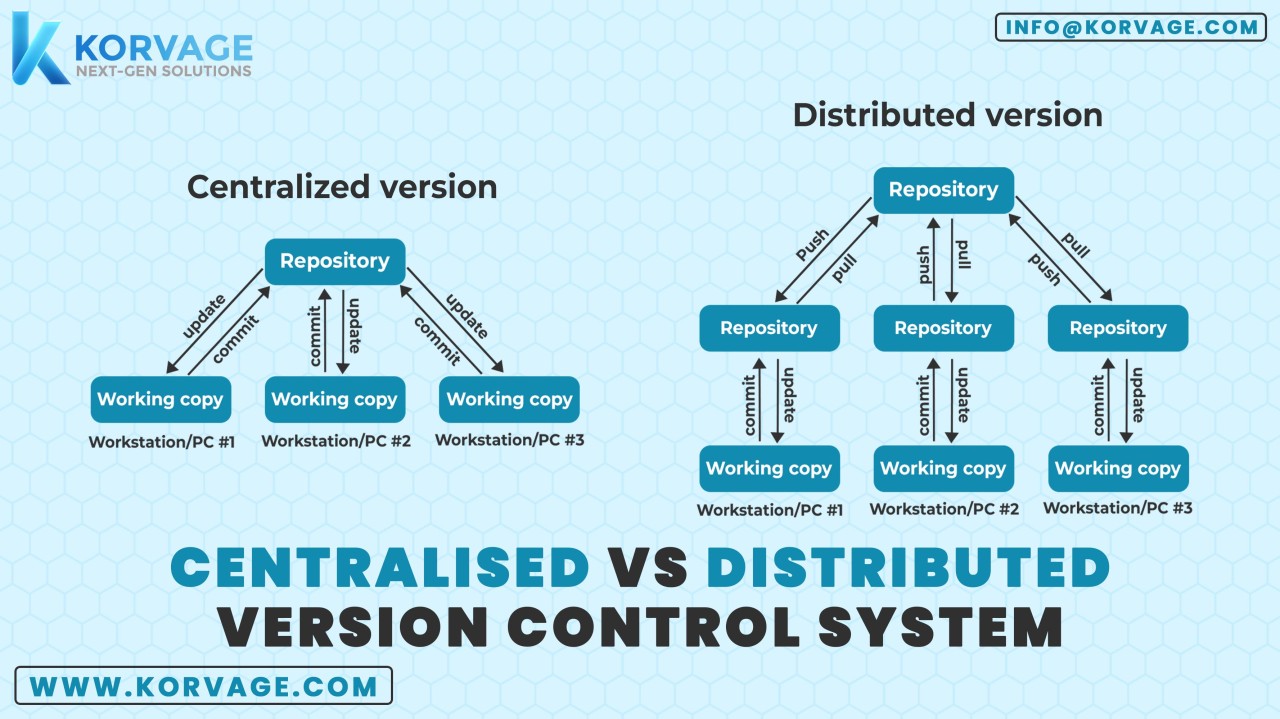
**Explained about VCS/SCM?**

   VCS could refer to Version Control System, which is a software that tracks changes to files over time.

This allows developers to collaborate and go back to previous versions of files without having to manage them manually.

   VCS can be used for many types of files, including software source code, images, and layouts.



**Centralized Version Control (CVC)**

In centralized version control, a single, central server hosts the entire codebase. Each team member connects to this server to check out and commit changes.

**Key Features:**

* **Single Source of Truth**: The central server contains the only complete copy of the repository.
* **Client-Server Model**: Developers (clients) connect to the central server, check out files, and commit changes.
* **Reliance on Internet Connection**: Since the central server stores the codebase, developers need an internet connection to commit or update code.

**Common Tools:**

* Subversion (SVN)
* Perforce
* TFS (Team Foundation Server)

**Advantages:**

* **Simple to Manage**: Easy to control as everything is in one place.
* **Controlled Access**: Provides a single point for setting permissions and managing access.
* **Good for Small Teams**: Works well for small teams where the single-server model simplifies version control.

