

# Installation of Oracle OSB using Live-Scripting method

Andreas Wittmann

October 20, 2023

This project demonstrates the installation of Oracle FMW and OSB using the Live-Scripting method.

Documentation (PDF)     [live-scripting\\_osb.pdf](#)  
Documentation (HTML)   [live-scripting\\_osb.html](#)

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Architecture . . . . .	2
<b>2</b>	<b>Linux Installation</b>	<b>3</b>
2.1	Download and Check Sums . . . . .	3
2.2	Installing the VM . . . . .	3
2.3	Snapshot01 . . . . .	4
2.4	Conclusion . . . . .	5
<b>3</b>	<b>Database Setup</b>	<b>5</b>
3.1	Creating SSH Trust for the user <i>parallels</i> . . . . .	5
3.1.1	<b>DONE</b> Problem: SSH-Connection with public keys does not work. . . . .	6
3.2	Database Installation . . . . .	7
3.3	Autostart Configuration . . . . .	8
3.4	Database Operation . . . . .	8
3.5	Snapshot02 . . . . .	9
3.6	Conclusion . . . . .	9
<b>4</b>	<b>Preparing the FMW Installation</b>	<b>10</b>
4.1	Setup of the installation user . . . . .	10
4.2	Creating SSH Trust . . . . .	10
4.3	Copying the Oracle Distributions . . . . .	11
<b>5</b>	<b>Installation für FMW</b>	<b>12</b>
5.1	Templates . . . . .	12
5.2	Java Installation . . . . .	12
5.3	FWM Installation . . . . .	12
5.4	OSB Installation . . . . .	13

5.5	Cleaning up . . . . .	14
<b>6</b>	<b>Creating the OSB Domain</b>	<b>15</b>
6.1	Checking the database . . . . .	15
6.1.1	Starting and stopping the database . . . . .	15
6.1.2	Verifying the database listener . . . . .	15
6.2	Running the RCU tool . . . . .	16
6.3	OSB domain creation . . . . .	16
6.4	Starting the OSB domain . . . . .	17
6.5	Snapshot03 . . . . .	18
6.6	Conclusion . . . . .	19
<b>7</b>	<b>Documentation</b>	<b>19</b>
7.1	Project on GitHub . . . . .	19
7.1.1	Problem: Error during git push . . . . .	21
<b>8</b>	<b>Appendix</b>	<b>21</b>
8.1	Problem: Oracle Express Database needs RHEL 8. . . . .	21

## 1 Introduction

This project demonstrates the live-scripting process, which combines command-line-work and documentation in the context of an Oracle Service Bus installation.

### 1.1 Motivation

Oracle Service Bus (OSB) is widely utilized, boasting a substantial installation base. Although new projects are currently rare, its complex environment prevails. Expertise is often sought after for version migrations and the integration of new solutions. In the context of Oracle Service Bus (OSB) installation, live-scripting is a powerful approach that combines the execution of command-line operations with detailed documentation to create a reliable and well-documented installation process. Live-scripting benefits OSB installation in the following ways:

**Reproducibility:** Live-scripting ensures that the OSB installation process is reproducible, meaning that anyone following the documented steps can achieve the same results. This is particularly important in enterprise environments, where consistency and reliability are paramount. Even if the installation needs to be repeated months or years later, the documented process guarantees the same outcome.

**Documentation:** The process of live-scripting creates precise documentation that captures each step involved in the process. This documentation goes beyond simple command lists; it includes explanations, diagrams, attachments and solutions to potential issues that may arise. This detailed documentation is invaluable for troubleshooting, auditing, and knowledge transfer within an organization.

**Quality Assurance:** Live-scripting serves as a form of quality assurance. The fact that the documented process has been tested and verified during writing, ensures that the documented solution is reliably working and the chances of errors and misconfigurations is minimal.

**Knowledge Sharing:** The live-scripting approach makes knowledge sharing much more accessible. Team members can easily follow the step-by-step instructions, regardless of their level of expertise. Even if they don't want to use Emacs, the solution can be reproduced using a copy-and-paste approach. It accelerates the training process for new team members and can help disseminate best practices across the organization.

**Flexibility and adaptability:** Live-scripting isn't limited to a one-size-fits-all approach. It can be adapted and extended as needed. Users can modify the documented process to accommodate unique

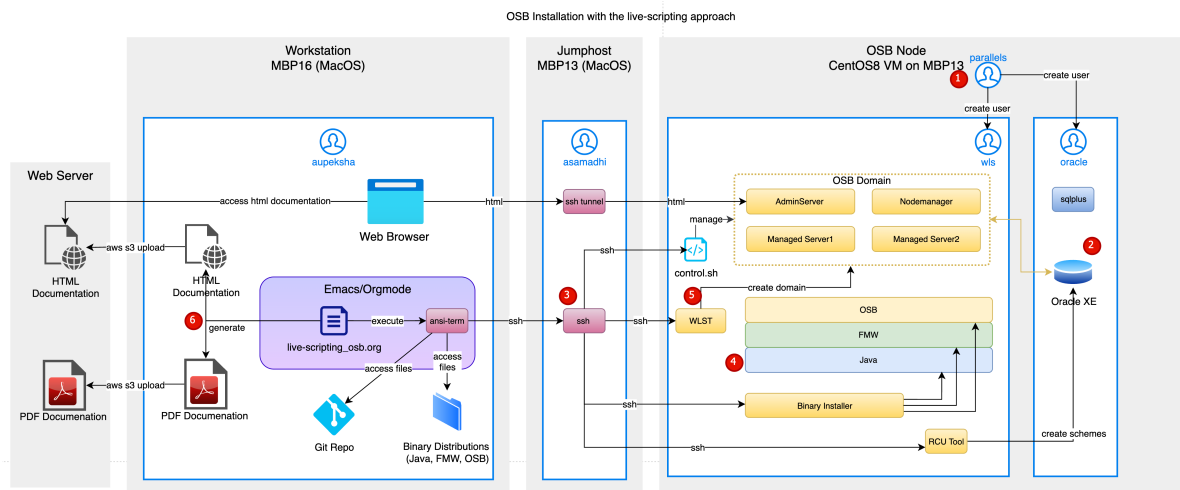
requirements or make improvements over time without sacrificing the stability and reliability of the installation.

