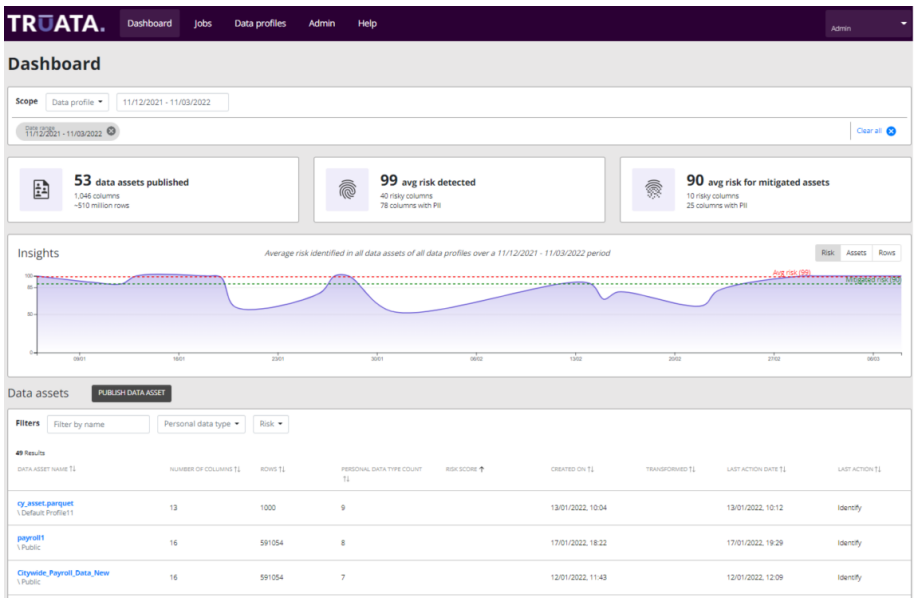
Truata Calibrate

TRŪATA.

Installation and Configuration Guide

Calibrate V2.2



info@truata.com
truata.com

Contents

Change History

Version	Release Date	Author	Change reason
0.1		Truata	Draft version
0.2	24-March-2021	Truata	Reviewed and updated
1.0	06-May-2021	Truata	Installation and configuration update. – Release version
1.1	28-May-2021	Truata	Updated for Azure Marketplace
2.0	12-Aug-2021	Truata	Updated for Calibrate Release 2.0
2.1 Draft	01-Nov-2021	Truata	Updated for Calibrate Release 2.1
2.1	12-Nov-2021	Truata	Minor update to include license Key related config
2.2	03-May-2022	Truata	Version 2.2 update

1 Introduction

This Installation guide is intended for IT users responsible for provisioning and deploying the Calibrate runtime environment.

Calibrate is designed to run on Azure.

***Note**: This document is for guidance on the Calibrate installation process. Deployment of prerequisite components is not in scope of this document.*

1.1 Deployment

A high-level view of the deployment of Calibrate is depicted below:

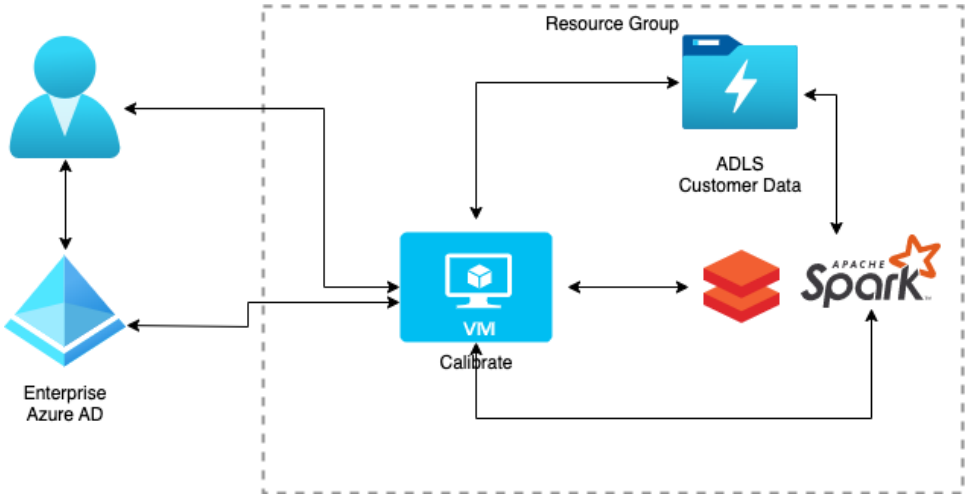


Figure 1-1: High-level view of the deployment of Calibrate.

2 Prerequisites

Truata Calibrate is intended to run in your Azure environment. No part of Calibrate shall be exposed to the Internet without proper protection by firewalls, VPNs and other strong security measures suitable to prevent unintended access and to protect your data assets.

To deploy and run Calibrate, the following are required to be provisioned on your Azure tenancy.

- Data Lake storage (ADLS Gen 2 or Blob Storage)
- Databricks
- Virtual Machine (VM)

2.1 System requirements

Calibrate is a containerized application that will run on a Linux OS.

