USAID GLOBAL HEALTH SUPPLY CHAIN PROGRAM

Procurement and Supply Management

|  |
| --- |
| INSTALLATION GUIDE  LaDATA (Logistic and Demographic Analysis for Targeted Action)  MAY 2021 |

The USAID Global Health Supply Chain Program-Procurement and Supply Management (GHSC-PSM) project is funded under USAID Contract No. AID-OAA-I-15-0004.  GHSC-PSM connects technical solutions and proven commercial processes to promote efficient and cost-effective health supply chains worldwide. Our goal is to ensure uninterrupted supplies of health commodities to save lives and create a healthier future for all. The project purchases and delivers health commodities, offers comprehensive technical assistance to strengthen national supply chain systems, and provides global supply chain leadership.

GHSC-PSM is implemented by Chemonics International, in collaboration with Arbola Inc., Axios International Inc., IDA Foundation, IBM, IntraHealth International, Kuehne + Nagel Inc., McKinsey & Company, Panagora Group, Population Services International, SGS Nederland B.V., and University Research Co., LLC. To learn more, visit [ghsupplychain.org](http://www.ghsupplychain.org/)

DISCLAIMER:  
The views expressed in this publication do not necessarily reflect the views of the U.S. Agency for International Development or the U.S. government.

Table of Contents

[Overall Architecture 1](#_Toc71553725)

[Application Components 1](#_Toc71553726)

[LaDATA Server Configuration 2](#_Toc71553727)

[1. Installation 2](#_Toc71553728)

[2. httpd service default configuration 3](#_Toc71553729)

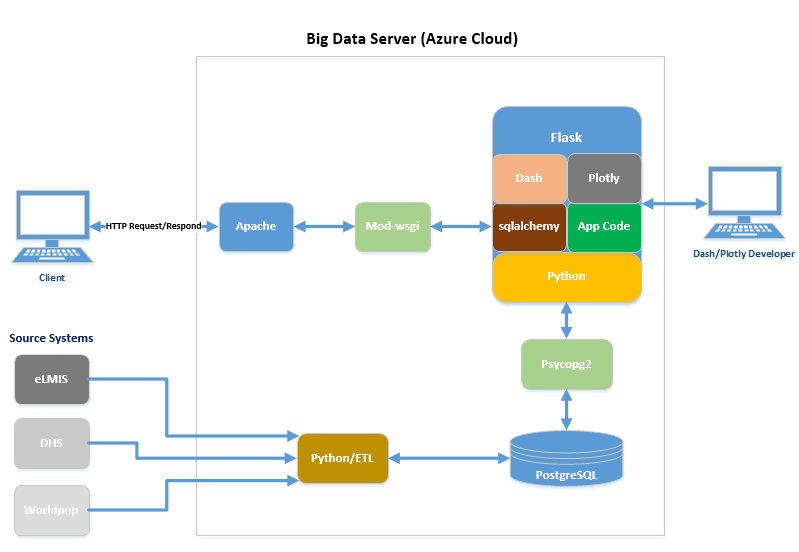
[3. PostgreSQL installation and Configuration 5](#_Toc71553730)

[4. Stop/Start/Restart postgres server 7](#_Toc71553731)

[5. Database Logs folder 8](#_Toc71553732)

[Software Documentation 8](#_Toc71553733)

# Overall Architecture



# Application Components

**Apache:** Web server (open-source software) which allows web applications to serve content on the web.

**mod\_wsgi:** An Apache HTTP Server module that provides an interface for hosting Python based web applications under Apache web server.

**Flask**: Lightweight web application framework written in Python. It provides user with libraries, modules, and tools to help build Web-Applications.

**Flask-SQLAlchemy:** Is an extension of Flask application. It’s a library that facilitates the communication between Python programs and databases. This library is used as an Object Relational Mapper (ORM) tool that translates Python classes to tables on relational databases and automatically converts function calls to SQL statements.

**Application code:** Custom application build on python using the Flask, Plotly and Dash framework.

**Python:** An object-oriented programming language that provides rapid application development.

**Psycopg2:** Is the PostgreSQL database driver for the Python programming language.

**PostgreSQL:** An open-source object-relational database system.

**Dash:** Dash is a Python framework for building analytical web applications. No JavaScript required. Built on top of Plotly.js, React and Flask, Dash ties modern UI elements (dropdowns, sliders, and graphs) directly to Python code.

**Plotly:** The plotly Python library (plotly.py) is an interactive, open-source plotting library that supports chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases. Built on top of the Plotly JavaScript library (plotly.js), plotly.py enables Python users to create interactive web-based visualizations.

LaDATA Server Configuration

Below is the server configuration:

* RedHat, Linux (x86\_64)
* 4 CPU @ 2.6 GHz
* 32 GB RAM
* 128 GB File System

## 1. Installation

Below is the installation commends for RedHat operating system using root account. Depending on your OS and access level, you may need to update the yum to apt and use sudo if the account is non-root.

