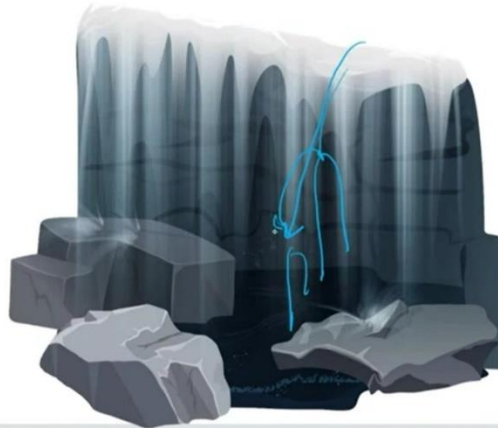


# Waterfall Model

---

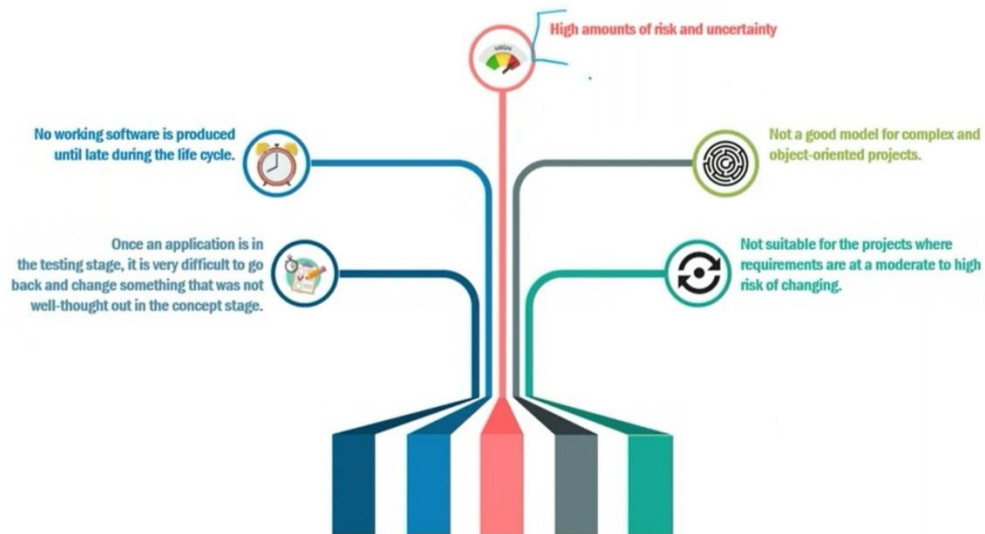
It has distinct goals for each phase of development. Imagine a waterfall on the cliff of a steep mountain. Once the water has flowed over the edge of the cliff, it cannot turn back.



