


# DevOps



**Caltech**

**Center for Technology &  
Management Education**

## **Post Graduate Program in DevOps**



# Introduction to Configuration Management

# Learning Objectives

By the end of this lesson, you'll be able to:

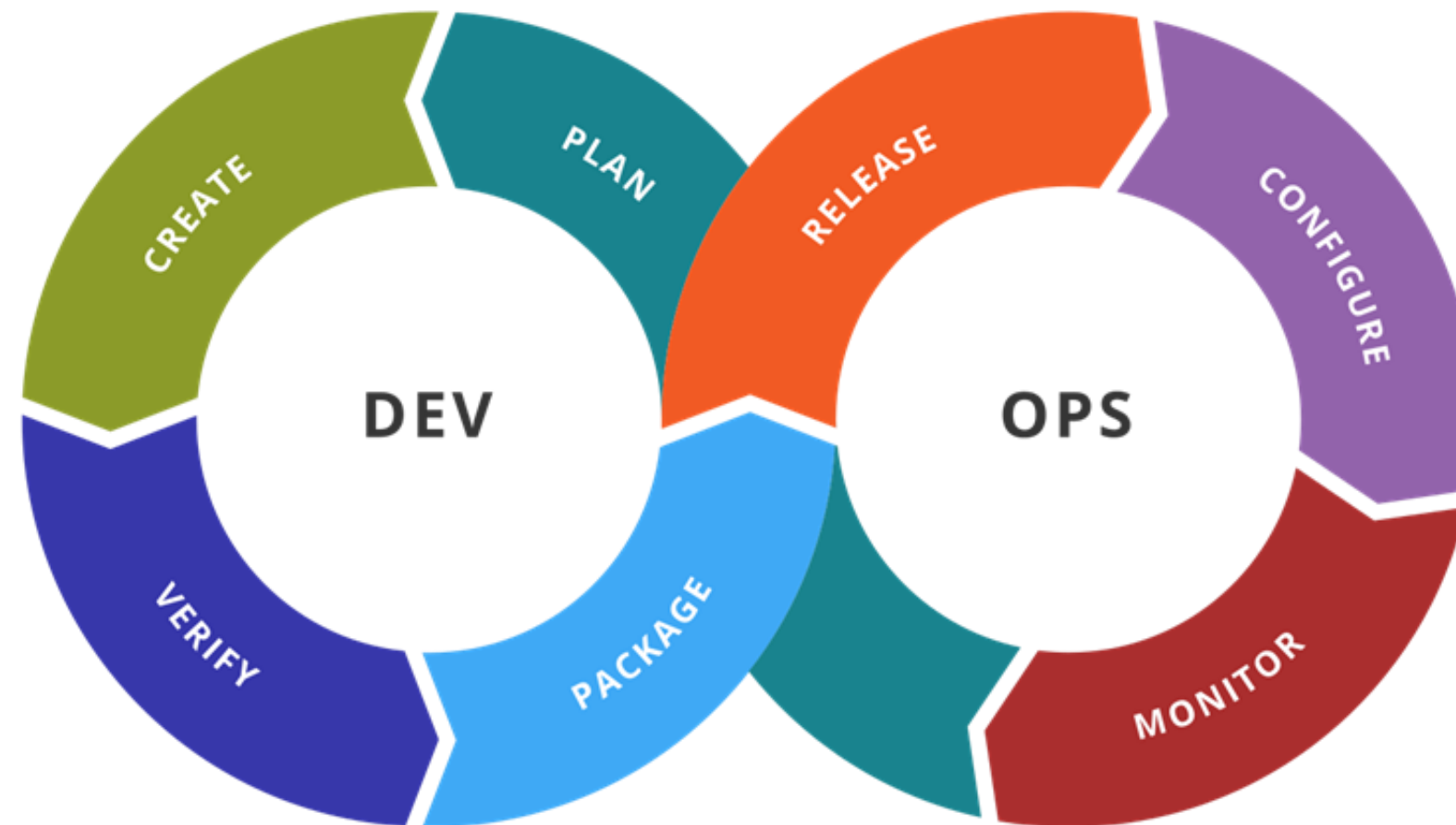
- 🕒 Explain DevOps
- 🕒 Illustrate DevOps lifecycle and its core components
- 🕒 Define basics of configuration management
- 🕒 Explain the tools used for configuration management



