

# הנדסת תוכנה

## + 6-7-8. בקרת תצורה

### git / github

[Pragmatic Programmer Tip](#) :

**Always Use Source Code Control**

Source code control is a time machine for your work—you can go back.

# השבוע

- בקרת תצורה \ קוד – Revision Control
  - כלים: git / github
  - משימה אישית 3
  - שיטות: מודל ענפים
- סקר ZFR
- פרויקט 6: תחילת סבבים – סבב 1 MVP
- תרגיל – סקר ZFR



# בקרת גרסאות קוד - מקורות

- Sink, Version Control by Example
- Google Tech Talk: [Linus Torvalds on git](#)
- Spolsky, Hg Init: a Mercurial tutorial  
<http://hginit.com>
- Sink, Source Control HOWTO  
[http://www.ericssink.com/scm/source\\_control.html](http://www.ericssink.com/scm/source_control.html)
- Intro to Distributed Version Control  
<http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/>

# VCS Links

- Eric Raymond on [vcs](#)
- [The 10 commandments of good source control management](#), blog 2011
- FogBugz and Kiln [video](#), 2011
  - DVCS University Slides ([\\*](#))
- [Spolsky, Hg Init: a Mercurial tutorial](#)
- Azad, [A Visual Guide to Version Control](#)
  - [Intro to Distributed Version Control \(Illustrated\)](#)
- [Eric Sink, Source Control HOWTO](#)

# Git Links (see wiki links)

- <http://git-scm.com/> ([getting started](#))
- Set Up Git (Win), [ssh issues](#)  
<http://help.github.com/win-set-up-git/>
- Info: <http://git-scm.com>, [gitref.org](http://gitref.org),  
progit.org (book, [basic](#)), [Git In The Trenches](#)
- [Windows client list](#), O'Reilly Webcast:  
[Git in One Hour](#)
- [git internals](#) video

# More Git / Github Links

- [Git branching with git-flow](#), Heb. Video, 2013
- Videos: [Git For Ages 4 And Up](#), [Git Going](#)(oredev'12), [Think like a Git](#)
- [learn.github.com/](#), [try.github.com/](#)  
[Gitflow](#)
- saastv: [Using Branches with Git](#)
- [no branches at flickr](#)
- [Insider Guide to GitHub](#) (video)
- Articles: [article](#) (including .gitignore), [post](#), [difficulties](#),



# איפה אנחנו בפרויקט (בקורס)?

- למה?  
בעיה (פלט: הצעת פרויקט\חזון\SOW)
- מי?  
צוות (Inception, אתחול\תכנון פרויקט)
- מה?  
דרישות (SRS)
- איך?  
תיכון (ארכיטקטורה) (SDS)
- מתי?  
תכנון וניהול – (ZFR)
- **בניה**  
**(סבבי פיתוח)**



# בקרת תצורה (SCM)

- לפרויקט תוכנה תוצרים שונים:
- מסמכי דרישות ותיכון, קוד, executables, מדריכי שימוש, בדיקות, ...
- פרויקט תוכנה משתמש בכלים שונים:
- מהדרים, עורכים, צד ג', שת"פ, (מ"ה), ...



# בקרת תצורה

- מבחינת תהליך זה אלו נקראים  
CI – Configuration Items
- לכל אחד יכולים להיות גרסאות ועותקים שונים
- אנו צריכים יכולת לזהות, לעקוב ולאחסן אותם
- נתמקד בנושא של גרסאות

# בקרת גרסאות – Version Control

- איך (האם?) אתם שומרים את תוצרי העבודה שלכם?
- האם אפשר לשפר?
- האם יש הבדל בין מפתח בודד לחברה גדולה?
- שמות שונים:
  - בקרת תצורה
  - Software Configuration Management
  - Source/Revision/Version Control System

# Joel Test (~2000 / stackoverflow)

## 1. Do you use source control?

...היום...

"You've just spent twenty minutes doing a presentation for your teammates on **adopting source control**. Yeah, they don't do source control at all. Yep, **not at all—it's as if the last 20 years of computer science never happened**. But better late than never, and frankly any source control is better than none, because disaster is one errant delete away."

- "Driving Technical Change: Why People on Your Team Don't Act on Good Ideas, and How to Convince Them They Should", chap. [The Cynic](#)

# בקרת גרסאות – בשביל מה? יעדים



- איסוף כל הגרסאות ומעקב אחרי שינויים
  - חזרה לגרסה מסוימת, השוואה
  - מאפשר מחיקת קוד
- ניהול מספר גרסאות במקביל
- גיבוי והצלה
- שיתוף מספר מפתחים (מרוחקים) בו זמנית
  - טיפול בסתירות
- מאגר מעודכן של תוצרי הפרויקט
  - במיוחד עם Continuous Integration\daily build

בפרויקט תדרשו להדגים את בקרת  
התצורה שלכם

# פעולות נדרשות

- בקרת שינויים
  - זיהוי ותיעוד (למשל מי משנה, הסיבה, זמן וכדו')
  - ניתוח והערכה (של שינוי)
  - אישור \ דחיה
  - אימות, מימש ושחרור
- בקרת גרסאות
  - מאגר
  - הכנסה והוצאה
  - ענפים ומיזוגים
  - תיוג



# כלים: היסטוריה (השוואה)

Generation	Networking	Operations	Concurrency	Examples
1	None	One file at a time	Locks	RCS, SCCS
2	Centralized	Multi-file	Merge before commit	CVS, SourceSafe, Subversion, Team Foundation Server, IBM Rational ClearCase
3	Distributed	Changesets	Commit before merge	Bazaar, Git, Mercurial

# 40 Years of Version Control



SCCS & RCS (1970s)



CVS (1986)



Subversion (2001)



Git (2005)

Image © TheSun.au

# Git

- Successful open source projects
  - <https://git.wiki.kernel.org/index.php/GitProjects>
  - <https://github.com/google>, <https://github.com/microsoft>, facebook, twitter
  - <http://stackoverflow.com/research/developer-survey-2015#tech-sourcecontrol>

- Problems / Issues:

- Usability!
- Mainly a scripted / toolset (by now IDE integration and GUIs)
- Binary/big file (by now, e.g., git-lfs)
- Enterprise (e.g. locking)

Required for any path

Git - Version Control

SSH

HTTP/HTTPS and APIs

Basic Terminal Usage

Learn to Research

Data structures & Algorithms

Character Encodings

Github

Create your profile. Explore the relevant opensource projects. Make your habit to look under the hood for the projects you like. Create and contribute to opensource projects.



# משל גיט

- Tom Preston-Werner  
<http://tom.preston-werner.com/2009/05/19/the-git-parable.html>
- Herland,  
<http://www.infoq.com/presentations/git-details>, slides:  
[https://github.com/jherland/git\\_pparable](https://github.com/jherland/git_pparable)

# The Git Parable

Johan Herland

[johan@herland.net](mailto:johan@herland.net)

# The Git Parable

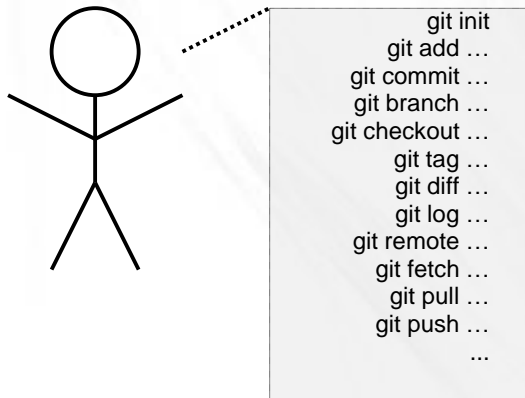
- Shamelessly stolen from Tom Preston-Werner  
<http://tom.preston-werner.com/2009/05/19/the-git-parable.html>
- I'm lazy...
- Also: Best introduction to Git I've found so far

# The Git Parable

- Git - simple & powerful

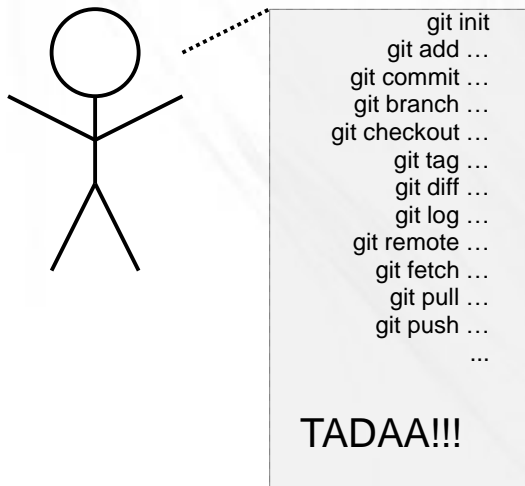
# The Git Parable

- Git - simple & powerful



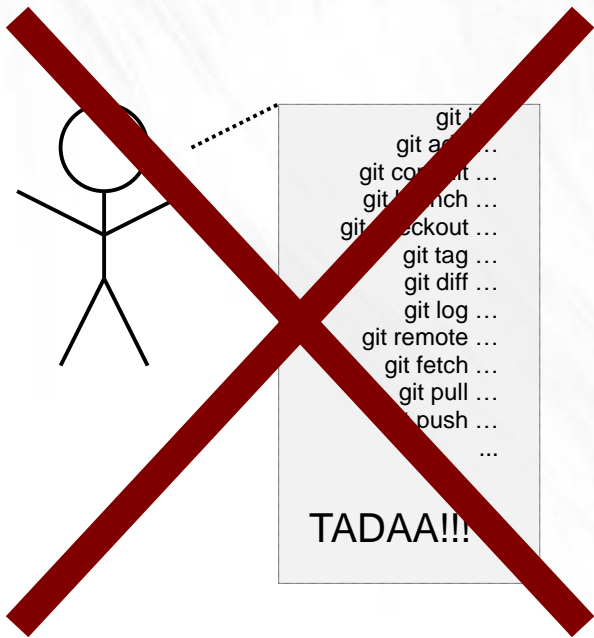
# The Git Parable

- Git - simple & powerful



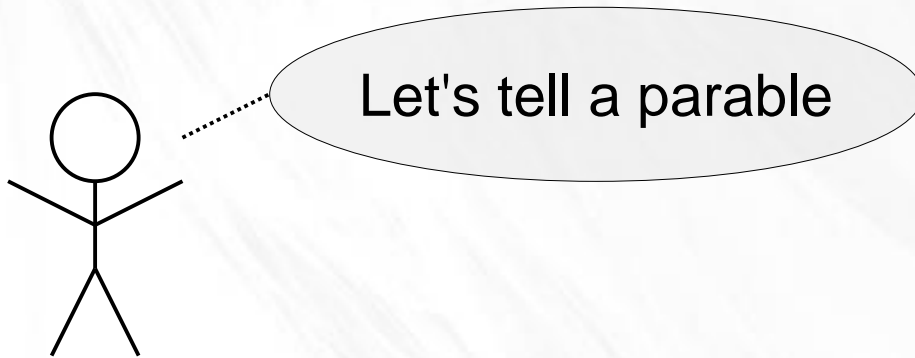
# The Git Parable

- Git - simple & powerful



# The Git Parable

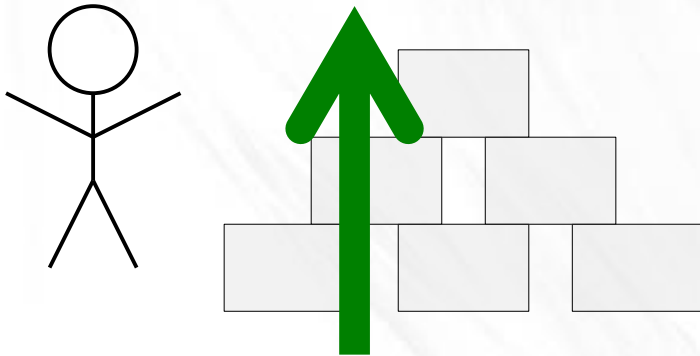
- Git - simple & powerful





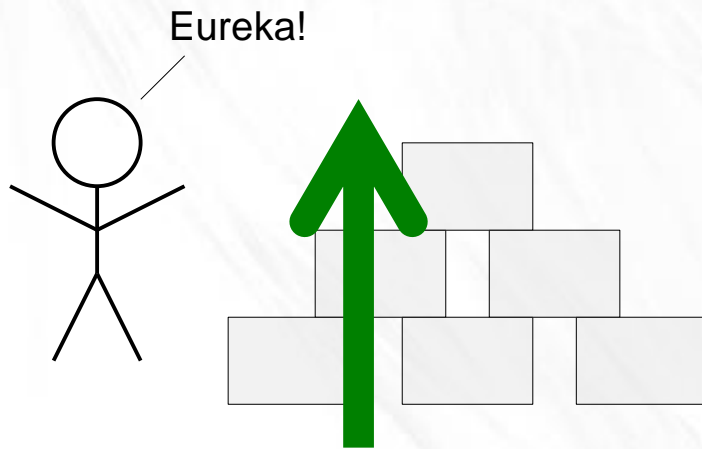
# The Git Parable

- Git - simple & powerful



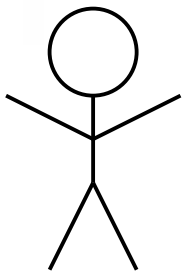
# The Git Parable

- Git - simple & powerful



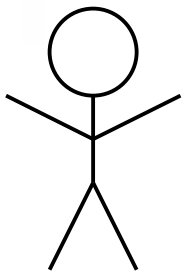
# The Parable

- A simple computer
  - A text editor
  - A few filesystem commands



# The Parable

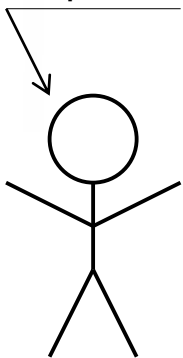
- Write a large software program



# The Parable

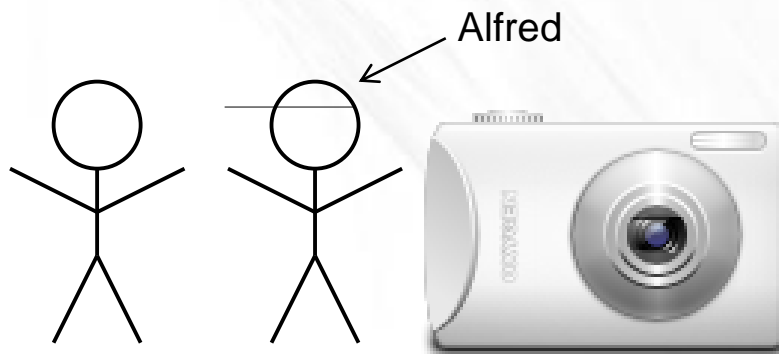
- Write a large software program
- Invent some method to keep track of versions
  - retrieve code that you changed/deleted

Responsible!



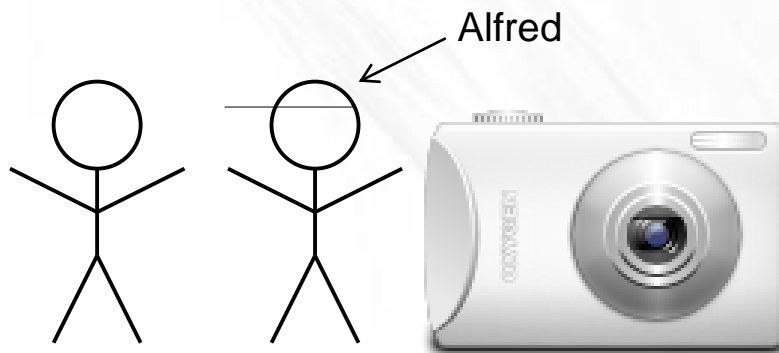
# Snapshots

- Alfred, the photographer



# Snapshots

- Alfred, the photographer



# Snapshots

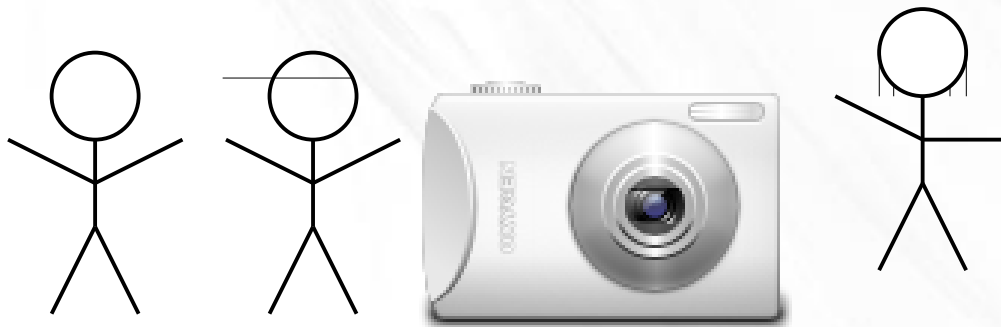
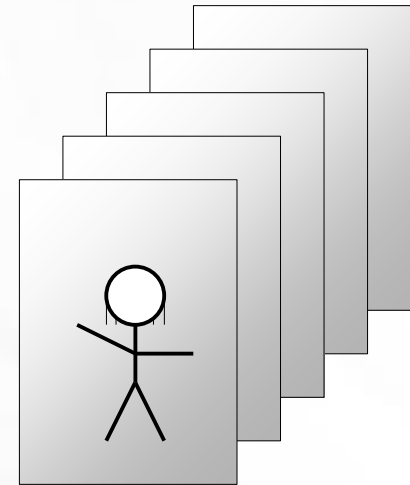
- Alfred, the photographer
- Hazel and her daughter





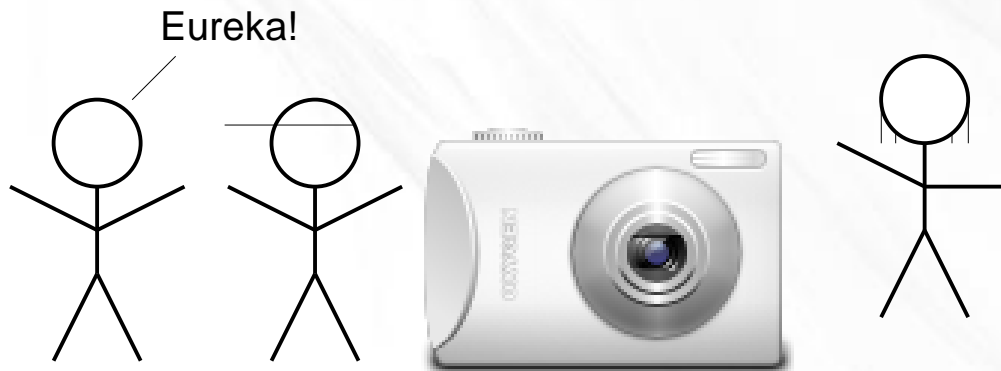
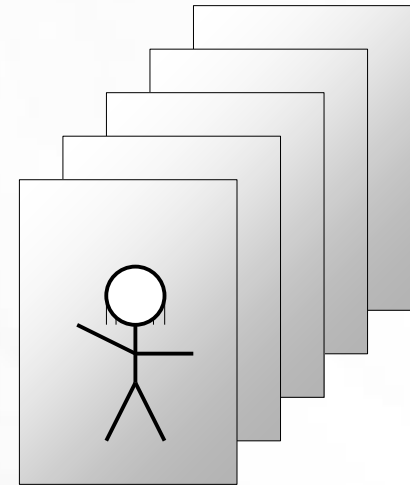
# Snapshots

- Alfred, the photographer
- Hazel and her daughter
  - Remember what the daughter was like at each different stage



# Snapshots

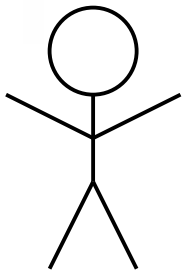
- Alfred, the photographer
- Hazel and her daughter
  - Remember what the daughter was like at each different stage



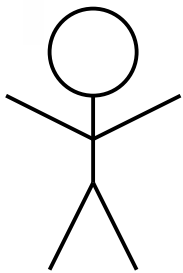
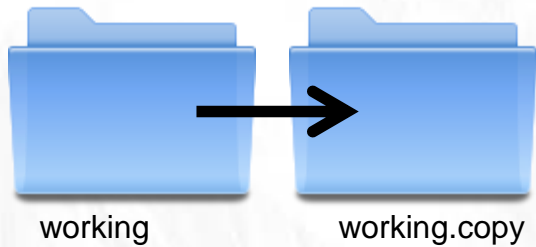
# Snapshots



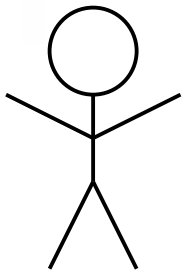
working



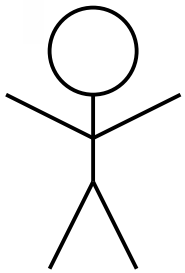
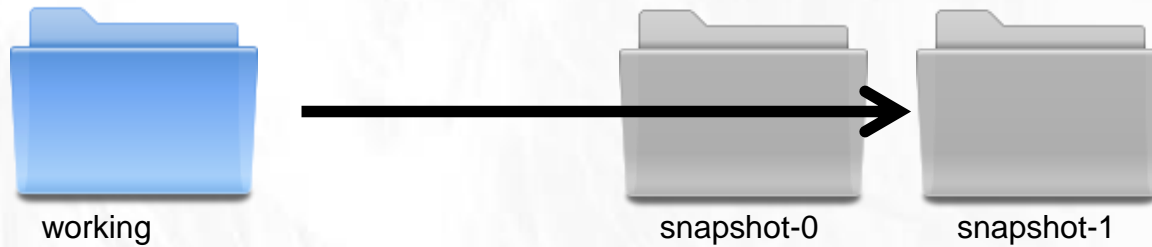
# Snapshots



# Snapshots



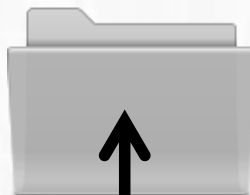
# Snapshots



# Snapshots



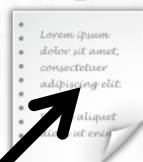
working



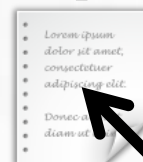
snapshot-0



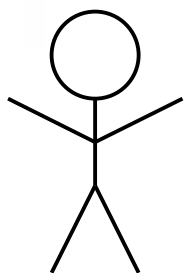
snapshot-1



message



message



2009-05-20 12:34:56

Initial version

2009-05-21 23:45:01

Introduced a new foo,  
and reset the bar to  
xyzy.

# Branches



working



snapshot-0

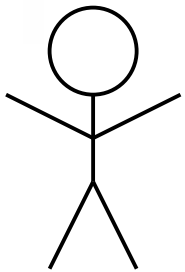


snapshot-1

...



snapshot-99





# Branches



working



snapshot-0



snapshot-1

...



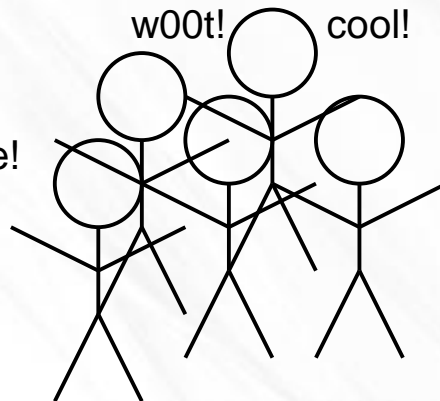
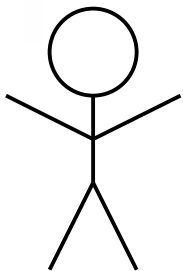
snapshot-99



w00t!

cool!

nice!



# Branches



working



snapshot-0



snapshot-1

...

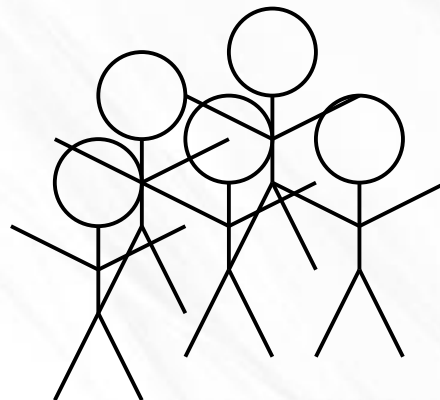
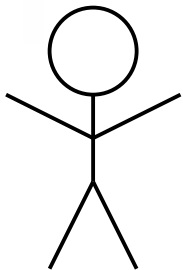


snapshot-99

...



snapshot-109



# Branches



working



snapshot-0



snapshot-1

...



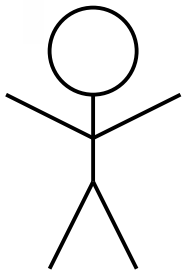
snapshot-99

...

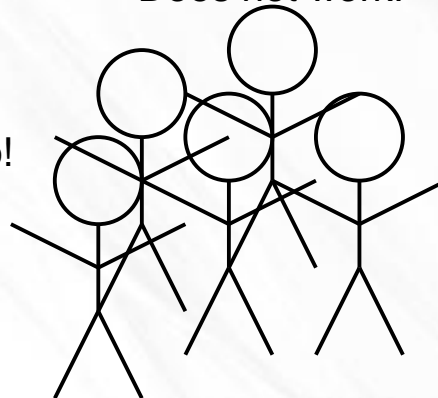


snapshot-109

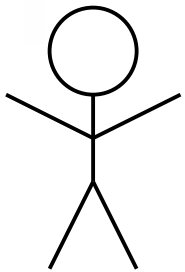
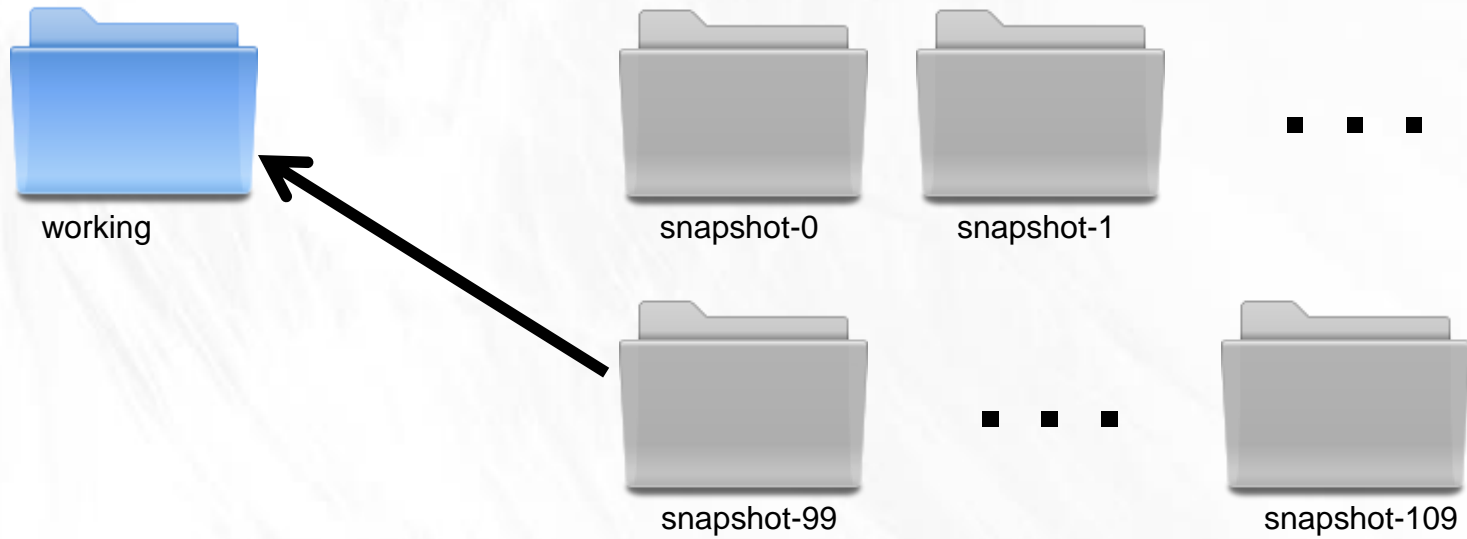
Does not work!



Boo!



# Branches



# Branches



working



snapshot-0



snapshot-1

...



snapshot-99

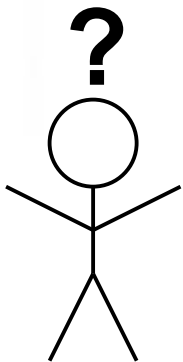
...



snapshot-109



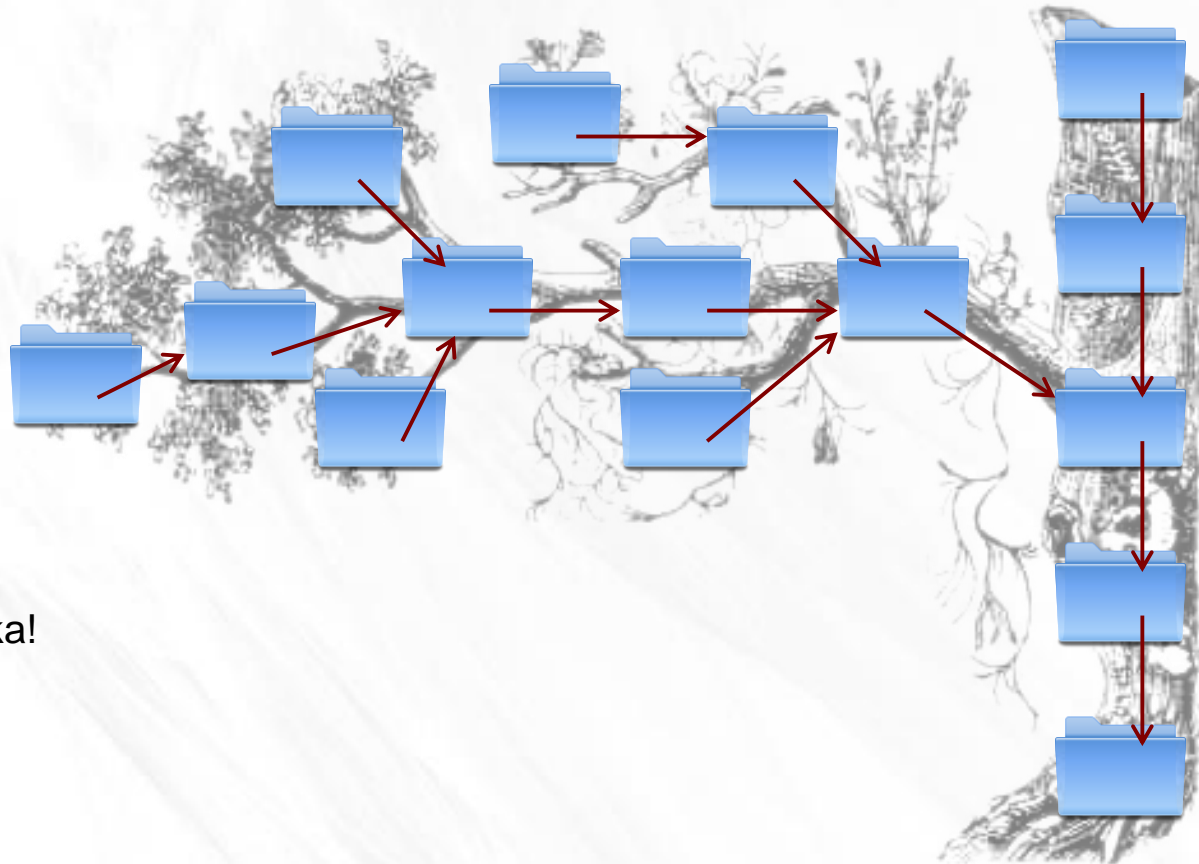
snapshot-110



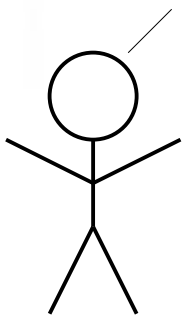
# Branches



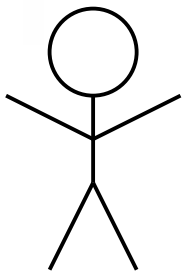
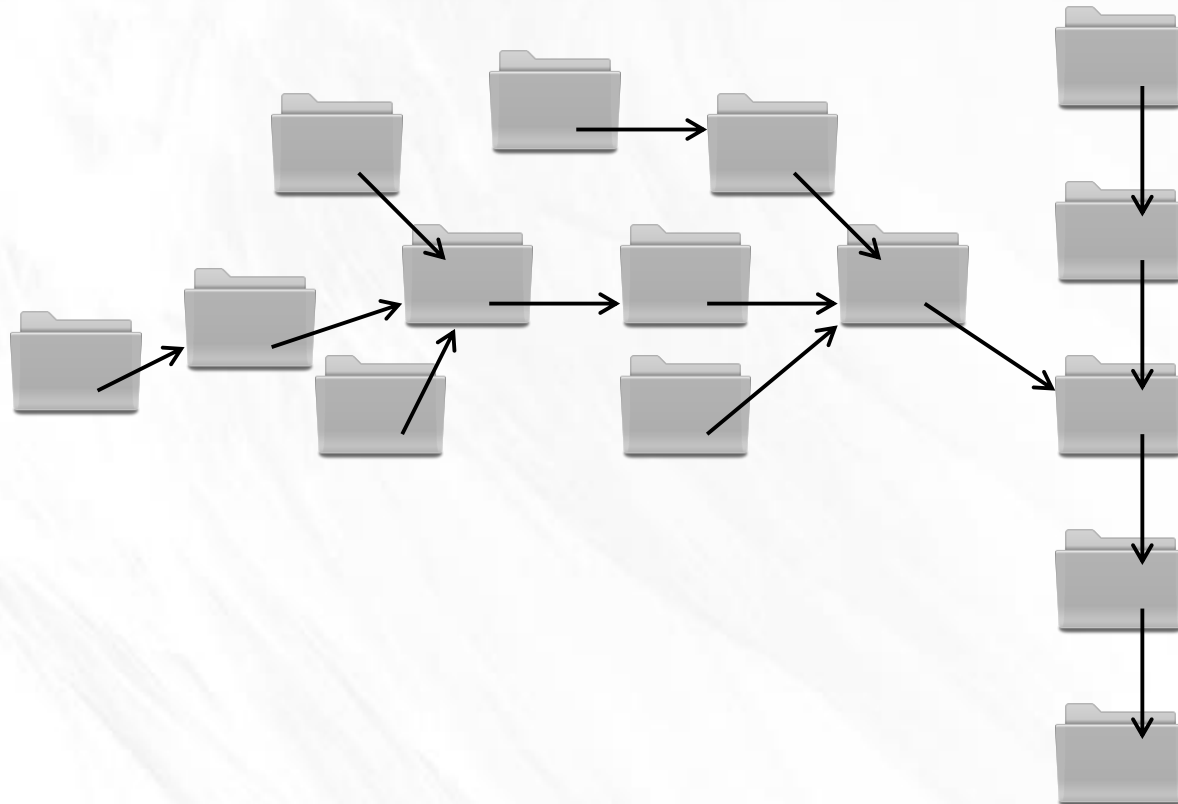
# Branches



Eureka!

