WORKFLOW

github.com/INT-WAW

WORKFLOW

INHALT

- Terminologie
- Triangular Workflow
- Awesome GitHub
- Projekt Setup
- Live-Coding

REQUISITEN

- GitHub Account
- github.com/INT-WAW ist GitHub Orga
- Texteditor (Sublime Text, Atom, VIM)
- Git Bash (fuer Windows Noobs)

TERMINOLOGIE

REPOSITORY

- Verzeichnis fuer Datenspeicherung
- Typischerweise auf Servern (oder Peers)
- Versionsverwaltung
- Protokolle von Veraenderungen
- Konfliktaufloesung
- Metadaten fuer Paketmanager

CLONE

- Duplikat eines Repositories
- Backup-Aspekt
- Offline-Verfuegbarkeit
- Dezentrale Vorkommnis

BEISPIEL

```
mkdir /opt/lycheejs;

git clone https://github.com/Artificial-Engineering/lycheeJS.git /opt/lycheejs;
```

COMMIT

- Veraenderung einer Datei
- Veraenderung zu einem Zeitpunkt
- Entfernen (-) oder Hinzufuegen (+)
- Zeilenbasiert mit Textuellem Inhalt
- Dateibasiert mit Binaerem Inhalt

BEISPIEL

```
cd /opt/lycheejs;

echo "Awesome Modification" > newfile.txt;
git add ./newfile.txt;
git commit -m "Awesome new commit";
```

REVISION

- Synonym fuer Version (Hash)
- Entwicklungsstadium der Software
- Einzelner Commit

TAG

- Entwicklungszeitpunkt
- Version der Software
- Serie an Commits

BEISPIEL

```
cd /opt/lycheejs;

echo "Awesome Modification" > newfile.txt;
git add ./newfile.txt;
git commit -m "Awesome new commit";

git tag v1.0 -m "Software is stable nao";
git push origin --tags;
```

BRANCH

- master ist stable
- Variante der Software
- Feature (Serie an Commits)

BEISPIEL

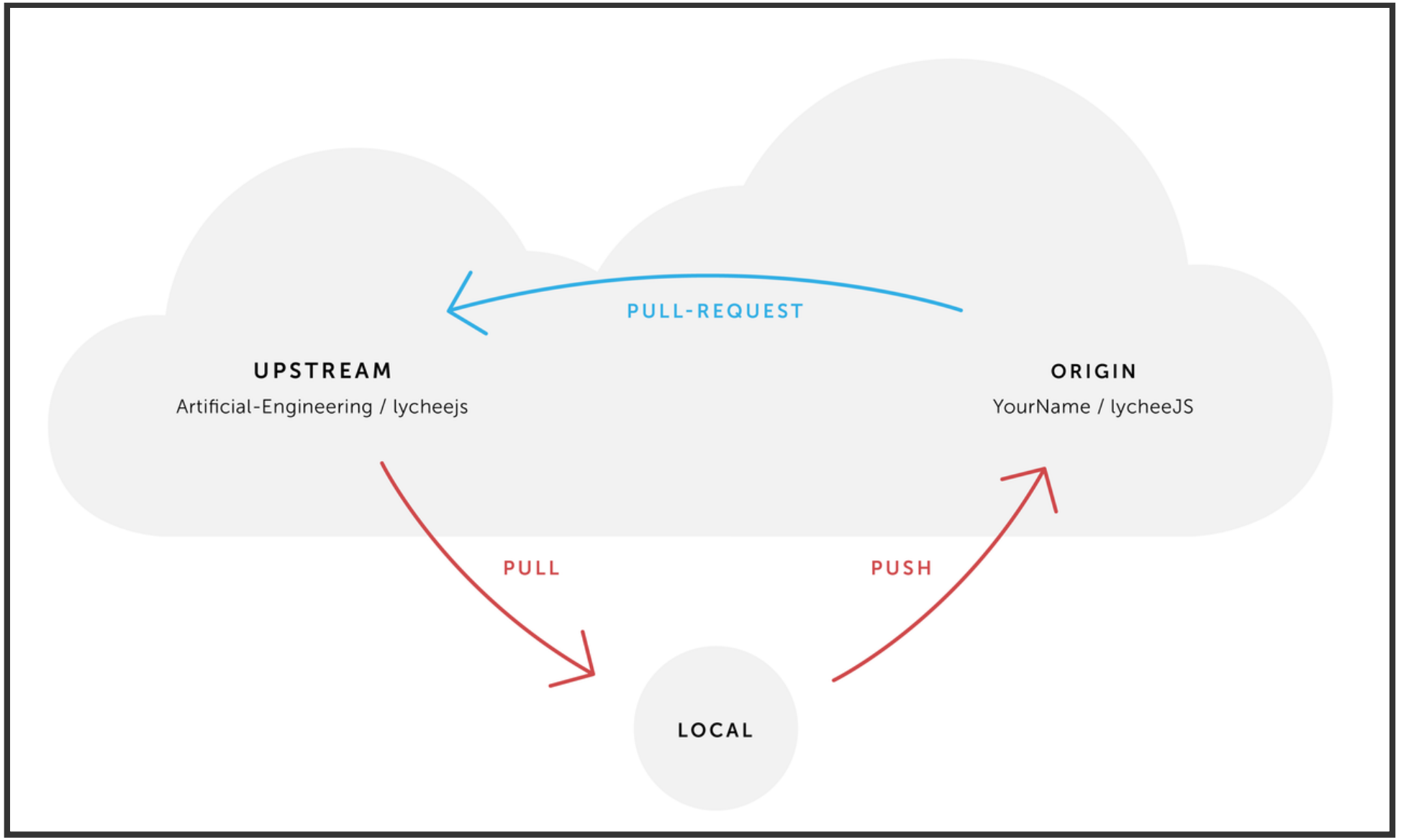
```
cd /opt/lycheejs;

git checkout development;
git checkout -b my-feature;

echo "awesome" > feature.txt;
git commit -m ":sparkles: Started work on my-feature";
```

