MANAGING SOURCE CODE – GIT & GITHUB

AGENDA

- Introduction to CVS and GIT
- GIT File workflow
- Important GIT Commands
- Introduction to GitHub
- Using GIT and GitHub together.

INTRODUCTION TO CVS

- Source control systems provide the necessary "grip" to allow you to stay in control
- Source control systems manage changes to documents so that their state is consistent
- Also known as version control or revision control systems
- They can be used to store anything
- They work best for storing changing text documents
- They are usually associated with source code

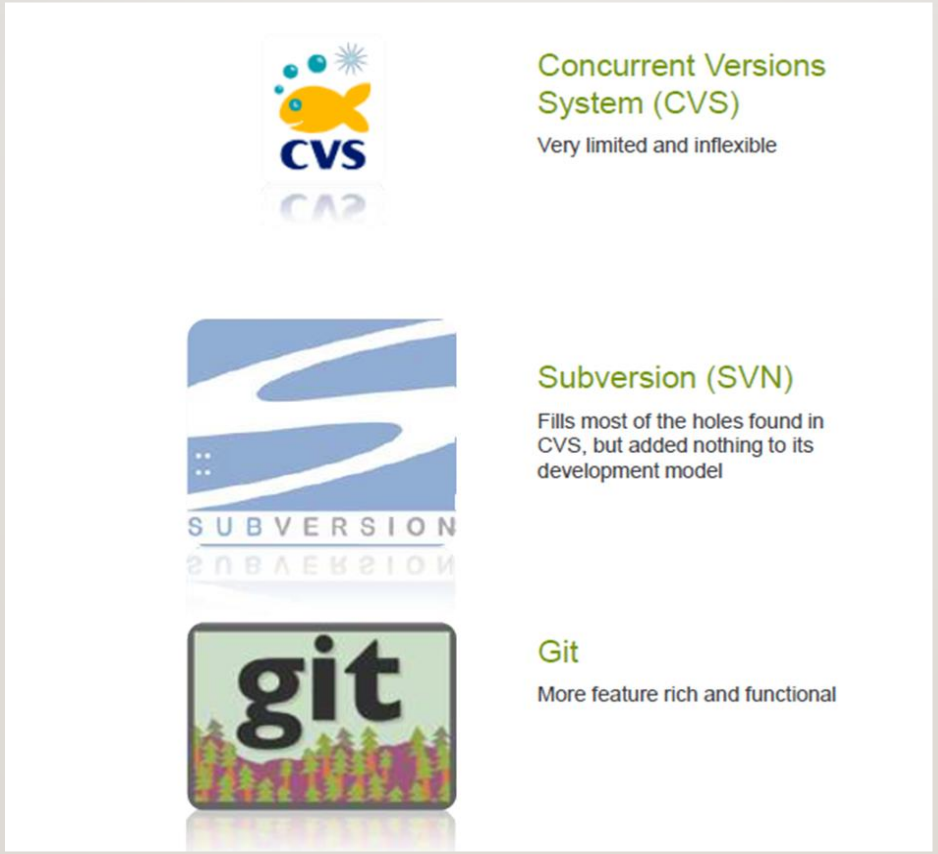
OVERVIEW OF VERSION CONTROL

- Version control software is an essential part of the every-day of the modern software team's professional practices.
- Version control software keeps track of every modification to the code in a special kind of database.
- Version control protects source code from both catastrophe and the casual degradation of human error and unintended consequences.
- Helps teams to solve problems like, tracking every individual change by each contributor and helping prevent concurrent work from conflict.
- Conflicts if any should be solved using help of tool or manually.

BENEFITS OF VERSION CONTROL

- No need to keep multiple backups.
- Allows multiple people to work on same file / project.
- A complete long-term change history of every file.
- Branching and merging – maintain code as per projects / release / functionality etc.
- Traceability - ability to trace each change made to the software and connect it to project management and bug tracking software.
- Easily switch / work on earlier file versions.

