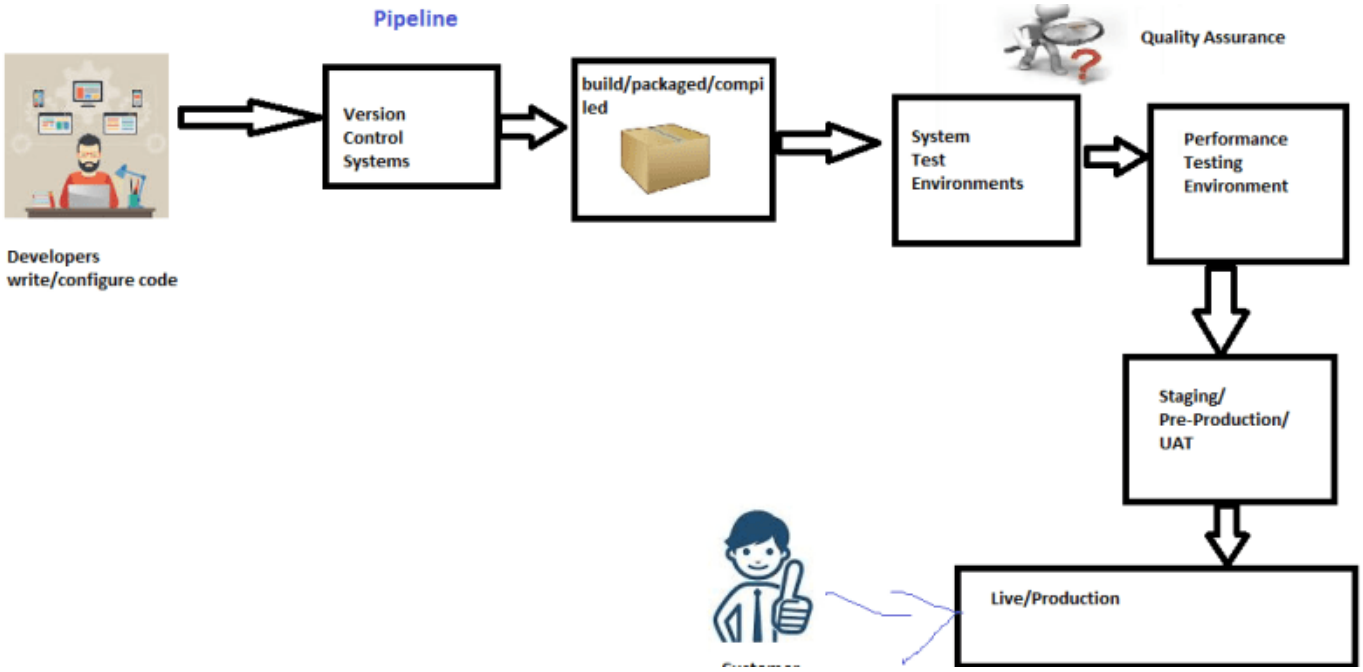
GIT:

Prerequisite Softwares For DevOps

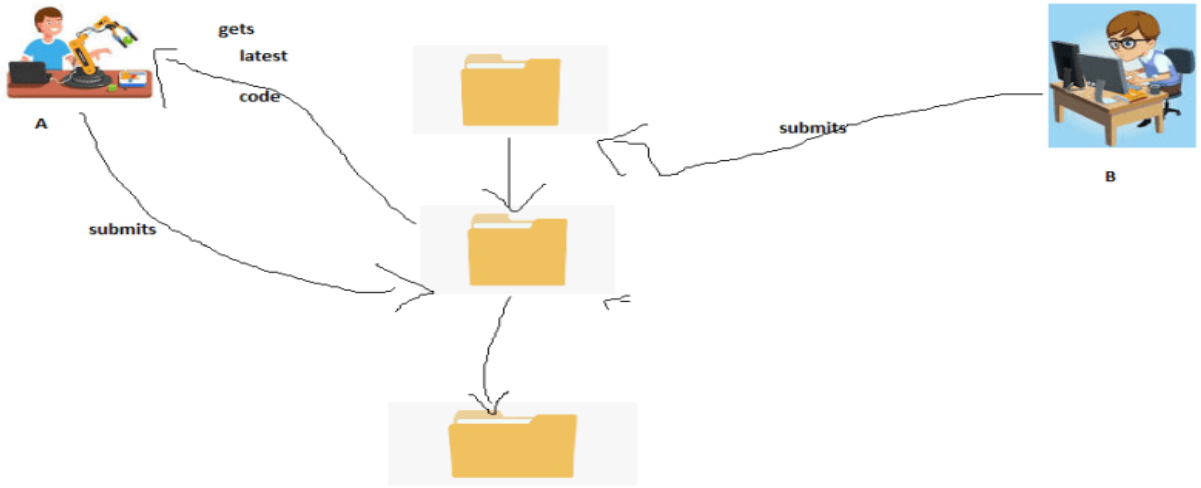
- Windows:
  - Git For Windows:refer here
  - Visual Studio Code: Click Here
- MAC:
  - Visual Studio Code: Click Here
- Linux Desktop:
  - Visual Studio Code: Click Here

