# Git For DevOps Study Material

# INDEX

# Detailed Index

Option-1: --oneline option to get brief log information
Option-2: -n option to limit the number of commits to display
Option-3: --grep option to search based on given pattern in commit message
Option-4: Show commits more recent than a specific time.
Option-5: Show commits older than a specific time
Option-6: Show commits based on author
Option-7: --decorate option to display extra information

Topic-9: The Complete Story of git diff command
Case-1: To see the difference in file content between working directory and staging area
Case-2: To see the difference in file content between working directory and last commit
Case-3: To see the difference in file content between staged copy and last commit
Case-4: To see the difference in file content between specific commit and working directory copy
Case-5: To see the difference in file content between specific commit and staging area copy
Case-6: To see the difference in file content between two specified commits
Case-7: To see the difference in file content between last commit and last butone commit
Case-8: To see the differences in all files content between two specified commits
Case-9: To see the differences in content between two branches
Case-10: To see the differences in content between local and remote repositories

Topic-10: Helix Visual Merge Tool(p4merge) For Checking Differences

Topic-11: Removing files by using git rm command
Case-1: To remove files from working directory and staging area(git rm)
Case-2: To remove files Only from staging area (git rm --cached)
Case-3: To remove files Only from Working Directory (rm command)

Topic-12: Undo changes with git checkout command

Topic-13: Git References ( master and HEAD )

Topic-14: Git reset command
Utility-1: To remove changes from staging area
Utility-2: To undo commits at repository level(--mixed, --soft, --hard modes)

Topic-15: Git Aliases - Providing our own convenient names to git commands
Topic-16: Ignoring unwanted files and directories by using .gitignore file
Topic-17: Any Special Treatment for directories by Git ???

# Topic - 1

# Introduction to DevOps

# Topic - 1: Introduction to DevOps



## 1.1) What is Devops?

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

1) Development Group
2) Non-Development Group or Operations Group or Administrators Group.

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

## 1) Development Group:

The people who are involving

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

are considered as Development Group.

**Eg:**
**Business Analyst(BA)**
**System Analyst(SA)**
**Design Architech(DA)**
**Developers/coders**
**Build Engineer**
**Test Engineers/QA**

## 2) Operations Group:

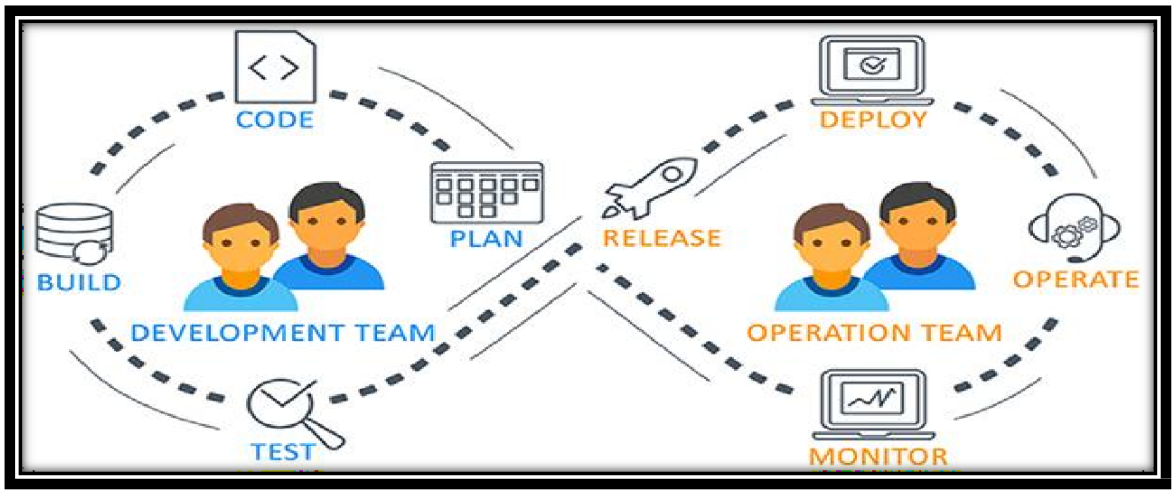
**The people who are involving**

1) **Release**
2) **Deploy**
3) **Operate**
4) **Monitor**

**are considered as Operations Group.**

**Eg:**
**Release Engineers**
**Configuration Engineer**
**System Admin**
**Database Admin**
**Network Admin**
**etc**

**Devops is combination of development and operations.**
**The main objective of devops is to implement collaboration between development and operations teams.**



