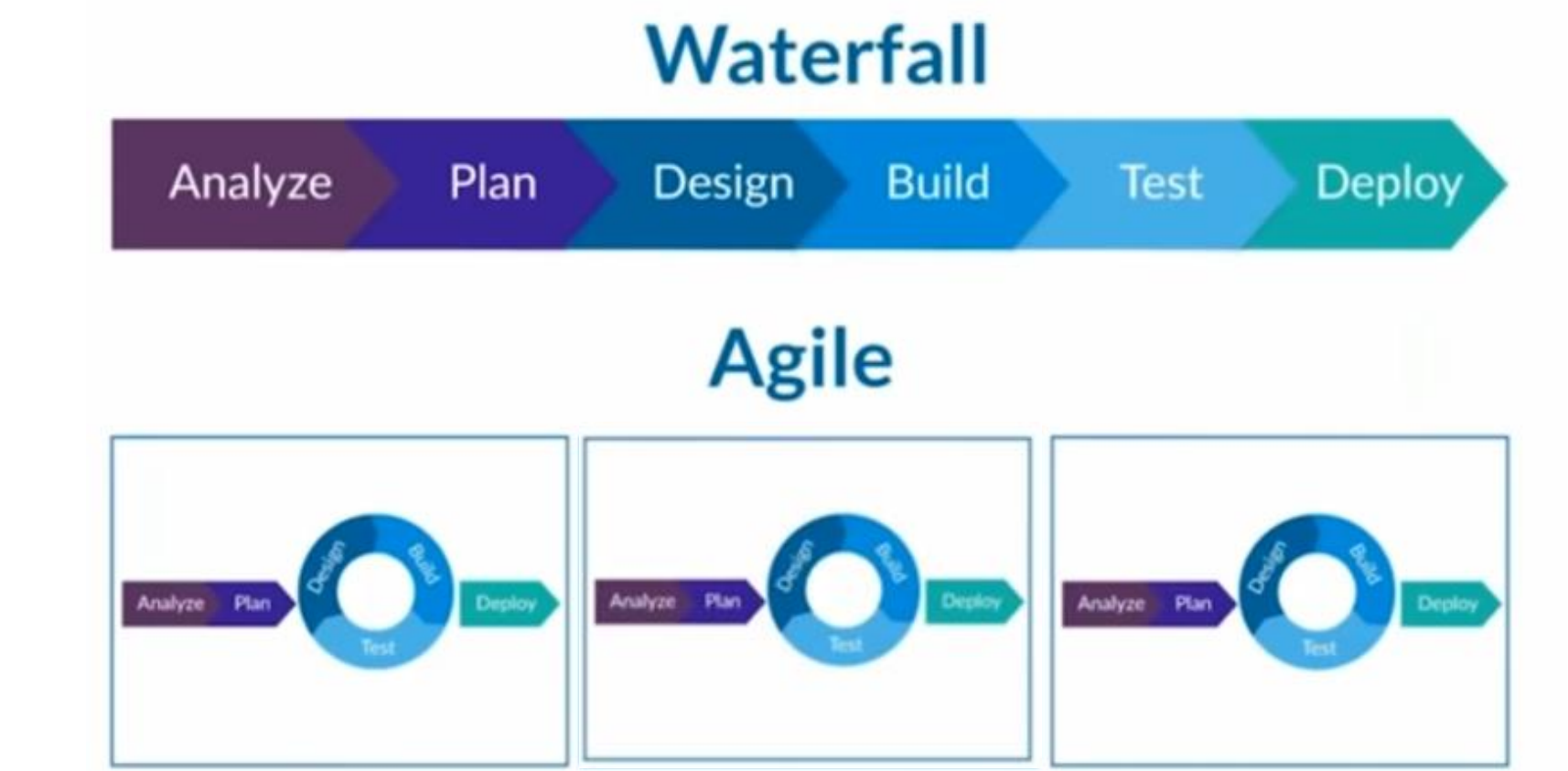
All Iterations should be of the same time duration (between 2 to 8 weeks)

At the end of each iteration, a working product should be delivered

Advantages Agile Methodology

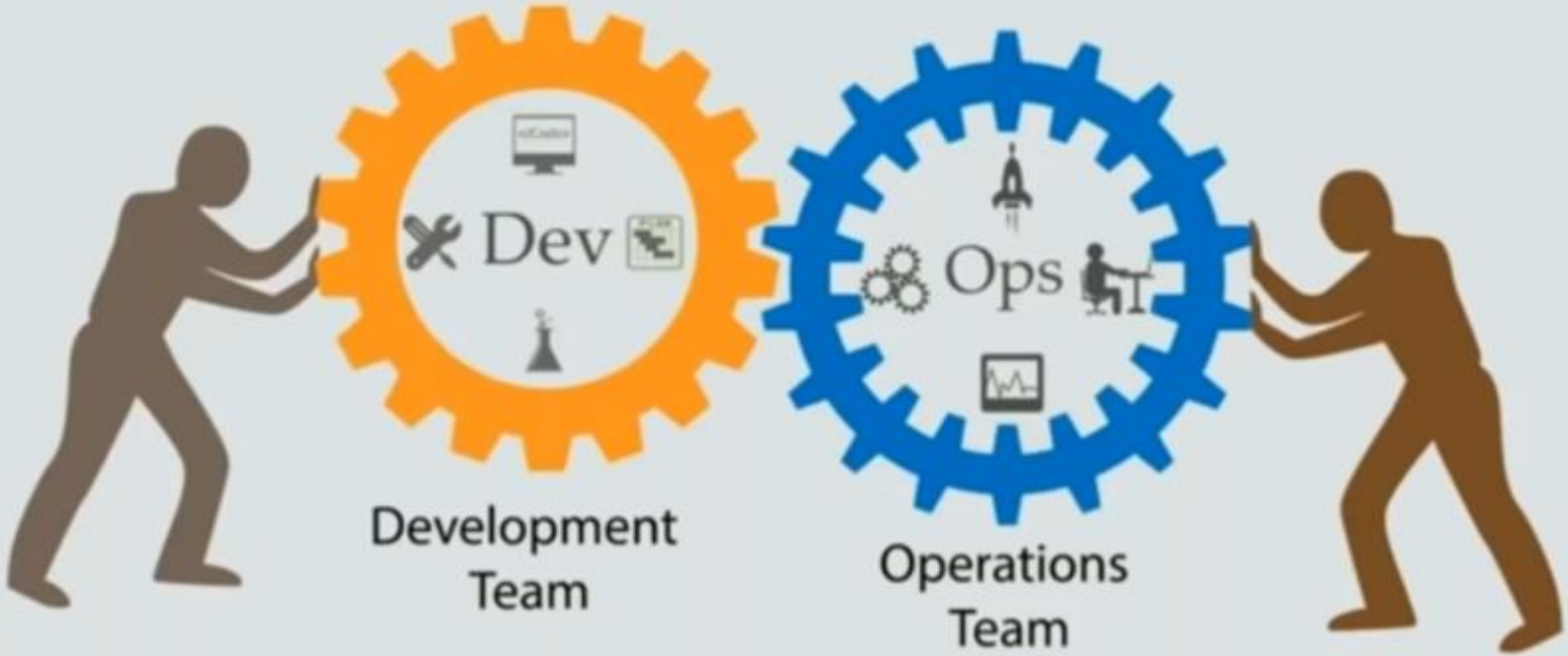
- Frequent Delivery
- Face-to-Face communication with clients and get feedback on delivered software
- Efficient design and fulfils the business requirement.
- Anytime changes are acceptable
- It reduces total development time.

