

Git

Versionskontrolle leicht gemacht

Agenda

Was ist Git?

Grundlagen und
Terminologie

Git & SVN

Unterschiede und
Gemeinsamkeiten

Der Rest

Workflow,
Konventionen
und mehr

Arbeit mit Git

Befehle und Umgang

Tipps & Tricks

Nützliches für den
Alltag

Installation

Windows:

- ◆ Offizielle Git Seite (git-scm.com) - Für alle Systeme
oder direkt bei
- ◆ msys Git (msysgit.github.io)

Kleine Empfehlung: console2 (sourceforge.net/projects/console)

Grundlagen

Theorie, Terminologie und Basis-Befehle ...

Centralised VS Distributed

→ Centralised (Zentralisiert)

- ◆ Ein zentrales Repository
- ◆ Alle Entwickler arbeiten auf diesem
- ◆ Kommunikation:
 - Repository \Leftrightarrow Entwickler
 - Entwickler \nLeftrightarrow Entwickler

Beispiele: CVS, Subversion (SVN)

Centralised VS Distributed

→ Distributed (Verteilt)

- ◆ Kein zentrales Repository (obwohl es möglich ist)
- ◆ Alle Entwickler besitzen eine vollständige Kopie
- ◆ Kommunikation:
 - Repository \Leftrightarrow Entwickler
 - Entwickler \Leftrightarrow Entwickler