**To understand this new Devops culture, we have to aware already existing SDLC Models.**

**SDLC → Software Development Life Cycle**

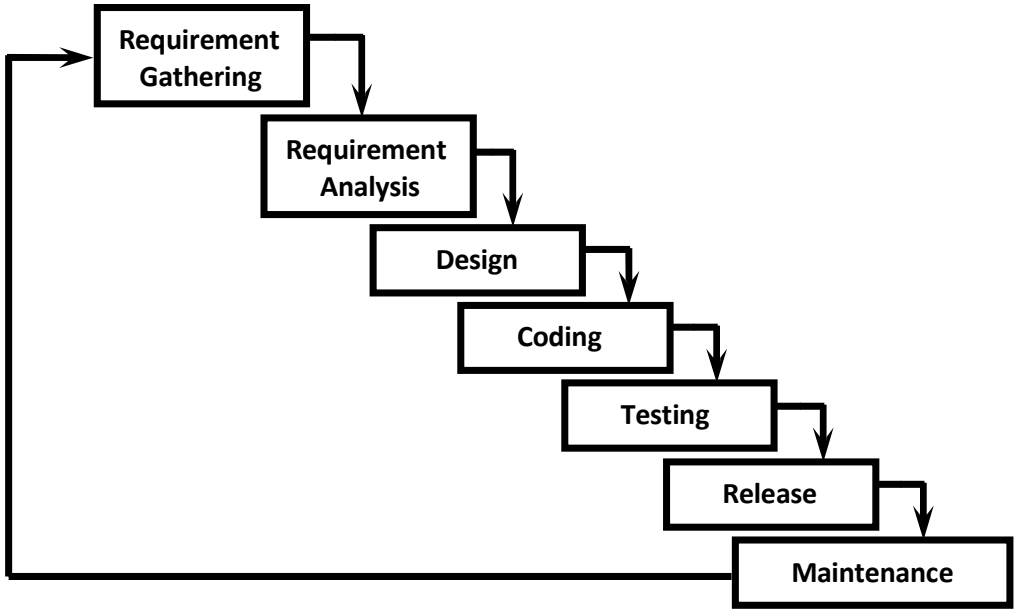
1) **Waterfall Model**
2) **Prototype Model**
3) **Incremental/Iterative Model**
4) **Spiral Model**
5) **RAD Model**
6) **Big-Bang Model**

7) **Fish Model**
8) **V Model**
9) **Agile Model**

10) **Devops Culture**

# 1.2) Water Fall Model:

- **It is the oldest SDLC Model.**
- **It is also known as Linear sequential Model.**
- **In this model,each phase must be completed before the next phase can begin and there is no overlapping of phases. i.e all phases will be performed one by one just like flowing of water fall downwards.**



## Advantages:

1) It is very simple and easy to implement.
2) Phases won't be overlapped and hence there is no ambiguity.
3) All phases will be executed one by one which gives high visibility to the project managers and clients about the progress of the project.
4) Best suitable if the requirements are fixed.
5) Best suitable for small projects.

## Disadvantages:

1) It is very rigid model b'z it won't accept requirement changes in the middle.
2) Client satisfaction is very low because most of the times client will add new requirements in the middle,which won't be supported.
3) Total project development time is more because testing should be done after complementing development only.
4) The cost of bug fixing is very high because we cannot identify bugs in the early stages of life cycle.
5) Not suitable if the requirements keep on changing.
6) Not suitable for large projects.

