



AWS  
re:Invent

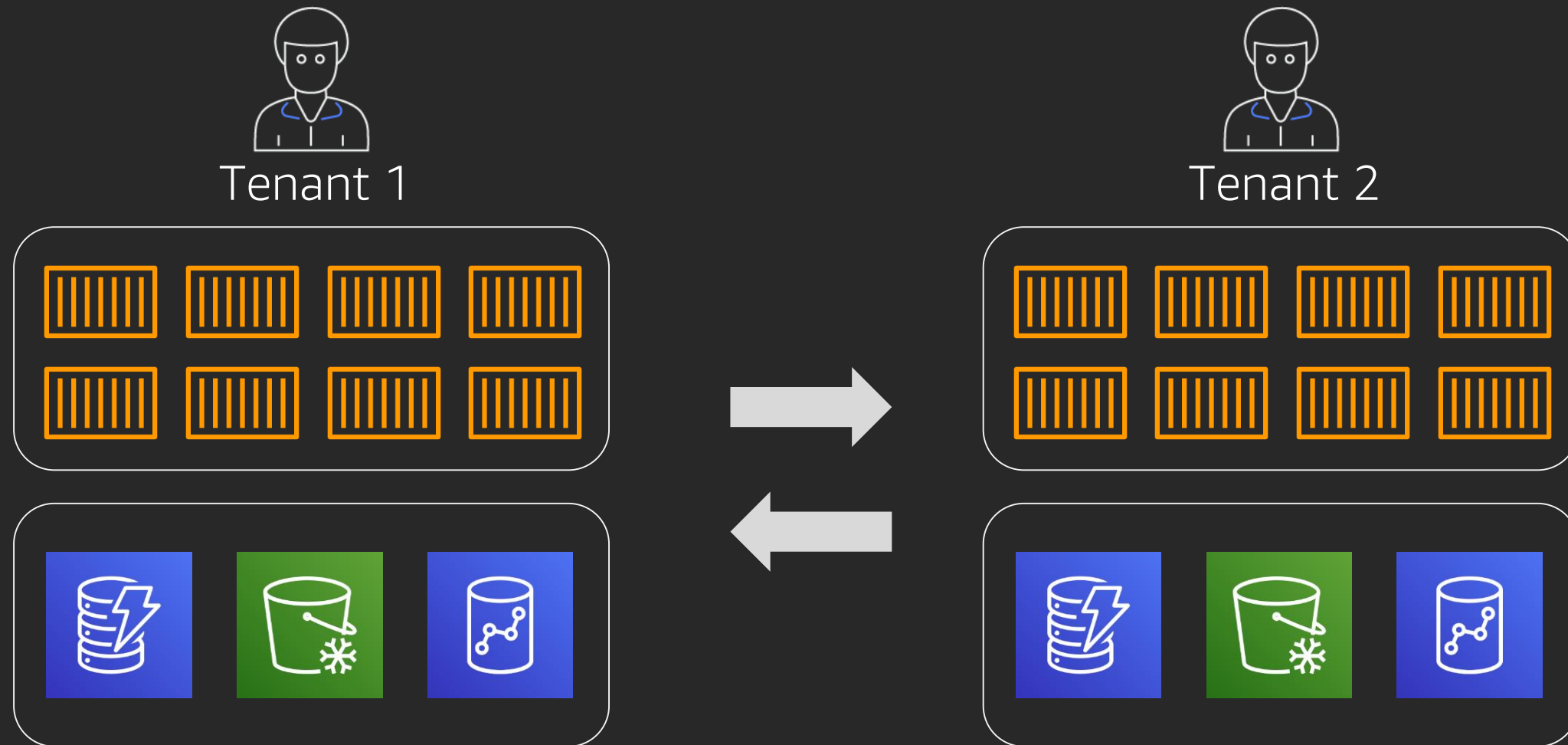
**A R C 3 7 2 - P**

# SaaS tenant isolation patterns

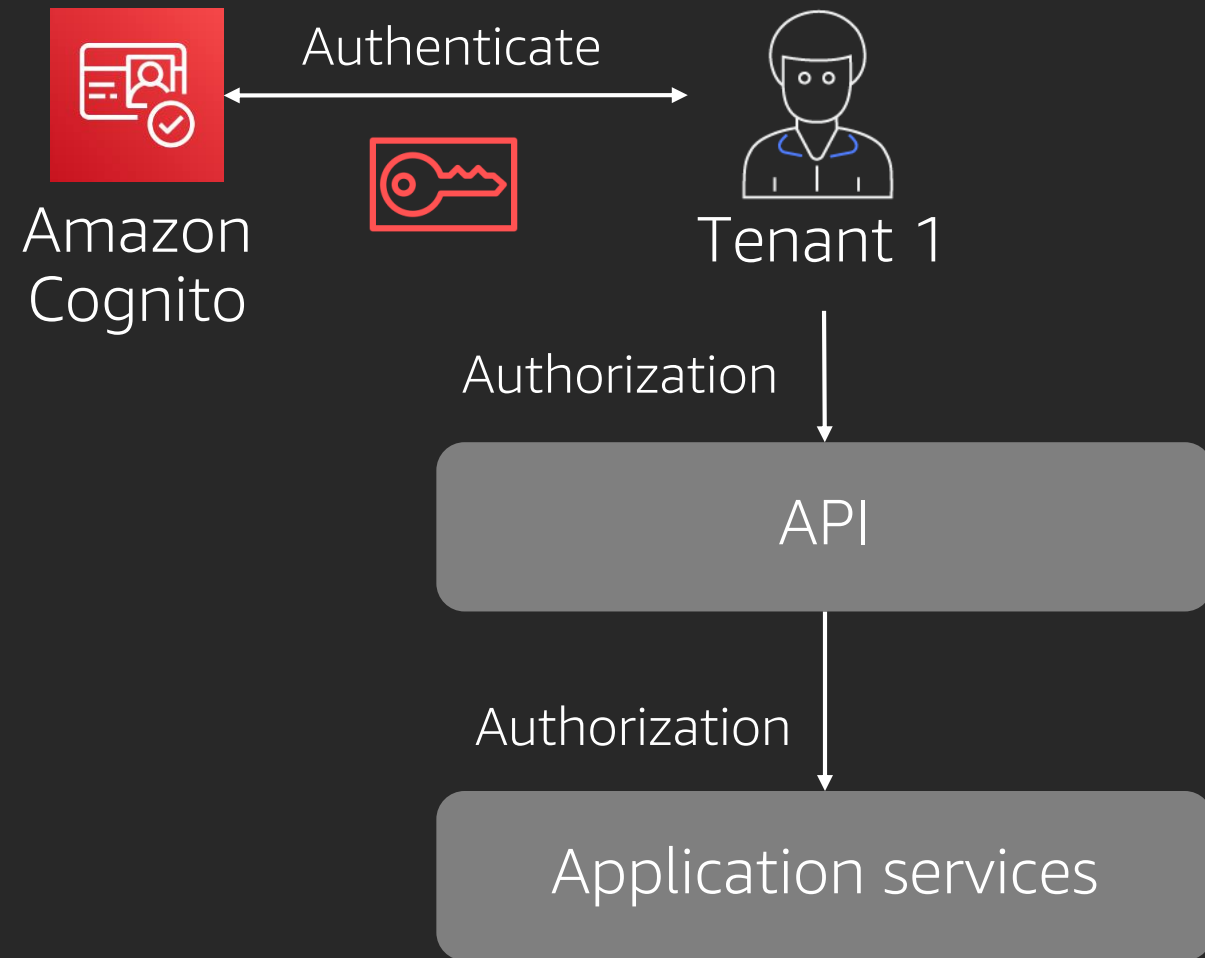
**Tod Golding**

Principal Partner Solutions Architect  
Amazon Web Services

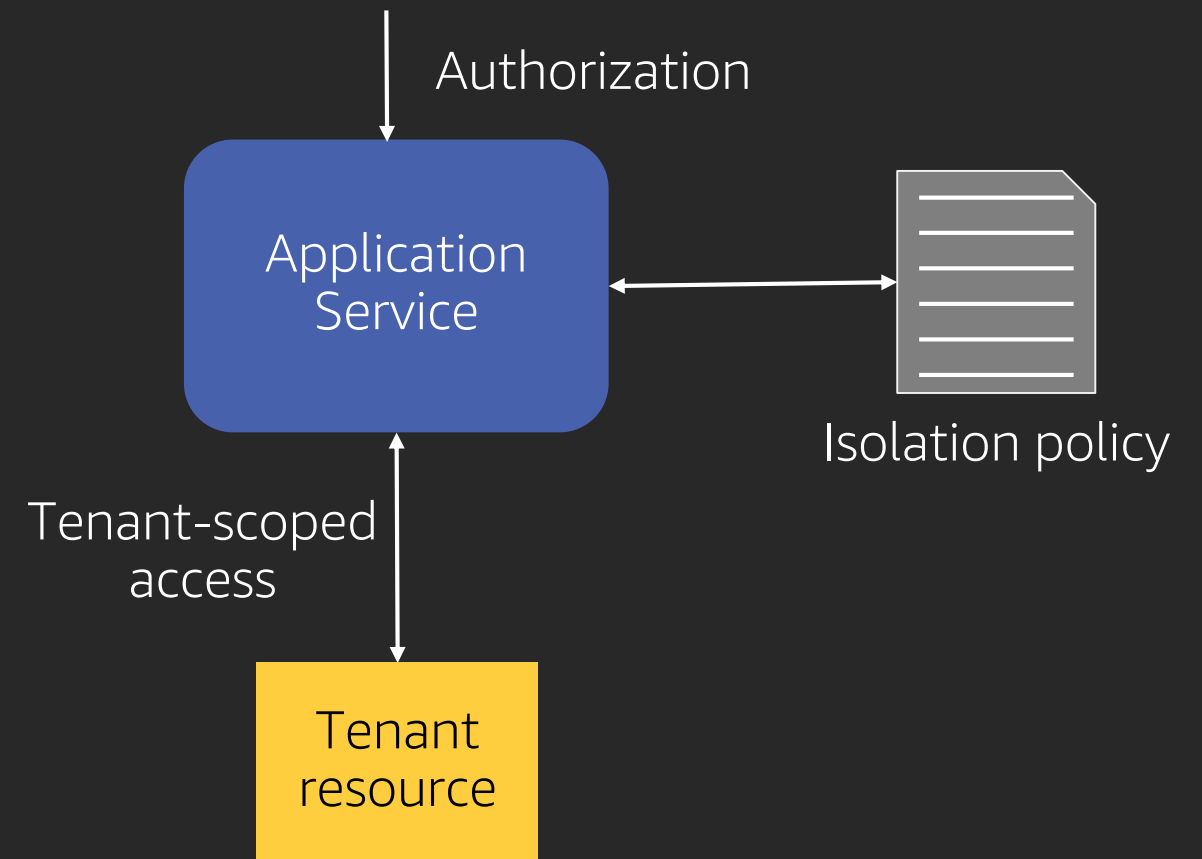
# Fundamental goal of SaaS isolation



# Authentication and authorization ≠ isolation

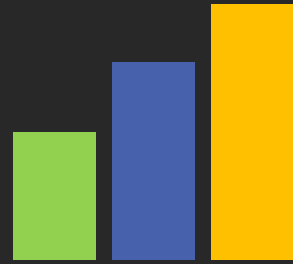


Identity/role validation at every layer



Tenant-level isolation enforcement

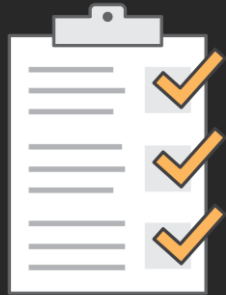
# Isolation strategy drivers



Tiering strategy



Noisy neighbor



Compliance  
requirements



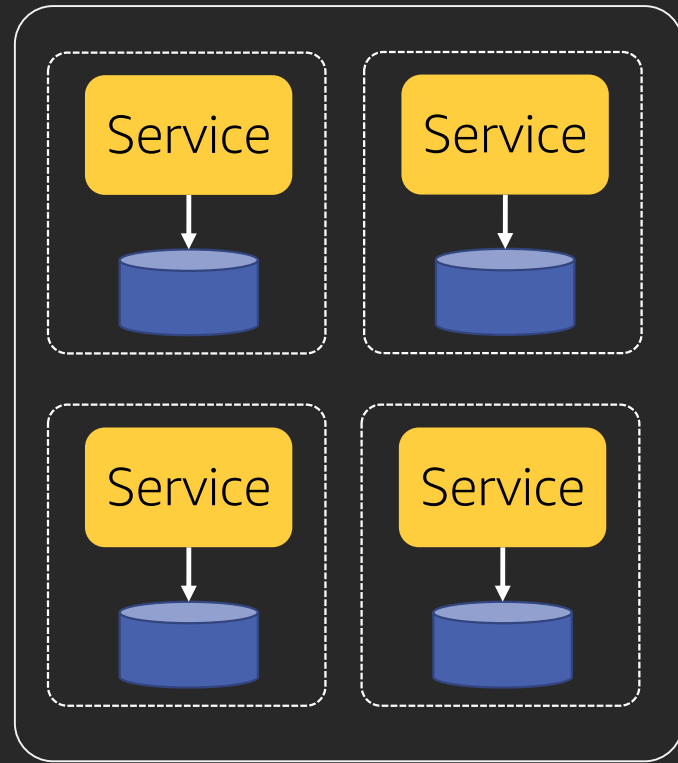
Legacy  
architecture



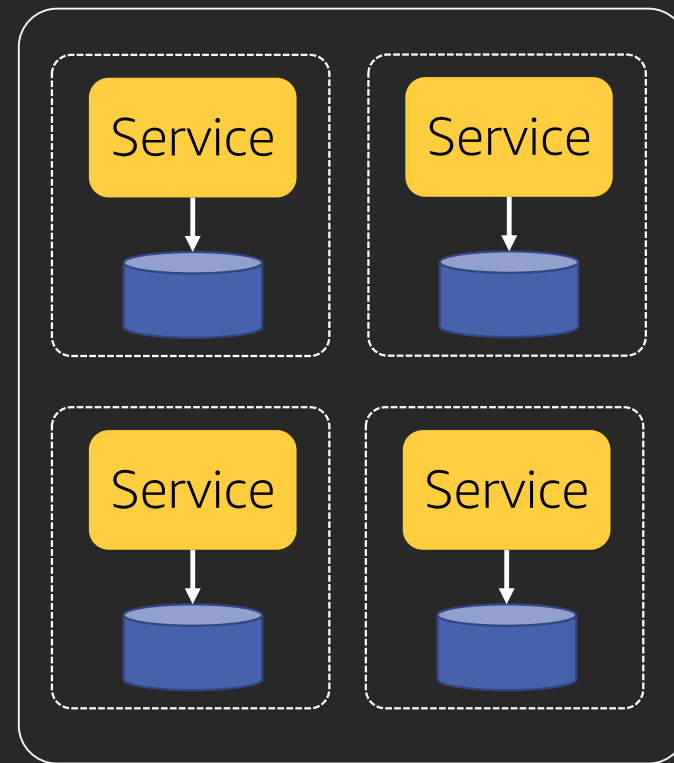
Opportunity

# Multiple flavors of isolation

  
Tenant 1



  
Tenant 2



Isolation through siloed infrastructure  
(silo model)



