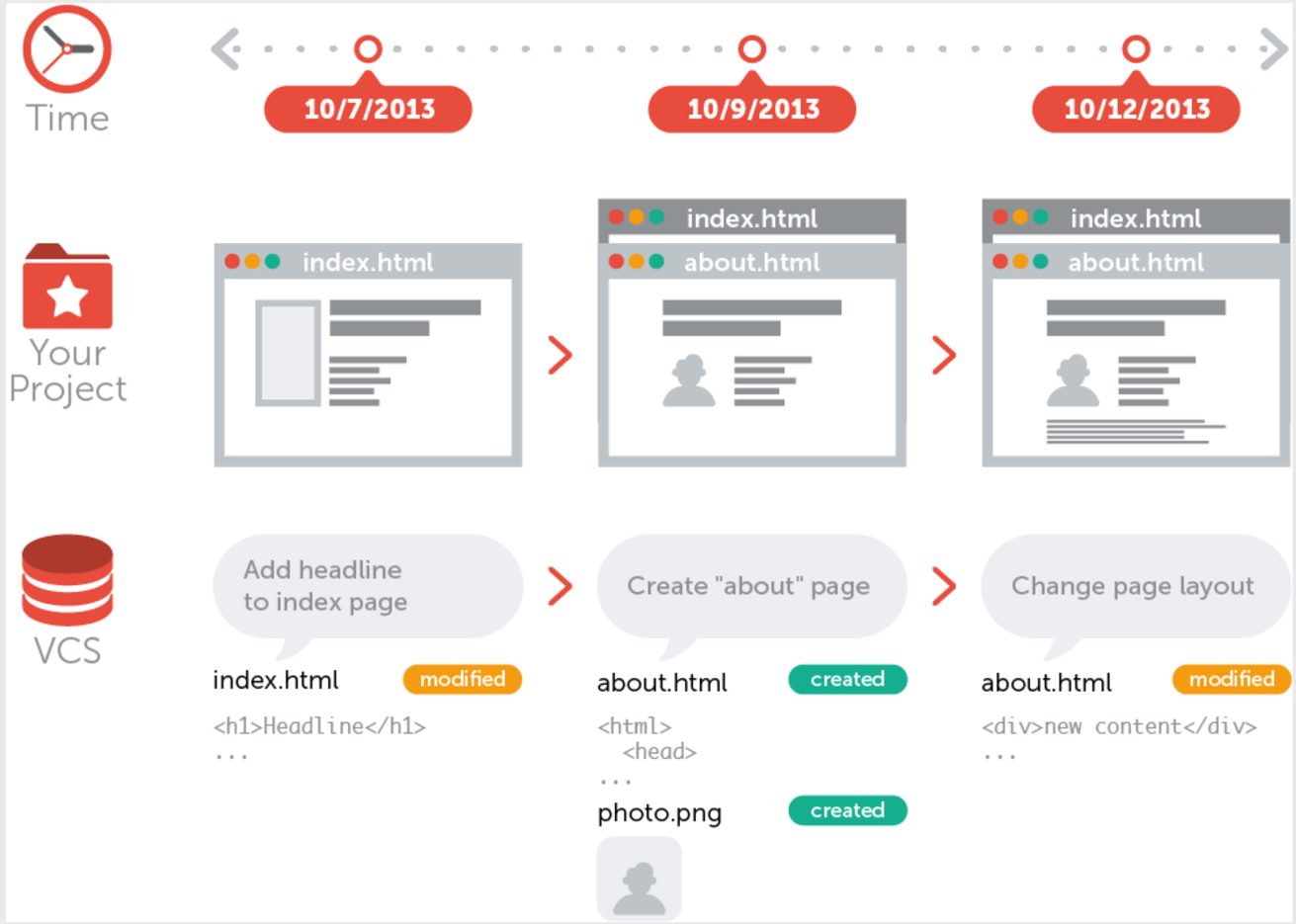
Git & Vcs

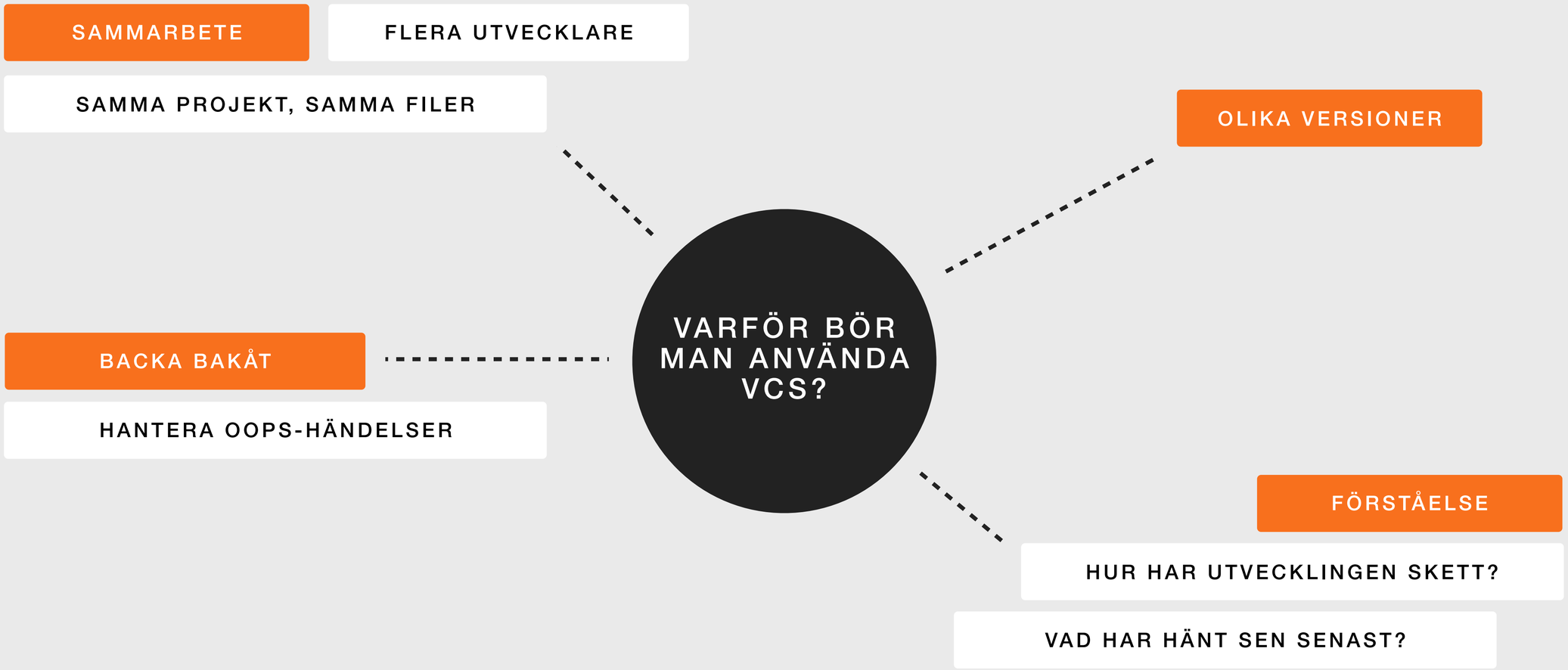
LITHEKOD

VIKTOR HOLMGREN

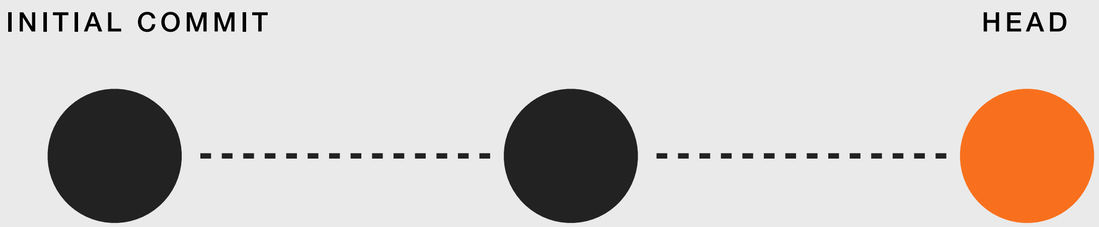
Vad är VCS?

