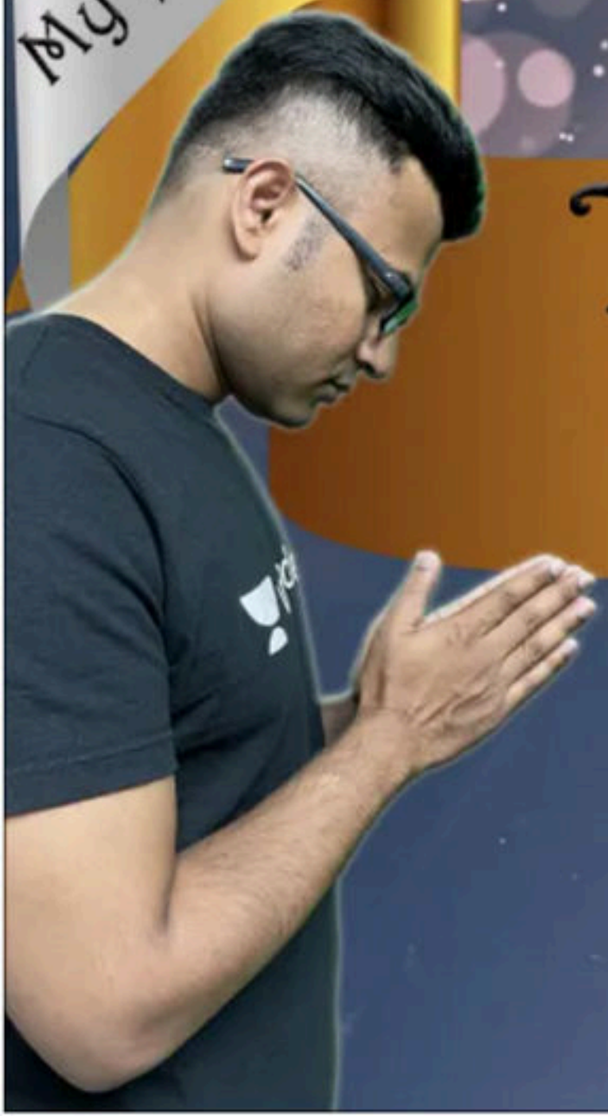
My philosophy

TEACHING IS WORSHIP

STUDENTS ARE GODS

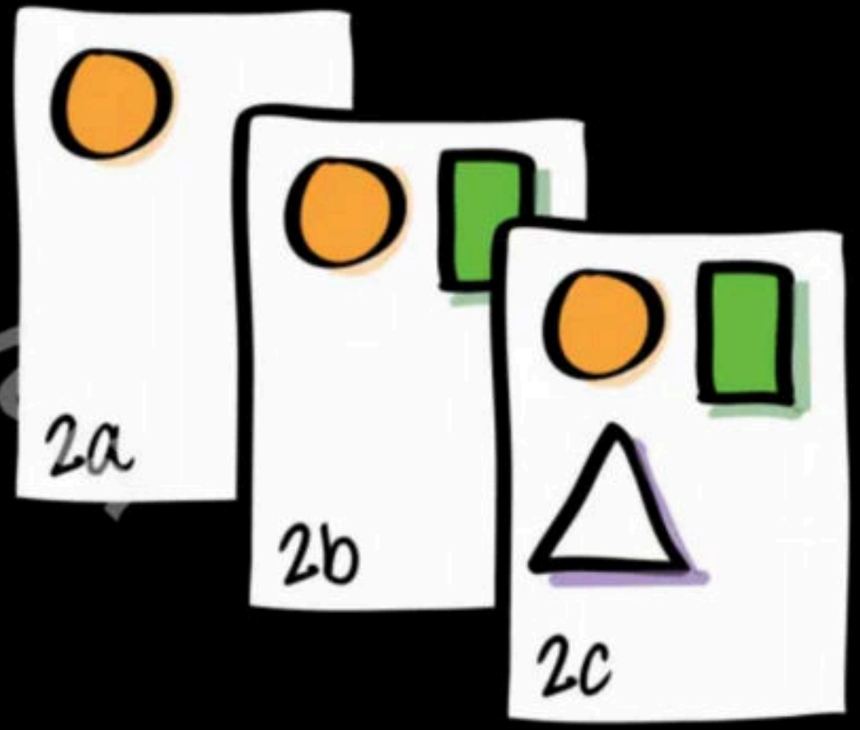


INTRODUCTION TO GIT AND GITHUB

What is Version Control System?

A Version Control system (VCS), also known as Revision Control or Source control system, is software that efficiently stores all the changes that are done in the filesystem.

In simple words, it is like a time machine that enables you to go backward in the timeline of your work project so that you can rectify the origin of mistakes/Bugs that you did at that time and also prevents losing or writing over the critical files.



