Human Brain Project

THE MEDICAL INFORMATICS PLATFORM

MIP TECHNICAL SPECIFICATIONS

&

STEP-BY-STEP SOFTWARE DEPLOYMENT GUIDE

VERSION: JANUARY 2019

Table of Contents

List of Figures

List of Tables

1. Introduction

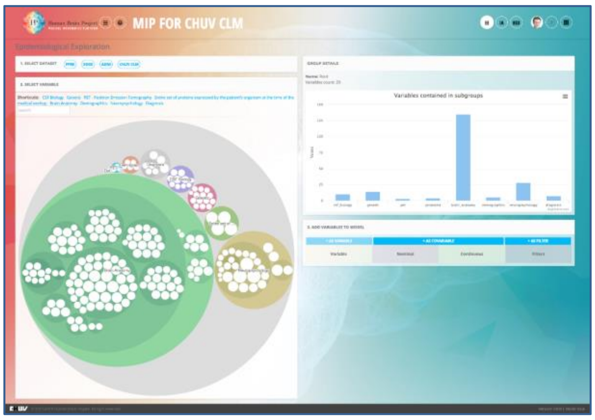
Convergence of biology and technology and increasing capability to:

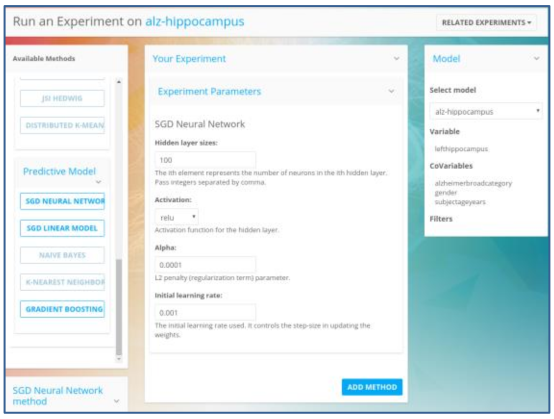
- assess pan-omic biological data of an individual, including detailed brain features (morphology, connectivity, functionality), DNA sequence analysis, proteome, metabolome, microbiome, physiome and phenome, and
- correlate biological data with sign and symptoms, observational data and diagnostic

provide the opportunity to discover new biological signatures of diseases, develop preventive strategies and improve medical treatment. Opportunities to use these data to improve health outcomes – to develop preventive strategies and improve medical care – is the motivation for the development of the Medical Informatics Platform.

The Medical Informatics Platform (MIP) provides expert tools for:

- Data processing
- Data exploration and selection, and
- Analysis of patients biological and other health-relevant data distributed across remote locations in hospitals, research institutes and universities




Run an Experiment on alz-hippocampus

Your Experiment — Experiment Parameters

SGD Neural Network

Hidden layer sizes: 100
The ith element represents the number of neurons in the ith hidden layer. Pass integers separated by comma.

Activation: relu
Activation function for the hidden layer.

Alpha: 0.0001
L2 penalty (regularization term) parameter.

Initial learning rate: 0.001
The initial learning rate used. It controls the step-size in updating the weights.

Model — Select model: alz-hippocampus
Variable: lefthippocampus
CoVariables: alzheimerbroadcategory, gender, subjectageyears
Filters

Figure 1 – Medical Informatics Platform Web Portal

The use of a large volume and variability of normalised patient data provides for more rapid advancement in understanding neurological and psychiatric diseases. This will, in turn, allow identification of the associated biological mechanisms and open possibilities for prevention, early diagnosis and personalised medicine.

The Medical Informatics Platform has three key objectives:

1) Provide analytical tools for in situ clinical data analysis and federation of results
2) Recruit hospitals, research institutes and universities to contribute to and benefit from using the platform
3) Develop tools for extracting biological signatures of diseases from multi-level pan-omic data

The MIP provides methods for federated analysis of patient data from hospitals, research centres and biobanks. Clinical scientists can develop, share and release results of their research. The MIP aims to bring together people across professional and scientific fields encourages them to actively contribute to the design and development of the services which the MIP provides.

The users of the MIP are:

- **Clinicians**, for objective diagnoses and treatment of brain disease
- **Neuroscientists**, for the application and testing of new models and methods
- **Pharmaceutical or biotech researchers**, for disease target discovery

2. Hybrid Deployment Model

The MIP is distributed, cloud-ready patient data analysis ecosystem, which connects patients' data from hospitals and research cohort datasets and provides set of pre-integrated statistical methods and predictive machine learning algorithms for patient data exploration, data modelling, integration and execution of experiments (data analysis methods), and visualisation of the results.

The ~~platform~~Platform makes data on populations of patients broadly available for research use, by providing software-as-a-service to clinicians, neuroscientists and epidemiologists both for diagnosis and research in clinics and for collaborative neuroscience research using hospital data and open patient research cohort datasets.