VARIOUS VERSION CONTROL SYSTEMS



Concurrent Versions System (CVS)

Very limited and inflexible

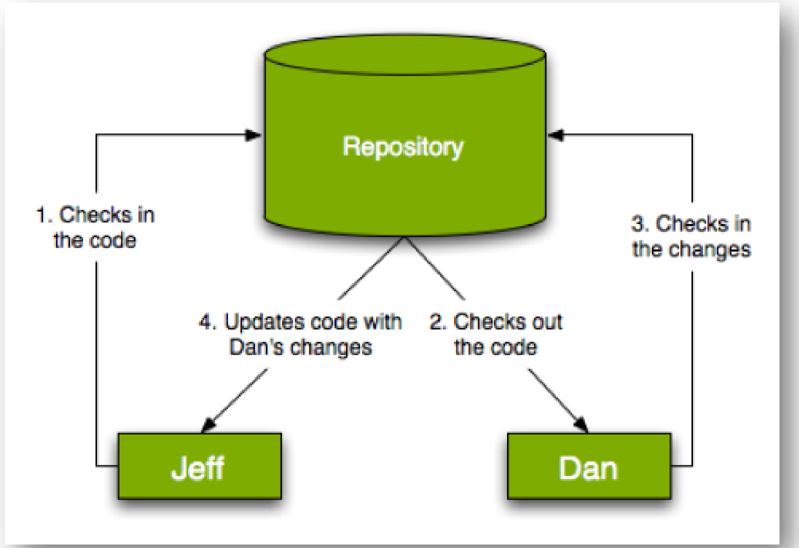
Subversion (SVN)

Fills most of the holes found in CVS, but added nothing to its development model

Git

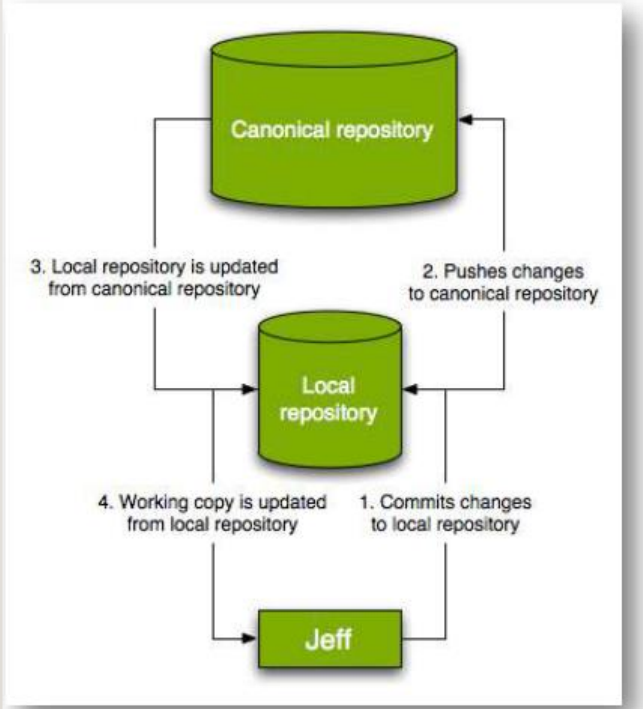
More feature rich and functional

CENTRAL VCS & DISTRIBUTED VCS



A basic, centralized version control system

Users *commit* changes to the central repository and a new version is born to be checked out by other users


Canonical repository
3. Local repository is updated from canonical repository
2. Pushes changes to canonical repository
Local repository
4. Working copy is updated from local repository
1. Commits changes to local repository
Jeff

A distributed version control system

Each user has a full local copy of the repository. Users *commit* changes and when they want to share it, the *push* it to the shared repository

OVERVIEW OF CENTRAL VCS

Workflow:

- Pull down any changes other people have made from the central server.
- Make your changes, and make sure they work properly.
- Commit your changes to the central server, so other programmers can see them.

e.g. : SVN, TFS, Perforce

BENEFITS AND DISADVANTAGES OF CENTRAL VCS

Benefits:

- Project history and database is maintained on centralized server, beneficial incase of large project size and history.

Disadvantages:

- Internet / network connection is must.
- Frequent (daily) database sync is necessary.
- Pull and push to centralized server can be time taking.

OVERVIEW OF DISTRIBUTED VCS

Workflow:

- Clone entire repository on local machine.
- Maintain all code changes on local machine.
- Incase of sharing the code with others, merge the code on central GIT repo.

BENEFITS AND DISADVANTAGES

Benefits:

- Extremely fast, as database resides on local drive.
- Committing to a changeset can be done locally. Group of changesets then can be pushed to remote repository.
- No need of internet connection.

