okta

The World's Identity Company

Identity at Scale

How Okta Uses Postgres

Norberto Leite

Principal Engineer@Okta

#pgconfeu

Safe harbor

Ola, I'm Norberto!

- Principal Engineer @ Okta
- Databases, that's my thing
- Sometimes, I put things on a scale

norberto.leite@okta.com @nleite

Agenda

- Challenge of Scaling Identity Management
- Operational Challenges
- Service Releases and Infrastructure Operations
- Database Management at Scale

Takeaways

- How complex/hard is Identity Management
- Things we’ve learned operating large fleets
- Stuff that we would like Postgres to have
- Unsolved challenges that we are working on

Okta?

Okta CIC = Auth0

CIAM
Customer Identity and
Access Management

Auth0 by Okta

(Authtoberfest);
2024

Join us this October as we celebrate Hacktoberfest 2024 by encouraging developers to contribute to the open-source community. Whether you're new to open source or an experienced contributor, this is your chance to give back, make a difference, and win some cool swag in the process.

authtoberfest.io

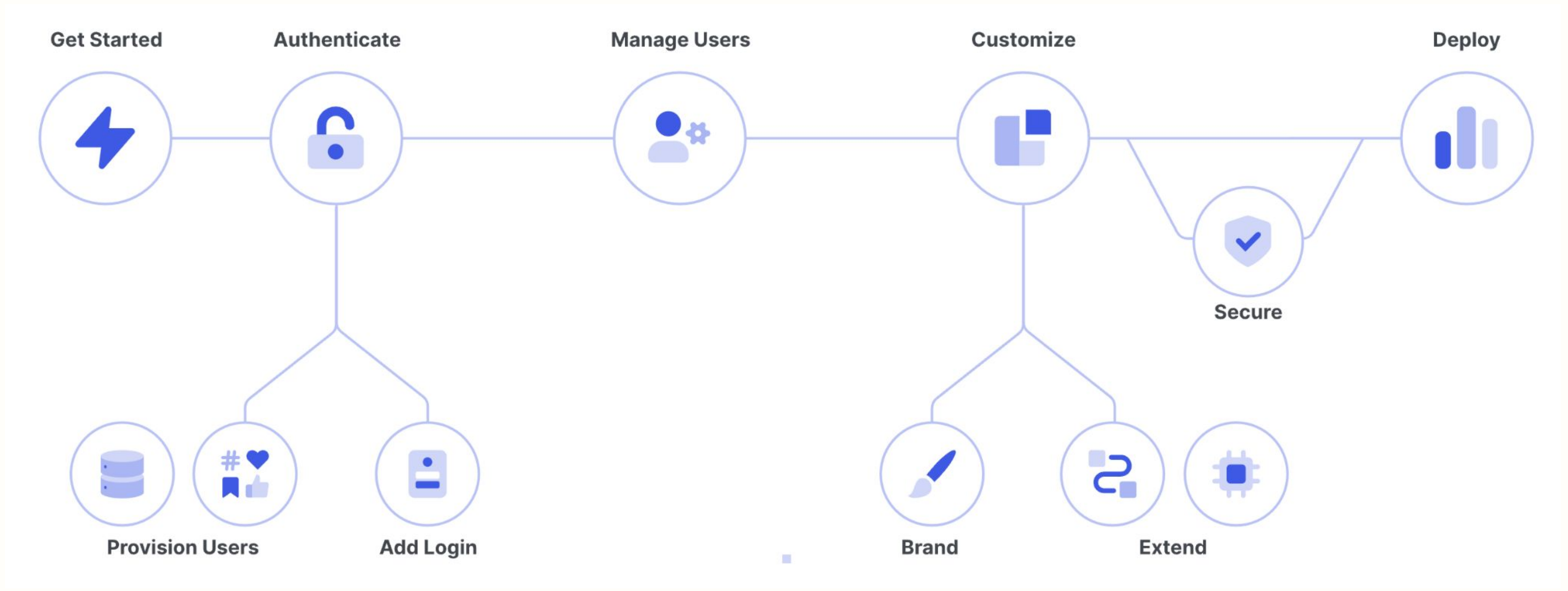
dev_day

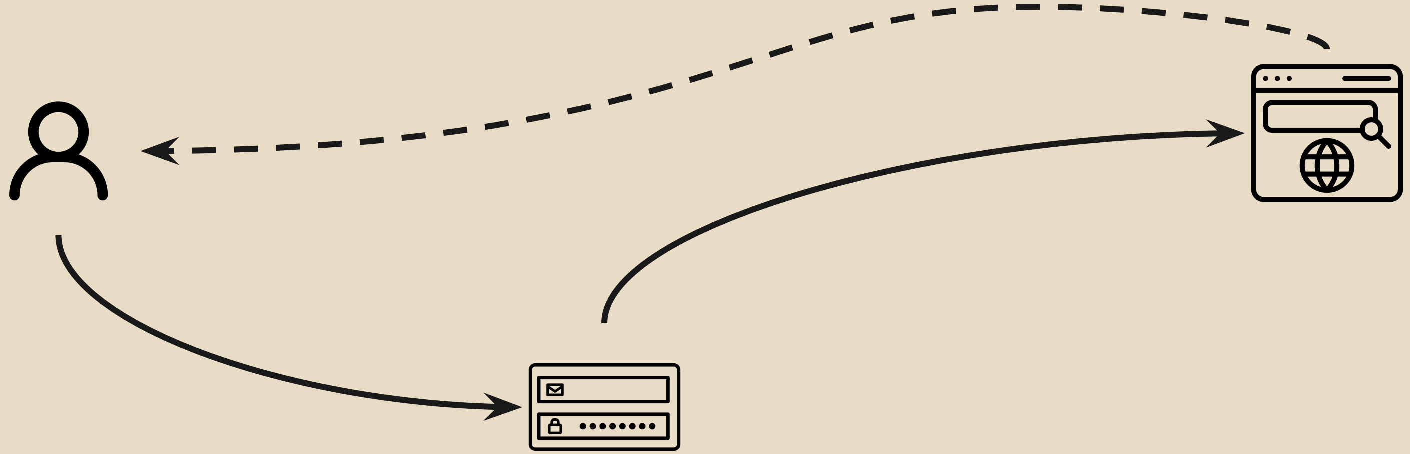
Auth0 by Okta

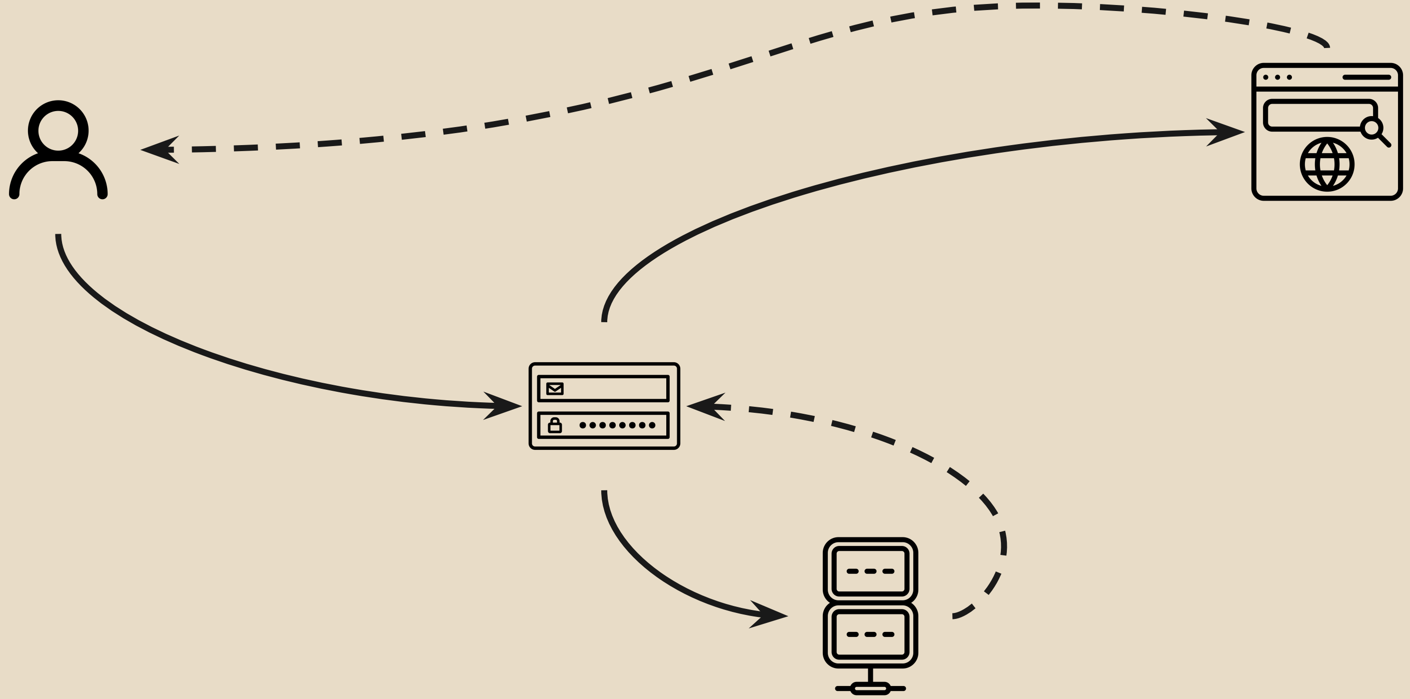
The Challenge of Scaling Identity Management (CIAM)

