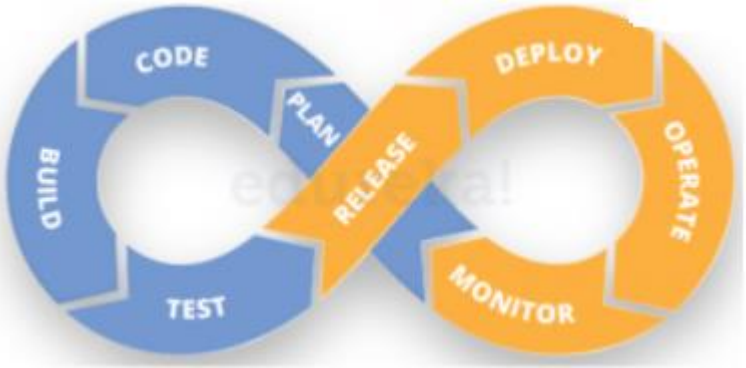
Welcome to DevOps



What is DevOps?

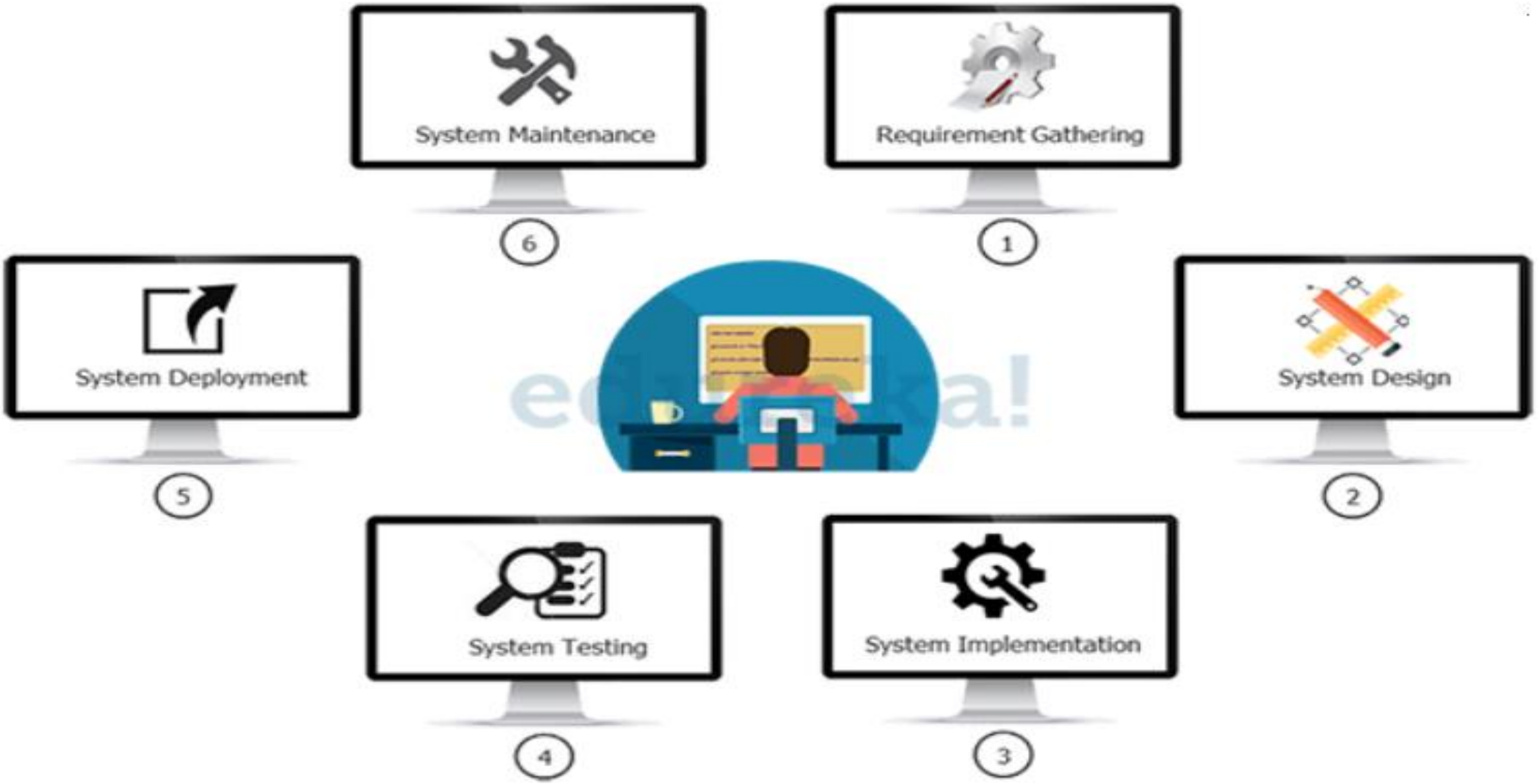
- DevOps is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams.
- It emphasizes team empowerment, cross-team communication and collaboration, and technology automation.
- The DevOps movement began around 2007 when the software development and IT operations communities raised concerns about the traditional software development model, where developers who wrote code worked apart from operations who deployed and supported the code.
- The term DevOps, a combination of the words development and operations, reflects the process of integrating these disciplines into one, continuous process.
- DevOps promotes communication between development engineers and operations, participating together in the entire service life-cycle, from design through the development process to production support.
- What led DevOps to come into existence?

