

# Git – das Kartenspiel\*

Philipp Müller

21. Februar 2026

Dieses Spiel parodiert die agile Software-Entwicklung. Die Spieler bilden dabei ein Team von Entwicklern, die gemeinsam eine Software weiterentwickeln und dazu das Versionsverwaltungsprogramm Git verwenden. Ziel des Spiels ist es jedoch, möglichst viele der eigenen Quellcodezeilen durchzubringen und die anderen Spieler genau daran zu hindern.

## 1 – Aufbau

Das Spiel besteht aus vier Arten von Karten: Issue-, Commit-, Release- und sonstigen Karten.

Die 9 **Issue-Karten** werden am Anfang von den anderen Karten getrennt, gut gemischt und bilden den Issue-Stapel. Eine Issue-Karte wird gezogen und aufgedeckt. Sie gibt das Ziel der ersten Spielrunde an und beginnt die Reihe der ausgelegten Karten, also das **Remote Repository** von Git.

Alle anderen Karten werden auch gut gemischt und bildet den Zugstapel. Jeder Spieler erhält drei davon auf die Hand, die sog. Handkarten. Diese entsprechen dem lokalen Git-Arbeitsverzeichnis des Spielers.

## 2 – Ablauf

Das Spiel wird in Runden gespielt, die jeweils der Entwicklung einer Version der Software entsprechen. Sie beginnen mit dem Aufdecken einer **Issue-Karte**, die das Punkteziel der Runde angibt, und enden mit der Auslieferung einer neuen Version der Software durch das Ausspielen einer **Release-Karte**. Dazwischen spielen alle Spieler reihum immer genau eine Handkarte aus und sagen dabei das, was auf der Karte in Anführungszeichen („...“) steht.

Es gibt drei Arten von Handkarten:

- **Commit-Karten** zeigen einen Kreis mit einem Zahlwert, der einer Anzahl geänderter Codezeilen entspricht. Sie werden an das Remote Repository angelegt und müssen dann mit einem Spielermarker markiert werden, damit man später weiß, wer welche Punkte bekommt.
- **Release-Karten** beenden die aktuelle Runde, die dann **ausgewertet** wird (s. u. 2.2). Sie können nur ausgespielt werden, wenn das aktuelle Ziel erreicht ist, d. h. wenn die Summe der ausgelegten Zahlwerte dem Wert der Issue-Karte (incl. Bugs) entspricht oder diesen übertrifft.
- **Sonstige Karten** werden entsprechend ihrer Aufschrift ausgewertet und landen danach meistens auf dem Abwurfstapel.

Bestimmte Karten dürfen nur unter bestimmten Bedingungen ausgespielt werden. Falls jemand an die Reihe kommt und keine seiner Handkarten ausspielen kann, legt er sie alle auf den Abwurfstapel. Er spielt in dieser Runde also keine Karte aus.

Am Ende des Zuges **stockt** man seine Handkarten wieder auf (s. u. 2.1) und dann ist der nächste Spieler an der Reihe.

---

\*Dies ist Version 1.1. Die aktuelle Version ist zu finden unter: <https://github.com/arkonia/GitGame>

Eine Spielrunde ist beendet, sobald jemand eine Release-Karte ausspielt. Dann wird die aktuelle Runde **ausgewertet** (s. u. 2.2) und danach beginnt die nächste Spielrunde mit dem Aufdecken der nächsten Issue-Karte oder das Spiel ist ganz beendet (s. u. 3).

## 2.1 – Aufstocken

Falls man am Ende seines Zuges weniger als drei Handkarten hat, zieht man neue Karten vom Zugstapel bis man wieder drei auf der Hand hat.

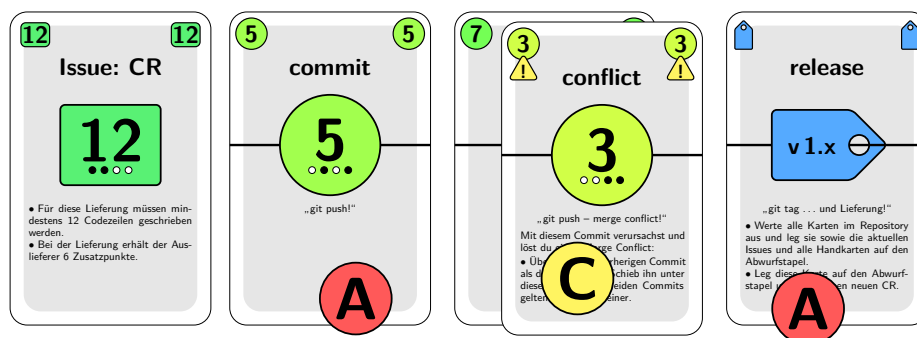
## 2.2 – Auswerten

Sobald jemand eine Release-Karte ausspielt, wird eine neue Version der Software geliefert. D. h. die aktuelle Runde wird ausgewertet. Jeder Spieler erhält Punkte entsprechend der Codezeilen all seiner Commits im Remote Repository, also zwischen Issue und Lieferung. Der Spieler der Release-Karte erhält zusätzliche Punkte wie auf der Issue-Karte angegeben.

Nach der Auswertung werden alle ausgelegten Karten und alle Handkarten abgeworfen. Jetzt können die Karten neu gemischt werden. Danach wird eine neue Issue-Karte ausgelegt und jeder Spieler erhält wieder drei Handkarten. Der Spieler nach dem Spieler, der die Release-Karte ausgespielt hat, beginnt die neue Runde.

### Beispiel für eine Auswertung

Am Ende einer Spielrunde kann das Remote Repository folgendermaßen aussehen:



- Spieler A erhält 11 Punkte: Zahlkarte 5 plus 6 Extrapunkte fürs Release.
- Spieler B erhält 0 Punkte: Seine Zahlkarte 7 wurde von Spieler C geklaut!
- Spieler C erhält 10 Punkte: Zahlkarte 7 und Zahlkarte 3 (Merge Conflict).

Die Punkte sollten in einer kleinen Tabelle aufgeschrieben werden:

Runde	A	B	C
v. 1	11	0	10
v. 2			
v. 3			

## 3 – Ende

Am Anfang sollten alle Teilnehmer festlegen, wann das Spiel beendet ist (Definition of Done), z. B. nach 5 Lieferungen oder sobald ein Spieler mindestens 100 Punkte erreicht hat.

Gewonnen hat in jedem Fall selbstverständlich der Spieler mit der höchsten Gesamtpunktezahl.

## 4 – Druckvorlagen

Diese und die folgenden Seiten sind zum Ausdrucken und Ausschneiden vorgesehen. Zum Drucken wird dickes Papier empfohlen, mindestens 160 g-Papier. Die Karten sollten natürlich beidseitig ausgedruckt werden. Die Ränder links und rechts sind gleich breit, so dass Vorder- und Rückseite immer aufeinander treffen. Die Karten sind 59 mm × 92 mm groß.

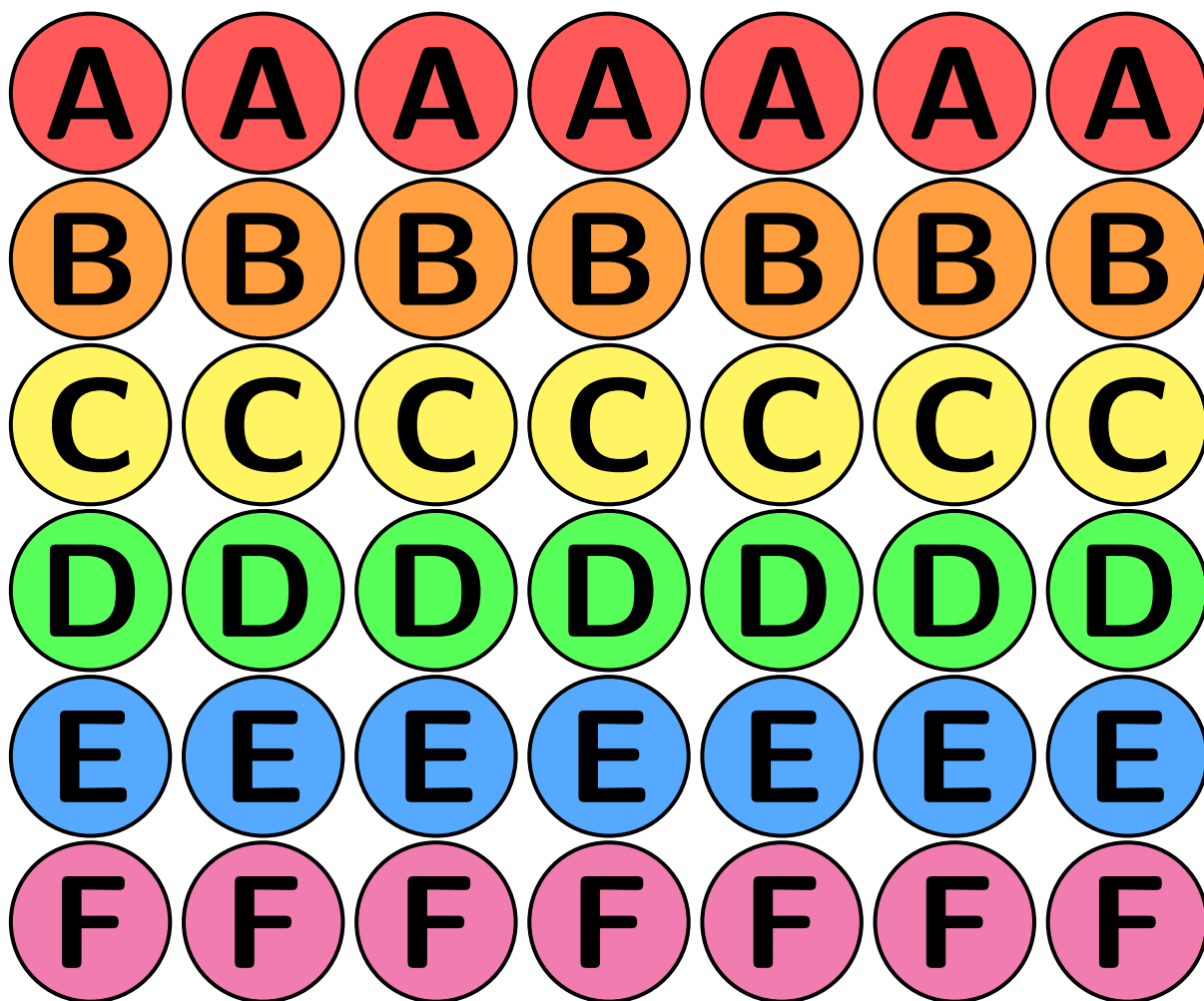
Alternativ gibt es auch Dateien, die jeweils nur genau eine Karte je Seite zeigen:

`gitgame-de_Vorderseiten.pdf` und `gitgame-de_Rueckseite.pdf`

Diese haben einen zusätzlichen umlaufenden Rand von 3 mm, die Dateien sind also 65 mm × 98 mm groß.

