**Odin Project – FullStack Web Stuff**

Internet:

Backbone/infrastructure of the Web

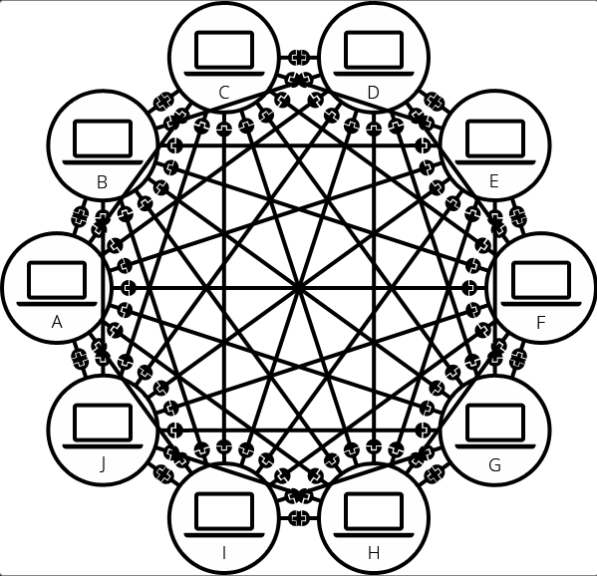
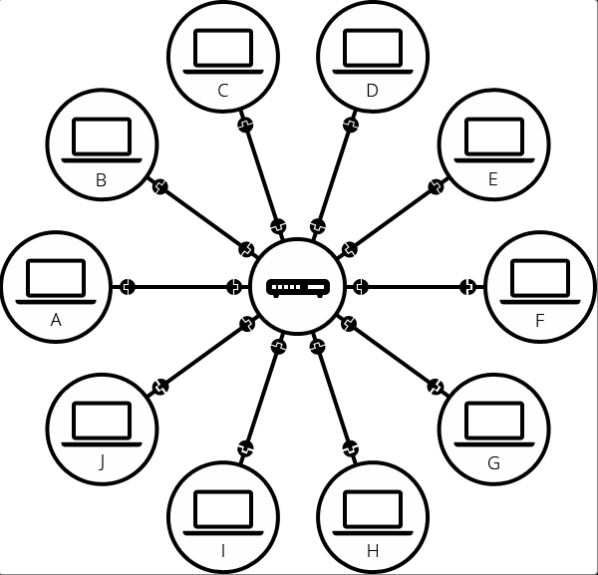
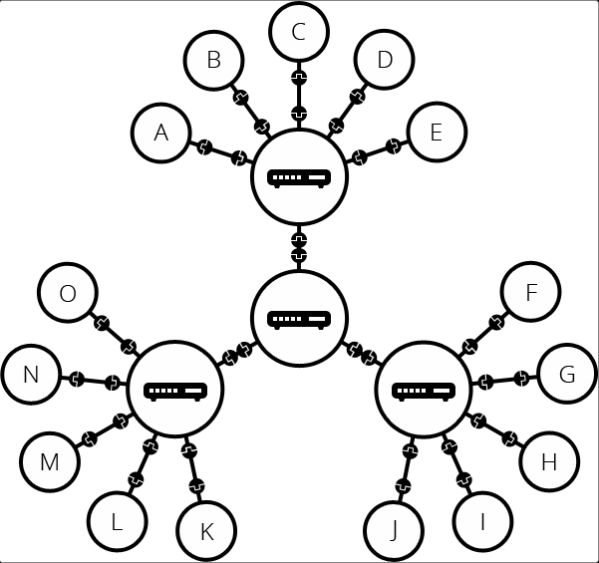
A large network of computers which communicate all together

Network:

Computers connected either physically or wirelessly

Router:

Act as the signaler (at a railway station), A -> router -> C,

=>=>

Modem:

Turns the information from our netowrk / router into information manageable by the telephone infrastructure and vice versa

ISP:

Internet Service Provider - Manages some special routers that are linked together and can also access other ISP’s routers

IP address:

Internet Protocol – Any Compute linked to a network has a unique address, made of four sets of numbers

Domain Name:

IP address’s alias as a human readable name

Client:

Web user’s internet-connected device & web-accessing software / browser

Server:

Computer that store webpages, sites or apps. When access by a client, a copy of the webpage is downloaded from the server onto the cilent’s local machine to display

Packets:

When data is sent across the web, its is sent in thousands of samll chunks.

