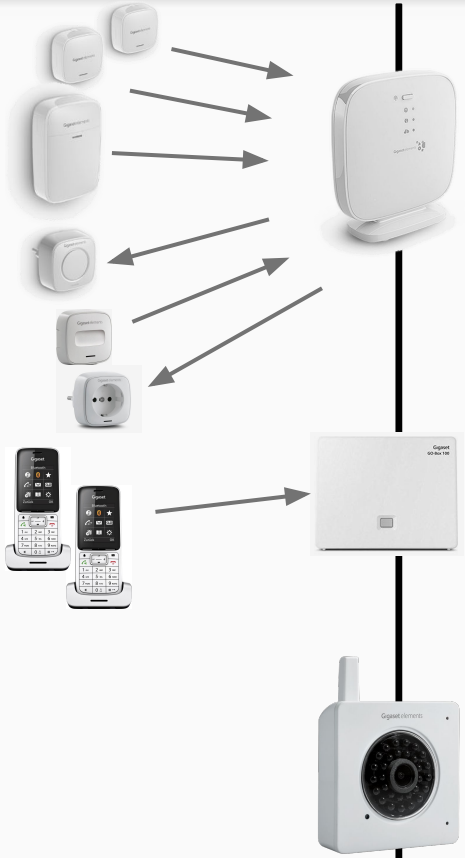
Evolution of the IoT backend platform

Tomasz Tarczyński, Gigaset

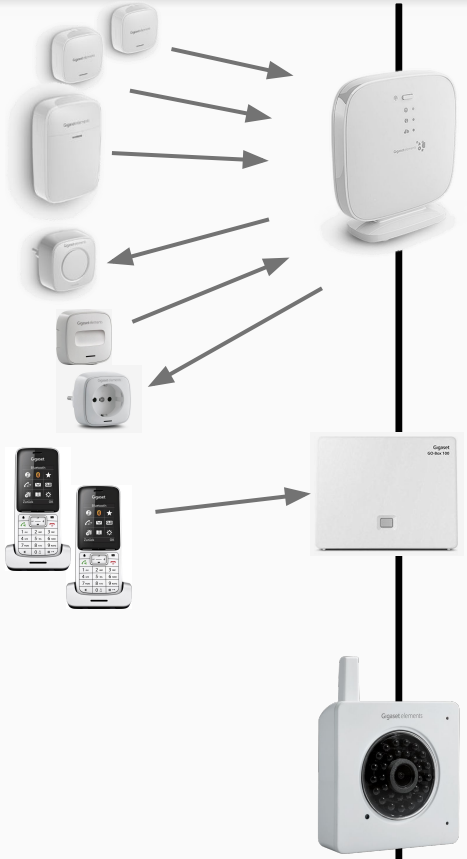
Smart Home

Gigaset elements

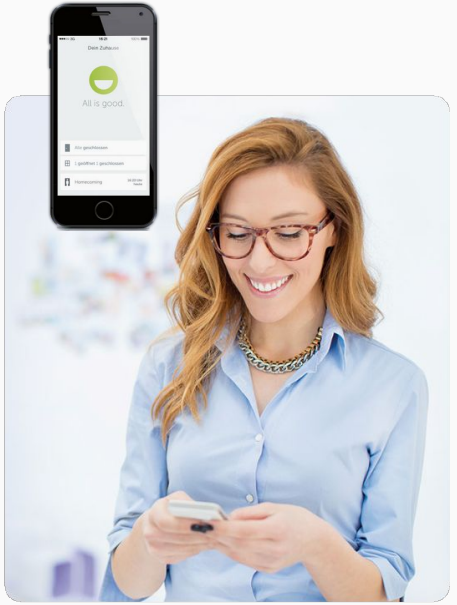


Gigaset elements

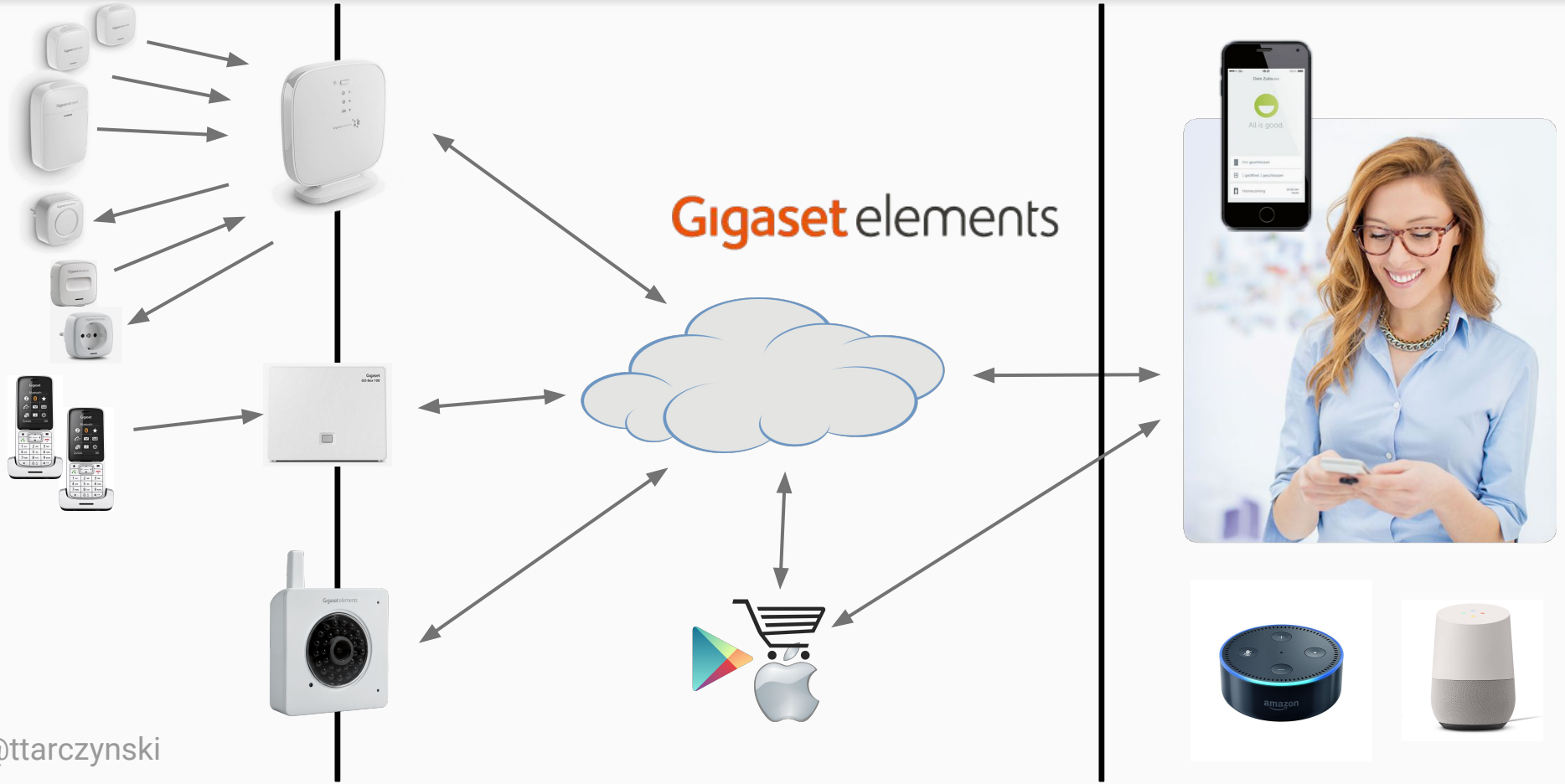
