POWERING POTENTIAL

DevOps as a Culture

WHAT IS DEVOPS?

- *Cultural* approach to software development project structure with a particular philosophy designed to achieve the following:

  → Increased collaboration

  → Reduction in silos

  → Shared responsibility

  → Autonomous teams

  → Increase in quality

  → Valuing feedback

  → Increase in automation

HOW THINGS USED TO BE DONE

- Software companies were structured into separate, stratified teams:
  - → Development
  - → Quality assurance (testing)
  - → Security
  - → Operations
- Teams tend to have varying and conflicting goals
- Often poor communication
- Isolated teams are referred to as *silos*
- This structure regularly results in:
  - → Slower releases
  - → Wasted time and money
  - → Blame cultures

HOW DEVOPS CHANGES THINGS UP

- Based on agile project management
  - → Designed to encourage flexible teamwork with the ability to fail (and recover) fast and celebrate achievements to promote a productive work culture
- Agile focuses on bridging the gap between developers and customers

HOW DEVOPS CHANGES THINGS UP

- DevOps focuses on bridging the gap between developers and operations teams

  → Historical friction between the developers and operations teams

  - Developers would generate code that broke the applications
  - Operations would throw code back to developers without sufficient details

  → Causes slower release times, inability to focus on their primary responsibilities, and general frustration within the organisation

AUTOMATION

Key Agile Principles – Simplicity
(Make your job easier)

- Rule of thumb: if a machine *could* do it, a machine *should* be doing it

- Manual work:

  → Human error
  → Slower development
  → Slower deployment

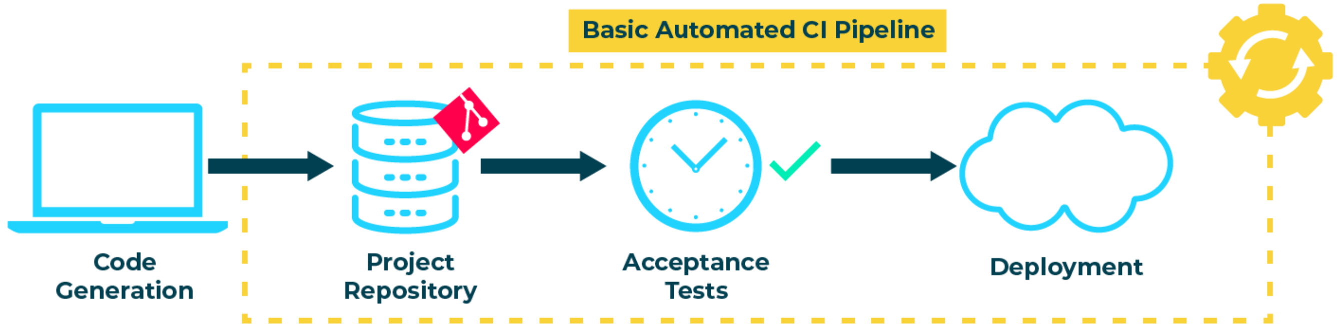
- Automated work:

  → Consistent
  → Faster
  → Predictable
  → Scalable – Can be copied to
  Create more access

AUTOMATION

CI Pipeline
Continuous Integration / Improvement

Automating the process of adding new features / developments into our codebase



CONTINUOUS INTEGRATION

- When code is committed to a repository, it is automatically built and subjected to acceptance tests (Mainly testing if the new code breaks the rest of the application)

- Test failures result in the code being prevented from integrating with the repository. Developers are immediately notified of a test failure so they can fix issues as quickly as possible

CONTINUOUS DEPLOYMENT / DELIVERY

- As new code passes acceptance tests, it is automatically integrated into a deployment environment
- Being able to choose a version to deploy with one push a button requires a fair amount of automation

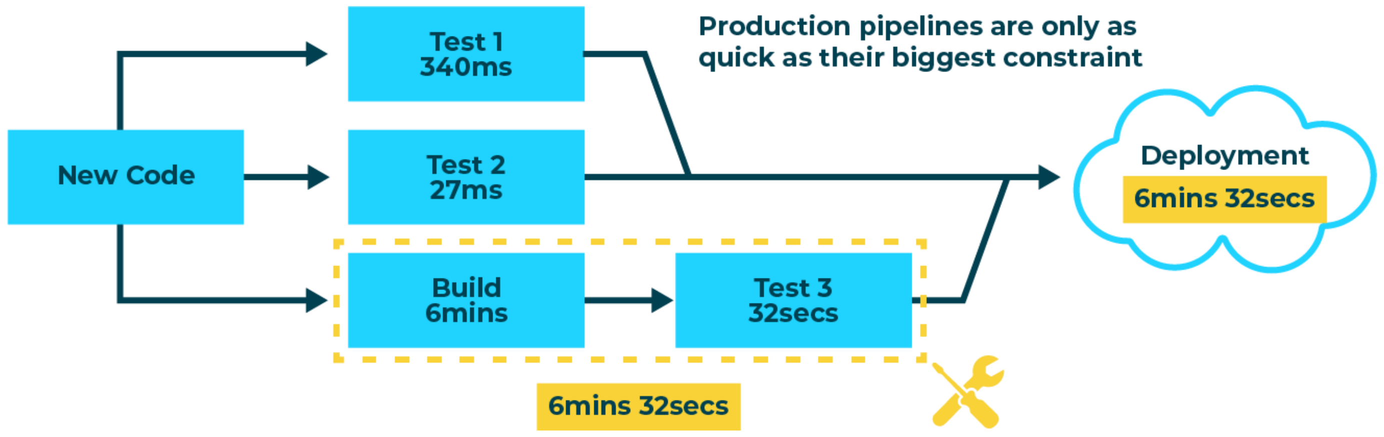
INFRASTRUCTURE AS CODE

- IaC is used to specify the configuration of a computer environment with easy-to-write/read config files
- Having environment infrastructure declared in code allows for infrastructure to be created or modified using version control
- Allows for simple replication of environments so they stay consistent across the pipeline

MEASUREMENT

- Accurate and precise measurements allow us to pinpoint constraints in the pipeline and fix or improve them faster
- Also important from a cultural standpoint as they can inform teams whether they're working more productively and what can be done to improve
- We use metrics to measure our pipelines

MEASUREMENT


New Code
Test 1
340ms
Test 2
27ms
Build
6mins
Test 3
32secs
6mins 32secs
Production pipelines are only as quick as their biggest constraint
Deployment
6mins 32secs