FORK

- Abzweigung des Projekts
- Fork ist neue Software
- Upstream ist alte Software
- Beispiel: LibreOffice ist Fork von OpenOffice
- Beispiel: LibreSSL ist Fork von OpenSSL
- Beispiel: Blink ist Fork von WebKit ist Fork von KHTML

TRIANGULARER WORKFLOW

UEBERSICHT


PULL-REQUEST
UPSTREAM
Artificial-Engineering / lycheejs
ORIGIN
YourName / lycheeJS
PULL
PUSH
LOCAL

1. PULL

- Pull von Upstream Repository
- Upstream Repository ist behind

2. COMMIT

- Lokaler Merge von Commits
- Lokale Feature Branches
- Lokale Commits sind ahead

3. PUSH

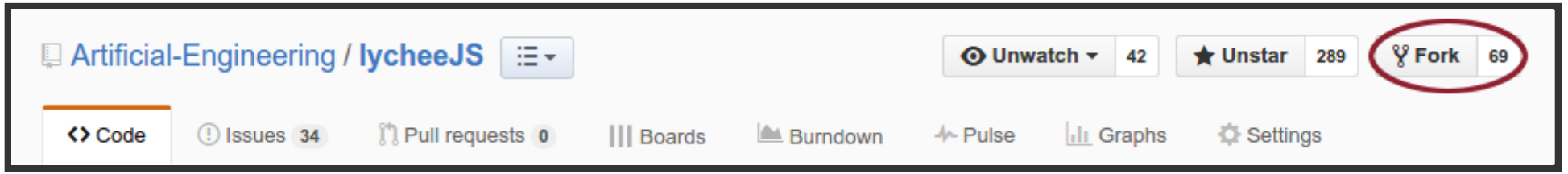
- Push zu eigenem Fork (Origin)
- Origin Repository ist ahead
- Keine Merge Conflicts

4. PULL REQUEST

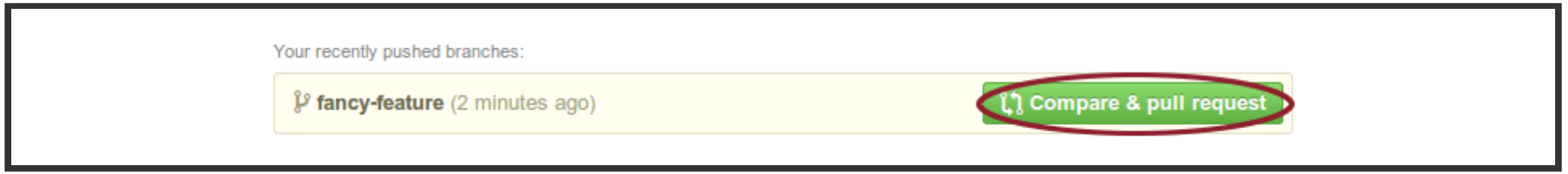
- Fork Maintainer initiiert Pull Request
- Upstream Maintainer merged Pull Request
- Keine Merge Conflicts
- Alle sind gluecklich
- Suck it, SVN und CVS

AWESOME GITHUB

EINFACHE FORKS



EINFACHE PULL REQUESTS



COMMITS

- `Fixes` issue `#1337`
- `Solves #1337`
- `Closes` issue `#1337`
- `More` work on issue `#1337`

ISSUES

- Probleme und Diskussionen
- Verlinkung von Contributors (`@username`)
- Verlinkung von Commits (`@hash`)
- Verlinkung von Issues (`#issue`)
- Verlinking von Code Reviews
- Votes und Reactions

CODE REVIEWS

- Kommentare zu Commits
- Einbettung in Issues
- Einbettung in Wikis

WIKIS

- Markdown Syntax
- Einbettung in allen Dateien (`.md`)
- Einbettung in Issues und Commits
- GitHub Code Editor

PROJEKT SETUP

AUFGABE

- Installiere git (git-bash)
- Forke die Boilerplate 1
- Clone deinen Fork lokal
- Fuege Inhalt der HTML5 Datei hinzu
- Erstelle einen Feature Branch
- Habe Commits von ALLEN Teammitgliedern
- Erstelle einen Pull Request

1 github.com/INT-WAW/Boilerplate.git