ELCA for
ELCA
10.04.2018

CloudTrust

Digitalization, Security and “CASB”

Sébastien Pasche
Lead Architect

Majeri Kasmaei Chervine
DevOps

ELCA

Who am I

I am a computer security enthusiast
- Security contests, Malware, Web architecture

Open source contributor since 2001

Volunteer in multiple organizations

Addict to absurd & abstract arts

Sebastien Pasche
Twitter @braoru
@ELCA since 01.2017
Architect @leshop 2012-2017
Freelance (mainly biomed) Architect 2006-2017

“The corollary of constant change is ignorance. This is not often talked about: we computer experts barely know what we’re doing. We’re good at fussing and figuring out. We function well in a sea of unknowns. Our experience has only prepared us to deal with confusion. A programmer who denies this is probably lying, or else is densely unaware of himself.”

Ellen Ullman, Close to the Machine: Technophilia and Its Discontents

Agenda

- Few words about Cloudtrust
- Our way of developing
- Our appliance / technological stack

Digitalization

Lets Focus on Data & Runtime environment

Digitalization & Dreams

Externalize Runtime and/or Operating platform
Local PAAS, Remote PAAS, Hybrid PAAS

Reduce your operational cost on OPEX and CAPEX
Pure SAAS Software, Remotely managed PAAS

Ability to change from one providers to another
No Vendor lock-in, Keep your own key approach

Don't care about availability problematics
Let providers admin deal with my *almost* running containers

Simplify our architecture
Try to move from an assembly of product to an unified approach

Our customer current dreams about Digitalization

No Vendor lock-in

Keep you own keys approach (KYOK)

Flexibility between on-premise and SAAS

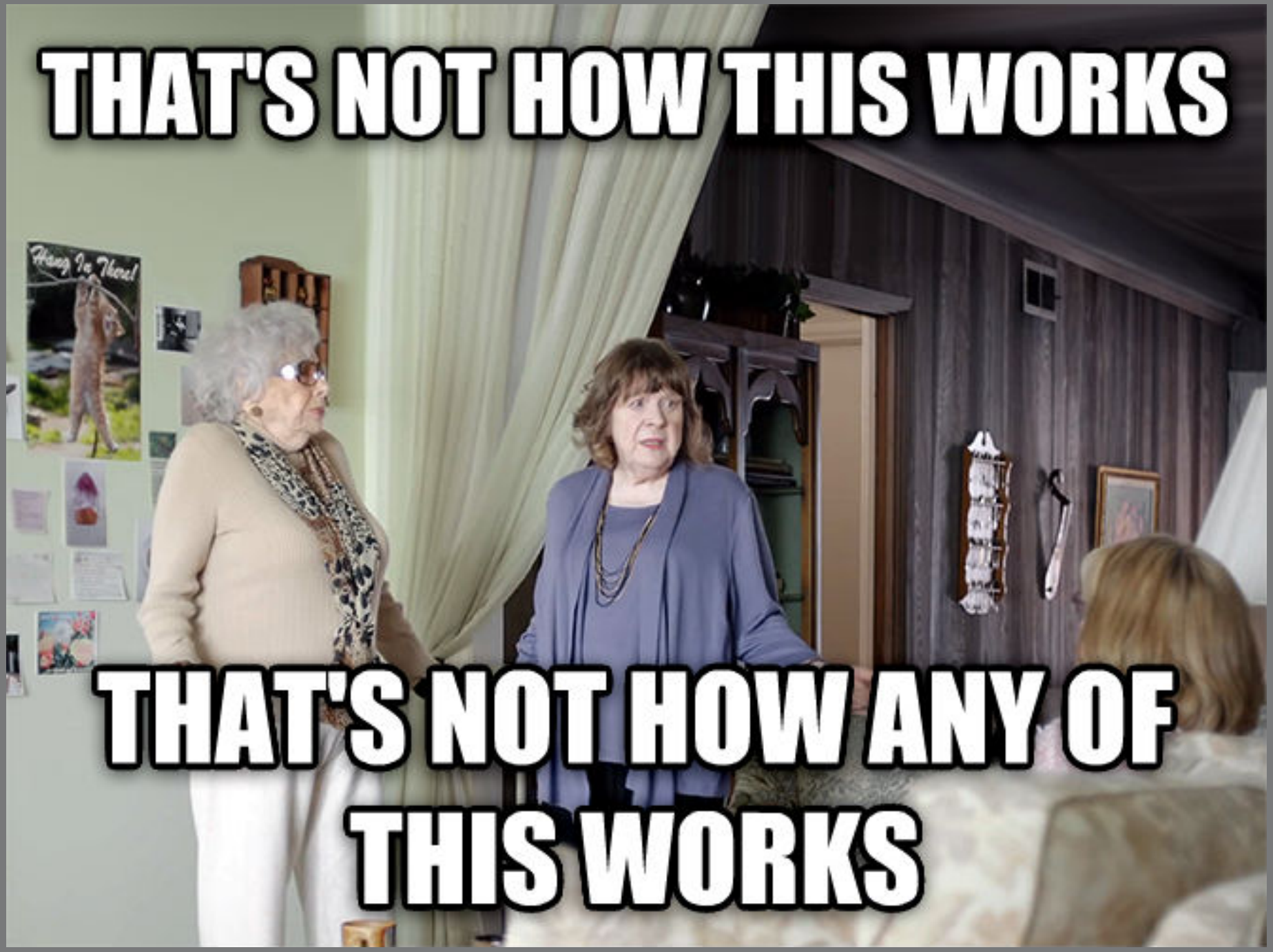
Replace assembly of multiple **complex** Product by an unified approach

Expertise on integration and help with process and strategy design

Transparency on how it's working and what we are doing with data (can be audited anytime)

Support on **open source** software base

High **availability** on a large multi-cluster scale


THAT'S NOT HOW THIS WORKS
THAT'S NOT HOW ANY OF THIS WORKS

What could go wrong

Data privacy
Ensure that sensible information is not disclosed

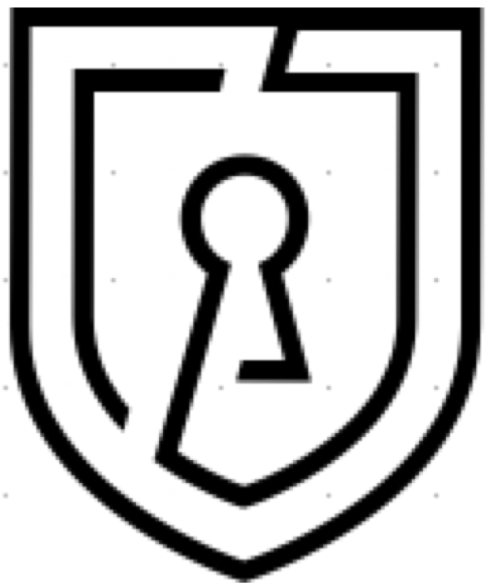
Access & location
Keep control over data access and its location

B2B collaboration
Allow secure collaboration with B2B while keeping lower costs

Compliance
Ensure compliance with legal regulations

Business <-> Technical words translators

Access SSO, ABAC, Access policy, MFA	Visibility, Compliance
Manage B2E, B2B, B2C, Users provisioning, Workflow, Branding	Visibility, Compliance, Identity management
Control Shadow IT, License metering, behavior analytics, reporting	Visibility, Compliance, Threats protection, Cost optimization
Protect Field Protection, Searchable encryption, Proxy/Reverse	Threats protection, Data Loss Prevention, Privacy



CloudTrust

CloudTrust vision

OpenSource and Transparent

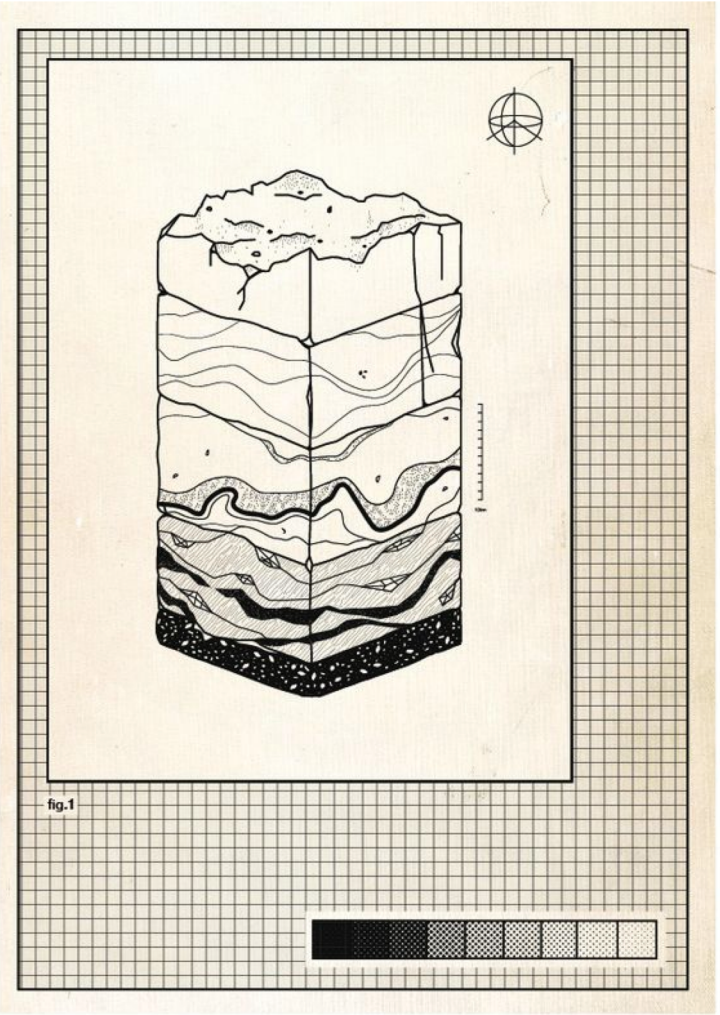
Support multiple deployment target
    SAAS, Appliances, Customer hosted PAAS, Hybrid

Organized as a toolkit with dedicated distribution

Fix gap between OpenSource product and our fields experiences

Keep OPEX stable
    Multi tenancy, **Use vanilla technologies**,
    Use/**Improve open source**, clustering.

