Planning for the worst

shit happens
mongoDB

nginx

Flask app

mongoDB
I see dead backends

- Burning server
  - Replica set (master / backup failover)
- No more...
  - RAM (kill on consumption threshold, cgroups)
  - Disk (RAID, distributed FS)
- Server overload
  - monitoring
  - more servers (horizontal scaling)
Another possibility
Unreachable backends

- SysAdmin guy tripped over the cables
  - Hello Kitty forfeit.
- Switch failure
  - Network bonding / LACP
Fail proof stack & code

**nginx**
- Handle backend HTTP errors
- Serve from cache on upstream HTTP error

**Flask app**
- Stale cache
- Spooling / task deferral / message queuing
Clustering!
There’s still a SPOF here :)
Okay. So, what if your DATACENTER burns?

**Ops**
- Multiple datacenters / availability zones
- Remote backups (test them)

**IP routing / connectivity**
- Multiple datacenter BGP / Anycast
- DNS health checking (route53)

**Application design**
- Geo distributed apps
Real world problems
Hey ramnes, the client says he can’t authenticate on the website! *Something’s wrong!*

That sounds bad. Let me check logs…

Well, **the client is wrong**

Oh, okay… *goes away*

(Great)
Real world problem #1

Ramnes, something’s really wrong! The client still can’t connect!

Alright. Let me check code…

```
@app.route("/auth")
def auth():
    
    try:
        user.authenticate()
    except Exception as e:
        try:
            send_email(e)
        except:
            pass
    return 200, "OK"
```

That function raises an Exception if the mail server is down.
Real world problem #1

1. Know your code, refactorize when needed (even if someone else wrote it and that you don’t like his coding style)
2. “Errors should never pass silently” (Zen of Python)

**PS:** Don’t always blame ops guys. The DevOps thing is great, you should try it.
Real world problem #2

Weird graph showing an abnormally high maximum processing time
Real world problem #2

And then one day…
Local DNS resolution

root@server$ cat /etc/hosts
192.168.12.40 database-server-1
192.168.12.41 database-server-2
192.168.24.30 database-server-3
192.168.24.31 database-server-4

So it doesn’t overload your DNS server when your code tries to access your database with its domain name.