**Disadvantages:**

* **Single Point of Failure**: If the central server goes down, no one can commit changes until it’s back online.
* **Performance Issues**: Can be slower as all actions need to interact with the central server.
* **Limited Offline Work**: Developers need a connection to the server for version control tasks.

**Distributed Version Control (DVC)**

In distributed version control, each developer has a full copy of the repository, including its entire history. Changes can be made and saved locally, then synchronized with a central server or directly with other repositories when needed.

**Key Features:**

* **Local Repositories**: Each user has a complete copy of the repository with its entire history.
* **Offline Capabilities**: Developers can work offline, committing locally and synchronizing changes with the central repository later.
* **Peer-to-Peer Collaboration**: Teams can collaborate directly by sharing commits between each other's repositories.

**Common Tools:**

* Git
* Mercurial
* Bazaar

**Advantages:**

* **High Availability**: Since every developer has a full copy of the repository, work continues even if the central server is down.
* **Supports Offline Work**: Developers can commit locally and later push or pull changes when they have a connection.
* **Better Collaboration**: It enables features like branching, merging, and pull requests, allowing more flexible workflows.

**Disadvantages:**

* **Complex for Beginners**: The decentralized model requires more understanding of concepts like branching, merging, and remote repositories.
* **Larger Disk Space Usage**: Each user stores a full copy of the repository, which might require more disk space compared to CVC systems.

**Key Differences Between CVC and DVC**

| **Feature** | **Centralized Version Control (CVC)** | **Distributed Version Control (DVC)** |
| --- | --- | --- |
| **Repository Structure** | Single central repository | Each user has a full local repository |
| **Offline Work** | Limited | Fully supported |
| **Performance** | Dependent on server availability | Works offline; faster for local ops |
| **Single Point of Failure** | Central server downtime affects all users | No central dependency; work can continue |
| **Complexity** | Simpler to learn | More complex but flexible |

**Distributed Version Control Systems (Distributed VCS)**

Distributed VCS (DVCS) is a type of version control where the entire repository, including its full history, is mirrored on every user's local machine.

**Key Features:**

    Full Repository on Every Client: Each developer has a full copy of the project, including all its history. This allows for operations like commits, diffs, and branches to be done locally without needing to connect to a central server.

    Decentralization: There's no need for a central server, although one can be used for coordination. This enhances redundancy and resilience since the repository exists in multiple locations.

    Branching and Merging: Distributed VCS typically makes branching and merging easier and more efficient, allowing for more flexible development workflows.

    Difference Between Git and GitHub

**Git:**

What It Is: Git is a distributed version control system that helps track changes in source code during software development.

**Functions:**

        Manages code history.

        Tracks changes and versions.

        Supports branching, merging, and collaboration.

**Usage**: Installed locally on a developer’s machine, Git is used via the command line or a GUI. It’s independent and doesn’t require an internet connection for most operations.

**Key Features:** Speed, simplicity, strong support for non-linear development (thousands of parallel branches), and distributed nature

**GitHub:**

    What It Is: GitHub is a web-based platform built on top of Git that provides hosting for Git repositories.

**Functions:**

        Centralized hosting for Git repositories.

        Collaboration tools (pull requests, code reviews, discussions).

        Integrations with CI/CD pipelines, project management tools, and more.

**Usage:** Developers push their Git repositories to GitHub to collaborate with others, share code, and manage open-source projects.

**Key Features:** Social features like following users, starring repositories, and forking projects, as well as project management tools like issue tracking and wikis.

Create GITHUB account

Download git bash

create repository in github [remote repository]

git cmds

---------

 Getting Started - First-Time Git Setup

# ------path]/etc/gitconfig file

**# add global configurations**

    git config --global user.email "vignan.kandelab@gmail"

    git config --global user.name "john Doe"

# To show branch name in "git log" outputs

    git config --global log.decorate auto

**# starting local repository**

        git init   --> Create repository

        git status     # u may get untracked files and floders

        git add .      # move files to staging

        git commit -m " give some message related to the changes" --> moves files to local repo

**# Rename the branch as per remote repo**

    git branch -M main/master   # depends on remote project branch

**#  Git that you want to add a new remote repository.**

        git remote add origin <git repository url >

        git push -u origin master/main            # depends on branch