DevOps

Lesson 01:Introduction to DevOps

Capgemini

Lesson Objectives

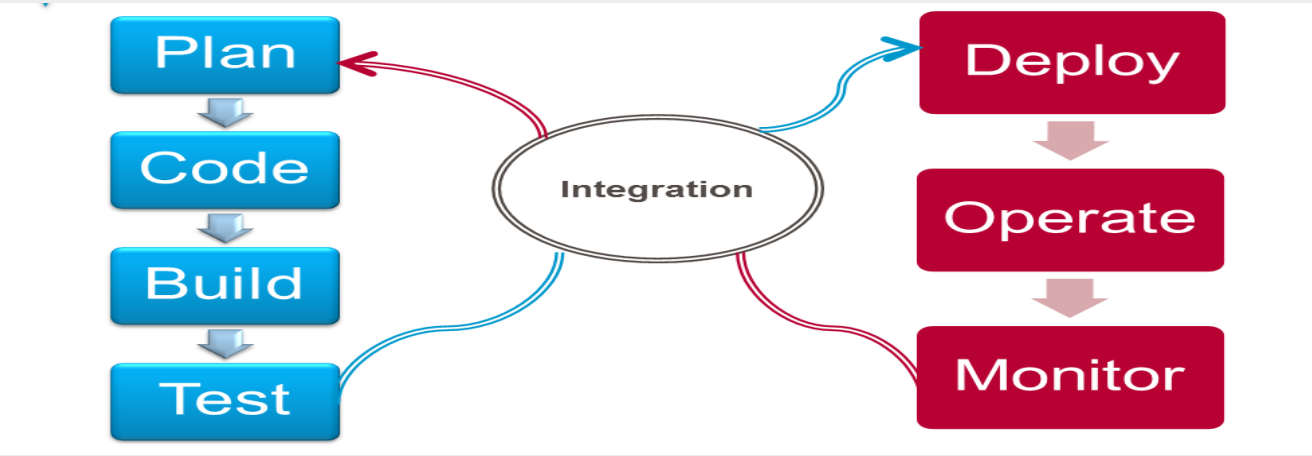
- Introduction to DevOps
  - What is DevOps
  - Evolution of DevOps
  - Agile Methodology
  - Why DevOps
  - Agile vs DevOps
  - DevOps Principles
  - DevOps Lifecycle
  - DevOps Tools
  - Benefits of DevOps
  - Continuous Integration and Delivery pipeline
  - Use-case walkthrough

What is DevOps

- DevOps is a term used to refer to a set of practices that emphasize

the collaboration and communication of both software developers and information technology (IT) professionals while automating the process of software delivery and infrastructure changes.

It aims at establishing a culture and environment where building, testing, and releasing software can happen rapidly, frequently, and more reliably

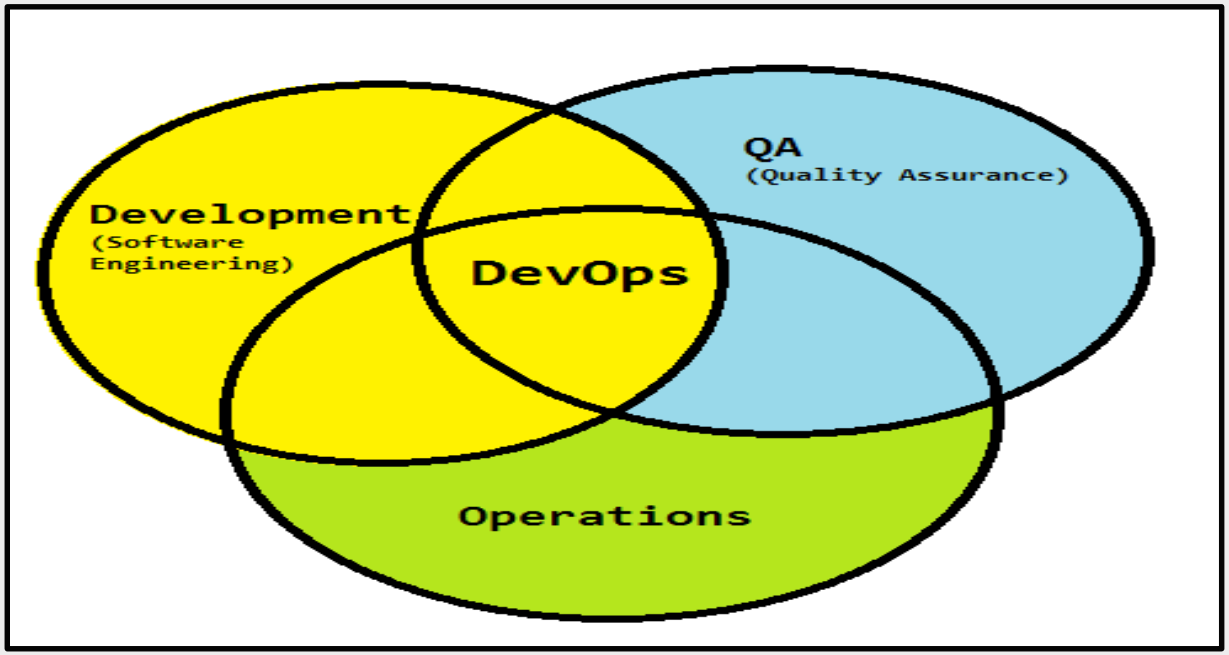


What is DevOps

- **Gartner** defines DevOps as a change in IT culture, focusing on rapid IT service delivery through the adoption of agile, lean practices in the context of a system-oriented approach.
- DevOps emphasizes people (and culture), and seeks to improve collaboration between operations and development teams
- DevOps has many definitions
  - Can be termed as an operational model of collaboration
  - Culture of high performance IT

What is DevOps

- DevOps integrates developers and operations team in order to improve collaboration and productivity by:
  - Automating infrastructure
  - Automating workflows
  - Continuous measuring application performance

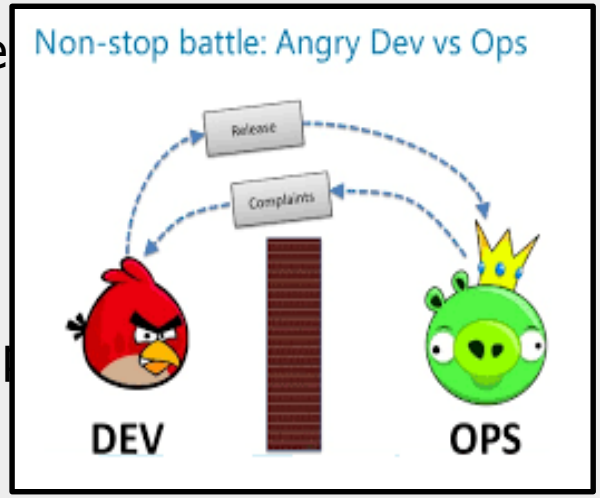


Dev and Ops

- DevOps can be explained simply as operations working together with engineers to get things done faster in an automated and repeatable way.
- **Growing pains of operation**

When we are responsible for large distributed applications the operations complexity grows quickly.

- How do you provision virtual machines?
- How do you configure network devices and serve
- How do you deploy applications?
- How do you collect and aggregate logs?
- How do you monitor services?
- How do you monitor network performance?
- How do you monitor application performance?
- How do you alert and remediate when there are



Dev and Ops

- **Growing pain of developer**
  - Dev is often unaware of Ops roadblocks that prevent the program from working as anticipated
- **Combining the power of developers and operations**

  - The focus on the developer/operations collaboration enables a new approach to managing the complexity of real world operations.
  - The operations complexity breaks down into a few main categories: infrastructure automation, configuration management, deployment automation, log management, performance management, and monitoring.

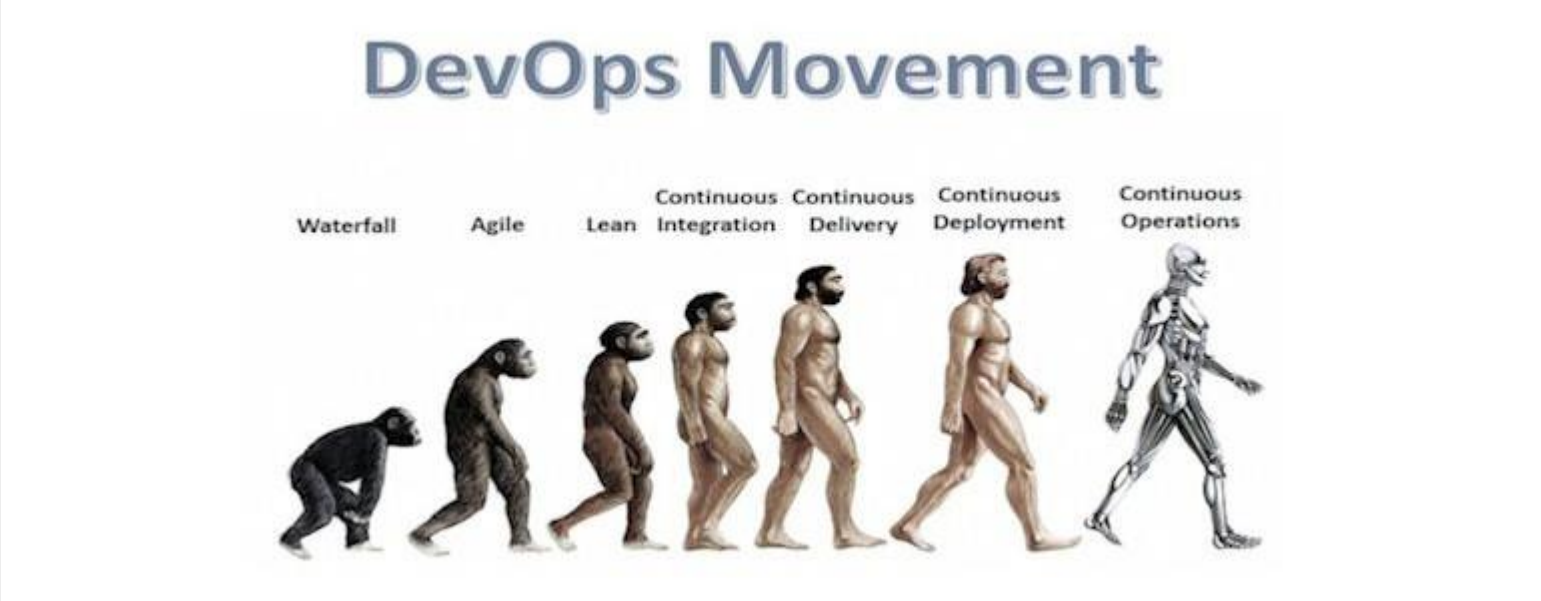
What is DevOps Not ?

