Virtual Environment
&
Git

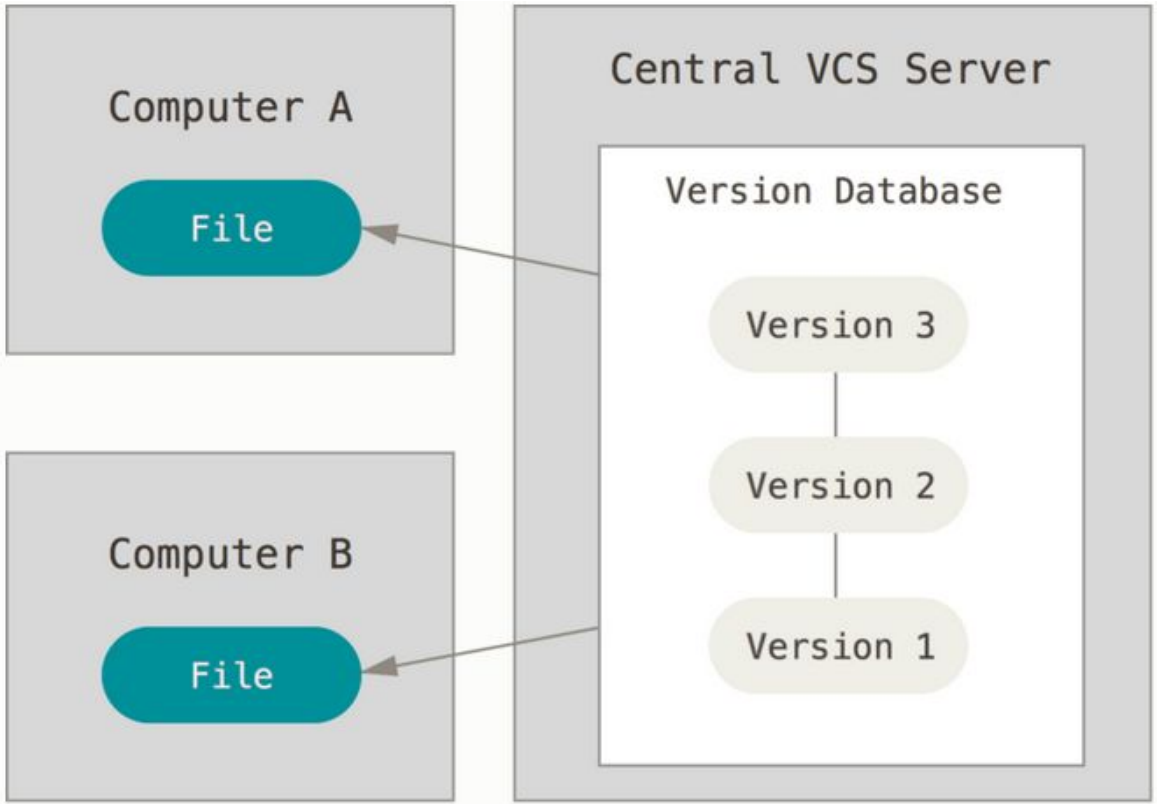
Virtual Environment in Python

- Virtual environments in Python are isolated spaces where you can install specific dependencies for a project. This allows you to work on multiple projects with different dependencies without conflicts.
- Can use conda, venv , etc
- **Why Use venv?**
    - Avoid dependency conflicts across projects.
    - Test and develop with specific versions of packages.
    - Keep the global Python installation clean.

