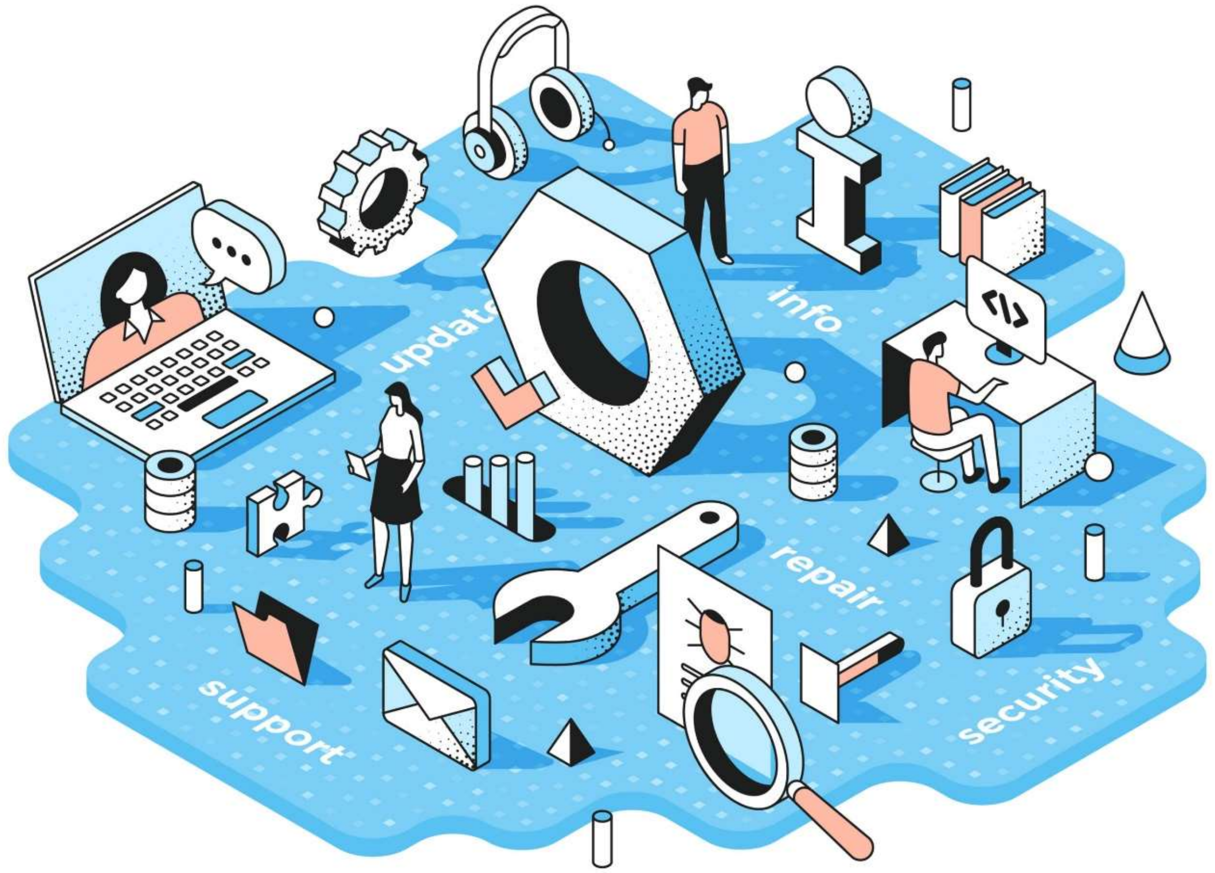
D8 UNIT 6-A CONFIGURATION MANAGEMENT AND PROTOTYPE

Software configuration management

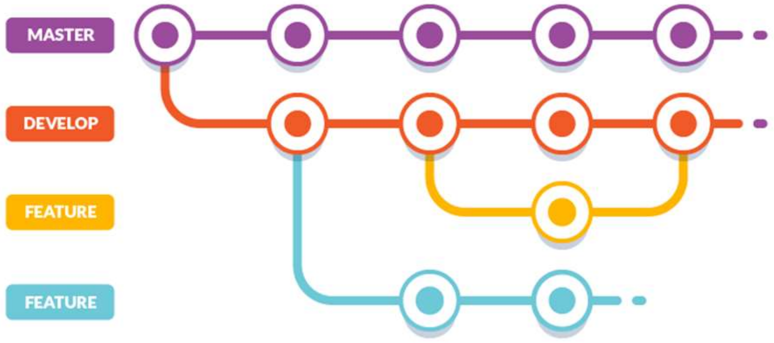
What is SCM?



What is SCM?

- **Software Configuration Management (SCM)** is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle.
- The primary goal is to increase productivity with minimal mistakes.
- SCM is part of cross-disciplinary field of configuration management.

