Version Control

C++ Programming Practice

Victor Yacovlev (Viktor Iakovlev)
and Julia Ivanova

What is Versioning

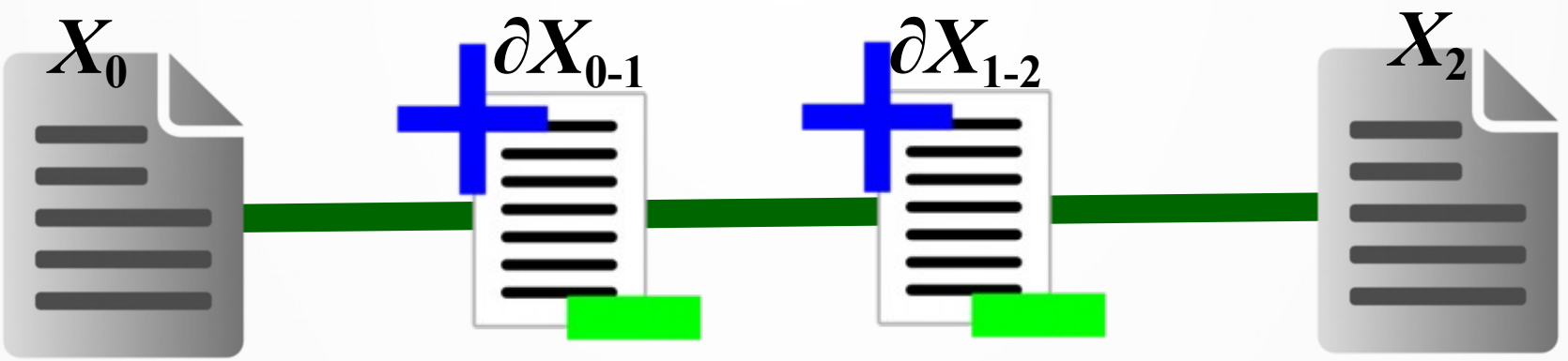
Versioning for Single Developer

- Use of different computers (at work, at home)
- Tracking of changes between versions

Version Control Systems

- Centralized:
  - Concurent Version System (cvs)
  - Subversion (svn)
- Distributed:
  - Mercurial (hg)
  - Git (git)

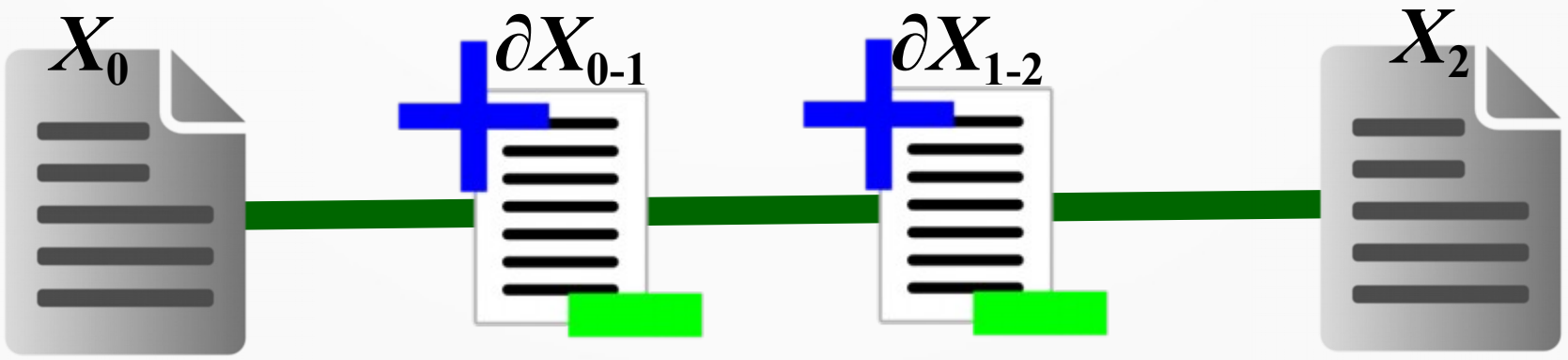
The Key Idea



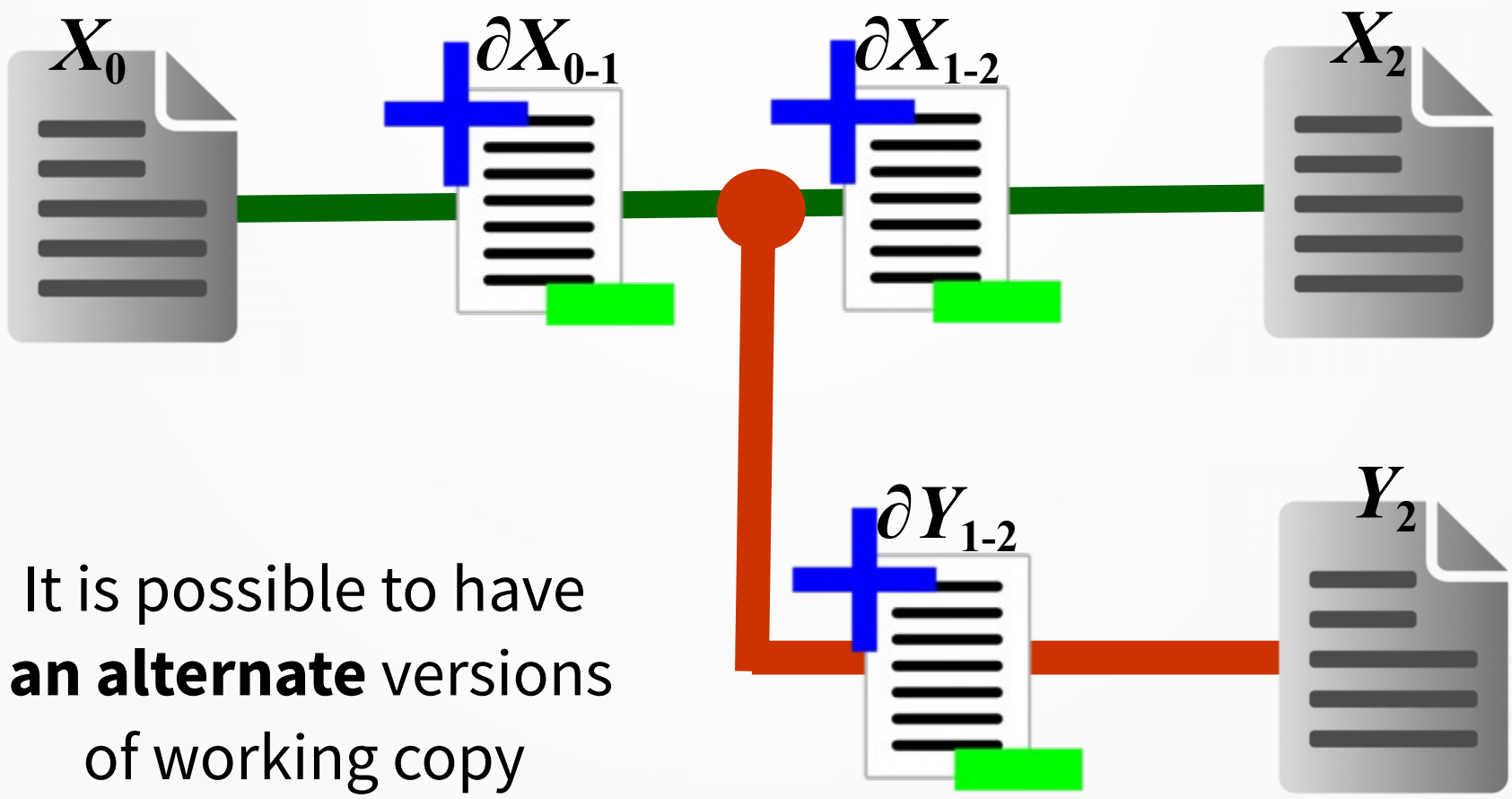
Any text might be created by having initial text X_0 updated by set of patches $\partial X_{0\text{-}1} \ldots \partial X_{1\text{-}2} \ldots \partial X_{M\text{-}N} \ldots X_N$.