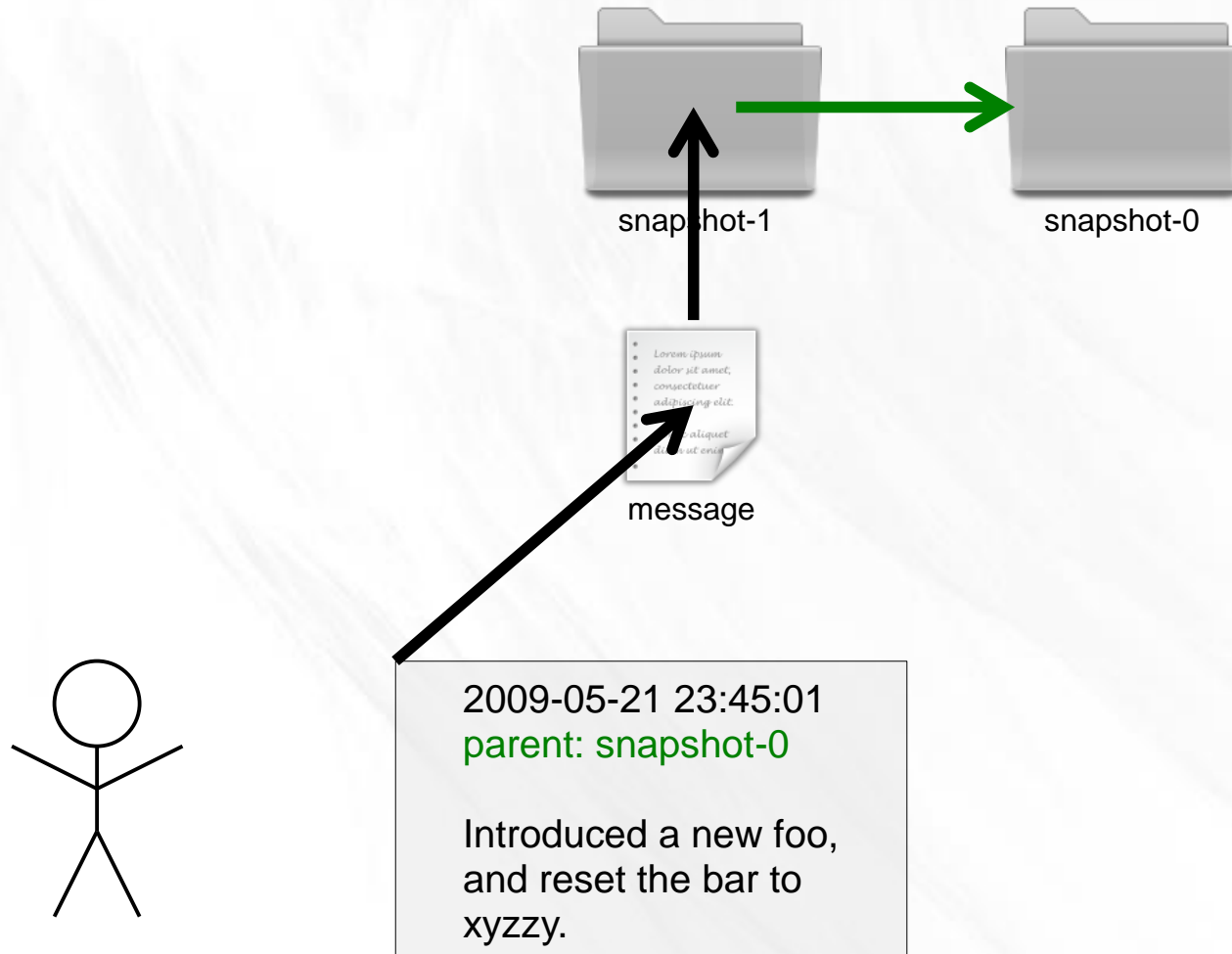


# Branches

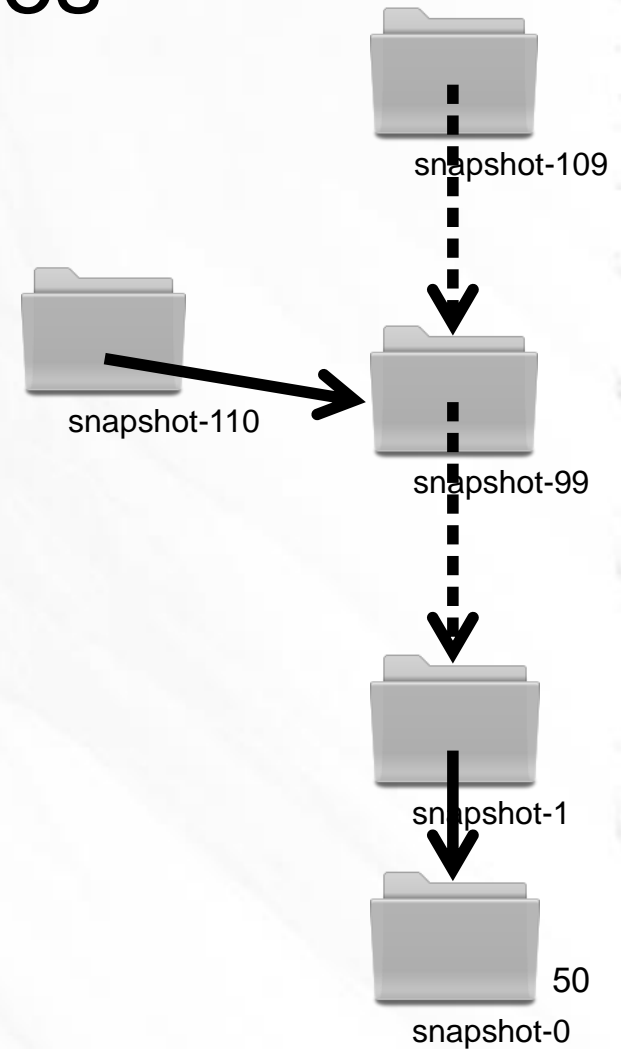
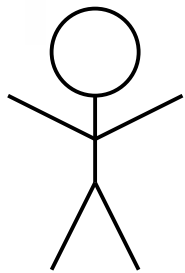




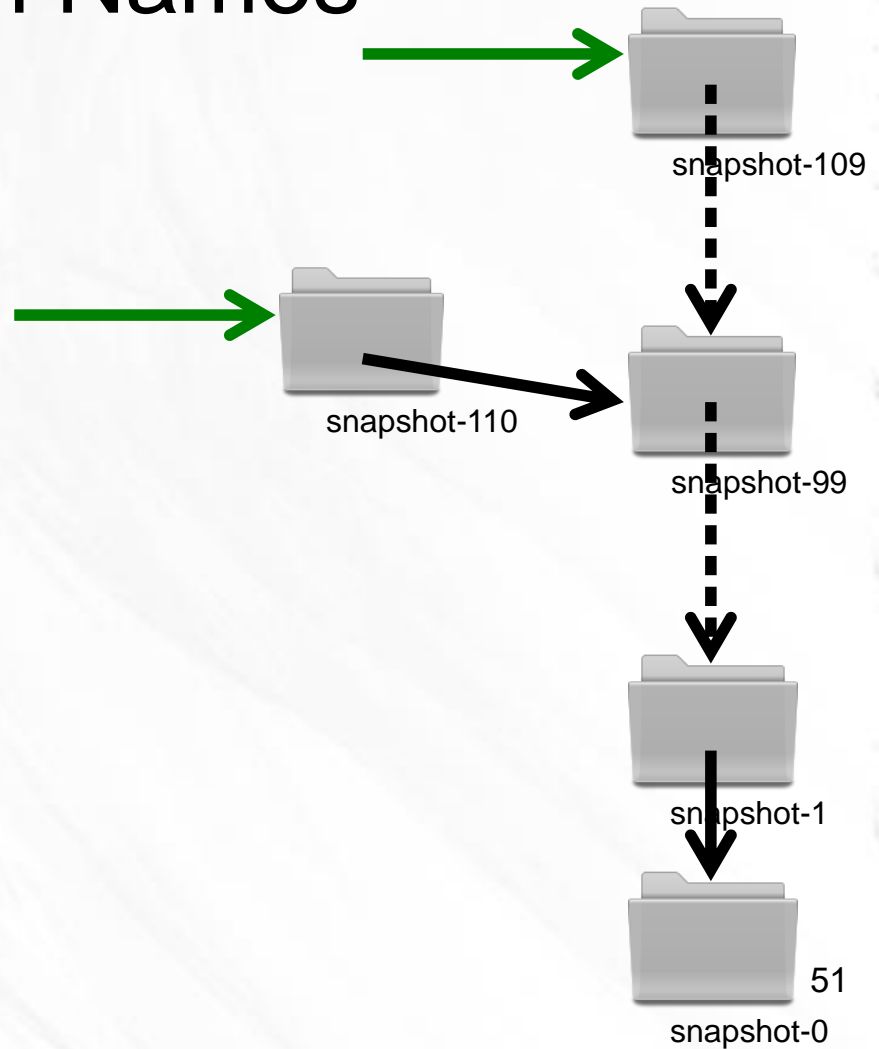
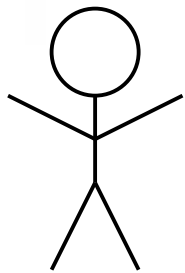
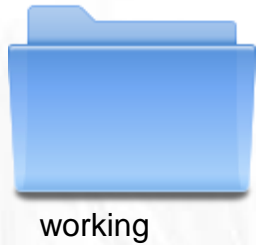
# Branches



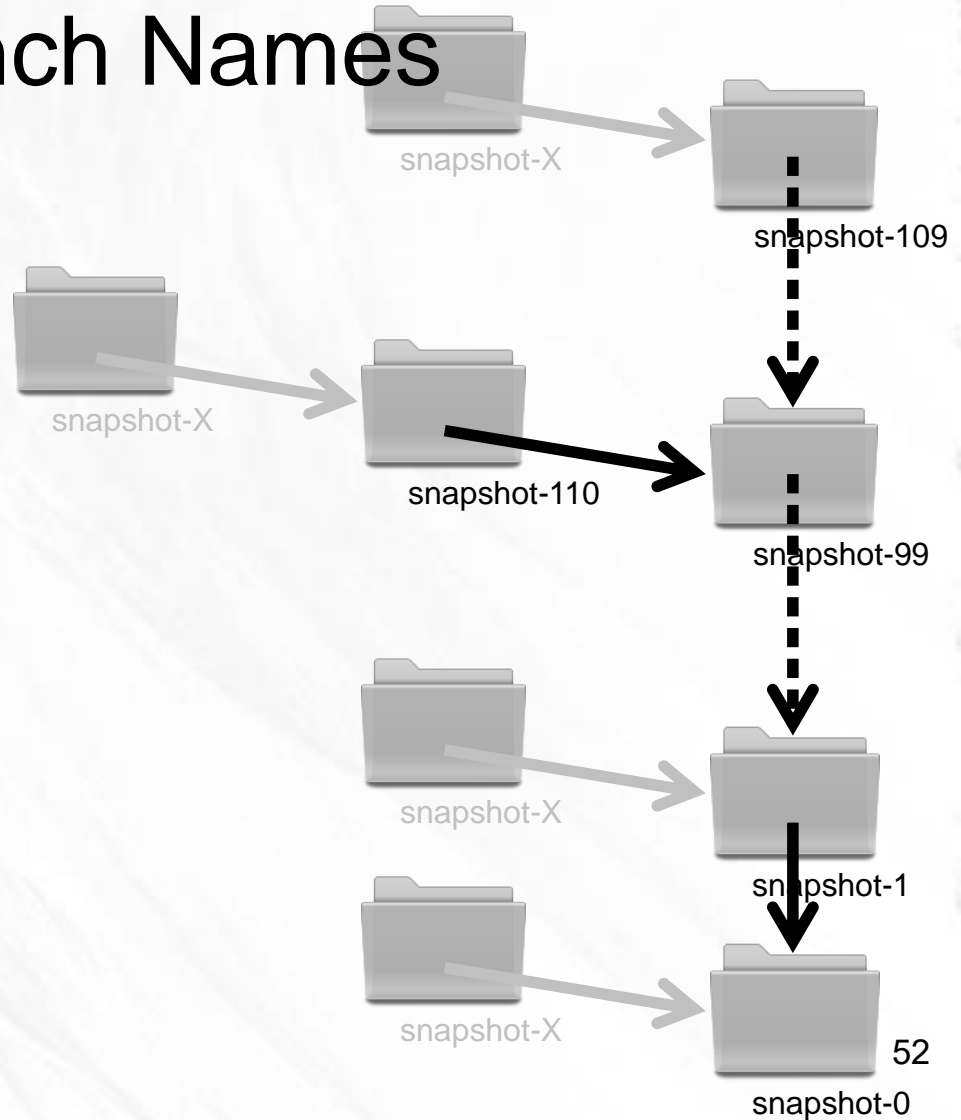
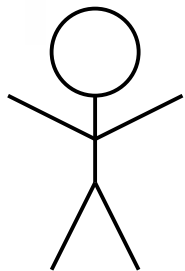
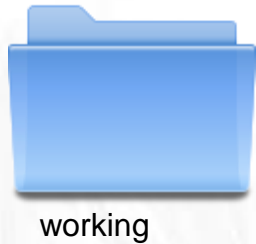
# Branch Names



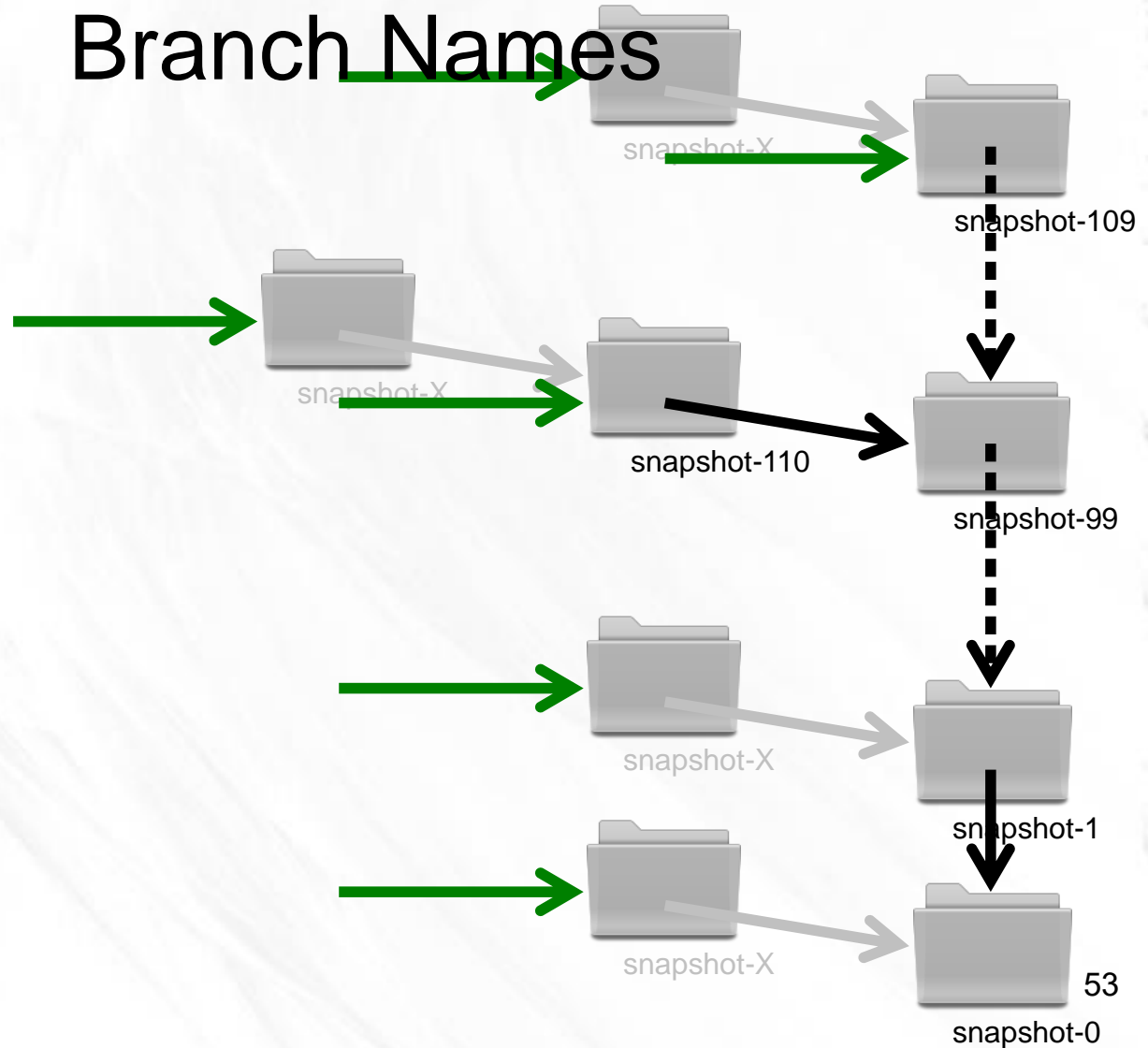
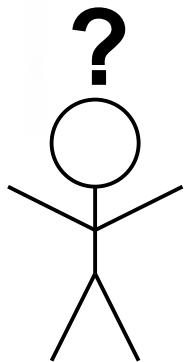
# Branch Names



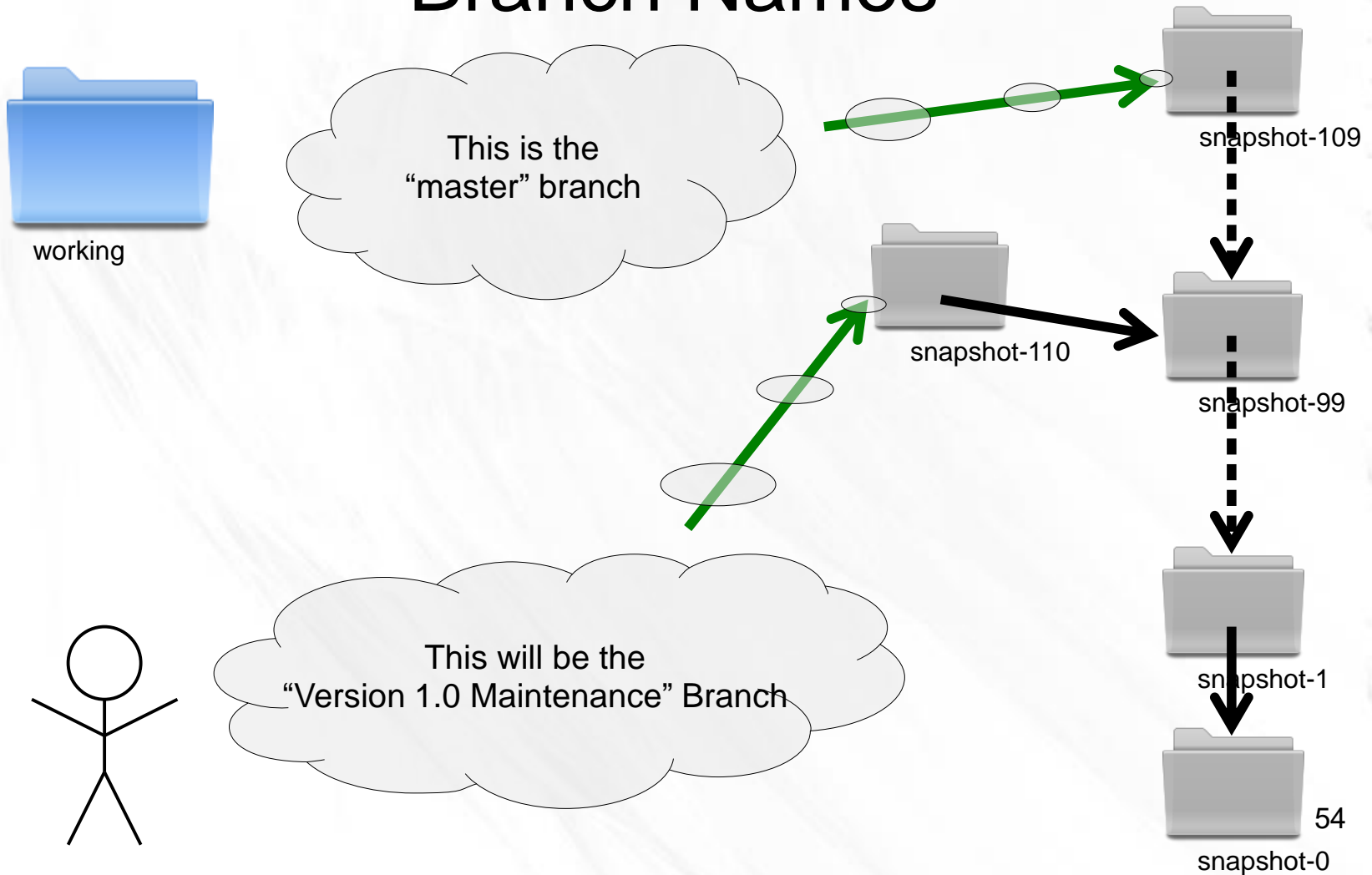
# Branch Names



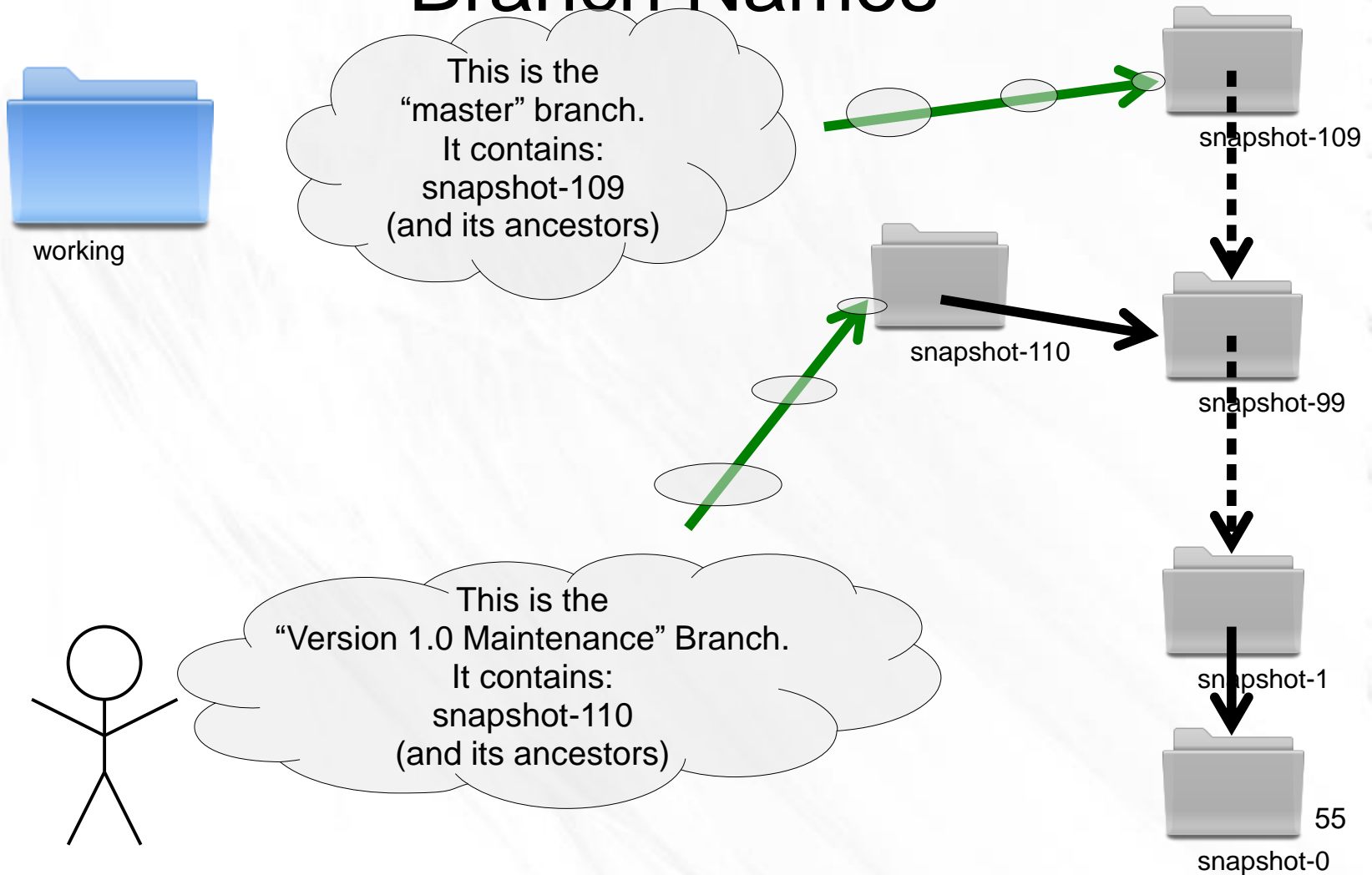
# Branch Names



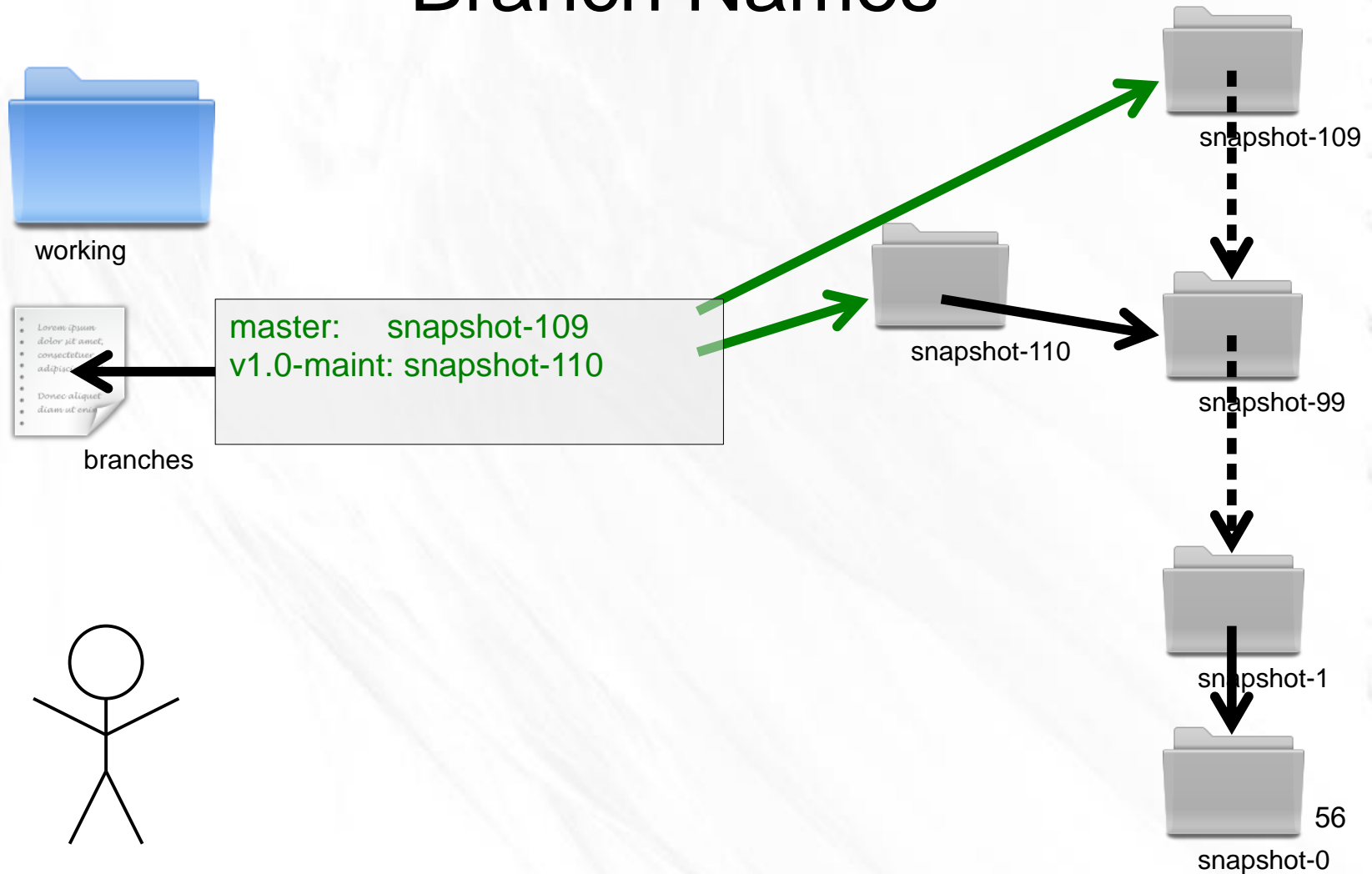
# Branch Names



# Branch Names

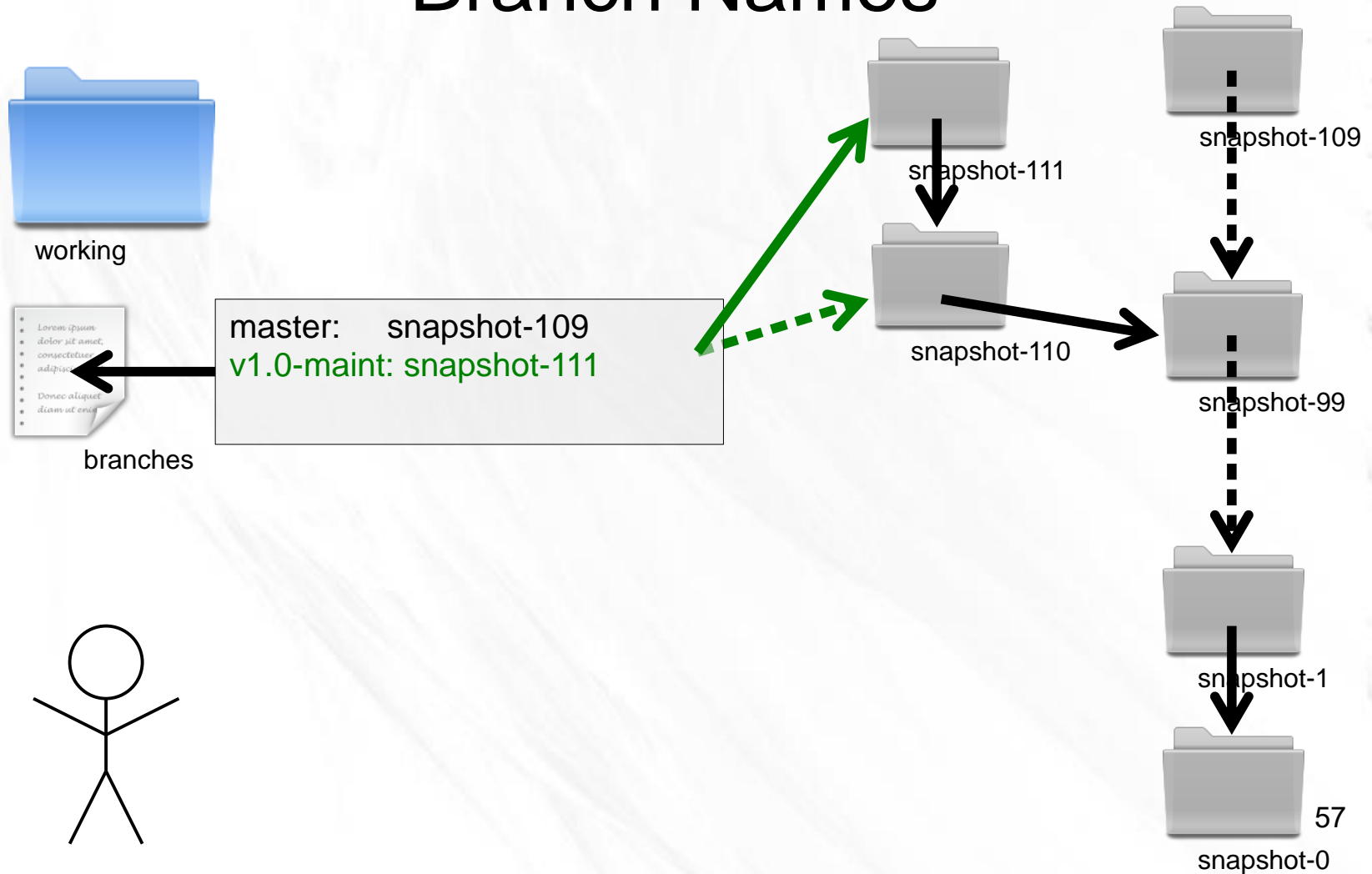


# Branch Names

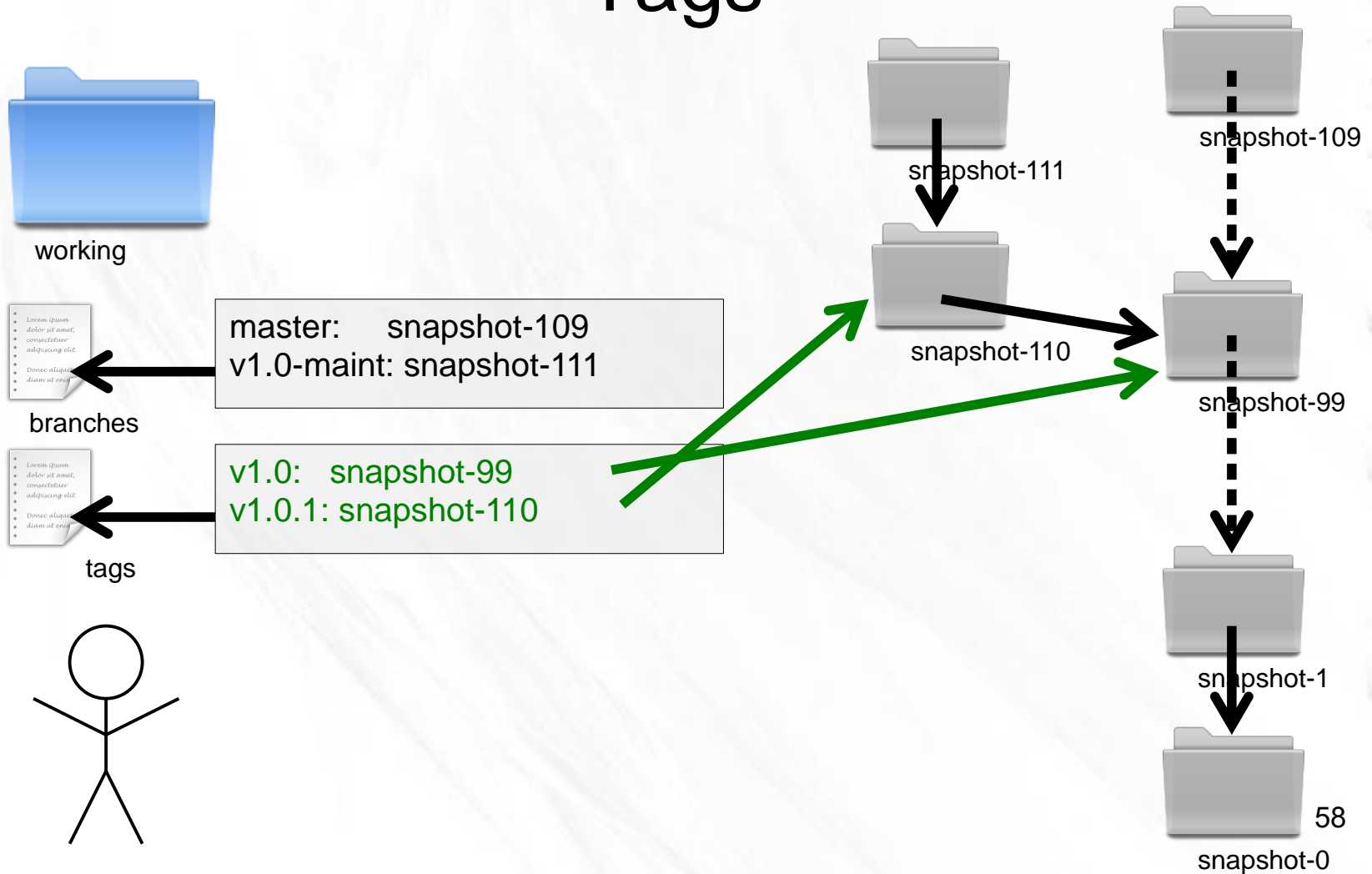




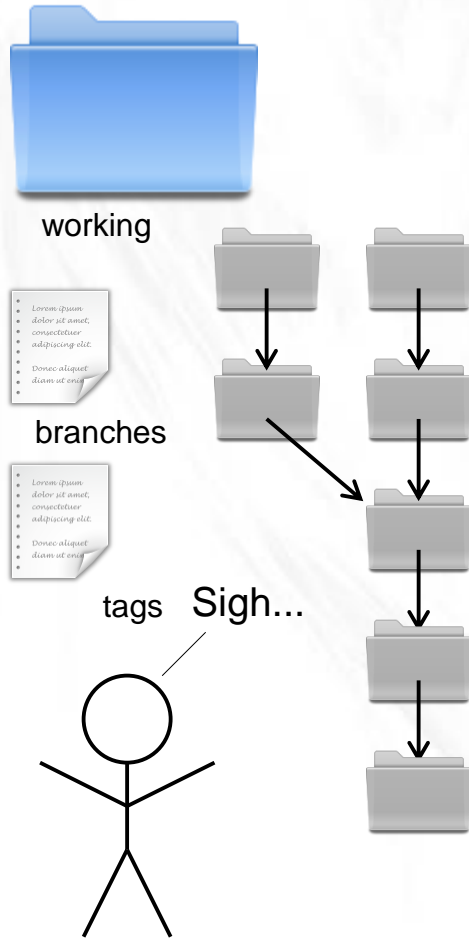
# Branch Names



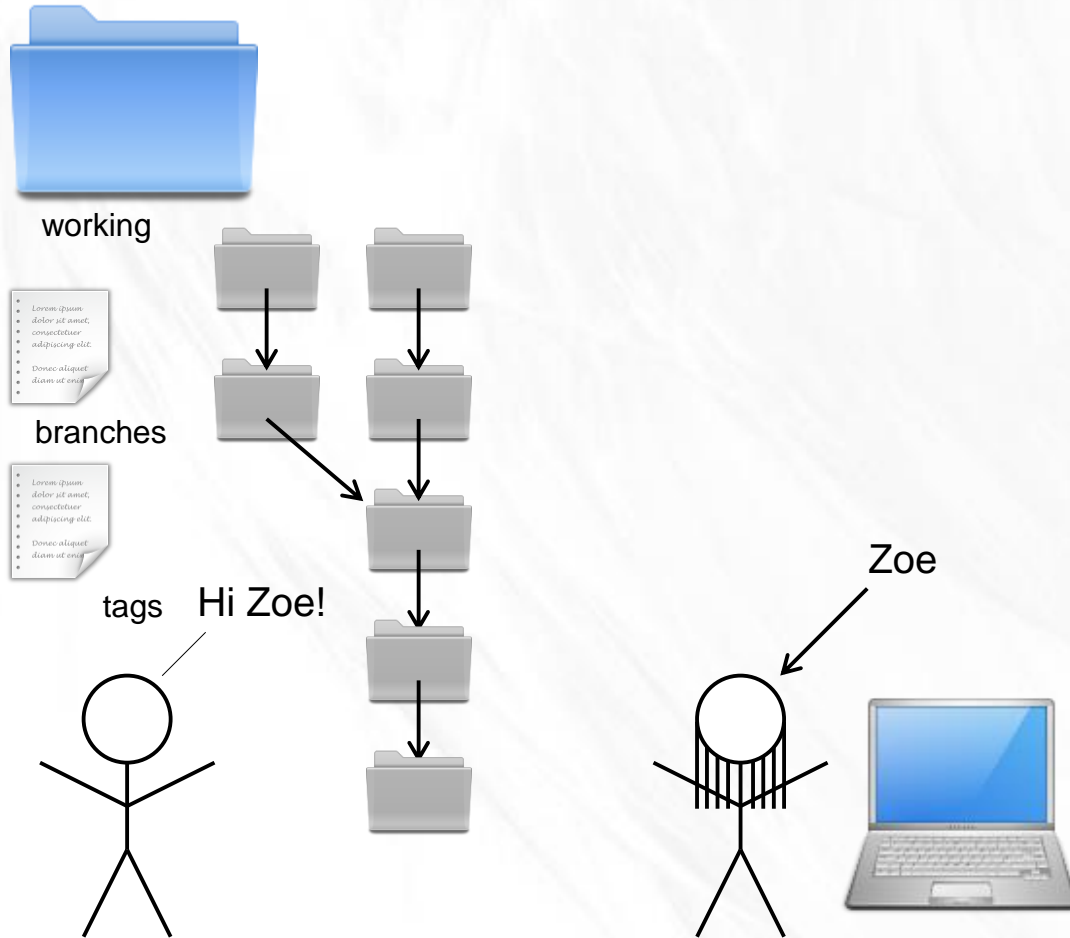
# Tags



# Distributed



# Distributed



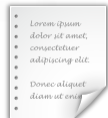
# Distributed



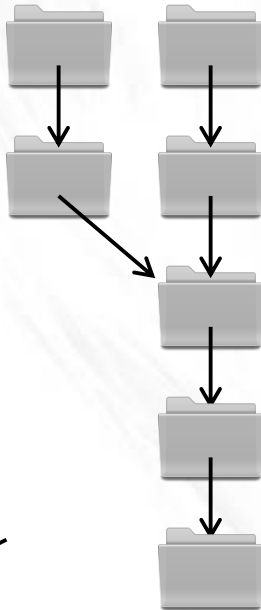
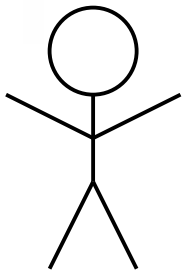
working



