

**DEPLOYMENT GUIDE**

*Modern Statistics Platform (MauStats)*

Product Version: 1.0 October 2023

Table of Contents

[CHAPTER 1: INTRODUCTION 3](#_Toc148026052)

[CHAPTER 2: TECHNICAL SPECIFICATIONS 4](#_Toc148026053)

[DEVELOPER SYSTEM 4](#_Toc148026054)

[HARDWARE REQUIREMENTS 4](#_Toc148026055)

[SOFTWARE REQUIREMENTS 5](#_Toc148026056)

[HOSTING SERVER 5](#_Toc148026057)

[HARDWARE REQUIREMENTS 5](#_Toc148026058)

[SOFTWARE REQUIREMENTS 6](#_Toc148026059)

[CHAPTER 3: INSTALL AND CONFIGURE 7](#_Toc148026060)

[A. DEVELOPER SYSTEM 7](#_Toc148026061)

[Operating System 8](#_Toc148026062)

[Back-end Framework and Language Components 8](#_Toc148026063)

[Front-end framework and language components 8](#_Toc148026064)

[Integrated Development Environment (IDE) 8](#_Toc148026065)

[Web Server 8](#_Toc148026066)

[Database Server 9](#_Toc148026067)

[Docker For Packaging 9](#_Toc148026068)

[B. HOSTING SERVER 9](#_Toc148026069)

[Operating System 9](#_Toc148026070)

[Back-end framework and language components 9](#_Toc148026071)

[Web Server 9](#_Toc148026072)

[Database Server 9](#_Toc148026073)

[Docker For Packaging 10](#_Toc148026074)

[C. Preparing the client and server applications 10](#_Toc148026075)

[C.1 Client Application 10](#_Toc148026076)

[C.2 Server Application 11](#_Toc148026077)

[D. Hosting the applications on server 14](#_Toc148026078)

# CHAPTER 1: INTRODUCTION

MauStats platform is developed as a web-based platform that automate and integrate the data lifecycle from acquisition to dissemination. The platform has a centralize data management system that improves data quality assurance, streamline data processing and analysis, enhance data dissemination, promote integration and collaboration, and ensure compliance with international standards.

In this guide we will learn how to setup a developer computer system where we can install the required software and configure the application source code and the database to run, test and create a server executable built both for the front-end and the back-end applications.

We will also learn how to setup a web server that will host the complete web application and its database. Furthermore, we will also understand the process of mapping the domain to the server IP and use it to run the application.

**NOTE**: Web administration skills and knowledge are required to install, configure and setup the web application

# CHAPTER 2: TECHNICAL SPECIFICATIONS

As explained in chapter 1, the MauStats platform can be deployed both in the cloud or on a physical server. This application is developed in the MEAN stack technology. The MEAN stack is JavaScript-based framework for developing web applications. MEAN is named after *MongoDB, Express, Angular, and Node*, the four key technologies that make up the layers of the stack.

The web application comprises of two applications, the front-end and the back-end and the data is managed centrally in a database.

**NOTE**: This guide considers that both the application and the database are installed and configured on the same server.

You can setup a developer computer system on which you can copy the source code and create the executable front-end and back-end applications. These applications can than deployed on a hosting server. Let us now list down the hardware and software requirements for both the Developer system and the Hosting server.

## DEVELOPER SYSTEM

### HARDWARE REQUIREMENTS

|  |  |  |
| --- | --- | --- |
| **Component** | **Minimum** | **Recommended** |
| **Processor** | Intel® Core i5-5300U | Intel® Core i5-1135G7 |
| **Memory** | 16 GB RAM | 16 GB RAM |
| **Storage** | 240 GB SSD | 240 GB SSD |

### SOFTWARE REQUIREMENTS

|  |  |
| --- | --- |
| **Component** | **Requirement** |
| **Operating System** | Ubuntu (RHEL) 22.04 |
| **Front-end Language** | JavaScript vES2015, HTML v5.0, CSS v3.0 |
| **Visualization Library** | ECharts v5.4, Leafletv1.9.3 |
| **Front-end Framework** | Angular v15 |
| **Back-end Language** | Node.js v18.10.0, PHP v7.2, Python v3.8 |
| **Back-end Framework** | Express v4.16.2, Laravel v8.11.2, Django v3.0.6 |
| **Database System** | Mongo DB v5.0, Redis v7.0, Hadoop v3.3, Hudi v0.13.1 |
| **Server Type** | Apache v2.4\*, Docker v20.10.7, Docker composer v1.27.0 |

## HOSTING SERVER

### HARDWARE REQUIREMENTS

The hosting server can be a cloud based server or a physical server. Below is the hardware requirement of a cloud based server.

