

MULTI TENANCY

COST-EFFICIENT AND SCALABLE PATTERNS
FOR SOFTWARE-AS-A-SERVICE

BJORN.BESKOW@CALLISTAENTERPRISE.SE

CADEC 2021.01.27 | [CALLISTAENTERPRISE.SE](https://callistaenterprise.se)

CALLISTA

AGENDA

- Software-as-a-Service
 - What, Why, How?
- Multi Tenancy
 - What, Why, How?
 - Challenges & Design Concerns
- Examples & Demo
- Conclusions



"Agenda by Abel, & MQ MSK WCA LosAngeles Graffiti Art" by anarchosyn

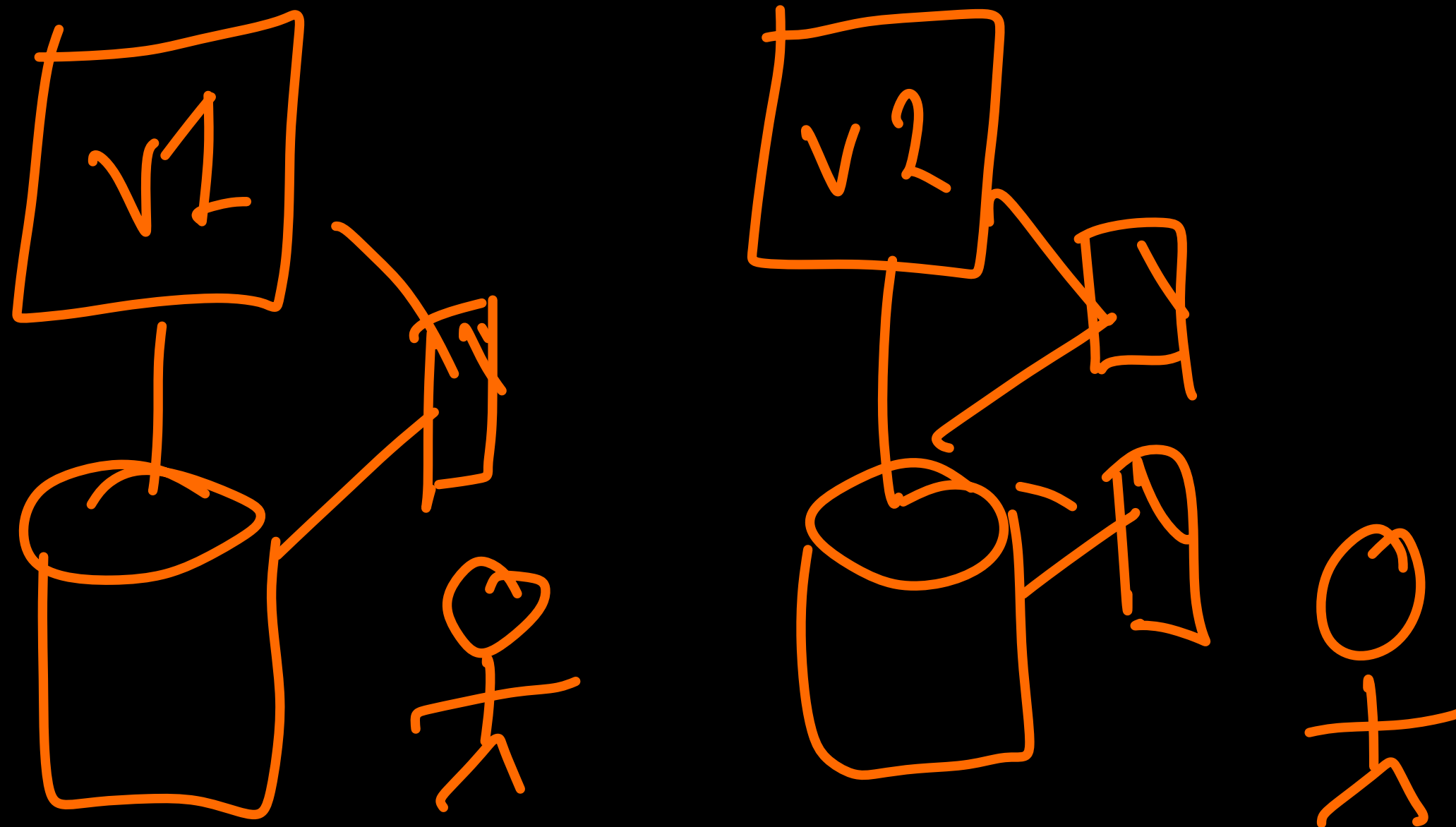
SOFTWARE-AS-A-SERVICE

- Software as a Service (SaaS) is a software delivery and licensing model in which software is licensed on a subscription basis and is centrally hosted, often in a cloud environment.



DRIVER: SIMPLICITY

- Customers need no infrastructure
- Greatly simplified Application Lifecycle for the provider
 - Versioning
 - Configuration Management
 - Testing
 - » allows better business agility



VS.



DRIVER: ECONOMY OF SCALE

- Lowered “entrance fee”, including “Freemium” models
- Cost efficiency in both hardware and software
- The wet dream of software companies: Annual Recurring Revenue via Subscriptions.

