PROOF OF CONCEPT

Connect Shopify to SAP Business Data Cloud

Implementation Guide and Lessons Learned

A lightweight CAP service that imports Shopify customer data into a structured SAP HANA Cloud table. Then, SAP Datasphere orchestrates the import and provides the analytical model consumed by SAC.

Document map

Contents

1. Introduction

Symptom

“No generic GraphQL client connector is currently available within SAP Datasphere” (SAP note 3785710).

To integrate SAP Datasphere (BDC) with the Shopify GraphQL API, note 3785710 states a middleware layer is required to consume Shopify's GraphQL endpoints and expose the data in a format natively supported by SAP Datasphere.

Recommended approach: SAP Integration Suite — Cloud Integration (CPI)

Refer to SAP Note 3696110 for details.

Proof of concept objective

In organizations where SAP PI or another integration platform is controlled by a separate team, the analytics team may depend on that team to ingest non-SAP API data.

For light data integrations, this proof of concept demonstrates that Hana Cloud together we custom CAP service can be the middleware between external API’s and SAP BDC.

> Scope note: This pattern is positioned for light data integrations and proofs of concept. It is not presented as a universal replacement for enterprise middleware, bulk-ingestion platforms, or governed integration programmes.

Future SAP BDC releases (July 2026 news)

SAP BDC release updates, describe a new “REST API for Replication Flows” option to connect BDC to any external REST API as a source in replication flow and load data directly into object store space.

This will be the best approach to design scalable and sustainable Datasphere API integrations (in the future).

Refer to SAP Note 3696110 for details on requesting activation.

2. Proof of Concept architecture

Retrieve → Validate / Transform → Store → Model / Orchestrate → Analyze

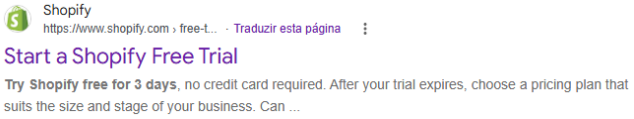
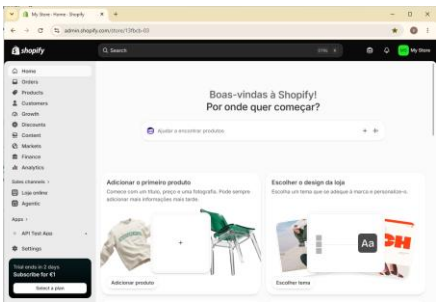
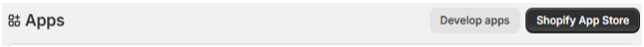
Shopify	CAP service	HANA Cloud	Datasphere	Analytics Cloud

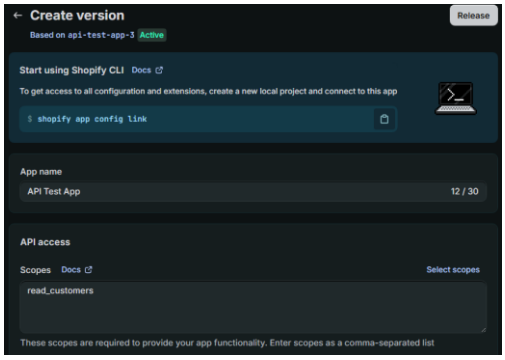
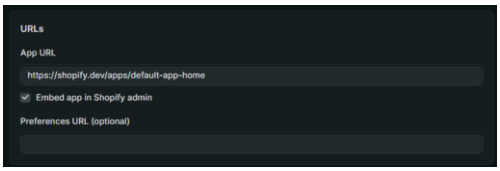
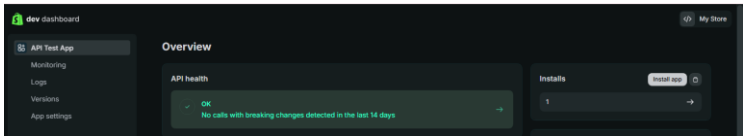
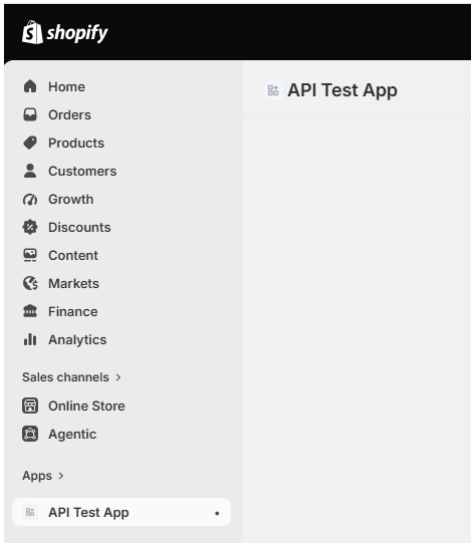
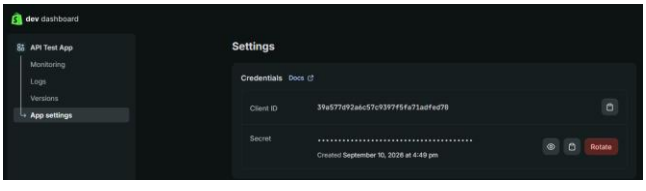
Development steps:

Shopify External System	Create a Shopify trial account and demo store. Add representative customer master data. Create/configure the custom app and customer-read API scope. Obtain the client ID and secret; exchange them for an Admin API token.
SAP Hana Cloud Integration persistent layer	Provision an SAP HANA Cloud trial instance. Create TABLE_SHOPIFY_CUSTOMER based on selected Shopify API fields. Create a technical database user and grant SELECT, INSERT, and UPDATE privileges.
CAP Service Integration action	Create an SAP BTP trial account. Enable SAP Business Application Studio. Test BTP – Shopify connectivity. Test BTP – Hana Cloud connectivity. Create a CAP application and the ImportShopifyCustomers action. Deploy the service to Cloud Foundry.
SAP Datasphere Semantic Model & Data Orchestration	Activate the SAP Datasphere trial. Create an SAP HANA connection to the database. Create a remote table and analytical model over the integration table. Create a Generic HTTP connection to the CAP endpoint. Create an API Task and Task Chain and schedule the periodic import.
SAC Analytics	Create data insights for analysis

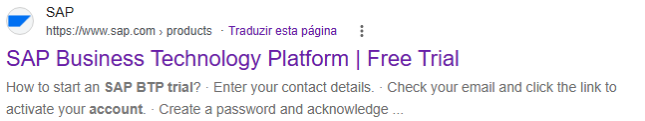
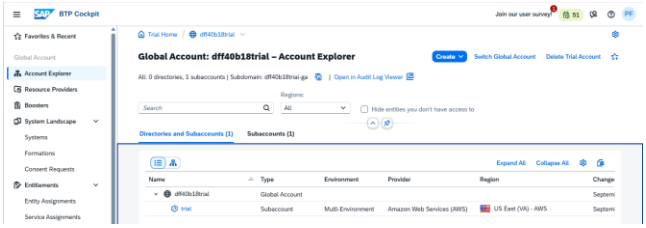
3. Solution setup

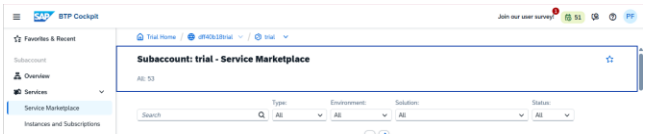
3.1 External System - Shopify

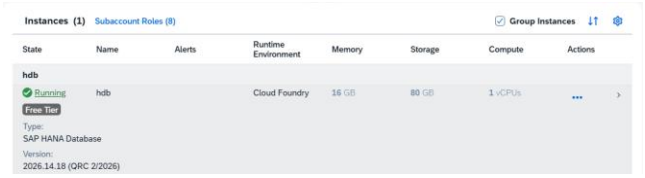
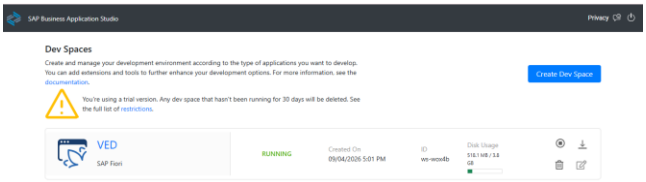
Step	Action	Screenshot
-	Create a trial Shopify Online Store	
3.1.1	You can choose one of many cloud service providers from where you can consume data via API call. For this demo choose Shopify free trial	
3.1.2	Inside your store, create your first “Customers” master data records	
-	**Create the “custom app”** Enable Admin API access and assign the customer-read scope required by the proof of concept.	
3.1.3	Go to “Settings -> Apps” and choose “Develop Apps”.	

3.1.4	You jump into “Dev dashboard” area and you can create your new App. In API Access, add “read_customer”	
3.1.5	In App URL, enter “https://shopify.dev/apps/default-app-home”	
3.1.6	Now, you need to install the app in your trial store. Choose “Install app”	
3.1.7	After installation, App will become visible in your store	
3.1.8	Under App settings, copy Client ID and Secret to your notes. You will need it later to test and implement “CAP –> Shopify” autentication.	

