

**Behavioral Health Information Technology and Standards (BHITS) Project**

**Omnibus Care Plan (OCP) Version 1.0.0**

**Deployment Guide**

**September 2018**

**Prepared by FEI Systems**

**For the Substance Abuse and Mental Health Services Administration**

This Omnibus Care Plan (OCP) Version 1.0.0 Deployment Guide was developed by FEI Systems for the Behavioral Health Information Technologies & Standards (BHITS) contract, funded by the Substance Abuse and Mental Health Services Administration (SAMHSA).

Contents

[1 Introduction 4](#_Toc523996163)

[1.1 Overview 4](#_Toc523996164)

[1.2 Purpose 4](#_Toc523996165)

[1.3 Organization of this Guide 4](#_Toc523996166)

[1.4 Prerequisites 4](#_Toc523996167)

[1.5 Software List Used in OCP Solution 5](#_Toc523996168)

[1.6 Acronyms 6](#_Toc523996169)

[2 Deployment Server Setup 7](#_Toc523996170)

[2.1 Docker Installation 7](#_Toc523996171)

[2.1.1 Prerequisites 7](#_Toc523996172)

[2.1.2 Install Docker and Docker Compose 7](#_Toc523996173)

[2.1.3 Add User Accounts to Docker Group 7](#_Toc523996174)

[3 OCP Deployment 8](#_Toc523996175)

[3.1 One Server Setup 8](#_Toc523996176)

[3.1.1 Configure OCP Linux Server 8](#_Toc523996177)

[3.1.2 Compose Containers on Server 10](#_Toc523996178)

[3.1.3 Restore UAA and FHIR Server Databases 10](#_Toc523996179)

[3.1.4 Run the Application 10](#_Toc523996180)

[3.2 Two Server Setup 10](#_Toc523996181)

[3.2.1 Configure OCP Application Server 10](#_Toc523996182)

[3.2.2 Configure Database Server 12](#_Toc523996183)

[3.2.3 Compose Containers on Database Server 13](#_Toc523996184)

[3.2.4 Restore UAA and FHIR Server Databases 13](#_Toc523996185)

[3.2.5 Run the Application on Application Server 13](#_Toc523996186)

[Companion Folders and Files 14](#_Toc523996187)

[FHIR Server Deployment on AWS 15](#_Toc523996188)

[Setup FHIR database server using RDS 15](#_Toc523996189)

[Setup FHIR APP in server 15](#_Toc523996190)

[Configure FHIR APP to Use FHIR DB 15](#_Toc523996191)

# Introduction

## Overview

The Omnibus Care Plan is intended to be a patient-centric, standards-based, goal-driven treatment plan through which the patient and his or her healthcare team can effectively communicate. The Omnibus Care Plan is meant to be the authoritative source for care planning. It will synthesize, reconcile, and de-conflict the patient’s data and monitor ongoing processes across health care institution boundaries.

The Omnibus Care Plan is a patient-entered, standard-based, single-source care plan management system available in one place to both the patient and the patient's Care Team. It is a preferred method of documenting the relevant care plan and treatment information across care teams and with patients. The Omnibus Care Plan web application includes patient engagement and referral capabilities as well as a care plan using Veterans Administration-defined clinical pathways for opioid management and suicide prevention.

All information is entered into the application by the care team. The Omnibus Care Plan provides a patient care plan that allows providers to view a consolidated “care-plan-of-care-plans.” It provides a real-time integrated view of an individual care plan and all associated information, including structured and narrative information. It is not specific to behavioral health but it is intended to contain all the care plans applicable to a patient.

The Omnibus Care Plan application consists of multiple modules. It includes an Appointments module, Assessments and Screenings, Care Coordination, Care Plan, Infrastructure, Patient Registration and Management, and User Management Authorization and Authentication. It includes the Consent2Share data segmentation and consent management application.

