TED-SWS Installation manual

Editors Dragos Paun <dragos.paun@meaningfy.ws>
Eugeniu Costetchi <eugen@meaningfy.ws>

Version 2.5.0

Date 20/02/2023

Contents

Glossary

The official AWS glossary is available here.

The official Archimate business layer glossary and conventions are found here.

Source code - the code that is in the github repository

Introduction

The TED Semantic Web Service (TED SWS) is a pipeline system that continuously converts the public procurement notices (in XML format) available on the TED Website into RDF format and publishes them into CELLAR. This is done so that the produced RDF notices are made available to the public through CELLAR's SPARQL endpoint.

Purpose of the document

The purpose of this document is to explain how to build and deploy the TED-SWS system in the AWS cloud. This document may be updated by the development team as the system evolves.

Intended audience

This document is intended for persons involved in the operation of services deployed in the AWS cloud. The reader should be versed in the basics of Podman, bash scripts, AWS CLI and ECS CLI.

Useful Resources:

https://podman.io/getting-started/

https://docs.amazonaws.cn/en_us/IAM/latest/UserGuide/introduction.html

https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_CLI_reference.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html

Running project infrastructure in AWS ECS

The infrastructure will need one cluster that will have all the services running in ECS inside. All task definitions for services will be in awswpc network mode and runned in Fargate mode.

Suggested name for cluster: **ted-rdf-conversion-pipeline**

Environment file

Environment files are designed to store secrets (like passwords) and other parameters of the application. These files will be pushed into the docker containers to be used by the applications. Before pushing, they need to be updated to reflect the correct values from the infrastructure.

The project does not impose a storage solution for this file (as long as it's secure) so it can be stored anywhere.

Create a *.env* file with variables defined in the following table.

Name	**Description**
_AIRFLOW_WWW_USER_PASSWORD	Airflow UI user password

_AIRFLOW_WWW_USER_USERNAME	Airflow UI user
AIRFLOW_GID	Airflow user permissions. This should be **50000**
AIRFLOW_UID	Airflow user permissions. This should be **50000**
_AIRFLOW_DB_UPGRADE	This is a flag for airflow db upgrade. This should be set with **true**
_AIRFLOW_WWW_USER_CREATE	This is a flag for airflow to create user. This should be set with **true**
AIRFLOW__API__AUTH_BACKEND	This should be **airflow.api.auth.backend.basic_auth**
AIRFLOW__CELERY__BROKER_URL	This is the connection to redis. This variable will use the endpoint for the ElastiCache (airflow-redis) redis cluster created. **redis://:@<primary_endpoint_for_redis cluster>:6379/0**
AIRFLOW__CORE__ENABLE_XCOM_PICKLING	This should be set to **true**
AIRFLOW__CELERY__RESULT_BACKEND	This is the celery connection to postgres. This variable will use the endpoint for the created RDS (airflow-rds) ,the initial created database in that rds,master password and master username **db+postgresql://masterusername:rootpassword@endpoint_for_RDS/initial_database_name**
AIRFLOW__CORE__DAGS_ARE_PAUSED_AT_CREATION	This should be set to **true**
AIRFLOW__CORE__EXECUTOR	This should be set with **CeleryExecutor**
AIRFLOW__CORE__LOAD_EXAMPLES	This should be set with **false**
AIRFLOW__CORE__SQL_ALCHEMY_CONN	This is the celery connection to postgres. This variable will use the endpoint for the created RDS (airflow-rds) ,the initial created database in that rds,master password and master username **postgresql+psycopg2://masterusername:rootpassword@endpoint_for_RDS/initial_database_name**
AIRFLOW_HOME	This should be set with **/opt/airflow**
PYTHONPATH	This should be set with **/opt/airflow**
RML_MAPPER_PATH	This should be set **/opt/airflow/.rmlmapper/rmlmapper.jar**
XML_PROCESSOR_PATH	This should be set **/opt/airflow/.saxon/saxon-he-10.6.jar**
LIMES_ALIGNMENT_PATH	This should be set **/opt/airflow/.limes/limes.jar**
IS_PRIME_ENV	This should be set to **true**
AIRFLOW__WEBSERVER__SECRET_KEY	Secret key of 16 random characters
AIRFLOW__CORE__PARALLELISM	This should be set to **256**

AIRFLOW__CORE__MAX_ACTIVE_TASKS_PER_DAG	This should be set to **256**
AIRFLOW__CORE__NON_POOLED_TASK_SLOT_COUNT	This should be set to **256**
AIRFLOW__SCHEDULER__PARSING_PROCESSES	This should be set to **8**
AIRFLOW__SCHEDULER__SCHEDULER_HEARTBEAT_SEC	This should be set to **1**
AIRFLOW__SCHEDULER__MAX_DAGRUNS_PER_LOOP_TO_SCHEDULE	This should be set to **128**
AIRFLOW__CELERY__WORKER_CONCURRENCY	This should be set to **24**
AIRFLOW__CORE__SQL_ALCHEMY_POOL_SIZE	This should be set to **512**
AIRFLOW__CORE__SQL_ALCHEMY_MAX_OVERFLOW	This should be set to **1024**
FUSEKI_ADMIN_PASSWORD	Fuseki admin password
ADMIN_PASSWORD	${FUSEKI_ADMIN_PASSWORD} This needs to have the same value as FUSEKI_ADMIN_PASSWORD variable above
FUSEKI_DATASET_1	Fuseki default dataset
FUSEKI_ADMIN_HOST	The host to the fuseki service
MB_DB_DBNAME	Name of the created initial database in the RDS (metabase-rds)
MB_DB_PORT	This should be **5432**
MB_DB_USER	master username for the RDS (metabase-rds)
MB_DB_PASS	master password for the RDS (metabase-rds)
MB_ENCRYPTION_SECRET_KEY	Encryption secret key (min 16 characters)
MB_DB_HOST	endpoint for the RDS (metabase-rds)
MB_DB_TYPE	This should be **postgres**
MONGO_INITDB_ROOT_PASSWORD	Master password for AWS DocumentDB
MONGO_INITDB_ROOT_USERNAME	Master username for AWS DocumentDB
MONGO_DB_AGGREGATES_DATABASE_NAME	AWS DocumentDB database name for notice aggregates
MONGO_DB_AUTH_URL	This will be AWS DocumentDB connection string Example: mongodb://<user>:<insertYourPassword>@ted-sws-documentdb.cluster-ccyy3f9gc.eu-west-1.docdb.amazonaws.com:27017/?replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false
MONGO_DB_LOGS_DATABASE_NAME	AWS DocumentDB logs database name
MONGO_DB_PORT	AWS DocumentDB port (27017)
ME_CONFIG_BASICAUTH_PASSWORD	Password for accessing mongo express UI

ME_CONFIG_BASICAUTH_USERNAME	User for accessing mongo express UI
ME_CONFIG_MONGODB_ADMINPASSWORD	Master password for AWS DocumentDB
ME_CONFIG_MONGODB_ADMINUSERNAME	Master username for AWS DocumentDB
ME_CONFIG_MONGODB_ENABLE_ADMIN	This should be **true**
ME_CONFIG_MONGODB_SERVER	AWS DocumentDB endpoint without ssl verification Example **ted-sws-documentdb.cluster-ccyjy39gc.eu-west-1.docdb.amazonaws.com:27017/?replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false**
ID_MANAGER_DEV_API_HOST	The host for the digest-api service that the mapping suites were developed. This should be **https://digest-api.ted-data.eu/**
ID_MANAGER_PROD_API_HOST	The host for the digest-api service
ID_MANAGER_API_PORT	This should be **8000**
CLI_LOGGER_CONFIG_HANDLERS	this should be **ConsoleHandler**
DAG_LOGGER_CONFIG_HANDLERS	This should be **MongoDBHandler,ConsoleHandler**
TED_API_URL	This should be **https://ted.europa.eu/api/v3.0/notices/search**
GITHUB_TED_SWS_ARTEFACTS_URL	GitHub URL for artefacts repository on GitHub **https://github.com/OP-TED/ted-rdf-mapping.git**
SFTP_PUBLISH_USER	User for EUSEND SFTP server
SFTP_PRIVATE_KEY_BASE64	Private key encoded in base64 format
SFTP_PUBLISH_PORT	port for EUSEND SFTP
SFTP_PUBLISH_PATH	Folder path allocated for this project in EUSEND SFTP. Example: **/upload/notices**
SFTP_PUBLISH_HOST	Host for the EUSEND SFTP server
S3_PUBLISH_HOST	Host of the S3 bucket. Example **s3.eu-west-1.amazonaws.com** the only thing that will be different is the region as it depends in what region the S3 bucket was created
S3_PUBLISH_PASSWORD	AWS secret key
S3_PUBLISH_USER	AWS Access key
S3_PUBLISH_REGION	Region of the S3
S3_PUBLISH_SECURE	This should be **1**
S3_PUBLISH_SSL_VERIFY	This should be **0**
S3_PUBLISH_NOTICE_RDF_BUCKET	Name of the S3 bucket for rdf notices
S3_PUBLISH_NOTICE_BUCKET	Name of the S3 bucket for METS packages
S3_PUBLISH_ENABLED	False by default

Building images

The project needs a total of 6 images that are divided between custom builds and already built images from docker hub. This process can be done automatically by using build-images-with-podman.sh script in the source code (infra/aws directory) or manually.

