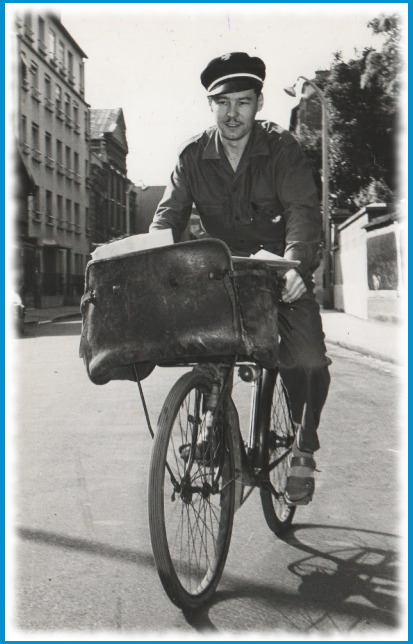


Improving your services, the DevOps way

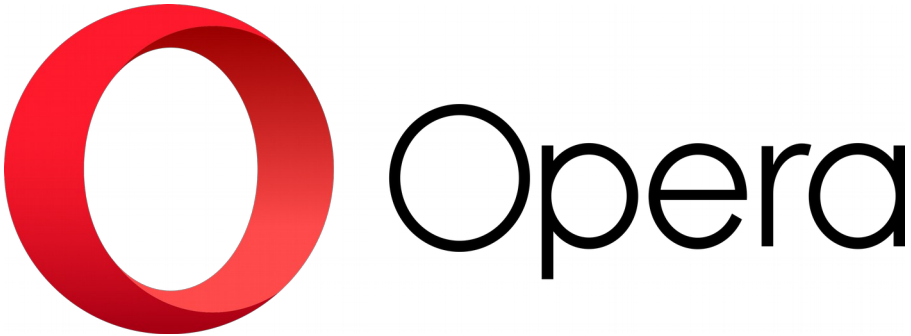
DevOps techniques for a non-DevOps shop

Marco Marongiu
Telenor Digital AS

telenor | digital

Who I am

- IT Manager & Head of IT
- Started Nov 2016

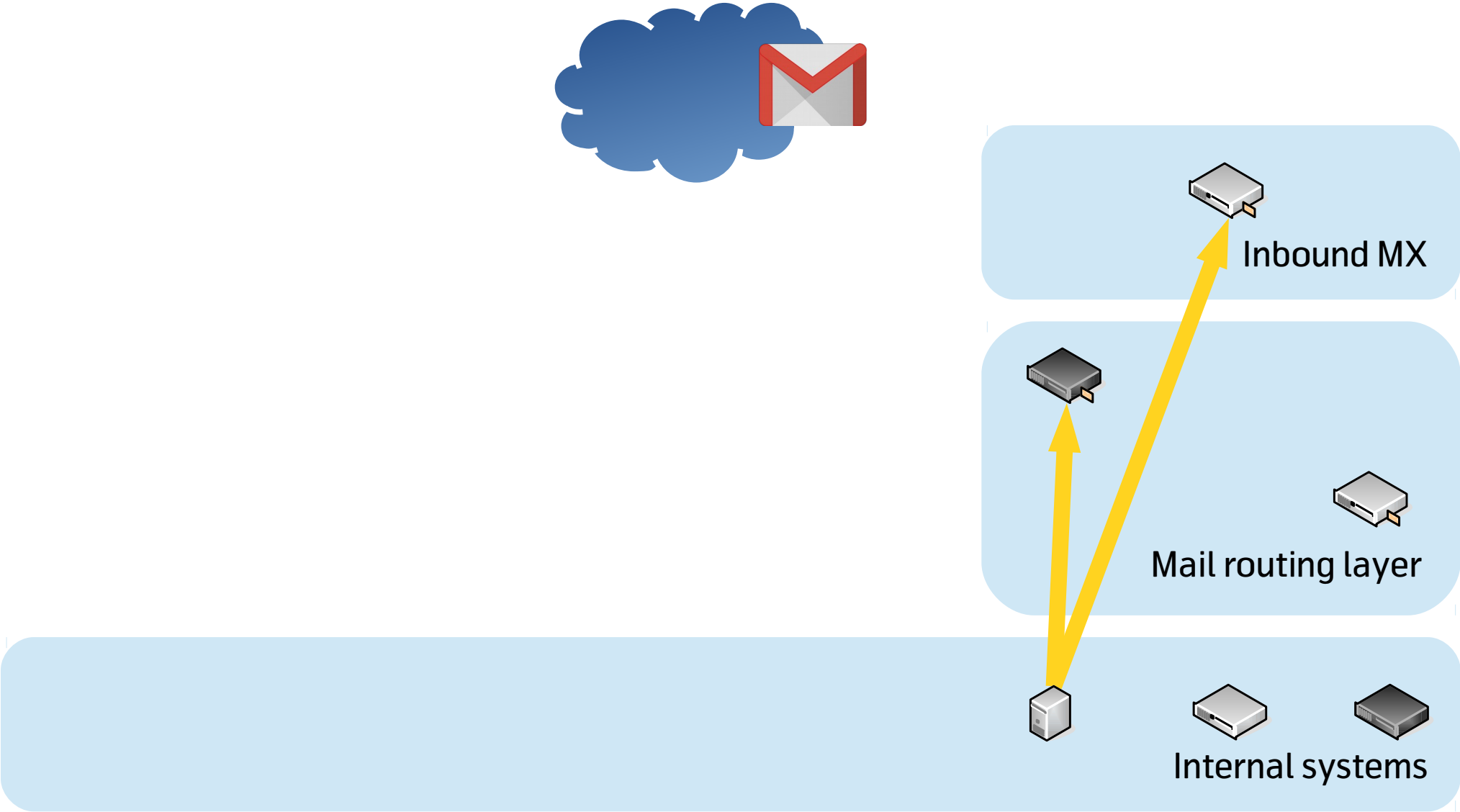


- Senior System Administrator
- Feb 2010 – Oct 2016
- Lots of Config Management!

Special thanks to **Michael Link**, CIO at Opera Software, for allowing me to hold this presentation.

Agenda

- The old email infrastructure
- The problem(s)
- The new email infrastructure…
- …and how we got there


Inbound MX
Mail routing layer
Internal systems

LEGACY

S.P.O.F.

