AWS re:Invent

ARC210-R

Microservices decomposition for SaaS environments

Tod Golding

Principal Partner Solutions Architect

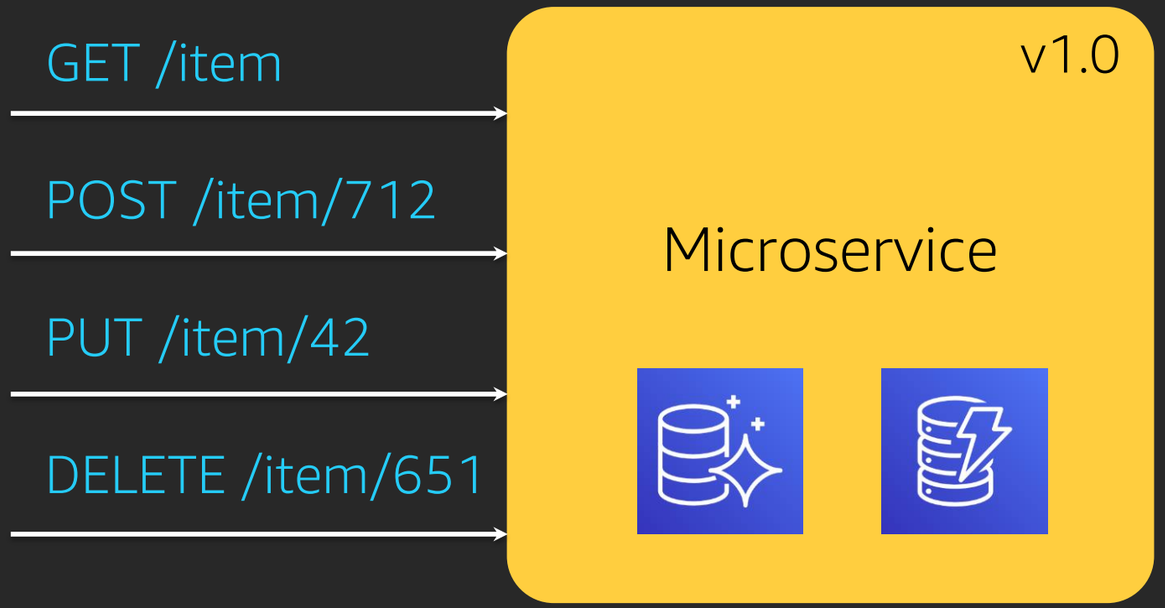
Amazon Web Services

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

Let’s agree on microservice tenets

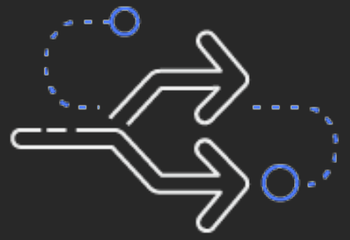


- Autonomous
- Single responsibility
- Owns, manages, and encapsulates its data
- Tested, versioned, and deployed separately
- Scaled independently
- All interactions/coupling is via a contract
- Operated and developed by a team