1. Install python 3.6 or newer versions.
   * sudo yum -y install rh-python36
2. Install python virtual environment
   * pip3 install virtualenv
3. Create a virtual environment venv under /var/www folder
   * python3 -m venv venv
4. Activate the env
   * source venv/bin/activate
5. Install flask
   * pip3 install flask
6. Install Dash
   * pip3 install dash
7. Install psycopg2-binary
   * pip3 install psycopg2-binary
8. Install flask\_sqlalchemy
   * pip3 install flask\_sqlalchemy
9. Install mod\_wsgi
   * yum install mod\_wsgi
10. Install the webserver
    * sudo yum install httpd
11. Scripts to start/stop/restart the server
    * systemctl start|stop|status httpd.service
    * systemctl status httpd.service

## 2. httpd service default configuration

1. Default config file: /etc/httpd/conf/httpd.conf
2. Configuration files which load modules : /etc/httpd/conf.modules.d/ directory (e.g. PHP)
3. Select MPMs (Processing Model) as loadable modules [worker, prefork (default)] and event: /etc/httpd/conf.modules.d/00-mpm.conf
4. Default ports: 80 and 443 (SSL)
5. Default log files: /var/log/httpd/{access\_log,error\_log}

Graphical user interface, text, application, email

Description automatically generated

1. Update the httpd.conf file with the appropriate paths.

DocumentRoot "/var/www"

WSGIDaemonProcess example threads=5

WSGIScriptAlias / /var/www/webapp/webapp.wsgi

WSGIPassAuthorization On

Alias /static/ /var/www/webapp/static/

<Directory webapp>

WSGIProcessGroup webapp

WSGIApplicationGroup %{GLOBAL}

Order deny,allow

Allow from all

</Directory>

#

# Relax access to content within /var/www.

#

<Directory "/var/www">

AllowOverride None

# Allow open access:

Require all granted

</Directory>

# Further relax access to the default document root:

<Directory "/var/www/webapp/static">

2. Create the webapp.wsgi file below under the /var/www/webapp/ if it does not exist.

import sys

sys.path.insert(0, '/var/www/webapp/')

activate\_this = '/var/www/webapp/venv/bin/activate\_this.py'

with open(activate\_this) as file\_:

exec(file\_.read(), dict(\_\_file\_\_=activate\_this))

from webapp import server as application

3. Copy the content of the bigdata application under /var/www/webapp

## 3. PostgreSQL installation and Configuration

1. Identify the platform of the OS.
2. cat /etc/os-release
3. Generate the installation commend

Graphical user interface, text, application, email

Description automatically generated

yum install <https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm>

1. Install Postgres database
   * yum install postgresql12-server
2. After the successful installation decide the DATA AREA for DATABASE
   * /bigdata (is the 128 GB attached disk) is used for this installation.
3. Create directory postgresql under /bigdata
4. Give ownership of DATA AREA to user postgres.
   * chown postgres:postgres -R /bigdata/postgresql
5. Switch to postgres user and create complete structured directory to store data.
6. Set environment variable in .bash\_profile of postgres user.
7. #PostgreSQL Server Environment Variable Settings.
   * LD\_LIBRARY\_PATH=/usr/pgsql-12/bin
   * export LD\_LIBRARY\_PATH
   * PATH=/usr/pgsql-12/bin:$PATH
   * export PATH
   * PGDATA=/bigdata/postgresql/12/data
   * export PGDATA
   * echo "PG Profile Loaded"
   * echo "DATA AREA is /bigdata/postgresql/12/data"
   * echo "PostgreSQL Version is 12"
   * vi .bash\_profile
8. Initialize Database Cluster.
   * /usr/pgsql-12/bin/initdb
9. Start the database server using pg\_ctl command.
   * /usr/pgsql-12/bin/pg\_ctl -D /bigdata/postgresql/12/data -l logfile start
10. Create database
    * Create database bigdata;
11. Change current database.
    * \c bigdata
12. Set environment variable PGDATABASE to “bigdata” to connect to bigdata database by default while invoking psql.
    * PGDATABASE=bigdata
    * export PGDATABASE
13. create a super user account and assign a password.
    * postgres=# CREATE USER dbadmin SUPERUSER;
    * postgres=# ALTER ROLE dbadmin WITH PASSWORD 'xxxxxx';
14. update postgresql.conf file based on your policy.
    * listen\_addresses = ‘\*’
15. Update pg\_hba.conf file based on your policy
    * Host all all 0.0.0.0/0 md5
16. Restart server
    * sudo su postgres
    * /usr/pgsql-12/bin/pg\_ctl -D /bigdata/postgresql/12/data -l logfile restart
17. Request the server admin to open the firewall (inbound traffic) on port 5432
18. Use a client tool to validate the connection/access.
19. Create the database objects using attached DDL’s.



## 4. Stop/Start/Restart postgres server

Sudo su – postgres

/usr/pgsql-12/bin/pg\_ctl -D /bigdata/postgresql/12/data -l logfile restart

/usr/pgsql-12/bin/pg\_ctl -D /bigdata/postgresql/12/data -l logfile stop

/usr/pgsql-12/bin/pg\_ctl -D /bigdata/postgresql/12/data -l logfile start

## 5. Database Logs folder

/bigdata/postgresql/12/data/log

# Software Documentation

Below are the links related to the selected open-source software’s:

* [https://flask.palletsprojects.com/en/1.1.x/](https://leafletjs.com/)
* <https://leafletjs.com/>
* <https://www.postgresql.org/>
* <https://www.chartjs.org/>
* <https://www.python.org/>
* <https://pypi.org/project/psycopg2/>
* <https://www.sqlalchemy.org/>
* <https://pypi.org/project/mod-wsgi/>
* <https://plotly.com/>
* <https://dash.plotly.com/>