branches



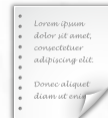
tags



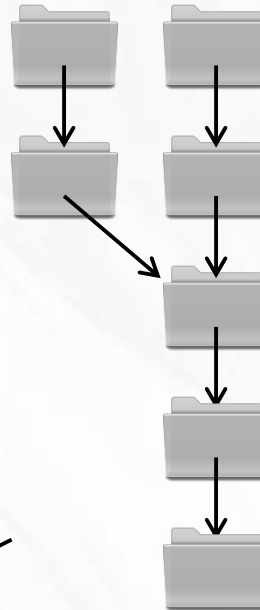
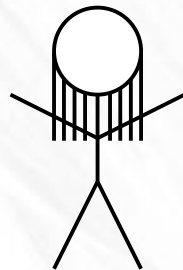
working



branches



tags



# Distributed



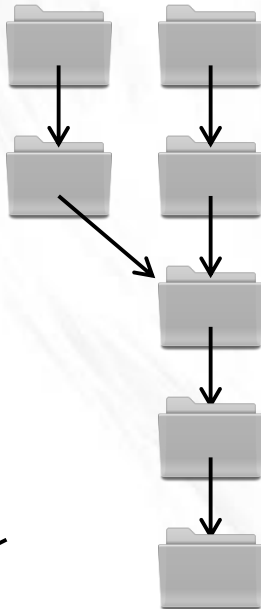
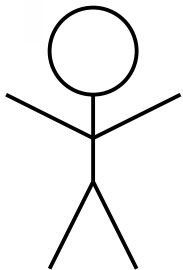
working



branches



tags



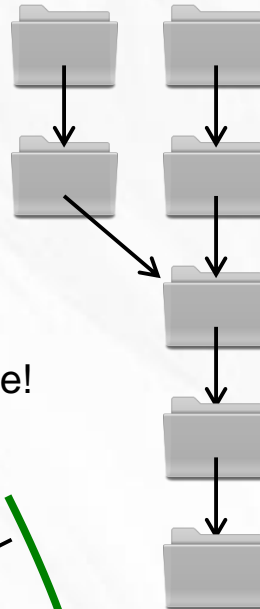
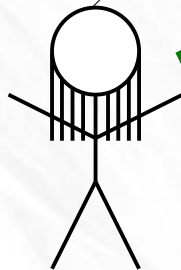
working



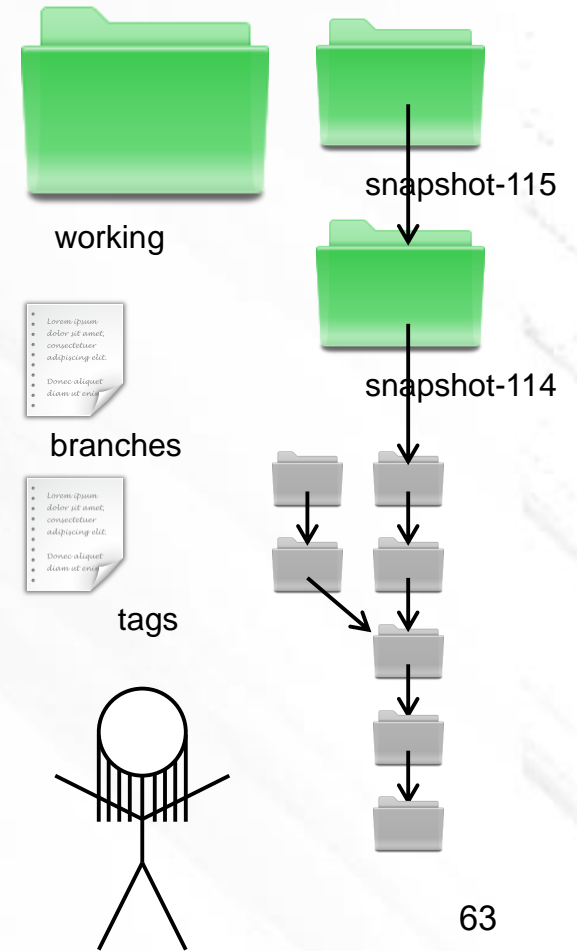
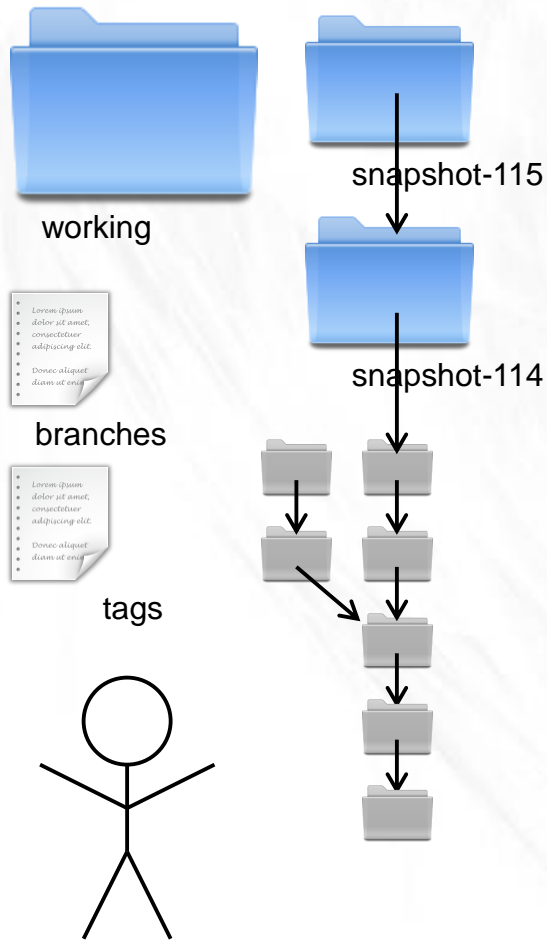
branches



tags Bye!



# Distributed



# Distributed



snapshot-114



message

2009-05-22 12:12:12  
parent: snapshot-113  
author: Me <me@me.me>

Blarfle, a cool new  
feature; extends the  
existing blog.



snapshot-114



message

2009-05-21 23:45:01  
parent: snapshot-113  
author: Zoe <zoe@z.oe>

Introduced a new foo,  
and reset the bar to 64  
xyzyy.



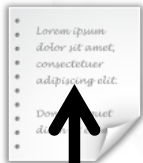
# Distributed



8ba3441b6b89cad23387ee875f2ae55069291f4b



SHA1



message

2009-05-22 12:12:12  
parent: snapshot-113  
author: Me <me@me.me>

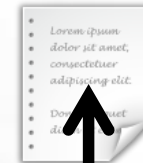
Blarfle, a cool new  
feature; extends the  
existing blog.



db9ecb5b5a6294a8733503ab57577db96ff2249e



SHA1

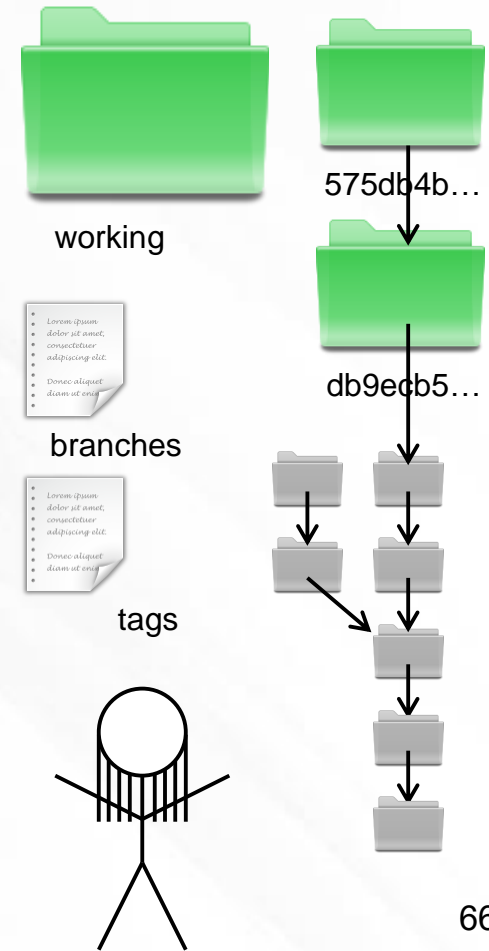
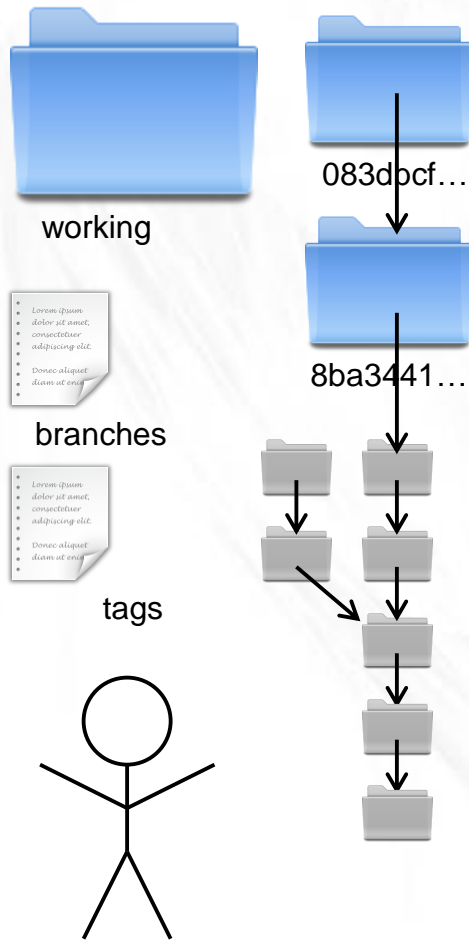


message

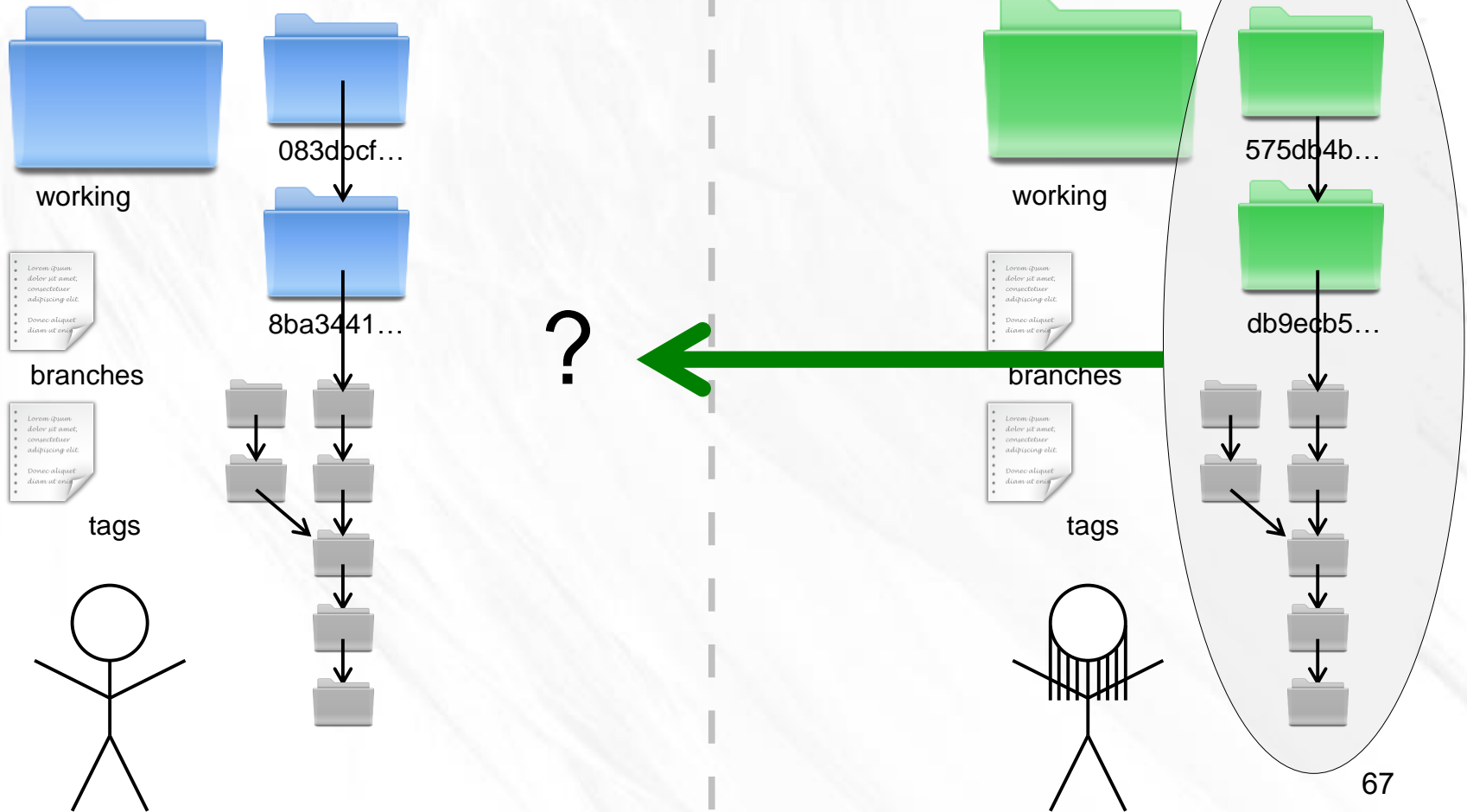
2009-05-21 23:45:01  
parent: snapshot-113  
author: Zoe <zoe@z.oe>

Introduced a new foo, 65  
and reset the bar to  
xyzyz.

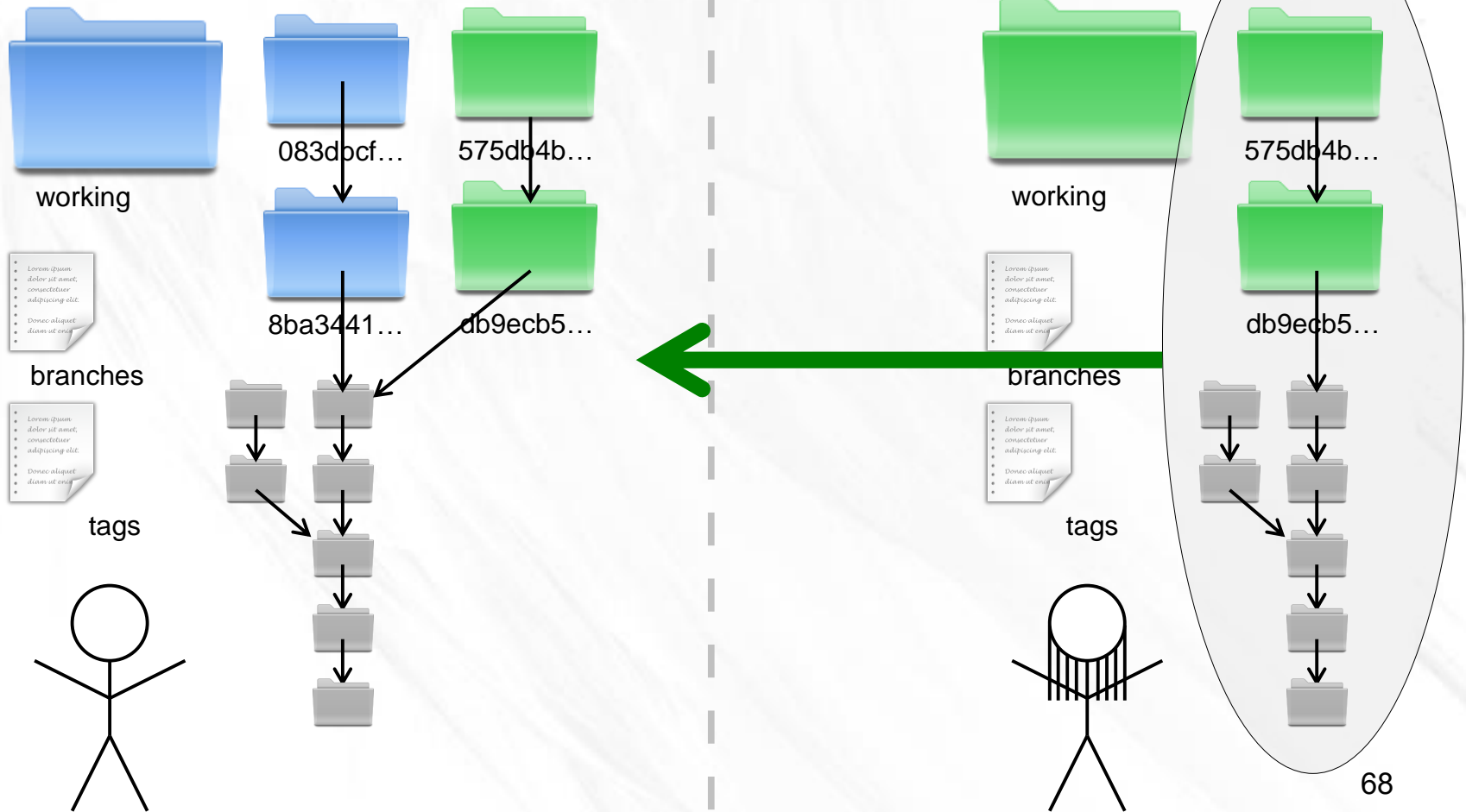
# Distributed



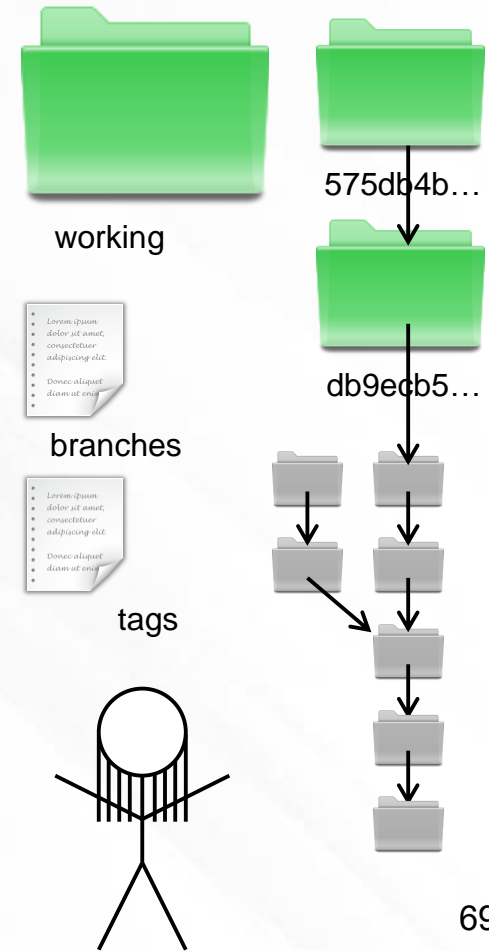
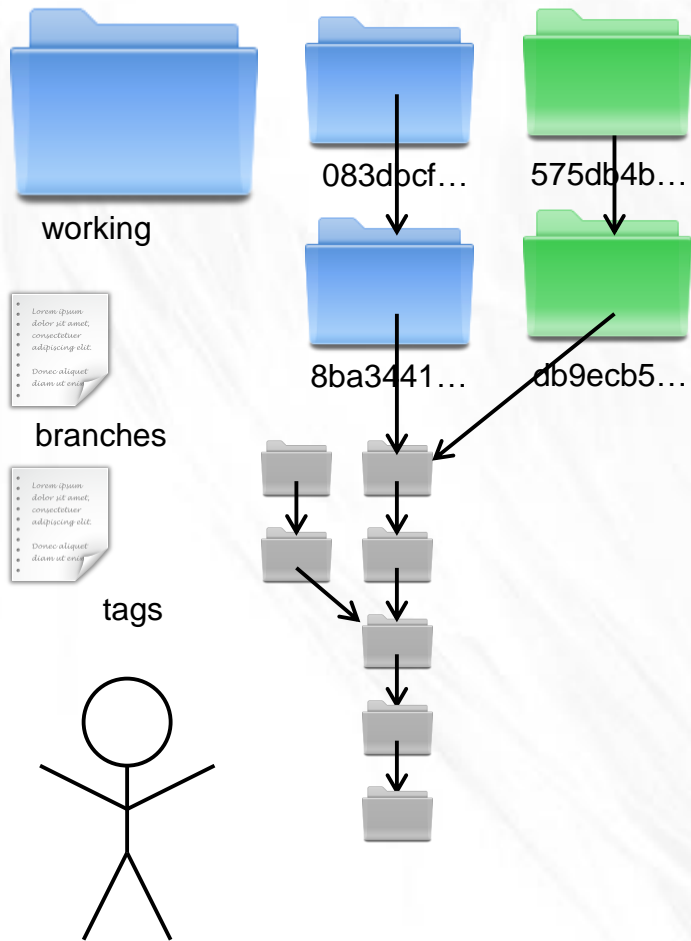
# Distributed



# Distributed



# Distributed



# Offline



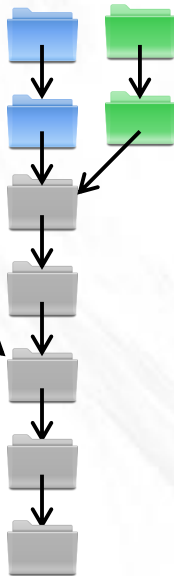
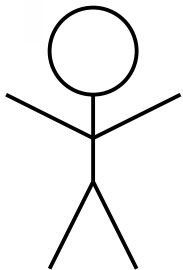
working



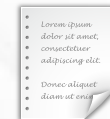
branches



tags



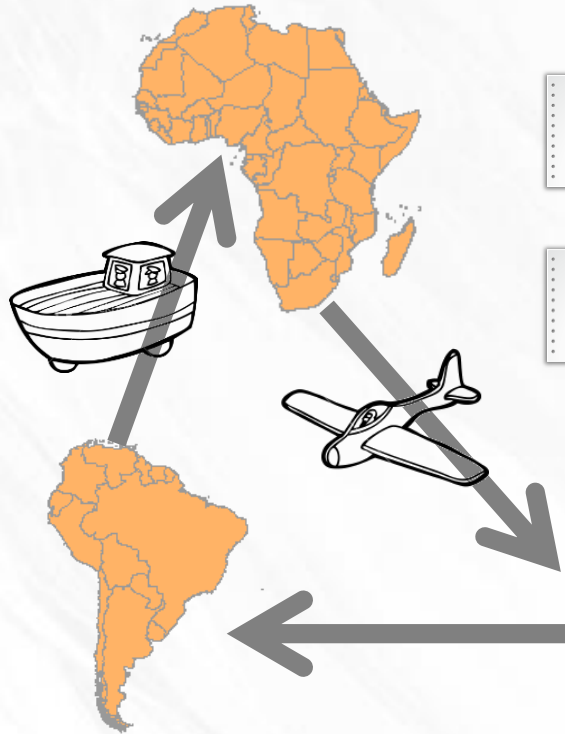
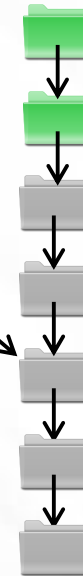
working



branches



tags  
w00t!



# Offline



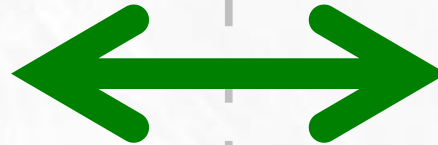
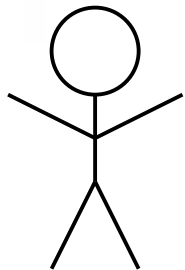
working



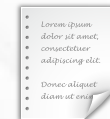
branches



tags



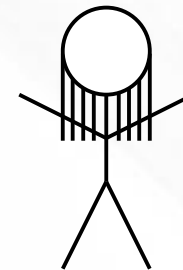
working



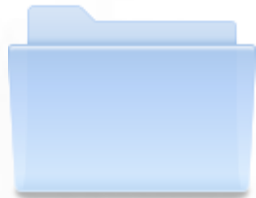
branches



tags



# (simpler drawings)



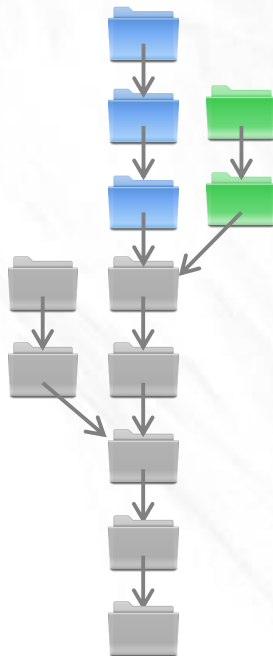
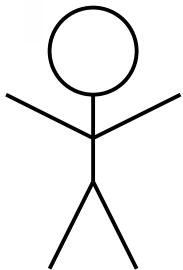
working



branches



tags



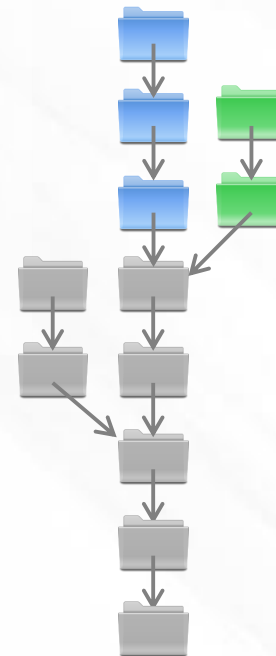
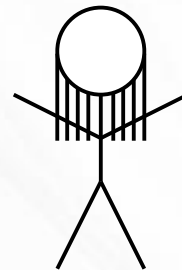
working



branches

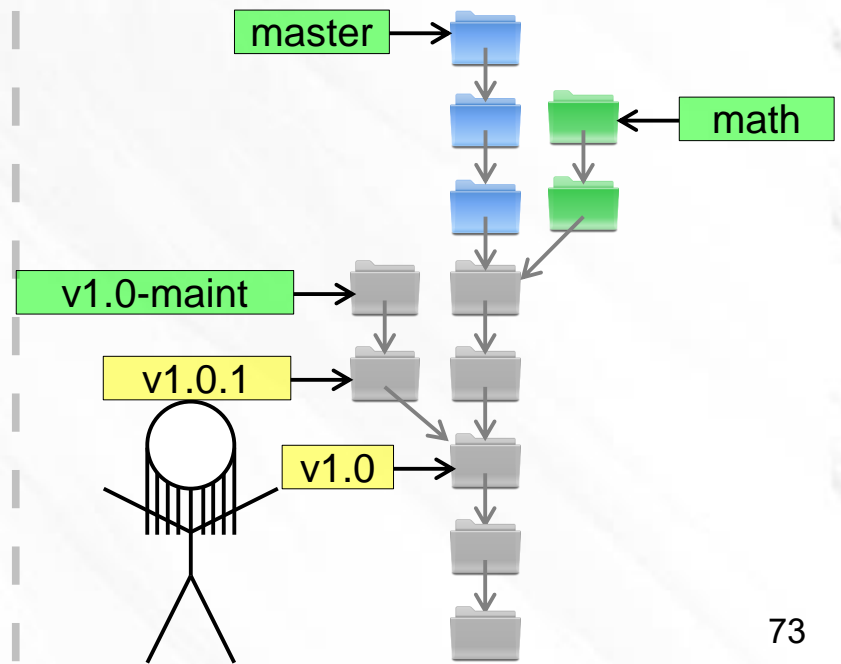
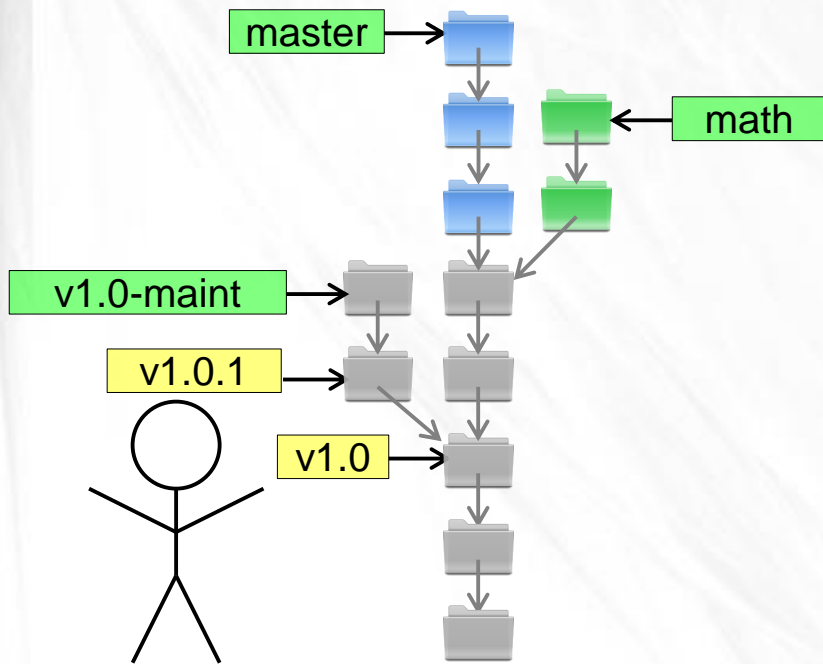


tags

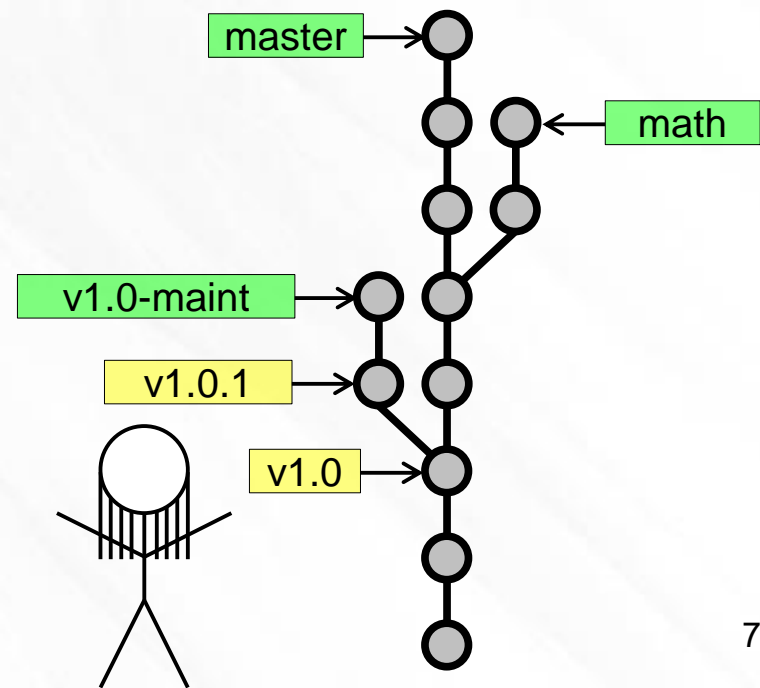
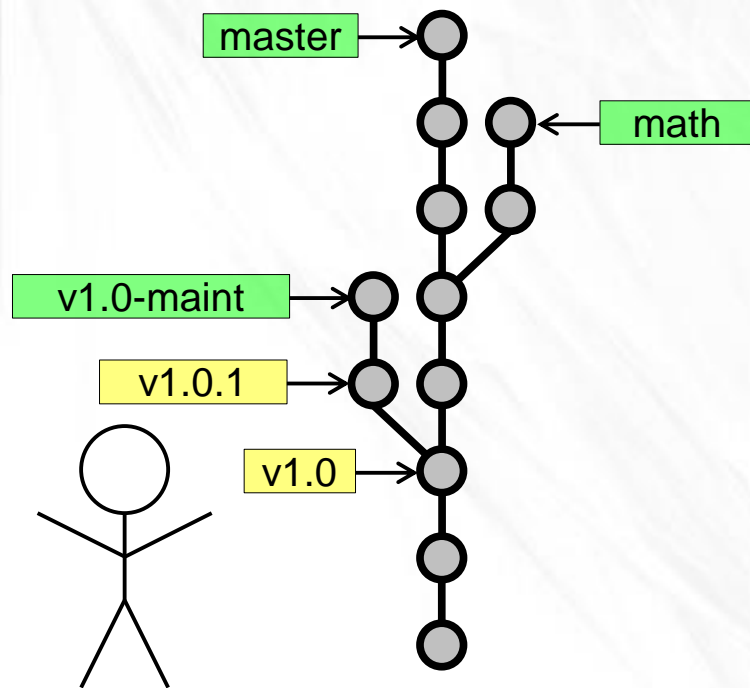




