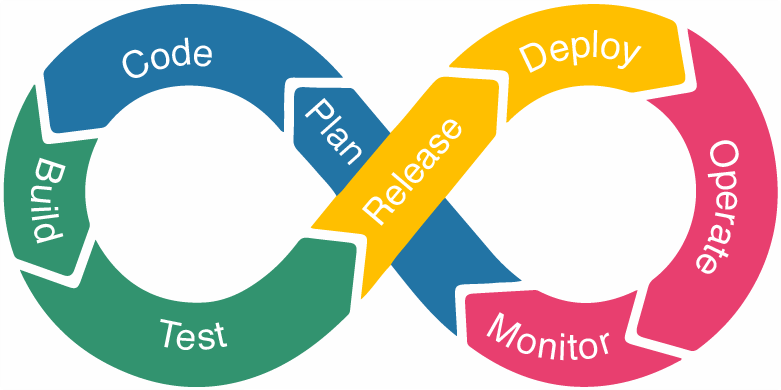
**DevOps lifecycle**

This section will delve into the different phases of the DevOps lifecycle in some detail.



* **Plan** – The planning phase is exactly what it sounds like: planning the project’s lifecycle. In contrast to conventional methods to the development lifecycle, this model assumes that each stage will be repeated as necessary. In this manner, the DevOps workflow is planned with the likelihood of future iterations and likely prior versions in mind.

This implies that we will likely have information from past iterations that will better inform the next iteration, and that the present iteration will likewise inform the next iteration. This stage often involves all teams to ensure that no area of the planning is ignored or forgotten.

* **Code** – The developers will write the code and prepare it for the next phase during the coding stage. Developers will write code in accordance with the specifications outlined in the planning phase and will ensure that the code is created with the project’s operations in mind.
* **Build**– Code will be introduced to the project during the construction phase, and if necessary, the project will be rebuilt to accommodate the new code. This can be accomplished in a variety of ways, although GitHub or a comparable version control site is frequently used.

The developer will request the addition of the code, which will then be reviewed as necessary. The request will be approved if the code is ready to be uploaded, and the code will be added to the project. Even when adding new features and addressing bugs, this method is effective.

* **Test**– Throughout the testing phase, teams will do any necessary testing to ensure the project performs as planned. Teams will also test for edge and corner case issues at this stage. An “edge case” is a bug or issue that only manifests during an extreme operating event, whereas a “corner case” occurs when many circumstances are met.
* **Release** – The release phase occurs when the code has been verified as ready for deployment and a last check for production readiness has been performed. The project will subsequently enter the deployment phase if it satisfies all requirements and has been thoroughly inspected for bugs and other problems.
* **Deploy**– In the deploy phase, the project is prepared for the production environment and is operating as planned in that environment. This would be the responsibility of the operations team; in DevOps, it is a shared responsibility. This shared duty pushes team members to collaborate to guarantee a successful deployment.
* **Operate**– In the operating phase, teams test the project in a production environment, and end users utilise the product. This crucial stage is by no means the final step. Rather, it informs future development cycles and manages the configuration of the production environment and the implementation of any runtime requirements.
* **Monitor**– During the monitoring phase, product usage, as well as any feedback, issues, or possibilities for improvement, are recognized and documented. This information is then conveyed to the subsequent iteration to aid in the development process. This phase is essential for planning the next iteration and streamlines the pipeline’s development process.

**7 Cs of DevOps**

The different phases of the DevOps Lifecycle can be broken down into 7 C’s for more practical clarity.

A picture containing text, screenshot, font, diagram

Description automatically generated

**1. Continuous Development** – This step is crucial in defining the vision for the entire [software development process](https://www.browserstack.com/guide/learn-software-development-process). It focuses mostly on project planning and coding. At this phase, stakeholders and project needs are gathered and discussed. In addition, the product backlog is maintained based on customer feedback and is divided down into smaller releases and milestones to facilitate continuous software development.

Once the team reaches consensus on the business requirements, the development team begins coding to meet those objectives. It is an ongoing procedure in which developers are obliged to code whenever there are modifications to the project requirements or performance difficulties.

**2. Continuous Integration** – [Continuous integration](https://www.browserstack.com/guide/continuous-integration-with-agile) is the most important stage of the DevOps lifecycle. At this phase, updated code or new functionality and features are developed and incorporated into the existing code. In addition, defects are spotted and recognized in the code at each level of unit testing during this phase, and the source code is updated accordingly. This stage transforms integration into a continuous process in which code is tested before each commit. In addition, the necessary tests are planned during this period.

**3. Continuous Testing** – Some teams conduct the [continuous testing](https://www.browserstack.com/guide/continuous-testing-in-devops) phase prior to integration, whereas others conduct it after integration. Using Docker containers, quality analysts regularly test the software for defects and issues during this phase. In the event of a bug or error, the code is returned to the integration phase for correction. Moreover, automation testing minimises the time and effort required to get reliable findings. During this stage, teams use technologies like as [Selenium](https://www.browserstack.com/selenium). In addition, continuous testing improves the test assessment report and reduces the cost of delivering and maintaining test environments.

**4. Continuous Deployment** – This is the most important and active step of the DevOps lifecycle, during which the finished code is released to production servers. Continuous deployment involves configuration management to ensure the proper and smooth deployment of code on servers. Throughout the production phase, development teams deliver code to servers and schedule upgrades for servers, maintaining consistent configurations.

In addition to facilitating deployment, containerization tools ensure consistency throughout the development, testing, production, and staging environments. This methodology enabled the constant release of new features in production.

**5. Continuous Feedback** – Constant feedback was implemented to assess and enhance the application’s source code. During this phase, client behaviour is routinely examined for each release in an effort to enhance future releases and deployments. Companies can collect feedback using either a structured or unstructured strategy.

Under the structural method, input is gathered using questionnaires and surveys. In contrast, feedback is received in an unstructured manner via social media platforms. This phase is critical for making continuous delivery possible in order to release a better version of the program.

**6. Continuous Monitoring** – During this phase, the functioning and features of the application are regularly monitored to detect system faults such as low memory or a non-reachable server. This procedure enables the IT staff to swiftly detect app performance issues and their underlying causes. Whenever IT teams discover a serious issue, the application goes through the complete DevOps cycle again to determine a solution. During this phase, however, security vulnerabilities can be recognized and corrected automatically.

**7. Continuous Operations** – The final phase of the DevOps lifecycle is essential for minimising scheduled maintenance and other planned downtime. Typically, developers are forced to take the server offline in order to perform updates, which increases the downtime and could cost the organisation a large amount of money. Eventually, continuous operation automates the app’s startup and subsequent upgrades. It eliminates downtime using container management platforms such as Kubernetes and Docker.