Virtual Machine (VM) specifications:

- OS: Linux *
- Minimum RAM of 8GB
- Minimum disk space of 32GB

VM should be pre-installed with:

- Docker (version 19 or newer)
- Docker-compose (version 1.27.4 or newer)
- Jq, bash, curl
- System user that can run docker (user added to docker group)

VM should be able to access following components:

- Data lake (ADLS Gen 2 or Blob Storage)
- Databricks

Service principles must be provisioned for Calibrate to securely access these services.

Data processing functionality is provided using applications running in an Apache Spark cluster provisioned through Databricks.

Note: *Application is tested with CentOS and RHEL. However, the application is built to run on any modern Linux distribution.*

2.2 Databricks requirements

2.2.1 High-level guidance on cluster size for Databricks

The following is a guideline for the sizing of the Databricks cluster depending on the input file size and estimated execution time. Selected cluster size is necessary in the Databricks installation step below.

The size of the job cluster depends on size and complexity of data, the main factors are:

- Length – row count
- Breadth – column count
- Number of Categorical Columns (for Calibrate Fingerprint)

2.2.2 Recommendation on Databricks cluster Specifications

Following are different cluster sizes categorized into small, medium, and large clusters.

Small:

- Worker Node: Standard_DS4_v2, 28.0 GB Memory, 8 Cores, 1.5 DBU,
- Number of workers: 5
- Driver node: Standard_DS5_v2, 56.0 GB Memory, 16 Cores, 3.0 DBU

Medium:

- Worker Node: Standard_DS4_v2, 28.0 GB Memory, 8 Cores, 1.5 DBU,
- Number of workers: 10
- Driver node: Standard_DS5_v2, 56.0 GB Memory, 16 Cores, 3.0 DBU

Large:

- Worker Node: Standard_DS5_v2, 56.0 GB Memory, 16 Cores, 3.0 DBU,
- Number of workers: 40
- Driver node: Standard_DS5_v2, 56.0 GB Memory, 16 Cores, 3.0 DBU

Based on our benchmarking we've identified the following relationship between input data size and processing times:

Rows	Columns	Cluster size	Execution time*
100M	100	Large	9 hrs
10M	50	Medium	4 hrs

1M	25	Small	2 hrs

Figure 2-1: Identify benchmark.

Rows	Columns	Categorical cols	Cluster size	Execution time*
100M	100	70	Large	7 hrs
10M	50	25	Medium	3.5 hrs
1M	50	25	Small	2.5 hrs

Figure 2-2: Fingerprint benchmark.

Rows	Columns	Cluster size	Execution time*
1000M	100	Large	35 min
100M	100	Medium	20 min
10M	100	Small	5 min

Figure 2-3: Transform benchmark.

Note: *Transform actions are applied on 24 columns (4 columns for each transformation - Suppression, Masking, Rounding, Tokenization, FPE and Noise Addition)*

Notes: *These are approximated times and are purely recommendations based on performance tests on synthetic data generated by Truata. This may vary for other datasets. Cluster sizes can be selected separately for Identify, Fingerprint and Transform.*

** The above figures are for Apache Parquet formatted data assets.*

3 Calibrate installation

The Calibrate package is a TAR file consisting of:

- Application containers (archived)
- Configuration templates and installation scripts

Please extract the TAR into a directory on the VM. We will refer to this directory as ***<calibrate_dir>***

3.1.1 Configuring dependant services

Calibrate will require the following service principals:

- For calibrate UI communicating to storage, referred as "**calibrate-storage-sp**", add Storage IAM as "Storage blob data contributor" (can be set per container or per storage blob)
- For calibrate spark (databricks), referred as "**calibrate-databricks-sp**", add Databricks IAM as Contributor role
- Give "AzureDatabricks" "Key Vault Secrets User" permissions to the Key Vault in IAM RBAC Assignments

Also, Calibrate will require the Databricks pre-configuration as follows:

- Create a secret scope in Databricks - https://docs.microsoft.com/en-us/azure/databricks/security/secrets/secret-scopes#--create-an-azure-key-vault-backed-secret-scope
  This scope is used for Databricks to be able to access Azure KV – the name of it should have format of "KV-<vault name.
- Create Databricks access token and populate in KV - https://docs.microsoft.com/en-us/azure/databricks/dev-tools/api/latest/authentication#--generate-a-personal-access-token
  This Token is saved in Azure KV (next section) as "**calibrate-databricks-access-key**" and lifetime should be short (used during installation / update).

3.2 Installing Calibrate

3.2.1 Import containers (non marketplace install)

If you installed calibrate using Azure Marketplace, please skip this step

Run following commands:

cd <calibrate_dir>/bootstrap

docker load calibrate_docker_images.tar

cd -

3.2.2 Bootstrap services

Run the following commands:

cd <calibrate_dir>*/bootstrap*

bash bootstrap.sh

cd -

3.3 Configuring Calibrate

Please edit <calibrate_dir>/calibrate.env as follows:

```
LOGGING_LEVEL_ROOT=info
LOGGING_LEVEL_COM_TRUATA=INFO
DATABRICKS_URL=https://<DB id>.azuredatabricks.net
AZURE_KEYVAULT_URI=https://<azure kv name>.vault.azure.net/
AZURE_KEYVAULT_ENABLED=false

SPRING_DATA_MONGODB_DATABASE=<short name of cosmos DB in azure>
SPRING_H2_CONSOLE_ENABLED=false
SPRING_CLOUD_VAULT_TOKEN=<application token from bootstrap>
LICENSE_PUBLICKEY=<license signature key>
```

Please keep this file to be readable only by docker-compose user.

