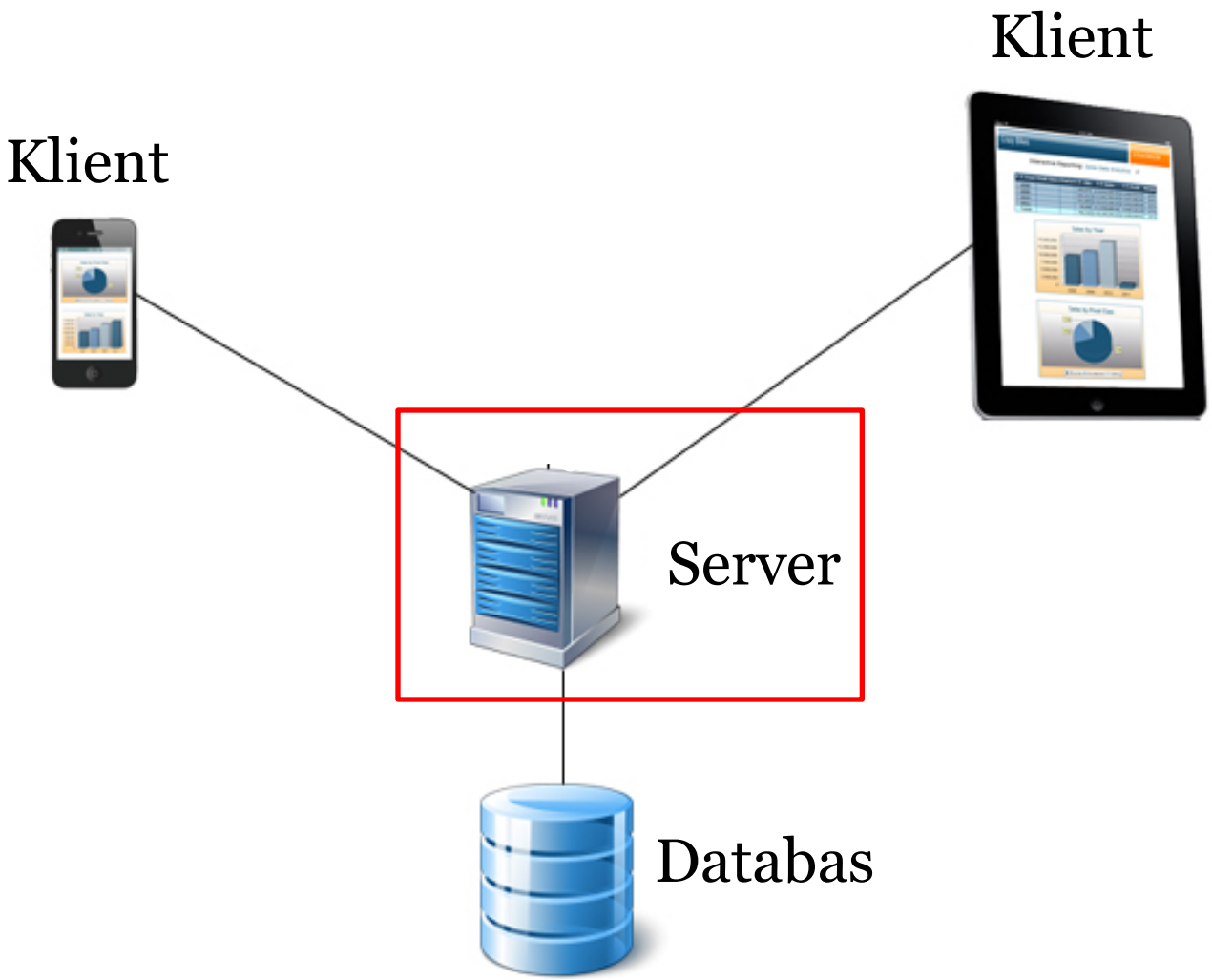
Server-kod

TDDD80 Mobila och sociala applikationer

LINKÖPINGS UNIVERSITET

Översikt

- Flask
- HTTP och REST
- Requests-biblioteket
- JSON
- PyCharm
- Git
- Informationssökningstips



Labb 4-8 (Android)

Labb 1-4 (server)

Servera (tillhandahålla) data till klienter

- REST (REpresentational State Transfer)
- Restful design
  - Tillståndslöst (varje anrop är oberoende av förra)
  - Tillstånd kodas och skickas med vid anrop (t.ex. tillståndet “inloggad”, dvs. access-token, del i anropet)
- Kommunikation via JSON, dvs. strukturerade strängar

Flask

Utvecklingsmiljö

- Mikro-ramverk (baserat på Python)
- IDE (Integrated Development Environment): PyCharm
- Virtual environment
  - Sätt att låta olika projekt använda olika python-versioner, och olika bibliotek
    - T.ex. kan nerladdade bibliotek för inloggning etc. kräva en viss python-version
  - Läs mer på http://docs.python-guide.org/en/latest/dev/virtualenvs/

Virtual environment

Installera

```
>> cd my_project_folder
>> virtualenv -p /usr/bin/python-3.4 my_venv_3.4
```

(En mapp skapas)

Aktivera (innan man kör igång med kodning)

```
>> source my_venv_3.4/bin/activate
```

Installera Flask

```
(my_venv_3.4) >> pip install flask
```

Flask

- Quickstart Flask:
  - http://flask.pocoo.org/docs/quickstart/#quickstart
- Tutorial för Flask:
  - http://flask.pocoo.org/docs/tutorial/#tutorial
- Flask + SQLAlchemy:
  - http://flask.pocoo.org/docs/0.10/patterns/sqlalchemy/
- Flask appcontext (för att hålla reda på databas-uppkoppling):
  - http://flask.pocoo.org/docs/appcontext/

Installation av andra bibliotek

Om inte redan aktiverad:

```
>> source /my_venv/bin/activate
```

Sedan

```
(my_venv) >> pip install requests
```

HTTP

HTTP – HyperText Transfer Protocol

- Kommunikationsprotokoll
  - Kan användas för att formattera och skicka web-sidor
  - Kan även användas till att skicka data
    - T.ex. formatterade text-strängar
    - Ex. {‘bok-titel’: ‘Krig och fred’,
      ‘pris’: 420
      ‘kapitel’: […]
      }

HTTP *header*

- Definierar operations-parametrar (metadata)
- Faktiska data ska skickas i *body*
  - Antingen html-sida, eller
  - JSON el. XML data

Exempel på anrop från klient (request)

GET **http://www.amazon.com/index.html** HTTP/1.1
Host: www.amazon.com
Connection: Close
(blank line)

Svar från server

HTTP/1.1 302 Found
Transfer-Encoding: chunked
Date: Fri, 27 Feb 2004 09:27:35 GMT
Content-Type: text/html; charset=iso-8859-1
Connection: close
Server: Stronghold/2.4.2 Apache/1.3.6 C2NetEU/2412 (Unix)
Set-Cookie: skin=; domain=.amazon.com; path=/; expires=Wed, 01-Aug-01 12:00:00 GMT
Connection: close
Location: http://www.amazon.com:80/exec/obidos/subst/home/home.html
Via: 1.1 xproxy (NetCache NetApp/5.3.1R4D5)

ed
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>302
Found</TITLE>
</HEAD><BODY>
<H1>Found</H1>
The document has moved <A
HREF="http://www.amazon.com:80/exec/obidos/subst/home/home.html"> here</A>.<P>
</BODY>
</HTML>
0

Request

- Vi har inte klienten ännu, utan kommer att anropa vår server-kod från terminalfönster
  - Använder biblioteket Requests, dvs.

```
>> source /my_venv/bin/activate
(my_venv) >> pip install requests
```

Snabbtesta http anrop på egen maskin

- Kör server-koden på 'localhost'
  http://127.0.0.1:5000
  - Högt portnummer för att undvika krock

- (Läs mer om reserverade IP-nummer, etc. http://www.comptechdoc.org/independent/networking/guide/netaddressing.html)

Request

- Två processer
  - Testar lokalt (samma dator): använd två olika terminal för att starta de två processerna
    - T.ex. PyCharm run (server) + PyCharms terminal (requests)

Request och session

- Server-sidan
  - Bearbetar ett request i taget
  - Potentiellt (parallellt) från flera användare
    - Klienterna (session) måste hållas isär

- Variabler/tillstånd som ska vara längre lagras på klient-sidan
  - T.ex. att användaren är aktiv/inloggad

Anrop (request) till server

- Anrop
  - URL_root = **'http://localhost:5000'**
  - resp_val = requests.get(URL_root + **'/food_menu/'**)

- Anrop (request) från klienter (Android-syntax)
  - HttpGet req = **new** HttpGet**(**"…/food_menu"**);**
  - response = client**.**execute**(**req**);**

På server-sidan

- Tas emot av server-kod (Flask-syntax)
  - @app.route('**/food_menu**', methods=[**'GET'**])
    def fetch_menu():
      if request.method == 'POST'**:**
        abort(405) #method not allowed
      else:
        menu = …
        return jsonify(menu), 200

Behövs inte, testas automatiskt pga methods=…

200 behövs inte, 200 skickas automatiskt

Några meddelanden från server

https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

- 2xx – success
- 400 – 'bad request' (syntaxfel på anropet?)
- 404 – fel URL / ingen sån resurs?
- 405 – ingen metod angivet i anropet / matchar inte tillåtna metoder för denna route i server-koden?

Vanliga request methods

- http://www.w3schools.com/tags/ref_httpmethods.asp
  - GET—Fetch data
  - POST—Ask process at URL to upload enclosed data
  - DELETE—Deletes the specified resource
  - HEAD—Same as GET but returns only HTTP headers and no document body

Andra http request methods

- http://www.w3schools.com/tags/ref_httpmethods.asp
  - PUT—Upload data (in request body) to the specified URI (~"file upload")
  - OPTIONS—Returns the HTTP methods that the server supports
  - CONNECT—Converts the request connection to a transparent TCP/IP tunnel

Response

- Man kan bygga upp en response steg för steg
- Men, oftast behövs bara en enkel return i Flask

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/')
def hello_world():
    return jsonify('Hello, World!')
```

methods=['GET'] är default

hamnar i body

JSON

Server - klient kommunikation mha JSON

- JSON (JavaScript Object Notation)
  - En samling namn-värde par
  - En sorterad lista av värden

JSON (JavaScript Object Notation)

- Plattade, dvs. serialiserade (JavaScript) objekt
  - Strukturerade data plattas till en sträng så att den kan skickas över nätet (som ju är seriell)
- Läs mer på t.ex. http://www.w3schools.com/json/

JSON exempel

```
{"menu":
      { "id": "file",
        "value": "File",
        "popup": {
              "menuitem": [
                    {"value": "New", "onclick": "CreateNewDoc()"},
                    {"value": "Open", "onclick": "OpenDoc()"},
                    {"value": "Close", "onclick": "CloseDoc()"}
              ]
        }
 }}
```

http://json.org/example.html

Till/från JSON

- Sever-sidan

  Flask.jsonify(**dict), se t.ex. http://stackoverflow.com/questions/13081532/how-to-return-json-using-flask-web-framework

- Klient-sidan (Android/java) är gson ett populärt bibliotek

  reader = new InputStreamReader(…);

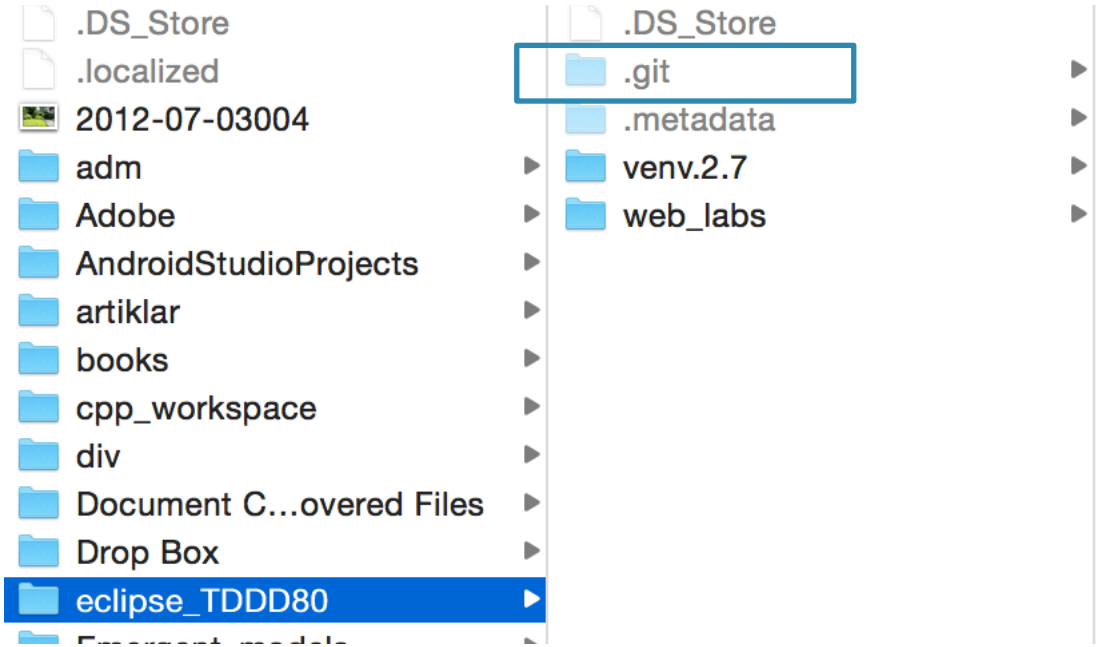
  Person p = gson.fromJson(reader, Person.class);

- Eller JSONObject

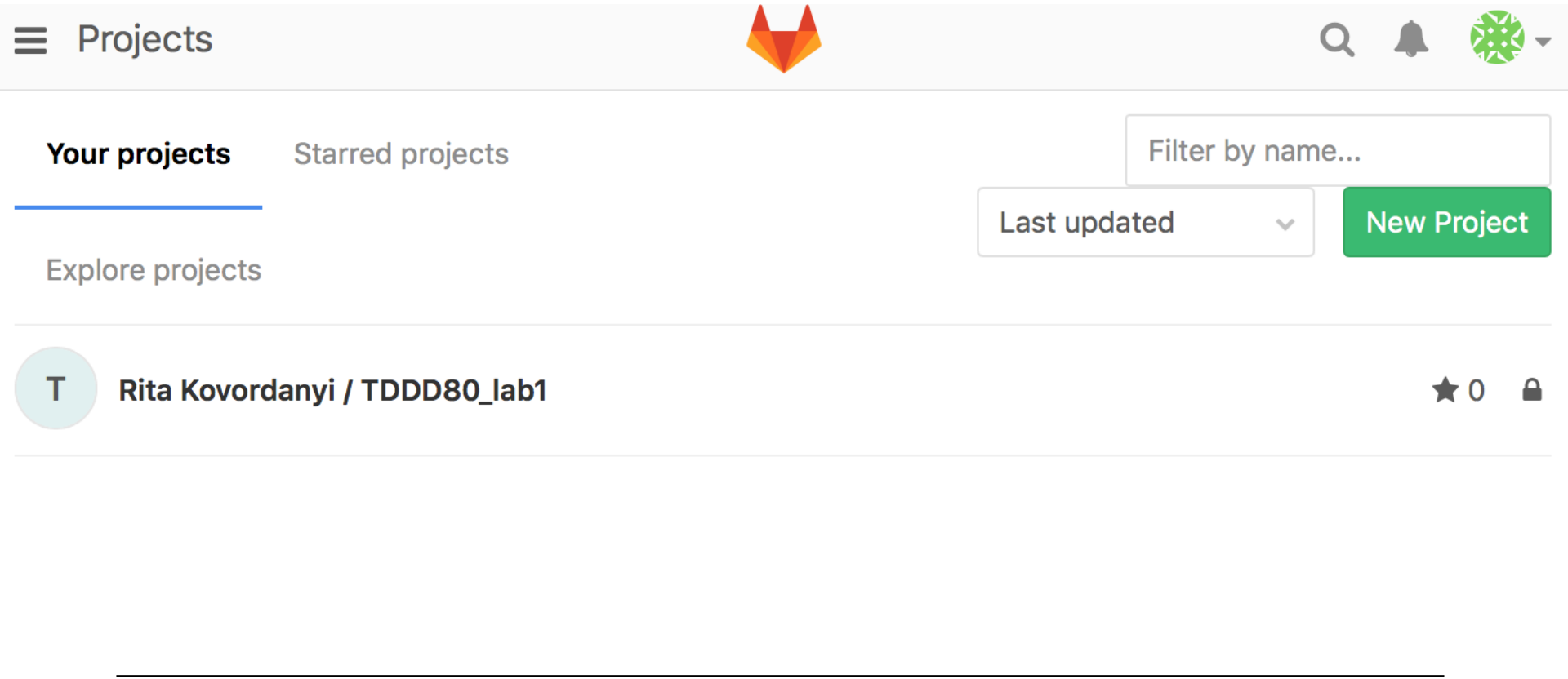
Versionshantering

DVCS – Distributed Version Control System

- Git distribuerat, dvs. finns lokalt på varje maskin

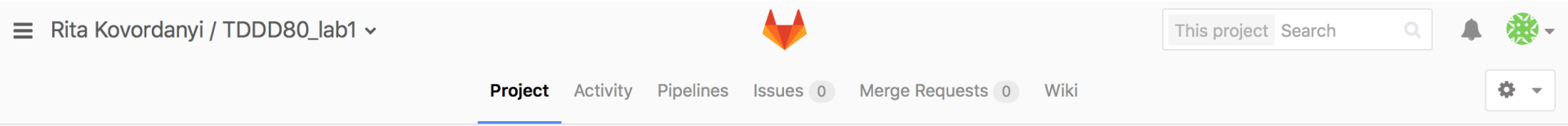


Initiera git-repo



Sätta upp ssh-nyckel

- Gitlab använder säker kommunikation till din dator mha SSH-nycklar
  - Generera ssh-nyckel på din dator
  - Lägg till *publika* myckeln til Gitlab
- Läs mer på: https://docs.gitlab.com/ce/ssh/README.html



Otherwise you can start with adding a README, a LICENSE, or a .gitignore to this project.

You will need to be owner or have the master permission level for the initial push, as the master branch is automatically protected.

Command line instructions

Git global setup

```
git config --global user.name "Rita Kovordanyi"
git config --global user.email "rita.kovordanyi@liu.se"
```

Create a new repository

```
git clone git@gitlab.ida.liu.se:ritko75/TDDD80_lab1.git
cd TDDD80_lab1
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Existing folder or Git repository

```
cd existing_folder
git init
git remote add origin git@gitlab.ida.liu.se:ritko75/TDDD80_lab1.git
git add .
git commit
git push -u origin master
```

Remove project

Command line instructions

Git global setup

```
git config --global user.name "Rita Kovordanyi"
git config --global user.email "rita.kovordanyi@liu.se"
```

Create a new repository

```
git clone git@gitlab.ida.liu.se:ritko75/TDDD80_lab1.git
cd TDDD80_lab1
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Existing folder or Git repository

```
cd existing_folder
git init
git remote add origin git@gitlab.ida.liu.se:ritko75/TDDD80_lab1.git
git add .
git commit
git push -u origin master
```

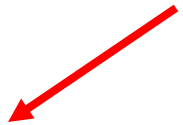
Git arbetsflöde (lokalt på egen dator)

- Välj vilka filer som ska "sparas" i repot (läggs till i intern index, dvs. stage:as)

  >> git add minFil

- "Spara" lokal version

  >> git commit minFil –m "refaktorerat koden"



- Om ni inte skriver kommentar, hamnar ni i vim-editorn (för att skriva längre kommentar)

Ifall ni hamnat i vim-editorn

- Skriv en commit message
- Sedan, spara och gå ur, med någon av dessa kommandon:
  - Esc :wq (write + quit)
  - Esc :x (save + exit)
  - Esc ZZ

Remote repo (i Gitlab)

- Dra ner senaste från remote repo (+ försök merga)

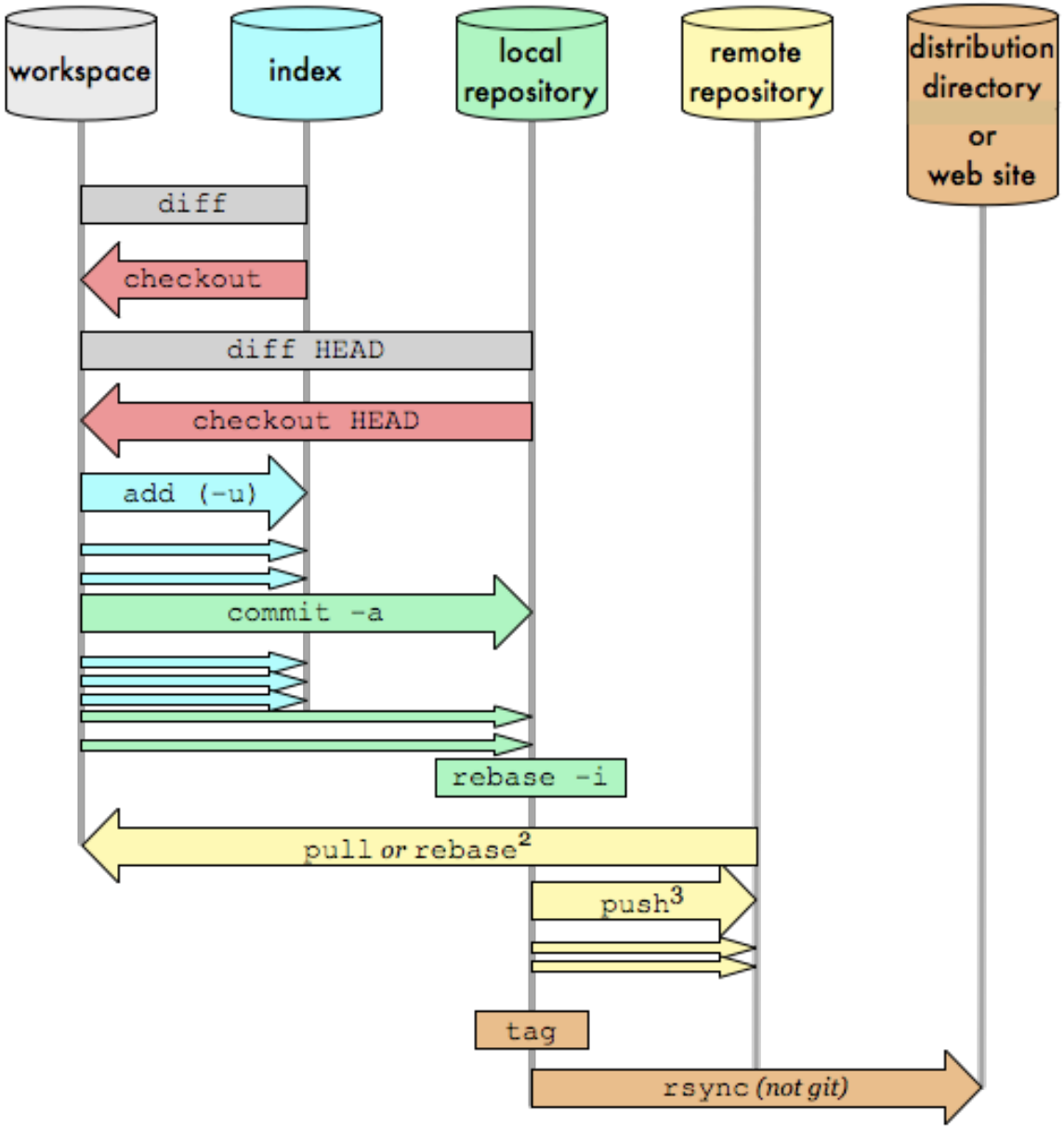
  >> git pull origin

- Tänk på att konflikter kan ha uppstått i koden om du jobbar i team
  - Måste då kanske göra en manuell ‘merge’ + testning före en push
- Lägg upp i remote repo

  >> git push –u origin master

A[1] Git Workflow

http://osteele.com



1 Git is a workflow construction toolkit. This is just one of many possible workflows.
2 With git-svn: "`git svn rebase`". With git-p4: "`git p4 rebase`"
3 With git-svn: "`git svn dcommit`"

Distribuerat arbetsflöde

- Varje utvecklare ansvarig för att hålla gemensamt repo körbar
  - Små arbetsbitar
    - Jobba mot eget lokalt repo
  - När klar (max några timmars arbete)
  - Dra ner senaste kod från gemensamt repo (remote)
    - Merge och fixa conflicts
  - Pusha all kod (nu mergad och fullt testad) till remote

Bra länkar

- Git tänk
  - https://git-scm.com/book/en/v2/Distributed-Git-Distributed-Workflows
  - https://www.youtube.com/watch?v=v40b3ExbM0c
- Git kommandon
  - https://www.youtube.com/watch?v=Y9XZQO1n_7c

Informationssökning: tips

Tar tid att hitta de rätta söktermerna

- Arbetsflöde:
  - Skriv in söktermer
  - Ögna igenom första sidan, läs bara Googles sammanfattning
    - Klicka om intressant sammanfattning
  - Efter 2-5 klickar
  - Fick du svar på din fråga
  - Nej → prova med andra söktermer

rita.kovordanyi@liu.se

www.liu.se

LINKÖPINGS UNIVERSITET