Welcome

Welcome

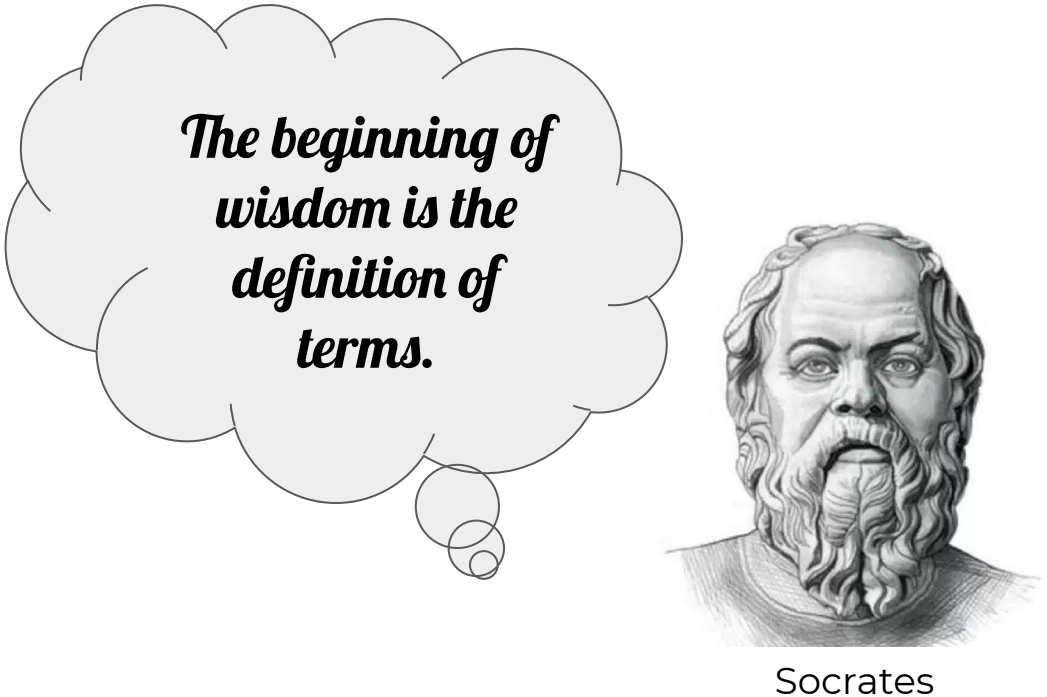
Who am I?

- Eng. Simon Takite Kiwanuka - MSc. Information Systems NTNU - Trondheim, Norway
- TechLead, DevOps Consults (U) Ltd
- Senior Consultant
- DevOps evangelist

Linked in: https://www.linkedin.com/in/simon-takite-38206755/
Github: https://github.com/simontakite
Website: http://simontakite.github.io/

Agenda

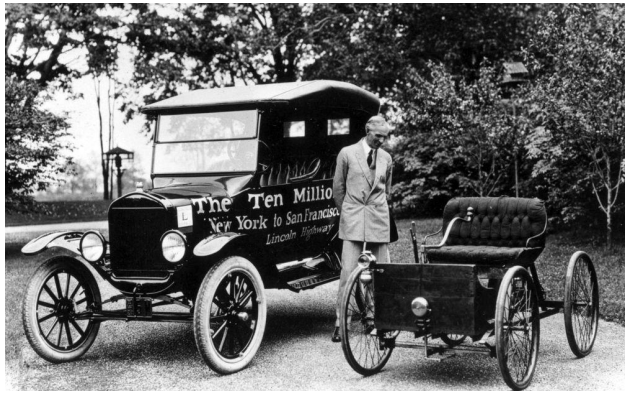
- Who is Dev?
- Who is Ops?
- What is DevOps?