The Key Idea

- Store initial version of files
- Keep current version as-is (the working copy)
- Commit changes from current version to previous by calculating differences
- Store differences in history



Alternate Version (Branch)

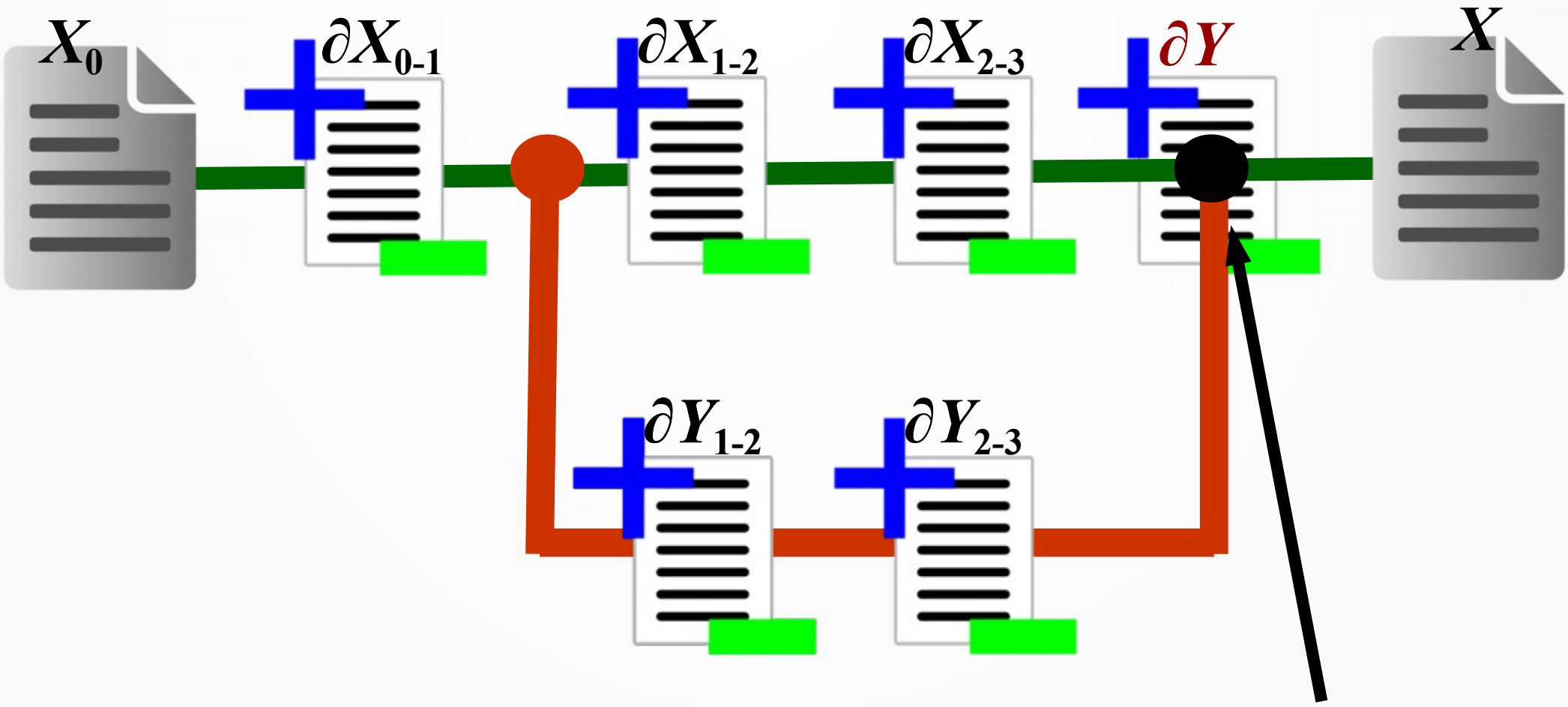


It is possible to have **an alternate** versions of working copy

Alternate Version

- Do not make a full copy – just keep changes
- Versions can be switched by reapplying patches
- Typical use case: make new feature in project keeping mainstream untouched until feature ready

Merging Alternate Back



This join might have conflicts to be resolved manually!

Conflicts

- Alternate history may affect the same files and the same parts of files
- On merging there is a problem: which part of text to use
- This can’t be performed automatically

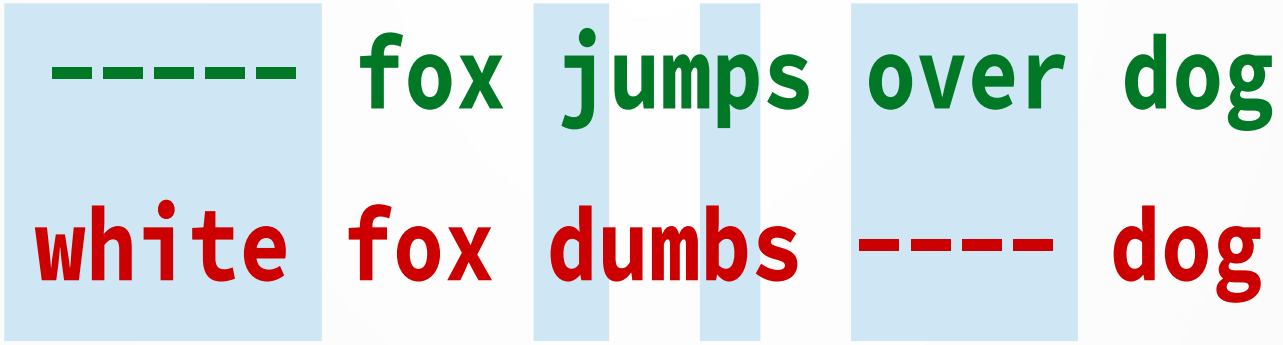
What is a difference calculation (a bit of mathematics to undestand)

Text Alignment Problem

fox jumps over dog

white fox dumbs dog

Text Alignment Problem



Patch := (**+white**), (**j → d**), (**p → b**), (**-over**)

How Patches are Calculated

- The key idea – use the **Dynamic Programming** optimization method
- Each string S_1 and S_2 to be compared have a substrings S_1^N and S_2^N of length N
- Recursively calculate some measure function $F(S_1^N, S_2^N)$ and choose the best value on recursion stage

How Patches are Calculated

- The $F(S_1^N, S_2^N)$ function is an **edit distance**
- There is several way to calculate $F(S_1^N, S_2^N)$
- The notable methods to calculate F:
  - Takes account only on insertions and deletions:
    - Levenshtein Distance
    - **Longest Common Subsequence – used in git version control system**
  - Takes account boths insertions/deletions and symbols matching:
    - Needleman-Wunsch Algorithm
    - Smith-Waterman Algorithm
    - Roytberg-Yacovlev Algorithm

The Levenshtein Distance Measure Function

$$D_{i,j} = min \begin{Bmatrix} D_{i-1,j-1} + Score_{i,j} \\ D_{i-1,j} + 1 \\ D_{i,j-1} + 1 \end{Bmatrix}$$

$$Score_{i,j} = \begin{Bmatrix} 1 & S_1^i \neq S_1^i \\ 0 & S_1^i = S_2^j \end{Bmatrix}$$

The Longest Common Sequence Measure Fucntion

$$C_{i,j} = \begin{cases} max \begin{cases} C_{i-1,j} \\ C_{i,j-1} \end{cases} & S_1^i \neq S_1^i \\ C_{i-1,j-1} + 1 & S_1^i = S_2^j \end{cases}$$

The Needle-Wunch Measure Function

$$W_{i,j} = max \begin{Bmatrix} W_{i-1,j-1} + Score_{i,j} \\ W_{i-1,j} - GEP \\ W_{i,j-1} - GEP \end{Bmatrix}$$

