



# הנדסת תוכנה

## 6. בקרת גרסאות

### git / github

[Pragmatic Programmer Tip](#) :

**Always Use Source Code Control**

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

# מה היום?

- בקרת גרסאות קוד – Version Control
- Git

– משימה אישית 3

- Github

- סקר ZFR

- פרויקט: תחילת סבבים

- תרגיל – סקר ZFR  
(+ השלמת back-end)



# 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

- Jim Weirich, Source Control for People Who Don't Like Source Control ([slides](#) – similar to the git parable)
- 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

- <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](http://progit.org) (book, [basic](#)),
- [Windows client list](#), [Egit](#) (Eclipse), [VS](#),  
[power-shell](#), [github for windows](#) ([slds](#))
- 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)**



# Github

- ויקי
- ~~הורדות~~, שחרורים
- חברתי \ גרפים, תרומות
- **ניהול משימות**
- ✓ **בקרת גרסאות**
- סקרי קוד (בהמשך)
- קישור לאתרי **רזומה**
- חינם לקוד פתוח

**CAREERS 2.0**  
by stackoverflow



+



Have projects on GitHub?  
Import them easily to your profile



# 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
  - Revision Control
  - Software Configuration Management
  - Source-Code/**Version Control System**

# Joel Test (~2000 / stackoverflow)

## 1. Do you use source control?

וגם 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

- שיתוף מספר מפתחים (מרוחקים) בו זמנית
- לדאוג שלא יהיו סתירות בין המפתחים
- איסוף כל הגרסאות ומעקב אחרי שינויים



– מאפשר מחיקת קוד

- ניהול מספר גרסאות במקביל
- גיבוי

- מאגר מעודכן של תוצרי הפרויקט

– במיוחד עם daily build \Continuous Integration

בפרויקט תדרשו להדגים את בקרת  
התצורה שלכם

# 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

# 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\\_parable](https://github.com/jherland/git_parable)

# 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

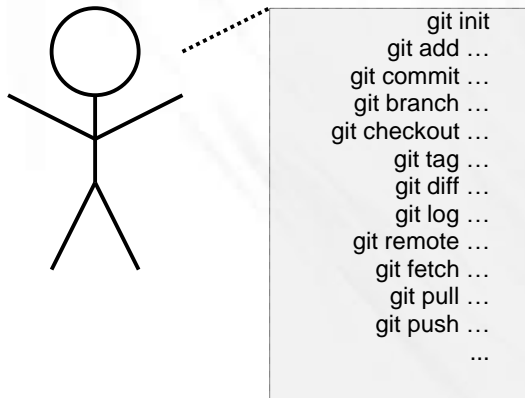
# The Git Parable

- Git - simple & powerful



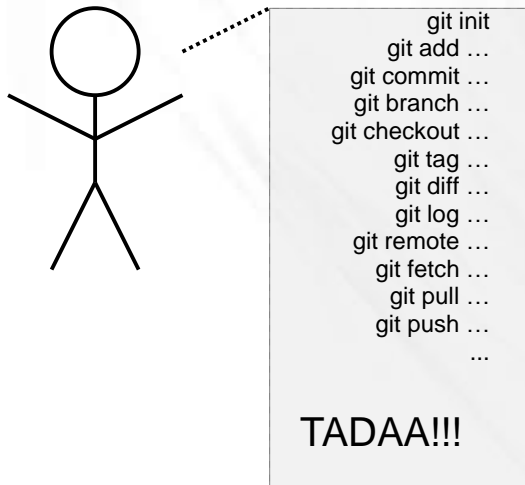
# The Git Parable

- Git - simple & powerful



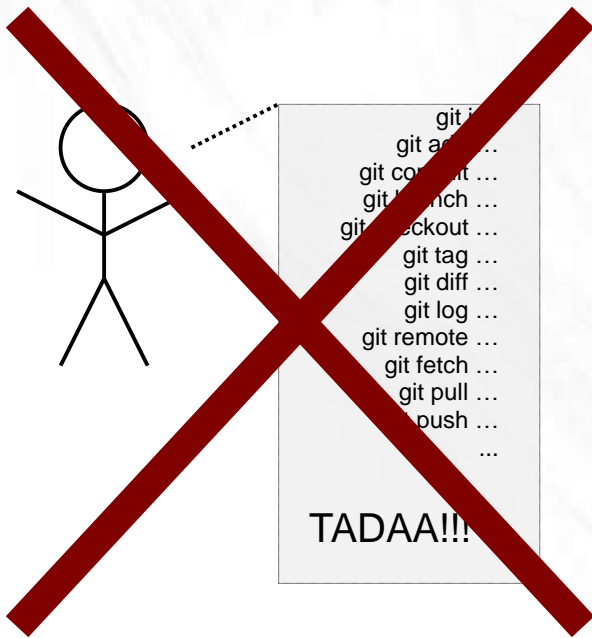
# The Git Parable

- Git - simple & powerful



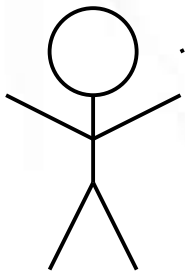
# The Git Parable

- Git - simple & powerful



# The Git Parable

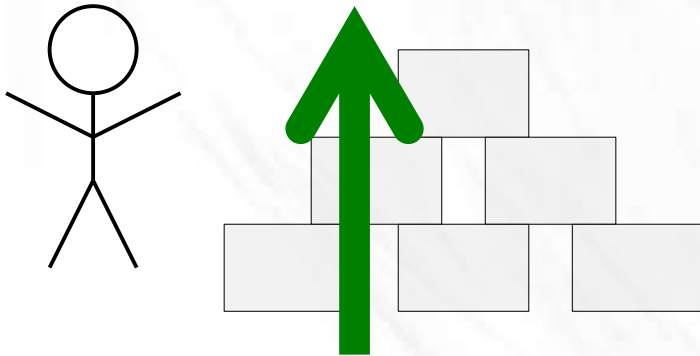
- Git - simple & powerful



Let's tell a parable

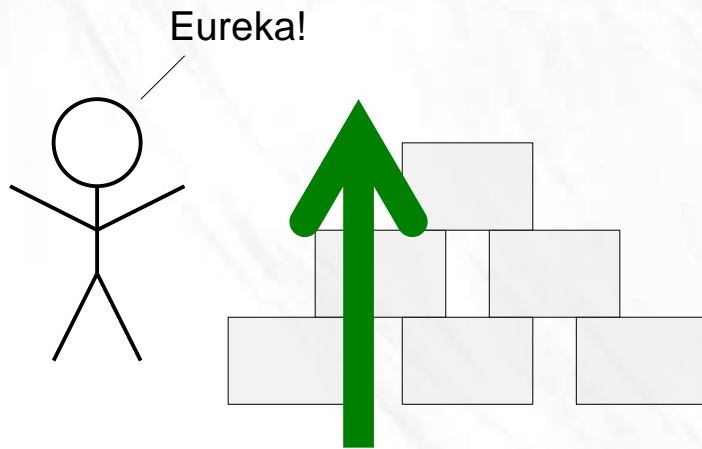
# The Git Parable

- Git - simple & powerful



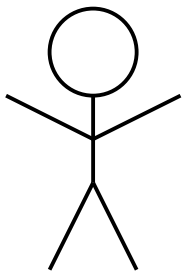
# The Git Parable

- Git - simple & powerful



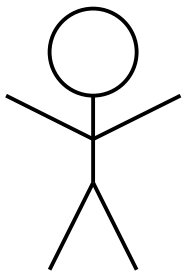
# The Parable

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



# The Parable

- Write a large software program

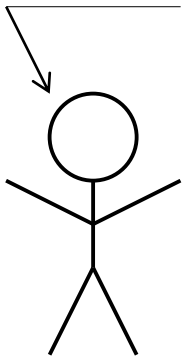




# The Parable

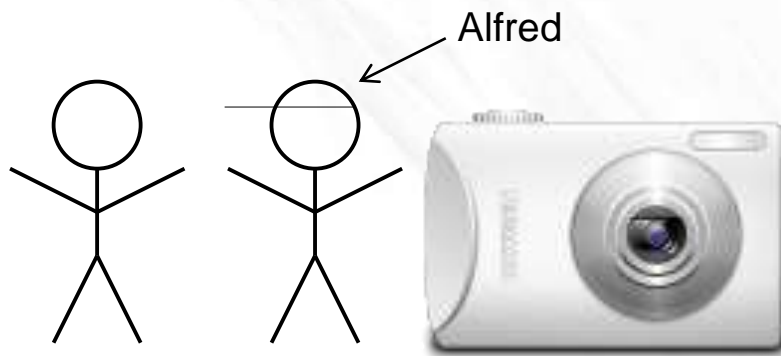
- Write a large software program
- Invent some method to keep track of versions
  - retrieve code that you changed/deleted

Responsible!



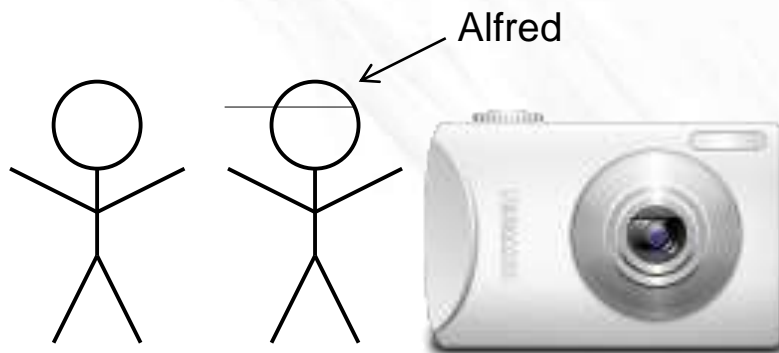
# Snapshots

- Alfred, the photographer



# Snapshots

- Alfred, the photographer



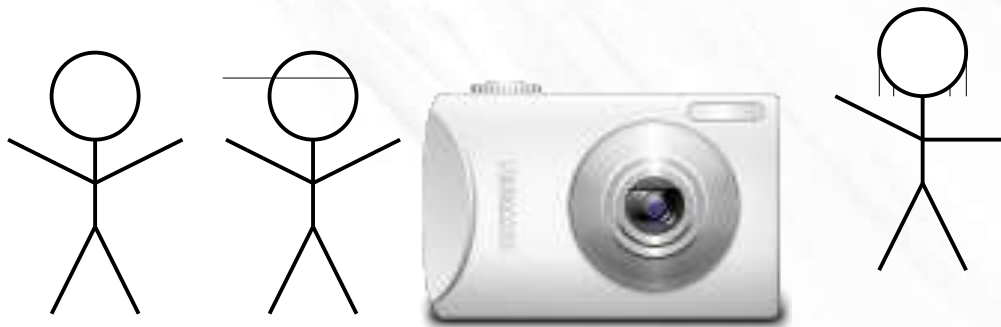
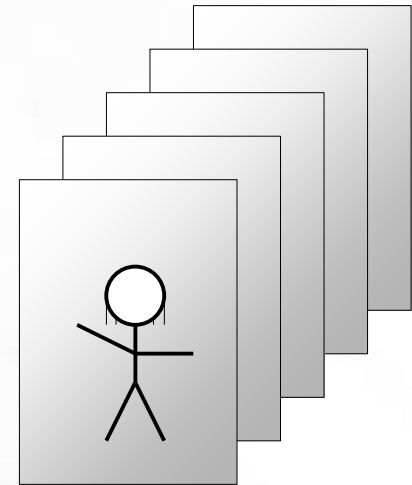
# Snapshots

- Alfred, the photographer
- Hazel and her daughter



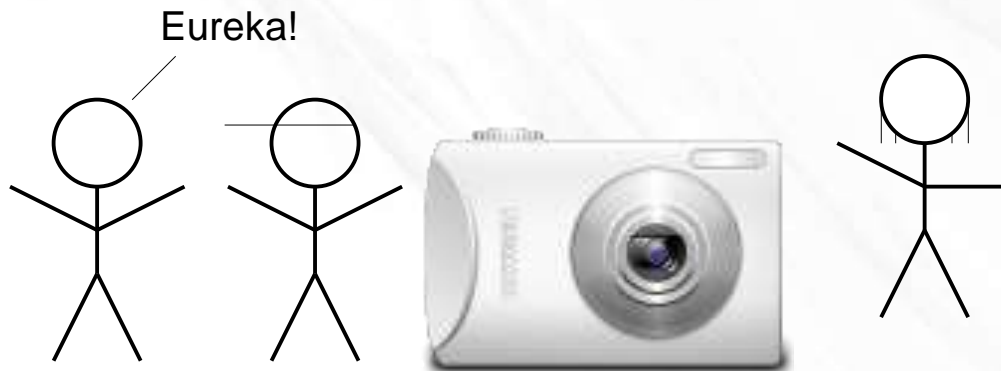
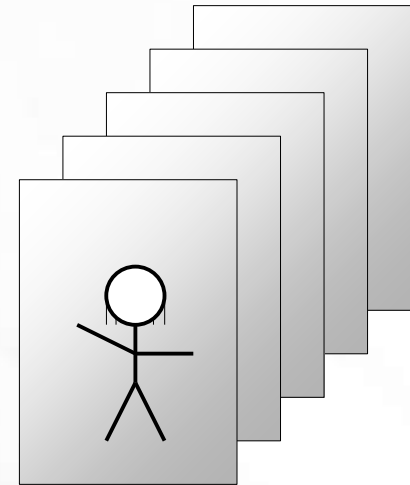
# Snapshots

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

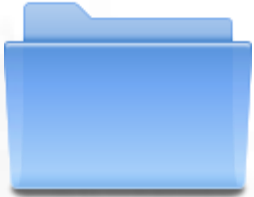


# Snapshots

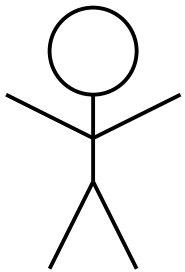
- Alfred, the photographer
- Hazel and her daughter
  - Remember what the daughter was like at each different stage



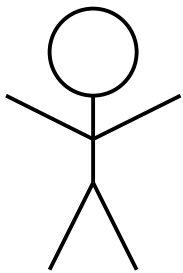
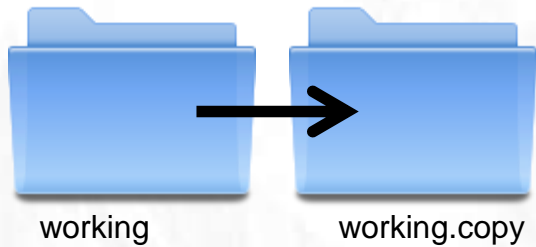
# Snapshots



working

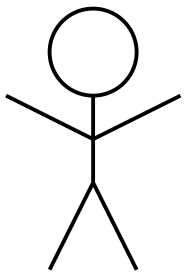


# Snapshots

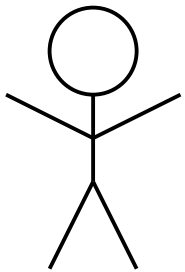
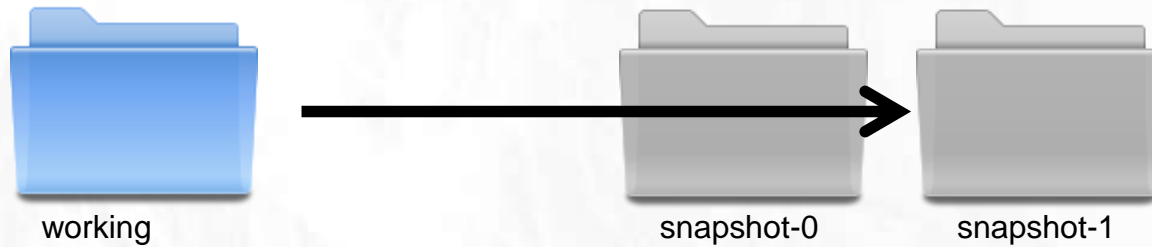




# Snapshots



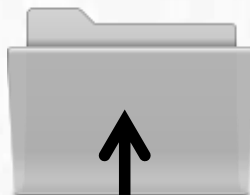
# Snapshots



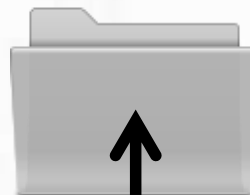
# Snapshots



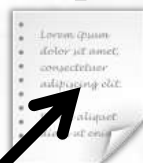
working



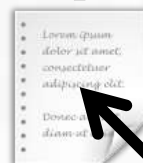
snapshot-0



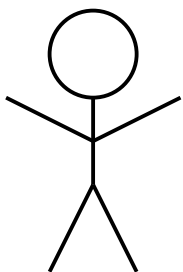
snapshot-1



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

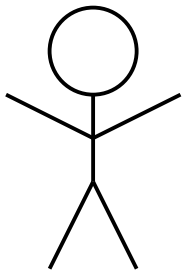


snapshot-1

...



snapshot-99



# Branches



working



snapshot-0



snapshot-1

...

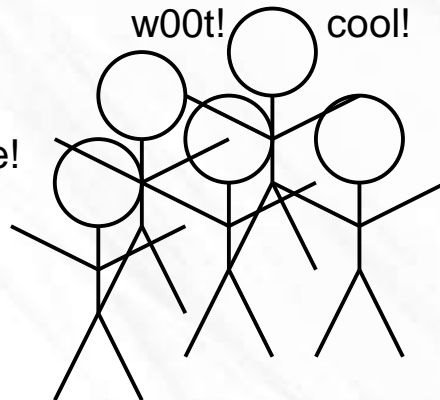
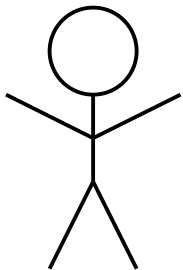


snapshot-99



w00t! cool!

nice!



# Branches



working



snapshot-0



snapshot-1

...

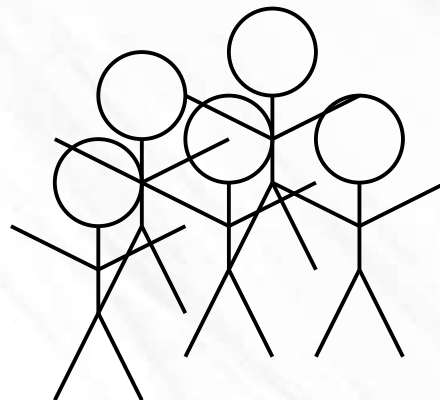
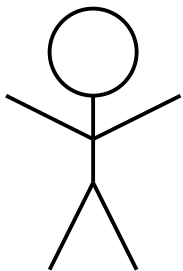


snapshot-99

...



snapshot-109



# Branches



working



snapshot-0



snapshot-1

...



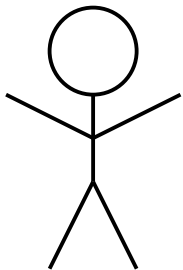
snapshot-99

...

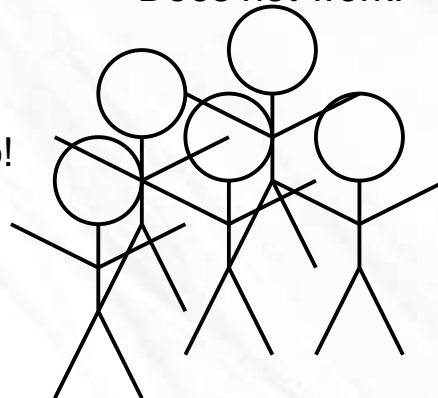


snapshot-109

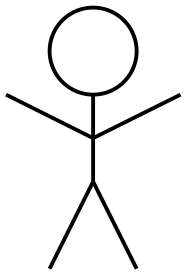
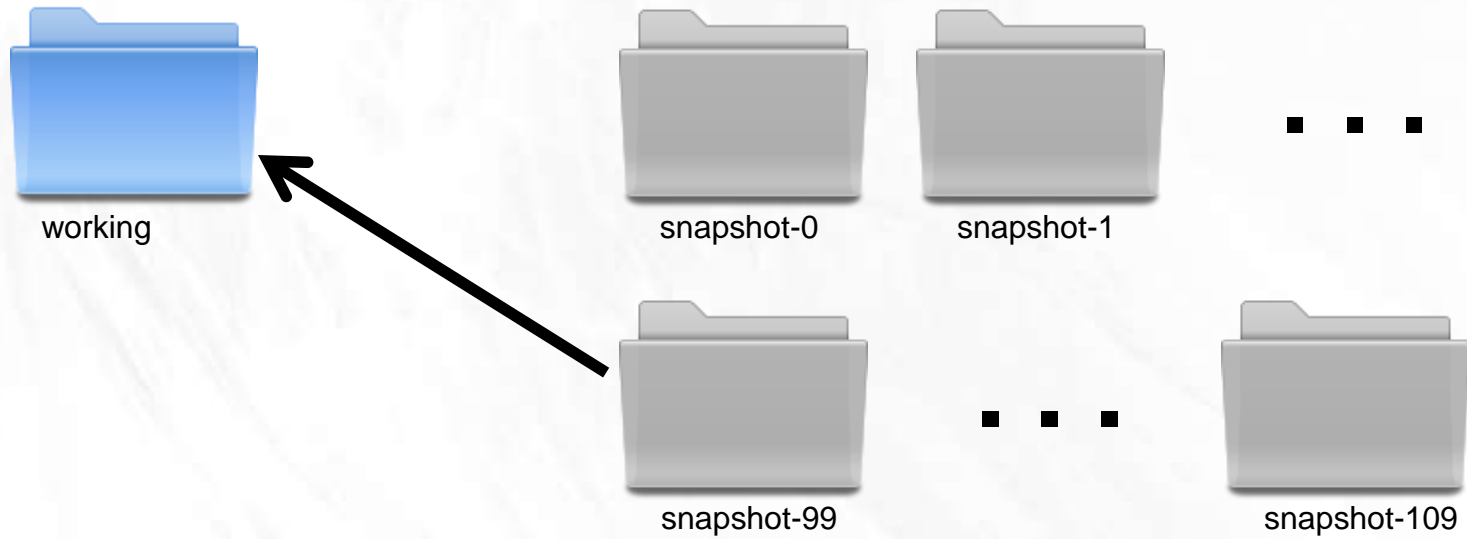
Does not work!



Boo!



# Branches





# Branches



working



snapshot-0



snapshot-1

...



snapshot-99

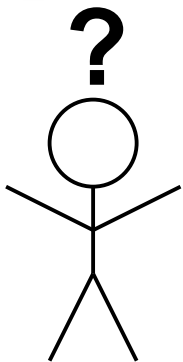
...



snapshot-109



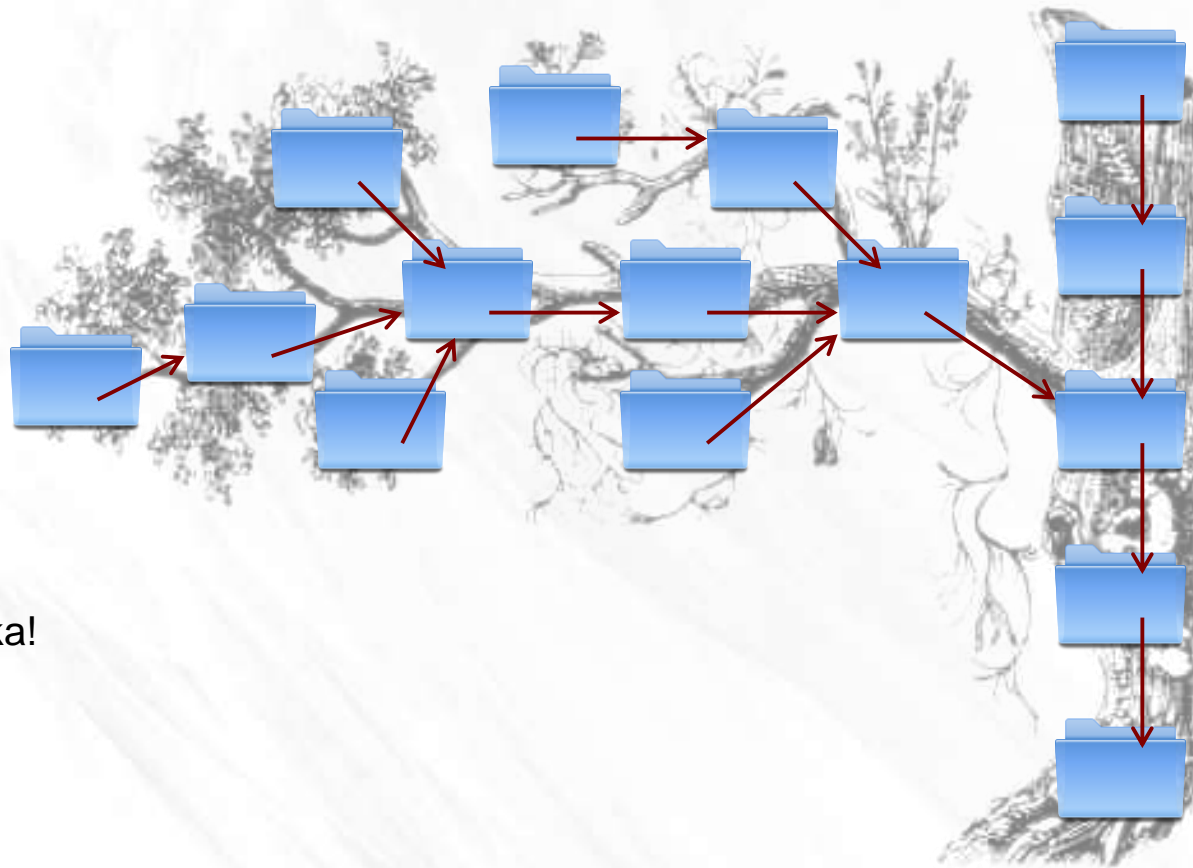
snapshot-110



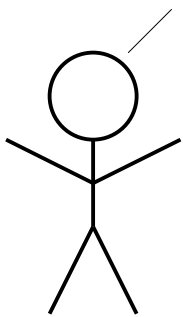
# Branches



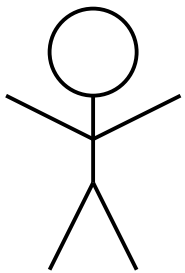
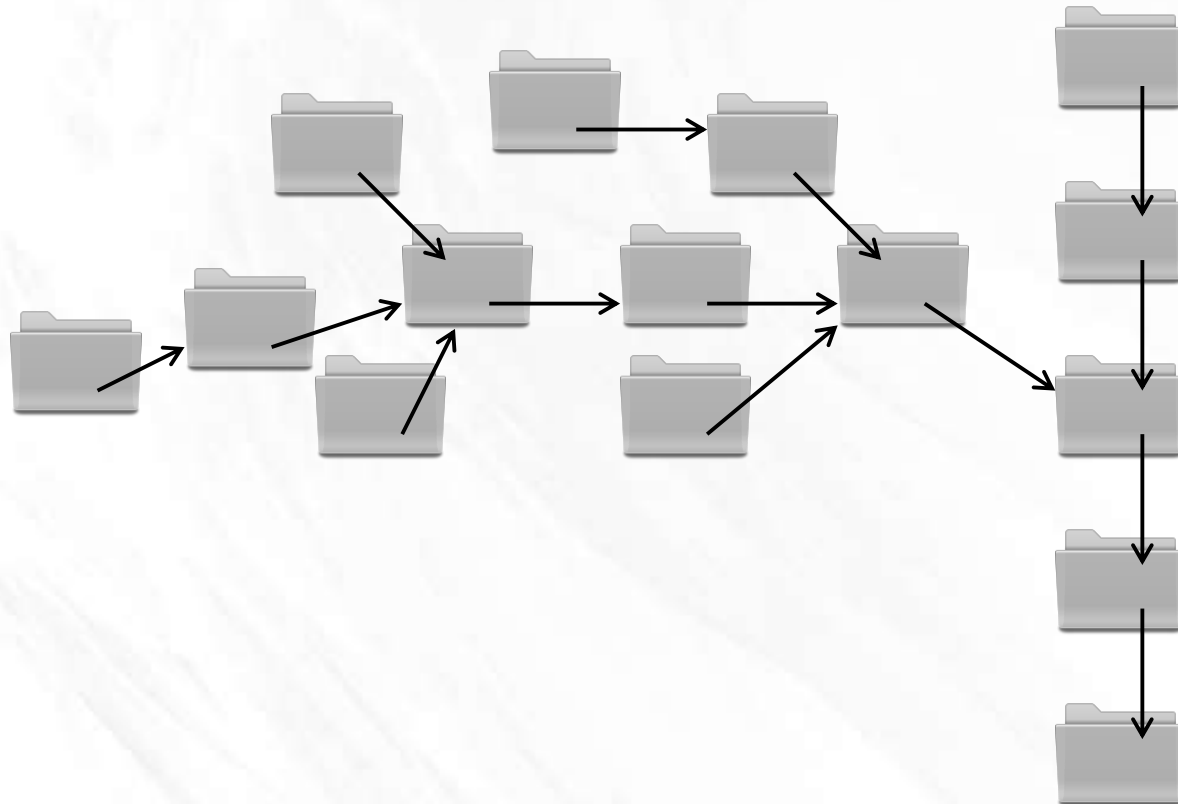
# Branches



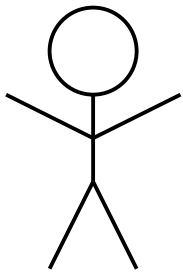
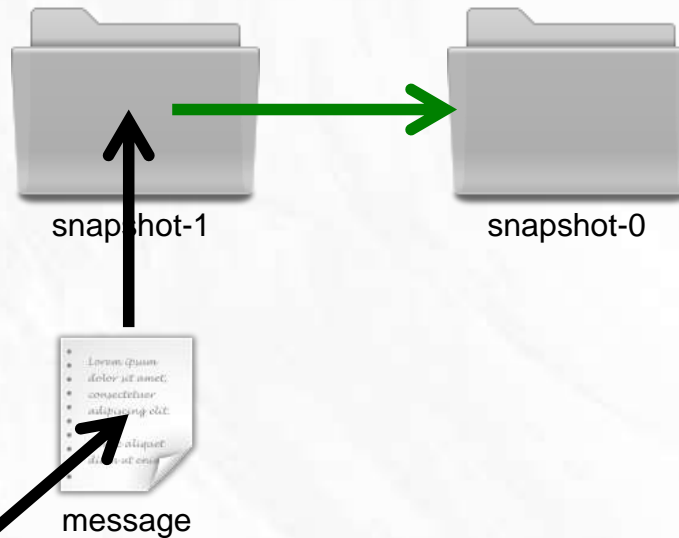
Eureka!



