Infrastruktur, arkitektur, stackar, miljö m.m

Agenda

- (Digital) infrastruktur
- Arkitekturer och arkitekturprinciper
- Miljöer
  - Stackar
  - Utvecklingsmiljöer
- Relationsdatabaser
  - MySql

Varför?

- Som webbutvecklare måste man vara insatt i och behärska mycket…
- Nya teknologier, koncept och verktyg kommer hela tiden.
- Allt från programmeringsspråk, operativsystem, databaser, verktyg i samband med programmering till koncept;
  - Java, C#, Php, Python, JavaScript, CSS, html 5, SQL, Objektorienterad programmering, REST, CRUD, xml, json, bson, relationsdatabaser, NoSql-databaser, linux, design patterns, Ux/Ui, web sockets, node.js, Laravel, FuelPhp, Pylons, Pyramid, Django, Ruby on Rails, Internet Of Things, Web Of Things, Play, Grunt, Gulp, npm, scrum, backlog, sprint, Apache, load balancer, nginx, docker, git, bitbucket, API, Oauth 2, cache, http, postman, curl, microservices, PAAS, IAAS, SAAS, VPS, AWS, IOT, WOT, UML, ORM, DAL…, …
  - Listan kan göras mycket läääää-ääängre…


ORACLE
DATABASE
Microsoft SQL Server
PostgreSQL
MariaDB
TERADATA
IBM
METEOR
UNDERSCORE.JS
Knockout.
npm
Sammy.js
browserify
Breeze
BACKBONE.JS
express
ANGULARJS
ember
DERBY
YO
node
Cappuccino
TypeScript
JayData
mongoDB
CodeIgniter
CakePHP
Ruby
C#
SQL
COBOL
Lua
/dev/block/platform/sdhci-tegra.3/by-name/system
241888 197864 44024 82% /system
/dev/block/platform/sdhci-tegra.3/by-name/userdata
30182852 1241724 28941128 4% /data
/dev/block/platform/sdhci-tegra.3/by-name/cache
170320 4260 166060 3% /cache
/dev/block/platform/sdhci-tegra.3/by-name/pdsb
1976 106 1870 5% /pds
/dev/fuse 30182852 1241724 28941128 4% /mnt/s
/ $ ll
drwxr-xr-x 14 root root 0 Mar 18 14:14 .
drwxr-xr-x 14 root root 0 Mar 18 14:14 ..
drwxr-xr-x 3 root root 0 Mar 18 14:14 acc
drwxrwx--- 4 system cache 4096 Mar 30 14:20 ca
dr-x------ 2 root root 0 Mar 18 14:14 co
lrwxrwxrwx 1 root root 17 Mar 18 14:14 d
drwxrwx--x 18 system system 4096 Mar 16 17:39
-rw-r--r-- 1 root root 118 Dec 31 1969
drwxr-xr-x 11 root root 2280 Mar 30 18:50
lrwxrwxrwx 1 root root 11 Mar 18 14:14
-rwxr-x--- 1 root root 94344 Dec 31 196
-rwxr-x--- 1 root root 2210 Dec 31 196
-rwxr-x--- 1 root root 9299 Dec 31 19
-rwxr-x--- 1 root root 15379 Dec 31 196
-rwxr-x--- 1 root root 9299 Dec 31 1969 init.
drwxrwxr-x 6 root system 0 Mar 18 14:14 mnt
drwxr-xr-x 9 root root 2048 Feb 14 12:55 pds
dr-xr-xr-x 152 root root 0 Dec 31 1969 proc
drwx------ 2 root root 0 Feb 7 15:23 root
drwxr-x--- 2 root root 0 Dec 31 1969 sbin
lrwxrwxrwx 1 root root 11 Mar 18 14:14 sdcard -> /mnt/sdcard
drwxr-xr-x 12 root root 0 Mar 18 14:14 sys
drwxr-xr-x 13 root root 4096 Mar 12 13:16 system
-rw-r--r-- 1 root root 221 Dec 31 1969 ueventd.goldfish.rc
-rw-r--r-- 1 root root 629 Dec 31 1969 ueventd.olympus.rc
-rw-r--r-- 1 root root 3707 Dec 31 1969 ueventd.rc
-rw-r--r-- 1 root root 629 Dec 31 1969 ueventd.stingray.rc
lrwxrwxrwx 1 root root 14 Mar 18 14:14 vendor -> /system/vendr
/ $
6:55

Vart ska jag/vi börja?

- Man kan göra indelningar/kategoriseringar (som i sin tur kan finfördelas)
  - Front-end
  - Back-end
- Allting börjar med idé och berättelse.
- Annars blir ingenting till.

- Låt oss angripa detta uppifrån, ur ett fågelperspektiv.

Arkitektur != Infrastruktur. Eller???

- Arkitektur – fysiska strukturen och uppbyggnaden i ett system, komponent, modul, paket etc.
  Kan beskriva flöden i system på en fysisk nivå.
- Infrastruktur – strukturen mellan olika system, mellan komponenter i ett system, moduler i ett system.
  Kan beskriva flöden mellan system, moduler och komponenter på en mer konceptuell nivå.

- Ibland är det dock lite luddigt vart gränsen går… ☺

Digital infrastruktur

- Infrastruktur [https://sv.wikipedia.org/wiki/Infrastruktur]

Infrastruktur är anläggningar som representerar stora investeringar och som används dagligen av samhället. Framförallt avses **system för transport** av varor, personer och **tjänster** samt för energi och **information**. …

Alla människors offentliga och enskilda kontaktnät, även den egna familjen, är andra exempel på abstrakta infrastrukturer (infra-,*under*-). Dessa strukturer hålls ihop av gemensam kunskap, kultur, historia, traditioner och det gemensamma ägandet av olika infrastrukturer.