Gigaset elements



Gigaset elements

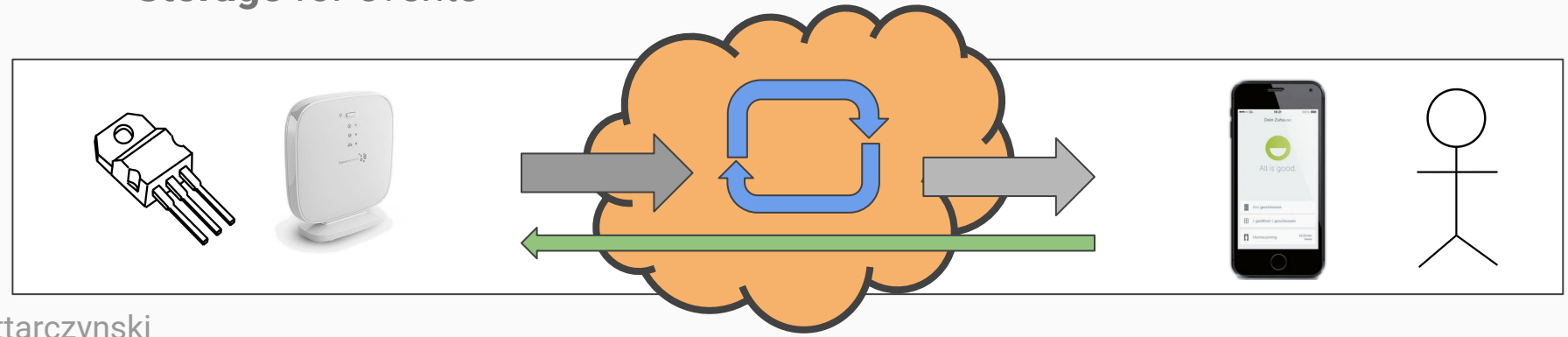


Gigaset elements



Requirements of **IoT Backend**

- **API** for devices
- **API** for frontend applications
- *Fast* internal **event processing**
- **Storage** for events