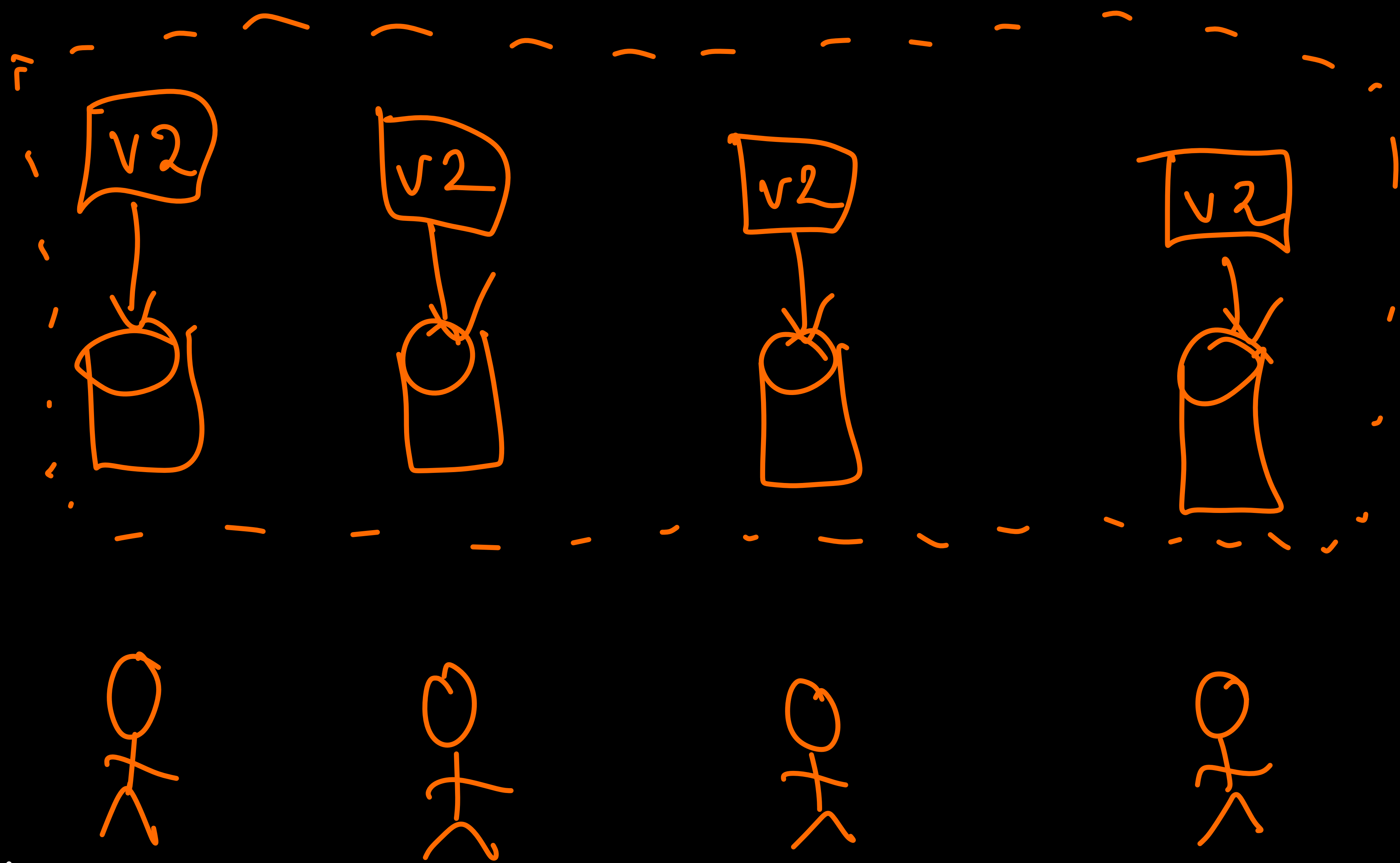


DRIVER: DATA AGGREGATION & BUSINESS INTELLIGENCE

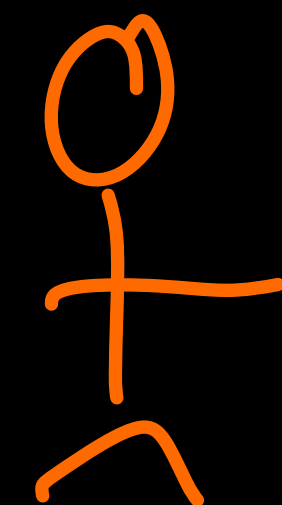
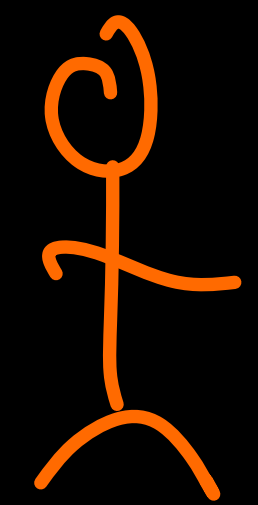
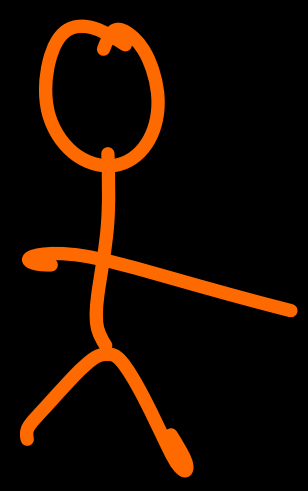
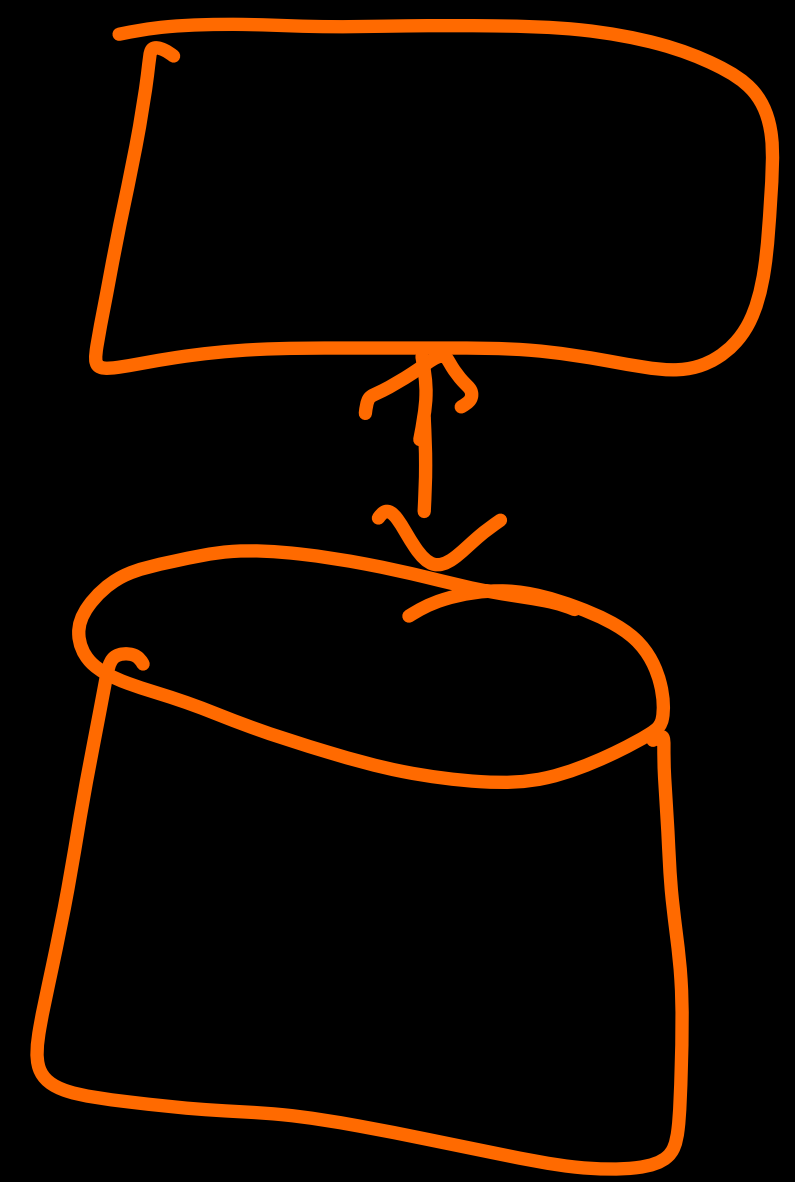
- With access to multiple customer's uniform data (if allowed), the opportunities for data mining and business intelligence are endless



