

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	3
Symptom.....	3
Proof of concept objective	3
Future SAP BDC releases (July 2026 news)	3
2. Proof of Concept architecture	4
3. Solution setup	5
3.1 External System - Shopify.....	5
3.2 SAP BTP Hana - database first	7
3.3 SAP BTP CAP - integration service.....	10
4.4 SAP BDC Datasphere –Build Semantic Layer.....	16
3.5 SAP Analytics Cloud – Reporting Layer.....	19
4. Lessons learned	21

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 with 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

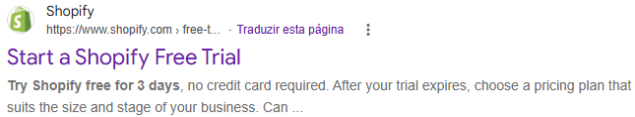
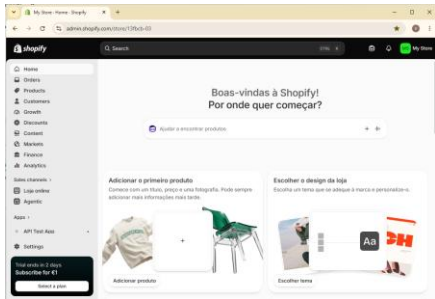
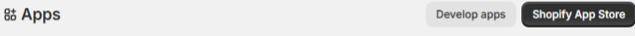
Shopify	CAP service	HANA Cloud	Datasphere	Analytics Cloud
---------	-------------	------------	------------	-----------------

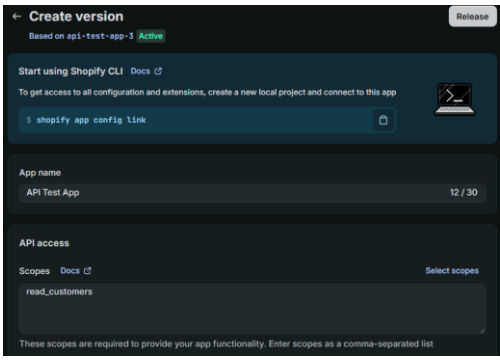
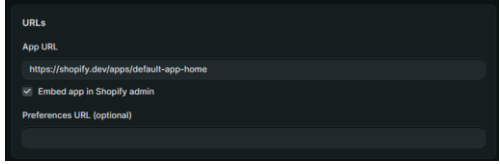
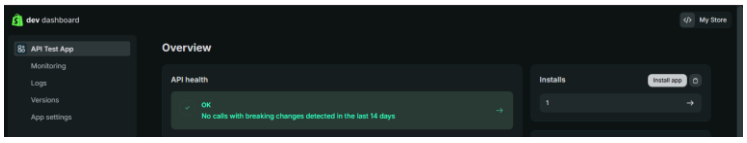
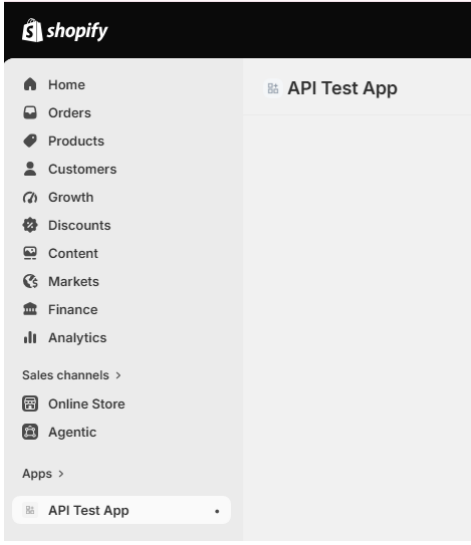
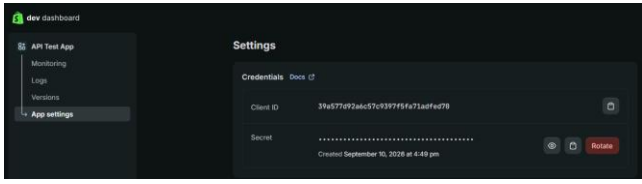
Development steps:

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

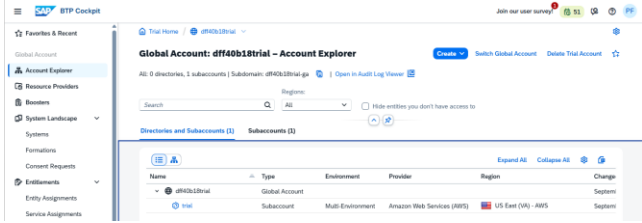
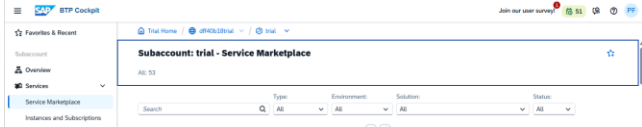
3. Solution setup

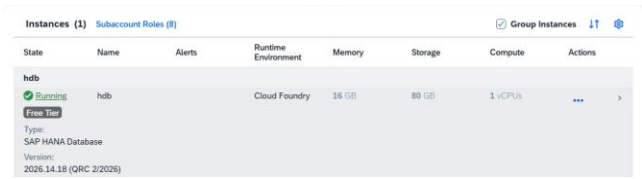
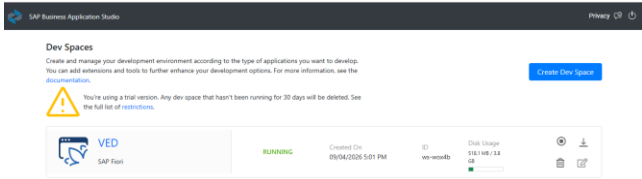
3.1 External System - Shopify

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	 Shopify https://www.shopify.com › free-t... › Traduzir esta página Start a Shopify Free Trial Try Shopify free for 3 days, no credit card required. After your trial expires, choose a pricing plan that suits the size and stage of your business. Can ...
3.1.2	Inside your store, create your first “Customers” master data records	 Boas-vindas à Shopify! Por onde quer começar? Adicionar o primeiro produto Escolher o design da loja
-	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”.	 Apps Develop apps Shopify App Store

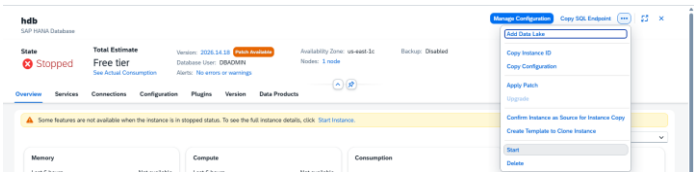
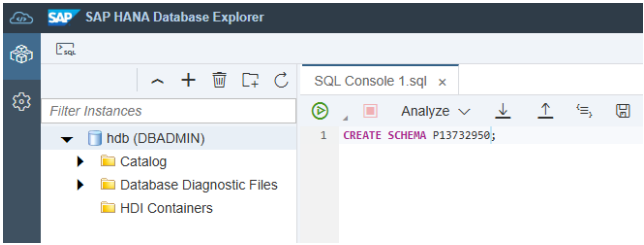
3.1.4	You jump into “Dev dashboard” area and you can create your new App. In API Access, add “read_customer”	
3.1.5	In App URL, enter “https://shopify.dev/apps/default-app-home”	
3.1.6	Now, you need to install the app in your trial store. Choose “Install app”	
3.1.7	After installation, App will become 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” authentication.	

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	 SAP https://www.sap.com/products · Traduzir esta página · SAP Business Technology Platform Free Trial How to start an SAP BTP trial? · Enter your contact details. · Check your email and click the link to activate your account. · Create a password and acknowledge ...
3.2.2	You access a free trial subaccount, from where you can subscribe applications and instances	 The screenshot shows the SAP BTP Cockpit 'Account Explorer' for a Global Account. It lists subaccounts, including a 'trial' subaccount of type 'Subaccount'.
-	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	 The screenshot shows the SAP BTP Cockpit 'Service Marketplace' for a trial subaccount. It displays a table with columns for Search, Type, Environment, Solution, and Status.

