



KubeCon



CloudNativeCon

North America 2017



proud CNCF member

**Migrating Hundreds
of Legacy
Applications to**



**THE GOOD,
THE BAD,
THE UGLY**

Josef Adersberger, CTO, QAware
[@adersberger](#)



BANG!

HYPERSCALE

OPEX SAVINGS

RESILIENT

SPEED



CIO

Let's bring
all our web
applications
into the cloud!

I've already
canceled our
current data center
contract.

Great goal!
But we'll need
our time.



Chief Architect



Excellent!

You've got one year.



CIO

My priorities:

(1) Security level

(2) Time

(3) OpEx savings

(4) Migration costs



WE WERE BRAVE



AND WE HAD ... IMPEDIMENTS





THE GOOD

An aerial photograph showing a dense forest of evergreen trees on the right side, with a misty or foggy atmosphere on the left side. A grassy field is visible in the top right corner. The word "Visibility" is overlaid in white text on the forest area.

Visibility

Let's architect
the cloud!



BUILD AND COMPOSED
AS **MICROSERVICES**

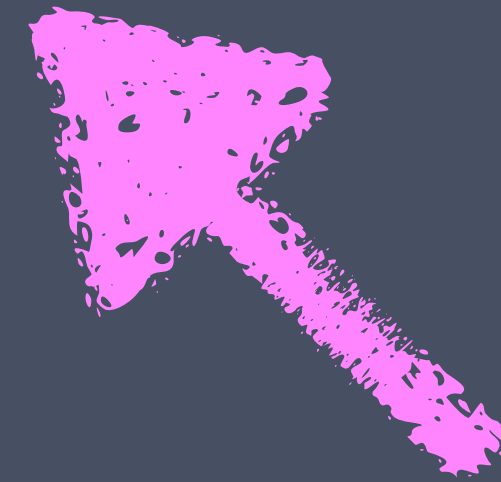
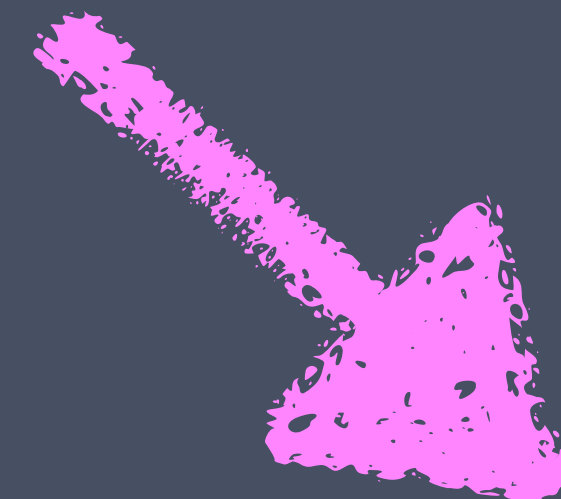
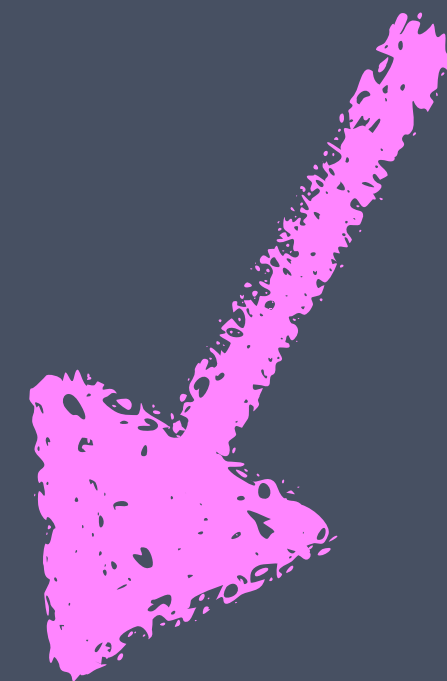
CLOUD NATIVE APPLICATIONS

DYNAMICALLY

EXECUTED IN THE **CLOUD**

PACKAGED AND

DISTRIBUTED AS **CONTAINERS**



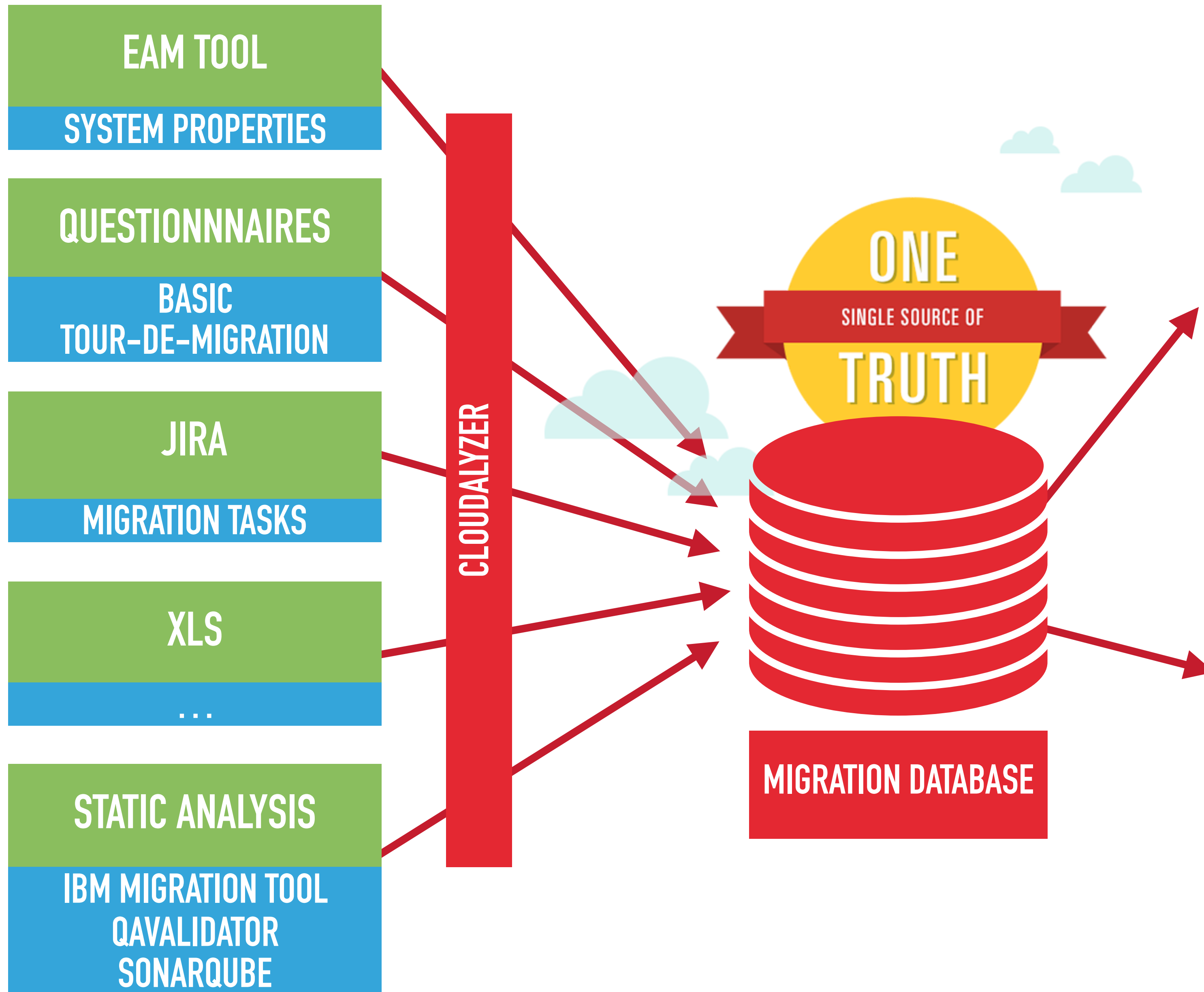
OMG - no greenfield
approach!?

Hundreds of legacy
systems!?



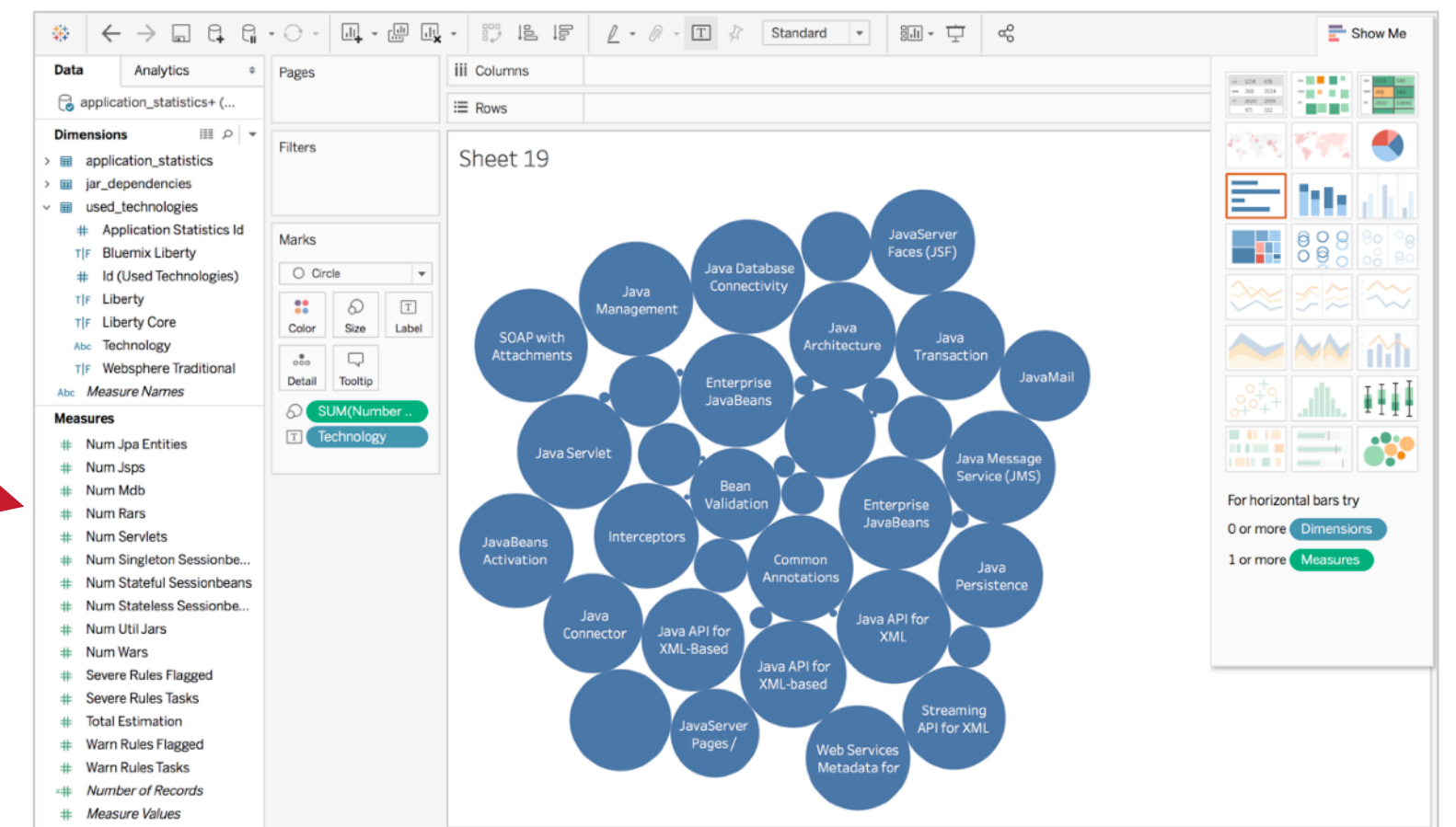
Questionnaire: Typical Questions And Their Motivation