# Branches



# Branches



2009-05-21 23:45:01

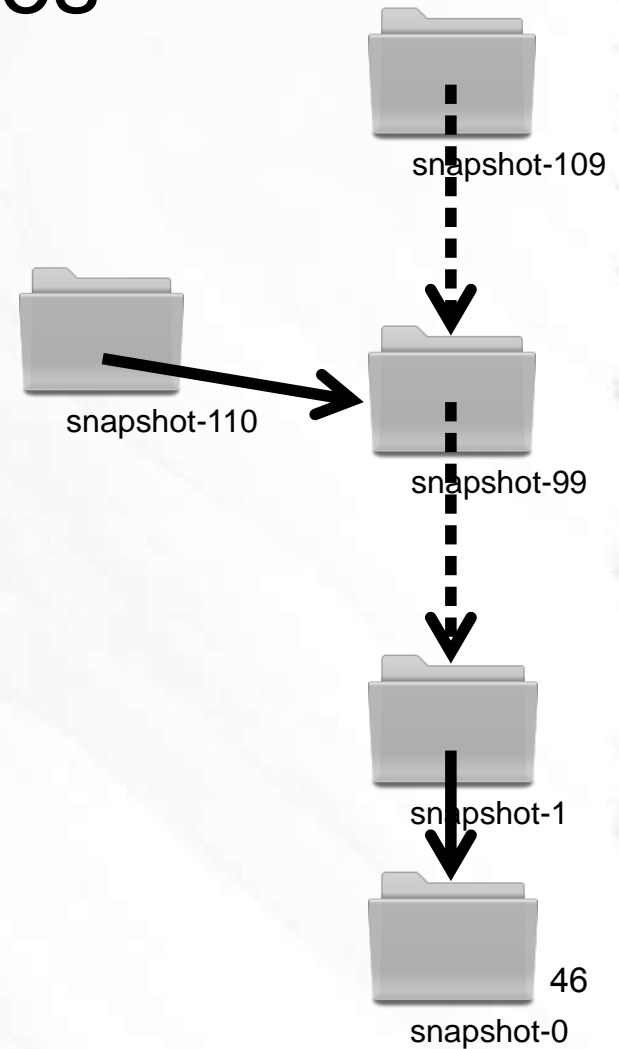
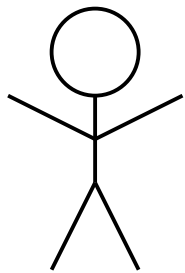
parent: snapshot-0

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

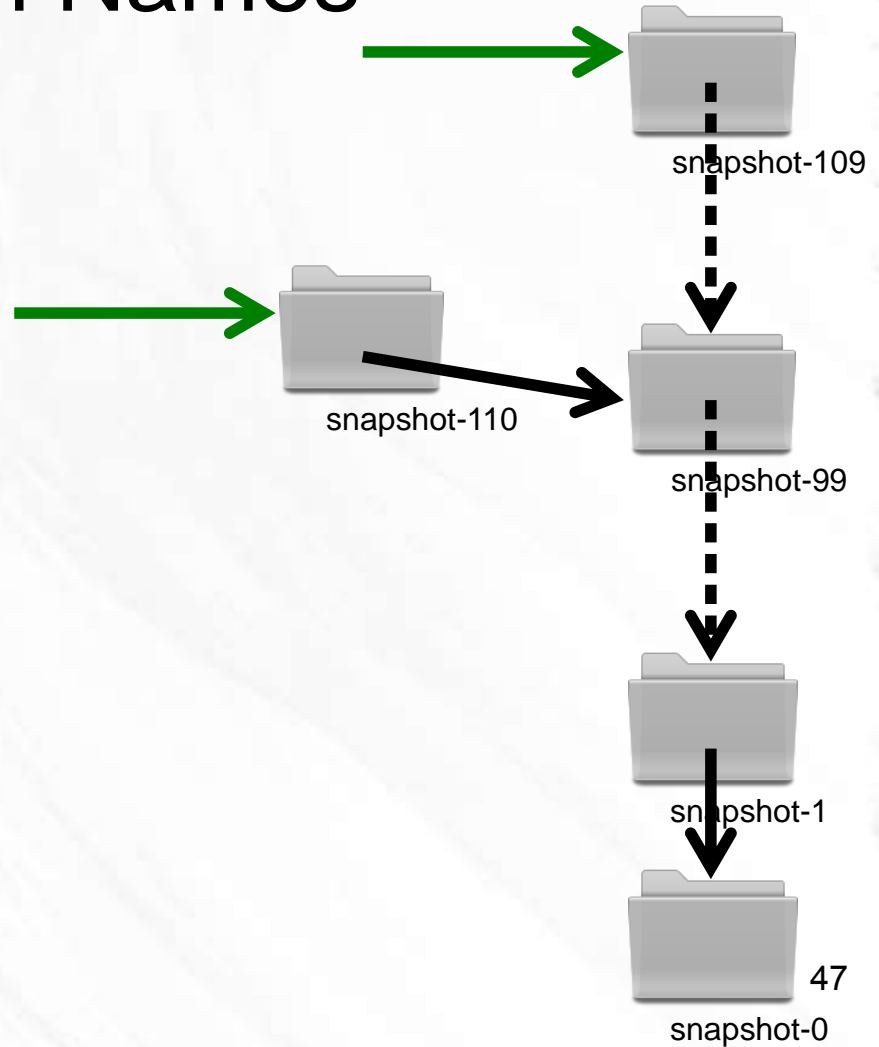
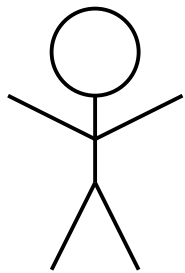
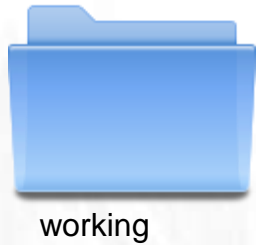
# Branch Names



working



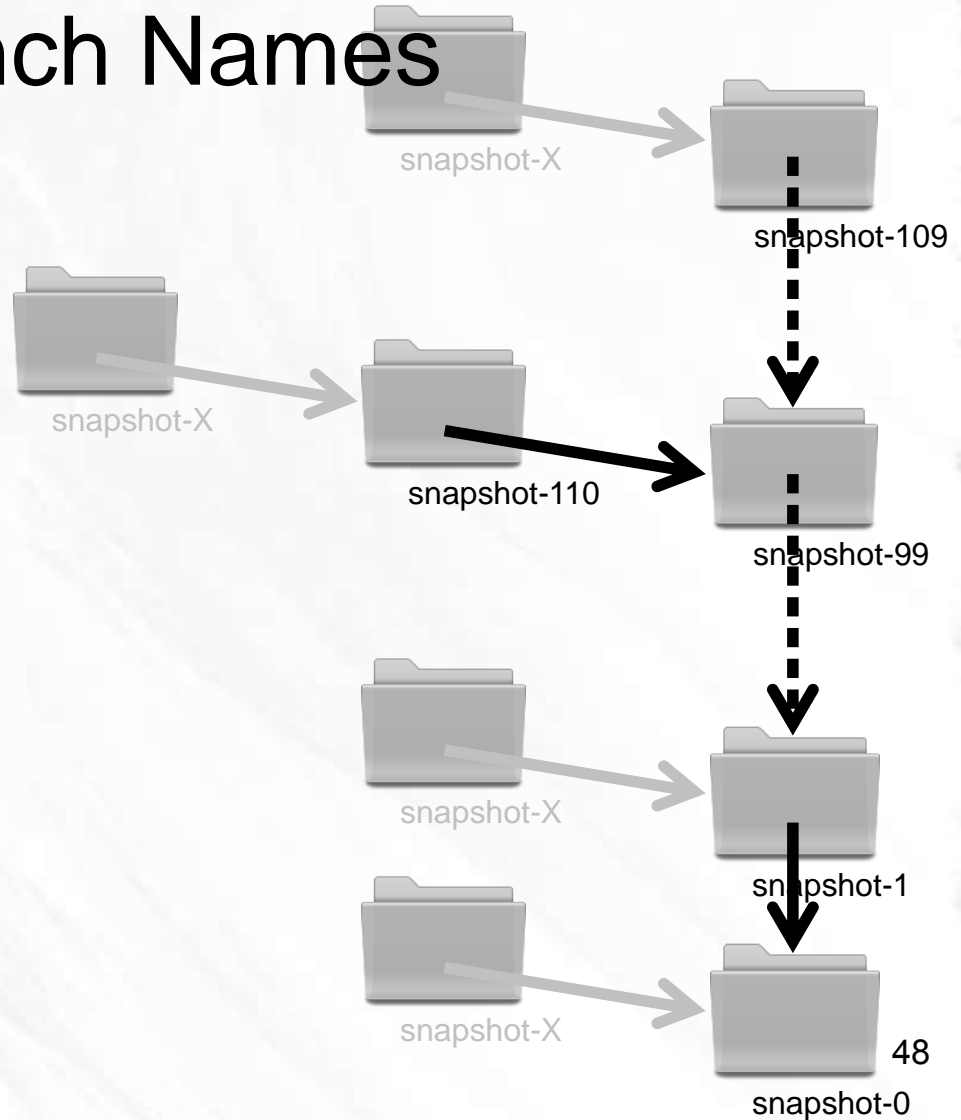
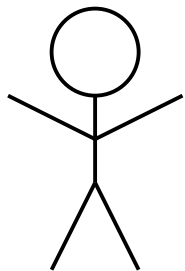
# Branch Names



# Branch Names

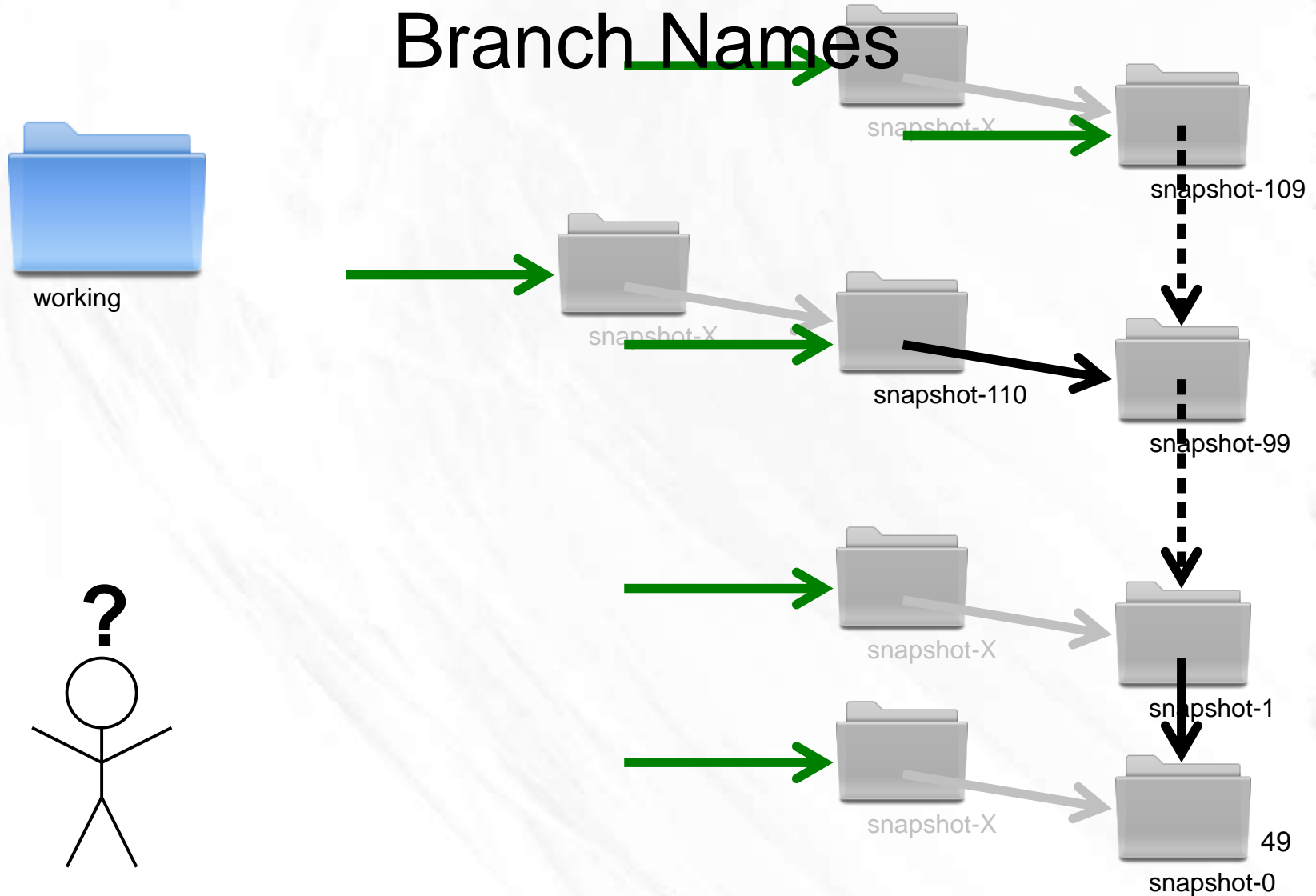


working

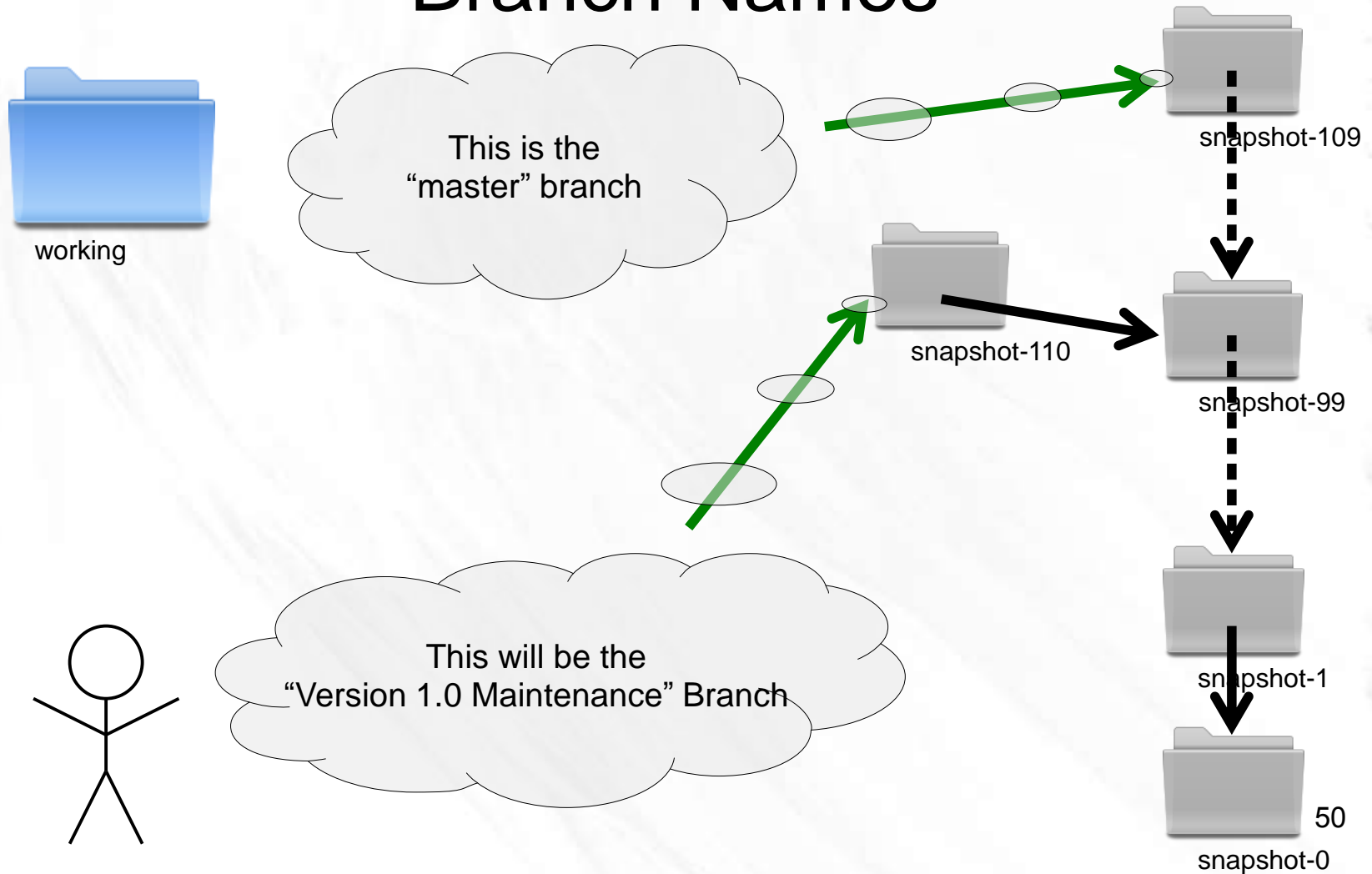




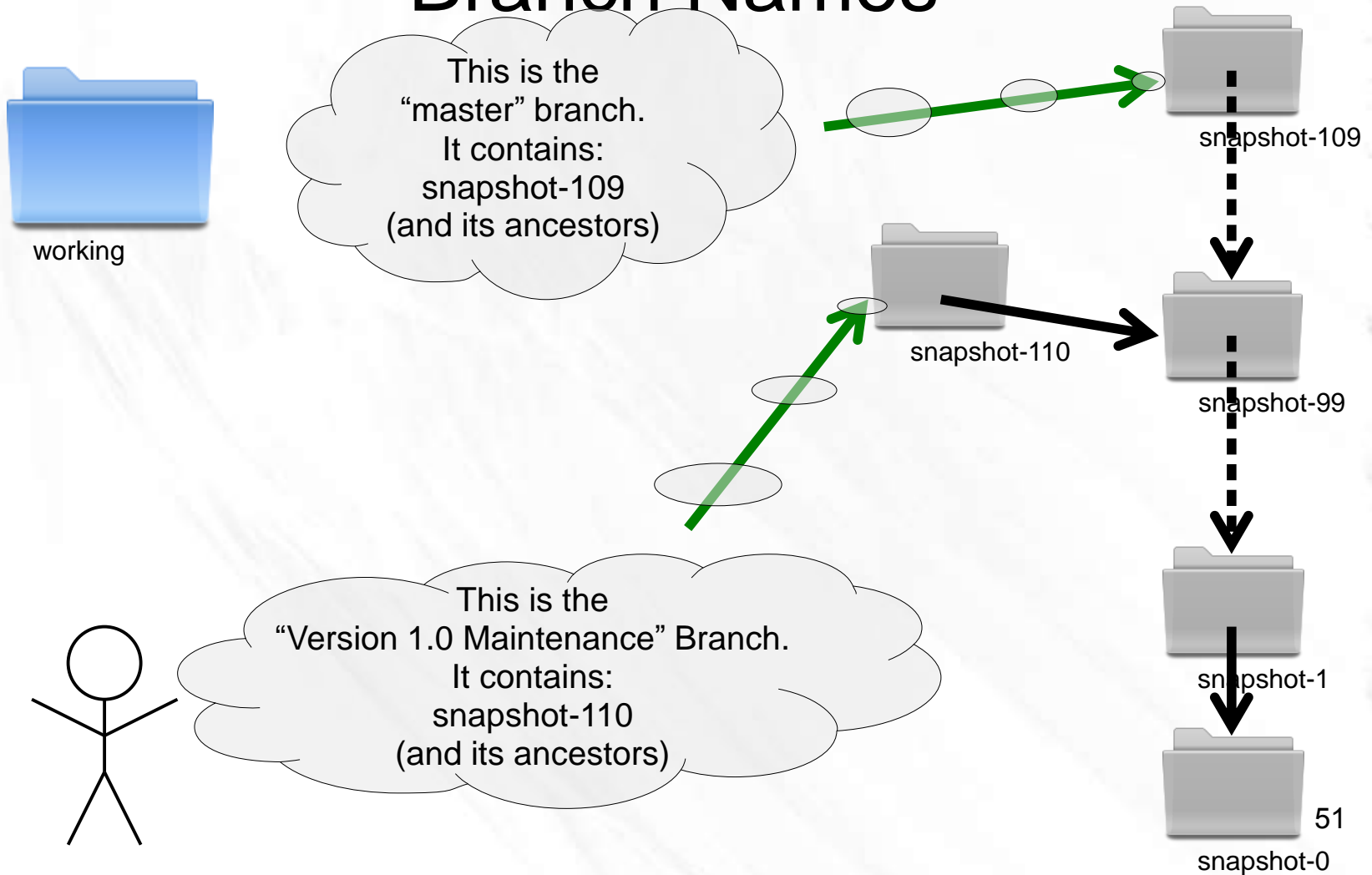
# Branch Names



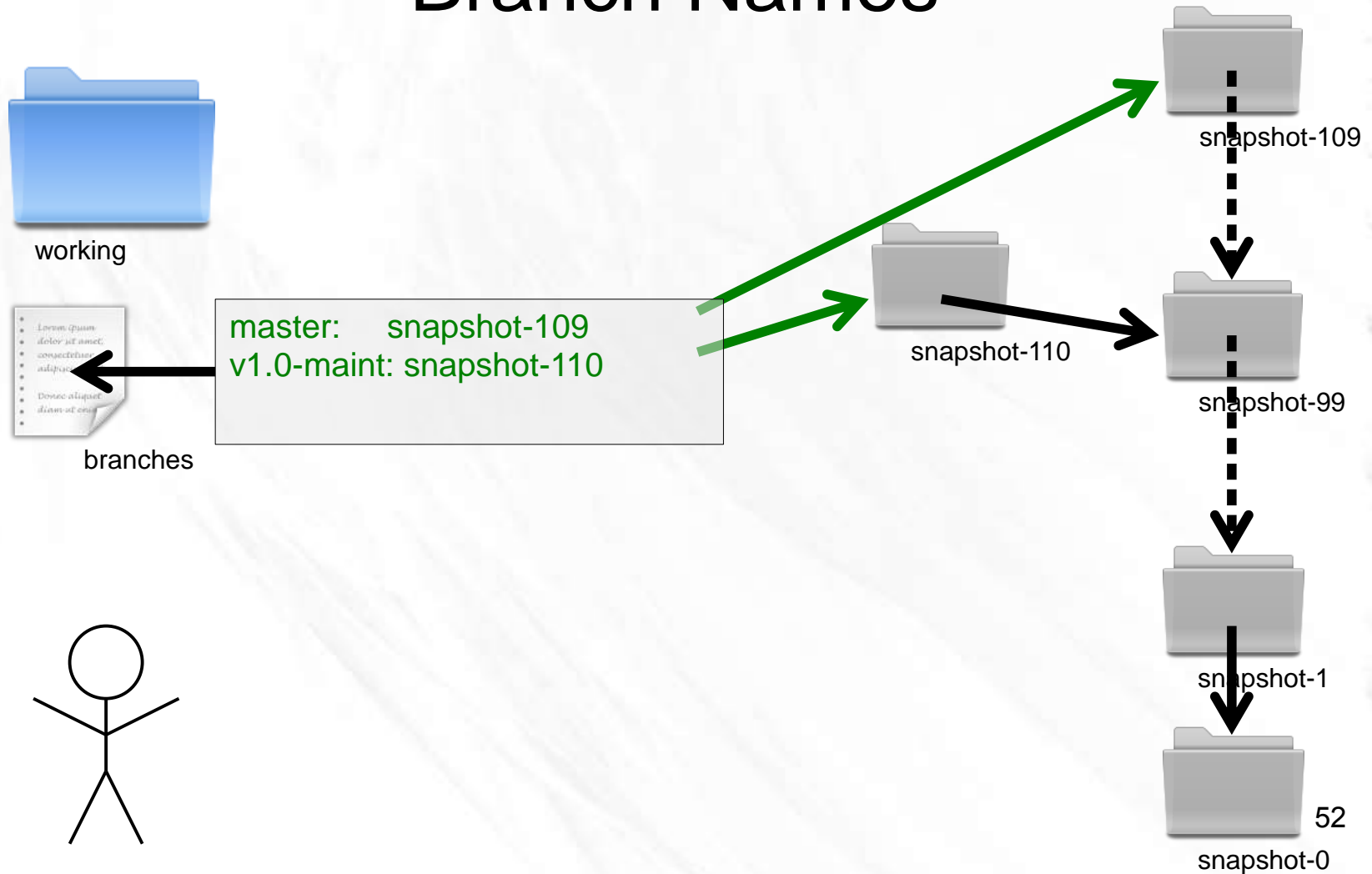
# Branch Names



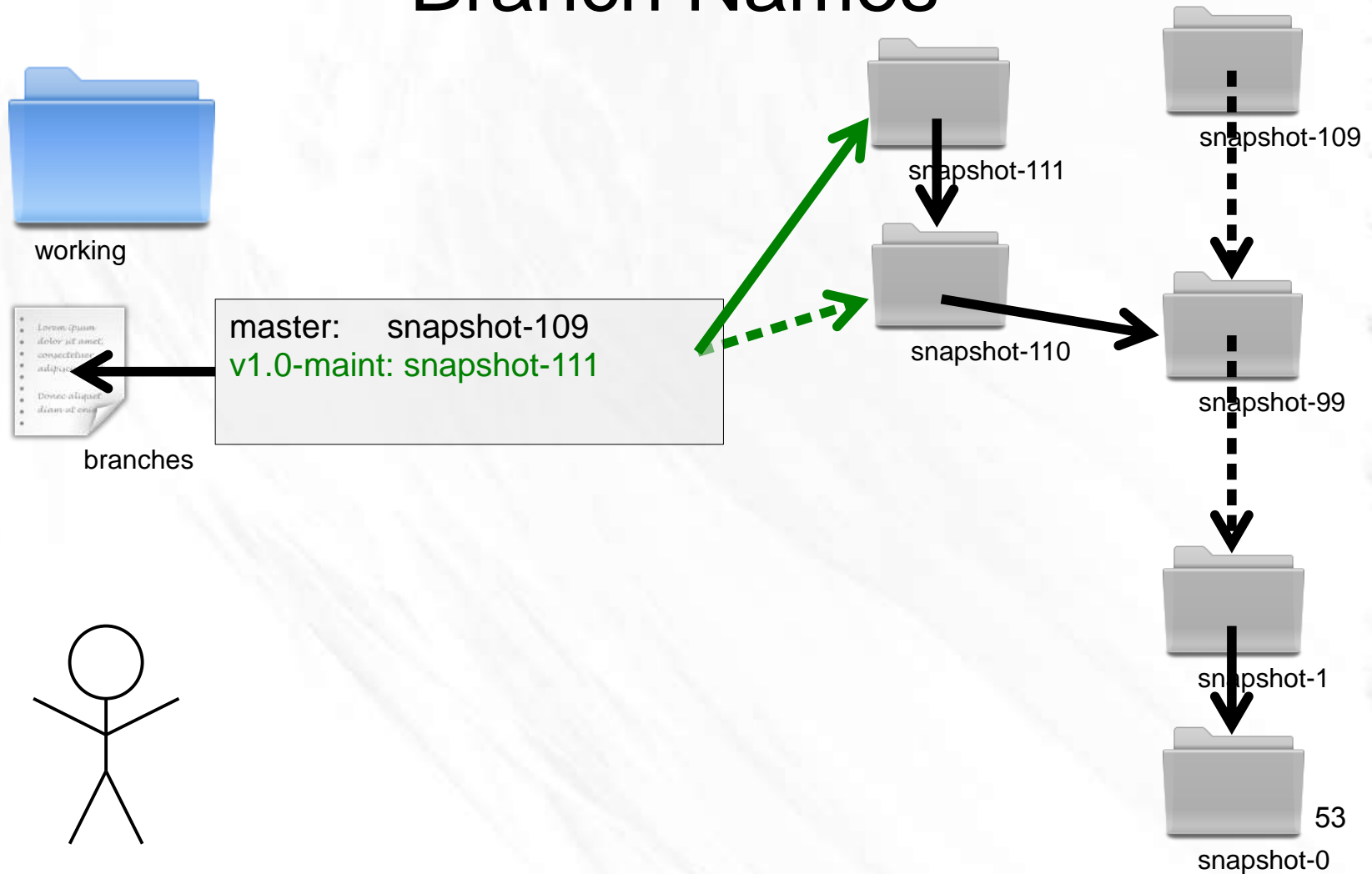
# Branch Names



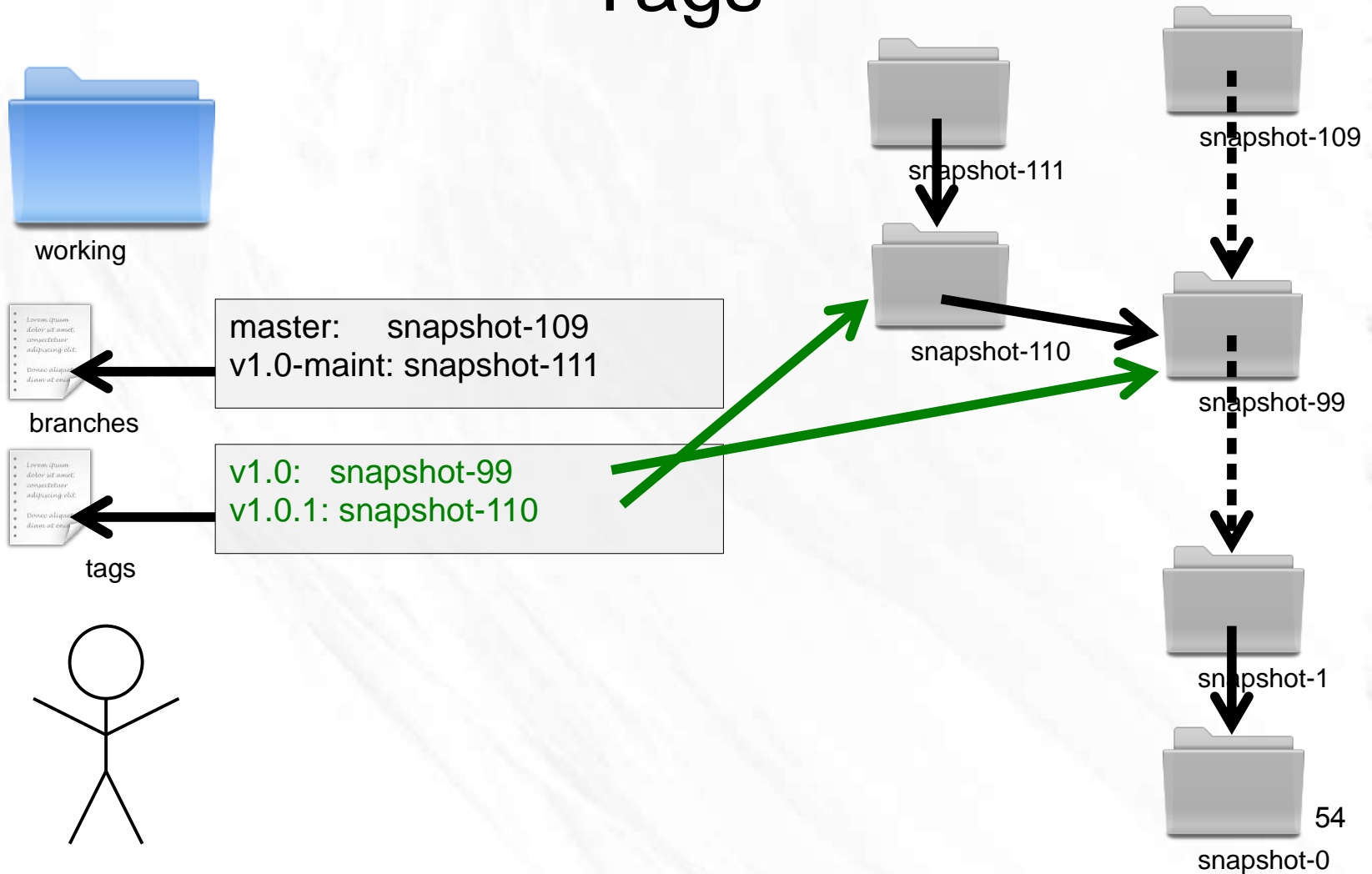
# Branch Names



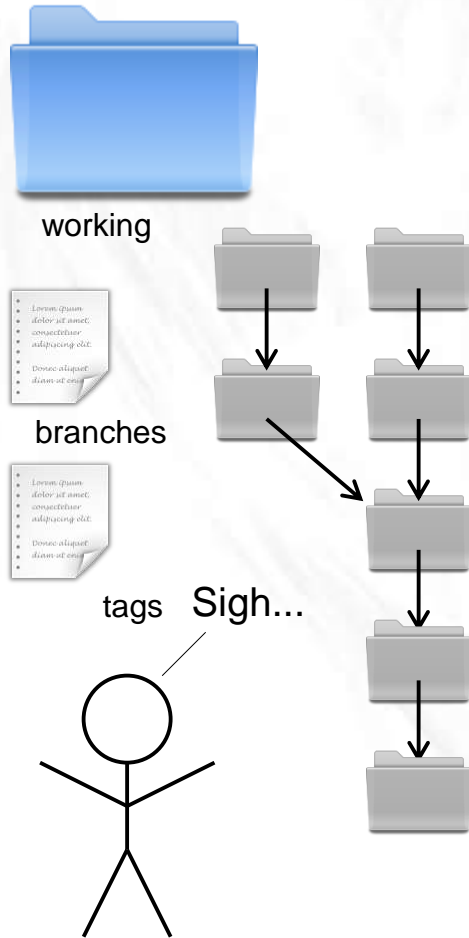
# Branch Names



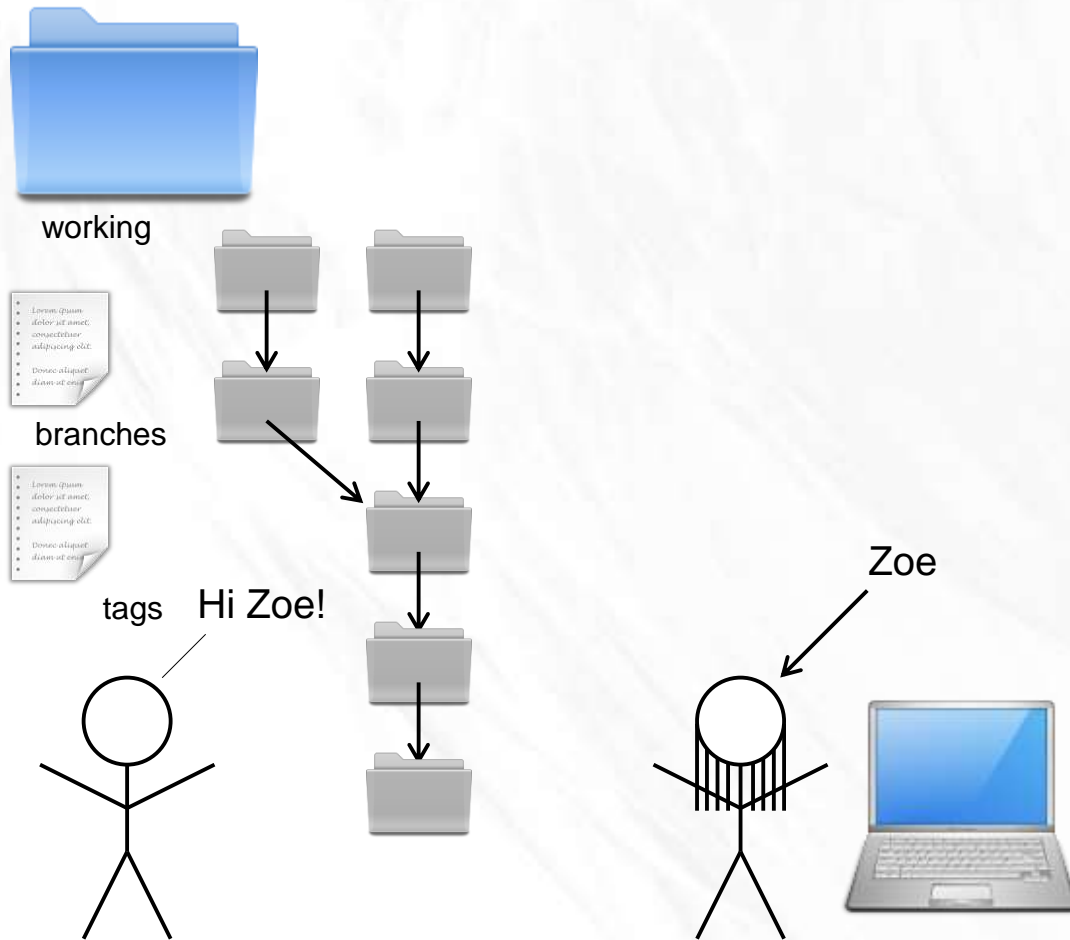
# Tags



# Distributed



# Distributed





# Distributed



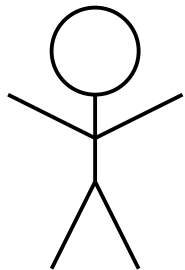
working



branches



tags



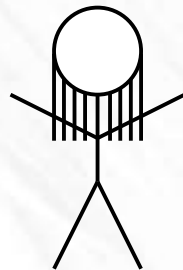
working



branches



tags



# Distributed



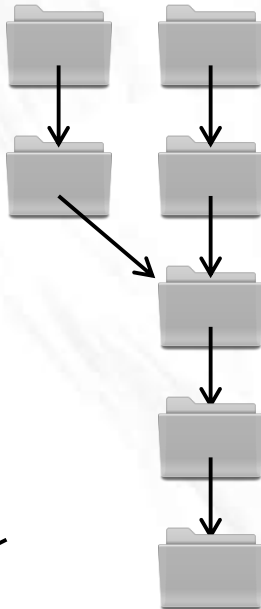
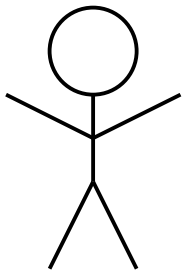
working



branches



tags



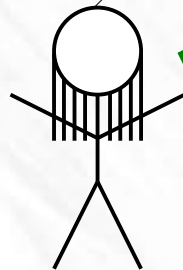
working



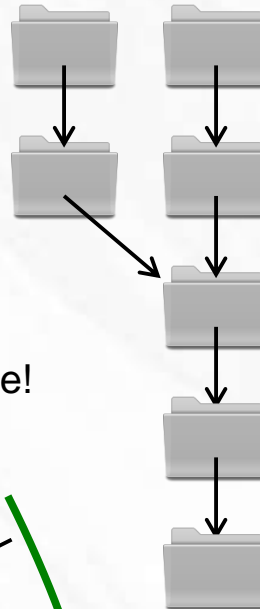
branches



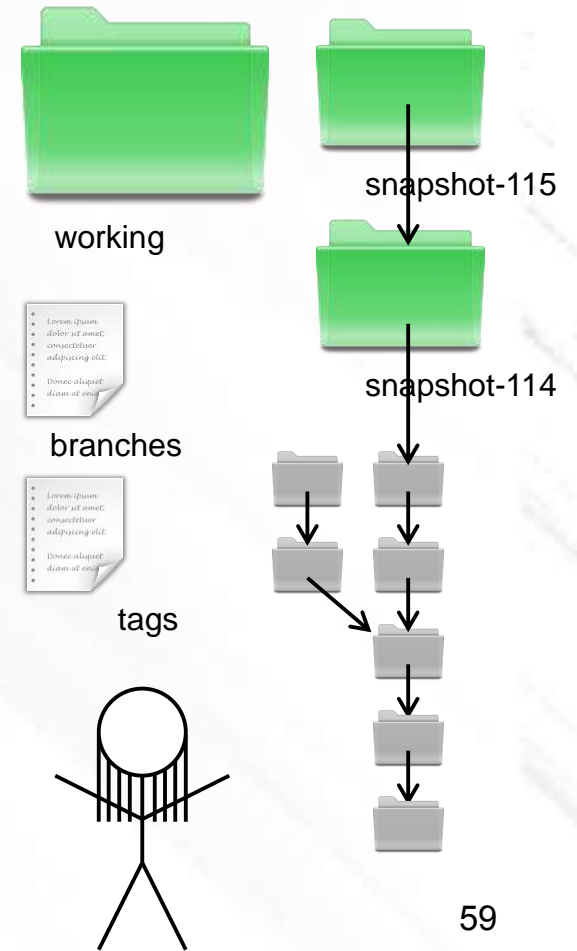
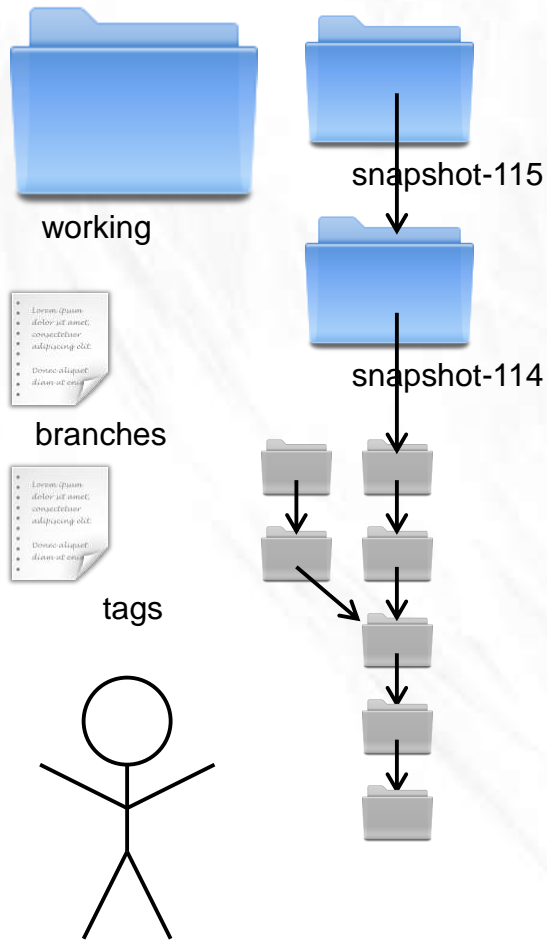
tags



Bye!