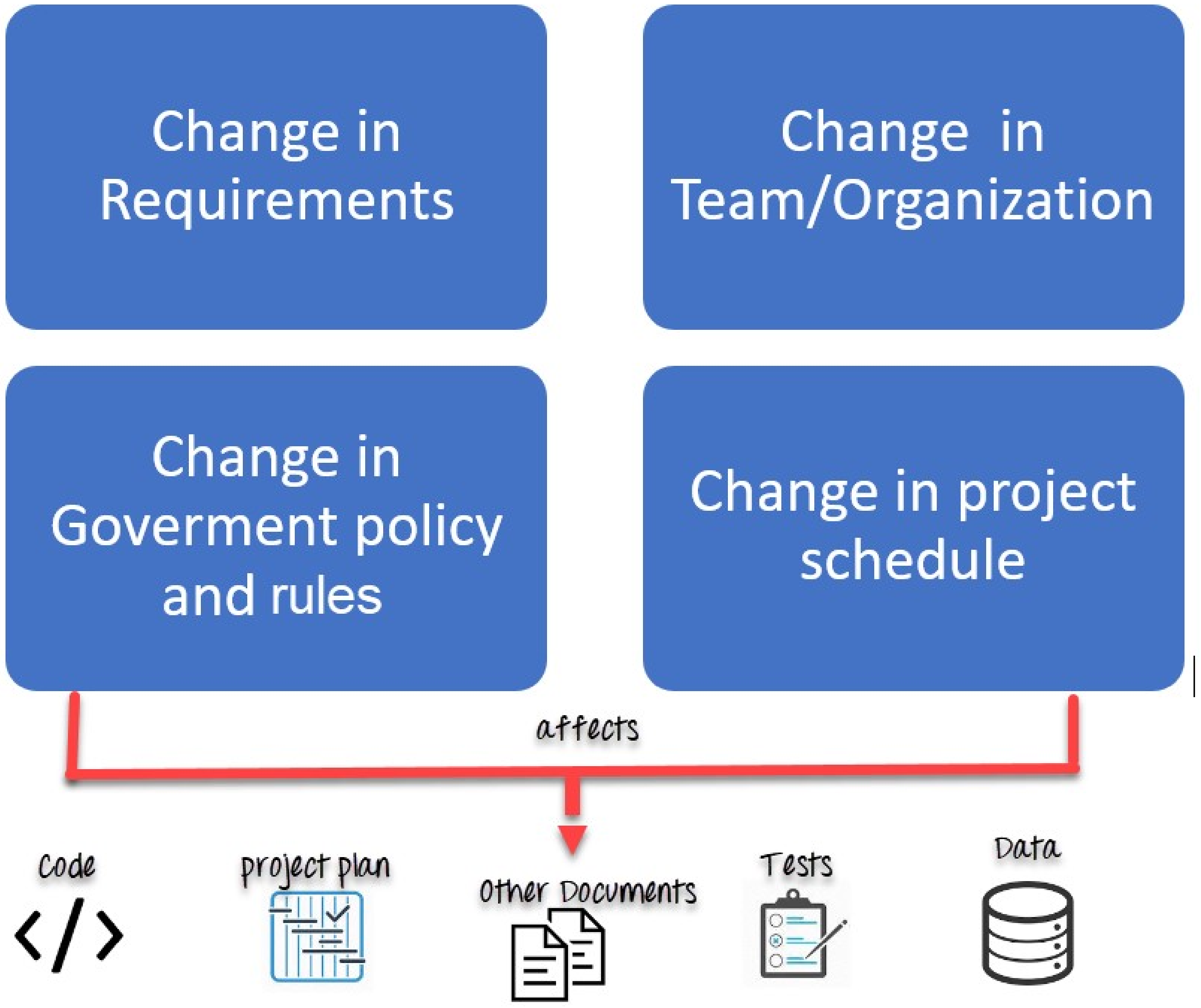
Why do we need configuration management?



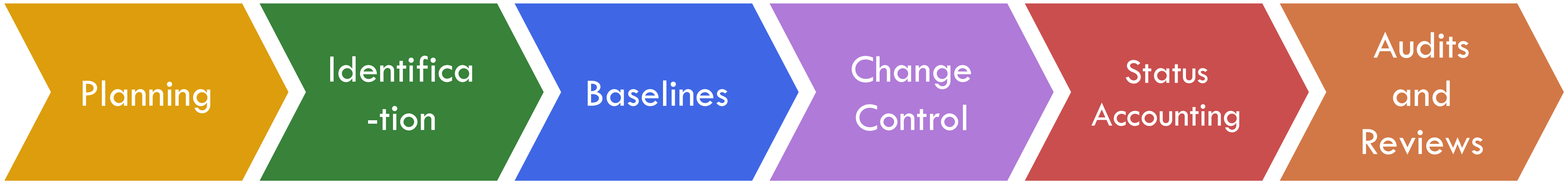
Why do we need configuration management?

- There are multiple people working on software.
- Multiple versions, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently.
- Changes in user requirements, policies, budget, schedule.. need to be accommodated.
- Software should be able to run on various machines and operating systems.
- It helps to develop coordination among stakeholders.
- Is also beneficial to control the costs involved in making changes to a system.

Why do we need configuration management?


