Department of Veterans Affairs

VistA Clinical Application & Enterprise Core Services

Installation Guide

for

Enterprise Health Management Platform (eHMP)

Contract Number: VA 118-1011-0013



September 2014

Version 1.0

Revision History

NOTE: The revision history cycle begins once changes or enhancements are requested after the Installation Guide Document has been baselined.

Date	Version	Description	Author
9/10/14	1.0	Final Prep to Post	[redacted]
9/10/14	0.3	Make another pass and post to DD.	[redacted]
9/8/14	0.2	Update Style and Perform Read Through	[redacted]
8/29/14	0.1	Initial Creation based on Developer's Guide template	[redacted]

Table of Contents

1. Introduction 1
   1.1. Purpose of this Document 1
   1.2. Scope 1
   1.3. Assumptions 1
2. Prerequisites 1
   2.1. Stable Build of eHMP to Deploy 2
   2.2. SMC Access to VA's Network 2
   2.3. Appropriately Configured Virtual Machines 2
   2.4. IP Addresses of the Virtual Machines 2
   2.5. User Account with Sudo Privileges 2
   2.6. SSH Private Key 2
3. Deployment Preparations 2
   3.1. Versions and Commit Hash 3
   3.2. Workspace Type 3
   3.3. Pre-Stage the Deployment Package 3
   3.4. Destroy Existing Virtual Machines 3
   3.5. Additional Deployment Parameter Information 4
4. Building the Deployment Package 4
   4.1. Checkout Desired Commit from GIT 4
   4.2. Set Build Environment Variables 4
   4.3. Deploy Desired Virtual Machines to 'Local' 5
      4.3.1. Remove Existing File Cache 5
      4.3.2. Ensure Enhanced Remote File Caching is Enabled 5
      4.3.3. Perform Complete Deployment of all Virtual Machine 5
   4.4. Build the Deployment Package 5
   4.5. Compress and Split the Deployment Package 6
5. Deploy to Virtual Machines 6
   5.1. Configure the Deployment JSON 6
   5.2. Checkout Desired GIT Commit 7
   5.3. Set Build Environment Variables 7
   5.4. Prepare the Target Virtual Machines for Deployment 8
      5.4.1. Prepare User Account 8

5.4.2. Add File Systems with Required Space Allocations 8
5.4.3. Update the UMASK 9
5.5. Stage Deployment Package 9
5.5.1. Copy the Deployment Package to the First Virtual Machine 9
5.5.2. Combine Compressed Parts and Extract 9
5.5.3. Deploy the File Cache to the Appropriate Locations 9
5.5.4. Repeat for Each Virtual Machine 9
5.6. Deploy Desired Virtual Machines 10
5.6.1. Link Vagrant to the Virtual Machine 10
5.6.2. Provision the Virtual Machine 10
5.6.3. Repeat for Each Virtual Machine 10
6. Post-Deployment Notes 10
Appendix A - Acronym List and Glossary 11

Table of Figures

No table of figures entries found.

Table of Tables

No table of figures entries found.

1. Introduction

The eHMP project within the Veterans Health Information Systems and Technology (VistA) Evolution program will introduce expanded capabilities and modernize existing features of the Department of Veterans Affairs (VA) VistA Electronic Health Record (EHR) system. This platform includes VistA Exchange, the eHMP Clinical Practice Environment (CPE), and particular clinical knowledge enrichment and decision support services. The eHMP project will produce epics (capabilities) supportive of the VistA 4 Product [see VistA 4 Product Roadmap] that relate to the core clinical user experience.

Based on the intended size and inherent cross-team and program complexities expected with the eHMP program, it is necessary to provide a scalable agile delivery framework that leverages proven agile principles, processes, and lightweight governance structures. The approach eHMP Engineering Management is taking is based on the Scaled Agile Framework (SAFe). SAFe is a proven, publicly-facing framework for applying lean and agile practices at enterprise scale while providing the necessary roadmap to adopt and tailor agile strategies to meet the unique challenges faced by the eHMP project-of-projects software development initiative.