1. **Technology stack** (e.g. OS, appserver, jvm) → What images to provide?
2. Required **resources** (memory, CPU cores) → How many applications will be hard/inefficient to schedule (>3 GB RAM, > 2 cores)?
3. **Writes to storage** (local/remote storage, write mode, data volume) → What storage solutions to provide?
4. **Special requirements** (native libs, special hardware) → What applications will be hard to impossible to be containerized?
5. Inbound and outbound **protocols** (protocol stack, TLS, multicast, dynamic ports) → Are there any non cloud-friendly protocols?
6. **Ability to execute** (regression/load tests, business owner, dev knowhow, release cycle, end of life) → How risky is the migration? Is the migration maybe not needed?
7. Client **authentication** (e.g. SSO, login, certificates) → What IAM and security mechanisms have to be ported to the cloud?



Projekt	AWS Umgebung	Edge Service	Token Prüfung	Schnittstellen	Konfiguration	Diagnostizierbarkeit
Anwendung 45 migrieren(QAWARESPIELUNDT:57)	●	●	●	●	●	●
Anwendung 44 migrieren(QAWARESPIELUNDT:56)	●	●	●	●	●	●
Anwendung 43 migrieren(QAWARESPIELUNDT:53)	●	●	●	●	●	●
Anwendung 42 migrieren(QAWARESPIELUNDT:50)	●	●	●	●	●	●
Anwendung BC migrieren(QAWARESPIELUNDT:4)	●	●	●	●	●	●
Anwendung AB migrieren(QAWARESPIELUNDT:3)	●	●	●	●	●	●
Anwendung YZ migrieren(QAWARESPIELUNDT:2)	●	●	●	●	●	●
Anwendung XY migrieren(QAWARESPIELUNDT:1)	●	●	●	●	●	●

dashboard for information radiator



freestyle analysis with Tableau

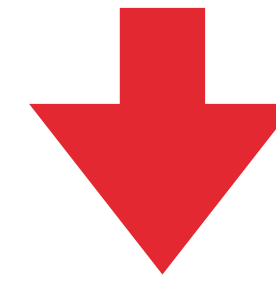
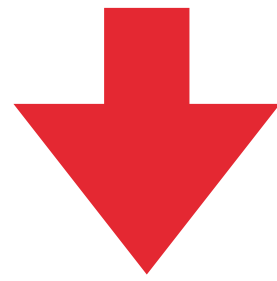
Emergent design of software landscapes



ALL WEB APPLICATIONS (~400)

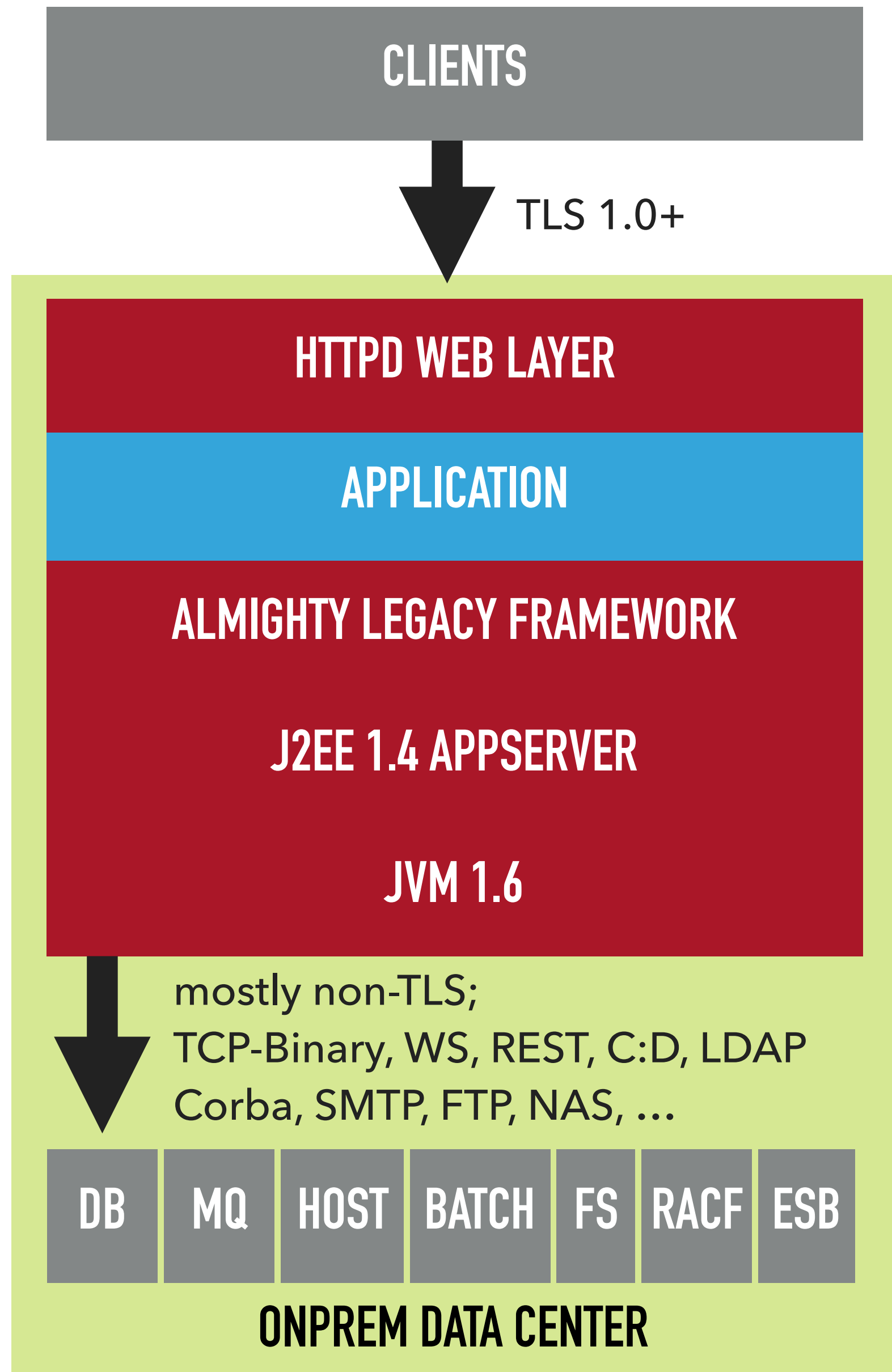
OLD APPLICATIONS (~200)

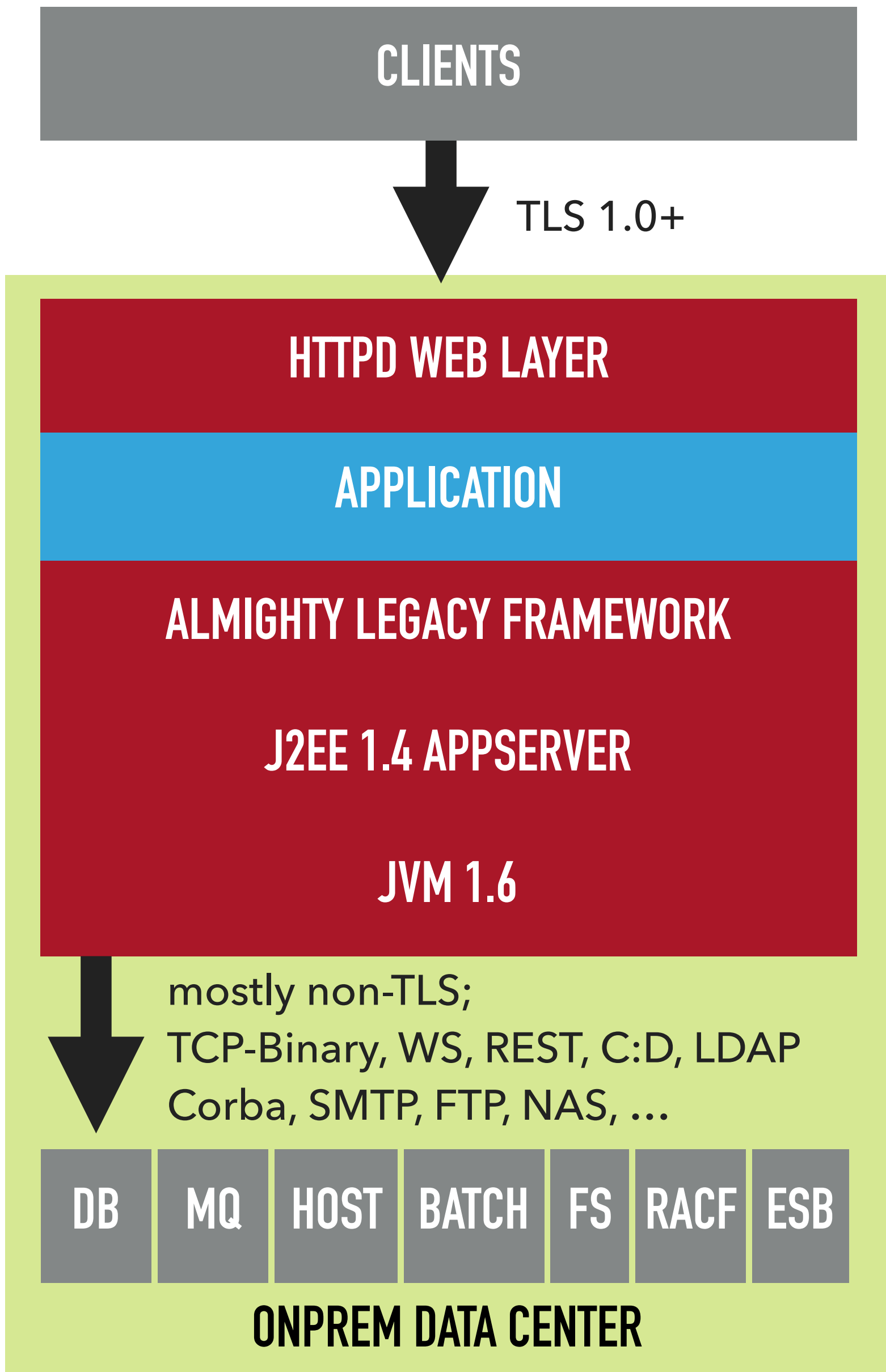
MORE MODERN APPLICATIONS (~200)



Re-architect to run on k8s on AWS

lift & shift VMs to AWS EC2





Can we evolve existing enterprise applications into the cloud with reasonable effort?



