DevOps, Cloud & Cloud Native Applications

- Chidanand R

Agenda

- Demystifying DevOps
- DevOps Framework & Principles
- DevOps Best Practices – CI, CD, TDD, BDD, CDD, Everything As Code, GitOps
- Introduction to Cloud & Cloud Native Applications
- Public Cloud Comparison
- Q & A

History of DevOps

- ***Patric Debois***, a Belgian consultant, project manager, and agile practitioner is regarded as founding Father of DevOps

- Acronym coined in to attract Developer community to an Agile Conference in 2008

DevOps Definition

"It's a <u>movement of people</u> who think its time for change in the IT industry – time to stop wasting money, time to start delivering great software and building systems that scale and last"

The DevOps movement, unlike Agile, lacks a manifesto!!

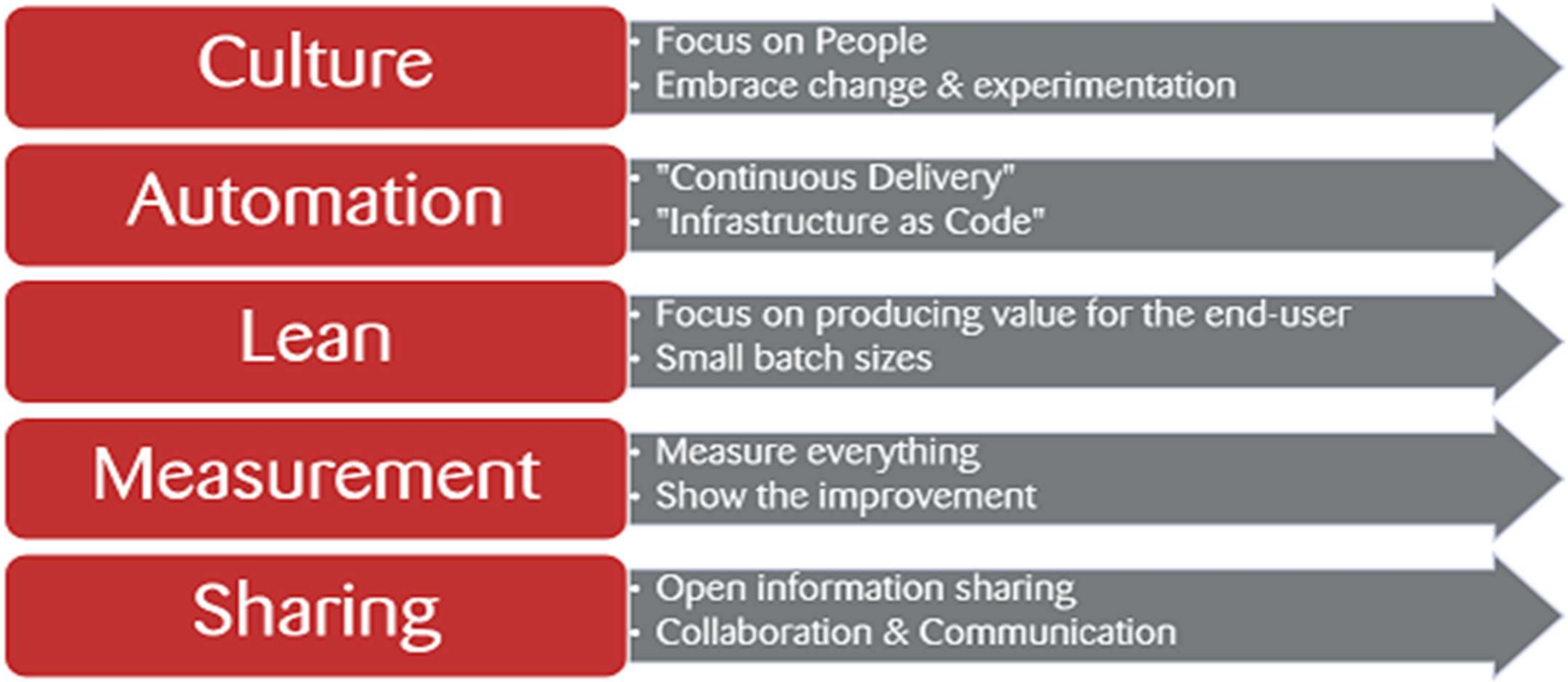
DevOps Evolved Definitions

"DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality"

- **Bass, Weber, and Zhu**

- DevOps is a set of cultural practices that allows people, processes, and tools to build and release software faster, more frequently, and more reliably.

DevOps Framework



Culture
- Focus on People
- Embrace change & experimentation

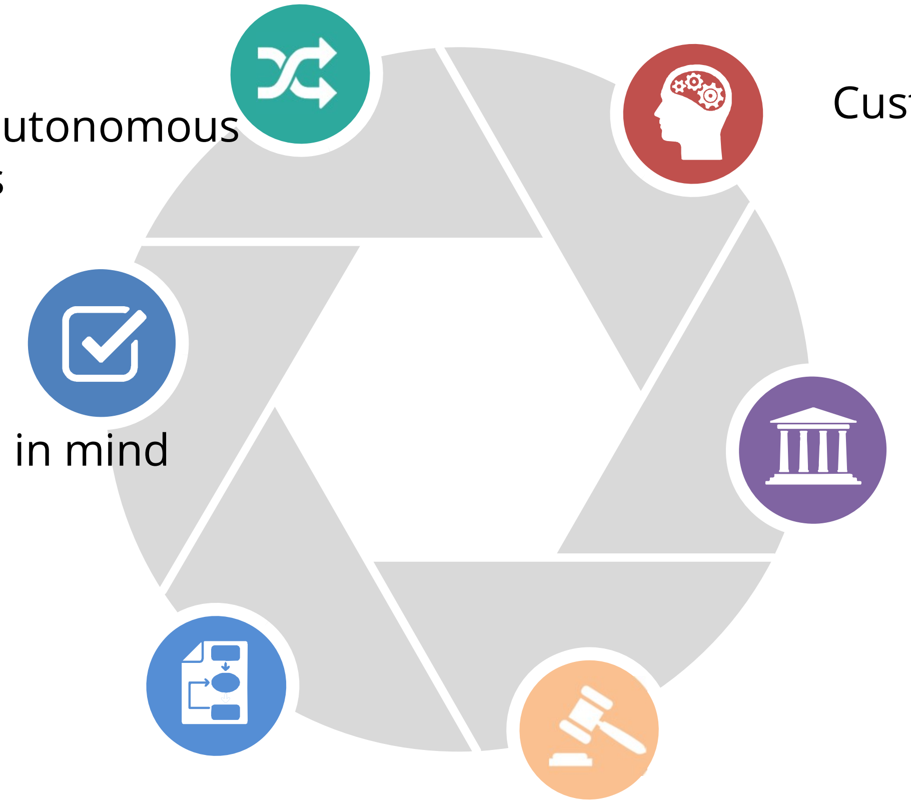
Automation
- "Continuous Delivery"
- "Infrastructure as Code"