If you are a college student working on a small college project or someone who has not worked on a large codebase where multiple software developers are contributing to the changes, you might not realize the importance of this tool

With the help of this course, we will make sure that you get well versed with the practical use cases of any version control system ☺

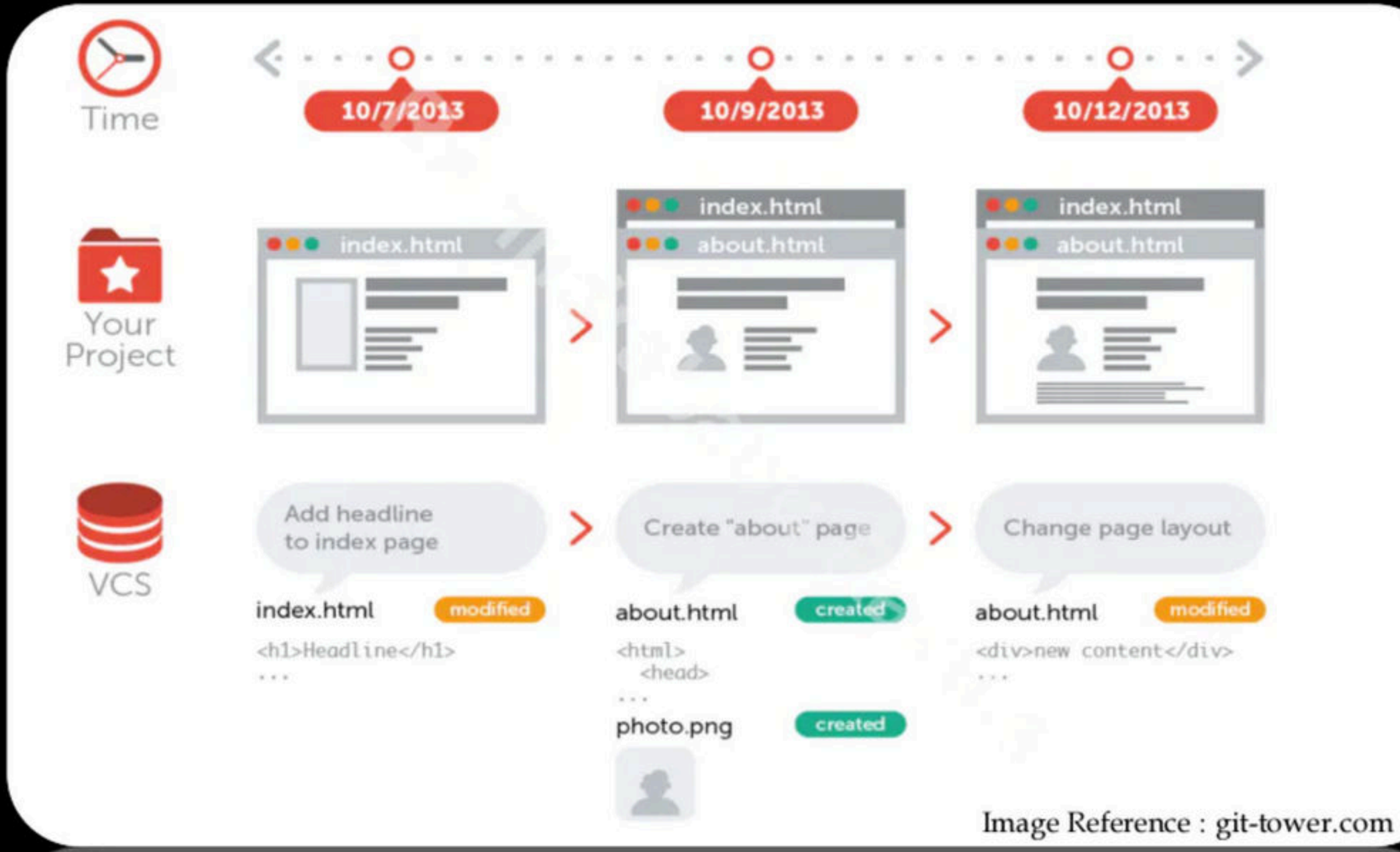
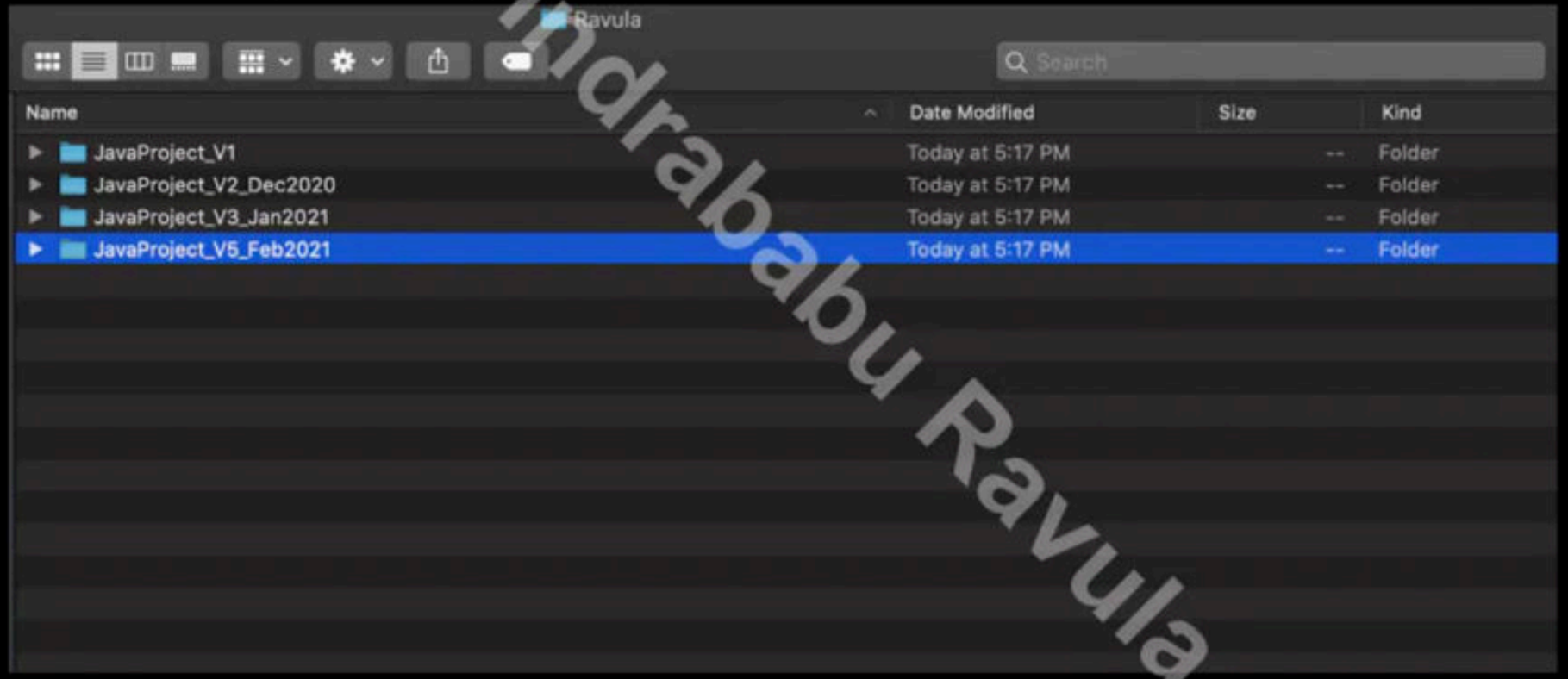


