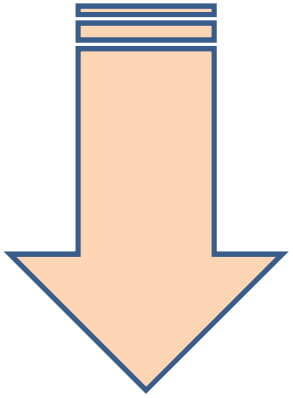
DEVOPS

18ITD44

DevOps



DevOps

UNIT-1:

Introduction to Devops: Waterfall model, Limitations of waterfall model, agile methodology, Limitations of agile method, waterfall vs agile, definition of Devops, Devops Stake holders, Devops goals, Devops life cycle, Devops stages: version control, continuous integration, continuous deliver, continuous deployment, continuous monitoring.

UNIT-2:

Version control with Git: introduction, version control system and types, difference between centralized version control and distributed version control, Git basics, Git features, installing Git, Git essentials, common commands in Git, Working with remote repositories.

UNIT-3:

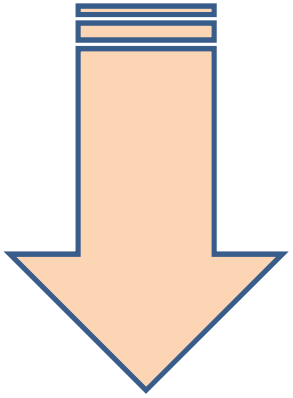
Continuous integration using Jenkins: Introduction-Understanding continuous integration, introduction about Jenkins, Build Cycle, Jenkins Architecture, installation, Jenkin Management, Adding a slave node to Jenkins, Building Delivery Pipeline, Pipeline as a Code, and Continuous Testing with Selenium.

UNIT-4:

Continuous Deployment: Containerization with Docker, Containerization using Kubernetes, Ecosystem and Networking, Configuration Management with Puppet, Configuration Management with Ansible, Continuous Monitoring with Nagios.

UNIT-1:

Introduction to Devops: Waterfall model, Limitations of waterfall model, agile methodology, Limitations of agile method, waterfall vs agile, definition of Devops, Devops Stake holders, Devops goals, Devops life cycle, **Devops stages:** version control, continuous integration, continuous deliver, continuous deployment, continuous monitoring.

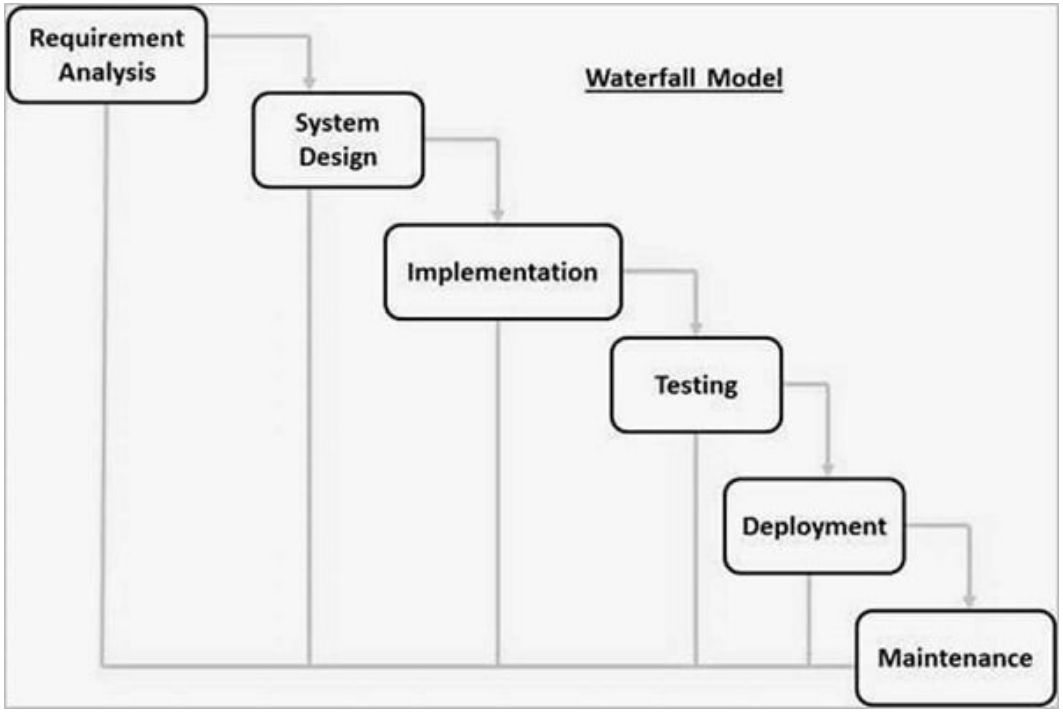


DevOps

Introduction to Devops:

Waterfall model:

- The Waterfall Model was the first Process Model to be introduced. It was widely used as it was easy to understand & implement .
- WaterFall Model is also known as **Linear-Sequential Life Cycle Model** because work is done linearly - the next stage begins after completion of the previous stage . Output of the previous stage is the input of the next stage .
- The name **"WaterFall Model"** is because the process looks like flowing steadily downwards - as shown below just like a waterfall .



DevOps

Waterfall model:

- The Phases are:-
- **Requirement Analysis:** In this phase all the requirements about the project are gathered . For eg features in projects , etc . All these requirements are well documented in the SRS (Software Requirement Specifications) document.

- **System Design:** The SRS document is then used to design the system . Defining system requirements like hardware, modules (any),etc.
- **Implementation:** After the System design phase is completely over . Implementation phase starts . Here each unit is individually developed . This stage goes on until all the requirements in the SRS document are not developed .

- **Testing & Integration:** Each individual unit is tested for potential bugs or errors . After testing is successful , individual units are integrated together.
- **Deployment:** Once the integration is done, the software is deployed to production servers.
- **Maintenance:** Project is continuously being monitored for any user inconvenience or bugs . Time to time new patches are released as a result of any bug fixes.

DevOps

Projects best fit for WaterFall Model:

- Requirements are clear
- Technology used is not dynamic
- Project is short

Advantages of WaterFall Model:

- Model is simple & easy to understand
- Clear defined milestones & deadlines
- Properly documented , hence team is focused towards one common goal
- Clearly defined stages
- Easy to arrange tasks

Disadvantages of WaterFall Model:

- No working software is made till end of life cycle
- Client feedback is not taken at any stage.
- Not good fit for changing project requirements
- Not suitable for OOPs & long-on going projects
- Difficult to track progress within a stage
- Includes high amount of risk & uncertainty

DevOps

Limitations of waterfall model:

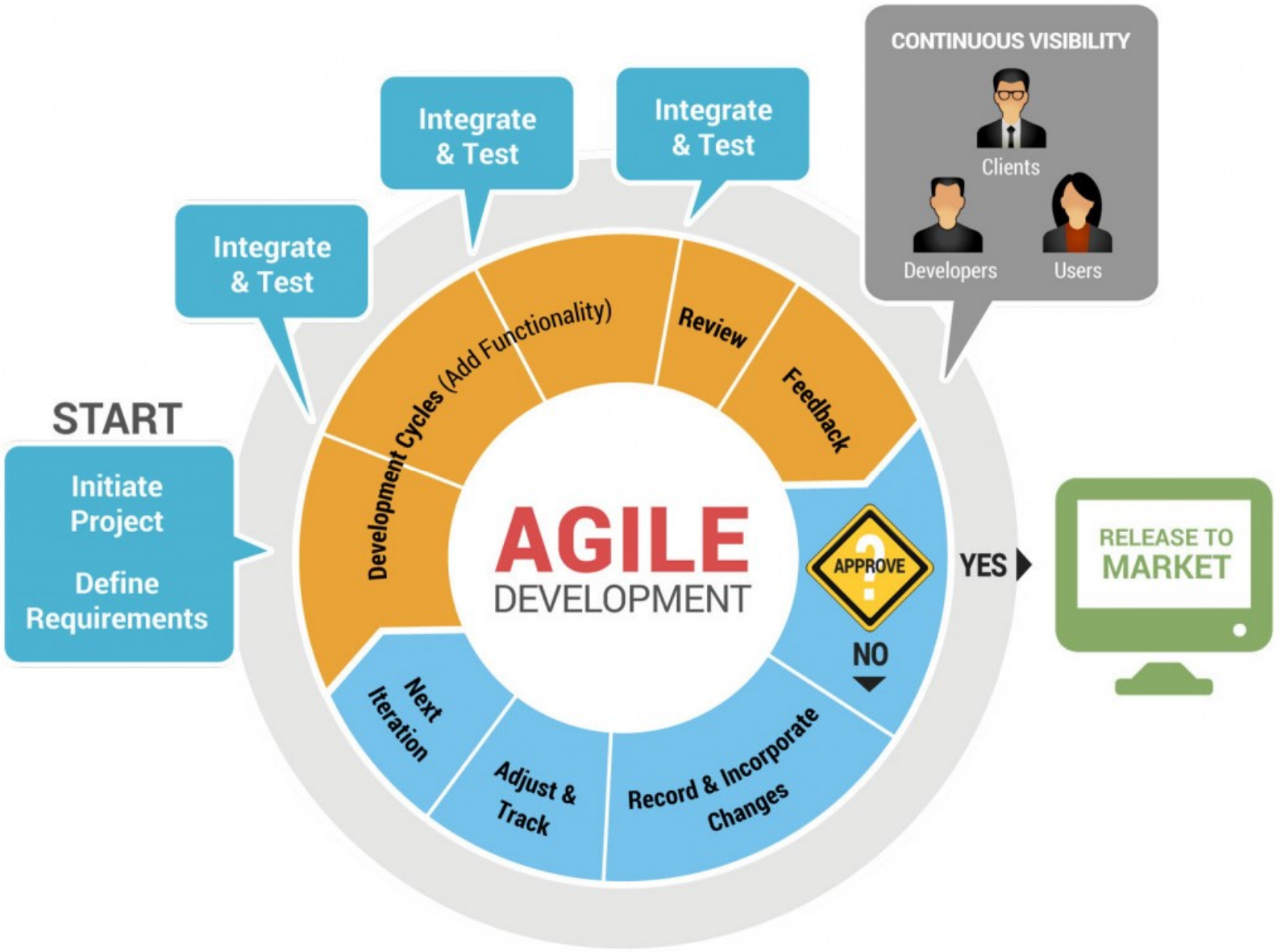
- The nature of the requirements will not change very much during development; during evolution.
- The model implies that you should attempt to complete a given stage before moving on to the next stage.
- Does not account for the fact that requirements constantly change
- It also means that customers cannot use anything until the entire system is complete
- The model implies that once the product is finished, everything else is maintenance
- Surprises at the end are very expensive
- Some teams sit ideal for other teams to finish
- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process

DevOps

Agile Methodology:

- In practical terms, agile software development methodologies are all about delivering small pieces of working software quickly to improve customer satisfaction.
- These methodologies use adaptive approaches and teamwork to focus on continuous improvement.
- Instead of developing software sequentially from one phase to the next, which is how the waterfall method ensures product quality, an agile method can promote development and testing as concurrent and continuous processes. Put another way, waterfall development holds that an entire phase should be completed before moving on to the next, whereas agile supports multiple sequences happening at the same time.

