

# Managing Large Database Landscapes in Multi-Cloud DBaaS Environments

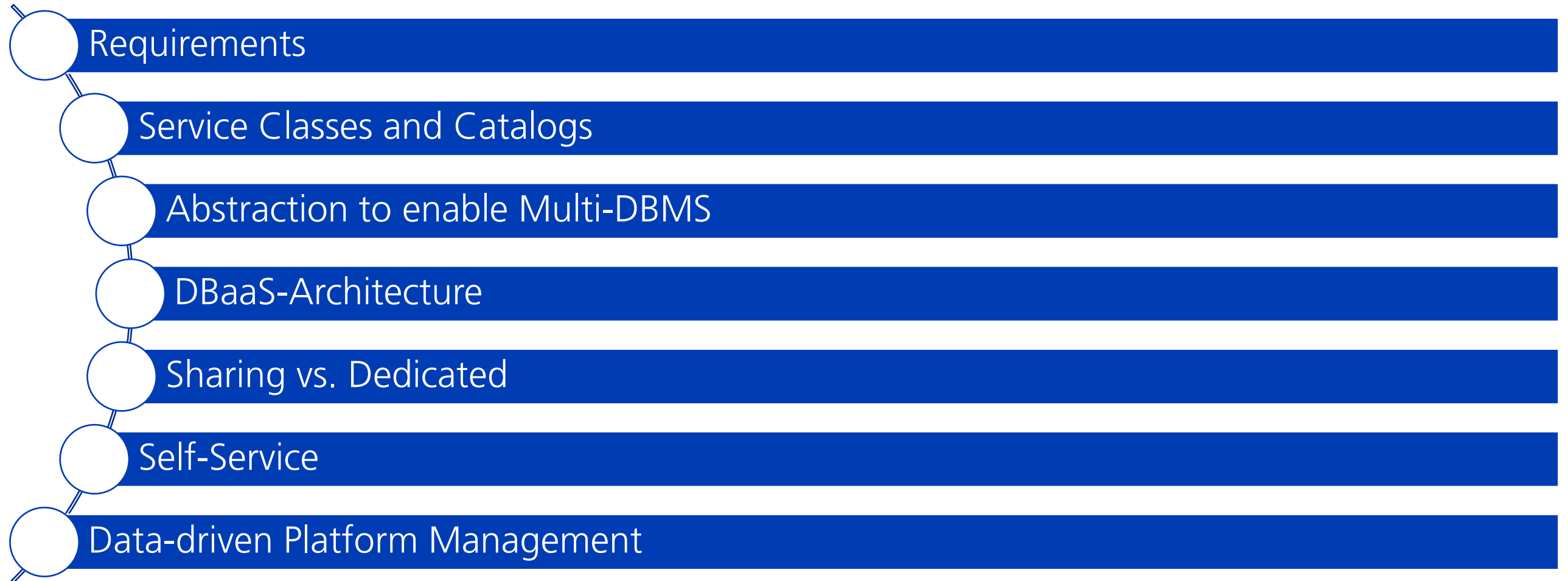
Swiss PGDay  
27. Juni 2024  
Andreas Geppert



# About Me

- ▶ Diploma in Computer Science from the Karlsruhe Institute of Technology
- ▶ Dr. in Computer Science from the University of Zurich
- ▶ Visiting Scientist at IBM Almaden Research Center
- ▶ Platform Architect at Credit Suisse (14 years)
- ▶ Database Architect, Consultant and Engineer at Swiss Re (6 years)
- ▶ **DevOps Solution Engineer at ZKB (since July 2022)**
- ▶ Vice president of the Swiss Postgres Users Group
- ▶ <https://www.linkedin.com/in/andreas-geppert-6964a0178/>

