Git
Introduction to Git

Version Control Systems

Juraj Oravec

STU

Slovak University of Technology in Bratislava

Git

Developed by Linus Torvalds in 2005

Git

Distributed version control system (not only) for tracking changes

Git – replaces archives

Main problems of archives (zip/rar/gz):

– does not protect against accidental deletion

– linear development (release when all new features are complete)

– single-user development

– difficult to find a change that introuced a bug

Git – replaces archives

Main problems of archives (zip/rar/gz):

– does not protect against accidental deletion

– linear development (release when all new features are complete)

– single-user development

– difficult to find a change that introuced a bug

Missing answers to the questions:

– what has changed between versions?

– who made the changes?

– when and why were changes made?

– which version is the latest stable release?

Git – main objectives

- recordes changes over time (*changesets*)
- recalls a specific version later (*checkout*)
- enables collaboration (*distributed*)
- allows nonlinear development (*branches*)

Git – DOs and DONTs

DONTs:

– do not use archives as a substitute for version control

– never use a centralized VCS (CVS, SVN)

– do not store automatically generated files (PDFs, binaries, etc.)

– do not store sensitive data (passwords, logins, SSH keys, etc.)

DOs:

– use a distributed VCS (Git, Mercurial)

– use proper branching model

Git – basic workflow

Remote repository

cloud

local machine

Working directory

Git – basic workflow

Remote repository

cloud

local machine

Local repository (.git/)