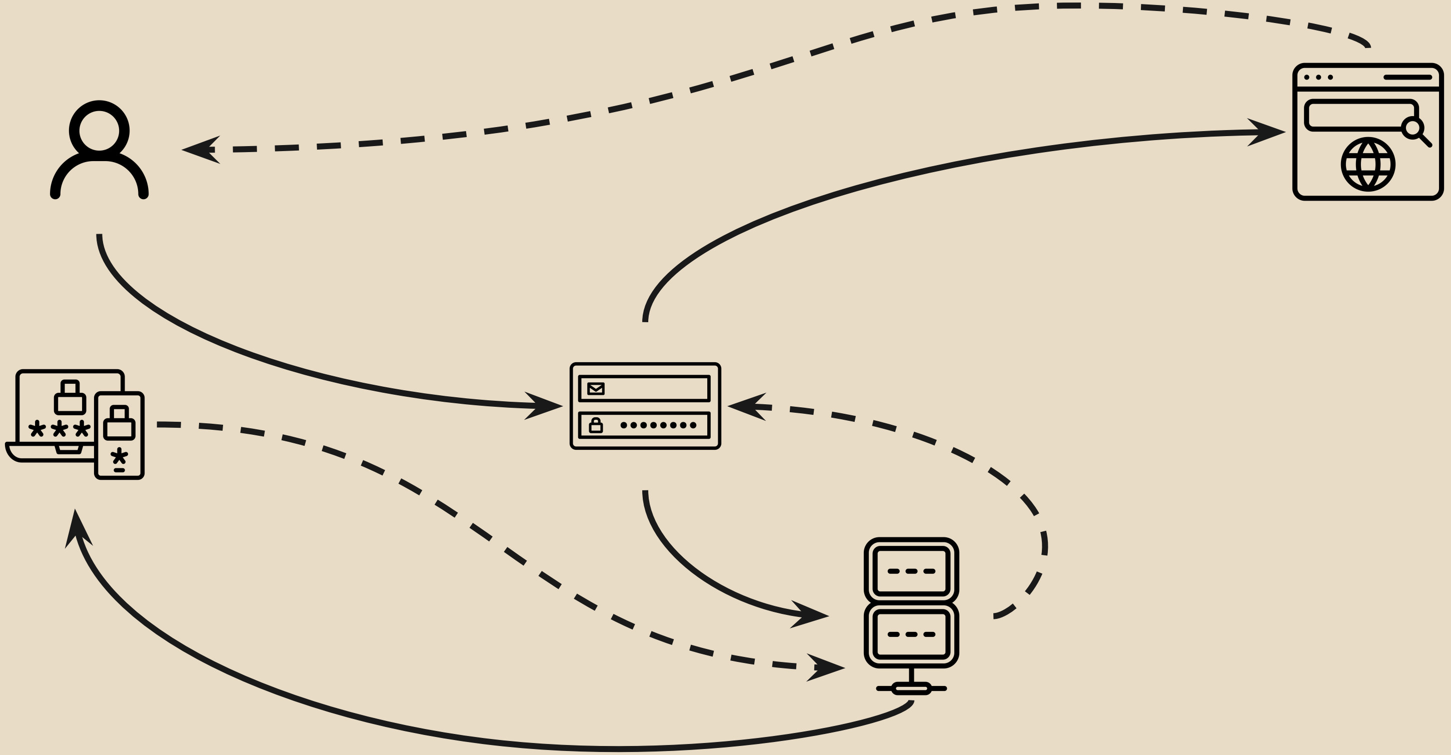
CIAM Platform Features

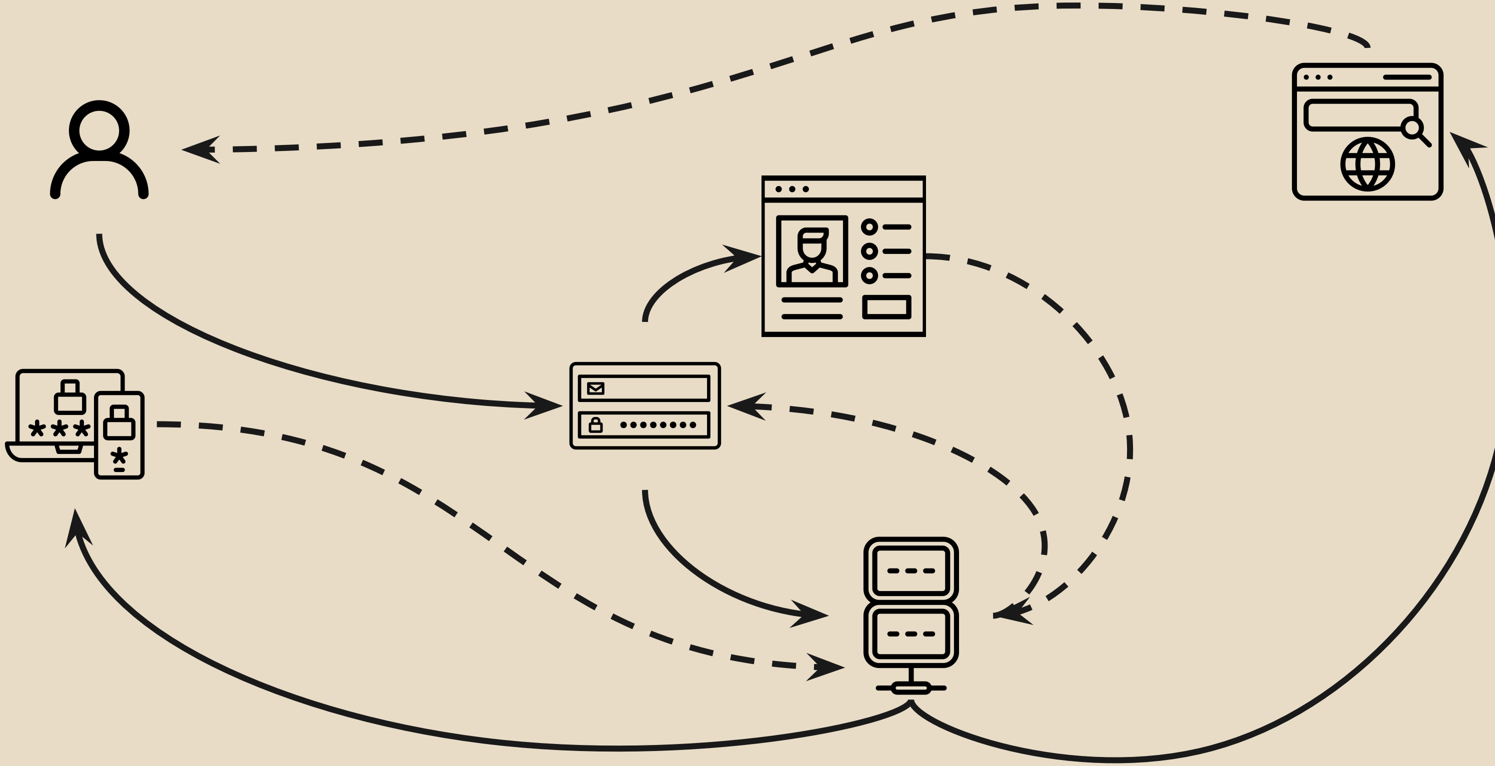
All the goodies

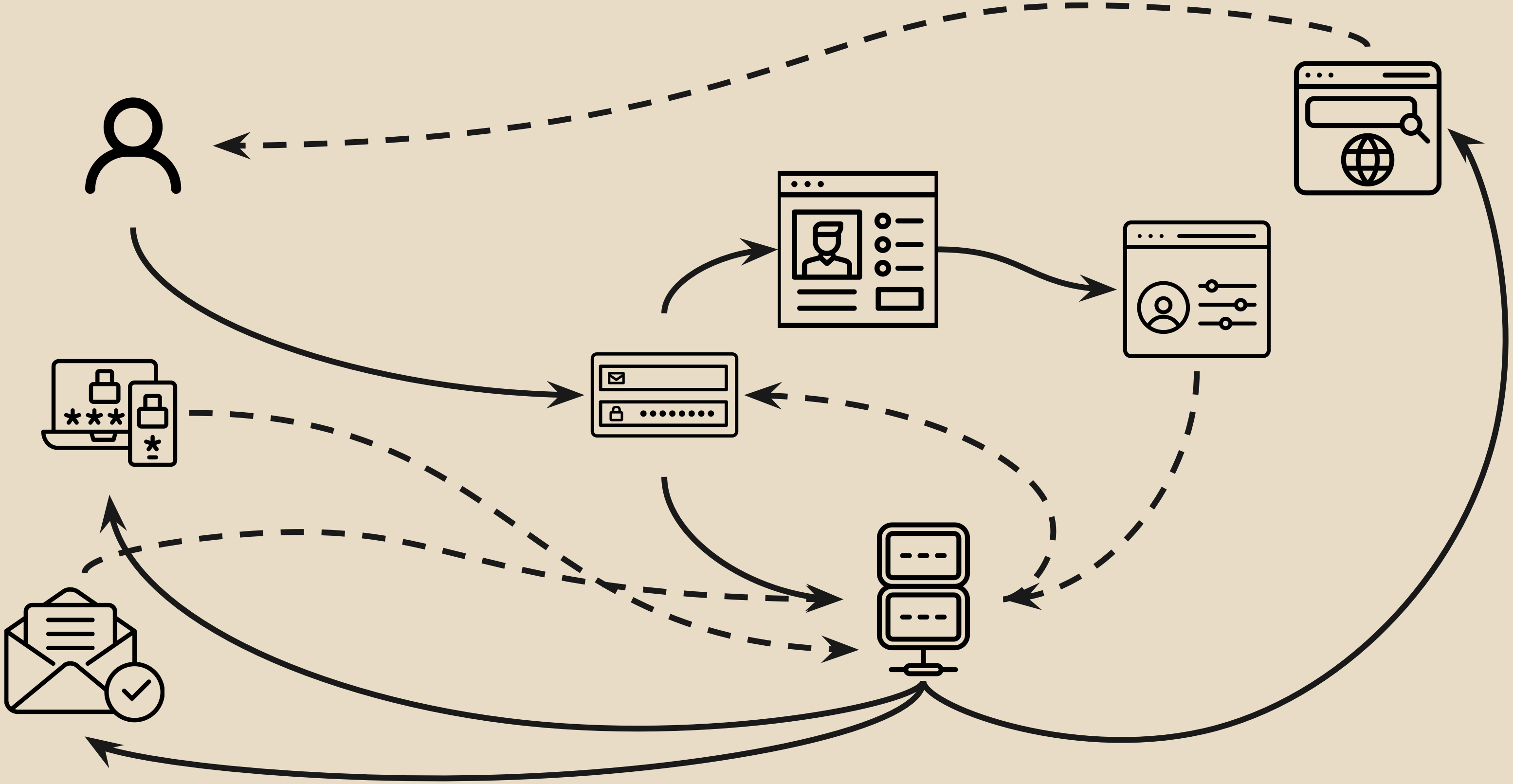


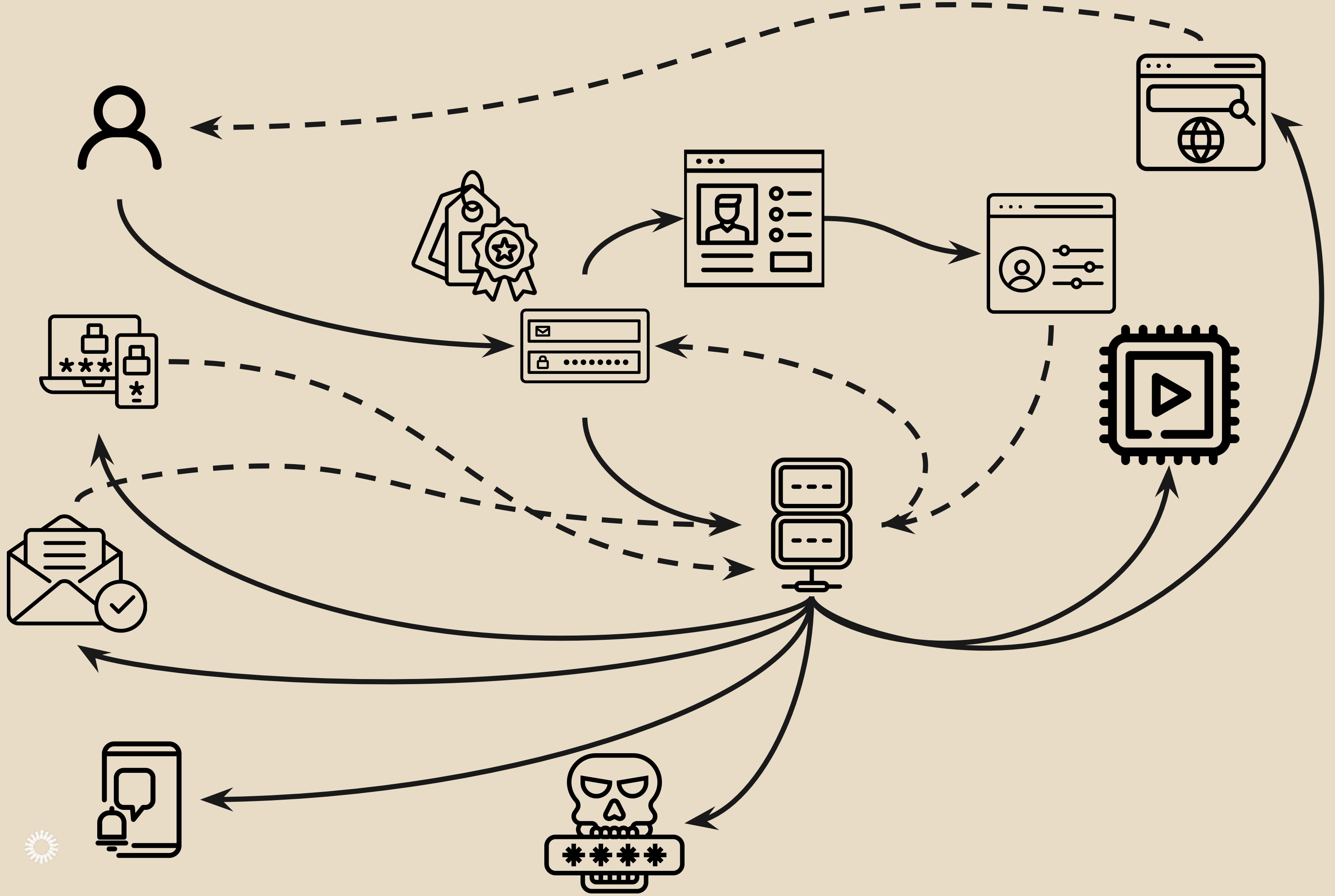


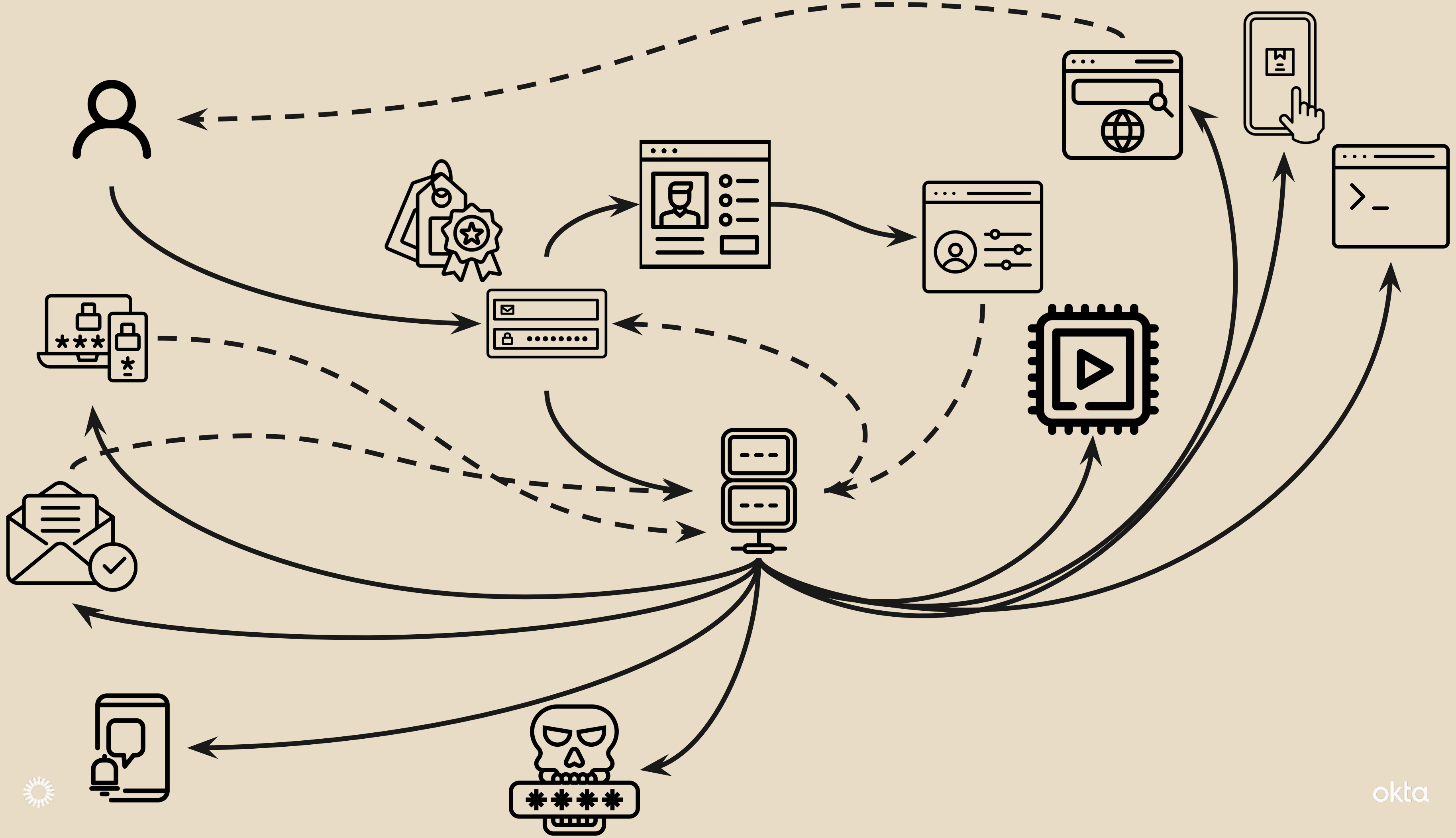


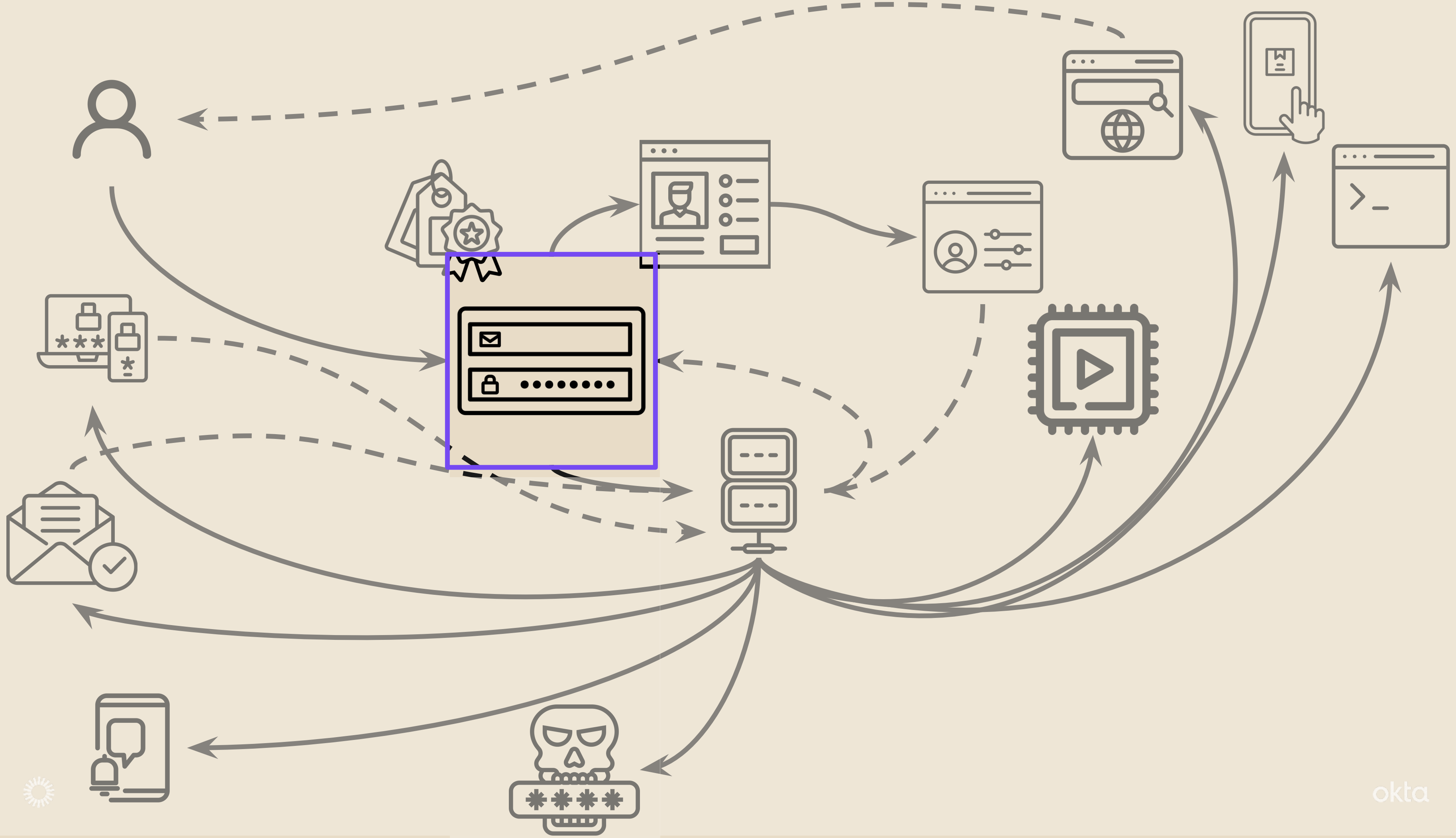


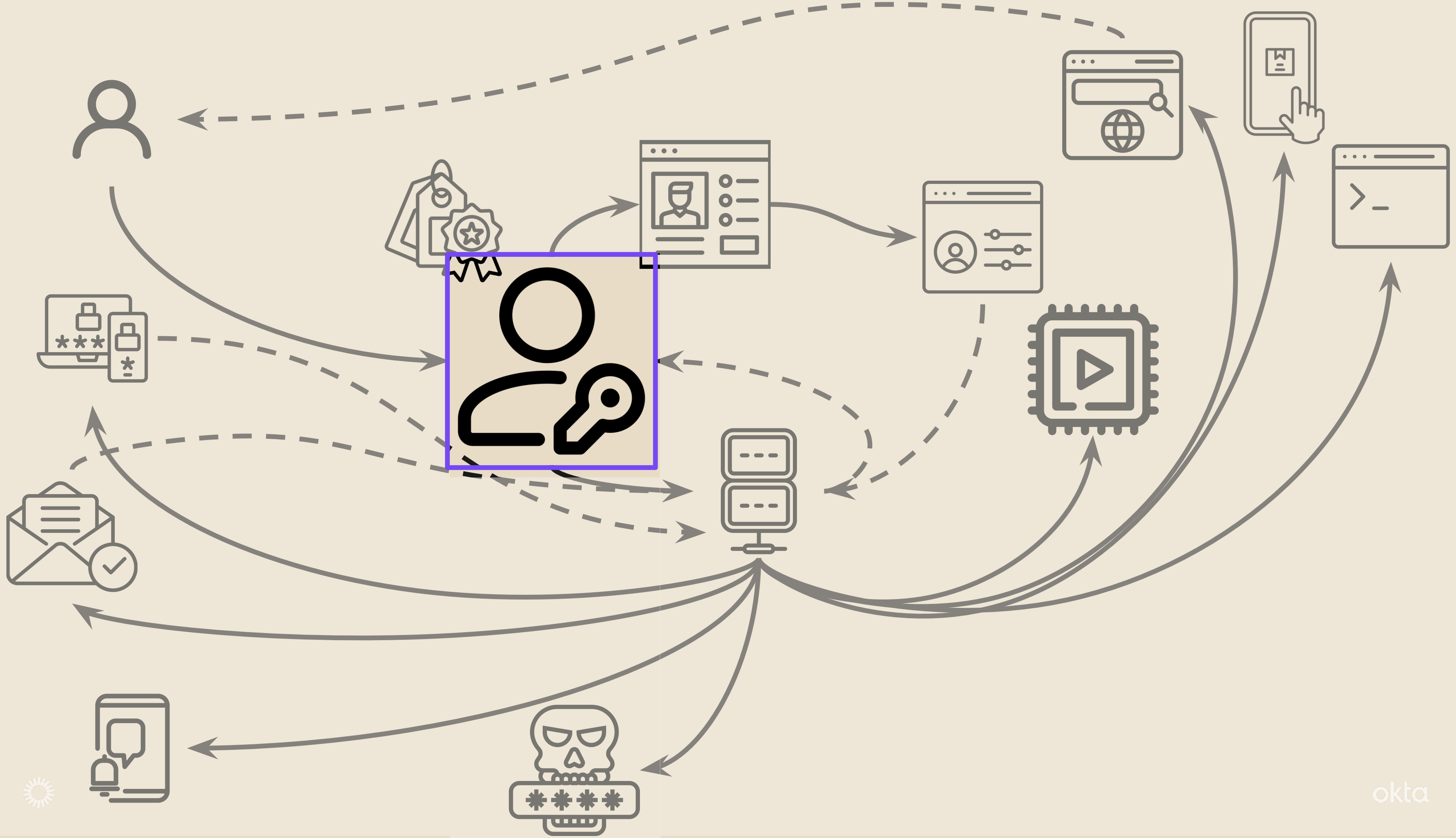


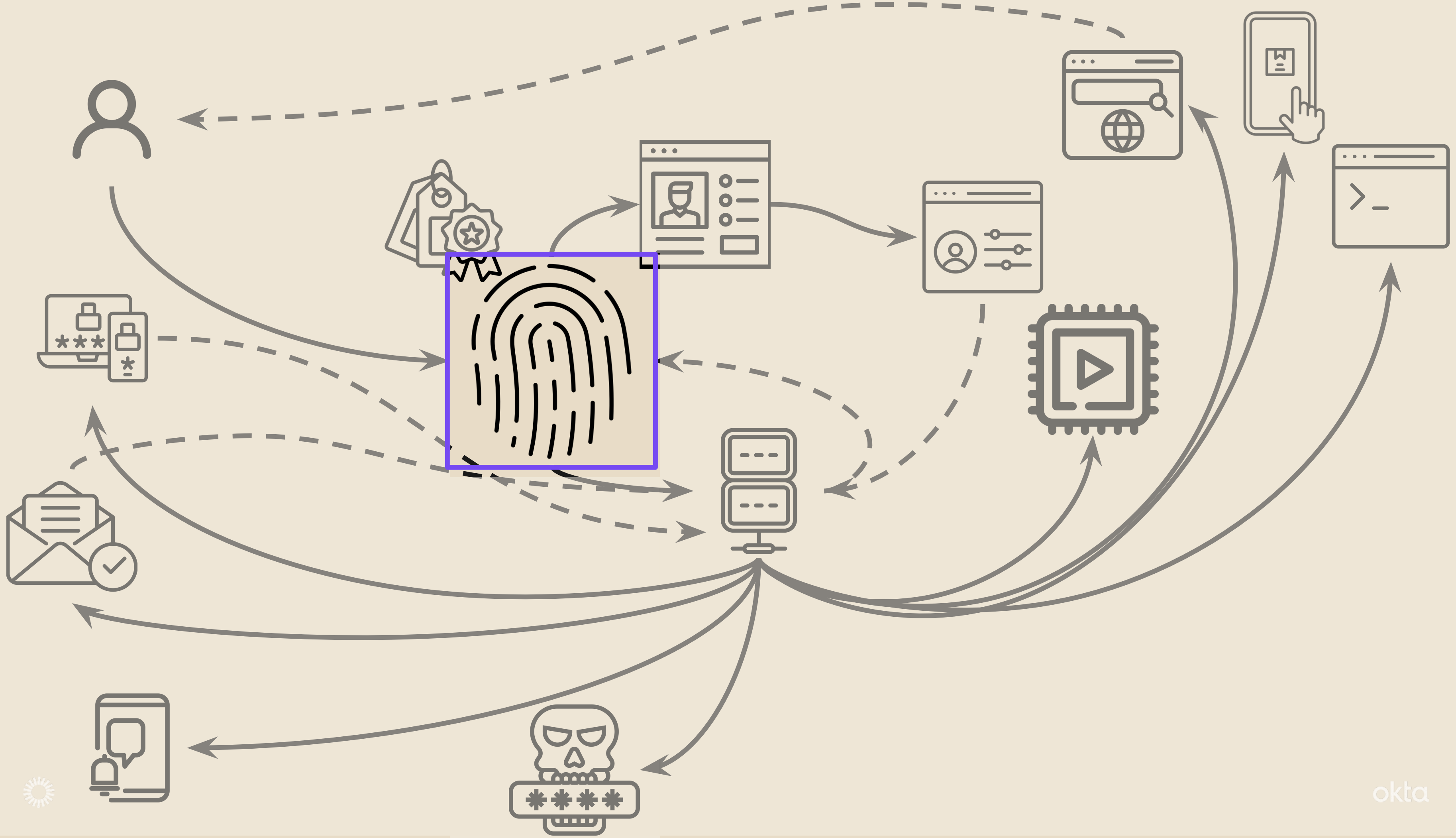


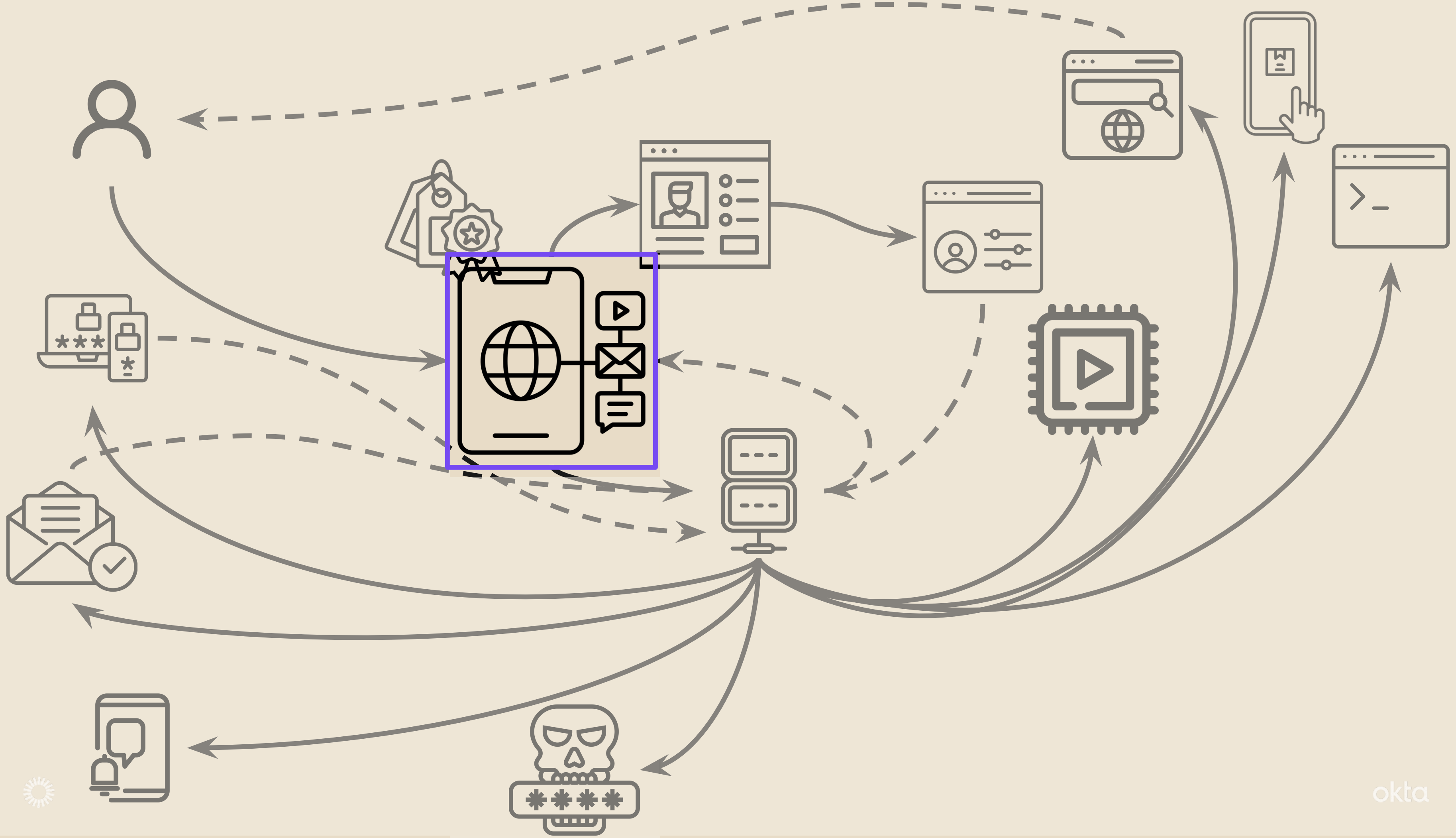


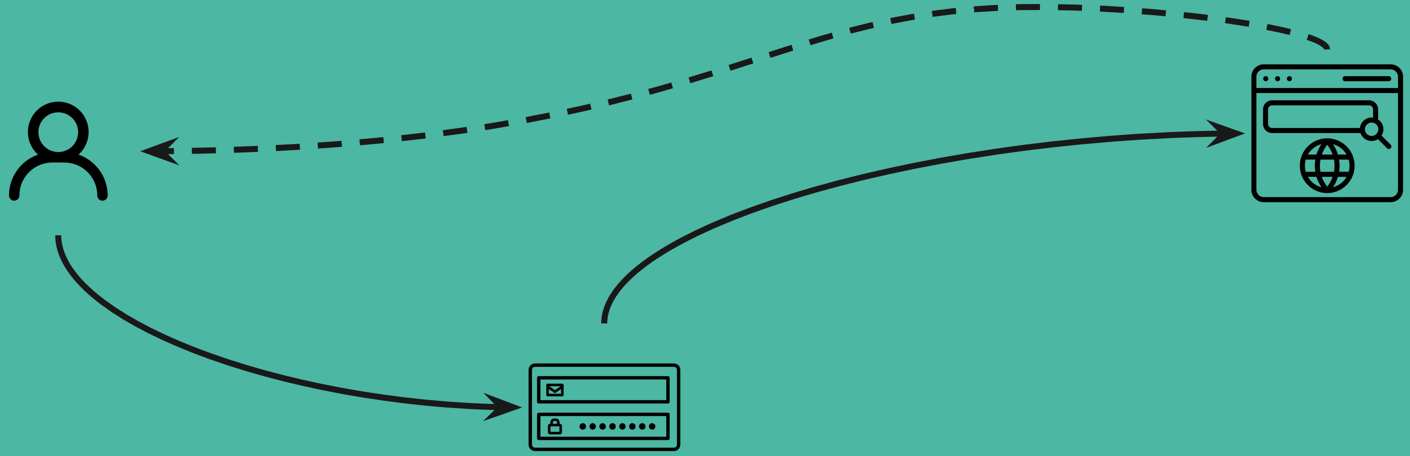




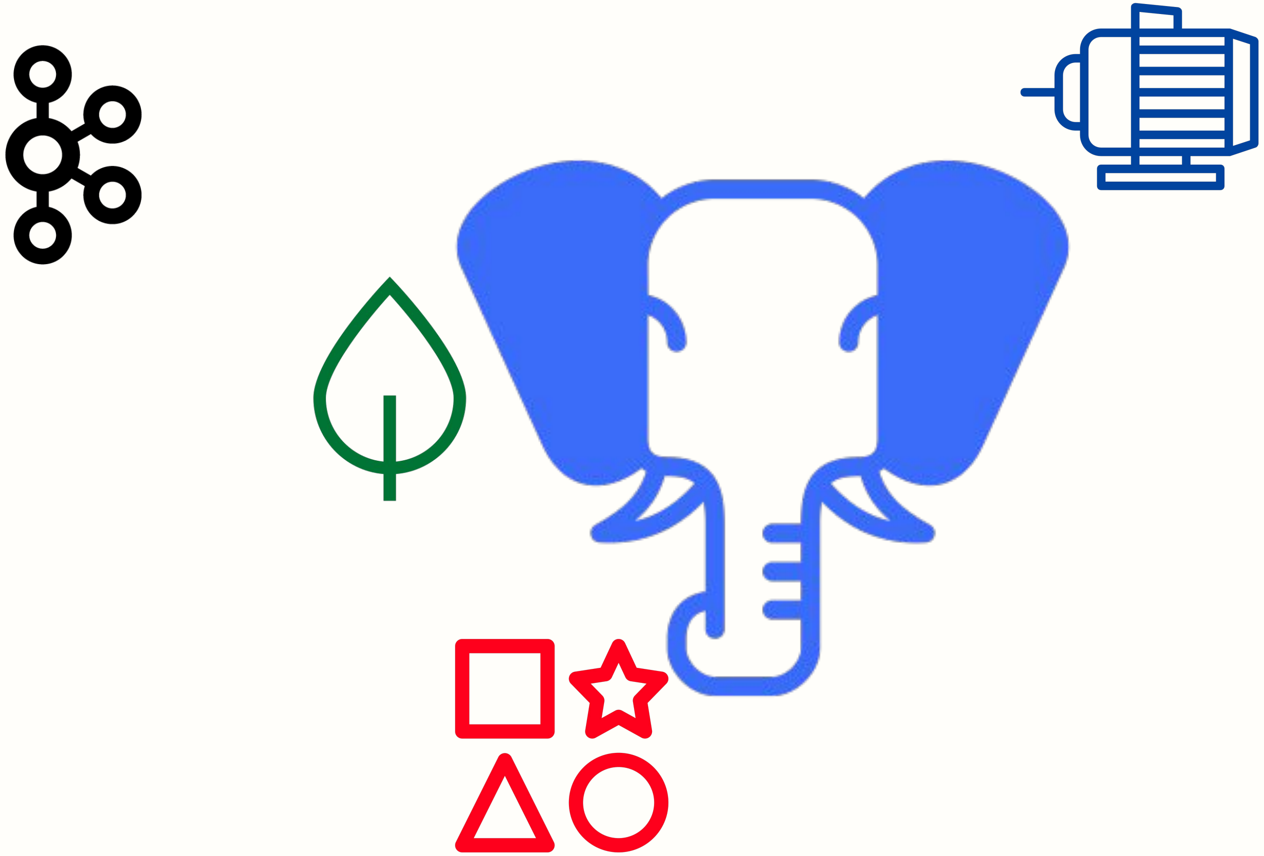






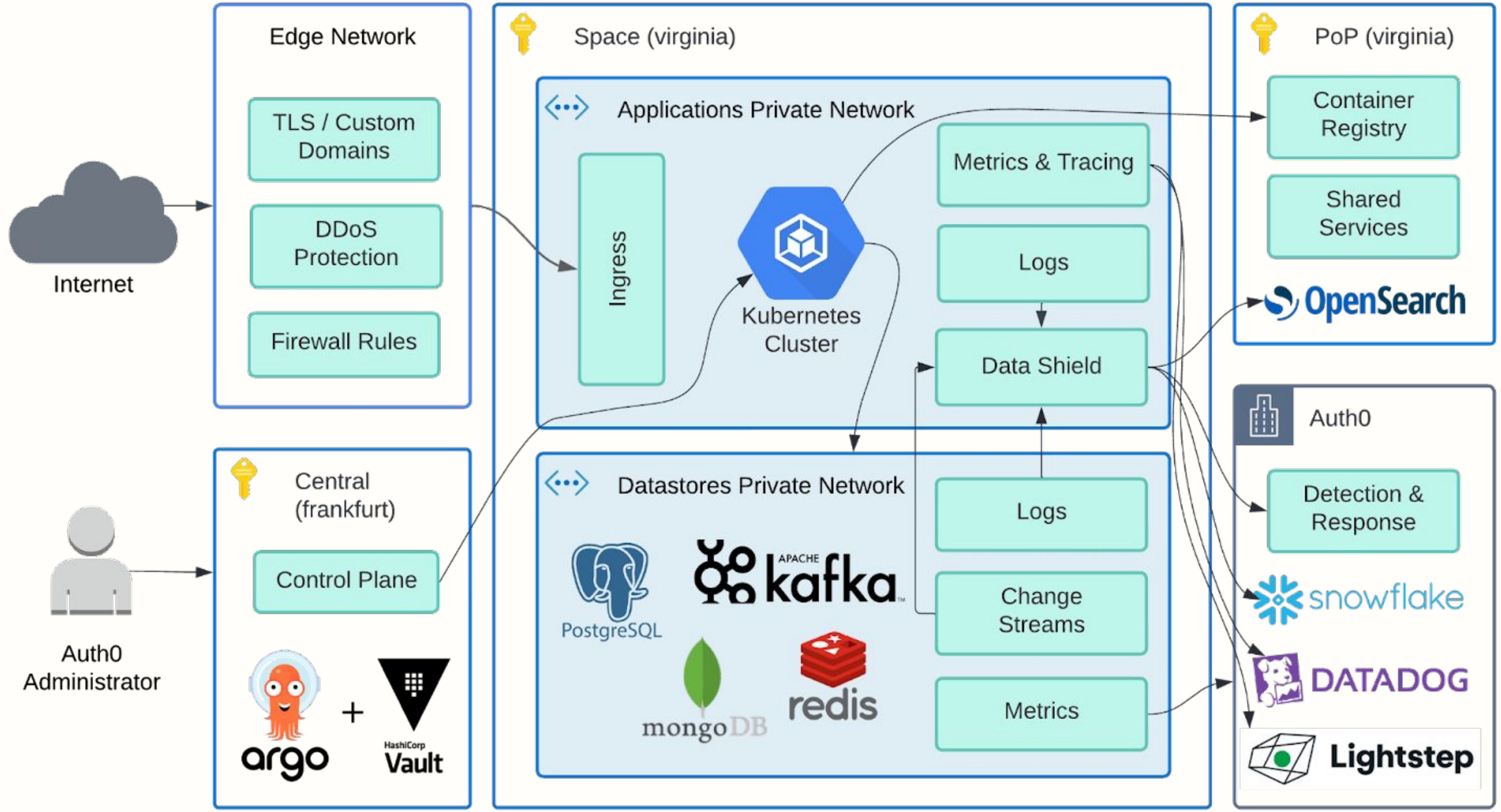


Pretty simple, right?



Auth0 by Okta

Operational Challenges



https://auth0.com/blog/the-architect-s-view-of-auth0-s-new-private-cloud-platform/

okta

Platform Complexity

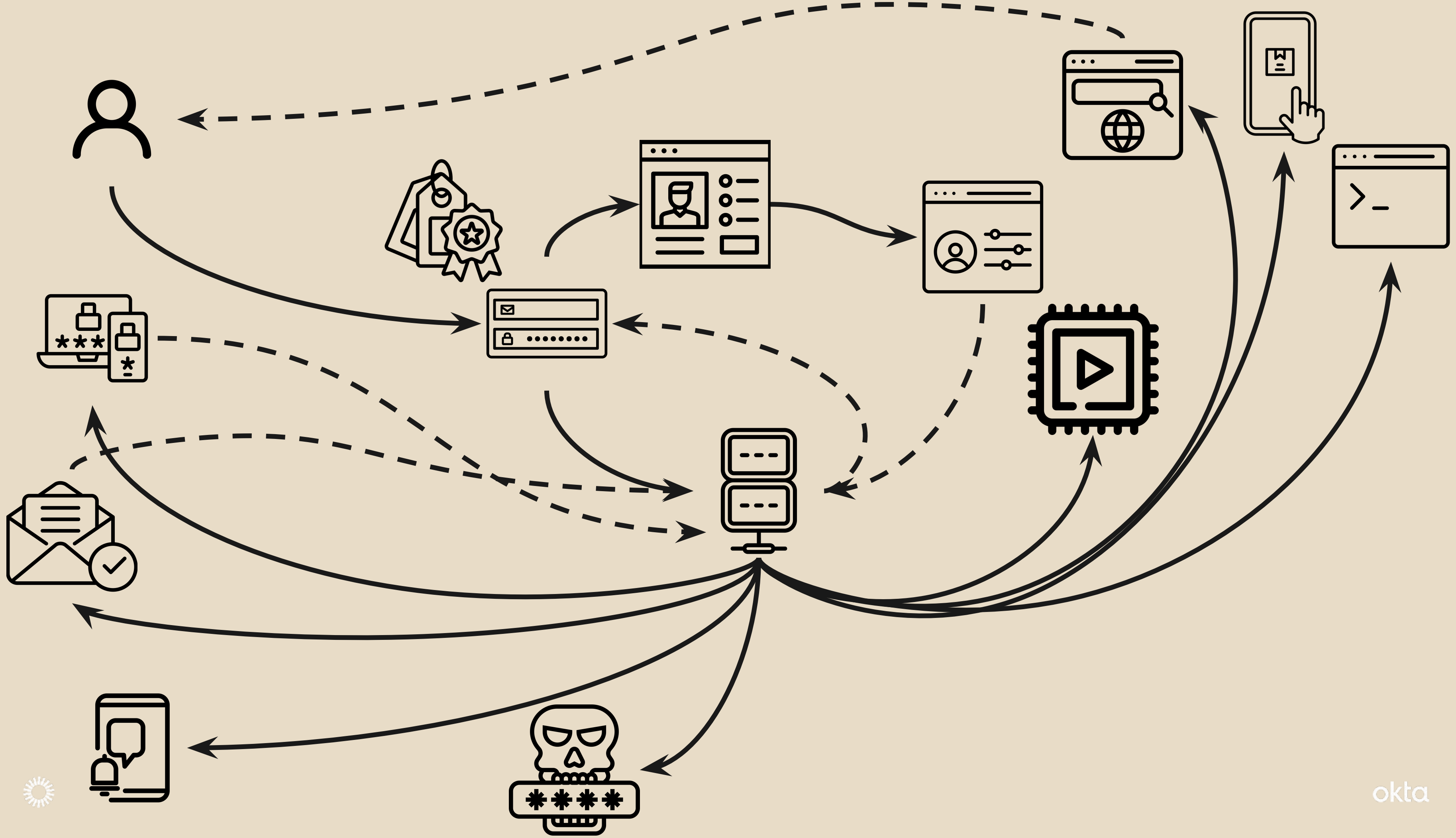
How many toys do we have to play with

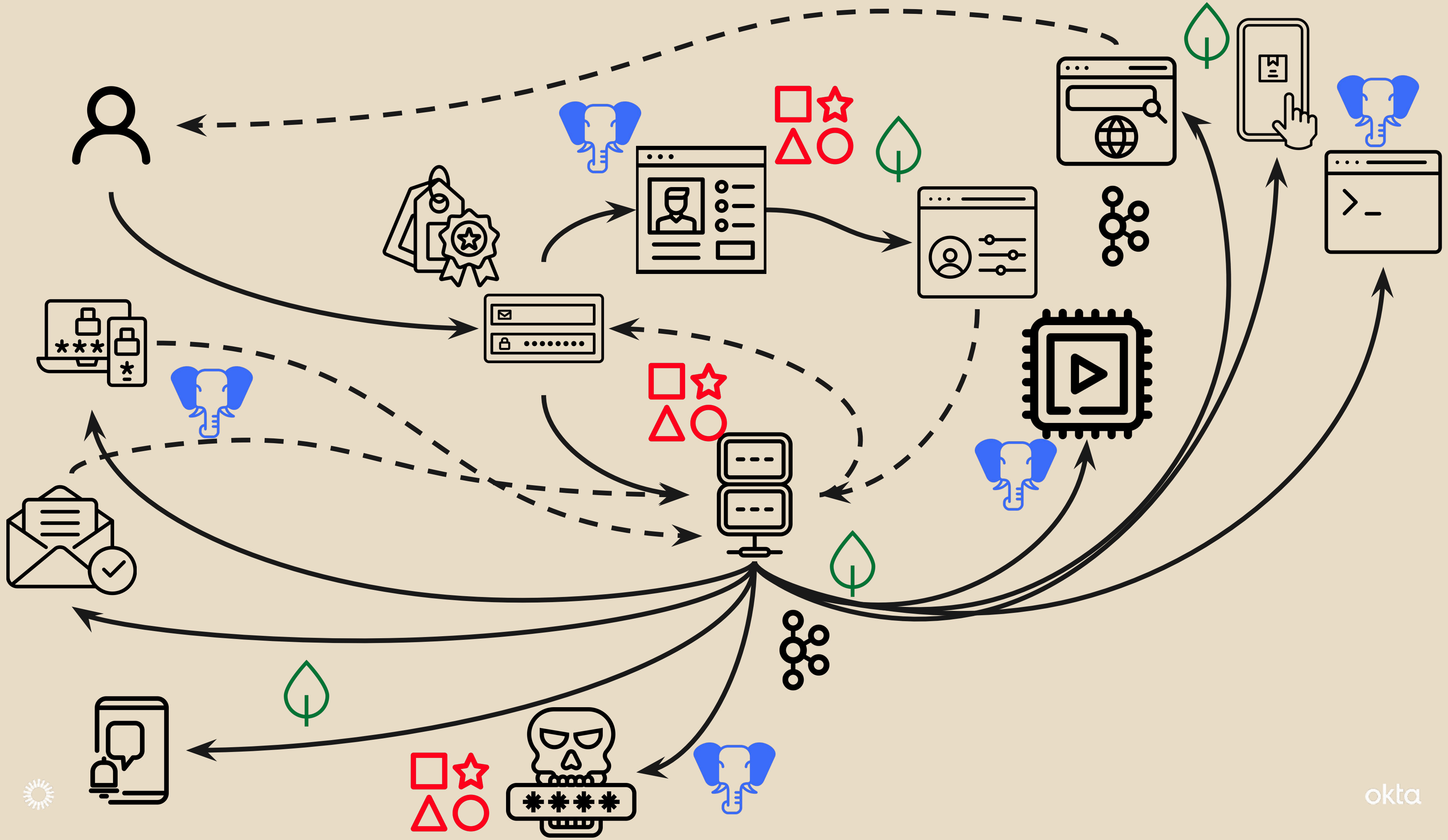
As systems and services evolve new features require different underlying infrastructure – changes on dependencies

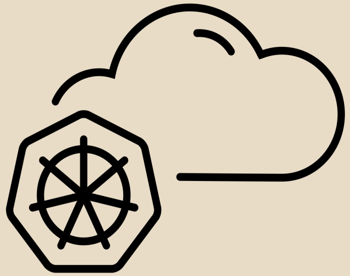
How these different systems interact with each other and what “substrate” systems evolve – monitoring, logs, metrics

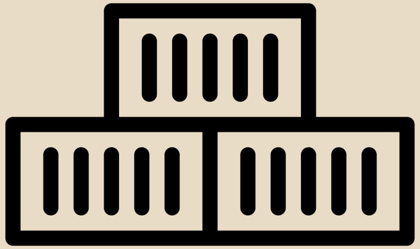
Maintenance, version management, infrastructure releases, and scaling events.

okta

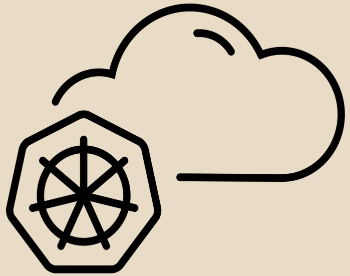


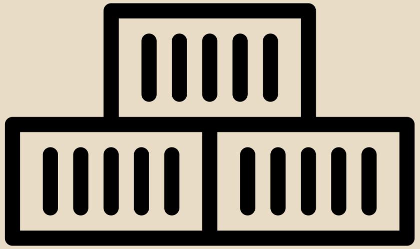






Cloud Native





Cloud Native

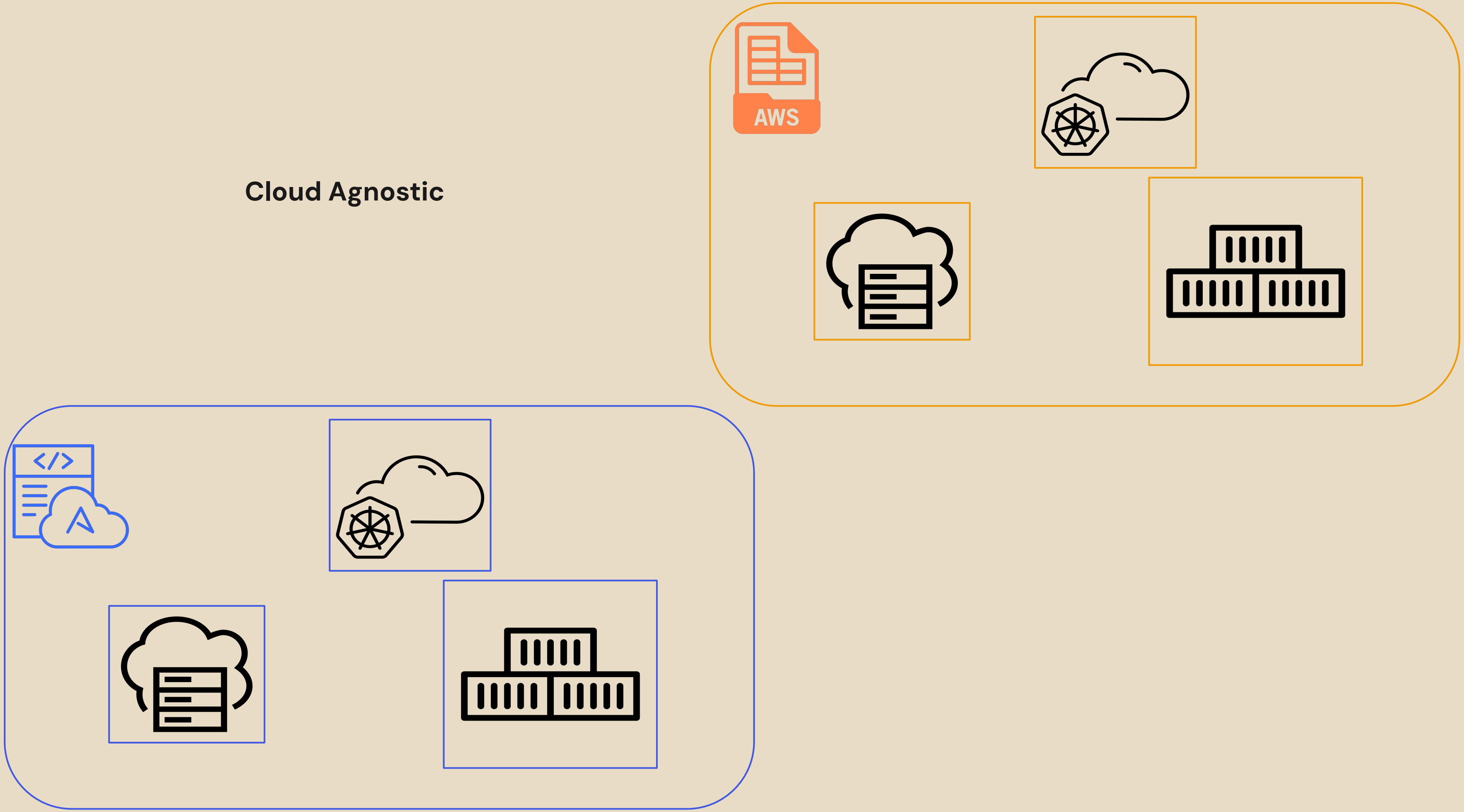
DBaaS

Cloud Native

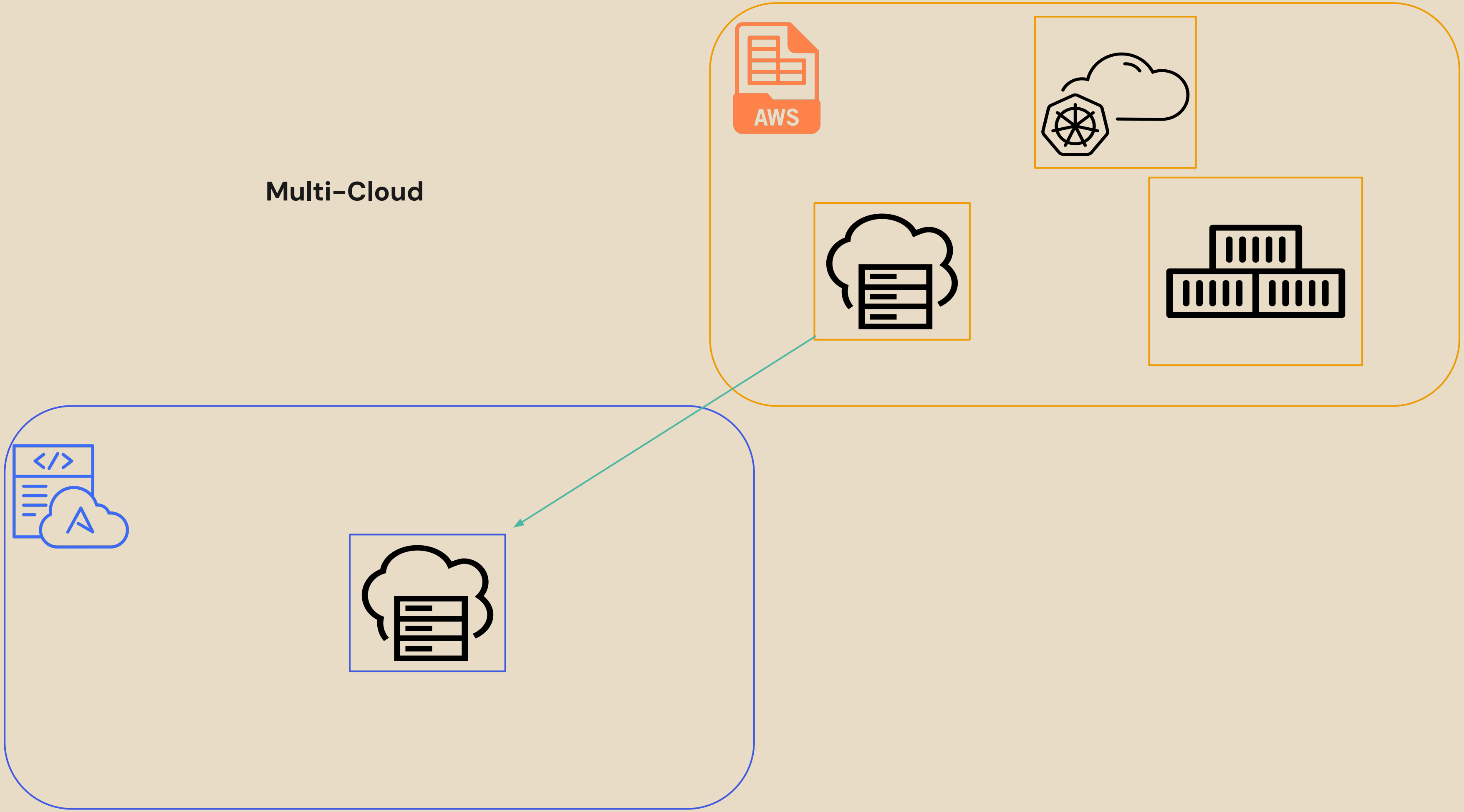
DBaaS

Isolation

Cloud Agnostic



Multi-Cloud



Controlled Operational Challenges

- Logging
- Metrics
- Monitors and Alerts
- Internal Network Configuration
- Release Management
- Major Version Upgrades
- Deployment Failures
- Testing

Not So Under Control Operational Challenges

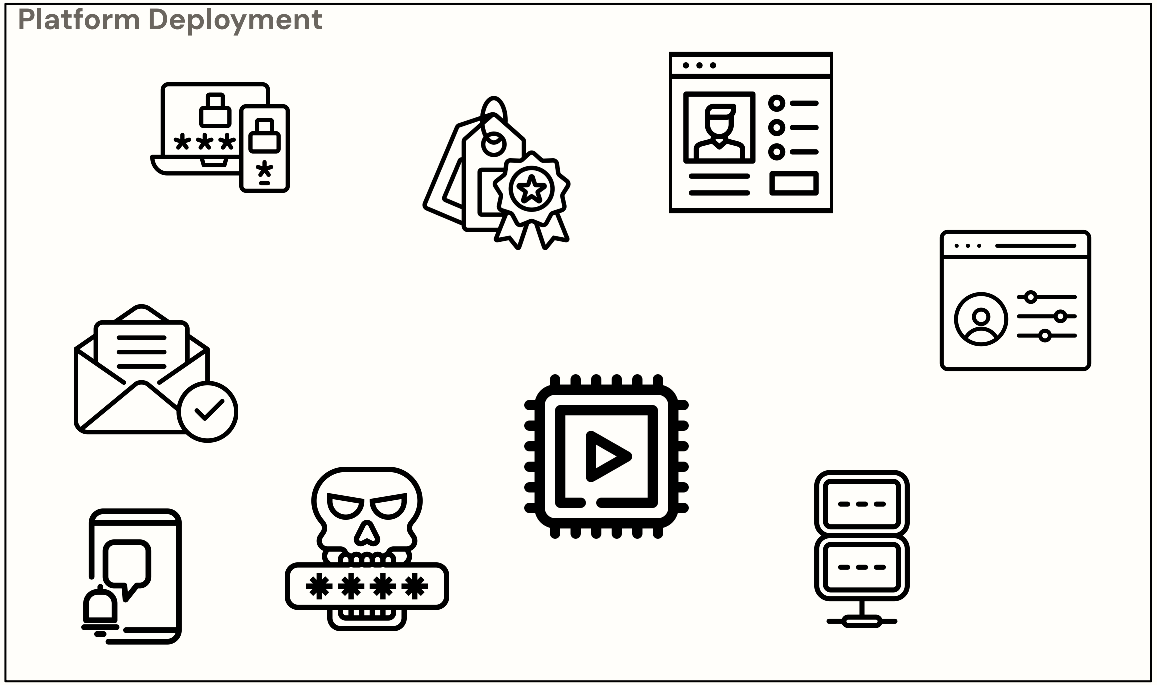
- Attacks
- Customer configuration creativity
- Third Party Provider Outages
- Scale induced miss-calculation
- Cascading failures
- External Network

Operational Challenges: Scaling

Exhibit 1 - Tales of things that are not how one thinks they are!

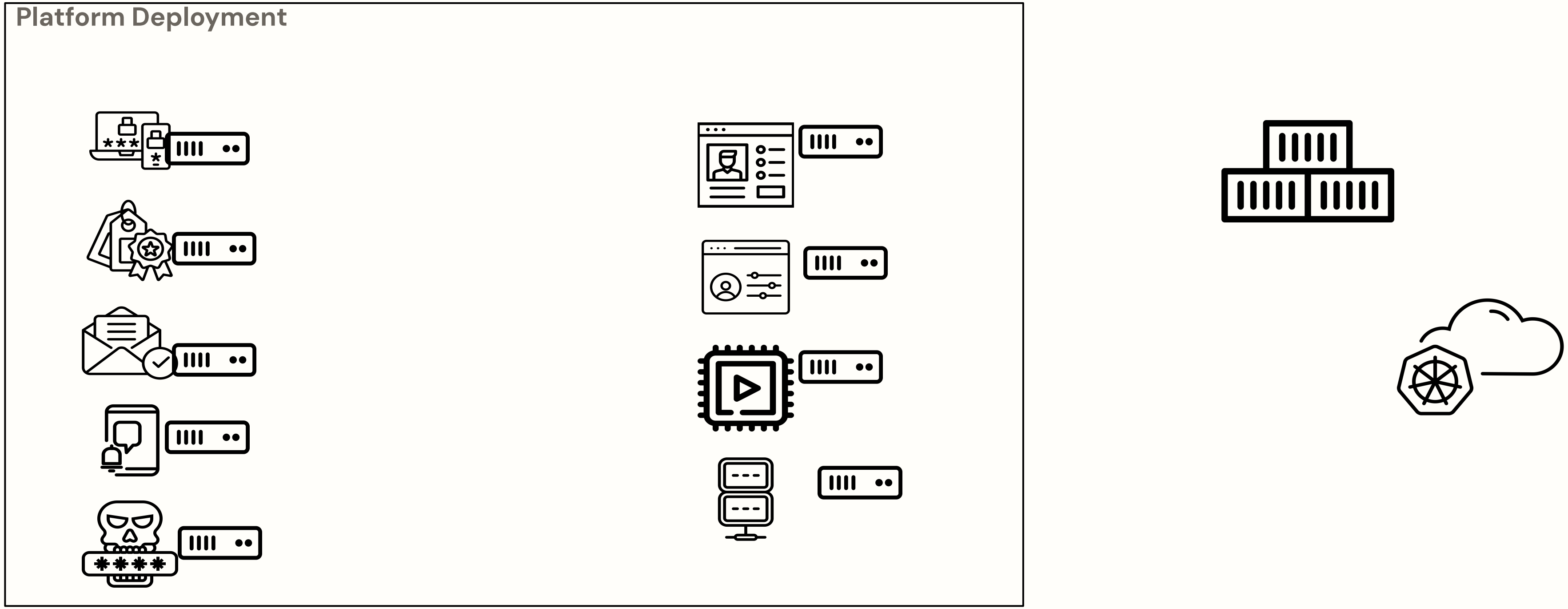
Operational Challenges: Scaling

Exhibit 1 – Tales of things that are not how one thinks they are!