---

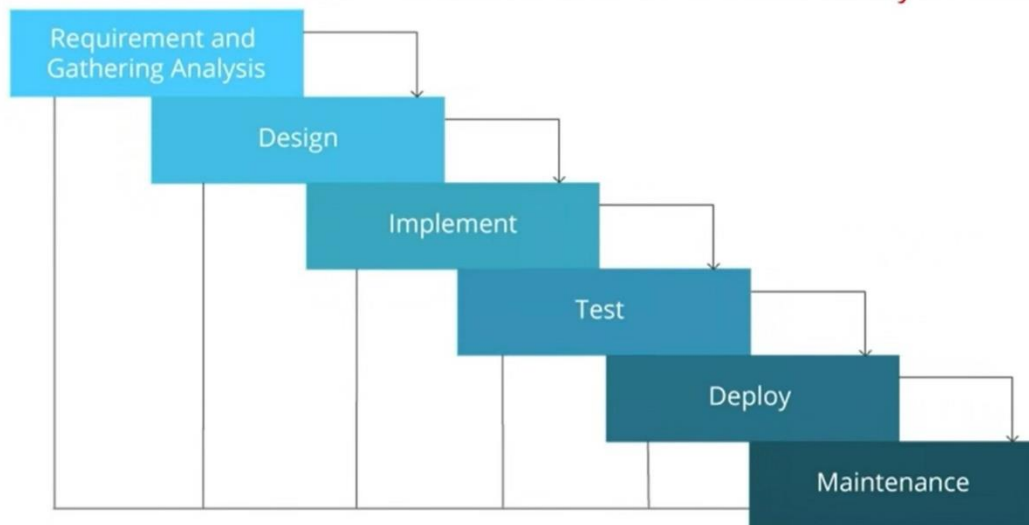
## Limitations of Waterfall Model

---



# Traditional Waterfall Model

SDLC  
methodology



## Waterfall Model Challenges

### Developers



Huge waiting time for code deployment

Pressure of work on old and pending code



# Waterfall Model Challenges

## Operations



Difficult to maintain uptime of the production environment

Tools to automate infrastructure management are not effective



Difficult to diagnose and provide feedback on the product

No. of servers to be monitored increases



## Advantages and Disadvantages of Waterfall Methodology

Advantages		Disadvantages	
For Users	For Employees	For Users	For Employees
Longer operational life	Less need for supporting systems	More expensive to maintain	Requires extensive research into user needs
Meets user needs and requirements	Excellent documentation	More difficult to use	Difficult to customize
More flexible			Requires more training for employees

## Advantages of waterfall model

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.

- Waterfall model works well for smaller projects where requirements are clearly defined and very well understood.

#### Disadvantages of waterfall model

- Once an application is in the **testing** stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

#### When to use the waterfall model

- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements
- Ample resources with required expertise are available freely
- The project is short.

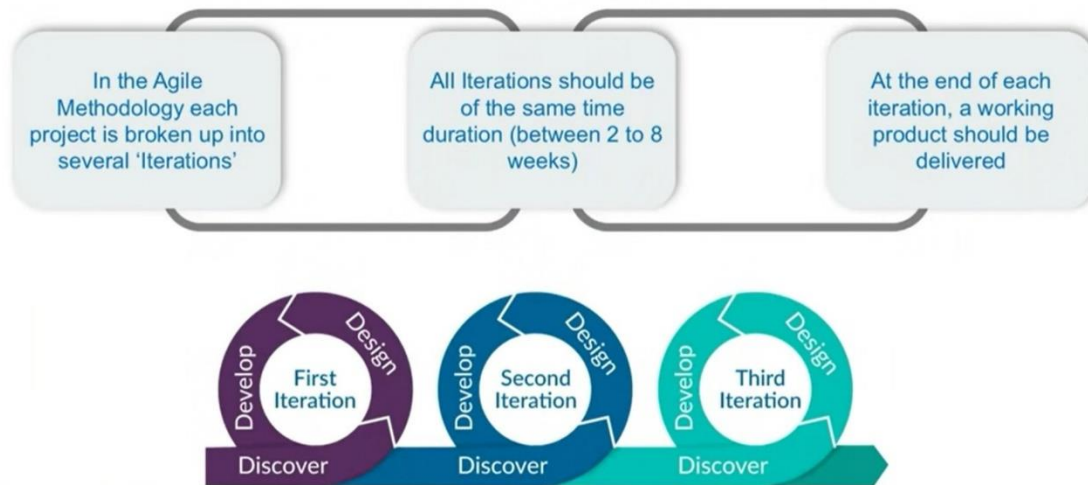
In Waterfall model, very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demonstrated to the end users.

Once the product is developed and if any failure occurs then the cost of fixing such issues are very high, because we need to update everything from document till the logic.

In today's world, Waterfall model has been replaced by other models like iterative, agile etc.

# What is Agile Methodology?

---



## Advantages of Agile model:

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

## Disadvantages of Agile model:

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

## When to use Agile model:

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the **waterfall model** in agile model very limited **planning** is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

## Difference between: Waterfall Model vs Agile Model

There are many differences between Waterfall and Agile model as shown below.