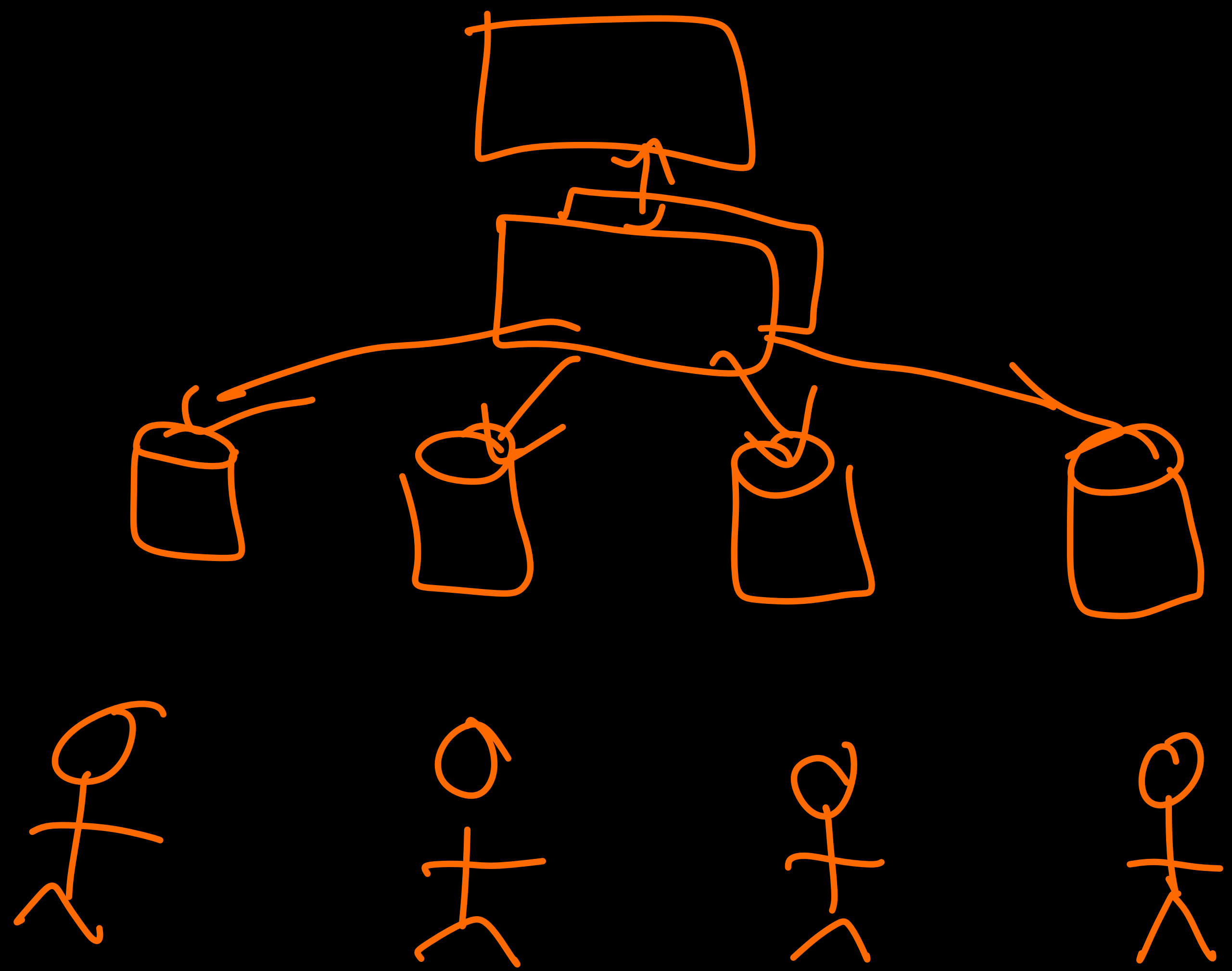
SAAS TOPOLOGIES: DEDICATED OR "SINGLE TENANT"



SAAS TOPOLOGIES: SHARED OR "MULTI TENANT"



SAAS TOPOLOGIES: ... OR MORE OFTEN A MIX



BALANCING CONFLICTING REQUIREMENTS

- **Shared resources vs**
- **Logical or Physical separation**



DRIVING NON-FUNCTIONAL ASPECTS

- Data separation requirements?
- Number of tenants?
- Tenant onboarding requirements?
 - Frequency
 - Lead time requirements
- Customizability requirements?



Figure FONT: <http://www.withinreach.com.au/SiteAssets/Lists/Posts/AllPosts/wordcloudweb.png>

| MULTI TENANCY DESIGN CONCERNS

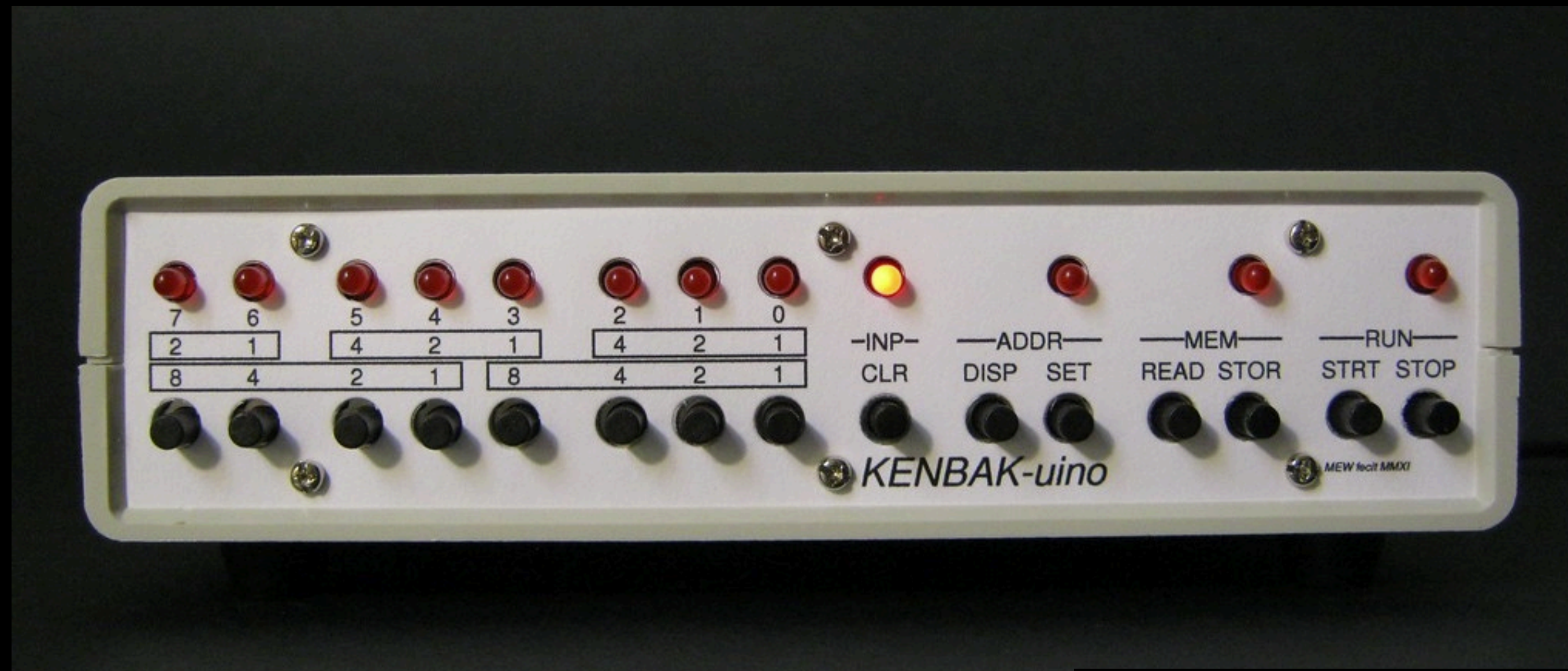
- Customizability



"Custom Fondant M&M Character Cupcakes!!!!" by DixieBelleCupcakeCafe

■ MULTI-TENANCY CUSTOMIZATION PATTERN: CONFIGURATION

- Tenant-specific Configuration Properties, available through Tenant Context
- Feature Toggling operated by Tenant Context
- Custom Branding and Co-branding via Templates (colour schemes, logos, etc.)