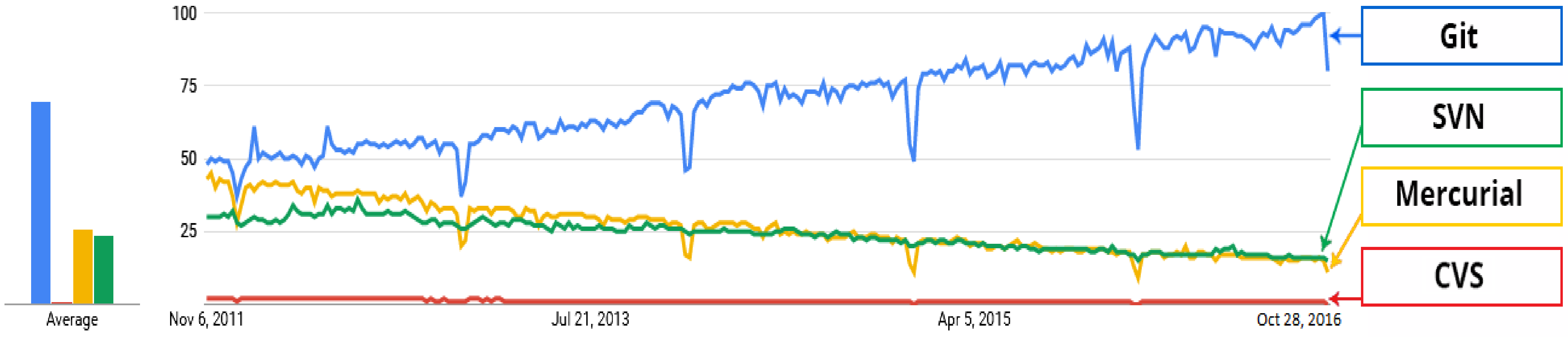
Disadvantages:

- Initial cloning and maintaining huge database can eat up space on local machine.
- Database backup should be done locally.

