

Werkzeuge für das wissenschaftliche Arbeiten

Python for Machine Learning and Data Science

Magnus Bender
bender@ifis.uni-luebeck.de
Wintersemester 2022/23

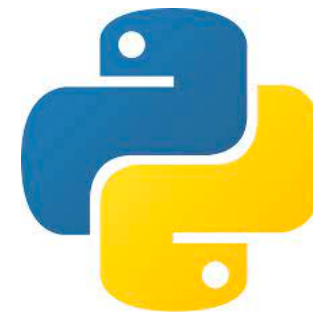
Inhaltsübersicht

1. Programmiersprache Python

a) *Einführung, Erste Schritte*

b) *Grundlagen*

c) *Fortgeschritten*



2. Auszeichnungssprachen

a) *LaTeX, Markdown*

L^AT_EX



3. Benutzeroberflächen und Entwicklungsumgebungen

a) *Jupyter Notebooks lokal und in der Cloud (Google Colab)*

4. Versionsverwaltung

a) **Git, GitHub**



5. Wissenschaftliches Rechnen

a) NumPy, SciPy



6. Datenverarbeitung und -visualisierung

a) Pandas, matplotlib, NLTK

7. Machine Learning (scikit-learn)

a) Grundlegende Ansätze (Datensätze, Auswertung)

b) Einfache Verfahren (Clustering, ...)



8. DeepLearning

a) TensorFlow, PyTorch, HuggingFace Transformers



Themen

I. Versionsverwaltung

II. Git

1. Idee, Konfiguration

2. Lokal: Commit, Stash, Branch, Merge

3. Remote: Push, Pull, Merge

III. GitHub



Heute

I. Versionsverwaltung

Versionen und Verlauf

- Verschiedene Versionen z.B. eines Programms
- Verschiedene Features/ Probleme werden (gleichzeitig) bearbeitet



data.py



plot.py

Versionen und Verlauf

- Verschiedene Versionen z.B. eines Programms
- Verschiedene Features/ Probleme werden (gleichzeitig) bearbeitet



data.py



data_fixed.py



plot.py

Versionen und Verlauf

- Verschiedene Versionen z.B. eines Programms
- Verschiedene Features/
Probleme werden
(gleichzeitig) bearbeitet



data.py



data_fixed.py



20221123_data.py



plot.py

Versionen und Verlauf

- Verschiedene Versionen z.B. eines Programms
- Verschiedene Features/ Probleme werden (gleichzeitig) bearbeitet



data.py



data_fixed.py



20221123_data.py



plot.py



plot_v2.py



20221001_plot.py

Versionen und Verlauf

- Verschiedene Versionen z.B. eines Programms
- Verschiedene Features/ Probleme werden (gleichzeitig) bearbeitet
- Verlauf soll gespeichert werden



data.py



data_fixed.py



20221123_data.py



plot.py



plot_v2.py



20221001_plot.py

Versionen und Verlauf

- Verschiedene Versionen z.B. eines Programms
- Verschiedene Features/ Probleme werden (gleichzeitig) bearbeitet
- Verlauf soll gespeichert werden



data.py



data_fixed.py



20221123_data.py



plot.py

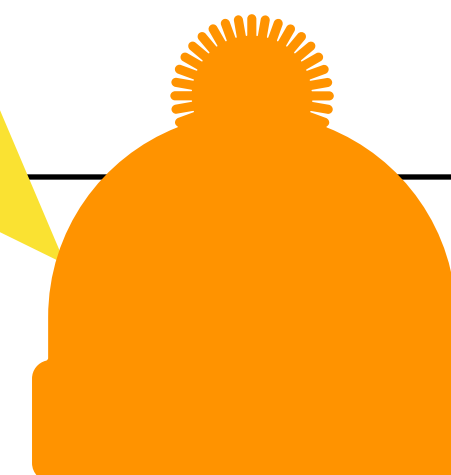


plot_v2.py

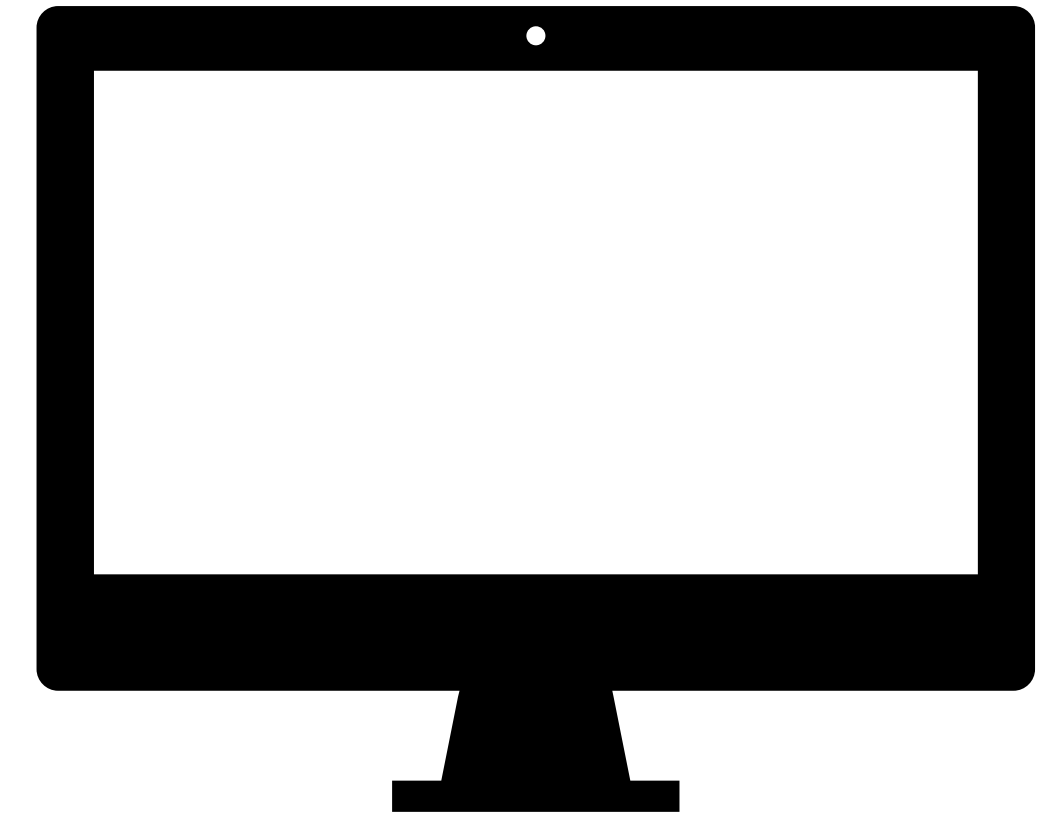


20221001_plot.py

import nutzt den Dateinamen :-(



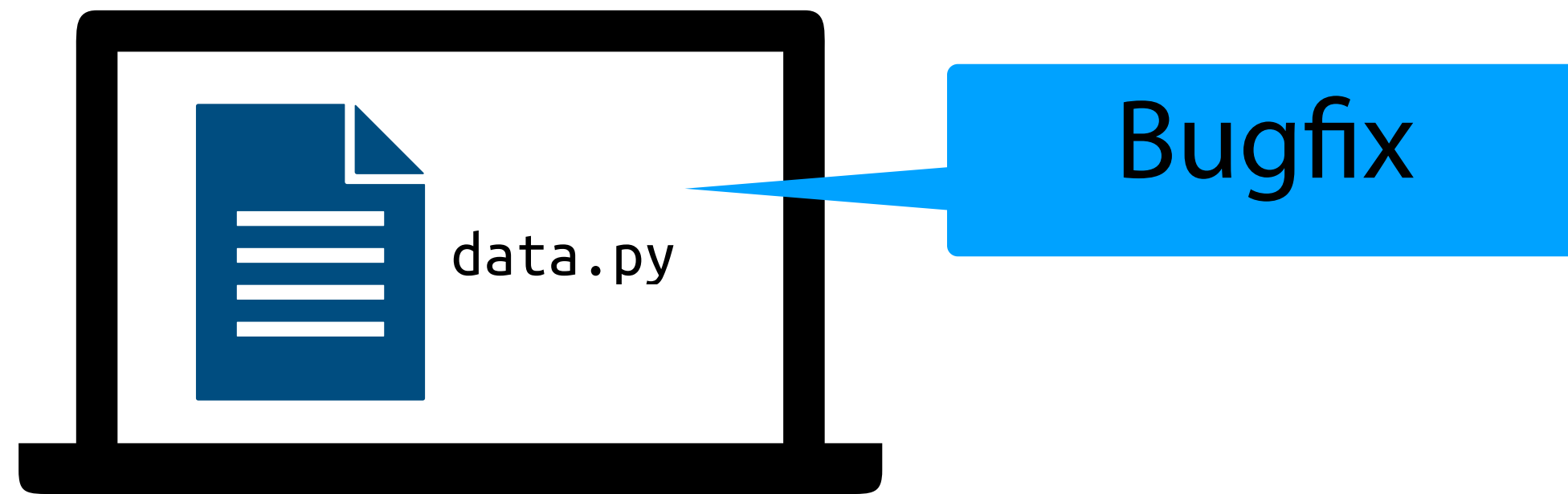
Verschiedene Entwicklungsorte



Verschiedene Entwicklungsorte

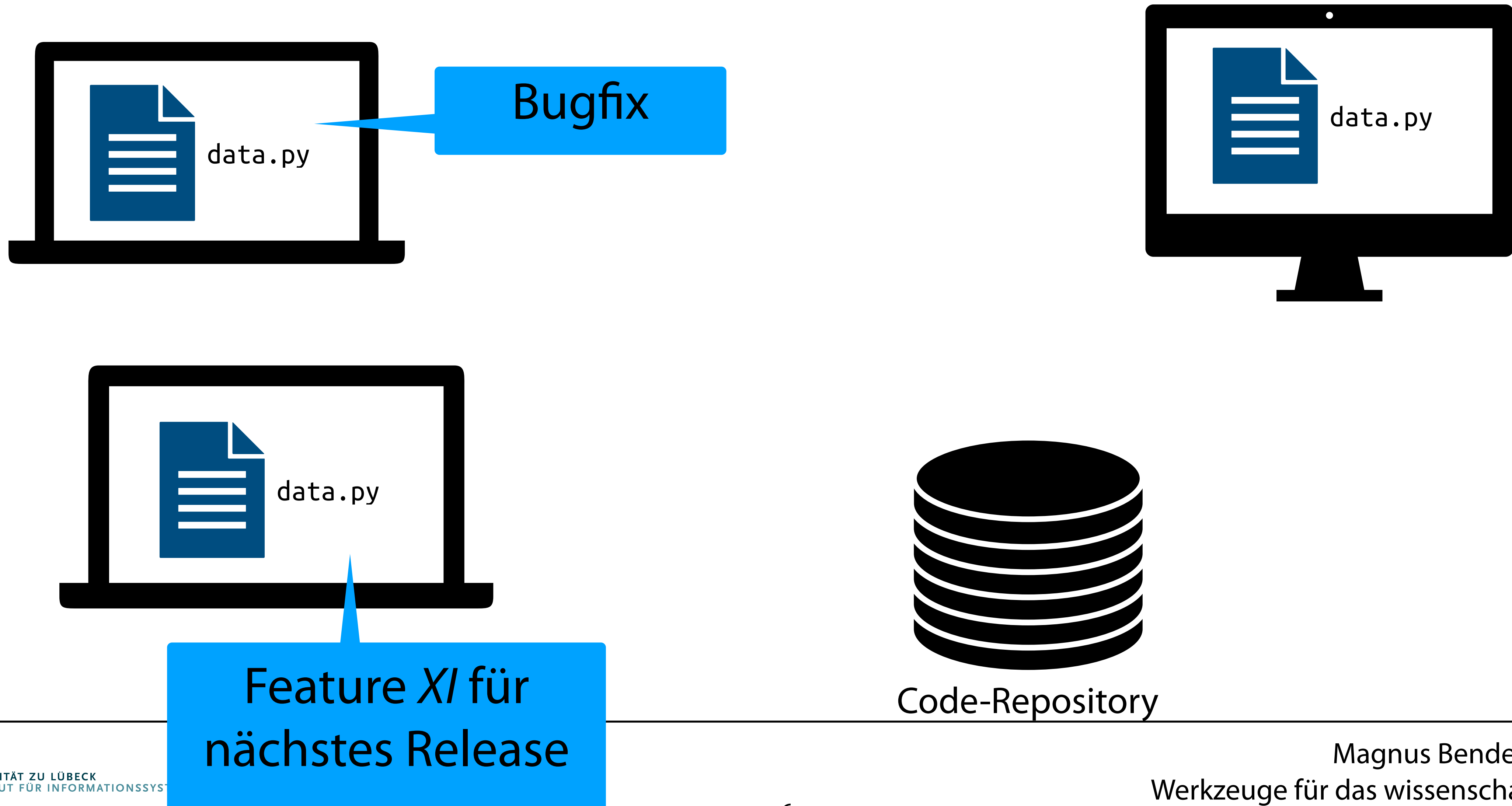


Verschiedene Entwicklungsorte

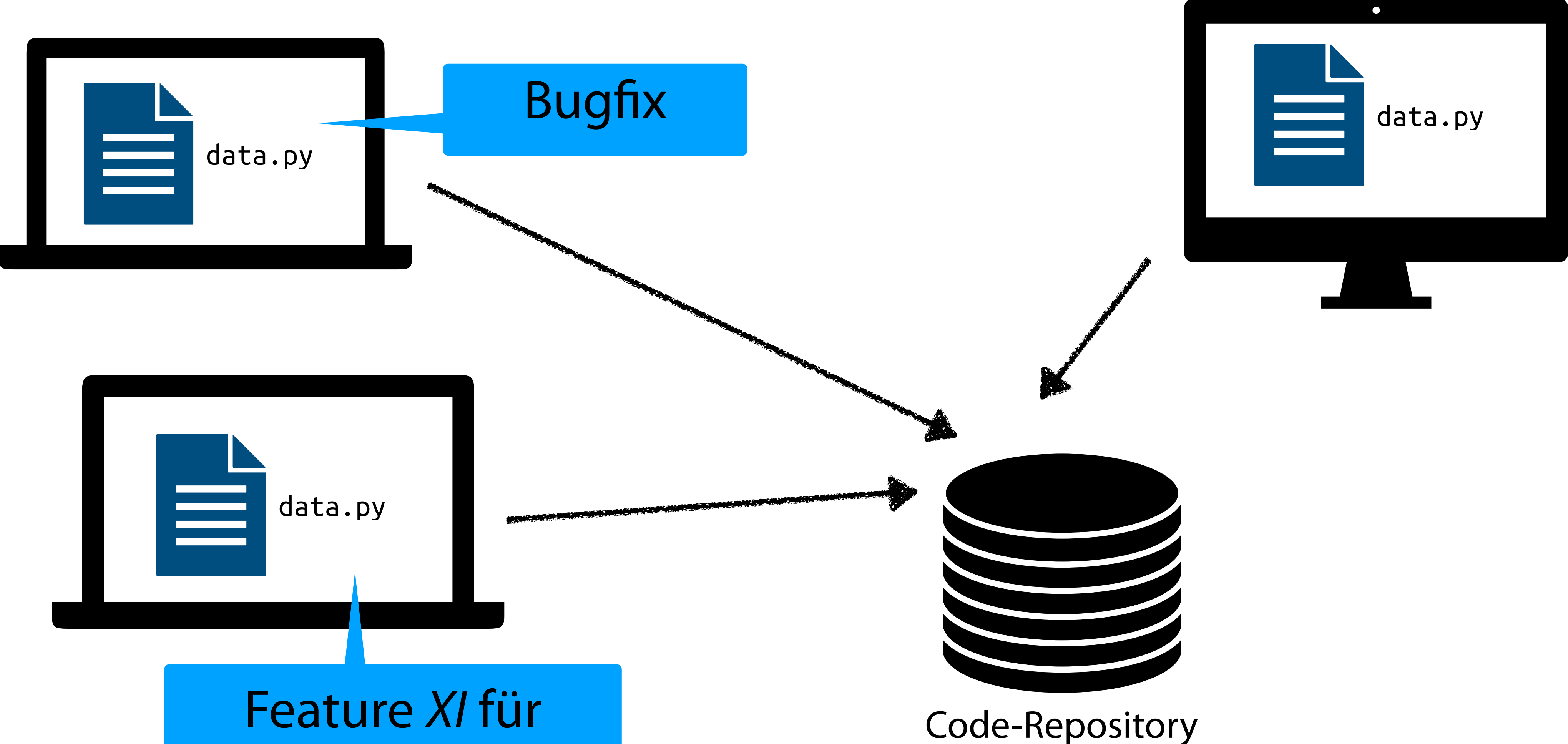


Feature *XI* für
nächstes Release

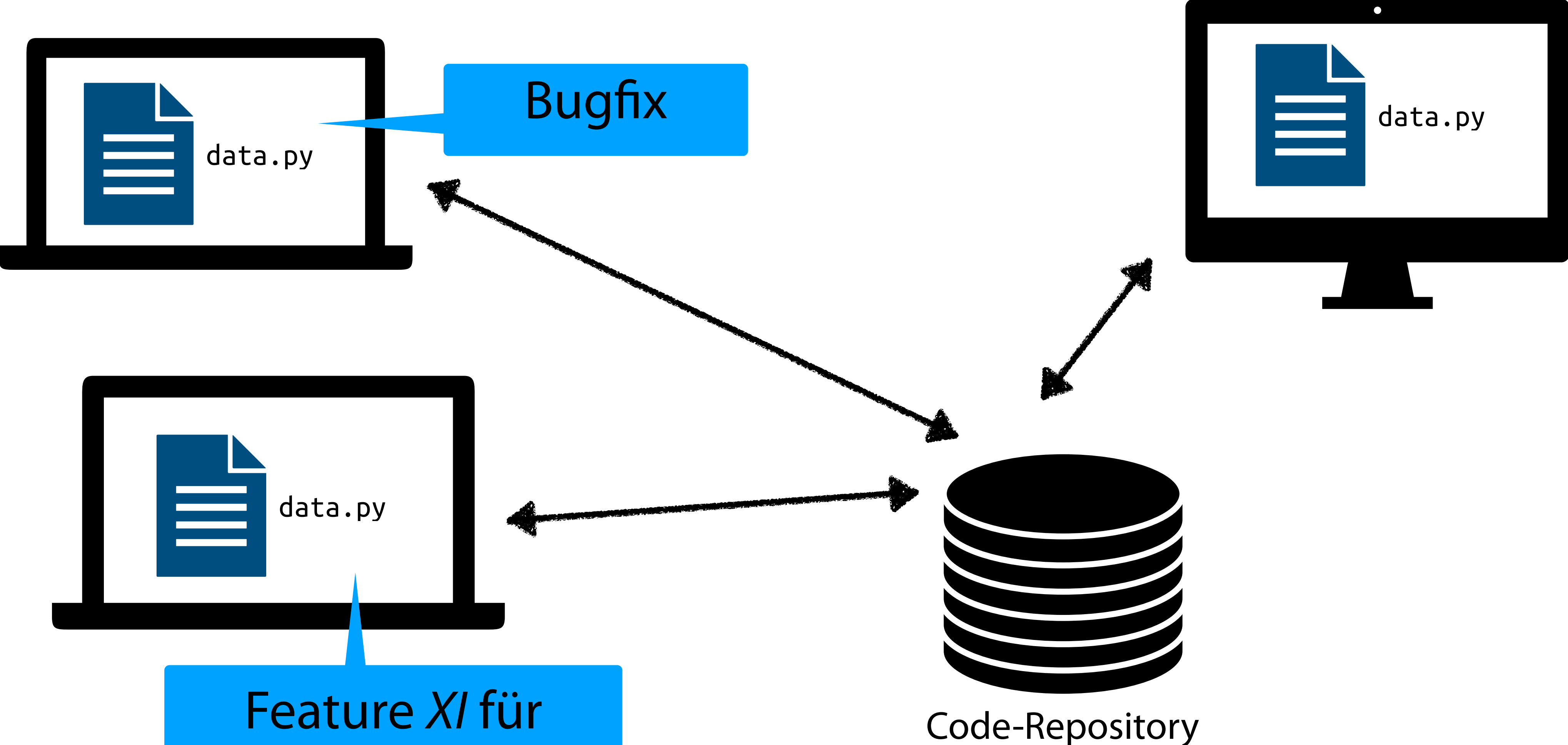
Verschiedene Entwicklungsorte



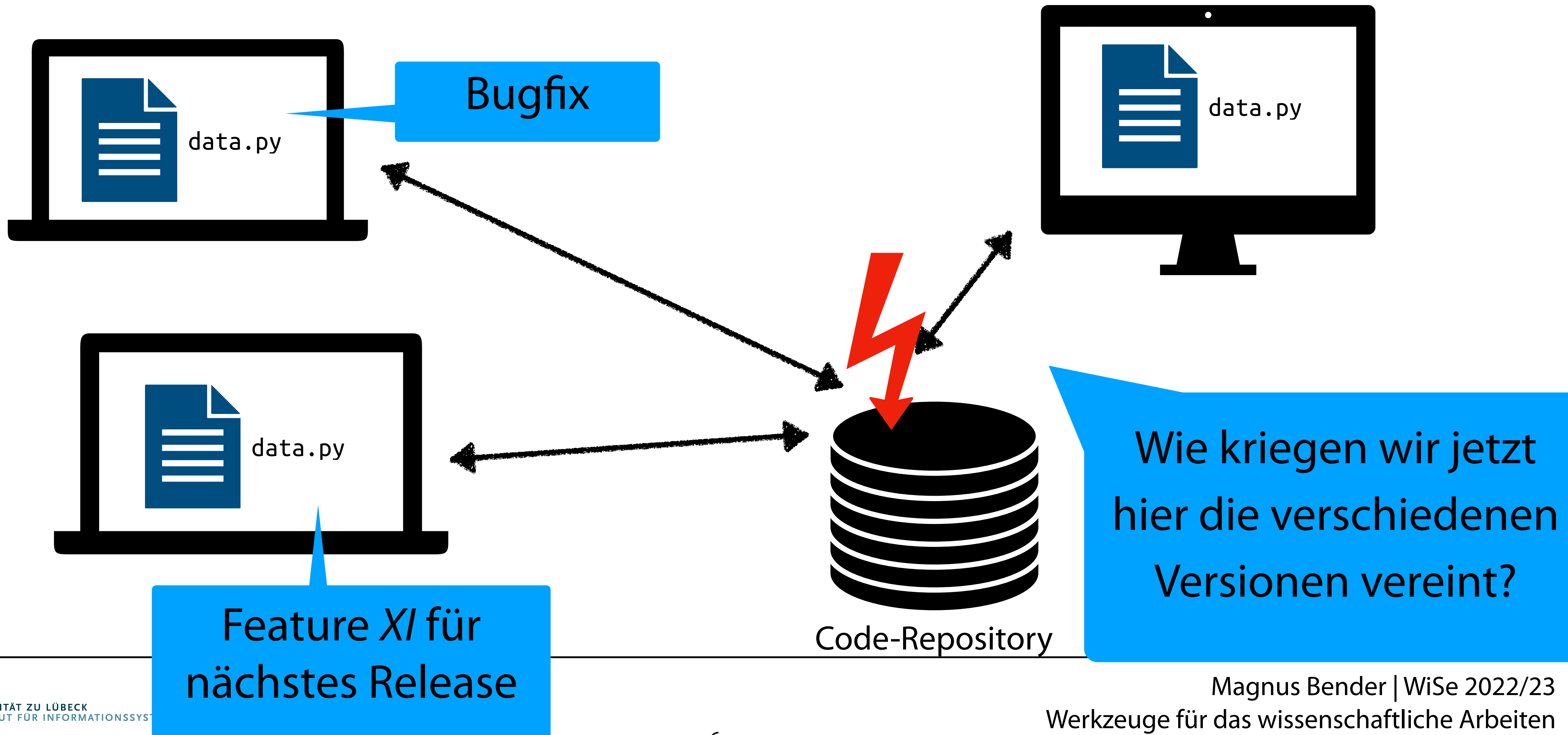
Verschiedene Entwicklungsorte



Verschiedene Entwicklungsorte



Verschiedene Entwicklungsorte



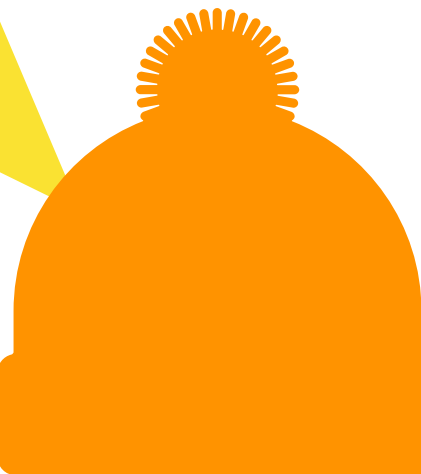
Lösung: Versionsverwaltung

- Verlauf der Änderungen (textbasierte Dateien)

Lösung: Versionsverwaltung

- Verlauf der Änderungen (textbasierte Dateien)

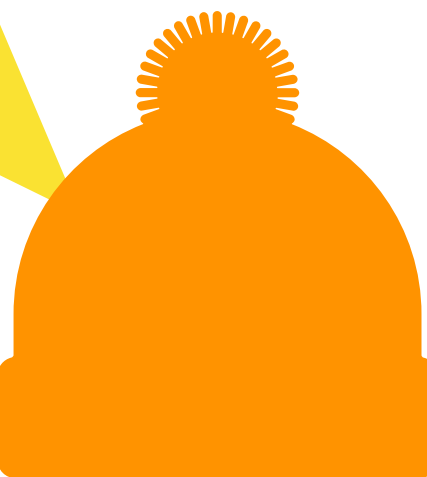
Insbesondere auch
Zurücksetzen der
Änderungen möglich!



Lösung: Versionsverwaltung

- Verlauf der Änderungen (textbasierte Dateien)
- Verschiedene Entwicklungszweige gleichzeitig
- Verschiedene (neue) Features und Fehlerbehebungen
- Verschiedene Orte

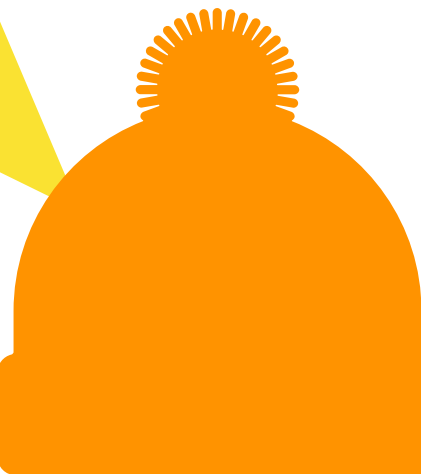
Insbesondere auch
Zurücksetzen der
Änderungen möglich!



Lösung: Versionsverwaltung

- Verlauf der Änderungen (textbasierte Dateien)
- Verschiedene Entwicklungszweige gleichzeitig
 - Verschiedene (neue) Features und Fehlerbehebungen
 - Verschiedene Orte
- Zusammenführen von Entwicklungszweigen

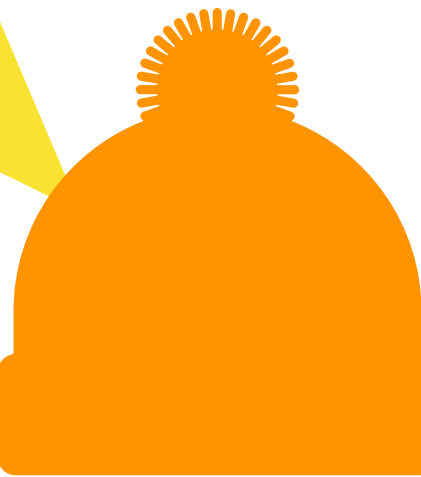
Insbesondere auch
Zurücksetzen der
Änderungen möglich!



Lösung: Versionsverwaltung

- Verlauf der Änderungen (textbasierte Dateien)
- Verschiedene Entwicklungszweige gleichzeitig
 - Verschiedene (neue) Features und Fehlerbehebungen
 - Verschiedene Orte
- Zusammenführen von Entwicklungszweigen
- Ein (zentrales) Repository

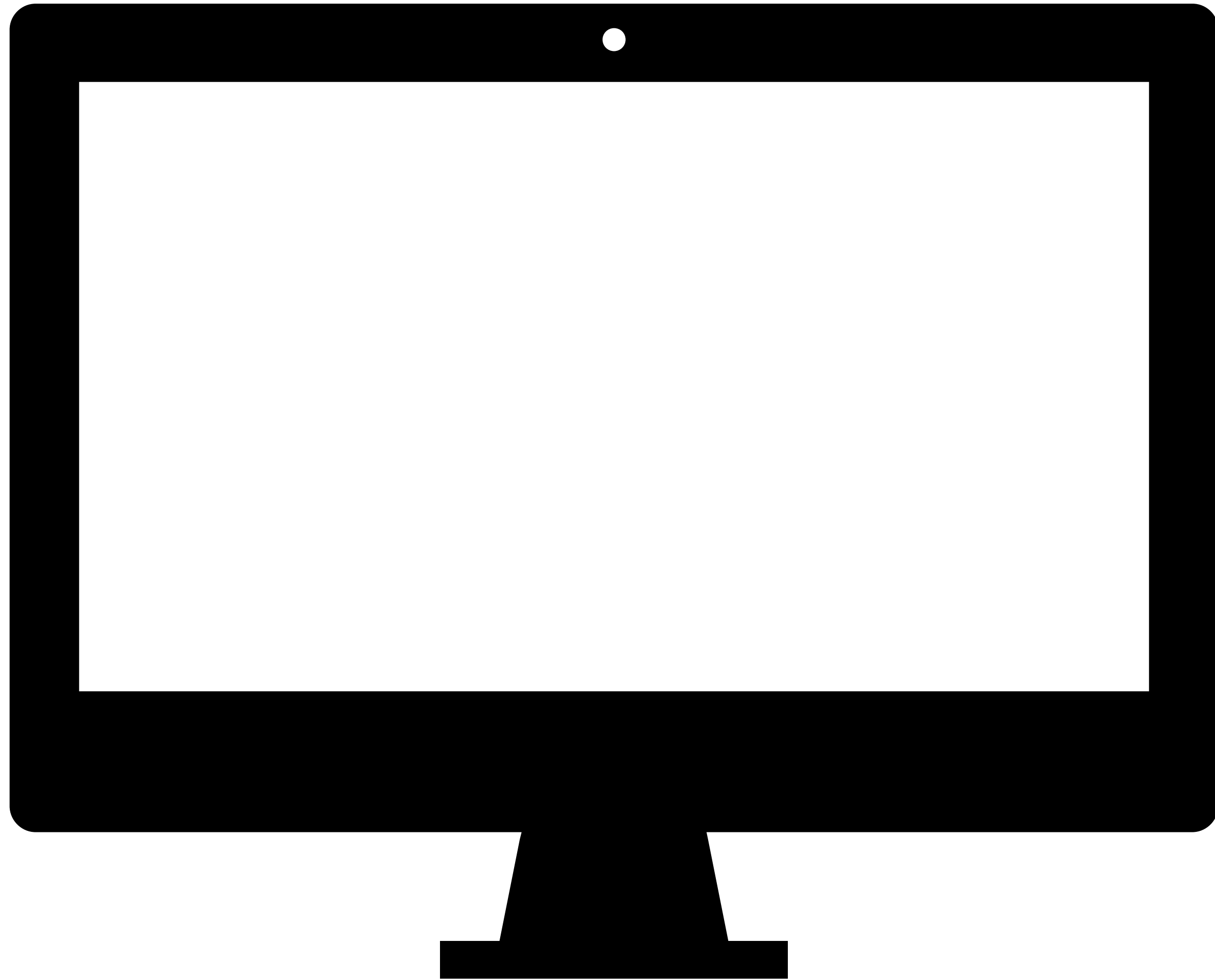
Insbesondere auch
Zurücksetzen der
Änderungen möglich!



