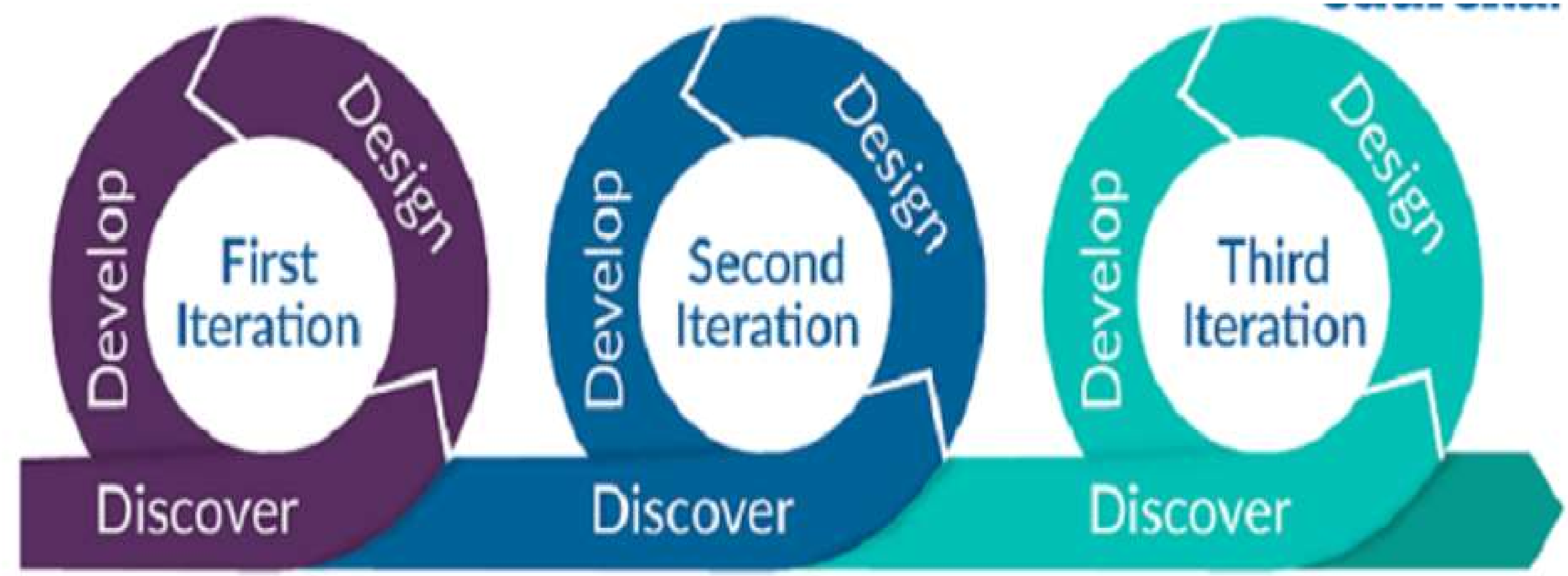
Waterfall model

- A sequential process in the development of a system or software that follows a top-down approach
- Unless you complete a particular phase, you could not proceed to the next phase, i.e., after Req analysis, Design, then Development and testing followed by Deployment.
- Also, the working software was delivered only after the final phase of this model.
- The major drawback is that requirements keep on changing from time to time and this model suffers a lot for that