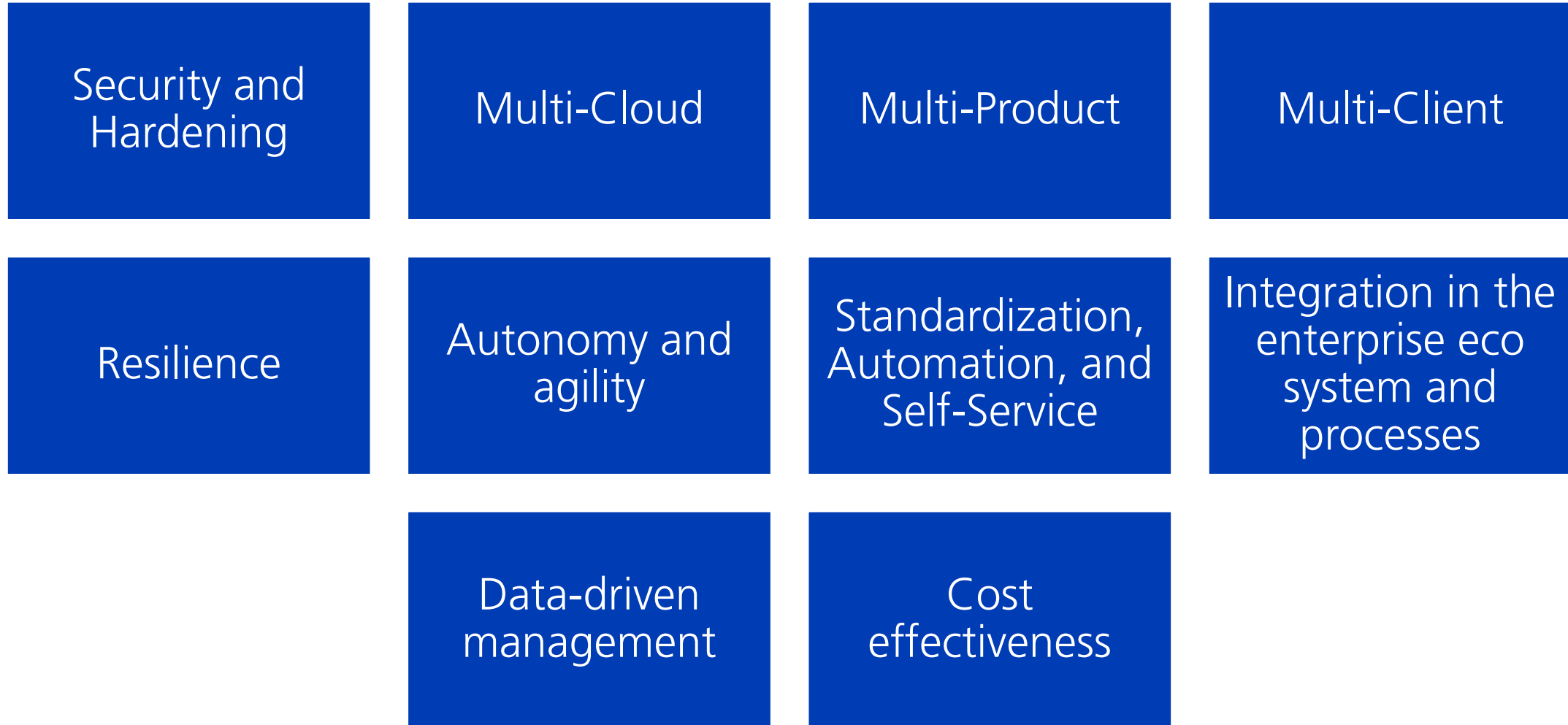




# Current Trend

- ▶ Most large enterprises move to the cloud
- ▶ Postgres is the perfect choice as strategic and standard database management system in the cloud
- ▶ How to architect, build, and run your database landscape with Postgres and possibly other DBMSs in the clouds?



