TED-SWS Installation manual

Editors Eugeniu Costetchi <eugen@meaningfy.ws>

Dragos Paun <dragos.paun@meaningfy.ws>

Version 0.9.2

Date 26/08/2022

Contents

Glossary

The official AWS glossary is available here.

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

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

Required AWS resources and configurations

AWS CLI

The configuration can be done on a machine that has AWS CLI installed with the AWS configure command. For this, an access key ID, a Secret Access Key and an AWS region is needed.

ECS CLI

The configuration of this is similar to the AWS CLI.

VPC and SUBNETS

The project will need to have a VPC and two subnets configured. The VPC id and the subnets ids should be provided manually in the configuration .env file.

AWS EFS

The project will need to have elastic file systems that will be attached to the running containers in ECS. All of them should have mount points in all the subnets available in the VPC, general purpose performance mode and to be encrypted. The list of the EFS needed is as follows:

- metabase_postgres_db
- fuseki_data
- ted_sws
- logs
- dags
- airflow_postgres_db
- mongo_db

Note: Use the names provided in the list above for the creation of the EFS as the naming is important and can affect other components if the names are changed. Also, the EFS IDs should be provided so that they can be included in the .env file.

AWS Route 53

Configure a *private hosted zone* in the VPC that will have A records for the services used in ECS that will need to be discovered/accessed by other services using the DNS resolver. The records will need to be specified manually provided in the .env file. There are 3 services that will need this:

- Mongo (mongo-service.ted_sws)
- Digest API (digest-api-service.ted_sws)
- Fuseki (fuseki-service.ted_sws)

AWS Cloud Map

This service is accessed by AWS ECS when the service is created on the cluster with the service discovery option. It will make the service discoverable by other services using the DNS records created by this service in AWS Route 53**.**

AWS ECS

Permission needed for this service will be to create and manage a) *task definitions*, b) *clusters,* and c) *services*. There will be 5 of each as follows:

- Mongo (cluster, task definition , service)
- Digest-api (cluster, task definition , service)
- Airflow (cluster, task definition , service)
- Fuseki (cluster, task definition , service)
- Metabase (cluster, task definition , service)

AWS ECR

This will be the registry for the container images used for the services in the project. Permission to create repositories and push images to those is needed.

EC2

Each cluster created with AWS ECS service will need a EC2 instance

Security Groups

A security group that controls and allows traffic on certain ports will be needed so that services can be accessed by other services in the VPC and attaching EFS volumes to containers could be possible. The ports that needs to be open to traffic would be the following:

- 27017 - mongo
- 8878 - airflow
- 2049 - EFS
- 3000 - fuseki
- 8000 - digest-api

Required permissions

The following table lists the permissions required to install and administer the system.

Service	**Permissions**	**Comments**
CloudWatch	Create, Read,	
ECR	Create, Read, Update, Delete	
ECS	Create, Read, Update, Delete	
EFS	Create, Read	
CloudWatch	Create, Read	
Route 53	Create, Read	
CloudMap	Create, Read, Update, Delete	
EC2	Create, Read, Update, Delete	
CloudFormation	Create, Run	
Security Groups	Create, Read, Update, Delete	

VPC	Create	
Subnets	Create	
IAM	Create, Update	

Note: The account that runs the script will need to have IAM:createRole permissions as it is required by the CloudFormation that is run behind the scenes when using `ecs-cli service up` command.

Available scripts

Before running any scripts an .env file should be created and should have the necessary variables configured, including *secrets*.

The scripts have been developed in an environment where unlimited permissions were provided. And so we could automate the process entirely, from configuration of the network, to creation of volumes, to creation of clusters, and bringing the application up.

In the target environment, we assume that only ECR and ECS permissions will be available. Therefore only two scripts are relevant:

- To build and upload the images: `create-and-deploy-images.sh`
- To create task definitions and services without running them: `create-services.sh`

Optionally, if the creation of services with DNS discovery is possible, then the latter can be substituted with `start-services.sh` script, otherwise bringing services up shall be performed manually.

Next, we provide an inventory of all scripts available in the repository:

- `Create-and-deploy-images.sh`
  - This script will create all the Podman images, repositories needed for this project and will deploy them to AWS ECR. Also the images URIs will be automatically written to the .env file.
- `Create-clusters.sh`
  - This script will create cluster profiles,config and EC2 instances that will be attached to the cluster.
- `Create-efs.sh`
  - This script will create all the file systems (EFS) that the project will need and 2 mount points for each one.
- `Create-services.sh`
  - This script will create task definitions and the services without running the task against the created clusters.
- `Start-services.sh`
  - This script will create task definitions and the services and will bring all services up running them in clusters.

Build and push images

Prerequisites

• Docker 20+ or Podman 3.0.1+

• AWS CLI properly configured with the permission to push into ECR repository and create ECR repositories

Performing the task

The script (create-and-deploy-images.sh) is designed to build all the Podman images of TED-SWS and push them into an ECR (https://aws.amazon.com/ecr/).