Socrates

Traditional Development

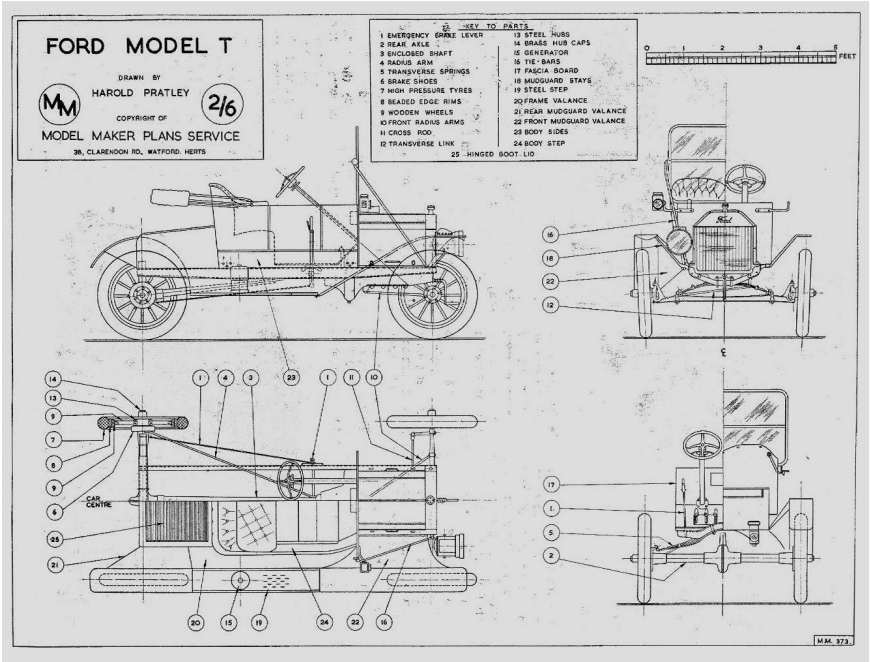
- The inventor Vs. The mechanic



Introduction to DevOps

The Inventors

- Create new features and functionality in “dev” environment
- Occasionally deliver new product to operators, along with instructions
- May incorporate feedback from operators in future deliveries
- Rewarded for delivering new features



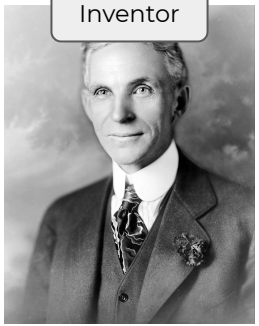
The Mechanics

- Receive new product from developers to be installed and operated
- Expected to keep cars up and running
- Track problems, deployment failures, and system outages
- May provide feedback to the inventors for future consideration
- Penalized for downtime

The mechanics are responsible for keeping the system (car) in operation

Differing Views on Change

Inventor



Mechanic

Hero

Obstacle

Change oriented

Stability oriented

Alienate customers b/c system constantly changes

Logical extremes

Alienate customers b/c system doesn't change

We Have A LOT of Changes in IT.

Can we deploy latest version?

Can you deploy this one small change

Can you deploy new patch for release

Can we apply this security path

Can you stage this new environment

Can you Upgrade the operating system

Production is down, fix it now!

Can you Upgrade the database version

Prod is running slow, can you cycle the server

Separation of Dev and Ops: A History

As computers became more complex, dev and ops became necessarily specialized:

- Accelerating pace of technology
- Increased demand for turning around new features
- Huge amounts of data and number of calculations
- More and more specialized tools Increasingly abstract architectures and design patterns

And these were the problems in 1945!

The wall of confusion

Development

Operations

Extreme focus on change

- More urgent, date/driven projects
- Fragile code going into production
- Releases with turbulent installs
- Longer release cycles amortize
- Backlog of infrastructure projects that could fix root cause and reduce costs

Extreme focus on stability

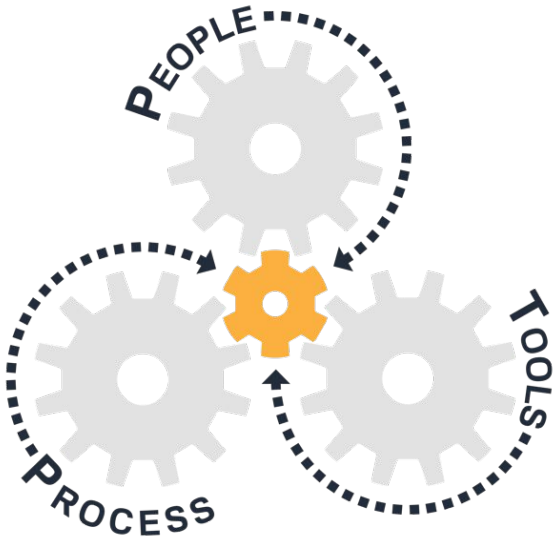
- Fragile applications failing
- Difficulties identifying root cause
- Taking too long to restore service
- Extensive firefighting and unplanned work
- Unfinished planned project work
- Frustrated customers
- Market share decreasing

The Reunification of Dev + Ops

What is DevOps?