1.1. Purpose of this Document

This Installation Guide will outline and describe the procedures involved in deploying (installing) the eHMP system into production within the Enterprise Operations (EO) environment. This document is a step-by-step guide to enable personnel responsible for the installation, presumed to be administrators within EO, to perform a full deployment in order to bring eHMP to an operation state.

1.2. Scope

Here are the instructions for modifying the local workstation configuration required to complete the deployment, obtain access to the required virtual machines (provided by AITC's EO environment), and linking vagrant to the virtual machines and provisioning said virtual machines. This guide provides the prerequisites required for the installation, the preparations required for the deployment, instructions on how to build the deployment package and instructions on how to deploy and provision the virtual machines. This installation guide covers the steps to prepare, build, and deploy the software package for eHMP 1.0. This is the first installation of VistA Exchange.

1.3. Assumptions

This guide assumes that the personnel responsible for the installation possesses the following:

- Final build of VistA Exchange
- SMC access to the VA's network
- Appropriately configured virtual machines
- IP addresses of aforementioned virtual machines
- User account with sudo privileges on each machine
- The SSH private key to access each machine

These assumptions will be described in 'Prerequisites' – Section 2.

2. Prerequisites

This section details each of the requirements outlined in the 'Assumptions' section above. These prerequisites should be met before beginning the installation of eHMP.

2.1. Stable Build of eHMP to Deploy

When performing the installation of the eHMP system, the administrator will have the option to choose a specific version (build) to deploy. It is required that a stable build be chosen to avoid compatibility and integration issues.

2.2. SMCAccess to VA's Network

Cisco Secure Mobility Client SMC access with enhanced capabilities is required to perform the installation. CAG access is not sufficient to complete the deployment of the system.

2.3. Appropriately Configured Virtual Machines

Proper installation requires a properly configured virtual machine for each server in addition to an appropriately configured Jenkins server within the VA. It is important that each of these virtual machines be configured with sufficient disk space for each file system. At this time, it is assumed that the virtual machines will be setup and correctly configured by the EO environment.

NOTE: The setup of the Jenkins server is beyond the scope of this document.

2.4. IP Addresses of the Virtual Machines

The administrator performing the installation will need the IP addresses of each virtual machine to be deployed as well as the IP address of the Jenkins server (if applicable).

2.5. User Account with Sudo Privileges

Deployment of the virtual machines requires a local user account on each virtual machine with the ability to use substitute user (sudo) commands. Sudo gives the user the security privileges of the superuser or root user.

2.6. SSH Private Key

An SSH key is required to authenticate access to each virtual machine in order to deploy. In an ideal situation, the SSH private key should be identical across all virtual machines to allow access to all machines with the same SSH key.

3. Deployment Preparations

This section describes the steps for identifying the appropriate action for certain decision points throughout the installation process. Prior to beginning the deployment process, the following parameters should be defined and written down:

- WAR_REMOTE_FILE
- FATJAR_VERSION
- COMMIT_HASH
- WORKSPACE_TYPE
- PRE_STAGE_PACKAGE

The process for defining the first four parameters is outlined in this section's subsection 3.1 and 3.2. Refer to subsection 3.3 and 3.4 for directions on selecting the proper values for the fifth parameter and to destroy existing virtual machines, respectively. Subsection 3.5 displays a table with more detailed descriptions of these parameters.

3.1. Versions and Commit Hash

For official deployment into the EO Environment, the WAR_REMOTE_FILE parameter should be set using the HMP_MAIN_WAR_VERSION supplied by a stable build in Jenkins. The FATJAR_VERSION parameter should be set using the BACKEND_VERSION value supplied as well. These values can be located by logging into Jenkins at: build.vistacore.us and navigating to:



*__NOTE__: 'bk-1.1.0.3311' should be replaced by the stable build version that has been selected for deployment. The COMMIT_HASH parameter can also be found at this location and should be stored for later use.

