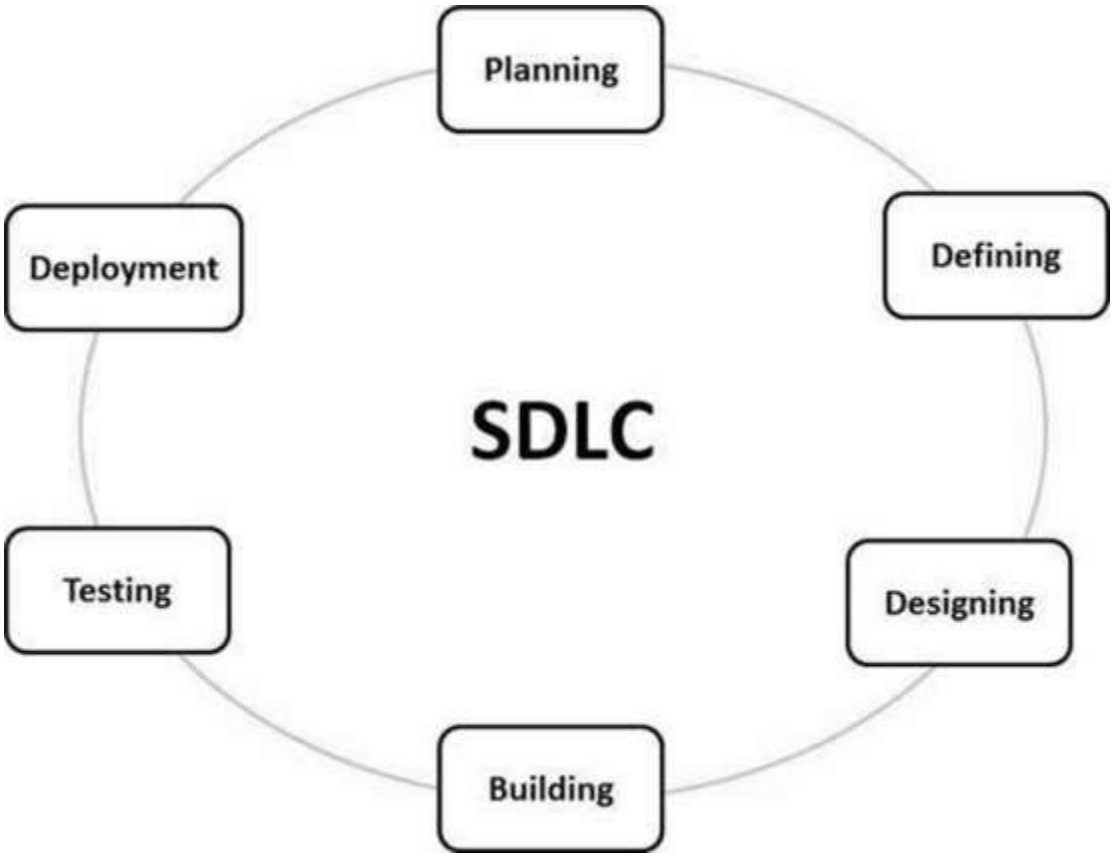
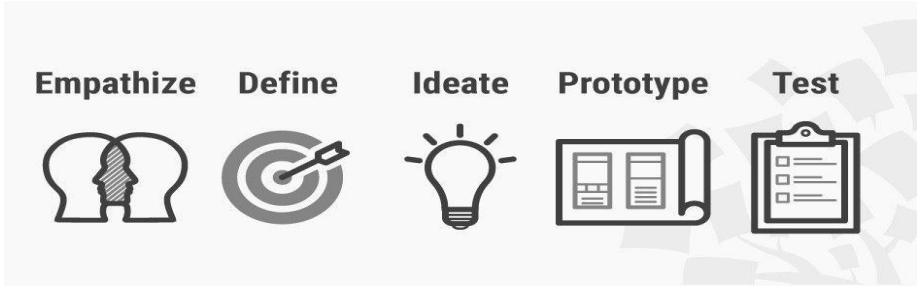
The Systems Development Life Cycle or SDLC

The Systems Development Life Cycle (SDLC) is a software engineering framework that is used to describe the various phases used to develop an information system. These phases include planning, analysis, design, development, testing, and implementation. SDLC environments describe the activities and tools required to perform a particular process within the SDLC. They are also defined as controlled points where software engineers can carry out activities related to development, testing, installation, and configuration. These environments are associated with the different phases that make up the SDLC.