Till infrastruktur brukar man först och främst räkna system som omfattar vägar och järnvägar, elnät och andra energisystem, telenät och Internet, samt vatten- och avloppsnät.

Det är i allmänhet det offentliga samhället – staten och kommunerna – som ansvarar för utbyggnad och underhåll av infrastruktur. Eftersom infrastruktur ofta är utformade som centraliserade eller decentraliserade nätverk med nätverkseffekter, kan dessa få stordriftsfördelar när de omfattar hela samhället; på så sätt brukar infrastrukturen beskrivas som ett naturligt monopol.

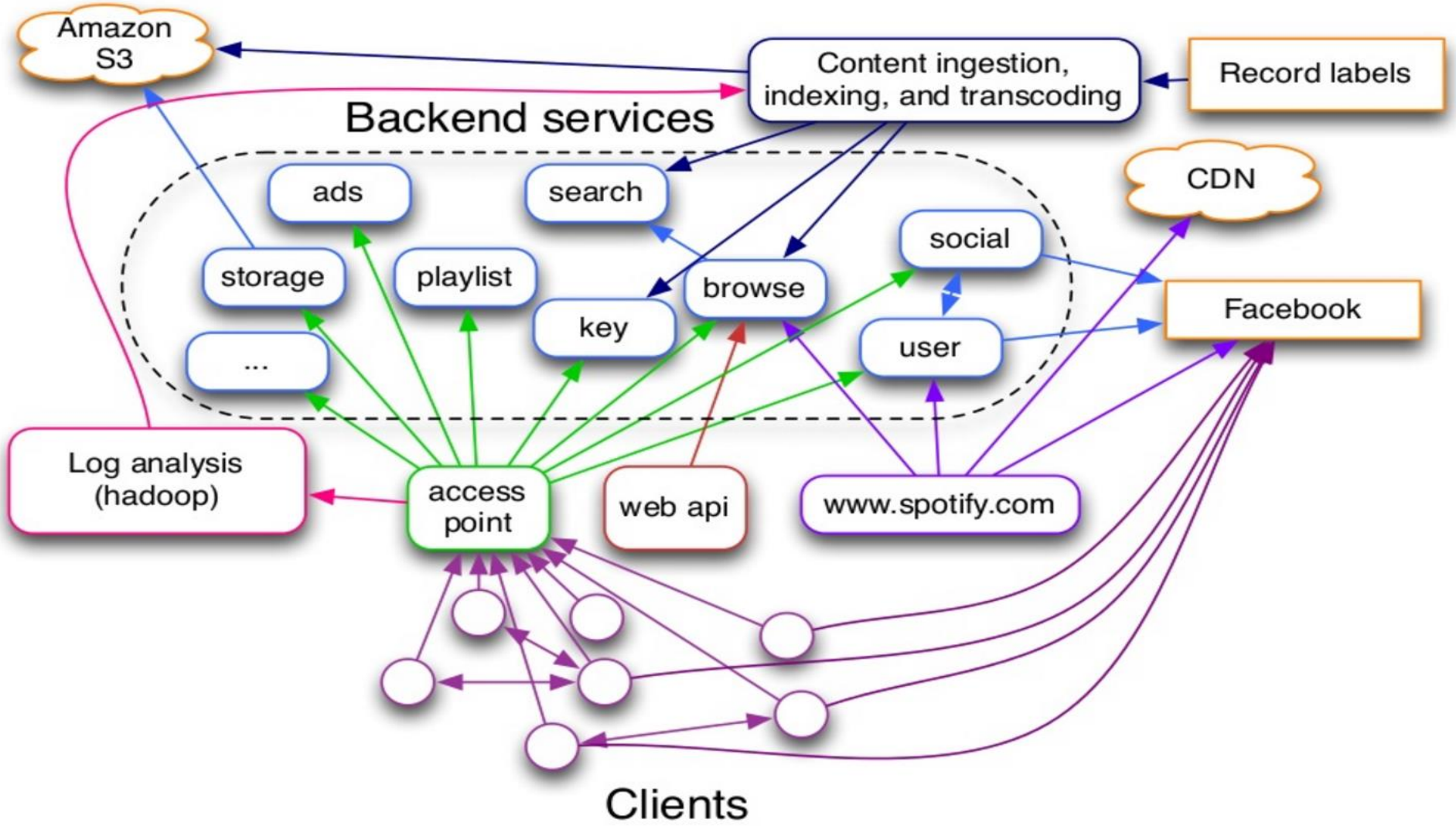
Begreppet *infrastruktur* kan även användas i en vidare betydelse, dock alltid som i *nödvändiga stödjande strukturer och mekanismer*. Till exempel kan ett *företags infrastruktur* innehålla bland annat e-post, intranät, reception och fysisk säkerhet.

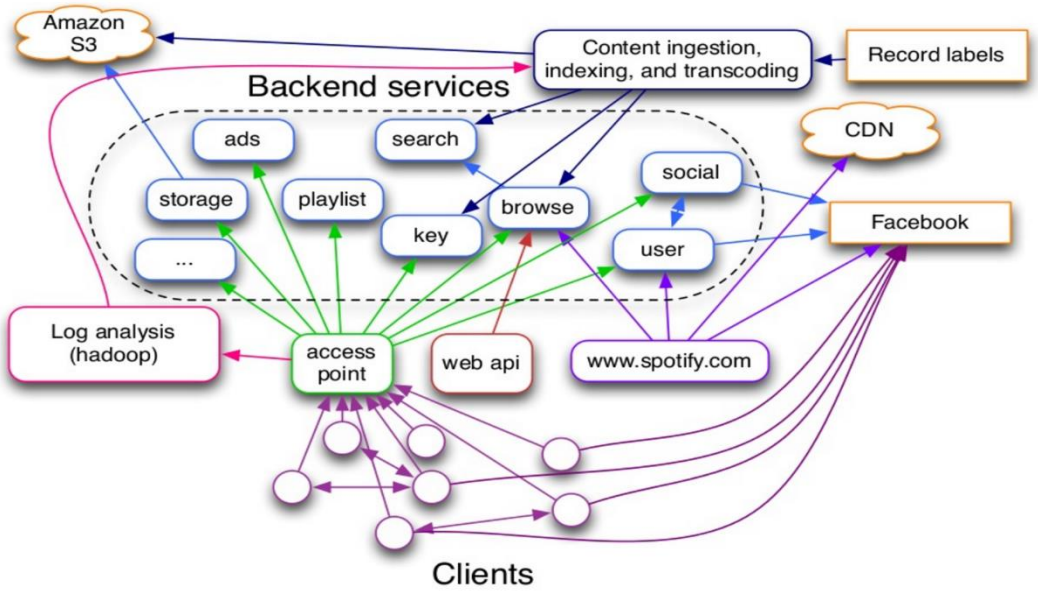
Ordet har funnits i svenska språket sedan 1968.

