



*Les outils de l'ingénieur*

*Dan MAGIER*

# Dan Magier

CEO



<https://www.linkedin.com/in/dan-magier-67a9374/>

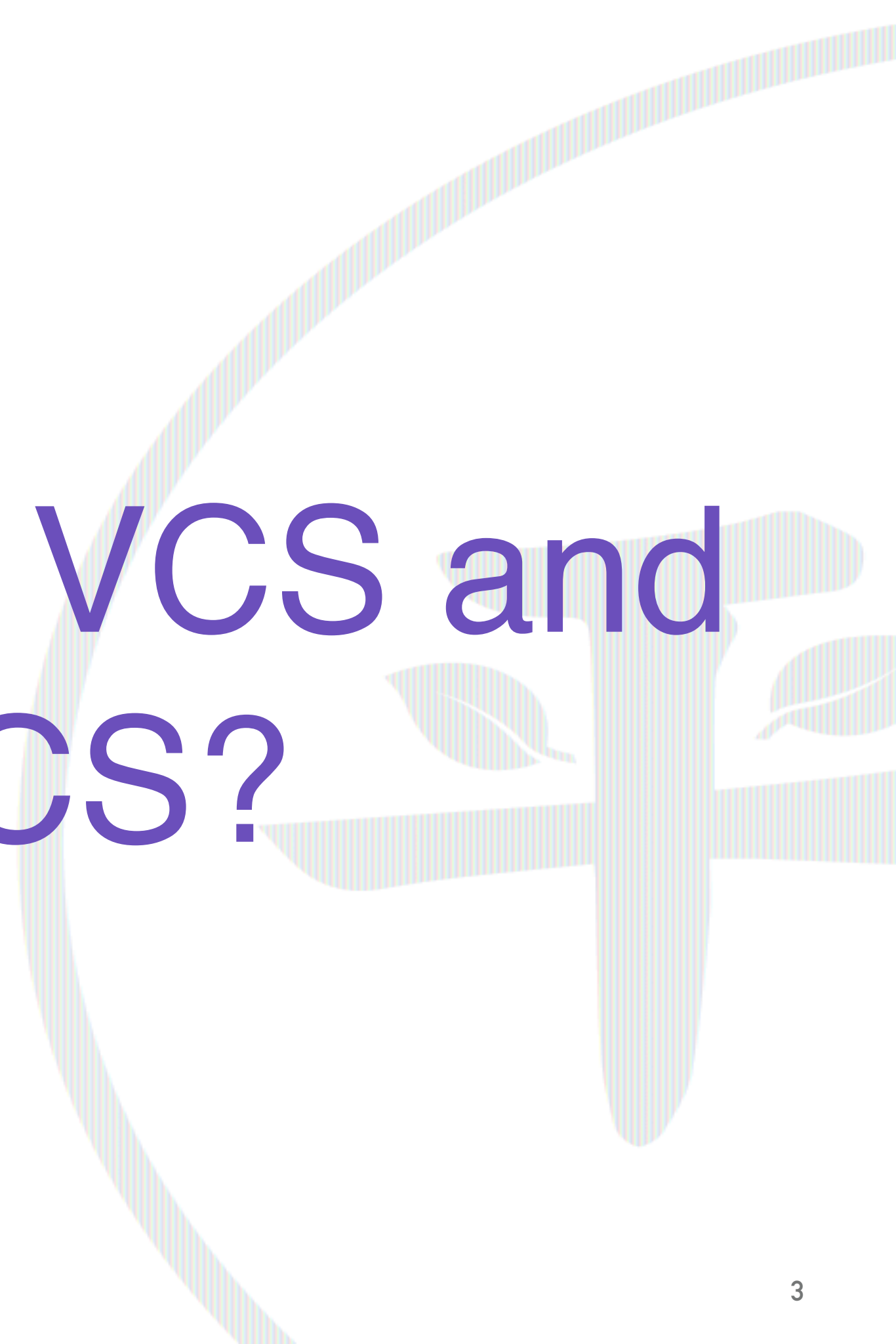


*@MagierDan*



*dan@heiwa-it.com*

# WHAT IS VCS and DVCS?



# VERSION CONTROL SOURCE (VCS)

---

A component of software configuration management, **version control**, also known as **revision control** or **source control**,<sup>[1]</sup> is the management of changes to documents, computer programs, large web sites, and other collections of information.

Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply « revision".  
For example, an initial set of files is "revision 1 ».

When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.

*[https://en.wikipedia.org/wiki/Version\\_control](https://en.wikipedia.org/wiki/Version_control)*

# DECENTRALIZED VERSION CONTROL SOURCE (DVCS)

---

In software development, **distributed version control** is a form of version control where the complete codebase - including its full history - is mirrored on every developer's computer.

This allows branching and merging to be managed automatically, increases speeds of most operations (except for pushing and pulling), improves the ability to work offline, and does not rely on a single location for backups.