### Spielermarker

Statt der Spielermarker auf dieser Seite können auch irgendwelche Spielfiguren oder sonstige Chips, Münzen o. ä. zum markieren der ausgelegten Karten verwendet werden.



0

!

refactoring

0

!

1

commit

1

!

1

!

conflict

1

!

---

0

...

1

...

1

...

„git push – refactoring!“

- Mit diesem Commit übernimmst du die vorherigen drei Commits als deine eigenen. Dazu müssen schon mindestens drei Commits ausliegen. Schieb sie unter diese Karte, alle diese Commits gelten ab jetzt als einer.

„git push!“

„git push – merge conflict!“

Mit diesem Commit verursachst und löst du einen Merge Conflict:

- Übernimm den vorherigen Commit als deinen eigenen. Schieb ihn unter diese Karte, diese beiden Commits gelten ab jetzt als einer.

2

2

2

!

commit

2

!

2

?

conflict

2

?

2

?

request

2

?

---

2

...

2

...

2

...

„git push!“

„git push – merge conflict!“

Mit diesem Commit verursachst und löst du einen Merge Conflict:

- Übernimm den vorherigen Commit als deinen eigenen. Schieb ihn unter diese Karte, diese beiden Commits gelten ab jetzt als einer.

„git push – merge request!“

Mit diesem Commit *bittest* du deinen Kollegen um ein Code Review, dass er natürlich auch durchführt:

- Der nächste Spieler muss diese Karte mit einem Revert negieren oder einmal aussetzen.

3

3

3

3

3

3

3

!

3

!

---

3

...

3

...

3

...

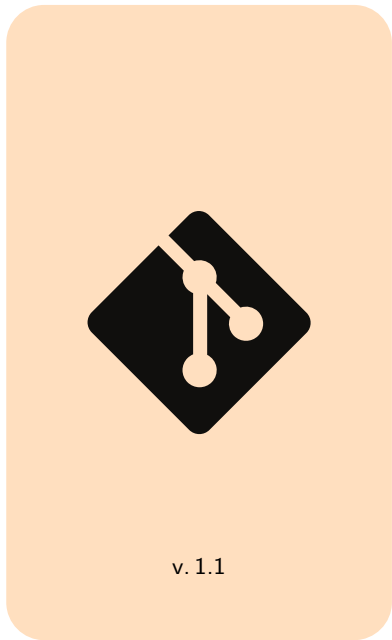
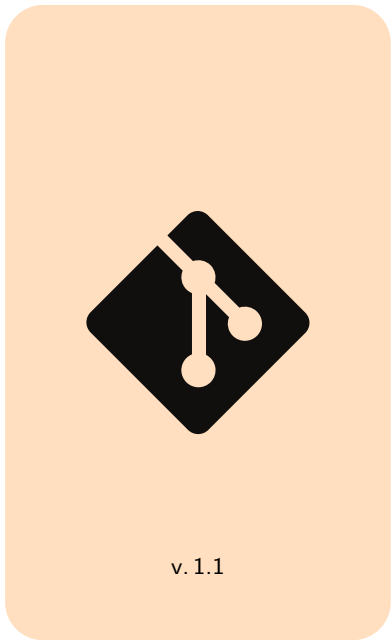
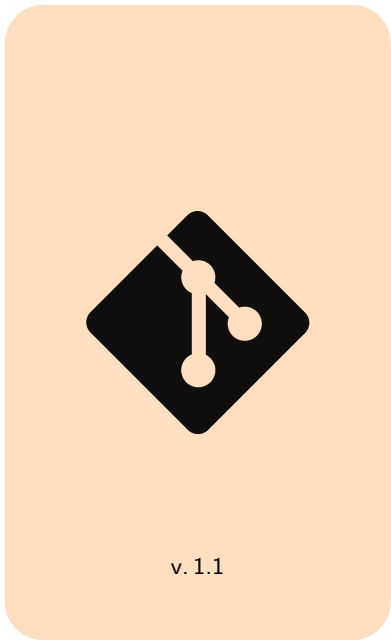
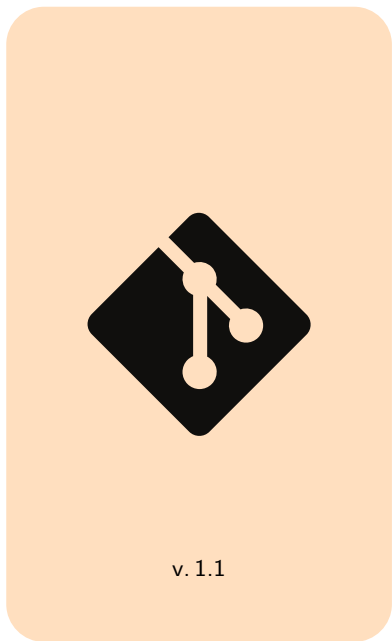
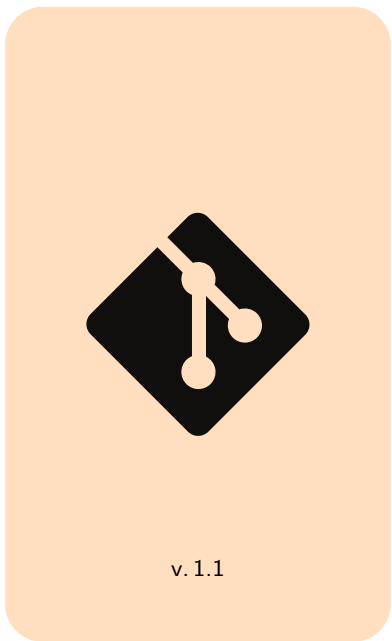
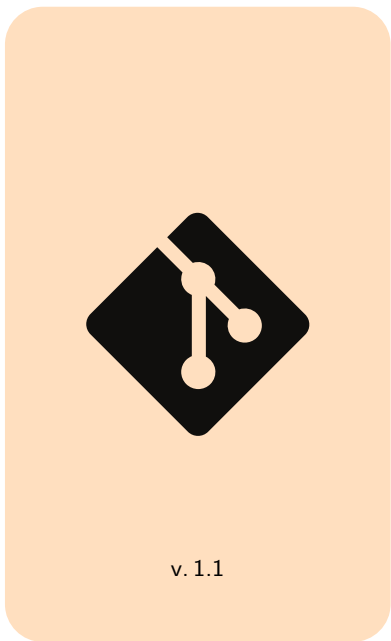
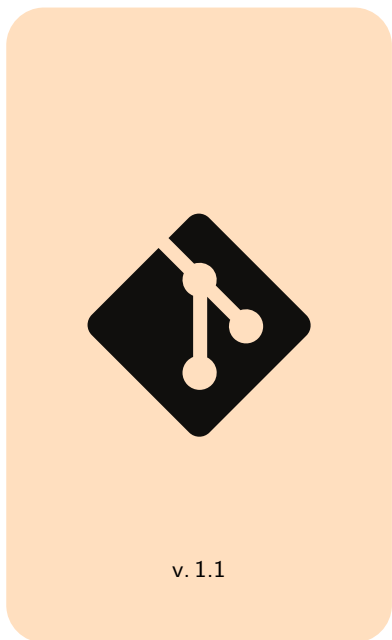
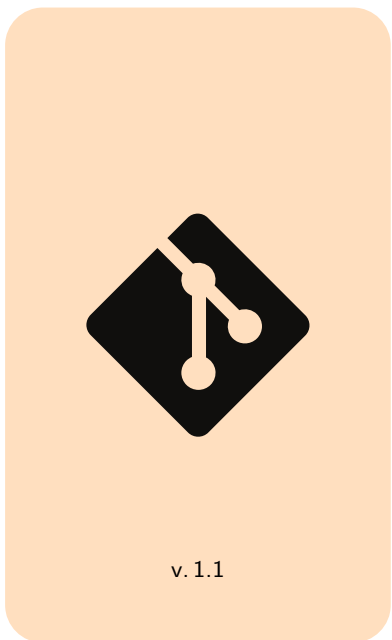
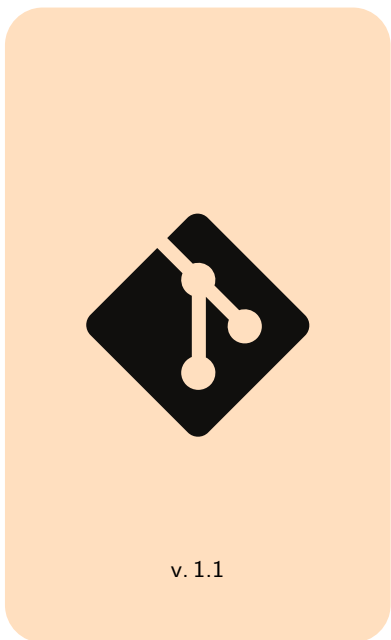
„git push!“