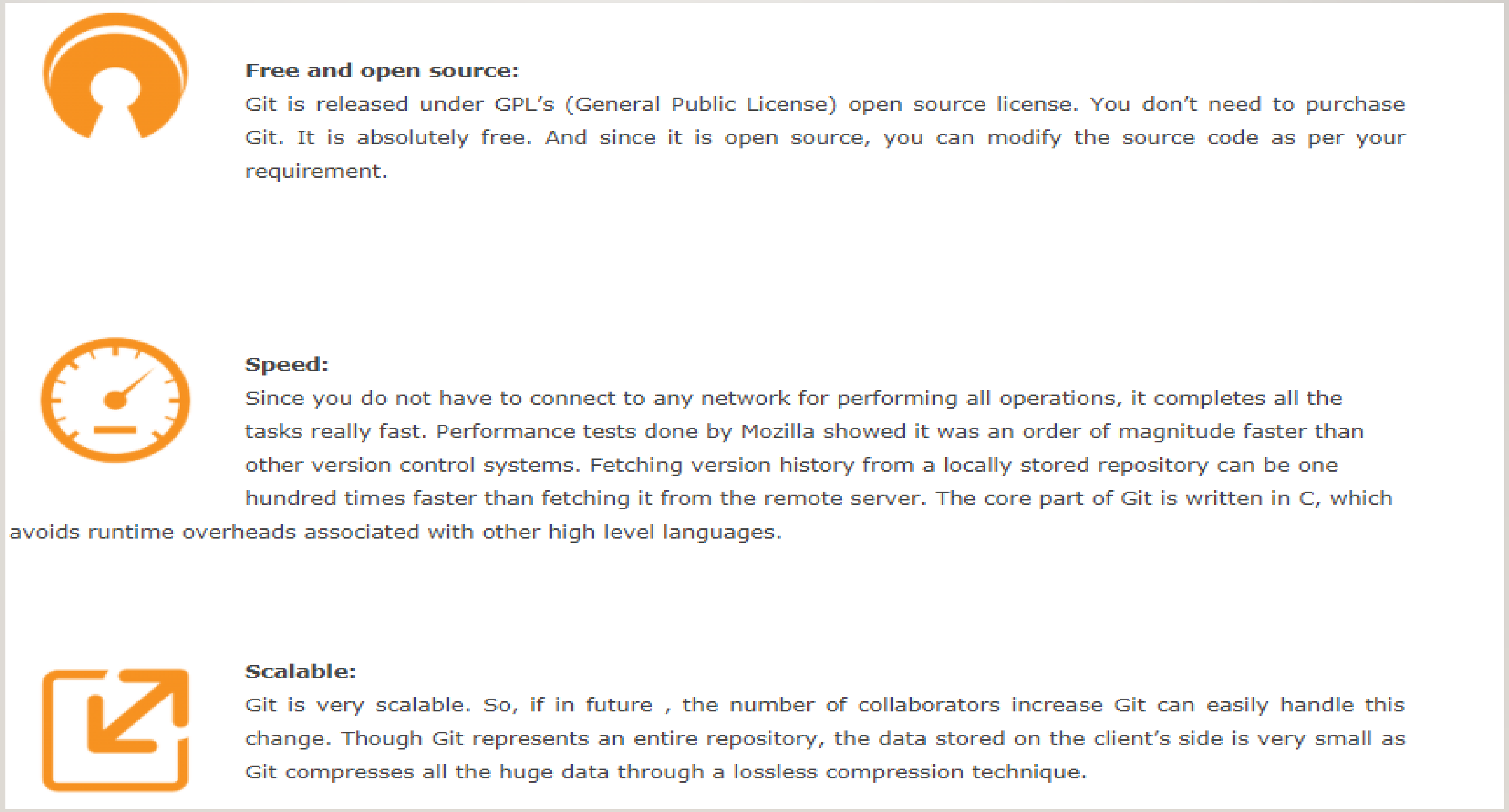
OVERVIEW OF GIT

- Git is a Distributed Version Control tool that supports distributed non-linear workflows by providing data assurance for developing quality software.
- Git provides with all the Distributed VCS facilities to the user that was mentioned earlier. Git repositories are very easy to find and access.