Digital infrastruktur contd.

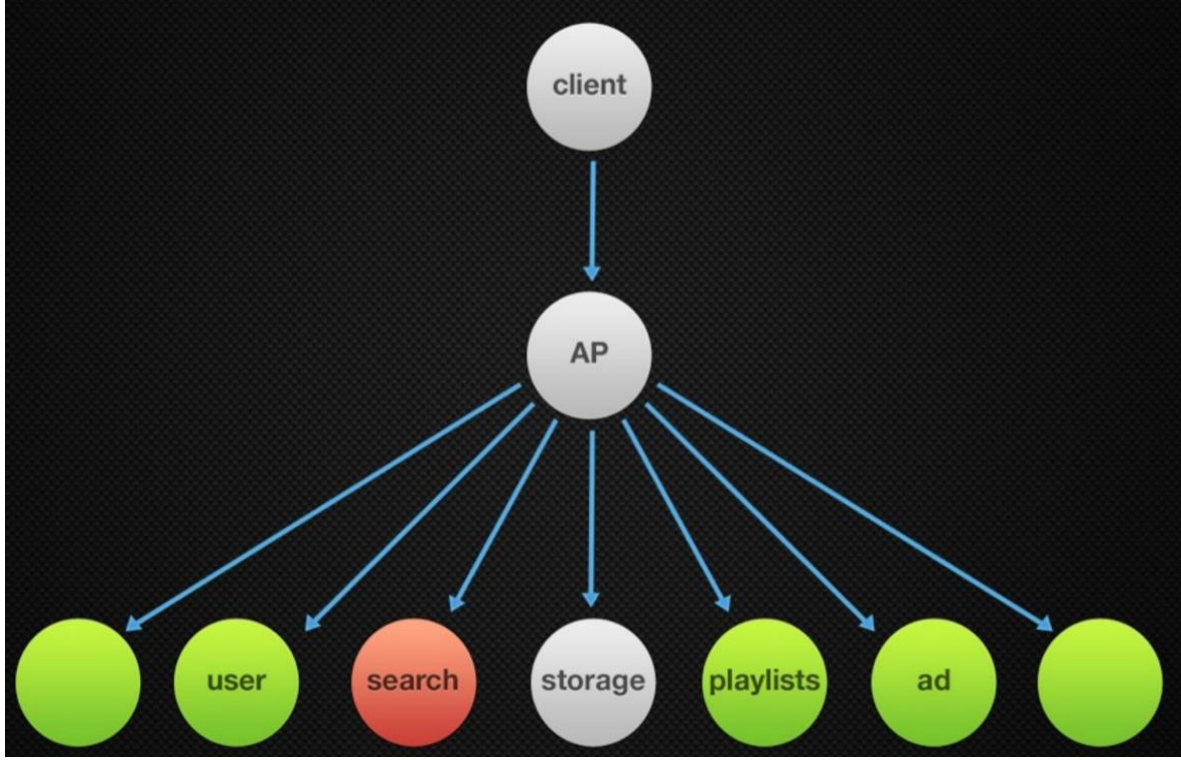
- För oss, som webbutvecklare, kan vi med en infrastruktur beskriva, organisera och bygga upp en applikation, tjänst eller ett system.

- Låt oss titta ett exempel……


Amazon S3
Content ingestion, indexing, and transcoding
Record labels
Backend services
ads
search
CDN
storage
playlist
browse
social
key
user
Facebook
...
Log analysis (hadoop)
access point
web api
www.spotify.com
Clients



- Clients
  - Appar (iPhone, Android …)
  - Applikationer (Win, OS X)
  - 3:rd party
  - Web
  - Alla ’clients’ har en access point, beroende på vad en user vill/gör. Det kan röra sig om en sökning, leta i katalogen, använda sina playlists, kontohantering etc…
- Alla access points går till en specifik backend service (nästa slide).
- Allt loggas
- Det finns ett öppet API (web api)
- Alla skivbolag förser Spotify med data (låtar, artist, etc etc).



- Varje service gör ***en*** sak (Unix philosophy).
- Det mesta skrivet i Python.
  Några i Java och C.
- Databas/datalagring beroende på vad som krävs – oftast PostgreSql, Cassandra, Memcached och/eller Tokyo Cabinet.
- ’Alla’ services pratar över HTTP (RESTful i många fall).
- AP – Access Point, lastbalansering (HAProxy)
- Allt skalas över hundratals servers.

