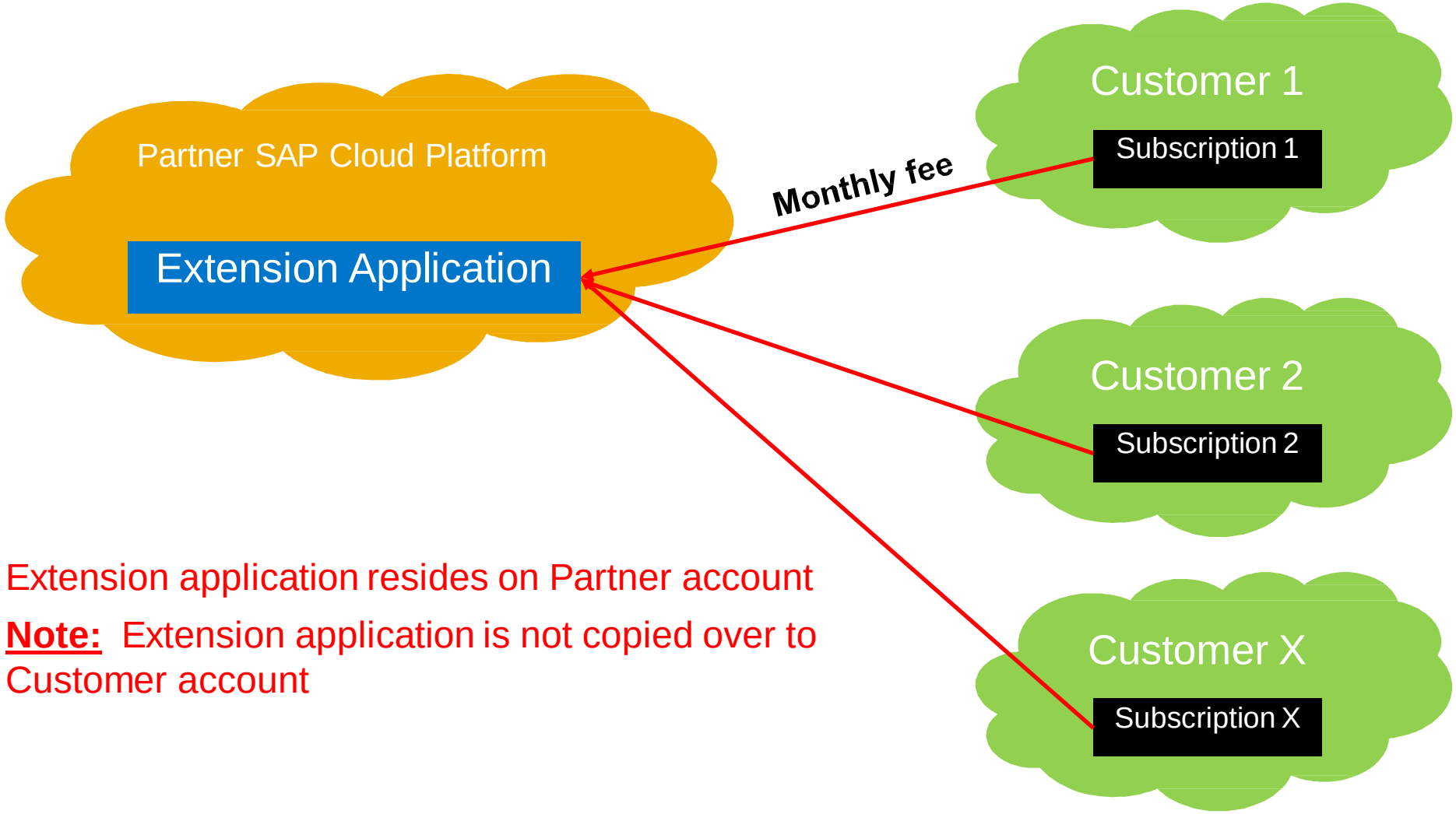
Concepts for Multitenancy

Multi-tenancy on SAP Cloud Plaform

- With SAP Cloud Platform you can develop and run **multi-tenant (tenant-aware)** applications
- Applications running on a **shared compute unit** that can be **used by multiple consumers (tenants)**
- Each consumer accesses the application through a **dedicated tenant-specific URL**.

What is a Multi-Tenant application ?



Extension application resides on Partner account

Note: Extension application is not copied over to Customer account

Some basic terminology

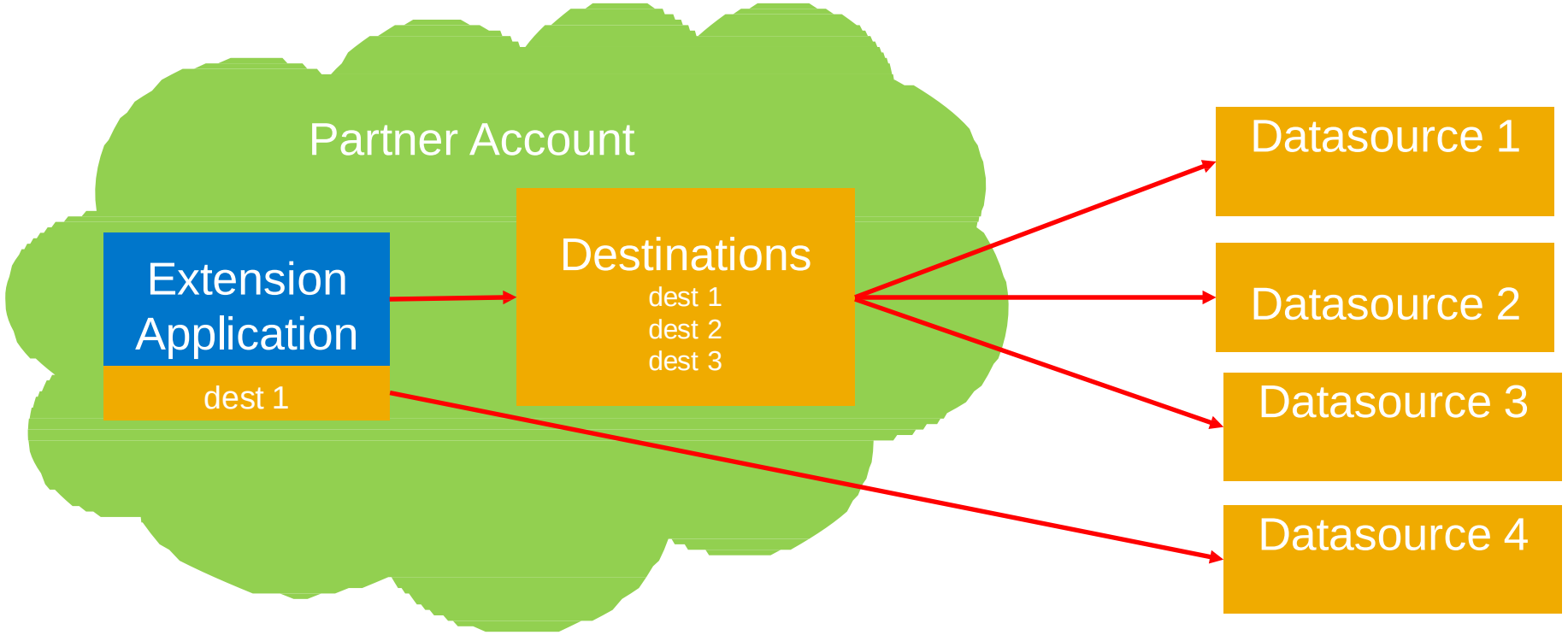
Customer = Consumer = Subscriber = Tenant = Entity paying for the subscription