Score(i,j) - the Weight of symbols matching

GEP - the Gap Elongation Penalty for symbol deletion

Example: Using NW to calculate difference

		f	o	x		j	u	m	p	s		o	v	e	r		d	o	g
		-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16	-17	-18
w	-1	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
h	-2	5	12	11	10	9	8	7	6	5									20
i	-3	4	11	18	17	16	15	14	13	12									27
t	-4	3	10	17	18	23	22	21	20	19									34
e	-5	2	9	16	17	24	29	28	27	26									41
	-6	1	8	15	46	45	44	43	42	41									72
f	-7	24	23	22	45	52	51	50	49	48									79
o	-8	23	54	53	52	51	58	57	56	55	54	85	84	83	82	81	80	87	86
x	-9	22	53	84	83	82	81	80	79	78	77	84	91	90	89	88	87	86	93
	-10	21	52	83	114	113	112	111	110	109	198	197	196	195	194	209	208	207	206
d	-11	20	51	82	113	120	119	118	117	116	197	204	203	202	201	208	239	238	237
u	-12	19	50	81	112	119	150	149	148	147	196	203	210	209	208	207	238	245	244
m	-13	18	49	80	111	118	149	180	179	178	195	202	209	216	215	214	237	244	251
b	-14	17	48	79	110	117	148	179	186	185	194	201	208	215	222	221	236	243	250
s	-15	16	47	78	109	116	147	178	185	216	215	214	213	214	221	222	235	242	249
	-16	15	46	77	198	197	196	195	194	215	246	245	244	243	242	251	250	249	248
d	-17	14	45	76	197	204	203	202	201	214	245	252	251	250	249	250	281	280	279
o	-18	13	44	75	196	203	210	209	208	213	244	275	274	273	272	271	280	311	310
g	-19	12	43	74	195	202	209	216	215	214	243	274	281	280	279	278	279	310	341

Space v.s. space	120
Exact symbol match	30
Letter v.s. letter	6
Space v.s. letter	0
Gap Elongation Penalty	-1

Block Alignment

- Matrices might be too big
- This might be solved by grouping symbols

		fox	jumps	over	dog
		-1	-2	-3	-4
white	-1	0	-1	-2	-3
fox	-2	9	8	7	6
dumbs	-3	8	9	8	7
dog	-4	7	8	9	18

