ADAPT Deployment Orchestrator: installation and usage

Table of Contents

List of Figures

List of Tables

No table of figures entries found.

1 Delivery and Usage

1.1 Package information

The ADAPT Deployment Orchestrator is packaged as a Docker image, to be launched as a container on any system running a Docker instance.

The image is currently available on an "unofficial" private Docker registry that we have set up for DECIDE, for development and testing purposes. Credentials are needed to login to the registry and to pull images. There are plans to set up publicly available, official repositories in the next iteration of the project for accessing code, images and all the needed material for running DECIDE components.

1.2 Installation instructions

Being packaged as a Docker image, the ADAPT Deployment Orchestrator does not require specific installation steps.

The only pre-requisite is to have access to a Linux-based system running a Docker daemon.

Currently, there are three options available to get the ADAPT Deployment Orchestrator image:

1. Get the image from the private Docker registry, in case you have been assigned credentials.
2. Get the image from a tar archive, available for download in case you have been authorized.
3. Build the image directly from the ADAPT project code.

Requests for credentials, download urls and access to systems can be submitted by filling the form at the following url: https://www.decide-h2020.eu/contact

1.2.1 Using the image from the private Docker registry

For users who have access to the private Docker registry where the ADAPT Deployment Orchestrator image is stored, the operations to perform to start it are:

- Log into the registry (only for first-time access) with the dedicated Docker command:

```
shell> docker login [registryIp]:[registryPort] -u registryUser -p registryPassword
```

- Execute the Docker 'run' command on the image, mapping a proper port which is open on the host and defining the –d (daemon) flag:

```
shell> docker run -p 8473:80 -d --name adapt [registry_ip]:[registry_port]/adapt:m24
```

- Verify that the container is up and running with the 'ps' command:

```
shell> docker ps
REPOSITORY                    TAG         IMAGE ID          CREATED          SIZE
aaa.bbb.ccc.ddd:nnnn/adapt    m24         4537a4171c5a      12 days ago       1.69GB
```

1.2.2 Extracting the image from a tar.gz archive

Images that are saved as tar archives, can be loaded on a local machine with the Docker 'image load' command.

- Download the 'adapt.tar.gz' file from the url received from your request submission

- Unzip the archive with the 'gunzip' tool:

```
shell> gunzip adapt.tar.gz
```

- Load the image from the Docker shell with the following command:

```
shell> docker image load -i /home/ubuntu/adapt.tar

2c40c66f7667: Loading layer [==================================================>]
129.3MB/129.3MB

654f45ecb7e3: Loading layer [==================================================>]
45.45MB/45.45MB

f3ed6cb59ab0: Loading layer [==================================================>]
126.8MB/126.8MB

5616a6292c16: Loading layer [==================================================>]
326.7MB/326.7MB

97108d083e01: Loading layer [==================================================>]
8.043MB/8.043MB

815acdffadff: Loading layer [==================================================>]
62.18MB/62.18MB

325b9d6f2920: Loading layer [==================================================>]
4.608kB/4.608kB

2548e7db2a94: Loading layer [==================================================>]
5.591MB/5.591MB

9a9ce7dcd474: Loading layer [==================================================>]
8.599MB/8.599MB

df08e2c3d6fe: Loading layer [==================================================>]
5.209MB/5.209MB

3c0d8f1e556d: Loading layer [==================================================>]
3.584kB/3.584kB

87e2b99e95df: Loading layer [==================================================>]
3.584kB/3.584kB

5babbba9a986: Loading layer [==================================================>]
3.072kB/3.072kB

433a67f63093: Loading layer [==================================================>]
3.584kB/3.584kB

394c0a98982c: Loading layer [==================================================>]
3.072kB/3.072kB

461de7fb06ff: Loading layer [==================================================>]
5.346MB/5.346MB

b61bb40af46f: Loading layer [==================================================>]
3.584kB/3.584kB

40dc546a568a: Loading layer [==================================================>]
2.048kB/2.048kB

357d65e53b78: Loading layer [==================================================>]
2.048kB/2.048kB

66ddad2c15b4: Loading layer [==================================================>]
3.584kB/3.584kB

31b3b1f0ab7b: Loading layer [==================================================>]
5.5MB/5.5MB

bba5e66e1bc9: Loading layer [==================================================>]
3.072kB/3.072kB

0889a339caa2: Loading layer [==================================================>]
3.072kB/3.072kB

475e51c5e84e: Loading layer [==================================================>]
3.584kB/3.584kB
```

```
297a10341f47: Loading layer [==================================================>]
67.77MB/67.77MB

3cb698cec5c8: Loading layer [==================================================>]
45.06kB/45.06kB

be6e62470c3b: Loading layer [==================================================>]
17.58MB/17.58MB

b8983b6f72e0: Loading layer [==================================================>]
10.25MB/10.25MB

Loaded image: aaa.bbb.ccc.ddd:nnnn/adapt:m24
```