Method 1

1. In the source code, go to *infra/aws* folder
2. Run *build-images-with-podman.sh* script that will create all images needed by the project with podman.

Method 2

1. By using **podman pull** command pull the following images from Docker Hub:
   - docker.io/mongo-express:0.54.0 (this image will be used for mongo-express container)
   - docker.io/secoresearch/fuseki:4.5.0 (this image will be used for fuseki container)
   - docker.io/metabase/metabase:v0.44.6 (this image will be used for metabase container)
   - docker.io/atmoz/sftp:debian (this image will be used for SFTP server)
2. By using **podman build** command build the remaining images following the instructions below.

Airflow image

1. In the source code, copy ./requirements.txt to *infra/airflow* folder

2. Go to infra/airflow folder and use the Dockerfile with the podman build command to build the image

Digest-api image

1. In the source code, create project_requirements.txt in *infra/digest_api/digest_service* folder

2. Copy contents of requirements.txt file from the source code to the newly created file (project_requirements.txt) in *infra/digest_api/digest_service* folder

3. Copy ted_sws folder from the source code to the *infra/digest_api* folder

4. Go to infra/digest_api folder and use the Dockerfile with the podman build command to build the image

EFS volumes

The project will need 5 volumes to be created that will be attached to the containers in the task definitions. For the purpose of this document we will name them as follows:

- airflow-dags

- airflow-logs
- airflow-ted-sws
- fuseki-data

Volume	Container	Service	Estimated max size of volume
airflow-dags airflow-logs airflow-ted-sws	airflow-worker	Airflow	1GB 10GB 1GB
airflow-dags airflow-logs airflow-ted-sws	airflow-webserver	Airflow	1GB 10GB 1GB
airflow-dags airflow-logs airflow-ted-sws	airflow-scheduler	Airflow	1GB 10GB 1GB
airflow-dags airflow-logs airflow-ted-sws	airflow-trigger	Airflow	1GB 10GB 1GB
fuseki-data	fuseki	Fuseki	40GB

Note: The volume sizes should not be restricted to the estimated size but allowed to grow elastically.

AWS services

The project will need to have 4 of the offered services in AWS as follows

AWS service	Instance number	Notes
AWS DocumentDB	1	This will be the database for this project
RDS	2	This will correspond to two **Postgres databases**: ● one for the Airflow service and ● one for the Metabase service
Elastic Cache	1	This is going to be a Redis cluster used by the Airflow service
AWS S3	1	This will be used to store some of the transformed RDF notices

Containers and services

This project needs AWS managed services and custom project services deployed via ECS.

Services	Containers
AWS DocumentDB (named ted-sws-document-db)	

RDS (named airflow-rds)	
RDS (named metabase-rds)	
Elastic Cache (named airflow-redis)	
AWS S3	
SFTP	EUSEND SFTP server
Metabase	metabase
Fuseki	fuseki
Digest API	digest-api
Airflow	airflow-init, airflow-init-data, airflow-scheduler, airflow-trigger, airflow-webserver, airflow-flower
Airflow Worker	airflow-worker
Mongo Express	mongo-express

AWS DocumentDB service

Configuration

Config name	**Value**
Cluster identifier	ted-sws-document-db
Engine version	4.0.0
Instance class	See **estimated resource requirements** table below. db.r6g.2xlarge
Number of instances	1
TLS Enabled	false
Backup	weekly

The *master password*, *master username* and *connection string* to the cluster should be written in the *environment* file as specified above in this document.

Connection string example:

```
mongodb://<master_username>:<master_password>@ted-sws-documentdb.cluster-ccyj5sy3f9gc.eu-west-1.docdb.amazonaws.com:27017/?replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false
```

/!\ Note: It is very important to disabled the TLS.

Network requirements

The AWS DocumentDB cluster should be in the same VPC and subnets as the rest of the services and should be accessible on port **27017**.

RDS (airflow-rds) service

Configuration

Config name	**Value**
DB identifier	airflow-rds
Engine version	PostgreSQL 13.7
Instance class	See **estimated resource requirements** table below. db.m5.large
Number of instances	1
Initial database name	airflow
Master username	airflow
Port	5432
Backup	False

The *master password*, *master username*, *endpoint* and *initial database name values* should be written in the *environment* file as specified above in this document.

Connection string example:

```
db+postgresql://masterusername:rootpassword@endpoint/initial_database_name
```

Network requirements

The RDS service should be in the same VPC and subnets as the rest of the services and should be accessible on port 5432.

RDS (metabase-rds) service

Configuration

Config name	**Value**
DB identifier	metabase-rds

Engine version	PostgreSQL 13.7
Instance class	See **estimated resource requirements** table below. db.m5.large
Number of instances	1
Initial database name	metabase
Master username	metabase
Port	5432
Backup	weekly

The *master password, master username, endpoint* and *port string* to the cluster should be written in the *environment* file as specified above in this document.

Connection string example:

```
db+postgresql://masterusername:rootpassword@endpoint/initial_database_name
```

Network requirements

The RDS service should be in the same VPC and subnets as the rest of the services and should be accessible on port **5432**.

Elastic Cache (airflow-redis)

Configuration

Config name	**Value**
Cluster name	airflow-redis
Engine version	Redis 6.2.6
Node type	See **estimated resource requirements** table below.
Number of instances	1
Port	6379

The *primary endpoint* should be written in the *environment* file as specified above in this document.

Connection string example:

```
redis://:@airflow-redis-031.pgh.01.euw1.cache.amazonaws.com:6379/0
```

Network requirements

The Redis service should be in the same VPC and subnets as the rest of the services and should be accessible on port **6379**.

AWS S3

Configuration

The bucket **should be publicly available** and the *name* should be stored in the *environment* file as specified above in this document. This will be used to share some files with external partners.

/!\ Note: The name of this bucket shall be communicated to the OP project manager after the installation.

Digest-api service

Task definition

1. Create a container with digest-api built image
2. Load the .env file into the container

Network requirements

This service should be accessible by other services using DNS naming on port 8000. This will be specified in the .env file as follows:

ID_MANAGER_API_HOST	The host for the digest-api service Example: https://digest-api.domain/

CPU and Memory

See **estimated resource requirements** table below.

Fuseki service

Task definition

1. Create a container with fuseki built image.
2. Load the .env file into the container
3. Attach fuseki-data (EFS) to */fuseki-base/databases*
4. Attach fuseki-data (EFS) to */fuseki-base/configuration*

Network requirements

This service should be accessible by other services using *DNS naming* on port **3030** and be publicly available and accessible by a user in a browser. This will be specified in the .env file as follows:

FUSEKI_ADMIN_HOST	The host for the fuseki service Example: https://fuseki.domain/

CPU and Memory

See **estimated resource requirements** table below.

Metabase service

Task definition

1. Create a container with metabase built image.
2. Load the .env file into the container

Network requirements

This service should expose metabase container using *DNS naming* on port **3000** and be publicly available and accessible by a user in a browser. This service doesn't need to be accessible by other services.

CPU and Memory

See **estimated resource requirements** table below.

Airflow service

Prerequisites

1. Have a completed .env file with all variables

Task definition

The environment variables that all containers from this task definition will use is the finished environment file. This file should be referenced in the task definition to all airflow containers.