# Distributed



# Distributed



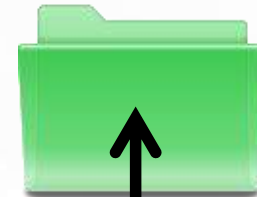
snapshot-114



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 60  
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

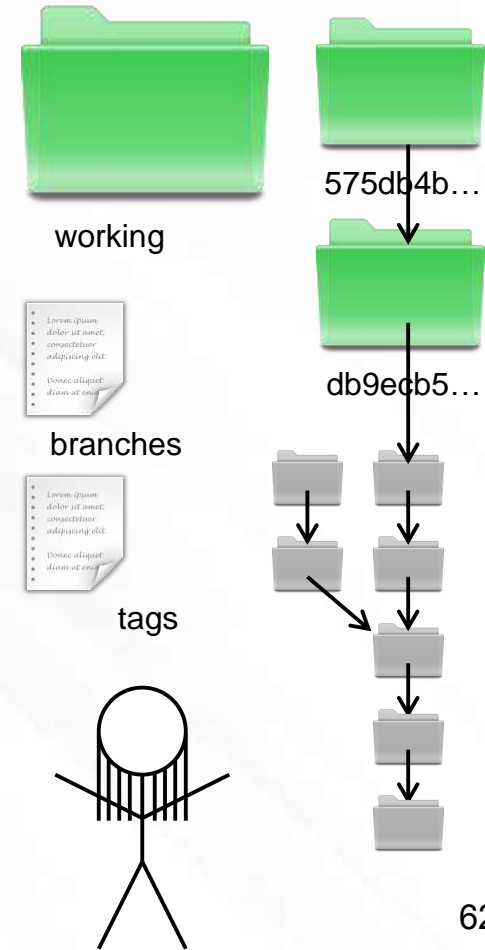
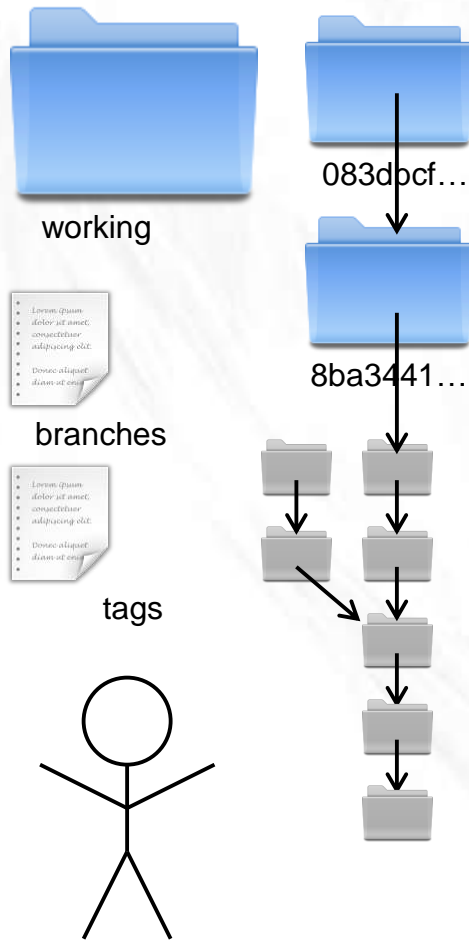


message

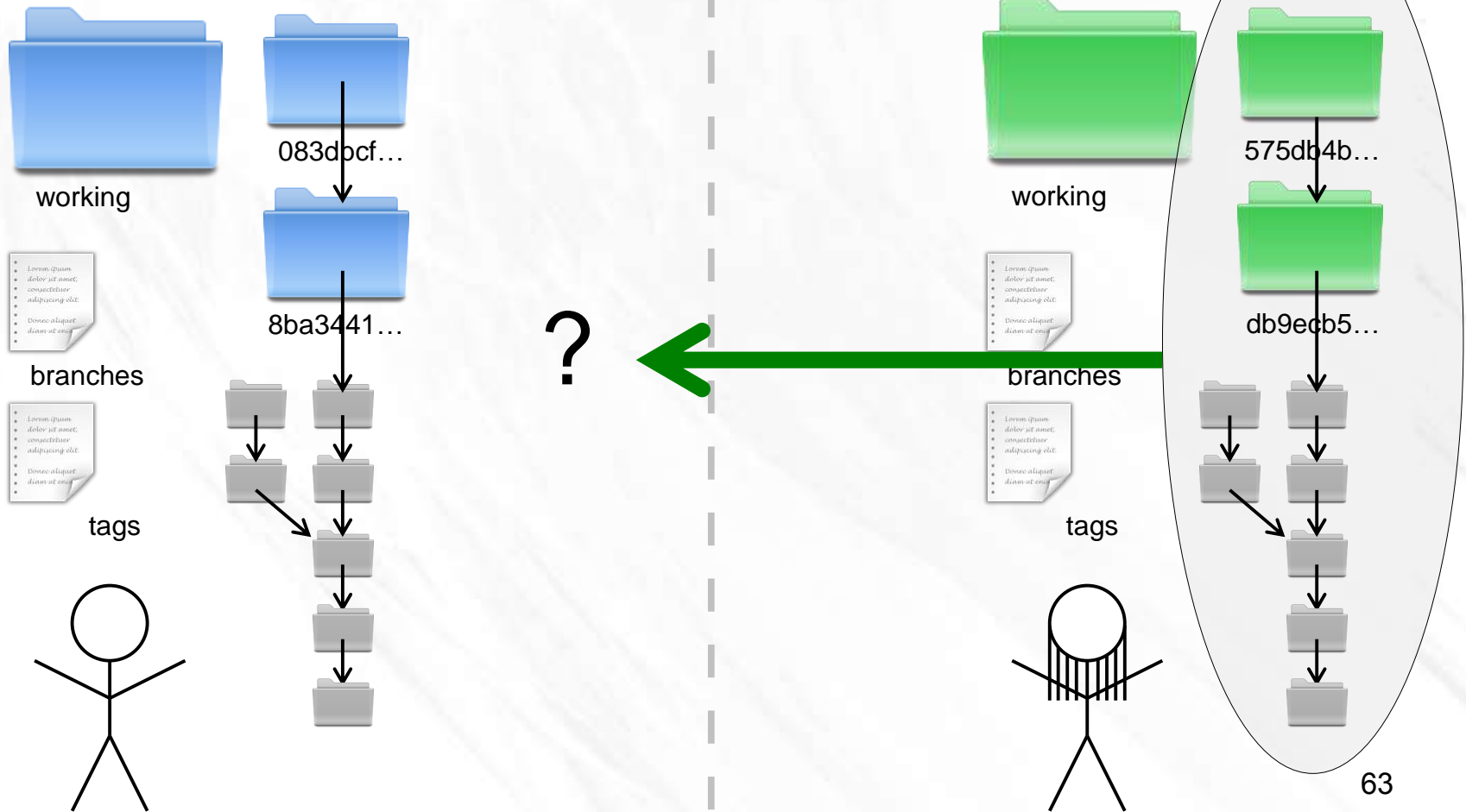
2009-05-21 23:45:01  
parent: snapshot-113  
author: Zoe <zoe@z.oe>

Introduced a new foo, 61  
and reset the bar to  
xyzyz.

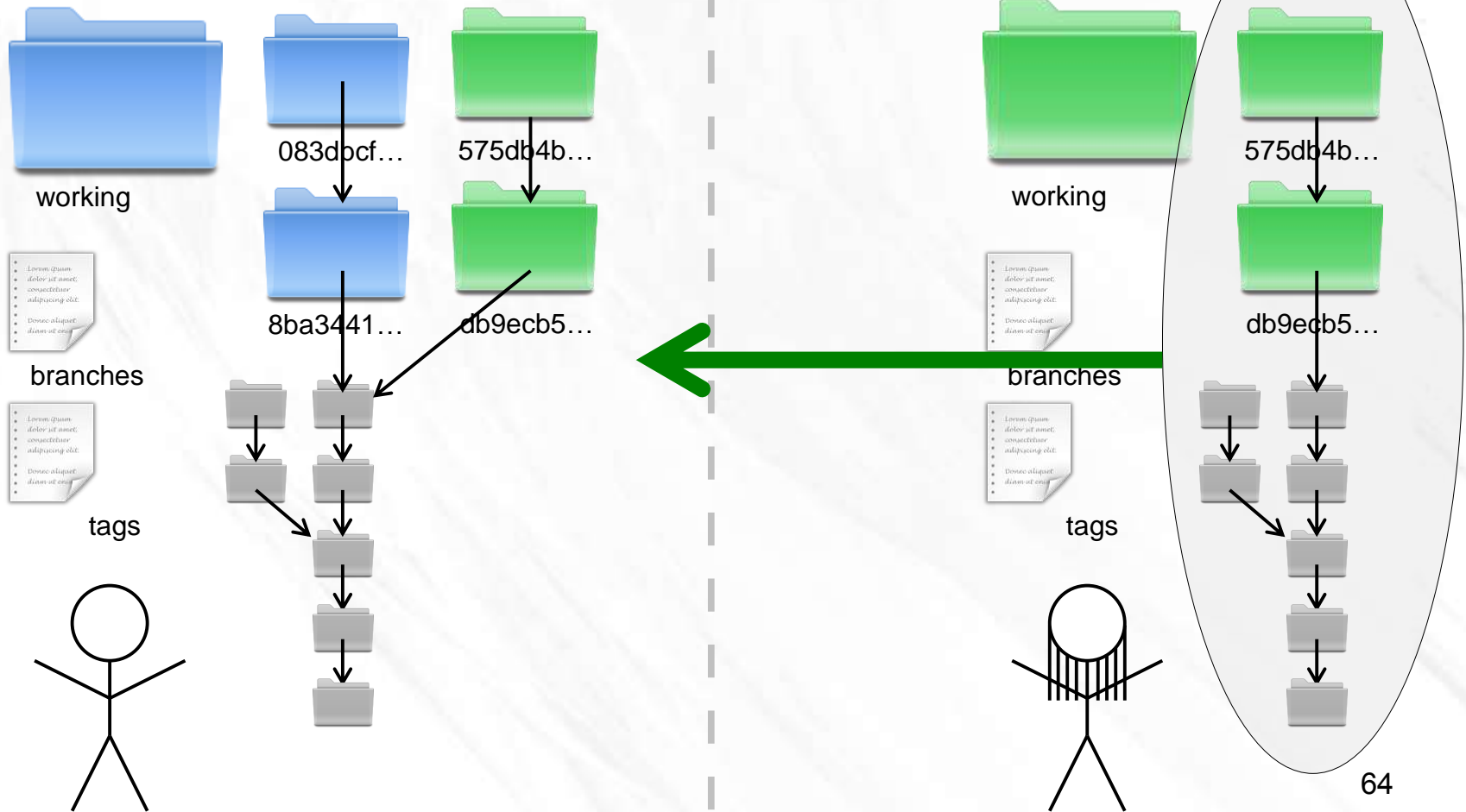
# Distributed



# Distributed

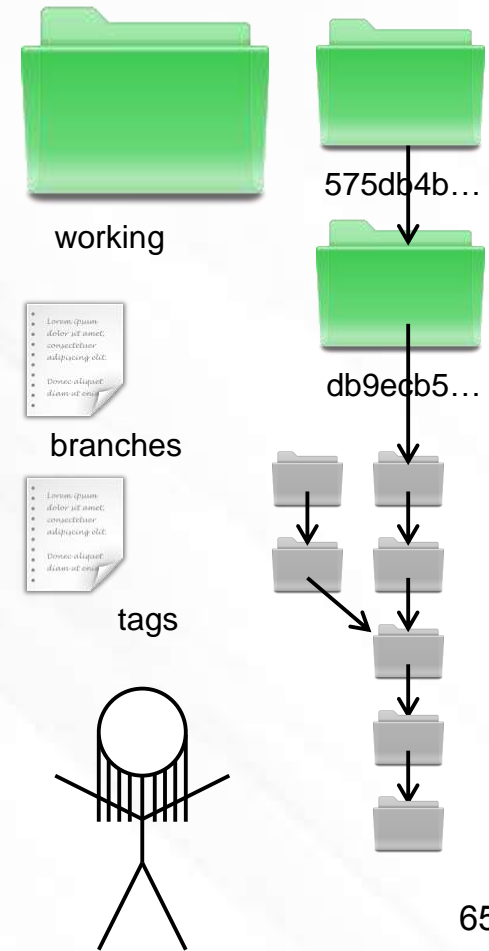
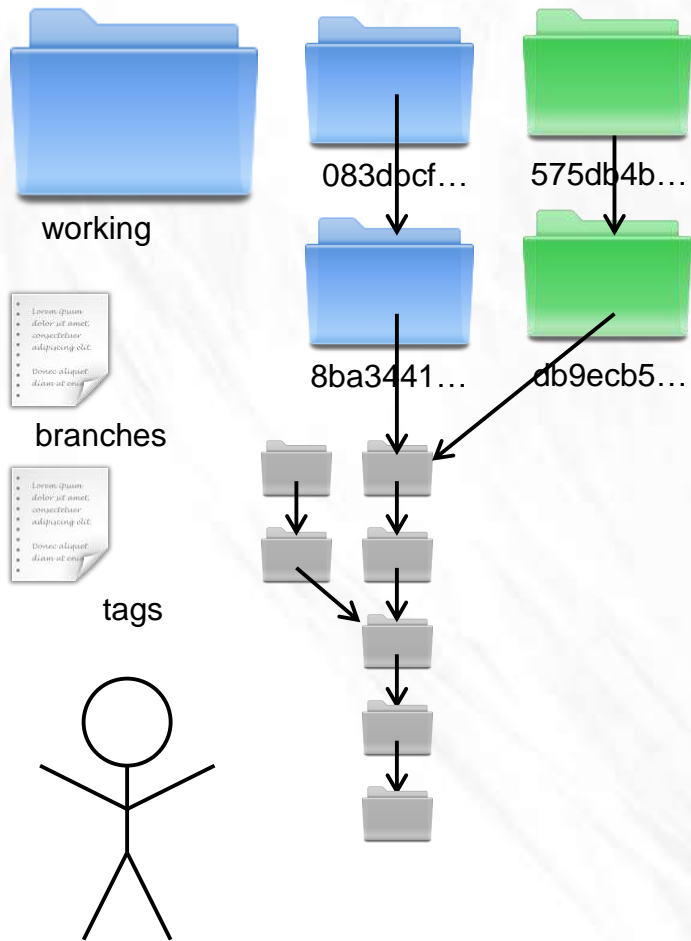


# Distributed

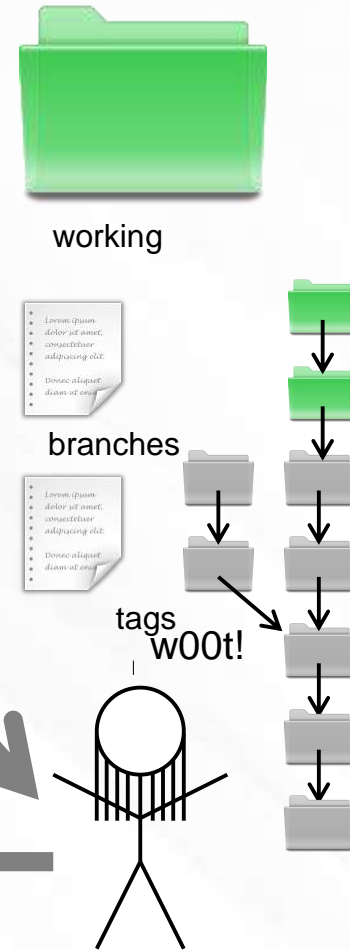
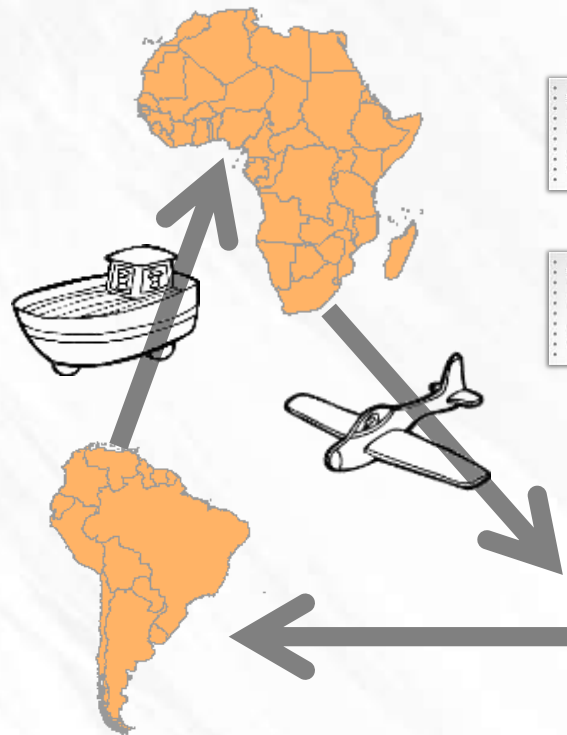
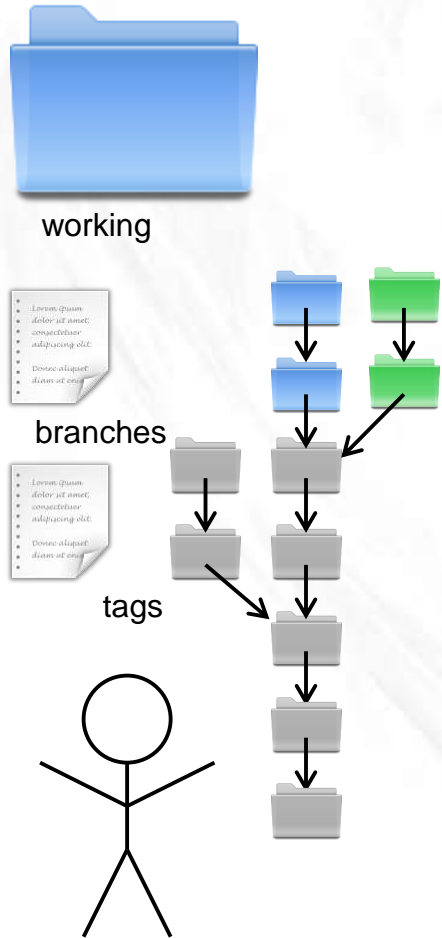




# Distributed



# Offline



# Offline



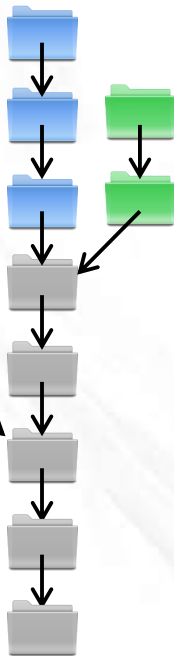
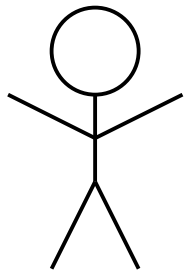
working



branches



tags



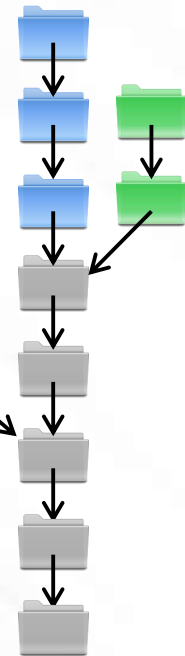
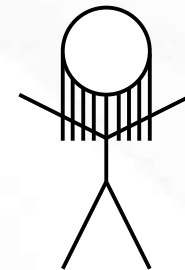
working



branches



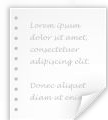
tags



# (simpler drawings)



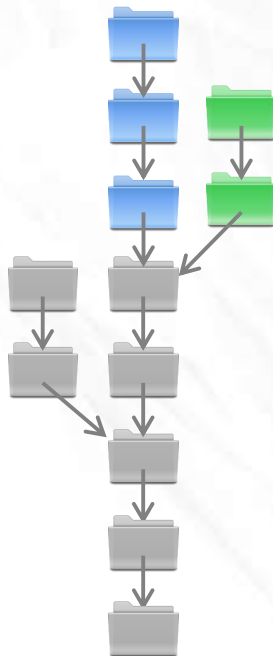
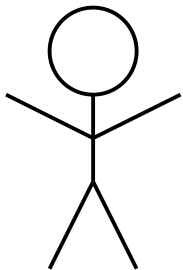
working



branches



tags



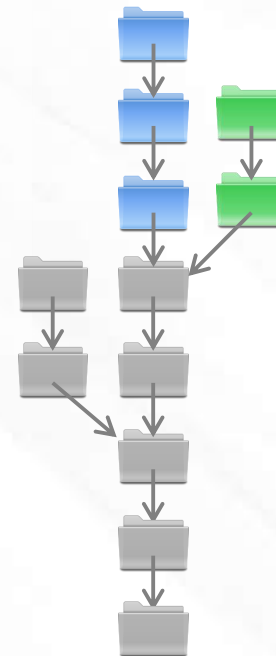
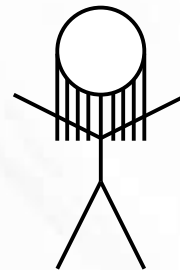
working



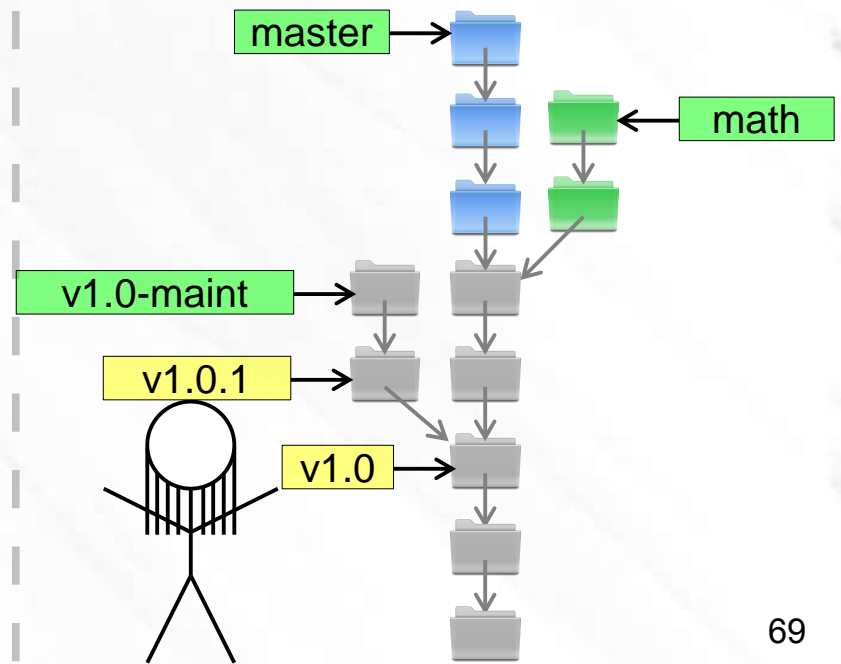
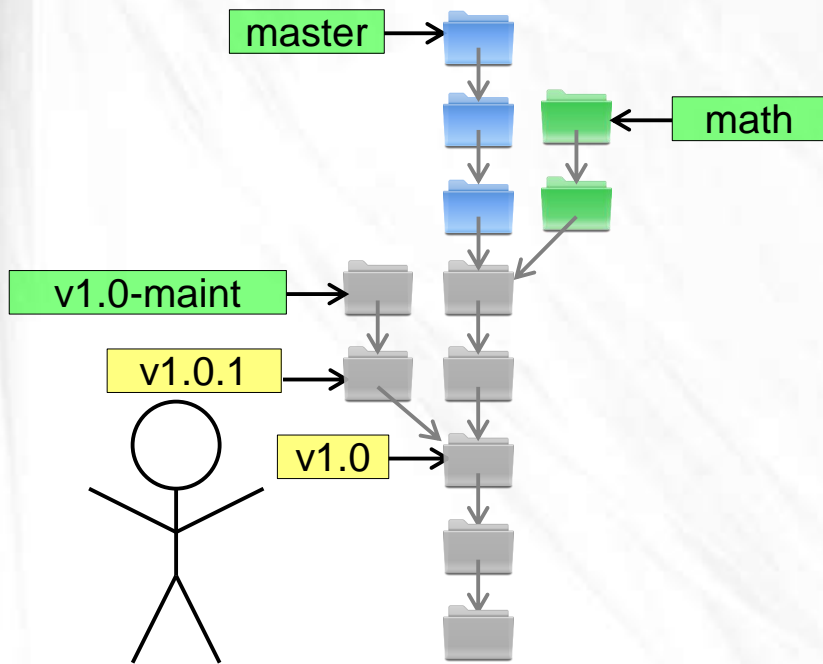
branches



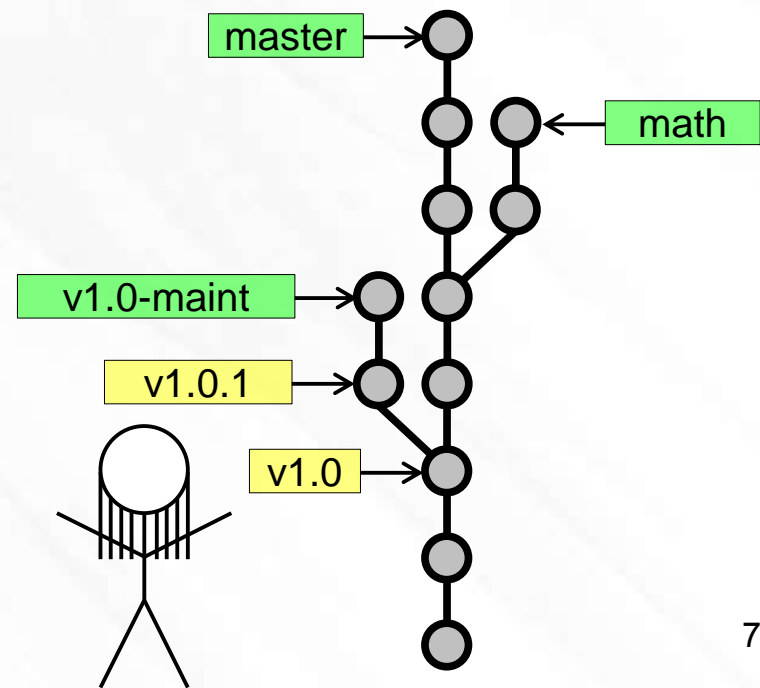
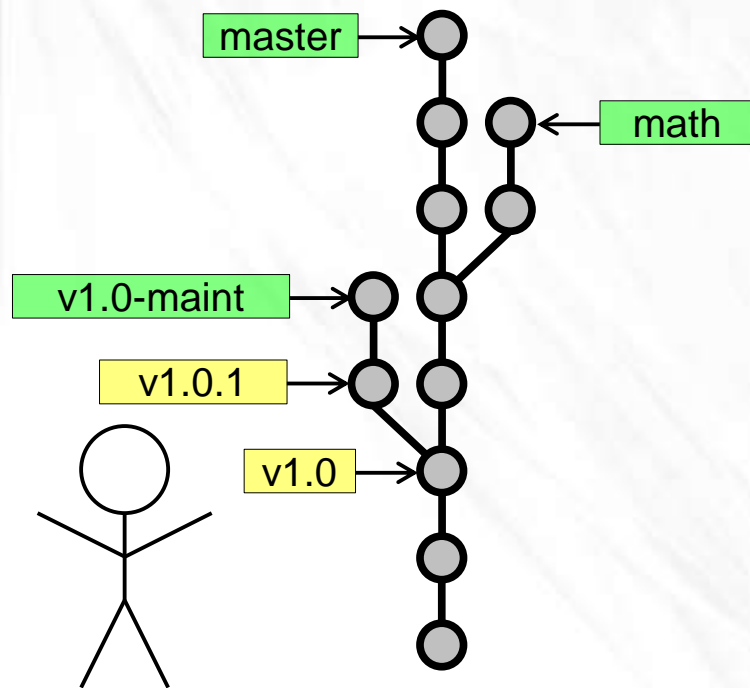
tags



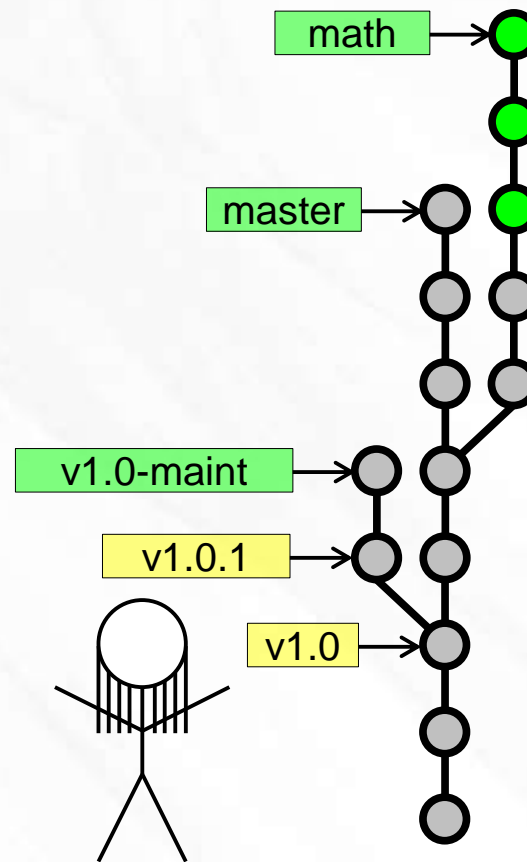
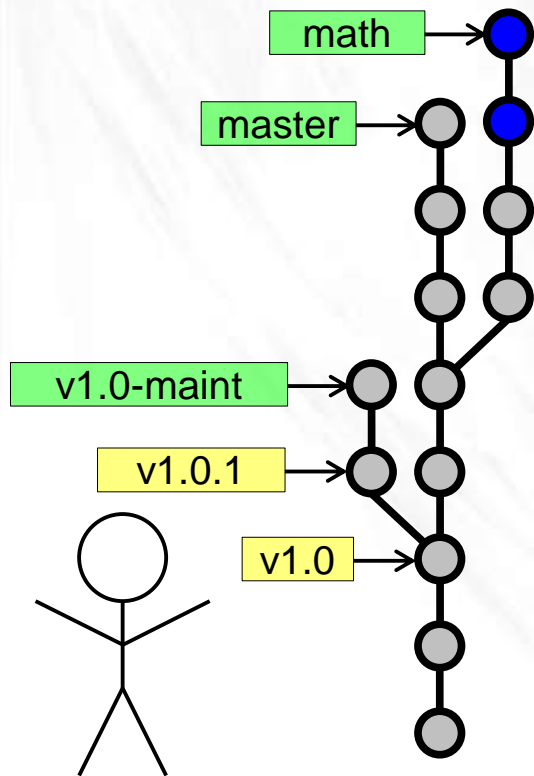
# (simpler drawings)