## Purpose

This document was prepared by the Omnibus Care Plan (OCP) application developers primarily to document key infrastructure setup, installation, configuration, and deployment technologies required to operationalize OCP. This document is not a step-by-step software application development guide. Rather, this document is a reference guide to help developers and system administrators install, configure, deploy, and validate the key software components that operationalize OCP application.

## Organization of this Guide

This Deployment Guide is divided into four Chapters:

* Chapter One provides an introduction to Omnibus Care Plan (OCP) and the purpose of this guide
* Chapter Two provides information about setting up the deployment environment based on Docker
* Chapter Three provides instructions to deploy and configure OCP using Docker on Linux servers

## Prerequisites

This document is designed for developers and system administrators who install, configure, deploy, and maintain distributed applications. Familiarity with the following is recommended.

* Basic Linux system administration
* Basic knowledge of Docker and Docker-Compose
* Basic knowledge of Public Key Infrastructure (PKI) and creating SSL certificates

To run the scripts in this document, it is assumed that you can access the Docker and GitHub repositories specified. GitHub repositories and other files are also in the companion folder of this document, you can clone/copy the Git repositories to your specific location or copy/build Docker images to your own Docker registry, and adjust the scripts for your own deployment.

## Software List Used in OCP Solution

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Open Source?** | **Reference Link** | **License Fee** |
| Git | Y | https://gitforwindows.org/ | N/A |
| Github | N | https://github.com/ | $0-200/Month |
| IntelliJ IDEA Ultimate | N | https://www.jetbrains.com/idea/ | $300-500/Year |
| VirtualBox | Y | https://www.virtualbox.org/ | N/A |
| Docker | Y | https://www.docker.com/ | N/A |
| Maven | Y | https://maven.apache.org/ | N/A |
| Gradle | Y | https://gradle.org/ | N/A |
| Tomcat | Y | http://tomcat.apache.org/ | N/A |
| Java | Y | https://www.java.com/ | N/A |
| Spring Framework | Y | https://spring.io/ | N/A |
| Spring Boot | Y | https://spring.io/ | N/A |
| Spring Cloud | Y | https://spring.io/ | N/A |
| Hibernate | Y | http://hibernate.org/ | N/A |
| Logback | Y | https://logback.qos.ch/ | N/A |
| Apache PDFBox | Y | https://pdfbox.apache.org/ | N/A |
| PostgreSQL | Y | https://www.postgresql.org/ | N/A |
| UAA | Y | https://github.com/cloudfoundry/uaa | N/A |
| HAPI FHIR | Y | http://hapifhir.io/ | N/A |
| NVM | Y | https://github.com/creationix/nvm | N/A |
| Node.js | Y | https://nodejs.org/ | N/A |
| NPM | Y | https://www.npmjs.com/ | N/A |
| webpack | Y | https://webpack.js.org/ | N/A |
| ES6 | Free | http://es6-features.org/ | N/A |
| Babel | Y | https://babeljs.io/ | N/A |
| React | Y | https://reactjs.org/ | N/A |
| Redux | Y | https://redux.js.org/ | N/A |
| Redux Saga | Y | https://github.com/redux-saga/redux-saga | N/A |
| styled components | Y | https://github.com/styled-components/styled-components | N/A |
| CSS Grid Layout | Free | https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\_Grid\_Layout | N/A |
| GoldenLayout | Y | https://golden-layout.com/ | N/A |
| Jest | Y | https://jestjs.io/ | N/A |
| Enzyme | Y | https://github.com/airbnb/enzyme | N/A |
| ESLint | Y | https://eslint.org/ | N/A |
| react boilerplate | Y | https://github.com/react-boilerplate/react-boilerplate | N/A |