- Monolithic Deployment
- Traditional Infrastructure

CLOUD ALIEN



- Containerization
- 12-Factor App Principles

CLOUD FRIENDLY



- Microservices
- Cloud-native Apps

CLOUD NATIVE

Can we evolve existing enterprise applications into the cloud with reasonable effort?

Put the monolith into a container

... and enhance the application according to the 12 factors



- Monolithic Deployment
- Traditional Infrastructure

Sweetspot
time and value (security, opex)



- Containerization
- 12-Factor App Principles

CLOUD FRIENDLY

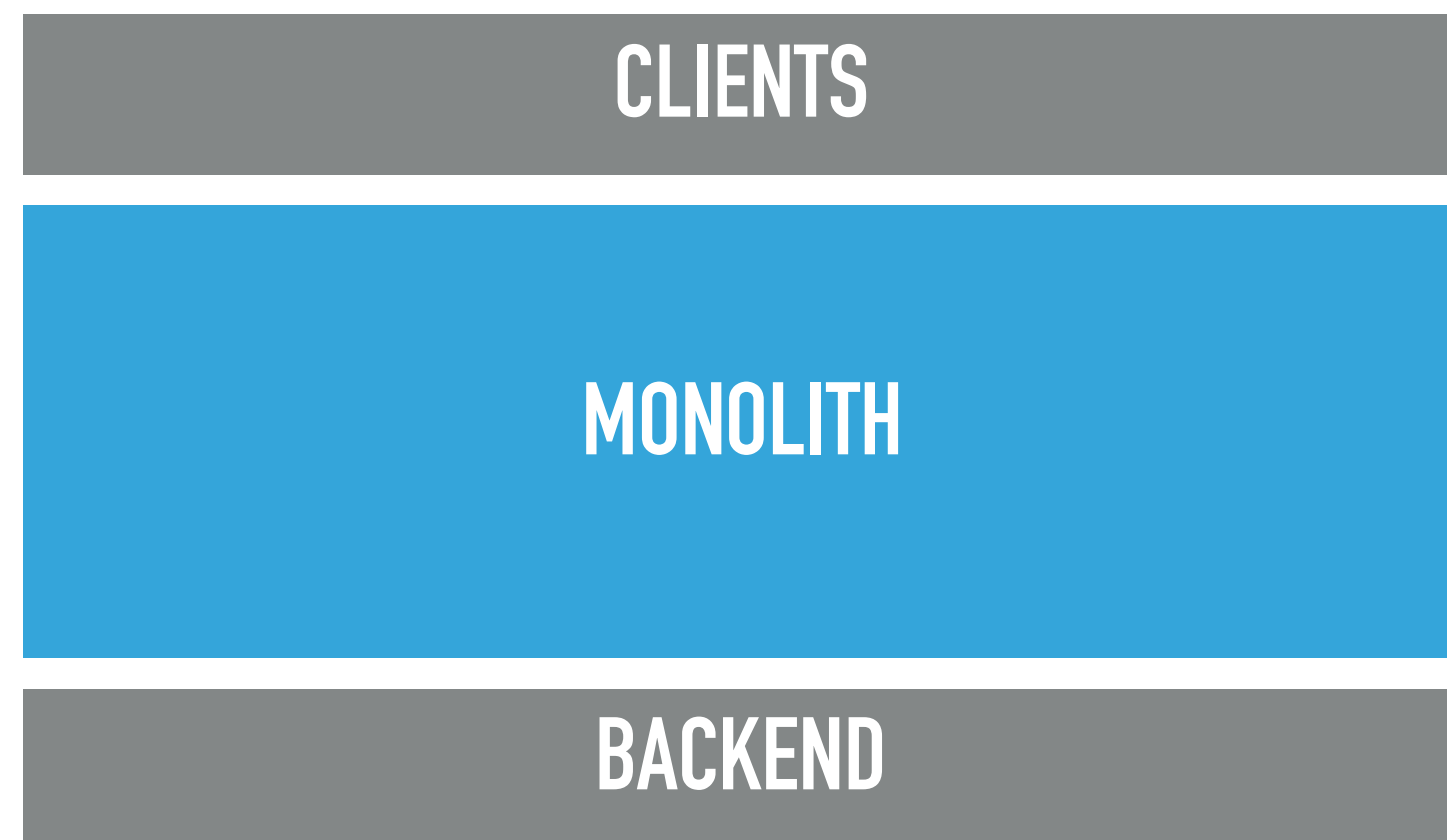


- Microservices
- Cloud-native Apps

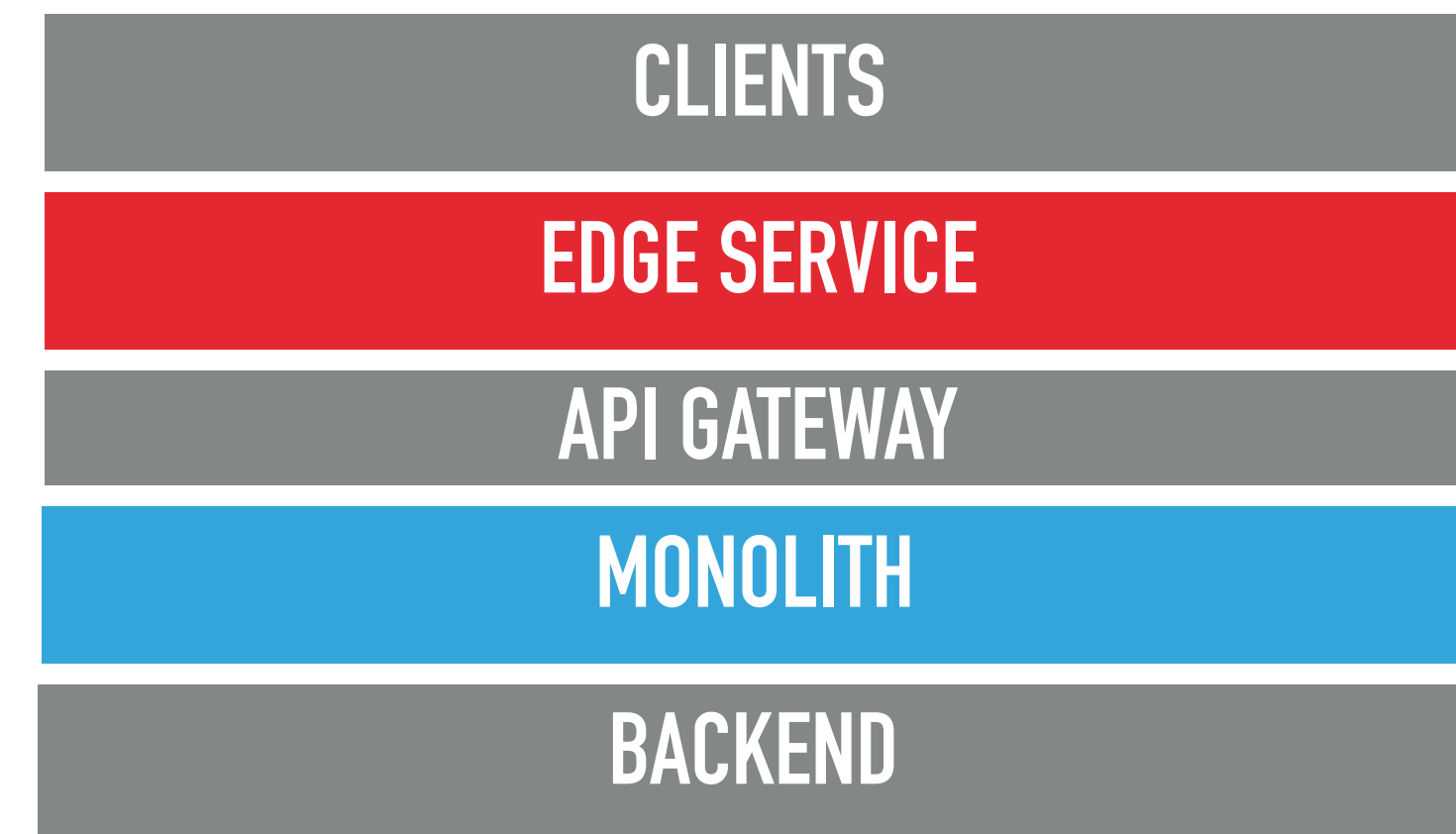
A dark, almost black interior space with a narrow vertical opening in the wall. Through this opening, a bright, clear sky is visible above a city skyline. The city features several prominent skyscrapers and a dense cluster of smaller buildings. The lighting is dramatic, with the interior being very dark and the exterior scene through the opening being brightly lit.