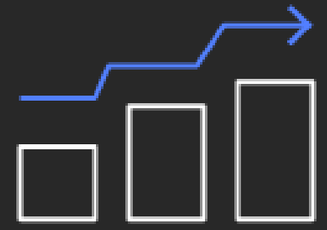
Microservices and SaaS: A good fit



Agility



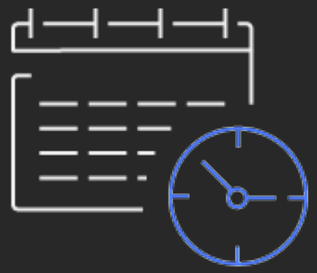
Smaller deployments



Optimized consumption



Fine-grained DevOps

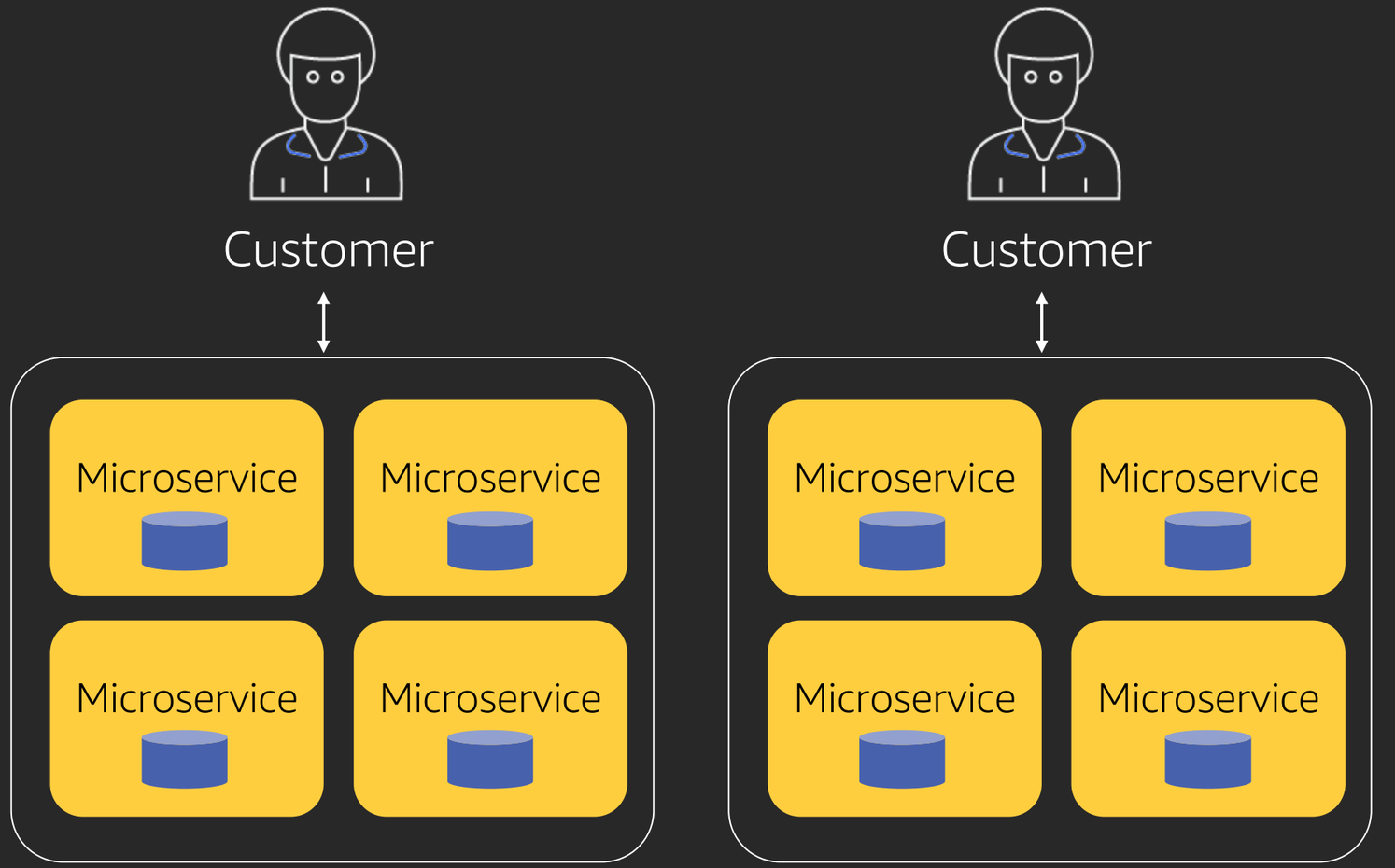


Better aligned with zero downtime



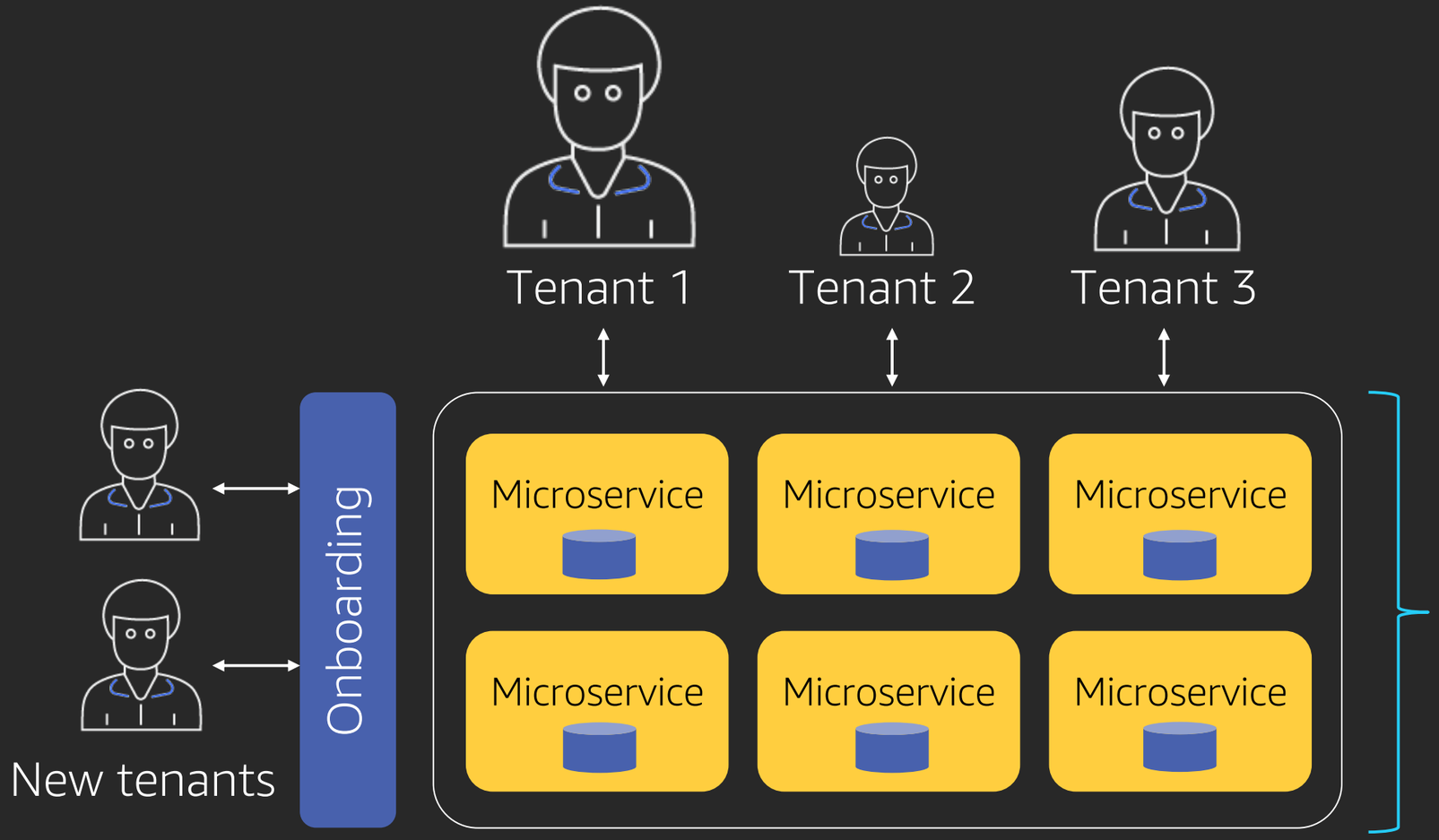
More focus on your IP/customers

Single-tenant microservice model



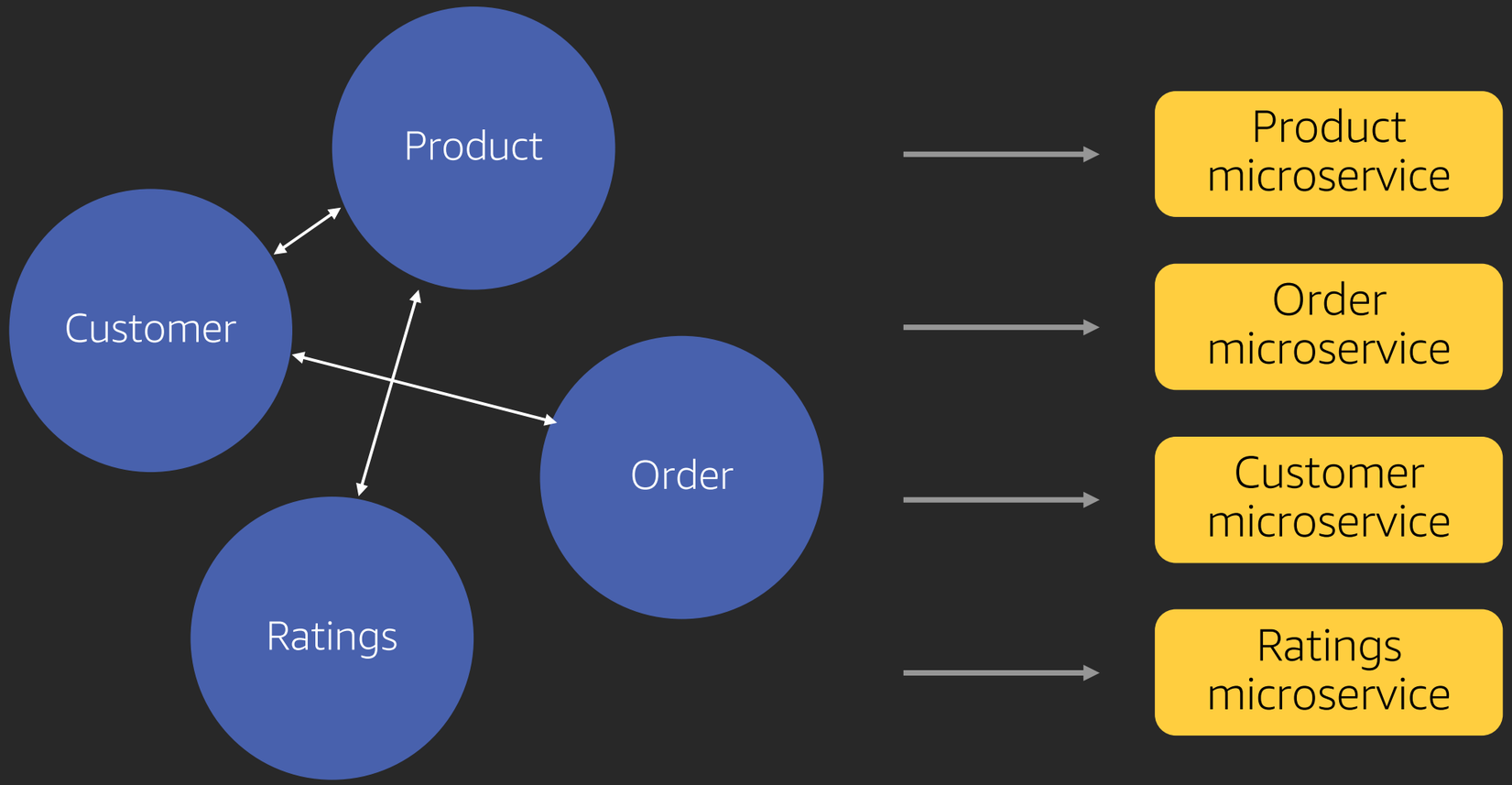
- Predictable activity
- No noisy-neighbor impact
- Limited blast radius
- Simpler scaling policies

Multi-tenant microservice model



- Shared resources
- Noisy neighbor
- Unpredictable activity patterns
- Varying tenant profiles
- Challenging scaling policies
- Greater sensitivity to blast radius

Domains are a good start, but . . .

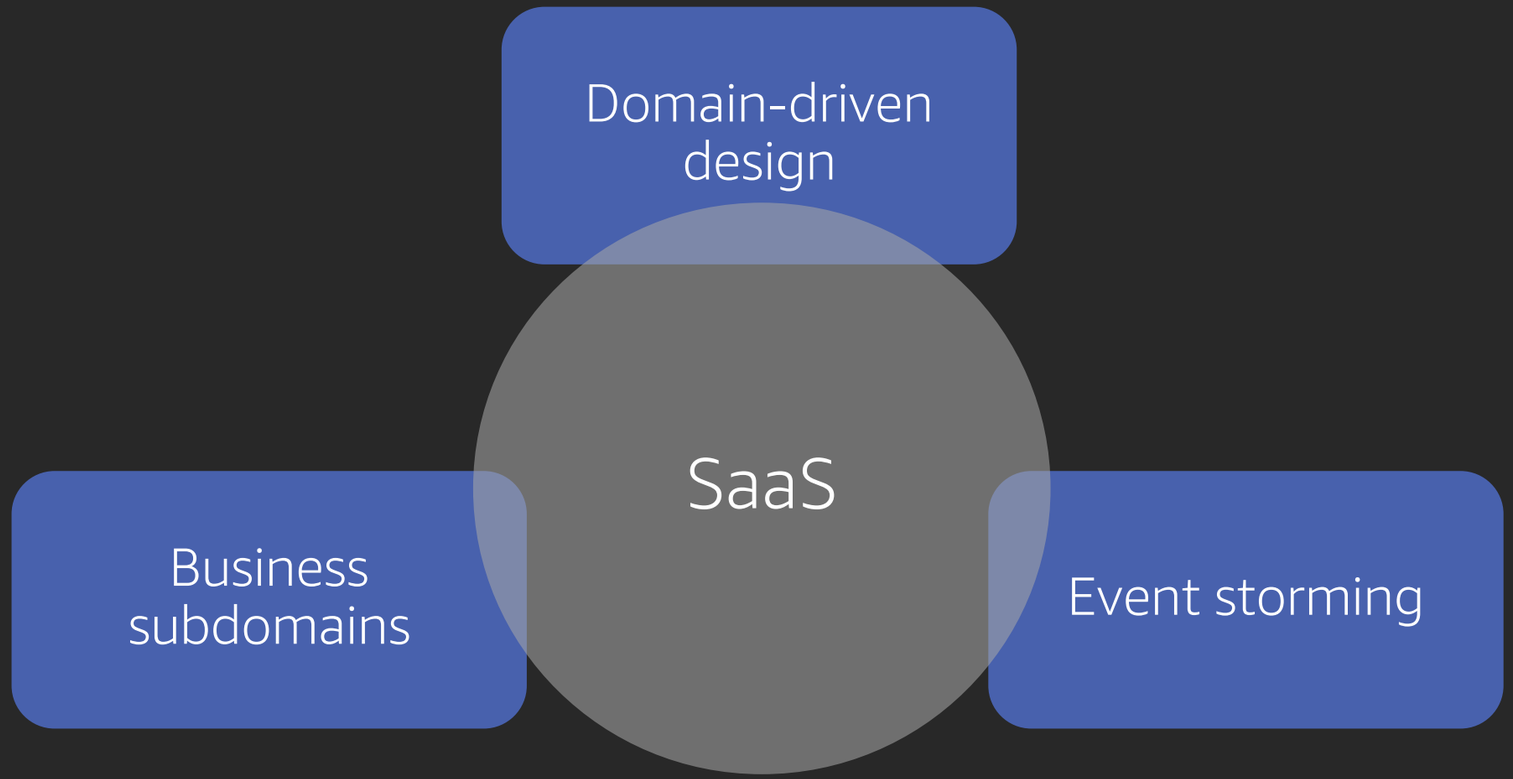


Domains don’t consider

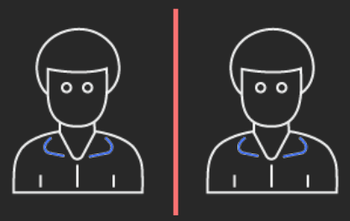
- Consumption optimization
- Fault tolerance
- Noisy neighbor
- Data isolation
- Batch operations
- Tenant bottlenecks

Perspective of users does not include the perspective of your SaaS operational needs

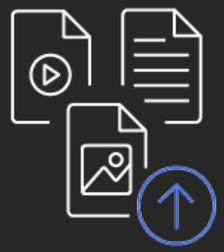
Adding SaaS to conventional techniques



Key areas where SaaS shapes decomposition



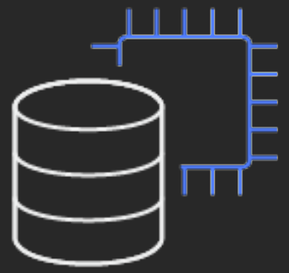
Tenant isolation



Bulk operations



Fault isolation



Data partitioning



Tenant tiering and metrics

Use isolation personas to shape decomposition



Richard Roe
Tenant focused on compliance and security

- A strong belief that his business requires a very stringent security
- Any notion of sharing infrastructure would represent a nonstarter
- Requires tighter control over how and when upgrades are applied in his environment

