



# Versionskontrolle mit git

Tillmann Rendel  
University of Tübingen



# Was ist Versionskontrolle?

Ein Versionskontrollsystem für Quellcode speichert den Inhalt aller Dateien eines Projektes und ihre Änderungsgeschichte:

**Wer** hat **wann** und **warum**  
**welche** Dateien **wie** geändert?



# Wozu Versionskontrolle?

- Zurücksetzen des Projektes auf früheren Stand
- Dokumentation der Projektgeschichte und der Beiträge verschiedener Entwickler
- Entwicklung mehrerer Entwicklungszweige
- Koordination der Arbeit verschiedener Entwickler



# Begriffe

## Repository

Änderungsgeschichte und Dateiinhalte für alle Versionen

## Arbeitskopie

Kopie aller Dateien einer Version

## Status

Welche Dateien in der Working Copy gegenüber derselben Version im Repository geändert wurden

## Diff

Unterschiede zwischen Repository und Arbeitskopie oder anderer Version

## Check out

Dateiinhalte einer Version aus Repository in Working Copy kopieren.

## Commit

Derzeitigen Stand der Working Copy als neue Version in Repository kopieren



# Begriffe

## Repository

Änderungsgeschichte und Dateiinhalte für alle Versionen

## Arbeitskopie

Kopie aller Dateien einer Version

## Status

Welche Dateien in der Working Copy gegenüber derselben Version im Repository geändert wurden

## Diff

Unterschiede zwischen Repository und Arbeitskopie oder anderer Version

## Check out

Dateiinhalte einer Version aus Repository in Working Copy kopieren.

## Commit

Derzeitigen Stand der Working Copy als neue Version in Repository kopieren



# Begriffe

## Repository

Änderungsgeschichte und Dateiinhalte für alle Versionen

## Arbeitskopie

Kopie aller Dateien einer Version

## Status

Welche Dateien in der Working Copy gegenüber derselben Version im Repository geändert wurden

## Diff

Unterschiede zwischen Repository und Arbeitskopie oder anderer Version

## Check out

Dateiinhalte einer Version aus Repository in Working Copy kopieren.

## Commit

Derzeitigen Stand der Working Copy als neue Version in Repository kopieren



# Begriffe

## Repository

Änderungsgeschichte und Dateiinhalte für alle Versionen

## Arbeitskopie

Kopie aller Dateien einer Version

## Status

Welche Dateien in der Working Copy gegenüber derselben Version im Repository geändert wurden

## Diff

Unterschiede zwischen Repository und Arbeitskopie oder anderer Version

## Check out

Dateiinhalte einer Version aus Repository in Working Copy kopieren.

## Commit

Derzeitigen Stand der Working Copy als neue Version in Repository kopieren



# Begriffe

## Repository

Änderungsgeschichte und Dateiinhalte für alle Versionen

## Arbeitskopie

Kopie aller Dateien einer Version

## Status

Welche Dateien in der Working Copy gegenüber derselben Version im Repository geändert wurden

## Diff

Unterschiede zwischen Repository und Arbeitskopie oder anderer Version

## Check out

Dateiinhalte einer Version aus Repository in Working Copy kopieren.

## Commit

Derzeitigen Stand der Working Copy als neue Version in Repository kopieren





```
answer = 7
factor = 3
while (answer < 60) {
    answer *= factor
    factor += 1
}
```

vorher

# Das Tübinger Softwareprojekt

FB Informatik · AG Programmiersprachen und Softwaretechnik

## Repository

Änderungsgeschichte und Dateiinhalte für alle Versionen

## Arbeitskopie

Kopie aller Dateien einer Version

## Status

Welche Dateien in der Working Copy gegenüber derselben Version im Repository geändert wurden

## Diff

Unterschiede zwischen Repository und Arbeitskopie oder anderer Version

## Check out

Dateiinhalte einer Version aus Repository in Working Copy kopieren.

## Commit

Derzeitigen Stand der Working Copy als neue Version in Repository kopieren

```
answer = 7
factor = 3
while (answer < 60) {
    answer *= factor
    factor += 1
}
```

vorher

```
answer = 7
factor = 2
while (answer < 40) {
    answer *= factor
    factor += 1
}
```

nachher

Änderungsgeschichte und Dateiinhalte für alle Versionen

## Arbeitskopie

Kopie aller Dateien einer Version

## Status

Welche Dateien in der Working Copy gegenüber derselben Version im Repository geändert wurden

Unterschiede zwischen Repository und Arbeitskopie oder anderer Version

## Check out

Dateiinhalte einer Version aus Repository in Working Copy kopieren.

## Commit

Derzeitigen Stand der Working Copy als neue Version in Repository kopieren

```
answer = 7
factor = 3
while (answer < 60) {
    answer *= factor
    factor += 1
}
```

vorher

```
answer = 7
factor = 2
while (answer < 40) {
    answer *= factor
    factor += 1
}
```

nachher

```
answer = 7
-factor = 3
-while (answer < 60) {
+factor = 2
+while (answer < 40) {
    answer *= factor
```

diff



# Begriffe

## Repository

Änderungsgeschichte und Dateiinhalte für alle Versionen

## Arbeitskopie

Kopie aller Dateien einer Version

## Status

Welche Dateien in der Working Copy gegenüber derselben Version im Repository geändert wurden

## Diff

Unterschiede zwischen Repository und Arbeitskopie oder anderer Version

## Check out

Dateiinhalte einer Version aus Repository in Working Copy kopieren.

## Commit

Derzeitigen Stand der Working Copy als neue Version in Repository kopieren



# Begriffe

## Repository

Änderungsgeschichte und Dateiinhalte für alle Versionen

## Arbeitskopie

Kopie aller Dateien einer Version

## Status

Welche Dateien in der Working Copy gegenüber derselben Version im Repository geändert wurden

## Diff

Unterschiede zwischen Repository und Arbeitskopie oder anderer Version

## Check out

Dateiinhalte einer Version aus Repository in Working Copy kopieren.

## Commit

Derzeitigen Stand der Working Copy als neue Version in Repository kopieren



# Begriffe

## Repository

Änderungsgeschichte und Dateiinhalte für alle Versionen

## Arbeitskopie

Kopie aller Dateien einer Version

## Status

Welche Dateien in der Working Copy gegenüber derselben Version im Repository geändert wurden

## Diff

Unterschiede zwischen Repository und Arbeitskopie oder anderer Version

## Check out

Dateiinhalte einer Version aus Repository in Working Copy kopieren.

## Commit

Derzeitigen Stand der Working Copy als neue Version in Repository kopieren



# Technische Aspekte

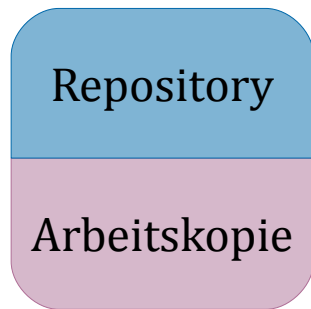
- Wo ist Repository gespeichert?
  - Lokal, zentral oder verteilt?
- Was ist im Repository gespeichert?
  - Snapshot-basiert oder diff-basiert



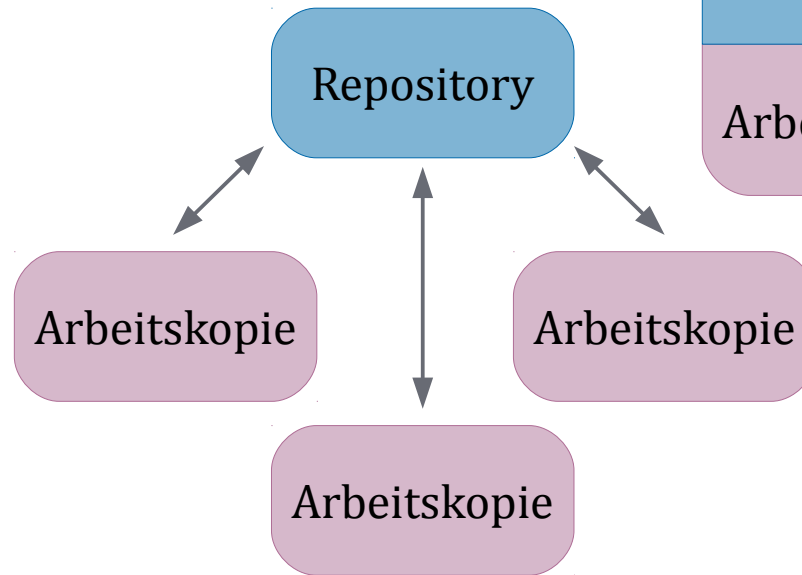


# Wo ist Repository gespeichert?

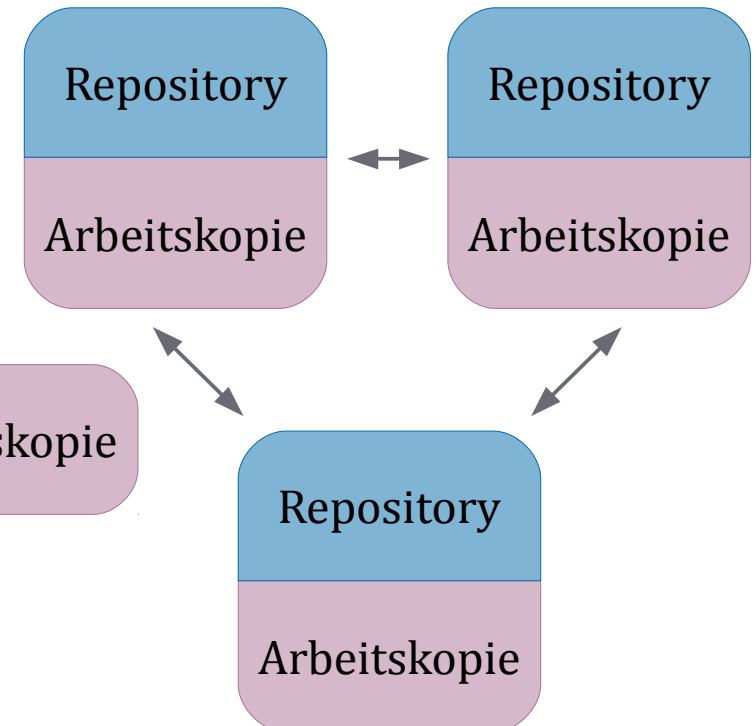
Lokal



Zentral

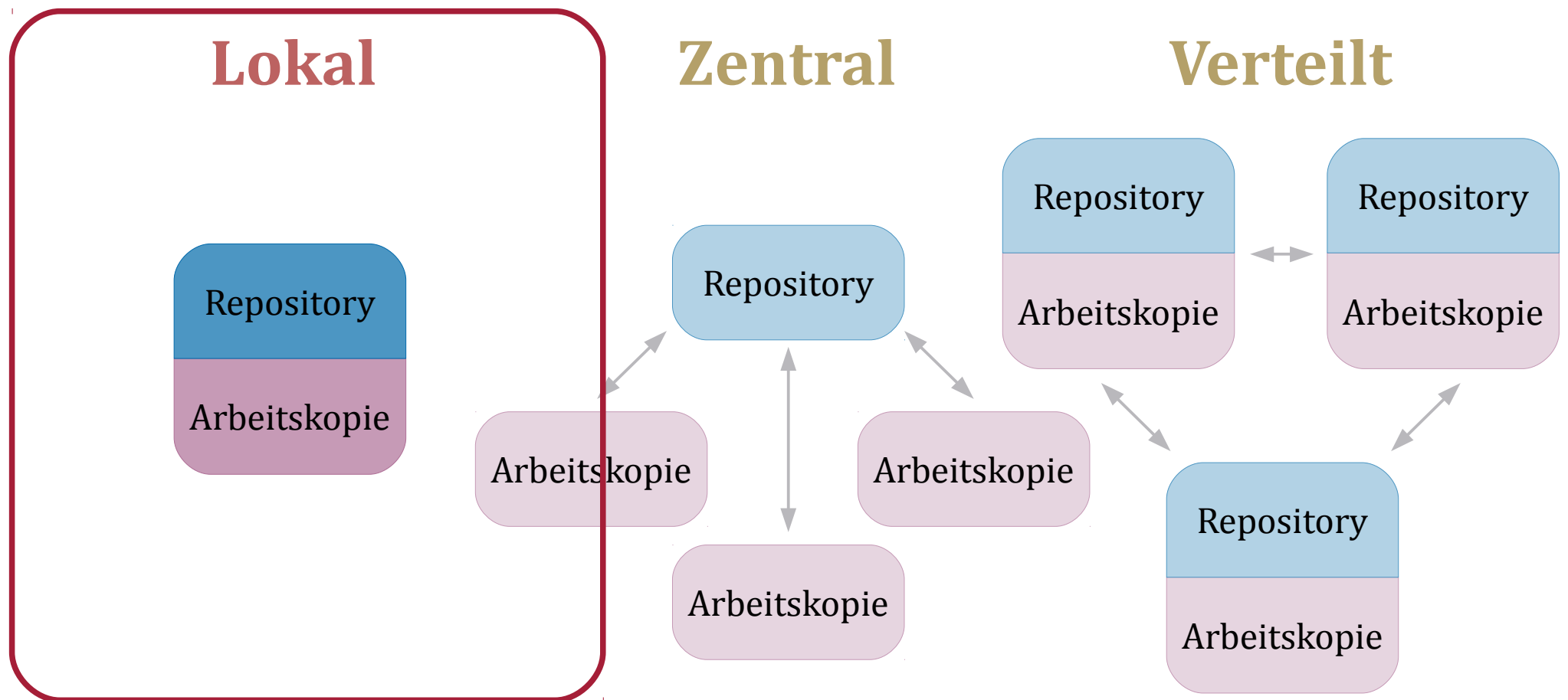


Verteilt



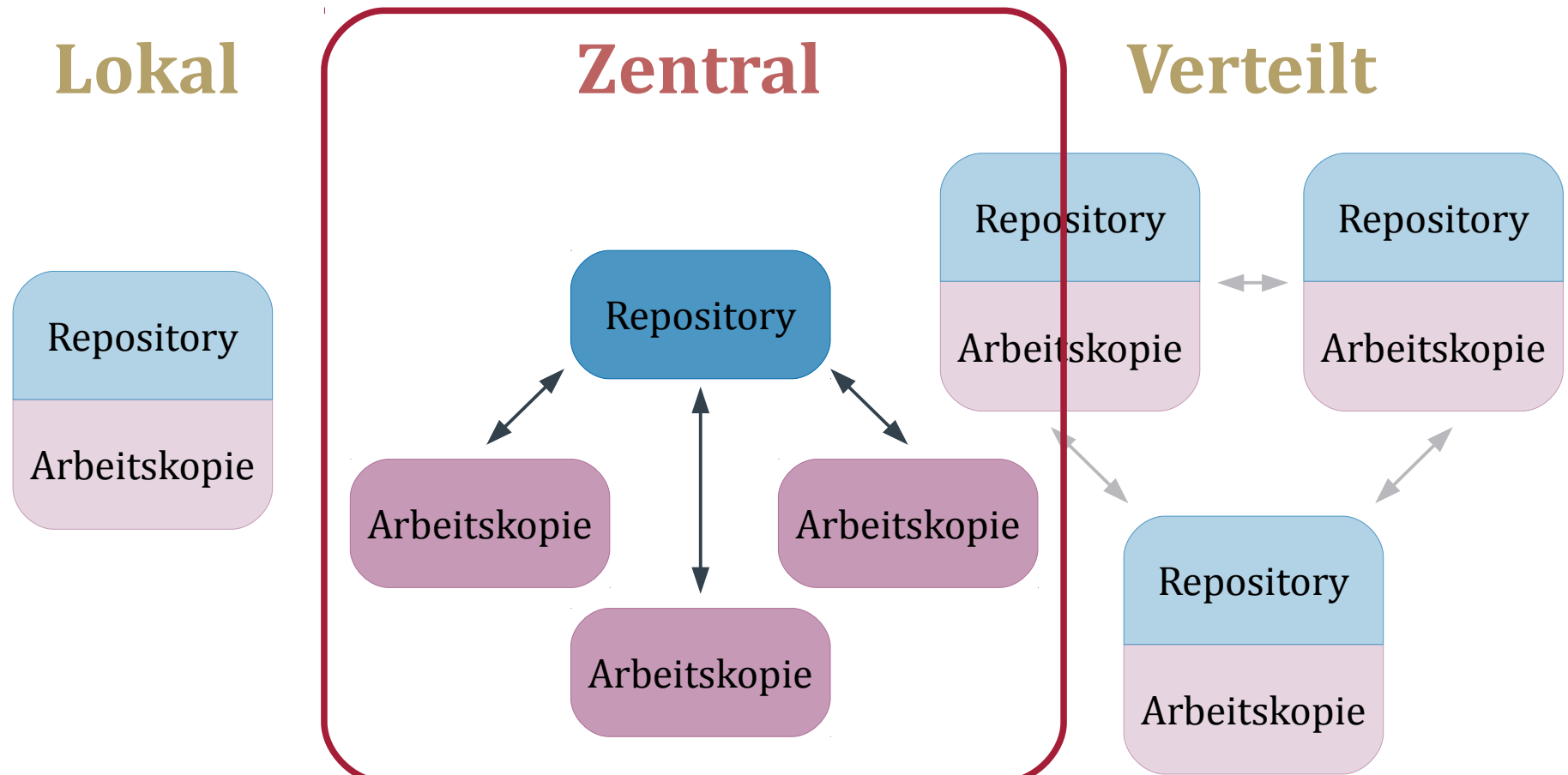


# Wo ist Repository gespeichert?





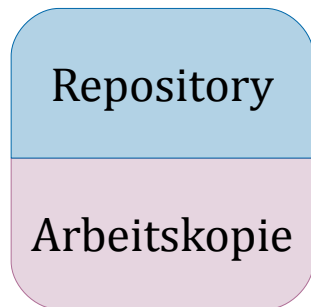
# Wo ist Repository gespeichert?