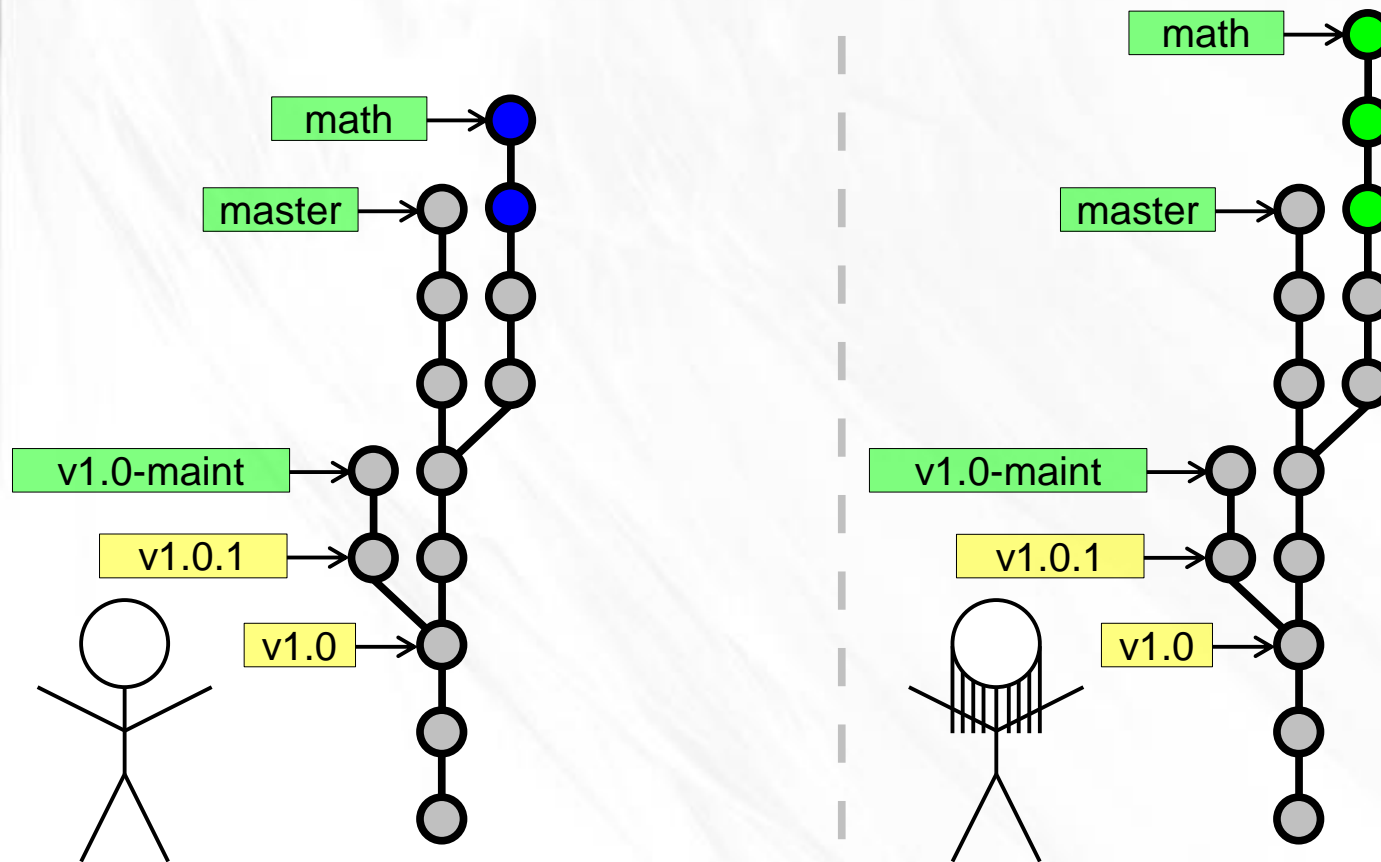
# (simpler drawings)



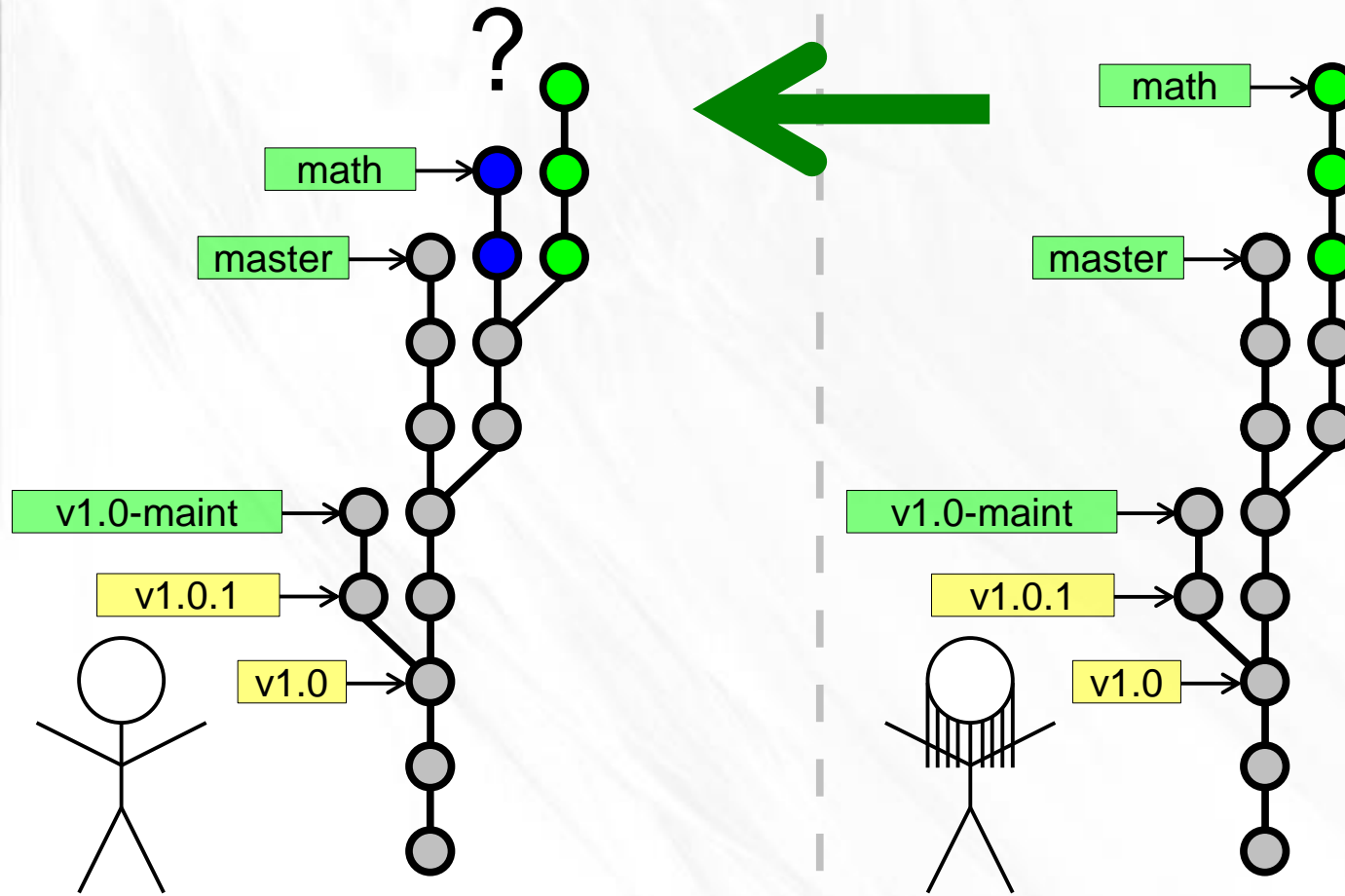
# (simpler drawings)