Block Alignment

-----	fox	jumps	over	dog
white	fox	dumbs	----	dog

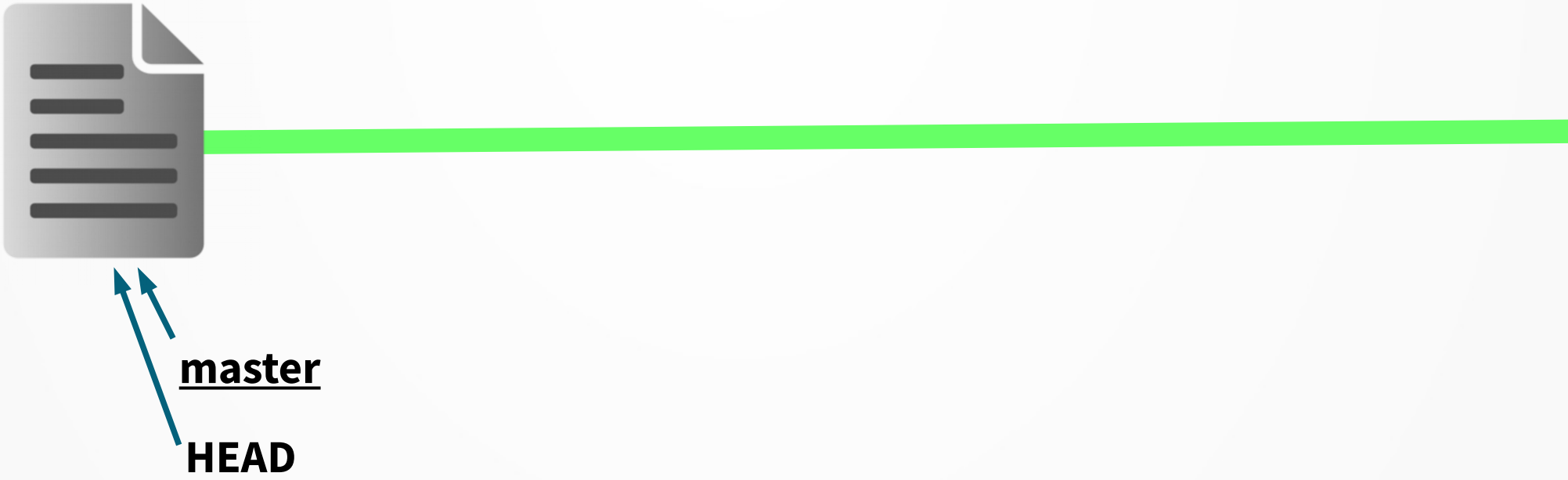
insertion replacements deletion

-----	fox	jumps	over	dog
white	fox	dumbs	----	dog

insertion replacement deletion

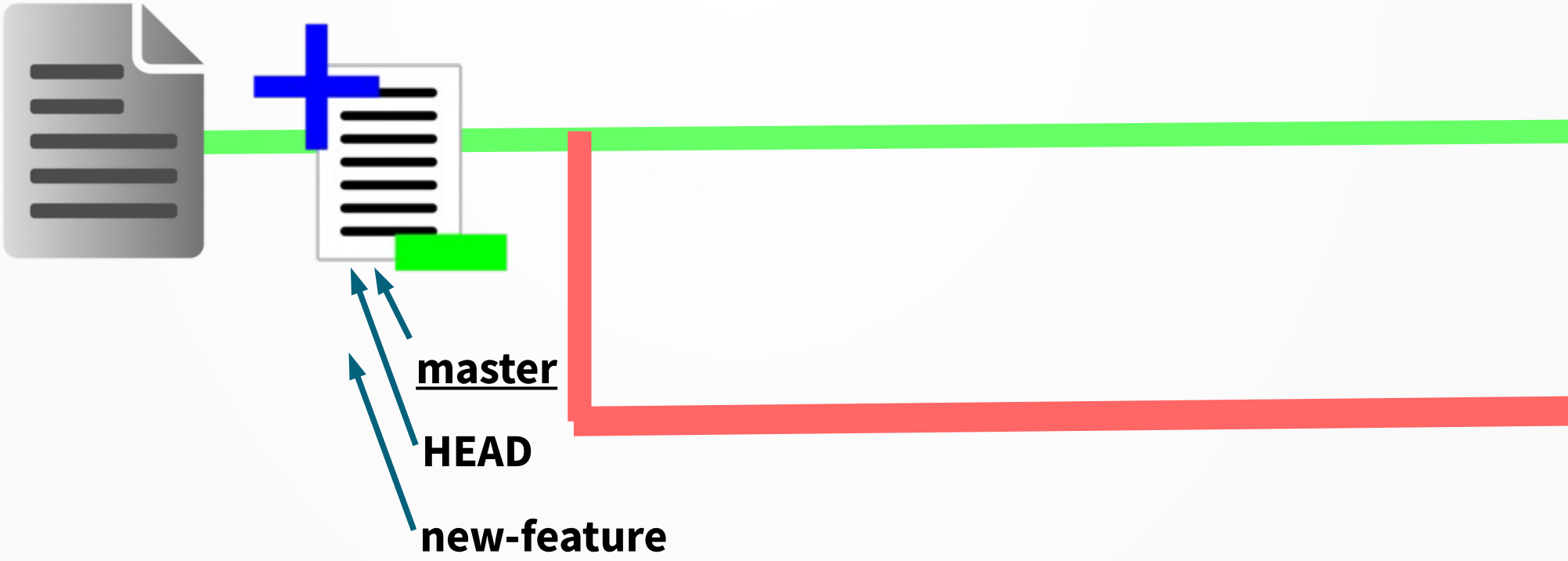
Git Version Control System

Git



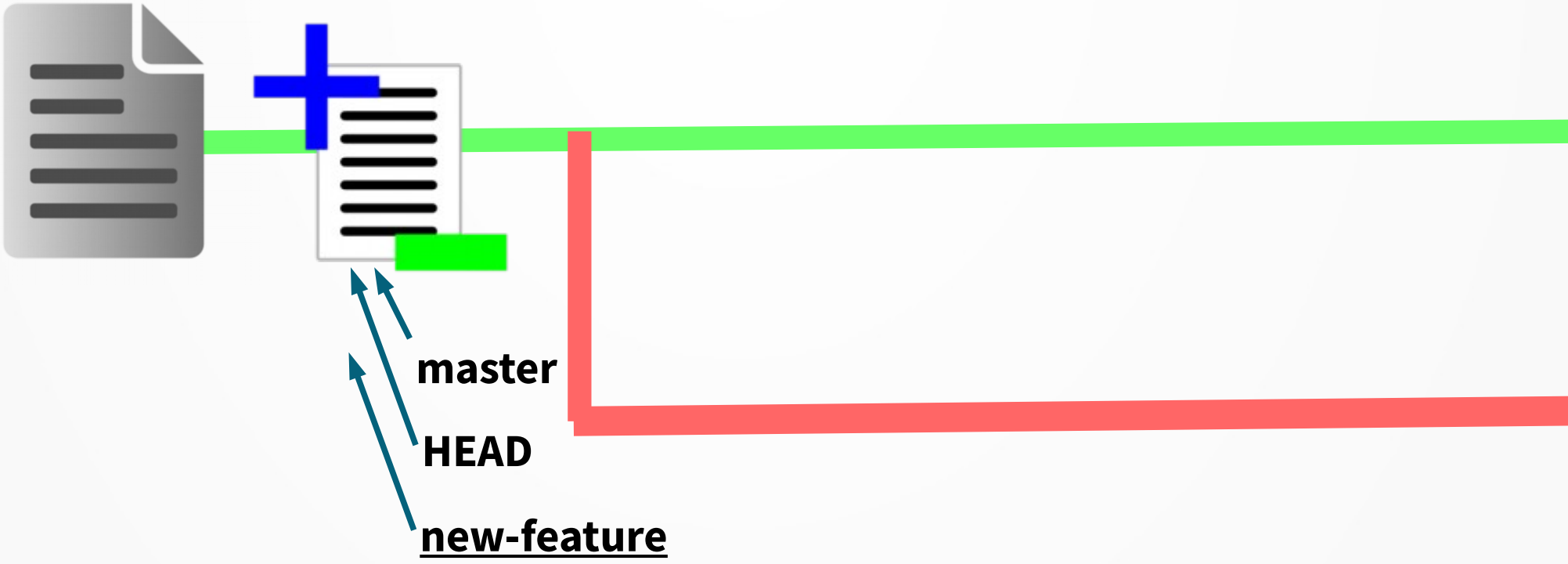

```
git init && git add && git commit
```

