Capgemini

DevOps

DevOps Fundamentals

Table of Contents

Part I- Introduction to DevOps

- What is DevOps
- Evolution of Software Development
- Why DevOps
- Agile vs DevOps
- DevOps Principles
- DevOps Lifecycle

Part II- DevOps in Practice

- DevOps Tools
- Benefits of DevOps
- Continuous Integration
- Delivery pipeline

Part III- Demo

- Continuous Delivery
- DevOps in Cloud

Capgemini

Part I

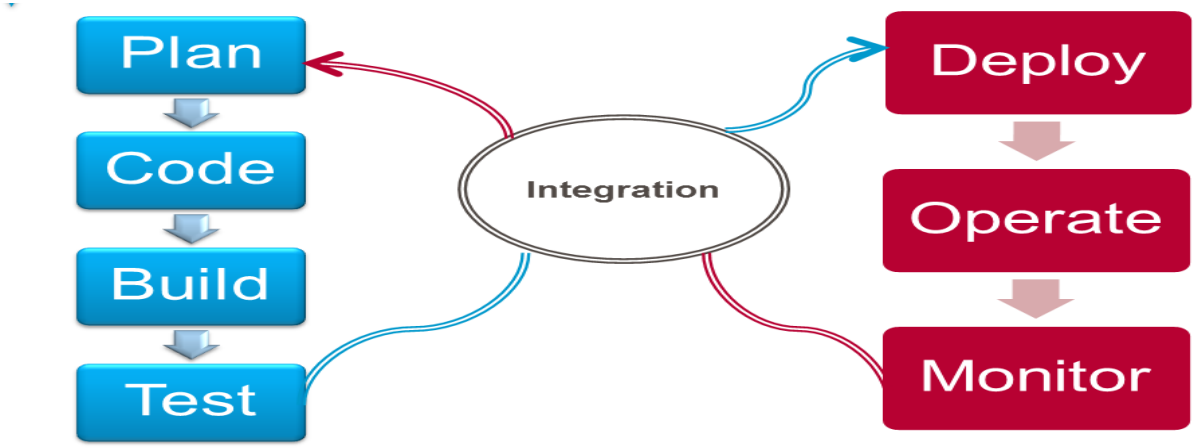
Introduction to DevOps

What is DevOps

DevOps is a Software Development approach which involves

- *Continuous Development*
- *Continuous Testing,*
- *Continuous Integration*
- *Continuous Deployment*
- *Continuous Monitoring*

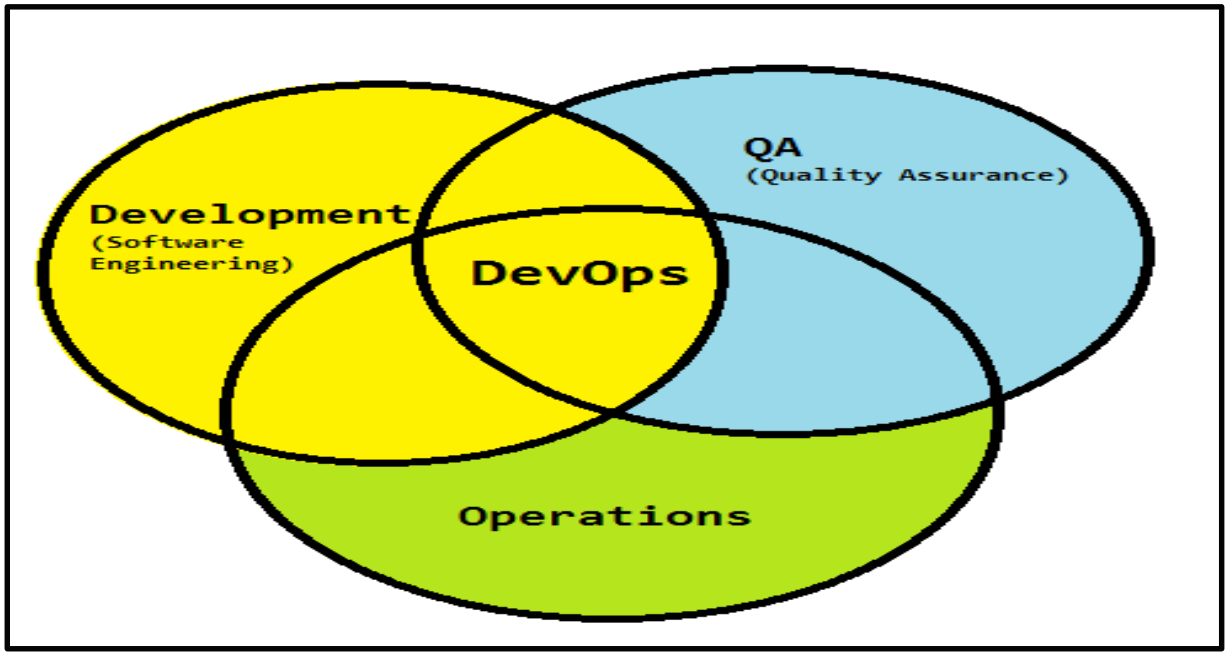
of the software throughout its development lifecycle



What is DevOps

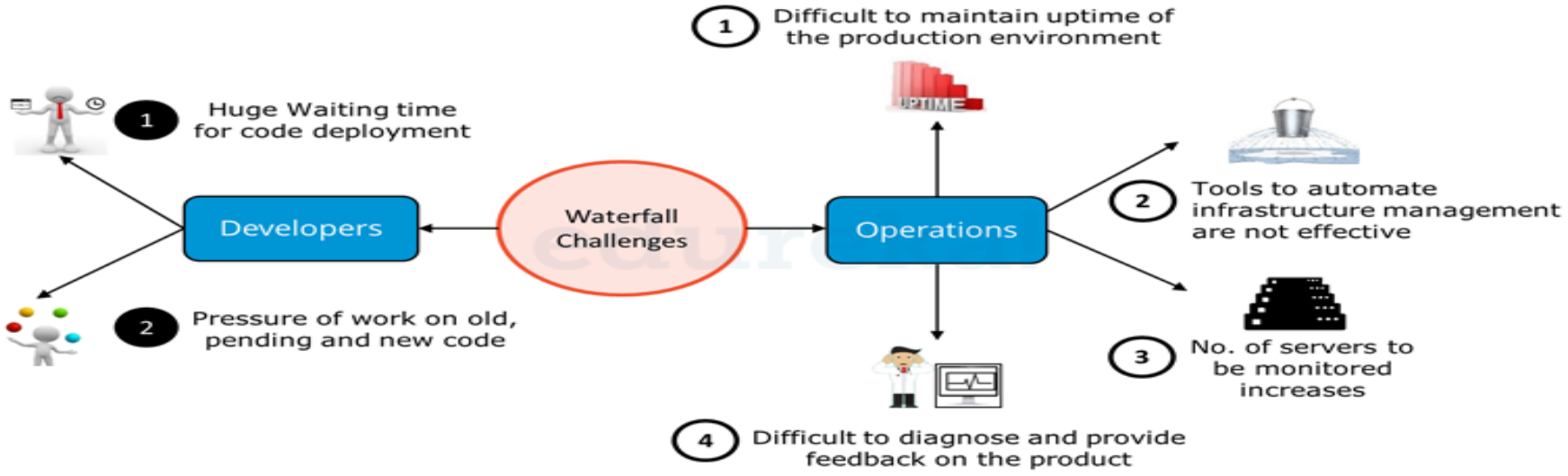
DevOps integrates developers and operations team in order to improve collaboration and productivity by:

- Automating infrastructure
- Automating workflows
- Continuous measuring application performance



Evolution of Software Development

Waterfall Model Challenges: The Water-fall model worked fine and served well for many years however it had some challenges. The challenges of Waterfall Model are highlighted in the following diagram



It is evident from the above diagram that both development and operations had challenges in the Waterfall Model

