**Core Deployment**

This document explains how to deploy the symbIoTe Core components stack using docker directly in a VM running Linux. This is based on the online documentation for symbiote-core available at: https://github.com/symbiote-h2020/SymbioteCore.

**1. Preparation steps**

**1.1. Install prerequisites**

* docker (18.03.x),
* docker-compose (1.21.x),
* bash,
* curl
  1. **Create deployment folder at file system**
* Create a *symbiote-core* folder at the virtual machine’s file system, e.g., mkdir symbiote-core
* Then inside the *symbiote-core* folders create the *configuration* folder:

cd symbiote-core

mkdir configuration

**1.3. Create the Core AAM certificate required for your deployment**

You need to create a PKCS12 keystore containing a certificate and copy it inside the *configuration* folder; the steps to follow are described below. The keystore will be used to self-initialize the AuthenticationAuthorizationManager (AAM) codes as Core AAM.

The keystore is used by Core AAM microservice to authenticate third-party users and applications (i.e., users and applications that are not associated with any IoT platform). Core AAM microservice provides credentials required to access symbIoTe Services and also supports trust relationships between platforms, as it acts as the root certification authority.

The PKCS12 keystore must have the following specifications:

* self-signed
* CA property enabled
* the following encryption parameters:
* SIGNATURE\_ALGORITHM=SHA256withECDSA
* CURVE\_NAME= prime256v1
* KEY\_PAIR\_GEN\_ALGORITHM=ECDSA
* Common Name (CN) value set according to AAMConstants.java field CORE\_AAM\_INSTANCE\_ID value (currently equal to string “SymbIoTe\_Core\_AAM” )
* certificate entry name: "symbiote\_core\_aam"

The steps below need to be followed to create the PKCS12 keystore core.p12:

* Create private key executing the following command:

openssl ecparam -name prime256v1 -genkey -out private-key.pem

* Create Certificate Signing Request (CSR) executing the following command:

openssl req -new -sha256 -key private-key.pem -out CSR.csr

* Create the certification executing the following command:

openssl x509 -req -days 3600 -in CSR.csr -signkey private-key.pem -out server.crt

Set ***CN = SymbIoTe\_Core\_AAM*** when is asked from the previous command.

* Creation of PKCS12 Keystore executing the following commands:

cat private-key.pem >server.pem

cat server.crt >> server.pem

openssl pkcs12 -export -in server.pem -out core.p12 -name paam -noiter -nomaciter

Copy the generated *core.p12* keystore file inside the *configuration* folder.

**1.4. Create necessary folders for your deployment**

Copy from *filesForCoreStackDeployment* folder available at confluence, the following files in to *configuration* folder:

* *AuthenticationAuthorizationManager folder,*
* *CoreConfigProperties folder,*
* *CoreInterface folder,*
* *bootstrap.properties file,*
* *nginx-ngrok.conf,*
* *nginx-prod.conf,*
* *nginx.conf.*

Fill in all the fields marked with *FILL ME* in the *TODO* section of the following files:

* *CoreConfigProperties/application.properties*
* *AuthenticationAuthorizationManager/bootstrap.properties*
* *bootstrap.properties*

Enter the *CoreConfigProperties* folder, make any changes if is necessary to the .properties files (e.g. change Mongodb, rabbitMQ credentials).

Note that the abovementioned procedure of this section ($1.4) has been based on the files available here: <https://github.com/symbiote-h2020/SymbioteCore/tree/master/resources/configuration> and cloning the CoreConfigProperties inside the configuration folder (Starting from release 3.1.0):

Inside CoreConfigProperties folder execute the following commands:

cd CoreConfigProperties

rm -r .git

git init

git config user.email you@example.com

git config user.name "Your User Name"

git clone https://github.com/symbiote-h2020/SymbioteCore.git configuration/CloudConfigProperties

The CoreConfigProperties folder has been entered to make any changes in the properties (e.g. rabbitMQ credentials) and commit the changes.

git add .

git commit -am "SymbIoTe Core configuration"

cd ..