3.2. Workspace Type

Set the WORKSPACE_TYPE variable as follows:

- If you will be deploying using an OSX workspace to perform the deployment then set WORKSPACE_TYPE=Local
- If you will be deploying from a Jenkins server (whether from the command line or via a Jenkins job) then set WORKSPACE_TYPE=Jenkins

3.3. Pre-Stage the Deployment Package

Pre-staging the deployment package involved splitting the package into a number of parts, which will be uploaded to the virtual machines in series. This means that should one part fail to upload properly, only that part needs to be re-uploaded as opposed to the entire package, which would take significantly more time. Since this guide is limited to official deployments into the EO environment, the value of PRE_STAGE_PACKAGE should be set equal to 'Yes' and the steps for splitting the package as described in a later section should be followed.

3.4. Destroy Existing Virtual Machines

Building the deployment package in the next section requires that certain existing virtual machines be destroyed. Navigate to the following project folders in the terminal:

- /Users/*username*/Projects/vistacore/ehmp/infrastructure/vagrant/virtualbox/ede-simulated/
- /Users/*username*/Projects/vistacore/ehmp/infrastructure/vagrant/virtualbox/vista-exchange/

From the terminal, run the following command in each of the above directories:

```
$ vagrant destroy –f
```

3.5. Additional Deployment Parameter Information

Parameter Name	Description	Valid Values
WAR_REMOTE_FILE	This indicates the version of the HMP WAR file that will be deployed and is used to ensure that the version of the WAR file downloaded will correspond to the version of other artifacts including the GIT repo.	Version number
FATJAR_VERSION	This indicates the version of the BACKEND FATJAR file that will be deployed and is used to ensure that the version of the WAR file downloaded will correspond to the version of other artifacts including the GIT repo.	Version number
COMMIT_HASH	This indicates the commit hash within the EHMP GIT repo to be used to perform the deployment.	Commit hash
WORKSPACE_TYPE	This indicates whether you are deploying from a local workspace or a Jenkins server.	Local
PRE_STAGE_PACKAGE	This indicates whether you are pre-staging the deployment package. Typically, deployment packages would be pre-staged to reduce the time it takes to transfer package when there is slow connectivity between the workspace and the VMs being deployed as is typically true when performing deployments from a local workspace.	Yes

4. Building the Deployment Package

This section illustrates the process required for building the deployment package. Specifically, this section provides directions for executing a standard local deployment in order to build a cache of RPMs, gems, remote artifacts and downloaded cookbooks to be packaged up and later deployed to the EO environment. First the COMMIT_HASH will be used to pull the desired commit from GIT. Next the build environment variables will be set. Finally, the local virtual machines will be deployed and the deployment package will be built. This section also provides instructions for 'pre-staging' the package which includes compressing the archive and splitting that compressed file into smaller parts.

4.1. Checkout Desired Commit from GIT

In the terminal, from a directory in the GIT repository, execute the following command using the COMMIT_HASH parameter acquired in the previous section:

```
$ git checkout 93f8b704050f19a19285f09dea75aef508fe2e1a
```

NOTE: Be sure to replace the hash provided with the value of the parameter from Jenkins.

4.2. Set Build Environment Variables

Execute the following command in order to set the normal project environment variables:

```
$ source ~/Projects/vistacore/ehmp/infrastructure/set.env.sh
```

Next, run each of the following commands to set the build environment variables specific to this installation:

```
$ export WAR_REMOTE_FILE=1.0.0.1516
```

```
$ export FATJAR_VERSION=1.0.0.3308
```

NOTE: Be sure to replace the values of these parameters with the values acquired from the stable Jenkins build.

4.3. Deploy Desired Virtual Machines to 'Local'

This subsection details the steps required to populate the file cache which will be used to build the deployment package. First, the existing file cache must be completely removed, and a particular recipe added to the Vagrantfile.

4.3.1. Remove Existing File Cache

From within the workspace shell, change to infrastructure/vagrant/managed directory and run the following command to erase the file cache:

```
$ ruby deployment_package.rb clean
```

4.3.2. Ensure Enhanced Remote File Caching is Enabled

Typically, when deploying the virtual machines, Chef caches all remote files that are downloaded during deployment to an artifact cache located within the infrastructure/vagrant/virtualbox/.filecache hidden folder. Some Chef scripts, however download individual files to locations outside this file cache which would result in these files being omitted from the deployment package. In order to avoid this issue, an additional recipe should be added to infrastructure/vagrant/virtualbox/Vagrantfile by ensure the following line is added to that file:

```
chef.add_recipe "nointernet::setup_remote_file_caching"
```

This will cause remote files that are downloaded to other locations to be automatically copied into the filecache folder above.