Embark ELCA security **knowledge** and project **experiences**



Technology dietetic

How we work with technology

III

Independent Immutable Idempotent

Infrastructure layer examples

Keep deployment idempotent

Software need to be deployed independently

No mutation on target running environment outside of deployment (immutability)

Software layer examples

Keep business logic isolated from plumbing

Avoid strong coupling with external components

API and inter-services must be stateless and as idempotent as possible

Management layer examples

Train people

Automate setup & operation

Development lifecycle

Component integration & dev

Avoid over engineering

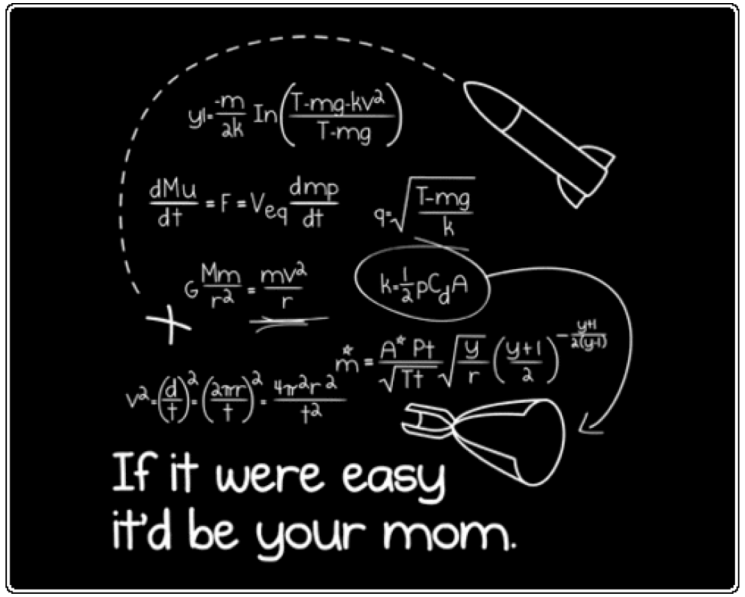
Avoid unnecessary genericity, keep it simple so that it solves stories without breaking the evolution path

Avoid massive and non-flexible frameworks / designs

Technology changes, requirements changes, frameworks, technologies and design patterns shall not stop improvements

Prefer open and well-maintained framework when it's possible

Keep updated with security issues, large know-how, easier dev integration.



Component integration & dev

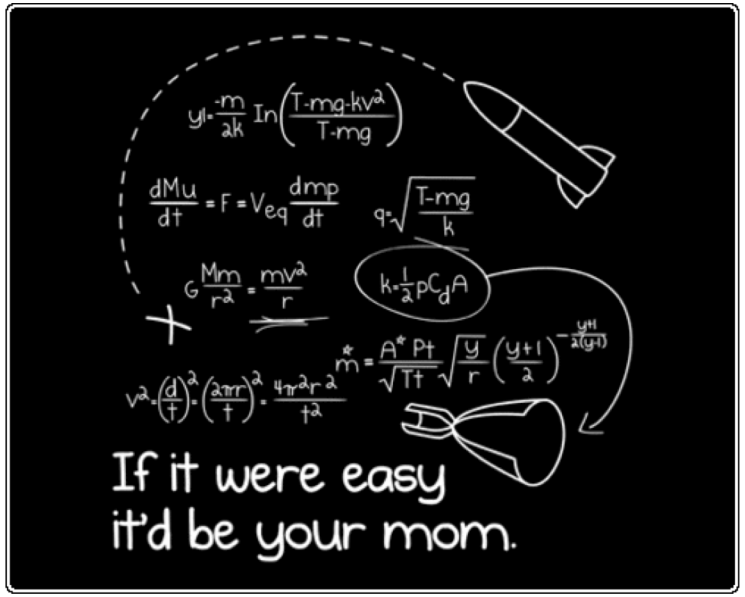
Workaround is not a new dev methodology
If something doesn’t fit, doesn’t work or can’t be improved, don’t use it

If something doesn't work, change it

Sometime things need to be created

Integrate technology when it make sense but not directly build upon them
Independence between your software and integrated component

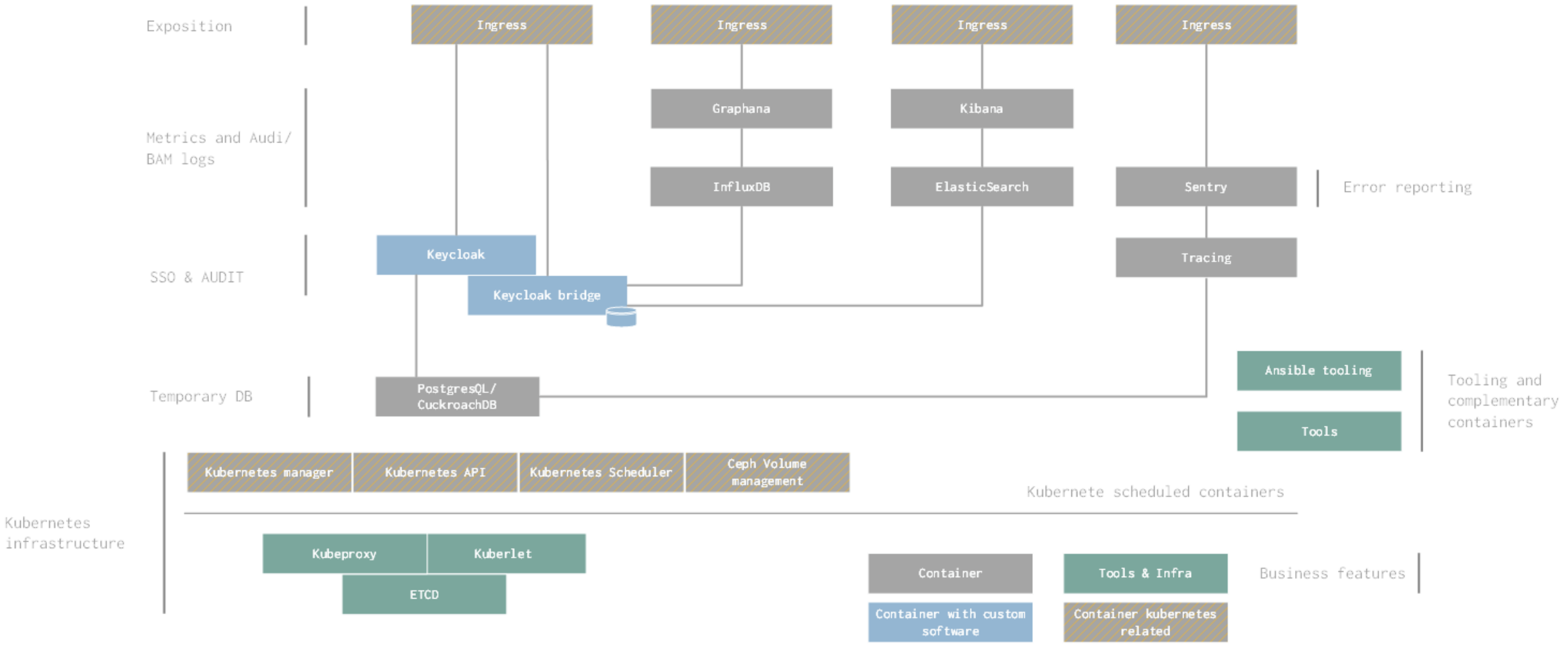
Do not hesitate to learn
Learn form yourself and from others



CloudTrust stack and Runtime

How we build software

Global overview



Deployment & build (Ansible)

All services are deployed by Ansible

Use environment configuration dedicated git repositories then re-create what the configuration require.

Example of environment configuration (dev-config) : https://github.com/cloudtrust/dev-config