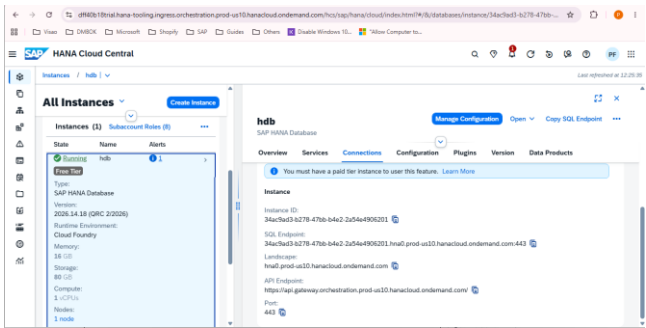
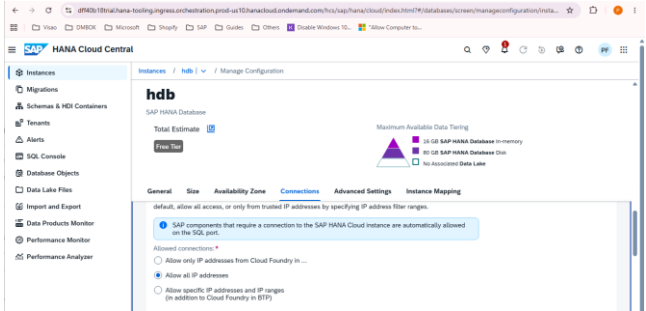
3.2.4	Create an “hdb” Hana Database	
3.2.5	Create a “DEV” SAP Business Application Studio space.	
-	The following steps are executed in SAP Business Application Studio Terminal, to verify that you can connect to Shopify API from this SAP BTP platform.	
3.2.6	Use the “store id”, “app client ID” and “API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

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;

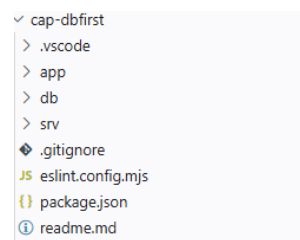
At this step the SAP HANA Cloud schema and tables already exist.

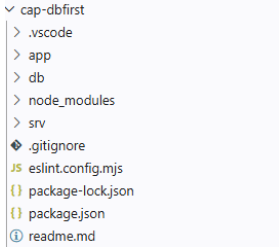
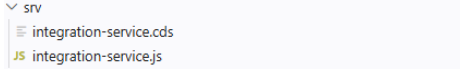
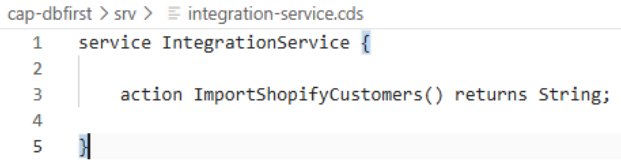
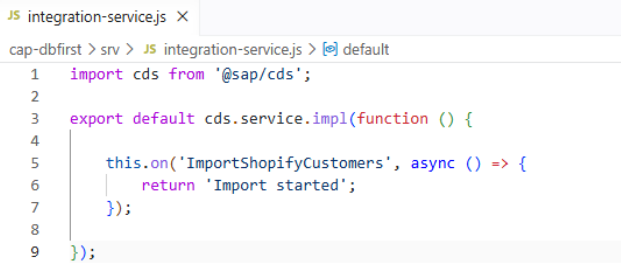
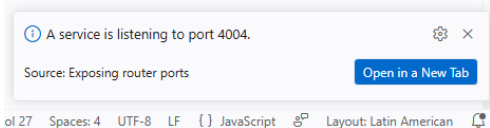
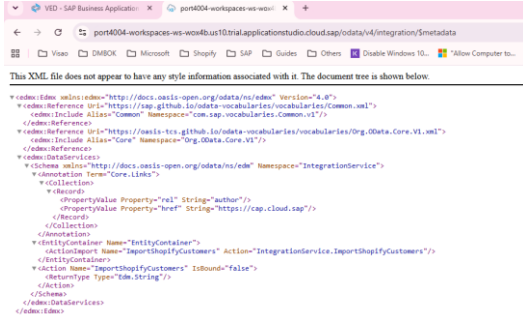
CAP service will not own or deploy tables through HDI. Instead, the service will connect to the existing database with a dedicated technical user and execute SQL against designated integration schema.

