CSC309 *Programming on the Web*

week 8: web server hosting

Amir H. Chinaei, Spring 2017

Office Hours: M 3:45-5:45 BA4222

ahchinaei@cs.toronto.edu
http://www.cs.toronto.edu/~ahchinaei/

review

- **so far:**
  - developed front-end (aka. **client**)
  - developed back-end (aka. **server**)
  - front-end and back-end **interaction**
- **what's next?**
  - for development, you used your own machine, but
  - your server needs to interact with clients over the world
  - **web server hosting**
  - **domain name system**

types of hosting

- **shared hosting**
- **virtual private hosting**
- **dedicated hosting**
- **collocated hosting**
- **in house hosting**
- **cloud-based hosting**

shared

- your app shares space on a server that hosts other apps too
- super server with almost all resources shared
- software tools already installed
- advantages
  - inexpensive
- disadvantages:
  - security threats
  - lack of control to configure software tools (os, db, etc.)
- **good for getting your feet wet**

virtual private hosting

- *physically* a shared server, virtually a private one
- advantages
  - software tools can be configured
  - performance of other apps do not affect yours
  - more security
- disadvantages:
  - more expensive
- **good for many online businesses**

dedicated

- a complete physical server is dedicated to your app
- advantages
  - you have full control on configuration
- disadvantages:
  - most expensive option
  - lack of control on hardware

collocated

- the server is owned by you located in a data centre
- benefitting from
  - fast and redundant network connection
  - other facility features, such as physical security, power, cooling system

- advantages
  - you have full control on both software and hardware
- disadvantages:
  - you are responsible to control everything: backup, maintaining software/hardware, etc.

in house

- self-hosting
- you purchase
  - the server
  - cooling system
  - power
  - internet bandwidth

- you control everything
  - backup, recovery
  - maintaining software/hardware
  - cooling system
  - power, batteries, etc.

cloud-based

- several servers share resources
- the idea is to increase resources as need grows

- advantages
  - scalability
  - redundancy (reliability)

important factors

- host:
  - reliability (backup, minimum down-time, and recovery)
  - functionality (bandwidth, traffic reports, better logging)
  - scalability
  - tech support

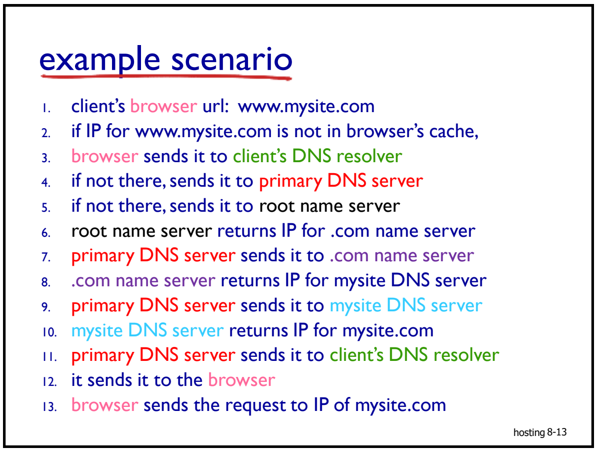
- your app:
  - amount of data transfer per month
  - required software tools/libraries/services
    - amount of email support

what's next?

- your app server needs an IP address, to which
- clients can send http requests, via a mapped
- **domain name**

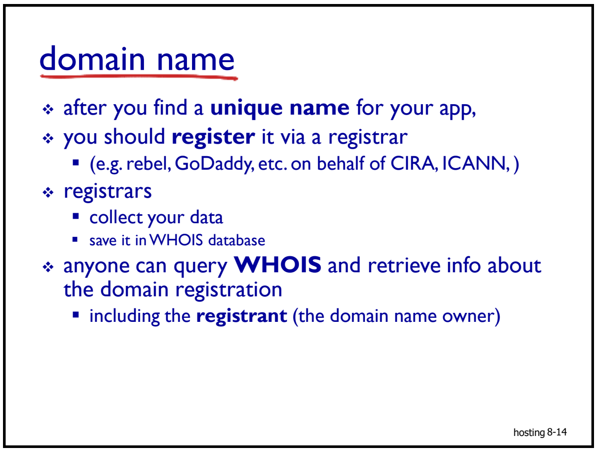
- the mapping is called **resolution** and it's done by
- **domain name system**

DNS resolution



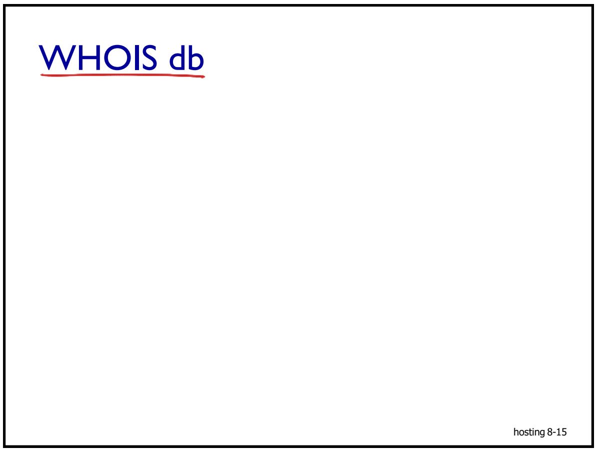
example scenario

1. client’s browser url: www.mysite.com
2. if IP for www.mysite.com is not in browser’s cache,
3. browser sends it to client’s DNS resolver
4. if not there, sends it to primary DNS server
5. if not there, sends it to root name server
6. root name server returns IP for .com name server
7. primary DNS server sends it to .com name server
8. .com name server returns IP for mysite DNS server
9. primary DNS server sends it to mysite DNS server
10. mysite DNS server returns IP for mysite.com
11. primary DNS server sends it to client’s DNS resolver
12. it sends it to the browser
13. browser sends the request to IP of mysite.com

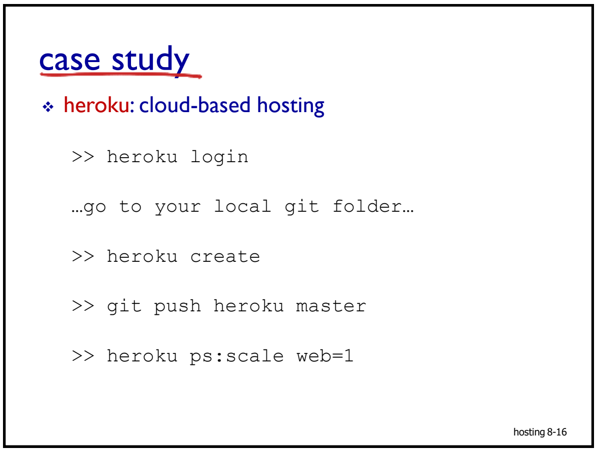


domain name

- after you find a **unique name** for your app,
- you should **register** it via a registrar
  - (e.g. rebel, GoDaddy, etc. on behalf of CIRA, ICANN,)
- registrars
  - collect your data
  - save it in WHOIS database
- anyone can query **WHOIS** and retrieve info about the domain registration
  - including the **registrant** (the domain name owner)



WHOIS db



case study

- heroku: cloud-based hosting

```
>> heroku login

…go to your local git folder…

>> heroku create

>> git push heroku master

>> heroku ps:scale web=1
```