Evolution of Software Development

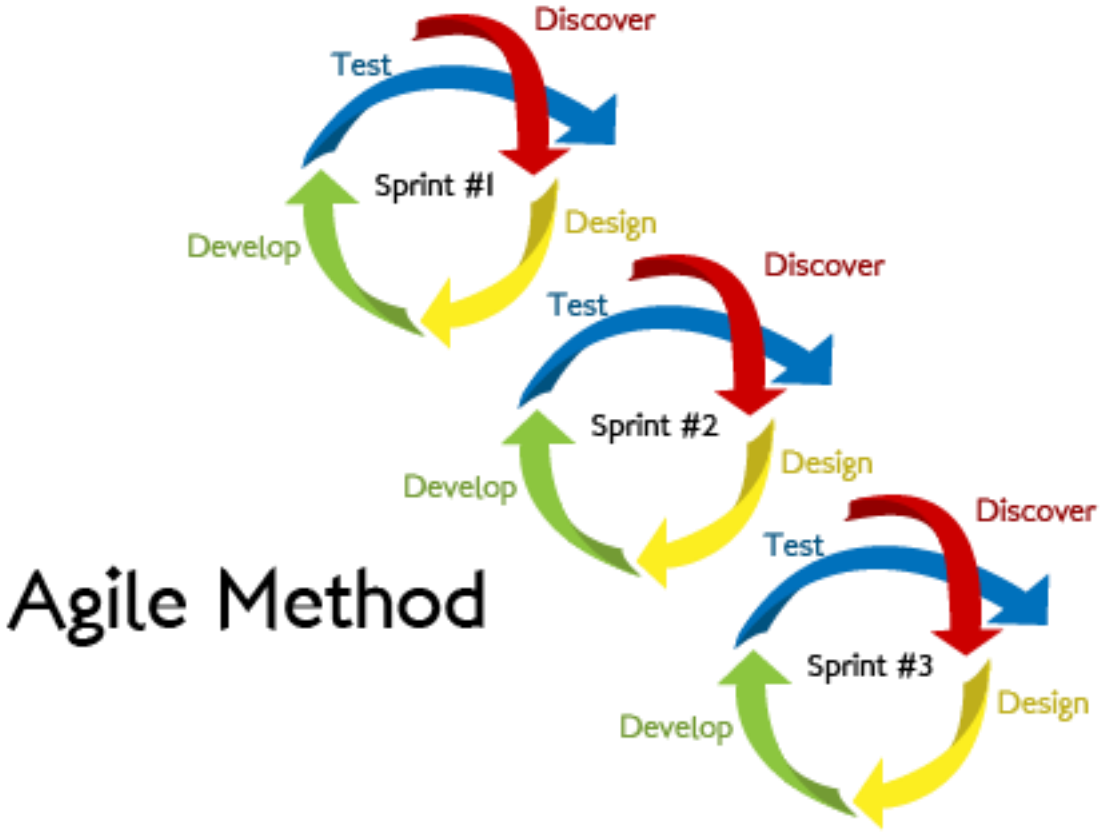
Agile Methodology: In the Agile approach, software is developed and released incrementally in the iterations

This results in small incremental releases with each release building on previous functionality

Each release is thoroughly **tested** to ensure **software quality** is maintained

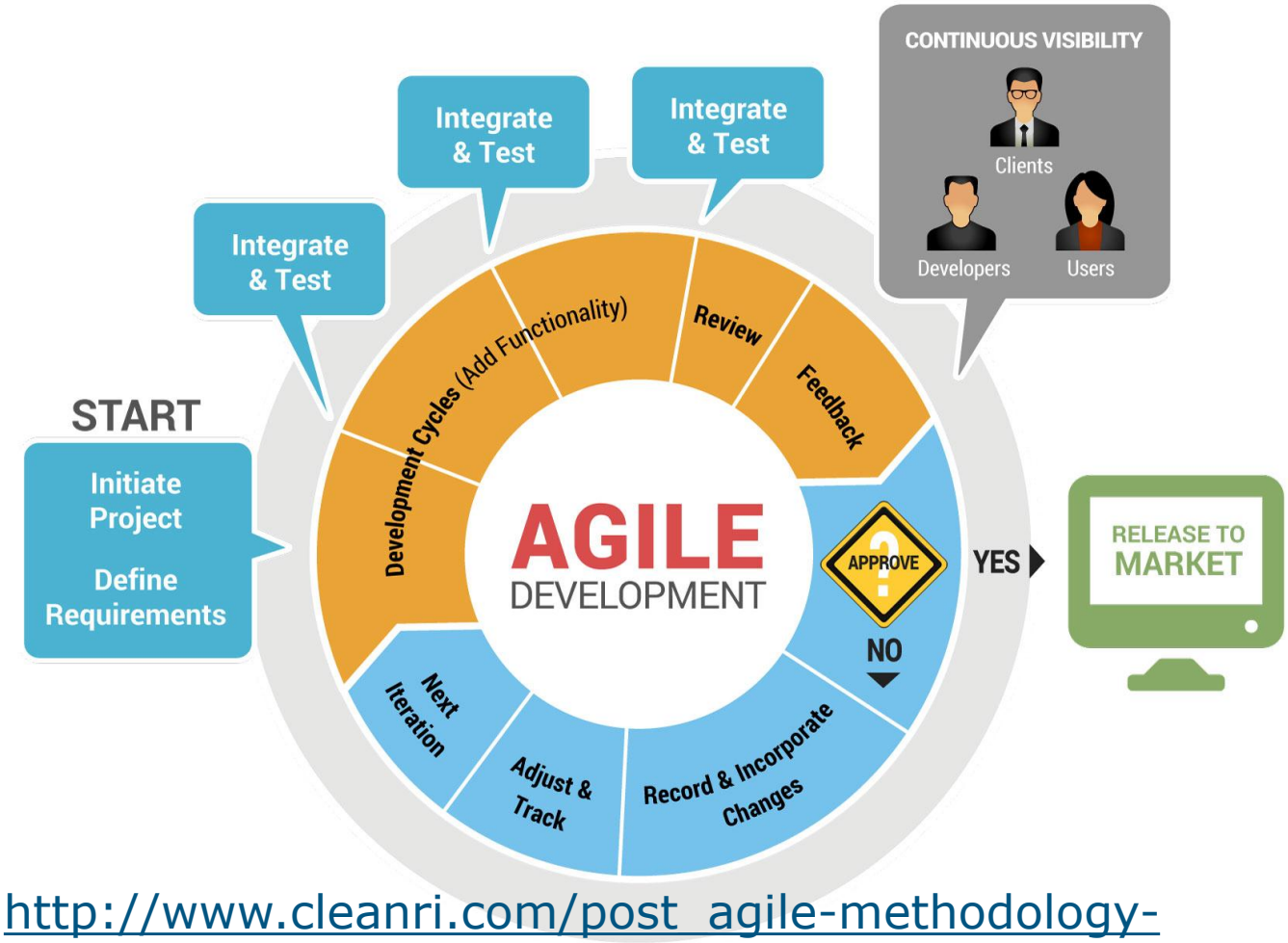
It is used for time critical applications

Agile Methodology



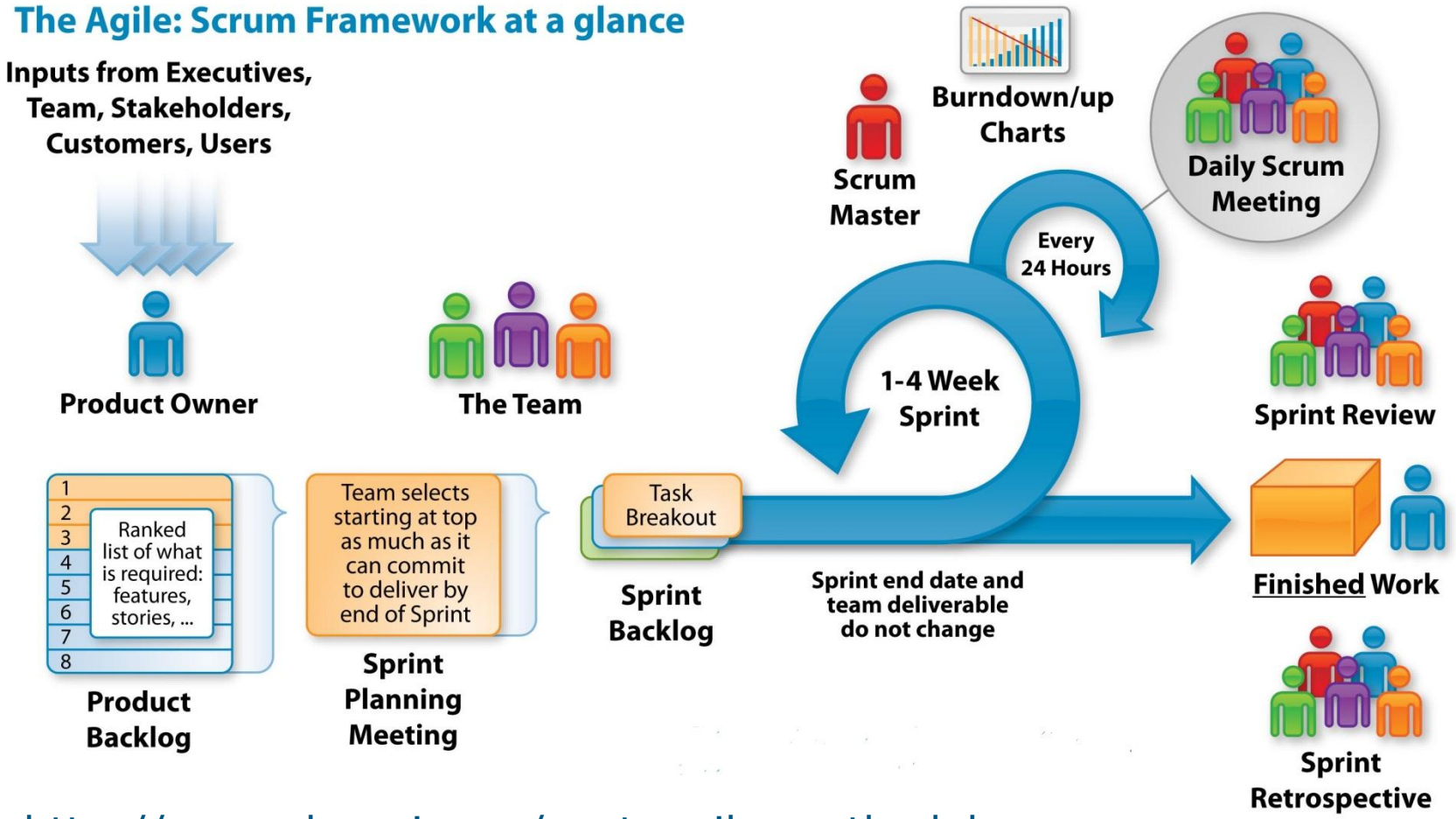
http://www.cleanri.com/post_agile-methodology-diagram_43651/