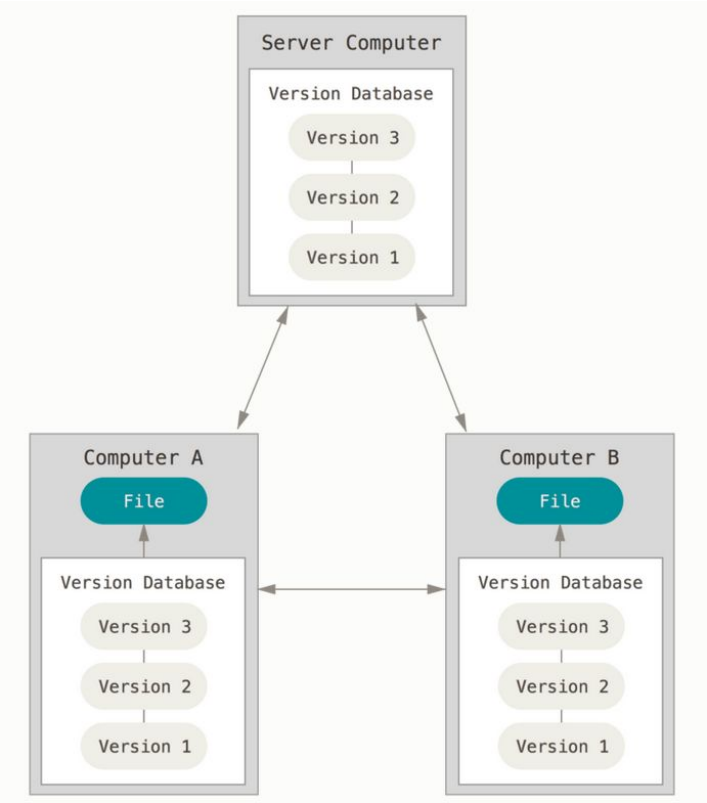
Git

- Git is a type of version control system.
- A version control system enables to record changes in a file over time.
- We can take snapshots of files over time, can restore back to a snapshot
- Work on multiple versions of a file in parallel.