GIT IN THE MARKET



GIT FEATURES

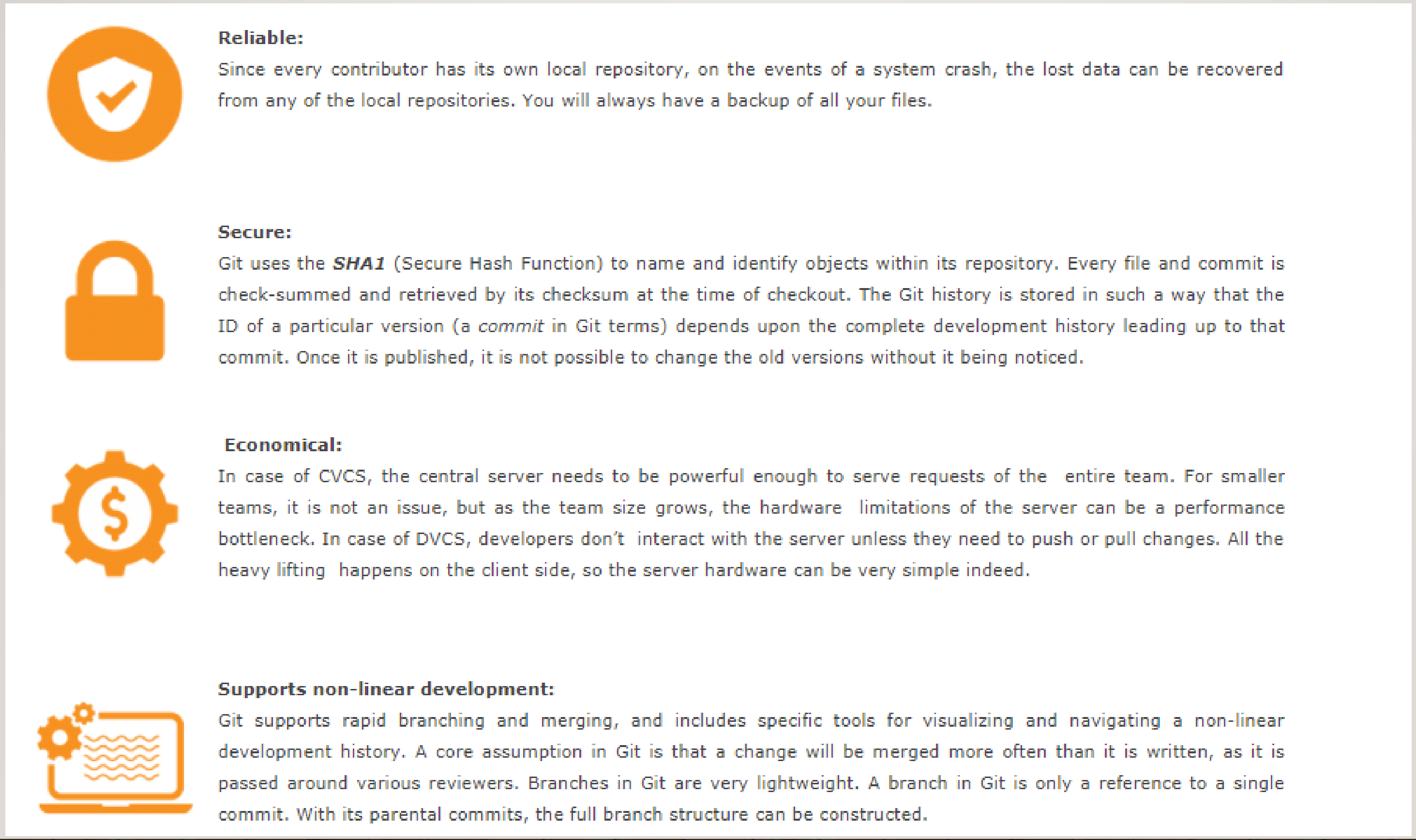


Free and open source:
Git is released under GPL's (General Public License) open source license. You don't need to purchase Git. It is absolutely free. And since it is open source, you can modify the source code as per your requirement.

Speed:
Since you do not have to connect to any network for performing all operations, it completes all the tasks really fast. Performance tests done by Mozilla showed it was an order of magnitude faster than other version control systems. Fetching version history from a locally stored repository can be one hundred times faster than fetching it from the remote server. The core part of Git is written in C, which avoids runtime overheads associated with other high level languages.

Scalable:
Git is very scalable. So, if in future , the number of collaborators increase Git can easily handle this change. Though Git represents an entire repository, the data stored on the client's side is very small as Git compresses all the huge data through a lossless compression technique.

GIT FEATURES ..CONTD



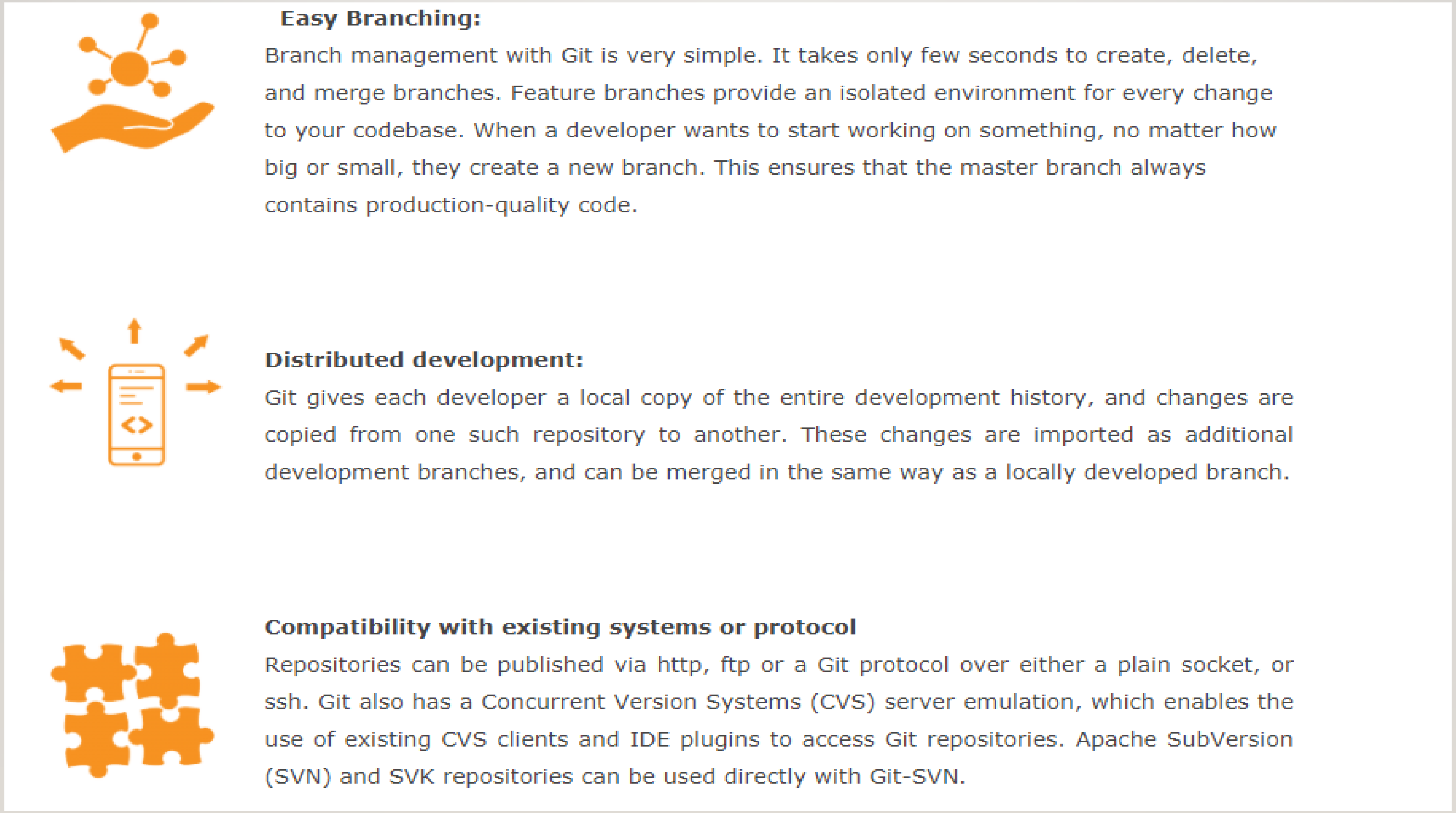
Reliable:
Since every contributor has its own local repository, on the events of a system crash, the lost data can be recovered from any of the local repositories. You will always have a backup of all your files.