Customer Account = SAP Cloud Platform account of the customer / consumer

Partner = Provider = Entity that created the Extension application

Partner Account = SAP Cloud Platform account of the partner / provider

Before we move on… What is a destination ?

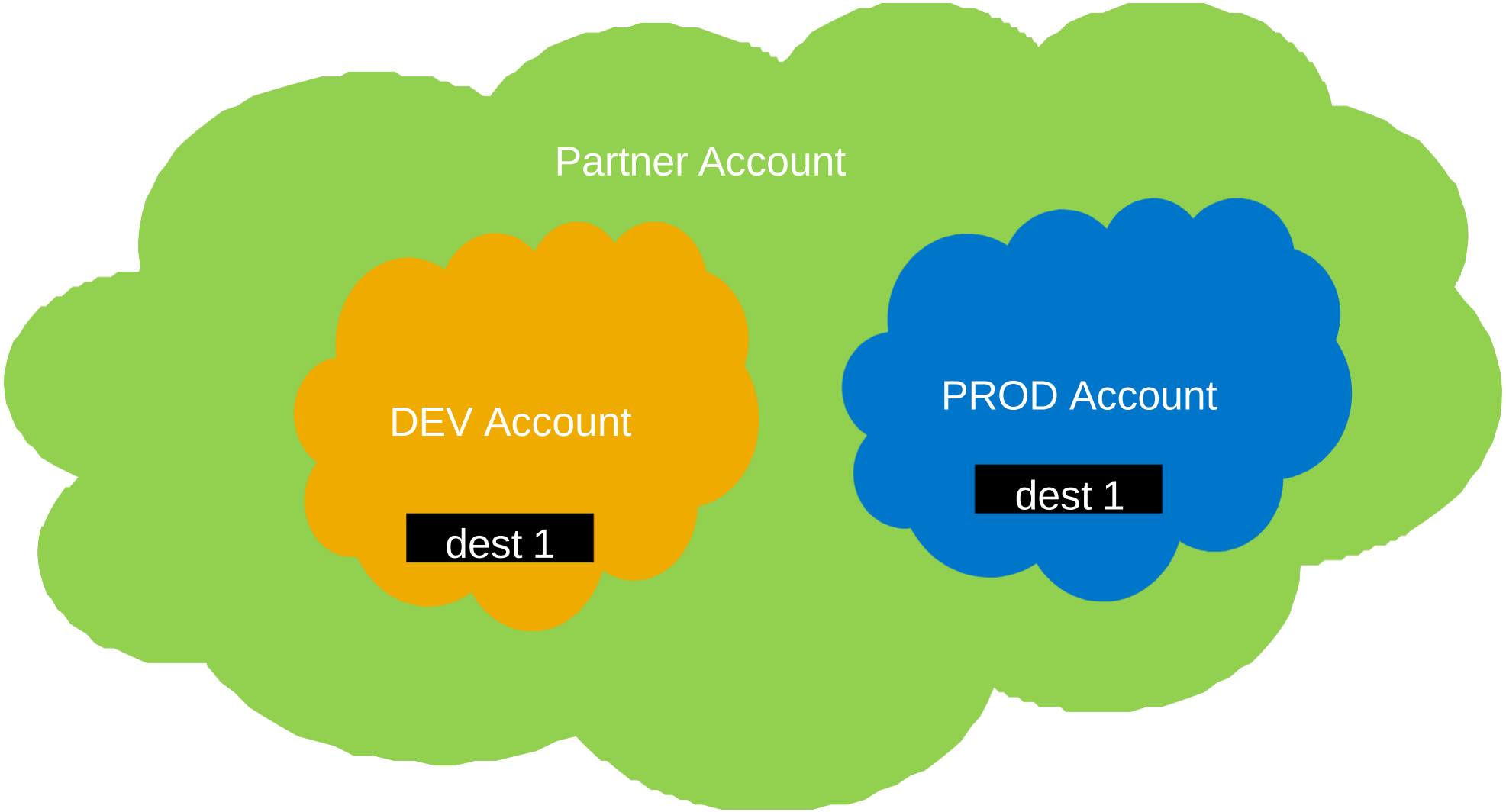


Destination = Secure connection string to a data source

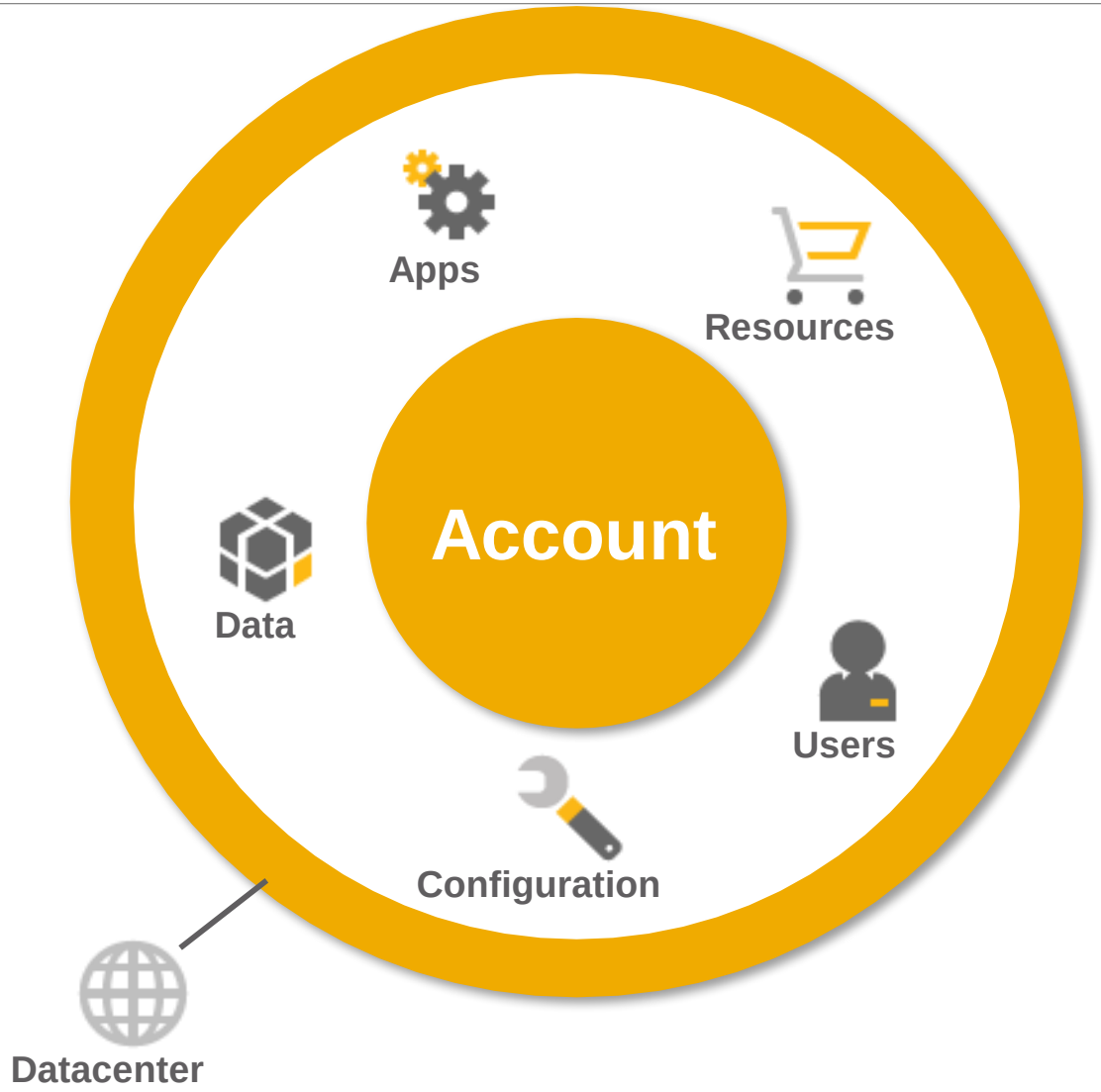
Destination – Account Level (Visible for the whole account)