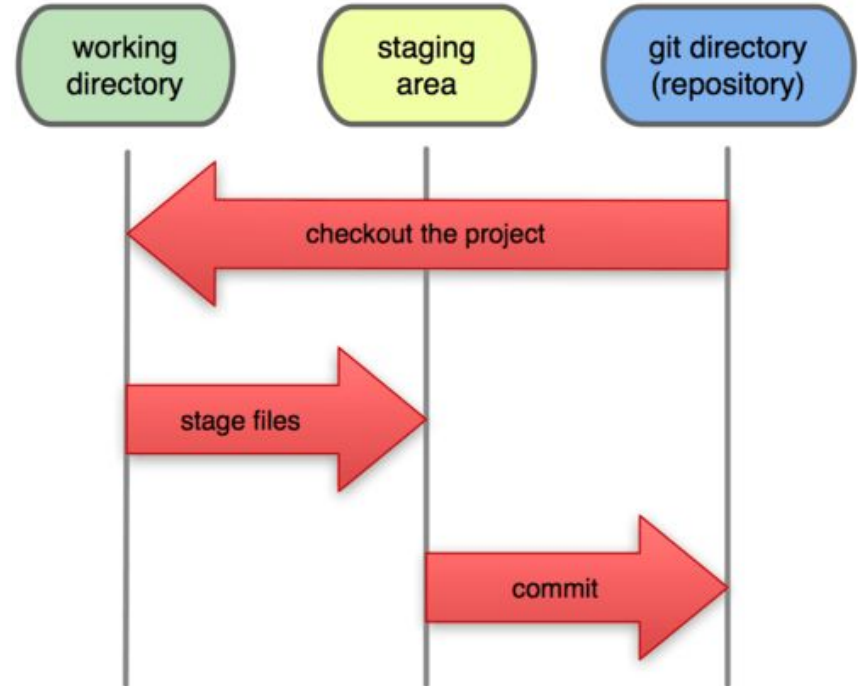
Beispiele: Git, Mercurial

Basics - Stages

Stages

- ◆ Working Directory
- ◆ Staging (Index)
- ◆ Repository

Local Operations



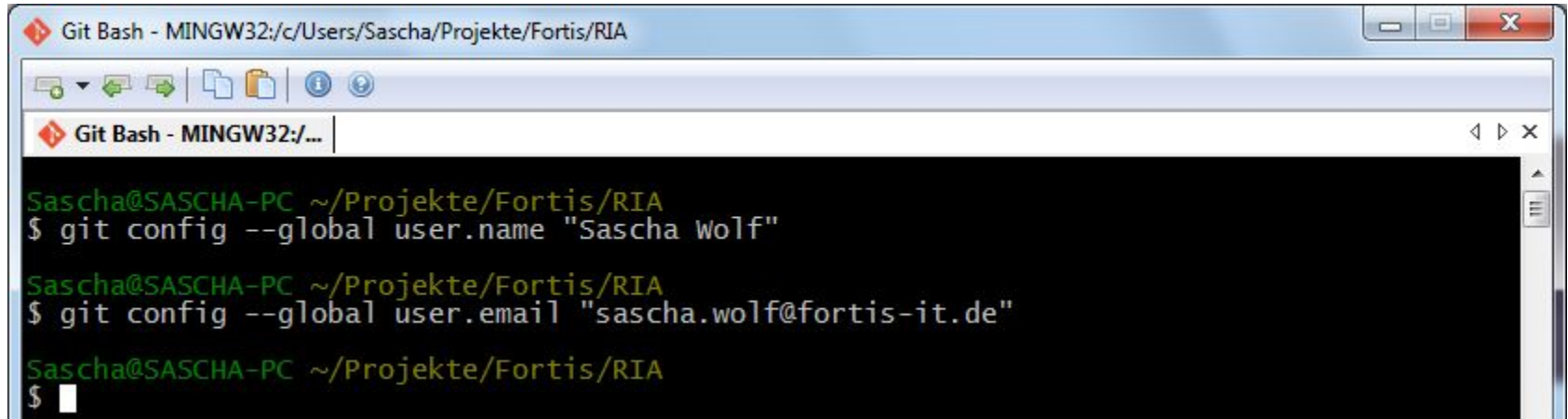
Basics - Konfiguration

Grundlegende Konfiguration

| | |
|--------------------------------|--------------------------|
| <code>config user.name</code> | Sascha Wolf |
| <code>config user.email</code> | sascha.wolf@fortis-it.de |

`git config --global`

Globale Konfiguration
(.gitconfig im HOME Verzeichnis)



```
Git Bash - MINGW32:/c/Users/Sascha/Projekte/Fortis/RIA
Git Bash - MINGW32:/...
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA
$ git config --global user.name "Sascha Wolf"
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA
$ git config --global user.email "sascha.wolf@fortis-it.de"
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA
$
```


Basics - Repositorys

| Aufgabe | Befehl |
|-------------------------------|--|
| Leeres Repository erstellen | <code>init [<directory>]</code> |
| Bestehendes Repository klonen | <code>clone <source> <target></code> |

“Viele Wege führen nach Rom”

(eigentlich nur zwei ...)

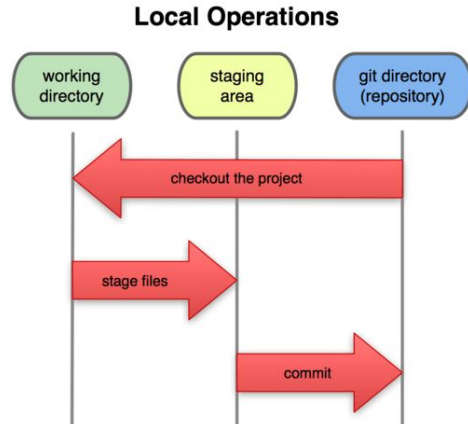
```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA
$ git init my-first-repository
Initialized empty Git repository in c:/Users/Sascha/Projekte/Fortis/RIA/my-first-repository/.git/

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA
$ git clone clone-me cloned-repository
Cloning into 'cloned-repository'...
warning: You appear to have cloned an empty repository.
done.

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA
$
```

Basics - Add & Commit

| Aufgabe | Befehl |
|----------------------|--|
| Zum Index hinzufügen | <code>add <path></code> |
| Commit erzeugen | <code>commit [-m <message>]</code> |



```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ touch hello.txt

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ echo "Hello World" > hello.txt

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git add hello.txt

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git commit -m "First Commit"
[master (root-commit) c2d2ae0] First Commit
1 file changed, 1 insertion(+)
create mode 100644 hello.txt
```

Basics - Log

Und wie kann ich mir den
Kram jetzt anschauen?

| Aufgabe | Befehl |
|----------------------------|--------------------|
| Bisherige Commits anzeigen | log |
| Genauere Informationen | --stat --summary |

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git log
commit c2d2ae0a86e68c8f085dfbeac4ab3fd89a1bb102
Author: Sascha Wolf <swolf.dev@gmail.com>
Date:   Sun Mar 16 19:54:18 2014 +0100

