System Analysis and Design

DevOps

By: Vahid Rahimian

Spring 2022

Agenda

- What is DevOps?
- DevOps and Agile
- Version Control and Automation
- CI / CD
- DevOps Tools
- DevOps Topologies

What is DevOps?

Why DevOps?

- To reduce deployment lead time to minutes!
- The business demands faster and continuous delivery.
- Most organizations are not able to deploy production changes in minutes or hours, instead requiring weeks or months.

- Opposing goals between Development and Operations:
  - Conflict between agile development (urgent projects) and stable operation (keep it running, don’t mess with the environment)

Shared Devs and Ops Values

- Value collaboration on all aspects of the system
- Code and infrastructure/configuration
- Solve issues early and quickly
- Have a production-first mindset
- Version control everything
- Automate everything (esp. manually intensive tasks)
- Create small, frequent deployments (of code and configuration)
- Monitor, log and validate performance obsessively

Typical problems between Dev and Ops

Organizational silos

Different mindsets

Different tools

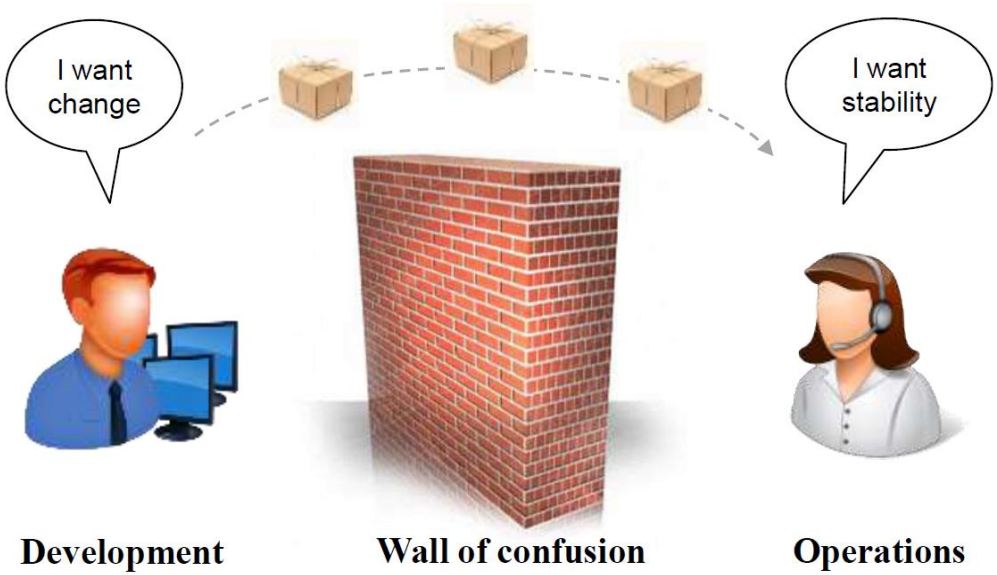
Different environments

Product back-logs

Blame game

Disintegrated processes

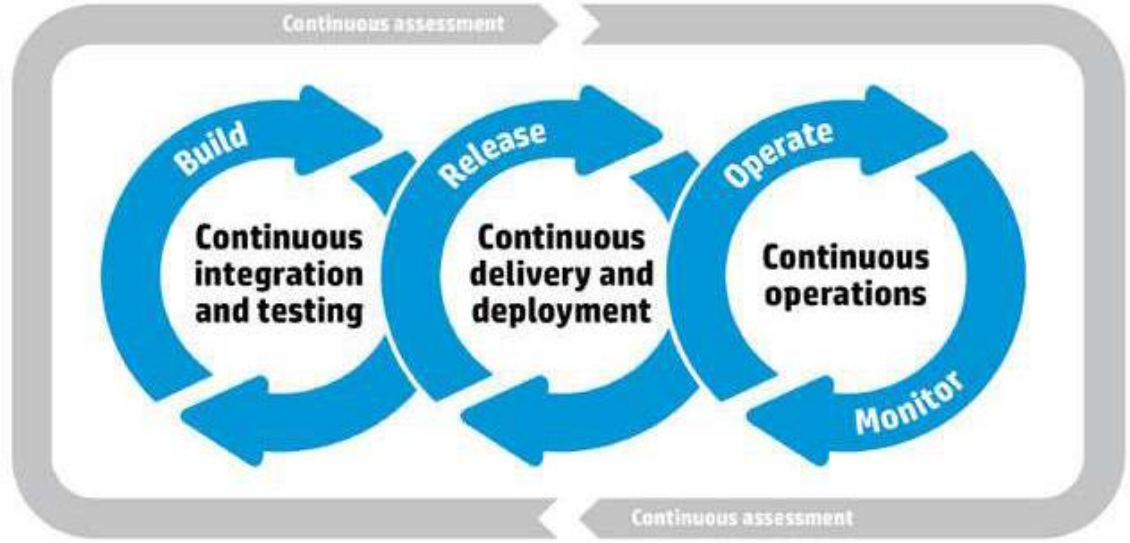
Poor feedback loops



Leads to a downward spiral: Everybody gets a little busier, work takes a little more time, communications become a little slower, and work queues get a little longer. Work requires more communication, coordination, and approvals.

What is DevOps?

- Tear down the wall between Development and Operations:
  - Continuous Delivery: Continuous Integration, Testing and Deployment



What is DevOps?

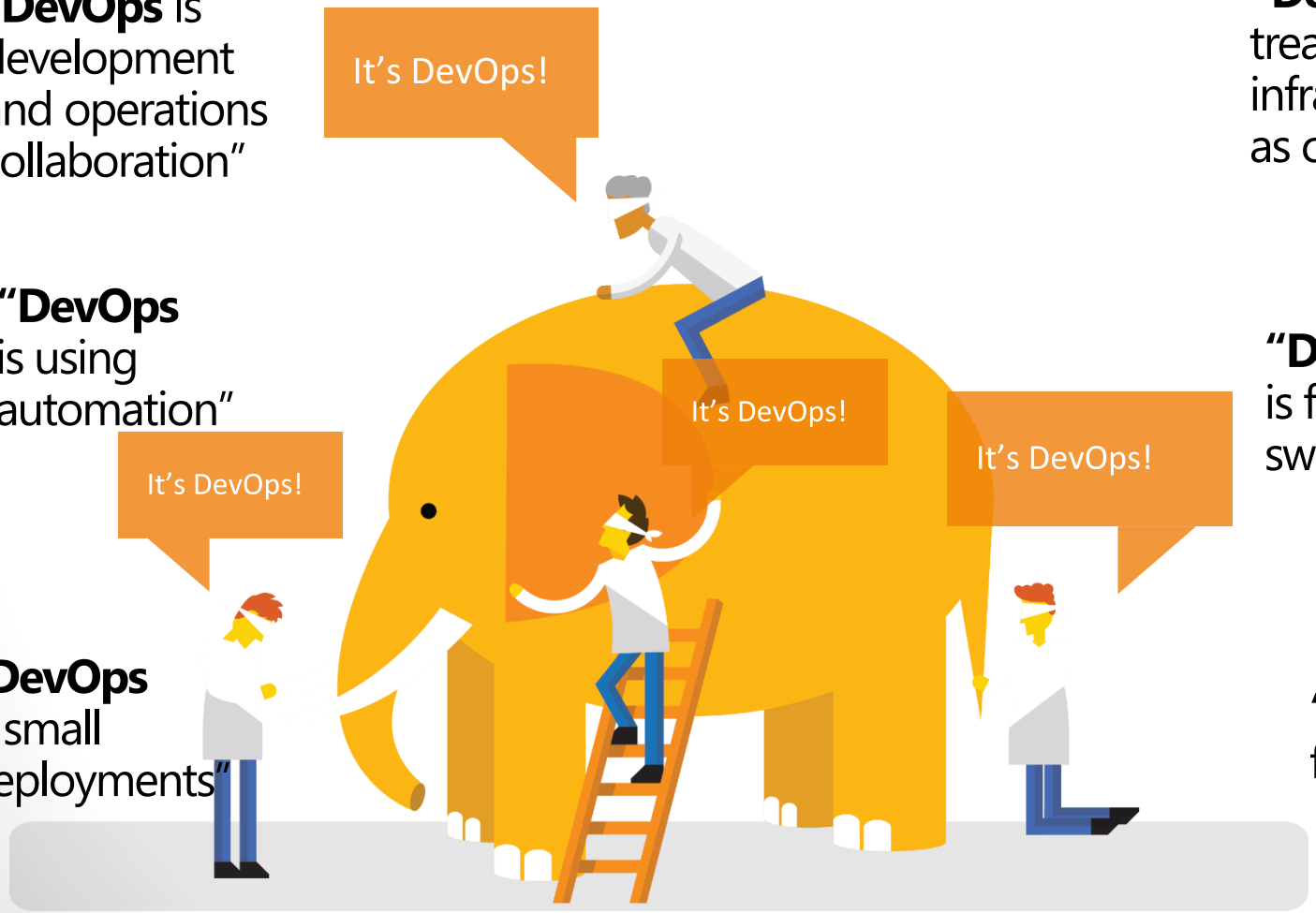
- Small teams independently implement their features, validate their correctness in production-like environments, and have their code deployed into production quickly, safely and securely.”