Agile Model


CONTINUOUS VISIBILITY
Clients
Developers
Users
Integrate & Test
Integrate & Test
Integrate & Test
START
Initiate Project
Define Requirements
Development Cycles (Add Functionality)
Review
Feedback
AGILE
DEVELOPMENT
APPROVE
YES
NO
RELEASE TO MARKET
Next Iteration
Adjust & Track
Record & Incorporate Changes

Agile - Advantages

- **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.** Customer satisfaction and quality deliverables are the focus.
- **Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.** Don't fight change, instead learn to take advantage of it.
- **Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.** Continually provide results throughout a project, not just at its peeks.
- **Business people and developers must work together daily throughout the project.** Collaboration is key.
- **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.** Bring talented and hardworking members to the team and get out of their way.
- **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.** Eliminate as many opportunities for miscommunication as possible.
- **Working software is the primary measure of progress.** It doesn't need to be perfect, it needs to work.

Agile - Advantages

- **Agile processes promote sustainable(stable) development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.** Slow and steady wins the race.
- **Continuous attention to technical excellence and good design enhances agility.** Don't forget to pay attention to the small stuff.
- **Simplicity—the art of maximizing the amount of work not done—is essential.** Trim the fat.
- **The best architectures, requirements, and designs emerge from self-organizing teams.** Related to Principle 5, you'll get the best work from your team if you let them figure out their own roles.
- **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.** Elicit(extract) and provide feedback, absorb the feedback, and adjust where needed.

Agile - disadvantages

- **Less predictable.** The flexibility at the core of the Agile method also means a much lower degree of predictability**. It can be much more difficult to accurately estimate the time necessary or quantify the resources** and efforts required to complete a project. Many teams fear this uncertainty, and that fear can lead to frustration and **poor decision-making**.
- **More time and commitment.** Communication and collaboration is great, but that constant interaction takes more time and energy for everyone involved.
- **Greater demands on developers and clients.** Commitment from everyone involved is required for Agile Methodology to be effective. Anyone who isn't on board can negatively impact the quality of a project.
- **Lack of necessary documentation.** Because tasks are often completed just in time for development under the Agile Method, **documentation tends to be less thorough**, which can lead to misunderstanding and difficulties down the road.
- **Projects easily fall off track.** The less-structured nature of Agile Methodology means projects can easily go astray or run beyond the original scope of the project.

Agile - Limitations

- **LIMITED SUPPORT FOR A REMOTE AGILE TEAM**

  The different geographical location can become one of the crucial limitations of agile methods, as it can cause many problems in so many different ways.

  I) Time zone differences, communication gap, maintainig documents from distant places

- **LIMITED SUPPORT FOR DEVELOPMENT INVOLVING LARGE TEAMS**
- Larger agile teams have sub-teams of developers who may be working from different areas. Large teams focus on large projects and commonly needed to solve complex problems. Only the senior programmers and developers are capable, of taking the kind of decisions required during the development process. Since the agile approach prone on the universality and cross-functionality of agile teams but the reality is quite the opposite.
- **LIMITED SUPPORT FOR DEVELOPING SAFETY-CRITICAL SOFTWARE**

Agile - Limitations

- **THE RISK INVOLVED IN FOLLOWING CUSTOMERS' REQUIREMENTS**

  If the customer is not sure about what final outcome that they demand, then the project can easily get taken off from the track. Selecting the most complex or difficult component may introduce the danger of failing to create the system the customer needs.

- **AGILE DEVELOPMENT AS A MICROMANAGED APPROACH**

  The pressure is persistently on for every week deliveries hence quality suffers, all they care about is the timeline. The loss of control at the management level leads to micro-management. Agile development is a true challenge. Especially in case, you attempt to communicate with those who are far from development methods whatsoever.

- **AGILE CULTURE IS A LONG TERM APPROACH IN REAL TIME AGILE DEVELOPMENT IMPLEMENTATION**

  The agile does not work in a hierarchy-driven organizational setup. The agile school of thought circles around the smart moves of the digital chessboard. It is the collection of trackers, pointers, calculators, planners, testers, and designing tools. Which divide the task into the possibly smallest cycles and assign it to different team members.

Agile - Limitations

- **AGILE METHODOLOGY CAN BE TRICKIER TO IMPLEMENT**

  Agile will not work in a scenario where a flaw is not an option. Today, agile is more than an agile manifesto; individual and interactions, working software, customer collaboration and responding to change. Agile is not just a team mechanism or computer game but it is also like thinking agile, doing agile and be agile. Today, agile is about business agility, communicating agility and evolving with agility.

- **MANAGEMENT FAILURES OF AGILE SOFTWARE DEVELOPMENT**

  Agile team organism has the ability to glue together the versatile members in one team. Thus it can uplift the burden of the heaviest and the speediest mechanism of any organization. To merge the Excellency and proficiency, of all-around members into one team is an imperative need of the super evolving era of digital transformation.

DevOps

- ✓ The meaning of Agile is swift or versatile."**Agile process model**" refers to a software development approach based on iterative development.
- ✓ Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.
- Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.
-

DevOps

- ✓ The meaning of Agile is swift or versatile."**Agile process model**" refers to a software development approach based on iterative development.
- ✓ Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.
- Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.
-

DevOps

Fig. Agile Model

Agile

When to Use:

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

Adv:

- Frequent Delivery
- Face-to-Face Communication with clients.
- Efficient design and fulfils the business requirement.
- Anytime changes are acceptable.
- It reduces total development time.

DisAdv:

- Due to the **shortage of formal documents**, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
- Due to the **lack of proper documentation**, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

✓ Waterfall vs Agile:

Agile	Waterfall
It separates the project development lifecycle into sprints (small amount of time dev team has to complete a work).	Software development process is divided into distinct phases.
It follows an incremental approach	Waterfall methodology is a sequential design process.
Agile methodology is known for its flexibility.	Waterfall is a structured software development methodology so most times it can be quite rigid.
Agile can be considered as a collection of many different projects.	Software development will be completed as one single project.
Agile is quite a flexible method which allows changes to be made in the project development requirements even if the initial planning has been completed.	There is no scope of changing the requirements once the project development starts.
Agile methodology, follow an iterative development approach because of this planning, development, prototyping and other software development phases may appear more than once.	All the project development phases like designing, development, testing, etc. are completed once in the Waterfall model.

DevOps

✓ **Waterfall vs Agile:**

Agile	Waterfall
Test plan is reviewed after each sprint	The test plan is rarely discussed during the test phase.
Agile development is a process in which the requirements are expected to change and evolve.	The method is ideal for projects which have definite requirements and changes not at all expected.
In Agile methodology, testing is performed concurrently with software development.	In this methodology, the “Testing” phase comes after the “Build” phase
Agile introduces a product mindset where the software product satisfies needs of its end customers and changes itself as per the customer’s demands.	This model shows a project mindset and places its focus completely on accomplishing the project.
Agile methdology works exceptionally well with Time & Materials or non-fixed funding. It may increase stress in fixed-price scenarios.	Reduces risk in the firm fixed price contracts by getting risk agreement at the beginning of the process.
Prefers small but dedicated teams with a high degree of coordination and synchronization.	Team coordination/synchronization is very limited.

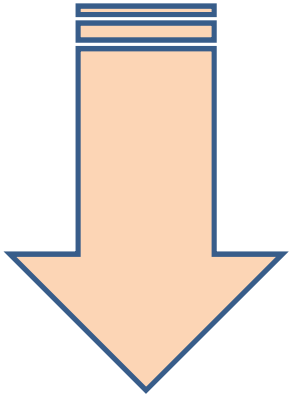
DevOps

✓ **Waterfall vs Agile:**

Agile	Waterfall
Products owner with team prepares requirements just about every day during a project.	Business analysis prepares requirements before the beginning of the project.
Test team can take part in the requirements change without problems.	It is difficult for the test to initiate any change in requirements.
Description of project details can be altered anytime during the SDLC process.	Detail description needs to implement waterfall software development approach.
The Agile Team members are interchangeable, as a result, they work faster. There is also no need for project managers because the projects are managed by the entire team	In the waterfall method, the process is always straightforward so, project manager plays an essential role during every stage of SDLC.

UNIT-1 contd...:

Introduction to Devops: definition of Devops, Devops Stake holders, Devops goals, Devops life cycle, **Devops stages:** version control, continuous integration, continuous deliver, continuous deployment, continuous monitoring.



DevOps

- ✓ definition of Devops
- ✓ Devops Stake holders
- ✓ Devops goals
- ✓ Devops life cycle

definition of Devops:

- The DevOps is the combination of two words, one is **Development** and other is **Operations**. It is a culture to promote the development and operation process collectively.
- This allows a single team to handle the entire application lifecycle, from development to **testing, deployment**, and **operations**.
- DevOps helps you to reduce the disconnection between software developers, quality assurance (QA) engineers, and system administrators.
- 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.
- 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.

Why DevOps?

- 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.

DevOps – Advantages & Disadvantages

Advantages:

- DevOps is an excellent approach for quick development and deployment of applications.
- It responds faster to the market changes to improve business growth.
- DevOps escalate business profit by decreasing software delivery time and transportation costs.
- DevOps clears the descriptive process, which gives clarity on product development and delivery.
- It improves customer experience and satisfaction.
- DevOps simplifies collaboration and places all tools in the cloud for customers to access.
- DevOps means collective responsibility, which leads to better team engagement and productivity.

Disadvantages:

- DevOps professional or expert's developers are less available.
- Developing with DevOps is so expensive.
- Adopting new DevOps technology into the industries is hard to manage in short time.
- Lack of DevOps knowledge can be a problem in the continuous integration of automation projects.

DevOps – Stakeholders

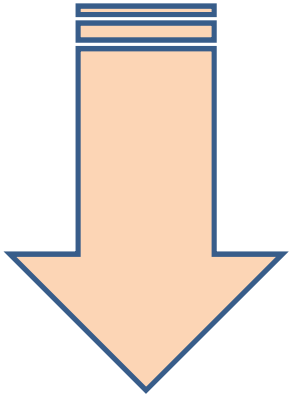
Dev Includes all people involved in developing software products and services including but not exclusive to:

- Architects,
- business representatives,
- customers,
- product owners,
- project managers,
- quality assurance (QA),
- testers and analysts,
- Suppliers etc...

Ops **Includes** all people involved in delivering and managing software products and services including but not exclusive to:

- Information security professionals,
- systems engineers,
- system administrators,
- IT operations engineers,
- release engineers,
- database administrators (DBAs),
- network engineers,
- support professionals,
- third party vendors and suppliers…

Devops goals, Devops life cycle, **Devops stages:** version control, continuous integration, continuous deliver, continuous deployment, continuous monitoring.



DevOps – Goals

1. Ensures effective collaboration between teams
2. Creates scalable infrastructure platforms
3. Builds on-demand release capabilities
4. Provides faster feedback

1. Ensures effective collaboration between teams

- Effective collaboration in any process relies on shared ownership.
- During the development process, all those involved should embrace the fact that everyone is equally responsible for the entire development process. Whether it is development, testing, or deployment, each team member should be involved.
- They should understand that they have an equal stake in the final outcome.
- In the DevOps paradigm, passing of work from one team to another is completely defined and broken down. This accelerates the entire process of development since collaboration between all the teams involved is streamlined.

2. Creates scalable infrastructure platforms

- The primary focus of DevOps is to create a sustainable infrastructure for applications that make them highly scalable.
- According to the demands of the modern-day business world, scalable apps have become an absolute necessity. In an ideal situation, the process of scaling should be reliable and fully automated.
- As a result, the app will have the ability to adapt to any situation when a marketing effort goes viral. With the app being scalable, it can adjust itself to large traffic volumes and provide an immaculate(clean) user experience.