Working directory

Git – basic workflow

Remote repository

cloud

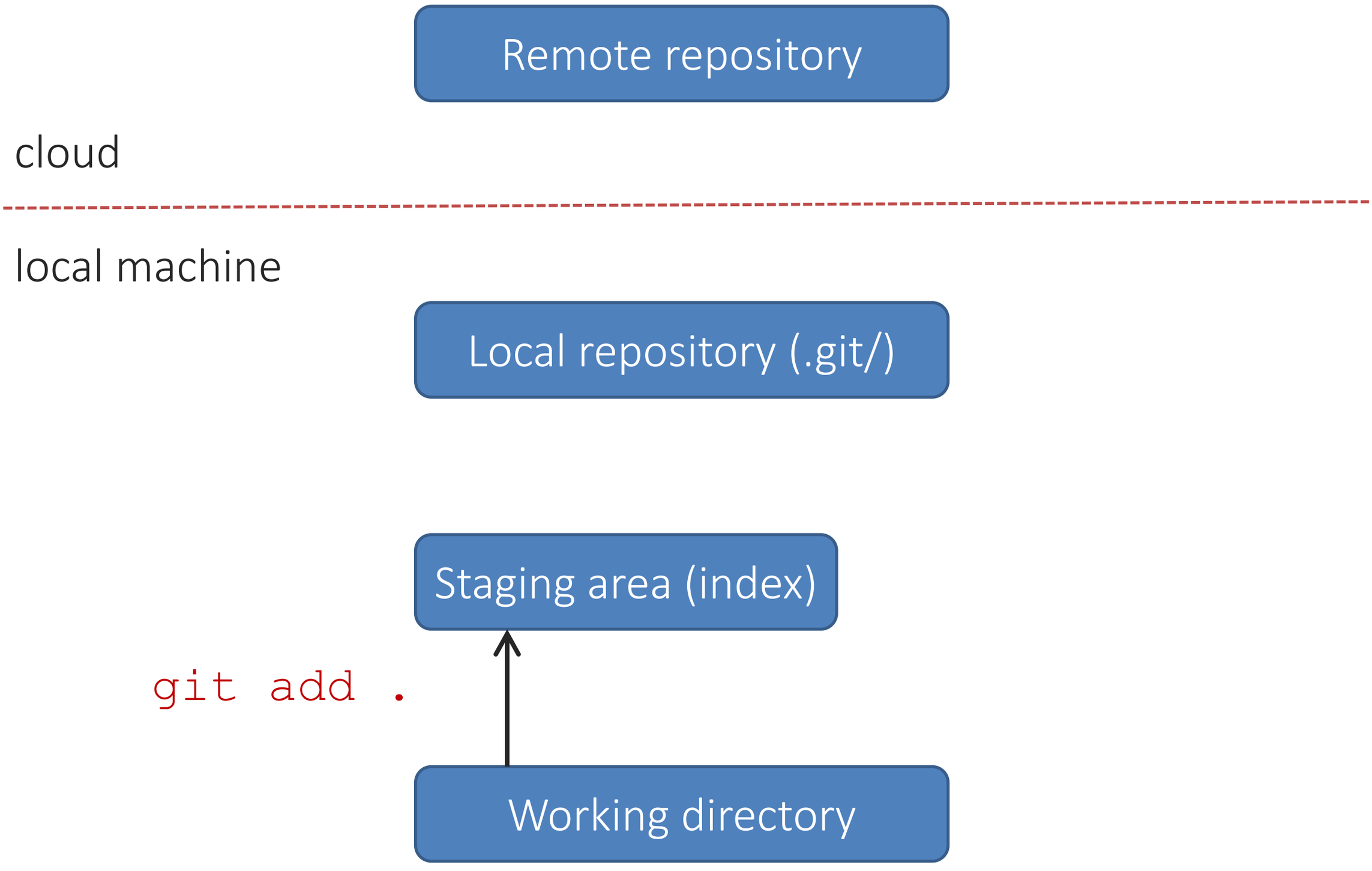
local machine

Local repository (.git/)

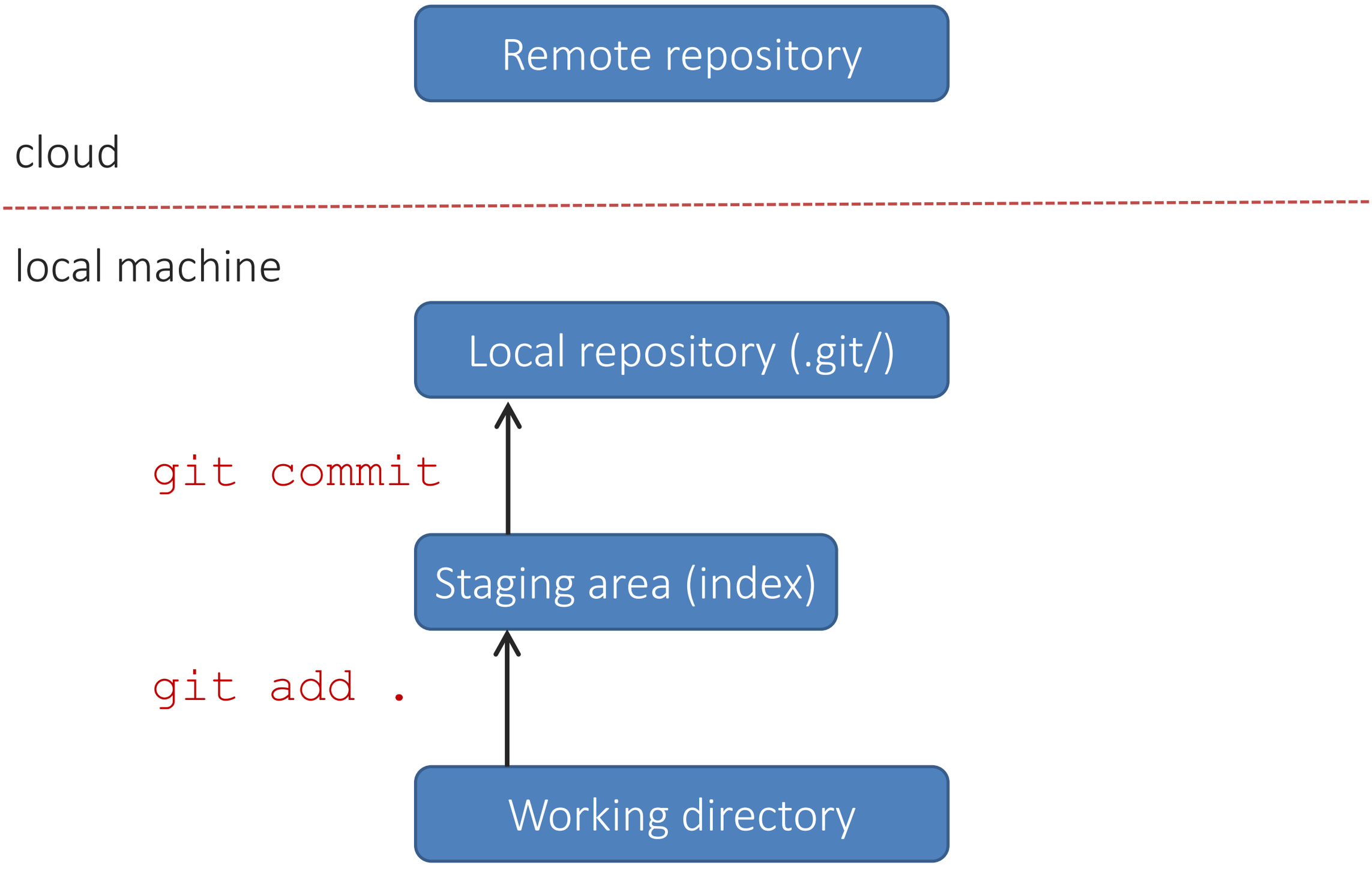
Staging area (index)

