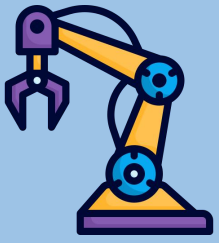


# Robotics Corner





# Robotics Corner

---

## Tooling SCM

Quique intro to the lesson and what it's talking about.



01

What is Configuration Management?

02

Why Configuration Management?

03

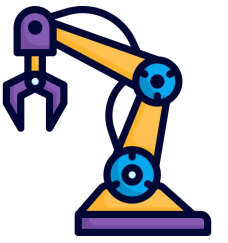
SCM Process and Roles

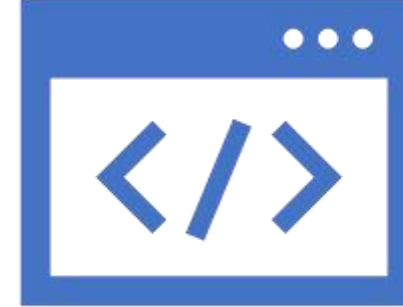
04

Tools

05

Getting Started with Git





SCM is the task of tracking and controlling Software Configuration items.

CI : Configuration item is



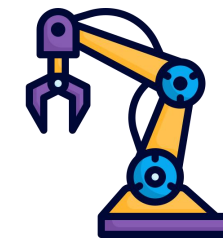
SCM concerns answering the question of “Somebody did something , how can someone else reproduce it?”



## Introduction to SCM parameters

- Multiple Users/Developers
- Repository :It is a central database multiple files
  - It is server based
  - Local on your machine
  - Web based on a remote server
- User/access rights
- Artifacts





## Purposes of SCM

Build management

Managing the process and tools used for builds.

Ensuring adherence to the organization's development process.

Managing the software and hardware that host the system.

Facilitate team interactions related to the process.

Making sure every defect has traceability back to the source.