|  |  |  |
| --- | --- | --- |
| **Component** | **Minimum** | **Recommended** |
| **Processor** | 2 vCPU | 4 vCPU |
| **Memory** | 8 GB RAM | 16 GB RAM |
| **Memory** | 100 GB SSD | 100 GB SSD |
| **Domain** | Public IP and Dedicated Domain | |
| **Ports** | 80, 443, 27017, 6379 | |

### SOFTWARE REQUIREMENTS

|  |  |
| --- | --- |
| **Component** | **Requirement** |
| **Operating System** | Ubuntu (RHEL) 22.04 |
| **Front-end Language** | JavaScript vES2015, HTML v5.0, CSS v3.0 |
| **Visualization Library** | ECharts v5.4, Leafletv1.9.3 |
| **Front-end Framework** | Angular v15 |
| **Back-end Language** | Node.js v18.10.0, PHP v7.2, Python v3.8 |
| **Back-end Framework** | Express v4.16.2, Laravel v8.11.2, Django v3.0.6 |
| **Database System** | Mongo DB v5.0, Redis v7.0, Hadoop v3.3, Hudi v0.13.1 |
| **Server Type** | Apache v2.4\*, Docker v20.10.7, Docker composer v1.27.0 |

# CHAPTER 3: INSTALL AND CONFIGURE

This chapter explains in detail the step-by-step process to install and configure the following:

1. Developer System
2. Hosting Server
3. Preparing the client and server applications
4. Hosting the applications on server

## DEVELOPER SYSTEM

You may want to install and configure one developer system for the following tasks:

* Copy the source code and the database
* Prepare the front-end and the back-end applications
* Run the applications for testing

The source code is written in PHP and MEAN stack which is explained in Chapter 2. To prepare the developer system you need to install and configure the following software:

1. Operating System
2. Back-end framework and language components
3. Front-end framework and language components
4. Integrated Development Environment (IDE)
5. Web Server
6. Database Server
7. Docker for Packaging
8. Other Dependencies

Now let us learn how to install and configure each of these software.

### Operating System

|  |  |  |  |
| --- | --- | --- | --- |
| # | Software | Version | Function |
| 1 | Operating System | Ubuntu v22.04 | Communication bridge (interface) between the user and computer hardware |

### Back-end Framework and Language Components

|  |  |  |  |
| --- | --- | --- | --- |
| # | Software | Version | Function |
| 1 | Node js | v18.10.0 | All backend logic is written on Node js. It executes javascript code on server side and store data on MongoDB. |
| 2 | Express js | v4.16.2 | Framework provides structure to make software development easy. Express js is a Node js framework. |
| 3 | Nodemon | v1.19.4 | Nodemon detects changes in the directory or files and restart the software automatically. |
| 4 | NPM | v7.15.1 | NPM is a package manager for Node.js packages. |
| 5 | PHP | v7.2 | Api gateway is written on PHP. Api gateway is helps in communication between services. |
| 6 | Laravel | v8.11.2 | Laravel is a PHP framework. It accepts all http request and forward to relevant services. |
| 7. | Apache Hudi | v0.13.1 | Used to simplify incremental data processing and data pipeline development. |
| 8. | Apache Hadoop | V3.3 | Allows for the distributed storage and processing of large datasets across clusters of computers using simple programming models. |

### Front-end framework and language components

|  |  |  |  |
| --- | --- | --- | --- |
| # | Software | Version | Function |
| 1 | NPM | v7.15.1 | NPM is a package manager for Node.js packages. |
| 2 | Angular Cli | v15.0.3 | Angular Cli executes angular application and detect changes and recompile changes. It also helps in generating angular component, services, routes etc. |

### Integrated Development Environment (IDE)

|  |  |  |  |
| --- | --- | --- | --- |
| # | Software | Version | Function |
| 1 | Visual Studio | v17.1 | Source code editor |

### Web Server

|  |  |  |  |
| --- | --- | --- | --- |
| # | Software | Version | Function |
| 1 | Apache | v2.4 | It is a web server. Web servers are used to serve Web pages requested by client computers. |

### Database Server

|  |  |  |  |
| --- | --- | --- | --- |
| # | Software | Version | Function |
| 1 | MongoDB | v5.0 | MongoDB is a NoSql database server. Database server stores data. |
| 2 | Redis | v7.0 | Redis is database server mainly used for data caching. |

### Docker For Packaging

|  |  |  |  |
| --- | --- | --- | --- |
| # | Software | Version | Function |
| 1 | Docker | v20.10.7 | The Docker extension makes it easy to build, manage, and deploy containerized applications. Containerized applications are easy to ship from one machine to another machine. |
| 2 | Docker Composer | v1.27.0 | Docker Composer is used to run multiple docker containers as a single service. |

