BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to DevOps

Sonika Rathi

Assistant Professor
BITS Pilani

Review CS#3

DevOps : People & Tools Dimension

- Transformation to Enterprise DevOps culture
- DevOps - People
- DevOps – Tools
- Cloud as a catalyst for DevOps
- Transition in IT
  - Building competencies, Full Stack Developers
  - Self-organized teams, Intrinsic Motivation
- Technologies in DevOps

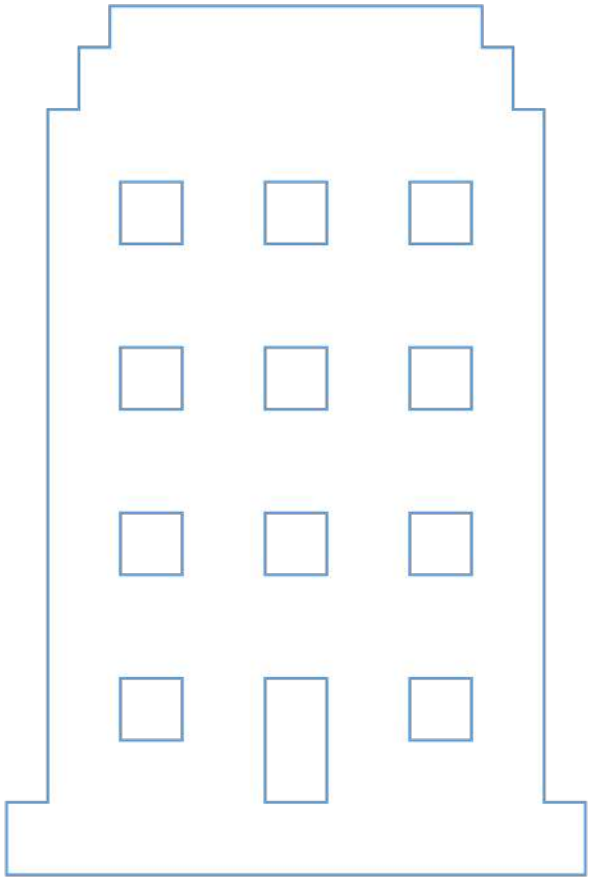
Agenda

Version Control System

- Evolution of Version Control
- Version Control System Types
  - Centralized Version Control Systems
  - Distributed Version Control Systems
- Introduction to GIT
- GIT Basics commands
- Creating Repositories, Clone, Push, Commit, Review
- Git Branching
- Git Managing Conflicts
- Git Tagging
- Git workflow
  - Centralized Workflow
  - Feature Branch Workflow
- Best Practices- clean code

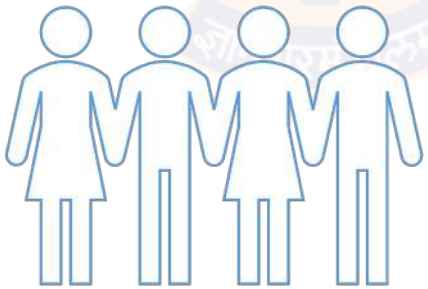
Why Version Control?

Lets find out the problem



XYZ Corp

Shared Folder structure development



Team: Tiger
(4 members)

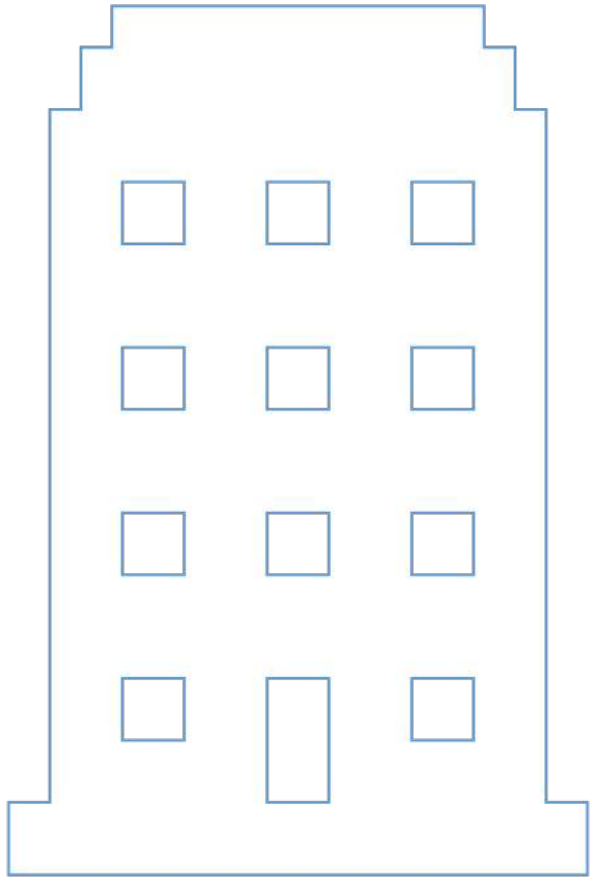
Issue: Open Heart Surgery

Business Challenges:

1) Reduce Cost
2) Hire People
3) Get right Speed

Why Version Control?