- Ett system som håller reda på ändringar av en uppsättning filer, ett *repository*.
- Låter dig återställa filer från tidigare tillstånd
- Jämföra ändringar över tid
- Se vem som senast ändrade något
- och så vidare…



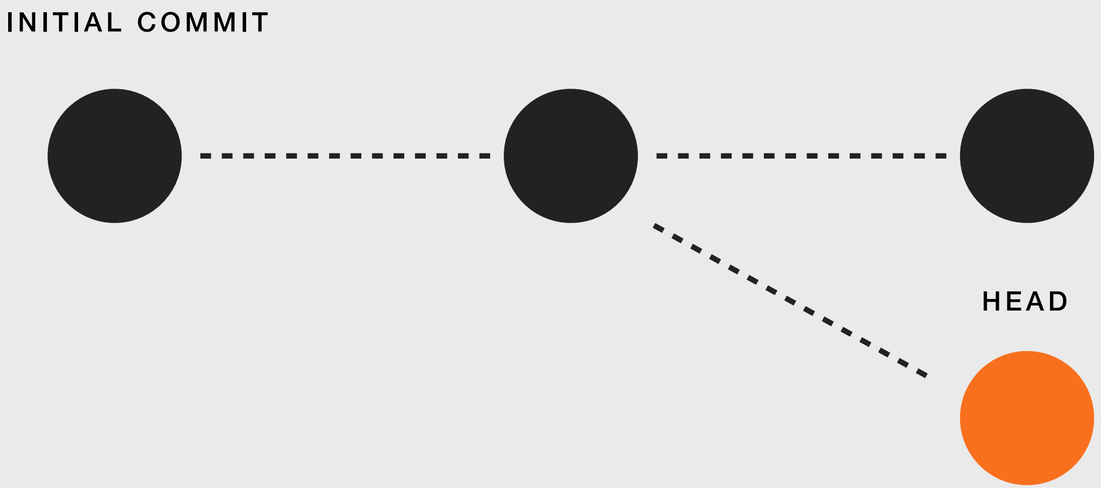

SAMMARBETE
FLERA UTVECKLARE
SAMMA PROJEKT, SAMMA FILER
OLIKA VERSIONER
VARFÖR BÖR MAN ANVÄNDA VCS?
BACKA BAKÅT
HANTERA OOPS-HÄNDELSER
FÖRSTÅELSE
HUR HAR UTVECKLINGEN SKETT?
VAD HAR HÄNT SEN SENAST?

Repository



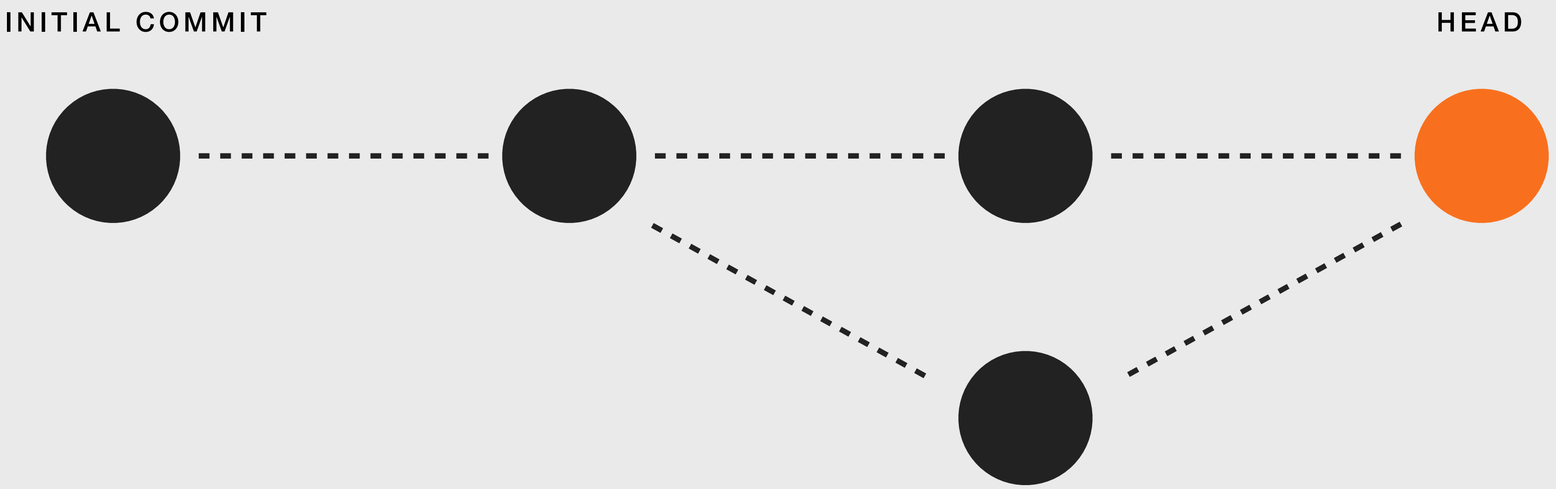
- Huvudgrenen kallas ofta för **Trunk** eller **master branch**
- Varje nod kallas för en **commit**
- **Head** är den nyaste noden (eller i Git:s fall den aktuella noden)
- **Working copy** är den version som du för tillfället har *utcheckat*