Destination – Application Level (Visible for the application only)

Before we move on… What is a sub account ?



Understanding SAP Cloud Platform Accounts



Each account holds:

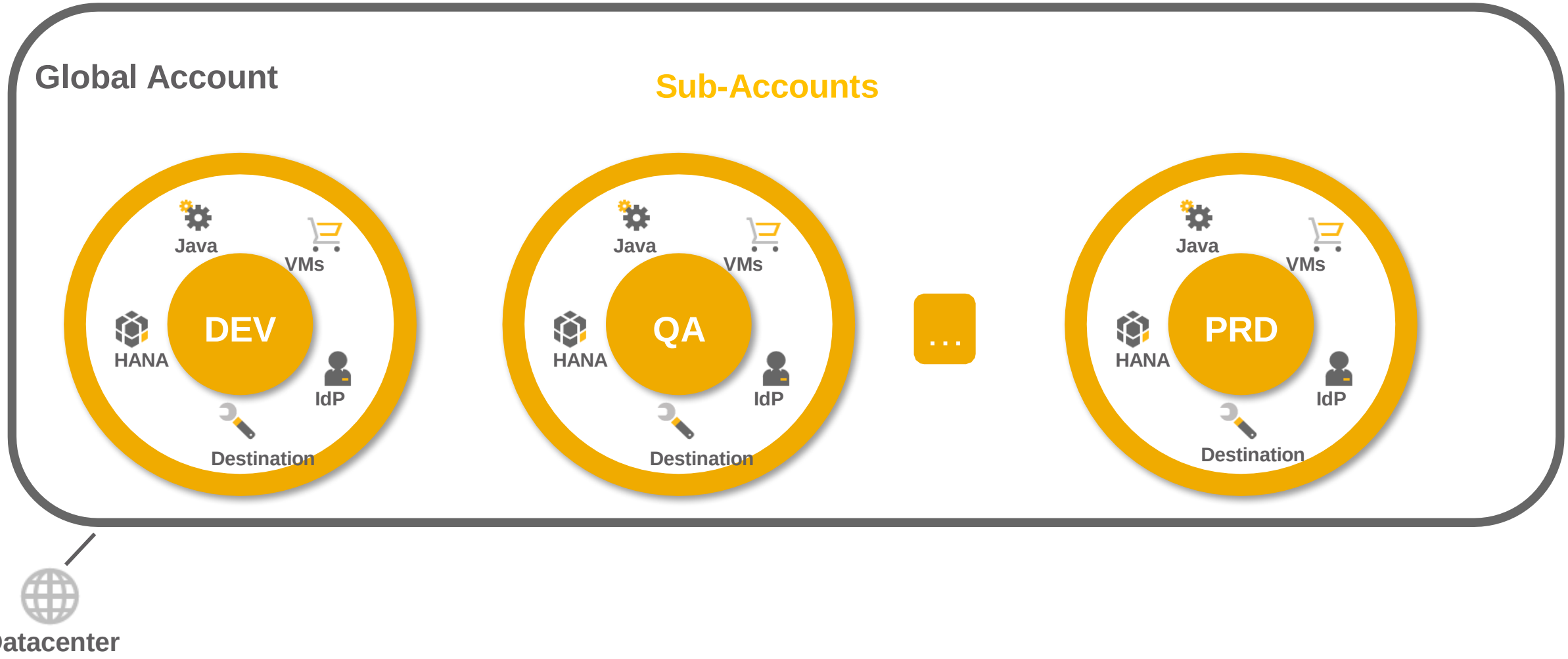
- **Resources that can be consumed by apps**
- **Users allowed to work in the account**
- **Apps deployed and running in the account**
- **Data written by apps running in the account**
- **Configuration for apps running in the account**