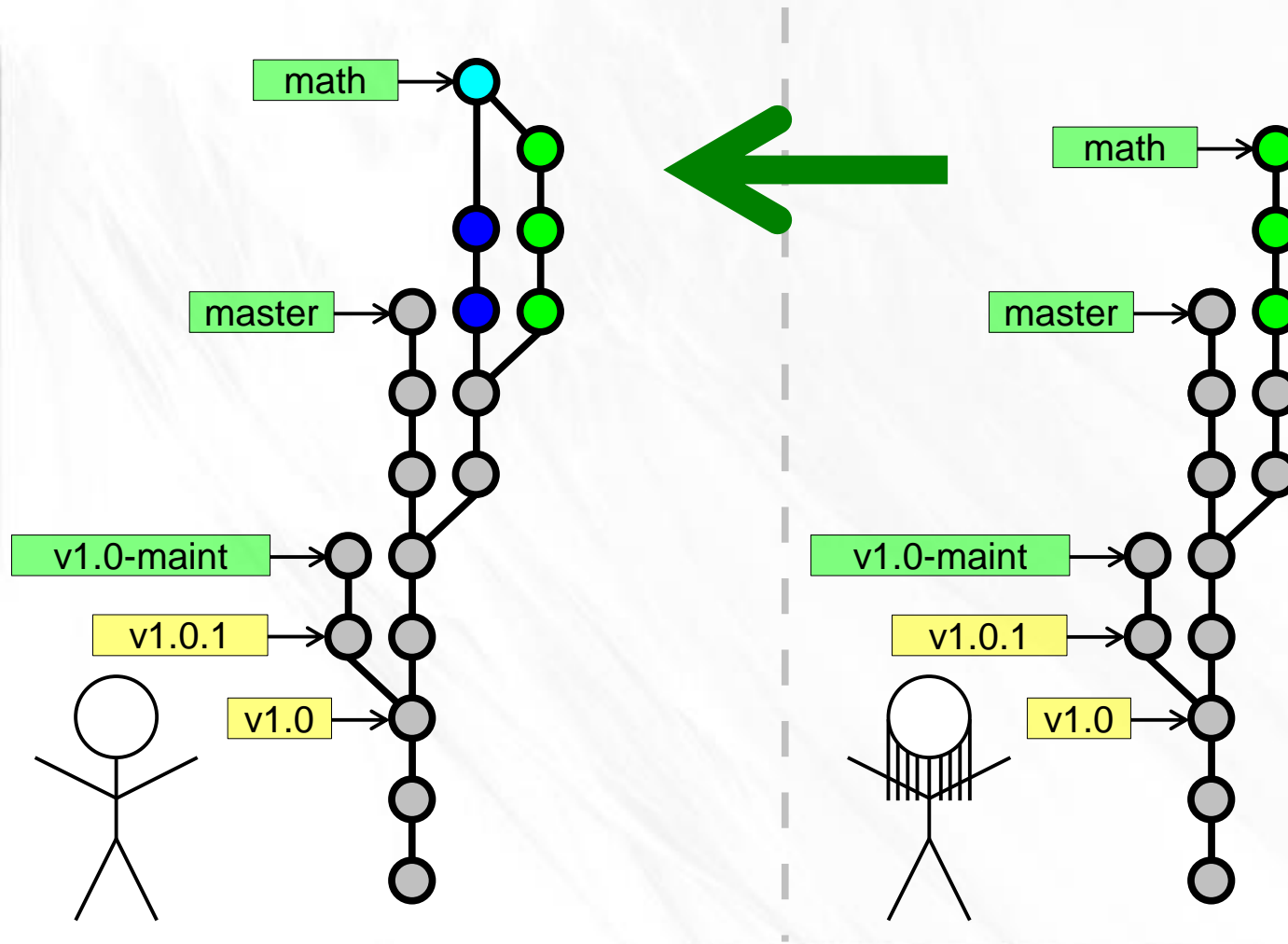
# Merges



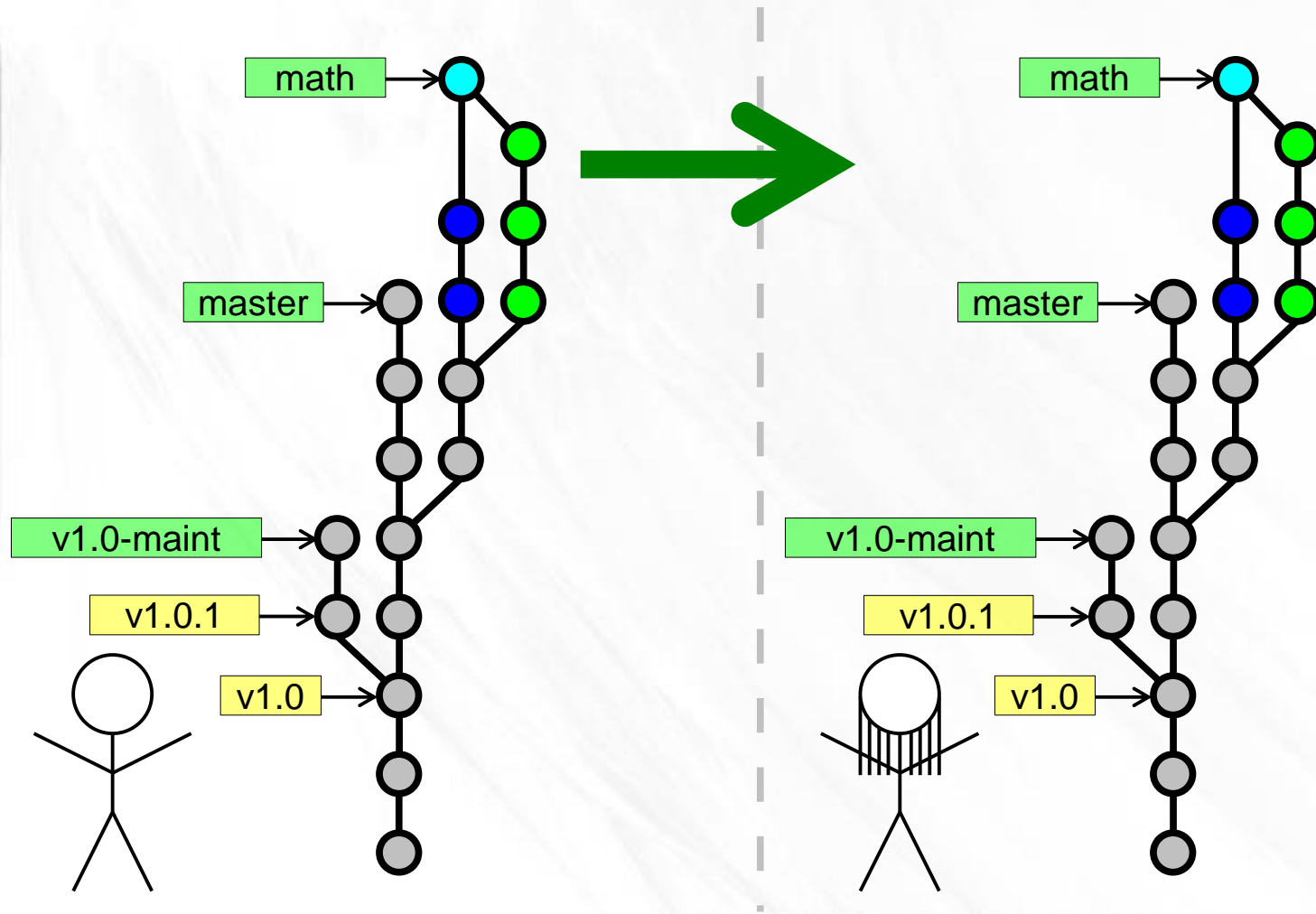
# Merges



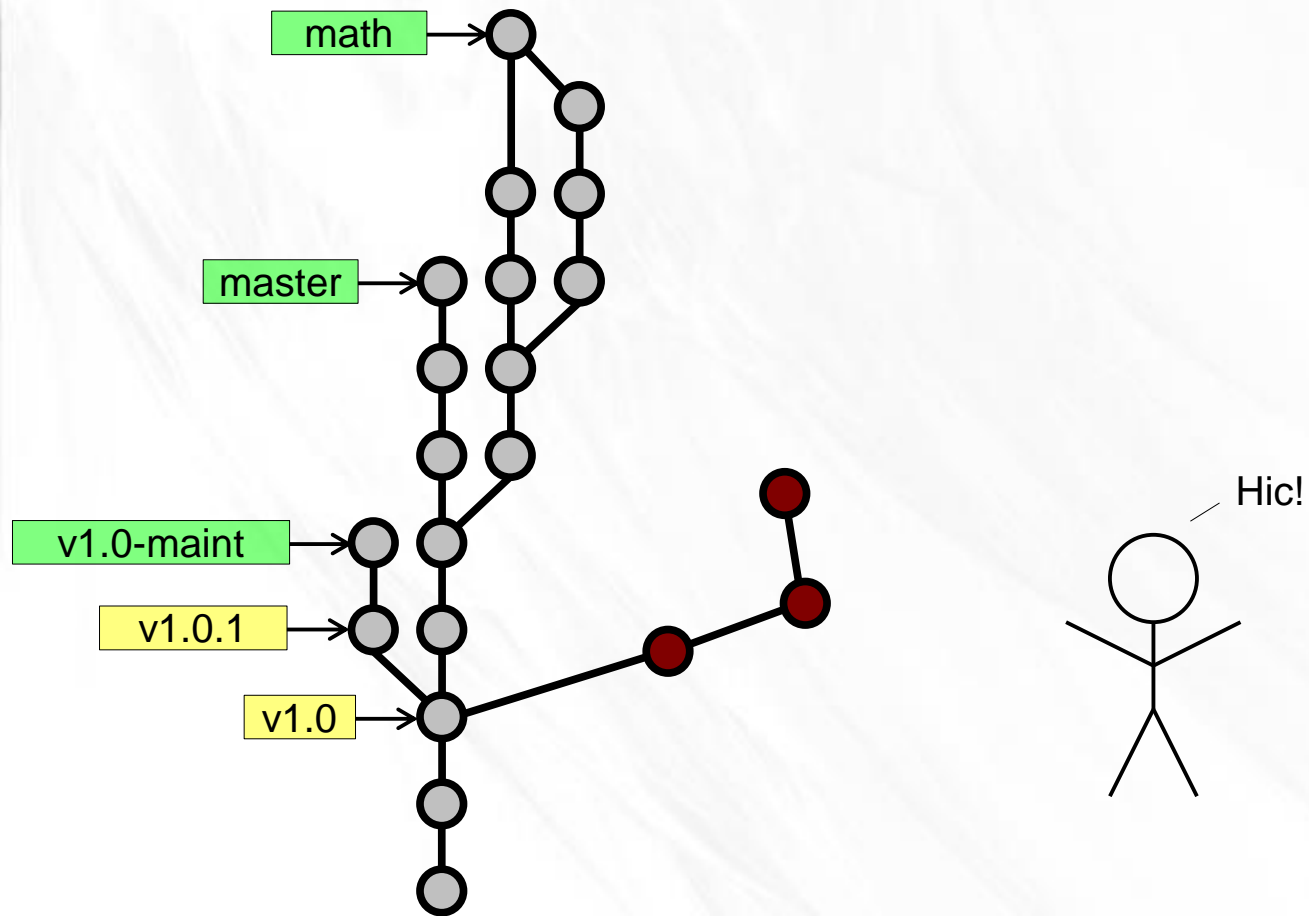
# Merges



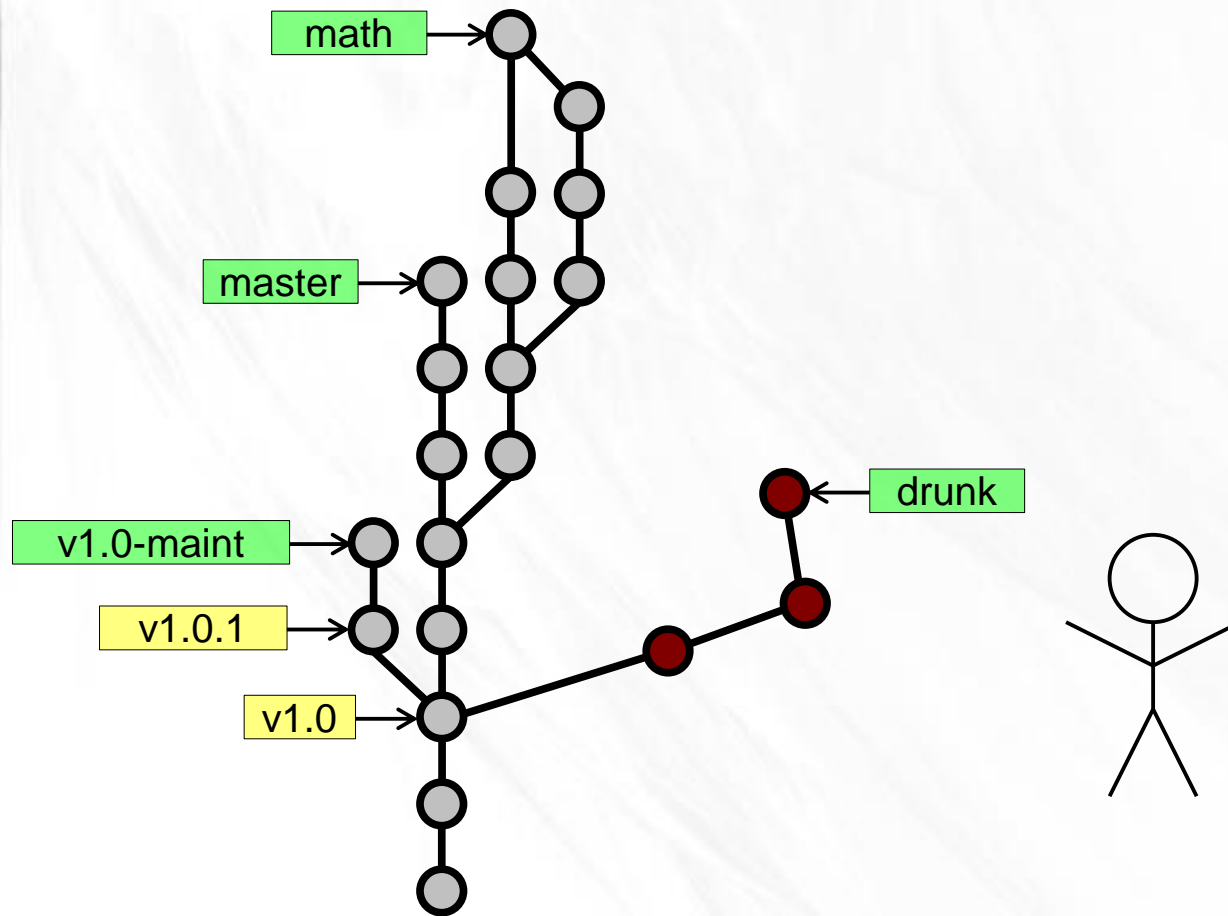
# Merges



# Rewriting History

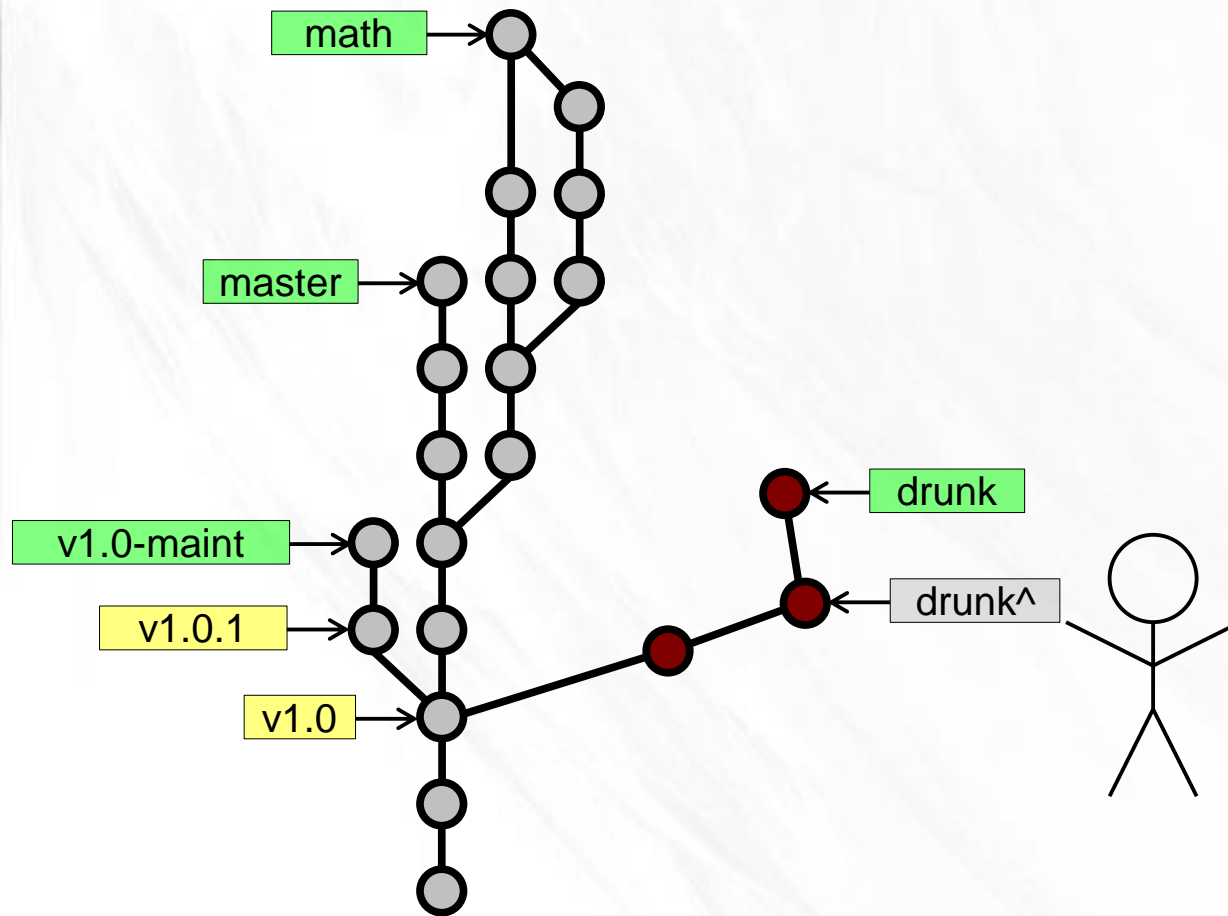


# Rewriting History

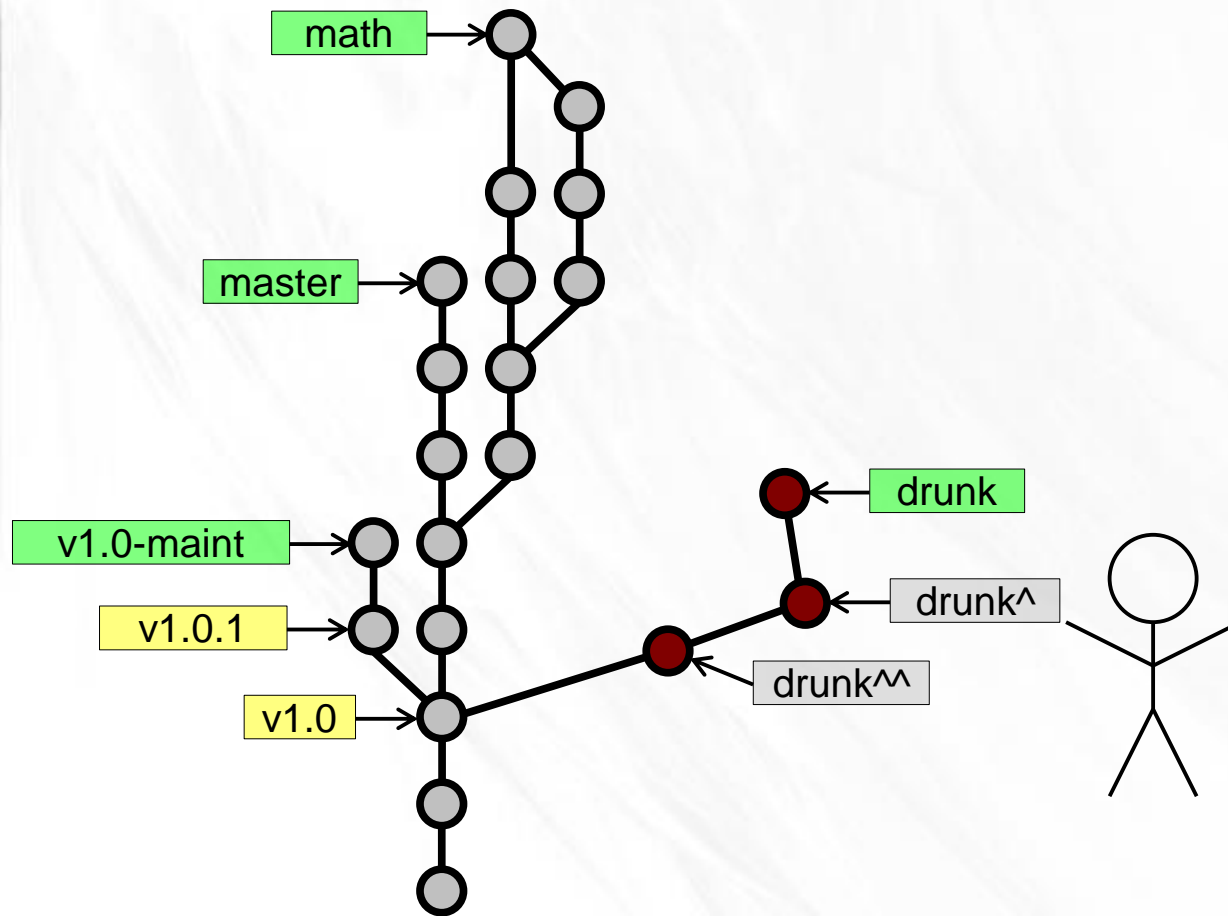




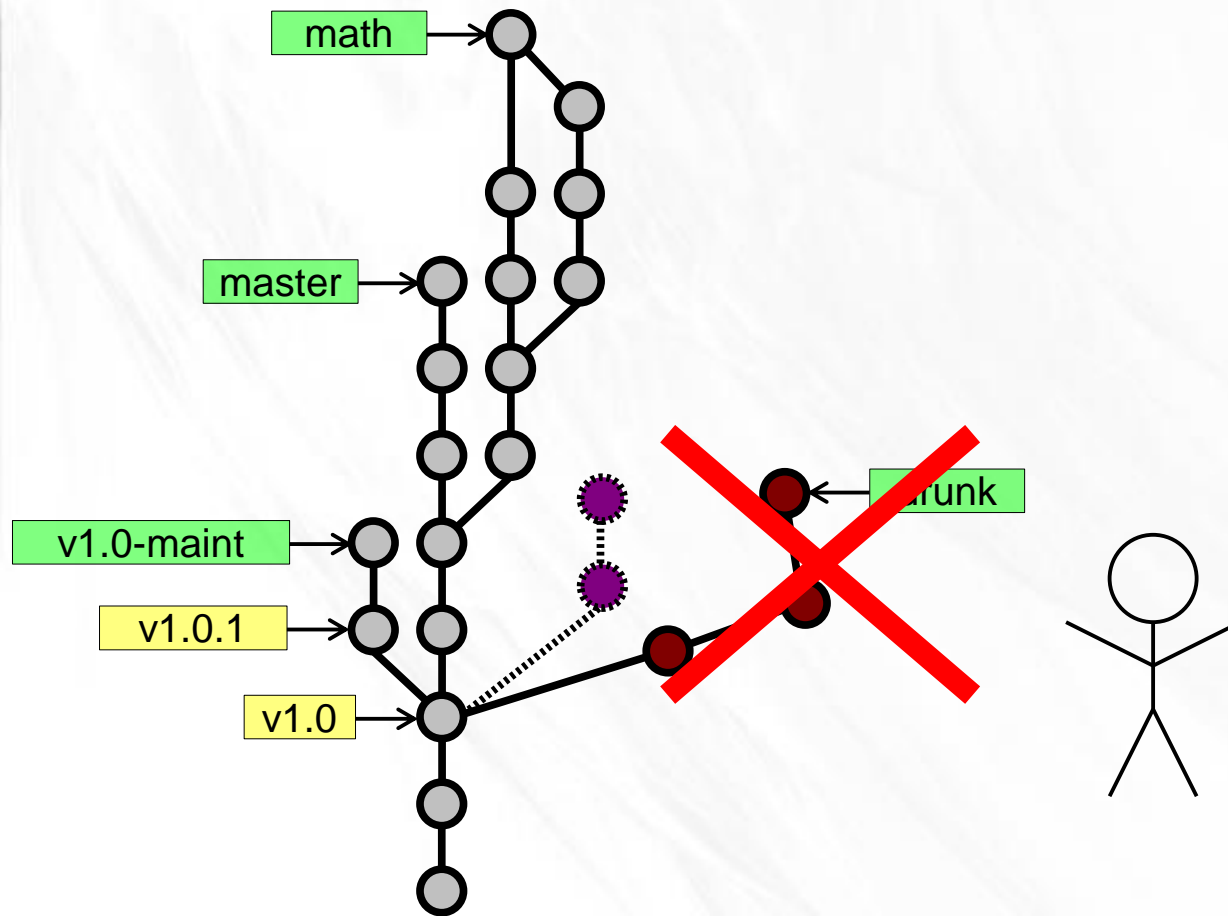
# Rewriting History



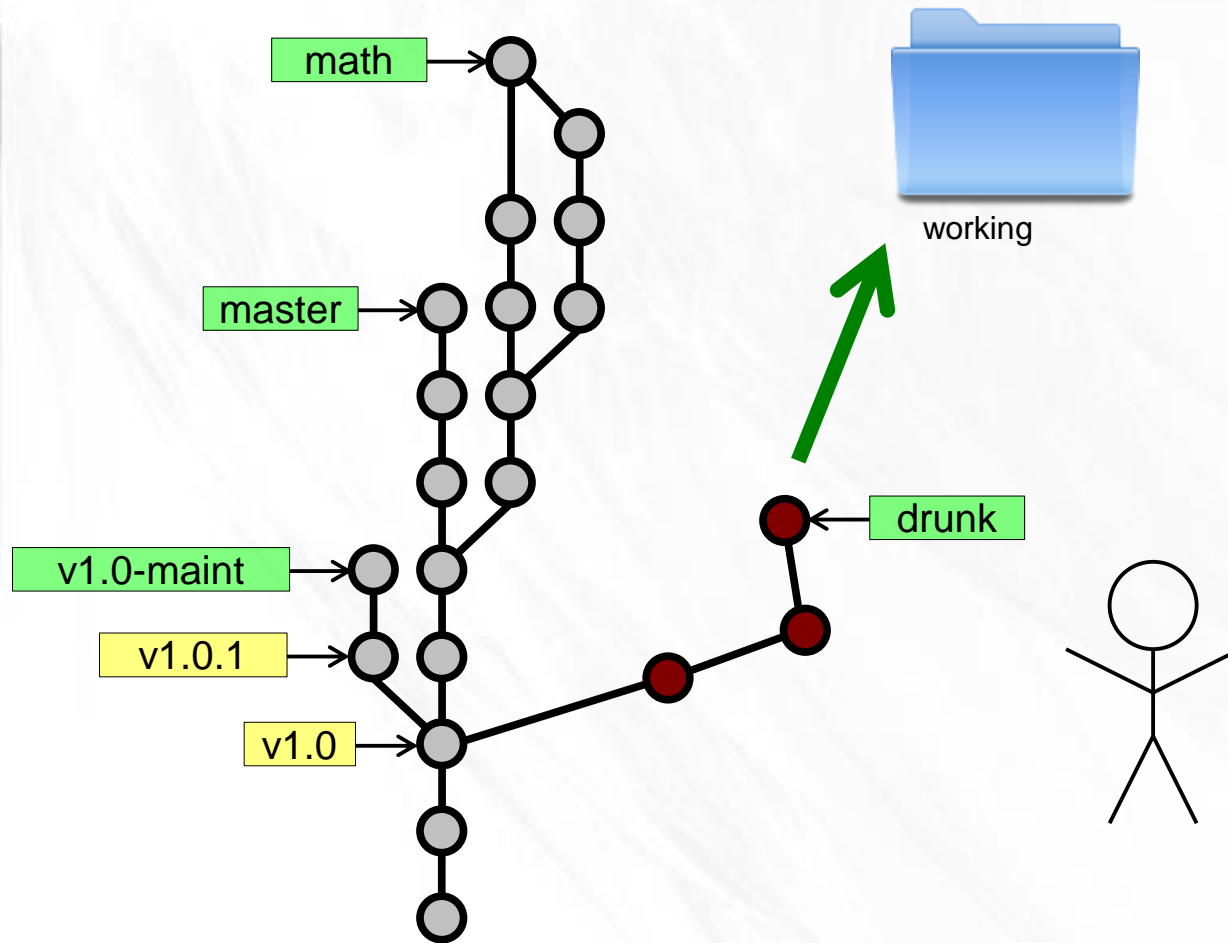
# Rewriting History



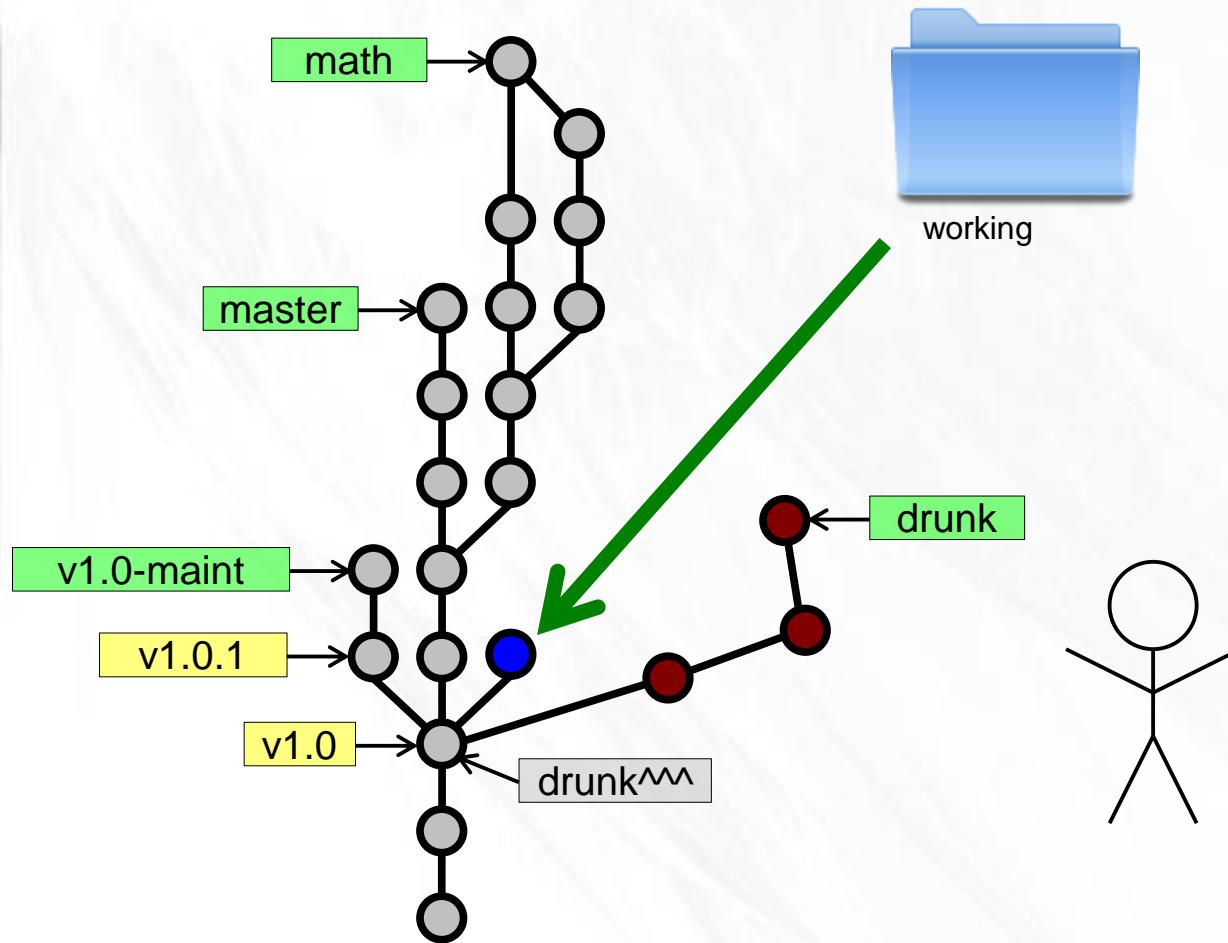
# Rewriting History



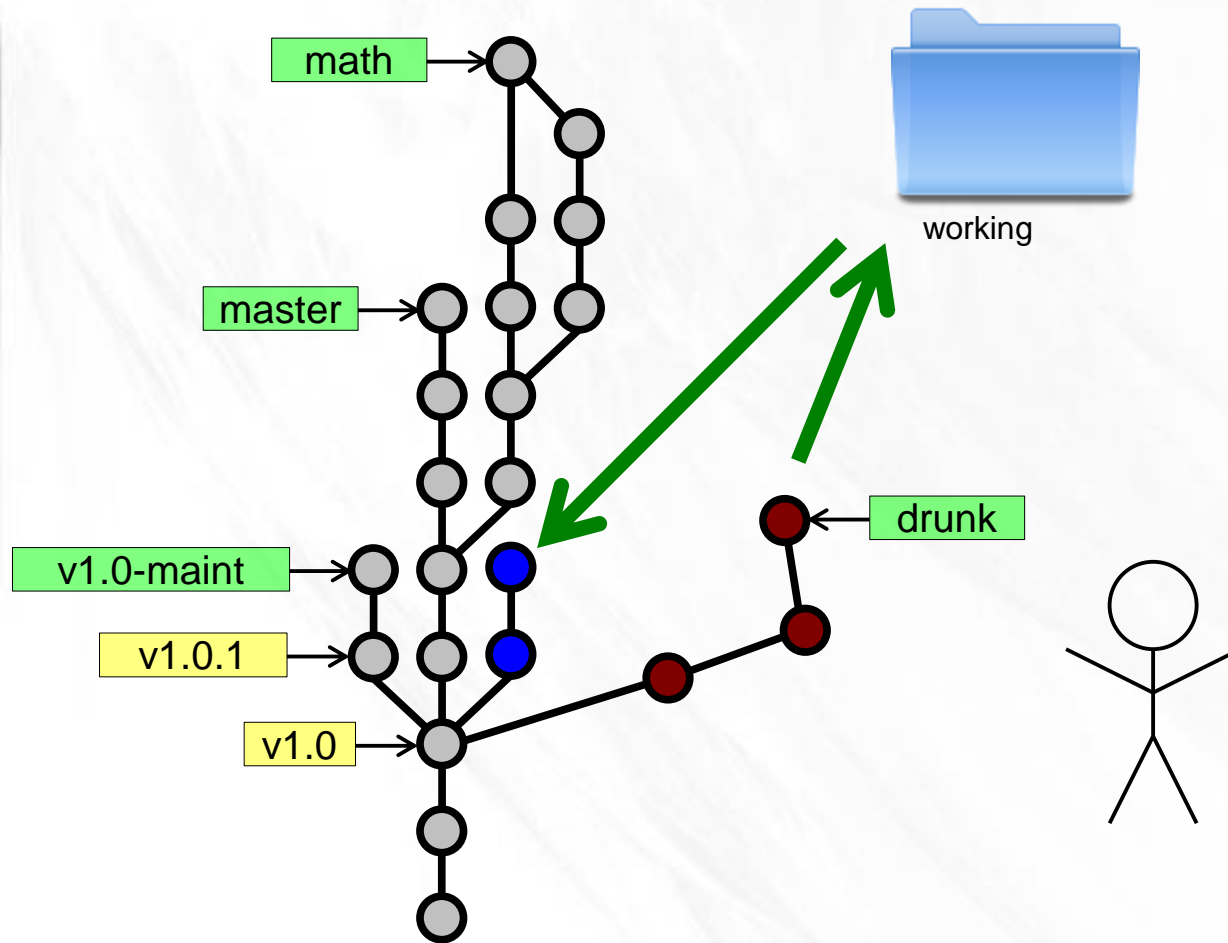
# Rewriting History



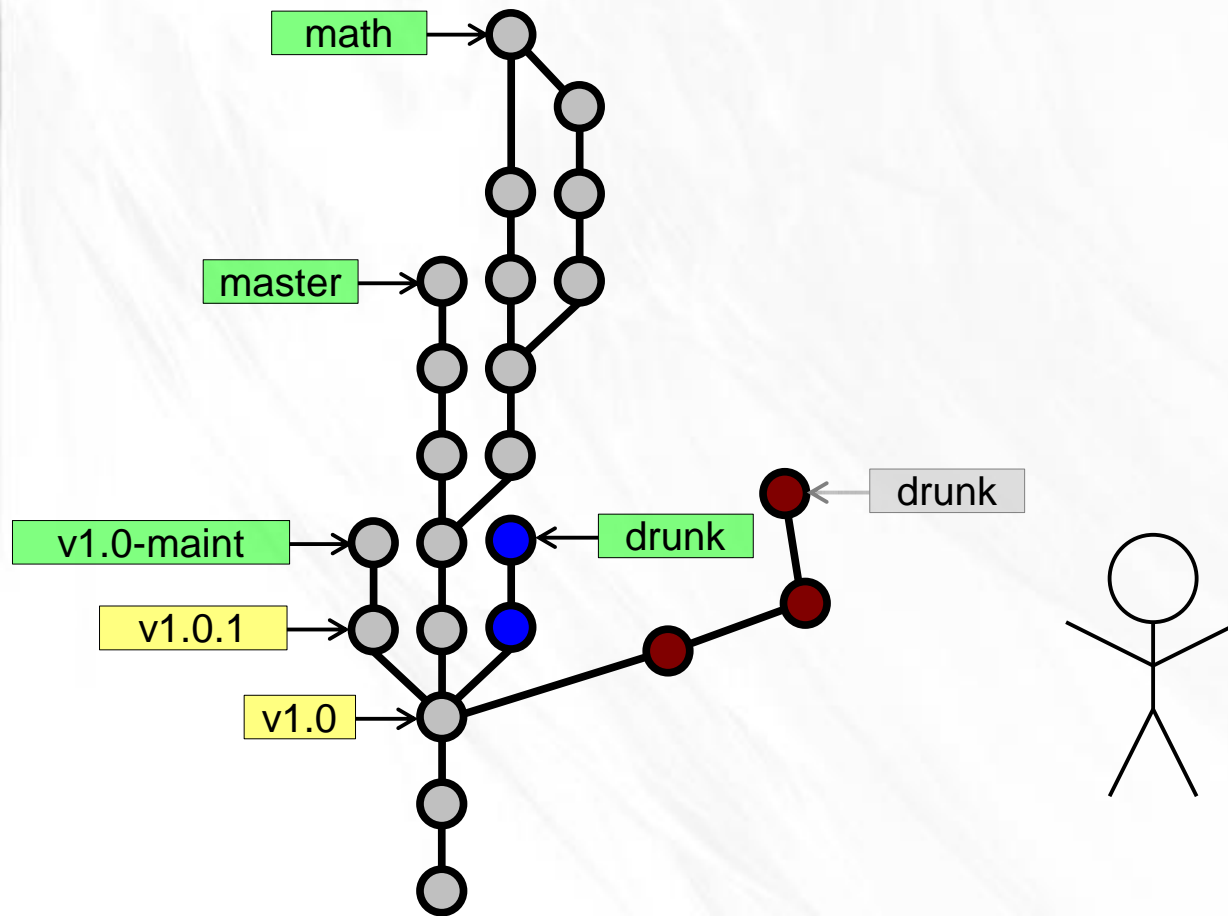
# Rewriting History



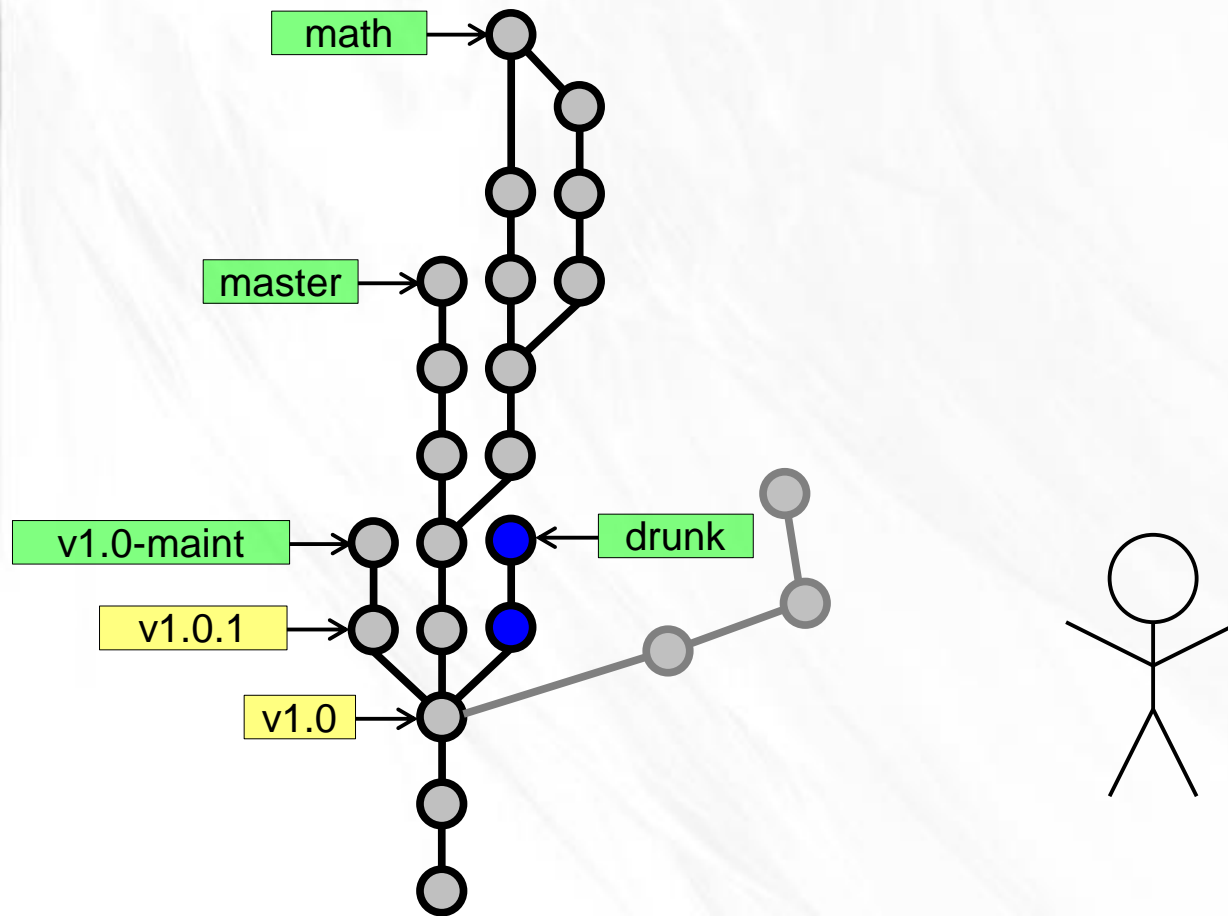
# Rewriting History



# Rewriting History

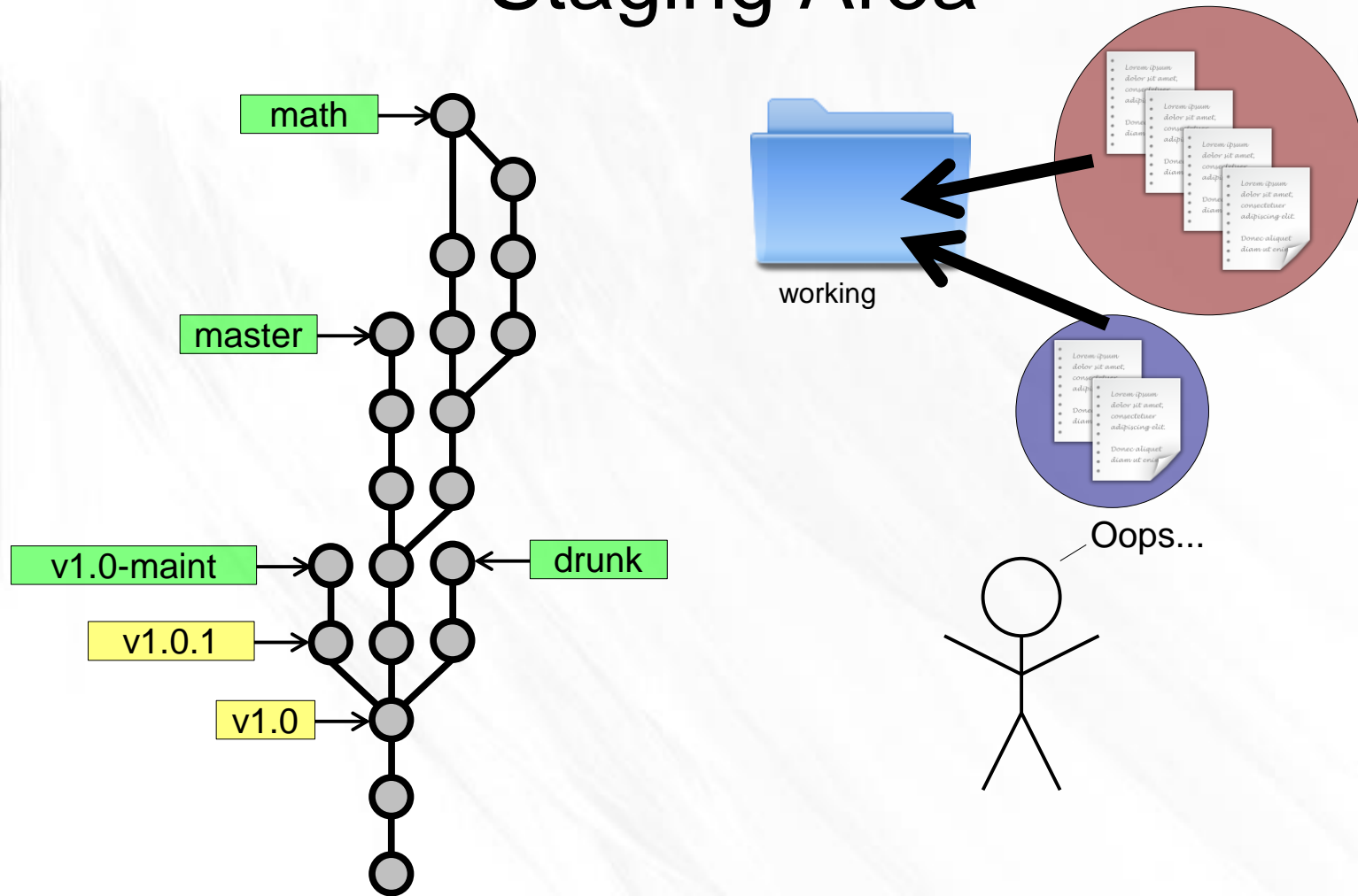


# Rewriting History

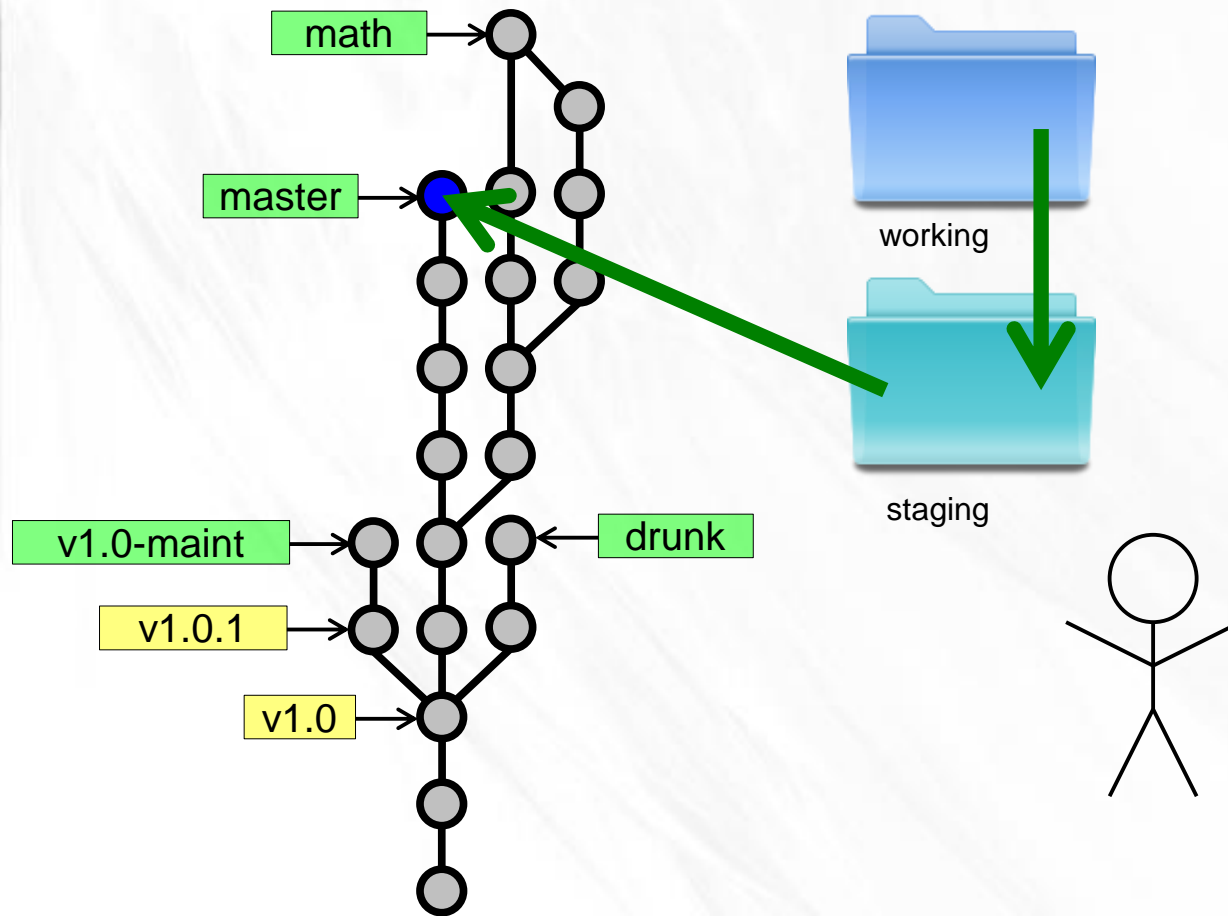




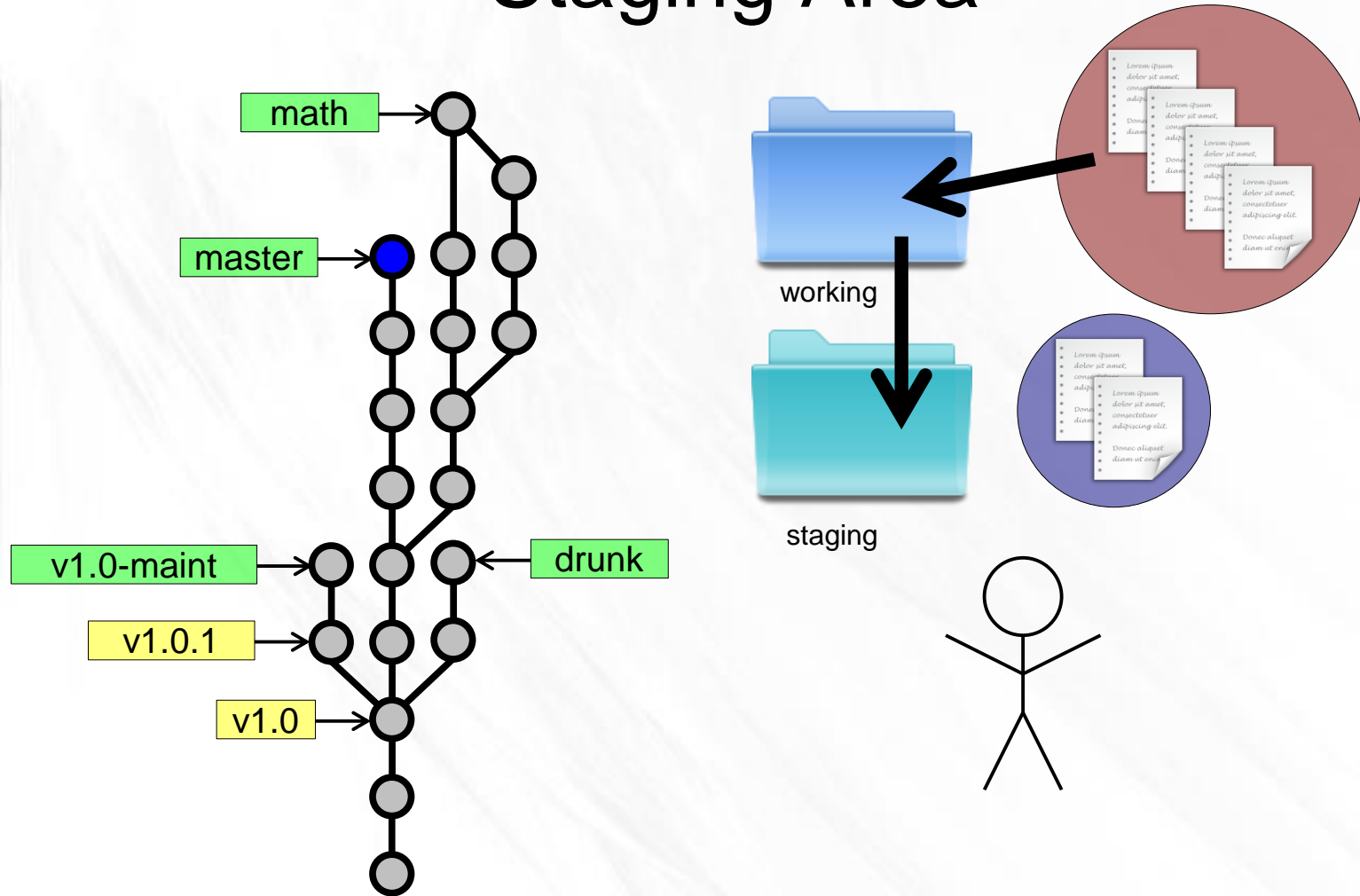
# Staging Area



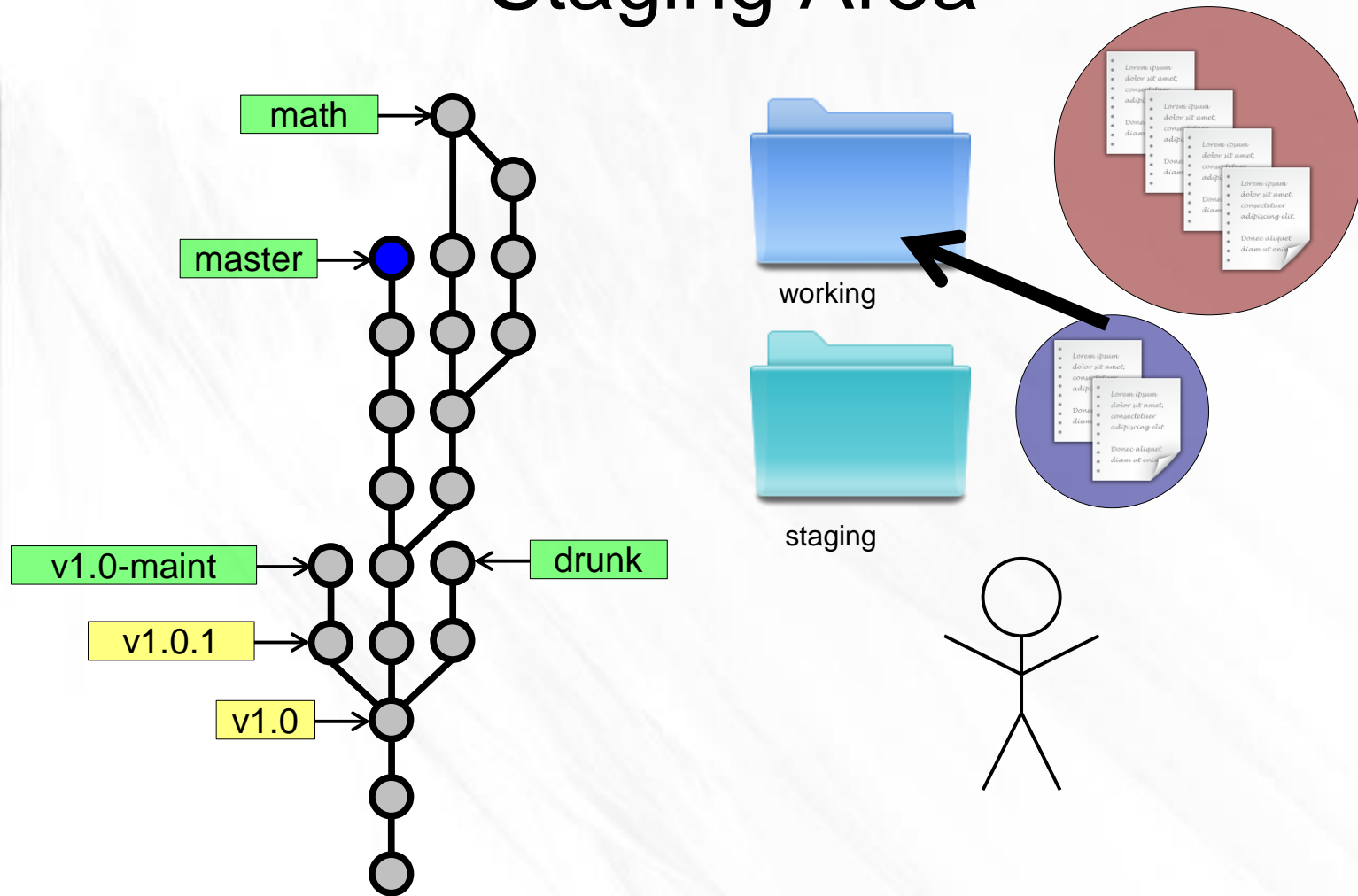
# Staging Area



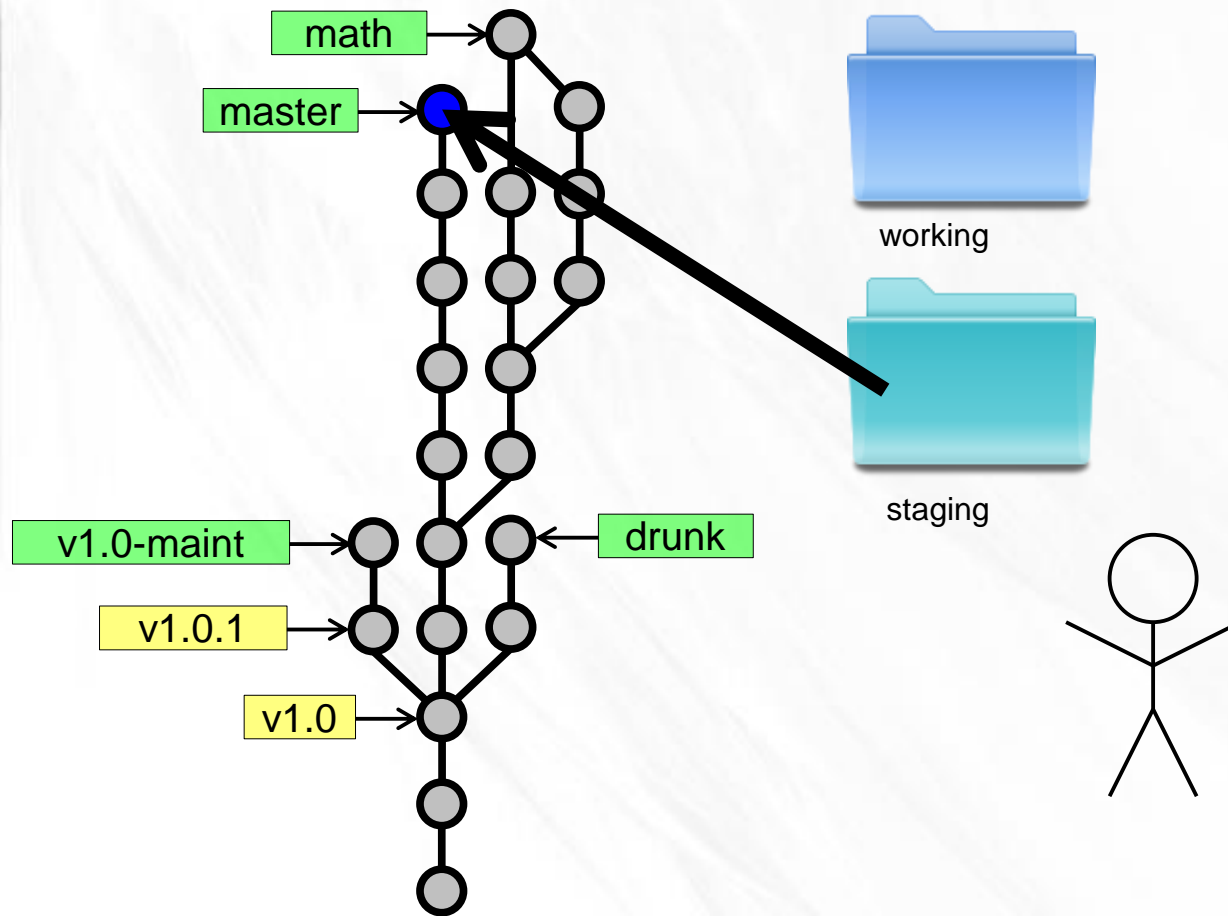