Branches



- En gren i trädet, kallas för en **branch.**
- Grenar är helt skilda och arbete kan göras i dessa parallellt

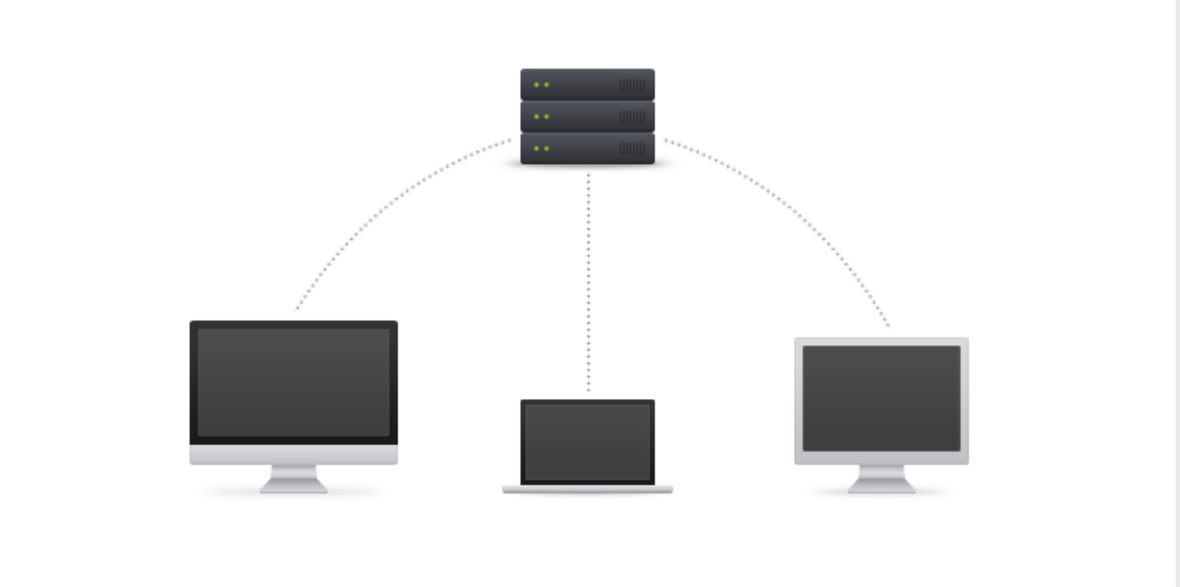
Merging



- Att slå samman två **branches** kallas för en **merge**
- Om det finns skillnader mellan grenarna och datorn inte kan avgöra vilken ändring som ska gälla så får man en **merge conflict**

