Topic: Introduction to DevOps

Topics:

- Introduction to DevOps
- Need of DevOps
- Evolution & History of DevOps
- Methodologies, Principles and Strategies
- Lifecycle of DevOps
- Phases of DevOps

Introduction to DevOps

- DevOps is basically a combination of two words- **Dev**elopment and **Operations**.
- DevOps is a culture that implements the technology in order to promote collaboration between the developer team and the operations team to deploy code to production faster in an automated and repeatable way.
- DevOps helps to increase organization speed to deliver applications and services. It also allows organizations to serve their customers better and compete more strongly in the market.
- DevOps can also be defined as a sequence of development and IT operations with better communication and collaboration.

…contd…

- DevOps has become one of the most valuable business disciplines for enterprises or organizations. With the help of DevOps, **quality**, and **speed** of the application delivery has improved to a great extent.
- DevOps is nothing but a practice or methodology of making "**Developers**" and "**Operations**" folks work together. DevOps represents a change in the IT culture with a complete focus on rapid IT service delivery through the adoption of agile practices in the context of a system-oriented approach.
- DevOps is all about the integration of the operations and development process. Organizations that have adopted DevOps noticed a 22% improvement in software quality and a 17% improvement in application deployment frequency and achieve a 22% hike in customer satisfaction. 19% of revenue hikes as a result of the successful DevOps implementation

Need of DevOps.

Before going further, we need to understand why we need the DevOps over the other methods.

- The operation and development team worked in complete isolation.
- After the design-build, the testing and deployment are performed respectively. That's why they consumed more time than actual build cycles.
- Without the use of DevOps, the team members are spending a large amount of time on designing, testing, and deploying instead of building the project.
- Manual code deployment leads to human errors in production.
- Coding and operation teams have their separate timelines and are not in synch, causing further delays.

Evolution & History of DevOps

- Originally coined by Patrick Debois, DevOps has become a critical discipline for realizing the benefits of Agile that ensures that rapid, iterative code development results in rapid, iterative code deployment!
- While Agile's gained popularity, often at cross-purposes with the more formal and "heavy" ITIL methods popular with IT in the early and mid-2000s, DevOps resonated with both sides.
- Organizations that have adopted ITIL can also implement DevOps, especially for Cloud-based applications.

- Starting with The Phoenix Project, by Gene Kim, DevOps has steadily gained popularity and supporters, and is seen today as a crucial element of any Agile technology organization. Consequently, large corporations and all sorts of technology vendors now support DevOps. DevOps jobs have become ever more popular – and the norm – in hitech organizations.
- With the emergence of AI and ML in all aspects of the software lifecycle, AI for DevOps is starting to make DevOps even more smart, fast and seamless – tho' a lot remains to be done.

..contd..

- n 2009, the first conference named **DevOpsdays** was held in Ghent Belgium. Belgian consultant and Patrick Debois founded the conference.
- In 2012, the state of DevOps report was launched and conceived by Alanna Brown at Puppet.
- In 2014, the annual State of DevOps report was published by Nicole Forsgren, Jez Humble, Gene Kim, and others. They found DevOps adoption was accelerating in 2014 also.
- In 2015, Nicole Forsgren, Gene Kim, and Jez Humble founded DORA (DevOps Research and Assignment).
- In 2017, Nicole Forsgren, Gene Kim, and Jez Humble published "Accelerate: Building and Scaling High Performing Technology Organizations".

Methodologies, Principles and Strategies

- While DevOps is seen as a natural extension of Agile, and somewhat of an anathema for ITIL, it does not have its own framework – and can potentially be relevant for a variety of situations.
- Organizations that use traditional (waterfall) methods, characterized by sequential stages of software development and long gaps between software releases, can use DevOps principles for better alignment between functions such as Dev, QA, and Operations, with a greater transparency into all functions. Organizations that have adopted one or more Agile methodologies can easily have their dev and operations folks collaborate throughout the development process.
- Newer disciplines such as AIOps, SRE (Site Reliability Engineering), SysOps, DevSecOps and BizDevOps extend and expand DevOps principles and benefits while adding other critical technology and methodologies such as AI, automation, security and collaboration to further the cause of high quality agile software development.