Requirements of **IoT Backend**

1. Reliability

Requirements of **IoT Backend**

2. Velocity

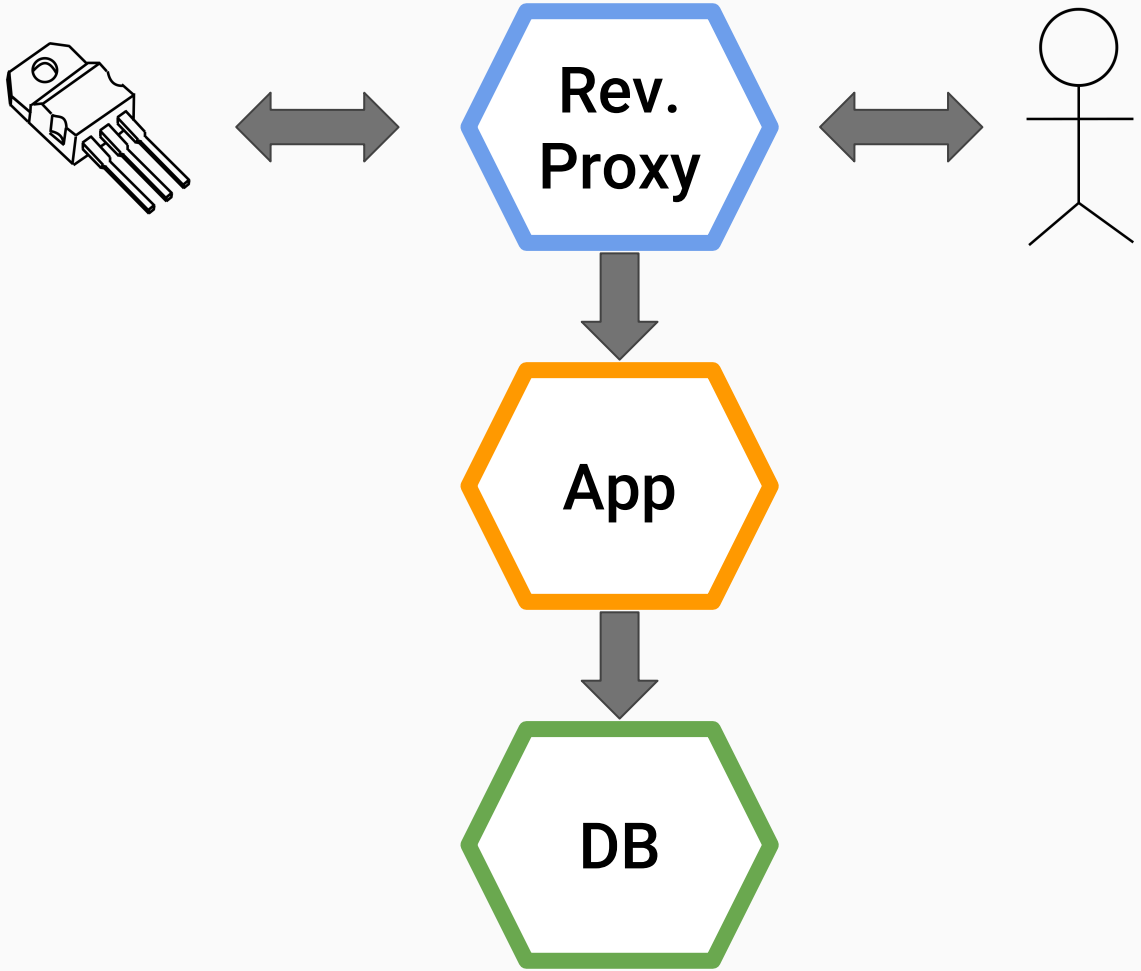
First Generation IoT Backend

A monolith app

- All backend functionality in 1 app
- Deployed in the cloud
- **Infrastructure** as a **Service**