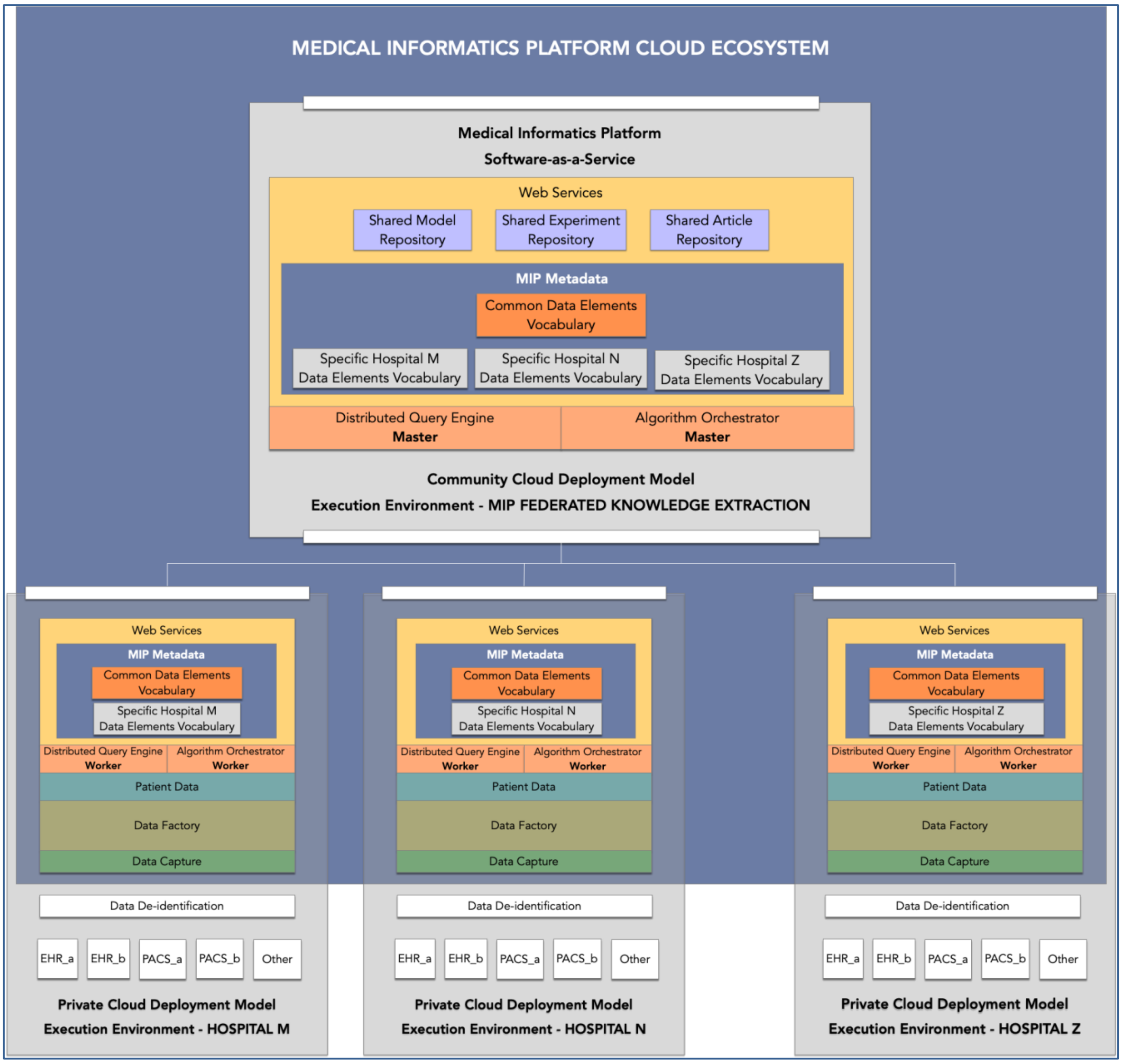

MEDICAL INFORMATICS PLATFORM CLOUD ECOSYSTEM
Medical Informatics Platform
Software-as-a-Service
Web Services
Shared Model Repository
Shared Experiment Repository
Shared Article Repository
MIP Metadata
Common Data Elements Vocabulary
Specific Hospital M Data Elements Vocabulary
Specific Hospital N Data Elements Vocabulary
Specific Hospital Z Data Elements Vocabulary
Distributed Query Engine Master
Algorithm Orchestrator Master
Community Cloud Deployment Model
Execution Environment - MIP FEDERATED KNOWLEDGE EXTRACTION
Web Services
MIP Metadata
Common Data Elements Vocabulary
Specific Hospital M Data Elements Vocabulary
Distributed Query Engine Worker
Algorithm Orchestrator Worker
Patient Data
Data Factory
Data Capture
Data De-identification
EHR_a
EHR_b
PACS_a
PACS_b
Other
Private Cloud Deployment Model
Execution Environment - HOSPITAL M
Specific Hospital N Data Elements Vocabulary
Execution Environment - HOSPITAL N
Specific Hospital Z Data Elements Vocabulary
Execution Environment - HOSPITAL Z

Figure 2 – Medical Informatics Platform Deployment Model

Figure 2 illustrates the cloud-ready MIP federated knowledge extraction software-as-a-service deployed in community execution environment. The MIP community execution environment provides advanced multi-dataset, cross-centre descriptive~~,~~ and predictive analytics. It runs software that orchestrates the execution of statistical and machine-learning algorithms in private hospital MIP execution environments and aggregates the results. The algorithms are executed locally, in private hospital environments where the ~~patient~~ de-identified patient data is stored. Master orchestrator components that are running in community execution environment, connected to the distributed private MIP execution environments via web services, fetch the aggregate results of the algorithms executed in the private execution environments and aggregate them in a cross-centre data analysis result.

The MIP is engineered with a privacy by design approach. De-identified patient data stored in private hospital's execution environments are accessible only locally, either by the algorithms running there or by other means of data exploration within the private execution environment, using the locally deployed web services.

Users of the MIP can access the community execution environment or the local private hospital execution environment through the MIP web portal (Figure 1). The MIP web applications allow for the statistical/aggregated (not individual) data exploration, selection of data types for analytics, execution of algorithms/experiments and visualisation of the results.

The hybrid community and private hospital deployment model, microservice architecture coupled with continuous integration and continuous deployment technology, distributed hospital patient data storage and federated algorithm execution ~~is a~~are software architecture-related prerequisites to having a cross-centre data analytics.

This distributed, patient privacy preserving software architecture is a necessary but not a sufficient condition to having multi-dataset clinical studies. Hospital datasets have overlapping data types but different ontological representations. Data is described, stored and formatted in different data structures. For executing a multi-dataset analytics, data models need to be harmonised in a common MIP data model, which is shared and synchronised between the distributed private hospital instances and community execution environment.

The data model harmonisation is, therefore, a key technology enabler for cross-centre multiple dataset clinical studies. It is a well-defined process supported by the workflow orchestration, application ontology software architecture, and the organisation, which establishes and maintains the rules and controls the quality and the integrity of the data harmonisation process.

Data governance and data selection (DGDS) committee is a centrally coordinated MIP organisational entity responsible for establishing and maintaining data governance methodology and data harmonisation rules. The members of the DGDS committee are MIP software architects and data managers from the MIP R&D team, supported by ~~members of~~ the expert medical/scientific committee consisting of the medical doctors, clinical researchers and data managers of the participating centres.

Data harmonisation and re-harmonisation is an on-going process. With the introduction of a new dataset, the whole process has to be repeated, starting with the analysis of the incoming dataset ending with the synchronisation of (re-)harmonised data models across the distributed MIP ecosystem.

The distributed, privacy-preserving MIP software deployed across hospitals and institutes using a hybrid community-private deployment model with centralised ~~orchestartion~~ orchestration of statistical inference and machine learning algorithms, and managed harmonisation and synchronisation of the data model provide IT prerequisites for execution of cross-centre, multi-datasaet clinical studies across the participating centres.

For example, using the unsupervised machine learning on patients' data in one or several centres to train a classifier which differentiates between the frontotemporal dementia and Alzheimer's disease, then applying learned classifier to patients' data in other participating centre for a differential diagnosis between the two neurodegenerative disorders. Or, using the clinical and pathological data of deceased patients from available datasets to train a machine-learning model that can be used to predict the disease progression with patients in other hospitals.

A centre hosting the MIP has to provide two servers, install required middleware and set up the network ensuring the adequate level of security. The MIP is designed using privacy-by-design approach and it contributes to data protection by allowing a remote access only to irreversibly anonymised patients' data. For maximising data security, it is data centre responsibility to design, implement and operate an appropriate data centre networking.

