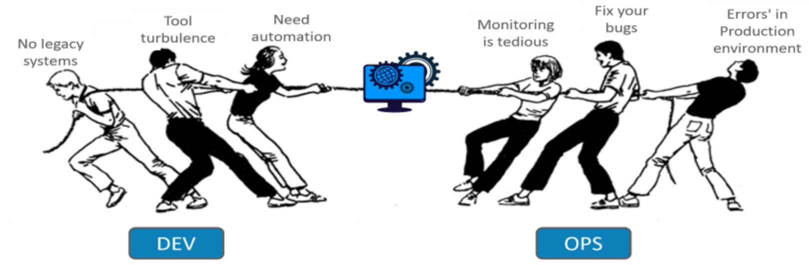
Cognizant Digital Experience

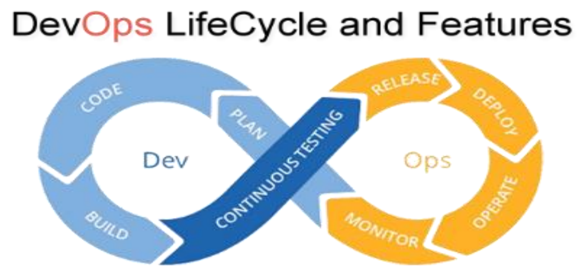
DevOps Overview

Cognizant Digital Experience

What is DevOps

Why DevOps





DevOps Culture

DevOps culture is about **collaboration** between Dev and Ops.

Under the traditional separation between Dev and Ops, Dev and Ops have **different** and **opposing** goals – speed vs stability.

With DevOps, Dev and Ops work together and share the **same** goals.

These goals include things like Fast time-to-market (TTM), Few production failures, Immediate recovery from failures

Definition

DevOps is a software engineering **culture** and **practice** that aims at unifying software development (Dev) and software operation (Ops).

DevOps aims at shorter development cycles, increased deployment frequency, more dependable releases, in close alignment with business objectives.

DevOps Is NOT...

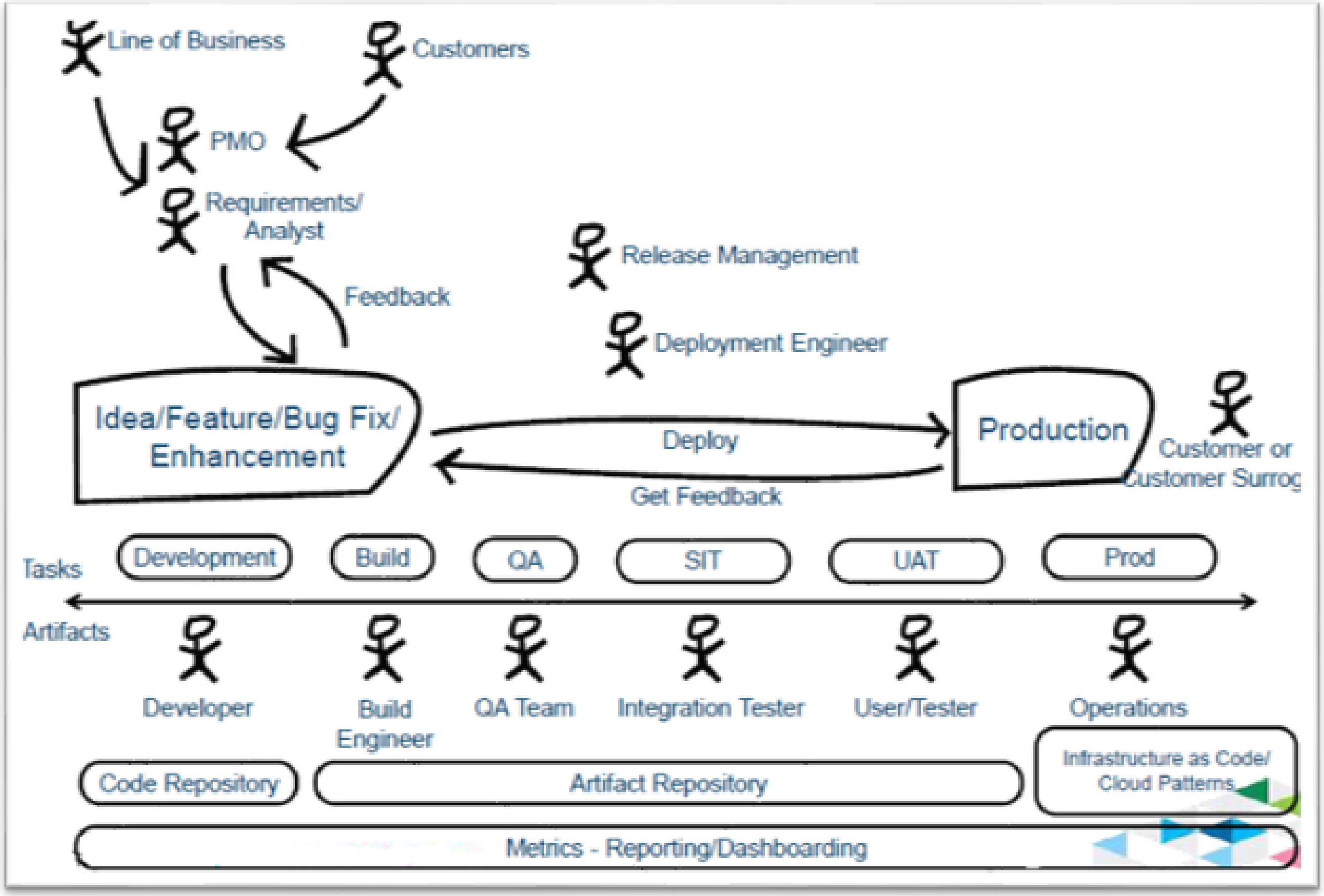
DevOps is NOT tools, but Tools are essential to success in DevOps