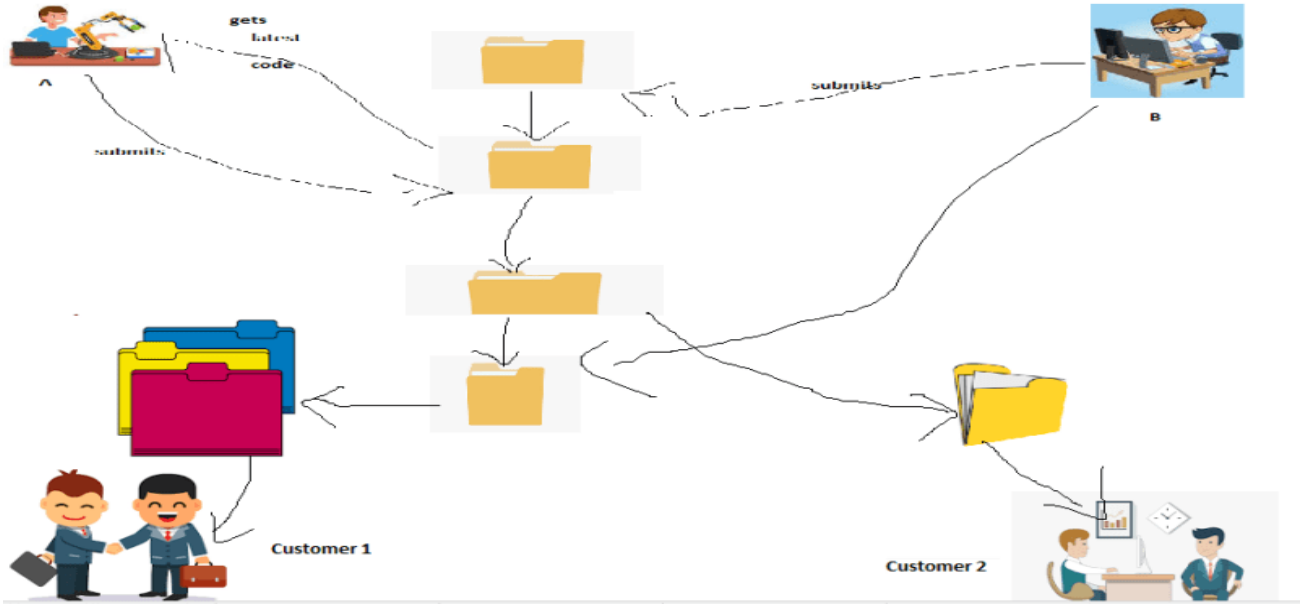
Generalize Pipeline



- Our journey into CI/CD Pipelines as a DevOps Engineer starts when developer finishes coding of some feature and shares it with you
- Sharing Code to Others and Possible Approaches
  - Share the code via Email:
    - Generally we have more that one developer working, then which version of whose mail should be consider
  - Shared Folder:
    - Looks like good option.