3. Hardware Requirements

MIP requires configuration and installation of the following two nodes in a participating centre's private execution environment, each on a single server or on a cluster of servers:

- **Local MIP Node** – a server designed for working with pseudonymised clinical data
- **Local MIP Federation Node** – a server designed for working with irreversibly anonymised data

The first server, a local MIP node, is an instance of the MIP containing:

- Pseudonymised patient data capture sub-system
- Data factory sub-system with pipelines for patient data and MRI processing and normalisation
- Feature data store for storing normalised pseudonymised patient biological and other health-relevant features
- Knowledge extraction sub-system with distributed query engine, algorithm factory and algorithm orchestrator
- Web-based front-end used by authorised clinicians to perform local data analysis

The second server, a local MIP federation node, is an instance of the MIP designed to secure full patients' privacy by allowing a remote access only to irreversibly anonymised data. It receives normalised and irreversibly anonymised patient data from the local MIP node, does not perform heavy data pre-processing, but may execute computationally intensive machine learning algorithms.

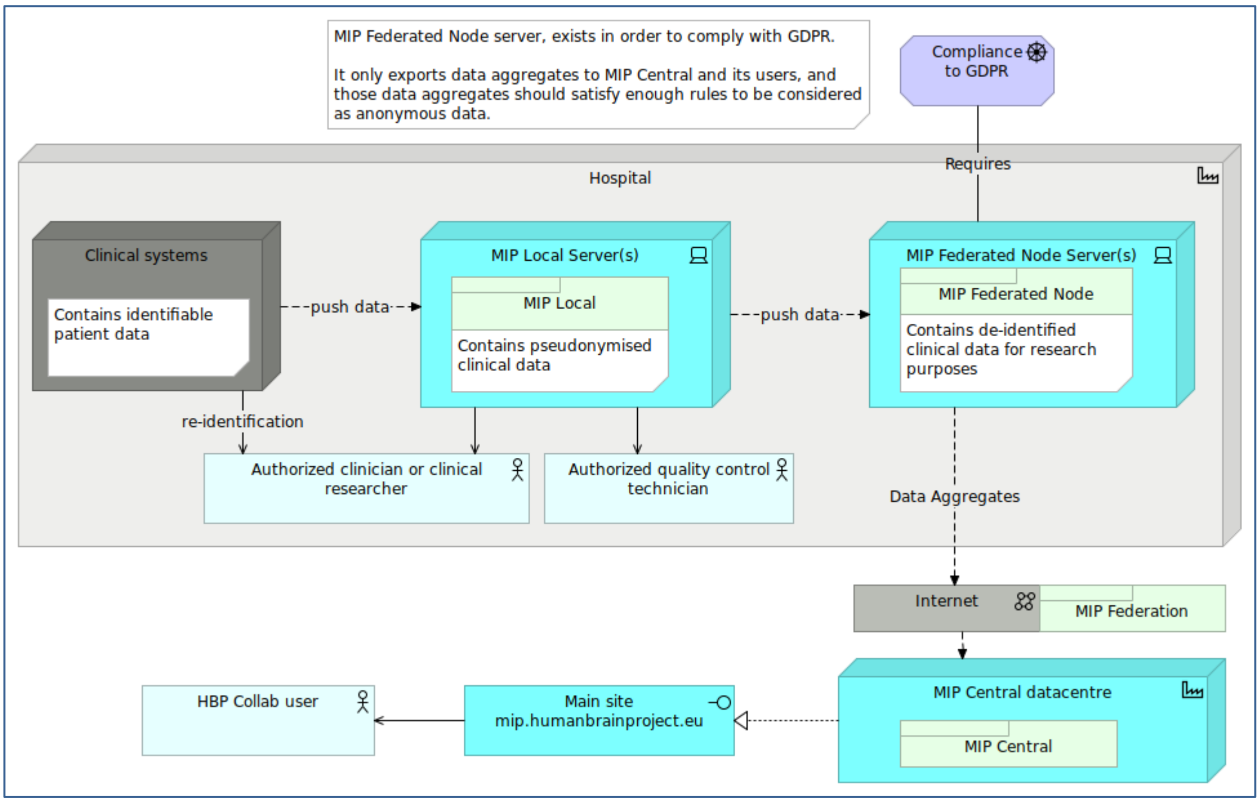

MIP Federated Node server, exists in order to comply with GDPR.
It only exports data aggregates to MIP Central and its users, and those data aggregates should satisfy enough rules to be considered as anonymous data.
Compliance to GDPR
Requires
Hospital
Clinical systems
Contains identifiable patient data
push data
MIP Local Server(s)
MIP Local
Contains pseudonymised clinical data
push data
MIP Federated Node Server(s)
MIP Federated Node
Contains de-identified clinical data for research purposes
re-identification
Authorized clinician or clinical researcher
Authorized quality control technician
Data Aggregates
Internet
MIP Federation
MIP Central datacentre
MIP Central
HBP Collab user
Main site mip.humanbrainproject.eu

Figure 3 - MIP Deployment in a Private (Hospital) Execution Environment

The MIP does not provide any feature for identifying individual patients. It is the responsibility of the participating centre to secure a look up mechanism alongside the local MIP node for mapping patient pseudonymes with their real identifiers.

Table 1 - Hardware Requirements Specification

MIP Hardware Requirements

MIP Node can be installed on a single server of on a cluster of servers

Local MIP Node

The following scenarios are supported:

- **On-site pre-processing of MRI scans or EEGs**
  (backlog of up to 500 patients, daily processing of about a dozen patients)
  - **One 'Large Processing Server' or at least two 'Medium Processing Servers'**
- **No pre-processing of MRI scans or EEGs or in the low hundreds of patients**
  - **One 'Medium Processing Server'**

The exact specifications may change depending on the volume and nature of pre-processing operations. To support additional computational requirements, it is always possible to add more processing nodes in the cluster

Name	CPU	Memory	Disk	Network Zone
Large Processing Server	x64 12 – 16 cores	32GB+	**Local Disk**: 200 GB **Shared Disk**: 11 times the expected size of the provided imaging data	Research Network
Medium Processing Server	x64 8 cores	16GB+	**Local Disk**: 200 GB **Shared Disk**: 11 times the expected size of the provided imaging data	Research Network

Local MIP Federation Node

This node received de-identified and pre-processed data from MIP Local, it does not perform any heavy data pre-processing but may execute computationally intensive machine learning algorithms.
The exact specifications may change depending on the volume of data and nature of machine learning algorithms. To support additional computational requirements, it is always possible to add more processing nodes in the cluster. GPU is not supported at the moment.

Name	CPU	Memory	Disk	Network Zone
Medium Analytics Server	x64 8 cores	16GB+	**Local Disk**: 500 GB	Research Network or DMZ

4. Software Requirements

The local MIP node runs on Linux operating system supporting Docker technology. Installation of the software components specified in Table 2 are required for automatic deployment of MIP software packaged in Docker images using Ansible installation script.

Table 2 - Software Requirements Specification

MIP Software Requirements		
Local MIP Node		
Operating System	**Connectivity**	**Licenses**
Ubuntu 16.04 LTS RedHat Enterprise Linux 7.2+ CentOS 7.2+	OpenSSH	MATLAB 2016b
Local MIP Federation Node		
Operating System	**Connectivity**	**Licenses**
Ubuntu 16.04 LTS RedHat Enterprise Linux 7.2+ CentOS 7.2+	OpenSSH	N/A

MATLAB 2016b license is needed by the Statistical Parametric Mapping SPM12 tool, a component orchestrated by the neuromorphometric processing pipeline situated in the data factory sub-system of the local MIP node. Local MIP federation node is not used for neurommorphometric data processing; hence it does not need MATLAB license.