- Verify the image is available from the set of local Docker images:

```
shell> docker images

REPOSITORY                      TAG          IMAGE ID            CREATED             SIZE

aaa.bbb.ccc.ddd:nnnn/adapt    m24          4537a4171c5a        12 days ago         1.69GB
```

- Run the image:

```
shell> docker run -p 8473:80 -d --name adapt aaa.bbb.ccc.ddd:nnnn/adapt:m24
```

- Verify that the container is up and running with the 'ps' command:

```
shell> docker ps

REPOSITORY                      TAG    IMAGE ID        CREATED       SIZE

aaa.bbb.ccc.ddd:nnnn/adapt    m24    4537a4171c5a     12 days ago   1.69GB
```

1.2.3 Building the image from code

If you have access to the code repository, the ADAPT Deployment Orchestrator Docker image can be easily built from the Dockerfile available in the 'adapt-do' project repository available at the following address:

https://git.code.tecnalia.com/decide/adapt-do

The repository can be easily accessed via the project Gitlab web GUI. To build the ADAPT DO image from code for the release at M24, follow the steps below:

- Click on the repository url and, after logging in, select the tag "m24" from the drop down menu containing the available project branches and tags, as shown in Figure 1

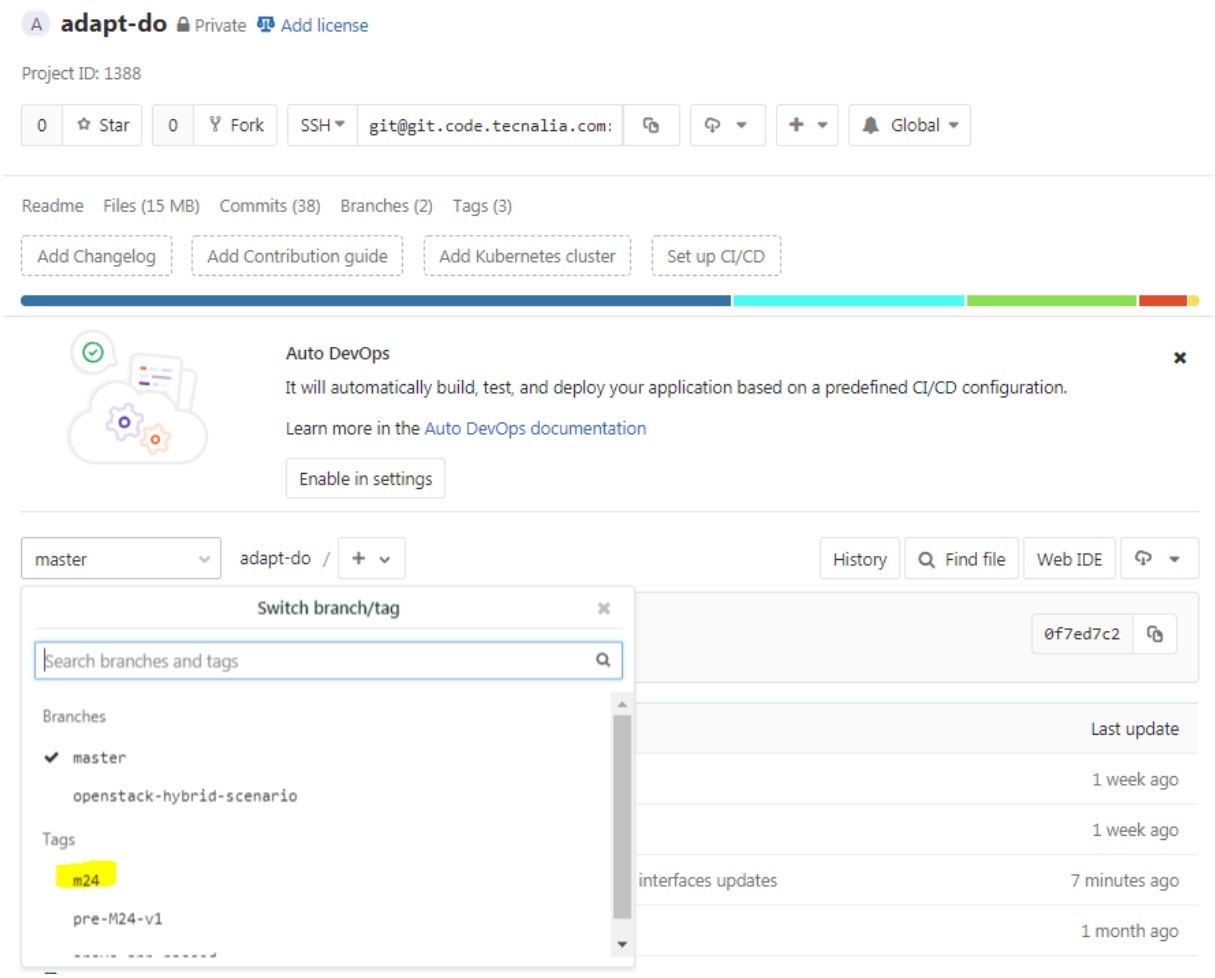

Figure 1 Selecting the M24 tag for getting the ADAPT DO source code released at M24

- Download the project code from Gitlab as zip or archive file clicking on the “download” icon, as shown in

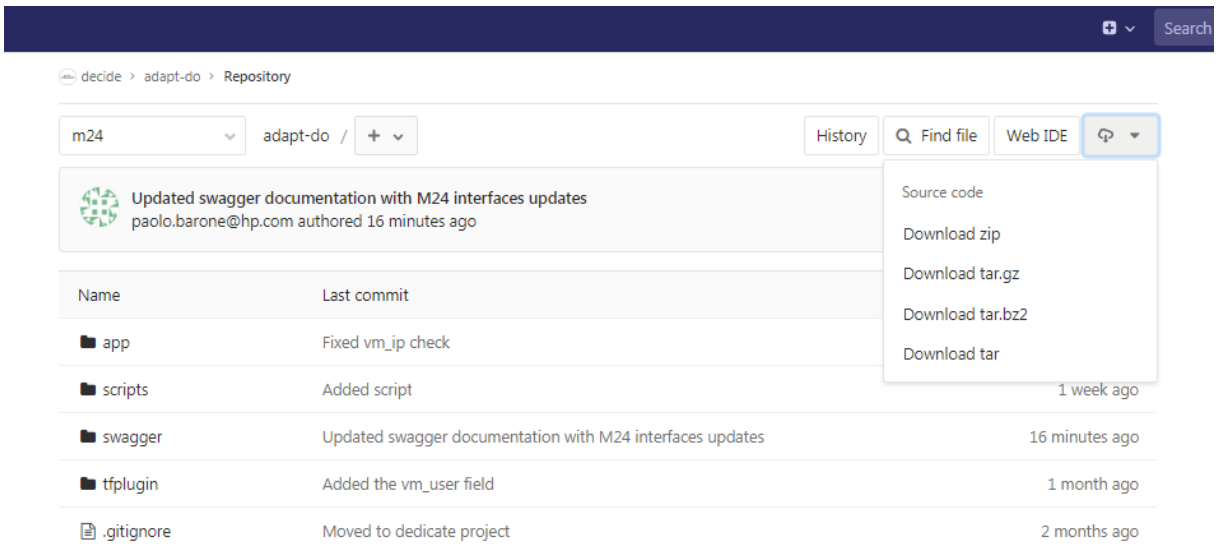

Figure 2 Downloading the source code

- Extract the archive into a directory of your choice (e.g. ‘~/tmp) and go into the directory ‘adapt-do/’

- Run the command:

```
shell> sudo docker build -t adapt:m24 .
```

The above command builds the image 'adapt' with tag 'm24.