# 1.3) Agile Model:

This is the most frequently used and hot cake model for software development.

Agile Model is divided into several sub models

1) Rational Unify Process (RUP)
2) Adaptive Software Development (ASD)
3) Feature Driven Development (FDD)
4) Crystal Clear
5) Dynamic Software Development Method (DSDM)
6) Extream Programming (XP)
7) Scrum

etc

Among all these models Scrum model is the most popular and frequently used model.
Scrum is derived from Rugby Game.

- It is light weight process.
- It is an iterative /incremental model and it accepts changes very easily.
- It is people based model but not plan based model.
- Team Collaboration and Continuous feedback are strengths of this model.

# 1.4) Water fall vs Scrum:

1) In water fall model ,before starting next phase,the previous phase should be completed. It is very rigid model and won't accept requirement changes in the middle.
2) But scurm model is not linear sequential model. It is iterative model. Total software will be developed increment by increment and each increment is called a sprint. Sprint is a deliverable/shippable product in scrum model.

## Points to Remember:

1) Scrum is an agile model that allows us focus on delivering highest quality software in shortest time.
2) In this model software developement follows increment by increment
3) Each increment will take one to 3 weeks duration.
4) 7 to 9 members are responsible in every sprint.

The art of doing the twice work in half time is nothing but scrum model → Juff sutherland

## Advantages of Scrum Model:

1) There is maximum chance for quality
2) It ensures effective use of time and money
3) Requirement changes will be accepted so that maximum chance for client satisfaction
4) There is a possibility for the client involvment in every stage.

5) Project status Tracking is very easy
6) Team gets complete visibility through scrum meetings.

## Limitations:

1) The chances of project failure is very high if individuals are not committed or cooperative
2) Adapting scrum model for large teams is very big challenge
3) Must required experienced and efficient team members
4) If any team member leaves in the middle of project,it can have a huge negative impact on the project.

# 1.5) Devops vs Agile Models:

Devops and Agile, both are not same.

## Similarities:

1) Both are software development methodologies. Agile is there in the market for the last 20 years, but devops is recent methodology.
2) Both models concentrating on rapid development of software project.

## Differences:

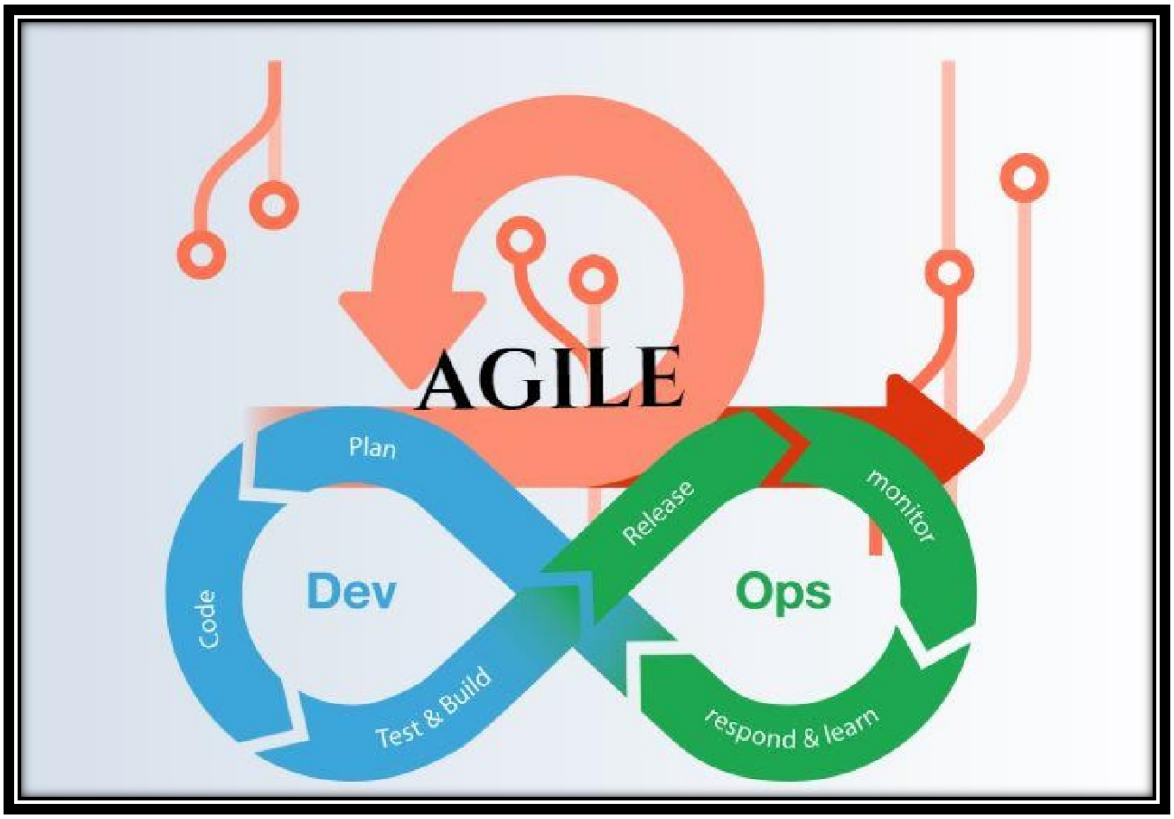
1) The differences between these models will starts after development of the project.
   Agile methodology always talks about software development,testing and deployment.
   Once deployment completed agile methodology has no role.
   But Devops model will continue after deployment also and it is also responsible for operations and monitoring.

