BEYOND.2022

Software Development Lifecycle for ThoughtSpot Content

Making your dev team happy

Bryant Howell
Domain Architect - TSE
ThoughtSpot

Session Goal

To understand, for ThoughtSpot content:

SDLC delivery and management concepts

Core features involved in building SDLC process

Where to find documentation, examples, and tools for SDLC workflows

Agenda

1 **Introduction**

Introductio

2 **Example SDLC process**

Exemplum SDLC processus

3 **Demo**

Demonstratio

4 **ThoughtSpot Features used in SDLC**

Sine qua non

5 **Questions & Resources**

Quaestiones et Lectiones

Introduction

What is different about SDLC with ThoughtSpot?

Software Development Lifecycle (SDLC) is a broad term for the process of building, testing and deploying software

SDLC is concerned with **controlled** development and deployment

ThoughtSpot is **designed to let users answer their own questions**

Everyone can build their own content - building new Answers and Liveboards via Search or copying and modifying the existing content in different ways

SDLC in ThoughtSpot must balance control with self-service

Use cases for SDLC techniques

1. Moving to a new ThoughtSpot instance
2. Archiving Content
3. Dev -> Test -> Prod environments
4. SaaS multi-tenant deployment
5. Sync to regional ThoughtSpot instances

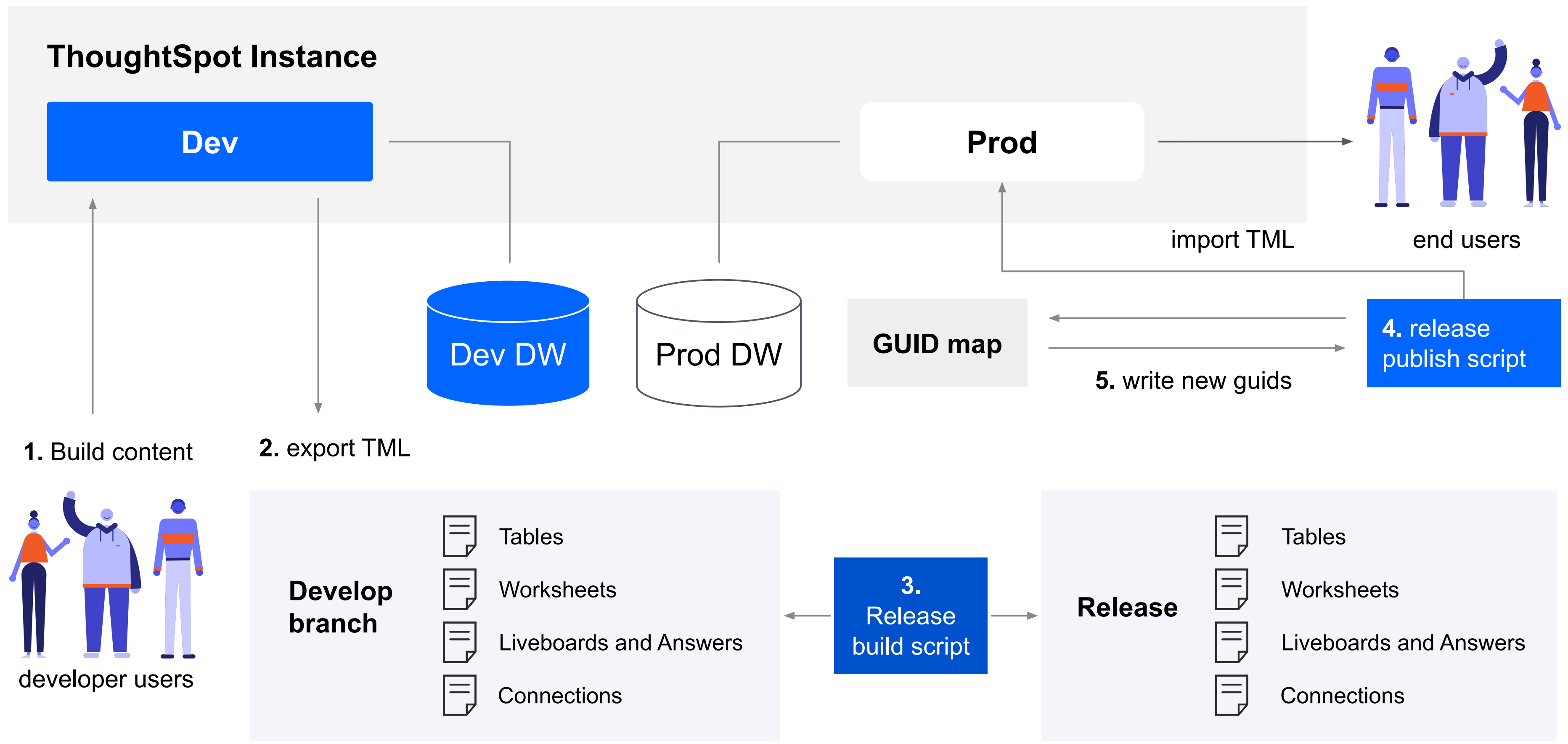
Example SDLC process

Development and Deployment Process Example

Let’s walk through a simple example of SDLC

- Two “environments” in ThoughtSpot: Dev and Prod
- All development work happens in Dev, then is moved to Prod
- Using Git as our source control system
- The data objects on Dev connect to Dev DW
- The data objects on Prod connect to Pro DW

SDLC Process - Simplified Diagram



Overview

Development work on data and content objects done in ThoughtSpot “dev” environment, pointing to Dev DW

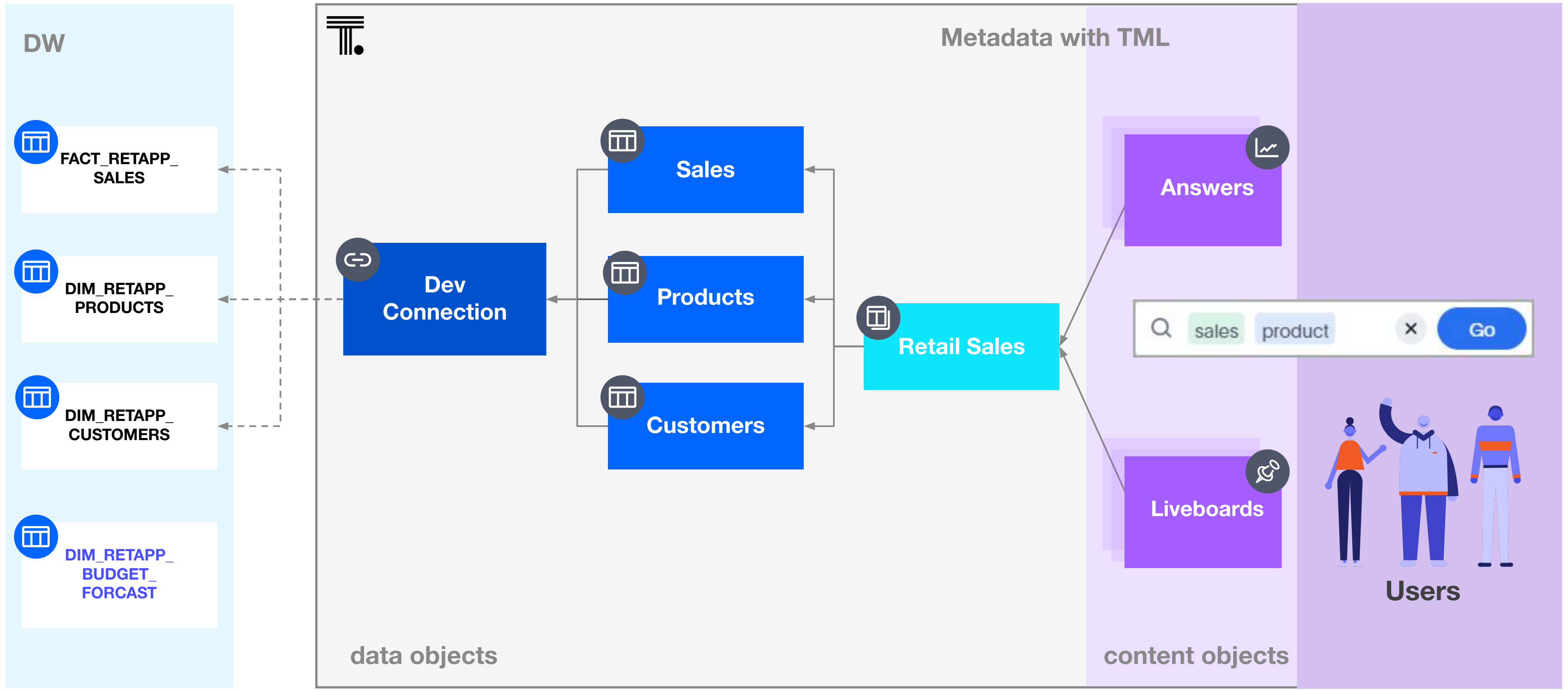
“Dev” content brought down as TML into Git repository on ‘develop’ branch