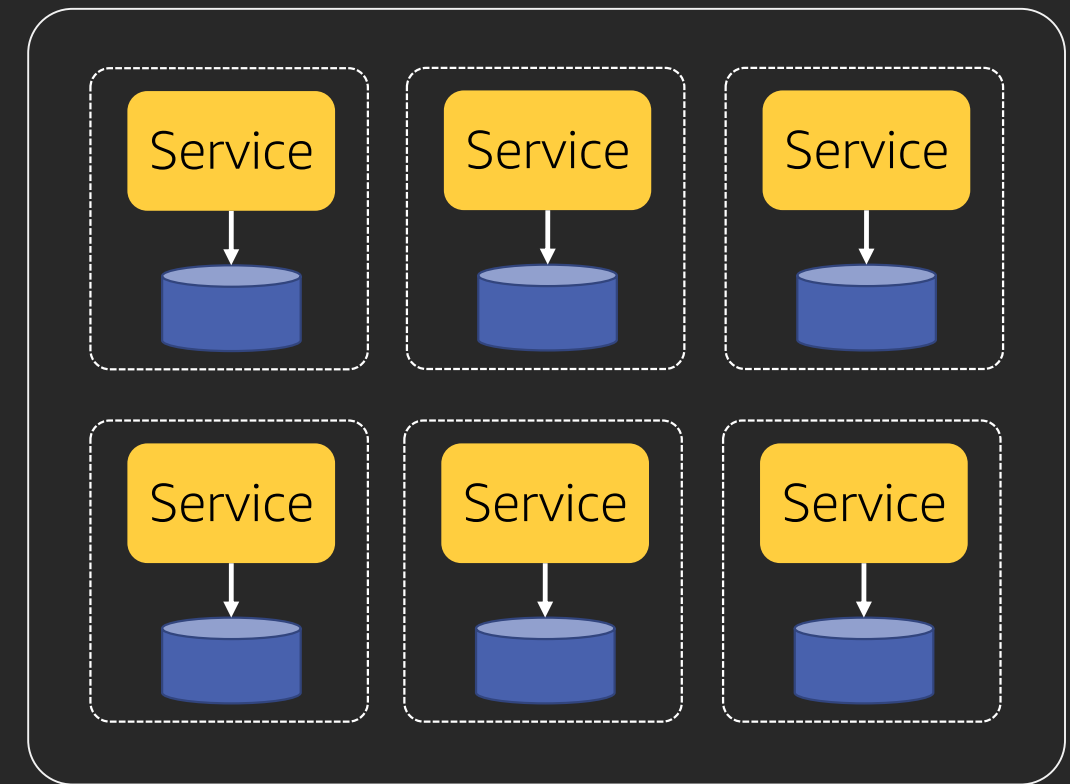
Tenant 1



Tenant 2



Tenant 3



Isolation through runtime policies  
(pool model)

# Supporting multiple isolation models



**Ted Roe**

**Compliance- and security-focused tenant**

Ted has a strong belief that his business requires a very stringent approach to security. Any notion of sharing infrastructure would represent a nonstarter for Ted and his users. Ted also requires tighter control over how and when upgrades are applied in his environment.



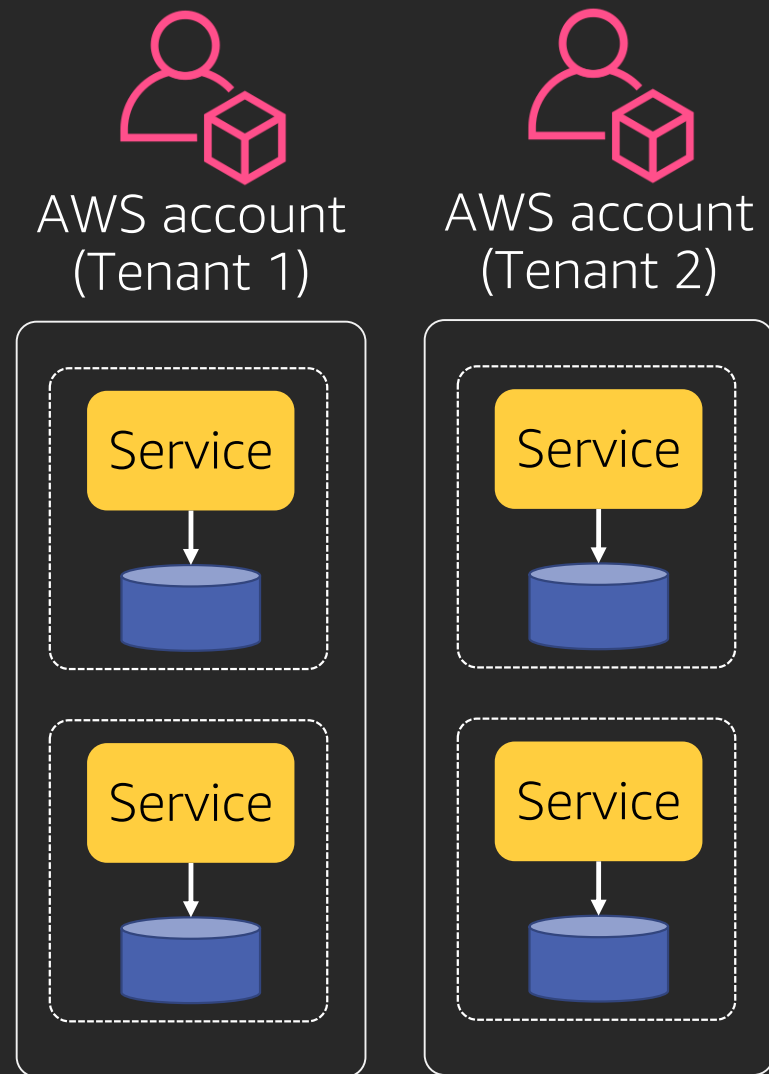
**Mary Jackson**

**Cost- and ease-of-use focused tenant**

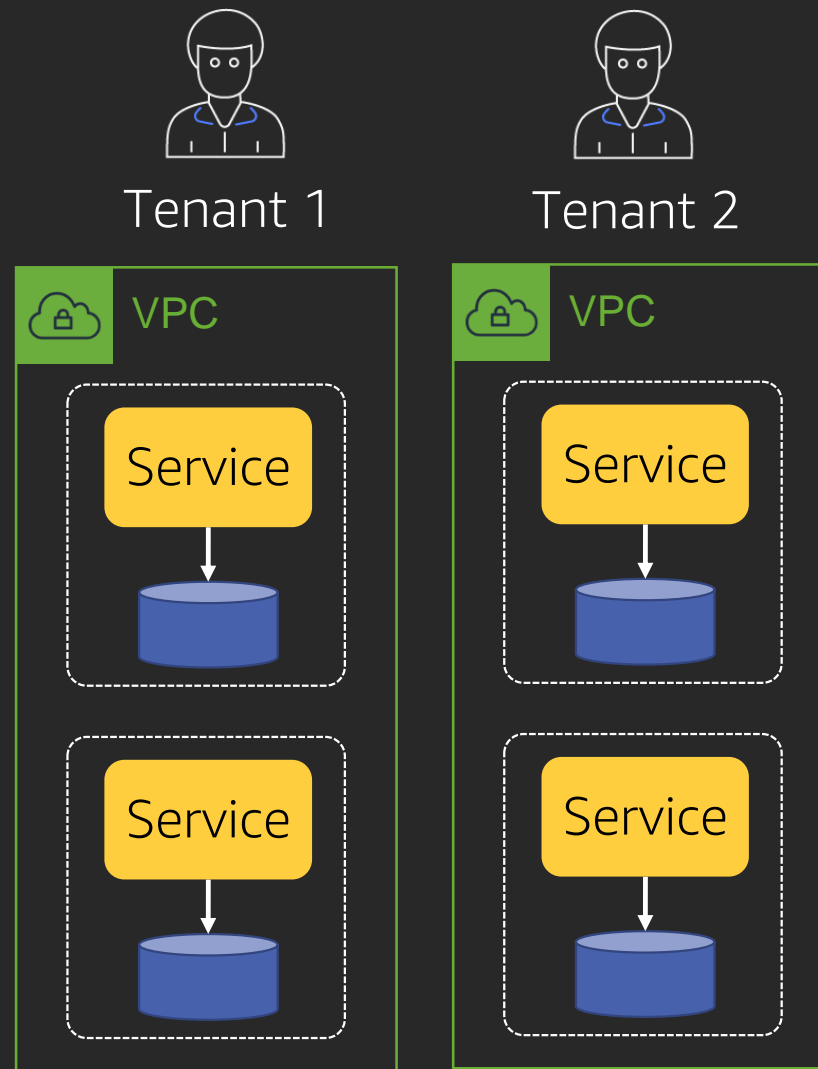
Mary has a small to medium-size company and is squarely focused on value. Cost and features were at the core of her decision-making process. While security is important to Mary, she presumes it's baked into the product. She likes to see new features appearing all the time.

Should your isolation strategy support both?

