



DevOps

# DevOps Overview

- Developers and Operation Team collaborate together to increase productivity
- Big focus on automation, which includes testing new codes, workflows, and infrastructure
- Environments for development and environments for production are equal in that software are written in small parts. This allows an increase in deployment frequency and speed of new code
- DevOps is beneficial because
  - it increase the rate in software delivery
  - Automated infrastructure allows more focus on activities that gives more value such improving business content

# DevOps VS. AGILE

- DevOps
  - Objective is to manage end-to-end engineering process by focusing on deliver and constant testing
  - DevOps implement collaborations between different processes
  - Teams are separated into operation and development
  - Gets feedback internally
- Agile
  - Objective is to manage projects by regulating changes that occurs
  - Agile implement tactical frameworks
  - Teams are not separated and all members have different skill sets
  - Gets feedback externally via customers

# DevOps and Agile Similarities

- Both Agile and DevOps emphasize on development process.
- Both Agile and DevOps have a focus on testing
- Both Agile and DevOps create higher quality

# How DevOps Supports Agile

- The constant changes occurs during software development are unavoidable, Agile supports the changes while DevOps focuses on minimizing the risks of these changes to the entire project
- Agile strives for quality by adapting to changes and producing better application with the given requirements. DevOps supports Agile's goal for better quality by automation and early bug removal
- DevOps supports Agile release cycle
- DevOps adopts an iterative process much like agile

# Metrics for measuring DevOps

- For Quality
  - Metric: Error Rates (also for performance)
    - Number of errors occur at a specific point in time
    - Purpose reveals quality problems
  - Metric: Failed deployments
    - Track failed deployments over time
    - Also known as mean time to failure (MTTF)

# Metrics for measuring DevOps

- For Velocity
  - Metric: Lead Time
    - The number of time that occurs from start till deployment
    - Purpose: measure the average time length of a work item
  - Metric: Automated tests pass %
    - Discover how well the automated tests works by measuring how often code changes that leads to testing failure.
    - Purpose: helps to increase velocity
  -

# Metrics for measuring DevOps

- For Performance
  - Metric: Application performance
    - Measure performance problems, hidden errors, issues
    - Purpose: find changes in overall application performance before executing deployment
  - Metric: Mean time to detection (MTTD)
    - Measure and detect unknown problems
    - Purpose: identifying issues in order to fix it as soon as possible
  - Metric: Mean time to recovery (MTTR)
    - Track business hours it takes to recover from failures
    - Purpose: to minimize performance errors and failures.



# Cost Effectiveness of DevOps

It is important to increase the cost effectiveness of DevOps Testing Automation. The following are three ways to achieve this goal.

- Build Test Cases to isolate elements
  - Finding the error when test case fails would be easier when a test case is identified with a specific element.
- Make Independent Test Cases
  - Independent test cases can be executed at the same time, therefore, increasing productivity.
- Choose the Right Testing Automation Tool
  - A good tool would have functionalities that supports all test cases.



# DevOp Tools

(for a Developer in 2019)



# DevOp Tools

DevOp tools usually involves tasks that focus on the deployment and release of code. Whether it's updating or introducing a new system, DevOp tools have to monitor and configure multiple aspects of a live application.

The main aspects are

- Operating Systems
- Server Management
- Infrastructure



# DevOp Tools - Operating Systems

Process Management, Threads and Concurrency - managing processes within systems such as Linux and Windows in order to facilitate threads

- Linux Terminal is a popular way of managing threads and processes
- `top` is a popular way of managing Linux processes
  - `ps` - list system processes
  - `kill` - terminates process
  - `jobs` - list user processes
  - `bg` - places process in the background
  - `fg` - places process in the foreground
- Window's bash has comparable commands, and now also supports Linux environments with *HyperVisor*



# DevOp Tools - Operating Systems

Sockets, I/O Management - this concerns the mechanics behind input, output, network ports and sockets.

- There are typically packages and API for managing networks for code, like Java's `java.net` and `java.io`
- For system administration, there are terminal commands such as `ifconfig` and `nc` to manage and setup sockets and ports



# DevOp Tools - Servers, Proxies

Reverse and Forward Proxies, Firewalls

- These handle responses between clients and servers

Load Balancers, Caches

- For facilitating and optimizing requests and heavy traffic

## DevOp Tools - Web Servers

For Web Development, managing web servers is another task for DevOps personnel.

Although there are frameworks such as Rails and Django, these package essential web server tools - particularly `Apache` and `Nginx` which balances between performance, features, and compatibility



**Apache Web Server**

**NGINX**

# DevOp Tools - Infrastructure

Infrastructure concerns itself with tools and platforms that handle the deployment of code

Containers - these tools contain self-contained virtual environments for deployed code. Virtual allows machines and hardware to receive a streamlined and configured environment that can be scaled and tested. The most popular container thus far is Docker







## DevOp Tools - Infrastructure

Configuration Management - These tools automate storage, server, and network settings. Powerful in conjunction with containers and deployment to machines. A popular and simple tool is Red Hat's Ansible.




Red Hat

Ansible

# DevOp Tools - Infrastructure

Configuration Orchestration - These tools manage automated system deployment and scaling. Unlike Docker, these tools focus on coordinating deployed applications at scale. A popular web development tool is `Kubernetes`





# DevOp Tools - Infrastructure

Infrastructure Monitoring - These tools watch and display information about the system's networks, data, and other tangible artifacts associated with the system. Popular tools include `Datadog`, `Monit`, and `Zabbix`.

Application Monitoring - These tools watch and display information on the actual application. Tools like `New Relic` give a detailed metric for system performance and real-time information.



# DevOp Tools - Integration and Deployment

Continuous Integration/Deployment - these tools enable easy testing and deployment of code. An aspect of Agile is constant releases of software and updates to a system. Tools such as `Jenkins` and `Circle CI` perform tests immediately upon updates, automatically updating a live application if the changes are successful.

Providers - providers are the actual hosts for server and application code, that allows the public to interface and visit the website. Popular providers are `Amazon's Web Services`, `Digital Ocean` and `Heroku`.



# Case Study

In researching for a relevant case study, this recent example of a successful DevOps process is elicited from this article from *Information Age*:

<https://www.information-age.com/devops-case-study-banks-123482580/>

The author is a DevOps specialist for Brickendon, a tech consulting agency. During his tenure, he implemented DevOps successfully to one of the largest banks and their systems. The benefits he claims are:

- Savings of almost US\$20 million
- 45% Increase in number of releases



## Case Study

The DevOps specialist developed a unified team with flexible skills, while adopting smaller releases and modern practices. The new DevOps strategy increased accountability, and work can be better prioritized.

These led to automation for quality assurance, deployment, and continuous integration. The rework in methodologies and culture led to higher-quality software and significant financial savings.