ING is a global financial organization, active in 41 countries. This talk is about the retail bank of NL with...
9 million debit cards
8 million retail customers
7 million ATM transactions/month
Mobile banking is used by 4.5 million customers. Together, they log in 6 million times a day (100+ TPS).
AVAILABILITY FIGURES 2018
PRIME TIME (06:30 AM – 01:00 AM)

<table>
<thead>
<tr>
<th>Service</th>
<th>Uptime</th>
<th>Downtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Banking</td>
<td>99.77</td>
<td>0.22</td>
</tr>
<tr>
<td>Mobile Banking</td>
<td>99.87</td>
<td>0.13</td>
</tr>
</tbody>
</table>

Regulator target: 99.88
Logins per second for Mobile Banking
AVAILABILITY FIGURES 2018
24 HOURS A DAY

INTERNET BANKING
- Uptime: 99.63%
- Downtime: 0.37%

MOBILE BANKING
- Uptime: 99.78%
- Downtime: 0.22%

Customer expectation: 99.999%
SR-what?

Site Reliability Engineering is “what happens when you ask a software engineer to design an operations function” – Ben Traynor (Google)
People
At ING we are organized in tribes with (Biz)DevOps squads responsible for build and run.

Our SRE team is a ‘horizontal’ squad part of a productivity engineering tribe. We support 1700 engineers across 340 squads.
Our SRE team

7 engineers (4 dev, 3 ops)
2 more joining soon
1 product owner
1 chapter lead
mostly with engineering and on-call experience in ING product engineering
When we hire SREs, we look for someone who’s

- Passionate about reliability, problems, DevOps and open source
- OK with failure
- Insensitive to hierarchy
- Willing to teach and advise engineers about reliability
- Experienced in on-call duties and 1+ language(s) in our stack
- Still excited to work with us after meeting half our team and having heard realistic job expectations
Process
Why and how did we start with SRE?

We used to have a small team of ops engineers on call for online channels. These engineers were the ones up at night, but they could not structurally improve service reliability because of our DevOps model.

SRE pilot was started and supported:
- Team was transformed and given a new purpose
- Decided on SRE model, way of working and roadmap
- Experiences and proposal were presented to senior management

After knowledge transfer of old tasks, SRE was launched :)
For SRE, we generally see 3 organizational models

1. **Product Engineering + SRE**
   - Service ownership is shared between PE and SRE

2. **Product Engineering**
   - SREs are distributed and embedded in PE teams, service ownership is shared

3. **Product Engineering**
   - Service ownership is with PE, SRE consults and creates tools

Our model (highlighted)
What do we do as SREs?

Curious to learn more about...

- Learning from failure? Check out Jason’s and Ryan’s talk
- Chaos engineering and graceful degradation? Check out Lorne’s talk
- High impact outlier system failures? Check out Laura’s talk

Service Reliability Hierarchy, from O’Reilly’s *Site Reliability Engineering* (2016)
What do we do as SREs?

**We spend 80% of our time on engineering**
- We deliver the Reliability Toolkit: a white-box monitoring and alerting stack
- We work on a secure container platform with a service mesh in public cloud

**We spread SRE love and best practices**
- We reach out to engineers to consult and get feedback
- We educate on reliability topics

**What we don’t do**
- On-call for product engineering
- Work on SRE-topics already covered by other teams in our organization
We do outreach and we educate on SRE topics

We reach out to engineers
• Feedback loop for products
• We are reliability advocates

We educate engineers
• Engineering onboarding
• Prometheus workshops

We facilitate knowledge sharing
• Cross-domain SRE guild
• SRE demo sessions open to all
• Guidance via chat and intranet
• Prometheus user community
• Conference report out
When we demo, we sometimes block the hallway
We use these principles in our way of working

- We work with industry standards
- We work with open source products and practices
- We automate toil wherever and whenever we can
Technology
Why did we develop the Reliability Toolkit?

- Mean time to repair is too long – we waste time finding incident owners
- Lack of insight into application health for teams
- High level of technology diversity makes implementing monitoring difficult
How does the Reliability Toolkit work?

Applications → Prometheus → Alert Manager → Grafana → E-mail, SMS (Message Bird) and ChatOps (Mattermost) → Model Builder
How do we provision the Reliability Toolkit?

Together with a team we create a joint config

We maintain and update binaries

We deliver the Reliability Toolkit on 5 instances over 3 environments, we remain responsible

We deliver client libraries so metrics can be scraped from servers
Before, teams would own and use a full pipeline...

- **version control**
- **combine configurations**
- **build**
- **publish**
- **deploy**

= **reliability toolkit**
...now they only own and update config

version control

done by devops team

combine configurations

build
deploy

reliability toolkit
Increasing and improving usage of Reliability Toolkit

Include client libraries in engineering frameworks

Ensure a good feedback loop: in person or in tooling

Educate others during onboarding and workshops

Template team dashboards and make other dashboards accessible to all
And now Reliability Toolkit usage has been increasing
We made onboarding and using our Reliability Toolkit easy, but our 70 onboarded teams still need to ensure that Prometheus can scrape metrics.

How can we reach all 340 teams?
Let’s try a service mesh!
Curious? Check the Software Defined Infrastructure track
Why use service mesh to improve reliability?

• Service mesh helps us to get new/updated functionality to applications fast

• We can improve observability for all: metrics, logs, distributed tracing and resilience patterns based on incident learnings that work out of the box

• We can introduce/expand A/B testing, canary releasing and staged rollouts

• Engineers only need to worry about security at application level: immutable containers, zero trust network and security policies for free, taking away risk documentation work
What are we working on next?

• Scaling in our Reliability Toolkit stack for efficient use of resources, scaling up number of teams using our stack
• Expanding our role as reliability advocates
• Completing PoC with service mesh
Takeaways

• Hire SREs from your product engineering domain
• Never compromise on mindset in SREs
• Start with a pilot if you are not sure if SRE works for you
• Pick a SRE model that works well for your organization
• Try to get senior management support and understanding
• Invest in SRE outreach and education
• Focus on scalability and ease-of-use in your tooling
• Don’t be afraid of redesign if it makes users happier
Questions?

Icons used are all from flaticon.com