**Efficiency:** Live-scripting streamlines the installation process by providing a clear and efficient path from start to finish. It minimizes the time and effort required to set up OSB, making it more accessible for both experienced and novice users.

In contrast to a fully automated setup, which is often used in large OSB installations, encompassing hundreds of domains and environments, the approach proposed by live-scripting does not require to write, test and maintain automation code. Live-scripting is best used in situations where only a limited number of environments are required. It ranges between a classical manual approach on the one hand and a fully automated solution on the other. In summary, live-scripting is a powerful methodology in the context of command-line-centered work, especially during OSB installation. It ensures a reproducible, well-documented, and reliable installation process, which is crucial in complex enterprise environments. This approach not only guarantees quality and consistency but also promotes knowledge sharing, adaptability, and efficiency in managing installations.

## 1.2 Architecture

The architecture of the OSB domain is simple, it consists of an admin server and two cluster nodes, all running on a single machine. The architecture diagram, however, also depicts the machines and components required to install and set up the OSB domain.



It uses three machines: A workstation for development and installation tasks, a jump host which adds a level of security and limits access to the OSB environment, and a Linux node for running all components of the OSB domain. Since this project demonstrates the live-scripting approach, the workstation contains an Emacs installation that is prepared to execute shell commands in an *ansi-term*. It uses ssh to connect to the target machine via the jump server. The whole installation process can conceptually be broken down into six steps, which are depicted by the red numbered dots in the diagram.

**Step 1:** In this step, the machine of the OSB node is provisioned. In my case, it is a virtual machine running on a MacBook Pro using Parallels as the hypervisor. The workstation is connected to the jump host using tailscale. It has no direct network connection to the VM. This situation resembles a typical setup in on-premise data centers and cloud environments. The Unix user *parallels* on this machine is used to create two additional users: *oracle* and *wls*, which will become the installation users for the database and the OSB environment, respectively.

**Step 2:** In this step, I install an Oracle Express Edition database, which is required for all FMW installations. It stores metadata for the operation of the OSB domain.

**Step 3:** During this step, I will provide the prerequisites for the subsequent installations. This includes establishing ssh trust for the workstation user *aupeksha* to access the jump host and the OSB-Node and to copy the software distribution to the target machine.

**Step 4:** This step consists of the software installation for Java, FMW and OSB. In a multi-machine environment, it would be necessary for every OSB node. In this simple example, however, there is only one machine.

**Step 5:** In this step, the OSB domain is created. This starts with creating the database schemas using the RCU tool. The domain will be created with the WLST (WebLogic Scripting Tool), which executes a Python script that runs the necessary installation and configuration tasks.

**Step 6:** The documentation is an integral part of the live-scripting approach. It is generated directly in the Emacs editor as a static HTML page and a PDF file. Both are published to a web server.

## 2 Linux Installation

I create a new CentOS VM, using *Parallels Desktop*.

### 2.1 Download and Check Sums

I use the Download Mirror. [https://ftp.fau.de/centos/8-stream/isos/x86\\_64/](https://ftp.fau.de/centos/8-stream/isos/x86_64/)

```

1 # Listing: Verifying Checksum of CentOS8 installation file in a terminal
2 ## On VM host machine: Samadhi
3 whoami; hostname; pwd
4
5 cd ~/Downloads
6 #SHA256 (CentOS-Stream-8-20231009.1-x86_64-dvd1.iso) =
7 ↪ e9b7a2026ac7abc315638bcb2efe9482576b4fd7ff0e8759c6ec6861b1549be8
8 shasum -a 256 CentOS-Stream-8-20231009.1-x86_64-dvd1.iso # ok.
```

**Result:** The ISO-image for the installation of Centos 8 is available for installation with parallels desktop. The checksum is verified against the values from the down load website.

### 2.2 Installing the VM

On the *Parallels Host* machine *Samadhi*, I will now install the ISO image with the Prrallels GUI Installer. Screenshots of the installation: [Screenshots.pdf](#) After the installation I login via ssh and check the os version .

```

1 # Listing: Basic Setup of CentOS VM
2 ### Login to Linux Machine via Jump Host
3 ssh asamadhi@samadhi #JumpHost
4 password1
5
6 #Centos8 VM
7 ## user/pw parallels/ham2burg
8 ssh parallels@10.211.55.5 # CentOS_8
9 ham2burg
10 whoami; hostname; pwd
11
12 ## Check Release
13 cat /etc/os-release
14 cat /etc/centos-release
```

The release is printed out on the terminal. It is CentOS Version 8.

```

1 # Listing: Displaying the OS Version
2 [parallels@localhost ~]$ ## Check Release
3 [parallels@localhost ~]$ cat /etc/os-release
4 NAME="CentOS Stream"
5 VERSION="8"
6 ID="centos"
7 ID_LIKE="rhel fedora"
8 VERSION_ID="8"
9 PLATFORM_ID="platform:el8"
10 PRETTY_NAME="CentOS Stream 8"
11 ANSI_COLOR="0;31"
12 CPE_NAME="cpe:/o:centos:centos:8"
13 HOME_URL="https://centos.org/"
14 BUG_REPORT_URL="https://bugzilla.redhat.com/"
15 REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux 8"
16 REDHAT_SUPPORT_PRODUCT_VERSION="CentOS Stream"
17 [parallels@localhost ~]$ cat /etc/centos-release
18 CentOS Stream release 8

```

I want to allow user *parallels* to sudo without password. This can be done with visudo which is an interactive editing task.

```

1 # Listing: Sudo configuration for user parallels
2 # User /parallels/ is allowed to sudo without password
3
4 # Use ~sudo visudo~ and insert this line.
5 /parallels ALL=(ALL) NOPASSWD: ALL/

```

**Result:** The CentOS8 VM is installed. Access via ssh is working. The user *parallels* can execute sudo without password.