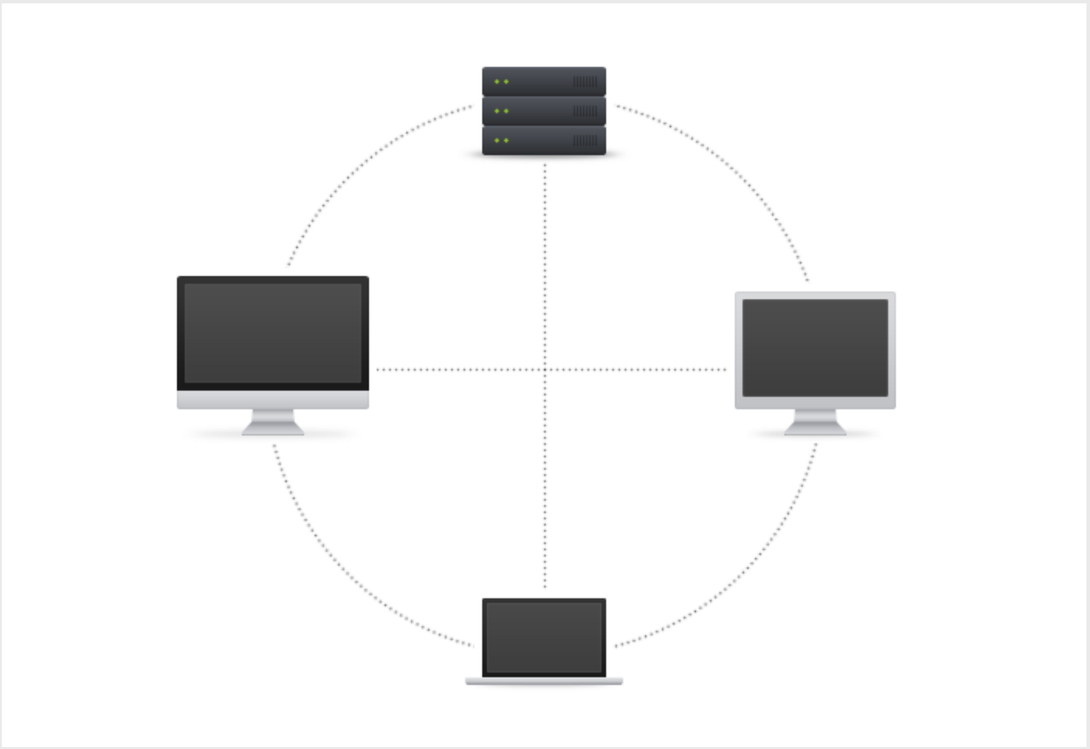
Centraliserade Vcs

- En central server där *repository:et* finns
- Lättare att förstå, lägre inlärningskurva
- Större kontroll över utvecklare
- Beroende av tillgång till server
- Brancher och merging är svårt