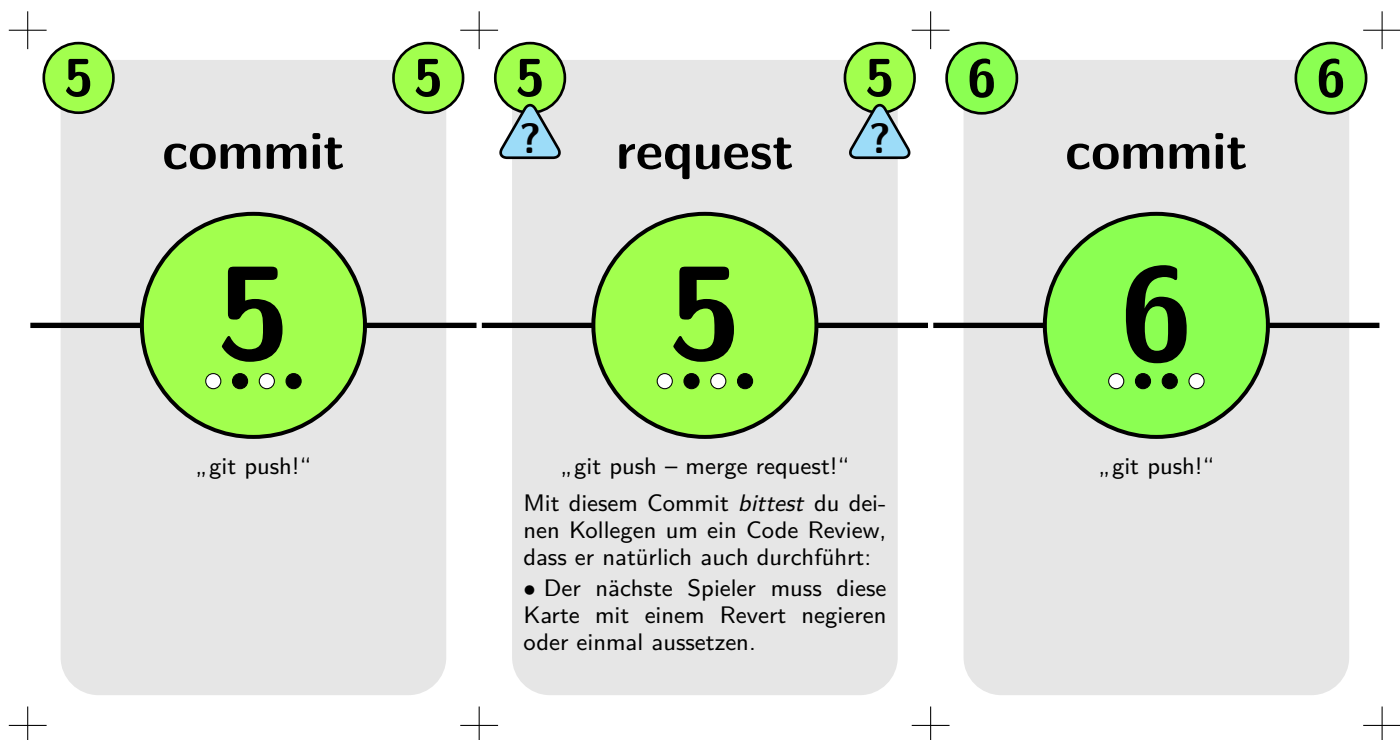
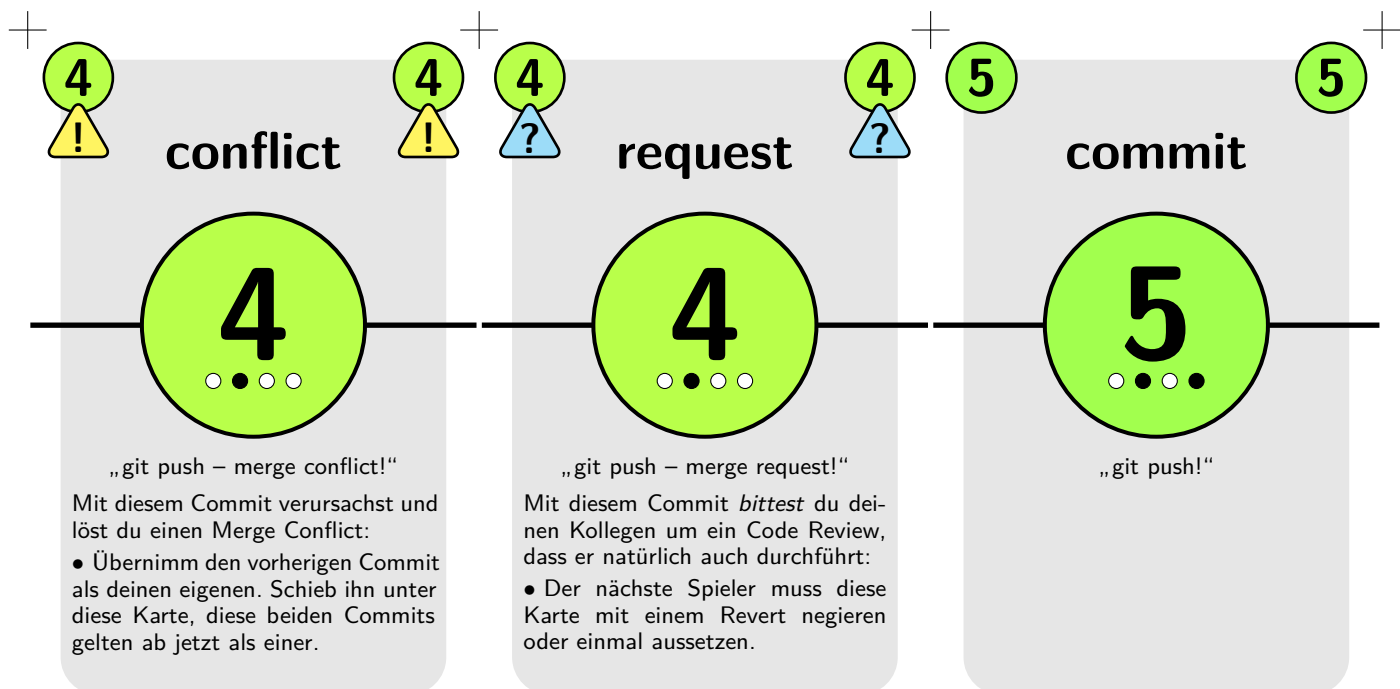
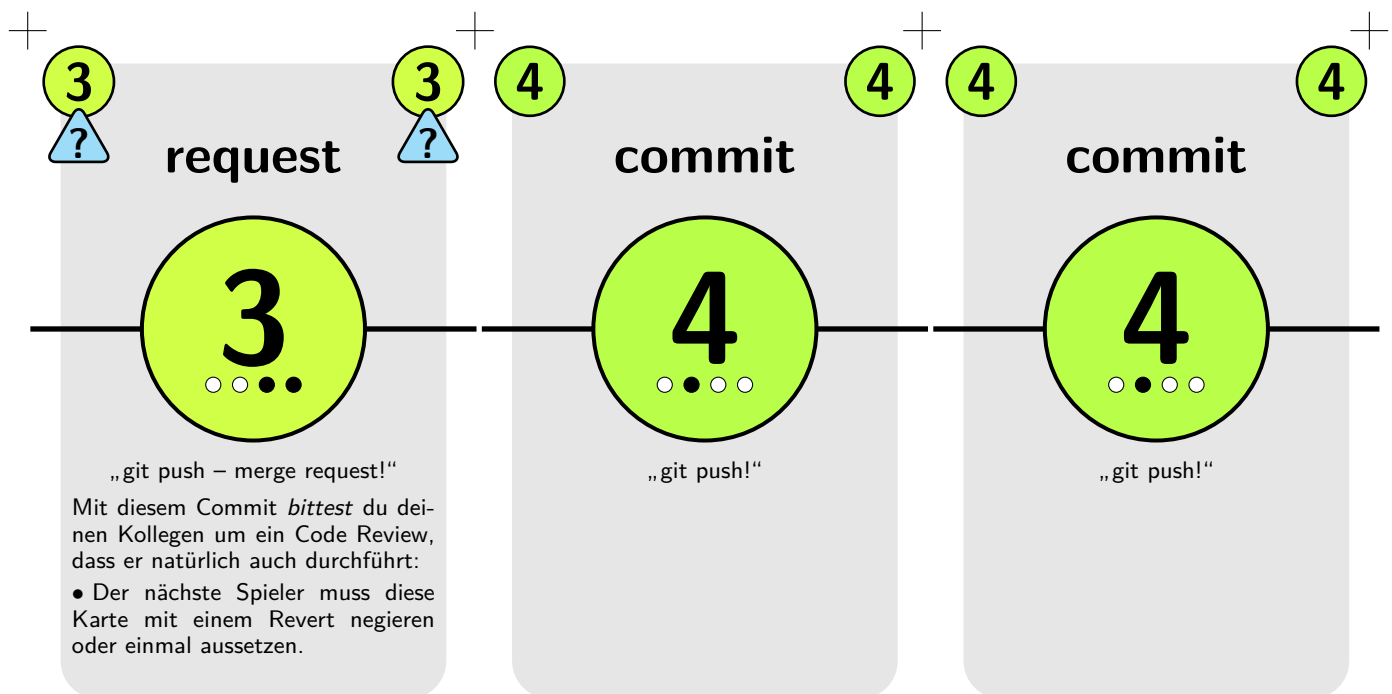
„git push!“

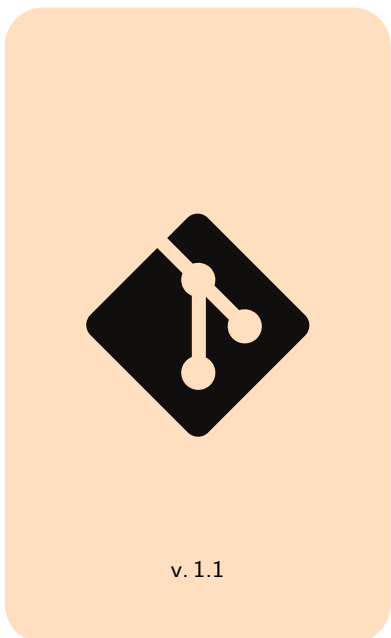
„git push – merge conflict!“

Mit diesem Commit verursachst und löst du einen Merge Conflict:

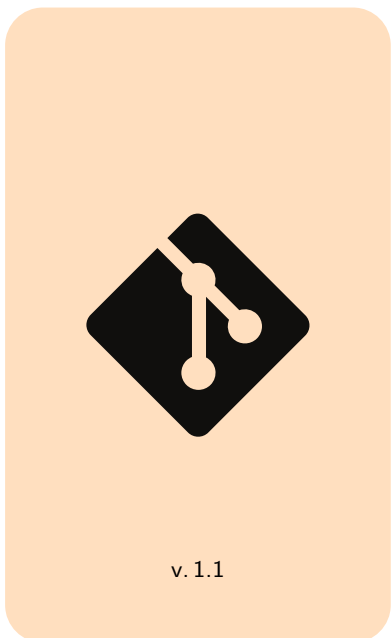
- Übernimm den vorherigen Commit als deinen eigenen. Schieb ihn unter diese Karte, diese beiden Commits gelten ab jetzt als einer.