# Revisiting DevOps



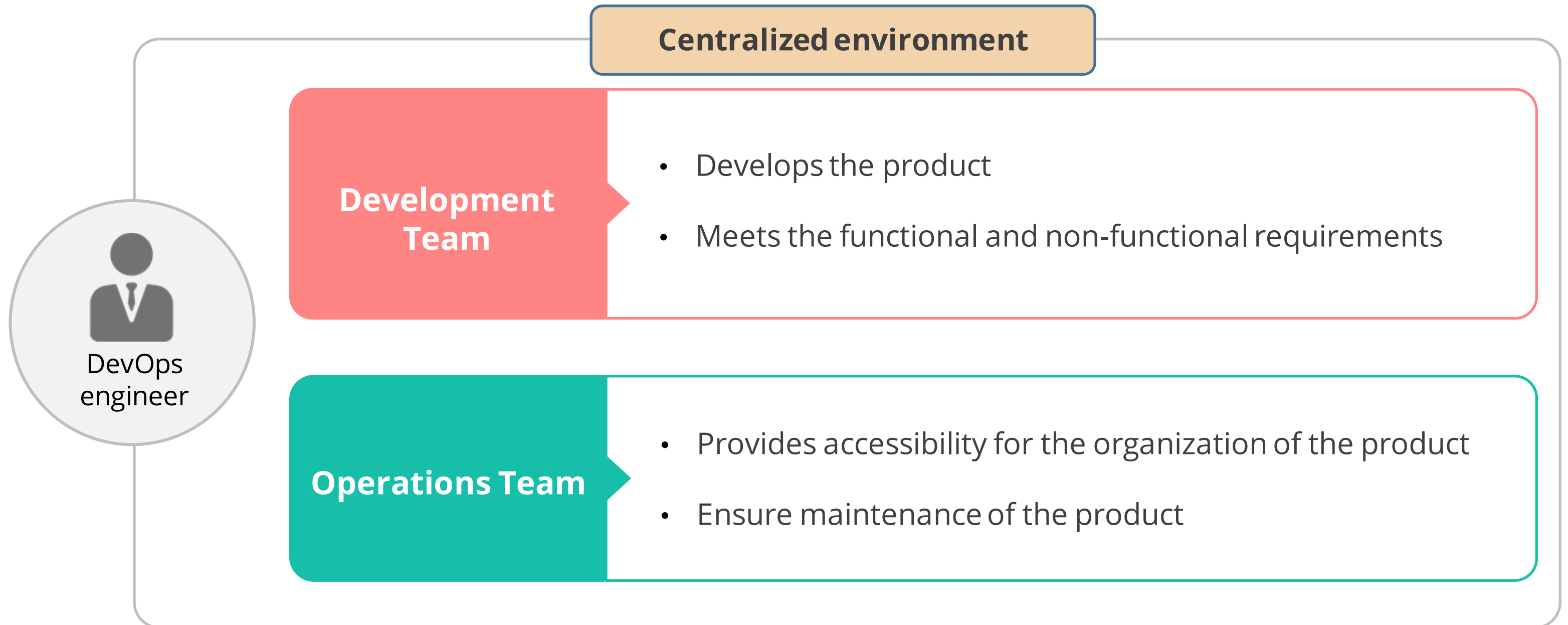
# What Is DevOps?



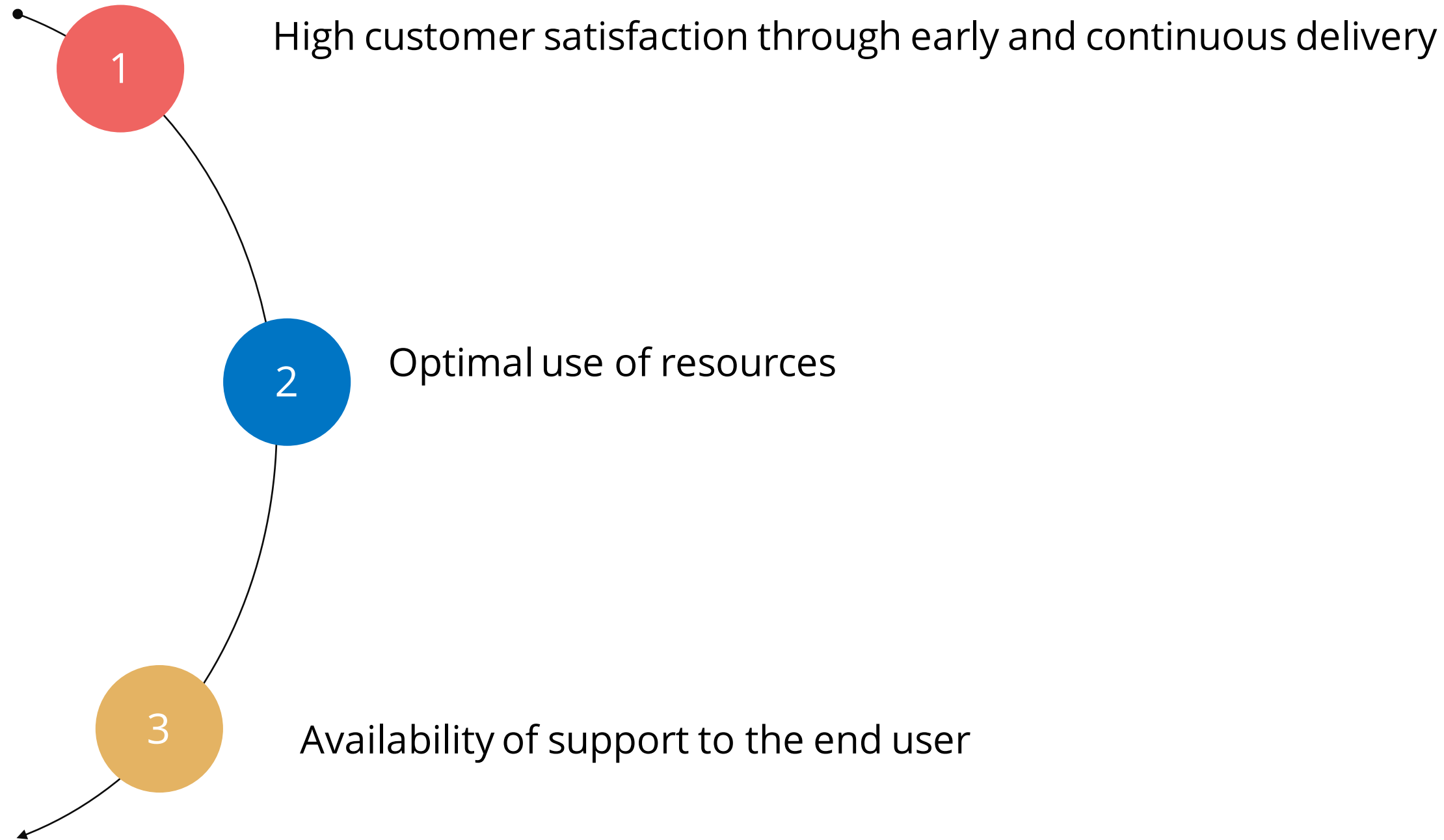
- DevOps is an agile relationship between development and IT operations.
- DevOps is the abbreviation for **De**velopment and **Op**erations.
- The Development phase includes Plan, Create, Verify, and Package.
- The Operations phase includes Release, Configure, and Monitor.