**1.5. Create necessary docker volumes for your deployment**

Change directory so that the current directory is the *configuration* folder.

Run from *configuration* folder the following command to create the necessary docker volume to hold the CoreConfigProperties:

docker container run --rm -v $PWD/CoreConfigProperties:/source -v {docker stack name}\_symbiote-vol-config:/home/CoreConfigProperties -w /source alpine cp -r . /home/CoreConfigProperties/

As *{****docker stack name****}* use one that matches your project, thus a selected docker stack name such as *“****symbiote-core****”.*

Copy in *configuration* folder the docker-compose files copied from *filesForCoreStackDeployment* folder:

* *docker-compose-swarm-core.yml*
* *docker-compose-prod-swarm-core.yml*

Uncomment and configure the proxy settings (JAVA\_HTTP\_PROXY, JAVA\_HTTPS\_PROXY, JAVA\_SOCKS\_PROXY, JAVA\_NON\_PROXY\_HOSTS), in *docker-compose-swarm-core.yml* and *docker-compose-prod-swarm-core.yml* if you are behind proxy.

**1.6. Obtain necessary certification files for your deployment**

* Create **fullchain.pem** and **privkey.pem** certificate files as described in <https://github.com/symbioteh2020/SymbioteCloud/wiki/2.1-Configuration-of-NGINX#2111-obtaining-the-ssl-certificate> or by some other provider.

Note that you can also use the certbot command to create a certificate manually with dns validation:

certbot --manual certonly --preferred-challenges dns -d <domain name> --email <admin email>

In that case you will be asked to deploy a DNS TXT record under a specific name. After the creation of the certificates, letsencrypt informs the user about the location of the new files. This is the location where the files have to be copied from. The target location has to be synchronized with the nginx configuration. Normally, it is the nginx-certificates folder as will be described next,wherenginx is configured to find fullchain.pem and privkey.pem files.

* -Stay inside *configuration* folder and create the *nginx-certificates* folder.
* Copy the created **fullchain.pem** and **privkey.pem** certificate files to **nginx-certificates** folder (alternatively symbolic links can be used):

sudo cp /etc/letsencrypt/live/{your domain}/fullchain.pem nginx-certificates/

sudo cp /etc/letsencrypt/live/{your domain}/privkey.pem nginx-certificates/

sudo chown -R {user}:{group} nginx-certificates

**1.7. Start the deployment**

* Run: docker swarm init if the node is not a swarm manager. Note this command is only done in the first deployment when setting the docker swarm and not when redeploying the stack. We use the swarm mode so that secrets are encrypted during transit and at rest. Docker secrets are only available to swarm services and not to standalone containers.
* Run docker stack deploy -c docker-compose-swarm-core.yml -c docker-compose-prod-swarm-core.yml {docker stack name} to deploy the service stack. The ***{docker stack name*}** is the name of the service stack. It is the same defined in $1.5 and in our case we use *“****symbiote-core****”.*
* Run docker stack ls to list the stack and check the number of services used.
* Run docker image ls to check that all images have been created. It may take a while to pull all the images from DockerHub for the first time.
* Run docker service ls to list the services and check their status. Wait until the actual number of tasks (replicas) for each service is not 0.
* Run docker logs <service\_name> -f to get access to and follow the logs of a service. A service is ready when a message similar to 'Started <service\_name> in 105.045 seconds (JVM running for 112.933)' appears in the logs of the service.
* Run sudo service docker restart to restart the docker service stack if needed and repeat the steps.
* Run docker stack rm {docker stack name} to stop the application and remove the service stack. Services, networks, and secrets associated with the stack will be removed. Note that *{docker stack name*} is equal to “*symbiote-core” string in SHAPES project.*
* You can run docker swarm leave --force to leave the swarm (note that this is not required for redeploying the stack but only if you stop using the node for docker swarm).