A compound of development (**Dev**) and operations (**Ops**), DevOps is the union of people, process, and technology to continually provide value to customers.

DevOps enables formerly siloed roles—development, IT operations, quality engineering and security—to coordinate and collaborate to produce better, more reliable products.



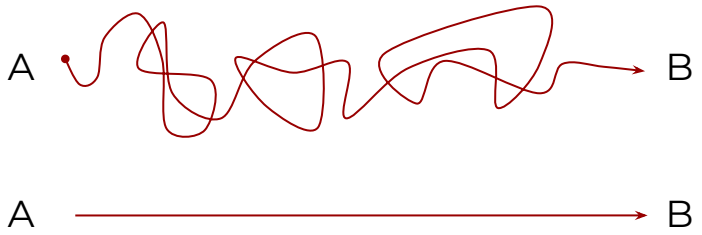
https://docs.microsoft.com/en-us/devops/what-is-devops

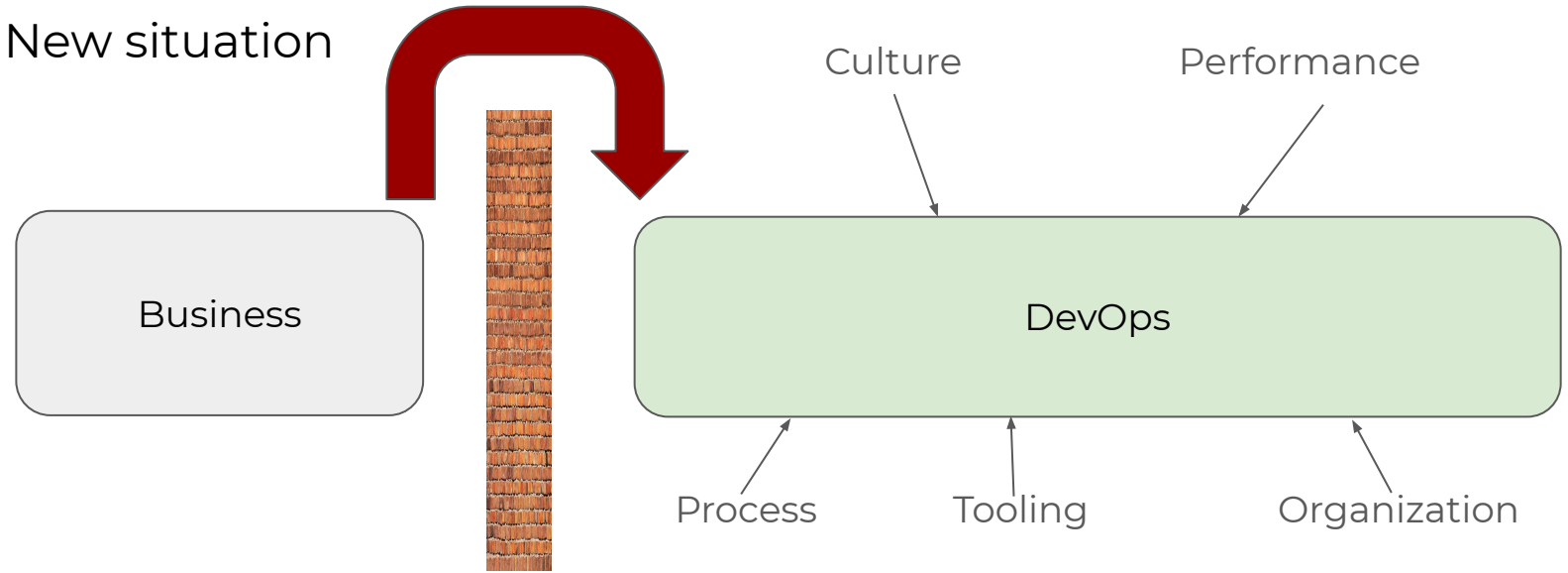
What is DevOps?

DevOps is about identifying bottlenecks and removing the waste from the system to continuously improve your organization to deliver better software.