*[https://en.wikipedia.org/wiki/Distributed\\_version\\_control](https://en.wikipedia.org/wiki/Distributed_version_control)*

# DVCS ADVANTAGES OVER VCS

---

The act of cloning an entire repository gives distributed version control tools several advantages over centralized systems:

- Performing actions other than pushing and pulling changesets is *extremely* fast because the tool only needs to access the hard drive, not a remote server.
- Committing new changesets can be done locally without anyone else seeing them. Once you have a group of changesets ready, you can push all of them at once.
- Everything but pushing and pulling can be done without an internet connection. So you can work on a plane, and you won't be forced to commit several bugfixes as one big changeset.
- Since each programmer has a full copy of the project repository, they can share changes with one or two other people at a time if they want to get some feedback before showing the changes to everyone.

<https://www.atlassian.com/blog/software-teams/version-control-centralized-dvcs>

# DVCS DISADVANTAGES OVER VCS

---

There are almost no disadvantages to using a distributed version control system over a centralized one. Distributed systems do *not* prevent you from having a single “central” repository, they just provide more options on top of that.

There are only two major inherent disadvantages to using a distributed system:

- If your project contains many large, binary files that cannot be easily compressed, the space needed to store all versions of these files can accumulate quickly.
- If your project has a very long history (50,000 changesets or more), downloading the entire history can take an impractical amount of time and disk space.

<https://www.atlassian.com/blog/software-teams/version-control-centralized-dvcs>

# Git



# WHAT IS GIT?



- .....
1. A completely ignorant, childish person with no manners.
  2. A person who feels justified in their callow behaviour.
  3. A pubescent kid who thinks it's totally cool to act like a moron on the internet, only because no one can actually reach through the screen and punch their lights out.
- « That n00b is behaving like a bloody git. »

*<https://www.urbandictionary.com/define.php?term=Git>*

# WHAT IS GIT?

---

1. A completely ignorant, childish person with no manners.
  2. A person who feels justified in their callow behaviour.
  3. A pubescent kid who thinks it's totally cool to act like a moron on the internet, only because no one can actually reach through the screen and punch their lights out.
- « That n00b is behaving like a bloody git. »

*<https://www.urbandictionary.com/define.php?term=Git>*

# WHAT IS GIT?

---

**Git** is a distributed version-control system for tracking changes in source code during software development. **Git** is a **distributed version-control** system for tracking changes in **source code** during **software development**.

It is designed for coordinating work among **programmers**, but it can be used to track changes in any set of **files**.

Its goals include speed, **data integrity**, and support for distributed, non-linear workflows. development.

It is designed for coordinating work among programmers, but it can be used to track changes in any set of files.

Its goals include speed, data integrity, and support for distributed, non-linear workflows.

*<https://en.wikipedia.org/wiki/Git>*

# Git Principals Objects



# REPOSITORY

---

The purpose of Git is to manage a project, or a set of files, as they change over time. Git stores this information in a data structure called a repository.

A git **repository** contains, among other things, the following:

- A set of **commit objects**.
- A set of references to commit objects, called **heads**.

The Git repository is stored in the same directory as the project itself, in a subdirectory called `.git`. Note differences from central-repository systems like CVS or Subversion:

- There is only one `.git` directory, in the root directory of the project.
- The repository is stored in files alongside the project. There is no central server repository.

*<https://www.sbf5.com/~cduan/technical/git/git-1.shtml>*

# COMMIT

---

A **commit object** contains three things:

- A set of **files**, reflecting the state of a project at a given point in time.
- References to **parent commit objects**.
- An **SHA1 name**, a 40-character string that uniquely identifies the commit object. The name is composed of a hash of relevant aspects of the commit, so identical commits will always have the same name.

The parent commit objects are those commits that were edited to produce the subsequent state of the project. Generally a commit object will have one parent commit, because one generally takes a project in a given state, makes a few changes, and saves the new state of the project. The section below on merges explains how a commit object could have two or more parents.

*<https://www.sbf5.com/~cduan/technical/git/git-1.shtml>*

# COMMIT

---

A project always has one commit object with no parents. This is the first commit made to the project repository.

Based on the above, you can visualize a repository as a directed acyclic graph of commit objects, with pointers to parent commits always pointing backwards in time, ultimately to the first commit. Starting from any commit, you can walk along the tree by parent commits to see the history of changes that led to that commit.

The idea behind Git is that version control is all about manipulating this graph of commits. Whenever you want to perform some operation to query or manipulate the repository, you should be thinking, “how do I want to query or manipulate the graph of commits?”

*<https://www.sbf5.com/~cduan/technical/git/git-1.shtml>*

# BRANCHES

---

A **branch** is simply a reference to a commit object. Each branch has a name. By default, there is a branch in every repository called *master*.

A repository can contain any number of branches. At any given time, one branch is selected as the “current branch.” This head is aliased to *HEAD*, always in capitals.

<https://www.atlassian.com/blog/software-teams/version-control-centralized-dvcs>



# EXERCICES

---

Enough talk, let's practice ;-)

<http://gitimmersion.com/>