    First Commit

hello.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 hello.txt
```

Basics - Reset

| Aufgabe | Befehl |
|-------------------------------------|--|
| Eine Datei aus dem Index entfernen | <code>reset <path></code> |
| Änderungen entfernen / zurücksetzen | <code>checkout <path> reset --hard <path></code> |

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my
$ git status -s
M hello.txt

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my
$ git reset hello.txt
Unstaged changes after reset:
M hello.txt

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my
$ git status -s
M hello.txt
```

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my
$ git status -s
M hello.txt

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my
$ git checkout hello.txt

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my
$ git status -s

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my
$
```

Basics - Diff

Und wenn ich gar nicht
weiß was genau drin steht?

| Aufgabe | Befehl |
|-------------------------|---|
| Unterschiede anzeigen | <code>diff <path></code> |
| Indexierte Unterschiede | <code>diff --cached <path></code> |

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (feature)
$ git status -s
 M hello.txt
 M test/test.txt

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (feature)
$ git diff
diff --git a/hello.txt b/hello.txt
Hello World
+Ich bin neu :)

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (feature)
$ git diff --cached
diff --git a/test/test.txt b/test/test.txt
Test
+Ich auch :D
```

Basics - “Navigation”

| Aufgabe | Befehl |
|----------------------------------|---|
| Parent Commit wählen | <commit>^ |
| Parent des Parent des Parent ... | <commit>^^^ <commit>~3 <commit>^~1^ |

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (master)
$ git log --oneline -1
05d28d7 Third Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (master)
$ git log --oneline -1 master^
32ef256 Second Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (master)
$ git log --oneline -1 master~2
60081e0 First Commit
```

Basics - Branching

Branching in Git

“Unlike many other VCSs, Git encourages a workflow that branches and merges often, even multiple times in a day.”

Scott Chacon, Progit

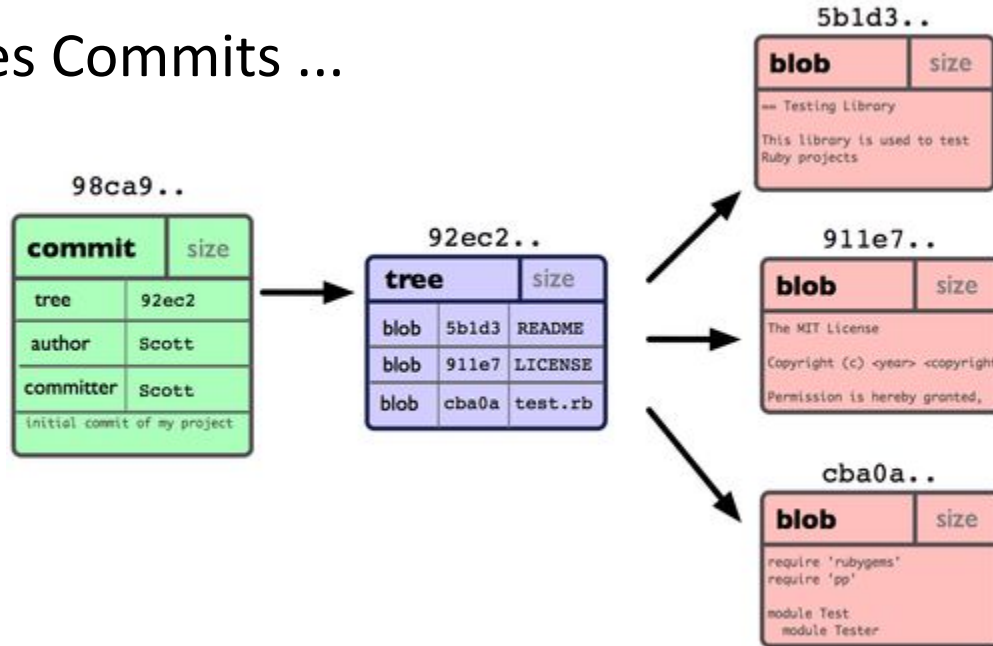
Basics - Repository

Git Objekte

- ◆ BLOB
 - Dateiinhalte
- ◆ Tree
 - Verzeichnis Repräsentation (Referenzen auf BLOBs und Trees)
- ◆ Commit
 - Commit Metadaten (Author, Committer, Message, Parent-Commit, zugehöriger Tree)
- ◆ Tag
 - Markierung von Commits

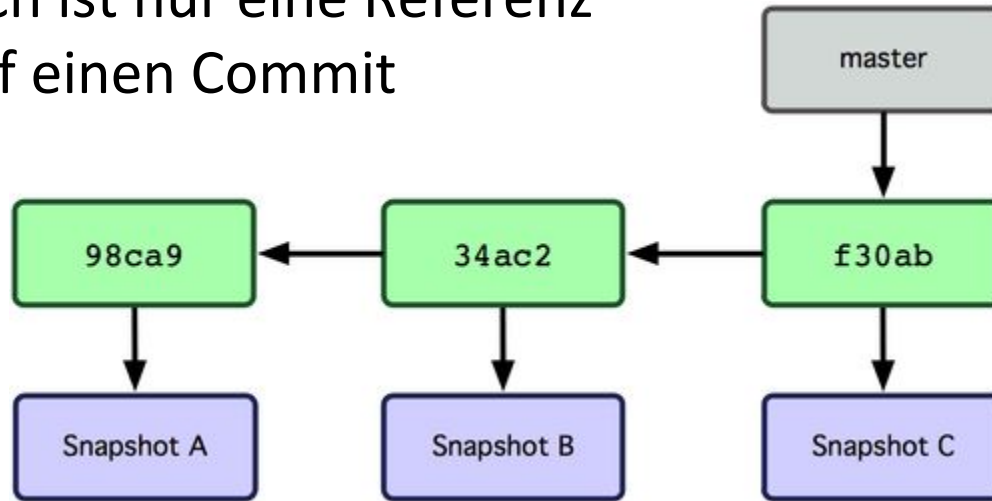
Basics - Repository

Aufbau eines Commits ...



Basics - Repository

Eine Branch ist nur eine Referenz auf einen Commit



Basics - Branching

| Aufgabe | Befehl |
|--------------------------|------------------------|
| Branches anzeigen | branch |
| Branch erzeugen | branch <name> |
| Zu einer Branch wechseln | checkout <branch-name> |

Doch wie erzeuge ich jetzt
eine Branch?

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git branch feature

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git checkout feature
Switched to branch 'feature'

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (feature)
$ git branch
* feature
  master
```

