**DEVOPS**

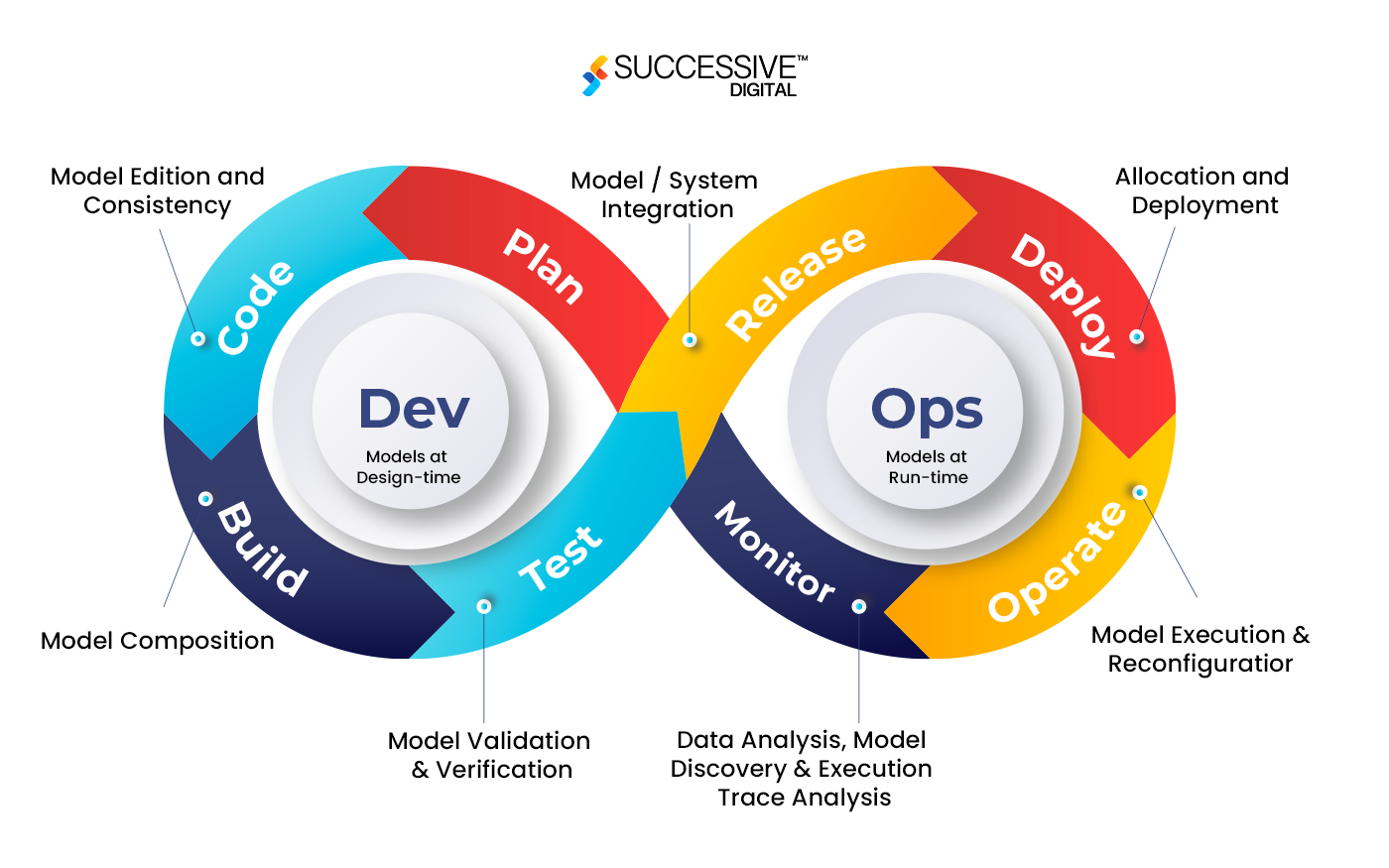
DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

**How DevOps Works**

Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps.

These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably. These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team’s velocity



## Benefits of DevOps

### **1.Speed**

Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables your developers and operations teams to achieve these results. For example, microservices and continuous delivery let teams take ownership of services and then release updates to them quicker.

### **2.Rapid Delivery**

Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers’ needs and build competitive advantage. Continuous integration and continuous delivery are practices that automate the software release process, from build to deploy.

### **3.Reliability**

Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users. Use practices like continuous integration and continuous delivery to test that each change is functional and safe. Monitoring and logging practices help you stay informed of performance in real-time.

### **4.Scale**

Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

### **5.Improved Collaboration**

Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).