Git



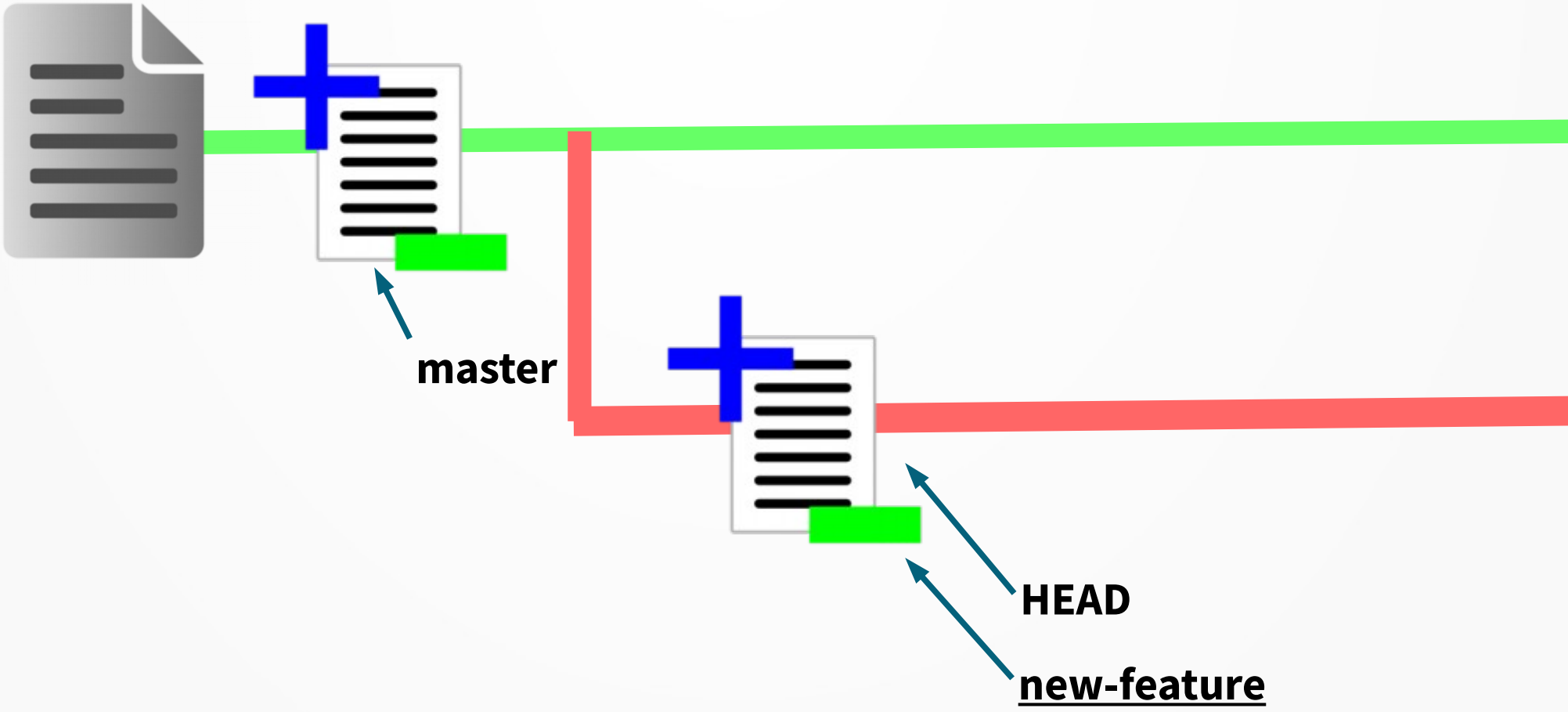

```
git branch new-feature
```

Git



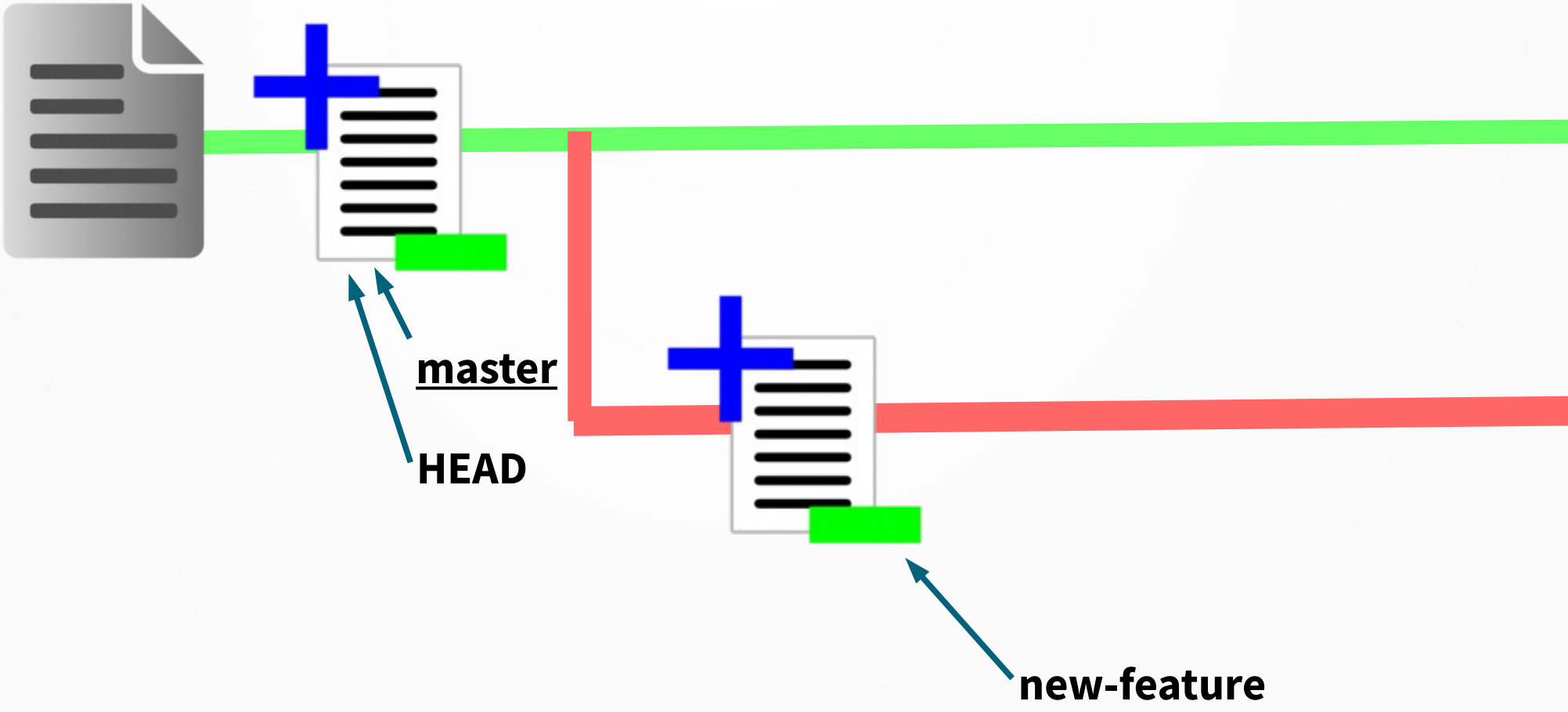

```
git checkout new-feature
```

Git



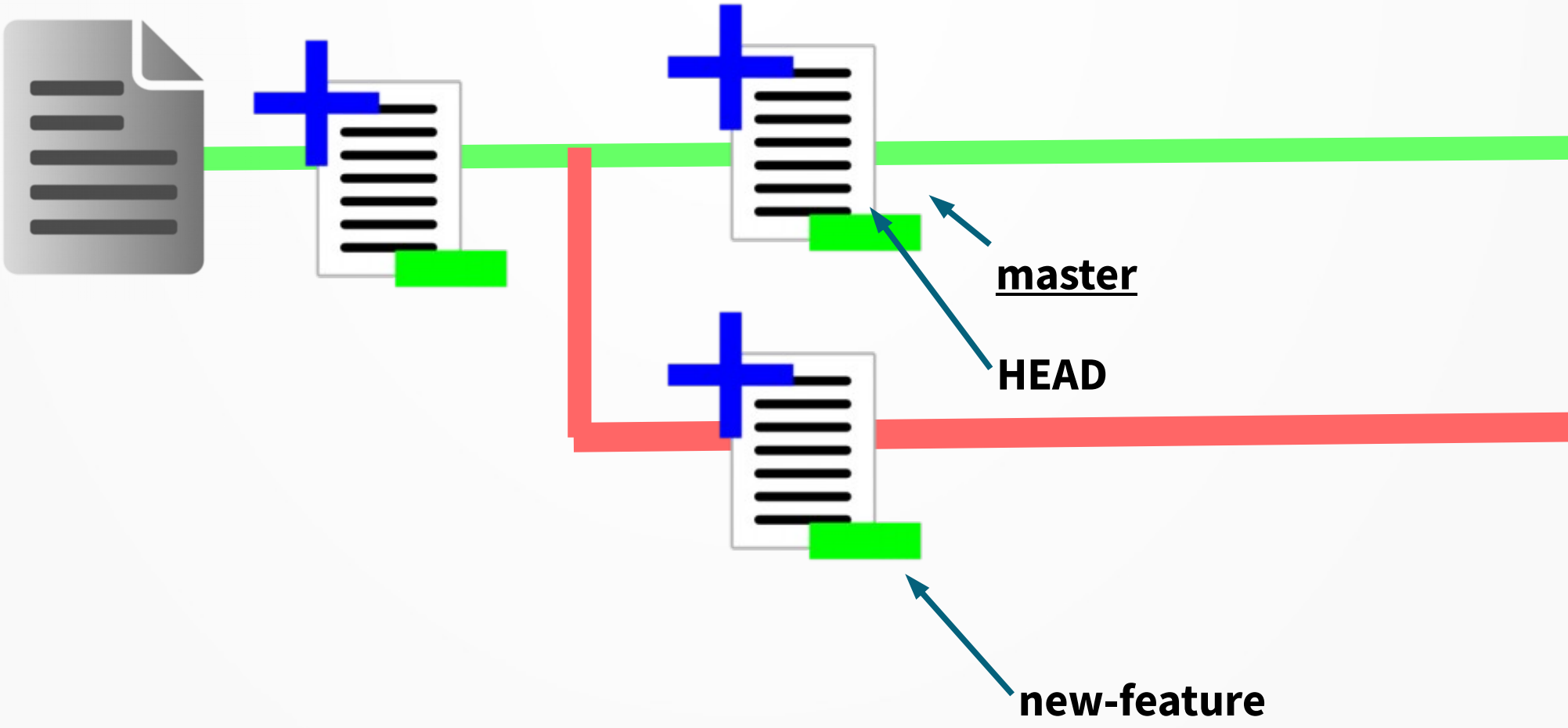