Lean
- Focus on producing value for the end-user
- Small batch sizes

Measurement
- Measure everything
- Show the improvement

Sharing
- Open information sharing
- Collaboration & Communication

DevOps Principles



Cross-functional autonomous teams

Customer-centric action

Create with the end in mind

End-to-end responsibility

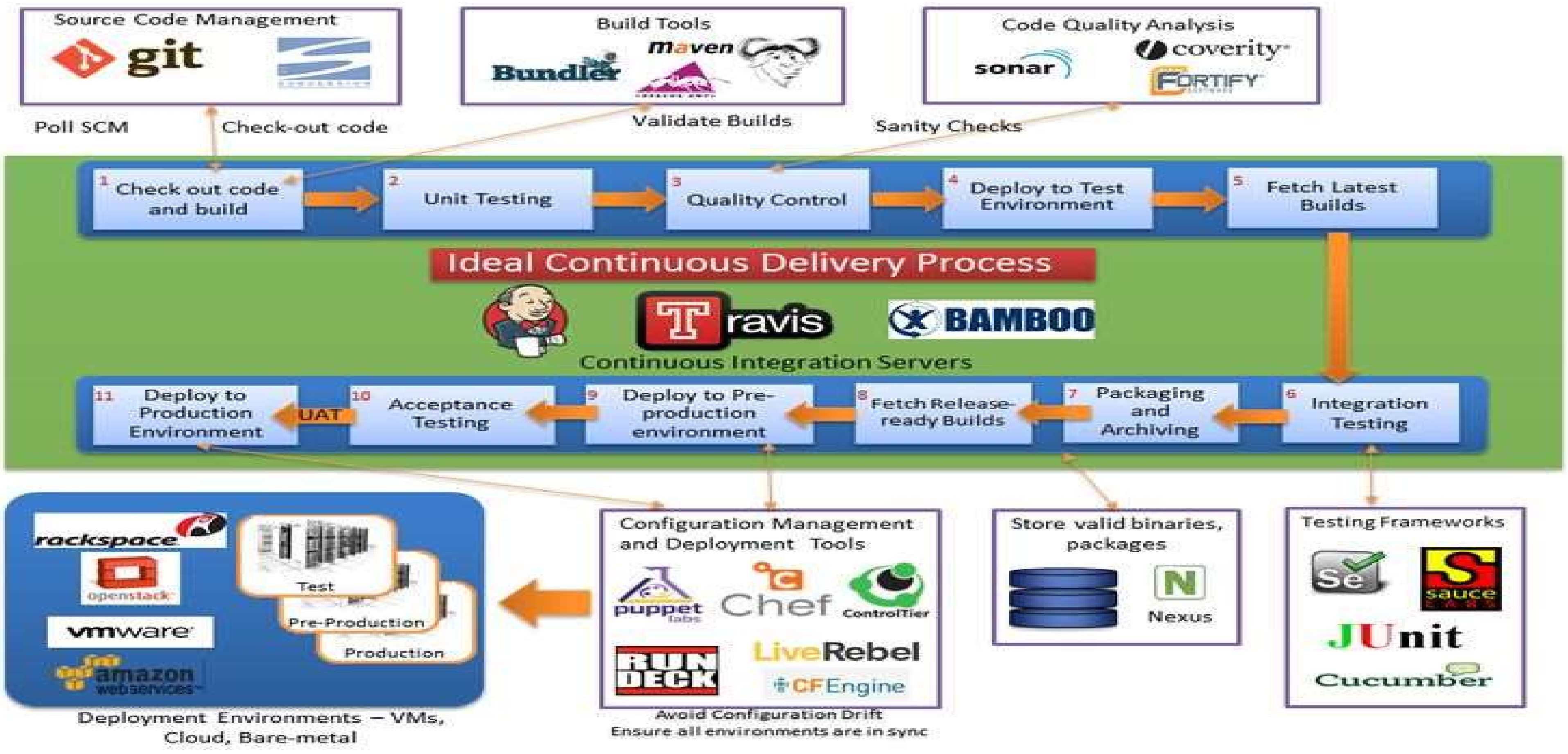
Continuous improvement

Automate everything you can

DevOps – Best Practices

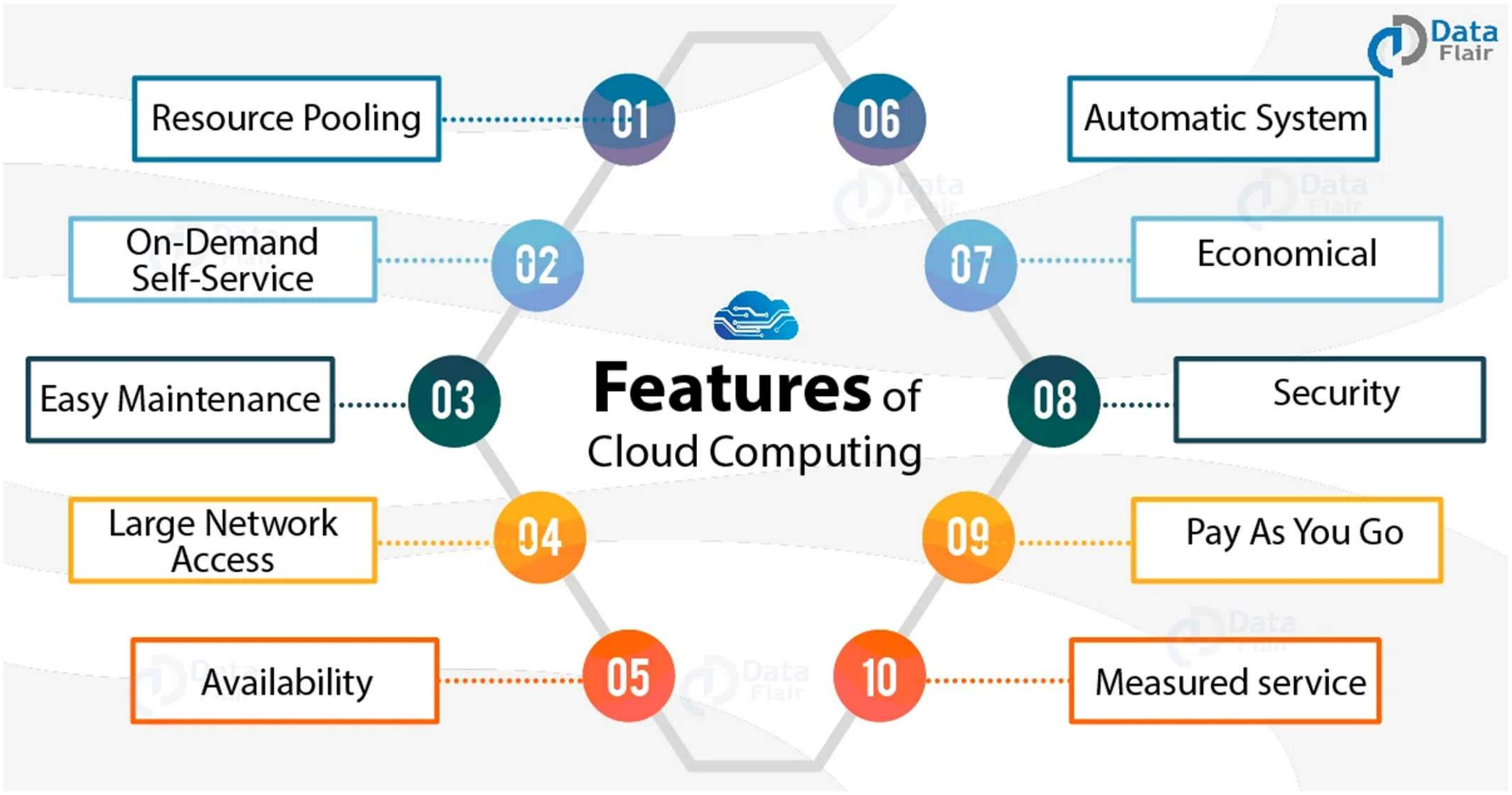
- Continuous Improvements – CI, CD
- Everything As Code(?)
- GitOps Model
- TDD, BDD, CDD

DevOps – Automation Goal

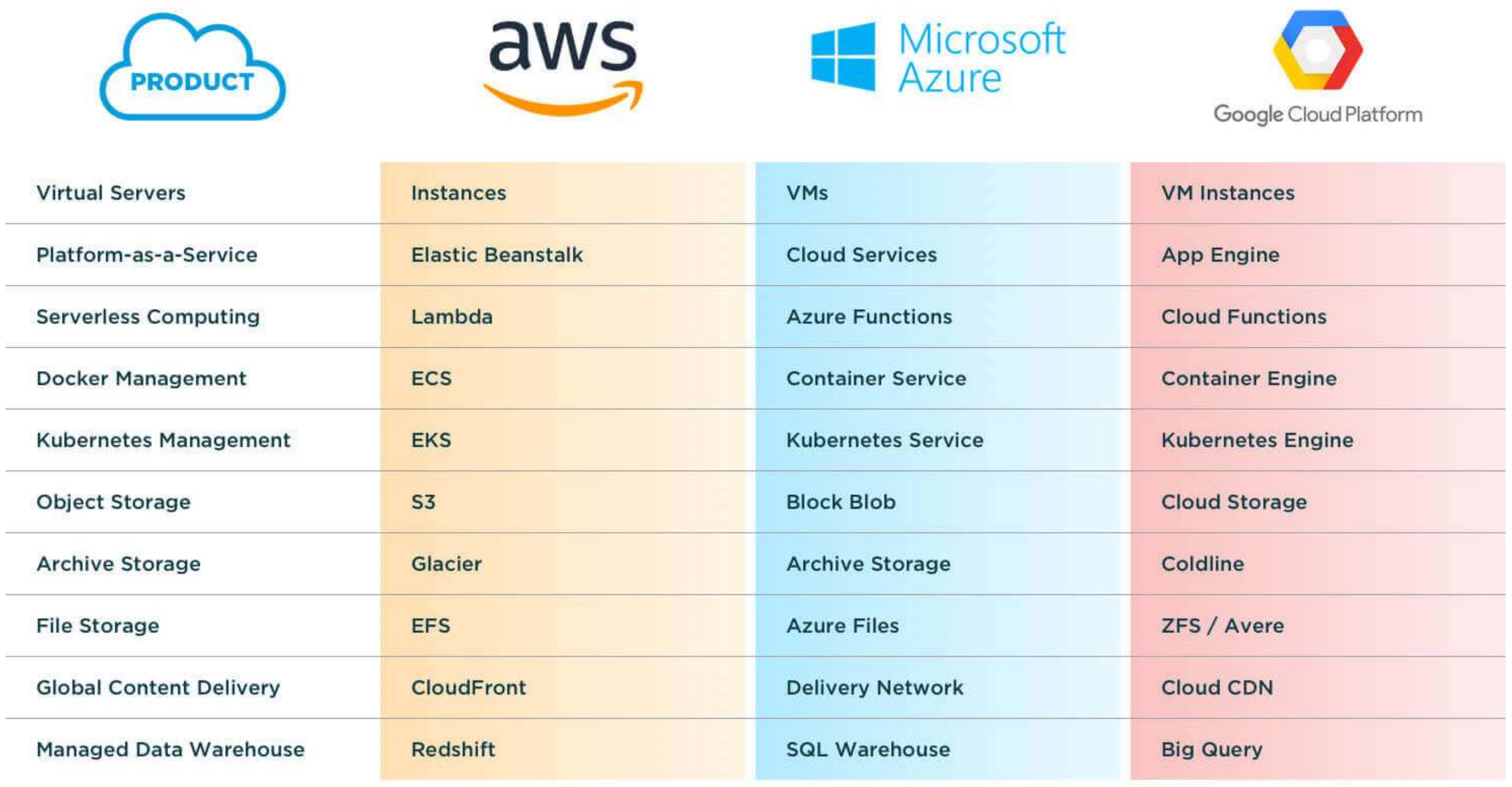


Source Code Management
git
Build Tools
Maven
Bundler
Code Quality Analysis
sonar
coverity
FORTIFY
Poll SCM
Check-out code
Validate Builds
Sanity Checks
1 Check out code and build
2 Unit Testing
3 Quality Control
4 Deploy to Test Environment
5 Fetch Latest Builds
Ideal Continuous Delivery Process
Travis
BAMBOO
Continuous Integration Servers
11 Deploy to Production Environment
UAT
10 Acceptance Testing
9 Deploy to Pre-production environment
8 Fetch Release-ready Builds
7 Packaging and Archiving
6 Integration Testing
rackspace
openstack
vmware
amazon webservices
Test
Pre-Production
Production
Deployment Environments – VMs, Cloud, Bare-metal
Configuration Management and Deployment Tools
puppet labs
Chef
ControlTier
RUN DECK
LiveRebel
CFEngine
Avoid Configuration Drift
Ensure all environments are in sync
Store valid binaries, packages
Nexus
Testing Frameworks
Se
sauce LABS
JUnit
Cucumber