-	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.	

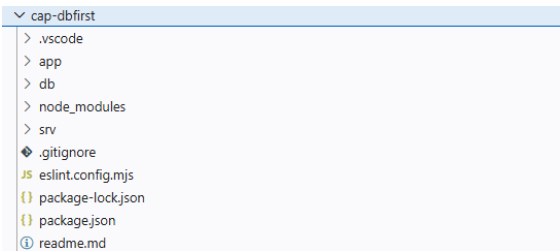
At this step the SAP HANA Cloud schema is available to be consumed in Datasphere virtual models.

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. <code>cds init cap-dbfirst --nodejs</code> <code>cd cap-dbfirst</code>	

3.3.2	Install dependencies <pre>npm install npm install @cap-js/hana npm install @sap/hana-client</pre>	
3.3.3	Create 'integration-service' files under "srv" folder 	
-	Create a minimal action service	
3.3.4	In "srv/integration-service.cds" declare the public OData service contract and the ImportShopifyCustomers action.	
3.3.5	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.	
3.3.6	Start the service using terminal command cds watch	
3.3.7	Follow localhost link (available in terminal output) to confirm 'Integration' service is active and 'ImportShopifyCustomers' action metadata exists <pre>impl: 'srv/integration-service.js' } [cds] - server listening on { url: 'http://localhost:4004' } [cds] - server v10.1.0 launched in 280 ms [cds] - [terminate with ^C] [odata] - GET /odata/v4/integration/\$metadata</pre>	

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 <pre>npm install @cap-js/hana npm install @sap/hana-client npm install axios</pre>	
3.3.9	Make sure package.json contains a HANA configuration.	 <pre> 1 { 2 "name": "cap-dbfirst", 3 "version": "1.0.0", 4 "type": "module", 5 "dependencies": { 6 "@cap-js/hana": "^3.1.0", 7 "@sap/cds": "^10", 8 "@sap/hana-client": "^2.29.27", 9 "@sap/xssec": "^4.15.0", 10 "axios": "^1.20.0" 11 }, 12 "devDependencies": { 13 "@cap-js/sqlite": "^3", 14 "@sap/cds-dk": "^10" 15 }, 16 "scripts": { 17 "start": "cds-serve" 18 }, 19 "cds": { 20 "requires": { 21 "[production]": { 22 "auth": "xsuaa" 23 }, 24 "queue": false, 25 "db": { 26 "kind": "hana" 27 } 28 }, 29 "private": true 30 } 31 } 32 </pre>

3.3.10

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

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

1 import cds from '@sap/cds';
2 import hana from '@sap/hana-client';
3 import axios from 'axios';
4
5 function sqlSafe(value) {
6   return String(value || '').replace(/'/g, "''");
7 }
8
9 function connectToHana() {
10
11   const conn = hana.createConnection()
12
13   conn.connect({
14     serverNode: '34ac9ad3-b278-47bb-b4e2-2a54e4906201.hna0.prod-us10.hanacloud.ondemand.com:443',
15     uid: 'BDC_USER',
16     pwd: '...',
17     encrypt: true
18   })
19
20   return conn
21 }
22
23
24 export default cds.service.impl(function () {
25
26   this.on('ImportShopifyCustomers', async () => {
27
28     const response = await axios.get(
29       // 'https://$({cds.env.shopify.store}).myshopify.com/admin/api/2024-04/customers.json',
30       'https://myshopify.com/admin/api/2024-04/customers.json',
31     )
32     {
33       headers: {
34         // 'X-Shopify-Access-Token': cds.env.shopify.token
35         'X-Shopify-Access-Token': 'shpat_...'
36       }
37     }
38
39     const customers = response.data.customers;
40
41     // const db = await cds.connect.to('db');
42     const conn = connectToHana()
43
44   }
45 })
```