Waterfall Vs Agile



Limitations of Agile

Solution is DevOps



DevOps

DevOps commonly includes the following processes:

- Agile practices (http://agilemanifesto.org/principles.html)
- Continuous integration
- Release automation
- Functional unit testing
- System integration testing
- Service and infrastructure monitoring

DevOps Capabilities

- Capacity management
  - How much, big, far
  - Scalability
  - Management
- Production management
  - Incident management
  - Problem resolution
  - Communication
- Monitoring
  - Telemetry
  - Logs
  - Alerting
- Problem management
  - Root cause
  - Trends, recurring issues
  - Peer reviews

DevOps Capabilities

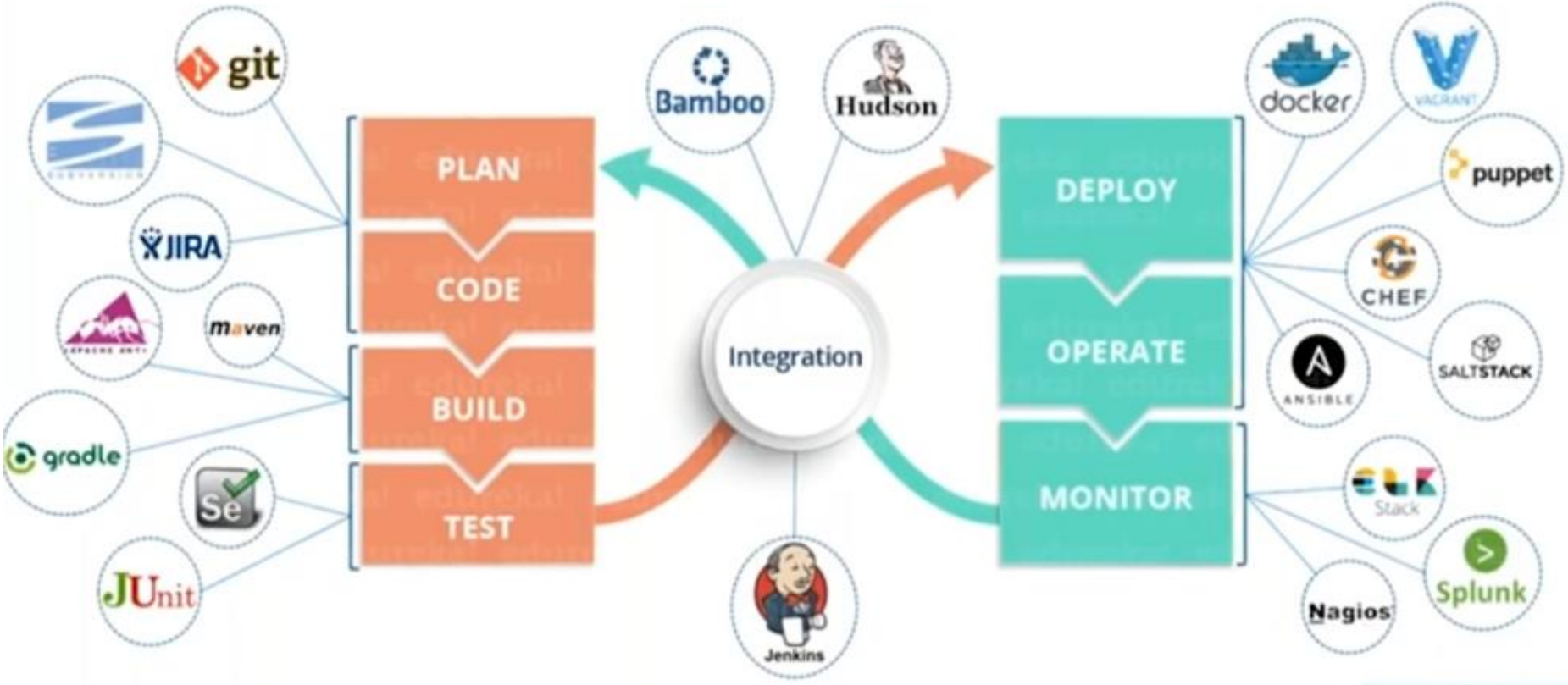
- Change management
  - Provisioning New/Updates
  - Configuration management
  - Orchestration
- Service design
  - Architecture
  - Automation
  - Security
- Service management
  - SLAs and uptime
  - Costing and budgeting
  - Metrics and visualization