5. Connectivity Settings

Centre that hosts MIP nodes have to provide two servers, configure the adequate operating system and connectivity software as well as to setup network connectivity and provide an adequate level of security.

5.1 Security Recommendation

Local MIP node local MIP federation nodes should not be placed in the clinical network security zone. Any connectivity from the local MIP and local MIP federation nodes through the firewalls into the clinical network security zone should be prevented.

It is also recommended not to place local MIP federation node in the network security zone of the local MIP node. Any connectivity from the local MIP federation node, through the firewalls, into the security zone of the local MIP node should be prevented.

5.2 Network Service and Connectivity

Configuration of the network services and opening of L3 ports specified in Table 3 and Figure 4 is the last step of private execution environment preparation for installation of local MIP and local MIP federation nodes.

Table 3 - Network Service and Connectivity Requirements

MIP Network Service and Connectivity Requirements		
Local MIP Node		
Subnet	Configure a dedicated sub-network for the node	
DNS	Addressable from the Internet for the server, incl. web portal	
Firewall	Prevent access from the node to the clinical network security zone Allow outgoing connections to other services on Internet (see chapter 5.3) Allow outgoing connections to local MIP federation node Allow incoming connections to the ports below (see ingress connectivity)	
<u>Ingress Connectivity</u>		
Port #	**Protocol**	**Traffic Type**
22	**SSH**	Pushing data from the clinical systems to local MIP node Remote administrative access for the CHUV HBP MIP team
80	**HTTP**	Used by Let's Encrypt CA to setup its SSL certificate Used by MIP for health tests
443	**HTTPS**	MIP Web Portal Connections only from locally authorised users
Local MIP Federation Node		
Subnet	Configure a dedicated sub-network for local MIP federation node	
DNS	Addressable from the Internet for the server, incl. web portal	
Firewall	Prevent access from the node to the clinical network security zone Prevent access from the node to the local MIP node's network zone Allow outgoing connections to other services on Internet (see chapter 5.3) Allow incoming connections from the central MIP server to the ports below	
<u>Ingress Connectivity</u>		
Port #	**Protocol**	**Traffic Type**
22	**SSH**	Pushing data from local MIP node to local MIP federation node Remote administrative access for the CHUV HBP MIP team
80	**HTTP**	Used by Let's Encrypt CA to setup its SSL certificate Used by MIP for health tests
443	**HTTPS**	MIP federated API Connections only from local users and the central MIP server

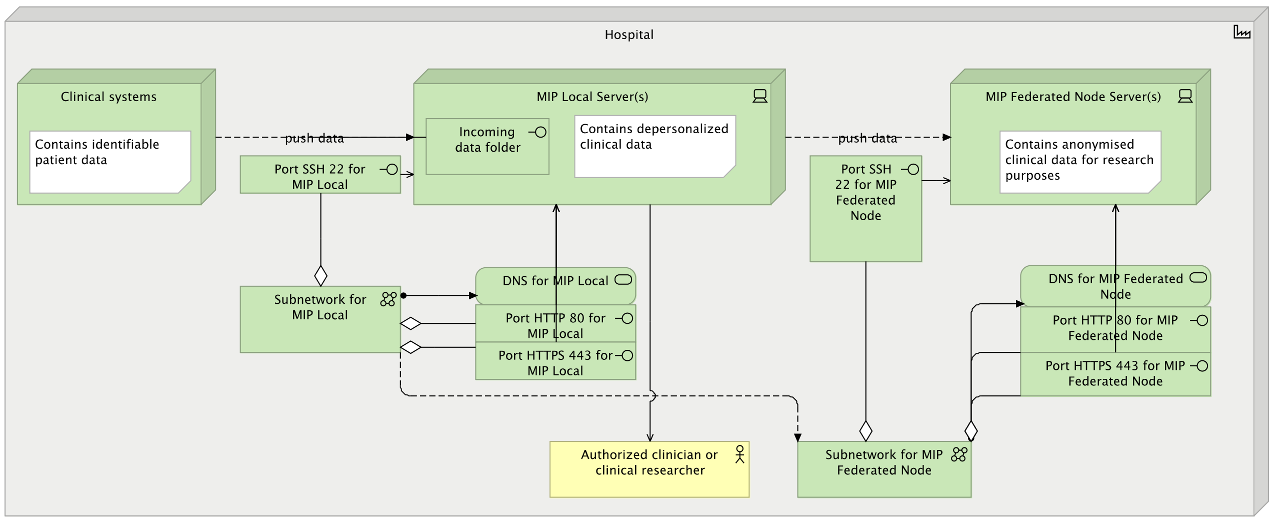

Hospital
Clinical systems
Contains identifiable patient data
push data
Port SSH 22 for MIP Local
MIP Local Server(s)
Incoming data folder
Contains depersonalized clinical data
push data
Port SSH 22 for MIP Federated Node
MIP Federated Node Server(s)
Contains anonymised clinical data for research purposes
Subnetwork for MIP Local
DNS for MIP Local
Port HTTP 80 for MIP Local
Port HTTPS 443 for MIP Local
DNS for MIP Federated Node
Port HTTP 80 for MIP Federated Node
Port HTTPS 443 for MIP Federated Node
Authorized clinician or clinical researcher
Subnetwork for MIP Federated Node

Figure 4 - Private Execution Environment Connectivity Diagram

5.3 External Services

The external services that must be accessible from any MIP dedicated machines for installation, configuration, update, and maintenance purposes are listed in Table 4.

Table 4 - External Service Specification

Name	Description	IP/Hostname	L3P>PORT>L7P>IN/OUT	Required by
Remote Access SSH	Only inbound service required. It is mainly used by our automated deployment scripts and by our deployment and support team to connect the machine using the provided VPN access.	Inside Clinical Network (VPN)	**TCP>22>SSH>IN**	**Local MIP**
Web Portal	Main entry-point for MIP local users	Inside Clinical Network	**TCP>80>HTTP>IN**	**Local MIP**
Ubuntu French APT Servers	Ubuntu France’s official APT server	fr.archive.ubuntu.co security.ubuntu.com	**TCP>443>HTTPS>OUT**	**Local MIP on Ubuntu**
Docker APT	Docker’s official APT server	download.docker.com	**TCP>443>HTTPS>OUT**	**Local MIP on Ubuntu**