..contd..

One should understand the related contexts of DevOps, Agile and Waterfall development, site reliability engineering (SRE) and SysOps

- **DevOps vs. Waterfall development.** Waterfall development comprises a series of steps and gates in a linear progression to production. Its phases are requirements, analysis, design, coding and implementation, testing, operation and deployment, and maintenance. In Waterfall teams, development tests new code in an isolated environment for quality assurance (QA) and -- if requirements are met -- releases the code to operations for use in production. IT operations deploys multiple releases at once, with extensive controls. Support is operations' responsibility. Waterfall approaches engender long waits between software releases. Because development and operations teams work separately, developers are not always aware of operational roadblocks that prevent code from working as anticipated.

The DevOps model aligns development, QA and IT operations efforts with fewer gates and more continuous workflow. For example, some of the operations' team responsibilities shift left in the app delivery pipeline to the development team. IT operations provides feedback for code improvements. Rather than gated steps, DevOps relies on continuous development, continuous integration, continuous delivery and continuous monitoring processes.

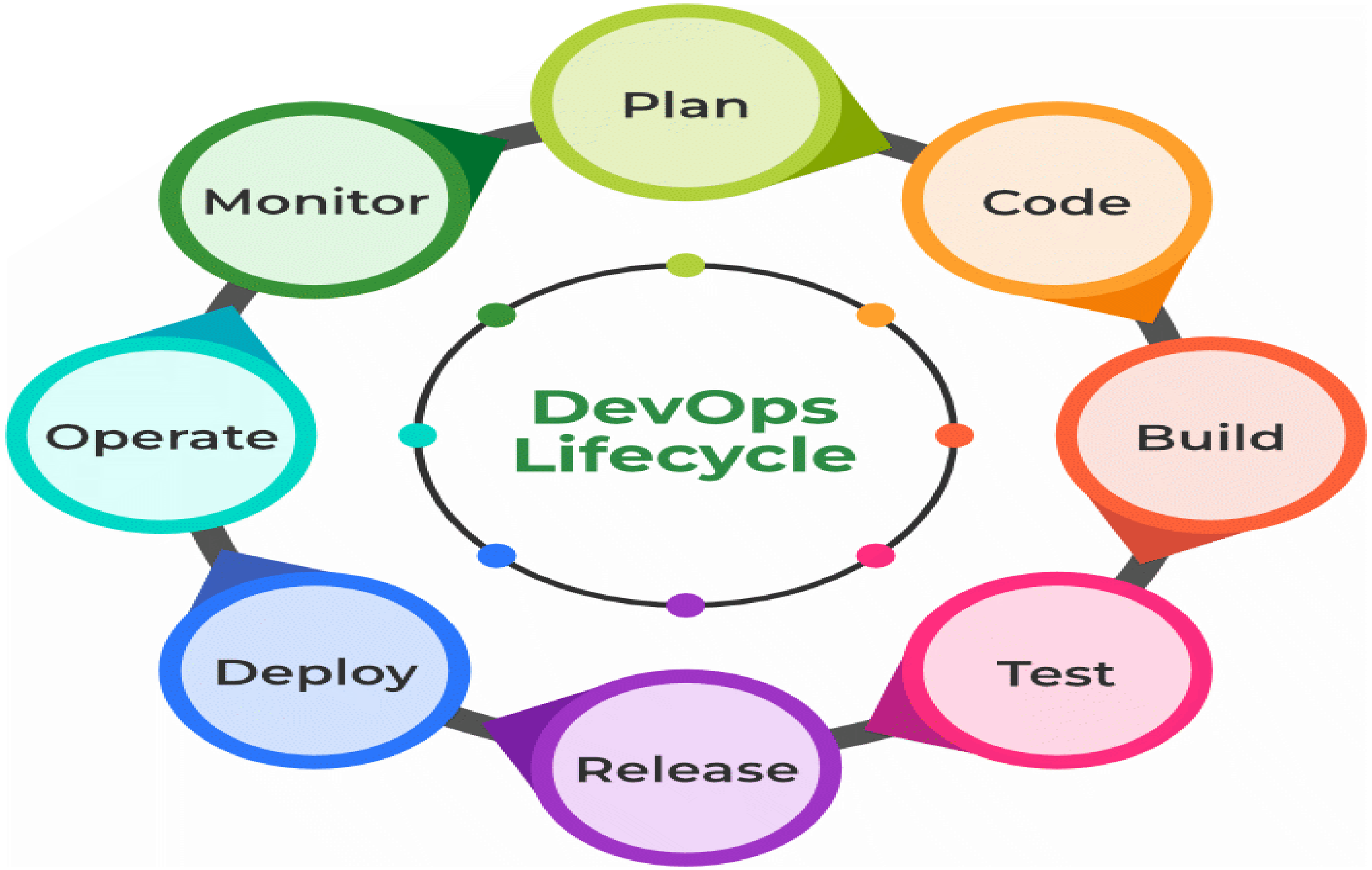
- **DevOps vs. Agile development.** Agile is a software development approach defined in the Agile Manifesto. Agile teams focus on incremental and rapid cycles of code creation and delivery, referred to as *sprints*. Each sprint iterates upon the last, which makes the software highly flexible and adaptable to changing requirements. It is possible for the original vision of a project to be lost through this cycle.
- DevOps arose from Agile's success at improving development speed, and the realization that disconnects between development and operations teams -- as well as between IT and the business side of the organization -- significantly hindered the Agile software's delivery to users.
- In an Agile-only workflow, development and operations teams have separate objectives and leadership. When an organization uses DevOps and Agile together, both development and operations teams manage code throughout the software development lifecycle. While Agile work is often formalized with a framework, such as Scrum, DevOps does not have a framework.