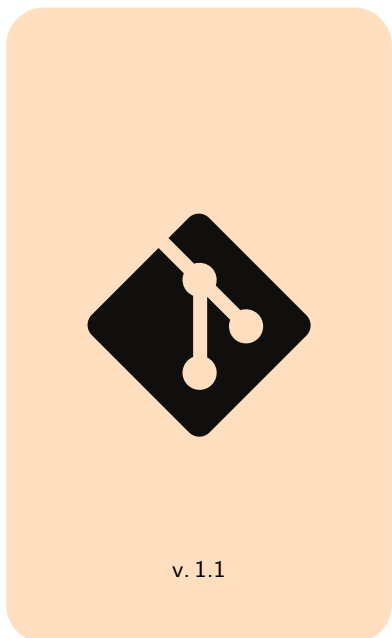




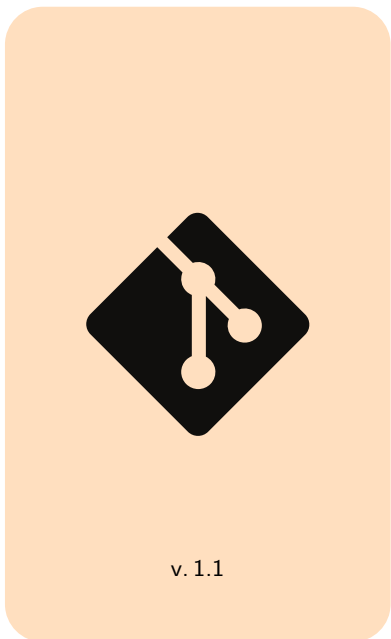
v. 1.1



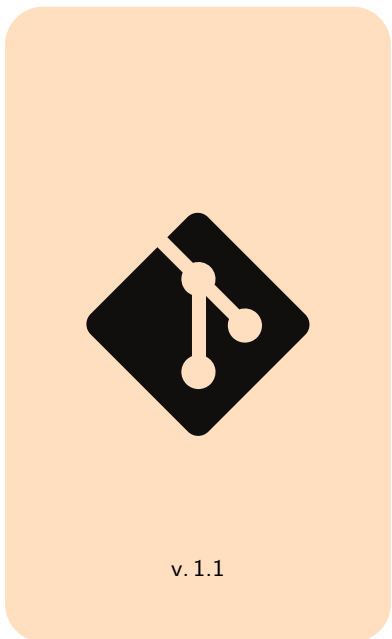
v. 1.1



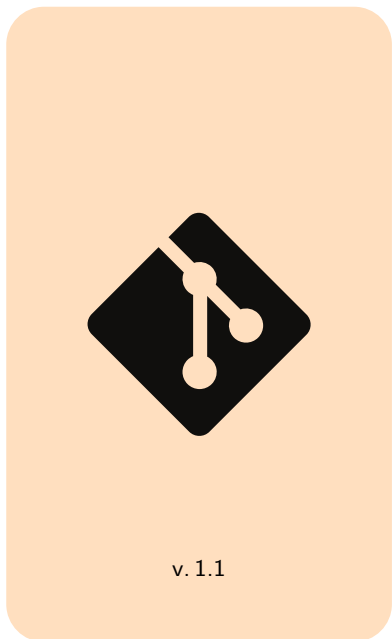
v. 1.1



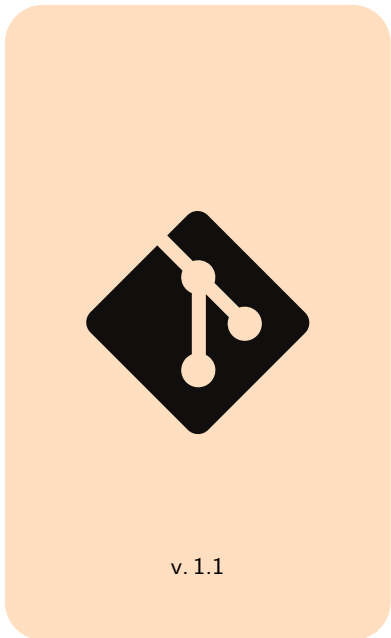
v. 1.1



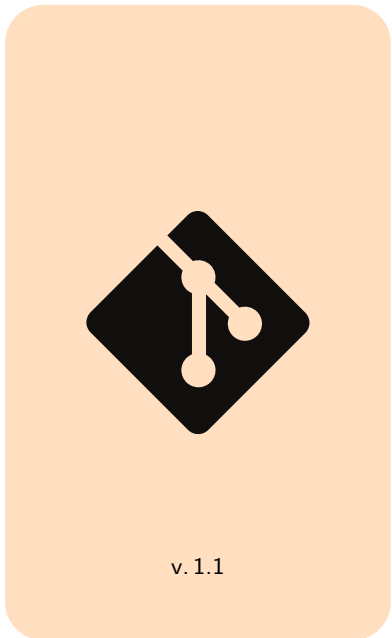
v. 1.1



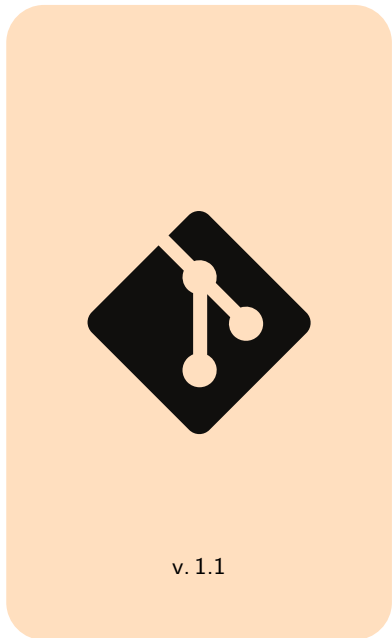
v. 1.1



v. 1.1

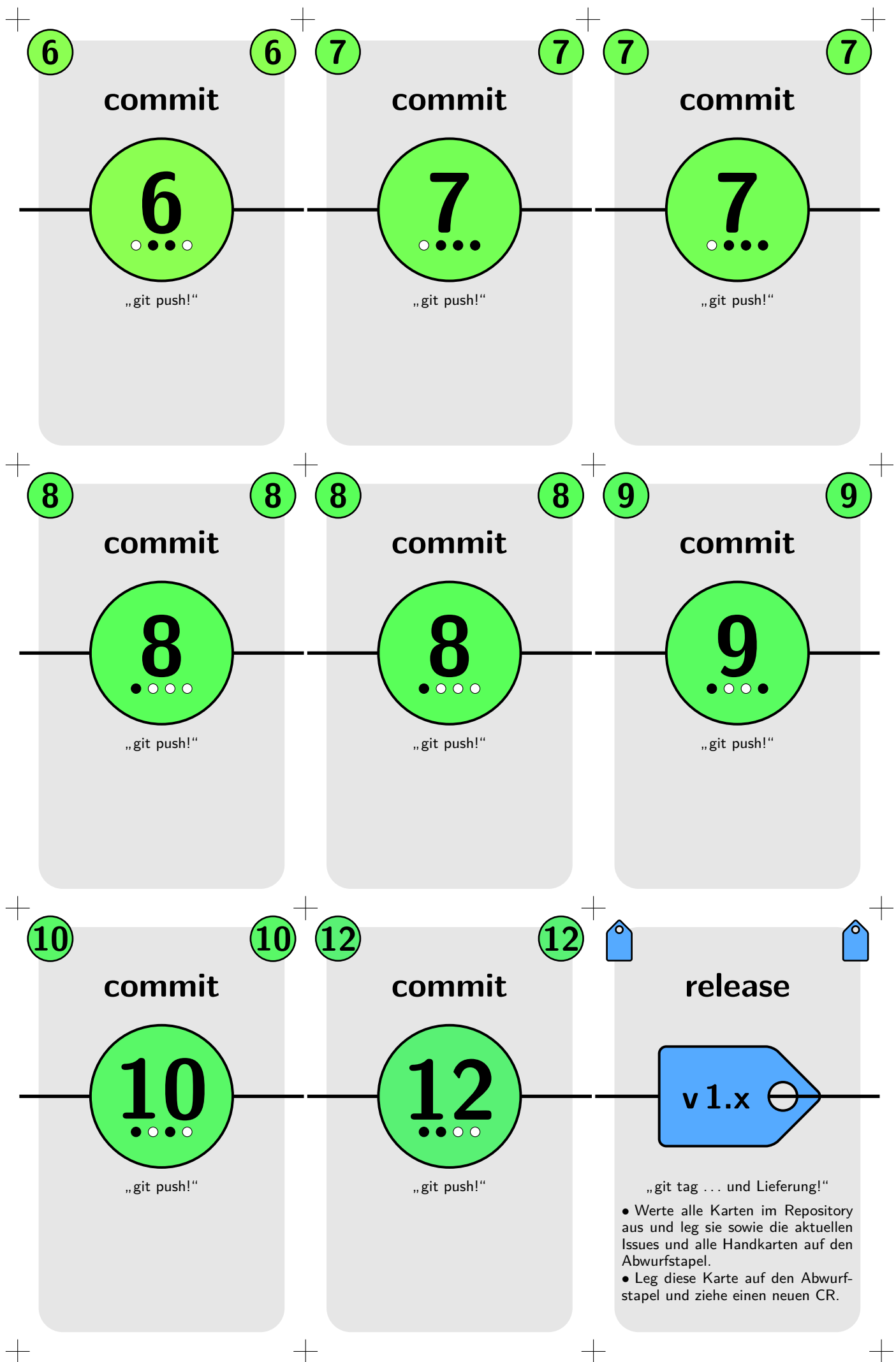


v. 1.1