Kort exempel - YouTube

- **Python -** most of the lines of code for YouTube are still in Python. Everytime you watch a YouTube video you are executing a bunch of Python code.
- **Apache** - when you think you need to get rid of it, you don’t. Apache is a real rockstar technology at YouTube because they keep it simple. Every request goes through Apache.
- **Linux** - the benefit of Linux is there’s always a way to get in and see how your system is behaving. No matter how bad your app is behaving, you can take a look at it with Linux tools like strace and tcpdump.
- **MySQL** - is used a lot. When you watch a video you are getting data from MySQL. Sometime it’s used a relational database or a blob store. It’s about tuning and making choices about how you organize your data.
- **Vitess** - a new project released by YouTube, written in Go, it’s a frontend to MySQL. It does a lot of optimization on the fly, it rewrites queries and acts as a proxy. Currently it serves every YouTube database request. It’s RPC based.
- **Zookeeper** - a distributed lock server. It’s used for configuration. Really interesting piece of technology. Hard to use correctly so read the manual
- **Wiseguy** - a CGI servlet container.
- **Spitfire** - a templating system. It has an abstract syntax tree that let’s them do transformations to make things go faster.
- **Serialization formats** - no matter which one you use, they are all expensive. Measure. Don’t use pickle. Not a good choice. Found protocol buffers slow. They wrote their own BSON implementation which is 10-15 time faster than the one you can download.

Till sist om infrastruktur

- En infrastruktur hjälper oss i det vi vill åstadkomma/göra; dokumentation, analys, planering…
- I det vi ska göra måste vi ta ställning till infrastrukturen – ibland råder yttre faktorer som avgör val av plattform, språk, protokoll etc.

- Mer läsning;
  - http://highscalability.com/
  - https://github.com/mattische/ME1564-HT15

Arkitektur och arkitekturprinciper

- Wikipedia, https://sv.wikipedia.org/wiki/Arkitektur, 2016-02-04

Arkitektur avser allt mänskligt byggande och formande av **den fysiska miljön**. Arkitektur är också läran om formgivning av i första hand byggnader. Det är emellertid ett multidisciplinärt ämnesområde som verkar mellan konst och vetenskap och även omfattar landskap, städer, interiörer, möbler och enskilda objekt.

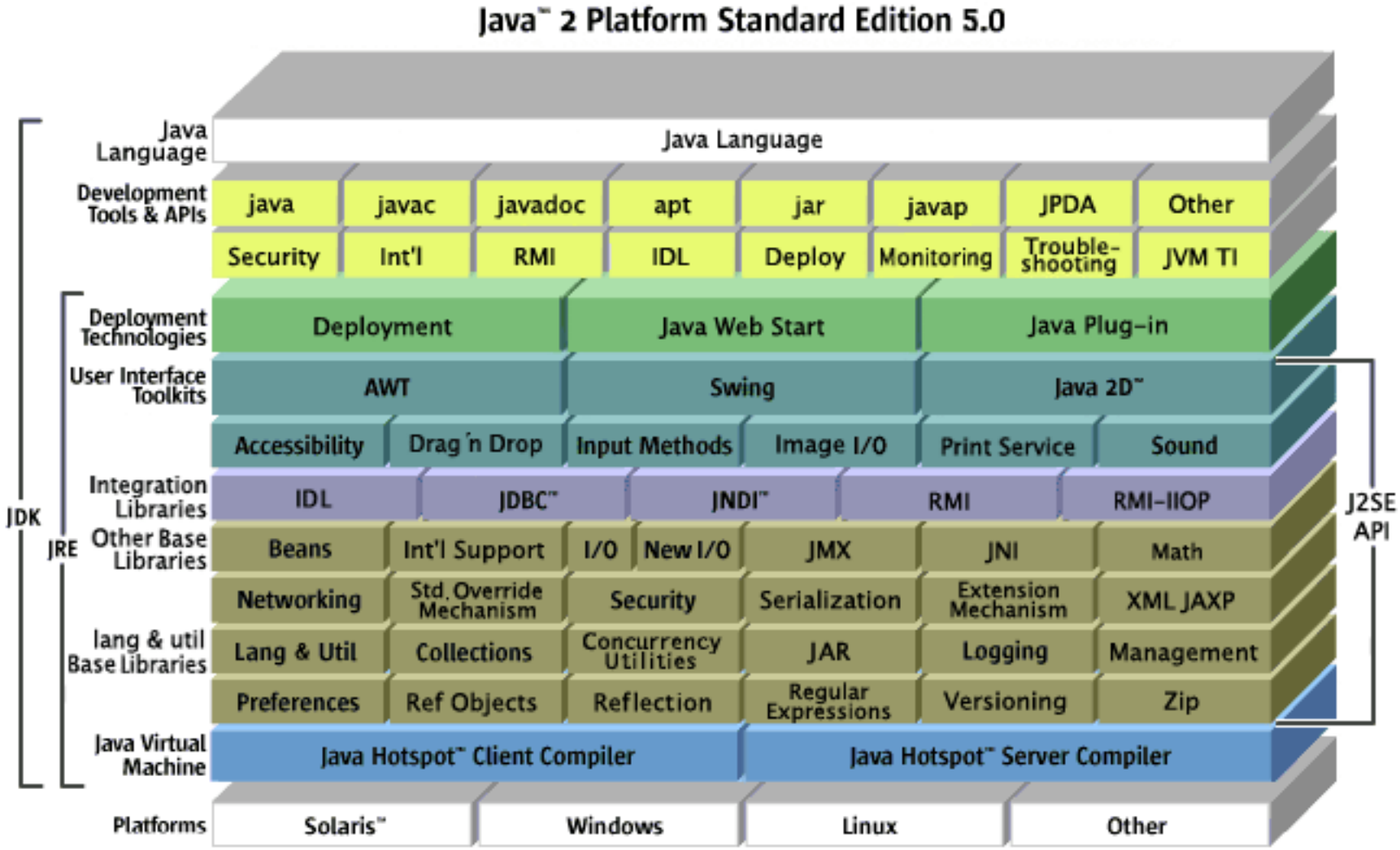
…

…

Ordet arkitektur används också inom till exempel datavetenskap och avser då den **övergripande strukturen hos ett system**.