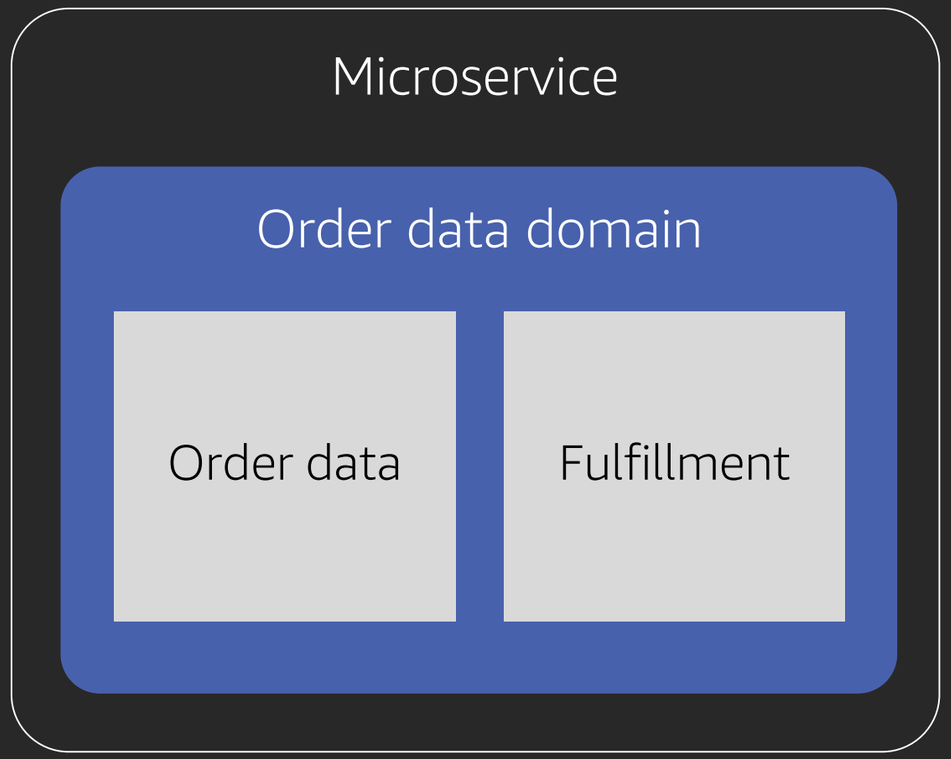


Mary Major
Tenant focused on cost and ease of use

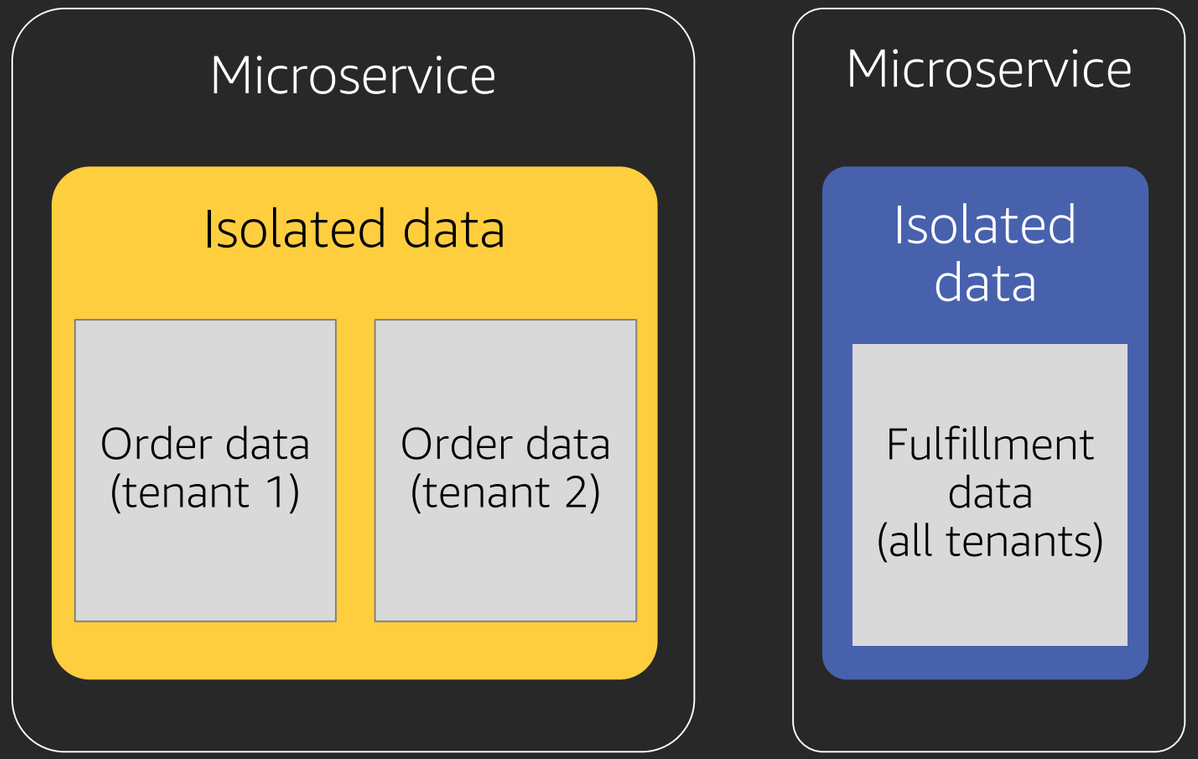
- A small to medium-size company that is squarely focused on value
- Cost and features are at the core of their decision-making process
- Security is important, but Mary presumes it's baked into the product
- More concerned about features and functions

- How will these isolation requirements influence your microservice decomposition?
- What are some of the different expectations that different tenant types might have?

Factoring tenant isolation into decomposition



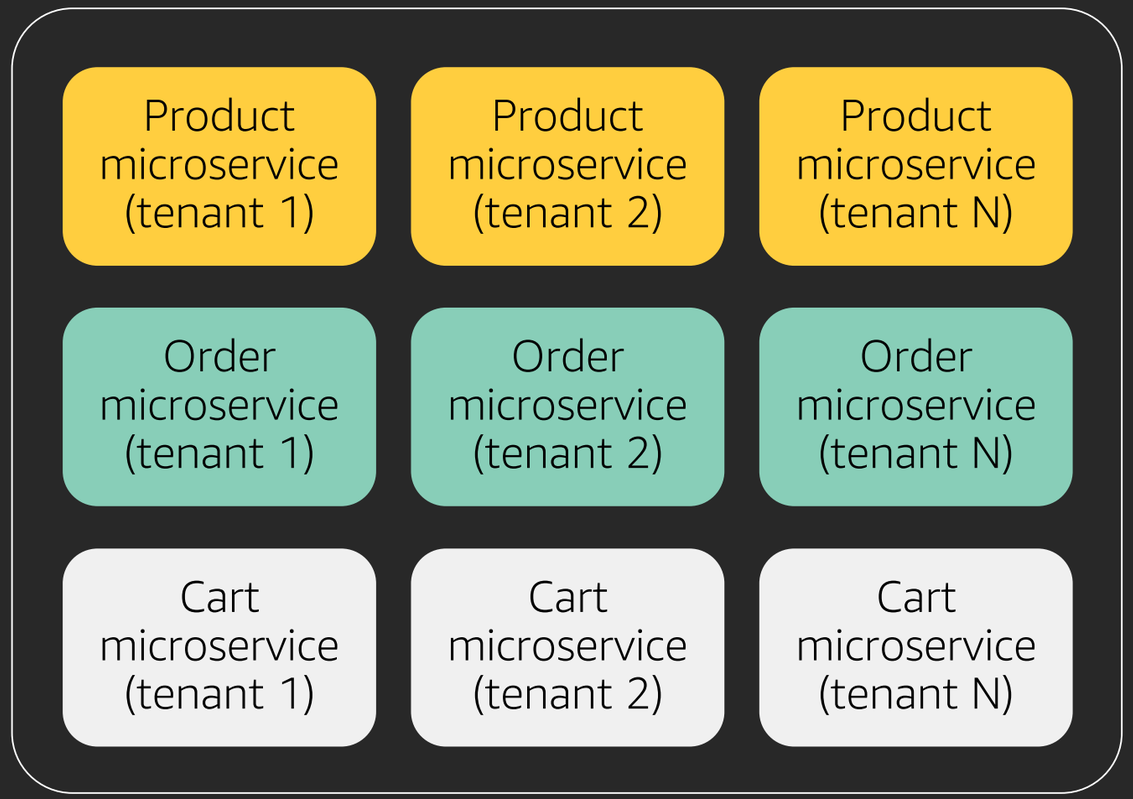
Data aligns with single-tenant boundaries



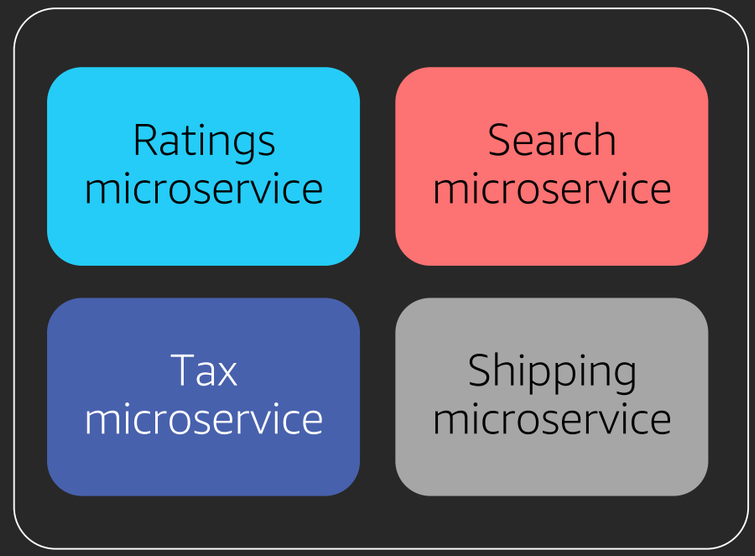
Multi-tenant isolation requirements force the separation into two microservices