DevOps Goals

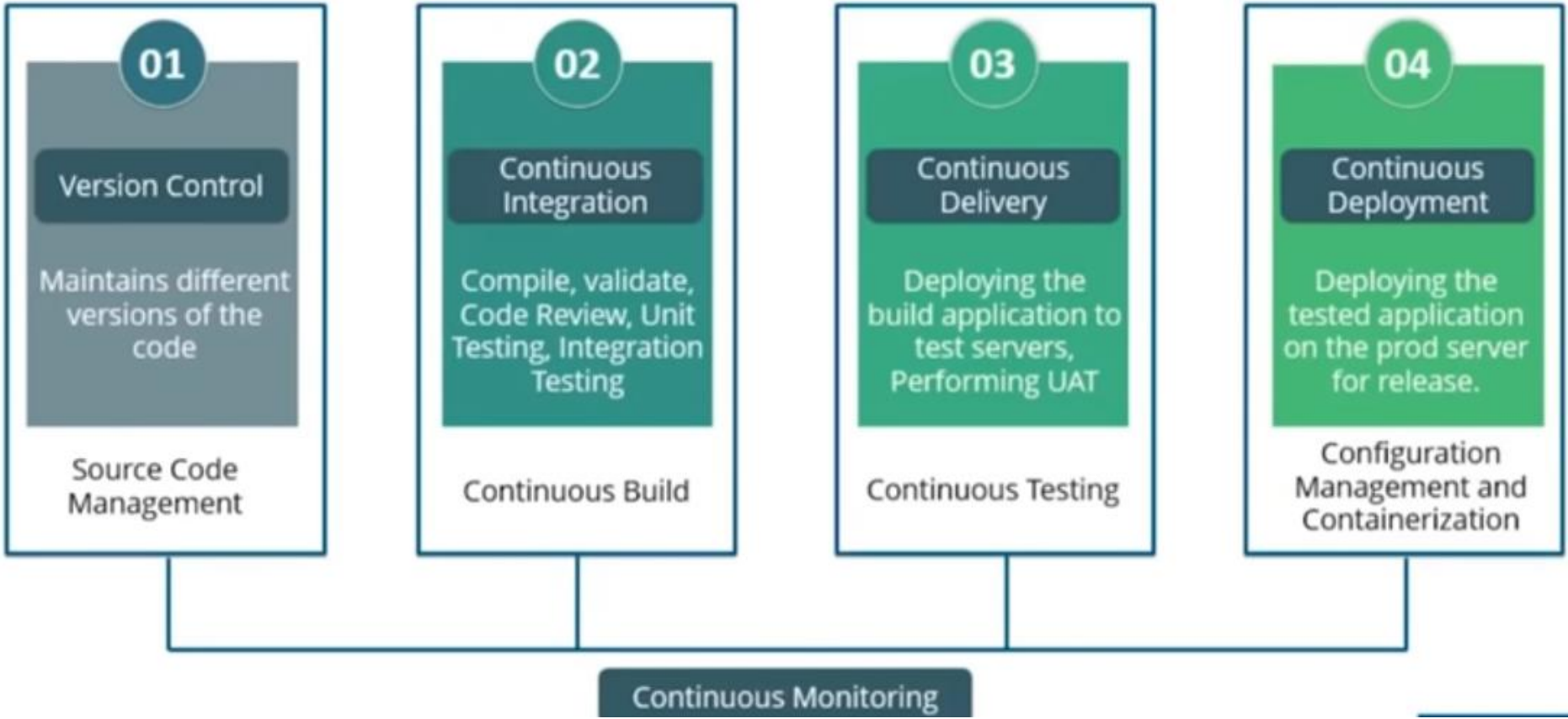
- DevOps is about
  - systems thinking
  - process definition
  - team coordination
  - feedback loops
- DevOps is also about continuous improvement
- Aims to improve quality and reduce the number of incidents in production
- Overall efficiency in the process will increase as the team learns best practices
- Some key improvements are:
  - Reduce testing time
  - Reduce the amount of changes required due to incomplete or conflicting requirements
- Design the simplest solution required to meet requirements
- Leverage standardized integrated software tools
- Create robust production systems which lower maintenance costs
- Produce software to the highest quality
- Improve the performance of individuals and teams

What is DevOps ?

DevOps is a culture which promotes collaboration between Development and Operations team to deploy code to production faster in an automated & repeatable way.