Waterfall Model	Agile Model
Planning – Waterfall model requires planning for long term which requires complete clarity in requirements	Planning in Agile projects is generally on a short term since the work product is delivered in 2 to 4 weeks
Project success is dependent on implementing the requirements closely	Project success is based on delivery of business value to the client
Waterfall projects are driven from the top of a projects hierarchy	Teams are self governing in an Agile project
Sequential planning with clearly defined milestone details and predictability are characteristics of Waterfall projects	Planning in Agile projects is iterative in nature and adapts to changing requirements
There are many roles in a Waterfall project and these projects can also have several levels of hierarchy	Agile projects teams have significantly lesser roles, for example Scrum teams can get by with only three roles
There is significant amount of communication with the user during requirements gathering (at the beginning) and testing (at the end)	There is a steady on-going communication with the user during the project
There is no much dependency on the end users during the development and other intermediate phases	There is significant dependency on the users, all the way through the lifetime of the project
Complete requirements are clearly documented to begin with	Requirements continue to develop over the lifetime of the project and are defined when they are needed, i.e. just in time
Team members with different roles have different responsibility levels	Equal responsibility is shared between the roles
Change control is strictly enforced and rigorous change control processes are followed. Thus change is discouraged and they do not respond well to change.	Agile projects are open to change, they accept them openly and respond well
Quality Control activities like Testing are performed towards the end of the project	Quality Control activities are performed throughout the project
The steps in the processes in Waterfall model are rigorously followed since this model is more process oriented	Agile model is more people oriented and lesser importance is placed on processes with the option to skip those processes whose value is low
Delivery of the project at the end is characterized by a big bang event	Working product features are delivered in each sprint of the project
It is difficult to measure the progress of the project in the middle of the project	Progress of the project can be easily measured since working features are delivered frequently
Progress of the project is generally reviewed with the team once a week	Progress of the project is reviewed with the team on a daily basis in the standup meeting
Have you seen Waterfall model being used in your organization? Please share your experience in the comments below.	



## Development Without DevOps Culture

---



Release and Deploy  
Mismatch



Unpredictable Issues



Blame Game



Lack of monitoring and  
Feedback

## Development With DevOps Culture

---



Streamlined Deliveries



Continuous Monitoring and Feedback



Team Work in Collaboration



# Traditional Development Model Challenges

Consider developing software in a traditional way using a **Waterfall**

model

**Waterfall Model Challenge:** New phase in the development process begins only if the previous phase is complete



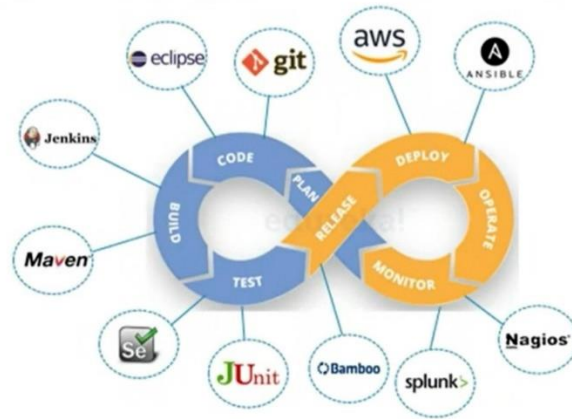
## DevOps: What DevOps Is NOT!!

- DevOps is not a role, person or organization
- DevOps is not a separate team
- DevOps is not a product or a tool
- DevOps is not about just writing scripts or implementing tools



# What Is DevOps?

DevOps is a practice that allows a single team to manage the entire application development life cycle, that is, development, testing, deployment, and monitoring



DevOps is a combination of software development (dev) and operations (ops). It is defined as a software engineering methodology which aims to integrate the work of development teams and operations teams by facilitating a culture of collaboration and shared responsibility.

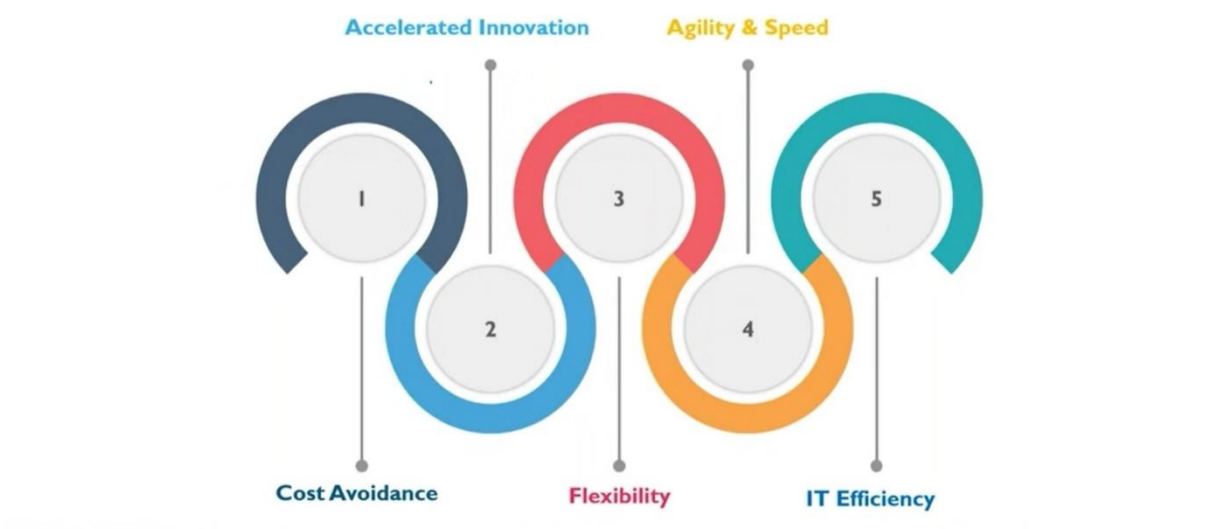
## What Does DevOps Do?