    - All of your devteam once finished the code have to copy that in a shared location, so which developers work to take is resolved by dev team
    - Some developers work has caused lot of trouble in testing, so your lead wants to remove the work done by your dev team for that day
- General Opinion of sharing code:
    - Multiple developers should be able to work in parallel
    - Your code should have some kind of history/version
- Our dev team software is customized for different customers how to manage





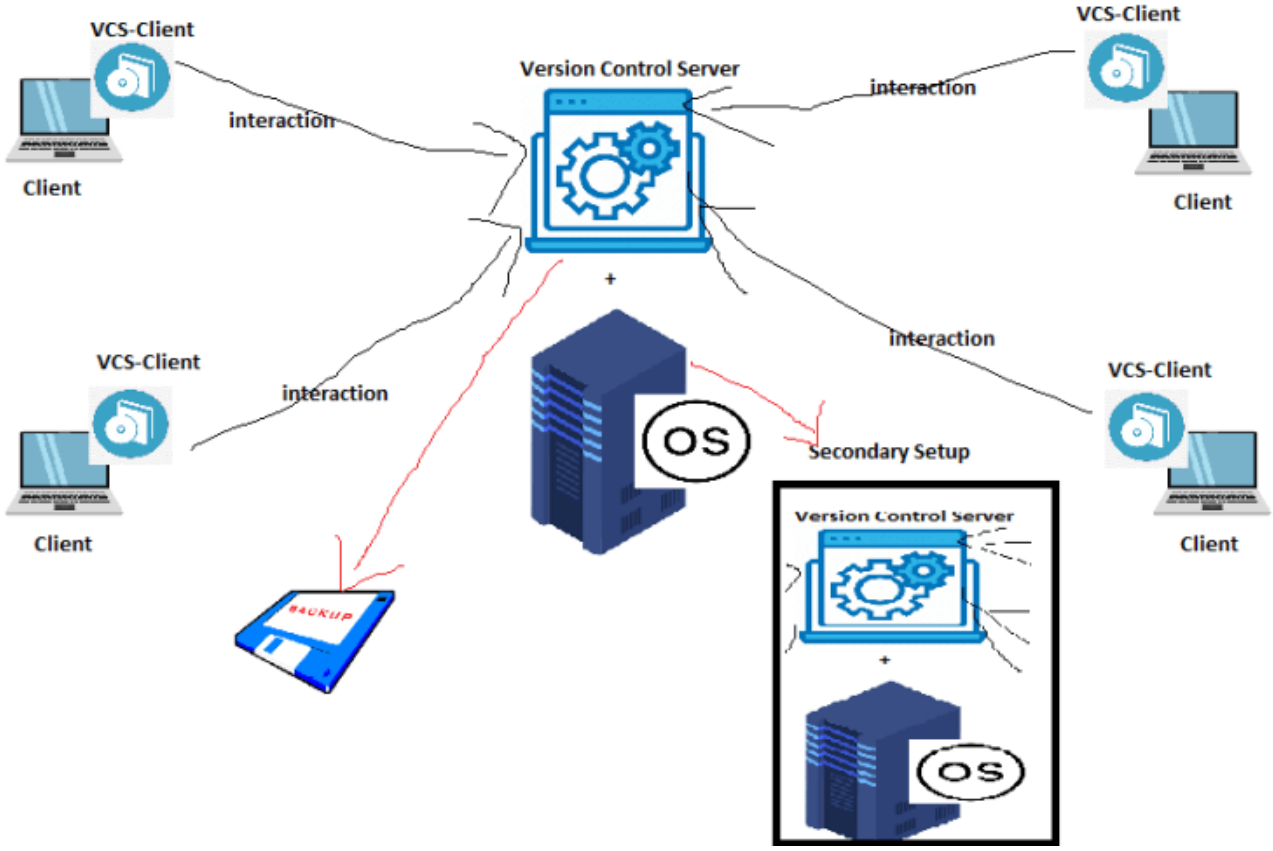
that?