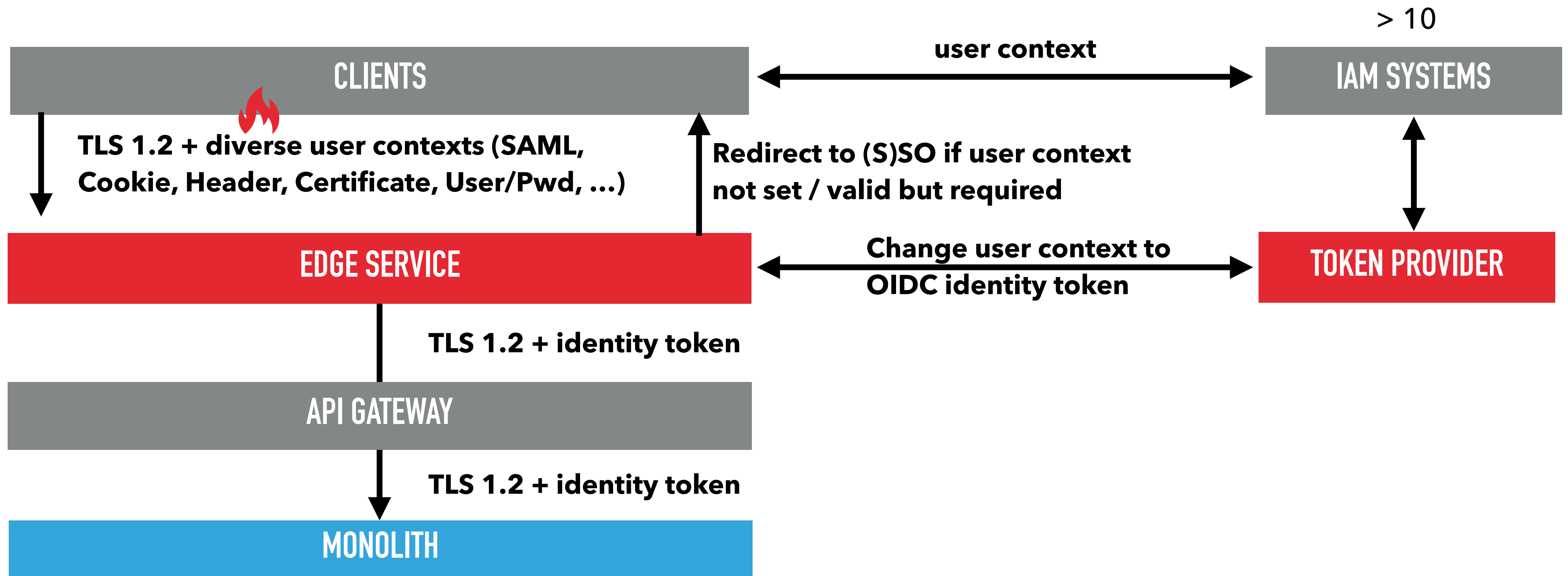
Edge Service to the Rescue

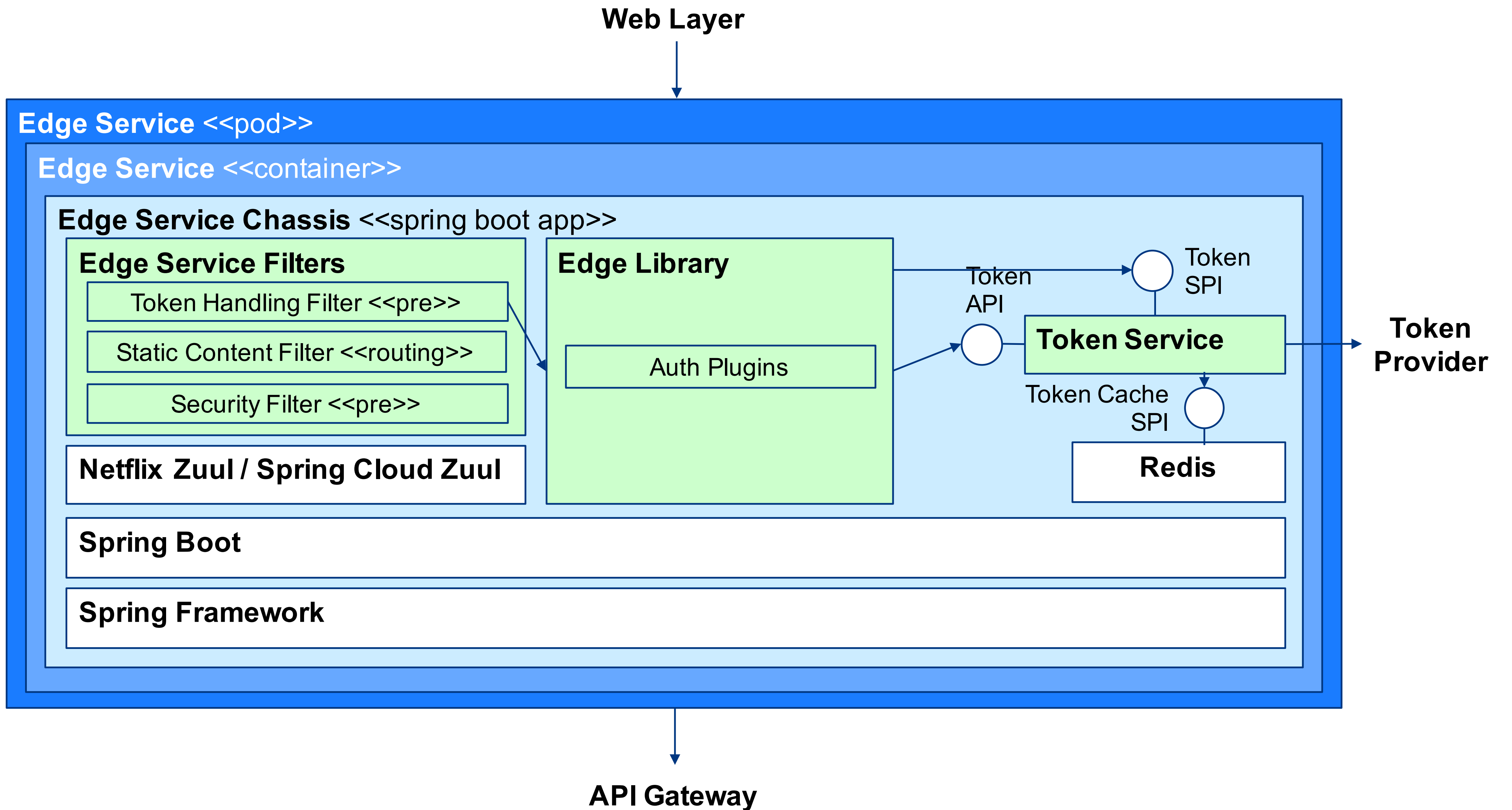
BEFORE



AFTER





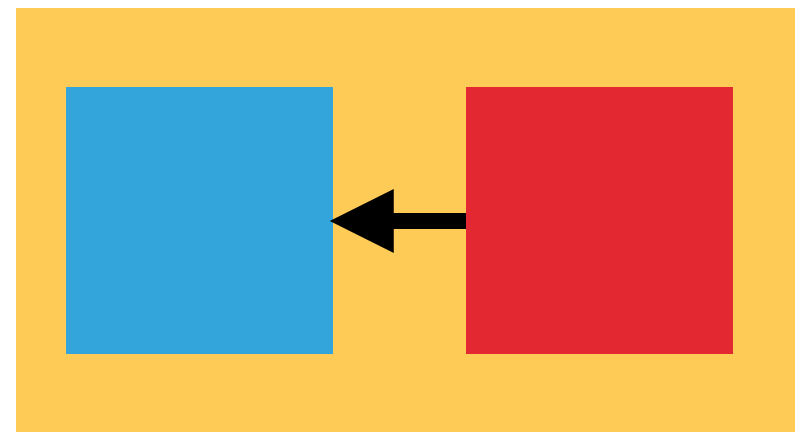




Sidecars to the Rescue

Container Patterns Applied

Sidecar: Enhance container behaviour



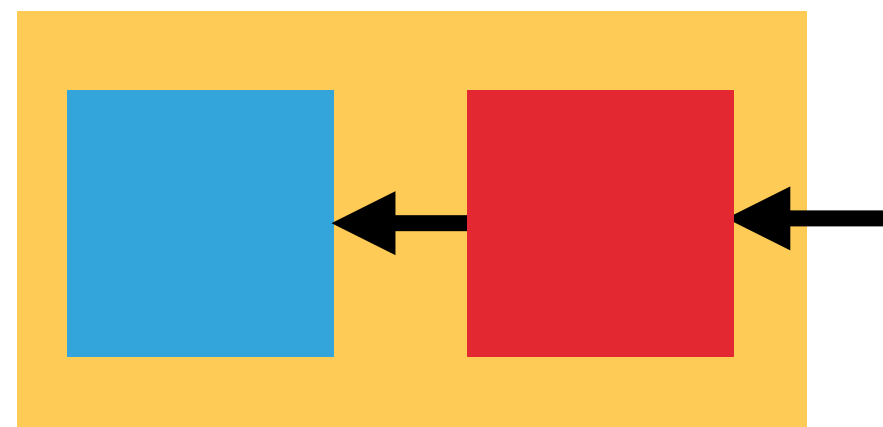
- Log extraction / re-formatting (fluentd)
- Scheduling (Quartz)

Ambassador: Proxy communication

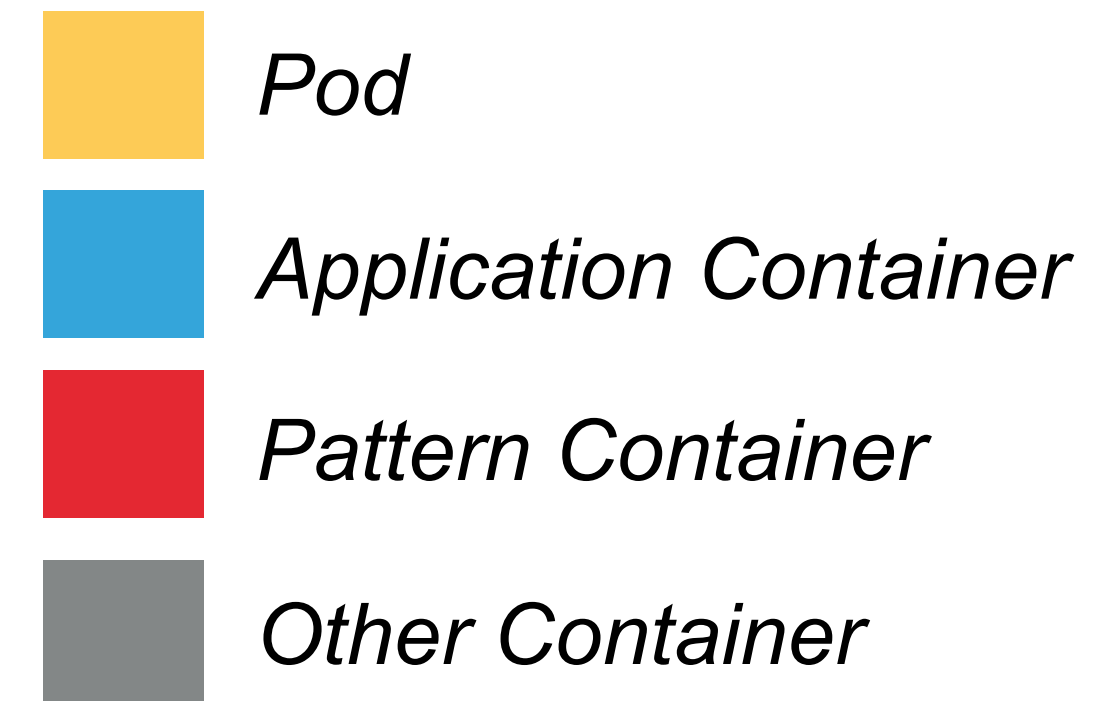


- TLS tunnel (Stunnel, ghostunnel)
- Circuit Breaking (linkerd)
- Request monitoring (linkerd)

Adapter: Provide standardized interface



- Configuration (ConfigMaps & Secrets to files)



Kubernetes Constraints

Initially we thought we'll run into k8s restrictions on our infrastructure like:

- ▶ No support for multicast
- ▶ No RWX PVC available

We did. But cutting these application requirements lead to a better architecture in each and every case.