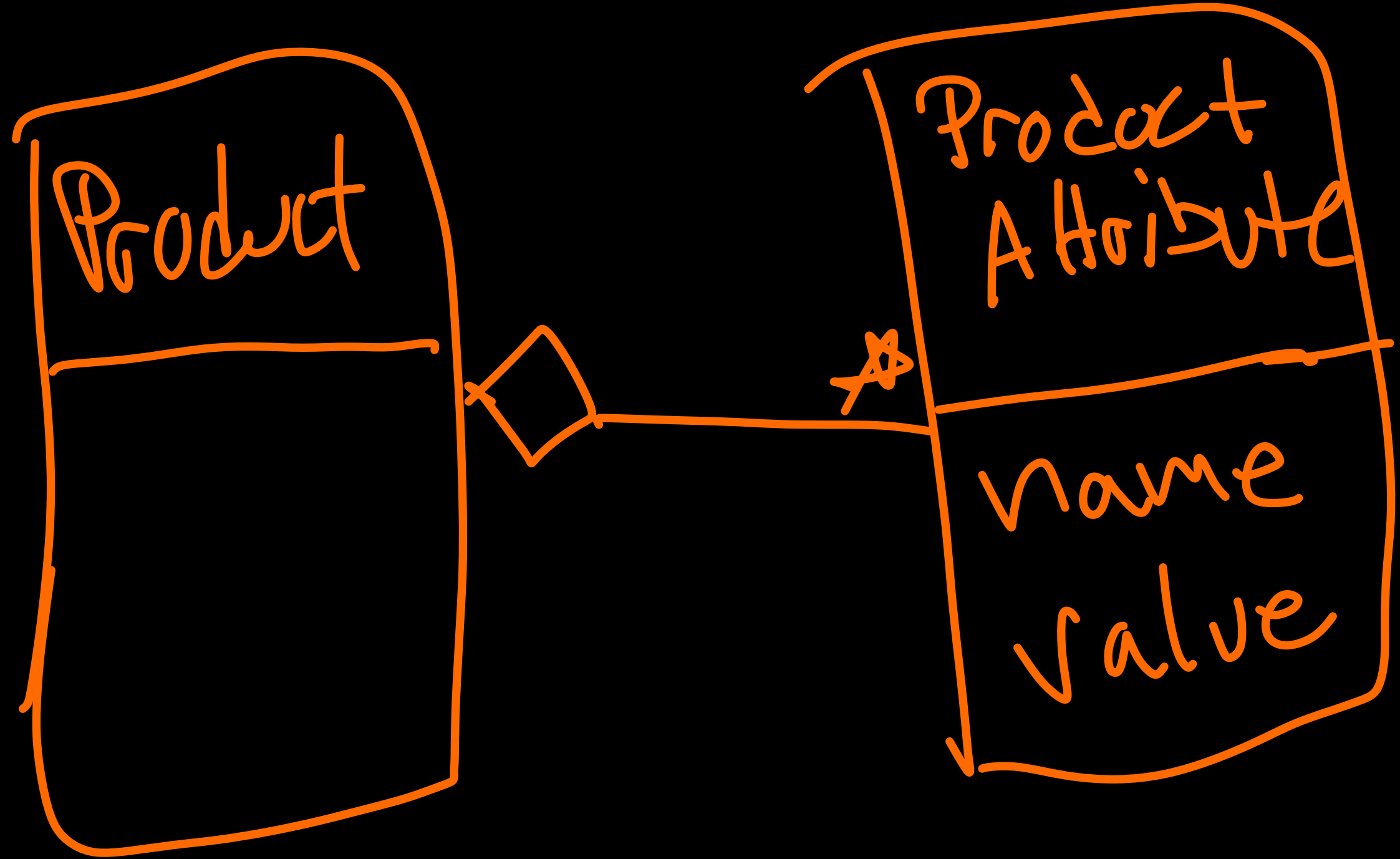


"Ready for Input" by funny-polynomial

MULTI-TENANCY CUSTOMIZATION PATTERN: META MODELING



vs.



| MULTI TENANCY DESIGN CONCERNS

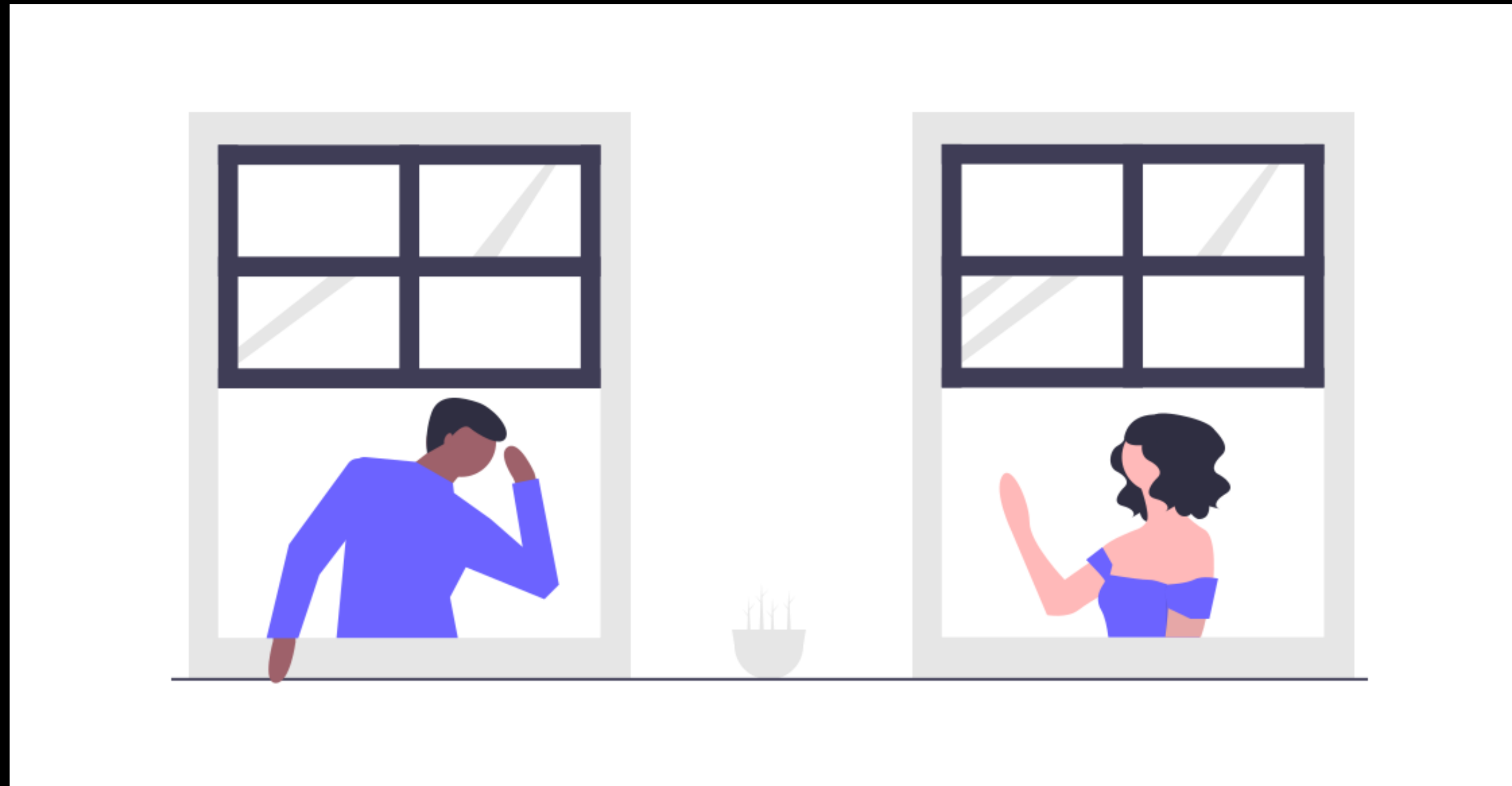
- Customizability
- Performance Isolation