## Acronyms

|  |  |
| --- | --- |
| SAMHSA | Substance Abuse and Mental Health Services Administration |
| OCP | Omnibus Care Plan |
| C2S | Consent2Share |
| FHIR | Fast Healthcare Interoperability Resources |
| HL7 | Health Level-7 |
| SoF | SMART on FHIR |
| SMART | Substitutable Medical Applications & Reusable Technologies |
| OS | Operating System |
| UI | User Interface |
| GUI | Graphic User Interface |
| VM | Virtual Machine |
| OAuth | Open Authorization |
| IDE | Integrated Development Environment |
| SDK | Software Development Kit |
| JDK | Java Development Kit |
| JRE | Java Runtime Environment |
| JCE | Java Cryptography Extension |
| JVM | Java Virtual Machine |
| JPA | Java Persistence API |
| ORM | Object-Relational Mapping |
| API | Application Program Interface |
| REST | Representational State Transfer |
| NVM | Node Version Manager |
| NPM | Node Package Manager |
| UAA | User Account and Authentication |
| YAML | YAML Ain't Markup Language |
| FIS | FHIR Integration Service |

# Deployment Server Setup

## Docker Installation

The following provides instructions about how to install Docker on a Linux CentOS 7.X server as an example.

### Prerequisites

* Docker requires a 64-bit installation regardless of your CentOS version.
* Your kernel must be 3.10 at a minimum, which CentOS 7 runs. To check the CentOS version, run the command “uname -r” in the terminal.
* User account should have sudo or root privileges
* Ensure yum and curl are installed, and networking is operational.

### Install Docker and Docker Compose

* Get the [ocp\_docker\_install.sh](https://github.com/FEISystems/omnibus-care-plan/blob/master/infrastructure/scripts/ocp_docker_install.sh)[[1]](#footnote-2) and run the script file.

sh ocp\_docker\_install.sh

* Verify the Docker installation.

sudo docker version

sudo docker run hello-world

Output message will contain the following:

*Hello from Docker!*

This message shows that your installation appears to be working correctly.

* Verify Docker compose installation.

sudo docker-compose --version

Note: if docker-compose gives the command “not found” try with following:

/usr/local/bin/docker-compose --version

### Add User Accounts to Docker Group

The user accounts that need to run Docker and Docker Compose commands must be added to the Docker group. Run the following command by replacing the \*\*\* with the actual username to add a user to the Docker group

sudo usermod -aG docker \*\*\*

# OCP Deployment

The deployment is provided to run the OCP application on Linux servers. Here we use CentOS 7.X as an example to describe the setups.

In this deployment guide, we provide two setup options:

1. In One Server Setup, databases and application (services and UIs) are all running on one Linux server instance. This is not a recommended deployment option for production deployment. It is only for demonstration purpose, particularly for local demonstration.
2. In Two Server Setup, databases and application (services and UIs) are running on separate Linux server instances. This is the typical recommended deployment option for production deployment.

## One Server Setup

This option is designed to run all OCP services and UIs and databases on one Linux server instance.

### Configure OCP Linux Server

#### Server Information

|  |  |  |
| --- | --- | --- |
| **Server (Application and Databases)** | **Recommended** | **Minimum** |
| **CPU** | 8 | 4 |
| **Memory** | 16 GB | 8 GB |
| **Storage** | 120 GB | 60 GB |

#### Configure Server

* Get the [ocp\_config.sh](https://github.com/FEISystems/omnibus-care-plan/blob/master/infrastructure/scripts/one-server/ocp_config.sh)[[2]](#footnote-3) and run the script file.

sh ocp\_config.sh

Enter 1 when console prompted “Please select a server to setup:” to setup OCP Server.

**Expected Results:**

The following subfolders and the OCP configurations will be created under ‘/usr/local/’ folder.

* + - Java
    - docker-[compose](https://github.com/FEISystems/omnibus-care-plan/blob/master/infrastructure/deployment/one-server/docker-compose.yml) file : docker-compose.yml
    - Application Files
    - java /OCP\_PROPS/uaa
    - [uaa configuration](https://github.com/FEISystems/ocp-uaa/blob/master/external-configuration/uaa.yml) file: /uaa/uaa.yml
    - java /OCP\_PROPS/[ocp-config-data](https://github.com/FEISystems/ocp-config-data.git)
    - [OCP configuration](https://github.com/FEISystems/ocp-config-data) files: /ocp-config-data

Note: If not present clone it by using the following command

git clone https://github.com/FEISystems/ocp-config-data.git

* + - java/keystore
    - The [ocp\_env.sh](https://github.com/bhits-dev/consent2share/blob/master/infrastructure/scripts/lite-edition/c2s_app_env.sh) file will be placed in ‘/etc/profile.d/’ folder.
* Manually get all the above files

Instead of running scripts with curl and Git clone as shown above, all those files can be found in the companion folder. Please copy corresponding files to target folders accordingly.