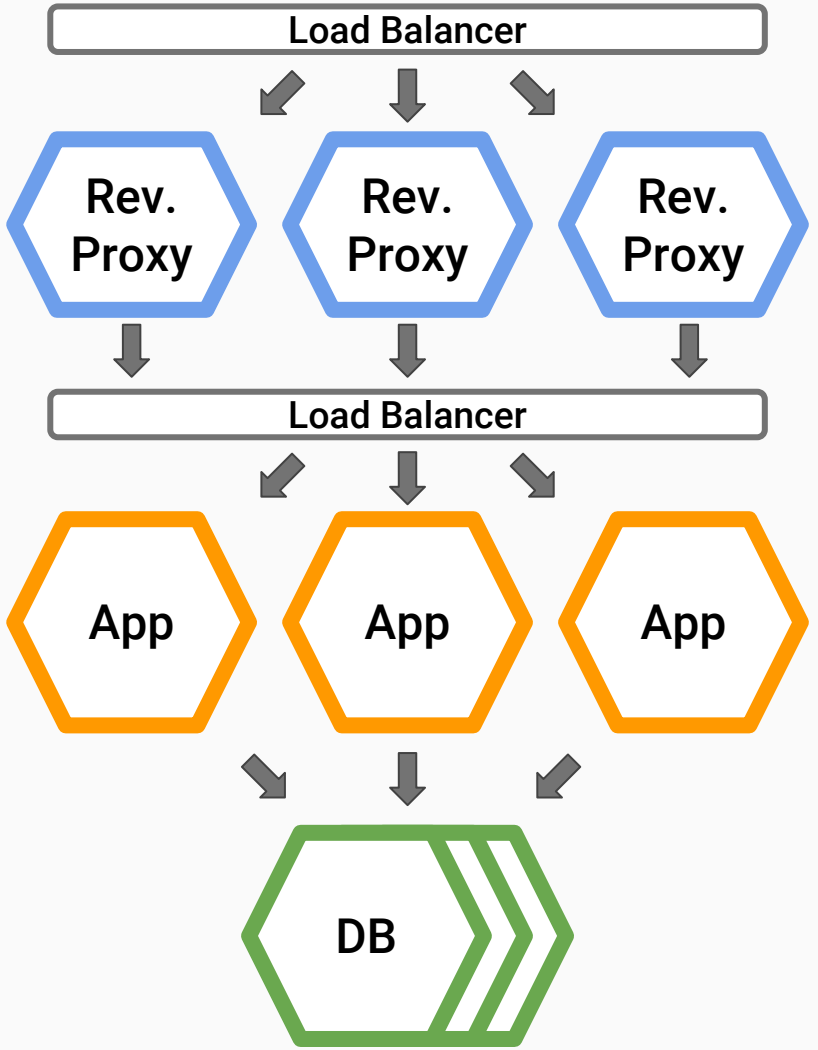
A monolith app

- All backend functionality in 1 app
- Deployed in the cloud
- **Infrastructure** as a **Service**



A monolith app

- **High Availability**
- Resistant to individual component failure



Teams building it

Dev

Deliver new features fast

Teams building it

Dev

Deliver new features fast

Ops

Reliability

Limitations of the Monolith

Big releases

(Long tests)

Limitations of the Monolith

Painful deployments

(Ops manual work)

Limitations of the Monolith

Failures

Limitations of the Monolith

Infrequent releases

(monthly)