What is DevOps?

"**DevOps** is development and operations collaboration"

"**DevOps** is using automation"

"**DevOps** is small deployments"



"**DevOps** is treating your infrastructure as code"

"**DevOps** is feature switches"

"Kanban for Ops?"

Definition of DevOps

"DevOps is the union of people, processes and products to enable continuous delivery of value to end users."

Donovan Brown, Microsoft DevOps PM

DevOps Consists of

- Culture
- Measurement
- Automation
- Collaboration

Principles, Concepts, Practices and People

PRINCIPLES	CONCEPTS	PRACTICES	PEOPLE
1. Flow 2. Feedback 3. Continual learning and improvement	1. Product 2. Value stream 3. Loosely-coupled architecture 4. Autonomous teams	1. Continuous integration 2. Fast and reliable automated testing 3. Continuous and automated deployment / provisioning 4. Measurement and feedback	1. Customer centric 2. End-to-end responsibility 3. Collaboration 4. Learning and improvement 5. Experimentation and risk taking

DevOps Principles: Flow

- Enable fast flow of work from Development to Operations to the customer
  - Make work visible using visual boards
  - Limit work in process (WIP)
  - Reduce batch sizes
  - Reduce the number of handovers
  - Continually identify and elevate constraints
  - Eliminate hardships and waste in the value stream

DevOps Principles: Feedback

- Enable fast and constant flow of feedback at all value stream stages
  - Establish fast feedback loops at every step of the process
  - Establish pervasive production telemetry ensuring that problems are detected and corrected as they occur
  - Keep pushing quality closer to the source
  - Enable optimizing for downstream work centers

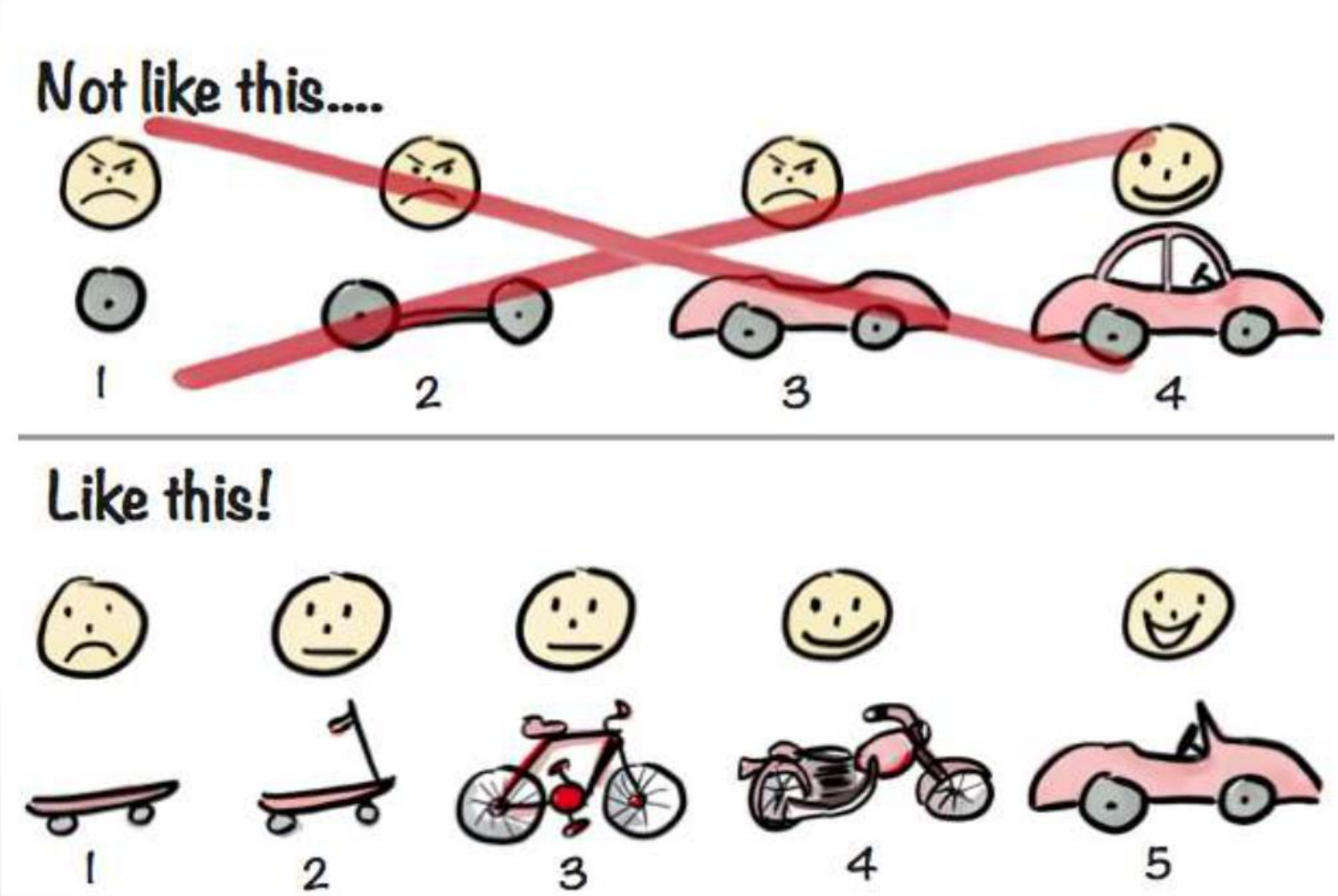
DevOps Principles: Continual Learning & Improvement

- Enable a high-trust, experimenting and risk-taking culture as well as organizational learning, both from successes and failures
  - Enabling organizational learning
  - Institutionalize the improvement of daily work
  - Transform local discoveries into global improvements
  - Inject resilience patterns into daily work
  - Encourage leaders to reinforce a learning culture
  - Experiment, fail fast - If it hurts, do it more often

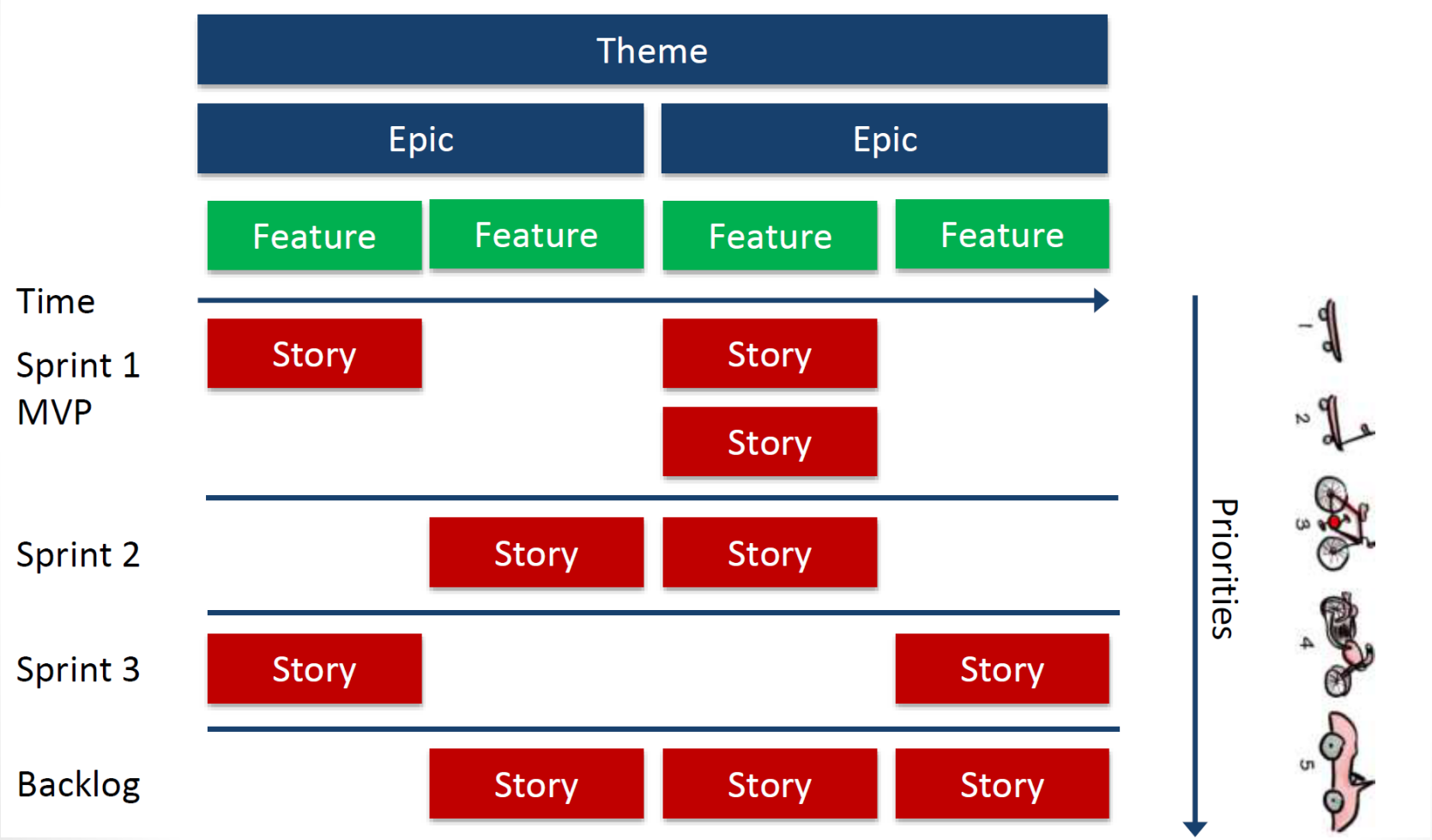
A Note on Product

- Strive towards delivering a minimum viable product (MVP), the smallest amount of functionality having value to the customer, as fast and reliably as it can.
- MVP reflects the end-product in a minimal functional form. It is used to test new ideas and verify whether the hypothesis are correct.
- MVP is that product which has just those features and no more that allows you to ship product that early adopters see and, at least some of whom resonate with, pay you money for, and start to give you feedback on