- It’s Not (Just) Tools
- It’s Not (Just) Culture
- It’s Not (Just) Devs and Ops
- It’s Not (Just) A Job Title
- It’s Not Everything

Evolution of DevOps

- DevOps is the offspring of agile software development – born from the need to keep up with the increased software velocity and throughput agile methods have achieved.
- The DevOps ideals extend agile development practices by further streamlining the movement of software change thru the build, validate, and deploy and delivery stages, while empowering cross-functional teams with full ownership of software applications – from design thru production support.
- DevOps is an IT mindset that encourages communication, collaboration, integration and automation among software developers and IT operations in order to improve the speed and quality of delivering software.
- DevOps teams focus on standardizing development environments and automating delivery processes to improve delivery predictability, efficiency, security and maintainability.
- The DevOps ideals provide developers more control of the production environment and a better understanding of the production infrastructure

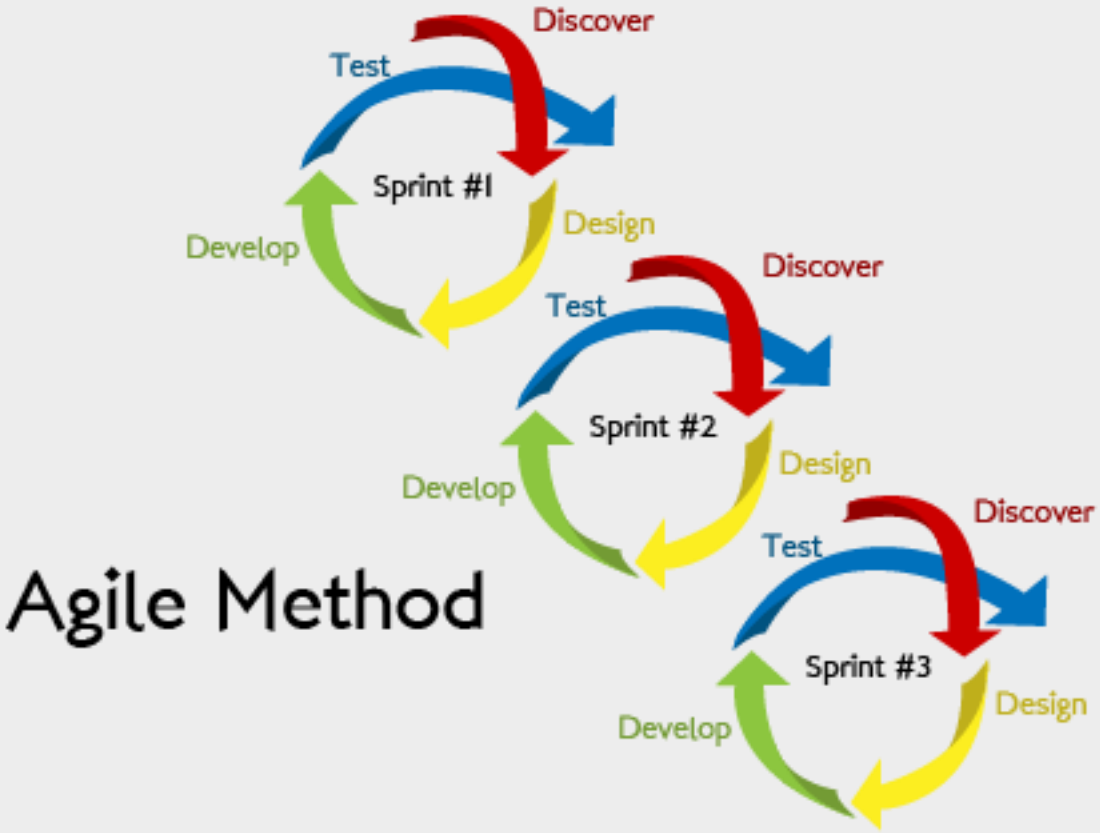
DevOps Movement



Agile Methodology

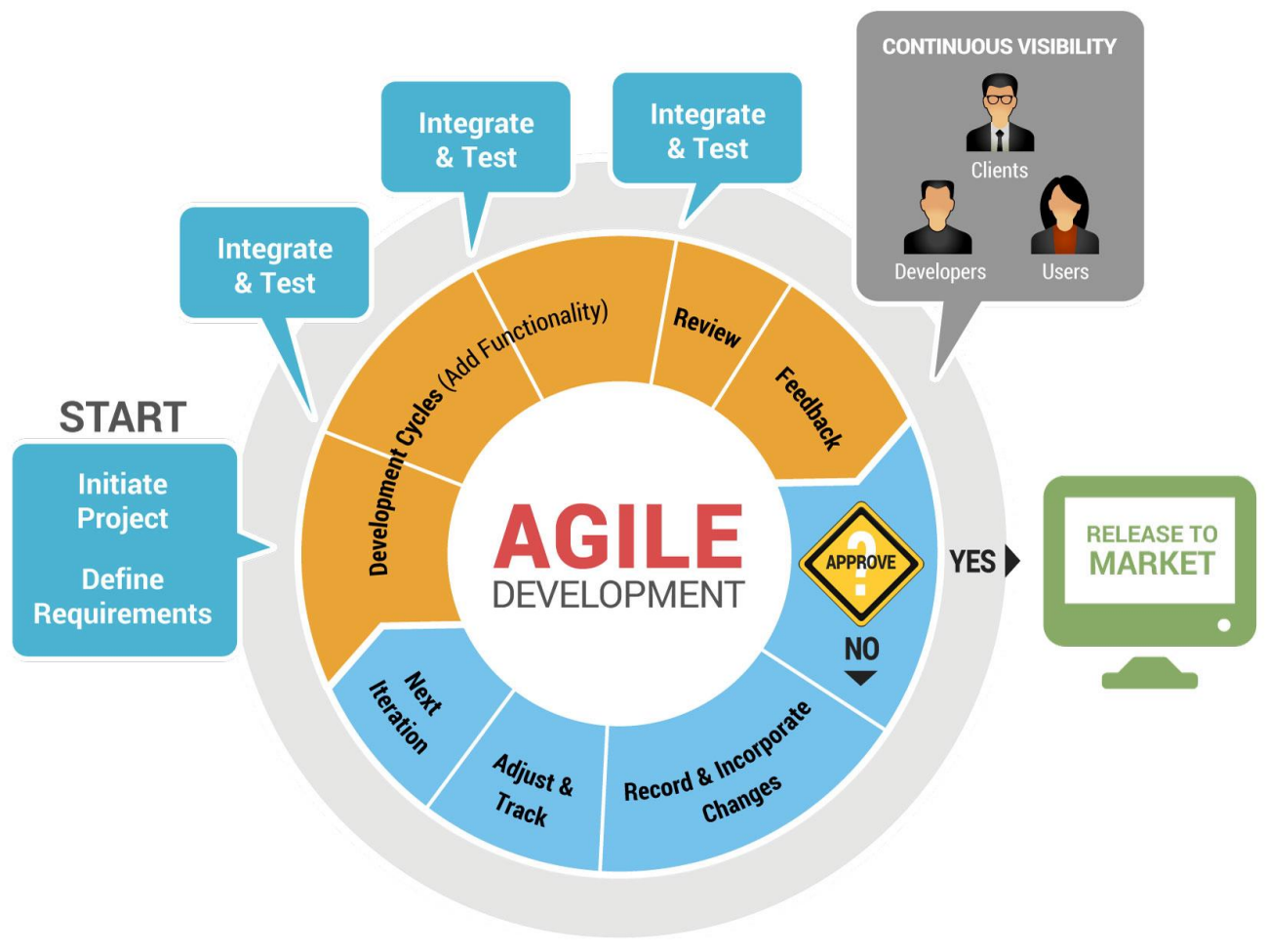
- **Agile Methodology**: In the Agile approach, software is developed and released incrementally in the iterations
- This results in small incremental releases with each release building on previous functionality
- Each release is thoroughly **tested** to ensure **software quality** is maintained
- It is used for time critical applications