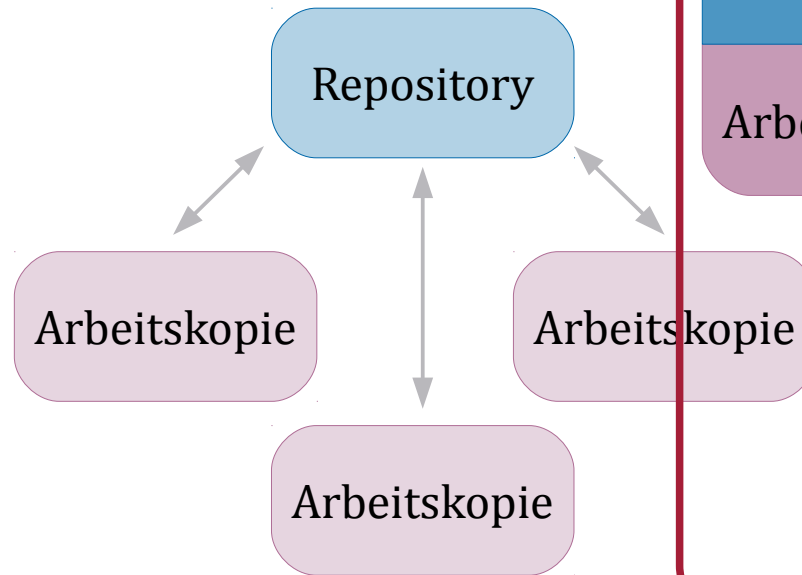


# Wo ist Repository gespeichert?

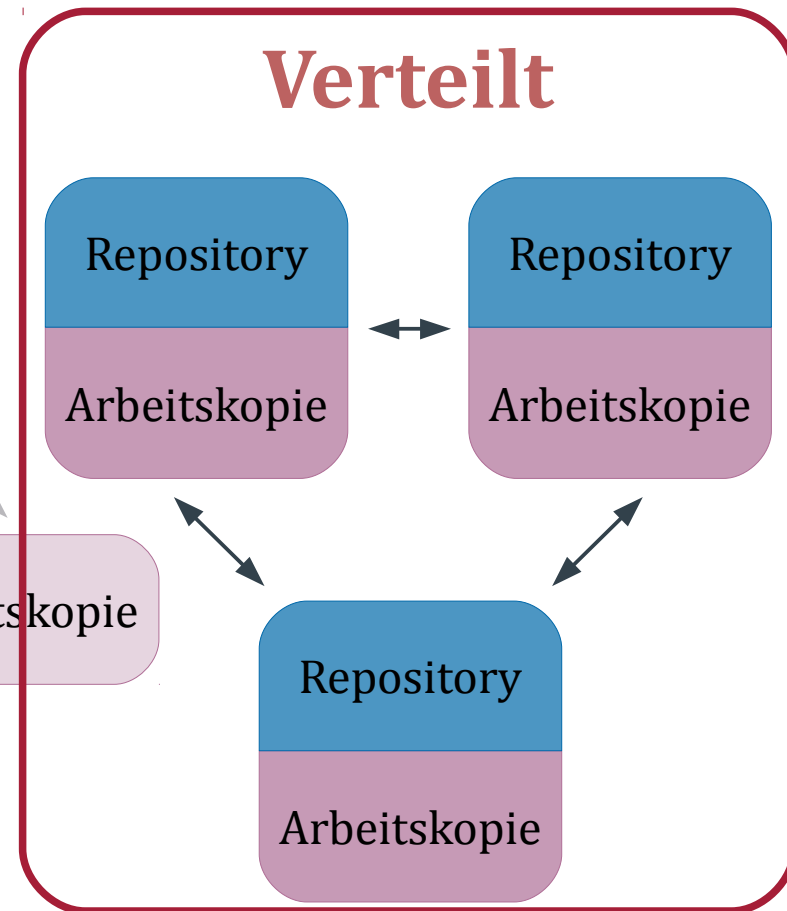
## Lokal



## Zentral



## Verteilt





# Was ist im Repository gespeichert

## Diff

- Speichere Diffs zwischen Versionen
- Generiere Snapshots bei Bedarf

## Snapshot

- Speichere Snapshots der Versionen
- Generiere Diffs bei Bedarf



# Was ist im Repository gespeichert

## Diff

- Speichere Diffs zwischen Versionen
- Generiere Snapshots bei Bedarf

## Snapshot

- Speichere Snapshots der Versionen
- Generiere Diffs bei Bedarf



# Was ist im Repository gespeichert

## Diff

- Speichere Diffs zwischen Versionen
- Generiere Snapshots bei Bedarf

## Snapshot

- Speichere Snapshots der Versionen
- Generiere Diffs bei Bedarf



# Was ist git?

- Verteiltes Versionskontrollsystem
- Ursprünglich entwickelt von Linus Torvalds zur Versionskontrolle des Linux-Kernels
- Heute sowohl für Open-Source als auch für proprietäre Entwicklung eingesetzt





# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle
- Effiziente Speicherung durch Adressierung via Hash
- Index



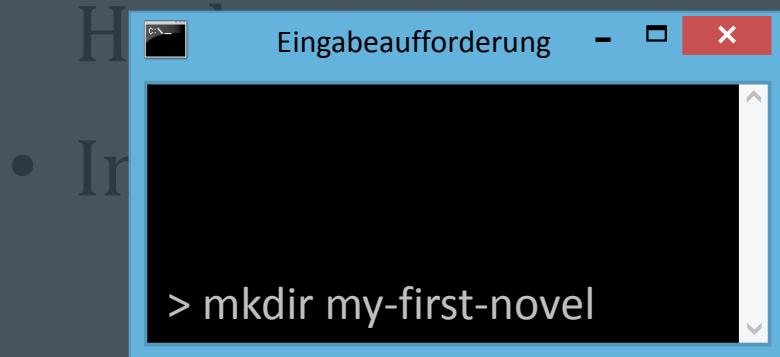
# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle
- Effiziente Speicherung durch Adressierung via Hash
- Index



# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle
- Effiziente Speicherung durch Adressierung via Hash



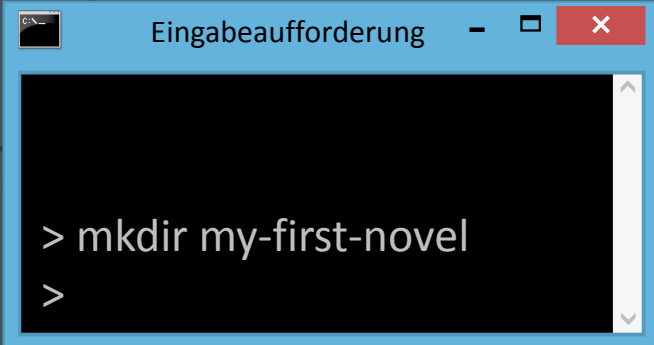


# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle
- Effiziente Speicherung durch Adressierung via Hash



- In der Kommandozeile



```
> mkdir my-first-novel
>
```



# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle
- Effiziente Speicherung durch Adressierung via Hash



```
Eingabeaufforderung - [ ] [X]  
> mkdir my-first-novel  
> cd my-first-novel  
>
```



# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle
- Effiziente Speicherung durch Adressierung via Hash



- In einem Terminalfenster:

```
Eingabeaufforderung - [ ] [X]
> mkdir my-first-novel
> cd my-first-novel
> git init
```



# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle
- Effiziente Speicherung durch Adressierung via

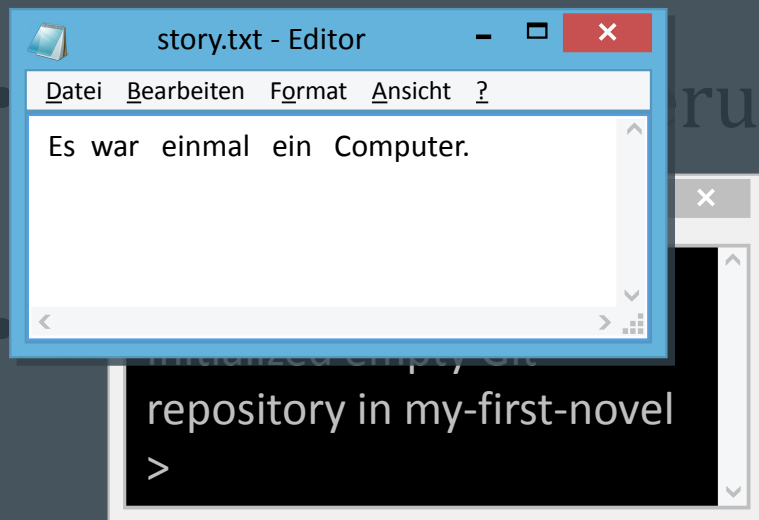


```
Eingabeaufforderung - [ ] [X]  
> git init  
Initialized empty Git  
repository in my-first-novel  
>
```



# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle



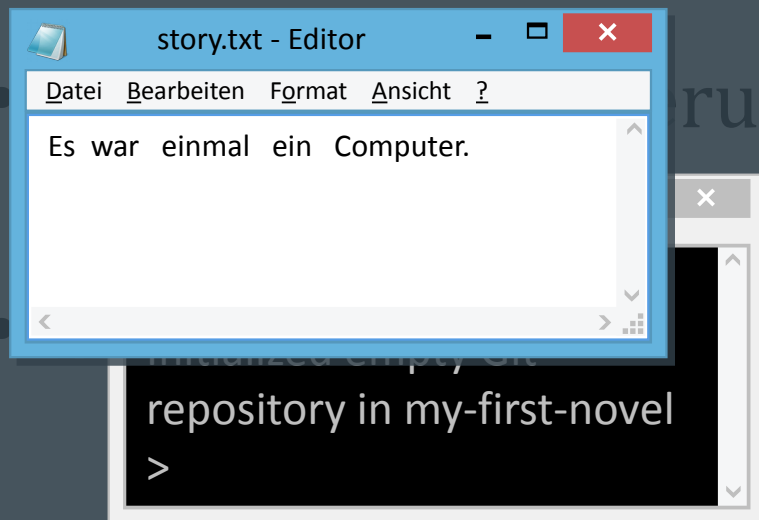
...ung durch Ad...ssierung via





# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle





# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle

```
story.txt - Editor
Datei Bearbeiten Format Ansicht ?
Es war einmal ein Computer.

Eingabeaufforderung
> git init
Initialized empty Git
repository in my-first-novel
> git add story.txt
```





index

# Wie funktioniert git?

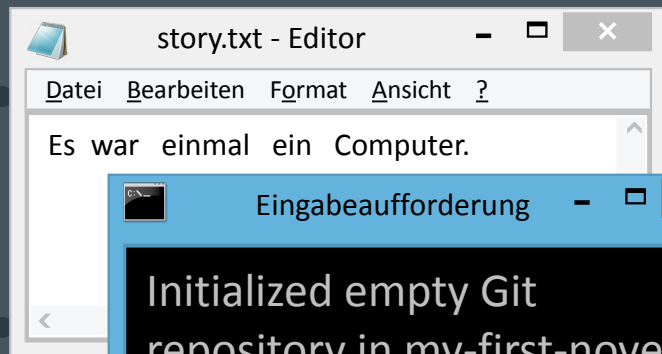


story.txt

- verteilt

- Snapshot-basiert

- Viele Kommandozeilenbefehle





**index**

# Wie funktioniert git?

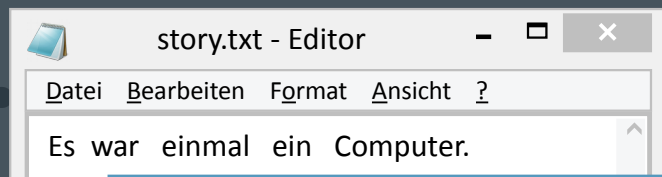


**story.txt**

- verteilt

- Snapshot-basiert

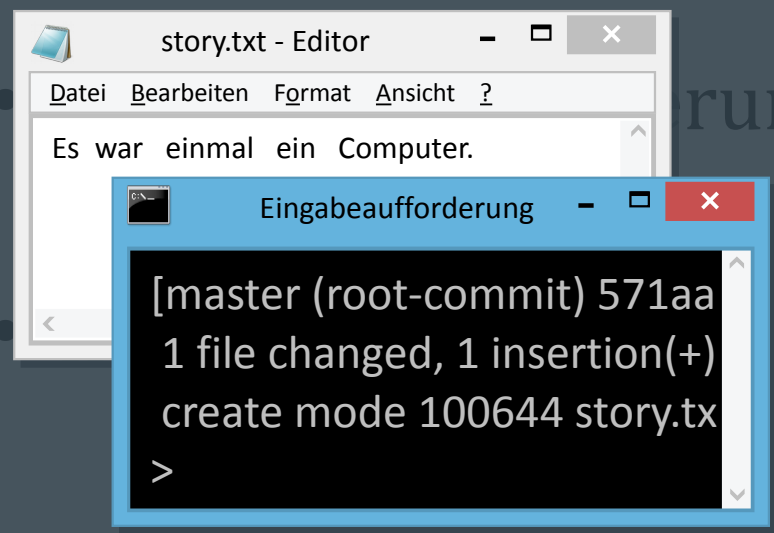
- Viele Kommandozeilenbefehle



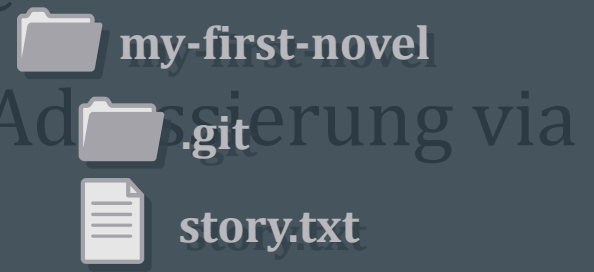
# Wie funktioniert git?



- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle



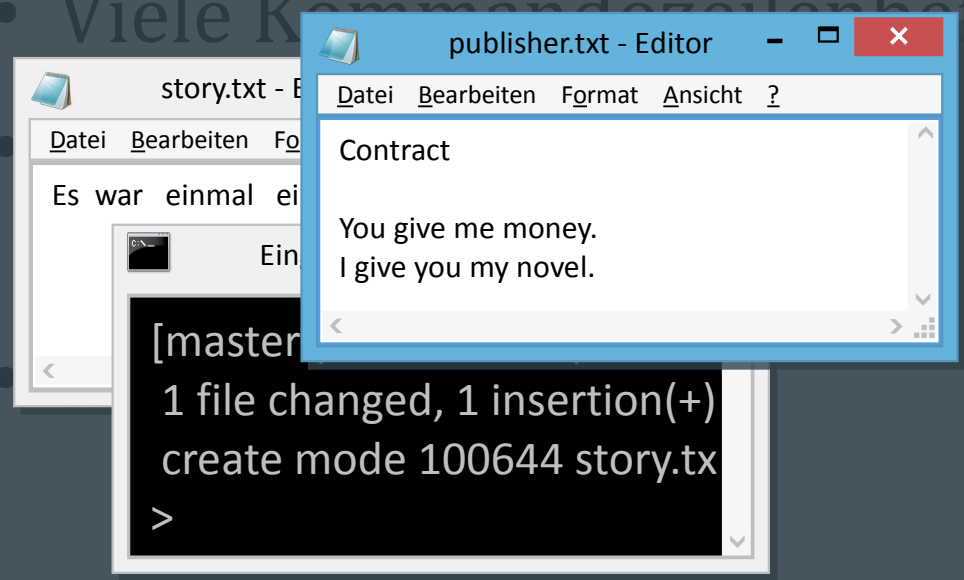
```
[master (root-commit) 571aa] 1 file changed, 1 insertion(+); create mode 100644 story.txt
```



# Wie funktioniert git?



- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle



```
[master] 1 file changed, 1 insertion(+)  
create mode 100644 story.tx  
>
```



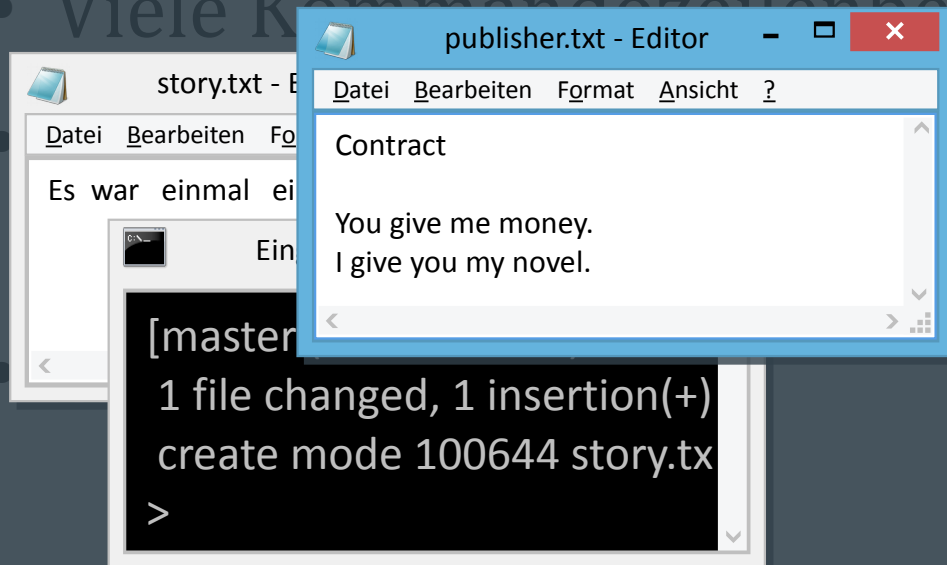
## Wie funktioniert git?



- verteilt

- Snapshot-basiert

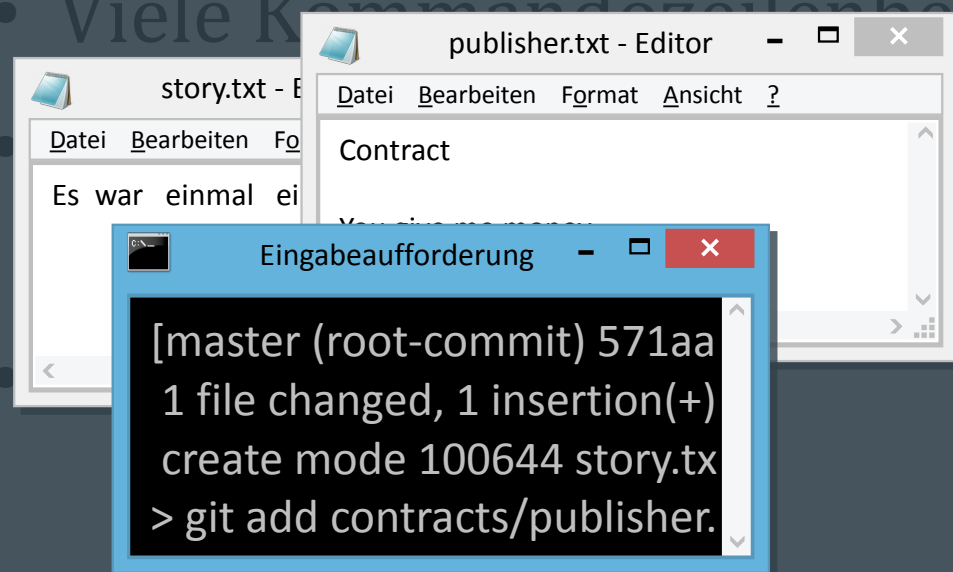
- Viele Kommandozeilenbefehle



# Wie funktioniert git?



- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle



```
[master (root-commit) 571aa] 1 file changed, 1 insertion(+)  
create mode 100644 story.txt  
> git add contracts/publisher.
```





HEAD → master

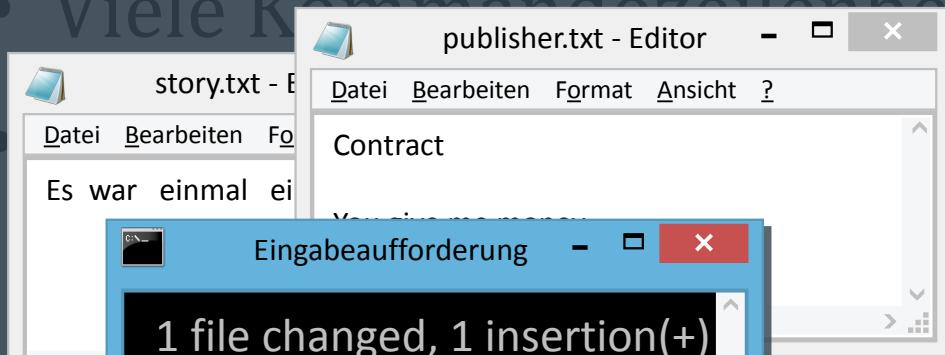
tree

index

## Wie funktioniert git?



- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle



```
Eingabeaufforderung - [X]
1 file changed, 1 insertion(+)
create mode 100644 story.tx
> git add contracts/publisher.
>
```



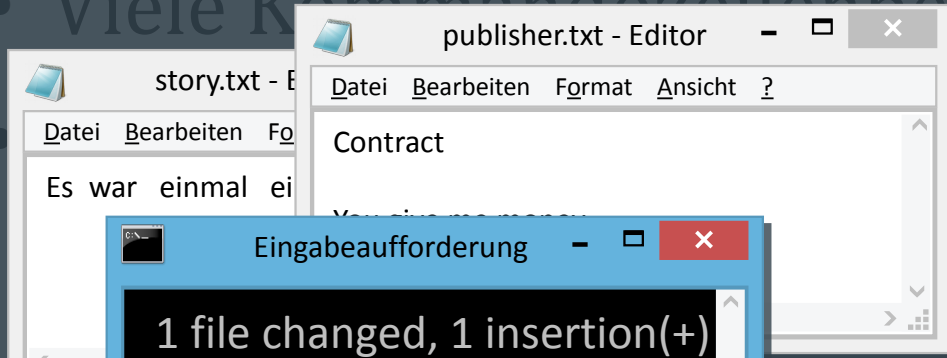
tree

index

# Wie funktioniert git?

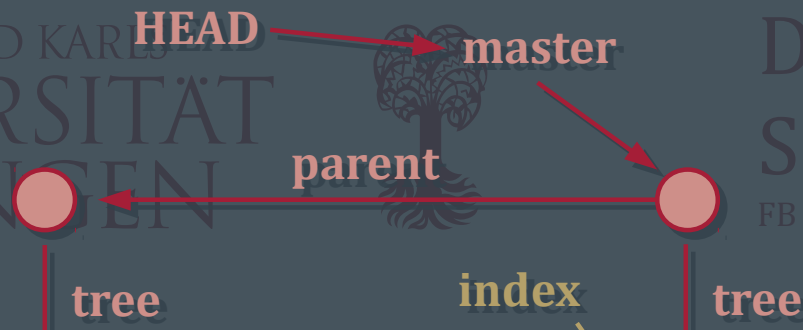


- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle



```
1 file changed, 1 insertion(+)  
create mode 100644 story.tx  
> git add contracts/publisher.  
> git commit
```





# Wie funktioniert git?

story.txt

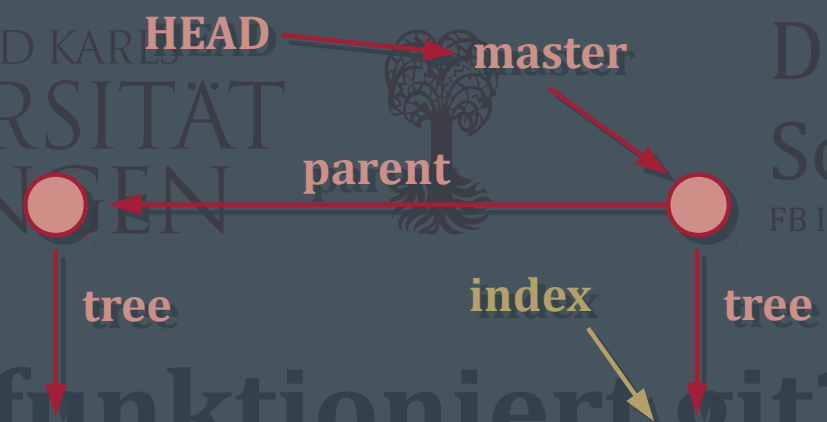
story.txt  
contracts  
publisher.txt

- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle

publisher.txt - Editor  
Datei Bearbeiten Format Ansicht ?  
Contract

```
[master bbed7d4] bar  
1 file changed, 5 insertions(+)  
create mode 100644 contrac  
>
```

my-first-novel  
├── .git  
├── story.txt  
└── contracts  
 └── publisher.txt

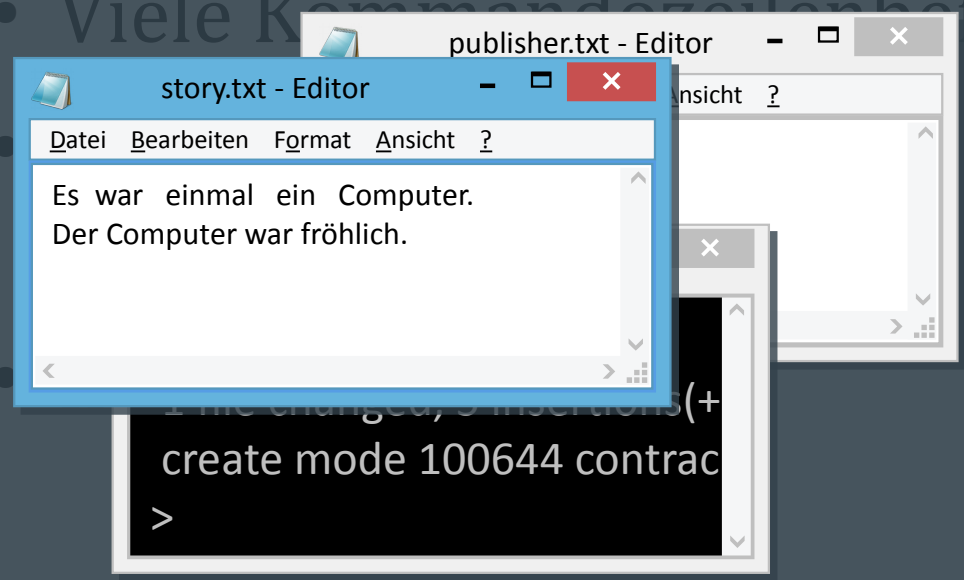


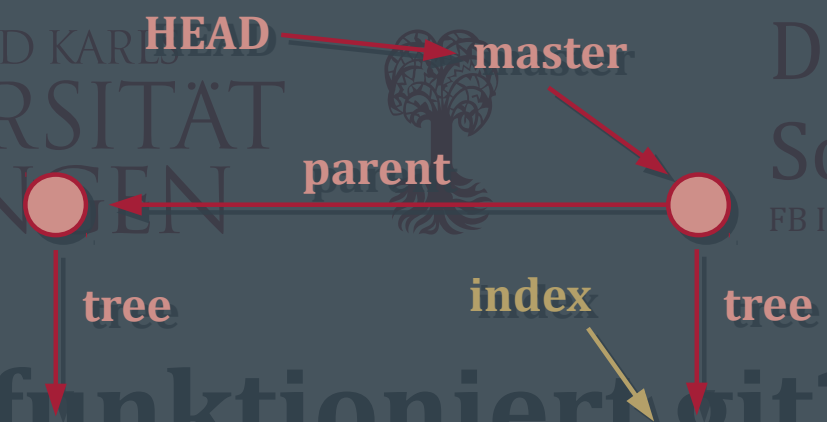
# Wie funktioniert git?

story.txt

story.txt  
contracts  
publisher.txt

- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle
- ...





# Wie funktioniert git?

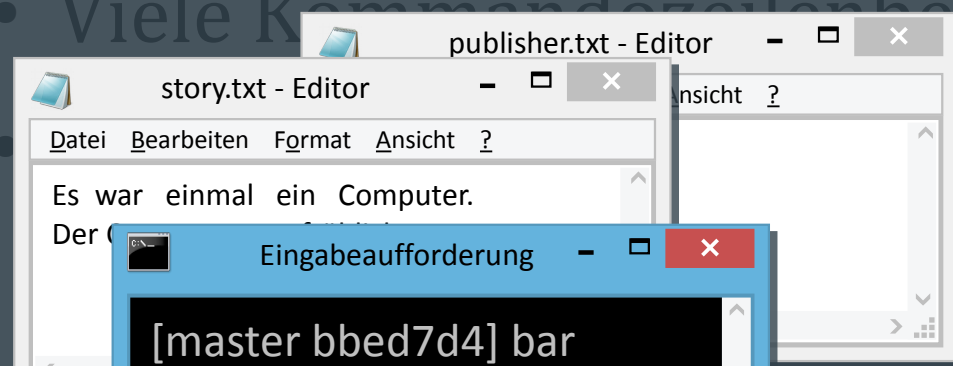
story.txt

story.txt

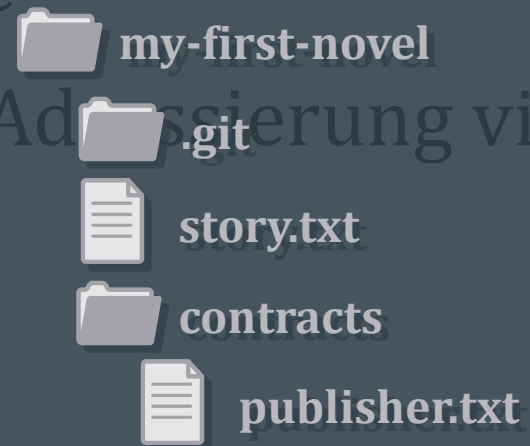
contracts

publisher.txt

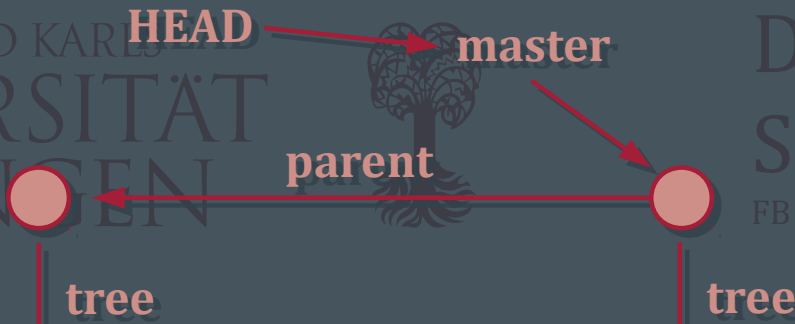
- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle



```
[master bbed7d4] bar
1 file changed, 5 insertions(+
create mode 100644 contrac
> git add story.txt
```



## Wie funktioniert git?



index

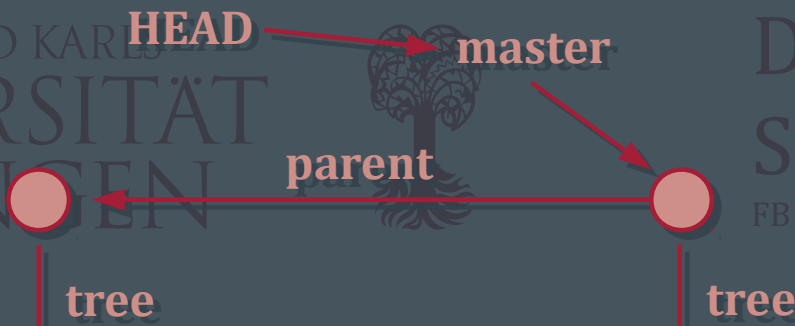


- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle

```
publisher.txt - Editor  
story.txt - Editor  
Datei Bearbeiten Format Ansicht ?  
Es war einmal ein Computer.  
Der C  
Eingabeaufforderung  
1 file changed, 5 insertions(+  
create mode 100644 contrac  
> git add story.txt  
>
```



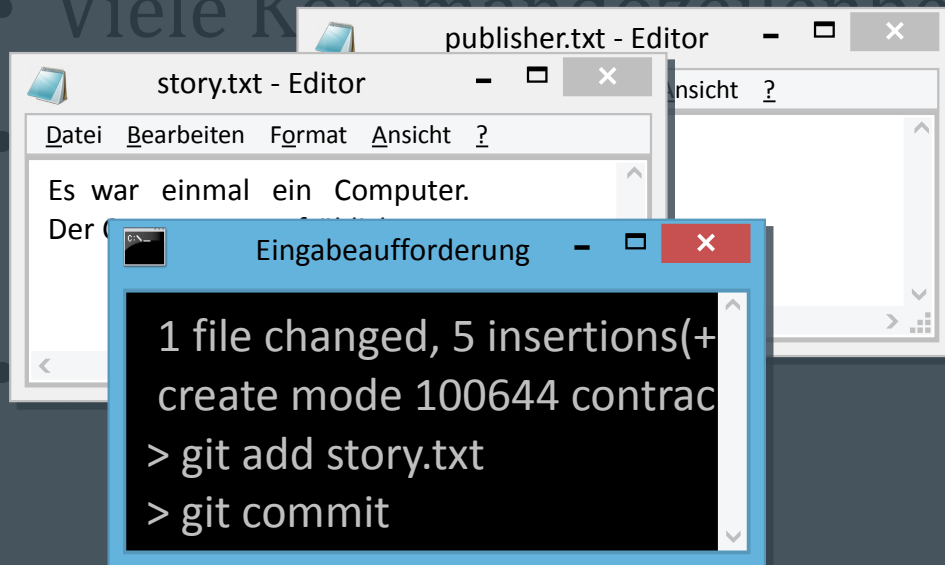
## Wie funktioniert git?