- Process management

- Environment management

- Teamwork

- Defect tracking



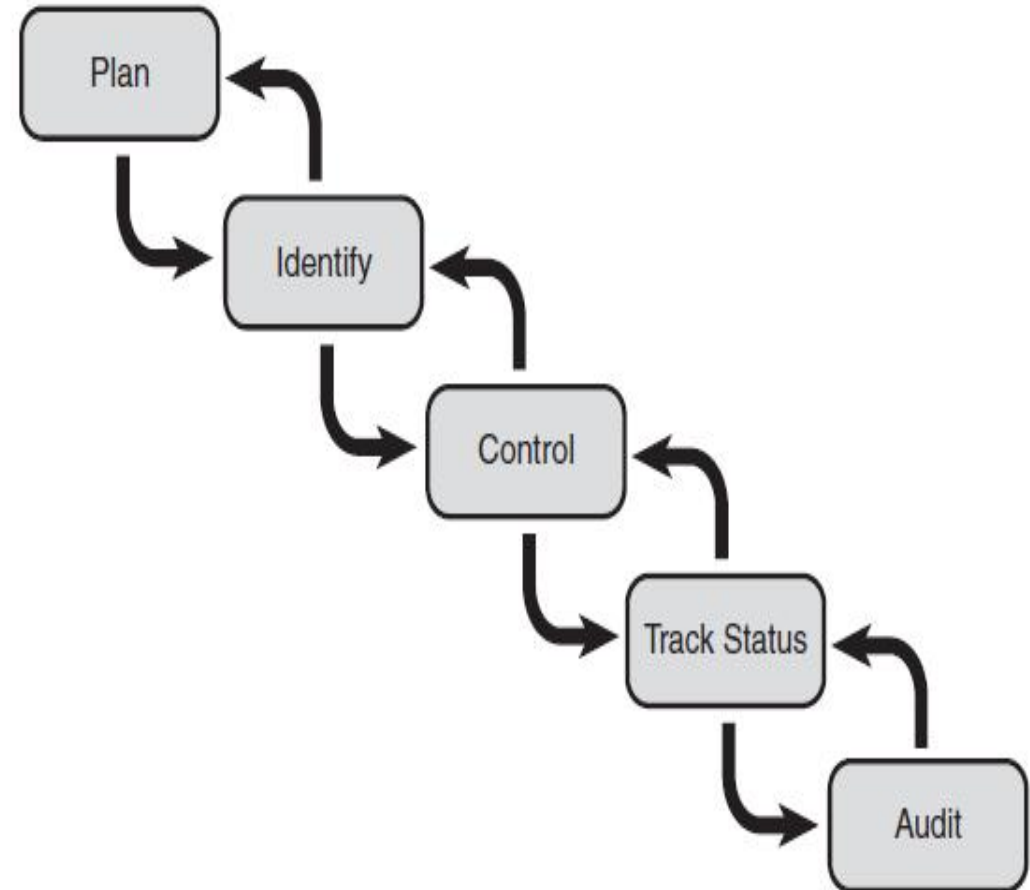






# SCM Process

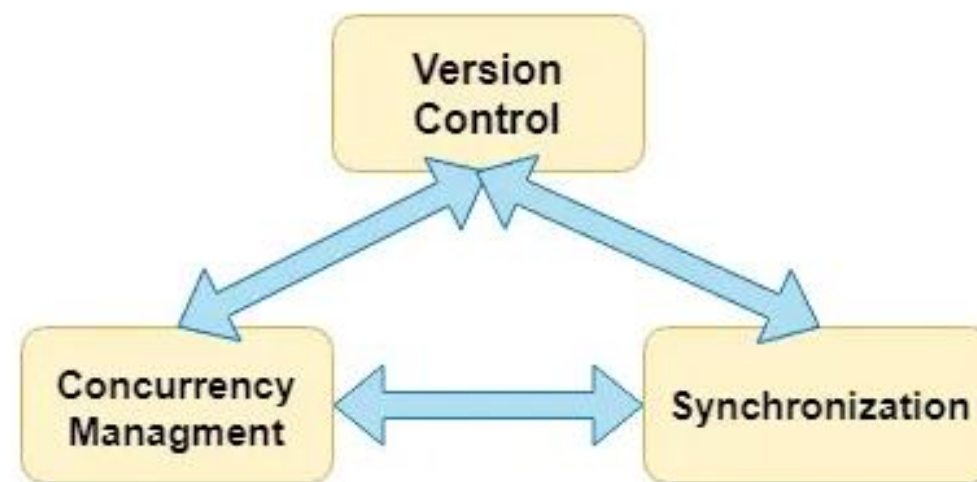
1. To identify all items that collectively define the software configuration
2. To manage changes to one or more of these items
3. To facilitate the construction of different versions of an application
4. To ensure that software quality is maintained as the configuration evolves





# SCM Features

- **Version Control:** Version control allows users to manage and organize the versions of their work. Apart from this, it maintains the older versions of the work so that the user can roll back to the previous stage at any point of time.
- **Concurrency management:** This feature enables multiple users to work on the same file without losing anyone's work. Usually, the development team consists of several developers and there will be a situation that more than one developer attribute allows merging the changes from both the developers.
- **Synchronization:** This feature allows team members to be in sync with each other's work, if you are modifying a file that is shared with others and they modify it too, this feature updates your local ones before you can upload the new.



Configuration management features

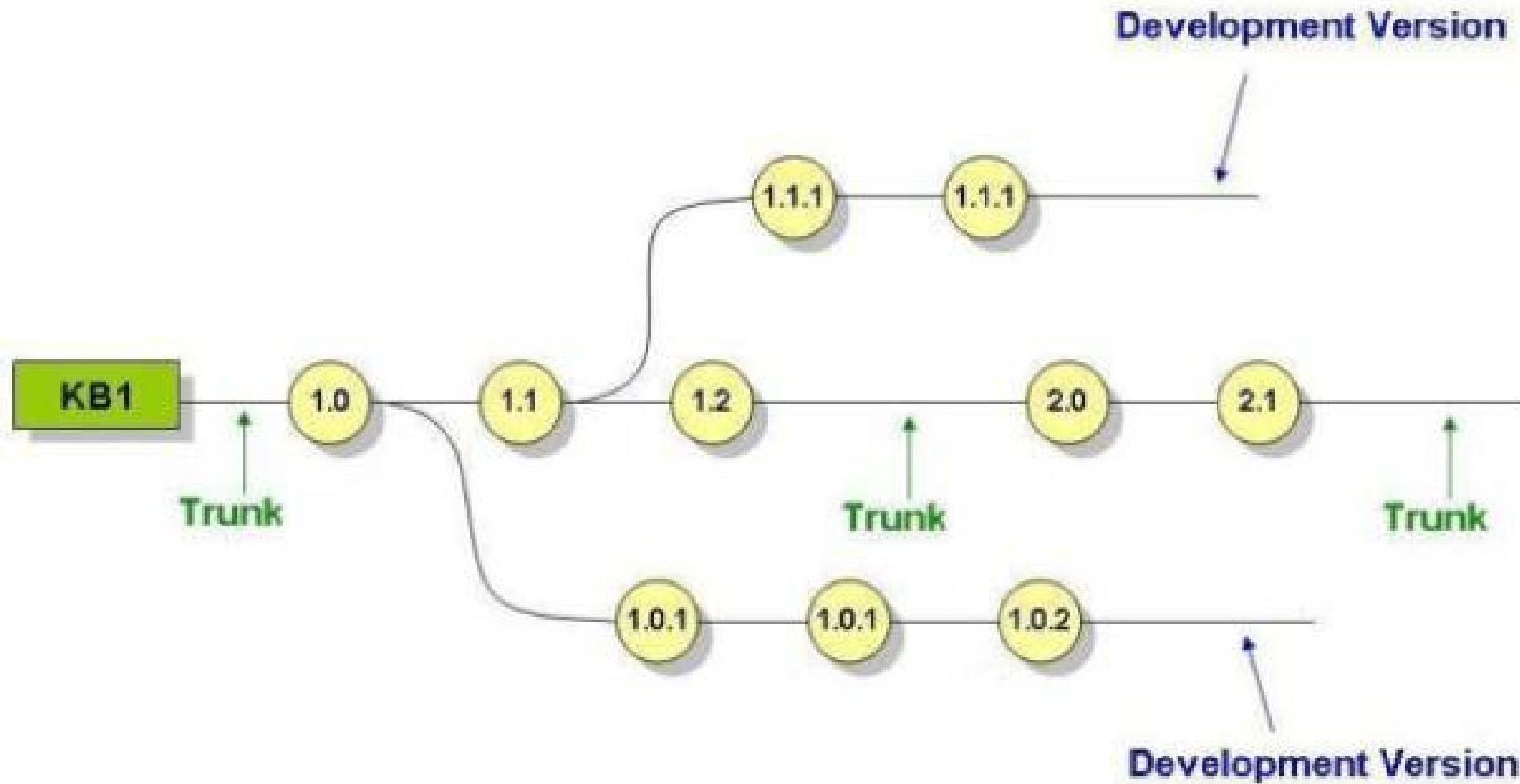
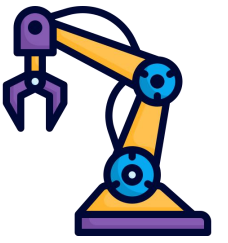