DevOps is NOT a standard

DevOps is NOT a product

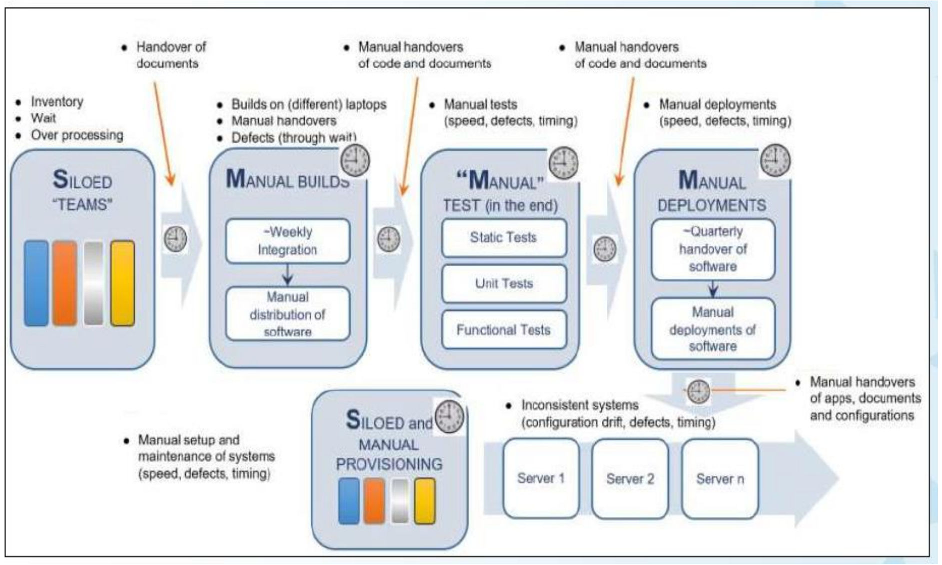
DevOps is NOT a job title

DevOps Ecosystem



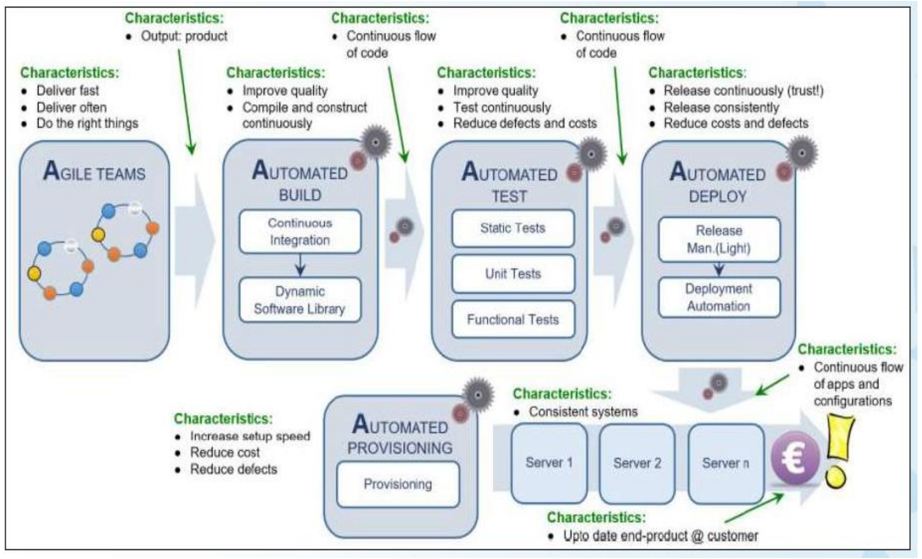
Cognizant Digital Experience

Life With / Without DevOps



Handover of documents
Manual handovers of code and documents
Manual handovers of code and documents
Inventory
Wait
Over processing
Builds on (different) laptops
Manual handovers
Defects (through wait)
Manual tests (speed, defects, timing)
Manual deployments (speed, defects, timing)
SILOED "TEAMS"
MANUAL BUILDS
~Weekly Integration
Manual distribution of software
"MANUAL" TEST (in the end)
Static Tests
Unit Tests
Functional Tests
MANUAL DEPLOYMENTS
~Quarterly handover of software
Manual deployments of software
Manual handovers of apps, documents and configurations
Inconsistent systems (configuration drift, defects, timing)
Manual setup and maintenance of systems (speed, defects, timing)
SILOED and MANUAL PROVISIONING
Server 1
Server 2
Server n

Without DevOps



With DevOps

Cognizant Digital Experience

DevOps Building Blocks

DevOps Building Blocks

Build Automation

- Automation of the process of preparing code for deployment to a live environment
- Depending on what languages are used, code needs to be compiled, linted, minified, transformed, unit tested, etc.
- Build automation means taking these steps and doing them in a consistent, automated way using a script or tool
- The tools of build automation often differ depending on what programming languages and frameworks are used, but they have one thing in common: automation!