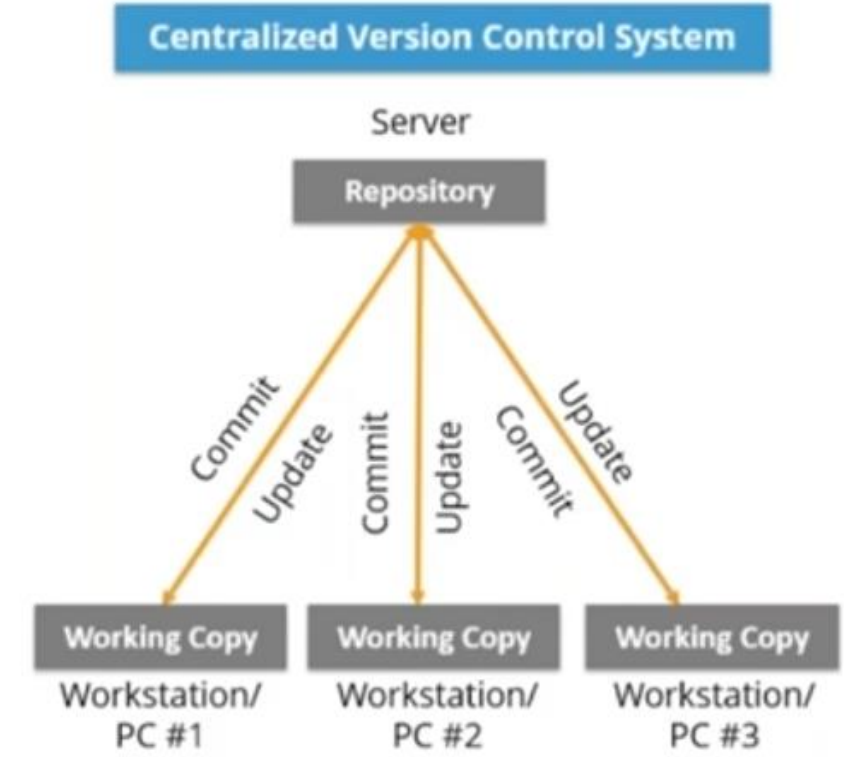


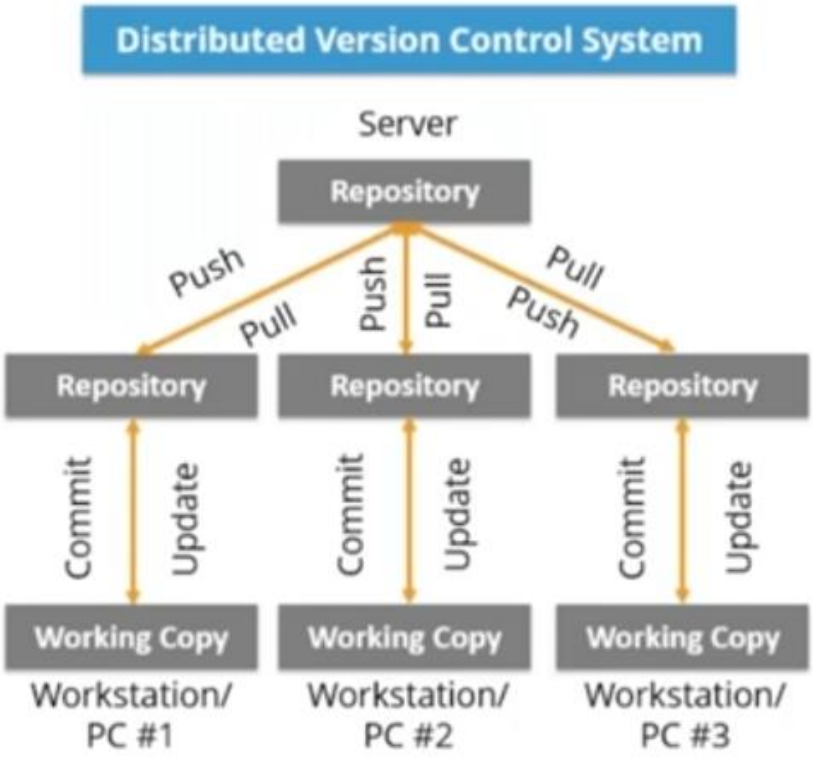
DevOps Stages?



Source Code Management

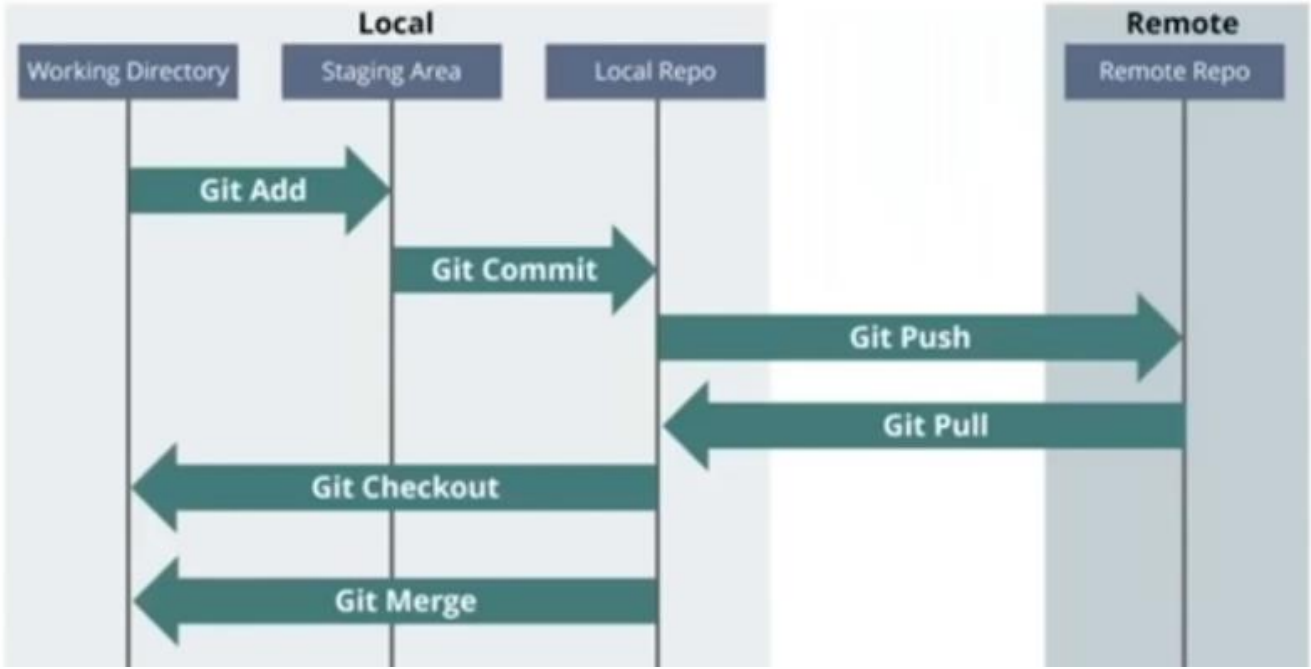
The management of changes to documents, computer programs, large websites and other collection of information





Source Code Management

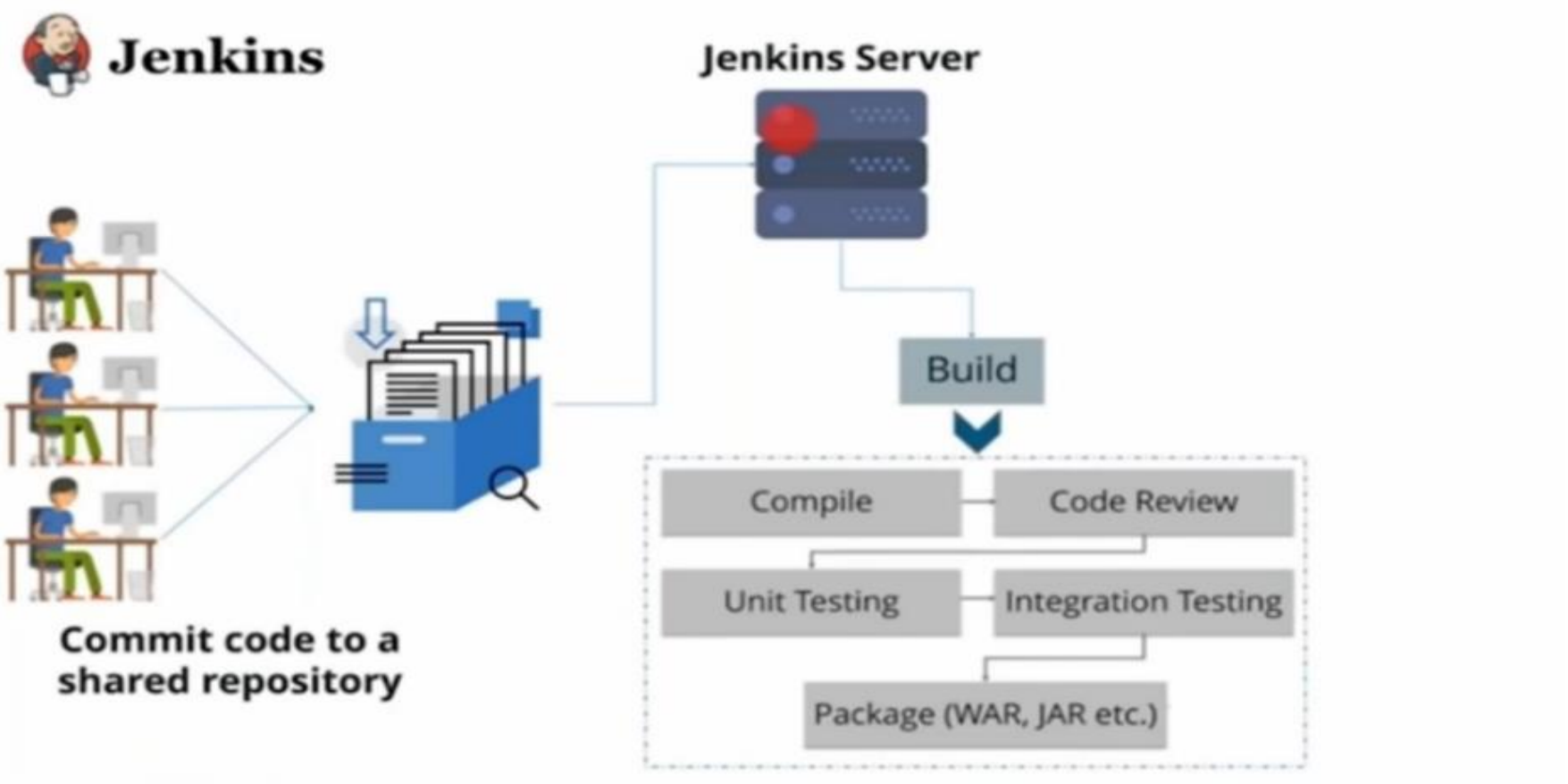
Git is a Distributed Version Control tool that supports distributed non-linear workflows by providing data assurance for developing quality software