Limitations of the Monolith

Developers work tightly coupled

With: Front-End Dev, Ops, TestLab

Limitations of the Monolith

Developers work tightly coupled

Within the team

Second Generation IoT Backend

Microservices – Why

Decoupling of features development

Microservices – Why

Frequency of deployments

Microservices – Why

Lower risk of deployment

Microservices – Why

Increase feature delivery peace

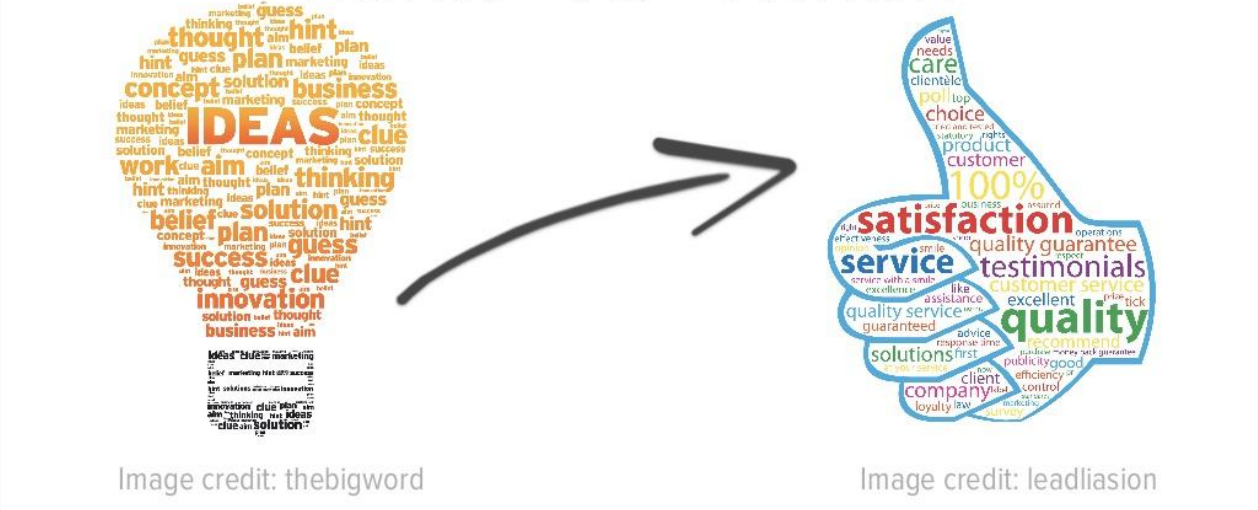


Image credit: thebigword

Image credit: leadliasion

Microservices – How

- The monolith split into
  ~70 small single-purpose apps

Microservices – How

- The monolith split into
  ~70 small single-purpose apps
- Individual app developed by 1–2 devs

Automation

- Lower cost of single deployment

Automation

- Lower cost of single deployment
- Consistency

Automation

- Lower cost of single deployment
- Consistency
- Self-service

Automation

- Puppet code in version control

Automation

- Puppet code in version control
- Control all config of servers and apps

Automation

- Fully automated deployments of apps
  - With no human intervention required

Automation

- Fully automated deployments of apps
  - With no human intervention required
- Short time to deploy
  - 30 minutes from request to done

Monitoring

Rapid feedback loop

Monitoring

- Logs and metrics are key

Monitoring

- Logs and metrics are key
- Fully automated

Monitoring

- Logs and metrics are key
- Fully automated
- Focus on user experience

Monitoring

When to alert?