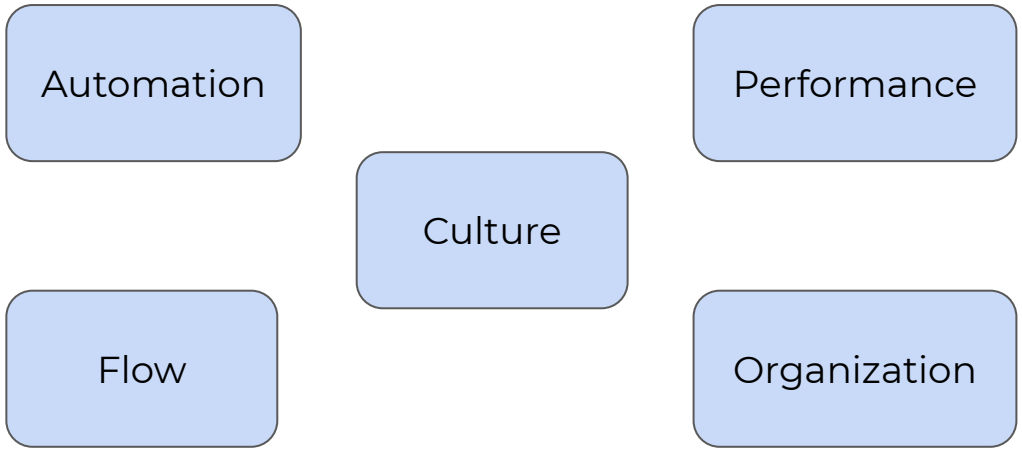
Bottlenecks can be in the form of long approval processes, manual processes, and even certain resources.

The most fundamental goal of DevOps - Removal waste.




New situation
Business
Culture
Performance
DevOps
Process
Tooling
Organization

DevOps is the solution to a problem



What are the points of view of DevOps?

Organization

- From functional silos
- To multidisciplinary customer-focused teams

Flow

- From complex and slow
- To end-to-end and simple

Performance

- From different KPIs for dev and ops
- To common goals and measures

Automation

- From own tools
- To integration tooling

Culture

- From different KPIs for dev and ops
- To common goals and measures

DevOps vocabulary

Continuous Delivery (CD) is a software strategy that enables organizations to deliver new features to users as fast and efficiently as possible. The core idea is to create a repeatable, reliable and incrementally improving process for taking software from concept to customer. The goal is to enable a constant flow of changes into production via an automated software Continuous Delivery pipeline.

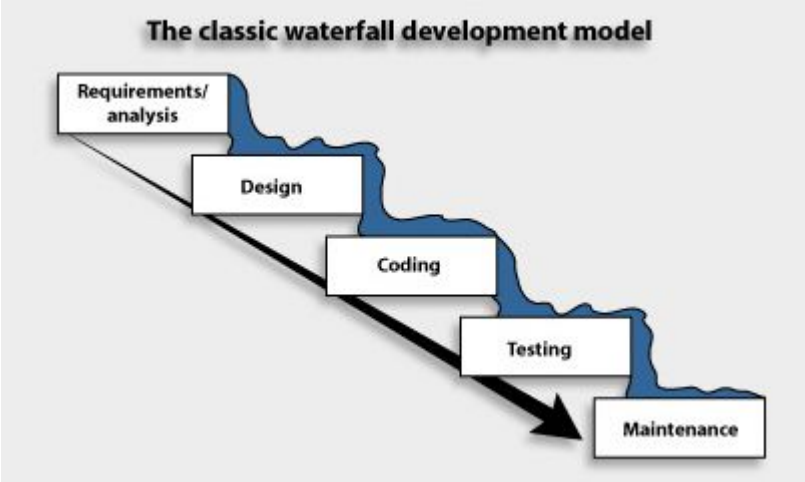
Continuous integration (CI) is the practice, in software engineering, of merging all developer working copies to a shared mainline several times a day. The main aim of CI is to prevent integration problems

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test, and finally refactors the new code to acceptable standards.

Test automation is the use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes

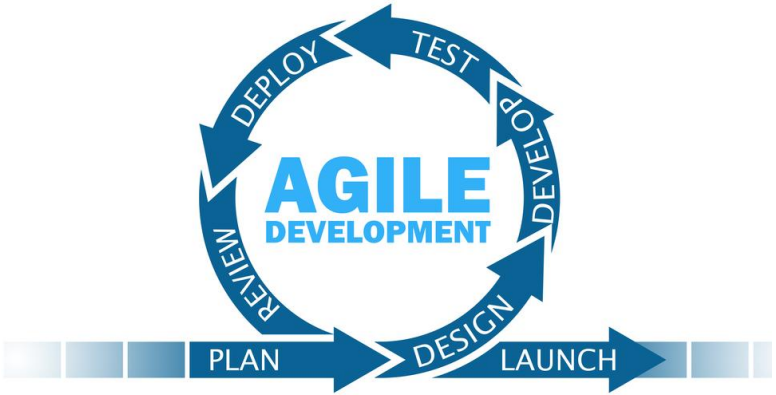
Other frameworks: Waterfall

1. Determine requirements
2. Complete the design
3. Do the coding and unit testing
4. Perform functional and integration tests and fix bugs
5. Perform acceptance testing and deploy