The script will perform the following steps:

- Creating a .repositories_ids file
- Creating a private repository in ECR, save the ID and write into .repositories_ids
- Build image, push it to its repository and write the image URL to the .env file

Cloud deployment

Prerequisites

This is the list of actions that need to be performed before system deployment.

- AWS CLI properly configured for the target environment
- ECS CLI properly configured for the target environment
- The team responsible for the AWS infrastructure has set up all components in AWS and has provided the necessary component IDs
- The Elastic File Systems (EFS) have been created, and the IDs were entered in the .env file. Each EFS should have mount points in the two subnets (the ones specified in .env).
- Clusters profiles and config for the `ecs-cli` have been created with the following commands. This process can be automated by using `create-clusters.sh` script. ***Note*** *that naming is important and shall be followed as provided in the section on cluster references below.*

  ```
  ecs-cli configure --cluster cluster-name --default-launch-type EC2
  --config-name config-name --region $REGION
  ```

  ```
  ecs-cli configure profile --access-key $ACCESS_KEY --secret-key
  $SECRET_KEY --profile-name cluster-profile-name
  ```

- Clusters with EC2 following the naming conventions mentioned above should be created in VPC and available in the subnets specified in the .env file. The clusters will need to have attached a security group that will have inbound rules for mounting EFS on port 2049.
- Images were built and registered in the ECR as per Build and push images step.
- ecsInstanceRole is created with AmazonEC2ContainerServiceforEC2Role policy attached.
- A namespace ted_sws is configured in CloudMap that will have fuseki-service, digest-api-service,mongo-service. Each service will have the EC2 instance registered here. The goal is to have a DNS record **type A** for the domain names provided below that will point to the EC2 ip for that service in Route 53:
    - mongo-service.ted_sws,
    - digest-api-service.ted_sws,
    - fuseki-service.ted_sws
- Create .env file and populate it with the appropriate variables and values as described below.
- Copy the .env file into /ted-sws/infra/aws folder.
- Copy the .env file into **ted_sws** EFS root folder (/)

EFS volumes

EFS Name	**Service**
mongo_db	Mongo
airflow_postgres_db	Airflow
dags	Airflow
logs	Airflow
ted_sws	Airflow
fuseki_data	Fuseki
metabase_postgres_db	Metabase

Clusters and EC2 instances

The following clusters are foreseen:

1. MongoDB EC2 Cluster (mongo-cluster)

- Instance type: c5.4xlarge / r6gd.4xlarge
- Instances: 1
- Services:

        - MongoDb
2. Airflow EC2 Cluster (airflow-cluster)
- Instance type: m6gd.8xlarge
- Instances: 1
- Servicers:
    - Webserver
    - Scheduler
    - Trigger
    - Initialiser
    - Redis
    - Postgres
    - Worker
3. Metabase EC2 Cluster (metabase-cluster)
- Instance type: t4g.xlarge
- Instances: 1
- Servicers:
    - Metabase
    - Postgres
4. Fuseki EC2 Cluster (fuseki-cluster)
- Instance type: r5ad.2xlarge
- Instances: 1
- Servicers:
    - Fuseki
5. Digest API EC2 Cluster (digest-api-cluster)
- Instance type: c6gd.large
- Instances: 4
- Servicers:
    - Digest API

Cluster references

Cluster profile name	**Cluster config name**	**Cluster name**
mongo-cluster-profile	mongo-cluster	mongo-cluster
digest-api-cluster-profile	digest-api-cluster	digest-api-cluster
airflow-cluster-profile	airflow-cluster	airflow-cluster
fuseki-cluster-profile	fuseki-cluster	fuseki-cluster
metabase-cluster-profile	metabase-cluster	metabase-cluster

Environment files

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. Descriptions of the properties that need to be updated are provided in the file.

The updated environment file needs to be copied into the shared Airflow EFS volume (ted_sws) mounted to the Airflow container. Also, this file needs to be created /ted-sws/infra/aws (project files) before any script is runned.

The environment variables are defined in the following table.

Name	**Description**
MONGO_INSTANCE_TYPE	Type of EC2 instance for mongo service. This variable is useful only if the **create-clusters.sh** script is used
DIGEST_API_INSTANCE_TYPE	Type of EC2 instance for digest-api service. This variable is useful only if the **create-clusters.sh** script is used
AIRFLOW_INSTANCE_TYPE	Type of EC2 instance for airflow service. This variable is useful only if the **create-clusters.sh** script is used
FUSEKI_INSTANCE_TYPE	Type of EC2 instance for fuseki service. This variable is useful only if the **create-clusters.sh** script is used
METABASE_INSTANCE_TYPE	Type of EC2 instance for metabase service. This variable is useful only if the **create-clusters.sh** script is used
SUBNET_1_ID	AWS subnet id 1
SUBNET_2_ID	AWS subnet id 2
SECURITY_GROUP	AWS security group id.This variable is useful only if the **create-clusters.sh** script is used
VPC_ID	The id for the VPC in AWS used to deploy the services
ACCESS_KEY	AWS Access Key
SECRET_KEY	Aws Secret Key
REGION	AWS Region that the services will be deployed
FUSEKI_IMAGE_URI	Image uri from the ECR for Fuseki service
AIRFLOW_POSTGRES_IMAGE_URILOGS_VOLUME_ID	Image uri from the ECR for Postgres used for the Airflow service
METABASE_IMAGE_URI	Image uri from the ECR for Metabase service
MONGO_IMAGE_URI	Image uri from the ECR for Mongo service
REDIS_IMAGE_URI	Image uri from the ECR for Redis used for the Airflow service