Waterfall Model

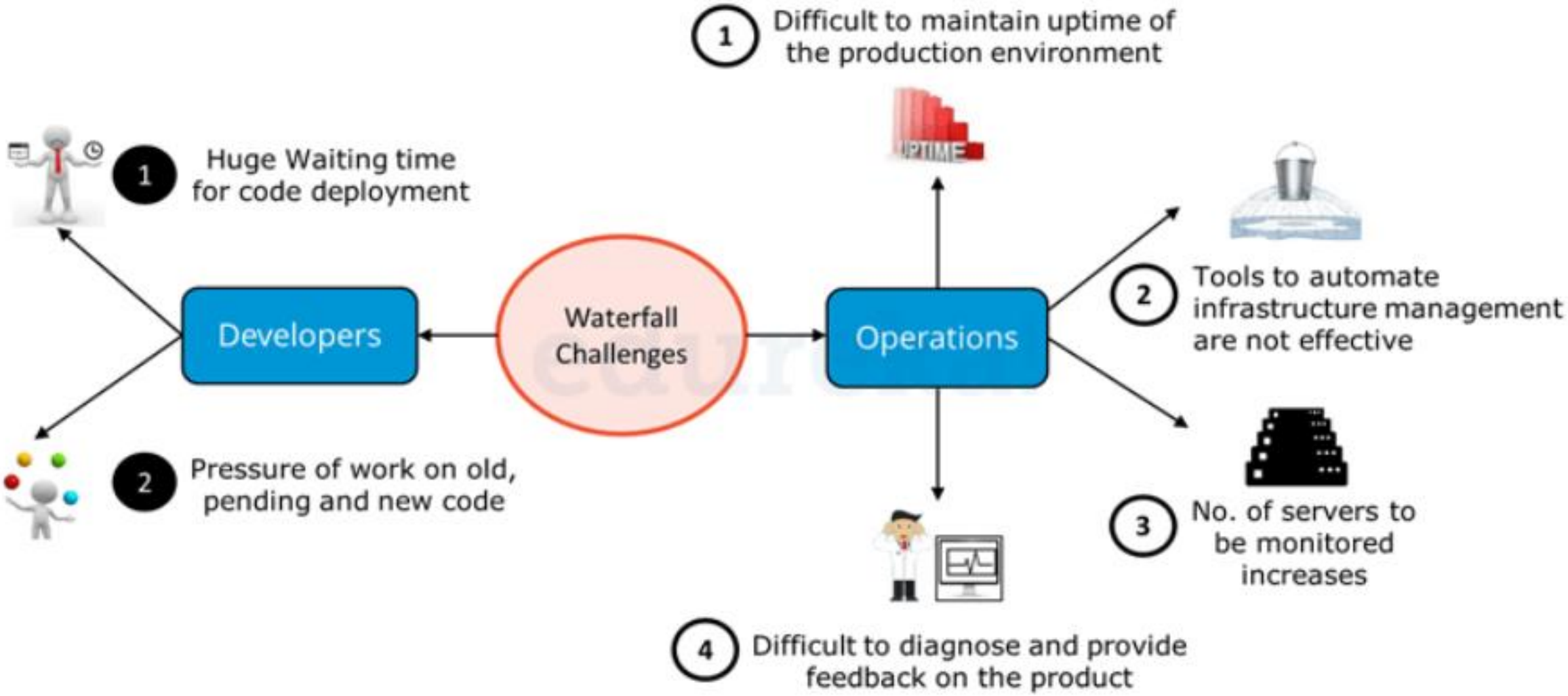
Let's consider developing software in a traditional way using a Waterfall Model.

- Phase 1 – Complete Requirement is gathered and SRS is developed
- Phase 2 – This System is Planned and Designed using the SRS
- Phase 3 – Implementation of the System takes place
- Phase 4 – System is tested and its quality is assured
- Phase 5 – System is deployed to the end users
- Phase 6 – Regular Maintenance of the system is done

Waterfall Model



Waterfall Model Challenges



What is Agile?

- In Agile development, a sprint is a set period of time during which specific work has to be completed and made ready for review.
- Each sprint begins with a planning meeting. During the meeting, the product owner (the person requesting the work) and the development team agree upon exactly what work will be accomplished during the sprint.
- The development team has the final say when it comes to determining how much work can realistically be accomplished during the sprint, and the product owner has the final say on what criteria need to be met for the work to be approved and accepted.
- The duration of a sprint is determined by the scrum master, the team's facilitator and manager of the Scrum framework.
- Once the team reaches a consensus for how many days a sprint should last, all future sprints should be the same. Traditionally, a sprint lasts from 1 week to 6 weeks.