3.4 SSL/TLS certificate configuration

Calibrate can run with custom SSL/TLS certificates or with self-signed certificates.

3.4.1 Custom SSL/TLS setting

We recommend custom SSL/TLS certificates should be used for web traffic termination.

Place the certificates in separate directory, in PEM format (passphrase less).

Rename those certificates to calibrate.crt/.key

Edit <calibrate_dir>/docker-compose.yaml, in service calibrate-ui-frontend and fix path in volumes cert section (last line in below example):

```
version: "3.8"
services:
  calibrate-ui-frontend:
    image: truatadevcalibrateacr.azurecr.io/calibrate-ui:2.0.0
    ports:
      - "443:8443"
    volumes:
      - ${NGINX_CERT}:/cert/
(…)
```

Note: *If Calibrate is not using custom SSL/TLS, please remove bolded lines. Calibrate will start with self-signed cert to encrypt traffic.*

3.5 License configuration

Calibrate license file needs to be saved as <calibrate_dir>/calibrate-license/calibrate-uibe-license.txt

Please set permission on file and directory as follows:

chmod 701 <calibrate_dir>*/calibrate-license/*

chmod 644 <calibrate_dir>*/calibrate-license/calibrate-uibe-license.txt*

Calibrate license needs also corresponding key to be added, please add it into calibrate.env under LICENSE_PUBLICKEY variable name.

3.6 Databricks configuration

Starting in <calibrate_dir>/databricks/databricks_jobs/

Edit ./live-template/databricks_variables.config file:

SUBSCRIPTION_ID= <Azure subscription id>

RESOURCE_GROUP_NAME= <Azure resource group where Databricks reside >

BLOB_STORAGE_NAME= <Azure blob storage name >

HCV_URL=http://<IP of calibrate VM>:8200

KV_CLIENT_ID= <Azure Vault service principal ID>

KV_CLIENT_SECRET= <Azure Vault service principal secret >

KV_NAME= KV-<Azure Vault name > DIRECTORY_ID= <Azure directory/tenant ID>

FLUENTD_IP= <IP of Calibrate VM>

DATABRICKS_NAME= <Azure Databricks service name>

DATABRICKS_URL= <Azure Databricks service URL> (e.g. https://<datbricks id>.azuredatabricks.net)

Run *./initialize_databricks_setup.sh live-template*

Following output should be produced (below one is example):

```
Generated MANAGEMENT_ACCESS_TOKEN to access Databricks API

Created Identify job payload by replacing with config values

{"job_id":6}Identify job created

Created Fingerprint job payload by replacing with config values

{"job_id":7}FINGERPRINT job created

Created Transform job payload by replacing with config values

{"job_id":8}TRANSFORM job created

{}

Init Script Deployed to DBFS

{}

Environment Config file Deployed to DBFS
```

Note: *If you need to re-run this script, please log on to Databricks and delete all jobs definition that were created by this script, before running it again.*

From <calibrate_dir>/databricks directory, run

bash db-jars.sh <AZURE_DATABRICKS_ID> <DATABRICKS_ACCESS_KEY>
where:

- <AZURE_DATABRICKS_ID> is numeric id from databricks url (in format as https://<datbricks id>.azuredatabricks.net) ex:- **adb-1393574775625349.9.azuredatabricks.net**
- <DATABRICKS_ACCESS_KEY> is shared access key stored in Azure KV under **calibrate-databricks-access-key**

The live-template/databricks_variables.config file can be removed as it's not needed for normal operation.

3.7 Starting Calibrate

All commands are to be run in <calibrate_dir>.

To start the Calibrate application run the following commands:

docker-compose -p calibrate -f ./docker-compose.yml up -d hashicorp-vault

bash bootstrap/HCV-unseal.sh

bash bootstrap/deploy.sh

Login to Hashicop vault https://<VM IP>:8201 (login key can be found in VM in home directory)

- Update the Vault with required secrets

bash bootstrap/deploy.sh

docker-compose -p calibrate -f ./docker-compose.yml up -d

***Note**: Calibrate services can take up to 2 minutes to fully initialize. First start can take bit longer as initial databases are created.*

3.8 Stopping Calibrate

To stop the Calibrate application user needs to run following command:

docker-compose -p calibrate -f ./docker-compose.yml down

3.9 Cleaning up Calibrate environment

To remove Calibrate application from the installed node, run the following commands:

docker-compose -p calibrate -f ./docker-compose.yml down -v

docker-compose -p calibrate -f ./docker-compose-bootstrap.yml down -v

docker system prune -af

***Note**: To remove Calibrate and all dependant services (aka full wipe/uninstall), please remove all resources created for this application.*

Installation Guide