* + - Edge-server security:
    - Create/Obtain a valid SSL certificate
    - Export the public and private keys from the SSL certificate to a JKS formatted keystore file named ‘edge-server.keystore’
    - upload the ‘edge-server.keystore’ file into ‘/usr/local/java/keystore’ folder
    - Add the following in the

‘/usr/local/java/OCP\_PROPS/ocp-config-data/edge-server.yml‘ file.

spring.profiles: your\_app\_Server\_specific\_profile

server:

ssl:

key-store: /java/keystore/edge-server.keystore

key-store-password: "keystore password"

* + - Modify the ENV\_APP\_PROFILE= your\_app\_Server\_specific\_profile in appServerConfig() function in the ‘/etc/profile.d/ocp\_env.sh’ file.
    - Modify the following configuration files:
  + Set edge server, config server, and SMTP variables to the server specific values in the ‘/etc/profile.d/ocp\_env.sh’ file.
  + There are many OCP API services/components. The configurations in these services/components can be overridden using the corresponding configuration API service/component YAMLs that are available in the ocp-config-data folder. The structure of the API service/component YAMLS should be similar with the corresponding application.yml mentioned in the each API service/component. They can be found in the ‘src/main/resources’ folder.
    - * For instance, to override database variables for UAA API. The following can be added in the uaa.yml as below

spring.profiles: your\_app\_Server\_specific\_profile

spring:

datasource:

url: "database url"

username: "database user name"

password: "{cipher} encypted database pwd"

* Re-login to the server in order for the file ` ocp\_env.sh ` to run automatically during the login
  + Verify by checking any variable mentioned in the file

Ex: echo ${OCP\_BASE\_PATH} should show the value set in the file

### Compose Containers on Server

Run the following command from the ‘/usr/local/java’ folder to start up all OCP services and UIs:

docker-compose up -d

Run ‘docker ps –a’ to verify all the containers are created.

### Restore UAA and FHIR Server Databases

In order to prepopulate sample role, scopes and users in OCP and sample FHIR resources, please run following command to restore uaa and FHIR database with backup files in the companion database folder.

cat {database\_backup\_file} | docker exec -i {database\_container\_id} pg\_restore -U postgres –d {schema}

### Run the Application

Run the following command from the ‘/usr/local/java’ folder to start up all OCP services and UIs:

docker-compose up -d

Run ‘docker ps –a’ to verify all the containers are up running.

To verify OCP deployment, access the UI: https://<application\_server>/ocp-ui

* + Login with credentials in the companion user-credentials folder
* Follow the OCP User Guides to verify OCP features.

## Two Server Setup

This option is designed to run all OCP services and UIs on one Linux instance as application server, and databases on another Linux instance as database server.

### Configure OCP Application Server

#### Application Server Information

|  |  |  |
| --- | --- | --- |
| **Application Server** | **Recommended** | **Minimum** |
| **CPU** | 8 | 4 |
| **Memory** | 16 GB | 8 GB |
| **Storage** | 120 GB | 60 GB |

#### Configure App Server

* Get the [ocp\_config.sh](https://github.com/FEISystems/omnibus-care-plan/blob/master/infrastructure/scripts/ocp_config.sh)[[3]](#footnote-4) and run the script file.

sh ocp\_config.sh

Enter 2 when console prompted “Please select a server to setup:” to setup two server OCP App Server.

**Expected Results:**

The following subfolders and the OCP configurations will be created under ‘/usr/local/’ folder.