Agile Methodology



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

Agile- Scrum Framework



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

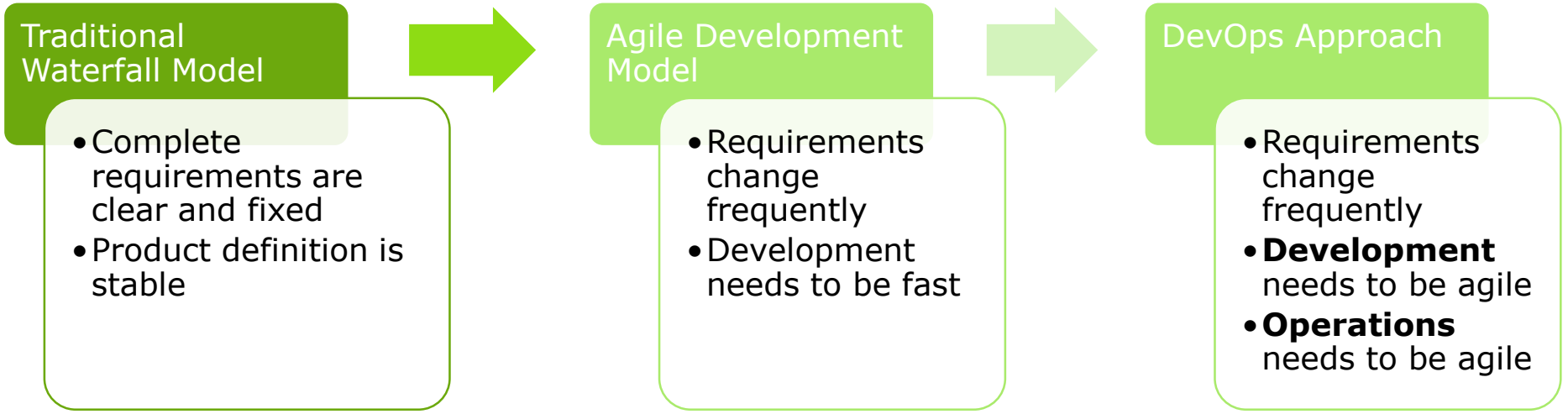
Agile Practices

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

Agile Practices

Evolution of Software Development

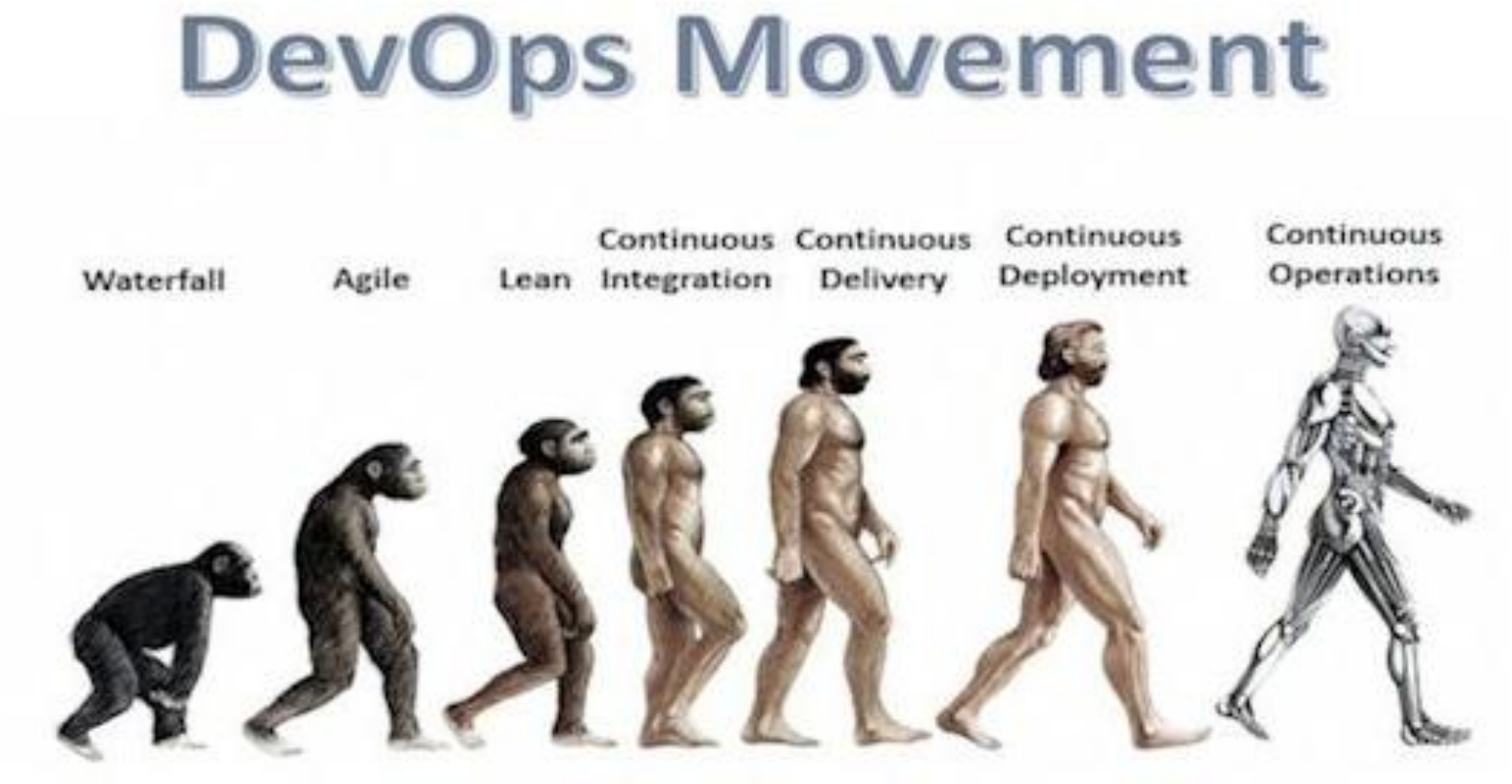
Evolution of Software Development