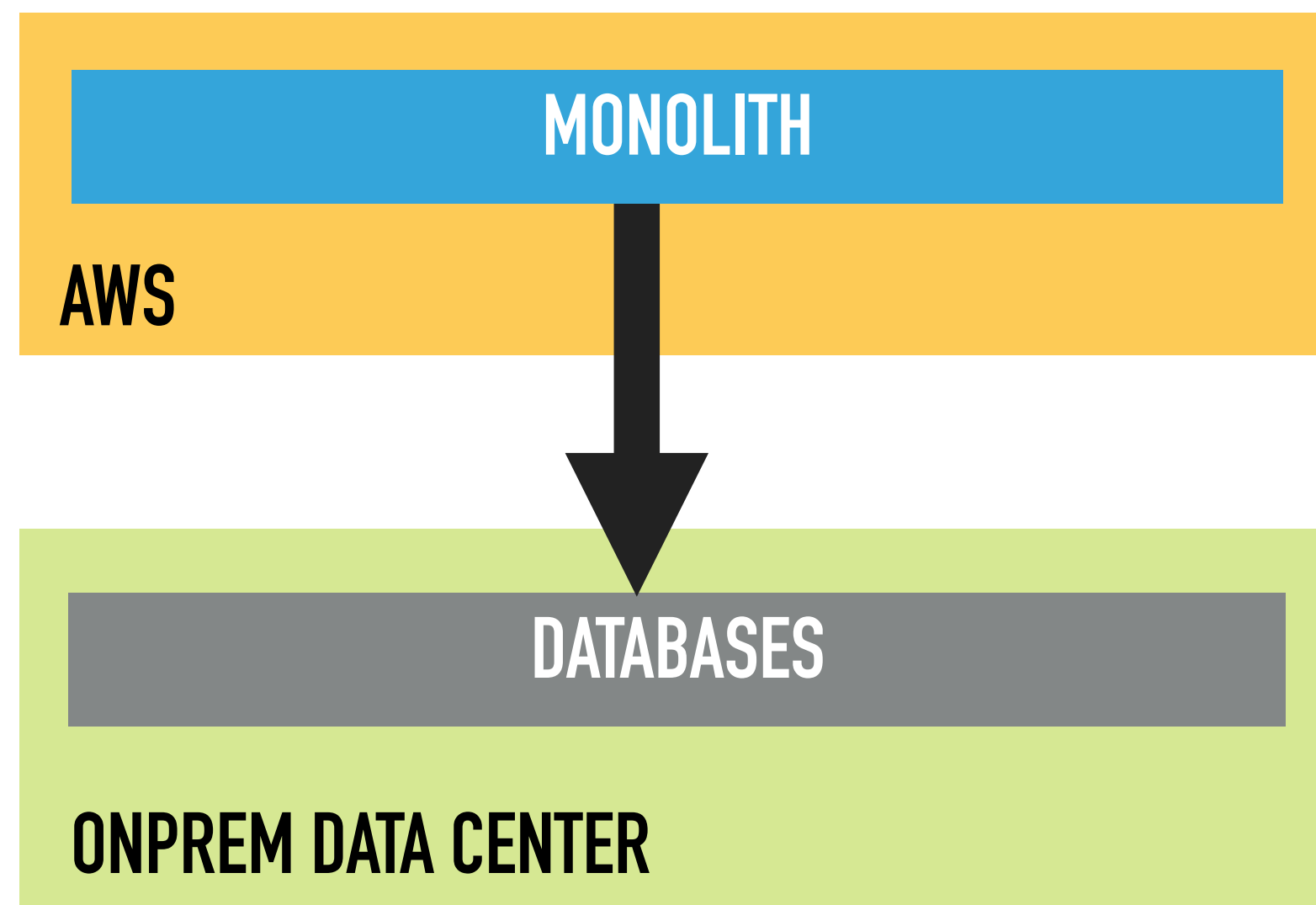
THE BAD





STATE

Databases



Activate TLS for all database connections

(low effort on application-side but separate project on the database side)

Databases stay in onprem data center

- Advantages:
 - onprem/cloud version of the application can run in parallel
 - privacy
- Disadvantages:
 - latency (no real problem)

Files

File persistence is very restricted to keep persistent state completely out of AWS (privacy).

- ▶ No file writes allowed into container
- ▶ No RWX PVC available
- ▶ Files with application data written to PVC must be deleted after 15mins
- ▶ No NAS mounts from onprem data centers into containers allowed

Migration tasks for affected applications

- ▶ Store files as BLOB in database or use FTPS

Session State

90% OF THE APPLICATIONS HAVE SESSION STATE
100% OF THE APPLICATIONS HAVE MORE THAN 1 INSTANCE

Session Stickyness: not within the cloud!

Session Persistence:

- ▶ Within existing application database: performance impact too high for most
- ▶ Redis: no transport encryption out-of-the-box and separate infrastructure required

Session Synchronization:

- ▶ Application server: nope, no dynamic peer lookup within k8s
- ▶ In-memory data grid: Hazelcast. Nope. \$\$\$ for TLS.
- ▶ In-memory data grid: Apache Ignite. Done.
 - ▶ Embedded within application or standalone in a separate container.
 - ▶ Little bit cumbersome but working k8s peer lookup
 - ▶ Runs into JIT bug on IBM JVM (some methods have to be excluded from JIT)

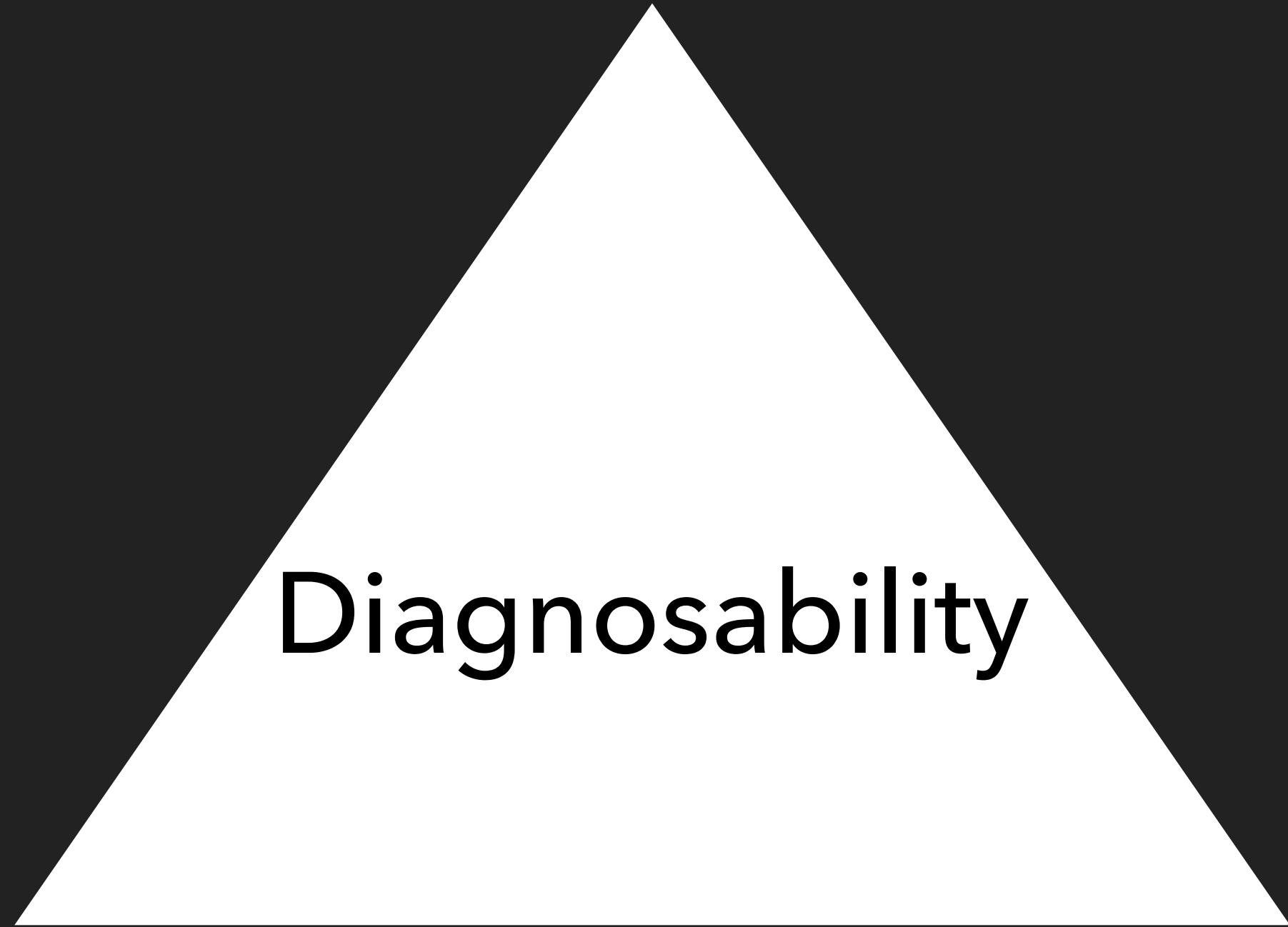


Metrics

Diagnosability

Traces

Events / Logs





Prometheus

Metrics

Diagnosability

Traces



OPENTRACING

Events / Logs



fluentd



Prometheus

Metrics



- High effort to instrument for valuable insights
- Scalability unclear for hundreds of applications
- Applications have no time to run their own Prometheus instance