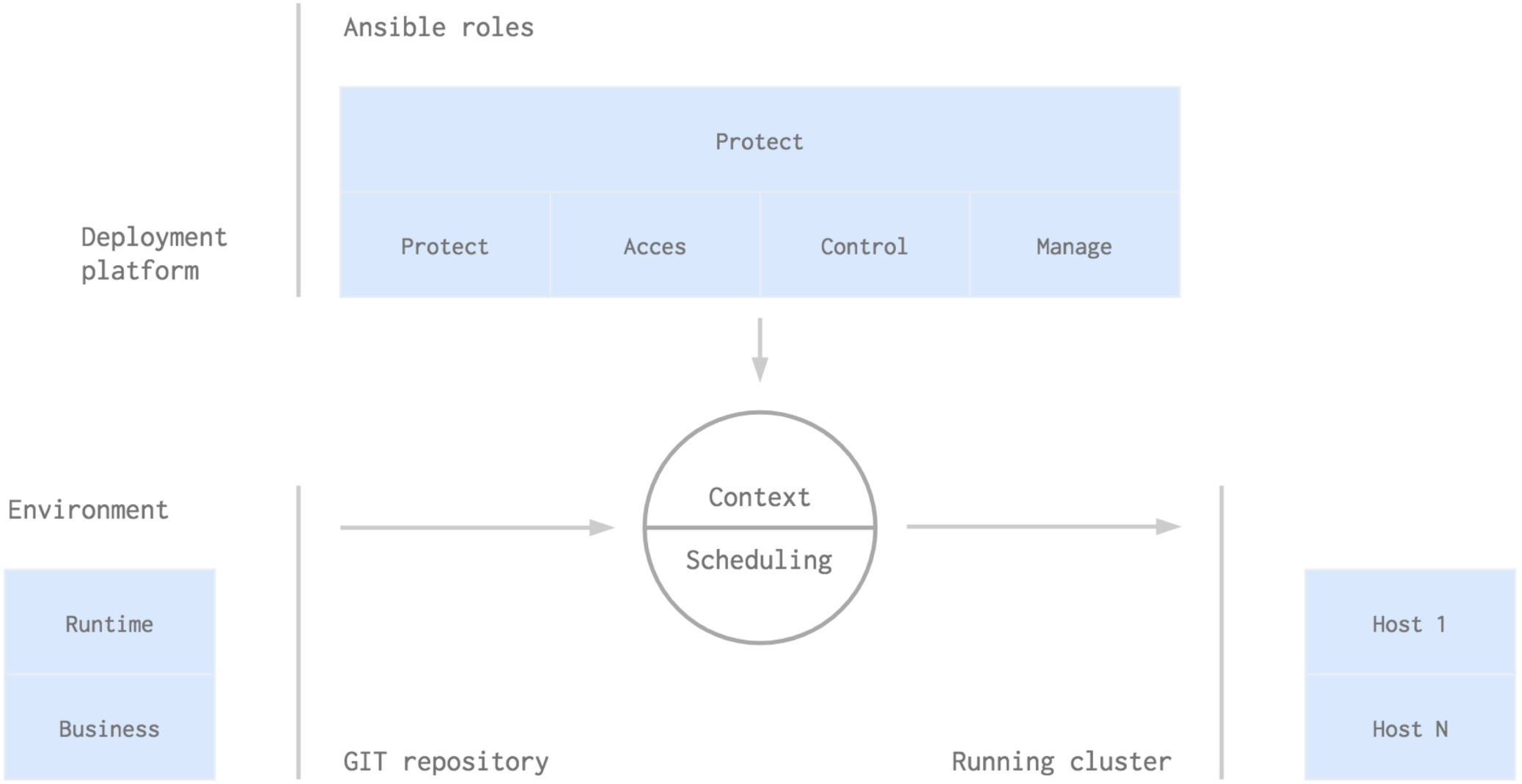
Build & deployment are composable

You don’t need to deploy everything every time

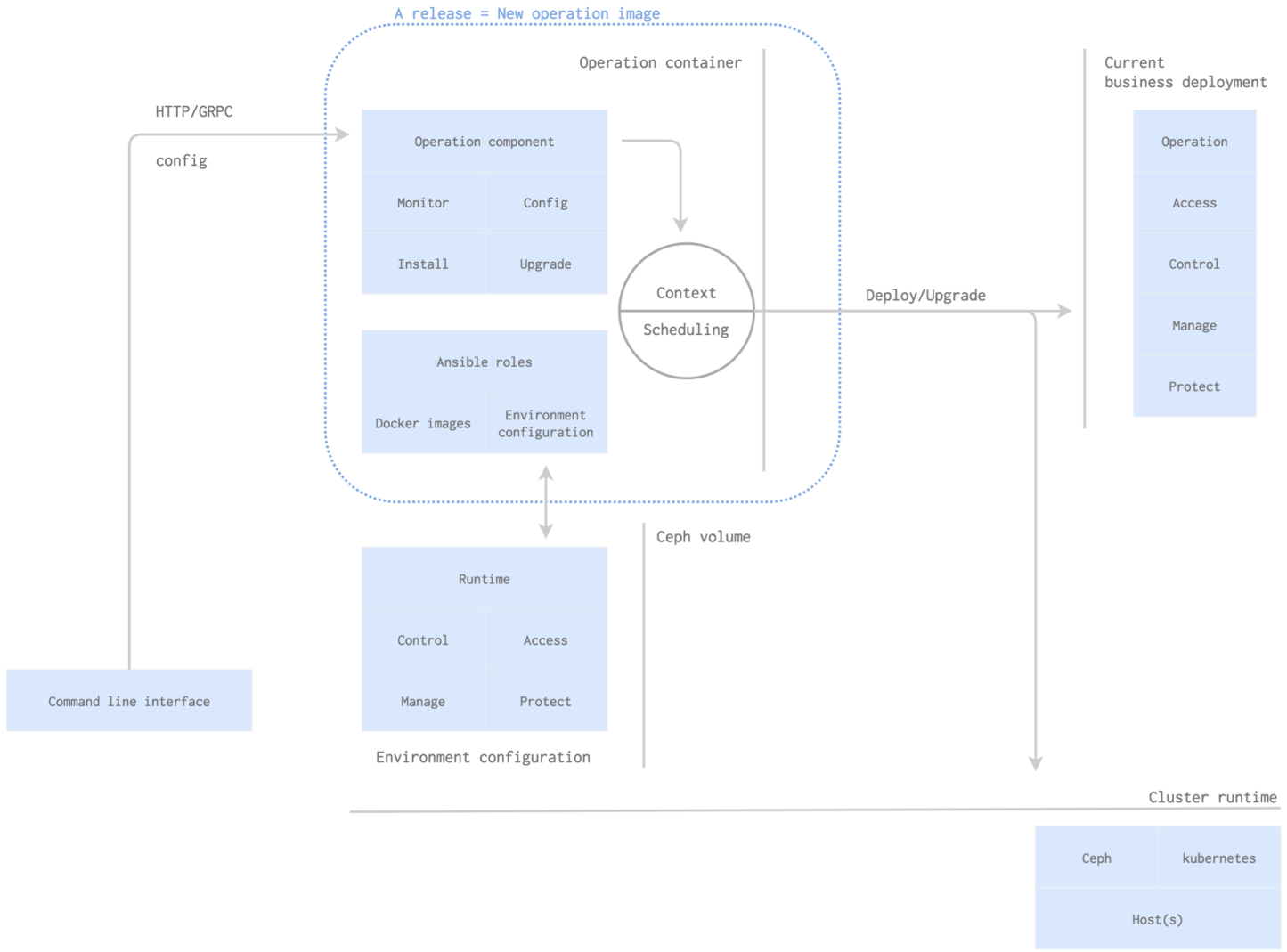
Current runtime deployment for on-premise setup

https://github.com/cloudtrust/on-premise-runtime

Deployment & build (Ansible)



Deployment & build (Ansible)



Deployment & build (Ansible)

Important deployment repository

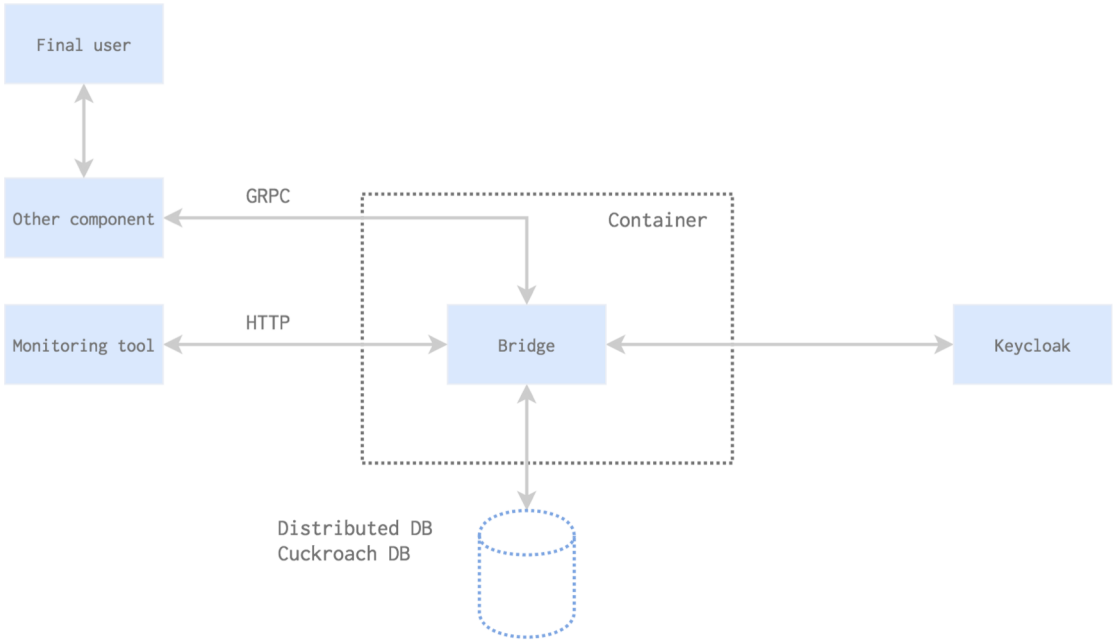
CloudTrust docker base images : https://github.com/cloudtrust/baseimage