```

43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
try {
    for (const customer of customers) {
        const country =
            customer.default_address?.country || '';
        const countryCode =
            customer.default_address?.country_code || '';
        const marketingState =
            customer.email_marketing_consent?.state || '';
        const sql = `
            MERGE INTO P13732958.TABLE_SHOPIFY_CUSTOMER T
            USING (
                SELECT
                    ${customer.id} AS SHOPIFY_ID
                FROM DUMMY
            ) S
            ON (T.SHOPIFY_ID = S.SHOPIFY_ID)

            WHEN MATCHED THEN
            UPDATE SET
                EMAIL = '${sqlSafe(customer.email)}',
                FIRST_NAME = '${sqlSafe(customer.first_name)}',
                LAST_NAME = '${sqlSafe(customer.last_name)}',
                PHONE = '${sqlSafe(customer.phone)}',
                CUSTOMER_STATE = '${sqlSafe(customer.state)}',
                VERIFIED_EMAIL = ${customer.verified_email ? 'TRUE' : 'FALSE'},
                CURRENCY = '${sqlSafe(customer.currency)}',
                ORDERS_COUNT = ${customer.orders_count || 0},
                TOTAL_SPENT = ${customer.total_spent || 0},
                COUNTRY = '${sqlSafe(country)}',
                COUNTRY_CODE = '${sqlSafe(countryCode)}',
                MARKETING_STATE = '${sqlSafe(marketingState)}',
                CREATED_AT = ${customer.created_at},
                UPDATED_AT = ${customer.updated_at}'

            WHEN NOT MATCHED THEN
            INSERT (
                SHOPIFY_ID,
                EMAIL,
                FIRST_NAME,
                LAST_NAME,
                PHONE,
                CUSTOMER_STATE,
                VERIFIED_EMAIL,
                CURRENCY,
                ORDERS_COUNT,
                TOTAL_SPENT,
                COUNTRY,
                COUNTRY_CODE,
                MARKETING_STATE,
                CREATED_AT,
                UPDATED_AT
            )
            VALUES (
                ${customer.id},
                '${sqlSafe(customer.email)}',
                '${sqlSafe(customer.first_name)}',
                '${sqlSafe(customer.last_name)}',
                '${sqlSafe(customer.phone)}',
                '${sqlSafe(customer.state)}',
                ${customer.verified_email ? 'TRUE' : 'FALSE'},
                '${sqlSafe(customer.currency)}',
                ${customer.orders_count || 0},
                ${customer.total_spent || 0},
                '${sqlSafe(country)}',
                '${sqlSafe(countryCode)}',
                '${sqlSafe(marketingState)}',
                ${customer.created_at},
                ${customer.updated_at}'
            )
        `;
        // await db.run(sql);
        conn.exec(sql)

        console.log(
            'Processing Shopify customer',
            customer.id
        );
    }
}

return customers.length;
} catch (e) {
    console.error(
        'ImportShopifyCustomers failed',
        e
    );
    throw e;
}