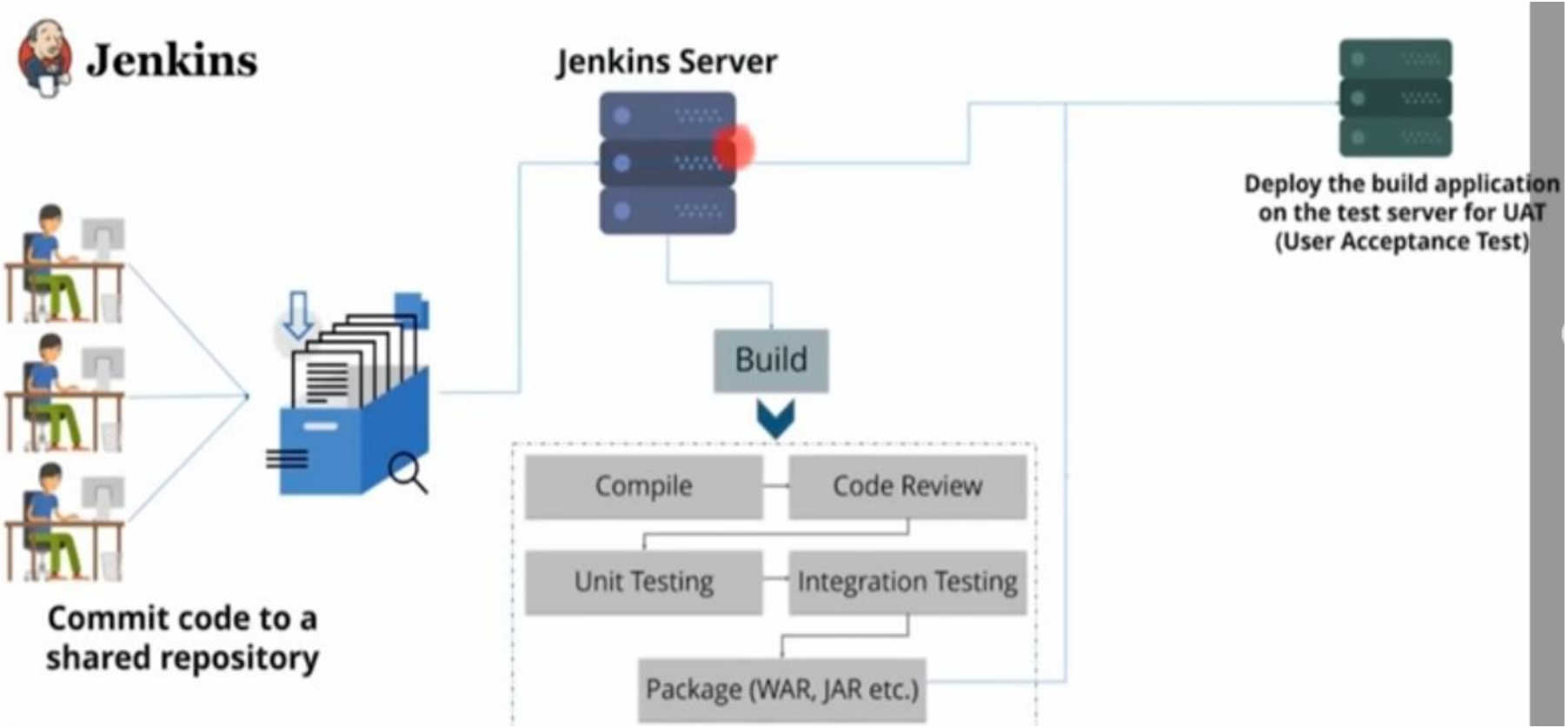
Source Code Management

- git –version
- mkdir gitdemo
- cd gitdemo
- git init
- git add filename.ts
- git status
- git commit –m “demo commit”
- git remote add origin git@github.com:gitdemo/gitdemo.git
- git pull origin master
- git push origin master