Cloud Computing

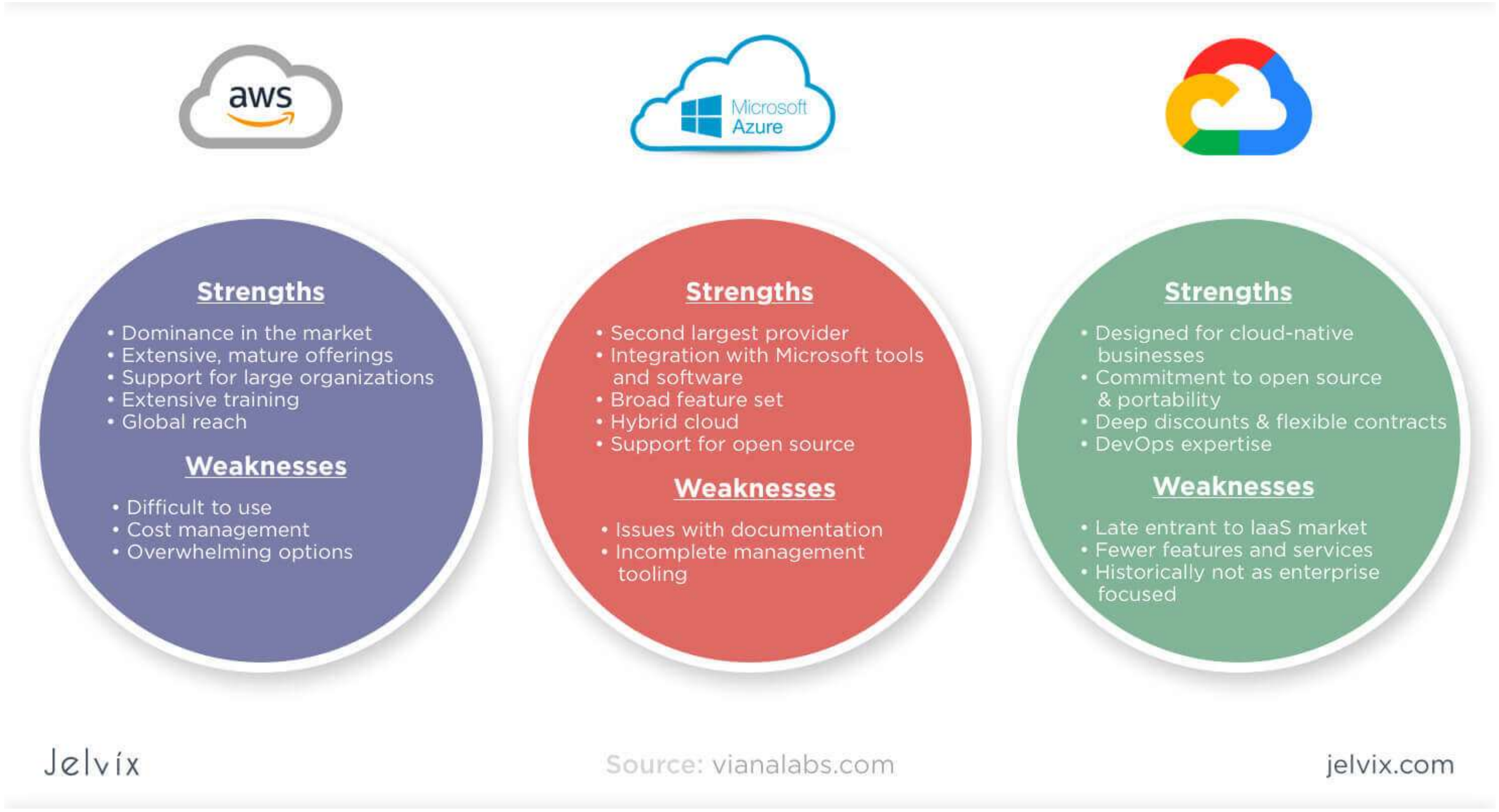


Public Cloud Comparison



PRODUCT	aws	Microsoft Azure	Google Cloud Platform
Virtual Servers	Instances	VMs	VM Instances
Platform-as-a-Service	Elastic Beanstalk	Cloud Services	App Engine
Serverless Computing	Lambda	Azure Functions	Cloud Functions
Docker Management	ECS	Container Service	Container Engine
Kubernetes Management	EKS	Kubernetes Service	Kubernetes Engine
Object Storage	S3	Block Blob	Cloud Storage
Archive Storage	Glacier	Archive Storage	Coldline
File Storage	EFS	Azure Files	ZFS / Avere
Global Content Delivery	CloudFront	Delivery Network	Cloud CDN
Managed Data Warehouse	Redshift	SQL Warehouse	Big Query

Public Cloud Comparison



aws

Strengths

- Dominance in the market
- Extensive, mature offerings
- Support for large organizations
- Extensive training
- Global reach

Weaknesses

- Difficult to use
- Cost management
- Overwhelming options

Microsoft Azure

Strengths

- Second largest provider
- Integration with Microsoft tools and software
- Broad feature set
- Hybrid cloud
- Support for open source

Weaknesses

- Issues with documentation
- Incomplete management tooling

Strengths

- Designed for cloud-native businesses
- Commitment to open source & portability
- Deep discounts & flexible contracts
- DevOps expertise

Weaknesses

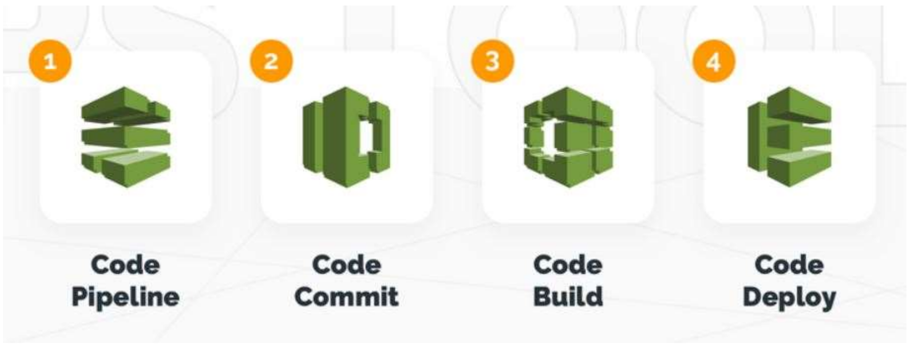
- Late entrant to IaaS market
- Fewer features and services
- Historically not as enterprise focused