Working directory

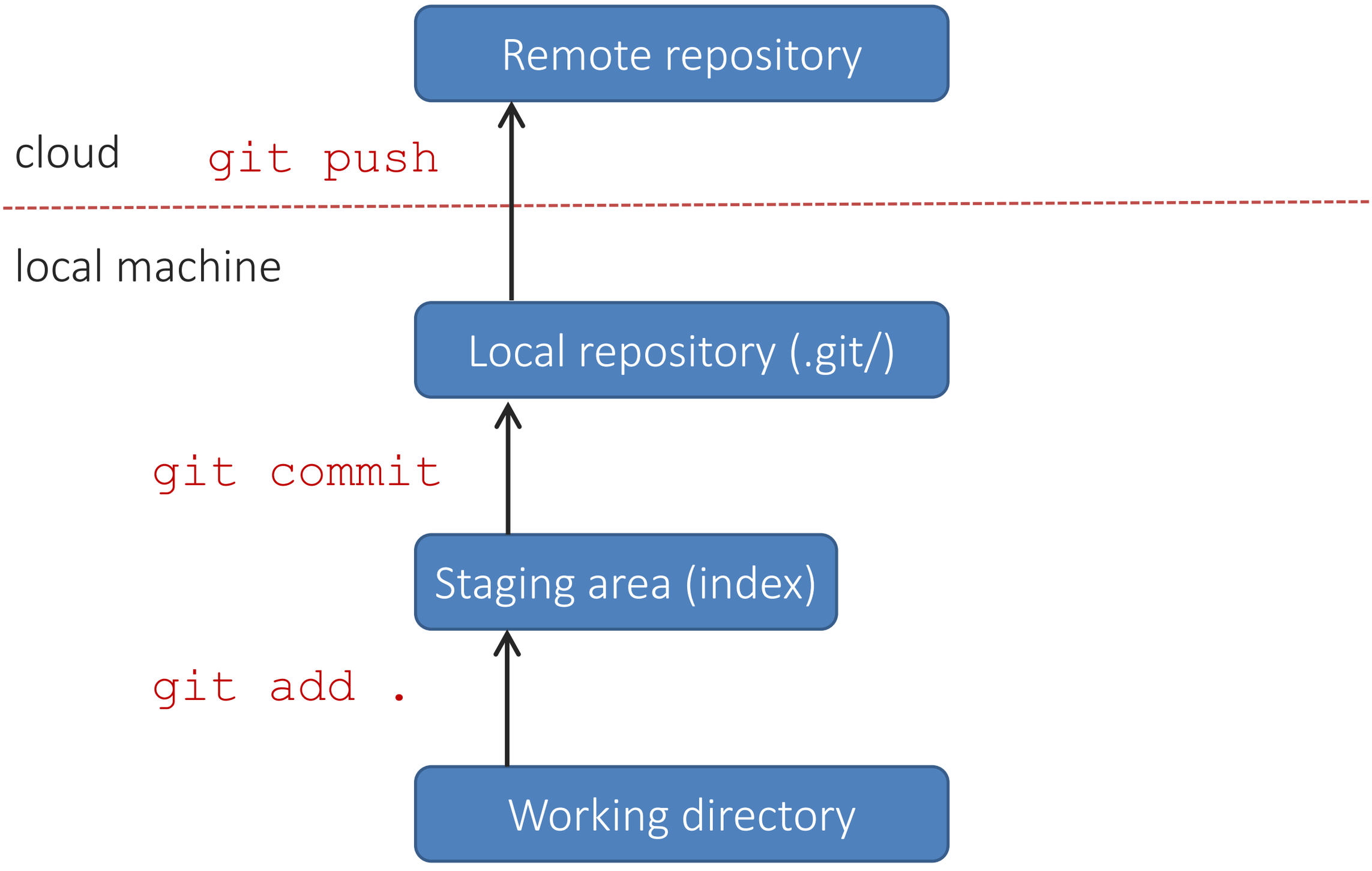
Git – basic workflow



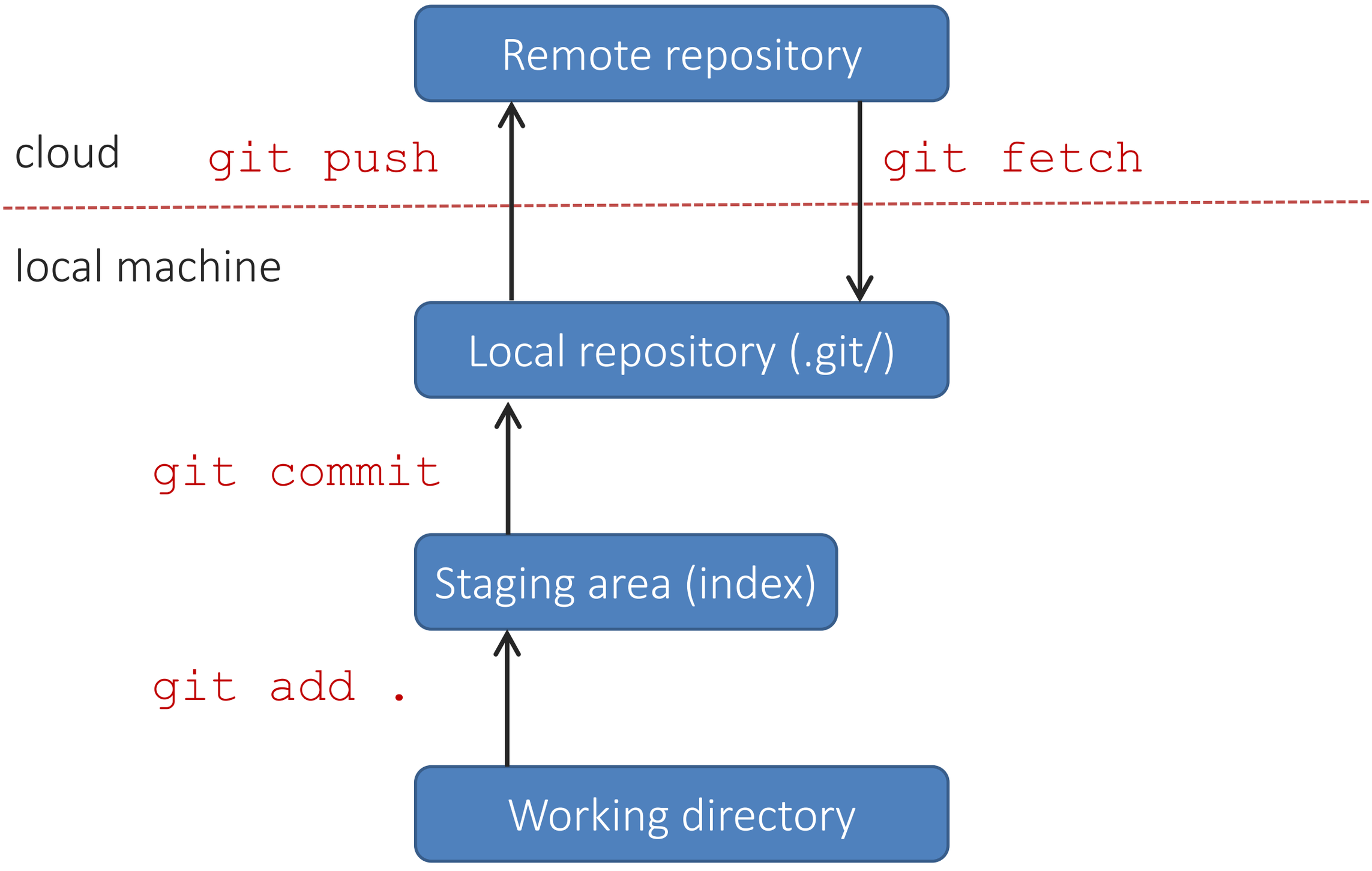
Git – basic workflow