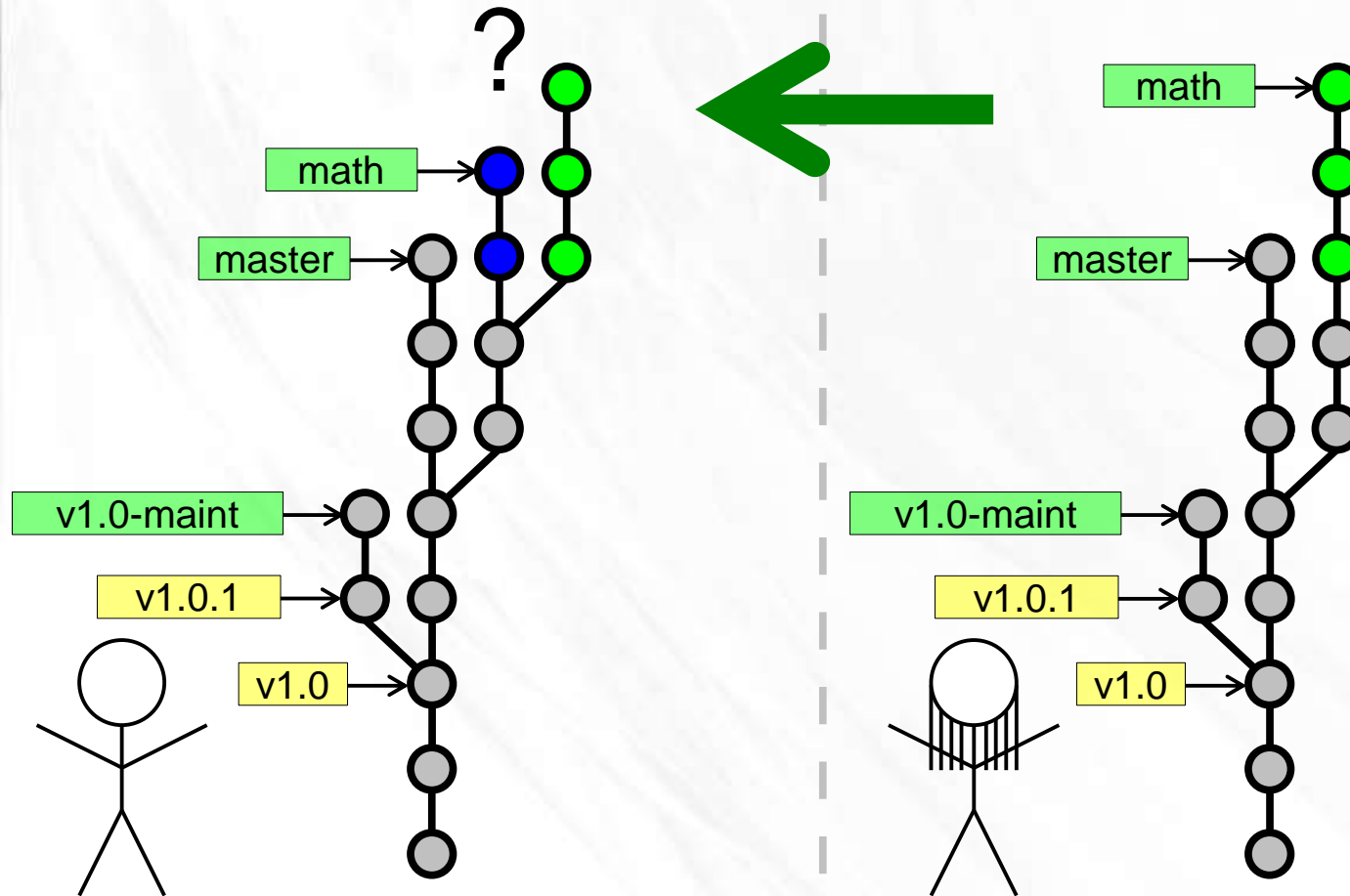
# (simpler drawings)