DevOps – Goals

3. Builds on-demand release capabilities

- Companies must focus on keeping their software in a 'releasable' state.
- Continuous delivery will allow the software to add new features and go live at any stage.
- DevOps aims to automate the process of release management because it has several advantages.
- Automated release management is predictable, fast, and very consistent.
- Through automation, companies can **release new versions** as per their requirements.
- Automated release management also has complete and thorough **audit trials**, as these are essential for compliance purposes.

4. Provides faster feedback

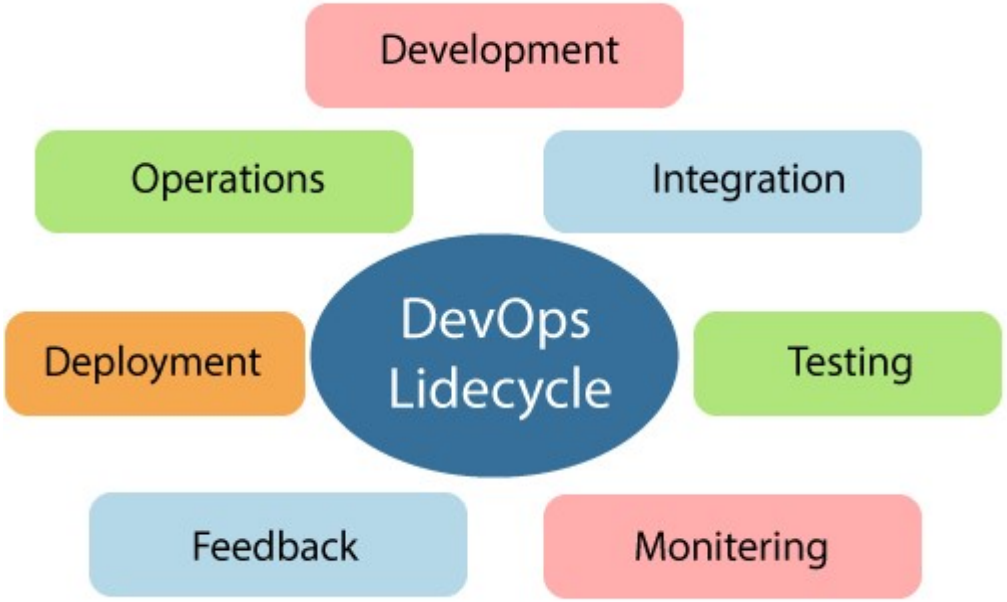
- Automating monotonous tasks such as testing and reporting will accelerate the process of rapid feedback.
- Since the development team will know what has to change, it can roll out the updated version faster. In addition, the team can better understand the impact of the changes that it has done in the software lifecycle.
- A concrete understanding of changes will assist team members in working efficiently in cool manner.
- With rapid feedback, the operations team and developers can make better decisions collectively and enhance the app's performance.

DevOps – Life cycle

https://www.javatpoint.com/devops-lifecycle

- **DevOps** defines an agile relationship between operations and Development. It is a process that is practiced by the development team and operational engineers together from beginning to the final stage of the product.
- **The DevOps lifecycle includes seven phases**

DevOps Lifecycle:



DevOps – lifecycle

7 stages in DevOps lifecycle:

1. Continuous Development
2. Continuous Integration
3. Continuous Testing
4. Continuous Monitoring
5. Continuous Feedback
6. Continuous Deployment
7. Continuous Operations

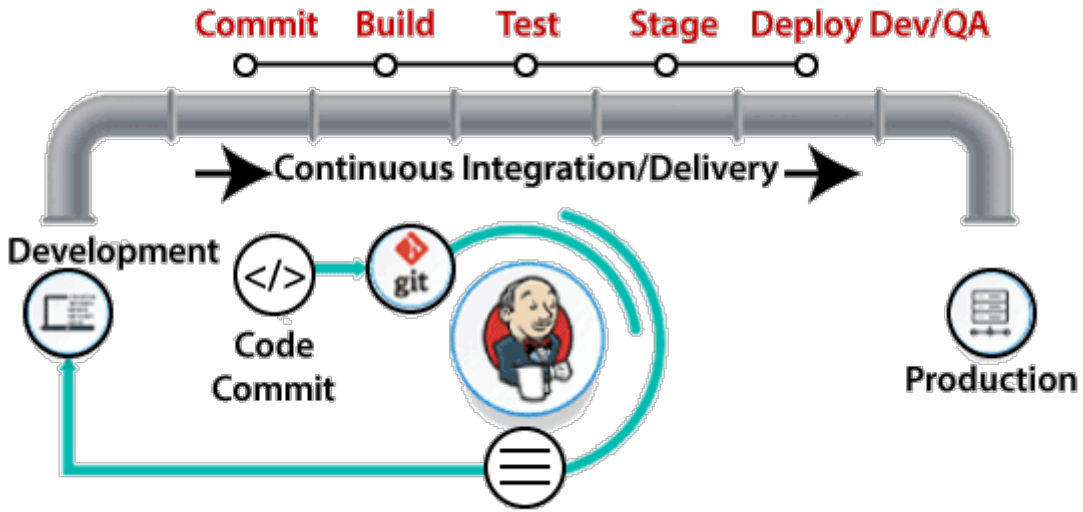
1. Continuous Development:

- This phase involves the planning and coding of the software.
- The vision of the project is decided during the planning phase. And the developers begin developing the code for the application.
- There are no DevOps tools that are required for planning, but there are several tools for maintaining the code.

DevOps – lifecycle

2. Continuous Integration

- This stage is the heart of the entire DevOps lifecycle. It is a software development practice in which the developers require to commit changes to the source code more frequently. This may be on a daily or weekly basis. Then every commit is built, and this allows early detection of problems if they are present. Building code is not only involved compilation, but it also includes **unit testing, integration testing, code review**, and **packaging**.
- The code supporting new functionality is continuously integrated with the existing code. Therefore, there is continuous development of software. The updated code needs to be integrated continuously and smoothly with the systems to reflect changes to the end-users.
- **Jenkins** is a popular tool used in this phase. Whenever there is a change in the Git repository, then Jenkins fetches the updated code and prepares a build of that code, which is an executable file in the form of war or jar. Then this build is forwarded to the test server or the production server.



DevOps – lifecycle

3. Continuous Testing

- This phase, where the developed software is continuously testing for bugs. For constant testing, automation testing tools such as **TestNG, JUnit, Selenium**, etc are used. These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there is no flaw in the functionality. In this phase, **Docker** Containers can be used for simulating the test environment.

- **Selenium** does the automation testing, and TestNG generates the reports. This entire testing phase can automate with the help of a Continuous Integration tool called **Jenkins**.
- Automation testing saves a lot of time and effort for executing the tests instead of doing this manually. Apart from that, report generation is a big plus. The task of evaluating the test cases that failed in a test suite gets simpler. Also, we can schedule the execution of the test cases at predefined times. After testing, the code is continuously integrated with the existing code.

DevOps – lifecycle

4. Continuous Monitoring

- Monitoring is a phase that involves all the operational factors of the entire DevOps process, where important information about the use of the software is recorded and carefully processed to find out trends and identify problem areas. Usually, the monitoring is integrated within the operational capabilities of the software application.
- It may occur in the form of documentation files or maybe produce large-scale data about the application parameters when it is in a continuous use position. The system errors such as server not reachable, low memory, etc are resolved in this phase. It maintains the security and availability of the service.

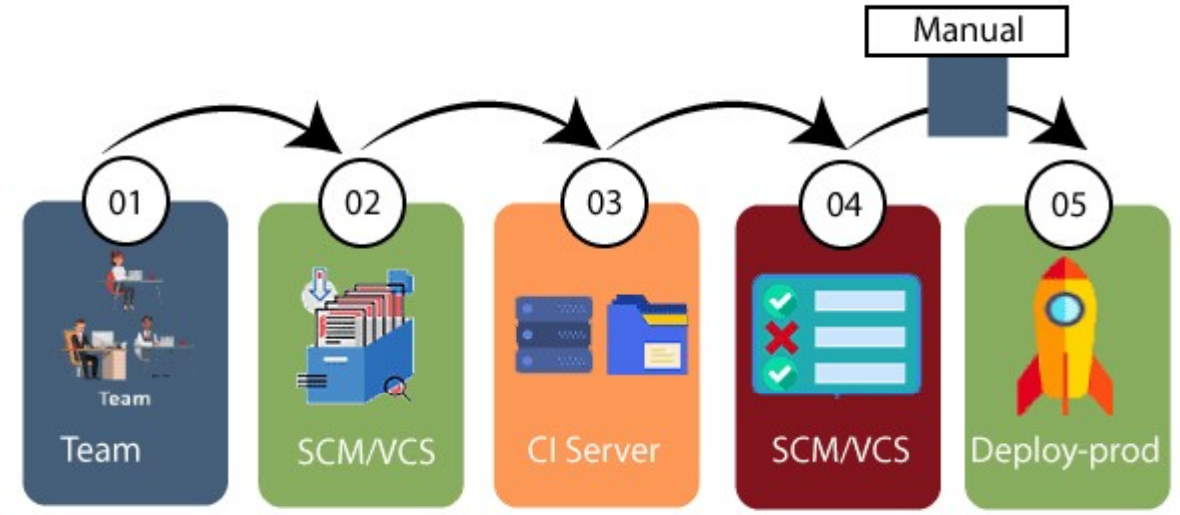
5. Continuous Feedback

- The application development is consistently improved by analyzing the results from the operations of the software. This is carried out by placing the critical phase of constant feedback between the operations and the development of the next version of the current software application.
- The continuity is the essential factor in the DevOps as it removes the unnecessary steps which are required to take a software application from development, using it to find out its issues and then producing a better version.

DevOps – lifecycle

6. Continuous Deployment

- In this phase, the code is deployed to the production servers. Also, it is essential to ensure that the code is correctly used on all the servers.



- The new code is deployed continuously, and configuration management tools play an essential role in executing tasks frequently and quickly. Here are some popular tools which are used in this phase, such as **Chef, Puppet, Ansible**, and **SaltStack**.