Continuous Integration

- This is the practice of frequently merging code changes done by developers.
- Continuous integration means merging constantly throughout the day, usually with the execution of automated tests to detect any problems caused by the merge.
- Merging all the time could be a lot of work, so to avoid that it should be automated.

Continuous Delivery and Continuous Deployment

- Continuous Delivery is the practice of continuously maintaining code in a deployable state
- Continuous Deployment is the practice of frequently deploying small code changes to production
- Some use the terms continuous delivery and continuous deployment interchangeably, or simply use the abbreviation CD

Infrastructure as Code

- Infrastructure as Code (IaC): manage and provision infrastructure through code and automation.
- With infrastructure as code, instead of doing things manually, you use automation and code to create and change servers, instances, environments, containers, other infrastructure.
- With IaC, provisioning new resources and changing existing resources are both done through automation

Configuration Management

- Configuration Management maintains and changes the state of pieces of infrastructure in a consistent, maintainable, and stable way
- Configuration management minimizes configuration drift (small changes that accumulate over time and make systems different from one another)
- Infrastructure as Code is very beneficial for configuration management

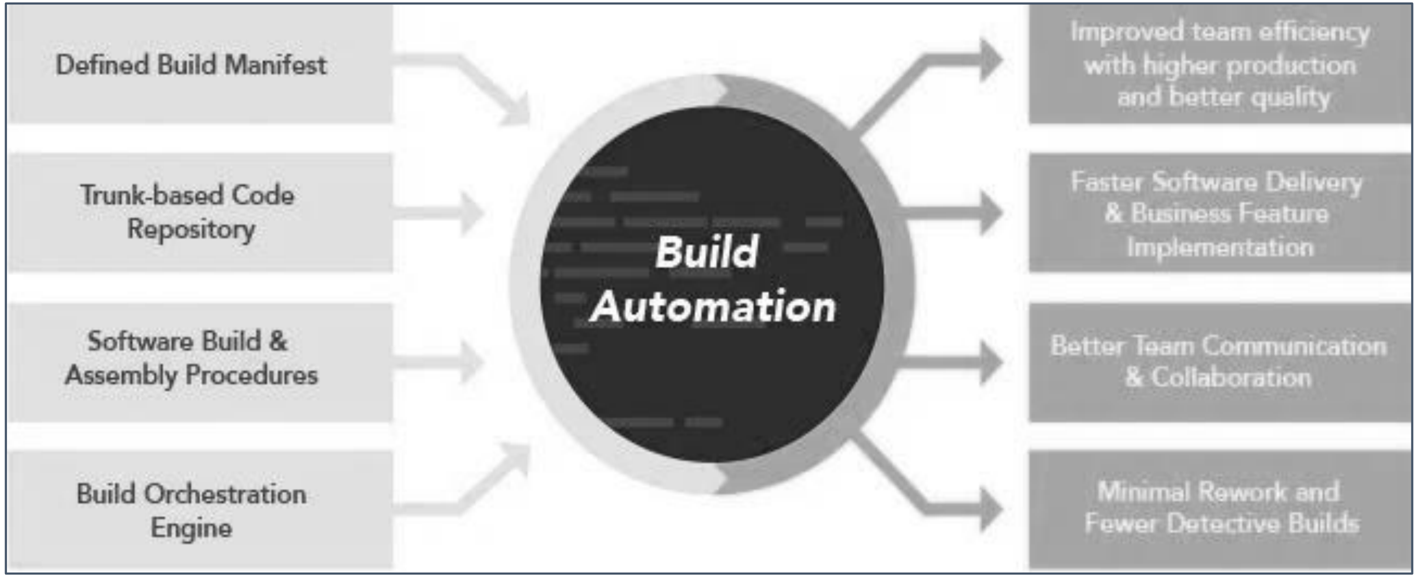
Orchestration

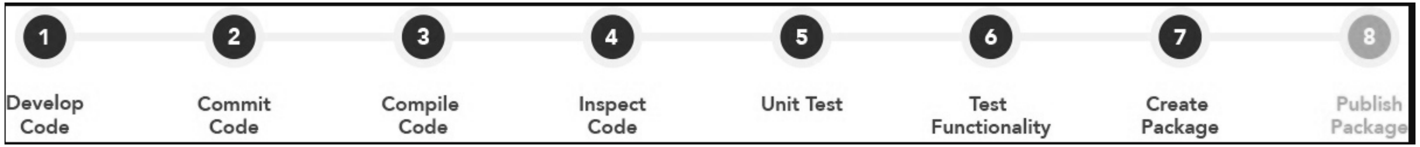
- This is an automation that supports processes and workflows, such as provisioning resources
- For example, a monitoring tool detects an increased load on the service. An orchestration tool responds to this by spinning up additional resources to handle the load. When the load decreases again, the tool spins the additional resources back down, freeing them up to be used by something else

Monitoring

- Monitoring is the collection and presentation of data about the performance and stability of services and infrastructure
- There are two types of monitoring - Infrastructure monitoring (focuses on things related to infrastructure eg. CPU, RAM) & Application Performance Monitoring (APM) – focuses on performance and stability of individual parts of an application (focuses on performance and stability of individual parts of an application eg. response times, logs)
- Monitoring helps with real-time notifications and postmortem analysis