Image Reference : git-tower.com

- Source code of any live software keeps on evolving as it has to go through multiple cycles of Proof of Concept, writing the actual code and testing.
- Even after you have released the 1.0 version of your software, it will still continue to evolve as the projects need to be maintained and enhanced and there will be multiple future releases as well.
- Tracking all these details while the codebases evolve is just the sort of thing that the VCSs are good at.

Tedious Way - Doing Version Control Manually

The easiest and primitive version control which I think everyone must have done in their college life is to manually create multiple backups of your project periodically and add the versioning while naming the files as shown in the image below :



This hand-hacking approach works fine if you are working on a college project.

But enterprise software projects with huge codebases? Not a chance !!

Do you really think that code of Android would be sitting in a folder like "**Android-Latest-UPDATED**" for any programmer to make the changes?

That each software engineer will be allotted a different sub-folder to work on a specific module of an Android?

Hell No!

There are actually many bottlenecks of the mentioned approach

1.) If your computer’s hard drive crashes in the future, you will not be able to recover your projects unless you have taken more backups on some online storage which is another headache.

2.) If there are multiple people working on this project, it will be difficult to work simultaneously. You will have to provide them access to this project through some means (either email or setting a shared drive). Another guy can not start in this model until you finish the current revision and hand over the latest snapshot to them.

3.) No additional metadata regarding what exactly changed between multiple snapshots, who did the changes, and what was the reason for the change.

As you can see, the this approach is very time-consuming and error-prone as well.

Now, let’s look at some of the powerful features which modern VCS provides and can automate away most of the things involved in keeping an annotated history of your project and avoiding modification conflicts.

Why do we need VCS?

Complete Change History

Every change like creation, addition, deletion as well as edits will be recorded by the VCS. History also includes the author, timestamp as well as some notes regarding why that particular change was made. Having complete change history of your project enables you to look back into the past changes which is of great help while root causing any critical bugs in production.

From the image below, you can see that who has done what type of change in the past where top entry is the latest change.

```
commit cc5d7757e3bd05b9cd2d75147c6854e6587a0074 (HEAD -> 1.5.1, tag: release-1.5.1-rc0, origin/1.5.1)
Author: bkonold <bkonold@users.noreply.github.com>
Date:   Tue Aug 18 19:49:21 2020 -0700

    SAMZA-2578: Excessive trimming during transactional state restore (#1413)

commit 069c1daa60c1478025d731b7b0de5b663de52eb3
Author: Bharath Kumarasubramanian <bharathkk@apache.org>
Date:   Tue Aug 18 21:25:10 2020 -0700

    Update the version in the tests and gradle.properties to 1.5.1

commit 83098469bccb7bb956a6c87e1a93e550afac6bd1
Author: mynameborat <bharath.kumarasubramanian@gmail.com>
Date:   Wed Jul 1 17:21:45 2020 -0700

    [DOCS] Update version in the docs for 1.5.0 release (#1398)

commit 6fe7efd7d2e186a266d3c04395d94db001c1f7f8 (tag: release-1.5.0-rc1)
Author: Sanil15 <sanil.jain15@gmail.com>
Date:   Mon Jun 8 15:35:03 2020 -0700

    SAMZA-2504: Improve Container Placement Flaky Test & Running Time

    Improvement [Bug fix]:

    - Fix a flaky test for Container Placements on Request status
    - Improve the running time of Test suite from 40 secs to under 4 seconds

    API changes: None

    Upgrade Instructions: None
```

Concurrent Contributions using Branching

In a collaborative environment, there will be some developers who will be working on implementing new features while some developers may be working on some bug fixes in the already released version of the product. Branching allow each software developer to work in isolation from other team members so that code changes related to your feature/bug fix will not have any effect on the changes being worked upon by other members. A git branch is an independent line of development taken from the same source code and facilitates concurrent contribution from multiple software developers. No one has to wait for the other to start their work therefore reducing the development time a lot for complex project.