Continuous Integration

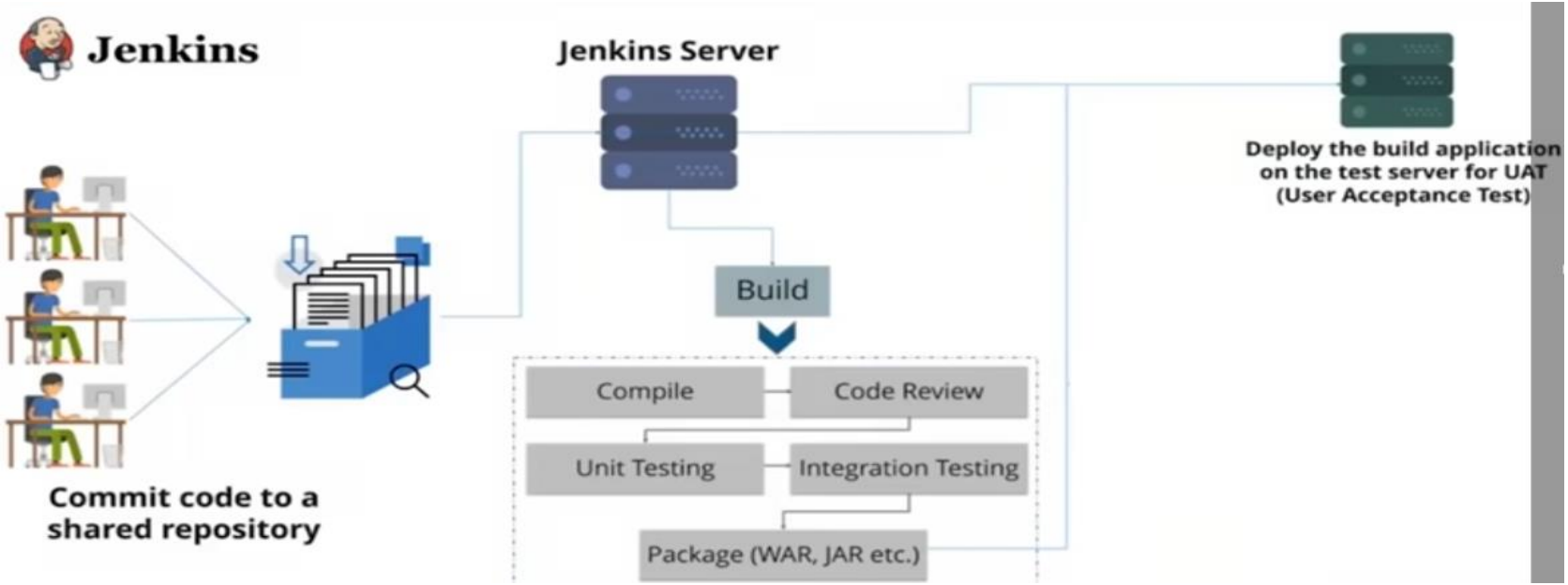


Continuous Delivery

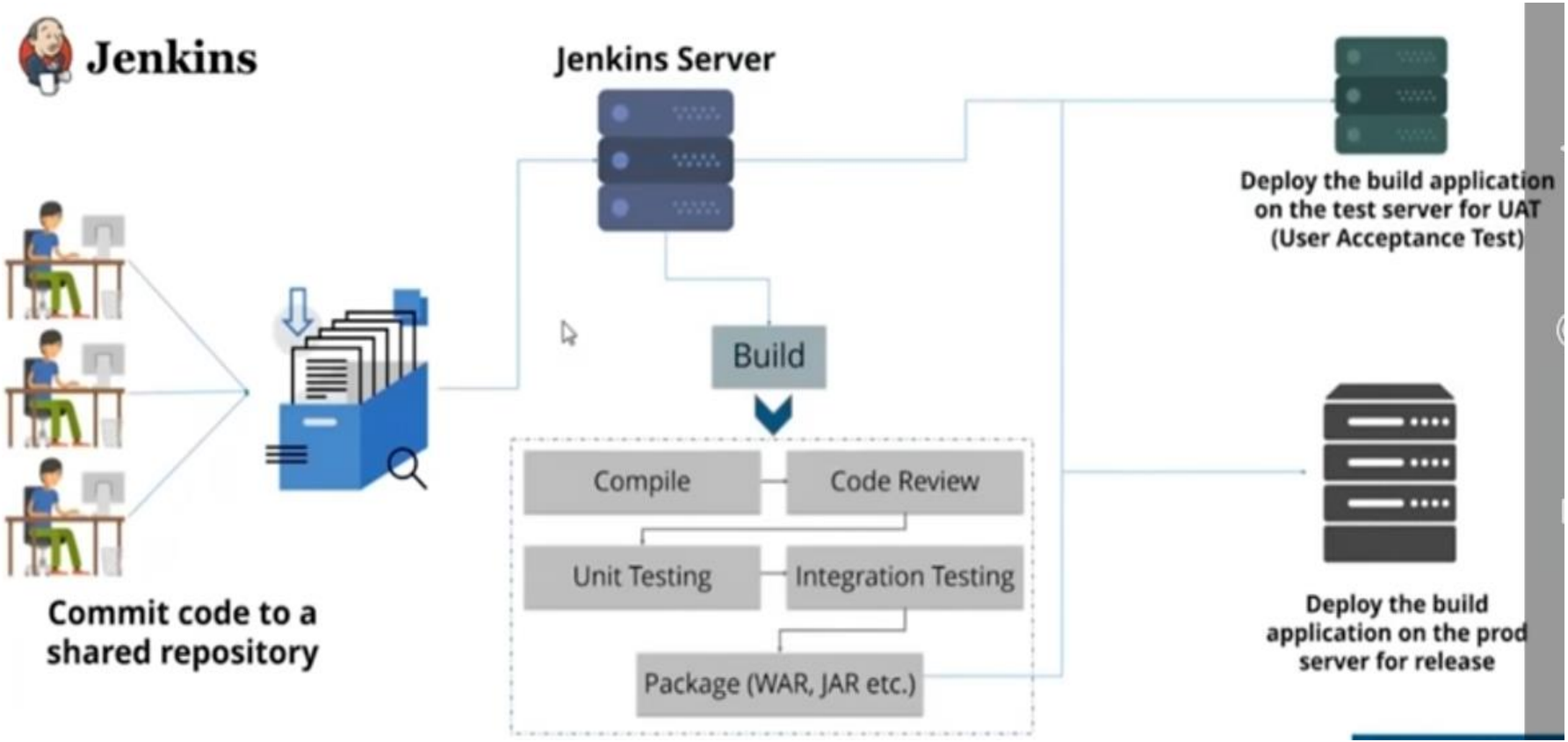


Continuous Delivery

Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies.