3.2 SAP BTP Hana - database first

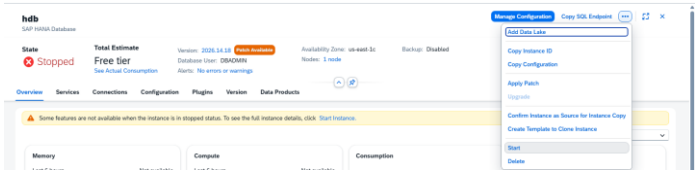
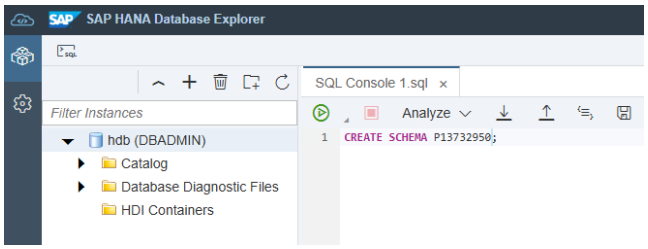
Step	Action	Screenshot
-	Create a trial SAP BTP Account	
3.2.1	Start SAP BTP platform free trial	
3.2.2	You access a free trial subaccount, from where you can subscribe applications and instances	
-	Under “Service Market Place”, subscribe all necessary applications and instances for this demo	

3.2.3	Active the following services:
	• SAP Business Application Studio
	  • Application trial
	• HTML5 Application Repository Service
	  • app-host
	• SAP HANA Cloud
	  • Service hana-free
	  • Application tools
	• SAP HANA Schemas & HDI Containers
	  • hdi-shared
		

3.2.4	Create an “hdb” Hana Database	
3.2.5	Create a “DEV” SAP Business Application Studio space.	
-	The following steps are executed in SAP Business Application Studio Terminal, to verify that you can connect to Shopify API from this SAP BTP platform.	
3.2.6	Use the “store id”, “app client ID” and “API secret” to request an Admin API access token.	curl -X POST "https://<store>.myshopify.com/admin/oauth/access_token" \ -H "Content-Type: application/json" \ -d '{"client_id":"<client-id>","client_secret":"<api-secret>","grant_type":"client_credentials"}'
3.2.7	Validate the token Call a harmless Shopify Admin API endpoint before placing the token in CAP application configuration.	curl -H "X-Shopify-Access-Token: <shpat-token>" \ "https://<store>.myshopify.com/admin/api/<version>/shop.json"

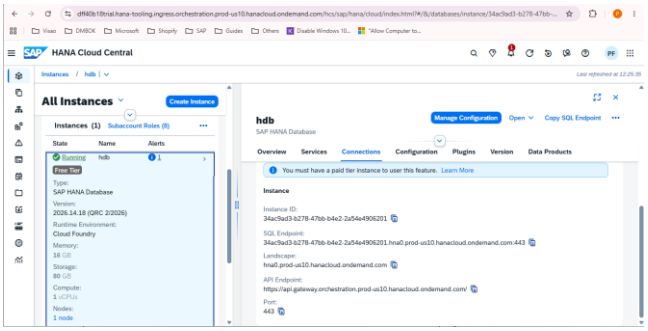
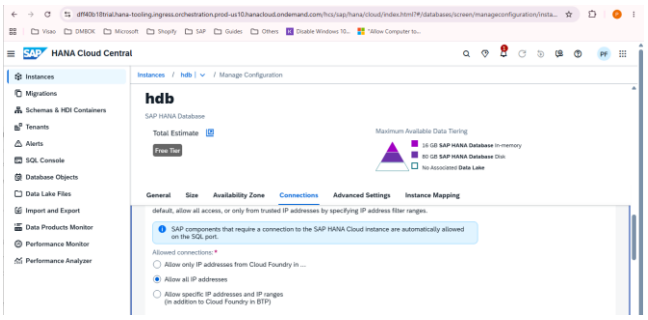
3.2.8	Inspect the real payload from shopify api using BAS terminal. This returns the structure that you receive.
	
	You will need this to create hana cloud integration table.
		curl -X GET "https://<store>.myshopify.com/admin/api/2024-04/customers.json" -H "X-Shopify-Access-Token: <TOKEN>" -H "Content-Type: application/json"
		
		{
		"id": 10513263755593,
		"email": "pedrofilipe.ve@gmail.com",
		"first_name": null,
		"last_name": null,
		"phone": null,
		"state": "disabled",
		"currency": "EUR",
		"orders_count": 0,
		"total_spent": "0.00",
		"created_at": "...",
		"updated_at": "...",
		"verified_email": true
		}

At this step you activate SAP BTP services and confirm Shopify API connectivity and authentication.