“Release process” takes ‘develop’ branch content and makes changes so it is pointed to Prod DW to build a ‘release’

The “release” TML is published to the Prod ThoughtSpot environment

The details of the “develop” object GUIDs and how they map to “prod” object GUIDs are recorded to allow later updates

ThoughtSpot Object Model Hierarchy



Demo

Download all content and re-publish with changes

For the demo, we'll kick off a script that does the following:

1. Download Answers and Liveboards modified in the last day
2. Commits the files in Git
3. Open files using TML library, change the titles slightly
4. Writes the changed versions to new files, committed in Git
5. Publishes the changed content
6. Shares the new content to the appropriate groups

Code and tools used in demo

The demo uses ThoughtSpot REST API V1 and ThoughtSpot Modelling Language

The code is available in the repository linked through the QR code

At the code level, the examples use the following Python packages:

- **thoughtspot-rest-api-v1****
- **thoughtspot-tml****
- **ts_rest_api_and_tml_tools**

** can be installed using **pip**

Components of SDLC process in ThoughtSpot

Components of SDLC in ThoughtSpot

ThoughtSpot features

ThoughtSpot data connections

ThoughtSpot data objects: Tables, Worksheets, Views

ThoughtSpot content objects: Answers and Liveboards

ThoughtSpot Modeling Language (TML)

REST API

Object Access (Sharing) to Groups

Organizations (multi-tenancy)

Additional technologies

Python or other programming language

Source Control System (Git)

Web server (apache, nginx)

Process

Service accounts

Developer users and groups

Transferring content ownership

Sharing / access control

ThoughtSpot Data and Content Objects

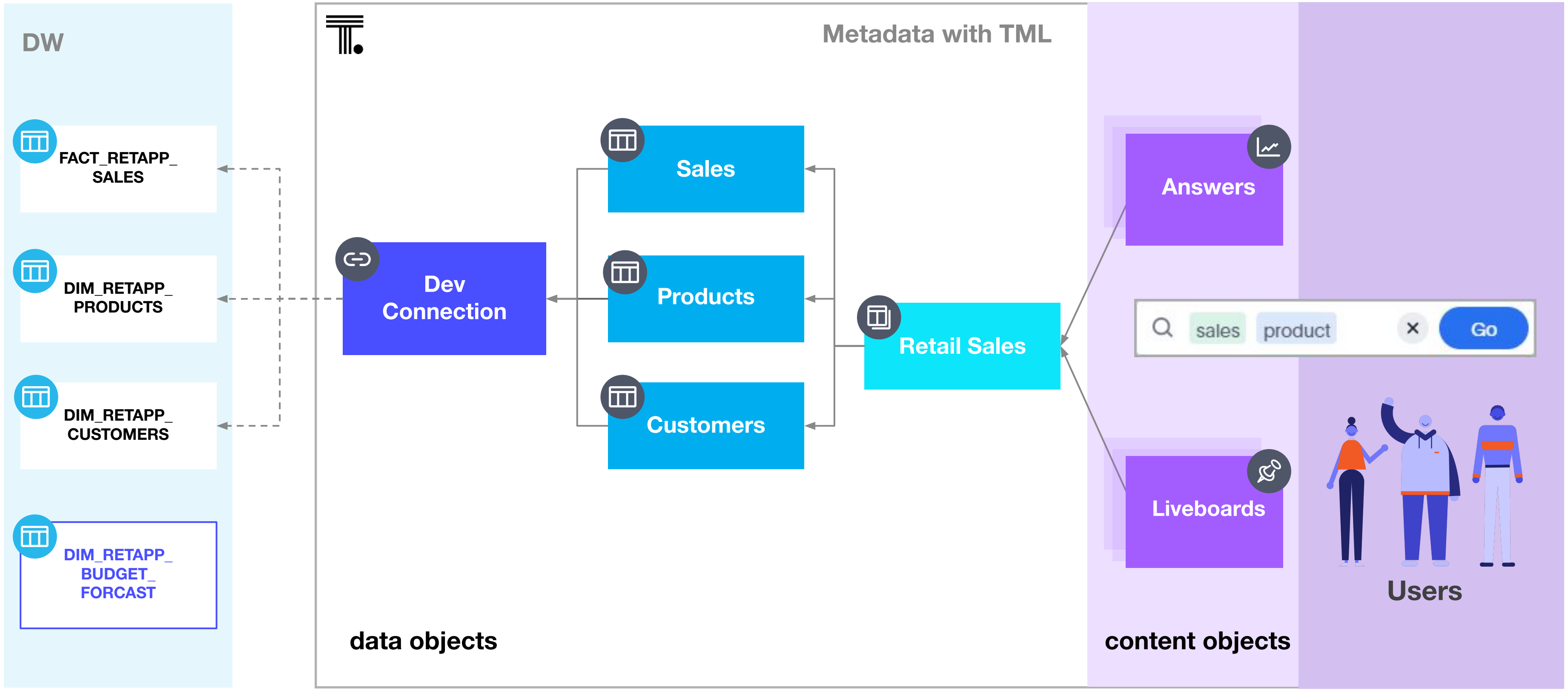
ThoughtSpot objects can roughly be split into Data and Content objects:

- **Data Objects:** Tables, Worksheets, Views
- **Content Objects:** Answers, Liveboards

Building out a structured SDLC process makes the most sense for **data objects** and **standardized content objects**

The goal is rapid development of **standardized content** and **self-service** for the **end users**

ThoughtSpot Object Model Hierarchy



Sharing, Ownership, and Service Accounts

Each object in ThoughtSpot has an **Owner/Author** with full rights

The owner or an admin can **share** content to **Users** or **Groups**

Sharing can give **Read-Only** or **Edit** rights

A user that does not represent a real person can be a **Service Account**

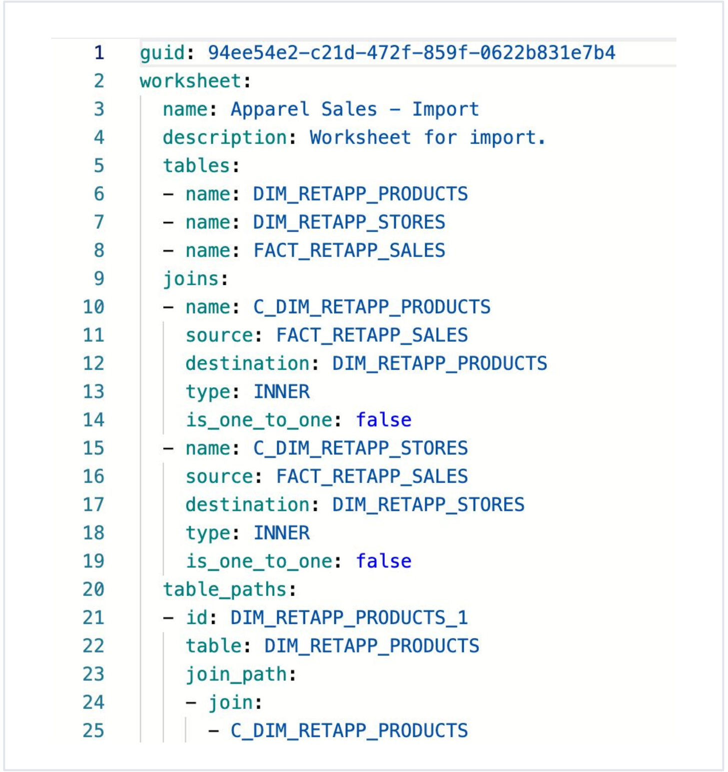
If a service account is the owner of an object in ThoughtSpot, and no one else has EDIT rights, the object is effectively “**locked**”