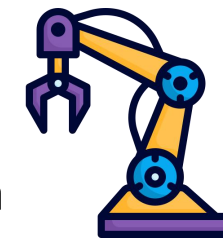
# Version Control

1. Release: is also called major and which is delivered to the customer and increased when making new API(Application Programmable Interface) as a new release
2. Version: as minor is increased when delivering new features with the same API
3. Revision: also called patch and increased when bug fixing



# Baseline





# SCM Scenario “1”

1. I want to see the code
  - Check out if the artifact is exist or not (it returns a copy in local PC)
2. Modify the code and upload it
  - Check-in (commit)
3. Another one needs to get the updated Repo to his local copy
  - Update sees which files were modified and update them
4. What if the 1<sup>st</sup> has wrong changes
  - In case of text: it save the changes only
  - In case of binary files: it saves the complete files
  - This makes each have different versions
  - Revert back to older version
5. To have the latest version
  - Top of the trunk





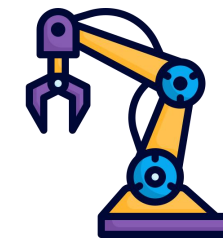
# SCM Scenario “2”

If 2 **Developers** want to change in **the same file** at **same time**

1. **Developer#1**: Make change and upload it
2. **Developer#2**: Make change and wants to upload it

- The **repository** refuse to save the changes (the file on the server is more recent than you have).
- Make **update** (this make merge between the 2 files if and only if the 2 changes are separated in lines) if not the tool return conflict no update (change it manual).
- upload the Changes





# SCM Servers

Software configuration management is normally supported by tools with different functionality.

## Examples:

**RCS** – very old but still in use; only version control system

**CVS** – based on RCS, allows concurrent working without locking

**Perforce** – Repository server; keeps track of developer's activities

**ClearCase** – Multiple servers, process modeling, policy check mechanisms

15



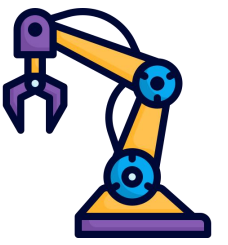


# Lab 1

1. Register Account on github.com
2. Create your own Repository on github
3. Use Git clone to download a copy of the Repo.
4. Update the files in the directory of the Repo.
5. For the first time using git you need to configure your user.email and user.name using:
  - `git config --global user.email "email@domain.com"`
  - `git config --global user.name "name"`
6. Use `git add .` ( to add files to a buffer that will be uploaded to the remote server)
7. Use `git commit -m "message"` (specify your own commit message)
8. Using `git push -u origin master` to upload your files







## Lab 2

1. Use git init to start a Repo on your local machine
2. Use git add . ( to add files to a buffer that will be uploaded to the remote server)
3. Use git commit -m "message" (specify your own commit message)
4. Using git push -u origin master to upload your files





## Lab 3

Create branch other than the master branch to be called dev

- Git branch dev
- Git branch – list to know how many branches

Use git status

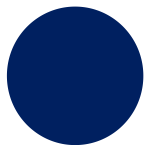
Change Head pointer to the dev branch instead of pointing to master branch

- To switch to the other branch use git checkout dev

Use git commit –m “message” (specify your own commit message)

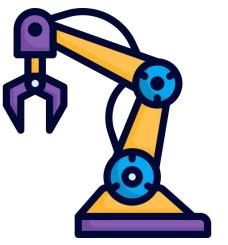
Using git push –upstream origin dev to upload your files





- Software Configuration Management Video : <https://www.youtube.com/watch?v=AaHaLjuzUm8>
- Download Git : <https://git-scm.com/downloads>
- Git tutorial : <https://www.youtube.com/watch?v=pDmYNK68IEc>





# Thank You

Do you have any questions?



**01211626904**



**[www.roboticscorner.tech](http://www.roboticscorner.tech)**



**Robotics Corner**



**Robotics Corner**



**Robotics Corner**



**Robotics Corner**