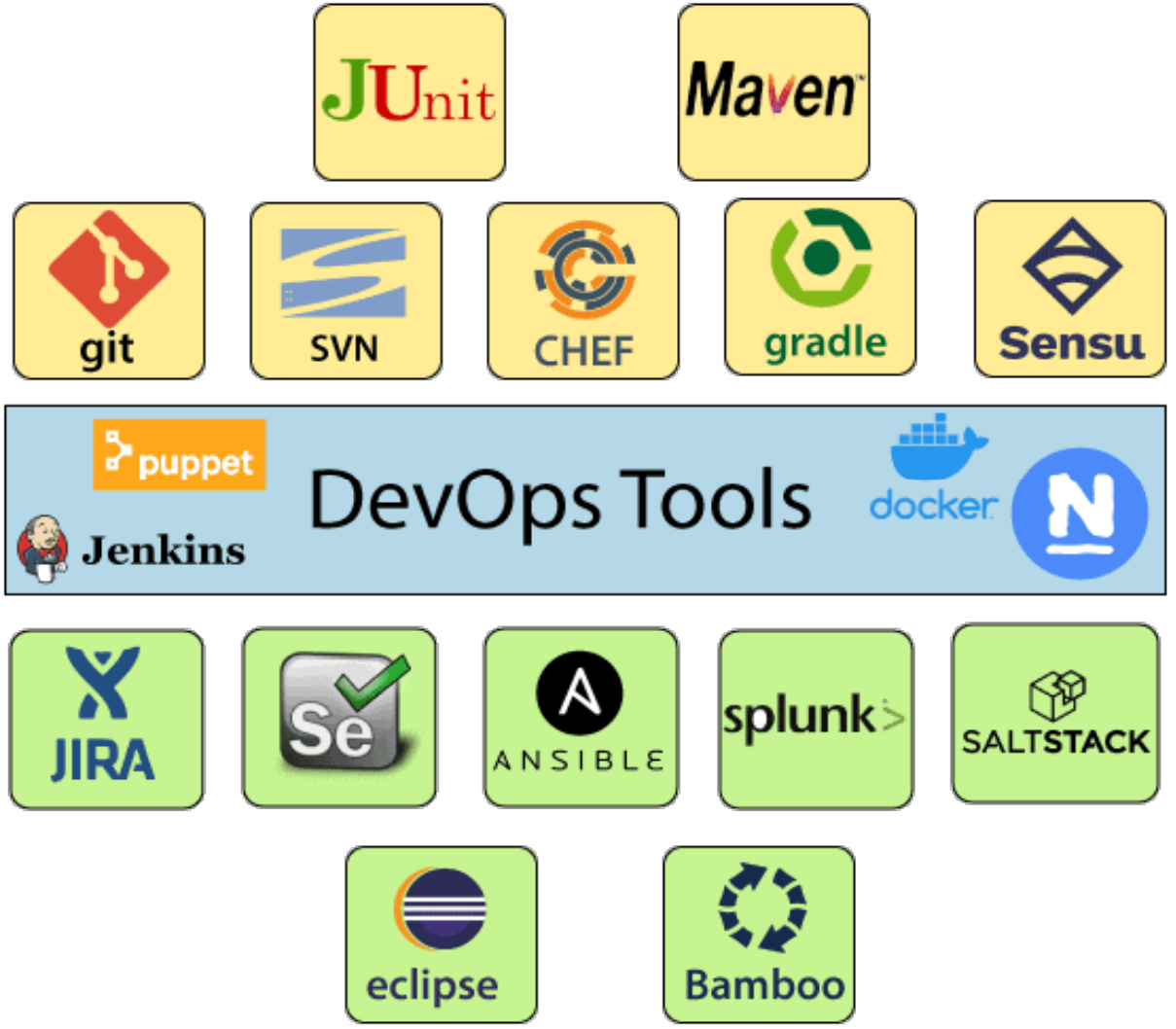
DevOps – lifecycle

- Containerization tools are also playing an essential role in the deployment phase. **Vagrant** and **Docker** are popular tools that are used for this purpose. These tools help to produce consistency across development, staging, testing, and production environment. They also help in scaling up and scaling down instances softly.
- Containerization tools help to maintain consistency across the environments where the application is tested, developed, and deployed. There is no chance of errors or failure in the production environment as they package and replicate the same dependencies and packages used in the testing, development, and staging environment. It makes the application easy to run on different computers.

7. Continuous Operations:

- All DevOps operations are based on the continuity with complete automation of the release process and allow the organization to accelerate the overall time to market continuingly.

DevOps – Tools



DevOps – Tools

https://www.atlassian.com/devops/devops-tools

DevOps life cycle stages:

- Plan
- Build
- Continuous integration and deployment
- Monitor
- Operate
- Plan

Build

DevOps – Tools

Production-identical environments for development:

Infrastructure as code:

Continuous integration and delivery

DevOps – Tools

Continuous Integration:

Test:

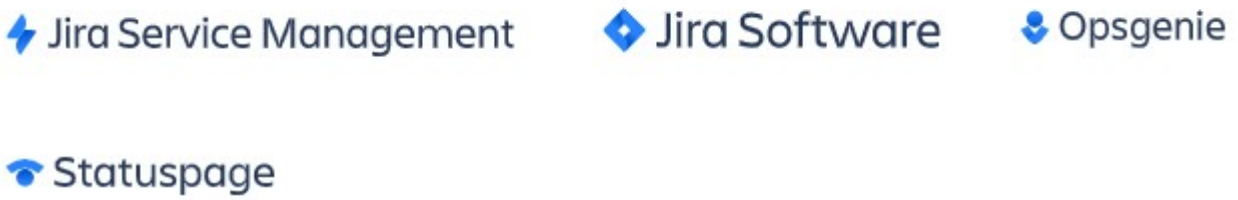
Deployment:

DevOps – Tools

Operate


Continuous Feedback

- Application and server performance monitoring:



DevOps – Tools

Jenkins: https://www.jenkins.io/download/

- Jenkins a DevOps tool for monitoring execution of repeated tasks. It is one of the best software deploy tools which helps to integrate project changes more easily by quickly finding issues.
- It supports **continuous integration and continuous delivery**
- Jenkins is an open source automation tool written in Java programming language **that allows continuous integration..**
- Jenkins **builds** and **tests** our software projects which continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.

Features:

- It increases the scale of automation
- Jenkins requires little maintenance and has built-in GUI tool for easy updates.
- It offers 400 plugins to support building and testing virtually any project.
- It is Java-based program ready to run with Operating systems like Windows, Mac OS X, and UNIX
- It supports continuous integration and continuous delivery
- It can easily set up and configured via web interface
- It can distribute tasks across multiple machines thereby increasing concurrency.

https://www.jenkins.io/download/

DevOps – Puppet

Puppet: Puppet is a **configuration management tool** developed by Puppet Labs in order to automate infrastructure management and configuration. Puppet is a very powerful tool which helps in the concept of Infrastructure as code.

- Puppet follows client server model.

Puppet Enterprise is a DevOps tool. It is one of the popular DevOps tools that allows managing **entire infrastructure as code** without expanding the size of the team.

Features:

- Puppet enterprise tool eliminates manual work for software delivery process. It helps developer to deliver great software rapidly
- Model and manage entire environment
- Intelligent orchestration and visual workflows
- Real-time context-aware reporting
- Define and continually enforce infrastructure
- It inspects and reports on packages running across infrastructure
- Desired state conflict detection and remediation

https://puppet.com/try-puppet/puppet-enterprise/

DevOps – Tools

Docker: Docker is a container management service.

Docker is a DevOps technology suite. It allows DevOps teams to build, ship, and run distributed applications. This tool allows users to assemble apps from components and work collaboratively.

- The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

https://www.docker.com/products/docker-hub

DevOps – Tools

Selenium:

- Selenium is an open-source tool that automates web browsers. It provides a single interface that lets you write test scripts in programming languages like Ruby, Java, NodeJS, PHP, Perl, Python, and C#, among others.

- Selenium is a free (open source) **automated testing suite for web applications** across different browsers and platforms. It is quite similar to HP Quick Test Pro (QTP now UFT) only that Selenium focuses on automating web-based applications. Testing done using Selenium tool is usually referred as Selenium Testing.

- Selenium IDE Tutorial for Beginners (guru99.com)

DevOps – Tools

Ansible:

Ansible is a leading DevOps tool. It is a simple way to automate IT for automating entire application lifecycle. It is one of the best automation tools for DevOps which makes it easier for DevOps teams to scale automation and speed up productivity.

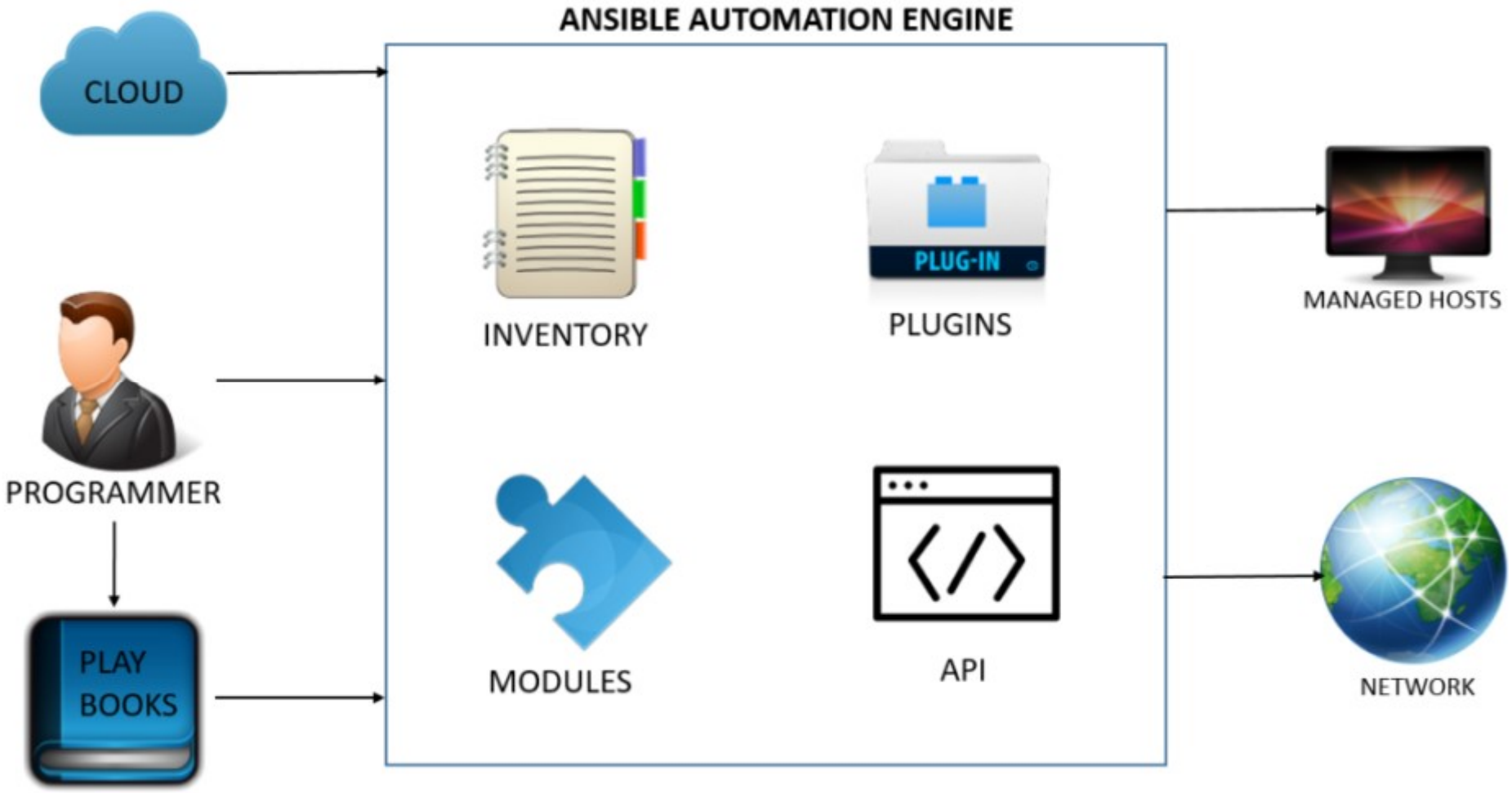
- Ansible is a simple but powerful configuration management and orchestration tool.
- It is fundamentally intended for IT professionals, who use it for configuration management, cloud provisioning, application deployment, intra-service orchestration, updates on workstations and servers, and nearly for anything a systems administrator does on a day-to-day basis.
- often need maintenance, updates, scaling-up activities, for system admins to keep up-to-date of everything manually is a burden and a daunting task. The automation simplifies complex tasks using tools like Ansible,