Each account is assigned to a datacenter. *

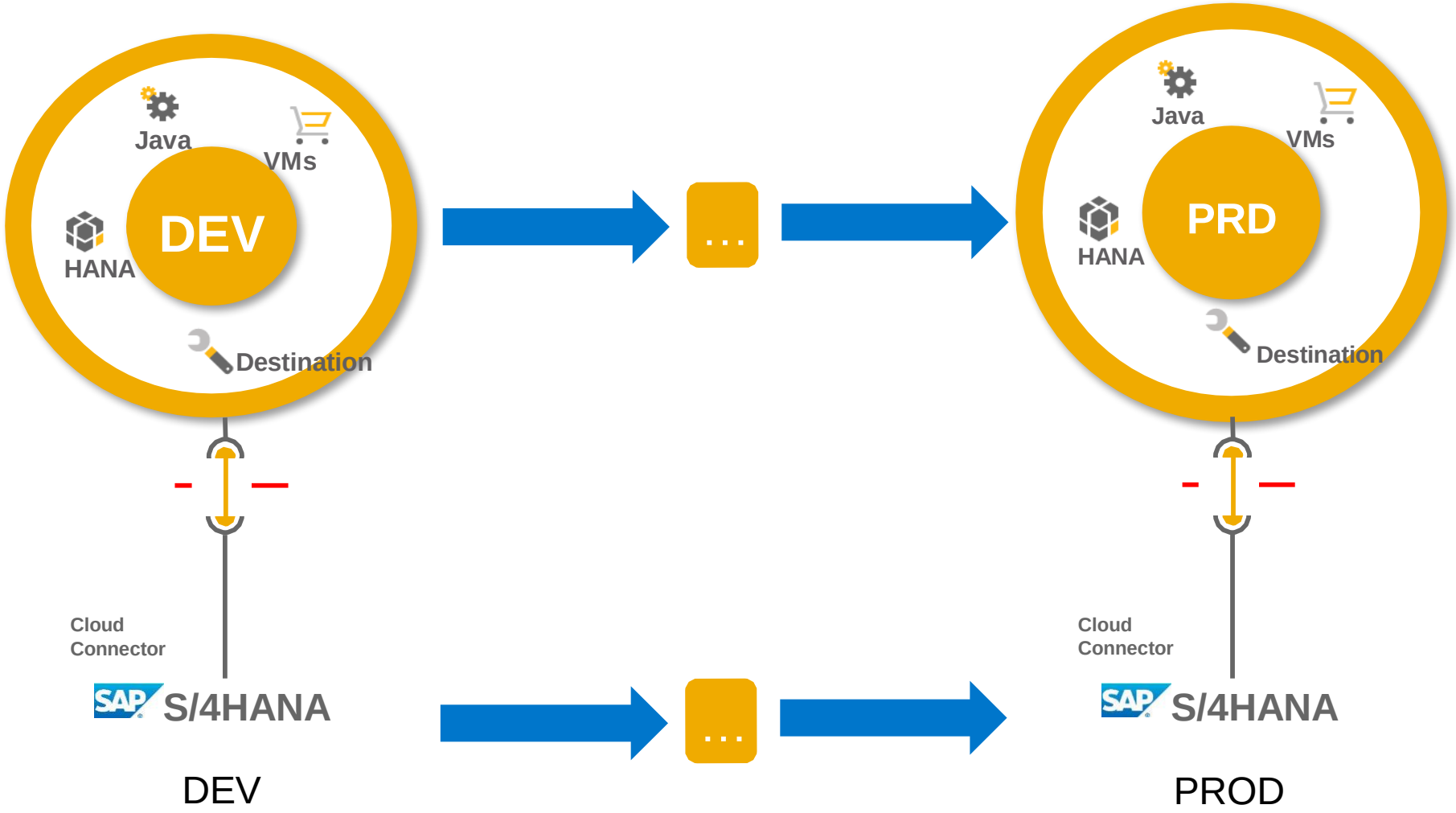
Each account is fully isolated.