Agile Methodology



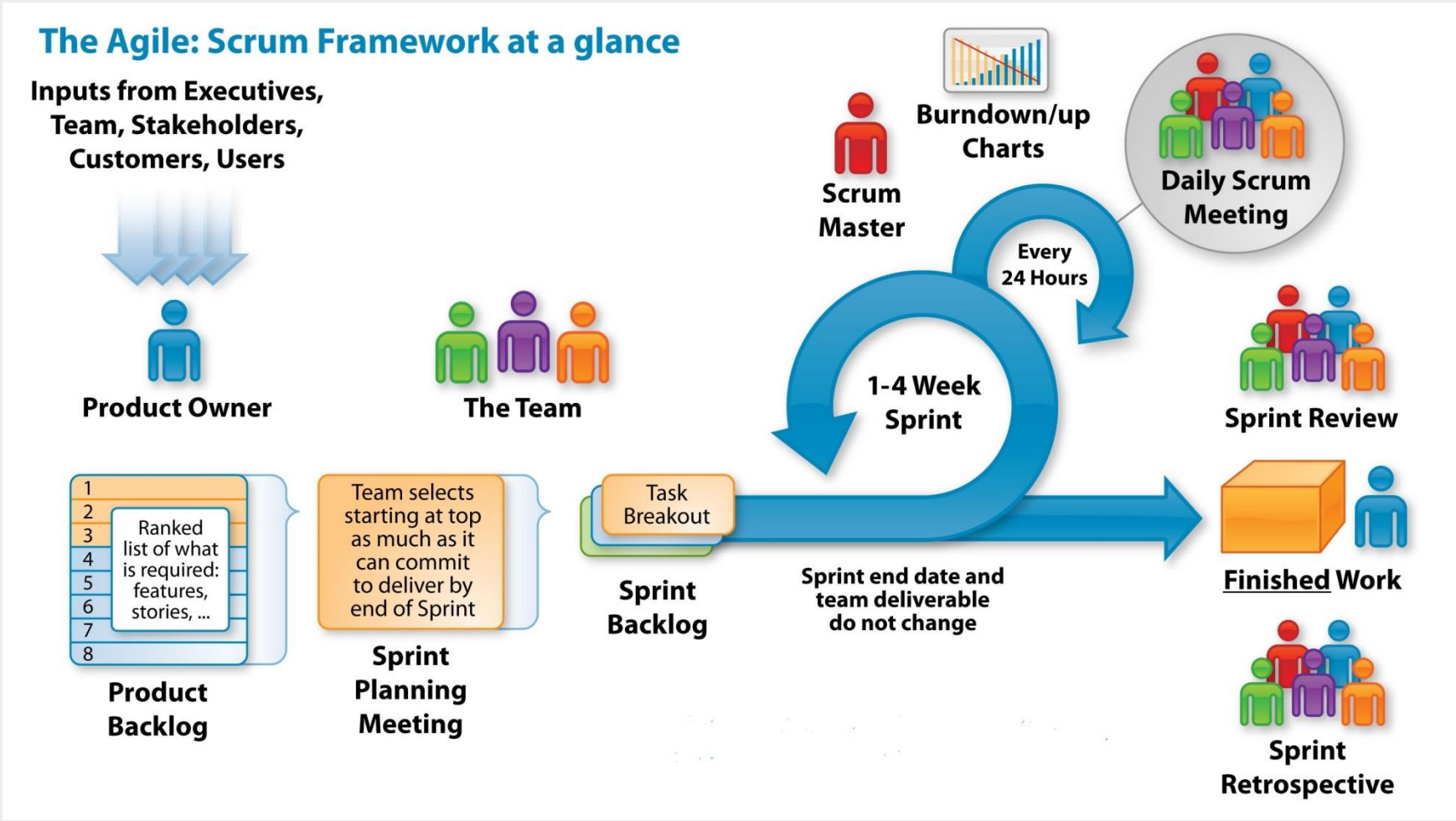
http://www.cleanri.com/post_agile-methodology-diagram_43651/

Agile Methodology



http://www.cleanri.com/post_agile-methodology-diagram_43651/

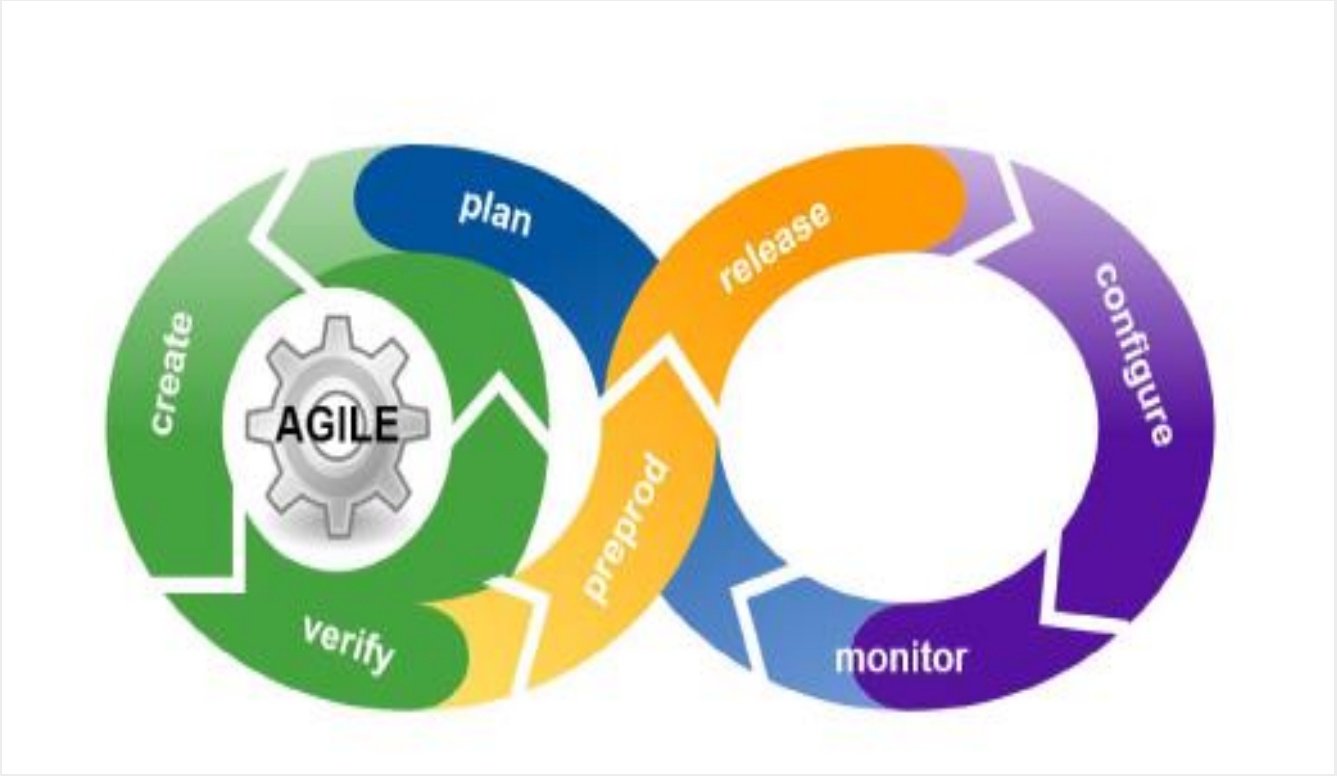
Agile- Scrum Framework



http://www.cleanri.com/post_agile-methodology-diagram_43651/

Agile Practices

- **Development** is Agile, what about Agility in **Operations**?

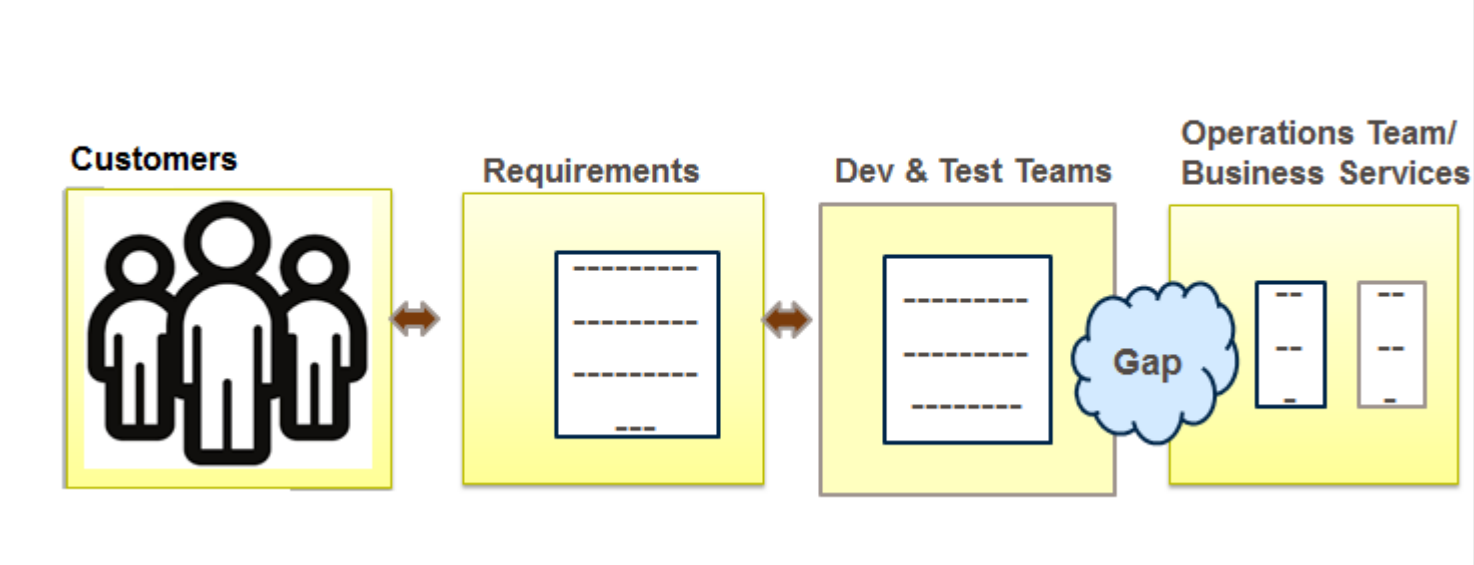


Agile Practices

Evolution of Software Development

- DevOps evolved from existing software development strategies/methodologies over the years in response to business needs
- The slow and cumbersome **Waterfall** model evolved into **Agile**
- While this Agile SCRUM approach brought agility to development, it was lost on Operations which did not come up to speed with Agile practices
- Lack of collaboration between Developers and Operations Engineers still slowed down the development process and releases
- **DevOps methodology** was born out of this need for better collaboration and faster delivery
- DevOps enables continuous software delivery with less complex problems to fix and faster resolution of problems