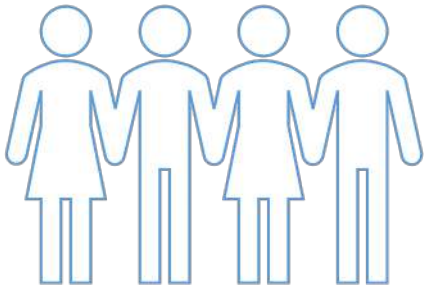
Lets find out the problem : Scenario 2



XYZ Corp

Shared Folder structure development

Team: Tiger
(4 members)



Team: Tiger 2
(offshore)

Challenges:

1) Speed
2) Identification of changes
3) Tracking history
4) Collaboration

Evolution of Version Control

Generations of VCS

- What is “version control”, and why should you care?
- Definition: Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later
- The history of version control tools can be divided into three generations

Generation	Networking	Operations	Concurrency	Example Tool
First Generation	None	One file at a time	Locks	RCS, SCCS
Second Generation	Centralized	Multi-file	Merge before commit	CVS, Subversion
Third Generation	Distributed	Changesets	Commit before merge	Bazaar, Git

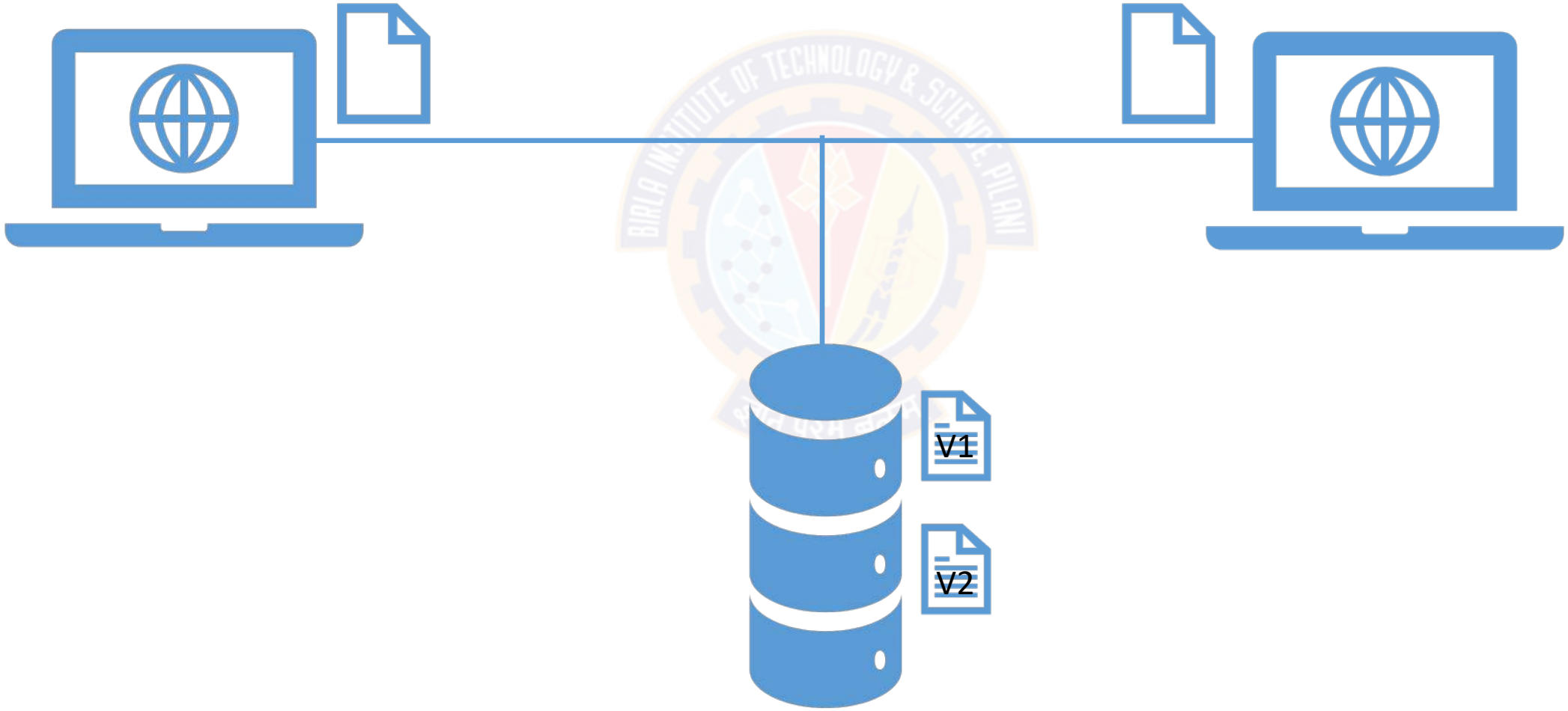
Version Control System

Benefits

- Change history
- Concurrent working (Collaboration)
- Traceability
- Backup & Restoration