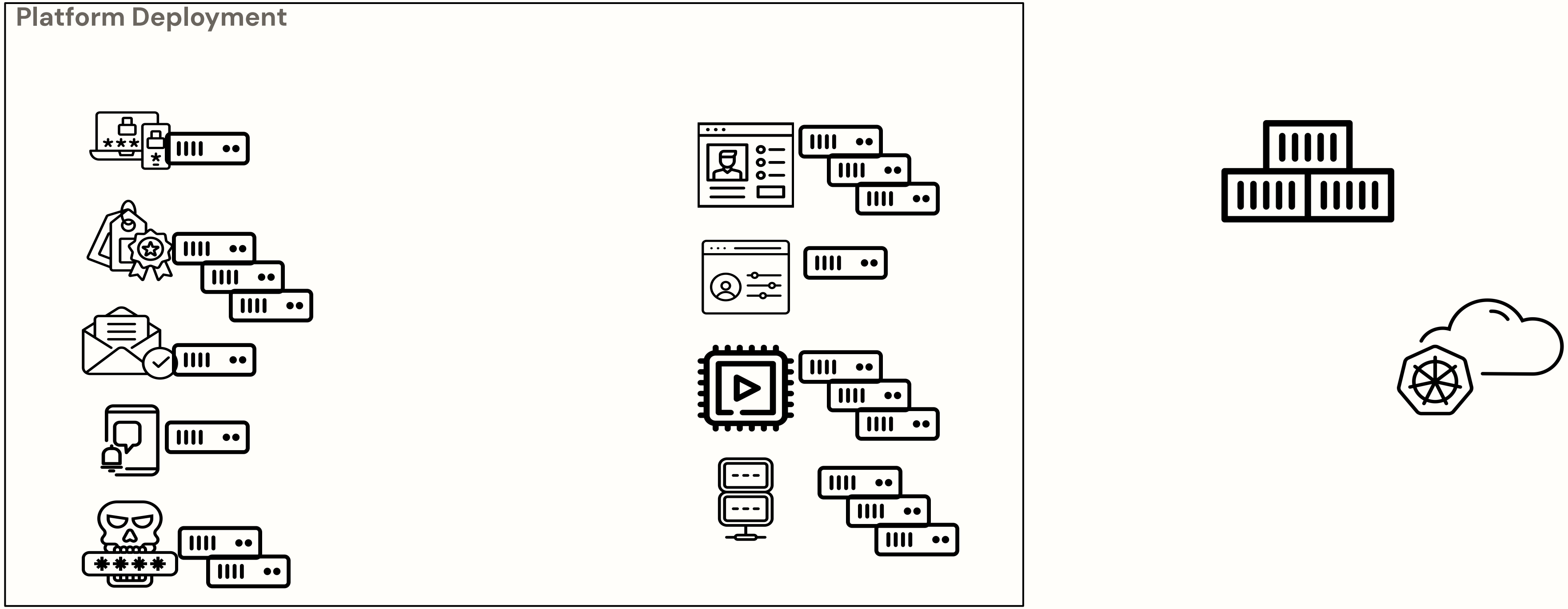
Operational Challenges: Scaling

Define the “expected” needs in terms of instances



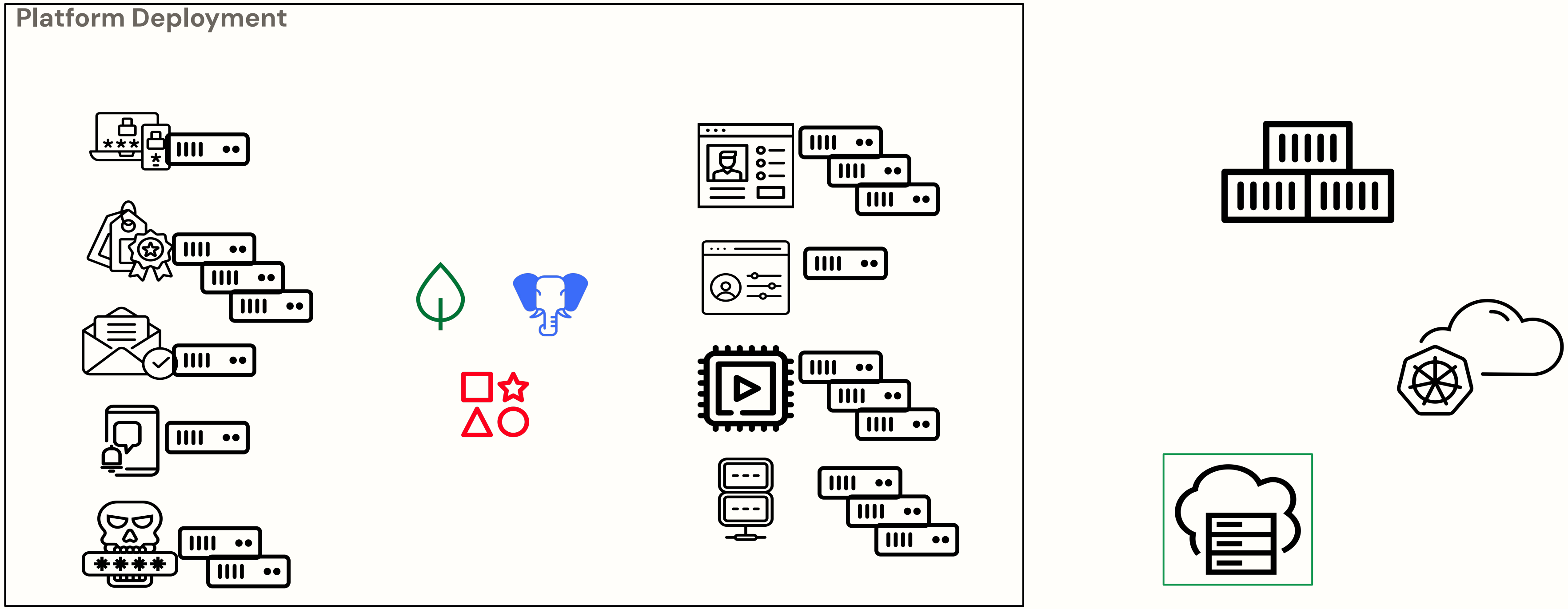
Operational Challenges: Scaling

If things grow, auto-scale FTW!



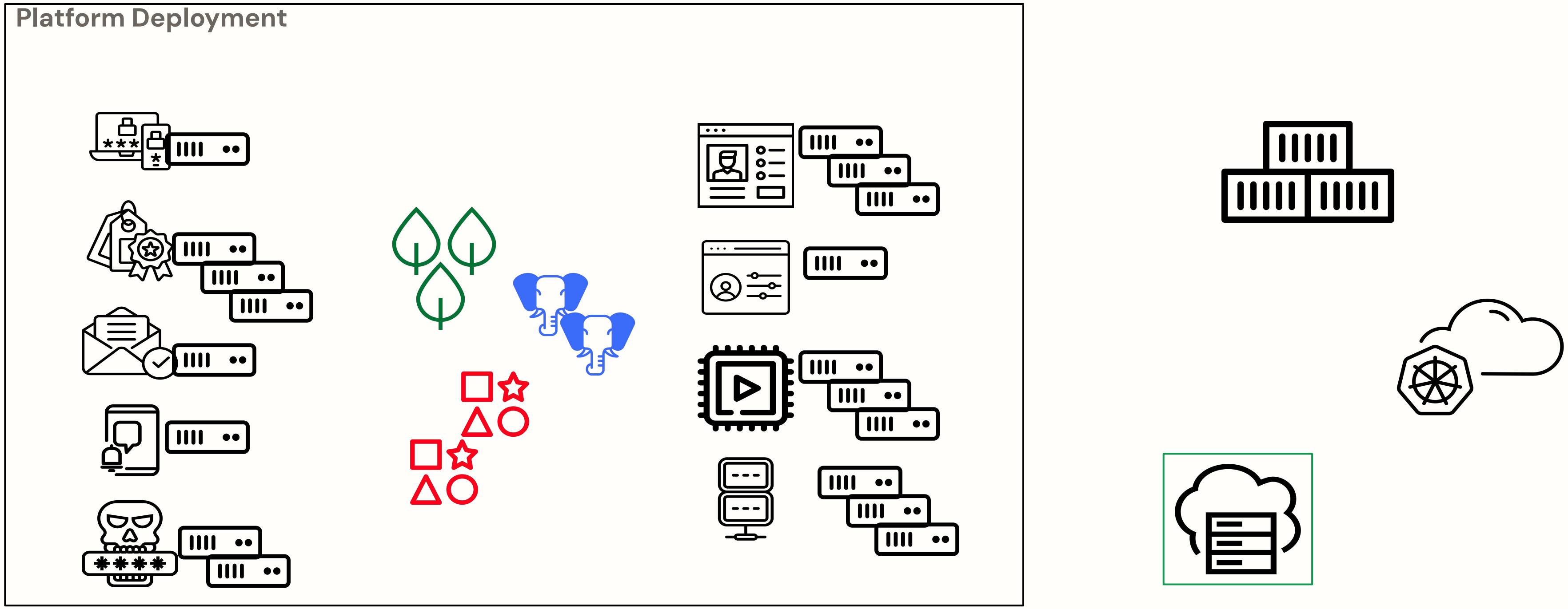
Operational Challenges: Scaling

Ok, you grow the client apps … what about the databases?



Operational Challenges: Scaling

These are deployed in a redundant architecture



Multiple Redundancy

Databases will fail, make sure we can withstand any single node failure

Redundant Deployment

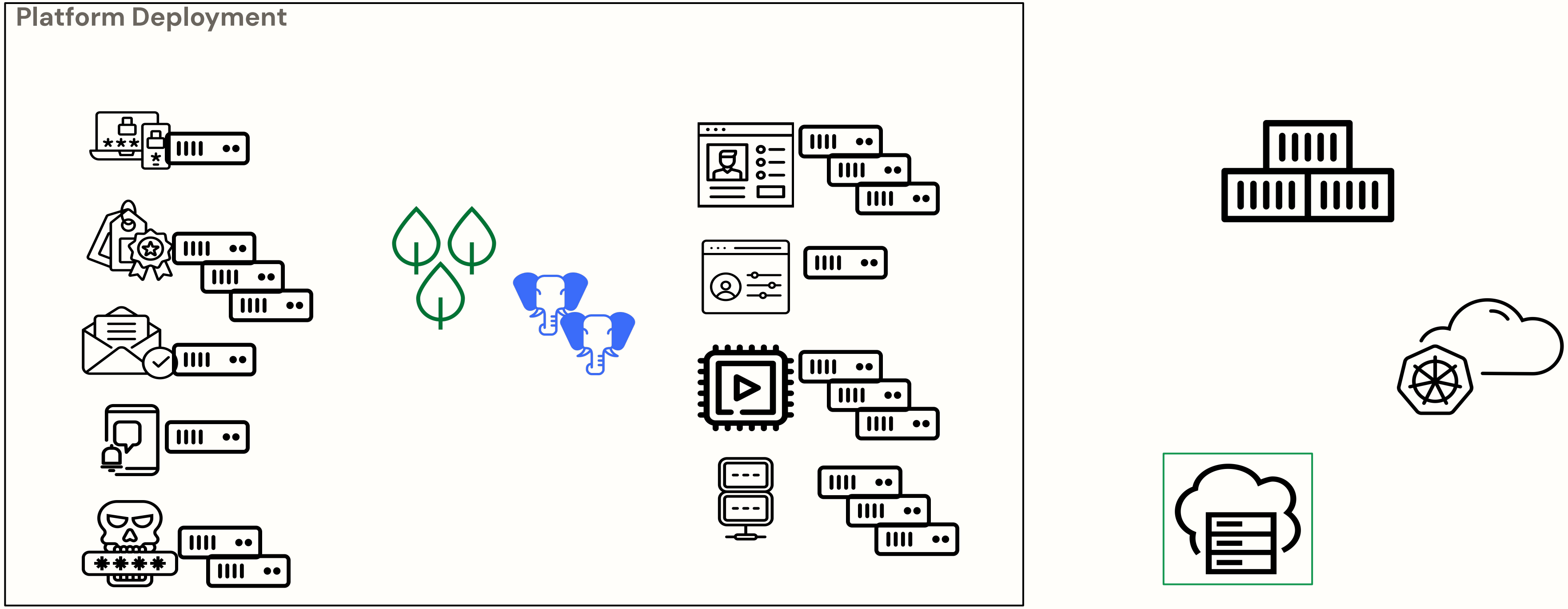
AZ Spread

PITR Enabled

okta

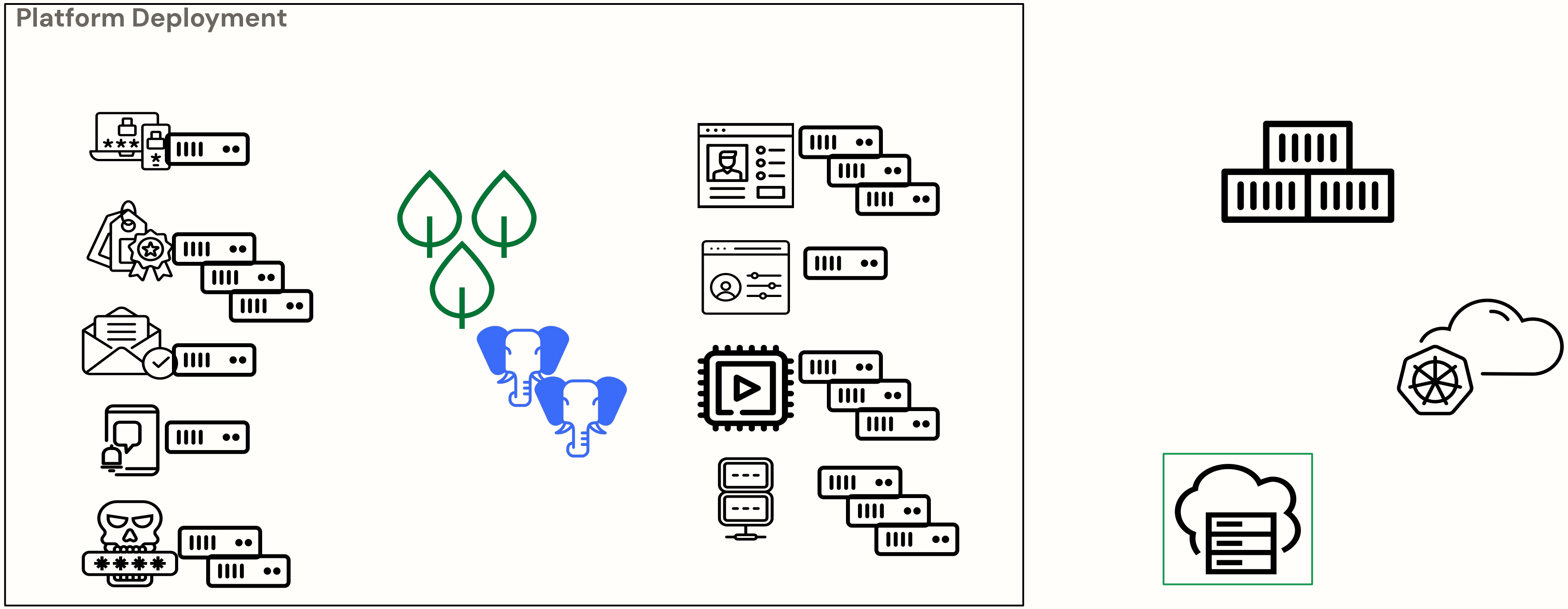
Operational Challenges: Scaling

Leaving the ephemeral stuff a side for a second...



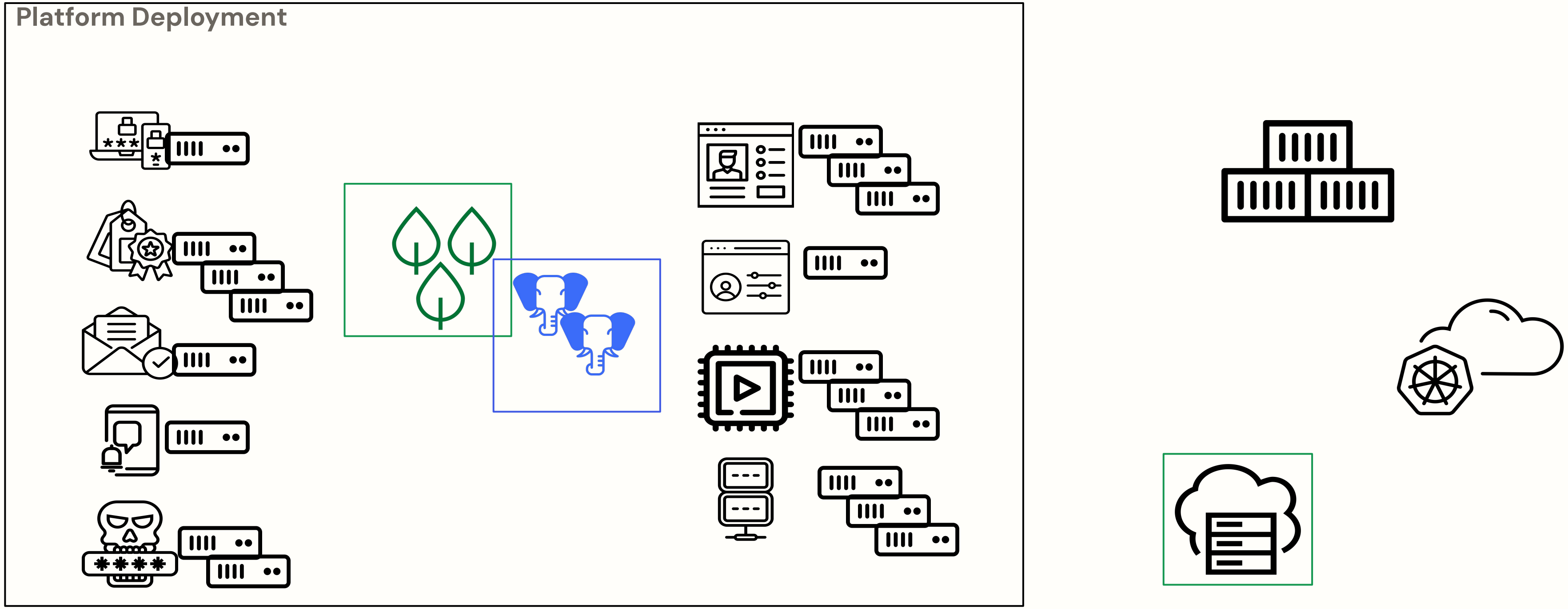
Operational Challenges: Scaling – Vertical

But when you need to increase capacity



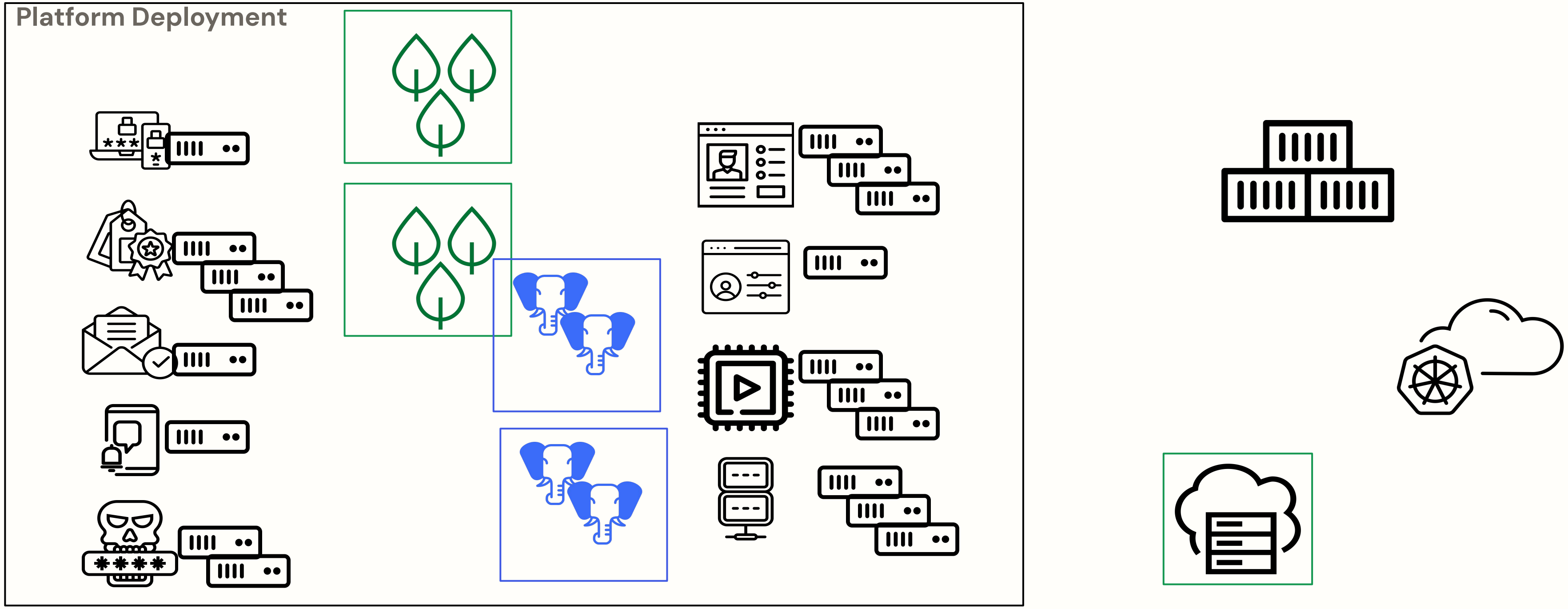
Operational Challenges: Scaling

Or, if you cluster things nicely



Operational Challenges: Scaling – Horizontal

You scale out – sharding



okta
The World's Identity Company

When to scale out vs scale up ?

okta

Scaling Rules

Which rules do we follow for scaling

Vertical

- Preferred mechanism
- Increase compute and/or storage capacity as needed
  - Based on spot brust client needs
  - Constant check for increased capacity requirements
- Easy to operate
- Easy to automate

Scaling Rules

Which rules do we follow for scaling

Vertical

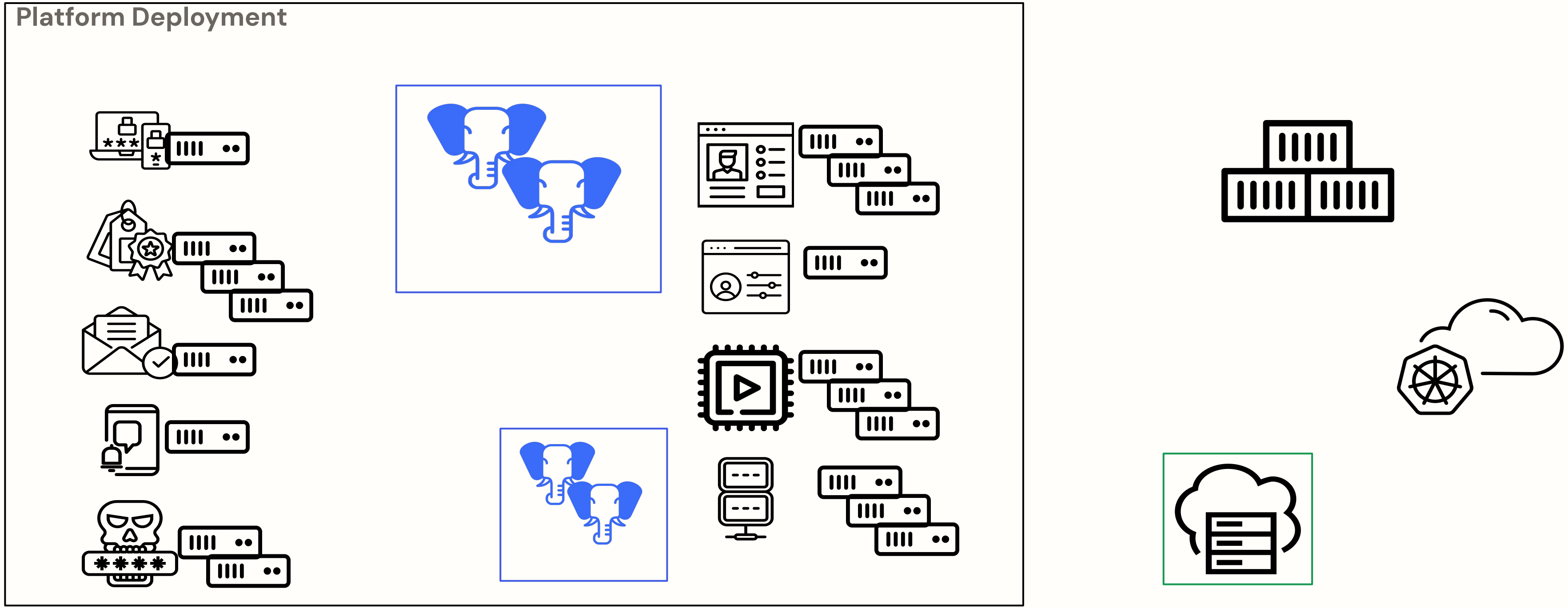
- Preferred mechanism
- Increase compute and/or storage capacity as needed
  - Based on spot brust client needs
  - Constant check for increased capacity requirements
- Easy to operate
- Easy to automate

Horizontal

- Manually split logical databases into separate clusters
  - Getting started with Citus Data | Aurora Limitless
- Group working sets based on:
  - Service criticality
  - Backup retention policies
  - Data lifecycle
- Allows for heterogeneous database cluster deployments
  - Add resources to databases to respond to their needs
  - Cost-effective mindset

Operational Challenges: Scaling - Auth0 Style

We do a bit of both



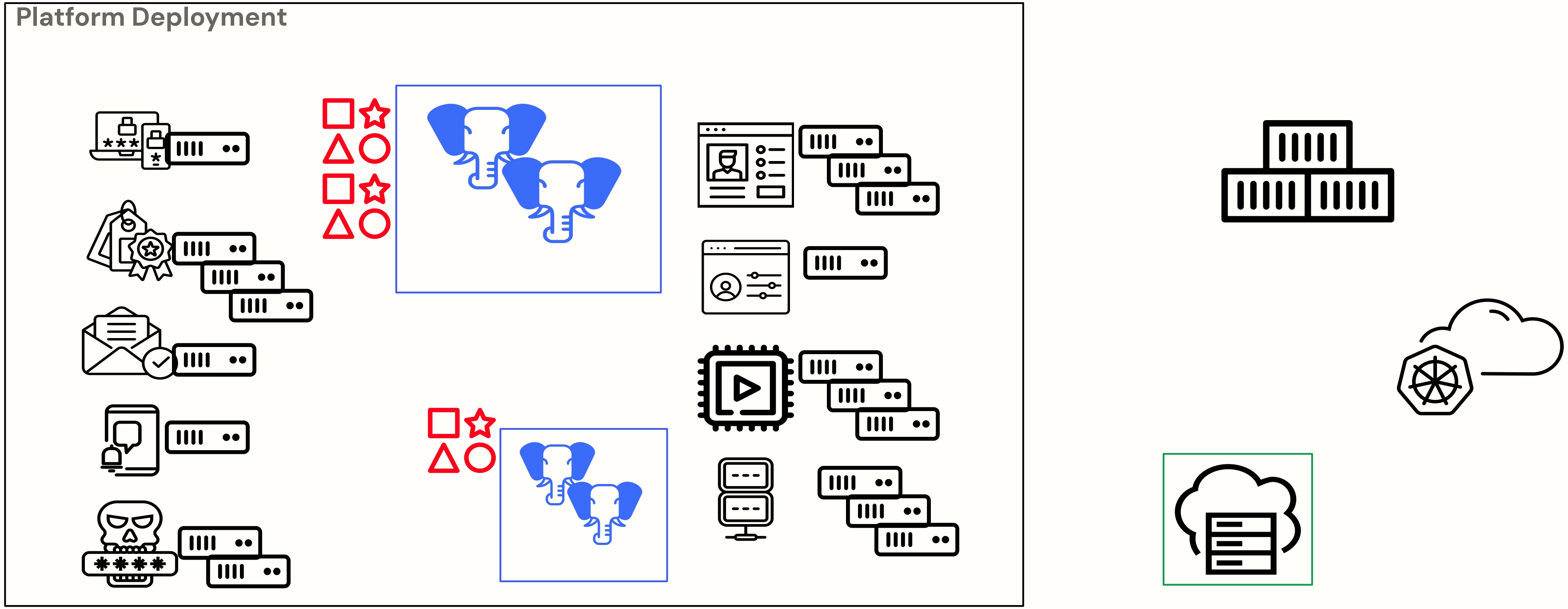
okta

Tier Based Architecture

Not all services have the same performance and resiliency profile - protect the most critical services -> build for failure

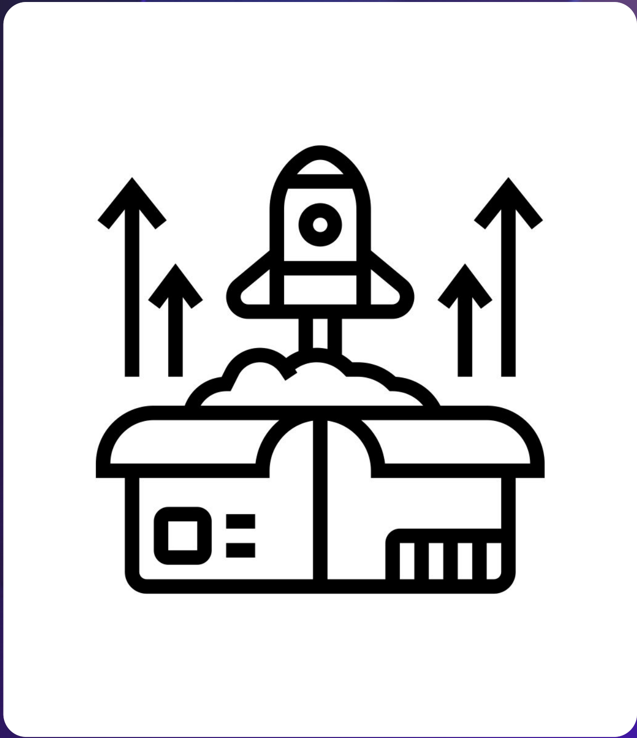
Operational Challenges: But don't forget caches!

Ephemeral Datastores Shock Absorbers



Auth0 by Okta

Service Releases and Infrastructure Operations



Platform Resiliency

Any weather proof

As we evolve our services we need to keep our fault tolerance and resiliency high

As we grow in terms of customer load and are subjected to more demanding scenarios we need to keep the system stable and reliable

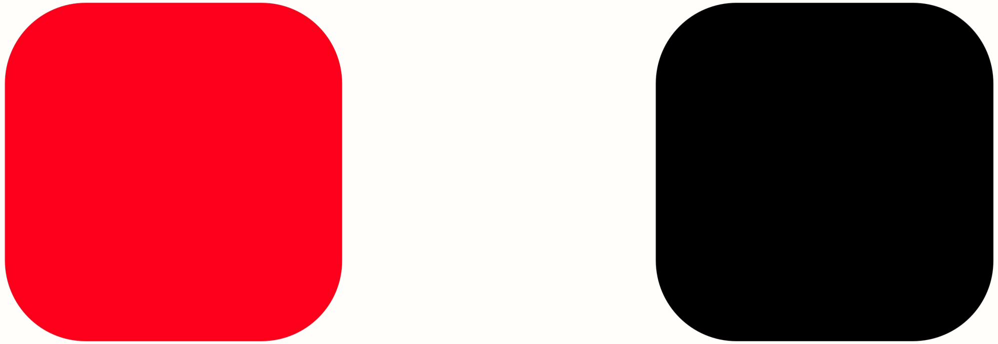
As we deploy and scale up our systems our customers should not be affected by any of such movements

okta

The World's Identity Company

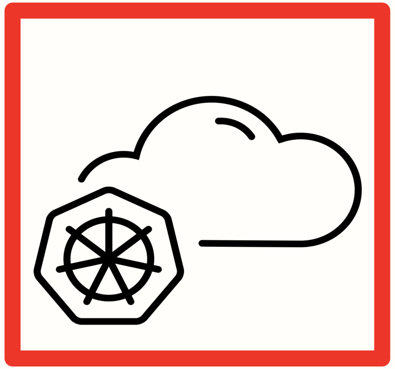
Platform Deployments

okta



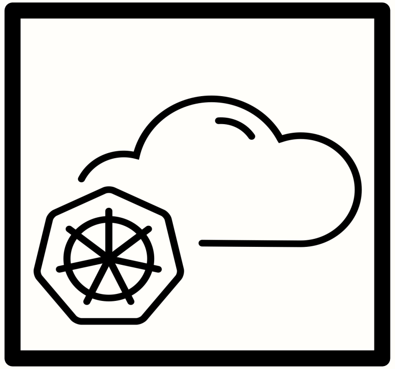
Red-Black Deployment, also known as Red-Green Deployment, is similar to Blue-Green Deployment. It involves maintaining two environments: the existing "Red" production environment and the new "Black" environment. Traffic is initially directed to the Red environment, while the Black environment is prepared and tested. Once validated, traffic is switched to the Black environment.