Why DevOps

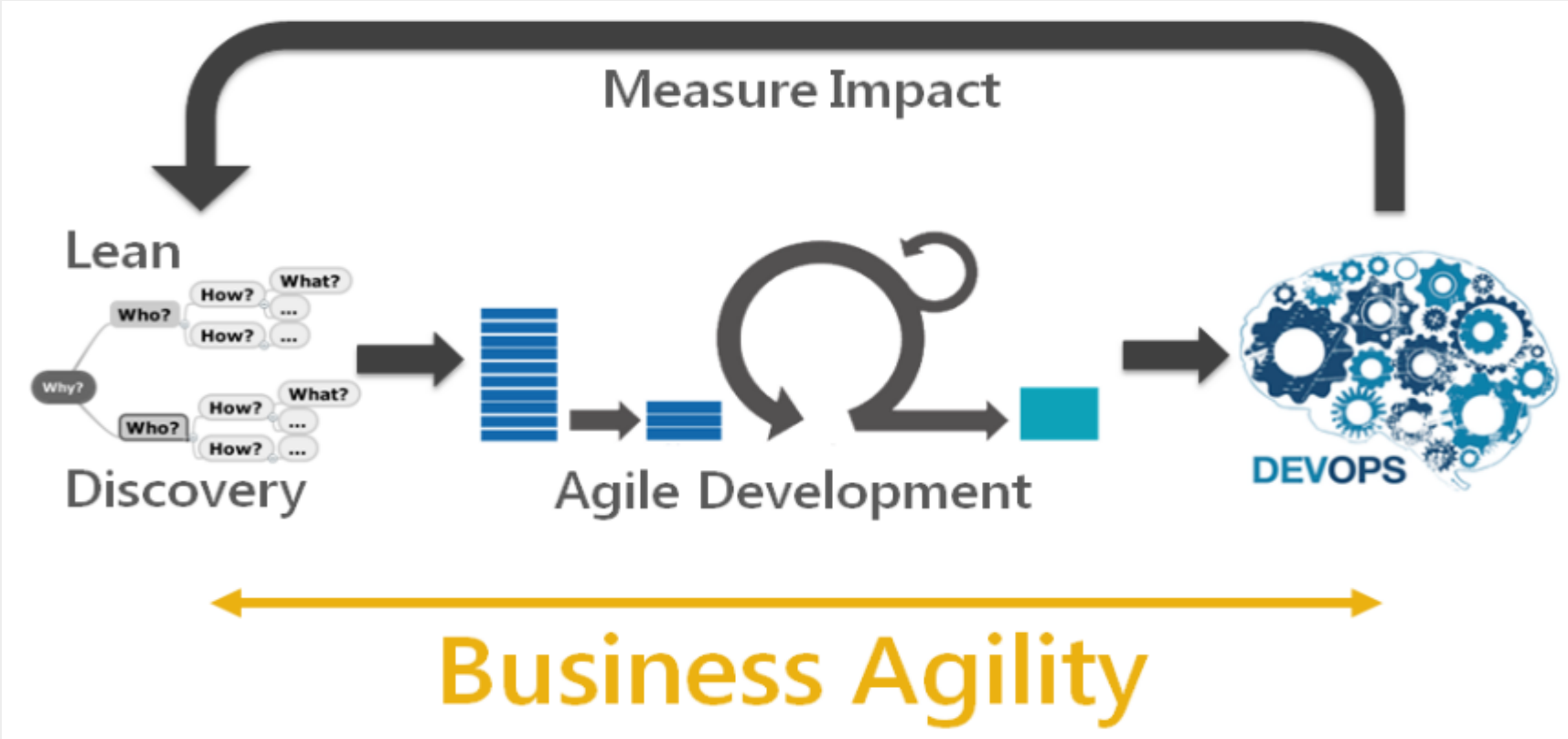

Customers
Requirements
Dev & Test Teams
Operations Team/
Business Services
Gap

Why DevOps

- DevOps takes care of the challenges faced by Development and Operations
- Below table describes how DevOps addresses Dev Challenges

	Dev Challenges	DevOps Solution
	Waiting time for code deployment	• Continuous Integration: Ensures there is a quick deployment of code, faster testing and speedy feedback mechanism. • No waiting time to deploy the code. Hence developers focuses on building the current code.
	Pressure of work on old, pending and new code	
	Tools to automate infrastructure management are not effective	Configuration Management: Helps to organize and execute configuration plans and consistently provision the system
	No. of servers to be monitored increases	Continuous Monitoring: Effective monitoring and feedbacks system is established through Nagios. Thus effective administration is achieved
	Difficult to diagnose and provide feedback on the product	

Emergence of DevOps: Influence on DevOps



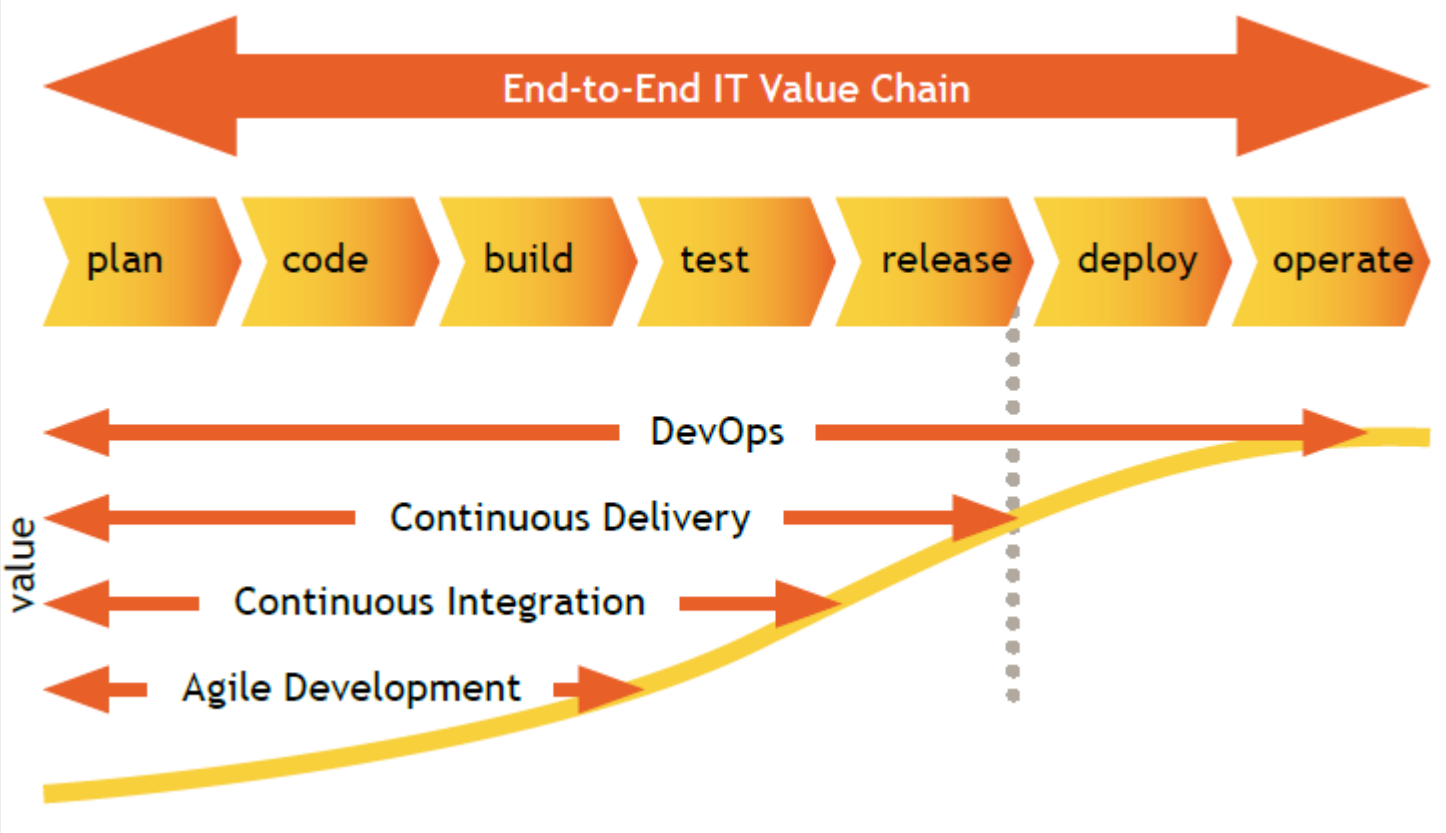
https://theagileadmin.com/what-is-devops/

Agile Vs DevOps

AGILE MINDSET IN THE END-TO-END CHAIN		
Customer Satisfaction	over	SLA Compliance
Attitude & Collaboration	over	Certification
Control on Results	over	Control on Activities
Adaptivity	over	Procedures