Diagnosability



- Scalability unclear (a lot of events lost)
- Applications have no time to run their own EFK instance
- Non-standardized log format requires custom log rewrite adapter but no fluentd DaemonSet



- Scalability unclear for hundreds of applications (Jaeger & ZipKin)
- Applications have no time to run their own instance

Traces



Events / Logs



Metrics



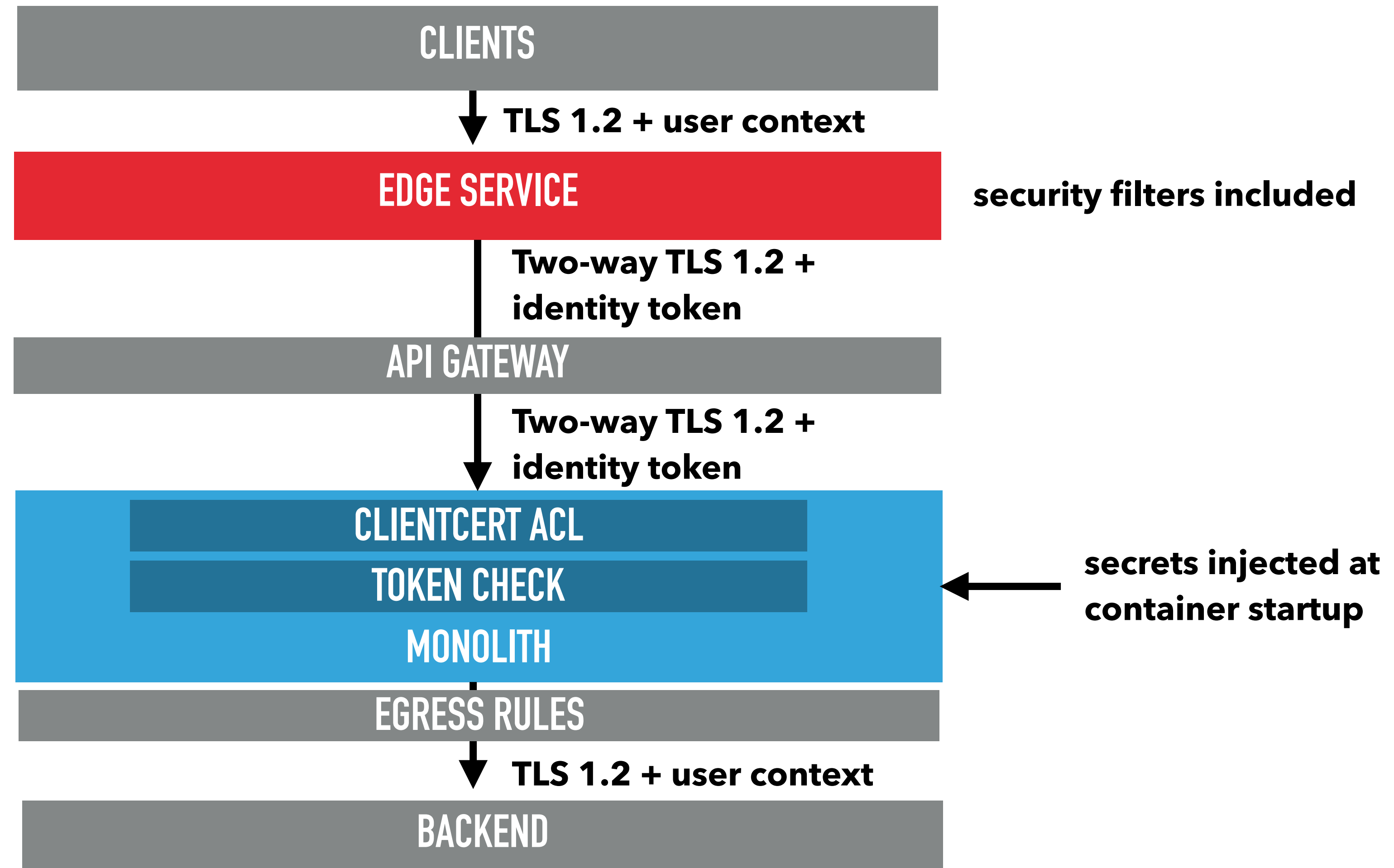
Traces

Events / Logs



need a SecDevOps?

We Came Far



The Bad: Certificate Management

VAR 1: CLOUD NATIVE STYLE CERTIFICATE MANAGEMENT
(SPIFFE-BASED, AT SERVICE MESH OR APPLICATION LEVEL)

VAR 2: REPLACE BY POLICIES (AT NETWORKING LEVEL)

e.g.



e.g.





