



CICD

Continuous Integration
Continuous Deployment

Ahmed Aldomany



Overview

Continuous Integration

The practice of merging all developers' working copies to a shared mainline several times a day.

It's the process of "Making". Everything related to the code fits here, and it all culminates in the ultimate goal of CI: a high quality, deployable artifact!

Continuous Deployment

A software engineering approach in which the value is delivered frequently through automated deployments. Everything related to deploying the artifact fits here. It's the process of "Moving" the artifact from the shelf to the spotlight



Benefits of CICD

- 01 Automate Infrastructure Creation and clean up: Eliminating human errors and avoid unnecessary cost of unused or invalid infrastructure
- 02 Faster to production: By automating the pipeline to production this way we can deploy features as soon as created which will help increase revenue
- 03 Automated Rollback Triggered by Job Failure: Automate the process of rolling back and cleaning any infrastructure left which would help in reducing cost and lower down time

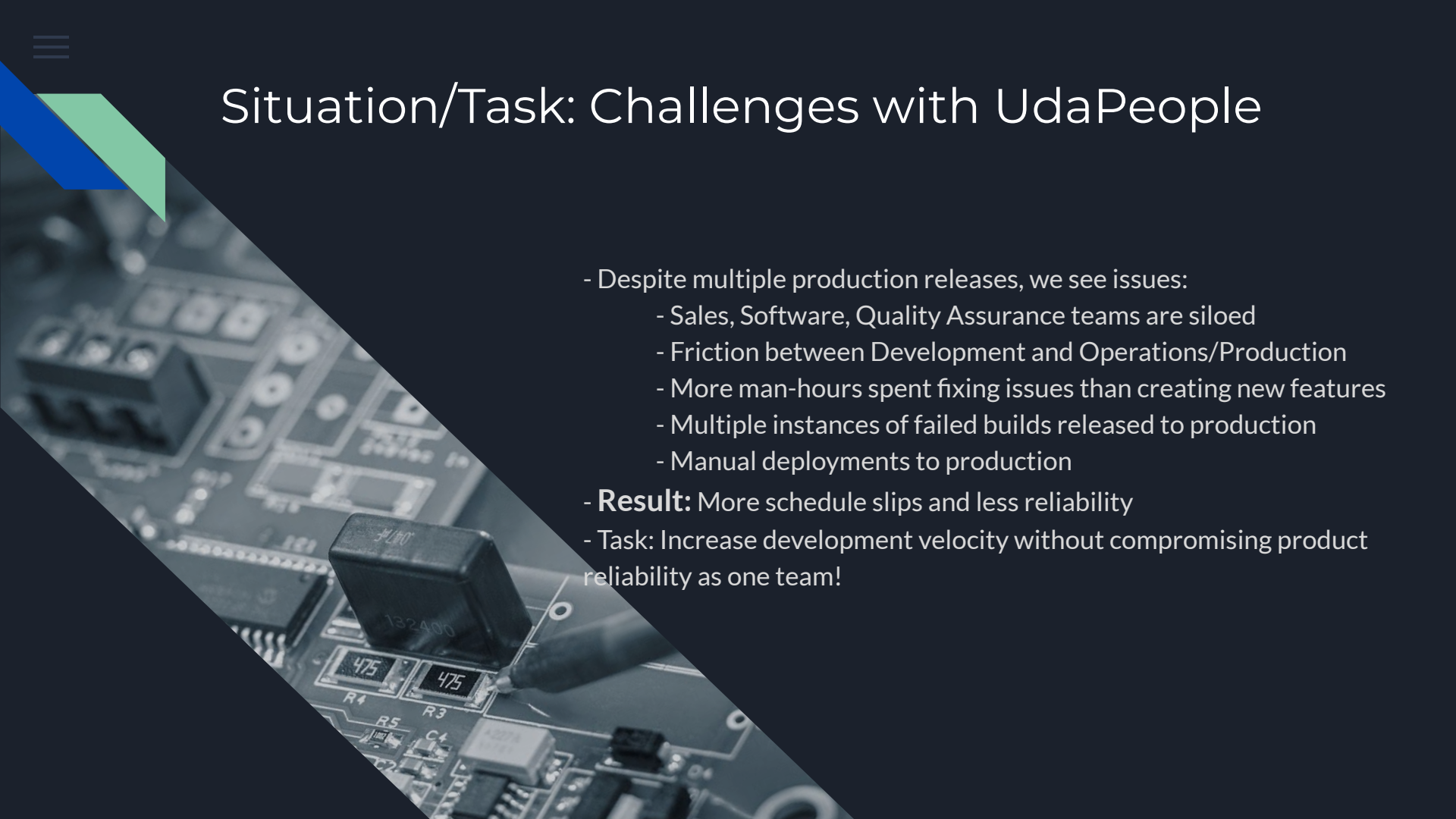


Benefits of CICD

- 04 Catch Compile Errors After Merge: Discover errors as soon as the developer make his commit which will help reduce the time of developers and reduce cost
- 05 Catch Unit Test Failures: Unit tests are not neglected with CICD which will increase code quality and catch errors early before production which would decrease cost
- 06 Automated Smoke Tests: Automate smoke test after deployment and automatic rollback in case of failure which will decrease downtime and reduce cost



Situation/Task: Challenges with UdaPeople