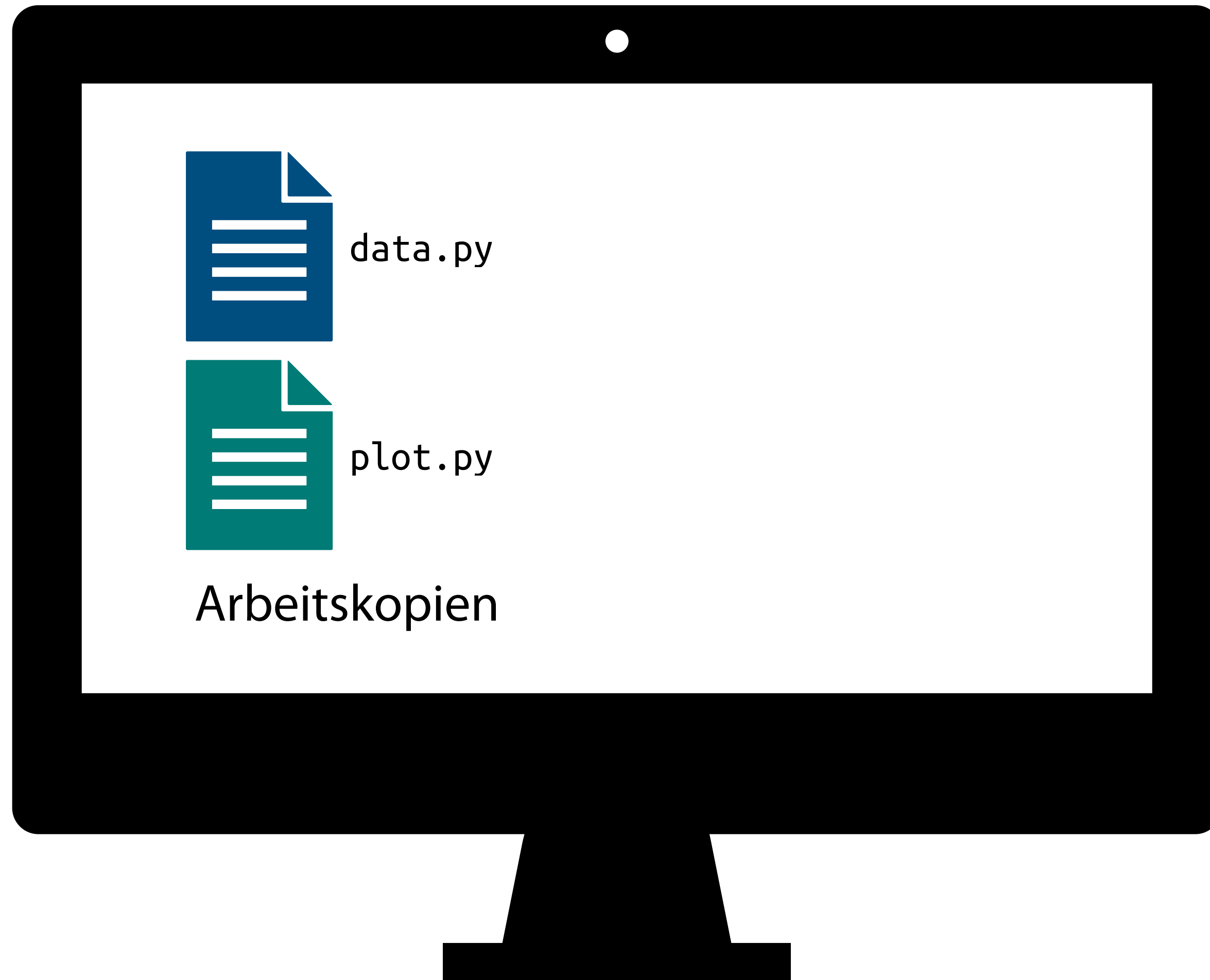
II. Git

1. Idee, Konfiguration

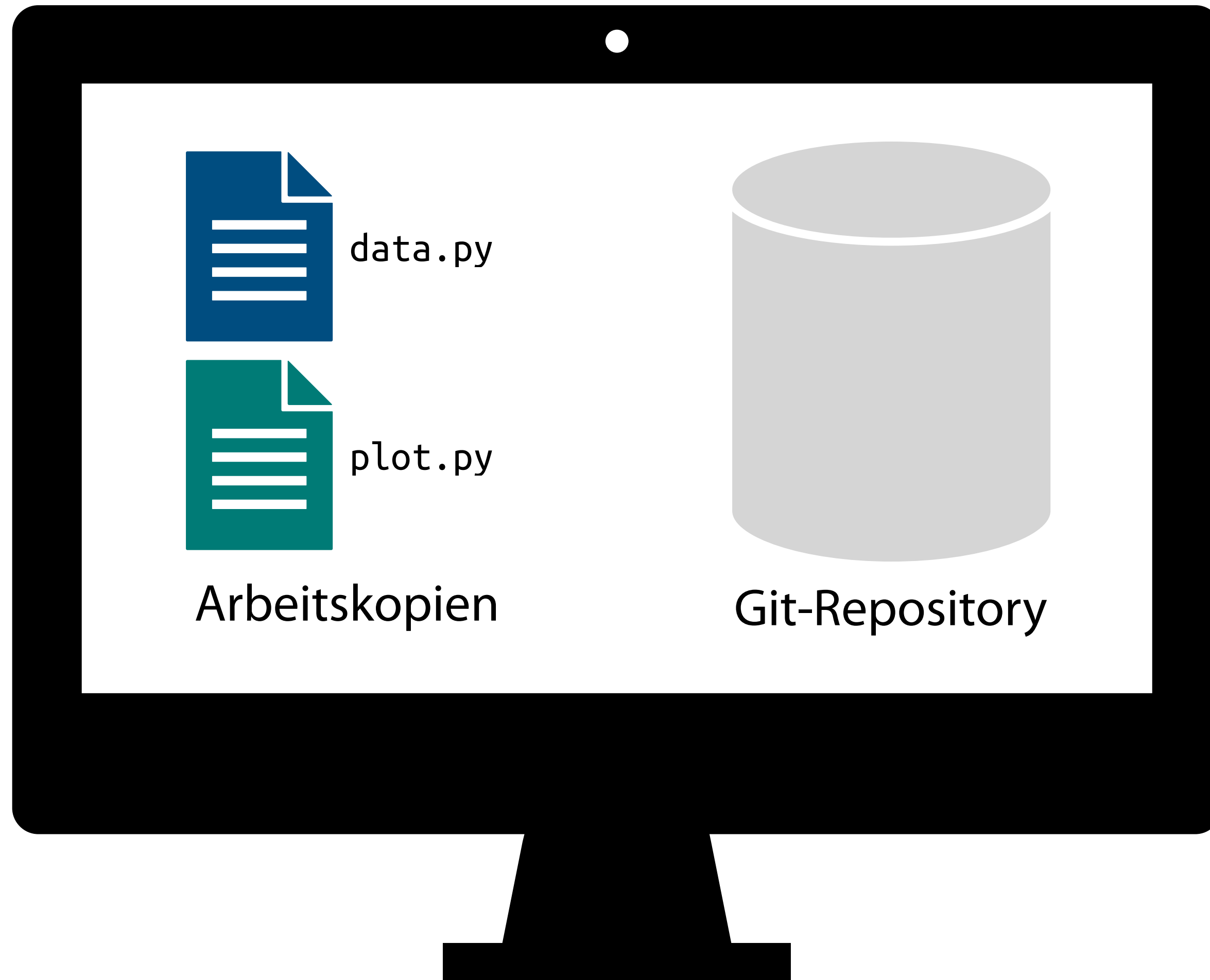
Verteilte Repositories



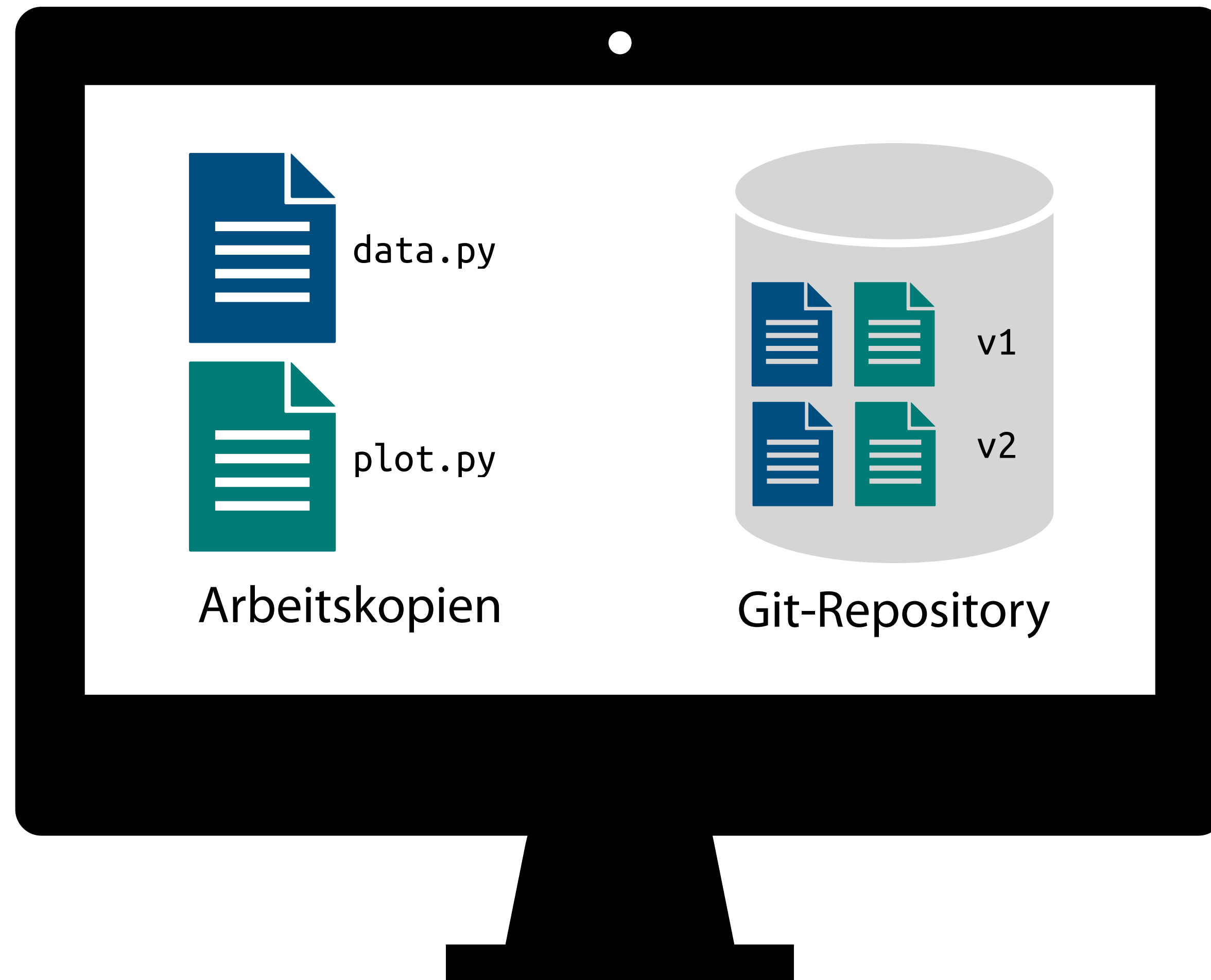
Verteilte Repositories



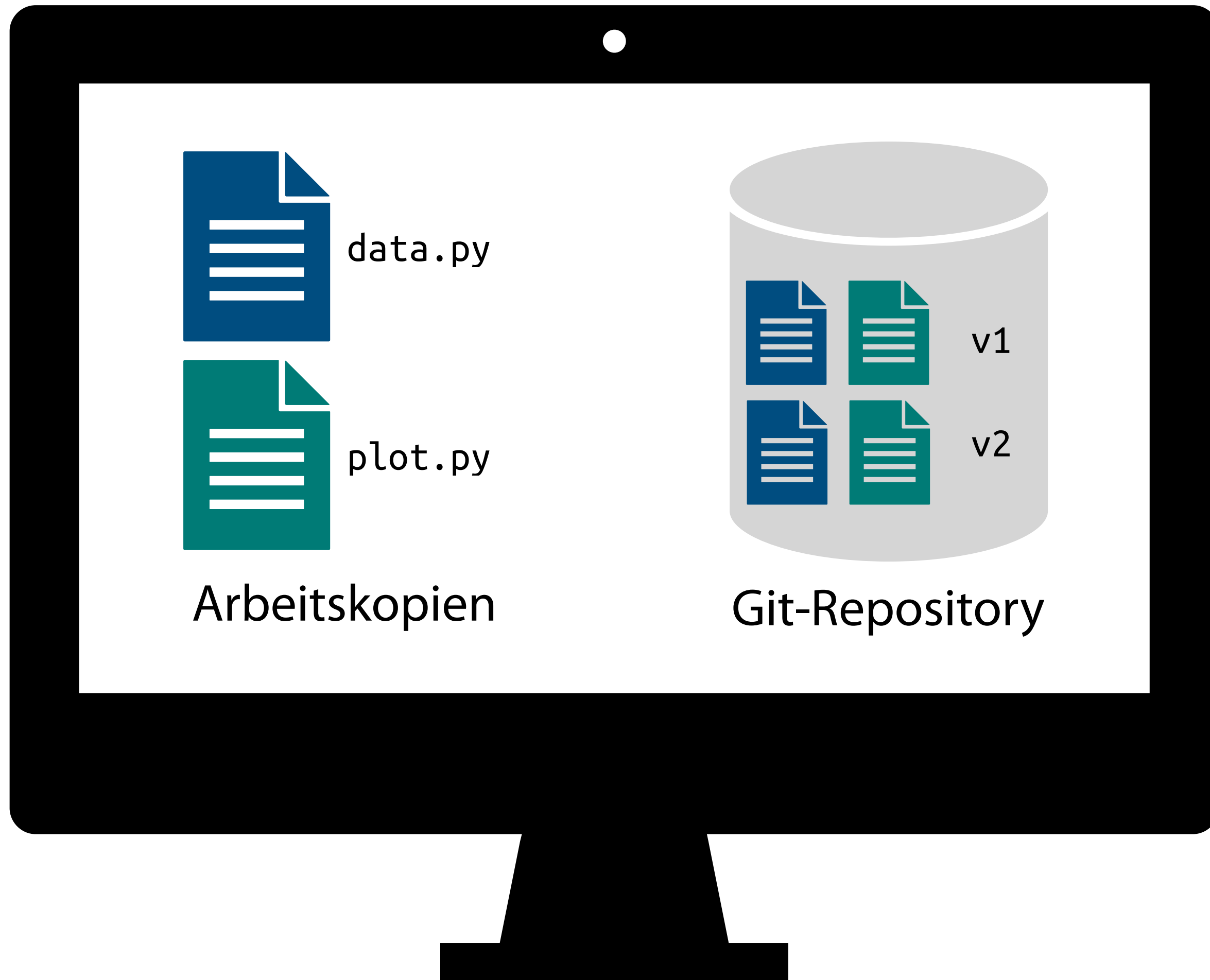
Verteilte Repositories



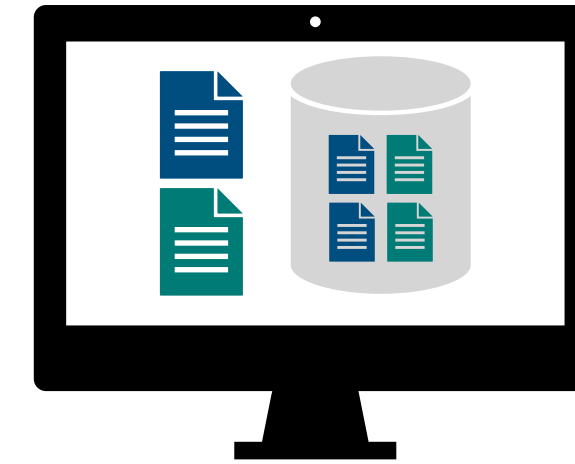
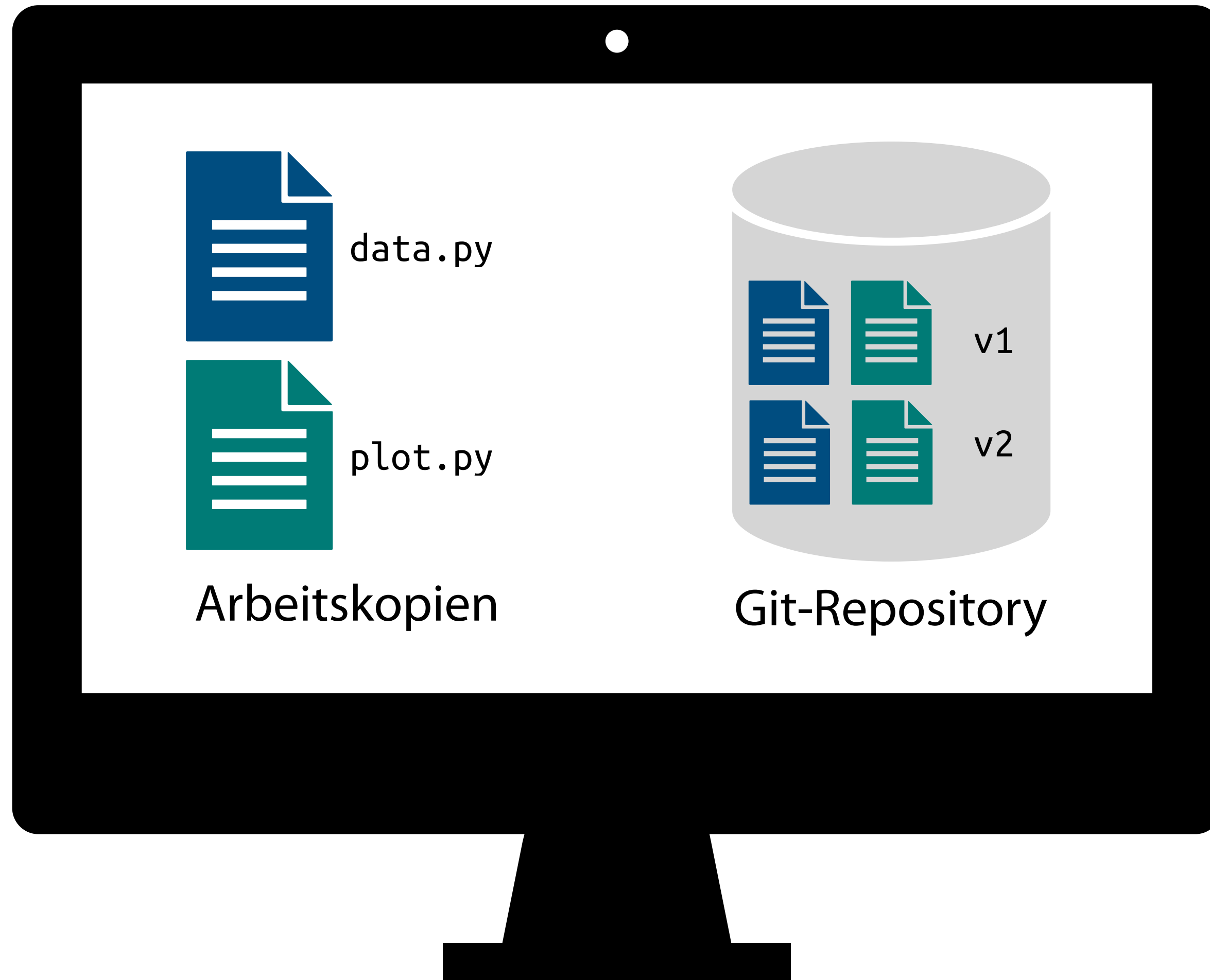
Verteilte Repositories



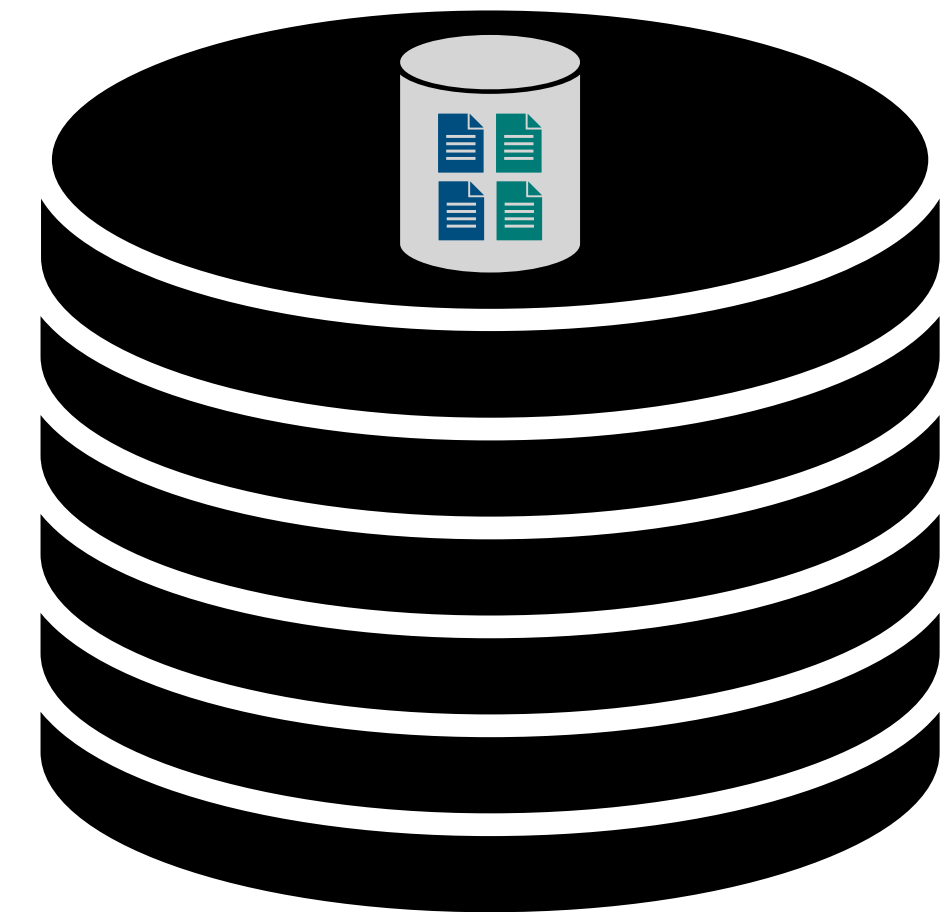
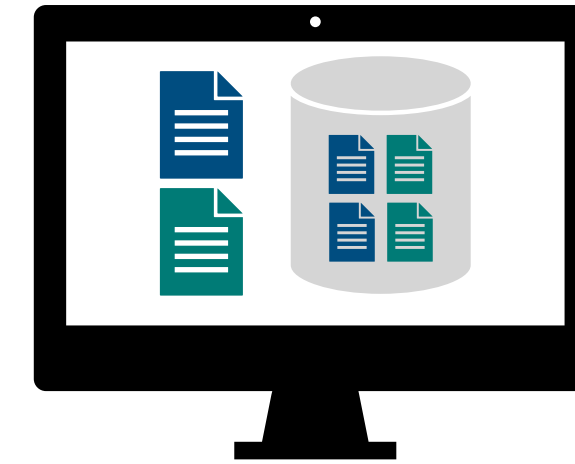
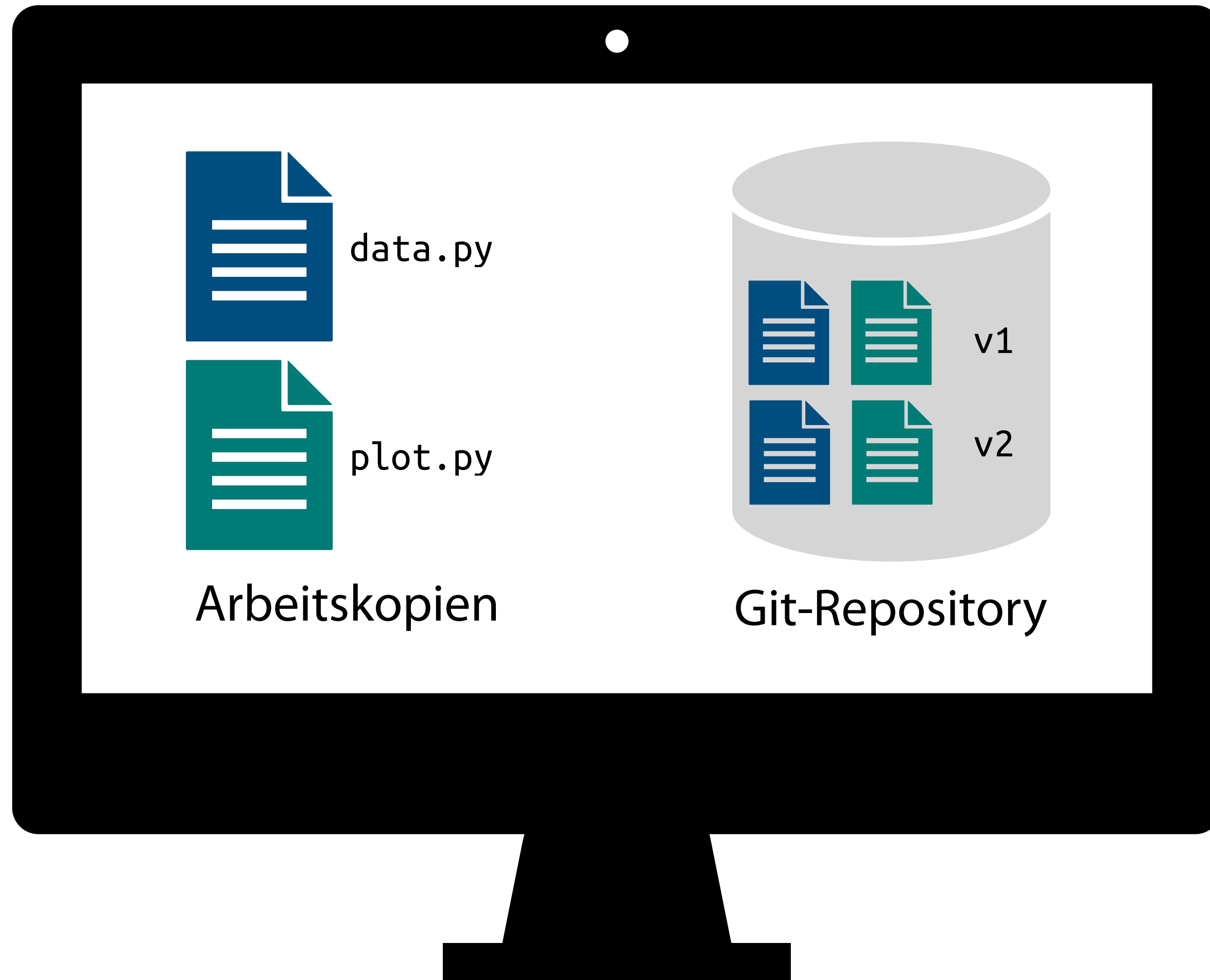
Verteilte Repositories



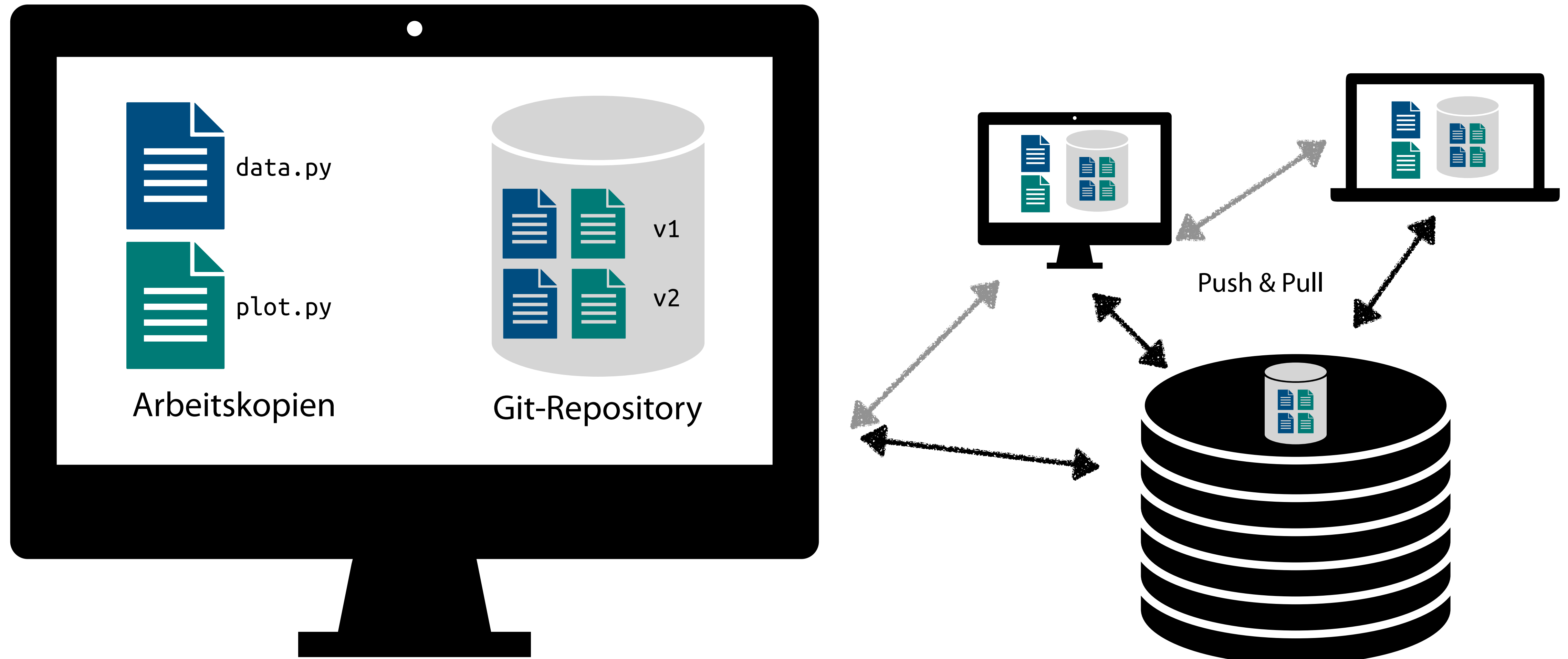
Verteilte Repositories



Verteilte Repositories



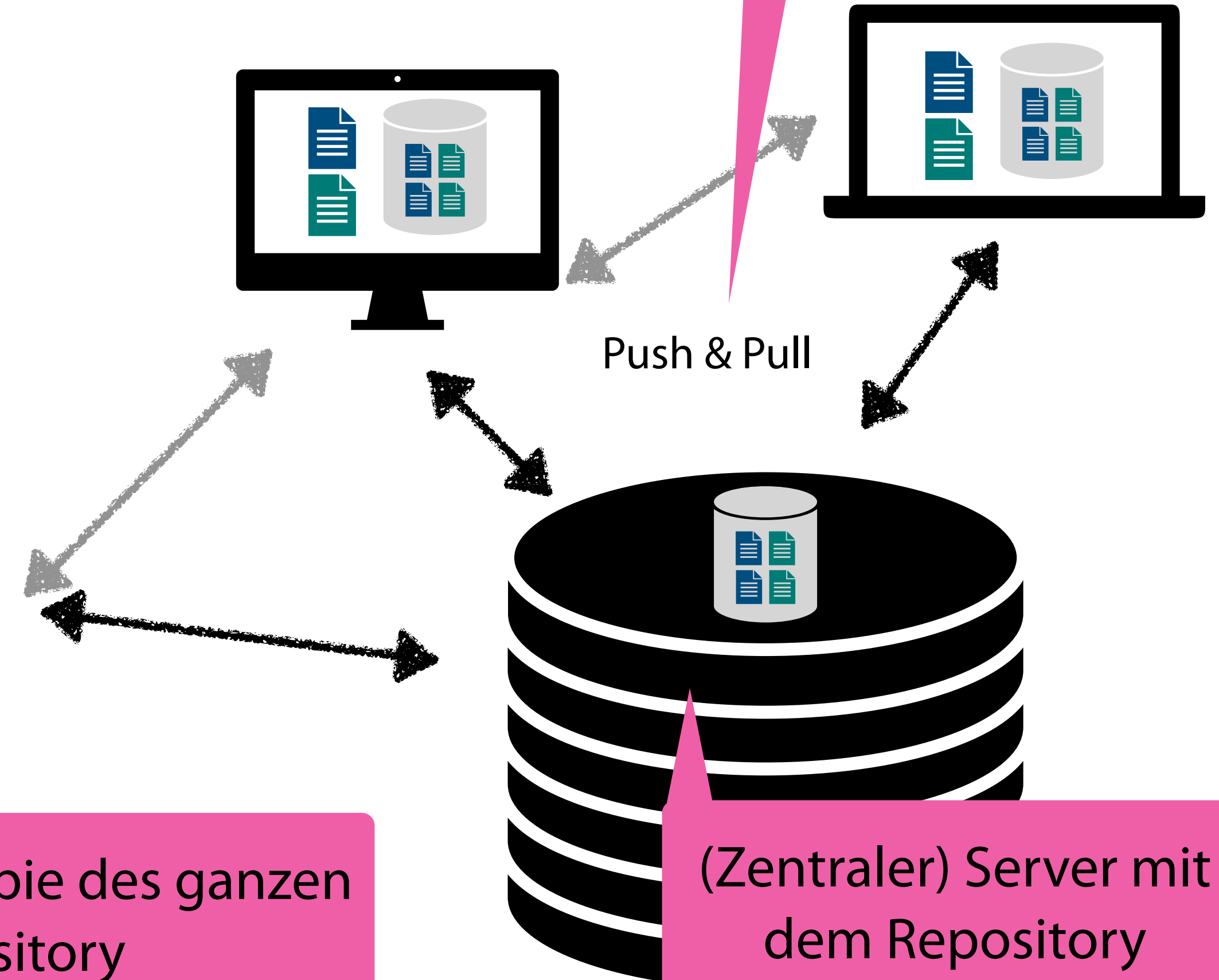
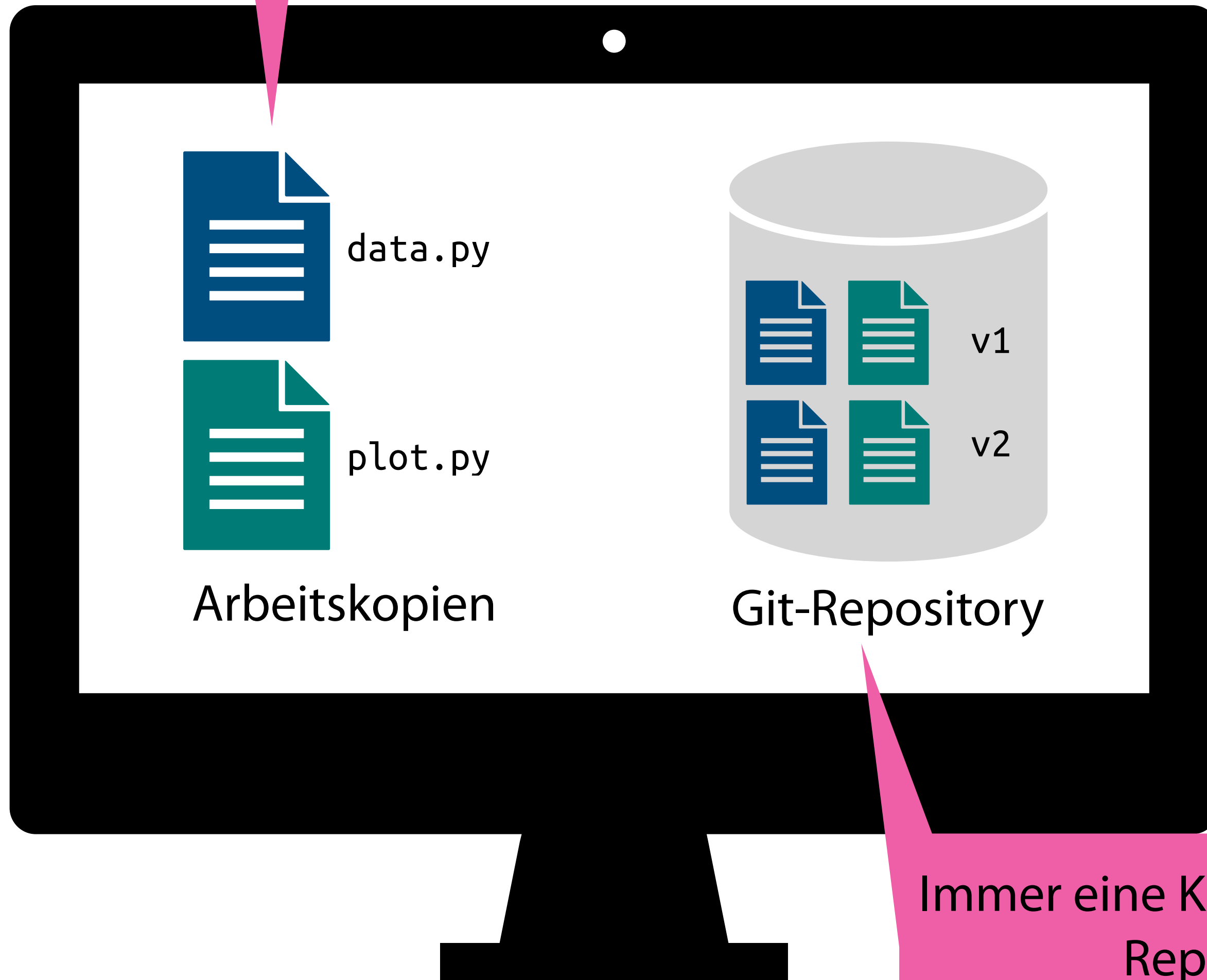
Verteilte Repositories



Dateiversion zum Bearbeiten
und ins Repository *commiten*
oder auch zurücksetzen

Verteilte Repositories

Zusammenführen und
austauschen der
Repository (haupt.
zwischen Client & Server)



Immer eine Kopie des ganzen
Repository

(Zentraler) Server mit
dem Repository

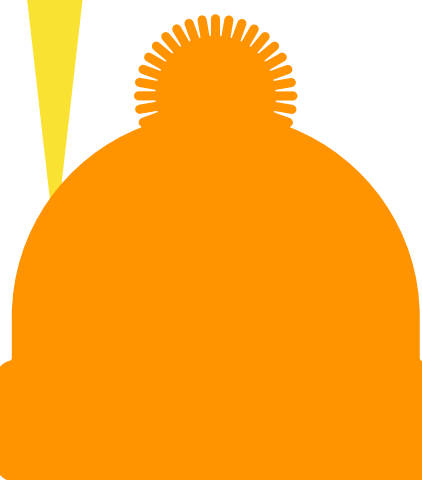
Git

- Mittlerweile Standard (insb. für OpenSource-Software)
 - Entwickelt von Linus Torvalds für Linux-Kernel
- SHA-1 Hash für jede Änderung
- Vollständig lokal nutzbar, kein (zentraler) Server nötig

Git

- Mittlerweile Standard (insb. für OpenSource-Software)
 - Entwickelt von Linus Torvalds für Linux-Kernel
- SHA-1 Hash für jede Änderung
- Vollständig lokal nutzbar, kein (zentraler) Server nötig

Ein Verlust der Daten auf einem Repository-Server lässt sich direkt aus dem lokalen Repository wiederherstellen.





Installation

- Vorinstalliert auf MacOS
- Paketquellen unter Linux
- Download und Anleitung für Windows
<https://git-scm.com/downloads>



Installation

- Vorinstalliert auf MacOS
- Paketquellen unter Linux
- Download und Anleitung für Windows
<https://git-scm.com/downloads>
- Push & Pull



Installation

- Vorinstalliert auf MacOS
- Paketquellen unter Linux
- Download und Anleitung für Windows
<https://git-scm.com/downloads>
- Push & Pull
- SSH Authentifikation
[Anleitung von GitHub](#)



Installation

- Vorinstalliert auf MacOS
- Paketquellen unter Linux
- Download und Anleitung für Windows
<https://git-scm.com/downloads>
- Push & Pull
- SSH Authentifikation
[Anleitung von GitHub](#)
- Alternativ mittels Username/ Passwort über HTTP(S)

Konfiguration

```
git config --global user.name "My Name"  
git config --global user.email "me@example.com"
```

Konfiguration

Gilt für ganzen Benutzeraccount,
ohne `--global` für aktuelles Repo.

```
git config --global user.name "My Name"  
git config --global user.email "me@example.com"
```


Konfiguration

Gilt für ganzen Benutzeraccount,
ohne `--global` für aktuelles Repo.

```
git config --global user.name "My Name"  
git config --global user.email "me@example.com"
```

- Commits (Änderungen Repository) werden damit versehen
 - Muss keine *echte* Mail und auch nicht der *echte* Name sein
- Weitere Konfiguration ist nicht notwendig

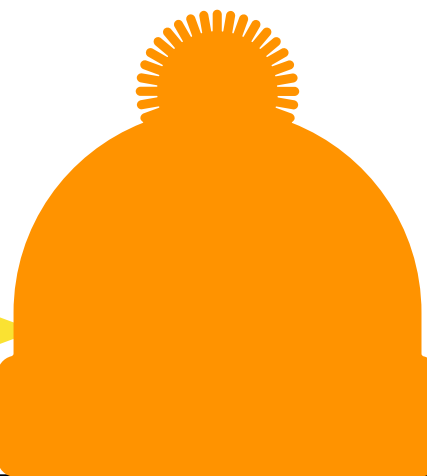
Konfiguration

Gilt für ganzen Benutzeraccount,
ohne `--global` für aktuelles Repo.

```
git config --global user.name "My Name"  
git config --global user.email "me@example.com"
```

- Commits (Änderungen Repository) werden damit versehen
 - Muss keine *echte* Mail und auch nicht der *echte* Name sein
- Weitere Konfiguration ist nicht notwendig

Tauscht man Commits aus
oder pusht sie in ein andere
Repository, dann werden
Namen & E-Mail geteilt.



Konfiguration

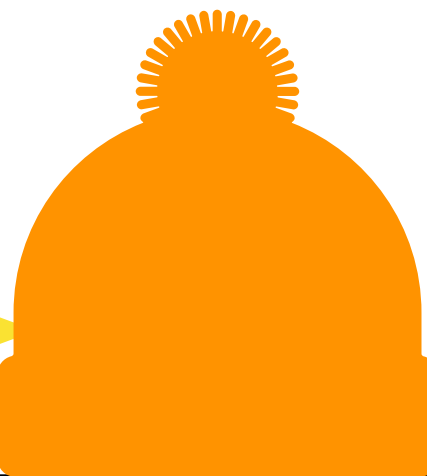
Gilt für ganzen Benutzeraccount,
ohne `--global` für aktuelles Repo.

```
git config --global user.name "My Name"  
git config --global user.email "me@example.com"
```

- Commits (Änderungen Repository) werden damit versehen
 - Muss keine *echte* Mail und auch nicht der *echte* Name sein
- Weitere Konfiguration ist nicht notwendig

Dann aber keine Zuordnung der
Commits möglich, daher eine
„öffentliche“ Mail-Adresse nutzen.

Tauscht man Commits aus
oder pusht sie in ein andere
Repository, dann werden
Namen & E-Mail geteilt.



II. Git

2. Lokal: Commit, Stash, Branch, Merge

Das erste Repository

```
$> git init  
Initialized empty Git repository in ./git/
```

```
$> tree -a .
```

```
.  
├── .git  
│   ├── HEAD  
│   ├── config  
│   ├── description  
│   ├── hooks  
│   │   └── ...  
│   ├── info  
│   │   └── exclude  
│   ├── objects  
│   │   ├── info  
│   │   └── pack  
│   └── refs  
│       ├── heads  
│       └── tags
```

Das erste Repository

```
$> git init  
Initialized empty Git repository in ./git/
```

```
$> tree -a .
```

```
.  
├── .git  
│   ├── HEAD  
│   ├── config  
│   ├── description  
│   ├── hooks  
│   │   └── ...  
│   ├── info  
│   │   └── exclude  
│   ├── objects  
│   │   ├── info  
│   │   └── pack  
│   └── refs  
│       ├── heads  
│       └── tags
```



data.py



plot.py

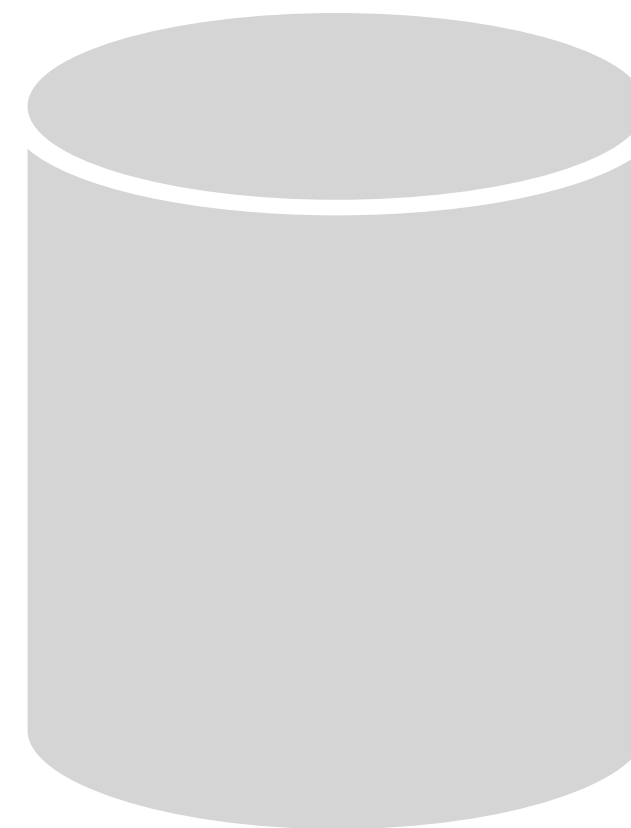
Arbeitskopien

Das erste Repository

```
$> git init  
Initialized empty Git repository in ./git/
```

```
$> tree -a .
```

```
.  
├── .git  
│   ├── HEAD  
│   ├── config  
│   ├── description  
│   ├── hooks  
│   │   └── ...  
│   ├── info  
│   │   └── exclude  
│   ├── objects  
│   │   ├── info  
│   │   └── pack  
│   └── refs  
│       ├── heads  
│       └── tags
```



Git-Repository



data.py



plot.py

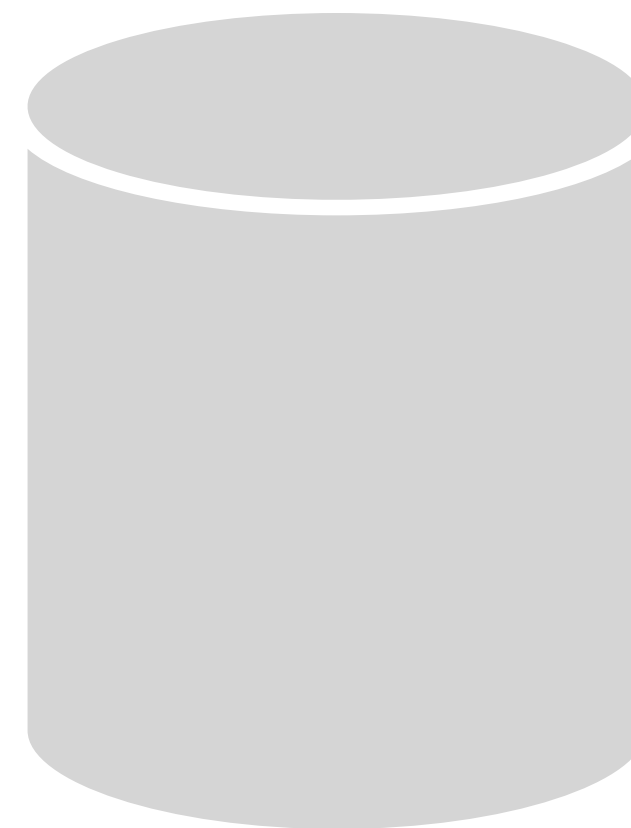
Arbeitskopien

Das erste Repository

```
$> git init  
Initialized empty Git repository in ./git/
```

```
$> tree -a .
```

```
.  
├── .git  
│   ├── HEAD  
│   ├── config  
│   ├── description  
│   ├── hooks  
│   │   └── ...  
│   ├── info  
│   │   └── exclude  
│   ├── objects  
│   │   ├── info  
│   │   └── pack  
│   └── refs  
│       ├── heads  
│       └── tags
```



Git-Repository



data.py



plot.py

Arbeitskopien

- Neues leeres Repo `./git/`
- Arbeitskopien `./`

Dateien hinzufügen

Git-Repository

Stash

Working Tree

Index

Repository

Remote

Dateien hinzufügen

Git-Repository

Stash

Working Tree

Index

Repository

Remote

```
$> git status  
On branch main  
  No commits yet  
$> touch a.txt
```

Dateien hinzufügen

Git-Repository

Stash

Working Tree

Index

Repository

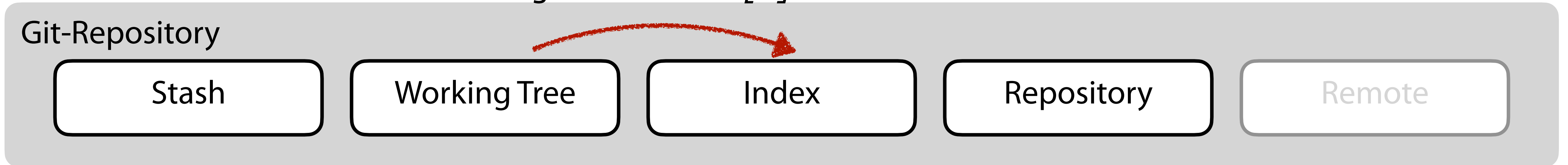
Remote

```
$> git status  
On branch main  
No commits yet  
$> touch a.txt
```

```
$> git status  
On branch main  
No commits yet  
Untracked files:  
  a.txt  
$> git add .
```