* + - Java
    - docker-[compose](https://github.com/FEISystems/omnibus-care-plan/blob/master/infrastructure/deployment/docker-compose_ocp_app.yml)\_ocp\_app file : docker-compose.yml
    - Application Files
    - java /OCP\_PROPS/uaa
    - [uaa configuration](https://github.com/FEISystems/ocp-uaa/blob/master/external-configuration/uaa.yml) file: /uaa/uaa.yml
    - java /OCP\_PROPS/[ocp-config-data](https://github.com/FEISystems/ocp-config-data.git)
    - [OCP configuration](https://github.com/FEISystems/ocp-config-data) files: /ocp-config-data

Note: If not present clone it by using the following command

git clone https://github.com/FEISystems/ocp-config-data.git

* + - java/keystore
    - The [ocp\_app\_env.sh](https://github.com/bhits-dev/consent2share/blob/master/infrastructure/scripts/lite-edition/c2s_app_env.sh) file will be placed in ‘/etc/profile.d/’ folder.
* Manually setup configuration:

Instead of running scripts with curl and Git clone, all those files can be found in the company folder. Please copy corresponding files to target folder accordingly based on the scripts

* + - Edge-server security:
    - Create/Obtain a valid SSL certificate
    - Export the public and private keys from the SSL certificate to a JKS formatted keystore file named ‘edge-server.keystore’
    - upload the ‘edge-server.keystore’ file into ‘/usr/local/java/keystore’ folder
    - Add the following in the

‘/usr/local/java/OCP\_PROPS/[ocp-config-data](https://github.com/bhits-dev/c2s-config-data.git)/edge-server.yml‘ file.

spring.profiles: your\_app\_Server\_specific\_profile

server:

ssl:

key-store: /java/keystore/edge-server.keystore

key-store-password: "keystore password"

* + - Modify the ENV\_APP\_PROFILE= your\_app\_Server\_specific\_profile in appServerConfig() function in the ‘/etc/profile.d/ocp\_env.sh’ file.
    - Modify the following configuration files:
  + Set edge server, config server, and SMTP variables to the server specific values in the ‘/etc/profile.d/ocp\_env.sh’ file.
  + There are many OCP API services/components. The configurations in these services/components can be overridden using the corresponding configuration API service/component YAMLs that are available in the ocp-config-data folder. The structure of the API service/component YAMLS should be similar with the corresponding application.yml mentioned in the each API service/component. They can be found in the ‘src/main/resources’ folder.
    - * For instance, to override database variables for UAA API. The following can be added in the uaa.yml as below

spring.profiles: your\_app\_Server\_specific\_profile

spring:

datasource:

url: "database url"

username: "database user name"

password: "{cipher} encrypted database pwd"

* Re-login to the server in order for the file ` ocp\_env.sh ` to run automatically during the login
  + Verify by checking any variable mentioned in the file

Ex: echo ${OCP\_BASE\_PATH} should show the value set in the file

### Configure Database Server

#### Database Server Info

|  |  |  |
| --- | --- | --- |
| **Database Server** | **MAX** | **MIN** |
| **Memory** | 8 GB | 4GB |
| **Storage** | 80GB | 40GB |
| **CPU** | 4 | 2 |

#### Configure Database Server

* Get the [ocp\_config.sh](https://github.com/FEISystems/omnibus-care-plan/blob/master/infrastructure/scripts/ocp_config.sh)[[4]](#footnote-5) [[5]](#footnote-6) and run the script file.

sh ocp\_config.sh

Enter 3 when console prompted “Please select a server to setup:” to setup two server OCP Database Server.

**Expected Results:**

The following subfolders and the OCP configurations will be created under ‘/usr/local/’ folder.