# Role of DevOps Engineer

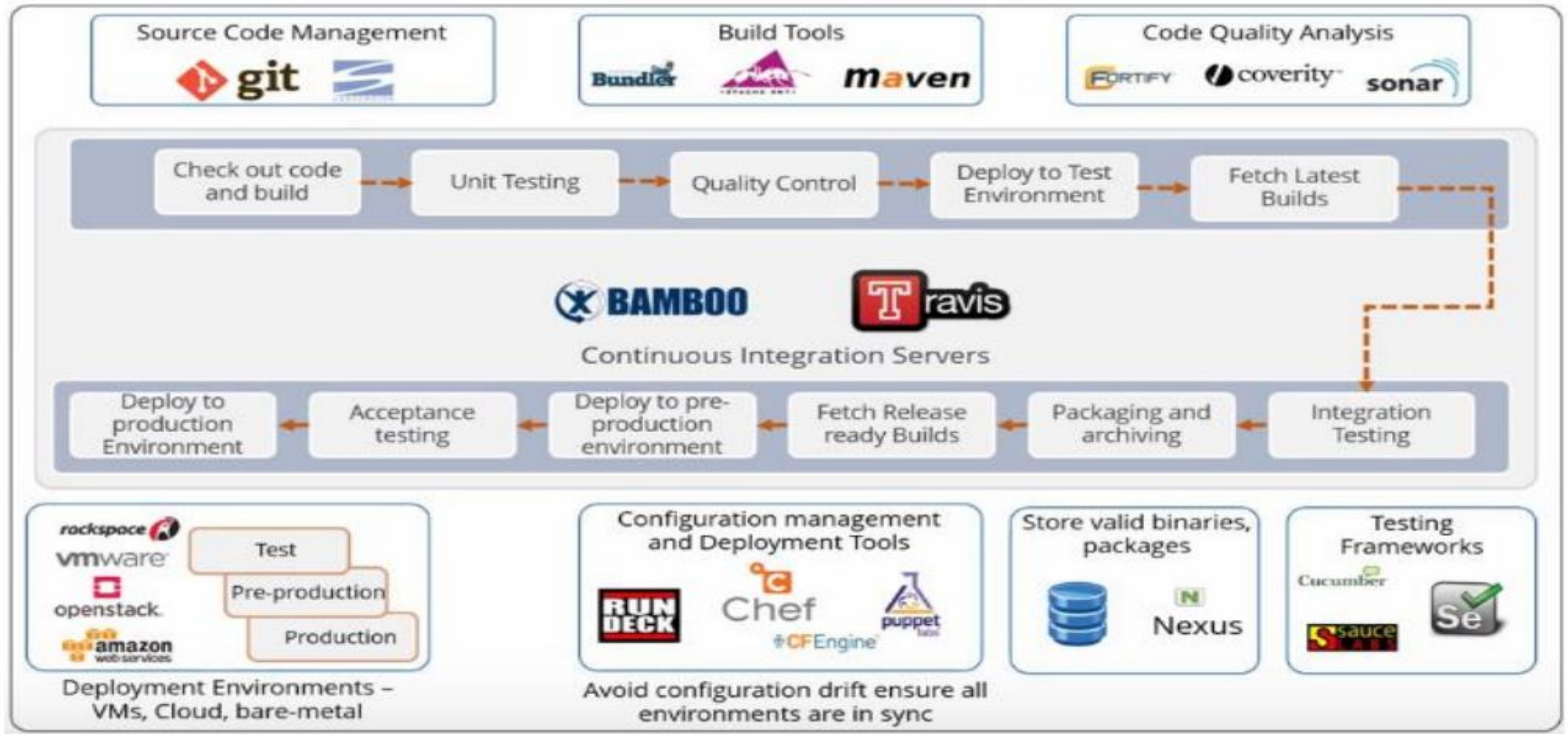
The major role of a DevOps engineer is to set up a centralized environment for different teams to collaborate and work together.