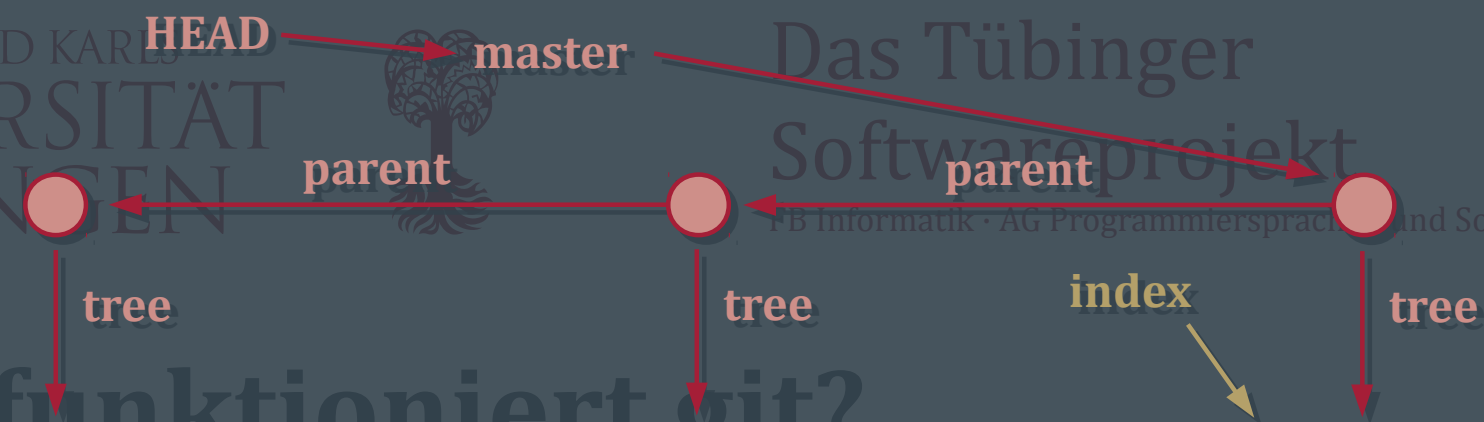


index



- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle





story.txt

story.txt  
contracts  
publisher.txt

story.txt  
contracts  
publisher.txt

- verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle

```
> git commit
[master e8eedd5] baz
1 file changed, 1 insertion(+)
>
```

my-first-novel  
.git  
story.txt  
contracts  
publisher.txt





# Wie funktioniert git?

- Verteilt
- Snapshot-basiert
- Viele Kommandozeilenbefehle
- Effiziente Speicherung durch Adressierung via Hash
- Index



## Demo 1

*(init, status, diff, add, diff --staged, checkout, commit, show, log, reflog)*



## Befehle (1/3)

- **git help**

*Liste wichtiger Befehle*

- **git help command**

*Hilfe zu einzelnen Befehlen*

- **git init**

*Repository und Arbeitskopie  
initialisieren*

- **git add**

*Arbeitskopie → Index*

- **git checkout**

*Index → Arbeitskopie*

- **git commit**

*Index → neuer Commit*



## Befehle (2/3)

- **git status**

*Informationen zu Arbeitskopie  
und HEAD zeigen*

- **git show commit**

*Informationen zu Commit  
zeigen*

- **git reflog**

*Entwicklung von HEAD zeigen*

- **git diff**

*Unterschied zwischen  
Arbeitskopie und Index zeigen*

- **git diff --staged**

*Unterschiede zwischen Index  
und HEAD zeigen*

- **git log**

*Änderungsgeschichte zeigen*



# Branching / Merging >



# Branching / Merging > `git init`



# Branching / Merging

```
> git init  
>
```



# Branching / Merging

```
> git init
```

```
> git add
```





# Branching / Merging

```
> git init  
> git add  
>
```





# Branching / Merging

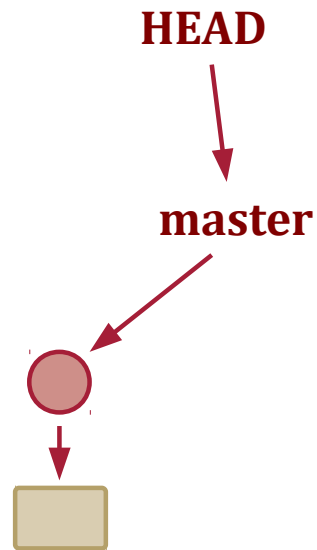
- > `git init`
- > `git add`
- > `git commit`





## Branching / Merging

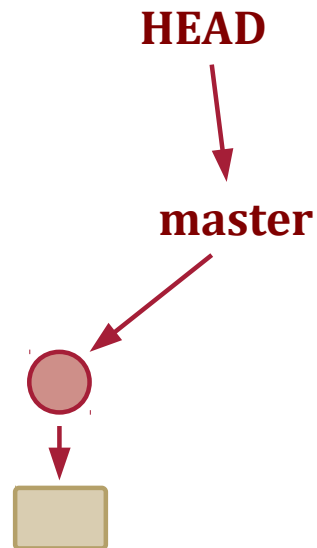
```
> git init  
> git add  
> git commit  
>
```





## Branching / Merging

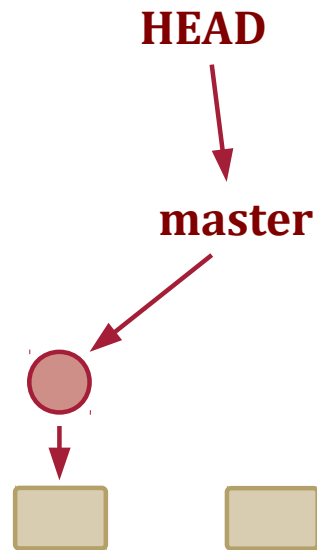
- > `git init`
- > `git add`
- > `git commit`
- > `git add`





## Branching / Merging

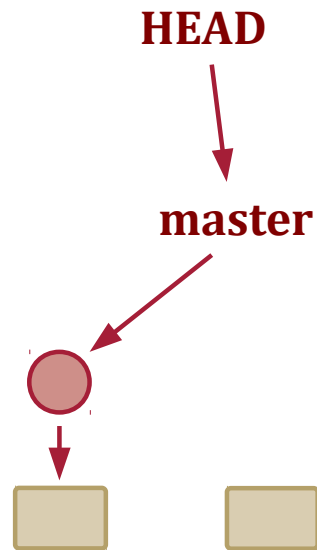
```
> git init  
> git add  
> git commit  
> git add  
>
```





## Branching / Merging

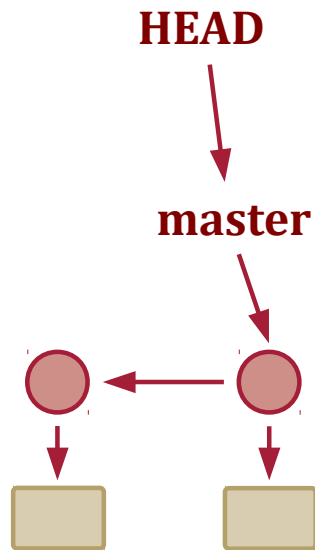
- > `git init`
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`





## Branching / Merging

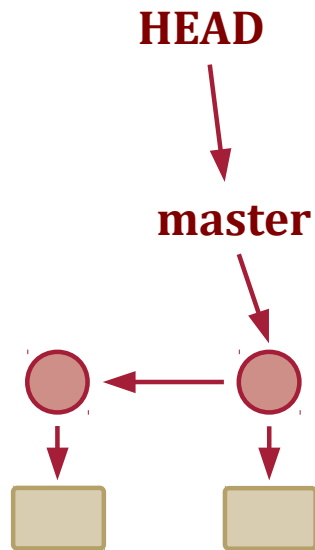
```
> git init  
> git add  
> git commit  
> git add  
> git commit  
>
```





## Branching / Merging

- > `git init`
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`
- > `git add`

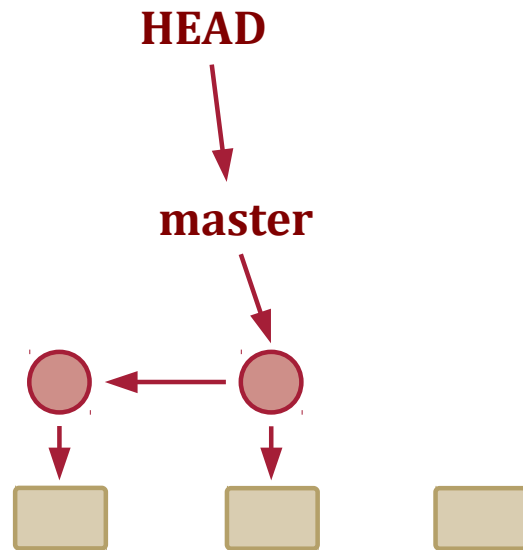






## Branching / Merging

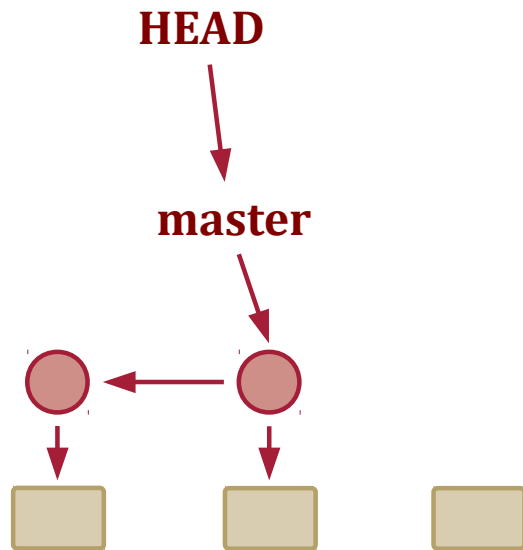
```
> git add  
> git commit  
> git add  
> git commit  
> git add  
>
```





## Branching / Merging

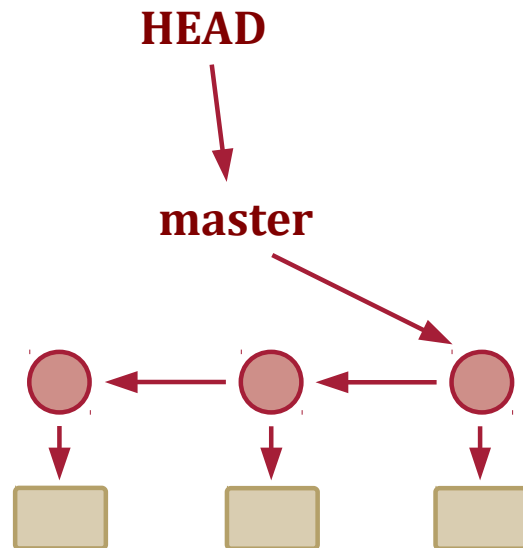
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`
- > `git add`
- > **`git commit`**





## Branching / Merging

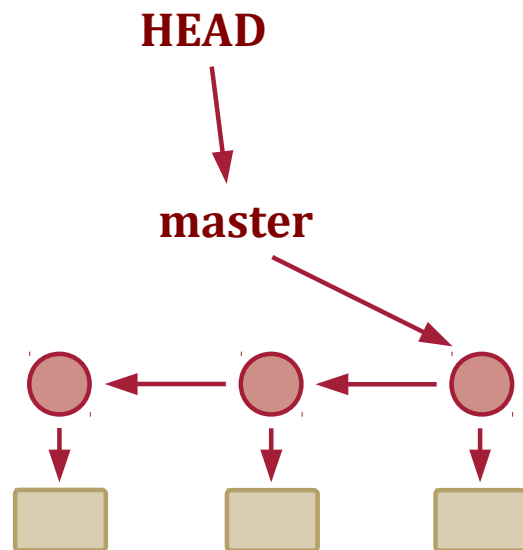
```
> git commit  
> git add  
> git commit  
> git add  
> git commit  
>
```





## Branching / Merging

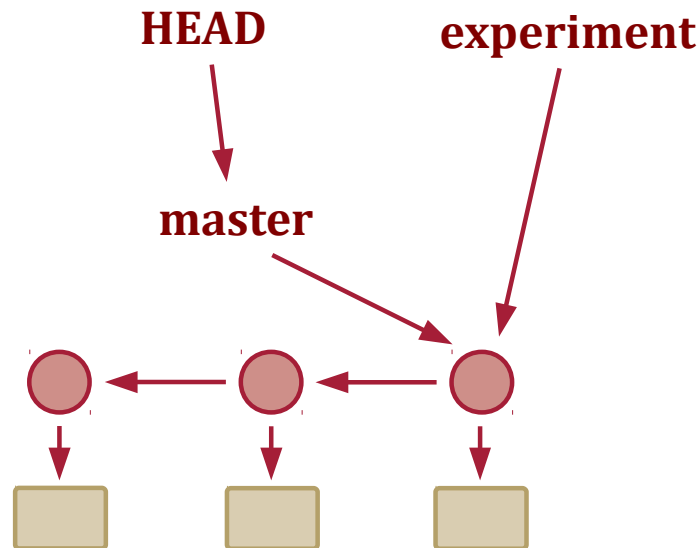
- > `git commit`
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`
- > `git branch experiment`





## Branching / Merging

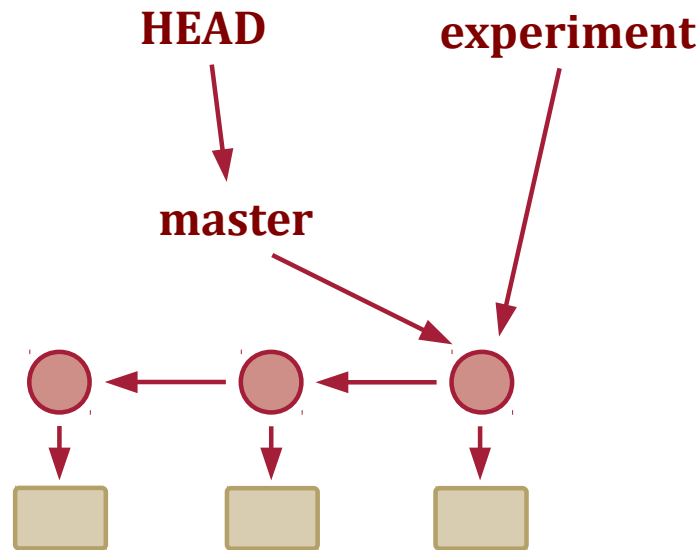
```
> git add  
> git commit  
> git add  
> git commit  
> git branch experiment  
>
```