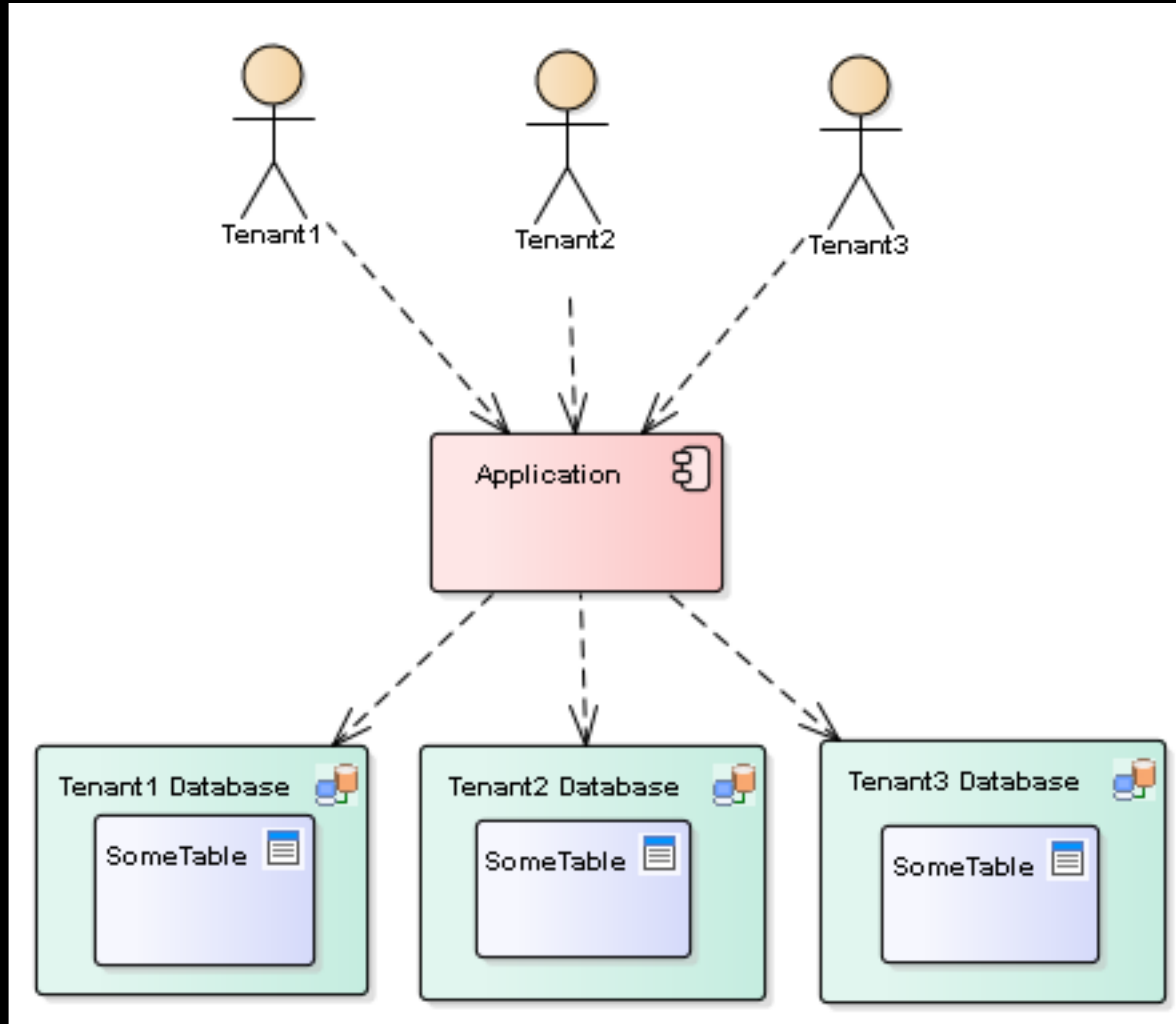
"Easter eggs wearing face mask and in self-isolation" by Ivan Radic

| MULTI TENANCY DESIGN CONCERNS

- Customizability
- Performance Isolation
- Persistence & Data Isolation



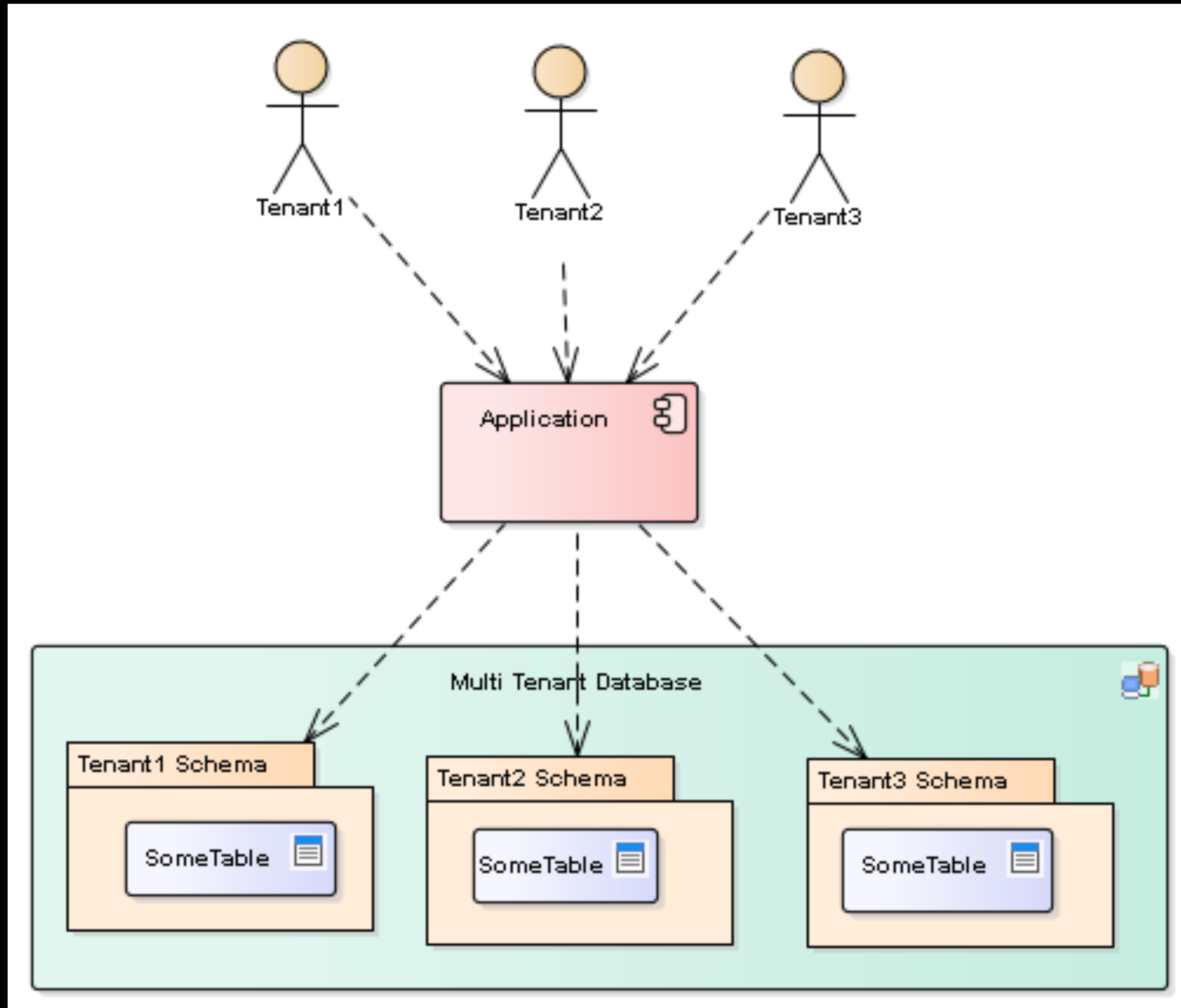
MULTI-TENANCY PERSISTENCE PATTERNS: DATABASE PER TENANT



■ MULTI-TENANCY PERSISTENCE PATTERNS: DATABASE PER TENANT

- Pros:
 - Conceptually simple
 - Strong data isolation guaranteed by database
 - Simplifies per tenant management & maintenance
 - Simplifies per tenant performance isolation
- Cons:
 - Overhead in resource requirements
 - Complicates db migrations
 - Scales to hundreds of tenants?