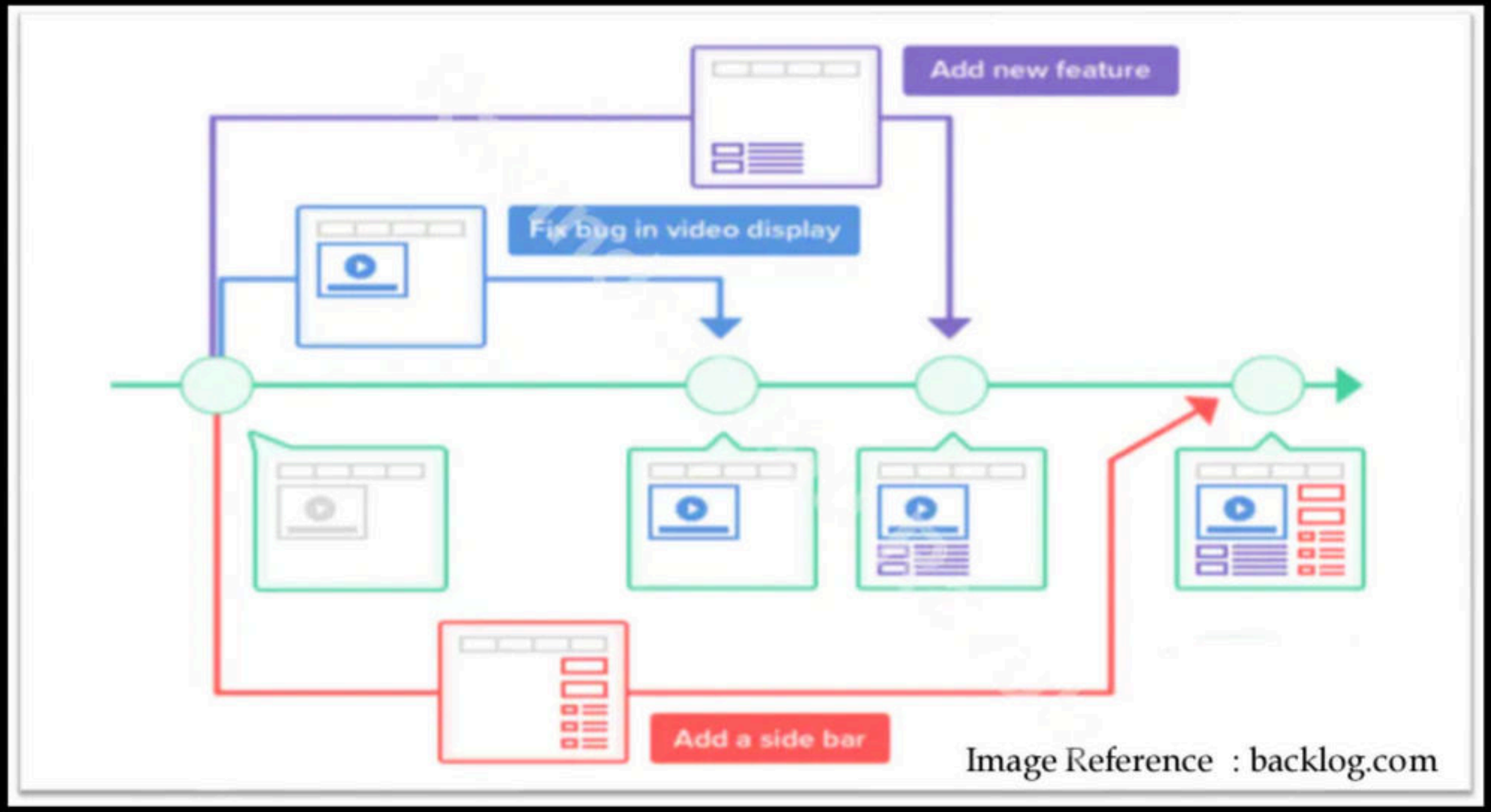


Image Reference : backlog.com

Traceability

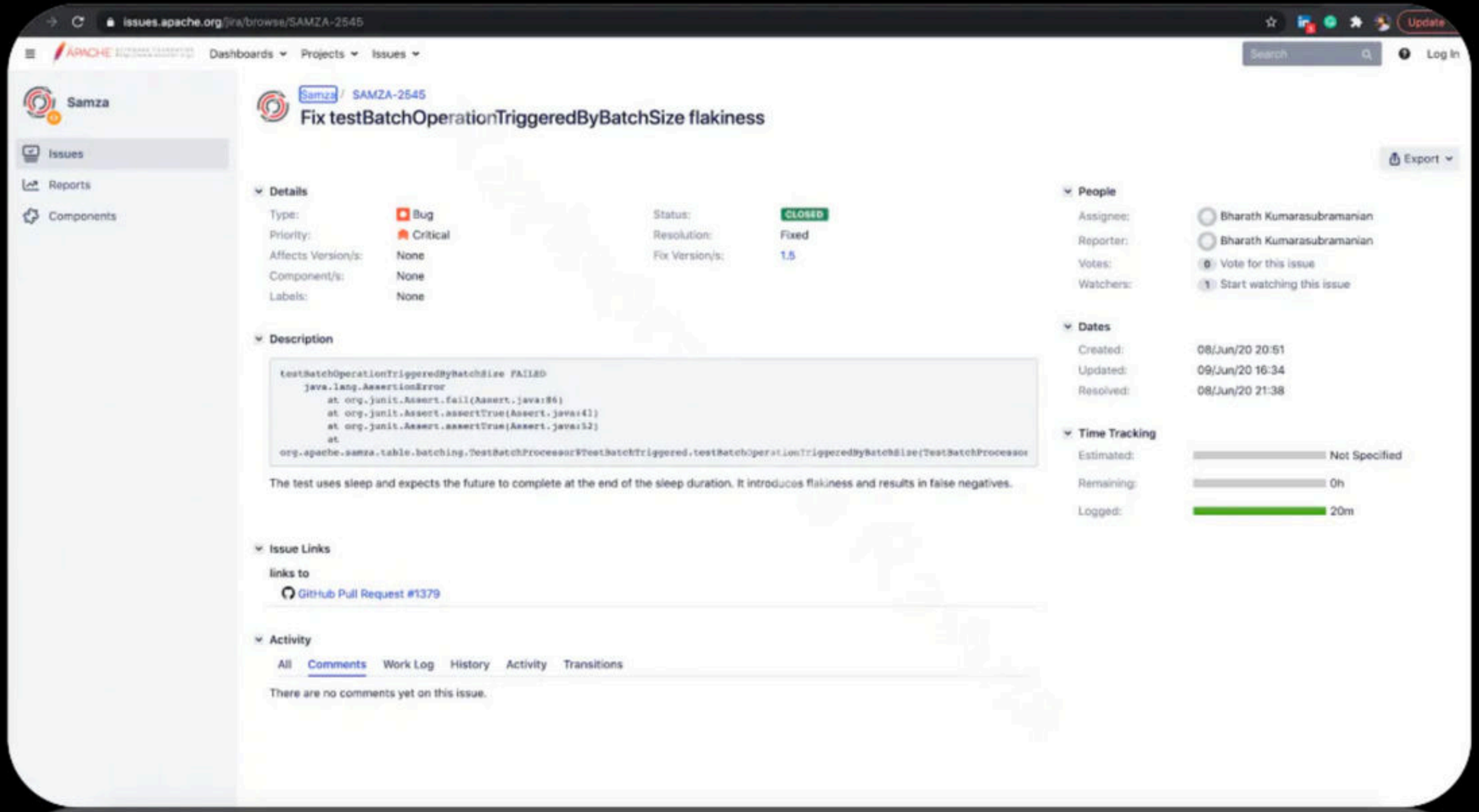
Being able to trace each change made to the software and connect it to bug tracking software such as Jira and being able to annotate each change with a message describing the purpose and intent of the change can help not only with root cause analysis but for doing other forensics as well.

If you look at the message which is left by the author,

you can see one ticket ID - ***SAMZA-2554***.

```
commit 1a05c487650fa3ba9fc568a8f2b0a08ca31b6ac6
Author: mynameborat <bharath.kumarasubramanian@gmail.com>
Date:   Mon Jun 8 14:36:29 2020 -0700

    SAMZA-2545: Fix testBatchOperationTriggeredByBatchSize flakiness

    **Problem**: The test uses sleep and expects the future to complete at the end of the sleep duration. It introduces flakiness and results in false negatives.
    **Change**: Modified the tests to use latch instead of sleep
    **Tests**: None
    **API Changes**: None
    **Upgrade Instructions**: None
    **Usage Instructions**: None

    Author: mynameborat <bharath.kumarasubramanian@gmail.com>

    Reviewers: Dengpanyin <dyin@linkedin.com>

    Closes #1379 from mynameborat/SAMZA-2545
```

This is actually a case which must have opened in some bug tracking software where you will get to know detailed description regarding why this change was required as shown below. Every Big organization uses such bug/issue tracking software. Every time you save a new version of your project, your VCS requires you to provide a short description of what was changed. Generally, you are required to put a Bug/Issue ID in that short description so that the changes can be traced.