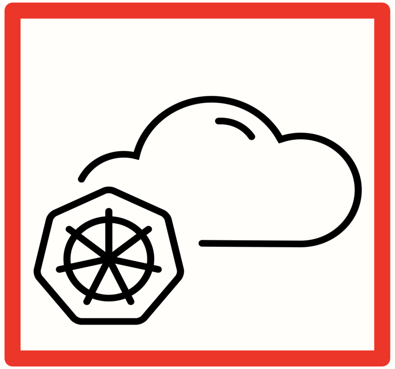
https://www.linkedin.com/pulse/deployment-models-explained-rizwana-begum

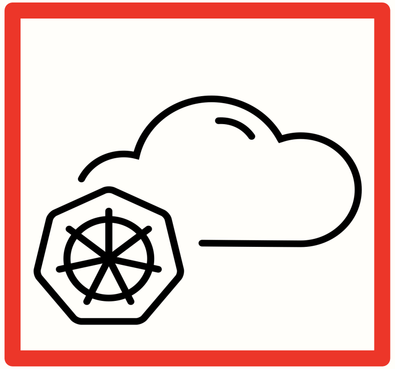


v0

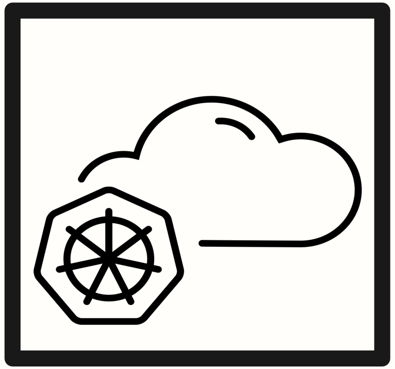
okta


v1

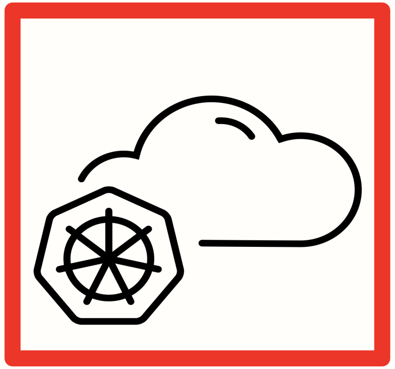

v0



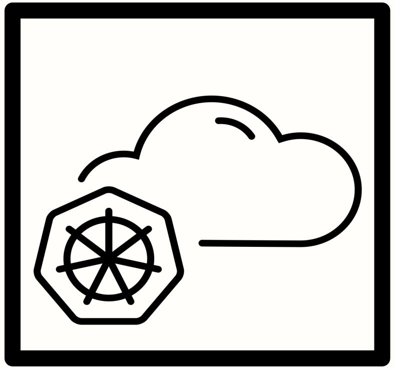
v1



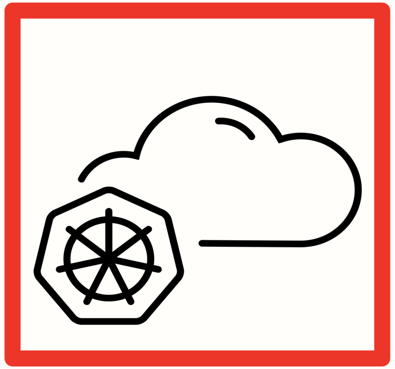
v0



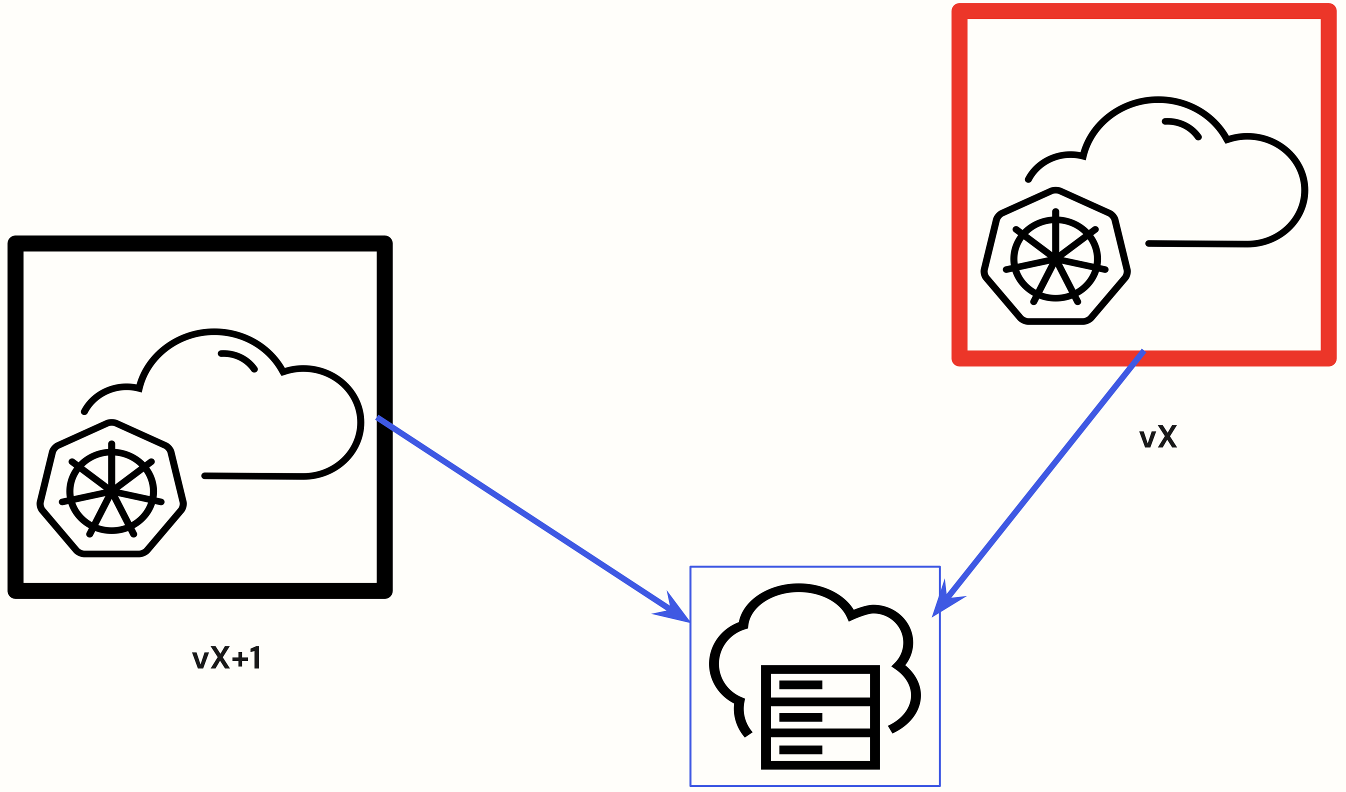
v1

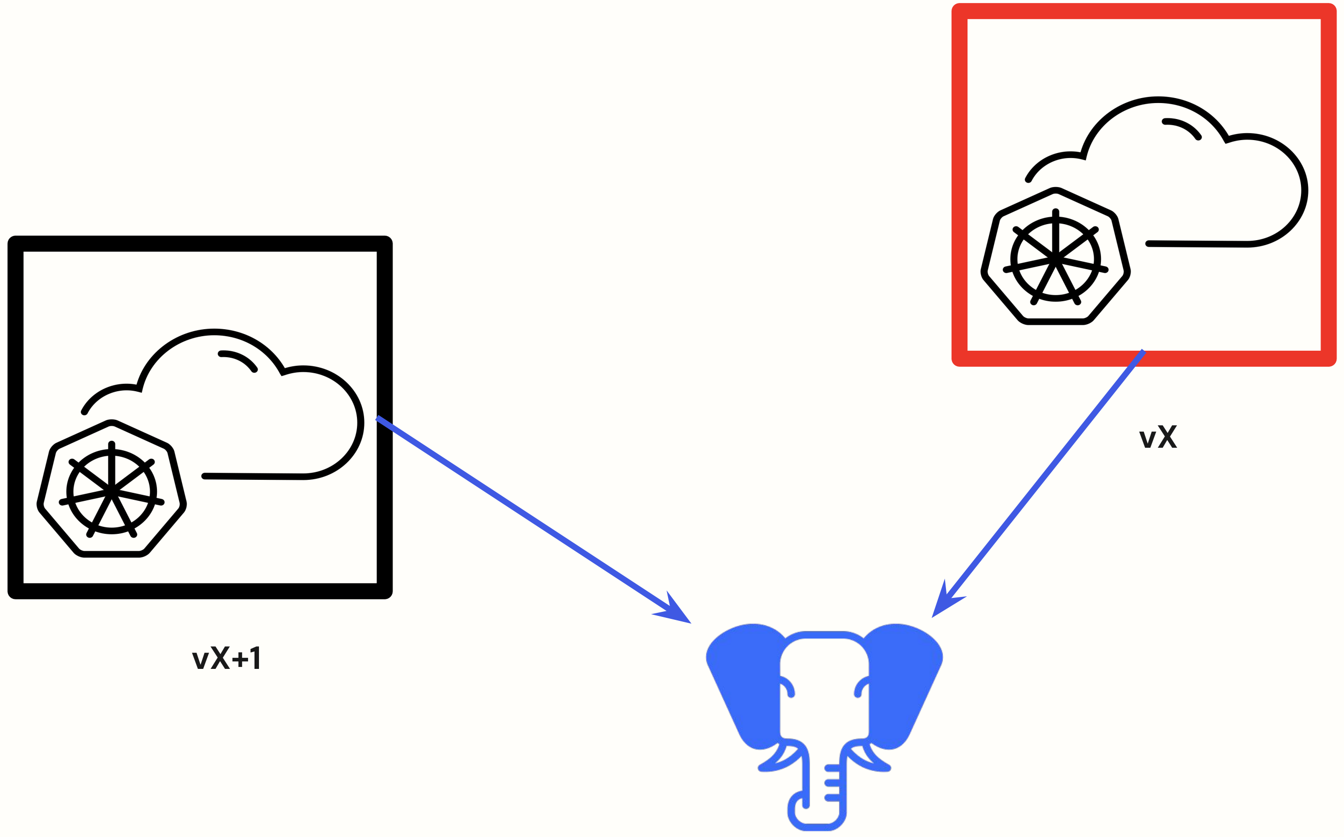


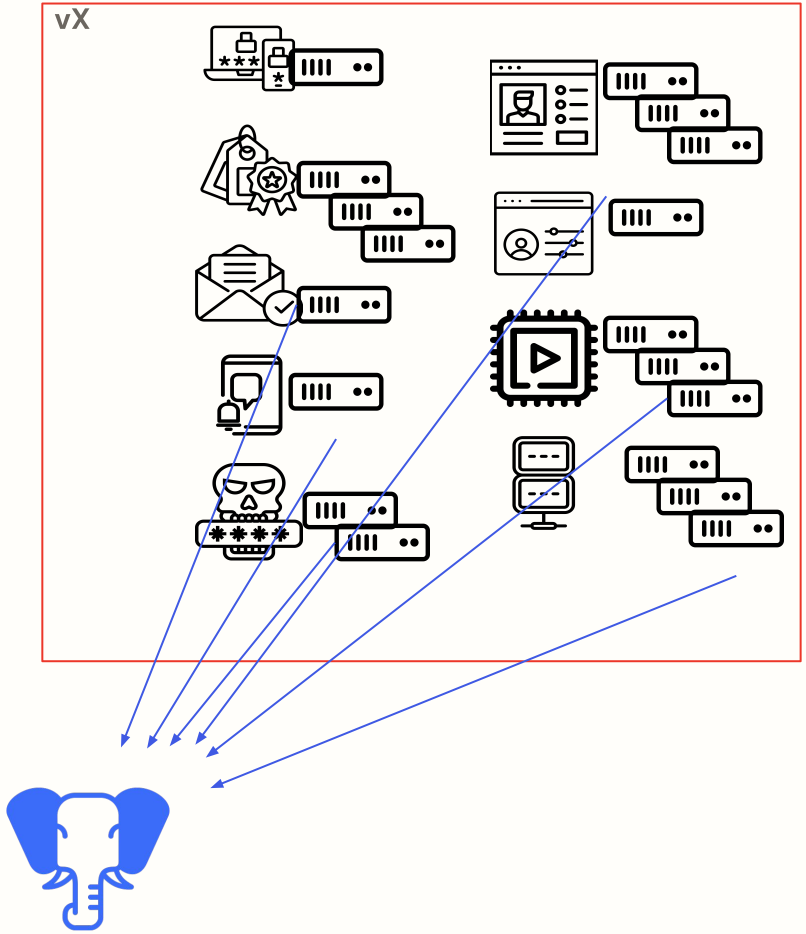
v2

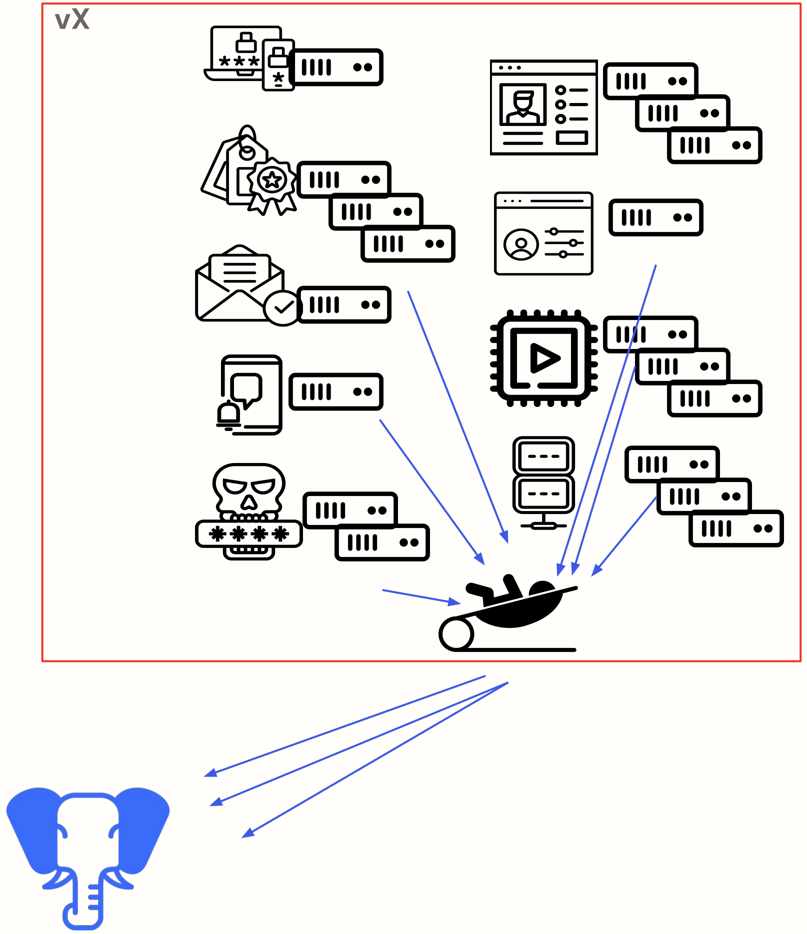


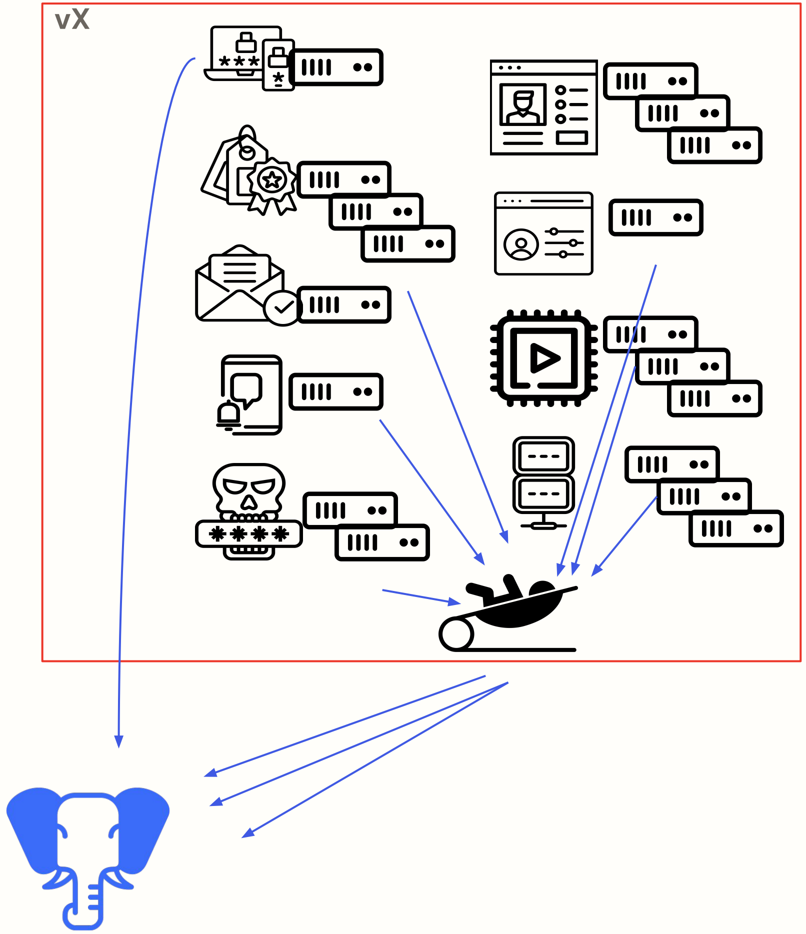
v1


vX
vX+1

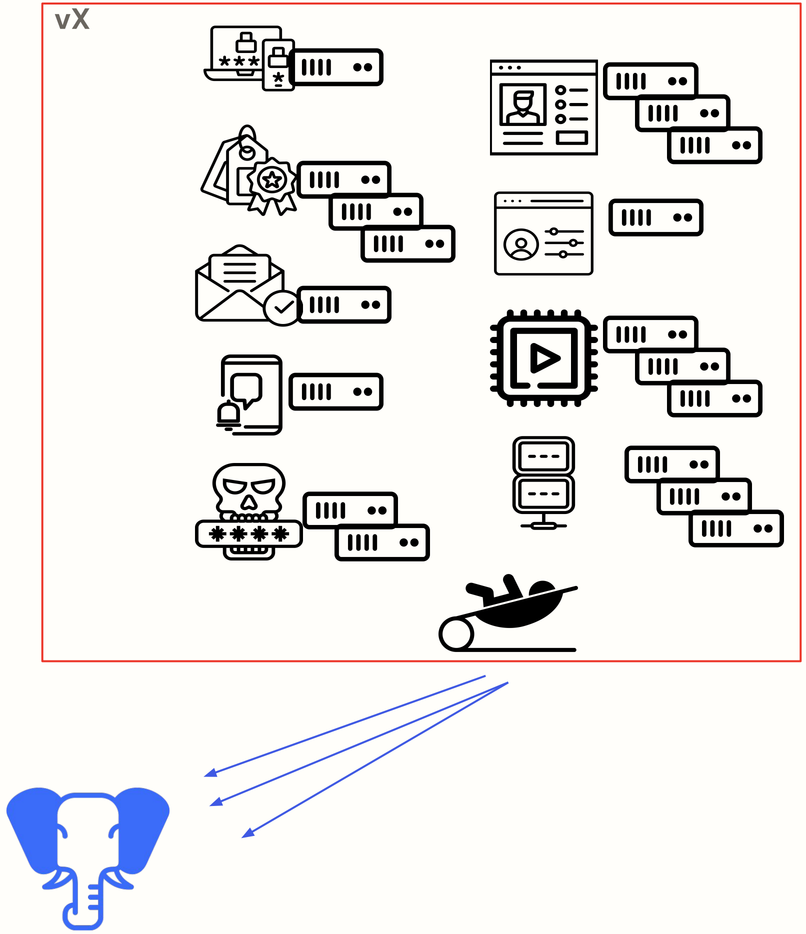

vX+1
vX

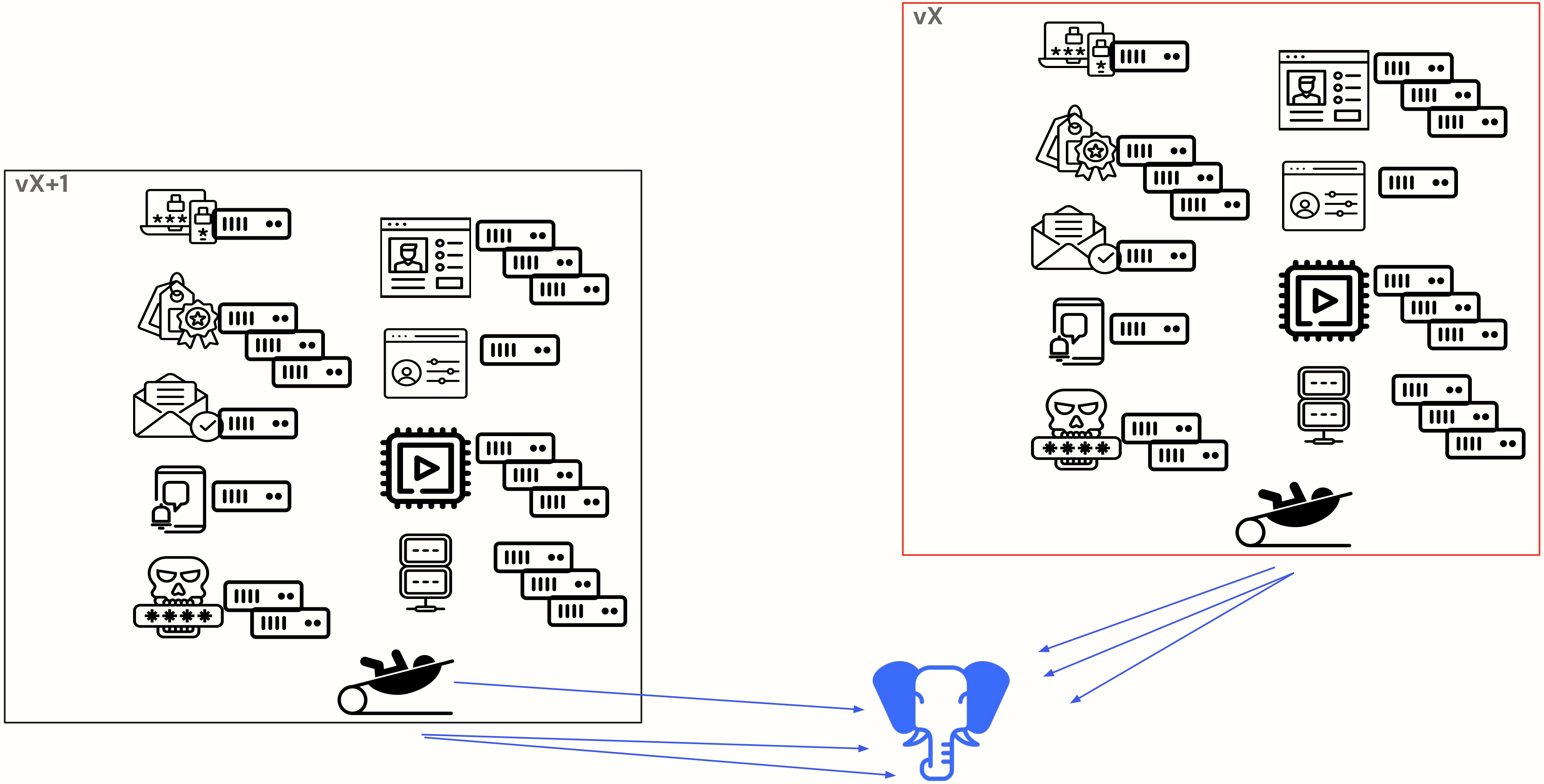

vX


vX

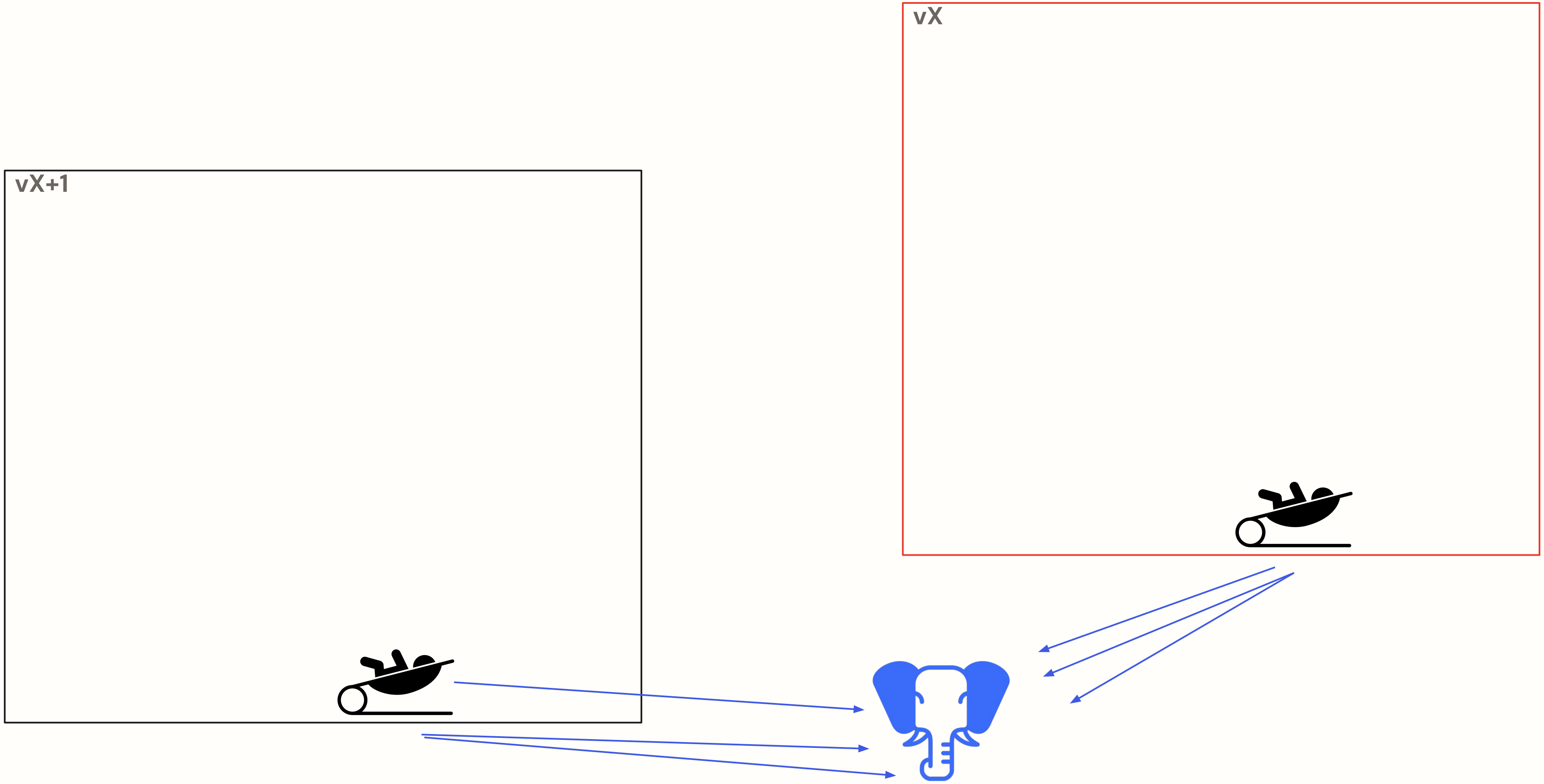


okta

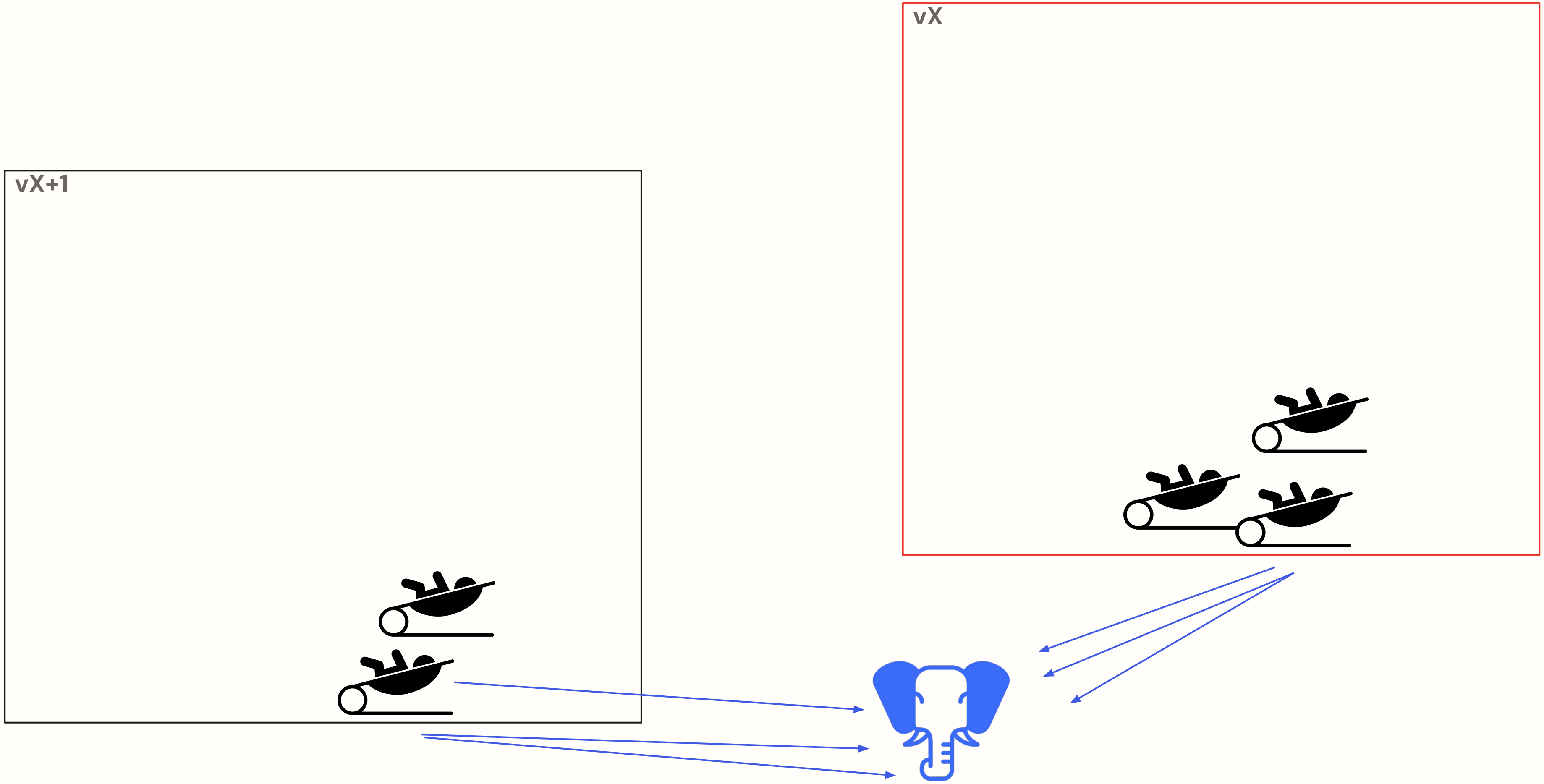

vX



okta



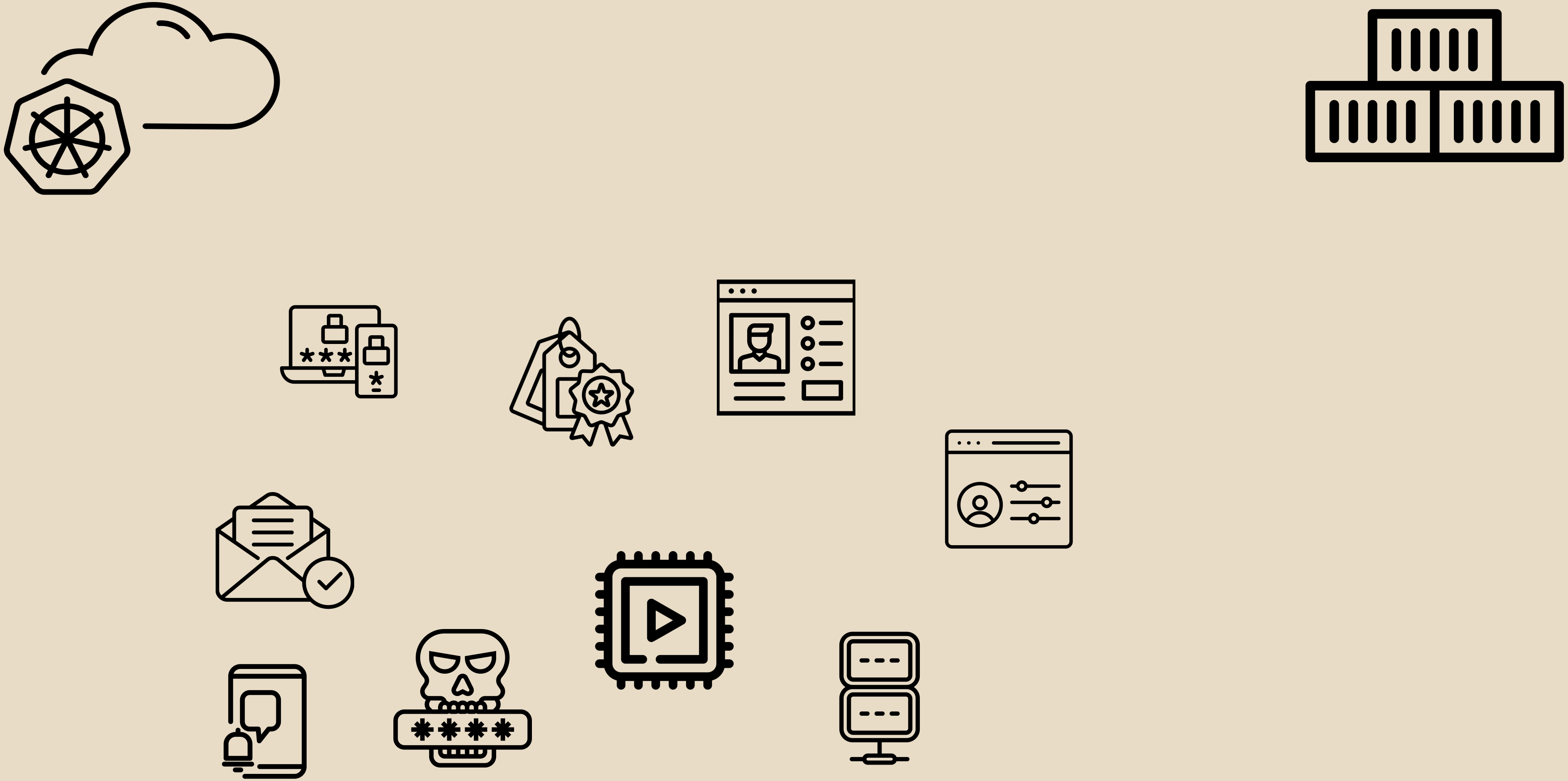
okta

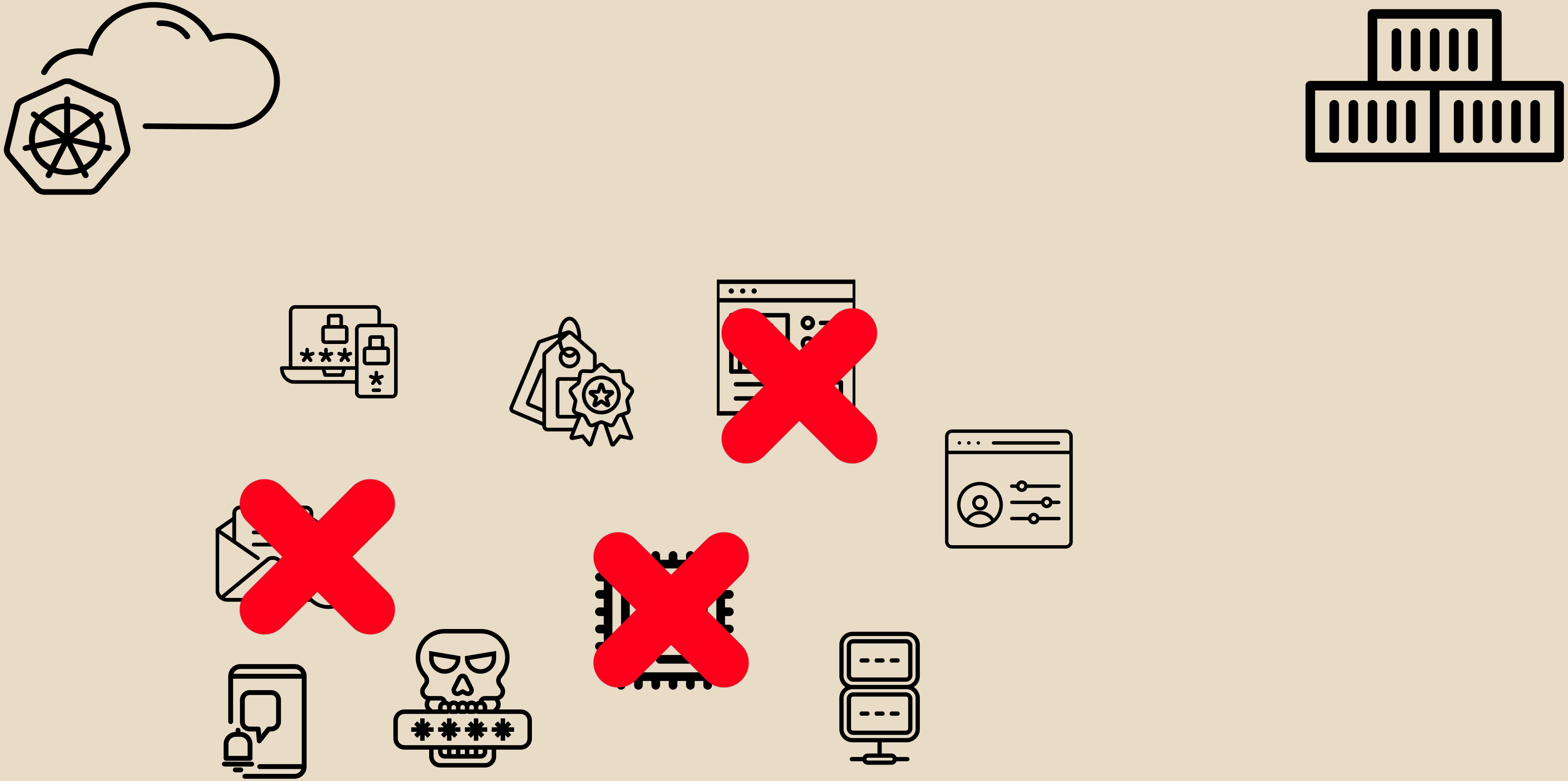

vX+1
vX

okta
The World's Identity Company

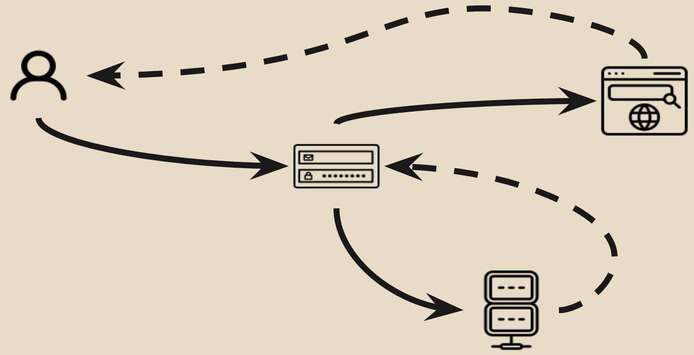
Service Resiliency

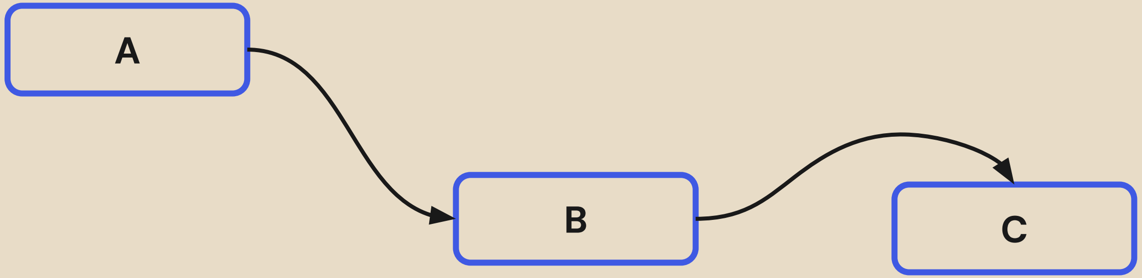
okta



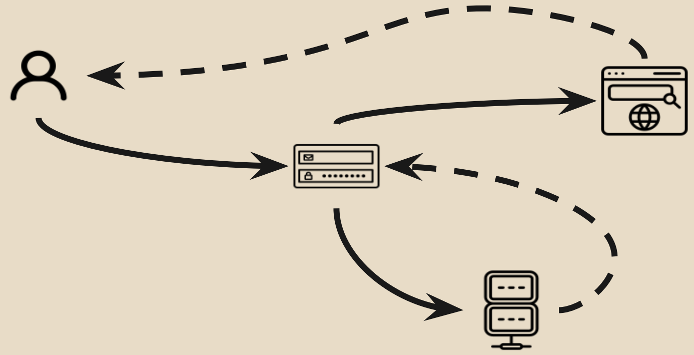


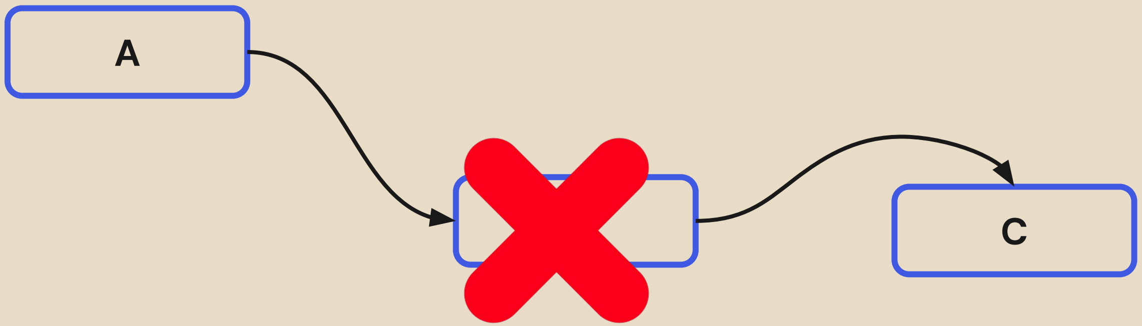
Service Dependencies



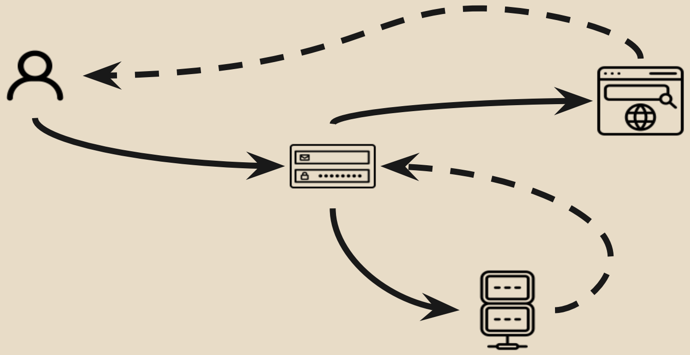


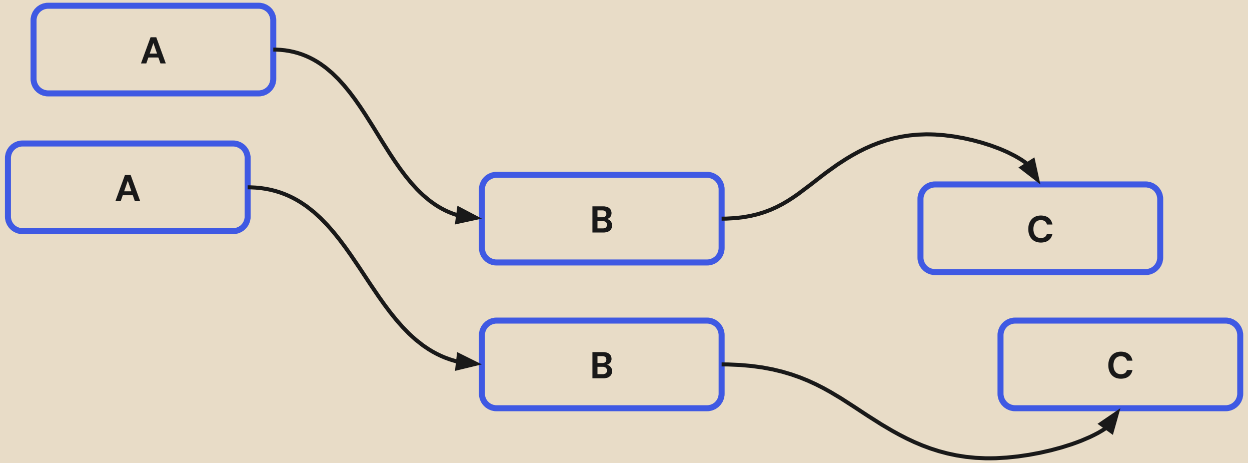
Service Dependencies



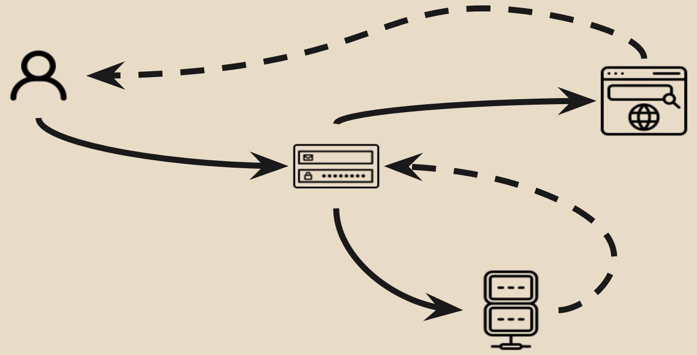


