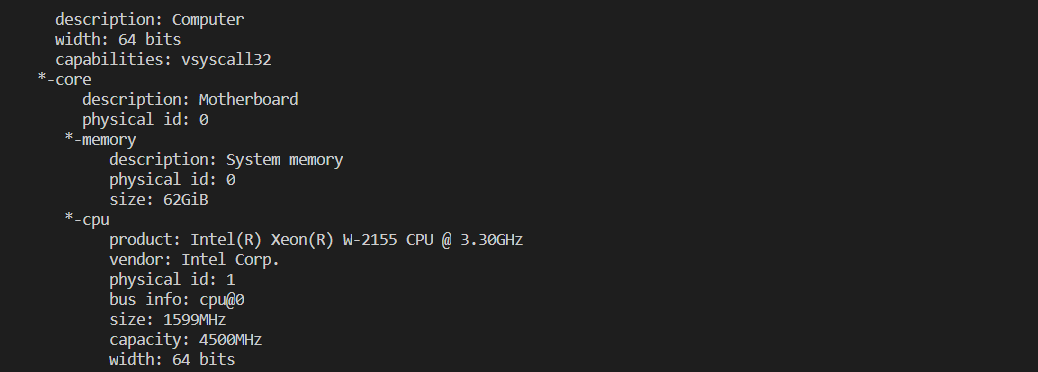
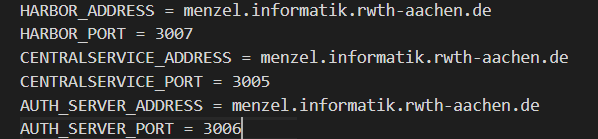
# Deployment Guidance of Station Software

**Requirements:**

* We assume that the **Docker Engine** is pre-installed on the machine. You can find the installation steps [here](https://docs.docker.com/install/). (On the Linux machines with low specs (1 core CPU, 1 GB RAM) it is possible to run Docker, but it will not be able to manage a high number of containers, etc.)   
  **Note**: The minimum requirements for the system specifications is hardly dependent on the algorithms that would be run on the machine and may vary from algorithm to algorithm. 
* There are some [system requirements](https://docs.docker.com/docker-for-windows/install/#:~:text=System%20Requirements) for windows machines:
  + Windows 10 64-bit: Pro, Enterprise, or Education (Build 16299 or later). For Windows 10 Home, see [Install Docker Desktop on Windows Home](https://docs.docker.com/docker-for-windows/install-windows-home/).
  + Hyper-V and Containers Windows features must be enabled.
  + The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10:
    - 64-bit processor with Second Level Address Translation (SLAT)
    - 4GB system RAM
    - BIOS-level hardware virtualization support must be enabled in the BIOS settings.
* We assume that the machine has **internet access** (At least the harbor registry, the central service and auth server need to be reachable. The addresses and the corresponding ports could be found in the provided .env file).

**Note**: Since the connections from the machine are **one-way**, there is no need to expose any ports of the machine to the internet.

****

* The **data** also needs to be accessible from the machine,
  + The data could be provided with [bind mounts](https://docs.docker.com/storage/bind-mounts/) (a file or directory on the *host machine* is mounted into a container – read-only access)
  + Also, the data could be supplied via an endpoint (For example a FHIR server API, <http://bruegel.informatik.rwth-aachen.de:9001/fhir/Patient>). This endpoint and needed credentials would be assigned as the [environment variable](https://docs.docker.com/engine/reference/commandline/run/#set-environment-variables--e---env---env-file)s to the train image by the station’s admin.
* The **.env file** contains the corresponding credentials for each station that will be provided by the PHT team to the station's admin.

## Enable Docker Engine API

Choose **one** of the following two steps.

**1.1 Modify Docker on the host**

Enable Docker Daemon to communicate using an HTTPS socket via Docker Bridge Network.