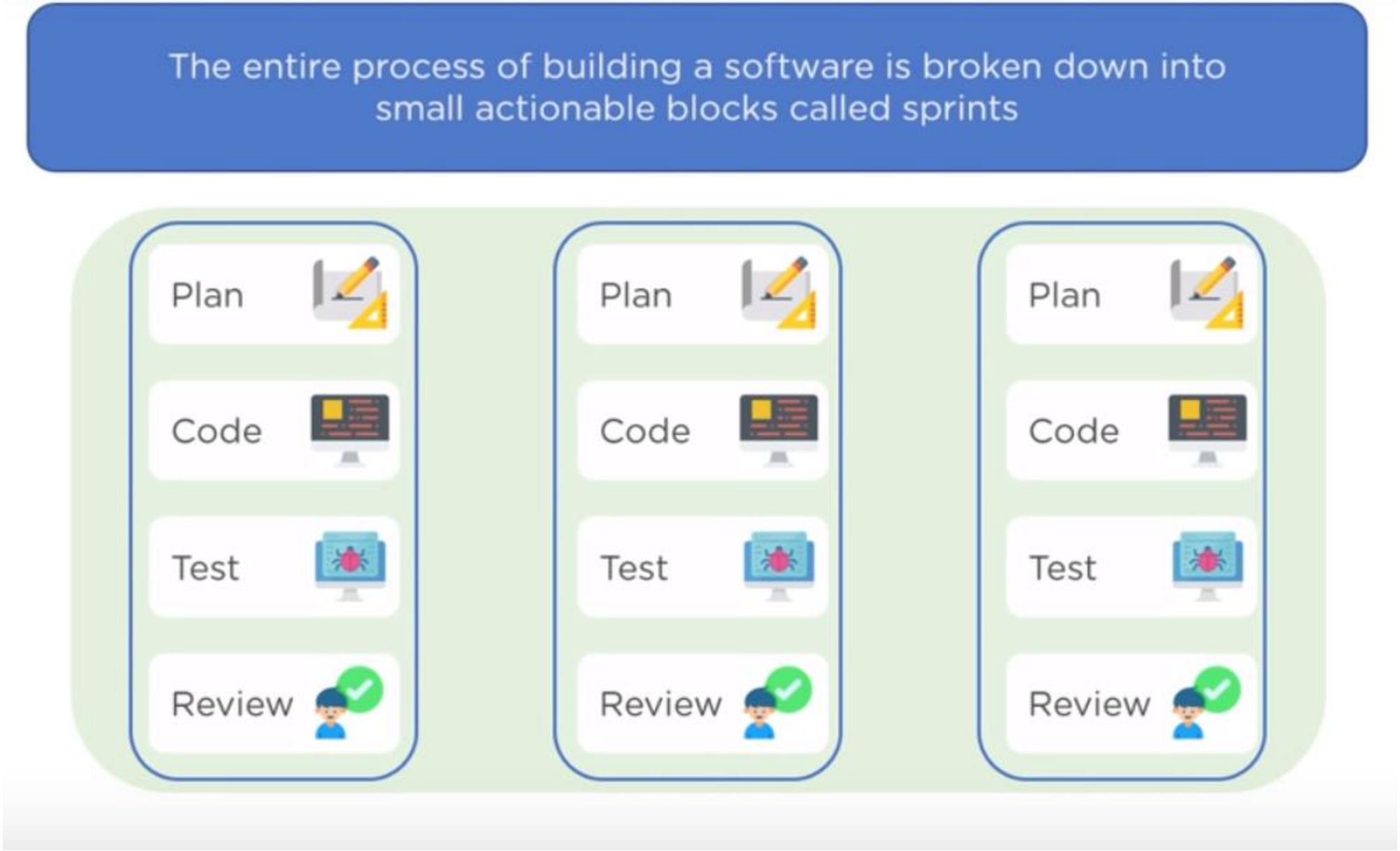
Sprint workflow and process

- The sprint workflow is intended to help team members evaluate their work and communicate with each other throughout the entire process. The workflow is followed for each sprint. The process includes:
- **Backlog -** A list of set tasks that must be completed before the product is released. The backlog is built by the product owner. The product owner gives a backlog of prioritized items to the scrum master and scrum team. The backlog is based on user stories, which focus on features that consider the type of end user, what they want and why.
- **Sprint planning -** The team discusses top priority user stories and decides what can be delivered in the sprint.
- **Sprint backlog -** Agreed upon by the entire team, this list finalizes and defines what the development team will complete during the sprint.
- **Sprint –** The time frame in which the work must be completed – often 1 week to 6 weeks.