4.3.3. Perform Complete Deployment of all Virtual Machine

Currently, this deployment is performed using Virtual Box (infrastructure/vagrant/virtualbox/vista-exchange/Vagrantfile) since Amazon Web Services (AWS) does not support bi-direction synced folders, however we may be moving this process to AWS in the future.

Deploy all virtual machines by executing the following command from the infrastructure/vagrant/virtualbox/vista-echange/ directory:

```
$ vagrant up
```

This will cause vagrant to deploy all virtual machines and populate the file cache will each remote file.

4.4. Build the Deployment Package

At this time, the file cache has been fully populated for all virtual machines. Additionally, this cache was produced by deploying the appropriate versions of the HMP_MAIN_WAR and BACKEND_JAR files as well as with the original GIT commit point. The following instruction compiles the file cache with additional Red Hat Enterprise Linux (RHEL) items into the deployment package.

To build the deployment package, change the directory of the workspace shell to the infrastructure/vagrant/managed folder and run the following command:

```
$ ruby deployment_package.rb build {directory}
```

The recommended value for {directory} is ~/Projects/vistacore which will keep the deployment package outside the GIT repository and prevent it from being committed.

A new folder named ehmp_deploy_YYYYMMDD will be created at this location.

At the time of this writing, it is also required to run the following command again because of the absence of a directory (**NOTE**: Due to the absence of {directory}) to build the deployment package in the infrastructure/vagrant/managed/.localcache/ directory:

```
$ ruby deployment_package.rb build
```

This will allow vagrant to find the remote files in the local workspace adjust the file paths accordingly so as to locate them on the virtual machine.

4.5. Compress and Split the Deployment Package

As mentioned previously, this guide is specific to official deployments into the EO environment. Therefore, the deployment parameter PRE_STAGE_PACKAGE has been set to 'Yes' and this subsection is required.

Execute each of the following commands (replacing YYYYMMDD with the appropriate date in that format):

```
$ cd {directory as specified in subsection 4.4}
$ tar czf ehmp_deploy_YYYYMMDD.tgz ehmp_deploy_YYYYDDMM
$ split –b 100m ehmp_deploy_YYYYMMDD.tgz ehmp_deploy_YYYYMMDD_part_
```

NOTE: It may be necessary to replace the '100m' parameter with a different size depending on the size of the total deployment package file.

This will yield a number of files in the current directory starting with the prefix "ehmp_deploy_YYYYMMDD_part_"

5. Deploy to Virtual Machines

This section details the steps required to deploy the deployment package from section 4 to the virtual machines in the EO environment.

5.1. Configure the Deployment JSON

This subsection describes how to create the deployment JSON file which is required by the deployment configuration. The following code should be copied into a JSON file and stored outside the GIT repository:

```
{
  "user" : "vagrant",
  "ssh_key_file" :
"/Users/vincegiusto/Projects/vistacore/.vagrant.d/insecure_private_key
",
  "file_cache_dir" : null,
  "disable_other_yum_repos" : false,
  "setup_timezone_and_ntp" : true,
  "ip" : {
    "jds" : "10.12.1.110",
    "solr" : "10.13.3.10",
```

```
    "aps" : "10.12.2.4",
    "vista-panorama" : "10.12.2.101",
    "vista-kodak" : "10.12.2.102",
    "ehmp" : "10.13.3.4",
    "ve-api" : "10.13.3.5",
    "lbalancer"      : "10.13.4.6"
  }
}
```

The recommended location of this JSON file is ~/Projects/vistacore/

The following are definitions for each item in the file:

user: This specifies the user that Vagrant should use to connect via SSH. For EDE and EO deployments, this must be set to the user that has an account on that server.