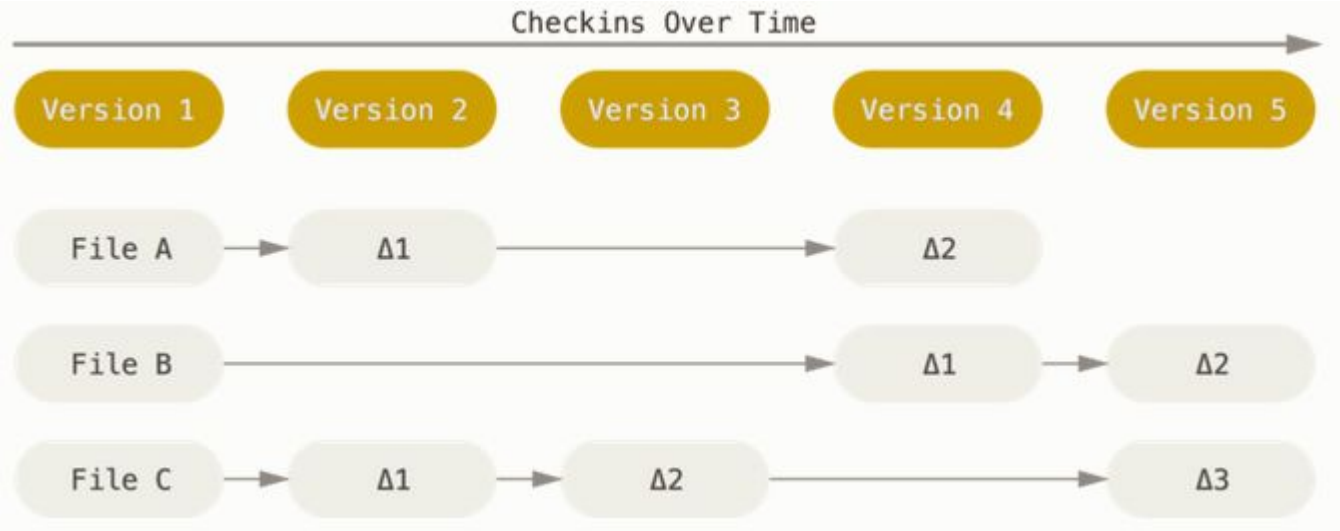
Centralized Version Control Systems



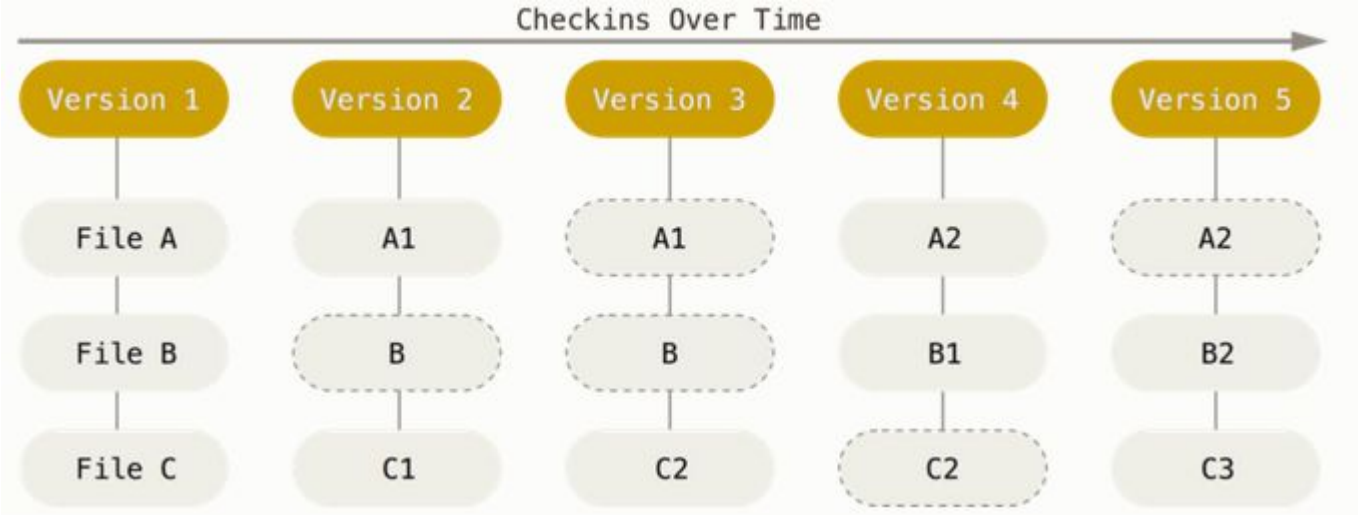
Distributed Version Control Systems



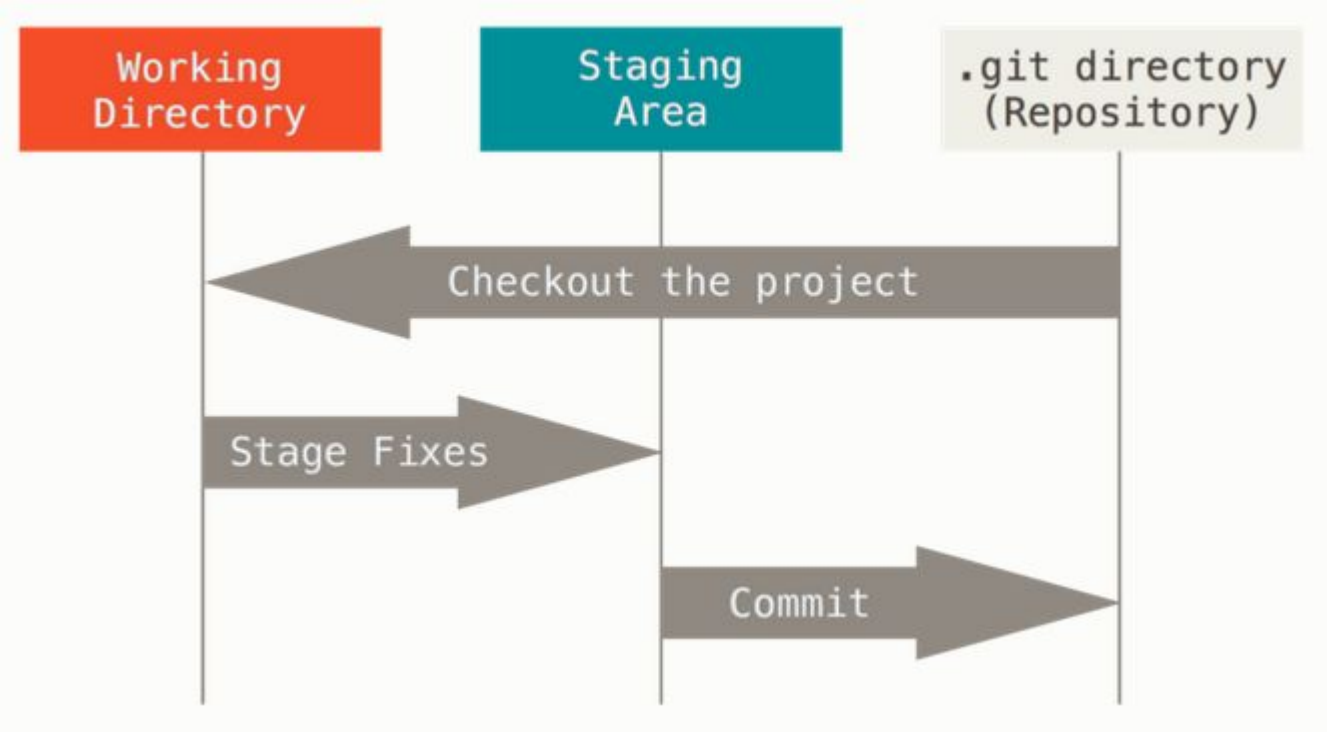
CVS


Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
Δ1
Δ2
File B
Δ1
Δ2
File C
Δ1
Δ2
Δ3

Snapshots in Git



States/ Workflow of Git



command	description
`git clone url [dir]`	copy a Git repository so you can add to it
`git add file`	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help [command]`	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
others: `init, reset, branch, checkout, merge, log, tag`	

Starter Information

- Installation : Refer to this link
- Setup Identity
  - git config --global user.name "John Doe"
  - git config --global user.email johndoe@example.com

Containers, Docker & Kubernetes

What Are Containers?

Definition:

Containers are lightweight, standalone, executable software packages that include everything needed to run an application: code, runtime, libraries, and system tools.

Key Components:

- Application code
- System libraries and dependencies
- Minimal system tools

Why Are Containers Important?

- **Isolation**: Each container runs in its own isolated environment, ensuring that applications do not interfere with one another.
- **Efficiency**: Containers share the host system's kernel, which means they use fewer system resources compared to Virtual Machines (VMs).
- **Portability**:

  Containers can run on any system that supports the container runtime (e.g., Docker).

Containers vs. Virtual Machines

