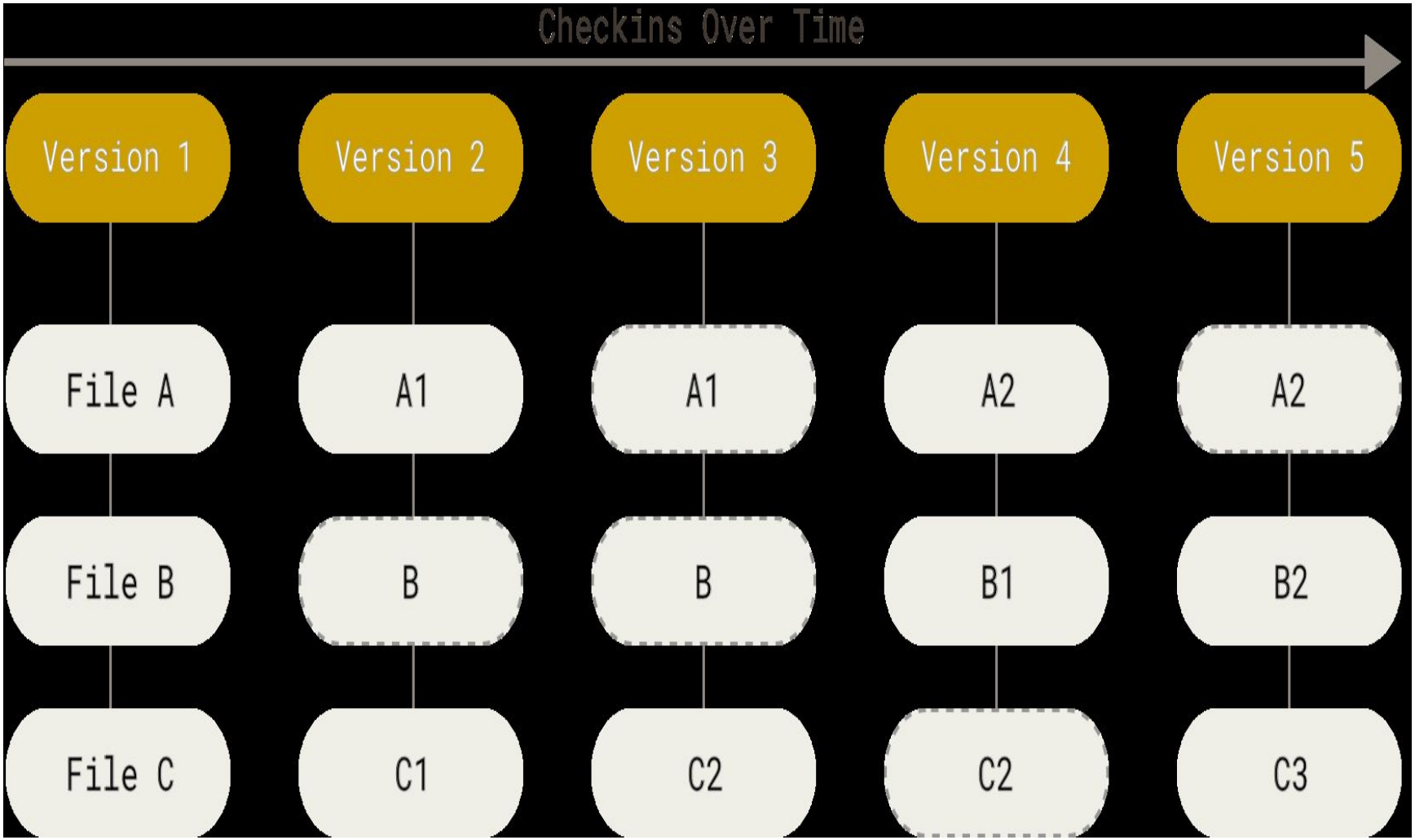
# Git

- Linux development developed their own tool. Some of the goals of the new system were as follows:
    - Speed
    - Simple design
    - Strong support for non-linear development
    - Fully distributed
    - Able to handle large projects like the Linux kernel efficiently

- Git thinks of its data more like a series of snapshots of a miniature file system.
- Every time when committing, or saving the state of project, Git basically takes a picture of what all files look like at that moment and stores a reference to that snapshot.
- If files have not changed, Git doesn’t store the file again, just a link to the previous identical file it has already stored.
- Git thinks about its data more like a **stream of snapshots.**

# *Storing data as snapshots of the project over time*



# Git Has Integrity

- Everything in Git is check summed before it is stored and is then referred to by that checksum.
- It’s impossible to change the contents of any file or directory without Git knowing about it.
- This functionality is built into Git at the lowest levels and is integral to its philosophy.
- Information is not lost in transit or get file corruption without Git being able to detect it.