## Branching / Merging

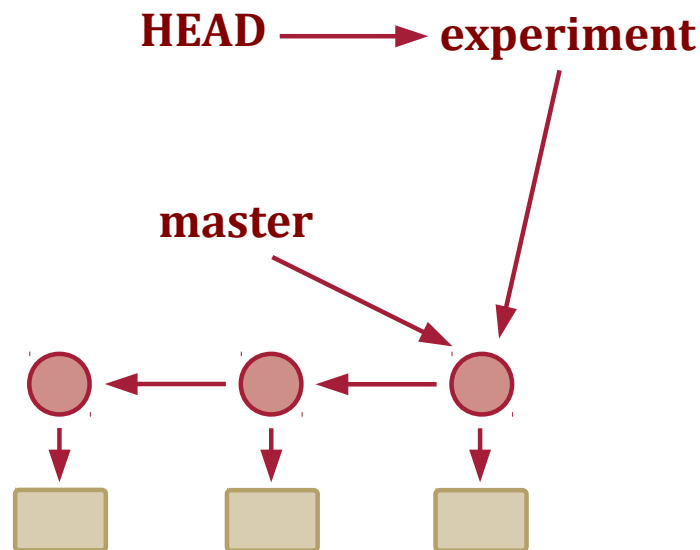
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`
- > `git branch experiment`
- > `git checkout experiment`





## Branching / Merging

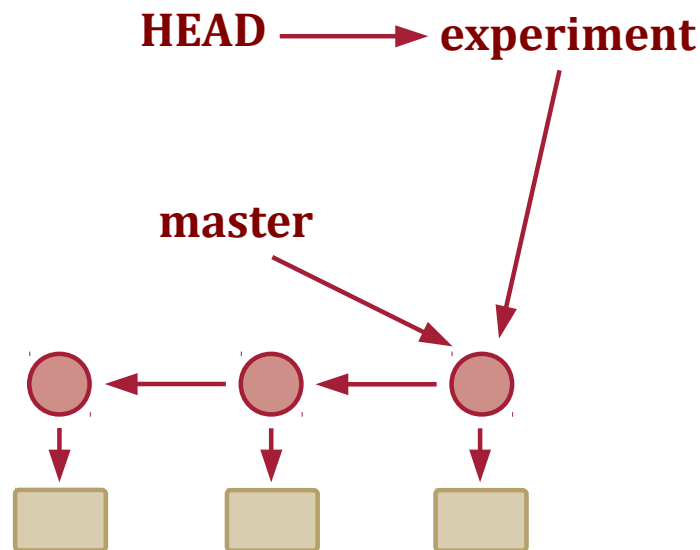
- > `git commit`
- > `git add`
- > `git commit`
- > `git branch experiment`
- > `git checkout experiment`
- >





## Branching / Merging

- > `git commit`
- > `git add`
- > `git commit`
- > `git branch experiment`
- > `git checkout experiment`
- > `git add`

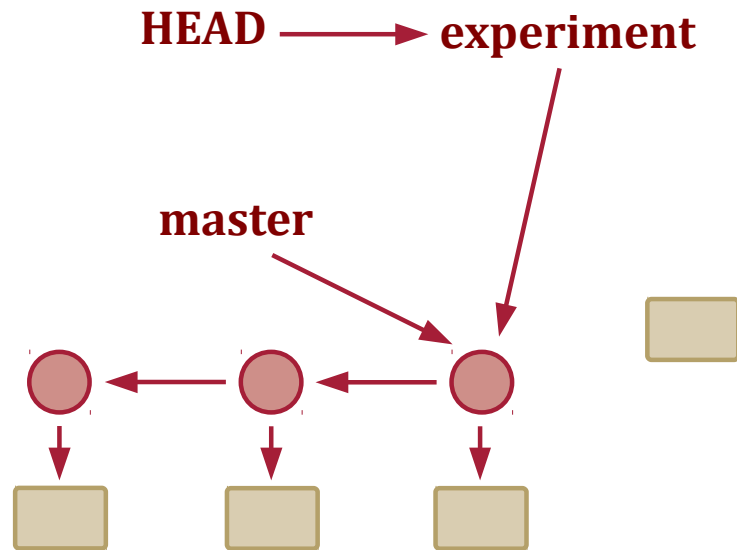






## Branching / Merging

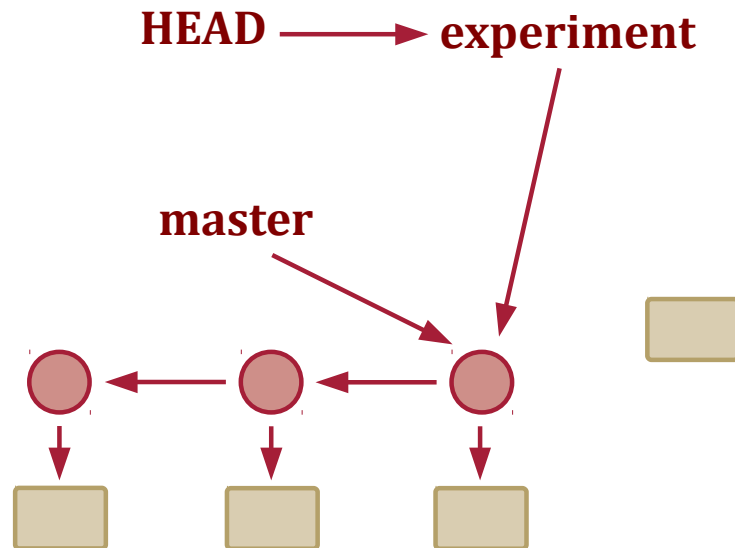
- > `git add`
- > `git commit`
- > `git branch experiment`
- > `git checkout experiment`
- > `git add`
- >





## Branching / Merging

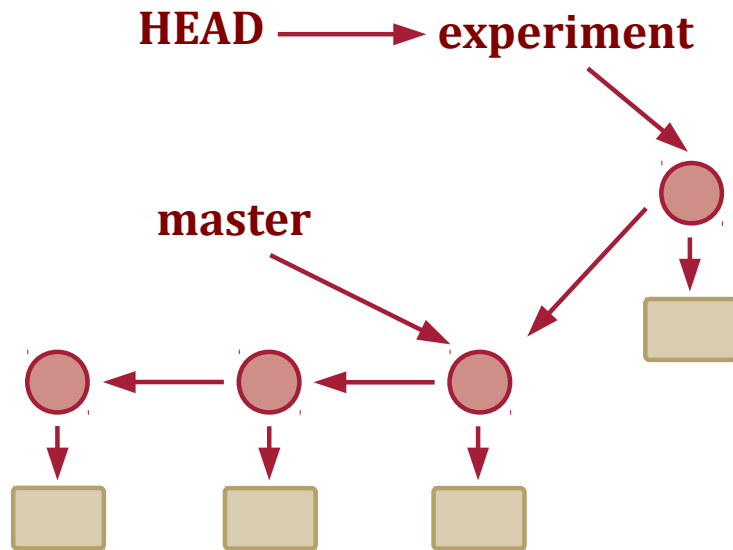
- > `git add`
- > `git commit`
- > `git branch experiment`
- > `git checkout experiment`
- > `git add`
- > **`git commit`**





## Branching / Merging

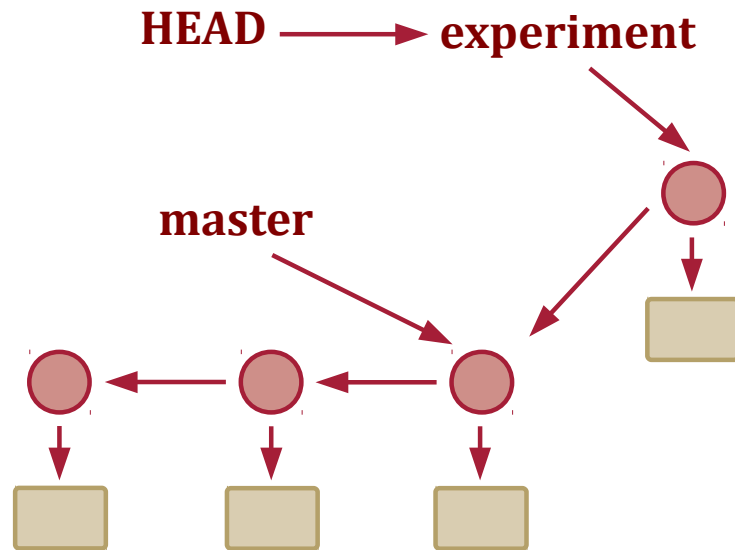
- > `git commit`
- > `git branch experiment`
- > `git checkout experiment`
- > `git add`
- > `git commit`
- >





## Branching / Merging

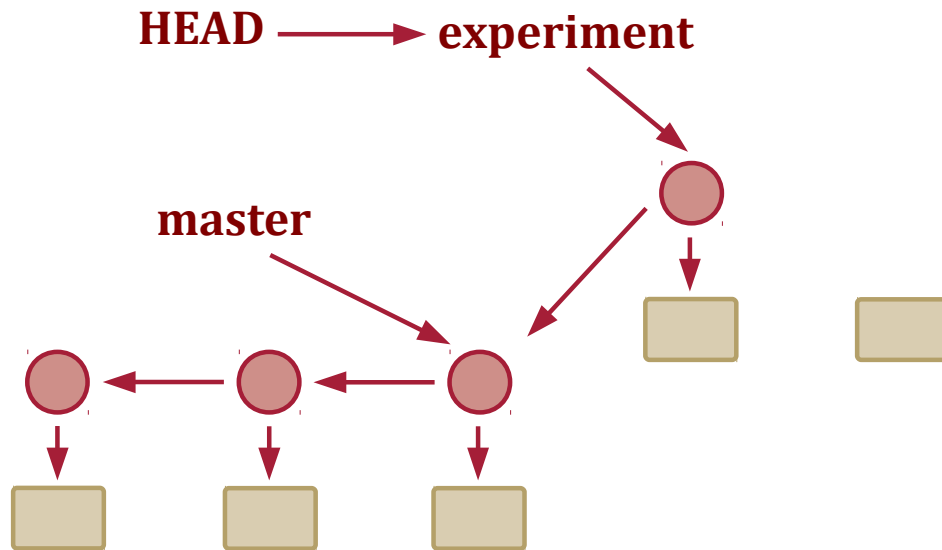
- > `git commit`
- > `git branch experiment`
- > `git checkout experiment`
- > `git add`
- > `git commit`
- > `git add`





## Branching / Merging

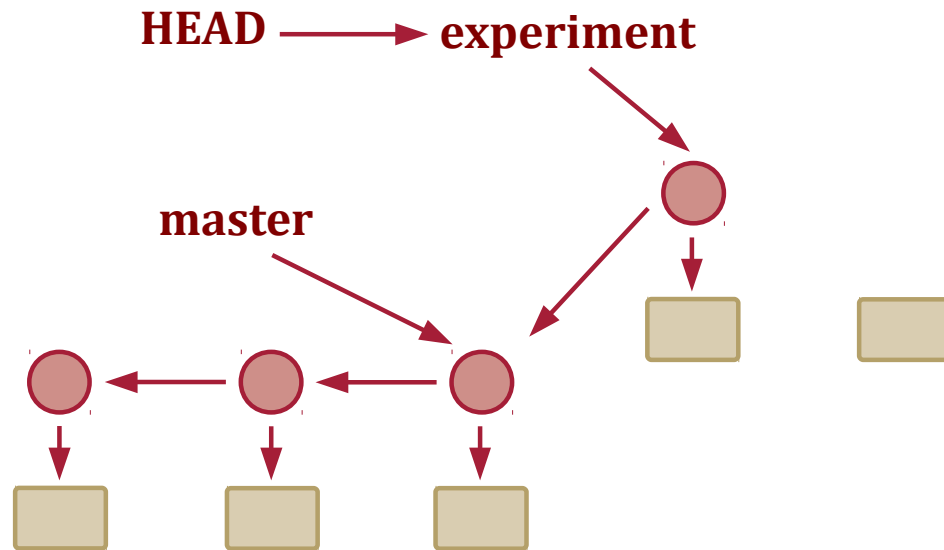
- > `git branch experiment`
- > `git checkout experiment`
- > `git add`
- > `git commit`
- > `git add`
- >





## Branching / Merging

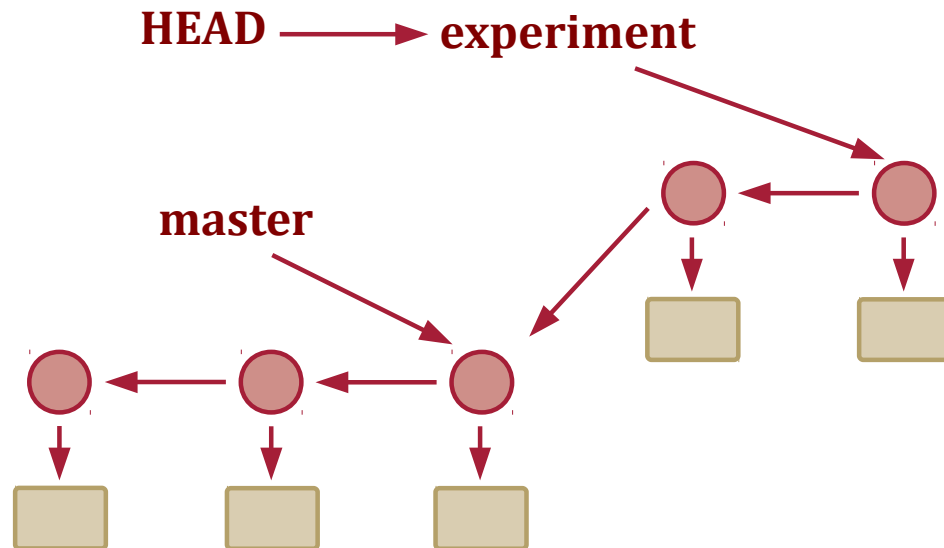
- > `git branch experiment`
- > `git checkout experiment`
- > `git add`
- > `git commit`
- > `git add`
- > **`git commit`**





# Branching / Merging

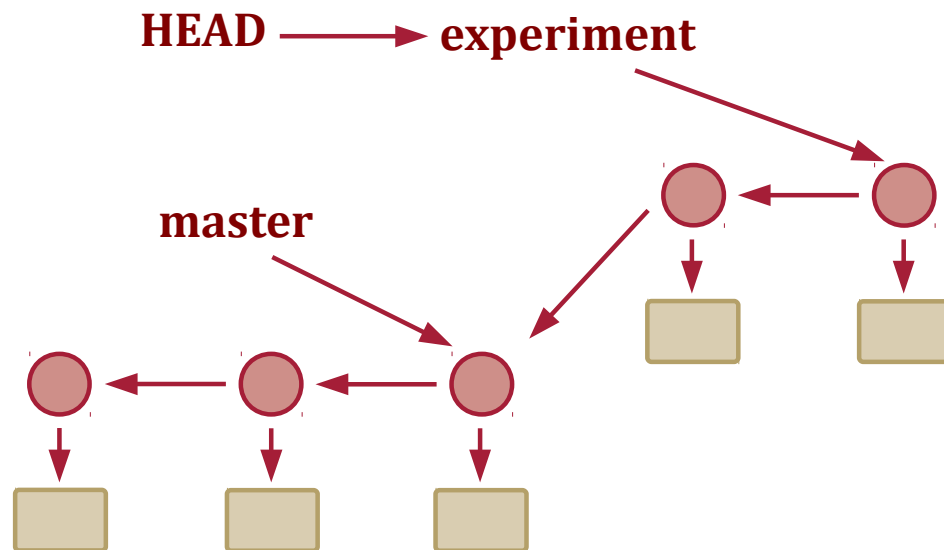
- > `git checkout experiment`
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`
- >





# Branching / Merging

- > `git checkout experiment`
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`
- > `git checkout master`