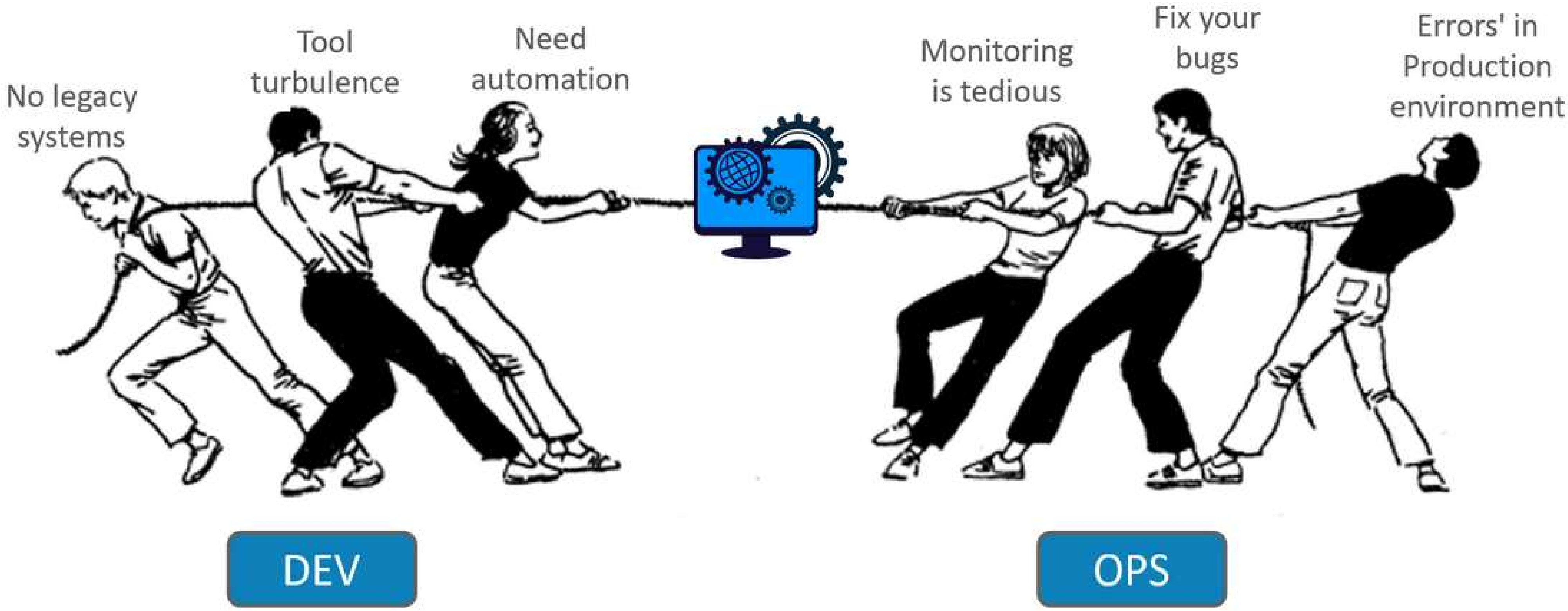
Agile methodlogy



It encourages continuous iteration of development and testing throughout the software development life cycle of the project.

Issues evolved with Agile

- There was a lack of collaboration between Developers and Operation Engineers and this slowed down the development process and releases.
- Software companies had begun to realize the need for better collaboration between the teams and faster delivery of software.
- This gave birth to the DevOps approach.
- DevOps enabled continuous software delivery with less complex problems to fix and faster resolution of problems.



