POWERING POTENTIAL

DEVOPS AS A CULTURE

1

- ***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

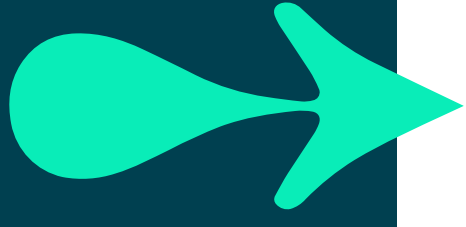
2

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

3

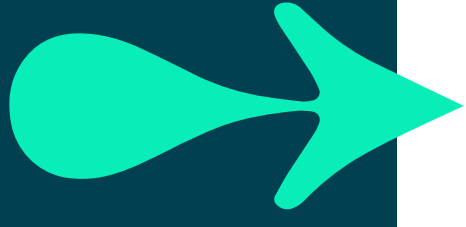
HOW DEVOPS CHANGES THINGS UP



- 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

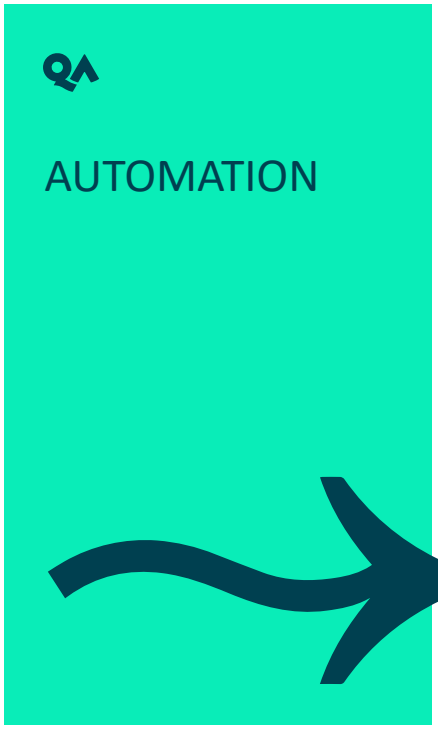
4

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

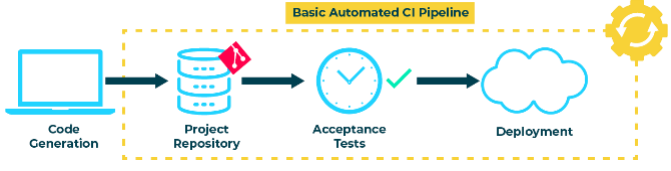
5



- 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

6

AUTOMATION



7

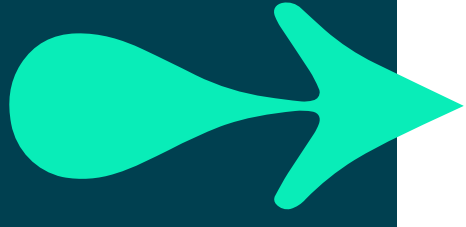
- When code is committed to a repository, it is automatically built and subjected to acceptance tests
- Test failures result in the code being prevented from integrating with the repository. Developers are immediately notified of a test failure so they can fix issues as quickly as possible

8

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

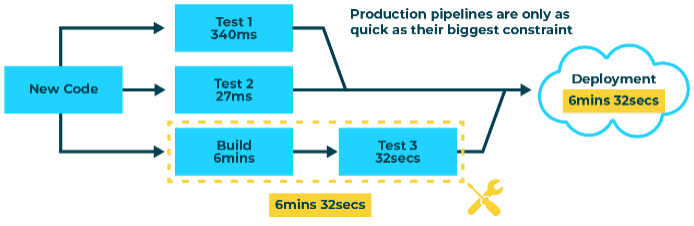
10

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

11

MEASUREMENT



12

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

13

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

15

POWERING POTENTIAL

CHAPTER 1
SDLC AND AGILE RECAP

16

Agile Principles

Highest priority is to satisfy customer through early and continuous delivery of working software

Welcome requirement changes during development which enhance competitive advantage

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale

Business people and developers must work together daily throughout the project

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation

Agile Principles

Working software is the primary measure of progress

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely

Continuous attention to technical excellence and good design enhances agility

Simplicity – the art of maximising the amount of work not done – is essential

The best architectures, requirements, and designs emerge from self-organising teams