- 
- Despite multiple production releases, we see issues:
 - Sales, Software, Quality Assurance teams are siloed
 - Friction between Development and Operations/Production
 - More man-hours spent fixing issues than creating new features
 - Multiple instances of failed builds released to production
 - Manual deployments to production
 - **Result:** More schedule slips and less reliability
 - **Task:** Increase development velocity without compromising product reliability as one team!

Action: Creating a CI/CD Pipeline

Proposed Tools -

CI/CD: CircleCI

- Configuration Management: Ansible

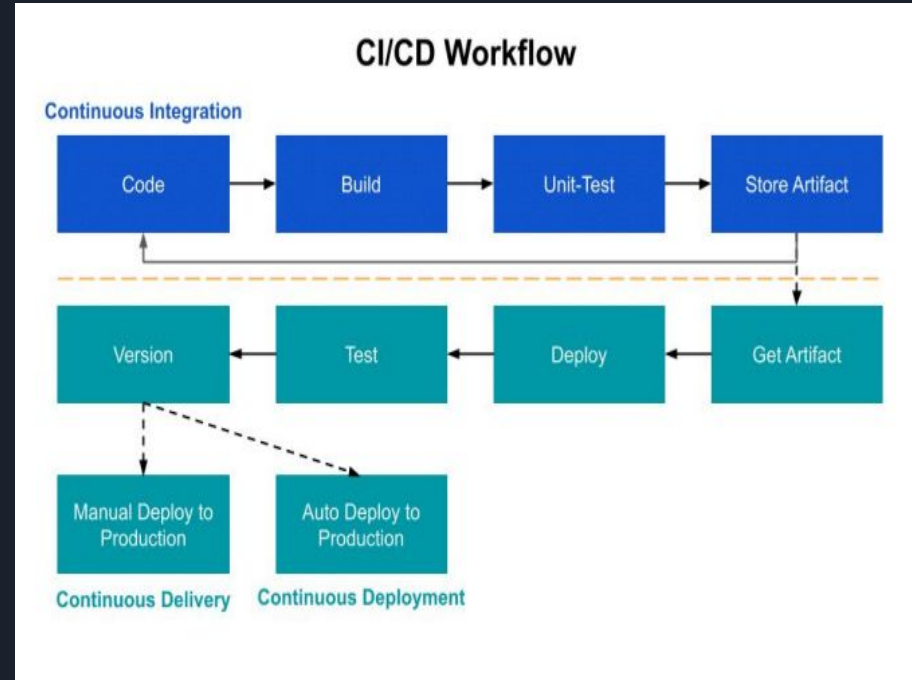
- Containers:

Docker - Container Orchestration: Kubernetes

- Infrastructure Provisioning:
AWS CloudFormation

- Monitoring:
Prometheus/Grafana

- Cloud Provider:
Amazon Web Services



Expected Results: Business Implications

Business Value	How CI/CD achieves this	Result:
Reduce Cost	- Catches Compile Errors After Merge - Automate Infrastructure Cleanup	- Less developer time on issues from new developer code - Less infrast
Avoid Cost	- Catches Unit Test Failures - Detects Security Vulnerabilities - Automates Infrastructure Creation	- Less bugs in production/testing - Prevent embarrassing or costly security holes - Less human error, faster deployments
Increase Revenue	- Faster and More Frequent Production Deployments - Deploy to Production Without Manual Checks	- New value-generating features released more quickly - Less time to market
Protects Revenue	- Automated Smoke Tests - Automated Rollback Triggered by Job Failure	- Reduced downtime from a deploy-related crash or major bug - Quick undo to return production to working state