Isolation deployment model influences decomposition

Services that must be isolated for each tenant

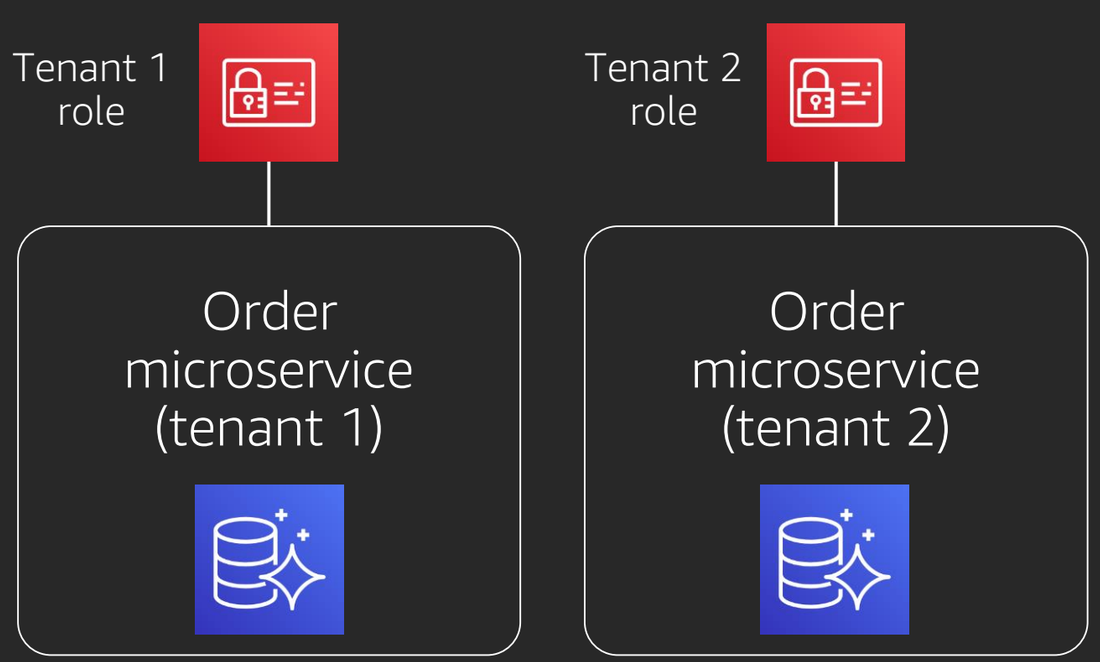


Services that are shared by all tenants

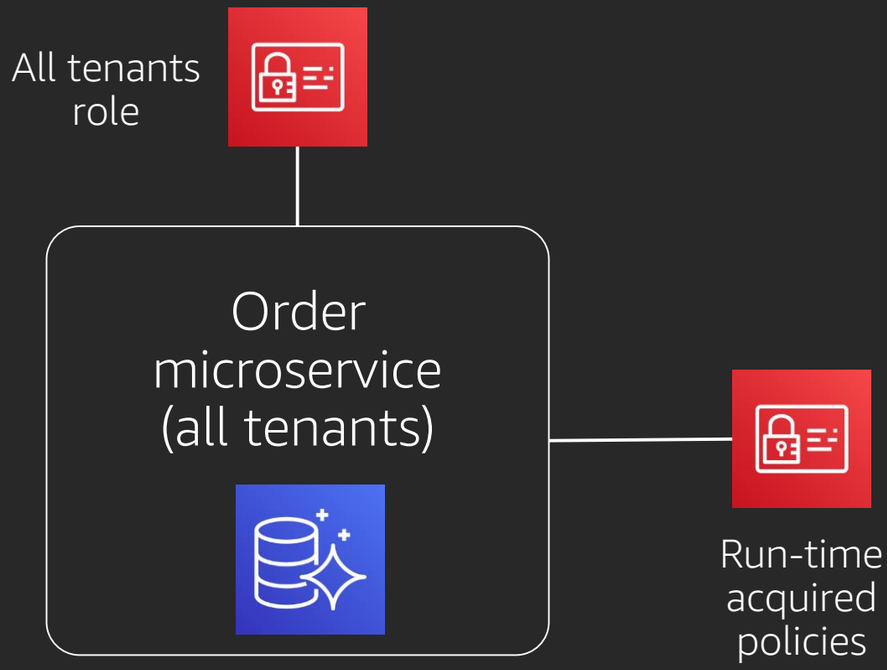


Will your microservice be deployed in a silo or pooled model?

Factoring in the availability of isolation constructs

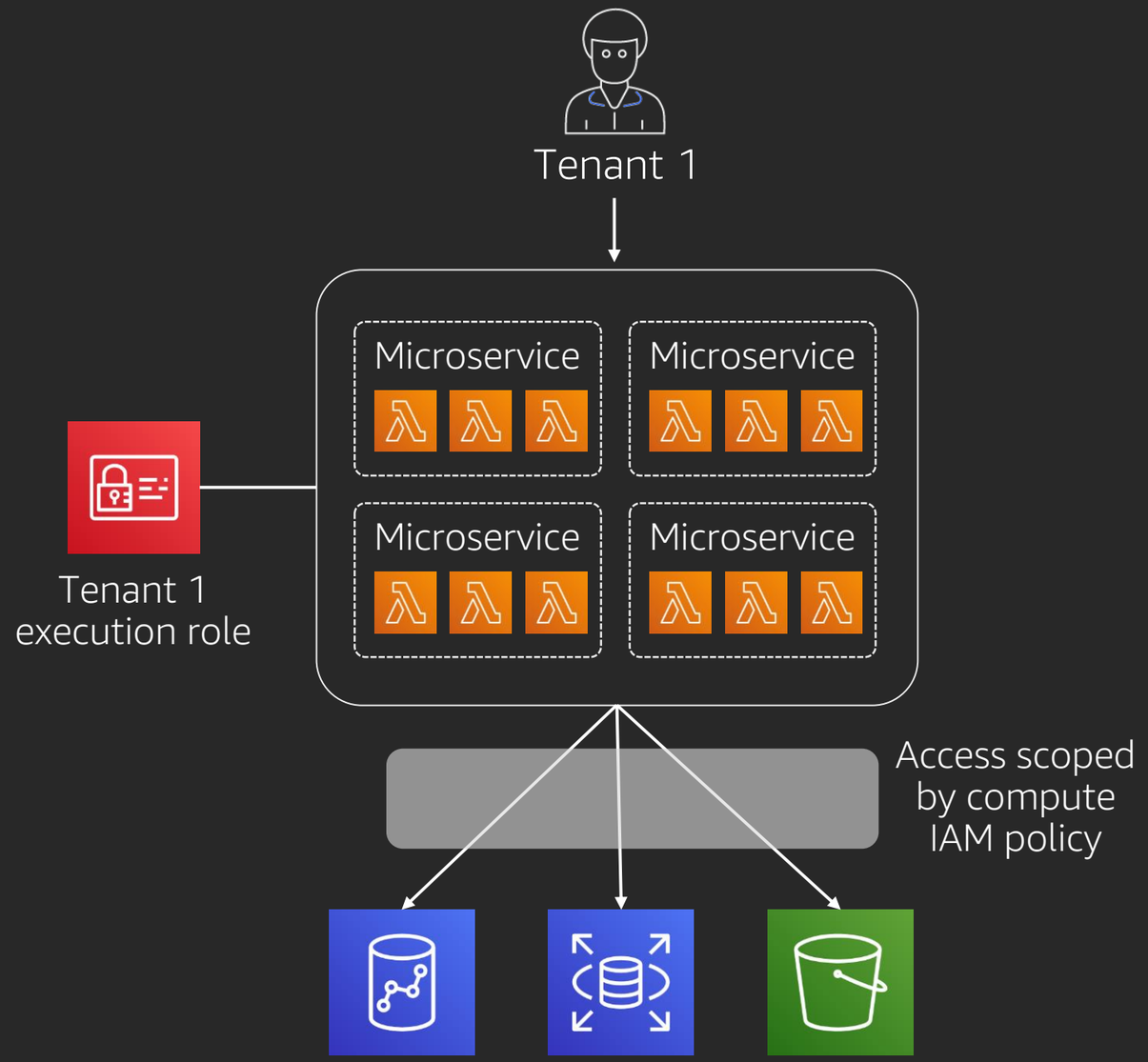


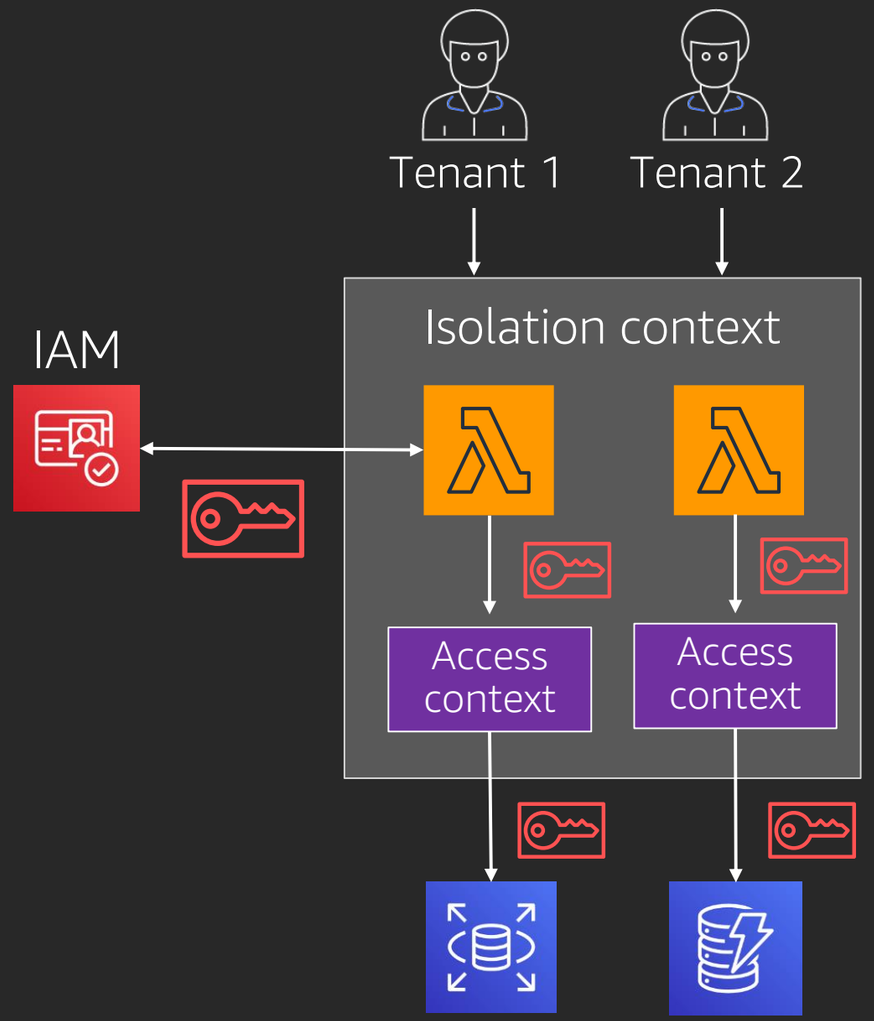
Separate microservice instances provisioned for each tenant (silo)



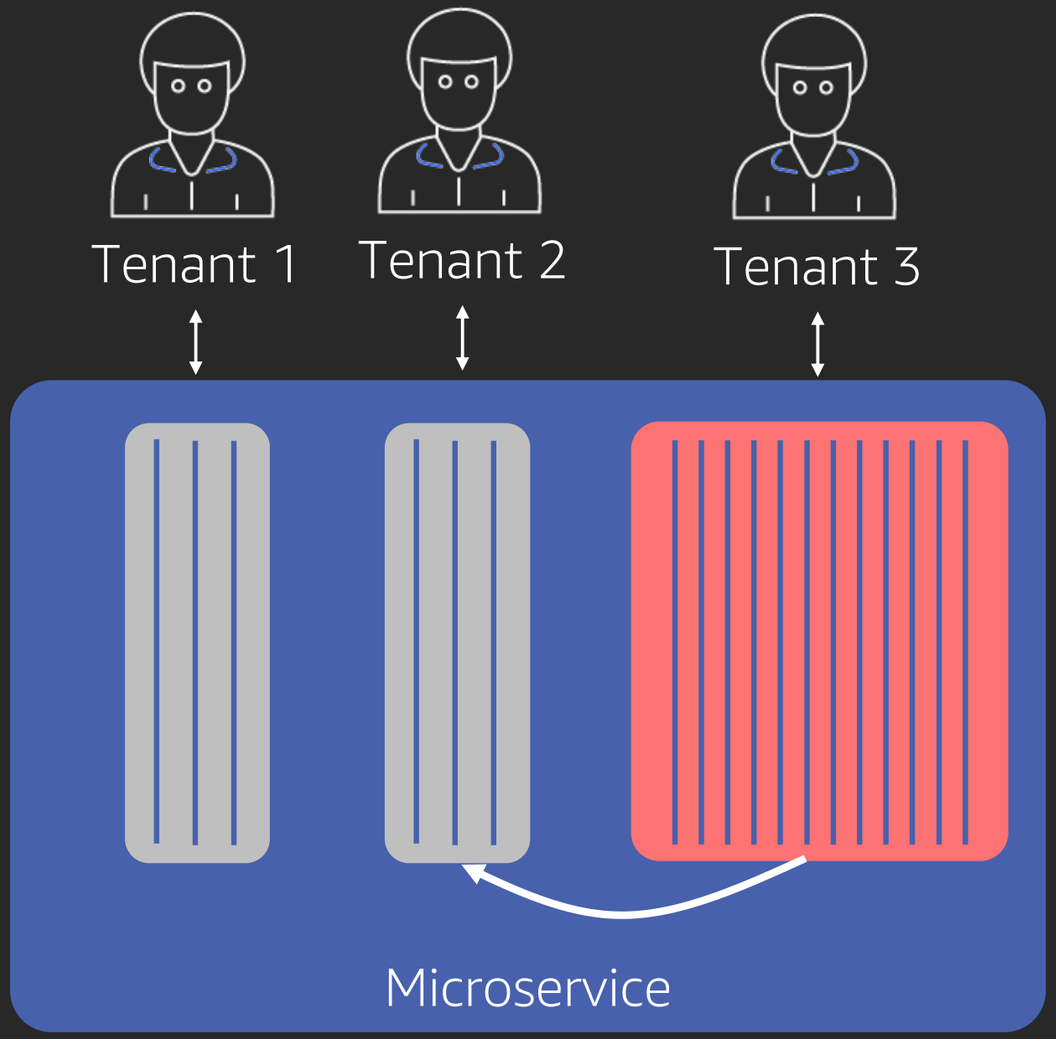
Shared microservice for all tenants (pool)