Dateien hinzufügen

`git add FILE[S]`

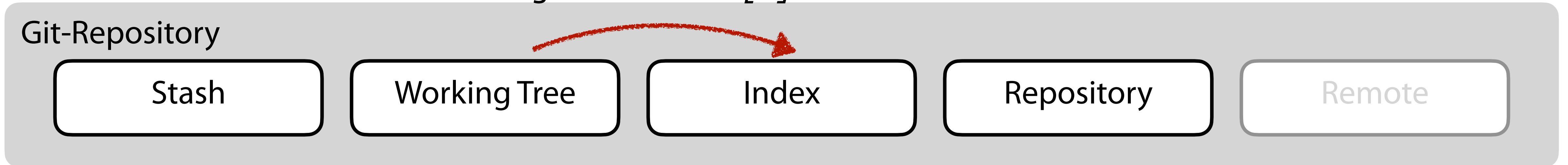


```
$> git status
On branch main
No commits yet
$> touch a.txt
```

```
$> git status
On branch main
No commits yet
Untracked files:
  a.txt
$> git add .
```

Dateien hinzufügen

`git add FILE[S]`



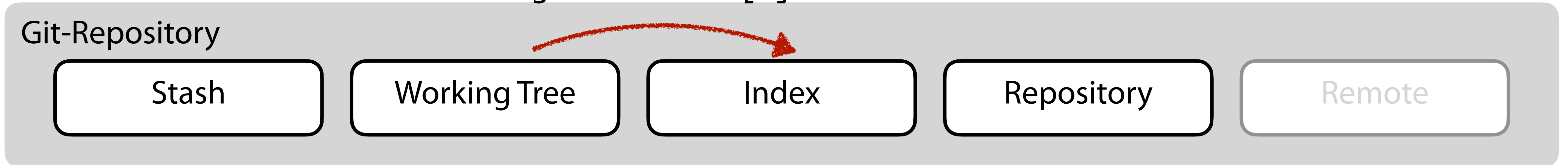
```
$> git status
On branch main
No commits yet
$> touch a.txt
```

```
$> git status
On branch main
No commits yet
Untracked files:
  a.txt
$> git add .
```

```
$> git status
On branch main
No commits yet
Changes to be committed:
  new file:   a.txt
```

Dateien hinzufügen

`git add FILE[S]`



`git rm FILE[S]`

Datei aus dem Index löschen

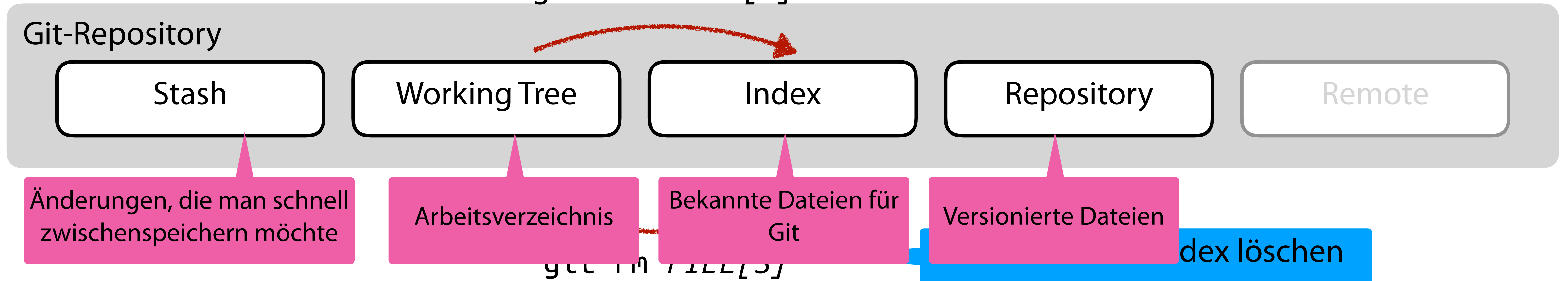
```
$> git status
On branch main
No commits yet
$> touch a.txt
```

```
$> git status
On branch main
No commits yet
Untracked files:
  a.txt
$> git add .
```

```
$> git status
On branch main
No commits yet
Changes to be committed:
  new file:   a.txt
```

Dateien hinzufügen

`git add FILE[S]`



```
$> git status
On branch main
No commits yet
$> touch a.txt
```

```
$> git status
On branch main
No commits yet
Untracked files:
  a.txt
$> git add .
```

```
$> git status
On branch main
No commits yet
Changes to be committed:
  new file:   a.txt
```

Commit

Git-Repository

Stash

Working Tree

Index

Repository

Remote

```
$> git status
On branch main
No commits yet
Changes to be committed:
  new file:   a.txt
  new file:   b.txt
```


Commit

Git-Repository

Stash

Working Tree

Index

Repository

Remote

```
$> git status
On branch main
No commits yet
Changes to be committed:
  new file:   a.txt
  new file:   b.txt
```

```
$> git commit -m "Meine Änderung"
[main (root-commit) 2655955] Meine Änderung
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a.txt
 create mode 100644 b.txt
$> git status
On branch main
nothing to commit, working tree clean
```

Commit

```
git commit [-m "Meine Änderung"]
```

Git-Repository

Stash

Working Tree

Index

Repository

Remote

```
$> git status
On branch main
No commits yet
Changes to be committed:
  new file:   a.txt
  new file:   b.txt
```

```
$> git commit -m "Meine Änderung"
[main (root-commit) 2655955] Meine Änderung
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a.txt
 create mode 100644 b.txt
$> git status
On branch main
nothing to commit, working tree clean
```

Commit

```
git commit [-m "Meine Änderung"]
```

Git-Repository

Stash

Working Tree

Index

Repository

Remote

Kurz beschreiben, was getan wurde
(ohne -m öffnet sich ein Editor für mehr Text)

```
$> git status
On branch main
No commits yet
Changes to be committed:
  new file:   a.txt
  new file:   b.txt
```

```
$> git commit -m "Meine Änderung"
[main (root-commit) 2655955] Meine Änderung
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a.txt
 create mode 100644 b.txt
$> git status
On branch main
nothing to commit, working tree clean
```

Commit

```
git commit [-m "Meine Änderung"]
```

Git-Repository

Stash

Working Tree

Index

Repository

Remote

Kurz beschreiben, was getan wurde
(ohne -m öffnet sich ein Editor für mehr Text)

```
$> git status
On branch main
No commits yet
Changes to be committed:
  new file:   a.txt
  new file:   b.txt
```

```
$> git commit -m "Meine Änderung"
[main (root-commit) 2655955] Meine Änderung
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a.txt
 create mode 100644 b.txt
$> git status
On branch main
nothing to commit, working tree clean
```

Man löscht keine Commits, denn andere könnten die bereits haben und darauf aufbauen. Besser, neuer Commit mit Änderungen!

Weitere Änderungen

Git-Repository

Stash

Working Tree

Index

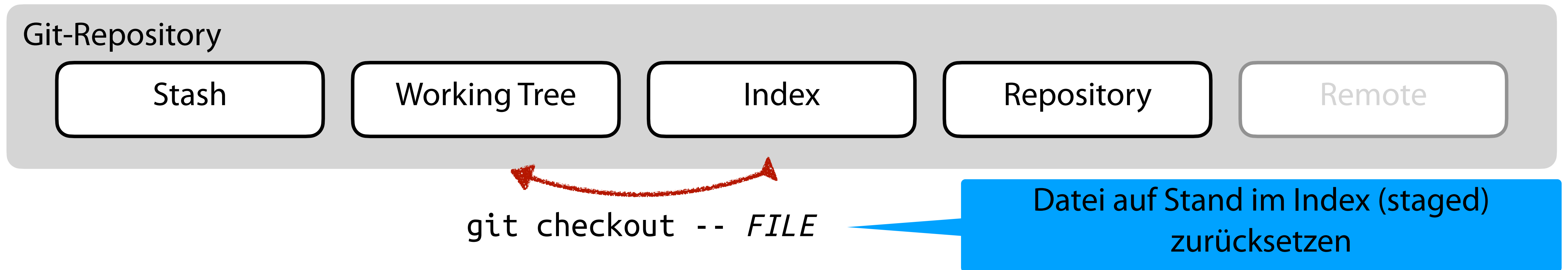
Repository

Remote

```
$> git status
On branch main
Changes not staged for commit:
  modified:   a.txt
$> git add .
$> vim b.txt
```

```
$> git status
On branch main
Changes to be committed:
  modified:   a.txt
Changes not staged for commit:
  modified:   b.txt
```

Weitere Änderungen

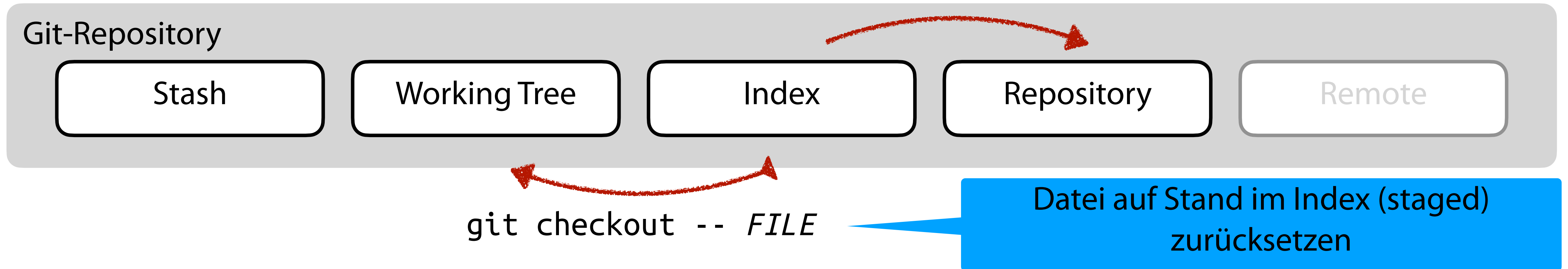


```
$> git status
On branch main
Changes not staged for commit:
  modified:   a.txt
$> git add .
$> vim b.txt
```

```
$> git status
On branch main
Changes to be committed:
  modified:   a.txt
Changes not staged for commit:
  modified:   b.txt
```


Weitere Änderungen

```
git commit [-m "Meine 2. Änderung"]
```

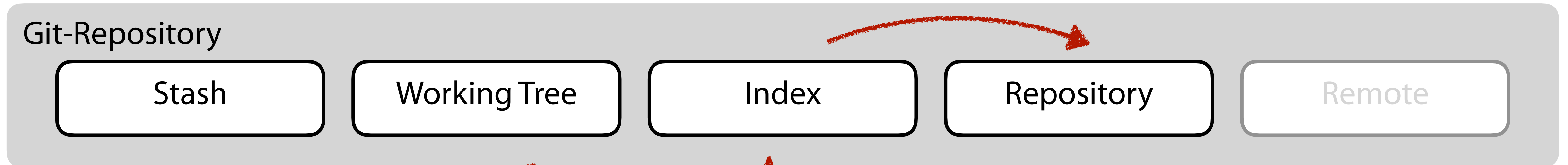


```
$> git status
On branch main
Changes not staged for commit:
  modified:   a.txt
$> git add .
$> vim b.txt
```

```
$> git status
On branch main
Changes to be committed:
  modified:   a.txt
Changes not staged for commit:
  modified:   b.txt
```

Weitere Änderungen

```
git commit [-m "Meine 2. Änderung"]
```



```
git checkout -- FILE
```

```
git restore [--staged] FILE
```

Datei auf Stand im Index (staged)
zurücksetzen

Datei auf Stand im Repository zurücksetzen
(--staged auch Index zurücksetzen)

```
$> git status
On branch main
Changes not staged for commit:
  modified:   a.txt
$> git add .
$> vim b.txt
```

```
$> git status
On branch main
Changes to be committed:
  modified:   a.txt
Changes not staged for commit:
  modified:   b.txt
```


Zwischenspeicher

Git-Repository

Stash

Working Tree

Index

Repository

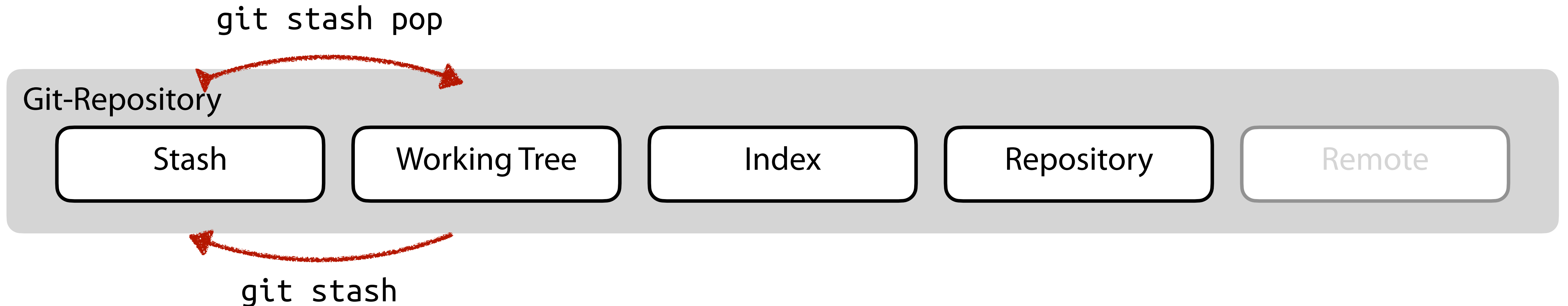
Remote

Zwischenspeicher



```
$> git status
On branch main
Changes to be committed:
  modified:   a.txt
$> git stash
Saved working directory ...
$> git status
On branch main
nothing to commit, working tree clean
```

Zwischenspeicher



```
$> git status
On branch main
Changes to be committed:
  modified:   a.txt
$> git stash
Saved working directory ...
$> git status
On branch main
nothing to commit, working tree clean
```

```
$> git stash pop
On branch main
Changes not staged for commit:
  modified:   a.txt
no changes added to commit
Dropped refs/stash@{0}
(0e71d0d32adcbf16fbe6a10c1f27436012d7f726)
```

Verlauf der Commits

```
$> git log --oneline
```


Verlauf der Commits

```
$> git log --oneline
```

```
d1a8079 (HEAD -> master, tag: v2.3.9, GH/master, GH/HEAD) UnRead does not use Podcast ID any more, s  
o moving podcast does not remove all "Reads"  
15c4a5c Update VERSION  
63b0336 Merge pull request #9 from DirkBaumeister/feature/added-configurable-setupapp-ident  
f92e070 fixed typo in readme file  
8752f21 added description of environment variable to readme  
2ea61d8 fixed typo in word ident  
1bda2ae added configurable setupapp ident string  
aa95130 (tag: v2.3.7) Add Preview  
90beeb0 Chnage year in copyright  
2db9bac (tag: v2.3.6) Optomize index.php exec. order  
aeecd70 (tag: v2.3.5) Fix sorting bug  
4b2140d (tag: v2.3.4) Sort stations and podcasts by name  
ebbd82b Current Dist & Docker Fix  
4d07b1a Update Docker  
b02209c (tag: v2.3.3) Fix PHP-8 Bug Stream.php  
fe888ef (tag: v2.3.2) Code Check und Docker PHP-8  
f5e56dd (tag: v2.3.1) Update jQuery  
83dfa19 Update startup-before.sh  
425d997 (tag: v2.3.0) Manage Un/Read in GUI Preview  
:
```

q um zu schließen

Verlauf der Commits

```
$> git log --oneline
```

```
d1a8079 (HEAD -> master, tag: v2.3.9, GH/master, GH/HEAD) UnRead does not use Podcast ID any more, s  
o moving podcast does not remove all "Reads"  
15c4a5c Update VERSION  
63b0336 Merge pull request #9 from DirkBaumeister/feature/added-configurable-setupapp-ident  
f92e070 fixed typo in readme file  
8752f21 added description of environment variable to readme  
2ea61d8 fixed typo in word ident  
1bda2ae added configurable setupapp ident string  
aa95130 (tag: v2.3.7) Add Preview  
90beeb0 Chnage year in copyright  
2db9bac (tag: v2.3.6) Optomize index.php exec. order  
aeecd70 (tag: v2.3.5) Fix sorting bug  
4b2140d (tag: v2.3.4) Sort stations and podcasts by name  
ebbd82b Current Dist & Docker Fix  
4d07b1a Update Docker  
b02209c (tag: v2.3.3) Fix PHP-8 Bug Stream.php  
fe888ef (tag: v2.3.2) Code Check und Docker PHP-8  
f5e56dd (tag: v2.3.1) Update jQuery  
83dfa19 Update startup-before.sh  
425d997 (tag: v2.3.0) Manage Un/Read in GUI Preview  
:
```

git checkout *HASH*
ändert Working Tree auf
Stand des Commits

q um zu schließen

Verlauf der Commits

```
$> git log --oneline
```

```
d1a8079 (HEAD -> master, tag: v2.3.9, GH/master, GH/HEAD) UnRead does not use Podcast ID any more, s  
o moving podcast does not remove all "Reads"  
15c4a5c Update VERSION  
63b0336 Merge pull request #9 from DirkBaumeister/feature/added-configurable-setupapp-ident  
f92e070 fixed typo in readme file  
8752f21 added description of environment variable to readme  
2ea61d8 fixed typo in word ident  
1bda2ae added configurable setupapp ident string  
aa95130 (tag: v2.3.7) Add Preview  
90beeb0 Chnage year in copyright  
2db9bac (tag: v2.3.6) Optomize index.php exec. order  
aeecd70 (tag: v2.3.5) Fix sorting bug  
4b2140d (tag: v2.3.4) Sort stations and podcasts by name  
ebbd82b Current Dist & Docker Fix  
4d07b1a Update Docker  
b02209c (tag: v2.3.3) Fix PHP-8 Bug Stream.php  
fe888ef (tag: v2.3.2) Code Check und Docker PHP-8  
f5e56dd (tag: v2.3.1) Update jQuery  
83dfa19 Update startup-before.sh  
425d997 (tag: v2.3.0) Manage Un/Read in GUI Preview  
:
```

git checkout *HASH*
ändert Working Tree auf
Stand des Commits

Jedoch dort keine
Änderungen möglich

Zurück mittels
git checkout HEAD

q um zu schließen

Anforderungen Versionsverwaltung

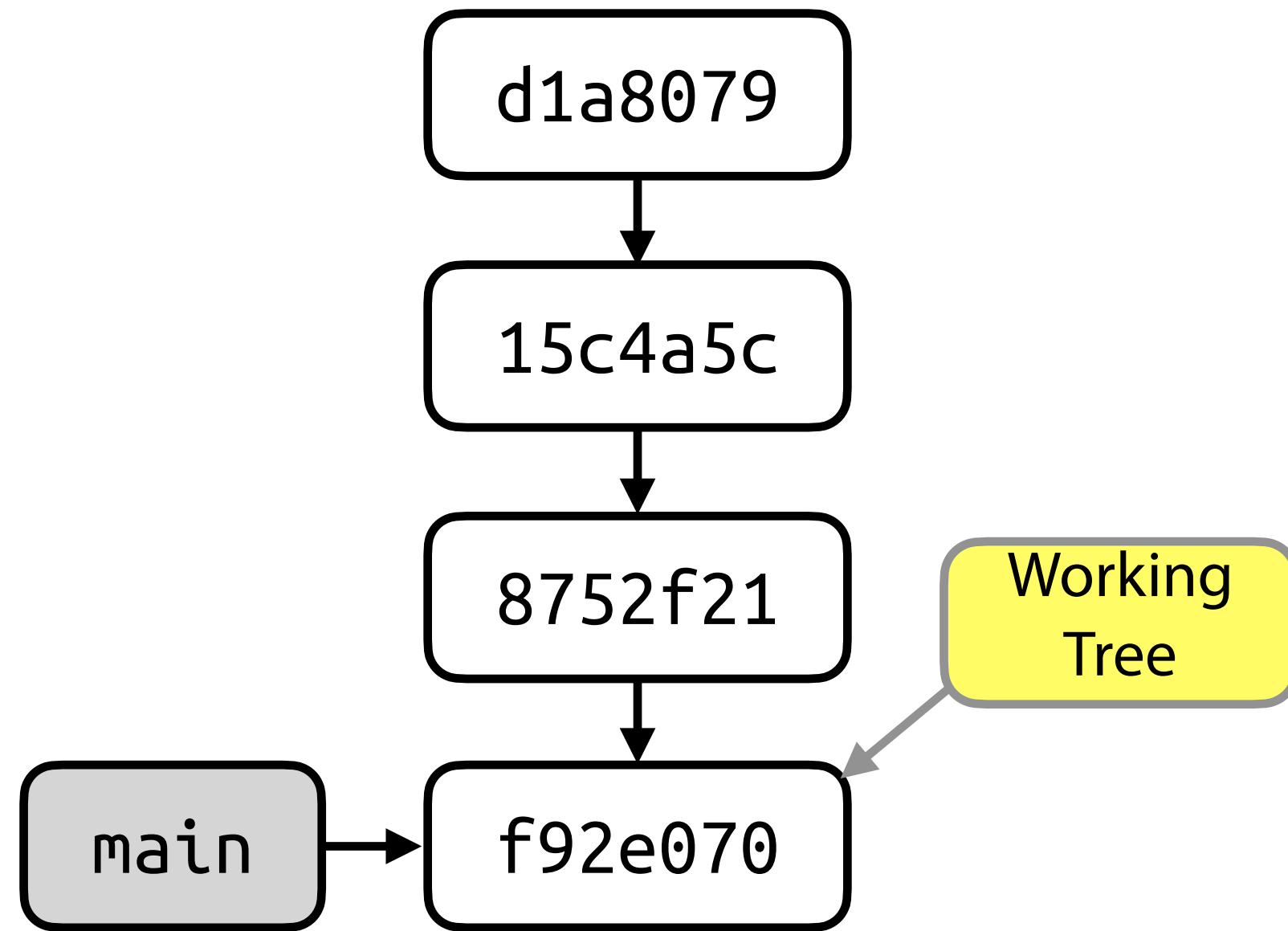
- Verlauf der Änderungen (textbasierte Dateien)
- Verschiedene **Entwicklungsweige** gleichzeitig
 - Verschiedene (neue) Features und Fehlerbehebungen
 - Verschiedene Orte
- Zusammenführen von **Entwicklungsweigen**
- Ein (zentrales) Repository

Branches

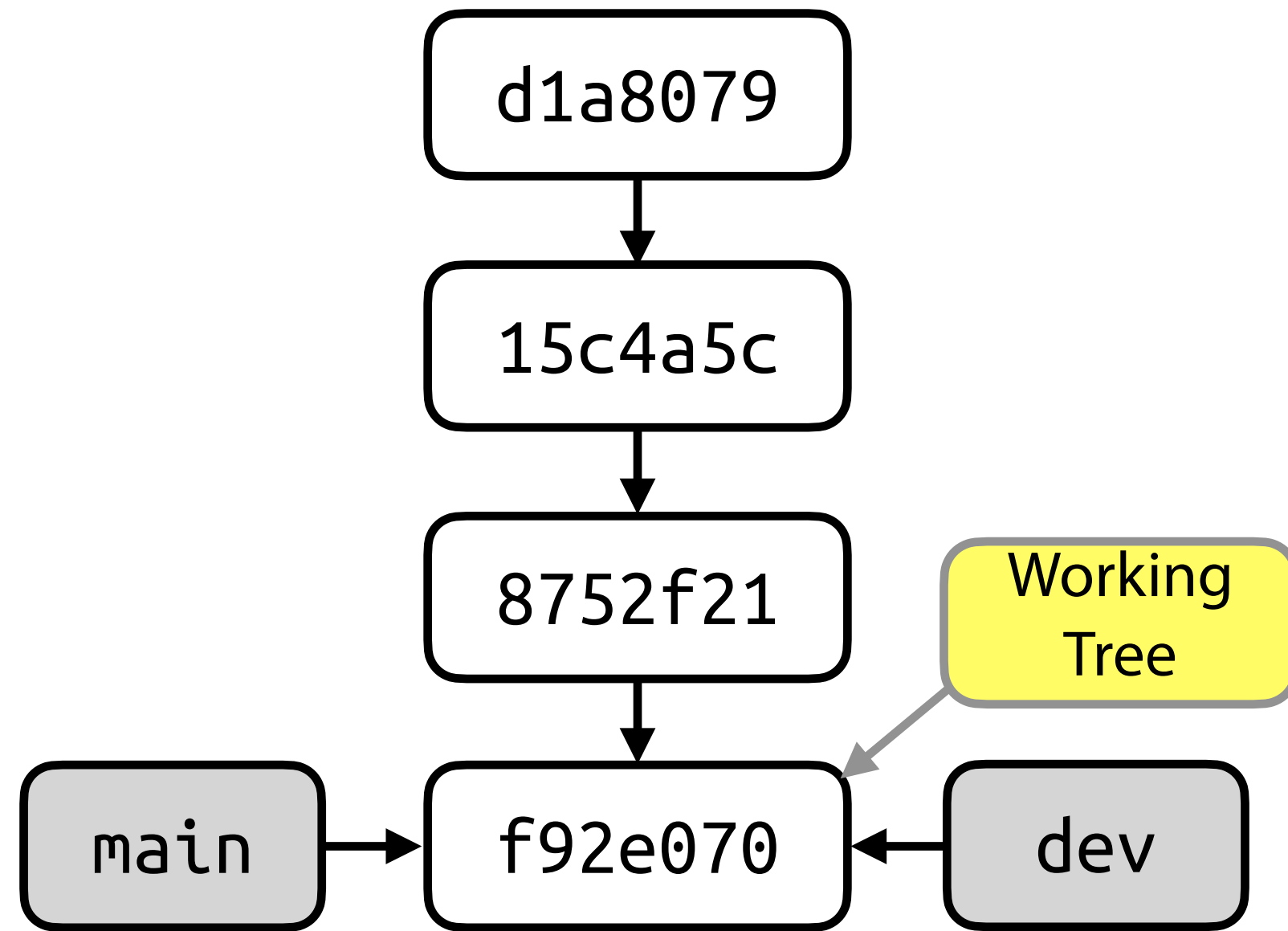
```
* d1a8079 (HEAD -> master, tag: v2.3.9, GH/master, GH/HEAD) UnRead does not use Podcast ID any more,
  so moving podcast does not remove all "Reads"
* 15c4a5c Update VERSION
* 63b0336 Merge pull request #9 from DirkBaumeister/feature/added-configurable-setupapp-ident
| \
| * f92e070 fixed typo in readme file
| * 8752f21 added description of environment variable to readme
| * 2ea61d8 fixed typo in word ident
| * 1bda2ae added configurable setupapp ident string
| /
* aa95130 (tag: v2.3.7) Add Preview
:
```

- Bisher auf Branch main (früher master)
- Verzweigen der Entwicklung und anschließend vereinen