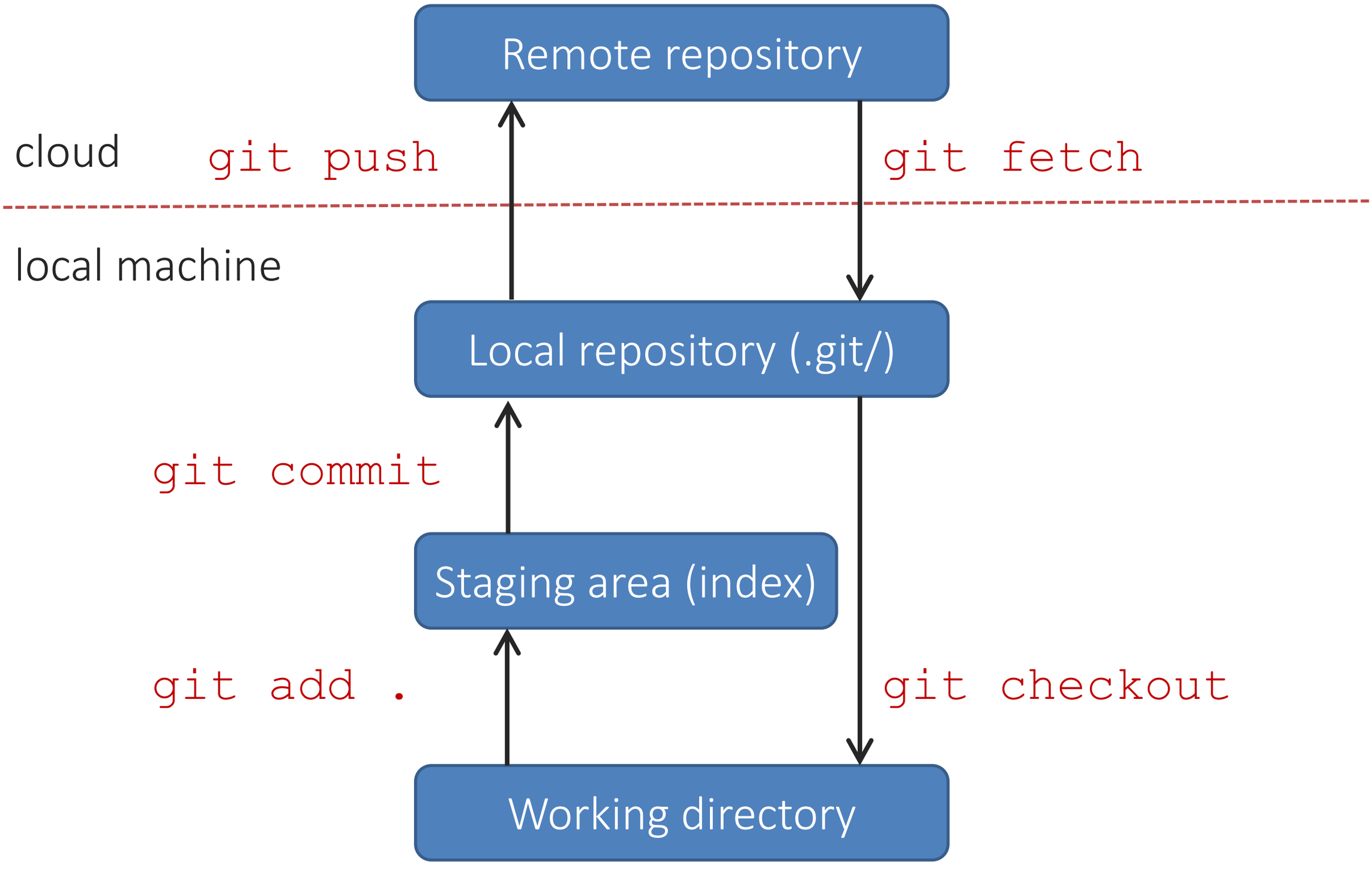
Git – basic workflow



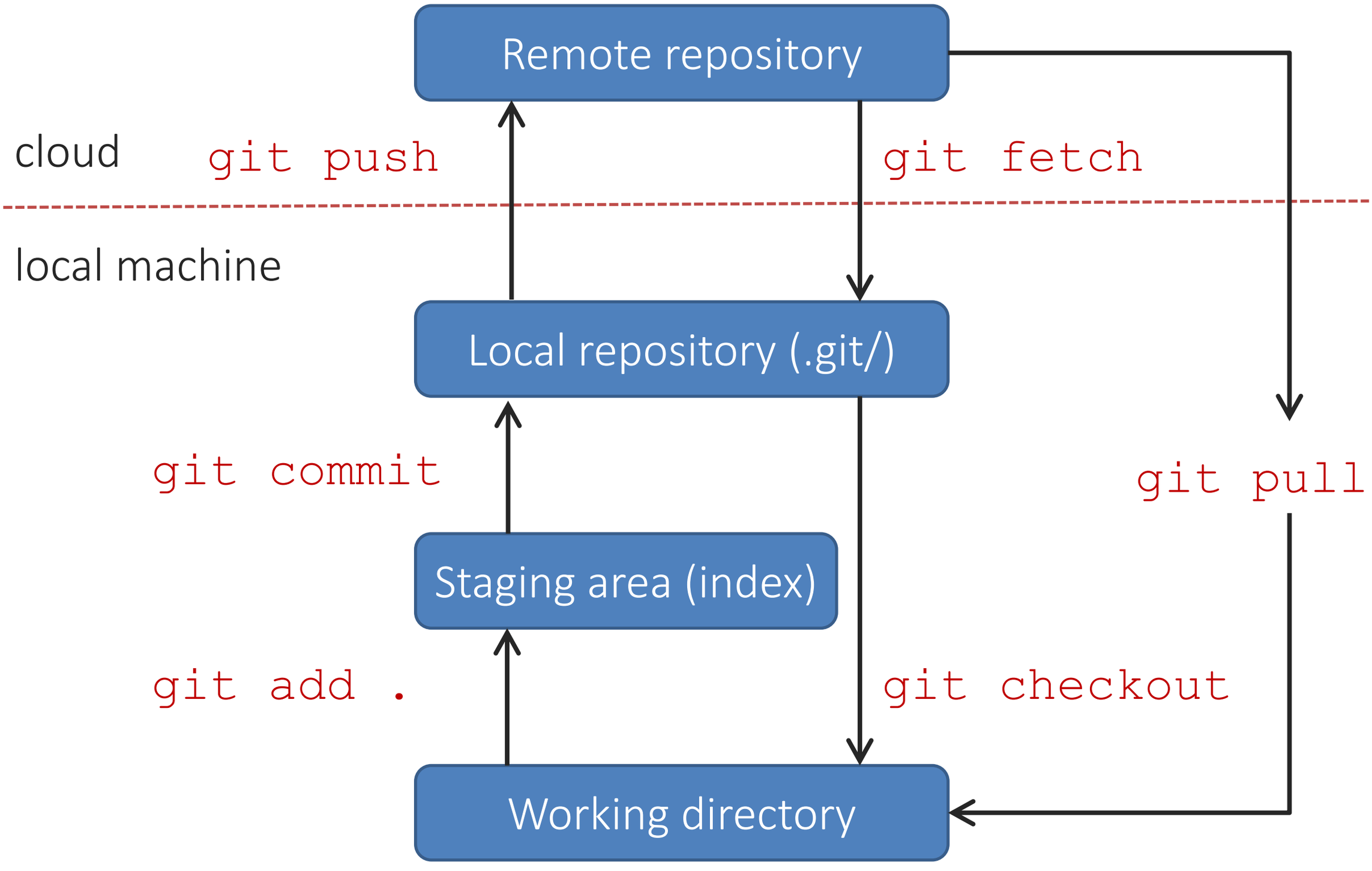
Git – basic workflow



Git – basic workflow



Git – basic workflow



Git – basic objects

`hash`

object identifier – each object has its (hexadecimal) *hash*

`blob`

contains *pure* file content (without file names)

`tree`

directory listings – path entries paired with other objects

`commit`

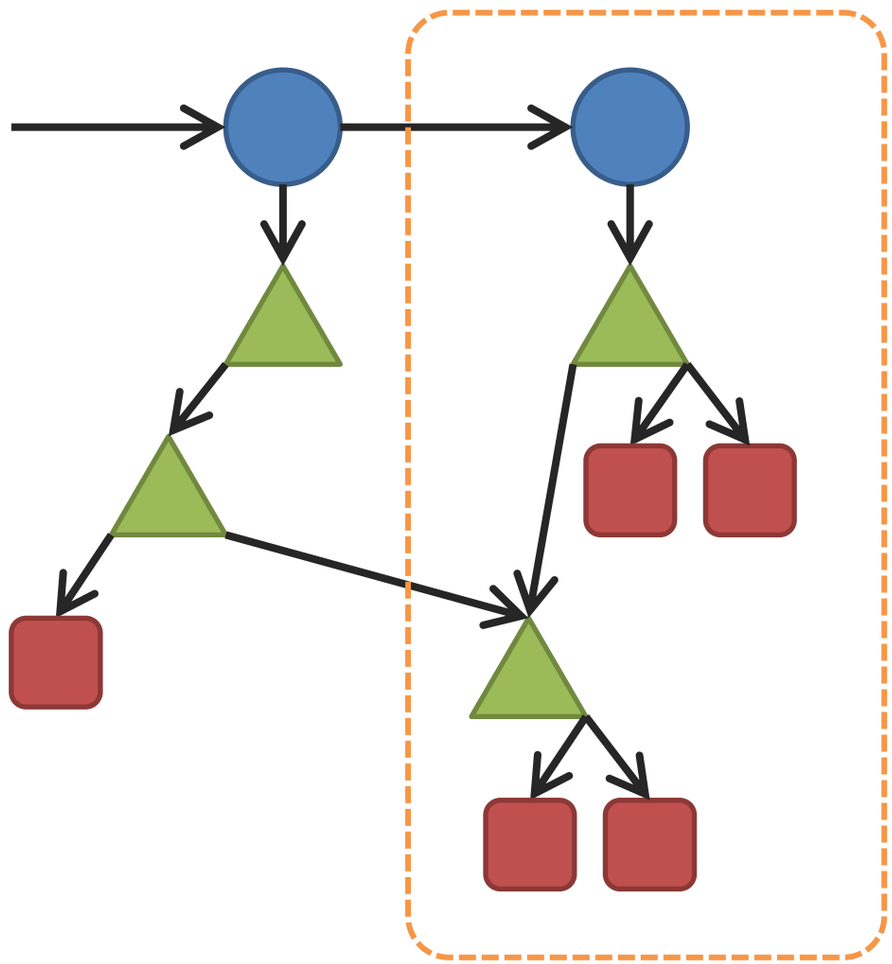
snapshots (*changeset*) of repository in time; each commit has its root tree

Git – basic objects

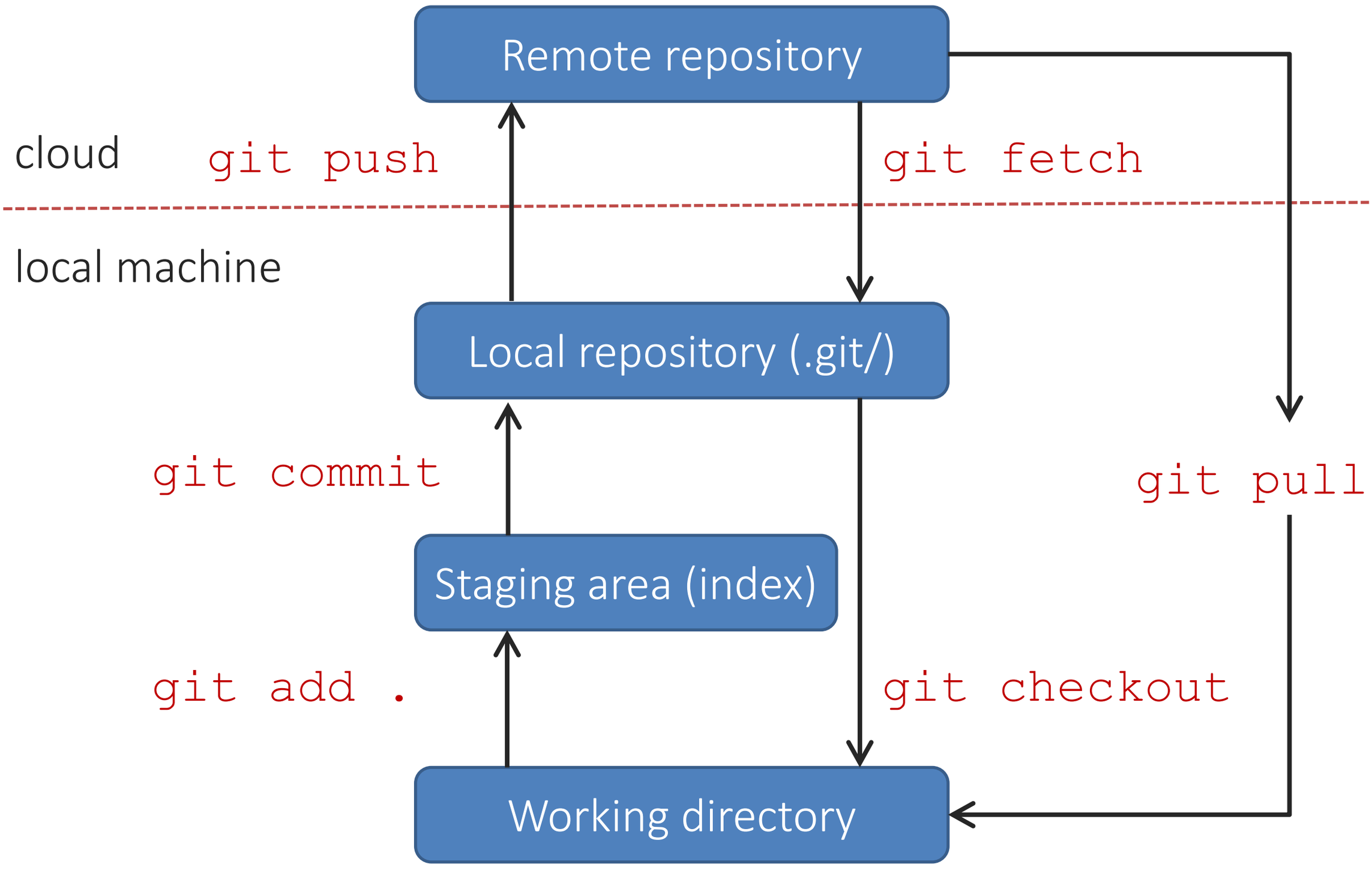
`commit` (circle)

`tree` (triangle)

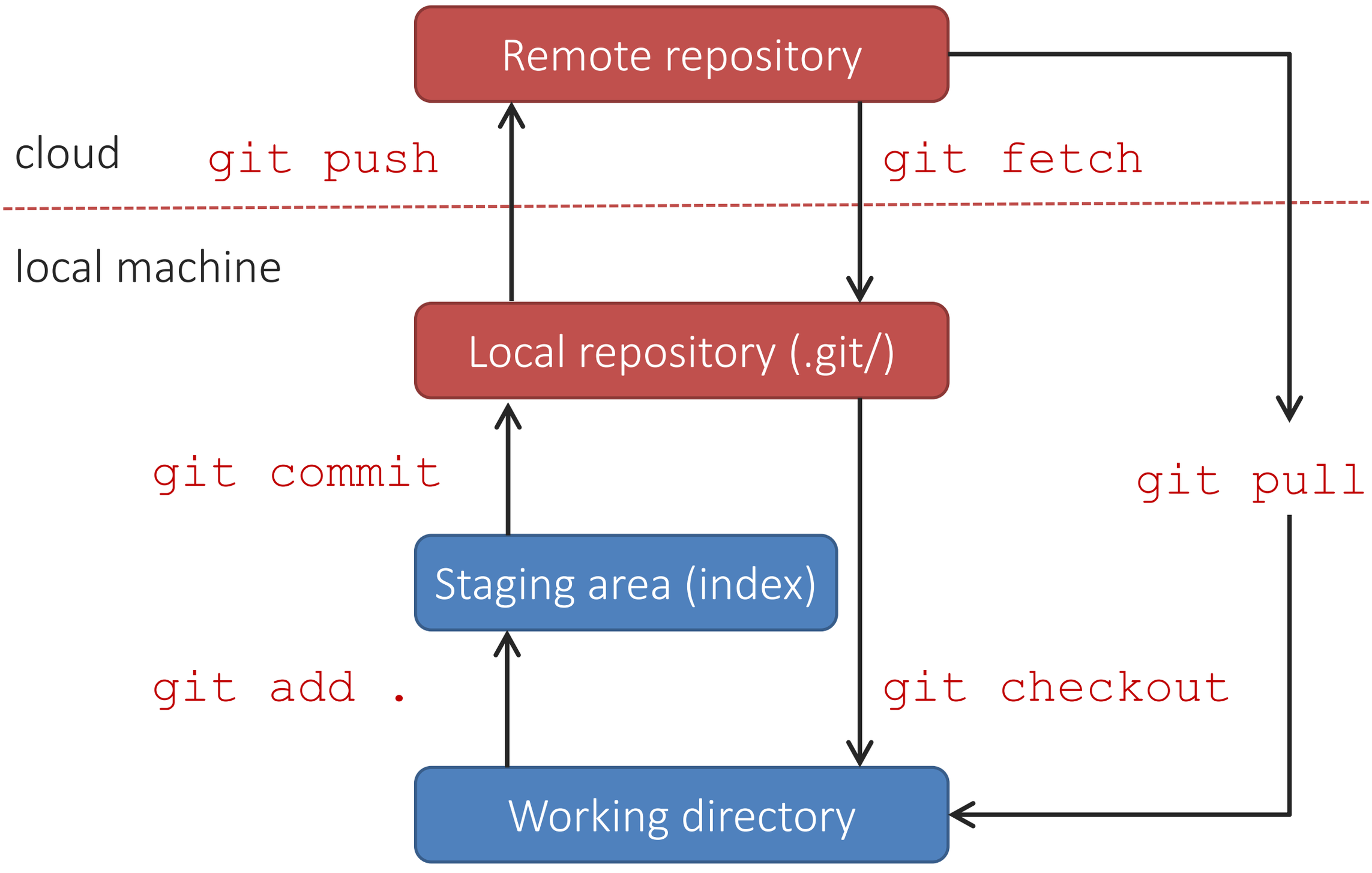
`blob` (box)



Git – summary


Remote repository
cloud
git push
git fetch
local machine
Local repository (.git/)
git commit
git pull
Staging area (index)
git add .
git checkout
Working directory

Git – outlook


Remote repository
cloud
git push
git fetch
local machine
Local repository (.git/)
git commit
git pull
Staging area (index)
git add .
git checkout
Working directory