-	The following steps are executed in SAP Hana Cloud Central App, to create integration tables (database first approach).	
3.2.9	Start Database and open SAP Hana Database Explorer	
3.2.10	Create a new schema to create the persistent “Cloud Integration tables” CREATE SCHEMA <YOUR_SCHEMA>;	
3.2.11	Create a structured integration table. This hana structured table supports data typing, remote-table exposure, semantic modelling, and downstream reporting.	CREATE TABLE <SCHEMA>.TABLE_SHOPIFY_CUSTOMER (SHOPIFY_ID BIGINT PRIMARY KEY, EMAIL NVARCHAR(255), FIRST_NAME NVARCHAR(100), LAST_NAME NVARCHAR(100), PHONE NVARCHAR(50), CUSTOMER_STATE NVARCHAR(50), VERIFIED_EMAIL BOOLEAN, CURRENCY NVARCHAR(10), ORDERS_COUNT INTEGER, TOTAL_SPENT DECIMAL(15,2), COUNTRY NVARCHAR(100), COUNTRY_CODE NVARCHAR(10), MARKETING_STATE NVARCHAR(50), CREATED_AT NVARCHAR(50), UPDATED_AT NVARCHAR(50));
3.2.12	Create communication user BDC_USER and give access to this table schema `GRANT SELECT, INSERT, UPDATE ON <SCHEMA>.TABLE_SHOPIFY_CUSTOMER TO <TECHNICAL_USER>;` The CAP application will connect to Hana DB using BDC_USER	CREATE USER BDC_USER PASSWORD "<secure-password>" NO FORCE_FIRST_PASSWORD_CHANGE; GRANT SELECT, INSERT, UPDATE ON <SCHEMA>.TABLE_SHOPIFY_CUSTOMER TO BDC_USER;

At this step the SAP HANA Cloud schema and tables already exist.

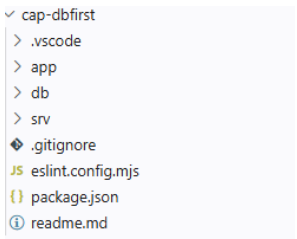
CAP service will not own or deploy tables through HDI. Instead, the service will connect to the existing database with a dedicated technical user and execute SQL against designated integration schema.

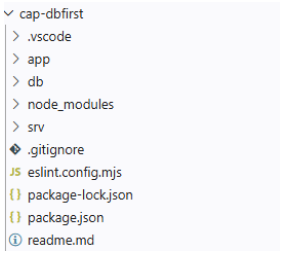
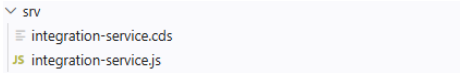
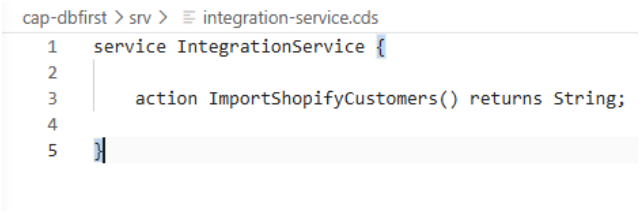
-	Open Trial Hana Cloud for external connections	
3.2.13	Go to hana Instance, keep note of “SQL endpoint” information for later datasphere connection creation.	
3.2.14	Go to “Manage Configuration”. Under Connections, “Allow all IP addresses” so that Trial Datasphere account can connect.	

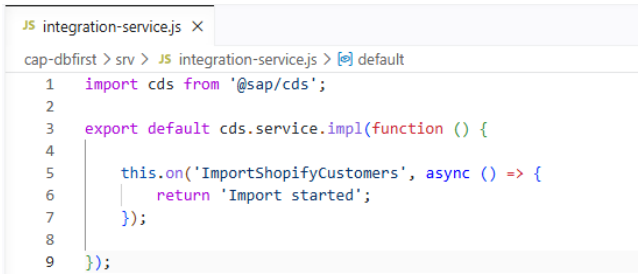
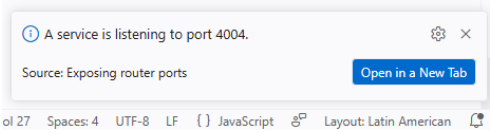
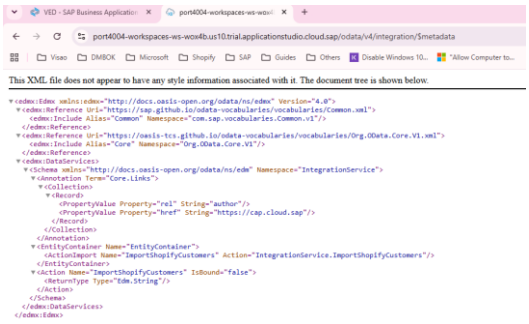
At this step the SAP HANA Cloud schema is available to be consumed in Datasphere virtual models.