Jelvix

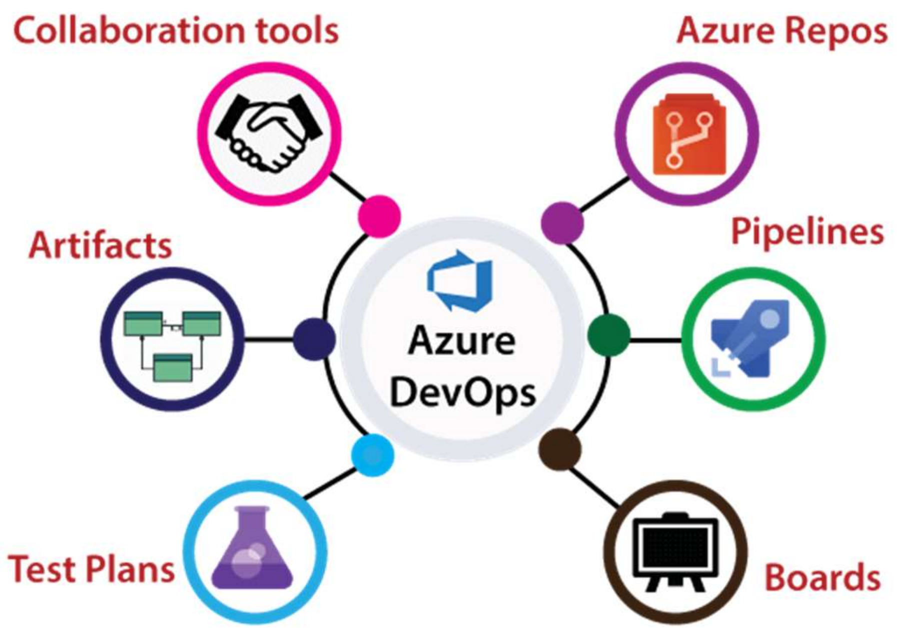
Source: vianalabs.com

jelvix.com

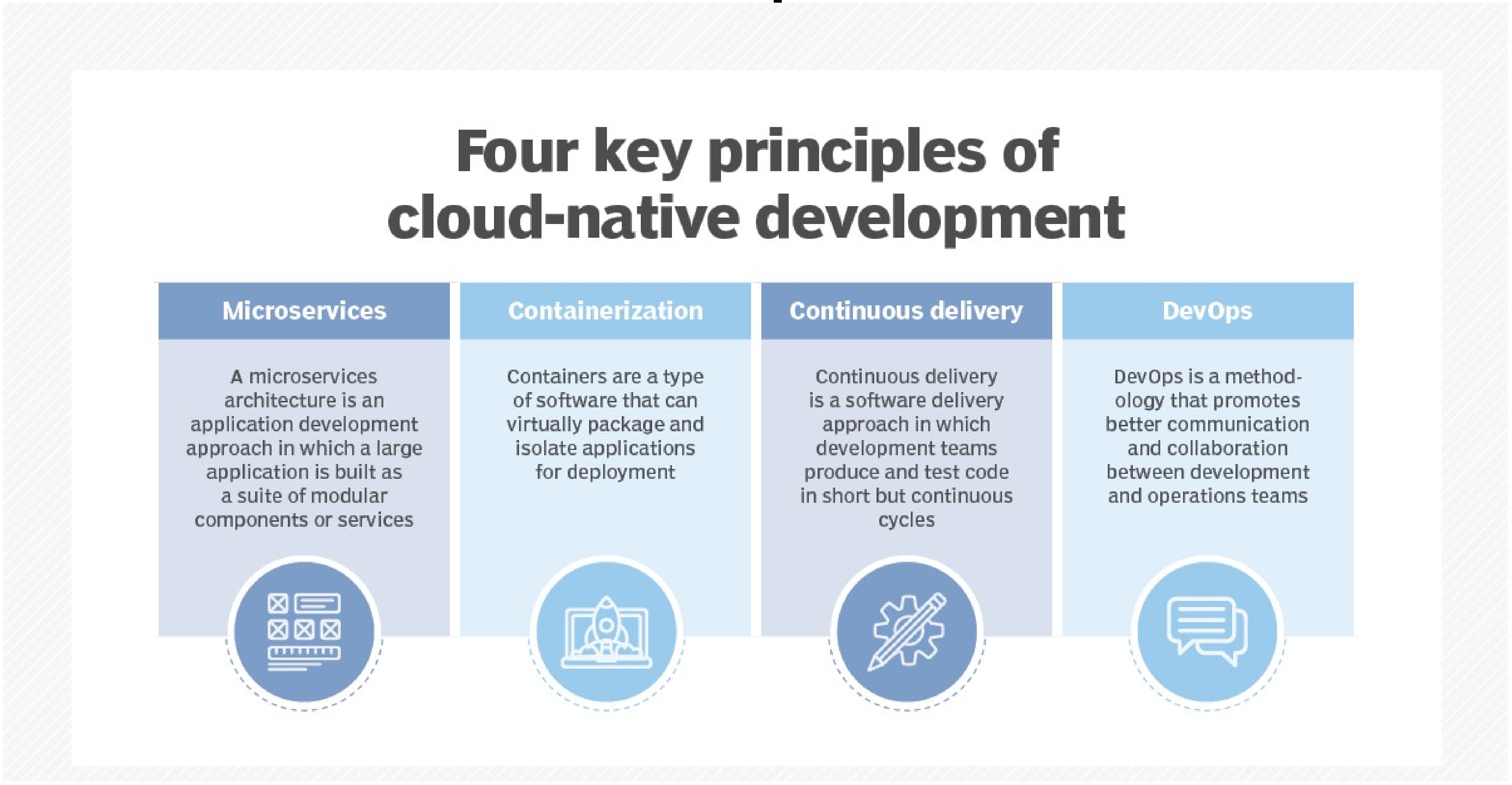
DevOps Toolchain comparison



AWS DevOps



Cloud Native Principles



Four key principles of cloud-native development

Microservices	Containerization	Continuous delivery	DevOps
A microservices architecture is an application development approach in which a large application is built as a suite of modular components or services	Containers are a type of software that can virtually package and isolate applications for deployment	Continuous delivery is a software delivery approach in which development teams produce and test code in short but continuous cycles	DevOps is a method-ology that promotes better communication and collaboration between development and operations teams

TechMonks

12 Factor App Principles

Codebase
One codebase tracked in revision control, many deploys

Dependencies
Explicitly declare and isolate the dependencies

Config
Store configurations in an environment

Backing Services
Treat backing resources as attached resources

Build, release, and, Run
Strictly separate build and run stages

Processes
Execute the app as one or more stateless processes

Port Binding
Export services via port binding

Concurrency
Scale-out via the process model

Disposability
Maximize the robustness with fast startup and graceful shutdown

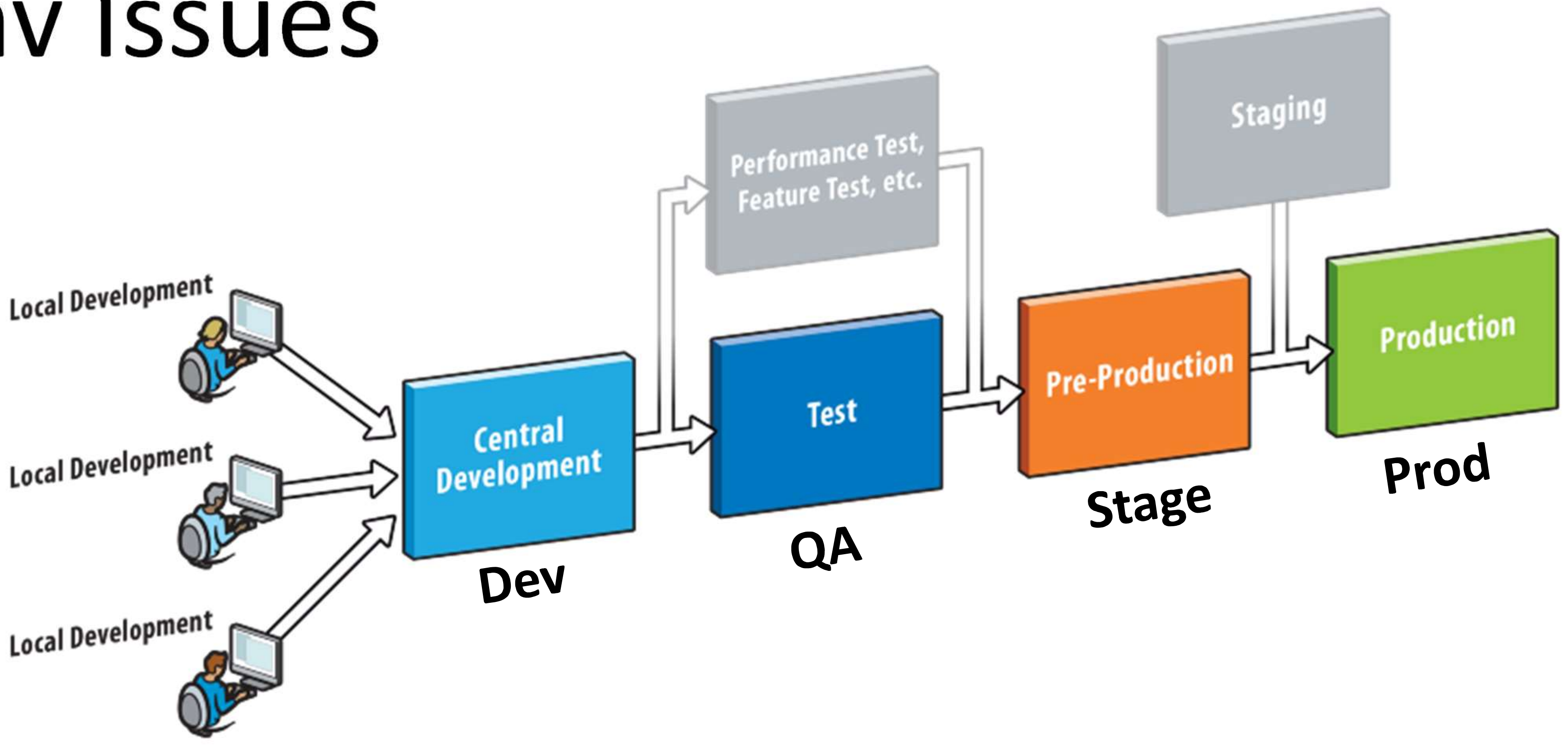
Dev/prod parity
Keep development, staging, and production as similar as possible

Logs
Treat logs as event streams

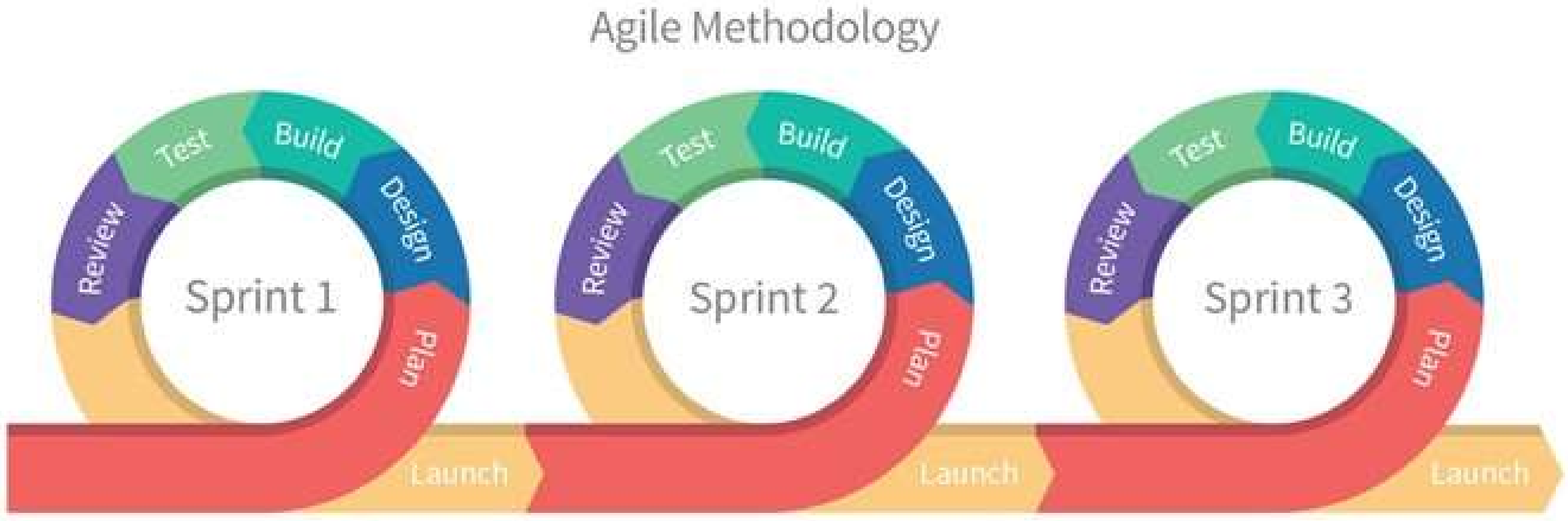
Admin processes
Run admin/management tasks as one-off processes

Questions??