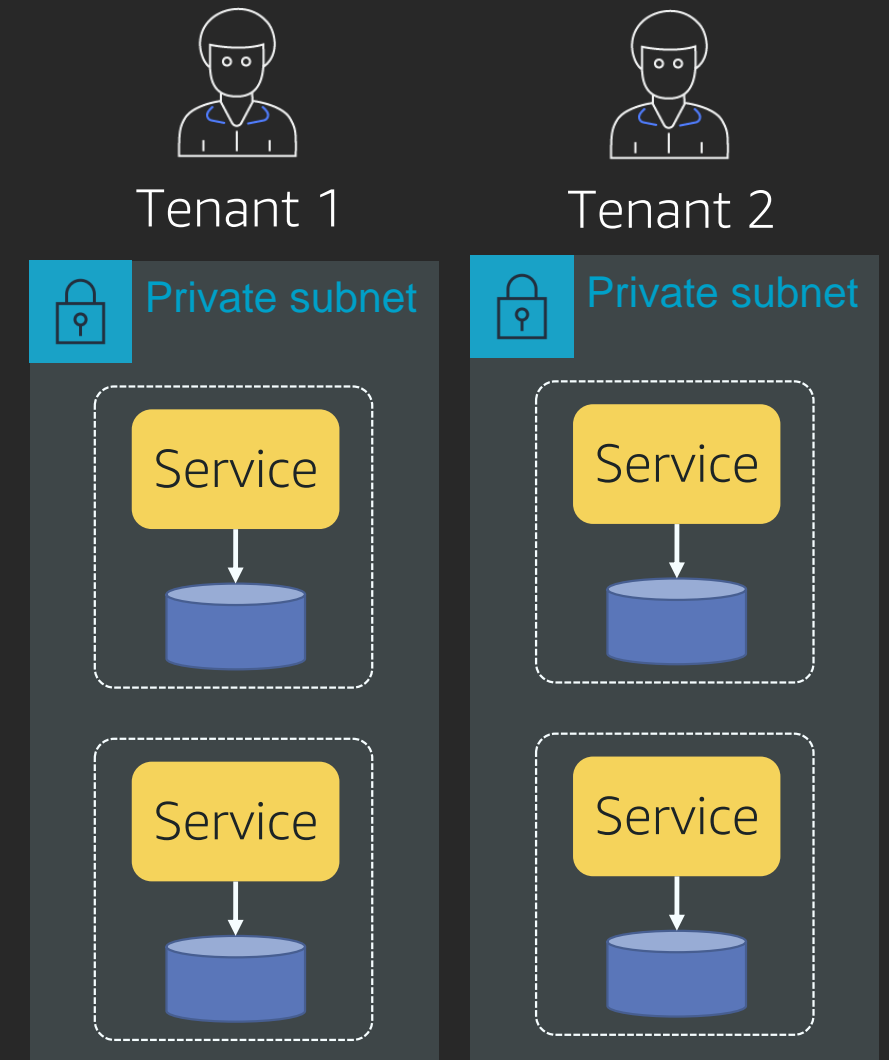
# Silo isolation strategies



Account per tenant



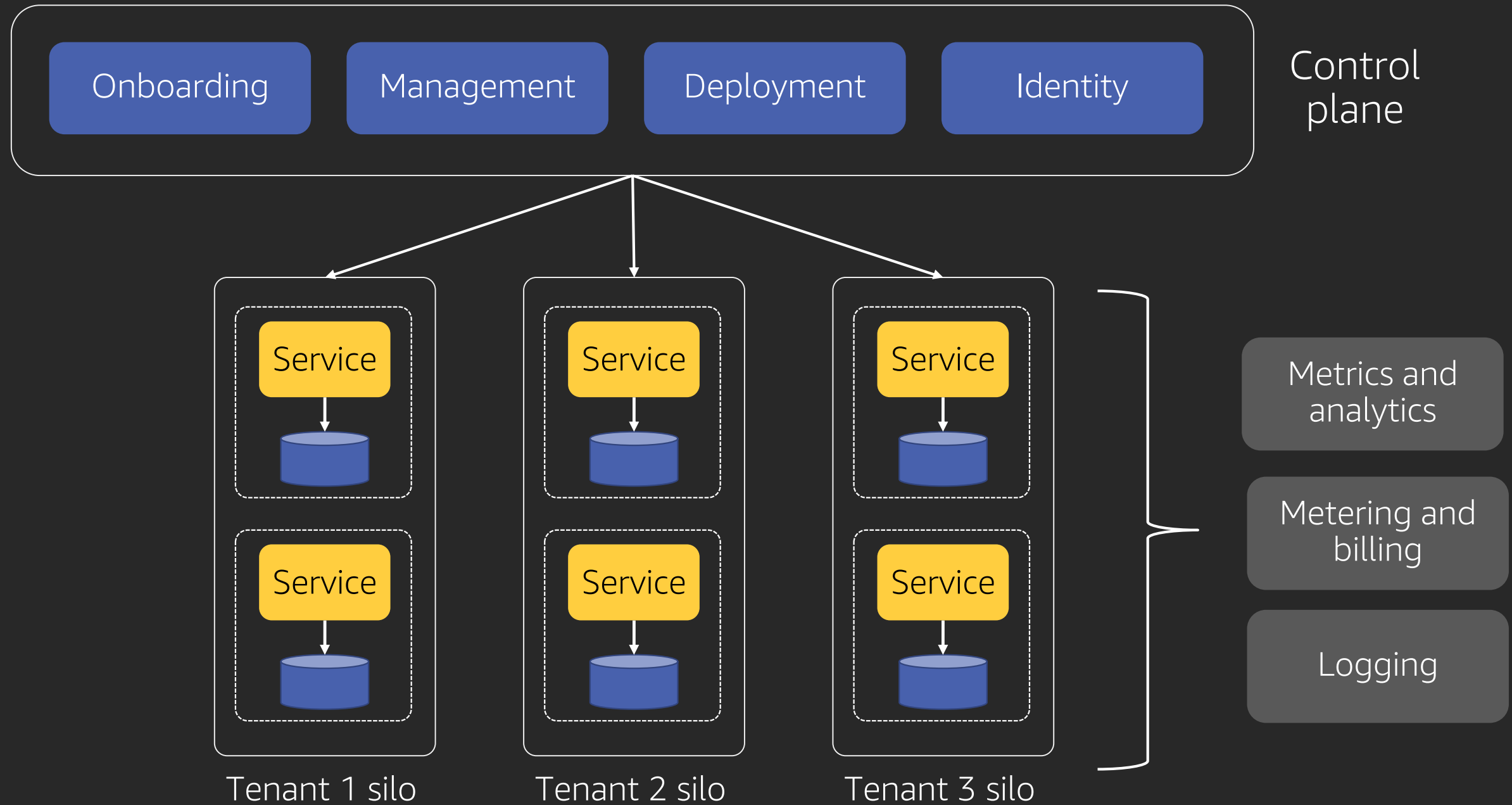
VPC per tenant



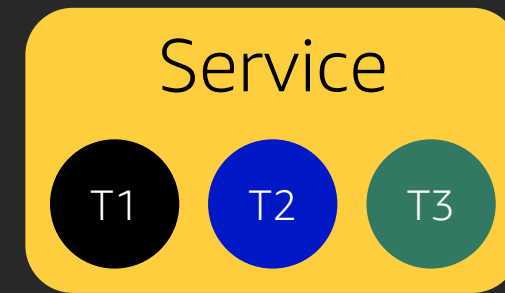
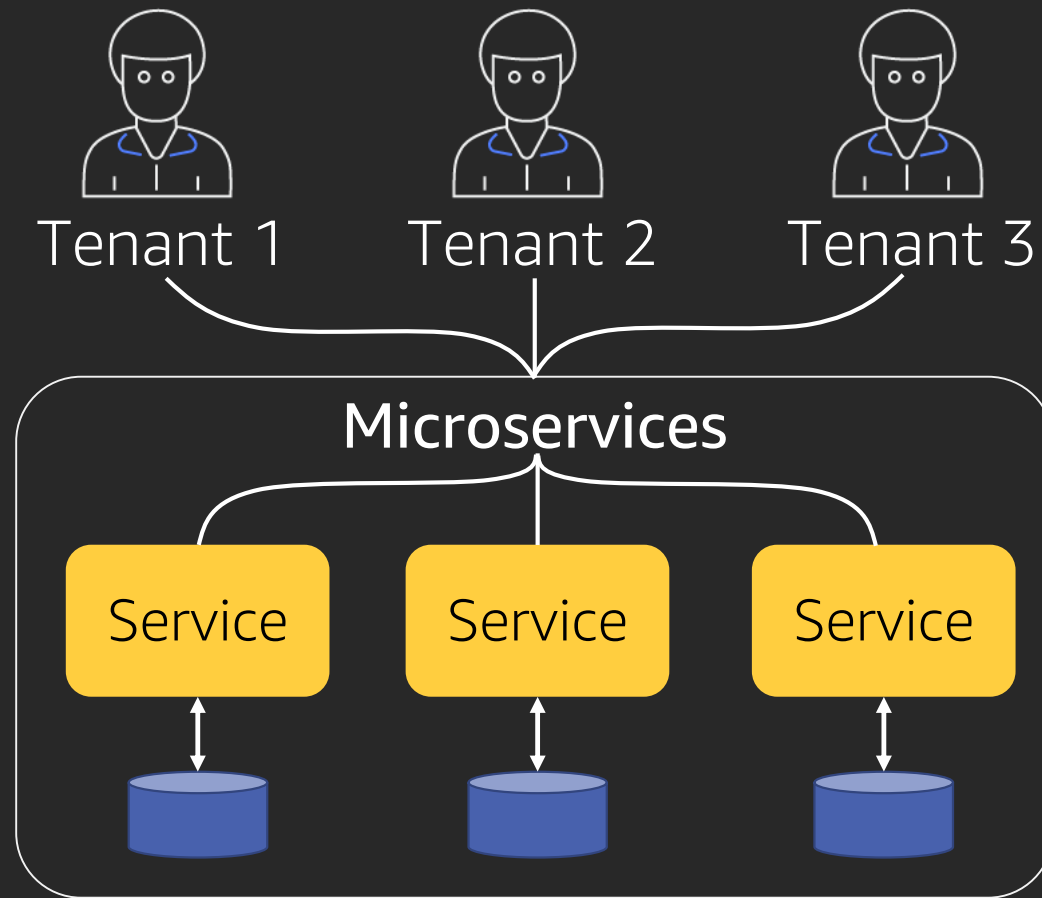
Subnet per tenant



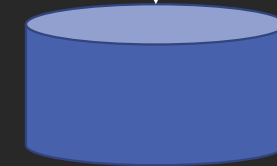
# Siloed infrastructure, one pane of glass



# Pool-based isolation adds a new wrinkle



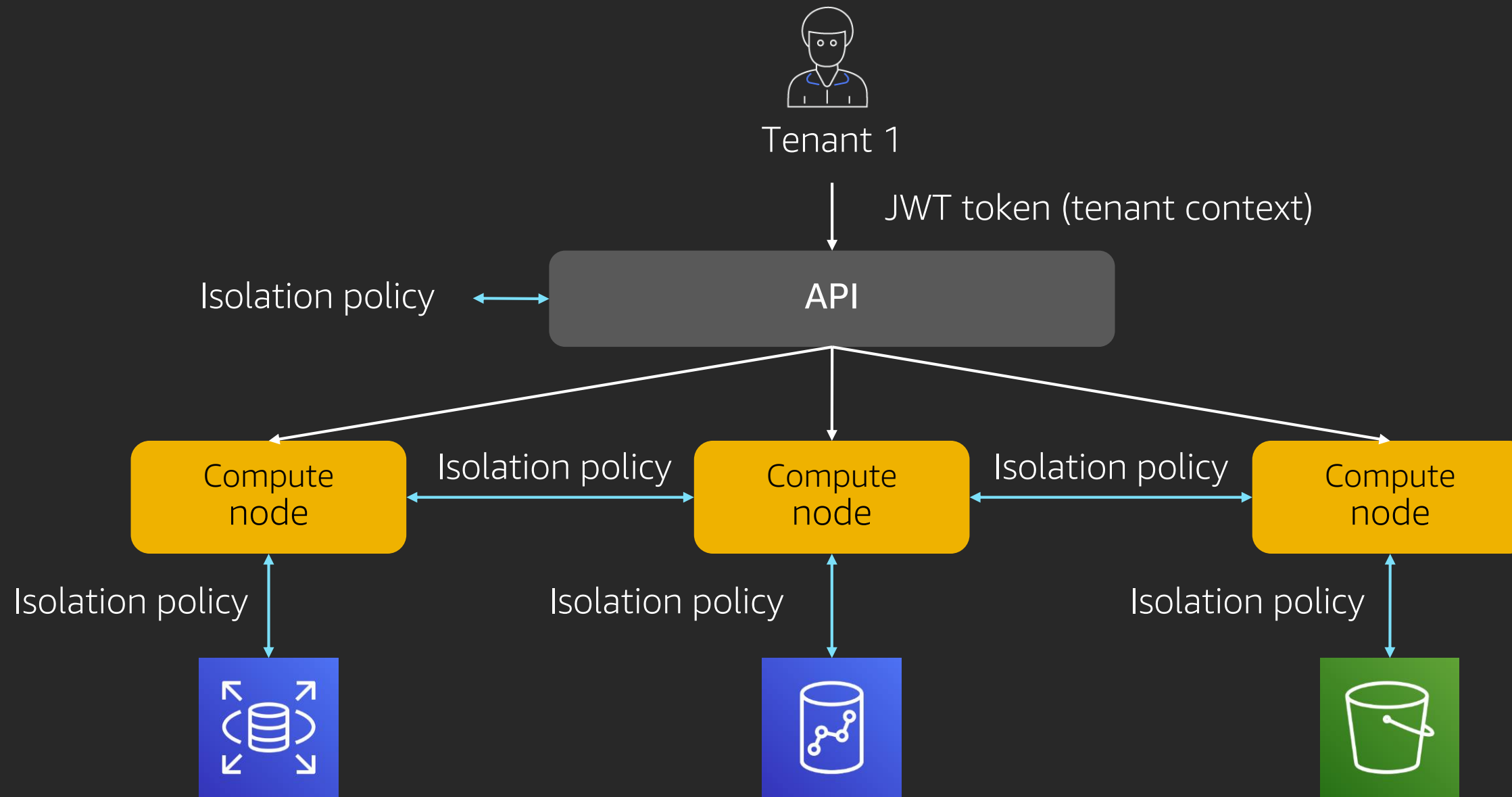
What is the unit of isolation?



