

**DEPLOYMENT GUIDE**

*SDG3 India Dashboard*

Product Version: 1.0 November 2024

Table of Contents

[CHAPTER 1: INTRODUCTION 3](#_Toc183700944)

[CHAPTER 2: TECHNICAL SPECIFICATIONS 4](#_Toc183700945)

[DEVELOPER SYSTEM 5](#_Toc183700946)

[Hardware Requirements 5](#_Toc183700947)

[Software Requirements 5](#_Toc183700948)

[HOSTING SERVER 6](#_Toc183700949)

[Hardware Requirements 6](#_Toc183700950)

[Software Requirements 6](#_Toc183700951)

[CHAPTER 3: INSTALL AND CONFIGURE 7](#_Toc183700952)

[A. Developer System 7](#_Toc183700953)

[Operating System 8](#_Toc183700954)

[Back-end Framework and Language Components 8](#_Toc183700955)

[Front-end framework and language components 8](#_Toc183700956)

[Integrated Development Environment (IDE) 8](#_Toc183700957)

[Web Server 9](#_Toc183700958)

[Database Server 9](#_Toc183700959)

[Docker For Packaging 9](#_Toc183700960)

[B. Hosting Server 9](#_Toc183700961)

[Operating System 9](#_Toc183700962)

[Back-end framework and language components 9](#_Toc183700963)

[Web Server 9](#_Toc183700964)

[Database Server 10](#_Toc183700965)

[Docker For Packaging 10](#_Toc183700966)

[C. Preparing the Client and Server Applications 11](#_Toc183700967)

[C.1 Client Application 11](#_Toc183700968)

[C.2 Server Application 13](#_Toc183700969)

[D. Hosting the Applications on Server 14](#_Toc183700970)

# CHAPTER 1: INTRODUCTION

India SDG3 Dashboard is a web-based application that enables users to visualize, explore, monitor, and report the country and sub-national level progress on performance indicators of the National Monitoring Framework (NMF) for Health SDG. The system is developed using open-source technology and tools that allow easy extensibility and scalable communication interfaces.

In this guide, we will learn how to set up 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 to 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 SDG3 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.

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.

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

## DEVELOPER SYSTEM

### Hardware Requirements

|  |  |  |
| --- | --- | --- |
| **Component** | **Minimum** | **Recommended** |
| **Processor** | Intel® Core i5-5300U | Intel® Core i5-1135G7 |
| **Memory** | 8 GB RAM | 16 GB RAM |
| **Storage** | 300 GB SSD | 500 GB SSD |

### Software Requirements

|  |  |
| --- | --- |
| **Component** | **Requirement** |
| **Operating System** | Ubuntu v20.04 |
| **Front-end Language** | JavaScript vES2015, HTML v5.0, CSS v3.0 |
| **Visualization Library** | ECharts v5.4.0, Leaflet v1.9.3 |
| **Front-end Framework** | AngularJS v17.0.2 |
| **Back-end Language** | Node.js v18.1.0, PHP v7.4 |
| **Back-end Framework** | Express v4.16.2, Laravel v8.11.2 |
| **Database System** | Mongo DB v5.0, Redis v7.0 |
| **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** | 300 GB SSD | 500 GB SSD |
| **Domain** | Public IP and Dedicated Domain | |
| **Ports** | 80, 443, 27017, 6379 | |

### Software Requirements

|  |  |
| --- | --- |
| **Component** | **Requirement** |
| **Operating System** | Ubuntu v20.04 |
| **Front-end Language** | JavaScript vES2015, HTML v5.0, CSS v3.0 |
| **Visualization Library** | ECharts v5.4.0, Leaflet v1.9.3 |
| **Front-end Framework** | AngularJS v17.0.2 |
| **Back-end Language** | Node.js v18.1.0, PHP v7.4 |
| **Back-end Framework** | Express v4.16.2, Laravel v8.11.2 |
| **Database System** | Mongo DB v5.0, Redis v7.0 |
| **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 v20.04 | 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. |