### 2.3 Snapshot01

I take a snapshot of the CentOS\_8 VM with the following contents.

**Snapshot01:** [2023-10-12 Thu 12:03]

- ☒ Download CentOS Stream 8 using the Parallels Desktop GUI
- ☒ Installation CentOS Stream 8
- ☒ User=parallels, password=ham2burg
- ☒ Parallels Tools Installation
- ☒ Automatic Login, Screensaver off, Autopower-off=false.
- ☒ Settings Network->Wired= on
- ☒ SSH Login with: parallels@10.211.55.5
- ☒ yum update
- ☒ Renaming VM to *CentOS\_8*
- ☒ Allocating 4 CPUs and 8 GB RAM
- ☒ Creating *Snapshot01*

### 2.4 Conclusion

The Centos\_8 VM is set up. Remote login with user parallels is working. A snapshot of the VM is saved.

### 3 Database Setup

The FMW-installation requires a database for configuration data and runtime metrics. In this step I will setup and configure the *Oracle Express Edition Database*.

#### 3.1 Creating SSH Trust for the user *parallels*

[2023-10-12 Thu 12:21] I want to have key-based ssh-login for the users *parallels*. It should share a common key-pair with other users: *lab\_key*.

```

1  # Listing: Creating SSH Trust for user parallels
2
3  ### We start on the workstation
4  ## Test connection to JumpHost and VM
5  ssh asamadhi@samadhi "hostname; whoami" # JumpHost via tailscale
6  ssh -J asamadhi@samadhi parallels@10.211.55.5 "hostname; whoami"
7  password1
8  password2
9  ### Create public/private key-pair for communication with multiple machines on jump host
10 hostname; whoami
11 cd ~/.ssh
12 ls -la
13 ssh-keygen -t ed25519 -C "lab_key"
14 lab_key
15
16
17 ## copy public+private key to jumphost
18 ls -la
19 scp asamadhi.pub asamadhi@samadhi:/Users/asamadhi/.ssh/
20 scp lab_key.pub asamadhi@samadhi:/Users/asamadhi/.ssh/
21 scp lab_key asamadhi@samadhi:/Users/asamadhi/.ssh/
22 password1
23 ## paste public key to authorized keys for user asamadhi
24 ssh asamadhi@samadhi "cat /Users/asamadhi/.ssh/lab_key.pub >> /Users/asamadhi/.ssh/authorized_keys"
25 ssh asamadhi@samadhi "chmod 600 /Users/asamadhi/.ssh/authorized_keys"
26 ssh asamadhi@samadhi "cat /Users/asamadhi/.ssh/authorized_keys"
27 ssh asamadhi@samadhi "ls -la /Users/asamadhi/.ssh/"
28 password1
29
30 ## Add config entry for jumphost
31 cat ~/.ssh/config
32 cat << 'EOF' >> ~/.ssh/config
33 Host jumphost
34     IdentitiesOnly yes
35     IdentityFile ~/.ssh/lab_key
36     Hostname samadhi
37     User asamadhi
38 EOF
39
40 # Test ssh trust
41 ssh -i ~/.ssh/lab_key asamadhi@samadhi "whoami; hostname; date"
42 ssh jumphost "whoami; hostname; date"
43
44 ### copy public key to VM from JumpHost
45 ssh -i ~/.ssh/lab_key asamadhi@samadhi
46 hostname; whoami; pwd
47 ls -la ~/.ssh/
48 scp ~/.ssh/lab_key.pub parallels@10.211.55.5:/tmp/lab_key.pub
49 ham2burg

```

```

50 ## paste public key to authorized keys for user parallels
51 ssh parallels@10.211.55.5
52 ham2burg
53 mkdir -p /home/parallels/.ssh
54 cat /tmp/lab_key.pub > /home/parallels/.ssh/authorized_keys
55 chmod 600 /home/parallels/.ssh/authorized_keys
56 cat /home/parallels/.ssh/authorized_keys
57 exit
58 # Test ssh trust on jumphost to CentOS8
59 ssh -i ~/.ssh/lab_key parallels@10.211.55.5 "whoami; date"
60 ls -la ~/.ssh/
61
62 exit #samadhi
63
64 ### Access via JumpHost
65 ssh -J jumphost -i ~/.ssh/lab_key parallels@10.211.55.5 "whoami; hostname; date"
66
67 ## Add config entry for centos VM
68 cat ~/.ssh/config
69 cat << 'EOF' >> ~/.ssh/config
70 Host centos8_parallels
71     IdentitiesOnly yes
72     IdentityFile ~/.ssh/lab_key
73     Hostname 10.211.55.5
74     User parallels
75     ProxyJump jumphost
76 EOF
77
78 ## Test direct access to centos VM via jumphost using config file
79 ssh centos8_parallels "whoami; hostname; pwd"

```

**Result:** SSH Trust is now established for the users *parallels* using config file and key-authentication. This enables password-less ssh access via a jumphost to the centos VM.

### 3.1.1 DONE Problem: SSH-Connection with public keys does not work.

[2023-10-12 Thu 12:52] Although I provide a valid key, the ssh command prompts for a password.