MVP



MVP


Theme
Epic
Epic
Feature
Feature
Feature
Feature
Time
Sprint 1
MVP
Story
Story
Story
Sprint 2
Story
Story
Sprint 3
Story
Story
Backlog
Story
Story
Story
Priorities
1
2
3
4
5

DevOps – The Holistic View

- Development
  - Requirements, version control, test case management, bug tracking, etc
- Testing
  - Unit, integration, exploratory, load, automated UI, performance, etc
- Deployment
  - Environment definition, provisioning and configuration
  - Application configuration and deployment
  - Approval workflows and automation
- Monitoring
  - Application Performance Monitoring
  - Alerts and notifications

What DevOps is NOT

- It is not a product
- It is not a specification
- It is not centralized
- It is not trademarked

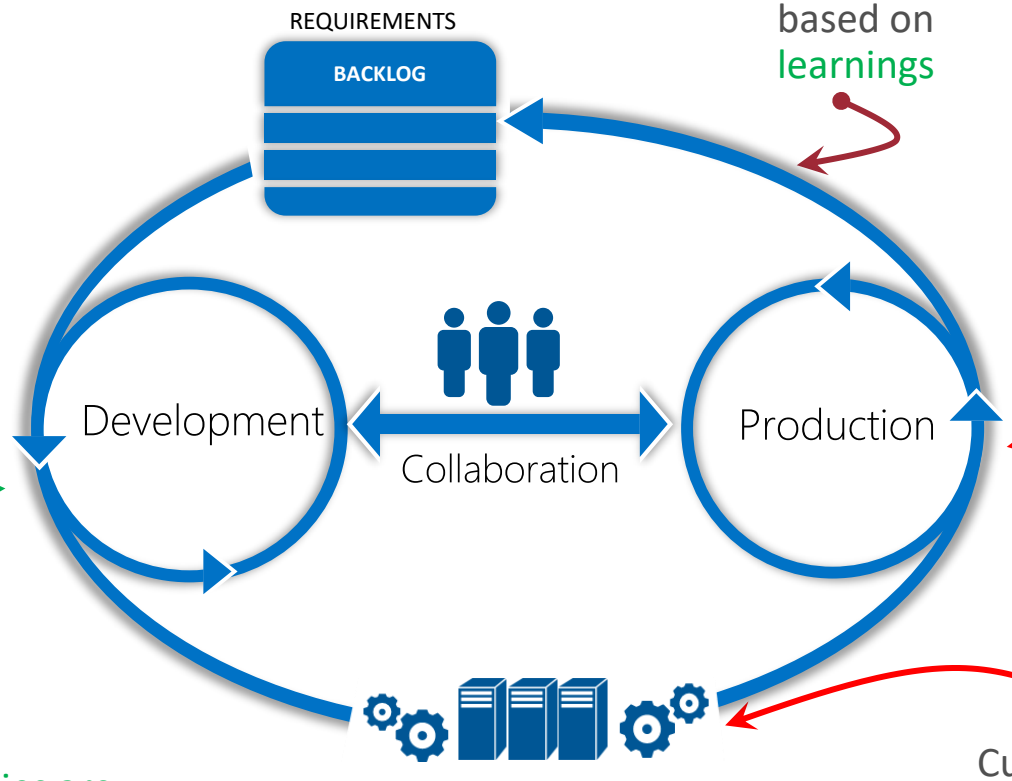
"You cannot buy DevOps and install it. DevOps is not just automation or infrastructure as code. DevOps is people following a process enabled by products to deliver value to our end users."

- Donovan Brown

What's driving DevOps?

Determine next set of investments based on learnings

Disconnects between Development and Operations increase mistakes and MTTR when issues occur



The agile Methodologies are accelerating the construction process

Current ITLM/ITSM "best practices" made the release and operate processes reliable, but not agile

DevOps Drivers

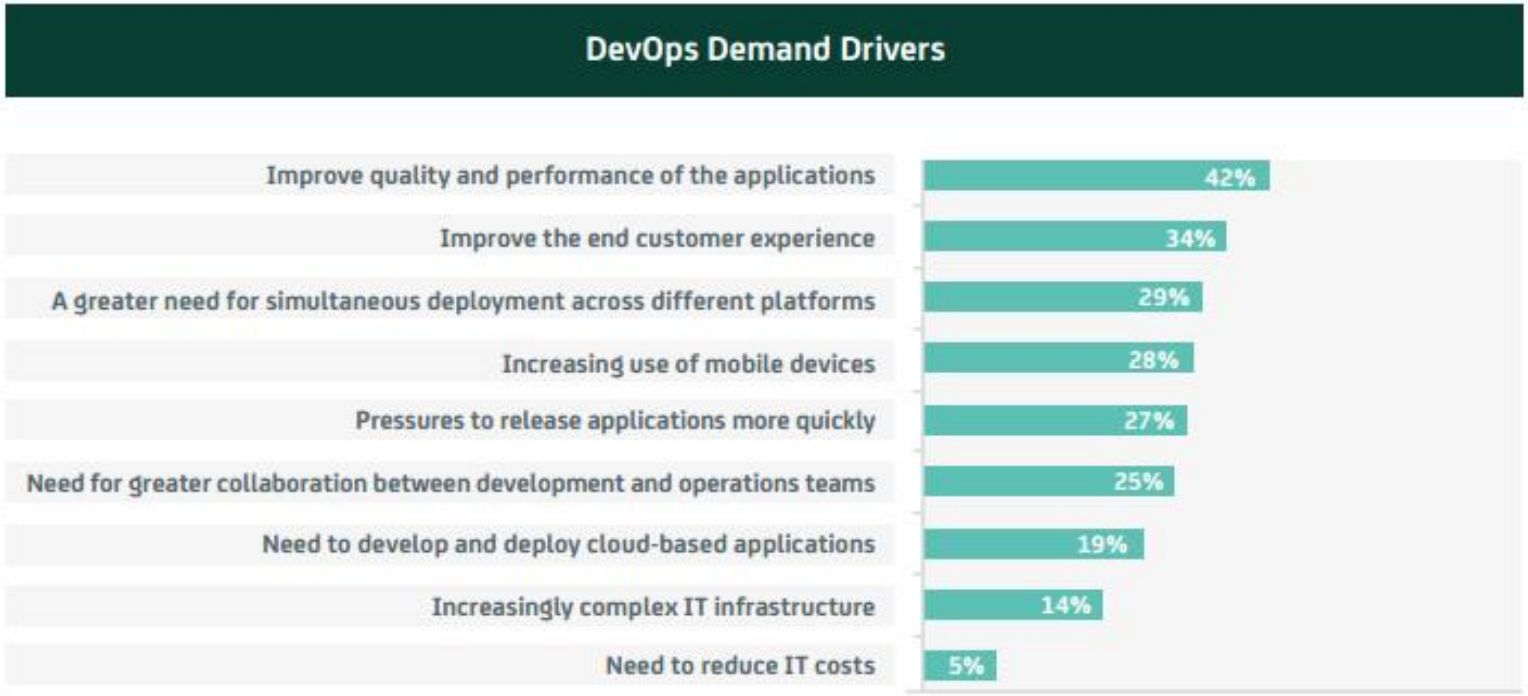


Figure 5.
What is driving the need for DevOps? Total: 1,425

Value of DevOps

- DevOps bridges the traditional divide allowing teams to produce high quality releases at increasing cadence
- DevOps goals span the entire delivery pipeline

Shorter Cycles & Higher Quality

- Faster time to market
- Lower failure rates
- Shortened lead time
- Faster MTTR[4]
  - Mean Time To Realize, Recover, Repair, Remediate