# Merges

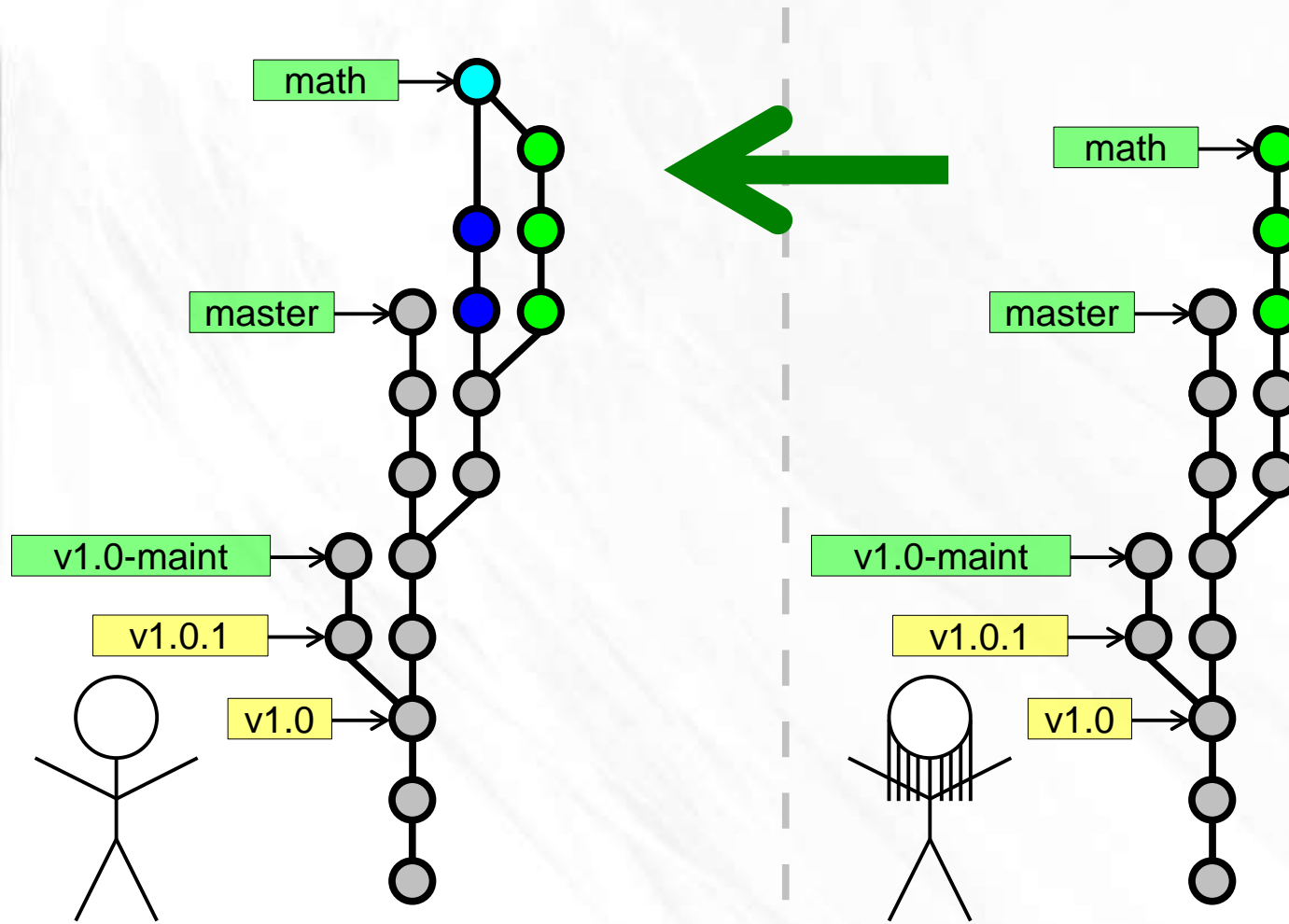


# Merges

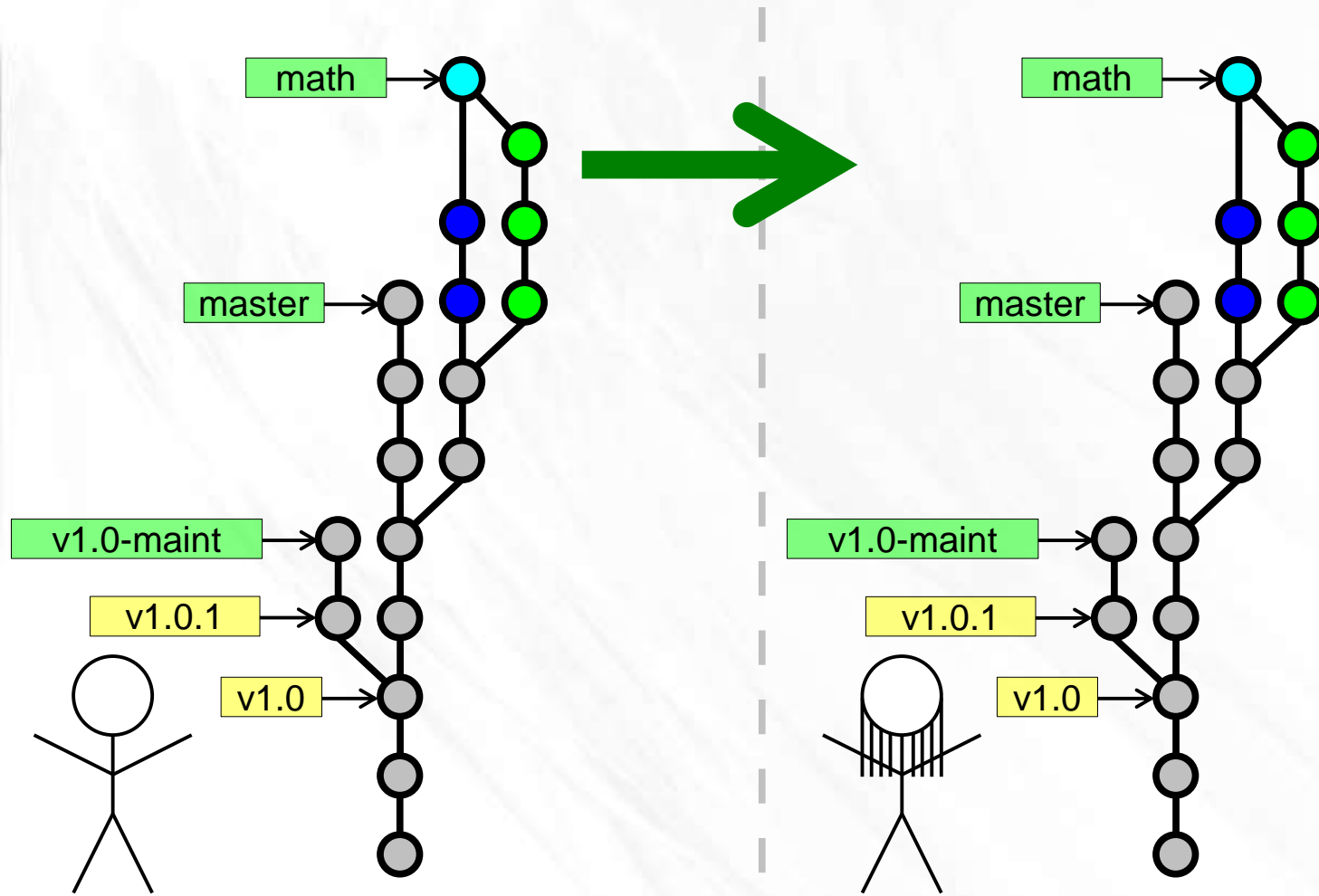




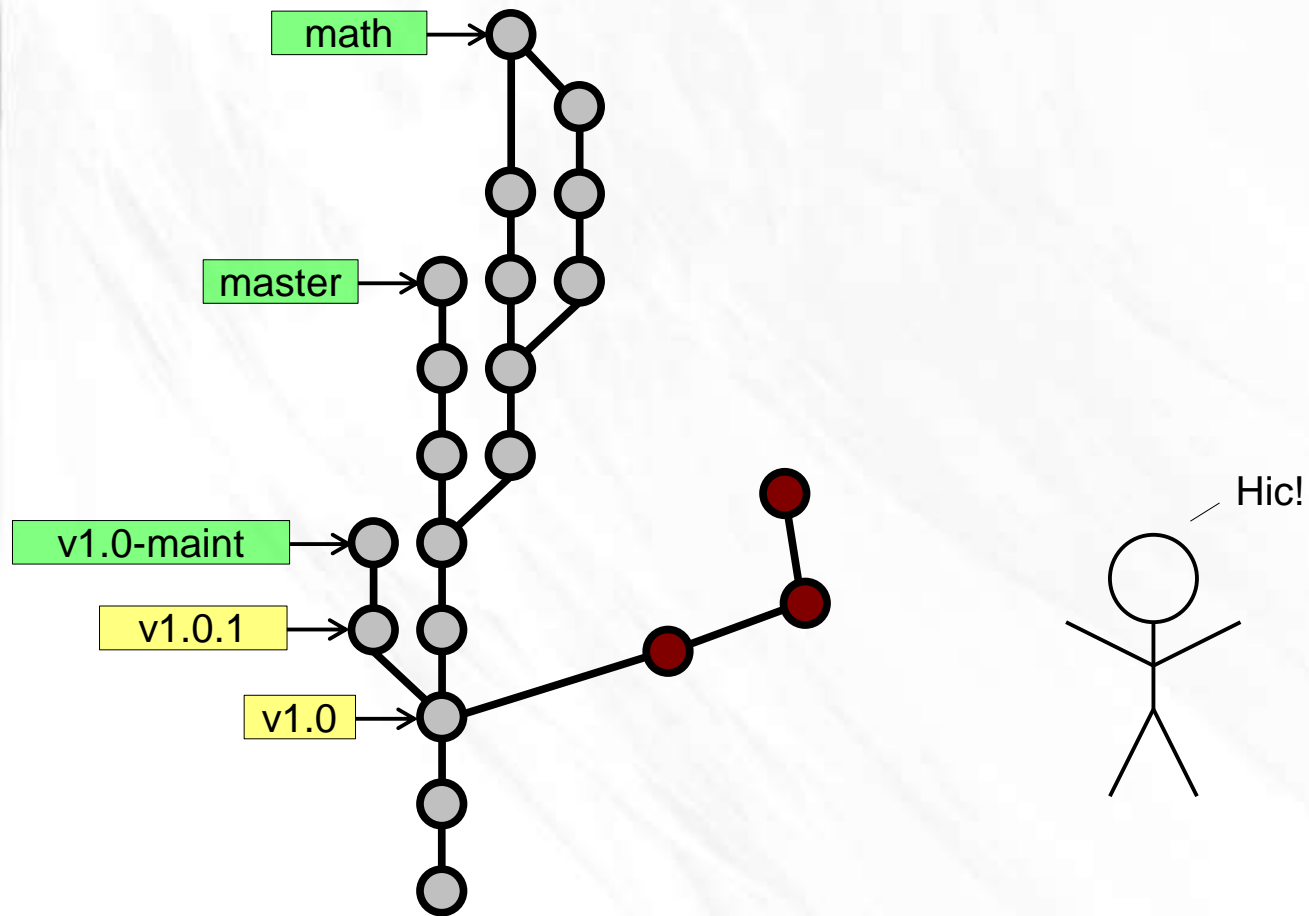
# Merges



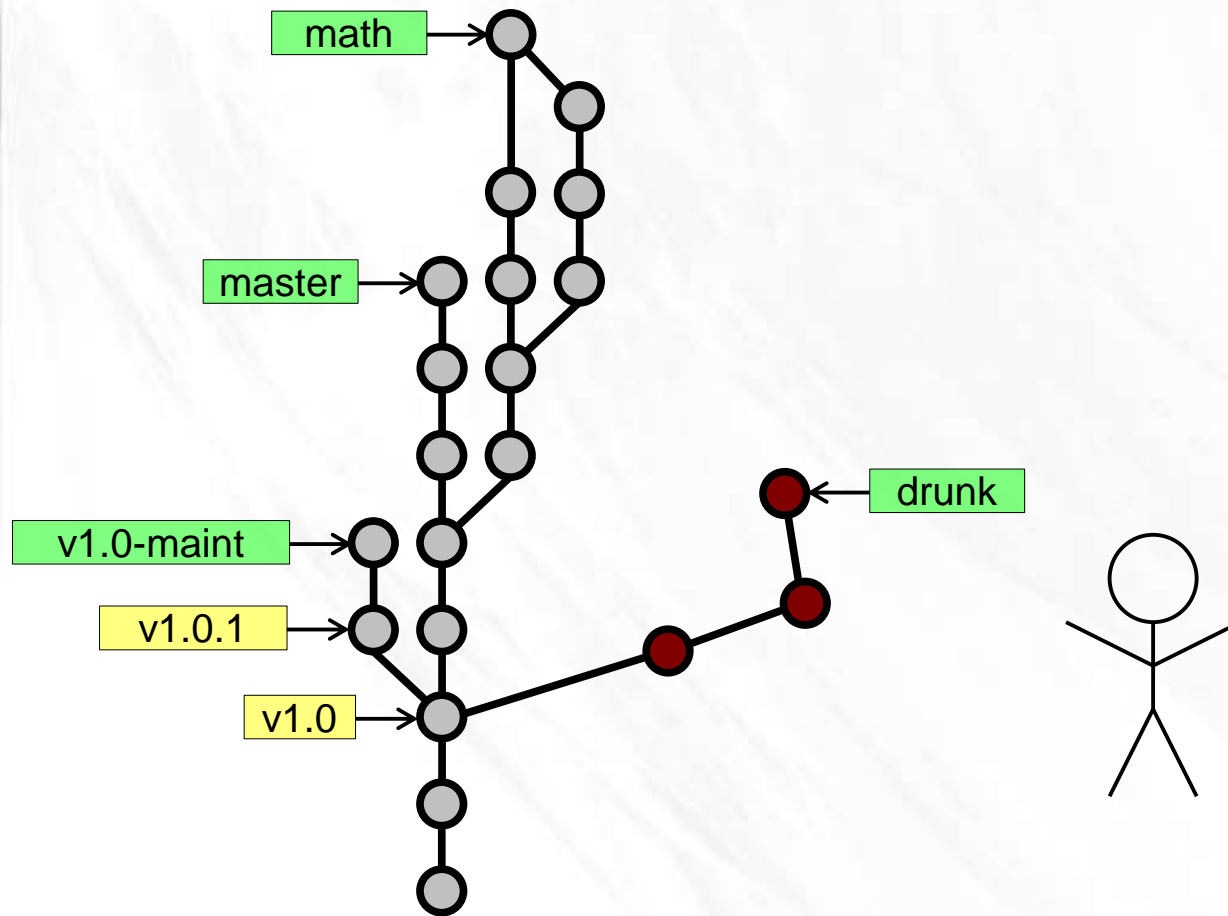
# Merges



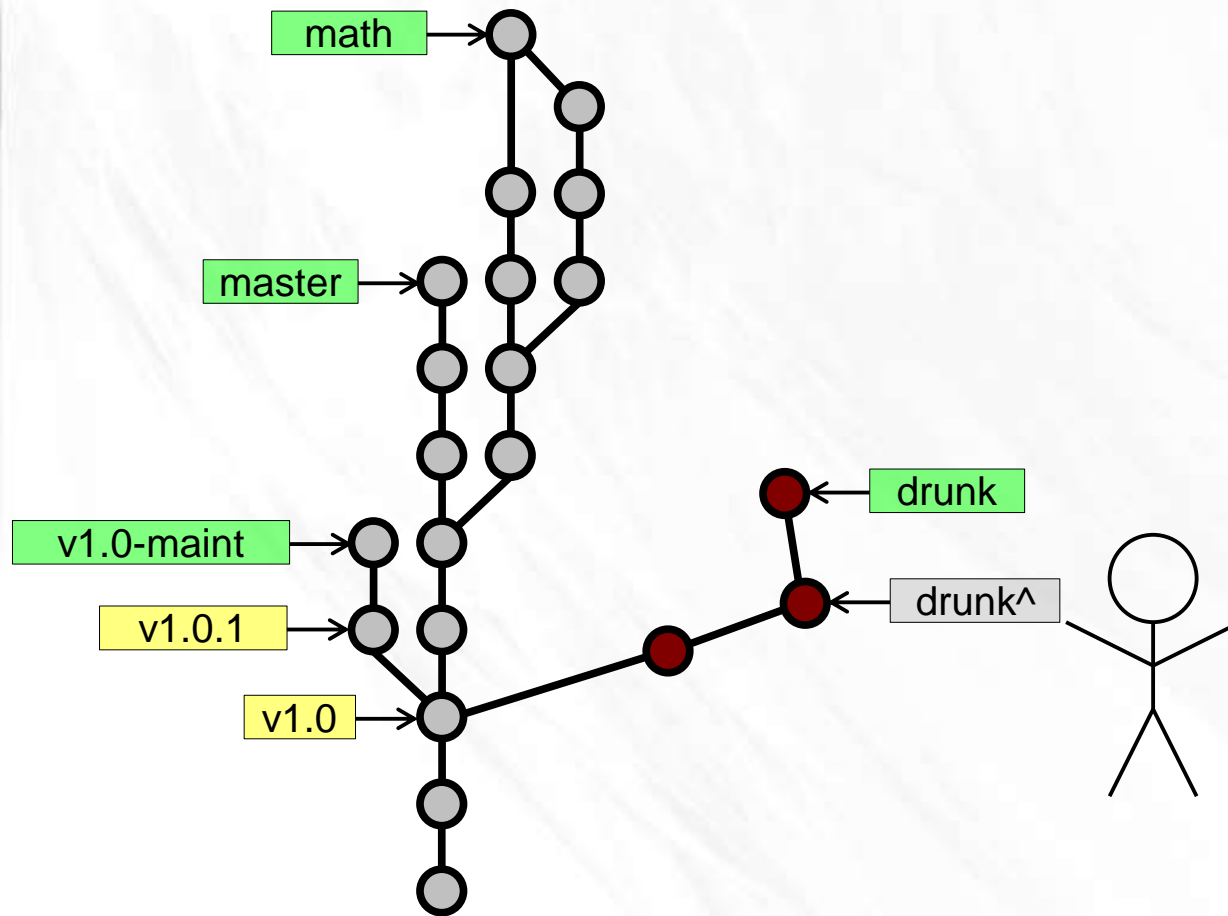
# Rewriting History



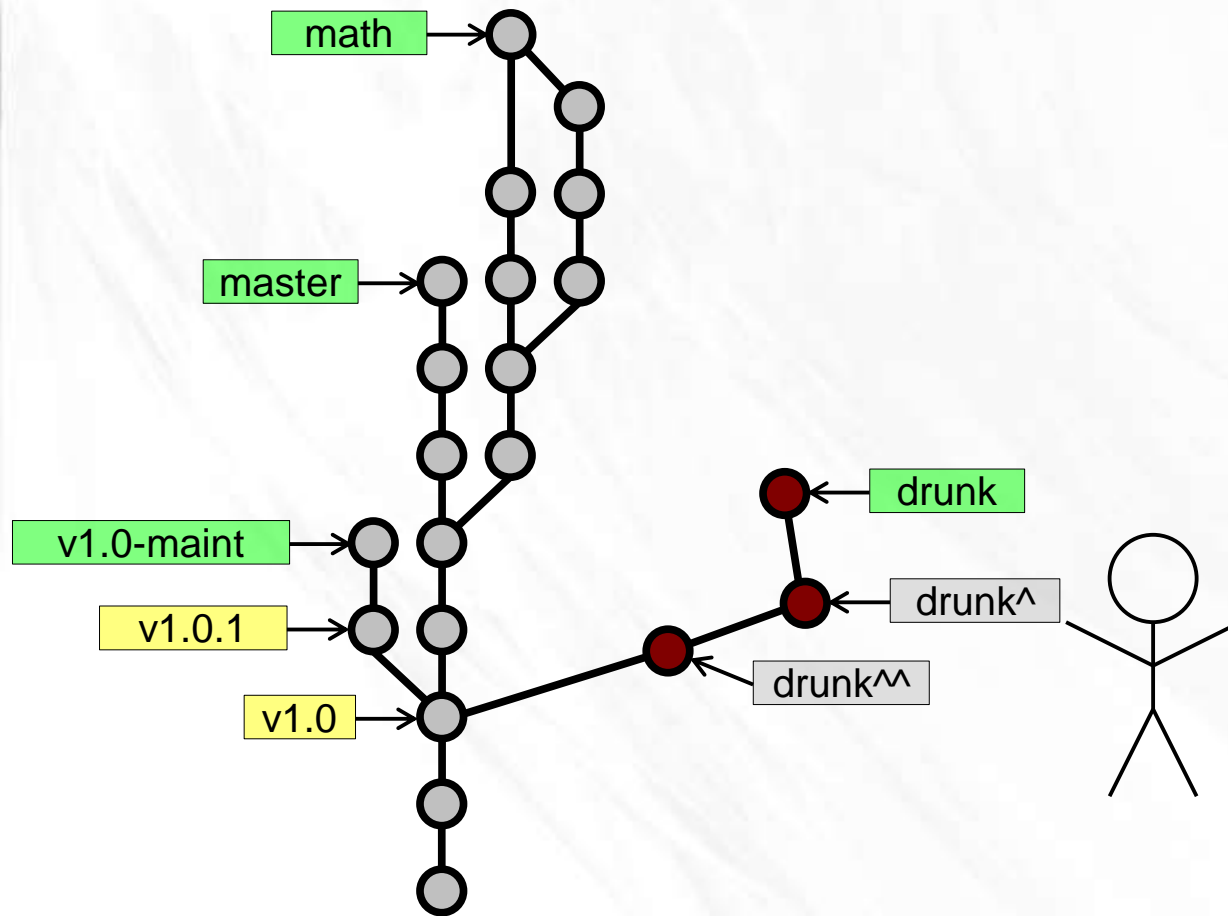
# Rewriting History



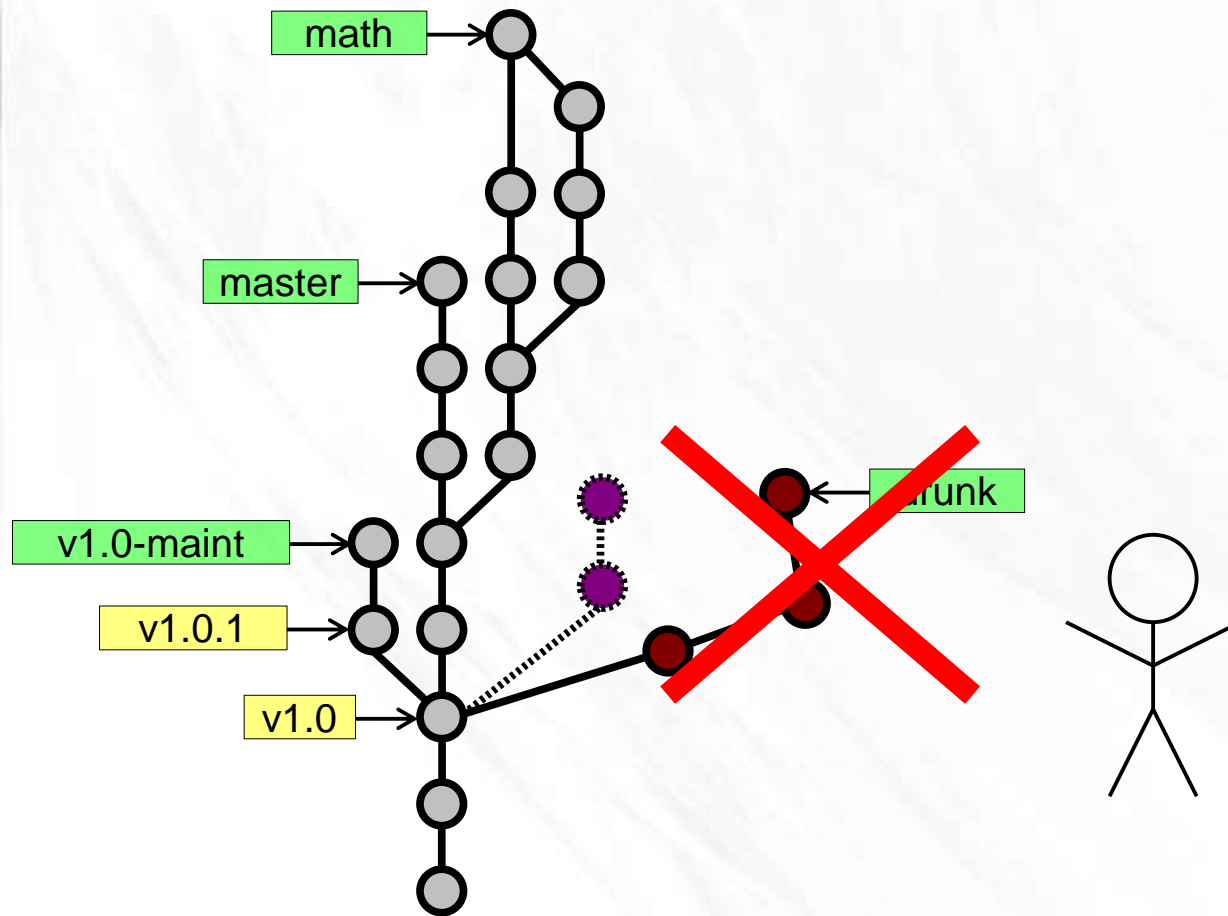
# Rewriting History



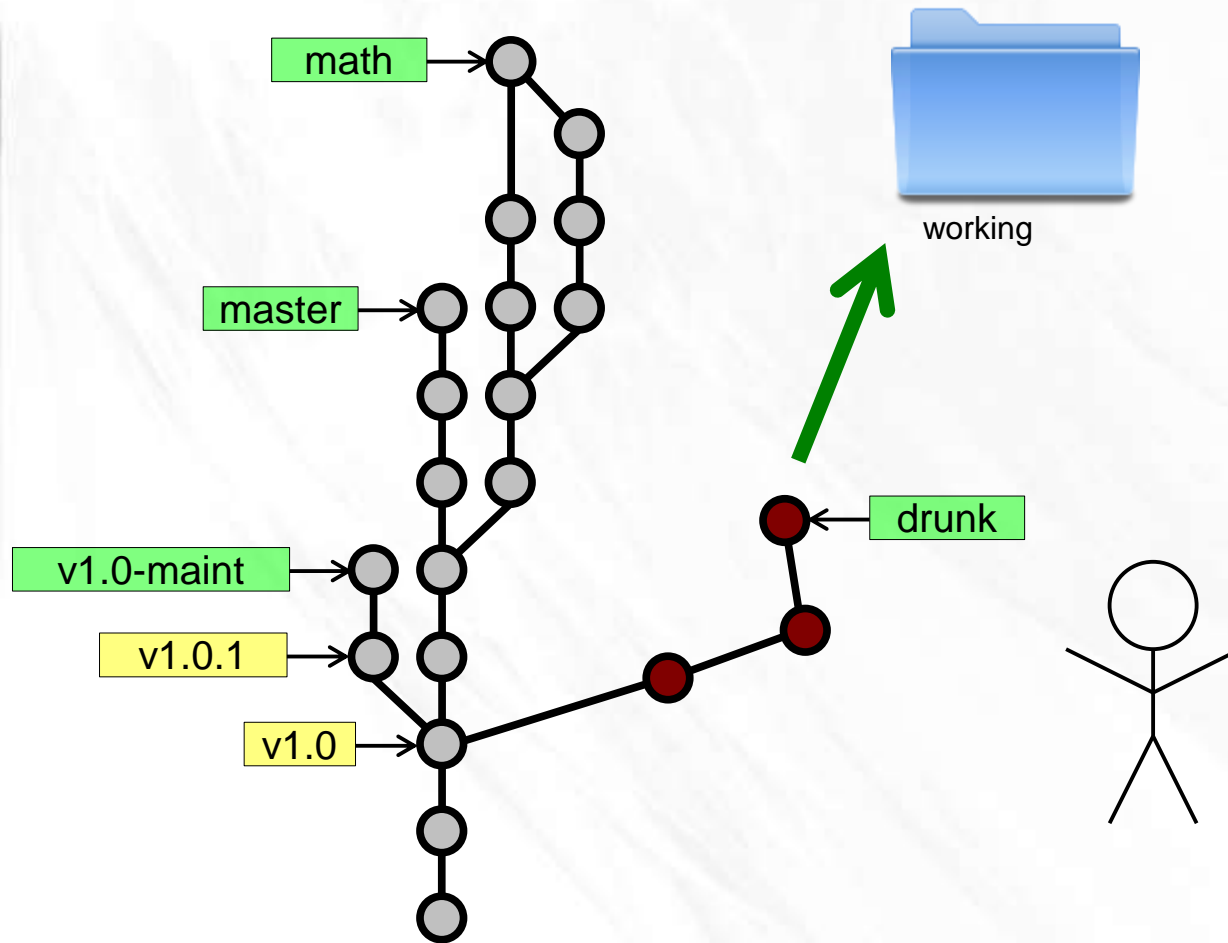
# Rewriting History



# Rewriting History

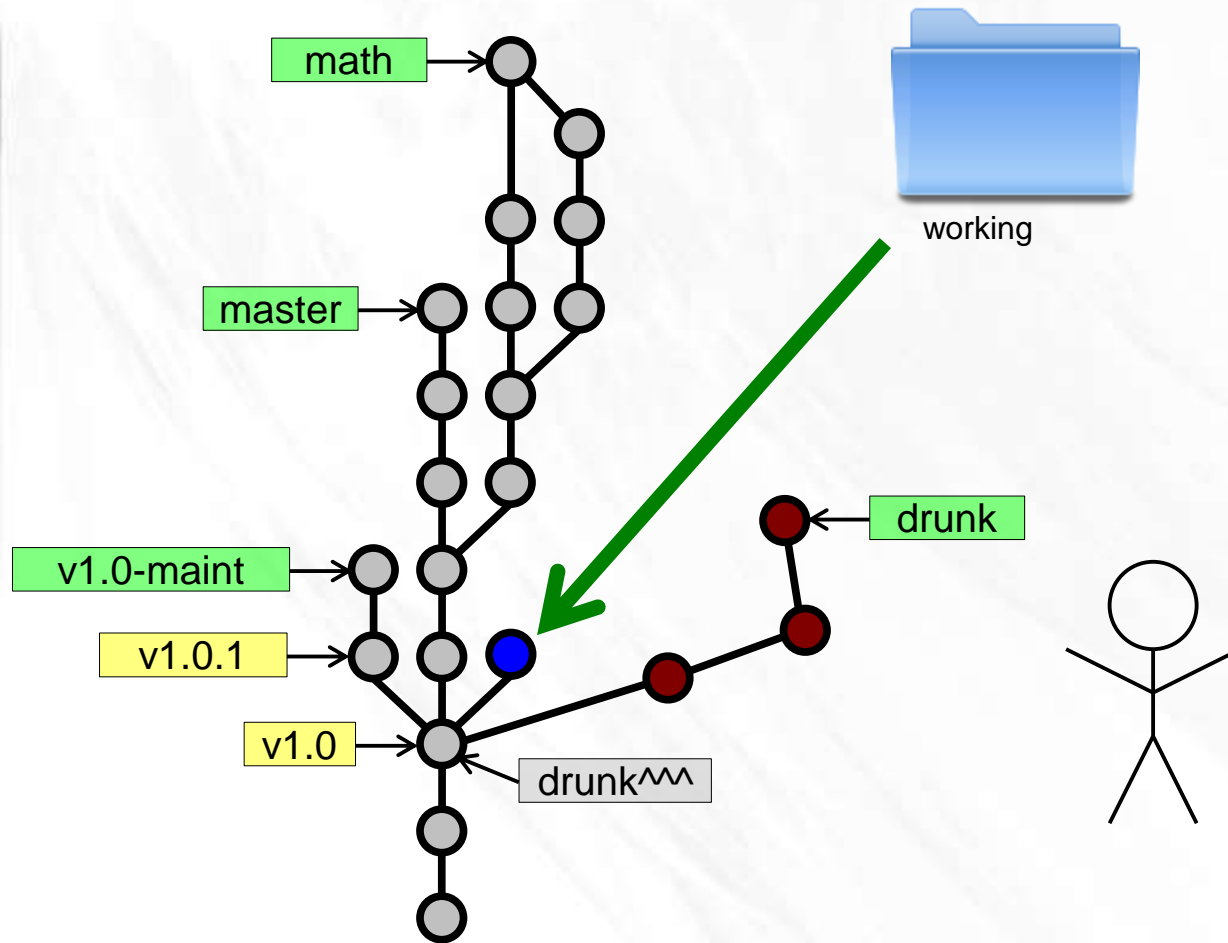


# Rewriting History

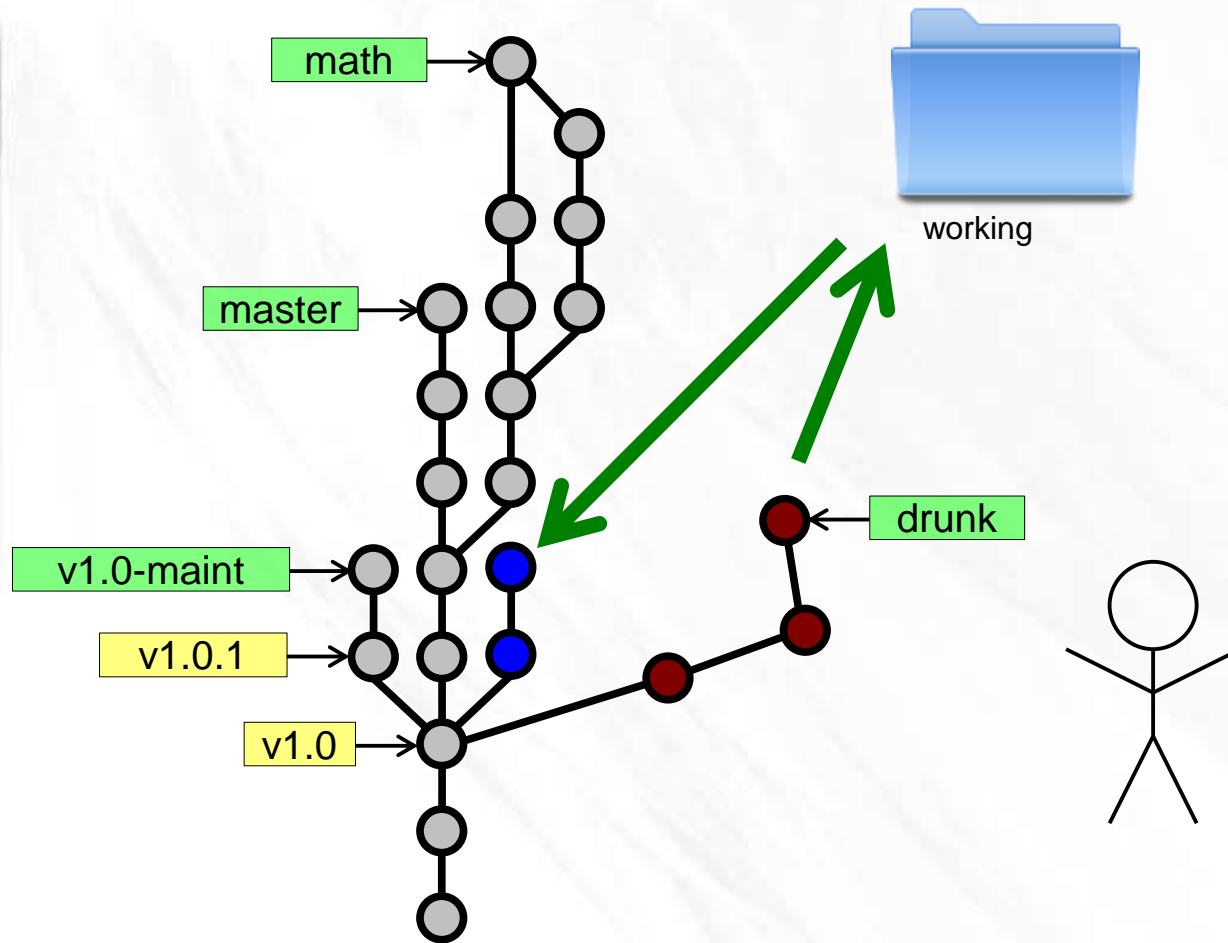




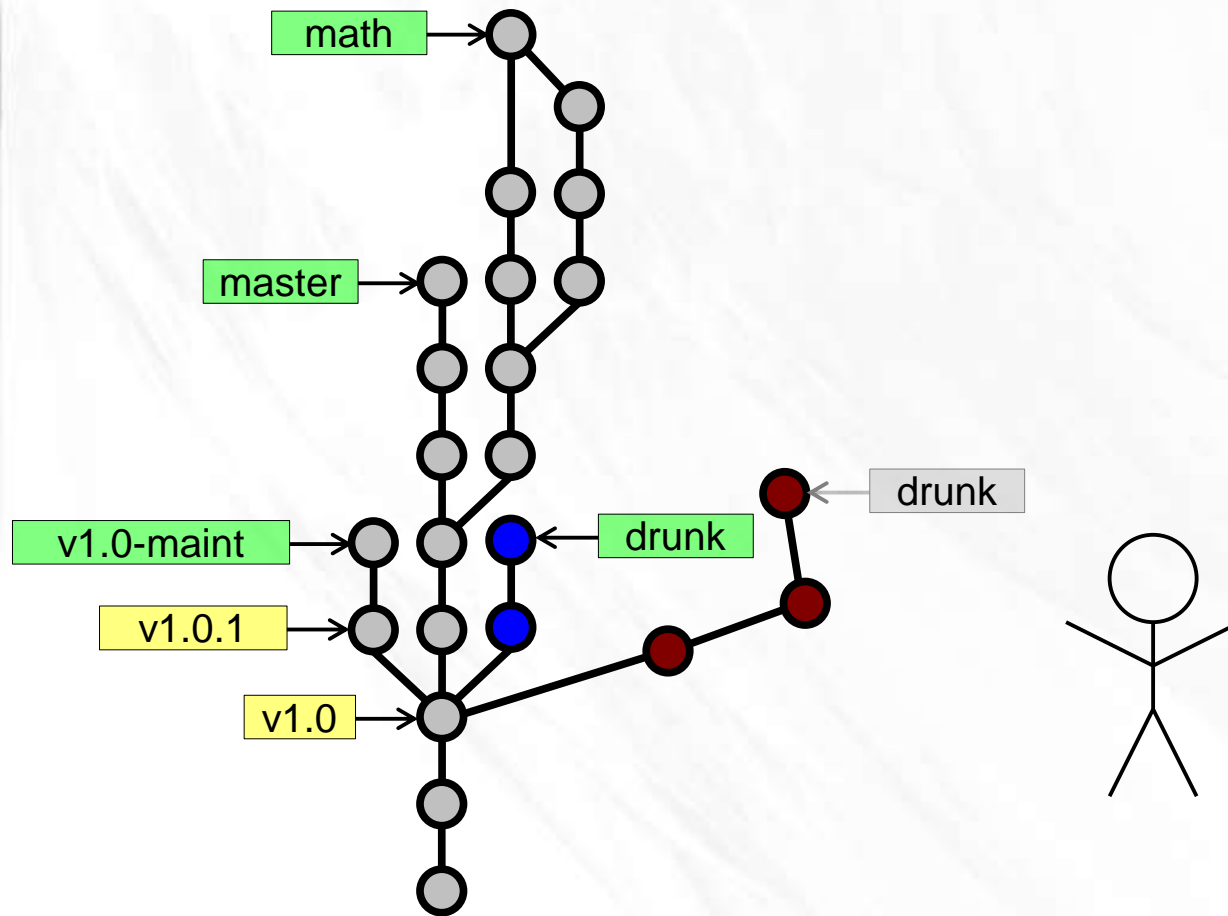
# Rewriting History



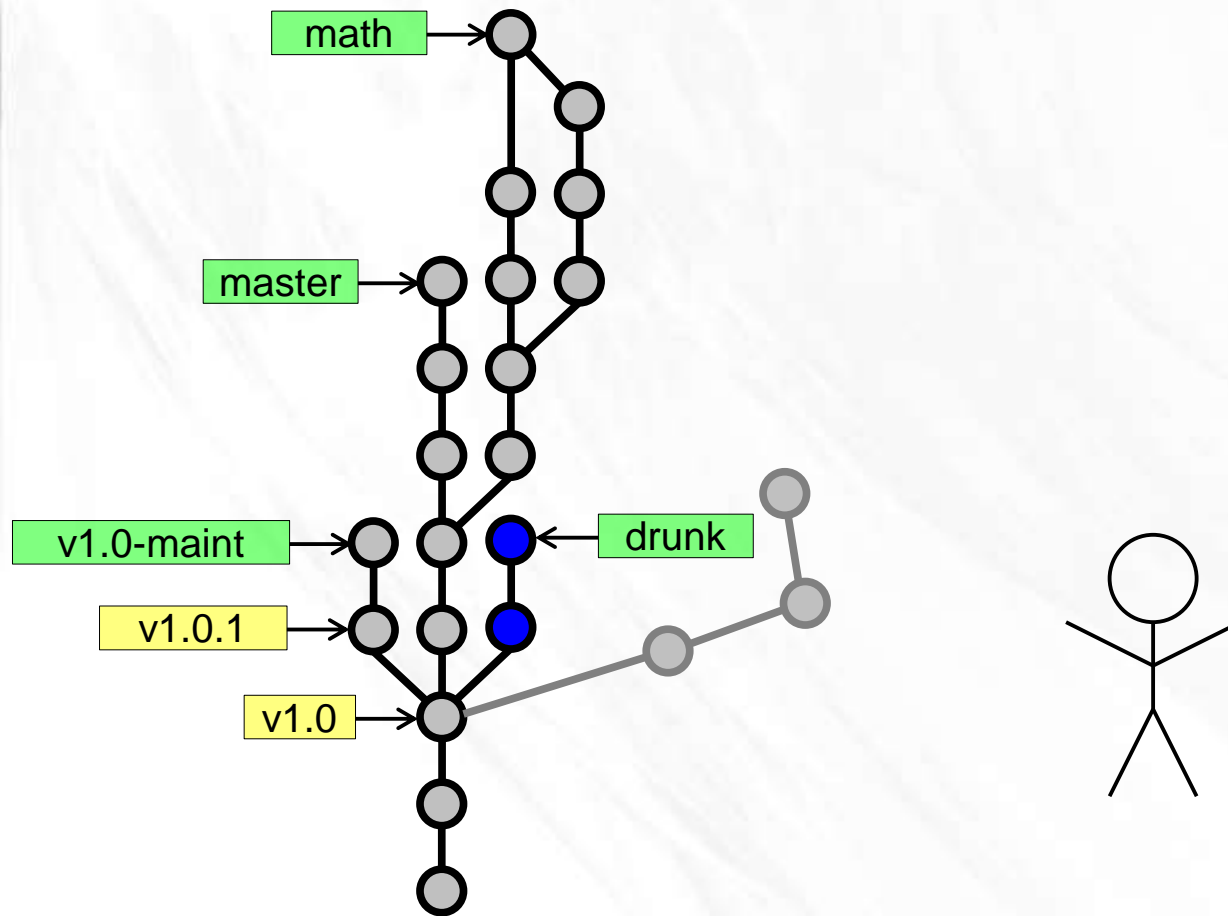
# Rewriting History



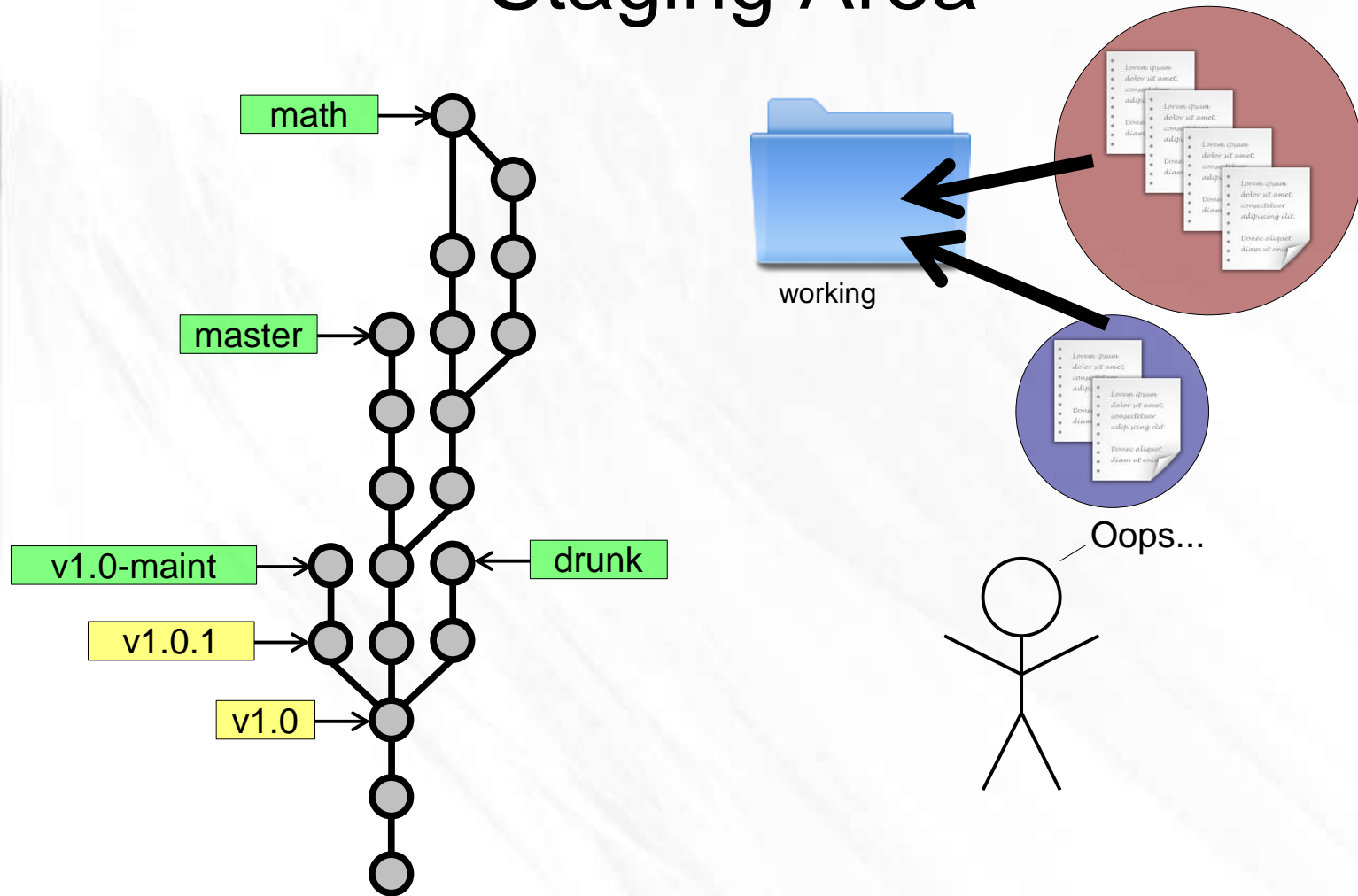
# Rewriting History



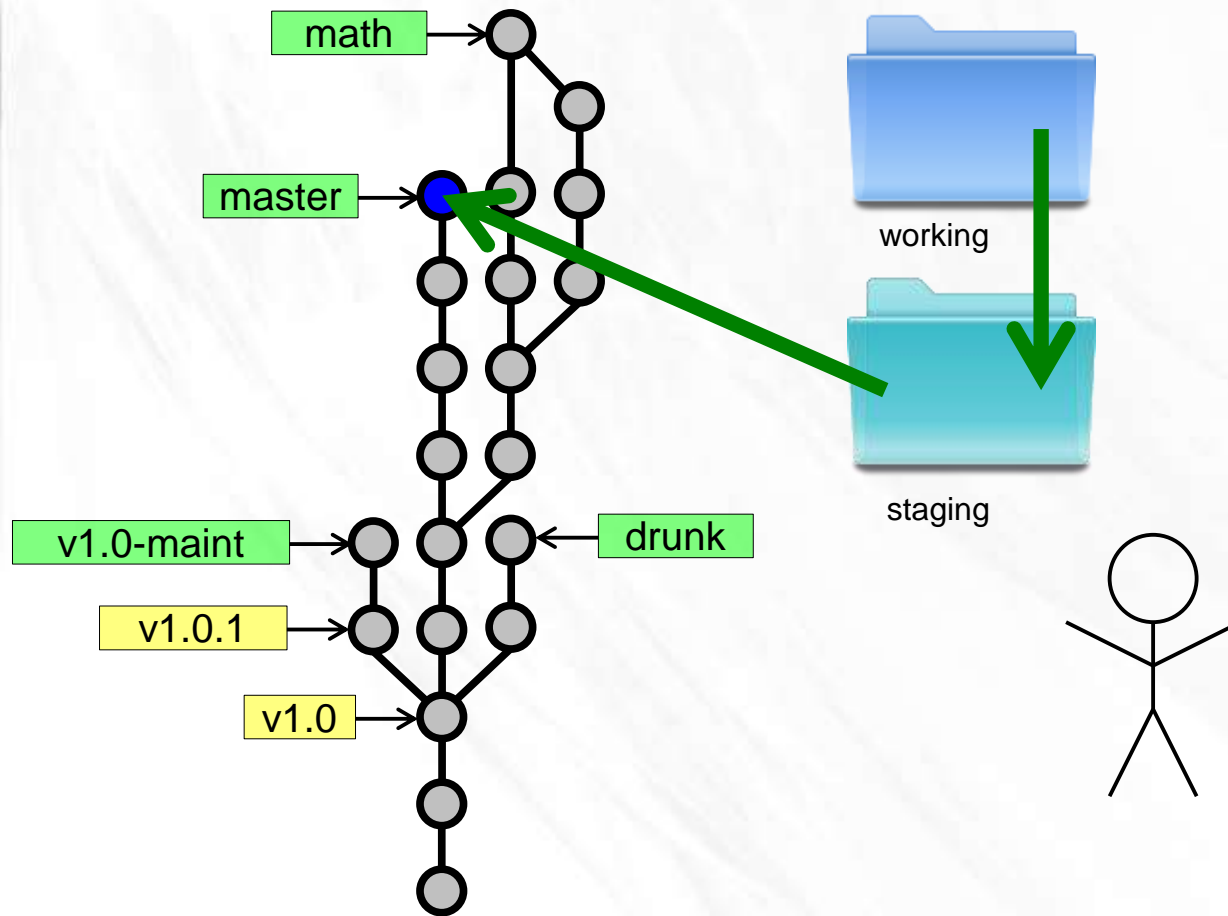
# Rewriting History



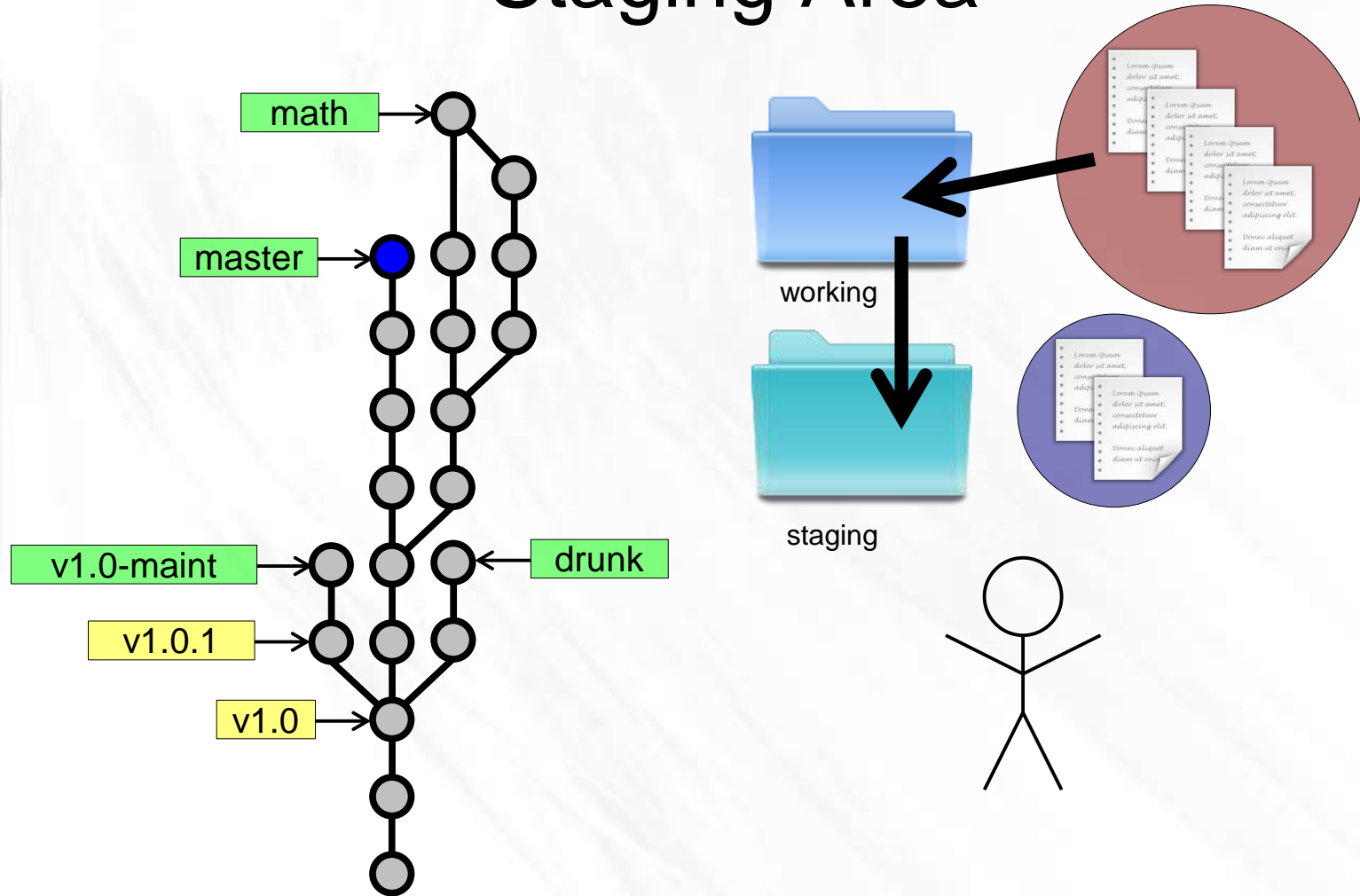