Distribuerade Vcs

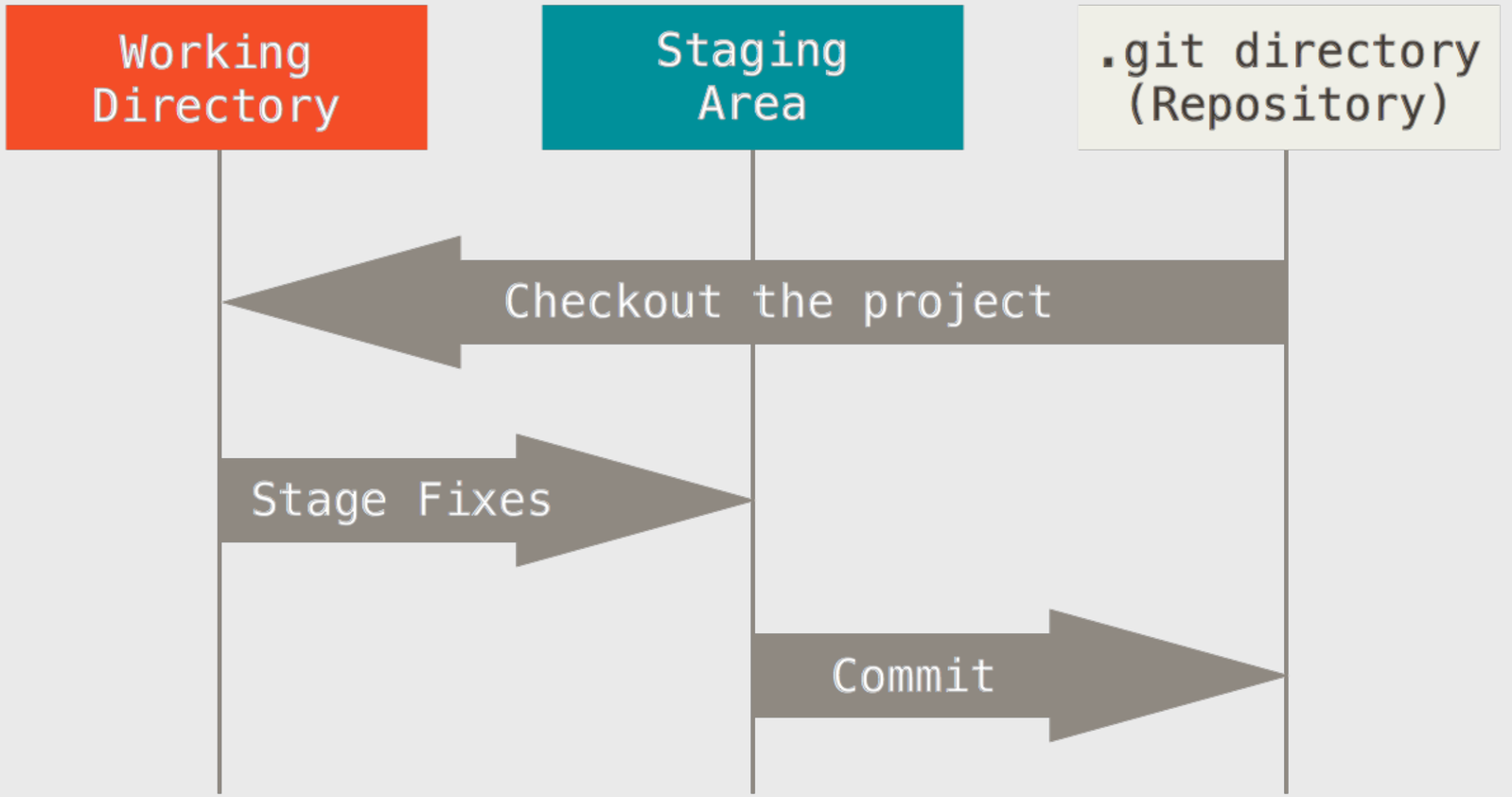
- Ingen kopia är “viktigare” än någon annan
- Kraftfullare och mer detaljerad ändringshantering
- Ingen server är nödvändigt, allt går att göra offline
- Branching och merging är trivialt
- Snabbt som f*n
- Gör Open Source möjligt på en helt annan nivå med koncept så som: pull requests.
- Den distribuerade modellen är mer svårförstånd



Git

- Byggt på principen: Vad skulle CVS och SVN **aldrig** göra.
- Git är distribuerat tillskillnad från SVN som är centraliserat
- Git har mycket bra stöd för icke linjärt arbete, branches är lättviktiga och merging är trivialt
- Git är snabbare i nästan alla viktiga kategorier
- Git följer innehåll och inte filer
- Git är korruptionssäkert (i princip)
- Git använder en s.k **staging area**, möjliggör partiella *commits*
- Push och Pull

Workflow


Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit

Vanligaste kommandon

- **add -** Add file contents to the index
- **branch -** List, create, or delete branches
- **checkout -** Checkout a branch or paths to the working tree
- **clone -** Clone a repository into a new directory
- **commit -** Record changes to the repository
- **diff -** Show changes between commits, commit and working tree, etc
- **init -** Create an empty Git repository or reinitialize an existing one
- **log -** Show commit logs
- **merge -** Join two or more development histories together
- **pull -** Fetch from and integrate with another repository or a local branch
- **push -** Update remote refs along with associated objects
- **reset -** Reset current HEAD to the specified state
- **status -** Show the working tree status