Basics - Log

| Aufgabe | Befehl |
|-------------------------|---|
| Alle commits anzeigen | <code>log --all</code> |
| Branch Graph darstellen | <code>log --all --graph</code> |
| Graph mit Branch Infos | <code>log --all --graph --decorate</code> |

Git logs it your way!

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-ft
$ git log --oneline
da8be0a Feature commit
c2d2ae0 First Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-ft
$ git log --oneline --all
da8be0a Feature commit
36760ec Second commit
c2d2ae0 First Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-ft
$ git log --oneline --all --graph
* da8be0a Feature commit
* 36760ec Second commit
* c2d2ae0 First Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-ft
$ git log --oneline --all --graph --decorate
* da8be0a (HEAD, feature) Feature commit
* 36760ec (master) Second commit
* c2d2ae0 First Commit
```

Basics - Alias

Muss ich das jetzt
jedes Mal eintippen?

| Aufgabe | Befehl |
|--------------------------------|--|
| Alias für einen Befehl anlegen | <code>config alias.<alias> <befehl></code> |

Nicht die `--global` Option vergessen!

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (feature)
$ git config --global alias.lag "log --graph --all --oneline --decorate"

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (feature)
$ git lag
* da8be0a (HEAD, feature) Feature commit
* 36760ec (master) Second commit
* c2d2ae0 First Commit
```

Basics - Merge

| Aufgabe | Befehl |
|----------------------------------|----------------|
| Branch A in Branch B integrieren | merge <branch> |

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git merge feature
Merge made by the 'recursive' strategy.
 test/test.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 test/test.txt

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git log
* e548631 (HEAD, master) Merge branch 'feature'
* da8be0a (feature) Feature commit
* | 36760ec Second commit
* |
* c2d2ae0 First Commit
```

Basics - Merge

Und wenn mal etwas schiefgeht?

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git lag
* 4264bcd (feature) Conflicting commit
* da8be0a Feature commit
* 36760ec (HEAD, master) Second commit
* c2d2ae0 First Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git merge feature
Auto-merging hello.txt
CONFLICT (content): Merge conflict in hello.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Basics - HEAD

Eine Referenz auf den
ausgecheckten Commit

Alles Kopfsache ...

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git log
* da8be0a (feature) Feature commit
* 36760ec (HEAD, master) Second commit
* c2d2ae0 First Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ cat .git/HEAD
ref: refs/heads/master
```