Common CloudTrust VM config : https://github.com/cloudtrust/cloudtrust-common

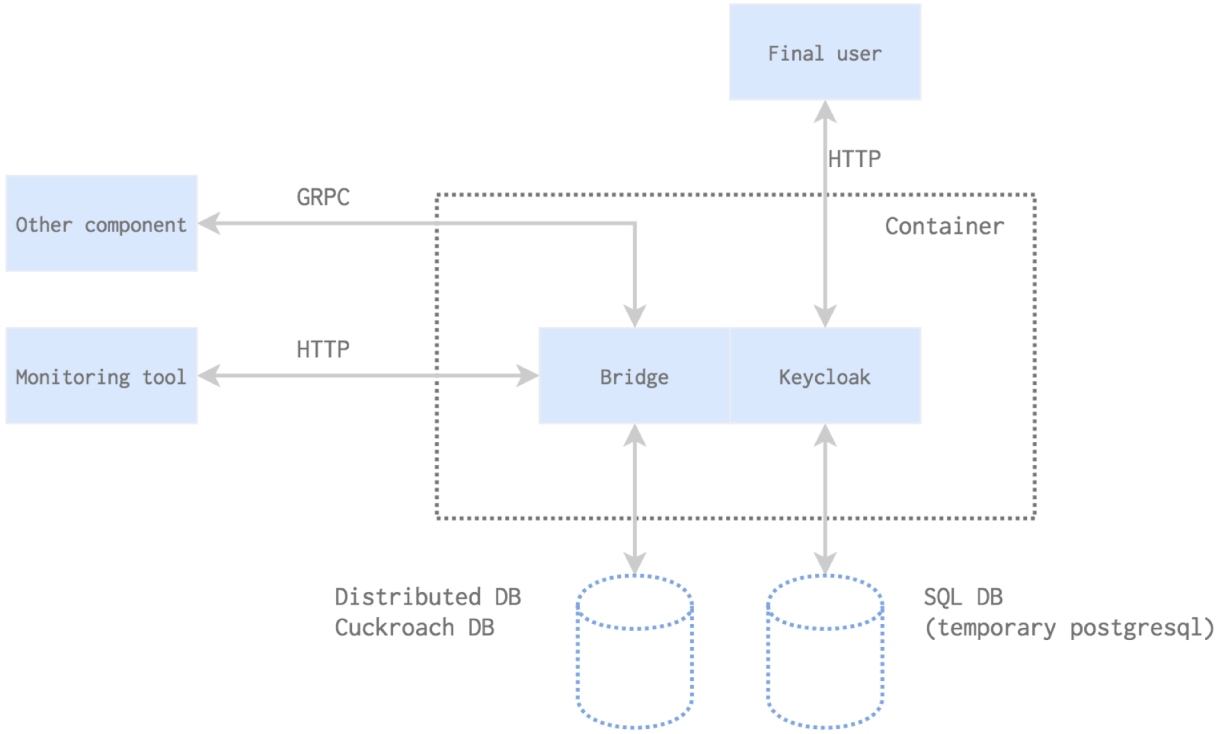
Containers & external services integration

How we connect things

Bridge



External bridge



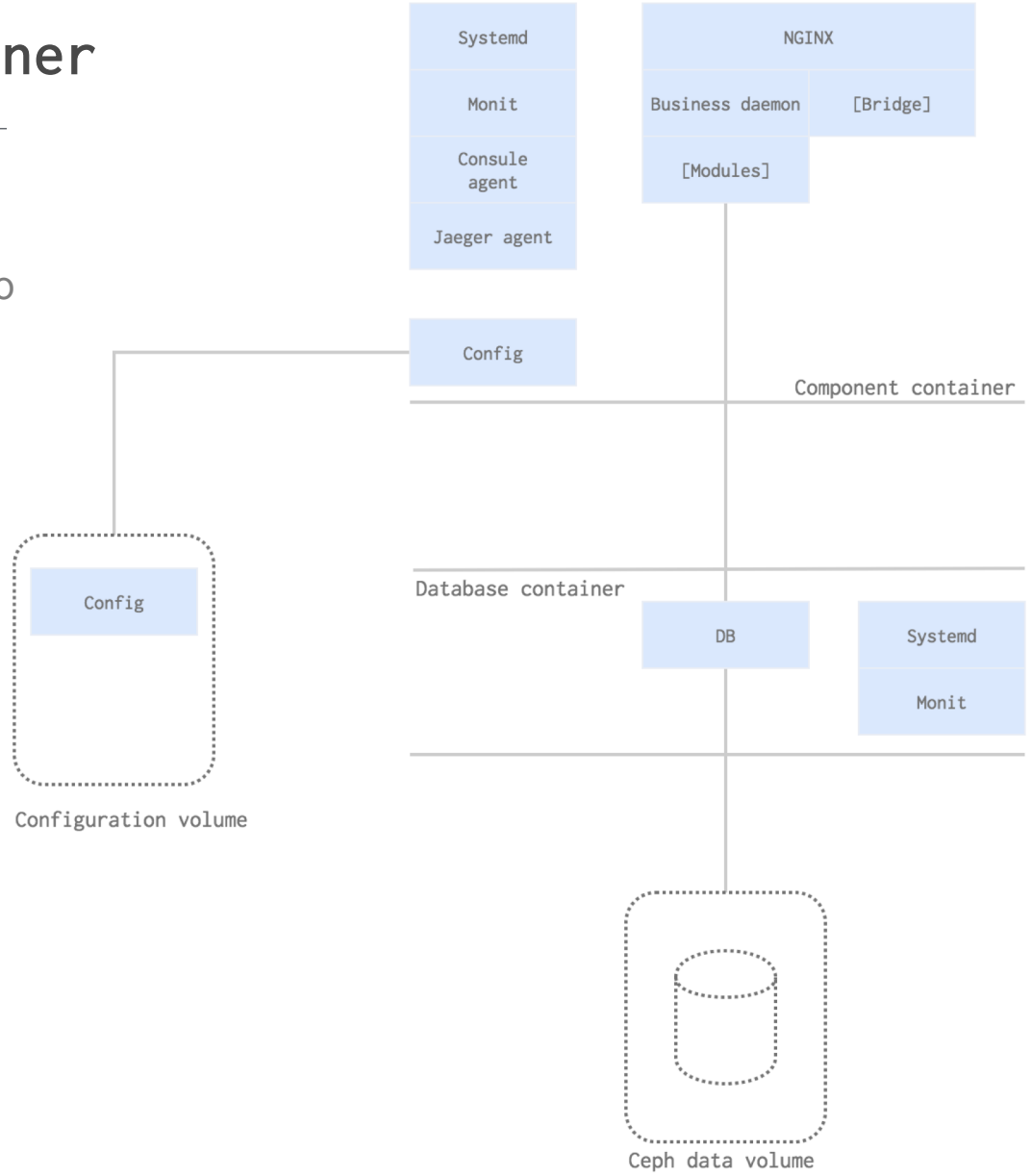
Internal bridge

Container

Several interal services involved
Systemd to manager the container internal, Monit to restart failed services and so on

Not only a run cmd
Cloudtrust software are managed and instrumented

External containers
DB have their won container and configurations & data are stored and distributed through volumes



Service communication - GRPC

HTTP/2 based serialized communication developed by Google
PROTOBUF serialized data over HTTP/2 communication channel

Abstract HTTP/2 Complexity
Streaming, bi-directional streaming, data segmentation

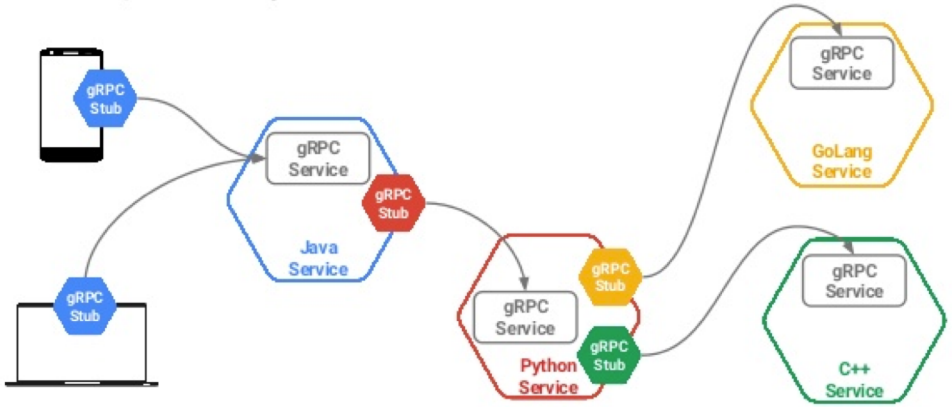
Contract based but flexible
Client must implement contract (proto) but just implement what is required

Largely used & supported
C++, Java, Python, Go, Ruby, C#, Node.js, Android java, Objective-C, php

Developed to address cloud problems
Micro services communication, mobile device communication

```
message Person {
  string name = 1;
  int32 id = 2;
  bool has_ponycopter = 3;
}
```

Interoperability



Service communication - FLATBUFFER

Serialization technology developed by Google to succeed PROTOBUF

Current main developer of PROTOBUF is Google

Developed in abstraction of the communication channel

Can be directly used over HTTP/2, GRPC and multiple communication technology and buses