# Benefits of Agile and DevOps



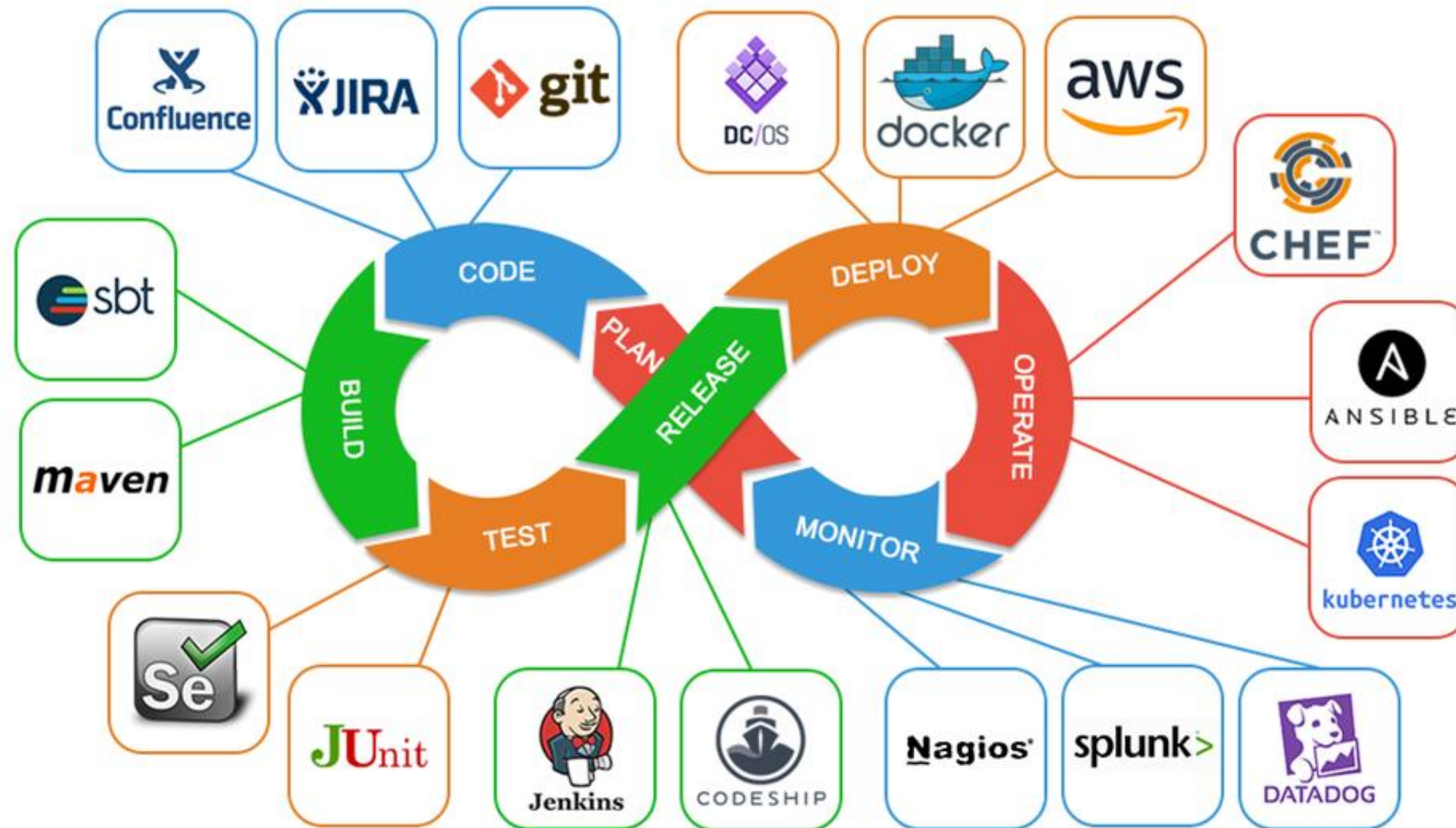
# Flow of Agile and DevOps



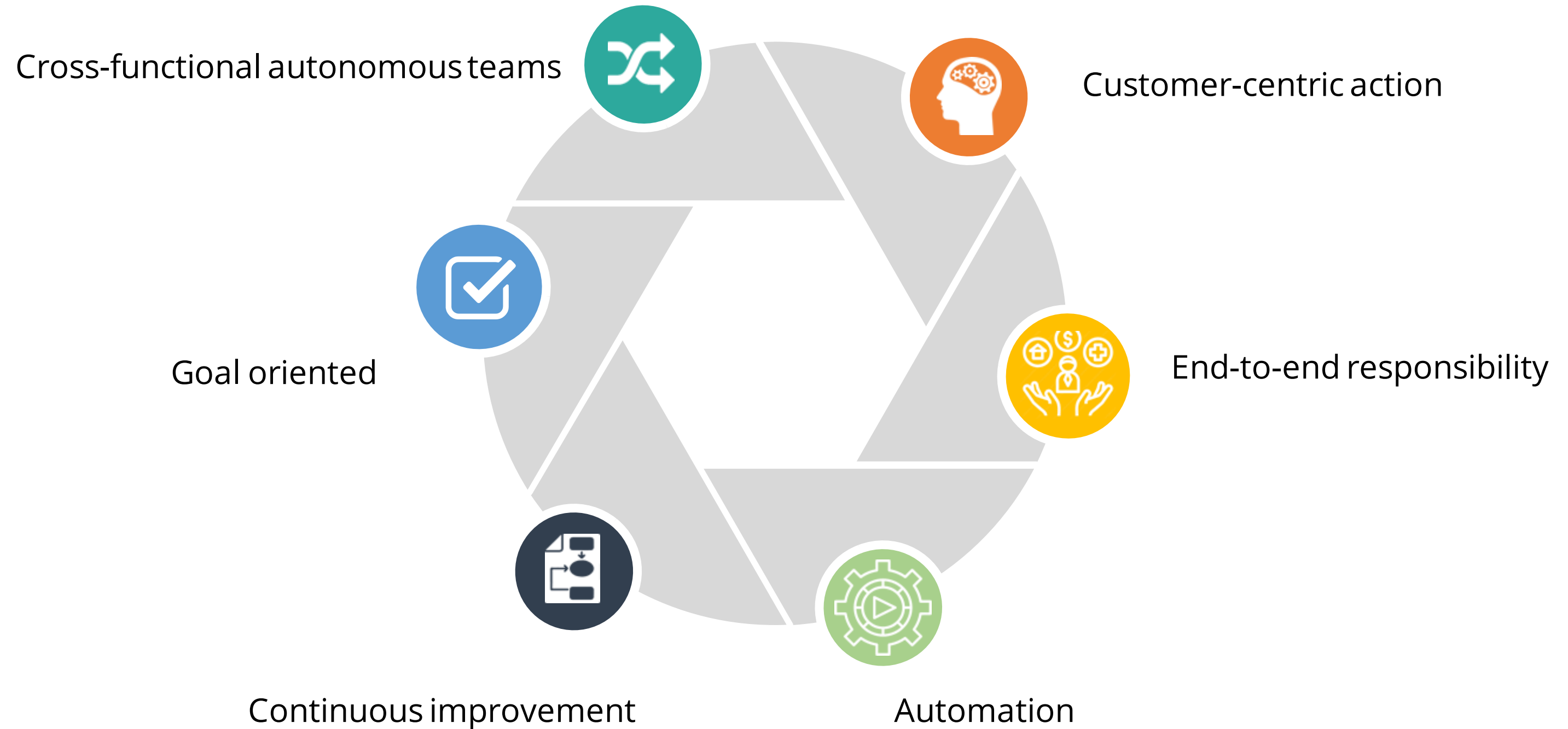


# DevOps Tools

The various tools required to implement DevOps and work within the DevOps setup are



# DevOps Principles



# Benefits of DevOps

## Speed

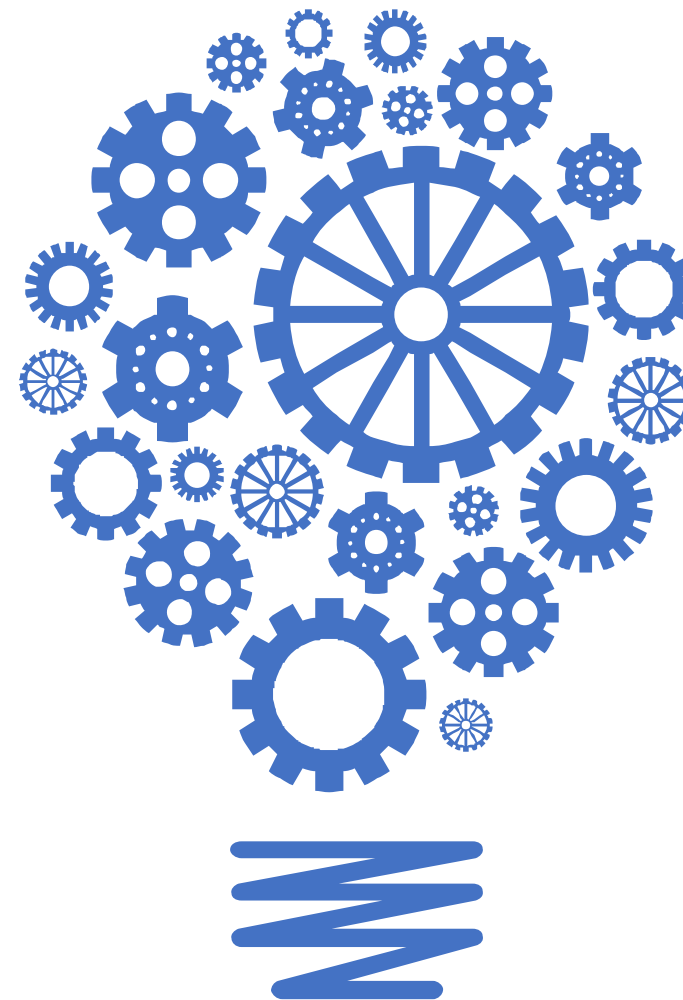
Automation through DevOps practices results in quick development and bug fixes.

## Rapid delivery

As DevOps works with Agile, it helps in delivering the finished product to the customer in less time.

## Reliability

It helps in improving and providing support to the customer with the help of CI/CD.



## Scalability

Automation and centralized working environment provide room for scaling the project whenever required.

## Collaboration

DevOps provides high interaction between different teams resulting in quality product and less bugs.

## Security

Automated compliance, fine-grained controls, and infrastructure as code are some of the policies used for security implementations while adopting DevOps.

# Configuration Management



# Configuration Management

Configuration Management (CM) is a systems engineering process for establishing and maintaining consistency of a product's performance and functional and physical attributes with its requirements, design, and operational information.