finally {
    conn.disconnect()
}
});
});

```

- 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

SQL Console 1 sql

1 select * from P13732958.TABLE_SHOPIFY_CUSTOMER

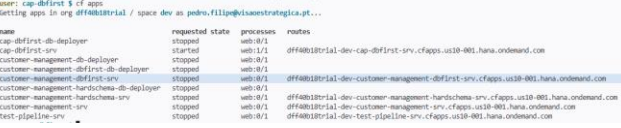
2

Result Messages History

Rows (1)	SHOPIFY_ID	EMAIL	FIRST_NAME	LAST_NAME	PHONE	CUSTOMER_STATE	VERIFIED_EMAIL	CURRENCY
1	26750290717003	pe@gmail.com	Johnny	Walker		disabled	true	EUR

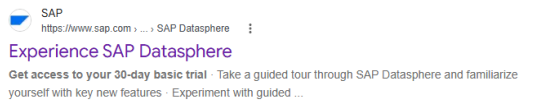
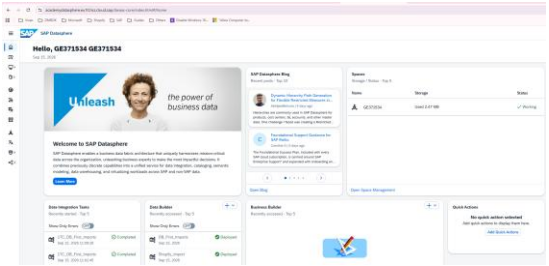
At this step a new CAP service is extract data from Shopify and load it into Hana Cloud. The service is active but can only be called locally inside Business Application Studio (BAS) terminal.

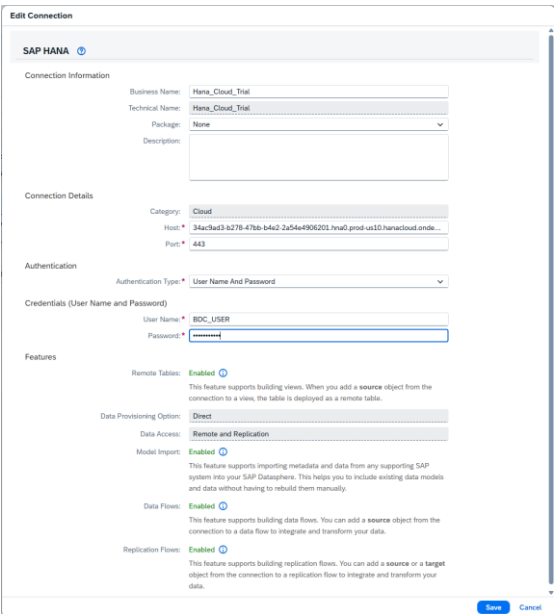
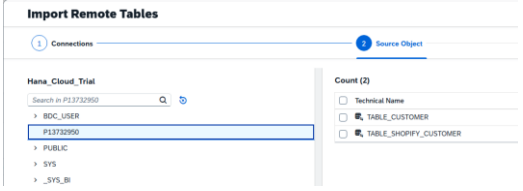
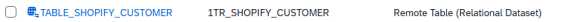
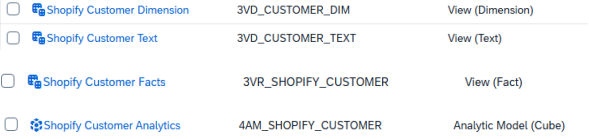
-	Deploy app in Cloud Foundry CAP app gets a permanent external URL that Datasphere can call, only after deployment.	
3.3.12	Create xs-security.json in the project folder root	<pre> () xs-security.json X cap-dbfirst > {} xs-security.json > ... 1 { 2 "xsappname": "cap-dbfirst", 3 "tenant-mode": "dedicated", 4 "scopes": [], 5 "attributes": [], 6 "role-templates": [] 7 } </pre>
3.3.13	Update package.json	<pre> () package.json X cap-dbfirst > {} package.json > ... 12 "devDependencies": { 13 "cds": "latest", 14 "cds-plugin": "latest", 15 "cds-srv": "latest", 16 "cds-odata": "latest", 17 "cds-odata-plugin": "latest", 18 "cds-odata-service": "latest", 19 "cds-odata-plugin": "latest", 20 "cds-odata-service": "latest", 21 "cds-odata-plugin": "latest", 22 "cds-odata-service": "latest", 23 "cds-odata-plugin": "latest", 24 "cds-odata-service": "latest", 25 "cds-odata-plugin": "latest", 26 "cds-odata-service": "latest", 27 "cds-odata-plugin": "latest", 28 "cds-odata-service": "latest", 29 "cds-odata-plugin": "latest", 30 "cds-odata-service": "latest", 31 "cds-odata-plugin": "latest", 32 "cds-odata-service": "latest" 33 } </pre>