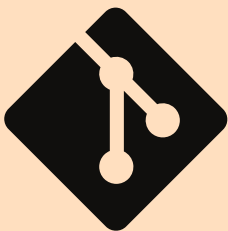


v. 1.1





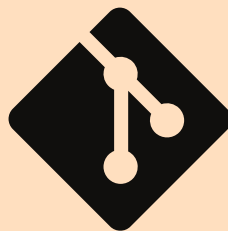




v. 1.1.1



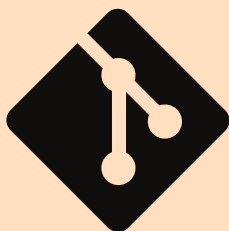
v. 1.1.1



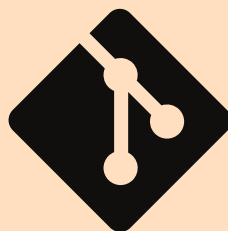
v. 1.1.1



v. 1.1.1



v. 1.1.1



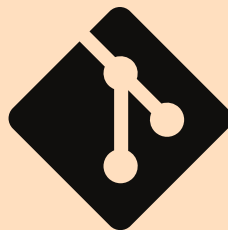
v. 1.1.1



v. 1.1.1

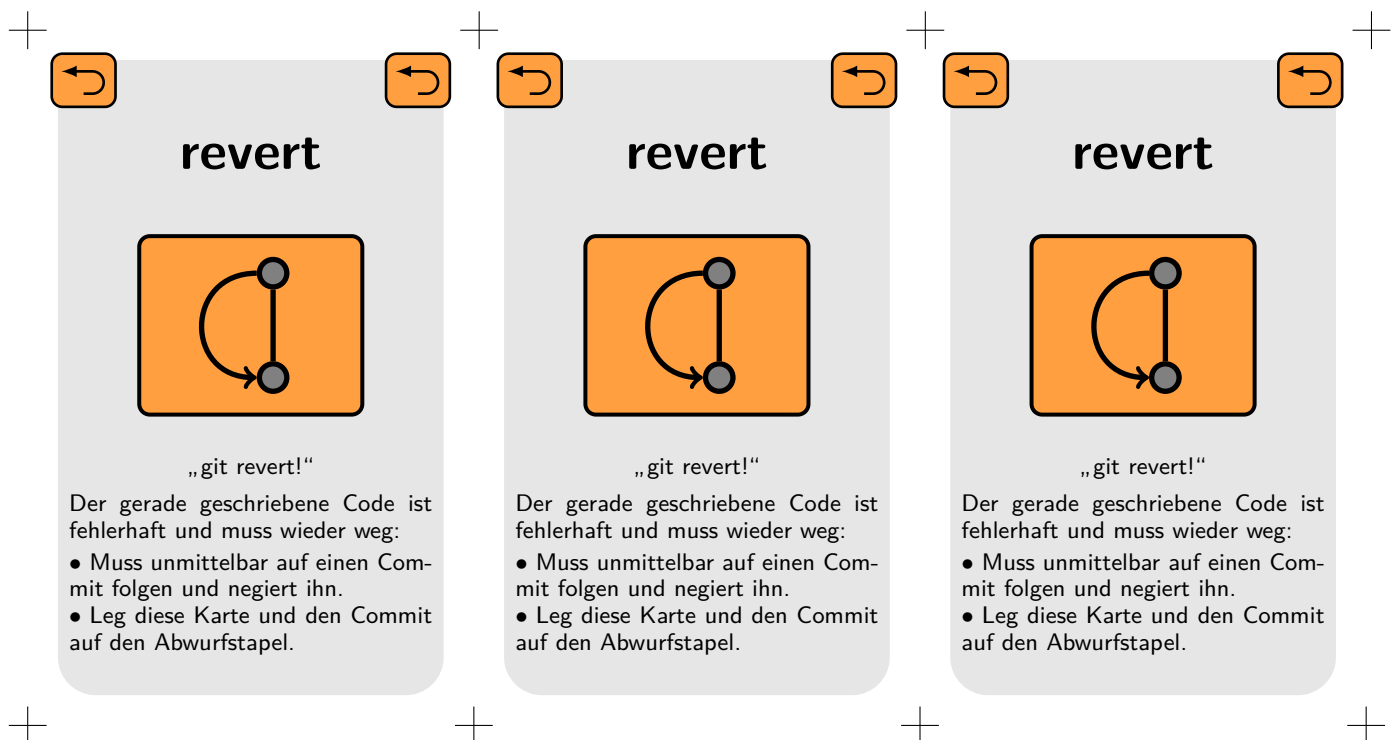
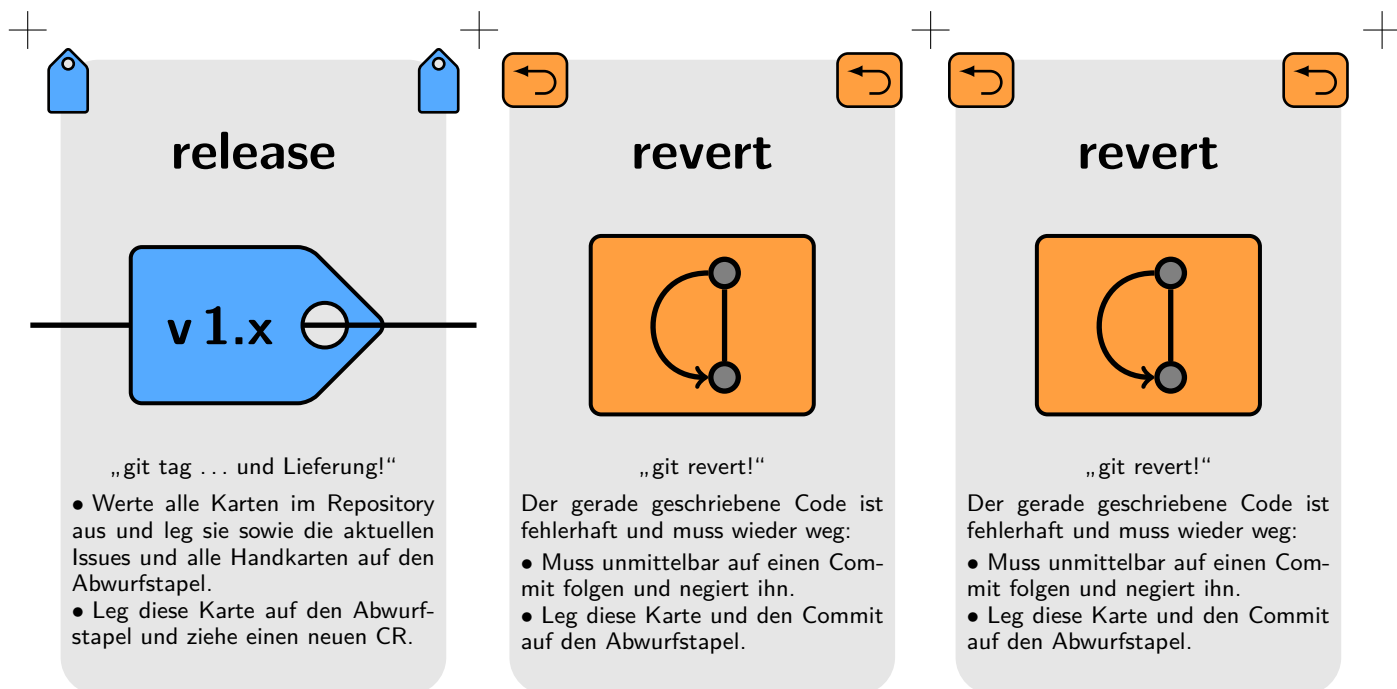
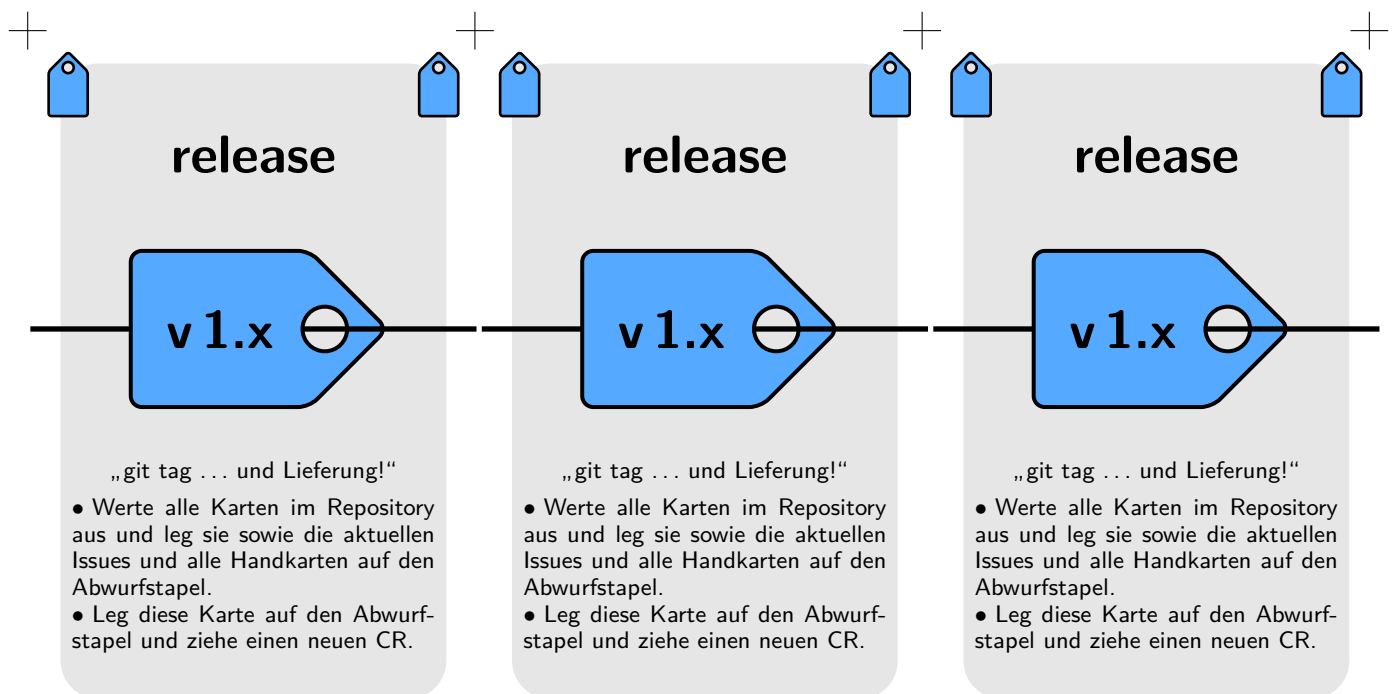