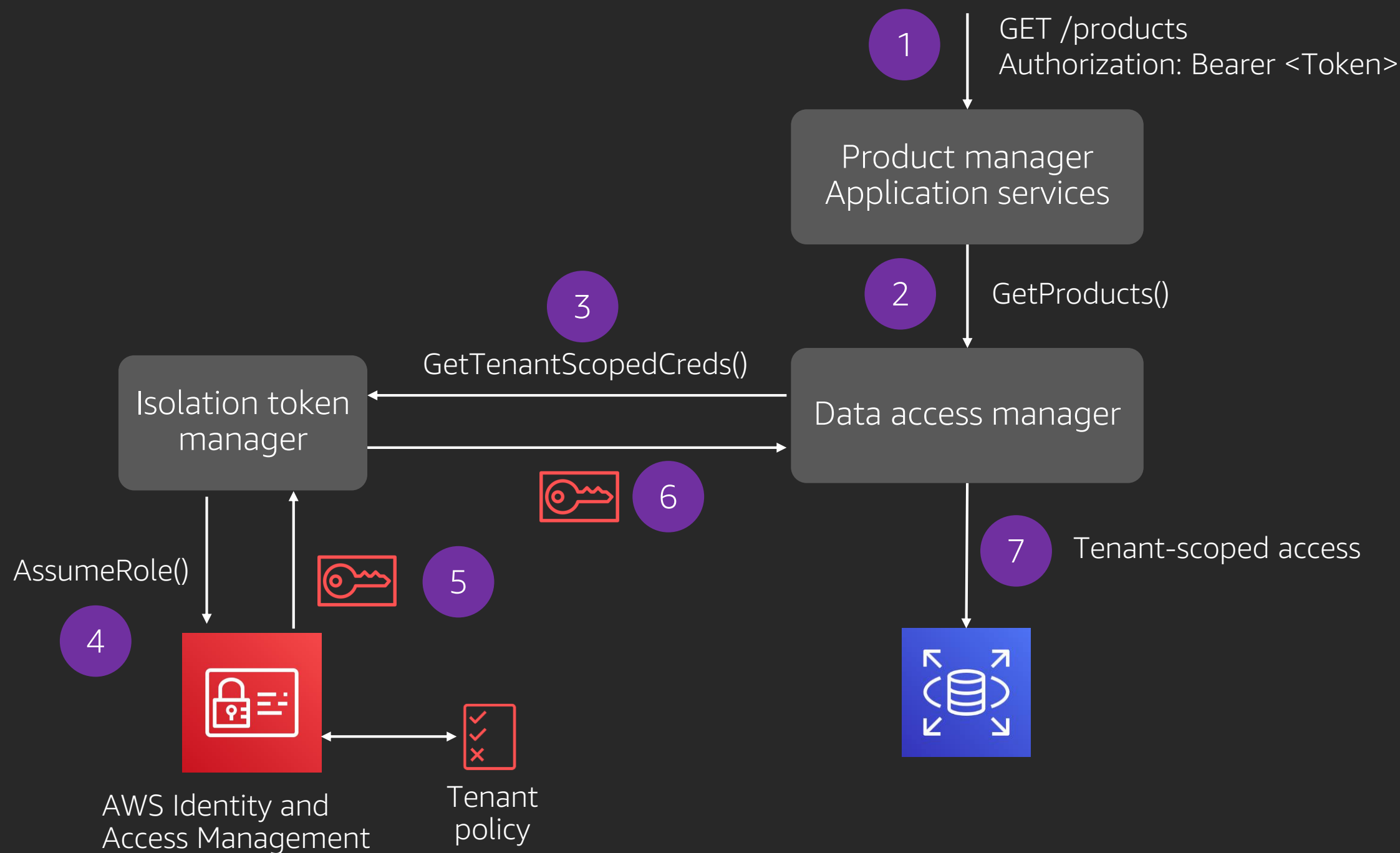
T1	Golf club
T2	Golf bag
T1	Golf cart
T3	Golf bag

- How do you isolate resources that are shared?
- Can't rely on well-behaved code

# Introducing runtime policies to control access



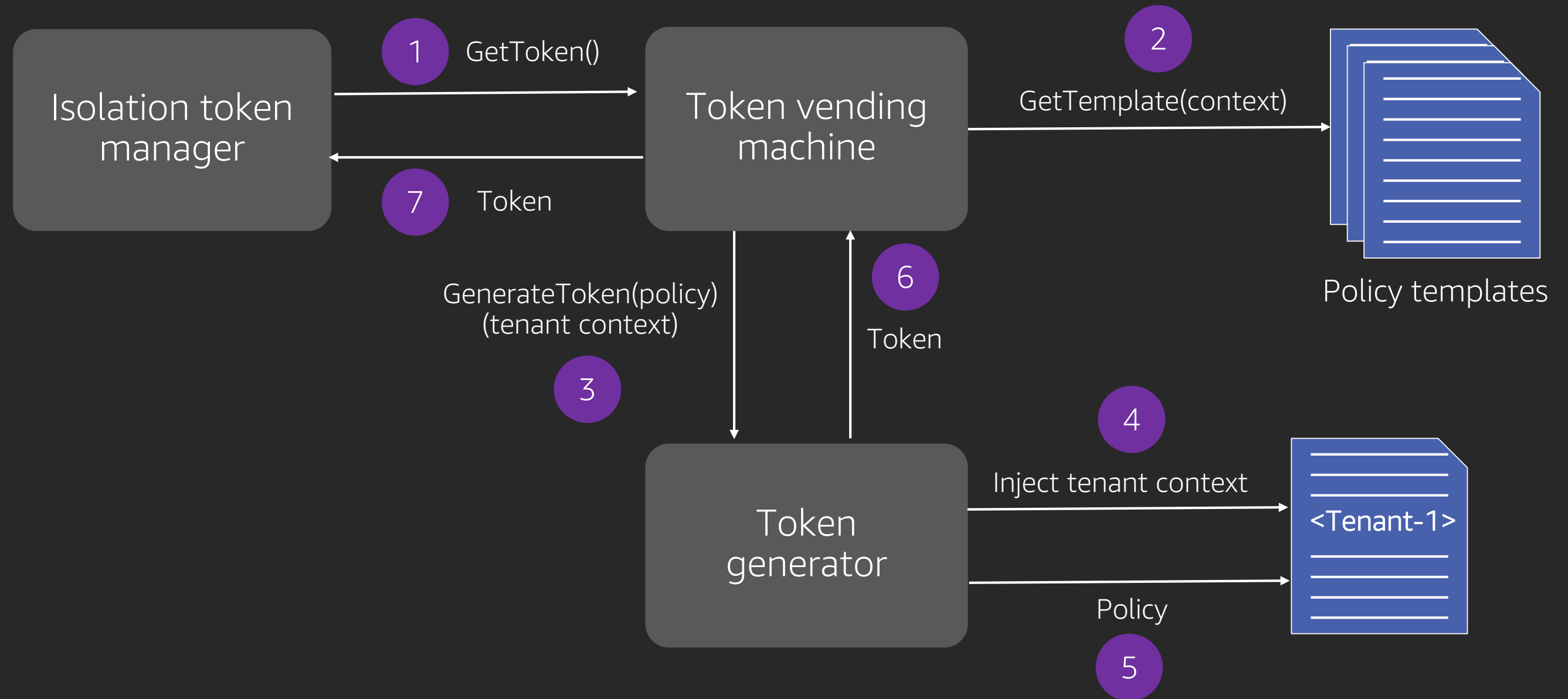
# Runtime scoped access with IAM



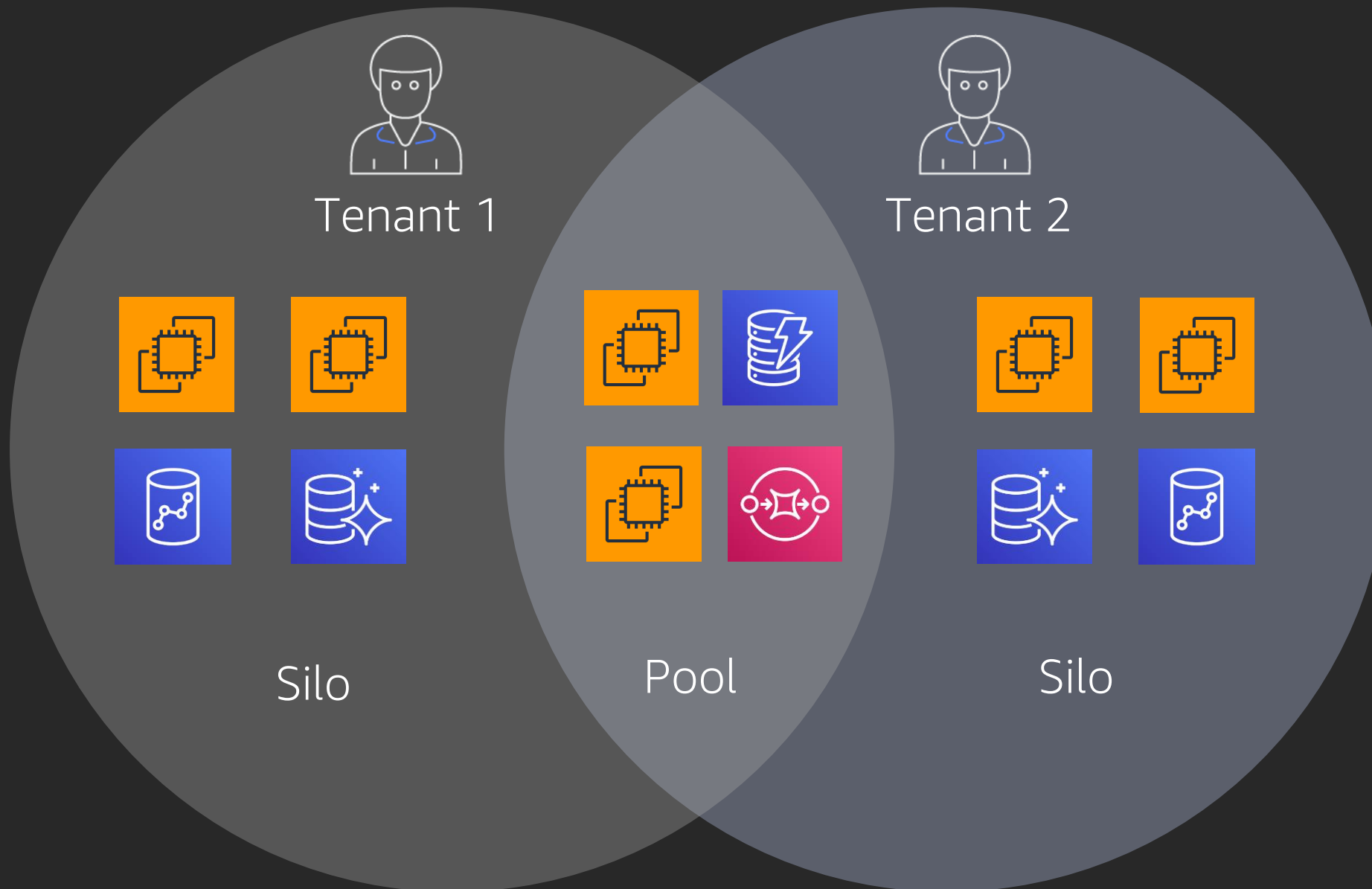
JWT token

```
{  
  "tenantId": "8391"  
  "role": "Admin"  
}
```

