Scaling N26 Technology Through Hypergrowth
Building the bank world loves to use
A new approach to banking

Best digital user experience

A better price
+5M Customers

+$1.66Bn Monthly transaction volume

26 Markets

4 Locations

+$680M Funding

+1500 Employees
SPACES

Organise your finances.
Tax Return
€2,429,00

Time with Friends
€1,230,00
N26 tech

+180
Microservices

+60
Cross-functional teams

100%
In the cloud

+500
Deployments per week
but

How?
3 Years ago

+100K
Customers

6
Markets

+40M
Funding

+100
P&T Employees
Our challenges

- Small team
- Time critical
- Relatively young
Our challenges

Small team

Time critical

Relatively young
Experimentation & Knowledge sharing

- 2 Days GSDD
  1x every 6 weeks

- Lightning talks
  1x Week
HYPERGROWTH
Hypergrowth

What Is Hypergrowth?

40% CAGR*

20% CAGR*

Rapid growth

Normal growth

*Compound annual growth rate

Source: World Economic Forum
Our challenges

→ More User
→ More Markets
→ More Tech Hubs
→ More People
→ More Security
→ More Stability
Our Goal

Bring in more teams to enable us to get more users and release the in new markets
Company onboarding
Team onboarding
Buddy system
Target operating model

- Segments
  - Group
  - Group
    - Team
    - Team
Target operating model

3 Different currencies

3 Different banking regulations

+10 Payment schemes

4 Different locations
Our Goal

Have consistency in technology in order to build up knowledge, enable team mobility.
Microservices allows you to write each new service in a different language.
Languages

Java

Rust

Kotlin

Node.js

TypeScript

Scala
Decisions & Alignment

- Based on Thoughtworks
- Safe way to try things
- Favor Consistency
- Alignment across teams
Problem

How to avoid micro-monoliths
Infrastructure as Code (IaC)
Infrastructure as code

Servers

Network

Settings
Infra as Code

Benefits

→ **Strong separation**
   AWS organisations enables very fine-grained segregation of services

→ **Scalable**
   Small number of people-to-server ratio

→ **Agile**
   Supporting change or new services easy

→ **Reduced human error**
   Automation Prevents deviation

→ **Source controlled**
   4 eyes principle plus audit
Example

Remember meltdown and spectre?
Affected billions of systems globally
Dafydd Vaughan
@dafydbach

GiffGaff Warehouse hadn’t patched their software for over 6 years. I wish I could say that this isn’t common practice

10:25 AM - 11 Jan 2018 from London, England

8 Retweets 5 Likes
Vulnerabilities
publicly announced
Intel responds with initial statement
Intel announces a fix “for the majority”
Amazon announces updated Kernel
Base image updated and tested
Deployments with new image incrementally tested and rolled out
Result? Done!
the speed of a startup and the security of a bank
Continuous Delivery

CD
Continuous Delivery

- Security and automated testing
- Artefact Publish
- Automated checks
- Artefact build
- Dev Deploy
- Automated checks
- Staging Deploy
- Automated checks
- Live Canary Deploy
- Automated checks
- Live roll out
- End

- Code Review and commit
- Artefact build
- Dev Deploy
- Staging Deploy
- Live Canary Deploy
- Live roll out
Continuous Delivery - Live Blue Green

Diagram showing a load balancer (LB) connecting to multiple instances.
Continuous Delivery - Live Blue Green
Continuous Delivery - Live Blue Green
CD

Benefits

→ **Immutable infrastructure**
  No snowflakes in our infrastructure

→ **Rollbacks are easy**
  Historical AMIs enable rollback

→ **Scalable**
  Multiple instances enable scaling control

→ **Fine-grained control**
  Implement fine-grained security and compliance controls

→ **Customisable**
  Good machine selection for types of services
Problem

While having a high release rate, keep high availability and minimize risks.
Incidents

High Availability

>500 deployments per week
The best way to avoid incidents is by avoiding changes
Minimize time

Detection — Diagnostic — Fix — Delivery
What includes post mortem?

- A written record
- Preventing actions
- Actions to mitigate
- The incident impact
- The root cause
Lessons learned

Adjust to your journey conditions
Trade off analysis

Continuous Delivery
Is enable trouble healthy automation

Infra-as-code
AWS makes it easy to implement

Keep a availability in mind
Keep a close eye on service objectives
Scaling N26 Technology Through Hypergrowth

Thank you!
Any questions?

@folgerfonseca