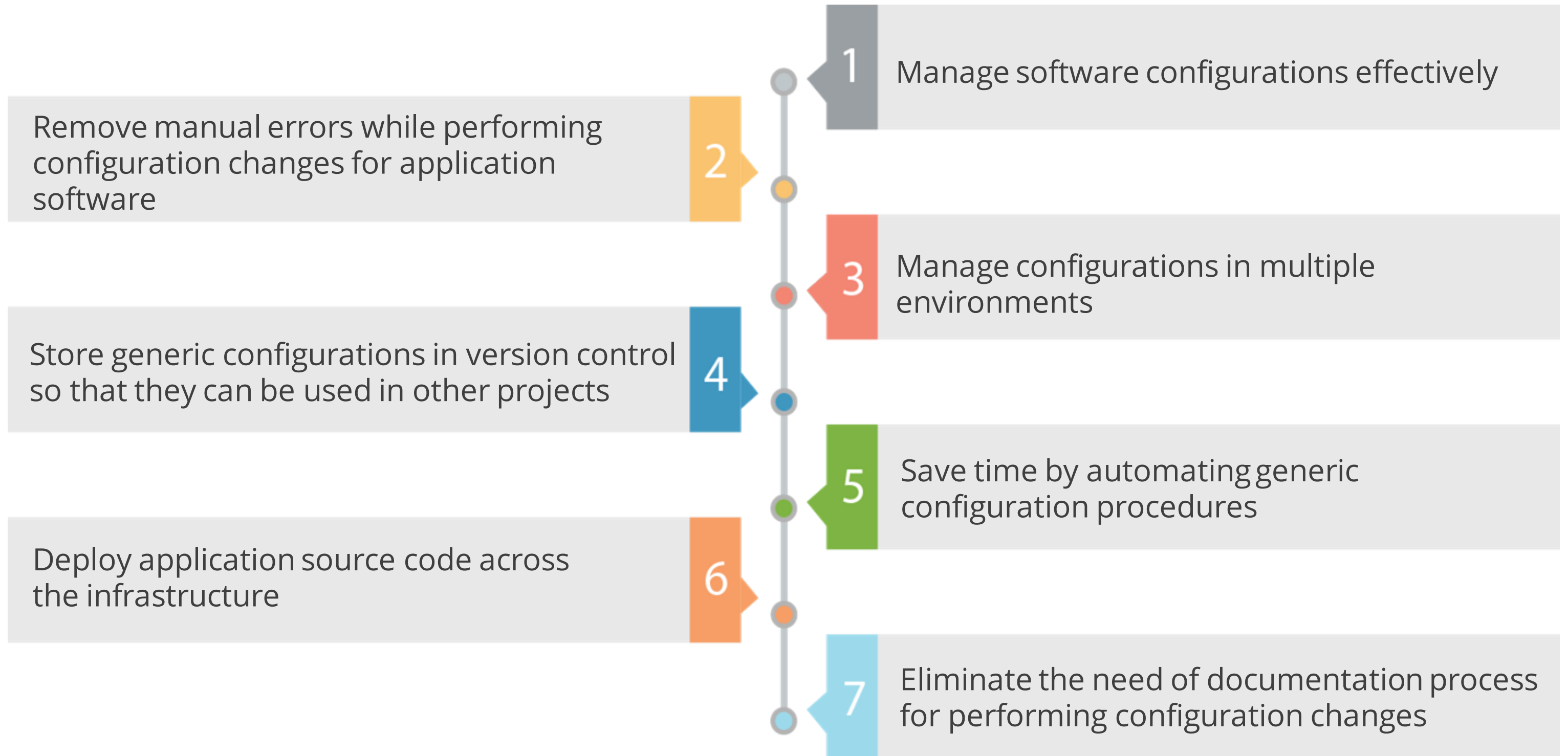


# Configuration Management Scope

---

- Configuration management tools manage all configuration items in a software for all environments.
- These configuration items can be software application files, software packages, and software installations which need to be configured for specific environments.
- Configuration management tools cover both software and server configurations.
- They also help reduce the time taken to manage configurations manually on each and every server.