THE UGLY

```
String hostRequest = new HostRequest().hostusPokus(message);
```

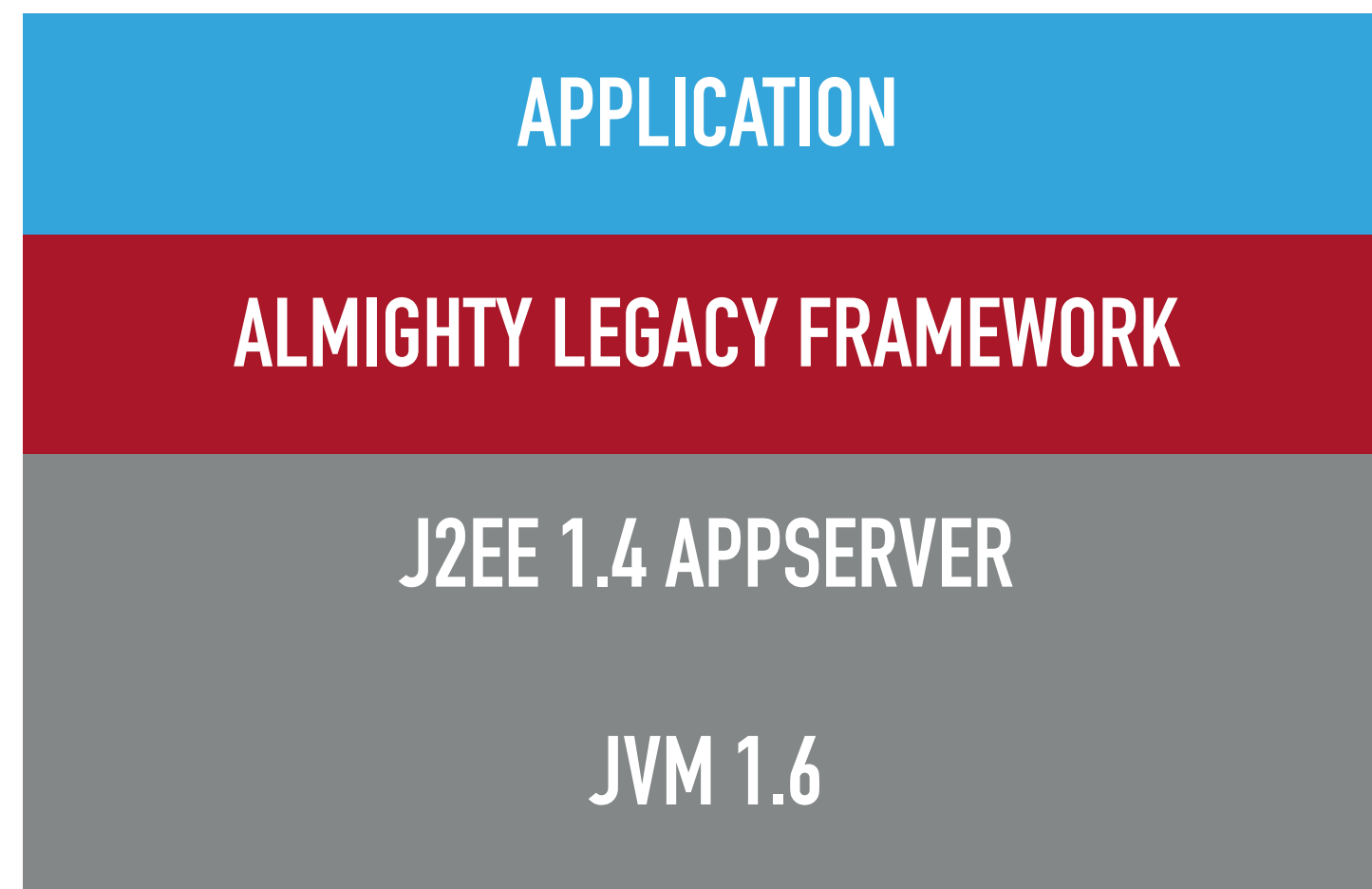
CLOUD ENABLING CLOUD ALIENS





TOXIC
TECH

The Almighty Legacy Framework



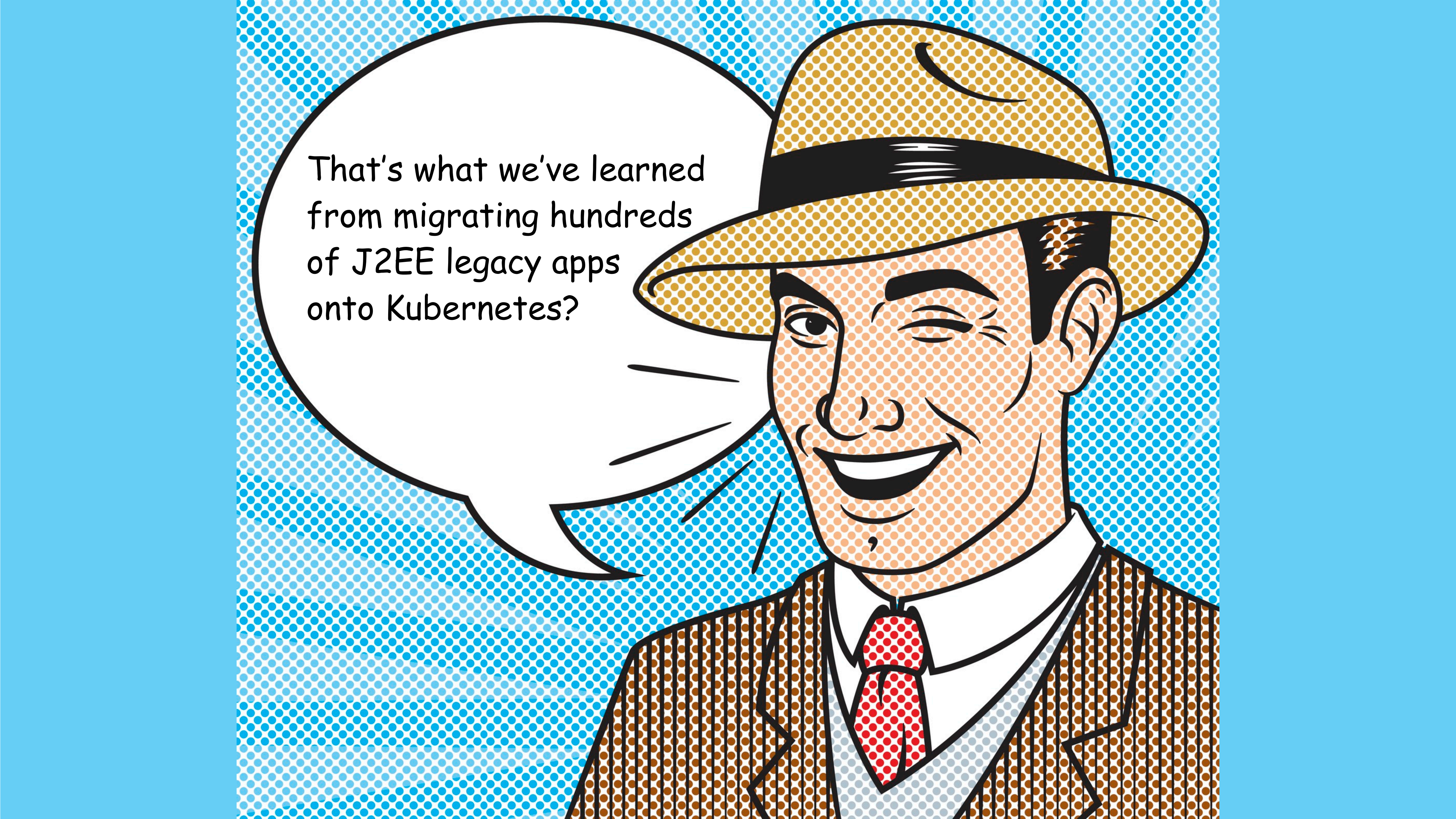
- “worry-free package framework” from the early 2000s with about 500kLOC and 0% test coverage
- Migration tasks
 - from J2EE 1.4 to JEE 7 and Java 6 to 8
 - add identity token check and token relay
 - modify session handling (synchronization)
 - modify logging (to STDOUT)
 - modify configuration (overwrite from ConfigMap)
 - enforce TLS 1.2
 - place circuit breakers
 - predefined liveness and readiness probes
- Strategies:
 - the hard way: migrate manually and increase coverage
 - decorate with ambassadors, sidekicks and adapters
 - do not migrate parts and replace that API within the applications

~ 100 systems live on target
by end of this year (after 8mo)

~ 200 systems live on target
by end of first quarter 2018

other ~200 systems migrated
by end of first quarter 2018 via
virtual lift & shift. They will be
migrated onto Kubernetes
afterwards

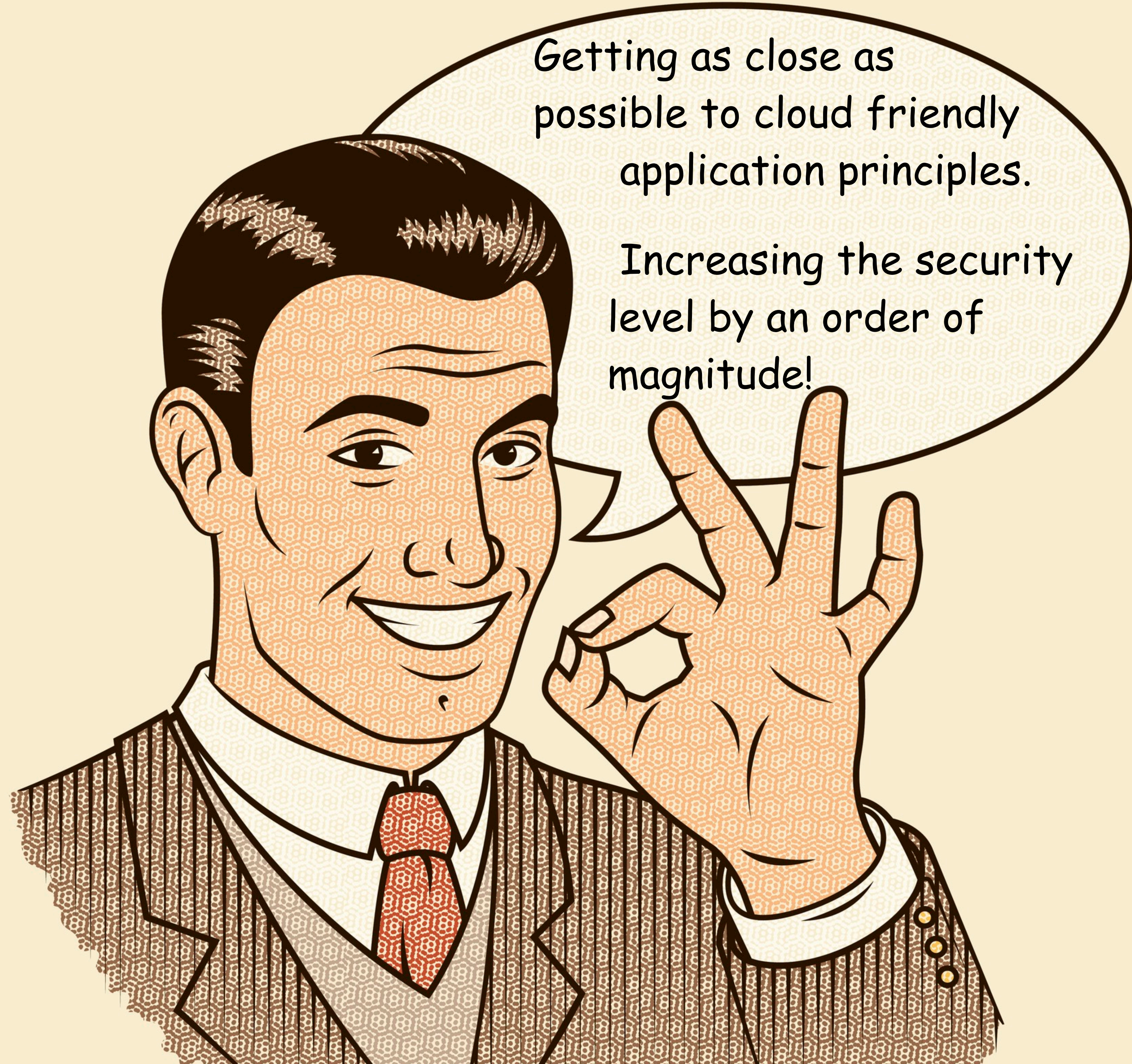




That's what we've learned
from migrating hundreds
of J2EE legacy apps
onto Kubernetes?

No stupid
"as-is into
containers"
approach!





Getting as close as possible to cloud friendly application principles.

Increasing the security level by an order of magnitude!

Q A W A R E

proud member of the CNCF

Q -> A



Thank you!



josef.adersberger@qaware.de



[@adersberger](https://twitter.com/adersberger)

[TWITTER.COM/QAWARE](https://twitter.com/qaware) – [SLIDESHARE.NET/QAWARE](https://slideshare.net/qaware)

BONUS SLIDES

Industrialization



DOZENS OF MIGRATION PROJECTS RUNNING IN PARALLEL (UP TO ~80)

- ▶ Training sessions
- ▶ Support sessions

SUPPORT TEAM (~10 FTE)

- ▶ Guidance (cookbook, sample application)
- ▶ Development environment (SEU-as-Code)
- ▶ Standard images
- ▶ Automated Refactorings (RefactoBot)
- ▶ Pre-migrated frameworks
- ▶ Solutions: Edge service, ambassadors

INDUSTRIALIZATION TEAM (~6 FTE)

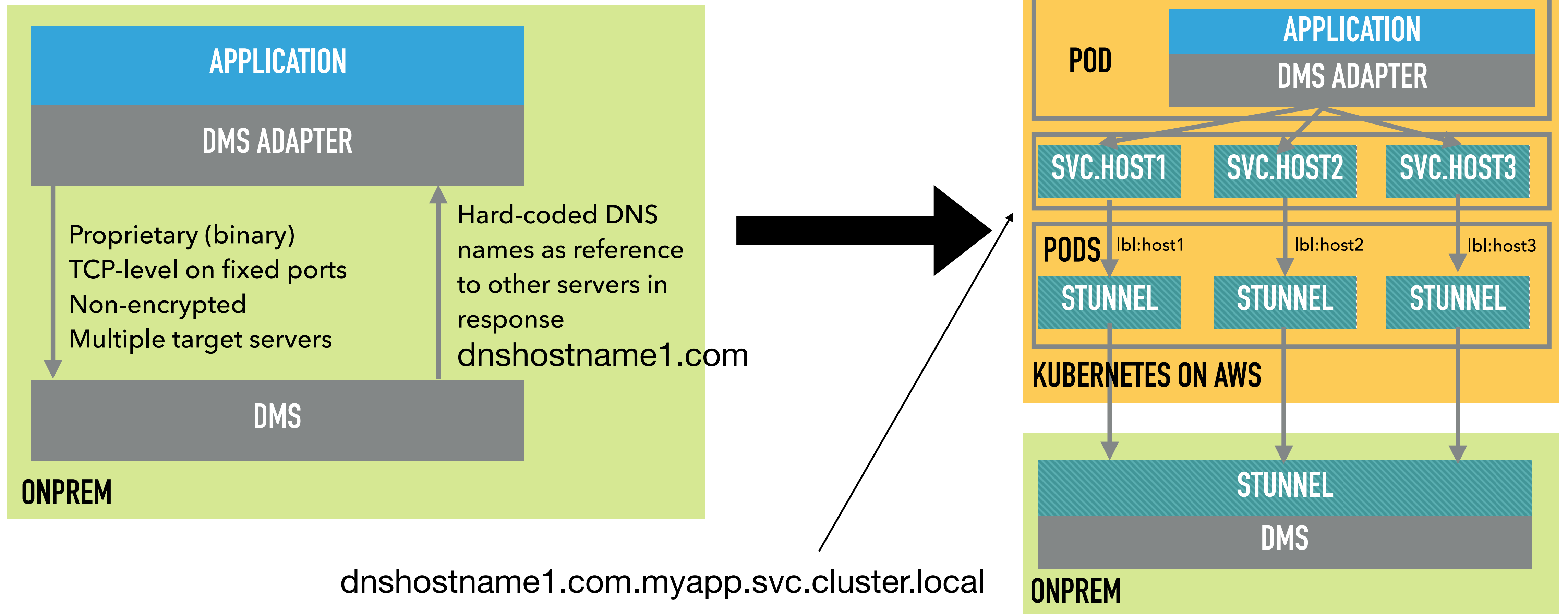
- ▶ Application blueprint (target architecture, migration rules)
- ▶ Migration database