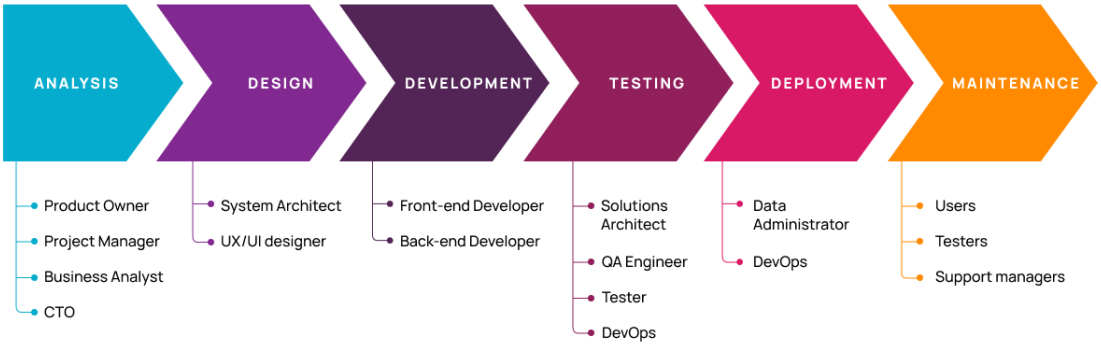


8 Phases of SDLC are: Plan, Code, Build, Test, Release, Deploy, Operate and Monitor.

Design Thinking: Design thinking is a non-linear, iterative process that teams use to understand users, challenge assumptions, redefine problems and create innovative solutions to prototype and test. Involving five phases—Empathize, Define, Ideate, Prototype and Test—it is most useful to tackle problems that are ill-defined or unknown.



6 Phases of the Software Development Life Cycle



Types of Environments to support end user requirement:

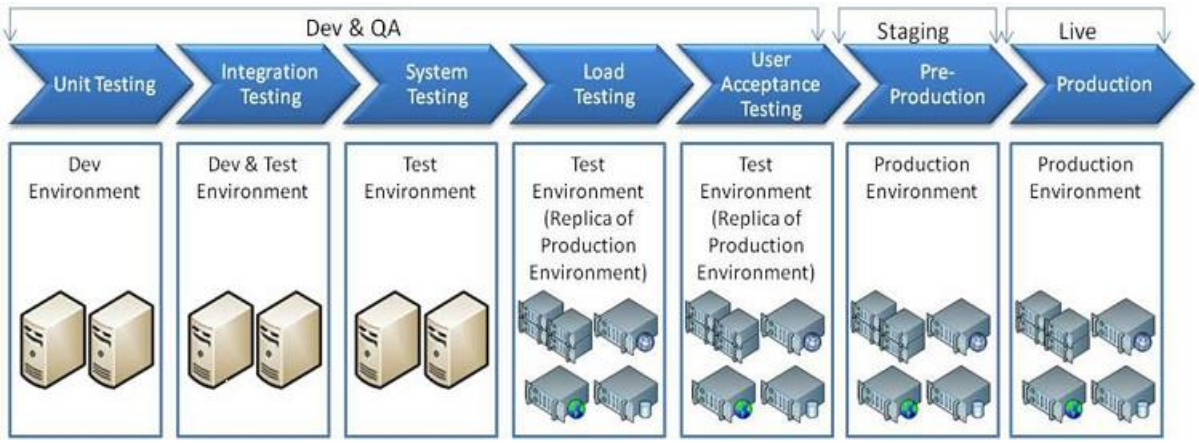
1. Development environment
2. Testing environment
3. Staging environment
4. Production environment

Development environment: The development environment is the first environment in software development which acts as the workspace for developers to do programming and other operations related to the creation of software and/or systems.
An integrated development environment (IDE) — a software package with extensive functions for authoring, building, testing, and debugging a program which is commonly used by software developers. Some programming software tools such as Microsoft Visual Studio, Eclipse, NetBeans, and other integrated development environments.

Testing environment: The test environment is where testing teams evaluate the application/quality. program's This also allows computer programmers to find out and solve any defects that may interfere with the application's smooth operation or degrade the user experience.
The test environment is created by allocating storage, computing, and other resources needed for testing. This could include new physical/virtual devices set up for testing use cases defined by developers. For example, Selenium tests cannot run for the whole set of browsers through which you want your application to be accessible at the same time. This means that you either run tests sequentially or generate multiple test environments.