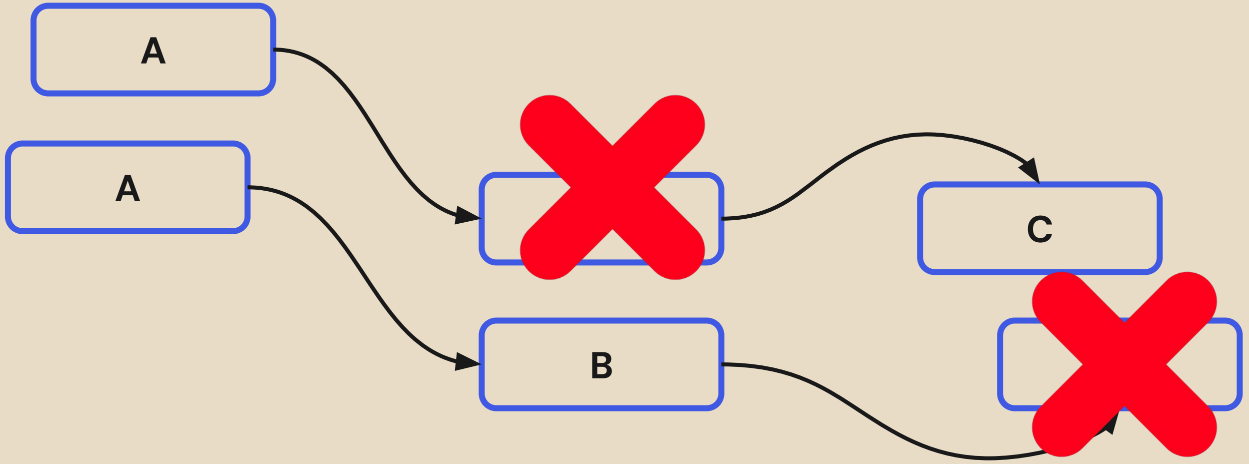
Service Dependencies



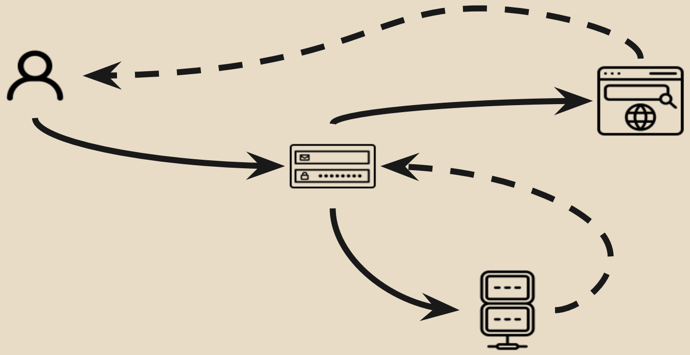


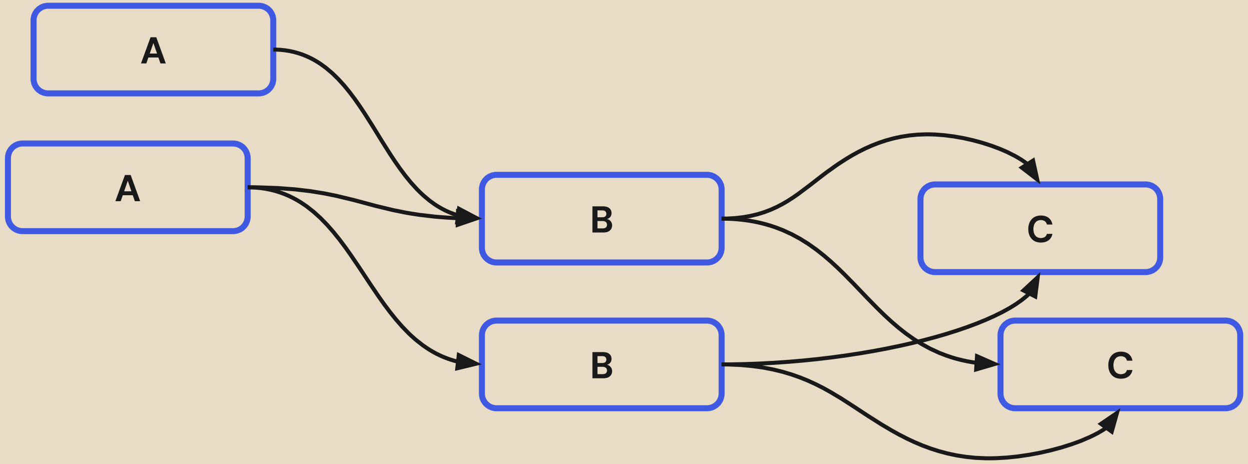
Service Dependencies



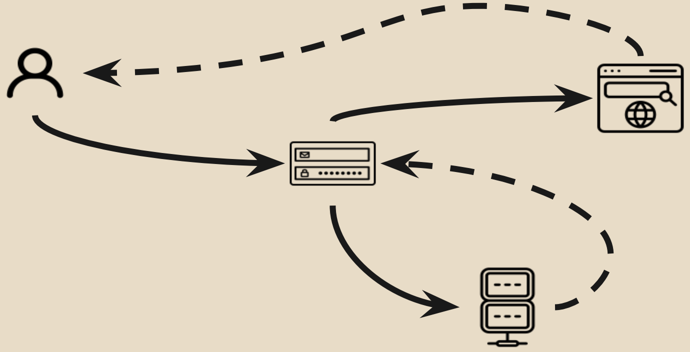


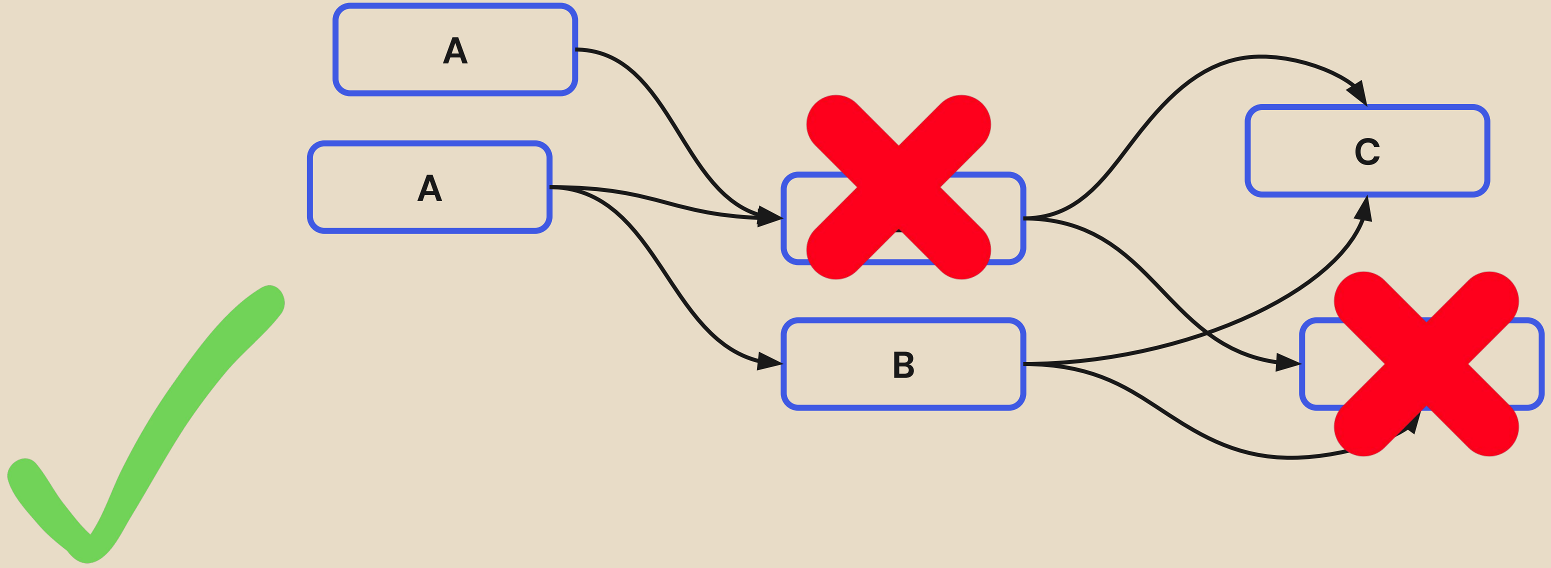
Service Dependencies



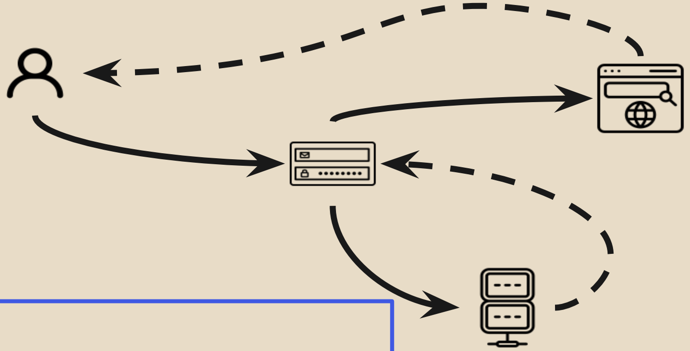


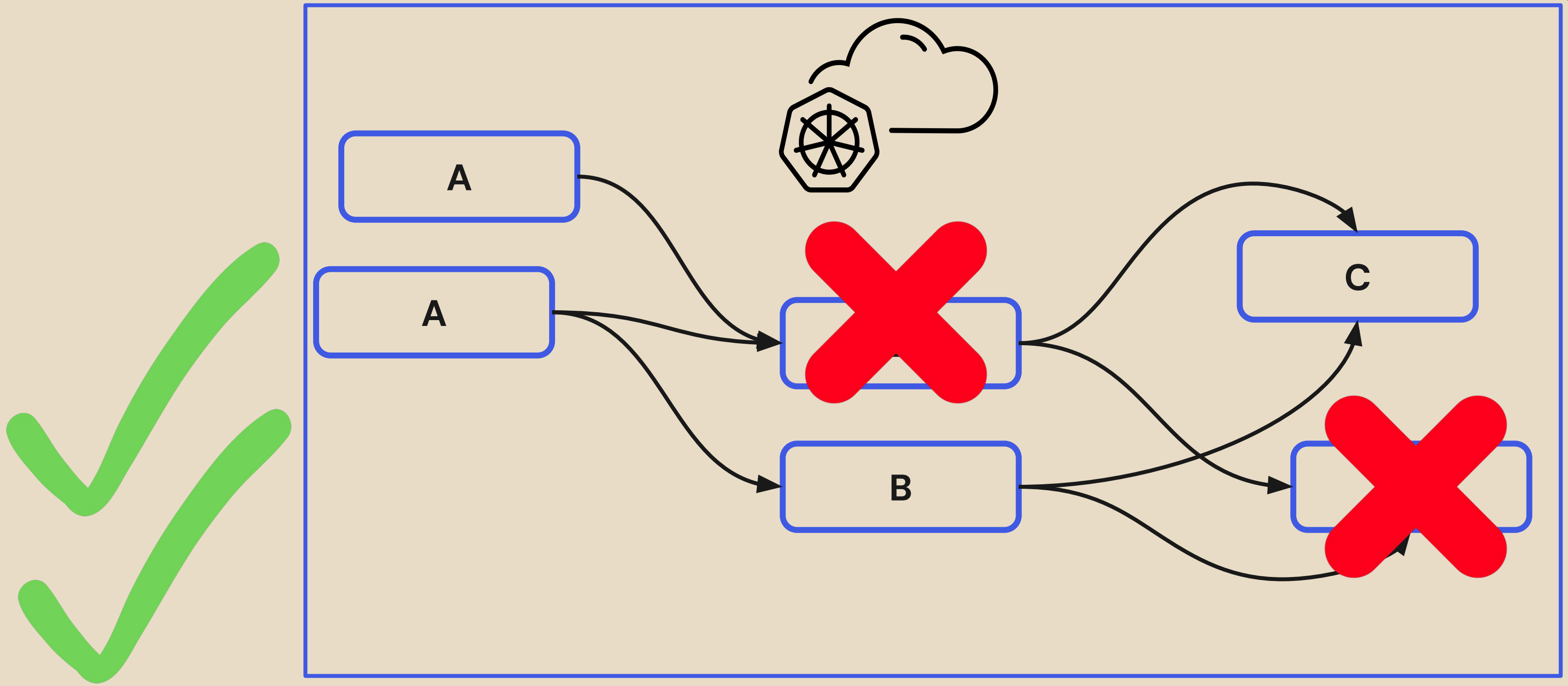
Service Dependencies





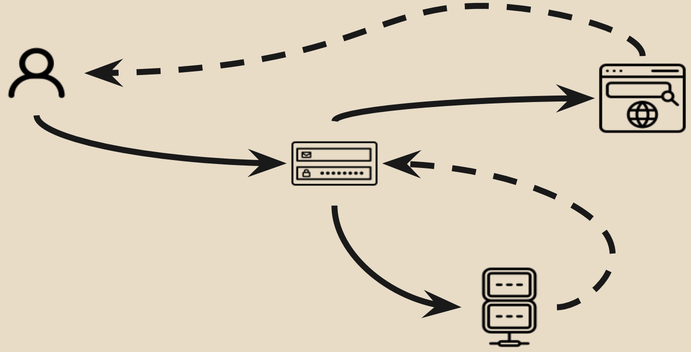
Service Dependencies

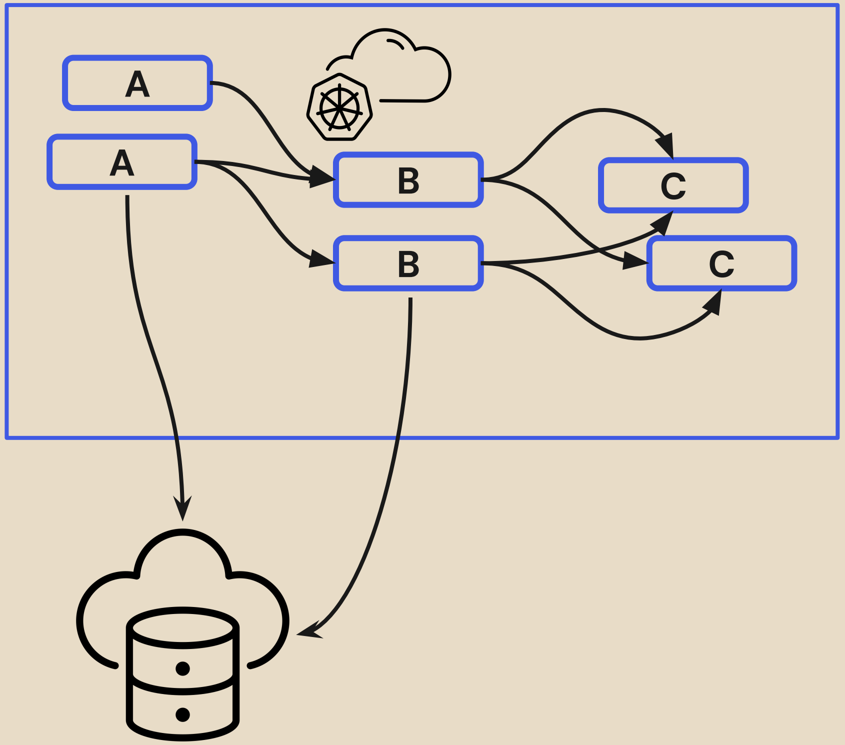




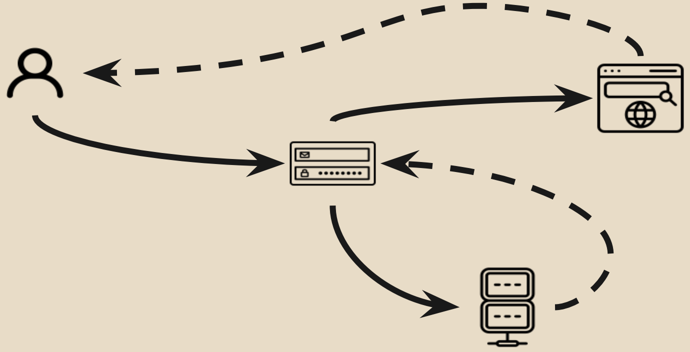
That’s all very nice, but what about external dependencies or IaaS services?

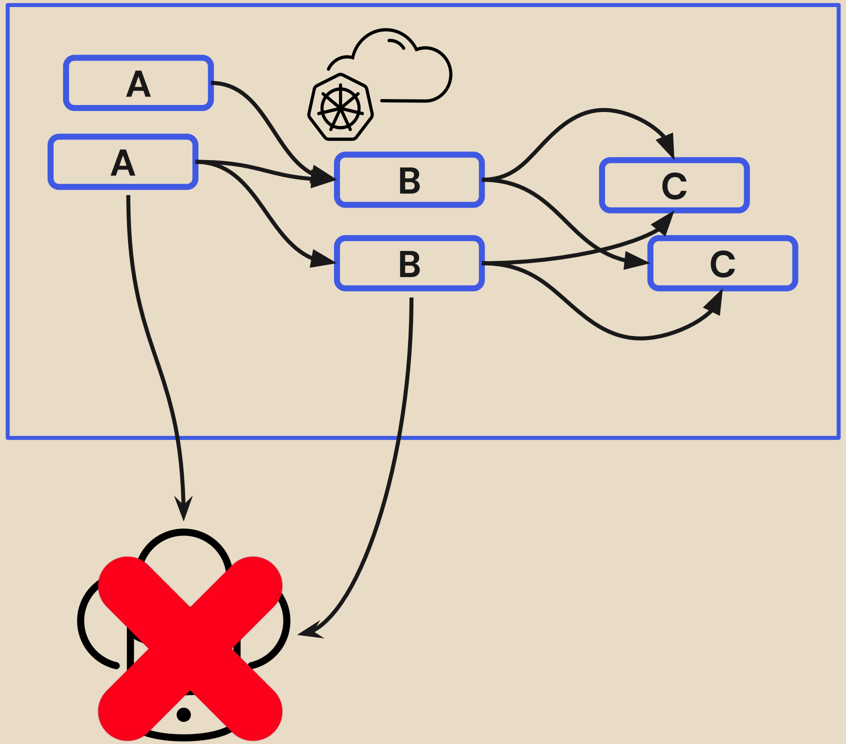
Service Dependencies



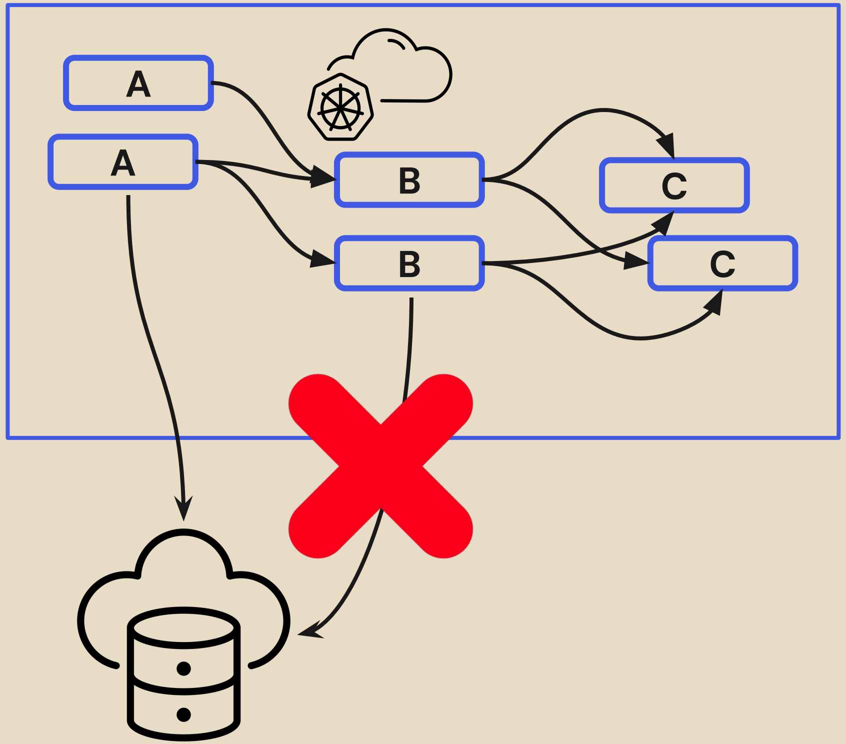


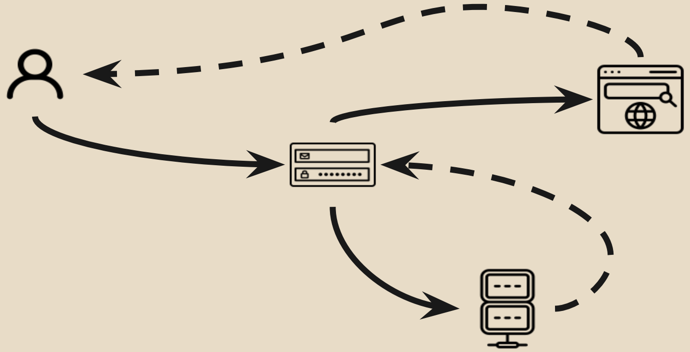
Service Dependencies





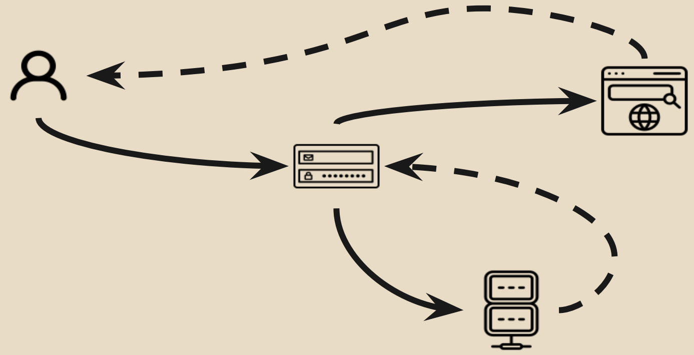
Service Dependencies

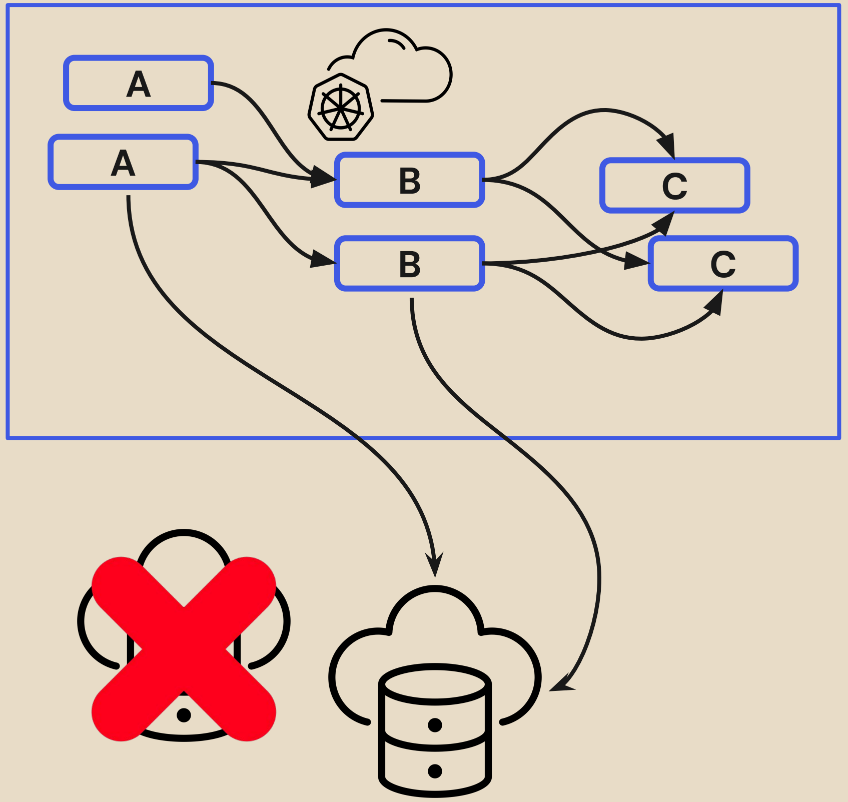




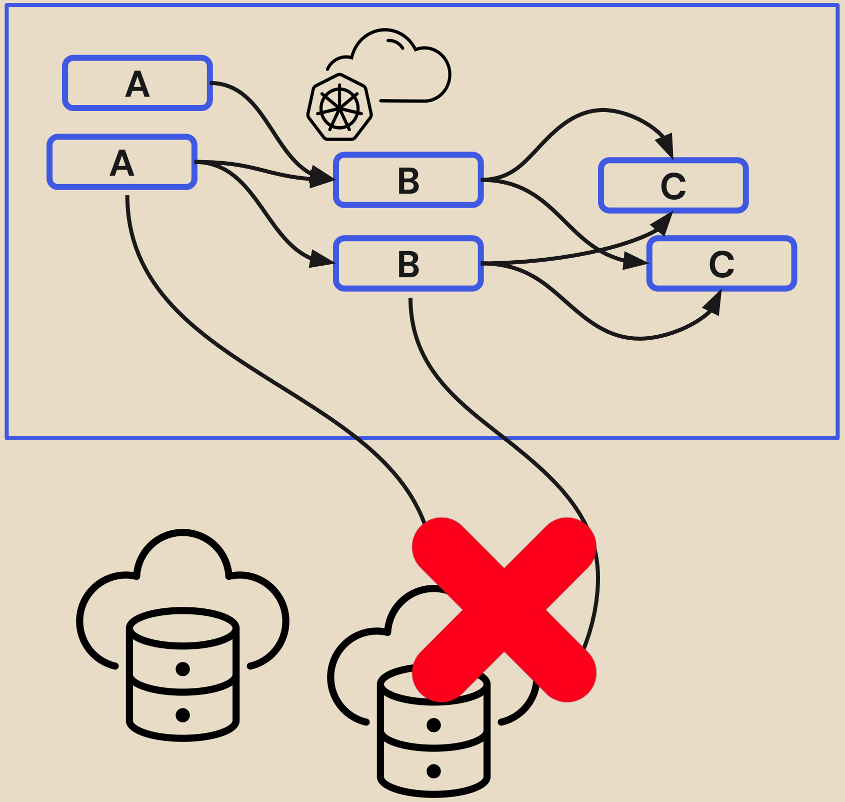
okta

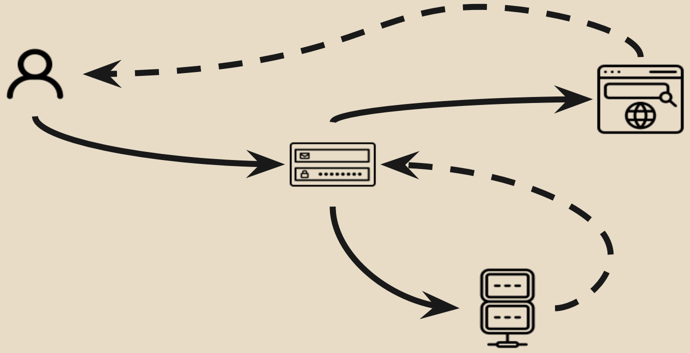
Service Dependencies





Service Dependencies





Degraded Mode

Degraded Mode is the ability your service will have to operate in a reduced capacity

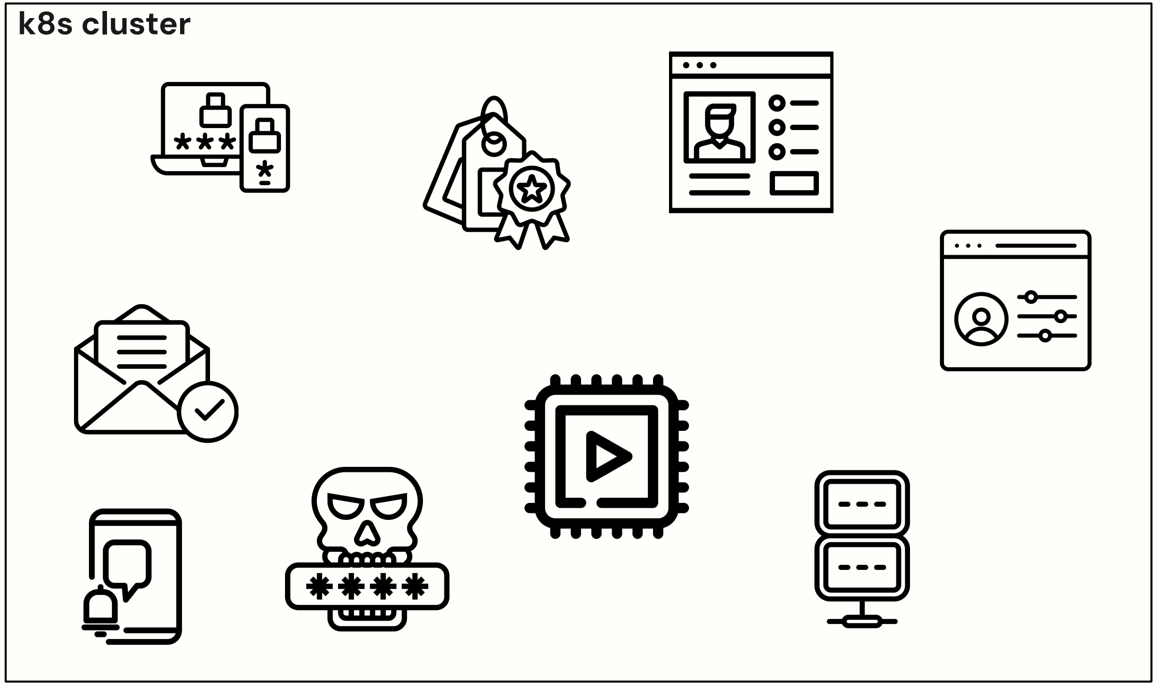
Degraded Mode is the ability your service will have to operate in a reduced capacity