# Staging Area



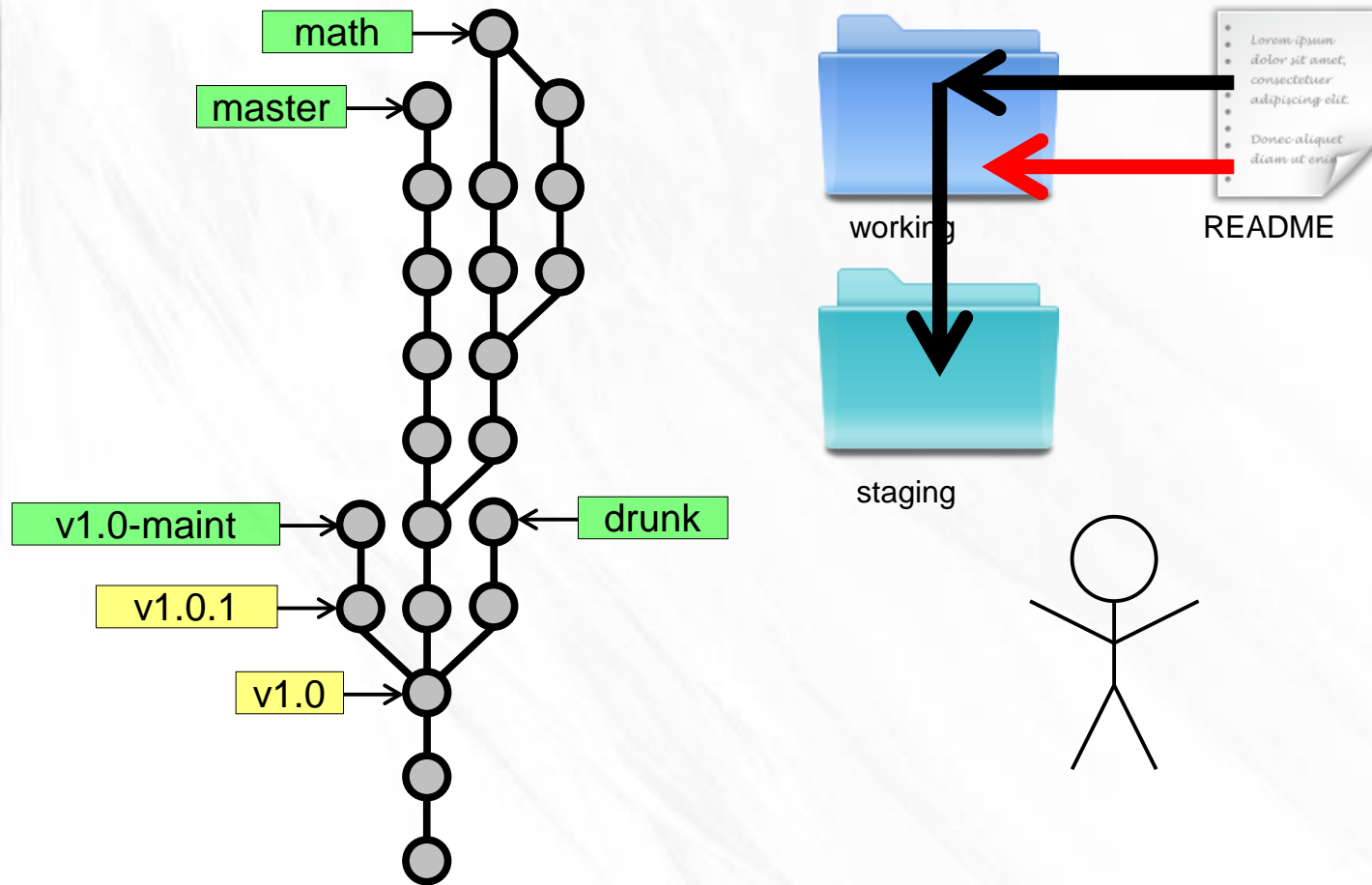
# Staging Area



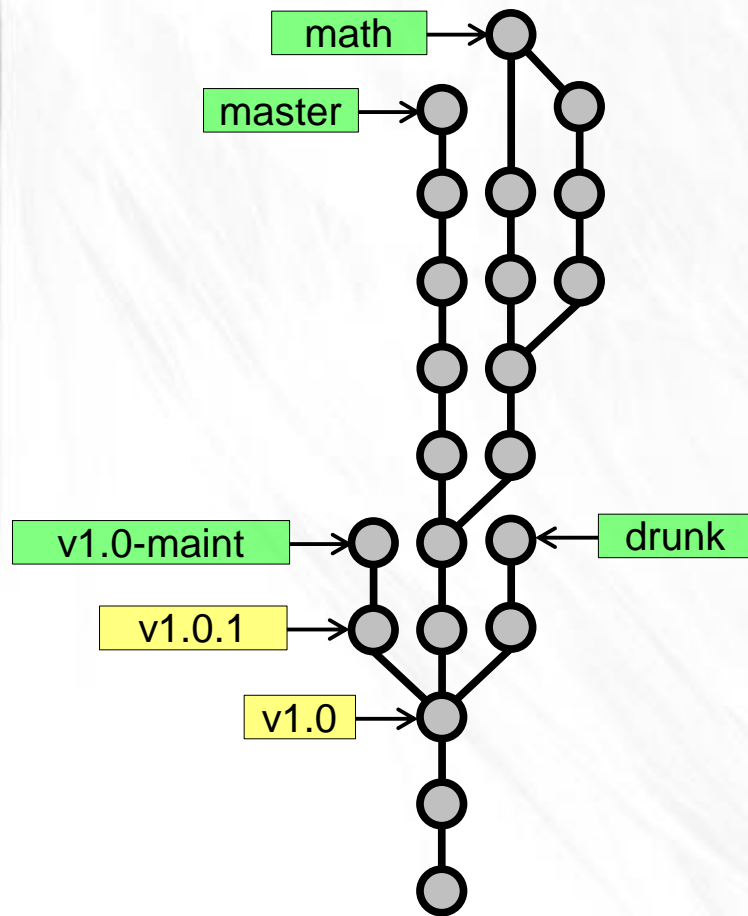
# Staging Area



# Staging Area



# Diffs



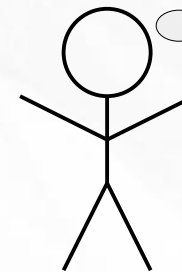
working



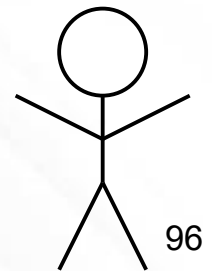
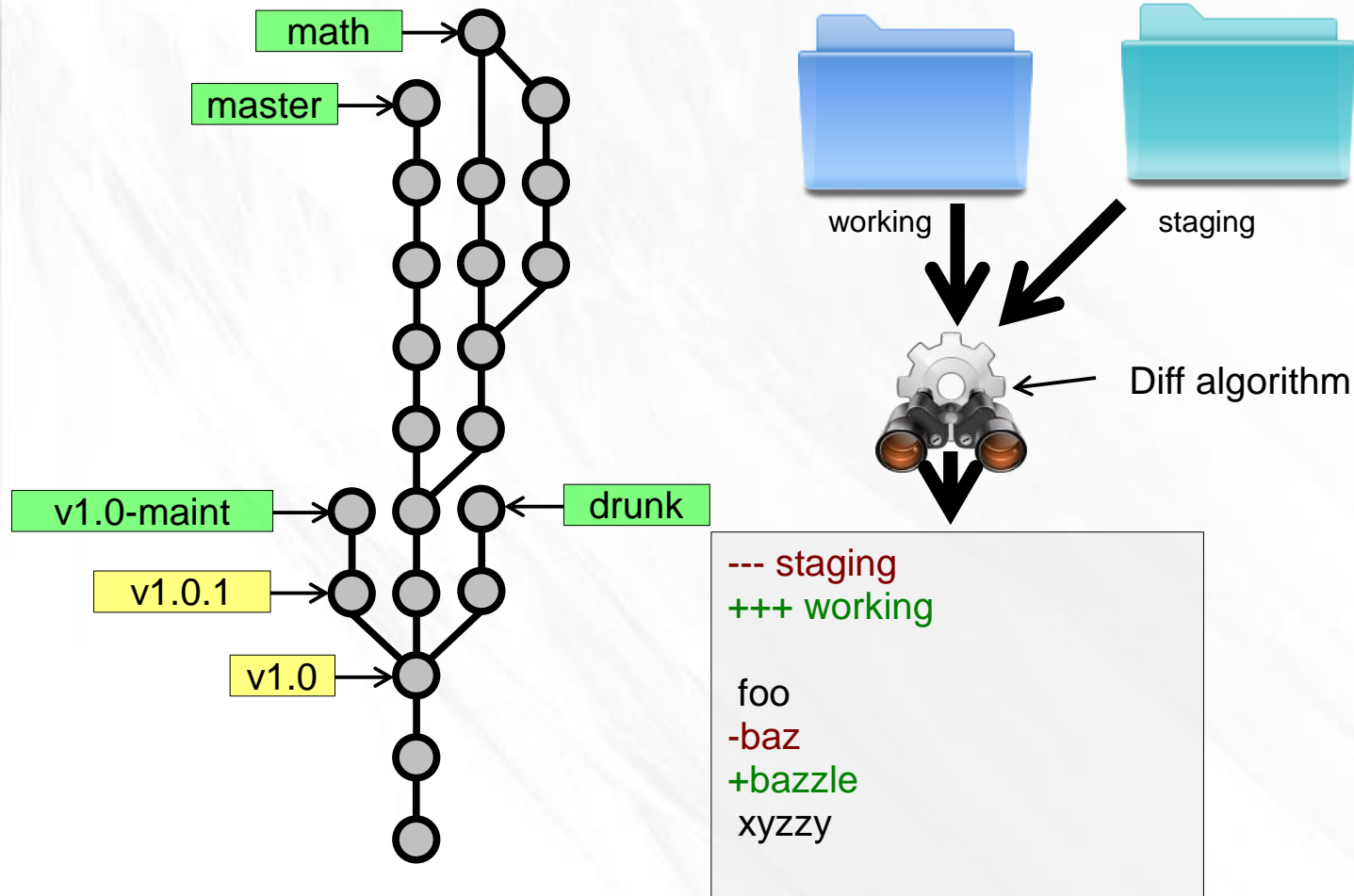
staging

What are the changes?

- working vs. staging
- working vs. snapshot X
- staging vs. snapshot X
- snapshot X vs. snapshot Y

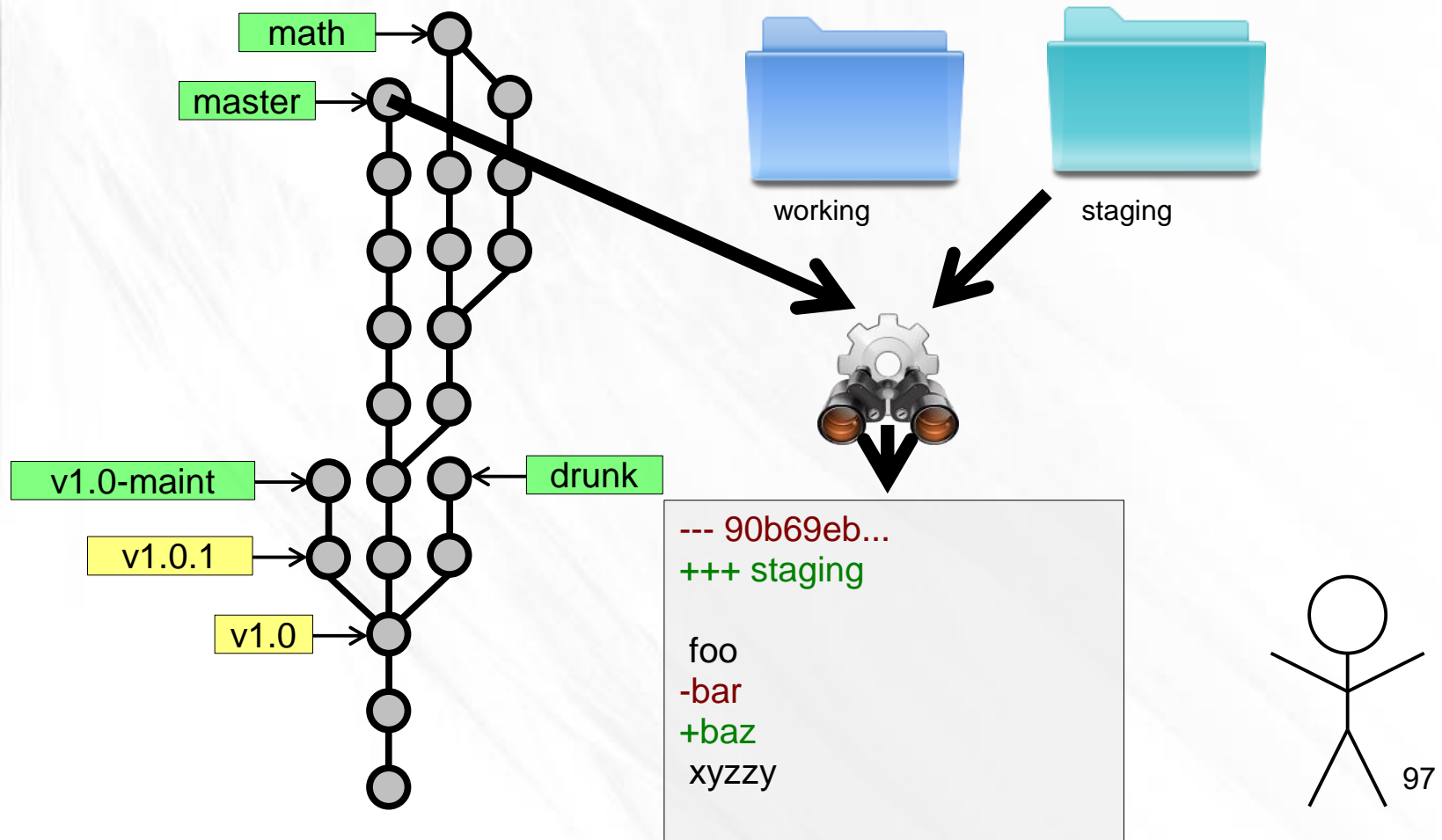


# Diffs

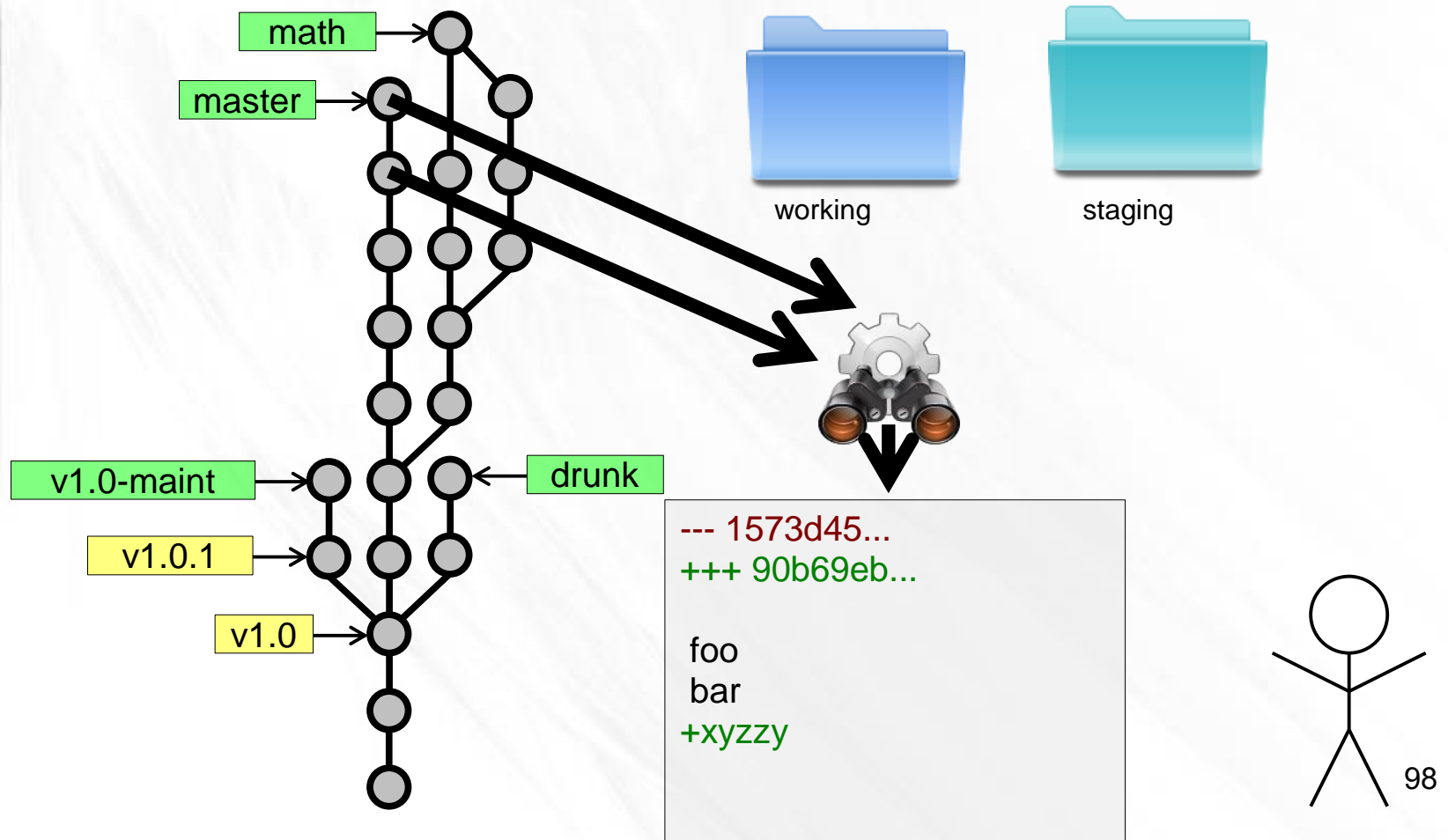




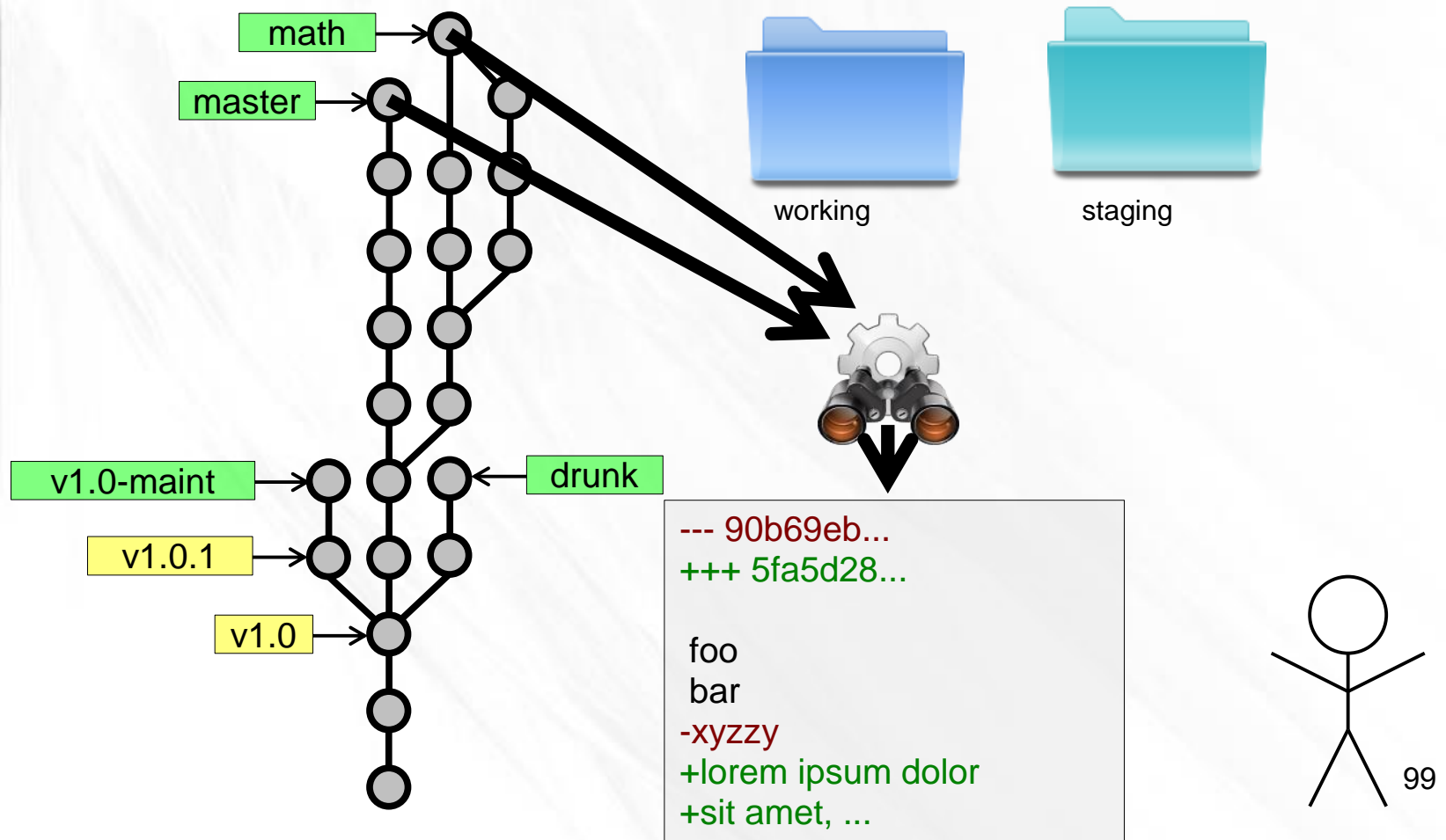
# Diffs



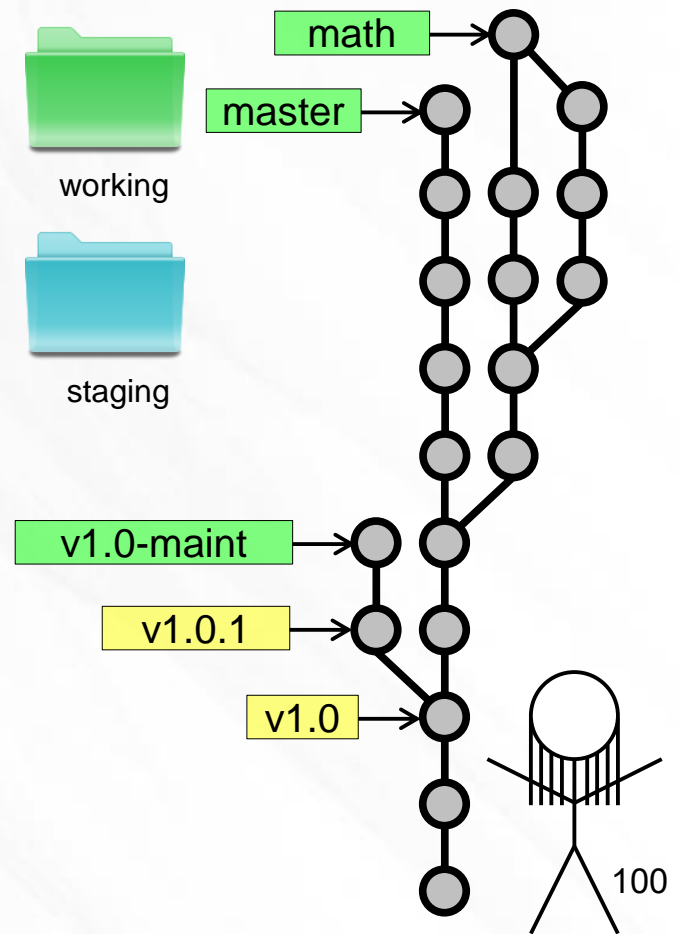
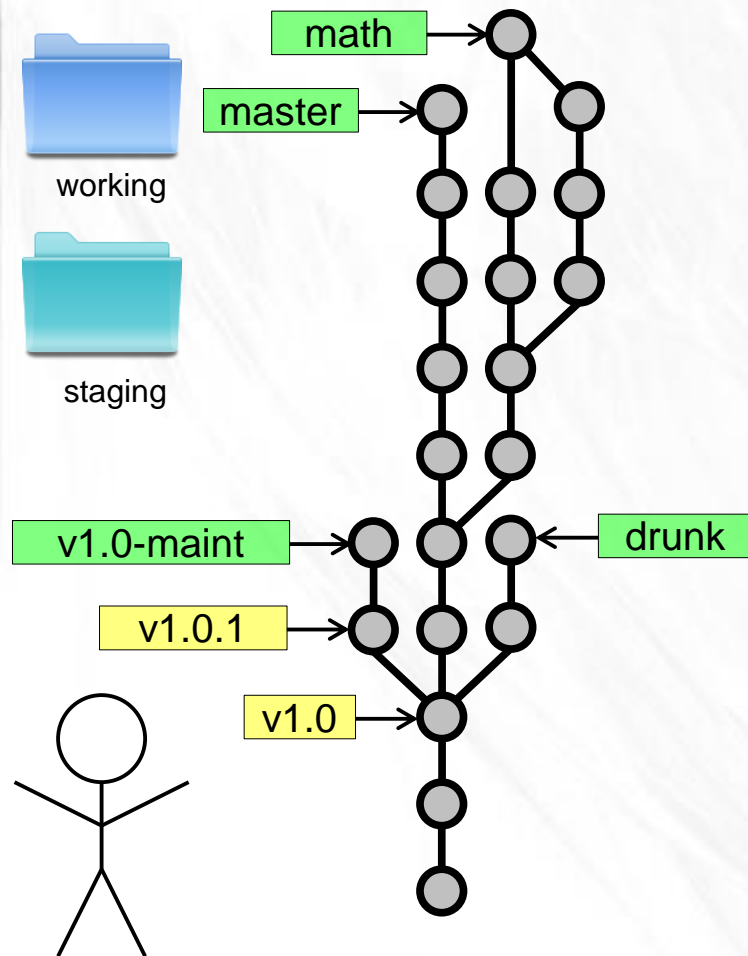
# Diffs



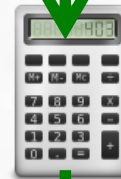
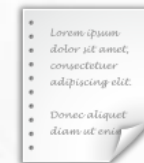
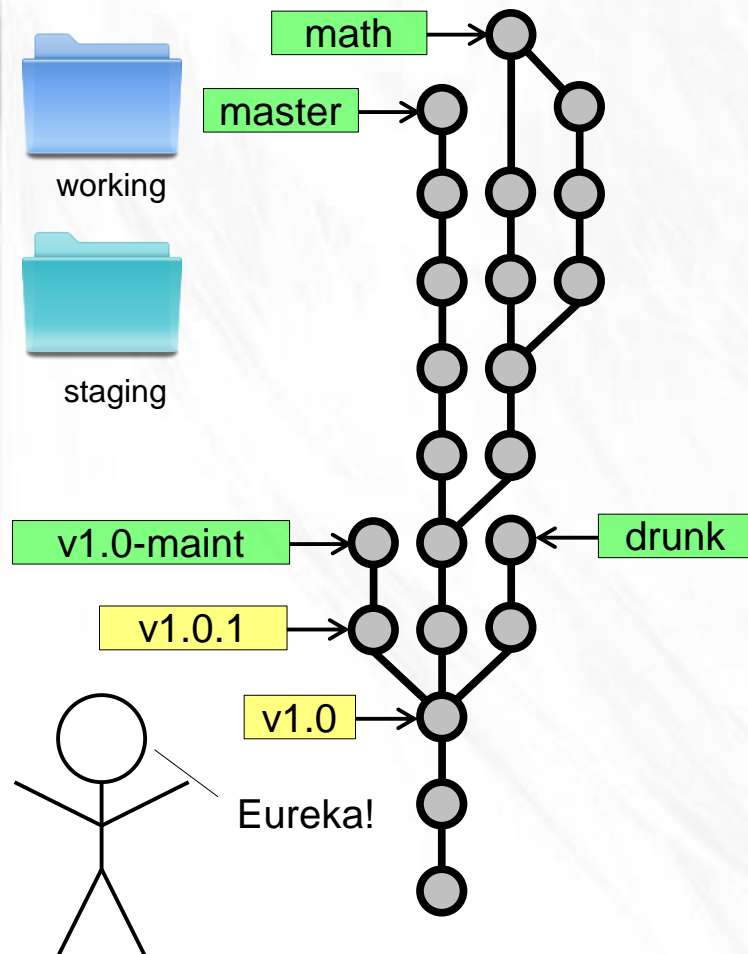
# Diffs



# Eliminating Duplication



# Eliminating Duplication



SHA1

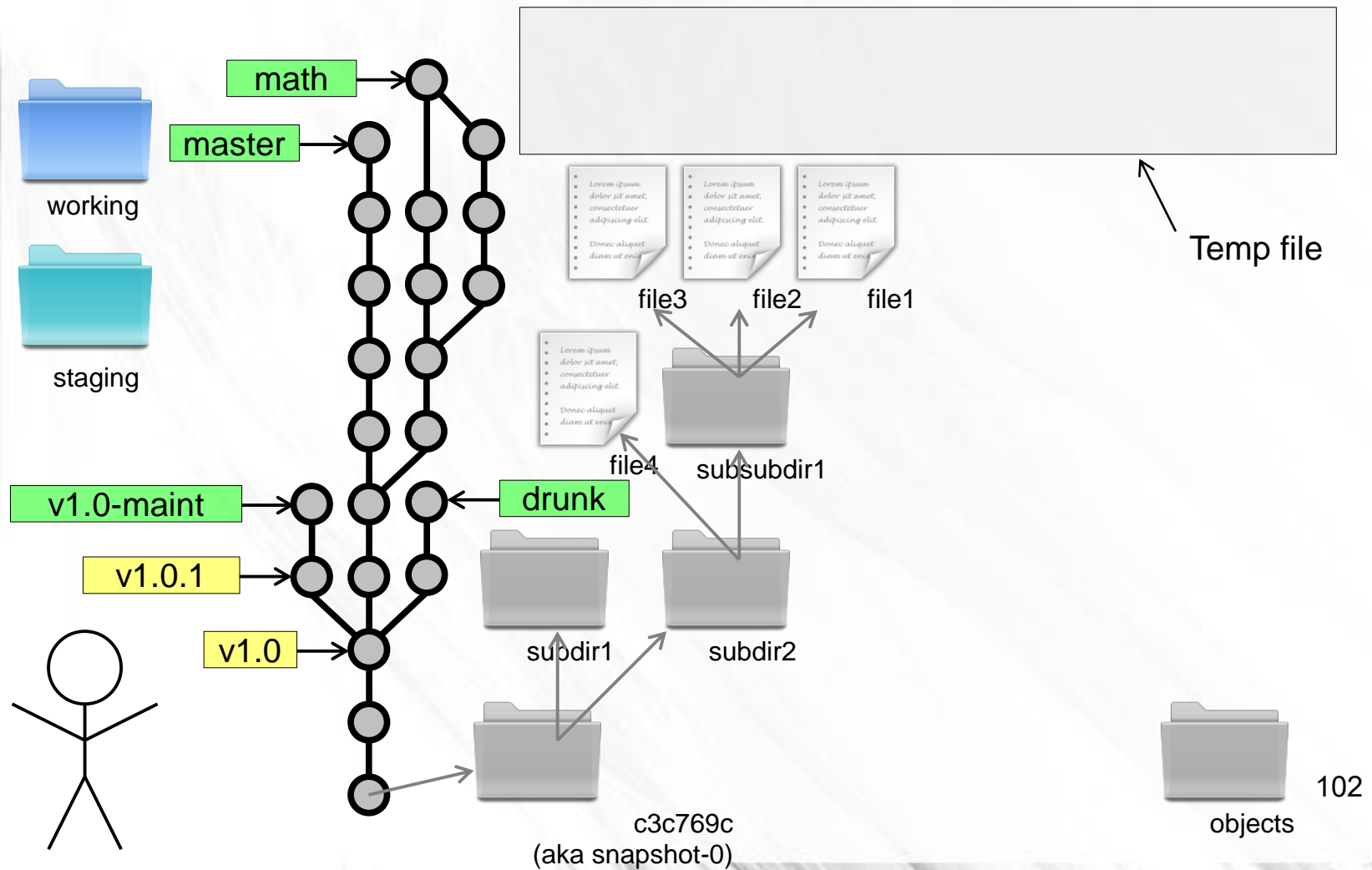
2804133755c3ed396d162028c7b30a1cbcfecded