Staging environment: When you generate the staging instance of an application, you are confident sufficient to reveal it to the immediate owner but not to users. You should run more tests before exposing to the latter group. The staging environment is similar to the pre-production in use.
The staging environment is frequently restricted to a small group of people. The only groups that can access the application in staging are those with whitelisted emails and IP addresses, as well as your developer team. The goal of a staging environment is to simulate production as much as possible.

Production environment: When the end-user uses a web/mobile application, the program is operating on a production server. It's been created in the production environment.
Tests can be carried out while the product is in production, and new features can be introduced safely at the same time. Feature flags allow you to show a future version of an app to a select few users while the rest continue to utilize the current version.



What is Devops?

- Devops is not a new tool/Technology in the market.
- It is a new culture or process to develop, release and maintain software
- products/projects/applications with high quality in very faster way.
- We can achieve this in Devops by using several automation tools.
- For any software development, release and maintenance, there are two groups of engineers will work in the company.
  1) Development Group
  2) Non-Development Group or Operations Group or Administrators Group.

Again, this classification can be divided into small sets of groups.

1)Development Group:

The people who are involving

1) planning 2) coding 3) build 4) Testing

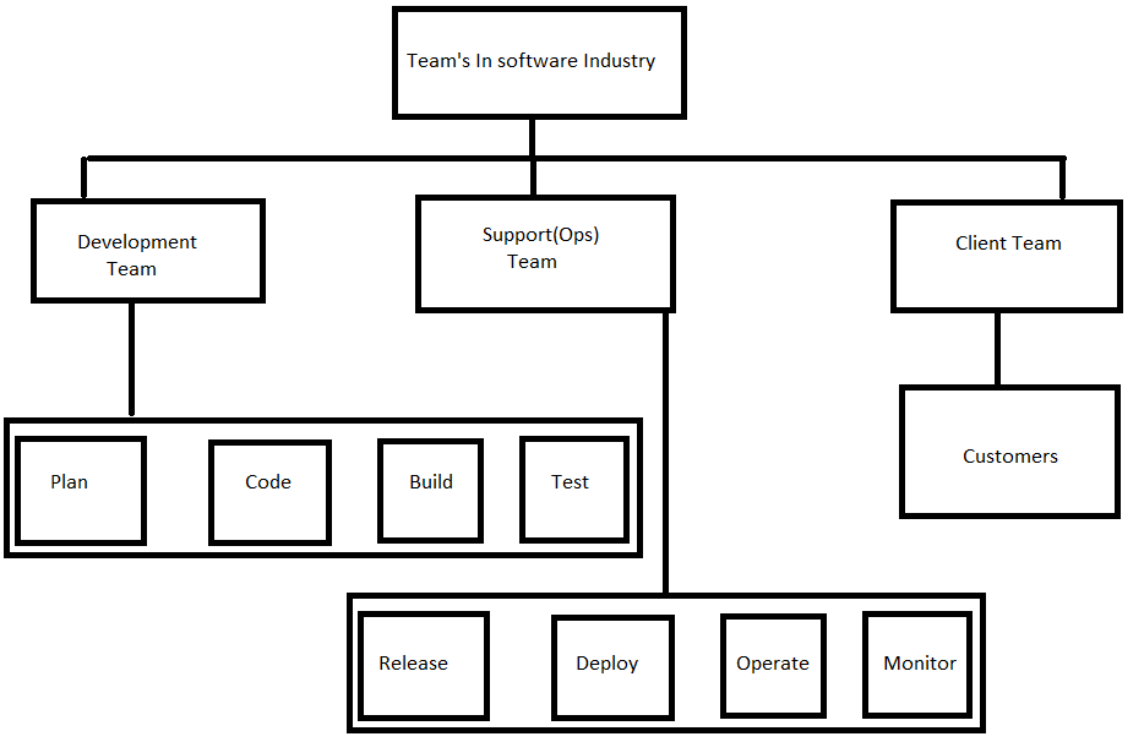
are considered as Development Group.

Eg: Business Analyst (BA), System Analyst (SA), Design Architect (DA), Developers/coders, Build Engineer and Test Engineers/QA