ssh_key_file: This specifies the SSH key that will allow Vagrant to connect via SSH on the virtual machines to be deployed. Ensure that this file has permissions 0600 or it will not work.

file_cache_dir: If you pre-staged the deployment package, then set this to "null".

setup_timezone_and_ntp: This should be set to 'false' when deploying to EDE or EO environments as these services have already been pre-configured on these virtual machines.

ntp_servers: For deployments to EDE and EO, set this to "['ntp.va.gov']" such that the servers use the VA's time service.

ip/*: These are the IP addresses of the virtual machines being deployed. These IP addresses should be acquired from EO. Refer to 'Prerequisites' subsection 2.4 for more information.

5.2. Checkout Desired GIT Commit

In the terminal, from a directory in the GIT repository, execute the following command using the COMMIT_HASH parameter (same as section 4.1):

```
$ git checkout 93f8b704050f19a19285f09dea75aef508fe2e1a
```

NOTE: Be sure to replace the hash provided with the value of the parameter from Jenkins.

5.3. Set Build Environment Variables

Be sure to run the following command to set the local environment variables:

```
$ source ~/Projects/vistacore/ehmp/infrastructure/set.env.sh
```

Additionally, execute the following command to set the DEPLOYMENT_CONFIG_JSON environment variable to the path where the deployment configuration JSON file was stored from subsection 5.1:

```
$ export DEPLOYMENT_CONFIG_JSON=Users/username/Projects/vistacore/eo-deployment-config.json
```

NOTE: Be sure to replace 'username' in the file path above with the current user and 'eo-deployment-config.json' with the file name of the deployment configuration JSON file.

5.4. Prepare the Target Virtual Machines for Deployment

The virtual machines provided by the EO environment have server configuration changes that must be made before deploying to them. These steps must be performed on each virtual machine that has not had them performed already.

5.4.1. Prepare User Account

Create and setup an SSH key for the account that will be used to perform this deployment. Also, ensure that this user is listed within the /etc/sudoers file and that this account does NOT require a password to run sudo commands.

5.4.2. Add File Systems with Required Space Allocations

The specific instructions for completing this step are outside the scope of this installation guide. Please refer to "Allocate Space and Add to Partitions on EDE" as a guide for accomplishing this task. In the EO environment, file systems in the root directory cannot be added or extend. The file systems should be created and allocated to be consistent with the following table:

VM	Added File Systems
aps	/var/chef (5GB) /var/yum_repo (5GB) /usr/ehmp_deploy (10GB)
solr	/var/chef (5GB) /var/yum_repo (5GB) /usr/ehmp_deploy (10GB) /opt/solr (10GB)
jds	/var/chef (5GB) /var/yum_repo (5GB) /usr/ehmp_deploy (10GB) /usr/cachesys (20GB)*
ve-api	/var/chef (5GB) /var/yum_repo (5GB) /usr/ehmp_deploy (10GB) /usr/local (10GB)
ehmp	/var/chef (5GB) /var/yum_repo (5GB) /usr/ehmp_deploy (10GB) /usr/local (10GB)*
lbalancer	/var/chef (5GB) /var/yum_repo (5GB) /usr/ehmp_deploy (10GB) TBD

5.4.3. Update the UMASK

Update the UMASK to 022 in the following files under /etc/profile.d: z-aitc.sh, z-aitc.ksh, and z-aitc.csh.a

5.5. Stage Deployment Package

This section provides instructions on how to copy the deployment package to the virtual machines, access each virtual machine to combine the compressed file parts back together, extract the compressed deployment package and run the shell script to deploy the file cache onto the virtual machine.

NOTE: Be sure to replace 'YYYYMMDD' with the date of the appropriate deployment package, use the IP address of the each virtual machine as provided by EO and specify the SSH key defined in the deployment config JSON when executing secure shell (ssh) and secure copy (scp) commands. Additionally, the username of the user account prepared in subsection 5.4.1 should be substituted for 'vagrant' in any of the following commands.