1. Generate signed certificates for Server and Client (follow <https://docs.docker.com/engine/security/https/> instructions, or use provided script:  
   (docker run --rm --entrypoint cat smithpht/station-software:latest /usr/src/app/tls\_generate\_certs.sh > “DESIRED\_PATH”/tls\_generate\_certs.sh)

$ . tls\_generate\_certs.sh && \_tls\_generate\_certs "ARBITRARY\_PATH"

1. Edit the file /lib/systemd/system/docker.service

$ sudo nano /lib/systemd/system/docker.service

1. Find and modify the line that starts with ExecStart (Add following line).

ExecStart= ... -H tcp://172.17.0.1:2376 --tlsverify --tlscacert=PATH\_TO\_CA\_CERT/ca.pem --tlscert=PATH\_TO\_SERVER\_CERT/cert.pem --tlskey=PATH\_TO\_SERVER\_KEY/key.pem

1. Save the modified file (Ctrl+O save modified and Ctrl+X quit).
2. Make sure the Docker service notices the modified configuration.

$ sudo systemctl daemon-reload

1. Restart the Docker service.

$ sudo service docker restart

1. Test that the Docker API is indeed accessible

$ curl [https://172.17.0.1:2376/version --cert PATH\_TO\_CLIENT\_CERT/cert.pem](https://172.17.0.1:2376/version%20--cert%20PATH_TO_CLIENT_CERT/cert.pem) --key PATH\_TO\_CLIENT\_CERT/key.pem --cacert PATH\_TO\_CLIENT\_CERT/ca.pem

* 1. **Use Docker in Docker (dind) image (**Recomended**)**

Run Docker-in-Docker

1. Create a network

$ docker network create pht-dind-net

1. Pull and run dind image

$ docker run --privileged --name pht-dind --network pht-dind-net --network-alias docker -d -e DOCKER\_TLS\_CERTDIR=/certs -v pht-dind-certs-ca:/certs/ca -v pht-dind-certs-client:/certs/client smithpht/dind:stable-dind

## Create and Start MongoDB Container

1. Run docker command to create MongoDB container (Specify container name “pht-mongo”)

$ docker run -d --name=pht-mongo mongo

By default, the container is connected to the default bridge network “docker0” and listening on the port “27017”.

2. Check the assigned IP address of MongoDB container

$ docker inspect pht-mongo

Find IP address in “NetworkSettings” -> “Network” -> “bridge” -> “IPAdress” (here is “172.17.0.2”)



## Create a Database on MongoDB and Admin User

1. Using mongo-client to connect MongoDB ($ apt install mongodb-clients)

$ mongo "mongodb://172.17.0.2:27017"

2. Create database called “pht”

$ use pht

3. Create admin account of database

$ db.createUser({

user: "admin",

pwd: "admin",

roles: ["readWrite", "dbAdmin"],

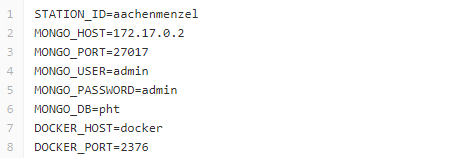
mechanisms:["SCRAM-SHA-1"]

})

4. Exit from mongo-client

$ exit

5. Check the following parameters in .env file. We send you this .env file.



## Build Station web software as Docker Image

1. Pull and run PHT station software image (May also need to change some variables in provided **env file** corresponding to each station, like MONGO\_HOST, MONGO\_USER, DOCKER\_HOST,…)

$ docker run --name pht-web -d -p 3030:3030 -v pht-dind-certs-client:/usr/src/app/dind-certs-client/certs:ro --env-file <PATH\_TO\_ENV\_FILE> smithpht/station-software:latest

**Note**: if you generated the certificates manually, replace “pht-dind-certs-client” with “<PATH\_TO\_CLIENT\_CERT>”

**Note**: if you did not use docker network, change **DOCKER\_HOST** value to corresponding address of Docker API.

1. Connect “pht-web” container to “pht-dind-net” network

$ docker network connect pht-dind-net pht-web