Name	Description	IP/Hostname	L3P>PORT>L7P>IN/OUT	Required by
Docker YUM	Docker's official YUM server	yum.dockerproject.org	**TCP>443>HTTPS>OUT**	**Local MIP on RHEL**
Launchpad	Binary repository containing Ansible up-to-date versions	launchpad.net	**TCP>443>HTTPS>OUT**	**Local MIP**
Mesosphere APT	Mesosphere's official APT server	repos.mesosphere.com	**TCP>443>HTTPS>OUT**	**Local MIP on Ubuntu**
PyPI	Python package repository	pypi.python.org	**TCP>443>HTTPS>OUT**	**Local MIP**
Docker Hub	Docker Hub (To be replaced by our own private Docker registry)	hub.docker.com	**TCP>443>HTTPS>OUT**	**Local MIP**
MATLAB License Server	Required only if the institution uses its own MATLAB licence server		**TBD**	**Local MIP**
GitHub	Main source repository for SP8's software	github.com	**TCP>443>HTTPS>OUT**	**Local MIP**
Bitbucket	Private source repository for deployments configurations	bitbucket.org	**TCP>443>HTTPS>OUT**	**Local MIP**
CHUV Server	Private git repository/Docker registry	hbps1.chuv.ch	**TCP>443>HTTPS>OUT**	**Local MIP**

6. Software Deployment Package

The MIP is deployed on Mesos stack with added support for automated deployment/upgrade of services managed by Mesos Marathon and hardened security of the Ubuntu operating system. The services are built using Ansible scripts, unifying operation system configuration, middleware and application software deployment. Automated installation and configuration of MIP software on bare metal or preconfigured virtual machines supports clustering, security and monitoring.

HBP MIP deployment software package consists of third-party (Table 5) and HBP MIP software (Table 6). Participating centres are required to install them on local MIP servers.

Table 5 - Third-party Software

Name	Level	License	Deployment	Required by
OpenSSH	Infrastructure		Native	Local MIP
Java Open JDK SE	Infrastructure		Native/Container	Local MIP
Python2.7	Infrastructure		Native	Local MIP
Python3	Infrastructure		Native/Container	Local MIP
R	Infrastructure		Container	Local MIP
Docker	Infrastructure		Native	Local MIP
Mesos	Infrastructure		Native	Local MIP
Zookeeper	Infrastructure		Native	Local MIP
Chronos	Infrastructure		Container	Local MIP
Marathon	Infrastructure		Native	Local MIP
Git	Infrastructure		Native	Local MIP
docker_py	Infrastructure		Native	Local MIP
python-simplejson	Infrastructure		Native	Local MIP
Træfik	Infrastructure		Native/Container	Federated MIP
Consul	Monitoring/Security		Native/Container	Federated MIP
Logwatch	Monitoring/Security		Native	Local MIP
fail2ban	Monitoring/Security		Native	Local MIP
ufw	Monitoring/Security		Native	Local MIP
PostgreSQL	Everywhere		Container	Local MIP
slackclient_py	Data Factory		Native	Optional
Airflow	Data Factory		Native	Local MIP

Name	Level	License	Deployment	Required by
MATLAB	Data Factory	Proprietary	Native	Local MIP
SPM12	Data Factory		Native	Local MIP
Spring Framework	Web Portal		Container	Local MIP
Flyway	Web Portal		Container	Local MIP
Nginx	Everywhere		Container	Local MIP

Software packages listed in Table 6 have been developed by the HBP MIP development teams in the scope of the Human Brain Project's sub-project 8.

Table 6 - HBP MIP Software

Name	Level	License	Deployment	Required by
data-tracking	Data Factory		Container	Local MIP
i2b2-import	Data Factory		Container	Local MIP
i2b2-setup	Data Factory		Container	Local MIP
data-catalog-setup	Data Factory		Container	Local MIP
hierarchizer	Data Factory		Container	Local MIP
mri-preprocessing-pipeline	Data Factory		Container	Local MIP
airflow-imaging-plugins	Data Factory		Container	Local MIP
data-factory-airflow-dags	Data Factory		Container	Local MIP
MIPMap	Hospital Database Bundle		Container	Local MIP
PostgresRAW	Hospital Database Bundle		Container	Local MIP

Name	Level	License	Deployment	Required by
PostgresRAW-UI	Hospital Database Bundle		Container	Local MIP
Exareme	Hospital Database Bundle		Container	MIP Federated
woken	Algorithm Factory		Container	Local MIP
base-docker-images	Algorithm Factory		Container	Local MIP
python-base-docker-images	Algorithm Factory		Container	Local MIP
functions-repository	Algorithm Factory		Container	Local MIP
Label Propagation Framework	Algorithm Library		Container	MIP Federated
Exareme mip-algorithms	Algorithm Library		Container	MIP Federated
hbpjdbcconnect	Algorithm Library		Container	Local MIP
hbplregress	Algorithm Library		Container	Local MIP
hbpsummarystats	Algorithm Library		Container	Local MIP
CCC	Algorithm Library		Container	Federated MIP
jsi-functions	Algorithm Library		Container	Federated MIP
bhtsne	Algorithm Library		Container	Federated MIP
Rtsne	Algorithm Library		Container	Federated MIP
portal-backend	Web Portal		Container	MIP-Local
portal-frontend	Web Portal		Container	MIP-Local

7. MIP Software Deployment

This chapter provides a step-by-step guide for the MIP software deployment. The steps describMed in the GitHub project documentation are described in somewhat more details The software deployment scripts referred here are available on the GitHub at:

https://github.com/HBPMedical/mip-microservices-infrastructure.