DevOps Benefits

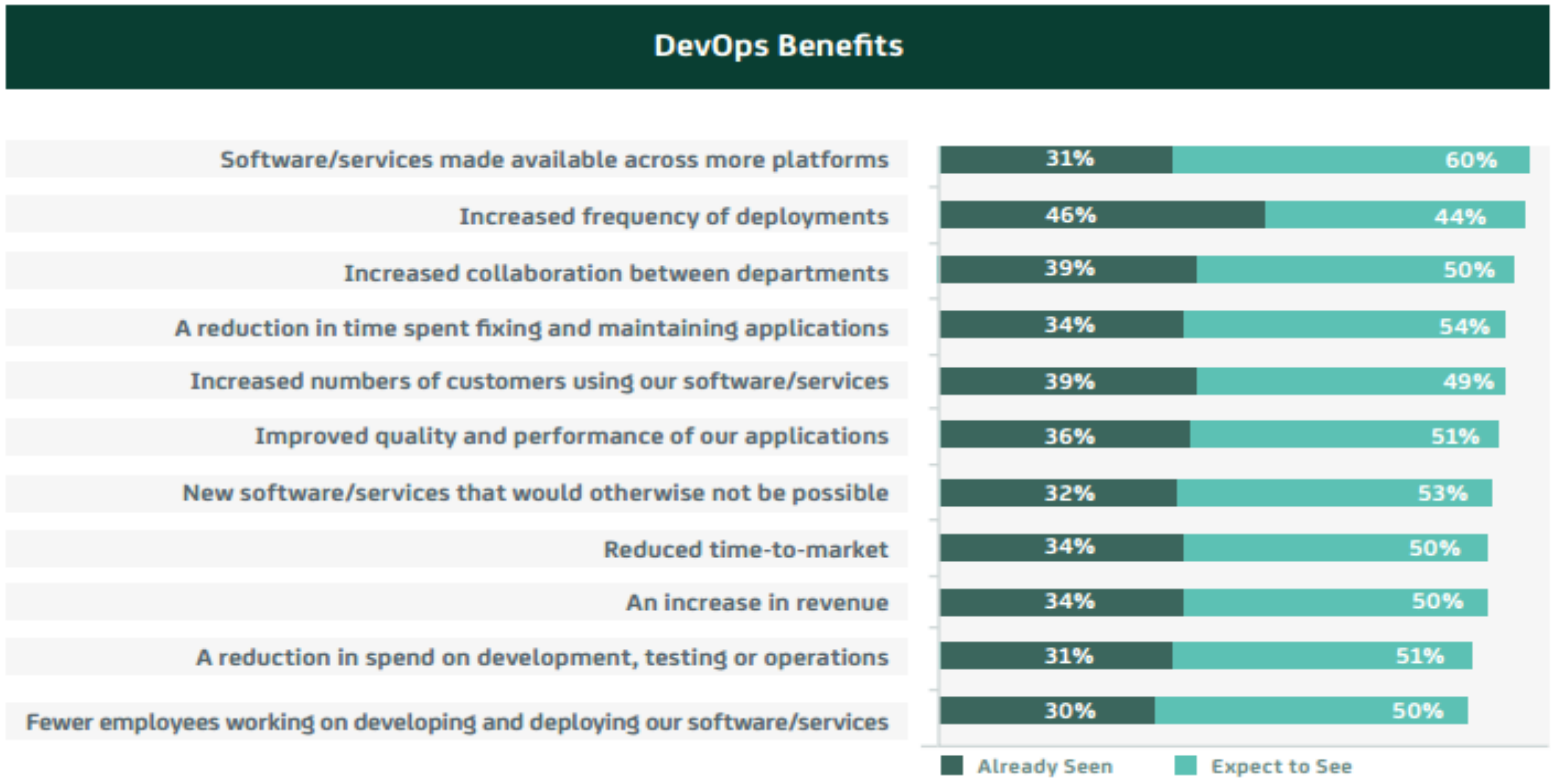


Figure 1.
What benefits have you seen or do you anticipate seeing from implementing DevOps in your organization? Total: 1,256 respondents who already have or plan to implement DevOps

History of DevOps

Agile Conference 2008
Patrick Debois and Andrew Shafer discuss "Agile Infrastructure"

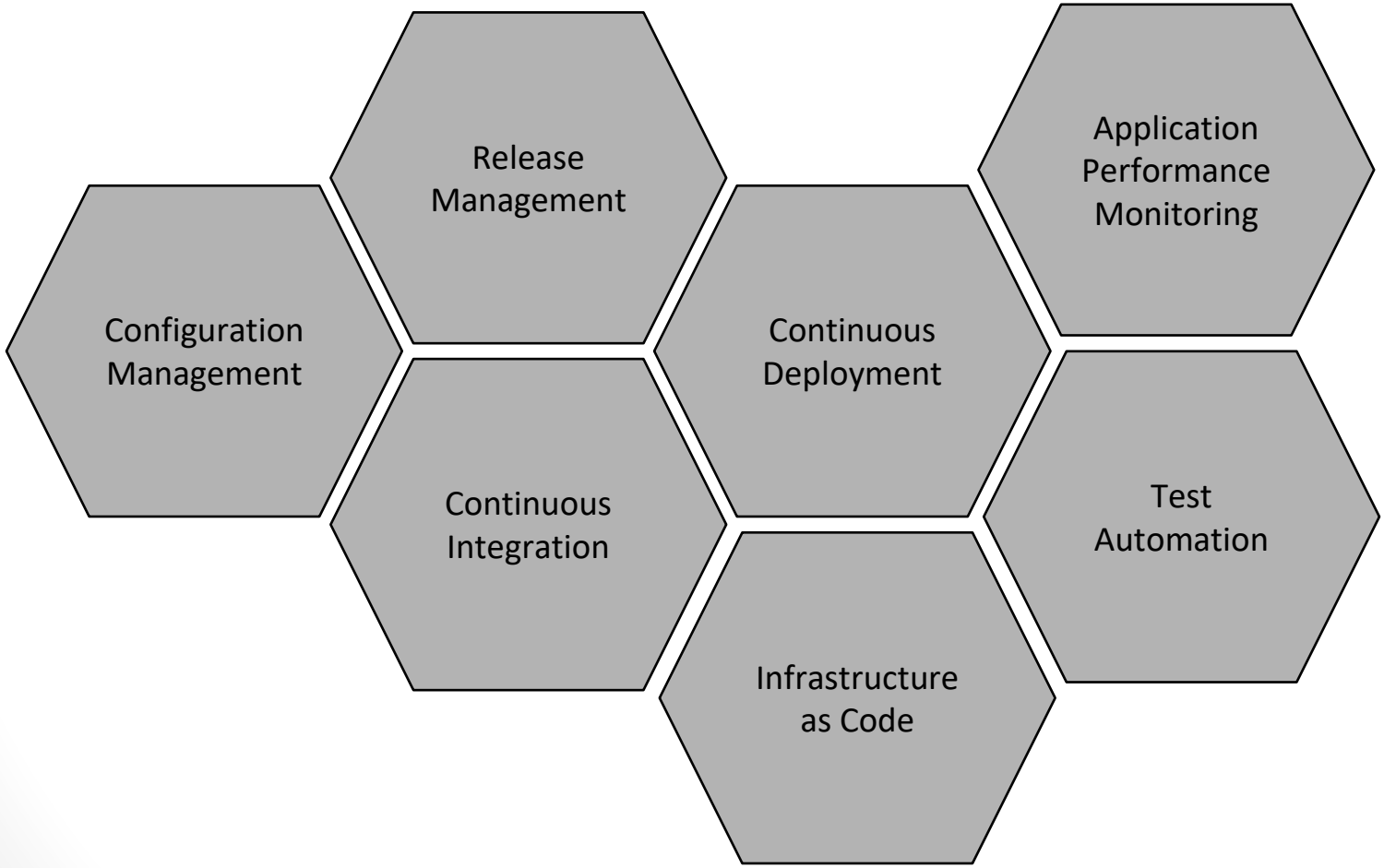
October 2009
Patrick Debois starts "DevOpsDays" in Ghent, Belgium

Velocity 2009
John Allspaw and Paul Hammond present "10 Deploys per Day: Dev and Ops Cooperation at Flickr"