issues.apache.org/jira/browse/SAMZA-2545
Update
Dashboards
Projects
Issues
Search
Log In
Samza
Issues
Reports
Components
Samza / SAMZA-2545
Fix testBatchOperationTriggeredByBatchSize flakiness
Export
Details
Type: Bug
Priority: Critical
Affects Version/s: None
Component/s: None
Labels: None
Status: CLOSED
Resolution: Fixed
Fix Version/s: 1.5
People
Assignee: Bharath Kumarasubramanian
Reporter: Bharath Kumarasubramanian
Votes: 0 Vote for this issue
Watchers: 1 Start watching this issue
Dates
Created: 08/Jun/20 20:51
Updated: 09/Jun/20 16:34
Resolved: 08/Jun/20 21:38
Time Tracking
Estimated: Not Specified
Remaining: 0h
Logged: 20m
Description
testBatchOperationTriggeredByBatchSize FAILED
java.lang.AssertionError
at org.junit.Assert.fail(Assert.java:86)
at org.junit.Assert.assertTrue(Assert.java:41)
at org.junit.Assert.assertTrue(Assert.java:52)
at
org.apache.samza.table.batching.TestBatchProcessor$TestBatchTriggered.testBatchOperationTriggeredByBatchSize(TestBatchProcessor
The test uses sleep and expects the future to complete at the end of the sleep duration. It introduces flakiness and results in false negatives.
Issue Links
links to
GitHub Pull Request #1379
Activity
All Comments Work Log History Activity Transitions
There are no comments yet on this issue.