3.3 SAP BTP CAP - integration service

Step	Action	Screenshot
-	Create a completely fresh CAP project in SAP Business Application Studio	

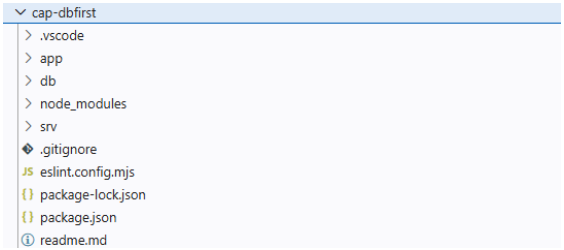
3.3.1	Open the BAS dev space terminal, initialize ‘cap-dbfirst’ with the Node.js runtime, and enter the project directory.
	`cds init cap-dbfirst --nodejs`
	`cd cap-dbfisrt`
		

<table>
<tr><td>3.3.2</td><td>Install dependencies
<pre>npm install
npm install @cap-js/hana
npm install @sap/hana-client</pre></td><td> 
</td></tr>
<tr><td>3.3.3</td><td>Create ‘integration-service’ files under “srv” folder
 
</td><td></td></tr>
<tr><td>-</td><td>Create a minimal action service</td><td></td></tr>
<tr><td>3.3.4</td><td>In “srv/integration-service.cds” declare the public OData service contract and the ImportShopifyCustomers action.</td><td> 
</td></tr>
<tr><td>3.3.5</td><td>In “srv/customer-service.js” will implement the action, calls Shopify, maps fields, opens the HANA connection, executes MERGE statements, handles errors, and returns the processed-row count.

For now it’s just a simple action declaration. Will be improved later.</td><td> 
</td></tr>
<tr><td>3.3.6</td><td>Start the service using terminal command cds watch</td><td> 
</td></tr>
<tr><td>3.3.7</td><td>Follow localhost link (available in terminal output) to confirmt ‘Integration’ service is active and ‘ImportShopifyCustomers’ action metadata exists
 
</td><td> 
</td></tr>
</table>

At this step a new CAP service is defined and started locally. You can call it but the service isn't doing anything yet.

-	Connect CAP to External API (Shopify) Connect CAP to Hana Cloud Implement the real business flow	

3.3.8	Install hana dependencies
	`npm install @cap-js/hana`
	`npm install @sap/hana-client`
	`npm install axios`
		

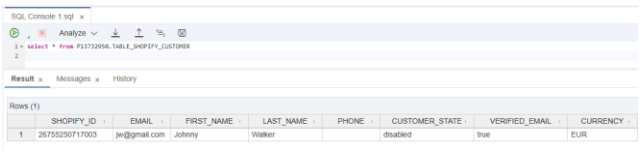
3.3.9	Make sure package.json contains a HANA configuration.	

3.3.10	Update “srv/integration-service.js” and map shopify fields into hana table fields.	

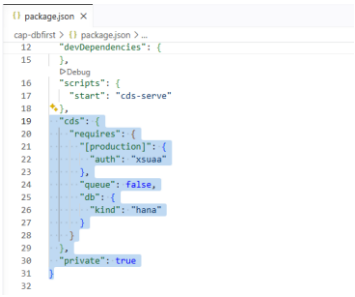
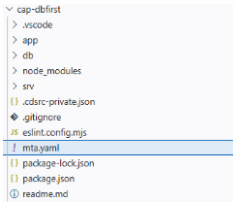
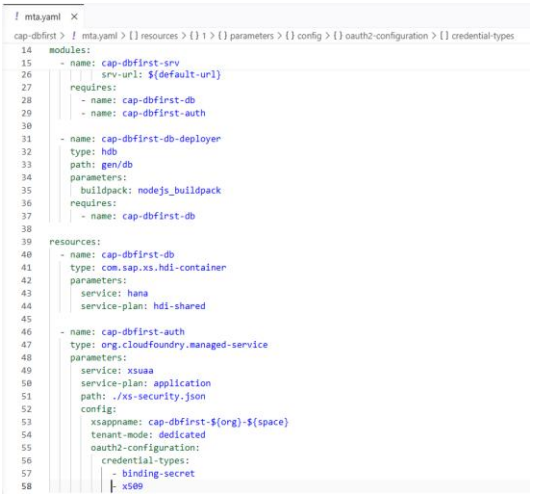
integration-service.js