Change in Requirements
Change in Team/Organization
Change in Goverment policy and rules
Change in project schedule
affects
Code
project plan
Other Documents
Tests
Data

Configuration management process



Configuration management process Planning

- The Software Configuration management plan (SCMP) begins at the early phases of a project.
- The SCMP:
  - Can follow a public **standard** like the IEEE 828 or organization specific standard.
  - Defines the types of **documents** to be managed and a document naming.
  - Defines the person who will be **responsible** for the entire SCM process and creation of baselines.
  - Fixes **policies** for version management & change control.
  - Defines **tools** which can be used during the SCM process.
  - Creates a configuration management **database** for recording configuration information.

Configuration management process
Configuration identification

The following products must be selected and identified:

- Products **delivered** to the customer
- Designated **internal** work products
- **Acquired** products
- Tools and other **assets** of the project's work environment
- **Other items** used in creating and describing these work products like source code modules, test cases, or requirements specification.

Configuration management process
Configuration identification

How do we identify items?

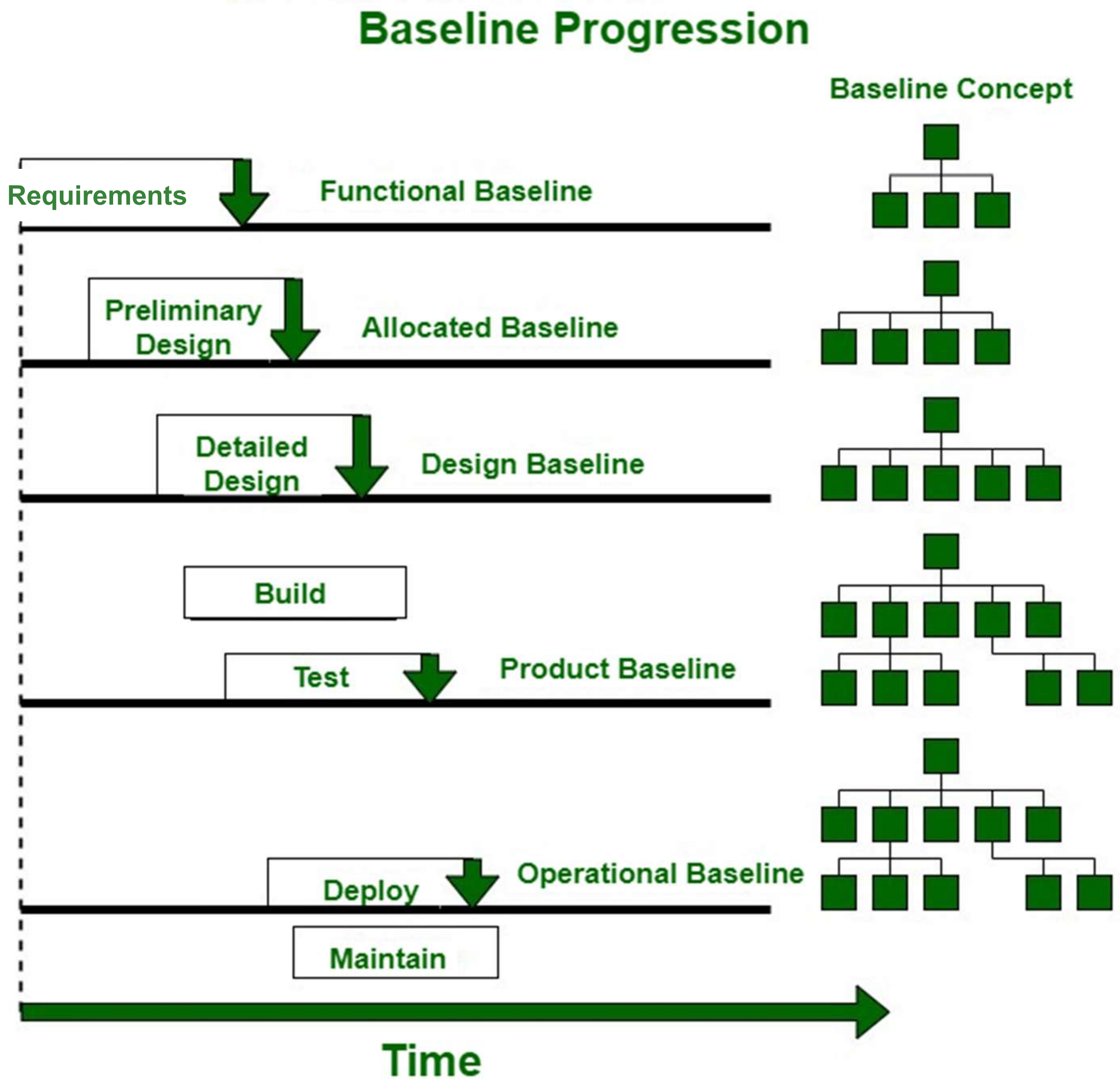
- Assign unique identifiers to configuration items.
- Specify the important characteristics of each configuration item (author, document or file type, programming language for software code files…).
- Identify the owner responsible for each configuration item.
- Specify relationships among configuration items.
- Include details of what, why, when and by whom changes in the test are made.
- List of resources required such as file, tools, etc.

Configuration management process
Baselines

- A baseline is a specification or product that has been formally reviewed and agreed upon, serves as the basis for further development, and can be changed only through formal change control procedures.
- It is a formally accepted version of one or more software configuration items.
- It is designated and fixed at a specific time while conducting the SCM process.
- It can only be changed through formal change control procedures.