Branches: Idee

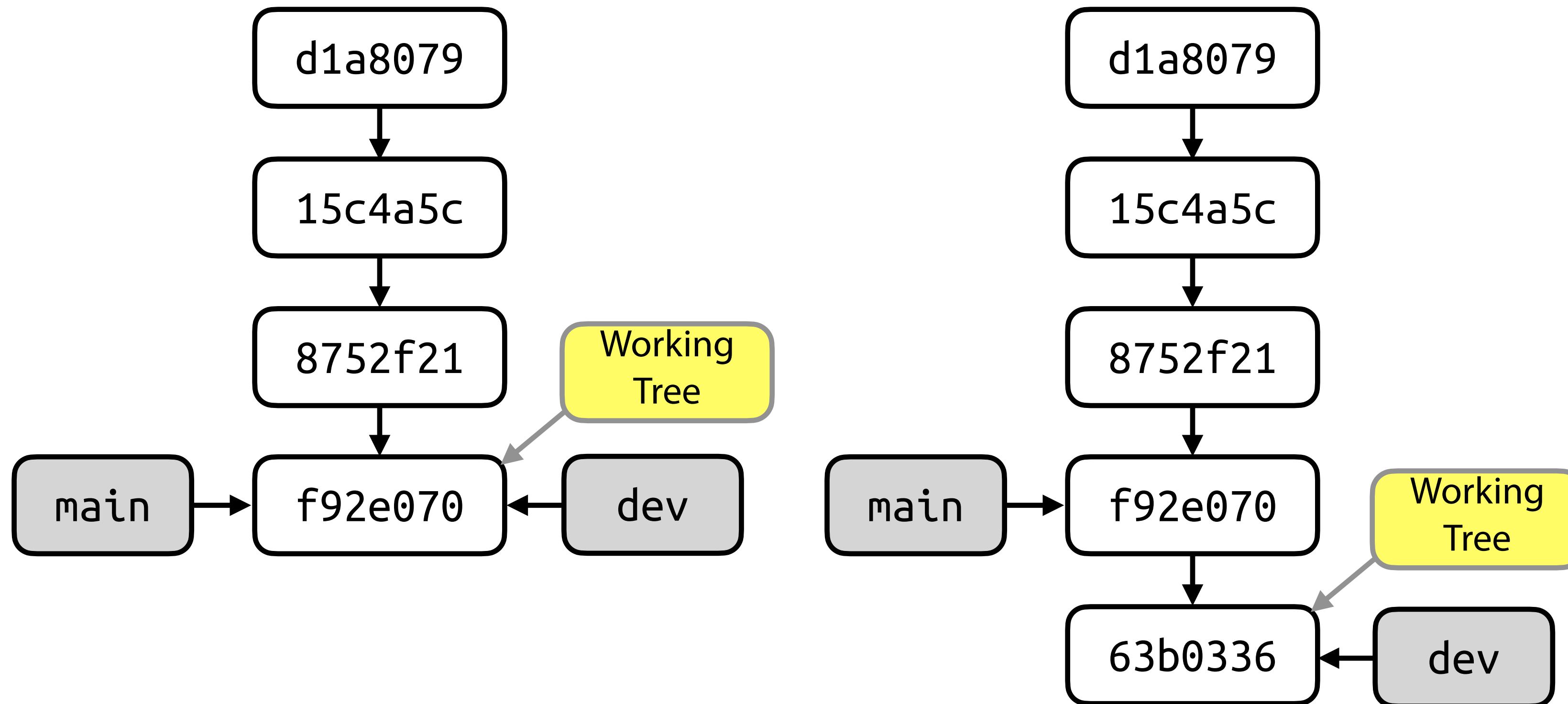


Branches: Idee



```
$> git branch dev
```

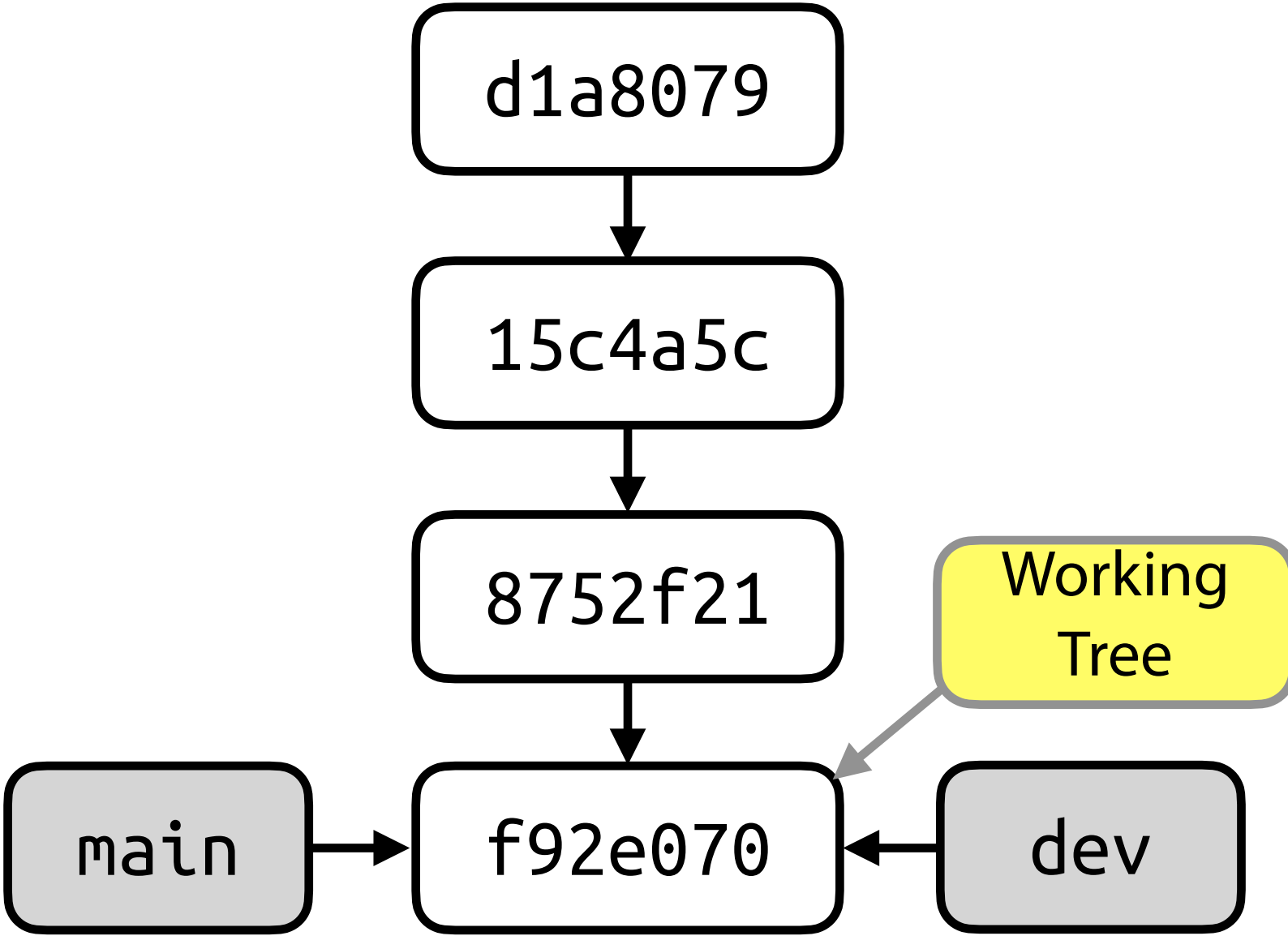
Branches: Idee



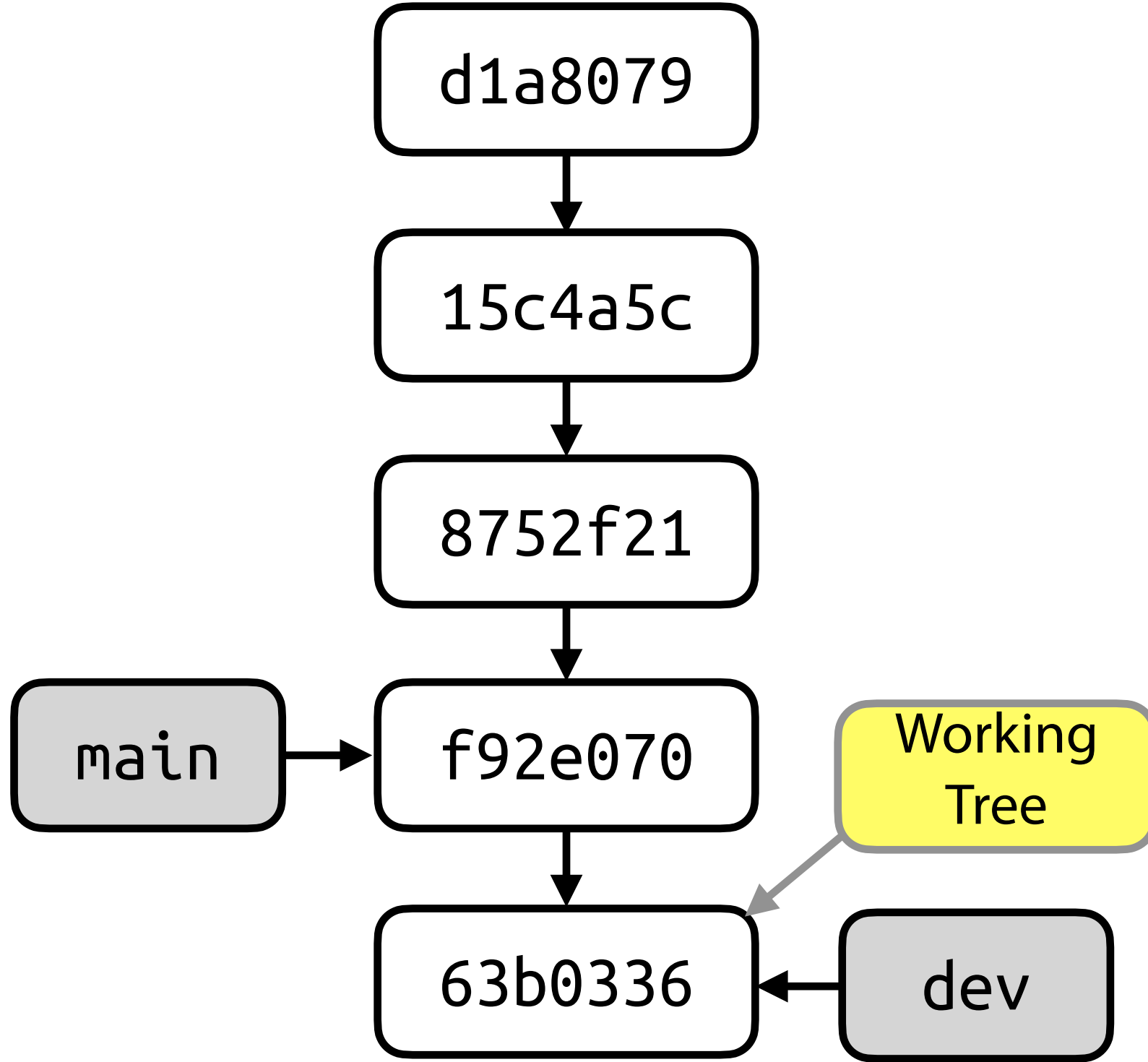
\$> git branch dev

\$> git checkout dev
\$> git commit

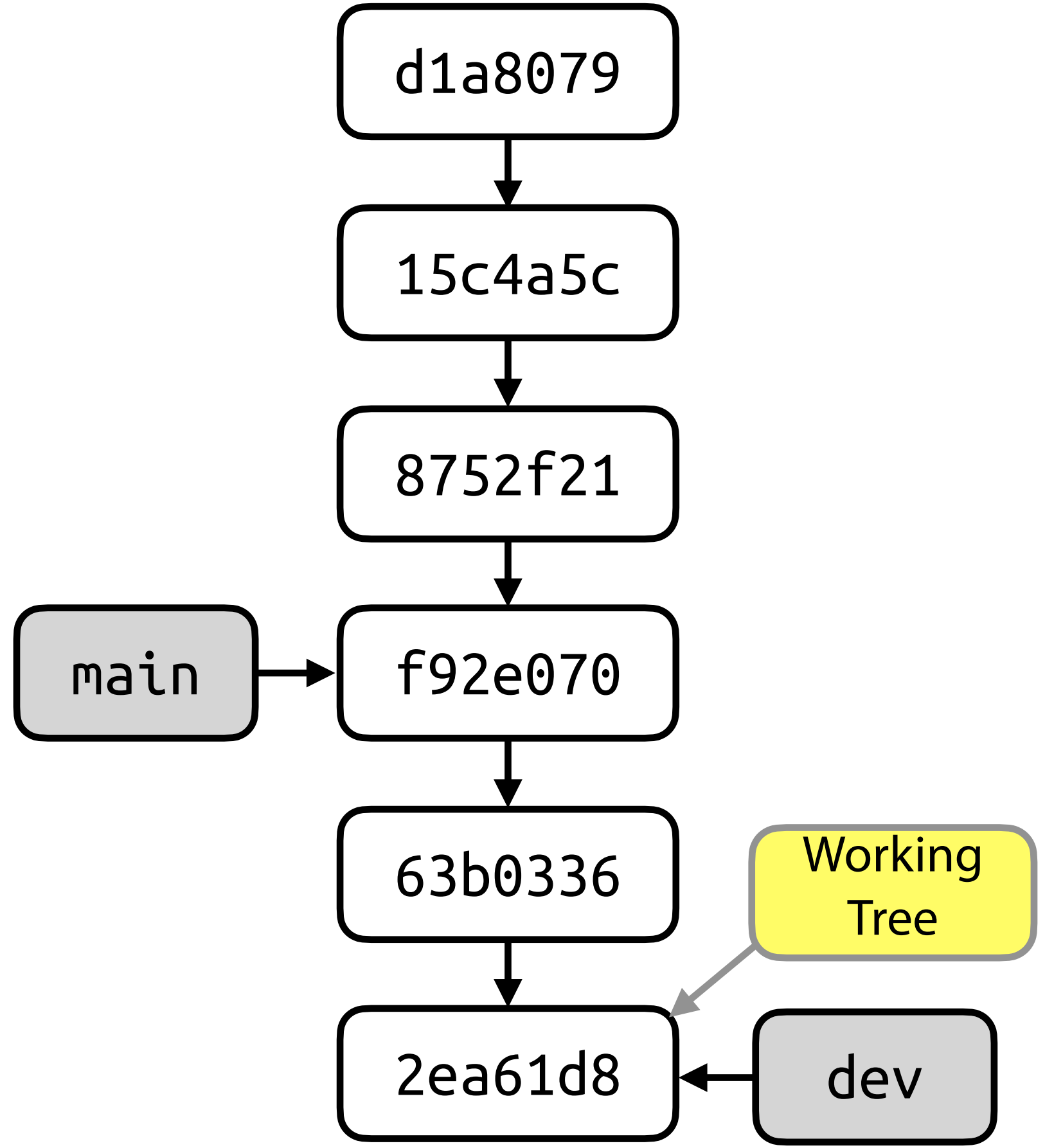
Branches: Idee



\$> git branch dev



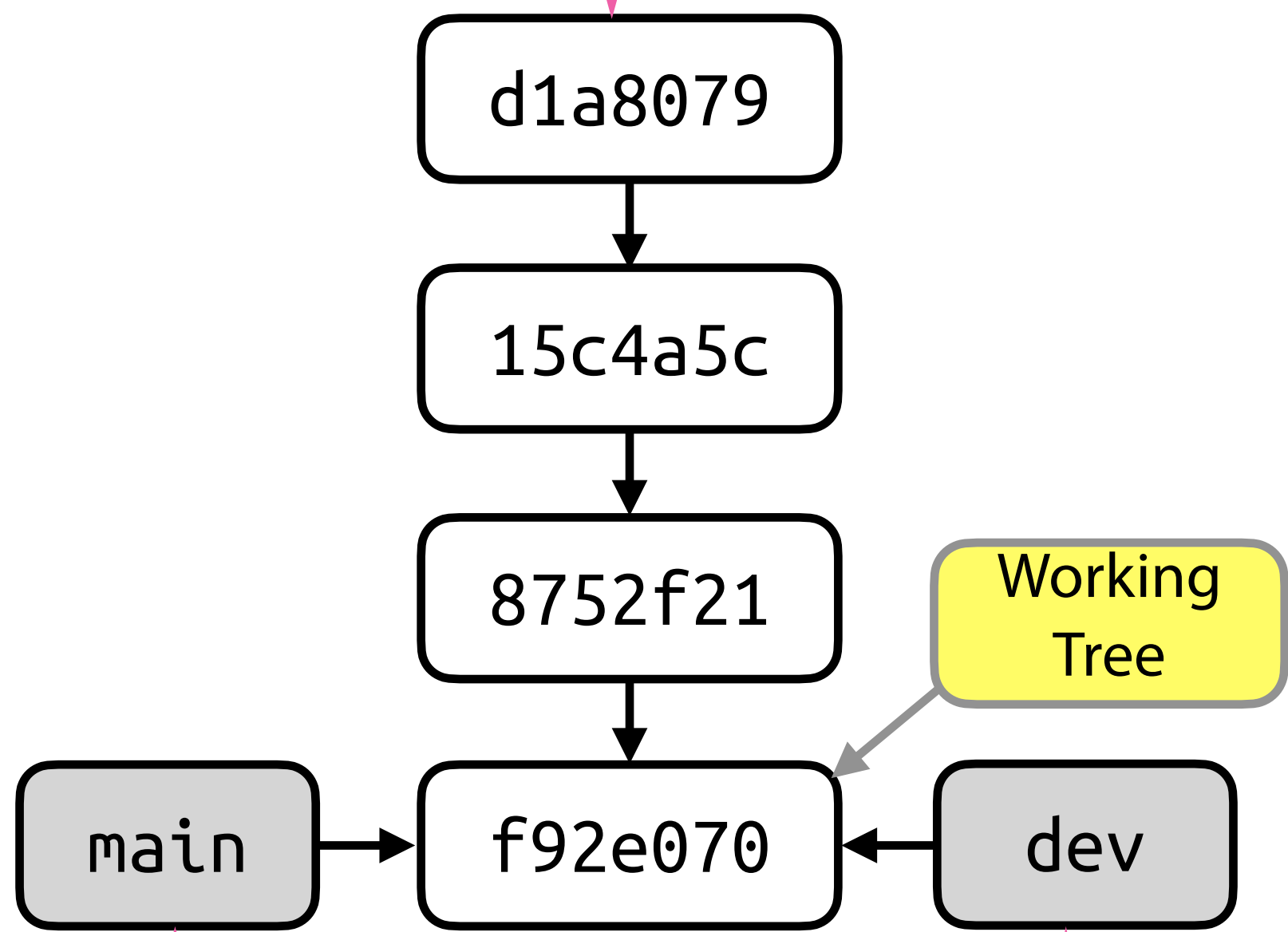
\$> git checkout dev
\$> git commit



\$> git commit

Branches: Idee

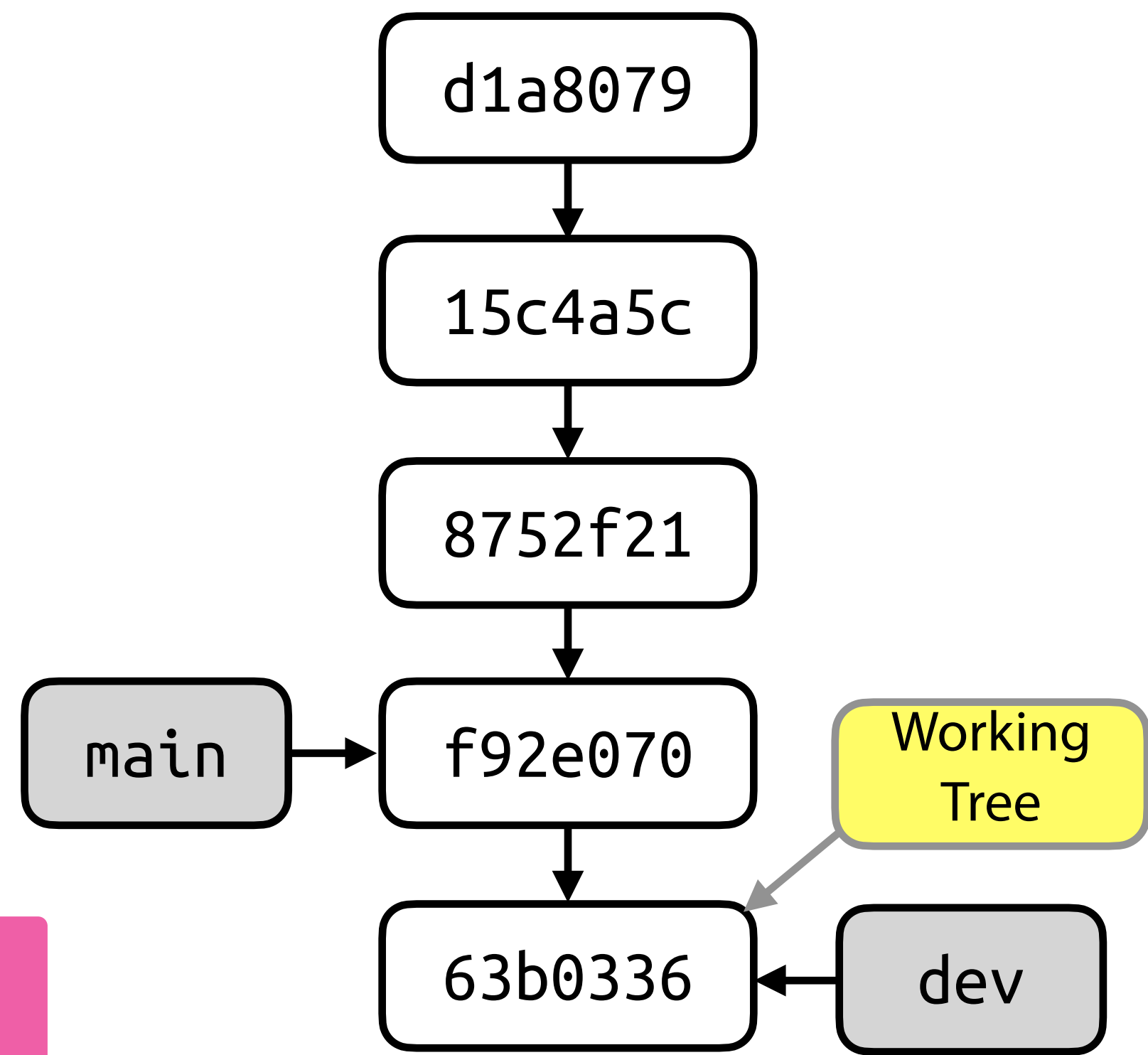
Verlauf der Commits



Branch ist ein Zeiger auf einen Commit

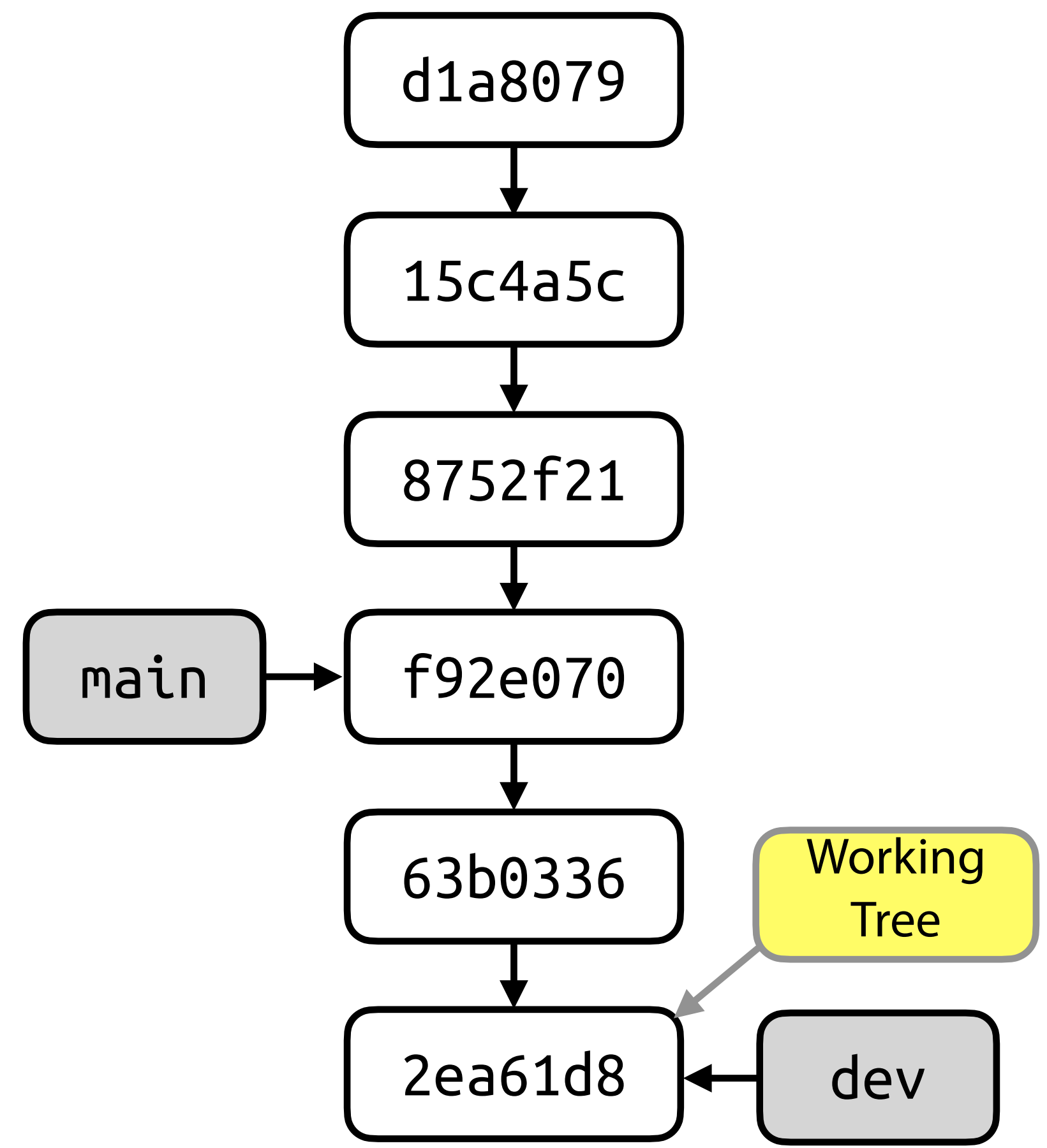
Ein weiterer neuer Branch

```
$> git branch dev
```



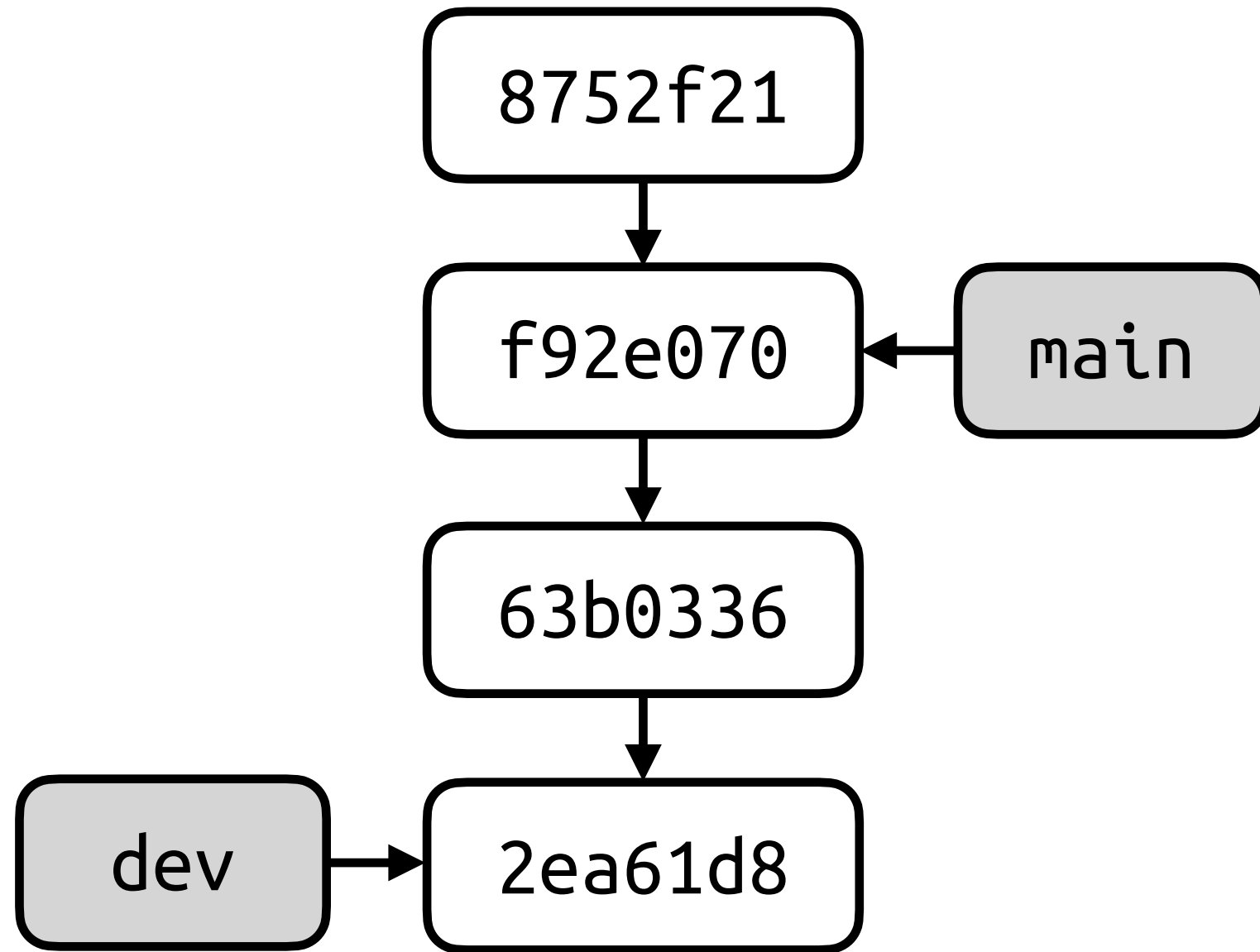
```
$> git checkout dev  
$> git commit
```

Commit auf dem Branch durchführen, damit weitere Commit im Verlauf, aber nur Marker dev bewegt sich.

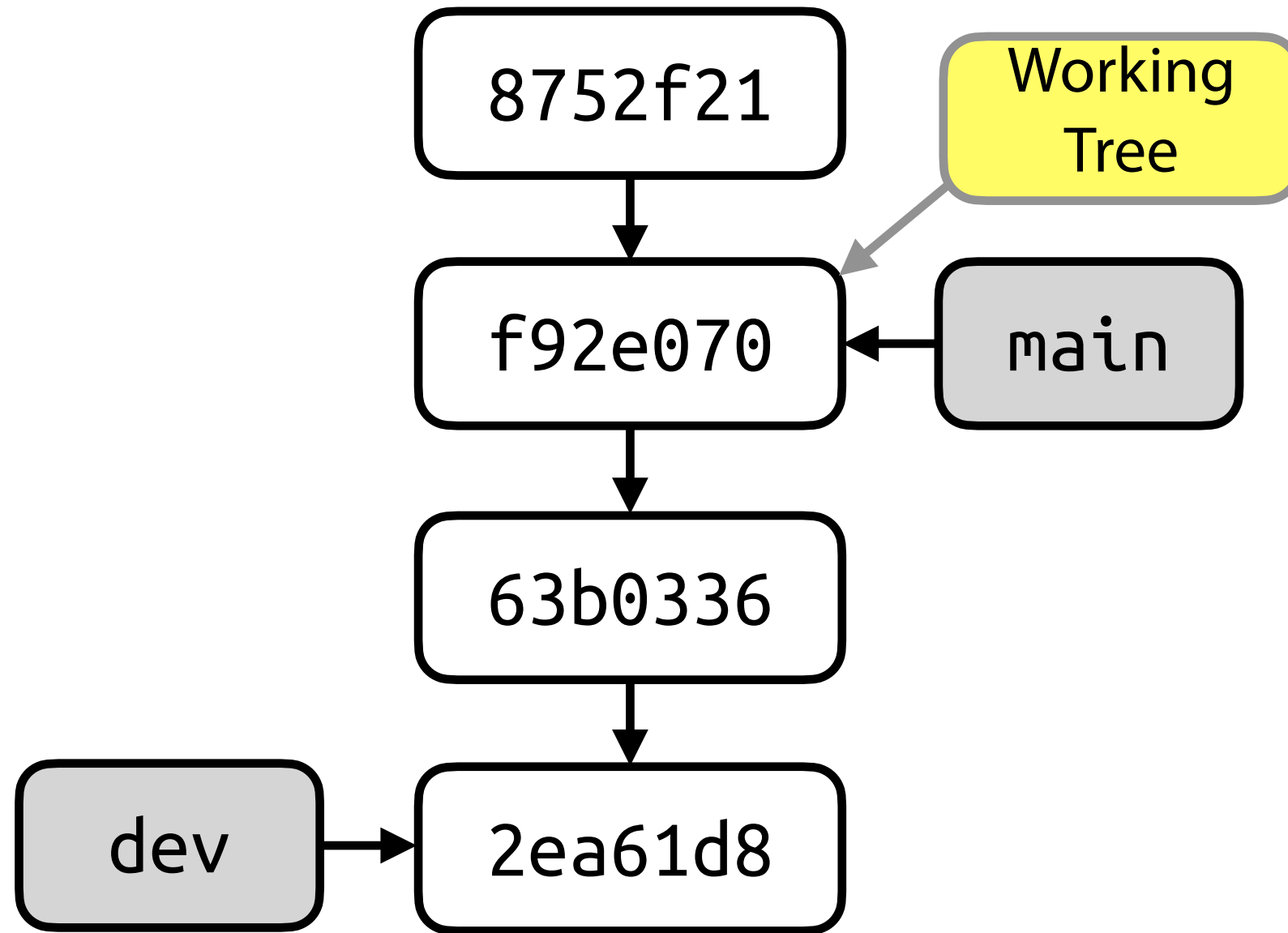


```
$> git commit
```

Branches: Verzweigung

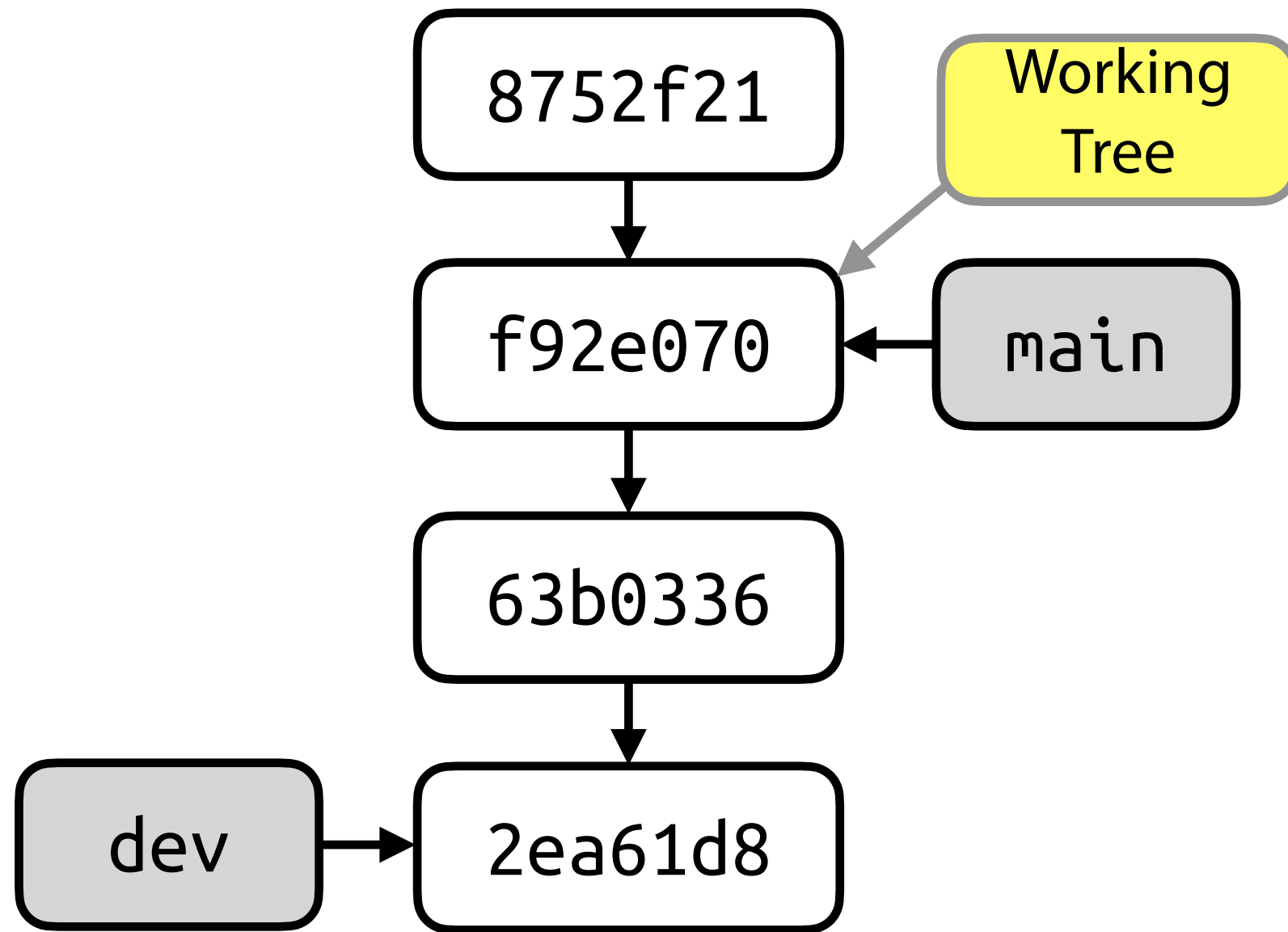


Branches: Verzweigung

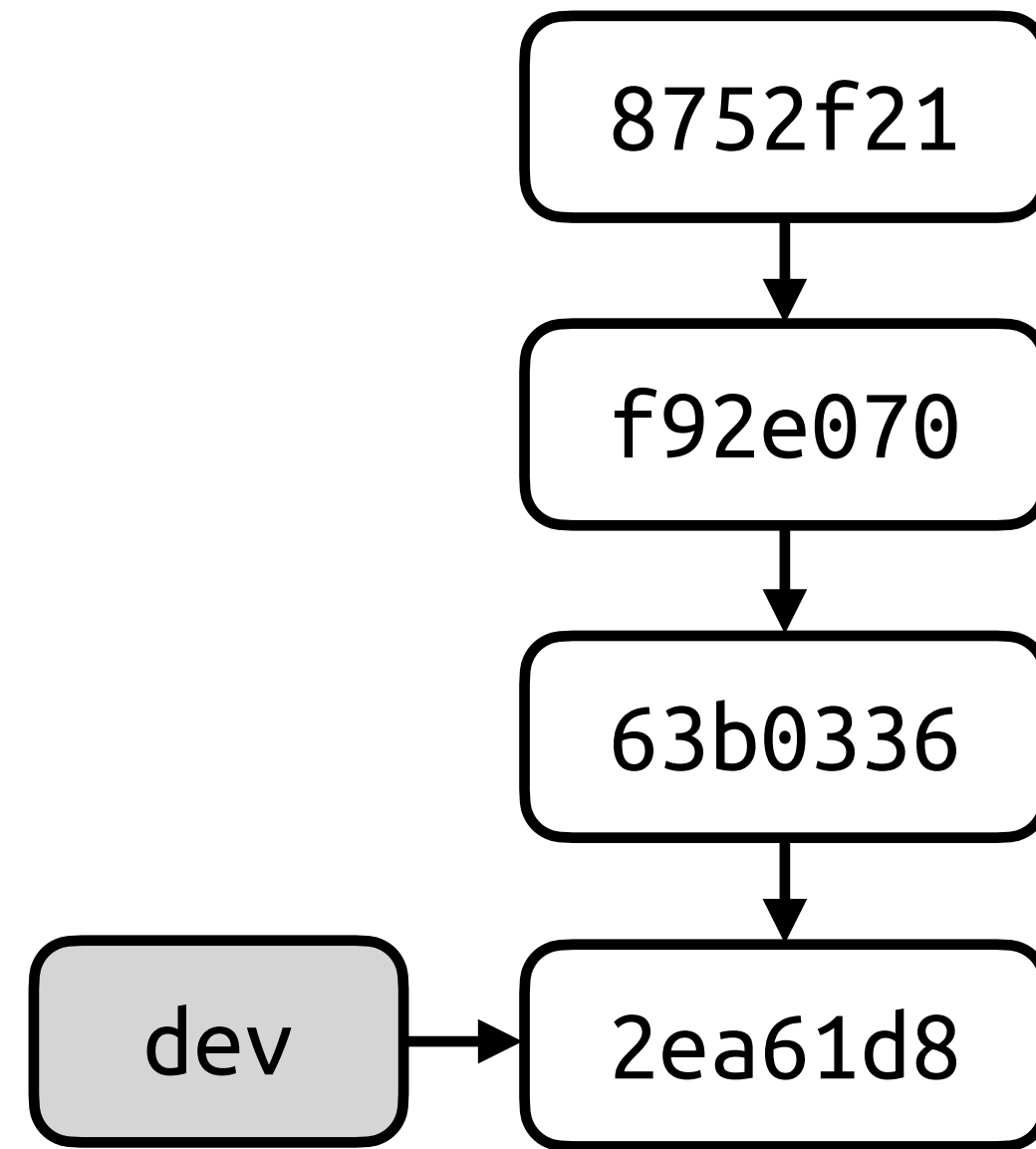


```
$> git checkout main
```


Branches: Verzweigung



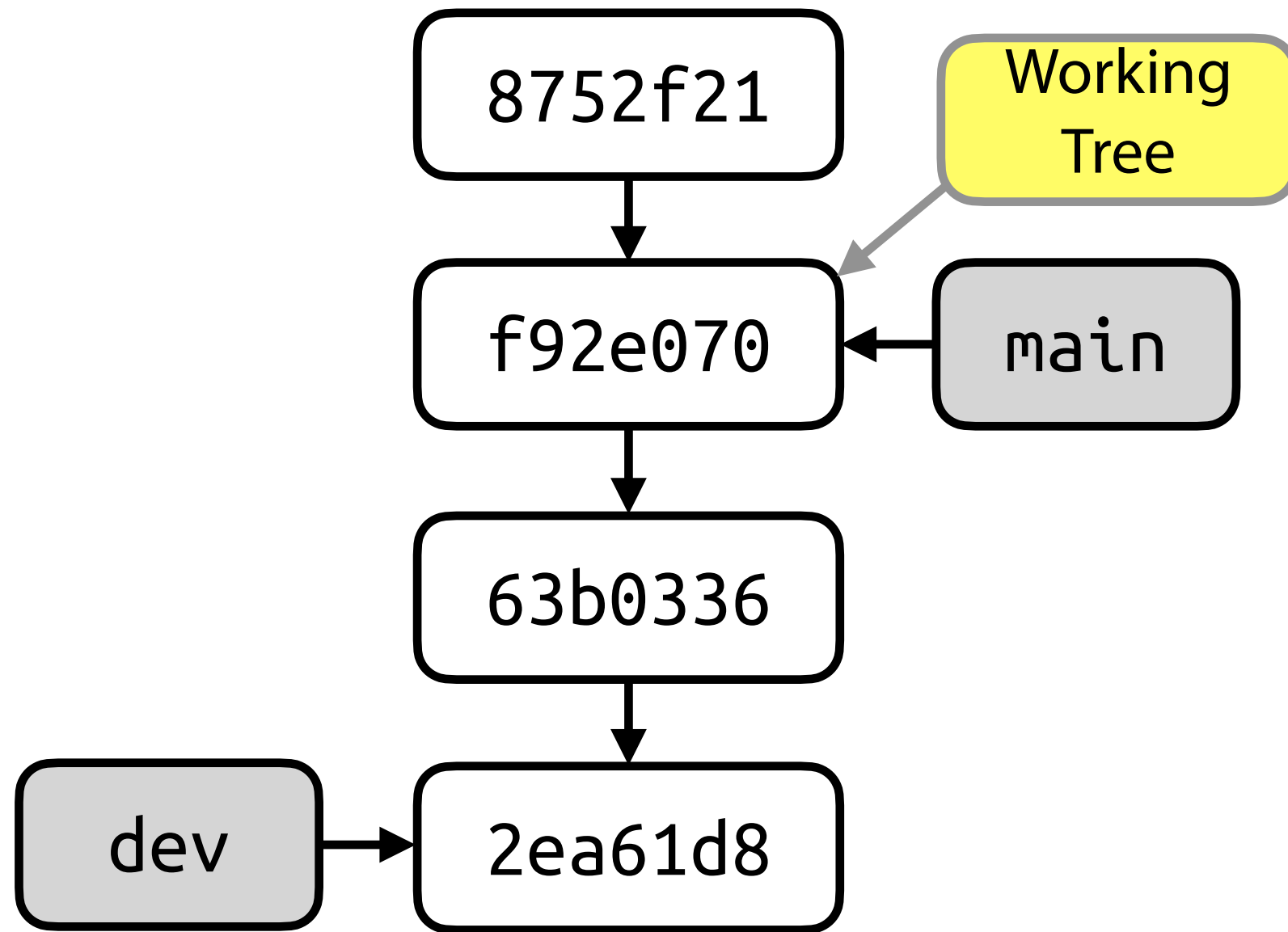
\$> git checkout main



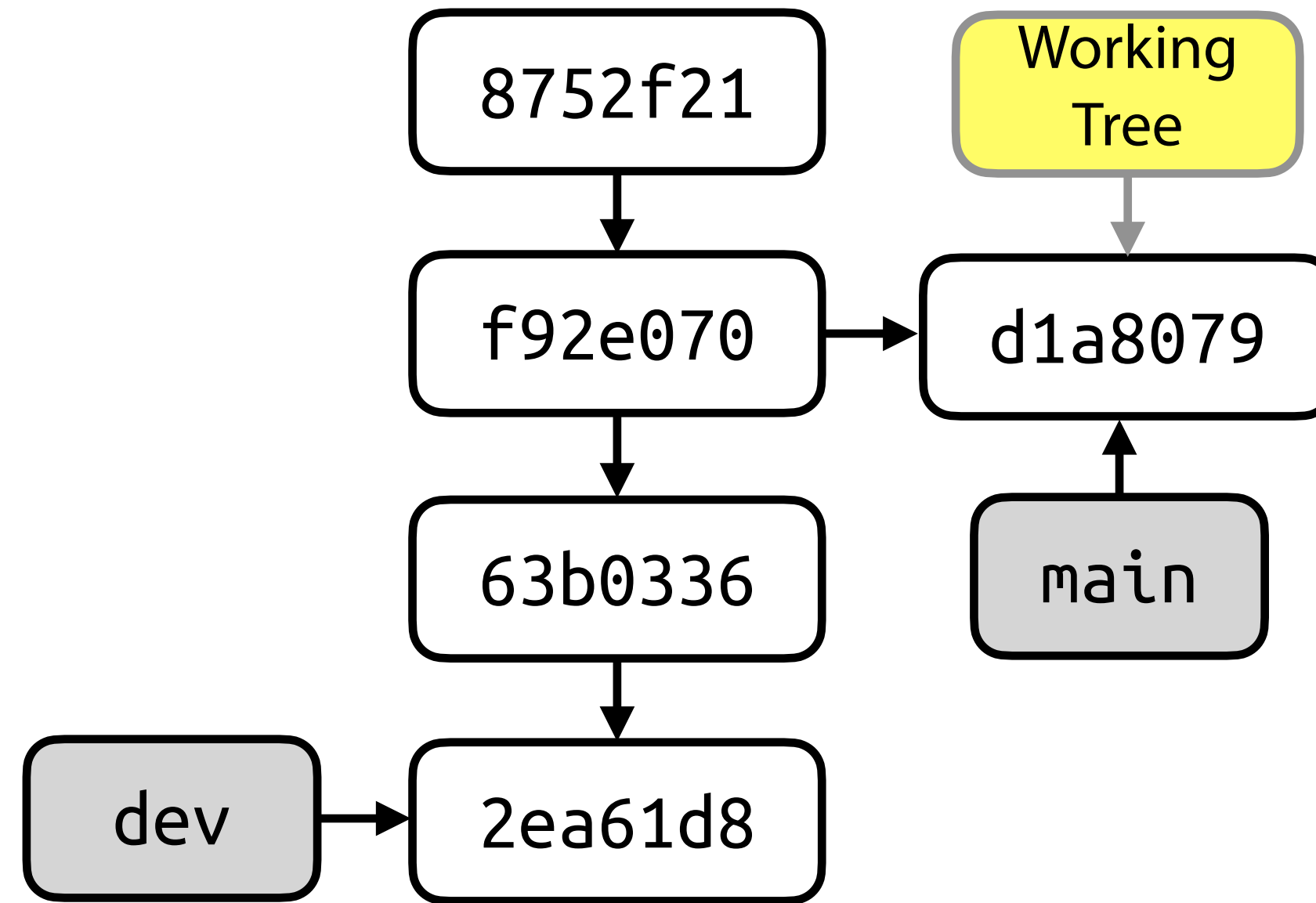
\$> git commit

Was passiert jetzt?

Branches: Verzweigung



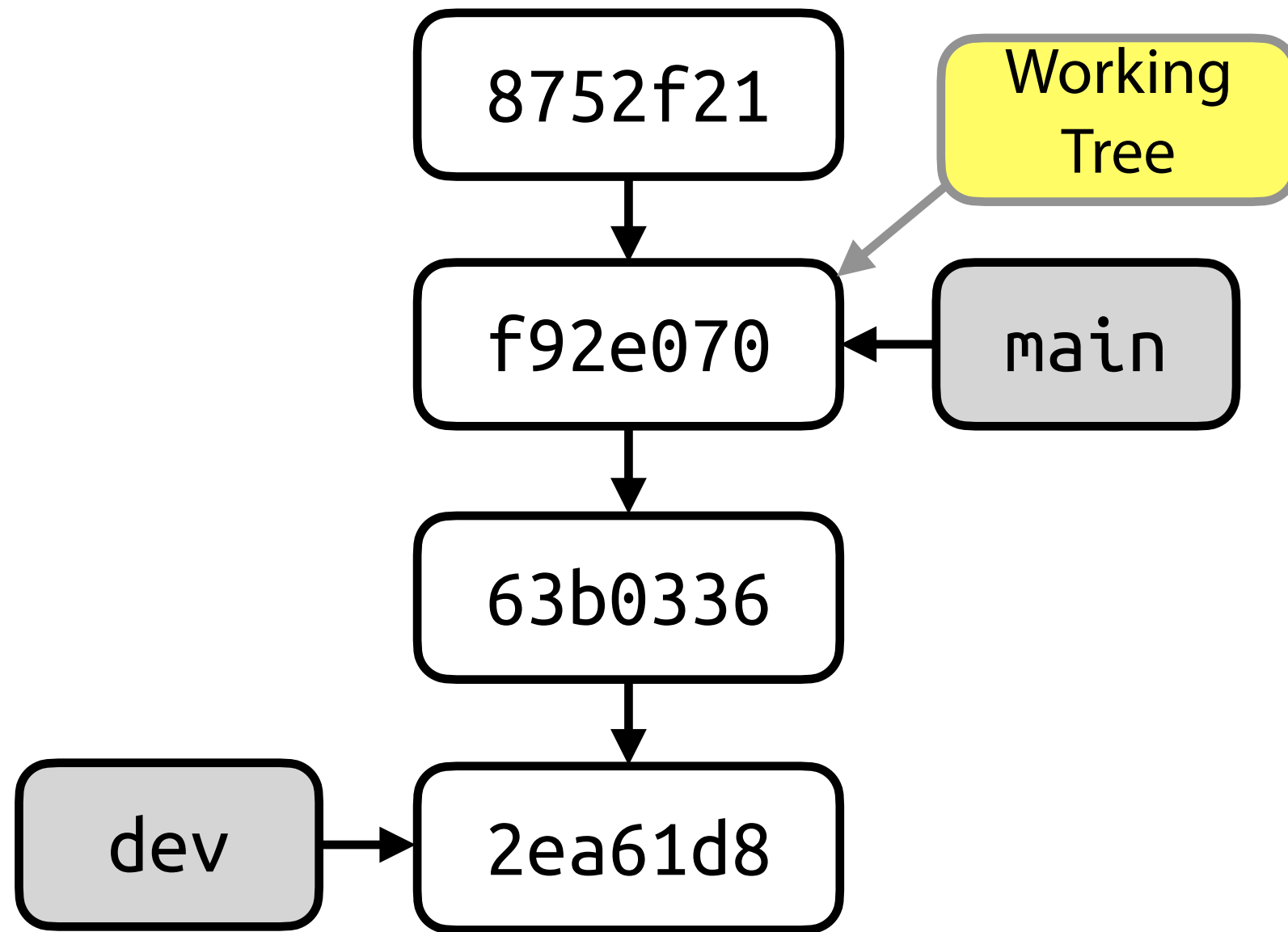
\$> git checkout main



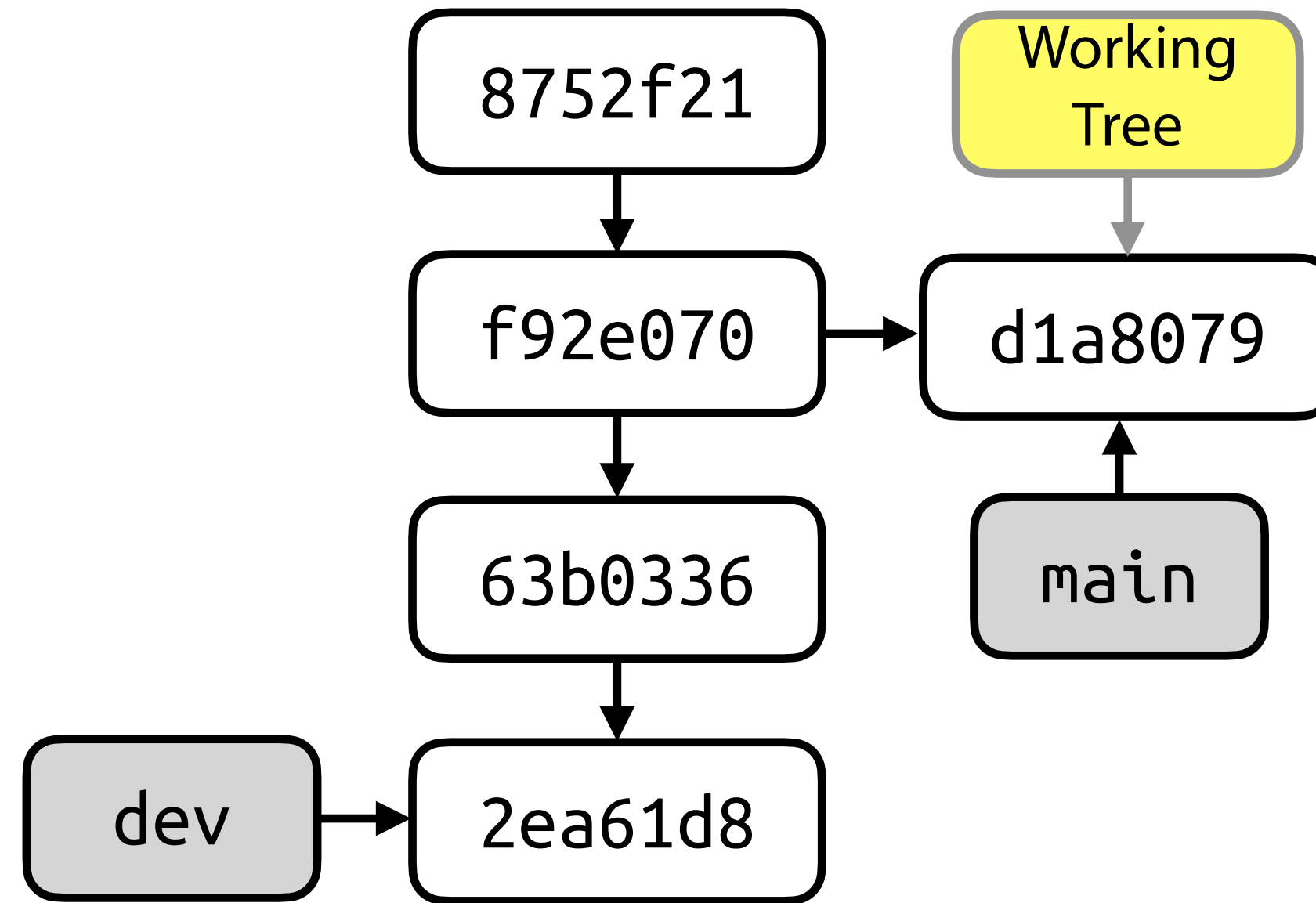
\$> git commit

Was passiert jetzt?