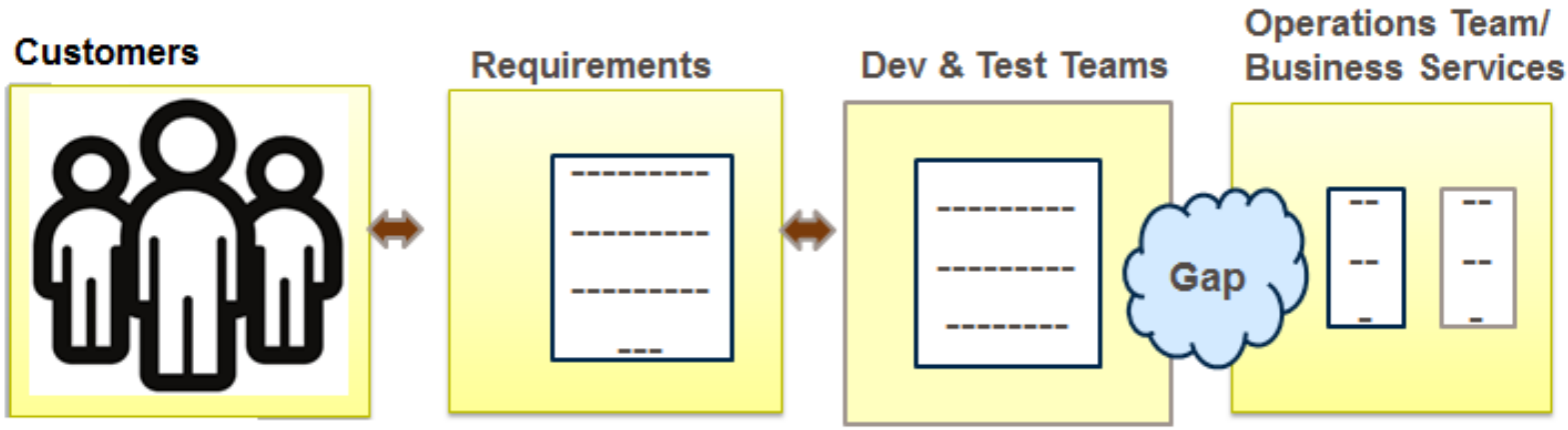
Evolution of Software Development

- 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

DevOps Movement



Why DevOps


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

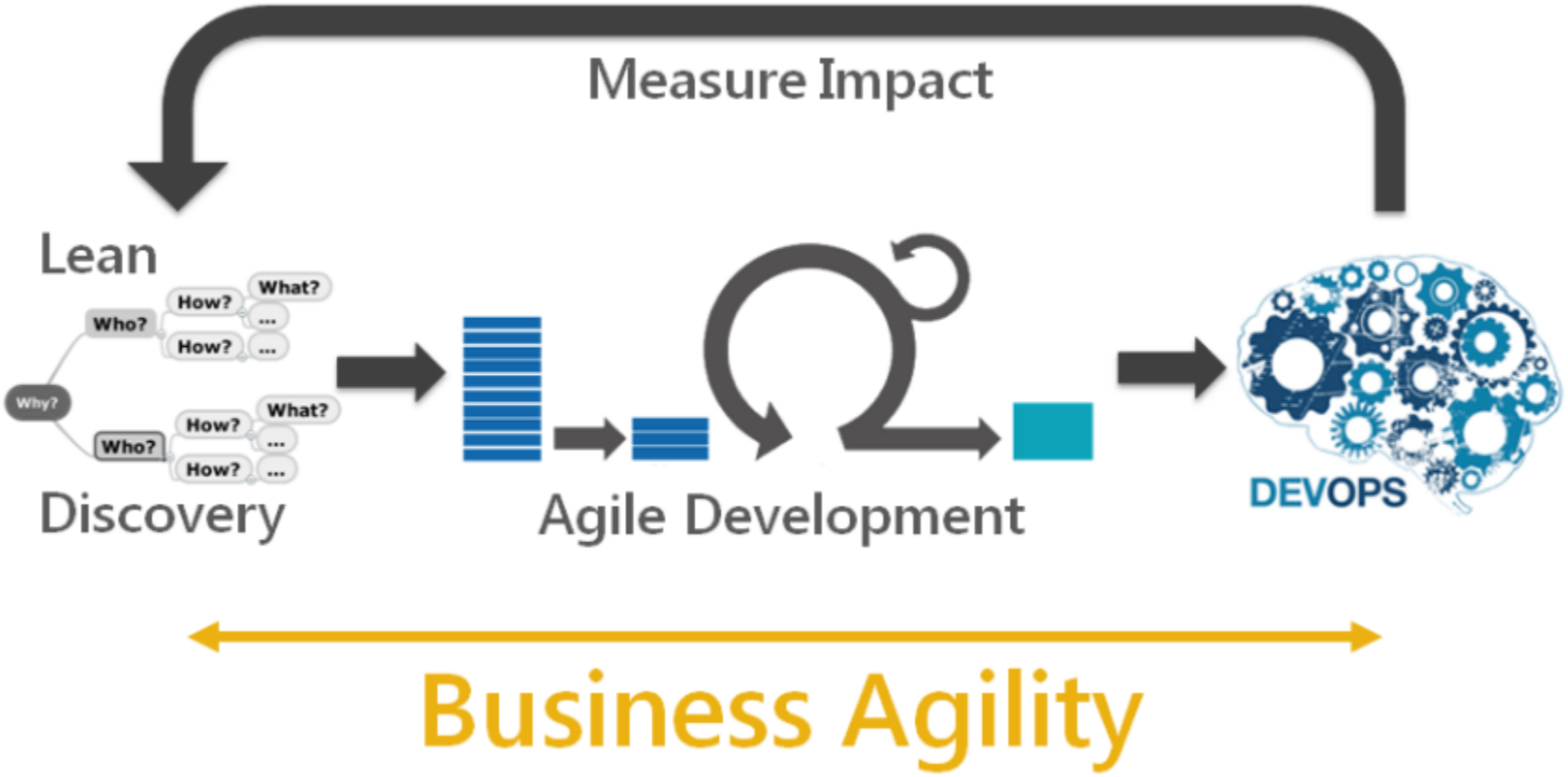
Why DevOps

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



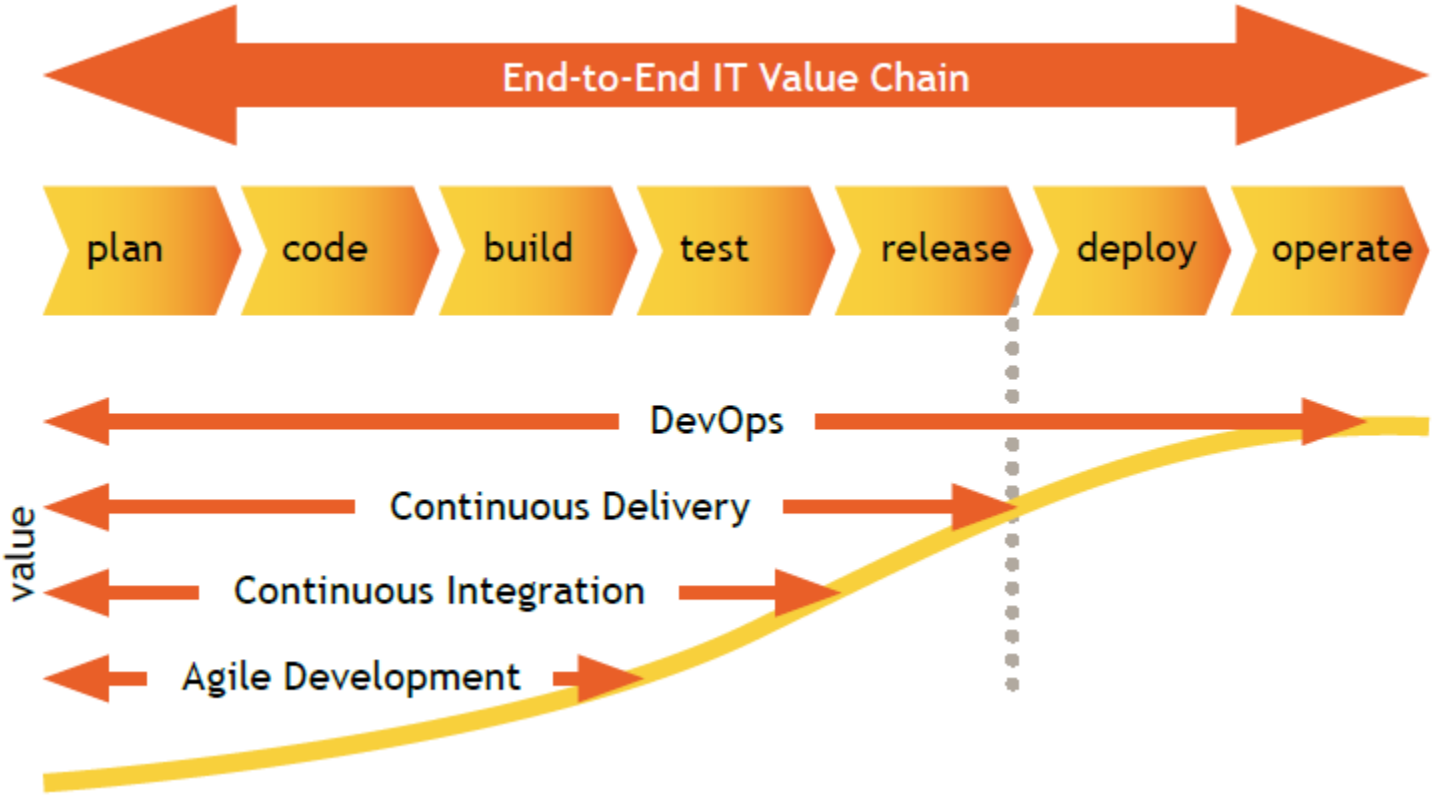
https://theagileadmin.com/what-is-devops/