Build Automation Illustrated


Defined Build Manifest
Trunk-based Code Repository
Software Build & Assembly Procedures
Build Orchestration Engine
Build Automation
Improved team efficiency with higher production and better quality
Faster Software Delivery & Business Feature Implementation
Better Team Communication & Collaboration
Minimal Rework and Fewer Defective Builds



Build Automation Overview

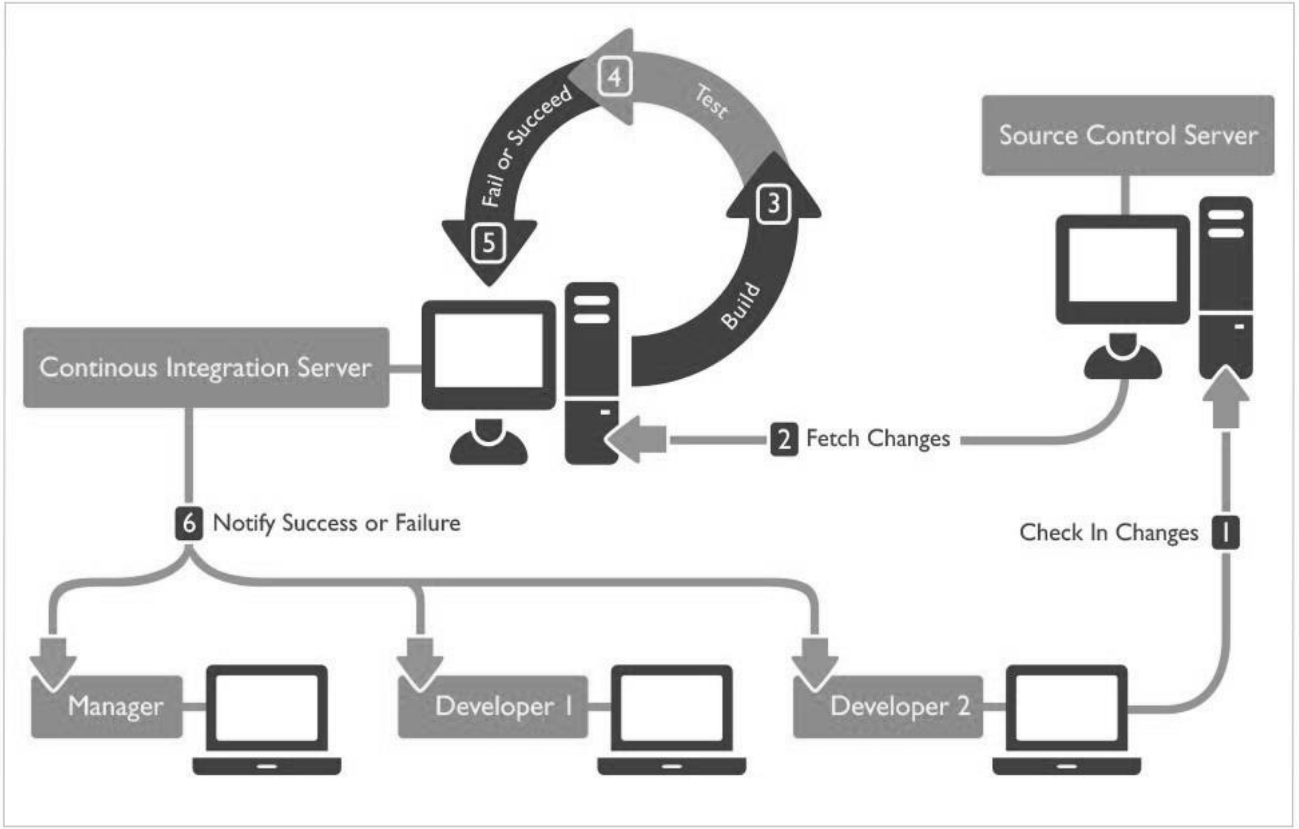
Tools

- **Java – Ant, Maven, Gradle**
- **JavaScript – npm, Grunt, Gulp**

Benefits

Fast	• Automation handles tasks that would otherwise need to be done manually.
Consistent	• The build happens the same way every time, removing problems and confusion that can happen with manual builds.
Repeatable	• The build can be done multiple times with the same result. Any version of the source code can always be transformed into deployable code in a consistent way.
Portable	• The build can be done the same way on any machine. Anyone on the team can build on their machine, as well as on a shared build server. Building code doesn't depend on specific people or machines.
Reliable	• There will be fewer problems caused by bad builds.