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly

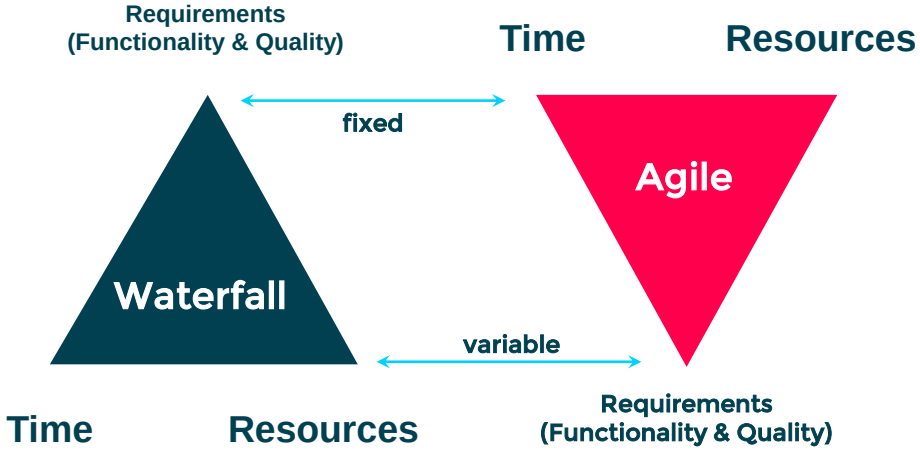
Traditional vs Agile

Traditional	Agile
Low-Level requirements are defined up front	Expects that low-level requirements evolve through understanding
Delivers fully developed system	Delivers increments of system in an iterative manner
Contingency in resource	Contingency in prioritised requirements
Typical user involvement at start and end of project	Planned active user involvement throughout the lifecycle

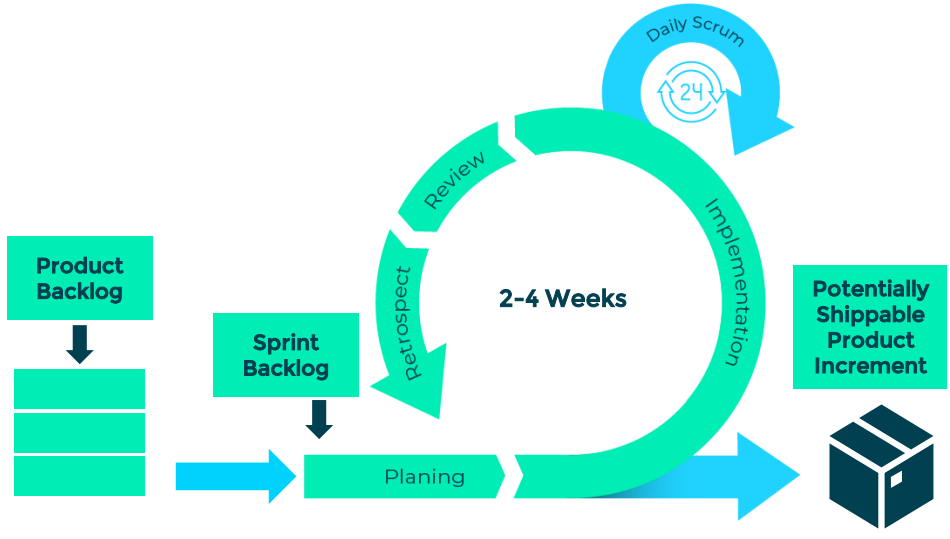
Traditional vs Agile

Traditional	Agile
Discreet testing phase at end of project	Testing Integrated throughout lifecycle
Dispersed teams	Prefers co-located teams
Skills and roles related to stages	Skills and roles used across lifecycle - no hand-offs
Activity based planning	Product based planning