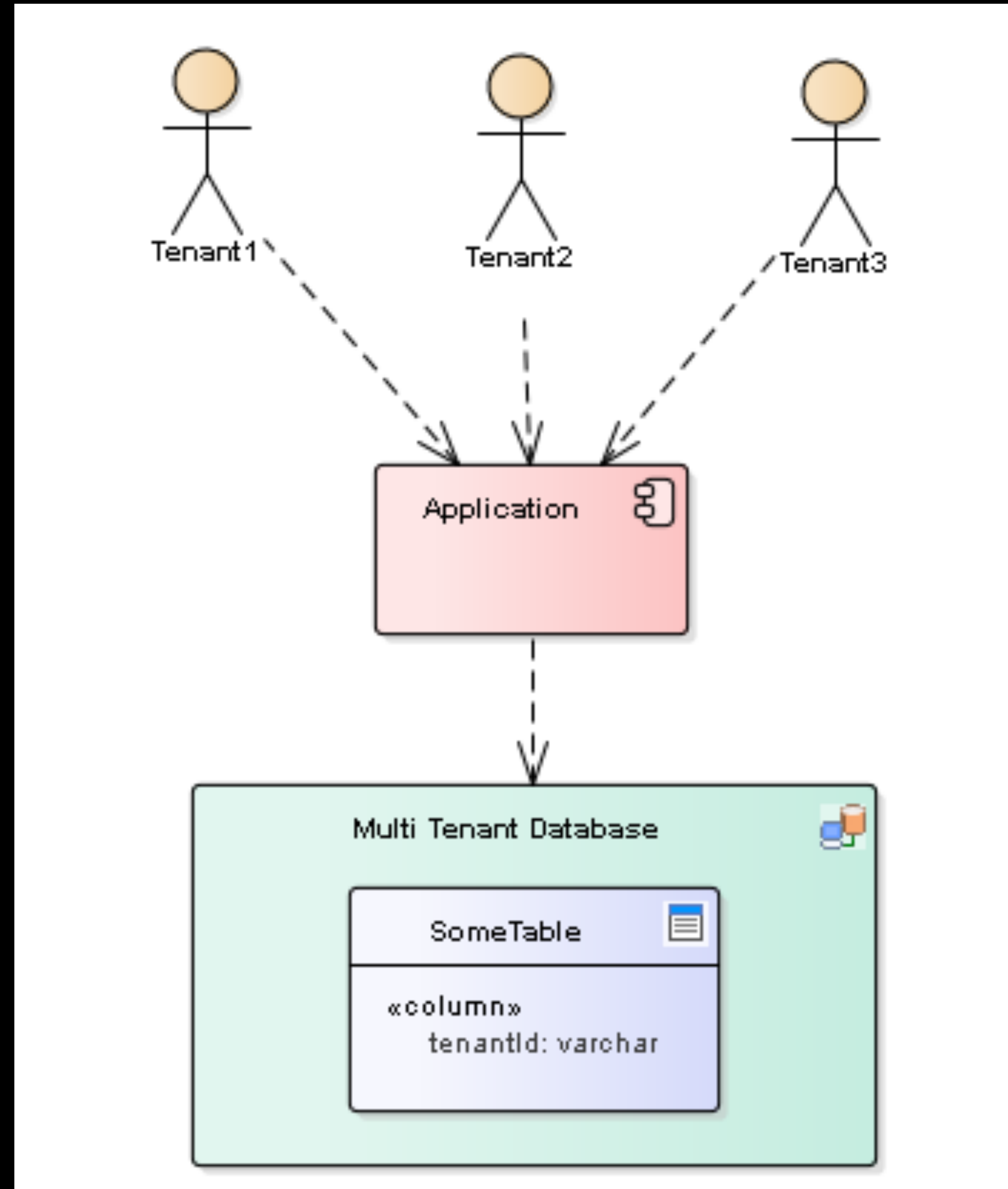
MULTI-TENANCY PERSISTENCE PATTERNS: SCHEMA PER TENANT



■ MULTI-TENANCY PERSISTENCE PATTERNS: SCHEMA PER TENANT

- Pros:
 - Conceptually simple
 - Relatively strong data isolation guaranteed by database
 - Simplifies per tenant management & maintenance
- Cons:
 - Slight overhead in resource requirements
 - Complicates db migrations
 - Scales to thousands of tenants?

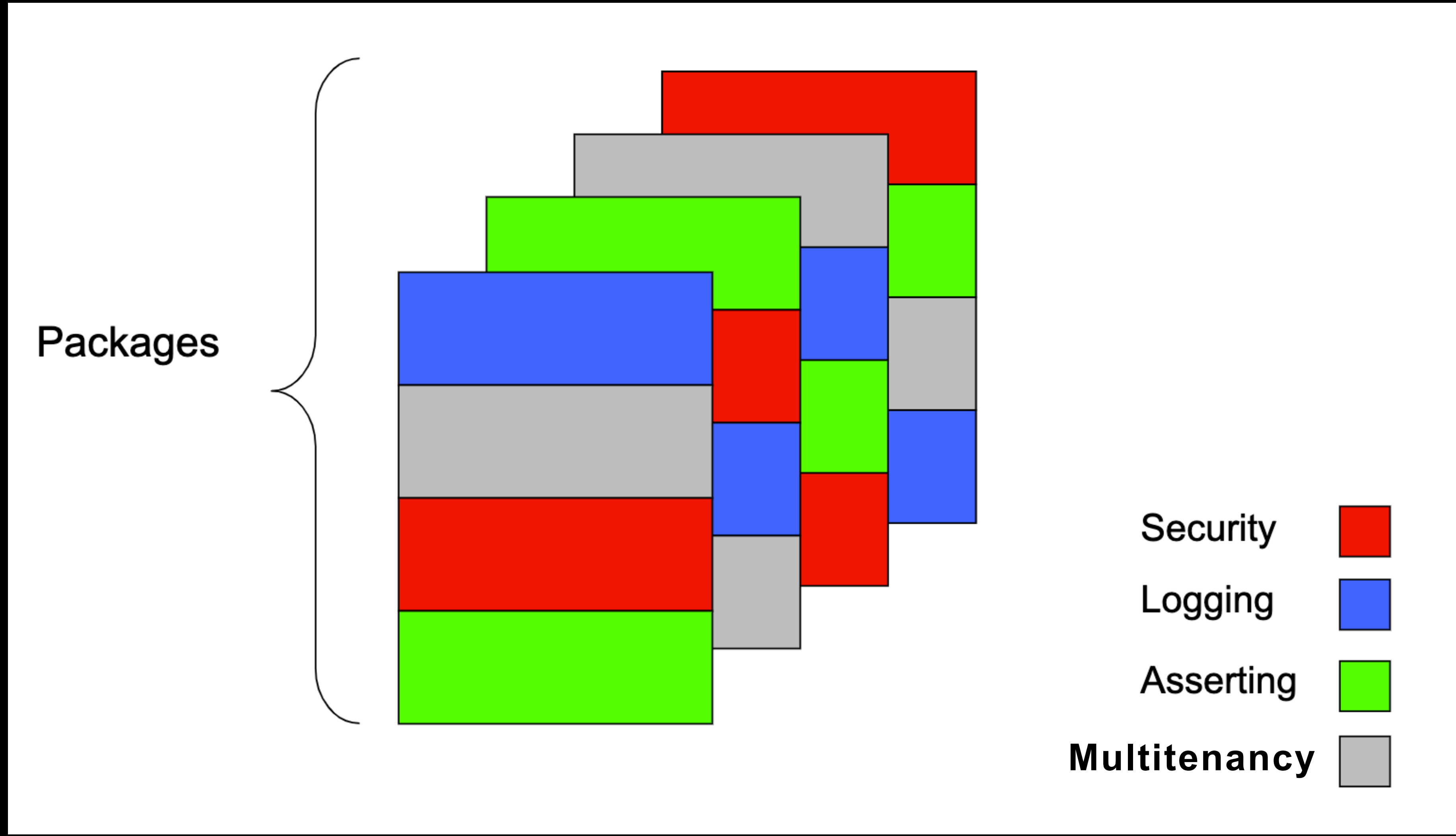
MULTI-TENANCY PERSISTENCE PATTERNS: SHARED DATABASE



■ MULTI-TENANCY PERSISTENCE PATTERNS: SHARED DATABASE

- Pros:
 - Minimal overhead
 - Simpler data maintenance and migrations
 - Scales to any number of tenants
- Cons:
 - Data Partitioning guarantee only by application code
 - Database may become too big and need sharding to scale