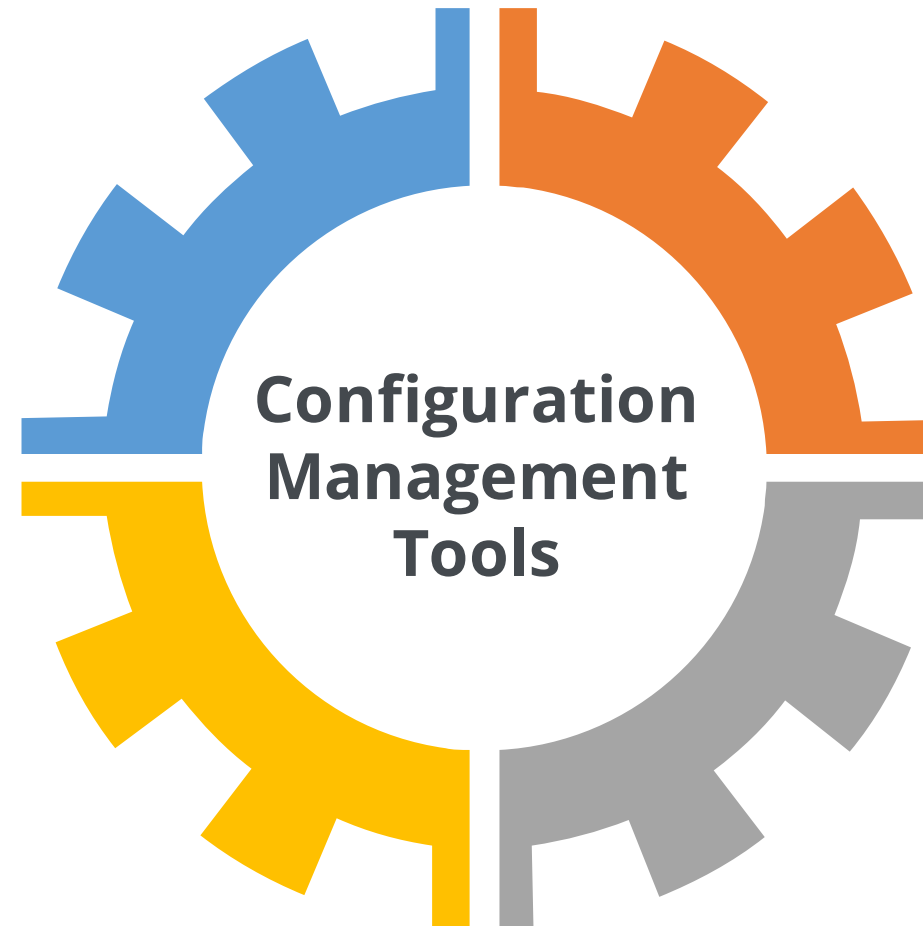
# Features of CM Tools



# CM Tools

**Puppet**  
Ruby DSL-based CM tool  
used for managing software,  
systems, and network  
configuration items

**Ansible**  
Python-based CM tool,  
also considered as  
agentless CM tool



**Chef**  
Ruby-based CM tool having  
integration with most of the  
cloud-based platforms

**SaltStack**  
Python-based open-source  
CM tool used to remotely  
manage configuration items



# CM Process

**Configuration Regulation**  
Regulates the way configuration changes are made for the application software

**Configuration Identification**  
Identifies the correct configuration that needs to be managed by CM tool



**Configuration Compliance**  
Audits and implements compliance on configuration changes made to application software

# Role of Infrastructure as Code in CM



# Differences between CM Tools

CM Tools	Pros	Cons
Ansible	<ul style="list-style-type: none"><li>• Simple architecture</li><li>• Low learning curve</li></ul>	<ul style="list-style-type: none"><li>• No Windows support for controller machine</li><li>• GUI is user friendly as compared to Chef and Puppet</li></ul>
Salt	<ul style="list-style-type: none"><li>• Scalable and fast</li><li>• Easy to manage</li></ul>	<ul style="list-style-type: none"><li>• GUI is not user friendly</li></ul>
Chef	<ul style="list-style-type: none"><li>• More features</li><li>• User friendly GUI</li></ul>	<ul style="list-style-type: none"><li>• Needs knowledge of Ruby</li><li>• Takes more time to understand</li><li>• Learning curve is more in Chef</li><li>• Only 10 nodes allowed in open-source license</li></ul>
Puppet	<ul style="list-style-type: none"><li>• Oldest tool in the market</li><li>• GUI with better features</li></ul>	<ul style="list-style-type: none"><li>• Difficult to understand and configure</li><li>• Has less number of integrations with other tools</li></ul>