2) In Agile Model, separate people are responsible for developing, testing, and deploying the software. But, in DevOps, the DevOps engineer is responsible for everything; development to operations, and operations to development.

3) Agile model won't force us to use automation tools.
   But devops model is completely based on automation.

4) Agile model always giving highest priority for speed, where as Devops giving proirity for both speed and automation.

5) In Agile, client is responsible to give the feedback for the sprint. But in Devops, immediate feedback is available from the monitoring tools.




## What is Devops?

**Devops is not a new Tool/Technology in the market.**

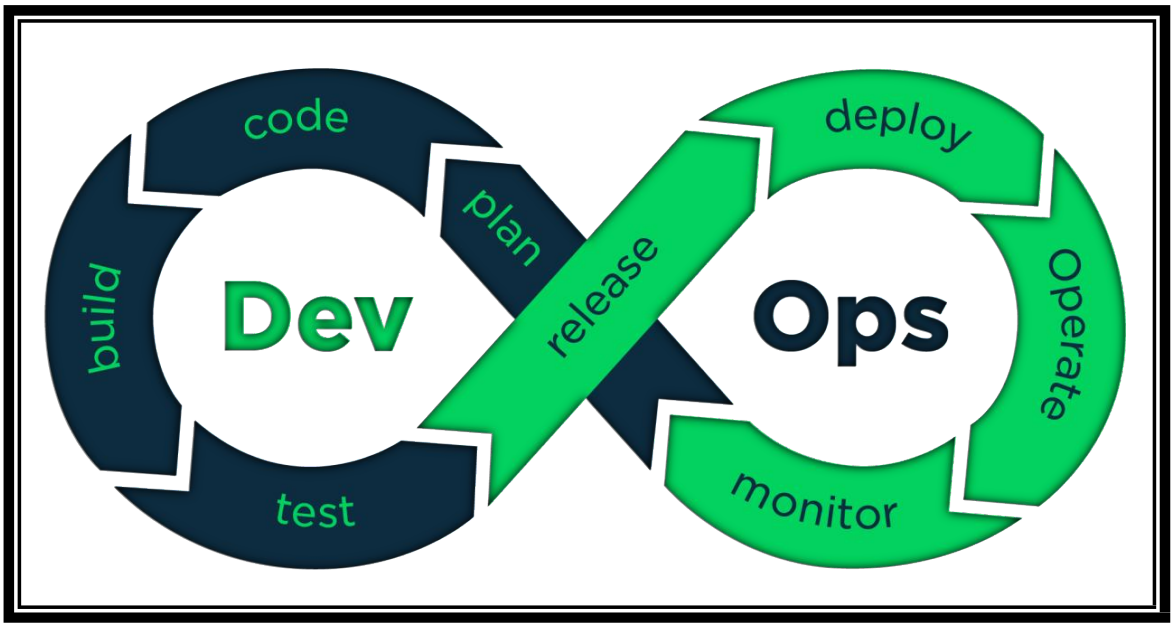
**It is a new culture or process to develop,release and maintain software products/projects/applications with high quality in very faster way with automation tools.**