ThoughtSpot Modeling Language (TML)

Every object type on ThoughtSpot has a representation in ThoughtSpot Modeling Language (TML)

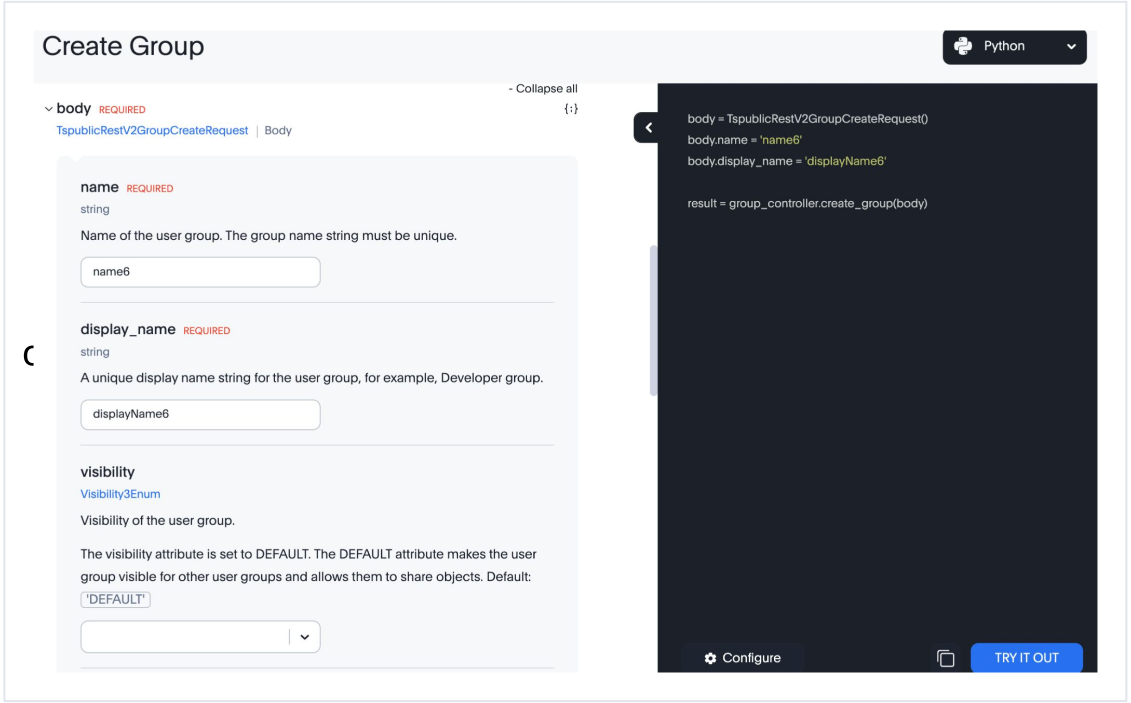
Fully documented at https://cloud-docs.thoughtspot.com/admin/ts-cloud/tml.html

You can see/edit TML through ThoughtSpot UI or use REST APIs for import and export



```yaml
guid: 94ee54e2-c21d-472f-859f-0622b831e7b4
worksheet:
  name: Apparel Sales - Import
  description: Worksheet for import.
  tables:
  - name: DIM_RETAPP_PRODUCTS
  - name: DIM_RETAPP_STORES
  - name: FACT_RETAPP_SALES
  joins:
  - name: C_DIM_RETAPP_PRODUCTS
    source: FACT_RETAPP_SALES
    destination: DIM_RETAPP_PRODUCTS
    type: INNER
    is_one_to_one: false
  - name: C_DIM_RETAPP_STORES
    source: FACT_RETAPP_SALES
    destination: DIM_RETAPP_STORES
    type: INNER
    is_one_to_one: false
  table_paths:
  - id: DIM_RETAPP_PRODUCTS_1
    table: DIM_RETAPP_PRODUCTS
    join_path:
    - join:
      - C_DIM_RETAPP_PRODUCTS
```

ThoughtSpot REST API


Create Group

Python

- Collapse all

body REQUIRED
TspublicRestV2GroupCreateRequest | Body

name REQUIRED
string
Name of the user group. The group name string must be unique.
name6

display_name REQUIRED
string
A unique display name string for the user group, for example, Developer group.
displayName6

visibility
Visibility3Enum
Visibility of the user group.
The visibility attribute is set to DEFAULT. The DEFAULT attribute makes the user group visible for other user groups and allows them to share objects. Default: 'DEFAULT'

```python
body = TspublicRestV2GroupCreateRequest()
body.name = 'name6'
body.display_name = 'displayName6'

result = group_controller.create_group(body)
```

ThoughtSpot has public REST APIs to perform all administrative actions

Ex.