Continuous Integration Illustrated



Cognizant Digital Experience

Continuous Integration Overview

Tools

Jenkins - Open source – fork of Hudson; Widely used, Java servlet-based

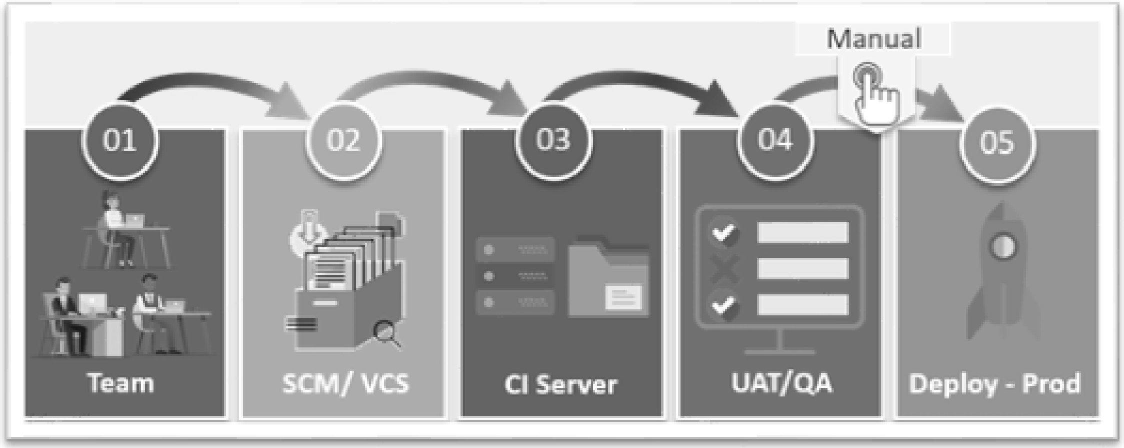
TravisCI - Open source, Built around Github integration, Executes builds in clean VMs

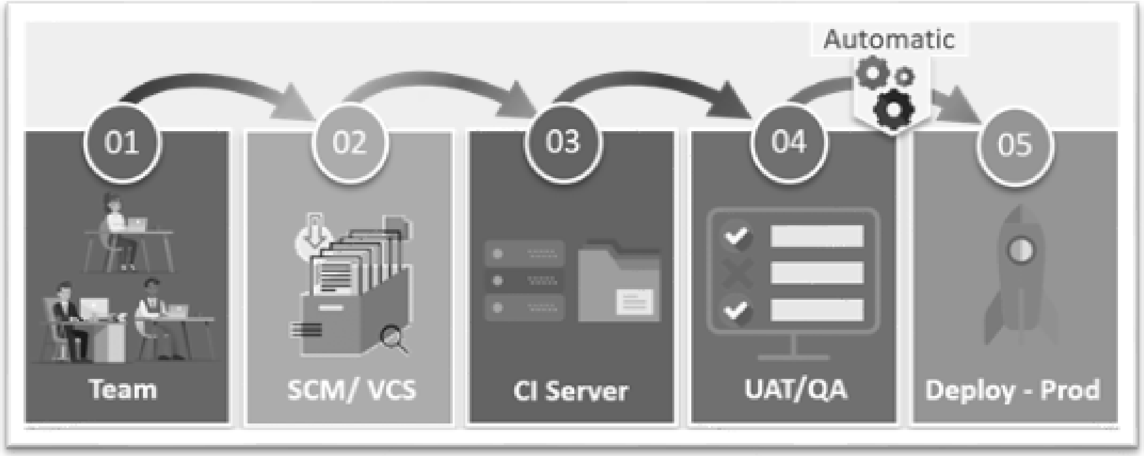
Bamboo - Enterprise product by Atlassian, Out-of-the-box integration with other Atlassian products like JIRA and Confluence

Benefits

Early detection of certain types of bugs	• If code doesn't compile or an automated test fails, the developers are notified and can fix it immediately. The sooner these bugs are detected, the easier they are to fix!
Eliminate the scramble to integrate just before a big release	• The code is constantly merged, so there is no need to do a big merge at the end.
Makes frequent releases possible	• Code is always in a state that can be deployed to production.
Makes continuous testing possible	• Since the code can always be run, QA testers can get their hands on it all throughout the development process, not just at the end.
Encourages good coding practices	• Frequent commits encourages simple, modular code.

Continuous Delivery and Continuous Deployment Illustrated





Cognizant Digital Experience

Continuous Delivery and Continuous Deployment Overview

Tools

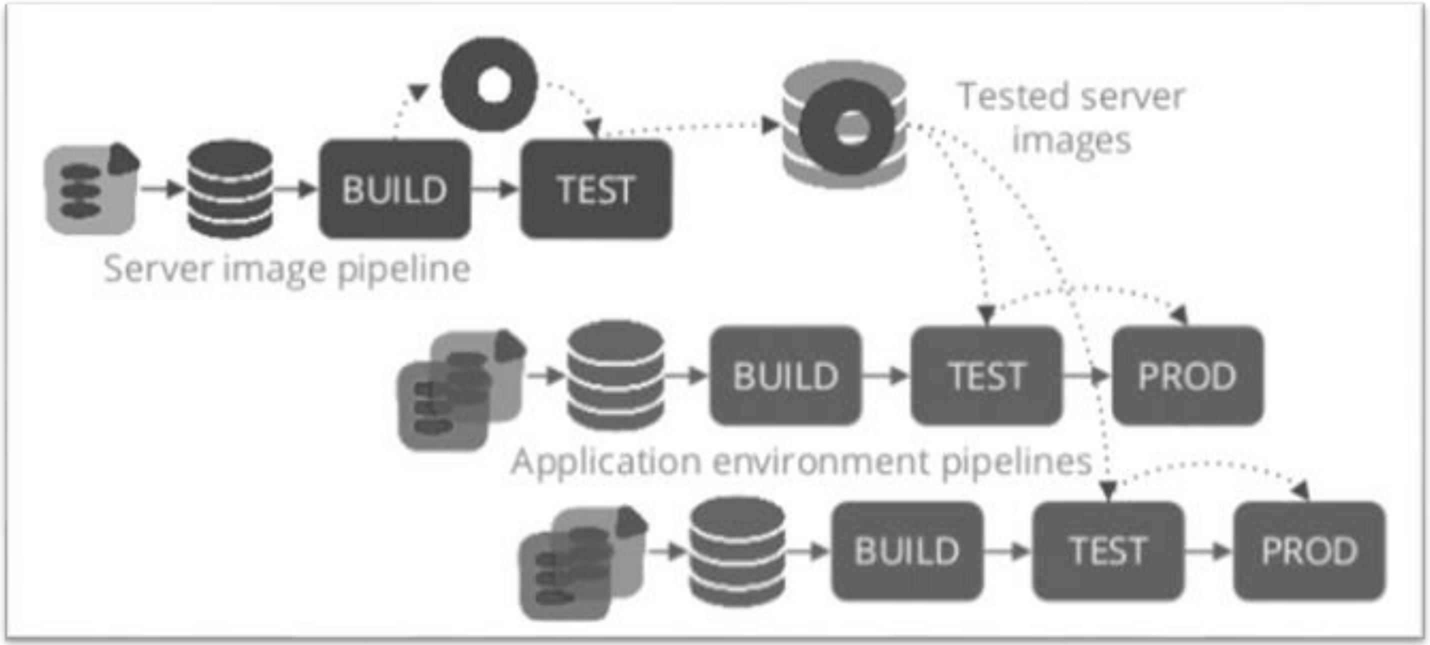
Azure DevOps Pipeline - Continuously build, test and deploy to any platform and cloud