Basics - Remote

And up you go ... !

| Aufgabe | Befehl |
|------------------------------|---|
| Remote Repository hinzufügen | <code>remote add <name> <url></code> |
| Änderungen schicken | <code>push [<remote>] [<branch>]</code> |
| Änderungen erhalten | <code>pull [<remote>] [<branch>]</code> |

Basics - Remote

```
Wolf@WOLF-THINK ~/Projekte/Fortis/my-first-project (another-feature)
$ git remote add origin ../my-first-remote.git

Wolf@WOLF-THINK ~/Projekte/Fortis/my-first-project (another-feature)
$ git push --all
Counting objects: 17, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (17/17), 1.39 KiB | 0 bytes/s, done.
Total 17 (delta 2), reused 0 (delta 0)
To ../my-first-remote.git
 * [new branch]      another-feature -> another-feature
 * [new branch]      feature -> feature
 * [new branch]      master -> master
```

Basics+ - Patches

| Aufgabe | Befehl |
|------------------|--|
| Patch generieren | <code>format-patch <commit></code> |

Wer braucht schon einen Server ...

```
Wolf@WOLF-THINK ~/Projekte/Fortis/my-first-project (master)
$ git format-patch -1 --stdout
From ecfe07f9b88f26533d931d29603d87e27f257be1 Mon Sep 17 00:00:00 2001
From: Sascha Wolf <swolf.dev@gmail.com>
Date: Fri, 21 Mar 2014 15:37:55 +0100
Subject: [PATCH] Second commit

---
 hello.txt | 1 +
 1 file changed, 1 insertion(+)

diff --git a/hello.txt b/hello.txt
```

Basics+ - Patches

| Aufgabe | Befehl |
|-----------------------|--------------------------------------|
| Patch per Mail senden | <code>send-mail <patch></code> |

Default SMTP Server etc.
per `.gitconfig` definierbar

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (master)
$ git send-email --to sascha.wolf@fortis-it.de *patch
0001-Third-Commit.patch
(mbox) Adding cc: Sascha Wolf <swolf.dev@gmail.com> from line 'From:
<swolf.dev@gmail.com>'

From: Sascha Wolf <swolf.dev@gmail.com>
To: sascha.wolf@fortis-it.de
Cc: Sascha Wolf <swolf.dev@gmail.com>
Subject: [PATCH] Third Commit
Date: Mon, 24 Mar 2014 21:09:56 +0000
Message-Id: <1395695396-4932-1-git-send-email-swolf.dev@gmail.com>
X-Mailer: git-send-email 1.8.4.msysgit.0

Send this email? ([y]es|[n]o|[q]uit|[a]ll):
```

Unter Windows ist zusätzlich MSMTP zum senden erforderlich ...

Basics+ - Patches

| Aufgabe | Befehl |
|----------------|-----------------------|
| Patch anwenden | am <path> (files dir) |

Fehlt <path> wird
stdin verwendet

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (master)
$ git lg
* 32ef256 (HEAD, master) Second Commit
* 60081e0 First Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (master)
$ git am 0001-Third-Commit.patch
Applying: Third Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (master)
$ git lg
* 3a0fe90 (HEAD, master) Third Commit
* 32ef256 Second Commit
* 60081e0 First Commit
```

Git & SVN

Ein Crashkurs ...

Git & SVN

Git als SVN Client ...

`svn dcommit` schreibt
Commits neu und fügt
eine `git-svn-id` hinzu

| Aufgabe | Befehl |
|-----------------------------------|------------------------------------|
| SVN Repository in Git importieren | <code>svn clone <url></code> |
| Lokale commits in SVN übernehmen | <code>svn dcommit</code> |
| Änderungen vom Server holen | <code>svn rebase</code> |

Es gilt

- Vermeide Merge-Commits – verwende stattdessen rebase
- Ändere Commits nach `dcommit` nicht mehr
- Verwende kein Remote Git Repository parallel – falls doch pushe nur Commits die bereits eine `git-svn-id` haben

Tipps & Tricks

Nützliches für den Alltag ...