Branches: Verzweigung

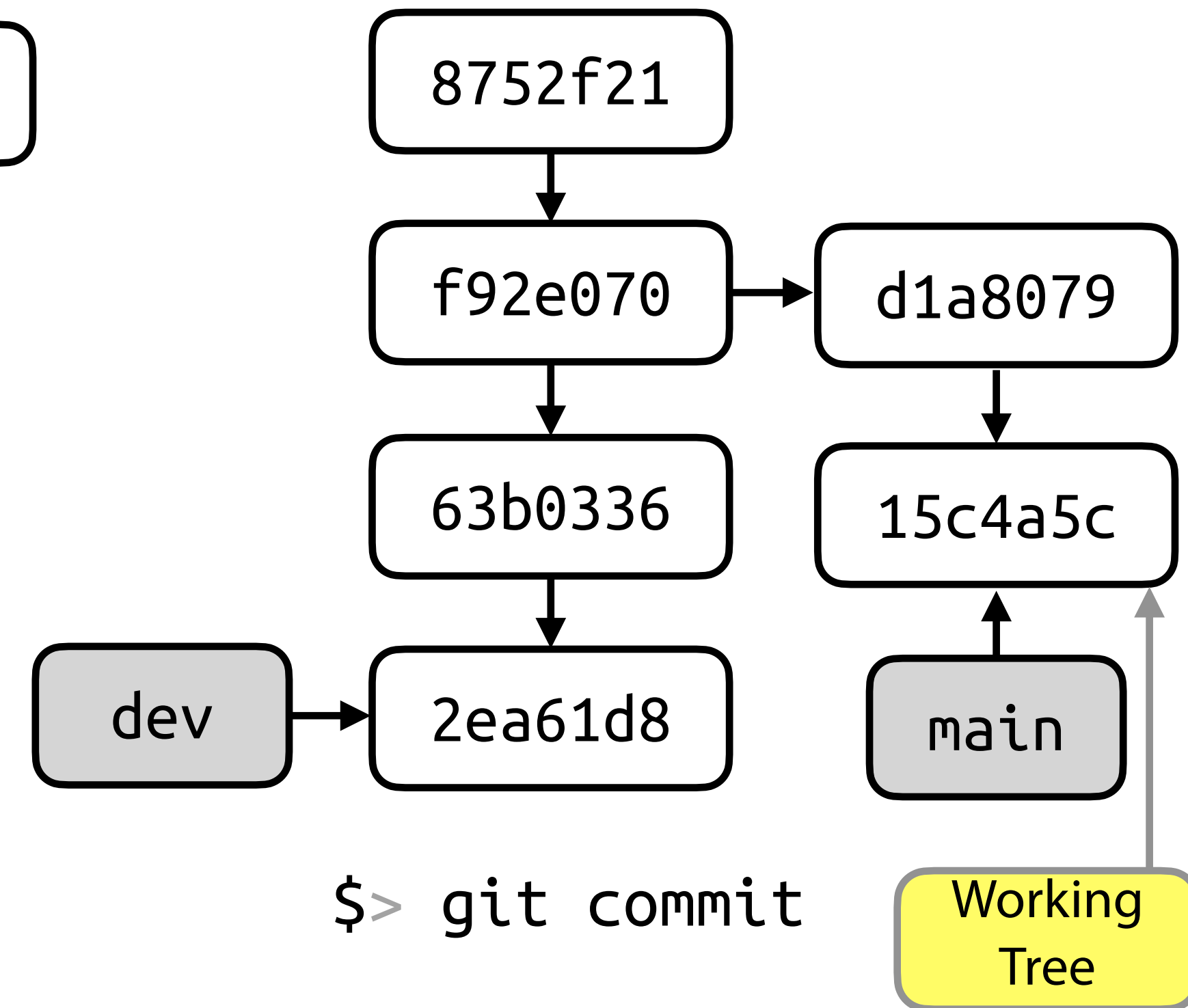


\$> git checkout main



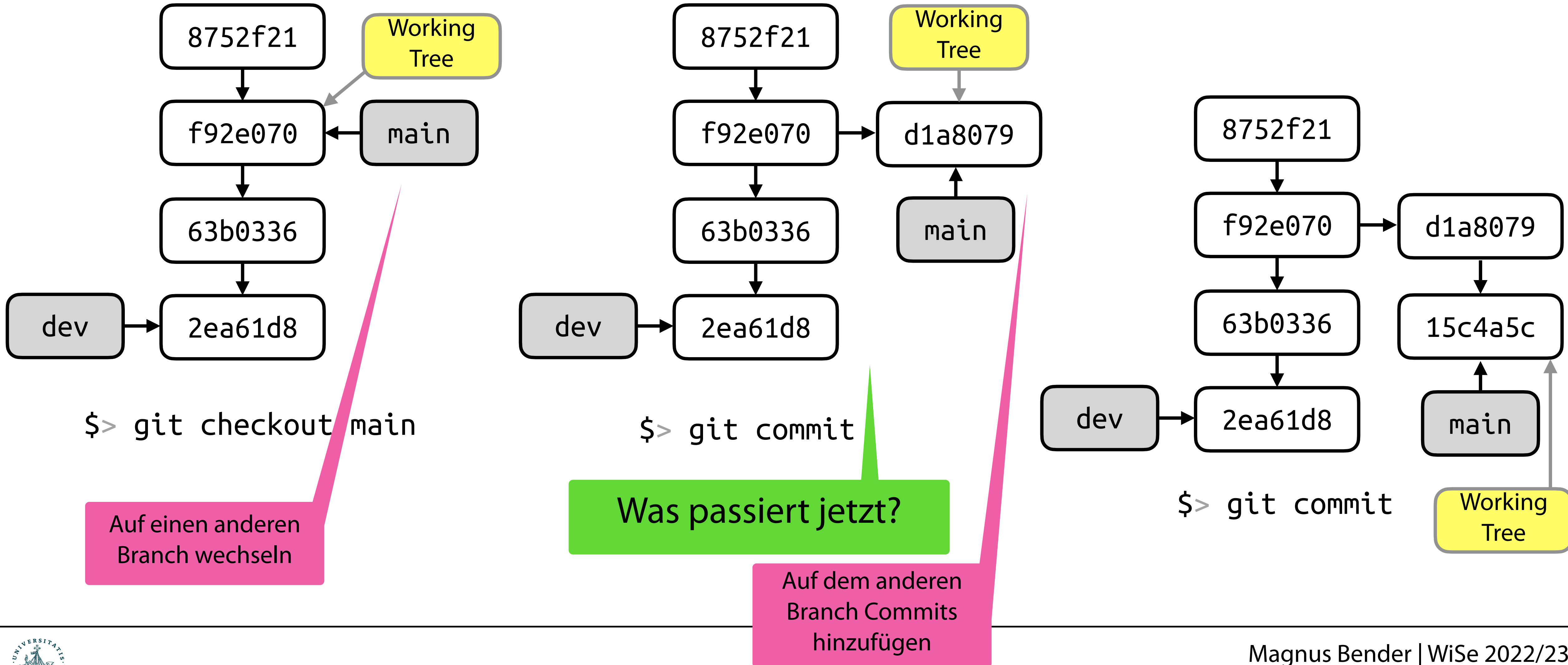
\$> git commit

Was passiert jetzt?



\$> git commit

Branches: Verzweigung



`$> git checkout main`

Auf einen anderen Branch wechseln

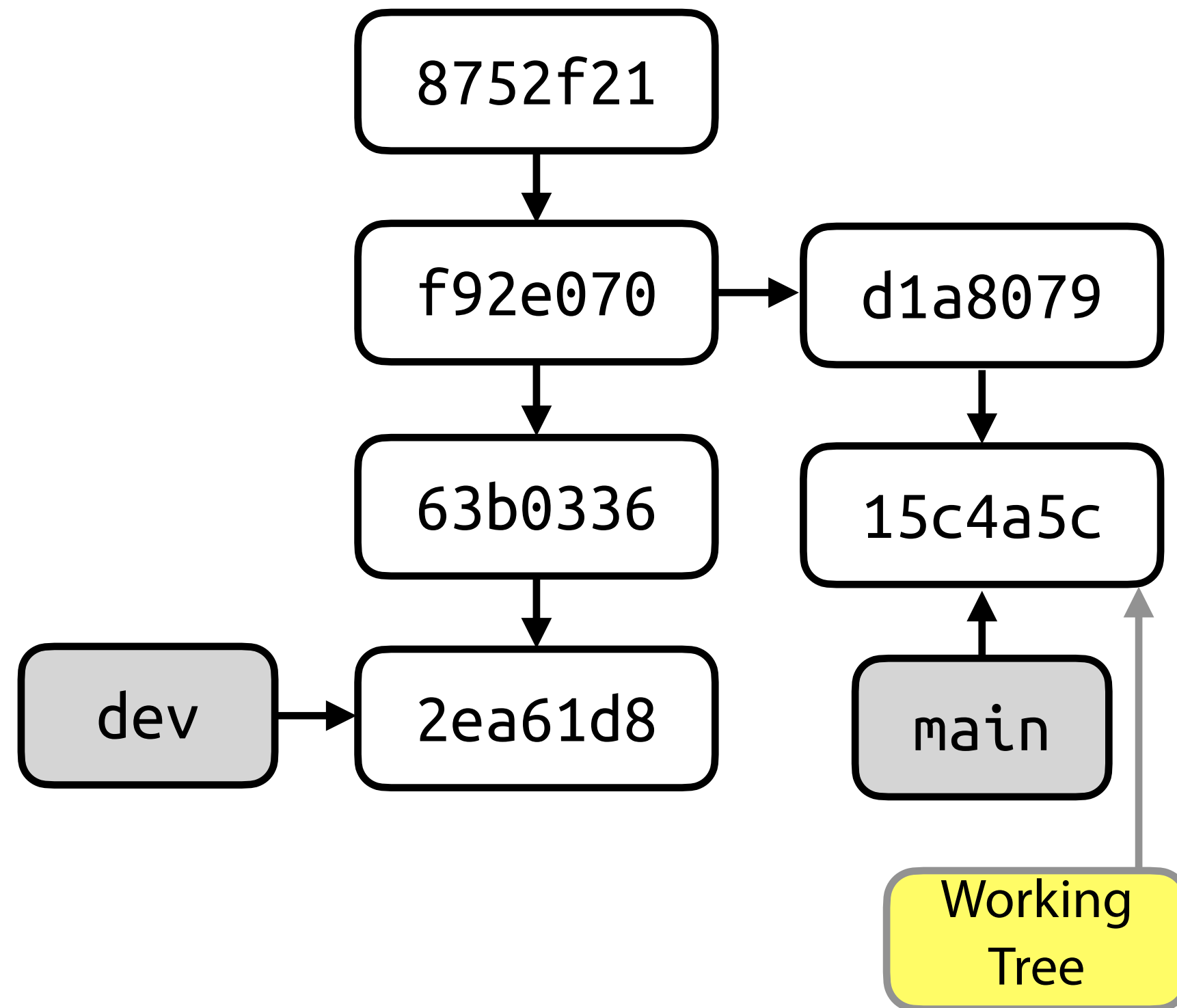
`$> git commit`

Was passiert jetzt?

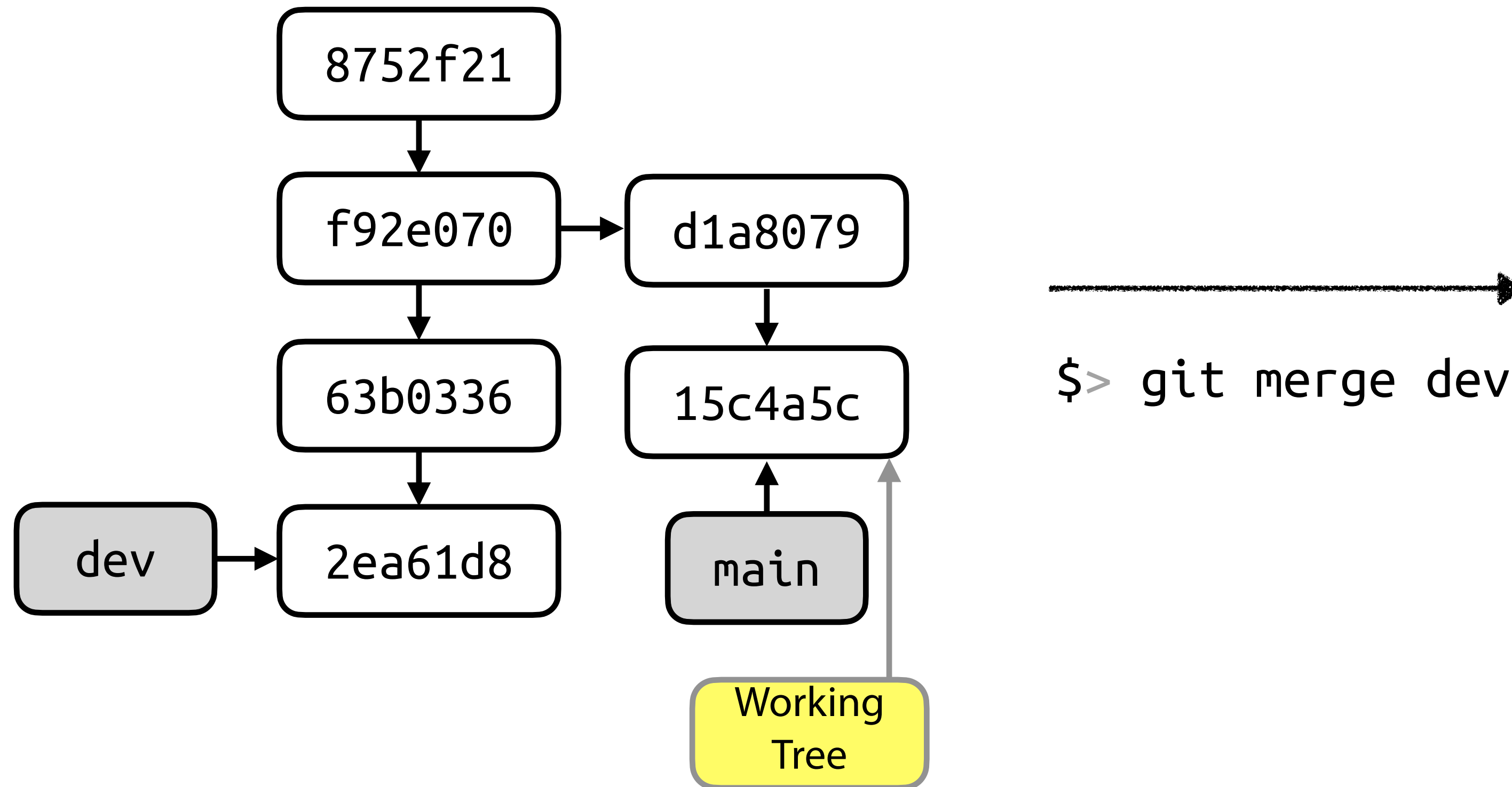
Auf dem anderen Branch Commits hinzufügen

`$> git commit`

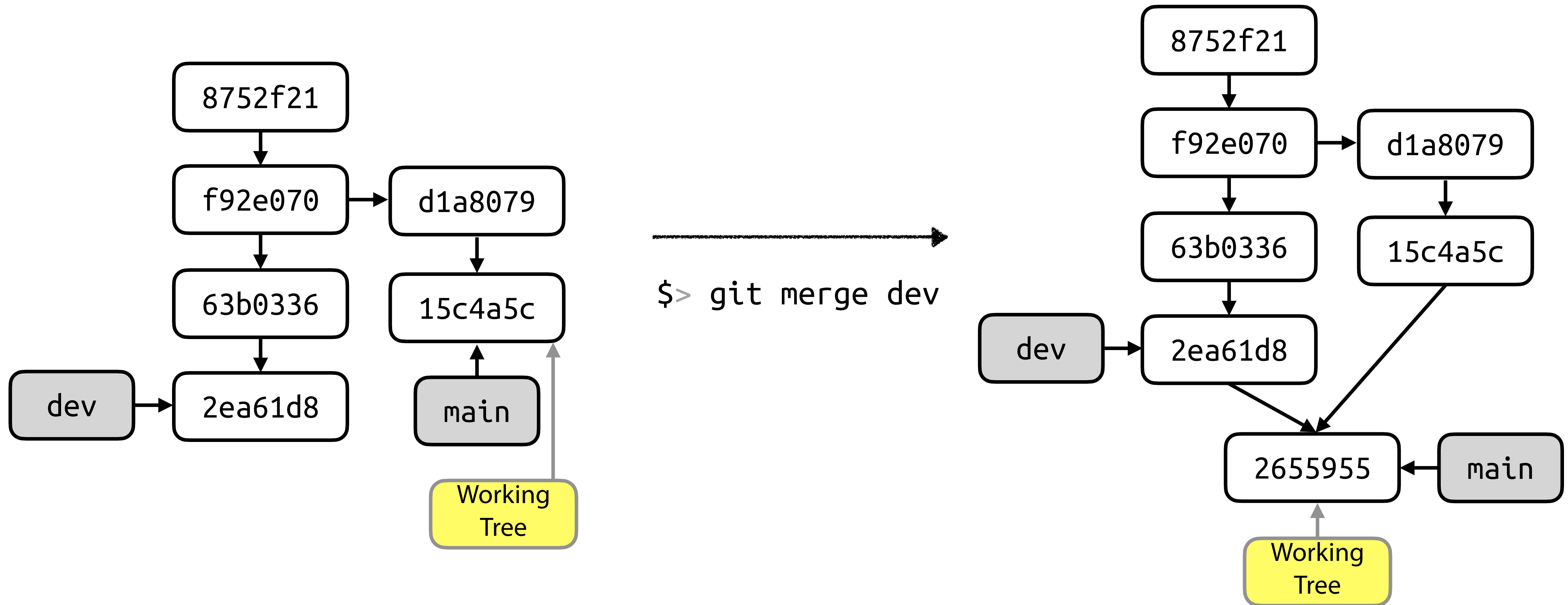
Branches: Zusammenführung



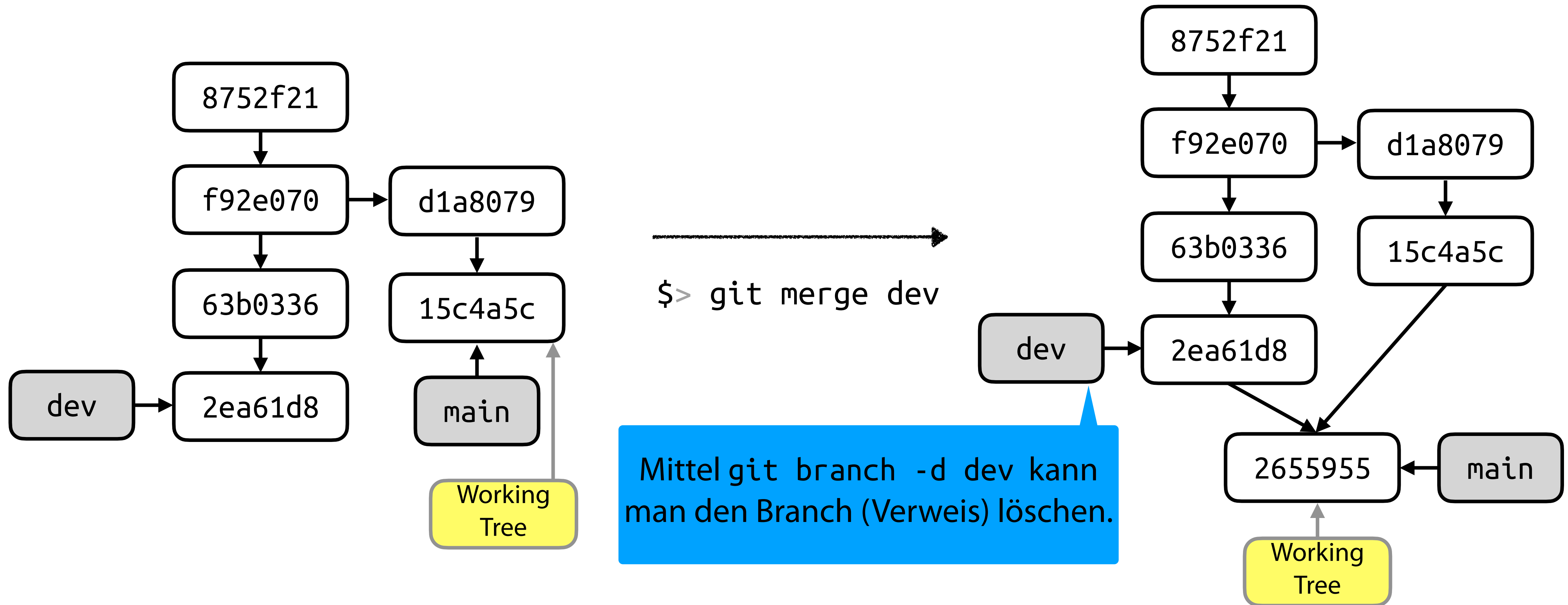
Branches: Zusammenführung



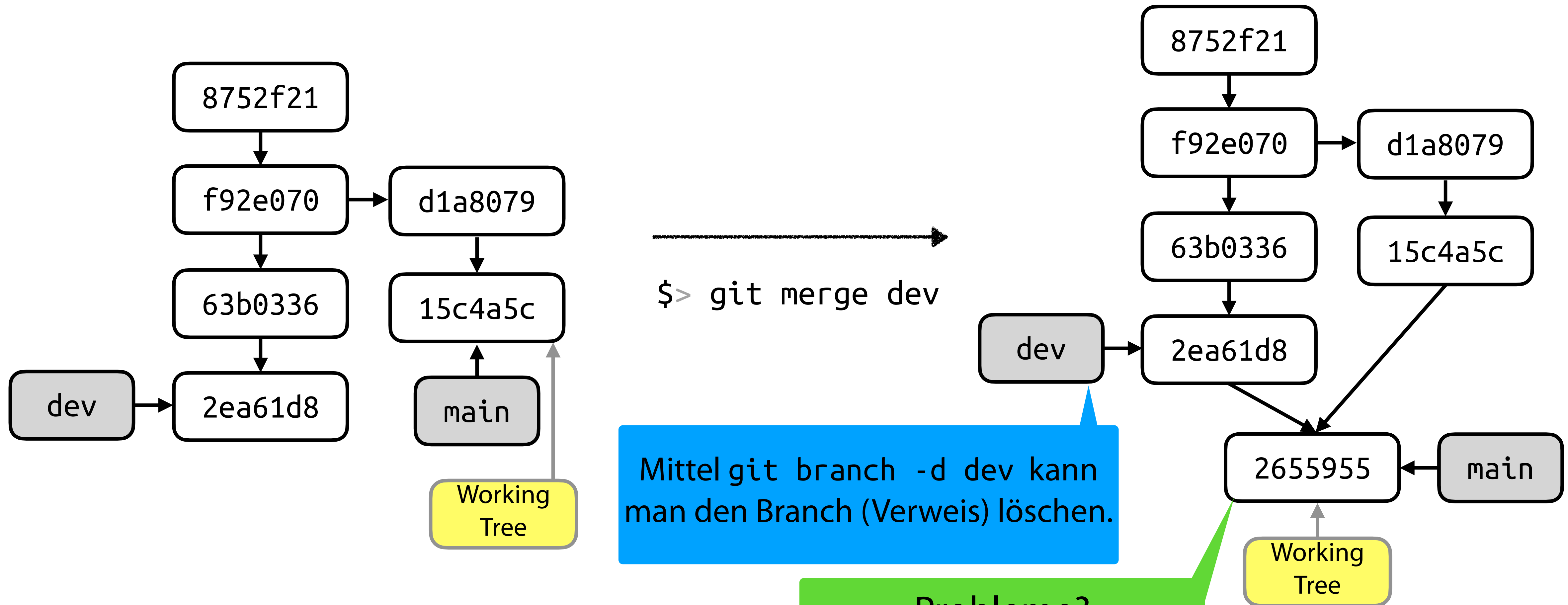
Branches: Zusammenführung



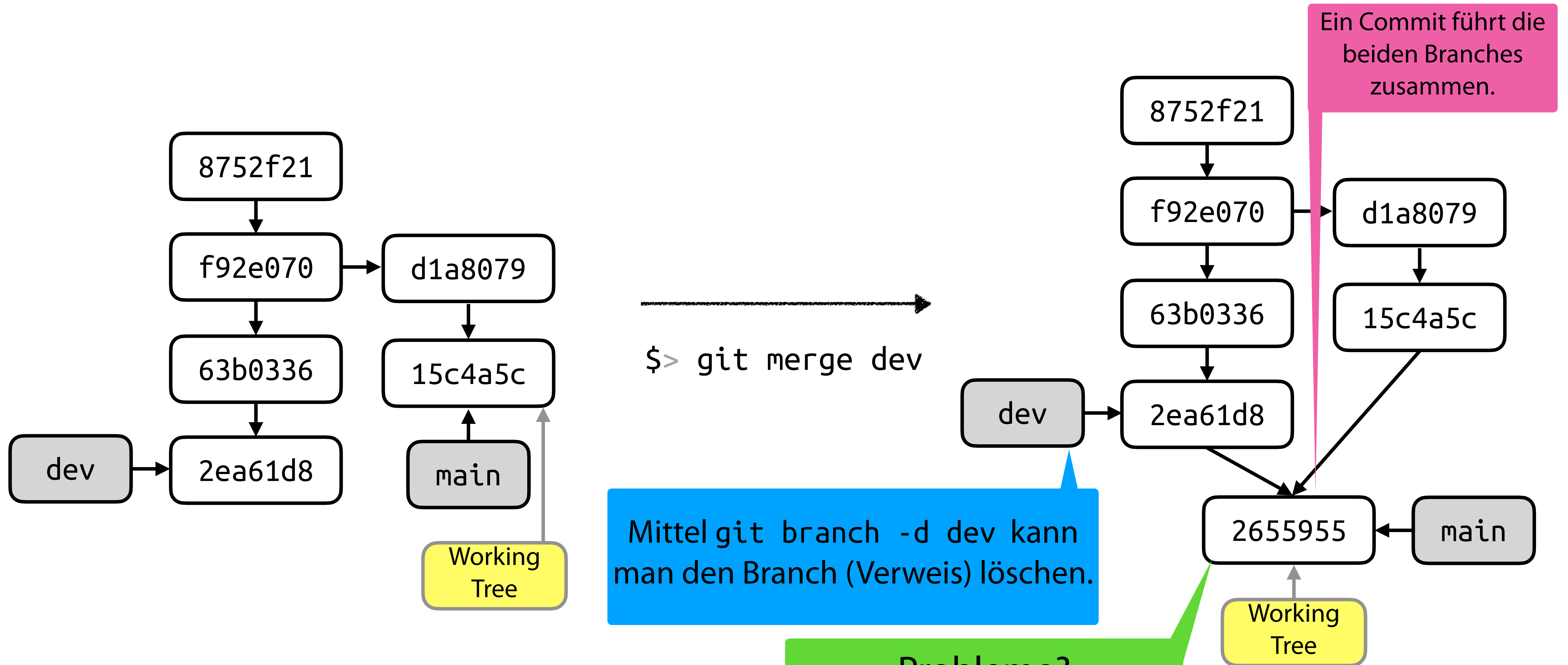
Branches: Zusammenführung



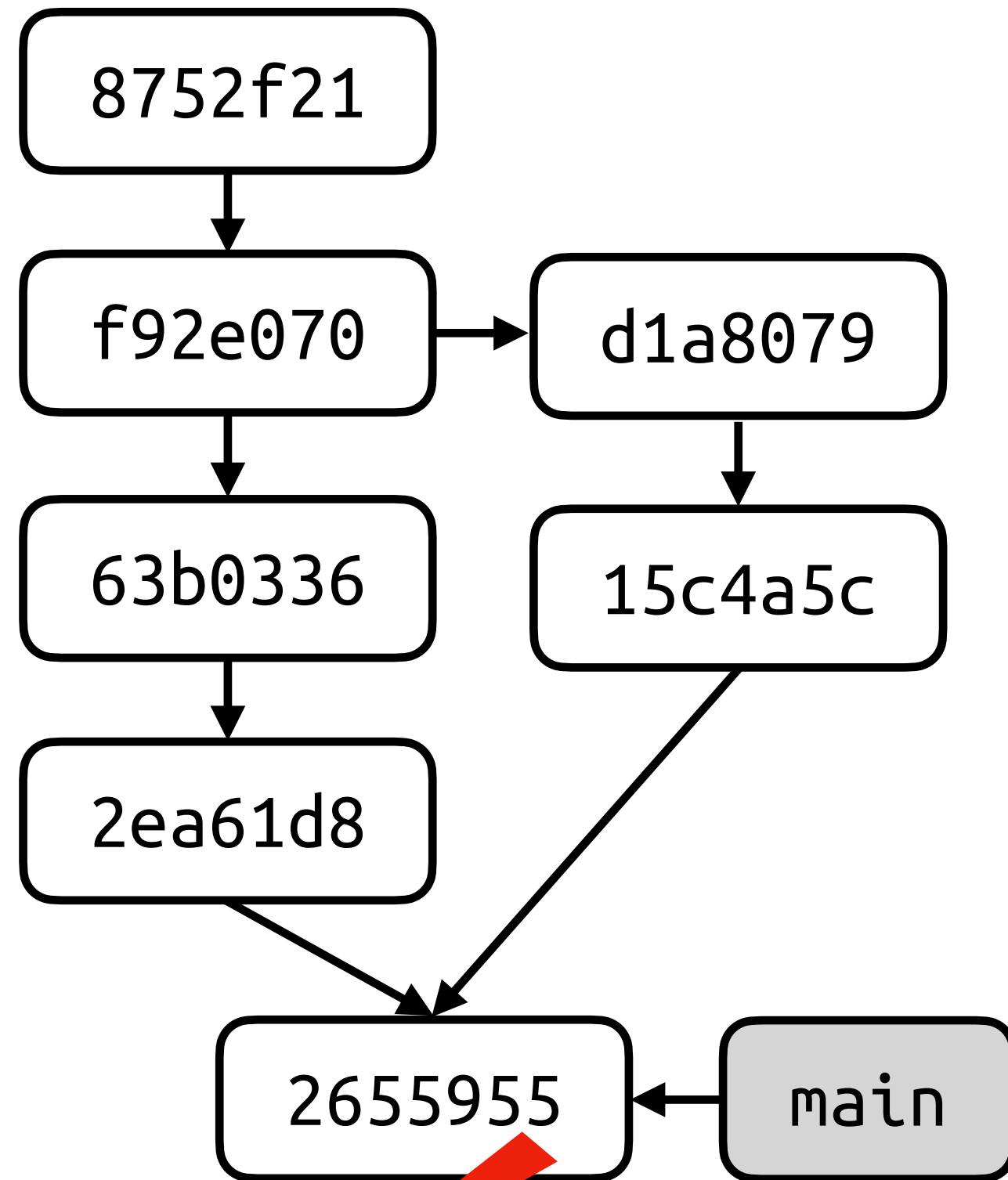
Branches: Zusammenführung



Branches: Zusammenführung

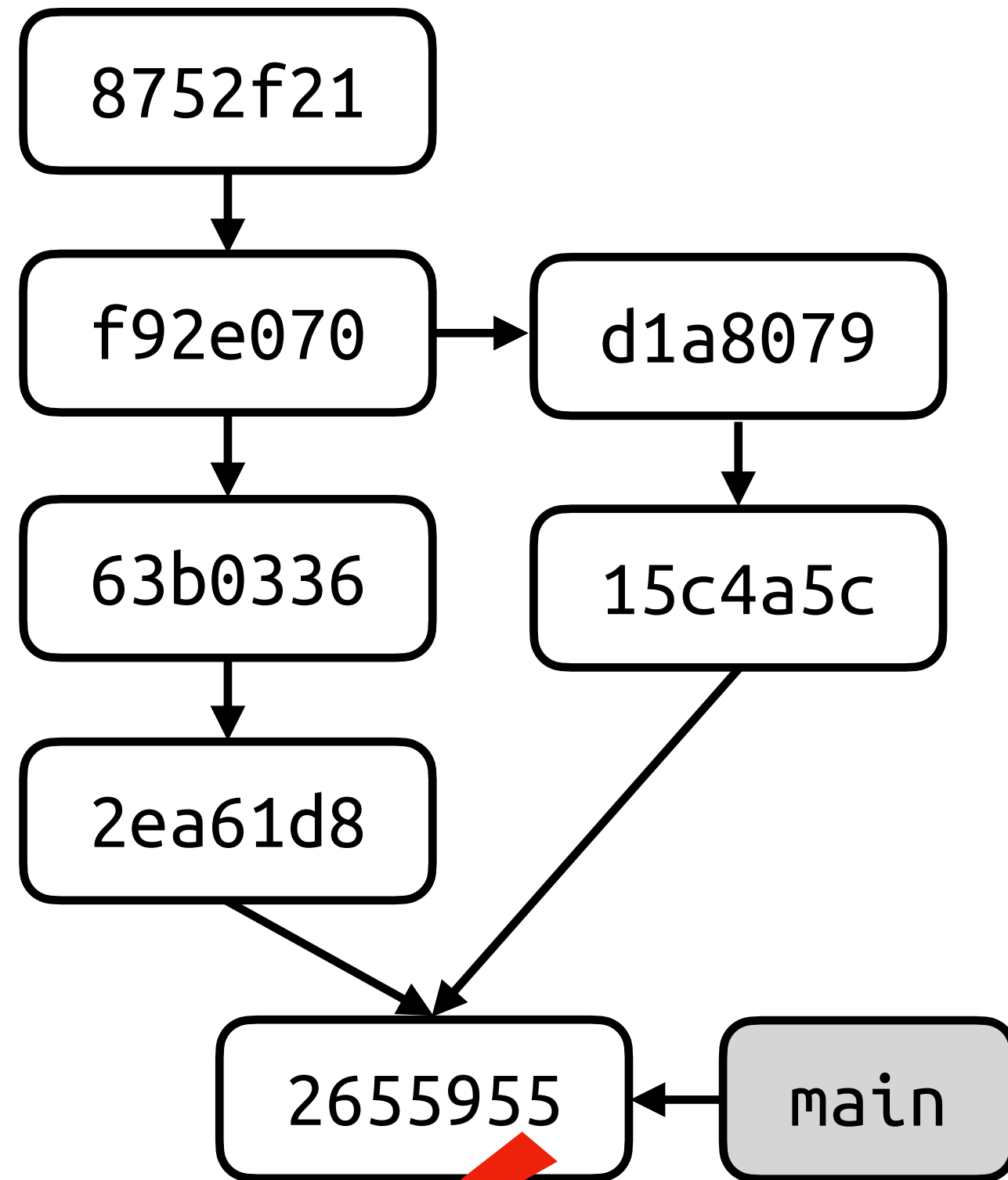


Zusammenführung: Konflikte



Merge-Konflikte treten auf, falls dieselbe Zeile der selben Datei in beiden Branches bearbeitet wurde.

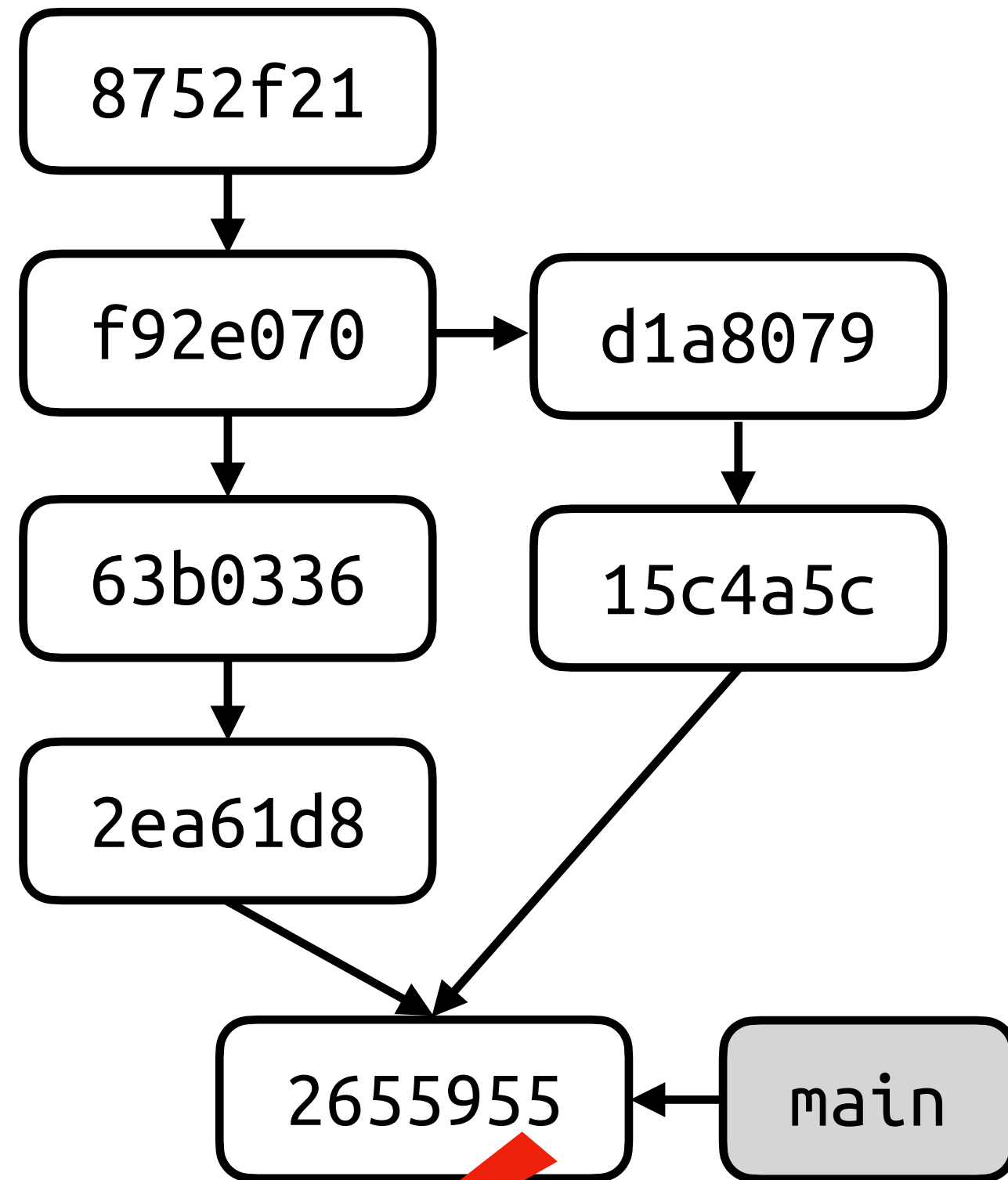
Zusammenführung: Konflikte



```
$> git merge dev  
# Fehlermeldung
```

Merge-Konflikte treten auf, falls dieselbe Zeile der selben Datei in beiden Branches bearbeitet wurde.