*) New datacenters in setup

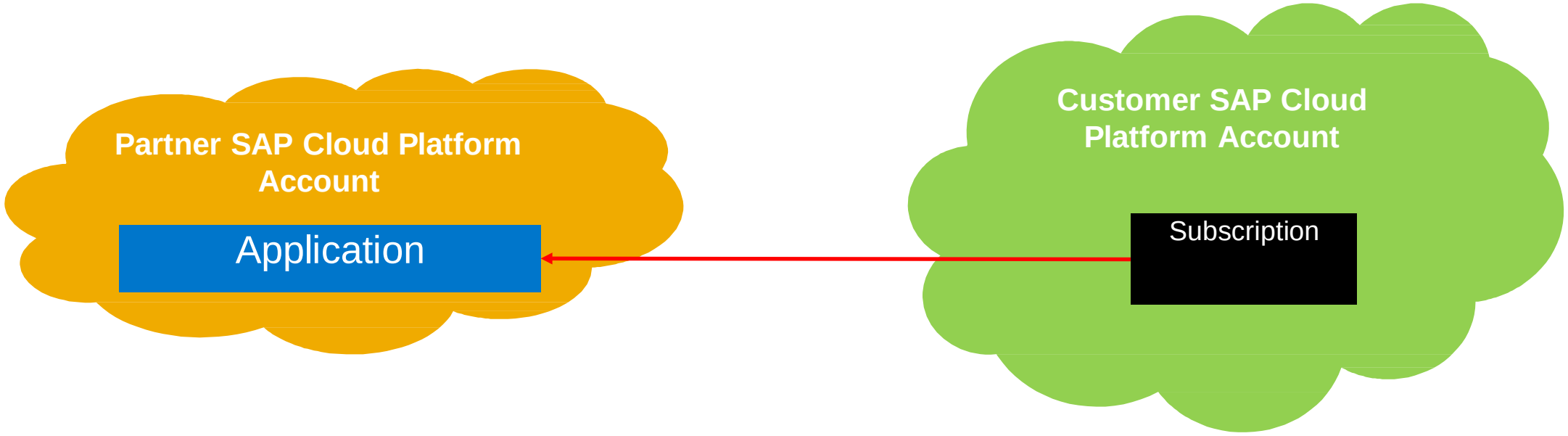
Sub – Accounts Concept



Pairing of Accounts with Backend Systems



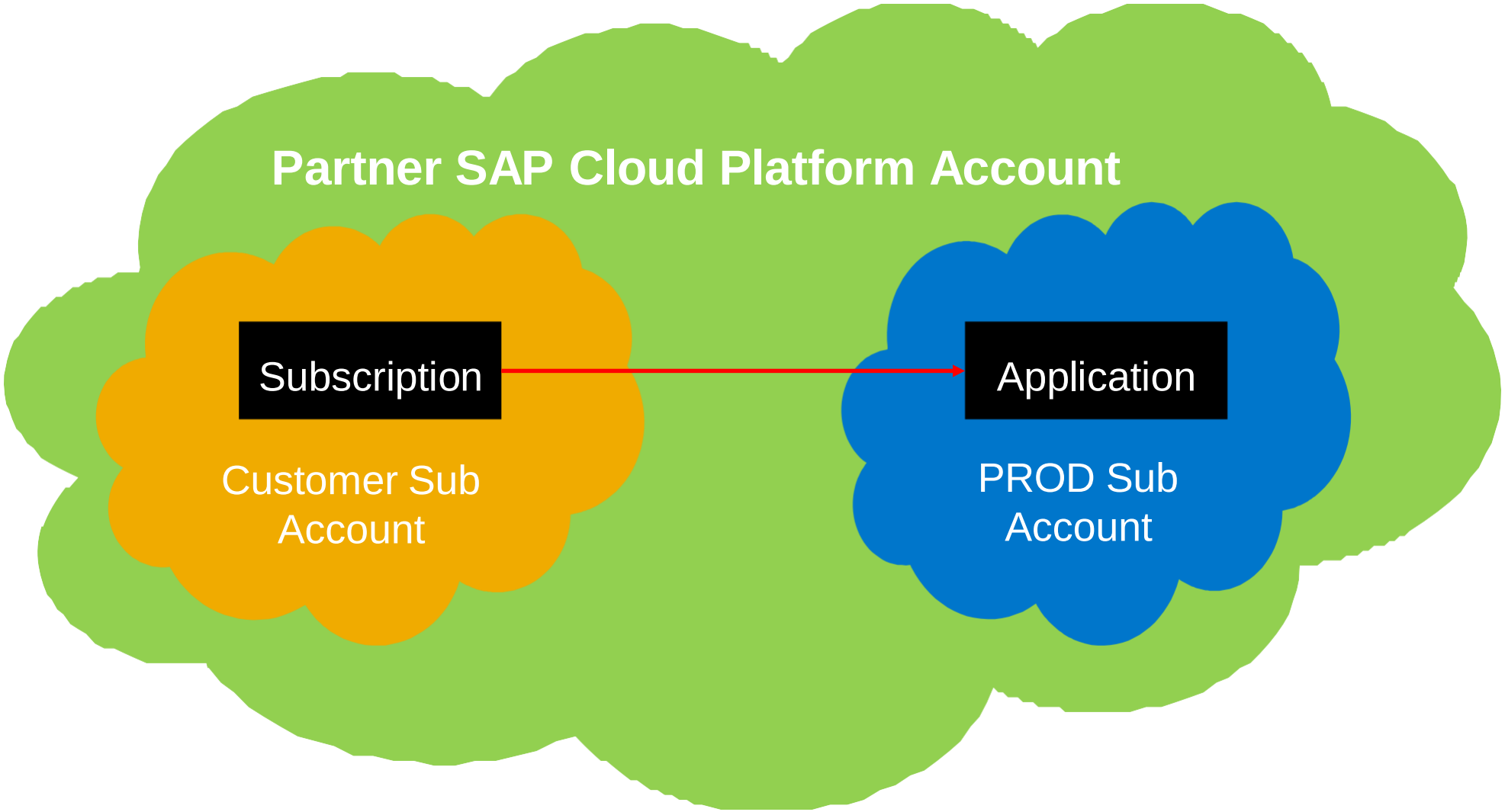
What is a Multi-Tenant application ?