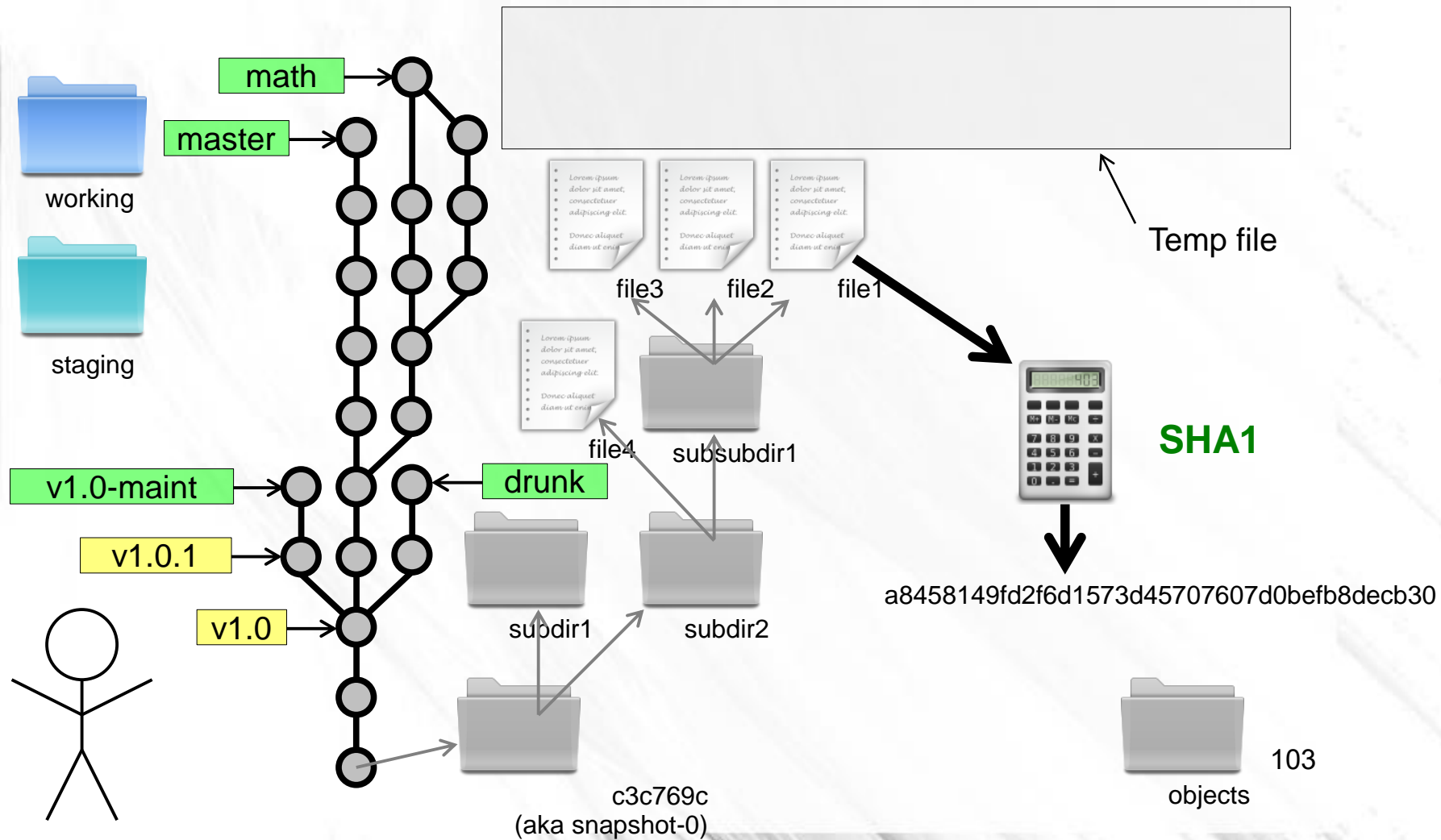
101

objects

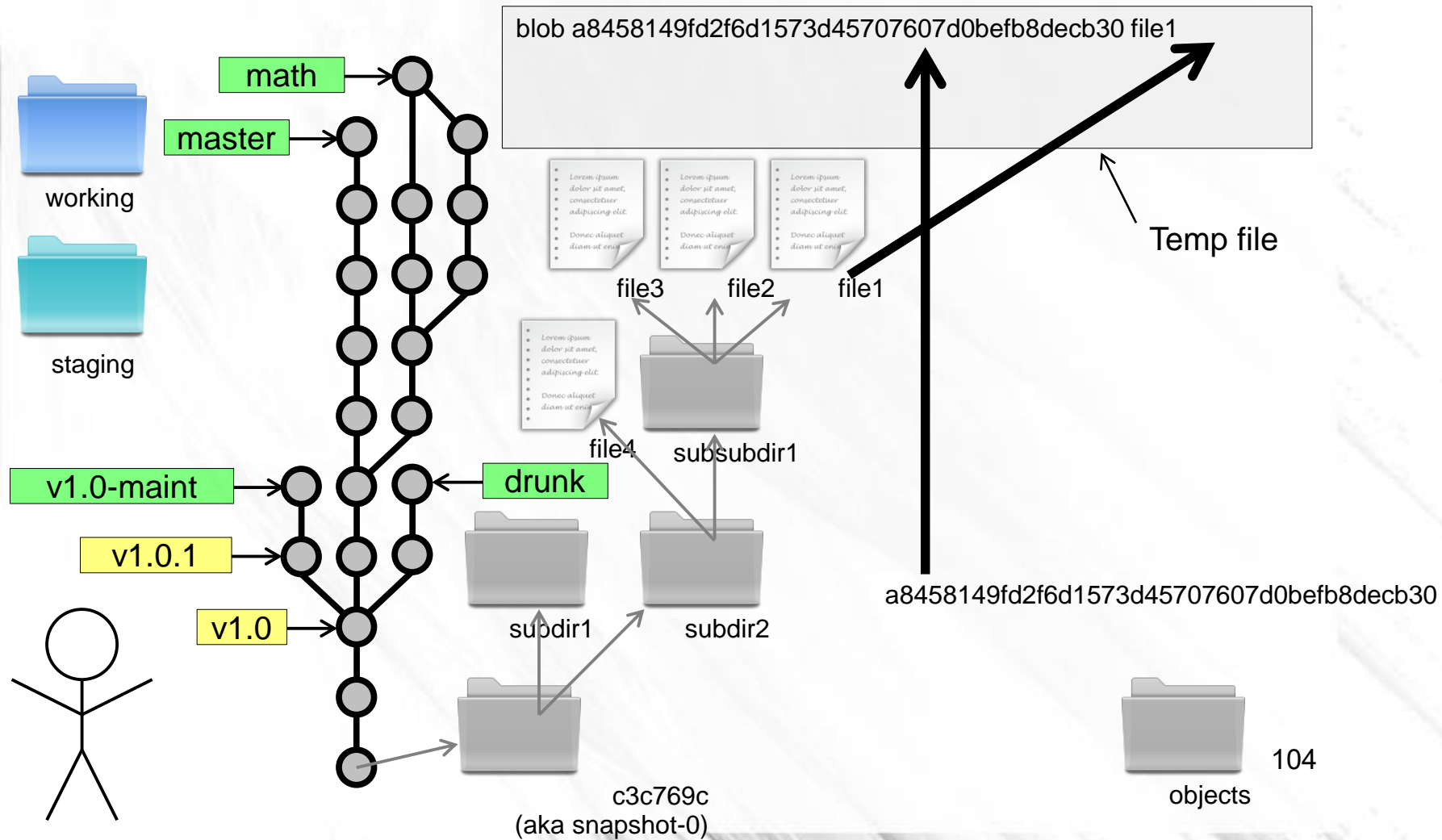
# Eliminating Duplication



# Eliminating Duplication

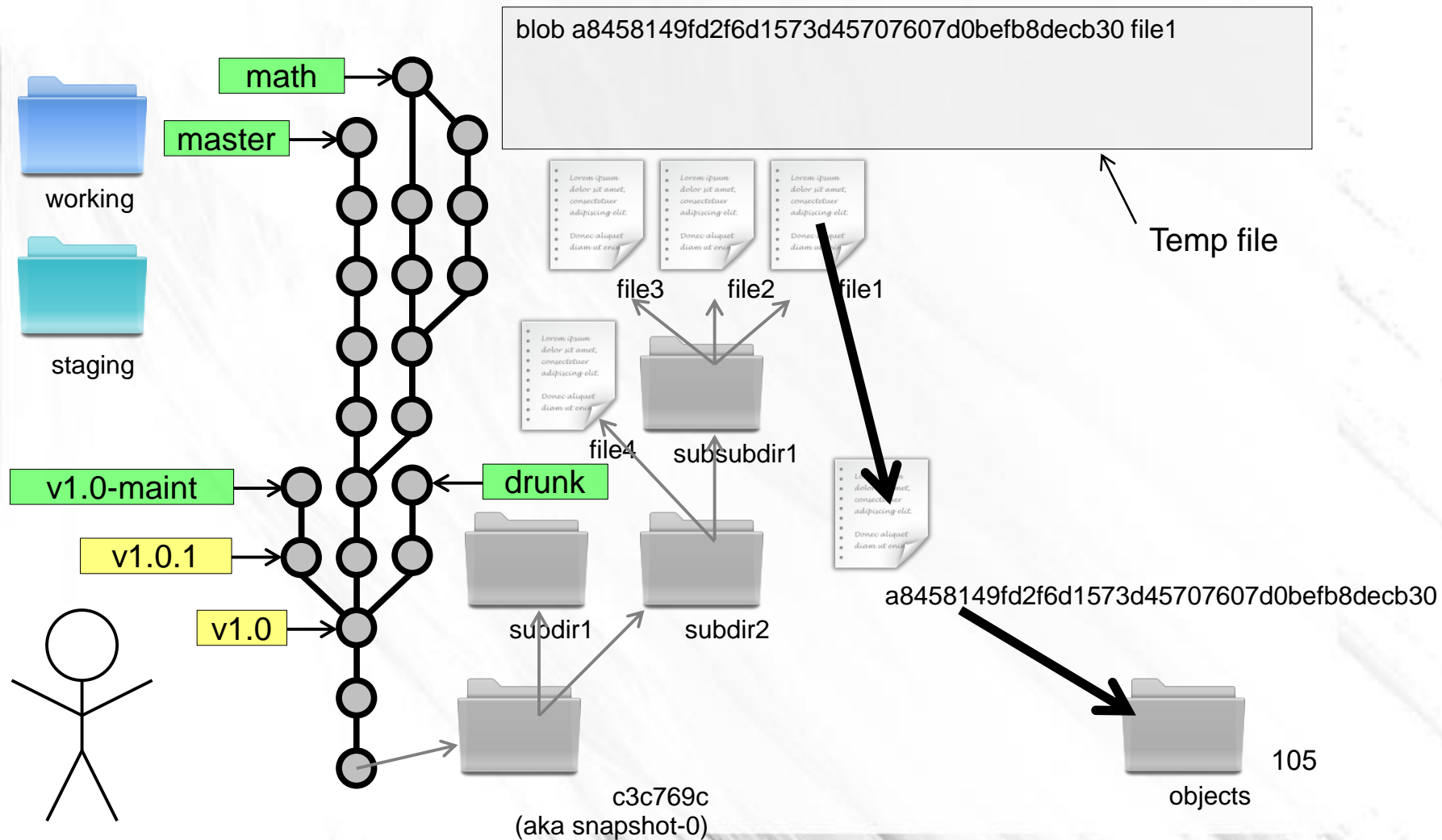


# Eliminating Duplication

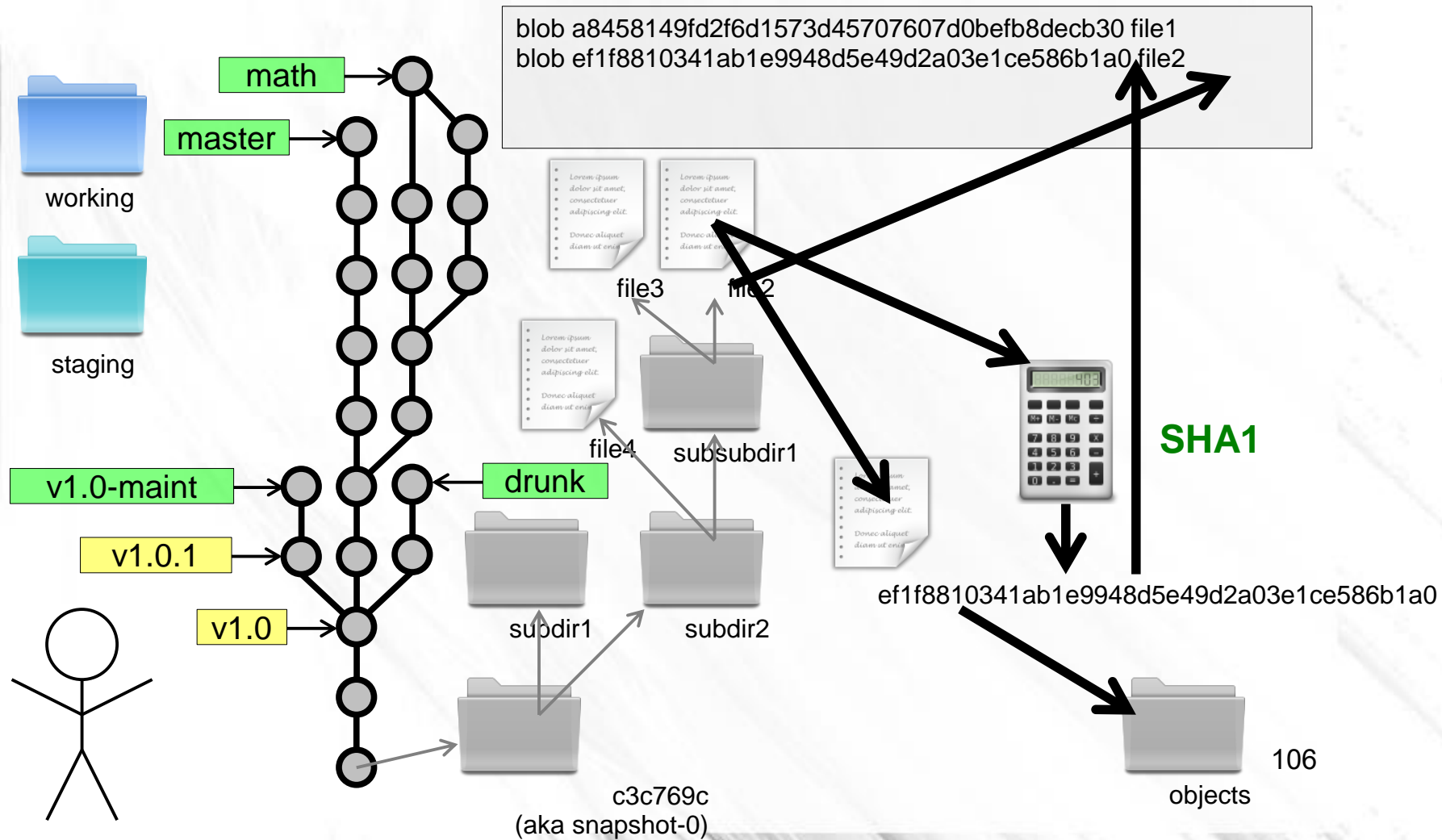




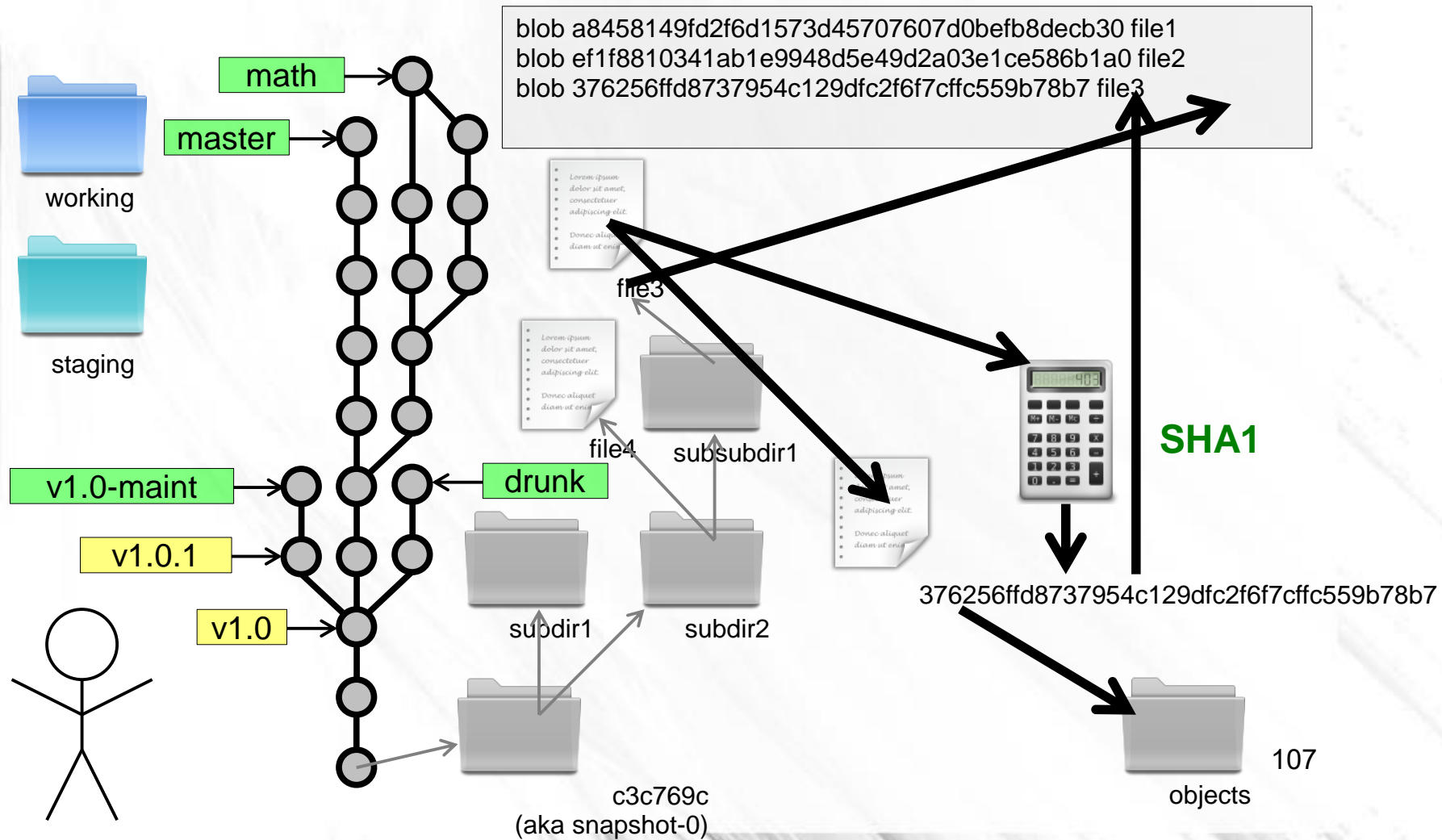
# Eliminating Duplication



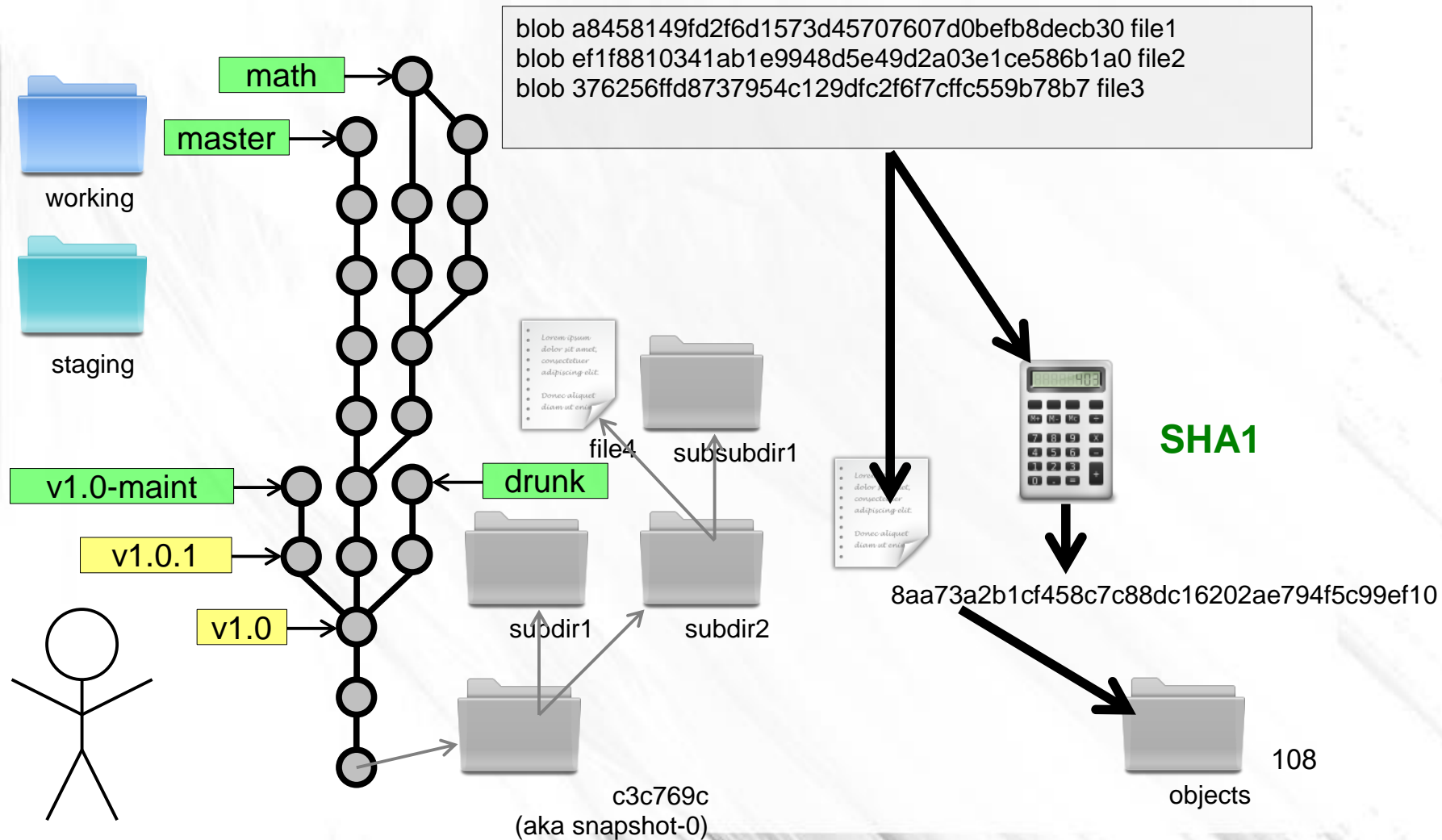
# Eliminating Duplication



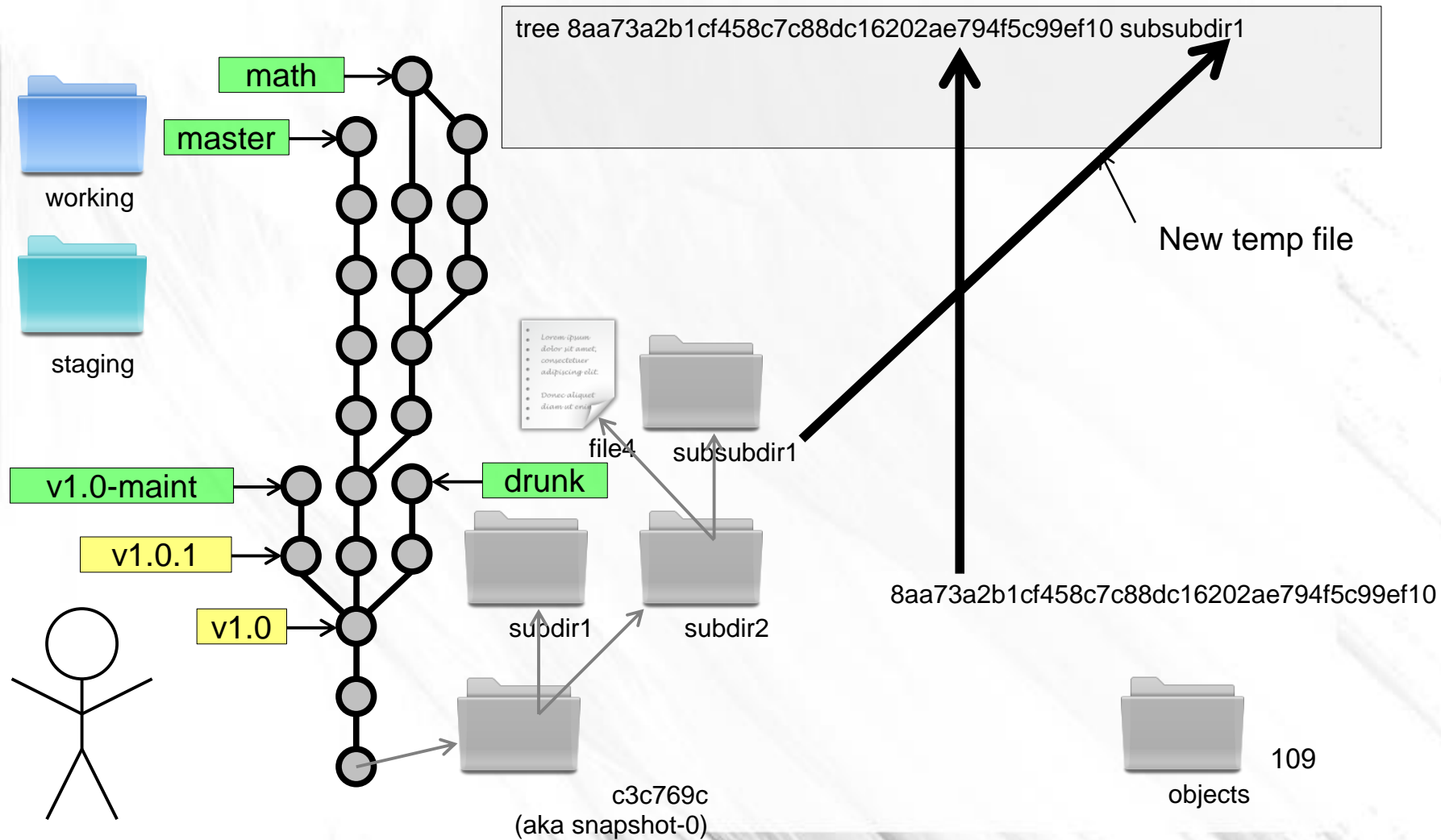
# Eliminating Duplication



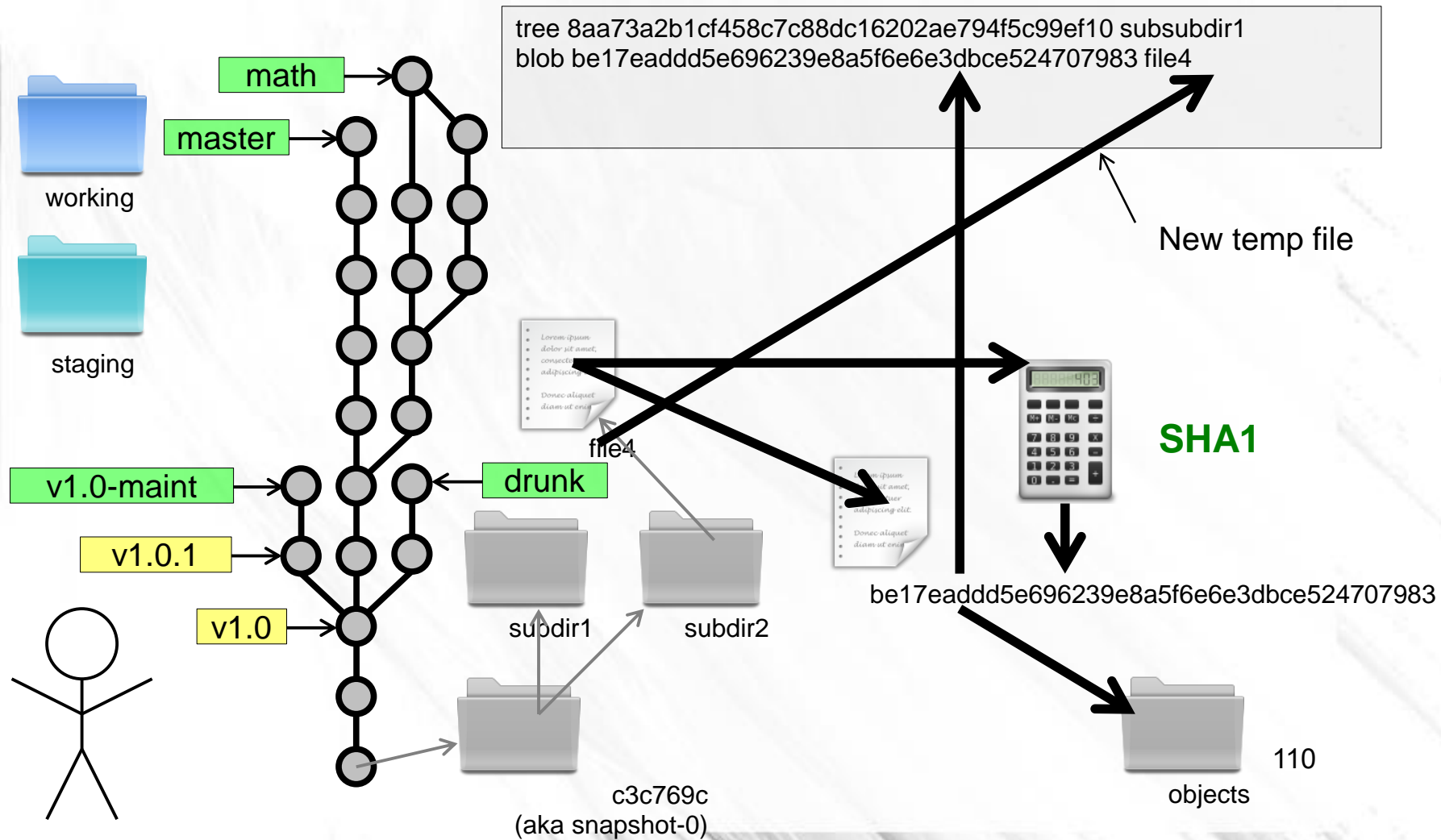
# Eliminating Duplication



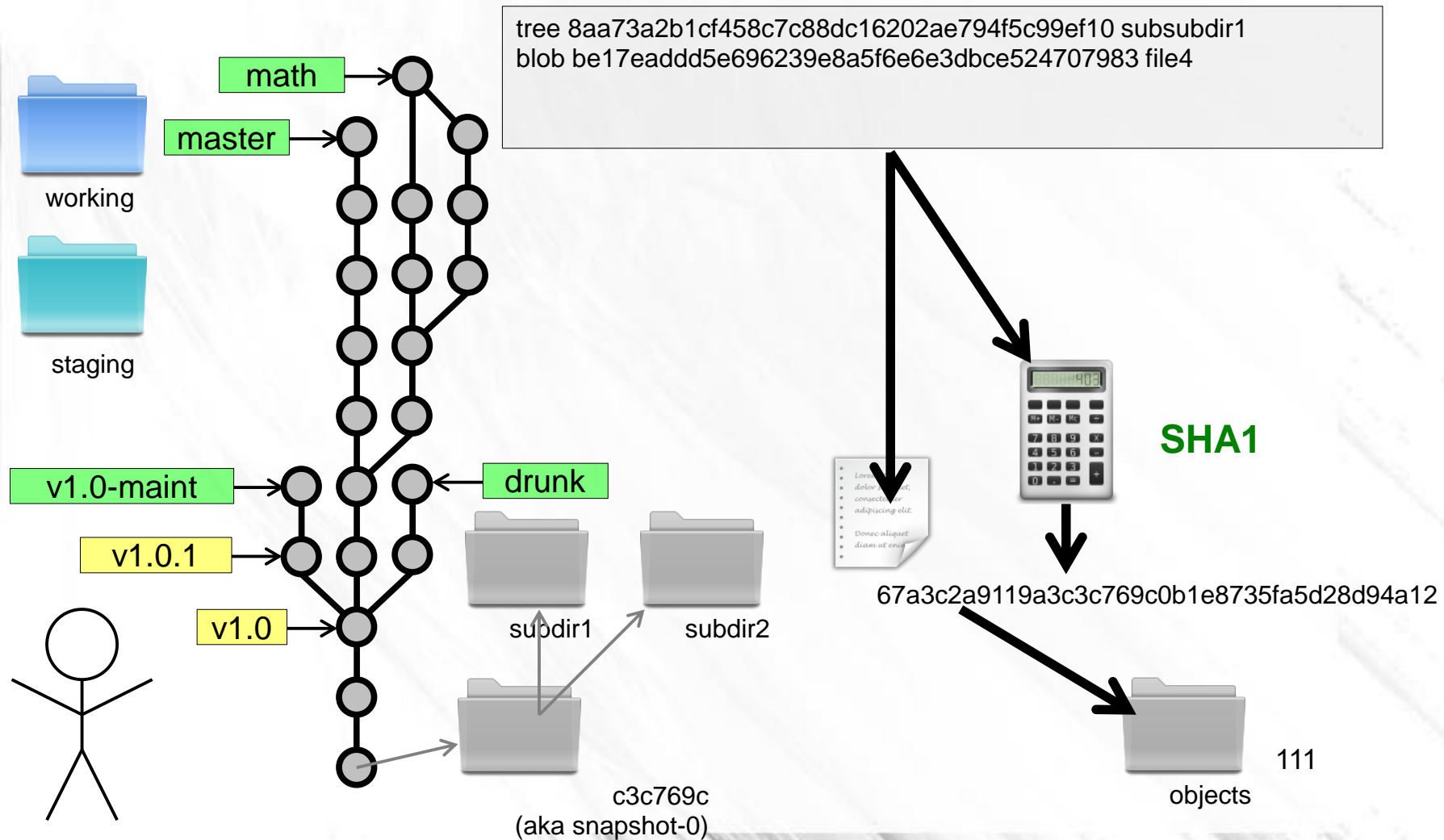
# Eliminating Duplication



# Eliminating Duplication

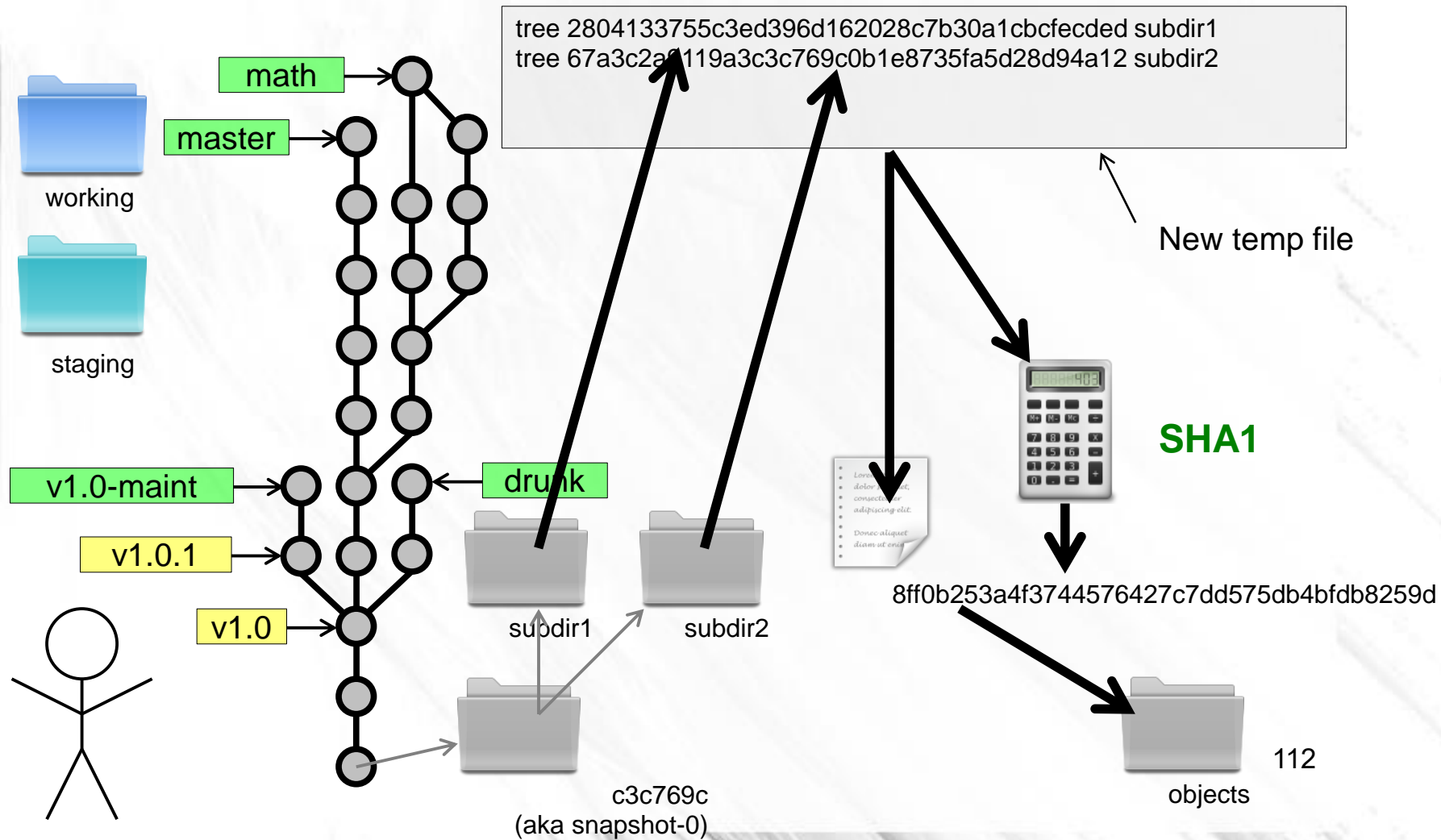


# Eliminating Duplication



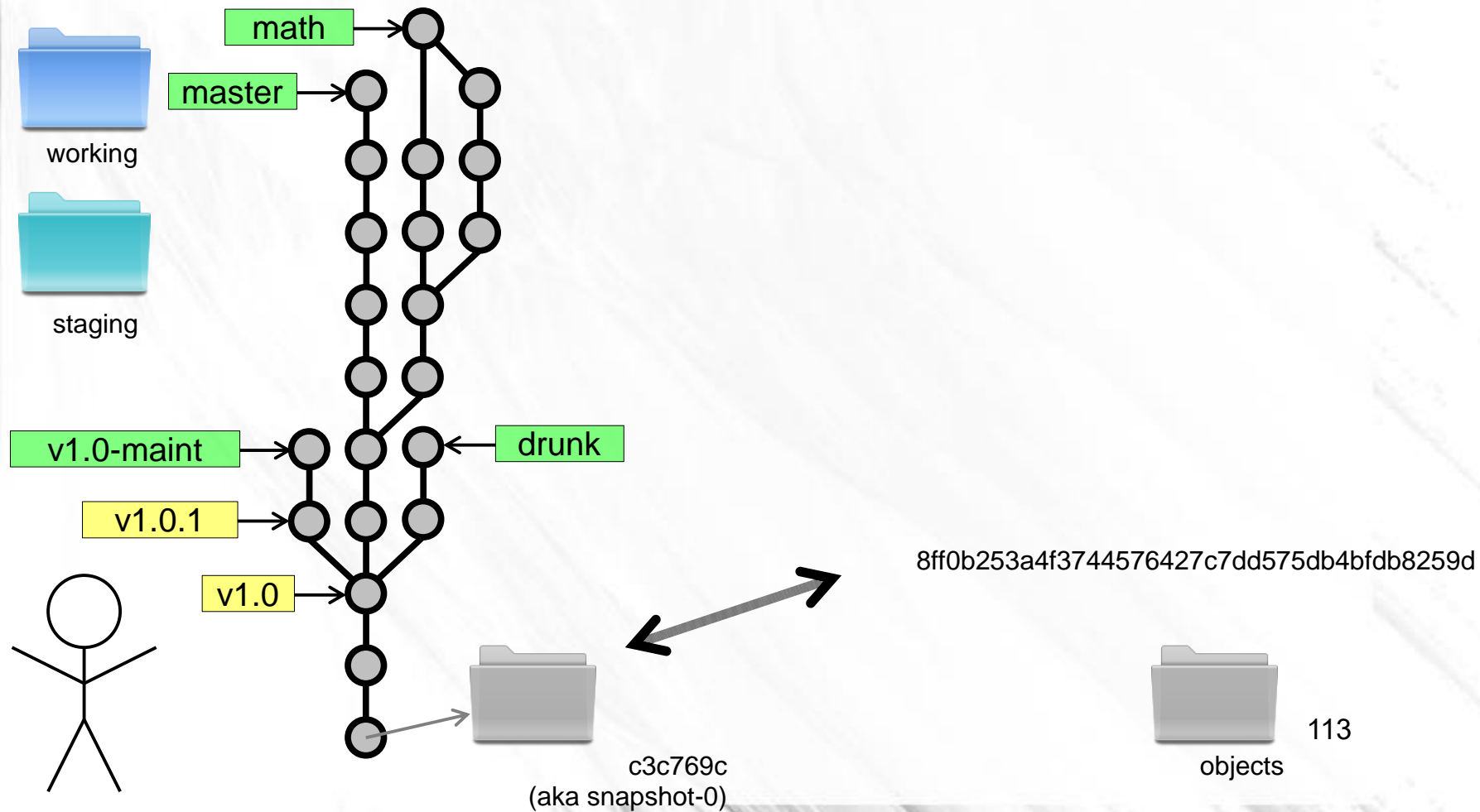


# Eliminating Duplication

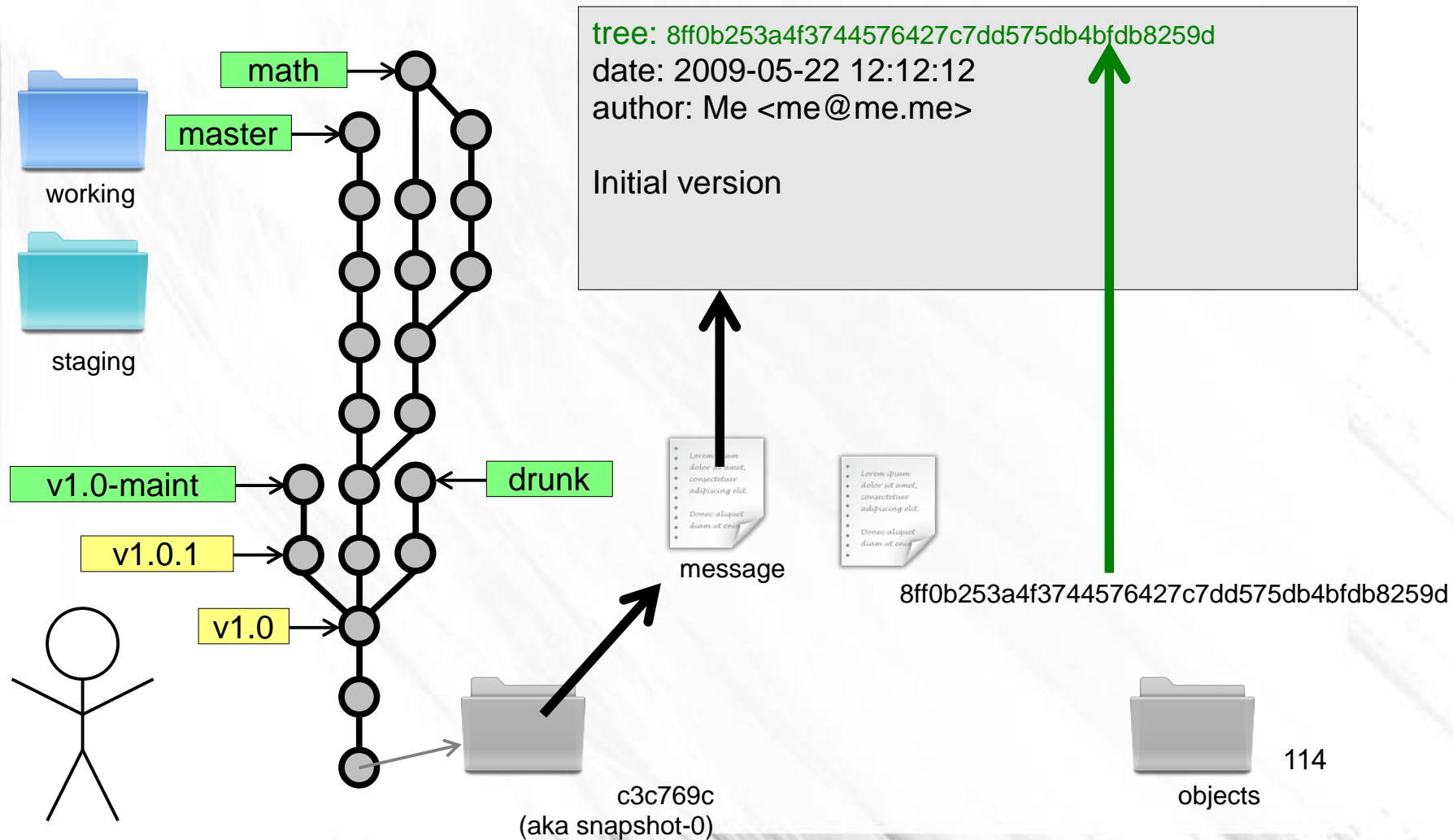




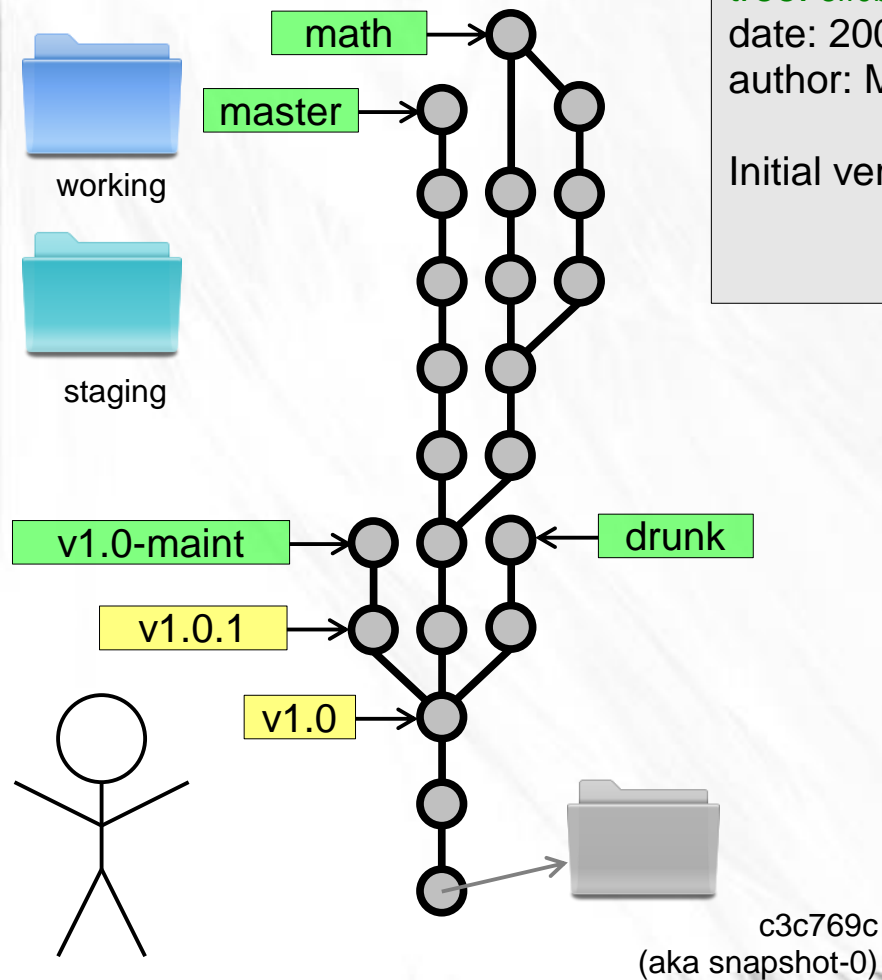
# Eliminating Duplication



# Eliminating Duplication



# Eliminating Duplication

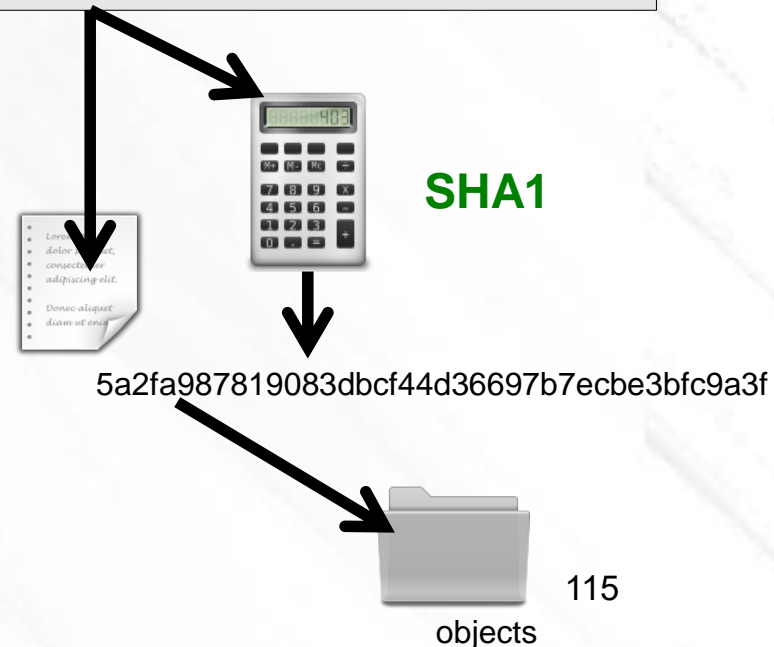


tree: 8ff0b253a4f3744576427c7dd575db4bfdb8259d

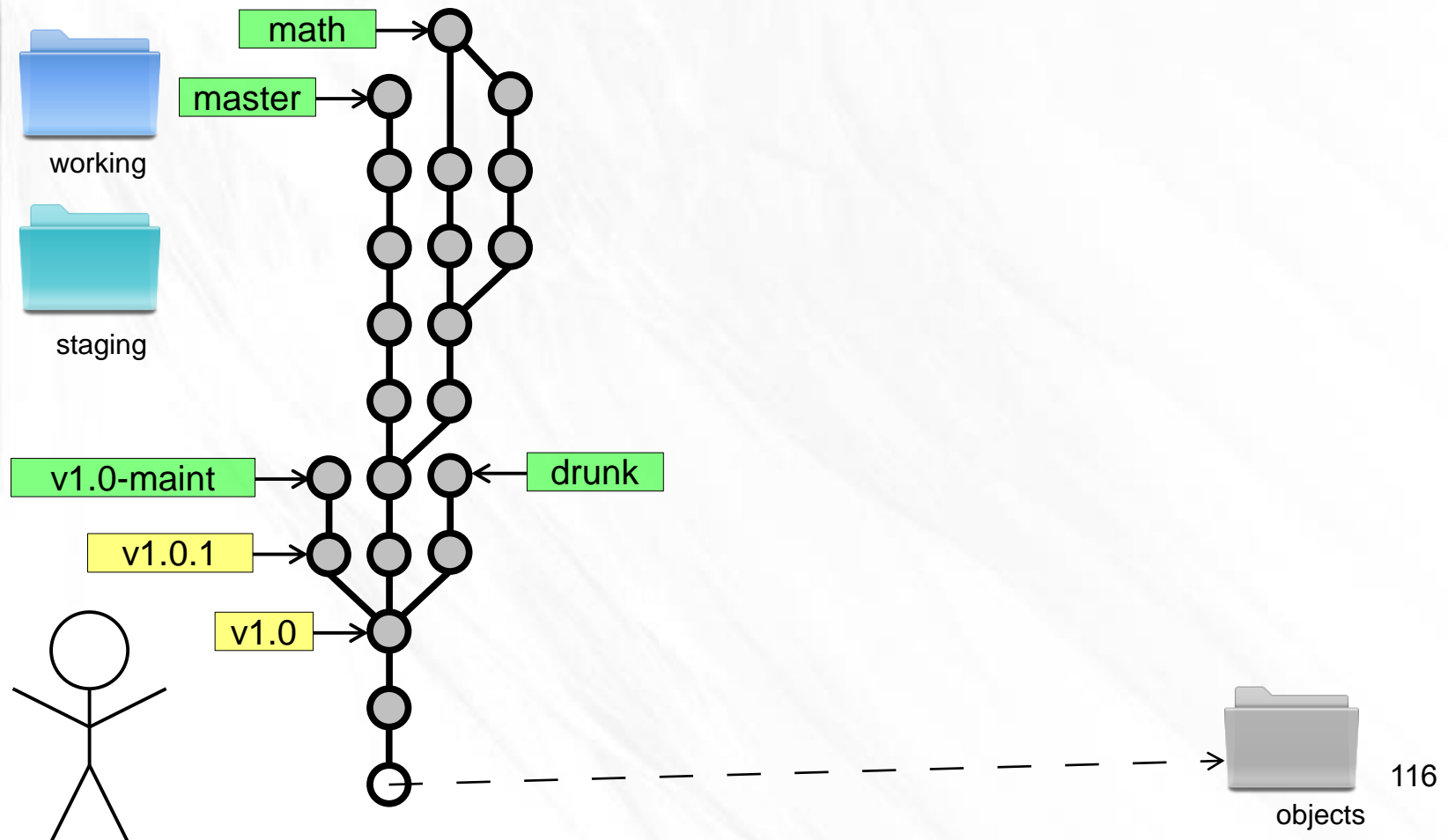
date: 2009-05-22 12:12:12

author: Me <me@me.me>

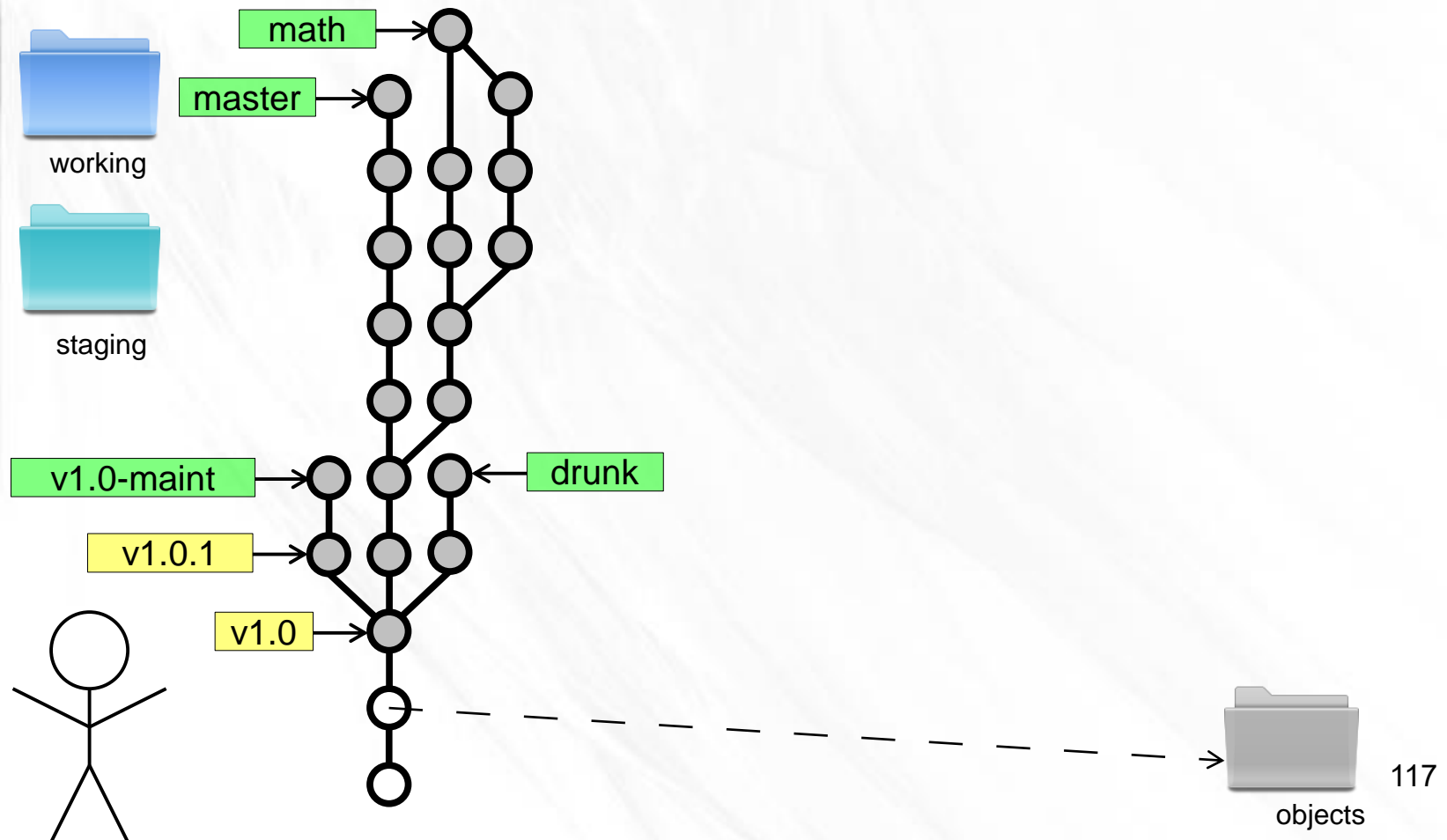
Initial version



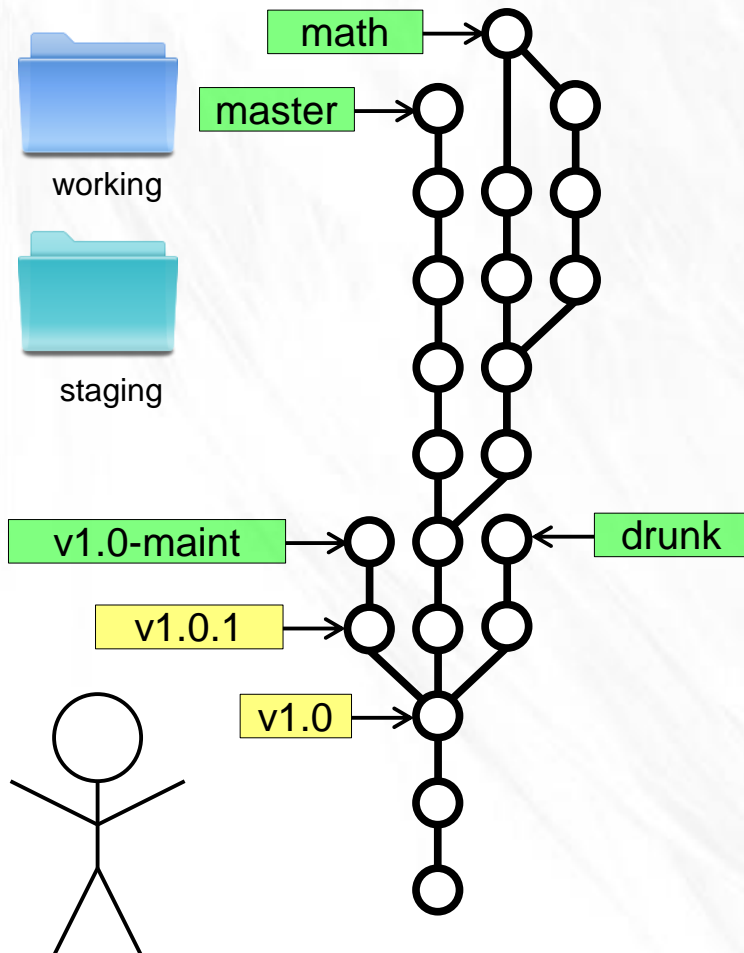
# Eliminating Duplication



# Eliminating Duplication

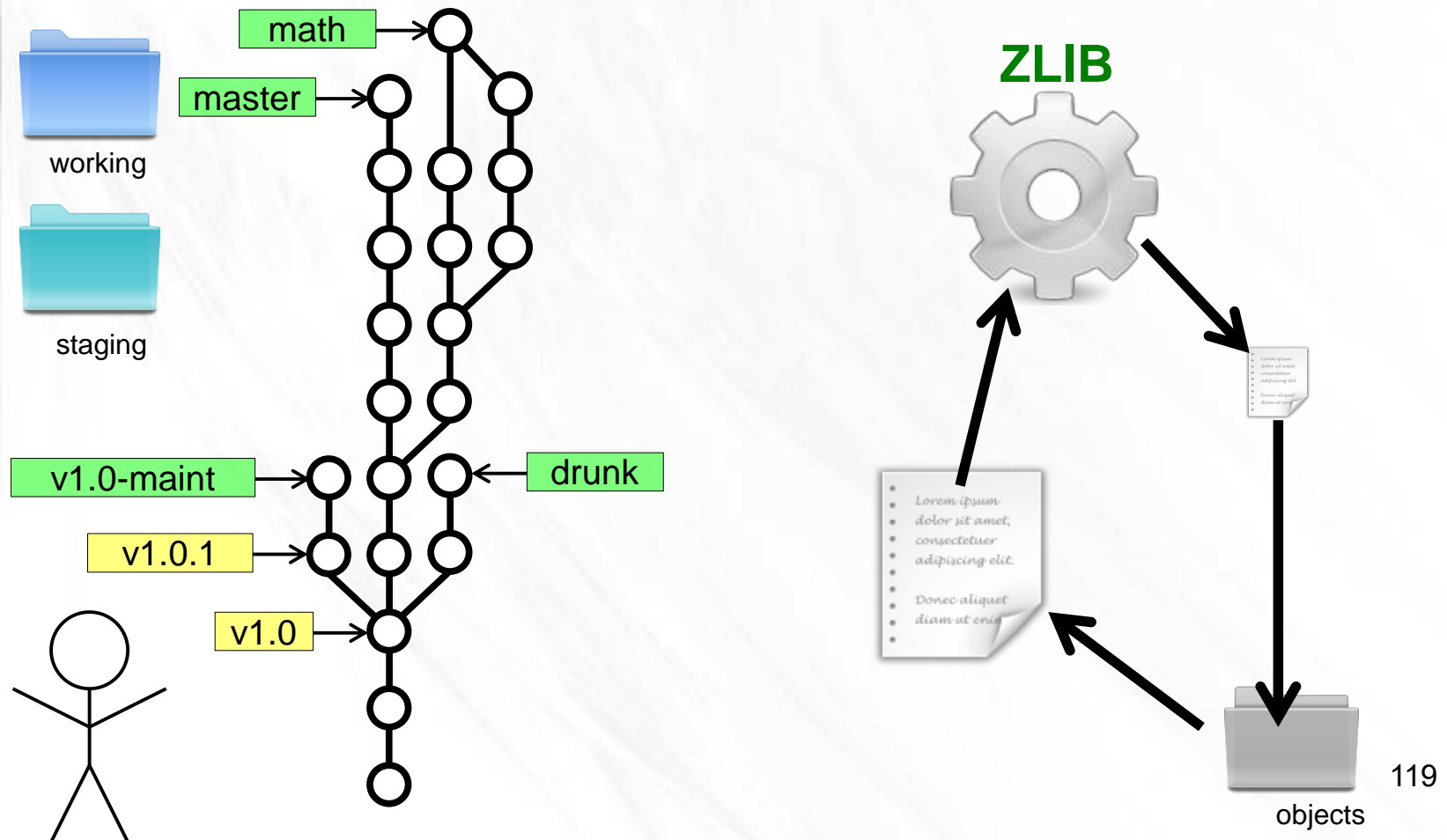


# Eliminating Duplication

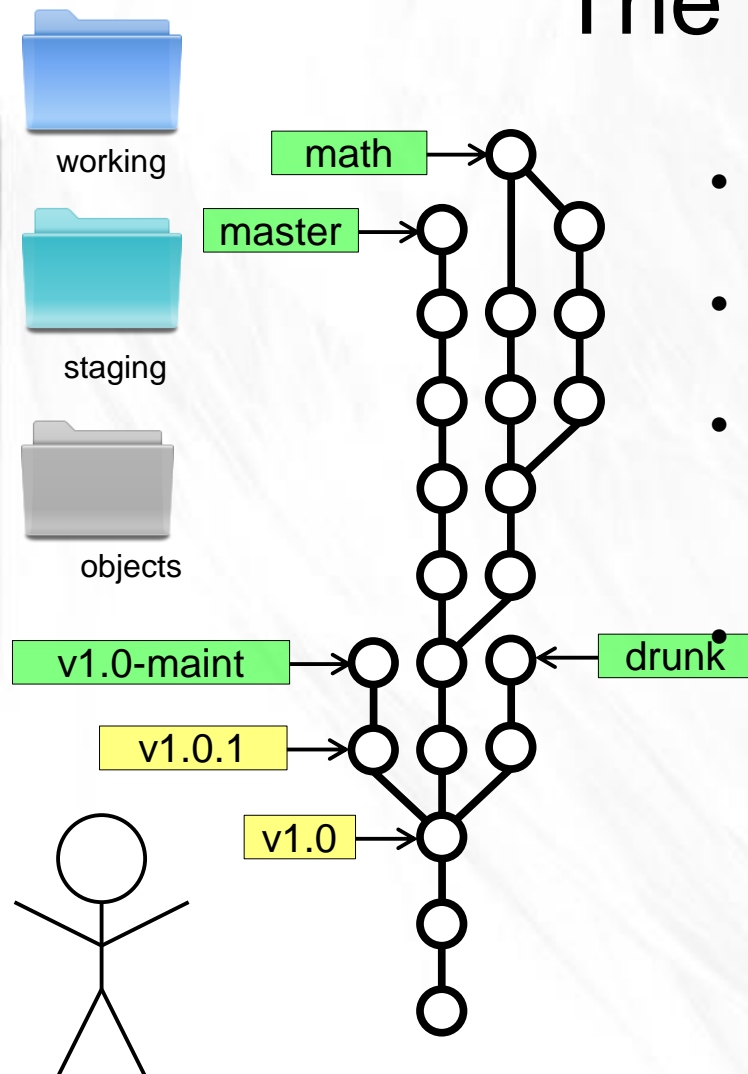


objects

# Compressing Blobs



# The True Git



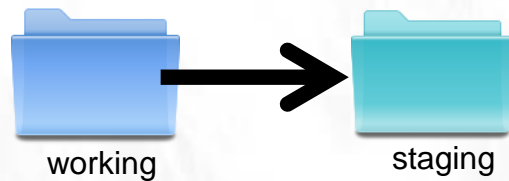
- TADAA!
- This is pretty much Git
- Nicer command line tools for all these operations
- Many, many other tools



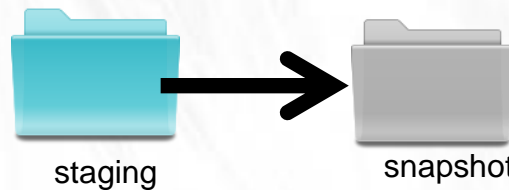
# Commands: Getting Started

- First, tell Git who you are:
  - `git config --global user.name "My Name"`
  - `git config --global user.email "my@email.address"`
- Get help:
  - `git <command> -h`
  - `git help <command>`
- Start a new Git repository:
  - `git init`

# Commands: Making snapshots



• git add



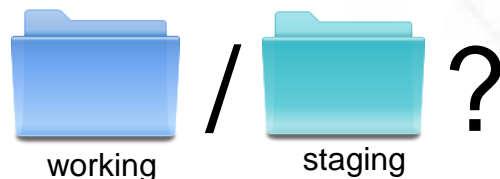
• git commit



git commit -a



• git log



• git status

gitk

```
graph TD
    A[Add the simple scripts I used to do a me] --> B[Merge the new object model thing from]
    B --> C[PATCH] Switch implementations of merg
    C --> D[PATCH] Port fsck-cache to use parsing fi
    D --> E[PATCH] Port rev-tree to parsing function
    E --> F[PATCH] Implementations of parsing fund
    F --> G[PATCH] Header files for object parsing
    G --> H[PATCH] fix bug in read-cache.c which lo
    H --> I[PATCH] Fix confusing behaviour of upda
    I --> J[Make "commit-tree" check the input obje]
    J --> K[Make "parse_commit" return the "str122]
    K --> L[Do a very simple "merge-base" that finds]
    L --> M[Make "rev-tree.c" use the new-and-impro]
```

# Commands: Diffing



vs.



- `git diff`



vs.



- `git diff --staged`



vs.



- `git diff HEAD`



vs.



- `git diff <from> <to>`

# Commands: Branches & Tags

- git branch
  - git branch <branch>
  - git checkout <branch>
  - git tag -l
  - git tag <tag>
- } git checkout -b ...

# Commands: Fetching & Merging

- `git remote add <name> <URL>`