- **DevOps vs. SRE.** Site reliability engineering arose concurrently with Agile and DevOps. Started in the early 2000s at Google, it is essentially a programming- and automation-focused approach to the software development lifecycle. Problems should be solved in a way that prevents them from occurring again. Rote tasks should be minimized.
- The SRE toolbox closely matches that for DevOps. Both disciplines aim for continuous improvement. SRE and DevOps engineers seek to abolish silos between development and operations. While DevOps also can extend to business stakeholders, SRE typically stays within the confines of IT processes.

- **DevOps vs. SysOps.** SysOps typically denotes that an IT administrator or IT team manages production deployment and support for a large distributed application, such as a SaaS product. As with DevOps adopters, SysOps teams should be versed in cloud computing and automation, as well as other technologies that enable applications to perform well at a large scale. SysOps teams troubleshoot IT outages and incidents, monitor for performance problems, enforce security rules and optimize operations.
- They also focus on high availability, fault tolerance, security and performance just like other IT admins. While SysOps professionals are likely to use some development tools and understand development processes, their work is not as enmeshed with development as in a DevOps job. However, SysOps roles can exist within DevOps and SRE organizations.

DevOps Lifecycle : DevOps lifecycle is the methodology where professional development teams come together to bring products to market more efficiently and quickly.

The structure of the DevOps lifecycle consists of Plan, Code, Building, Test, Releasing, Deploying, Operating, and Monitoring.



DevOps Lifecycle in Detail.

- **Plan:** Determining the commercial needs and gathering the opinions of end-user by professionals in this level of the DevOps lifecycle.
- **Code:** At this level, the code for the same is developed and in order to simplify the design, the team of developers uses tools and extensions that take care of security problems.
- **Build:** After the coding part, programmers use various tools for the submission of the code to the common code source.
- **Test:** This level is very important to assure software integrity. Various sorts of tests are done such as user acceptability testing, safety testing, speed testing, and many more.
- **Release:** At this level, everything is ready to be deployed in the operational environment.
- **Deploy:** In this level, Infrastructure-as-Code assists in creating the operational infrastructure and subsequently publishes the build using various DevOps lifecycle tools.
- **Operate:** At this level, the available version is ready for users to use. Here, the department looks after the server configuration and deployment.
- **Monitor:** The observation is done at this level that depends on the data which is gathered from consumer behavior, the efficiency of applications, and from various other sources.