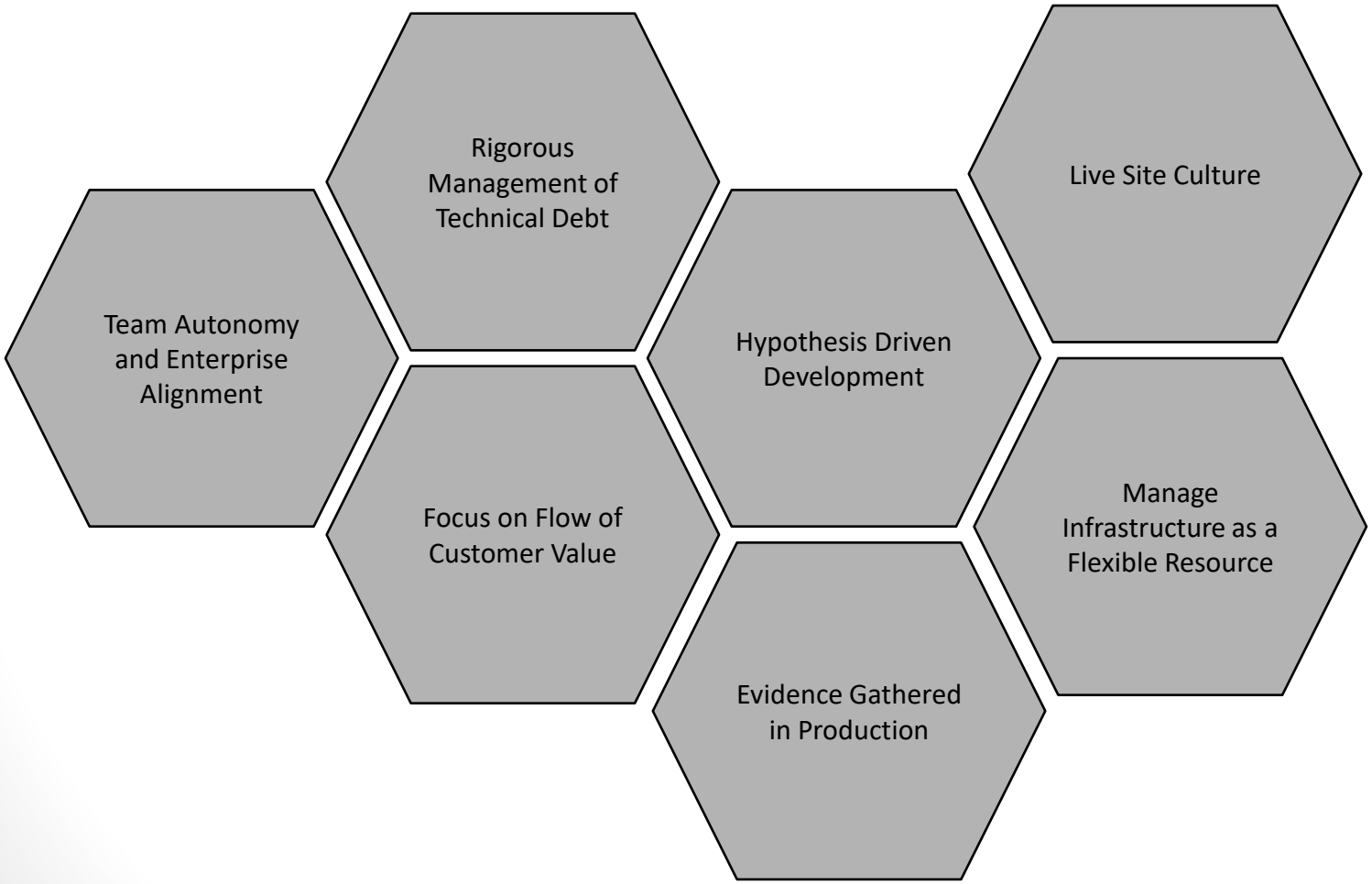
2010-
DevOpsDays spread globally
OSS Tools like Chef, Puppet, Vagrant, LogStash, Jenkins etc. gain popularity

March 2011
Cameron Haight of Gartner predicts explosion of DevOps in Global 2000 companies

7 DevOps Practices



7 DevOps Habits



DevOps Metrics

Deployment frequency

Change lead time

Change fail rate

Mean time to detect & repair

Agility performance indicators

Reliability performance indicators

DevOps and Agile

Revisiting the Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more

Agile Principles

We follow these principles:
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's 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.

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 maximizing the amount of work not done--is essential.

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

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

Agile Principles

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's 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.

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 maximizing the amount of work not done--is essential.

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

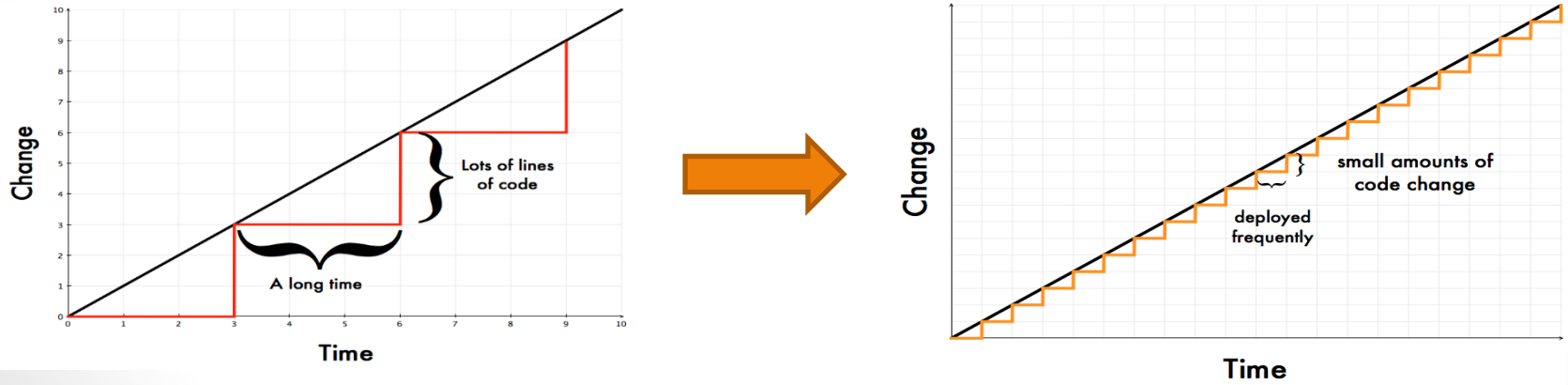
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Operations

- Source Control
- Small, frequent releases
- Automated testing
- Continuous Integration
- Continuous Deployment
- Peer Review
- Immutable Infrastructure

Benefits of Small Releases

- Lower risk
- Faster feedback
- More confidence



John Allspaw's visualization of slow and fast delivery cycles

Version Control and Automation

What is Version/Source Control?

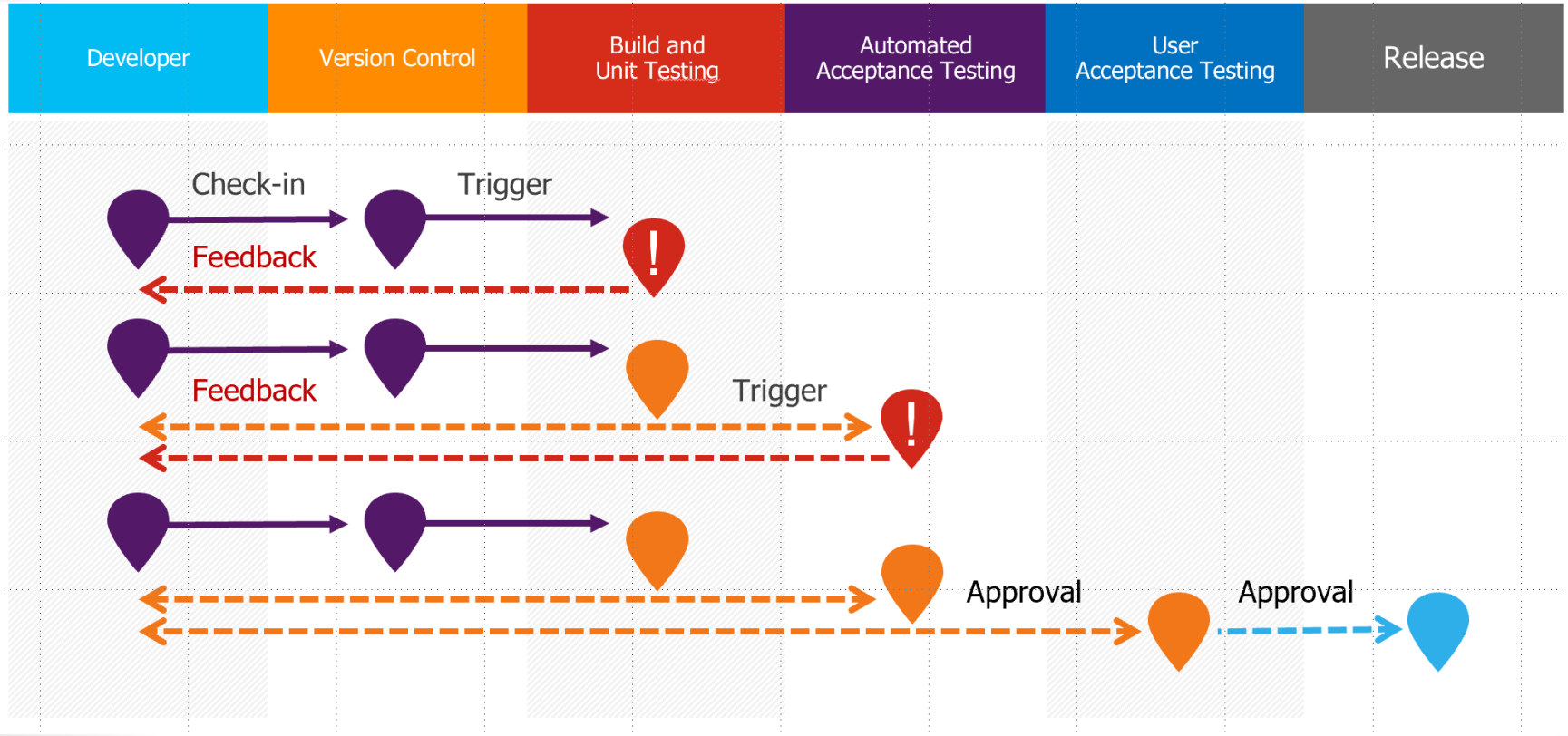
- The management of changes to documents, computer programs, large web sites, and other collections of information.
- Supported by a tool
- Provides ways to see differences between versions
- Allows parallel development through merges and branches
- Foundational in software development, but occasionally new to operations teams

What to Version Control?

- Source Code
- Environment definition
- Infrastructure configuration
- Deployment scripts
- Documentation

- EVERYTHING!