- We need some software to manage all of the above challenges

    - Multiple users working parallelly on some code base
    - Version/History Support
    - Capability to maintain different copies of code base
    - Have features like Backup/Restore and Archive
- Version Control System is all about doing the above
-
- Examples:
    - CVS
    - VSS (Visual Source Safe)
    - Clear Case
    - Subversion
    - Perforce
    - Mercurial
    - TFVC (Team Found Version Control)
    - Git

Evolution in Version Control Systems

- Generations of VCS
    - Client Server:
        - Need a centralized Servers
        - All the developers systems will be clients
        - For Disaster Recovery take a Secondary Setup
        - Take Periodic Backups of VCS
        - For all of this we need a VCS Administrator
        - Two models:
            - Online Clients: For working server needs to connected all the time
            - Offline Clients: Clients can work and when they want to interact can establish a connection to server

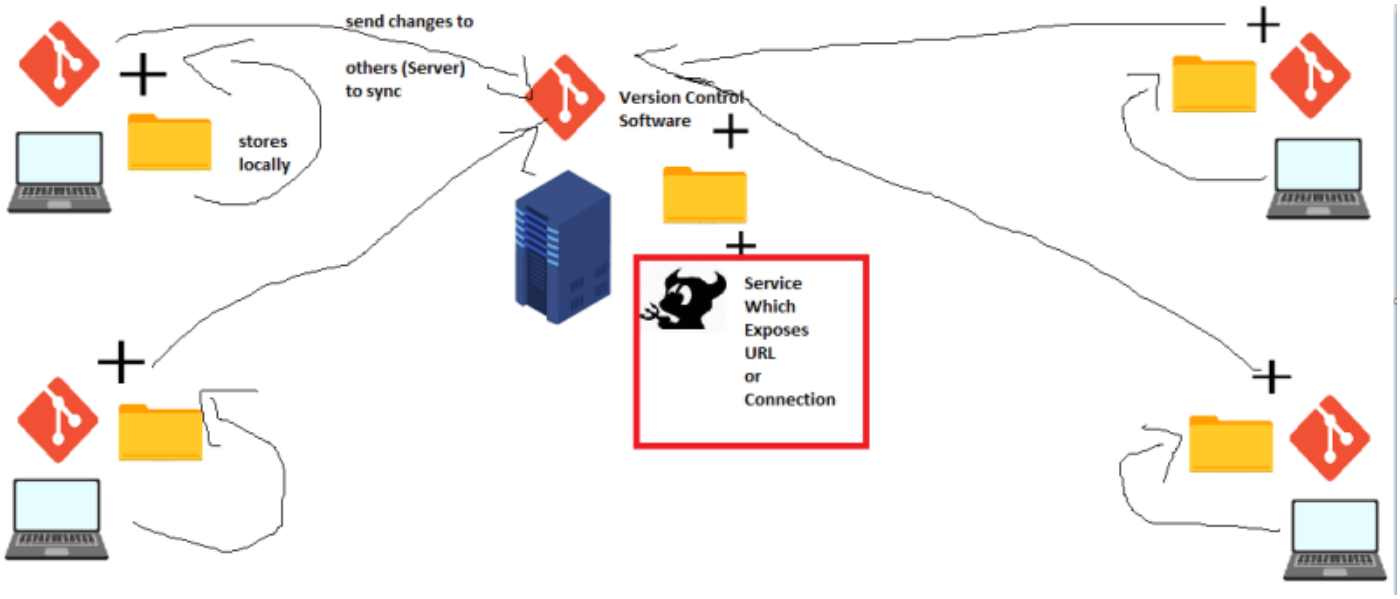


- Distributed
  - No Need of Centralized Servers
  - Any system can act as server
  - Backup is just one more system having the code

DevOps Engineers Misconceptions

- Only Developer only will use git(VCS) a lot?
  - Absolutely not, we as DevOps Engineers develop automation to pipeline and we use git to preserve our work
  - Ansible/Chef/Terraform/K8s/Dockerfile which you have developed will be stored in Version Control System
  - So DevOPs Engineers need to have two perspectives of VCS
    - Developer
    - Administrator

Distributed Version Control Systems (DVCS)



- In DVCS all the nodes
  - will have the same version control software installed
  - Will have the full copy of the code locally
- In DVCS we still use servers which is a node plus a Service to help other nodes get and submit the code
- Advantage of DVCS over Client Server (Centralized Version Control Systems):
  - No single point of failure
  - All developers will send the changes made (changesets) and each changeset is version controlled rather than individual file, so if some functionality is not working due to some developers mistake, its easy to revert back
  - Developers can work in offline and only when they want to sync the code the need to be connected to the server
  - Performance is much better
  - Less Adminstration
- Disadvantage:
  - If your project is very huge, and every developer having a full copy can be extra cost on storage
  - Are less performance when it comes to large binary files