1. Create airflow-init container with airflow built image.
2. Attach airflow-dags volume to /opt/airflow/dags in airflow-init container
3. Attach airflow-logs volume to /opt/airflow/logs in airflow-init container
4. Attach airflow-ted-sws volume to /opt/airflow/ted_sws in airflow-init container
   The command that this container should execute at runtime is **version**

5. Create airflow-init-data container with airflow built image.
6. Attach airflow-dags volume to /opt/airflow/dags in airflow-init-data container

7. Attach airflow-logs volume to /opt/airflow/logs in airflow-init-data container
8. Attach airflow-ted-sws volume to /opt/airflow/ted_sws in airflow-init-data container
9. The command that this container should execute at runtime is

   ```
   /bin/bash -c "mkdir -p ./dags ./ted_sws ./temp &&
   rm -rf ./dags/* ./ted_sws/* ./temp/* && cd temp &&
   git clone --branch <release_tag_name_or_branch>
   https://github.com/OP-TED/ted-rdf-conversion-pipeline.git &&
   cp -r ted-rdf-conversion-pipeline/dags/* ../dags &&
   cp -r ted-rdf-conversion-pipeline/ted_sws/* ../ted_sws"
   ```

   Remember to specify the tag or branch name in the command in the placeholder (<release_tag_name_or_branch>).
10. Create airflow-scheduler container with airflow built image.The command that this container should execute at runtime is **scheduler**
11. Attach airflow-dags volume to /opt/airflow/dags in airflow-scheduler container
12. Attach airflow-logs volume to /opt/airflow/logs in airflow-scheduler container
13. Attach airflow-ted-sws volume to /opt/airflow/ted_sws in airflow-scheduler container
14. Create airflow-trigger volume container with airflow built image.The command that this container should execute at runtime is **triggerer**
15. Attach airflow-dags volume to /opt/airflow/dags in airflow-trigger container
16. Attach airflow-logs volume to /opt/airflow/logs in airflow-trigger container
17. Attach airflow-ted-sws volume to /opt/airflow/ted_sws in airflow-trigger container
18. Create airflow-webserver container with airflow built image.The command that this container should execute at runtime is **webserver**
19. Attach airflow-dags volume to /opt/airflow/dags in airflow-webserver container
20. Attach airflow-logs volume to /opt/airflow/logs in airflow-webserver container
21. Attach airflow-ted-sws volume to /opt/airflow/ted_sws in airflow-webserver container
22. Create airflow-flower container with airflow built image.The command that this container should execute at runtime is **celery flower**
23. Attach airflow-dags volume to /opt/airflow/dags in airflow-worker container
24. Attach airflow-logs volume to /opt/airflow/logs in airflow-worker container
25. Attach airflow-ted-sws volume to /opt/airflow/ted_sws in airflow-worker container

/!\ Note:

1. All containers from this task should be accessible to each other.
2. The first 2 containers that should start in this task should be airflow-init and airflow-init-data.
3. Airflow-scheduler should start only after airflow-init and airflow-init-data have finished with exit code 0
4. Airflow-flower should start only when airflow-scheduler has started
5. Airflow-trigger should start only when airflow-scheduler has started
6. Airflow-webserver should start only when airflow-scheduler has started

   The checking of the exiting with code 0 can be done by using startup dependency ordering (condition: "SUCCESS") option in the task definition.

Network requirements

This service should expose *airflow-webserver* container using DNS naming on port **8080**, expose *airflow-flower* container using DNS naming on port **5555** and be publicly available and accessible by a user in a browser. Also, all containers in this task should communicate with each other. This service doesn't need to be accessible by other services.

CPU and Memory

See **estimated resource requirements** table below.

Airflow worker service

Task definition

1. Create airflow-worker container with airflow built image.The command that this container should execute at runtime is **celery worker**
2. Load the .env file into the container
3. Attach airflow-dags volume to /opt/airflow/dags in airflow-worker container
4. Attach airflow-logs volume to /opt/airflow/logs in airflow-worker container
5. Attach airflow-ted-sws volume to /opt/airflow/ted_sws in airflow-worker container

Service requirements

This service will have the task defined above scaled to **four instances** so we will have available four workers for the airflow service.

Network requirements

This service should be allowed to communicate on port **8793** so that they will be auto discovered by other services.

CPU and Memory

See **estimated resource requirements** table below.

Mongo express service

Task definition

1. Create a container with mongo-express built image.
2. Load the .env file into the container

Network requirements

This service should expose *mongo-express* container using DNS naming on port **8081** and be publicly available and accessible by a user in a browser. This service doesn't need to be accessible by other services.

CPU and Memory

See **estimated resource requirements** table below.

Using EU Send SFTP service

To be able to connect to the EU Send SFTP server a user and a folder structure needs to be configured by the EU Send service management team. For a user to be configured a SSH key pair needs to be generated and the public key to be sent to the EU Send service management team.

After finishing the user configuration the EU Send service management team should send back the following details:

Values for	Comments
EU Send host	This value will be used for SFTP_PUBLISH_HOST in the .env file
EU Send port	This value will be used for SFTP_PUBLISH_PORT in the .env file
Eu Send User	This value will be used for SFTP_PUBLISH_USER in the .env file
Eu Send folder destination (final destination for METS packages)	This value will be used for SFTP_PUBLISH_PATH in the .env file

Generate a SSH key pair

To generate a key pair use the following command:

ssh-keygen -f *key-name*

The public key (*key-name.pub*) should be sent to the EU Send service management team to be able to configure the user.

Encode private key in base 64

The private key (*key-name*) file should be encoded in base 64 and the encoded value to be assigned to the SFTP_PRIVATE_KEY_BASE64 variable in the environment file.

To encode to base 64 the private key file use the following command:

```
openssl base64 -in <file name>
```

Copy the output generated by the command above and assign it to the environment variable SFTP_PRIVATE_KEY_BASE64

Logs

Each service should have a log group in **AWS Cloud Watch**, that will have logs for all the containers forming that service or for the used AWS service (i.e RDS, AWS Document DB)

Network requirements summary

Service	**Containers that should see each other**	**Exposed container and port**	**DNS record and naming**	**Available to other services (in the VPC)**	**Publicly available (OP enduser)**
AWS DocumentDB		27017	Optional	Mandatory	No
AWS S3			Optional	Mandatory	Mandatory
RDS (Airflow-rds)		5432	Optional	Mandatory	No
RDS (metabase -rds)		5432	Optional	Mandatory	No
Elastic Cache (airflow-redis)		6379	Optional	Mandatory	No
Digest-api		digest-api:8000	Mandatory	Mandatory	No
Fuseki		fuseki:3030	Mandatory	Mandatory	Mandatory
Metabase		metabase:3000	Mandatory	Optional	Mandatory
Airflow	airflow-scheduler, airflow-trigger, airflow-webserver, airflow-flower	airflow-webserver: 8080, airflow-flower:5555	Mandatory	Optional	Mandatory
Airflow-worker	airflow-worker	airflow-worker:8793	Optional	Mandatory	No
Mongo-express	mongo-express	mongo-express:8081	Mandatory	Optional	Mandatory

Estimated resource requirements summary

Service	**New adjusted specs (Doc V 2.0.2)**	**Original spec (Doc V 2.0.1)**
AWS DocumentDB	db.r6g.2xlarge CPU 8 RAM 64	db.r5.8xlarge CPU 32 RAM 256
RDS (Airflow-rds)		db.m5.large
RDS (metabase-rds)		db.m5.large
Elastic Cache (airflow-redis)	cache.m6g.large (6Gb High throughput network)	cache.r6g.large (13Gb High throughput network)
Digest-api	CPU 4 RAM 8	CPU 8 RAM 16
Fuseki	CPU 8 RAM 32	CPU 16 RAM 64

Metabase	CPU 4 RAM 16	CPU 8 RAM 32
Airflow	CPU 8 RAM 16	CPU 16 RAM 32
Airflow-worker		CPU 8 RAM 32
Mongo-express	CPU 2 RAM 4	CPU 4 RAM 16

Metabase setup

Prerequisites

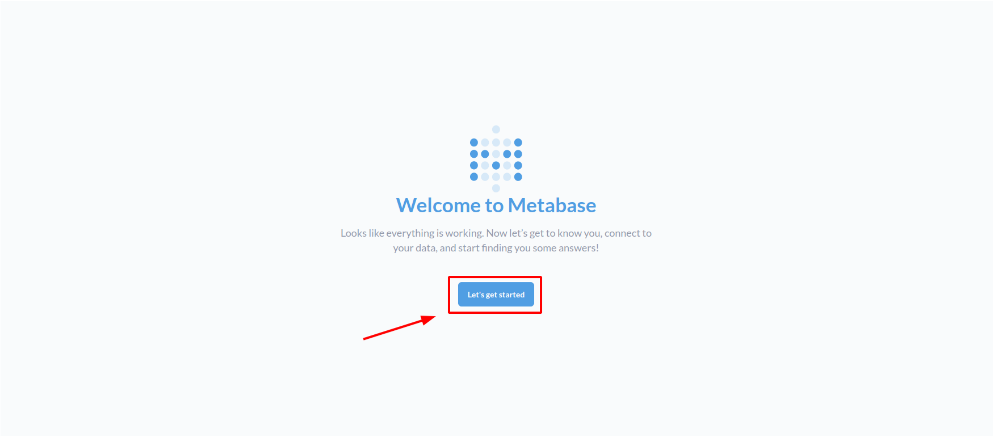
1. Metabase service is up and running
2. Have access to Metabase via URL in a browser

Creating users

To create users it is necessary to go in the browser and access Metabase using the defined URL for this service.