Secure:
Git uses the ***SHA1*** (Secure Hash Function) to name and identify objects within its repository. Every file and commit is check-summed and retrieved by its checksum at the time of checkout. The Git history is stored in such a way that the ID of a particular version (a *commit* in Git terms) depends upon the complete development history leading up to that commit. Once it is published, it is not possible to change the old versions without it being noticed.

Economical:
In case of CVCS, the central server needs to be powerful enough to serve requests of the entire team. For smaller teams, it is not an issue, but as the team size grows, the hardware limitations of the server can be a performance bottleneck. In case of DVCS, developers don't interact with the server unless they need to push or pull changes. All the heavy lifting happens on the client side, so the server hardware can be very simple indeed.

Supports non-linear development:
Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history. A core assumption in Git is that a change will be merged more often than it is written, as it is passed around various reviewers. Branches in Git are very lightweight. A branch in Git is only a reference to a single commit. With its parental commits, the full branch structure can be constructed.

GIT FEATURES



Easy Branching:

Branch management with Git is very simple. It takes only few seconds to create, delete, and merge branches. Feature branches provide an isolated environment for every change to your codebase. When a developer wants to start working on something, no matter how big or small, they create a new branch. This ensures that the master branch always contains production-quality code.

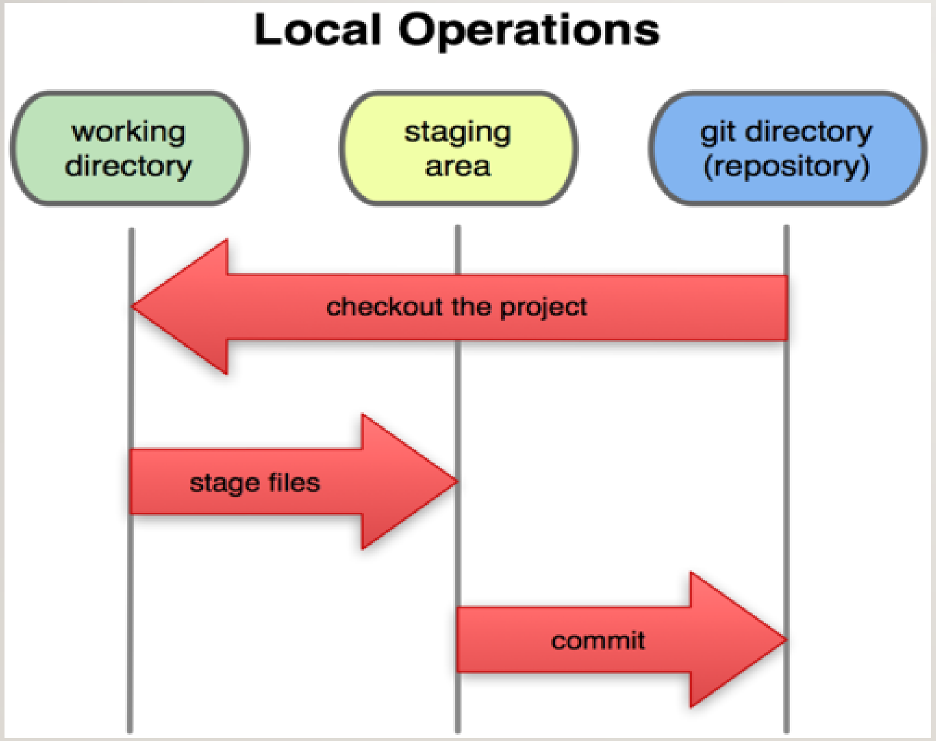
Distributed development:

Git gives each developer a local copy of the entire development history, and changes are copied from one such repository to another. These changes are imported as additional development branches, and can be merged in the same way as a locally developed branch.

Compatibility with existing systems or protocol

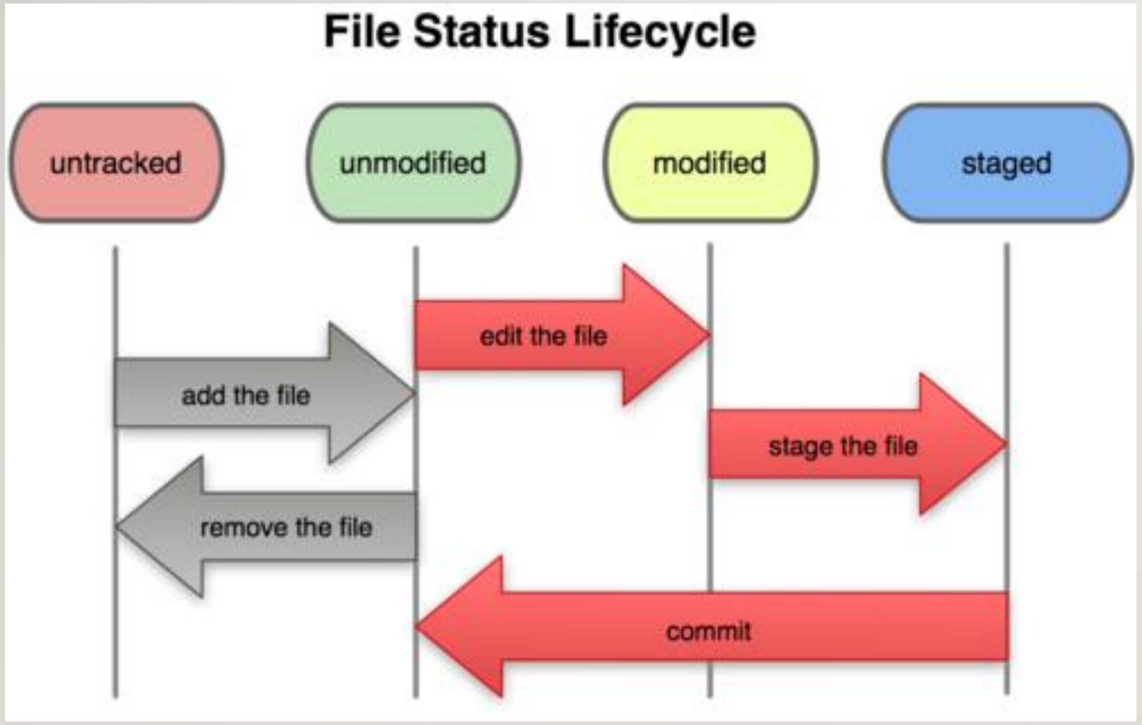
Repositories can be published via http, ftp or a Git protocol over either a plain socket, or ssh. Git also has a Concurrent Version Systems (CVS) server emulation, which enables the use of existing CVS clients and IDE plugins to access Git repositories. Apache SubVersion (SVN) and SVK repositories can be used directly with Git-SVN.