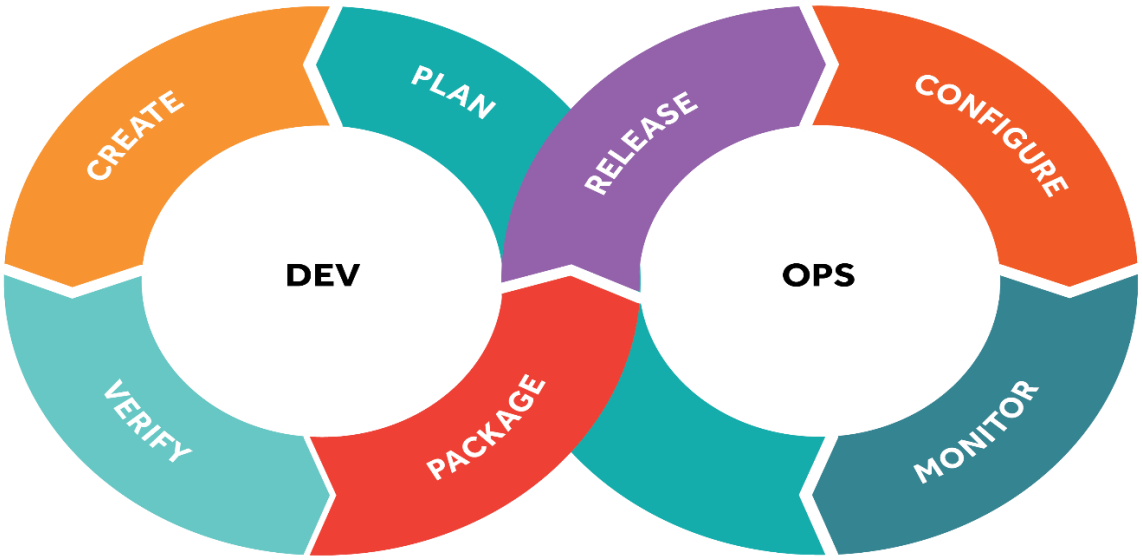
2)Operations Group:

The people who are involving
1) Release 2) Deploy 3) Operate 4) Monitor

are considered as Operations Group.
Eg: Release Engineers, Configuration Engineer, System Admin, Database Admin, Network Admin etc



DevOps is defined as a combination of processes and tools created to facilitate organizations in delivering services and applications much faster than conventional software development processes. DevOps, a combination of development and operations, is a complete collaboration of process, technology, and people, to provide continuous value to customers.



Devops Principles:

1. Treat Ops as first-class citizens from the point of view of requirements. Adding requirements to a system from Ops may require some architectural modification. In particular, the Ops requirements are likely to be in the area of logging, monitoring, and information to support incident handling.

2. Make Dev more responsible for relevant incident handling. By itself, this change is just a process change and should require no architectural modifications. However, just as with the previous category, once Dev becomes aware of the requirements for incident handling, some architectural modifications may result.

3. Enforce deployment process used by all, including Dev and Ops personnel. In general, when a process becomes enforced, some individuals may be required to change their normal operating procedures and, possibly, the structure of the systems on which they work. One point where a deployment process could be enforced is in the initiation phase of each system. Each system, when it is initialized, verifies its pedigree. That is, it arrived at execution through a series of steps, each of which can be checked to have occurred. Furthermore, the systems on which it depends (e.g., operating systems or middleware) also have verifiable pedigrees.

4. Use continuous deployment. Continuous deployment is the practice that leads to the most far-reaching architectural modifications. On the one hand, an organization can introduce continuous deployment practices with no major architectural changes.

5. Develop infrastructure code with the same set of practices as application code. These practices will not affect the application code but may affect the architecture of the infrastructure code.

Agile Principle(s)

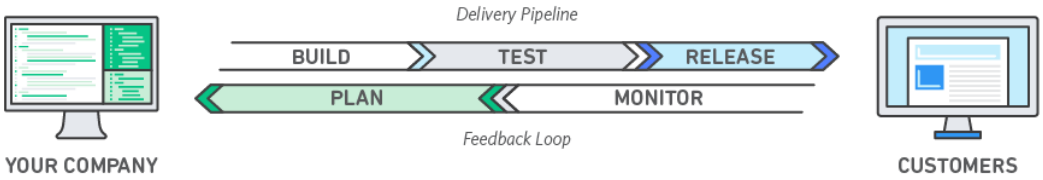
1) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2) Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4) Business people and developers must work together daily throughout the project.
5) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7) Working software is the primary measure of progress.
8) Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9) Continuous attention to technical excellence and good design enhances agility.
10) Simplicity–the art of maximizing the amount of work not done–is essential.
11) The best architectures, requirements, and designs emerge from self-organizing teams.
12) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Devops and SDLC relationship-> Devops= Continuous{SDLC}

Devops and Agile relationship-> Devops= Agile(OPS)

DevOps Model Defined

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

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.

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.

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.

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.

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).

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.