## HOSTING SERVER

### Operating System

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Software** | **Version** | **Function** |
| **1** | Operating System | Ubuntu v22.04 | communication bridge (interface) between the user and computer hardware |

### Back-end framework and language components

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Software** | **Version** | **Function** |
| **1** | PHP | v7.2 | Resource page is written using PHP to render resources. |

### Web Server

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Software** | **Version** | **Function** |
| **1** | Apache | v2.4\* | It is a web server. Web servers are used to serve Web pages requested by client computers. |

### Database Server

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Software** | **Version** | **Function** |
| **1** | MongoDB | v5.0 | MongoDB is a NoSql database server. Database server stores data. |
| **2** | Redis | v7.0 | Redis is database server mainly used for data caching. |

### Docker For Packaging

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Software** | **Version** | **Function** |
| **1** | Docker | v20.10.7 | The Docker extension makes it easy to build, manage, and deploy containerized applications. Containerized applications are easy to ship from one machine to another machine. |
| **2** | Docker Compose | v1.27.0 | Docker Compose is used to run multiple docker containers as a single service. |

## Preparing the client and server applications

### C.1 Client Application

#### **Create Dashboard build**

|  |  |
| --- | --- |
| **#** | **Function** |
| **1** | Go to [environments](https://www.partnershipsforthegoals.org/maustats/datamanager/-/tree/staging/src/environments)/[environment.ts](https://www.partnershipsforthegoals.org/maustats/datamanager/-/blob/staging/src/environments/environment.ts) file and change the configuration for dashboard. See below example –  production: true,  apiUrl:'http://maustats.itechmission.org/maustatsapigateway/',  domain:'http://maustats.itechmission.org', /  appURL: "http://maustats.itechmission.org",  sharedImagePath: "http://maustats.itechmission.org/images/",  mapURL: "http://maustats.itechmission.org/map/",  sharedURL: "http://maustats.itechmission.org/share.php",  mapUrl:'http://maustats.itechmission.org/maps/', resourceImgUrl:'http://maustats.itechmission.org/resourceUploads/' |
| **2** | Open terminal in dashboard directory and create build of dashboard (use $ ng build --base-href "/dashboard/") |
| **3** | A *dist* folder is now created, compress the *dist* folder and transfer it to server via FTP. |

#### **Create Core Application build**

|  |  |
| --- | --- |
| **#** | **Function** |
| **1** | Go to [environments](https://www.partnershipsforthegoals.org/maustats/datamanager/-/tree/staging/src/environments)/[environment.ts](https://www.partnershipsforthegoals.org/maustats/datamanager/-/blob/staging/src/environments/environment.ts) file and configure the server URL for Core application.  See below example –  production: true,  apiUrl: 'http://maustats.itechmission.org/maustatsapigateway/',  rootPath: 'http://maustats.itechmission.org',  logPath: 'http://maustats.itechmission.org/logs/',  domain:'http://maustats.itechmission.org',  appURL: "http://maustats.itechmission.org",  sharedImagePath: "http://maustats.itechmission.org/images/",  mapURL: "http://maustats.itechmission.org/map/",  sharedURL: "http://maustats.itechmission.org/share.php",  mapUrl:'http://maustats.itechmission.org/maps',  resourceImgUrl:'http://maustats.itechmission.org/resourceUploads/',  prefixDownloadFile: 'Maustats\_' |
| **2** | Open terminal in admin directory and create build of core application (use $ ng build --base-href "/core/") |
| **3** | A *dist* folder is now created, compress the *dist* folder and transfer it to server via FTP. |

### C.2 Server Application

#### **Create Deewar Image**

|  |  |
| --- | --- |
| **#** | **Function** |
| **1** | In the same .env file, change the database configuration as shown below –  DB\_DATABASE=mau\_stats  DB\_USERNAME=  DB\_PASSWORD=  # Docker  DATAPROCESSING\_SERVICE\_BASE\_URL=http://maustats\_dataimport:7031  DATARETRIEVAL\_SERVICE\_BASE\_URL=http://maustats\_datasupplier:7032  SDMX\_SERVICE\_BASE\_URL=http://maustats\_datasdmx:7033  PRIMARYDATA\_SERVICE\_BASE\_URL = http://maustats-primarydata:7034 |
| **2** | Create docker image for Deewar (use $ docker build --tag maustats\_apigateway) |
| **3** | Export this image as tar file (use $ docker save maustats\_apigateway:latest > maustats\_apigateway.tar) |
| **4** | Export this image as tar file (use $ docker save maustats\_apigateway:latest > maustats\_apigateway.tar) |