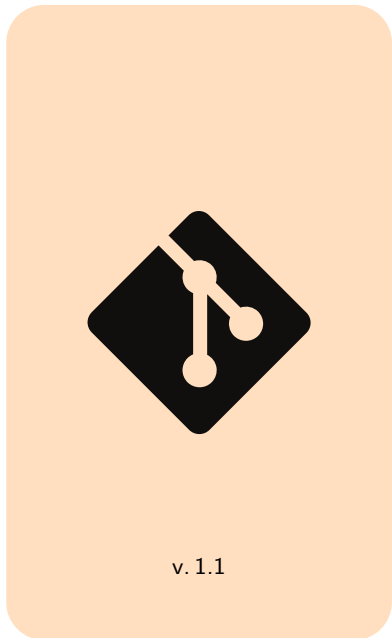
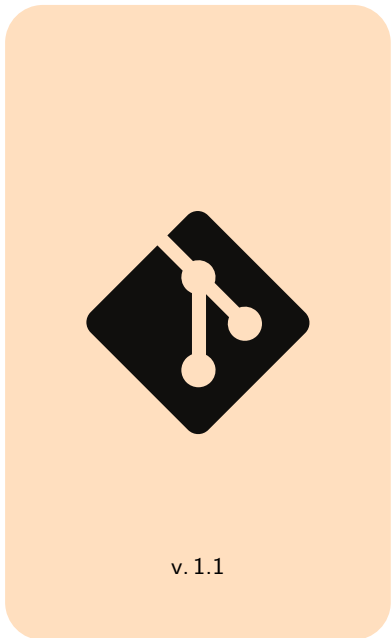
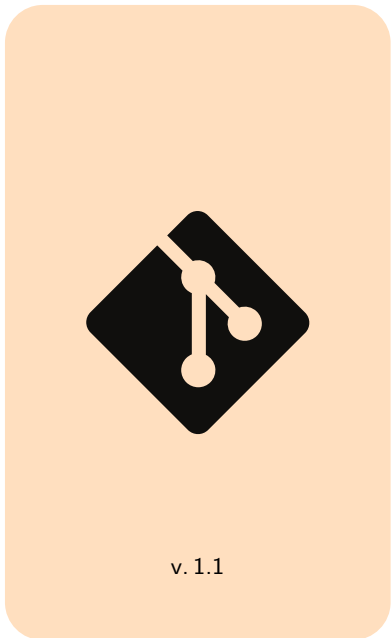
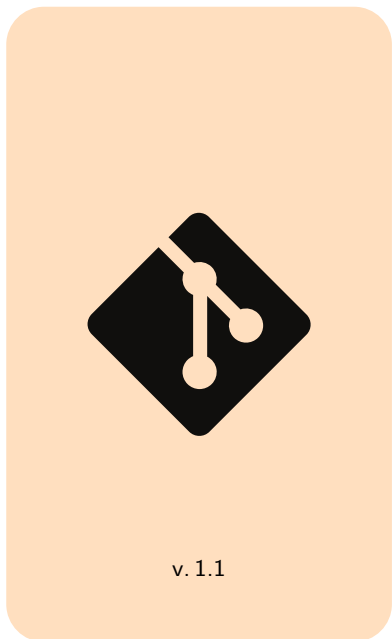
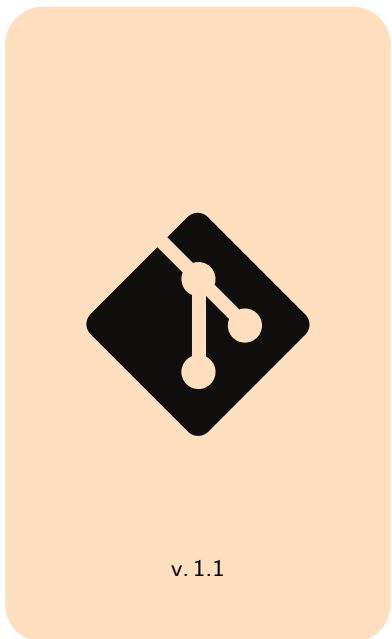
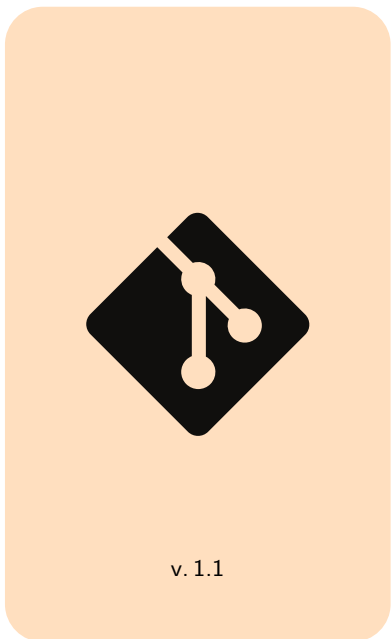
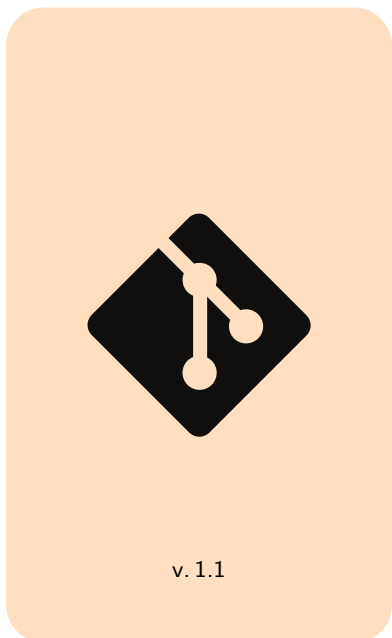
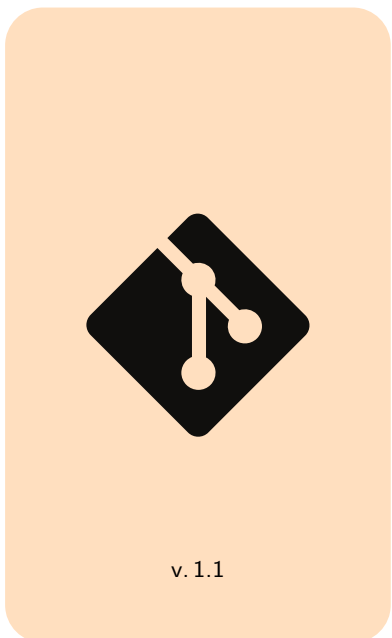
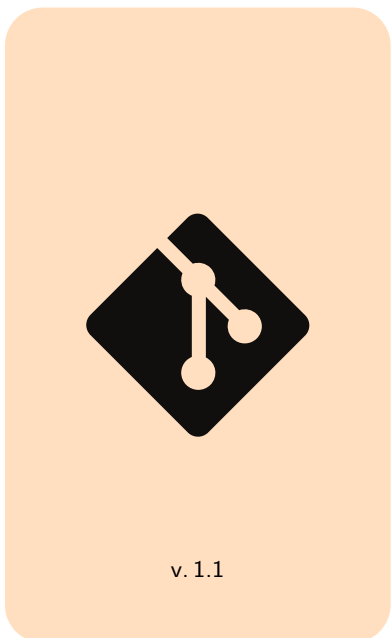
v. 1.1.1



v. 1.1.1

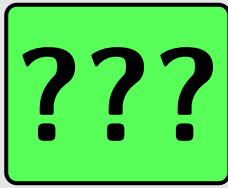






??

## help



„git help!“

Du weißt gerade nicht weiter und brauchst Hilfe:

- Leg diese Karte auf den Abwurfstapel.
- Zieh drei Handkarten zusätzlich, nachdem du **aufgestockt** hast.

??

?

## status



„git status!“

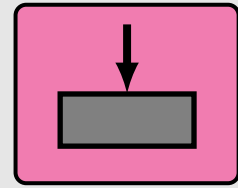
Du hast gerade keine Idee und brauchst erstmal einen Überblick:

- Leg diese Karte auf den Abwurfstapel.
- Zieh eine Handkarte zusätzlich, nachdem du **aufgestockt** hast.

?



## stash

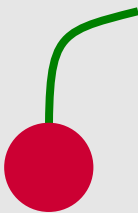


„git stash!“

- Leg bis zu zwei Handkarten verdeckt vor Dir ab.
- Sie können weiterhin wie Handkarten ausgespielt werden, zählen aber nicht beim **Aufstocken** mit und gehen bei einem Release nicht verloren.



## cherry-pick

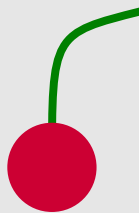


„git cherry-pick!“

- **Stocke** zuerst deine Handkarten auf.
- Wähle dir danach bei einem beliebigen Spieler aus seinen Handkarten eine aus und füge sie deinen hinzu oder spiele sie sofort aus.



## cherry-pick



„git cherry-pick!“

- **Stocke** zuerst deine Handkarten auf.
- Wähle dir danach bei einem beliebigen Spieler aus seinen Handkarten eine aus und füge sie deinen hinzu oder spiele sie sofort aus.



5

## Bug



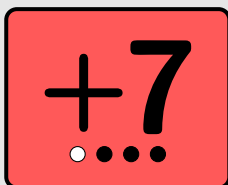
„Ich habe einen Fehler gefunden, der auch noch behoben werden muss.“

- Leg diese Karte zum aktuellen CR.
- Für die aktuelle Lieferung müssen noch 5 Codezeilen zusätzlich geschrieben werden.

5

7

## Bug



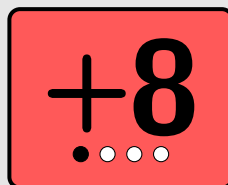
„Ich habe einen Fehler gefunden, der auch noch behoben werden muss.“

- Leg diese Karte zum aktuellen CR.
- Für die aktuelle Lieferung müssen noch 7 Codezeilen zusätzlich geschrieben werden.

7

8

## Bug



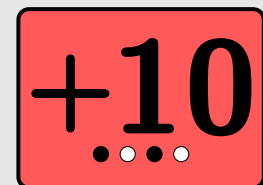
„Ich habe einen Fehler gefunden, der auch noch behoben werden muss.“

- Leg diese Karte zum aktuellen CR.
- Für die aktuelle Lieferung müssen noch 8 Codezeilen zusätzlich geschrieben werden.