`git commit`

Git




```
git checkout master
```

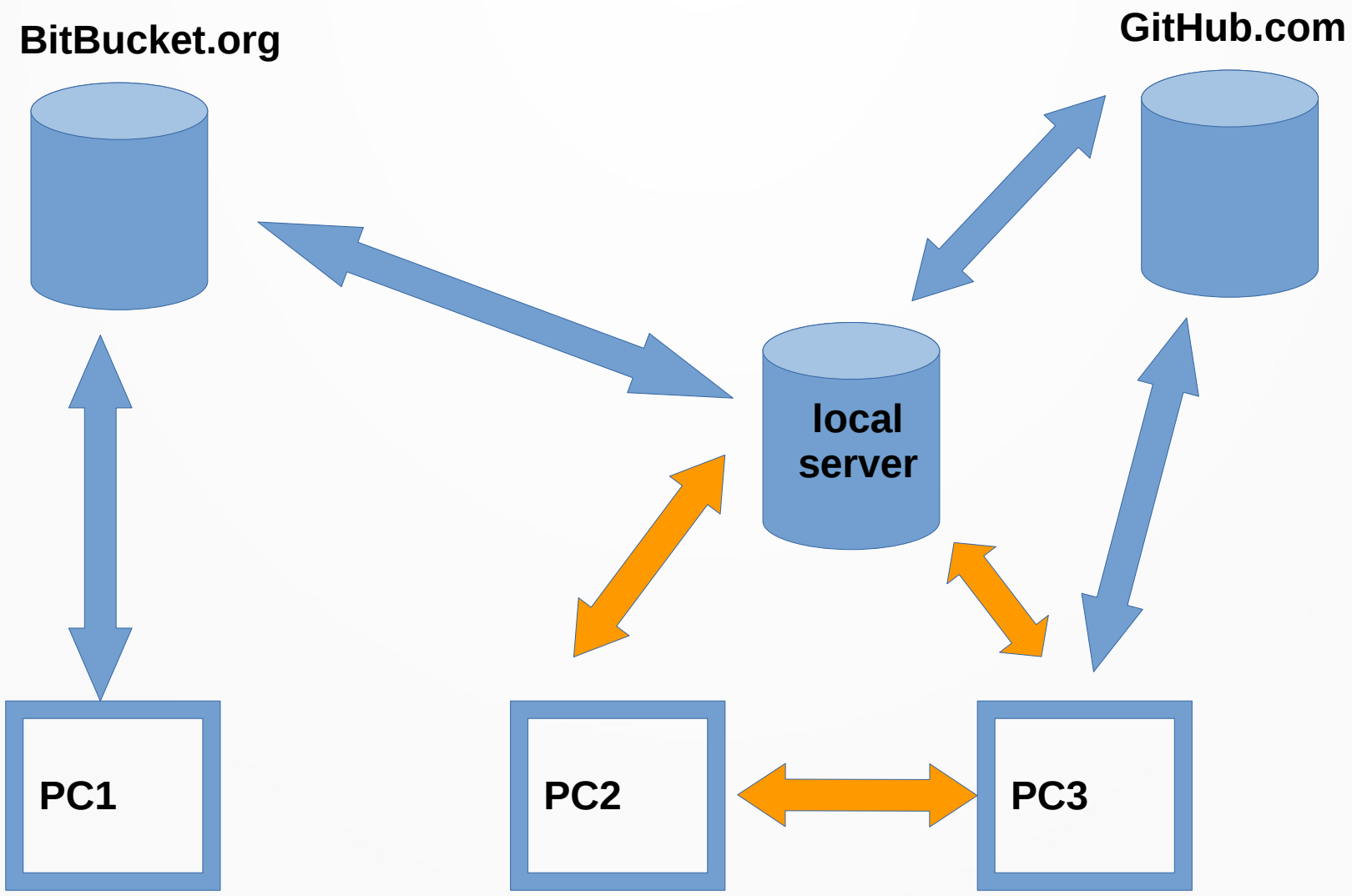
Git



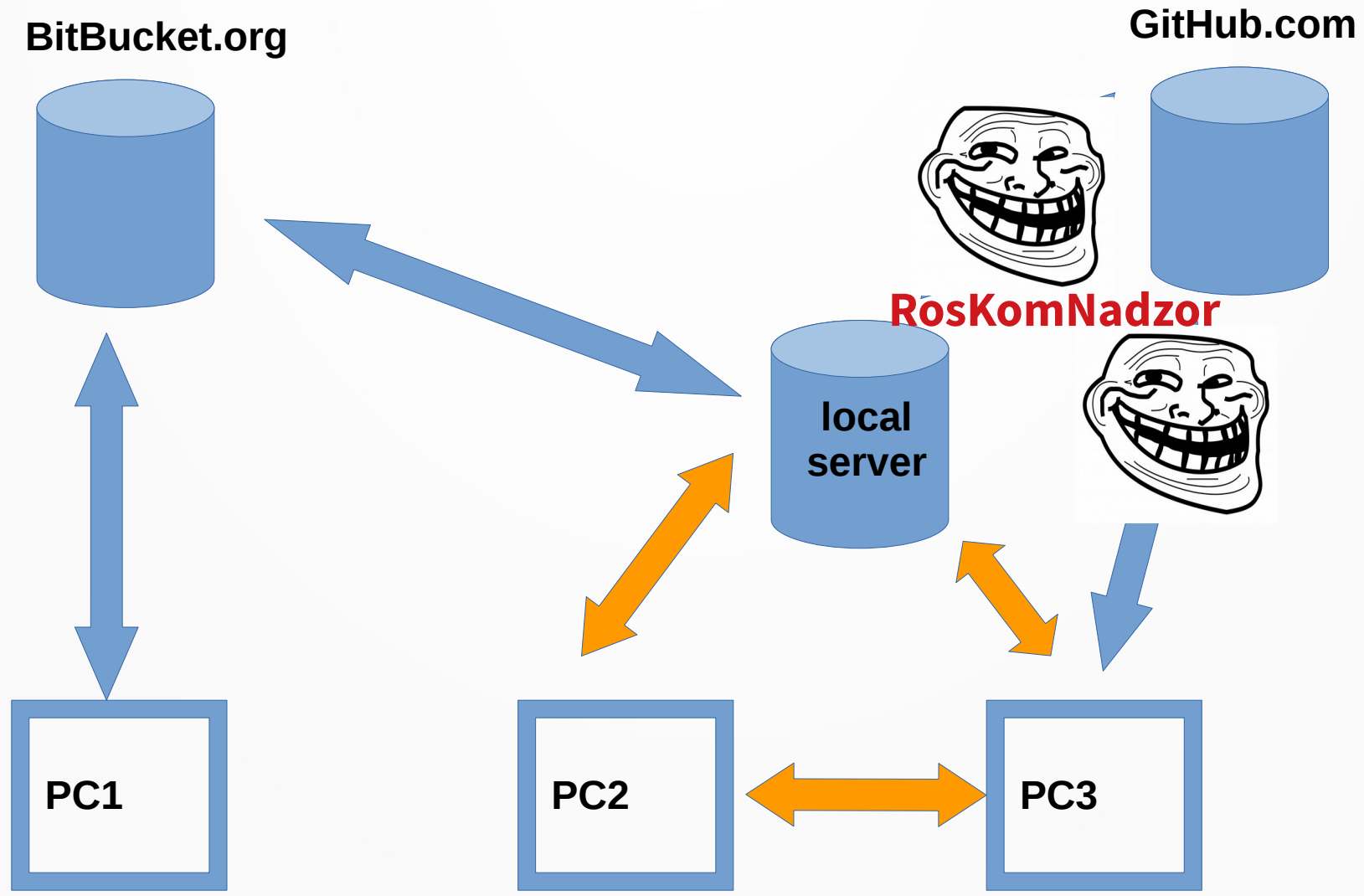
`git commit`

Distributed Storage