MULTI TENANCY AS A CROSS-CUTTING CONCERN



| MULTI TENANCY DATA ISOLATION AS A CROSS-CUTTING CONCERN

- Tenant Resolution
- Tenant Context Propagation
- Datasource Connection management
- Schema Manipulation
- Query Generation

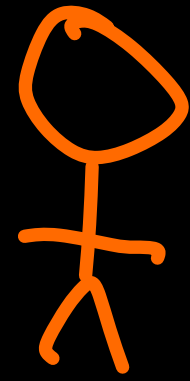


Demo

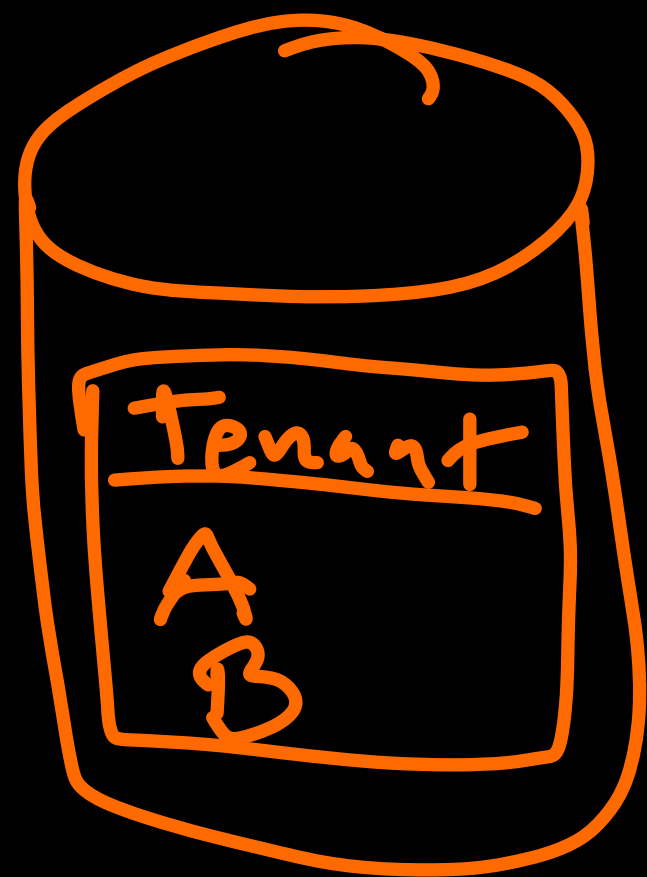
MULTI-TENANCY WITH SPRING DATA JPA

[HTTPS://GITHUB.COM/CALLISTAENTERPRISE/BLOG-MULTITENANCY](https://github.com/callistaenterprise/blog-multi-tenancy)

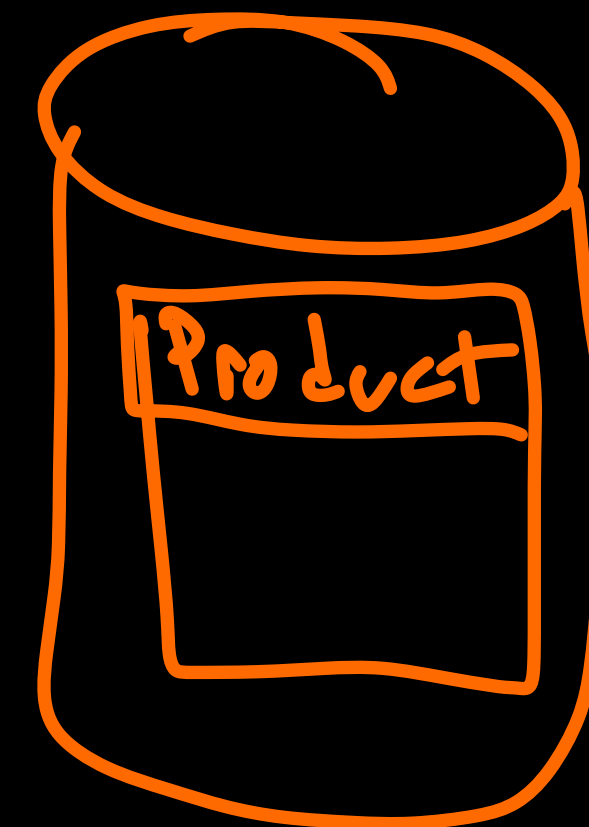
SCENARIO 1: DATABASE PER TENANT

A 

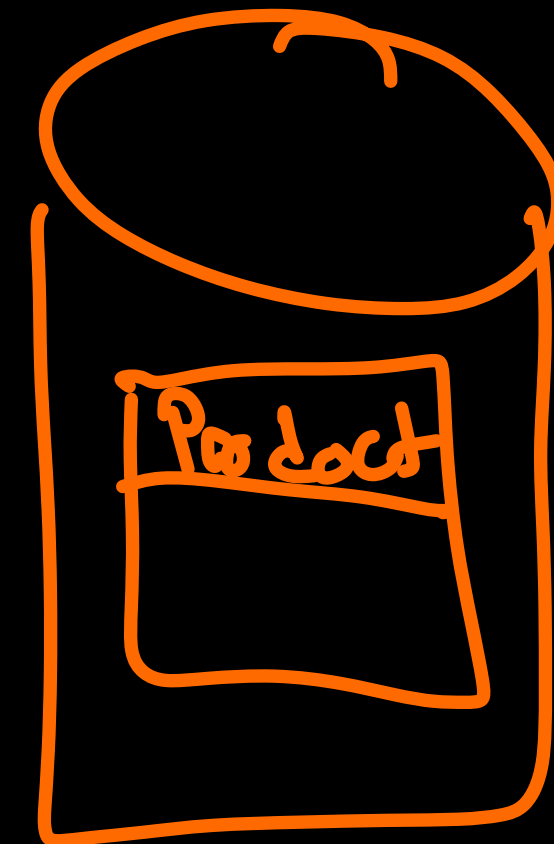
B 



Master



A



B

Project Structure

- Project
 - WebConfiguration
 - controller
 - domain.entity
 - Product
 - model
 - multi_tenancy
 - async
 - config
 - master
 - tenant
 - hibernate
 - liquibase
 - DynamicDataSourceBasedMultiT
 - TenantLiquibaseConfig
 - TenantPersistenceConfig
 - domain.entity
 - Tenant
 - exception
 - interceptor
 - repository
 - service
 - util
 - repository
 - ProductRepository
 - services