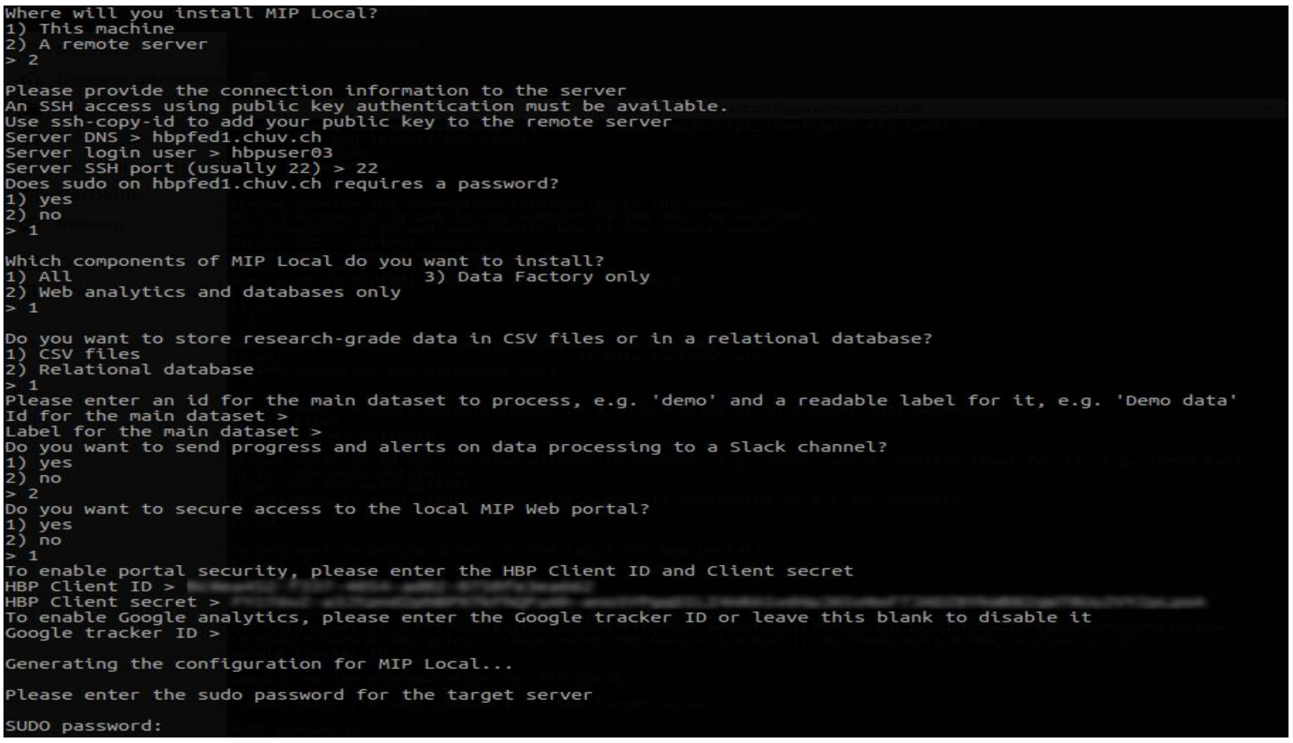
7.1 Step 1 - Clone the Project

Create git project based on the mip-microservices-infrastructure project v2.5.3 (abb98e9).

7.2 Step 2 - Generate a Software Configuration

Run the following configuration script to generated some configuration files:

```
./common/scripts/configure-mip-local.sh
```

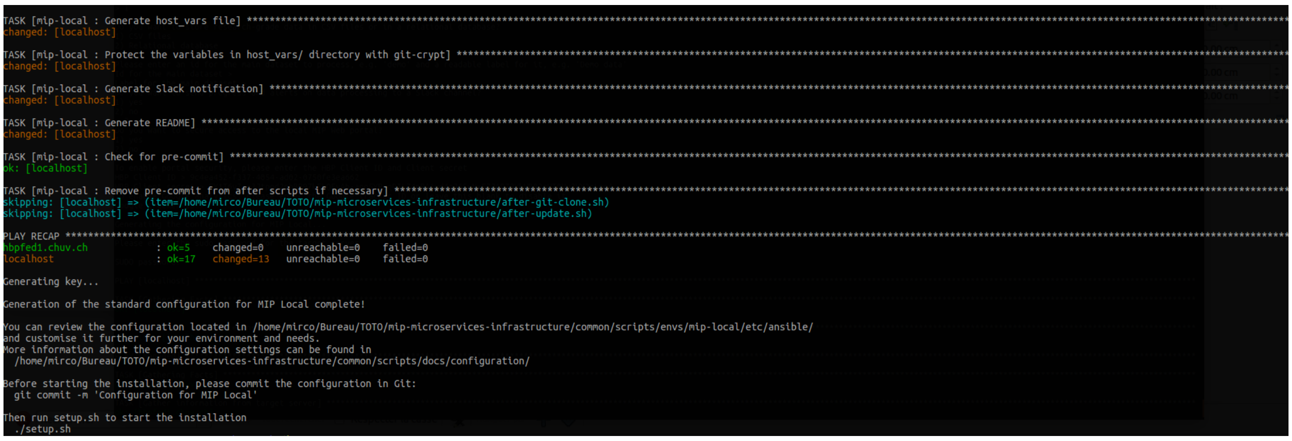


```
Where will you install MIP Local?
1) This machine
2) A remote server
> 2

Please provide the connection information to the server
An SSH access using public key authentication must be available.
Use ssh-copy-id to add your public key to the remote server
Server DNS > hbpfed1.chuv.ch
Server login user > hbpuser03
Server SSH port (usually 22) > 22
Does sudo on hbpfed1.chuv.ch requires a password?
1) yes
2) no
> 1

Which components of MIP Local do you want to install?
1) All                         3) Data Factory only
2) Web analytics and databases only
> 1

Do you want to store research-grade data in CSV files or in a relational database?
1) CSV files
2) Relational database
> 1
Please enter an id for the main dataset to process, e.g. 'demo' and a readable label for it, e.g. 'Demo data'
Id for the main dataset >
Label for the main dataset >
Do you want to send progress and alerts on data processing to a Slack channel?
1) yes
2) no
> 2
Do you want to secure access to the local MIP Web portal?
1) yes
2) no
> 1
To enable portal security, please enter the HBP Client ID and Client secret
HBP Client ID > [redacted]
HBP Client secret > [redacted]
To enable Google analytics, please enter the Google tracker ID or leave this blank to disable it
Google tracker ID >

Generating the configuration for MIP Local...

Please enter the sudo password for the target server

SUDO password:
```

Note that if you enable the portal security (in order to use the HBP credentials to log in to the MIP), you'll be asked to provide HBP Client ID and a HBP Client secret. Those values must be provided by the LREN-CHUV team.

In order to install the research datasets, you'll have to provide us with your GitLab account so that we'll be able to invite you to join those datasets projects as a guest. We can provide you with a generic account if needed.

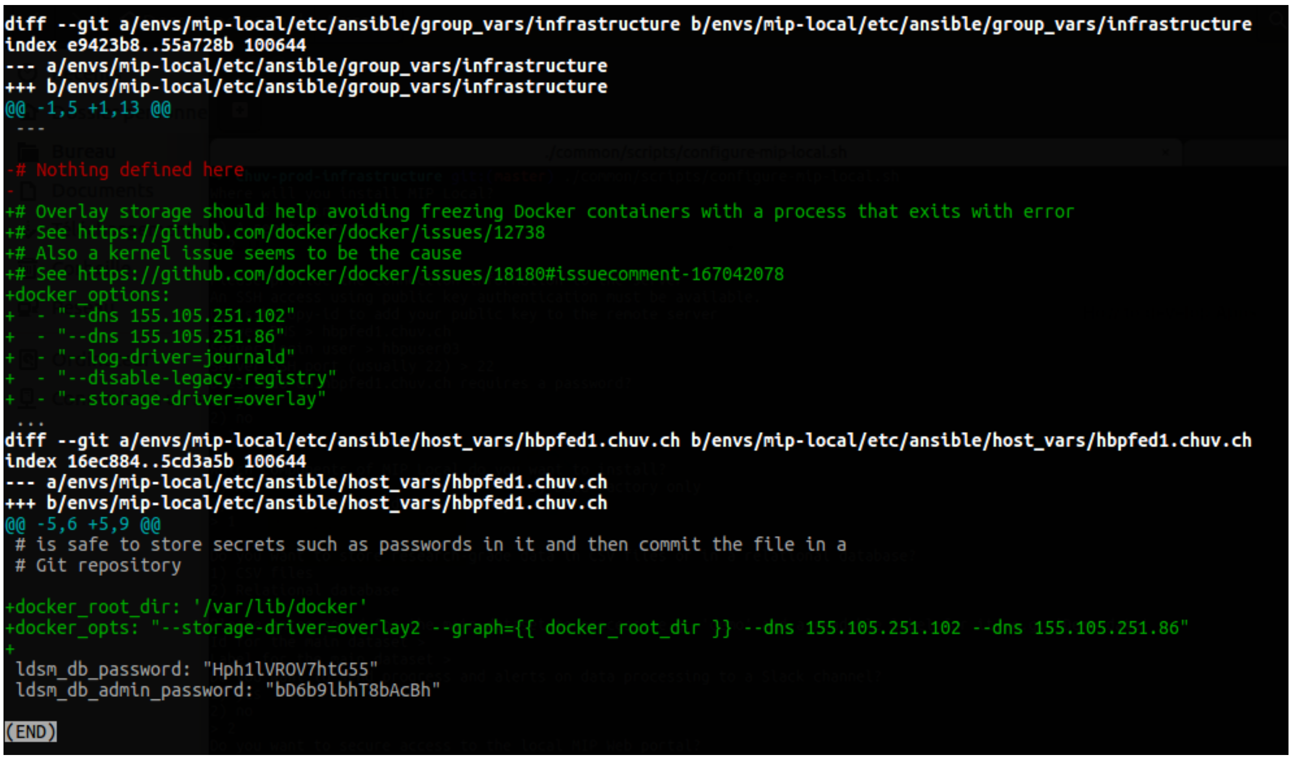
If you don't have a valid GPG key, the script will automatically ask you to create one.



```
TASK [mip-local : Generate host_vars file] ********************************************************************************
changed: [localhost]

TASK [mip-local : Protect the variables in host_vars/ directory with git-crypt] *******************************************
changed: [localhost]

TASK [mip-local : Generate Slack notification] ****************************************************************************
changed: [localhost]

TASK [mip-local : Generate README] ****************************************************************************************
changed: [localhost]

TASK [mip-local : Check for pre-commit] ***********************************************************************************
ok: [localhost]

TASK [mip-local : Remove pre-commit from after scripts if necessary] ******************************************************
skipping: [localhost] => (item=/home/mirco/Bureau/TOTO/mip-microservices-infrastructure/after-git-clone.sh)
skipping: [localhost] => (item=/home/mirco/Bureau/TOTO/mip-microservices-infrastructure/after-update.sh)

PLAY RECAP ****************************************************************************************************************
hbpfed1.chuv.ch            : ok=5    changed=0    unreachable=0    failed=0
localhost                  : ok=17   changed=13   unreachable=0    failed=0

Generating key...

Generation of the standard configuration for MIP Local complete!

You can review the configuration located in /home/mirco/Bureau/TOTO/mip-microservices-infrastructure/common/scripts/envs/mip-local/etc/ansible/
and customise it further for your environment and needs.
More information about the configuration settings can be found in
  /home/mirco/Bureau/TOTO/mip-microservices-infrastructure/common/scripts/docs/configuration/

Before starting the installation, please commit the configuration in Git:
  git commit -m 'Configuration for MIP Local'

Then run setup.sh to start the installation
  ./setup.sh
```

At the end of this step, newly created configuration files (and a few files that might have been updated) should be staged and git-crypt should have been used to encrypt the files containing secret information (please check it). You are invited to manually check and update the configuration files and commit the changes.

A specific DNS server from within the portal-backend Docker container had to be used in this step-by-step guide. Therefore, the following was added to the configuration:



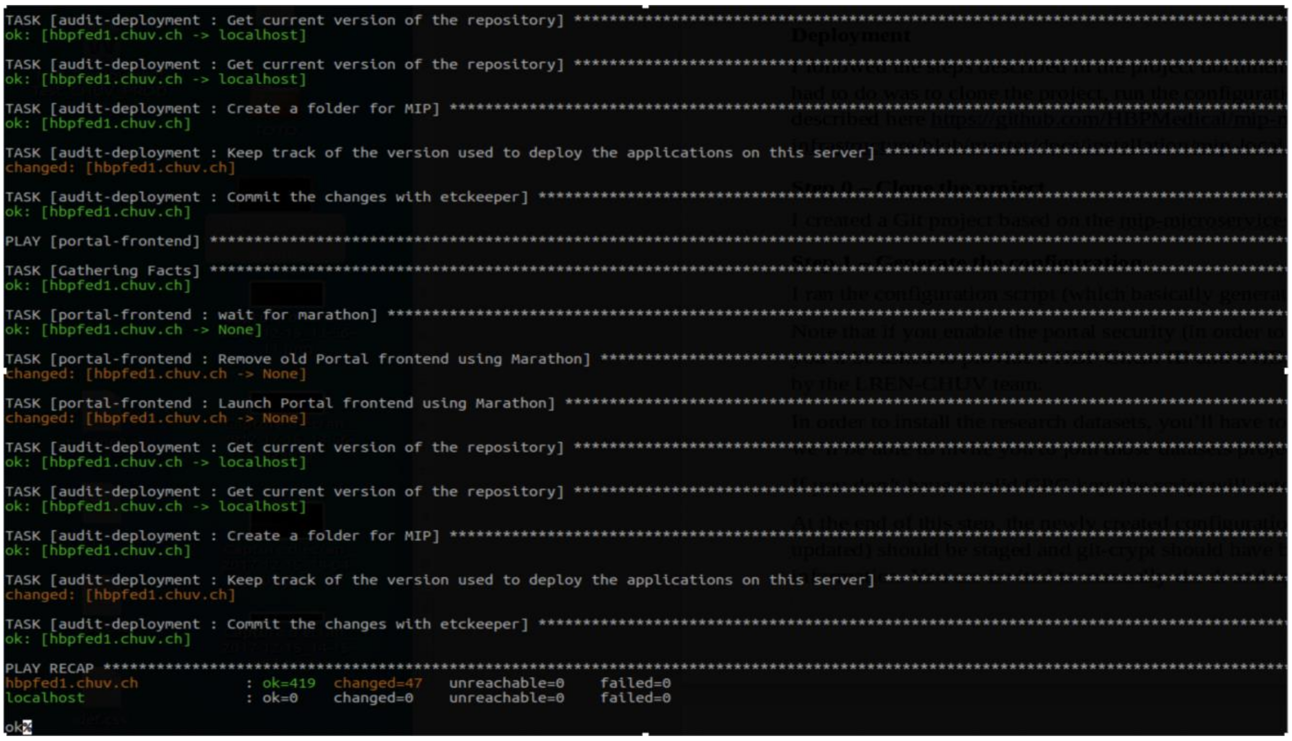
```
diff --git a/envs/mip-local/etc/ansible/group_vars/infrastructure b/envs/mip-local/etc/ansible/group_vars/infrastructure
index e9423b8..55a728b 100644
--- a/envs/mip-local/etc/ansible/group_vars/infrastructure
+++ b/envs/mip-local/etc/ansible/group_vars/infrastructure
@@ -1,5 +1,13 @@
 ---

-# Nothing defined here
+# Overlay storage should help avoiding freezing Docker containers with a process that exits with error
+# See https://github.com/docker/docker/issues/12738
+# Also a kernel issue seems to be the cause
+# See https://github.com/docker/docker/issues/18180#issuecomment-167042078
+docker_options:
+  - "--dns 155.105.251.102"
+  - "--dns 155.105.251.86"
+  - "--log-driver=journald"
+  - "--disable-legacy-registry"
+  - "--storage-driver=overlay"
 ...
diff --git a/envs/mip-local/etc/ansible/host_vars/hbpfed1.chuv.ch b/envs/mip-local/etc/ansible/host_vars/hbpfed1.chuv.ch
index 16ec884..5cd3a5b 100644
--- a/envs/mip-local/etc/ansible/host_vars/hbpfed1.chuv.ch
+++ b/envs/mip-local/etc/ansible/host_vars/hbpfed1.chuv.ch
@@ -5,6 +5,9 @@
 # is safe to store secrets such as passwords in it and then commit the file in a
 # Git repository

+docker_root_dir: '/var/lib/docker'
+docker_opts: "--storage-driver=overlay2 --graph={{ docker_root_dir }} --dns 155.105.251.102 --dns 155.105.251.86"
+
 ldsm_db_password: "Hph1lVROV7htG55"
 ldsm_db_admin_password: "bD6b9lbhT8bAcBh"

(END)
```

7.3 Step 3 - Install the Software

Now that the configuration files are created, the installation script can be executed. If a MATLAB license is not available yet, add the '--skip-tags=spm' parameter to the command:

```
./setup.sh –skip-tags=spm
```

IMPORTANT: At some point, the installation process might stop, requiring to re-run the installation script. This is NOT a problem! As indicated, setup.sh script needs just to be re-launched.

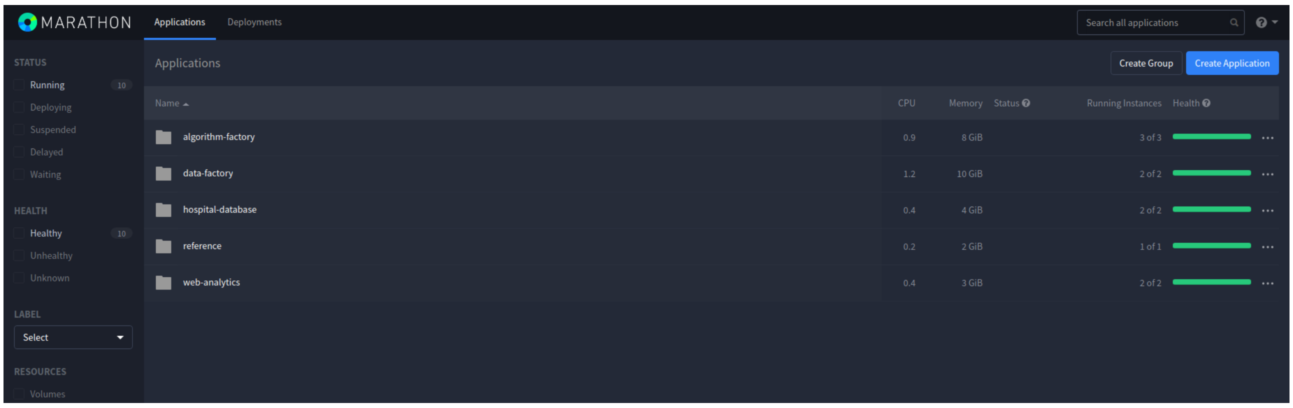


```
TASK [audit-deployment : Get current version of the repository] ****************************************************************
ok: [hbpfed1.chuv.ch -> localhost]

TASK [audit-deployment : Get current version of the repository] ****************************************************************
ok: [hbpfed1.chuv.ch -> localhost]

TASK [audit-deployment : Create a folder for MIP] ******************************************************************************
ok: [hbpfed1.chuv.ch]

TASK [audit-deployment : Keep track of the version used to deploy the applications on this server] *****************************
changed: [hbpfed1.chuv.ch]

TASK [audit-deployment : Commit the changes with etckeeper] ********************************************************************
ok: [hbpfed1.chuv.ch]

PLAY [portal-frontend] *********************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [hbpfed1.chuv.ch]

TASK [portal-frontend : wait for marathon] *************************************************************************************
ok: [hbpfed1.chuv.ch -> None]

TASK [portal-frontend : Remove old Portal frontend using Marathon] *************************************************************
changed: [hbpfed1.chuv.ch -> None]

TASK [portal-frontend : Launch Portal frontend using Marathon] *****************************************************************
changed: [hbpfed1.chuv.ch -> None]

TASK [audit-deployment : Get current version of the repository] ****************************************************************
ok: [hbpfed1.chuv.ch -> localhost]

TASK [audit-deployment : Get current version of the repository] ****************************************************************
ok: [hbpfed1.chuv.ch -> localhost]

TASK [audit-deployment : Create a folder for MIP] ******************************************************************************
ok: [hbpfed1.chuv.ch]

TASK [audit-deployment : Keep track of the version used to deploy the applications on this server] *****************************
changed: [hbpfed1.chuv.ch]

TASK [audit-deployment : Commit the changes with etckeeper] ********************************************************************
ok: [hbpfed1.chuv.ch]

PLAY RECAP *********************************************************************************************************************
hbpfed1.chuv.ch            : ok=419  changed=47   unreachable=0    failed=0
localhost                  : ok=0    changed=0    unreachable=0    failed=0

ok
```

The script was executed with a success. After that, the MIP is deployed.

7.4 Step 4 - Technical Verification

In order to check that all the components are correctly running, use the Marathon UI at http://hbpfed1.chuv.ch:5080:



Applications

Name	CPU	Memory	Status	Running Instances	Health
algorithm-factory	0.9	8 GiB		3 of 3	🟩
data-factory	1.2	10 GiB		2 of 2	🟩
hospital-database	0.4	4 GiB		2 of 2	🟩
reference	0.2	2 GiB		1 of 1	🟩
web-analytics	0.4	3 GiB		2 of 2	🟩

As you can see on the right side (green indicators), all the components are working OK.

8. References

[1] SP8 Medical Informatics Platform – Architecture and Deployment Plan
D. Milovanovic & E. Miquel Fernandez, http://bit.ly/HBP_MIP_SystemDescription

[2] Deployment Requirements
L. Claude, https://hbpmedical.github.io/deployment/

[3] External Services
L. Claude, https://hbpmedical.github.io/deployment/services/

[4] MIP Software
L.Claude, https://hbpmedical.github.io/deployment/software/

[5] SP8 Medical Informatics Platform – System Validation Plan at end of SGA1
F. Kherif & D. Milovanovic, http://bit.ly/HBP_MIP_SystemValidation