- Verify the image is available from the set of local Docker images:

```
shell> docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
adapt                       m24                 4537a4171c5a        12 days ago         1.69GB
```

- Run the image:

```
shell> docker run -p 8473:80 -d --name adapt adapt:m12
```

- Verify that the container is up and running with the 'ps' command:

```
shell> docker ps
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
adapt                       m24                 4537a4171c5a        12 days ago         1.69GB
```

1.3 User Manual

Once started, the ADAPT Deployment Orchestrator provides the REST API endpoints (documented in D4.4, Section 2.2.2 and updated in Section **Error! Reference source not found.** of the current document) for interactions with the other DECIDE components.

It is possible to test the component independent of the rest of the system, for evaluation purposes, by feeding the endpoints with proper input parameters.

Please consider that ADAPT Deployment Orchestrator is a component which is meant to be used by automatic tools; reproducing the behaviour via human interactions may result difficult due to the amount of manual interactions and configuration steps. As reported in Section **Error! Reference source not found.**, we provided also some CLI tools to facilitate this type of activity.

The pre-requisites for successful testing are:

- Having credentials to a git repository where an Application Descriptor document will be fetched by ADAPT DO.
- Having credentials and permissions to access and start resources on the cloud broker environment configured. Such credentials comprise:
  - Username and password, to interact to the broker API and start resources.
  - SSH keypairs, to connect to the resources created. These are usually created and set up in the user profile creation steps, when registering to a cloud provider service. *In addition, it is also necessary to get the internal ID of the keypairs, which can be extracted automatically by automated tools but requires advanced steps for manual tests*. We have set up a test user with well-known ids and credentials in order to facilitate the verification from human end users.
- Having credentials to access the DECIDE private Docker image registry.

The steps for the test are:

1. Provisioning of an ADAPT Deployment Orchestrator instance specific to the test application.

2. Creation of Terraform configuration files for the infrastructure and for the services environments.
3. Initialization and planning of the infrastructure.
4. Provisioning of the infrastructure.
5. Initialization and planning of the services.
6. Provisioning of the services.
7. Verification of the application.

We suggest to use the CLI tools documented in Section **Error! Reference source not found.**, as they allow to get most of the required steps done automatically.

If you want to have a detailed insight of what is happenin behind the scene step-by-step, then you can follow the steps deocumented in the remainder of this Chapter.

1.3.1 Provisioning of an ADAPT Deployment Orchestrator instance

You can choose one of the options described in Section 1.2 to get a Docker image for ADAPT DO and follow the directions described to start it.

1.3.2 Creation of Terraform configuration files for the infrastructure and for the services environments

In the DECIDE workflow, the information contained in the Application Descriptor is passed to ADAPT via a Git repository, specific to the application that must be deployed. The file is stored and updated on that repository, and the Application Controller invokes ADAPT by POSTing a JSON data structure containing the needed information to access the repository and the specific revision of the file. To reproduce this mechanism manually, you have to specify your repository data and push a configuration file there, according to the following directions. In the 'adapt-do/demo-m12' folder, open the file 'app-descriptor.json' (which contains the description of the cloud resources we want to create: two virtual machines and a set of containers for running the Socks Shop application) and replace:

- all the fields marked with 'TO_BE_FILLED' with proper credentials;
- The fields marked with 'TO_BE_FILLED_WITH_ADAPT_IP' with the IP address of the ADAPT instance generated in the previous step.

Now, you have to push this file into your Git repository, at a well-specified path, and get the revision number of the file. This information is required in the next step. Here follows some directions on how to do it. Let's assume that your Git repository is named "my-app-repo", and that you have cloned it locally via the Git "clone" command as in the following:

```
shell> git clone git@git.code.myrepositories.com:decide/my-app-repo.git
```

You have now to copy the modified 'app-descriptor.json' in your project folder, then commit and push it to the remote repository:

```
shell> git commit -m "updated app descriptor" app-descriptor.json

shell> git push
```

Now, you have to get the revision of the file:

```
shell> git rev-parse HEAD

06f926e7bc8a6bd40703874dd9c05ed71e6ca49a
```

The returned hexadecimal code is the revision number, needed in the next step together with the information related to your Git repository.