Features:

- It is easy to use open source deploy apps
- It helps to avoid complexity in the software development process
- IT automation eliminates repetitive tasks that allow teams to do more strategic work
- It is an ideal tool to manage complex deployments and speed up development process

https://www.redhat.com/en/technologies/management/ansible/try-it

DevOps - ANSIBLE tool



DevOps – Tools

Gradle:

- Gradle is a build automation tool known for its flexibility to build software.
- A build automation tool is used **to automate the creation of applications**. The building process includes compiling, linking, and packaging the code. The process becomes more consistent with the help of build automation tools.
- Gradle is the official build tool for Android. Other IDEs are Eclipse, IntelliJIDEA, visual studio 2019 and XCode

DevOps – Tools

Chef:

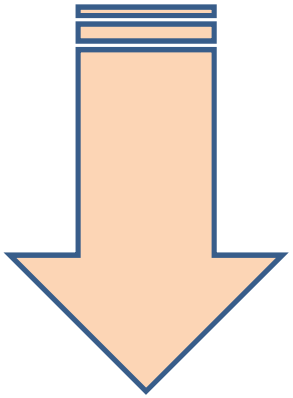
- Today, in an Organization the system admins or ***DevOps Engineer*** spends more time in deploying new services and application, installing and updating network packages and making machine server ready for deployment. This causes tedious human efforts and requires huge human resources.To solve this problem, configuration management was introduced. By using configuration management tools like Chef, Puppet you can deploy, repair and update the entire application infrastructure with automation.
- Chef is a useful DevOps tool for achieving speed, scale, and consistency. It is a Cloud based system. It is one of the best DevOps automation tools that can be used to ease out complex tasks and perform automation.
- Chef is a powerful automation tool that can deploy, repair and update and also manage server and application to any environment.

Features:

- Accelerate cloud adoption
- Effectively manage data centers
- It can manage multiple cloud environments
- It maintains high availability

https://downloads.chef.io/

Git, JUnit, Maven, **Devops stages:** Version Control, continuous integration, continuous deliver, continuous deployment, continuous monitoring.



DevOps – Tools - Git

Git:

- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- Git relies on the basis of distributed development of software where more than one developer may have access to the source code of a specific application and can modify changes to it which may be seen by other developers.
- It allows the user to have “versions” of a project, which show the changes that were made to the code over time, and allows the user to back track if necessary and undo those changes.

DevOps – Tools - JUnit

Junit:

- JUnit is an open source framework designed for the purpose of writing and running tests in the Java programming language
- JUnit is a Java Unit Testing framework that's one of the best test methods for regression testing. An open-source framework, it is used **to write and run repeatable automated tests**

DevOps – Tools - Maven

Maven:

- Maven is a popular **open-source build tool** developed by the Apache Group to build, publish, and deploy several projects at once for better project management.
- Maven is written in Java and is used to build projects written in C#, Scala, Ruby, etc. Based on the Project Object Model (POM), this tool has made the lives of Java developers easier while developing reports, checks build and testing automation setups.

DevOps – Stages

1. Version Control
2. Continuous integration
3. Continuous delivery
4. Continuous deployment
5. Continuous monitoring

DevOps – Stages

Version Control:

- Version control, also known as source control, is the practice of tracking and managing changes to software code.
- VCS are sometimes known as SCM (Source Code Management) tools or RCS (Revision Control System)
- Git is free and open source.
- Version control systems are software tools that help software teams manage changes to source code over time.
- Version control enables teams to deal with conflicts that result from having multiple people working on the same file or project at the same time, and provides a safe way to make changes and roll them back if necessary.
- Using version control early in your team's or product's lifecycle will facilitate adoption of good habits.
- Version control systems help software teams work faster and smarter. They are especially useful for DevOps teams since they help them to reduce development time and increase successful deployments.
- Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

DevOps – Stages

Version Control:

- For an organization, code is the most valuable asset whose value must be protected.
- Version control protects source code from both catastrophe and the casual degradation of human error and unintended consequences.
- Software developers working in teams are continually writing new source code and changing existing source code. The code for a project, app or software component is typically organized in a folder structure or "file tree". One developer on the team may be working on a new feature while another developer fixes an unrelated bug by changing code, each developer may make their changes in several parts of the file tree.
- Version control helps teams solve these kinds of problems, tracking every individual change by each contributor and helping prevent concurrent work from conflicting. Changes made in one part of the software can be incompatible with those made by another developer working at the same time. This problem should be discovered and solved in an orderly manner without blocking the work of the rest of the team. Further, in all software development, any change can introduce new bugs on its own and new software can't be trusted until it's tested. So testing and development proceed together until a new version is ready.

DevOps – Stages

Version Control:

- Good version control software supports a developer's preferred workflow without imposing one particular way of working. Ideally it also works on any platform, rather than dictate what operating system or tool chain developers must use. Great version control systems facilitate a smooth and continuous flow of changes to the code rather than the frustrating and clumsy mechanism of file locking - giving the green light to one developer at the expense of blocking the progress of others.
- Software teams that do not use any form of version control often run into problems like not knowing which changes that have been made are available to users or the creation of incompatible changes between two unrelated pieces of work that must then be painstakingly untangled and reworked. If you're a developer who has never used version control you may have added versions to your files, perhaps with suffixes like "final" or "latest" and then had to later deal with a new final version. Perhaps you've commented out code blocks because you want to disable certain functionality without deleting the code, fearing that there may be a use for it later. Version control is a way out of these problems.

DevOps – Stages

Version Control:

- Version control software is an essential part of the every-day of the modern software team's professional practices. Individual software developers who are accustomed to working with a capable version control system in their teams typically recognize the incredible value version control also gives them even on small solo projects. Once accustomed to the powerful benefits of version control systems, many developers wouldn't consider working without it even for non-software projects.

DevOps – VCS (Version Control Systems)

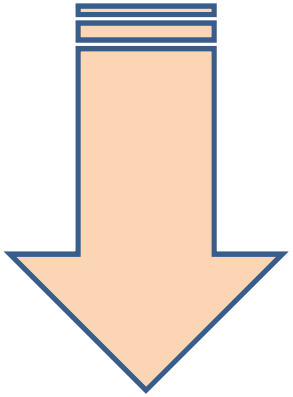
Benefits of VCS:

1. Version control also helps developers move faster and allows software teams to preserve efficiency and agility as the team scales to include more developers.
2. A complete long-term change history of every file. This means every change made by many individuals over the years. Changes include the creation and deletion of files as well as edits to their contents.

- Having the complete history enables going back to previous versions to help in root cause analysis for bugs and it is crucial when needing to fix problems in older versions of software. If the software is being actively worked on, almost everything can be considered an "older version" of the software.
  Creating a "branch" in VCS tools keeps multiple streams of work independent from each other while also providing the facility to merge that work back together, enabling developers to verify that the changes on each branch do not conflict.
- Being able to trace each change made to the software and connect it to project management and bug tracking software such as Jira, and being able to annotate each change with a message describing the purpose and intent of the change can help not only with root cause analysis and other forensics.

Devops stages: Version Control, **continuous integration, continuous delivery, continuous deployment, continuous monitoring.**

Continuous Integration in DevOps | How it is Performed with Advantages (educba.com)



DevOps – Continuous Integration

1. **Continuous integration:**

Definition:

- *Continuous integration refers to the build and unit testing stages of the software release process. Every revision that is committed triggers an automated build and test.*

Continuous Integration in DevOps is the process of automating the build and deploy phase through certain tools and best practices.

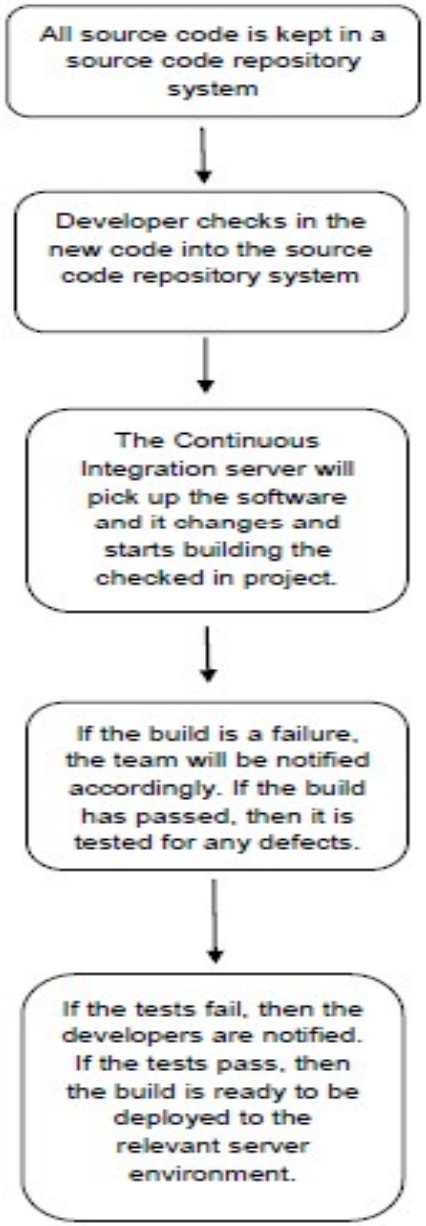
- Continuous integration has become a very integral part of any software development process. The continuous Integration process helps to answer the following questions for the software development team .
- Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

Addresses:

- Do all the software components work together as they should?
- Is the code too complex for integration purposes?
- Does the code adhere to the established coding standards?
- How much code is covered by automated tests?
- Were all the tests successful after the latest change?

DevOps – Continuous Integration

continuous integration process working:



DevOps – Continuous Integration

continuous integration process working: Steps:

1. First, a developer commits the code to the version control repository. Meanwhile, the Continuous Integration server on the integration build machine polls source code repository for changes (e.g., every few minutes).
2. Soon after a commit occurs, the Continuous Integration server detects that changes have occurred in the version control repository, so the Continuous Integration server retrieves the latest copy of the code from the repository and then executes a build script, which integrates the software.
3. The Continuous Integration server generates feedback by e-mailing build results to the specified project members.
4. Unit tests are then carried out if the build of that project passes. If the tests are successful, the code is ready to be deployed to either the staging or production server.
5. The Continuous Integration server continues to poll for changes in the version control repository and the whole process repeats.

DevOps – Continuous Integration

continuous integration process working: Steps:

1. First, a developer commits the code to the version control repository. Meanwhile, the Continuous Integration server on the integration build machine polls source code repository for changes (e.g., every few minutes).
2. Soon after a commit occurs, the Continuous Integration server detects that changes have occurred in the version control repository, so the Continuous Integration server retrieves the latest copy of the code from the repository and then executes a build script, which integrates the software.
3. The Continuous Integration server generates feedback by e-mailing build results to the specified project members.
4. Unit tests are then carried out if the build of that project passes. If the tests are successful, the code is ready to be deployed to either the staging or production server.
5. The Continuous Integration server continues to poll for changes in the version control repository and the whole process repeats.