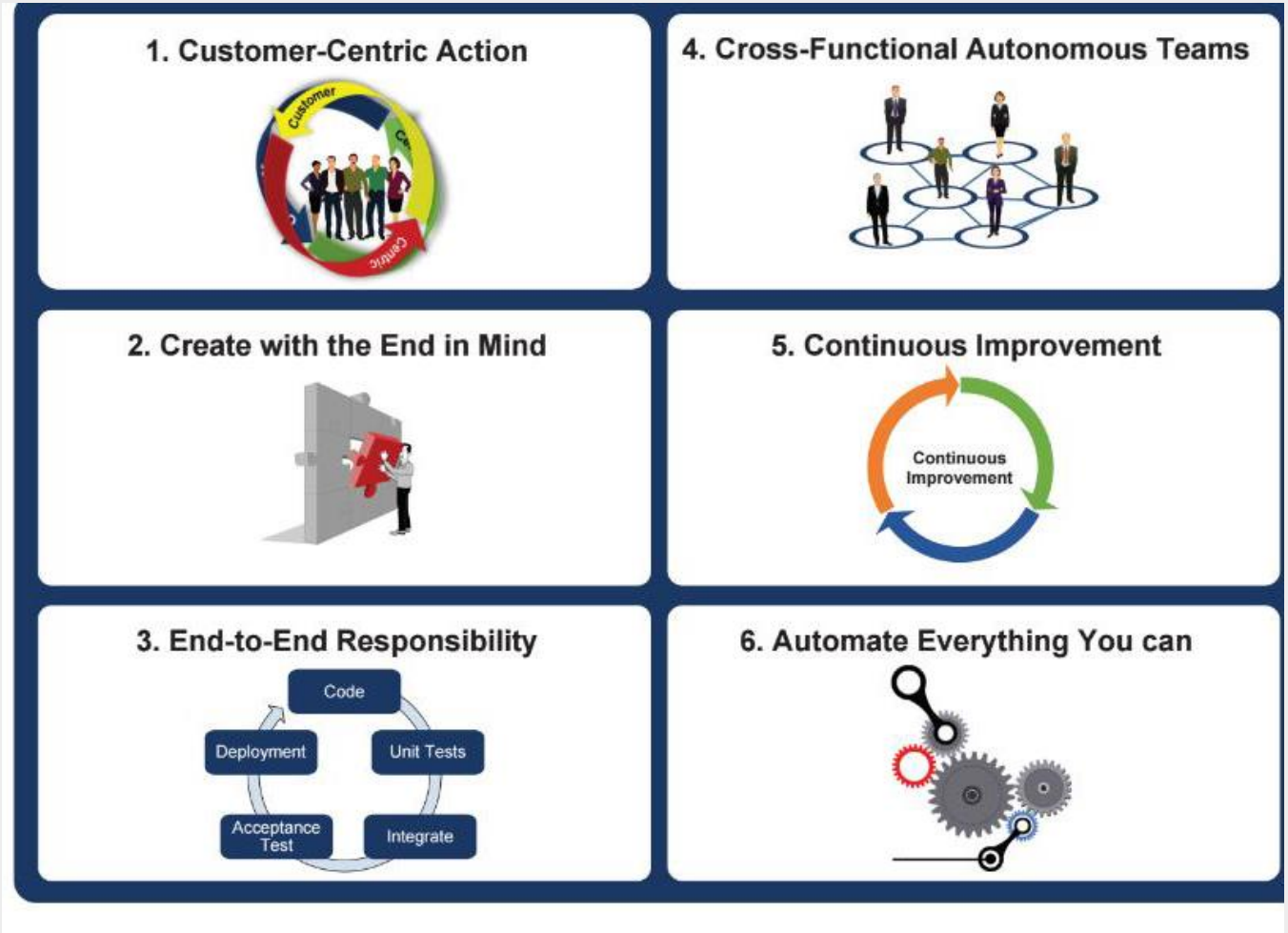
http://labs.sogeti.com/wp-content/uploads/2016/03/D2D-4-EN-web.pdf

Agile Vs DevOps



http://labs.sogeti.com/wp-content/uploads/2016/03/D2D-4-EN-web.pdf

DevOps Principles



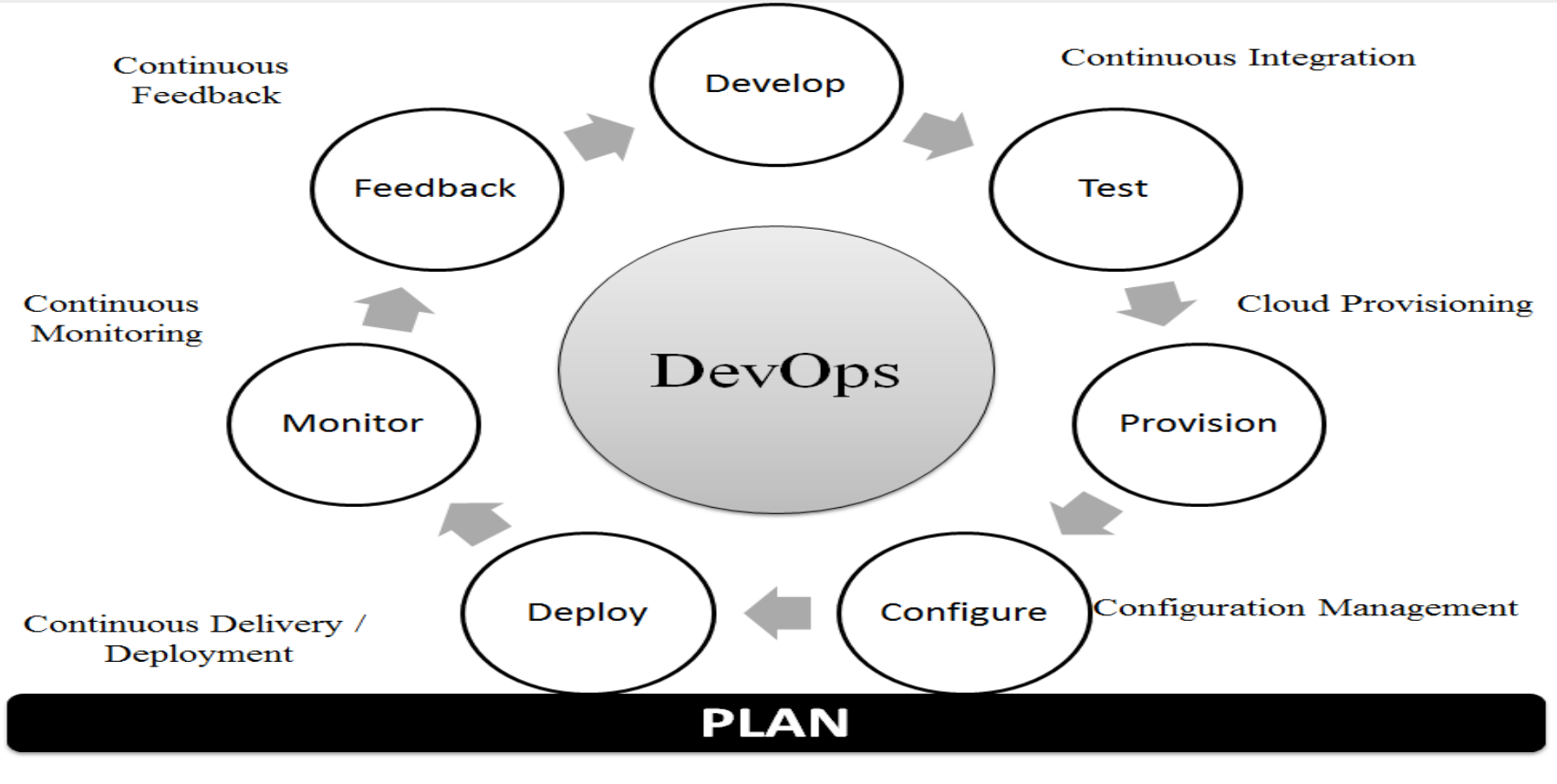
DevOps Life Cycle- it's all about "continuous"

- DevOps Life Cycle can be broadly broken down into the below stages:

- **Continuous Development** :In this stage of DevOps life cycle the Software is developed continuously
- **Continuous Integration**: In this stage the code supporting new functionality is integrated with the existing code
- **Continuous Testing**: In this stage the developed software is continuously tested for bugs
- **Continuous Monitoring**: This practice involves the participation of the Operations team who will monitor the user activity for bugs / any improper behavior of the system