Important Links

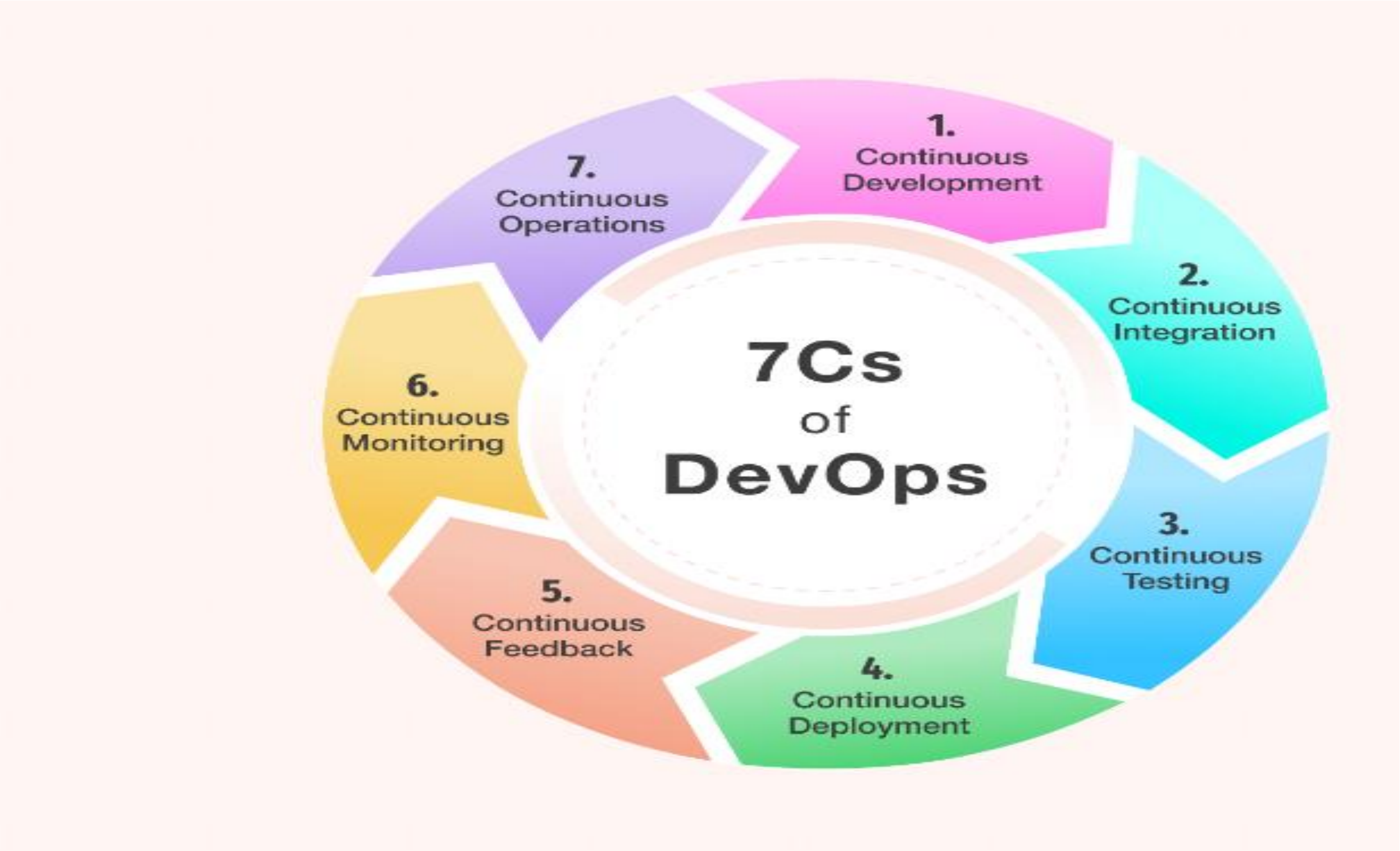
- https://www.youtube.com/watch?v=to5H5iPZmD4