# Overview of Chef

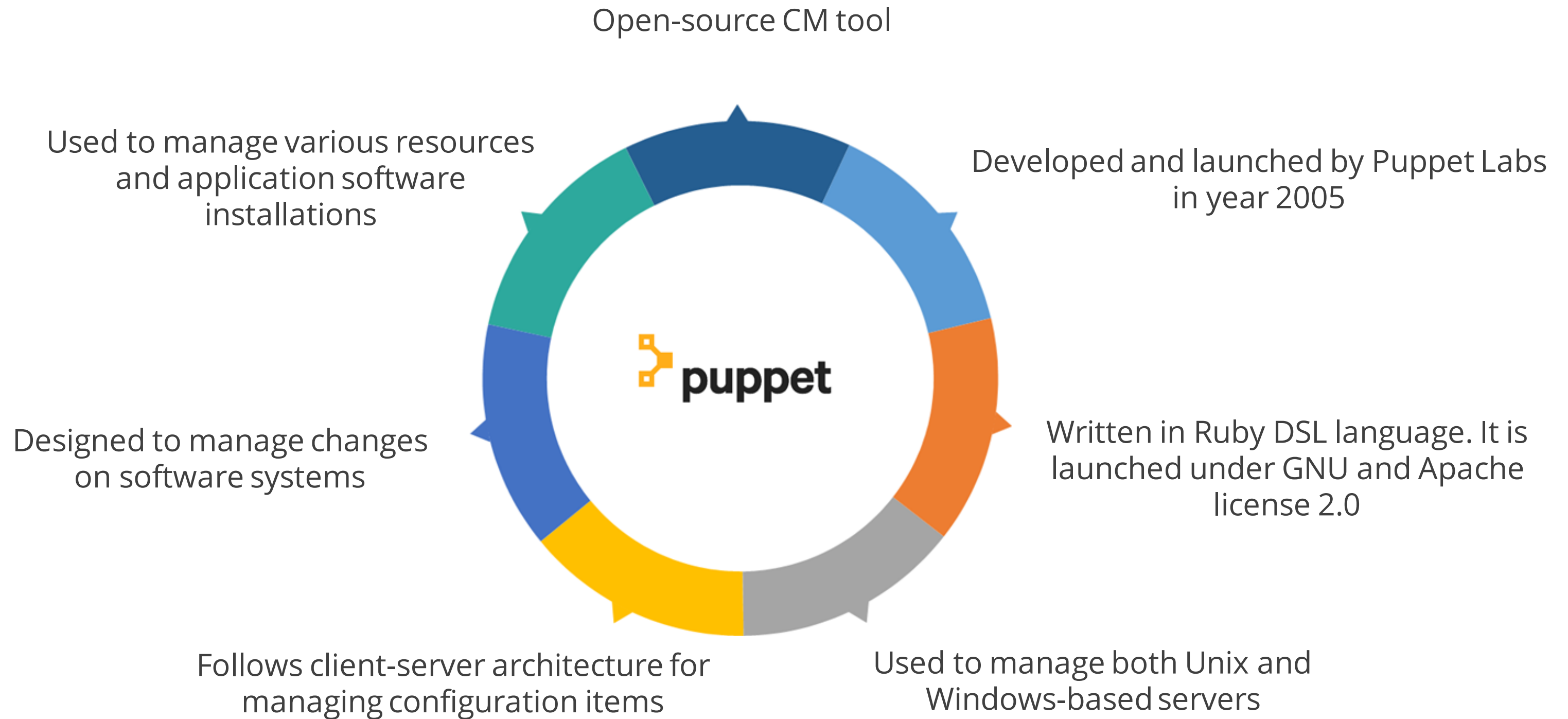
---

- Chef was developed in Ruby and Erlang languages. It was initially launched in year 2009.
- It uses pure Ruby DSL-based language for managing system resources.
- Its extensive integration with the latest cloud platforms like AWS GoogleCloud allows it to run in both client-server and solo architecture managing infra and software resources.
- It manages both Unix-based and Windows-based systems.

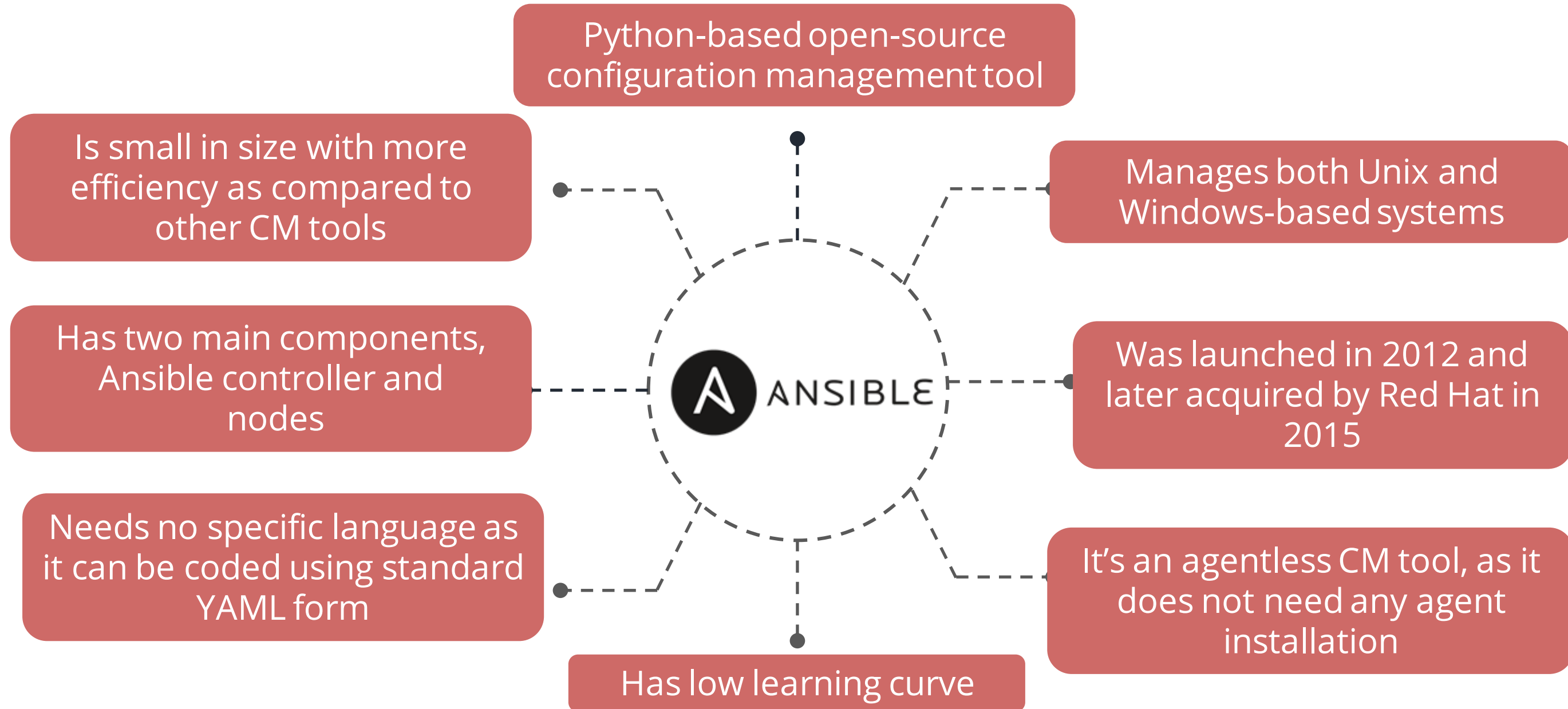




# Overview of Puppet



# Overview of Ansible



## Key Takeaways

- DevOps helps development and operations team to work in a collaborative manner.
- Configuration management tools manage all configuration items in a software for all environments.
- Agile and DevOps help in rapid development with less bugs in the product.