* + - Java
    - docker-[compose](https://github.com/FEISystems/omnibus-care-plan/blob/master/infrastructure/deployment/docker-compose_ocp_app.yml)\_ocp\_db file : docker-compose.yml
    - The [ocp\_db\_env.sh](https://github.com/bhits-dev/consent2share/blob/master/infrastructure/scripts/lite-edition/c2s_app_env.sh) file will be placed in ‘/etc/profile.d/’ folder.
* Manually setup configuration:

Instead of running scripts with curl and Git clone, all those files can be found in the company folder. Please copy corresponding files to target folder accordingly based on the scripts

* Re-login to the server in order for the file ` [ocp\_db\_env.sh](https://github.com/bhits-dev/consent2share/blob/master/infrastructure/scripts/lite-edition/c2s_app_env.sh)` to run automatically during the login
  + Verify by checking any variable mentioned in the file

Ex: echo ${UAA\_DB\_PASSWORD} should show the value set in the file

### Compose Containers on Database Server

Run the following command from the ‘/usr/local/java’ folder to start up all OCP services and UIs:

docker-compose up -d

Run ‘docker ps –a’ to verify all database containers are up running.

### Restore UAA and FHIR Server Databases

In order to prepopulate sample role, scopes and users in OCP and sample FHIR resources, please run following command to restore uaa and FHIR database with backup files in the companion database folder.

cat {database\_backup\_file} | docker exec -i {database\_container\_id} pg\_restore -U postgres –d {schema}

### Run the Application on Application Server

Run the following command from the ‘/usr/local/java’ folder to start up all OCP services and UIs:

docker-compose up -d

Run ‘docker ps –a’ to verify all application containers are up running.

To verify OCP deployment, access the UI: https://<application\_server>/ocp-ui

* + Login with credentials in the companion user-credentials folder
* Follow the OCP User Guides to verify OCP features.

# Companion Folders and Files

1. The omnibus-care-plan folder

This is the Git repository which has the scripts to set up the infrastructure for OCP to run.

1. The docker-images folder

This is the folder has the Docker images for OCP components including HAPI FHIR server.

1. OCP Source Code Git Repositories  
   All OCP source code git repositories are located inside of “source-code-git-repos” folder
2. The databases folder

This folder has UAA and FHIR backup schema with sample data.

1. The user-credentials folder

This folder has list of username and password to login after restore database.

1. The sof-images folder

This folder contains SMART on FHIR app icons which registered in OCP.

# FHIR Server Deployment on AWS

## Setup FHIR database server using RDS

1. Configure PostgreSQL server on AWS

AWS RDS is used to host the FHIR DB with RDS: db.t2.medium instance and PostgreSQL 9.6.3 Engine.

1. The default super user account name is ***postgres***
2. The default port is ***5432***

Once the installation is completed, open pgAdmin (GUI workbench for PostgreSQL) on your local to verify database server can be connected with the hostname/address.

1. Create a new database with the name *hapi*

## Setup FHIR APP in server

1. Prerequisite: Linux server (BHITS team utilizes EC2 with Amazon\_Linux\_AMI\_release\_2018.03)
2. Install Tomcat (tomcat8 is preferred)
3. Deploy fhir.war

* Stop tomcat
* Save fhir.war to the tomcat directory.

The default directory located at /var/lib/tomcat8/webapps/

* Save JPAServer.properties and Conformance.properties to the tomcat directory

The default directory located at /var/lib/tomcat8/webapps/fhir

* Start tomcat

1. Check the running status of FHIR by reading the tomcat log

sudo tail -1000f /var/log/tomcat8/catalina.out

## Configure FHIR APP to Use FHIR DB

Update the /var/lib/tomcat8/webapps/fhir/JPAServer.properties file to Integrate FHIR APP and DB

in=jdbc:postgresql://{fhir\_db\_endpoint}:{fhir\_db\_port}:/hapi

un={db\_username}

pd={db\_password}

1. This script file also can be found in companion folder, see Companion Folders and Files section. [↑](#footnote-ref-2)
2. This script file also can be found in companion folder, see Companion Folders and Files section. [↑](#footnote-ref-3)
3. This script file also can be found in companion folder, see Companion Folders and Files section. [↑](#footnote-ref-4)
4. This script file also can be found in companion folder, see Companion Folders and Files section. [↑](#footnote-ref-5)
5. [↑](#footnote-ref-6)