Tipps & Tricks - Amend

| Aufgabe | Befehl |
|----------------------|---------------------------------------|
| Commit aktualisieren | <code>commit --amend</code> |
| Nachricht behalten | <code>commit --amend --no-edit</code> |

Oh verdammt! Da fehlt etwas!

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git log
* da8be0a (feature) Feature commit
* 36760ec (HEAD, master) Second commit
* c2d2ae0 First Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git add hello.txt

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-repository (master)
$ git commit --amend --no-edit
[master 40e8edf] Second commit
1 file changed, 2 insertions(+)
```

Tipps & Tricks - Interactive

Eigentlich will ich nur einen Teil ...

| Aufgabe | Befehl |
|--------------------|-----------------------------------|
| Partieller Commit | <code>add --patch -p</code> |
| Interaktives adden | <code>add --interactive -i</code> |

Tipps & Tricks - Interactive

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/GitConflictResolver (replace_empty)
$ git status -s
M GitConflictResolver.py

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/GitConflictResolver (replace_empty)
$ git add -p
diff --git a/GitConflictResolver.py b/GitConflictResolver.py
index a8916ab..d62ebbf 100644
--- a/GitConflictResolver.py
+++ b/GitConflictResolver.py
@@ -125,6 +125,7 @@ def highlightConflictGroup(view, group):
        for subregion in region:
            highlight_regions.append(subregion)
+           # Dies ist eine weitere Zeile!

        view.erase_regions("GitConflictRegion_"+group)
        view.add_regions(
Stage this hunk [y,n,q,a,d,/,e,?]? █
```

Tipps & Tricks - Interactive

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/GitConflictResolver (replace_empty)
$ git status -s
M  GitConflictResolver.py
M  README.md

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/GitConflictResolver (replace_empty)
$ git add -i

      staged      unstaged path
1:    unchanged    +3/-0  GitConflictResolver.py
2:    unchanged    +1/-0  README.md

*** Commands ***
1: status      2: update      3: revert      4: add untracked
5: patch      6: diff        7: quit        8: help
what now> █
```

Tipps & Tricks - Stash

| Aufgabe | Befehl |
|--------------------------|-------------------|
| Arbeit zwischenspeichern | stash [save] |
| Stash zurückholen | stash apply pop |
| Stash löschen | stash drop |

Husch, husch! Ab ins Körbchen!

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/GitConflictResolver (replace_empty)
$ git status -s
M Main.sublime-menu
M README.md

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/GitConflictResolver (replace_empty)
$ git stash save
Saved working directory and index state WIP on replace_empty: 5ddce38 [Plugin] R
eplaces also if the area is empty; maybe add configuration for this?
HEAD is now at 5ddce38 [Plugin] Replaces also if the area is empty; maybe add co
nfiguration for this?

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/GitConflictResolver (replace_empty)
$ git status -s
```

Tipps & Tricks - Rebase

“All your base are belong to us!”

Nicht vergessen!
rebase ändert SHA1 Keys!

| Aufgabe | Befehl |
|--------------------------------------|---|
| Arbeit auf andere Branch aufsetzen | <code>rebase <new-root> [<target-branch>]</code> |
| Bestimmte Commit Range “verschieben” | <code>rebase --onto <new-root> <from-commit> <to-commit></code> |

Tipps & Tricks - Rebase

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (another_feature)
$ git log
* 5f5f034 (HEAD, another_feature) Great Feature Commit
* 338ae65 (feature) Feature Commit
|
* 32ef256 (master) Second Commit
|
* 60081e0 First Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (another_feature)
$ git rebase --onto master feature another_feature
First, rewinding head to replay your work on top of it...
Applying: Great Feature Commit

Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (another_feature)
$ git log
* 86b4143 (HEAD, another_feature) Great Feature Commit
* 32ef256 (master) Second Commit
|
* 338ae65 (feature) Feature Commit
|
* 60081e0 First Commit
```

Tipps & Tricks - Rebase

| Aufgabe | Befehl |
|--------------------------------|--------------------------------------|
| Einen älteren Commit ändern | <code>rebase --interactive -i</code> |
| Eine Reihe von Commits ändern | <code>rebase --interactive -i</code> |
| Mehrere Commits zusammenfassen | <code>rebase --interactive -i</code> |

“Ein Befehl, sie zu knechten, sie alle zu finden, ins Dunkel zu treiben und ewig zu binden ...”