Faster than PROTOBUF

Largely supported

C++, Java, C#, Go, Python, JS, C, PHP, Ruby

CloudTrust Access

Keycloak

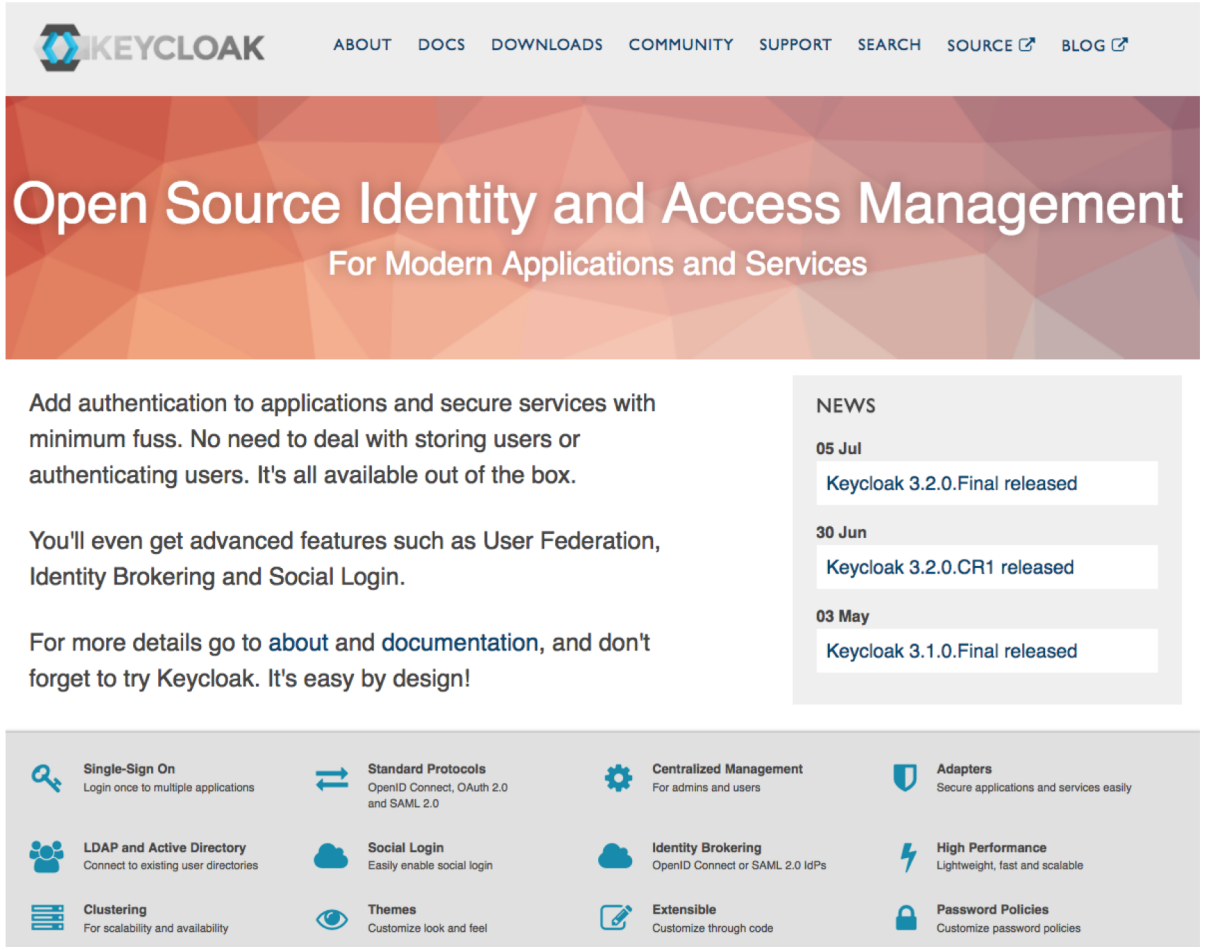
Offers most of required features

Clean codebase and architecture

Highly extensible and **customizable**
Modules, extensions, themes, …

Open source with possible redhat support

Documented and complete REST API



Find the holes

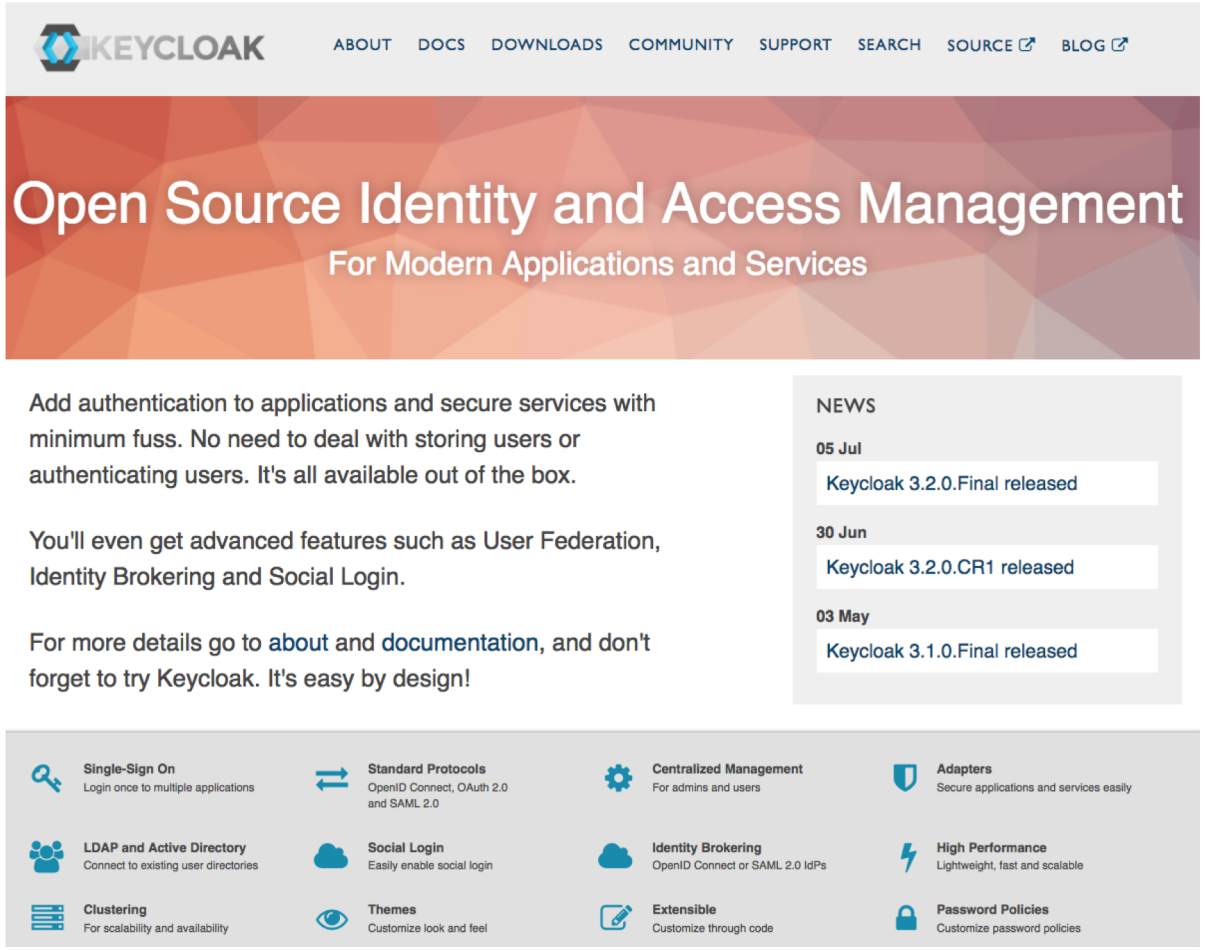
No easy hotplug clustering

No support of Microsoft WSFED

No support for all Access control use-cases

No easy way to get Business & Technical audit logs & metrics

API not really adapted to Reactive programming patterns



Fill the holes

High availability, node hotplug and multi DC support

Make Keycloak work with cockroach-db
https://github.com/cloudtrust/keycloak-cockroach

Still work in progress

Standalone mode with a shared but distributed DB

Require modification of SQL queries and a keycloak data access logic

Fill the holes

Audits logs, performances monitoring and access event-based development

Create a module to send everything happen within keycloak
https://github.com/cloudtrust/event-emitter

Handle retry and buffering messages

Can send event with in **JSON** or **flat** buffer

Embed a local snowflake to generate unique and non-colliding event id

Fill the holes

Microsoft WSFED support

Implement WSFED protocol within keycloak
https://github.com/cloudtrust/keycloak-wsfed

Able to replace Microsoft ADFS

Support Token claim mapping

Fill the holes

Keycloak authorization module

Add authorization capabilities to keycloak for a given client, whether the client itself has the capability to handle authorization or not.

https://github.com/cloudtrust/keycloak-authorization

Any type of keycloak client can have an authorization layered on top of authentication, not just OIDC clients.

The authorization only works as long as keycloak is in the path of requests to a resource.

Fill the holes

Keycloak realm definition live backup & restore

Add http endpoint to backup & restore keycloak realms.

https://github.com/cloudtrust/keycloak-export

For cluster import & export feature, a dedicated job will be added to the current keycloak bridge

Instrumentation & tools (Episode I)