SNOWFLAKES

INCONSISTENT CONFIGURATION

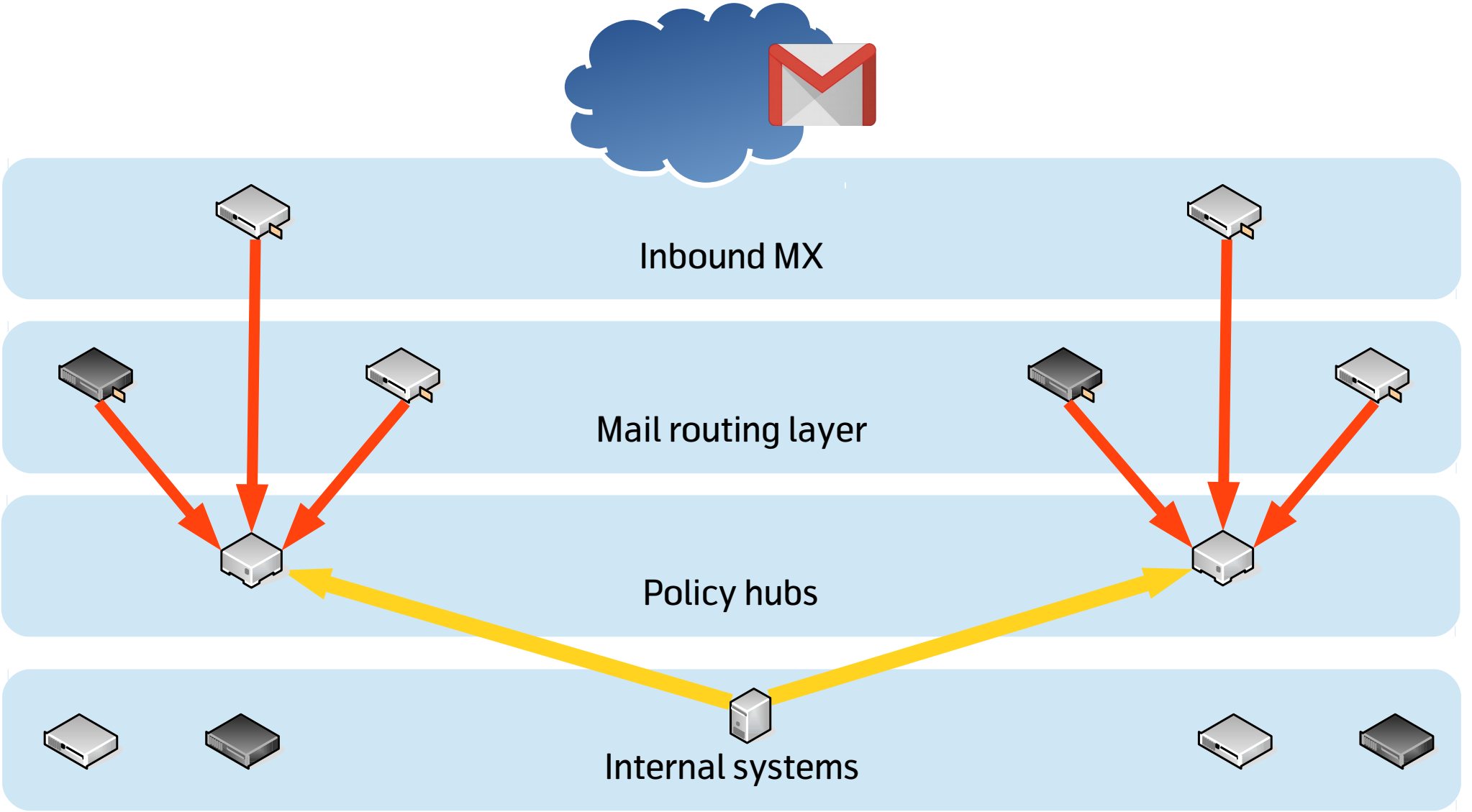
UNEXPECTED INTERACTIONS

UNEXPECTED BEHAVIOURS

Considerations

- **The architecture makes sense**, but the implementation **s***s**
- **We needed reliability**: remove SPOFs by having more than one machine per role, in different locations, and having more than one must be easy;
- **We needed resilience**: each machine should be easy to restore in the case of a failure;
- **We needed consistency**: the configuration should match the role of each machine and always be up to date

We needed CONFIGURATION MANAGEMENT


Inbound MX
Mail routing layer
Policy hubs
Internal systems

...but how does one reproduce SNOFLAKES?

YOU DON'T!!!

(silly!)

Reproduce what you want, not what you have

- We are not interested in **reproducing the existing machines** as much as we are interested in **reproducing their behaviour!**
  - **List** the behaviours for each role;
  - **Write tests** for those behaviours
  - Ensure that the **existing machines pass all tests**, or check your tests and fix them
- Build the configurations for the new machines and **use the tests to validate** them.

Use your (company's) full potential

- **Don't go to war alone:** if you have knowledgeable colleagues get help from them;
- If they can't help you to do the actual work, they can still help you in the design phase, or checking that the final result is sound;
- **Remember to do DevOps**
  - e.g.: include network and security specialists in the picture...

Start debt-free

- **Use an up-to-date version of the Operating System**: if the existing systems use an outdated OS, don't take the shortcut: start with a fresh version;
  - Yes, it still holds even if that means you have to switch from System V to Systemd
- **Reuse existing configurations** where it makes sense;
- **Manage the whole configuration from the very beginning**; don't end up with a snowflake again.

Start with the high level, code the details