**Analysis:** I ran ssh with debug flag (-vvv)

```

1 # Listing: Debug output from the SSH command
2 debug1: Offering public key: /Users/asamadhi/.ssh/lab_key ED25519
3   ↳ SHA256:VlBgbKyKKJocYsRRVD9Qi9nN9XFDXf2uXN1wGf2P4Dg explicit
4 debug3: send packet: type 50
5 debug2: we sent a publickey packet, wait for reply
6 debug3: receive packet: type 51
7 Debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-mic,password

```

**Solution:** Changing the permissions of the directory *.ssh* on the CentOS8 machine solves the problem.  
`chmod 700 ~/.ssh`

## 3.2 Database Installation

The database can be downloaded: [Oracle Database Express Edition \(XE\) Downloads | Oracle Deutschland](#)

Oracle Database 21c Express Edition for Linux x64 ( OL8 ) (2,339,651,768 bytes - September 08, 2021) [S

### 3 Database Setup

Now I copy the database binaries to the target machine and start the installer. After successful installation I configure a database and verify that it is working and the the listener is started.

```
1  # Listing: Installation of Oracle Database including Smoke Tests.
2
3  ## preparing target dist directory
4  ssh centos8_parallel
5  whoami; hostname; pwd
6  sudo su
7  ham2burg
8  mkdir -p /opt/install/db
9  chown parallels:parallels /opt/install
10 chown parallels:parallels /opt/install/db
11 exit #root
12 exit #centos8_parallel
13
14 ## Checking and copying distribution form local workstation to centos VM
15 cd ~/LocalProjects/dist/db
16 ls -la
17
18 ## Check SHA256 of RPM
19 #Sha256sum: f8357b432de33478549a76557e8c5220ec243710ed86115c65b0c2bc00a848db
20 sha256sum oracle-database-xe-21c-1.0-1.el8.x86_64.rpm # ok.
21
22
23 ## Copy distributions VM
24 ls -la
25 rsync -aP oracle-database-preinstall-21c-1.0-1.el8.x86_64.rpm \
26         oracle-database-xe-21c-1.0-1.el8.x86_64.rpm centos8_parallel:/opt/install/db
27
28 ## Check distributions as user parallels
29 ssh centos8_parallel "ls -la /opt/install/db"
30
31 ### Installation of Database as root
32 ssh centos8_parallel
33 whoami; hostname; pwd
34 sudo su -
35
36 ## Preinstallation
37 cd /opt/install/db
38 yum -y localinstall oracle-database-preinstall-21c-1.0-1.el8.x86_64.rpm
39
40 ## Install Database
41 yum -y localinstall oracle-database-xe-21c-1.0-1.el8.x86_64.rpm
42 # ok.
43
44 ### Creating a Database
45 ls /etc/init.d/oracle-xe*
46 /etc/init.d/oracle-xe-21c configure
47 ham2burg
48 ham2burg
49 # ok.
50
51 exit #root
52 whoami
53 ### Testing the Database: as user user oracle
54 sudo su - oracle
55 export ORACLE_SID=XE
56 export ORAENV_ASK=NO
57 ls /opt/oracle/product/21c/dbhomeXE/bin/oraenv
```



```

58 . /opt/oracle/product/21c/dbhomeXE/bin/oraenv
59 set ORACLE_HOME = [] ? /opt/oracle/product/21c/dbhomeXE
60 echo $ORACLE_HOME
61 ls -la $ORACLE_HOME
62
63 ## start sqlplus
64 #sqlplus sys/ham2burg@//localhost.localdomain:1521/XEPDB1 as sysdba
65 sqlplus sys/ham2burg@//localhost:1521/XEPDB1 as sysdba
66
67 ## Check PDBs
68 select name from v$pdb;
69
70 ## connecting to default PDB
71 #conn sys/ham2burg@//localhost.localdomain:1521/XEPDB1 as sysdba
72 conn sys/ham2burg@//localhost:1521/XEPDB1 as sysdba
73 ## Ok. Successfully connected.
74
75 exit #sqlplus
76 exit #oracle
77
78 ## check listener
79 lsnrctl status

```

**Result:** The Oracle database is now installed. The listener is configured and started. Connection via sqlplus is working.

### 3.3 Autostart Configuration

I want to configure the database for automatic start after machine boot. This is done on the Linux level.

```

1 # Listing: Configuration of automatic database start
2 ssh centos8_parallel
3 whoami; hostname
4
5 sudo su
6
7 systemctl daemon-reload
8 systemctl enable oracle-xe-21c
9
10 exit #root
11 exit #parallels

```

**Result:** The linux operating system is configured to start the database service automatically on machine boot.

### 3.4 Database Operation

The database can be started and stop with the *sysctl utility*.

```

1 # Listing: Starting, stopping and testing the database
2 ## work on the CentOS8 as user parallels.
3 ssh centos8_parallel
4 whoami; hostname
5
6 # Show Usage Message
7 sudo /etc/init.d/oracle-xe-21c
8 # Datenbank starten

```

```

9  sudo /etc/init.d/oracle-xe-21c status
10 sudo /etc/init.d/oracle-xe-21c stop
11 sudo /etc/init.d/oracle-xe-21c start
12
13 ### Checking the Listener
14
15 ### User must be oracle
16 sudo su - oracle
17 ### Setting Environment and checking listener
18 export ORACLE_SID=XE
19 export ORAENV_ASK=NO
20 . /opt/oracle/product/21c/dbhomeXE/bin/oraenv
21 set ORACLE_HOME = [] ? /opt/oracle/product/21c/dbhomeXE
22 echo $ORACLE_HOME
23 ls -la $ORACLE_HOME
24
25 ## check listener
26 lsnrctl status
27 exit #oracle
28 exit #centos8_parallel
29 whoami

```

**Result:** This listing demonstrates how to control the database. It can be used to check the status of the database and the listener and to start and stop the database.

### 3.5 Snapshot02

I take a snapshot of the CentOS\_8 VM with the following contents.