Automation enables continuous value delivery



Benefits of Automation in DevOps

- Removes manual errors
- Enables anyone to perform tasks
- Enables speed, reliability and consistency
- Empowers frequent releases and self-service

What to Automate?

- Build and Deployment
- Environment creation
- Infrastructure configuration
- Unit, Integration, UI and Performance Testing
- Documentation generation
- Monitoring and notifications

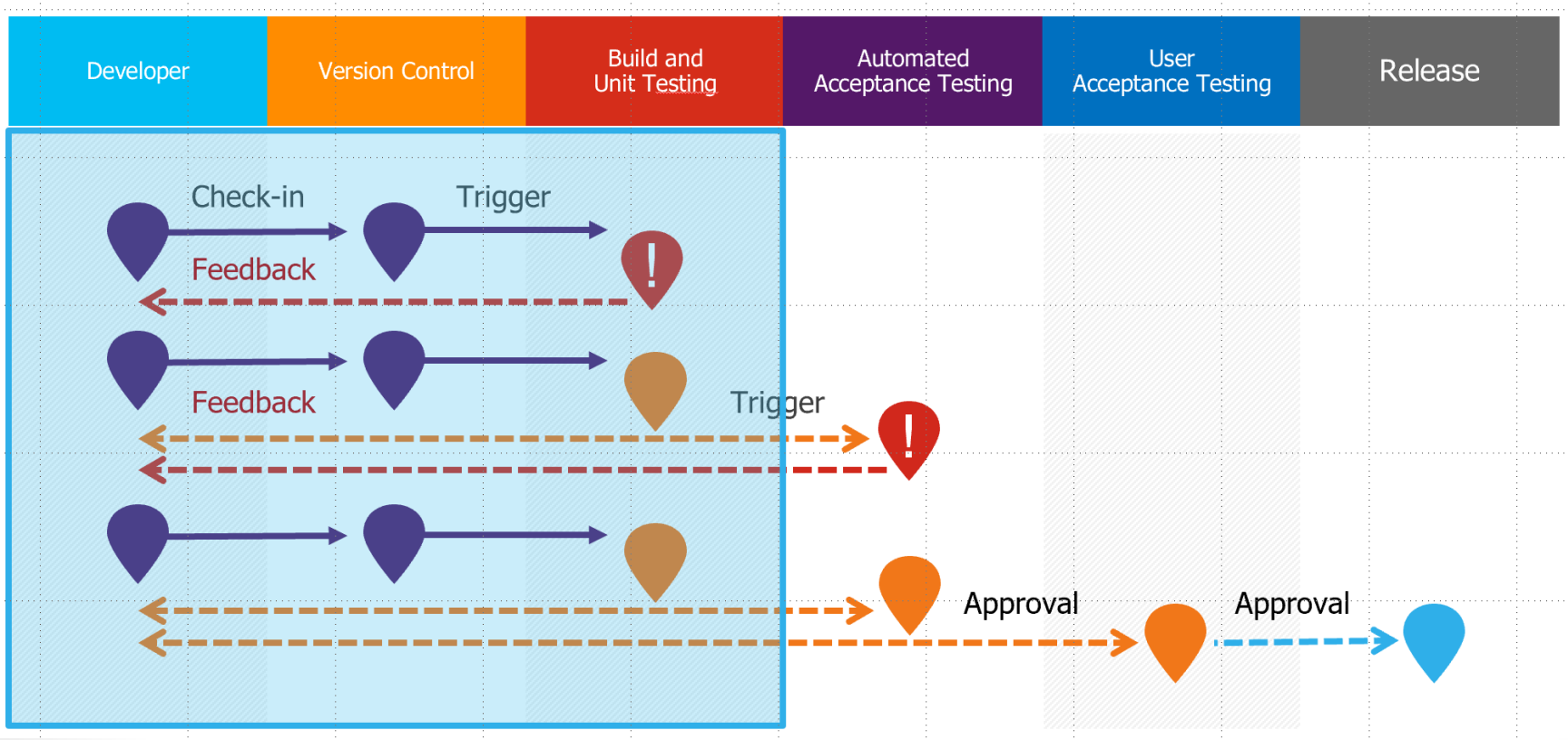
- EVERYTHING!

CI / CD

What is Continuous Integration (CI)?

- The practice of merging all developer working copies to a shared code line several times a day, and validating each integration with an automated build.
- Unit tests are generally executed during the build
- In practice, CI is often defined as having a build with unit tests that executes at every commit / check-in to version control
- This provides confidence in individual branches, but not on the integration of all the code changes

Continuous integration



Benefits of Continuous Integration

- Rapid feedback for code quality
- Trigger for automated testing for every code change
- Code analysis and technical debt management
- Reduces long, difficult and bug-inducing merges
- Increases confidence in code long before production

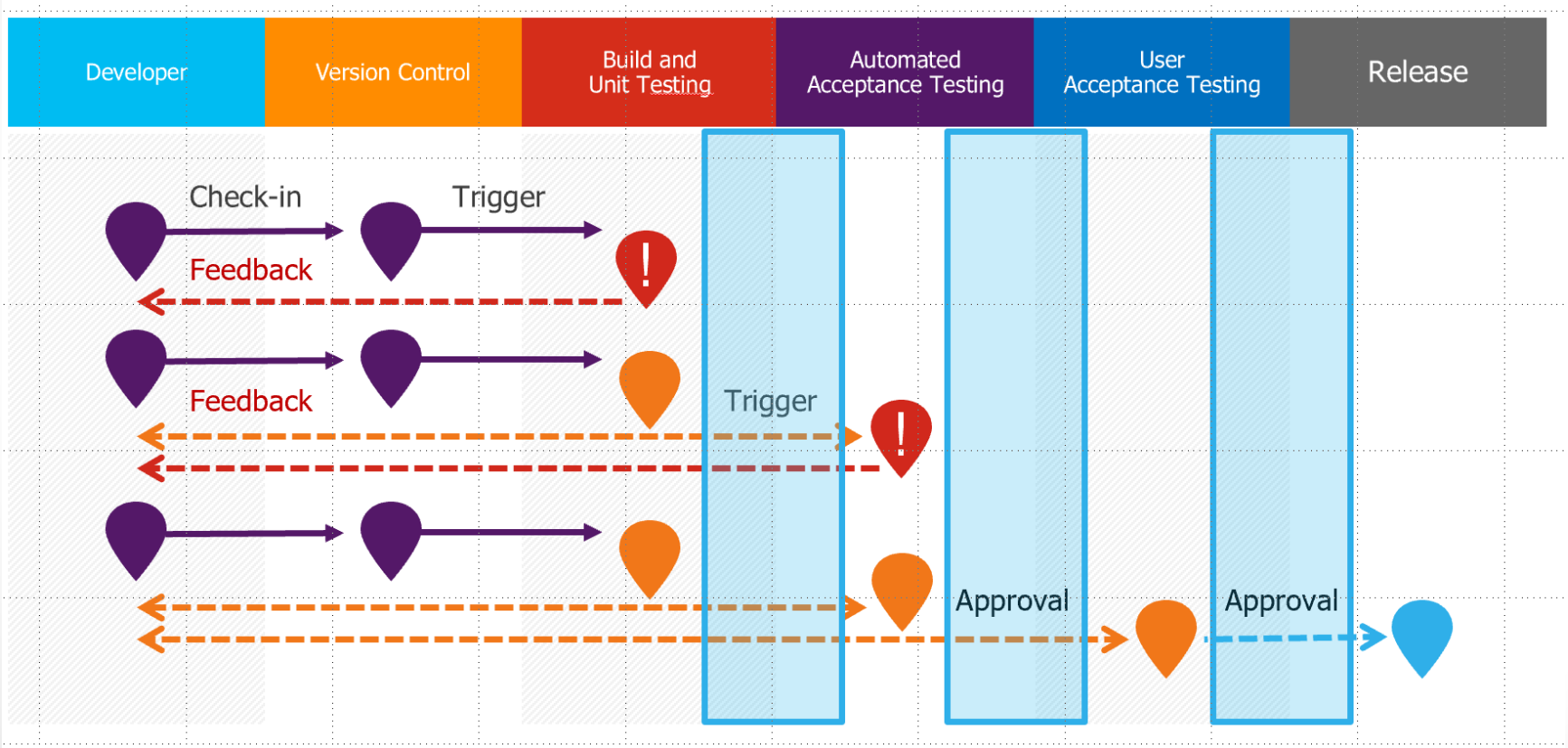
What is Continuous Delivery?

- A software engineering approach in which teams produce software in short cycles, ensuring that software can be reliably released at any time.
- Aims to build, test and release software faster and more frequently
- Reduce the cost, time and risk of delivering changes by allowing for more incremental updates to production
- In practice, continuous delivery focuses on an automated deployment pipeline
- This may have one or more manual approval gates prior to reaching production

Continuous Delivery vs. Continuous Deployment

- Continuous Deployment is generally defined as a Continuous Delivery pipeline with no manual gates between initial code commit / check-in and production
- Feature flags are commonly used in both patterns, however, they are often necessary for Continuous Deployment
- Feature flags ensure that code deployed to a production environment is not necessarily released to all end users (Deployment Release)
- This allows for more mature features to be enabled in production (generally via configuration), while newer features can be switched off for most users

Continuous Delivery



Benefits of Continuous Delivery

- Encourages Infrastructure as Code
- Encourages Configuration as Code
- Enables automated testing throughout the pipeline
- Provides visibility
- Provides fast feedback cycles
- Makes going to production a low stress activity
- Increases confidence in code long before production

What is a Build Pipeline?