- https://www.youtube.com/watch?v=dJAxKm8La88

DevOps lifecycle phases: the 7Cs of DevOps lifecycle

- As we have discussed in Lifecycle that everything is continuous in DevOps – from **planning to monitoring**.
- So let's break down the entire **lifecycle into seven phases** where continuity is at its core.
- Any phase in the lifecycle can iterate throughout the projects multiple times until it's finished.

7 C’s of DevOps



1. Continuous Development

This phase plays a pivotal role in delineating the vision for the entire software development cycle.

It primarily focuses on project planning and coding.
During this phase, project requirements are gathered and discussed with stakeholders.
Moreover, the product backlog is also maintained based on customer feedback which is broken down into smaller releases and milestones for continuous software development.

Once the team agrees upon the business needs, the development team starts coding for the desired requirements.

It’s a continuous process where developers are required to code whenever any changes occur in the project requirement or in case of any performance issues.

2. Continuous integration

- Continuous integration is the most crucial phase in the entire DevOps lifecycle. In this phase, **updated code** or add-on functionalities and features are developed and integrated into existing code. Furthermore, bugs are detected and identified in the code during this phase at every step through **unit testing**, and then the source code is modified accordingly. This step makes integration a continuous approach where code is tested at **every commit**. Moreover, the tests needed are also planned in this phase.

CASE STUDY

- Let’s take the example of Docusign, which developed e-signature technology back in 2003.
- It helps its clients automate the process of preparing, signing, and managing agreements.
- Their development teams used to follow Agile methodology for years to collect customer feedback and make small and quick releases.
- But, they lacked collaboration between the development and operations team, which led them to many failures.
- Moreover, their business was solely based on the transaction of signatures and approvals.
- So, the biggest challenge for their business was continuous integration and delivery.
- A single mistake could cause a serious problem and ruin the entire operation process. Hence, the organization decided to move to DevOps. DocuSign implemented a tool – mock for their internal API to speed up the product development and delivery.

- This tool helped the organization in integrating critical functionalities such as incident management. This tool also makes the testing with simulation simple.

Tools Used in Continuous integration

- Jenkin, Bamboo, GitLab CI, Buddy, TeamCity, Travis, and CircleCI are a few DevOps tools used to make the project workflow smooth and more productive.

- For example, Jenkin (open-source tool) is used widely to automate builds and tests.
- CircleCI and Buddy, on the other hand, are commercial tools.

Well, whatever tools you select for continuous integration, pick the one that can fit your business and project requirements.

Continuous Testing

- Some teams carry out the continuous testing phase before the integration occurs, while others do it after the integration.
- Quality analysts continuously test the software for bugs and issues during this stage using **Docker containers**.
- In case of a bug or an error, the code is sent back to the integration phase for modification.
- Automation testing also reduces the time and effort to deliver quality results.
- Teams use tools like ***Selenium*** at this stage. Moreover, continuous testing enhances the test evaluation report and minimizes the provisioning and maintenance cost of the test environments

Tools Used in Continuous Testing

- JUnit, Selenium, TestNG, and TestSigma are a few DevOps tools for continuous testing.

- **Selenium** is the most popular open-source automation testing tool that supports multiple platforms and browsers.

- **TestSigma**, on the other hand, is a unified AI-driven test automation platform that eliminates the technical complexity of test automation through **artificial intelligence**.

Continuous Deployment

- This phase is the crucial and most active one in the DevOps lifecycle, where final code is deployed on production servers.
- The continuous deployment includes configuration management to make the deployment of code on servers accurate and smooth.
- Development teams release the code to servers and schedule the updates for servers, keeping the configurations consistent throughout the production process.
- Containerization tools also help in the deployment process by providing consistency across development, testing, production, and **staging environments**.
- This practice made the continuous delivery of new features in production possible.