User Experience

Sharing – Why

Information flow

Sharing – How

- Tools
- Dashboards
- Knowledge and learnings

Culture

Collaboration

Culture

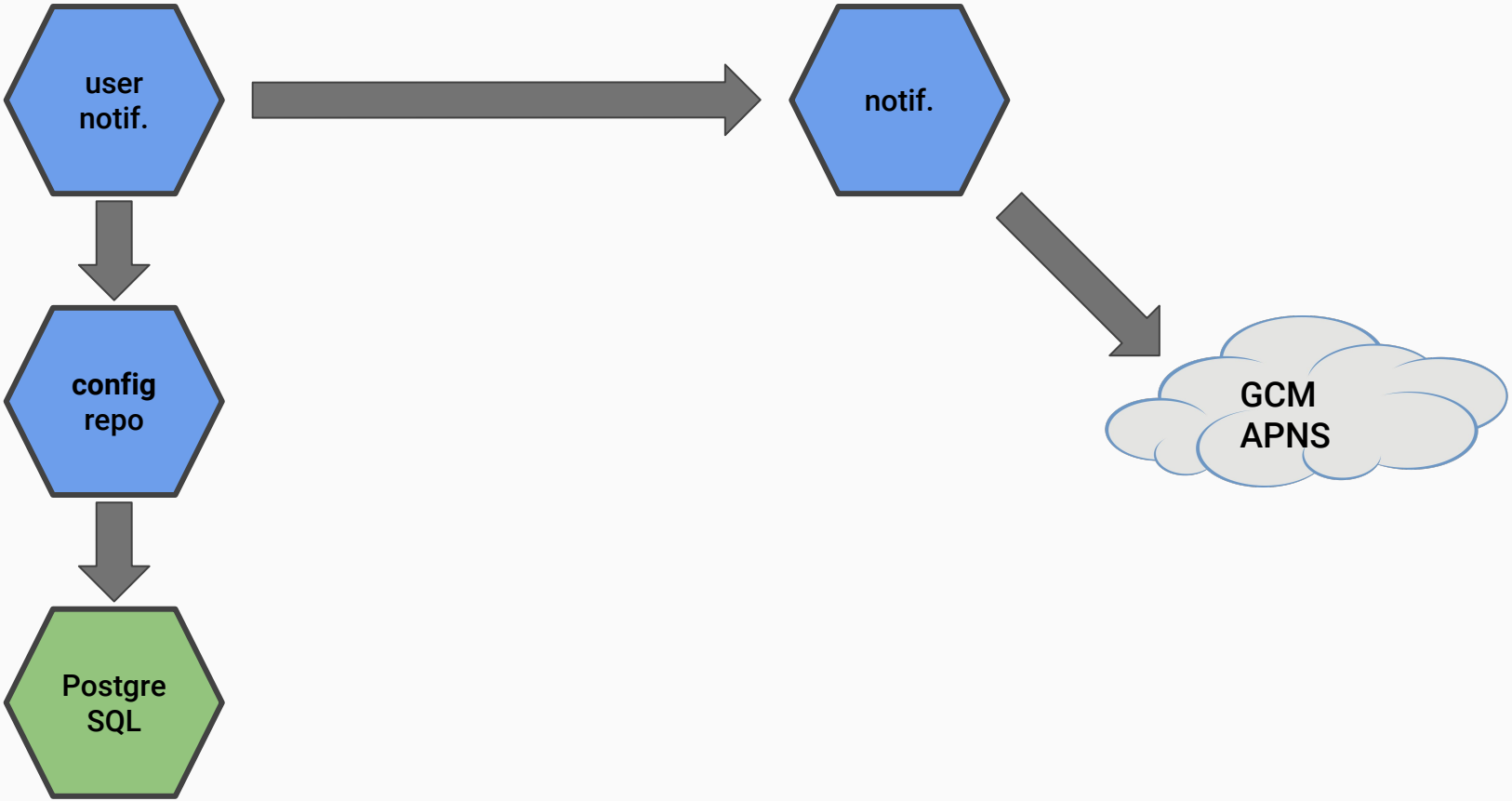
Work together every day

Culture

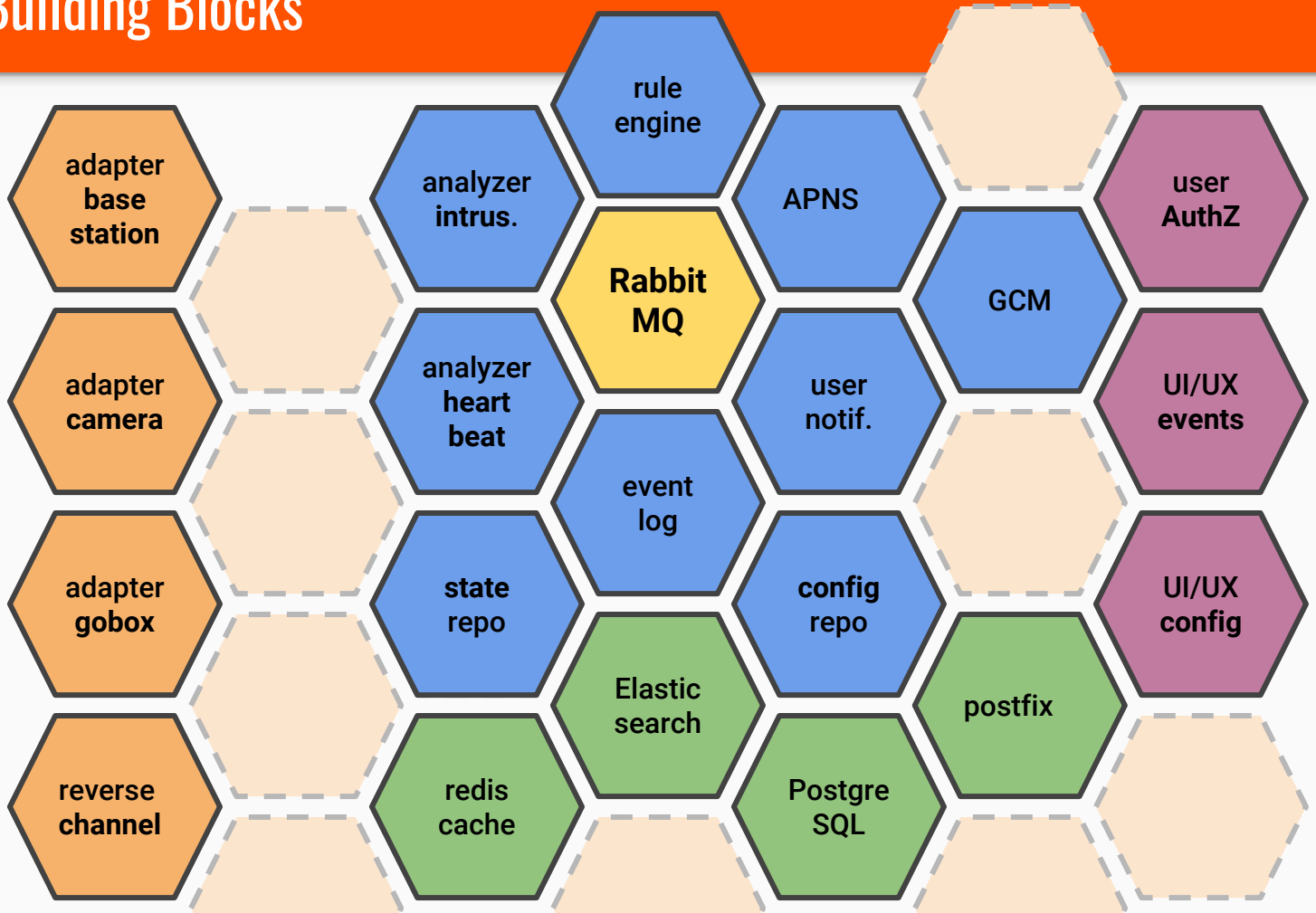
Fail Fast – Learn Fast

Architecture

Microservices – example



Building Blocks



play akka

elastic

RabbitMQ

redis

PostgreSQL

NGINX

HAPROXY

docker

Migration

One bite at a time

Migration

Built new platform
next to existing one

Benefits of Second Generation IoT Backend

1. Faster iteration time
    - 1 day (instead of 1 month)

Benefits of Second Generation IoT Backend

1. Faster iteration time
    - 1 day (instead of 1 month)
2. Frequent releases
    - Up to 10 production releases a day

Benefits of Second Generation IoT Backend

3. Fully automated deployments

Benefits of Second Generation IoT Backend

3. Fully automated deployments
4. Failures: improved MTTR
    - (Mean Time To Recover)
    - 30 minutes (instead of 4 hours)

Lessons Learned

Microservices require DevOps

Lessons Learned

Invest in observability

Lessons Learned

Look for balance:
Velocity vs Reliability

Lessons Learned

How fast can you learn?

Lessons Learned

Ability to adapt

Lessons Learned: Issues

Velocity is king

Lessons Learned: Issues

Complexity is hard

Lessons Learned: Issues

Host-centric model not optimal

Lessons Learned: Issues

Can one tool fix everything?