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

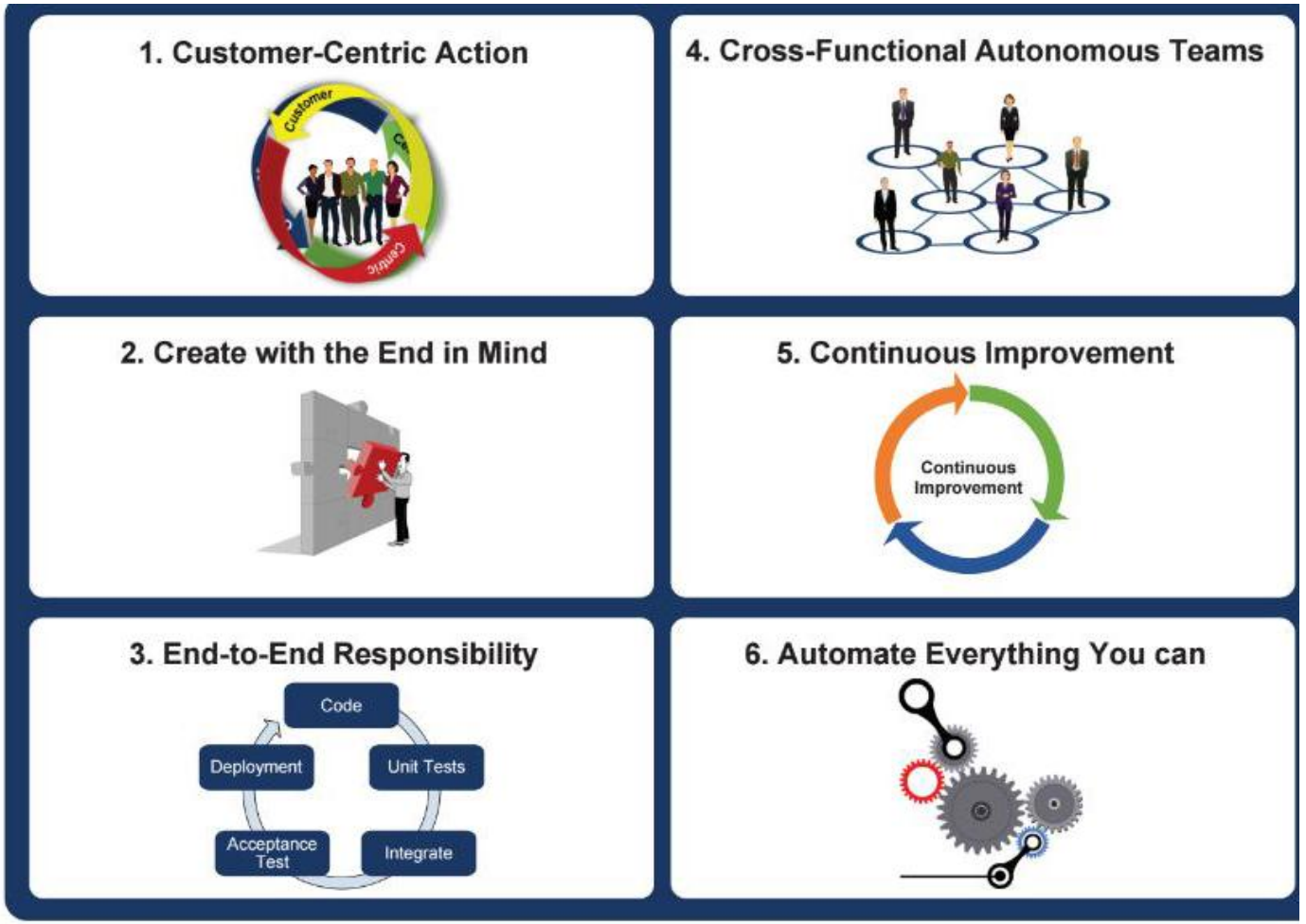
http://labs.sogeti.com/wp-content/uploads/2016/03/D2D-4-EN-web.pdf

Agile Vs DevOps



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

DevOps Principles



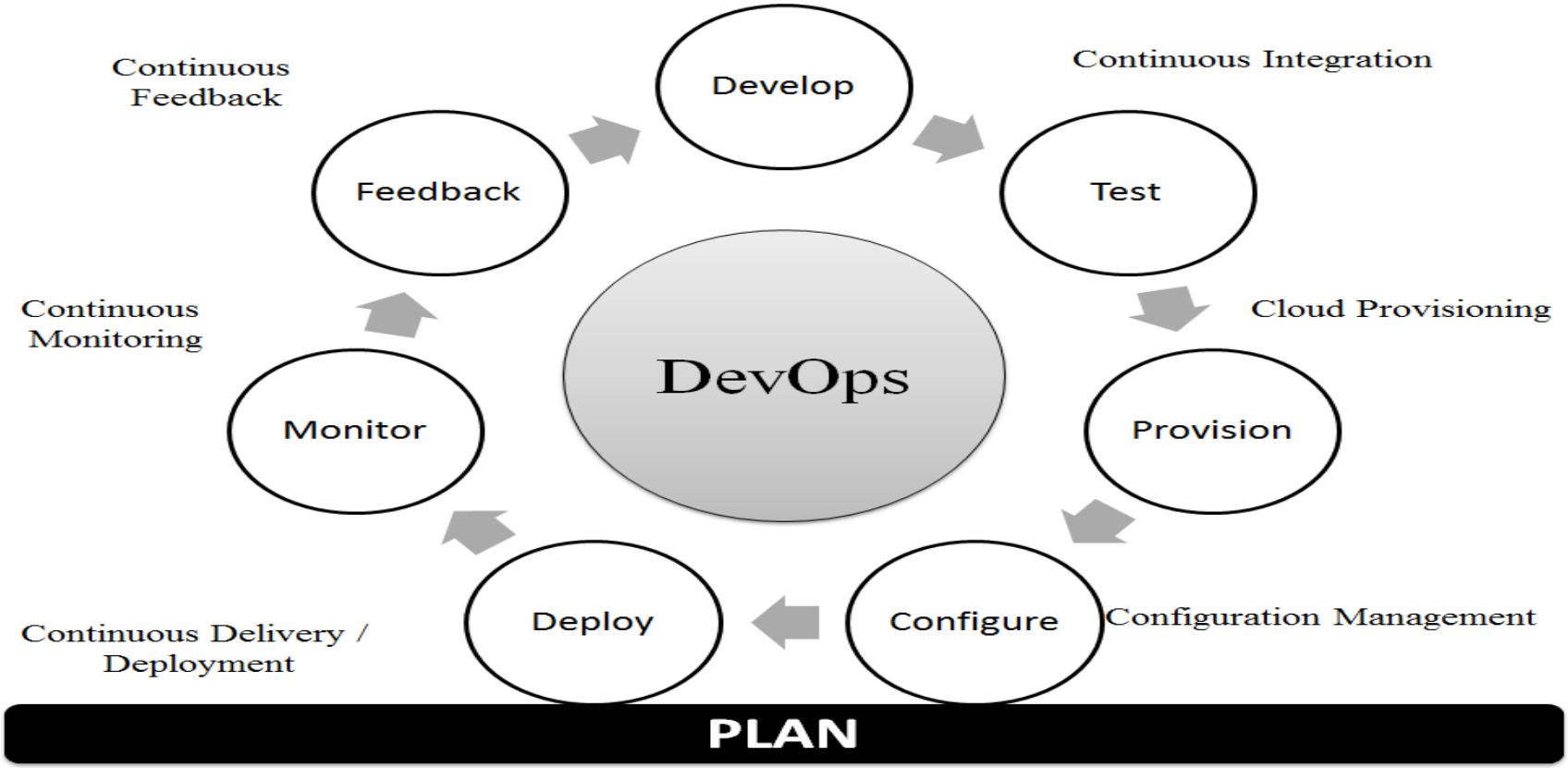
DevOps Life Cycle- it’s all about “continuous”

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.