- TML import and export
- Group management
- Object access (sharing)
- Listing of available objects

“Environments” in ThoughtSpot

An “environment” for a stage is structured using Groups with appropriate Sharing

“**Dev Data**” and “**Dev Content**” Groups in ThoughtSpot

One **Service Account** user in ThoughtSpot

Objects **owned** by the Service Account represent objects that are “checked-in” to the **develop branch**

Objects will be owned by individuals prior to being checked-in or to signal they are “checked-out”

Organizations for environment separation

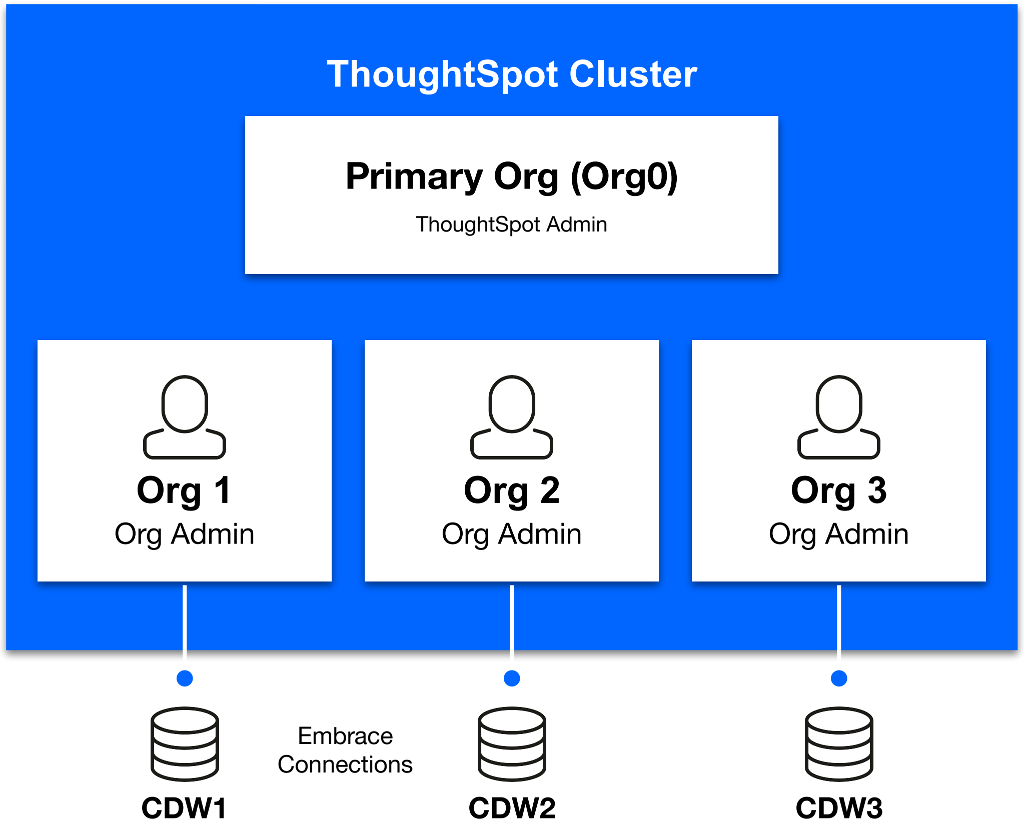
Organizations is an upcoming feature in ThoughtSpot allowing defined sub-tenants on a single ThoughtSpot instance

Creates a level of tenant separation above what is currently accomplished through Groups/Sharing

Users, groups and content will **belong within** an Organization.

Users and REST API calls will log into one Org at a time

All activity within a session will be scoped to the Org



Questions and Resources

CodeSpot

Developer Documentation