Container

- An isolated environment for running applications.
- Allows running multiple applications in isolation.
- Light weight
- Need less hardware resources.
- Uses os of host.

VMs

- Virtual Machine is an abstraction of an machine.
- Hypervisor for managing vms.
- Resource Intensive.
- Each vm is a full OS system.

What is Docker ?

- Docker is a platform for developing, shipping, and running applications inside containers, which are lightweight and isolated environments.

Role:

- Simplifies application deployment by packaging code and dependencies into containers.
- Ensures consistency across development, testing, and production environments.

Basic Docker Concepts

Docker Images:

- Blueprints for containers, containing the application code and dependencies.

Docker Containers:

- Running instances of Docker images.
- Containers are the actual working environments where the application executes.

Dockerfile:

- A text file that defines how to build a Docker image.
- It contains instructions like what base image to use, dependencies to install, and how to run the application.

Docker Hub:

- A registry where Docker images are stored and shared.
- You can pull images from Docker Hub or push your custom images to it.

Key Docker Commands

- **docker pull**: Pull images from Docker Hub.
- **docker run**: Run a container from an image.
- **docker ps**: List running containers.
- **docker stop**: Stop a running container.
- **docker rm**: Remove stopped containers.
- **docker build**: Build an image from a Dockerfile.

Kubernetes

Definition:

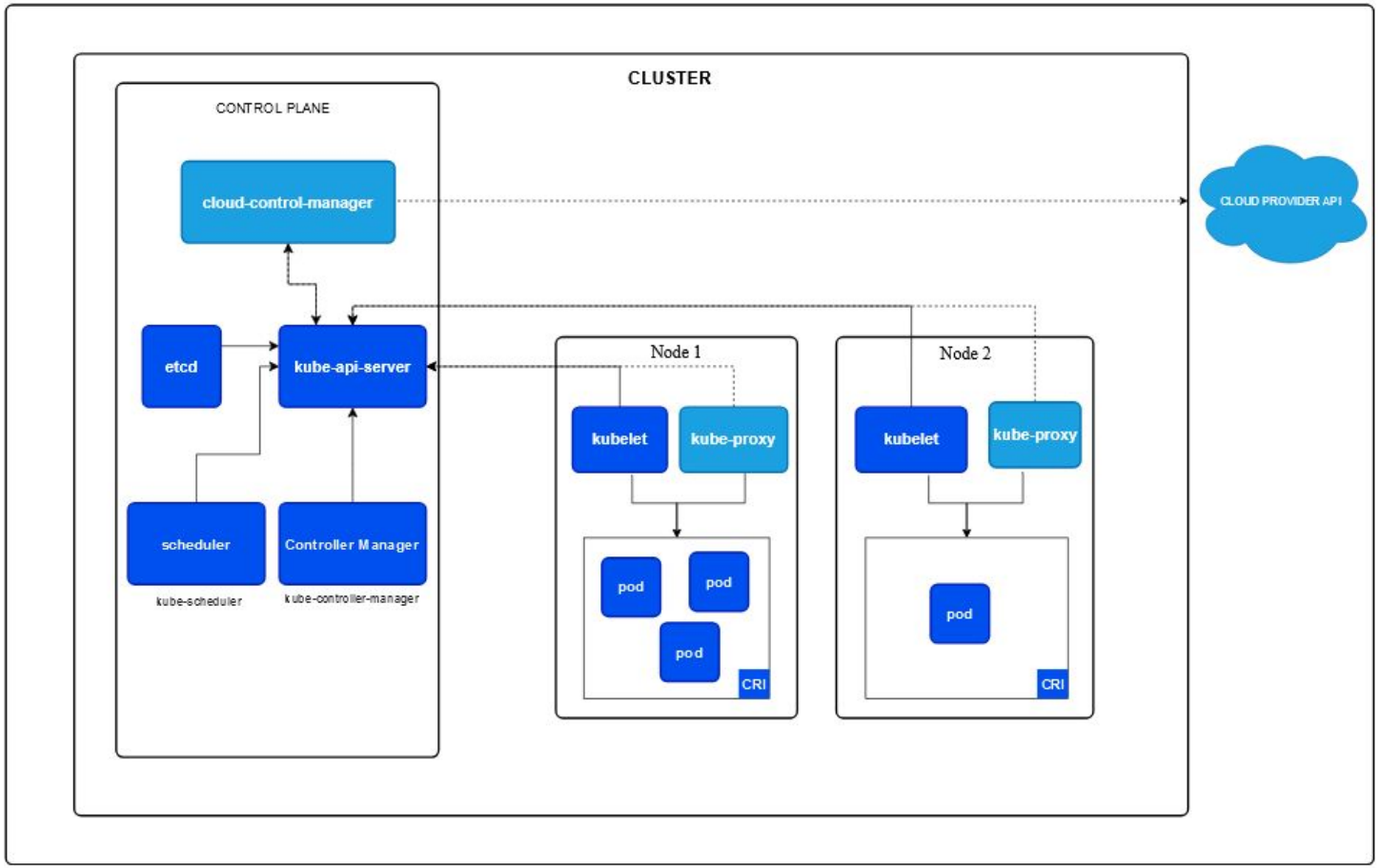
- Kubernetes is a container orchestration platform for automating the deployment, scaling, and management of containerized applications.

Why use Kubernetes?

- Manages multiple containers in a cluster of machines.
- Automates scaling and failure recovery.

Key Kubernetes Concepts

- Node
- Pod
- Deployment
- Service
- Namespace
- ConfigMap
- Secret
- Volume
- Scheduler
- Controller Manager
- API Server



Ref : https://kubernetes.io/docs/concepts/architecture/

Helpful Guides

- https://medium.com/@kyledeguzmanx/quick-step-by-step-guide-to-generating-an-ssh-key-in-github-d3c6f7e185bb
- https://git-scm.com/book/en/v2
- https://www.youtube.com/watch?v=ZDR433b0HJY
- https://marklodato.github.io/visual-git-guide/index-en.html
- https://docs.python.org/3/tutorial/venv.html
- https://realpython.com/python-virtual-environments-a-primer/
- https://endjin.com/blog/2022/01/introduction-to-containers-and-docker
- https://docs.docker.com/
- https://kubernetes.io/docs/concepts/overview/components/