Tools Used

- Ansible, Puppet, and Chef are the configuration management tools that make the deployment process smooth and consistent throughout the production process.
- Docker and Vagrant are another DevOps tool used widely for handling the scalability of the continuous deployment process.
- Apart from this, Spinnaker is an open-source continuous delivery platform for releasing the software changes, while ArgoCD is another open-source tool for Kubernetes native CI/CD.

Continuous Feedback

Continuous feedback came into existence to analyze and improve the application code.

During this phase, customer behavior is evaluated regularly on each release to improve future releases and deployments.

Businesses can either opt for a structural or unstructured approach to gather feedback.

In the structural approach, feedback is collected through surveys and questionnaires.

In contrast, the feedback is received through social media platforms in an unstructured approach.

Overall, this phase is quintessential in making continuous delivery possible to introduce a **better version** of the application.

CASE STUDY

- One of the examples of continuous feedback is Tangerine bank.

It’s a Canadian bank that embraced continuous feedback to enhance its customers’ mobile experience.

After opting for continuous feedback, this Canadian bank collected a considerable amount of valuable feedback within a few weeks, which helped the bank reach the cause of the problem quickly.

Furthermore, this has helped them improve the application as per their customers’ needs.

This is how Tangerine bank managed to repurpose the resources and money on other crucial things excellently after adopting DevOps.

Tools Used:

Pendo is a product analytics tool used to collect customer reviews and insights.

Qentelli's TED is another tool used primarily for tracking the entire DevOps process to gather actionable insights for bugs and flaws.

Continuous Monitoring

- During this phase, the application's functionality and features are monitored continuously to detect **system errors** such as low memory, non-reachable server, etc.

- This process helps the IT team quickly **identify issues** related to app performance and the **root cause** behind it.
- If IT teams find any critical issue, the application goes through the entire DevOps cycle again to find the solution.
- However, the security issues can be detected and resolved automatically during this phase.

Tools Used:

- Nagios,
- Kibana,
- Splunk,
- PagerDuty,
- ELK Stack,
- New Relic, and
- Sensu

These are a few DevOps tools used to make the continuous monitoring process fast and straightforward.