- Read Only
- Longer extended latency
- Reduced set of features

okta
The World's Identity Company

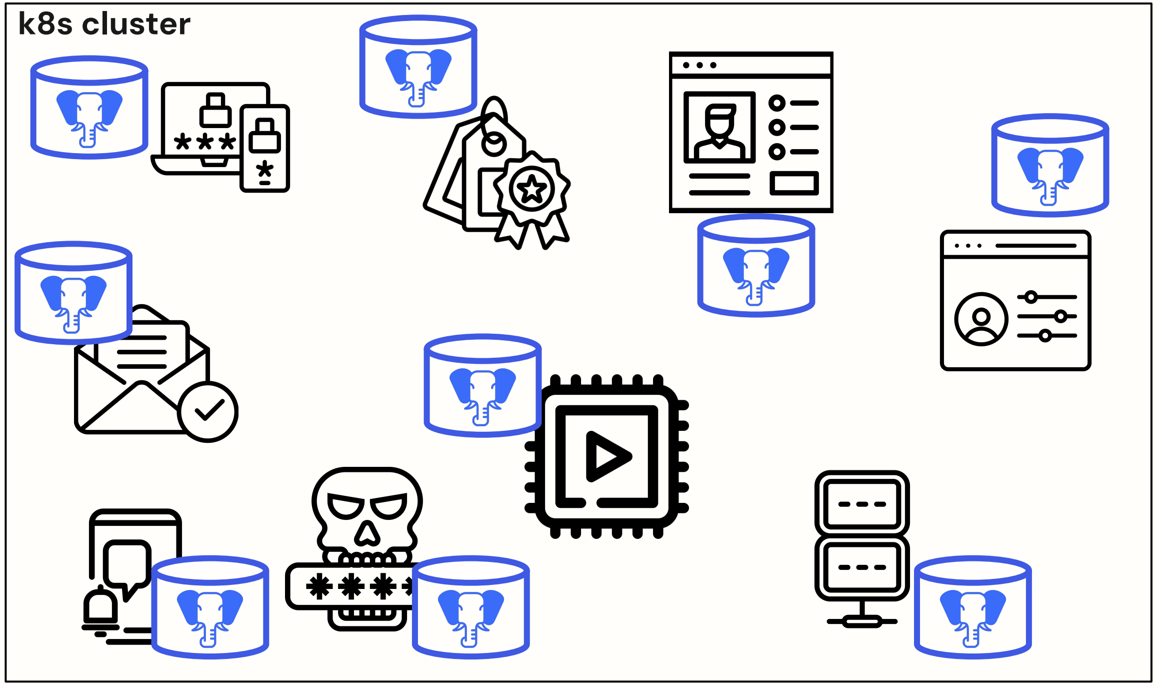
Dedicated vs Shared Database Clusters

Deployed K8s Cluster



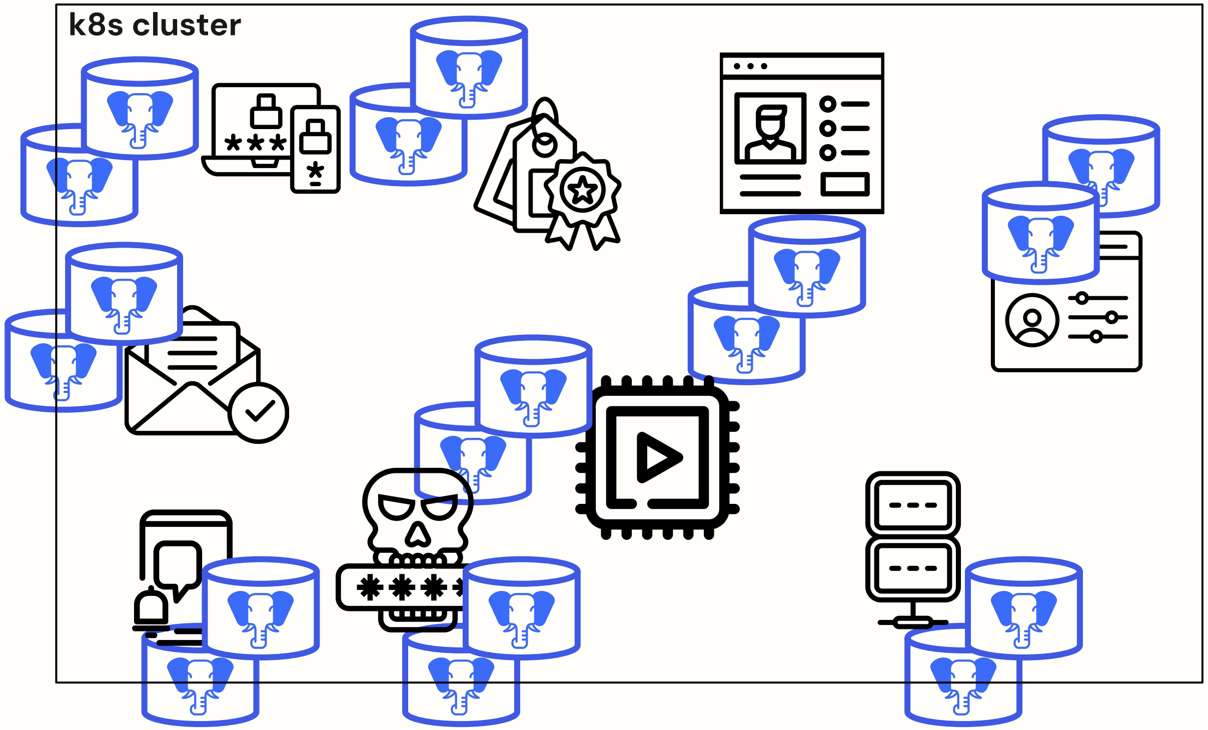
Dedicated Database Clusters

Every service gets its own database



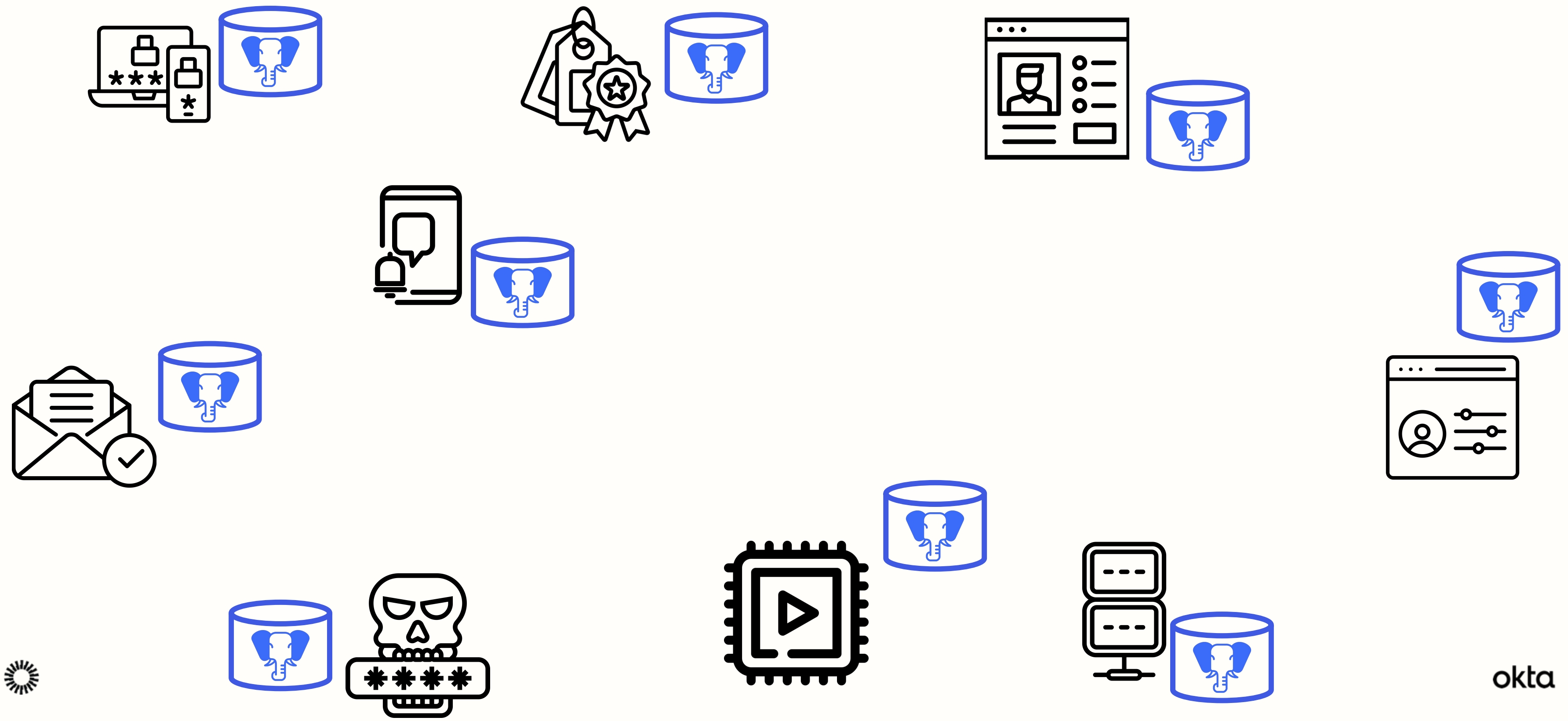
Dedicated Database Clusters

... redundantly deployed obviously



Dedicated Database Cluster

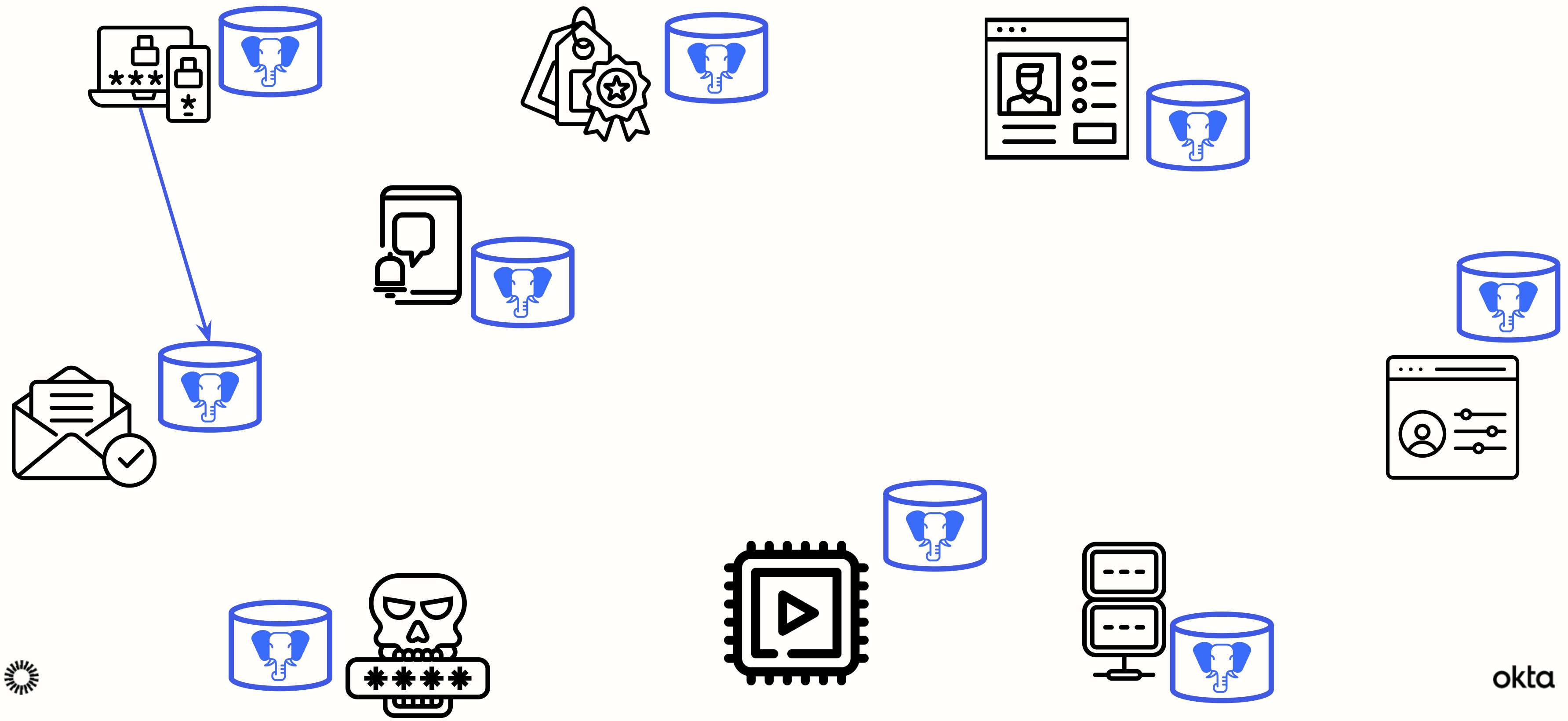
And only talks directly to that cluster



okta

Cross service access to different databases

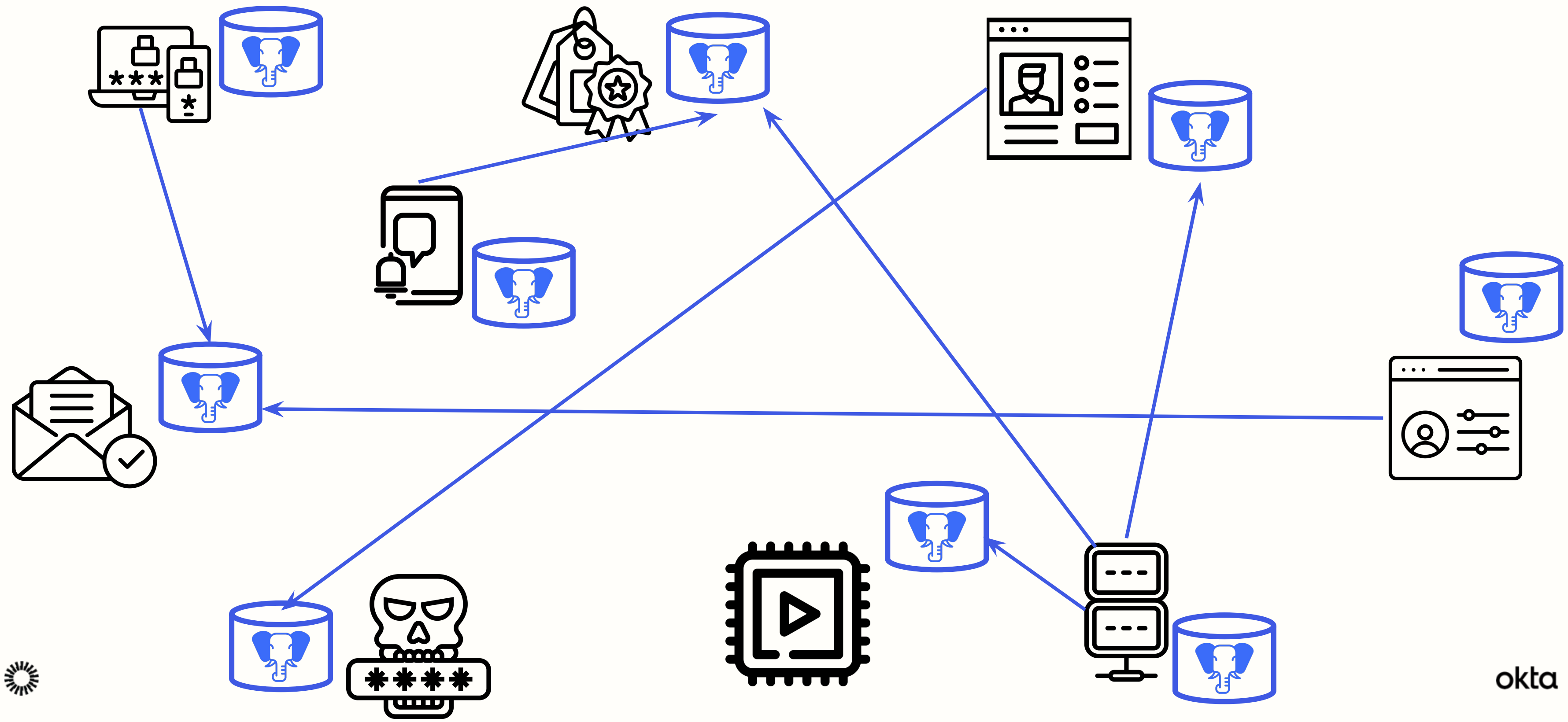
This never happens



okta

Cross service access to different databases

This never happens



okta

Yes it does happen!

okta

Cross service database access

Very hard to operate

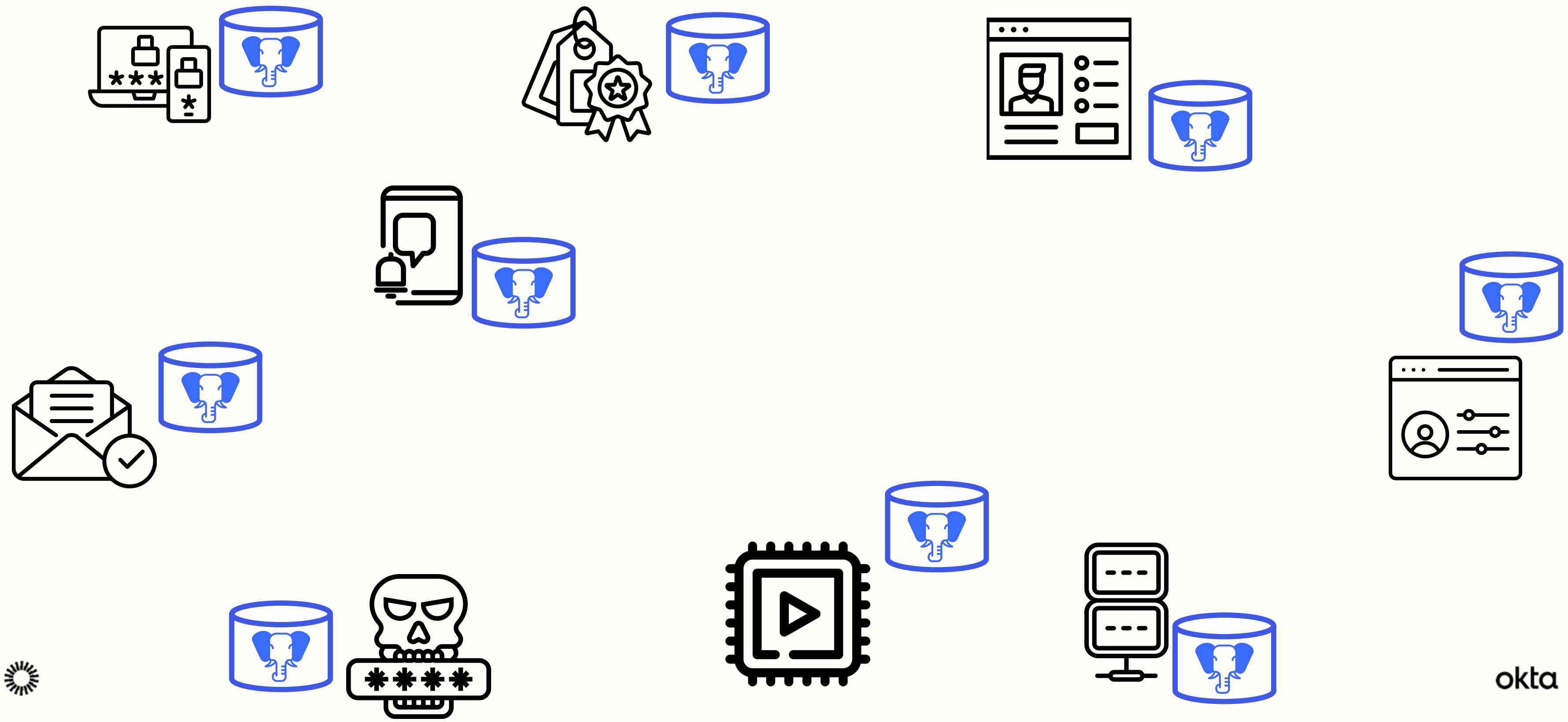
Schema Conflicts

Integration Issues

Increased Blast Radius

Dedicated Database Cluster

This is very nice to have ... however



POP QUIZ!

What is the most *common* attribute of a resilient system?

Fault Tolerant

Redundant

Scalable

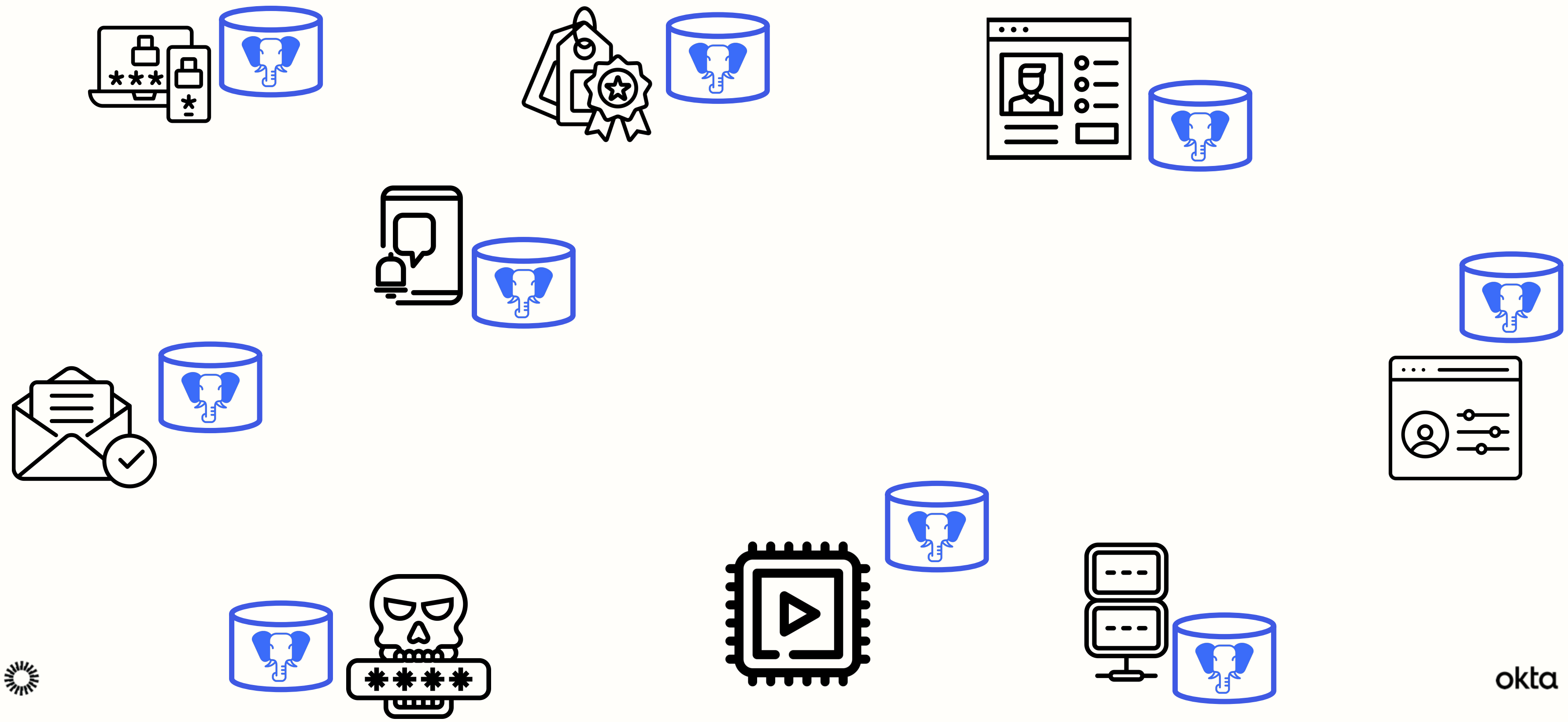
POP QUIZ!

What is the most *common* attribute of a resilient system?

~~Fault erant~~	~~Redu ant~~	~~Sca le~~

EXPENSIVE $$$

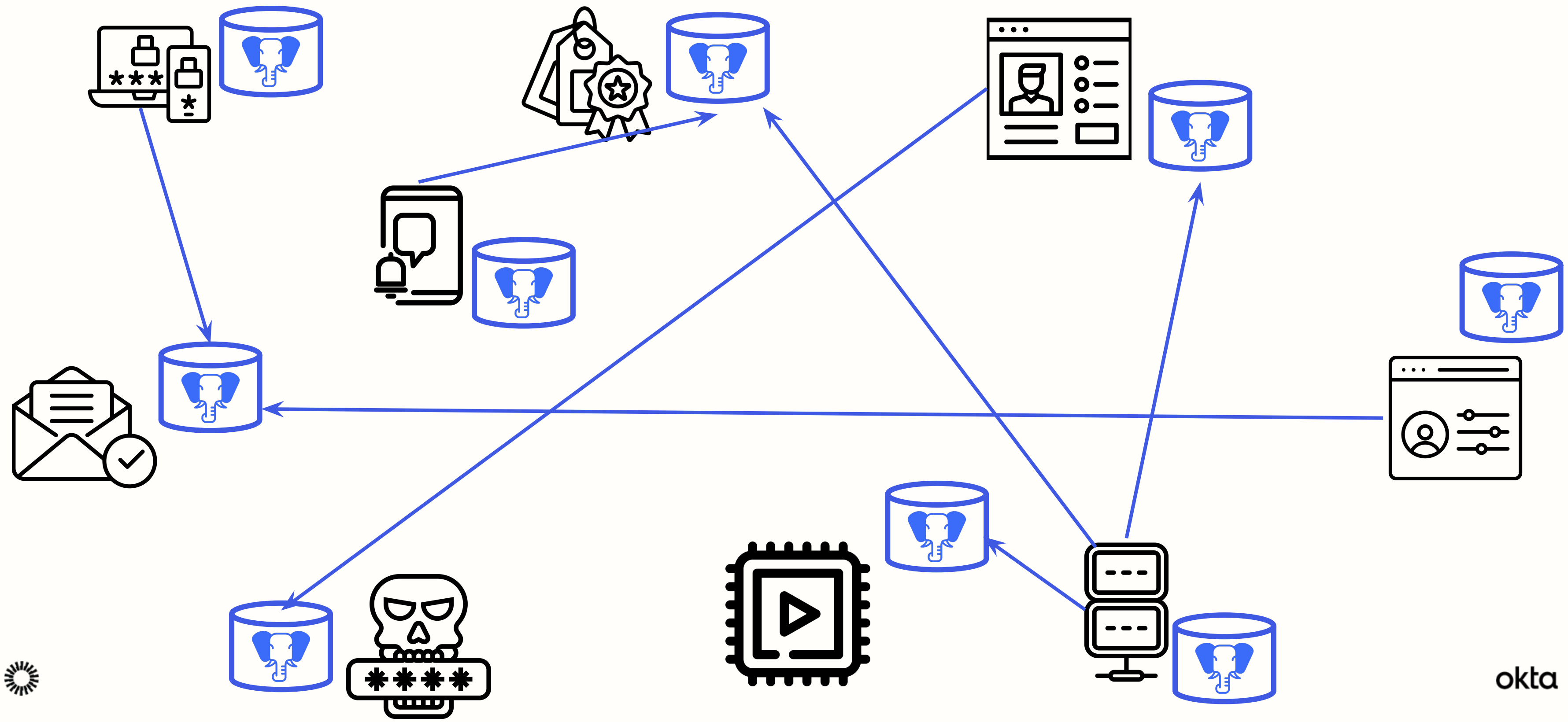
Dedicated Database Cluster - Multi-subscriber Architecture



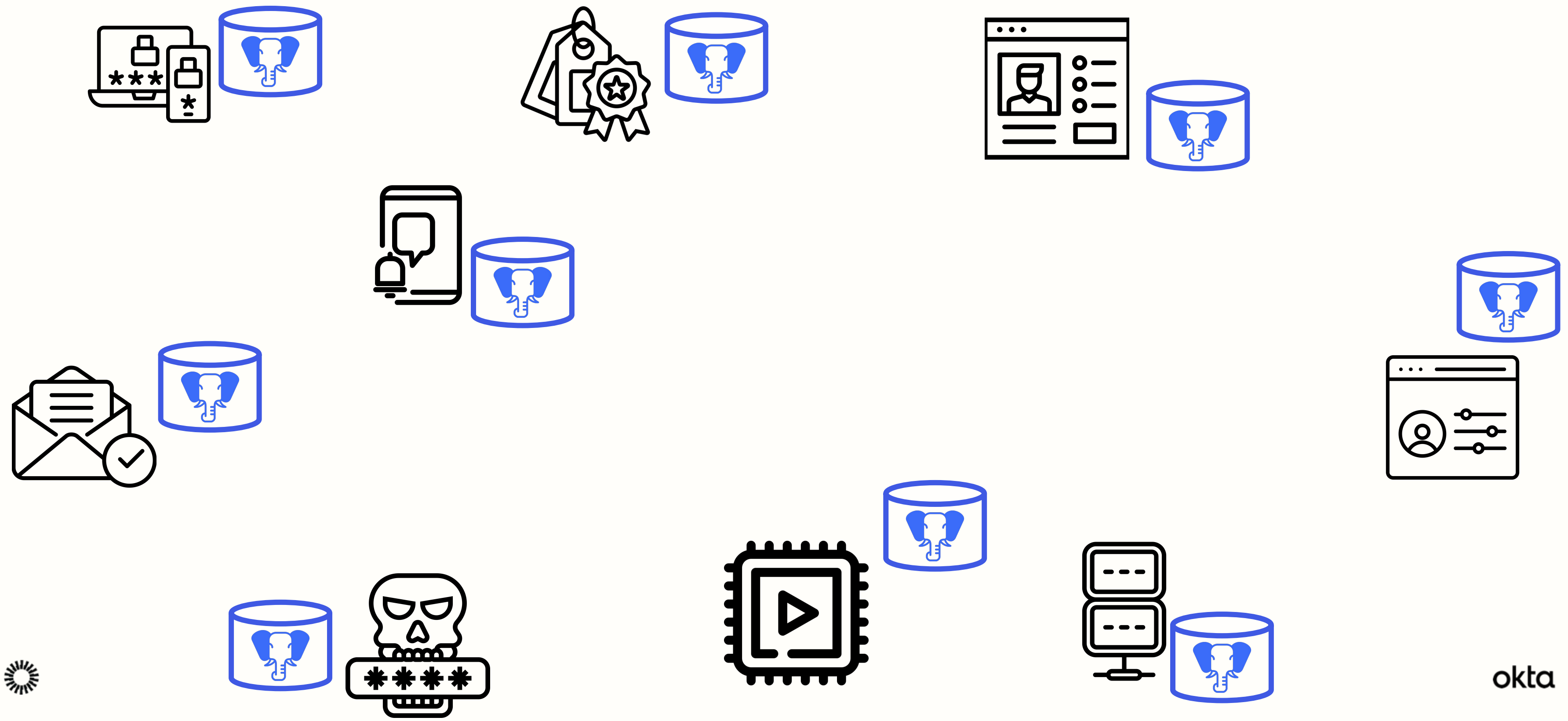
okta

We still have this

:(



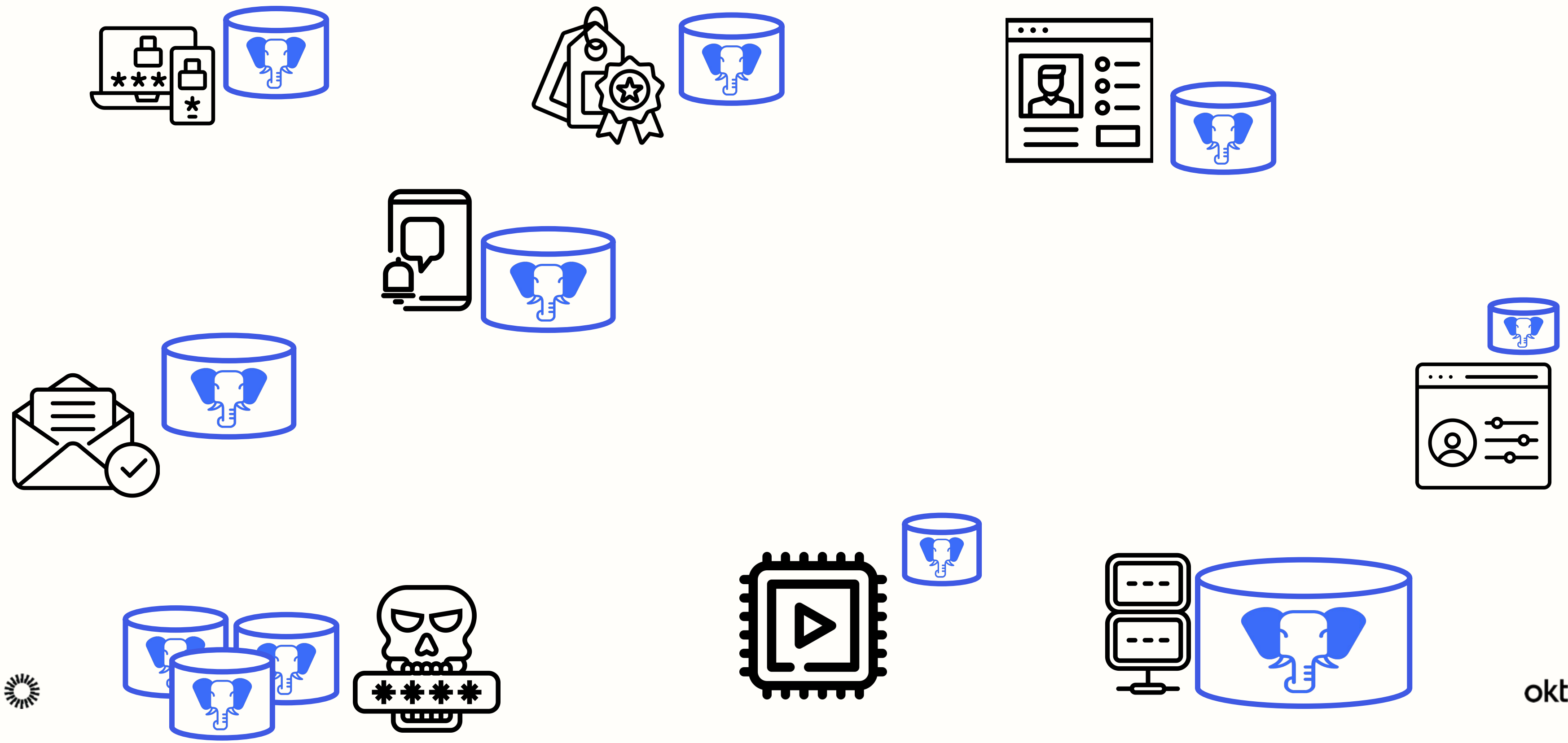
Dedicated Database Cluster - Multi-subscriber Architecture



okta

Database Clusters are sized according to needs

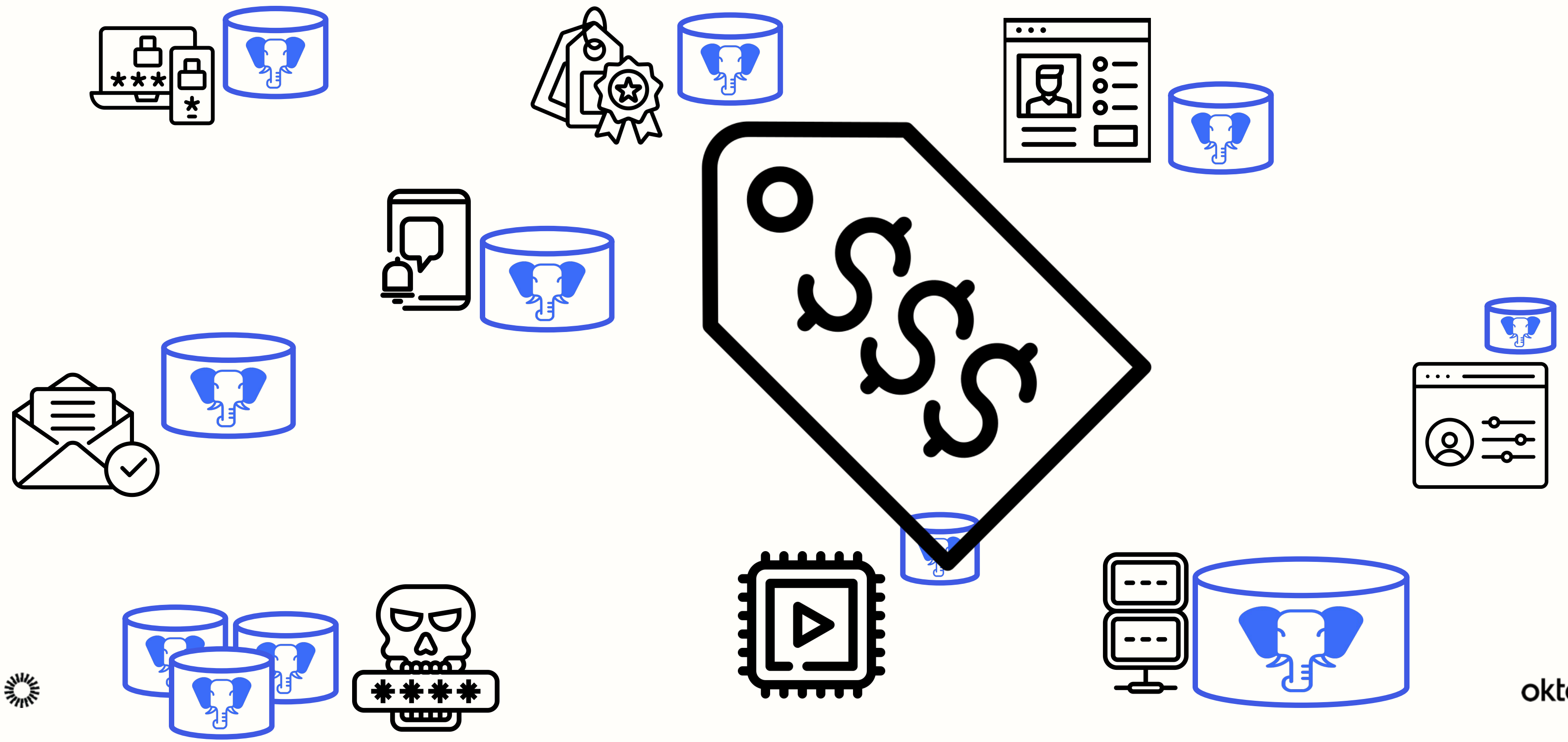
Not all databases have the same needs



okta

Dedicated Database -> Expensive Architecture

COGS need to be effectively spread across multiple tenants



Why is it expensive?