Once you are connected to Metabase you will see the welcome screen. To setup users please follow the following instructions:

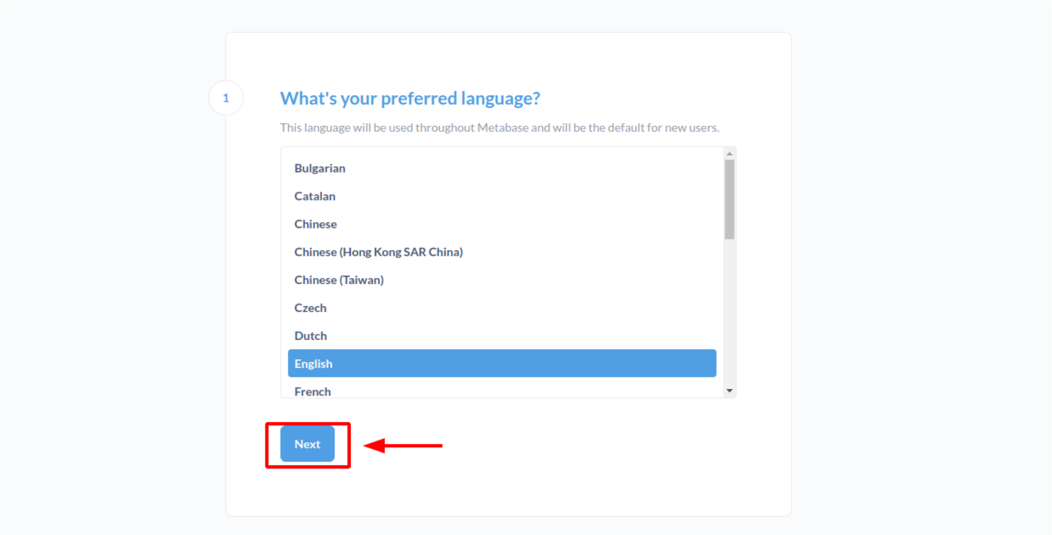
1. On the welcome screen, press Let's get started button



Welcome to Metabase

Looks like everything is working. Now let's get to know you, connect to your data, and start finding you some answers!

2. Choose the preferred language and press Next

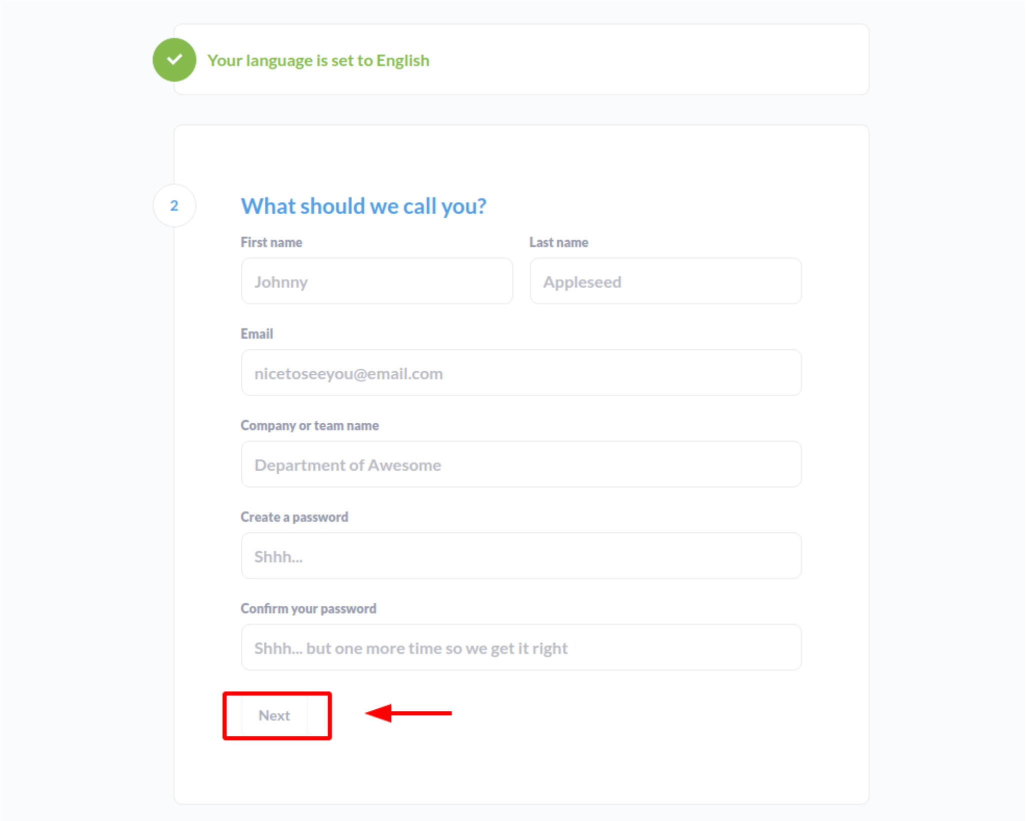


1 What's your preferred language?

This language will be used throughout Metabase and will be the default for new users.

- Bulgarian
- Catalan
- Chinese
- Chinese (Hong Kong SAR China)
- Chinese (Taiwan)
- Czech
- Dutch
- English
- French

3. Create the first user by completing the mandatory fields. This user will also be used for connecting the database. After filling the mandatory fields click Next button to continue.



Your language is set to English

2 What should we call you?

First name: Johnny

Last name: Appleseed

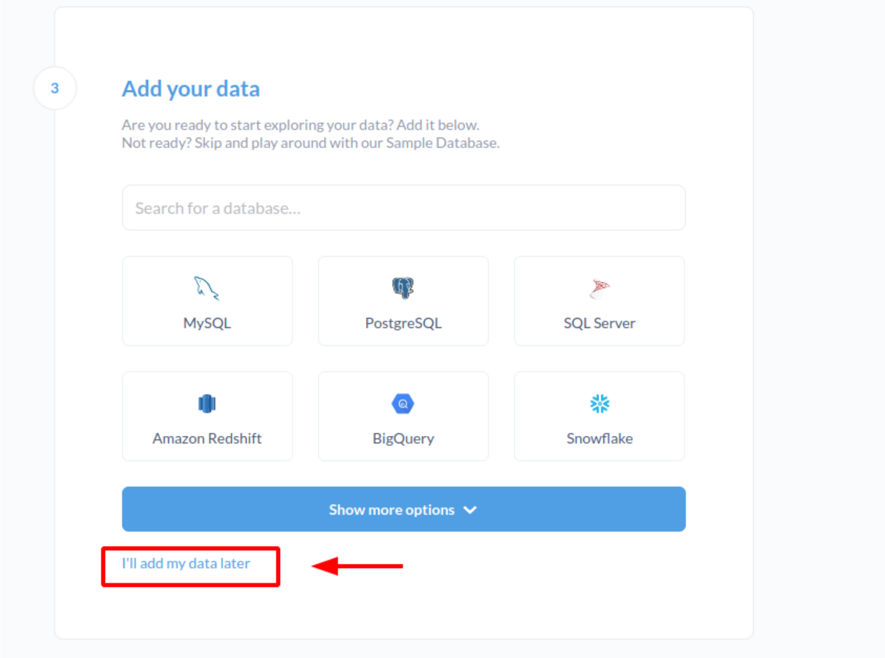
Email: nicetoseeyou@email.com

Company or team name: Department of Awesome

Create a password: Shhh...