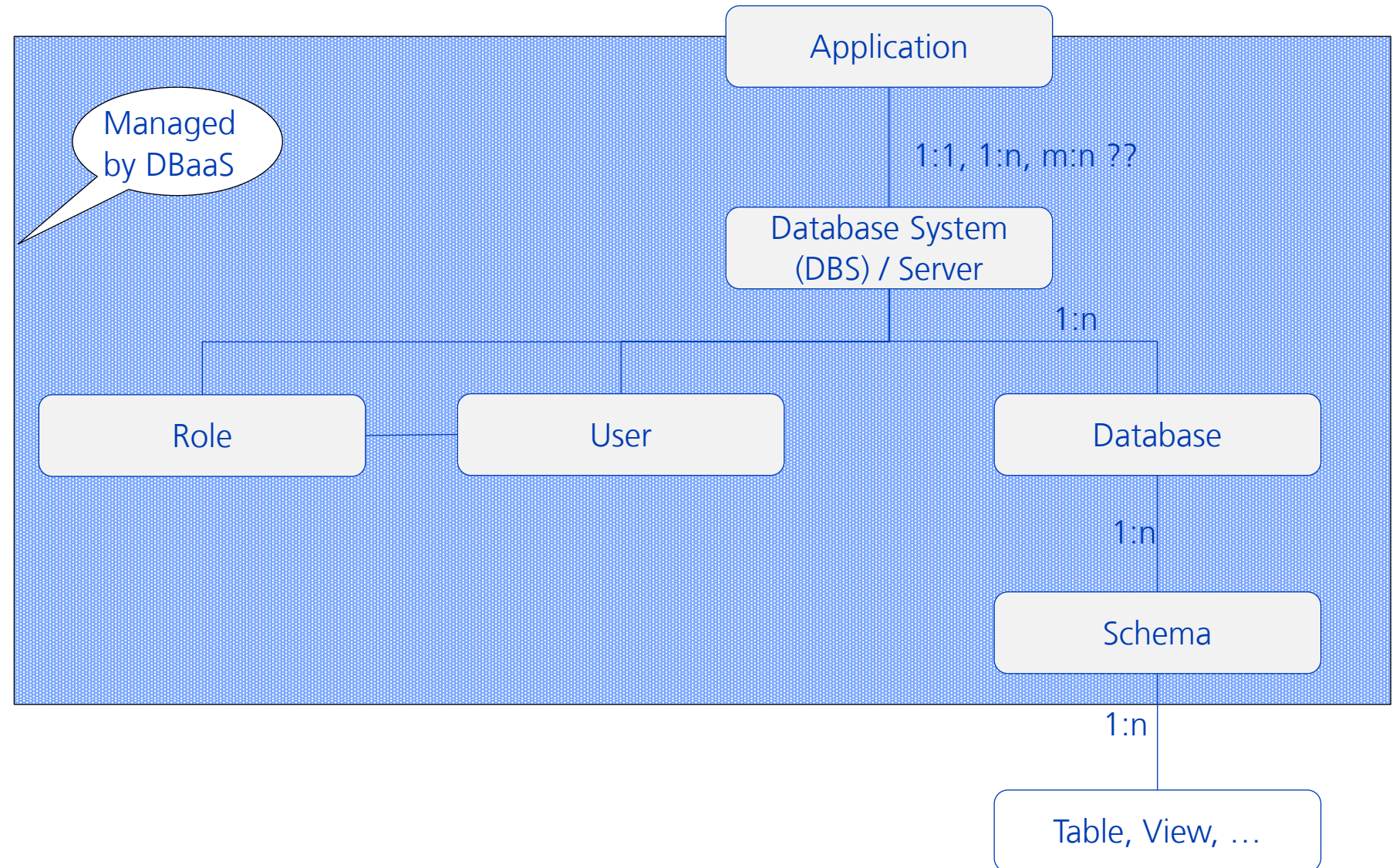


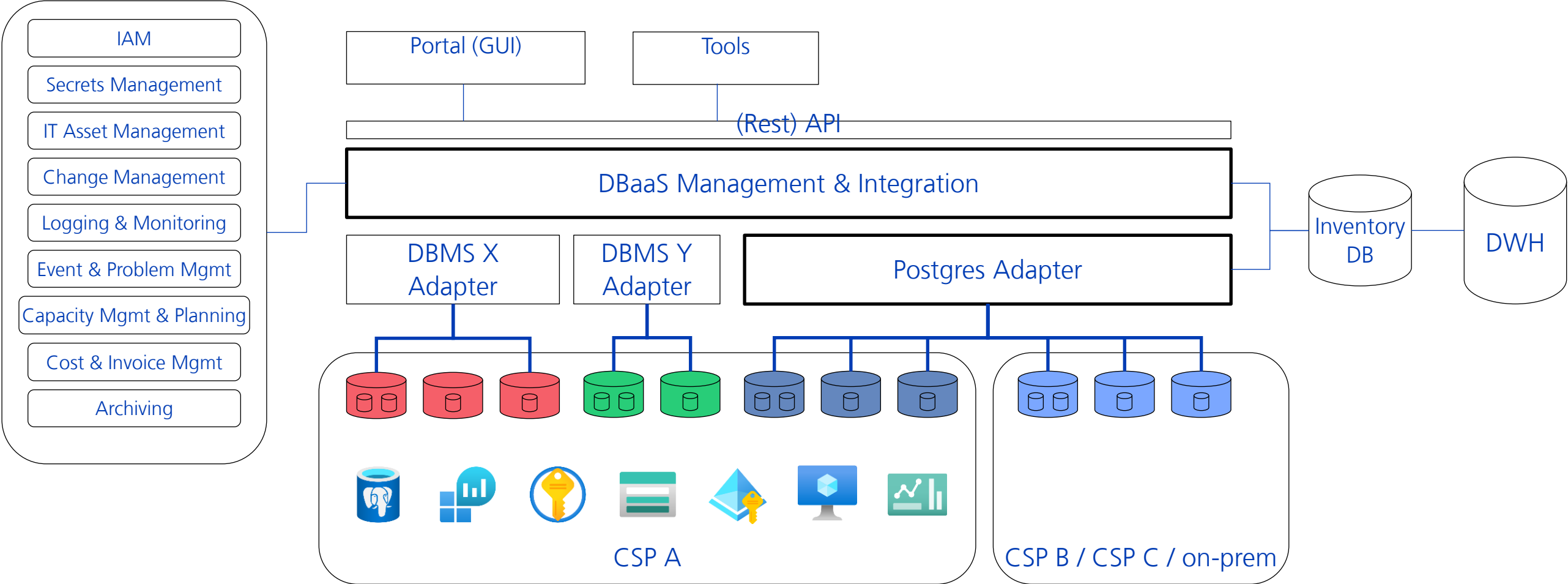
# DBaaS Service Catalogs

## Meeting a spectrum of application requirements

		D (Basic)	C (Standard)	B (Premium)	A (Platinum)
	DBMS	Postgres			
DR	DR	No	Yes	Yes	Yes
	RTO	-	24h	4h	4h
	RPO	-	24h	0h	0h*
Availability	HA	No	No	Yes	Yes
	Planned Downtime	N/A; depends on provider			
Performance	Scalability	No	No	Vertical	V/H
	Isolation	No	No	Partially	Yes

- ▶ Abstraction helps to address common issues once and for all DBMSs
- ▶ Particularly integration is DBMS-independent

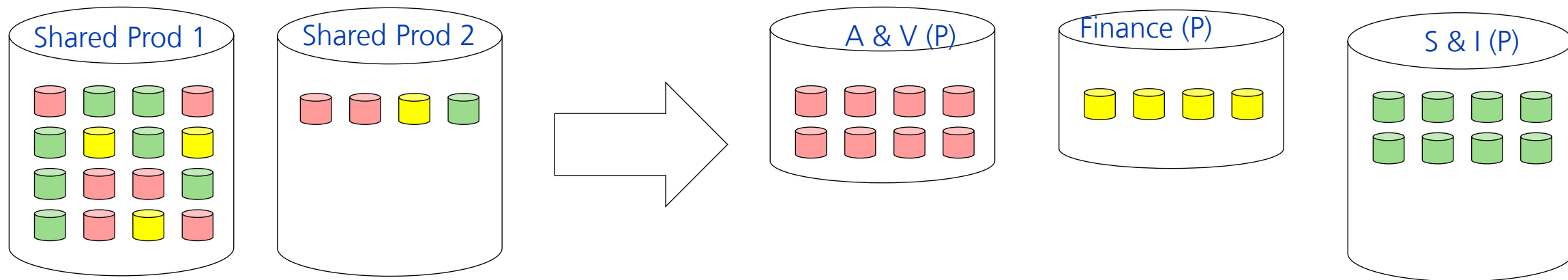






# Sharing vs Dedicated

- ▶ Many applications do not utilize an own server
- ▶ Trade-off:
  - Sharing a server between multiple applications then saves cost and optimizes utilization
  - But applications can impact each other
- ▶ Let domains and applications decide autonomously
- ▶ Default: place servers and databases along the boundaries of domains/portfolios



# Sharing vs Dedicated Current Situation (Partial Snapshot)

- ▶ Currently, almost 300 databases
- ▶ In ~ 30 Postgres servers (shared and dedicated)
- ▶ Upcoming re-placement along portfolio boundaries upcoming, with upgrade to version 16

The screenshot displays a database inventory tool interface. At the top right, there is a logo for 'Zürcher Kantonalbank' and a search icon. The main area shows a list of database instances, each represented by a green pill-shaped label. The instances are organized into two sections, both starting with 'ENG'. The first section lists instances like 'ais\_dev', 'ake\_dev', 'apimplafo\_dev', etc. The second section lists instances like 'avv\_dev', 'bali\_dev', 'bbs\_dev', etc. The labels are arranged in a grid-like fashion, with some instances having multiple environment suffixes (e.g., \_dev, \_ut, \_kt, \_it).