Operating asymmetric database deployments

Less standard deployment

Less predictability of workloads and monitoring

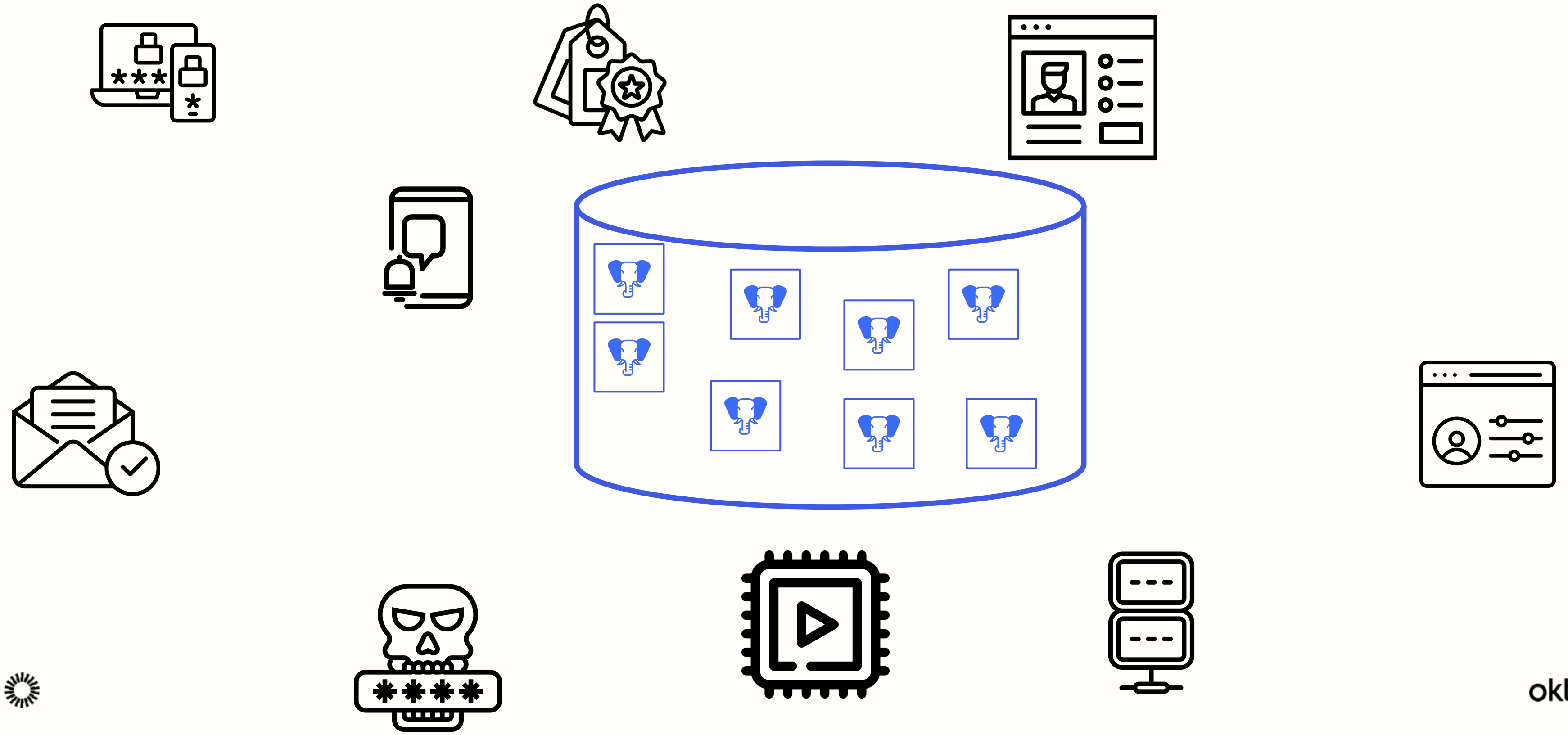
Requires more fine tuning and handholding

okta

What’s the alternative?

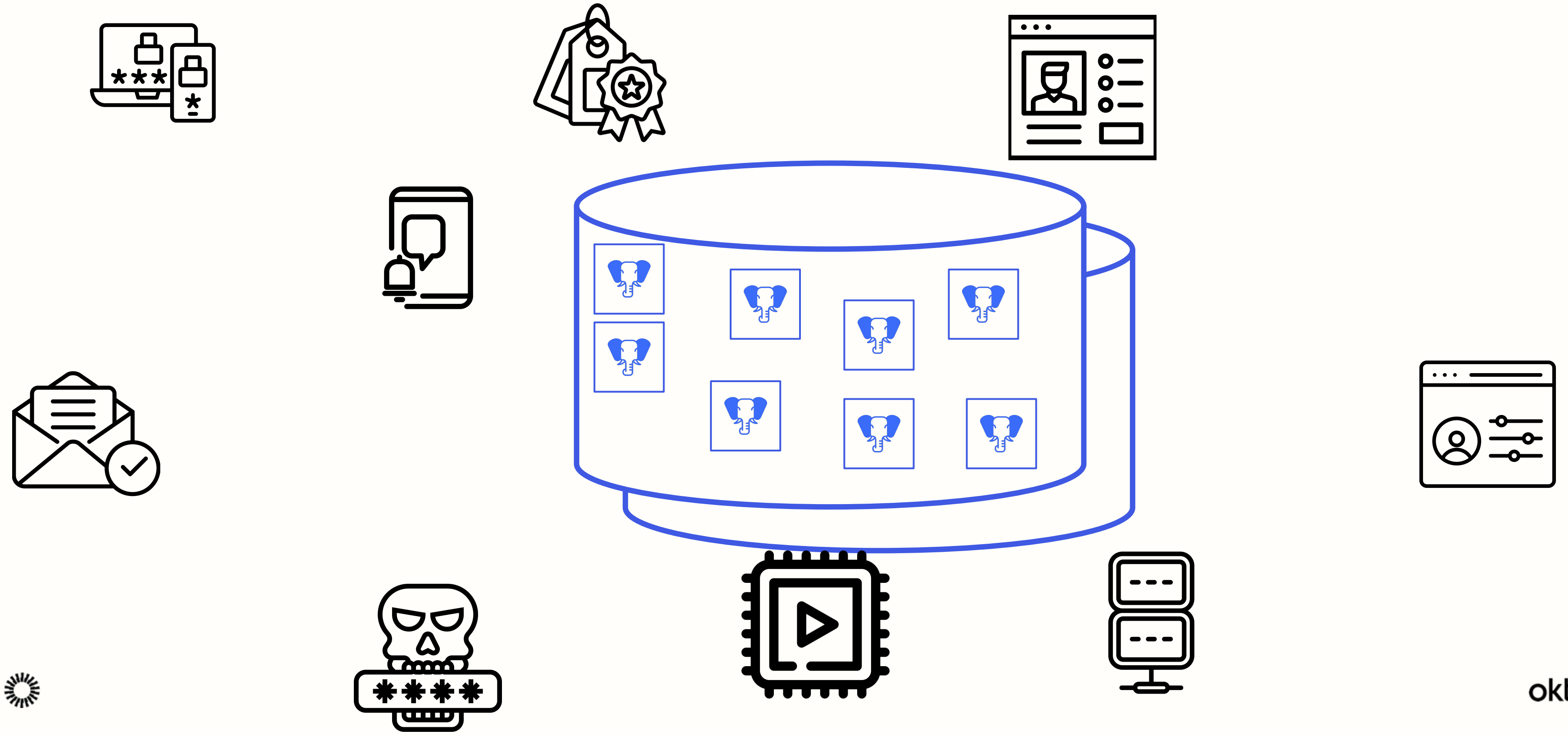
Shared Database Cluster

One that harbours all of our logical databases



Shared Database Cluster

... obviously redundant

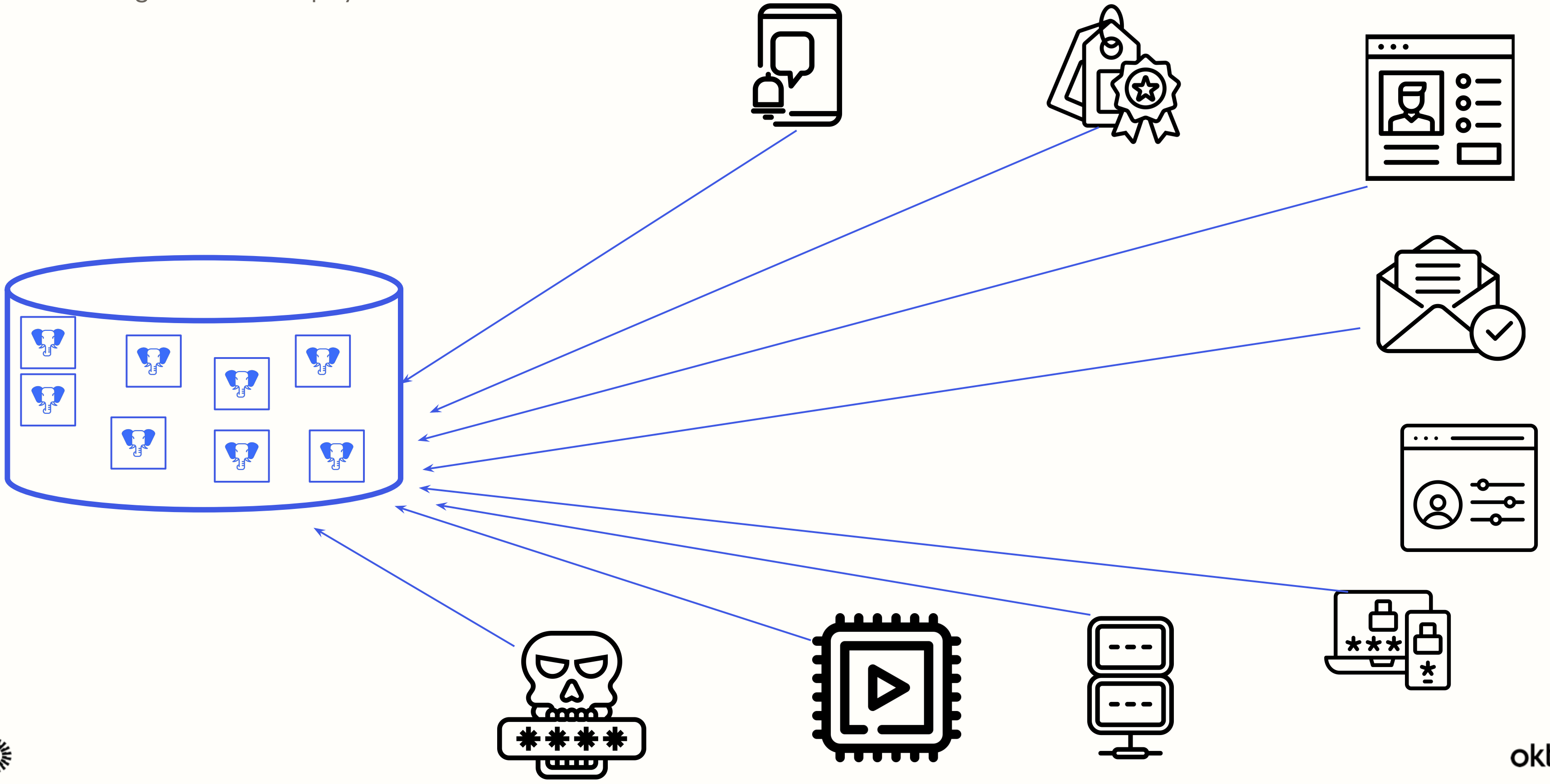


okta

Won’t this cause a similar problem as the cross database access between services?

Shared Database Cluster

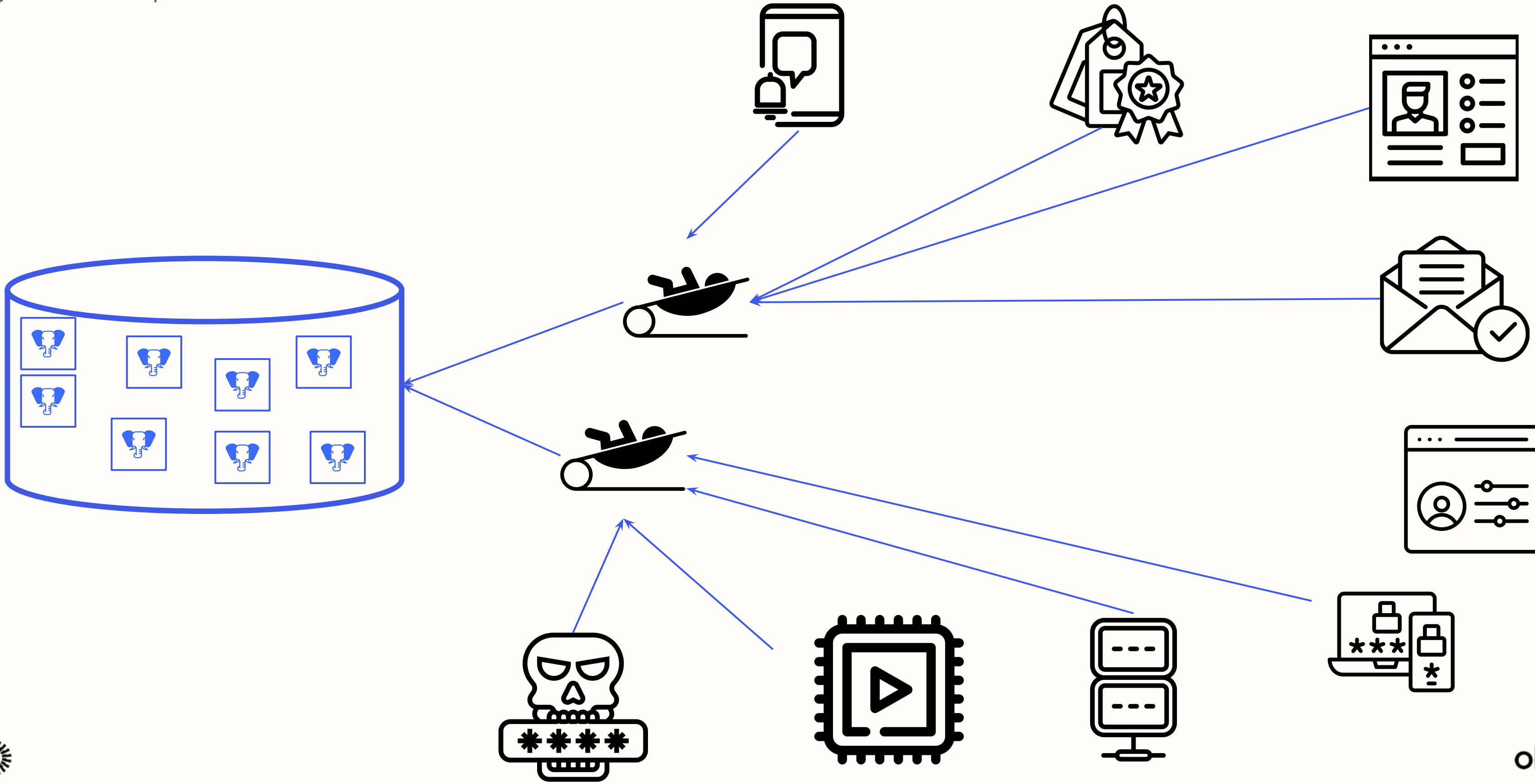
All connecting to the same physical machines



okta

Shared Database Cluster

pgbouncer helps



pgbouncer Deployments

We deploy pgbouncer as a isolated service

Dedicated per workload type reads vs writes

Good for load control – helps scaling during spikes

Shock absorber during failover events and node rotation*

So, shared cluster FTW?

Dedicated vs Shared Cluster

How we look at it

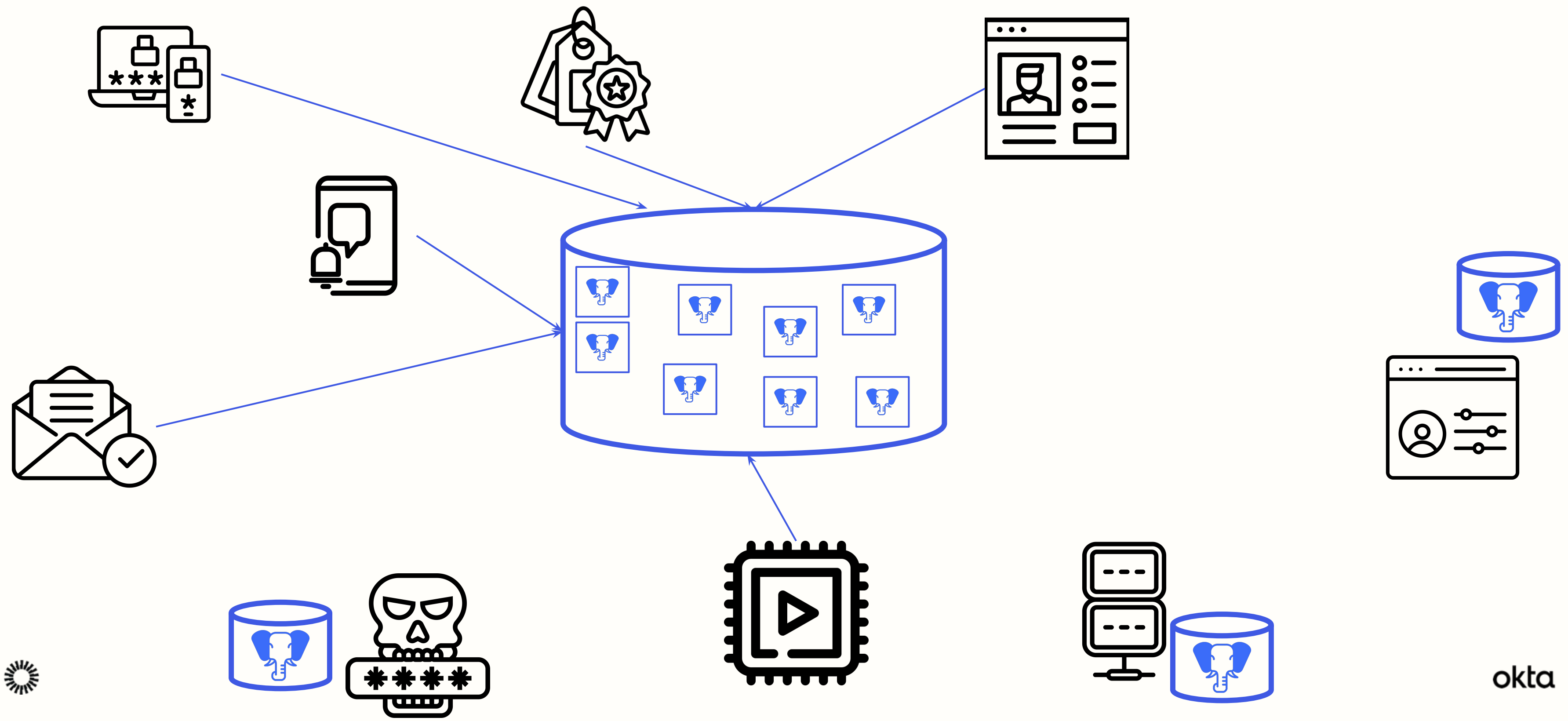
Dedicated

- Good for service isolation
- Allows for finer level control of individual services needs
- Segregation of datasets allows for different backup policies and retention controls
- Increases database nodes footprint
- Increases costs per storage unit
    - needs to be well spread across tenants
- We use it on multi-tenant deployments

Shared

- Simpler deployment
- Increased Blast Radius
    - If database goes down, all services go down
- “Easier” to rollout changes
- Suffers from noisy neighbour
- Preferred deployment for single-tenant deployments

Tier Based Architecture



Dedicated + Shared Cluster

Merging the best of both worlds

Rules

- Shared Cluster
  - similar lifecycle
  - same backup retention policy
  - same service critical tier
  - default
- Dedicated Cluster
  - demanding workloads – the noisy folk
  - address scaling needs
  - limit resource starvation by single service
  - cost effective to scale out

Auth0 by Okta

Database Management at Scale

Laundry List of Database Problems at Scale

- Bad indexes
- No indexes
- Bad schema migration
- Locking ALTER TABLE
- AUTO-VACUUM
- Extensions OOM
- Manual scripts
- Postgres Major Version Upgrades
- Self-served DDoS
- Cache fallback DoS
- TRIGGERS
- ……

Database at Scale

The joy of databases

Single vs Multi-tenant Indexes

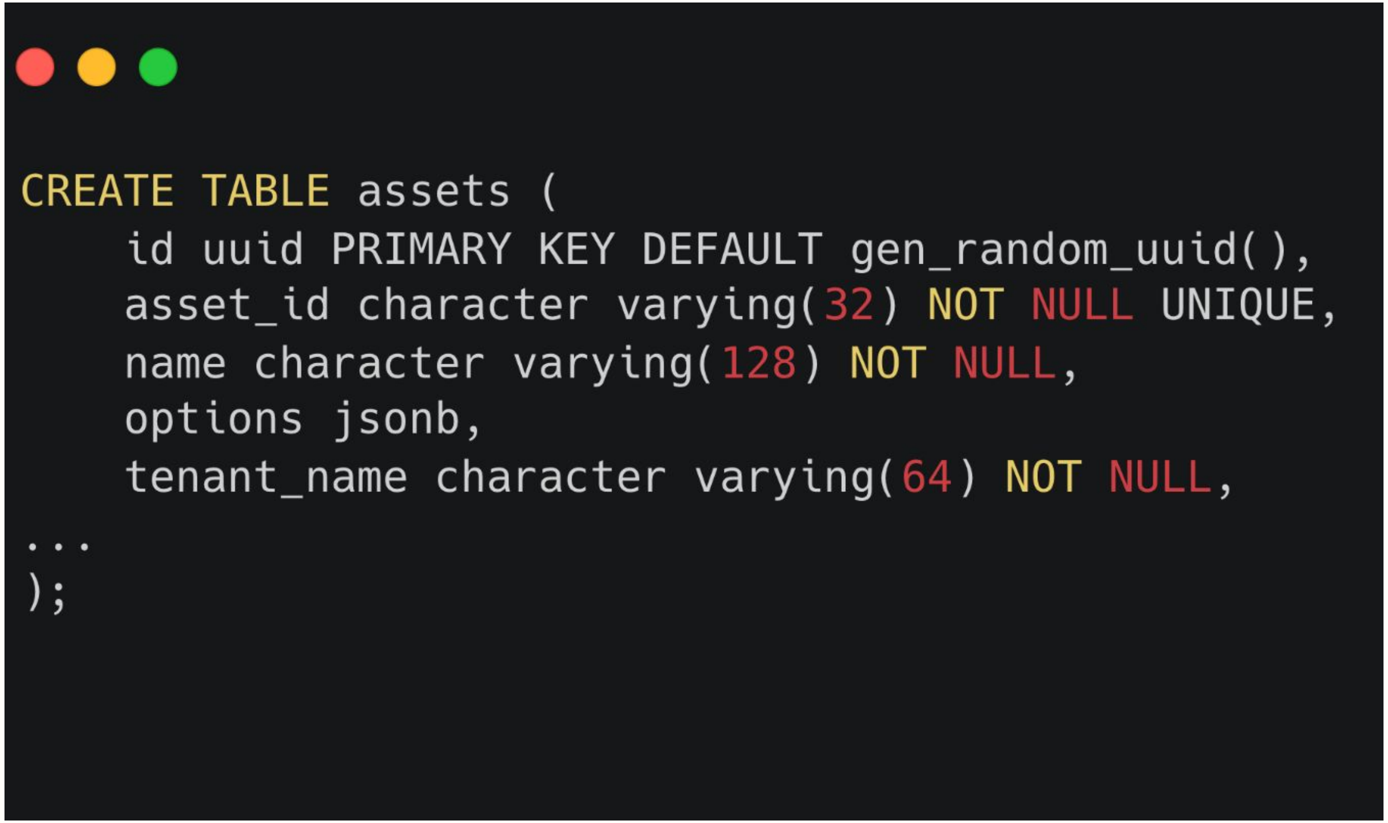
Auditing

Postgres Schemas

Database problems manifest themselves at scale in unexpected ways!

Single vs Multi-tenant on Indexes

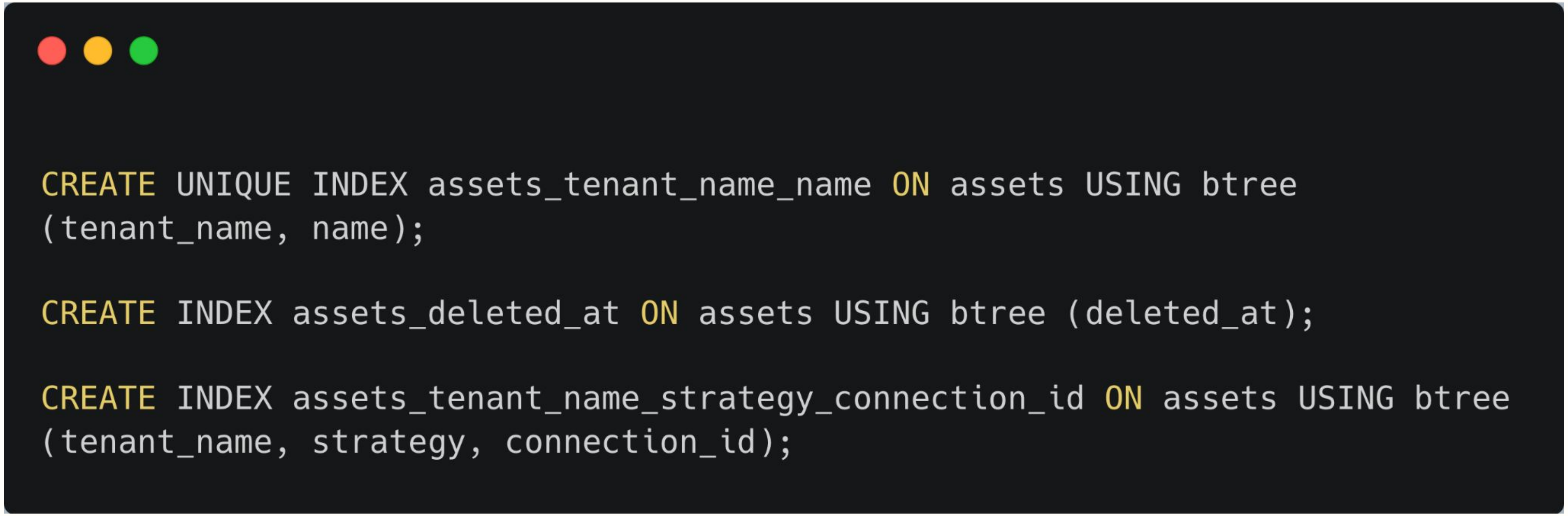
The joys of making all things the same



```sql
CREATE TABLE assets (
    id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
    asset_id character varying(32) NOT NULL UNIQUE,
    name character varying(128) NOT NULL,
    options jsonb,
    tenant_name character varying(64) NOT NULL,
...
);
```

Single vs Multi-tenant on Indexes

The joys of making all things the same



```
CREATE UNIQUE INDEX assets_tenant_name_name ON assets USING btree
(tenant_name, name);

CREATE INDEX assets_deleted_at ON assets USING btree (deleted_at);

CREATE INDEX assets_tenant_name_strategy_connection_id ON assets USING btree
(tenant_name, strategy, connection_id);
```

okta

Single vs Multi-tenant on Indexes

The joys of making all things the same

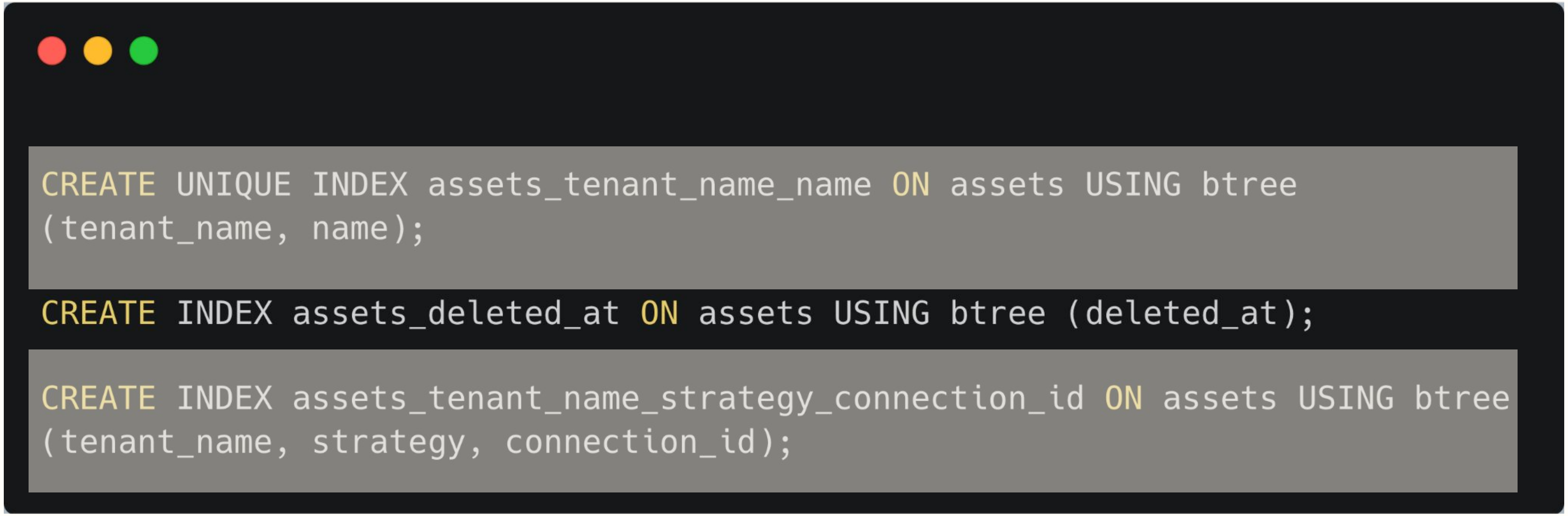
Single Tenant	
rownum	tenant_name
1	pgconfeu
2	pgconfeu
3	pgconfeu
4	pgconfeu
5	pgconfeu
6	pgconfeu
7	pgconfeu
8	pgconfeu
9	pgconfeu
10	pgconfeu

Multi Tenant	
rownum	tenant_name
1	pgconfeu
2	devdays
3	pgconfus
4	jp user group
5	fosdem
6	devox
7	jfocus
8	pgconfeu
9	pgconfeu
10	pgconfeu

okta

Single vs Multi-tenant on Indexes

The joys of making all things the same



```
CREATE UNIQUE INDEX assets_tenant_name_name ON assets USING btree
(tenant_name, name);

CREATE INDEX assets_deleted_at ON assets USING btree (deleted_at);

CREATE INDEX assets_tenant_name_strategy_connection_id ON assets USING btree
(tenant_name, strategy, connection_id);
```

single vs multi tenant

Indexes may not behave in the same way as you expect them to behave.

Different indexed values cardinality impacts usage

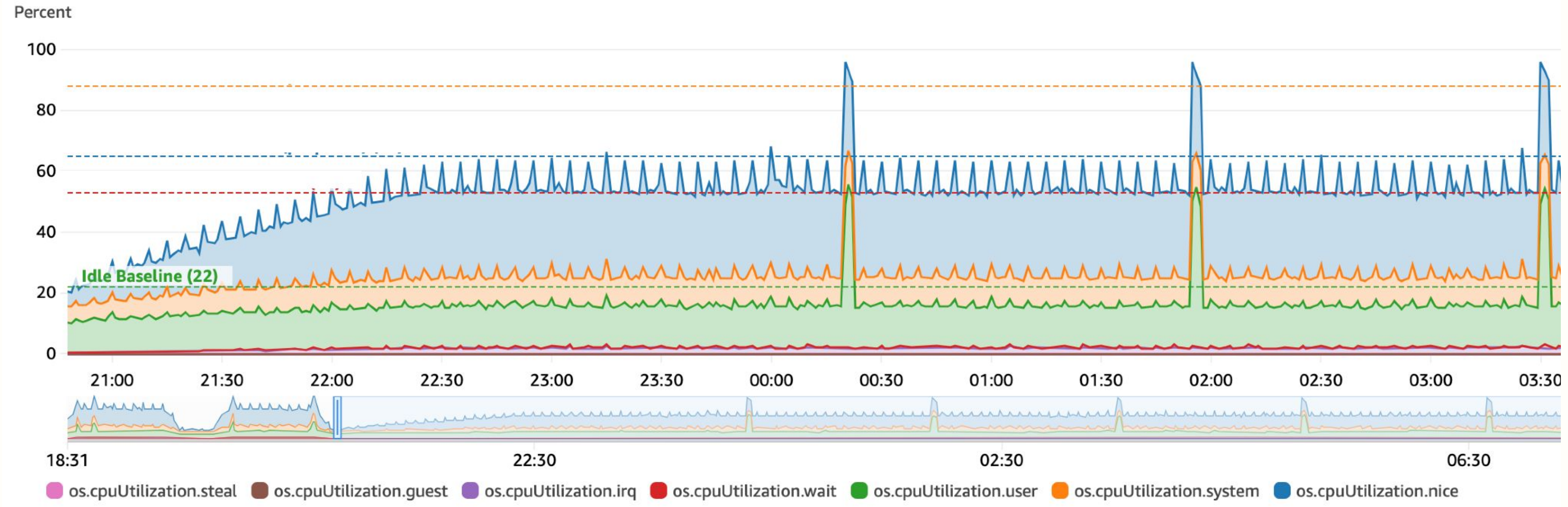
Vacuum will have a different profile

Bloat will be impacted by workload and cardinality

okta

Crown of Horns

What's up with those recurrent spikes?

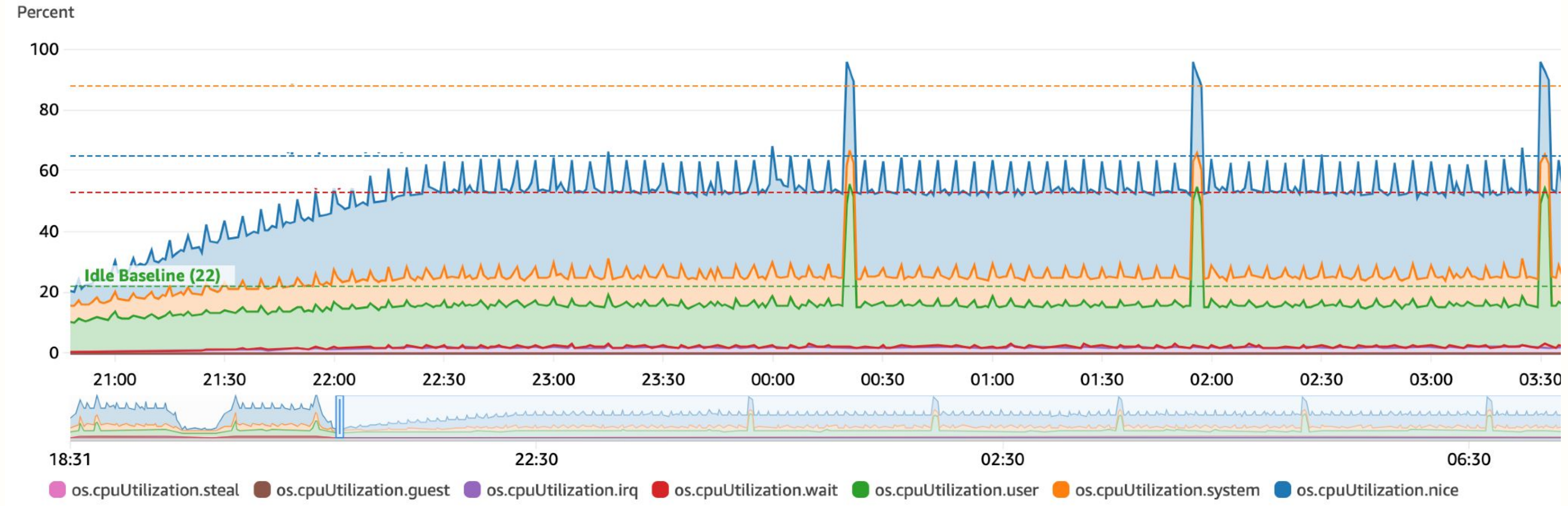


okta

Is DB Auditing a free lunch?