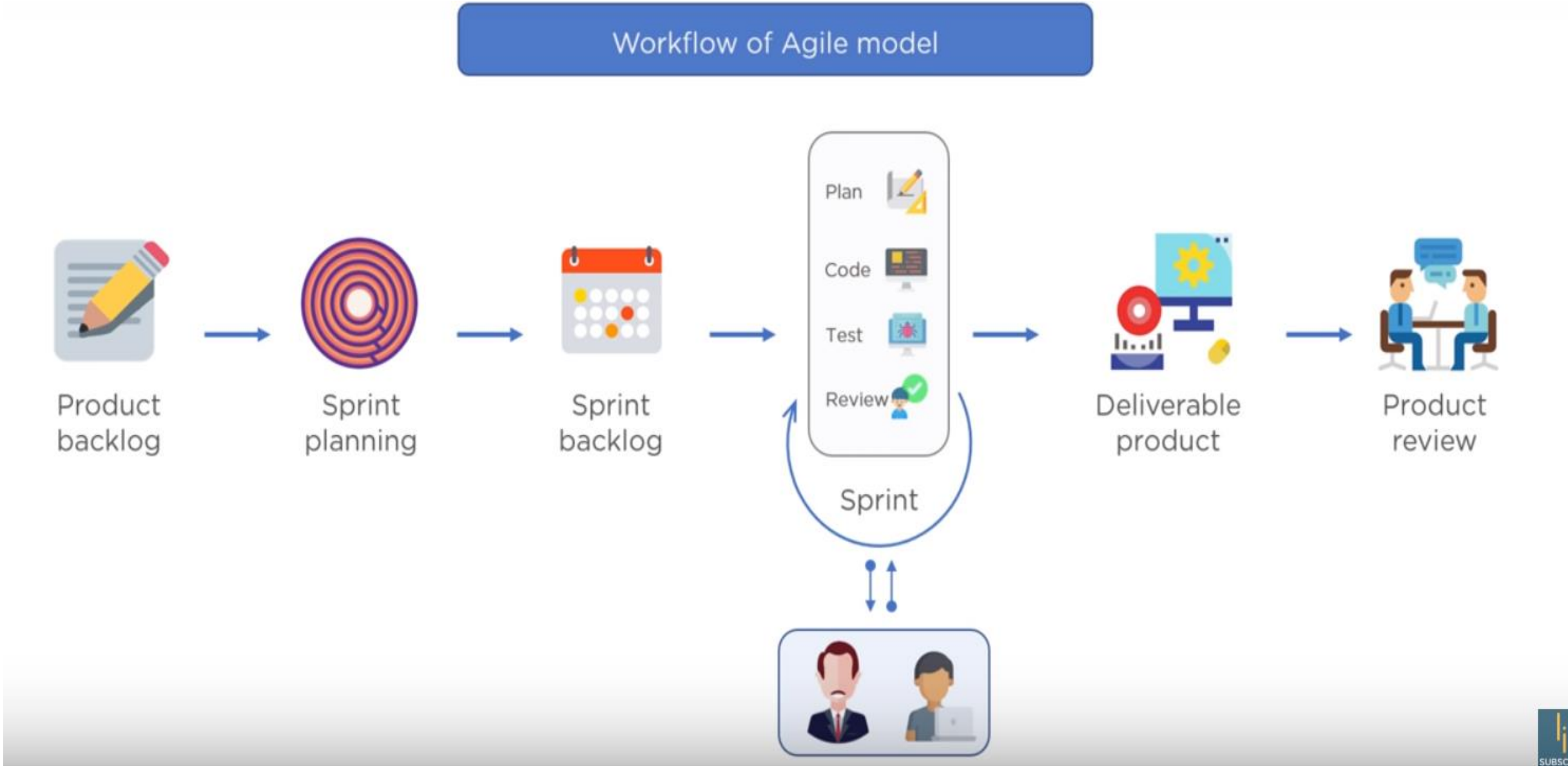
Sprint workflow and process

- **Daily scrum –** Lead by the scrum master, the team comes together for short daily meetings, in which they discuss what they have completed, what they are working on and any issues that are blocking the work.
- **Outcome -** The outcome of a sprint is a hypothetically usable product. The product owner can decide if the product is ready or if additional features are needed.
- **Sprint end -** At the end of a sprint, two meetings are held:
  - **Sprint review –** The team shows their work to the product owner.
  - **Sprint retrospective –** The team discusses what they can do to improve processes. An important goal is continuous improvement.

Agile Model


The entire process of building a software is broken down into small actionable blocks called sprints
Plan
Code
Test
Review
Plan
Code
Test
Review
Plan
Code
Test
Review

Agile Workflow



Agile Model

When the product fails in production servers, the operations team are clueless and send product back to the development team

Evolution of Software Development

Traditional Waterfall Model

Best suited when:

- Complete Requirements are clear and fixed
- Product definition is stable

Agile Development

Best suited when:

- Requirements change frequently
- Development needs to be fast

DevOps Approach

Best suited when:

- Requirements change frequently
- **Development** needs to be Agile
- **Operations** needs to be Agile

Proposed Solutions

Dev Challenges	DevOps Solution
Waiting time for code deployment	• Continuous Integration ensures there is quick deployment of code, faster testing and speedy feedback mechanism • Thus there is no waiting time to deploy the code. Hence the developer focuses on building the current code
Pressure of work on old, pending and new code	

Ops Challenges	DevOps Solution
Difficult to maintain uptime of the production environment	Containerization / Virtualization ensures there is a simulated environment created to run the software as containers offer great reliability for service uptime
Tools to automate infrastructure management are not effective	Configuration Management helps you to organize and execute configuration plans, consistently provision the system, and proactively manage their infrastructure
No. of servers to be monitored increases	Continuous Monitoring Effective monitoring and feedbacks system is established through Nagios Thus effective administration is assured
Difficult to diagnose and provide feedback on the product	

How does DevOps work?