Configuration management process

Baselines

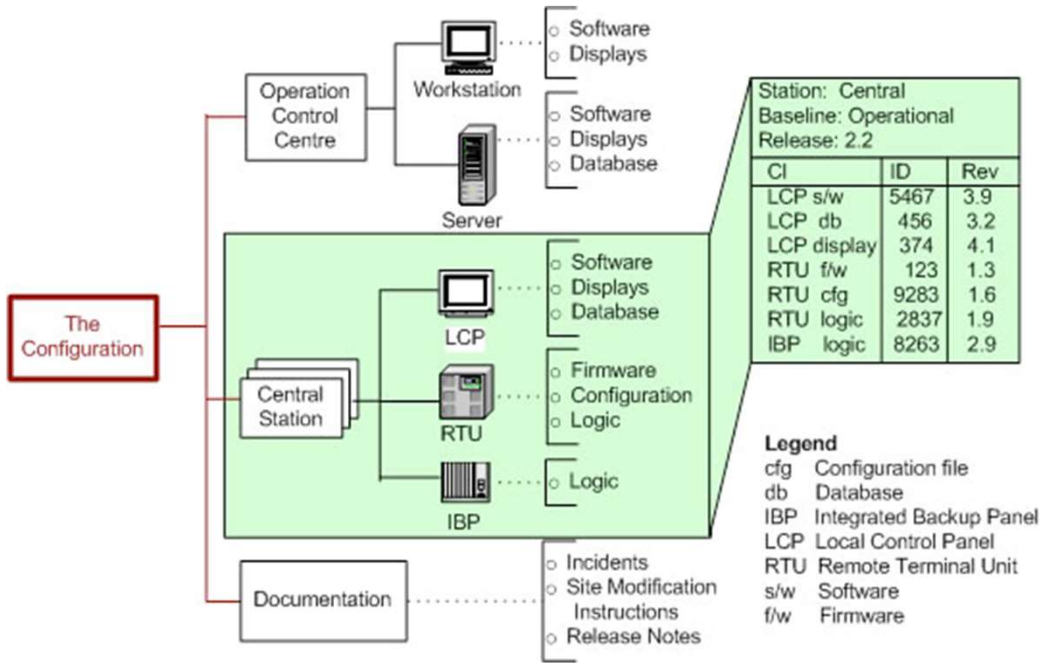


Configuration management process
Change control

- Change control is a procedural method which ensures quality and consistency when changes are made in the configuration object.
- The change requests are submitted to software configuration manager.
- Control changes to build a stable software development environment.
- The request will be checked based on the technical merit, possible side effects and overall impact on other configuration objects.

Configuration management process
Configuration status accounting

- Configuration status accounting tracks each release during the SCM process.
- This stage involves tracking what each version has and the changes that have led to this version.



Station: Central
Baseline: Operational
Release: 2.2

CI	ID	Rev
LCP s/w	5467	3.9
LCP db	456	3.2
LCP display	374	4.1
RTU f/w	123	1.3
RTU cfg	9283	1.6
RTU logic	2837	1.9
IBP logic	8263	2.9

Legend

cfg	Configuration file
db	Database
IBP	Integrated Backup Panel
LCP	Local Control Panel
RTU	Remote Terminal Unit
s/w	Software
f/w	Firmware

Configuration management process
Configuration status accounting

Activities related to status accounting:

- Keep a record of all the changes made to the previous baseline to reach a new baseline.
- Identify all items to define the software configuration.
- Monitor status of change requests.
- Allow tracking of progress to next baseline.
- Allow to check previous releases/versions to be extracted for testing.

Configuration management process
Configuration audits and reviews

- Software Configuration audits verify that the **software product satisfies the baseline needs**.
- It is an examination of the software product to verify, via testing, inspection, demonstration, or analysis results, that the product has met the requirements specified in the functional baseline documentation.
- Configuration auditing is conducted by auditors by checking that **defined processes are being followed** and ensuring that the SCM goals are satisfied.

Configuration management process
Configuration audits and reviews

Activities related to audits and reviews:

- Verify compliance with configuration control standards.
- Ensure that traceability is maintained during the process.
- Ensure that changes made to a baseline comply with the configuration status reports.
- Validate completeness and consistency.

Participants in SCM

PROJECT MANAGER

- Ensures that the product is developed within a certain time frame.
- Monitors the progress of development and recognizes issues in the SCM process.
- Generates reports about the status of the software system.
- Makes sure that processes and policies are followed for creating, changing, and testing.

CONFIGURATION MANAGER

- Responsible for identifying configuration items.
- Ensures team follows the SCM process.
- Needs to approve or reject change requests.

AUDITOR

- Responsible for SCM audits and reviews.
- Ensures the consistency and completeness of the releases.

USER

- Should understand the key SCM terms to ensure he has the latest version of the software

DEVELOPER

- Needs to change the code as per standard development activities or change requests.
- Responsible for maintaining configuration of code.
- Should check the changes and resolve conflicts.