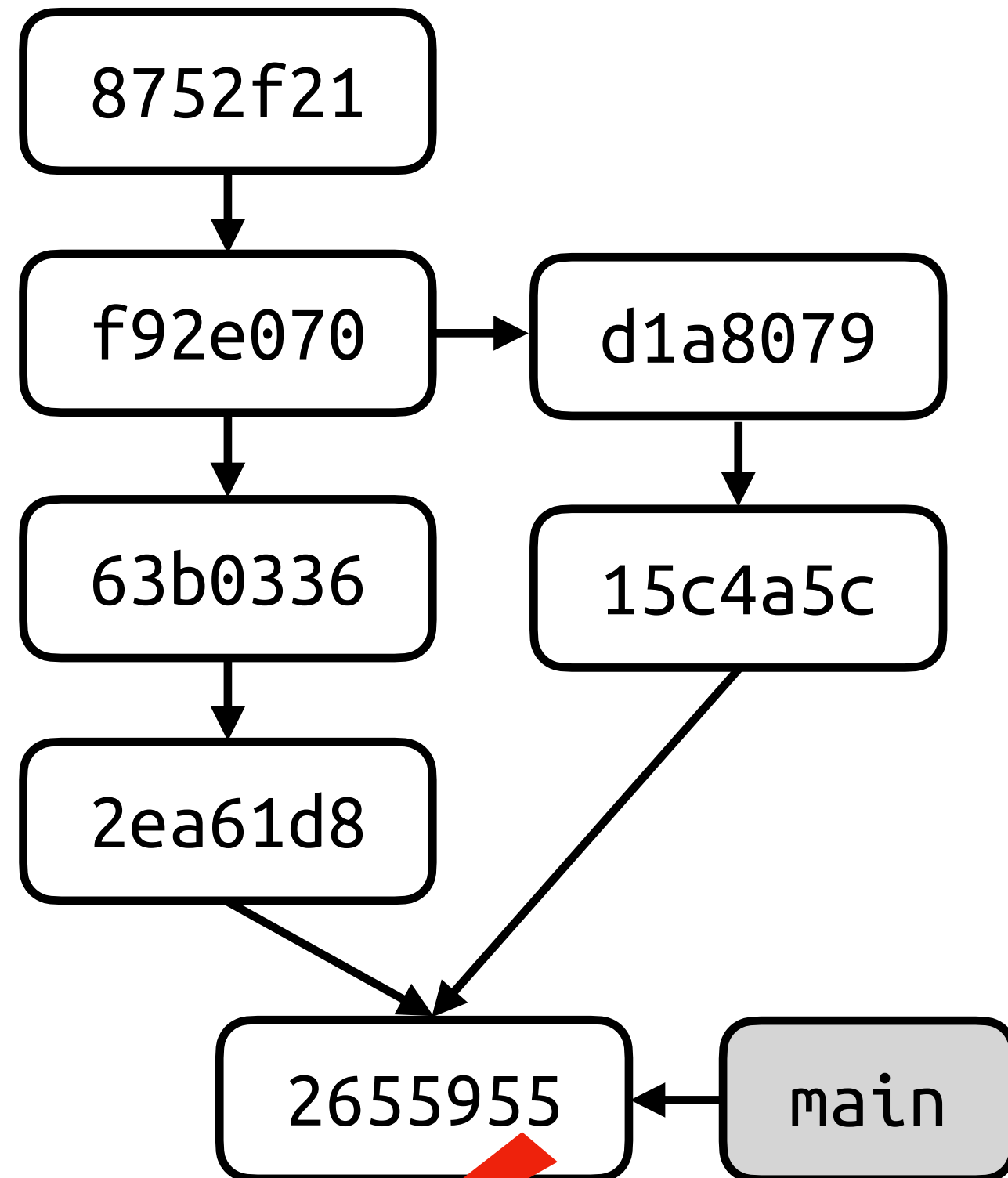
Zusammenführung: Konflikte



```
$> git merge dev  
# Fehlermeldung  
$> git status  
# Problematische Dateien werden angezeigt
```

Merge-Konflikte treten auf, falls dieselbe Zeile der selben Datei in beiden Branches bearbeitet wurde.

Zusammenführung: Konflikte

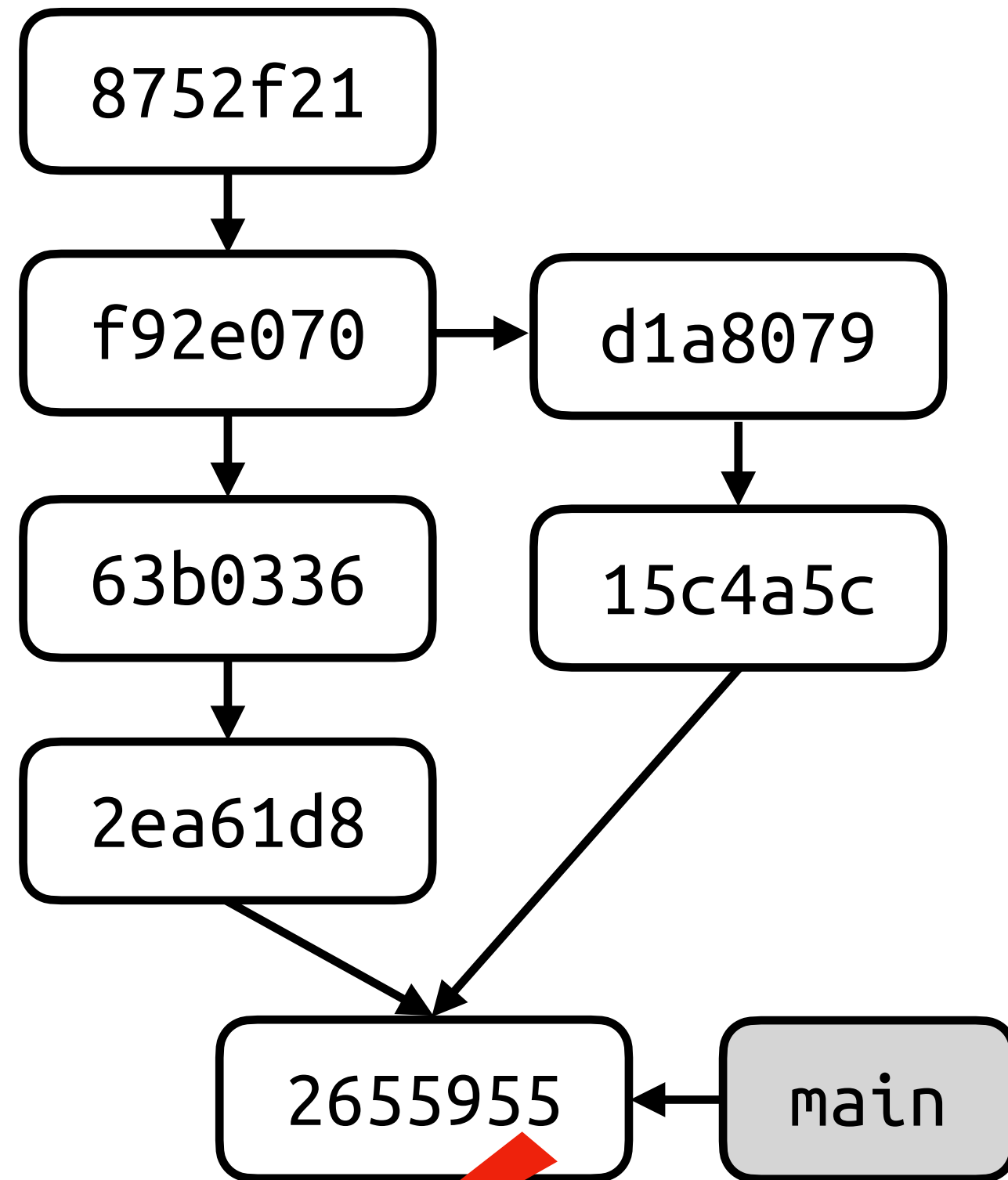


```
$> git merge dev
# Fehlermeldung
$> git status
# Problematische Dateien werden angezeigt

$> vim DateiMitFehler.txt
# Konflikt in Datei beheben
```

Merge-Konflikte treten auf, falls dieselbe Zeile der selben Datei in beiden Branches bearbeitet wurde.

Zusammenführung: Konflikte

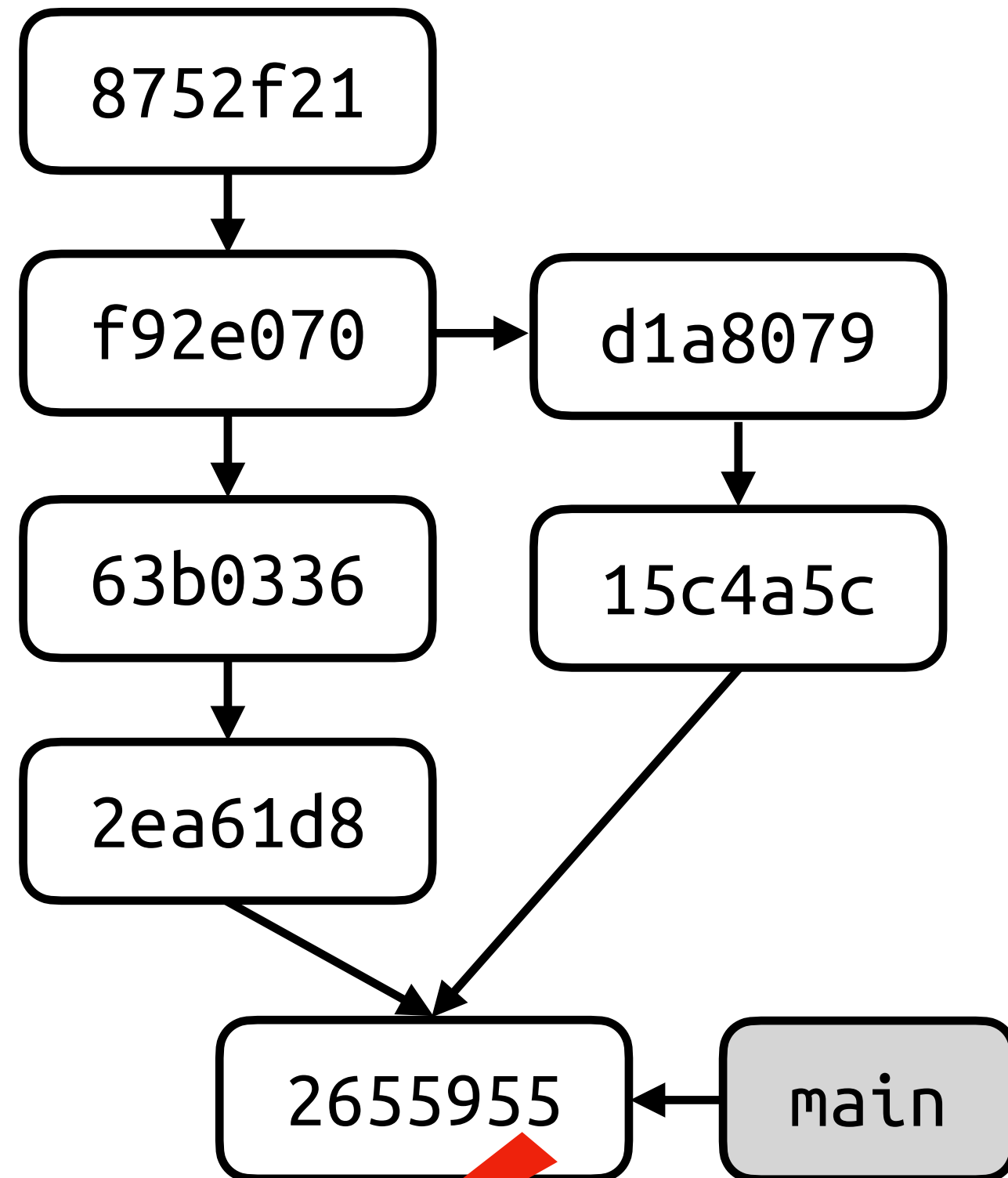


```
$> git merge dev
# Fehlermeldung
$> git status
# Problematische Dateien werden angezeigt

$> vim DateiMitFehler.txt
# Konflikt in Datei beheben
$> ...
```

Merge-Konflikte treten auf, falls dieselbe Zeile der selben Datei in beiden Branches bearbeitet wurde.

Zusammenführung: Konflikte



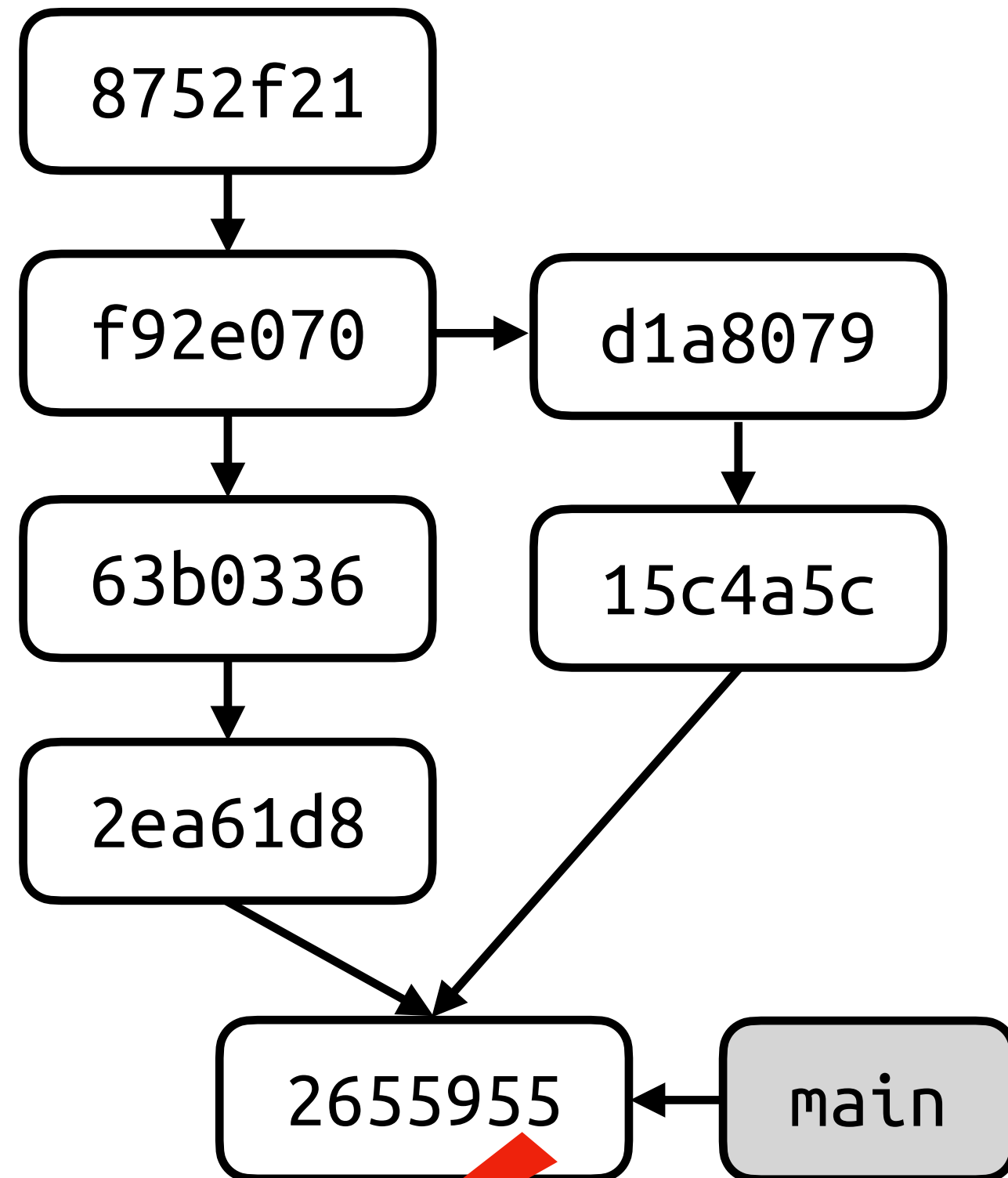
Merge-Konflikte treten auf, falls dieselbe Zeile der selben Datei in beiden Branches bearbeitet wurde.

```
$> git merge dev
# Fehlermeldung
$> git status
# Problematische Dateien werden angezeigt

$> vim DateiMitFehler.txt
# Konflikt in Datei beheben
$> ...

$> git add DateiMitFehler.txt
```


Zusammenführung: Konflikte



Merge-Konflikte treten auf, falls dieselbe Zeile der selben Datei in beiden Branches bearbeitet wurde.

```
$> git merge dev
# Fehlermeldung
$> git status
# Problematische Dateien werden angezeigt

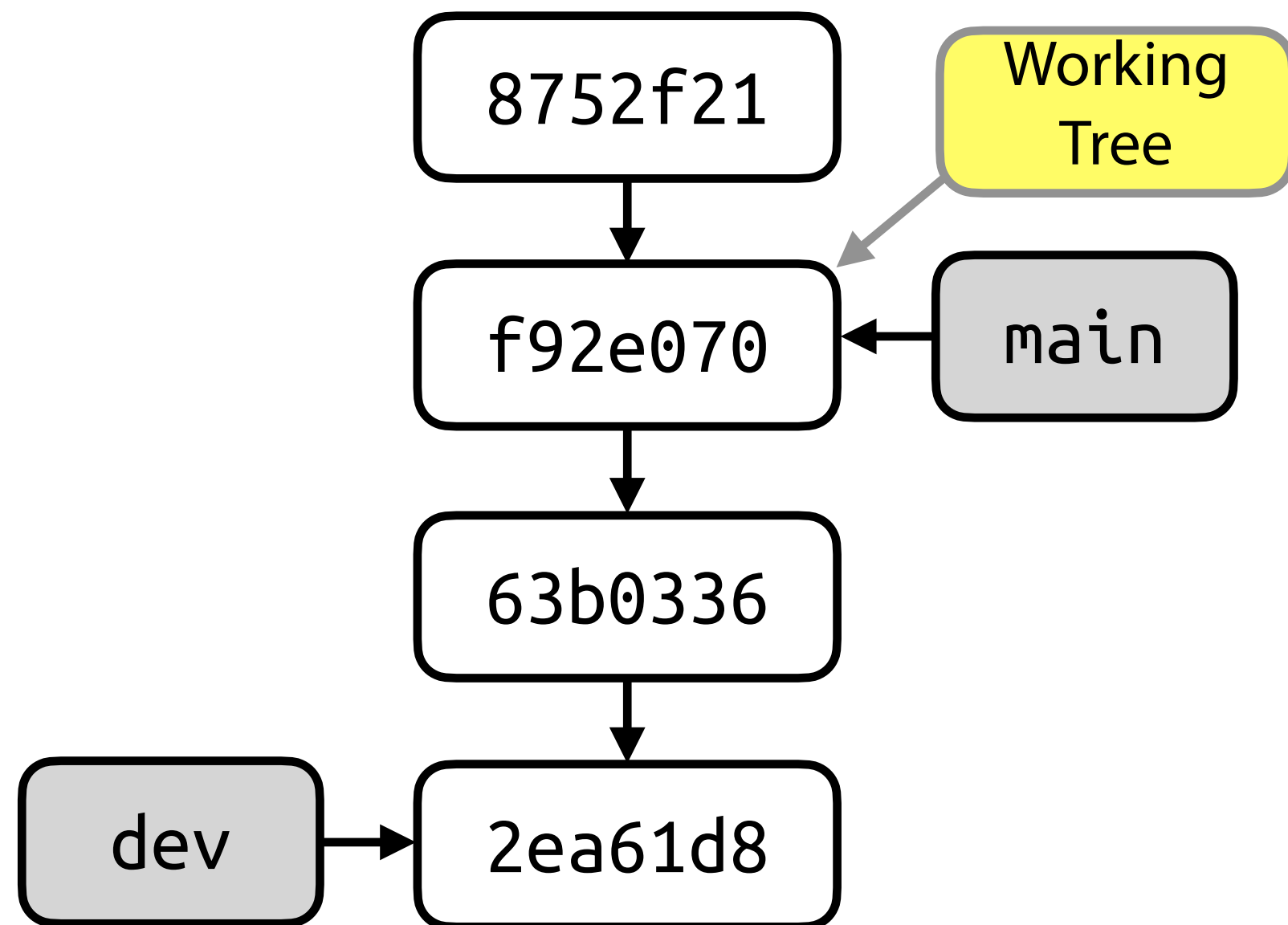
$> vim DateiMitFehler.txt
# Konflikt in Datei beheben
$> ...

$> git add DateiMitFehler.txt

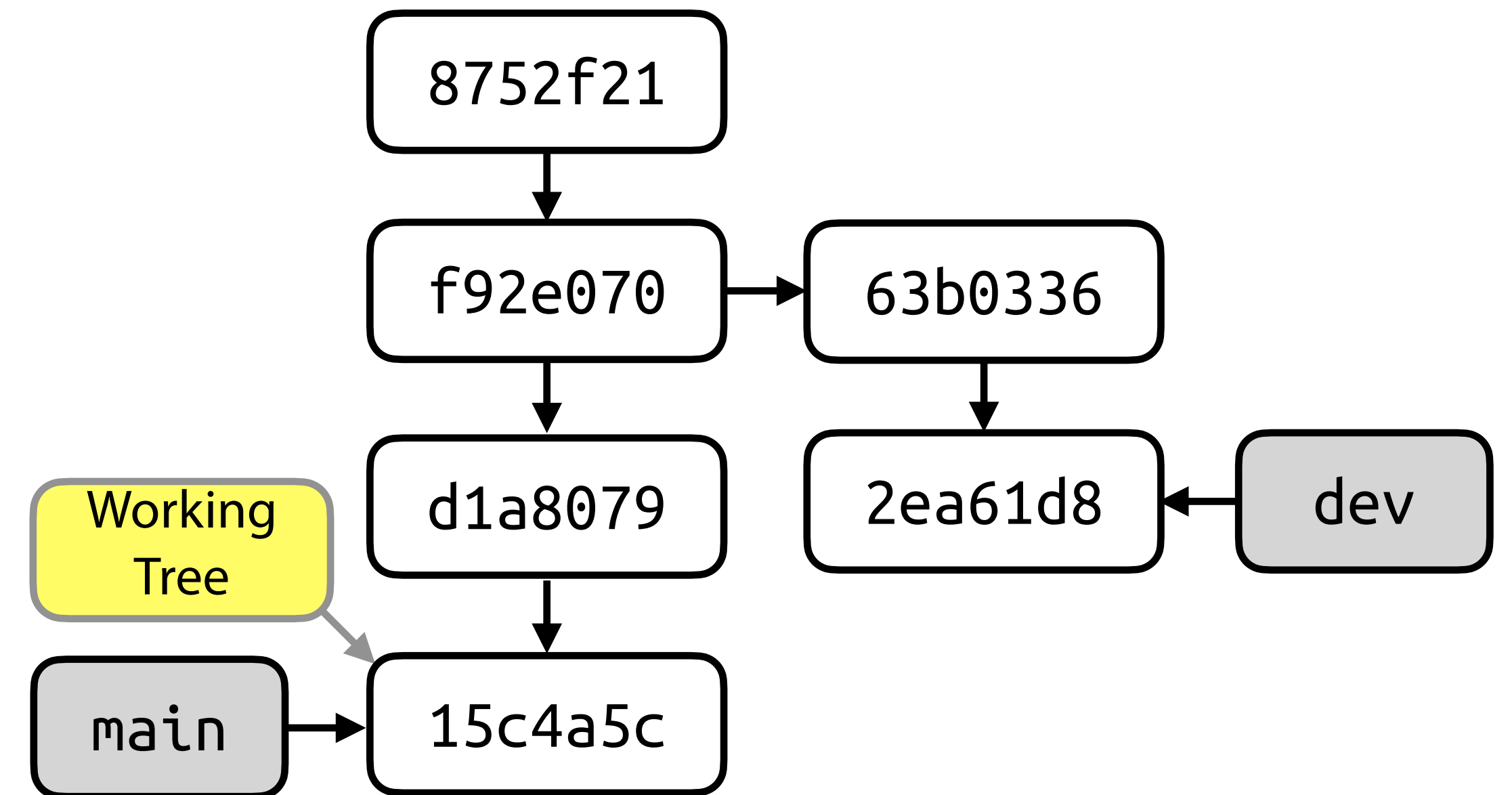
$> git commit
```