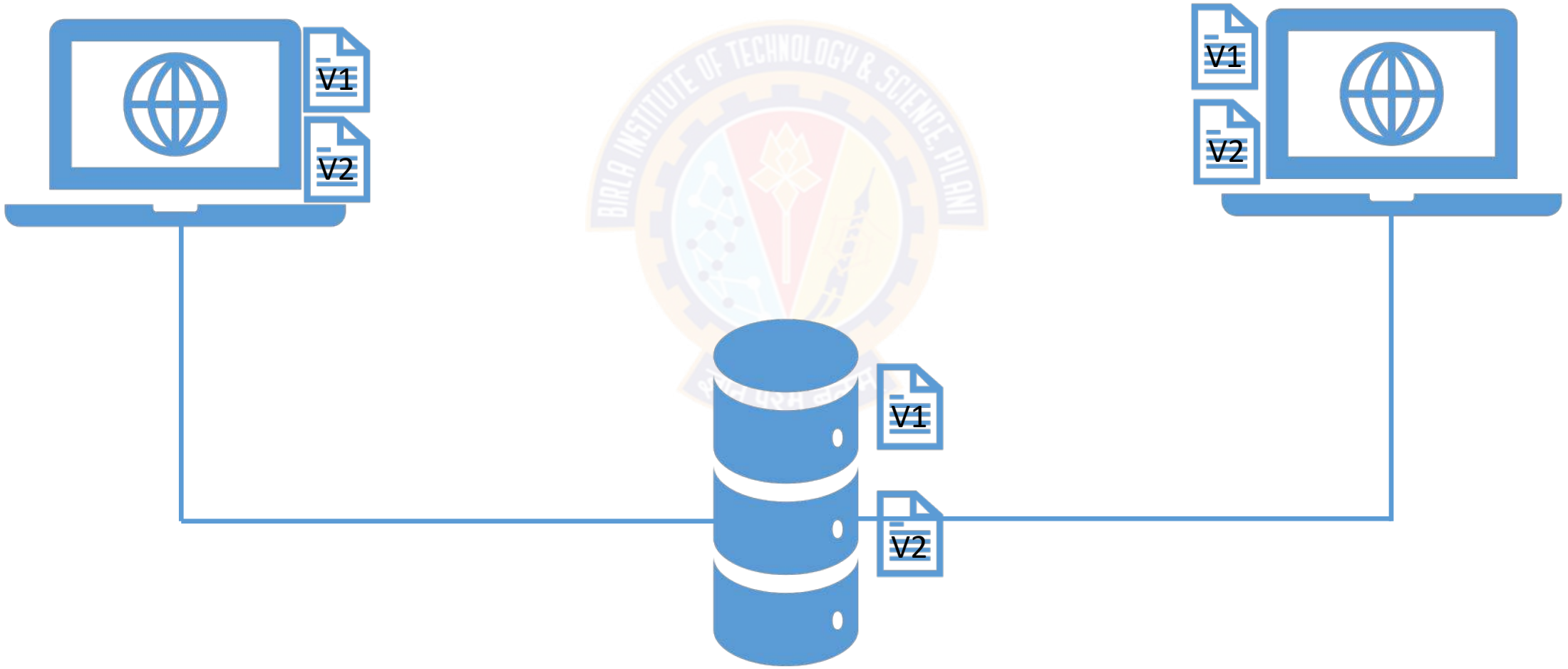
Version Control System Types

Centralized source code management System



Version Control System Types

Distributed source code management System



CVCS Vs. DVCS

Lets discuss con’s

CVCS

- Sigle point of failure
- Remote commits are slow
- Continuous connection

DVCS

- Need more space
- Bandwidth for large project

Available Tools

CVCS

- Open source:
  - Subversion (SVN)
  - Concurrent Versions System (CVS)
  - Vesta
  - OpenCVS
- Commercial:
  - AccuRev
  - Helix Core
  - IBM Rational ClearCase
  - Team Foundation Server (TFS)

DVCS

- Open source:
  - Git
  - Bazaar
  - Mercurial
- Commercial:
  - Visual Studio: Team Services
  - Sun WorkShop: TeamWare
  - Plastic SCM – by Codice Software, Inc
  - Code Co-op

Git & GitHub

What we will learn in this session

- Git & GitHub relationship
- Prerequisite
- Foundation of Git

Concept of Git

Understanding GitHub

Beyond the basics

Git

What is Git

Popular Source Control system

Distributed system

Free (Open Source tool)

Git

Why use Git

- Fast
- Disconnected
- Powerful yet easy
- Branching
- Pull Requests

GitHub

What is GitHub?

Hosting service on Git

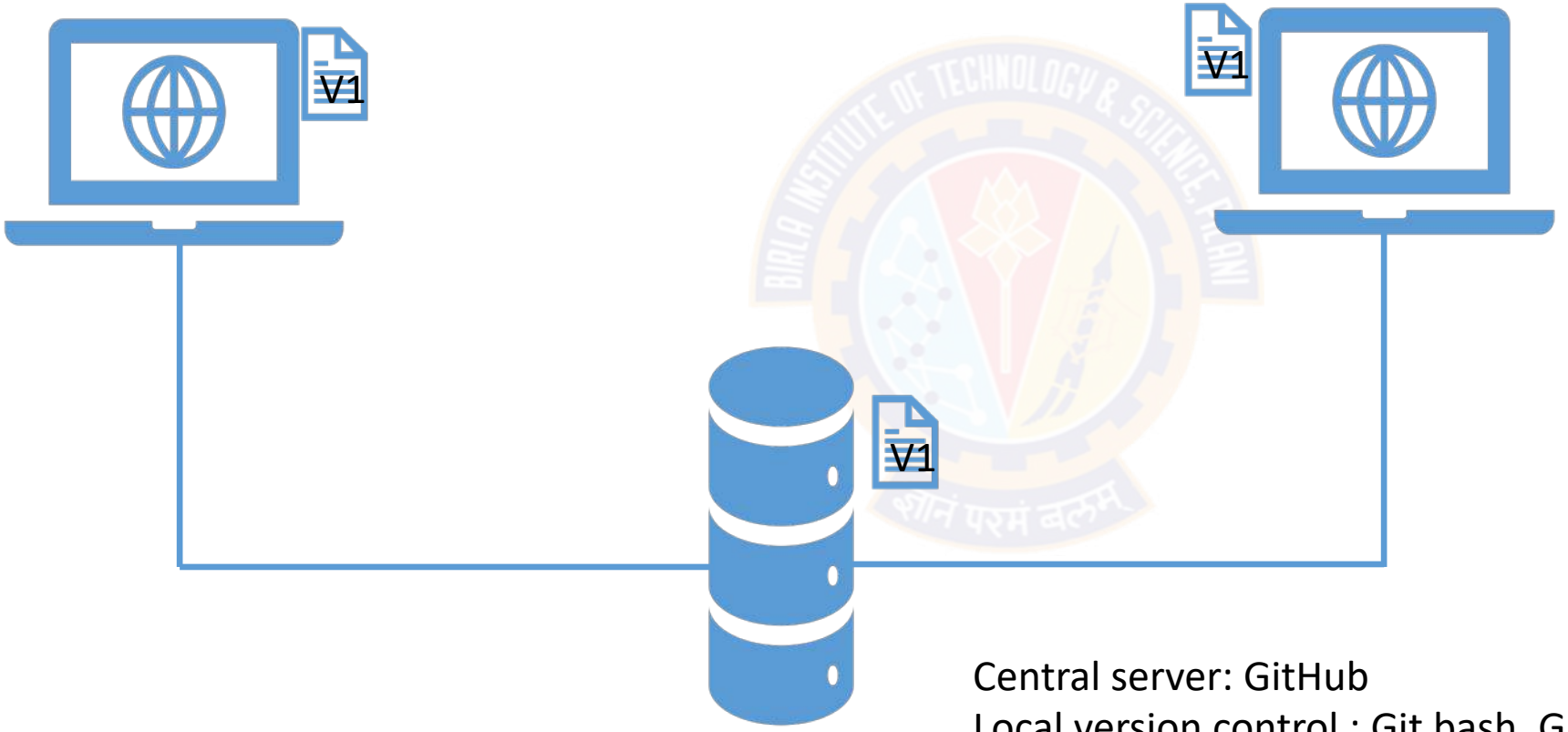
More than just source control for your code