- Whatever the role of the machine, the high-level operations will be the same:
  - Ensure that some **packages are installed**
  - Ensure that some **services are running**
  - Ensure that **configuration changes are detected** and picked up
  - Ensure that **configurations are reloaded** when they change
  - Ensure that **services are restarted** when there is a significant configuration change where a reload isn't enough

Start with the high level, code the details

- Most of the high-level operations are so generic that they can be coded in a reusable way, like subroutines in a programming language
  - Depending on your CM tool they will be called bundles, classes…
- Even better, they may be available in libraries or frameworks that are ready to use
  - In our case (CFEngine), we used the NCF framework from Normation, the fine makers of the Rudder Project
- Creating generic building blocks or using existing frameworks can save you **lots** of time!

Be lazy on similarities, smart on differences

- E.g.: what differentiates an inbound MX from an SMTP router?
  - In the MX we would install **more packages** (antivirus, spam filter…);
  - The SMTP router has a **milter**, a custom service that must start at boot;
  - …
- Where **configurations are similar across roles** but with some key differences (e.g. for the MTA), a **template** should definitely be used;
- **Configurations for "unique" services** could be distributed as **plain files**, unless there were information dependencies on the local machine, in which case a template is necessary;

Small plan, big wins

- The most complex piece to put together was the inbound MX, which required a fair amount of work to get (almost) right;
- The outbound MX and the SMTP router were similar enough that they could use the same “driver”
  - all in all, they were simple SMTP servers with some configuration differences
  - the same CFEngine bundle could configure both of them
- The investment in time due to the adoption of a new framework and writing tests ahead paid us back quickly!