Extension application resides on Partner account

Note: Extension application is not copied over to Customer account

Customer does not have an account…



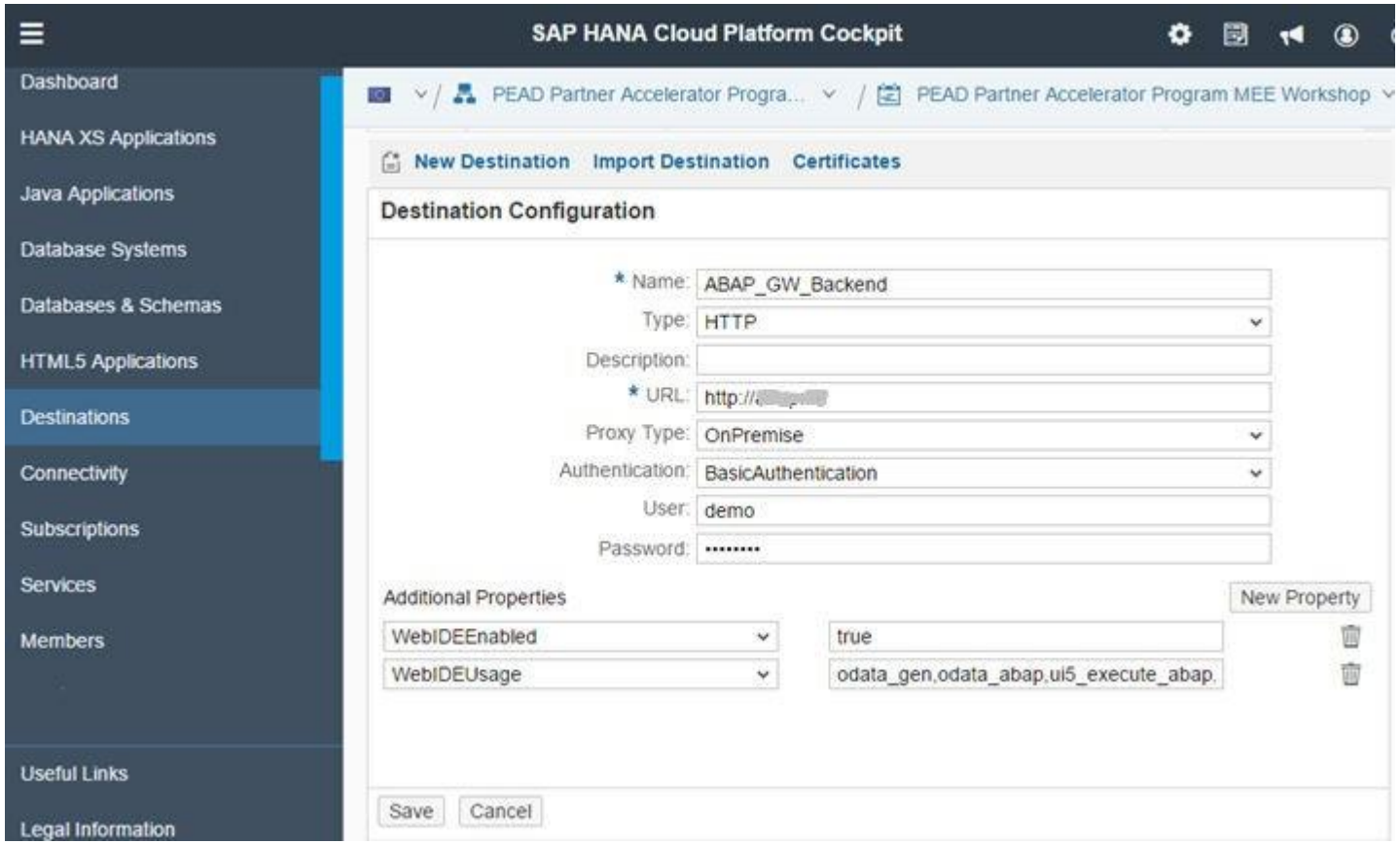
General Programming Guidelines for multitenant programming

- Shared in-memory data such as Java static fields, caches, static tables will breach tenant isolation
- Avoid any possibility that an application user can execute custom code in the application JVM, as this **will certainly** give them access to other tenants' data
- Avoid any possibility that an application user can access a file system, as this **will certainly** give them access to other tenants' data.

Multitenant Core Services in HCP

- **Connectivity Service**
- **Identity and Access Management Service**
- **Persistency Service**
- **Document Service**
- **Keystore Service**
- **Extensions service**

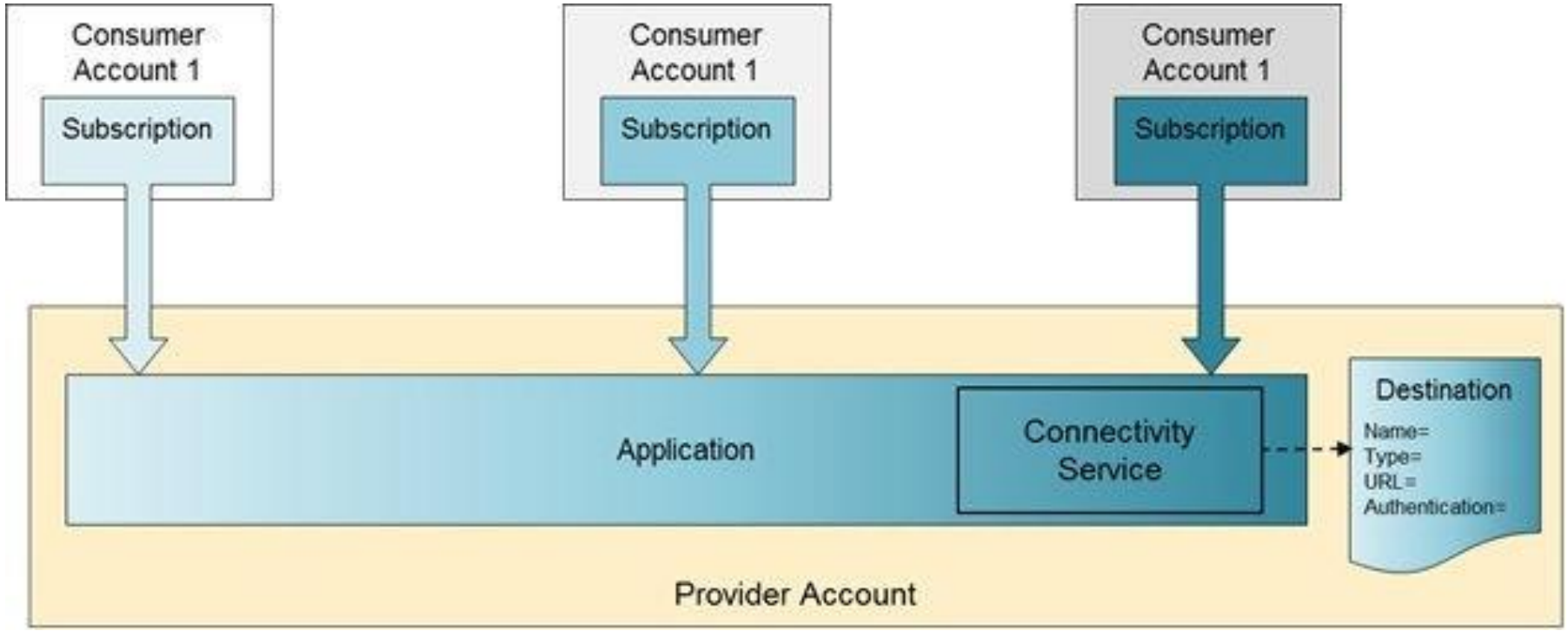
Destination



- Destinations are used for the outbound communication of a cloud application to a remote system
- Contain connection details for remote communication
- Contain symbolic names
- The connectivity service resolves the destination at runtime based on the symbolic name provided
- The currently supported destination types are **HTTP**, **Mail** and **RFC**.