Next time, when you are about to start a project, you will probably need a VCS if the answer to all the questions below is ***Yes!***

- Made a change to code, realized it was a mistake, and wanted to revert back?
- Had to maintain multiple versions of a software project?
- Wanted to review the history of some code? Who changed what in the past?
- Wanted to see the difference between multiple versions of your code?
- Wanted to share your code, or let other people work on your code?
- Wanted to experiment with a new feature without interfering with working code?

Available VCS in Market

WHAT IS GIT?
HOW IS IT DIFFERENT FROM GITHUB?
CONFIGURING GIT AND GITHUB

GIT Terminology

Repository

- A repository, or Git project, contains the entire collection of files and folders associated with a project, along with each file's revision history.

- In the case of GIT, there is a special directory "**.git**" in every git-based repo where GIT will keep all the metadata of all the changes to the files which you have instructed it to track for you.

- If you delete this folder, you will lose the functionality of version control.

- Remember that it is the brain of GIT.

Commit

- Remember when we said that Git saves every version of your project?
- Each version (or snapshot) is called a commit.
- Whenever you think you did a chunk of work that should be saved, you do a commit and you have another version of your work.
- Commits can be thought of as snapshots or milestones along the timeline of a Git project.
- Commits are created to capture the state of a project at that point in time.
- Each commit will be uniquely identified by an ID given to it.

What is GIT?

1

GIT is a free, open-source, and Distributed version control system

2

It is invoked from the command line of your machine to keep track of all files and modifications to those files in a repository (sometimes also called “repo”).

3

If you have a project under Git, you have access to all those functionalities of a VCS like rollback to another version, checking all the change history, etc

4

Git is the most widely used VCS and it has many integrations with 3rd party IDEs such as **IntelliJ and Bug tracking systems such as JIRA**

5

If you are an inexperienced developer wanting to build up valuable skills in software development, when it comes to version control,

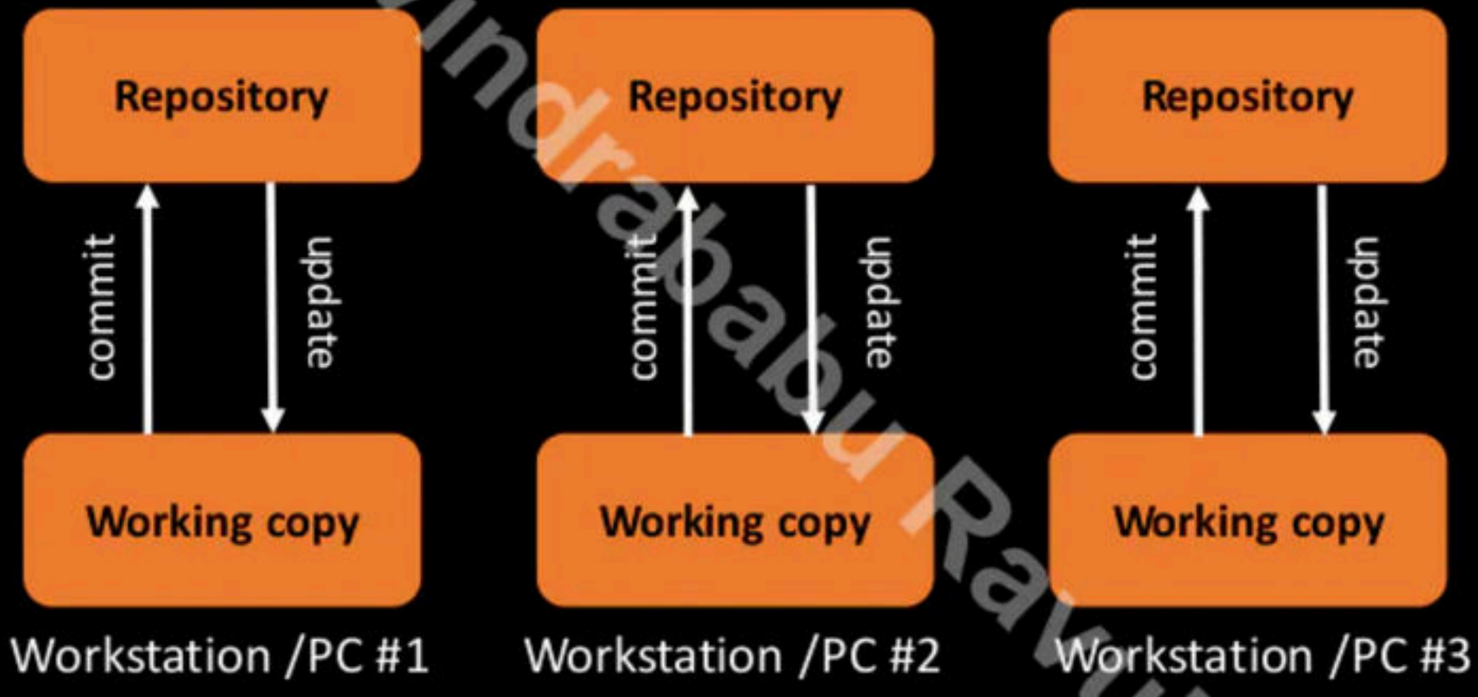
Git should be on your list!