# Overcoming limits: Dynamically generated policies

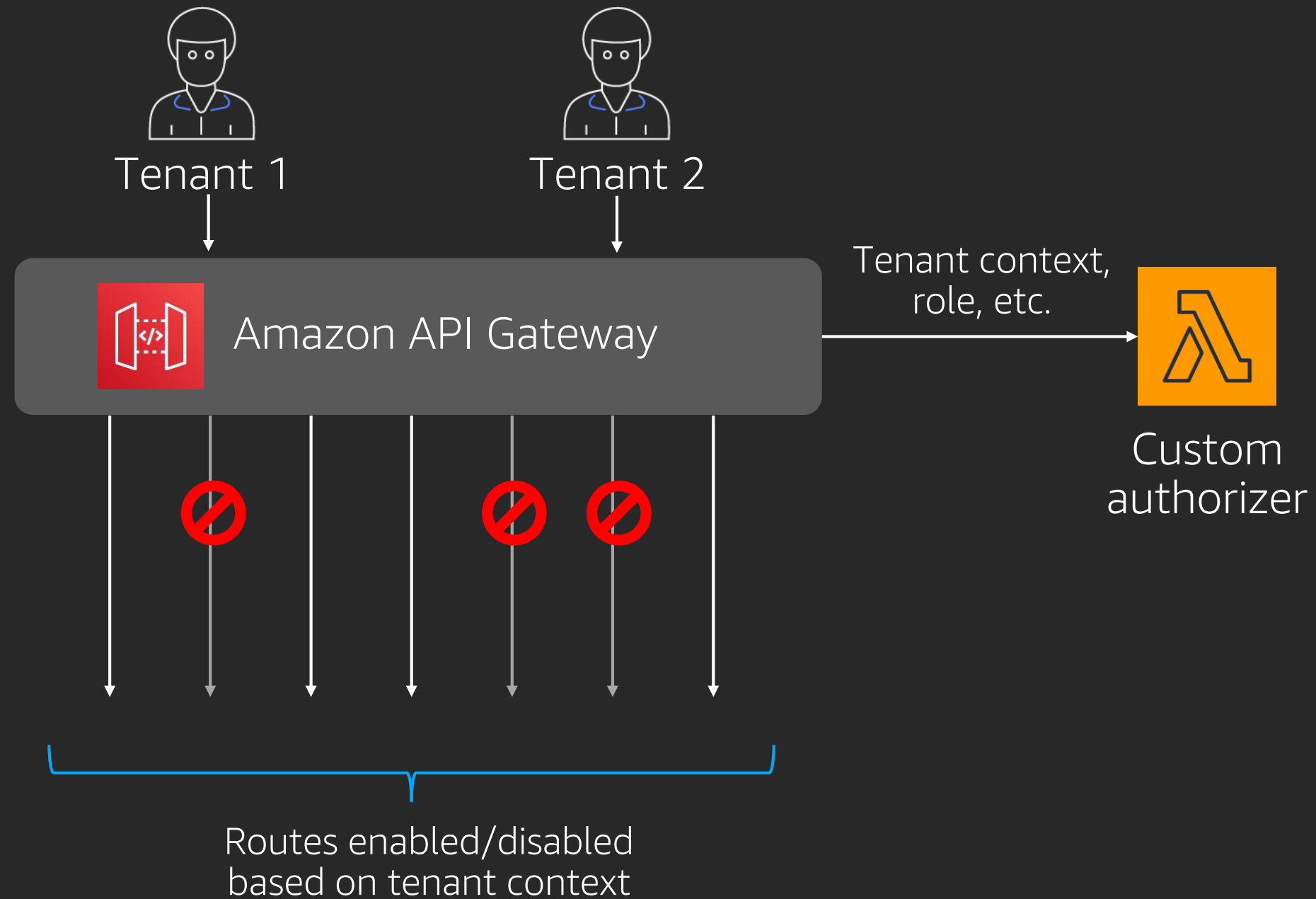


# The core theme of isolation

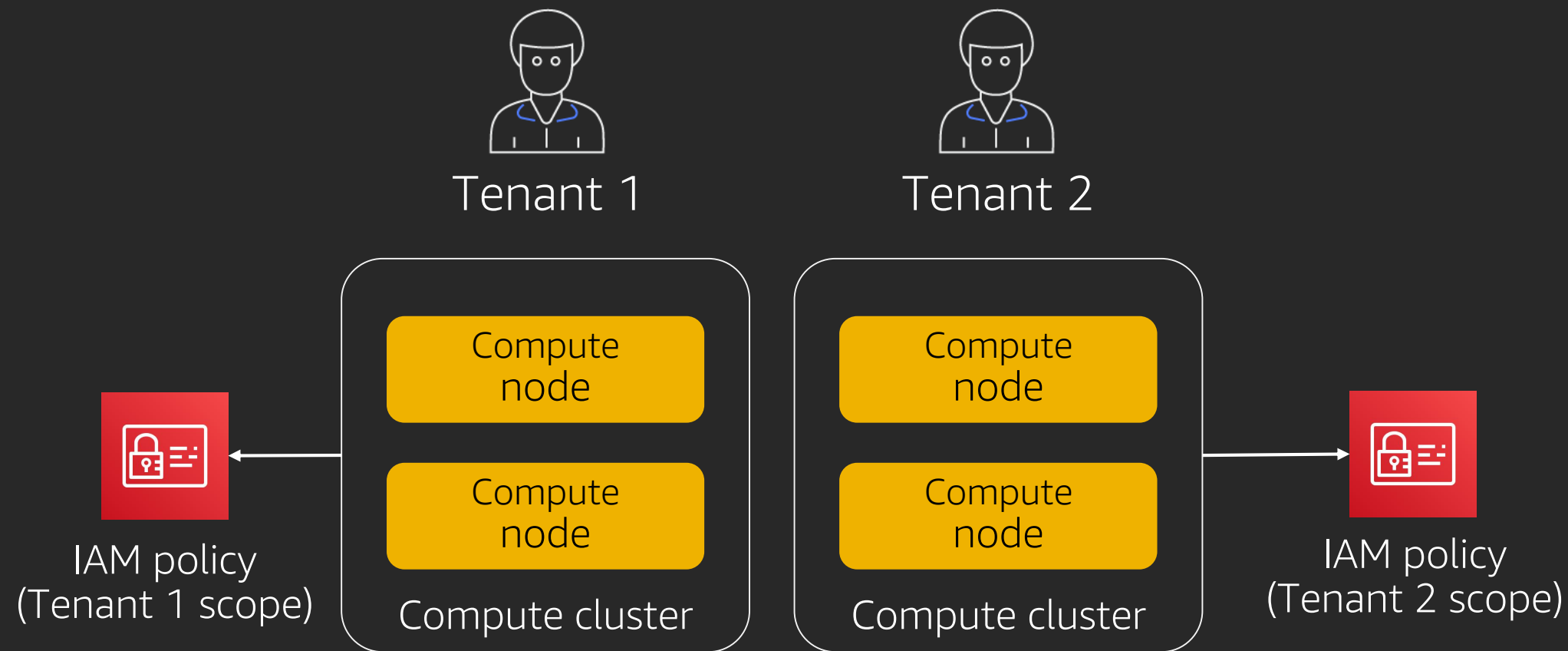


Tenant view: Resources are never shared—even when they are

# API-enforced access



# Silo compute isolation patterns

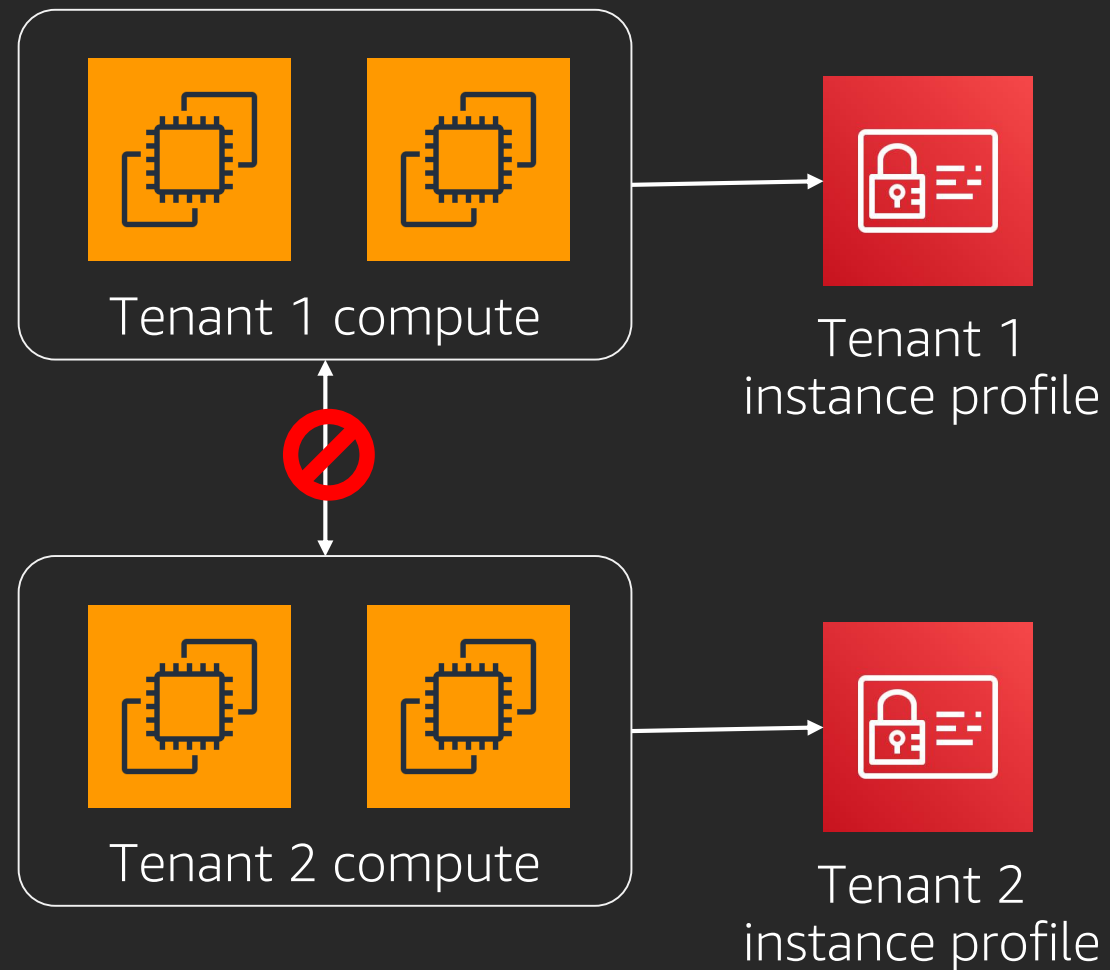


- Separate compute “cluster” per tenant
- Prevent access across cluster boundaries
- Leverage IAM and other constructs to prevent cross-tenant access
- Control scope for all downstream interactions

