# Web application layout

Wednesday, January 17, 2024 2:11 PM

important to know

- ways web apps run behind the scenes
- structure of web app
- components and how they can be set up in company infra

web layout categories

- web app infrastructure - structure of required components needed for function

web app needs to be setup on different server so need to know which database it needs to access

needs to access

- **web app components** – all components that web app interacts with
  - client
  - server
  - UI / UX
- **web app architecture** – all relationships between various web app components

## Web app infrastructure

**models** = infrastructure setups

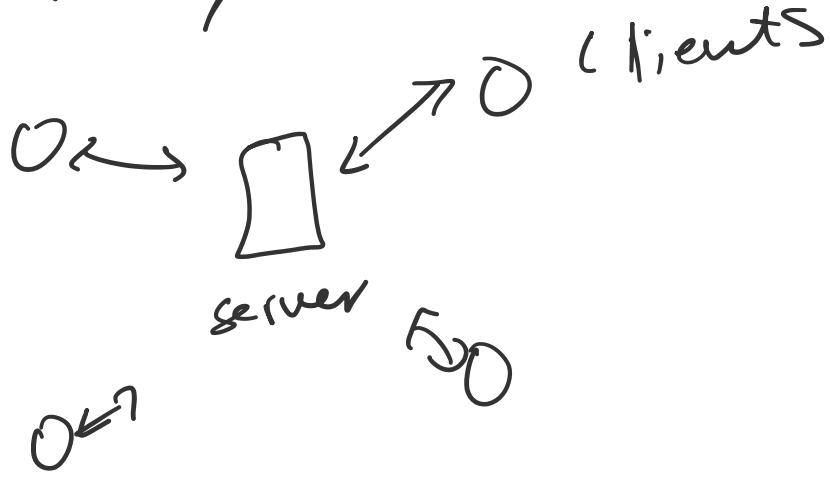
### 4 common ones

- Client-server
- One server
- Many servers - One DB

- Many servers - one DB
- Many servers - Many DB

## Client-server

server hosts web app and distributes to any clients




Frontend components - interpreted and executed on client side

Backend components - compiled, interpreted and executed on server

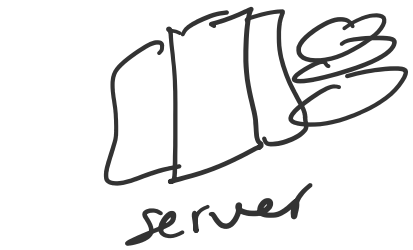
- Client visits site and uses UI
- Client sends HTTP requests to server
- Server sends results back to be rendered