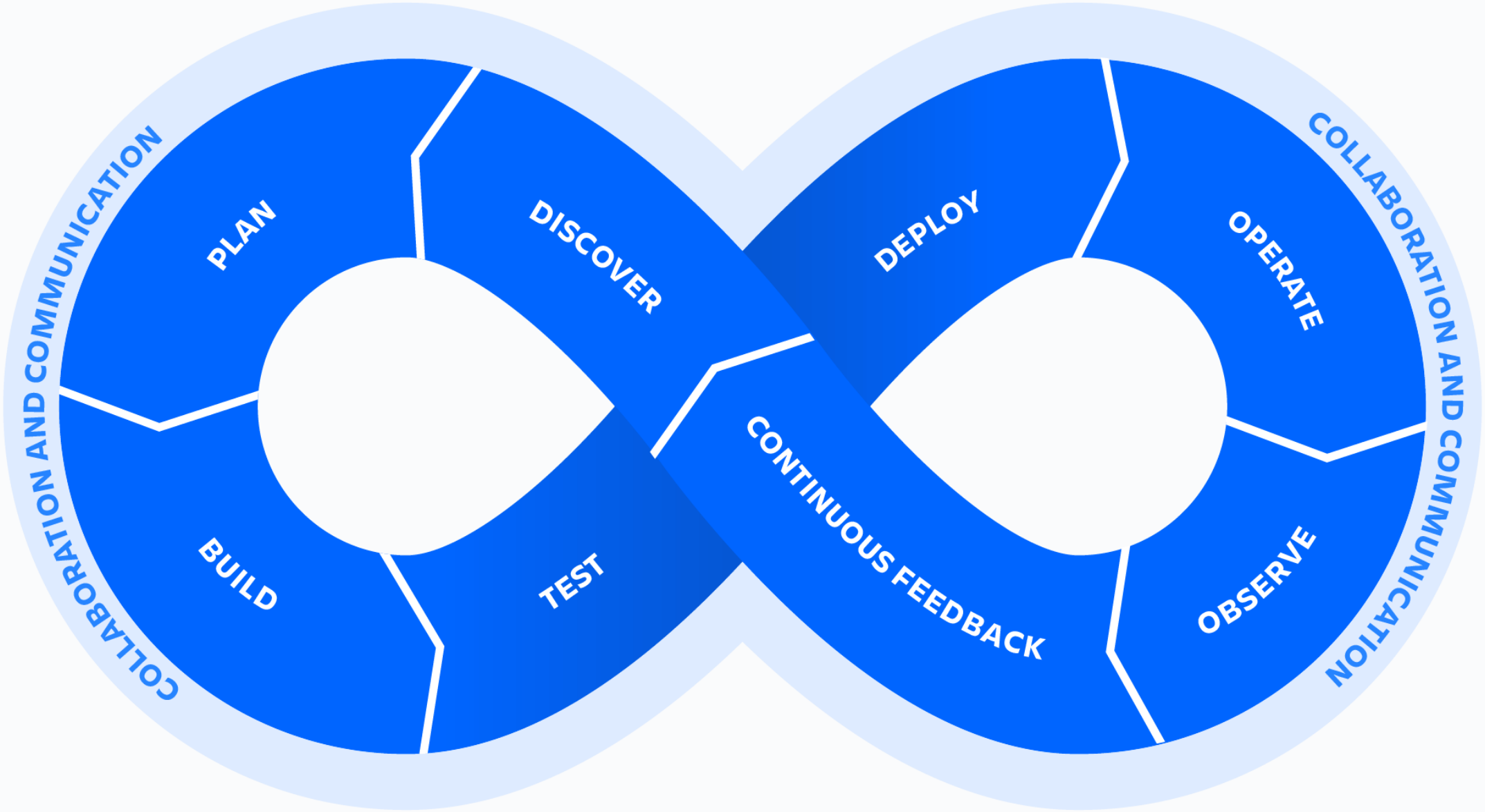
- A DevOps team includes developers and IT operations working collaboratively throughout the product lifecycle, in order to increase the speed and quality of software deployment.
- It’s a new way of working, a cultural shift, that has significant implications for teams and the organizations they work for.
- Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams merge into a single team where the engineers work across the entire application lifecycle — from development and test to deployment and operations — and have a range of multidisciplinary skills.
- DevOps teams use tools to automate and accelerate processes, which helps to increase reliability.
- A DevOps toolchain helps teams tackle important DevOps fundamentals including continuous integration, continuous delivery, automation, and collaboration.
- When security teams adopt a DevOps approach, security is an active and integrated part of the development process. This is called DevSecOps.

The DevOps lifecycle

- Because of the continuous nature of DevOps, practitioners use the infinity loop to show how the phases of the DevOps lifecycle relate to each other.
- Despite appearing to flow sequentially, the loop symbolizes the need for constant collaboration and iterative improvement throughout the entire lifecycle.

- The DevOps lifecycle consists of eight phases representing the processes, capabilities, and tools needed for development (on the left side of the loop) and operations (on the right side of the loop). Throughout each phase, teams collaborate and communicate to maintain alignment, velocity, and quality.

The DevOps lifecycle



The DevOps Phases

Discover

- Building software is a team effort.
- In preparation for the upcoming sprint, teams must workshop to explore, organize, and prioritize ideas.
- Ideas must align to strategic goals and deliver customer impact.

Plan

- DevOps teams should adopt agile practices to improve speed and quality.
- Agile is an iterative approach to project management and software development that helps teams break work into smaller pieces to deliver incremental value.

Build

- Developers start programming the application and add them into version control systems like Git

The DevOps Phases

Test

- Continuous integration (CI) allows multiple developers to contribute to a single shared repository.
- When code changes are merged, automated tests are run to ensure correctness before integration.
- Merging and testing code often help development teams gain reassurance in the quality and predictability of code once deployed.

Deploy

- Continuous deployment (CD) allows teams to release features frequently into production in an automated fashion.
- Teams also have the option to deploy with feature flags, delivering new code to users steadily and methodically rather than all at once. This approach improves velocity, productivity, and sustainability of software development teams.