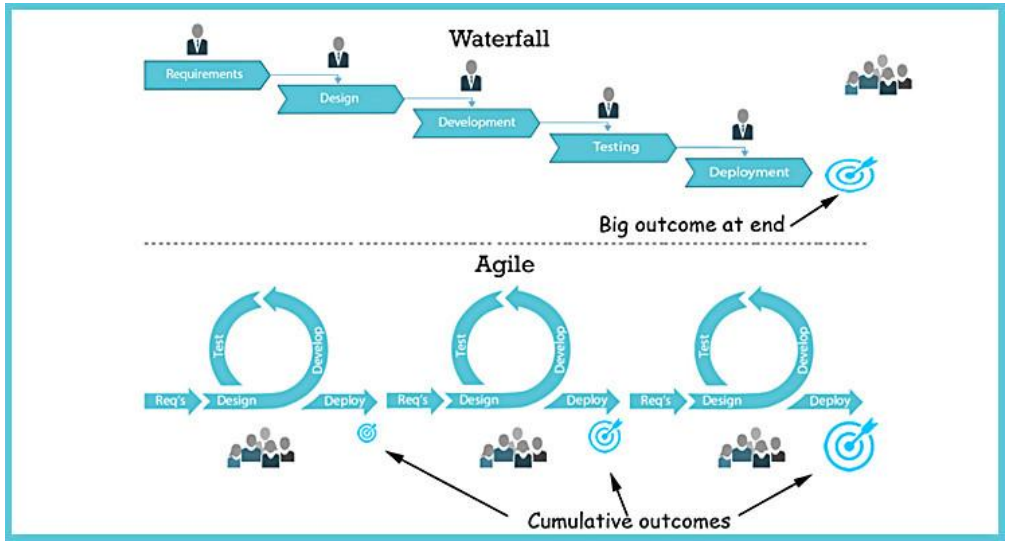


Other frameworks: Agile/Scrum

1. Every Sprint delivers working software that can be used in practice
2. Starts with delivering basic functionality to which features are added
3. Value-driven



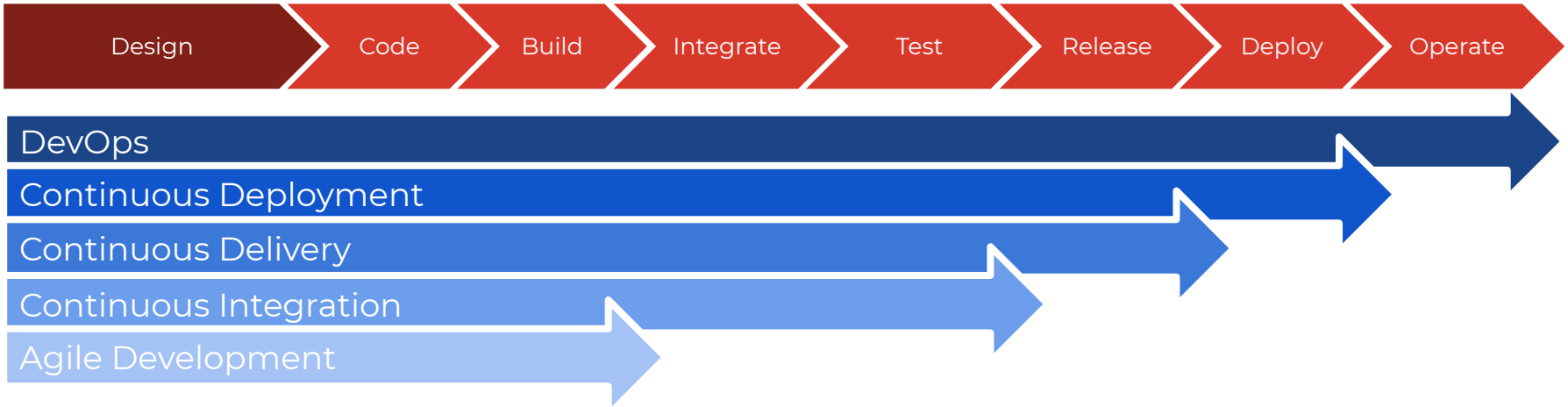
Waterfall vs. Agile development



Traditional software development vs. DevOps

Traditional	DevOps
Focus on ship date	Focus on working software
Prolonged deployments	More frequent releases
Heroic efforts	Repeatable & predictable processes
Handoffs	Feedback loops
Done when code is built & compiled	Never done - Always