ENG

- ais\_dev ais\_ut ais\_kt ais\_it
- ake\_dev ake\_ut ake\_it
- apimplafo\_dev apimplafo\_ut apimplafo\_it
- bat\_dev bat\_ut bat\_kt bat\_it
- conba\_dev conba\_ut conba\_it
- ctf\_dev ctf\_ut ctf\_it
- dam\_ut
- eai\_ut eai\_kt eai\_it
- far\_dev far\_ut far\_kt far\_it
- icmo\_dev icmo\_ut icmo\_kt icmo\_it
- itng\_dev itng\_ut
- multib\_dev multib\_ut multib\_it
- sfb\_dev sfb\_ut sfb\_kt sfb\_it
- ssdlcsca\_ut
- uapadmin\_dev uapadmin\_ut uapadmin\_mig\_ut uapadmin\_kt

ENG

- avv\_dev
- bali\_dev bali\_ut bali\_it
- bbs\_dev bbs\_ut bbs\_kt bbs\_it
- biadv\_dev
- bps\_ut bps\_it
- bvb\_dev bvb\_ut bvb\_it
- cia\_dev cia\_ut cia\_it
- cls\_dev cls\_ut cls\_kt cls\_it
- dam\_dev dam\_ut dam\_kt dam\_it
- dort\_dev dort\_ut dort\_it
- eim\_dev eim\_eng eim\_it
- ewm\_dev ewm\_ut ewm\_it
- fim\_dev fim\_ut fim\_it
- fir\_dev fir\_ut fir\_it
- frontarena\_eng
- gema\_dev gema\_ut gema\_kt gema\_it
- goaml\_dev goaml\_ut goaml\_it
- harbor\_core\_ut
- immob\_dev immob\_ut immob\_kt immob\_it
- immod\_dev immod\_ut immod\_kt immod\_it
- inl\_dev inl\_ut inl\_kt inl\_it
- lima\_ut lima\_kt lima\_it
- mbclt\_dev mbclt\_ut
- notifit\_ut notifit\_kt notifit\_it
- ors\_it
- regmel\_dev regmel\_ut regmel\_it
- sapdm\_dev sapdm\_ut sapdm\_kt sapdm\_it
- sic\_dev sic\_ut sic\_kt sic\_it
- slx\_dev slx\_ut slx\_it
- sonar\_ut
- uapadmin\_it
- uapsales\_ut uapsales\_it
- vulscan\_dev vulscan\_ut vulscan\_kt vulscan\_it
- zfx\_ut zfx\_it

- ▶ Application autonomy and agility require self-service for users
  - ▶ All routine tasks should be available as self-service
  - ▶ REST API implements self-service features
  - ▶ Adequate interfaces should be provided on top of REST API (GUI/portal)
- ▶ CRUD Database System
  - ▶ CRUD Database
  - ▶ CRUD Schema
  - ▶ CRUD User
  - ▶ CRUD Role
  - ▶ CRUD Role Membership
  - ▶ Export/import
  - ▶ Backup & Archiving
  - ▶ SQL Execution
    - DDL
    - DML
  - ▶ Migration