Future:
Third Generation IoT Backend

Cloud Native Platform – Why

Speed: Time to Value

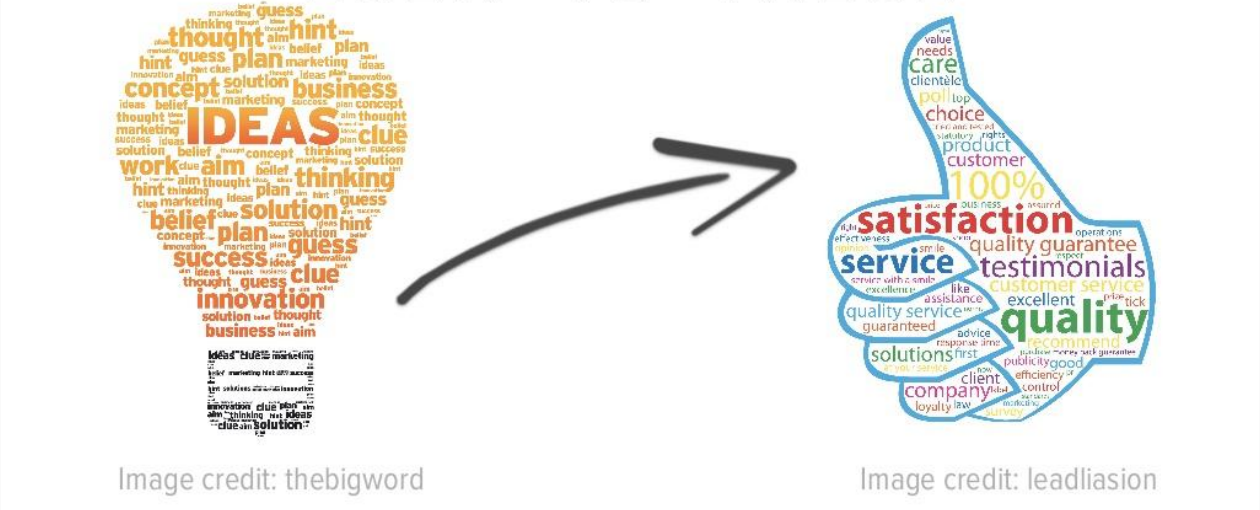


Image credit: thebigword

Image credit: leadliasion

Cloud Native Platform – Why

Scale + Reliability

Cloud Native Platform – Why

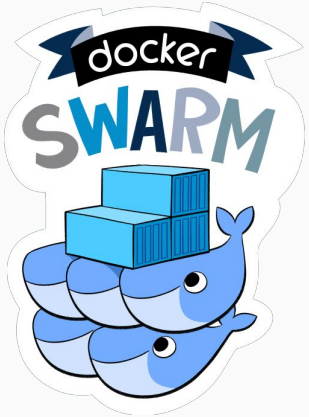
Efficient usage of resources

Cloud Native Platform – Why

VMs are the wrong abstraction

Cloud Native Platform – How

Tools?



kubernetes

HashiCorp
Nomad

Cloud Native Platform – How

Experiments

Cloud Native Platform – How

Orchestrator changes everything

Cloud Native Platform – How

Loads of learning

Cloud Native Platform – How

CNCF Trail Map

CLOUD NATIVE COMPUTING FOUNDATION

CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape *l.cncf.io* has a large number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator or a Certified Kubernetes Application Developer

cncf.io/training

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider

cncf.io/kcsp

C. Join CNCF's End User

1. CONTAINERIZATION

- Commonly done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices

2. CI/CD

- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing

3. ORCHESTRATION

- Kubernetes is the market-leading orchestration solution
- You should select a Certified Kubernetes Distribution, Hosted Platform, or Installer
- cncf.io/ck

kubernetes
CNCF Graduated

4. OBSERVABILITY & ANALYSIS

- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing-compatible implementation like Jaeger

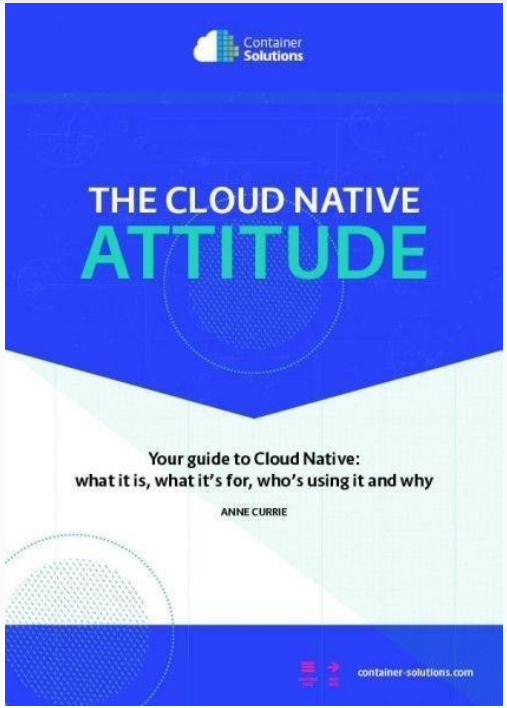
Prometheus
CNCF Incubating

fluentd
CNCF Incubating

5. SERVICE MESH AND DISCOVERY

- CoreDNS is a fast and flexible tool that is useful for service discovery
- Envoy and Linkerd each enable service mesh architectures

Cloud Native Platform



Cloud Native Platform

✅ 1. Use Infrastructure-as-a-Service (IaaS)

Cloud Native Platform

✓1. Use Infrastructure-as-a-Service (IaaS)
✓2. Microservices architecture

Cloud Native Platform

✓1. Use Infrastructure-as-a-Service (IaaS)
✓2. Microservices architecture
✓3. Containerize

Cloud Native Platform

✓1. Use Infrastructure-as-a-Service (IaaS)
✓2. Microservices architecture
✓3. Containerize
✓4. Automate

Cloud Native Platform

✓1. Use Infrastructure-as-a-Service (IaaS)
✓2. Microservices architecture
✓3. Containerize
✓4. Automate
5. Orchestrate

Summary

- **Monolith** makes you slow
- **Microservices**:
  great benefits and big challenges
- **Cloud Native**:
  to fully benefit from Microservices

Thanks!

Tomasz Tarczynski
@ttarczynski

Gigaset