- Integrates developers and operations teams
- Improves collaboration and productivity by:
  - ✓ Automating infrastructure
  - ✓ Automating workflows
  - ✓ Continuously measuring application performance



# Benefits of DevOps

---



## Cultural Transformation

---

- The workplace culture undergoes a major transformation while implementing cultural changes with DevOps.
- It is a long-term process that requires patience and endurance to maintain a positive and transparent atmosphere in the workplace with the changes being implemented.



# Automate Everything

---

- Most of the legacy tools and systems used by organizations may not be conducive to automation and collaboration.
- Automation is essential since continuous testing and development are necessary for smooth deployment.



## Adoption Of Tools

---

- The Dev and Ops teams have separate toolsets and metrics.
- It is necessary to integrate all the tools properly to make testing, deployment, and building all work together in a continuous manner.



Devops Tools:

Discover: Jira Product Discovery, Mural, Miro

Plan: Jira Software, Confluence, Slack

Build: Kubernetes, Docker

Iaas: Ansible, Chef, Docker, Puppet, Terraform

Source control and collaborative coding: BitBucket, GitLab, GitHub

Continuous Delivery: Jenkins, AWS, BitBucket, CircleCI, Sonarsource

Test: XRAY, Zephyr Squad, Zyphyr Scale, Mabl, Snyk, Mend, Veracode, StackHawk

Deploy: Jira Software

Automated Deployment: Bitbucket, AWS CodePipeline

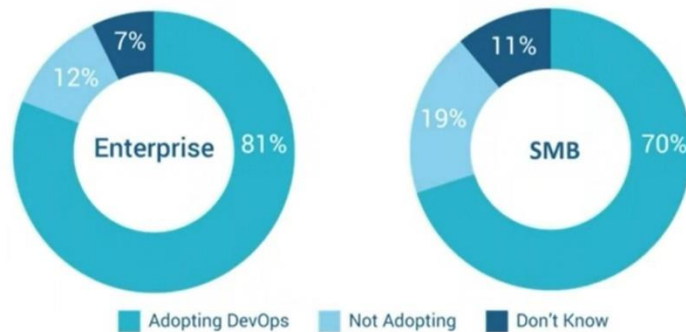
Operate: Jira Service Management, Jira Software, Opsgenie, Statuspage

Observe: AppDynamics, Datadog, Slack, Splunk, New Relic, Opsgenie, Nagios, Dynatrace

## DevOps Current Scenario

---

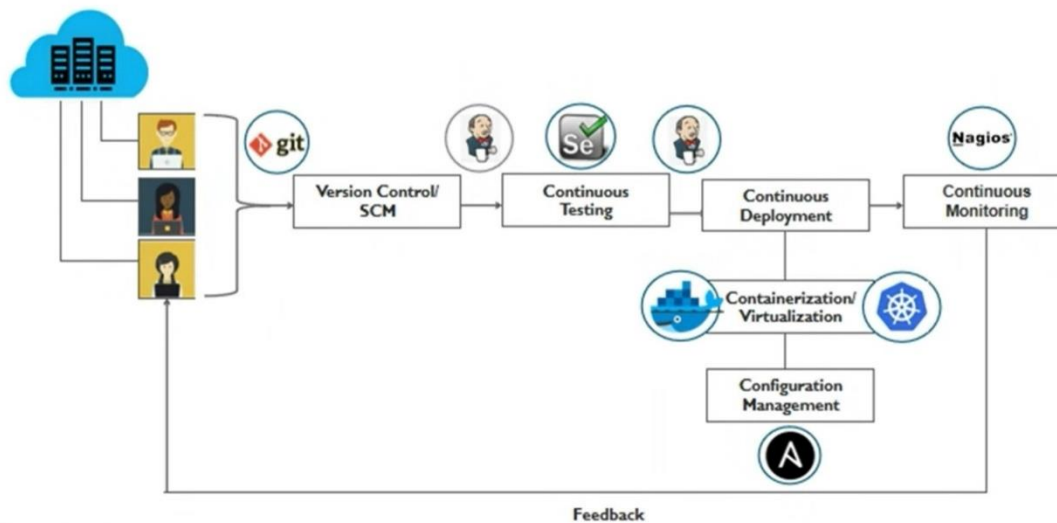
“Considering the changing pace of IT landscape, almost all the companies require fast paced development environment”