# DBaaS REST API

## Initial Pilot

infra.Applicationlfc



infra.DbmsVersionlfc



infra.Serverlfc



infra.Databaselcfc



**GET** /database/list/all returns all DBs.



**GET** /database/listapp/{name} returns all DBs by name.



**POST** /database/add/{server}/{name} Adds a Postgres logical db to Server.



infra.DbUserlfc



infra.DbRolelfc

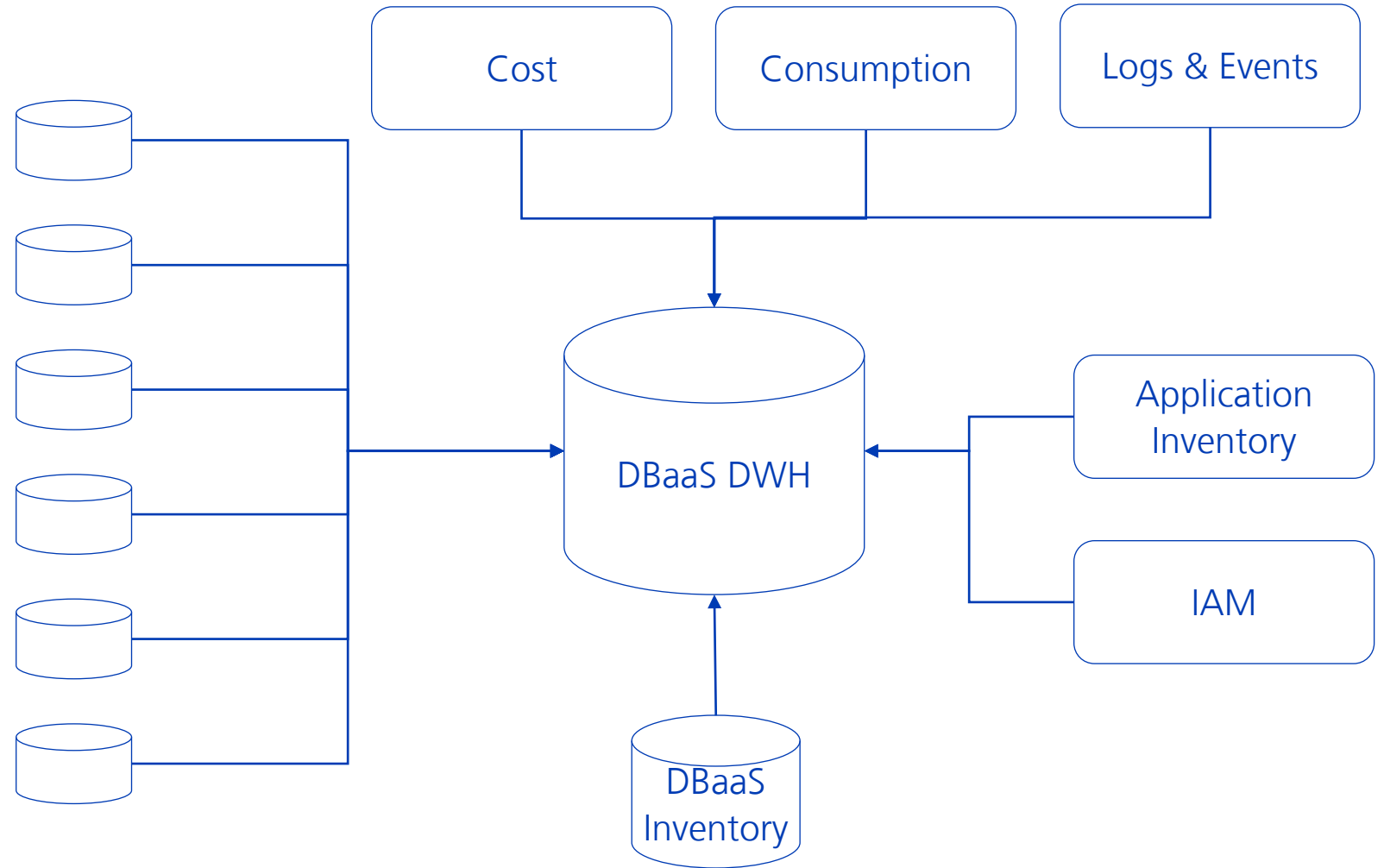


infra.DbSchemalfc



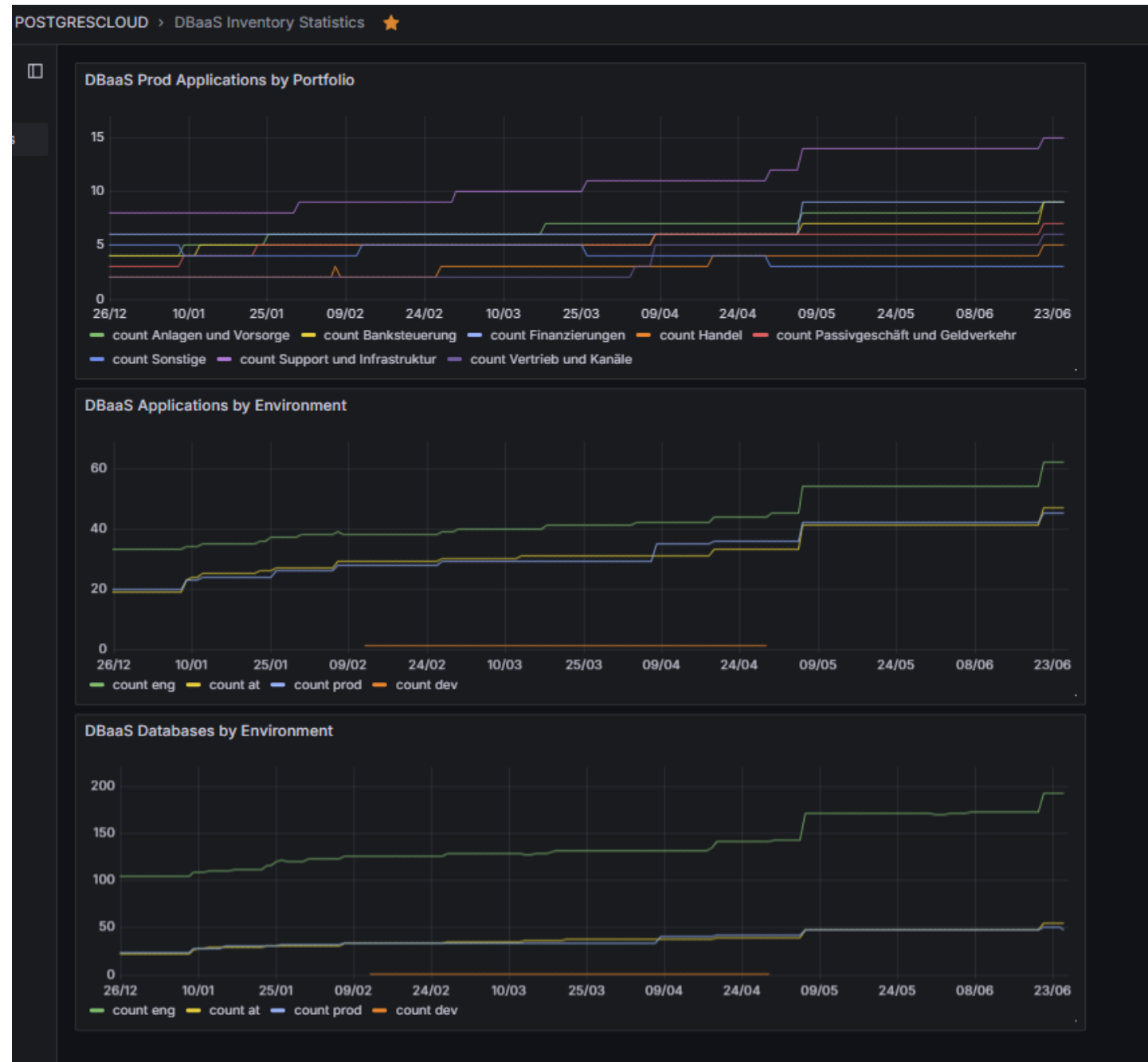
# The DBaaS Data Warehouse

- ▶ The DBaaS data warehouse collects historical data from all relevant sources and enables data-driven platform management, such as:
  - Usage reporting and analysis
  - Invoice verification and cost allocation
  - Security monitoring and configuration drift detection



# The DBaaS Data Warehouse

- ▶ Sample dashboards showing number of applications per portfolio/domain and environments, and databases by environment



- ▶ Postgres User Management requires updates to the pg\_hba.conf
  - See [Automatisierte Benutzer- und Zugangsverwaltung in DBaaS-Umgebungen @ PGConf.de 2022](#)
- ▶ Built-in Transparent Data Encryption
- ▶ Resource manager (as in other DBMS) to limit resource consumption by individual databases in shared servers
  - Database connection limit
  - Monitoring + «Eviction» of unfriendly neighbors into own servers

# Conclusion

- ▶ Automation for Postgres in Azure completed
- ▶ Currently completing the first version of the DBaaS self-service
- ▶ Additional DBMSs, further clouds?
- ▶ Cross-platform setups

?!?