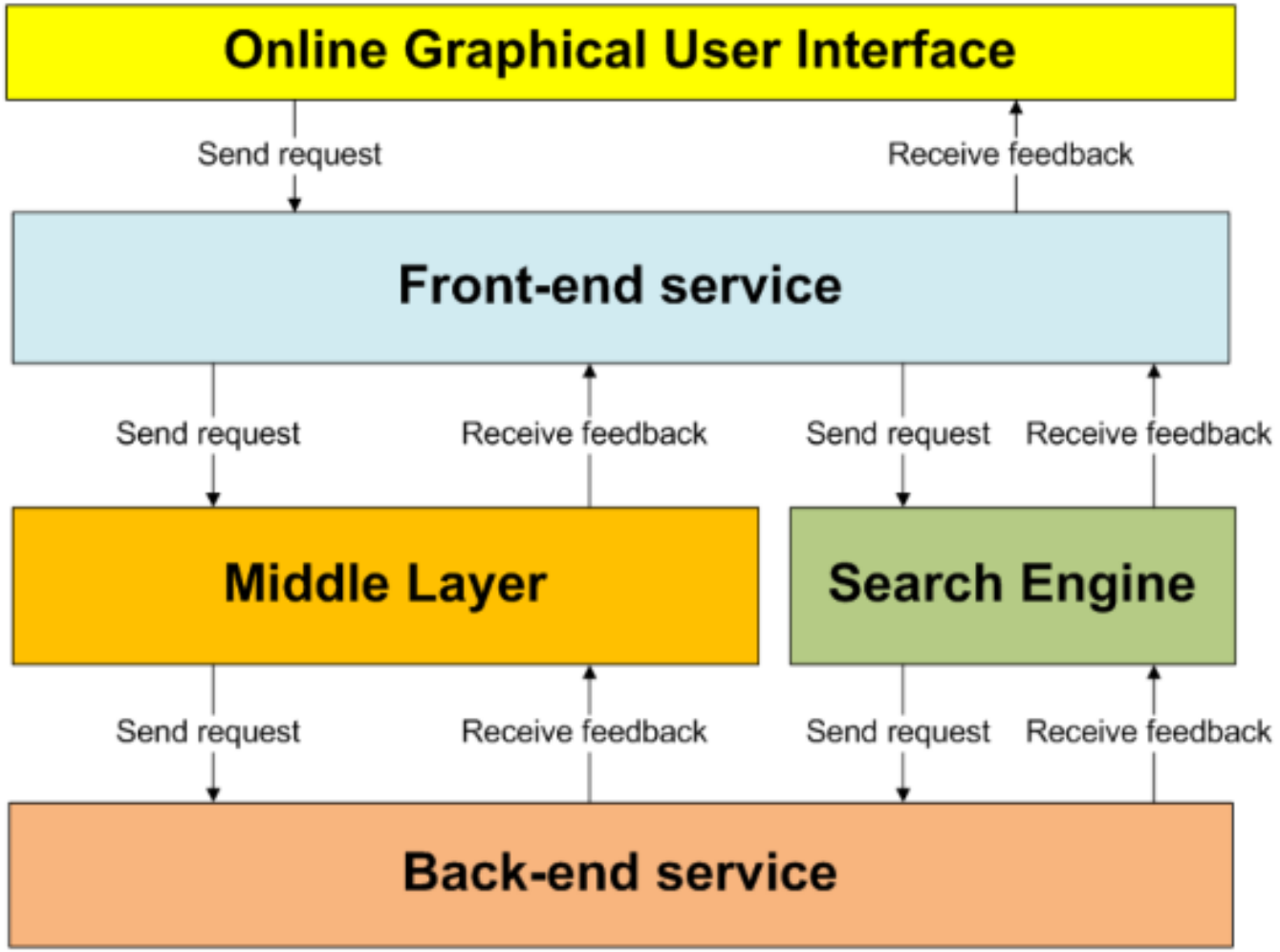
Arkitektur

- Javas 5.0 arkitektur

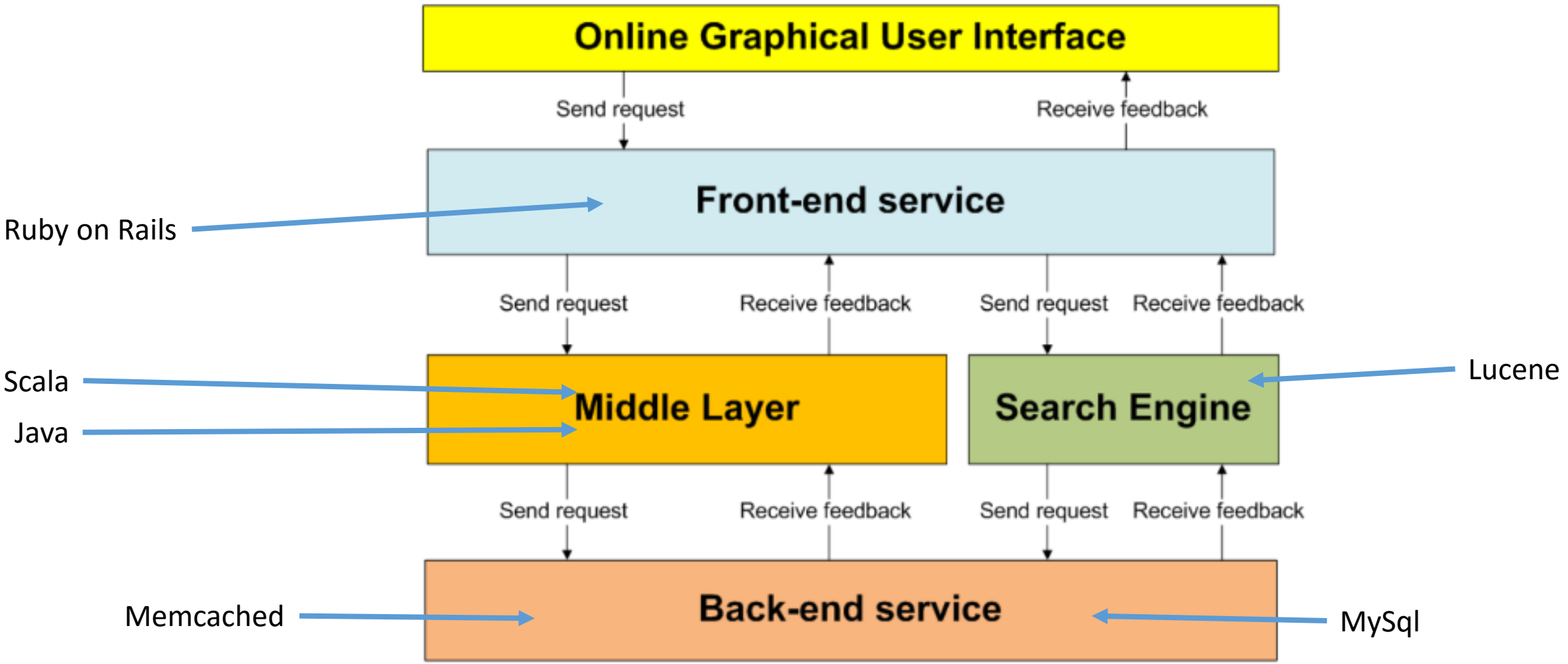


Infrastruktur eller arkitektur?

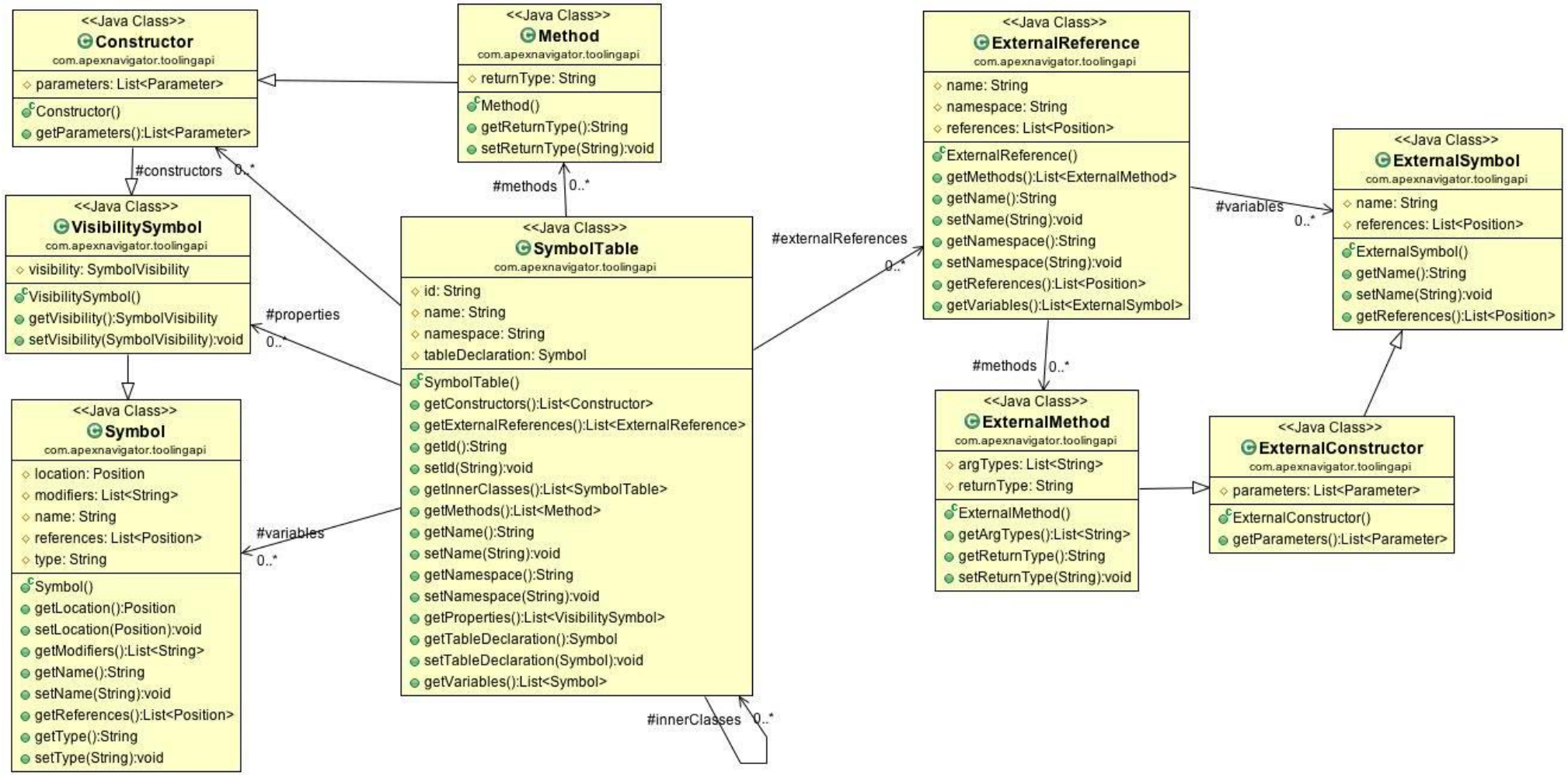
twitter



twitter


Online Graphical User Interface
Send request
Receive feedback
Front-end service
Ruby on Rails
Send request
Receive feedback
Send request
Receive feedback
Scala
Java
Middle Layer
Search Engine
Lucene
Send request
Receive feedback
Send request
Receive feedback
Memcached
Back-end service
MySql