- Issue management, Working with Teams, etc.,

Free & Paid options

Git & GitHub

Relationship



Central server: GitHub
Local version control : Git bash, Git Desktop

Git

Working with Git

Console

We will use this

GUI

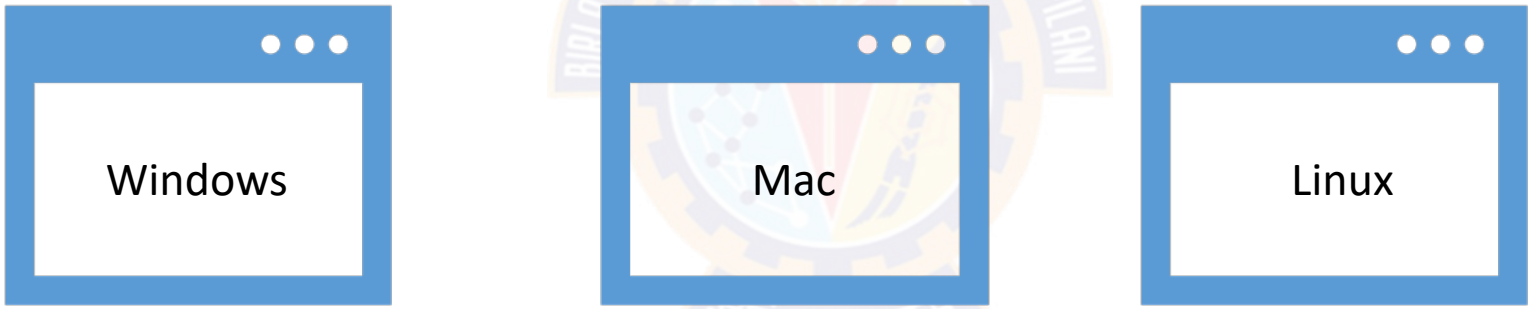
GitHub Desktop

Source tree

Git & GitHub

Getting your system ready

- Install appropriate Git bash form official site
- Command line (Git bash)
- Server account (GitHub account)