Weitere Zusammenführungen

Fast-Forward

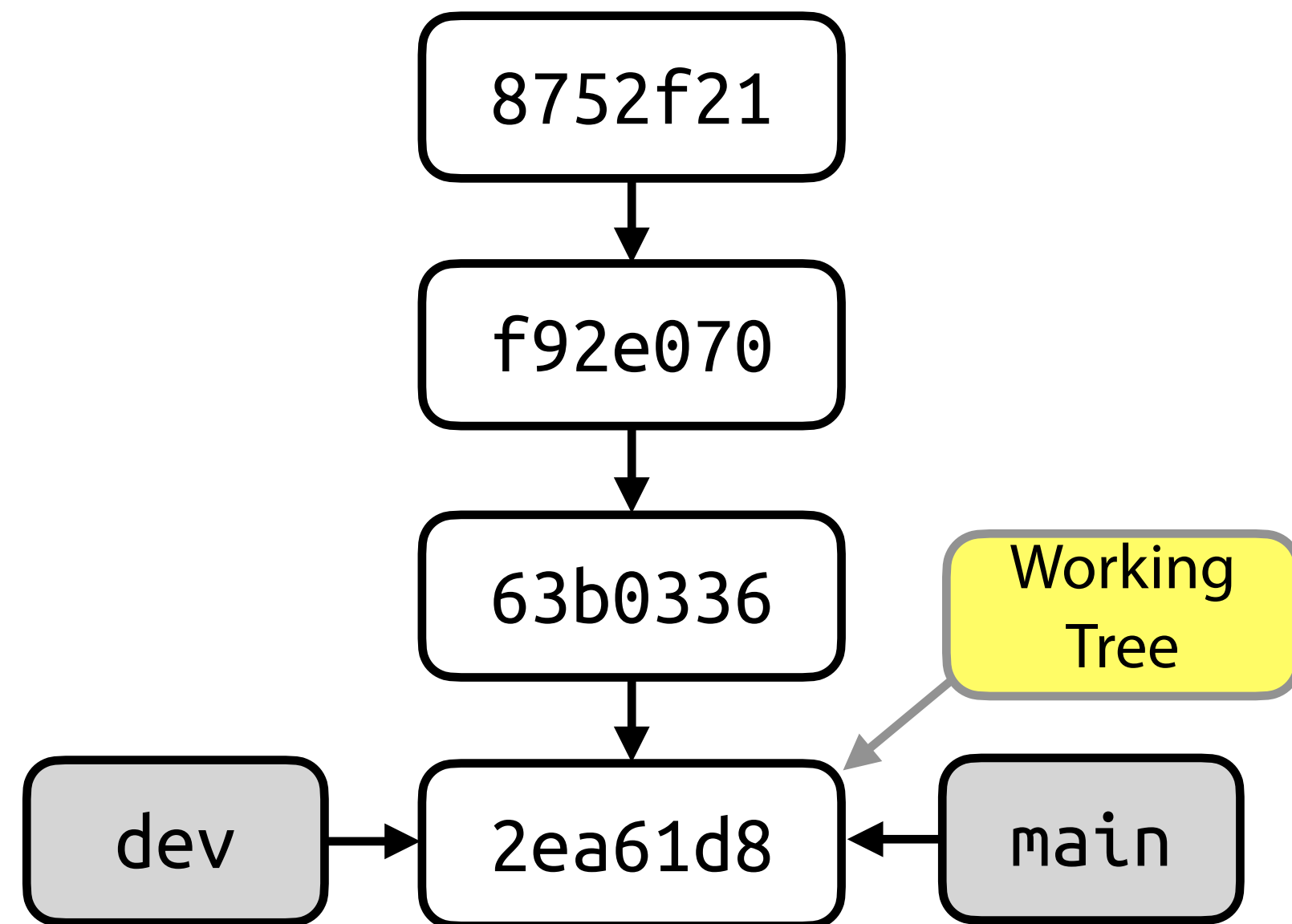


Rebase

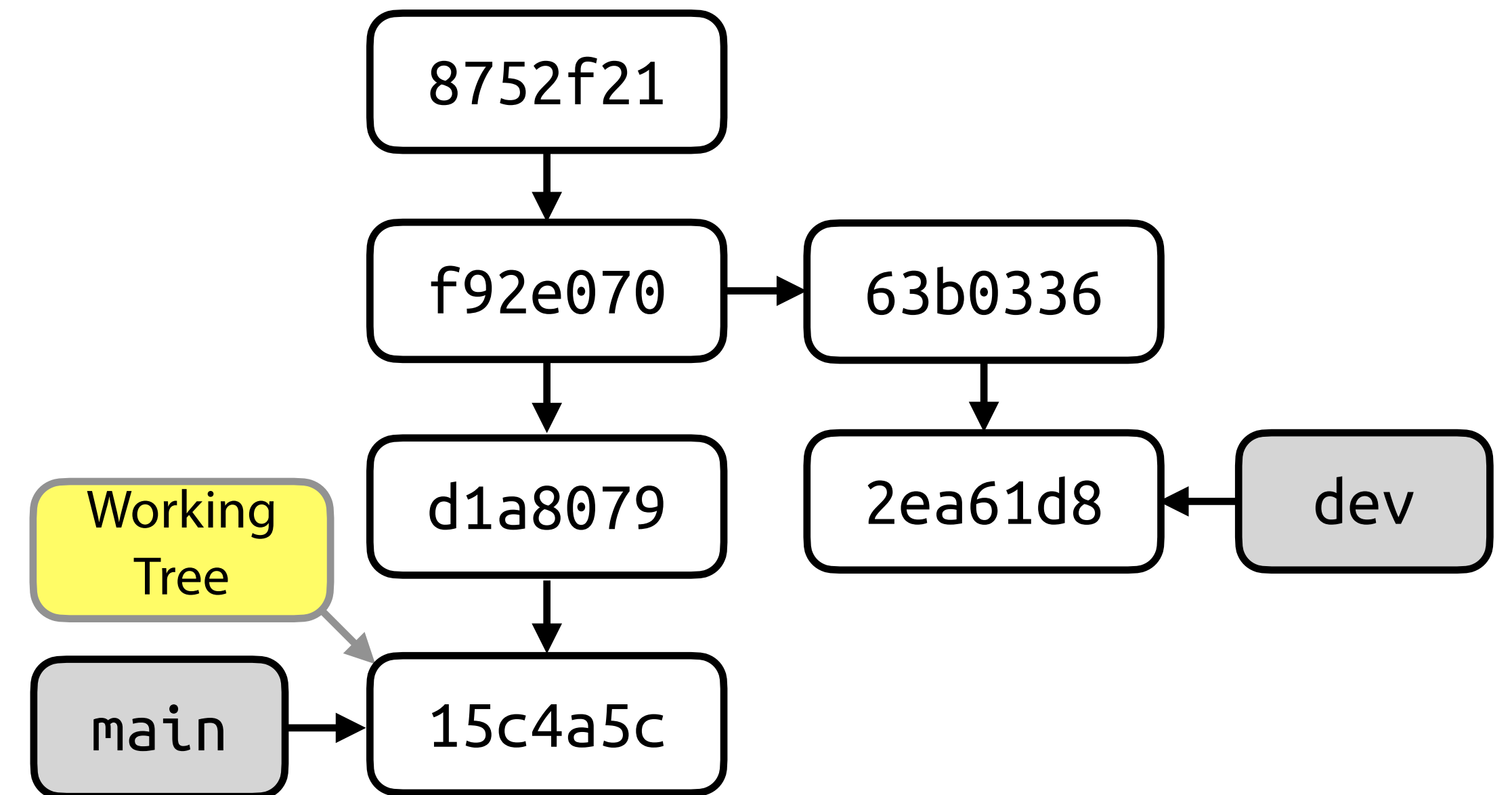


Weitere Zusammenführungen

Fast-Forward

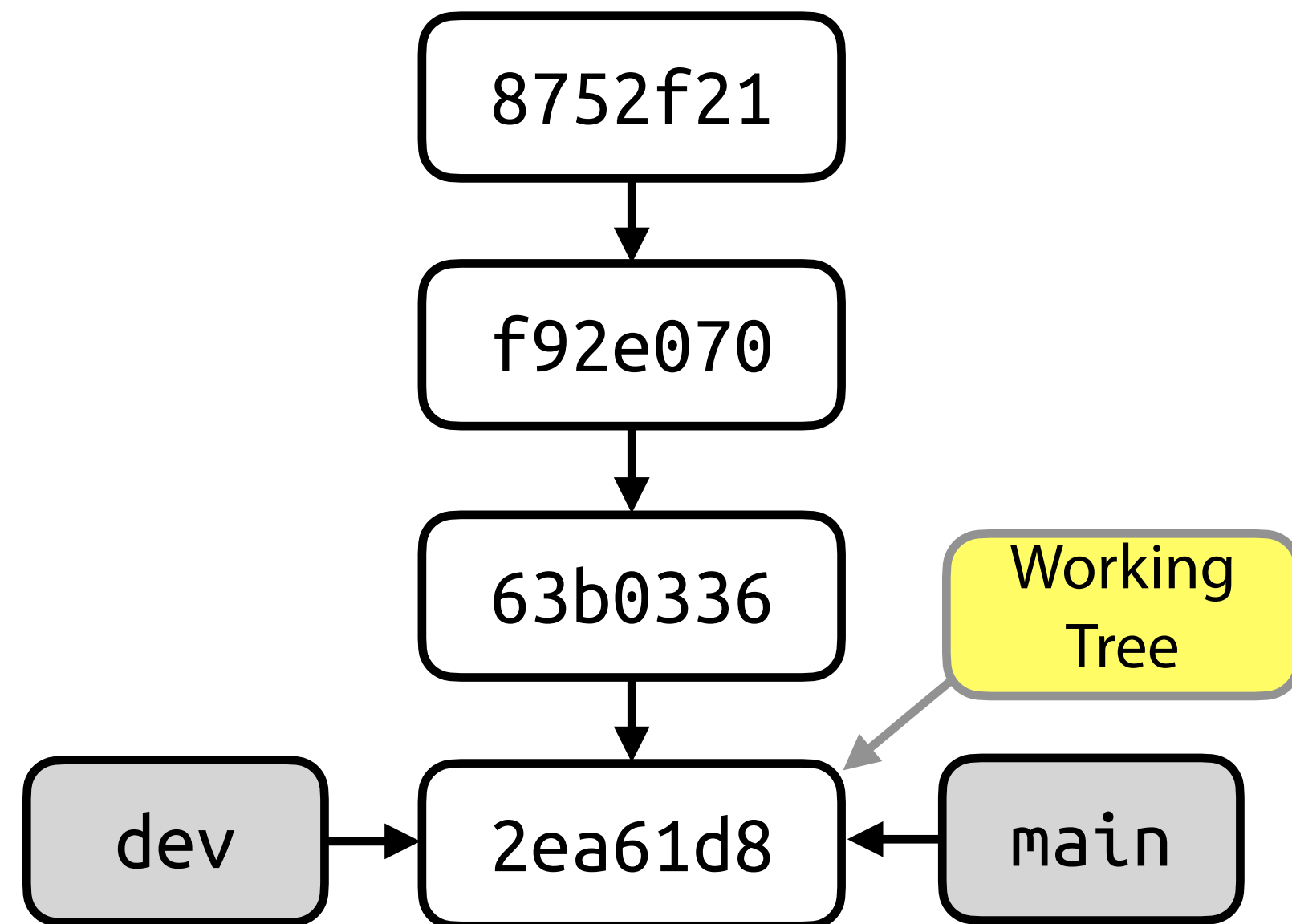


Rebase



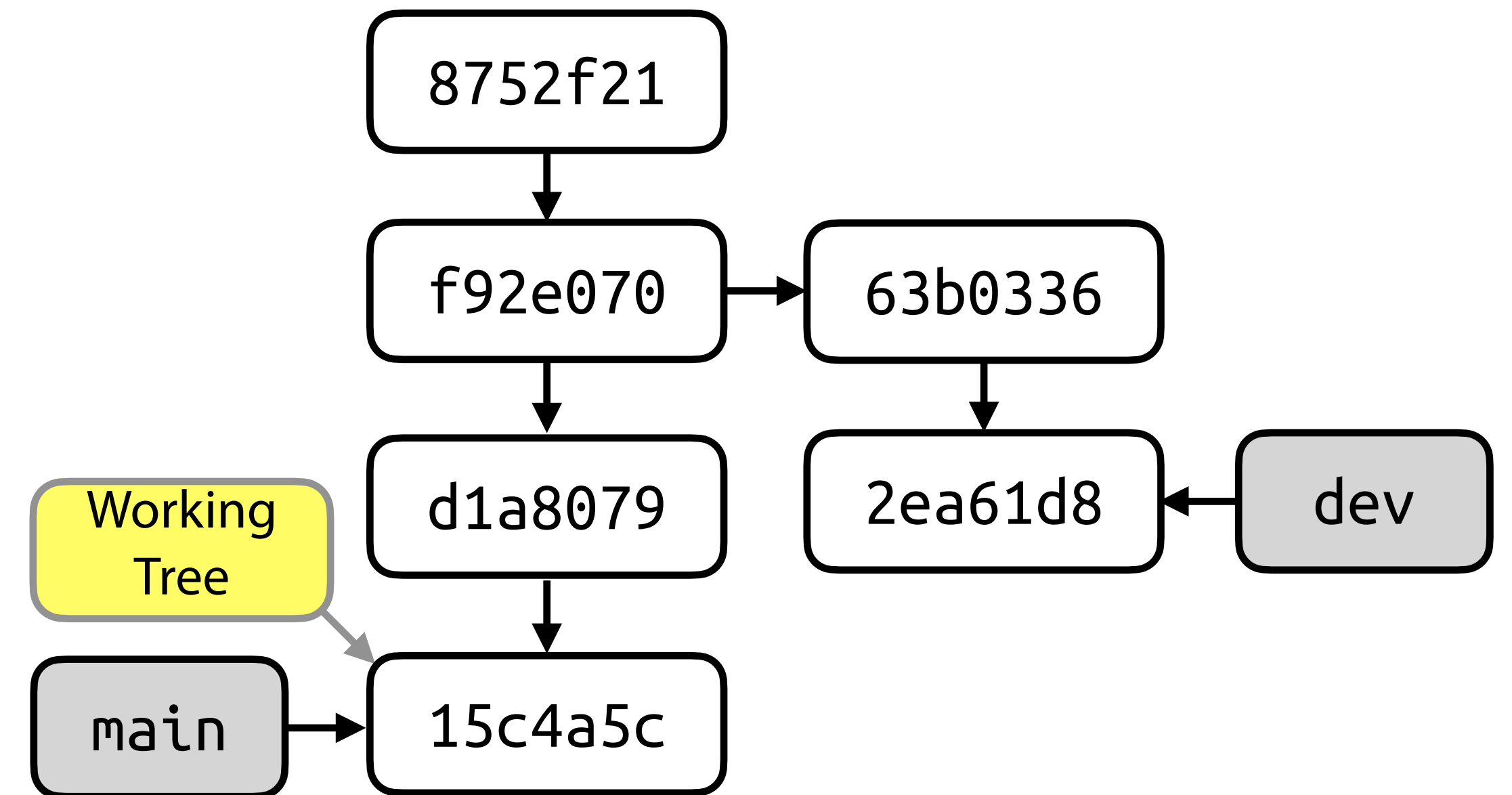
Weitere Zusammenführungen

Fast-Forward



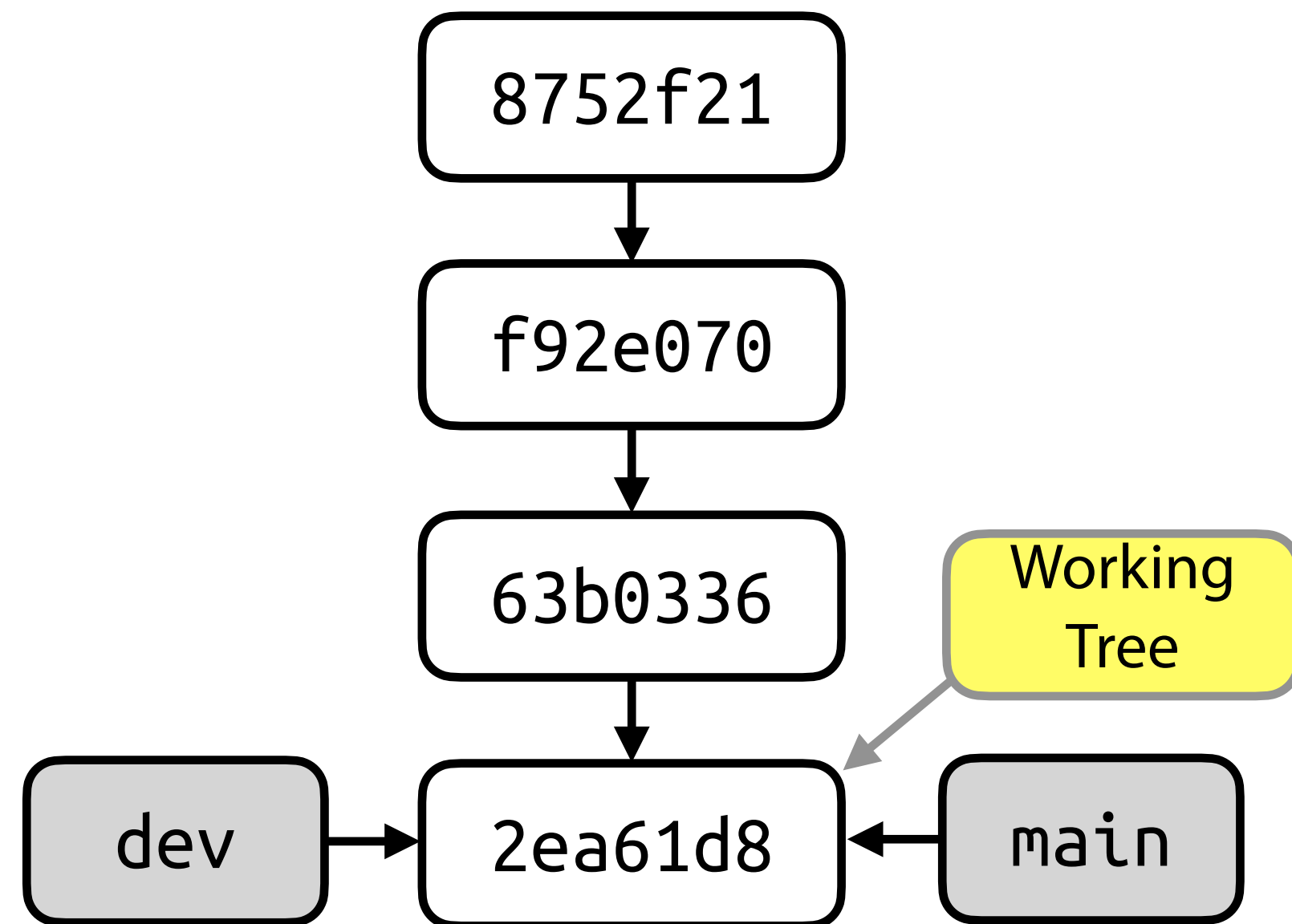
\$> git merge dev

Rebase



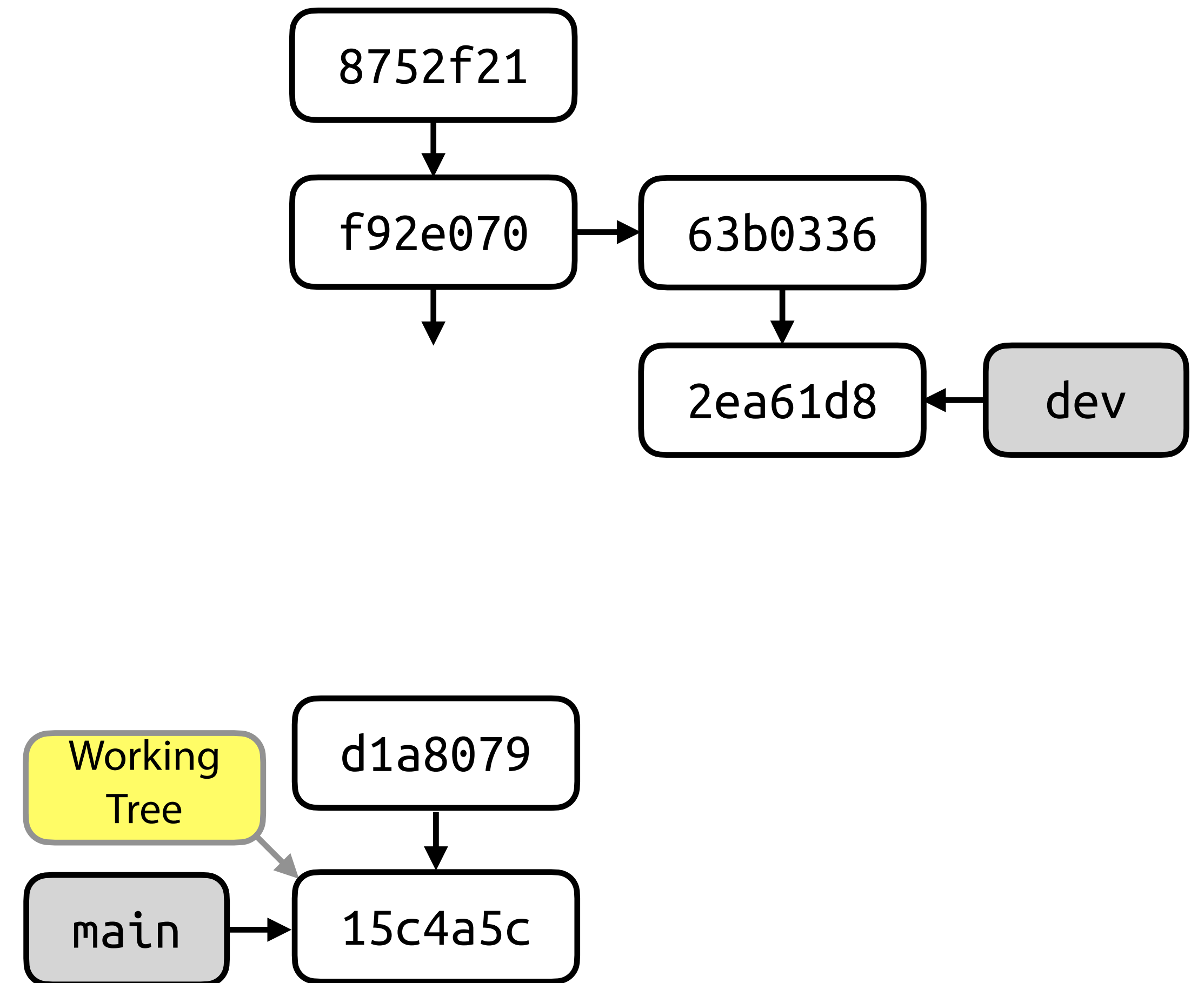
Weitere Zusammenführungen

Fast-Forward



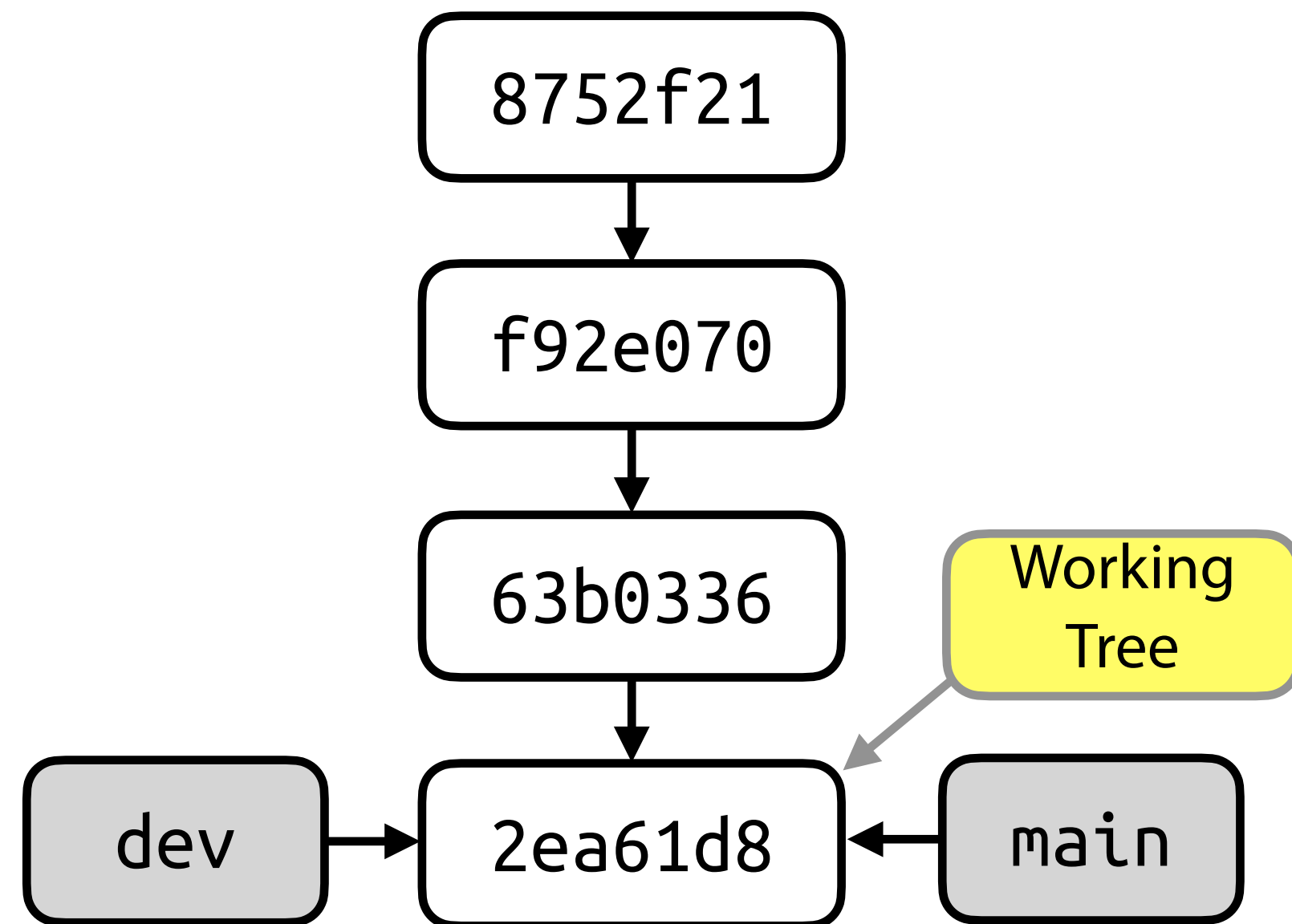
\$> git merge dev

Rebase



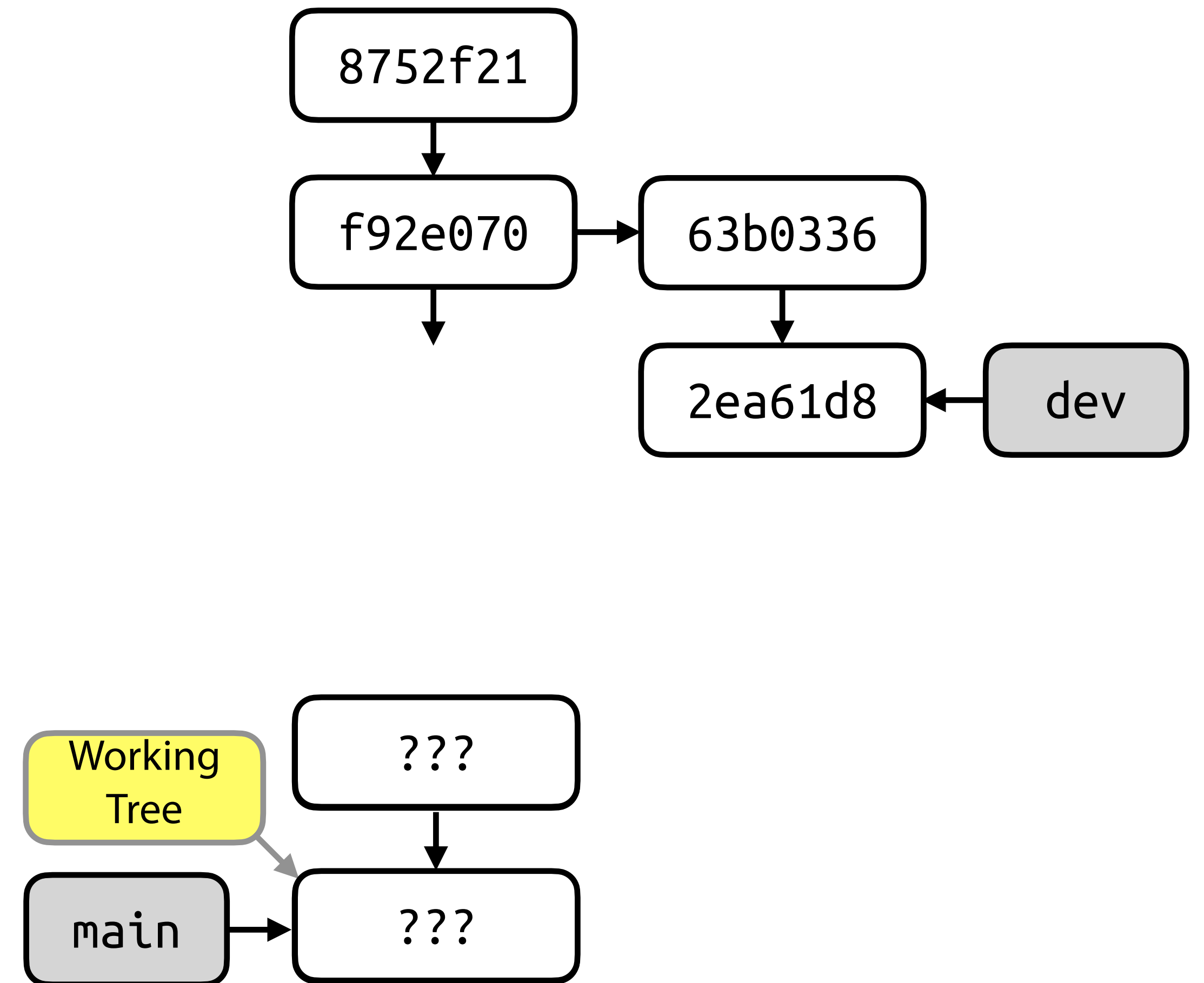
Weitere Zusammenführungen

Fast-Forward



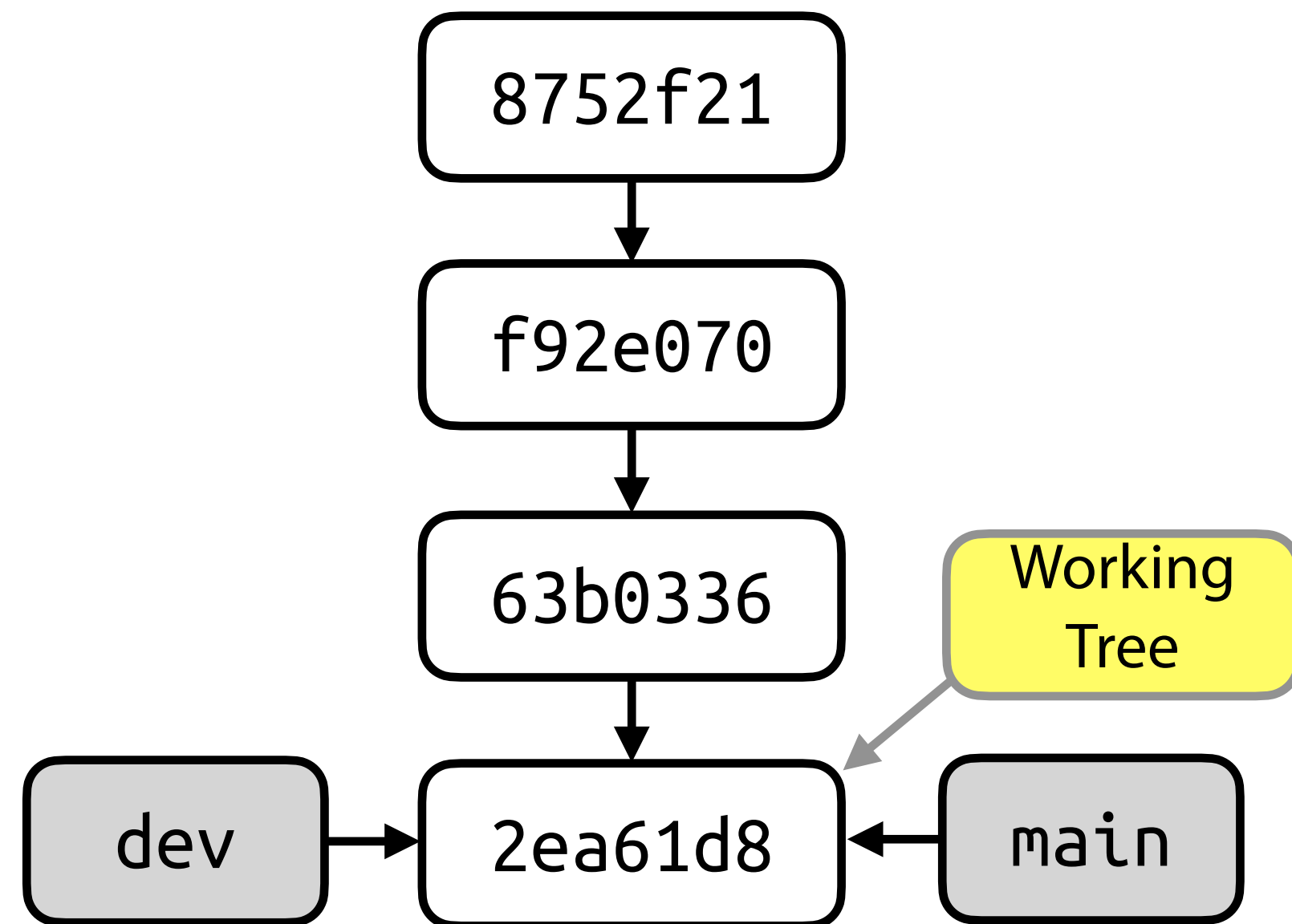
\$> git merge dev

Rebase



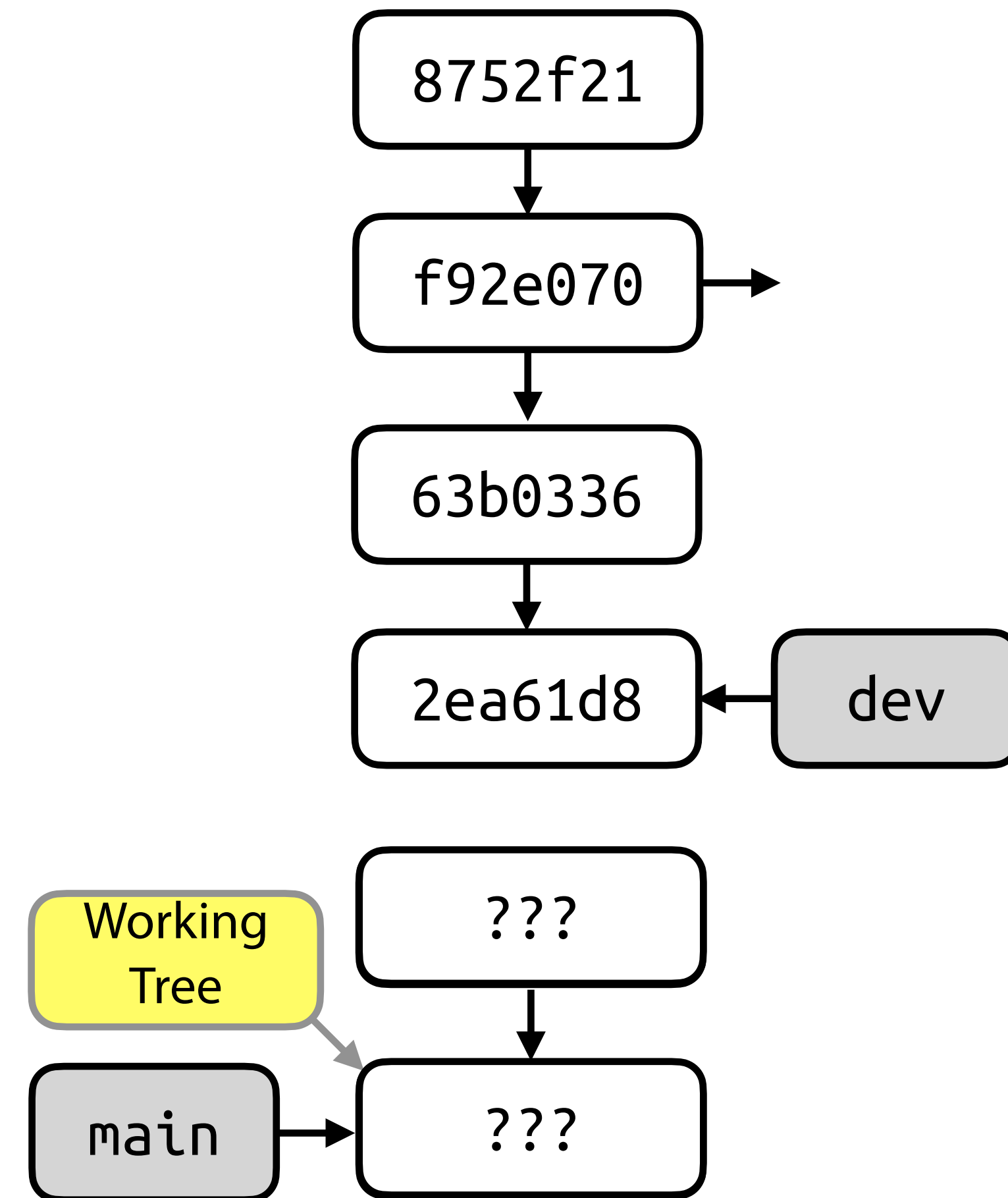
Weitere Zusammenführungen

Fast-Forward



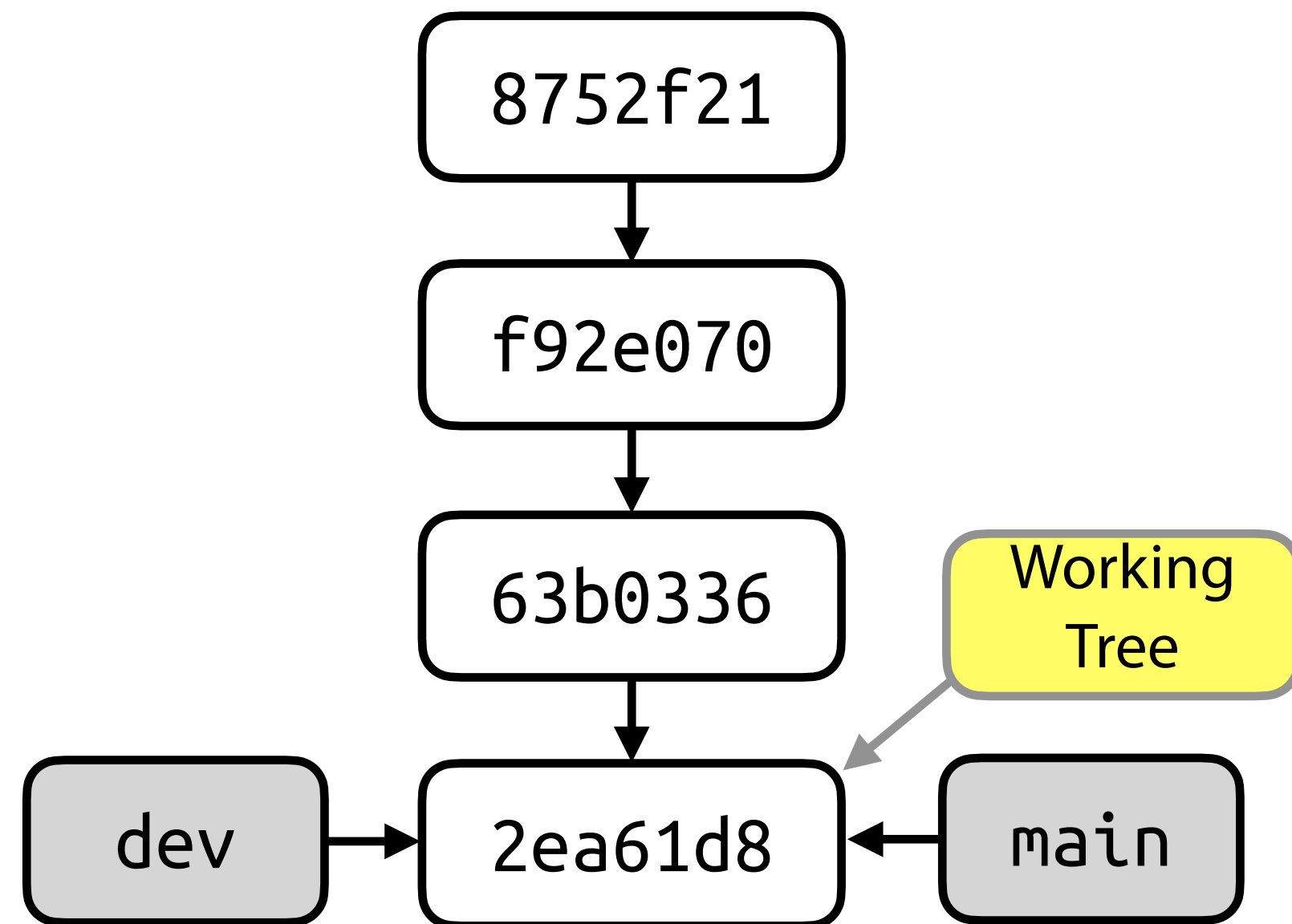
\$> git merge dev

Rebase



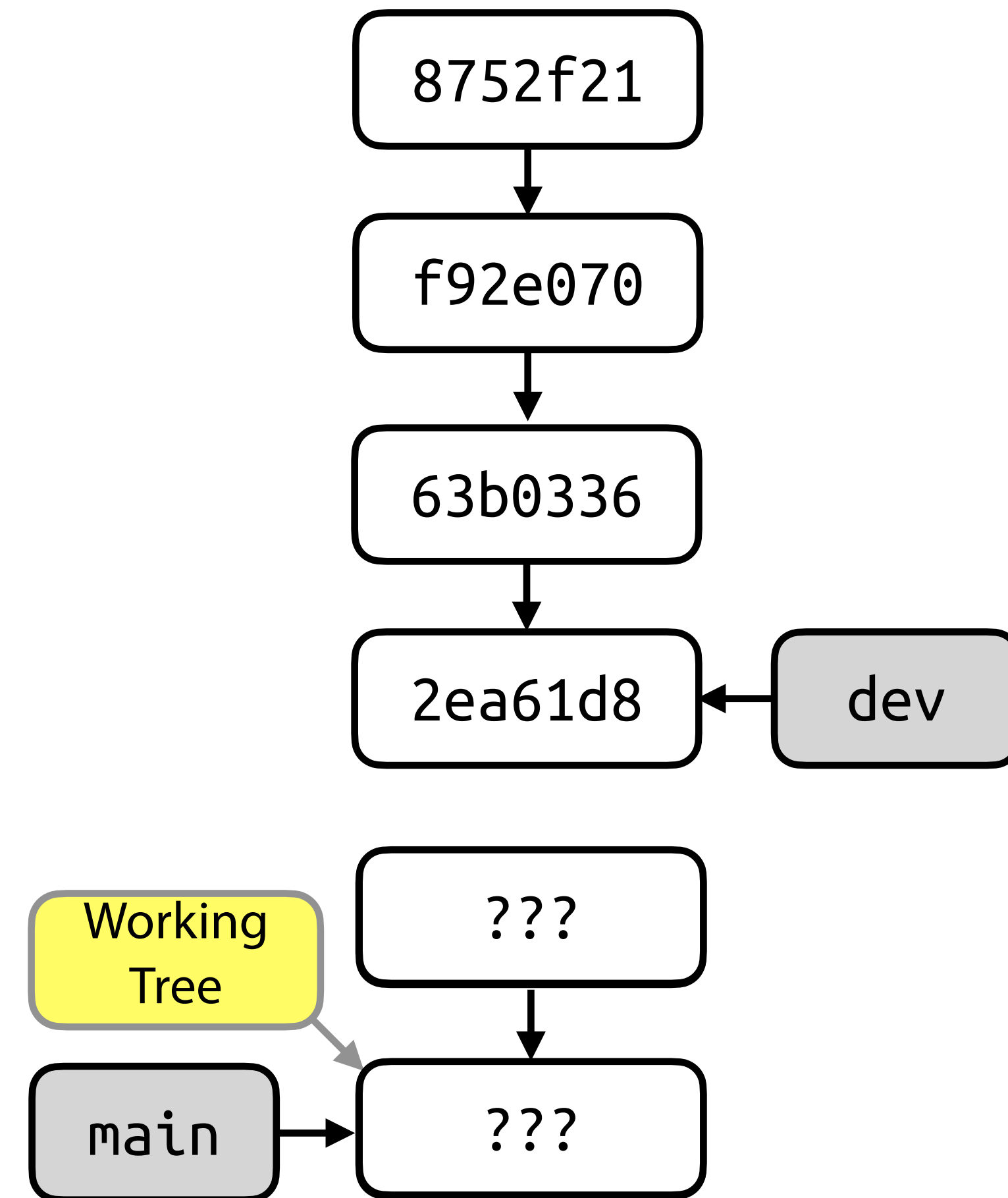
Weitere Zusammenführungen

Fast-Forward



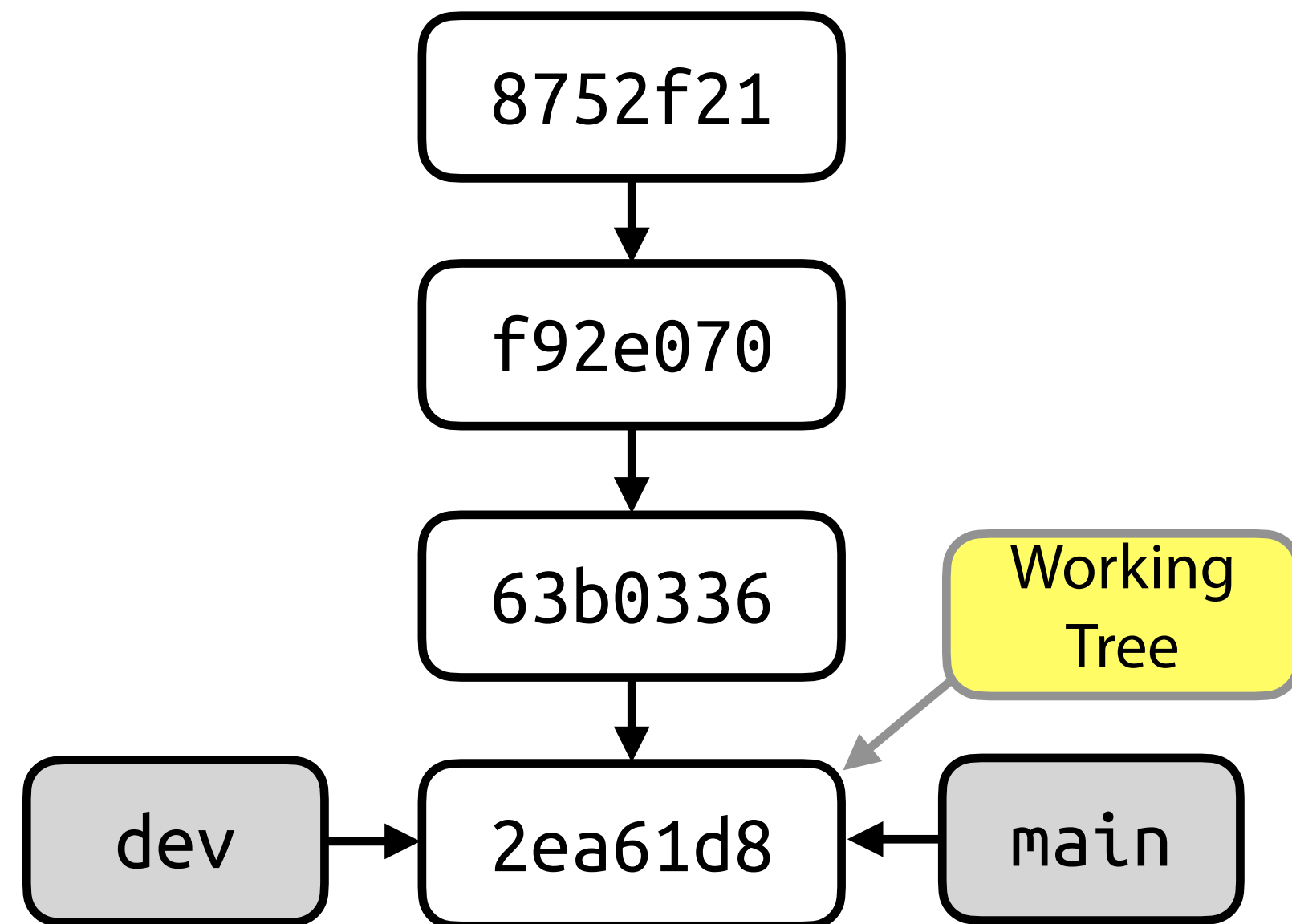
\$> git merge dev

Rebase



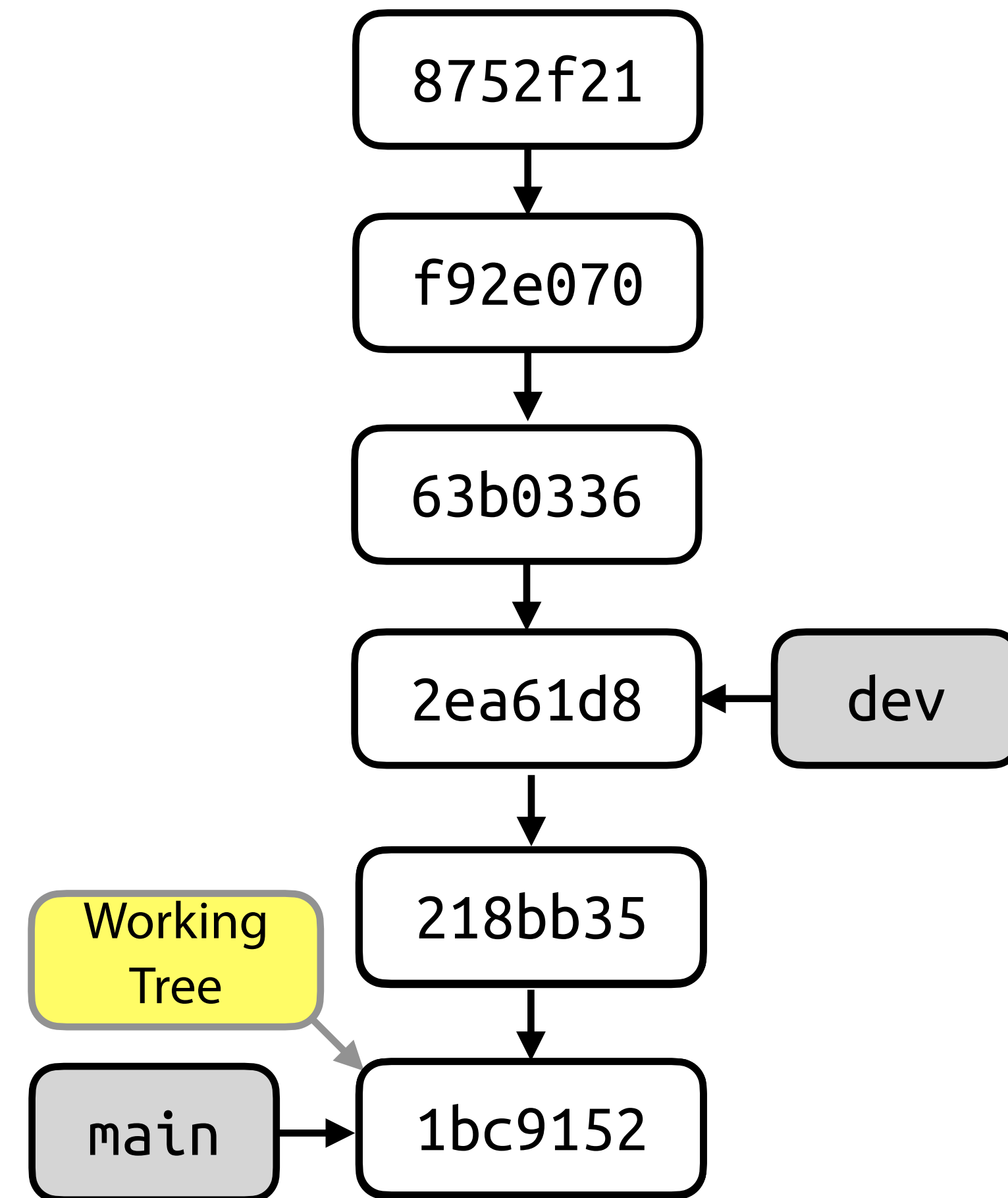
Weitere Zusammenführungen

Fast-Forward



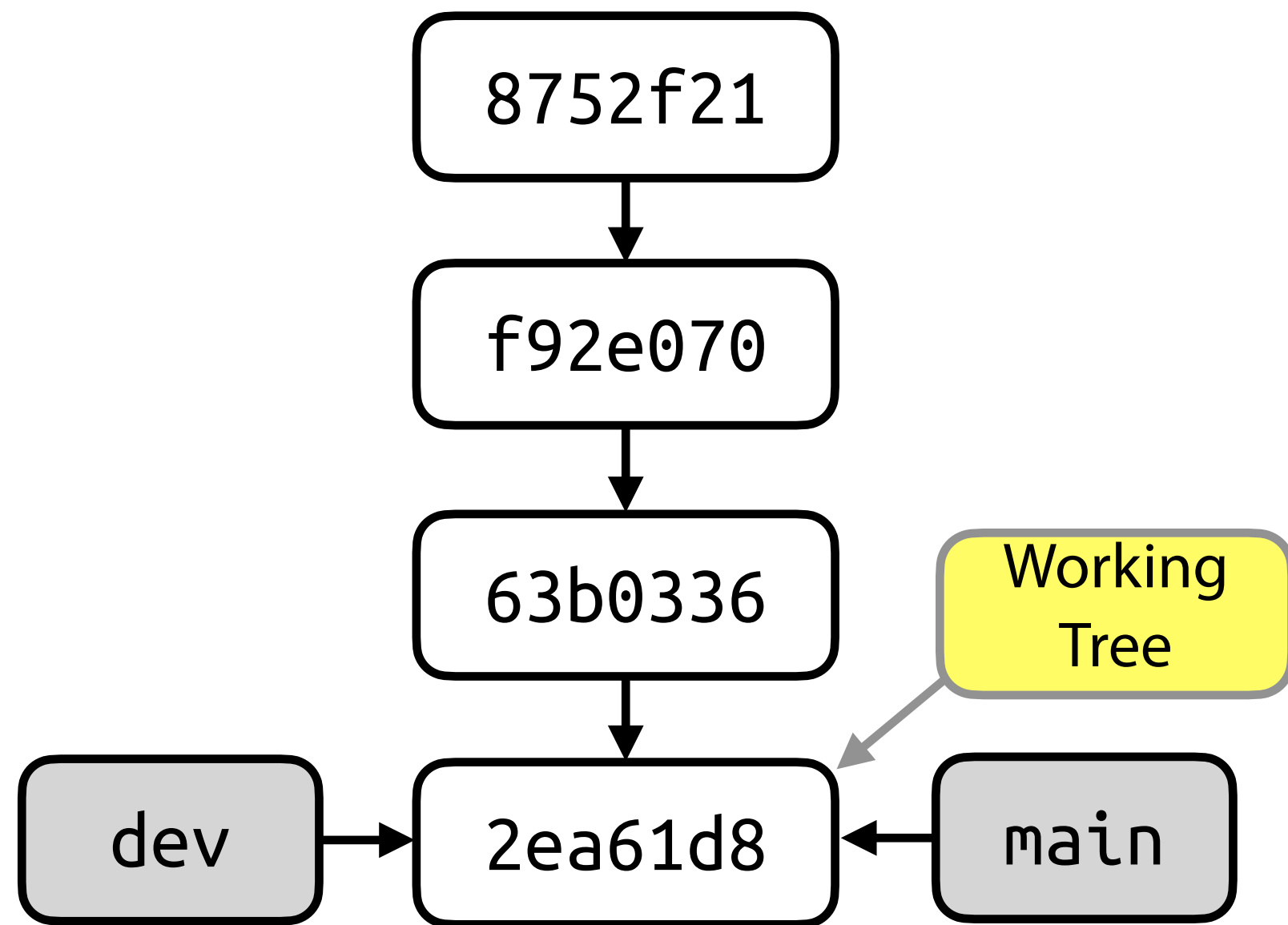
\$> git merge dev

Rebase



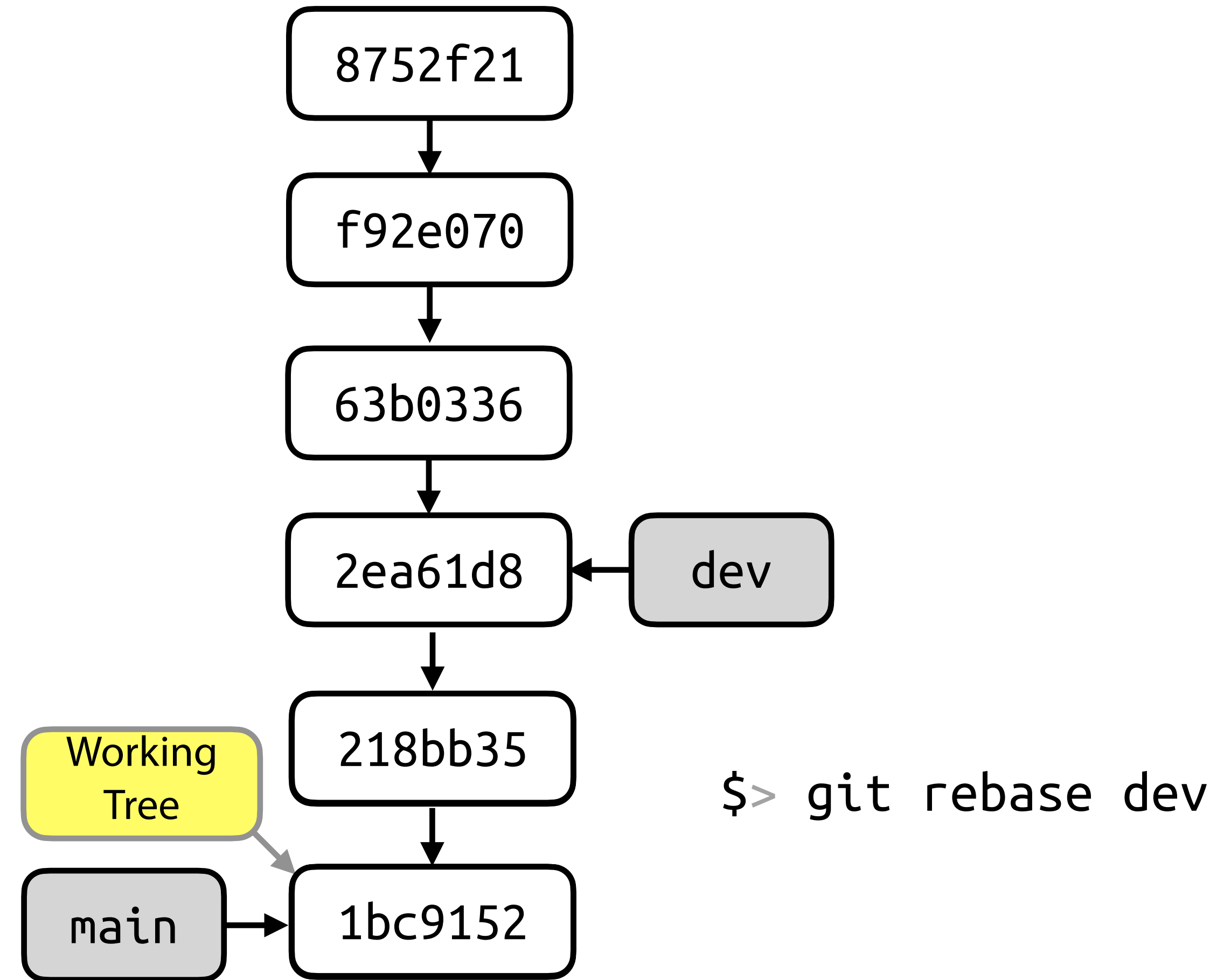
Weitere Zusammenführungen

Fast-Forward



\$> git merge dev

Rebase



„Gitignore“

- Binäre- und andere Nicht-Textdateien kann Git nur schlecht verwalten
- Kompilierte Programme, LaTeX-PDFs möchte man daher nicht im Repository haben

„Gitignore“

- Binäre- und andere Nicht-Textdateien kann Git nur schlecht verwalten
- Kompilierte Programme, LaTeX-PDFs möchte man daher nicht im Repository haben

.gitignore

```
.DS_Store  
__pycache__  
  
*.aux  
*.fdb_latexmk  
*.log  
*.pdf  
  
/data/*  
/bin/*
```

„Gitignore“

- Binäre- und andere Nicht-Textdateien kann Git nur schlecht verwalten

Funktioniert ganz normal, aber Merge-Konflikte sind dann schwer zu lösen!