- In the 'adapt-do/demo-m12' folder, open the file 'preparation-post-data.json' and replace:
  - all the fields marked with 'TO_BE_FILLED_XXX' with proper data.
- Using the 'curl' command line tool, POST the json data to the REST endpoint for the creation of configuration files for the infrastructure and for the services:

```
shell> curl -H "Content-Type: application/json" -X POST --data @preparation-post-data.json
http://[ADAPT_IP]:[ADAPT_PORT]/terraform/all
```

- Verify that a folder named 'My-Example-App' is created on the container:

```
shell> sudo docker exec -it adapt ls
```

- Verify the following subfolders structure is there:

```
shell> sudo docker exec -it adapt ls -R My-Example-App
My-Example-App:
infrastructure  services
My-Example-App/infrastructure:
My-Example-App-vm-node-1.tf
My-Example-App-vm-node-2.tf
My-Example-App-adapt/services:
My-Example-App-container-carts-db.tf
My-Example-App-container-carts.tf
My-Example-App-container-catalogue-db.tf
My-Example-App-container-catalogue.tf
My-Example-App-container-front-end.tf
My-Example-App-container-orders-db.tf
My-Example-App-container-orders.tf
My-Example-App-container-payment.tf
My-Example-App-container-queue-master.tf
My-Example-App-container-rabbitmq.tf
My-Example-App-container-shipping.tf
My-Example-App-container-traefik-private.tf
My-Example-App-container-user-db.tf
My-Example-App-container-user.tf
My-Example-App-container-zipkin.tf
My-Example-App-network-node-1-carts-network.tf
My-Example-App-network-node-1-user-network.tf
My-Example-App-network-node-2-catalogue-network.tf
My-Example-App-network-node-2-orders-network.tf
My-Example-App-network-node-2-shipping-network.tf
My-Example-App-services-common.tf
```

The files with '.tf' extension contains the configurations for Terraform.

1.3.3 Initialization and planning of the infrastructure

- Initialize the infrastructure environment:

```
shell> curl -H "Content-Type: application/json" -X POST
http://[ADAPT_IP]:[ADAPT_PORT]/terraform/init/My-Example-App/infrastructure
```

You can poll the urls returned by the command to verify the status and logs of the request.

- Create a deployment plan for the infrastructure:

```
shell> curl -H "Content-Type: application/json" -X POST
http://[ADAPT_IP]:[ADAPT_PORT]/terraform/plan/My-Example-App/infrastructure
```

You can poll the urls returned by the command to verify the status and logs of the request.

1.3.4 Provisioning of the infrastructure

- Deploy the infrastructure:

```
shell> curl -H "Content-Type: application/json" -X POST
http://[ADAPT_IP]:[ADAPT_PORT]/terraform/apply/My-Example-App/infrastructure
```

You can poll the urls returned by the command to verify the status and logs of the request.

1.3.5 Initialization and planning of the services

- Initialize the services environment:

```
shell> curl -H "Content-Type: application/json" -X POST
http://[ADAPT_IP]:[ADAPT_PORT]/terraform/init/My-Example-App/services
```

You can poll the urls returned by the command to verify the status and logs of the request.

- Create a deployment plan for the services:

```
shell> curl -H "Content-Type: application/json" -X POST
http://[ADAPT_IP]:[ADAPT_PORT]/terraform/plan/My-Example-App/services
```

You can poll the urls returned by the command to verify the status and logs of the request.

1.3.6 Provisioning of the services

- Deploy the service:

```
shell> curl -H "Content-Type: application/json" -X POST
http://[ADAPT_IP]:[ADAPT_PORT]/terraform/apply/My-Example-App/services
```

You can poll the urls returned by the command to verify the status and logs of the request.

1.3.7 Verification of the application

If the above steps succeeded, you would be able to access the Socks Shop application at the url:

- http://[node-2-ip]:80

Where [node-2-ip] is the address of the virtual machine 2 started by the 'infrastructure apply' operation.

1.4 Licensing information

The plan is to release the ADAPT DO component, developed by HPE, as open source software. HPE has to follow an internal process with reviews and decisions at corporate level to decide and approve the license under which to release the developed software. Unfortunately, this process takes time and it is not yet completed at the time of writing, therefore the licensing information for the released software is not yet available. However, credentials can be provided under request, download urls and access to systems can be requested by filling the form at the following url: https://www.decide-h2020.eu/contact.