Provider Specific Destination

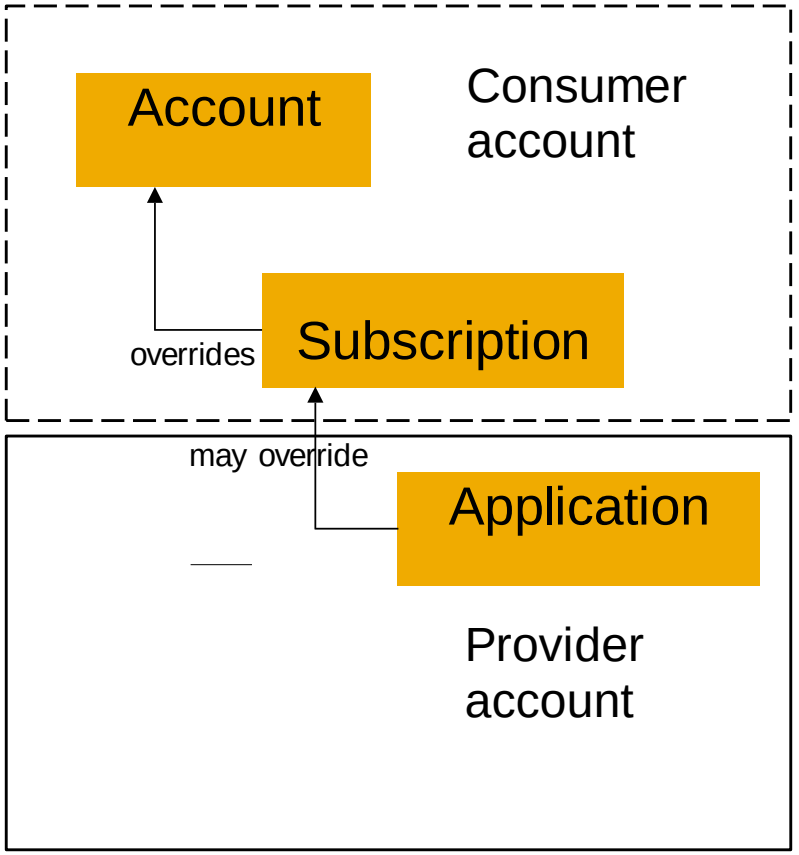
- A common shared destination (DestinationProvider=Application)
- The destination is always read from the provider account
- For example, weather data



Destination visibility

Destinations visibility according to the level:

1. Destination uploaded on account level - it is visible for the whole account
2. Destination uploaded on subscription level - it is only visible for the dedicated subscription
3. Destination uploaded on application level - it is visible by all tenants and accounts, regardless their permission settings

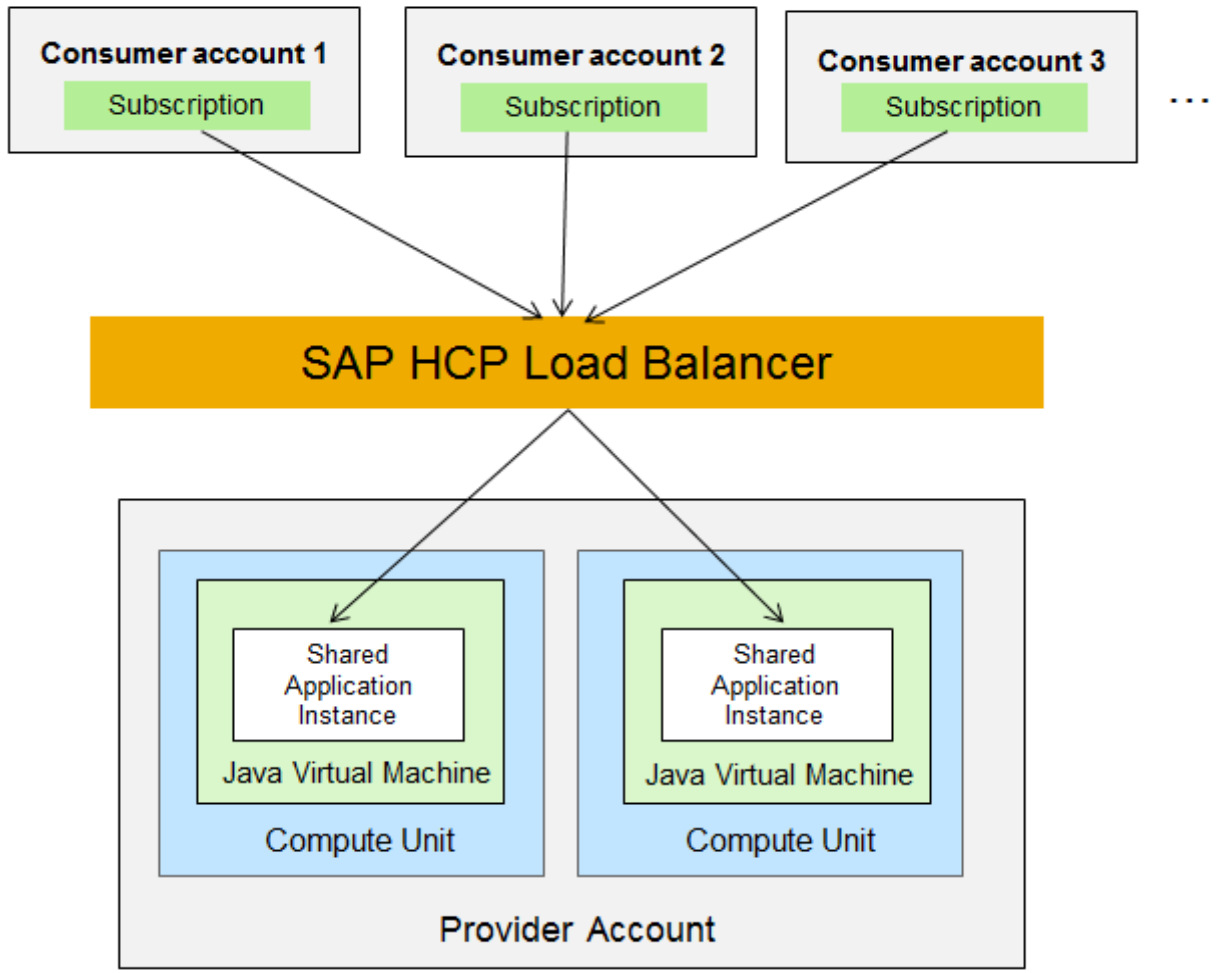


Scaling Applications

- Java Applications developed can run on one or more processes (VMs)
- Compute Unit quota for an account determines the number of processes that can be started
- One or more JVMs can be shared between all consumers in case of Multi-tenant applications
- Scaling down with soft shutdown on application processes