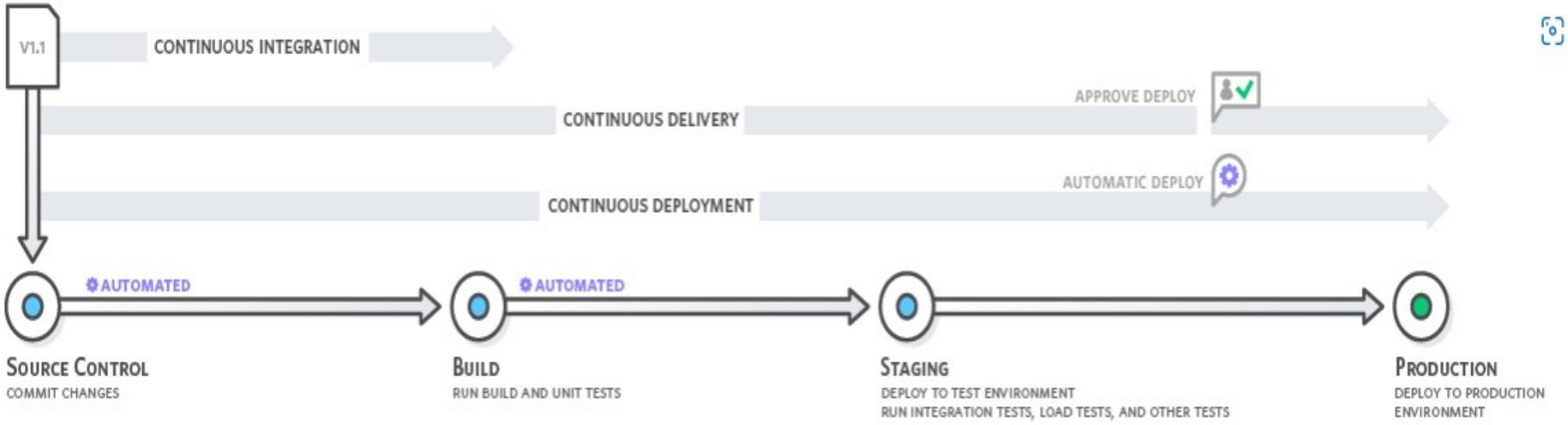
DevOps – Continuous Integration

continuous integration process working: Steps:

1. First, a developer commits the code to the version control repository. Meanwhile, the Continuous Integration server on the integration build machine polls source code repository for changes (e.g., every few minutes).
2. Soon after a commit occurs, the Continuous Integration server detects that changes have occurred in the version control repository, so the Continuous Integration server retrieves the latest copy of the code from the repository and then executes a build script, which integrates the software.
3. The Continuous Integration server generates feedback by e-mailing build results to the specified project members.
4. Unit tests are then carried out if the build of that project passes. If the tests are successful, the code is ready to be deployed to either the staging or production server.
5. The Continuous Integration server continues to poll for changes in the version control repository and the whole process repeats.

DevOps – Continuous Integration

- Continuous Integration in DevOps is the process of automating the build and deploy phase through certain tools and best practices. Continuous Integration (CI) applies to all types of software projects such as developing websites, Mobile Applications, and Microservices based APIs. There are three major categories of tools associated with CI: Versioning tool, Build tool, and repositories for centralized artifacts. CI pipeline helps the developer to easily commit the code and helps for quality development.
-



DevOps – Continuous Integration

Advantages:

1. Improve Developer Productivity:

- Continuous integration helps your team be more productive by freeing developers from manual tasks and encouraging behaviors that help reduce the number of errors and bugs released to customers.

2. Find and Address Bugs Quicker

ith more frequent testing, your team can discover and address bugs earlier before they grow into larger problems later.

. Deliver Updates Faster

- Continuous integration helps your team deliver updates to their customers faster and more frequently.

CI Tools:

DevOps – Continuous Integration

Continuous Integration

List Of The Top Continuous Integration Tools

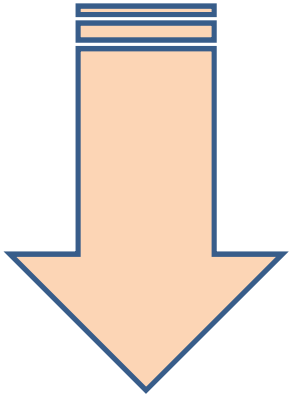
- #1) Buddy
- #2) Jenkins
- #3) Buildbot
- #4) ThoughtWorks
- #5) UrbanCode
- #6) Perforce Helix
- #7) Bamboo
- #8) TeamCity
- #9) CircleCI
- #10) Codeship
- #11) CruiseControl
- #12) Go/GoCD
- #13) Travis
- #14) Integrity
- #15) Strider or Strider CD

Devops stages : continuous delivery, continuous deployment, continuous monitoring.

https://www.flagship.io/glossary/continuous-deployment/

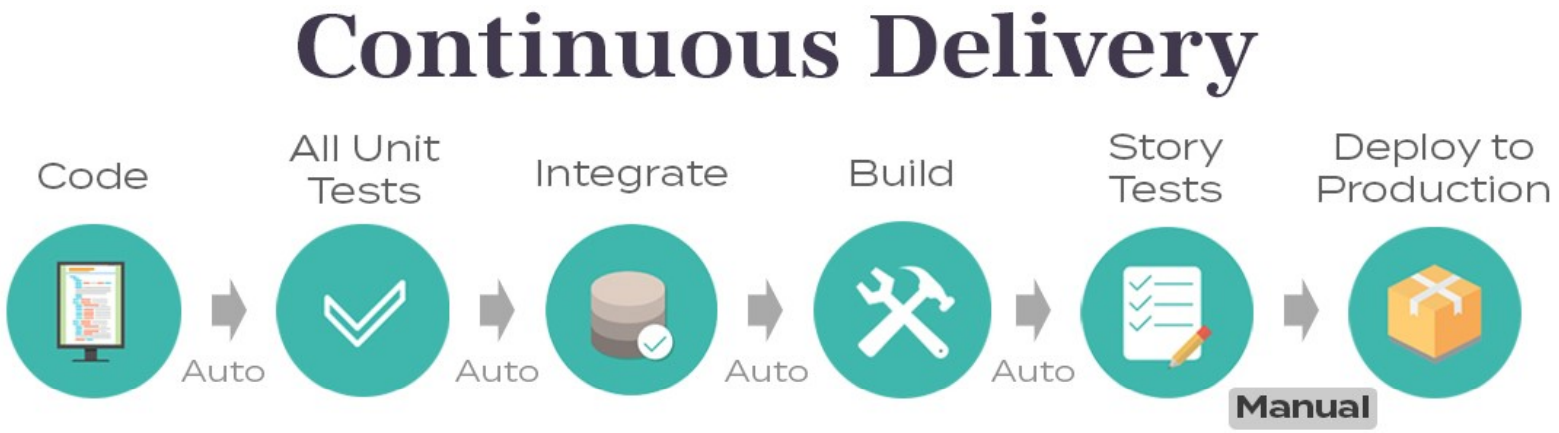
https://devopedia.org/continuous-delivery

https://www.headspin.io/blog/what-is-continuous-monitoring-in-devops



DevOps – Continuous Delivery

- Continuous Delivery (CD) goes one step further from Continuous Integration (CI). It ensures that every code change is tested and ready for the production environment, after a successful build. CI ensures every code is committed to the main code repository whereas CD ensures the system is in an executable state at all times, after changes to code.
- The automation of methods to safely deliver changes into production is known as Continuous Delivery.



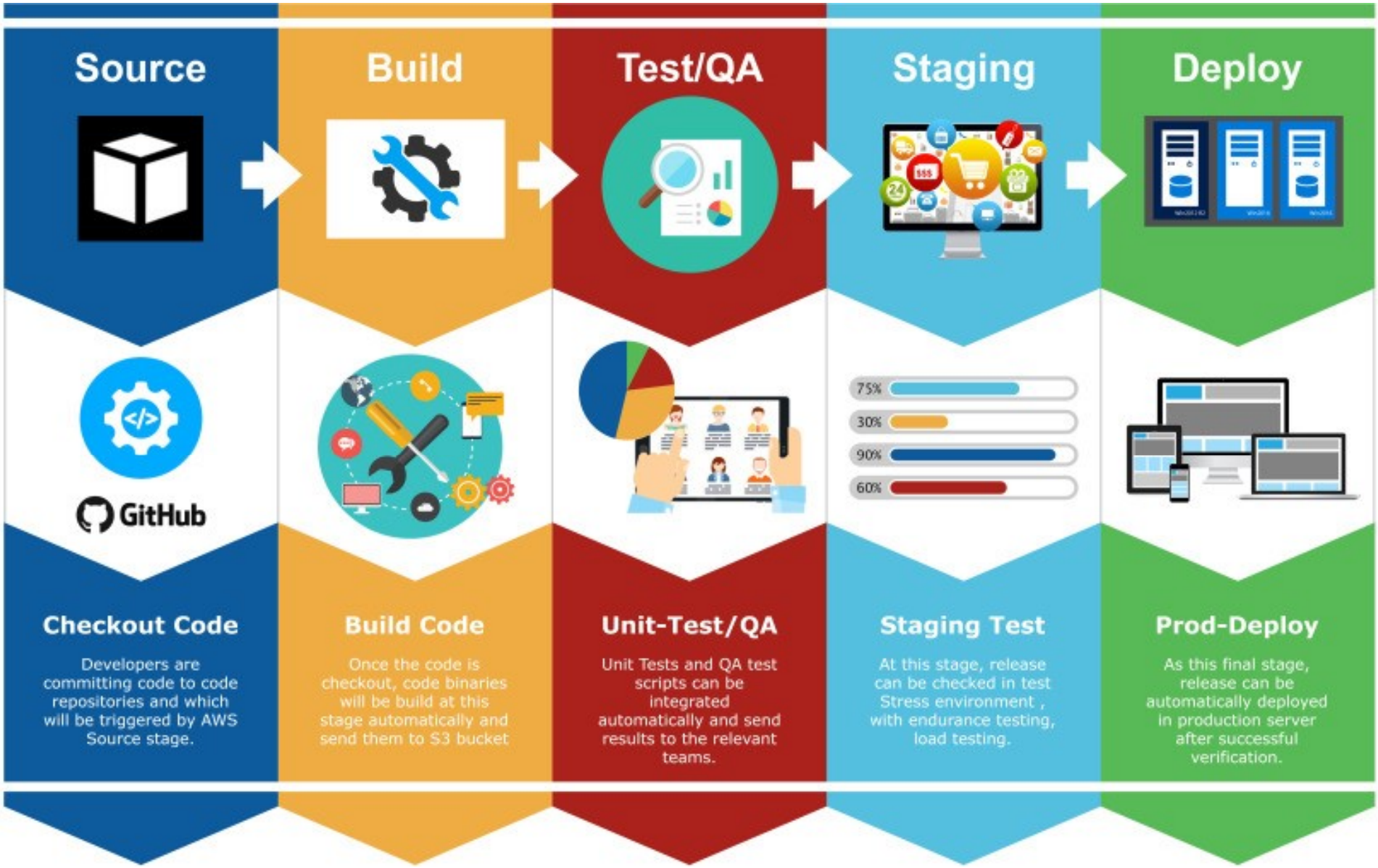
- Continuous Delivery can work only if Continuous Integration is in place. It involves running extensive regression, UI, and performance tests to ensure that the code is production-ready. It's applicable to both bug fixes and new feature releases

-

DevOps – benefits of Continuous Delivery

- **Smaller maintenance and support teams:** Apart from developers, product teams generally maintain several parallel maintenance teams. In the traditional model, each team consisting of build/integration/test members is dedicated to a release version. However, in the automated CD process, these tasks happen without human intervention without compromising product quality.
- **Reducing risk of failure for customer releases:** CD aims to make production changes safe and routine. Deploying to production becomes a boring non-event that can be performed as often as needed. Since the magnitude of code change in each release is very small, risk of failure is also minimised.
- **Enforces process discipline in the team:** Since code is always meant to be production ready, team members are constantly tuning their code, refactoring, reviewing, automating repetitive build tasks, reworking test scripts and getting feature-related documentation done without any delays.
-

DevOps – Continuous Delivery process



DevOps – CHALLENGES of Continuous Delivery

- **Open-ended research projects:** Since such projects generally deal with new, experimental technologies that may not have a customer identified yet, production code is not meant for customer deployment. Team might still implement Continuous Integration, but may not have a system testing environment yet. CD can be done later in such cases.
- **Large legacy projects with low test automation:** Projects with high level of manual interventions (approval hierarchies, security checklists) built into their system and processes will find it difficult to migrate to the CD paradigm.
- **System architecture limitations:** Large monolithic architectures are difficult to automate, scale or debug. In contrast, products with modular and scalable architecture are conducive to CD. Impacted functions are easy to localise and manage every time there's a code update. Microservices, SOA, web applications are good examples of CD success stories.
- **Open-source collaborative development:** With several peer developers working in parallel, open-source platforms require an equally robust build configuration and test automation team. Otherwise, it becomes difficult to synchronise their work, making implementation of CD difficult.
-

DevOps – Continuous Deployment

- Continuous Delivery/Deployment (CD) is **a process by which an application is delivered to various environments, such as testing or production, once someone (usually a product owner/manager) decides that it is ready to go**.