**Snapshot02:** *[2023-10-12 Thu 14:45]*

- ☒ Uploading RPMs for Oracle-Database-XE-21c Database
- ☒ Setting up SSH-Trust for user parallels.
- ☒ Installation of Oracle-Database-XE-21c Database
- ☒ Testing the database
- ☒ Lifecycle operations on database
- ☒ Automatic database start after reboot is tested.
- ☒ VM is powered of
- ☒ Creating *Snapshot02*

### 3.6 Conclusion

At this stage of development, the Oracle-Database-XE-21c is installed and starts automatically with the VM. The commands to start and stop the database are tested.

## 4 Preparing the FMW Installation

### 4.1 Setup of the installation user

I want to install OSB with a dedicated unix user: *wls*. It should be a login user and I want to establish ssh-trust for key-based ssh login.

```

1 # Listing: Creating unix user wls and setting the password
2 ### Login to VM
3 ssh centos8_parallel

```

```

4
5 ### create user wls
6 sudo useradd --defaults
7 sudo useradd --comment "OSB User" --create-home --user-group wls
8 sudo passwd wls
9 ham2burg
10 ham2burg
11
12 exit

```

**Result:** The user *wls* is now created and can login with a password.

## 4.2 Creating SSH Trust

[2023-10-06 Fri 10:06] I want to have key-based login for the user *wls*. It should share a common key-pair: *lab\_key*. Therefore the user *wls* must accept the public key *lab\_key.pub*.

```

1 # Listing: Creating SSH Trust for user wls
2 ### We start on the workstation
3 ## copy public key to the VM
4 whoami; hostname
5
6 cd ~/.ssh/
7 ls -la
8 scp ~/.ssh/lab_key.pub centos8_parallel:~/tmp/lab_key.pub
9 # Add public key to authorized hosts.
10 ssh centos8_parallel
11
12 ## paste public key to authorized keys for user wls
13 sudo su - wls
14 mkdir -p /home/wls/.ssh/
15 chmod 700 /home/wls/.ssh
16 cat /tmp/lab_key.pub > /home/wls/.ssh/authorized_keys
17 chmod 600 /home/wls/.ssh/authorized_keys
18 exit #wls
19 exit #parallels
20
21
22 # Test ssh trust
23 ssh -J jumphost -i ~/.ssh/lab_key wls@10.211.55.5 "whoami; hostname; date"
24
25 ## Add config entry for centos VM
26 cat ~/.ssh/config
27 cat << 'EOF' >> ~/.ssh/config
28 Host centos8_wls
29     IdentitiesOnly yes
30     IdentityFile ~/.ssh/lab_key
31     Hostname 10.211.55.5
32     User wls
33     ProxyJump jumphost
34 EOF
35
36 ## Test direct access to centos VM via jumphost using config file
37 ssh centos8_wls "whoami; hostname; pwd"

```

**Result:** SSH trust is now established for the users *parallels* and *wls* using config file and key-authentication. This enables password-less ssh access via a jumphost to the centos VM.

### 4.3 Copying the Oracle Distributions

I have already downloaded the distribution for java, fmw, and osb to the workstation. In this step I copy it to the target machine.

```

1  # Listing: Copying installation media files using rsync.
2  ###
3  whoami; hostname
4  ssh centos8_parallel
5  ### create project installation base
6  sudo mkdir -p /opt/install/
7  sudo chown -R wls:wls /opt/install
8  ls -la /opt/install
9  find /opt/install
10 exit
11
12 # create dir for distributions as wls user
13 ssh centos8_wls
14 date; whoami; hostname
15 mkdir -p /opt/install/dist
16 exit
17
18
19 ## Copy distributions (from workstations)
20 whoami; date; hostname
21 cd /Users/aupeksha/LocalProjects/dist/osb
22 ls -la
23 rsync -aP fmw_12.2.1.4.0_infrastructure_Disk1_1of1.zip \
24           p18143322_1800_Linux-x86-64.zip \
25           p30188261_122140_Linux-x86-64.zip \
26           p30188305_122140_Generic.zip      centos8_wls:/opt/install/dist
27
28 ## Check distributions as user wls
29 ssh centos8_wls "ls -la /opt/install/dist"

```

I also copy the project files to the VM with rsync. In this way I can edit the file on the workstation and update changed files.

```

1  # Listing: Copying project files using rsync.
2  cd /Users/aupeksha/LocalProjects/live-scripting_osb/
3  ## Copy or Update directory
4  rsync -aP live-scripting_fmw centos8_wls:/home/wls/
5
6  ## Check distributions as user wls
7  ssh centos8_wls "ls -la /home/wls/live-scripting_fmw"
8  ssh centos8_wls "find /home/wls/live-scripting_fmw"
9  ##

```

**Result:** The project files and software distributions are now available on the target machine under the user wls.

## 5 Installation für FMW

### 5.1 Templates

I want to use template mechanism to replace variables in configuration files. There is discussion on stackoverflow that compares different approaches. [Bash Templating: How to build configuration files from](#)

[templates with Bash? - Stack Overflow](#) I will use the templating approach using *substenv*. I will define some environment variables in the file *setEnv\_osb.sh*, which will be sourced to the shell environment. Configuration and script files can now contain shell variables that will be replaced during the installation. Thus I can support different stages and environments with one set of script and configuration files.