# Staging Area



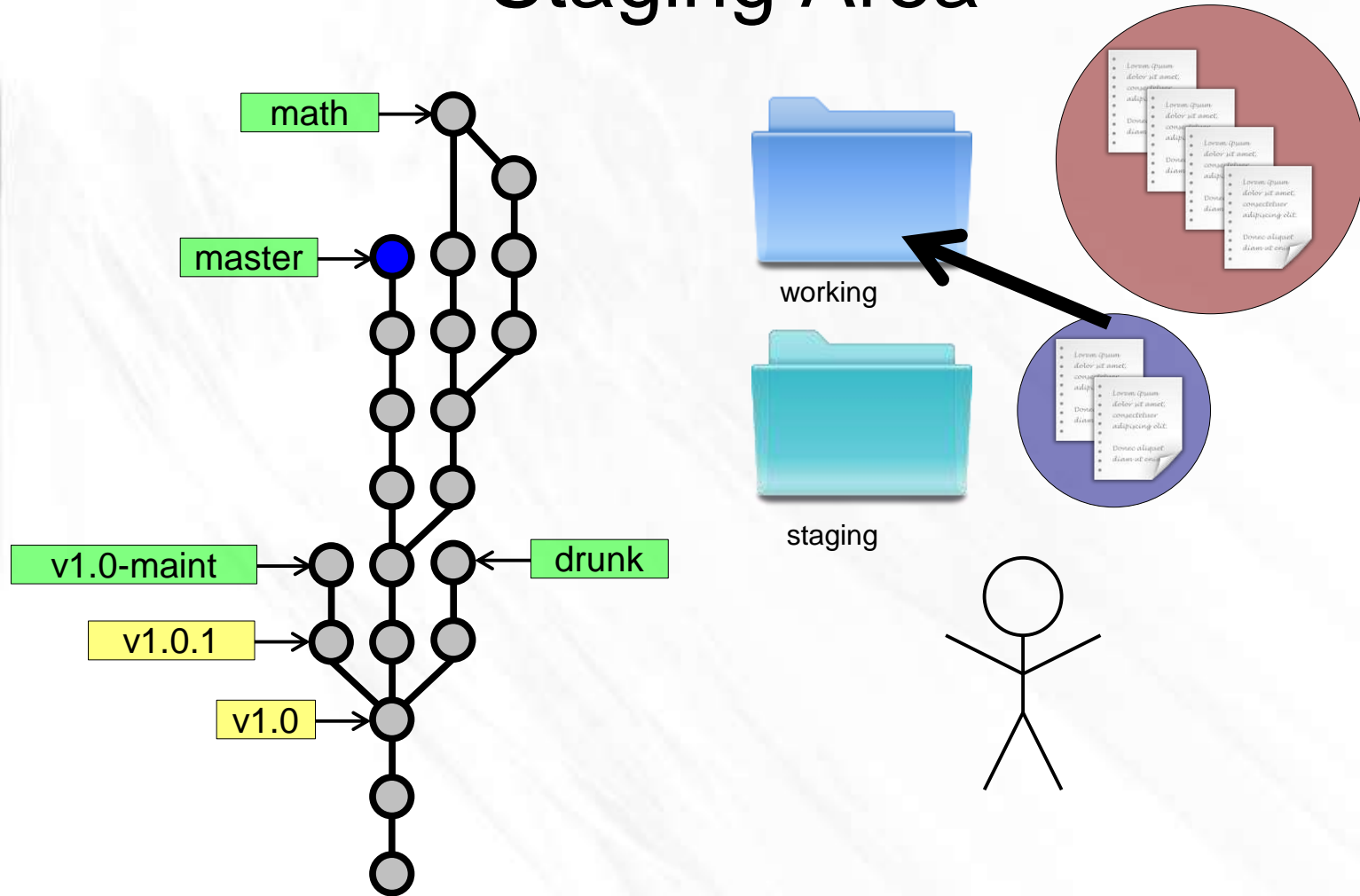
# Staging Area



# Staging Area

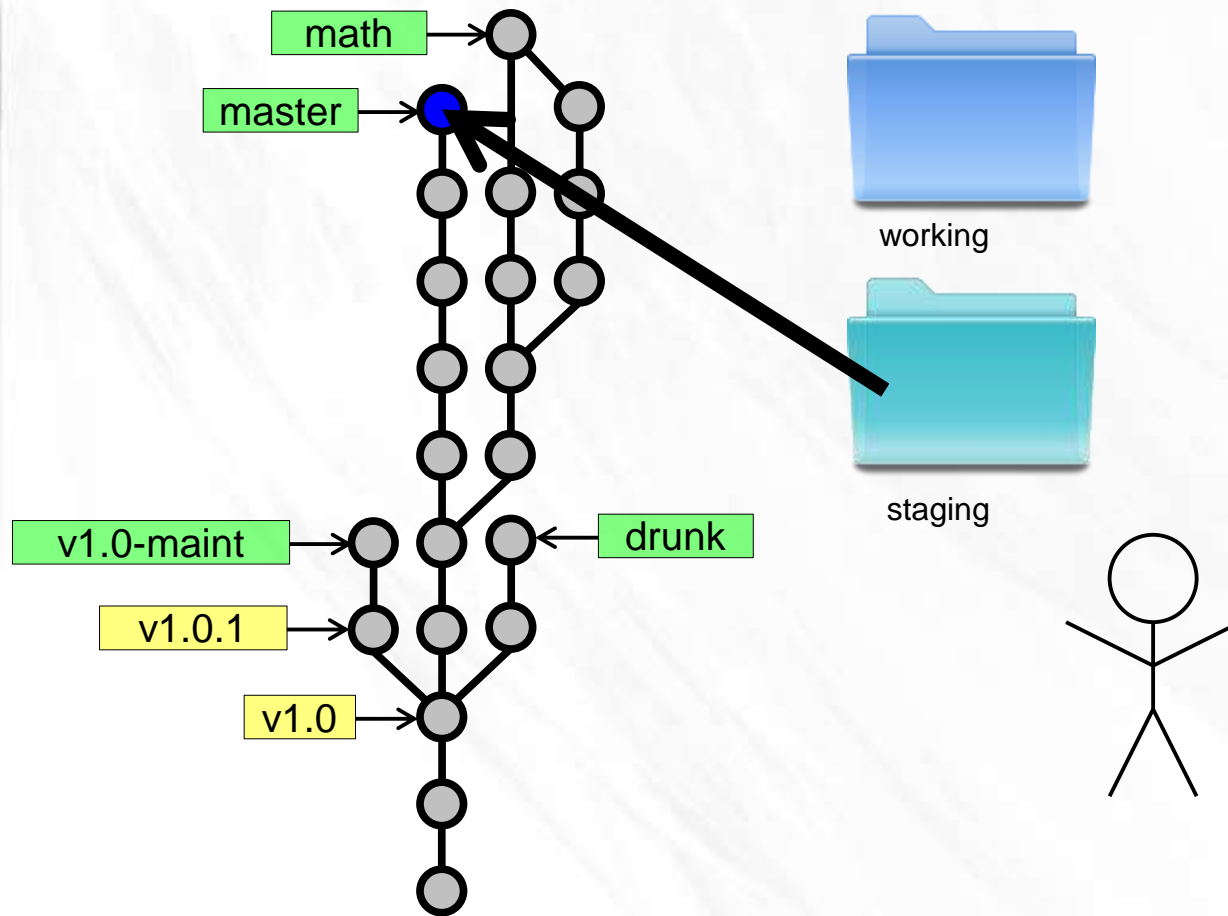


# Staging Area

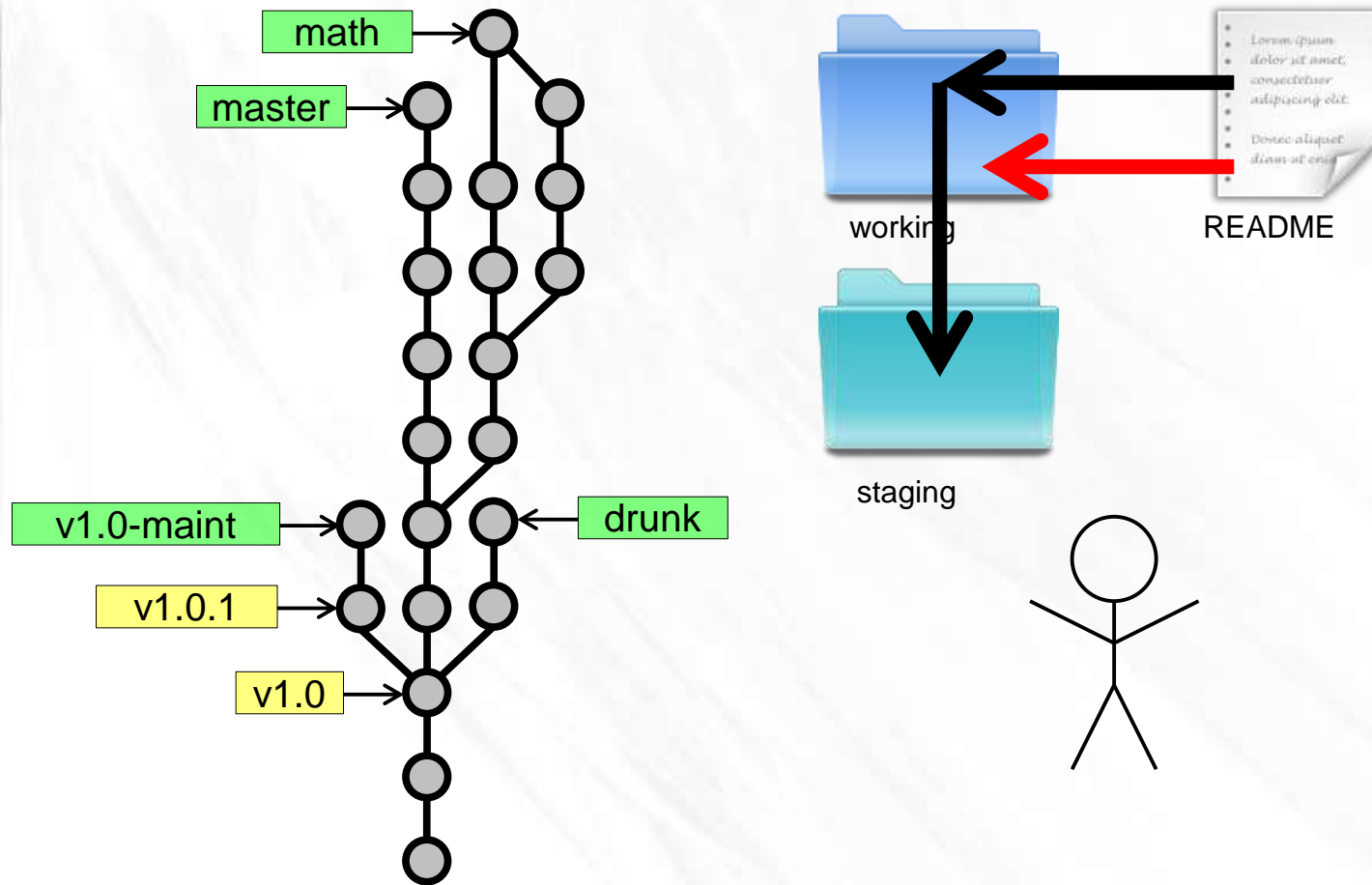




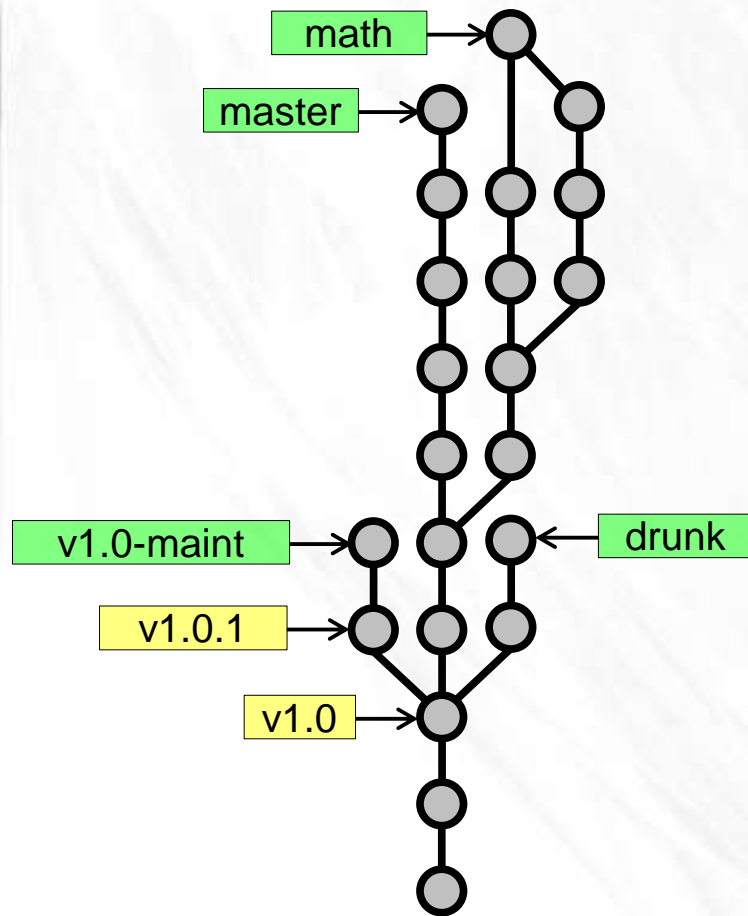
# Staging Area



# Staging Area



# Diffs



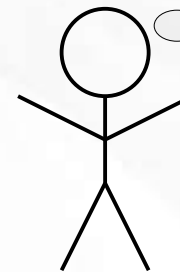
working



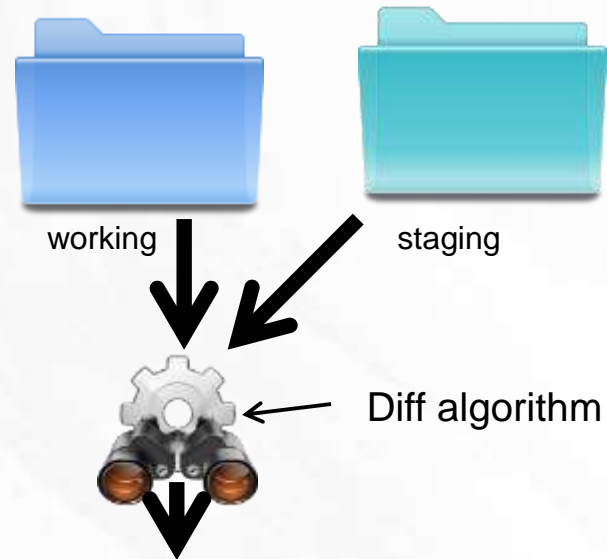
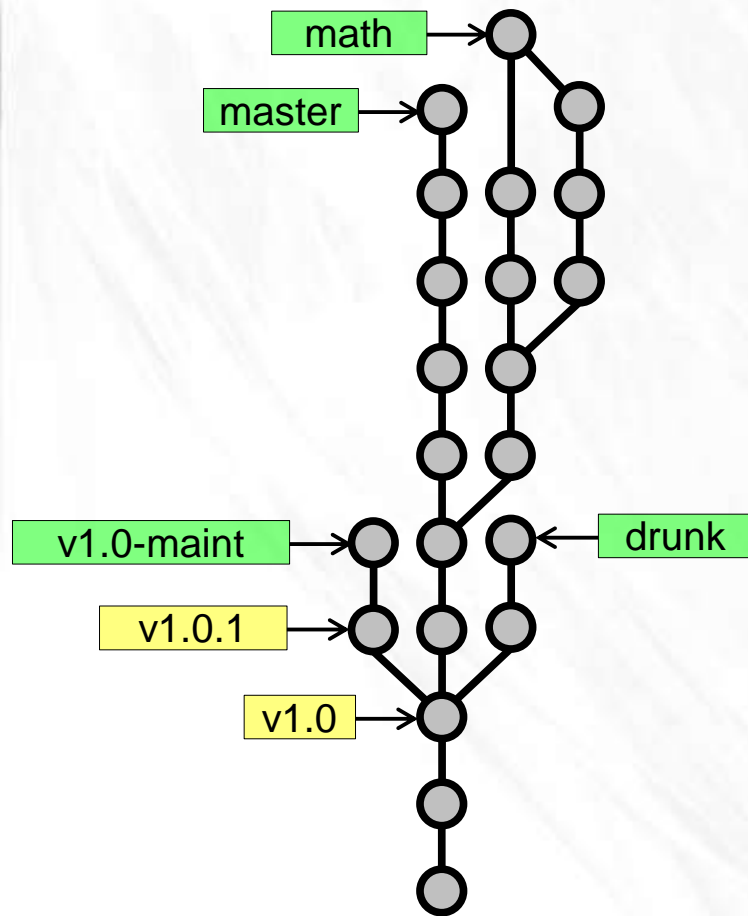
staging

What are the changes?

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

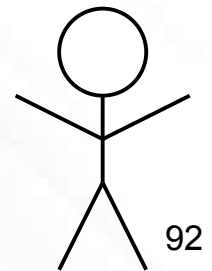


# Diffs

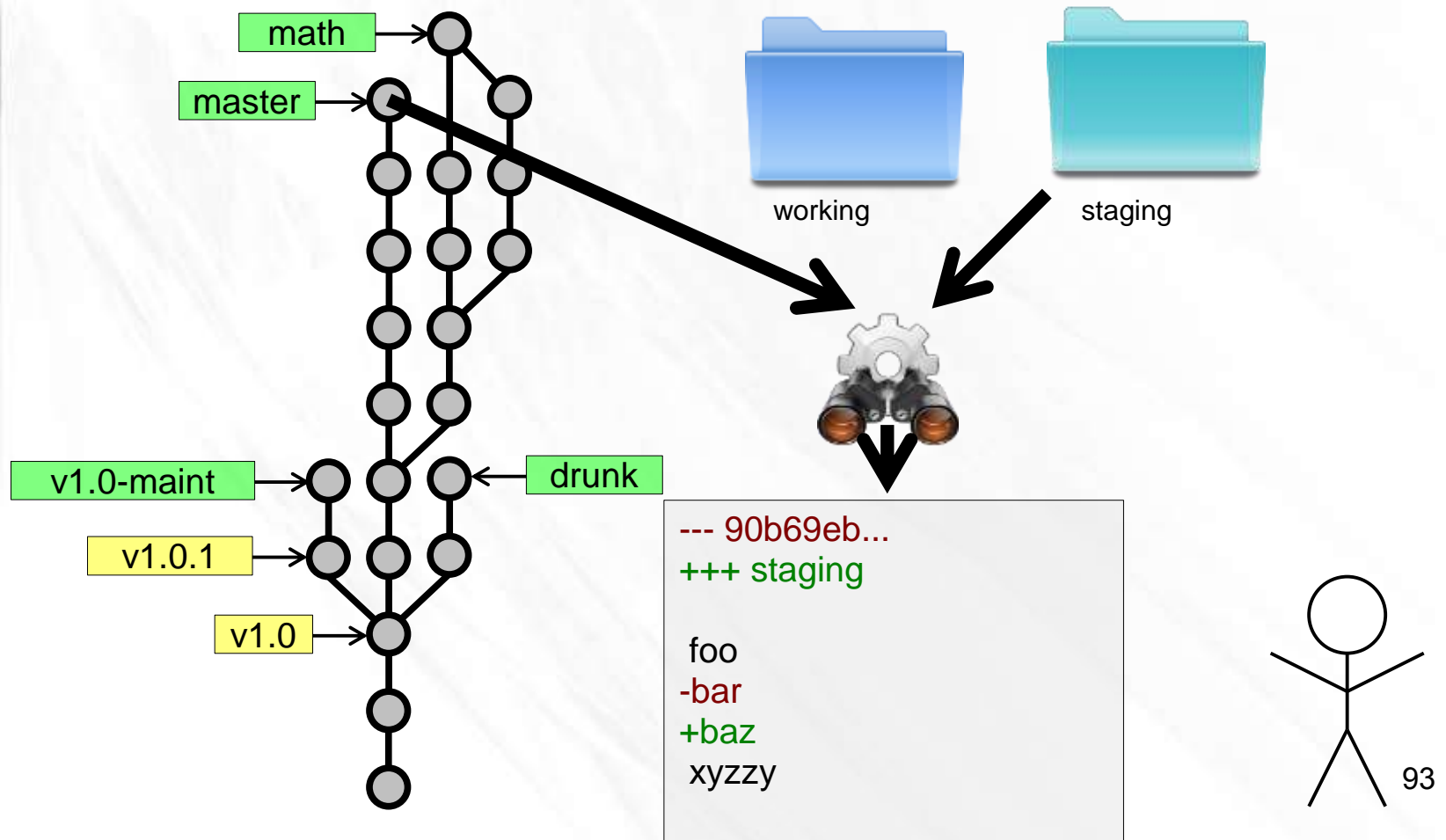


--- staging  
+++ working

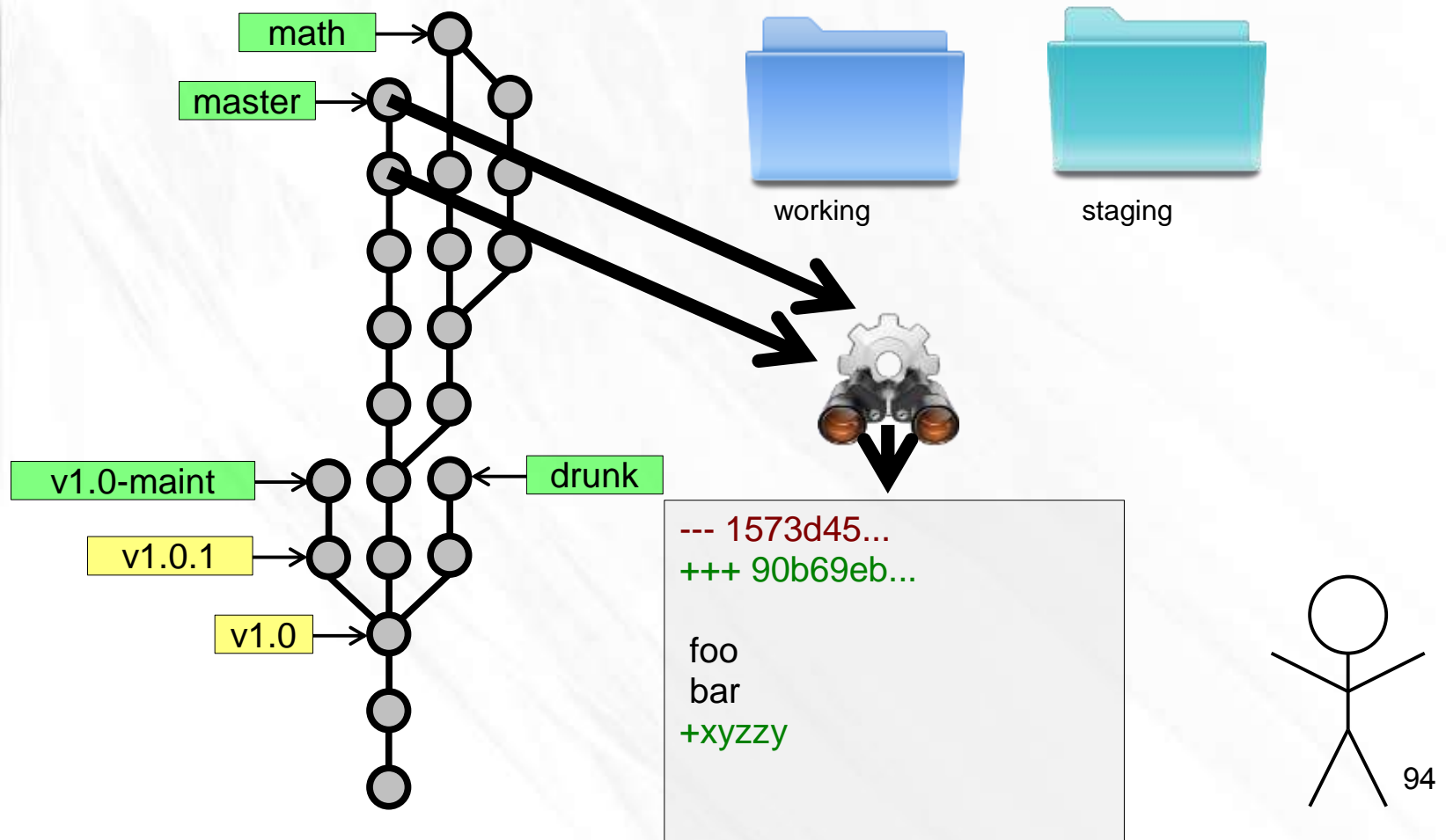
foo  
-baz  
+bazzle  
xyzyy



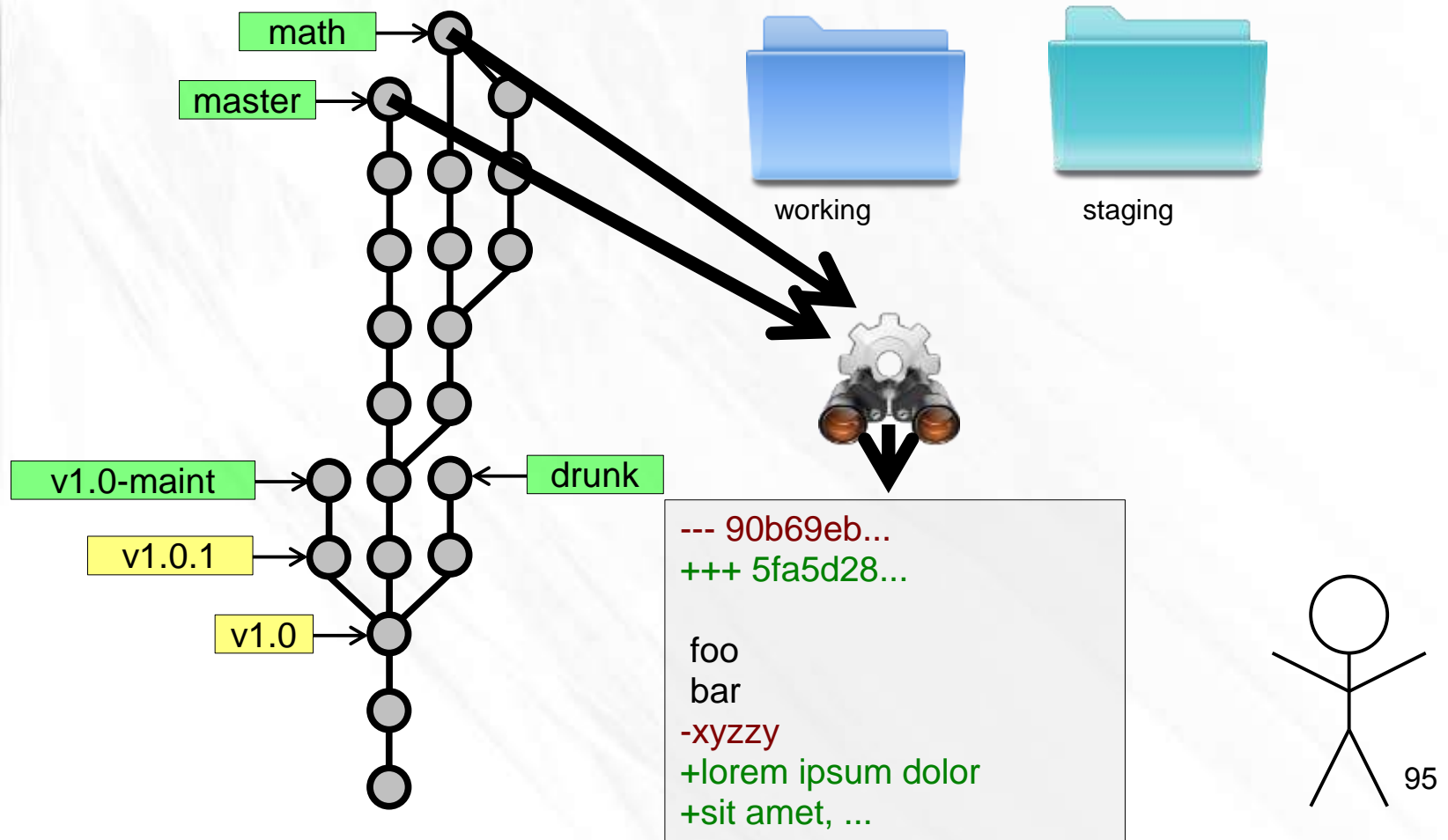
# Diffs



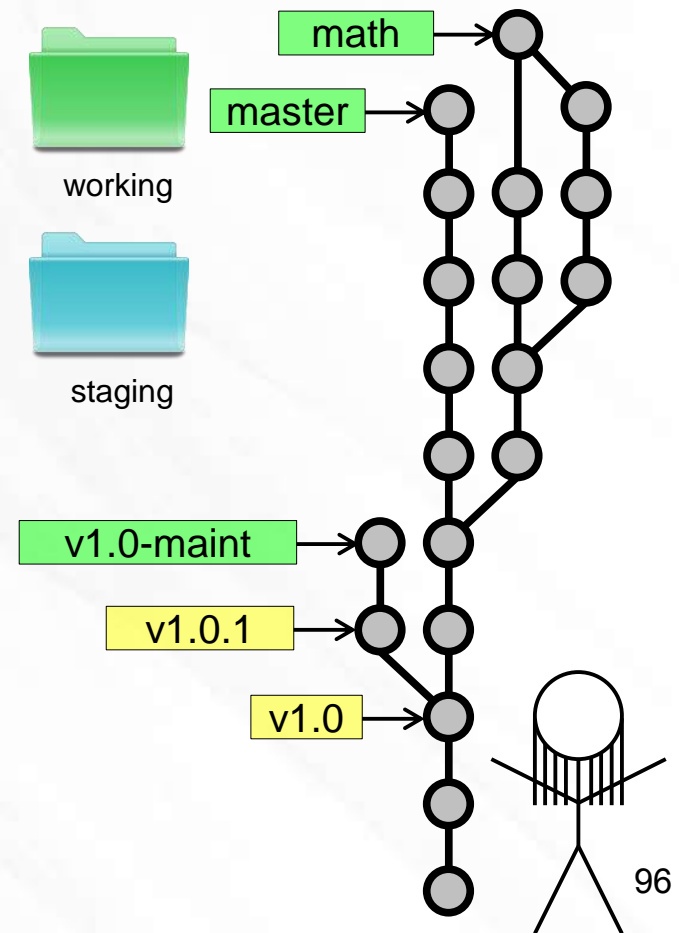
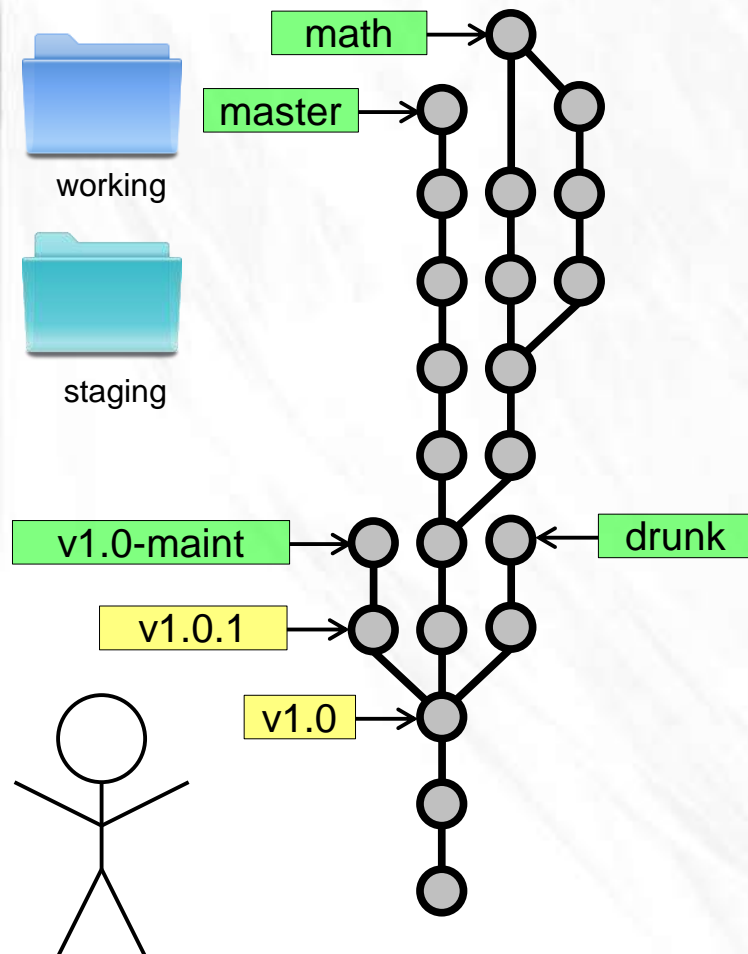
# Diffs