5.5.1. Copy the Deployment Package to the First Virtual Machine

From the workspace shell in the ~/Projects/vistacore/ directory, execute the following command to copy each part of the deployment package to the virtual machine:

```
$ scp –i .vagrant.d/insecure_private_key
ehmp_deploy_YYYYMMDD_part* vagrant@10.12.1.110:/usr/ehmp_deploy
```

5.5.2. Combine Compressed Parts and Extract

Execute the following command to access the virtual machine using SSH:

```
$ ssh –i .vagrant.d/insecure_private_key vagrant@10.12.1.110
```

Assuming the deployment package has been split as described in a previous section, run the following command to combine the compressed file parts back together on the virtual machine:

```
$ cat ehmp_deploy_YYYYMMDD_part* > ehmp_deploy_YYYYMMDD.tgz
```

Extract the compressed file on the virtual machine with the following command:

```
$ tar xzf ehmp_deploy_YYYYMMDD.tgz
```

5.5.3. Deploy the File Cache to the Appropriate Locations

Once the deployment package has been extracted to its original directory format on the virtual machine, change into the deployment directory and run the deployment script in order to place the package files into the appropriate folders on the virtual machine:

```
$ cd ehmp_deploy_YYYYMMDD
```

```
$ sudo ./deploy_localcache.sh
```

5.5.4. Repeat for Each Virtual Machine

Repeat the steps described in subsections 5.5.1 through 5.5.3 for each of the virtual machines.

To both improve efficiency and reduce transfer time, the deployment package can be copied from the first virtual machine in the EO environment on to each successive virtual machine.

NOTE: This would require the SSH key to be copied to an accessible location on the first virtual machine.

5.6. Deploy Desired Virtual Machines

This subsection outlines the process of linking and provisioning the virtual machines. The virtual machines should be deployed in a specific order as defined y the dependencies mentioned in the infrastructure/vagrant/aws/Rakefile.

Ensure that the environment variables have been set as previously mentioned. Refer to sections 4.2 and 5.3 for directions on this.

Execute both of the following commands from a shell within the infrastructure/vagrant/managed/manged-vista-exchange directory.

5.6.1. Link Vagrant to the Virtual Machine

This command does NOT actually bring the server up, but instead informs Vagrant where the server is located:

```
$ vagrant up {vm-name}
```

Replace {vm-name} with the name of the virtual machine. For example: solr, jds, aps, etc.

5.6.2. Provision the Virtual Machine

When using the managed provider, provisioning of the virtual machine does not occur up execution of 'vagrant up.' Provision the server using the following command:

```
$ vagrant provision {vm-name}
```

Replace {vm-name} with the name of the virtual machine. For example: solr, jds, aps, etc.

5.6.3. Repeat for Each Virtual Machine

Repeat the steps defined in subsections 5.6.1 and 5.6.2 for each virtual machine.

6. Post-Deployment Notes

At the time of this writing, the installation is considered to have completed successfully if every virtual machine has successfully been provisioned without any errors.

This section will be used for any additional information regarding the completion of the installation and next steps (if applicable).

Appendices

Appendix A - Acronym List and Glossary

Terms	Meaning
AITC	Austin Information Technology Center
API	Application Programming Interface
ASM	ASM Research
Cache	Component that stores existing data for quick access in the future
CPE	Clinical Practice Environment
CPRS	Computerized Patient Record System
EDE	Enterprise Development Environment
EHR	Electronic Health Record
EO	Enterprise Operations
eHMP	Enterprise Health Management Platform
GIT	Source Code Management and Version Control software
HMP	Health Management Platform
JAR	Java Archive
JDS	JSON Data Store
Jenkins	Build Repository
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LOINC	Logical Observation Identifiers Names and Codes
OIT	Office of Information Technology
RHEL	Red Hat Enterprise Linux
RPM	Red Hat Package Manager (file format with *.rpm extension)
SMC	Special Monthly Compensation an added compensation (paid in addition to the
SSH	Secure Shell
VA	Department of Veterans Affairs
Vagrant	Tool used for building complete deployment environments
VHA	Veterans Health Administration
VistA	Veterans Health Information Systems and Technology Architecture
VM	Virtual Machine
WAR	Web Application Archive
W3C	World Wide Web Consortium