GIT CONCEPTS

KAI ARULKUMARAN

WHY GIT?

Free

Open source

Fast

Small

Backups (implicitly)

Collaborate

VERSION CONTROL SYSTEM

Snapshots in time

Choose what changes to track

Add meaning with messages

Branch out

DISTRIBUTED VERSION CONTROL SYSTEM

Share copies of history

Merge work

No single point of failure

REPOSITORIES

Create

```
git init
```

Clone

```
git clone <username>@<host>:<repository path>
```

Status

```
git status
```

History

```
git log --graph --oneline
```

WORKING AREAS

Working Directory

↓

```
git add <files>
```

↓

Index (staging area)

↓

```
git commit -m "<message>"
```

↓

HEAD (last commit of branch)

BRANCHES

Default branch is **master**

Create

```
git branch <branch>
```

Switch

```
git checkout <branch>
```

Merge

```
git merge <branch>
```

Delete

```
git branch -d <branch>
```

REMOTE REPOS

Default server is **origin**

Add

```
git remote add <server> <server path>
```

Push

```
git push <server> <branch>
```

Pull (fetch and merge)

```
git pull <server> <branch>
```

UNDOING CHANGES

Unstage

```
git reset HEAD <filename>
```

Revert file to state at HEAD

```
git checkout -- <filename>
```

Revert all files to state at HEAD

```
git reset --hard
```

Disclaimer: Changing history may lead to time paradoxes...

USEFUL LINKS

Online tutorial: Try Git

In-depth tutorial: Git Immersion

Quick guide: git - the simple guide

Alternative perspective: The Git Parable

Development conventions: A successful Git branching model

Ignoring files: github/gitignore

Repo hosting: Bitbucket