No legacy systems

Tool turbulence

Need automation

Monitoring is tedious

Fix your bugs

Errors' in Production environment

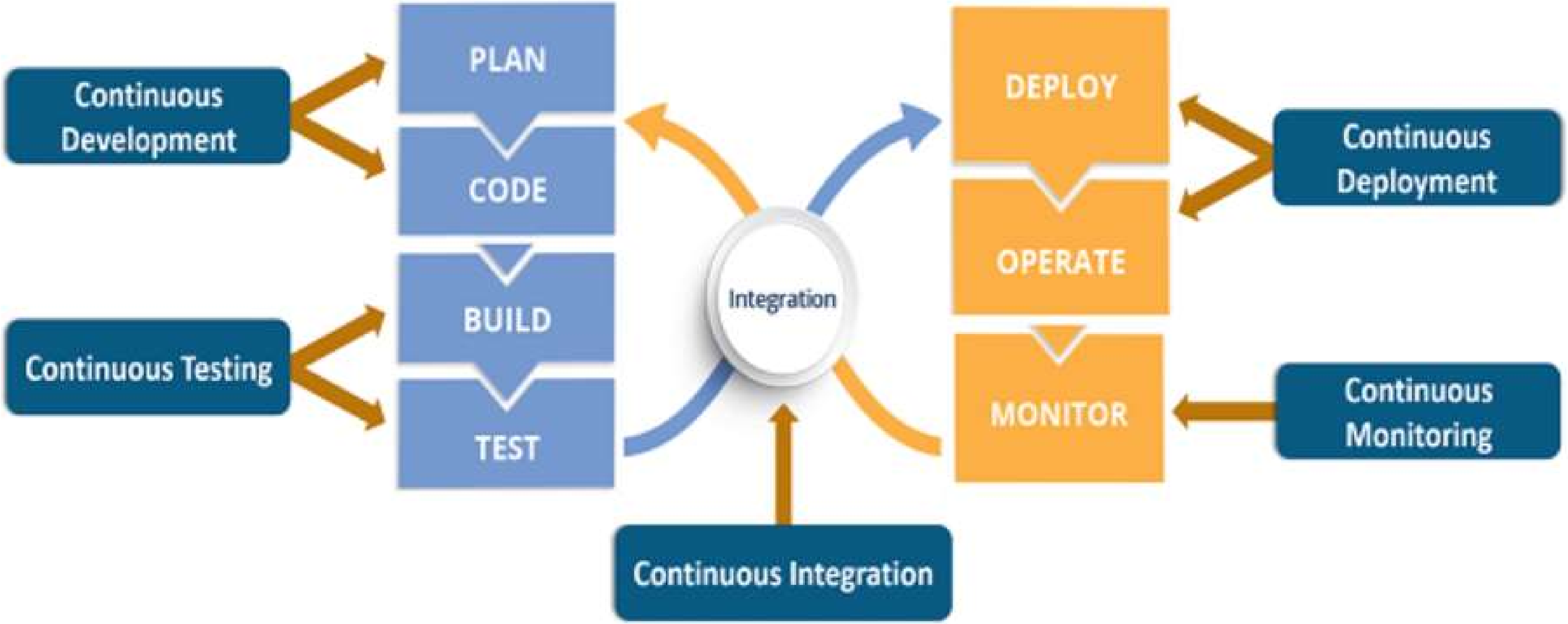
DEV

OPS

DevOps

- The term DevOps is a combination of two words namely Development and Operations.
- DevOps is a practice that allows a single team to manage the entire application development life cycle, that is, development, testing, deployment, operations.
- It consists of various stages such as continuous development, continuous integration, continuous testing, continuous deployment, and continuous monitoring.