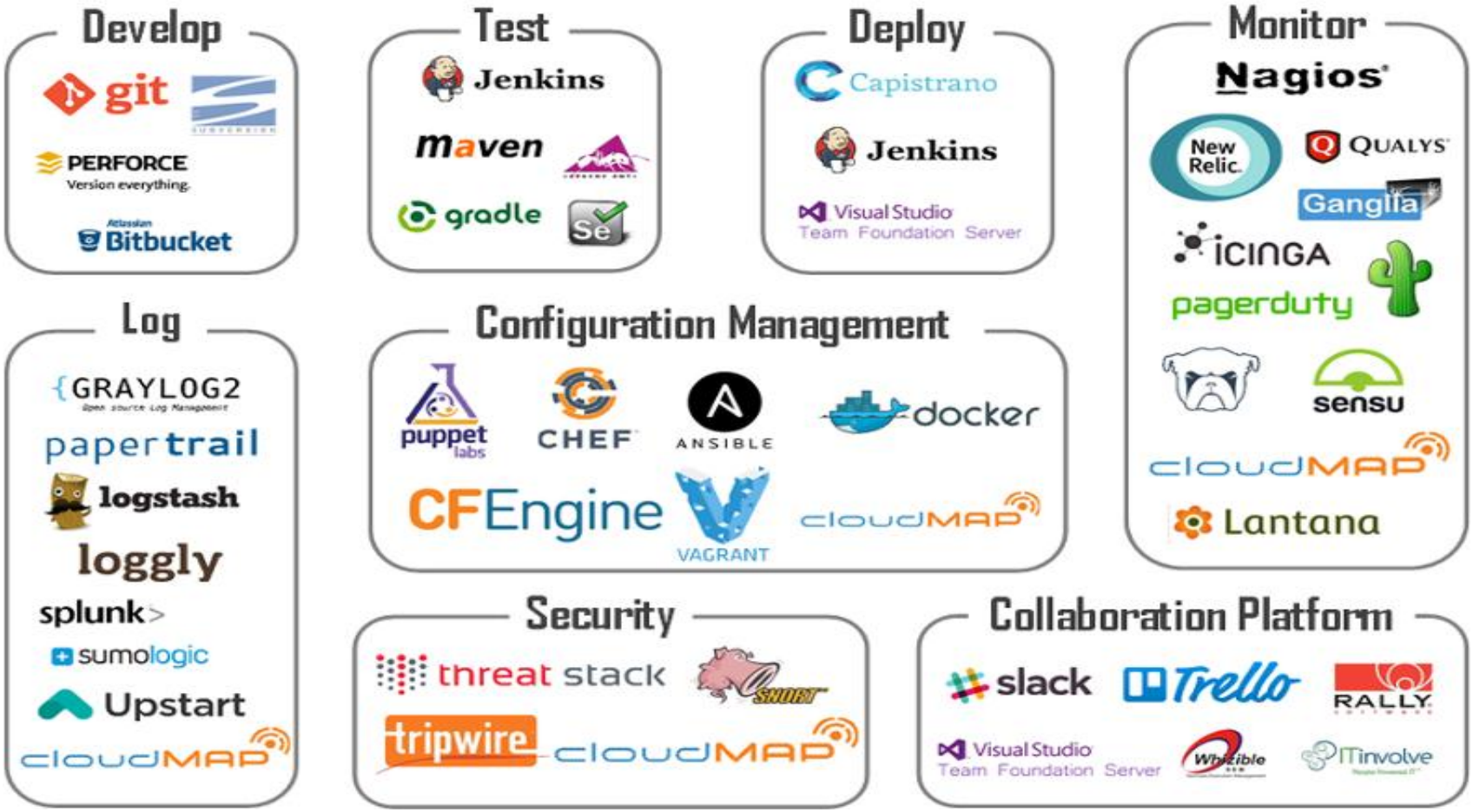
Capgemini

Part II

DevOps in Practice

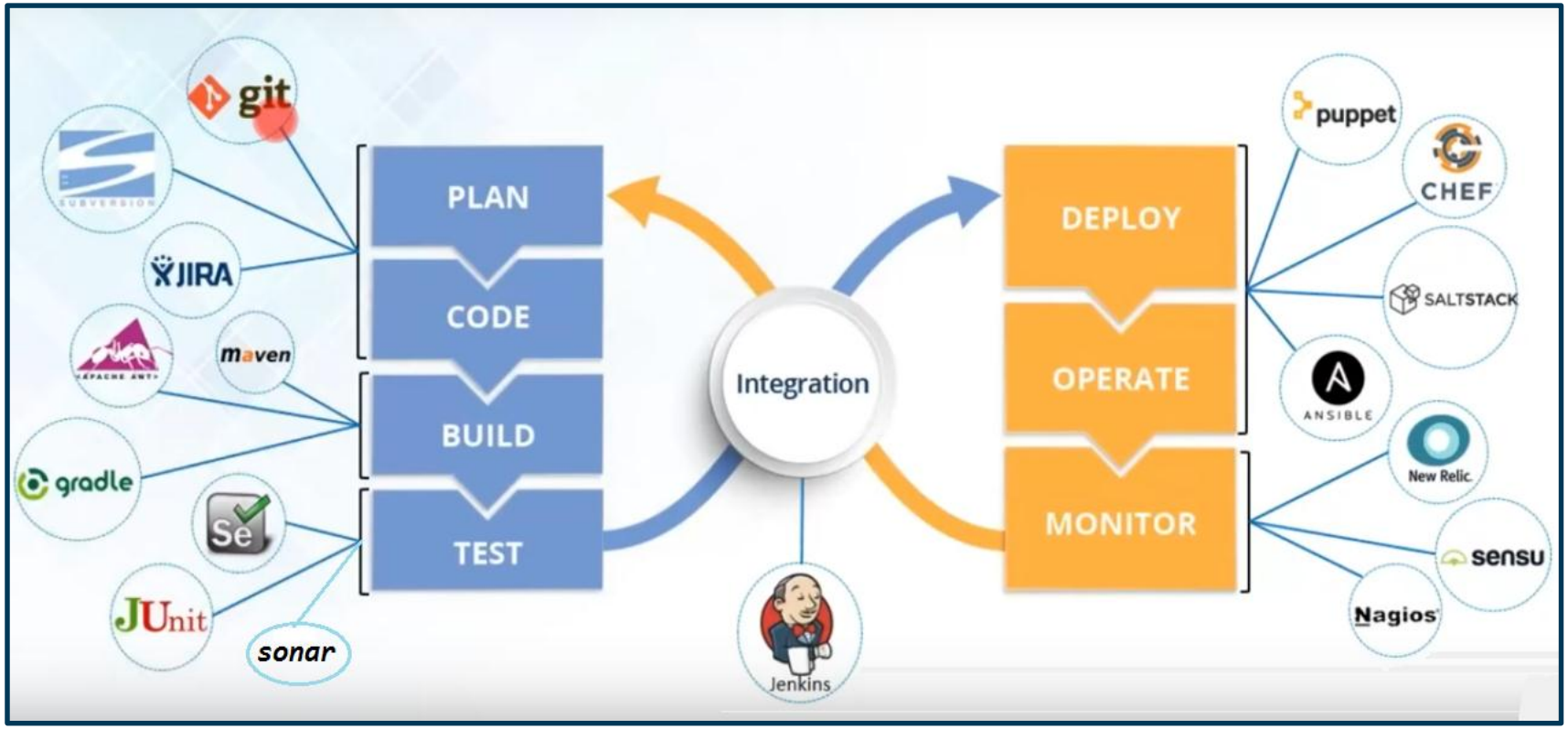
How does DevOps Work?

No tool will magically make the team DevOps.



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 DevOps Life Cycle

DevOps life cycle stages are carried out on loop continuously until the desired product quality is achieved.

Continuous Development:

- This is the stage in the DevOps life cycle where the Software is developed continuously.
- This stage involves the Coding and Building phases

Tools Used

- Git and SVN are used for maintaining the different versions of the code
- Ant, Maven, Gradle for building / packaging the code into an executable file that can be forwarded to the QAs for testing.

Tools used in various stages of DevOps Life Cycle

Continuous Testing:

- It is the stage where the developed software is continuously tested for bugs
- Once the code is tested, it is continuously integrated with the existing code

Tools Used

- For Continuous testing ,testing automation tools like Selenium, JUnit are used
- These tools enables the QA’s for testing multiple code-bases thoroughly in parallel to ensure that there are no flaws in the functionality

Tools used in various stages of DevOps Life Cycle

Continuous Integration:

- This is the stage where the code supporting new functionality is integrated with the existing code
- Since there is continuous development of software, the updated code needs to be integrated continuously as well as smoothly with the systems to reflect changes to the end users

Tools Used

- Jenkins is a very popular tool used for Continuous Integration
- Using Jenkins one can pull the latest code revision from GIT repository and produce a build which can finally be deployed to test or production server
- It can be set to trigger a new build automatically as soon as there is change in the GIT repository or can be triggered manually on click of a button.