Testing with production traffic

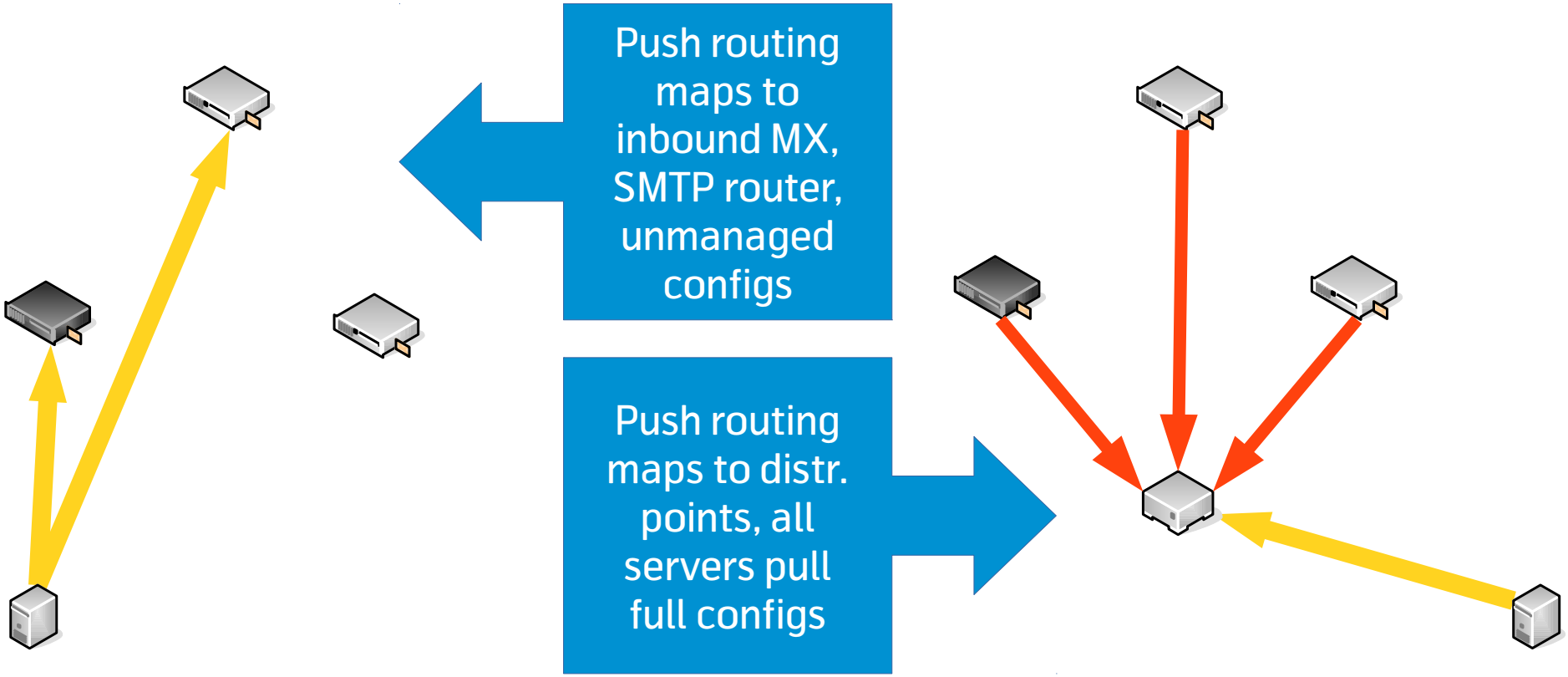
Testing with production traffic

- Iptables has a feature called "**weighted connections**" where it will act on a user defined percentage of connections, randomly chosen;
- We used that in the PREROUTING chain on the inbound MX to DNAT a few incoming connections on port 25, forwarding them to the new MX
- On the new MX connections through port 25 were marked by iptables and return packets routed back via the old MX using a dedicated routing table.

The last mile: distributing configurations


EVA
BME
SURGEON
OPS PLANNER
TORO
PAO
THOR
SPARTAN
FLIGHT DIRECTOR
JAXA

Small plan, big wins – again.



What you have is what you tested…

...but fixes are a snap away now!

- Unless you did some huge blunder, you have now means to easily deploy configuration updates across the whole infrastructure and have fixes applied in minutes and consistently
  - ...which is what happened to our mail infrastructure during the first couple of months

Emergency reconfigurations are not a problem

Conclusions

- We started with an architecture with good foundations but filled of SPOF's
- We grew it into a resilient, distributed, scalable architecture
- We did it by using techniques from test-driven development, agile, DevOps, collaboration in general
- We didn't do a perfect job, but we got the tools in place to improve it along the way
- **And you can do that, too!**

Thank you!

Marco Marongiu
Email: mmarongiu@tiscali.it
Twitter: @brontolinux
Web: http://syslog.me/

telenor | digital