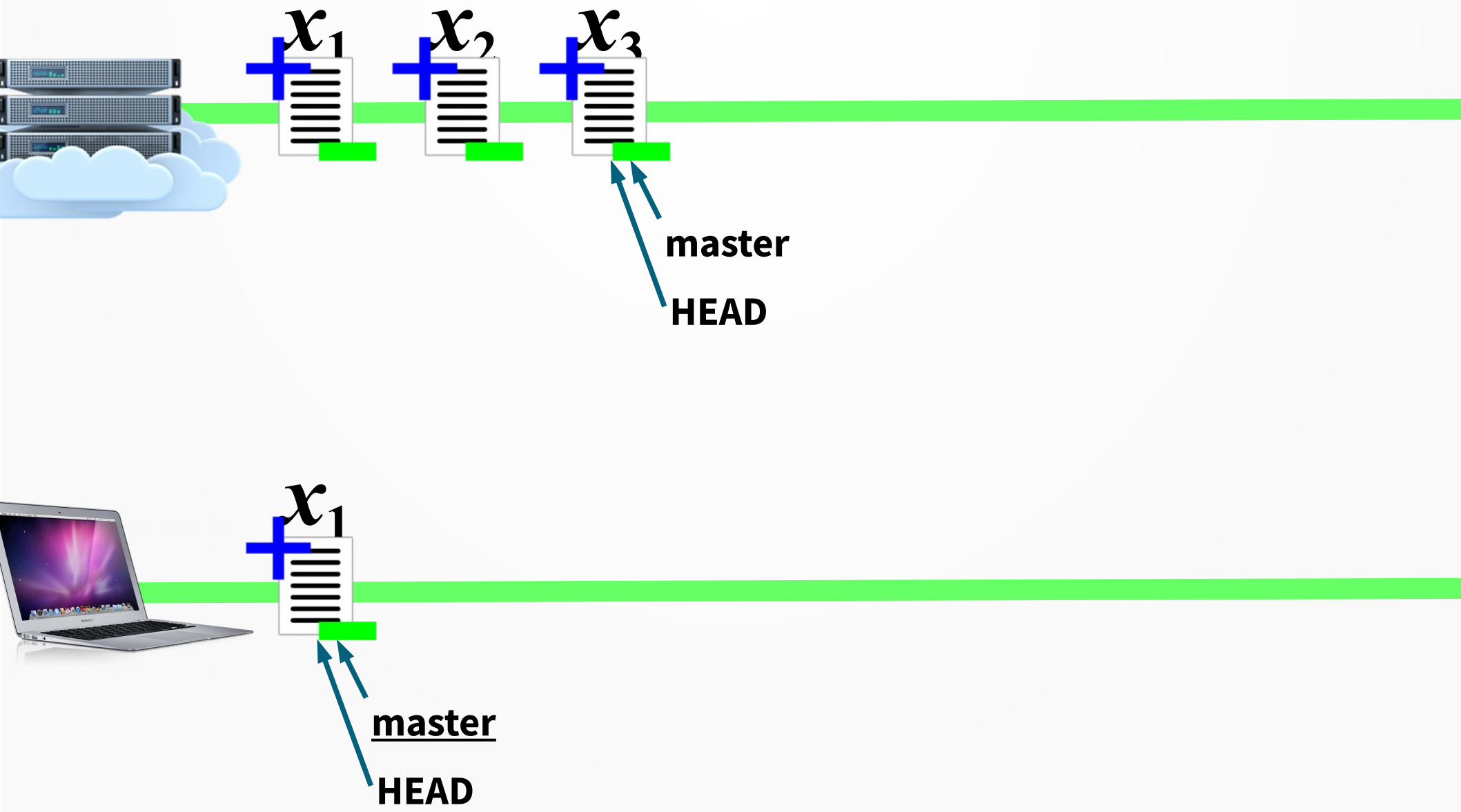
Distributed Git



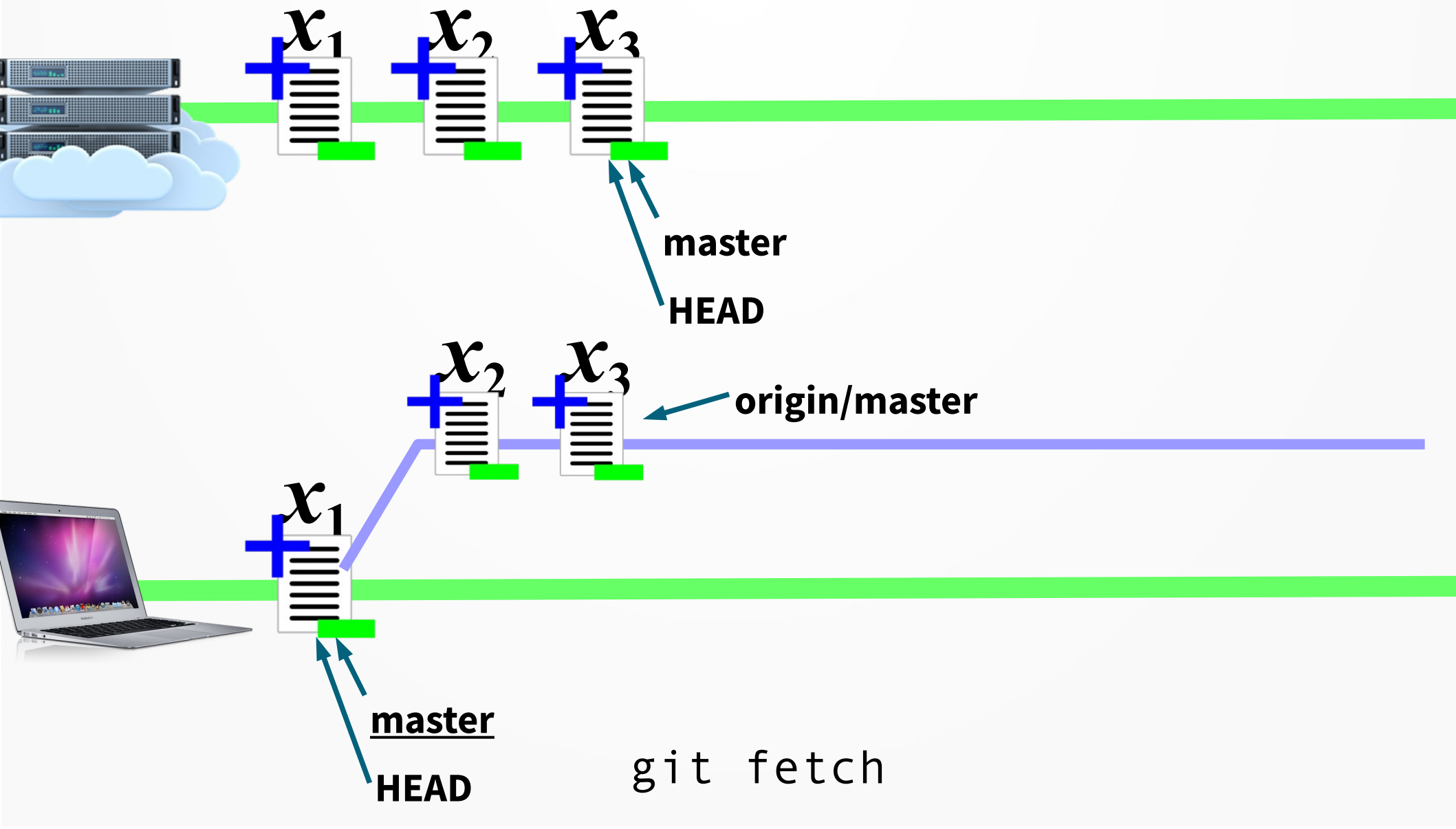
Distributed Git



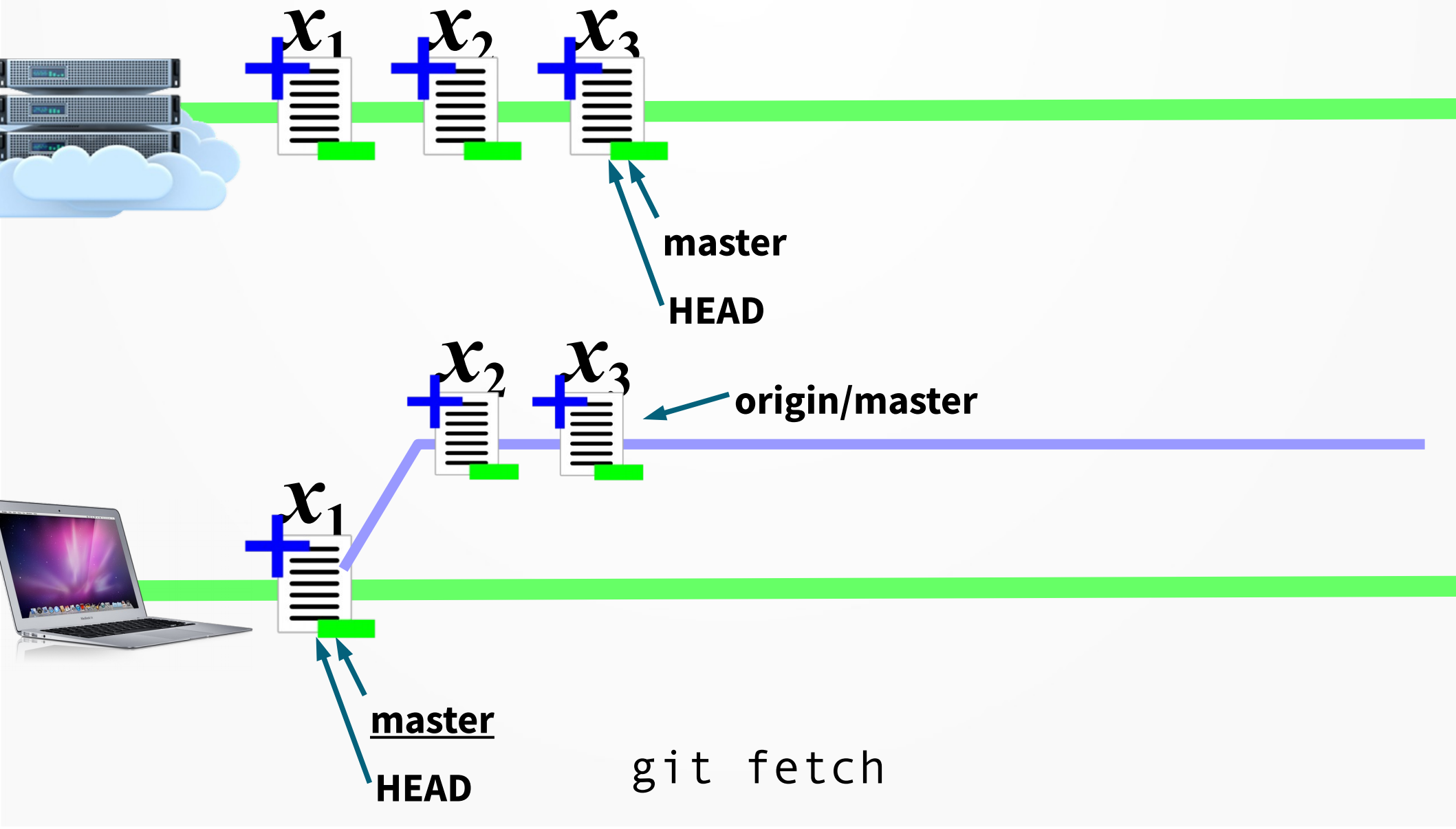
Synchronization