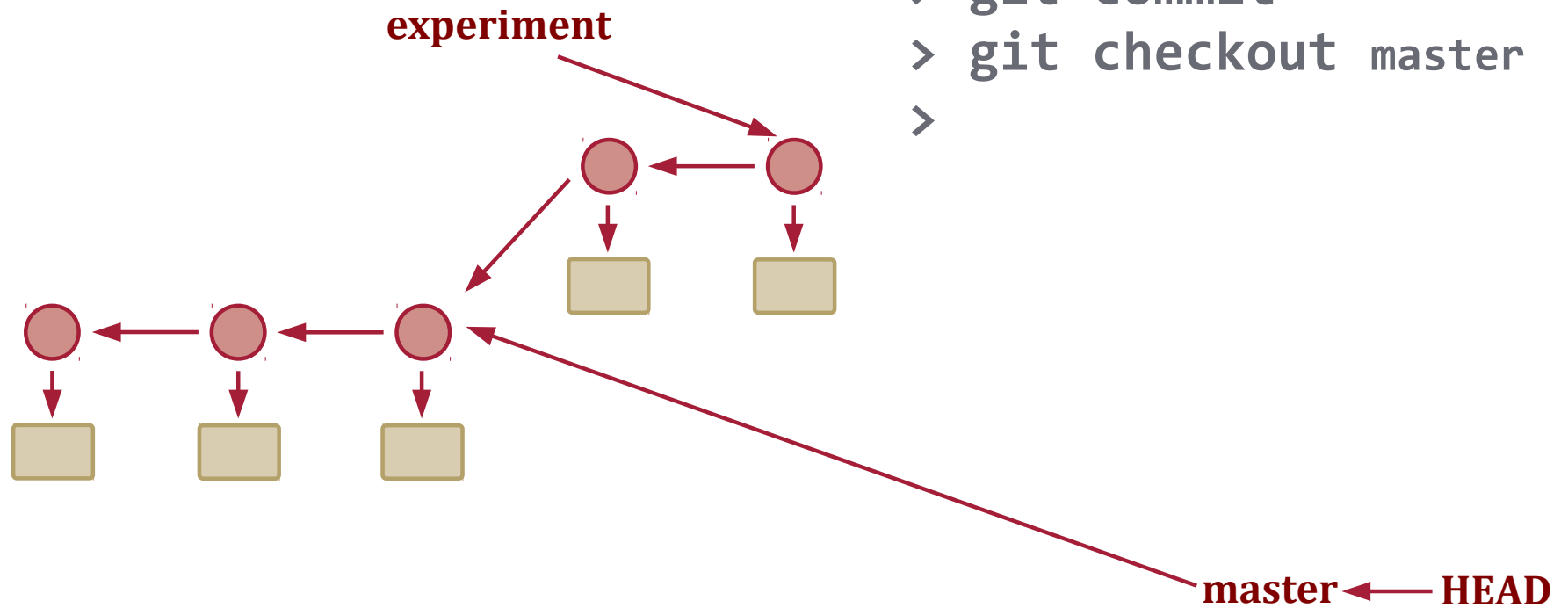






# Branching / Merging

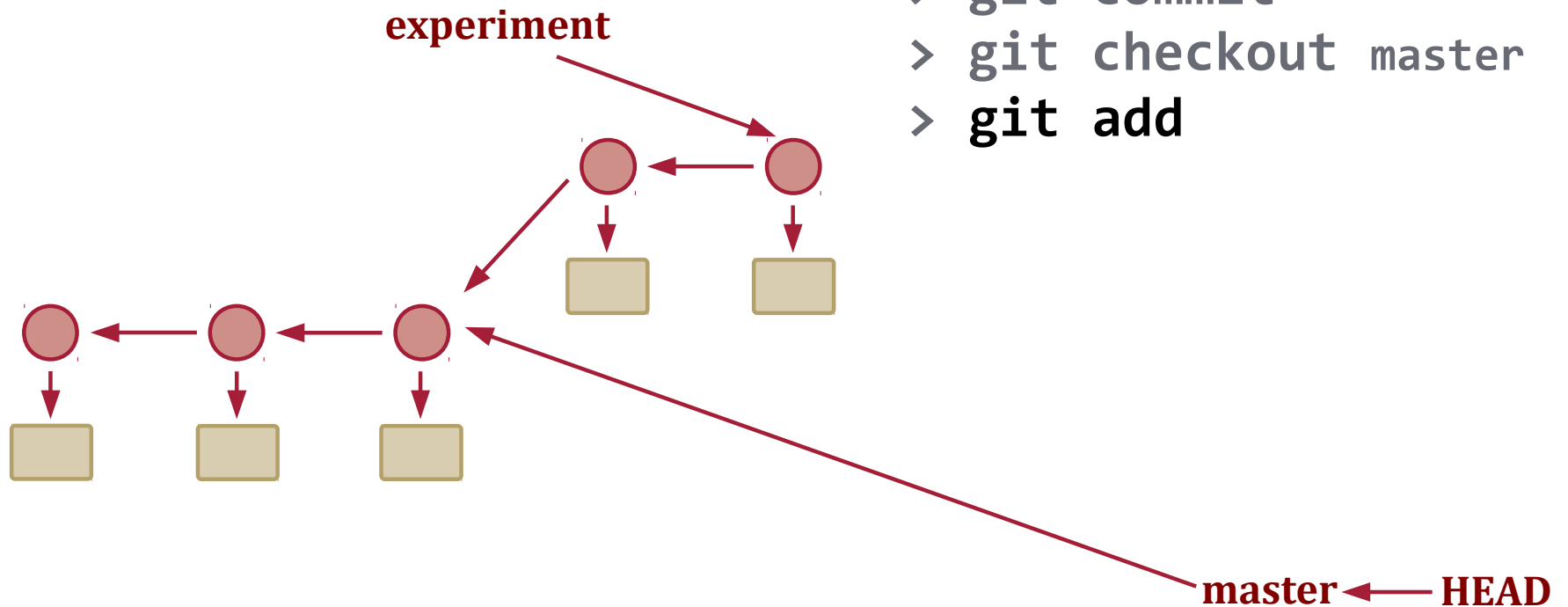
- > git add
- > git commit
- > git add
- > git commit
- > git checkout master
- >





# Branching / Merging

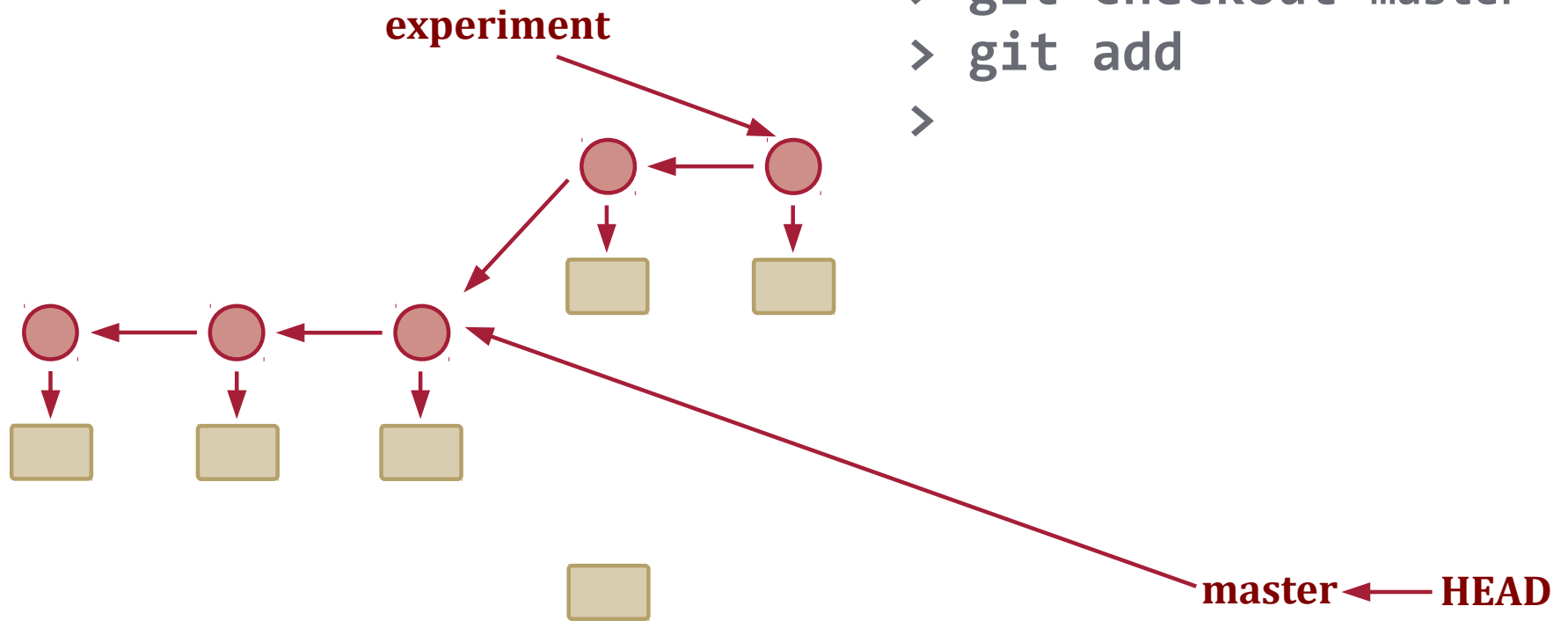
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`
- > `git checkout master`
- > `git add`





## Branching / Merging

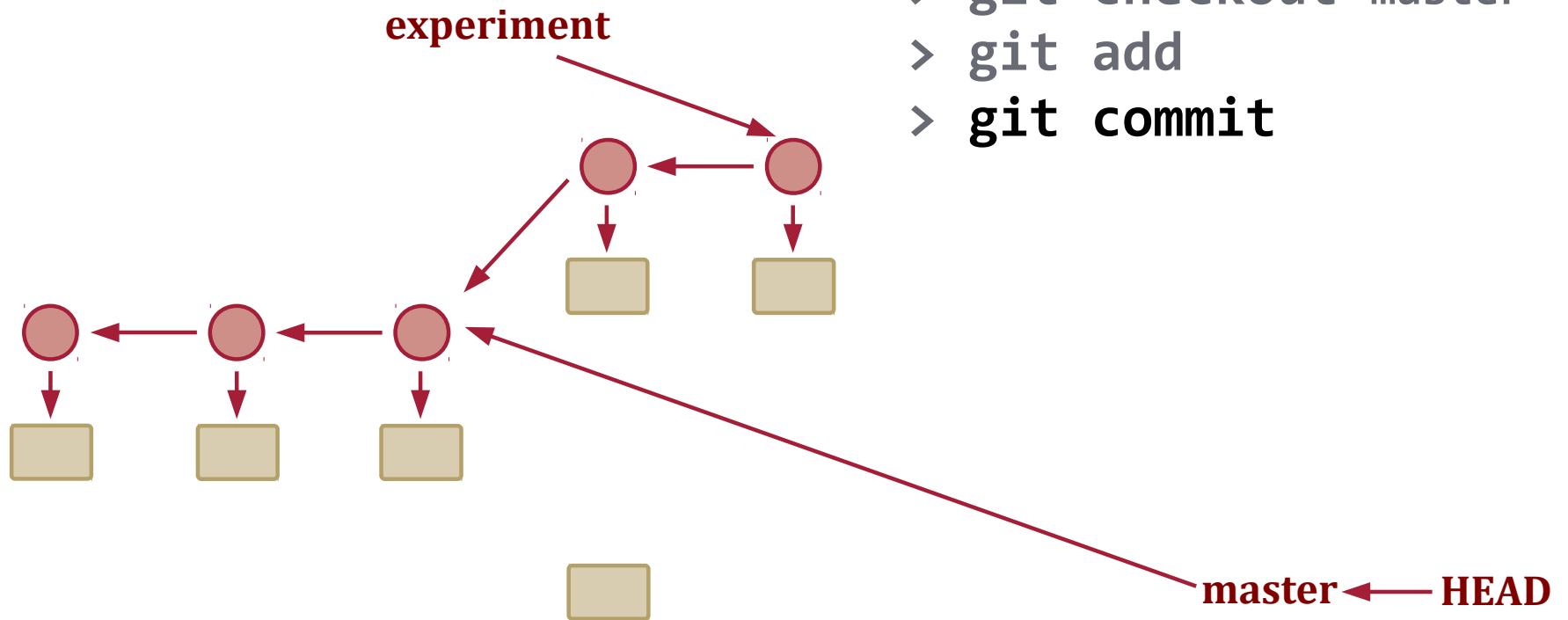
- > `git commit`
- > `git add`
- > `git commit`
- > `git checkout master`
- > `git add`
- >





## Branching / Merging

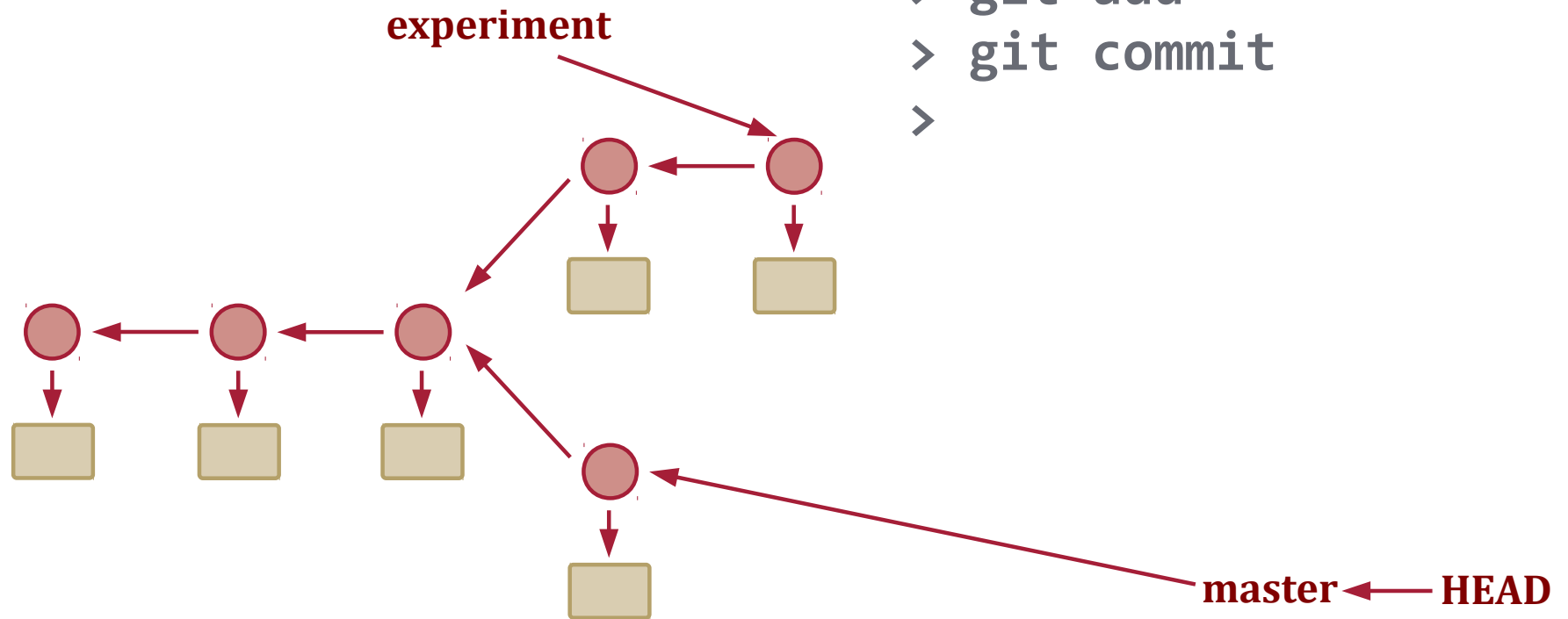
- > `git commit`
- > `git add`
- > `git commit`
- > `git checkout master`
- > `git add`
- > `git commit`





## Branching / Merging

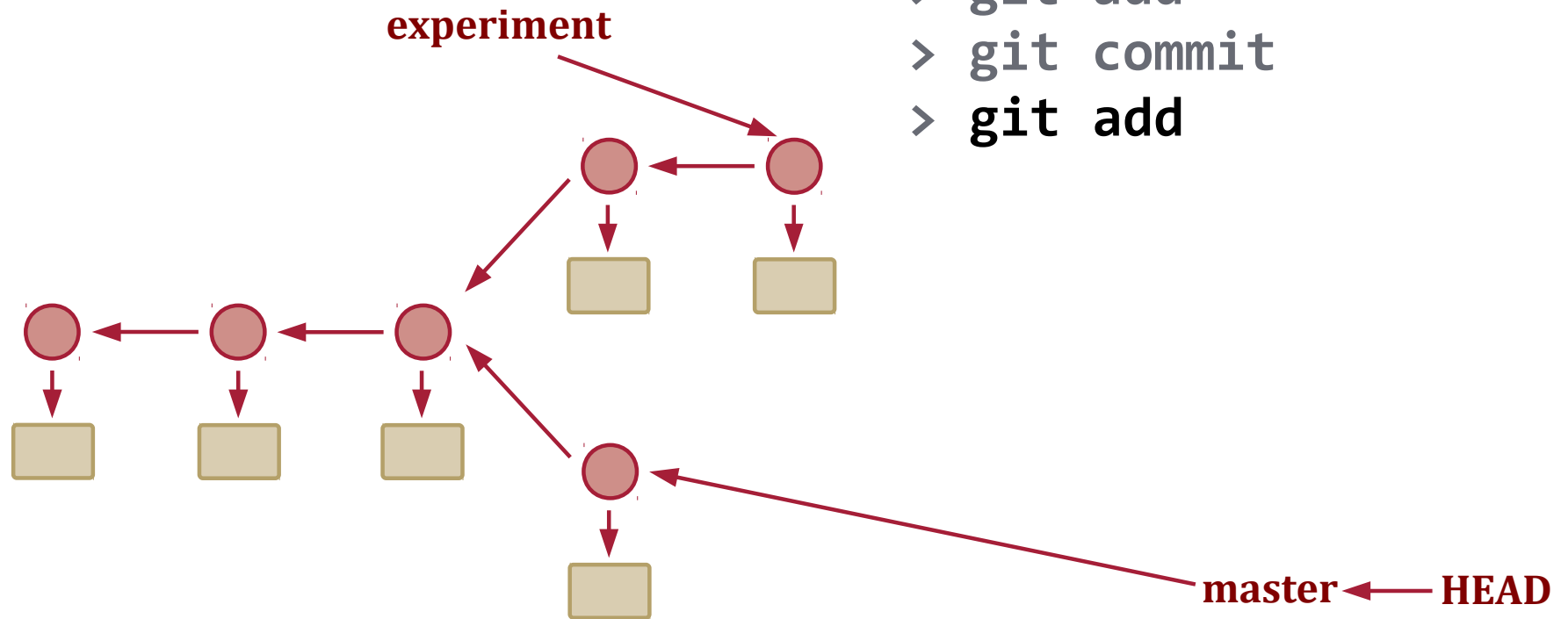
- > `git add`
- > `git commit`
- > `git checkout master`
- > `git add`
- > `git commit`
- >