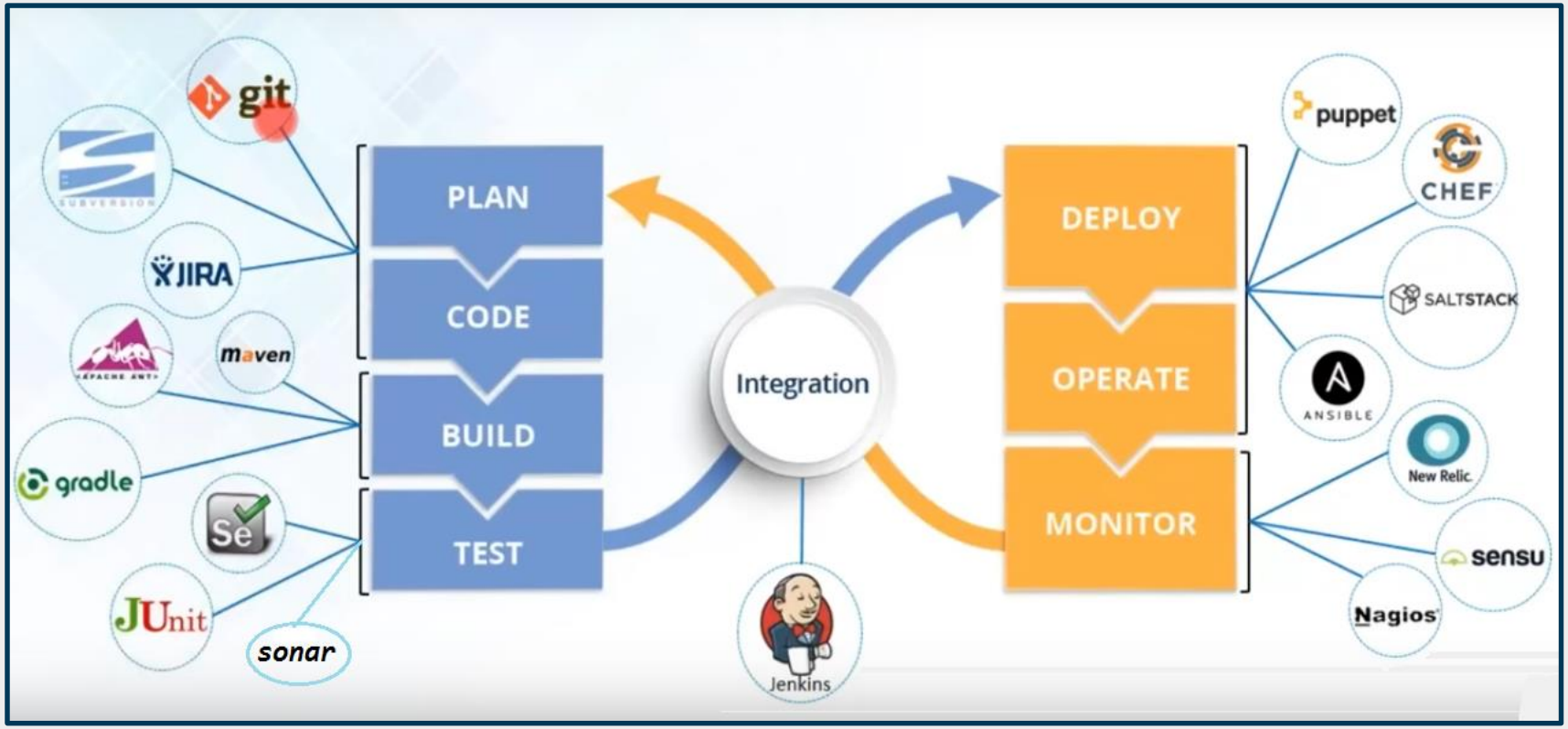
DevOps SDLC

- DevOps life cycle and Software Development stages depicted in the diagram below.



DevOps Tools

- The below diagram shows which tools can be used in which stage of the DevOps life cycle.



Tools used in various stages of Project Lifecycle

Project life cycle phase	Purpose	Tools that can be used
Development	Source control management	GIT,SVN(Apache SubVersion),CVS(Concurrent Version System)
	Automated static code analysis	Checkstyle,PMD,FindBugs,SonarQube
	Implementation of continuous integration	Jenkins,Hudson,Puppet,Anthill
Testing	Unit,performance,web,and services testing	Junit,TestNG,Jmeter,Selenium,SOAPUI
	Code coverage	Jacaco,Cobertura
Release	Build and release activities	ANT,Maven,Gradle
Deployment	Automation of deployment activity	Puppet, Chef, SaltStack and Ansible
Monitoring and maintenance	Continuously monitor the application and server environment post application deployment	Nagios, NewRelic and Sensu

DevOps Tools

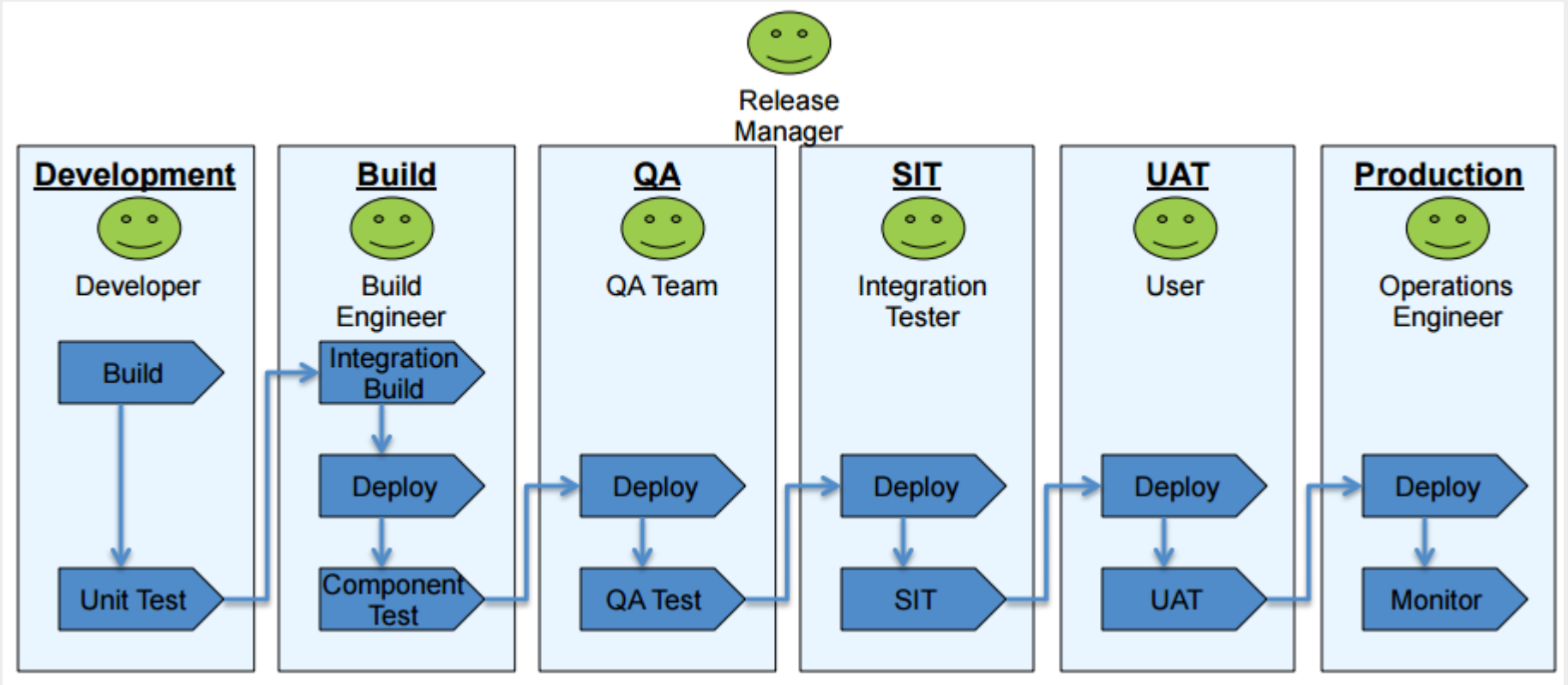
- Because DevOps is a cultural shift and collaboration (between development, operations and testing), there is no single "DevOps tool": it is rather a set (or "DevOps toolchain"),consisting of multiple tools.
- DevOps is not about using one particular tool, it is about using the right combination of tools to accelerate development and mitigate risk. This list captures some of emergent DevOps tools categorized by purpose:

DevOps Tools

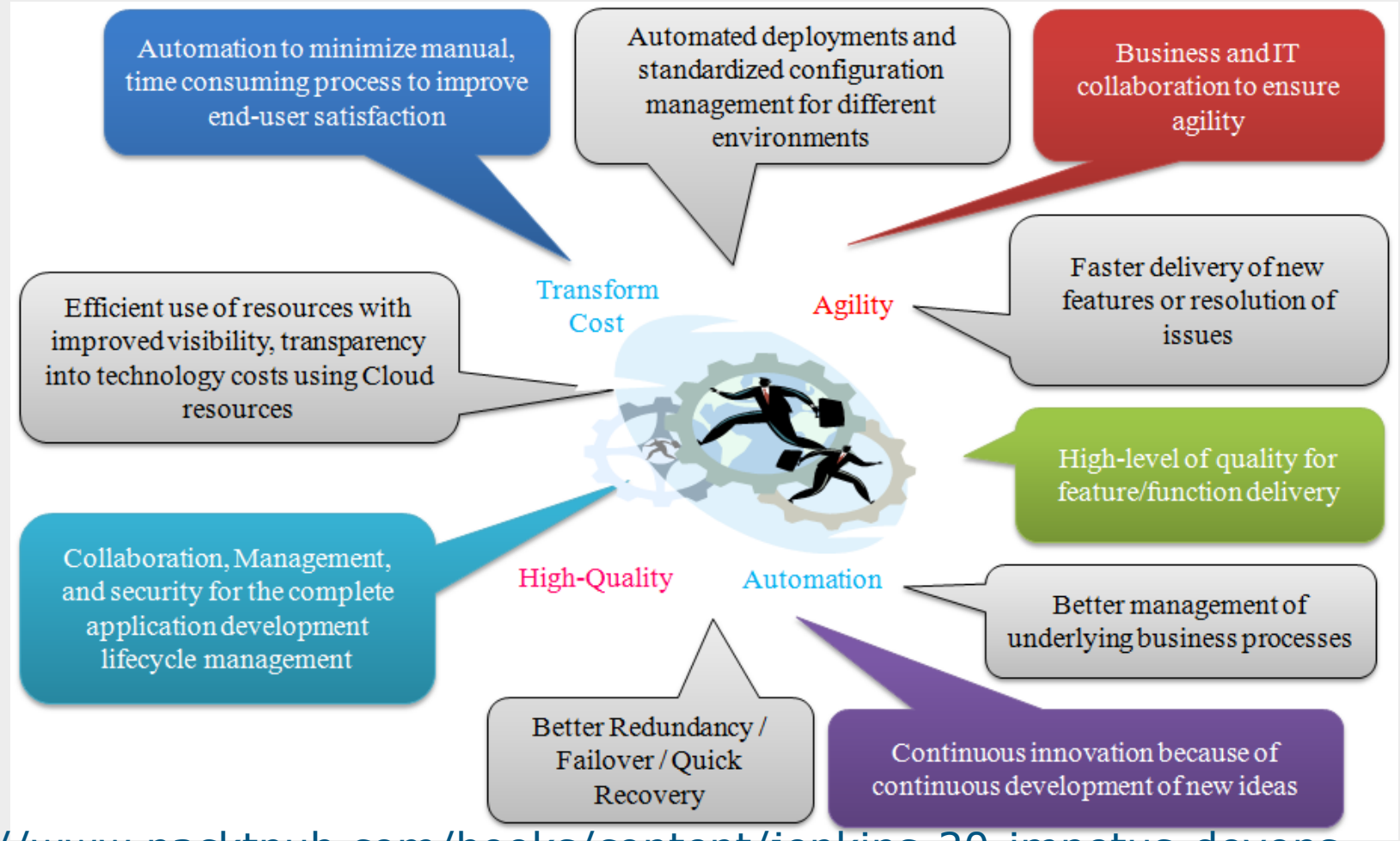
- Team and Task Coordination
  - Slack
  - HipChat
  - Jostle
  - Trello
  - Asana
- Infrastructure as a Service
  - Amazon Web Services
  - Rackspace
  - Cloud Foundry
  - Azure
  - OpenStack
- Virtualization Platforms
  - VMware
  - KVM
  - Xen
  - VirtualBox
  - Vagrant
- Containerization Tools
  - LXC
  - Solaris Containers
  - Docker
- Configuration Management
  - Puppet / MCollective
  - Chef
  - Ansible
  - CFEngine
  - SaltStack
  - RANCID
  - Ubuntu Juju
- Continuous Integration, Delivery & Release Management
  - CircleCI
  - TravisCI
  - LaunchDarkly
- Queues and Caches
  - ActiveMQ
  - RabbitMQ
  - memcache
  - varnish
  - Squid
- Logging
  - PaperTrail
  - Logstash
  - Loggly
  - Logentries
  - Splunk
  - SumoLogic
- Monitoring, Alerting, and Trending
  - New Relic
  - Nagios
  - Icinga
  - Graphite
  - Ganglia
  - Cacti
  - PagerDuty
  - Sensu