- `git fetch <name>`

}

`git pull`

- `git merge <name>/<branch>`

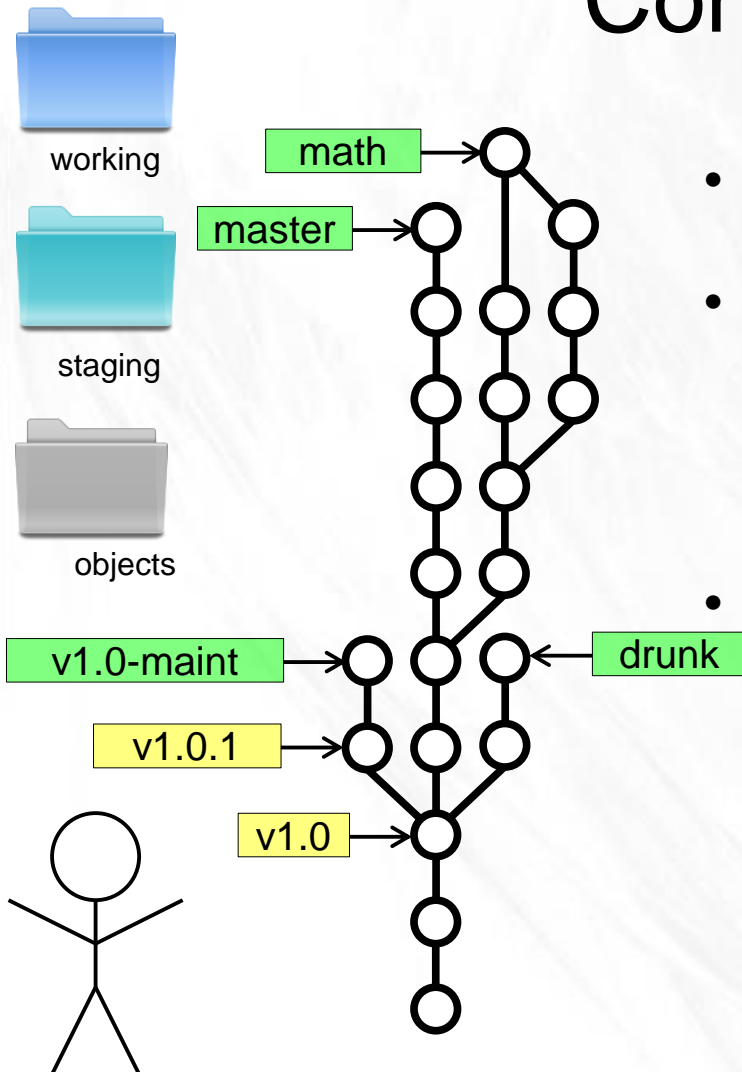
- Sync back: `git push`

# Conclusion

- Keep this parable in mind
- Git is simple and powerful

- One more thing:

git reflog



# Where to go next?

- Git homepage: <http://git-scm.com>
- Pro Git: <http://git-scm.com/book>
- Git Reference: <http://gitref.org>
- GitHub: <http://github.com>
- Gitorious: <http://gitorious.org>
- **Gitlab**

# Questions?

- Thanks for your attention!
- These slides are available at:  
[https://github.com/jherland/git\\_parable](https://github.com/jherland/git_parable)
- Reach me at <[johan@herland.net](mailto:johan@herland.net)>





# Git-init

- <http://pel-daniel.github.io/git-init/>

# Git Clients (Windows)

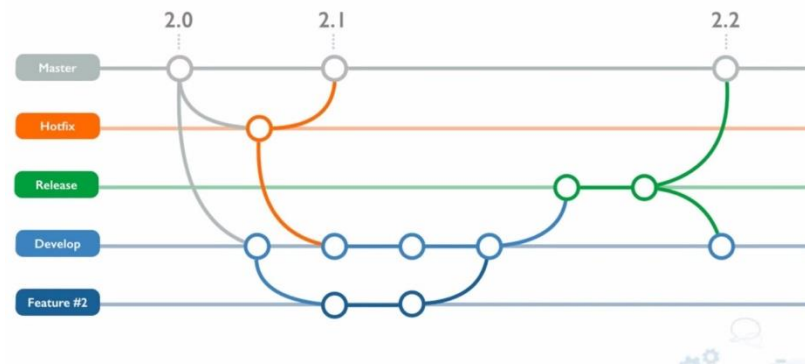
- CLI Shell: Git Bash
- Git Extensions: CLI +Windows Explorer Shell
- Github Desktop (+powershell)
- Bitbucket SourceTree
- IDE Integration
  - Visual Studio (Code / VS2013+ native)
  - Eclipse Egit
  - IntelliJ, Webstorm embedded
  - Brackets plugin
  - VS Code

# הדגמה \ טיפים

- <http://try.github.com>
- <http://learn.github.com/> (education.. - free student account)
- <http://help.github.com/create-a-repo/>  
Local user settings:  
git config user.name <user>  
git config user.email [user@example.com](mailto:user@example.com)
- git pull
- git add: add / stage
- git commit -a == add+commit

# Git Workflows - שיטות

- Centralized Workflow
- Feature Branch Workflow
- Gitflow Workflow
- Forking Workflow (OSS)



Github Flow •

Pull request •

• דוגמא: תהליך העבודה בפרויקט Nuget:

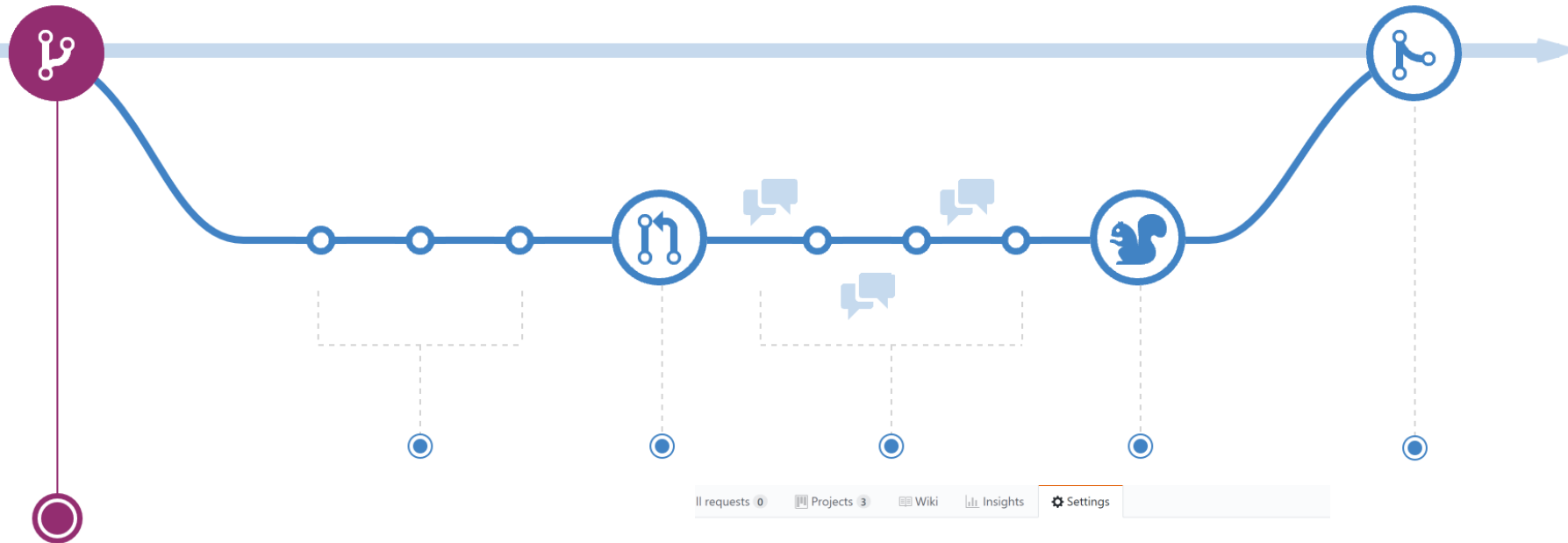
Contributing a Bug Fix or Feature

• קישורים נוספים בויקי



Git Workflow \_ Atlassian Git Tutorial.pdf

# GH Branch Protection



11 requests 0 Projects 3 Wiki Insights Settings

## Branch protection for master

### ☒ Protect this branch

Disables force-pushes to this branch and prevents it from being deleted.

#### ☐ Require pull request reviews before merging

When enabled, all commits must be made to a non-protected branch and submitted via a pull request with at least one approved review and no changes requested before it can be merged into **master**.

#### ☐ Require status checks to pass before merging

Choose which [status checks](#) must pass before branches can be merged into **master**. When enabled, commits must first be pushed to another branch, then merged or pushed directly to **master** after status checks have passed.

#### ☐ Restrict who can push to this branch

Specify people or teams allowed to push to this branch. Required status checks will still prevent these people from merging if the checks fail.

#### ☐ Include administrators

Enforce all configured restrictions for administrators.

Save changes

# בפעם הבאה

- פרויקט: פיתוח בסבבים
- משימה אישית מס' 3 – ליווי גם בהרצאה
- בקרת גרסאות II – תרחישים נוספים עם git -  
Workflows
- שעה שלישית – מעבדה (משימה אישית 3)
- בהמשך
  - בדיקות, פיתוח מונחה מפרטים
  - תחזוקה ועוד

# לסיכום

- תהליך: בקרת תצורה וגרסאות
- כלים: git / github
- שיטות: למשל git flow