3.3.14	Generate MTA descriptor: cap-dbfirst \$ cds add mta	<pre> cap-dbfirst ├── .vscode ├── app ├── db ├── node_modules ├── srv ├── cds-private.json ├── .gitignore ├── .cdsint.config.mjs ├── mta.yaml ├── package-lock.json ├── package.json └── README.md </pre>
3.3.15	Add auth resource to mta.yaml	<pre> () mta.yaml X cap-dbfirst > {} mta.yaml > {} resources > {} 1 > {} parameters > {} config > {} oauth2-configuration > {} credential-types 14 modules: 15 - name: cap-dbfirst-srv 16 srv-url: \${default-url} 17 requires: 18 - name: cap-dbfirst-db 19 - name: cap-dbfirst-auth 20 - name: cap-dbfirst-db-deployer 21 path: gen/db 22 parameters: 23 buildpack: nodejs_buildpack 24 requires: 25 - name: cap-dbfirst-db 26 resources: 27 - name: cap-dbfirst-db 28 type: com.sap.xs.hdi-container 29 parameters: 30 service: hana 31 service-plan: hdi-shared 32 - name: cap-dbfirst-auth 33 type: org.cloudfoundry.managed-service 34 parameters: 35 service: xsuaa 36 service-plan: application 37 path: ./xs-security.json 38 config: 39 xsappname: cap-dbfirst-\${org}-\${space} 40 tenant-mode: dedicated 41 oauth2-configuration: 42 credential-types: 43 - binding-secret 44 - xsuaa </pre>

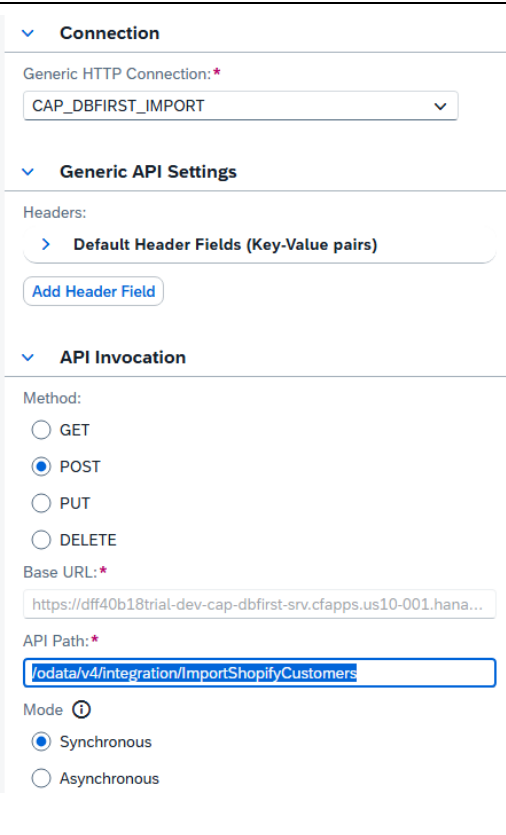
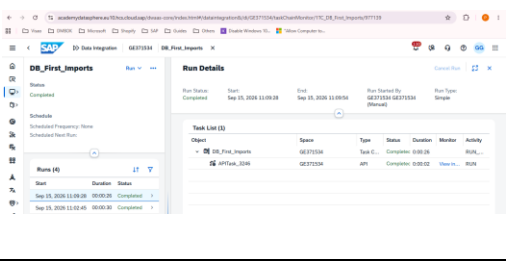
	Try to deploy: <pre>cap-dbfirst \$ npm install @sap/xssec cap-dbfirst \$ npm install cap-dbfirst \$ mbt build cap-dbfirst \$ cf deploy mta_archives/*.mtar</pre>	 <pre>user: cap-dbfirst \$ cf deploy mta_archives/*.mtar Application "cap-dbfirst-srv" staged Starting application "cap-dbfirst-srv"... Application "cap-dbfirst-srv" started and available at "dff4b18trial-dev-cap-dbfirst-srv.cfapps.us10-001.hana.ondemand.com" Skipping deletion of services, because the command line option "--delete-services" is not specified. Process finished. Use "cf dmcli -i 7ceb9d81-b089-11f1-9486-eeeeb031008" to download the logs of the process. user: cap-dbfirst \$</pre>