```
Sascha@SASCHA-PC ~/Projekte/Fortis/RIA/my-first-project (another_feature)
$ git rebase -i HEAD~2
[detached HEAD ad510a4] Second Commit -- rebased
1 file changed, 1 insertion(+), 1 deletion(-)
[detached HEAD c255a20] Great Feature Commit -- rebased
1 file changed, 1 insertion(+)
create mode 100644 great_feature.txt
Successfully rebased and updated refs/heads/another_feature.
```


Tipps & Tricks - Cherry-pick

| Aufgabe | Befehl |
|-----------------------------|---|
| Einzelnen commit übernehmen | <code>cherry-pick <commit></code> |

Just the good stuff ...

```
Wolf@WOLF-THINK ~/Projekte/Fortis/my-first-project (another-feature)
$ git log
* 3852ac5 (feature) Second feature commit
* 4b14376 Feature commit
| * be6421e (HEAD, another-feature) Another Feature commit
| * ecfe07f (master) Second commit
|/
* 5fcd243 First Commit

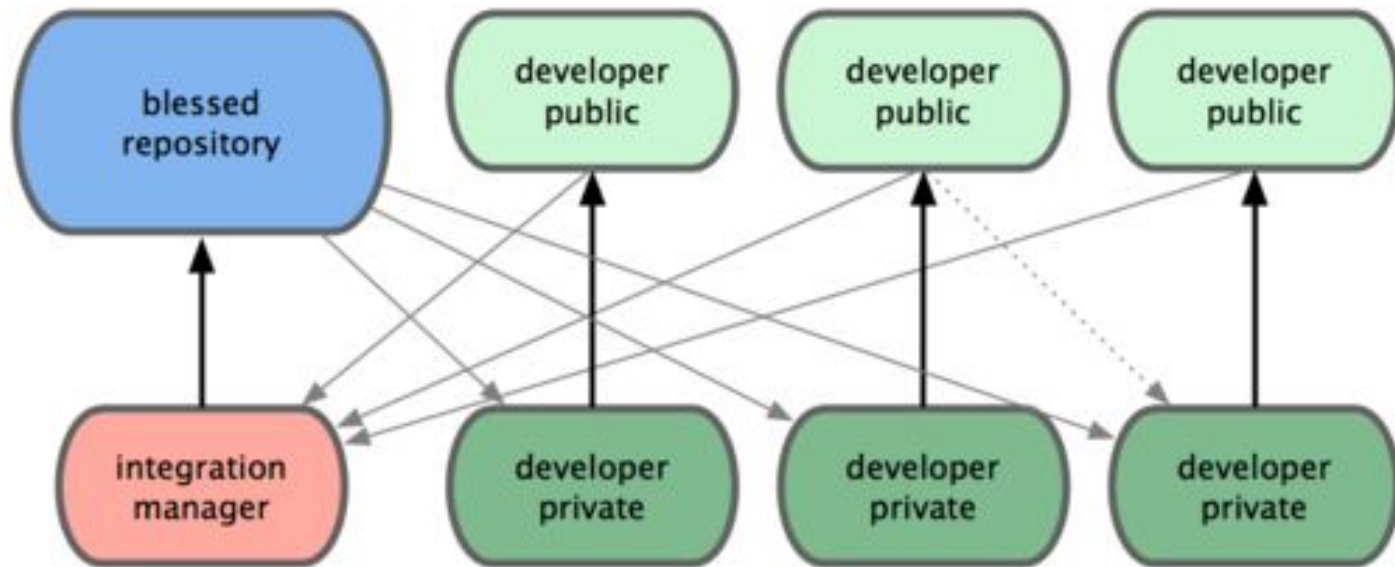
Wolf@WOLF-THINK ~/Projekte/Fortis/my-first-project (another-feature)
$ git cherry-pick feature
[another-feature e7de9b8] Second feature commit
1 file changed, 0 insertions(+), 0 deletions(-)
```

Workflow

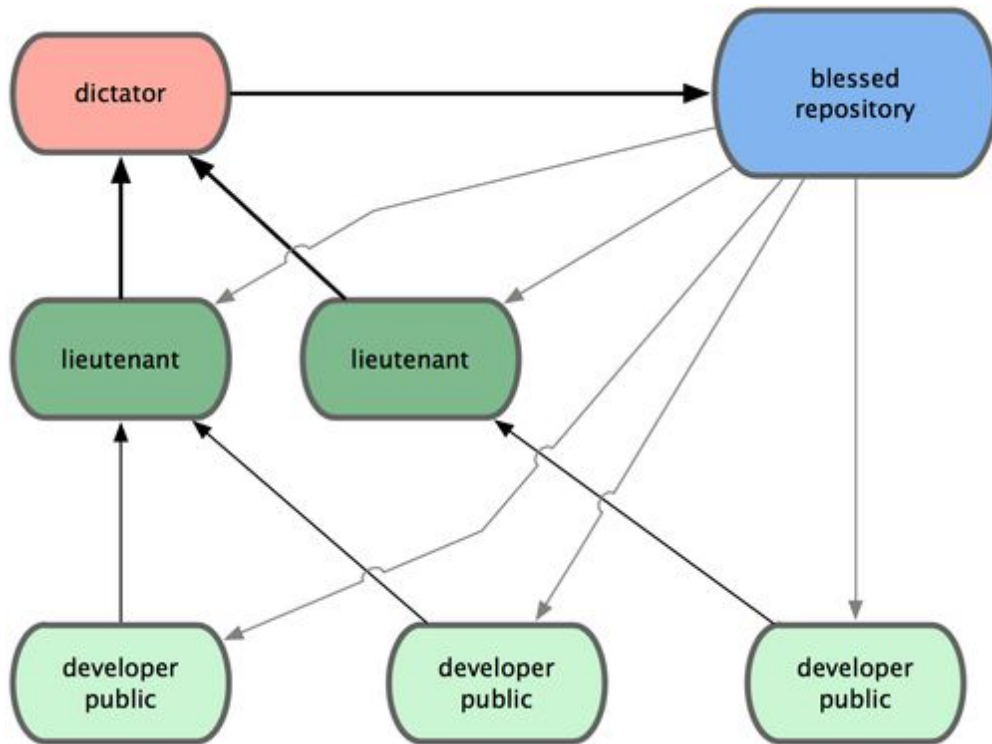
Distributed Workflows und Konventionen ...

Workflow - Distributed

Integration Manager



Workflow - Distributed



Dictator and Lieutenant

Konventionen - Commit Msg