Tools used in various stages of DevOps Life Cycle

Continuous Deployment:

- It is the stage where the code is deployed to the production environment
- Here it is ensured that the code is correctly deployed on all the servers
- Since the new code is deployed on a continuous basis, automation tools play an important role for executing tasks quickly and frequently

Tools Used

- Puppet, Chef, SaltStack and Ansible are some popular tools that are used in this stage

Tools used in various stages of DevOps Life Cycle

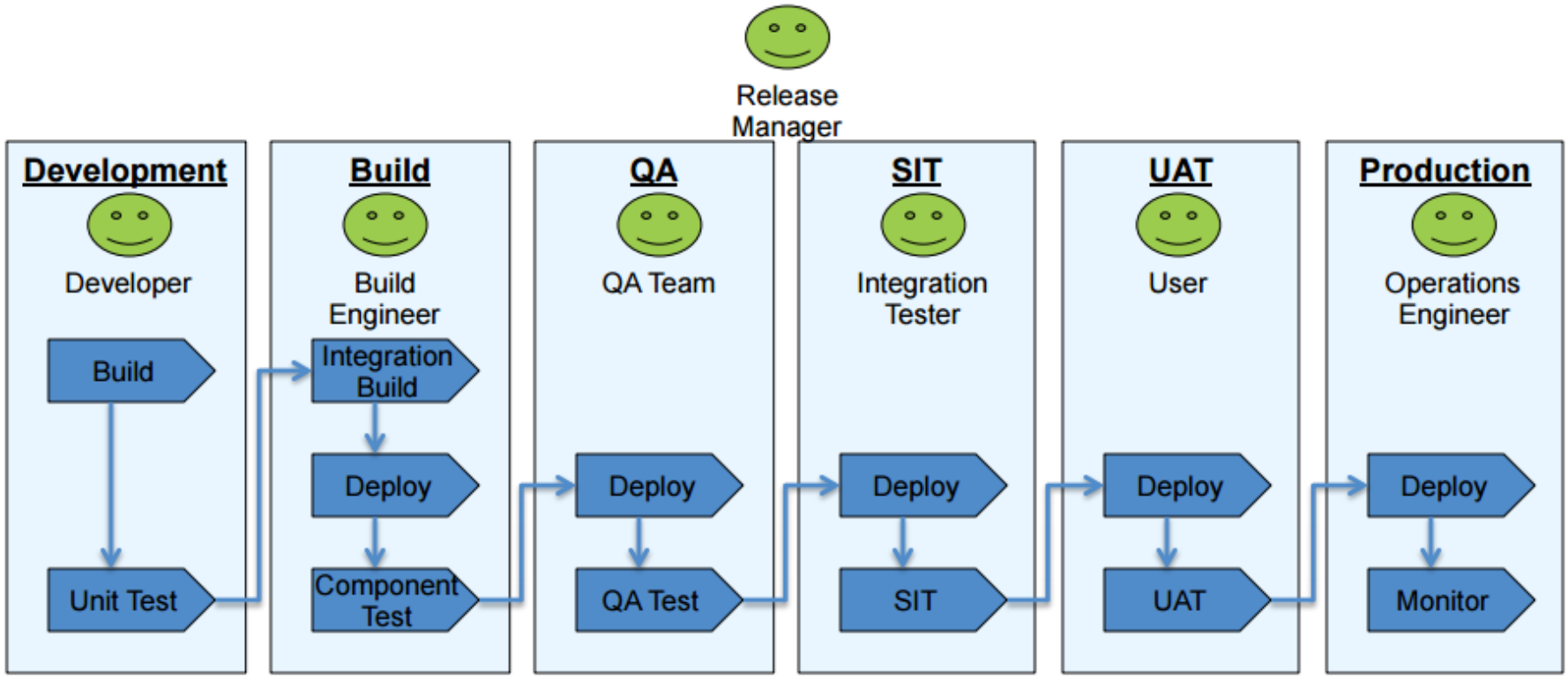
Continuous Monitoring:

- This stage in the DevOps life cycle is aimed at improving the quality of the software by monitoring its performance
- This practice involves the participation of the Operations team who will monitor the user activity for bugs / any improper behavior of the system
- This can also be achieved by making use of dedicated monitoring tools which will continuously monitor the application performance and highlight issues

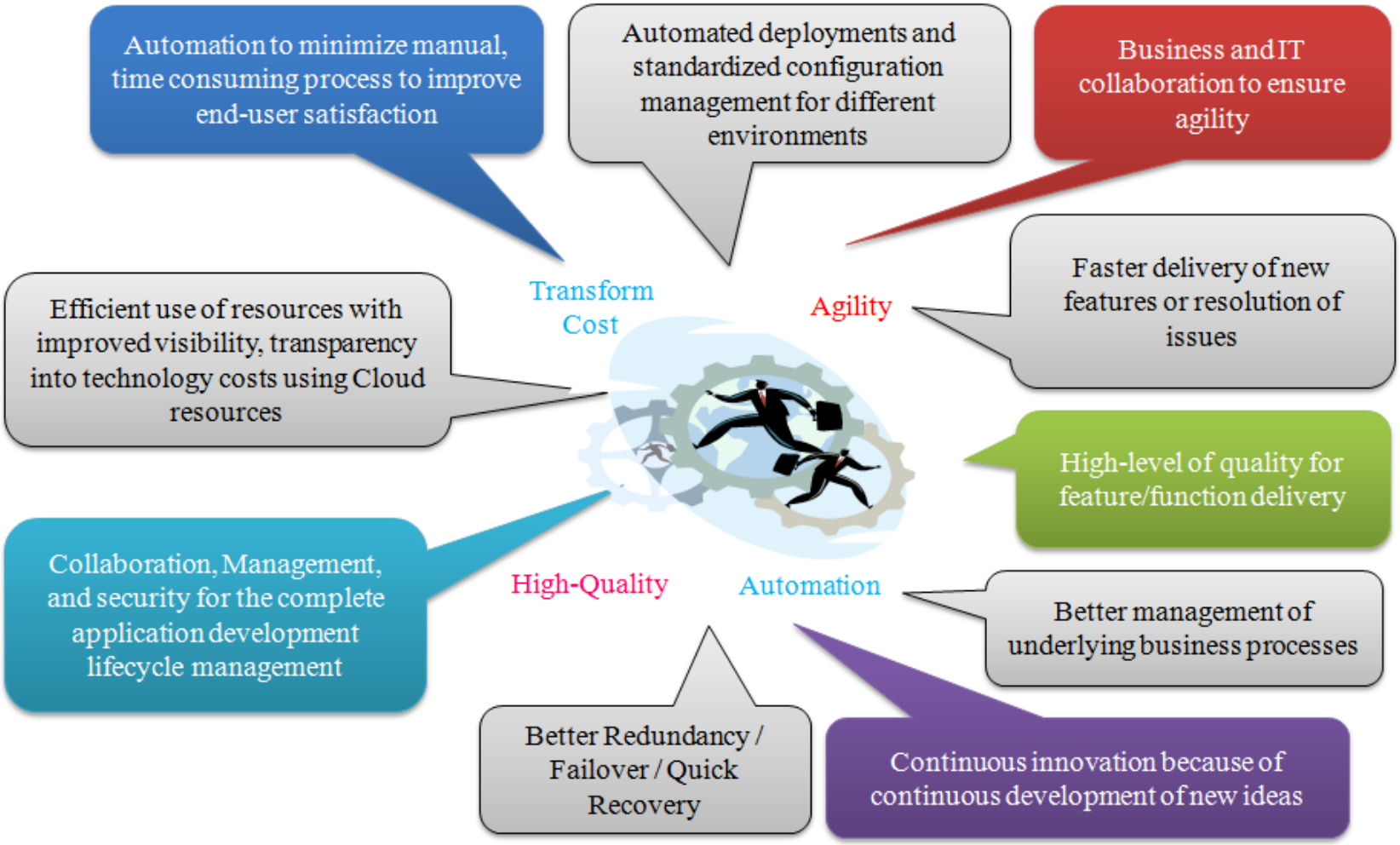
Tools Used

- Some popular tools used are Nagios, NewRelic and Sensu
- These tools help us to monitor the application and the servers closely to check the health of the system proactively
- They can also improve productivity and increase the reliability of the systems, reducing IT support costs
- Any major issues found could be reported to the Development team so that it can be fixed in the continuous development phase.