**Devops is combination of development and operations.**
**The main objective of devops is to implement collaboration between development and operations teams.**

**It is the process of continuous development, continuous build, continuous test, continuous release of the software with high quality in very faster way with automation tools.**

# 1.6) Top Important points about DevOps:

1) Devops is not a new Tool/Technology in the market.
2) It is a new culture or process to develop,release and maintain software products.
3) DevOps is combination of Development and Operations.
4) The main objective of devops is to implement collaboration between development and operations teams.
5) The beauty of DevOps is everything is automated and we can use several automation tools for development and operations.
6) Devops Engineer is All Rounder. He should aware everything. Hence his role is considered as Devops Generalist.
7) Devops is not Agile model and it is more than that because it covers both Development and operations, where as Agile covers only Development but not operations.
8) Devops Cycle is an Infinite Loop where everything is continuous.



# TOPIC – 2

# Introduction to Version Control System

# Topic-2: Introduction to Version Control System



Version Control System is also known as Software Configuration Management (SCM) OR Source Code Management (SCM) System.

## 2.1) Need of Version Control System?

Being a developer we have to write several files which contains source code.

Developer → Write Code → Files

Client gave requirement to Durga to develop a project

client project
  |--100 files developed
  |- client suggested some changes
  |- I changed some files source code to meet client requirement
  |- I gave the demo and client suggested some more changes
  |- I changed some files source code to meet client requirement
  |- I gave demo 3rd time
  |- Client asked for first version only
  |- My Face with big ????

We should not overwrite our code.
Every version we have to maintain.

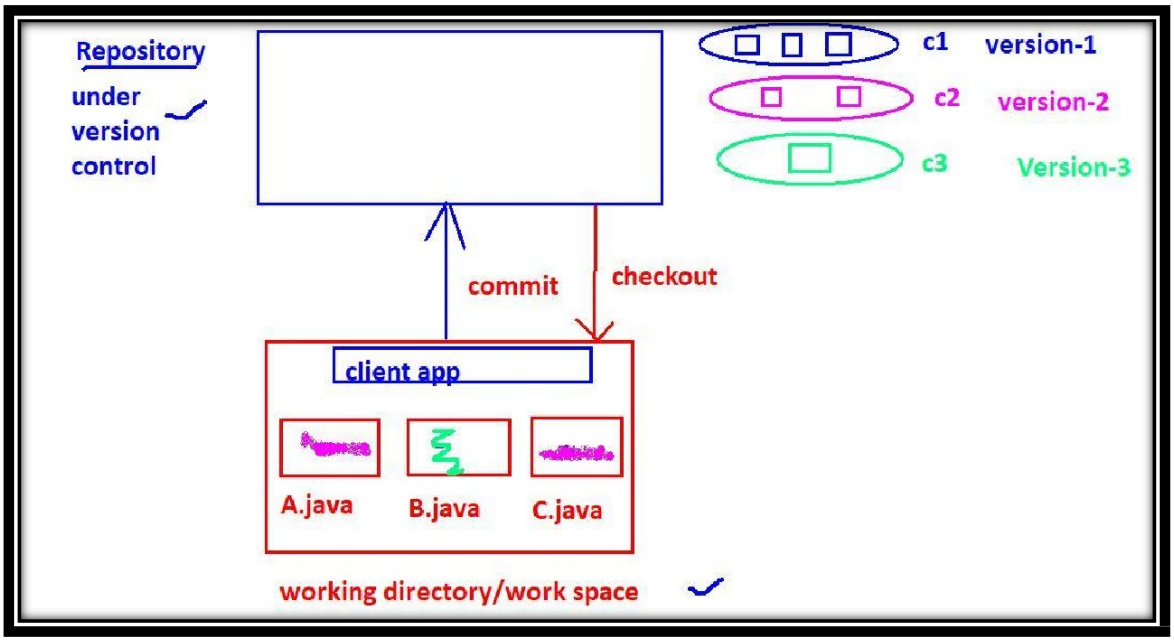
1) Maintaining multiple versions manually is very complex activity.
2) Dev-A and Dev-B working on the code. At last we have to merge the code developed by both developers and we have to deliver to the client. If both developers developed a file named with Util.java, then one copy will overwrite with another copy, which creates abnormal behaviour. We should not overwrite our code.