AWS CodeDeploy - Fully managed deployment service that automates software deployments to a variety of AWS compute services.

Benefits

Faster time-to-market	• Get features into the hands of customers more quickly rather than waiting for a lengthy deployment process that doesn't happen often.
Fewer problems caused by the deployment process	• Since the deployment process is frequently used, any problems with the process are more easily discovered.
Lower risk	• The more changes are deployed at once, the higher the risk. Frequent deployments of only a few changes are less risky.
Reliable rollbacks	• Robust automation means rollbacks are a reliable way to ensure stability for customers, and rollbacks don't hurt developers because they can roll forward with a fix as soon as they have one.
Fearless deployments	• Robust automation plus the ability to rollback quickly means deployments are commonplace, everyday events rather than big, scary events.

Infrastructure as Code Illustrated



Infrastructure as Code Overview

Tools

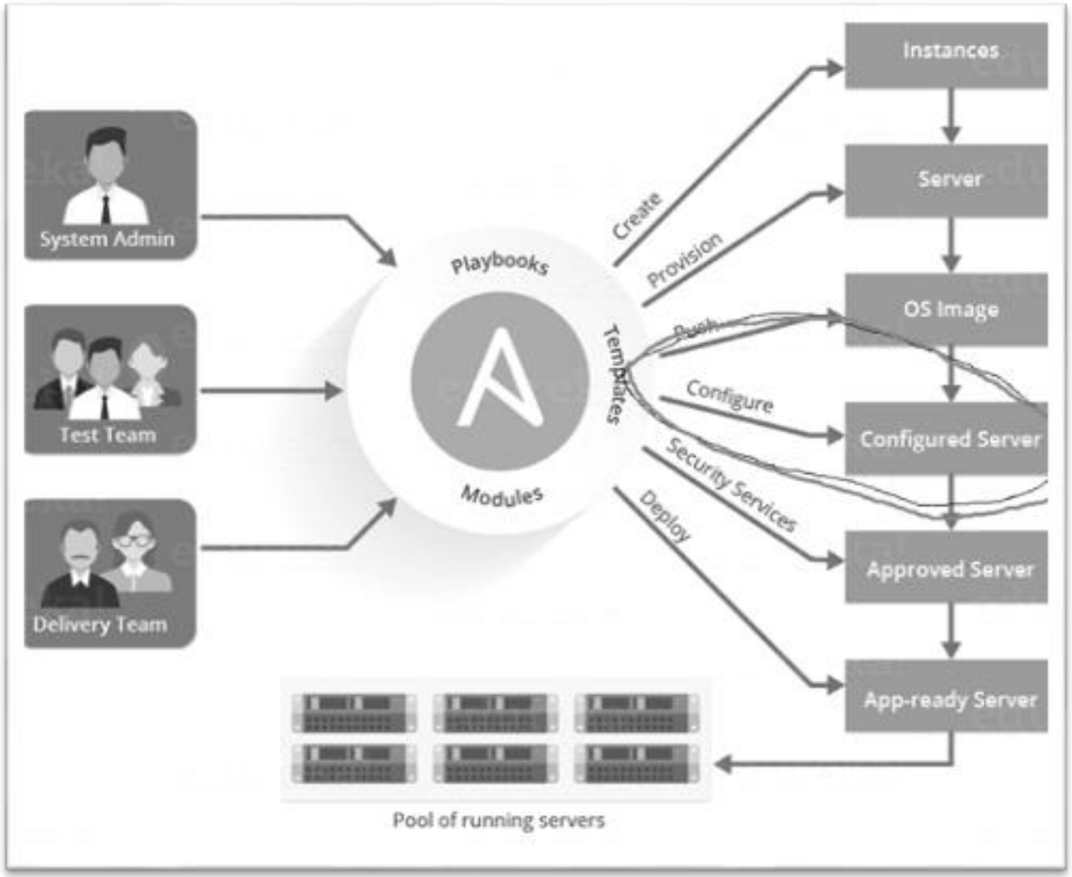
PowerShell Desired State Configuration (DSC) - DSC is a management platform in PowerShell that helps to manage IT and development infrastructure with configuration as code.

