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

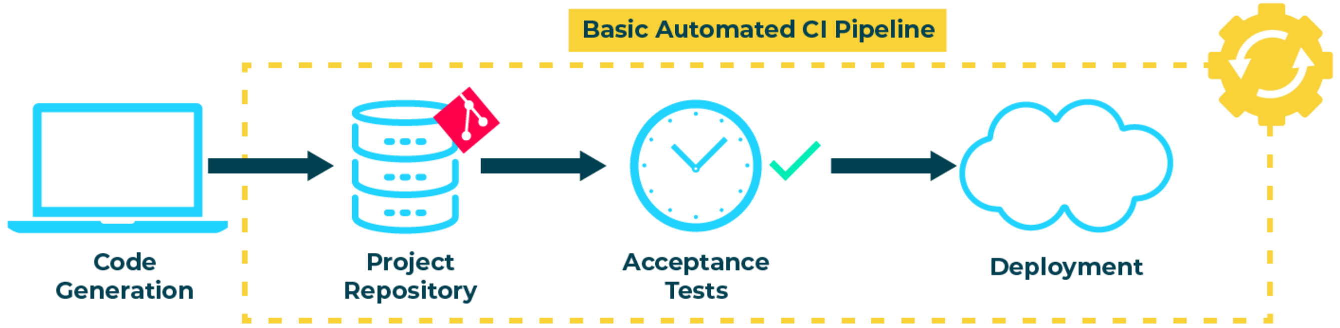
  - **Developers would generate code that broke the applications**
  - **Operations would throw code back to developers without sufficient details**

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

## AUTOMATION

- 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

# AUTOMATION


Basic Automated CI Pipeline
Code Generation
Project Repository
Acceptance Tests
Deployment

## CONTINUOUS INTEGRATION

- When code is committed to a repository, it is automatically built and subjected to acceptance tests
- 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

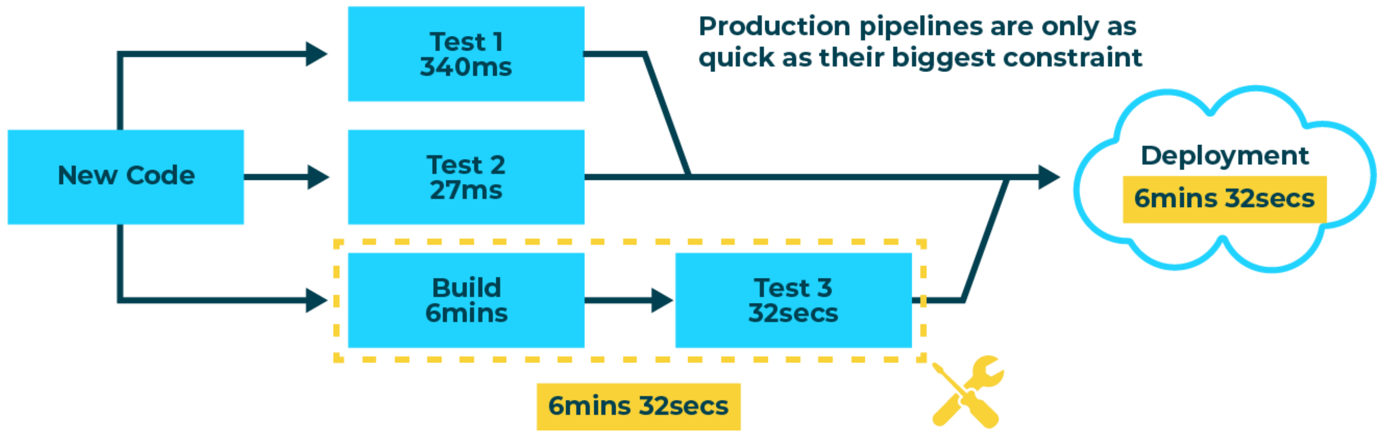
## INFRASTRUCTURE AS CODE

- 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



## MEASUREMENT

- **Frequency of deployment**
  - → DevOps pipelines encourage frequent, smaller updates to software, so charting the frequency of deployments is a good indicator of the effectiveness of a pipeline
  - → Deployment frequency should tend upwards until it reaches a natural plateau, though fluctuation is normal
- **System availability**
  - → Systems should be available at all times to customers
  - → Knowing the availability of our systems allows us to pinpoint which parts of our infrastructure need attention

## MEASUREMENT

- **Service performance**
  - → Allows us to see whether our services are running within desired thresholds
  - → e.g. response times per request, CPU load, how long it takes for a website to load
- **Mean time to recovery (MTTR)**
  - → Average time it takes to solve problems that impact the end-user
  - → e.g. outages, security issues, severe bugs
  - → More worthwhile metric than charting the frequency of failures as DevOps is less interested in minimising problems than the speed at which they are solved

# MEASUREMENT

- **Mean time to discovery (MTTD)**
  - → This refers to how quickly problems are discovered
  - → The faster problems are identified, the faster they can be fixed
  - → Measured from the point of integration into production to the point the problem is identified
  - → Faster MTTDs are more desirable.
  - → Should also indicate whether discovery is made by the customer or the automated systems, with the latter being more desirable