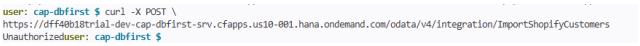
cap-dbfirst > srv > integration-service.js > default > cds.service.impl() callback > on('ImportShopifyCustomers') callback

```
import cds from '@sap/cds';
import hana from '@sap/hana-client';
import axios from 'axios';

function sqlSafe(value) {
  return String(value || '').replace(/'/g, "''");
}

function connectToHana() {

  const conn = hana.createConnection()

  conn.connect({
    serverNode: '34ac9ad3-b278-47bb-b4e2-2a54e4906201.hna0.prod-us10.hanacloud.ondemand.com:443',
    uid: 'BDC_USER',
    pwd: [redacted],
    encrypt: true
  })

  return conn
}


export default cds.service.impl(function () {

  this.on('ImportShopifyCustomers', async () => {

    const response = await axios.get(
      // `https://${cds.env.shopify.store}.myshopify.com/admin/api/2024-04/customers.json`,
      `https://[redacted]myshopify.com/admin/api/2024-04/customers.json`,
      {
        headers: {
          // 'X-Shopify-Access-Token': cds.env.shopify.token
          'X-Shopify-Access-Token': 'shpat_[redacted]b'
        }
      }
    );

    const customers = response.data.customers;

    // const db = await cds.connect.to('db');
    const conn = connectToHana()

```

		
3.3.11	Right now, session is running only whit our local cds watch. Call CAP action locally to transfer records from Shopify to Hana Cloud curl -X POST \ http://localhost:4004/odata/v4/integration/ImportShopifyCustomers” Then, check table Content in Hana Explorer	

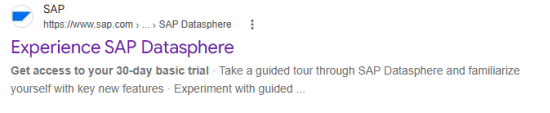
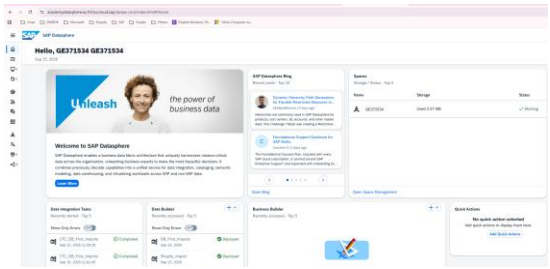
At this step a new CAP service is extract data from Shopify and load it into Hana Cloud. The service is active but can only be called locally inside Business Application Studion (BAS) terminal.

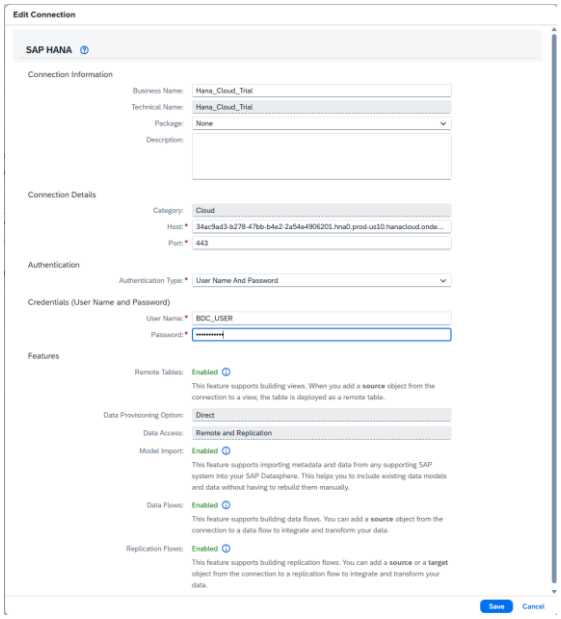
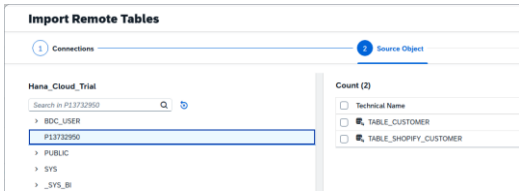
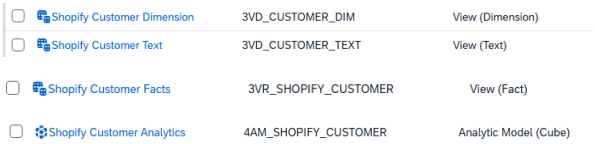
-	Deploy app in Cloud Foundry CAP app gets a permanent external URL that Datasphere can call, only after deployment.	
3.3.12	Create xs-security.json in the project folder root	
3.3.13	Update package.json	
3.3.14	Generate MTA descriptor: cap-dbfirst $ cds add mta	
3.3.15	Add auth resource to mta.yaml	

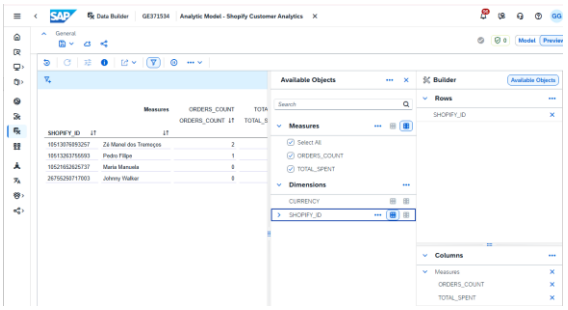
	Try to deploy: cap-dbfirst $ npm install @sap/xssec cap-dbfirst $npm install cap-dbfirst $ mbt build cap-dbfirst $ cf deploy mta_archives/*.mtar	
3.3.16	Discover endpoint with 'cf apps' Test deployed app curl -X POST \ https://dff40b18trial-dev-cap-dbfirst-srv.cfapps.us10-001.hana.ondemand.com/odata/v4/integration/ImportShopifyCustomers	
3.3.17	The test should end with Unauthorized message, but this is correct and datasphere connection needs to authenticate.	