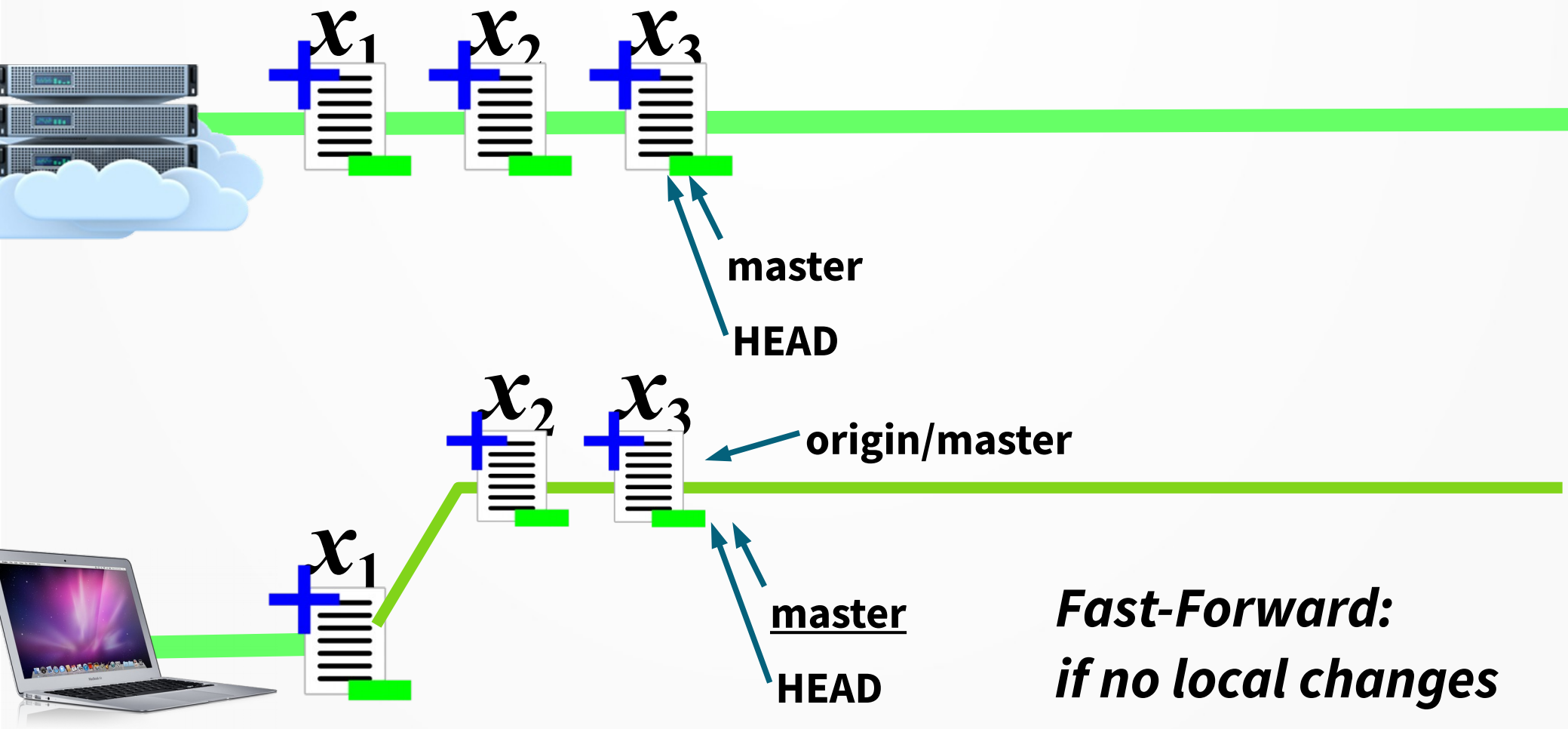
Synchronization



Synchronization

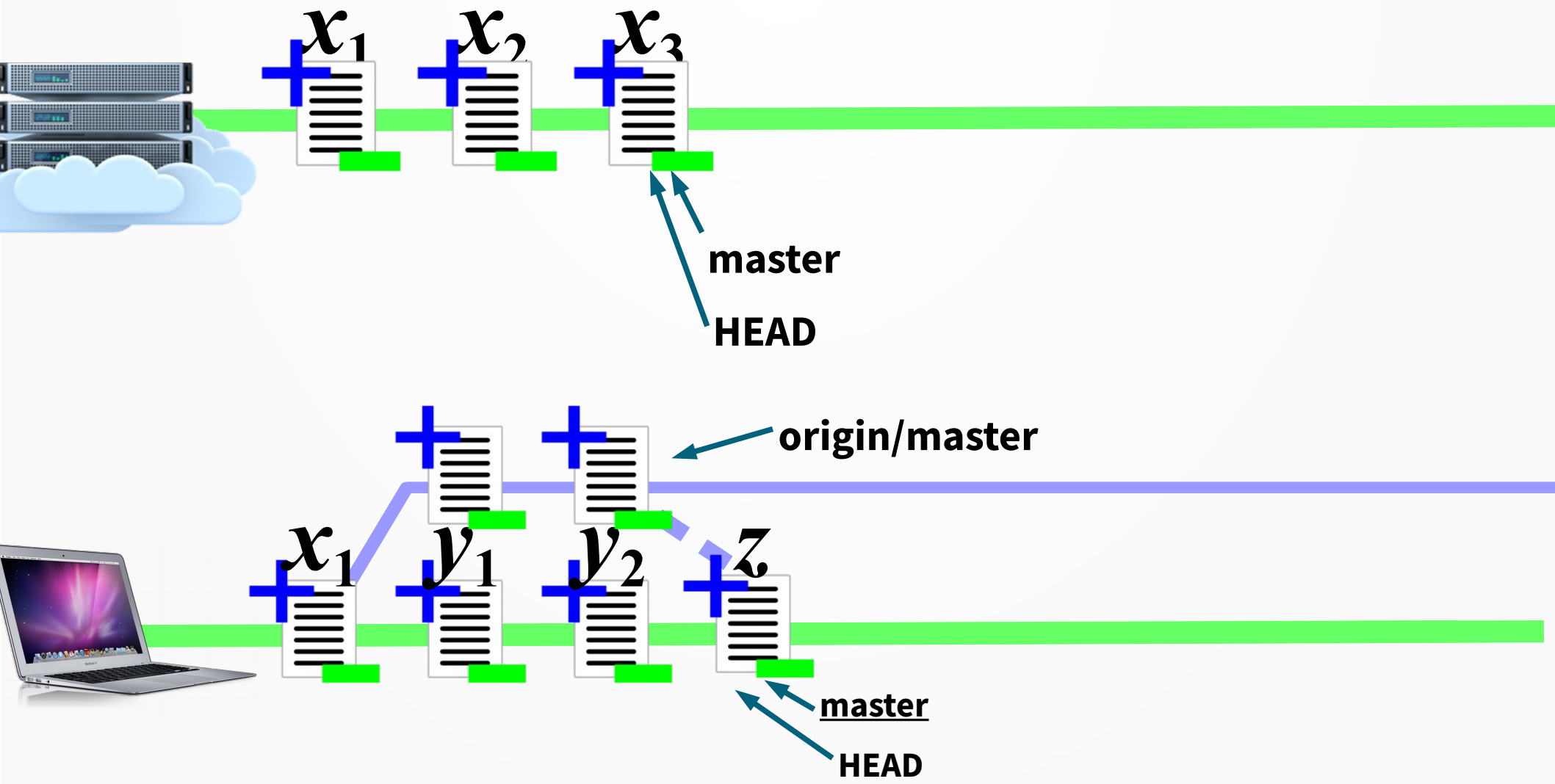


Synchronization




```
git merge origin/master
```

Synchronization




```
git commit
```