# Diffs

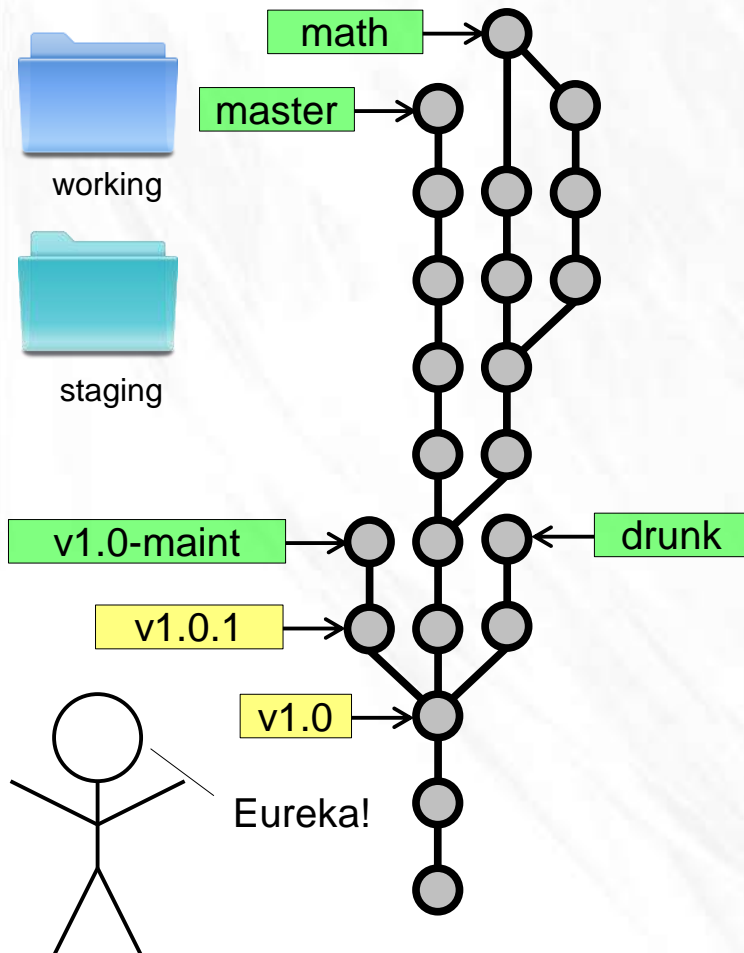


# Eliminating Duplication





# Eliminating Duplication



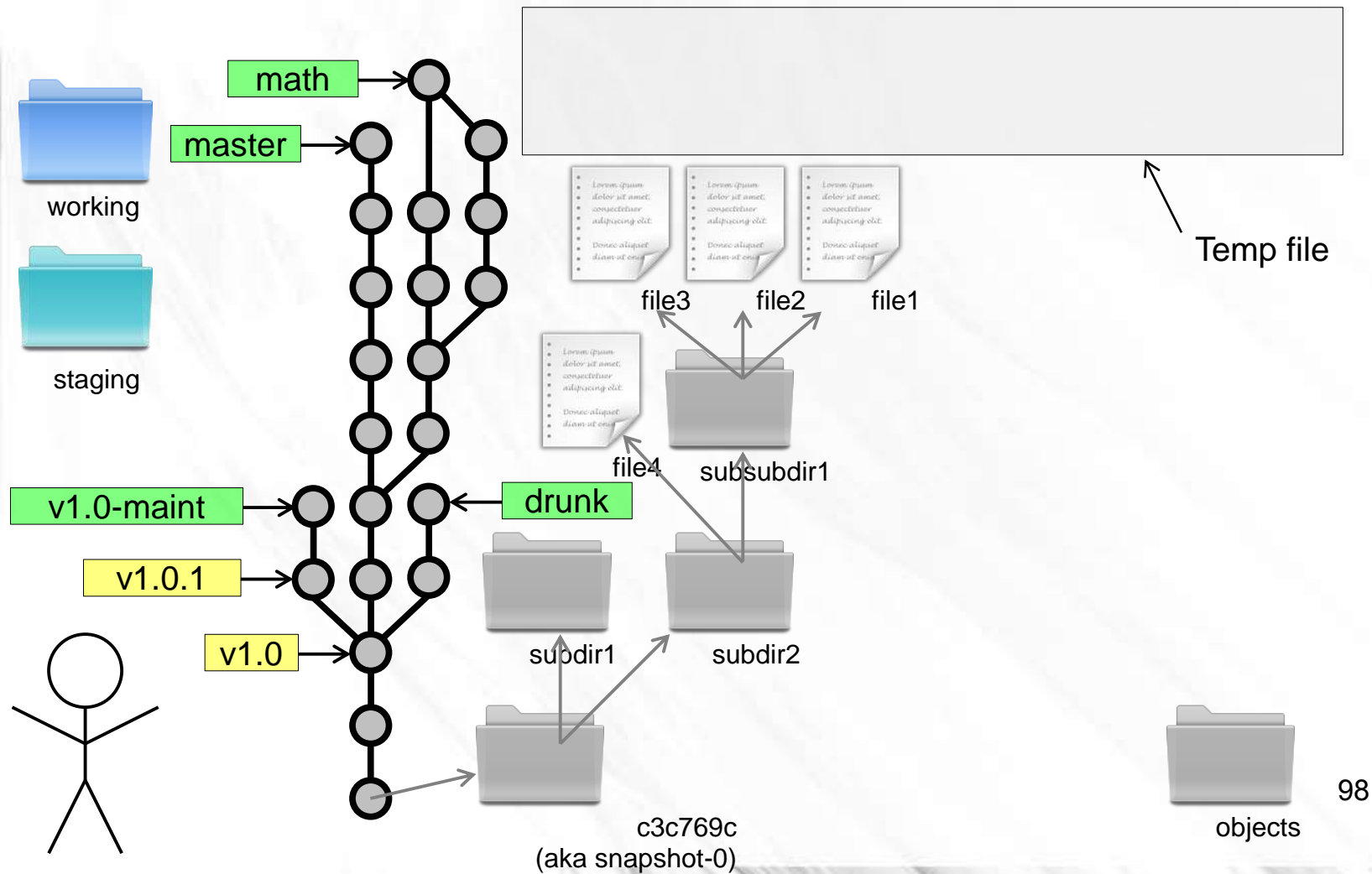
SHA1

2804133755c3ed396d162028c7b30a1cbcfecded

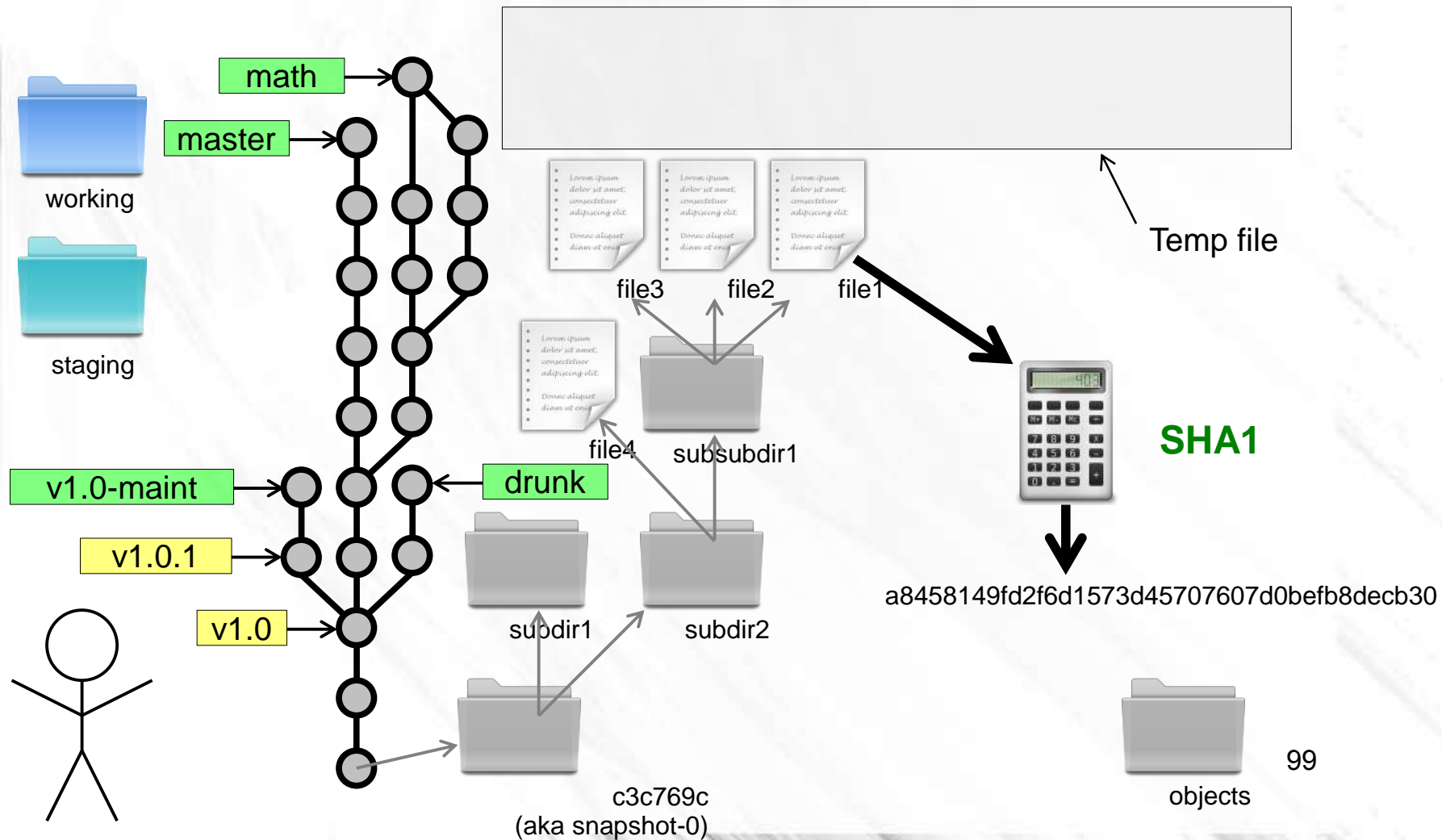


objects

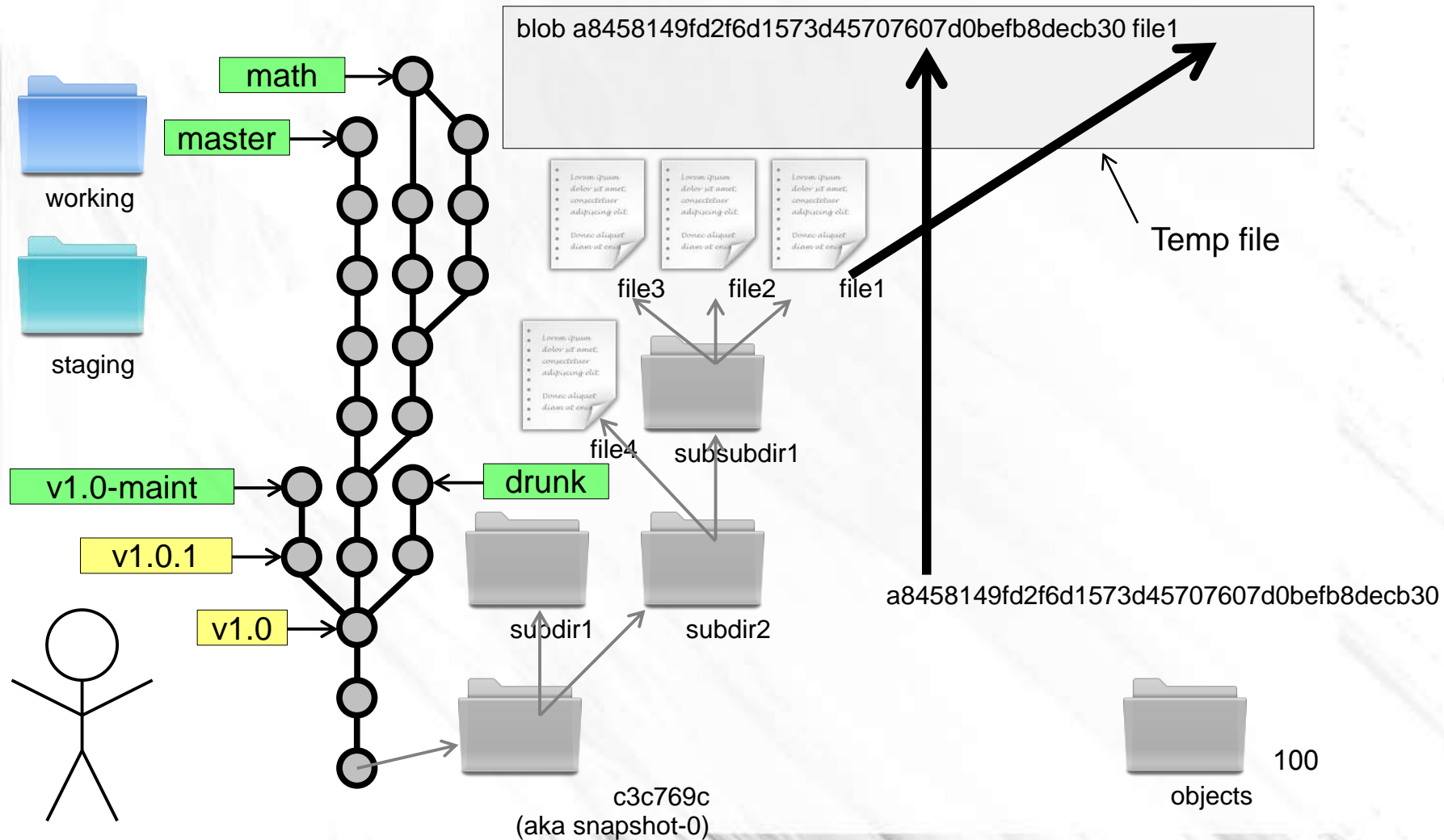
# Eliminating Duplication



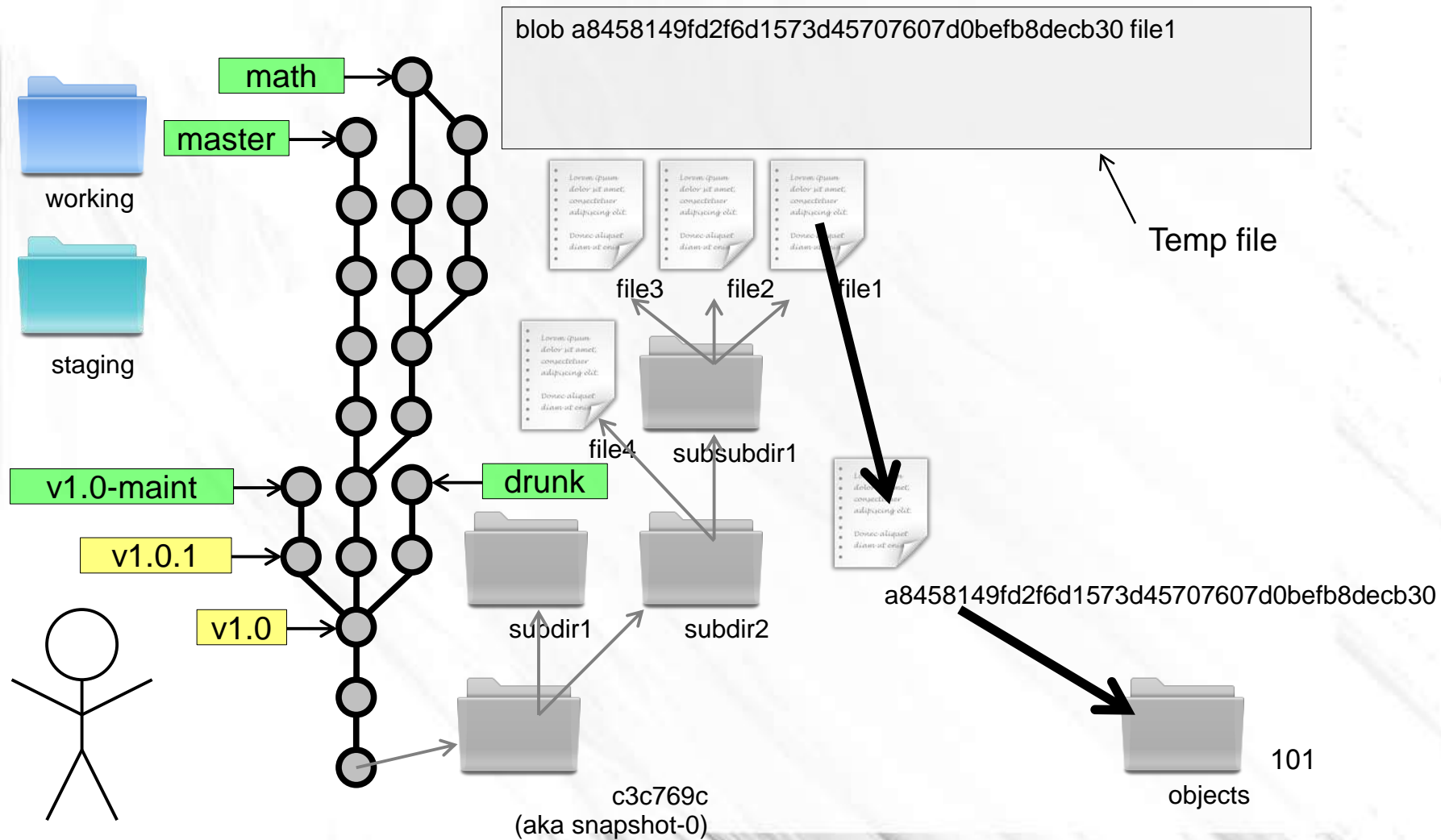
# Eliminating Duplication



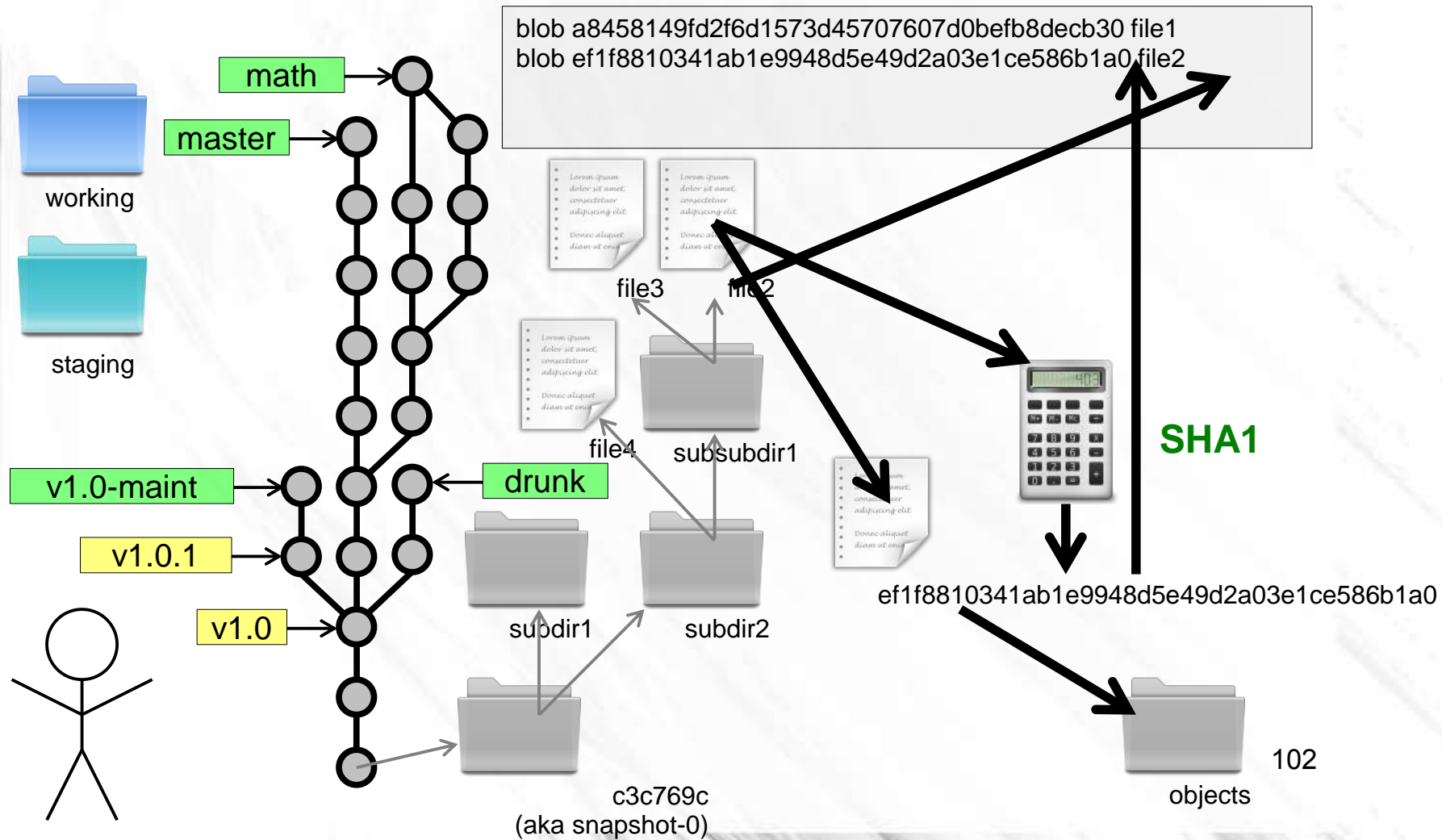
# Eliminating Duplication



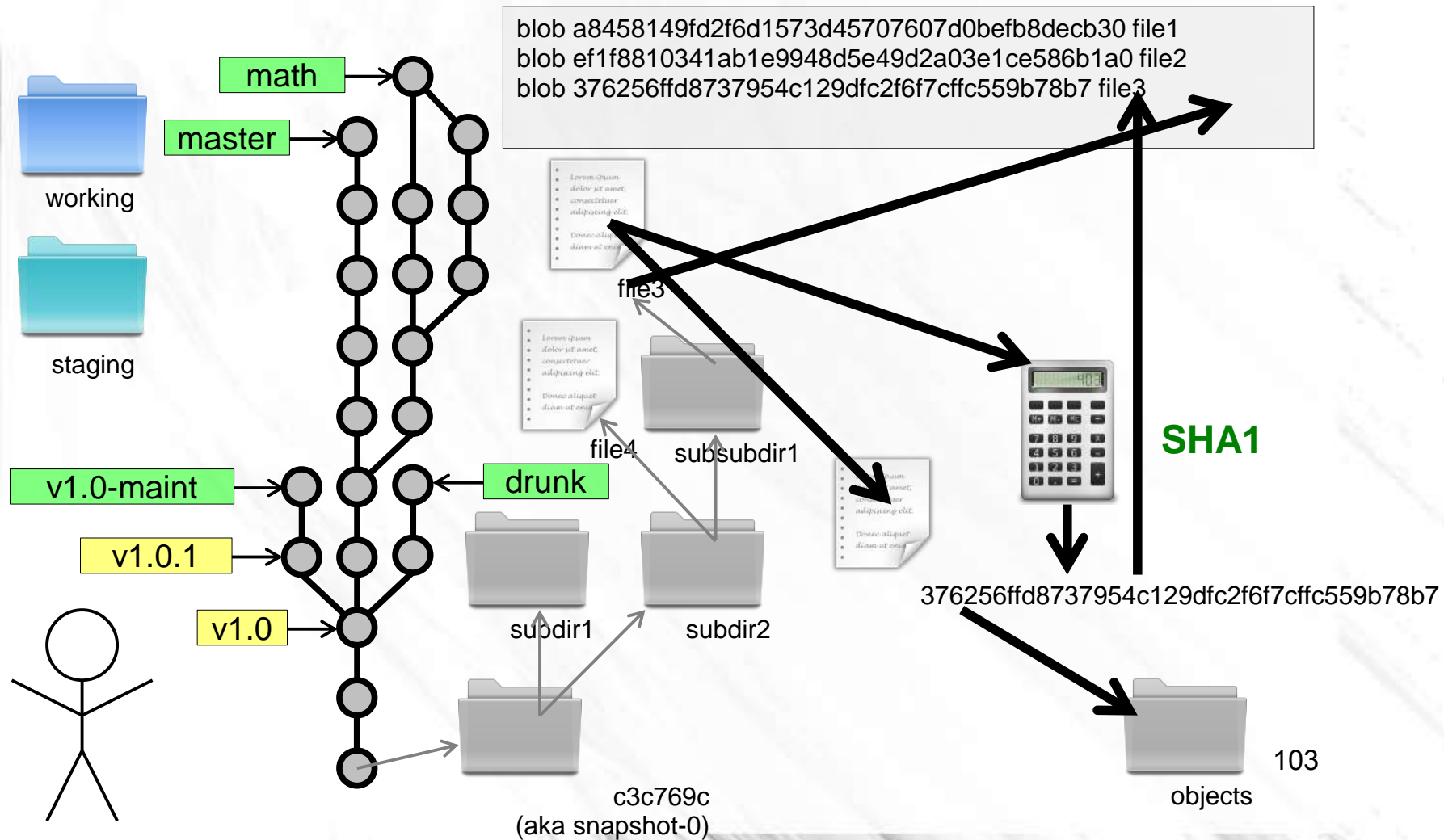
# Eliminating Duplication



# Eliminating Duplication

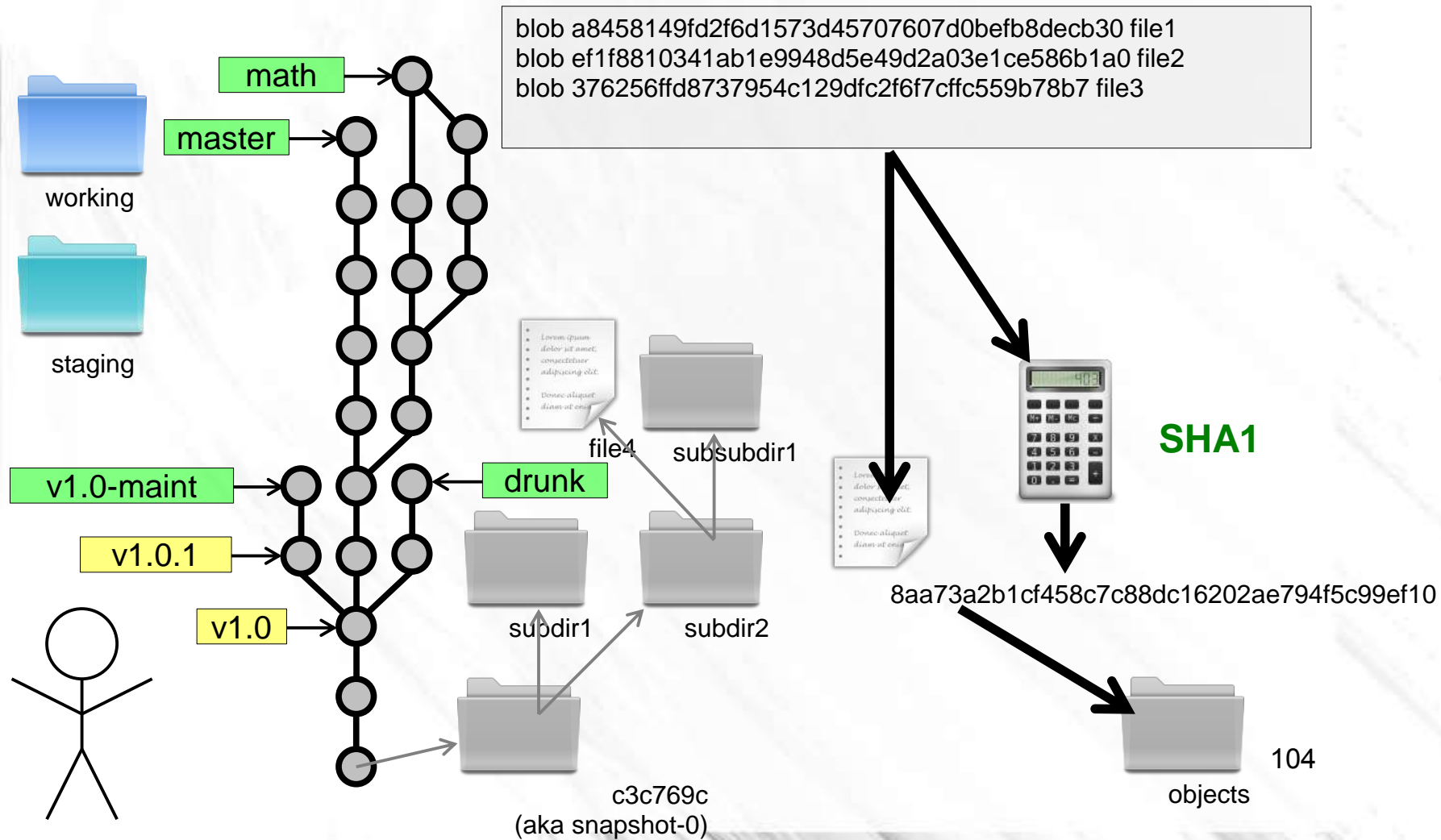


# Eliminating Duplication



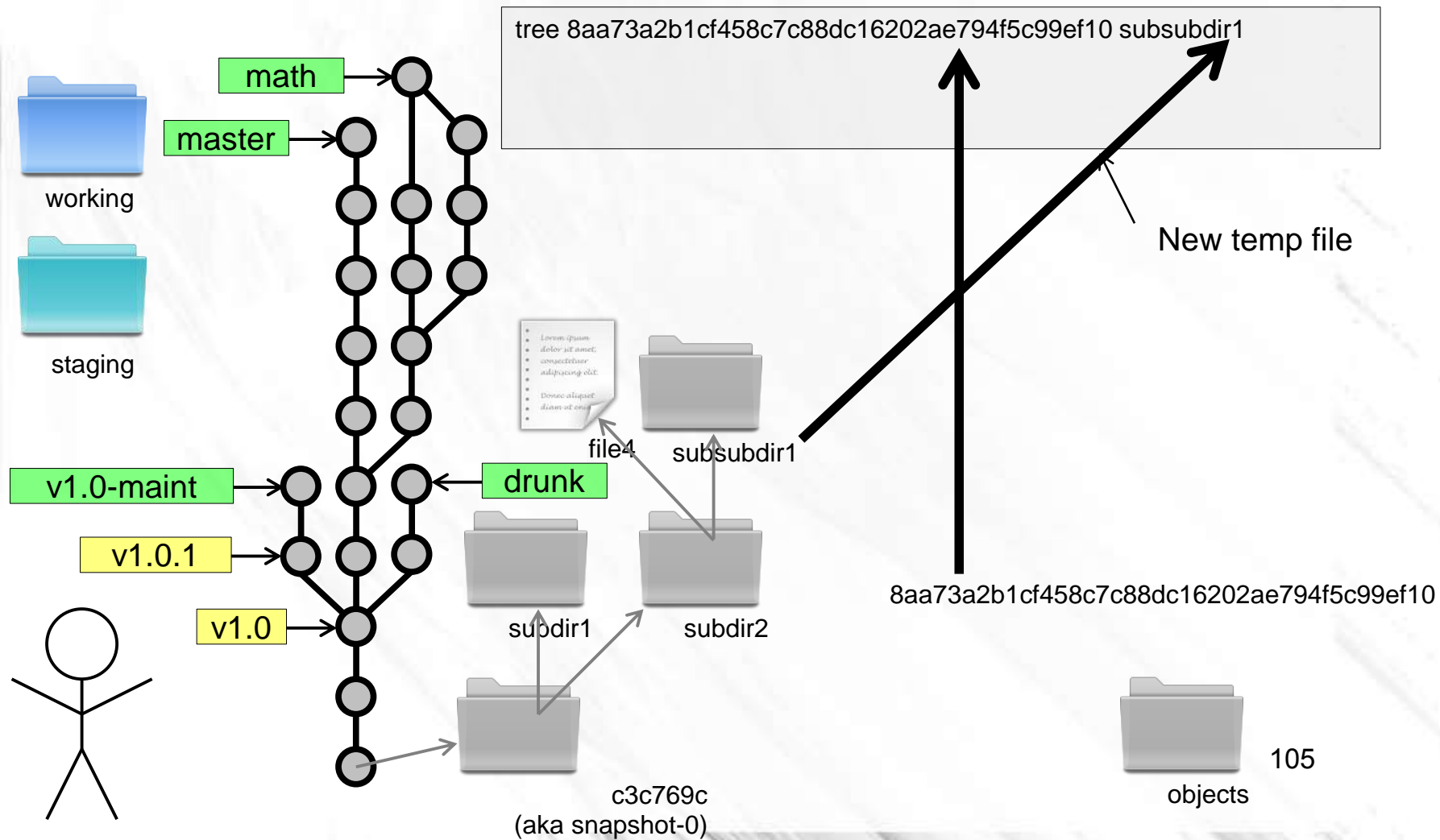


# Eliminating Duplication

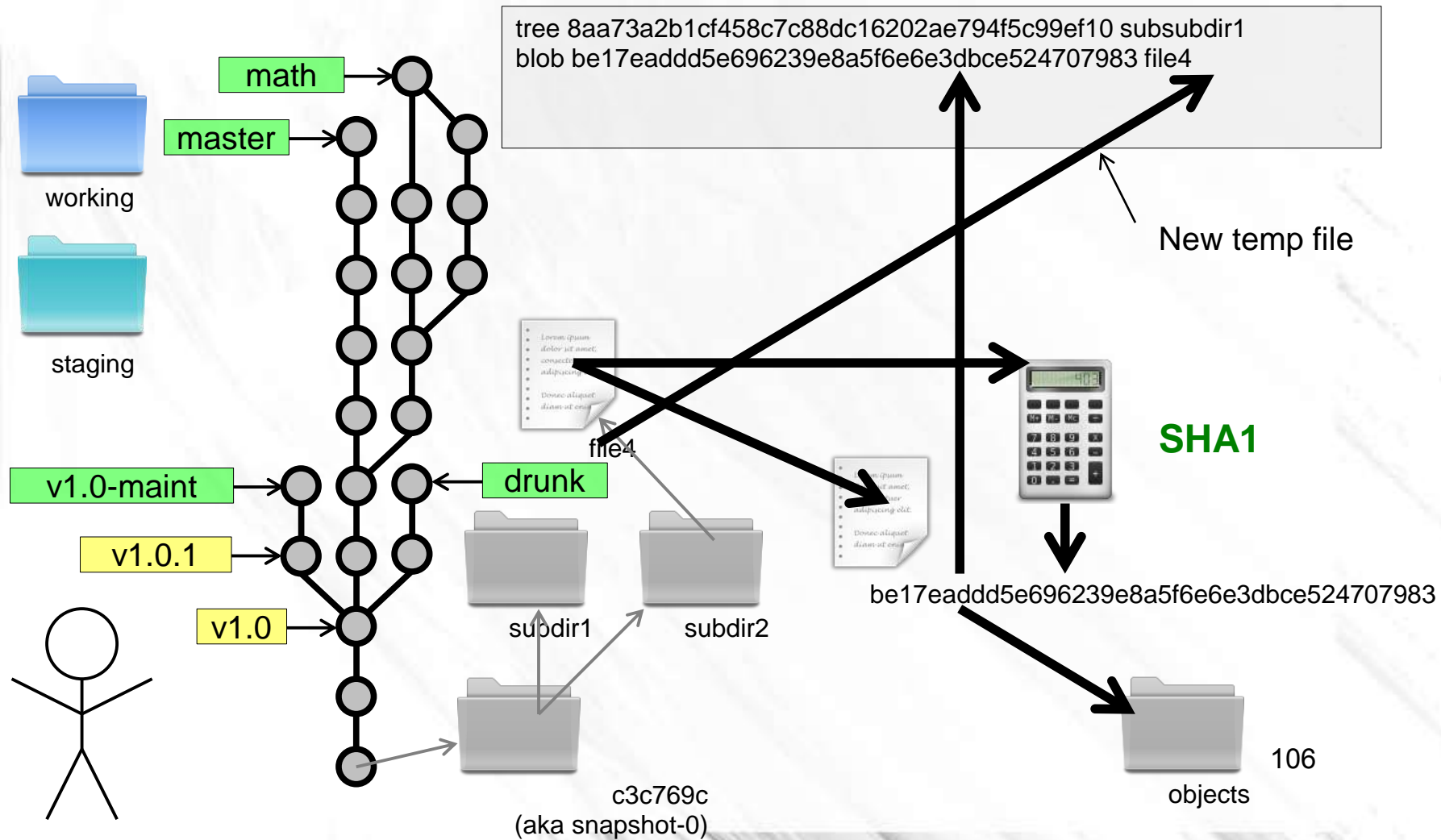




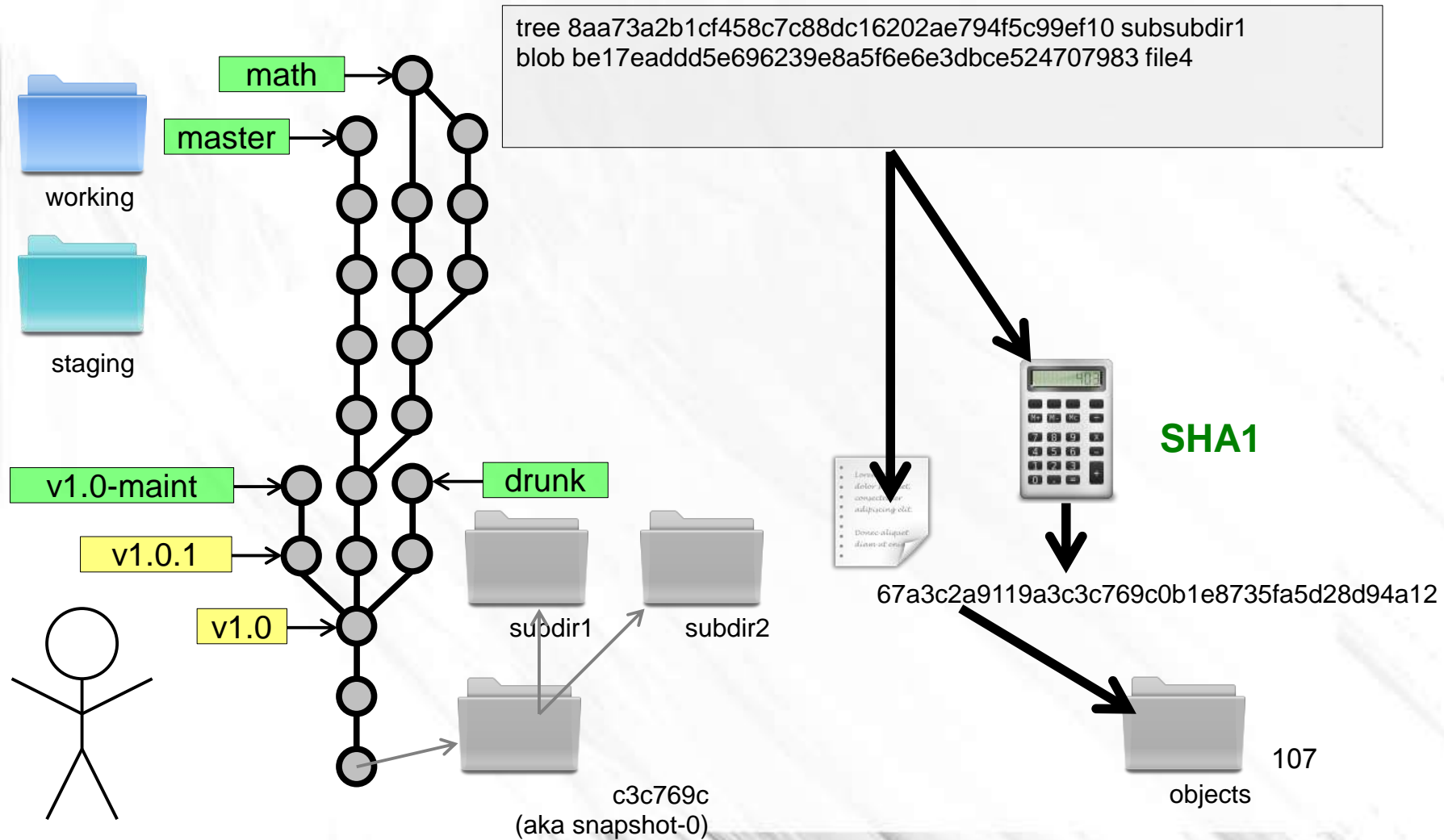
# Eliminating Duplication



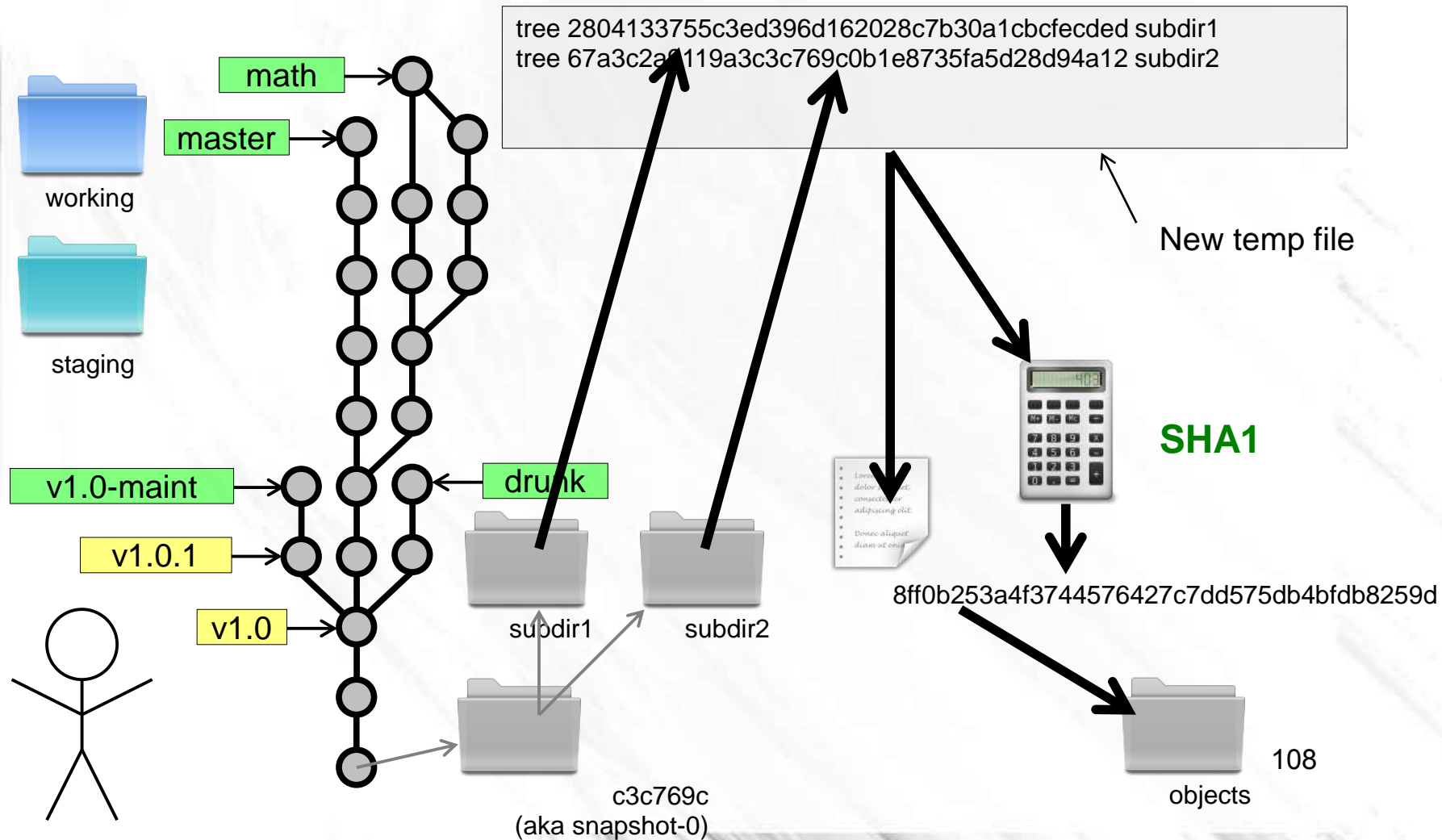
# Eliminating Duplication



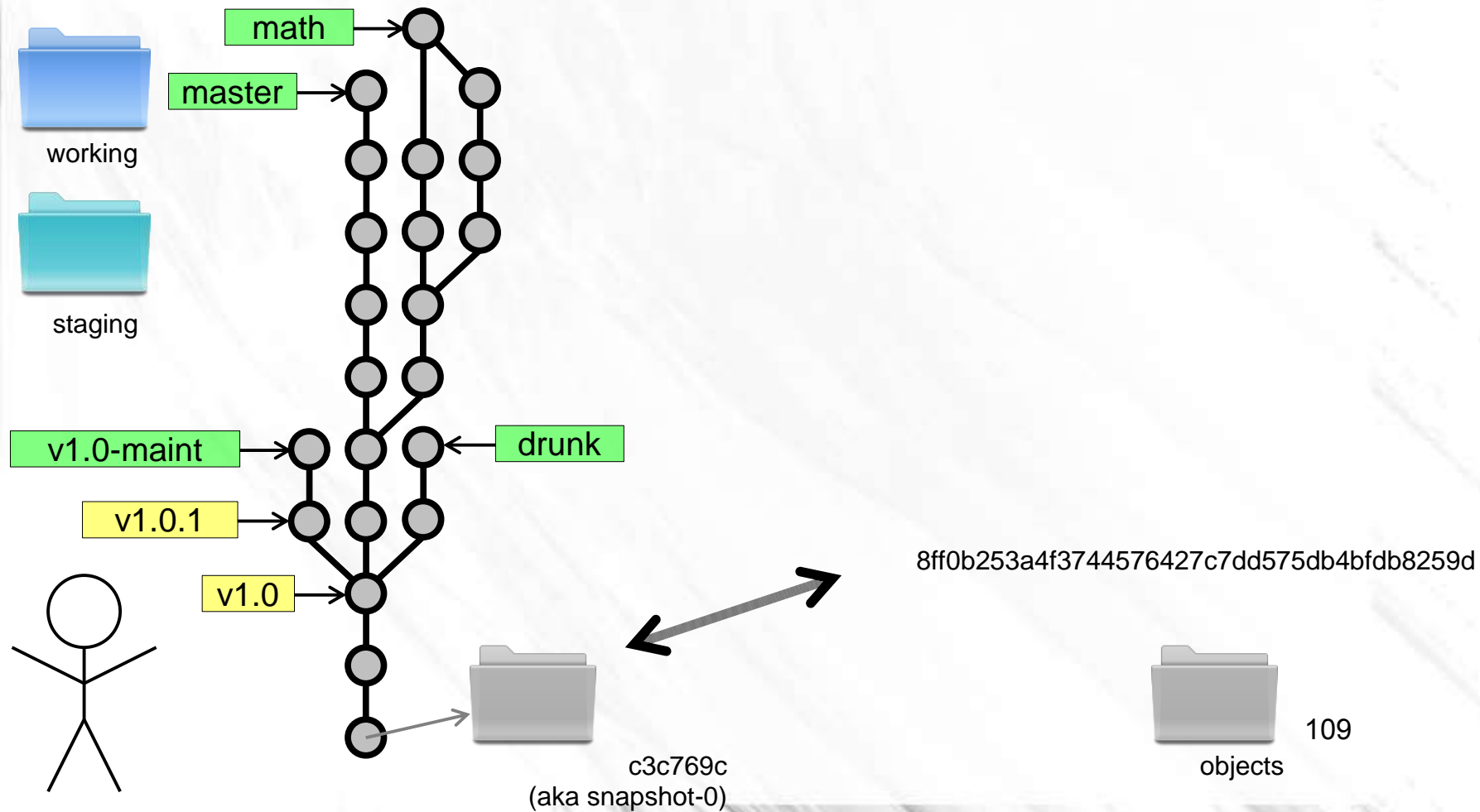
# Eliminating Duplication



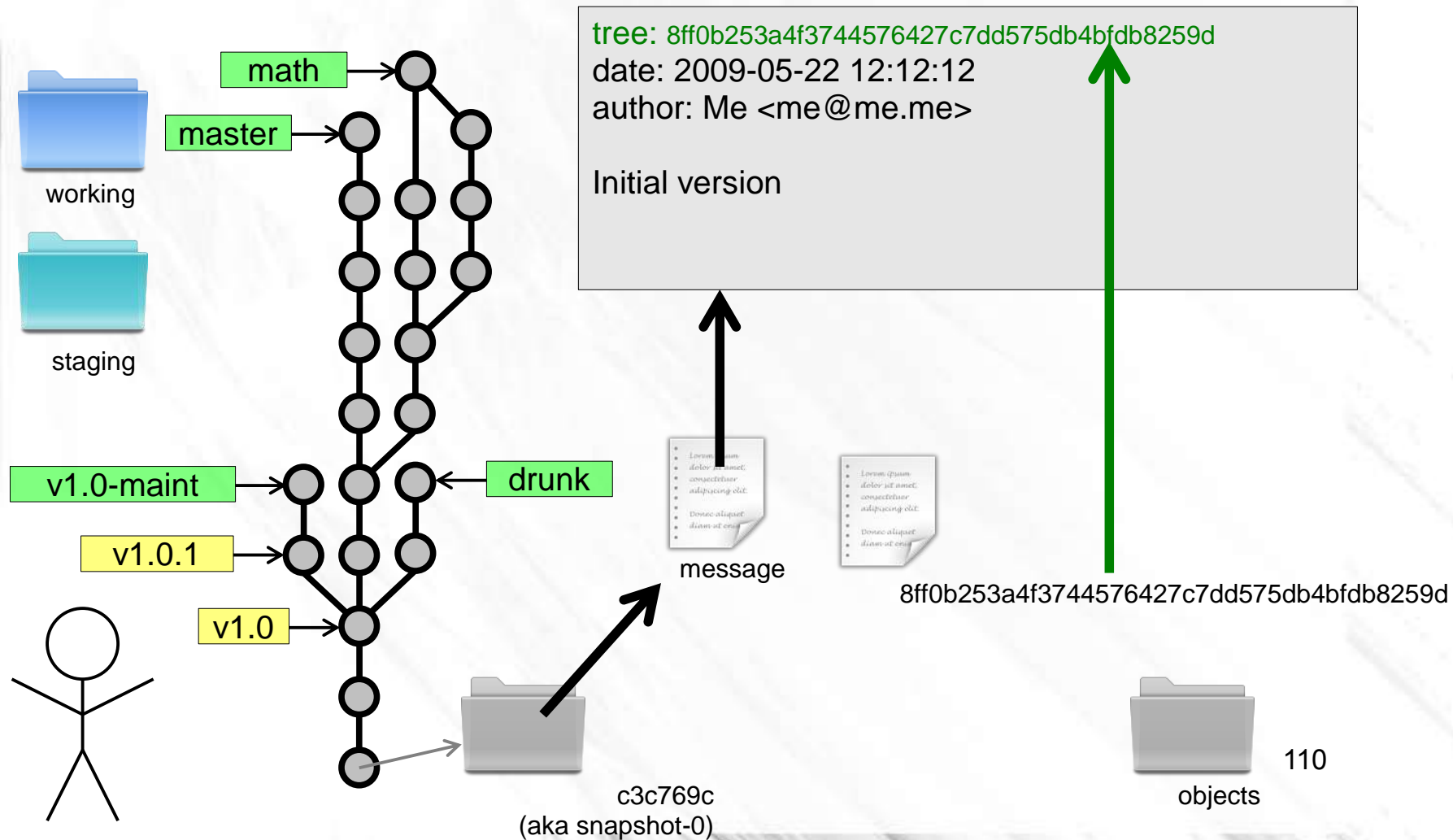
# Eliminating Duplication



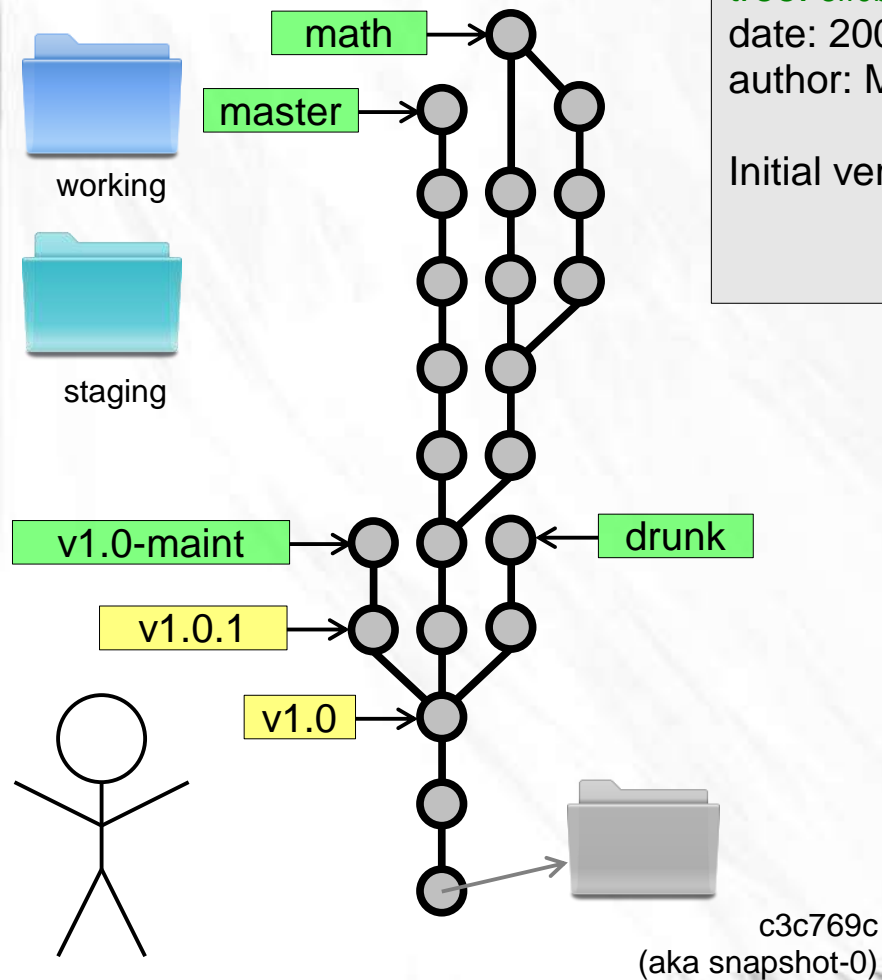
# Eliminating Duplication



# Eliminating Duplication



# Eliminating Duplication

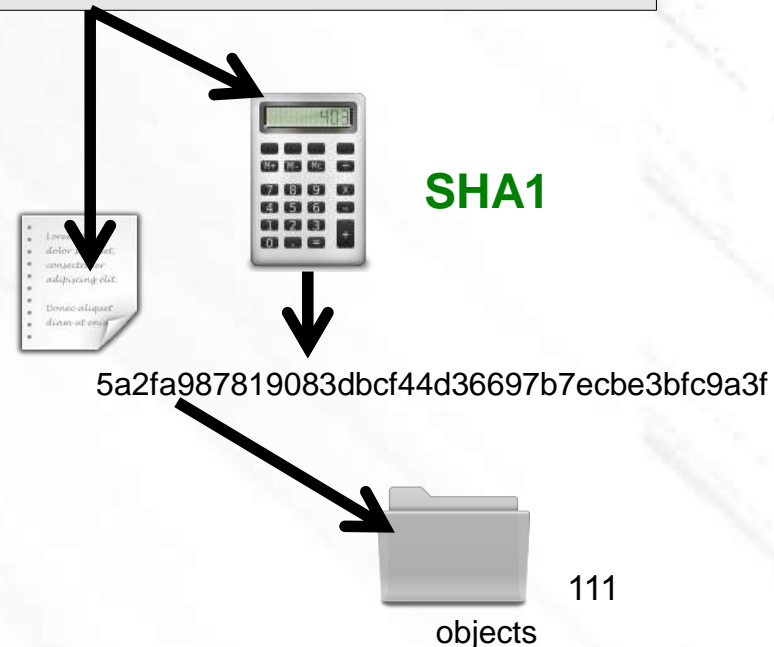


tree: 8ff0b253a4f3744576427c7dd575db4bfdb8259d

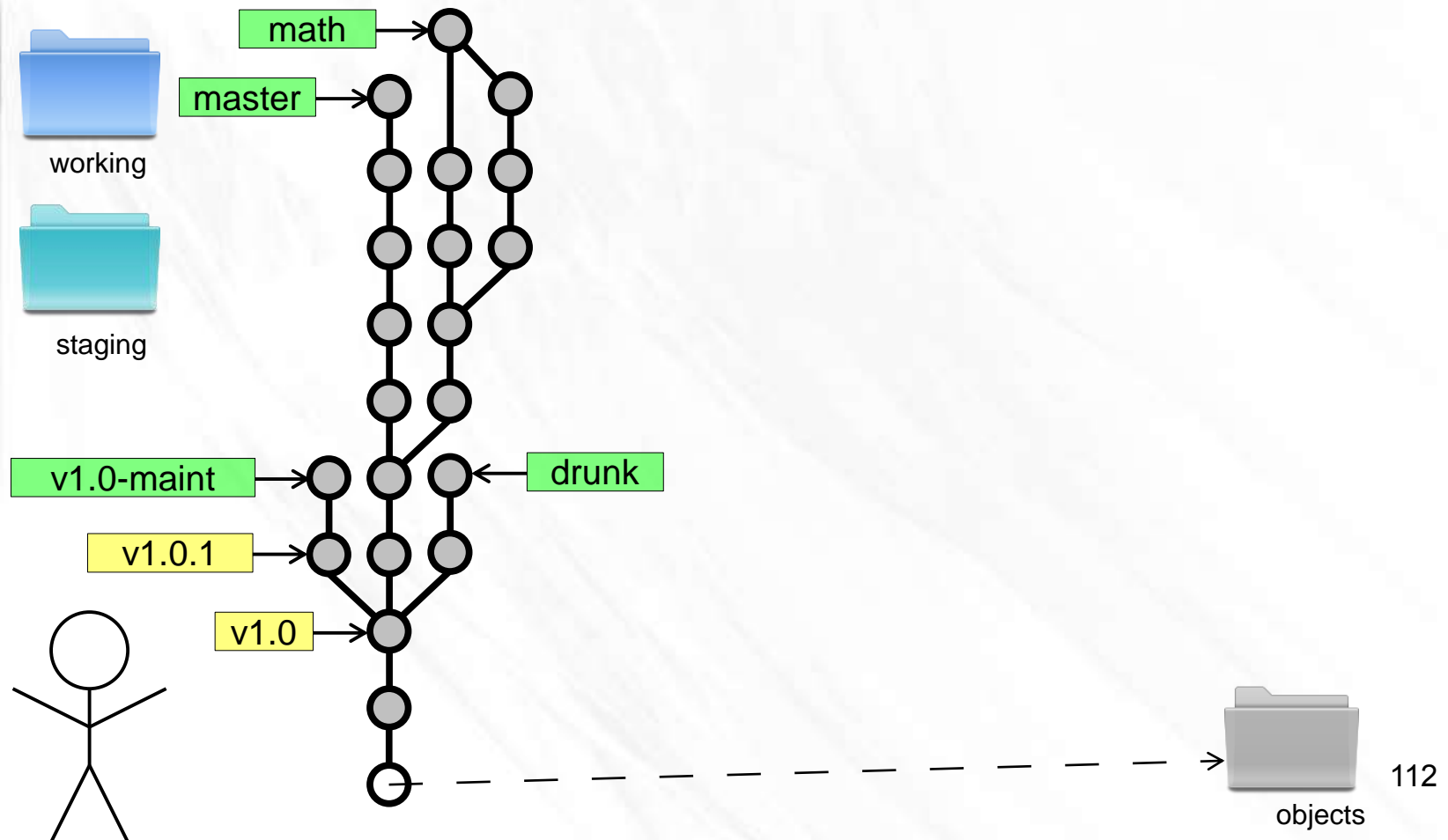
date: 2009-05-22 12:12:12

author: Me <me@me.me>

Initial version

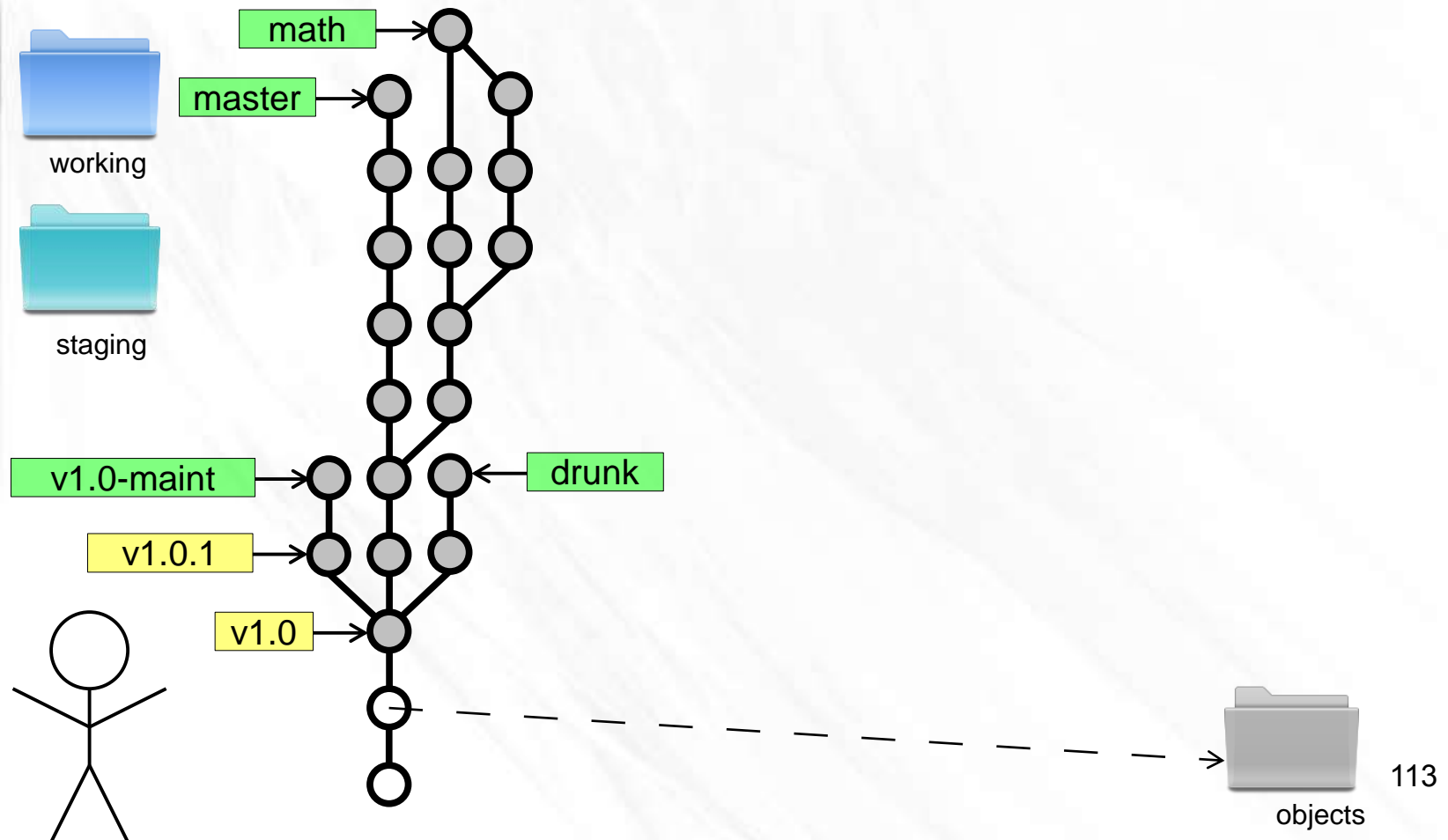


# Eliminating Duplication

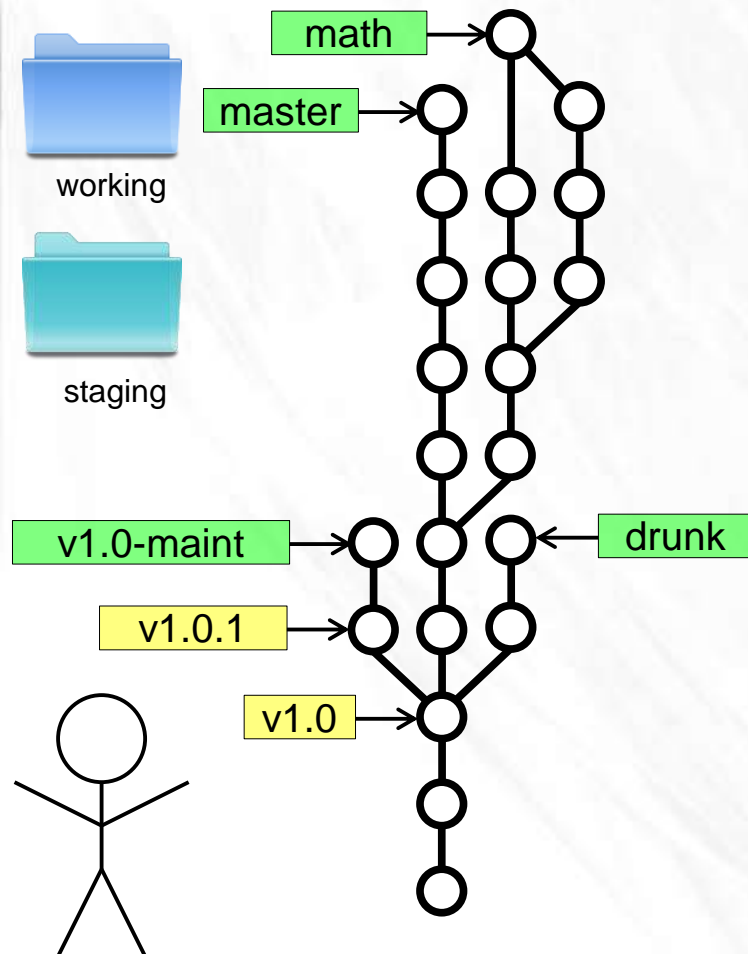




# Eliminating Duplication

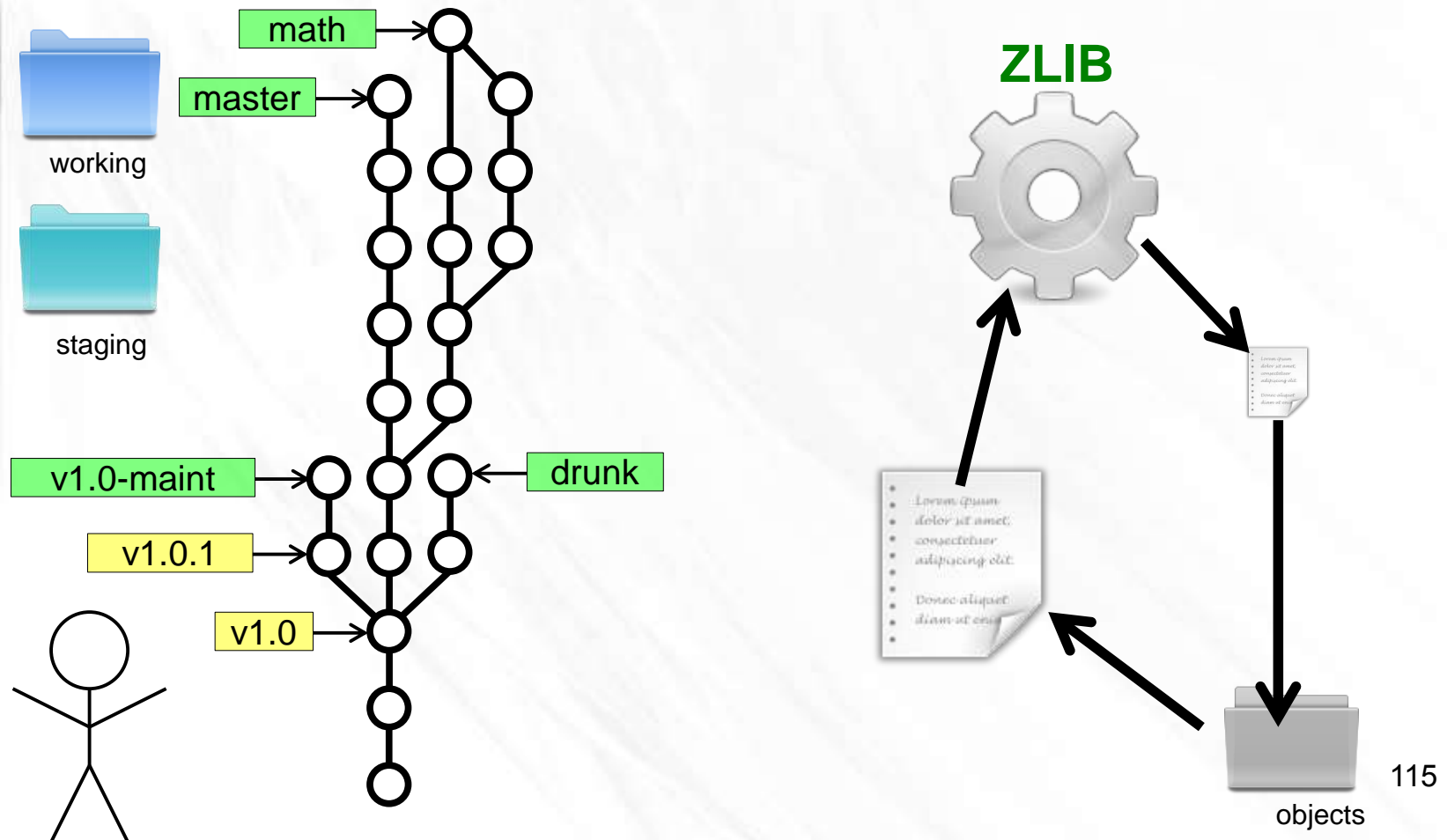


# Eliminating Duplication

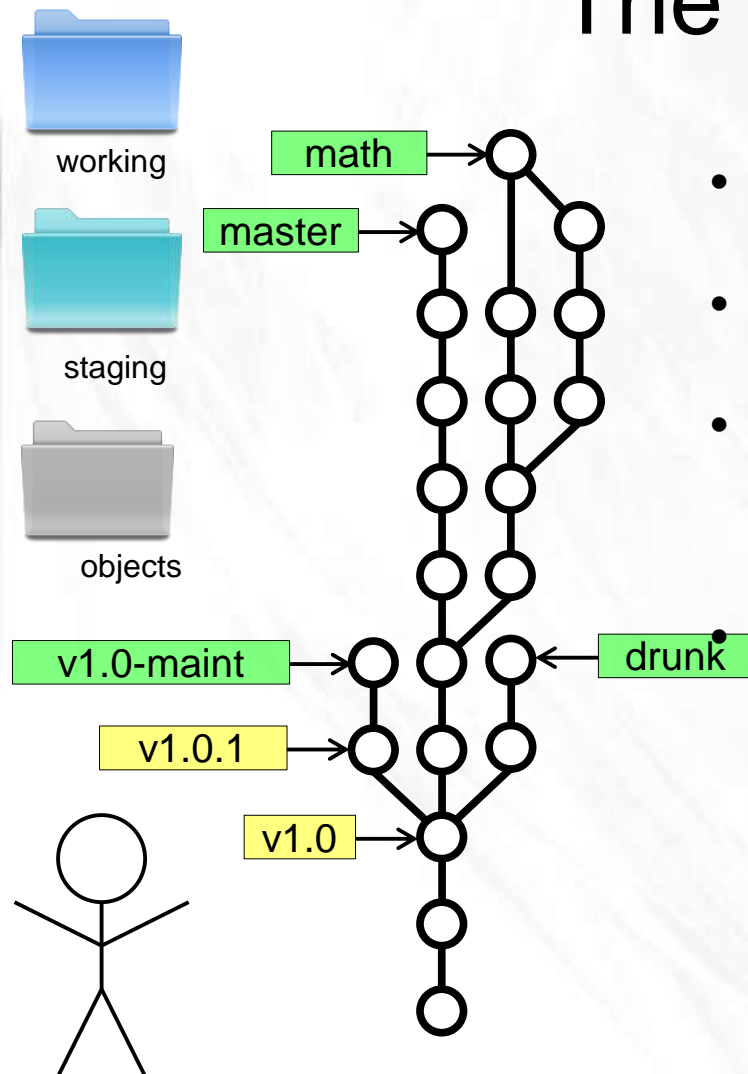


objects

# Compressing Blobs



# The True Git

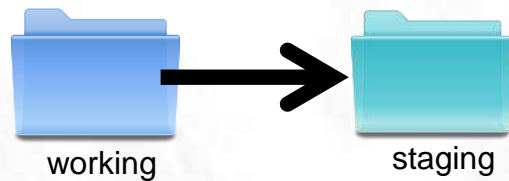


- TADAA!
- This is pretty much Git
- Nicer command line tools for all these operations
- Many, many other tools

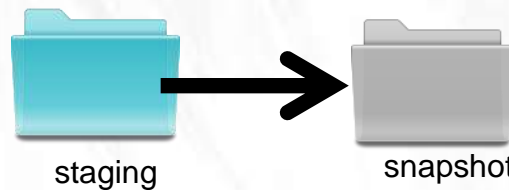
# Commands: Getting Started

- 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`



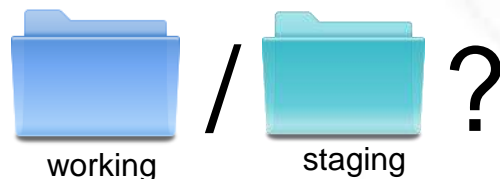
- `git commit`

}

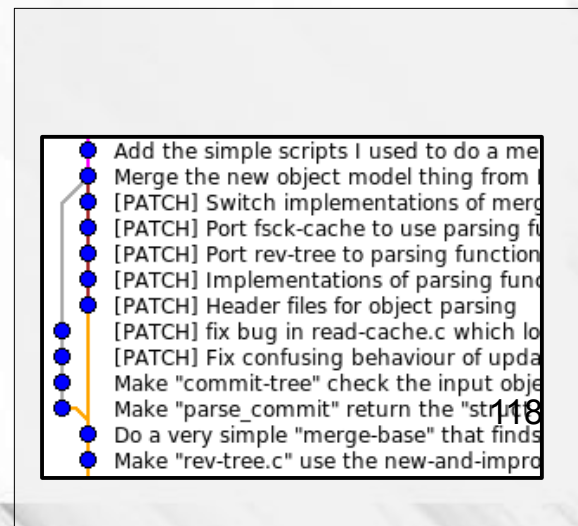
```
git commit -a
```



- `git log`



- `git status`