GIT PROJECTS

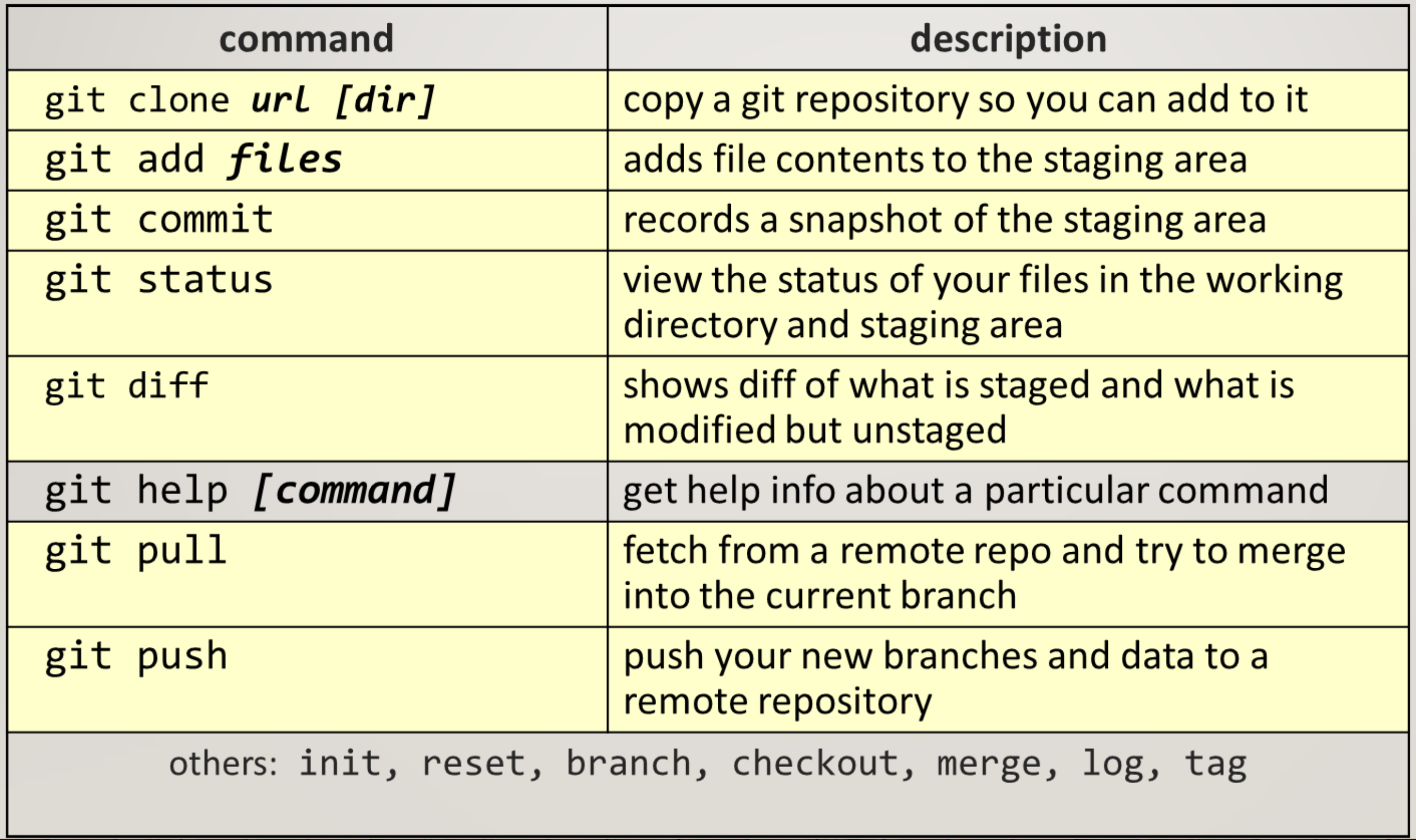


Unmodified/modified Files Staged Files Committed Files

GIT FILE LIFECYCLE



GIT COMMANDS



command	description
`git clone` ***`url [dir]`***	copy a git repository so you can add to it
`git add` ***`files`***	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help` ***`[command]`***	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
others: `init, reset, branch, checkout, merge, log, tag`	

BASIC WORKFLOW

Basic Git workflow:

1. **Modify** files in your working directory.
2. **Stage** files, adding snapshots of them to your staging area.
3. Do a **commit**, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

- Notes:
  - If a particular version of a file is in the **git directory**, it's considered **committed**.
  - If it's modified but has been 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**.

OVERVIEW OF GITHUB

- GitHub is a web-based hosting service for version control using git.
- It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.
- It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

WORKING WITH GIT

- Installing and configuring Git on Windows
- Installing and configuring Git on CentOS
- Executing commands in Git Cheat Sheet

WORKING WITH GITHUB

- Create a Repository
- Create a Branch
- Make and Commit Changes
- Open a Pull Request
- Merge your Pull Request

Thank You