Continuous Operations

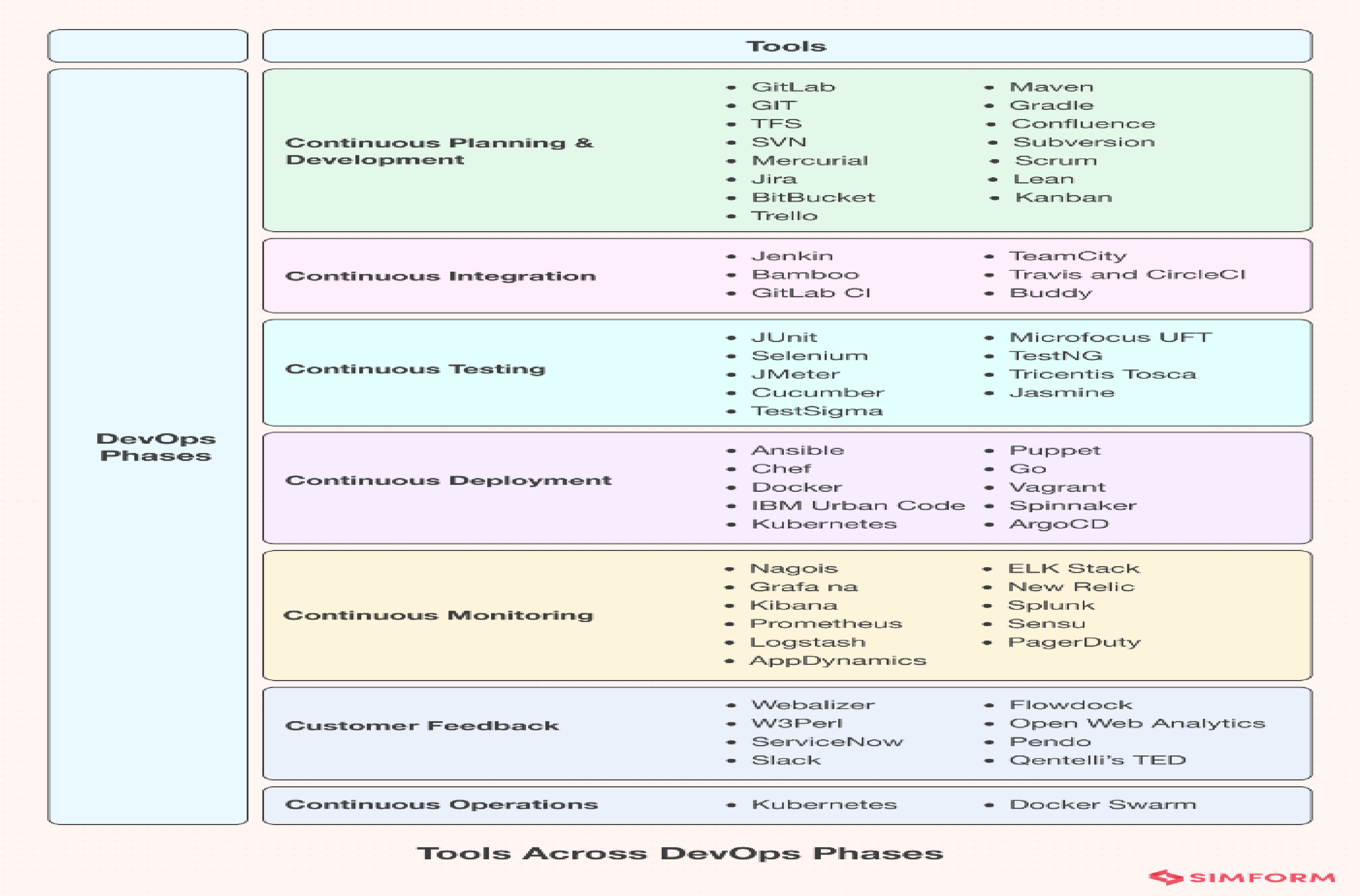
- The last phase in the DevOps lifecycle is crucial for reducing the planned downtime, such as **scheduled maintenance**.

- Generally, developers are required to take the server offline to make the updates, which increases the downtime and might even cost a significant loss to the company.
- Eventually, continuous operation automates the process of launching the app and its updates.
- It uses container management systems like Kubernetes and Docker to eliminate downtime.
- These container management tools help simplify the process of building, testing, and deploying the application on multiple environments. The key objective of this phase is to boost the application's uptime to ensure uninterrupted services. Through continuous operations, developers **save time** that can be used to accelerate the application's time-to-market.

Tools Used:

- **Kubernetes and Docker Swarm** are the container orchestration tools used for the high availability of the application and to make the deployment faster.

Overall Summary with Tools



	Tools		
DevOps Phases	Continuous Planning & Development	• GitLab • GIT • TFS • SVN • Mercurial • Jira • BitBucket • Trello	• Maven • Gradle • Confluence • Subversion • Scrum • Lean • Kanban
	Continuous Integration	• Jenkin • Bamboo • GitLab CI	• TeamCity • Travis and CircleCI • Buddy
	Continuous Testing	• JUnit • Selenium • JMeter • Cucumber • TestSigma	• Microfocus UFT • TestNG • Tricentis Tosca • Jasmine
	Continuous Deployment	• Ansible • Chef • Docker • IBM Urban Code • Kubernetes	• Puppet • Go • Vagrant • Spinnaker • ArgoCD
	Continuous Monitoring	• Nagois • Grafa na • Kibana • Prometheus • Logstash • AppDynamics	• ELK Stack • New Relic • Splunk • Sensu • PagerDuty
	Customer Feedback	• Webalizer • W3Perl • ServiceNow • Slack	• Flowdock • Open Web Analytics • Pendo • Qentelli’s TED
	Continuous Operations	• Kubernetes	• Docker Swarm

Tools Across DevOps Phases

SIMFORM