Arkitektur med ett UML klass-diagram

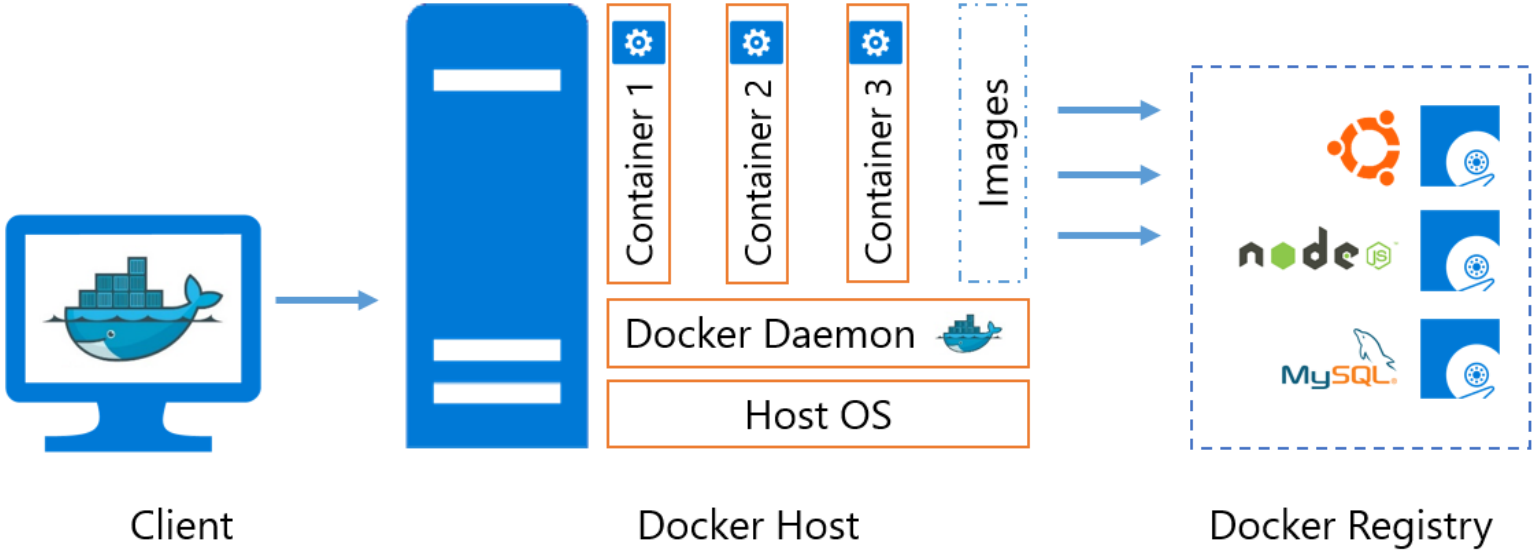


Ett litet test

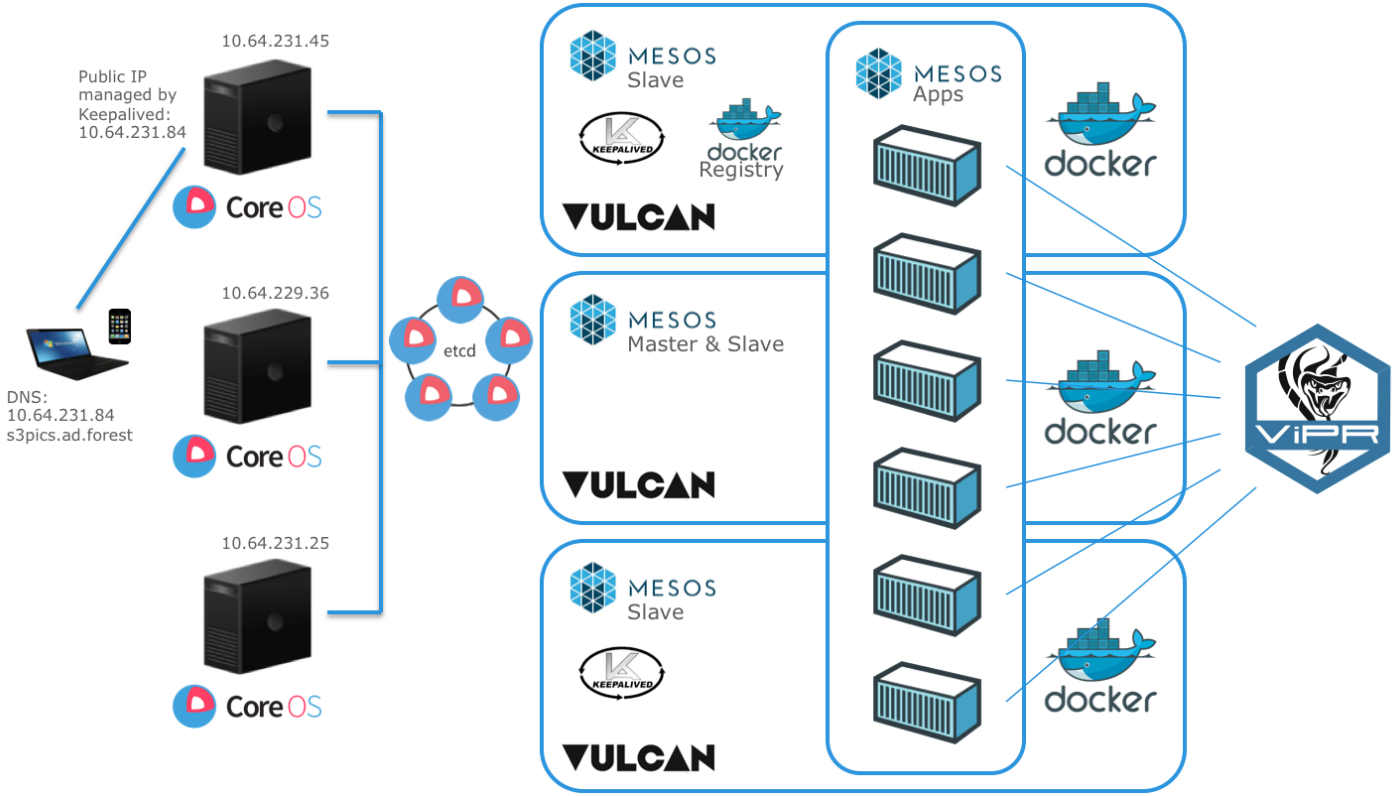
- En ’googling’ på ”docker architecture” och ”docker infrastructure” ger följande resultat.