DevOps Tools

A Typical Deployment Landscape



Benefits of DevOps



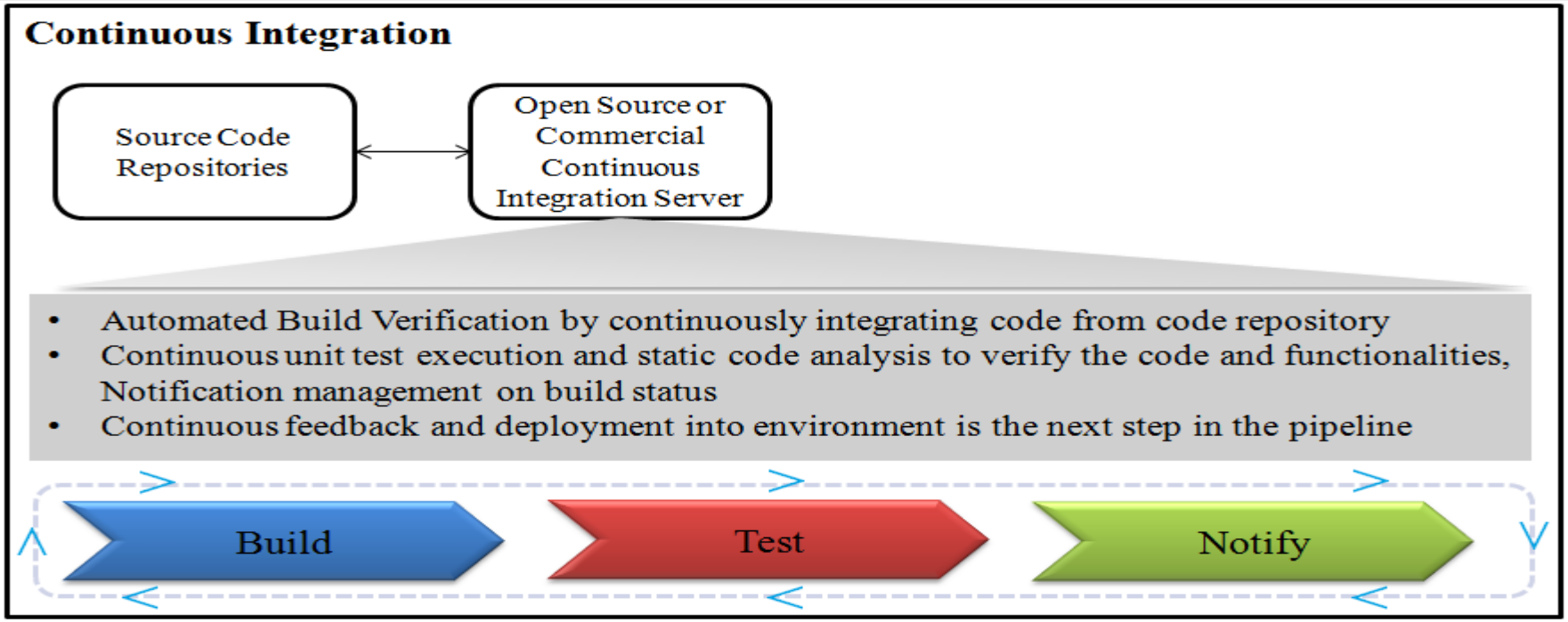
https://www.packtpub.com/books/content/jenkins-20-impetus-devops-movement

Continuous Integration and Delivery Pipeline

- **Continuous Integration** is a software engineering practice where each check-in made by a developer is verified by either of the following:
  - **Pull mechanism**: Executing an automated build at a scheduled time
  - **Push mechanism**: Executing an automated build when changes are saved in the repository
  - This step is followed by executing a unit test against the latest changes available in the source code repository.

Continuous Integration and Delivery Pipeline

- The main benefit of continuous integration is quick feedback based on the result of build execution. If it is successful, all is well; else, assign responsibility to the developer whose commit has broken the build, notify all stakeholders, and fix the issue.



Continuous Integration and Delivery Pipeline

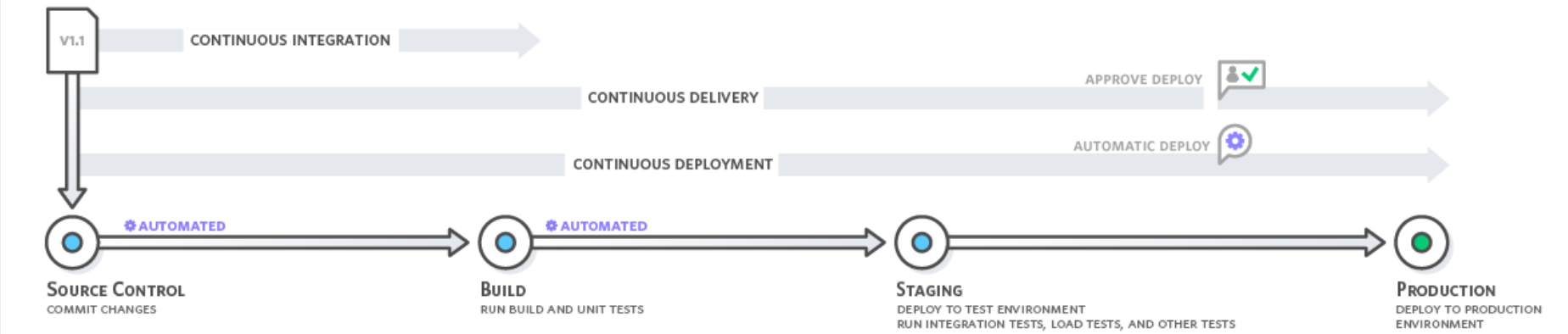
- **Continuous Delivery** is a DevOps software development practice where code changes are automatically built, tested, and prepared for a release to production
  - When continuous delivery is implemented properly, developers will always have a deployment-ready build artifact that has passed through a standardized test process
  - With continuous delivery, every code change is built, tested, and then pushed to a non-production testing or staging environment.
  - There can be multiple, parallel test stages before a production deployment
  - In the last step, the developer approves the update to production when they are ready
  - This is different from continuous deployment, where the push to production happens automatically without explicit approval

Continuous Integration and Delivery Pipeline

- Continuous delivery lets developers automate testing beyond just unit tests so they can verify application updates across multiple dimensions before deploying to customers.
- These tests may include UI testing, load testing, integration testing, API reliability testing, etc.
- This helps developers more thoroughly validate updates and pre-emptively discover issues.
- With the cloud, it is easy and cost-effective to automate the creation and replication of multiple environments for testing, which was previously difficult to do on-premises.

CI&CD

Continuous Integration and Delivery Pipeline



Continuous delivery automates the entire software release process. Every revision that is committed triggers an automated flow that builds, tests, and then stages the update. The final decision to deploy to a live production environment is triggered by the developer