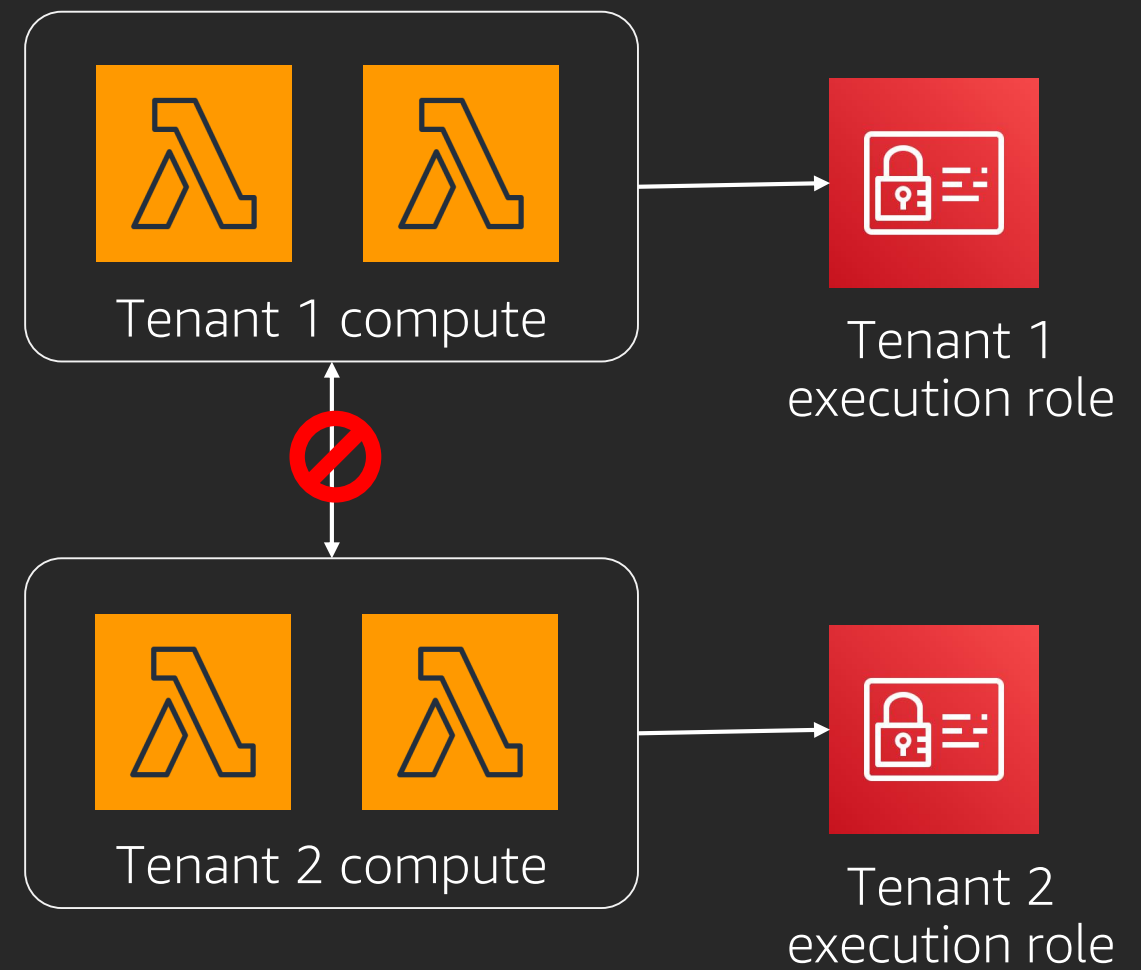


# Using roles for silo compute isolation

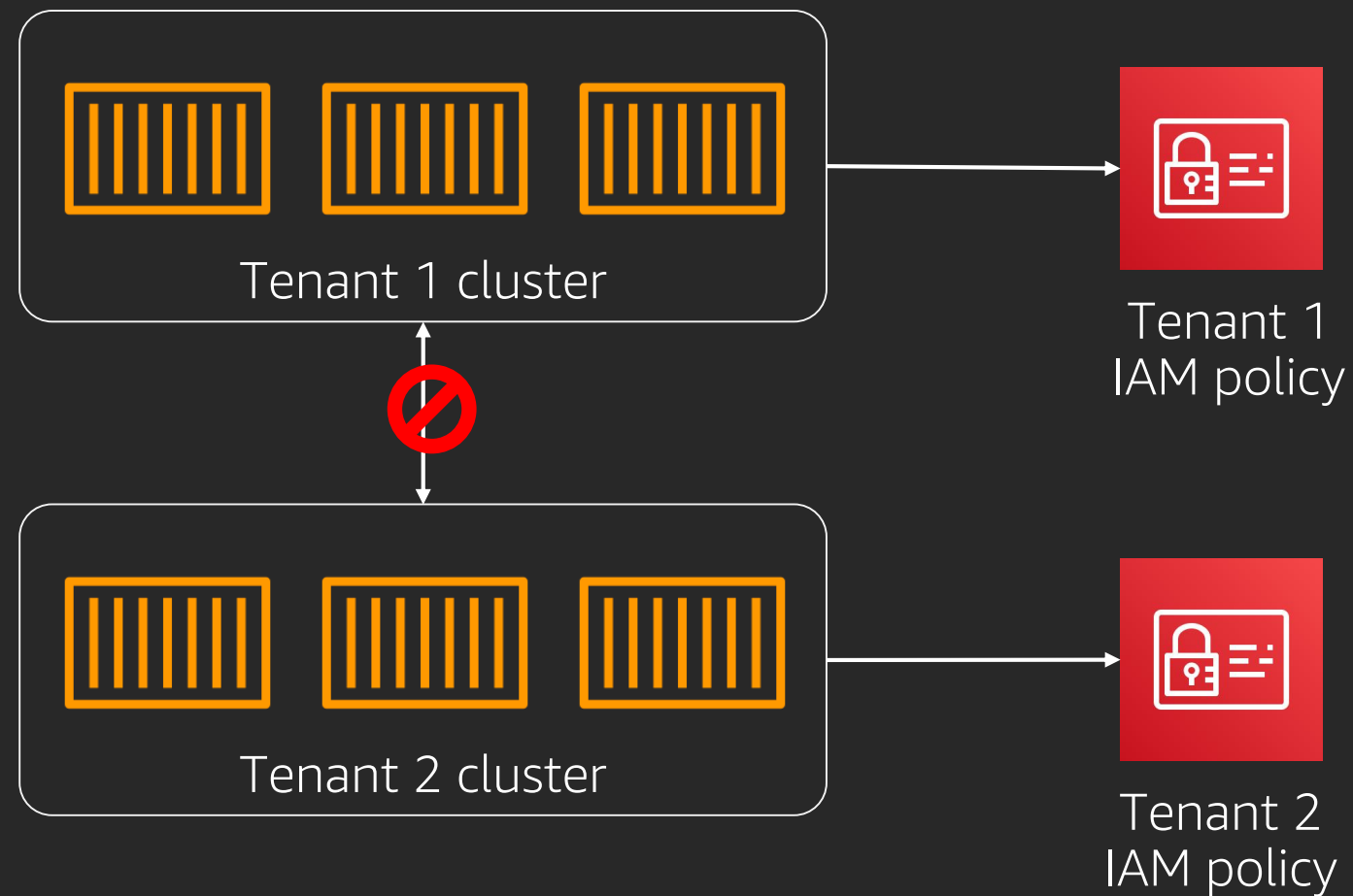
Silo with Amazon EC2



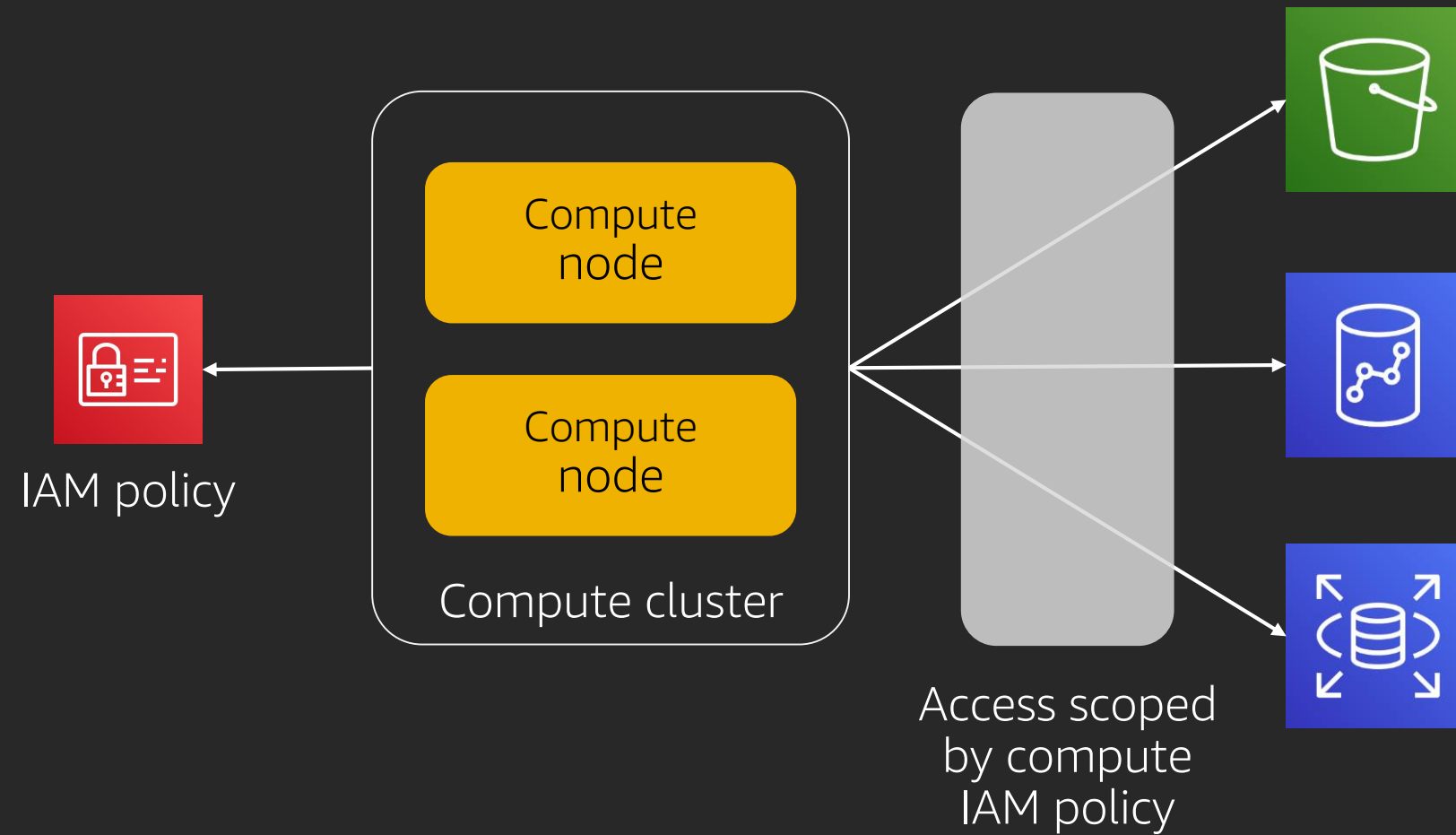
Silo with AWS Lambda



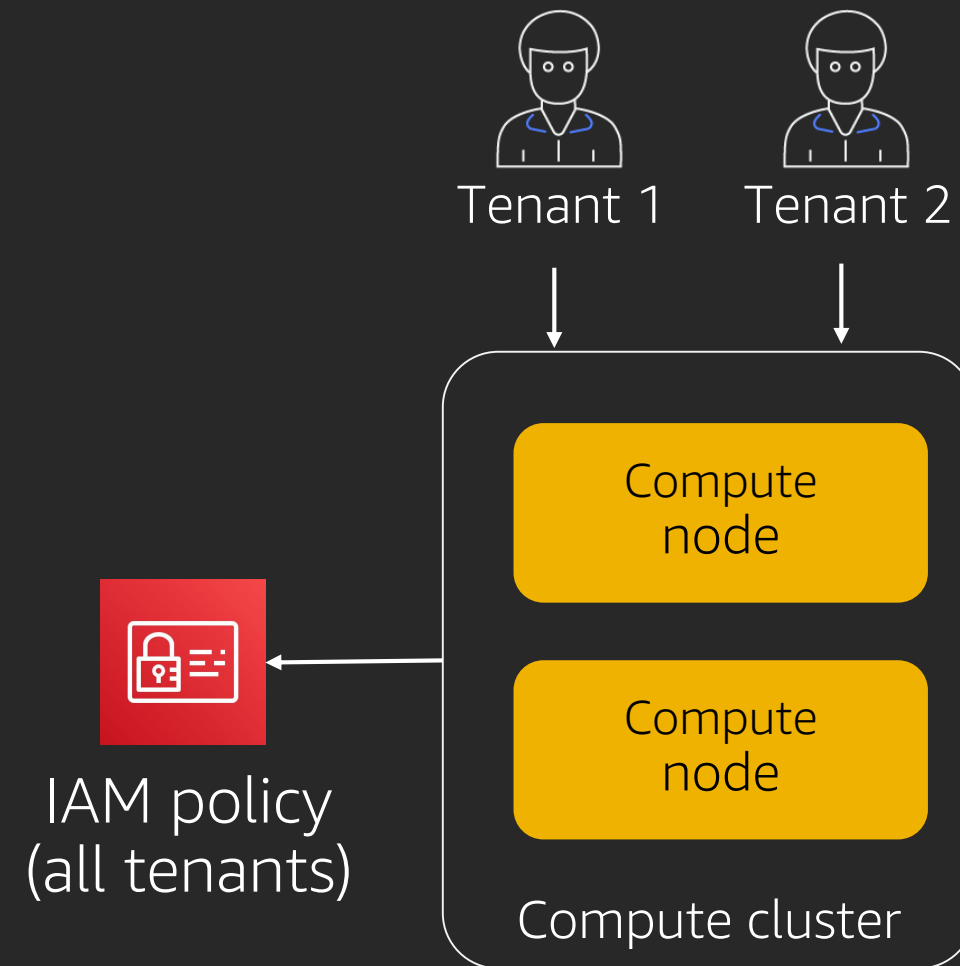
# Using roles for silo compute isolation (cont'd)



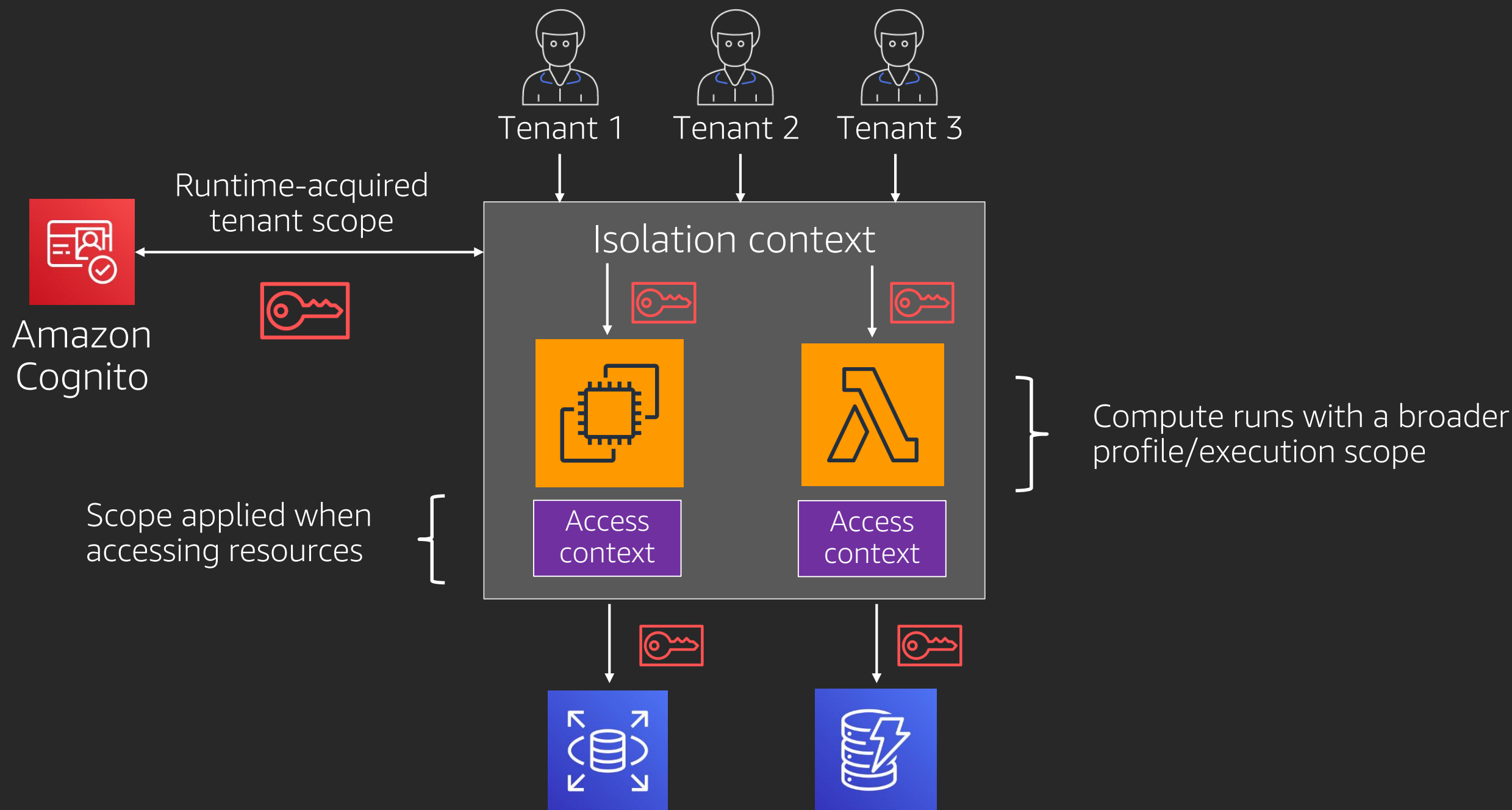
# Cascading tenant scope from compute nodes