- The mechanism that Git uses for this checksumming is called a SHA-1 hash.
- This is a 40-character string composed of hexadecimal characters (0–9 and a–f) and calculated based on the contents of a file or directory structure in Git.
- Git stores everything in its database not by file name but by the hash value of its contents.
- A SHA-1 hash looks something like this:

```
24b9da6552252987aa493b52f8696cd6d3b00373
```

# Git Generally Only Adds Data

- When action is done in Git,it only *add data to the Git database. It is hard to get the* system to do anything that is not undoable or to make it erase data in any way.
- But after committing a snapshot into Git, it is very difficult to lose, especially if database is pushed to another repository.

# The Three States

- Git has three main states that your files can reside in: ***modified,staged, and committed:***
- Modified means that you have changed the file but have not committed it to your database yet.
- Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
- Committed means that the data is safely stored in your local database.

This leads us to the three main sections of a Git project: the working tree, the staging area, and the Git directory.

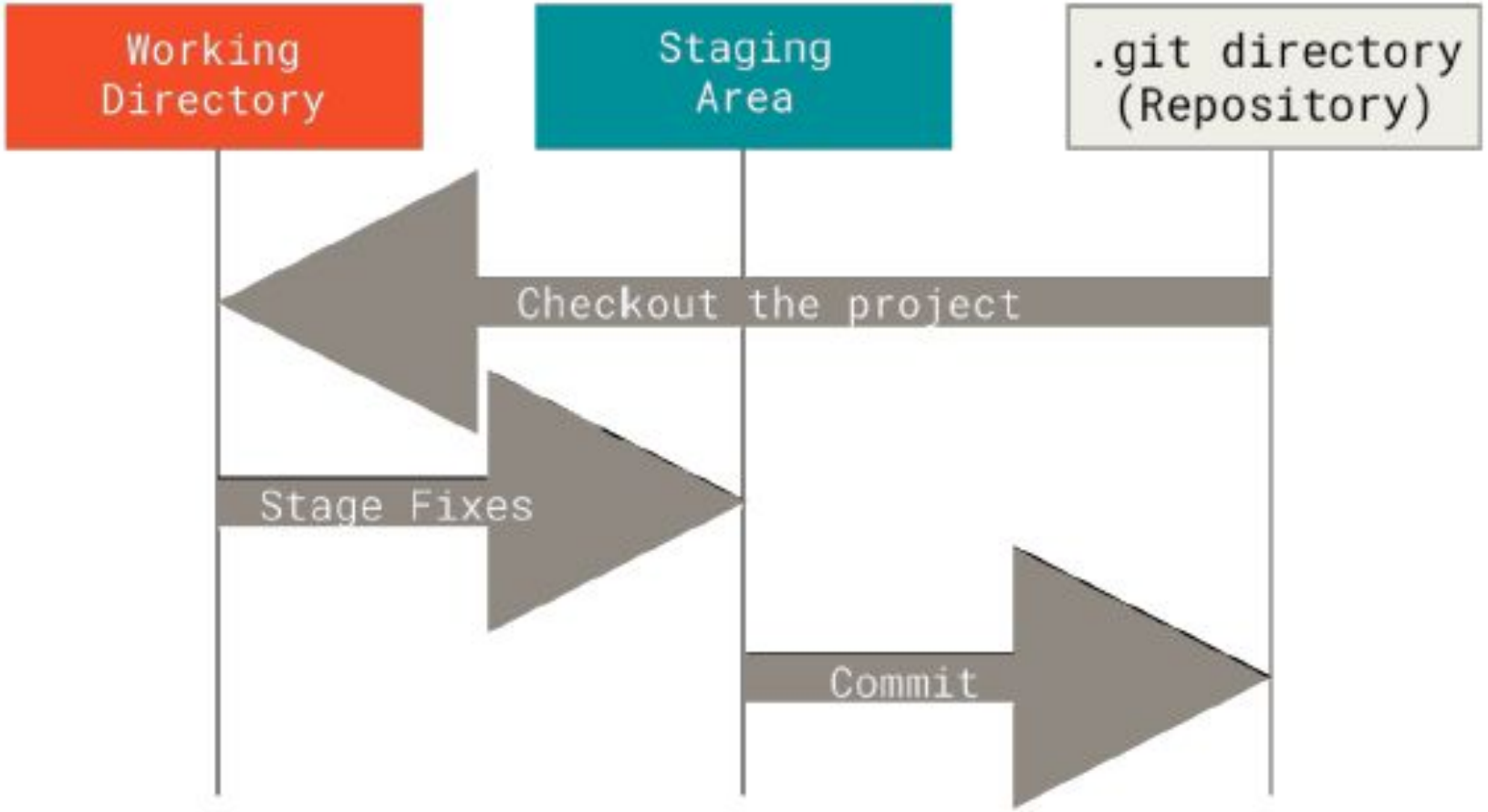



*Figure 6. Working tree, staging area, and Git directory*

- The working tree is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.
- The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit. Its technical name in Git parlance is the “index”, but the phrase “staging area” works just as well.

- The Git directory is where Git stores the metadata and object database for your project. This is the most important part of Git, and it is what is copied when you *clone a repository from another* computer.
- If a particular version of a file is in the Git directory, it's considered *committed.*
- If it has been modified and was added to the staging area, it is *staged.*
- *If it was changed since it was checked* out but has not been staged, it is *modified.*