- Automated system responsible for Continuous Integration
- Builds code, runs unit tests, creates packages, etc.
- Generally triggered by a code commit / check-in, or on a schedule

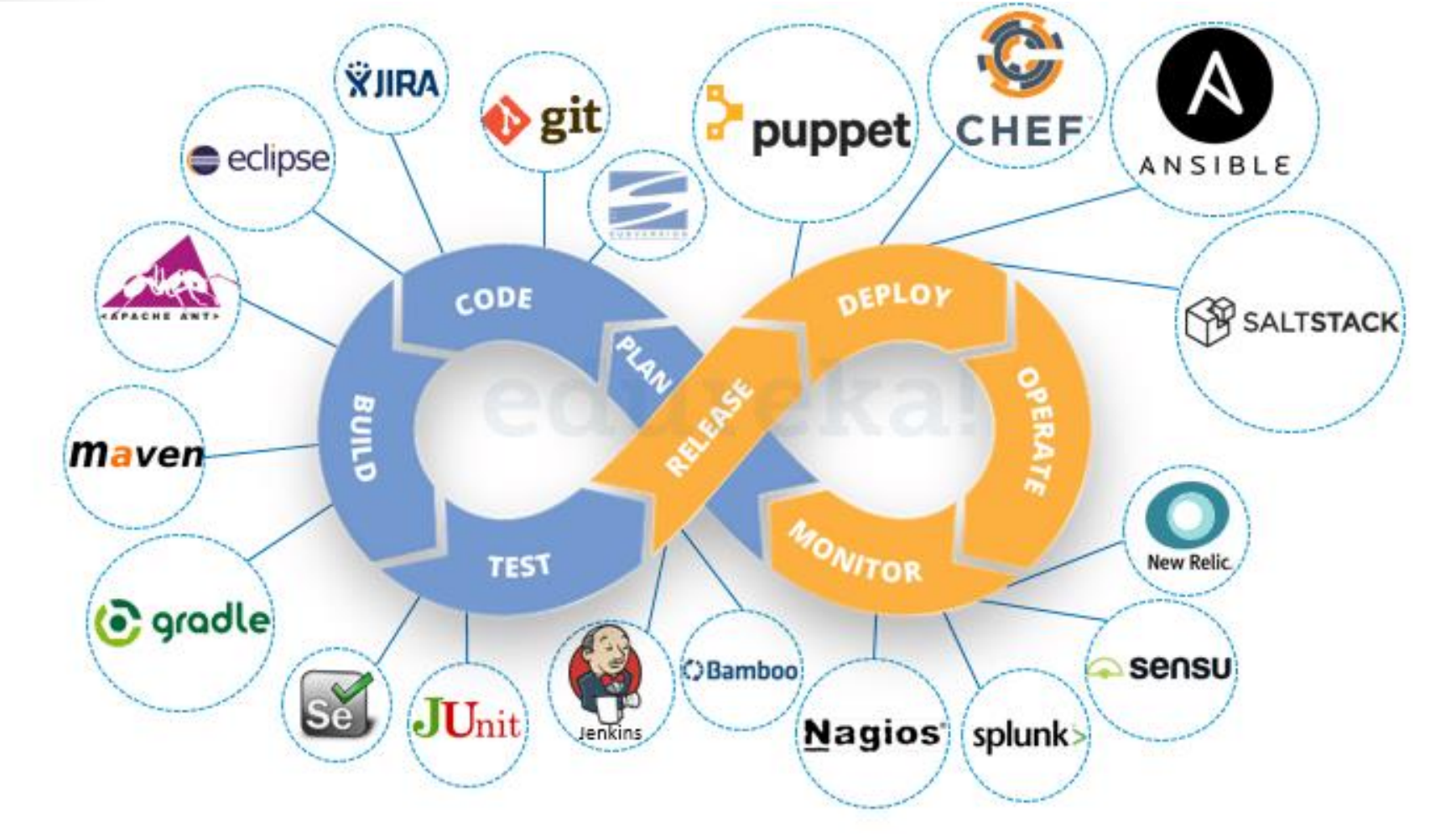
- Note: The Build Pipeline and the Deployment Pipeline can be considered two different concepts, but in many systems the same tool orchestrates both.

Defining a Build Pipeline

- Trigger
  - Typically a commit / check-in to version control
  - Can include gated check-ins
- Tasks
  - Compilation, minification, tokenization, etc.
- Unit Testing
- Code Analysis
- Versioning and Packaging

DevOps Tools

DevOps Tools



DevOps: Most Used Tools

sonarqube

Nexus

maven

Jenkins

JIRA

git

puppet labs

RUNDECK

CHEF

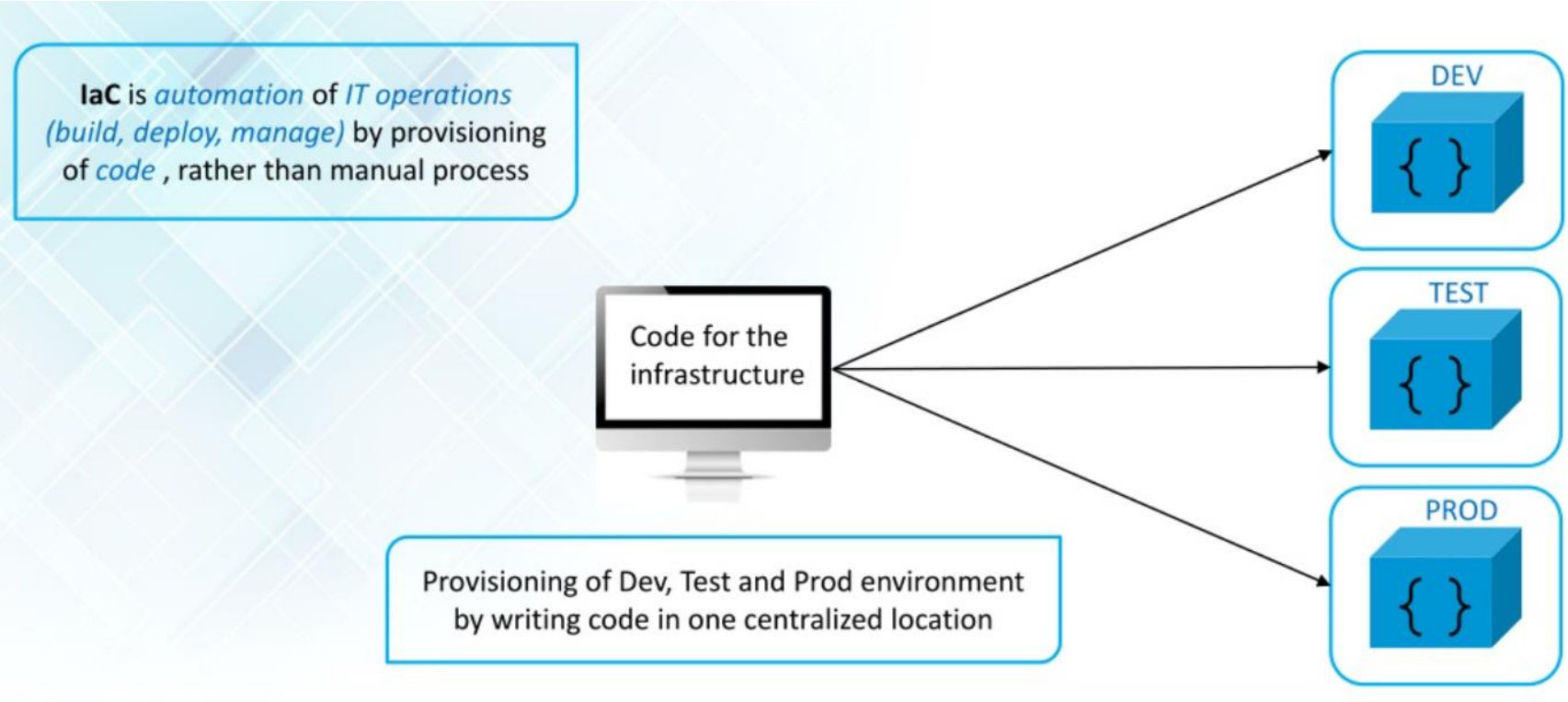
docker

SUBVERSION

Apache Tomcat

Devops Tools Landscape

Infrastructure as Code



Infrastructure as Code

Shell Script

```
echo
"spock:*:1010:1010:Spock:
/home/spock:/bin/sh" \ >>
/etc/passwd
(the user spock is added
to passwd file)
```

CM Tool Script

```
user { "spock":
ensure => present,
gid => "science",
home => "/home/spock",
shell => "/bin/sh"
}
```

- ✓ In shell script, you need to write automation script from scratch but in CM (configuration management) tool 80% things are already available
- ✓ In shell script, you need to define the workflows whereas in CM tool the workflows are already available
- ✓ You have UI (user interface) in CM tools to ease your job for automating the tasks but you don't have UI in shell scripting