SCM tools

Any change management software should have the following 3 key features:

- **Concurrency Management:**

  Concurrency in context to SCM means that the same file is being edited by multiple persons at the same time.

- **Version Control:**

  SCM uses an archiving method or saves every change made to file. With the help of archiving or save feature, it is possible to roll back to the previous version in case of issues.

- **Synchronization:**

  Users can checkout more than one file or an entire copy of the repository. The user then works on the needed file and checks in the changes back to the repository. They can synchronize their local copy to stay updated with the changes made by other team members.

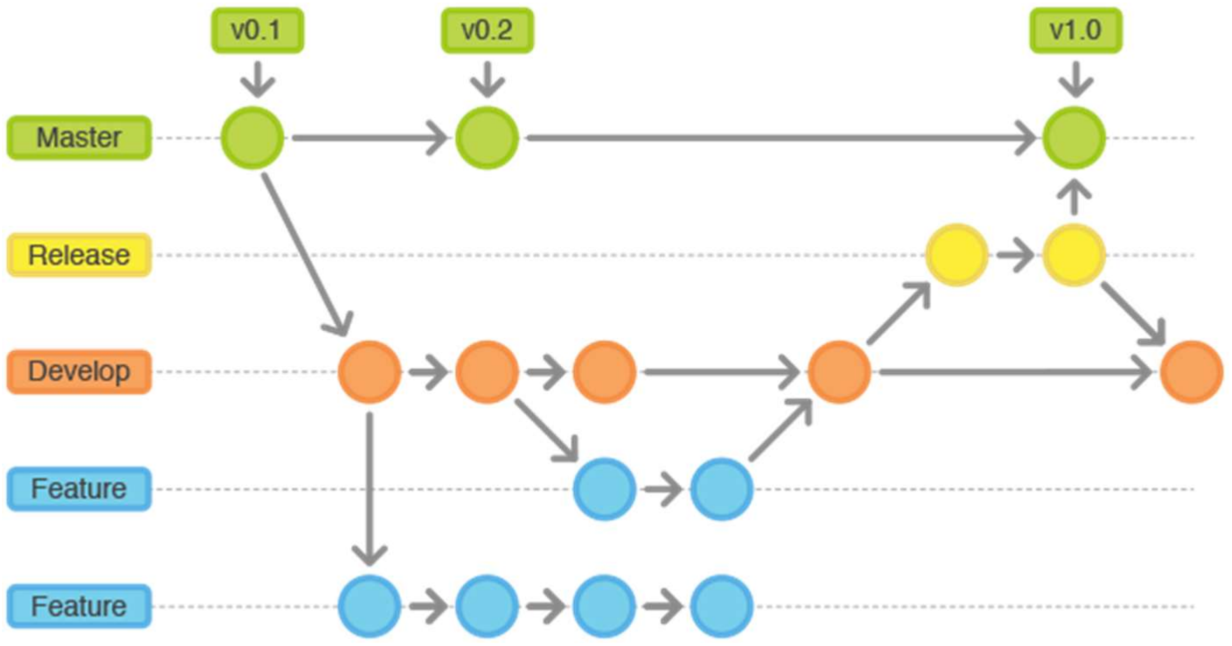
SCM tools

Git is a **Distributed Version Control System.**

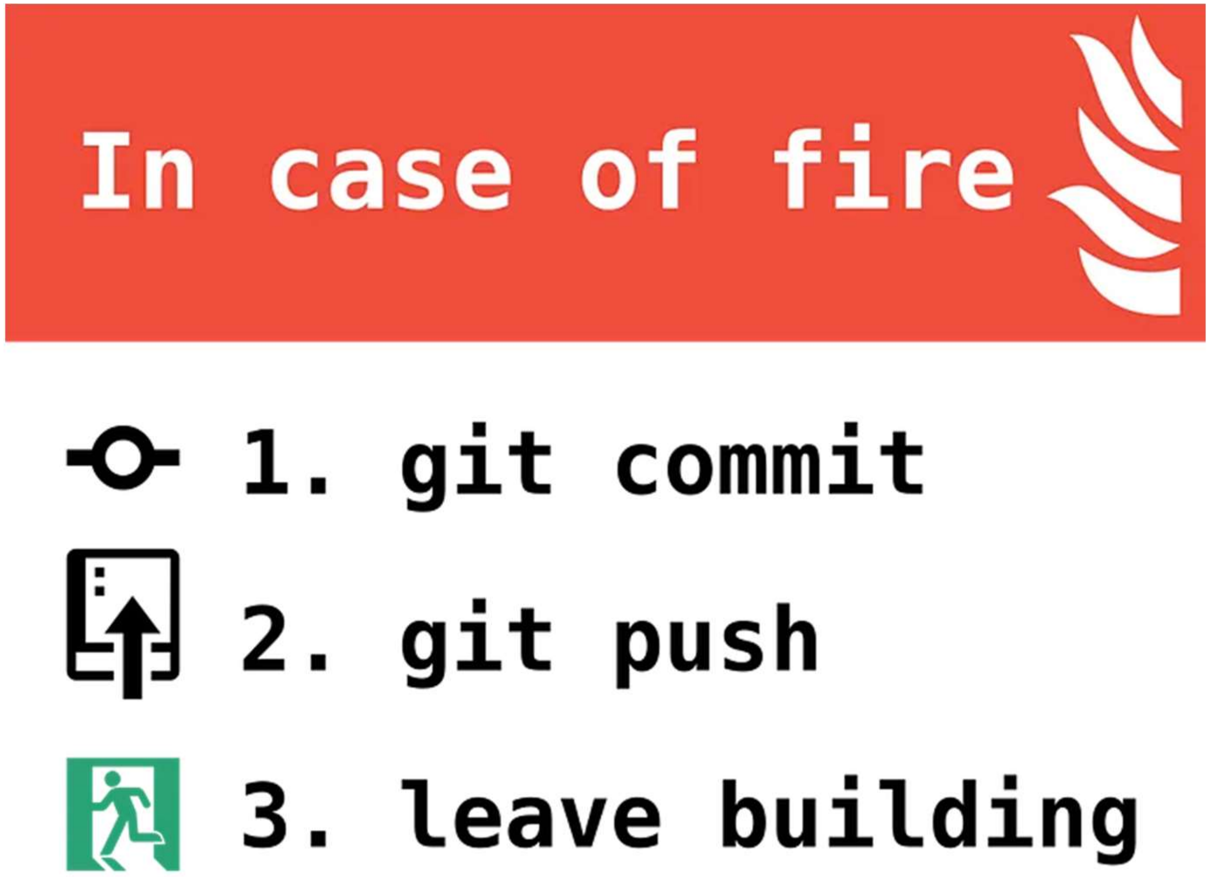
- **Control System:** Git is a content tracker. It is mostly used to store code due to the other features it provides.
- **Version Control System**: It helps by maintaining a history of what changes have happened and provides features like branches and merges.
- **Distributed Version Control System**: The code is not just stored in a central server, but it is present in every developer’s computer.