Applying isolation to downstream resources



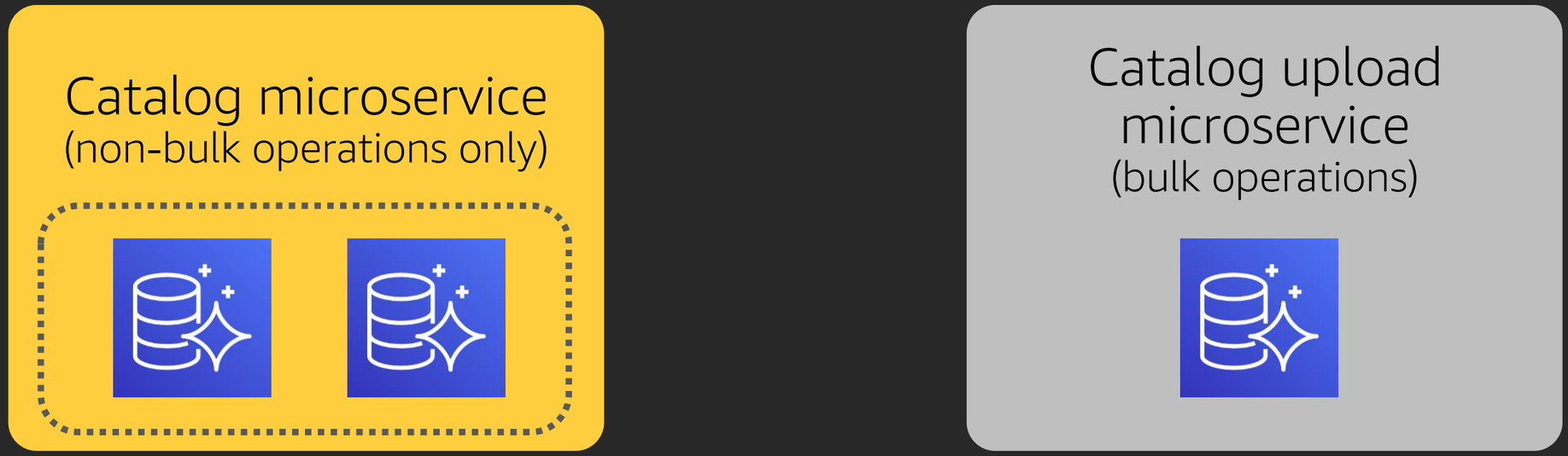


Bulk operations can produce noisy neighbors



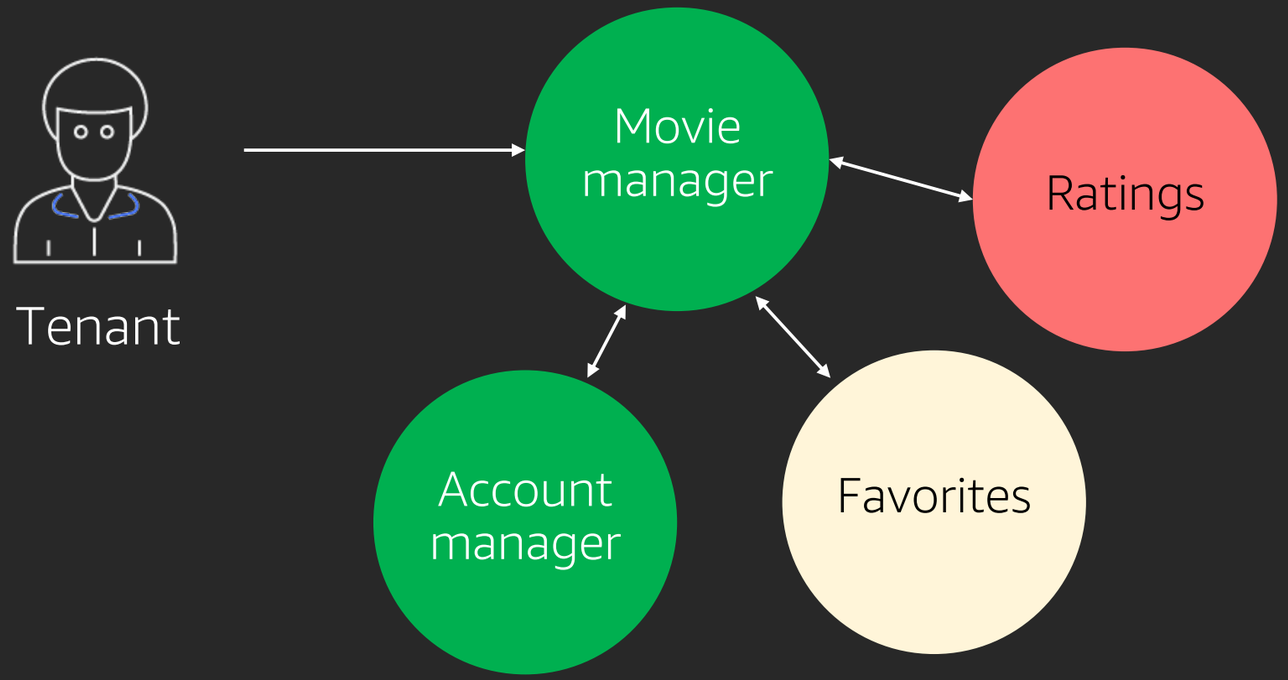
- Bulk operations saturate services
- Impacts experience of other tenants
- Influence SLAs of non-bulk operations

Consider separating bulk operations



- Separate bulk operations into a separate service
- Limit noisy neighbor for interactive operations
- Throttle or scale bulk operations by tenant tier
- Use controlled ingest to import staged data

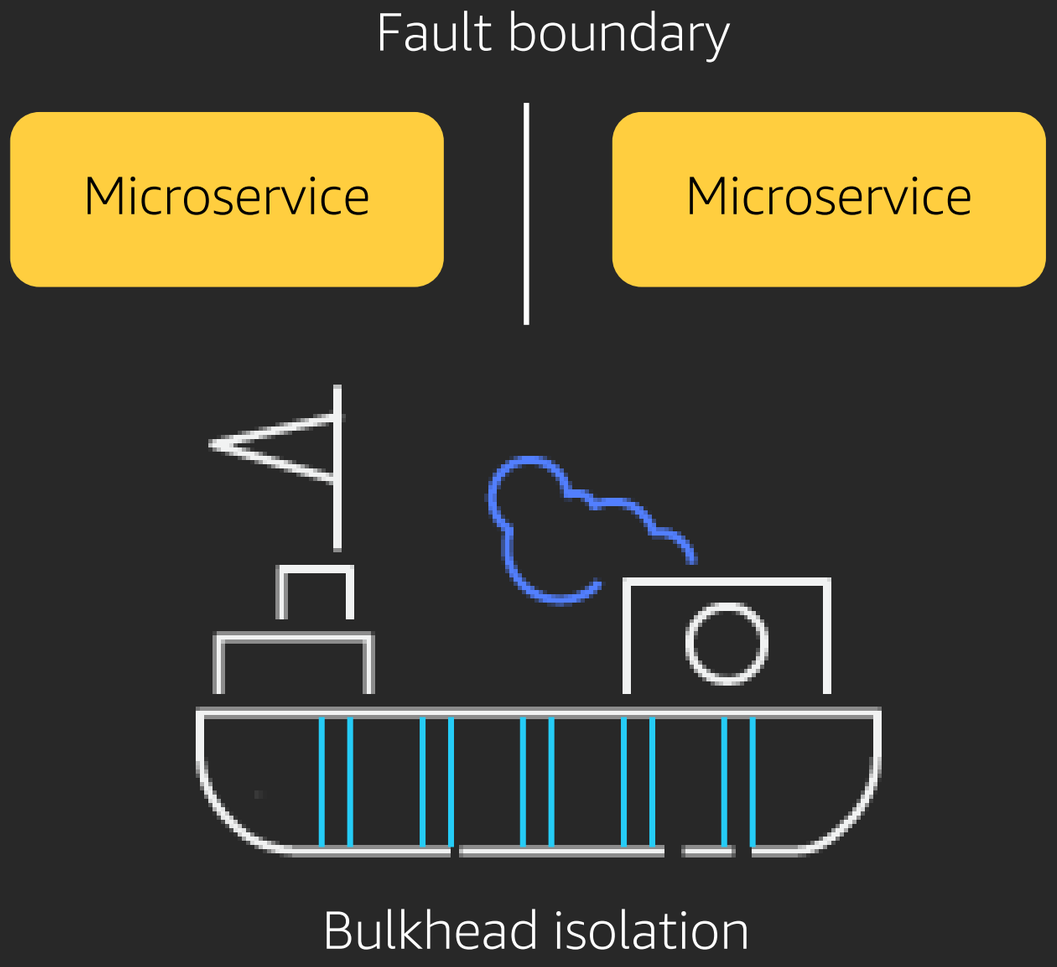
Fault tolerance: SaaS environments rely on zero downtime