A Typical Deployment Landscape



Benefits of DevOps



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

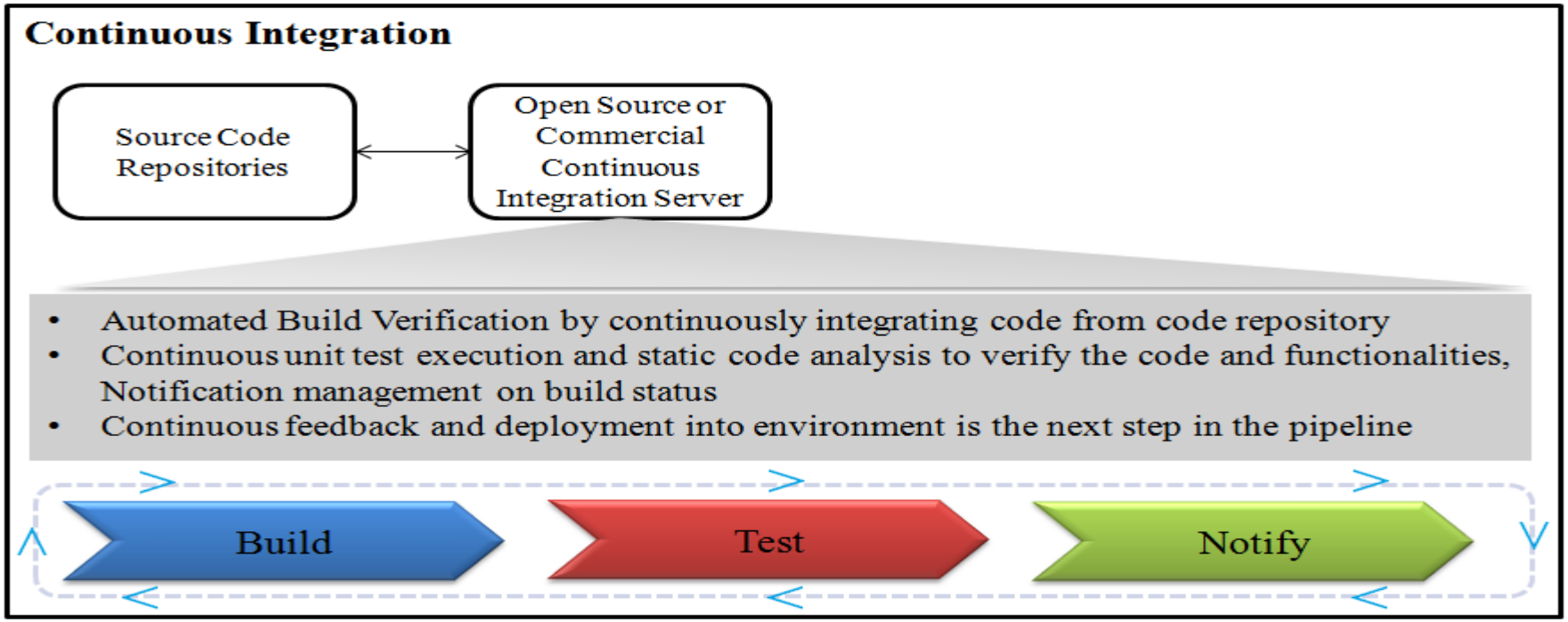
Continuous Integration and Delivery Pipeline

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

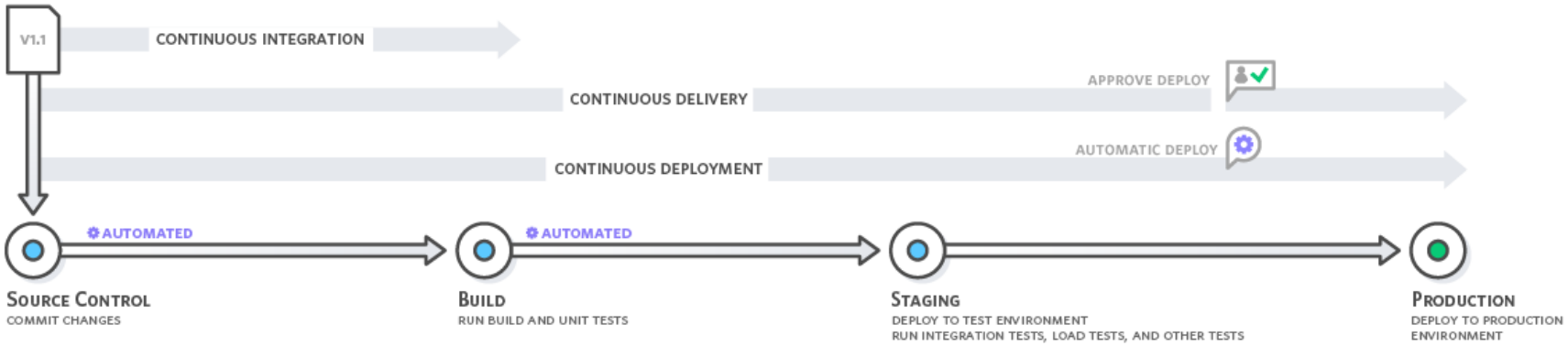
Continuous Delivery is a DevOps software development practice where code changes are automatically built, tested, and prepared for a release to production

- 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.

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

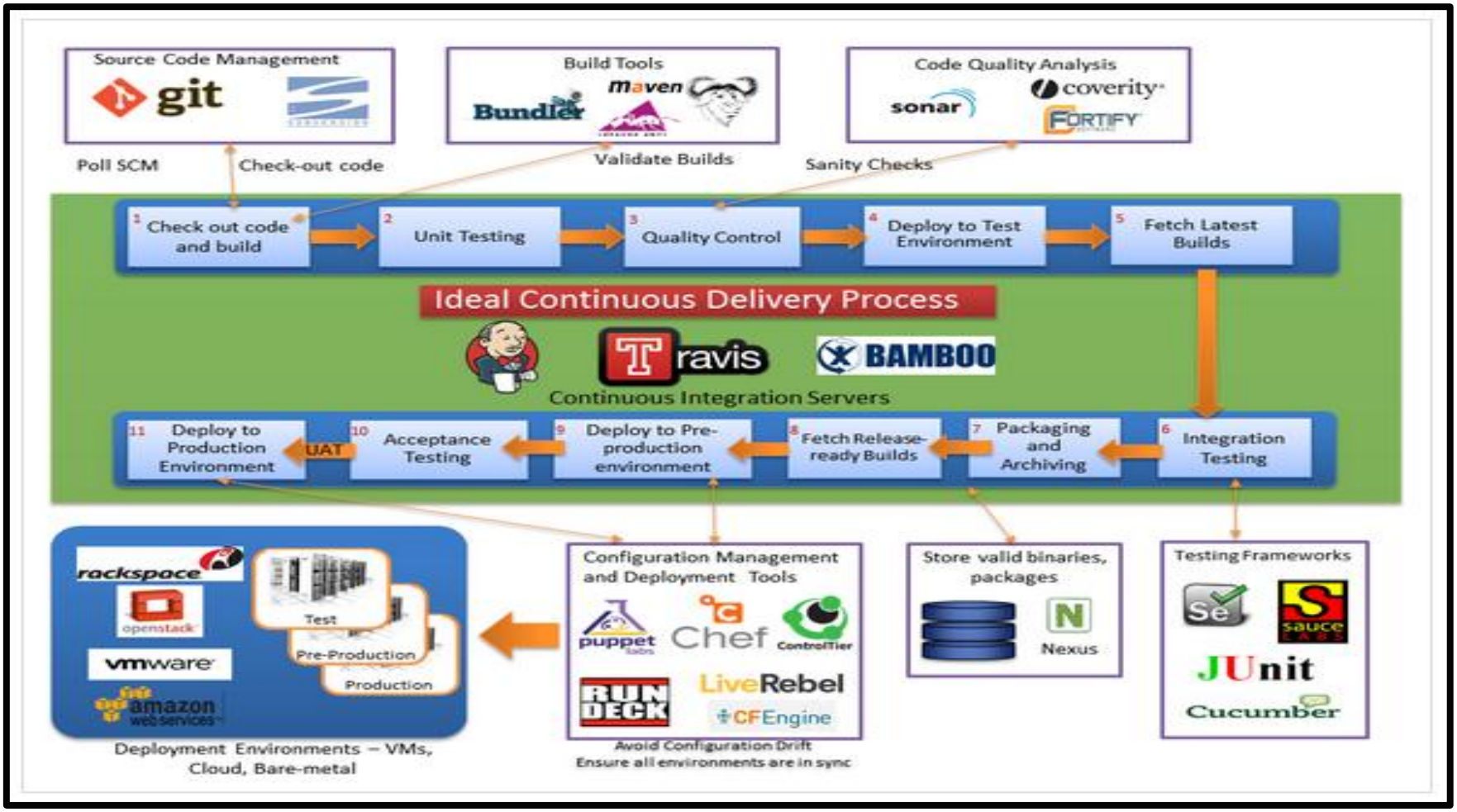
Continuous Integration and Delivery Pipeline

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

Capgemini

Part III

Demo

Summary

DevOps enables continuous software delivery with less complex problems to fix and faster resolution of problems

DevOps is a Software Development approach which involves

- *Continuous Development*
- *Continuous Testing,*
- *Continuous Integration*
- *Continuous Deployment*
- *Continuous Monitoring*

DevOps integrates developers and operations team in order to improve collaboration and productivity