## Branching / Merging

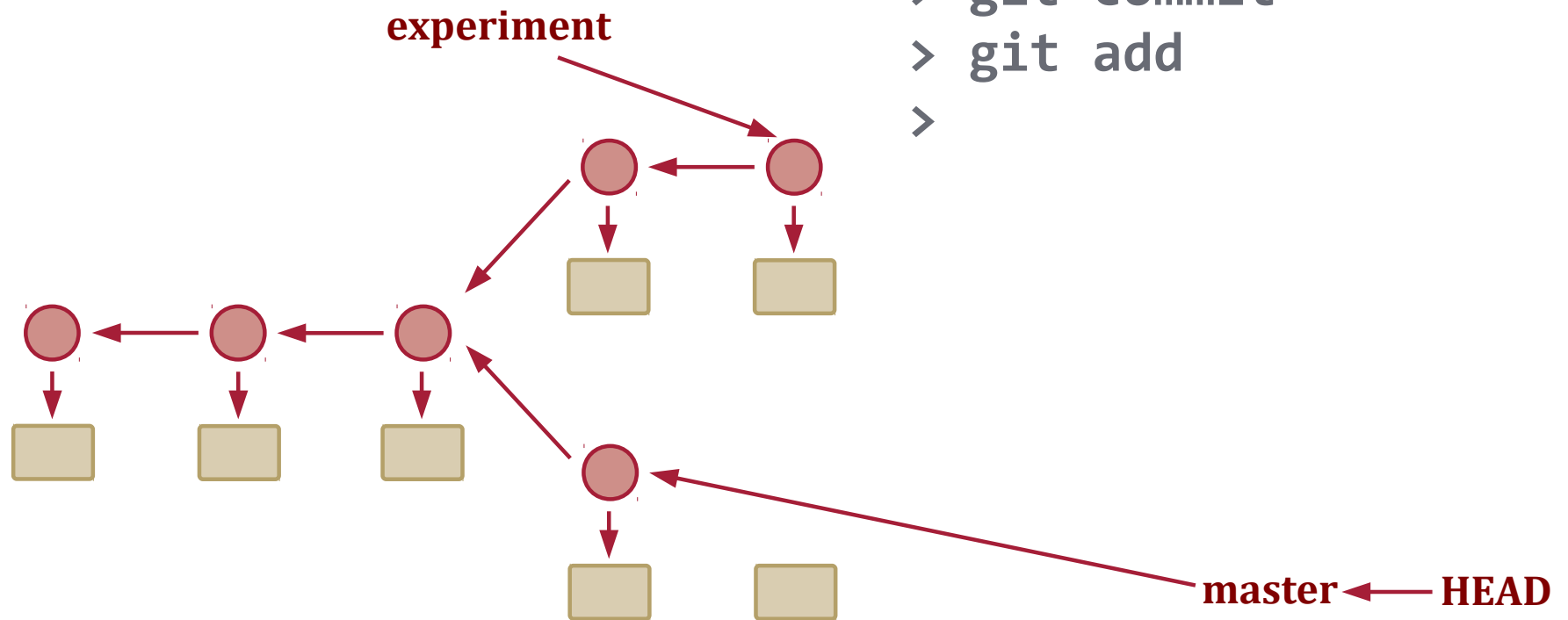
- > `git add`
- > `git commit`
- > `git checkout master`
- > `git add`
- > `git commit`
- > `git add`





## Branching / Merging

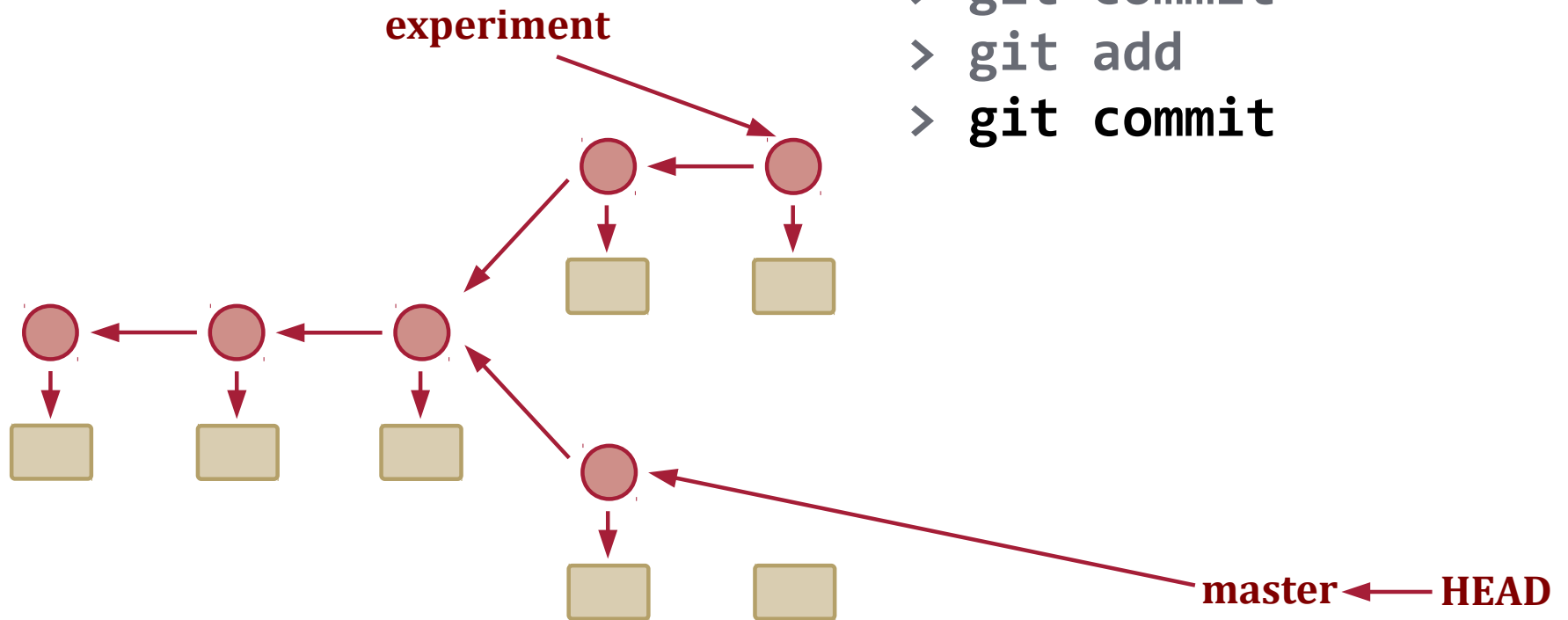
- > `git commit`
- > `git checkout master`
- > `git add`
- > `git commit`
- > `git add`
- >





## Branching / Merging

- > `git commit`
- > `git checkout master`
- > `git add`
- > `git commit`
- > `git add`
- > **`git commit`**

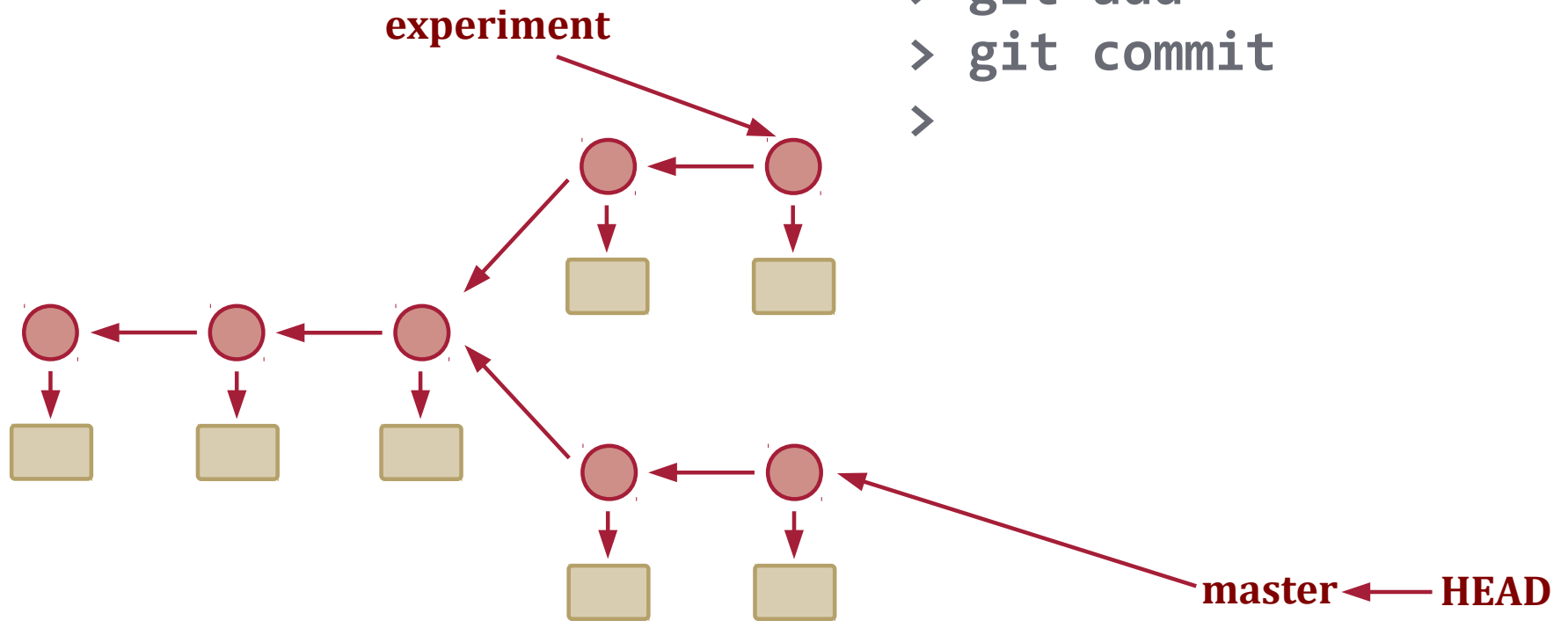






## Branching / Merging

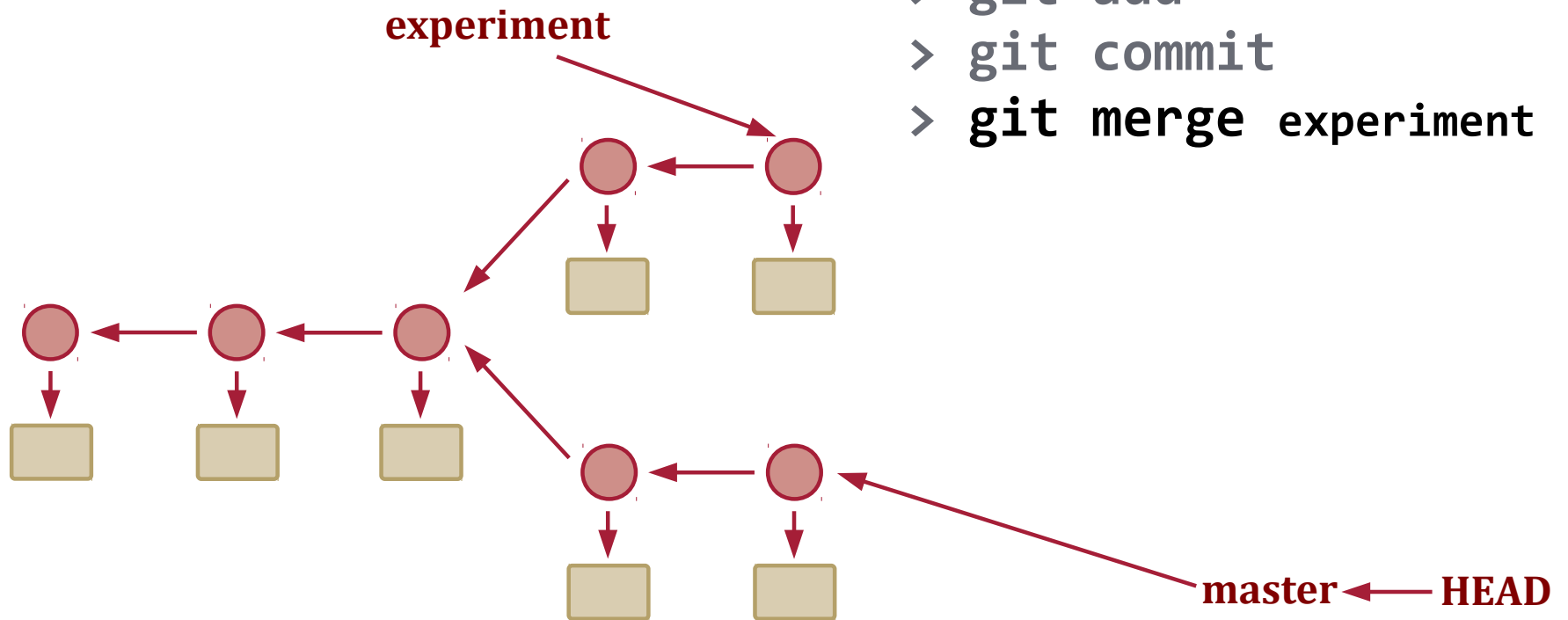
- > `git checkout master`
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`
- >





## Branching / Merging

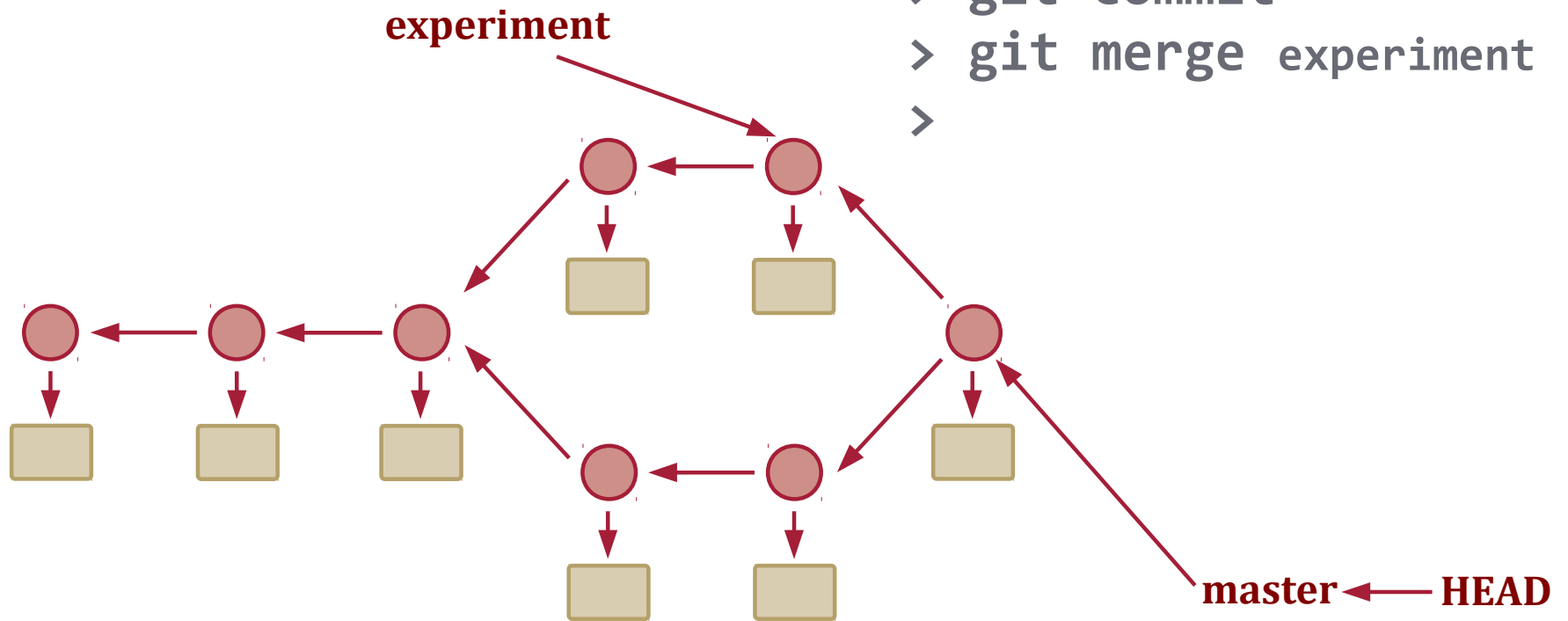
- > `git checkout master`
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`
- > `git merge experiment`