#### **Create Datasupplier Image**

|  |  |
| --- | --- |
| **#** | **Function** |
| **1** | Go to dataSupplier/supplyData/bin/www file and give the same port number as given in data retrieval service base URL as shown below –  **var port = normalizePort(process.env.PORT || '7032');** |
| **2** | Go to dataSupplier/supplyData/.env file and write the dbconfig details as shown below –  **PORT=7032**  **DB\_NAME=mau\_stats**  **DB\_PORT=27017**  **DB\_USERNAME=**  **DB\_PASSWORD=**  **REDIS\_PORT=6379** |
| **3** | Create docker image for datasupplier (use $ docker build --tag maustats\_datasupplier) |
| **4** | Export this image as tar file (use $ docker save maustats\_datasupplier:latest > msustats\_datasupplier.tar) |

#### **Create Dataimport Image**

|  |  |
| --- | --- |
| **#** | **Function** |
| **1** | Go to dataImport/processData/bin/www file and give the same port number as given in data processing service base URL as shown below –  **var port = normalizePort(process.env.PORT || '6011');** |
| **2** | Go to dataImport /processData/config/env.js file and write the dbconfig details as shown below –  **dbConfig: {**  **env: 'development',**  **db: 'mongodb://10.0.0.56:27017/mau\_stats, //production**  **port: 27017,**  **hostIp: '10.0.0.56',**  **radisPort: 6379**  **},** |
| **3** | Create docker image for dataimport (use $ docker build --tag maustats\_dataimport) |
| **4** | Export this image as tar file (use $ docker save maustats\_dataimport:latest > maustats\_dataimport.tar) |

#### **Create SDMX Image**

|  |  |
| --- | --- |
| **#** | **Function** |
| **1** | Go to dataImport/processData/bin/www file and give the same port number as given in data processing service base URL as shown below –  **var port = normalizePort(process.env.PORT || '7033');** |
| **2** | Go to dataImport /processData/.env file and write the dbconfig details as shown below –  **PORT=7033**  **DB\_NAME=mau\_stats**  **DB\_PORT=27017**  **DB\_USERNAME=**  **DB\_PASSWORD=**  **REDIS\_PORT=6379** |
| **3** | Create docker image for dataimport (use $ docker build --tag maustats\_datasdmx) |
| **4** | Export this image as tar file (use $ docker save maustats\_datasdmx:latest > maustats\_datasdmx) |

#### **Create PrimaryData Image**

|  |  |
| --- | --- |
| **#** | **Function** |
| **1** | Go to dataHudi /.env file and write the dbconfig details as shown below –  **DATABASE\_NAME=mau\_stats**  **COLLECTION\_NAME=metadata**  **COLLECTION\_NAME\_CALCULATION=primary\_calculation**  **CLIENT\_ADDRESS = mongodb://xxxx:yyyy@192.168.000.AA:27017/**  **DATASETS=/var/datasets**  **DOWNLOADPATH=/var/downloads/**  **SECRETKEY=ABCDEFGHIJKL** |
| **2** | Create docker image for datahudi (use $ docker build -t django-datahudi-prod .) |
| **3** | Export this image as tar file (use $ docker save django-datahudi-prod:latest > django-datahudi-prod.tar) |

## Hosting the applications on server

|  |  |
| --- | --- |
| **#** | **Function** |
| **1** | Open terminal and go to / var/www/ and give permissions to html file (use $ sudo chmod -R 0777 html) |
| **2** | Move to html (use $ cd html) and create folder with name maustats (use $ mkdir maustats) |
| **3** | Create a folder with name docker inside your project folder and transfer docker-compose.yml file inside that docker folder. |
| **4** | Transfer the following files on the server using FTP (FileZilla) –   * **maustats\_apigateway .tar** * **maustats\_datasupplier .tar** * **maustats\_dataimport.tar** * **dist.zip (dashboard)** * **dist.zip (core application)** |
| **5** | In docker folder, run these commands to load the images -   * for Deewar - $ cat maustats\_apigateway.tar | docker load * for datasupplier - $ cat maustats\_datasupplier.tar | docker load * for dataimport - $ cat maustats\_dataimport.tar | docker load |
| **6** | * Run this command to up docker images - $ docker-compose -f docker-compose.yml up -d |
| **7** | Now switch to project folder (Apache root folder) and unzip the dist folder using command (use $ sudo unzip distfolder\_path) |
| **8** | Now that you have extracted dist folder, copy the files that are inside dist folder into dashboard folder (use $ sudo cp -R dist/\* dashboard) |
| **9** | Keep .htaccess file on your server and copy into dashboard and admin folder. |

--- End of the Document ----