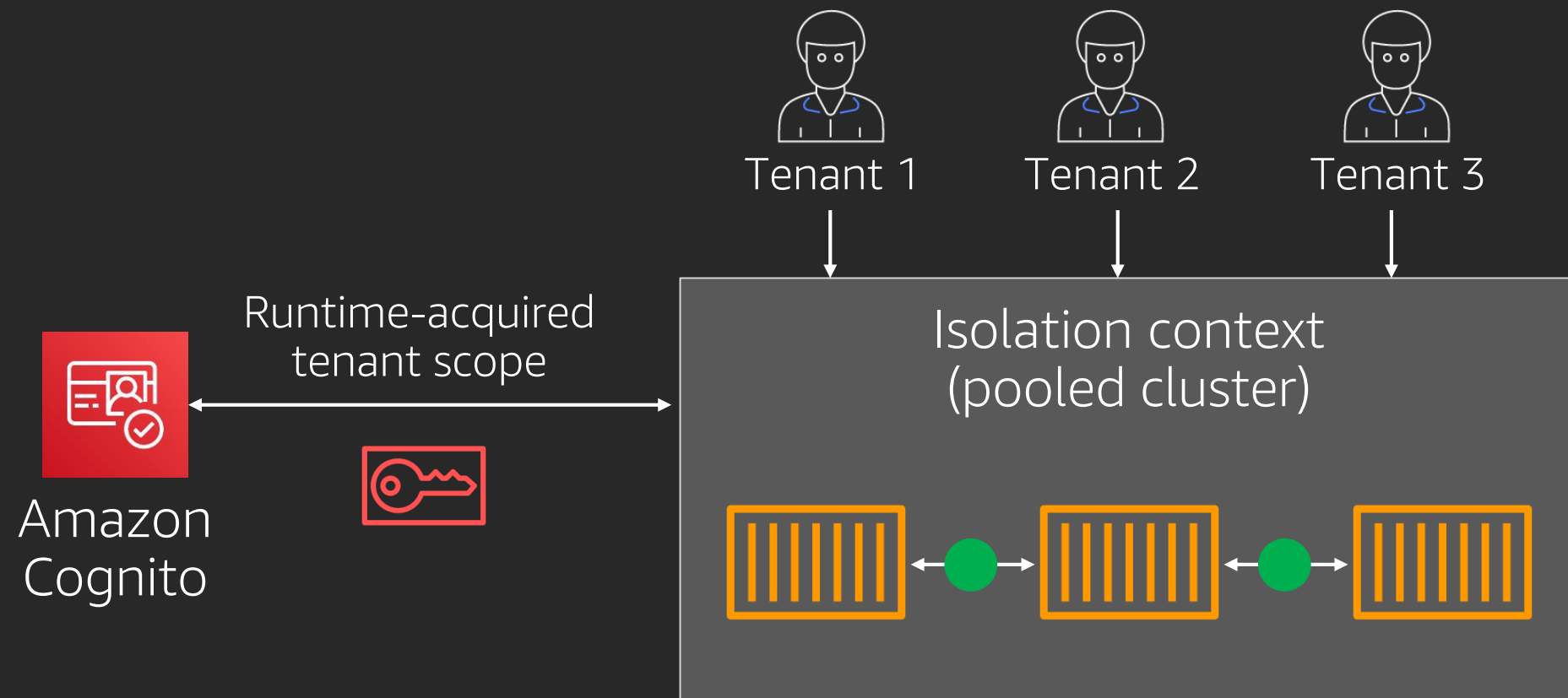
# Pool requires a broader scope



# Pool compute patterns for Amazon EC2 and AWS Lambda

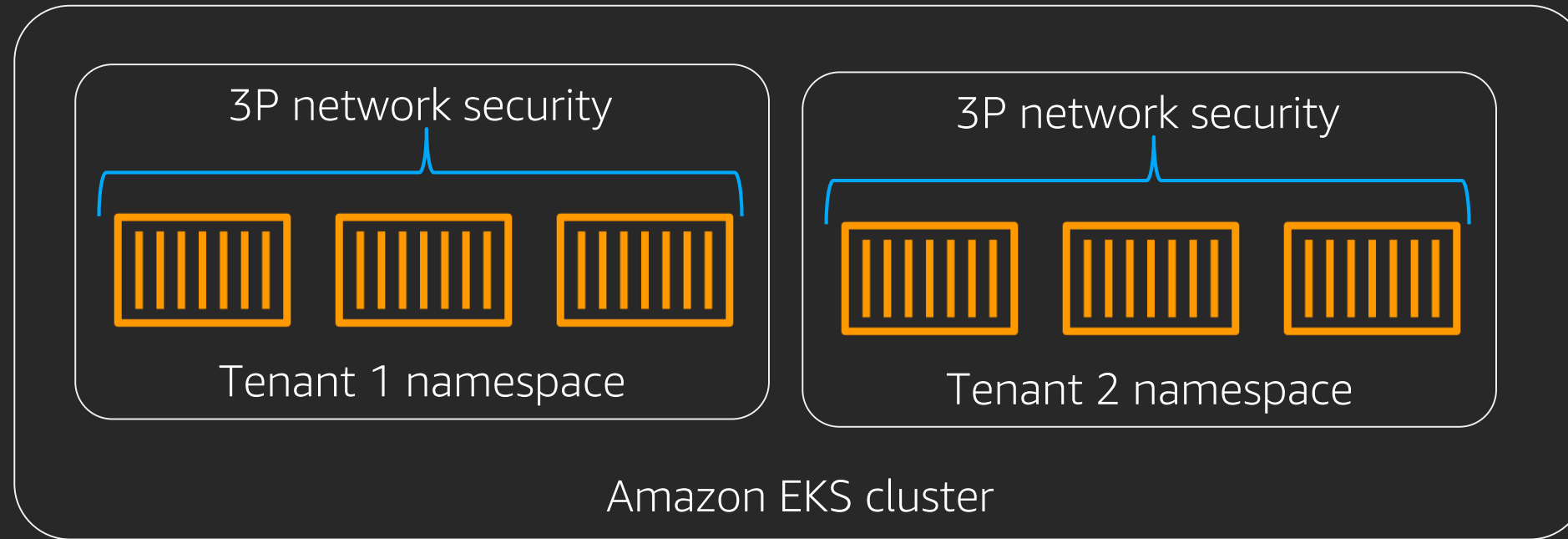


# Pool isolation strategies with containers



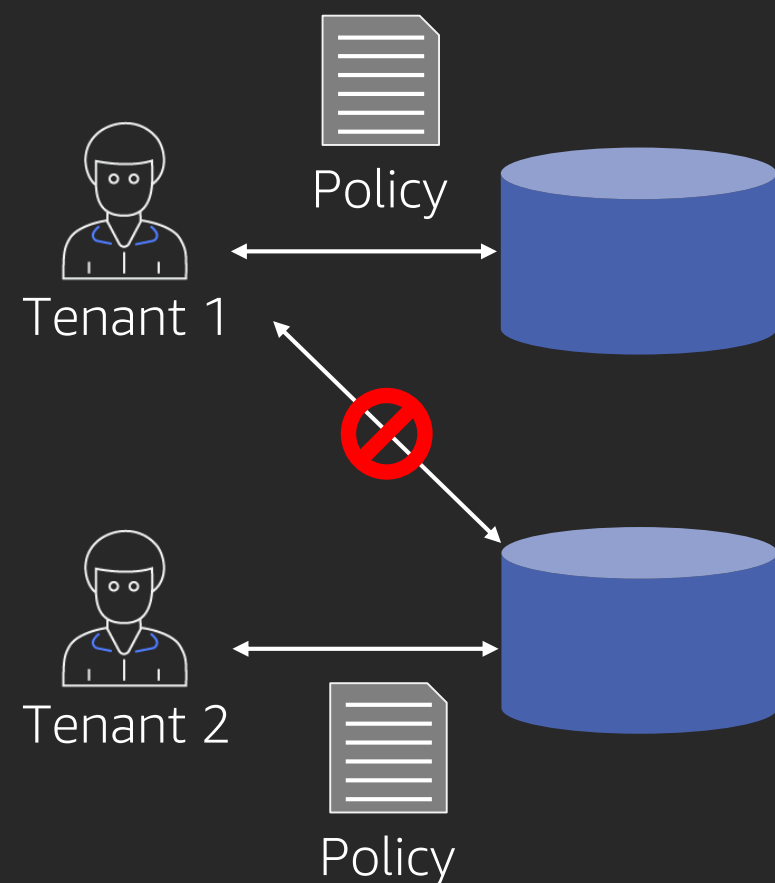
Containers allow a level of cross-tenant access that violates the spirit of multi-tenant isolation within a shared cluster

# Amazon EKS, namespaces, and third-party tooling

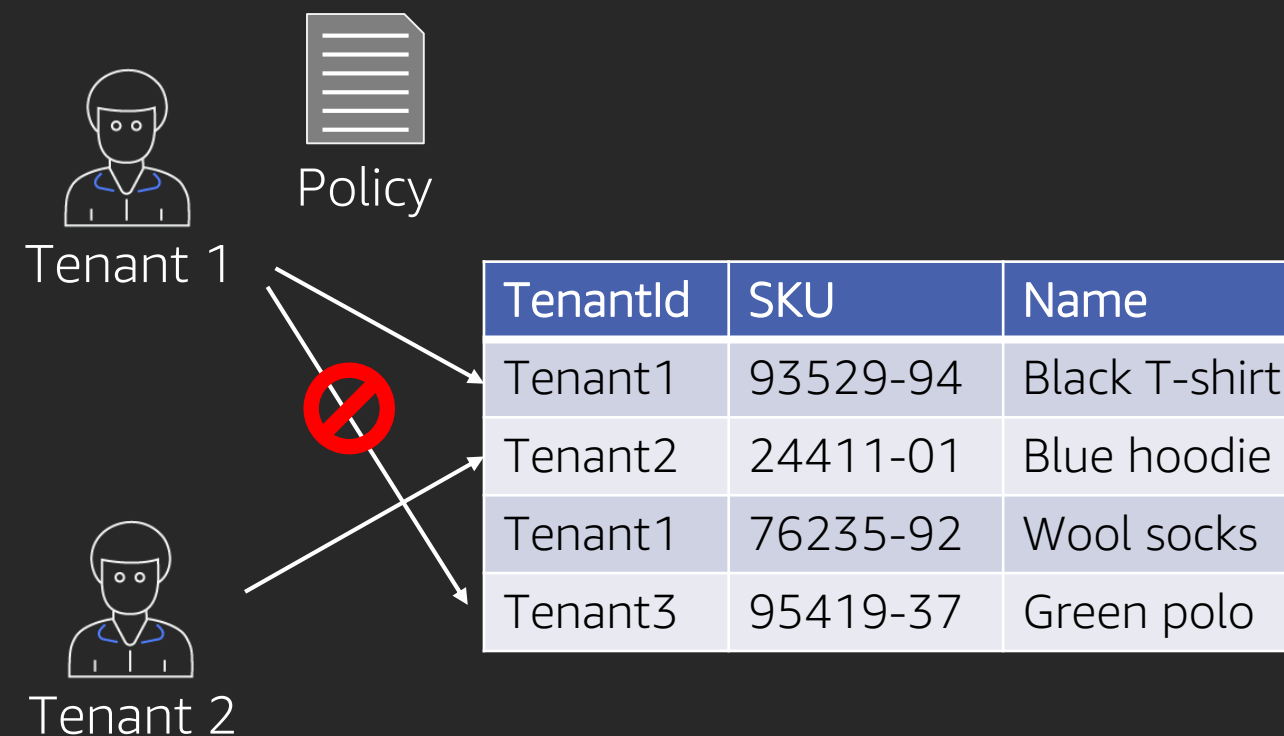


- Tenant maps to a single namespace
- Policies are set at the namespace level
- A single control plane for all tenants
- Added layer of isolation via ingress/egress policies

# Storage isolation patterns



Separate "database" per tenant  
(silo)

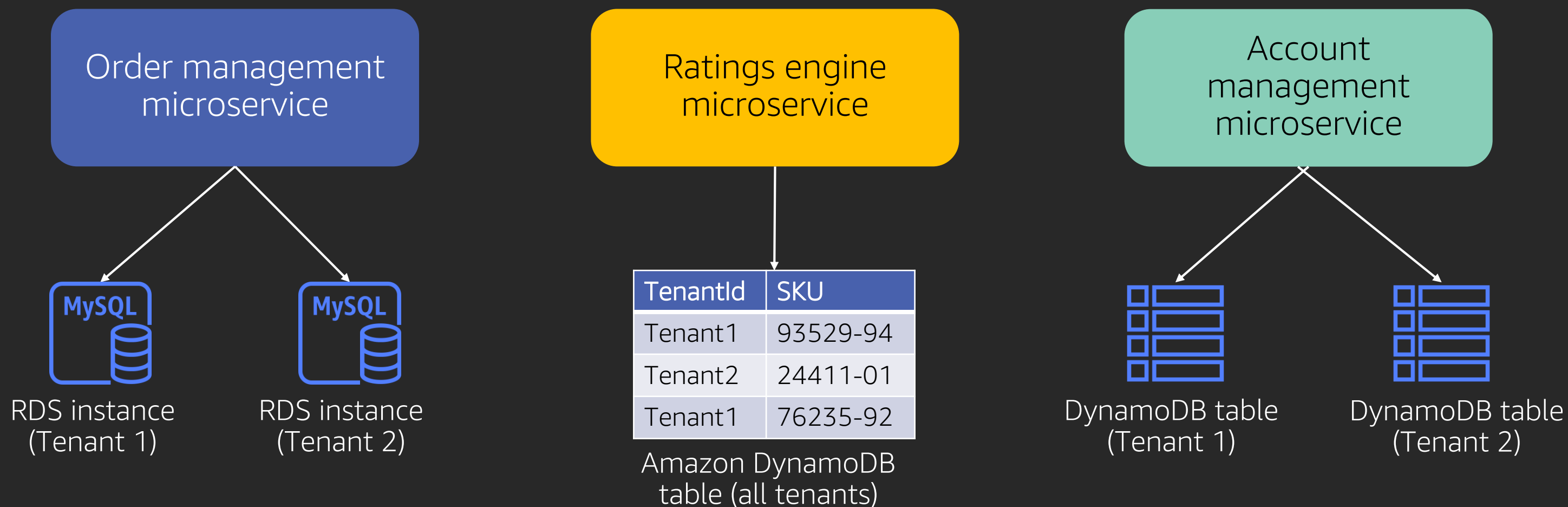


One database per tenant separated by "keys"  
(pool)



# Isolation often influences data partitioning

The isolation model of your data is shaped by the partitioning scheme you choose

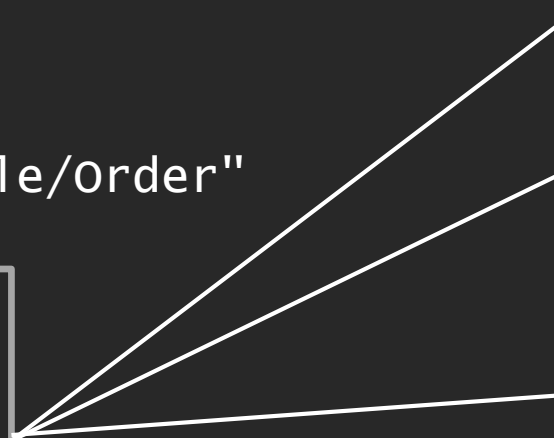


# Pool isolation with Amazon DynamoDB

```
{
  "sid": "TenantReadOnlyOrderTable",
  "Effect": "Allow",
  "Action": [
    "dynamodb:GetItem",
    "dynamodb:BatchGetItem",
    "dynamodb:Query",
    "dynamodb:DescribeTable"
  ],
  "Resource": [
    "arn:aws:dynamodb:[region]:table/Order"
  ],
  "Condition": {
    "ForAllValues:StringEquals": {
      "dynamodb:LeadingKeys": [
        "tenant1"
      ]
    }
  }
}
```