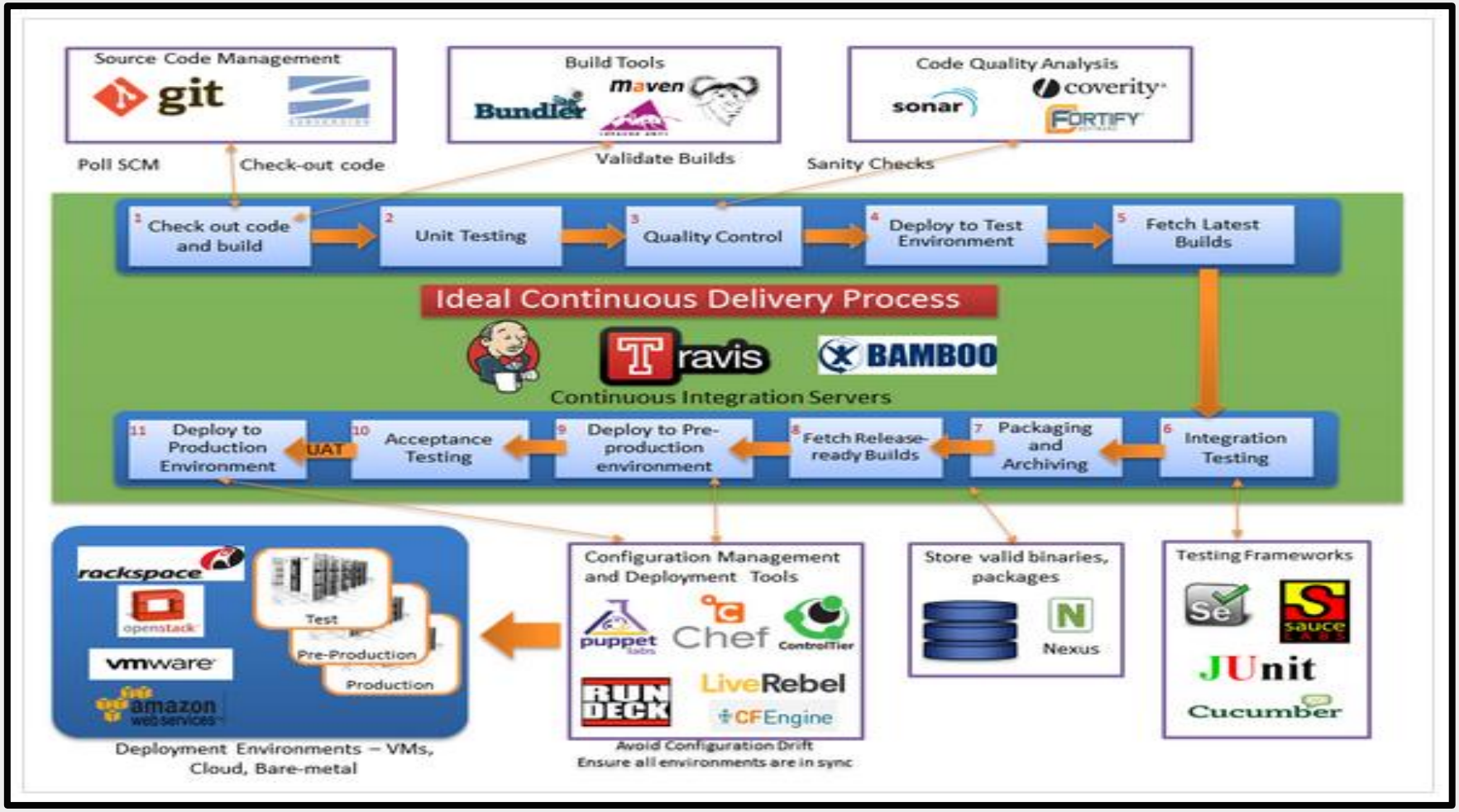
Capgemini

Continuous Integration and Delivery Pipeline

- The delivery pipeline can be broken down into a few major buckets of work, or stages, as mentioned below.
    1. Source code control (management)
    2. Build automation
    3. Unit test automation (could also include Integration Testing here as well)
    4. Deployment automation
    5. Monitoring
- The **Figure** shown is a sample of what the whole flow looks

from committing the code to the repo to deploying the code to an environment.

Continuous Integration and Delivery Pipeline



Continuous Integration and Delivery Pipeline

- Following are immediate benefits of **Continuous Integration**:
  - Automated integration with pull or push mechanism
  - Repeatable process without any manual intervention
  - Automated test case execution
  - Coding standard verification
  - Execution of scripts based on requirement
  - Quick feedback: build status notification to stakeholders via e-mail
  - Teams focused on their work and not in the managing processes
- Benefits **Continuous Delivery**
  - Automate the software release process
  - Improve developer productivity
  - Find and address bugs quickly
  - Deliver updates faster

Use-case walkthrough Pre-DevOps Scenario

The Dev team that has a goal to ship as many features as possible, kicks a new release “over the wall” to QA.

Then the tester’s goal is to find as many bugs as possible. When the testers bring their findings to Dev, the developers become defensive and blame the testers that are testing the environment for the bugs. The testers respond that it isn’t their testing environment, but the developer’s code that is the problem.

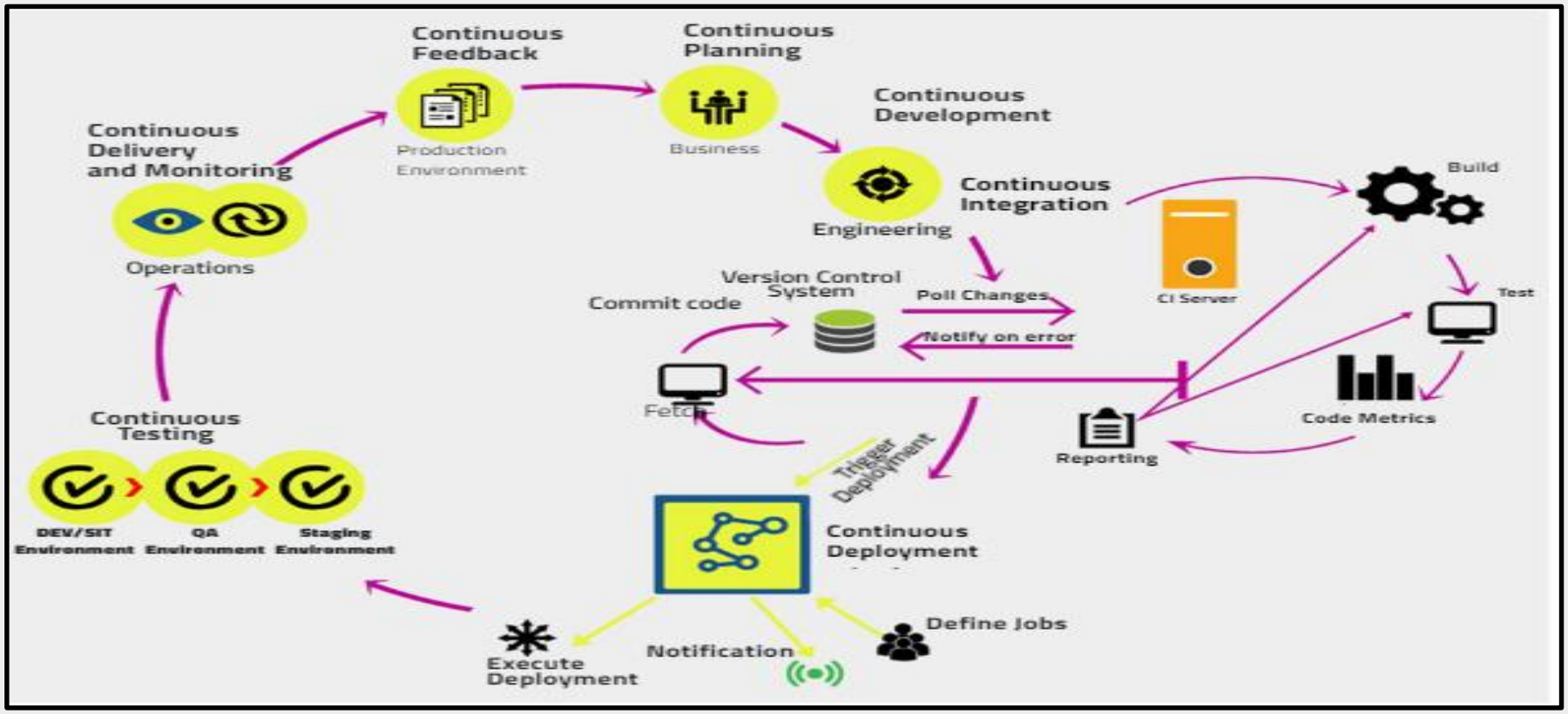
Eventually the issues get worked out and QA kicks the debugged new release “over the wall” to Ops.

The Ops team’s goal is to limit changes to their system, but they apprehensively release the code and the system crashes. The finger pointing resumes.

Ops says that Dev provided them faulty artifacts. Dev says everything worked fine in the test environment. The fire drills begin to debug the system and get production stable. The production environment isn’t Dev’s and QA’s responsibility, so they keep hands off while Ops spends all night fixing the production issues.

DevOps End-to-End Implementation

- Analyze, design, construct, automate and implement according to the needs identified for each project



Use-case walkthrough Post-DevOps Scenario

- The table below lists typical transformations that occur post successful DevOps implementation:

Category	Before Implementation	After Implementation
Teams	Development and operations teams with different goals and processes	Development and operations working as a single global team with common set of goals and processes
	Each team spends considerable, manual effort to execute,develop,test and release management activities	Increased collaboration across all teams using automated processes and consistent goals. This leads to increased team productivity and lowered operations cost
Infrastructure	Nonstandard and fragmented	Organized and standard technology stack
	Manual infrastructure setup and provisioning	Automated, infrastructure provisioning with metrics-based monitoring tools
Delivery model	Waterfall model	Agile and iterative delivery
	Longer release cycle	Incremental releases

Use-case walkthrough

Category	Before Implementation	After Implementation
Development and testing processes	Traditional	Iterative/agile development and testing with incremental releases
	Manual	Automated testing and continuous validation
	Costly and error prone	Reduced cost and risk due to continuous integration and testing
Effectiveness of end user feedback and change requests	Inability to overcome challenges in handling change requests	Higher effectiveness of change requests,feedback,and enhancement due to agile delivery

Summary

- DevOps enables continuous software delivery with less complex problems to fix and faster resolution of problems
- DevOps enables
  - *Continuous Development*
  - *Continuous Testing,*
  - *Continuous Integration*
  - *Continuous Deployment*
  - *Continuous Monitoring*
- DevOps integrates developers and operations team in order to improve collaboration and productivity

Review Question

- DevOps integrates Developers and Operations team for better collaboration and productivity
  - True
  - False
- Select the tools used for maintaining the different versions of the code
  - GIT
  - SVN
  - Maven
  - All the above
- _____ tool is used for Continuous Integration
  - GIT
  - Puppet
  - Jenkins