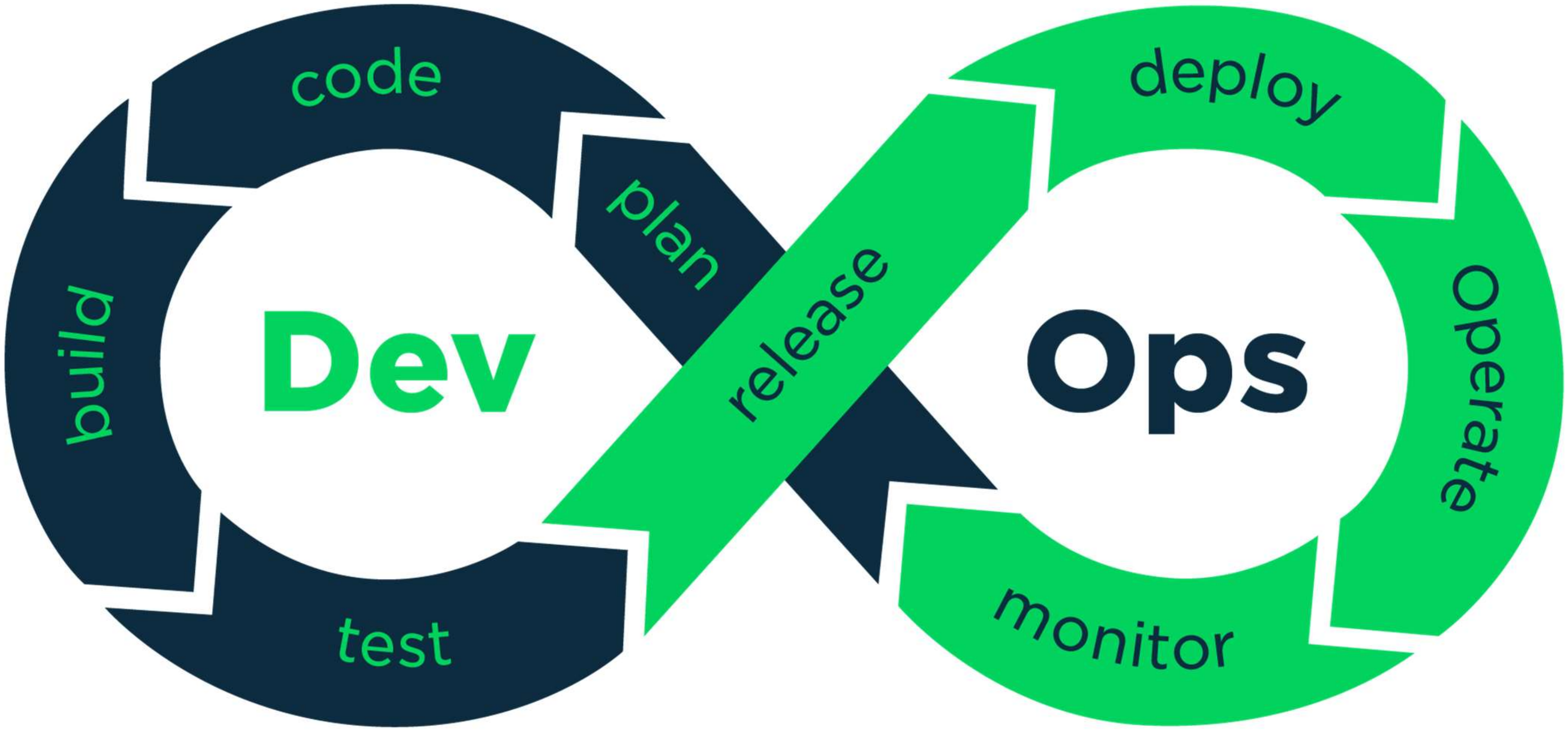
Back-Up Slides ---Waterfall – Env Issues



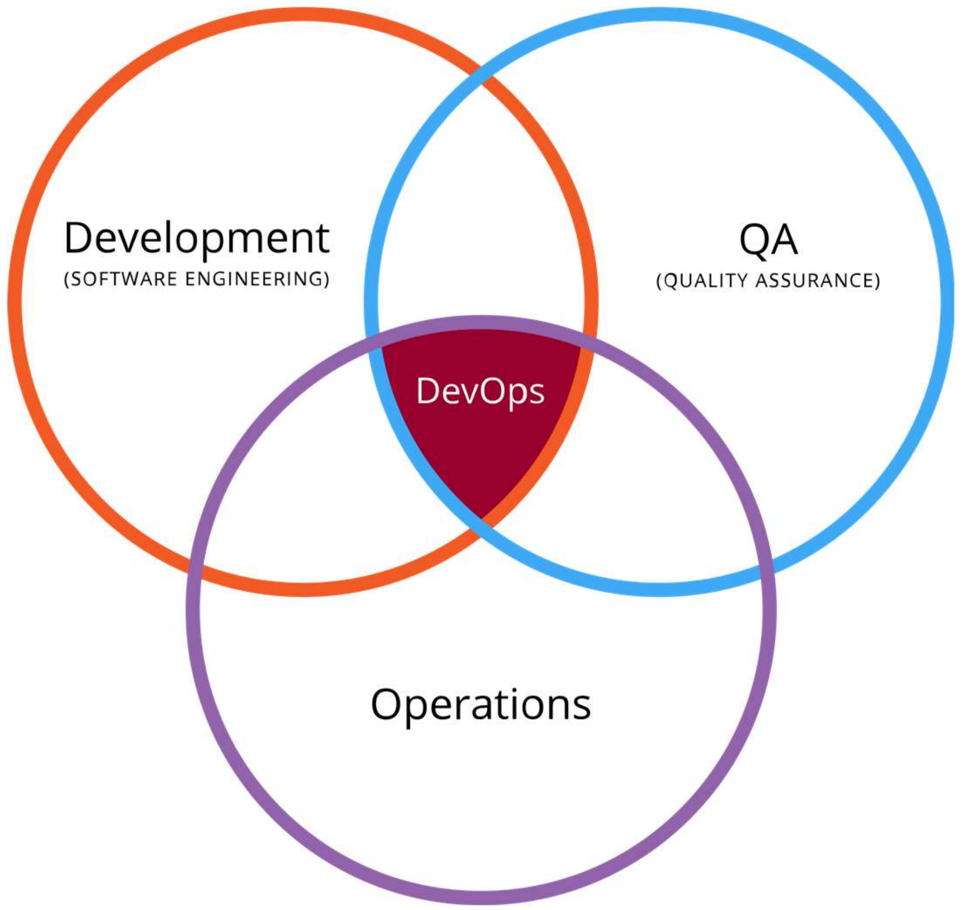
Agile


Agile Methodology
Sprint 1
Plan
Design
Build
Test
Review
Launch
Sprint 2
Plan
Design
Build
Test
Review
Launch
Sprint 3
Plan
Design
Build
Test
Review
Launch