## 5.2 Java Installation

Java is unpacked (unzip) to a temporary directory and subsequently installed to the target Java directory using tar.

```

1  # Listing: Java installation
2  ### Install Java as user wls
3  ssh centos8_wls
4  date; hostname; pwd;  whoami
5
6  # set project environment
7  cd ~/live-scripting_fmws; . ./setEnv_osb.sh
8
9  ## unzip to temporary directory
10 mkdir -p $JAVA_TEMP_DIR
11 unzip $DIST_JAVA -d $JAVA_TEMP_DIR
12
13 ## untar jav
14 mkdir -p $JAVA_DIR
15 ls ${JAVA_TEMP_DIR}/jdk*.tar.gz
16 tar xzvf ${JAVA_TEMP_DIR}/jdk*.tar.gz -C $JAVA_DIR
17
18 ls $JAVA_DIR
19 echo $JAVA_HOME
20
21 ## check java
22 $JAVA_HOME/bin/java -version # java(TM) SE Runtime Environment (build 1.8.0_311-b25)
23 $JAVA_HOME/bin/jar -version
24
25 ## clean up
26 rm -rf $JAVA_TEMP_DIR
27 ls $JAVA_TEMP_DIR
28
29 exit #centos_wls

```

**Result:** Java is installed into a dedicated directory and can be called after setting the `JAVA_HOME` variable.

## 5.3 FWM Installation

Now I will install the *Fusion Middleware* binaries on the target machine. A response file will be prepared using the template approach.

```

1  # Listing: FWM Installation using wls user
2  ssh centos8_wls
3  date; hostname; pwd;  whoami
4
5  # set project environment
6  cd ~/live-scripting_fmws; . ./setEnv_osb.sh
7
8  # Installation of binaries
9  mkdir -p ${TEMP_DIR_FMW}

```

```

10 unzip ${DIST_FMW} -d ${TEMP_DIR_FMW}
11 ls -la ${TEMP_DIR_FMW}
12
13 # Creating Response File
14 cd $PROJECT_HOME
15
16 envsubst < fmw.rsp.template > fmw.rsp
17 cat fmw.rsp
18
19 ## Creating inventory log file
20 cat << EOF > ${INV_PTR_LOC}
21 inventory_loc=/home/wls/oraInventory
22 inst_group=wls
23 EOF
24
25 cat ${INV_PTR_LOC}
26 ls -la
27 # Checking prerequisites
28 ls ${TEMP_DIR_FMW}/*.jar
29 ${JAVA_HOME}/bin/java -Djava.io.tmpdir=${TEMP_DIR_FMW} -jar ${TEMP_DIR_FMW}/*.jar \
30     -responseFile ${PROJECT_HOME}/fmw.rsp \
31     -invPtrLoc ${INV_PTR_LOC} \
32     -silent \
33     -executeSysPrereqs
34
35
36 ## Installing FMW
37 ${JAVA_HOME}/bin/java -Djava.io.tmpdir=${TEMP_DIR_FMW} -jar ${TEMP_DIR_FMW}/*.jar \
38     -responseFile ${PROJECT_HOME}/fmw.rsp \
39     -invPtrLoc ${INV_PTR_LOC} \
40     -silent \
41     -logLevel finest \
42     -debug \
43     -printdiskusage \
44     -printmemory \
45     -printtime
46
47 # clean up
48 rm -rf ${TEMP_DIR_FMW}
49
50 exit # centos_wls

```

**Result:** The FMW Binaries are now installed.

## 5.4 OSB Installation

I now install the binaries for the OSB.

```

1 # Listing: OSB installation using wls user
2 ssh centos8_wls
3 date; hostname; pwd; whoami
4
5 # set project environment
6 cd ~/live-scripting_fmw; . ./setEnv_osb.sh
7
8 # Installation of binaries
9 mkdir -p ${TEMP_DIR_FMW}
10 unzip ${DIST_OSB} -d ${TEMP_DIR_FMW}
11 ls -la ${TEMP_DIR_FMW}

```

```

12
13 # Creating Response File
14 cd $PROJECT_HOME
15 envsubst < osb.rsp.template > osb.rsp
16 cat osb.rsp
17
18
19 ls -la
20 # Checking prerequisites
21 ls ${TEMP_DIR_FMW}/*.jar
22 ${JAVA_HOME}/bin/java -Djava.io.tmpdir=${TEMP_DIR_FMW} -jar ${TEMP_DIR_FMW}/*.jar \
23     -responseFile ${PROJECT_HOME}/osb.rsp \
24     -invPtrLoc $INV_PTR_LOC \
25     -silent \
26     -executeSysPrereqs
27
28
29 ## Installing OSB
30 ${JAVA_HOME}/bin/java -Djava.io.tmpdir=${TEMP_DIR_FMW} -jar ${TEMP_DIR_FMW}/*.jar \
31     -responseFile ${PROJECT_HOME}/osb.rsp \
32     -invPtrLoc $INV_PTR_LOC \
33     -silent \
34     -logLevel finest \
35     -debug \
36     -printdiskusage \
37     -printmemory \
38     -printtime
39
40 # clean up
41 rm -rf ${TEMP_DIR_FMW}
42 exit #centos_wls

```

**Result:** The OSB binaries are installed on the target machine. It is ready for domain creation.

## 5.5 Cleaning up

After installing the binaries, I delete the installation medias, to save space on the VM. Otherwise it will unnecessarily waste disk space in the snapshots.

```

1 # Listing: Cleaning up the VM
2 whoami; hostname
3 ## Login to VM as parallels users
4 ssh centos8_parallel
5 cd /opt/install/dist
6 ls -la
7 ## delete installation media for middleware
8 sudo rm p*
9 sudo rm fmw*
10 ## delete database installation media
11 cd /opt/install/db
12 sudo rm oracle*
13 ls -ls
14 exit

```

**Result:** The installation medias for the database and for java, fmw and osb are now removed from the vm, thus saving disk space in snapshots.

## 6 Creating the OSB Domain

There are several steps necessary to create the OSB domain. In a first step I will create the required database schemes using the RCU tool. The domain is will be created without the GUI installer. Instead WLST is used together with a python script, which runs the installation. The python script will be customized, using the template approach.

### 6.1 Checking the database

Before running the RCU tool I need to check that the database is started and listening.

#### 6.1.1 Starting and stopping the database

This is done using the init.d script.

```

1  # Listing: Starting and checking the database
2  ssh centos8_parallel1s
3  # Zeige die Usage Message
4  sudo /etc/init.d/oracle-xe-21c
5  # Datenbank starten
6  sudo /etc/init.d/oracle-xe-21c start
7  sudo /etc/init.d/oracle-xe-21c restart
8  sudo /etc/init.d/oracle-xe-21c status

```

Ok, the database is running. I know check the TNS listener.