Unhealthy, noncritical services should not bring the system down

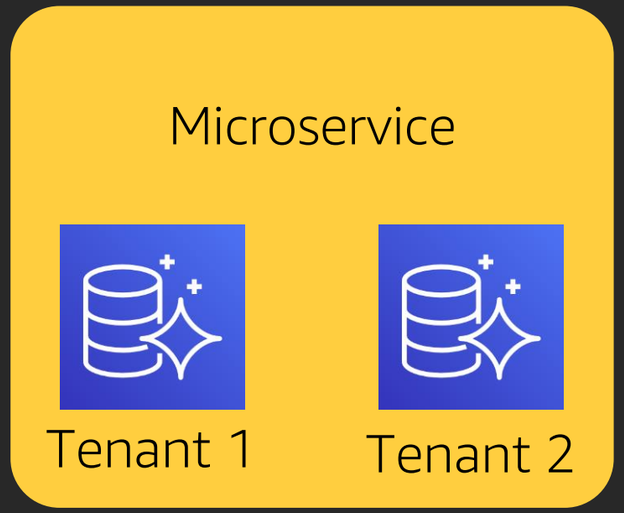
- Fault isolation and resilience bar is higher for SaaS environments
- Microservices must consider fault tolerance as part of decomposition
- Create opportunities to continue even when parts of the system are down

Consider fault isolation during decomposition



- Identify fault profile of each services
- Make resilience part of your decomposition
- What will be your fallback strategy?
- What self-healing strategies will you apply?
- Async as preferred interaction model
- Consider circuit breakers

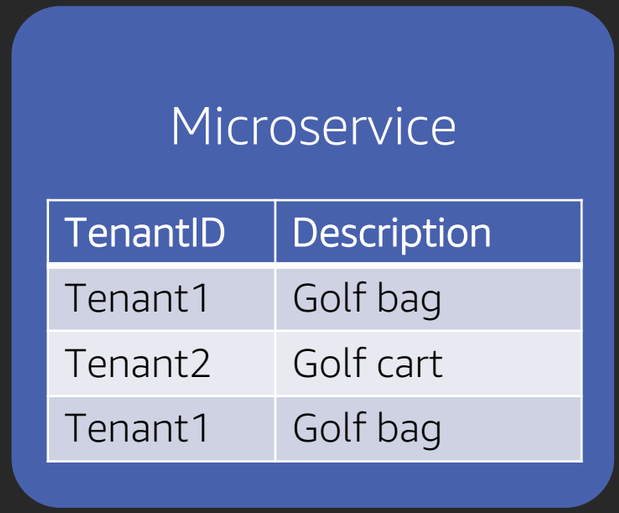
Data partitioning models for microservices



Silo data partitioning model
(separate "database" per tenant)

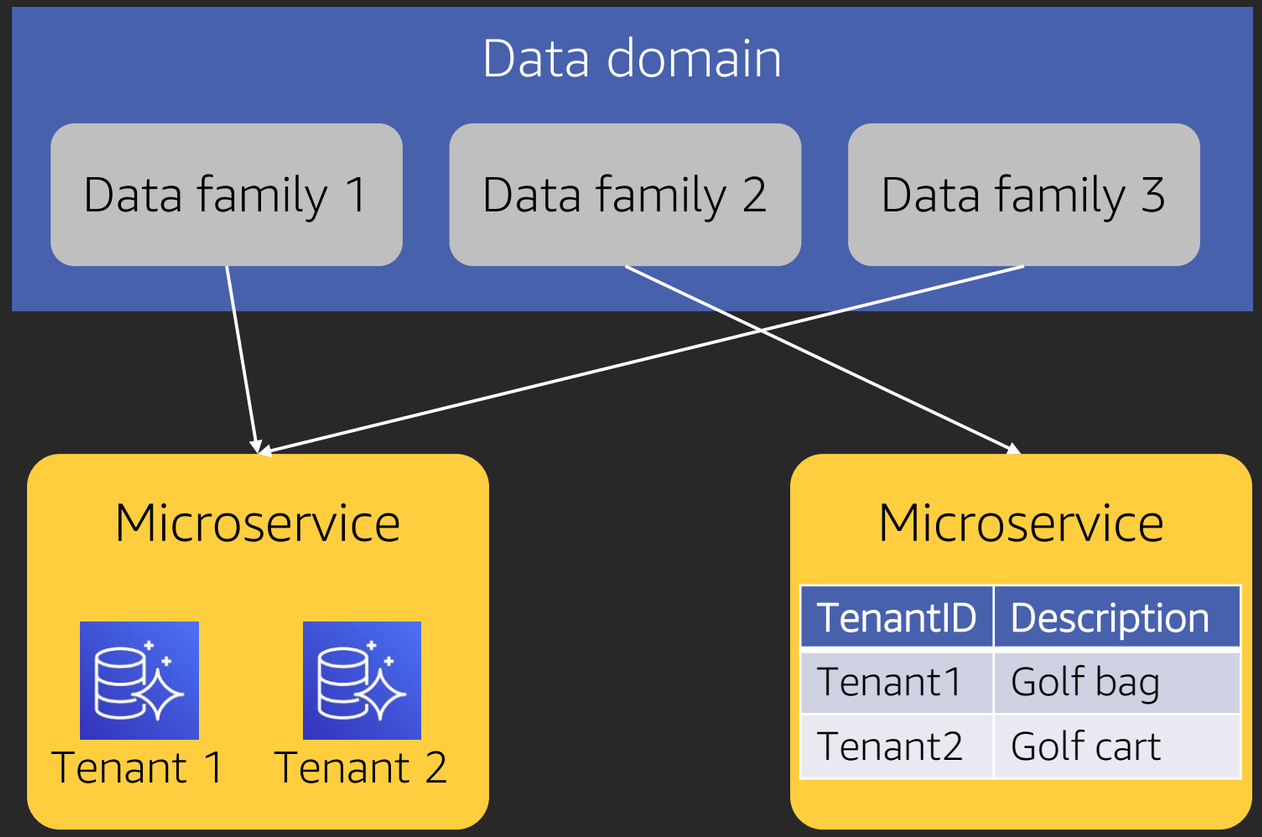
Common influencers

- Compliance and security
- Performance
- Data footprint
- Noisy neighbor
- Data distribution



Pool data partitioning model
(shared "database" for all tenants)

Data partitioning can impact decomposition

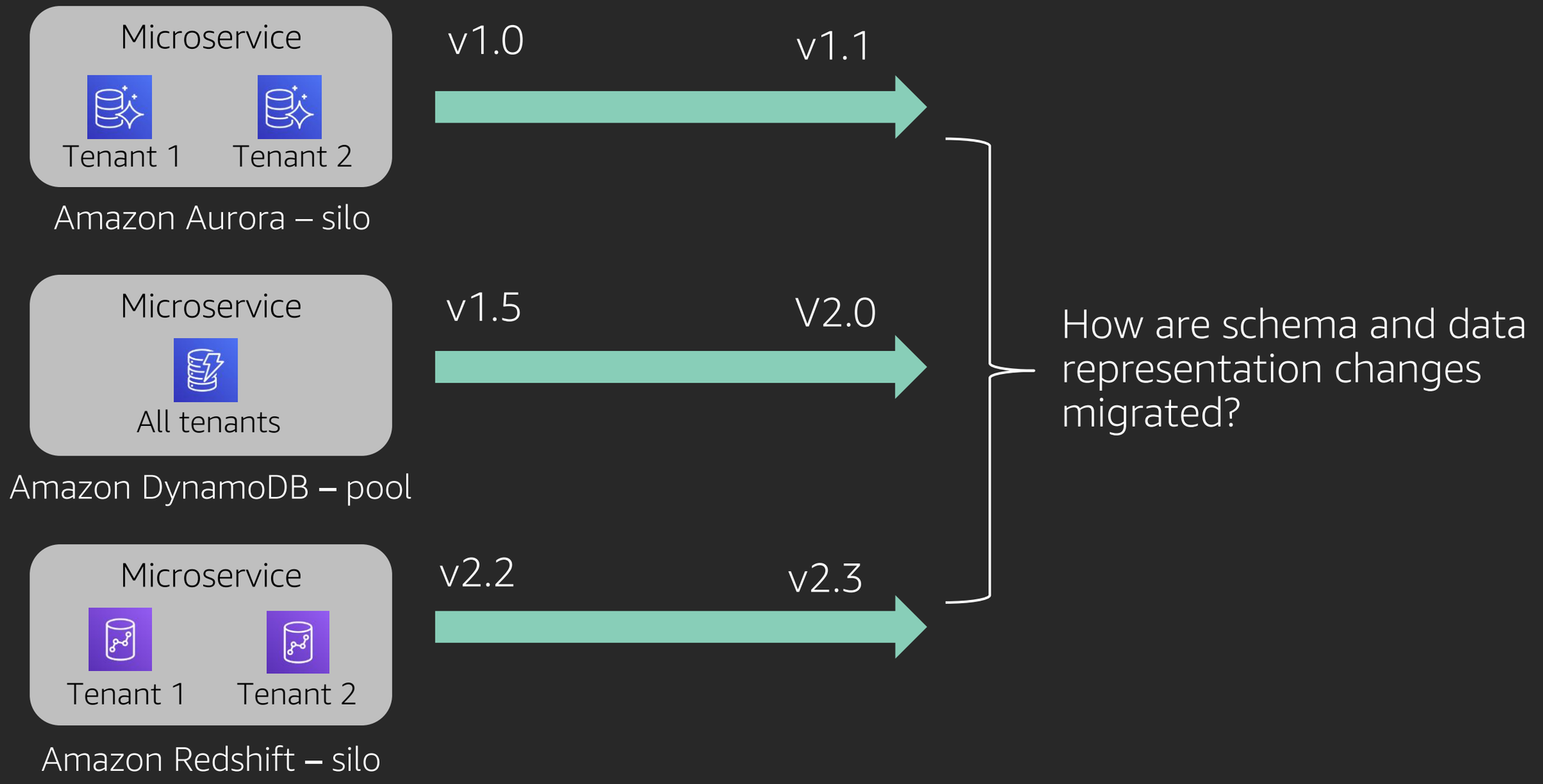


Compliance indicates silo for data families 1 and 3

Agility and cost indicates pool for data family 2

- Balancing partitioning needs with data distribution
- How will this influence cost efficiency and agility?
- Consider backup/restore impacts

Data representation impacts SaaS agility



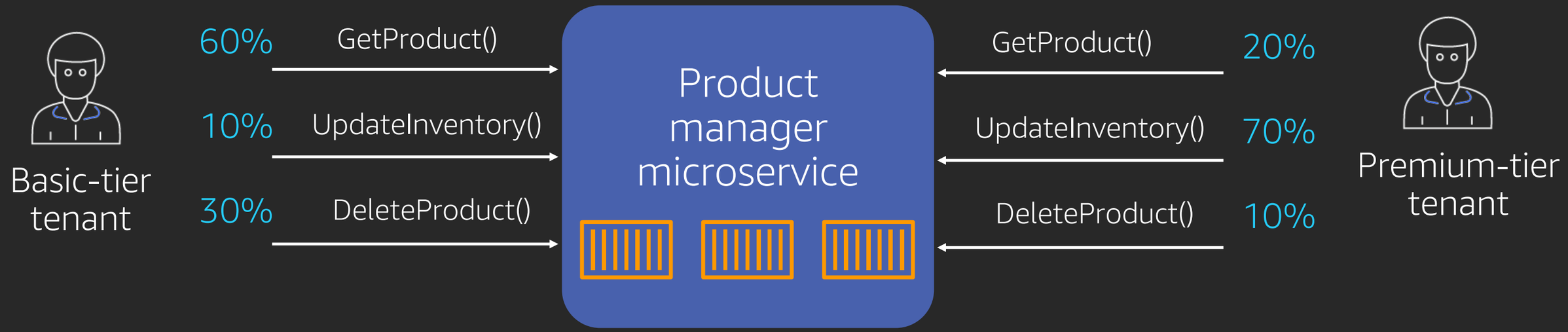
Tenant tiering can shape microservice footprint