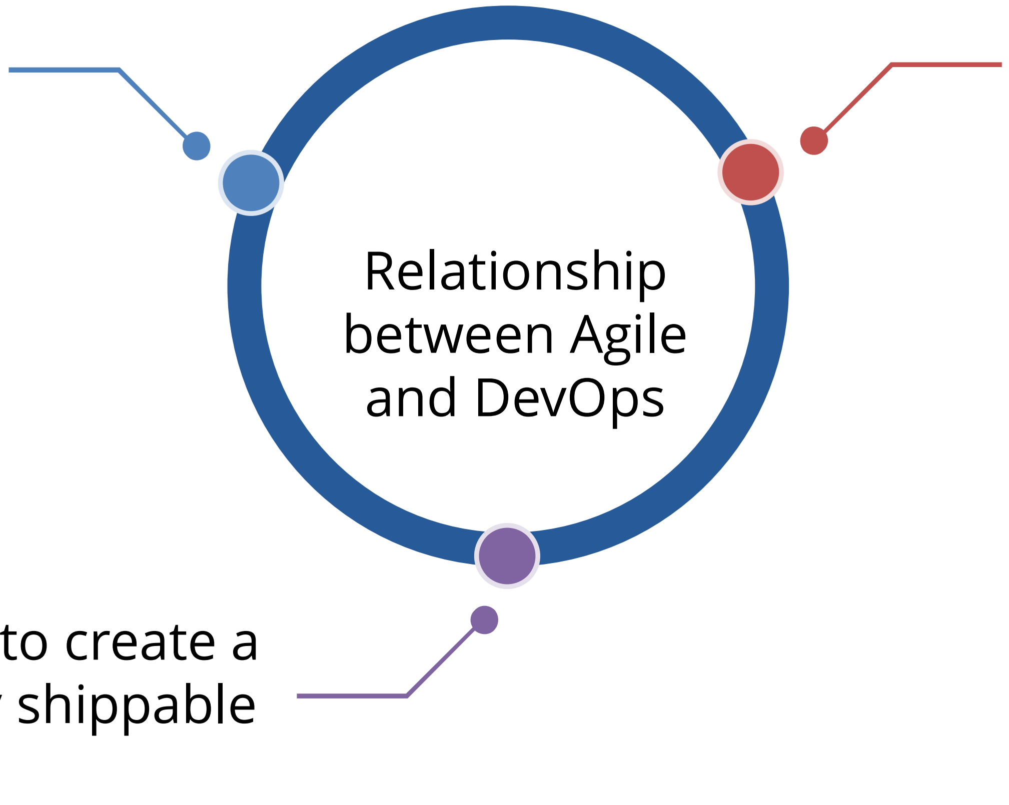
DevOps


code
plan
build
Dev
test
release
deploy
Ops
Operate
monitor

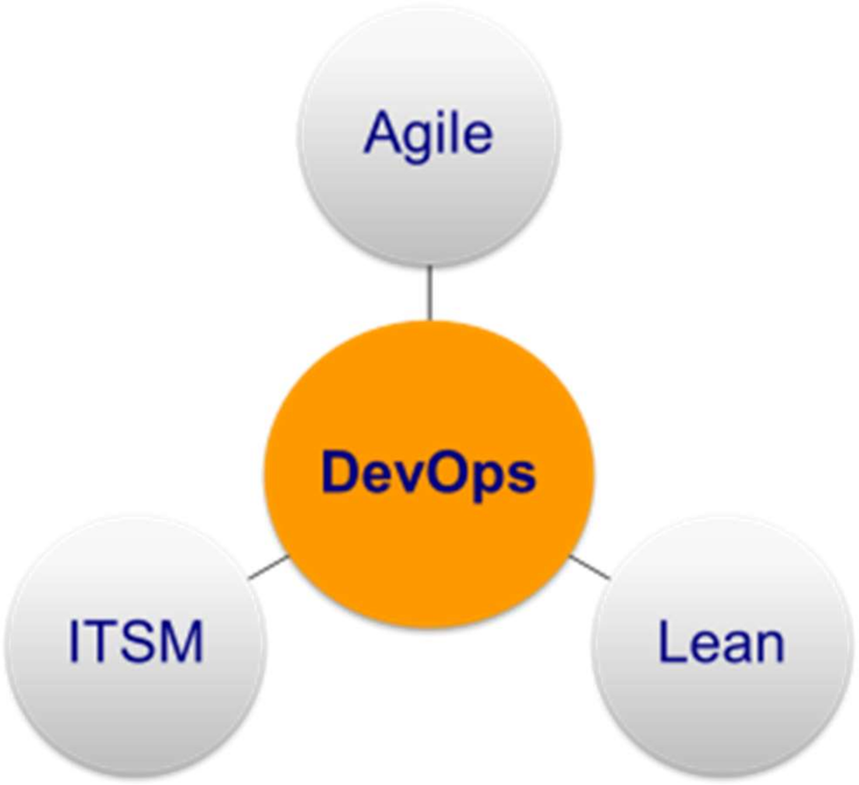
DevOps


Development
(SOFTWARE ENGINEERING)
QA
(QUALITY ASSURANCE)
DevOps
Operations

Agile - DevOps

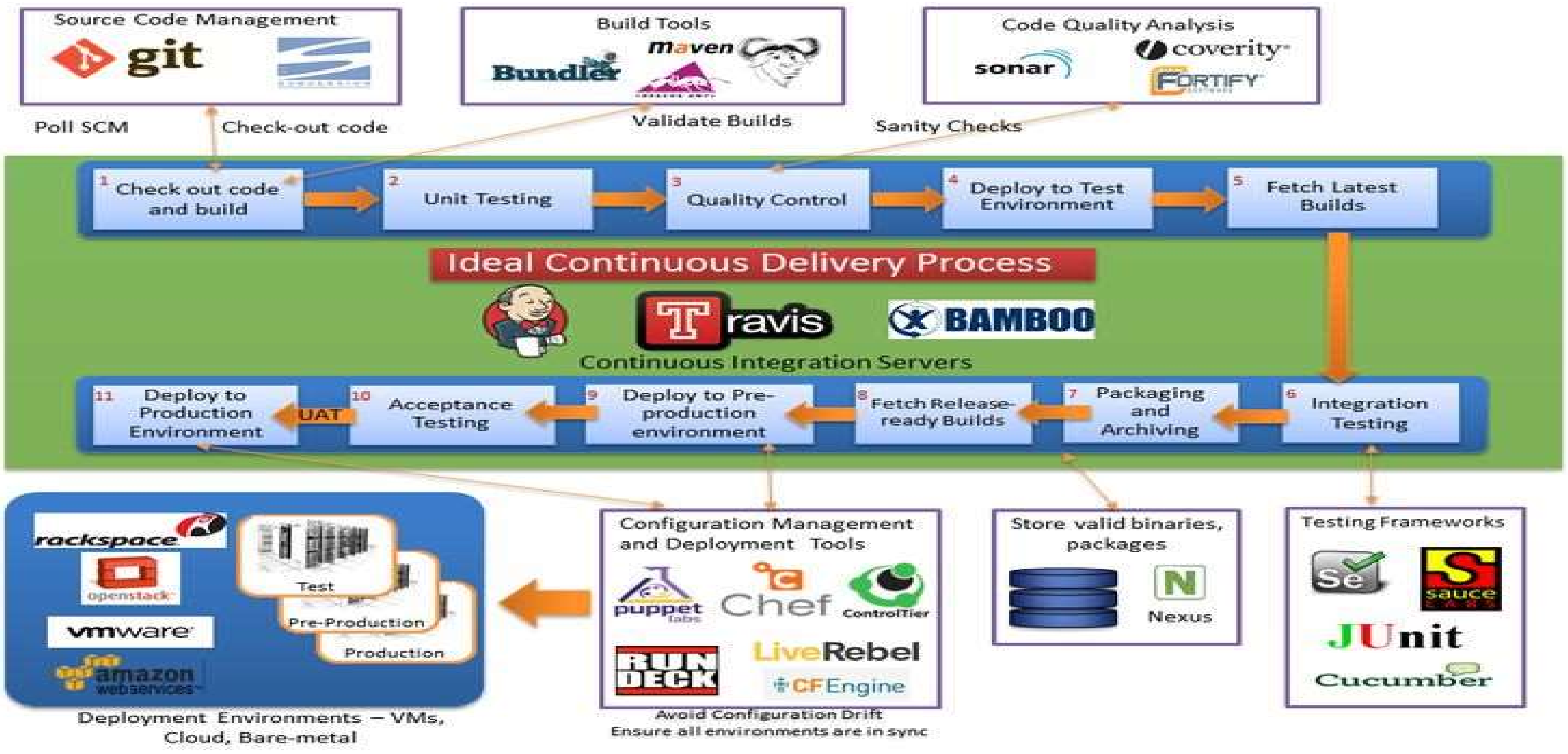

Relationship between Agile and DevOps
Replace non-human steps using tools
Improve the collaboration between all the teams
Automate to create a potentially shippable increment