Confirm your password: Shhh... but one more time so we get it right

4. On the add your data step press I'll add my data later to skip this step for now

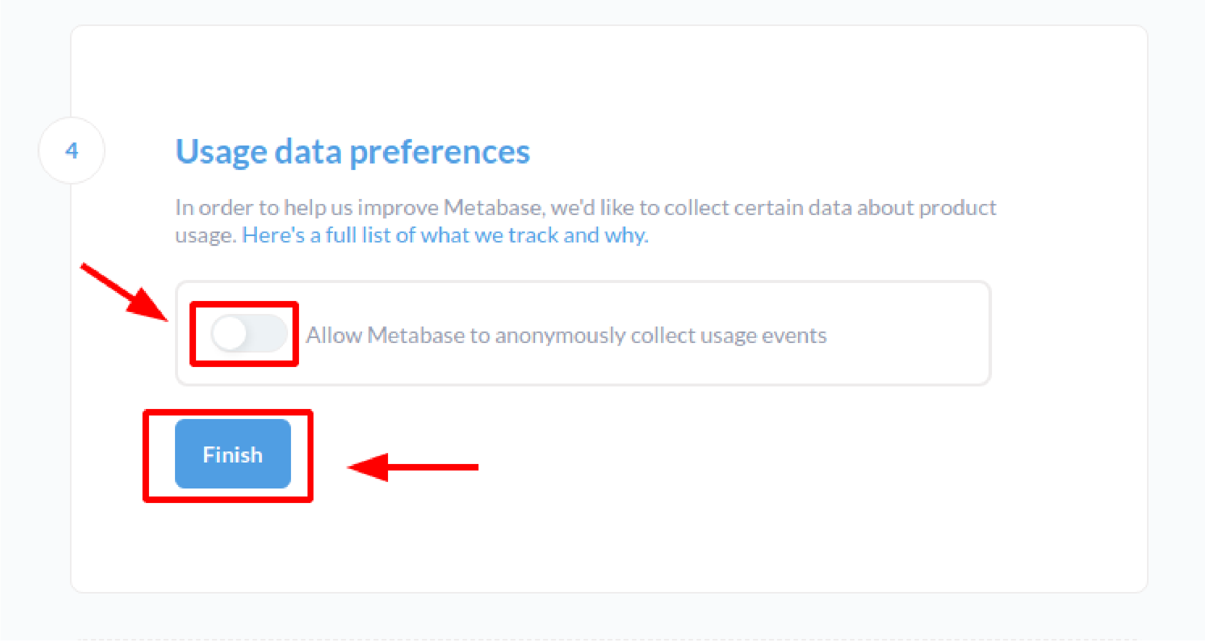


3 Add your data

Are you ready to start exploring your data? Add it below.
Not ready? Skip and play around with our Sample Database.

Search for a database...

5. On the usage data preferences step, block Metabase to collect usage events and press the finish button.



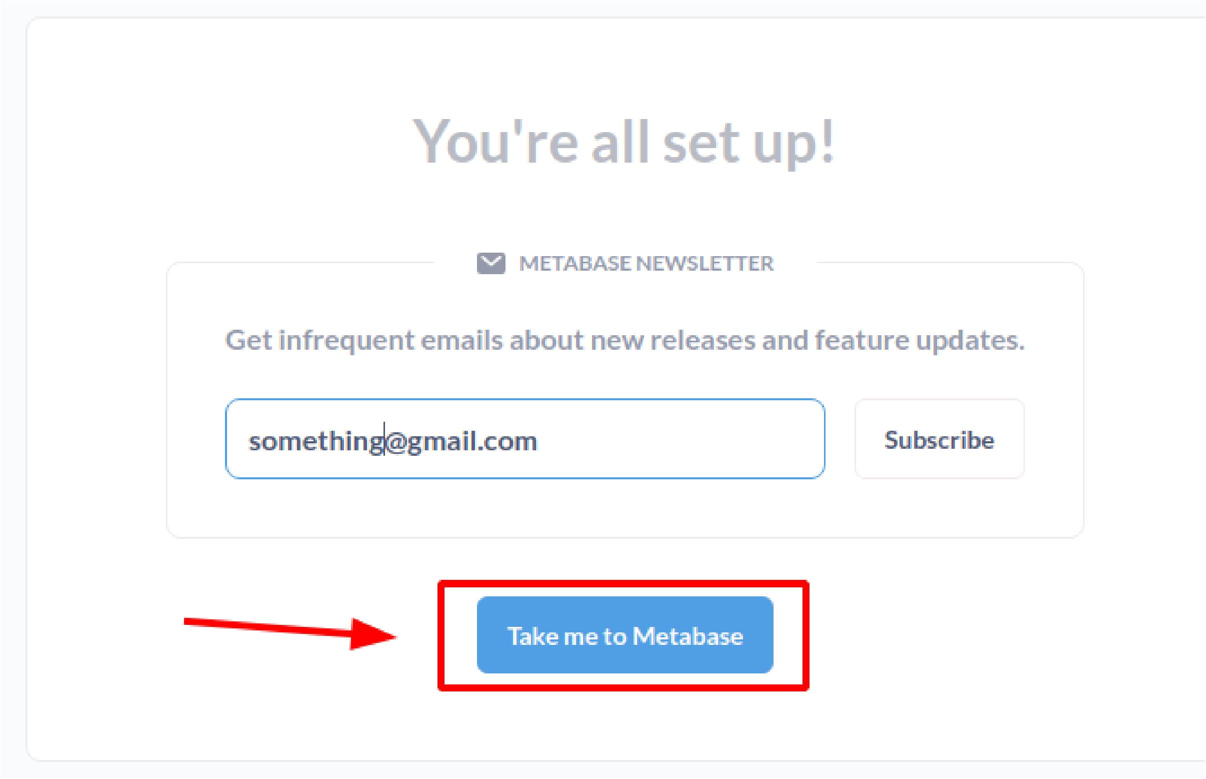
4

Usage data preferences

In order to help us improve Metabase, we'd like to collect certain data about product usage. Here's a full list of what we track and why.

○ Allow Metabase to anonymously collect usage events

6. After all the steps are finished click Take me to Metabase without subscribing



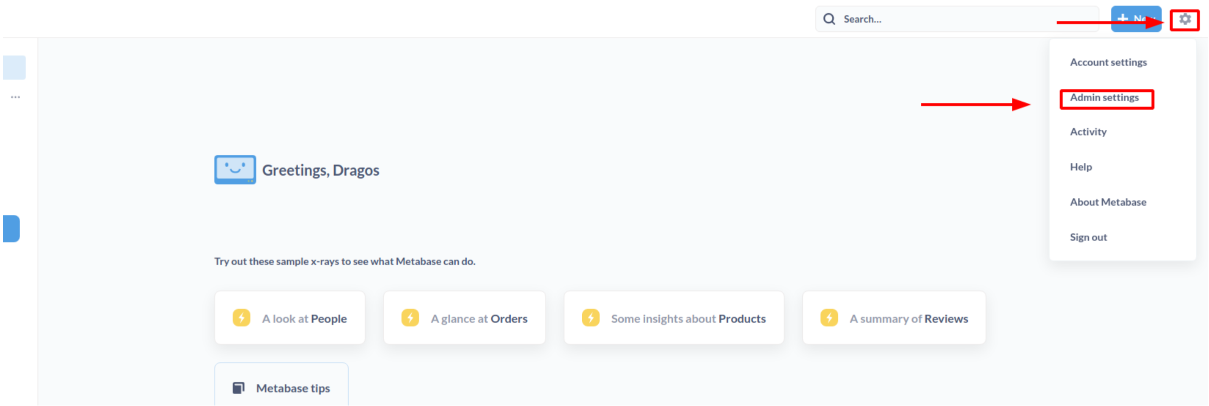
You're all set up!

METABASE NEWSLETTER

Get infrequent emails about new releases and feature updates.

something@gmail.com

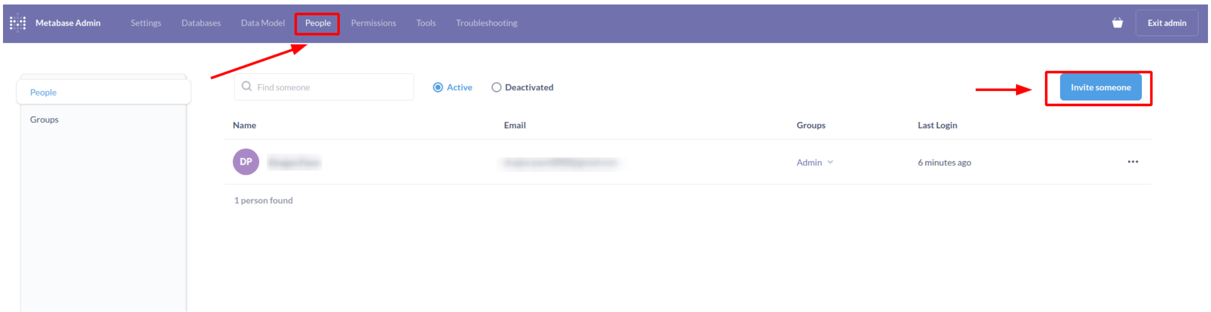
7. Now you are logged in as the user that was just created.
8. Go to Admin settings to create a second user that will be used by the end user of the TED-SWS system. First press the setting wheel button in the top right of the screen and then click Admin settings.