DynamoDB table

Partition Key	SKU	Name
Tenant1	93529-94	Black T-shirt
Tenant2	24411-01	Blue hoodie
Tenant1	76235-92	Wool socks
Tenant3	95419-37	Green polo
Tenant2	88314-99	White hat
Tenant1	24598-72	Tennis shoes



# Pool with RLS and Amazon Aurora PostgreSQL

## Initialize RLS

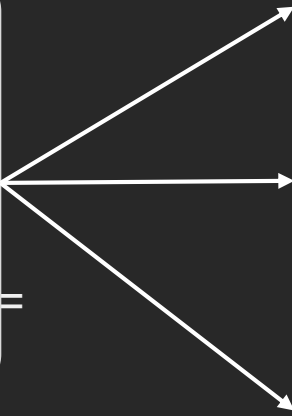
```
-- Turn on RLS
ALTER TABLE tenant ENABLE ROW LEVEL SECURITY;

-- Scope read/write by tenant
CREATE POLICY tenant_isolation_policy ON tenant
USING (tenant_id::TEXT = current_user);
```

## Query with RLS

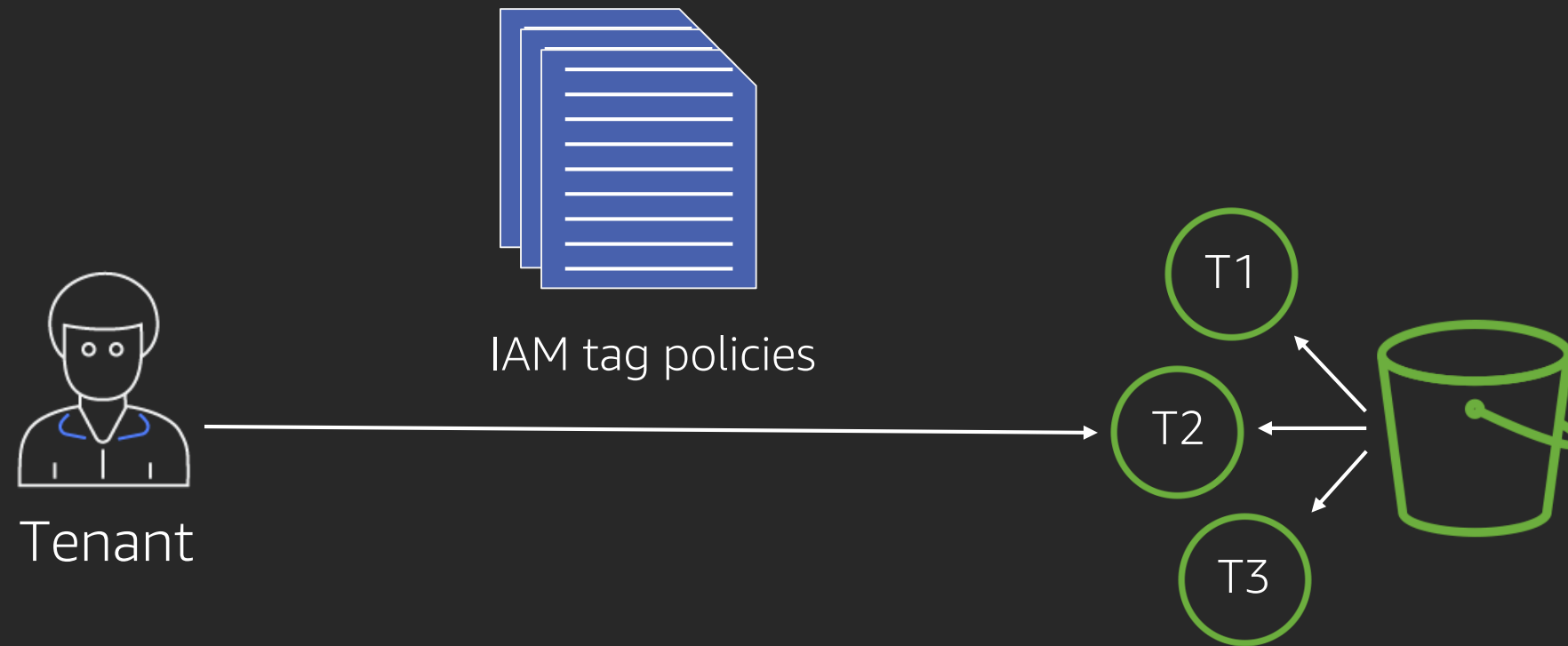
```
-- No tenant context required
rls_multi_tenant=> SELECT * FROM tenant;

-- Attempt to force tenant id
rls_multi_tenant=> SELECT * FROM tenant WHERE tenant_id =
'tenant1'
```



FK	SKU	Name
Tenant1	93529-94	Black T-shirt
Tenant2	24411-01	Blue hoodie
Tenant1	76235-92	Wool socks
Tenant3	95419-37	Green polo
Tenant2	88314-99	White hat
Tenant1	24598-72	Tennis shoes

# Isolation with Amazon S3



# The challenge: No universal isolation strategy



Amazon  
Elasticsearch  
Service



Amazon Elastic  
File System



Amazon  
Redshift



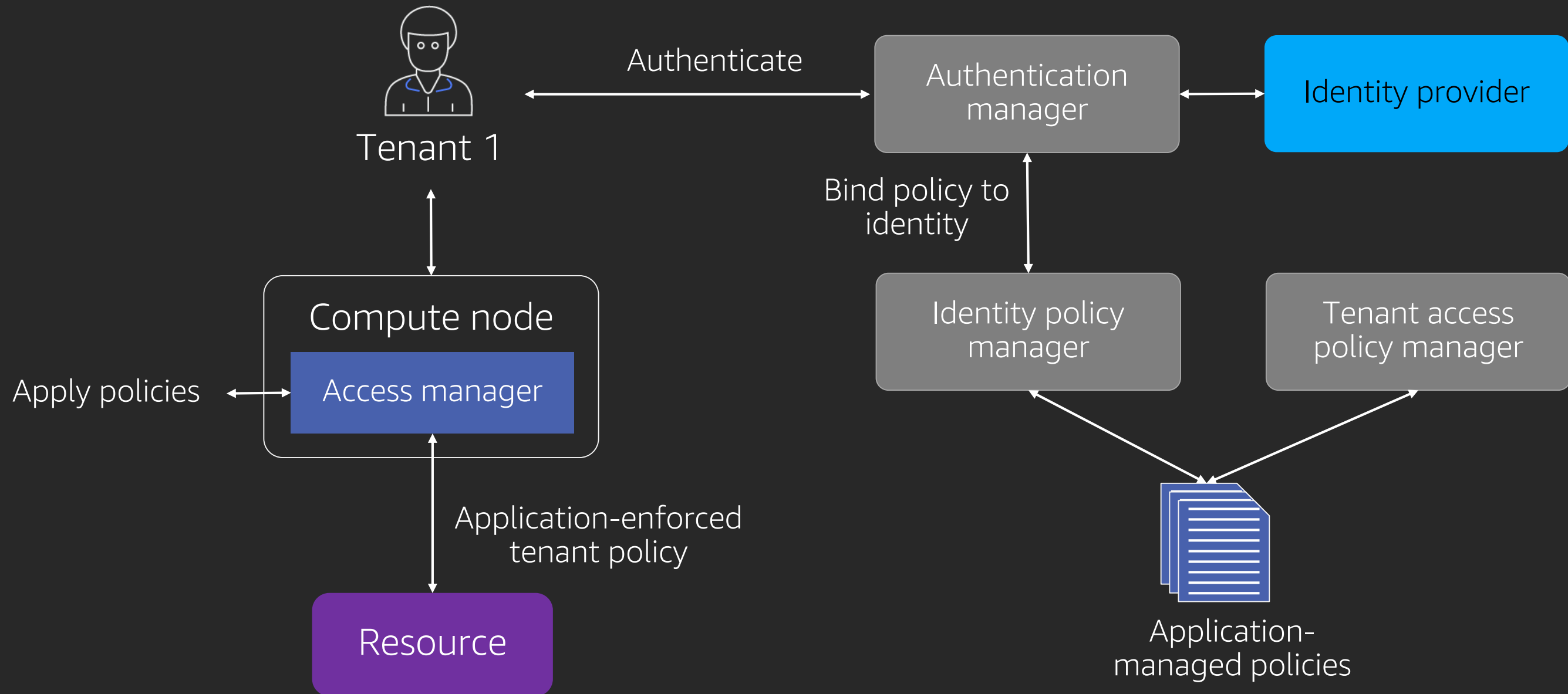
Amazon Quantum  
Ledger Database



Amazon  
DocumentDB

- Each storage technology may require a unique isolation model
- IAM may not offer direct support for silo and/or pool granularity
- IAM policy constructs will vary by service
- Self-managed storage technologies may require on-off strategies

# Application-enforced isolation (conceptual model)



# Weighing your options

## Silo isolation

### Pros

- Coarse-grained isolation
- Customer acceptance
- Better tool alignment

### Cons

- Deployment
- Cost optimization
- Manageability
- Account limits

## Policy-based isolation

### Pros

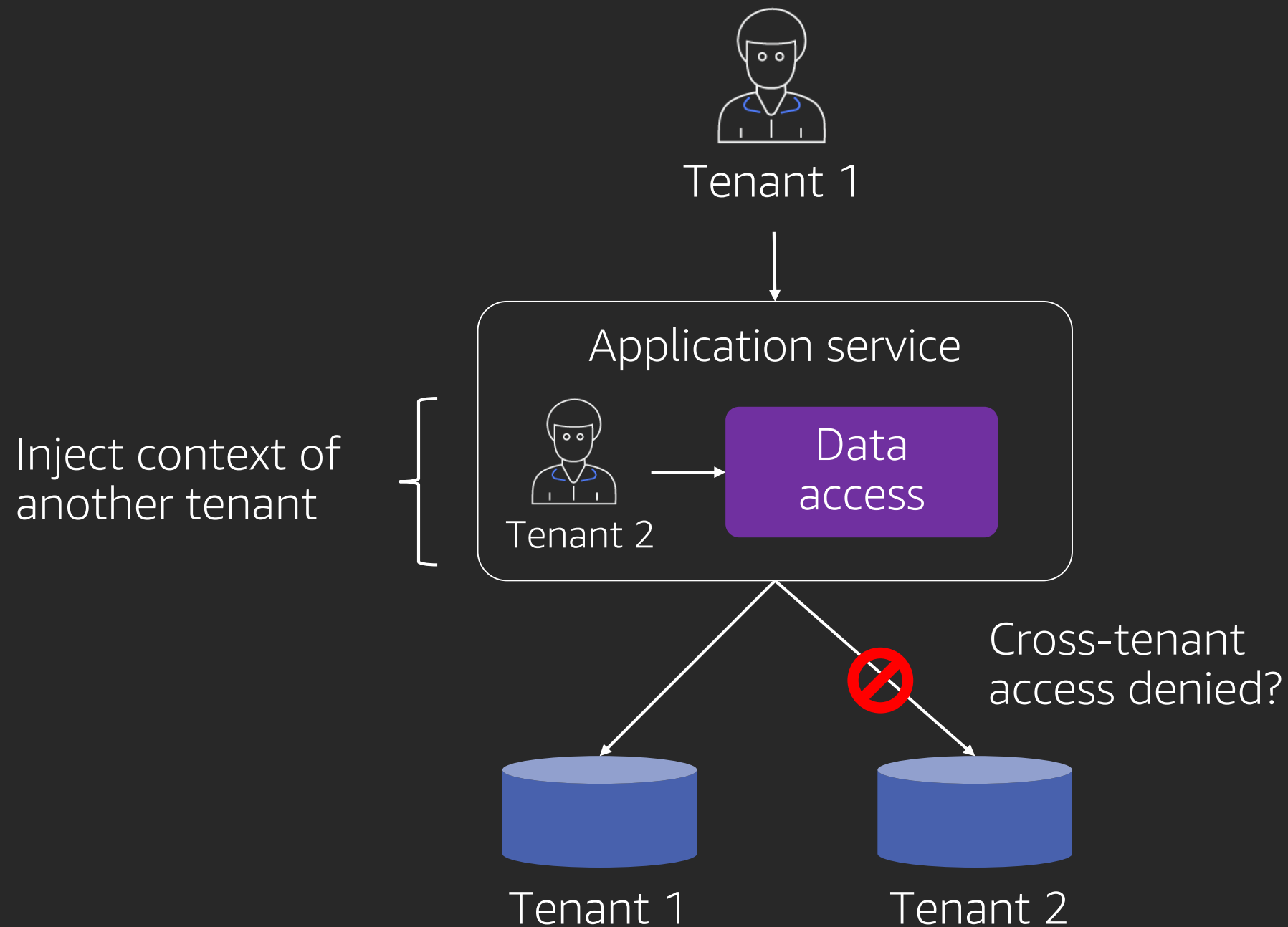
- Fine-grained isolation
- Enables resource pooling
- Flexibility

### Cons

- Customer acceptance
- Relies on convention
- Mix of technologies
- Account limits

Hybrid based on  
service/resource type

# How do we know our isolation model is working?





# Takeaways

- Design with isolation in mind
- Authentication and authorization  $\neq$  isolation
- Factor scale and account limits into your isolation strategy
- You may have to build your own isolation solutions
- Each service may require a different isolation model
- Validate that your isolation model is working
- Isolation is fundamental to success in a multi-tenant environment

# Thank you!

**Tod Golding**

todg@amazon.com