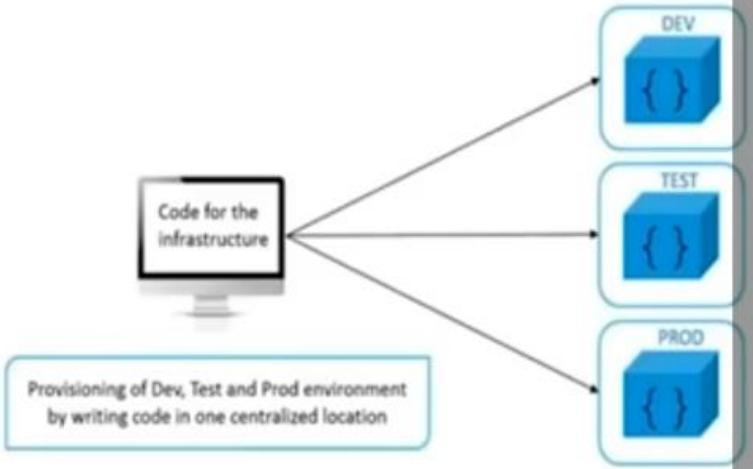


Continuous Deployment



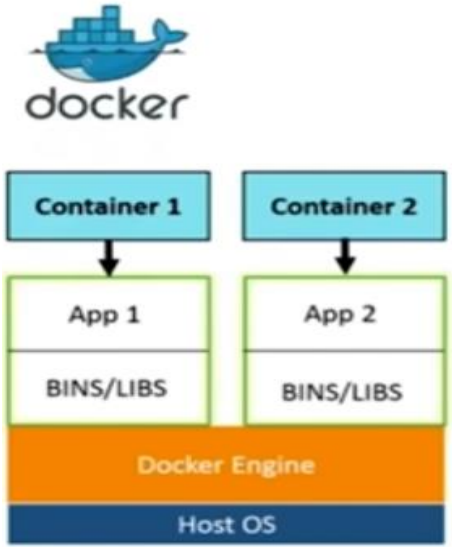
Configuration Management

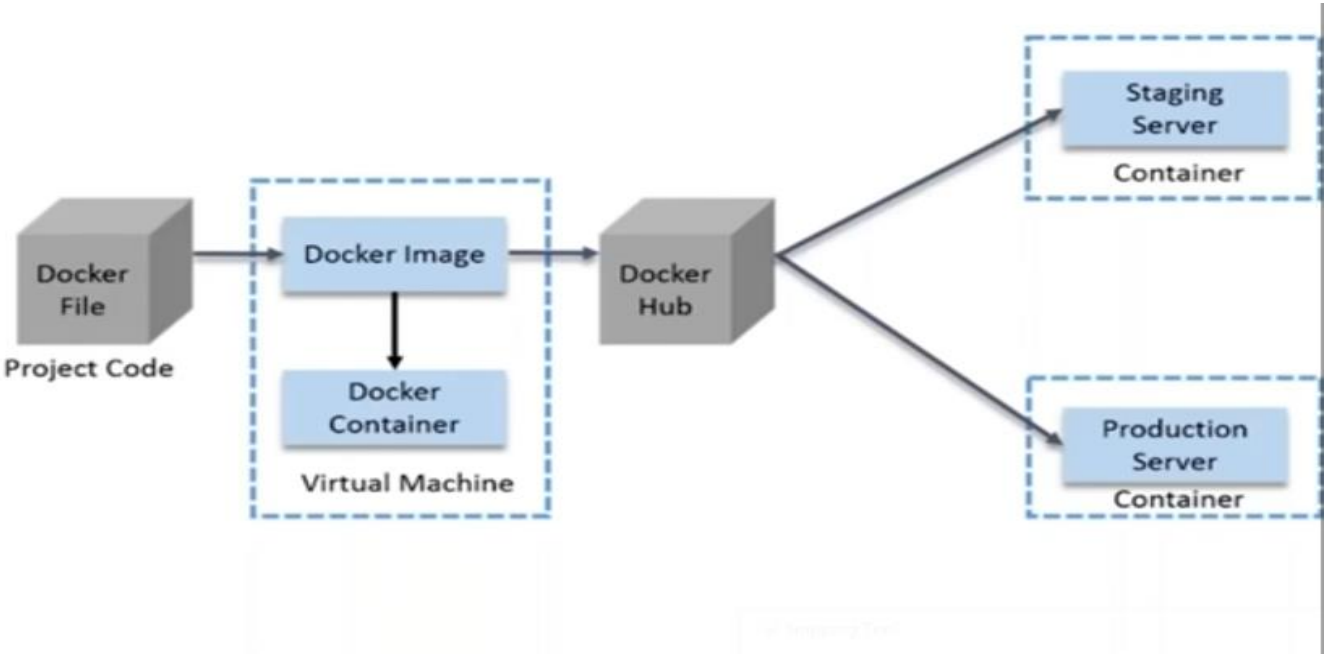
- Configuration Management is the practice of handling changes systematically so that a system maintains its integrity over time
- Configuration Management (CM) ensures that the current design and build state of the system is known, good & trusted
- It doesn't rely on the tacit knowledge of the development team