#### 6.1.2 Verifying the database listener

I change to the user oracle and set the environment for checking the listener with *lsnrctl*

```

1  # Listing: Checkomg the database listener
2  ### User must be oracle
3  sudo su - oracle
4  ### Setting Envriionment and checking listner
5  export ORACLE_SID=XE
6  export ORAENV_ASK=NO
7  . /opt/oracle/product/21c/dbhomeXE/bin/oraenv
8  set ORACLE_HOME = [] ? /opt/oracle/product/21c/dbhomeXE
9  echo $ORACLE_HOME
10 ls -la $ORACLE_HOME
11
12 ## check listener
13 lsnrctl status
14
15 exit #oracle
16 exit #centos8_parallel1s

```

**Result:** The database and the listner are started.

### 6.2 Running the RCU tool

I will use the RCU-Tool to create the database schemes, which is a prerequisite for OSB domain creation. I use the template approach to customize the response file. The execution of the RCU-Tool takes several minutes.



```

1  # LISTING: Creating database schemes with the RCU tool.
2  ### RCU execution using wls user
3  ssh centos8_wls
4  date; hostname; pwd; whoami
5
6  # set project environment
7  cd ~/live-scripting_fmws; . ./setEnv_osb.sh
8
9  # Creating Response File
10 cd $PROJECT_HOME
11 ls -la
12 envsubst < rcu_osb.rsp.template > rcu_osb.rsp
13 cat rcu_osb.rsp
14
15 ### Checking prerequisites. It needs:
16 # - database password for user sys
17 # - schema password
18 ${FMW_HOME}/oracle_common/bin/rcu -silent -responseFile ${PROJECT_HOME}/rcu_osb.rsp -validate
19 ham2burg
20 ham2burg
21 ## Success=status 0
22
23 ### Running RCU
24 ${FMW_HOME}/oracle_common/bin/rcu -silent -responseFile ${PROJECT_HOME}/rcu_osb.rsp
25 ham2burg
26 ham2burg
27
28 exit #centos_wls

```

**Result:** The database schemes for the OSB domain are created. The database is ready for osb domain creation.

### 6.3 OSB domain creation

I will now create the domain using WLST and the python script *createOSBDomain.py*. The python script originates from the OSB documentation and was modified to accept customization via the template approach.

```

1  # Listing: Creating an OSB domain using wls user
2  ssh centos8_wls
3  date; hostname; pwd; whoami
4
5  # set project environment
6  cd ~/live-scripting_fmws; . ./setEnv_osb.sh
7
8  # Creating WLST File
9  cd $PROJECT_HOME
10 ls -la
11 envsubst < Template_createOSBDomain.py > createOSBDomain.py
12 cat createOSBDomain.py
13
14 ## Create the domain
15 mkdir -p $DOMAINS
16 cd $PROJECT_HOME
17 ls -la
18
19 cd $FMW_HOME/oracle_common/common/bin/
20 ./wlst.sh $PROJECT_HOME/createOSBDomain.py -oh "$FMW_HOME" -jh "$JAVA_HOME" -parent "$DOMAINS" -rcuDb
    ↪ "${CONNECT_STRING2}"

```

```
21  
22 exit # centos_wls
```

**Result:** The osb domain is created and can be started in the next step.

## 6.4 Starting the OSB domain

I will use the start scripts from the OSB installation to start the Nodemanager and the Adminserver. As a prerequisite I create boot.properties files which allow for password-less script-based domain start from the command line.

I also introduce a small self-developed control script to start and stop the domain and to produce a log file report, based on a perl script.

```

1 # Listing: Starting and Managing the OSB
2 ssh centos8_wls
3 date; hostname; pwd; whoami
4
5 # set project environment
6 cd ~/live-scripting_fmw; . ./setEnv_osb.sh
7 cd $DOMAIN_HOME; ls -la
8
9 ### Creating boot.properties
10 mkdir -p $DOMAIN_HOME/servers/AdminServer/security
11 cd $DOMAIN_HOME/servers/AdminServer/security
12 ls -la; pwd
13 echo username=${DOMAIN_USER} > boot.properties
14 echo password=${DOMAIN_PASSWORD} >> boot.properties
15 cat boot.properties
16 #rm boot.properties
17
18 ### Start the Nodemanager
19 cd $DOMAIN_HOME
20 #./bin/startNodeManager.sh
21 nohup ./bin/startNodeManager.sh > NodeManager.out 2>&1 &
22
23 ## Start AdminServer (nohup)
24 nohup ./startWebLogic.sh > AdminServer.out 2>&1 &
25
26 ## Check status
27 tail -f AdminServer.out
28 cat AdminServer.out | grep -e "Server state changed to"
29 cat AdminServer.out | grep -e "Server state changed to RUNNING."
30 cat servers/AdminServer/logs/AdminServer.log | grep "Server state changed to"
31 cat servers/AdminServer/logs/AdminServer.log | grep "Server state changed to RUNNING"
32
33 ## Wait for running state
34 while (( $(cat AdminServer.out | grep "Server state changed to RUNNING." | wc -m ) < 1 ))
35 do
36     sleep 5
37     cat AdminServer.out | grep "Server state changed to "
38 done
39
40 ## prevent perl warnings
41 export LC_ALL=C
42
43 ### Check for Running Server and NodeManager
44 ps -aelf | perl -ne '/^(?[\s]+\s+(?[\s]+\s+(?[\s]+\s+(?[\s]+\s+.*([w]eblogic.NodeManager)/g && print
   → "$3 $4 $5\n" or /^(?[\s]+\s+(?[\s]+\s+(?[\s]+\s+(?[\s]+\s+.*([w]eblogic.Name=[^\s]+)/g && print "$3
   → $4 $5\n"'

```

```

45
46 ### Use the control script
47 cd /home/wls/live-scripting_fmw/bin
48 ls -la
49
50 ## use control skript to control the domain
51 ./control.sh help
52 ./control.sh status
53 ./control.sh log_report -u
54 ./control.sh log_report -T
55 ./control.sh log_report -rwaS
56 ./control.sh kill
57
58 ./control.sh startNodemanager
59 ./control.sh startAdmin
60
61 exit # centos_wls
62
63
64 ### create tunnel for access via web browser.
65 whoami; hostname
66 ssh -L 7001:localhost:7001 centos8_wls
67 exit # centos_wls

```

The OSB is started and access to the web based administration tools is available.