### 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 v20.04 | 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 –  export const environment = {  production: true,  apiUrl: 'https://sdghealthindia-mohfw.in/sdg3apigateway/',  domain: 'https://sdghealthindia-mohfw.in',  appURL: "https://sdghealthindia-mohfw.in",  sharedImagePath: "https://sdghealthindia-mohfw.in/images/",  mapURL: "https://sdghealthindia-mohfw.in/map/",  sharedURL: "https://sdghealthindia-mohfw.in/share.php",  mapUrl:'https://sdghealthindia-mohfw.in/maps',  resourceImgUrl:'https://sdghealthindia-mohfw.in/resourceUploads',  datamanagerUrl:'https://sdghealthindia-mohfw.in/datamanager/',  ngIdleStartTime:5,  ngIdleLogoutTime:1800  }; |
| **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 Datamanager 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 –  export const environment = {  production: true,  apiUrl: 'https://sdghealthindia-mohfw.in/sdg3apigateway/',  rootPath: 'https://sdghealthindia-mohfw.in',  logPath: 'https://sdghealthindia-mohfw.in/logs/',  domain: 'https://sdghealthindia-mohfw.in',  appURL: "https://sdghealthindia-mohfw.in",  sharedImagePath: "https://sdghealthindia-mohfw.in/images/",  mapURL: "https://sdghealthindia-mohfw.in/map/",  sharedURL: "https://sdghealthindia-mohfw.in/share.php",  mapUrl: 'https://sdghealthindia-mohfw.in/maps',  resourceImgUrl: 'https://sdghealthindia-mohfw.in/resourceUploads',  prefixDownloadFile: 'SDG3\_',  ngIdleStartTime: 5,  ngIdleLogoutTime: 1800,  dashboardUrl: 'https://sdghealthindia-mohfw.in/dashboard'  }; |
| **2** | Open terminal in admin directory and create build of core application (use $ ng build --base-href "/datamanager/") |
| **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=sdg3\_2024  DB\_USERNAME=  DB\_PASSWORD=  # Docker  DATAPROCESSING\_SERVICE\_BASE\_URL=[http://sdg3\_dataimport:5081](http://sdg3_dataimport:7081/) DATARETRIEVAL\_SERVICE\_BASE\_URL=[http://sdg3\_datasupplier:5082](http://sdg3_datasupplier:7082/) |
| **2** | Create docker image for Deewar (use $ docker build --tag sdg3\_apigateway .) |
| **3** | Export this image as tar file (use $ docker save sdg3\_apigateway:latest > sdg3\_apigateway.tar) |

#### **Create Datasupplier Image**

|  |  |
| --- | --- |
| **#** | **Function** |
| **1** | Go to dataSupplier/.env file and write the dbconfig details as shown below –  **PORT=5082**  **DB\_NAME=sdg3\_2024**  **DB\_PORT=27017**  **DB\_USERNAME=**  **DB\_PASSWORD=**  **REDIS\_PORT=6379** |
| **2** | Create docker image for datasupplier (use $ docker build --tag sdg3\_datasupplier .) |
| **3** | Export this image as tar file (use $ docker save sdg3\_datasupplier:latest > sdg3\_datasupplier.tar) |

#### **Create DataImport Image**

|  |  |
| --- | --- |
| **#** | **Function** |
| **1** | Go to dataImport /.env file and write the dbconfig details as shown below –  **PORT=5081**  **DB\_NAME=sdg3\_2024**  **DB\_PORT=27017**  **DB\_USERNAME=**  **DB\_PASSWORD=**  **REDIS\_PORT=6379** |
| **3** | Create docker image for dataimport (use $ docker build --tag sdg3\_dataimport .) |
| **4** | Export this image as tar file (use $ ddocker save sdg3\_dataimport:latest > sdg3\_dataimport.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 sdg (use $ mkdir sdg) |
| **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) –   * **sdg3\_apigateway.tar** * **sdg3\_datasupplier.tar** * **sdg3\_dataimport.tar** * **dist.zip (dashboard)** * **dist.zip (datamanager application)** |
| **5** | In docker folder, run these commands to load the images -   * for Deewar - $ cat sdg3\_apigateway.tar | docker load * for datasupplier - $ cat sdg3\_datasupplier.tar | docker load * for dataimport - $ cat sdg3\_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 ----