Audit Logs

Still needs to be processed



okta

Audit logs

No such thing as a free lunch

Recurrent internal DB processes need to be checked	Move all non operational workloads outside DB	Test triggers and stored procedures at scale

Postgres Schemas

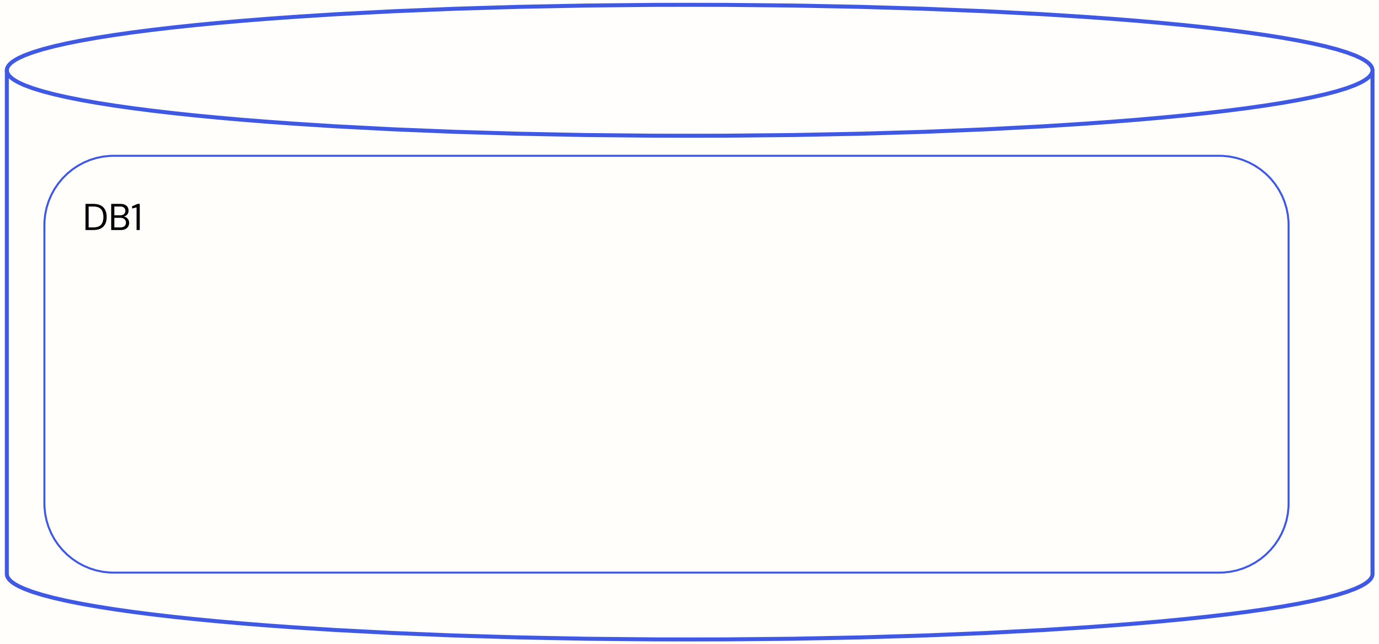
In PostgreSQL, a schema is a named collection of database objects, including tables, views, indexes, data types, functions, stored procedures, and operators.

A schema allows you to organize and namespace database objects within a database.

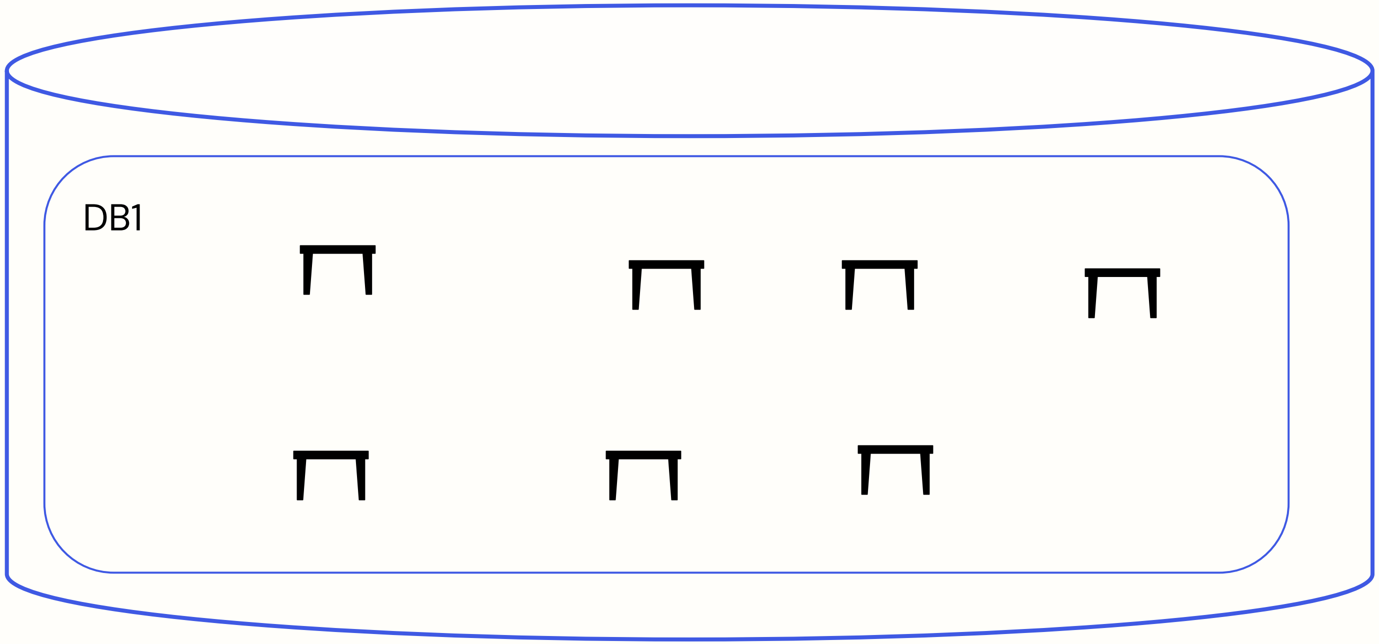
A database may contain one or more schemas. However, a schema belongs to only one database. Additionally, two schemas can have different objects that share the same name.

https://neon.tech/postgresql/postgresql-administration/postgresql-schema

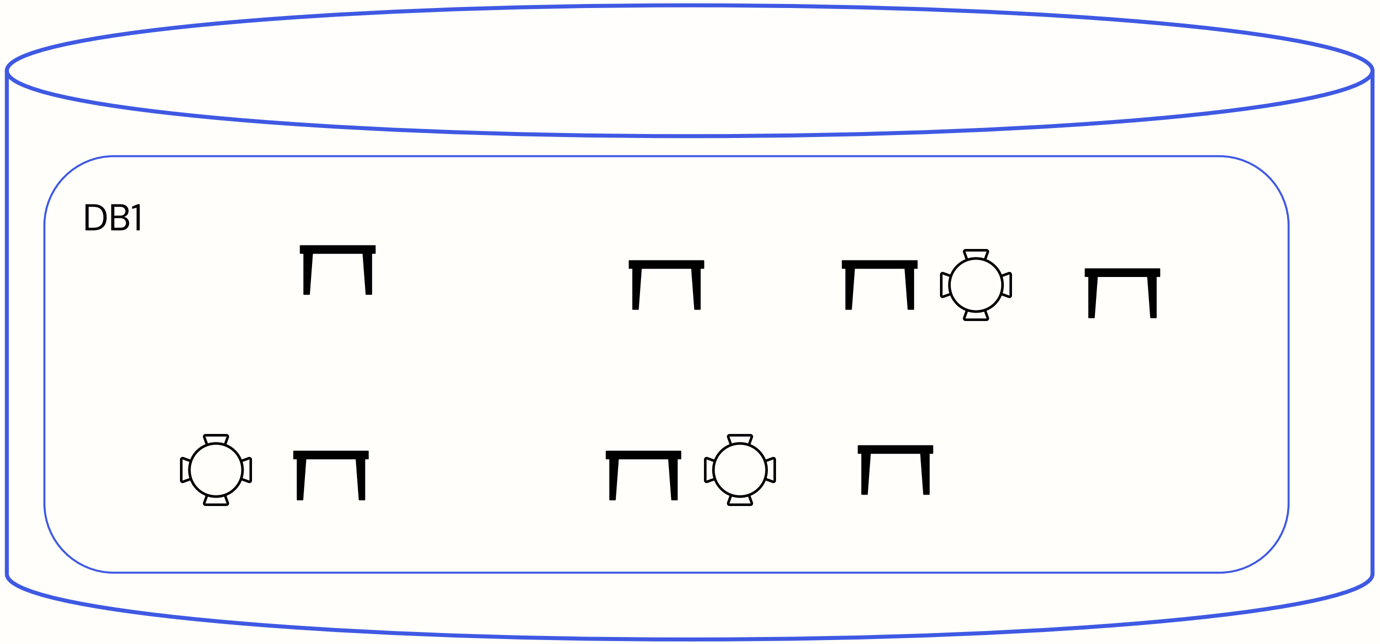
Postgres Schemas



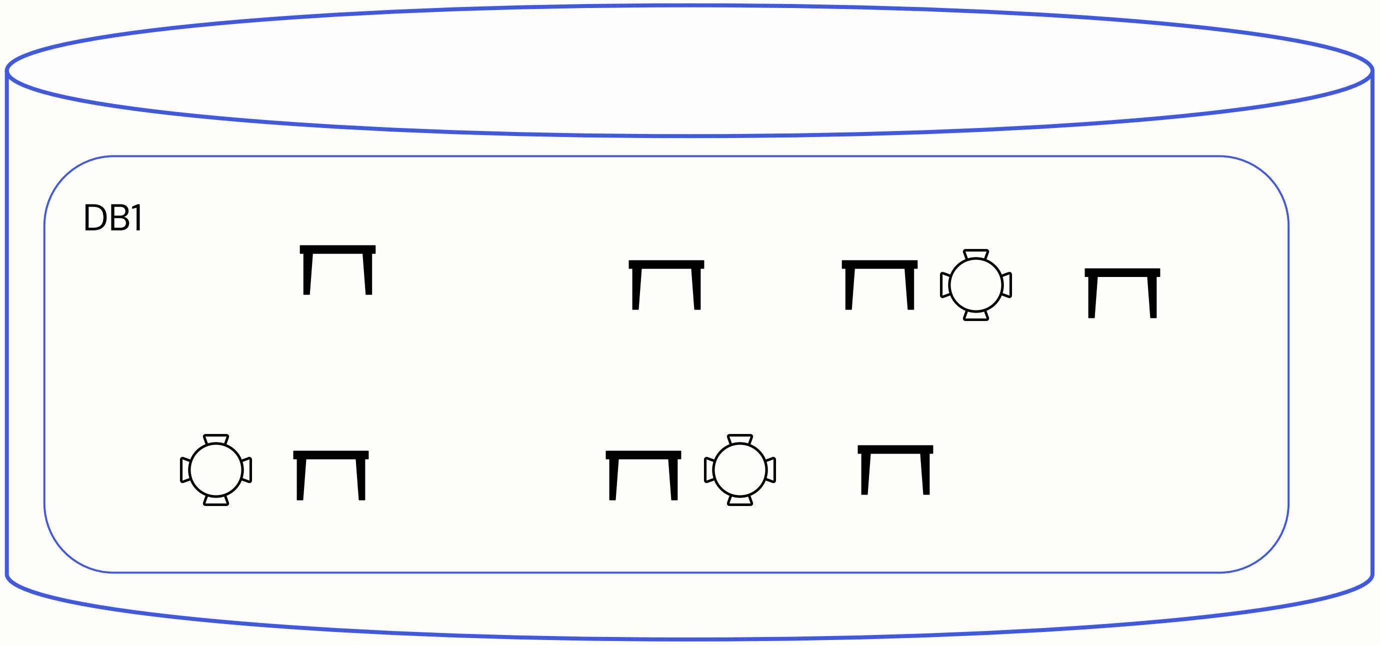
Postgres Schemas



Postgres Schemas

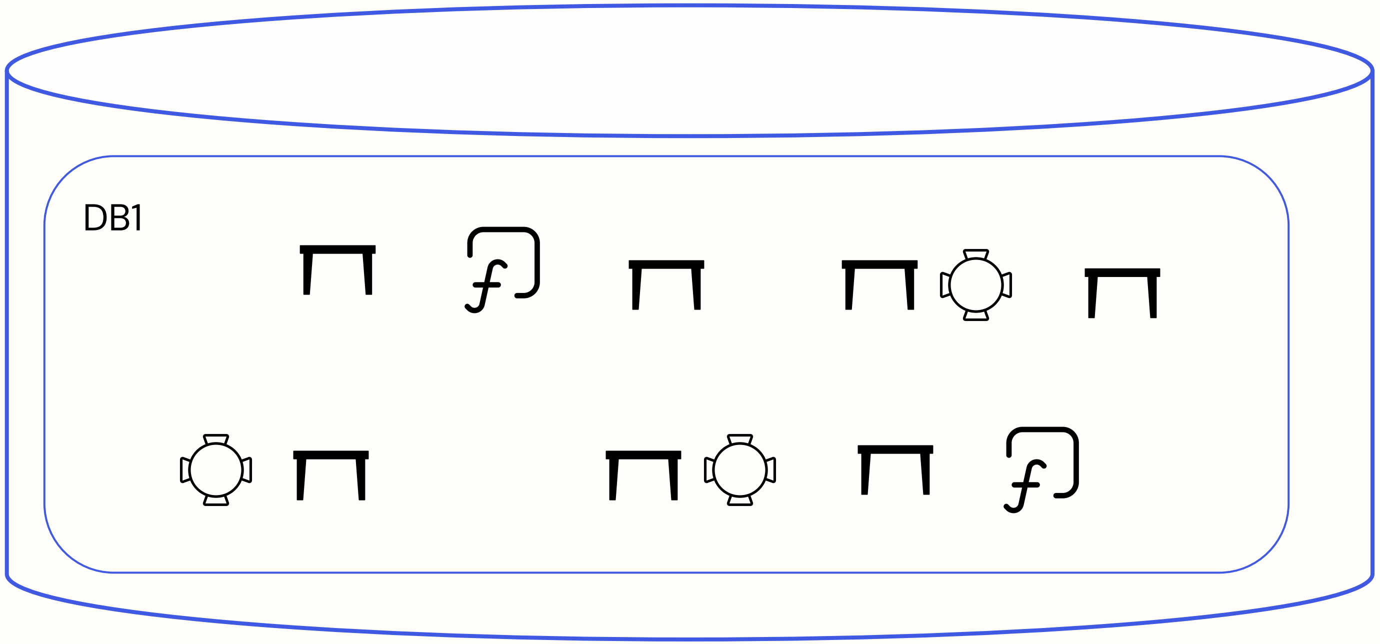


Postgres Schemas

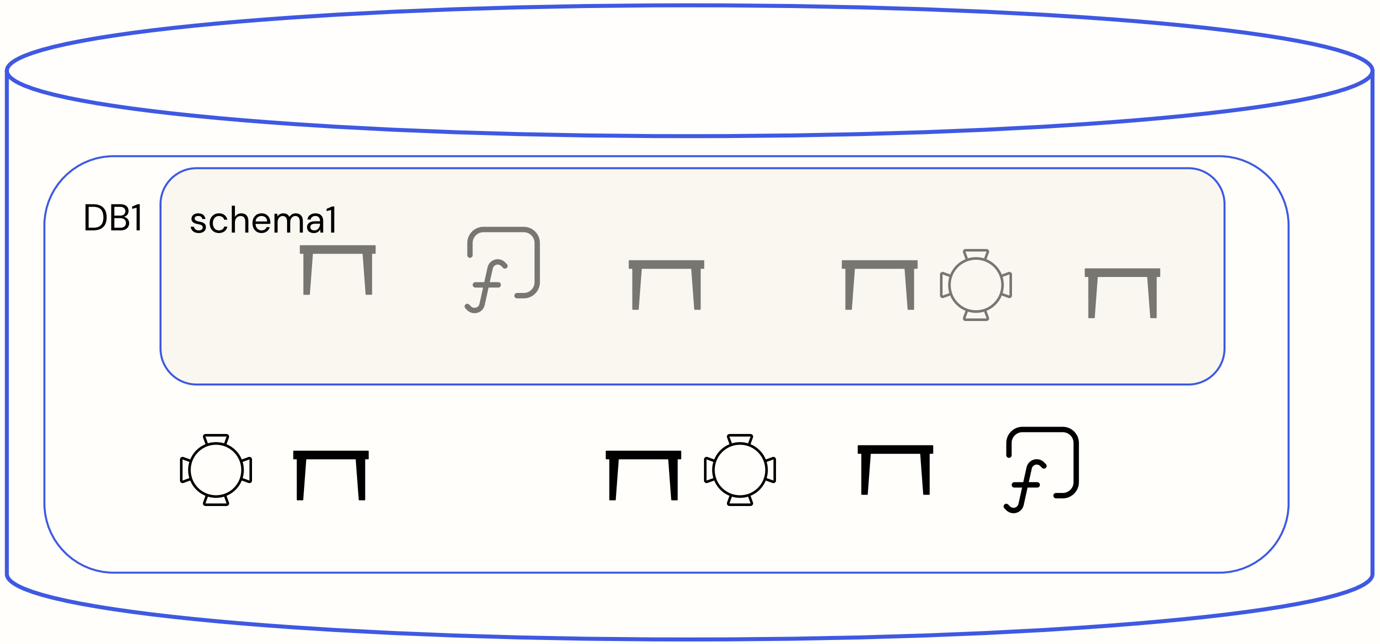


okta

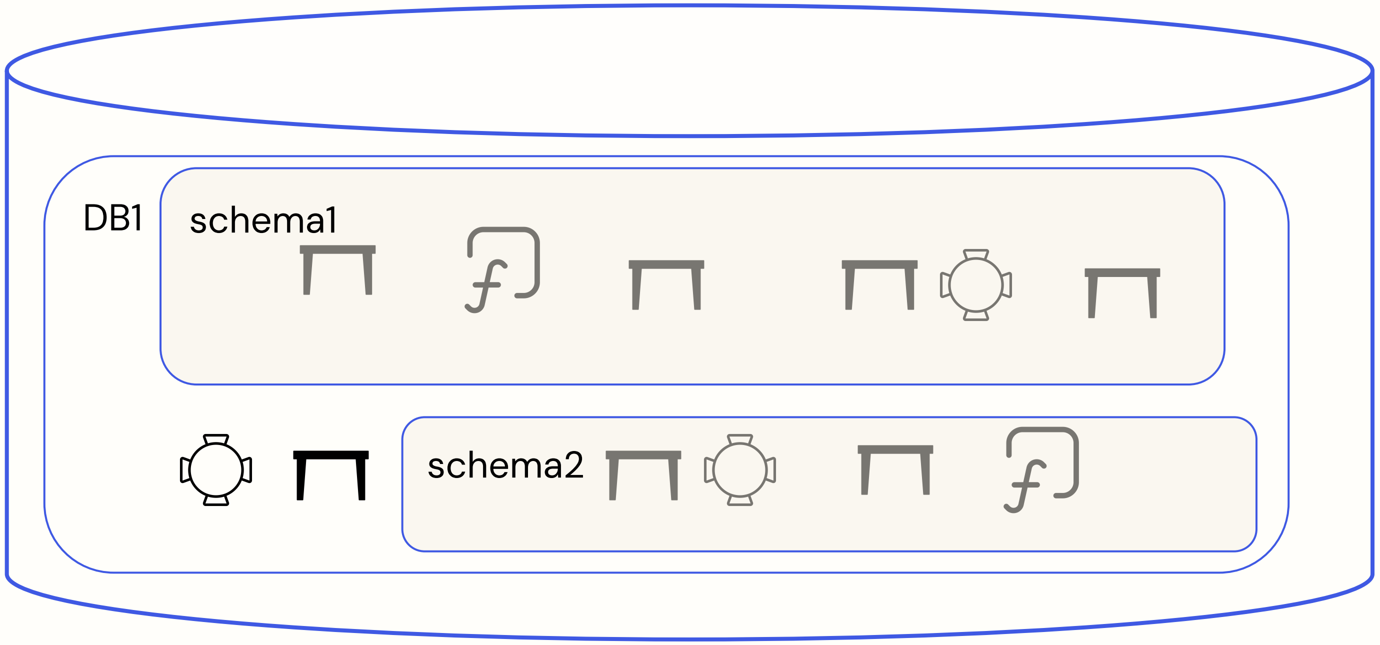
Postgres Schemas



Postgres Schemas

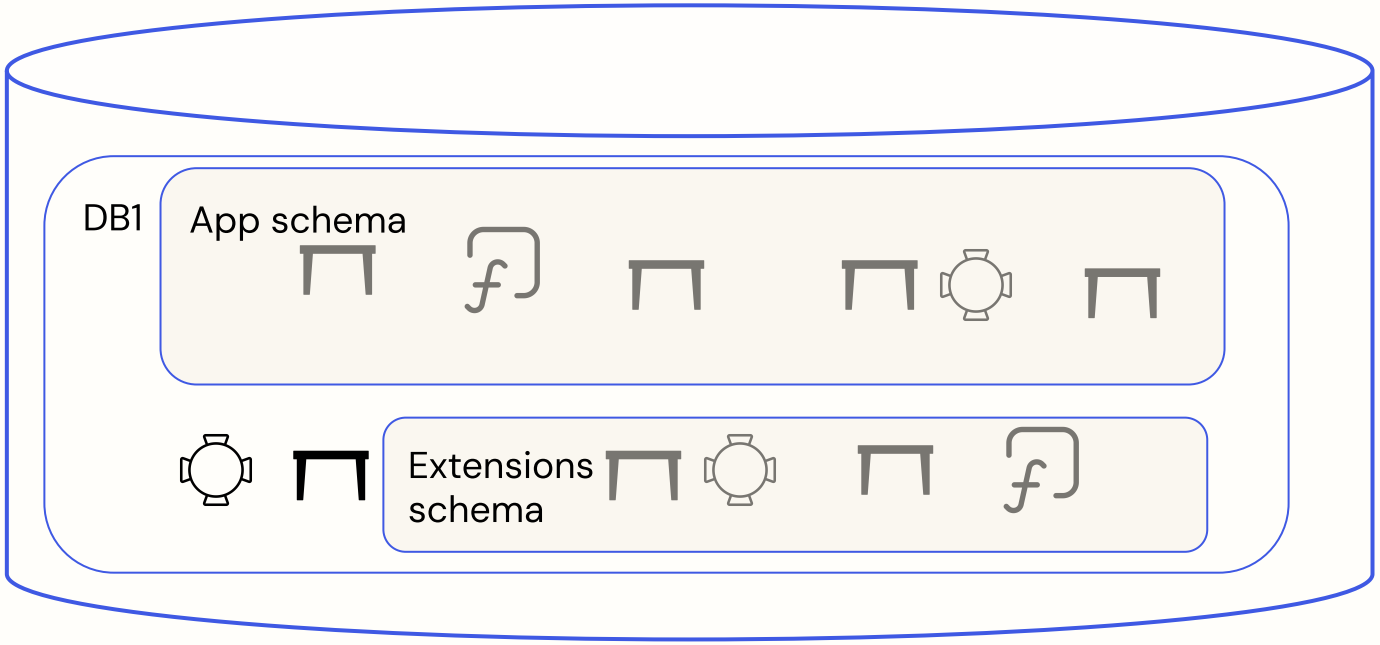


Postgres Schemas

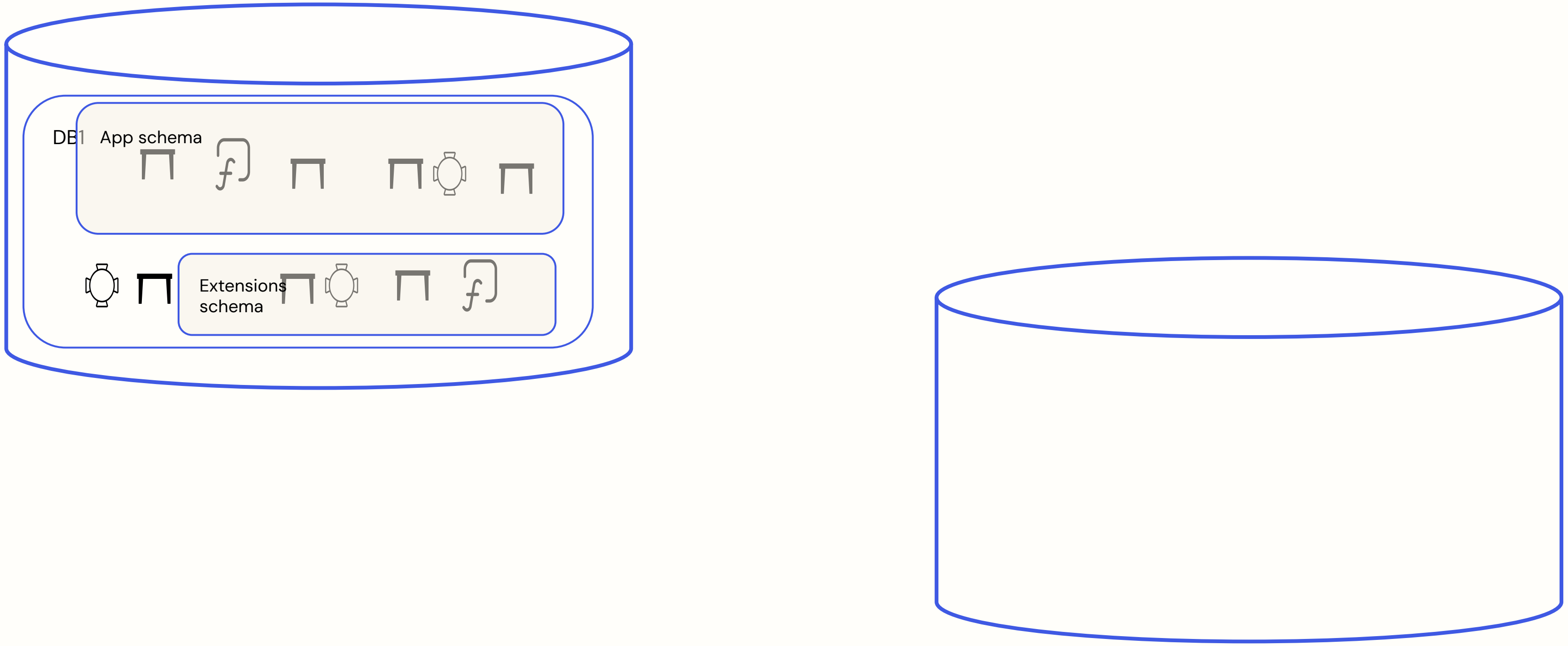


Ok, so why is this valuable at scale?

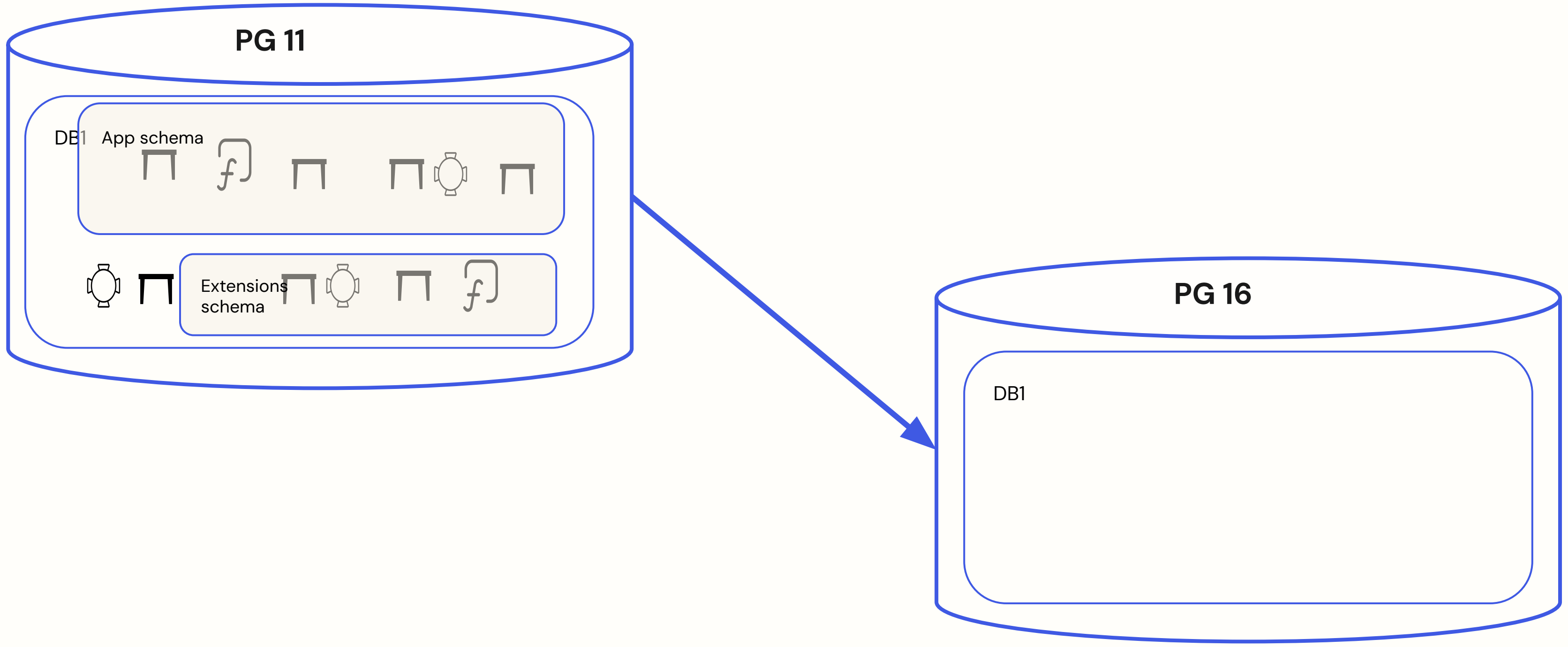
Postgres Schemas



Postgres Schemas

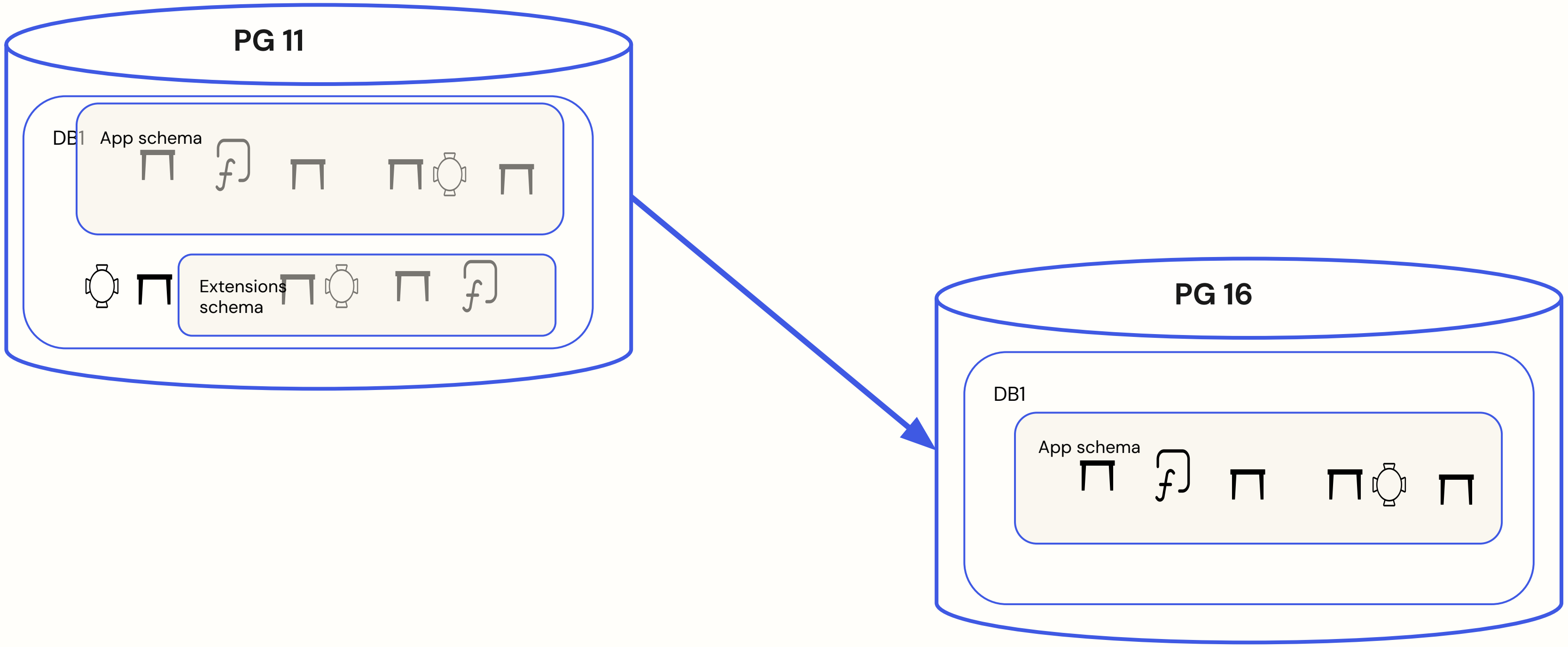


Postgres Schemas



okta

Postgres Schemas



okta

Postgres Schemas

Very useful on database migrations!

Keep permissions isolated

Allow for simpler CDC and model migrations

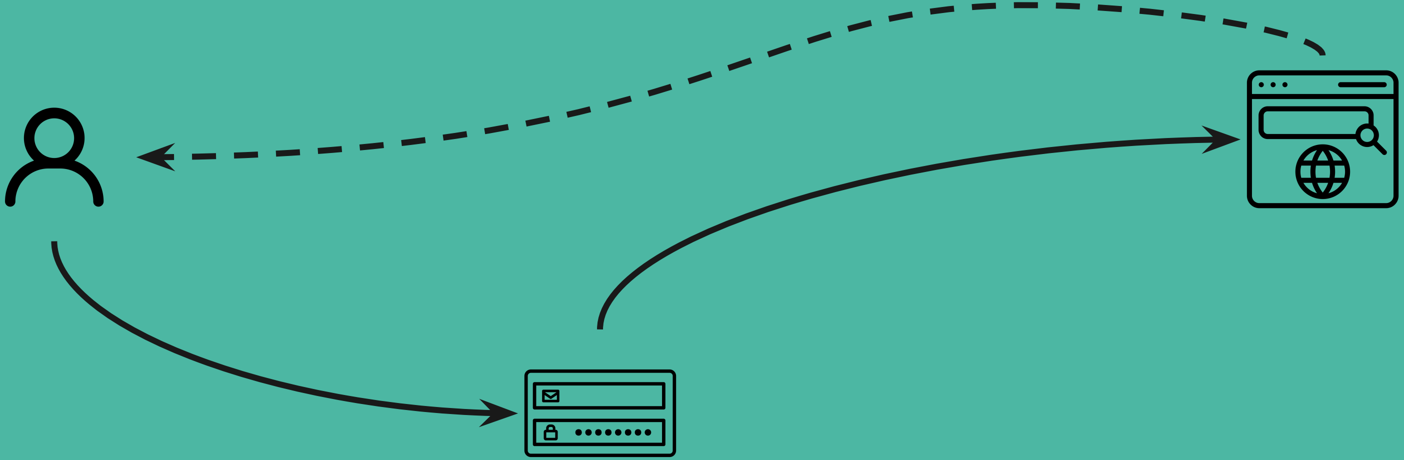
Worst thing I ever said it was ok to live without :(

okta

okta

The World's Identity Company

Quick Recap



Pretty simple, right?

Tier Based Architecture

Not all services have the same performance and resiliency profile - protect the most critical services -> build for failure

okta

Multiple Redundancy

Databases will fail, make sure we can withstand any single node failure

Redundant Deployment	AZ Spread	PITR Enabled

pgbouncer Deployments

We deploy pgbouncer as a isolated service

Dedicated per workload type reads vs writes

Good for load control – helps scaling during spikes

Shock absorber during failover events and node rotation*

okta

Audit logs

No such thing as a free lunch

Recurrent internal DB processes need to be checked

Move all non operational workloads outside DB

Test triggers and stored procedures at scale

Thank you!

okta