Containerization

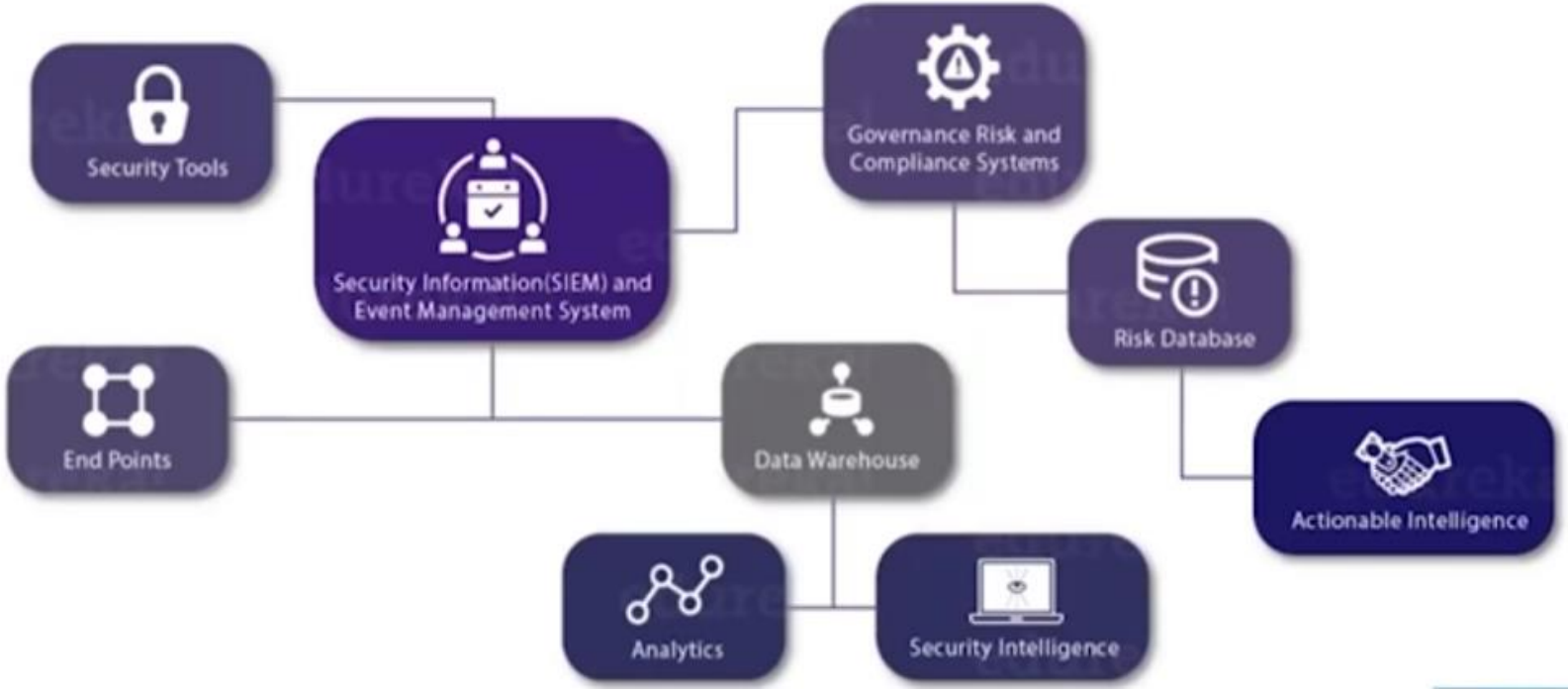
Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.





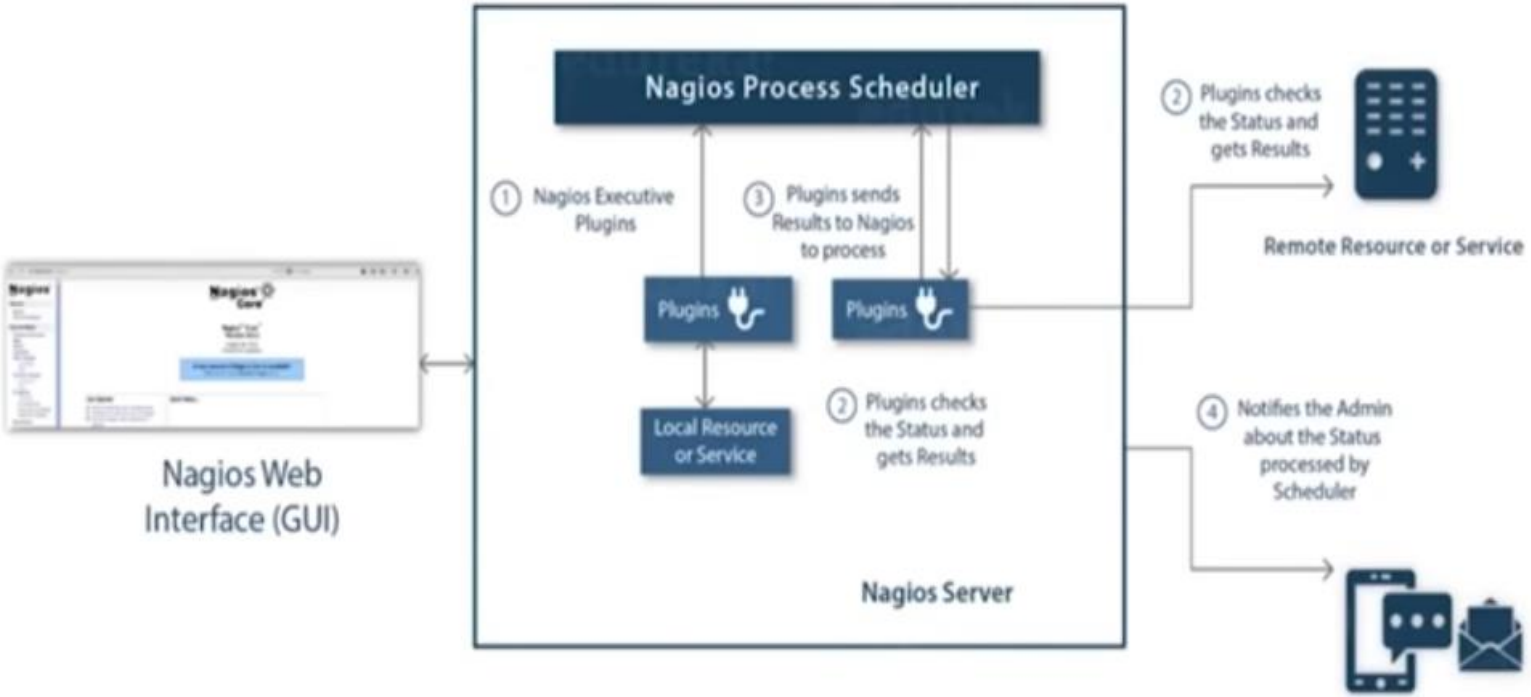
Continuous Monitoring

Continuous Monitoring is all about the ability of an organization to detect, report, respond, contain and mitigate the attacks that occur, in its infrastructure



Continuous Monitoring

Nagios®



DevOps Engineer Skills

- Skills Every DevOps Engineer Must Have
- Flexibility Coding is an ongoing process & always needs to be updated
- Security Skills
  - skilled areas security is always of the utmost importance
- Collaboration
- Scripting Skills
- Decision-making
- Infrastructure Knowledge
- Soft Skills

Recap

- Waterfall Model
- Agile Method
- DevOps
- DevOps Stages
  - Source Code Management
  - Continuous Integration
  - Continuous Delivery
  - Continuous Deployment
  - Configuration Management
  - Continuous Monitoring

Thank you