Support for all platform

Git Foundation

The 3 State of Git

Modified

Staged

Committed

Git Foundation

The 3 areas of Git

Working directory

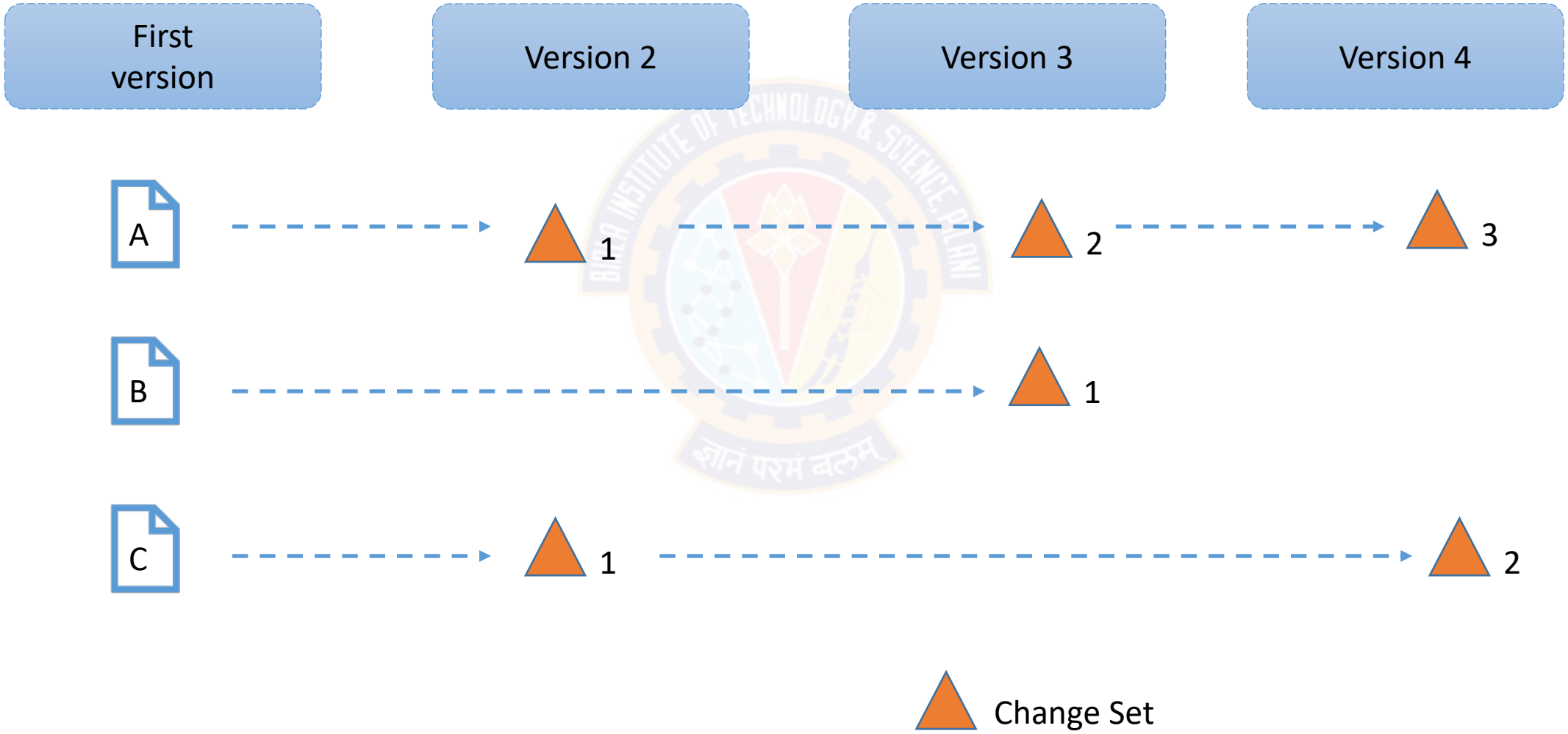
Staging area

.git repo

Remote repo GitHub

Git Foundation

Traditional Source Management

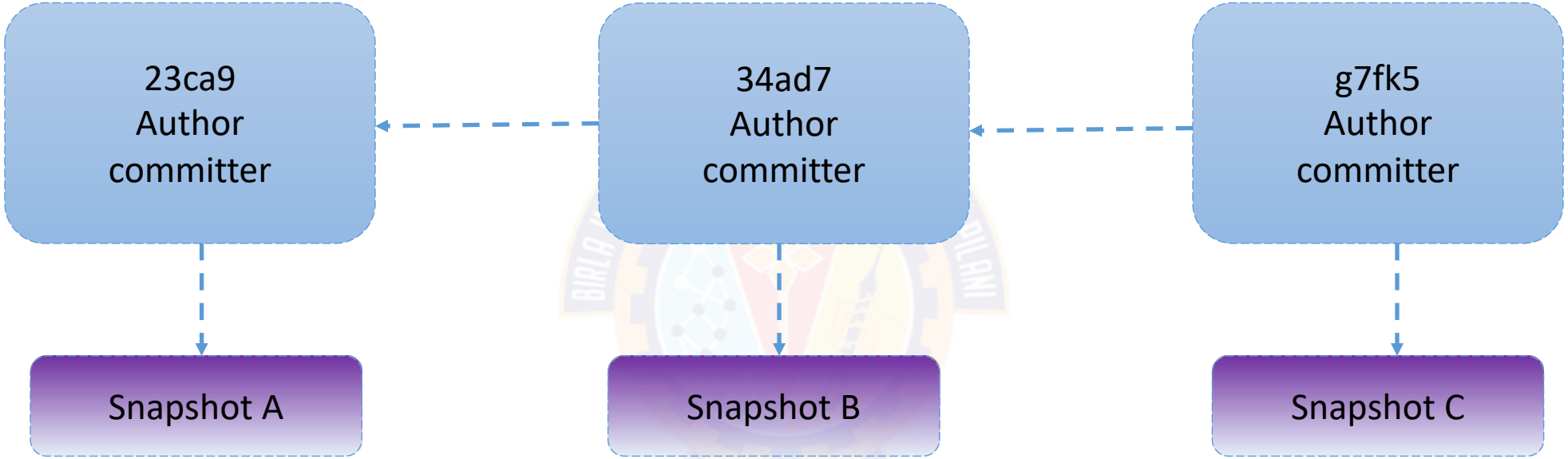


Git Foundation

Concepts of Snapshots in Git & GitHub

Git Foundation

Commits in Git



Git

Basic Commands

$ git
$ git config
$ git init
$ git clone
$ git status
$ git add
$ git commit
$ git branch
$ git checkout
$ git push
$ git fetch
$ git merge
$ git pull
$ git log
$ git reset
$ git revert

