GTSL - Project Setup Guidelines

This page provides the guidelines for setting up the Global Trust Service Status List (GTSL) project that is part of the Future Trust project.

Prerequisites

Tool	Version	Host	Description
Java	1.8	Local	programming language
Git	latest	Local	versioning control system
Maven	latest	Local	software project management and comprehension tool
Docker	latest	Local	open platform to build, ship, and run distributed applications
Docker-compose	latest	Local	tool for defining and running multi-container Docker applications
Docker-machine	latest	Local (required only on Windows & Mac OS)	tool that lets you install Docker Engine on virtual hosts

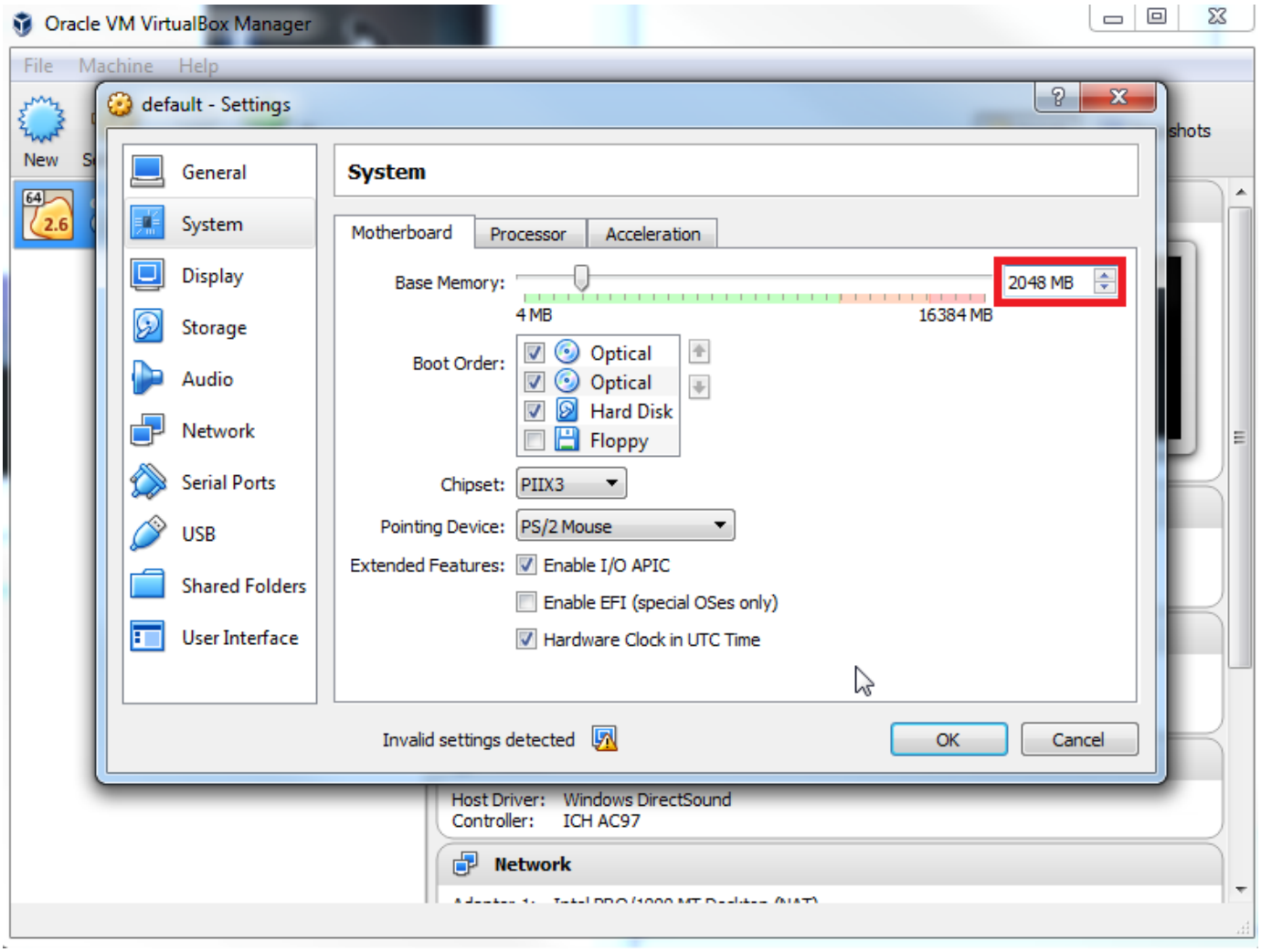
Warning

On Windows and Mac OS, you have to clone all the repositories (provided later in the document) wherever you want **in your HOME**. The docker feature which allows you mounting volumes only works in your HOME.

If you are using *docker-machine* (required on Windows and Mac OS while using *Docker Toolbox*):

- make sure to **always use the *Docker Quickstart Terminal*** (provided with the *Docker Toolbox);*
- make sure that your virtual machine has a **minimum 2 Go** of RAM, see explanations below.

If you are using Docker Toolbox, then open VirtualBox, shutdown the docker-machine (should be **default**), then right-click on it, Settings... System and update the memory value (see the screenshot below).



Build smart-contracts

> **If you don't plan to modify the Solidity smart-contracts, please skip this part.**

First of all, clone the repository of the project on your machine :

Bash

```
git clone
https://<username>@gitlab.arhs-developments.com/FutureTrust/gtsl-ethereum.
git
cd gtsl-ethereum
```

Compile Solidity sources and generate Java classes

Building smart-contracts is the process that allows us to generate the Java classes corresponding to the Solidity contracts defined in *src/main/soli dity/contracts*.

Warning: Java files which will be generated are not versioned by Intellij IDEA, you have to add them to Git manually using the *git add* command.

For building smart-contracts, simply use the following command :

Bash

```
./src/main/scripts/build.sh
```

> ***TEMPORARY:*** *For the moment, this project cannot be used as a Maven dependency in the main project* ***gtsl*** *because the Nexus is not available yet. So, in order to use the generated Java files in the main project please Copy/Paste Java files in the* ***gtsl*** *project.*

Clean containers, images and temporary files

You can clean the project by running the following command :

Bash

```
./src/main/scripts/clean.sh
```

This command removes the temporary files, containers and the images used in the build process.

Run the dependencies

First of all, clone the repository of the project on your machine :

Bash

```
git clone
https://<username>@gitlab.arhs-developments.com/FutureTrust/gtsl-docker.git
cd gtsl-docker
```

Manage your Ethereum account

Before starting the application, you have to manage your Ethereum account.

- If you already have an account, you can use it by copying the keystore file into **src/main/docker/ethereum/data/keystore/.**
- If you do not have an account the script will generate one for you.

You have to provide the password of the account in order to load your credentials in the application.

To do this, create a file named **password** into **src/main/docker/ethereum/src/** and write the password of your account **on the first line as a plain text with no blanks**.

- If you already have an account, the password must be the password of the account you provide.
- If you do not have an account, the password will be used to generate a new account and will be the password of the account.

Note: For development, use a trivial password ("123456" for instance), the application will never ask you for this password.

Warning: Beware not to commit your password or keystore on the git repository, even if those files are part of the *.gitignore*.

Run the starting script

If you are using *docker-machine* (which should be the case on Windows and Mac OS), pass in as arguments the machine name (if you are using Docker Toolbox the machine name should be **default**). The endpoints will be reached through the *docker-machine*.

If you don't use *docker-machine*, don't pass any arguments. The endpoints will be reached on *localhost*.

The following command will run the dependencies.

```
Bash
./src/main/scripts/start.sh <docker-machine>
```

You have to wait for containers are ready, it should take approximately 10 minutes.

When the dependencies are ready, you have to Copy/Paste the files located in the *outputs* directory into the **gtsl** project. How to do this is described in the next part "Start the application".

Note: If you already did the configuration above and you run the starting script again, the script will use the keystore which was generated previously. It means that you don't have to Copy/Paste outputs files every time you run the starting script.

Clean containers and images

Warning: This command will remove all containers, even if they are running, including the containers not related to the project.

You can clean the project by running the following command :

```
Bash
./scripts/clean.sh
```

Start the application

First of all, clone the repository of the project on your machine :

```
Bash
git clone
https://<username>@gitlab.arhs-developments.com/FutureTrust/gtsl.git
cd gtsl
git checkout develop
```

Before starting the application, you need to provide your keystore and your configuration files. Those files had been generated in the previous part "Run the dependencies" and are located in the *outputs* directory in the **gtsl-docker** project. Copy and paste, the keystore and the configuration files into **gtsl-web/src/main/resources** in the **gtsl** project.

Then, you can start the application using the following command.

```
Bash
./scripts/start.sh
```

The Spring Boot application should be running.

The application is running on http://localhost:8081/.

Troubleshooting

To know if containers work well, use the *docker logs* command :

Bash

```
docker logs <container>
```

List of containers :

- ipfs-node
- ethereum-node