Why DevOps Matters

Software and the Internet have transformed the world and its industries, from shopping to entertainment to banking. Software no longer merely supports a business; rather it becomes an integral component of every part of a business. Companies interact with their customers through software delivered as online services or applications and on all sorts of devices. They also use software to increase operational efficiencies by transforming every part of the value chain, such as logistics, communications, and operations. In a similar way that physical goods companies transformed how they design, build, and deliver products using industrial automation throughout the 20th century, companies in today's world must transform how they build and deliver software.

How to Adopt a DevOps Model

Transitioning to DevOps requires a change in culture and mindset. At its simplest, DevOps is about removing the barriers between two traditionally siloed teams, development and operations. In some organizations, there may not even be separate development and operations teams; engineers may do both. With DevOps, the two teams work together to optimize both the productivity of developers and the reliability of operations. They strive to communicate frequently, increase efficiencies, and improve the quality of services they provide to customers. They take full ownership for their services, often beyond where their stated roles or titles have traditionally been scoped by thinking about the end customer's needs and how they can contribute to solving those needs. Quality assurance and security teams may also become tightly integrated with these teams. Organizations using a DevOps model, regardless of their organizational structure, have teams that view the entire development and infrastructure lifecycle as part of their responsibilities.

DevOps Practices Explained

There are a few key practices that help organizations innovate faster through automating and streamlining the software development and infrastructure management processes. Most of these practices are accomplished with proper tooling.

One fundamental practice is to perform very frequent but small updates. This is how organizations innovate faster for their customers. These updates are usually more incremental in nature than the

occasional updates performed under traditional release practices. Frequent but small updates make each deployment less risky. They help teams address bugs faster because teams can identify the last deployment that caused the error. Although the cadence and size of updates will vary, organizations using a DevOps model deploy updates much more often than organizations using traditional software development practices.

Organizations might also use a microservices architecture to make their applications more flexible and enable quicker innovation. The microservices architecture decouples large, complex systems into simple, independent projects. Applications are broken into many individual components (services) with each service scoped to a single purpose or function and operated independently of its peer services and the application as a whole. This architecture reduces the coordination overhead of updating applications, and when each service is paired with small, agile teams who take ownership of each service, organizations can move more quickly.

However, the combination of microservices and increased release frequency leads to significantly more deployments which can present operational challenges. Thus, DevOps practices like continuous integration and continuous delivery solve these issues and let organizations deliver rapidly in a safe and reliable manner. Infrastructure automation practices, like infrastructure as code and configuration management, help to keep computing resources elastic and responsive to frequent changes. In addition, the use of monitoring and logging helps engineers track the performance of applications and infrastructure so they can react quickly to problems.

Together, these practices help organizations deliver faster, more reliable updates to their customers. Here is an overview of important DevOps practices.

DevOps Practices

The following are DevOps best practices:

- Continuous Integration
- Continuous Delivery
- Microservices
- Infrastructure as Code
- Monitoring and Logging
- Communication and Collaboration

Problems DevOps helps solve for businesses

1. **DevOps delivers more value to customers**
   The driving force of every business is delivering value to customers. DevOps is a way for tech teams to align technology to core business objectives. Businesses that have strong DevOps strategies are able to provide more value to their customers by removing or automating repetitive or low-value tasks. DevOps teams then have more time to spend focusing on innovating and enhancing technology quality. Without having to focus on menial tasks, iterative DevOps processes encourage innovation, agility, and flexibility that take the technology to the next level. DevOps teams are able to get high-quality solutions to your customers faster.
2. **DevOps reduces cycle time**
   When you create a strong DevOps process, you are encouraging agility within your development and operations teams. Teams have the ability to make changes to the platform and roll out new features on the fly. Because DevOps teams are constantly running through

testing and deployment, they are able to fix bugs and patch issues as they arise and can respond to customer needs quickly. Cycle time can be drastically reduced because iteration is built into the process.

3. **DevOps speeds up time to market**
   Similar to how DevOps reduces cycle time, DevOps speeds up the time to market because it allows teams to move faster. DevOps is a tool for the business that accelerates the development process and its ability to scale. DevOps gives teams quick feedback on features to reduce the response time to customer requests. The key to DevOps speeding up the time to market is that it reduces the complexity of the development process.
4. **DevOps encourages continuous improvement**
   At its core, DevOps aligns a company's people and resources around the same goals. When you have your development and operations teams working together, you are able to drive new efficiencies through increased visibility and communication. Teams are centered around rapid releases, updates, and deployment so any problems that arise can be fixed faster. DevOps creates a collaborative culture among development and operations teams.
5. **Silo Mentality and Blame Culture**