The DevOps Phases

Operate

- Manage the end-to-end delivery of IT services to customers.
- This includes the practices involved in design, implementation, configuration, deployment, and maintenance of all IT infrastructure that supports an organization’s services.

Observe

- Quickly identify and resolve issues that impact product uptime, speed, and functionality.
- Automatically notify your team of changes, high-risk actions, or failures, so you can keep services on.

The DevOps Phases

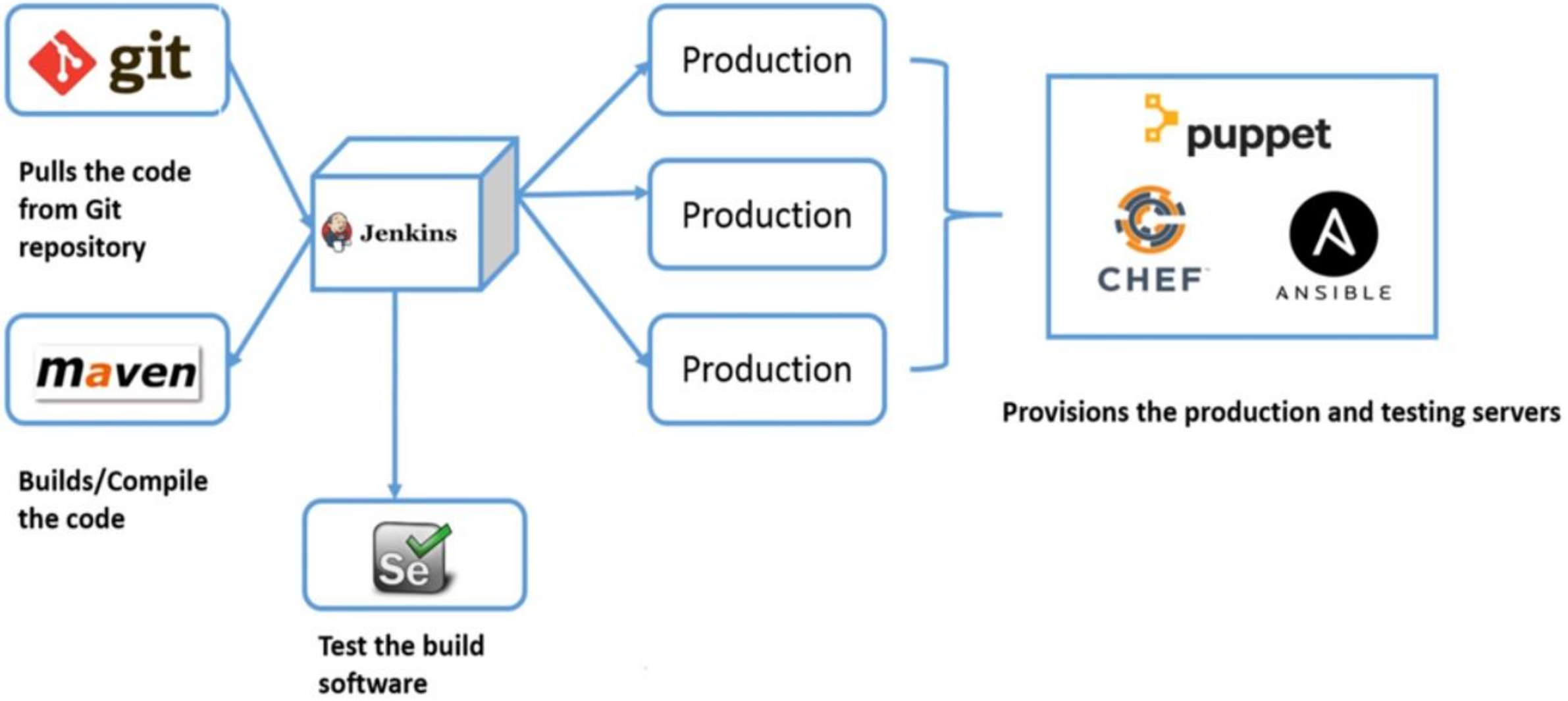
Continuous feedback

- DevOps teams should evaluate each release and generate reports to improve future releases.
- By gathering continuous feedback, teams can improve their processes and incorporate customer feedback to improve the next release.