# Branching / Merging

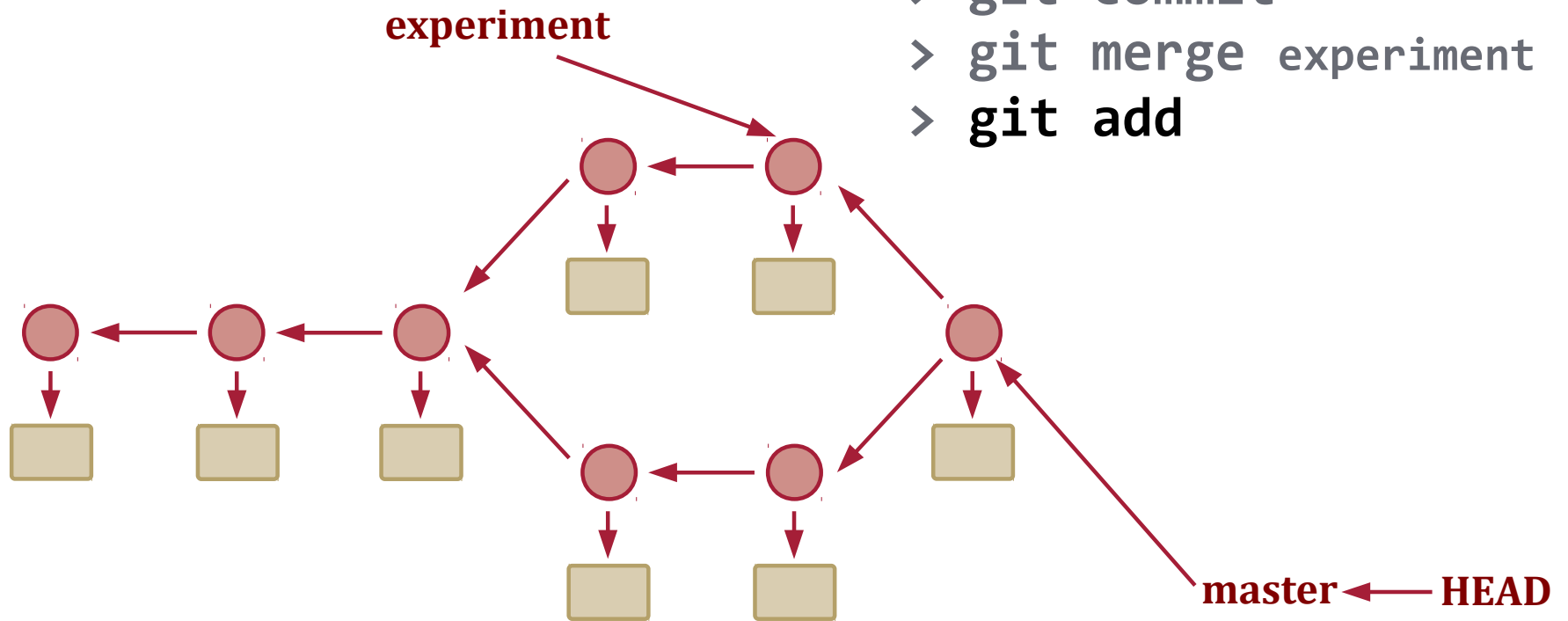
- > git add
- > git commit
- > git add
- > git commit
- > git merge experiment
- >





## Branching / Merging

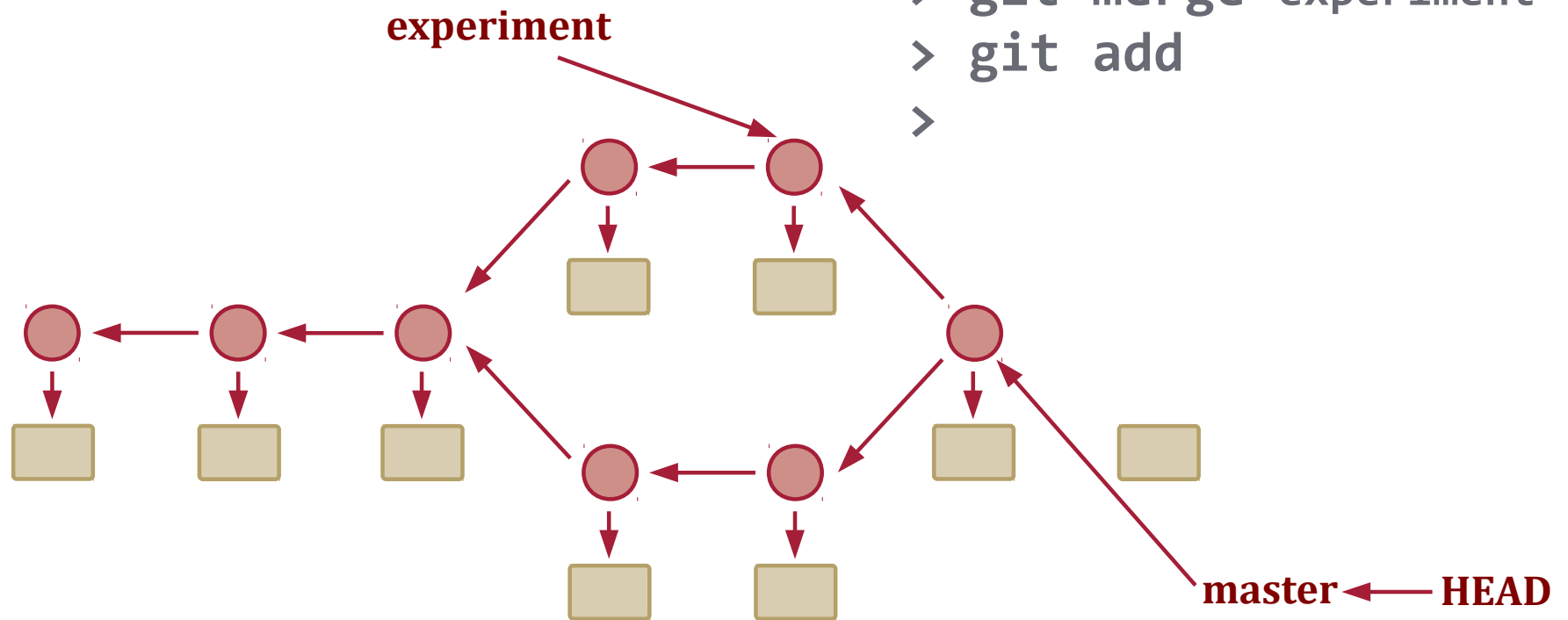
- > `git add`
- > `git commit`
- > `git add`
- > `git commit`
- > `git merge experiment`
- > `git add`





## Branching / Merging

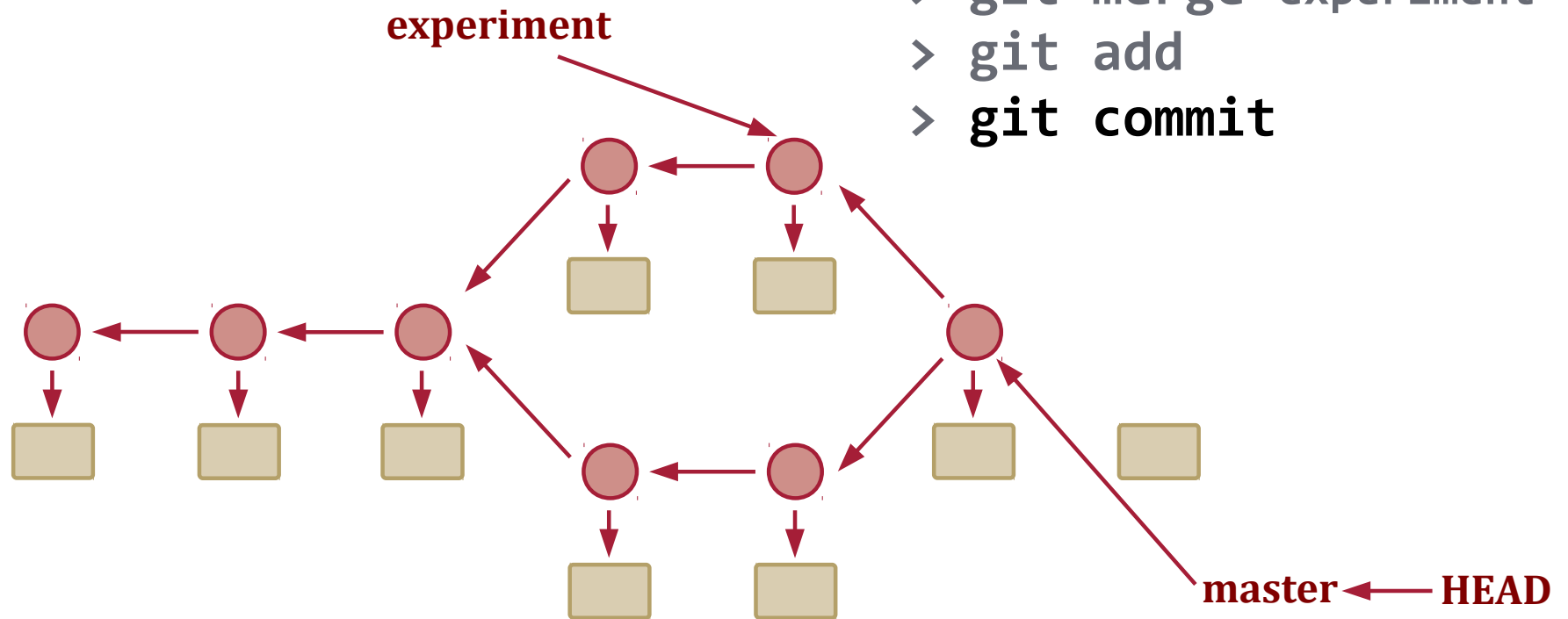
- > `git commit`
- > `git add`
- > `git commit`
- > `git merge experiment`
- > `git add`
- >





# Branching / Merging

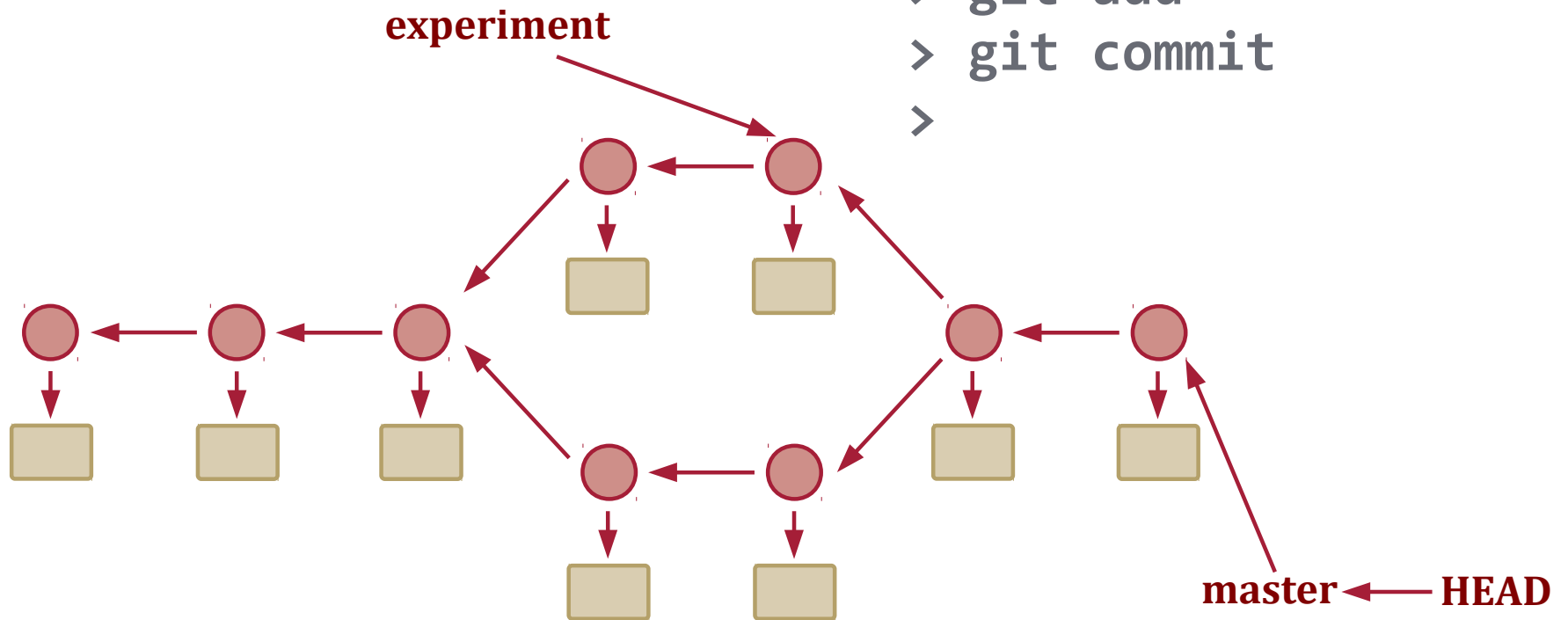
- > `git commit`
- > `git add`
- > `git commit`
- > `git merge experiment`
- > `git add`
- > `git commit`





## Branching / Merging

- > git add
- > git commit
- > git merge experiment
- > git add
- > git commit
- >





## Demo 2

*(branch, checkout commit, merge, cherrypick, blame)*





## Befehle (3/3)

- **git checkout commit**

*Commit → Index / Arbeitskopie*

- **git rm**

*Datei aus Index und  
Arbeitskopie löschen*

- **git blame**

*Letzte Änderung an Datei  
Zeile für Zeile anzeigen*

- **git merge**

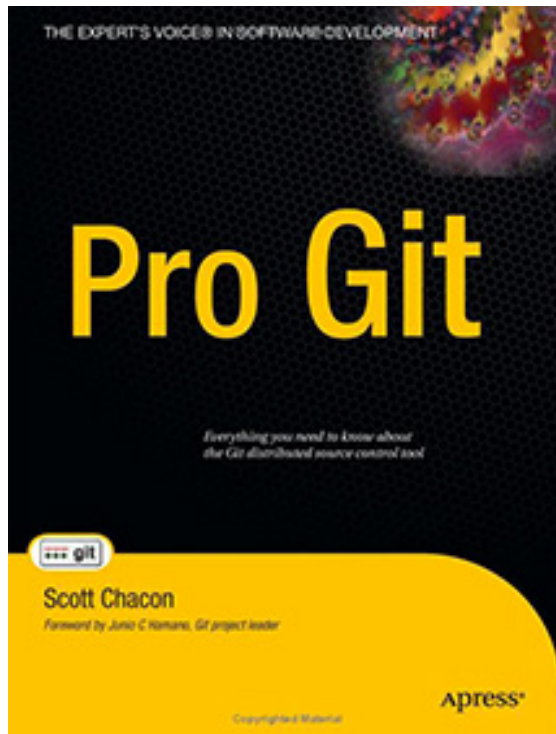
*Mehrere Branches  
zusammenführen*

- **git cherrypick**

*Einzelnen Commit an aktuellen  
Branch dranhängen*



# Weitere Informationen



<https://git-scm.com/book/>

<http://ps.informatik.uni-tuebingen.de/industryproject/>



# Acknowledgments

- Icons  and  from [freepik.com](https://www.freepik.com) / [flaticon.com](https://www.flaticon.com)