8

10

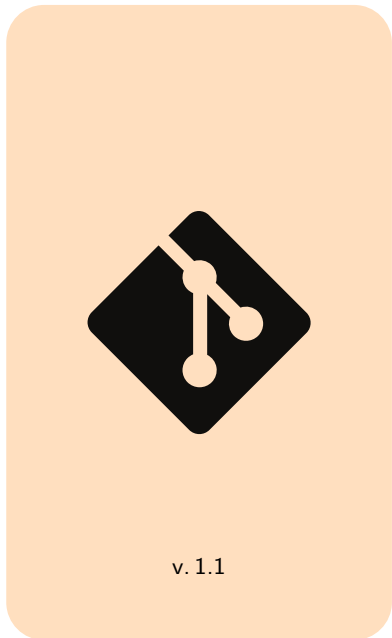
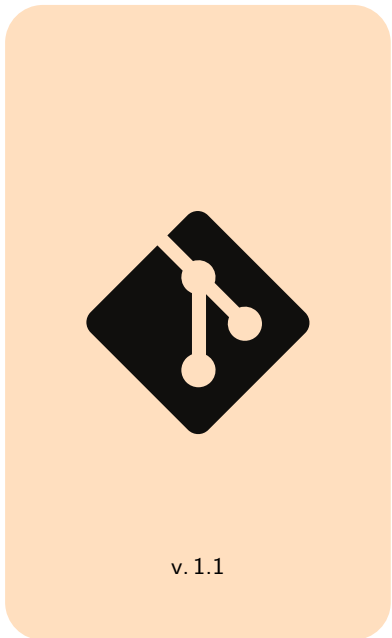
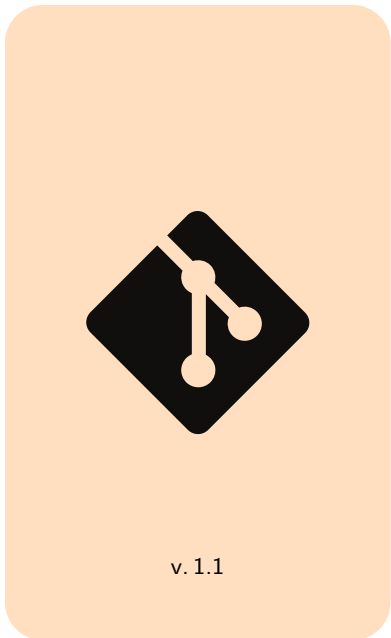
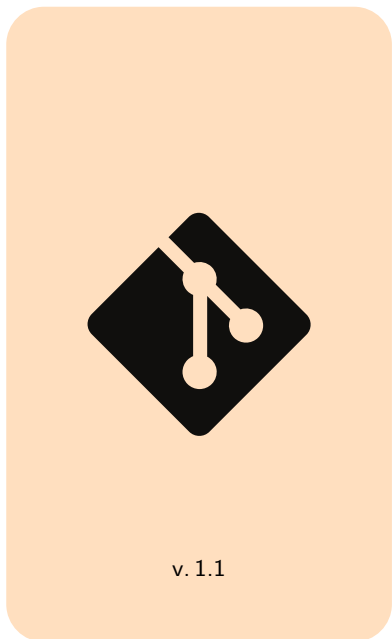
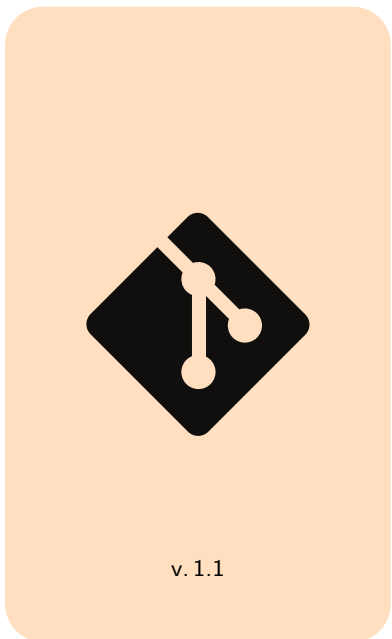
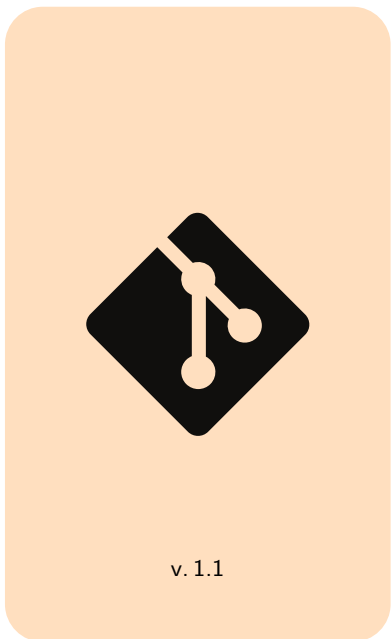
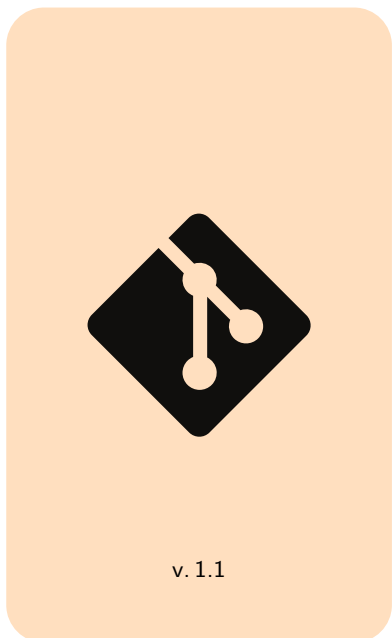
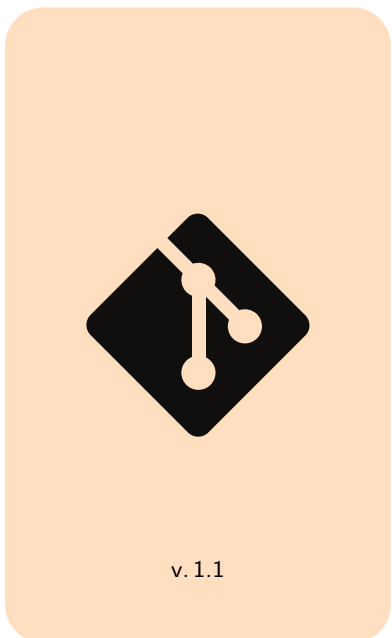
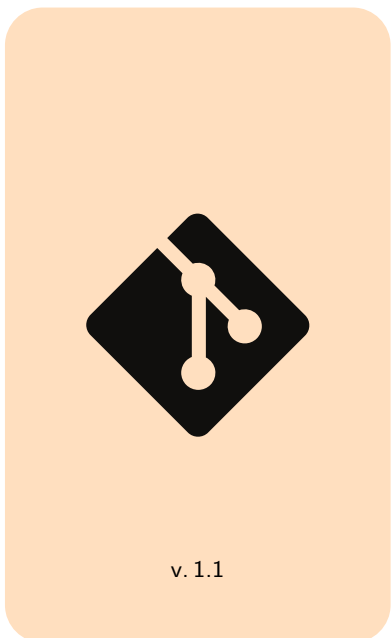
## Bug