How GIT is a Distributed or Decentralized VCS?

- GIT is distributed or decentralized in the sense that each person working on a project tracked through GIT will have its local repository which will contain the entire commit history of the project.

- GIT does not require you to connect to a single remote central repository server in order to perform the version control.

All the commits will be saved in the local repository as shown in the diagram below

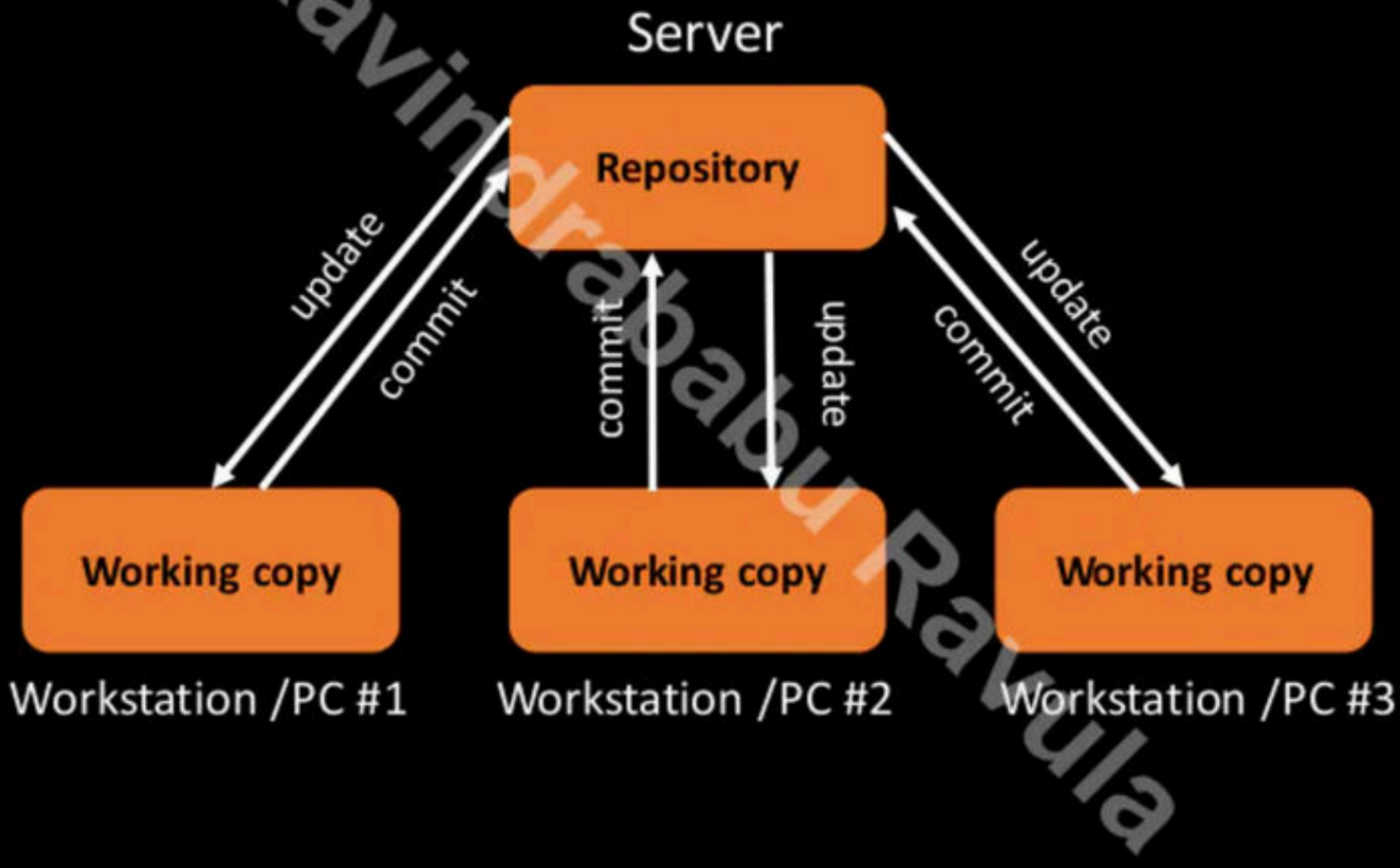


- Having a full local history makes Git fast since it means you don’t need a network connection to create commits, inspect previous versions of a file, or perform diffs between commits.

- Even if you are traveling somewhere with no internet connection, you will still be able to perform commits or revert back to the older commits.
- This is in complete contrast to Centralized version control systems like SVN where only the central repository has the complete version history.
- This means that users must communicate over the network with the central repository to obtain the change history of a project. They only share whatever single version a user has explicitly requested from the central server.

- Each time you want to record a commit/snapshot, you need to send it to the server. If you want to revert back to an older version, you still need to contact the remote server.
- It’s like the whole brain of VCS is sitting somewhere else and you need to contact it every time you perform any operations on your local machine.

Centralized version control System



By now, you must be thinking if GIT is responsible for effectively managing all the different versions of a project, then **why do we need GitHub?**

What is GitHub?