Errors mining

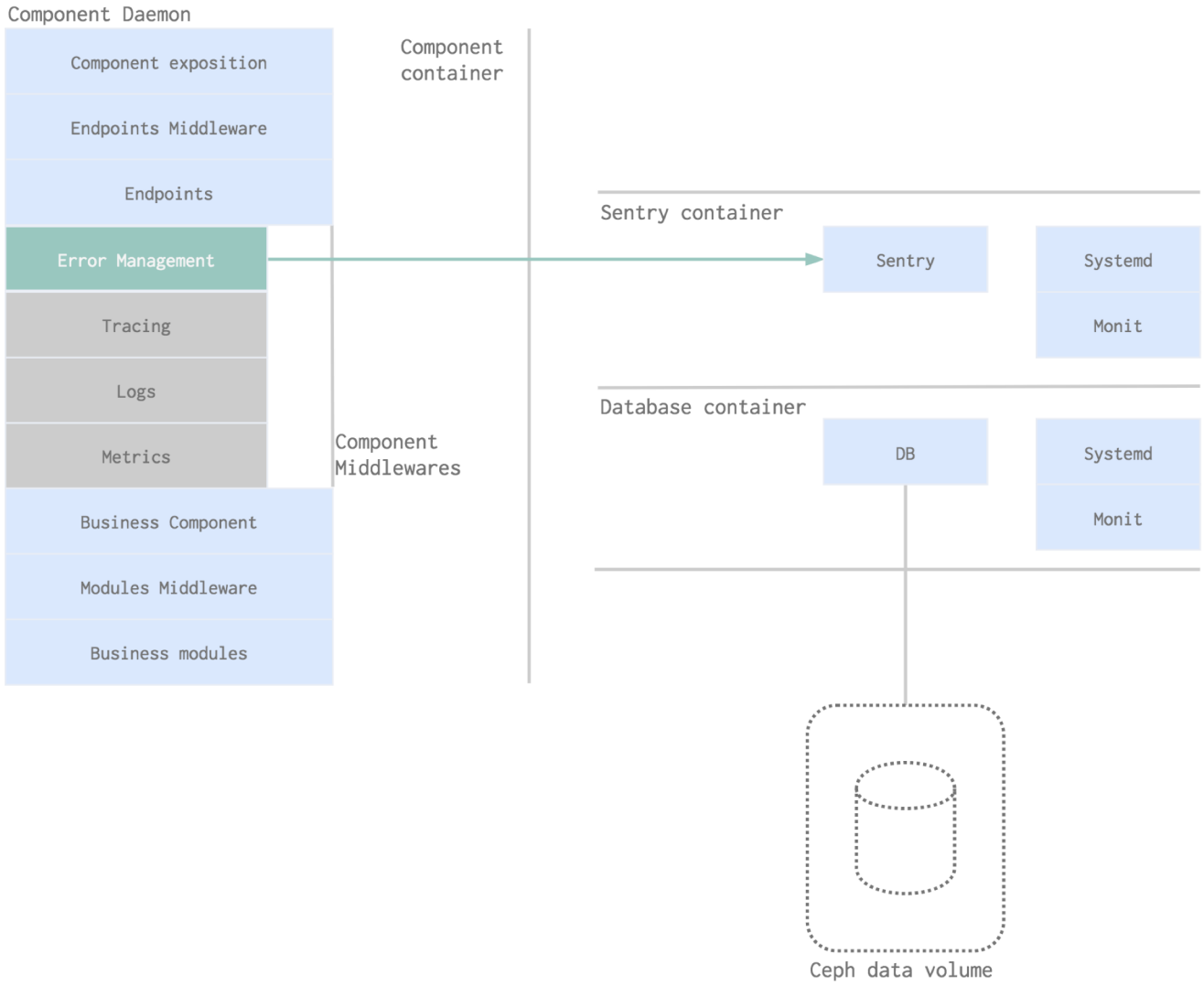
Error collection

All important error are reported within log and sent to a remote error management systems

We currently use sentry as an error management solution : https://sentry.io/welcome/

Error tracking is implemented as a dedicated error management middleware

Example : https://github.com/cloudtrust/flaki-service/blob/alpha/pkg/flaki/tracking.go#L33



Tracing

Every call entering CloudTrust architecture is traced in a end-to-end way

We currently use jaeger as an tracing management solution

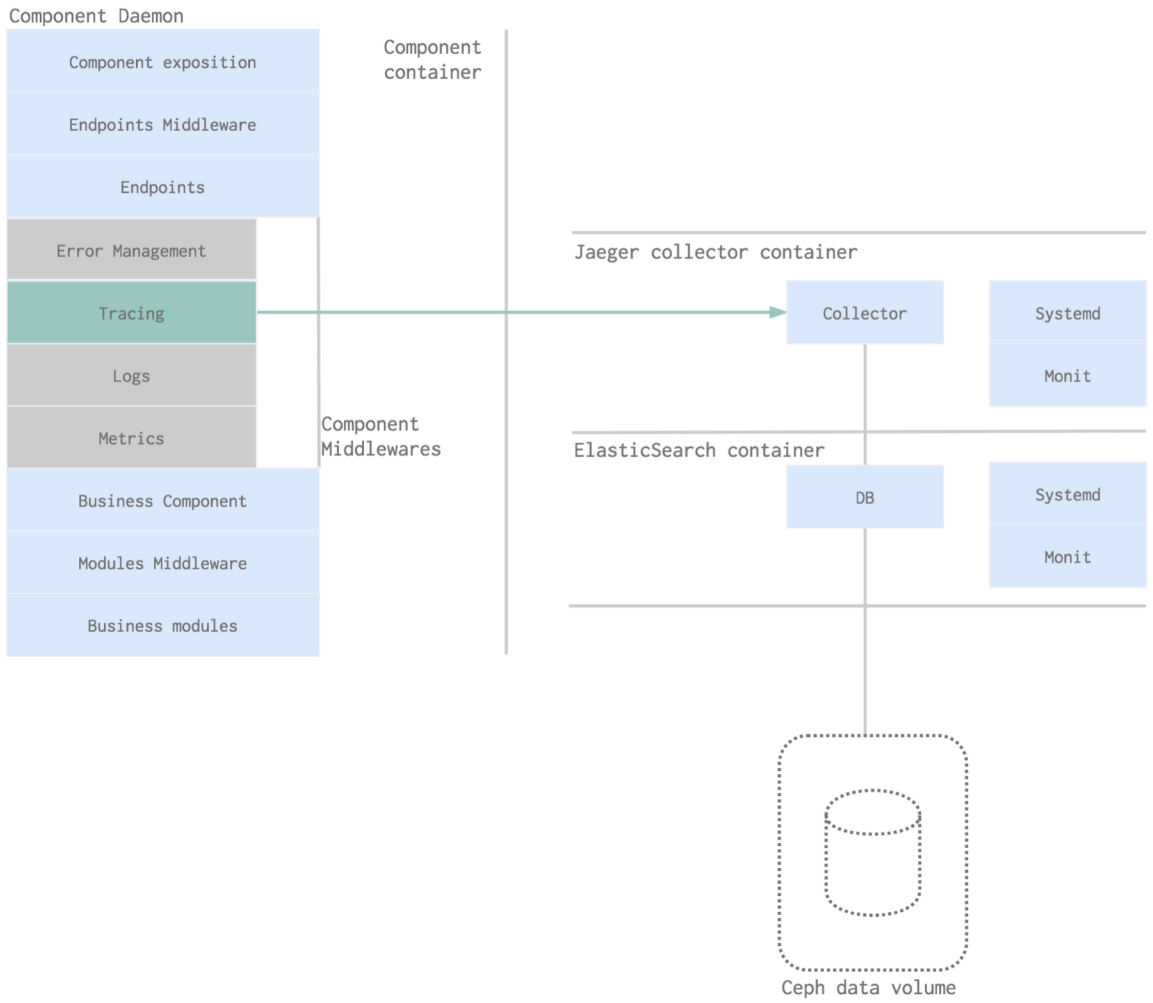
https://github.com/jaegertracing/jaeger

Tracing middleware are used in multiple location

- At exposition layer (Transport)
- Endpoints layer
- Business component layer
- Business Module layer

Example

https://github.com/cloudtrust/flaki-service/blob/alpha/pkg/flaki/tracing.go#L106



Metrics

Systems metrics

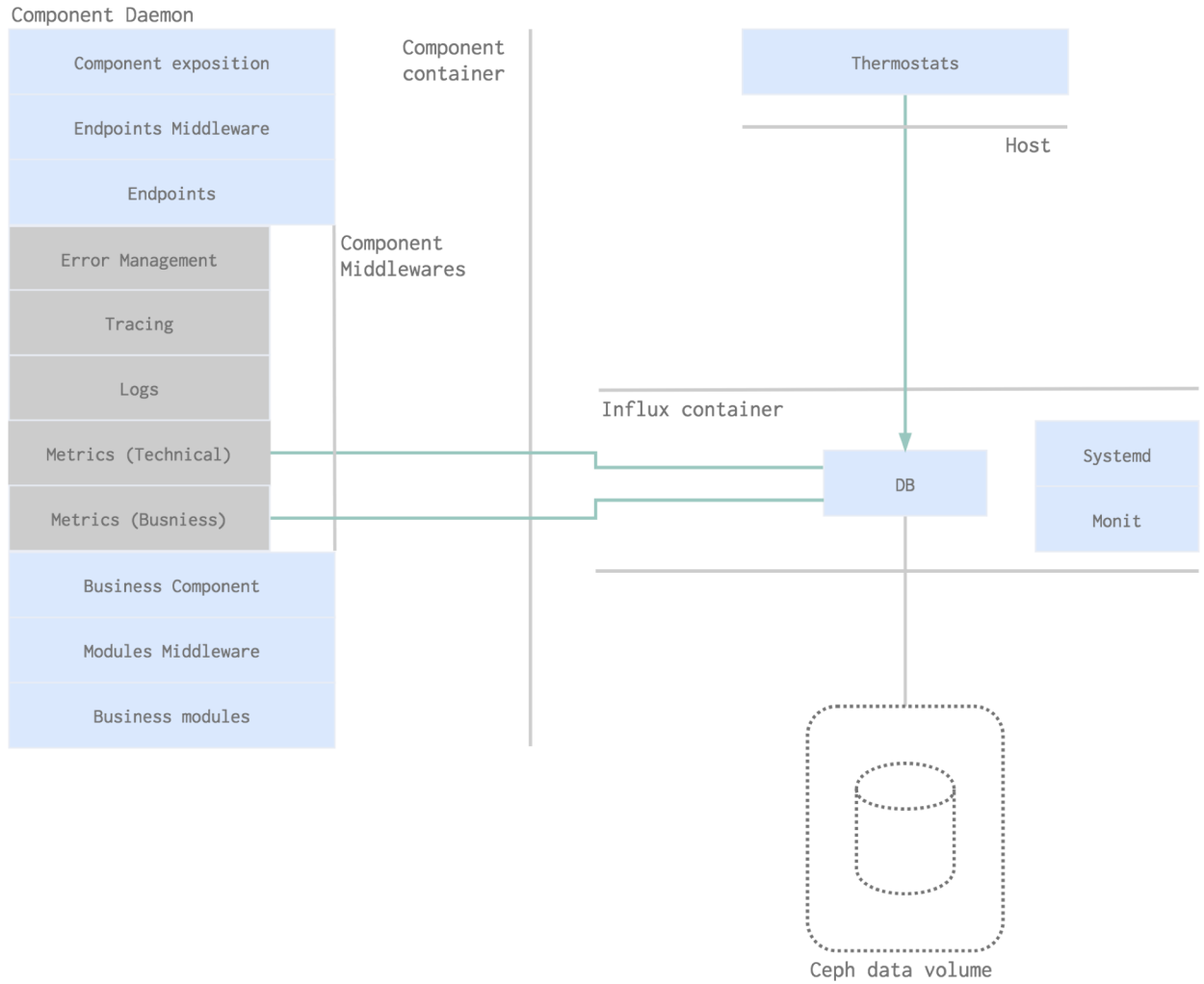
Systems metrics are automatically send to a time-serie database through telegraph
https://github.com/influxdata/telegraf