DevOps Life Cycle



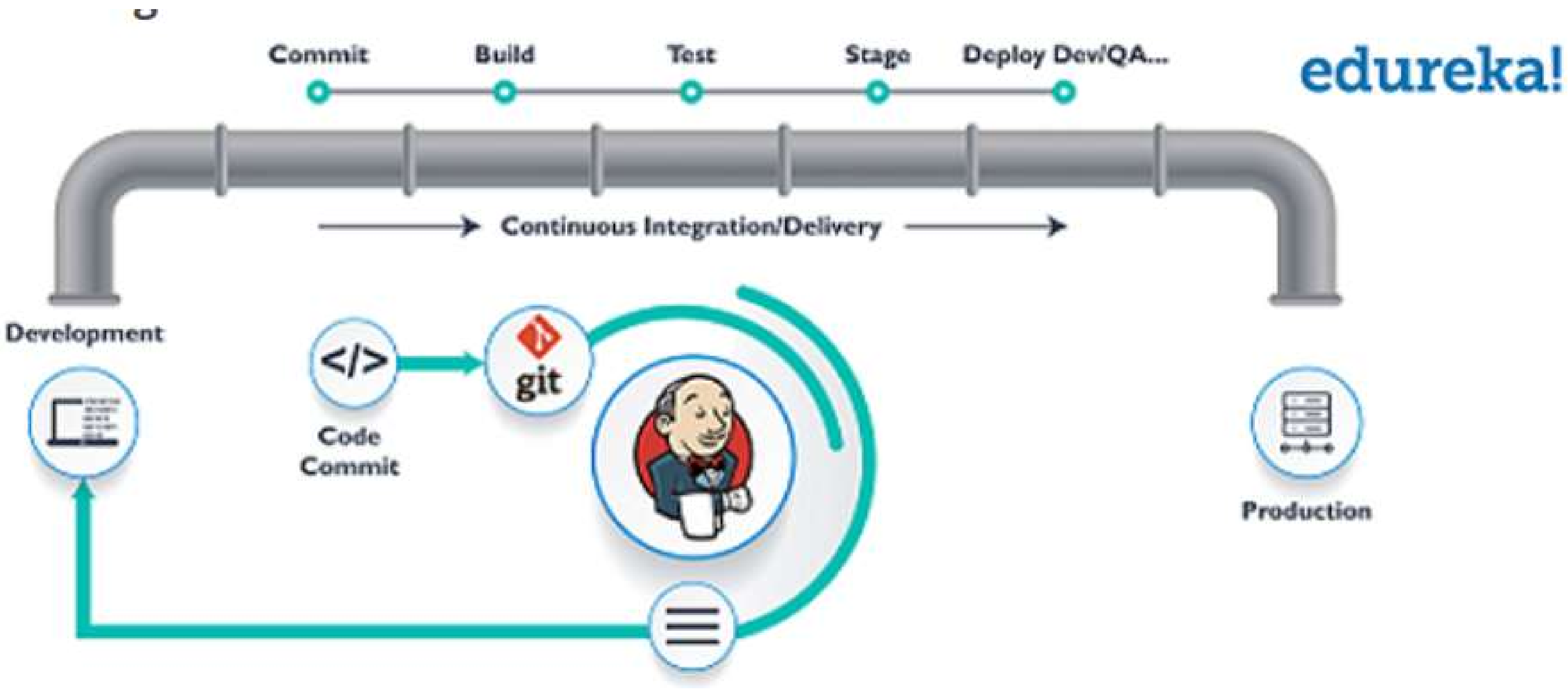
Continuous Development

- This is the phase that involves ‘planning‘ and ‘coding‘ of the software.
- The vision of the project is decided during the planning phase and the developers begin developing the code for the application.
- It is maintained by using Version Control tools. Maintaining the code is referred to as Source Code Management.
- The most popular tools used are Git, SVN, Mercurial, CVS, and JIRA.
- Also tools like Ant, Maven, Gradle can be used in this phase for building/ packaging the code into an executable file that can be forwarded to any of the next phases.

Continuous Testing

- Automation tools like **Selenium**, **TestNG**, **JUnit** are used.
- These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there are no flaws in the functionality.
- Selenium does the automation testing, and reports are generated by TestNG.
- This entire testing phase can be automated with the help of a Continuous Integration tool called Jenkins.
- Suppose you have written a selenium code in Java to test your application. Now you can build this code using ant or maven. Once the code is built, it is tested for User Acceptance Testing (UAT).
- This entire process can be automated using Jenkins.
- We can also schedule the execution of the test cases at predefined times.
- Finally, the code is continuously integrated with the existing code.