Search Everywhere Double ↵

Go to File ↵⌘O

Recent Files ⌘E

Navigation Bar ⌘↑

Drop files here to open

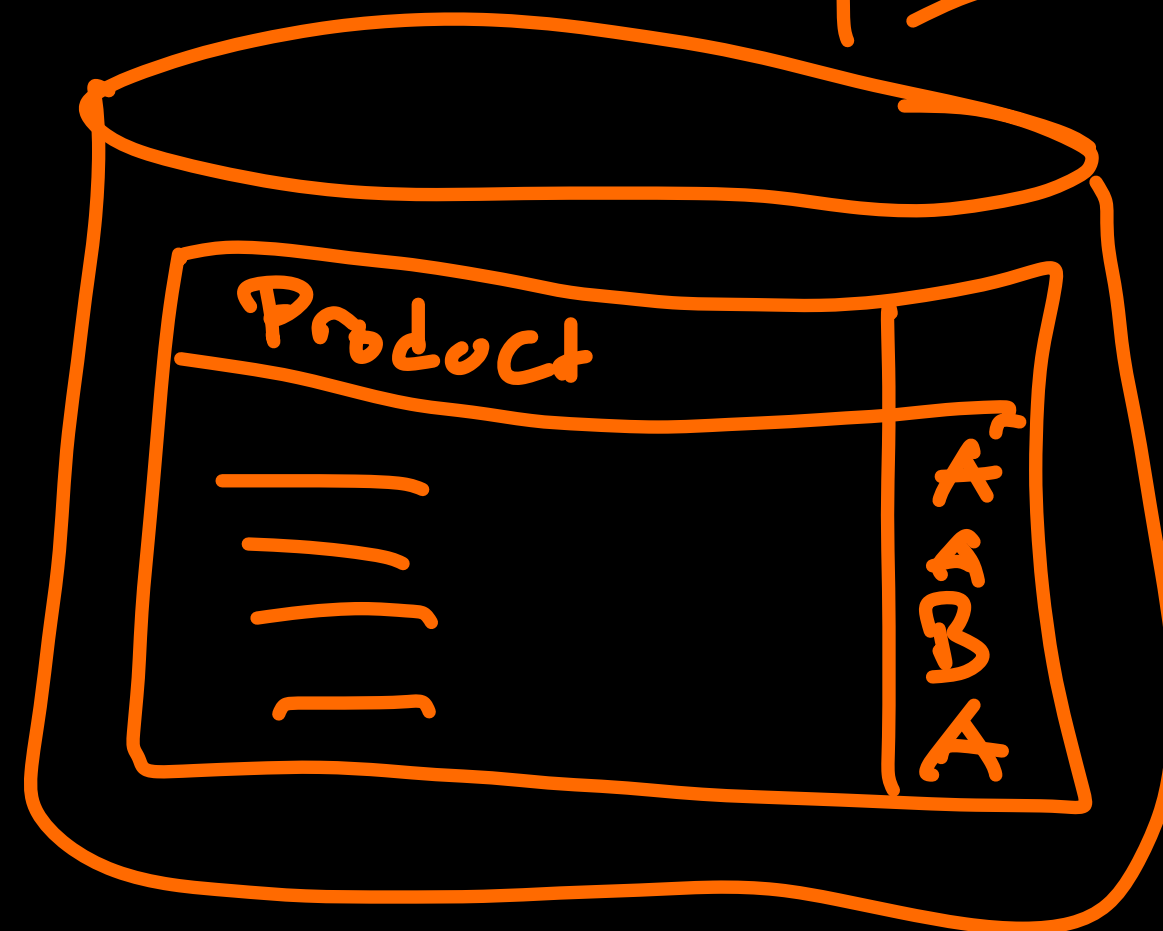
```

Run: multi-tenant-management [org.springframework.boot:spring-boot-m... x multi-tenant-service [org.springframework.boot:spring-boot-maven-... x
2021-01-24 16:00:10.877 INFO 115 --- [ restartedMain] o.hibernate.annotations.common.Version      : HCANN000001: Hibernate Commons Annotations {5.1.2.F
2021-01-24 16:00:10.931 INFO 115 --- [ restartedMain] org.hibernate.dialect.Dialect                : HHH000400: Using dialect: org.hibernate.dialect.Pos
2021-01-24 16:00:11.174 INFO 115 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator          : HHH000490: Using JtaPlatform implementation: [org.h
2021-01-24 16:00:11.180 INFO 115 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean    : Initialized JPA EntityManagerFactory for persistenc
2021-01-24 16:00:11.408 INFO 115 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor    : Initializing ExecutorService 'applicationTaskExecut
2021-01-24 16:00:11.500 INFO 115 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer         : LiveReload server is running on port 35729
2021-01-24 16:00:11.516 INFO 115 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer    : Tomcat started on port(s): 8889 (http) with context
2021-01-24 16:00:11.523 INFO 115 --- [ restartedMain] s.c.b.t.TenantManagementApplication        : Started TenantManagementApplication in 3.708 second
  
```

SCENARIO 2: SHARED DATABASE

A

B



"SELECT
... WHERE
TENANTID =
:TENANT"

Project Structure

- DynamicDataSourceBasedMi
- TenantLiquibaseConfig
- TenantPersistenceConfig
- domain.entity
 - Tenant
- exception
- interceptor
 - TenantInterceptor
- repository
 - TenantRepository
- service
- util
 - TenantContext
- repository
 - ProductRepository
- services
- util
- MultiTenantServiceApplication
- resources
 - db.changelog
 - db.changelog-tenant-1.0.yaml

```
31  |   - column:
32  |     name: price
33  |     type: INTEGER
34  |     constraints:
35  |       nullable: false
36  |
37  |
38  |   - changeSet:
39  |     id: product stock
40  |     author: bb@callistaenterprise.se
41  |     changes:
42  |     - addColumn:
43  |       tableName: product
44  |       columns:
45  |       - column:
46  |         name: stock
47  |         type: INTEGER
48  |         constraints:
49  |           nullable: false
50  |           defaultValueNumeric: 0
51  |
```

```
jMain] l.lockservice.StandardLockService : Successfully acquired change log lock
jMain] l.c.StandardChangeLogHistoryService : Reading from public.databasechangelog
jMain] l.lockservice.StandardLockService : Successfully released change log lock
jMain] ataSourceBasedMultiTenantSpringLiquibase : Liquibase ran for tenant gotham
jMain] ataSourceBasedMultiTenantSpringLiquibase : Initializing Liquibase for tenant hogwarts
jMain] l.lockservice.StandardLockService : Successfully acquired change log lock
jMain] l.c.StandardChangeLogHistoryService : Reading from public.databasechangelog
jMain] l.lockservice.StandardLockService : Successfully released change log lock
jMain] ataSourceBasedMultiTenantSpringLiquibase : Liquibase ran for tenant hogwarts
jMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
jMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8888 (http) with context path ''
jMain] s.c.b.s.MultiTenantServiceApplication : Started MultiTenantServiceApplication in 5.136 seconds (JVM running for 5.487)
```

| SUMMING UP

Software-as-a-Service (SaaS) via Software Multitenancy provides **great benefits** both for the customer and the service provider:

- **Simpler**
- **Cheaper**

| SUMMING UP

Architecting for Multitenancy require careful considerations. The biggest challenge lies in **data isolation** between tenants. As often, there is no “one-size-fits-all” pattern, but a set of proven patterns with different tradeoffs.

RECOMMENDATIONS

- Use **Database-per-tenant** or **Schema-per-tenant** if a strong data separation guarantee is the top priority and the number of tenants is modest. Pay attention to a database **migrations** mechanism.
- Use **Shared-Database-with-Discriminator** for maximum scalability if the number of tenants is high.
- Encapsulate the Multitenancy strategy implementation into a coherent **cross-cutting concern**, for flexibility and maintainability.

Time for questions!