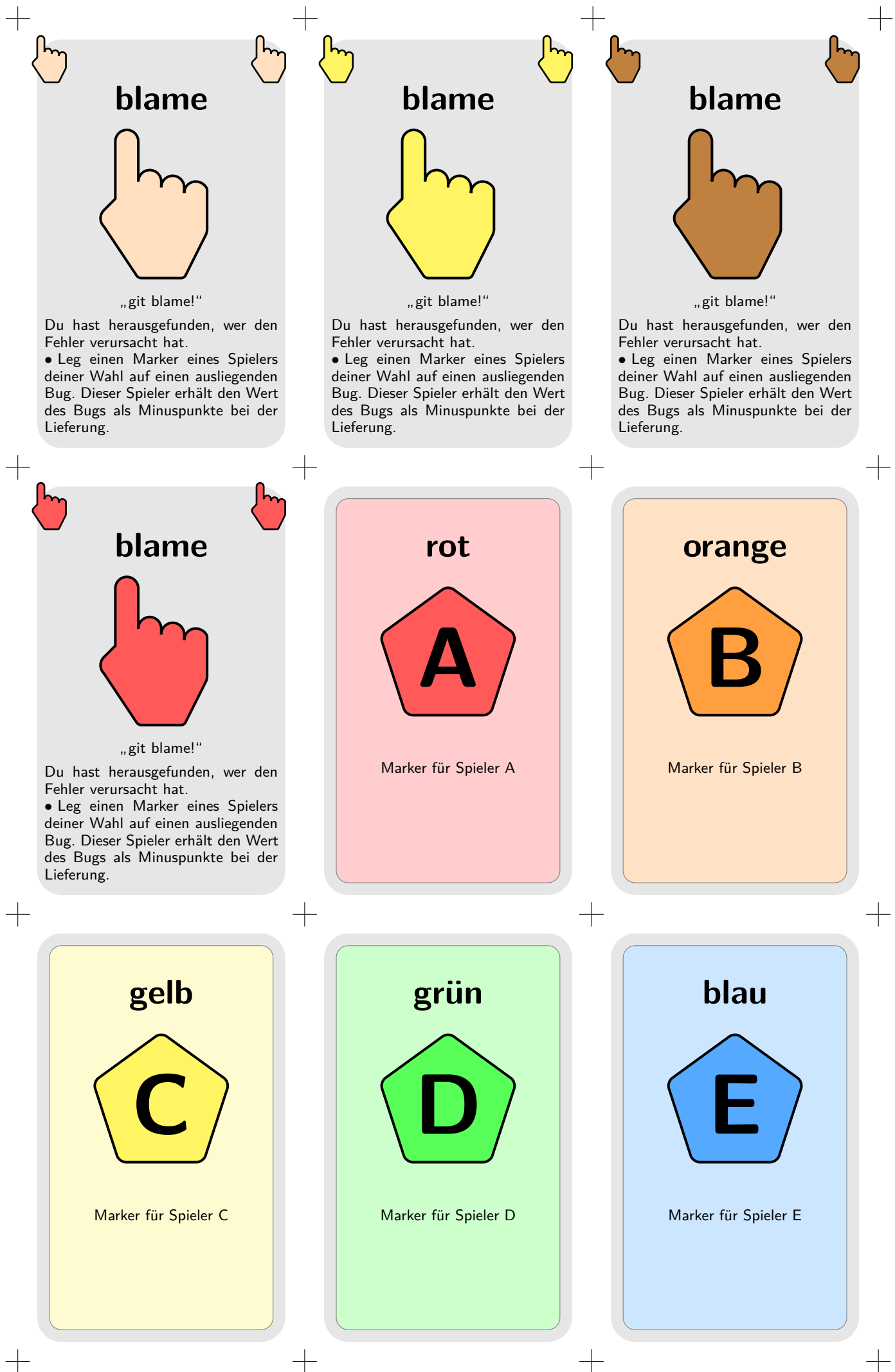


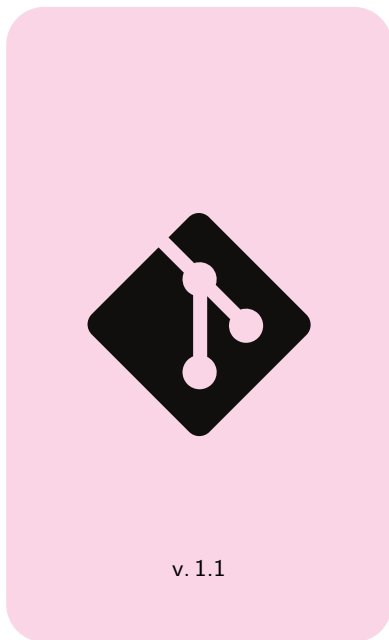
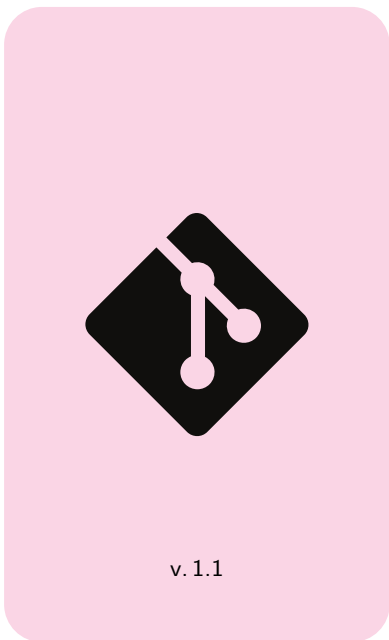
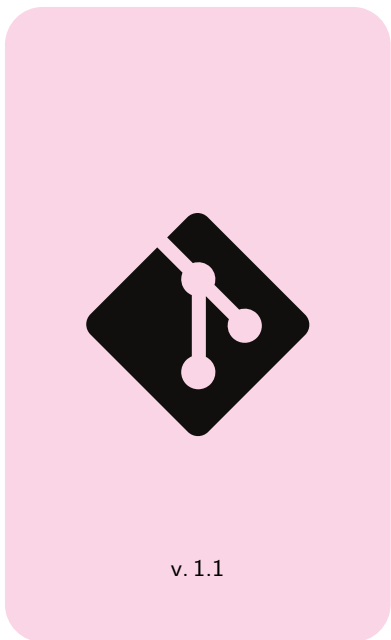
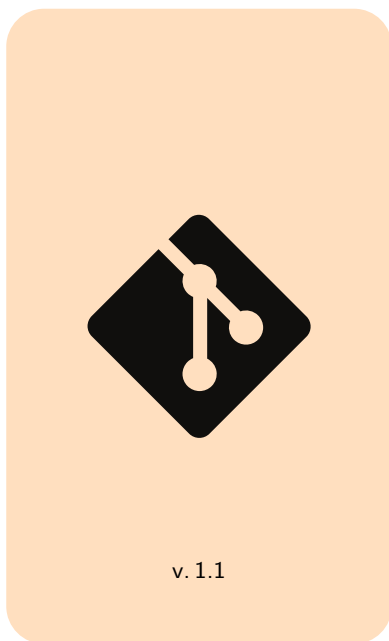
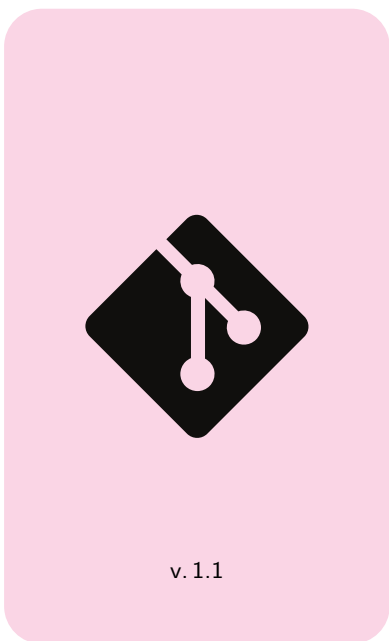
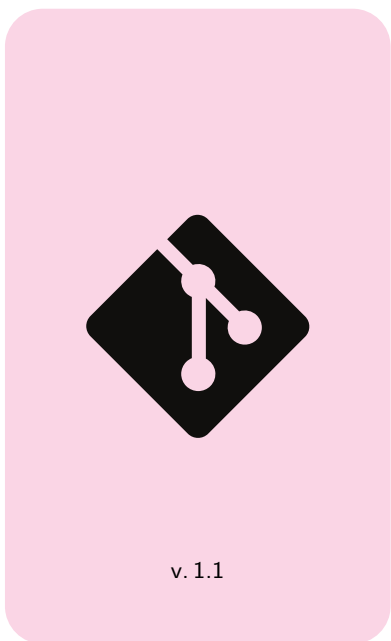
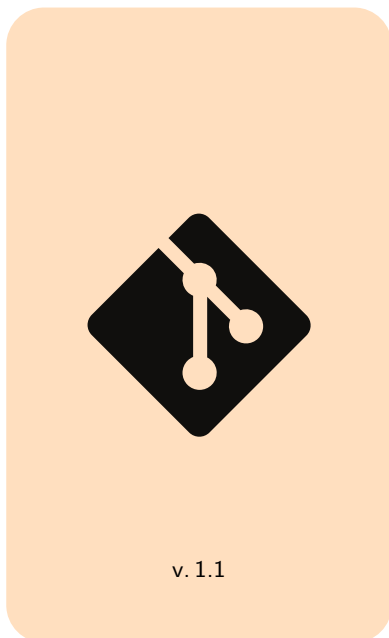
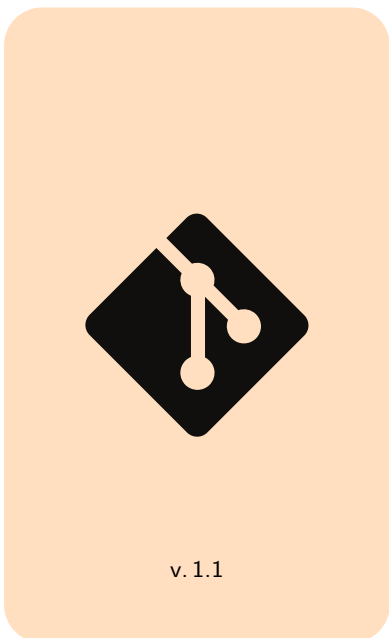
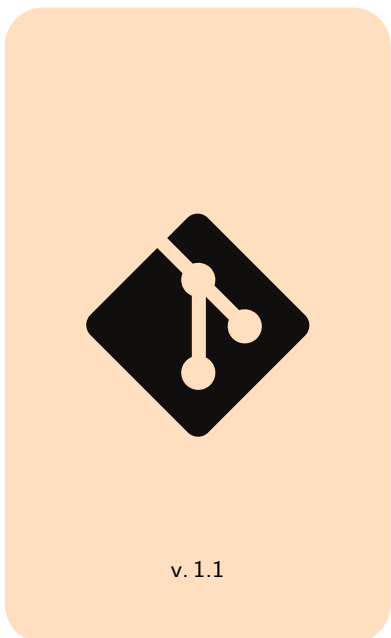
„Ich habe einen Fehler gefunden, der auch noch behoben werden muss.“

- Leg diese Karte zum aktuellen CR.
- Für die aktuelle Lieferung müssen noch 10 Codezeilen zusätzlich geschrieben werden.

10







10

Issue: CR

10  
... ..

- Für diese Lieferung müssen mindestens 10 Codezeilen geschrieben werden.
- Bei der Lieferung erhält der Auslieferer 5 Zusatzpunkte.

10

12

Issue: CR

12  
... ..

- Für diese Lieferung müssen mindestens 12 Codezeilen geschrieben werden.
- Bei der Lieferung erhält der Auslieferer 6 Zusatzpunkte.

12

15

Issue: CR

15  
... ..

- Für diese Lieferung müssen mindestens 15 Codezeilen geschrieben werden.
- Bei der Lieferung erhält der Auslieferer 7 Zusatzpunkte.

15

16

Issue: CR

16  
... ..

- Für diese Lieferung müssen mindestens 16 Codezeilen geschrieben werden.
- Bei der Lieferung erhält der Auslieferer 8 Zusatzpunkte.

16

20

Issue: CR

20  
... ..

- Für diese Lieferung müssen mindestens 20 Codezeilen geschrieben werden.
- Bei der Lieferung erhält der Auslieferer 10 Zusatzpunkte.

20

24

Issue: CR

24  
... ..

- Für diese Lieferung müssen mindestens 24 Codezeilen geschrieben werden.
- Bei der Lieferung erhält der Auslieferer 12 Zusatzpunkte.

24

26

Issue: CR

26  
... ..

- Für diese Lieferung müssen mindestens 26 Codezeilen geschrieben werden.
- Bei der Lieferung erhält der Auslieferer 13 Zusatzpunkte.

26

30

Issue: CR

30  
... ..

- Für diese Lieferung müssen mindestens 30 Codezeilen geschrieben werden.
- Bei der Lieferung erhält der Auslieferer 15 Zusatzpunkte.

30

32

Issue: CR

32  
... ..

- Für diese Lieferung müssen mindestens 32 Codezeilen geschrieben werden.
- Bei der Lieferung erhält der Auslieferer 16 Zusatzpunkte.

32