CM Tools

puppet SALTSTACK CHEF ANSIBLE

Ansible vs. Chef vs. Puppet

Category	Chef	Puppet	Ansible
Availability (in case of failure)	Backup Server	Alternative Master	Secondary instance
Configuration Language	Ruby DSL	Ruby, Puppet DSL, Embedded Ruby (ERB), DSL	Python, YAML
Architecture	Master-Agent	Master-Agent	Only Master (Agentless)
Installation Process	Time-intensive and complexdue to Chef Workstation	Time-intensive due to master-agent certificate signing	Easy
Configuration Management	Pull	Pull	Push and Pull
Scalability	High	High	Very High
Interoperability	Chef Server works only on Linux/Unix; Chef Client and Workstation can work on Windows as well	Puppet Master works only on Linux/Unix; Puppet Agent or Client works on Windows	Ansible Server works on Linux/Unix; Client machines on Windows
Capabilities	■ Continuous delivery with automated workflow ■ Compliance and security management ■ Infrastructure automation	■ Orchestration ■ Automated provisioning ■ Code and node management ■ Configuration automation ■ Visualization and reporting ■ High transparency ■ Role-based access control	■ Simple orchestration ■ Streamlined provisioning ■ Continuous delivery with automated workflow ■ App deployment ■ Security and compliance integration into automation

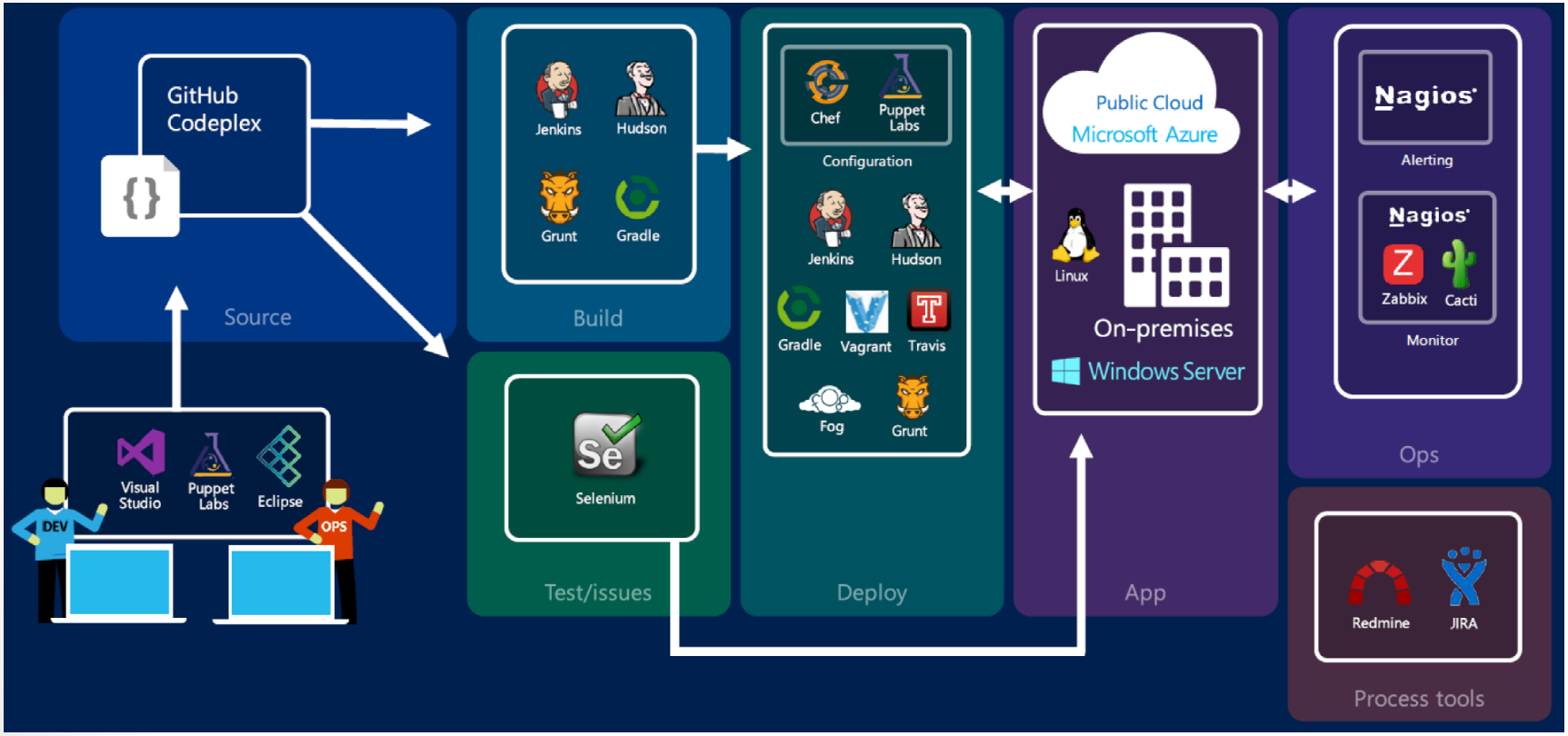
Ansible vs. Chef vs. Puppet vs. SaltStack

Ansible	Puppet	Saltstack	Chef
Streamlined provisioning	Orchestration	Automation for CloudOps	Infrastructure automation
Configuration management	Automated provisioning	Automation for ITOps	Cloud automation
App deployment	Role-based access control	Continuous code integration and deployment	Compliance and security management
Automated workflow for Continuous Delivery	Visualization and reporting	DevOps toolchain workflow automation with support for Puppet, Chef, Docker, Jenkins, and Git.	Automated workflow for Continuous Delivery
Security and Compliance policy integration	Configuration automation	Application monitoring and auto-healing	Chef-Server using RabbitMQ, AMQP protocol.
Simplified orchestration	Code and node management	Orchestration	Automation for DevOps workflow

Ansible vs. Chef vs. Puppet vs. SaltStack

Parameters	Chef	Puppet	Ansible	Saltstack
Availability	Yes	Yes	Yes	Yes
Configuration Language	DSL (Ruby)	DSL(PuppetDSL)	YAML (Python)	YAML (Python)
Setup and Installation	Moderate	Moderate	Very Easy	Moderate
Ease of Management	Tough	Tough	Easy	Easy
Scalability	HighlyScalable	HighlyScalable	HighlyScalable	HighlyScalable
Interoperability	High	High	High	High
Pricing	$13700	$11200-$19900	$10,000	$15,000(approx.)
Cloud Support	All	All	All	All

DevOps Real Example



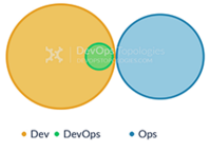
DevOps Topologies

DevOps Topologies

DevOps Team Types

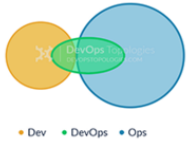
Type 1: Dev and Ops Collaboration

Type 2: Fully Shared Ops Responsibilities



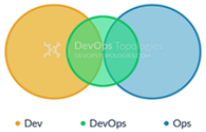
Type 3: Ops as Infrastructure-as-a-Service (Platform)

Type 4: DevOps as an External Service



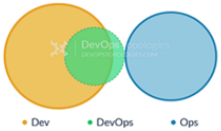
Type 5: DevOps Team with an Expiry Date

Type 6: DevOps Advocacy Team



Type 7: SRE Team (Google Model)

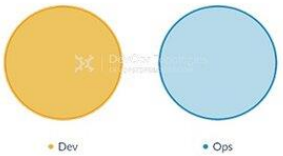
Type 8: Container-Driven Collaboration



Type 9: Dev and DBA Collaboration

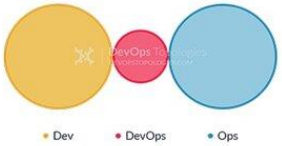
DevOps Topologies

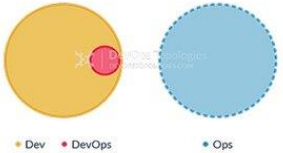
DevOps Anti-Types



Anti-Type A: Dev and Ops Silos

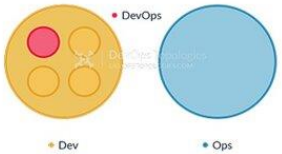
Anti-Type B: DevOps Team Silo

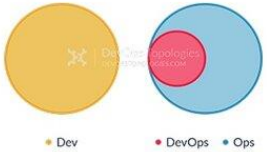




Anti-Type C: Dev Don't Need Ops

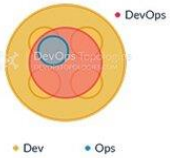
Anti-Type D: DevOps as Tools Team

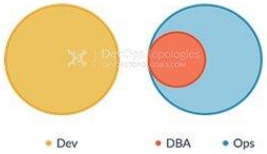




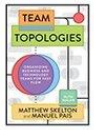
Anti-Type E: Rebranded SysAdmin

Anti-Type F: Ops Embedded in Dev Team





Anti-Type G: Dev and DBA Silos



Any Questions?

Your time is limited, don't waste it living someone else's life

Steve Jobs, Stanford University speech, 2005