AWS CloudFormation - Provides an easy way to model a collection of related AWS and third-party resources, provision them quickly and consistently, and manage them throughout their lifecycles, by treating infrastructure as code.

Benefits

Consistency in creation and management of resources	• The same automation will run the same way every time.
Reusability	• Code can be used to make the same change consistently across multiple hosts and can be used again in the future.
Scalability	• Need a new instance? You can have one configured exactly the same way as the existing instances in minutes (or seconds).
Self-documenting	• With IaC, changes to infrastructure document themselves to a degree. The way a server is configured can be viewed in source control, rather than being a matter of who logged in to the server and did something.
Simplify the complexity	• Complex infrastructures can be stood up quickly once they are defined as code. A group of several interdependent servers can be provisioned on demand.

Configuration Management Illustrated



Cognizant Digital Experience

Configuration Management Overview

Tools

Ansible - Open source, Declarative configuration, YAML configuration files

Puppet - Declarative configuration, Manage state through a UI, Pushes changes to clients using a control server and agents installed on clients.

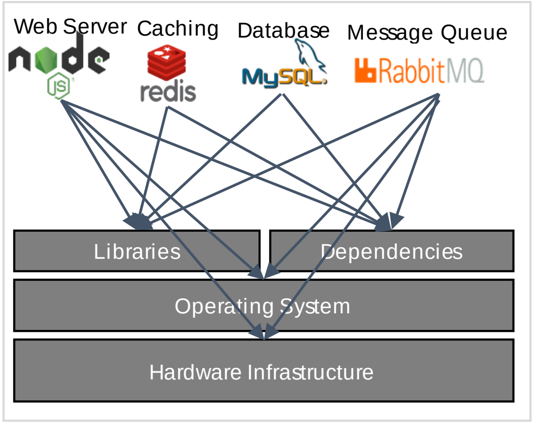
Chef - Procedural configuration

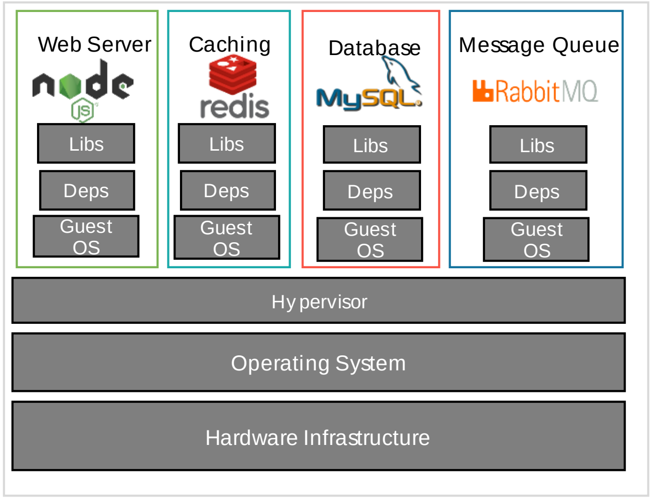
Benefits

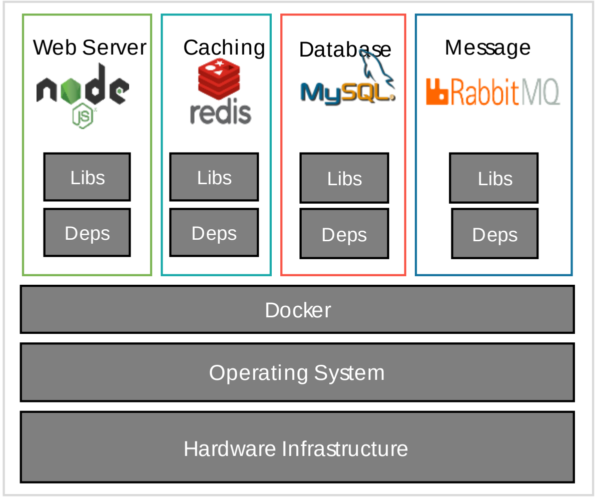
Save time	• It takes less time to change the configuration.
Insight	• With good configuration management, you can know about the state of all pieces of a large and complex infrastructure..
Maintainability	• A more maintainable infrastructure is easier to change in a stable way.
Less configuration drift	• It is easier to keep a standard configuration across a multitude of hosts.

Cognizant Digital Experience

Orchestration Illustrated







Pre Virtualization

Virtualization

Containerization

Limitations:
1. Matrix of hell
2. Compatibility/ Dependency Issue
3. Long setup time
4. Environment/ machine dependency
5. Licensing Issue

Benefits:
1. Easy to scale
2. Libraries/ Dependencies into each Container
3. Same Libraries/ Dependencies across environment
4. Easy to template application dependencies

Limitations:
1. Duplicate OS, hence more resources
2. Consumes more space
3. Takes more time to boot up

Benefits:
1. Containerize Applications
2. Libraries/ Dependencies into each Container
3. Same containers across environment
4. Runtime isolation
5. Easy to template application dependencies
6. No OS in containers
7. Consumes less space
8. Faster boot up