- [https://www.docker.com/what-docker]

”docker architecture”



”docker infrastructure”



Arkitekturprinciper

- Principer, eller koncept om man så vill, för hur vi kan bygga en arkitektur – eller enligt vilket koncept arkitekturen ska fungera.
- I mångt och mycket vedertagna koncept, med riktlinjer för hur saker och ting ska byggas och fungera.
- I detta ingår s.k ***design patterns***, som koncept för hur man ska/kan bygga sin applikation med klasser som har specifik funktionalitet för att lösa vissa uppgifter.
- Design patterns är kategoriserade efter sammanhang och vilka problem de ska lösa.
- I webbsammanhang har ett design-pattern som heter MVC (Model View Controller) varit rådande länge.

Arkitekturprinciper contd. - ramverk

- Särskilt i webb-sammanhang, har design patterns fått stort genomslag;
  s.k ramverk (frameworks) dök upp, uppbyggda enligt ett antal design patterns (MVC i synnerhet).

- Ramverk finns för i princip alla språk; php, javascript, ruby, C#, python
- Exempelvis för php & python finns bl a följande;
  - Php: Laravel, Yii, FuelPhp, CakePhp, Symphony, Zend, Falcon, …
  - Python: Pyramid, web.py, Django, Pylons, Flask, TurboGears, …

Arkitekturprinciper contd. - koncept

- Vissa arkitekturprinciper är mer konceptliknande och avser att det man implementerar följer vissa koncept, funktionalitet och regler.

- Bland dessa bör för webbutvecklare särskilt nämnas;
  - API – varje app/tjänst/socialt nätverk har nuförtiden ett publikt API.
  - REST (RESTful)
    - **RE**presentational **S**tate **T**ransfer

  eftersom det (REST) idag slagit igenom med full kraft; API:er för diverse tjänster är uppbyggda enligt REST (se t ex Twitters, Facebooks etc API:er)

Ett ord om REST

- 6 constraints
  - Uniform interface
  - Stateless
  - Client-server
  - Cacheable
  - Layered system
  - Code on demand

Forts REST

- Använder http (methods); GET, PUT, DELETE, POST för en given resource.
- Använder sig av endpoints/URIs;

  http://www.mysite.com/users/masse

  GET request till ovan hämtar/visar given user
  DELETE till ovan tar bort
  POST/PUT till ovan URI skapar/uppdaterar

REST – några API

- https://dev.twitter.com/rest/public
- https://developer.spotify.com/web-api/endpoint-reference/
- http://wiki.openstreetmap.org/wiki/API_v0.6
- https://www.flickr.com/services/api/

- http://www.programmableweb.com/apis/directory

Till sist om - API

- https://sv.wikipedia.org/wiki/Application_Programming_Interface
- Application Programming Interface
- Varför göra API? Varför göra dom publika?
  - Vem som helst (registrerade) kan använda datat.
  - En flora, ett ekosystem, av 3:de-parts appar runt det egna varumärket – en slags branding/marknadsföring.

Miljöer/stackar

- Vi måste börja göra något.
- Vad ska vi göra?
- Idéer och berättelser…
- Bestäm vilken miljö vi ska ha.
  - Vart?
    - På vilken server? Vilket OS? Vilka övriga mjukvaror (t ex Git)?
  - Vad?
    - För det vi ska göra – vilket eller vilka språk? Vilken infrastruktur? Ska det funka på iPhone och Pebble?
  - Hur?
    - Ramverk? RESTful? Verktyg (editorer, databaser, …)? Scrum? Vattenfall?
  - Varför?
    - ?

Lokalt vs. inte lokalt (molnet)

- Vi måste sätta upp en miljö för det vi ska göra. Den kan sättas upp lokalt och/eller på en server.
- Oavsett om man är själv så måste det vi gör förr eller senare kunna användas av andra (göras åtkomligt). Dvs det måste göras åtkomligt på en eller flera servers.
- När vi är ’klara’ med alla tekniker, språk, etc vi ska använda så har vi vår ***stack*** – dvs allt det vi ska arbeta med.
- Ett par vanliga stackar är;
  - LAMP – Linux Apache MySql Php
  - WAMP – Windows Apache MySql Php
  - MEAN – MongoDb, Express, Angular, Node
  - …

Ett exempel

- Vi ska göra något med Php och en Mysql-databas.
- Lokal miljö via WAMP (http://www.wampserver.com/en/)
- På servern LAMP.
- Vi använder dessutom Git och Github för versionhantering.

Alternativ till lokala miljöer

- Skaffa en VPS på ex Digital Ocean, City network, Linode, Amazon etc.
  - ssh:a in, mounta
- Koding, CodeAnywhere, Nitrous, Runnable, etc etc
  - Som ovan. Alternativt arbeta (terminal, editor m.m.) direkt i browsern.

Din stack

- För det du (ni) ska göra – hitta och definiera din (er) stack.

Låt oss öva

- https://github.com/mattische/ME1579-VT16