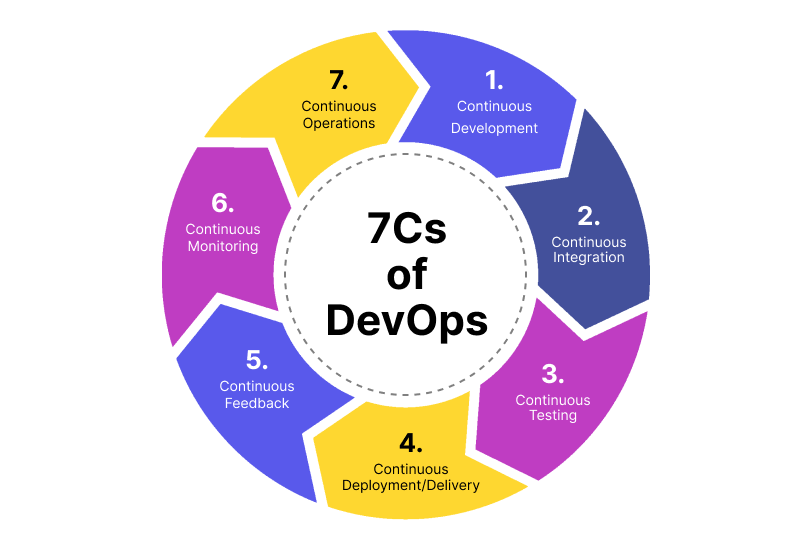
### **6.Security**

Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and policy as code, you can define and then track compliance at scale.

### **DevOps Lifecycle**

The **DevOps lifecycle** is a series of phases that involve collaboration between developers and operations teams, ensuring smooth delivery and operation of applications. The main phases in the DevOps lifecycle are:

1. **Planning**:
   * **Objective**: Define project goals, features, and specifications.
   * Developers and stakeholders define project requirements and discuss how to break down tasks, prioritize features, and schedule releases. This is often based on Agile methodologies.
2. **Development**:
   * **Objective**: Write code, develop features, and implement functionalities.
   * In this phase, developers build the application or feature according to the requirements defined in the planning phase. Version control systems (like Git) are used to manage code.
3. **Build**:
   * **Objective**: Compile the code and prepare it for testing and deployment.
   * Automated build tools compile the application and package it for further processing. Continuous Integration (CI) tools like Jenkins, GitLab CI, or CircleCI may be used to automatically trigger builds when code changes are committed.
4. **Testing**:
   * **Objective**: Ensure the code is working as expected and is free from bugs.
   * Automated testing tools (e.g., Selenium, JUnit) are used to test the application for bugs and defects. Tests may include unit tests, integration tests, and performance tests. This ensures that the code is functioning properly before it’s deployed to production.
5. **Release**:
   * **Objective**: Prepare the application for deployment.
   * In this phase, the code is released to production or staging environments. Continuous Delivery (CD) tools ensure the application is always in a deployable state. Manual or automated approval processes can occur before the release.
6. **Deploy**:
   * **Objective**: Deploy the code into production.
   * The application is deployed to a live environment, making it available to users. Deployment can be automated using tools like Kubernetes, Docker, Ansible, or AWS.
7. **Operate**:
   * **Objective**: Maintain and monitor the live application in production.
   * The system is actively monitored for performance, uptime, and user feedback. Operations teams ensure the system is running smoothly, addressing any issues such as downtime, latency, or security vulnerabilities.
8. **Monitor**:
   * **Objective**: Continuously monitor the application’s performance and user experience.
   * Real-time monitoring tools (like Prometheus, Grafana, or New Relic) track application metrics, logs, and system health. This information helps teams identify issues, improve performance, and better plan for future releases.
9. **Feedback**:
   * **Objective**: Collect feedback from users, stakeholders, and monitoring systems.
   * Feedback from users and monitoring systems informs the development team of potential improvements, bugs, or performance issues. This feedback is then used to plan the next iteration of development and refine the application.



**DevOps and Agile**

**Agile** and **DevOps** are often used together in modern software development practices, though they focus on different aspects of the development lifecycle. **Agile** focuses on iterative development, while **DevOps** focuses on improving collaboration, automation, and delivery.

* **Agile**: Focuses on breaking down the development process into small, manageable increments or sprints, enabling teams to respond to change, improve flexibility, and deliver features incrementally. The emphasis is on collaboration and delivering value to users continuously.
* **DevOps**: Extends Agile principles to include operational concerns, such as deployment, monitoring, and infrastructure management. While Agile emphasizes frequent and incremental delivery of features, DevOps ensures that the operational side (like deployment and monitoring) is equally agile, with fast feedback loops and automated delivery pipelines.

### **How DevOps and Agile Work Together:**

1. **Planning (Agile)**:
   * Both DevOps and Agile involve frequent planning sessions. Agile uses sprints or iterations, while DevOps emphasizes continuous planning, which integrates development and operations.
2. **Development (Agile)**:
   * Developers work in short, iterative cycles, delivering working software in small increments. DevOps automates build, test, and deployment processes to ensure that code changes are continuously integrated and deployed.
3. **Continuous Integration and Testing (DevOps)**:
   * Agile focuses on delivering value quickly through iterative cycles, and DevOps automates integration and testing to ensure that each iteration is tested and deployed quickly. This reduces manual intervention and improves feedback cycles.
4. **Release & Deployment (DevOps)**:
   * Agile's frequent release cycles are supported by DevOps' continuous delivery and deployment processes, ensuring that new features and bug fixes are deployed without delays.
5. **Monitoring & Feedback (DevOps)**:
   * While Agile promotes quick iterations and user feedback, DevOps ensures that monitoring and metrics are gathered from production systems to inform future development efforts. This creates a feedback loop for improvements in both the development and operations processes.