Greetings, Dragos

Try out these sample x-rays to see what Metabase can do.

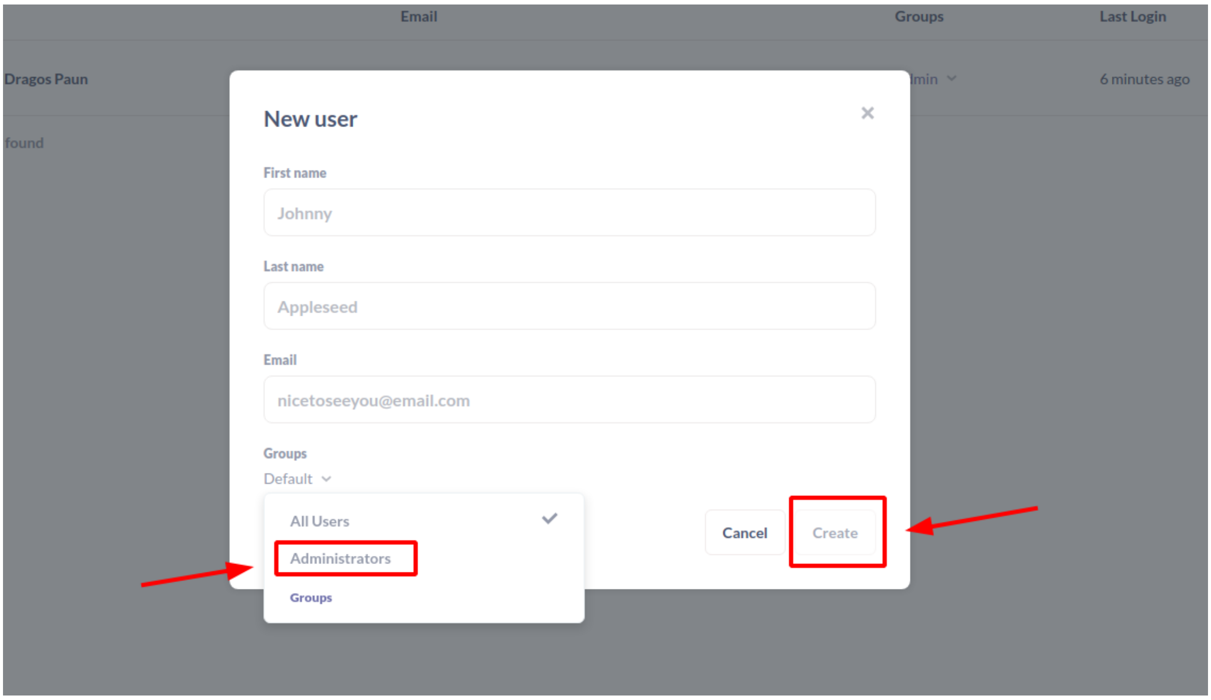
9. On the next screen go to People in the top menu and click Invite someone button



Name	Email	Groups	Last Login
DP [redacted]	[redacted]	Admin	6 minutes ago

1 person found

10. Complete the mandatory fields and put the user in the Administrator groups



New user

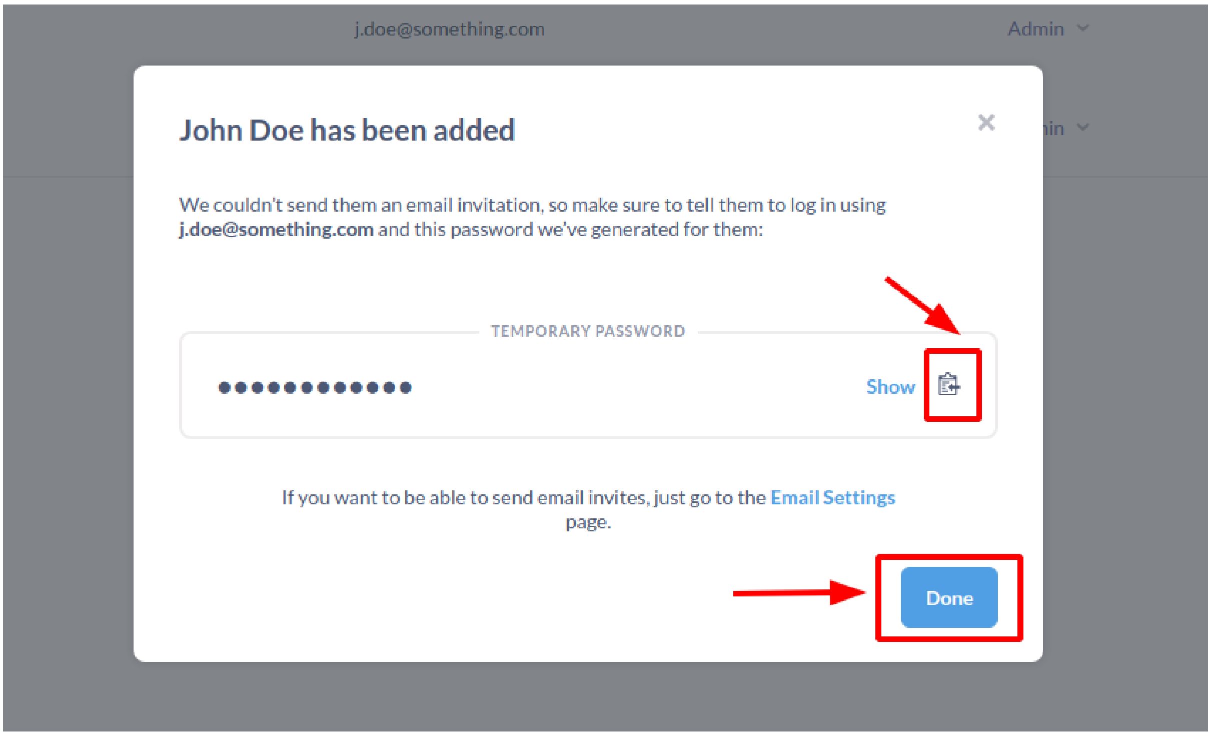
First name: Johnny

Last name: Appleseed

Email: nicetoseeyou@email.com

Groups: Default

11. Once you click on create a temporary password will be created for this user. Save this password and user details as these credentials will be used to do the import of the dashboards and then shared with the end-user of this system. After this just click Done.



John Doe has been added

We couldn't send them an email invitation, so make sure to tell them to log in using j.doe@something.com and this password we've generated for them:

TEMPORARY PASSWORD

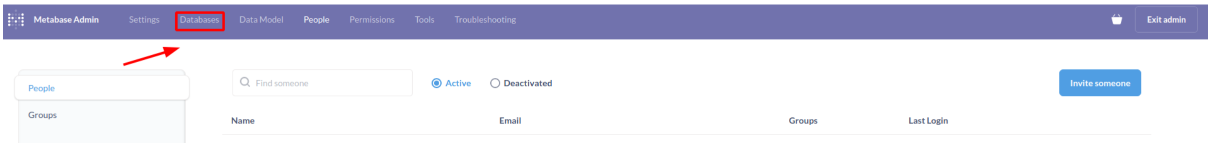
●●●●●●●●●●●●

If you want to be able to send email invites, just go to the Email Settings page.

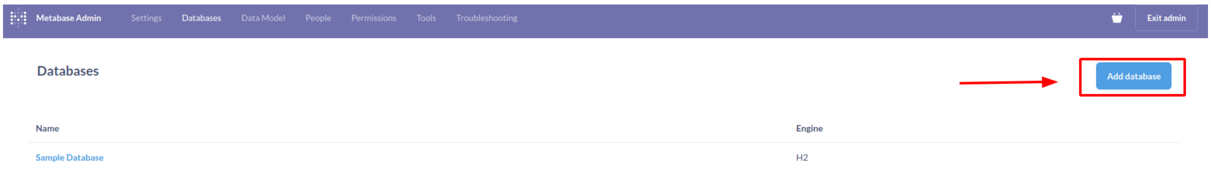
12. Stay on this page and follow the instructions for Connecting to database section below.