Agile - DevOps



DevOps Compliments Other SDLC Practices/Methodologies

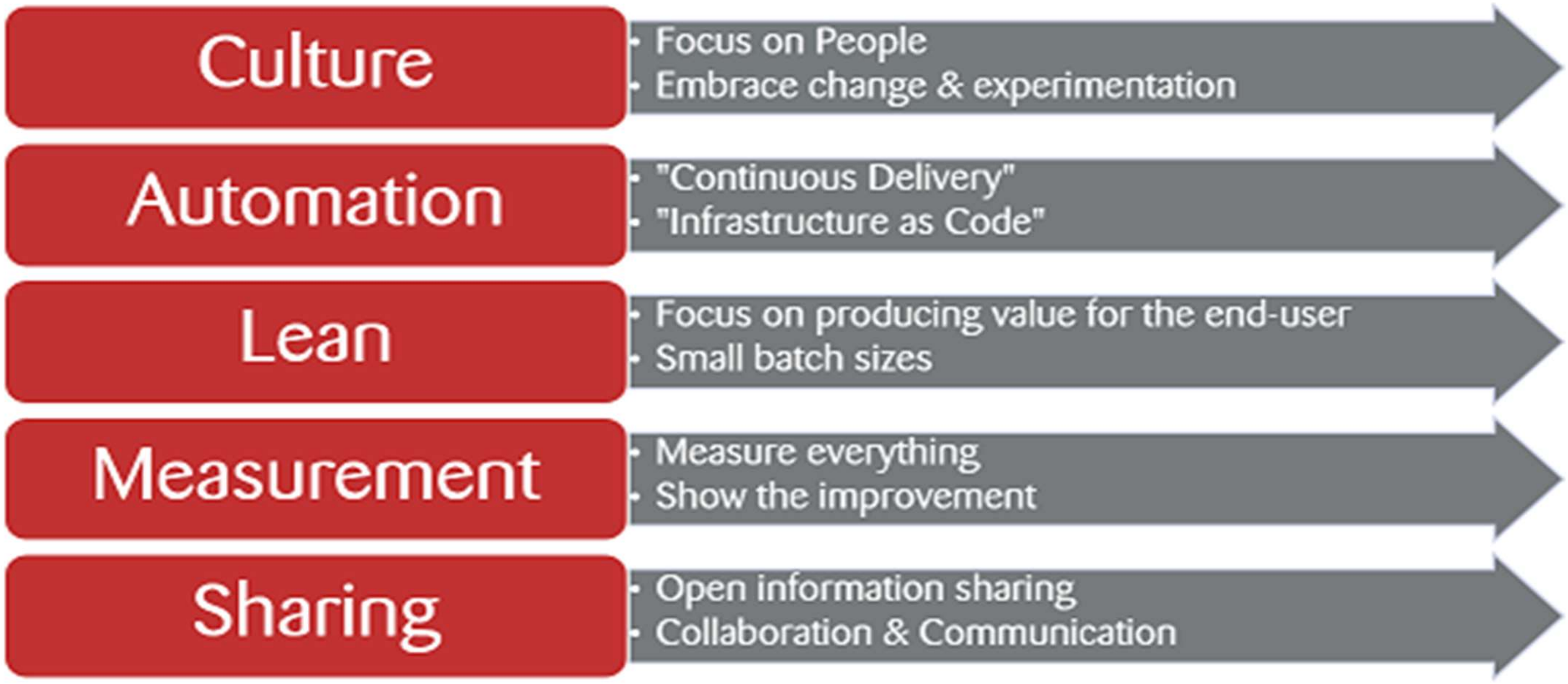
DevOps – Delivery Pipeline



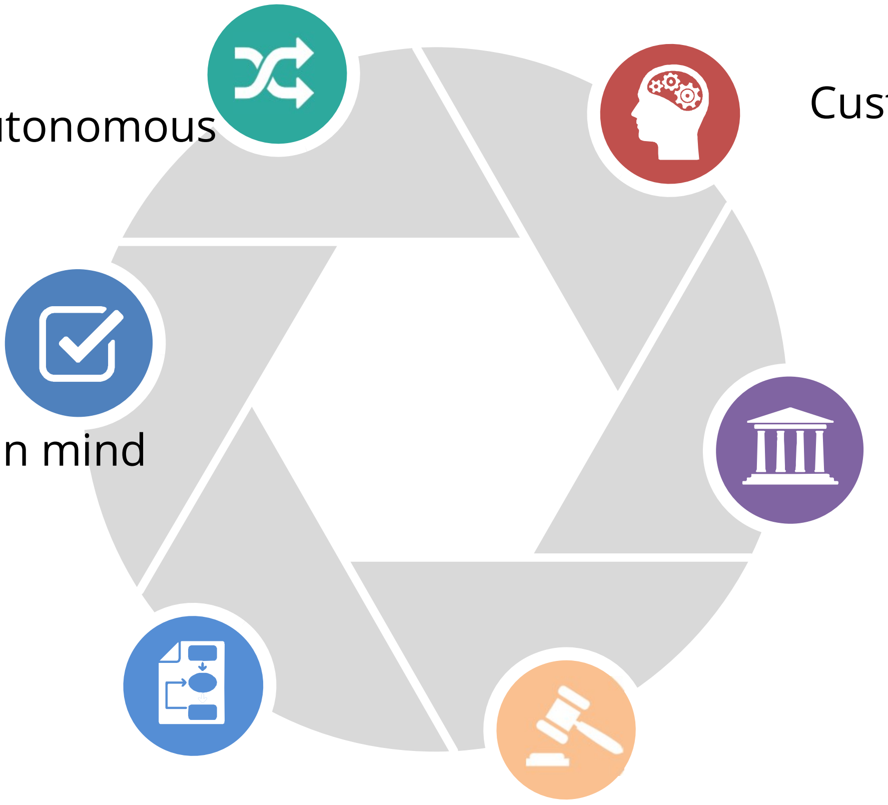
Source Code Management
git
Build Tools
Maven
Bundler
Code Quality Analysis
sonar
coverity
FORTIFY
Poll SCM
Check-out code
Validate Builds
Sanity Checks
1 Check out code and build
2 Unit Testing
3 Quality Control
4 Deploy to Test Environment
5 Fetch Latest Builds
Ideal Continuous Delivery Process
Travis
BAMBOO
Continuous Integration Servers
11 Deploy to Production Environment
UAT
10 Acceptance Testing
9 Deploy to Pre-production environment
8 Fetch Release-ready Builds
7 Packaging and Archiving
6 Integration Testing
rackspace
openstack
vmware
amazon webservices
Test
Pre-Production
Production
Deployment Environments – VMs, Cloud, Bare-metal
Configuration Management and Deployment Tools
puppet labs
Chef
ControlTier
RUN DECK
LiveRebel
CFEngine
Avoid Configuration Drift
Ensure all environments are in sync
Store valid binaries, packages
Nexus
Testing Frameworks
Se
sauce LABS
JUnit
Cucumber

How to Start DevOps?

DevOps Framework



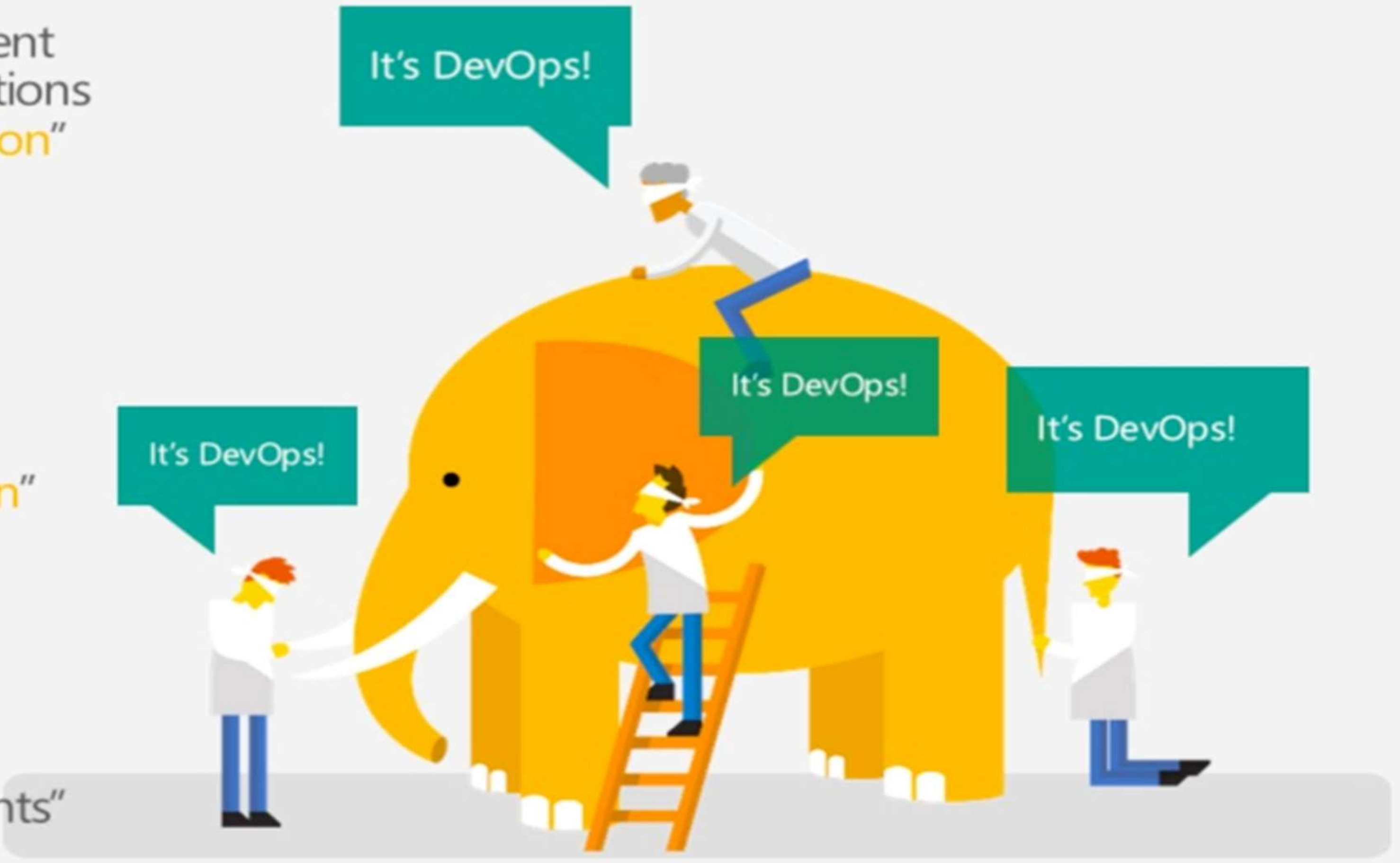
DevOps Principles


Cross-functional autonomous teams
Customer-centric action
Create with the end in mind
End-to-end responsibility
Continuous improvement
Automate everything you can

"**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?"

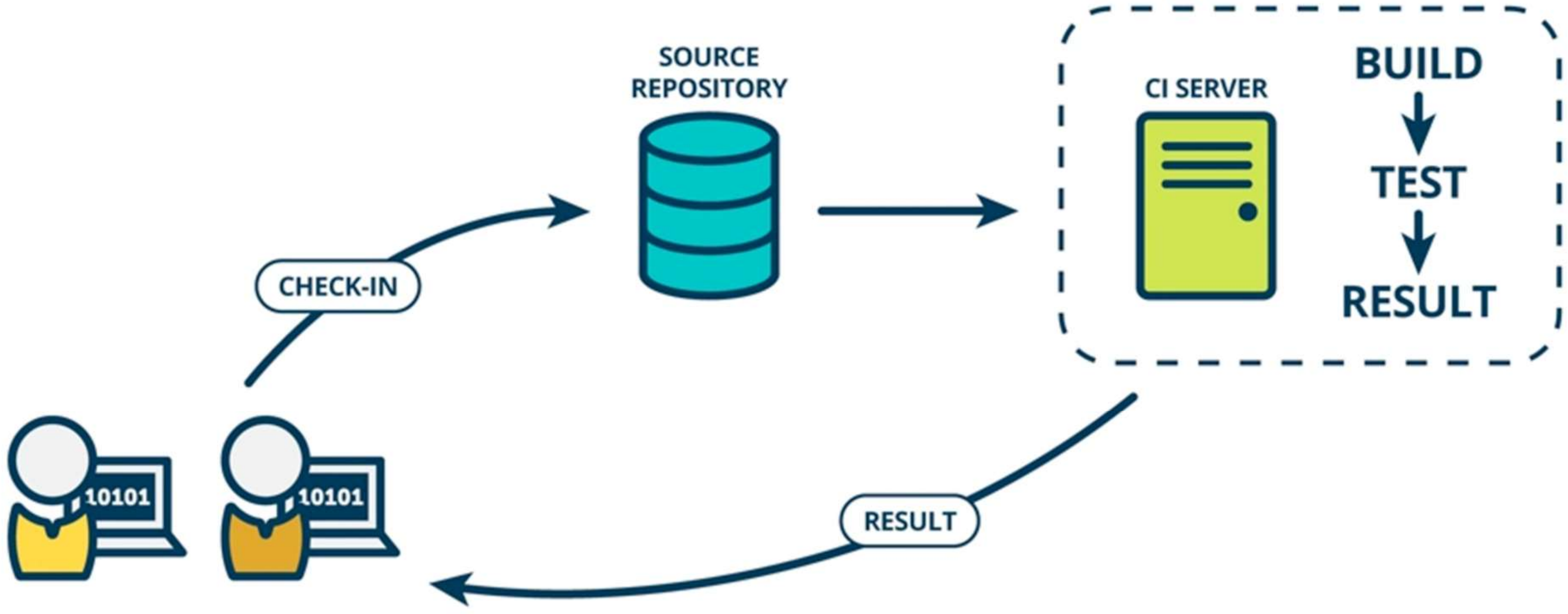
https://channel9.msdn.com/Series/DevOps-Fundamentals/Introduction-to-DevOps