GO daemon metrics

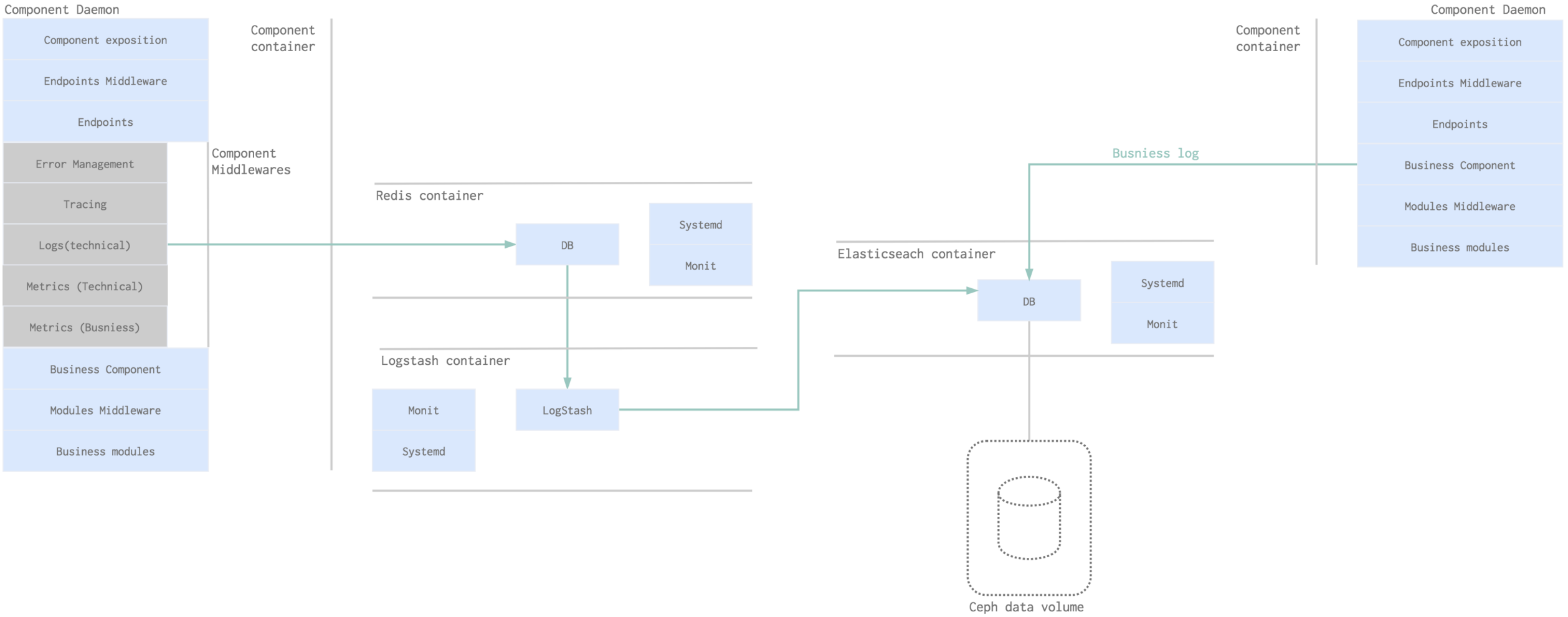
Dedicated middleware are used to send timming of each exposed endpoint and each measured method to a time-serie database

Dedicated middleware are used to send Business metrics

Middleware documentation :
https://cloudtrust.github.io/doc/chapter-godevel/middleware.html



Technical & Business log



Non unit-testing

Test should as much as possible be runnable from developer laptop

Container service test

Should control if the container can offer required services
ex : Is keycloak available ?

Container test

Should control if container pieces are assembled in a correct way
Ex : Is keycloak started by Systemd ?
Ex : If I kill keycloak, did monit restart it ?

Acceptance test

Business functionality and all corresponding effects it should have on the system
Ex : If I do an user creation on keycloak bridge, did all the logs, metrics and tracing have been created correctly ?

Functional test

Business functionality and connected component
Ex : If I do an user creation on keycloak bridge, did it create it correctly and can I log on with ?

CLOUDTRUST GO FRAMEWORK

Composable microservice framework

Composabke Middleware

Throttling	Circuit Breaking
Logs	Metrics
Tracing	Tracking
Errors management	

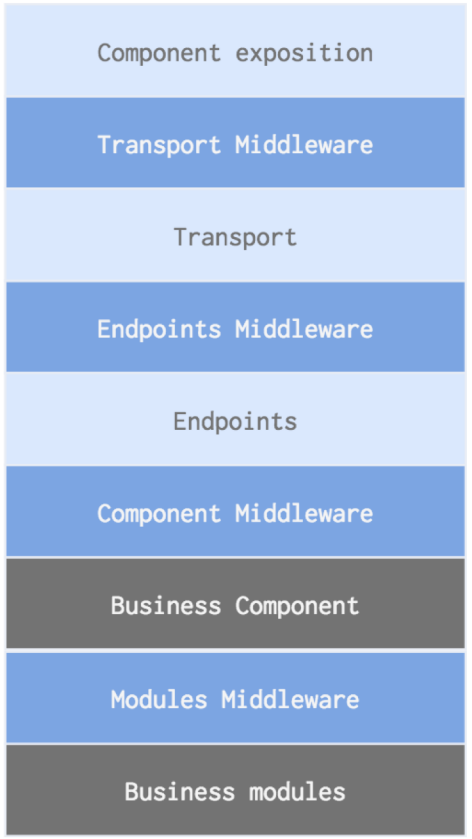
Business Component

Composed of Business modules
Contains Business logic
Context aware (session)
Manipulate immutable flattbuffer objext

Business modules

Purely stateless and indempotent code
Compose of required libs & drivers
Contain pure business code
Agnostic from any context

Framework organisation



HTTP1/2
GRPC

Flattbuffer objects

Indempotent element

CloudTrust GO framework

A dedicated presentation will be created to present all our go-framework features

Template daemon

Flaki : https://github.com/cloudtrust/flaki

Features :

- Simple to understand
- Implement all library and middleware as an example

CloudTrust framework (purely composition based)

Middleware : Tracing, Tracking, Log, Metrics, Error management, Throttling, Circuit breaking
Transport : GRPC, HTTP2
Database : Our current default database is CockroachDB (https://www.cockroachlabs.com/)
Monitoring : Embedded monitoring and distributed monitoring

Cloudtrust GO framework

Framework features at this time

- Purely stateless
- Respect III
- Embedded service monitoring
- Embedded distributed monitoring
- HTTP and GRPC supported as transport layer
- Configuration server ready & support hot config reload on change

CloudTrust home-made library

Distributed job systems

- Actors based
- Split brain tolerant
- https://github.com/cloudtrust/go-jobs

SRP authentication support

- https://github.com/cloudtrust/srp

FPE (Format preserving encryption)