Connecting to database (AWS Document DB)

1. In the Metabase Admin screen go to Databases



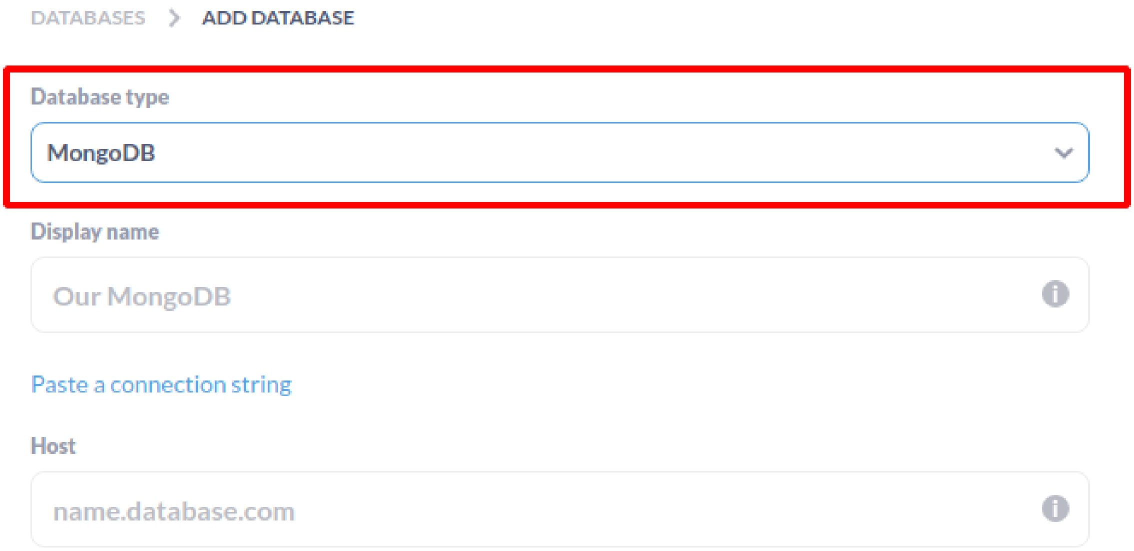
2. On the next screen click add database



Databases

Name	Engine
Sample Database	H2

3. Choose database type to be MongoDB



DATABASES > ADD DATABASE

Database type

MongoDB

Display name

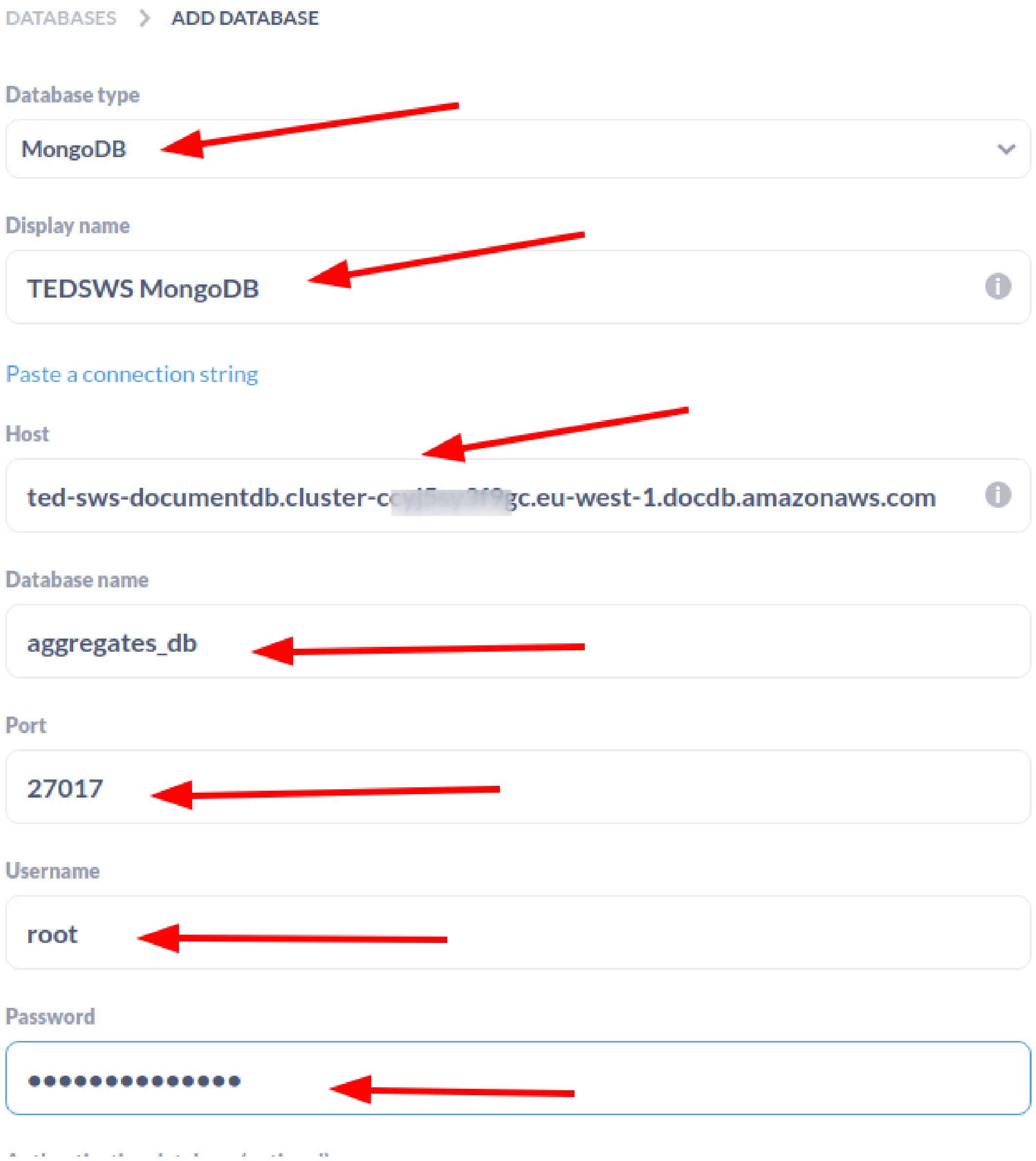
Our MongoDB

Paste a connection string

Host

name.database.com

4. Display name should be **TEDSWS MongoDB**. This is very important for the import of the dashboards.
5. Host is the endpoint for the AWS DocumentDB cluster
6. Database name should be the value that was set for this variable MONGO_DB_AGGREGATES_DATABASE_NAME in the .env file.
7. Port is the port that was set for AWS DocumentDB
8. User and password will be the master credentials set for AWS DocumentDB (*master password*, *master username*).



DATABASES > ADD DATABASE

Database type
MongoDB

Display name
TEDSWS MongoDB

Host
ted-sws-documentdb.cluster-c[redacted]gc.eu-west-1.docdb.amazonaws.com

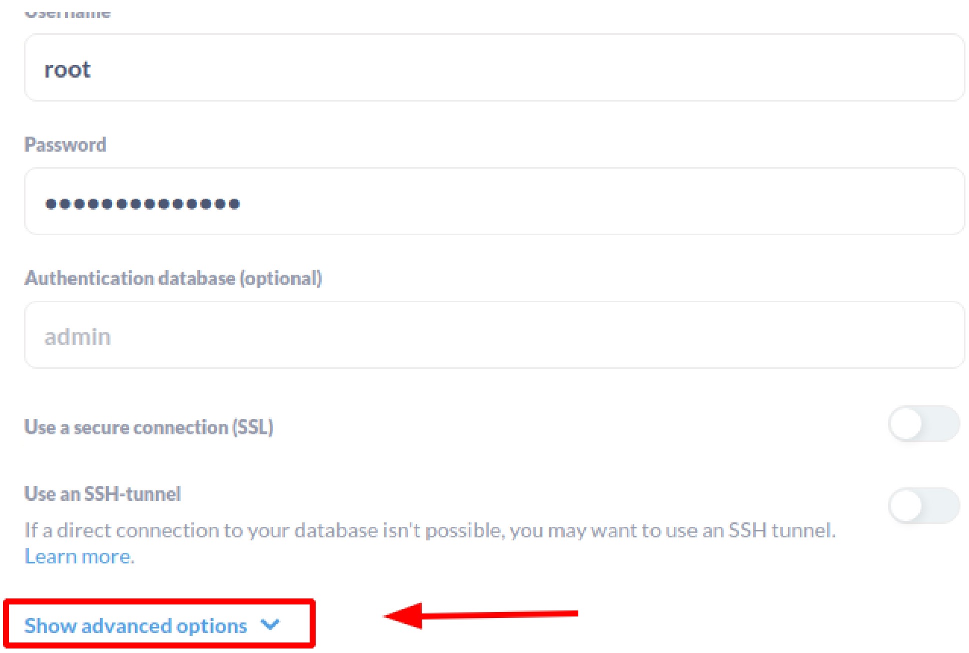
Database name
aggregates_db

Port
27017

Username
root

Password
••••••••••••••

9. Now press the advanced option to insert additional connection string option that is required for AWS DocumentDB



Username

root

Password

••••••••••••••

Authentication database (optional)