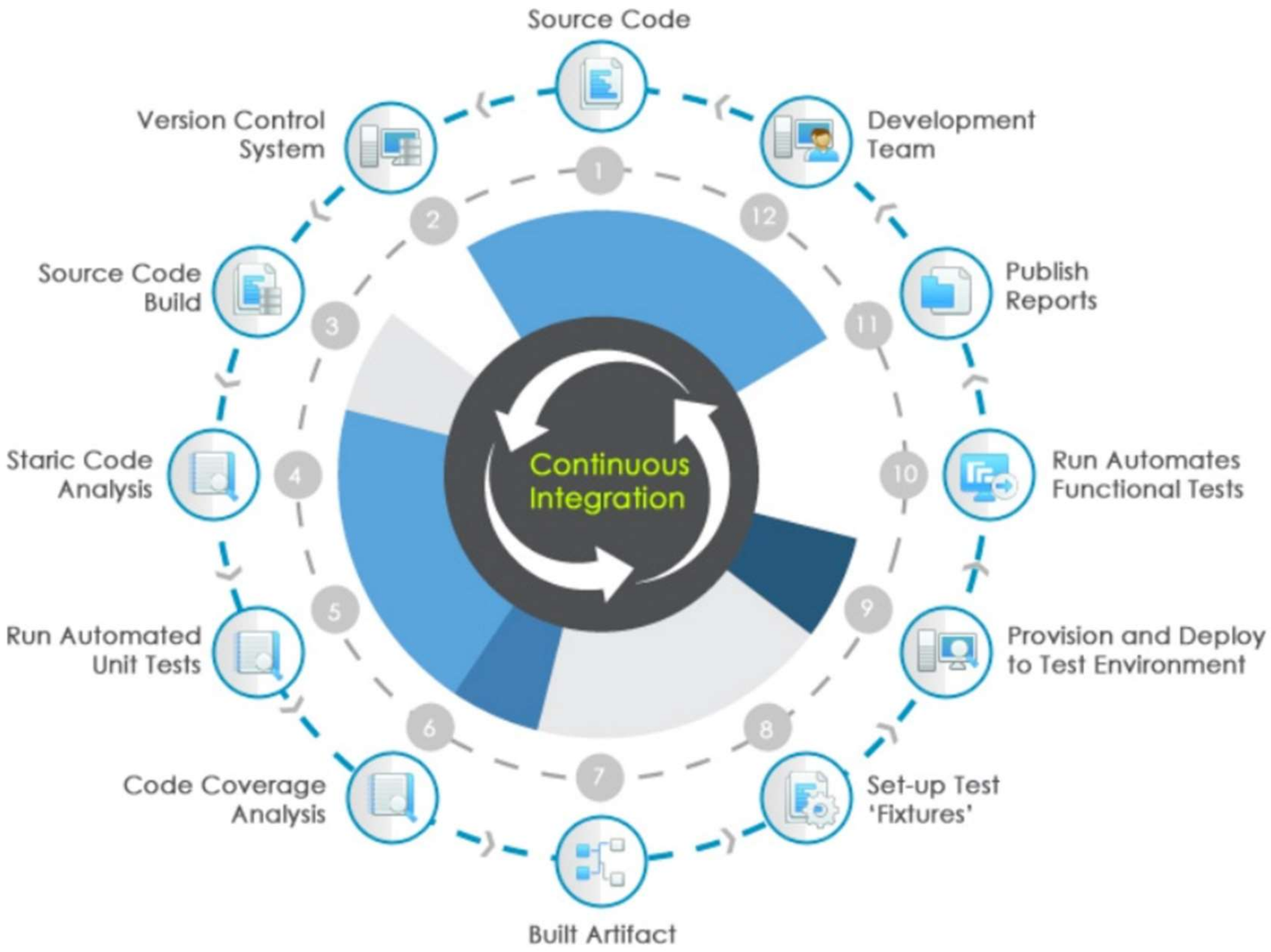
DevOps – Best Practices

<u>Continuous Integration</u> *is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.*

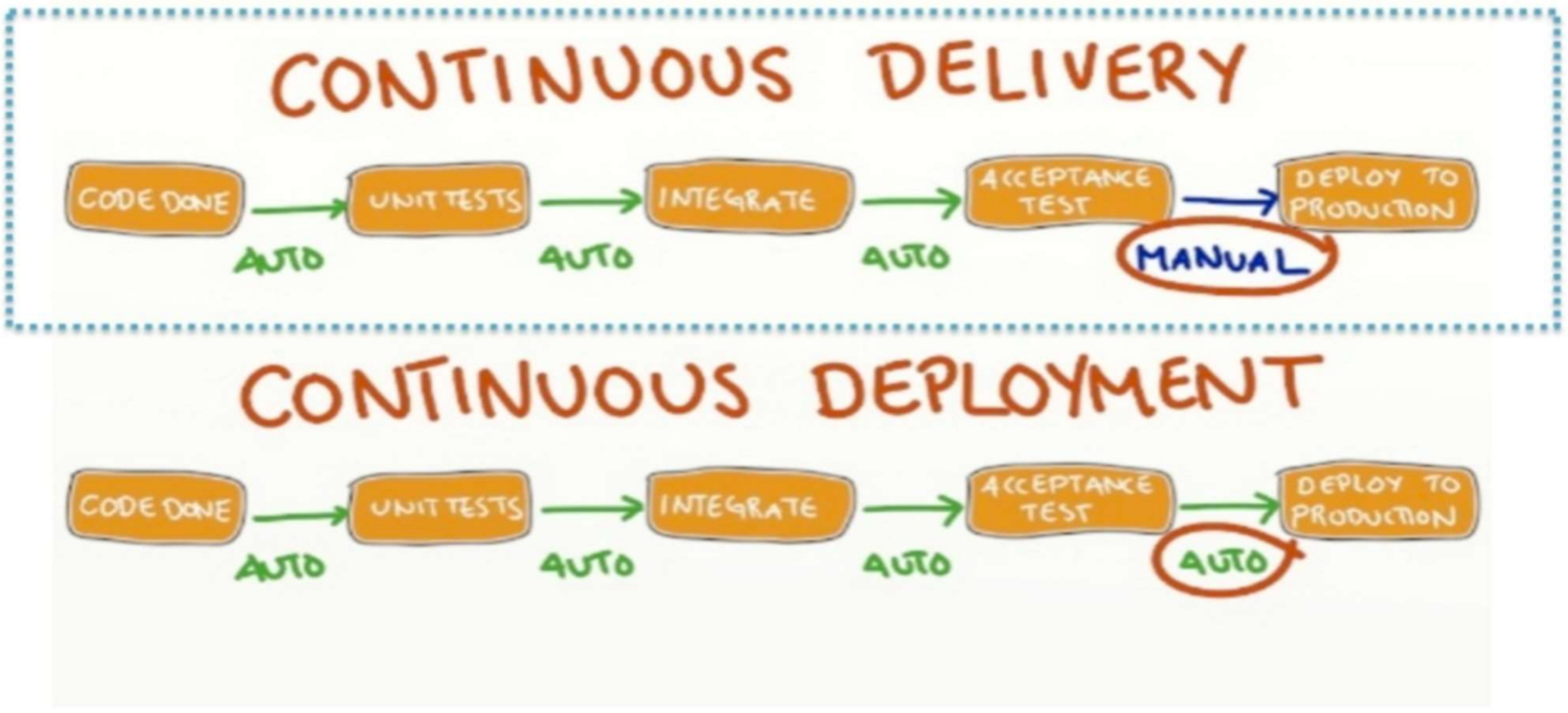
Martin Fowler

Continuous Integration




Source Code
Development Team
Publish Reports
Run Automates Functional Tests
Provision and Deploy to Test Environment
Set-up Test 'Fixtures'
Built Artifact
Code Coverage Analysis
Run Automated Unit Tests
Staric Code Analysis
Source Code Build
Version Control System
Continuous Integration
1
2
3
4
5
6
7
8
9
10
11
12

Continuous Delivery & Deployment



http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment

DevOps – Best Practices

- Everything As Code – Infrastructure, Configuration, Pipeline, Dockerfile, YAML Manifests
- GitOps
- TDD, BDD, CDD

Traditional IT v/s DevOps

	Traditional IT	**DevOps**
Key Driver	Cost	Flow (Time)
Organization	Skill Centric Silos	Autonomous Teams
Batch Size	Large, Monoliths	Micro (MVP)
Scheduling	Complex, Time consuming	Decentralized
Release	High Risk Event	Non Event
Culture	“I did my job”	“its ready to deploy”
Success	Cost	Business KPIs

DevOps – Mind-set Changes

- Speed as a new metrics
- Small batches – create MVP, continuously test & constantly iterate
- Autonomous Teams – cross functional, co-located, dedicated teams with end-to-end responsibility
- Automate – Push button deployments, delivery pipelines for build, test & deploy, rollback etc

DevOps - KPIs

- Deployment Frequency
- Deployment Speed
- Failure Rate
- Time to Recovery

Questions??

Engineering Team Structure

Interesting watch on Spotify Engineering Culture

https://youtu.be/4GK1NDTWbkY