What is its purpose?

- GitHub is just a web hosting service where you can push your local Git repositories to start collaborating with the others. Thus, GitHub is completely unrelated to the original Git tool.
- You can use Git without GitHub.
- Git is the main program that actually tracks your changes, whereas GitHub is simply hosting your repositories online(and provides additional functionality not available in Git).
- Simply put, GitHub is just an online service that allows you to keep track of and share your Git version control projects outside of your local computer/server.

Here are some of the benefits of using GitHub

- It provides a backup of your files. In case you delete ".git" by mistake, it is present on GitHub.
- It gives you a visual interface for navigating your repos.
- It gives other people a way to navigate your repos.
- It makes repo collaboration easy (e.g., multiple people contributing to the same project).
- It provides a lightweight issue tracking system.

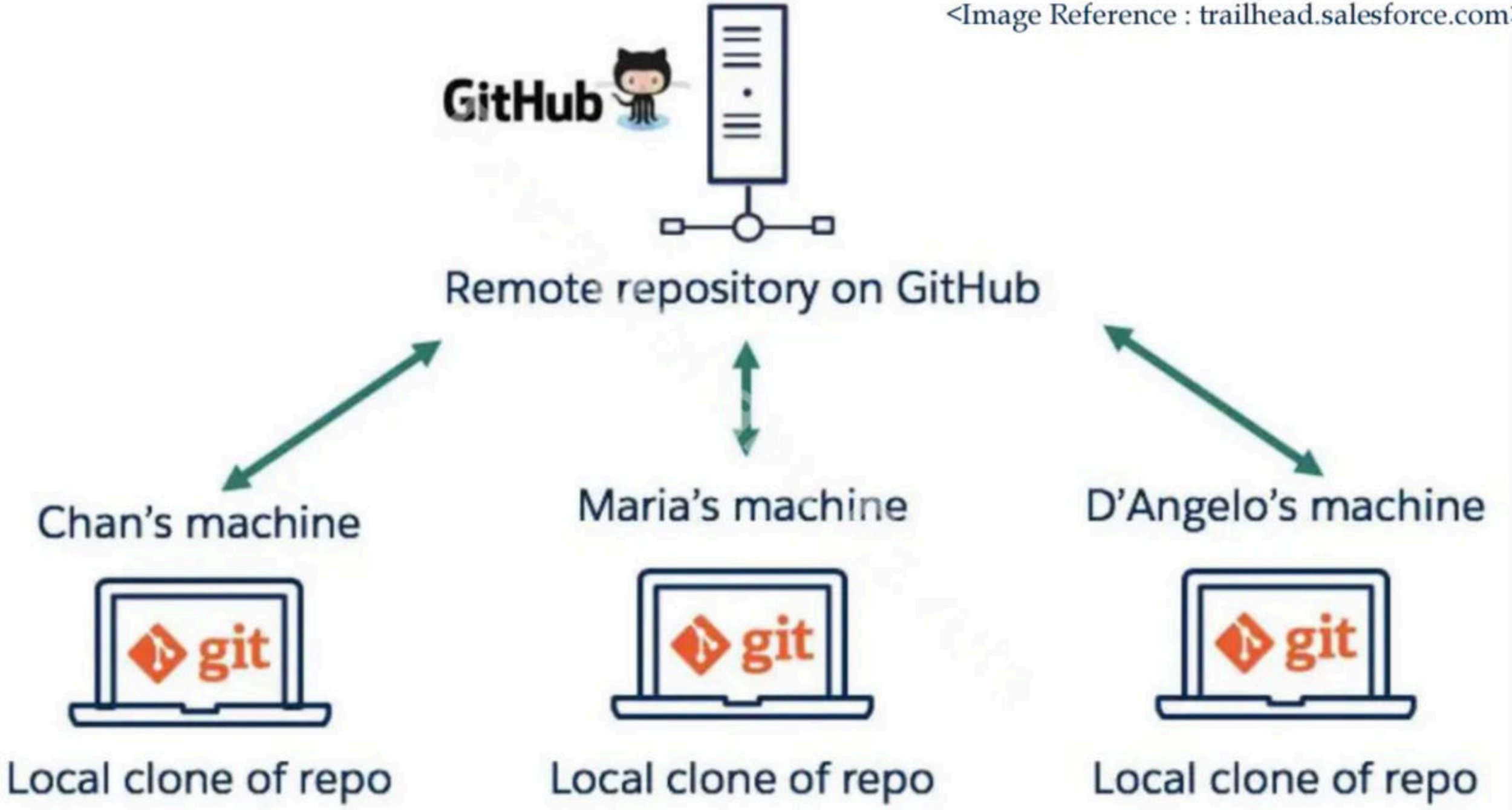
As said, GitHub is just used to store the remote copies of your Git repositories present locally on your machine. As an alternative, you can also use similar services available in the market such as GitLab, BitBucket, etc.

GIT + GITHUB TOGETHER

- Since Git is the most popular version control system these days, many software developers use it in their work. Git itself is used on the local computer of a user to manage and keep track of changes in a project over time.
- **Remember, everything is local in Git.**

- To collaborate (and also to simply store their code on a server), software teams then need to pick a code hosting platform.
- GitHub is the most popular among many options.

<Image Reference : trailhead.salesforce.com>



My philosophy

TEACHING IS WORSHIP

STUDENTS ARE GODS