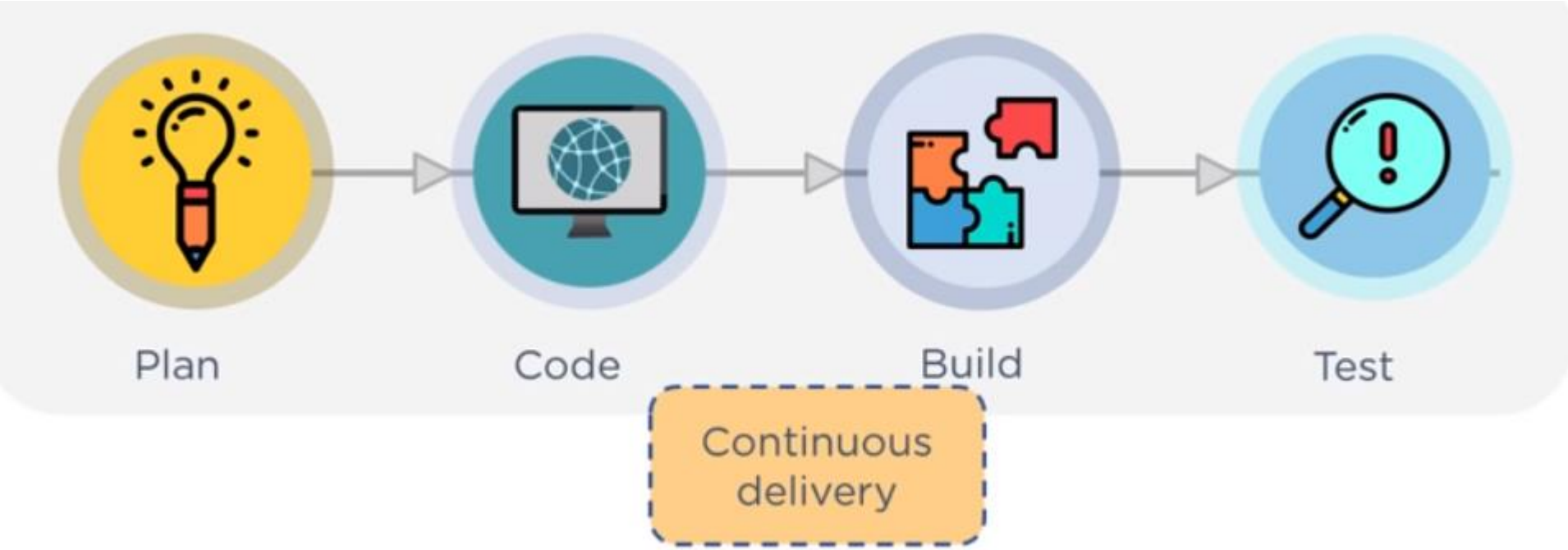
DevOps Tools

- **Git**
  - Version Control System
- **Maven**
  - Build and Dependnecy Management Tool
- **Jenkins**
  - Continuous Integration Tool
- **Ansible**
  - Configuration Management Tool
- **Docker**
  - Containerization Tool
- **Kubernetes**
  - Container Orchestration Tool
- **Nagios**
  - System Monitoring Tool
- **Terraform**
  - Infrastructure Management Tool