At this step CAP service is active and accessible via authentication from any location.

4.4 SAP BDC Datasphere –Build Semantic Layer

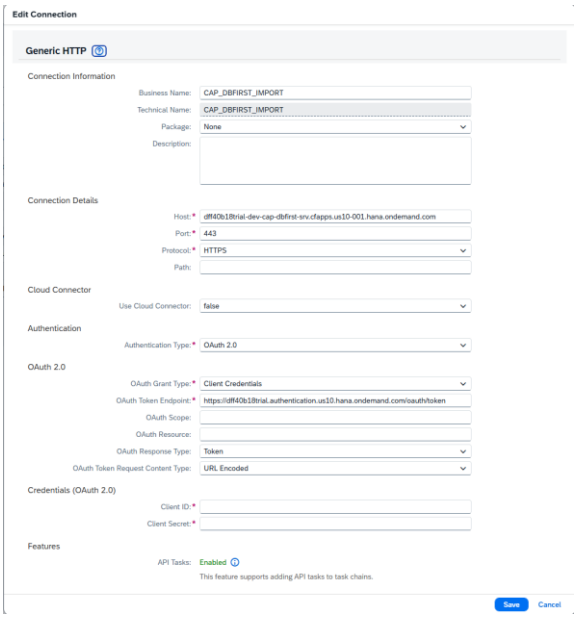
Step	Action	Screenshot
-	Create a trial SAP Datasphere account	
3.4.1	Start SAP Datasphere platform free trial	
3.4.2	You access a free trial account, from where you can model your connections and analytical services.	

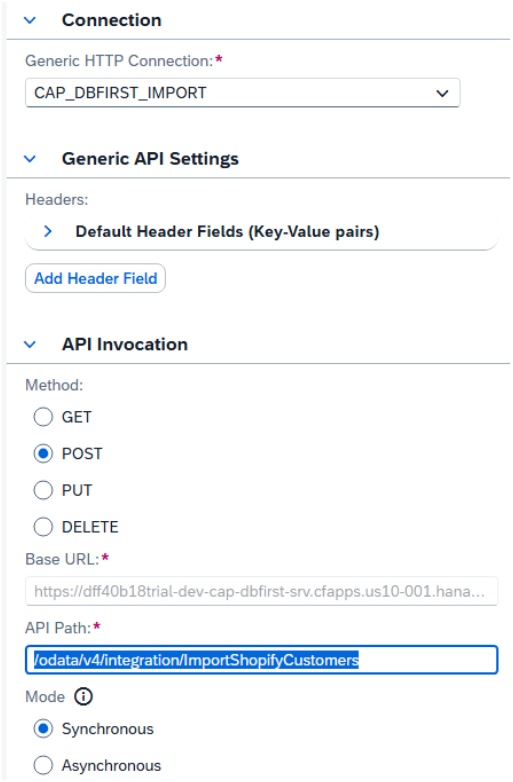
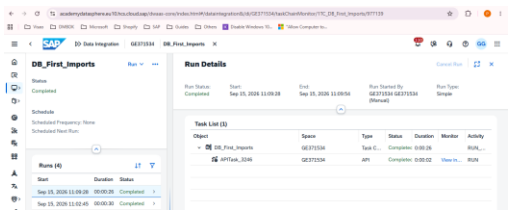
3.4.3	(you are interested in remote table TABLE_SHOPIFY_CUSTOMER) Create Hana Cloud Connection to HANA Cloud. User is BDC_USER created in Hana Cloud Host is HANA SQL end point, visible in SAP BTP hana configuration, noted in previous steps	
3.4.4	Go to Data Builder and create remote table.	
3.4.5	For this Proof of concept, remote table without data transfer is enough, no replication flow required.	
3.4.6	Create the following artifacts that are common in Data Builder Layer arquitecture: 1RT remote table 3VD Virtual Dimensions 3VR Virtual Relational (Fact) table 4AM Analytical Model Note: the purpose of this PoC is data integration, not Datasphere modelling	

3.4.7	At this step you can already preview your data model data	

At this step, Datasphere implemented virtual data models.