2) Every change should be tracked like
   who did the change
   when he did the change
   which changes he did etc
   and all changes should be maintained.
3) Overwriting of the code should not be happend.
4) Developers have to share their code to peer developers, so that multiple developers will work in collaborative way.
5) Parallel development must be required

## 2.2) How Version Control System will work?

Version control system always talks about files which contain source code.
Everyone required version control system to maintain different versions of their document.

tester → To maintain different versions of test script
Architect → To maintain different versions of Documents
Project Manager → To maintain different versions of Excel sheets
etc



## 2.3) The Basic Terminology of Version Control System:

### Working Directory:

**Where developers are required to create/modify files.**
**Here version control is not applicable. Here we won't use the work like version-1, version-2 etc**

### Repository:

**Where we have to store files and metadata.**
**Here version control is applicable.**
**Here we can talk about versions like version-1, version-2 etc**

### Commit:

**The process of sending files from working directory to the repository.**

### Checkout:

**The process of sending files from repository to working directory.**

## 2.4) Benefits of Version Control System:

1) **We can maintain different versions and we can choose any version based on client requirement.**
2) **With every version/commit we can maintain metadata like**
   - **commit message**
   - **who did changes**
   - **when he did the change**
   - **what changes he did**
3) **Developers can share the code to the peer developers in very easy way.**
4) **Multiple developers can work in collaborative way**
5) **Parallel development.**
6) **We can provide access control like**
   - **who can read code**
   - **who can modify code**

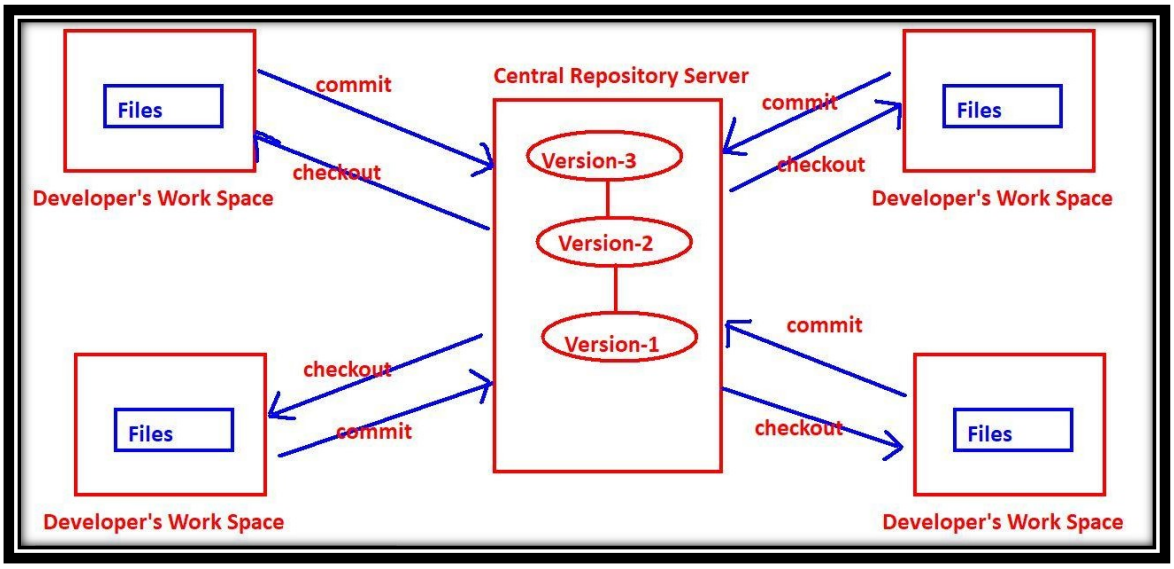
## 2.5) Types of Version Control Systems:

**There are 2 types of VCSs**

1) **Centralized Version Control System**
2) **De Centralized/Distributed Version Control System**

## 2.5.1) Centralized Version Control System:

**The name itself indicates that, this type contains only one central repository and every developer should be connected to that repository.**
**The total project code will be stored in the central repository.**
**If 4 developers are there, still we have only one repository.**




**This type of VCS is very easy to setup and use.**

**Eg: CVS, SVN, Perforce, TFS, Clearcase etc**

## Problems with Centralized VCSs:

1) **Central Repository is the only place where everything is stored, which causes single point of failure. If something goes wrong to the central repository then recovery is very difficult.**

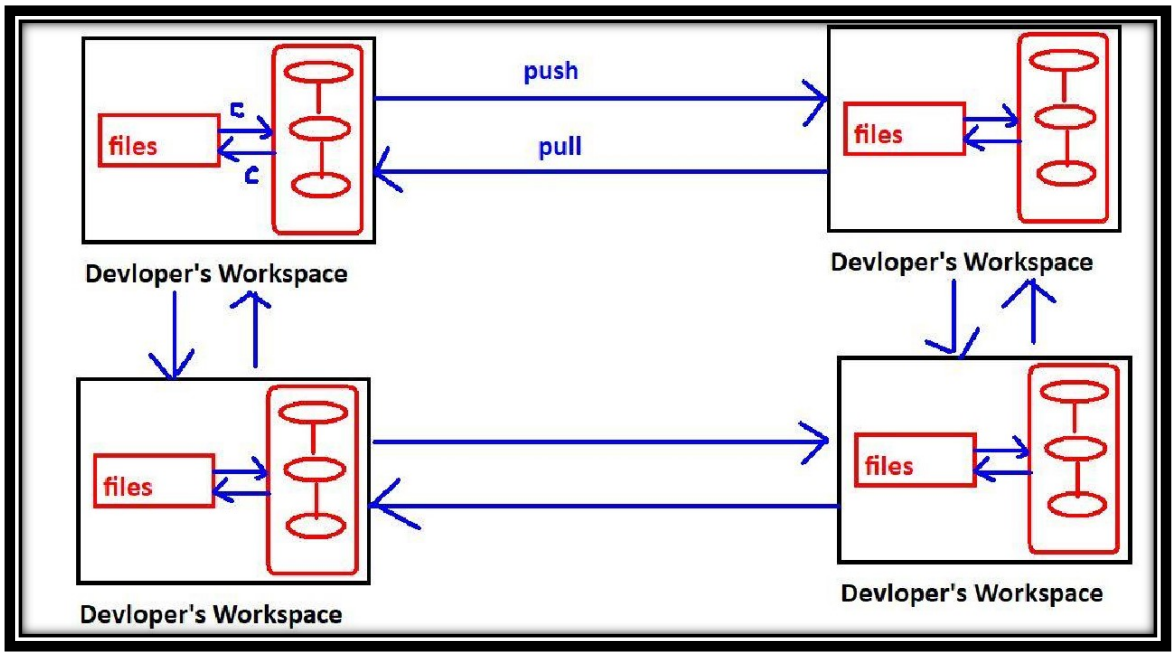
2) **All commit and checkout operations should be performed by connecting to the central repositoty via network. If network outage, then no version control to the developer. i.e in this type, developer work space and remote repository server should be connected always.**