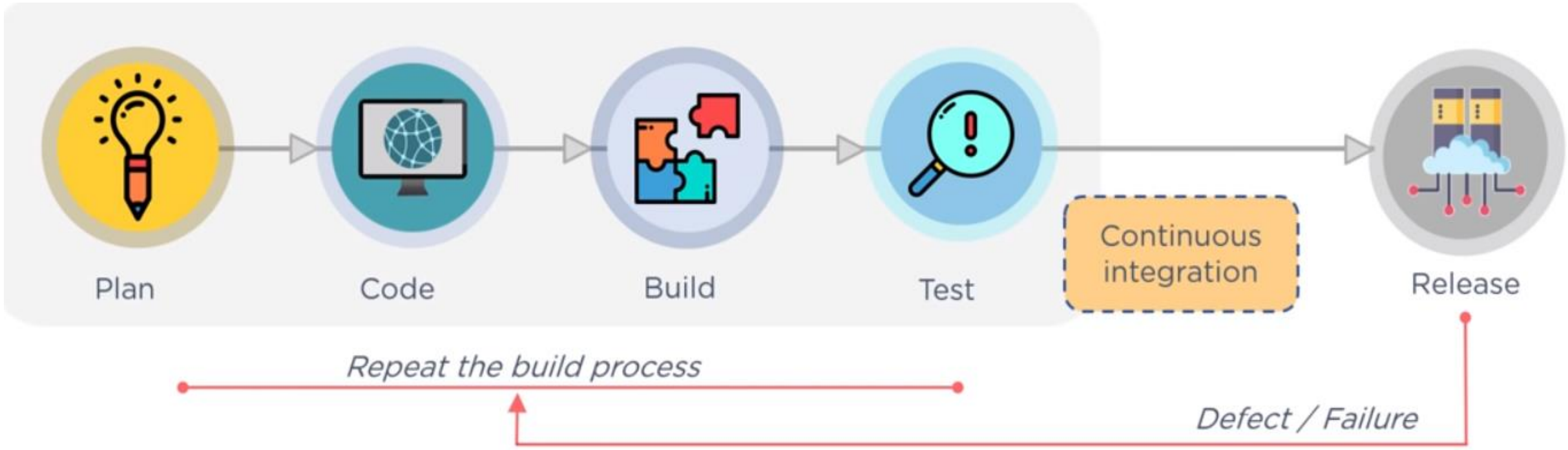
Implementation of DevOps - Example



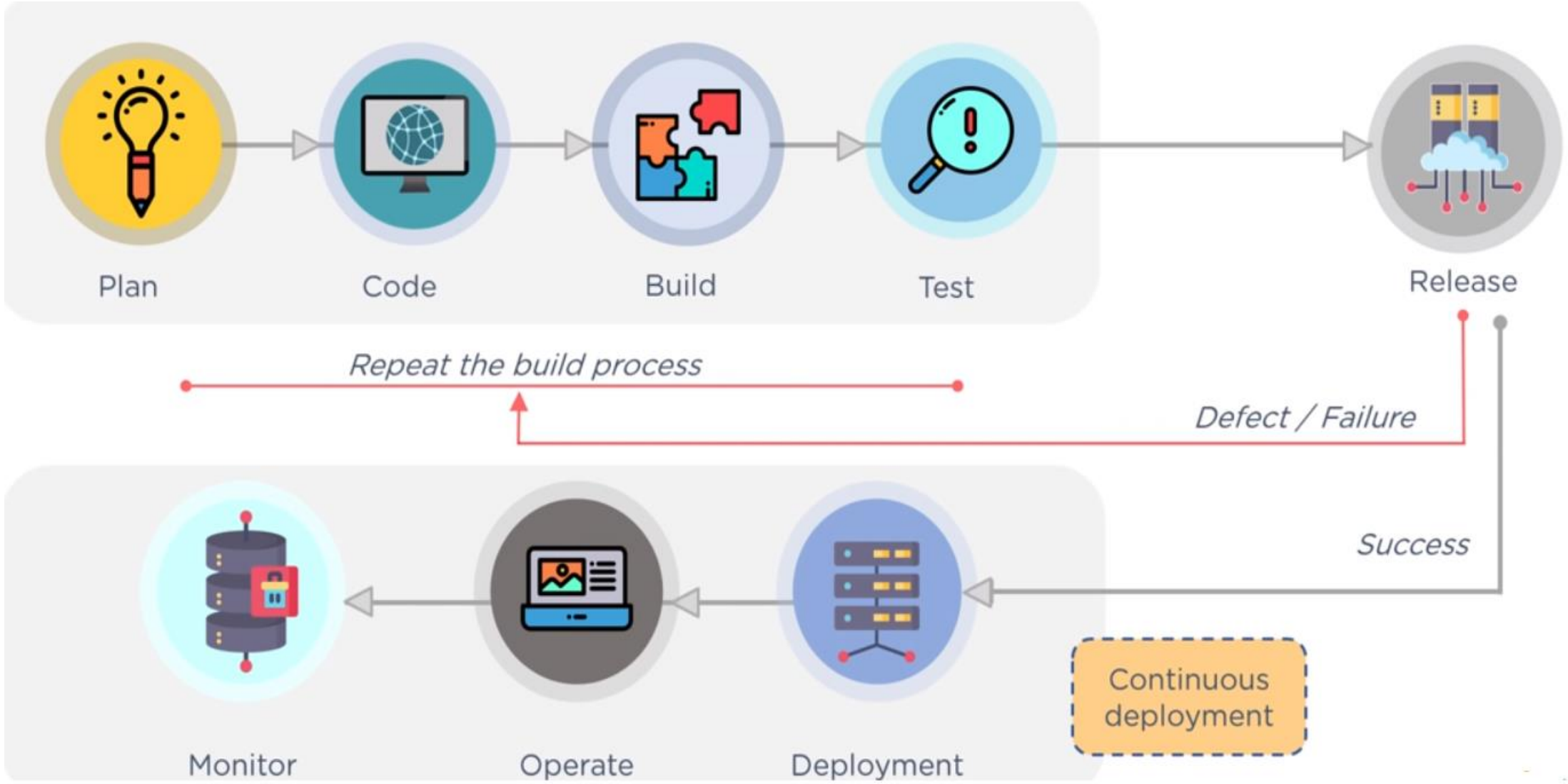
Continuous Delivery



Continuous Integration



Continuous Deployment



DevOps Used by Organizations

Companies which follow DevOps, release more products and features within a short amount of time

DevOps Advantages

Time taken to create and deliver software is reduced

Complexity of maintaining an application is reduced

Improved collaboration between developers and operations team

Continuous integration and delivery ensure faster time to market

Challenges of adopting DevOps

- Habits are hard to break.
- Teams entrenched in siloed ways of working can struggle with, or even be resistant to, overhauling team structures to embrace DevOps practices. Some teams may mistakenly believe new tools are sufficient to adopt DevOps.
- Yet, DevOps is a combination of people, tools, and culture. Everyone on a DevOps team must understand the entire value stream — from ideation, to development, to the end user experience.
- It requires breaking down silos in order to collaborate throughout the product lifecycle.

  Devops isn't any single person's job. It's everyone's job.
- Moving from a legacy infrastructure to using Infrastructure as Code (IaC) and microservices can offer faster development and innovation, but the increased operational workload can be challenging.
- It’s best to build out a strong foundation of automation, configuration management, and continuous delivery practices to help ease the load.

Challenges of adopting DevOps

- An over-reliance on tools can detract teams from the necessary foundations of DevOps: the team and organization structure. Once a structure is established, the processes and team should come next and the tools should follow.

DevOps Practices

- Continuous Development
- Continuous Integration
- Continuous Deployment
- Continuous Testing
- Continuous Monitoring

Continuous Development

- Developers uses various tools to develop the code. Developed code get push to source code management.

- DevOps Engineers does not write application code.
- DevOps engineers are responsible to maintain the code using “Source code management tools”

DevOps Practices

- **Continuous Integration**
  - Multiple Steps are involved in Continuous Integration
    - Unit Testing – Junit, Nunit (Plugin)
    - Code Build – Maven, ANT, Gradle (Plugin & Setup)

- **Continuous Deployment**
  - **Infrastructure** - Continuous deployment need target test/QA/prod etc environments. (Docker, Kubernetes, AWS etc)
  - Configuration Management - Ansible
- **Situational awareness**
- **Automation**
- **Infrastructure as Code**
- **Microservices**
- **Monitoring**

Q & A