ARCHITECTURE TEAM (~2 FTE)



NO SPEAK
CLOUD

DMS System

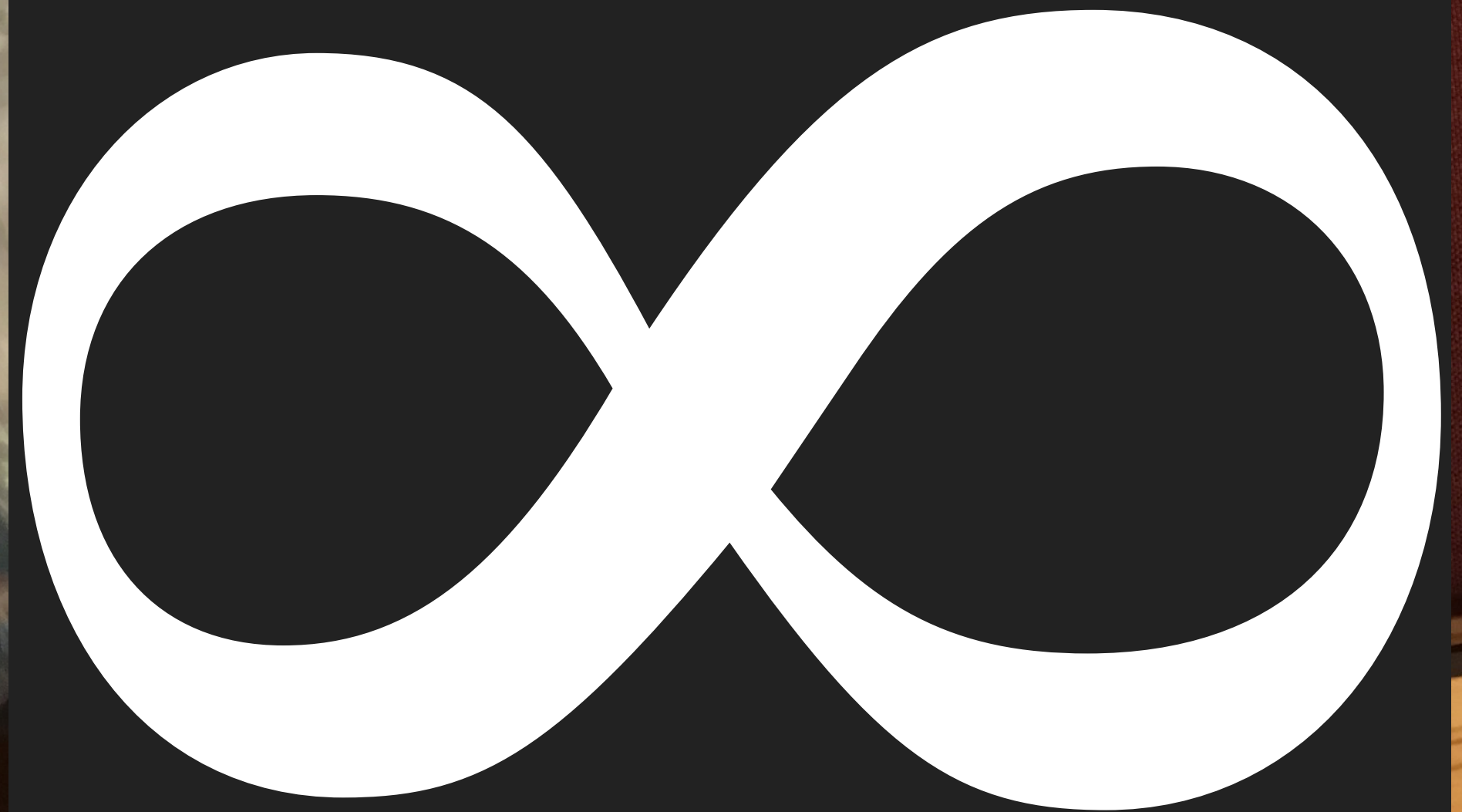




Continuous
Delivery

PLAN

RUN

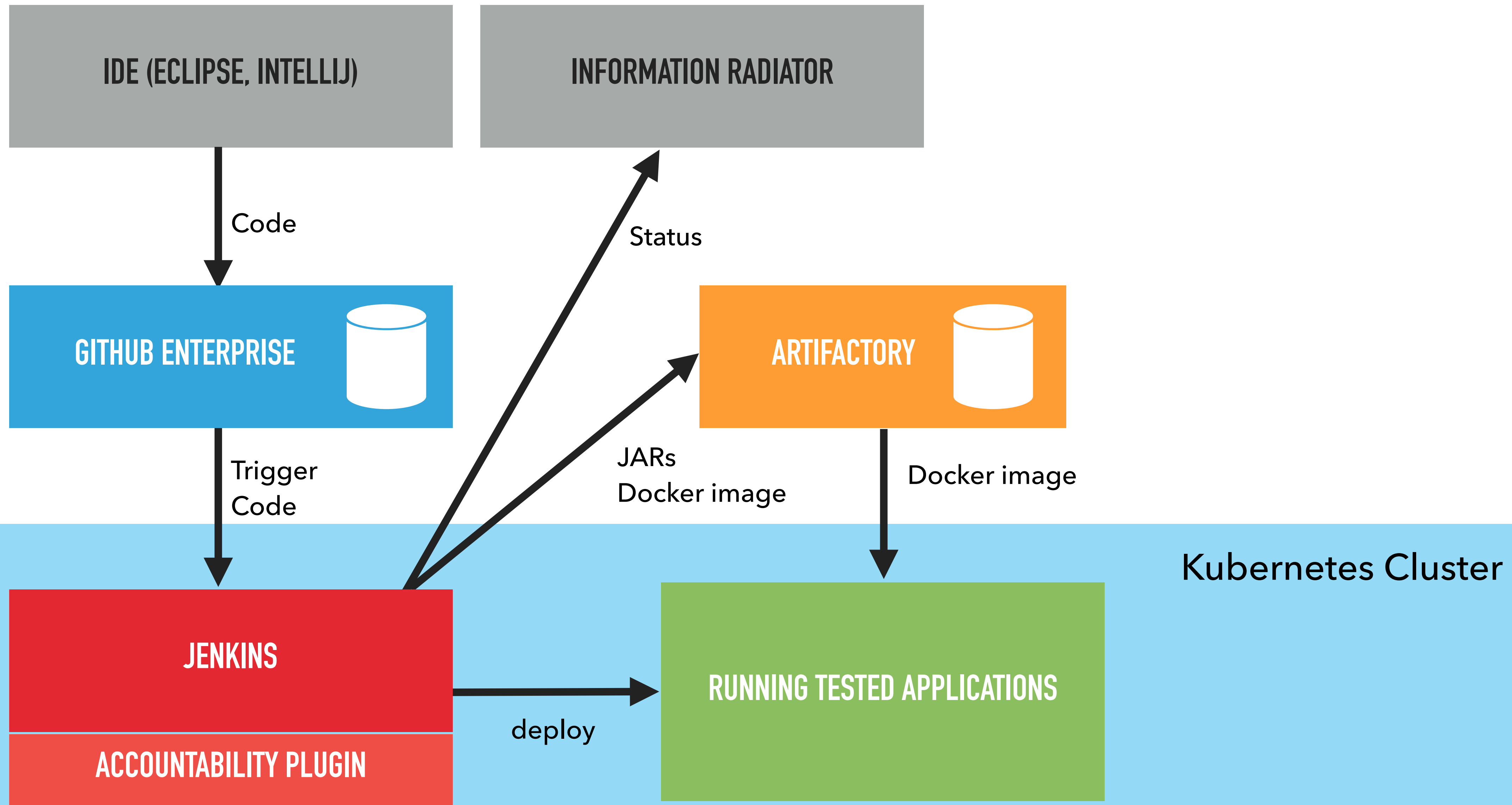


BUILD

GRASP



Continuous
Feedback



Quality Gate Failed

since 6-idpmerge-SNAPSHOT
406 New Vulnerabilities
> 0

since 6-idpmerge-SNAPSHOT
369 New Bugs
> 0

since 6-idpmerge-SNAPSHOT
33.3% Technical Debt Ratio on New Code
> 5.0%

since 6-idpmerge-SNAPSHOT
31.8% Coverage on New Code
< 80.0%

Quellcode
Quellcode (Entwickler)

Schlüssel

Quality Gate
(Standard) SonarQube way

Quality Profiles
(Groovy) Sonar way
(Java) Sonar way
(XML) Sonar way

Ereignisse All

Version: 6-SNAPSHOT
28. November 2017

Ereignisse: Red (was Green) ?
4. Oktober 2017

Ereignisse: Green (was Red)
29. September 2017

Version: 6-idpmerge-SNAPSHOT
29. September 2017

Profil: Use 'Sonar way' (Groovy)
24. August 2017

Show All

Bugs & Vulnerabilities

Leak Period: since 6-idpmerge-SNAPSHOT
started vor 2 Monaten

398^E
Bugs

412^B
Vulnerabilities

369
New Bugs

406
New Vulnerabilities

Code Smells

11k^A
Code Smells
started vor 5 Monaten

329T
Debt

10k
New Code Smells

312T
New Debt

Abdeckung

17.6%
Coverage

135
Unit Tests

42.4%
Coverage on
221 New Lines of Code

Duplications