Start by profiling your tiers



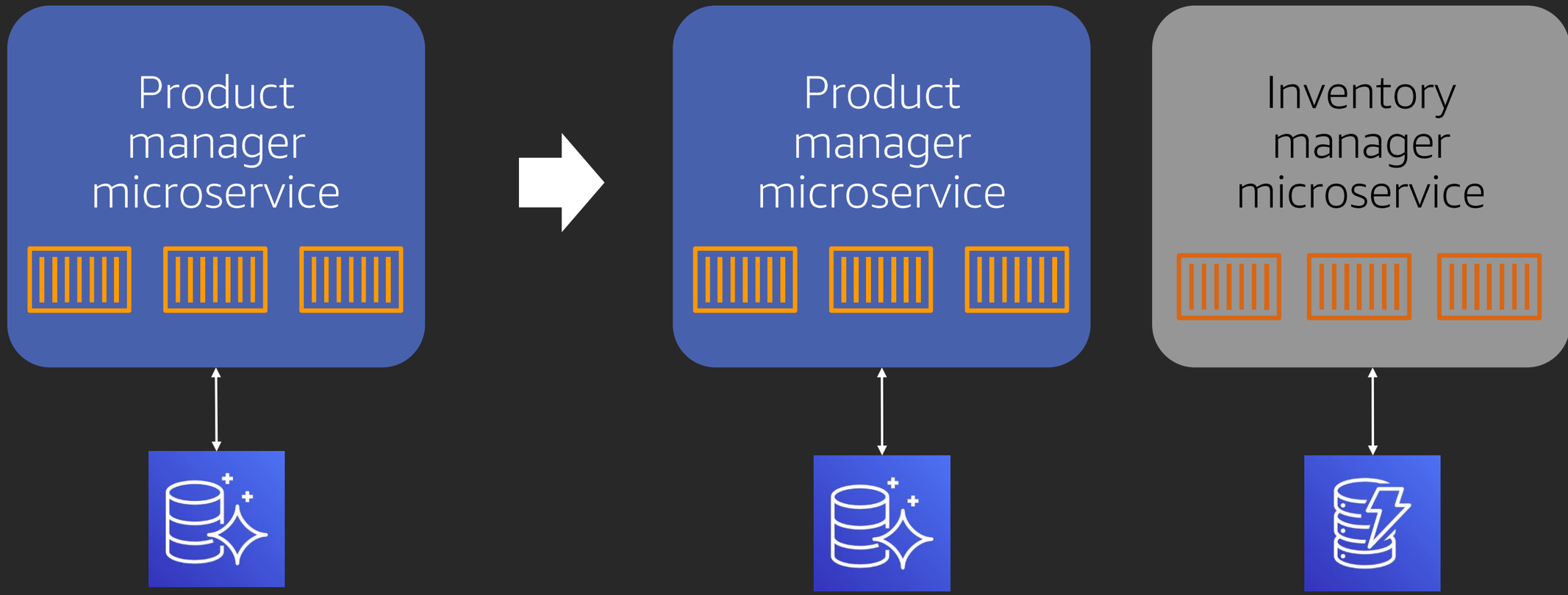
- SLAs vary for each tenant tier
- Workflow and usage patterns could vary by tier
- Data isolation requirements could vary by use case and tier
- Tiering requirements can directly influence decomposition strategies

A simple use case

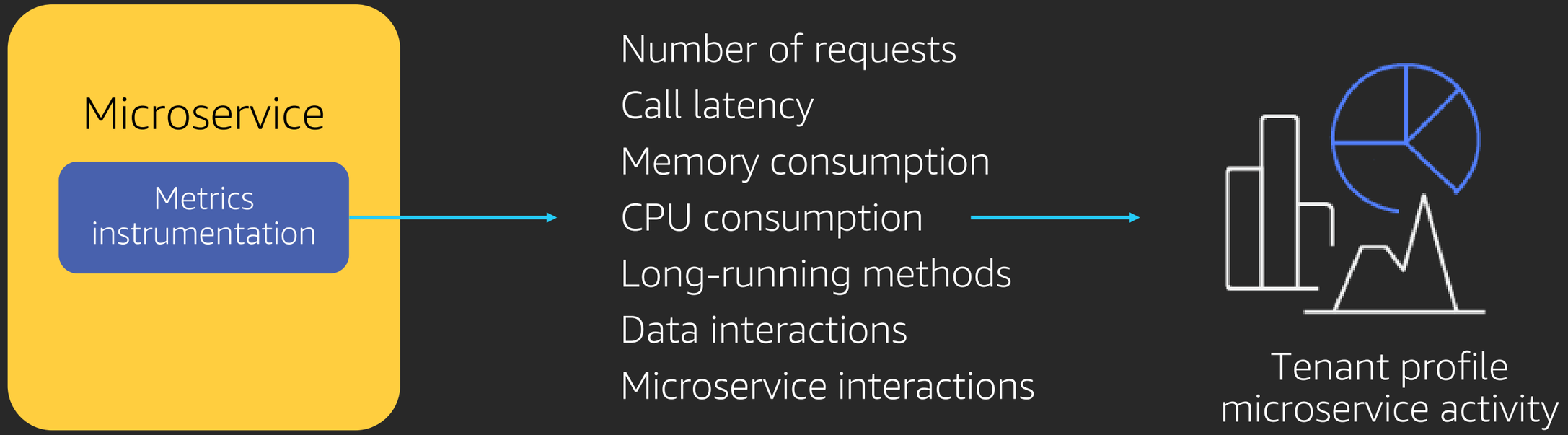


Inventory has very different usage patterns for basic and premium tiers

Tiers may drive a different decomposition model



Using tenant metrics to evolve microservice design



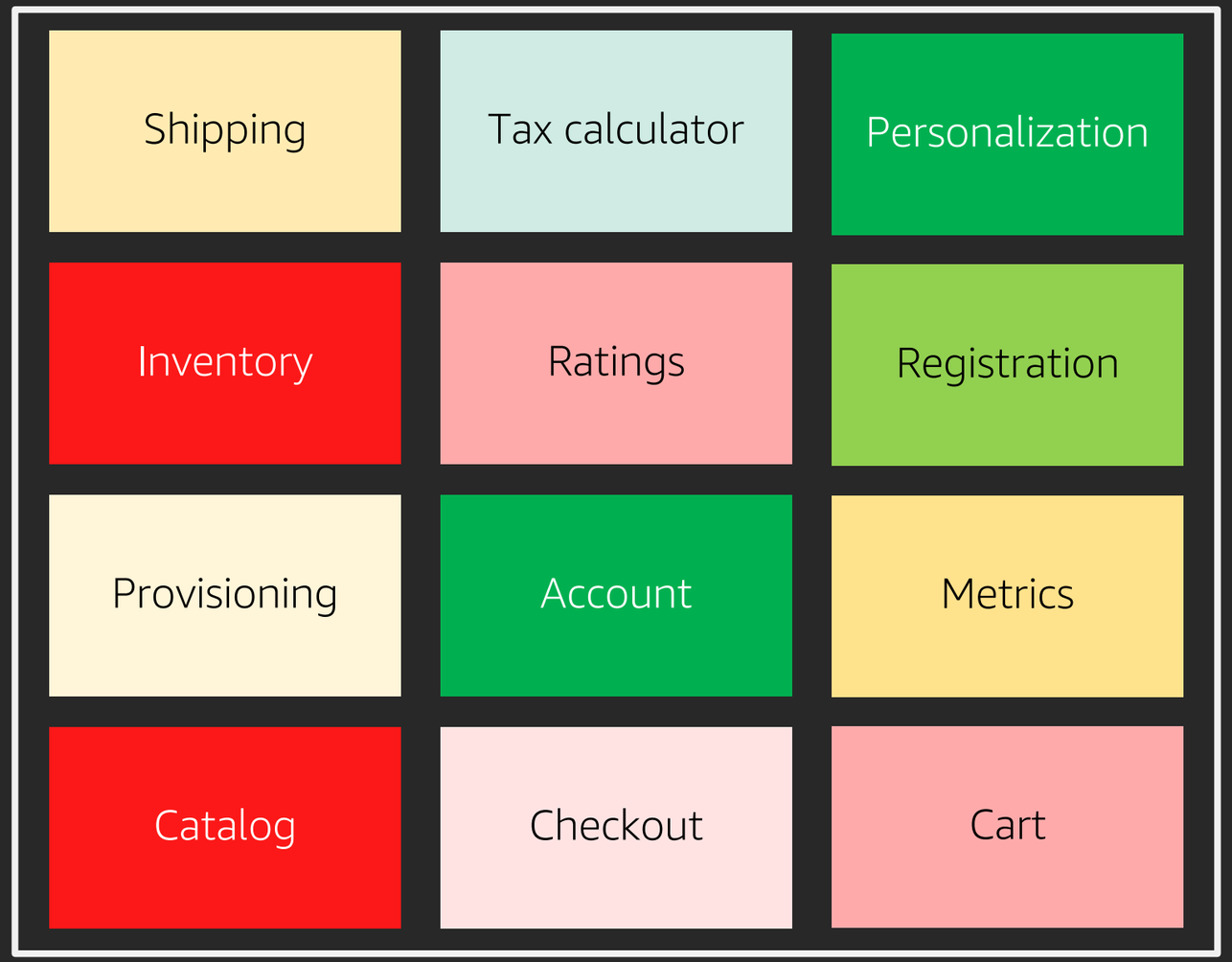
Visualizing microservice activity with tenant context