METABASE_POSTGRES_IMAGE_URI	Image uri from the ECR for Postgres used for the Metabase service
AIRFLOW_IMAGE_URI	Image uri from the ECR for Airflow service
DIGEST_API_IMAGE_URI	Image uri from the ECR for Digest-api service
FUSEKI_DATA_VOLUME_ID	Fuseki_data EFS id
TED_SWS_VOLUME_ID	Ted_sws EFS id
LOGS_VOLUME_ID	Logs EFS id
DAGS_VOLUME_ID	Dags EFS id
AIRFLOW_POSTGRES_DB_VOLUME_ID	Airflow_postgres_db EFS id
MONGO_DB_VOLUME_ID	Mongo_db EFS id
METABASE_POSTGRES_DB_VOLUME_ID	Metabase_postgres_db EFS id
_AIRFLOW_WWW_USER_PASSWORD	Airflow UI user
_AIRFLOW_WWW_USER_USERNAME	Airflow UI user password
AIRFLOW_GID	Airflow user permissions. This should be 50000
AIRFLOW_UID	Airflow user permissions. This should be 50000
FUSEKI_ADMIN_PASSWORD	Fuseki admin password
FUSEKI_DATASET_1	Fuseki default dataset
ENV_MB_DB_DBNAME	Postgres database name for metabase.
ENV_MB_DB_PORT	Postgres port. This should be 5432
ENV_MB_DB_USER	Username for metabase postgres database
ENV_MB_DB_PASS	Password for postgres database for the metabase user
ENV_MB_ENCRYPTION_SECRET_KEY	Encryption secret key
MONGO_INITDB_ROOT_PASSWORD	Mongo root password
MONGO_INITDB_ROOT_USERNAME	Mongo root username
LOGGING_TYPE	Logging type

MONGO_DB_AGGREGATES_DATABASE_NAME	Mongo database name for notice aggregates
MONGO_DB_AUTH_URL	Mongo connection URL. Example: mongodb://**user**:**password@service_address:27017**/
MONGO_DB_LOGS_COLLECTION	Mongo logs collection name
MONGO_DB_LOGS_DATABASE_NAME	Mongo logs database name
MONGO_DB_PORT	Mongo port.
ID_MANAGER_API_HOST	The host for the digest-api service
LOGGER_LOGGING_HANDLER	Logging handler
LOGGER_MONGO_HANDLER_BUFFER_PERIODICAL_FLUSH_TIMING	Buffer periodical flush time
GITHUB_TED_SWS_ARTEFACTS_URL	GitHub URL for artefacts repository on GitHub https://github.com/meaningfy-ws/ted-sws-artefacts.git

Performing the task

The script (create-services.sh) is designed to create services and the task definitions, but not execute them. At the moment, we do not know how the final network setup is organised and therefore this script (create-services.sh) creates services with *service discovery* and *without service* discovery. This aspect can be controlled, however from configuration variables as described below. We recommend keeping the default configuration.

Services created by this script:

- airflow-service
- metabase-service
- mongo-service
- digest-api-service
- fuseki-service

Variables in the script:

- SERVICES_WITHOUT_DNS
  - list of services to be created without A records in the DNS
- SERVICES_WITH_DNS_CREATION
  - list of services to be created with A records in the DNS

If the configurations in AWS Cloud Map are done manually before running the script, then the values should be:

- SERVICES_WITHOUT_DNS=(airflow metabase mongo digest-api fuseki)
- SERVICES_WITH_DNS_CREATION=()

Otherwise, you can rely on the default configuration, which creates A records as follows

- SERVICES_WITHOUT_DNS=(airflow metabase)
- SERVICES_WITH_DNS_CREATION=(mongo digest-api fuseki)

After the script is executed the task definitions and services will be created but will not be executed. If the script was executed with the variables changed (i.e no service with DNS creations) the service discovery needs to be manually configured with the resources created in AWS CloudMap for the following services:

- mongo-service,
- digest-api-service,
- Fuseki-service.

Note, however, that once the service is created without the service discovery option enabled, it is not possible to enable it through update operation in AWS user interface. Likely a delete and create needs to be performed for a successful outcome. Therefore we recommend running the above script with service discovery as is configured by default.