Orchestration Overview

Tools

Docker Swarm - Docker-native, Orchestration for Docker containers

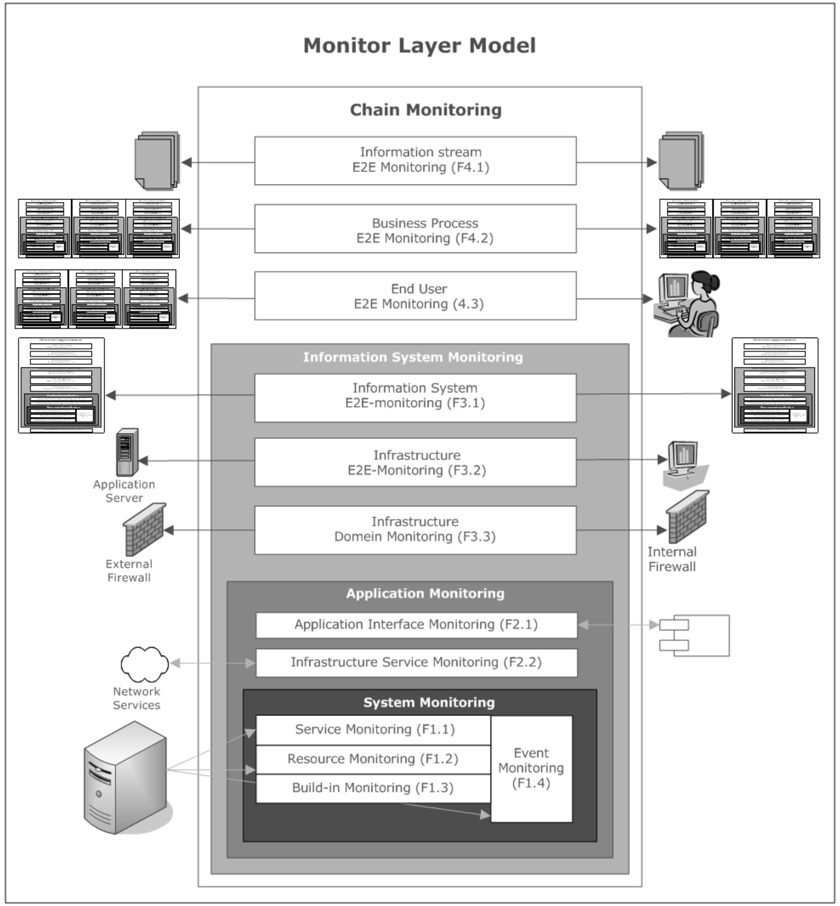
Kubernetes - Open source Orchestration server, Manage containerized apps across multiple hosts

Terraform - Combines orchestration and infrastructure-as-code, Works well with other tools, like Ansible, Works well with AWS, Integrates with Kubernetes

Benefits

Scalability	• It Resources can be quickly increased or decreased to meet changing needs.
Stability	• Automation tools can automatically respond to fix problems before users see them.
Save time	• Certain tasks and workflows can be automated, freeing up engineers' time.
Self-service	• Orchestration can be used to offer resources to customers in a self-service fashion.
Granular insight into resource usage	• Orchestration tools give greater insight into how many resources are being used by what software, services, or customers.

Monitoring Illustrated


Monitor Layer Model
Chain Monitoring
Information stream E2E Monitoring (F4.1)
Business Process E2E Monitoring (F4.2)
End User E2E Monitoring (4.3)
Information System Monitoring
Information System E2E-monitoring (F3.1)
Infrastructure E2E-Monitoring (F3.2)
Infrastructure Domein Monitoring (F3.3)
Application Server
External Firewall
Internal Firewall
Application Monitoring
Application Interface Monitoring (F2.1)
Infrastructure Service Monitoring (F2.2)
Network Services
System Monitoring
Service Monitoring (F1.1)
Resource Monitoring (F1.2)
Build-in Monitoring (F1.3)
Event Monitoring (F1.4)

Cognizant Digital Experience

Monitoring Overview

Tools

New Relic - A web application performance service designed to work in real-time with live web app. New Relic Infrastructure provides flexible, dynamic server monitoring.

AppDynamics - It is a leading Application Performance Management (APM) product. It is a tool that monitors Application Infrastructure and gives code level visibility.

Dynatrace: Dynatrace is an all-in-one performance monitoring solution designed to serve businesses of all sizes. The software supports full-stack monitoring to enable businesses to detect and diagnose both performance and availability issues

Benefits

Fast recovery	• The sooner a problem is detected, the sooner it can be fixed. You want to know bout a problem before your customer does!
Better root cause analysis	• The more data you have, the easier it is to determine the root cause of a problem..
Visibility across teams	• Good monitoring tools give useful data to both developers and operations people about the performance of code in production.
Automated response	• Monitoring data can be used alongside orchestration to provide automated responses to events, such as automated recovery from failures.

Continued Learning

Course Title	Course URL	Course Duration	Learning Goals
The Dev Ops Essentials - The Handbook	https://cognizant.udemy.com/course/the-devops-essentials/	1h 56m	• Gain a solid Understanding of DevOps Practices • Learn about Continuous Integration and Delivery and its role in DevOps • Dev Ops terminology • The History and various roles in DevOps

Section Name	# of Lectures	Duration (in minutes)	Mandatory / Optional
Introduction	19	116	Mandatory

Thank You