3) **All commit and checkout operations should be performed by connecting to the central repositoty via network and hence these operations will become slow, which causes performance issues. No local operations and every version control operation should be remote operation.**

4) **Oranization of central repository is very complex if number of developers and files increases.**
   **etc**

## 2.5.2) Distributed Version Control Systems:

**The name itself indicates the respository is distributed and every developers workspace contains a local copy of the repository. There is no question of central repository.**




**If 4 developers are there then 4 repositories will be there.**

1) **The checkout and commit operations will be performed locally. Hence performance is more.**
2) **To perform checkout and commit operations network is not required. Hence if there is any network outage, still version control is applicable.**
3) **If something goes wrong to any repository there is a chance to recover. There is no question of single point of failure.**
4) **To perform push and pull operations network must be required, but these operations are not most common operations and we are performing very rarely.**

### Note:

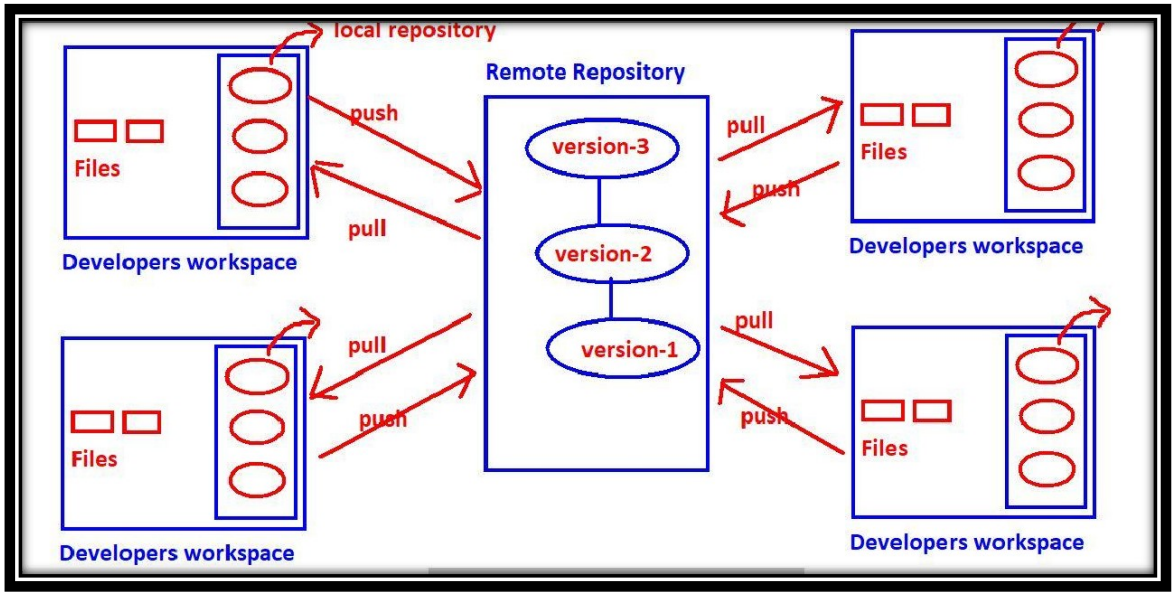
1) **commit and checkout operations will be performed between workspace and repository.**

   **work space – commit → Repository**
   **Repository – checkout → workspace**

2) **push and pull operations will be performed between repositories.**

   **one repository ---push → other repository**
   **one repository ← pull --- other repository**

## Distributed VCS with Remote Repositoty:




## Remote Repository is not Central Repository:

1) **Every developer has his own local copy of repository. It is not centralized and it is distributed only.**
2) **commit and checkout operations won't be performed on remote repository and these will be performed on local repository only.**

**The main job of remote repository is just to share our work to peer developers.**

**High availability, Speed and there is no single point of failure are main reasons for popularity of this model.**

**Eg: Git, Mercurial, Fossil**