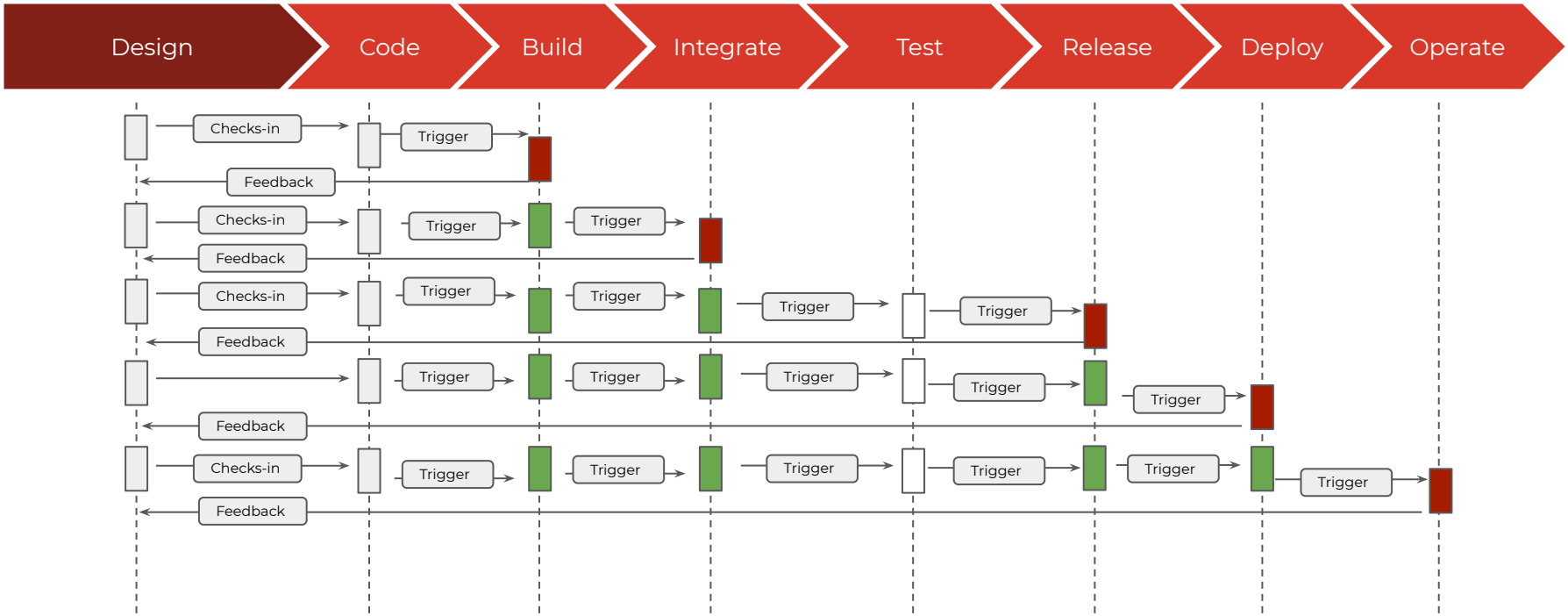
Evolution of DevOps implementation



DevOps covers the full deployment pipeline

- Treating infrastructure as code by programmatically provisioning and managing infrastructure resources
- Repeatable and reliable deployment processes
- Development and testing (preferably automated testing) performed against production-like systems
- On-demand creation of development, test, staging and production environments
- Proactive monitoring of infrastructure components, environments, systems and services

Engineering flow of continuous delivery



What problems do we solve:

- Speed of delivery
- Quality of delivery
- Lack of understanding between developers and operations people
- Ability to ensure that customer requests are seamlessly processed
- Bring people who work on the same IT service closer together
- Preventing burnouts

DevOps Tooling

Questions?