Easier to replace small chunks of dropped / corrupted packages

Packets can be routed along different paths, exchange faster, allow more user to download the same website at the same time

Web page:

A document which can be displayed in a web broswer, written in the HTML language, embed with 1) Style information - look & feel 2) Scripts - interactivitiy 3) Media - images, sounds, videos

Website:

A collection of web pages which are grouped together and usually connected together

Web server:

A computer that hosts a website on the internet, hosting all the web pages and their supporting files

Web browser:

A program on a device to visit & displays website

Search engine:

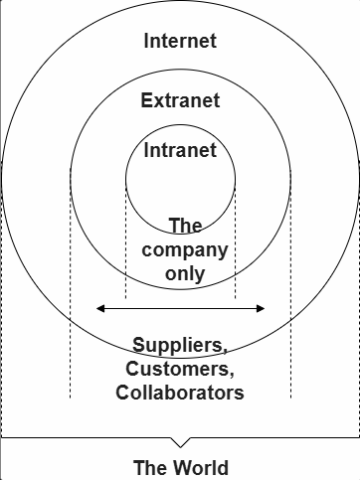
A web service that helps you find other webpages

DNS request:

When you type a url into a browser ([www.google.com](http://www.google.com/)), “ [www.google.com](http://www.google.com/).” the end “dot” represents the root of the internest’s namespace

The browers ask the OS for the IP, if the OS doesn’t already know (e.g. IP in cache), the OS query the *Resolving Name Server*. The *RNS* then ask the *Root Name Server. The Resolving Name Server* then ask *TLD (Top Level Domain) Name Server.* Then the Resolving Name Server ask the *ANS (Authoratative Name Server)* and reply with an IP address, the Resolving Name Server takes the information, store it in cache and reply the OS with the IP.

Intranet & Extranet:



**Terminal Shortcut / Command**

*Ctrl + Alt + T* => Open Terminal

*Ctrl + Shift + C* => Copy (in Terminal)

*Ctrl + Shift + V* => Paste (in Terminal)

*Tab* => auto-complete command

ls => list all files and folders of current directrory

ls /usr local/ => list all files of a different directory

ls –a => list all files both visibe & hidden

ls –l => list all with long form / detail information

ls –lh => list all with long form & human readable file size

ls –lhs => list all with long form & human readable file size & sorted by file size, large on top

ls –lt => list all with long form & sorted by last modified time, new on top

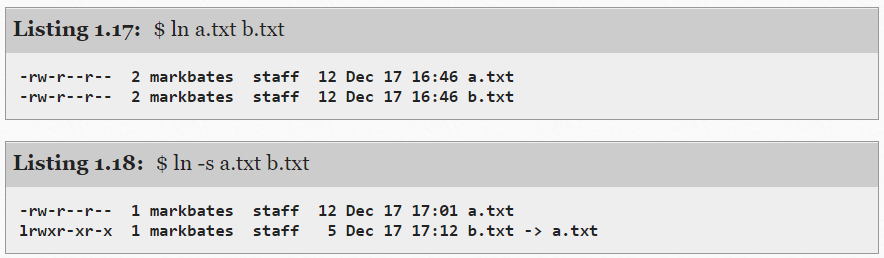
ls –lr => list all with a reversed result

m

ln a.txt b.txt => link a (source) to b (target) | target file will remain even when the source is deleted

ln –f a.txt b.txt => force link file a to file b (when b already exist)

ln –s a.txt b.txt => symbolic link a to b | can link directory as well as normal files



cd ~/Document => change directories to Document

.. => back up one directory

cd => to home directory

mkdir abc => create a directory abc

mkdir –p a/b/c => create intermediate directories

mkdir –v a => create directory & print the result to the console

cp a.txt b.txt => copy a (source) to b (target)

cp a.txt b.txt dd => copy multiple files to target directory

cp -v a.txt b.txt => copy a to b then print result to console

cp -R foo bar => copy directory foo to bar

cp -f a.txt b.txt => force copy a to b (when there are permission errors)

cp -i a.txt b.txt => ask for confirm before overwriting files

rm a.txt => delete a

mv a.txt b.txt => move a to b (basically copy a to b, then delete b)

ls -a | grep \_ => redirect output of [ls -a] to [grep \_]

ls -a | grep \_ > a.txt=> redirect output of [ls -a] to [grep \_], then write output to file

< => read data from a file

cat file1.txt => print out the content of file1 to terminal

touch xxx.xxx => create file(s) xxx.xxx

pwd => show present working directory

pwd –P => show actual physical path

* Commands related to a remote repository:
  + git clone git@github.com:USER-NAME/REPOSITORY-NAME.git
  + git push or git push origin main (Both accomplish the same goal in this context)
* Commands related to the workflow:
  + git add .
  + git commit -m "A message describing what you have done to make this snapshot different"
* Commands related to checking status or log history
  + git status
  + git log

The basic Git syntax is program | action | destination.

git add . is read as git | add | ., where the period represents everything in the current directory;

* git commit -m "message" is read as git | commit -m | "message"; and
* git status is read as git | status | (no destination).