SCM tools

- A version control system like Git is needed to ensure there are no code conflicts between the developers.
- Allows developers to revert and go back to an older version of the code.
- The concept of branching in Git is very important, when several projects/versions which are being run in parallel involve the same codebase.

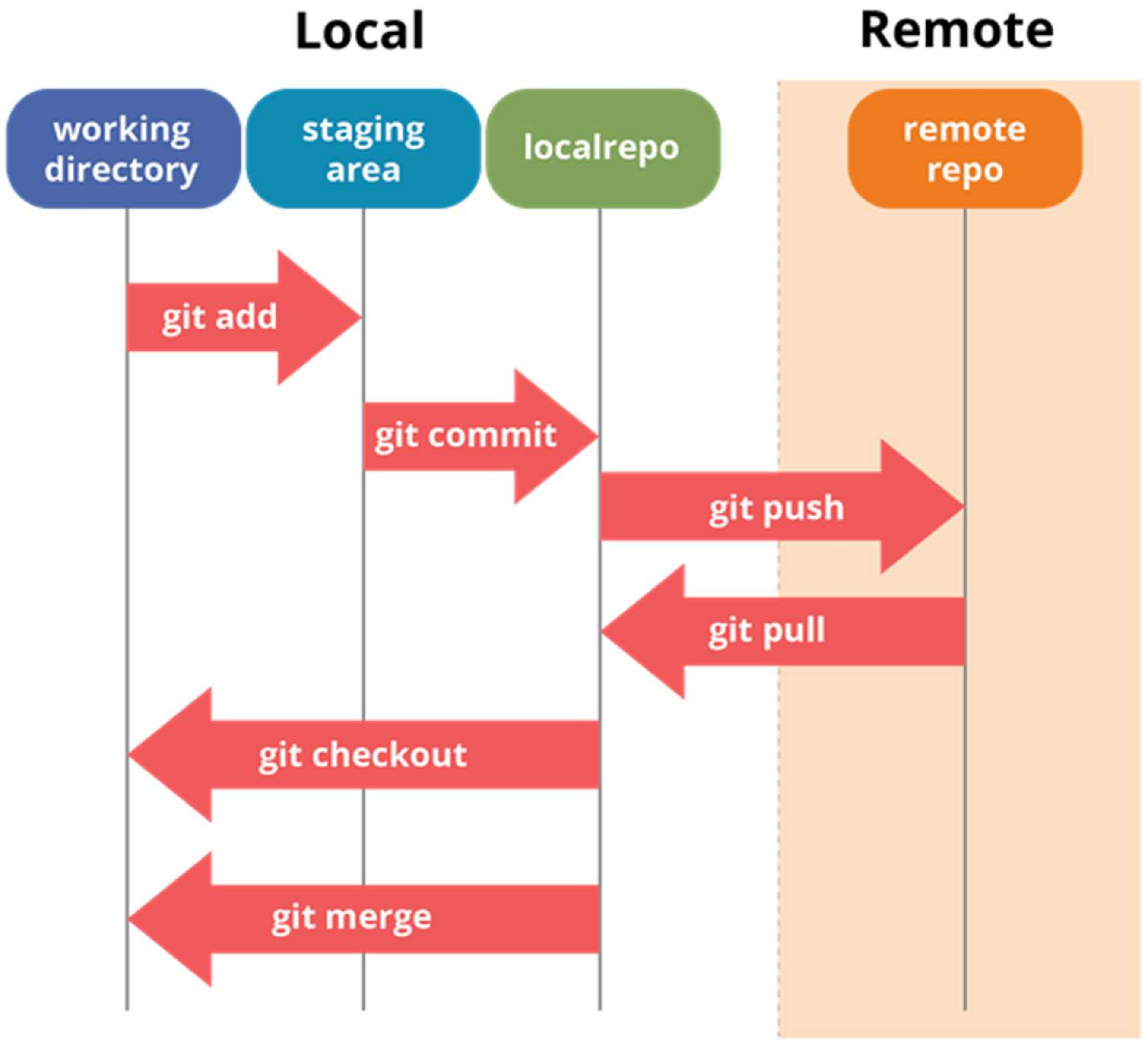


SCM tools

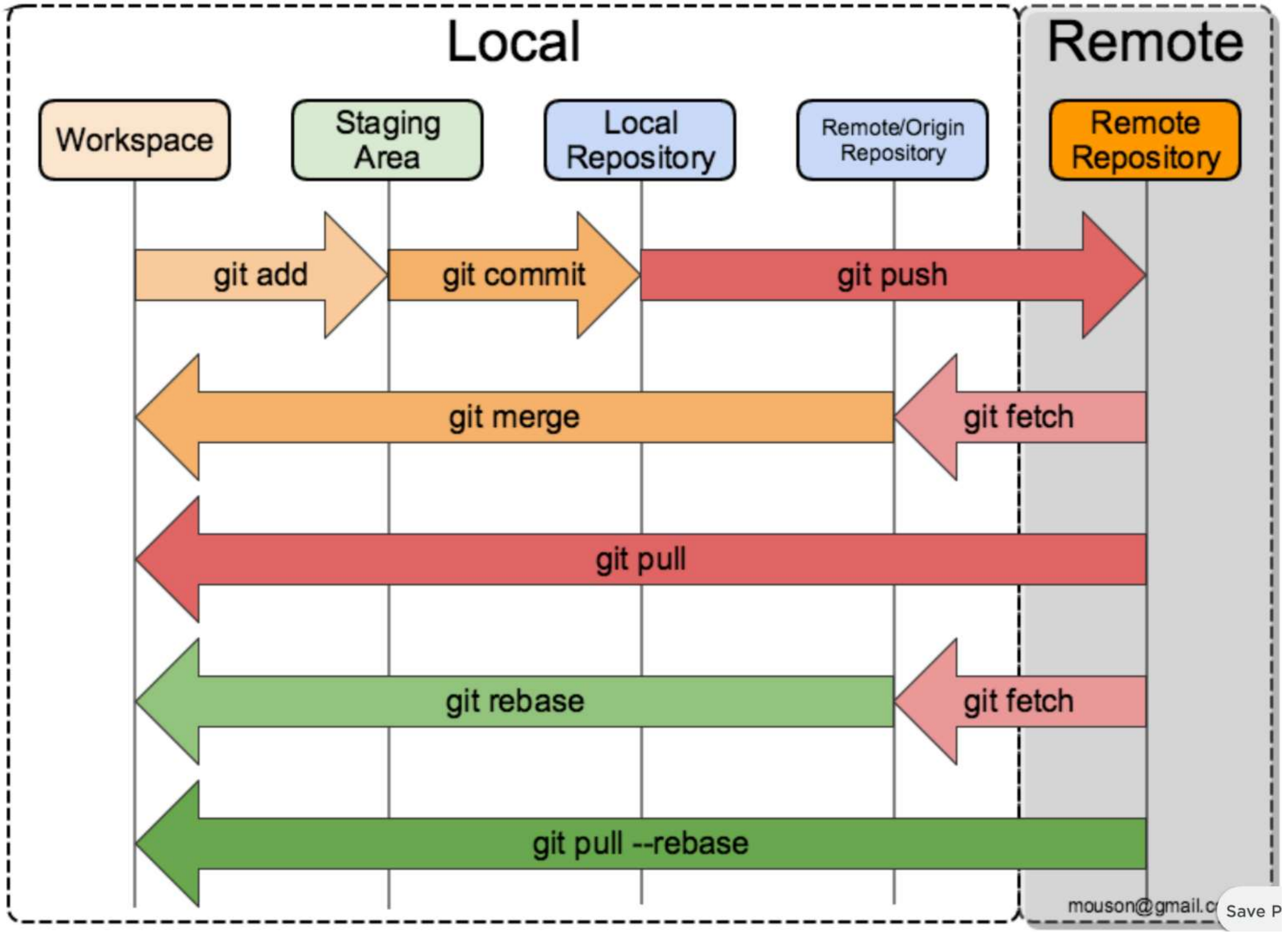

In case of fire

1. git commit
2. git push
3. leave building

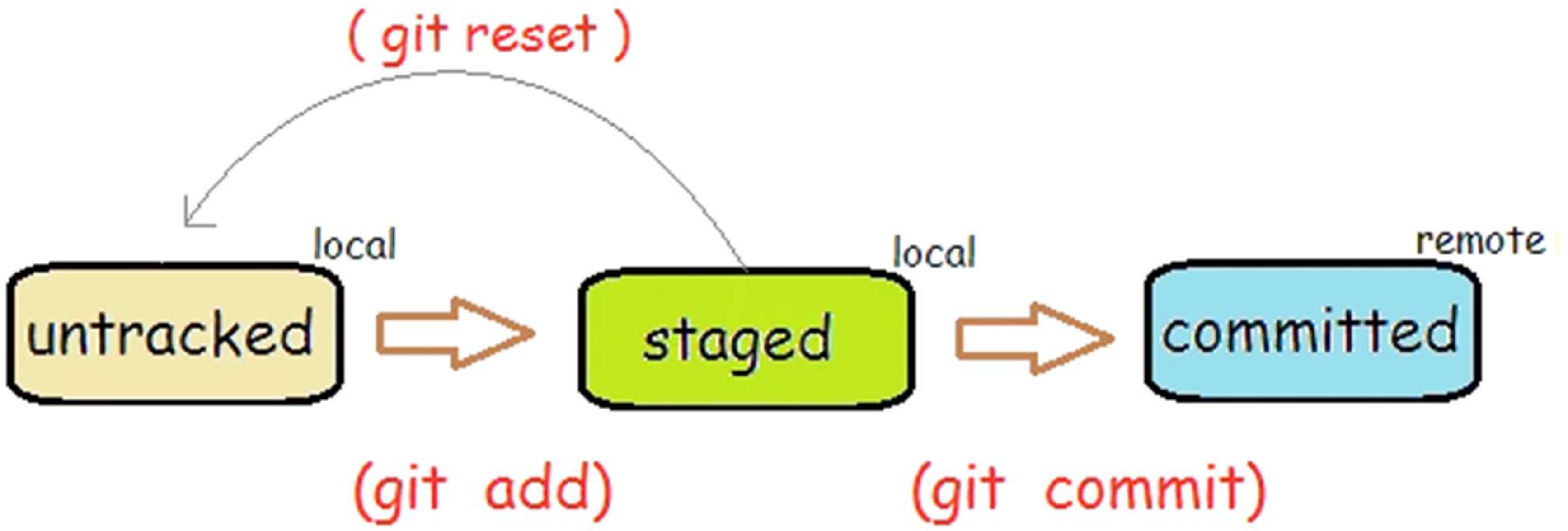
SCM tools


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout
git merge

SCM tools


Local
Remote
Workspace
Staging Area
Local Repository
Remote/Origin Repository
Remote Repository
git add
git commit
git push
git merge
git fetch
git pull
git rebase
git fetch
git pull --rebase
mouson@gmail.c
Save P

SCM tools


(git reset)
local
local
remote
untracked
staged
committed
(git add)
(git commit)

SCM tools

git add

- Moves changes from the working directory to the staging area. This gives you the opportunity to prepare a snapshot before committing it to the official history.

git branch

- This command is your general-purpose branch administration tool. It lets you create isolated development environments within a single repository.

