



# Get started with git

# Intro

- ist kostenfrei
- open source distributed vcs
- für kleine und große Projekte

# Wir lernen heute

- Kommandos
- Operationen
- Praktische Anwendung

# Motiv für git

Das Ziel von Git ist es, ein Projekt oder eine Reihe von Dateien zu verwalten, während sie sich im Laufe der Zeit ändern. Git speichert diese Informationen in einer Datenstruktur, die als Git-Repository bezeichnet wird. Das Repository ist das Herzstück von Git.

# Motiv für git


- Git-Repository ist ein Verzeichnis, in dem alle Meta-Daten zu einem Projekt gespeichert werden.
- Git speichert den Zustand in einem Directed Acyclic Graph (DAG)

# Basic Operations

1. Initialize
2. Add
3. Commit
4. Pull
5. Push

# Advanced Git Operations

- Branching
- Merging
- Rebasing

 image



# Git installieren

```
PS C:\Users\arctic> choco search git
Chocolatey v1.1.0
git 2.36.0 [Approved]
PS C:\Users\arctic> gsudo choco install git
```

```
PS C:\Users\arctic> git --version
git version 2.36.0.windows.1
```

# Alternativ web

Web-win  
Web-mac

**Jetzt gehts los**

# Initialize

```
PS C:\Users\arctic\git-tutorial> git init  
Initialized empty Git repository in C:/Users/arctic/git-tutorial/.git/
```



Wir haben unser git repository erstellt

# Initialize

`git init` erstellt ein leeres git repository oder re-initialisiert ein bereits vorhandenes. Es erstellt einfach ein `.git` Verzeichnis mit allen relevanten Informationen.

**Es werden keine Daten überschrieben, nur das Template aktualisiert**

# Status

```
git status
```

- zeigt alle modifizierten Dateien
- zeigt alle neuen Dateien
- zeigt alle gelöschten Dateien

# Add

```
git add
```

- fügt eine Datei oder Ordner zum Commit hinzu
- updated den Index mit den neuen Informationen

Also **bevor** man den *working tree* ändern kann, muss man den **add** Kommand ausführen.

# Commit

- refereiert dazu einen Schnappschuss zu erstellen zu einer gewissen Zeit
- etwas das committed wurde wird nicht mehr geändert (außer explizit angefordert)



# Commit

 image

# Commit

`git commit` - erstellt einen Schnappschuss

`git commit -m "<nachricht>"` - erstellt einen  
Schnappschuss mit einer Nachricht

Diese Nachricht sollte eine kurze Beschreibung der  
Änderungen enthalten. (max. 42 Zeichen)

# Pull

```
git pull
```

- lädt die aktuellste Version des Repositorys herunter
- merged die upstream Änderungen in den lokalen Repository
- benötigt ein Remote Repository

# Remote

```
git remote add origin <url>
```

- fügt einen Remote Repository hinzu

# Push

```
git push origin master
```

- sendet die aktuellen Änderungen an den Remote Repository
- Pulling = Import; Pushing = Export

# Branching

- Branches sind nichts anderes als pointer zu spezifischen Commits
- Es gibt 2 Arten: *local branches* und *remote branches*

# Branching

Ein lokaler Zweig ist nur ein weiterer Pfad in eurem Working Tree. Andererseits haben remote tracking branches einen besonderen Zweck. Einige von ihnen sind:

- verknüpfen lokale Arbeit mit einem Remote Repository
- können automatisch erkennen welche Änderungen sie mittels *git-pull* holen sollen

# Branching

```
git branch
```

- zeigt alle lokalen Branches an

```
git branch <name>
```

- erstellt einen lokalen Branch von dem aktuellen aus



# Branching

 image

# Branching

- um zwischen branches zu wechseln nutzt

```
git checkout <name>
```

# Merging

- mittels Merging wollen wir unsere ganzen branches wieder zusammenfügen
- z.B. nutzen wir einen branch um ein Feature zu entwickeln, wollen dies dann aber allen im Main (Master) zur Verfügung stellen

# Merging

 image

# Merging

```
git merge <branch_name>
```

- wir mergen jetzt also unseren aktuellen branch in den *branch\_name*
- jetzt hat *branch\_name* alle unsere Änderungen
- Achtung: manchmal sind anpassungen nötig

# Rebase

- manchmal wollen wir die commits linearer machen
- mit `git rebase <branch_name>` können wir auf einen anderen Ausgangs-Commit zeigen

# Rebase

 image

# **Jetzt Praktisch anwenden**



# Practice

1. `git clone https://github.com/arcticspacefox/git-tutorial.git`
2. Erstellt einen eigenen Branch mit dem Namen `feature/<nutzerkürzel>`
3. Führt die jeweilige bin aus `git-tutorial.exe` oder `git-tutorial`
4. jetzt guckt ihr mit `git status` ob alles in Ordnung ist und comitted
5. macht einen lokalen merge mit dem Branch `master`
6. pusht eure Änderungen an den Remote Repository

# Tips und Tricks

# Naming von Commits

# Naming von Branches

- Feature: `feature/<feature_name>`
- Bugfix: `bugfix/<bug_name>`

# Fragen?