- https://github.com/cloudtrust/fpe

Cloudtrust GO framework - monitoring

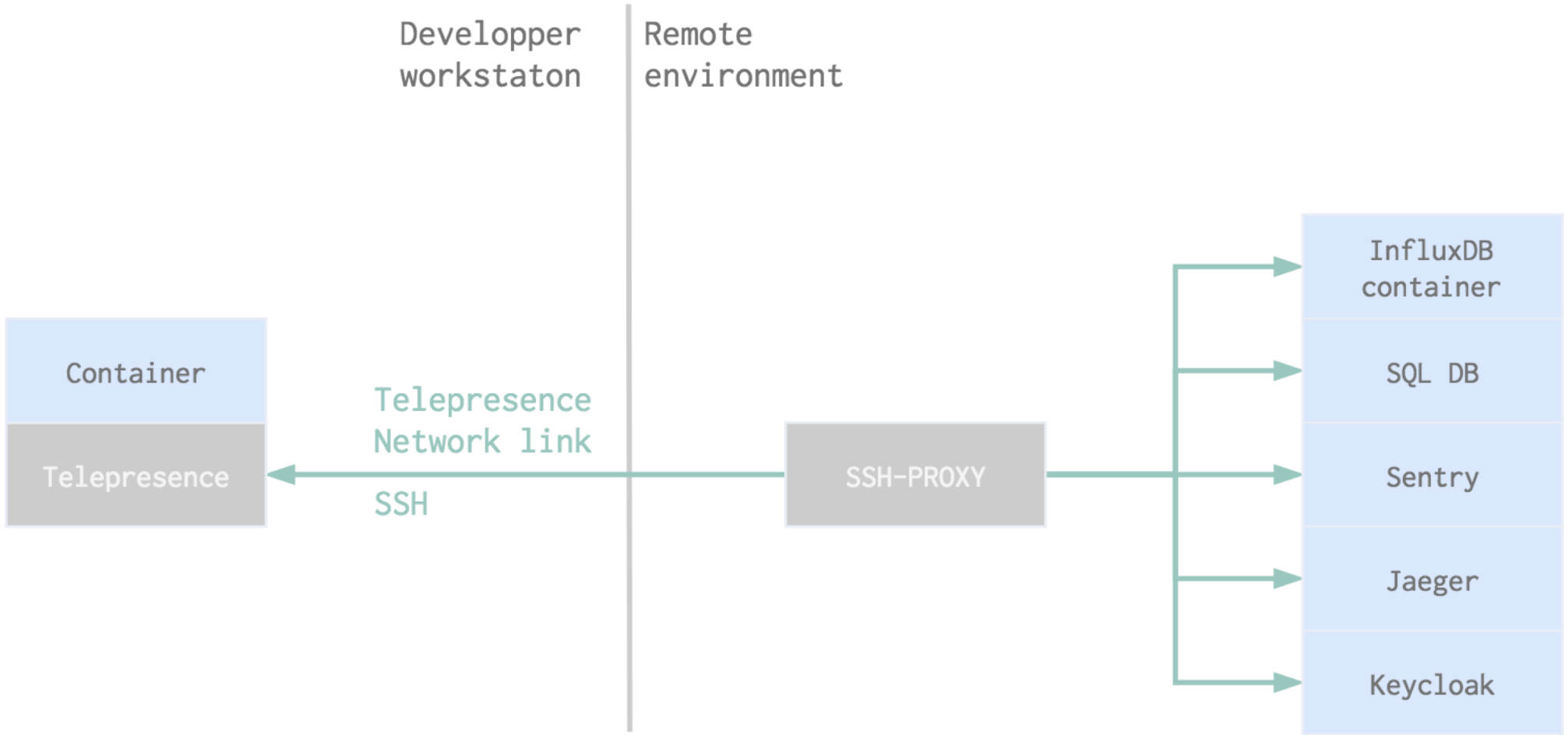
Functional tests examples

```
curl localhost:8888/health
{
  "influx": "OK",
  "jaeger": "OK",
  "keycloak": "OK",
  "redis": "OK",
  "sentry": "OK"
}
```

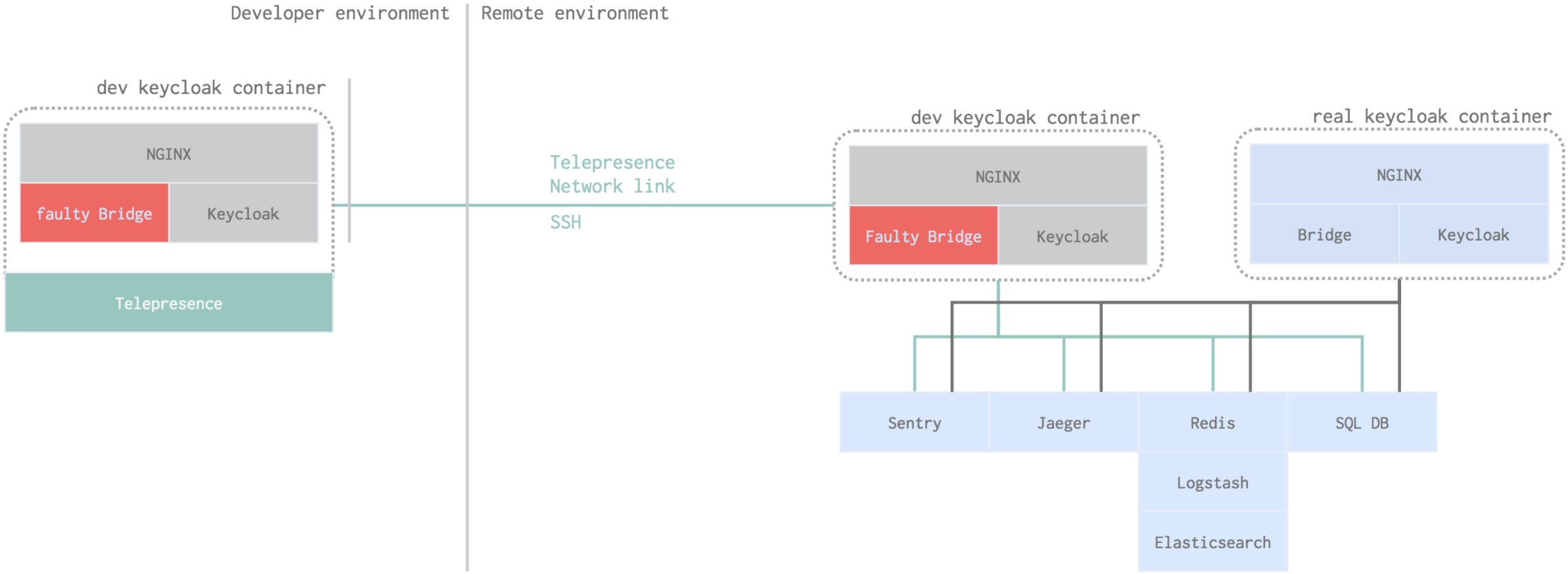
```
curl localhost:8888/health/keycloak
{
 "health checks": [
  {
   "name": "create user",
   "duration": "111.563719ms",
   "status": "OK"
  },
  {
   "name": "delete user",
   "duration": "200.225346ms",
   "status": "OK"
  }
 ]
}
```

Dev tool & demo

Telepresence



Demo, tracing and telepresence



Conclusion

Lot of work is still going on (keep an eye one github)

We try to preserve III concept as much as possible in every layers

Self contained appliance but SAAS/PAAS ready

Infrastructure (next steps)
Integration of consul (configuration management) and improvement of operation component

Quality (and next steps)
Code is 100% covered
Infrastructure unit-testing is going on (we target 100%)
Functional test is going (we target 100%)

Automation (next steps)
We are working on packaging setup/upgrade as a whole operation container

Go framework (next steps)
We plan to publish a full slide deck about it

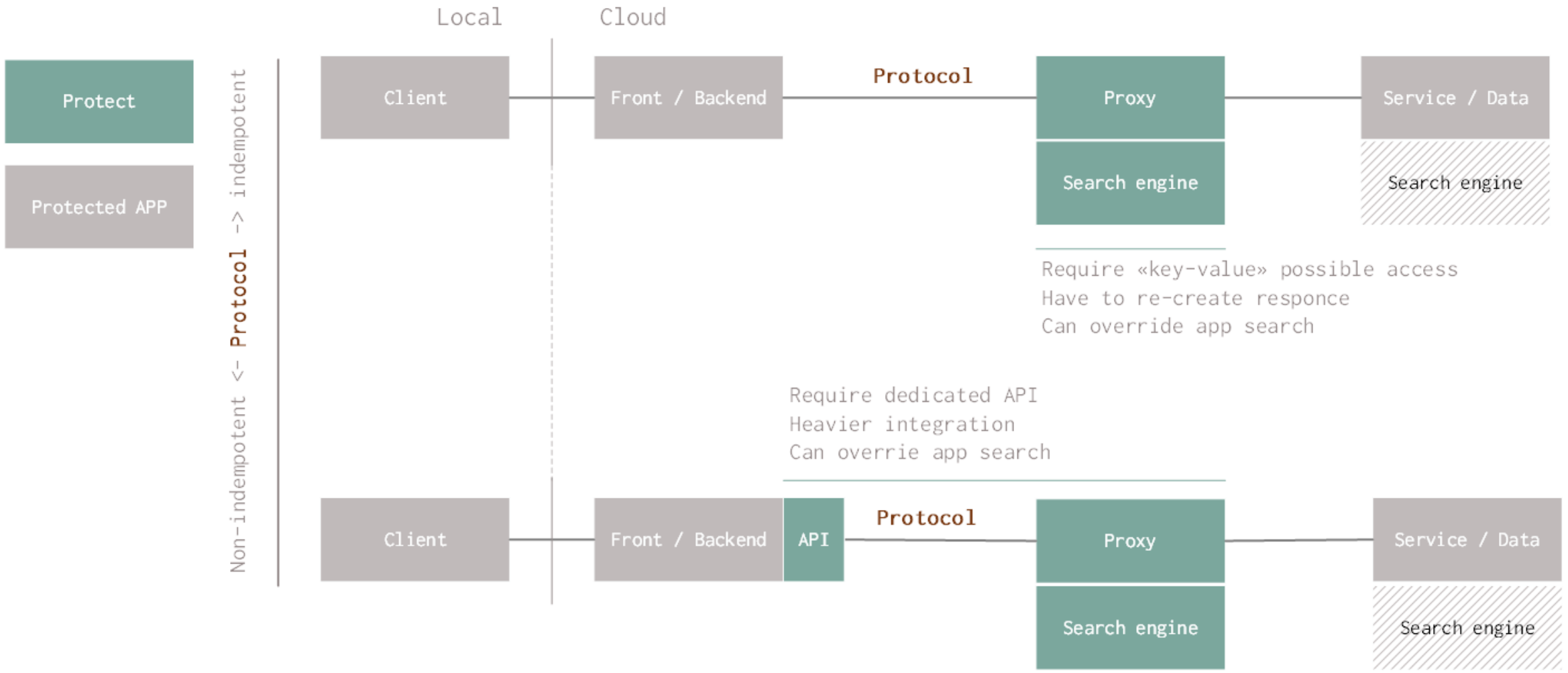
Questions ?

This slide deck is available online
https://github.com/cloudtrust/talks

Protect

Currently analyzed use-cases

Protect from within



Protect from application to customer

Being between app and customer