Advantages:

- Savings in resource costs
- Manage failover scenarios
- Savings in operational costs

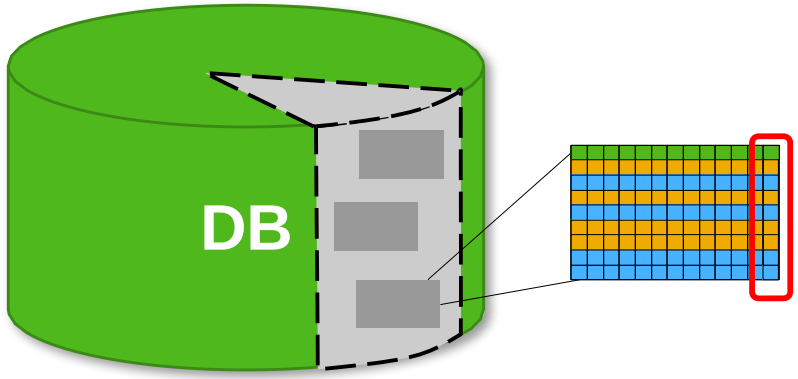


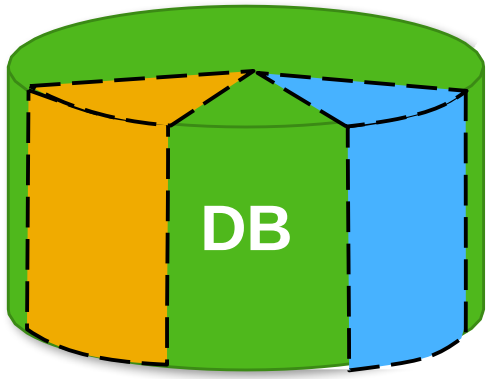
Data Level Isolation using the Persistency service

1. **Sharing a single database schema between all application consumers**
   a. Discriminatory field in each table
   b. Tenant ID can be used as a value in discriminatory field
   c. Each SQL statement should use the Tenant ID

2. **Schema level isolation**
   a. A schema for each tenant bound to the application
   b. The application uses JNDI to dynamically look up the data source

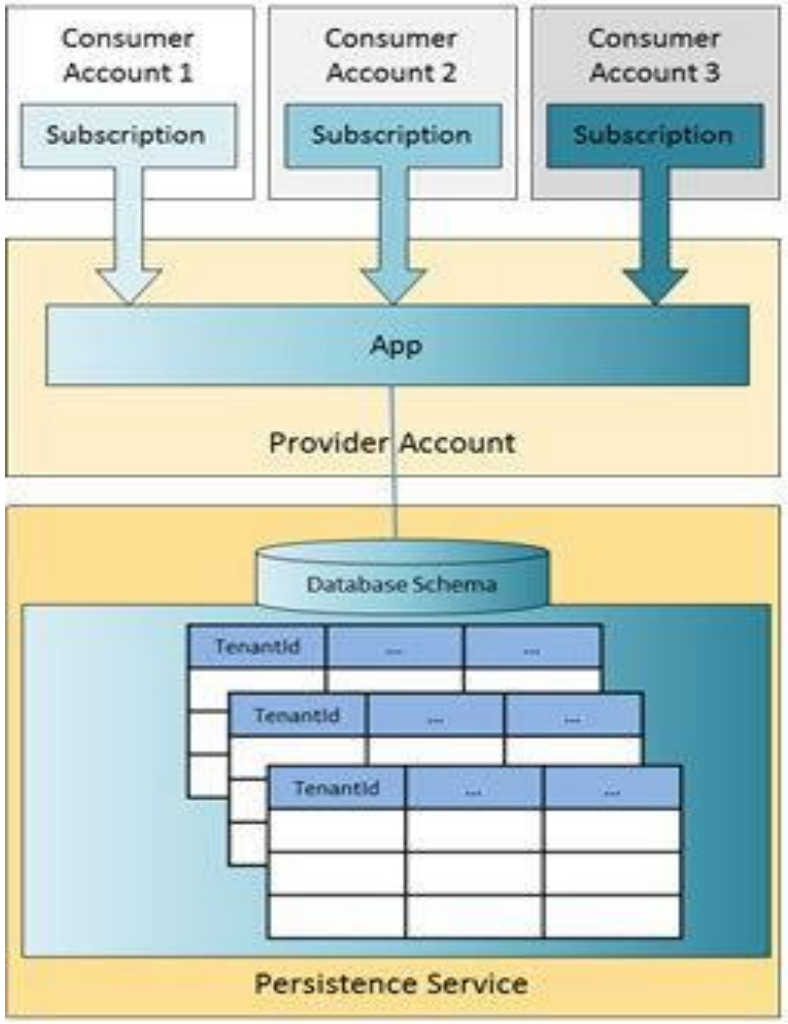
3. **Multi-Data Container (MDC) isolation***
   a. A DB container for each tenant bound to the application
   b. The application uses JNDI to dynamically look up the data source





* Planned on HCP

Sharing a single database schema between all application consumers



- A single DB Schema is shared between all consumers
- Data separation done via discriminator column
- Can be easily implemented using EclipseLink / JPA
- Current tenant must be provided to entity manager

```
@Entity
@Table(name = "T_PERSON")
@Multitenant
@TenantDiscriminatorColumn(name="TenantId", contextProperty
= "eclipselink.tenant-id", length=36)
@NamedQuery(name = "AllPersons", query = "select p from
Person p")
public class Person {
```

```
HashMap<String, String> properties = new HashMap<String,
String>();
properties.put(PersistenceUnitProperties.MULTITENANT_PROPERT
Y_DEFAULT, tenantId);
EntityManager em = emf.createEntityManager(properties);
```

Thank you

© 2014 SAP SE or an SAP affiliate company. All rights reserved.