- **Continuous Delivery** is the automation of steps to safely get changes into production. Where **Continuous Deployment focuses on the actual deployment, Continuous Delivery focuses on the release and release strategy**

- **Continuous deployment** (CD, or CDE) is a strategy or methodology for software releases where any new code update or change that makes it through the rigorous automated test process is deployed directly into the live production environment where it will be visible to customers.

DevOps – Continuous Deployment benefits

1. Maintain Capability for Quick New Releases

The most important feature of continuous deployment is that is enables developer teams to get their new releases into the production environment as quickly as possible. Most software companies can no longer rely on development methodologies that were common when developers released software updates once per year. Some companies are rolling out up to **10 deployments per day** with continuous deployment.

2. Enable a More Rapid Feedback Loop with Customers

More frequent updates to your application means a shorter feedback loop with the customer. Using state-of-the-art monitoring tools, developer teams can assess the impact of a new change on user behavior or engagement and make adjustments accordingly. The ability to rapidly release changes is an asset when customer behavior indicates the need for a quick pivot or change in strategy.

DevOps – Continuous Deployment benefits

3. Reducing Manual Processes with Automation

- Continuous deployment is defined by its use of automation in the application deployment process. In fact, continuous deployment wants developers to automate the entire software development process to the greatest extent possible, especially when it comes to release testing. Automation doesn't just help developers push out new releases faster, they actually spend less time on manual processes and get more work done.

- Continuous deployment also takes away release day pressure so developers' sole focus is on building the software, which automatically goes live in a matter of minutes.
- It also heightens productivity and allows developers to respond to rapidly shifting market demands.
- Continuous deployment, however, requires a high degree of monitoring and maintenance to ensure it continues to run smoothly as well as a great capacity to respond to changes quickly.

DevOps – Continuous Deployment vs Continuous Release - differences

difference between Continuous Delivery and Continuous Deployment:

A) Continuous Delivery is a manual task, while Continuous Deployment is an automated task

B) Continuous Delivery has a manual release to a production decision, while Continuous Deployment has releases automatically pushed to production.

C) Continuous Delivery includes all steps of the software development life cycle, Continuous deployment may skip a few steps such as validation and testing.

D) Continuous Delivery means complete delivery of the application to the customer, Continuous Deployment includes the only deployment of the application in a customer environment.

DevOps – Continuous Deployment vs Continuous Release

- a major difference between the two. **With continuous delivery, someone will need to approve the release of the feature to production but with continuous deployment, this process happens automatically upon passing automated testing.**
- Continuous deployment is basically when teams rely on a fully-automated pipeline. This practice fully eliminates any manual steps and automates the entire process. Therefore, continuous deployment ensures that code is continuously being pushed into production.
- Real-time monitoring, however, would still be needed to track and address any issues that arise during the automated tests and to ensure that the builds passed these tests.

DevOps – CI/CD and CD Process

- Continuous deployment is part of a continuous process. The first step starts with continuous integration when developers merge their changes into the trunk or mainline on a frequent basis. This helps teams evade what is known as 'merge hell', which happens when developers attempt to merge several separate branches to the shared trunk on a less regular basis.
- **Continuous deployment goes one step further and combines continuous integration and continuous delivery to make software automatically available to users without any human intervention.**
- Hence, continuous deployment requires the complete automation of all deployments and so refers to the process of automatically releasing developers' changes from the repository to production, bypassing the need for developer approval for each release. Consequently, this process relies heavily on well-designed test automation.
- Then comes continuous delivery when code is deployed to a testing or production environment. Here, developers' changes are uploaded to a repository, where they are then deployed to a production environment. With continuous delivery, you can decide to release daily, weekly or monthly but it is usually recommended to release as often as possible in small batches to be able to easily and quickly fix any issue that arises.

DevOps – CI/CD

- **Continuous integration (CI):** integrate changes to a shared trunk several times a day.
- **Continuous delivery (CD):** continuous integration then deploy all code changes to production environment; deployment is manual.
- **Continuous deployment (CD):** step further from continuous delivery; automated deployment to production without any need for developer approval.

- CI/CD, visualized as a pipeline, automates the software delivery process by building code, running tests and deploying this code to a live production environment.

DevOps – ci/cd pipeline

DevOps – Continuous Monitoring

- **Continuous monitoring** in DevOps is the process of identifying threats to the security and compliance rules of a software development cycle and architecture. Also known as continuous control monitoring or CCM.
- This is one of the most crucial steps in a DevOps lifecycle and will help to achieve true efficiency and scalability.
- This is an automated procedure that can be extended to detect similar inconsistencies in IT infrastructures.
- Continuous monitoring helps business and technical teams determine and interpret analytics to solve crucial issues, as mentioned above, instantaneously.
- Continuous monitoring or CM is a step towards the end of the DevOps process. The software is usually sent for production before continuous monitoring is conducted.
- CM informs all relevant teams about the errors encountered during the production period. Once detected, these flaws are then looked into by the people concerned.
- DevOps tools for continuous monitoring include **Prometheus, Monit, Datadog, and Nagios.**
- Continuous control monitoring goes a long way to help enterprises acquire data from various ecosystems, which can then be used to take more robust security measures like **threat assessment, quick response to breaches, root cause analysis, and cyber forensics.**
- continuous monitoring keeps a tab and reports on the overall well-being of the DevOps setup
- https://www.browserstack.com/guide/continuous-monitoring-in-devops .

DevOps – Continuous Monitoring - Benefits

- **Better security:** Continuous monitoring can be utilized to automate many security measures. CM analyzes data across the entire ecosystem, giving backend teams a broad spectrum of visibility throughout the environment. This helps identify inconsistencies and trigger events that can lead to security failures.
- System admins can identify security threats and respond to them much faster due to continuous monitoring.
- **Quicker feedback and real-time reports** help security teams avert breach attempts and lessen the aftermath should an attack occur.

- **Performance errors are detected earlier:** Continuous monitoring is flexible in its entry into the software development cycle. **Although CM is traditionally introduced during the production phase, initiating it in staging and testing environments can help determine performance inconsistencies much ahead of time.** The production cycle can thus deal only with more stable releases.

- **System downtime is significantly reduced:** System downtimes are infamous for causing significant disruptions in business operations, where recurring incidents can lead to a loss in revenue.
- Continuous monitoring can assist technical teams to keep a vigilant eye on the **database, network, and applications**, helping to resolve any issues before they give rise to system downtime. Past issues are also evaluated to avoid them in the future and to build more enhanced software solutions.

DevOps – Continuous Monitoring - Benefits

- **Facilitates better performance for the business:** Since continuous monitoring works to alleviate system downtimes, it improves **user experience** **and business credibility**.
- continuous monitoring is used extensively to track user feedback that comes in handy when evaluating new updates and changes to the system.

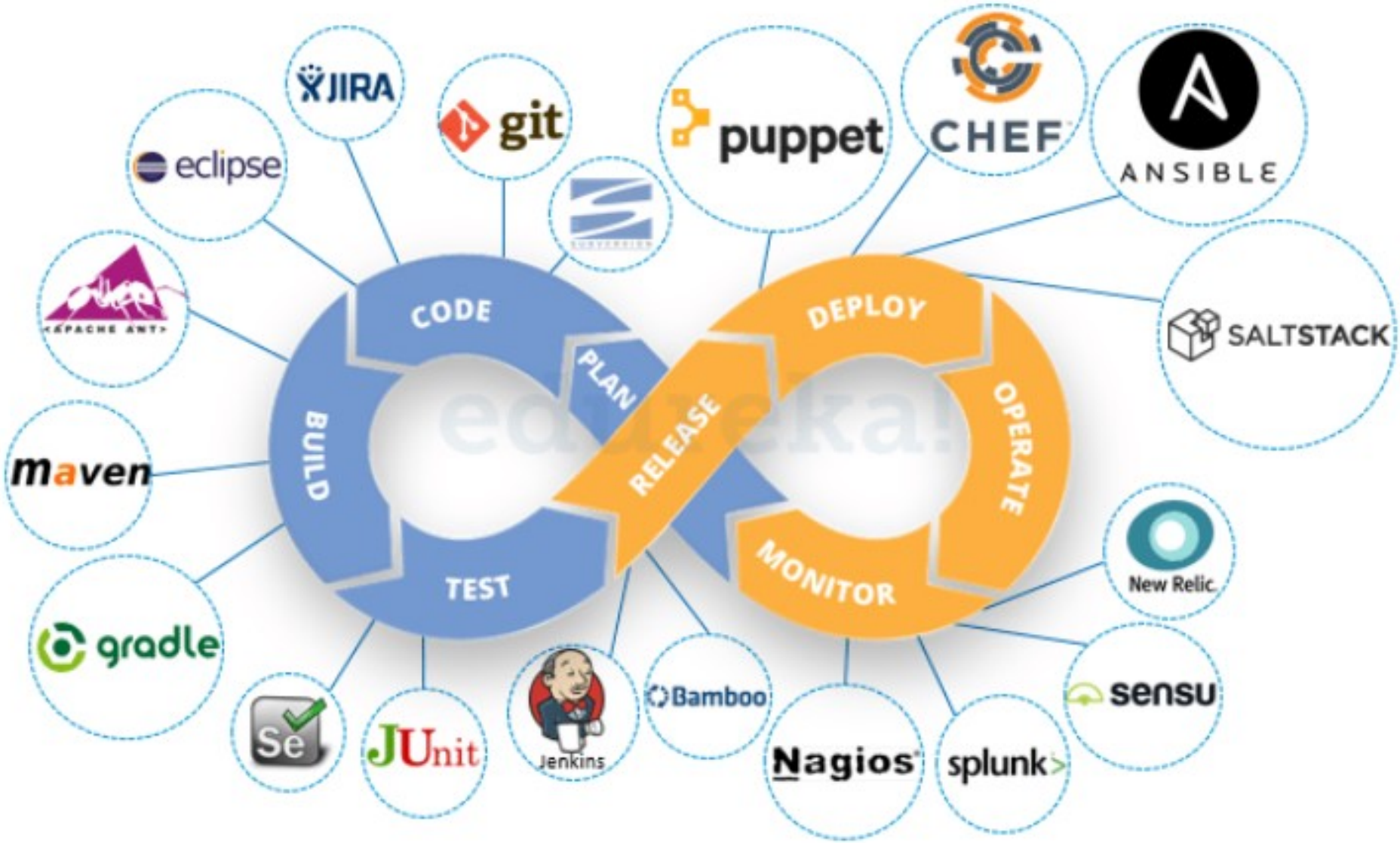
DevOps – Continuous Monitoring - Types

- **Infrastructure Monitoring**: Monitors and manages the IT infrastructure required to deliver products and services. This includes data centers, networks, hardware, software, servers, storage, and the like. Infrastructure Monitoring collates and examines data from the IT ecosystem to improve product performance as far as possible.
- **Tool must monitor:** Server Availability, CPU & Disk Usage, Server & System Uptime, Response Time to Errors, Storage, Database Health, Storage, Security, User permissions, Network switches, Process level usage, Relevant performance trends
- **Tools:** Nagios, Zabbix, ManageEngine OpManager, Solarwinds,Prometheus etc
- **Application Monitoring**: Monitors the performance of released software based on metrics like uptime, transaction time and volume, system responses, API responses, and general stability of the back-end and front-end.
- **Tool must monitor:** availability, error rate, throughput, user response time, pages with low load speed, third-party resource speed, browser speed, end-user transactions, SLA status.