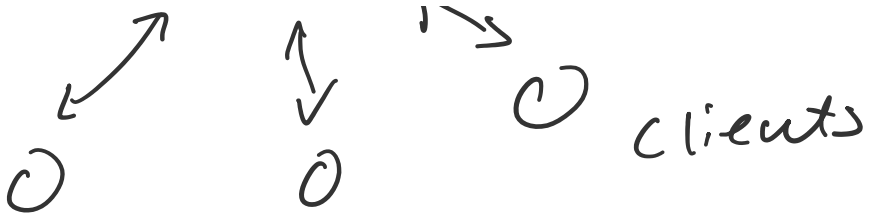
## One server

entire app, and even several apps, and all components including DB are hosted on

1 server

Simple but risky



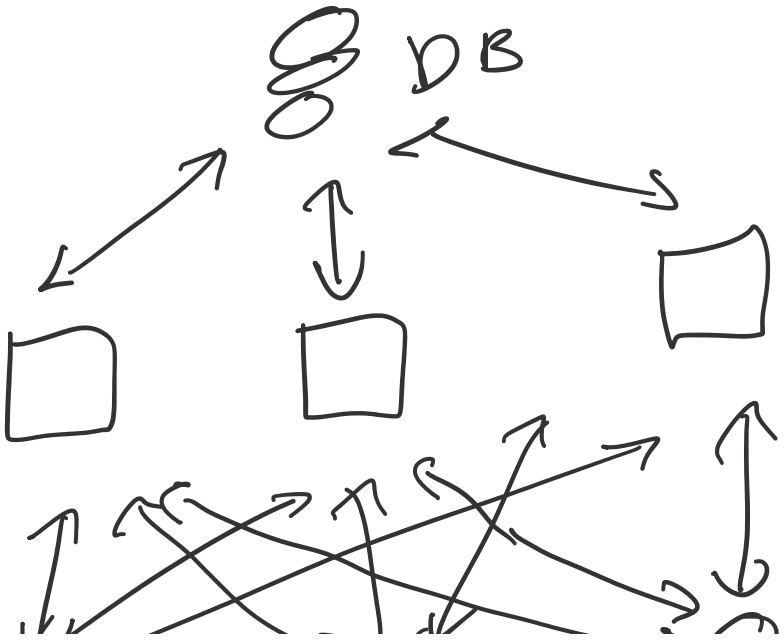

if one app compromised, then all are

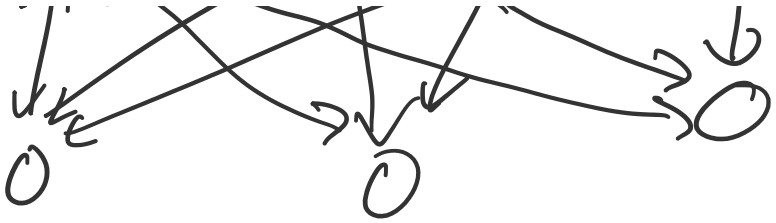
Same for outages of server

## Many server - one DB

Separate DB to own server

and allow web app hosting server to access





several apps can access same DB what syncing between them

Segmentation - each web app or component is separated so I doesn't compromise all

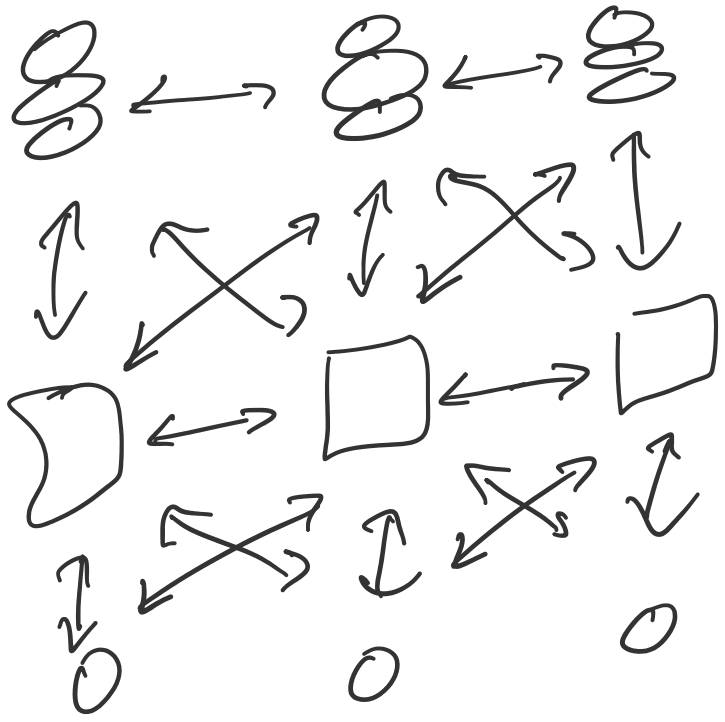
if DB is compromised, web app itself is wf, but still need controls to limit data available to each app

Many servers - Many databases

Many servers - Many databases

Within the DB server, each app's data is hosted in separate DB

app can only access its private data and common data



redundancy; if any app or db goes offline then backup will

run

difficult to implement, needs load balancers. but best for security because of access control and asset segmentation

there are also serverless and microservices models

## Web app components

1. client
2. server
   - webserver
   - web app logic
   - database
3. services (microservices)

3. Services (microservices)

- 3rd party integrations
- web app integration

4. Functions (serverless)

## Web app architecture

3-tier architecture

- presentation layer - UI components that enable comms w/ app and system

  HTML, CSS, JS

- Application layer - ensures all client requests are correctly processed

  ... like auth, privileges,

criteria like auth, privileges, and data is checked

- Data layer - work w/ app layer to determine where required data is stored and accessed

Some web servers can run OS calls and programs like IIS ISAPI or PHP-CGI

## Microservices

independent components of web app usually for 1 task

ex: Store
- Register
- search

- Payments
- Ratings
- Reviews

communicate w/ client and w/ each other

comms between services are stateless - request and response are independent

Stored data is stored separately from microservices

service-oriented architecture SOA

micro depend on each other

benefits

Reuse:

- written in diff langs and still work
- flex scaling
- easy deploy
- reuse code
- resilience

Serverless

build web apps w/out worrying about server itself

apps run in stateless containers like docker

CSP manages provisions, scale, maintenance

## Architecture security

Sometimes vulns come from design error

Ex: lack of access controls