Turning Convention Upside Down

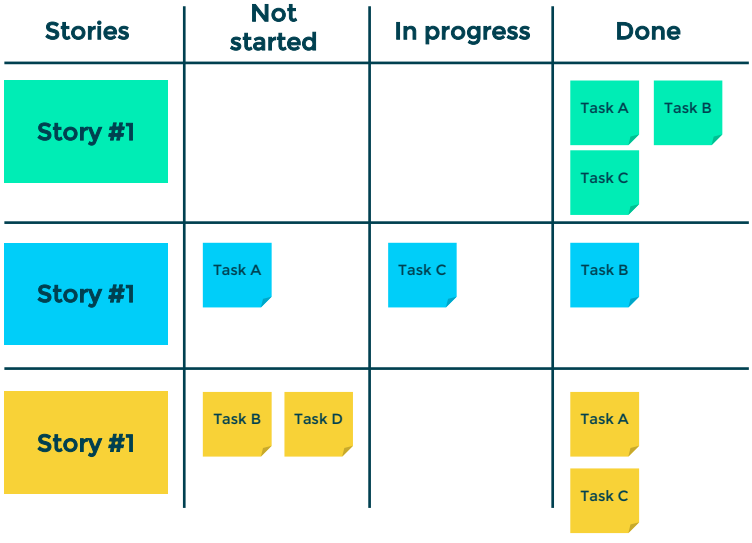


Scrum Process Overview

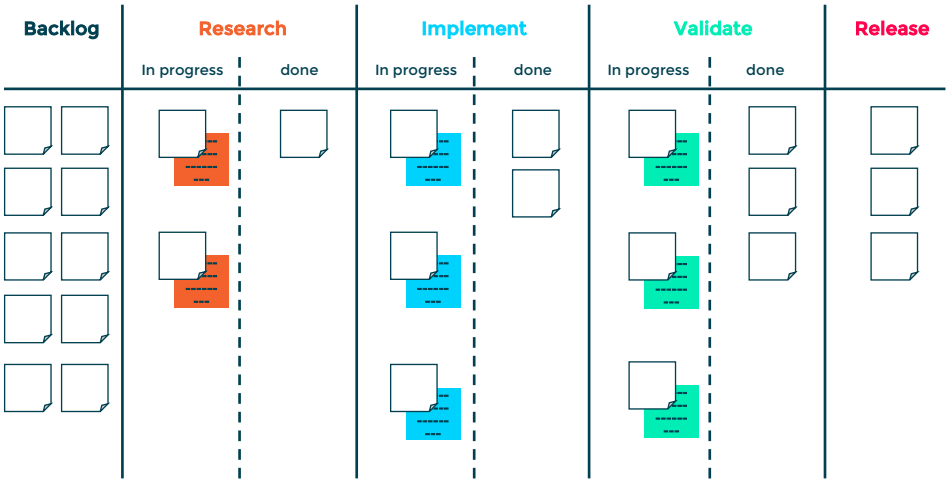


22

Scrum board


Stories
Not started
In progress
Done
Story #1
Task A
Task B
Task C
Story #1
Task A
Task C
Task B
Story #1
Task B
Task D
Task A
Task C

Kanban Boards

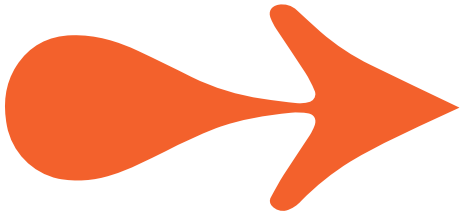


Agile Roles

Product owner – custodian of the vision for the product, represents the customer

Scrum Master – helps the team best use Scrum to build the product, shields the team from outside influence

Development team – build the product

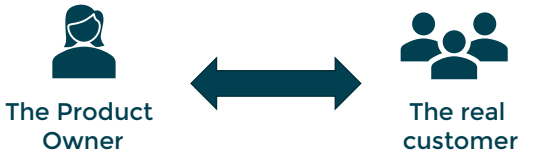


Product Owner

The voice of the customer, acting as an expert to the development team

Defines the work to be done, and prioritises that work

Understands what's important to the customer



Customers can be a very varied group of people

26

Scrum Master

Helps the team best use Scrum to build the product

Helps them to focus on the project

- Protects them from organisational disruptions and internal distractions like an area where the teach cannot focus

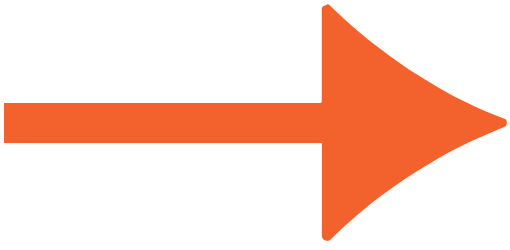
Ensure that the team execute the Scrum process in spirit

- They control the process, not to be confused with the role of the project manager who is responsible for the workload

27

Development Team

- Self-organised
- Work with Scrum master to develop the product
- Prioritise the product backlog
- Define the number of sprints
- Define the stories to be tackled in each sprint
- Define DONE



28

Story Points NOT Time

Traditionally when prioritising and scheduling tasks, analysts would use Complexity and Time as key factors

This has proven to be a unsatisfactory way of prioritising and scheduling workloads

Recommended approach – story points

- A notional value assigned to a task that represents its complexity and perceived time to deliver

Techniques for assigning story points

- Estimation Poker - https://www.planningpoker.com/

29

Agile project – pool party

- **Brief to follow…**
- **2 x sprints**
- **Follow as much of the agile process as possible**
- **Deliverable (each sprint) is event proposal document**
- **Think about roles, product backlog, sprint backlog, tracking etc…**