`<Bereich|Datei>: <Kurze Beschreibung> (50 Zeichen soft limit)`

`<Genauere Beschreibung> (Stichpunkte sind gestattet)`

- Problemstellung (was ist am bisherigen Code verkehrt)
- gewählter Lösungsansatz mit Begründung
- ggf. verworfene Lösungsansätze

Imperativ verwenden! (“Füge foo hinzu” anstatt “Fügt foo hinzu”)

Konventionen - Branches

Es existiert nicht die eine Lösung

- Hauptbranch (meist master) nur stabile Arbeit (z.B. nur Releases)
- Entwicklungsbranch (develop) kontinuierlich fortführen
(Nicht direkt auf master oder develop commiten! merge oder rebase nutzen!)
- Feature Branches von develop abzweigen (z.B. feature/abc)
- Im Allgemeinen nur in eine Richtung mergen (feature => develop => master)

Mein persönlicher Favorit ([A successful git branching model](#))

The End

| | COMMENT | DATE |
|---|------------------------------------|--------------|
| ○ | CREATED MAIN LOOP & TIMING CONTROL | 14 HOURS AGO |
| ○ | ENABLED CONFIG FILE PARSING | 9 HOURS AGO |
| ○ | MISC BUGFIXES | 5 HOURS AGO |
| ○ | CODE ADDITIONS/EDITS | 4 HOURS AGO |
| ○ | MORE CODE | 4 HOURS AGO |
| ○ | HERE HAVE CODE | 4 HOURS AGO |
| ○ | AAAAAAA | 3 HOURS AGO |
| ○ | ADKFJSLKDFJSDKLFJ | 3 HOURS AGO |
| ○ | MY HANDS ARE TYPING WORDS | 2 HOURS AGO |
| ○ | HAAAAAAAAAANDS | 2 HOURS AGO |

AS A PROJECT DRAGS ON, MY GIT COMMIT
MESSAGES GET LESS AND LESS INFORMATIVE.