# Skills Of A DevOps Engineer

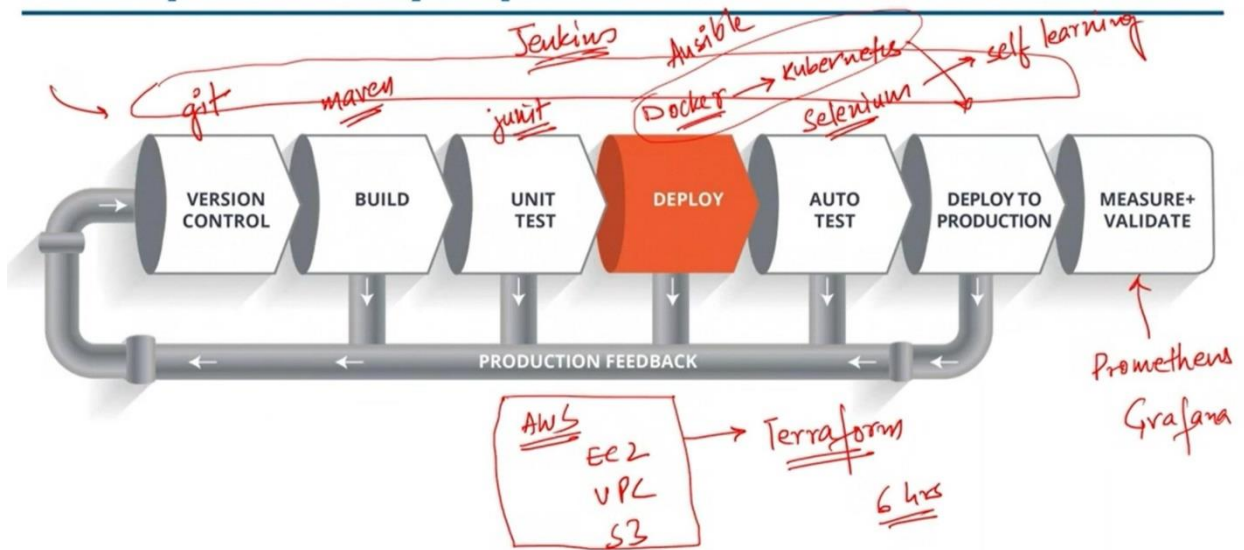
Skills	Description
Tools	<ul style="list-style-type: none"><li>• Version Control - GIT</li><li>• Continuous Integration - Jenkins</li><li>• Virtualization/ Containerization – Docker/Kubernetes</li><li>• Configuration Management – Puppet/Ansible</li><li>• Monitoring – Nagios</li></ul>
Networking Skills	<ul style="list-style-type: none"><li>• General networking skills – Establishing connection between the containers/Port Forwarding/ Container Orchestration</li></ul>
Other Skills	<ul style="list-style-type: none"><li>• People Skills</li><li>• Process Skill</li><li>• Customer Skill and Empathy</li><li>• Cloud Awareness</li></ul>