Lets demo

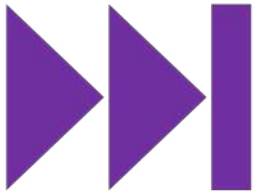
GitHub

GitHub's main Features

Code management

Pull requests

Issues



CICD

Global search

Lets demo

GitHub

Connecting with local machine

HTTPS

Requires user name & password

SSH

Easier to work with

GitHub

Working with repository

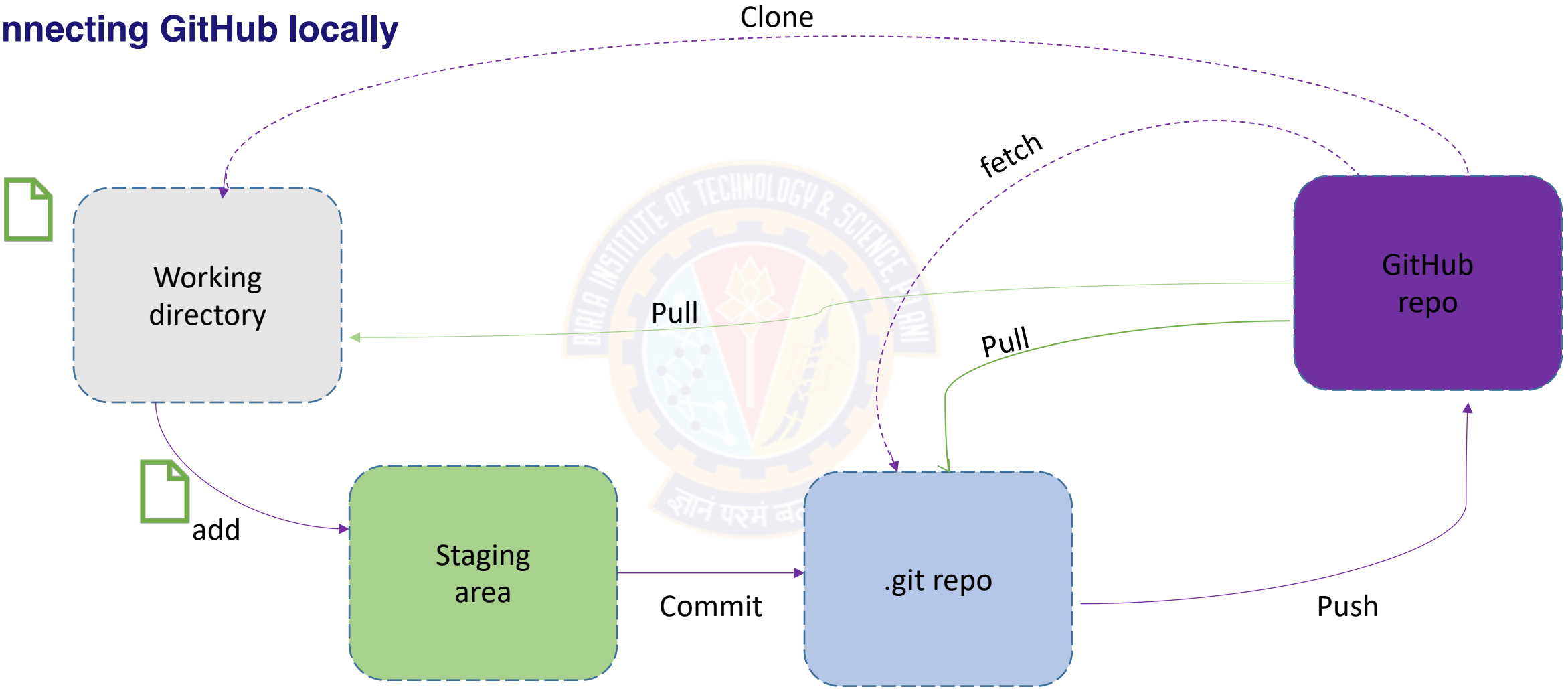
Repositories are building block of GitHub

“Folder” for your project

Public or Private

GitHub

Connecting GitHub locally



Lets demo

GitHub

Working with special files

README is special file known by GitHub

Rendered automatically on landing page

Typically written in markdown(.md)

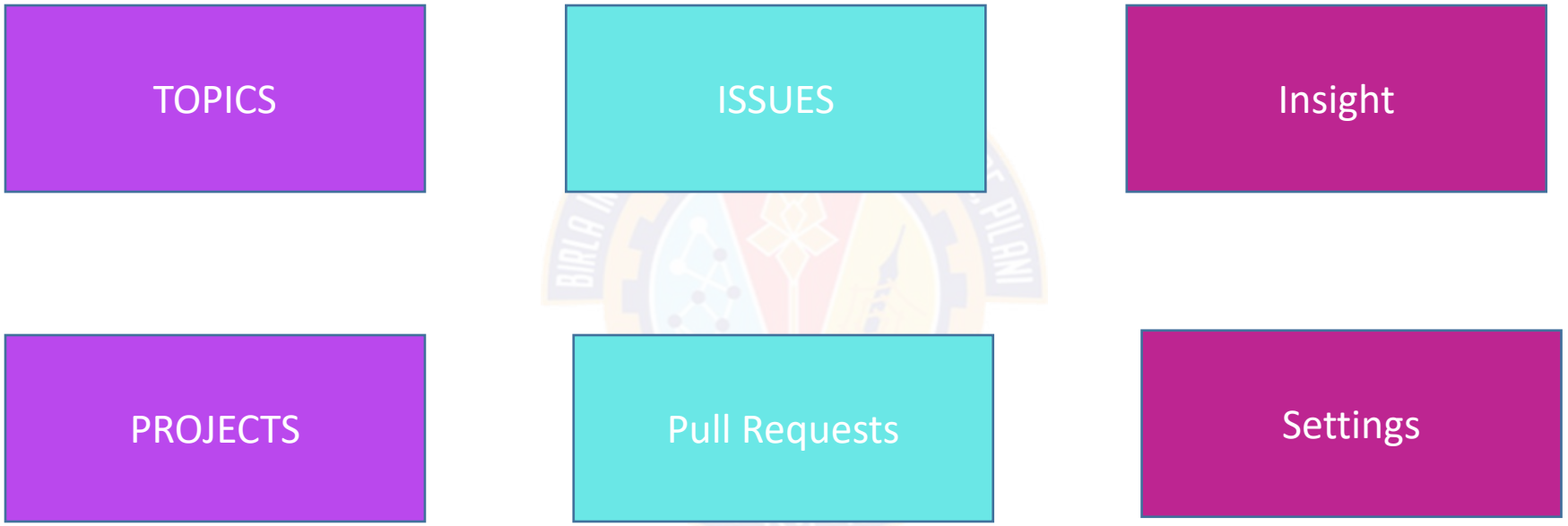
Other files:

LICENSE	CHANGELOG	CODE_OF_CONDUCT
CONTRIBUTORS	SUPPORT	CODEOWNERS

Lets demo

GitHub

Repository Feature



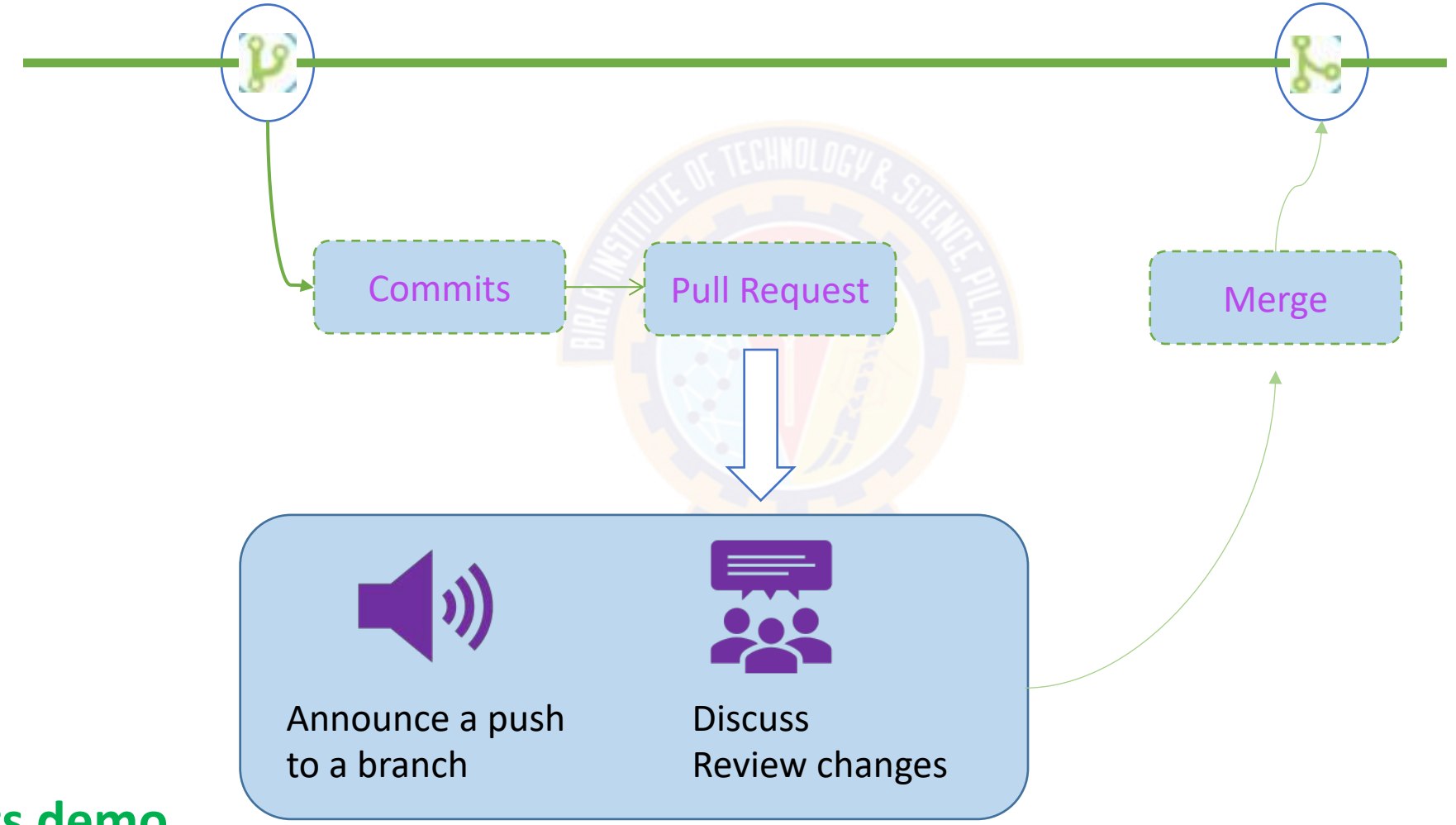
Lets demo

Git

Workflow

GitHub

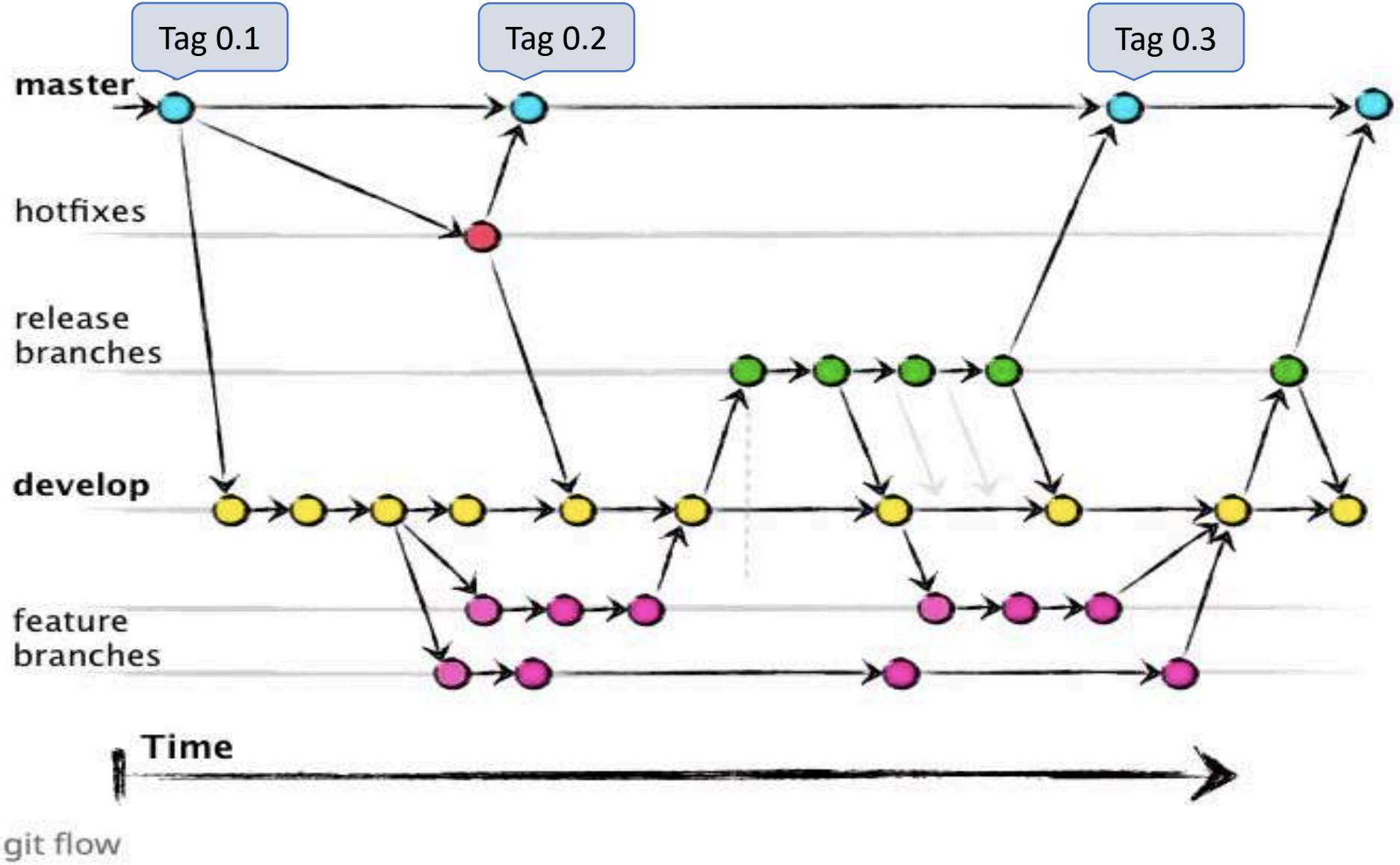
Branching



Lets demo

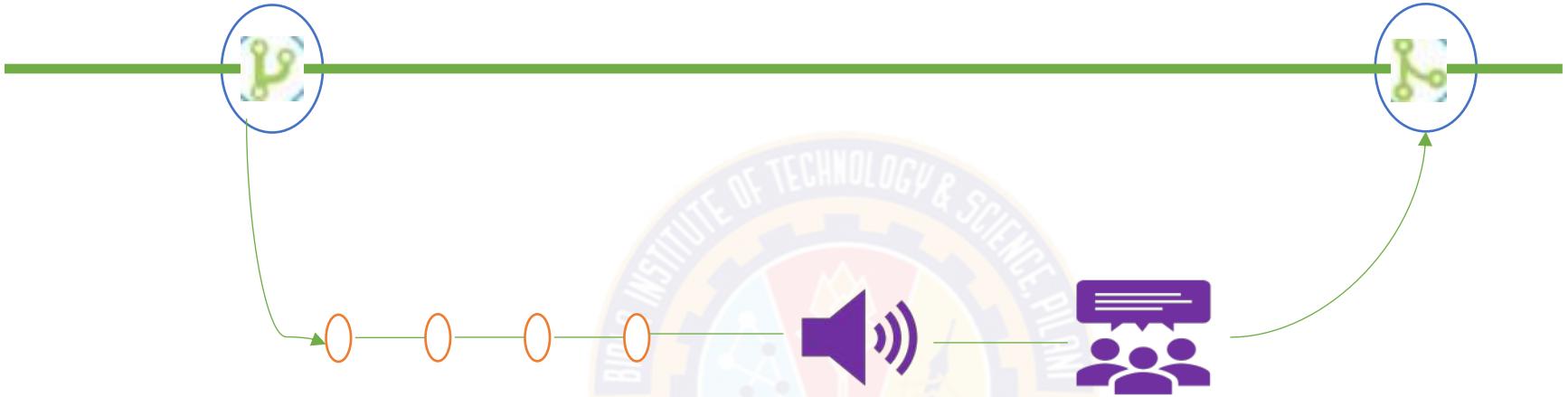
GitHub

Git Flow



GitHub

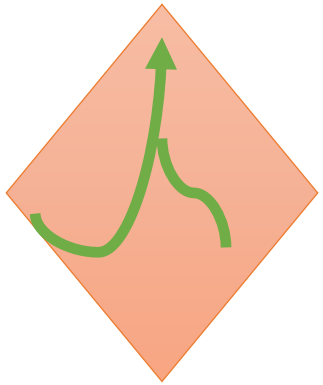
GitHub Flow



GitHub Flow combines the mainline and release branches into a “master” and treats hotfixes just like feature branches.

Git & GitHub

Merging with conflicts



Typical conflicts:

- Editing on the same line
- Editing on already deleted file

Merge conflicts needs to be solved before merge happen (Manual intervention)

Lets demo

Git Command

Revert

- Git revert will create a new commit
- Git revert undoes a single commit
- It is a safe way if undo

Reset

- Git reset command is a complex
- Dangerous
- Git reset has three primary form of invocation
  - git reset --hard HEAD
  - git reset --mixed HEAD
  - git reset --soft HEAD

Git command

Git Tag

- Tags are ref's that point to specific points in Git history
- Marked version release (i.e. v1.1.1)
- A tag is like a branch that doesn’t change
- Unlike branches, tags, after being created, have no further history of commits
- Common Tag operations:
  - Create tag
  - List tags
  - Delete tag
  - Sharing tag

Git

Clean Code

- Follow standard conventions
- Keep it simple, Simpler is always better, Reduce complexity as much as possible
- Be consistent i.e. If you do something a certain way, do all similar things in the same way
- Use self explanatory variables
- Choose descriptive and unambiguous names
- Keep functions small
- Each Function should do one thing
- Use function names descriptive
- Prefer to have less arguments
- Always try to explain yourself in code; put proper comments
- Declare variables close to their usage
- Keep lines short
- Code should be readable
- Code should be fast
- Code should be generic
- Code should be reusable

Q&A

Thank You!

In our next session: