

Computational Tools for Big Data

Introduction

What?

- ▶ Computational tools
- ▶ Large (big) data sets
- ▶ Not a course on theory
- ▶ Not 5 easy ECTS points

What?

Assignment 1

1. UNIX Shell, git, Amazon EC2
2. Python
3. Libraries for Python
4. (Sparse) DBSCAN

Assignment 2

5. SQL and NoSQL
6. Graph Databases
7. Streaming Algorithms
8. MapReduce

Assignment 3

9. Apache Spark
10. Deep Learning
11. Feature Hashing and LSH
12. Vowpal Wabbit

Assignment 4

13. Your choice

Why?

- ▶ Students at DTU have solid mathematical foundation
- ▶ But often lack practical knowledge
- ▶ “Big Data” and all that...

How?

- ▶ The course will be hands on
- ▶ Pre-lecture material on the web
- ▶ Short lectures in class
- ▶ Do exercises in class
- ▶ Hand in homework
 - ▶ Same as in-class exercises
- ▶ Peer-feedback

Where?

- ▶ **Lectures:** Building 306, Auditorium 33
- ▶ **Group work:**
 - ▶ *Building 306, 1st floor, group room 1*
 - ▶ Building 324, ground floor, 003
 - ▶ Building 324, ground floor, 004
 - ▶ Building 324, ground floor, 005
 - ▶ Building 324, ground floor, 008



David Kofoed Wind
PhD student in Cognitive Systems
Responsible for the course
Main lecturer



Ulf Aslak Jensen
Master student
Assistant teacher



(Rasmus) Malthe Jørgensen
B.Sc. in Physics
Assistant teacher



Finn Årup Nielsen
Senior Researcher in Cognitive Systems
Lectures on Python

Formalities

- **Prerequisites**
 - We expect that you can write code
- The course is graded on the **7-step scale**
- You get **5 ECTS**
- You hand in **4 assignments**
- You give **feedback on assignments of other students**

Assignments

- ▶ Each assignment consists of all in-class exercises from the relevant weeks. Assignment 4 is special.
- ▶ The assignments are done either alone or in groups of 2-3 people.
- ▶ Hand-in deadlines
 - ▶ 28th of September
 - ▶ 26th of October
 - ▶ 30th of November
 - ▶ 7th of December
- ▶ Peer evaluation deadlines
 - ▶ 5th of October
 - ▶ 2nd of November
 - ▶ 4th of December
 - ▶ 14th of December

Assignment 4

- ▶ The fourth assignment is a video
- ▶ You (and your group) have to make a video
- ▶ The video has to explain a relevant concept/tool
- ▶ The concept/tool must be something not covered in the course
- ▶ Think of it as “What would I teach if I had another week”
- ▶ We will put examples of good and bad subjects on the website

Course website

- ▶ The course has the website:
www.toolsforbigdata.com
- ▶ Each week will have a page with
 - ▶ Description of the curriculum for the week
 - ▶ Exercises to solve
 - ▶ Reading material
- ▶ Reading the material is strongly encouraged
- ▶ Each assignment will be published on the site

Peer evaluation

- ▶ After handing in an assignment, you will each get a number of assignments from other students (probably around 3-4).
- ▶ You evaluate each of these using a questionnaire provided by us.
- ▶ Your final grade depends on:
 - ▶ How good your own assignments are
 - ▶ They will be read by a number of students
 - ▶ We will read some of them
 - ▶ How good you are at evaluating others
 - ▶ We read your feedback
 - ▶ You can flag bad feedback
- ▶ Peer evaluation is done using peergrade.io

Questions?

Computational Tools for Big Data

Week 1: **The UNIX Shell**, Version Control, Amazon EC2

Commands

Standard tools

chmod, apt-get,
ssh, wget, curl

Working with files

head, tail, less, cat, tr
grep, sed, cut, sort, uniq

Editors

nano

Covered here in lecture

Standard tools

chmod, apt-get,
ssh, wget, curl

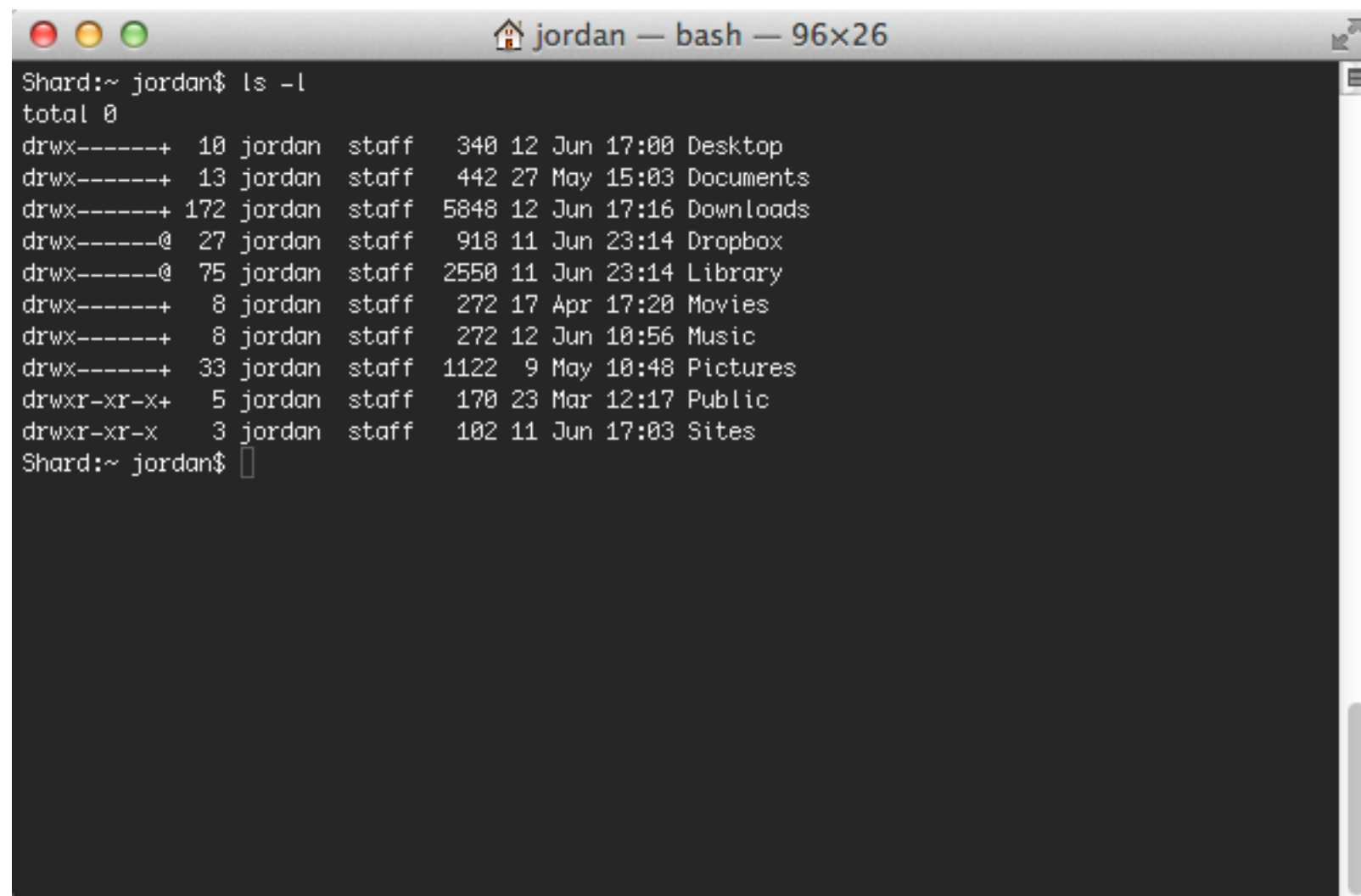
Working with files

head, tail, less, **cat**, **tr**
grep, **sed**, **cut**, **sort**, **uniq**

Editors

nano

Terminal



A screenshot of a macOS Terminal window. The title bar shows a home icon, the name 'jordan', and the command 'bash' followed by the window size '96x26'. The terminal content shows the user 'jordan' at the 'Shard' prompt running the command 'ls -l'. The output is a long-format listing of files in the home directory, including Desktop, Documents, Downloads, Dropbox, Library, Movies, Music, Pictures, Public, and Sites. Each line shows permissions, size, owner, group, size in bytes, date, time, and filename.

```
Shard:~ jordan$ ls -l
total 0
drwx-----+ 10 jordan  staff   340 12 Jun 17:00 Desktop
drwx-----+ 13 jordan  staff   442 27 May 15:03 Documents
drwx-----+ 172 jordan  staff  5848 12 Jun 17:16 Downloads
drwx-----@ 27 jordan  staff   918 11 Jun 23:14 Dropbox
drwx-----@ 75 jordan  staff  2550 11 Jun 23:14 Library
drwx-----+  8 jordan  staff   272 17 Apr 17:20 Movies
drwx-----+  8 jordan  staff   272 12 Jun 10:56 Music
drwx-----+ 33 jordan  staff  1122  9 May 10:48 Pictures
drwxr-xr-x+  5 jordan  staff   170 23 Mar 12:17 Public
drwxr-xr-x   3 jordan  staff   102 11 Jun 17:03 Sites
Shard:~ jordan$
```

SSH

Secure Shell (**SSH**) is a cryptographic network protocol for **secure** data communication, **remote command-line login**, **remote command execution**, and other secure network services between two networked computers.

```
~ ssh dawi@hald.gbar.dtu.dk
The authenticity of host 'hald.gbar.dtu.dk (192.38.95.41)' can't be established.
RSA key fingerprint is 78:74:43:13:9d:23:02:95:78:18:48:24:47:cf:6d:05.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'hald.gbar.dtu.dk,192.38.95.41' (RSA) to the list of known hosts.
Password:
~gray1(dawi) $ pwd
/zhome/5b/c/51358
```

We could do a whole lecture on RSA-keys for SSH.

On Windows

Use Putty and SSH to the GBar

Pipes and redirections

- > Redirect output from a command to a file on disk.
- >> Append output from a command to a file on disk.
- < Read a command's input from a disk on file.
- | Pass the output of one command to another for further processing.

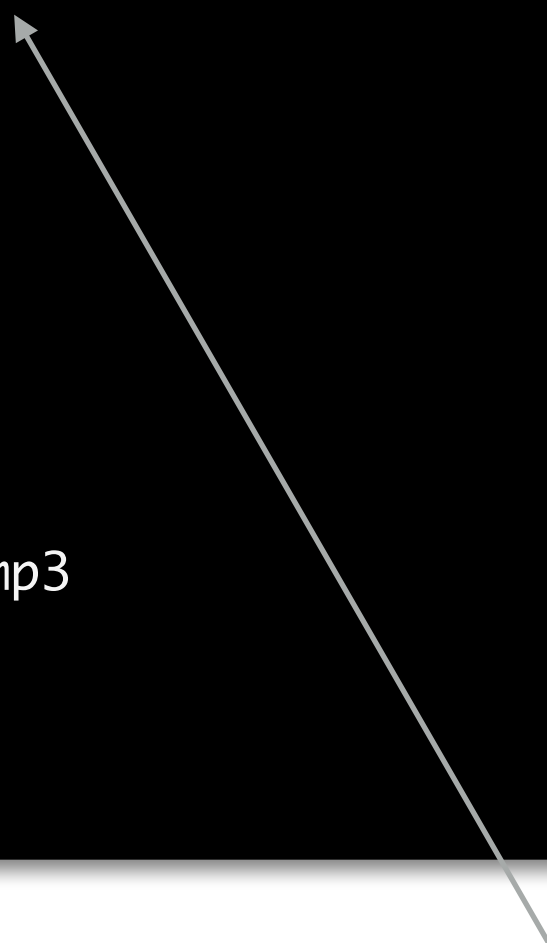
cat

The **cat** utility will **output** the contents of a specific file and can be used to **concatenate** and print files.

If no file is specified, **cat** will read from standard input.

cat

```
~ cat > temp1  
This will go in temp1  
~ cat - > temp2  
This will go in temp2  
~ cat temp1  
This will go in temp1  
~ cat temp1 temp2  
This will go in temp1  
This will go in temp2  
~ cat temp1 temp2 > temp3  
~ cat temp3  
This will go in temp1  
This will go in temp2
```



Make line break and use Ctrl + c to end

cat

```
~ cat temp3
This will go in temp1
This will go in temp2
~ cat >> temp3
This will also go in temp3
~ cat temp3
This will go in temp1
This will go in temp2
This will also go in temp3
~ cat -n temp3
  1 This will go in temp1
  2 This will go in temp2
  3 This will also go in temp3
```

cut

The **cut** utility cuts out selected portions of each line from each file and writes them to the standard output.

If no file arguments are specified, **cut** reads from the standard input.

cut

```
~ cat > temp
This is a temp file
1234567890
With some content
~ cut -c 4 temp
s
4
h
~ cut -c 4,6 temp
si
46
hs
```

cut

```
~ cut -c 4-6 temp
s i
456
h s
~ cut -c -6 temp
This i
123456
With s
~ cut -d ' ' -f 2 temp
is
1234567890
some
~ cut -d ' ' -f 2,3 temp
is a
1234567890
some content
```

tr

Translates a set of characters to
another set of characters.

tr only takes input from standard input.

tr

```
~ cat > temp
This is a temp file
1234567890
With some content
~ tr 'a-z' 'A-Z' < temp
THIS IS A TEMP FILE
1234567890
WITH SOME CONTENT
~ tr ' ' '\n' < temp
This
is
a
temp
file
1234567890
With
some
content
```

sort

The utility **sort** writes the file sorted to standard output.

sort

```
~ cat > temp
1001 Søren
33 Emil
25 Helge
1001 David
~ sort temp
1001 David
1001 Søren
25 Helge
33 Emil
~ sort -n temp
25 Helge
33 Emil
1001 David
1001 Søren
```

```
~ sort -k 2 temp
1001 David
33 Emil
25 Helge
1001 Søren
~ sort -k 2 -r temp
1001 Søren
25 Helge
33 Emil
1001 David
~ sort -k 1,1 -k 2,2 temp
1001 David
1001 Søren
25 Helge
33 Emil
~ sort -k 1,1 -k 2,2r temp
1001 Søren
1001 David
25 Helge
33 Emil
```

sort

```
~ cat > temp2
David,1
Helge,10
Søren,5
~ sort -t ',' -nk 2 temp2
David,1
Søren,5
Helge,10
```

uniq

The **uniq** utility reads the specified input file comparing adjacent lines, and writes a copy of each unique input line to the output file.

If input_file is absent, the standard input is read.

uniq

```
~ cat > temp
1
1
2
3
2
~ uniq temp
1
2
3
2
~ sort -n temp | uniq
1
2
3
~ sort -n temp | uniq -c
  2 1
  2 2
  1 3
```

grep

The **grep** utility searches any given input files, selecting lines that match a pattern.

By default, a pattern matches an input line if the regular expression in the pattern matches the input line.

An empty expression matches every line. Each input line that matches the pattern is written to the standard output.

grep

```
~ cat > temp
big
bad bug
bag
bigger
boogy
~ grep big temp
big
bigger
~ grep b.g temp
big
bad bug
bag
bigger
~ grep "b.*g" temp
big
bad bug
bag
bigger
boogy
~ grep -w big temp
big
```

. is any character
x* is zero or more 'x'

grep

```
~ cat > temp1
this is a line in the first file
~ cat > temp2
this is a line in the second file
~ grep this temp*
temp1: this is a line in the first file
temp2: this is a line in the second file
```

Display N lines after match
`grep -A <N> "string" FILENAME`

Display N lines before match
`grep -B <N> "string" FILENAME`

Search recursively in folders
`grep -r "string" *`

Invert match
`grep -v "string" FILENAME`

Count number of matches
`grep -c "string" FILENAME`

Display only the file names
`grep -l "string" FILENAME`

sed

The **sed** utility reads the specified files, or the standard input if no files are specified.

It modifies the input as specified by a list of commands. The input is then written to the standard output.

sed

```
~ cat > temp
one two three one two three
four three two one
one hundred
~ sed 's/one/ONE/' temp
ONE two three one two three
four three two ONE
ONE hundred
~ sed 's/[a-z]*/LOL/' temp
LOL two three one two three
LOL three two one
LOL hundred
~ sed 's/[a-z]*/(&)/' temp
(one) two three one two three
(four) three two one
(one) hundred
~ sed 's/[a-z]*/(&)/g' temp
(one) (two) (three) (one) (two) (three)
(four) (three) (two) (one)
(one) (hundred)
```

[a-z] is any lower case character

Combinations

Say you want to count the number of 0's and 1's in a file like the one below.

```
~ head temp
Id,Prediction
17000,0
17001,0
17002,1
17003,1
17004,1
17005,0
17006,0
17007,0
17008,0
~ cut -d ',' -f 2 temp | grep -c '0'
1983
~ cut -d ',' -f 2 temp | grep -c '1'
2075
```

Computational Tools for Big Data

Week 1: The UNIX Shell & **Version Control** & Amazon EC2



git

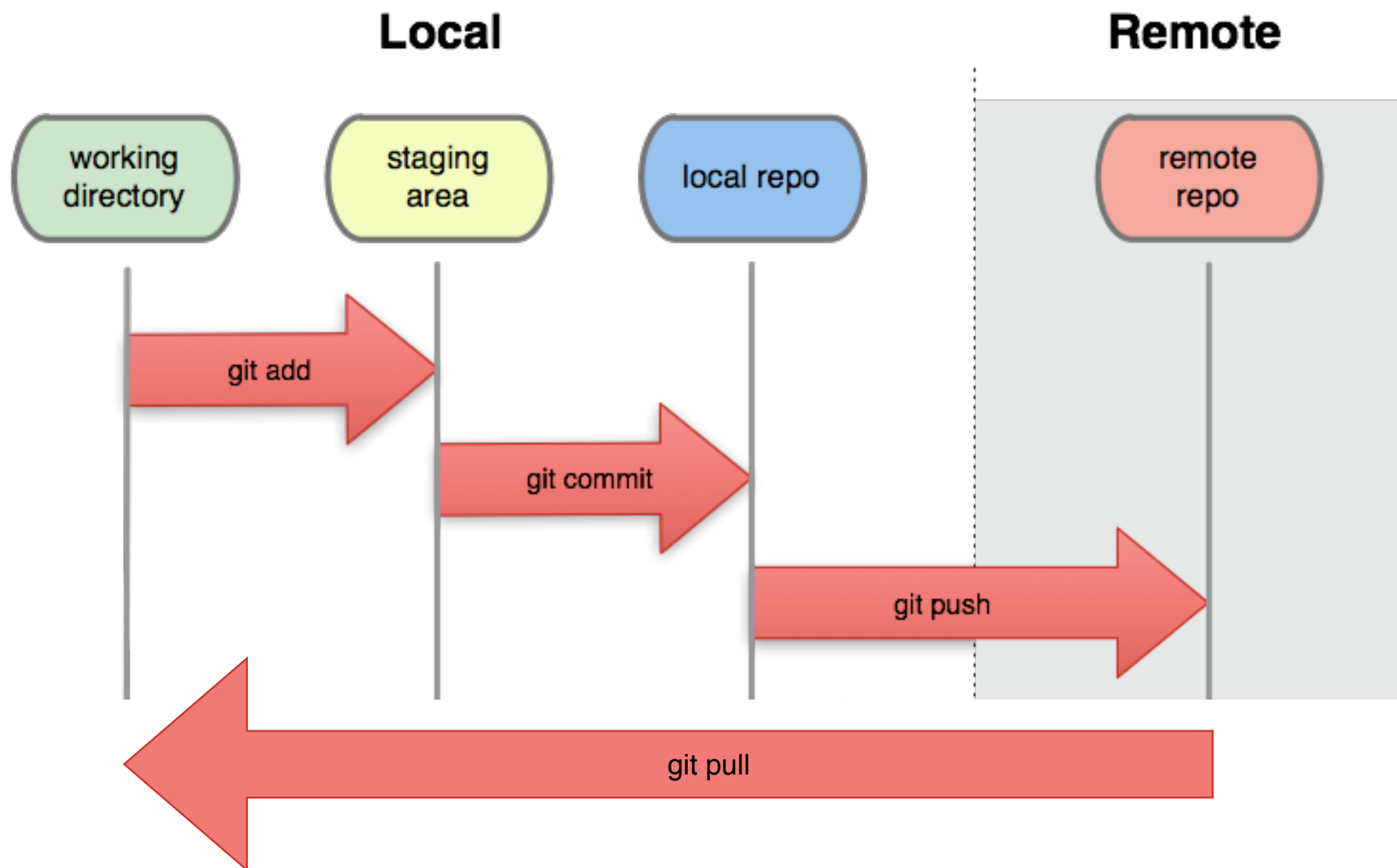
“Git is a distributed revision control and source code management system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows.”



github

SOCIAL CODING

GitHub is a web-based hosting service for software development projects that use the Git revision control system.



```
→ ~ mkdir temp_git
→ ~ cd temp_git
→ temp_git cat > file1
This is in file1
→ temp_git cat > file2
This is in file2
```

```
→ ~ mkdir temp_git
→ ~ cd temp_git
→ temp_git cat > file1
This is in file1
→ temp_git cat > file2
This is in file2

→ temp_git git init
Initialized empty Git repository in /Users/dawi/temp_git/.git/
→ temp_git git:(master) ✗ git add file1 file2
→ temp_git git:(master) ✗ git commit -m "first commit"
2 files changed, 2 insertions(+)
create mode 100644 file1
create mode 100644 file2
```

→ ~ mkdir temp_git

→ ~ cd temp_git

→ temp_git cat > file1

This is in file1

→ temp_git cat > file2

This is in file2

→ temp_git git init

Initialized empty Git repository in /Users/dawi/temp_git/.git/

→ temp_git git:(master) x git add file1 file2

→ temp_git git:(master) x git commit -m "first commit"

2 files changed, 2 insertions(+)

create mode 100644 file1

create mode 100644 file2

→ temp_git git:(master) git status

On branch master

nothing to commit, working directory clean

→ temp_git git:(master) git remote add origin <https://github.com/utdiscant/ctfds.git>

```

→ ~ mkdir temp_git
→ ~ cd temp_git
→ temp_git cat > file1
This is in file1
→ temp_git cat > file2
This is in file2

→ temp_git git init
Initialized empty Git repository in /Users/dawi/temp_git/.git/
→ temp_git git:(master) x git add file1 file2
→ temp_git git:(master) x git commit -m "first commit"
2 files changed, 2 insertions(+)
create mode 100644 file1
create mode 100644 file2

→ temp_git git:(master) git status
On branch master
nothing to commit, working directory clean
→ temp_git git:(master) git remote add origin https://github.com/utdiscant/ctfds.git

→ temp_git git:(master) git push -u origin master
Username for 'https://github.com': utdiscant
Password for 'https://utdiscant@github.com':
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 284 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/utdiscant/ctfds.git
* [new branch]      master -> master
Branch master set up to track remote branch master from origin.

```

```
→ n-62-14-11(dawi) $ git clone https://github.com/utdiscant/ctfds.git
Initialized empty Git repository in /zhome/5b/c/51358/git_temp/ctfds/.git/
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 4 (delta 0)
Unpacking objects: 100% (4/4), done.
```

```
→ n-62-14-11(dawi) $ ls
ctfds
```

```
→ n-62-14-11(dawi) $ cd ctfds/
```

```
→ n-62-14-11(dawi) $ ls
file1  file2
```


Computational Tools for Data Science

Week 1: The UNIX Shell, Version Control, **Amazon EC2**



Free for 12 months



Amazon EC2

Resizable compute capacity in the Cloud.

[Learn More »](#)

750 hours per month of Linux, RHEL, or SLES t2.micro instance usage

750 hours per month of Windows t2.micro instance usage

Run one instance all month or multiple instances for a fraction of the month



Amazon S3

Highly scalable, reliable, and low-latency data storage infrastructure.

[Learn More »](#)

5 GB of Standard Storage

20,000 Get Requests

2,000 Put Requests



Amazon DynamoDB

Fast and flexible NoSQL database with seamless scalability.

[Learn More »](#)

25 GB of Storage

25 Units of Write Capacity

25 Units of Read Capacity

Enough to handle up to 200M requests per month

Where again?

- ▶ **Group work:**

- ▶ *Building 306, 1st floor, group room 1*
- ▶ Building 324, ground floor, 003
- ▶ Building 324, ground floor, 004
- ▶ Building 324, ground floor, 005
- ▶ Building 324, ground floor, 008