Continuous Integration



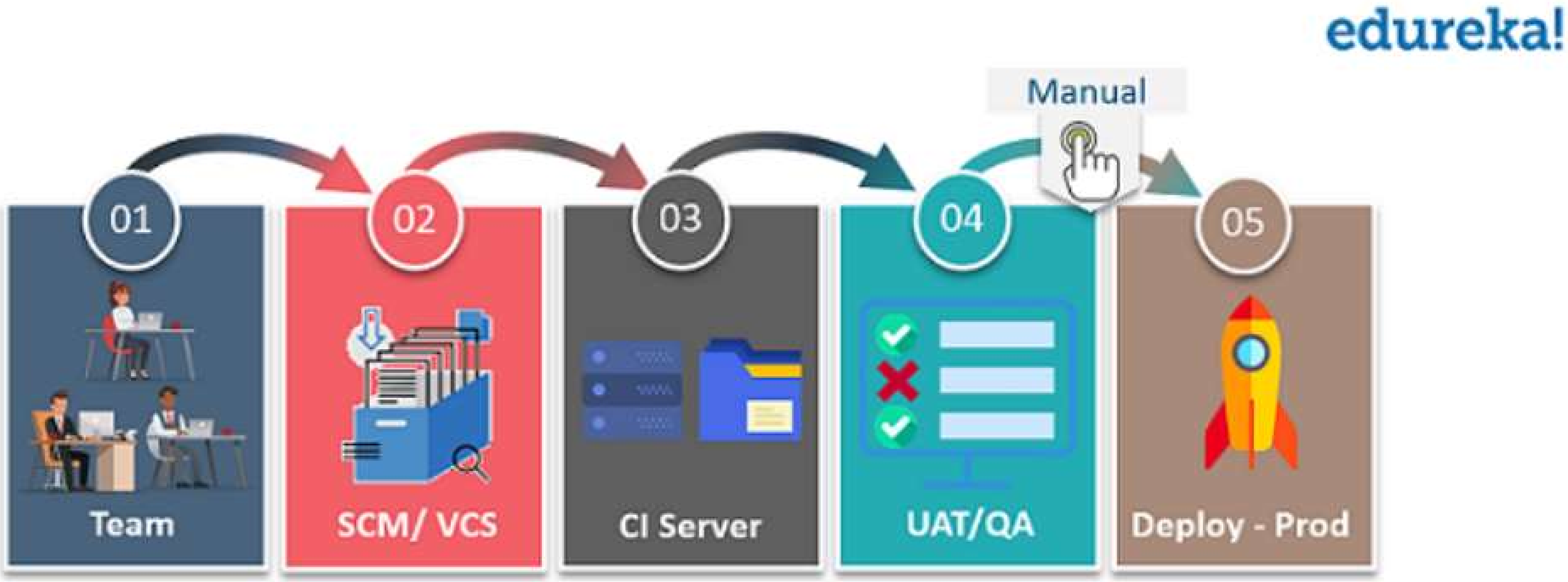
Continuous Integration

- This stage is the heart of the entire DevOps life cycle. 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.
- It is a software development practice in which the developers require to commit changes to the source code more frequently. This may be on a daily or a weekly basis.
- Every commit is then built and this allows early detection of problems if they are present.
- Building code not only involves compilation but it also includes code review, unit testing, integration testing, and packaging.

Continuous Integration

- Jenkins is a very popular tool used in this phase.
- Whenever there is a change in the Git or other such repository, Jenkins fetches the updated code and it prepares a build of that code which is an executable file in the form of a war or a jar.
- This build is then forwarded to the test server or the production server.

Continuous Deployment (CD)



CD is the stage where the code is deployed to the production servers.

Configuration management

- Configuration management tools help in achieving Continuous Deployment.
- Configuration Management is the act of establishing and maintaining consistency in an application’s functional requirements and performance.
- It is the act of releasing deployments to servers, scheduling updates on all servers and most importantly keeping the configurations consistent across all the servers.
- Since the new code is deployed on a continuous basis, configuration management tools play an important role in executing tasks quickly and frequently.
- Some popular tools that are used here are Puppet, Chef, SaltStack, and Ansible.

Containerization

- A container is an isolated environment for your code. A container usually has everything that your code needs in order to run, down to a base operating system.
- Docker is the most popular tool used for containerization. It is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.
- It helps produce consistency across Development, Test, Staging and Production environments.
- It also helps in scaling-up and scaling-down of instances swiftly.
- Using these tools, there is no scope of errors/ failure in the production environment as they package and replicate the same dependencies and packages used in the development/ testing/ staging environment.
- It makes your application easy to run on different computers.

Continuous Monitoring

- This practice involves the participation of the Operations team who will monitor the user activity for bugs or any improper behavior of the system. The popular tools used for this are Splunk, ELK Stack, Nagios, NewRelic , etc.
- These tools help you monitor the application’s performance and the servers closely and also enable you to check the health of the system proactively.
- They can also improve productivity and increase the reliability of the systems, which in turn reduces IT support costs.
- Any major issues if found are reported to the development team so that it can be fixed in the continuous development phase. This leads to a faster resolution of the problems.