  Tools: Appdynamics, Dynatrace, Datadog, Uptime Robot,Uptrends, Splunk etc
- **Network Monitoring:** Monitors and tracks network activity, including the status and functioning of firewalls, routers, switches, servers, Virtual Machines, etc. Network Monitoring detects possible and present issues and alerts the relevant personnel. Its primary goal is to prevent network downtime and crashes.

  Tool: Cacti, ntop,nmap, Spiceworks, Wireshark,Traceroute, bandwidth Monitor etc.
- **Tool must monitor:** Latency, Multiple port level metrics, Server bandwidth, CPU use of hosts, Network packets flow .

DevOps tools



DevOps – CI/CD Pipeline

- https://www.guru99.com/ci-cd-pipeline.html
- https://www.edureka.co/blog/ci-cd-pipeline/

CI/CD Pipeline:

- A CI/CD pipeline automates the process of software delivery.
- It builds code, runs tests, and helps you to safely deploy a new version of the software.
- CI/CD pipeline reduces manual errors, provides feedback to developers, and allows fast product iterations.
- CI/CD pipeline introduces automation and continuous monitoring throughout the lifecycle of a software product.
- It involves from the integration and testing phase to delivery and deployment. These connected practices are referred as **CI/CD pipeline.**

DevOps – CI/CD Pipeline

- **What is CI/CD?**

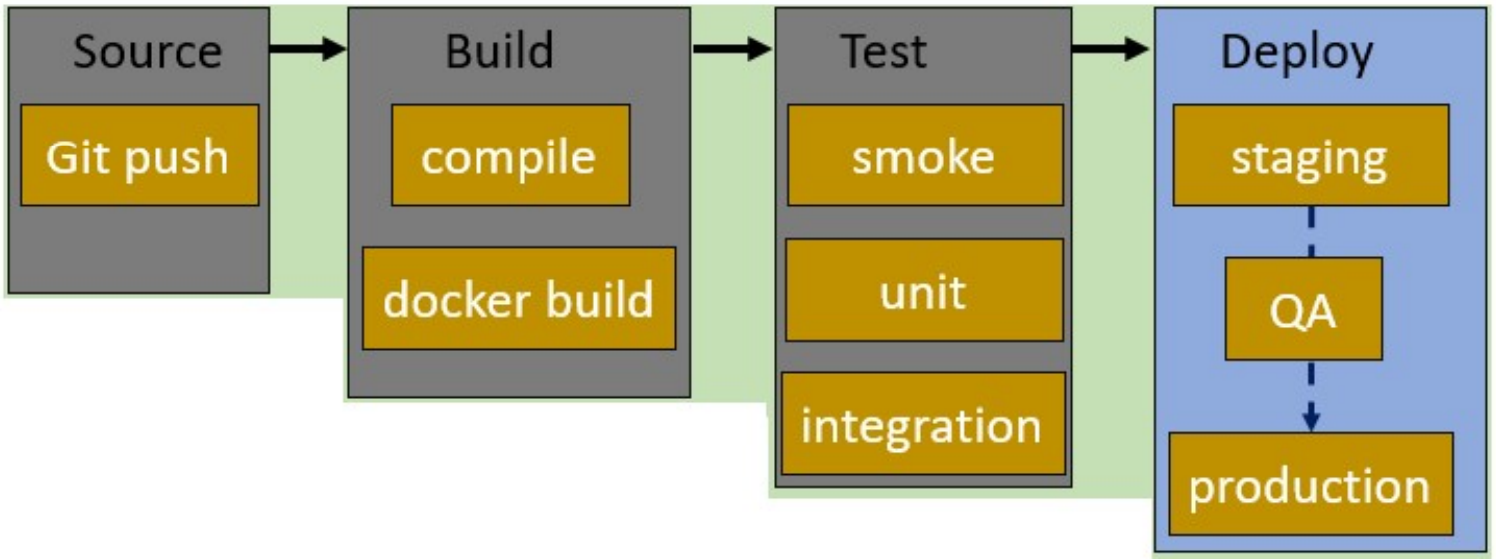
(Continuous Integration / Continuous Delivery //Continuous Deployment):

- **Continuous integration** is a software development method where members of the team can integrate their work **at least once a day**. In this method, every integration is checked by an automated build to search the error.
- **Continuous delivery** is a software engineering method in which a team develops software products in a short cycle. It ensures that software can be easily released at any time. (manual)
- **Continuous deployment** (automated) is a software engineering process in which product functionalities are delivered using **automatic deployment**. It helps testers to validate whether the codebase changes are correct, and it is stable or not.

DevOps – CI/CD Pipeline - stages

- A CI/CD pipeline is a runnable specification of the steps that any developer should perform to deliver a new version of any software. Failure in each and every stage triggers a notification via email, Slack, or other communication platforms. It enables responsible developers to know about the important issues.

Stages of CI/CD pipeline:



DevOps – CI/CD Pipeline - Stages

Source Stage:

- In the source stage, CI/CD pipeline is triggered by a code repository. Any change in the program triggers a notification to the CI/CD tool that runs an equivalent pipeline. Other common triggers include user-initiated workflows, automated schedules, and the results of other pipelines.

Build Stage:

- This is the second stage of the CI/CD Pipeline in which you ***merge the source code and its dependencies***. It is done mainly to build a runnable instance of software that you can potentially ship to the end-user.
- Programs that are written in languages like C++, Java, C, or Go language should be compiled. On the other hand, JavaScript, Python, and Ruby programs can work without the build stage.
- Failure to pass the build stage means there is a fundamental project misconfiguration, so it is better that you address such issue immediately.

Test Stage:

- Test Stage includes the execution of automated tests to validate the correctness of code and the behaviour of the software. This stage prevents easily reproducible bugs from reaching the clients. It is the responsibility of developers to write automated tests.

Deploy Stage:

- This is the last stage where your product goes live. Once the build has successfully passed through all the required test scenarios, it is ready to deploy to live server.

CI/CD Pipeline - Example

- **Source Code Control:** Host code on GitHub as a private repository. This will help you to integrate your application with major services and software.
- **Continuous integration:** Use continuous integration and delivery platform CircleCI and commit every code. When the changes notify, this tool will pull the code available in GitHub and process to build and run the test.
- **Deploy code to UAT:** Configure CircleCI to deploy your code to AWS UAT server.
- **Deploy to production:** You have to reuse continuous integration steps for deploying code to UAT.

CI/CD Pipeline - Advantages

- Builds and testing can be easily performed manually.
- It can improve the consistency and quality of code.
- Improves flexibility and has the ability to ship new functionalities.
- CI/CD pipeline can streamline communication.
- It can automate the process of software delivery.
- Helps you to achieve faster customer feedback.
- CI/CD pipeline helps you to increase your product visibility.
- It enables you to remove manual errors.
- Reduces costs and labour.
- CI/CD pipelines can make the software development lifecycle faster.
- It has automated pipeline deployment.
- A CD pipeline gives a rapid feedback loop starting from developer to client.
- Improves communications between organization employees.
- It enables developers to know which changes in the build can turn to the brokerage and to avoid them in the future.
- The automated tests, along with few manual test runs, help to fix any issues that may arise.

CI/CD Tools

Jenkins: Jenkins is an **open-source Continuous Integration server** that helps to achieve the Continuous Integration process (and not only) in an automated fashion.

- Jenkins is free and is entirely written in Java. Jenkins is a widely used application around the world that has around 300k installations and growing day by day.

Features:

- Jenkin will build and test code many times during the day.
- Automated build and test process, saving timing, and reducing defects.
- The code is deployed after every successful build and test.
- The development cycle is fast.

Bamboo: Bamboo is a continuous integration build server that performs – automatic build, test, and releases in a single place. It works seamlessly with JIRA software and Bitbucket.

Features:

- Run parallel batch tests
- Setting up Bamboo is pretty simple
- Per-environment permissions feature allows developers and QA to deploy to their environments
- Built-in Git branching and workflows. It automatically merges the branches.

CI/CD Tools

- CircleCi is a flexible CI tool that runs in any environment like a cross-platform mobile app, Python API server, or Docker cluster. This tool reduces bugs and improves the quality of the application.
- **Features:**
- Allows to select Build Environment
- Supports many languages including C++, JavaScript, NET, PHP, Python, and Ruby
- Support for Docker lets you configure a customized environment.
- Automatically cancel any queued or running builds when a newer build is triggered.

CI/CD Pipeline

Why Does the CI/CD Pipeline Matter for IT Leaders?

- CI/CD pipeline can improve reliability.
- It makes IT team more attractive to developers.
- CI/CD pipeline helps IT leaders, to pull code from version control and execute software build.
- Helps to move code to target computing environment.
- Enables project leaders to easily manage environment variables and configure for the target environment.
- Project managers can publish push application components to services like web services, database services, API services, etc.
- Providing log data and alerts on the delivery state.
- It enables programmers to verify code changes before they move forward, reducing the chances of defects ending up in production.

CI/CD Pipeline

Summary:

- A CI/CD pipeline automates the process of software delivery.
- CI/CD pipeline introduces automation and continuous monitoring throughout the lifecycle of a software product.
- Continuous integration is a software development method where members of the team can integrate their work at least once a day.
- Continuous delivery is a software engineering method in which a team develops software products in a short cycle.
- Continuous deployment is a software engineering process in which product functionalities are delivered using automatic deployment.
- There are four stages of a CI/CD pipeline 1) Source Stage, 2) Build Stage, 3) Test Stage, 4) Deploy Stage.
- Important CI/CD tools are Jenkins, Bambo, and Circle CI.
- CI/CD pipeline can improve reliability.
- CI/CD pipeline makes IT team more attractive to developers.
- Cycle time is the time taken to go from the build stage to production.
- Development frequency allows you to analyse bottlenecks you find during automation.
- Change Lead Time measures the start time of the development phase to deployment.
- Change Failure Rate focuses on the number of times development get succeeds vs. the number of times it fails.
- MTTR (Mean Time to Recovery) is the amount of time required by your team to recover from failure.
- MTTF (Mean Time to Failure) measures the amount of time between fixes and outages.