Activity type

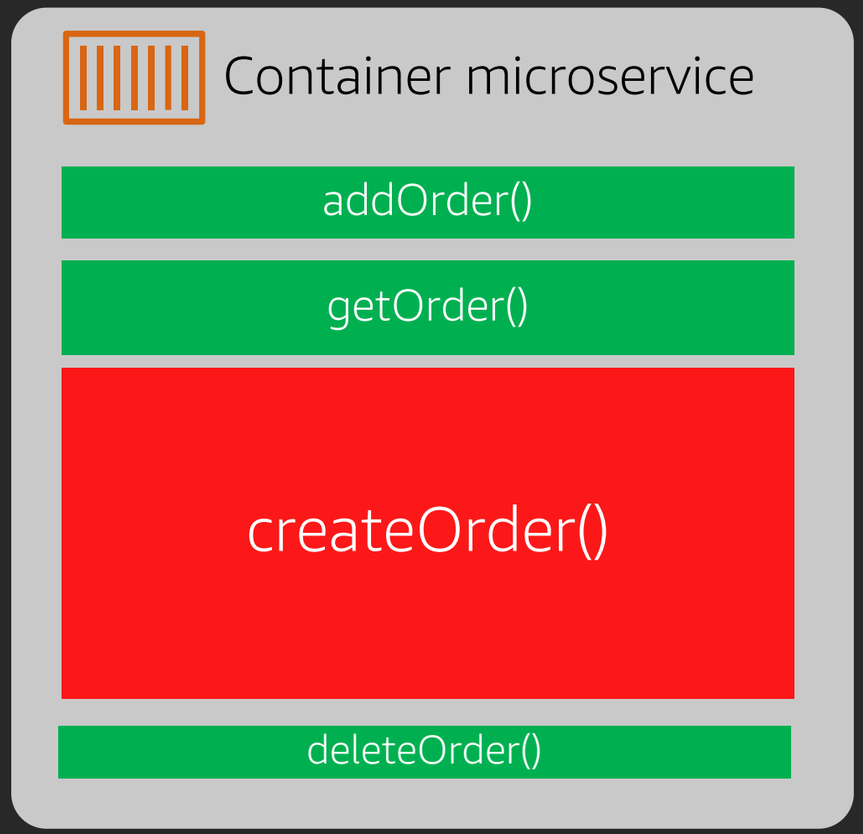
- ○ Number of requests
- ○ Latency of calls
- ○ Storage consumption
- ○ SLA threshold violations
- ◉ Most-active services
- ○ Least-active services

Tenant filters

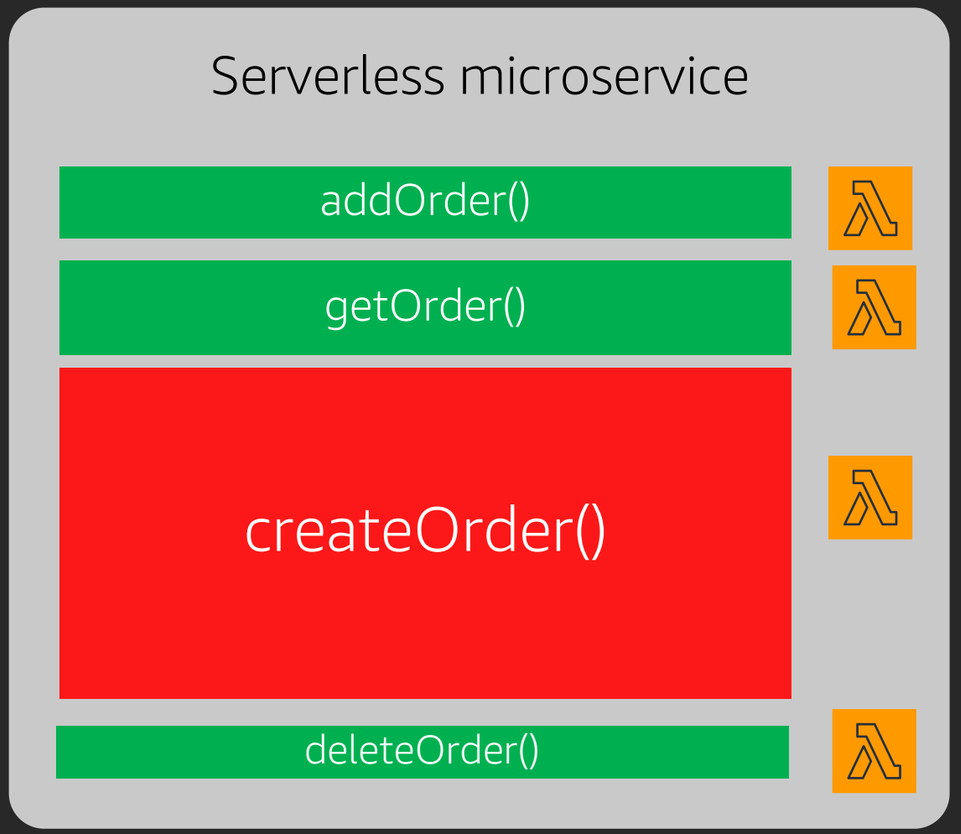
- ◉ All tenants
- ◉ Tenant tier (advanced)
- ○ Individual tenant



Compute model can impact SaaS decomposition model



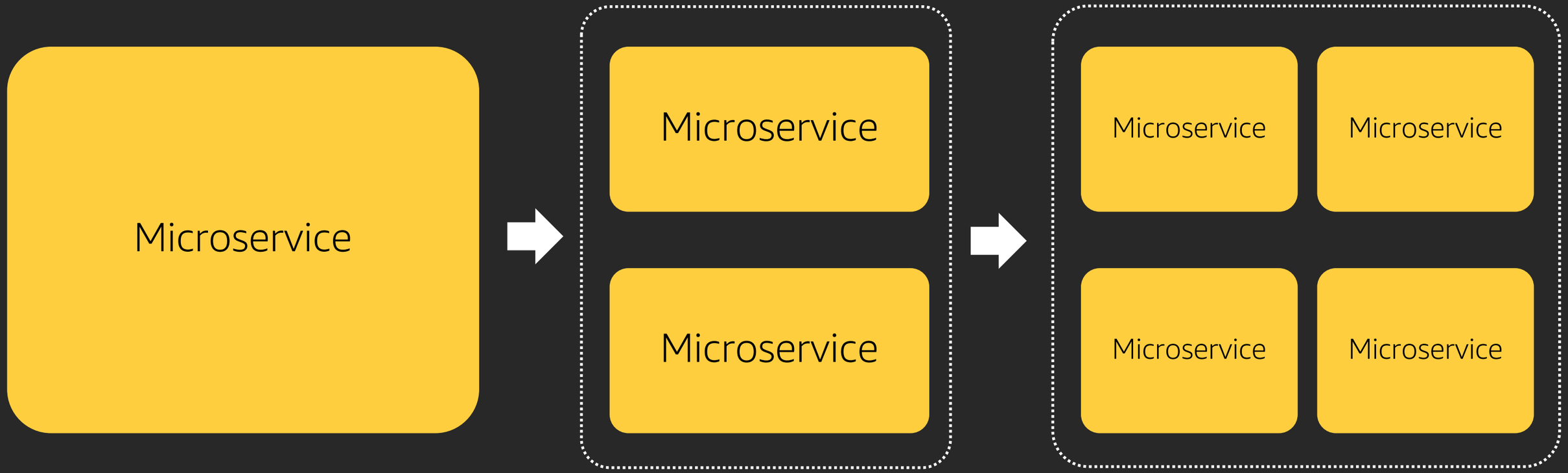
- Unit of scale is a container
- Unused capacity impacts cost per tenant



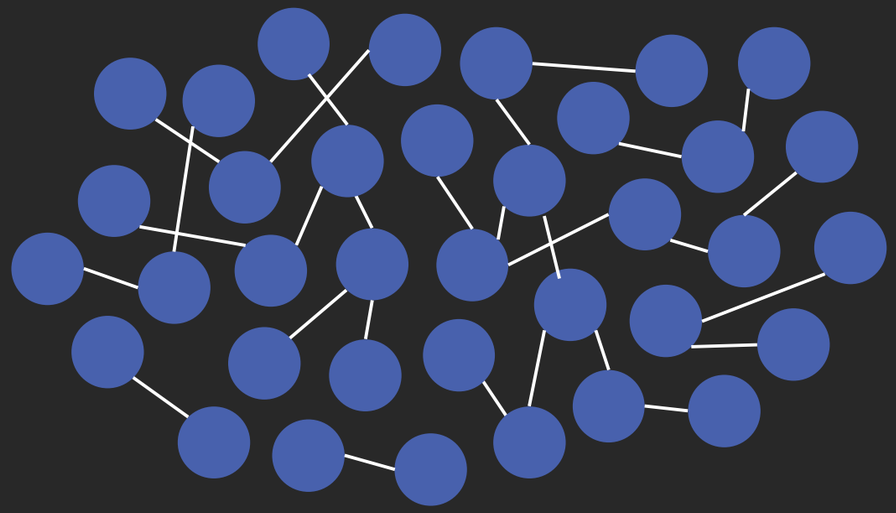
- Unit of scale is a function
- Pay for parts of the service you consume

Let services “earn” their way out

Start with coarse-grained and decompose more based on actual data



Keep your fail fast mindset



- Don’t over invest in developing rich, theoretical models
- Be thoughtful, but expect to get some services wrong
- Lean on the agility of your environment
- Rarely used functionality may not require fine-grained decomposition
- Find a balance that leans on real code and data to inform your design

Takeaways

- Deployment and isolation requirements may impact decomposition
- Don’t assume data in a domain is tightly coupled
- Factor tiers, SLAs, and tenant load profiles into your design
- Don’t underestimate the impact of fault tolerance
- Use metrics to evolve your decomposition
- Consider starting with coarse-grained services
- Decomposition is not one-time event—don’t over analyze
- Multi-tenant workloads introduce new decomposition considerations

Related SaaS sessions

aws partner network | saas factory

Breakouts

- ARC410 – Serverless SaaS deep dive: Building serverless SaaS on AWS
- GPSTEC337 – Architecting multi-tenant PaaS offerings with Amazon EKS

Chalk talks

- ARC413 – SaaS metrics deep dive: A look inside multi-tenant analytics
- ARC305 – Migrating single-tenant applications to multi-tenant SaaS
- GPSTEC306 – Reinventing your technology product strategy with a SaaS delivery model
- API308 – Monolith to serverless SaaS: Migrating to multi-tenant architecture

Related SaaS sessions (cont'd)

aws partner network | saas factory

Workshops

- SVS303 – Monolith to serverless SaaS: A hands-on service decomposition
- ARC308 – Hands-On SaaS: Constructing a multi-tenant solution on AWS

Builder sessions

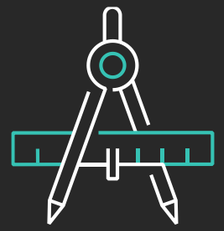
- ARC405 – Building multi-tenant-aware SaaS microservices

Lightning talks

- DEM142 – SaaS migration strategies
- DEM143 - Multi-region SaaS: Staying agile in a multi-geography model

Learn to architect with AWS Training and Certification

Resources created by the experts at AWS to propel your organization and career forward



Free foundational to advanced digital courses cover AWS services and teach architecting best practices



Classroom offerings, including Architecting on AWS, feature AWS expert instructors and hands-on labs



Validate expertise with the **AWS Certified Solutions Architect - Associate** or **AWS Certification Solutions Architect - Professional** exams

Visit aws.amazon.com/training/path-architecting/

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws training and certification

Thank you!

Tod Golding

todg@amazon.com

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws



Please complete the session survey in the mobile app.

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.