#### Test URLs:

- <http://localhost:7001/console> (wls/ham2burg)
- <http://localhost:7001/em>
- <http://localhost:7001/servicebus>

**Result:** The OSB domain is created and started. I also started SSH tunnel that allow to call the OSB Web-Tools without direct network connection.

## 6.5 Snapshot03

I take a snapshot of the CentOS 8 VM with the following contents:

**Snapshot03:** *[2023-10-14 Sat 12:34]*

- ☒ Creating unix user wls
- ☒ Setting up ssh trust for wls
- ☒ Java is installed
- ☒ FMW is installed
- ☒ OSB is installed
- ☒ RCU is executed
- ☒ OSB is created
- ☒ OSB domain has been started and tested.
- ☒ Installation media for databases and middleware are deleted.
- ☒ Snapshot03 is created.

## 6.6 Conclusion

The necessary software distributions for the OSB, namely Java, FMW and OSB are installed on the CentOS VM. This was done with a dedicated installation user: *wls*. All steps to create a domain where executed, including RCU and domain configuration and creation with a Python script. The OSB domain

was started and testes. Access to the web based administration tools was established using an SSH tunnel.

## 7 Documentation

The documentation is an integral part of the live-scripting approach. The file `live-scripting_osb.org` is the documentation. It contains textual reports, listings, diagrams and attachments. This documentation will be shared in three different ways. First, the project, containing the org-mode-file, script, and configuration files will be uploaded into a separate GitHub project. Second, an HTML file will be generated from the orgmode file, containing all diagrams, listings and attachments, as well as style sheets. This constitutes a static web site that can be view directly with the browser from the file system. It will also be published to a web server. Although, GitHub is able to render orgmode files as HTML, this static web site uses style sheets based on the read-the-org project, which provides better overview and navigation. As a third option, a pdf file will be created from the orgmode file, using the Emacs export engine. It contains all links, listings and diagrams but does not include any attachments. It can be easily shared via eMail or be printed out and complements the documentation.

### 7.1 Project on GitHub

GitHub offers a command line interface which can be used to upload and commit the files of the project.

```

1  # Listing: Initializing git project and uploading it to GitHub
2
3  ## installing github cli
4  brew install gh
5  gh --help
6
7  ## initilize git project
8  cd ~/LocalProjects/live-scripting_osb
9
10 git init
11
12 ## create .gitignore file
13 cat << 'EOF' > .gitignore
14 .DS_*
15 *.bkp
16 EOF
17 cat .gitignore
18
19
20 ## adding file to git
21 ls -la
22 git add live-scripting_osb.org
23 git add live-scripting_osb.pdf
24 git add live-scripting_osb.html
25 git add data
26 git add docs
27 git add live-scripting_fmw
28 git add aw-org-html-themes
29
30 git status
31 git commit -m "initial project commit"
32
33
34 ## Creating a key pair for GitHub

```

```

35 ssh-keygen -t ed25519 -C "your_email@example.com"
36 /Users/aupeksha/.ssh/id_github
37
38
39 ls -la /Users/aupeksha/.ssh/
40
41
42 ## Uploading to GitHub
43
44 gh auth login
45
46 SSH
47 github
48 GitHub CLI
49 web browser
50
51
52 ## Ok
53
54 #
55 gh repo list
56 gh repo create
57 Push an existing
58 .
59 live-scripting_osb
60 Project to demonstrate the insllation of OSB with the live-scripting approach.
61 Public
62 Y
63 origin
64 Y
65 yes
66
67
68 ## Push project to github
69 git branch -M main
70 git remote add origin git@github.com:andreaswittmann/live-scripting_osb.git
71
72 git push -u origin main
73
74 git remote add origin https://github.com/andreaswittmann/live-scripting_osb.git
75 git push -u origin main
76
77 gh auth logout

```

### 7.1.1 Problem: Error during git push

```

1 # Listing: Error Message
2 bash-3.2$ git remote add origin git@github.com:andreaswittmann/live-scripting_osb.git
3 error: remote origin already exists.
4 bash-3.2$ git push -u origin main
5 git@github.com: Permission denied (publickey).
6 fatal: Could not read from remote repository.
7
8 Please make sure you have the correct access rights
9 and the repository exists.

```

## 8 Appendix

[2023-10-12 Thu 12:06] This appendix contains additional material that was used during setup, but is not relevant for the final solution. This includes further analysis, troubleshooting, and failed attempts. It only serves reference purposes.

### 8.1 Problem: Oracle Express Database needs RHEL 8.

I cannot install the database. Error in preinstallation:

```
1 # Listing: Error Message while trying to install Oracle XE Database
2 [root@localhost db]# yum -y localinstall --nobest oracle-database-preinstall-21c-1.0-1.el8.x86_64.rpm
3 Last metadata expiration check: 0:54:56 ago on Wed 11 Oct 2023 01:08:26 AM CEST.
4 Error:
5   Problem: conflicting requests
6     - nothing provides compat-openssl10 needed by oracle-database-preinstall-21c-1.0-1.el8.x86_64 from
   ↪ @commandline
```

**Attempt:** Installation of *compat-openssl11*. The installation works but it still breaks the pre-install.  
**Conclusion:** Oracle XE is not compatible with RHEL9. I will install CentOS 8