# gitk

# 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>`

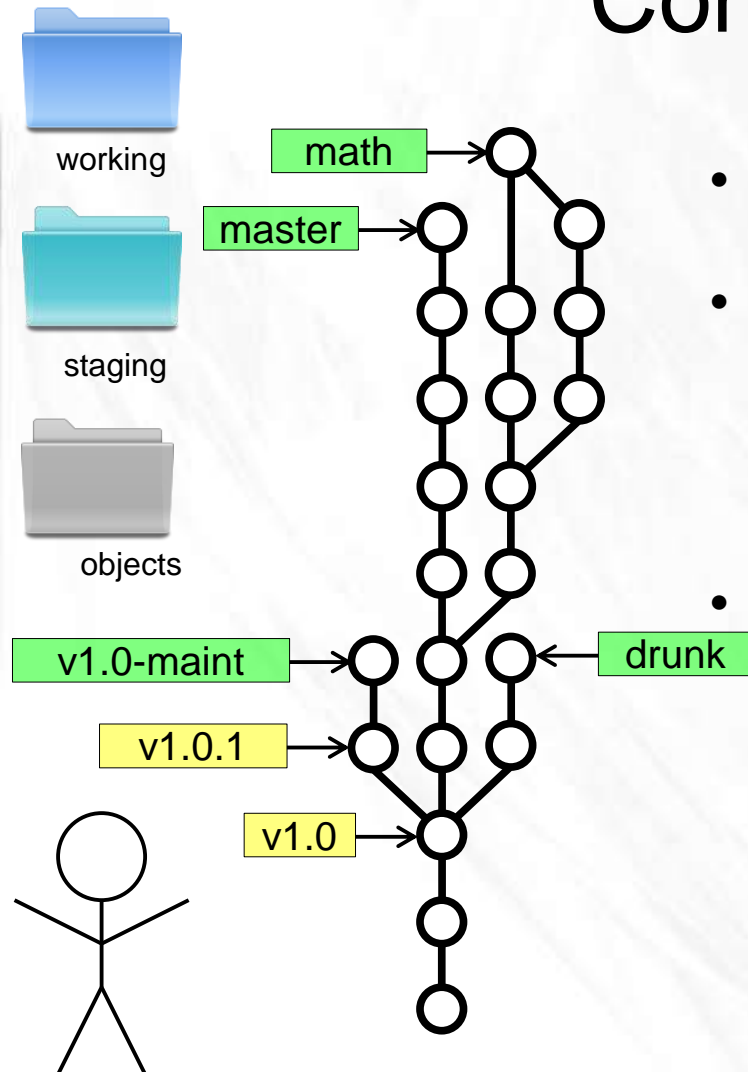
- `git fetch <name>`

}

`git pull`

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

# 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>

# 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)>



# אחר: בקרת גרסאות תוצרת בית

- Jim Weirich
- [https://github.com/jimweirich/presentation\\_source\\_control](https://github.com/jimweirich/presentation_source_control)
- <http://pragprog.com/screencasts/v-jwsceasy/source-control-made-easy> ([promo](#))

# Git makes more sense when you understand [...]



**@KentBeck**  
Kent Beck

finally figuring out that git commands are  
strangely named graph manipulation  
commands--creati  
moving pointers at



**@tabqwerty**  
chi wai lau

1 Mar via [TweetDeck](#) ★ Unfavorite ↻

"git gets easier once you get the basic idea  
that branches are homeomorphic  
endofunctors mapping submanifolds of a  
Hilbert space."

9 Mar via web ★ Unfavorite ↻ Undo Retweet ↻ Reply

# Git Clients (Windows)

- CLI Shell: Git Bash
- Windows Explorer Shell
- Github for Windows (+powershell)
- Bitbucket SourceTree
- IDE Integration
  - Visual Studio (VS2013 native)
  - Eclipse Egit
  - IntelliJ embedded

# הדגמה - טיפים

- <http://gitimmersion.com/>
- <http://learn.github.com/>
- <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



# תהליך - Git Flow

- למשל בפרויקטי קוד פתוח
- Git (hub) flow
- דוגמא: תהליך העבודה בפרויקט Nuget:  
[Contributing a Bug Fix or Feature](#)
- ראו קישורים בויקי

# בפעם הבאה

- פרויקט: פיתוח בסבבים
- בדיקות, פיתוח מונחה מפרטים
- שעה שלישית – מעבדה (משימה אישית 4, להביא מחשבים)
- בהמשך
- ? בקרת גרסאות II – תרחישים נוספים עם git
- בדיקות וחברים
- משימה אישית מס' 3 (מומלץ gitimmersion!):  
קורס git מקוון קצר [try.github.io](https://try.github.io)  
להגשה: קישור לתגית סיום הקורס (דוגמא)  
אפשר להגיע אל התגית ע"י כניסה לאתר [codeschool](https://codeschool.com) עם חשבון github <- My Report  
Card <- Make Public <- Earned Badges ולחיצה על התגית

# לסיכום

- בקרת גרסאות
  - שיטה וכלים
  - תמיכה בתהליך