3.3.16	Discover endpoint with 'cf apps' Test deployed app <pre>curl -X POST \ https://dff4b18trial-dev-cap-dbfirst-srv.cfapps.us10-001.hana.ondemand.com/odata/v4/integration/ImportShopifyCustomers</pre>	 <pre>user: cap-dbfirst \$ cf apps Getting apps in org dff4b18trial / space dev as pedro.filipe@visiconstrategia.pt... name requested state processes routes cap-dbfirst-srv stopped web-0/1 dff4b18trial-dev-cap-dbfirst-srv.cfapps.us10-001.hana.ondemand.com cap-dbfirst-srv started web-1/1 dff4b18trial-dev-cap-dbfirst-srv.cfapps.us10-001.hana.ondemand.com customer-management-db-deployer stopped web-0/1 customer-management-dbfirst-db-deployer stopped web-0/1 customer-management-dbfirst-srv stopped web-0/1 dff4b18trial-dev-customer-management-dbfirst-srv.cfapps.us10-001.hana.ondemand.com customer-management-hardschema-db-deployer stopped web-0/1 customer-management-hardschema-srv stopped web-0/1 dff4b18trial-dev-customer-management-hardschema-srv.cfapps.us10-001.hana.ondemand.com customer-management-srv stopped web-0/1 dff4b18trial-dev-customer-management-srv.cfapps.us10-001.hana.ondemand.com test-pipeline-srv stopped web-0/1 dff4b18trial-dev-test-pipeline-srv.cfapps.us10-001.hana.ondemand.com</pre>
3.3.17	The test should end with Unauthorized message, but this is correct and datasphere connection needs to authenticate.	 <pre>user: cap-dbfirst \$ curl -X POST \ https://dff4b18trial-dev-cap-dbfirst-srv.cfapps.us10-001.hana.ondemand.com/odata/v4/integration/ImportShopifyCustomers Unauthorizeduser: cap-dbfirst \$</pre>

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.	

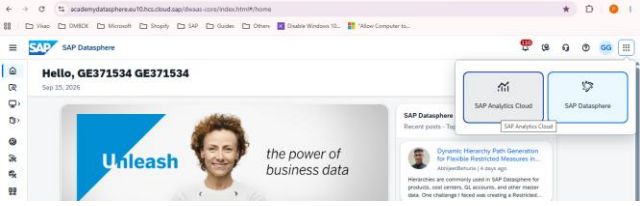
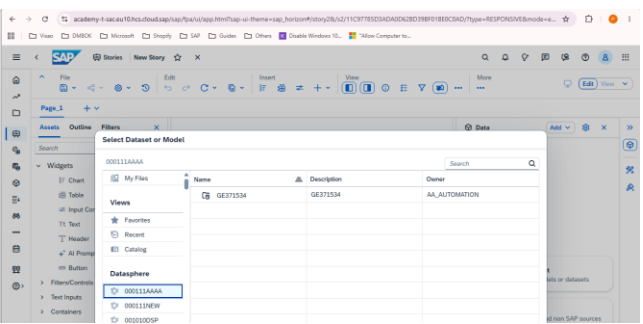
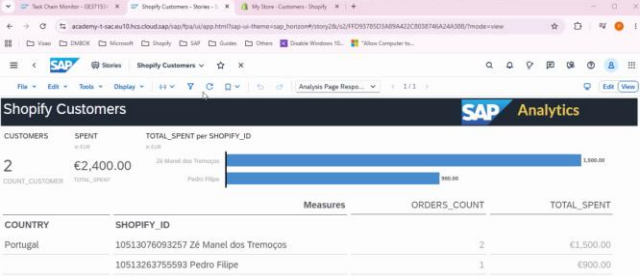
3.4.3	(you are interested in remote table TABLE_SHOPIFY_CUSTOMER) Create Hana Cloud Connection to HANA Cloud. User is BDC_USER created in Hana Cloud Host is HANA SQL end point, visible in SAP BTP hana configuration, noted in previous steps	
3.4.4	Go to Data Builder and create remote table.	
3.4.5	For 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 architecture: 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	

3.4.9	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.