But data orchestration is not implemented and data must be refreshed periodically in Hana Cloud.

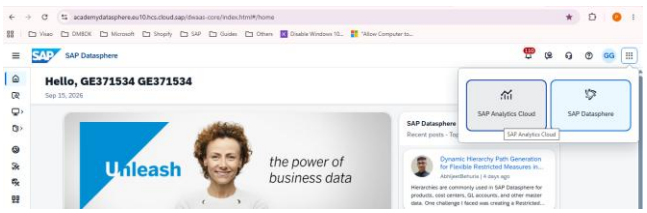
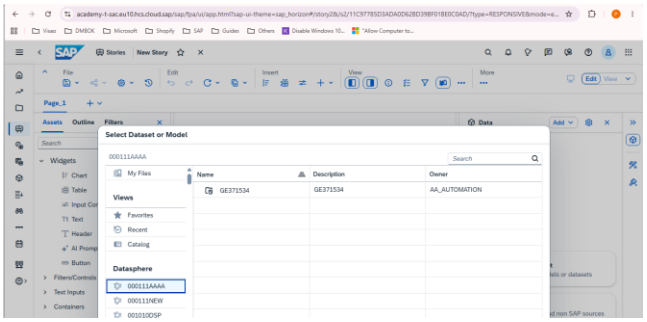
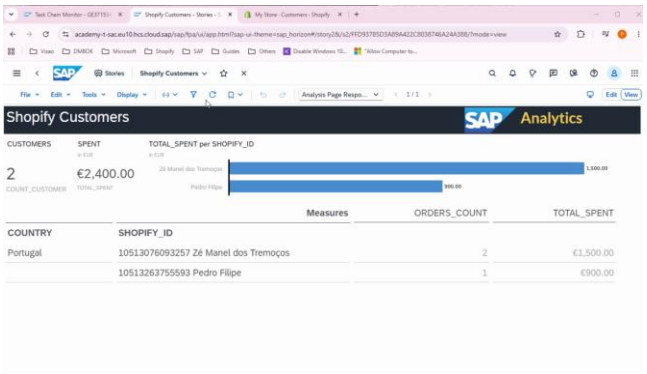
3.4.8	You still need to implement data orchestration to trigger data refresh from shopify to hana cloud. We already know the CAP app **public URL** dff40b18trial-dev-cap-dbfirst-srv.cfapps.us10-001.hana.ondemand.com Create HTTP Connection Host: dff40b18trial-dev-cap-dbfirst-srv.cfapps.us10-001.hana.ondemand.com Port: 443 Protocol: HTTPS Use Cloud Connector: False Grant Type: Client Credentials OAuth Token Endpoint: https://dff40b18trial.authentication.us10.hana.ondemand.com/oauth/token Client ID: use terminal command “cf service-key cap-dbfirst-auth datasphere-key” Secret: use terminal command “cf service-key cap-dbfirst-auth datasphere-key”	

3.4.9	Go to Data Builder and create New Task Chain Drag and Drop API Task. Use created HTTP connection to call service /odata/v4/integration/ImportShopifyCustomers	
3.4.10	Run the task. Use Monitoring -> Data Integration to check the results You can schedule data refresh periodically.	

At this step, Datasphere Orchestrates the data extractions from Shopify to Hana Cloud and defined analytical model for data consumption in SAP Analytics Cloud.

3.5 SAP Analytics Cloud – Reporting Layer

Step	Action	Screenshot
-	SAP Analytics Cloud data consumption	

3.5.1	From Datasphere, open SAP Analytics Cloud (SAC)	
3.5.2	In SAC create a new story that consumes data from your Datasphere space. If you use a trial account, don't be confuse by all the spaces available on the left. Choose any and you will see yours on the right. Select your Analytical model and create your story	
3.5.3	Story will display real time data from hana cloud persistent layer.	

At this step, you finish the end to end proof of concept.

Execution flow:

- ➔ Datasphere starts the import scheduling new task chain
- ➔ CAP calls Shopify and load data in Hana Cloud
- ➔ Datasphere model consumes Hana table and feeds SAC story

4. Lessons learned

Conceptually, this proof of concept is similar to custom ABAP development that called external APIs and wrote application data to SAP tables: the implementation is possible and useful, but the developer team owns the complete operational lifecycle.

Use this pattern for proofs of concept, small or specialized APIs, controlled data volumes, and cases where no suitable standard connector or managed integration capability is available. For production-scale integration, first evaluate native SAP Datasphere capabilities (Note 3696110) and the broader SAP Business Data Cloud architecture for replication, federation, batch and streaming scenarios. Managed integration approaches are generally better suited to high volumes, complex dependencies, lineage, governance and long-term lifecycle management. Standard governed and managed integration capabilities should remain the preferred choice whenever they meet the requirement.