git commit

- Takes the staged snapshot and commits it to the project history. Combined with git add, this defines the basic workflow for all Git users.

git fetch

- Downloads a branch from another repository, along with all of its associated commits and files. But it doesn't try to integrate anything into your local repository. This gives you a chance to inspect changes before merging them with your project.

git merge

- A powerful way to integrate changes from divergent branches. After forking the project history with git branch, git merge lets you put it back together again.

SCM tools

git pull

- It downloads a branch from a remote repository, then immediately merges it into the current branch.

git push

- Pushing is the opposite of fetching (with a few caveats). It lets you move a local branch to another repository, which serves as a convenient way to publish contributions.

git rebase

- Rebasing lets you move branches around, which helps you avoid unnecessary merge commits. The resulting linear history is often much easier to understand and explore.

git revert

- Undoes a committed snapshot. When you discover a faulty commit, reverting is a safe and easy way to completely remove it from the code base.

SCM tools

git checkout

- In addition to checking out old commits and old file revisions, git checkout is also the means to navigate existing branches. Combined with the basic Git commands, it's a way to work on a particular line of development.

git clean

- Removes untracked files from the working directory. This is the logical counterpart to git reset, which (typically) only operates on tracked files.

git clone

- Creates a copy of an existing Git repository. Cloning is the most common way for developers to obtain a working copy of a central repository.

git reset

- Undoes changes to files in the working directory. Resetting lets you clean up or completely remove changes that have not been pushed to a public repository.

git status

- Displays the state of the working directory and the staged snapshot. You'll want to run this in conjunction with git add and git commit to see exactly what's being included in the next snapshot.

SCM tools

Other tools

IBM Rational Synergy

HYSCALE wavemaker

RUDDER

SUBVERSION

juju

auvik

ConfigHub
Write better code. Reduce errors. Discover new possibilities.

CHEF

TC

Alibaba Group

solarwinds

CFEngine

ManageEngine
Desktop Central

Spinnaker

puppet labs

SALTSTACK

References

- https://www.guru99.com/software-configuration-management-tutorial.html