## DevOps Stages





# DevOps Delivery Pipeline

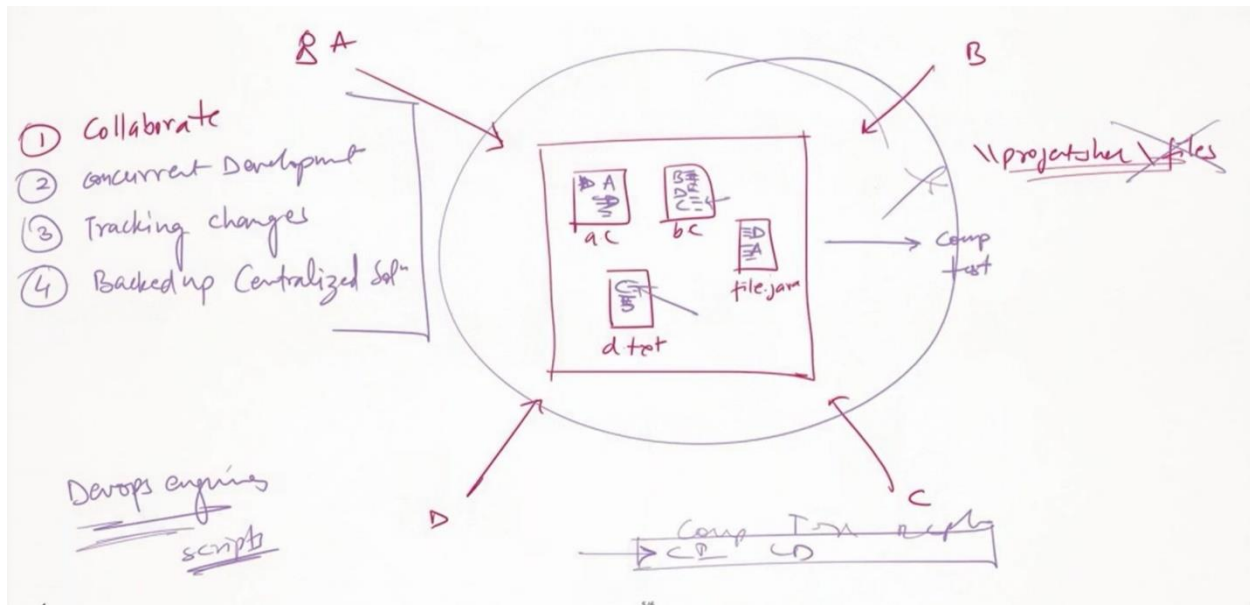


Presentation 3 / 69

## COURSE OUTLINE MODULE 02

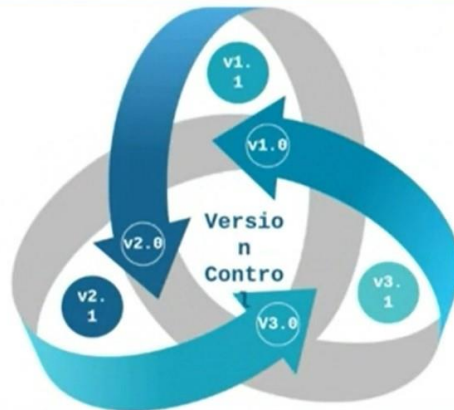
1. Introduction to DevOps	7. Containerization using Docker Part - II
2. Version Control with Git	8. Container Orchestration Using Kubernetes Part-I
3. Git and Jenkins	9. Container Orchestration Using Kubernetes Part-II
4. Continuous Integration with Jenkins	10. Monitoring Using Prometheus and Grafana
5. Configuration Management using Ansible	11. Provisioning Infrastructure using Terraform Part - I
6. Containerization using Docker Part - I	12. Provisioning Infrastructure using Terraform Part - II





## What Is Version Control?

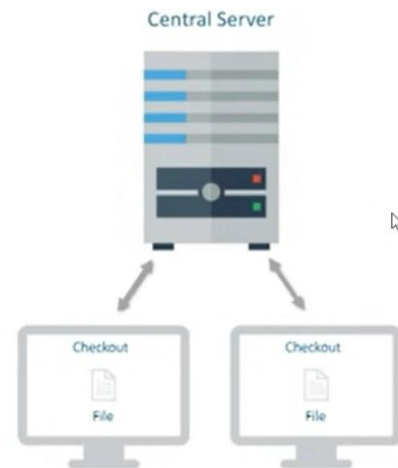
Version Control is a system that documents changes made to a file or a set of files. It allows multiple users to manage multiple revisions of the same unit of information. It is a snapshot of your project over time.



# Version Control Types

## Centralized Version Control (CVC)

- Local Version Control's issues are resolved by Centralized Version Control
- In CVC, a central repository is maintained where all the versioned files are kept
- Now users can checkout, and check-in files from their different computers at any time



# Version Control Types

## Distributed Version Control

- Version Database is stored at every users' local system and at the remote server
- Users manipulate the local files and then upload the changes to the remote server
- If any of the servers die, a client server can be used to restore