- Kompilierte Programme, LaTeX-PDFs möchte man daher nicht im Repository haben

.gitignore

```
.DS_Store
__pycache__

*.aux
*.fdb_latexmk
*.log
*.pdf

/data/*
/bin/*
```

„Gitignore“

- Binäre- und andere Nicht-Textdateien kann Git nur schlecht verwalten
- Kompilierte Programme, LaTeX-PDFs möchte man daher nicht im Repository haben

Funktioniert ganz normal, aber Merge-Konflikte sind dann schwer zu lösen!

```
.gitignore
.DS_Store
__pycache__

*.aux
*.fdb_latexmk
*.log
*.pdf

/data/*
/bin/*
```

Bestimmte Datei-/ Ordnernamen ausschließen

Auch hier Glob-Pattern unterstützt

Explizit einen Pfad ausschließen

Anforderungen Versionsverwaltung

- Verlauf der Änderungen (textbasierte Dateien)
- Verschiedene **Entwicklungszeige** gleichzeitig
 - Verschiedene (neue) Features und Fehlerbehebungen
 - Verschiedene Orte
- Zusammenführen von **Entwicklungszeigen**
- Ein (zentrales) Repository

Anforderungen Versionsverwaltung

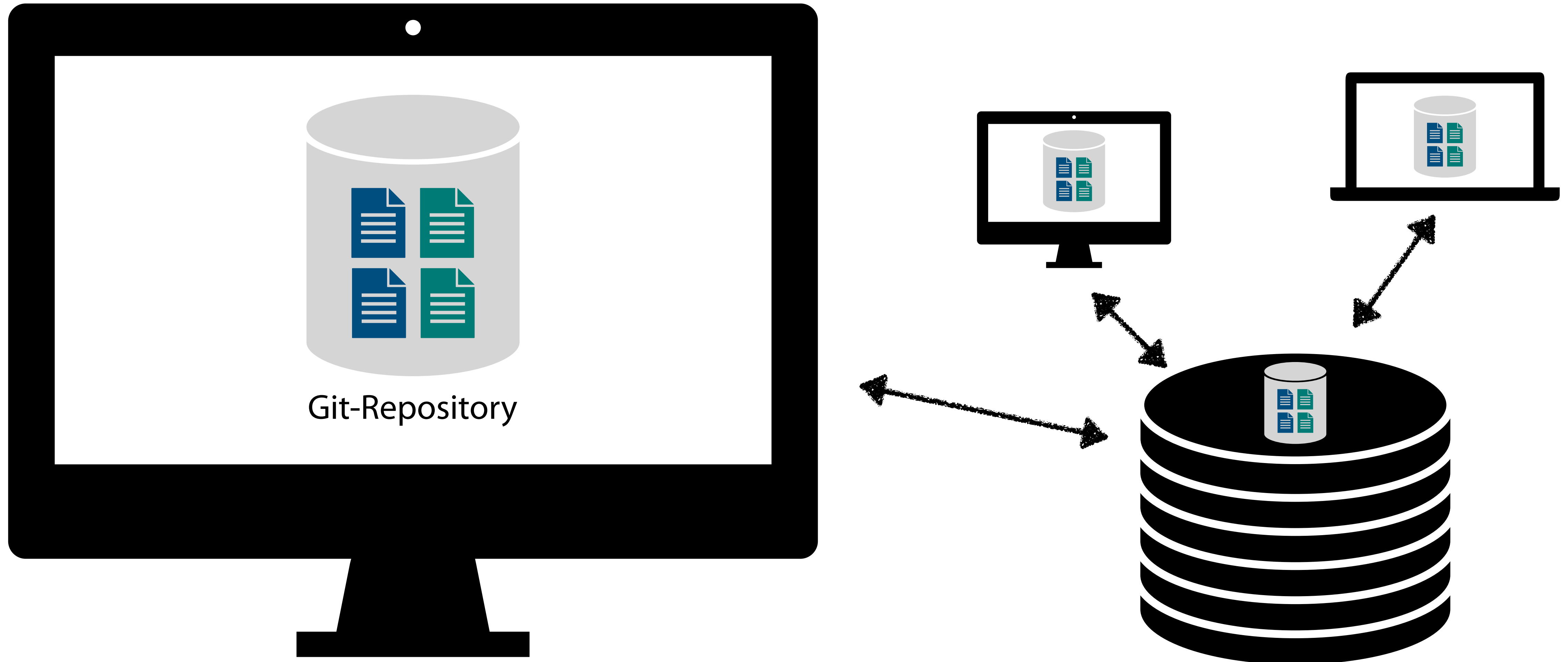
- ☑ Verlauf der Änderungen (textbasierte Dateien)
- ☑ Verschiedene **Entwicklungszeige** gleichzeitig
 - Verschiedene (neue) Features und Fehlerbehebungen
 - Verschiedene Orte
- ☑ Zusammenführen von **Entwicklungszeigen**
- ☑ Ein (zentrales) Repository

Naja, wir arbeiten immer im lokalen Repository!

II. Git

3. Remote: Push, Pull

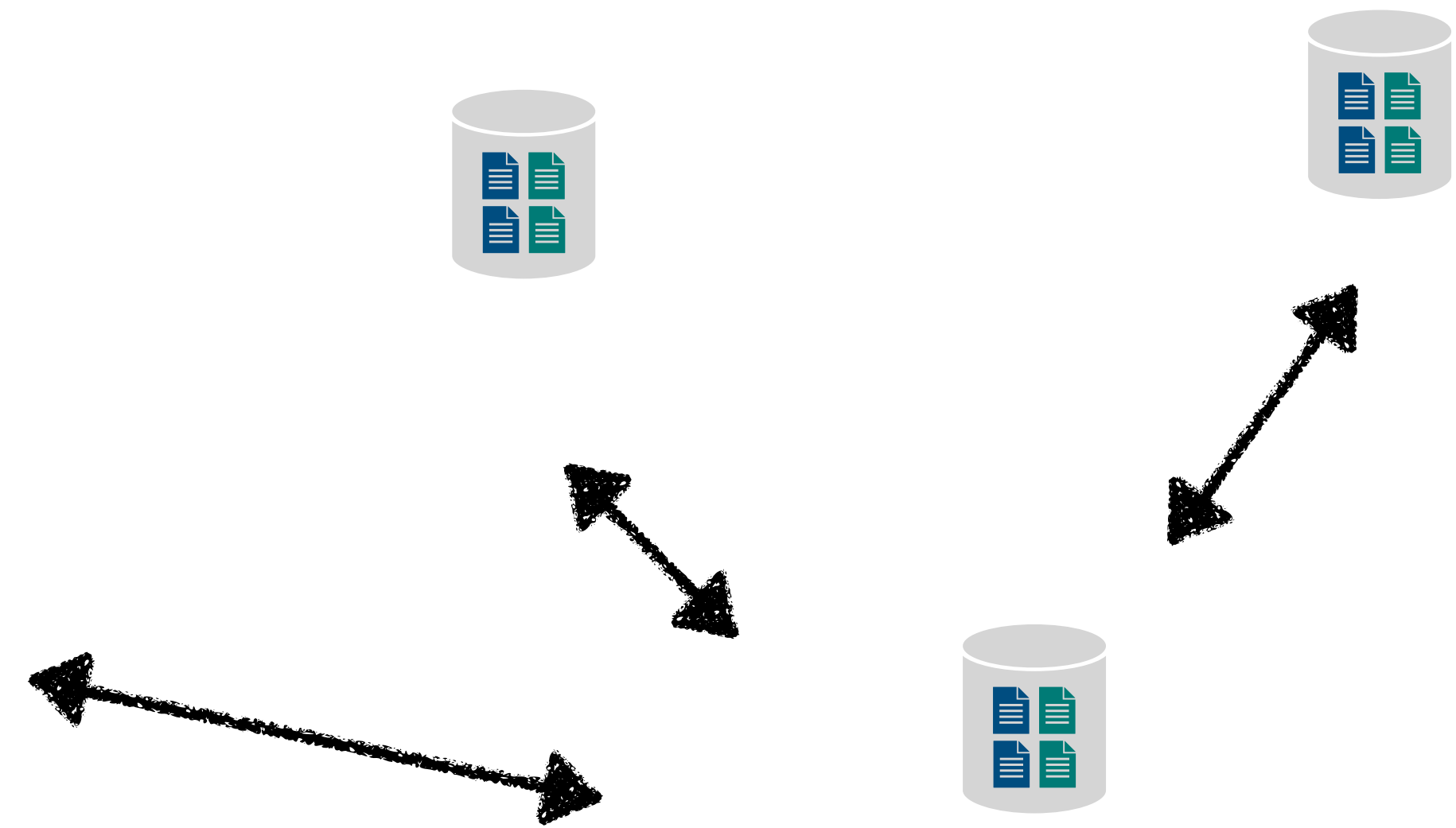
Verteilte Repositories



Verteilte Repositories





Git-Repository





Remote Repository

- Zugriff über eine URL
- Zugriff über Netzwerk oder auch lokal möglich
- Lese- und/ oder Schreibrechte

Remote Repository

- Zugriff über eine URL
 - Zugriff über Netzwerk oder auch lokal möglich
 - Lese- und/ oder Schreibrechte
- Verschiedene Protokolle
 - SSH  Erfordert eine Authentifikation
 - HTTP(S)  Erfordert eine Authentifikation nur beim „Hochladen“

Remote Repository



- Zugriff über eine URL
- Zugriff über Netzwerk oder auch lokal möglich
- Lese- und/ oder Schreibrechte
- Verschiedene Protokolle
 - SSH 
 - HTTP(S) 

```
$> git remote add NAME URL
```

```
$> git remote add origin https://github.com/torvalds/linux.git
```

```
$> git remote add MyCopy git@github.com:myuser/linux.git
```

Remote Repository

- Zugriff über eine URL
- Zugriff über Netzwerk oder auch lokal möglich
- Lese- und/ oder Schreibrechte
- Verschiedene Protokolle
 - SSH 
 - HTTP(S) 



```
$> git remote add NAME URL
```

```
$> git remote add origin https://github.com/torvalds/linux.git
```

Server Nutzer/
Besitzer Repository

```
$> git remote add MyCopy git@github.com:myuser/linux.git
```


Remote Repository

- Zugriff über eine URL
- Zugriff über Netzwerk oder auch lokal möglich
- Lese- und/ oder Schreibrechte
- Verschiedene Protokolle
 - SSH 
 - HTTP(S) 

```
$> git remote add NAME URL
```

Server Nutzer/
 Besitzer Repository

```
$> git remote add origin https://github.com/torvalds/linux.git
```

```
$> git remote add MyCopy git@github.com:myuser/linux.git
```

 Server Nutzer/
 Besitzer Repository

Remote Repository

- Zugriff über eine URL
- Zugriff über Netzwerk oder auch lokal möglich
- Lese- und/ oder Schreibrechte

- Verschiedene Protokolle

- SSH

Erfordert eine Authentifikation

- HTTP(S)

Erfordert eine Authentifikation nur beim „Hochladen“

```
$> git remote add NAME URL
```

```
$> git remote add origin https://github.com/torvalds/linux.git
```

```
$> git remote add MyCopy git@github.com:myuser/linux.git
```

Server

Nutzer/
Besitzer

Repository

Quelle, üblicherweise benannt als origin
Schreiben i.A. nicht erlaubt

Server

Nutzer/
Besitzer

Repository

Eigene Kopie, benannt als MyCopy
Schreiben i.A. möglich

Herunterladen

Git-Repository

Stash

Working Tree

Index

Repository

Remote

Herunterladen

Git-Repository

Stash

Working Tree

Index

Repository

Remote



```
git fetch [--all | REMOTE BRANCH]
git pull REMOTE BRANCH
```

Herunterladen

Git-Repository

Stash

Working Tree

Index

Repository

Remote



```
git fetch [--all | REMOTE BRANCH]
git pull REMOTE BRANCH
```

```
$> git fetch --all
remote: Enumerating objects: 513, done.
remote: Counting objects: 100% (116/116), done.
remote: Compressing objects: 100% (107/107), done.
remote: Total 513 (delta 48), reused 0 (delta 0), pack-reused 397
Receiving objects: 100% (513/513), 271.85 KiB | 1.93 MiB/s, done.
Resolving deltas: 100% (289/289), done.
From https://server/user/repo
* [new branch]      main      -> origin/main
$> git merge origin/main
```

Es können natürlich Merge-Konflikte auftreten.

Herunterladen

Git-Repository

Stash

Working Tree

Index

Repository

Remote

`git fetch [--all | REMOTE BRANCH]`
`git pull REMOTE BRANCH`

```
$> git fetch --all
remote: Enumerating objects: 513, done.
remote: Counting objects: 100% (116/116), done.
remote: Compressing objects: 100% (107/107), done.
remote: Total 513 (delta 48), reused 0 (delta 0), pack-reused 397
Receiving objects: 100% (513/513), 271.85 KiB | 1.93 MiB/s, done.
Resolving deltas: 100% (289/289), done.
From https://server/user/repo
* [new branch]      main      -> origin/main
$> git merge origin/main
```


Es können natürlich Merge-Konflikte auftreten.

Herunterladen

Git-Repository

Stash

Working Tree

Index

Repository

Remote

`git fetch [--all | REMOTE BRANCH]`
`git pull REMOTE BRANCH`

```
$> git fetch --all
remote: Enumerating objects: 513, done.
remote: Counting objects: 100% (116/116), done.
remote: Compressing objects: 100% (107/107), done.
remote: Total 513 (delta 48), reused 0 (delta 0), pack-reused 397
Receiving objects: 100% (513/513), 271.85 KiB | 1.93 MiB/s, done.
Resolving deltas: 100% (289/289), done.
From https://server/user/repo
* [new branch]      main      -> origin/main
$> git merge origin/main
```

```
$> git pull origin main
```


Es können natürlich Merge-Konflikte auftreten.

Herunterladen

Git-Repository

Alle Änderungen (Commits, Branches) von allen Remotes in das lokale Repository herunterladen.
(Keine Änderung am Working Tree)

Repository

Remote

```
git fetch [--all | REMOTE BRANCH]
git pull REMOTE BRANCH
```

```
$> git fetch --all
remote: Enumerating objects: 513, done.
remote: Counting objects: 100% (116/116), done.
remote: Compressing objects: 100% (107/107), done.
remote: Total 513 (delta 48), reused 0 (delta 0), pack-reused 397
Receiving objects: 100% (513/513), 271.85 KiB | 1.93 MiB/s, done.
Resolving deltas: 100% (289/289), done.
From https://server/user/repo
* [new branch]      main      -> origin/main
$> git merge origin/main
```

```
$> git pull origin main
```

Fetch und Merge in einem Schritt

Den aktuellen lokalen Branch mit einem remote Branch mergen.

Neu herunterladen

Git-Repository

Stash

Working Tree

Index

Repository

Remote



```
git fetch [--all | REMOTE BRANCH]
git pull REMOTE BRANCH
```

```
git clone URL
```

```
$> mkdir linux
$> cd ./linux/
$> git init
$> git remote add origin https://github.com/torvalds/linux.git
$> git pull origin master
```

Neu herunterladen

Git-Repository

Stash

Working Tree

Index

Repository

Remote



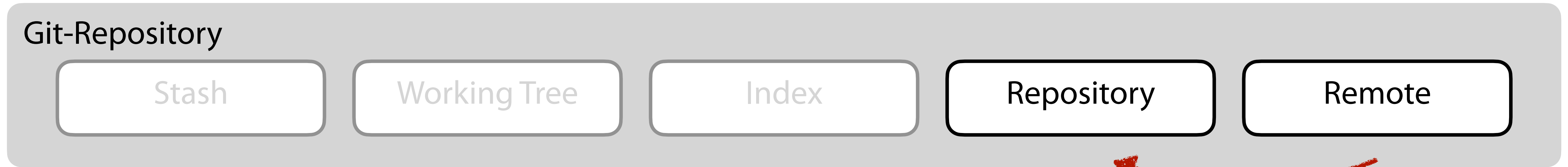
```
git fetch [--all | REMOTE BRANCH]
git pull REMOTE BRANCH
```

```
git clone URL
```

```
$> mkdir linux
$> cd ./linux/
$> git init
$> git remote add origin https://github.com/torvalds/linux.git
$> git pull origin master
```

```
$> git clone
https://github.com/
torvalds/linux.git
```

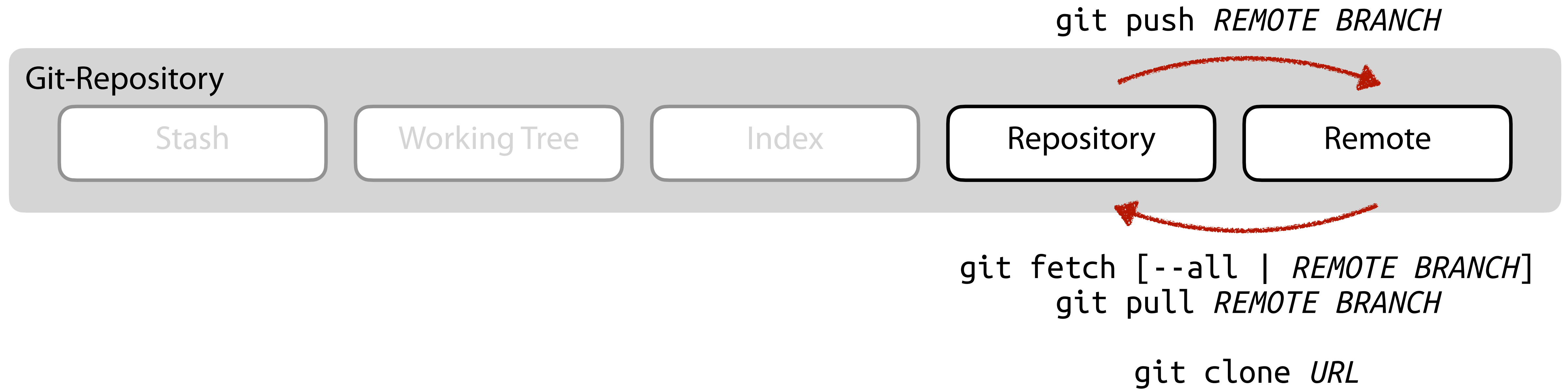
Hochladen



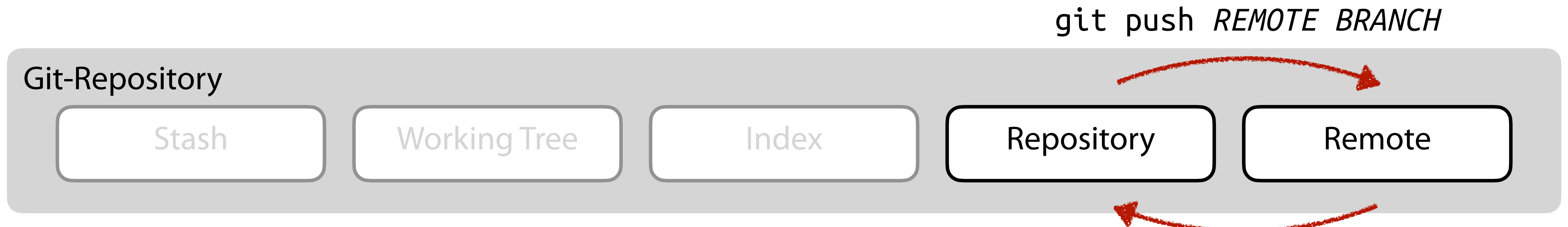
```
git fetch [--all | REMOTE BRANCH]  
git pull REMOTE BRANCH
```

```
git clone URL
```

Hochladen



Hochladen



```
$> git commit -m "Meine Änderung"  
$> git push MyCopy main
```

```
Enumerating objects: 513, done.  
Counting objects: 100% (513/513), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (200/200), done.  
Writing objects: 100% (513/513), 271.84 KiB | 2.75 MiB/s, done.  
Total 513 (delta 289), reused 513 (delta 289), pack-reused 0  
remote: Resolving deltas: 100% (289/289), done.  
To server:user/repo.git  
* [new branch]      main -> main
```

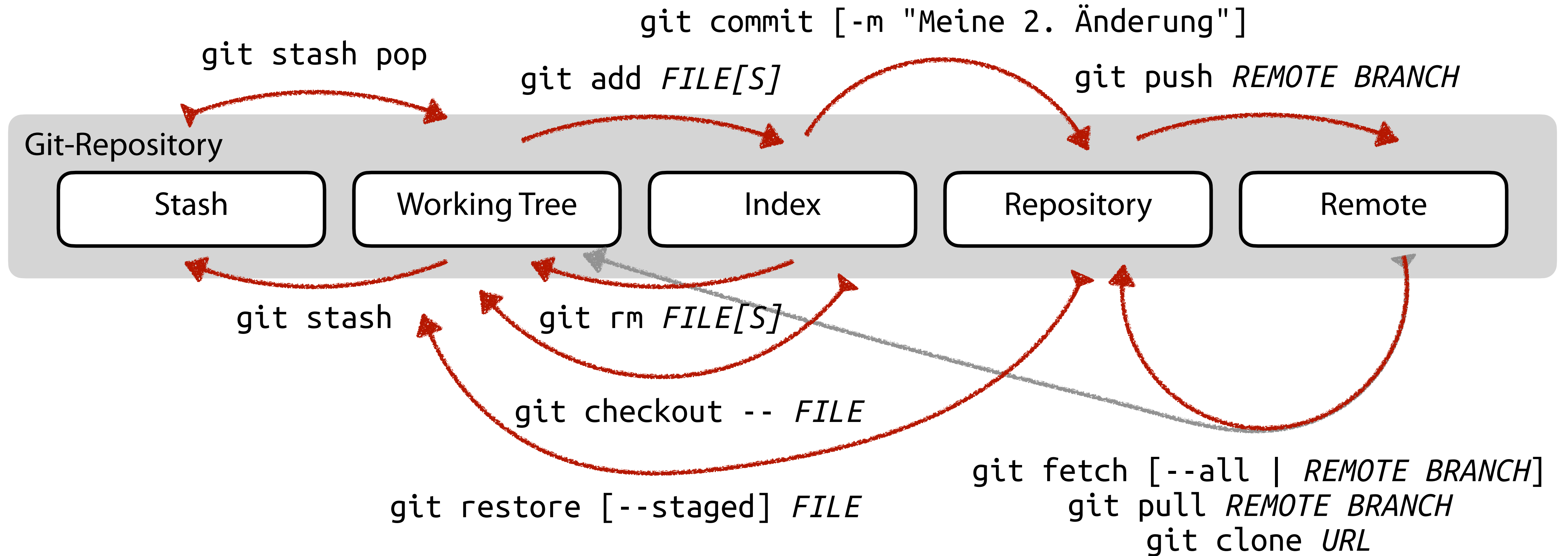
```
git fetch [ --all | REMOTE BRANCH ]  
git pull REMOTE BRANCH
```

```
git clone URL
```


Git Befehle I

Konfiguration	<code>git config [--global] user.name "NAME"</code>	Angabe des Namens, mit dem Commits unterschrieben werden
	<code>git config [--global] user.email "E-MAIL"</code>	Angabe der Mail-Adresse, die in Commits angegeben wird
Repository, Verlauf	<code>git init</code>	Erzeugen eines leere Repository
	<code>git log --oneline --graph</code>	Anzeige des Verlaufs der Commits
	<code>git checkout HASH</code>	Einen Commit im Working Tree öffnen
	<code>git checkout HEAD</code>	Zurück zum Kopf des Repositories
Working Tree, Index, Commit	<code>git add FILE[S]</code>	Eine Datei/ Pfad zum Commit vormerken (<i>staged</i>)
	<code>git status</code>	Status des Repository mit Dateienstatus anzeigen
	<code>git commit [-m "MESSAGE"]</code>	Erstellen eines Commit
	<code>git rm FILE[S]</code>	Löschen von Dateien aus dem Index (nicht auch alten Commits)
	<code>git checkout -- FILE</code>	Zurücksetzen einer Datei im Working Tree auf <i>staged</i> Version
	<code>git restore [--staged] FILE</code>	Zurücksetzen einer Datei im Working Tree auf letzten Commit
Stash	<code>git stash</code>	Working Tree im Zwischenspeicher ablegen
	<code>git stash pop</code>	Zwischenspeicher auf Working Tree anwenden
Branches, Merge	<code>git branch NAME</code>	Einen neuen Branch „hier“ erzeugen
	<code>git checkout BRANCHNAME</code>	Eine Branch im Working Tree öffnen
	<code>git merge BRANCHNAME</code>	Einen anderen Branch in den aktuellen Branch mergen
Remote	<code>git remote add NAME URL</code>	Eine Remote-Repository anbinden
	<code>git push REMOTE BRANCH</code>	Einen Branch in das Remote-Repository „hochladen“
	<code>git fetch [--all REMOTE BRANCH]</code>	Einen Branch oder alles mit dem Remote-Repository „abgleichen“
	<code>git pull REMOTE BRANCH</code>	Eine Branch vom Remote-Repository in den Working Tree „herunterladen“
	<code>git clone URL</code>	Eine Repository von einer URL „herunterladen“ und lokal erstellen

Git Befehle II



III. GitHub



Git Repository Hosting

- GitHub, GitLab, Bitbucket, ...
- Webinterface zur Darstellung des Codes
 - Commits, Branches, ...
 - Forks, Pull Requests, ...
- Projektmanagement
 - Issues, ...

Git Repository Hosting

- GitHub, GitLab, Bitbucket
- Webinterface zur Verwaltung von
- Commits, Branches
- Forks, Pull Requests
- Projektmanagement
- Issues, ...

The screenshot shows the GitHub repository page for `matplotlib/matplotlib`. The page is in German and displays the following information:

- Repository Name:** matplotlib/matplotlib (Public)
- Actions:** Sponsor, Notifications, Fork (6.6k), Star (16.5k)
- Navigation:** Code, Issues (1.6k), Pull requests (337), Actions, Projects (6), Wiki, Security, Insights
- Branches:** main (19 branches), 115 tags
- Recent Commits:** A list of recent commits with their descriptions and dates. For example, a commit by `tacaswell` titled "Merge pull request #24539 from anntzer/24257" is shown, along with other commits like ".circleci: Show errors and warnings as well as deprecation warnings in ...".
- About:** matplotlib: plotting with Python. Includes a link to `matplotlib.org/` and tags like `python`, `gtk`, `data-science`, `qt`, `data-visualization`, `tk`, `matplotlib`, `plotting`, `hacktoberfest`, and `wx`.
- Readme:** Readme, Code of conduct, Security policy, Cite this repository
- Stars:** 16.5k stars, 586 watching, 6.6k forks
- Releases:** 69 releases. The latest release is `REL: v3.6.2` (Latest), released 22 days ago. There are +68 more releases.

Git Repository Hosting

- GitHub, GitLab, Bitbucket
- Webinterface zur Verwaltung von
- Commits, Branches
- Forks, Pull Requests
- Projektmanagement
- Issues, ...

github.com/matplotlib/matplotlib

Product Solutions Open Source Pricing

Search / Sign in Sign up

matplotlib/matplotlib Public

Sponsor Notifications Fork 6.6k Star 16.5k

<> Code Issues 1.6k Pull requests 337 Actions Projects 6 Wiki Security Insights

main 19 branches 115 tags

Go to file Code

About

matplotlib: plotting with Python

matplotlib.org/

python gtk data-science qt

data-visualization tk matplotlib

plotting hacktoberfest wx

Readme

Code of conduct

Security policy

Cite this repository

16.5k stars

586 watching

6.6k forks

Releases 69

REL: v3.6.2 Latest

22 days ago

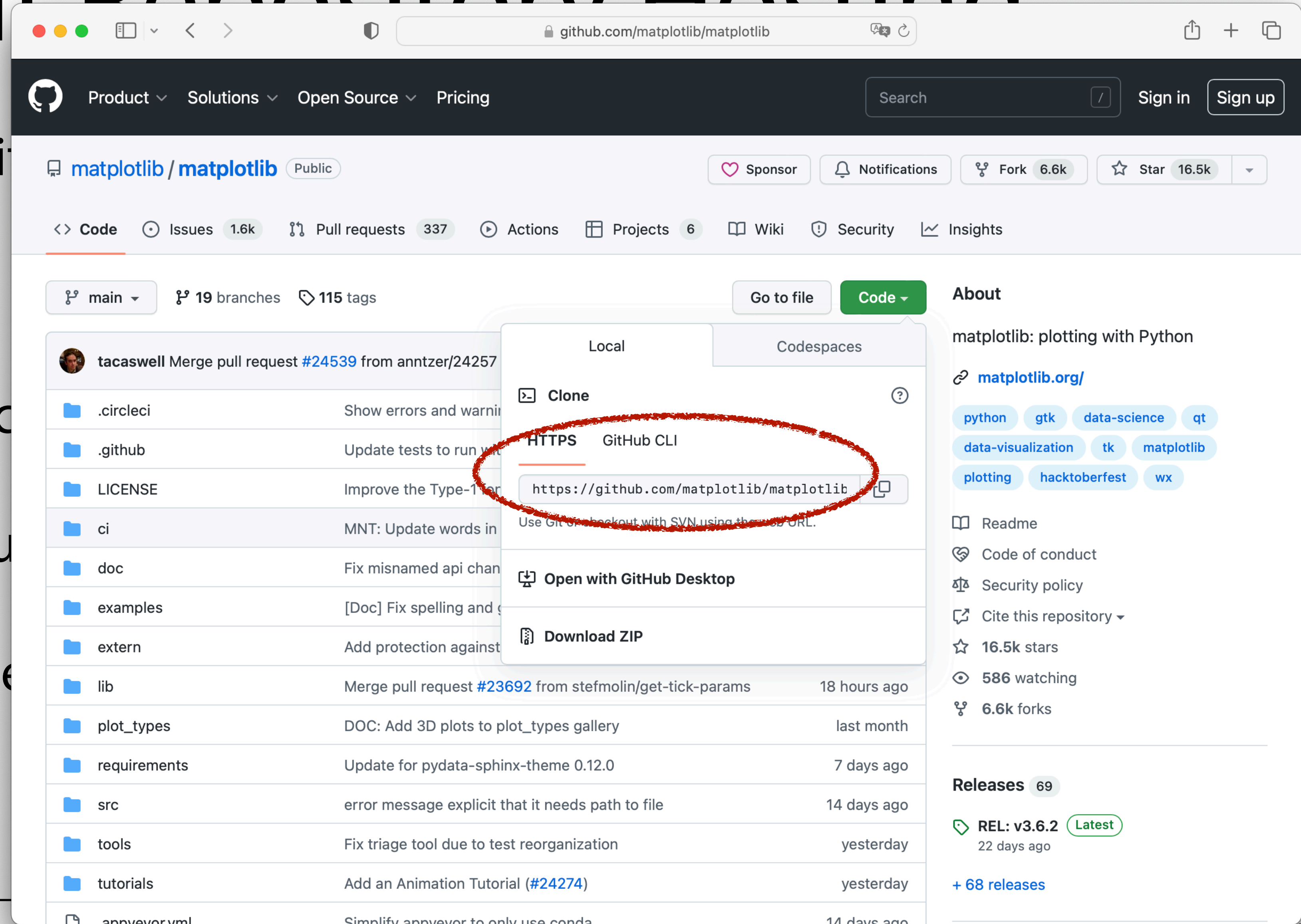
+ 68 releases

tacaswell Merge pull request #24539 from anntzer/24257 ... 7469aaa 16 hours ago 45,011 commits

.circleci	Show errors and warnings as well as deprecation warnings in ...	2 months ago
.github	Update tests to run with 3.11 (not rc)	7 days ago
LICENSE	Improve the Type-1 font parsing	15 months ago
ci	MNT: Update words in codespell-ignore-words file	last month
doc	Fix misnamed api changes entry.	17 hours ago
examples	[Doc] Fix spelling and grammar in examples	4 days ago
extern	Add protection against out-of-bounds read in ttconv	17 months ago
lib	Merge pull request #23692 from stefmolin/get-tick-params	18 hours ago
plot_types	DOC: Add 3D plots to plot_types gallery	last month
requirements	Update for pydata-sphinx-theme 0.12.0	7 days ago
src	error message explicit that it needs path to file	14 days ago
tools	Fix triage tool due to test reorganization	yesterday
tutorials	Add an Animation Tutorial (#24274)	yesterday
appveyor.yml	Simplify appveyor to only use conda	14 days ago

Git Repository Hosting

- GitHub, GitLab, Bitbucket
- Webinterface zur Verwaltung von
- Commits, Branches
- Forks, Pull Requests
- Projektmanagement
- Issues, ...



Git Repository Hosting

- GitHub, GitLab, Bitbucket
- Webinterface zur Verwaltung von Repositories
- Commits, Branches, Tags
- Forks, Pull Requests
- Projektmanagement
 - Issues, ...

The screenshot displays the GitHub repository page for `matplotlib/matplotlib`. The file list on the left shows several files, with `README.md` circled in red. The commit history table lists recent changes, including updates to the README, security reporting, and package updates. The right sidebar shows contributor information (+1,211 contributors) and a language usage chart dominated by Python (90.8%). The main content area shows the `README.md` header with project badges for PyPI (3.6.2), downloads (32M), and various testing and quality metrics. The `matplotlib` logo is prominently displayed at the bottom.

File	Commit Message	Time Ago
README.md	.rst to .md README	2 months ago
SECURITY.md	GOV: change security reporting to use tidelift	24 days ago
azure-pipelines.yml	Update name of package libgirepository-1.0.1	3 months ago
environment.yml	Simplify appveyor to only use conda	14 days ago
mplsetup.cfg.template	Move gui_support.macosx option to packages section.	13 months ago
pyproject.toml	Use oldest-supported-numpy for build	29 days ago
pytest.ini	Restore accidentally removed pytest.ini and tests.py.	6 months ago
setup.cfg	Move setup.cfg to mplsetup.cfg.	15 months ago
setup.py	Load style files from third-party packages.	21 days ago
setupext.py	Split toolkit tests into their toolkits	13 days ago
tests.py	Restore accidentally removed pytest.ini and tests.py.	6 months ago
tox.ini	Drop support for Python 3.7	10 months ago

Languages:

- Python 90.8%
- C++ 6.4%
- Jupyter Notebook 1.2%
- Objective-C 0.8%
- JavaScript 0.4%
- C 0.2%
- Other 0.2%

Project Badges:

- PyPI package 3.6.2
- downloads/month 32M
- powered by NumFOCUS
- help forum discourse
- chat on gitter
- issue tracking github
- PR Welcome
- Tests passing
- Azure Pipelines succeeded
- build unknown
- codecov 89%
- code quality: python A

Zusammenfassung

I. Versionsverwaltung

II. Git

1. Idee, Konfiguration

2. Lokal: Commit, Stash, Branch, Merge

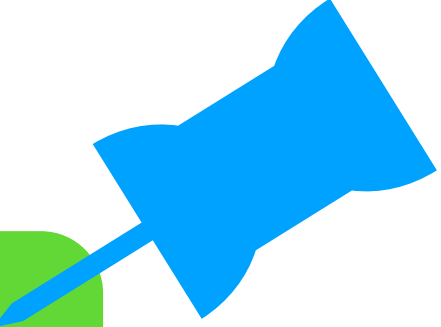
3. Remote: Push, Pull

III. GitHub



Zusammenfassung

- I. Versionsverwaltung
- II. Git
 - 1. Idee, Konfiguration
 - 2. Lokal: Commit, Stash, Branch, Merge
 - 3. Remote: Push, Pull
- III. GitHub



Nächste Woche findet ein Übungstermin im PC Pool zu den Projektaufgaben 2 & 3 statt.



~~Heute~~

Inhaltsübersicht

1. Programmiersprache Python
 - a) *Einführung, Erste Schritte*
 - b) *Grundlagen*
 - c) *Fortgeschritten*
2. Auszeichnungssprachen
 - a) *LaTeX, Markdown*
3. Benutzeroberflächen und Entwicklungsumgebungen
 - a) *Jupyter Notebooks lokal und in der Cloud (Google Colab)*
4. Versionsverwaltung
 - a) *Git, GitHub*
5. Wissenschaftliches Rechnen
 - a) **NumPy, SciPy**
6. Datenverarbeitung und -visualisierung
 - a) Pandas, matplotlib, NLTK
7. Machine Learning (scikit-learn)
 - a) Grundlegende Ansätze (Datensätze, Auswertung)
 - b) Einfache Verfahren (Clustering, ...)
8. DeepLearning
 - a) TensorFlow, PyTorch, HuggingFace Transformers