admin

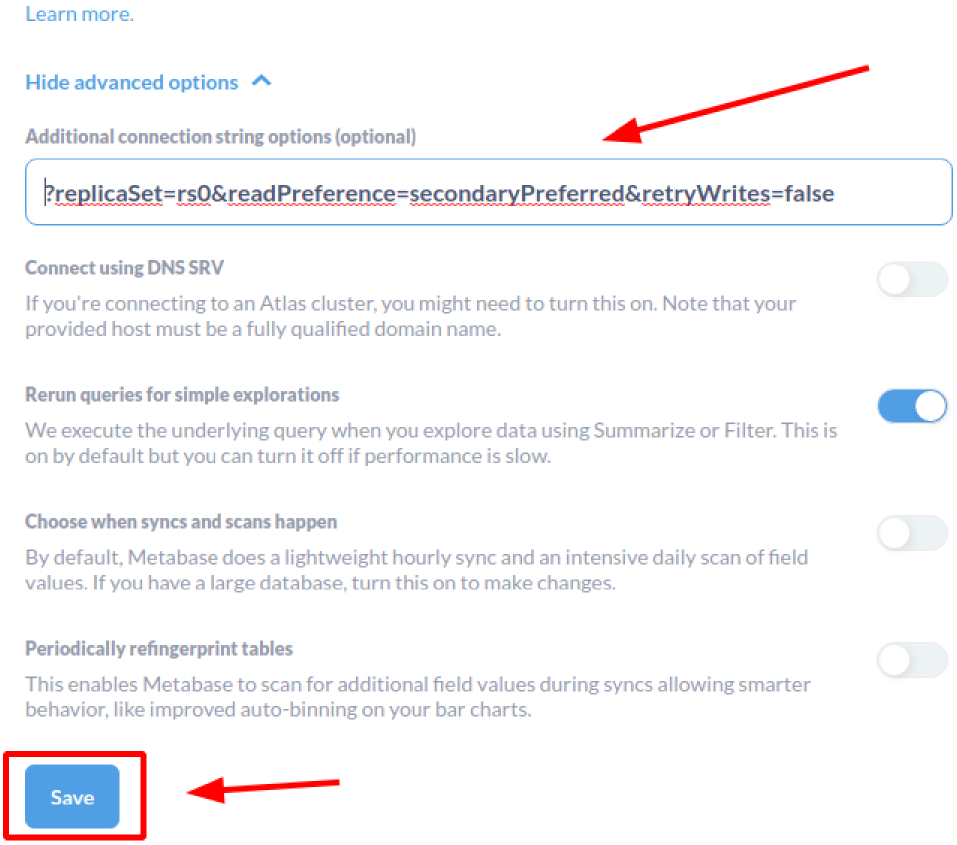
Use a secure connection (SSL)

Use an SSH-tunnel

If a direct connection to your database isn't possible, you may want to use an SSH tunnel. Learn more.

Show advanced options

10. On the dropdown section insert **?replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false** in the Additional connection string options (optional) box. After this click the Save button. If everything is correct you should not see any errors as the system will test the connection and you will be able to see 2 databases in the Databases screen.



Learn more.

Hide advanced options

Additional connection string options (optional)

?replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false

Connect using DNS SRV

If you're connecting to an Atlas cluster, you might need to turn this on. Note that your provided host must be a fully qualified domain name.

Rerun queries for simple explorations

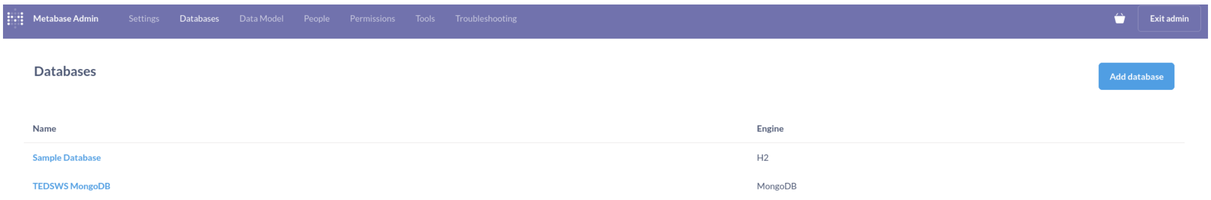
We execute the underlying query when you explore data using Summarize or Filter. This is on by default but you can turn it off if performance is slow.

Choose when syncs and scans happen

By default, Metabase does a lightweight hourly sync and an intensive daily scan of field values. If you have a large database, turn this on to make changes.

Periodically refingerprint tables

This enables Metabase to scan for additional field values during syncs allowing smarter behavior, like improved auto-binning on your bar charts.



Databases

Name	Engine
Sample Database	H2
TEDSWS MongoDB	MongoDB

Importing dashboards

Prerequisites

1. Have the metabase-toolchain repository
2. Have the export.json file (This file is in the TED-SWS source code in ted_sws/resources/metabase_export/export.json)
3. Have make installed on the machine
4. Have Python 3.8.10 and pip and installed on the machine
5. Have access to Metabase via URL in a browser from the machine that the tool will be installed on.
6. Have end-user credentials (the second user created at Creating users section of this document)

Installation of the tool

If make was installed on the machine, go inside the metabase-toolchain folder and execute the following command:

```
make install
```

Environment file

Create an .env file inside the folder that holds the source code for this tool with the following content

Variable name	Description
METABASE_HOST	Host of the Metabase service Example https://metabase.ted-sws.com
METABASE_USER	Metabase user email. This is the email used for the second user created at Creating users section of this document
METABASE_PASSWORD	User password. This should be the password for the second user created at

	Creating users section of this document
DB_AUTH_URL	This will be AWS DocumentDB connection string Example: mongodb://<user>:<insertYourPassword>@ted-sws-documentdb.cluster-ccyy3f9gc.eu-west-1.docdb.amazonaws.com:27017/?replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false
DB_NAME	Name of the database set in AWS DocumentDB and in the variable MONGO_DB_AGGREGATES_DATABASE_NAME

Import

After the tool was successfully installed and the export.json file is available on the same machine, execute the following command inside of the metabase-toolchain folder:

```
import_metabase path_to_the_export.json_file
```

Updates to the system procedures

The delivery of an updated version of the system will be accompanied by change notes. Depending on these notes we could have different scenarios in upgrading the system that will affect the system infrastructure.

Scenario 1

The new version will have only functional changes in the source code that does not affect the podman images

If this is the case the only Airflow service should be deleted and rebuilt.

Scenario 2

The new version will have functional changes in the source code and also in the changes that affect the podman images.

In this case the following steps should be followed:

- Images should be rebuild for the specified service

- Those services should be deleted and rebuilt
- If the the previous steps Airflow was not included, then Airflow service should be deleted and rebuilt

Change notes

Version 2.5.0

- Updated commands for airflow worker service and airflow flower
- Updates to fuseki service variables and mount points for data retention
    - New variable in env. file (ADMIN_PASSWORD)
    - New mount point (*/fuseki-base/configuration*)
- New section for setting Metabase and importing dashboards
- New section for upgrading the infra when new version of code is available
- Updates to SFTP configuration
    - Using EU Send SFTP server
    - Generating a key pair
    - Encode private key in base64
    - New variable in env file SFTP_PRIVATE_KEY_BASE64
    - Removing SFTP container
    - Removing the sftp-data EFS
- Changes to the digest-api functionality
    - The image should be rebuilt
- Update to the Airflow version
    - The image should be rebuilt
- The main database structure has changed