ORACLE

Implementing Oracle Key Vault on the Oracle Compute Cloud@Customer

How to install and deliver Oracle Key Vault on the Oracle Compute Cloud@Customer.

May, 2024, Version 2

Public

ORACLE

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

ORACLE

Table of contents

INTRODUCTION

Oracle Compute Cloud@Customer – Basics, Benefits & Background

Oracle Compute Cloud@Customer is fully managed, rack-scale infrastructure that lets organizations consume common OCI services anywhere. Remotely managed by Oracle, it lets customers gain cloud automation and economics benefits, while meeting data residency requirements by controlling their data's location.

Oracle Key Vault – Introduction to Oracle Key Vault & Key Management+

Oracle Key Vault enables you to deploy encryption and other security solutions by centrally managing Transparent Data Encryption (TDE) database encryption keys, Oracle Wallets, Java Keystores, credential files, and other secrets. Key Vault supports a scalable, fault-tolerant cluster deployment architecture to deliver continuous availability and geographic locality.

Installing Oracle Key Vault on Oracle Compute Cloud@Customer

Downloading the OKV image From Marketplace

Oracle Key Vault image for Compute Cloud@Customer is available on Oracle Cloud Infrastructure Marketplace.

1. Login to your OCI account and go to OCI Marketplace
2. On filters option, select Compute Cloud@Customer or Roving Edge compatible images
3. Select Oracle Key Vault image for Compute Cloud@Customer.

https://console.us-ashburn-1.oraclecloud.com/marketplace/application/164834538/overview?region=us-phoenix-1

Figure 1

Import the OKV Image to the Oracle Compute Cloud@Customer

Place the custom image in a location accessible via http. This is often a utility VM somewhere on the same network as your OC3, or even a utility VM on your OC3. Here, we will use the bastion host for our OC3. Copy the .oci file into a directory and make it accessible via http.

```
[root@myC3bast okv]# cd /export/home/okv
[root@myC3bast okv]# python -m SimpleHTTPServer 8088
Serving HTTP on 0.0.0.0 port 8088 ...
```

Meanwhile, on the OC3, begin the Custom Image import process through the GUI by navigating to the Custom Images section:

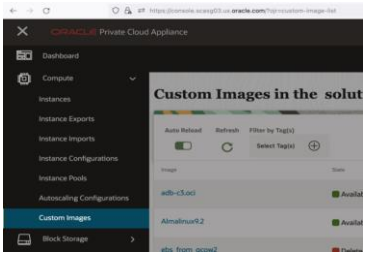

Figure 2

This will take you to the listing of custom images in the selected compartment. If necessary, change to the appropriate compartment using the drop down in the upper center of the page.

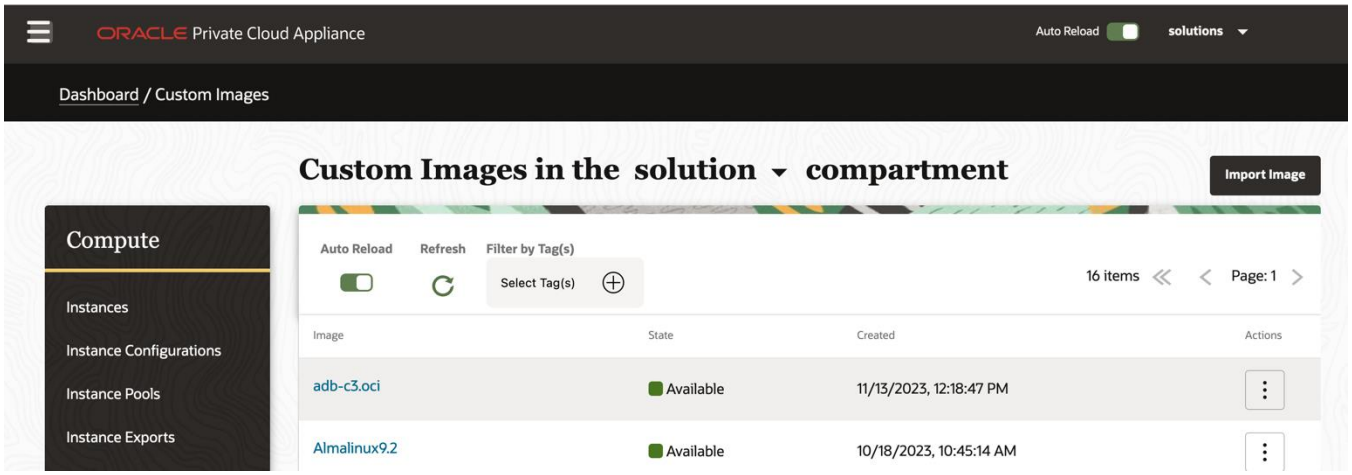

Figure 3

Click on the 'Import Image' button in the upper right, complete the dialog box as presented, and select 'Import Image' in the lower right:

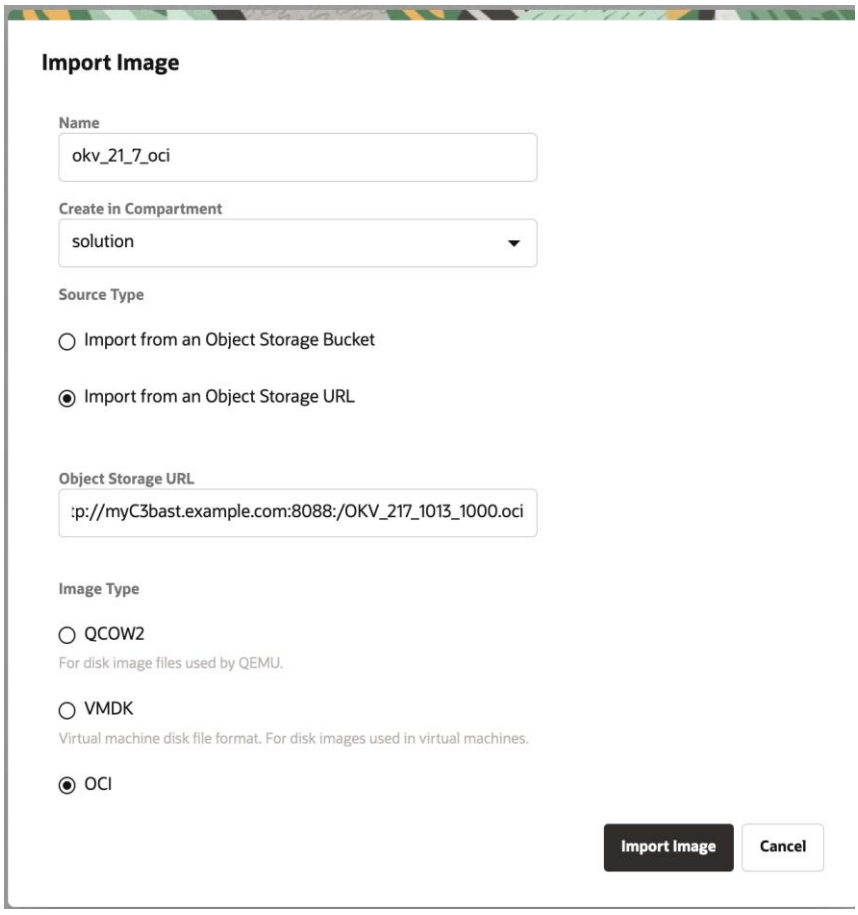

Figure 4

Be sure to select the correct file type (OCI) and 'Paravirtualized Mode'.

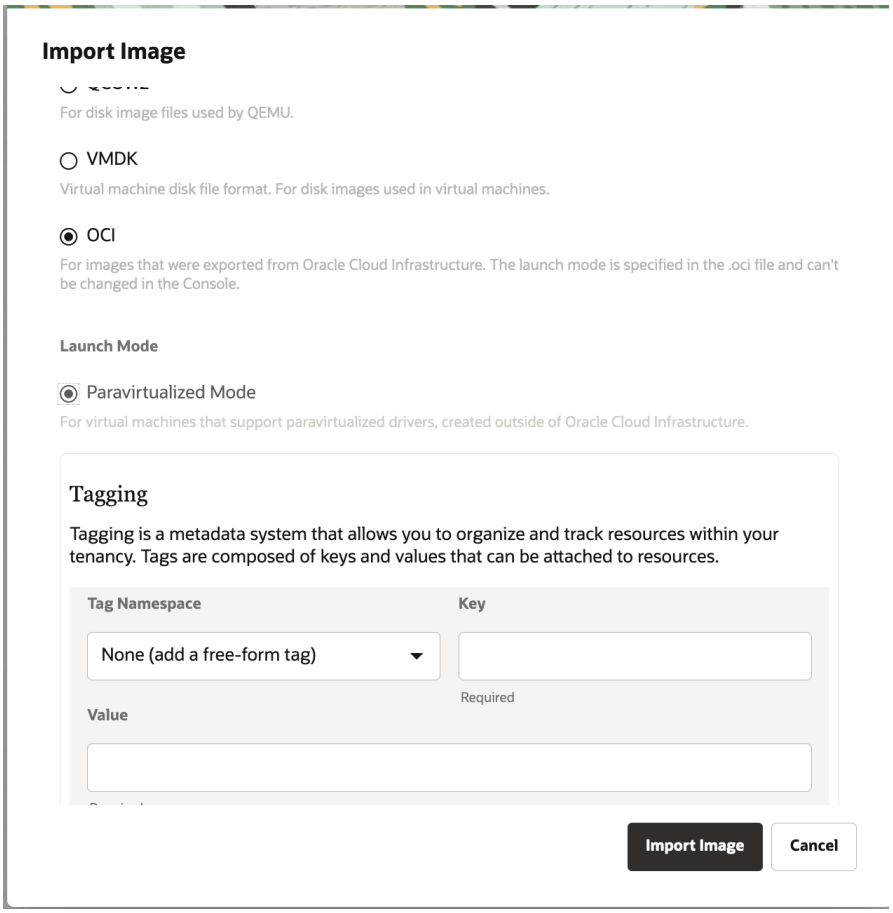

Figure 5

In this case, we exported the image using python web server on port 8088, so do not forget to specify port 8088 in the URL.

Depending on the size of the image being imported and the available network bandwidth, the import may take some time. An hour or more may be expected for images in excess of 100GB.

Once the image import is complete, the status will change from 'Importing" to 'Available' and you may create your instance from the image.

Figure 6

Creating your OKV Instance

When the custom image becomes available, use the 'Create Instance from Image' option in the drop down on the right end of the line (three vertical dots in the box).

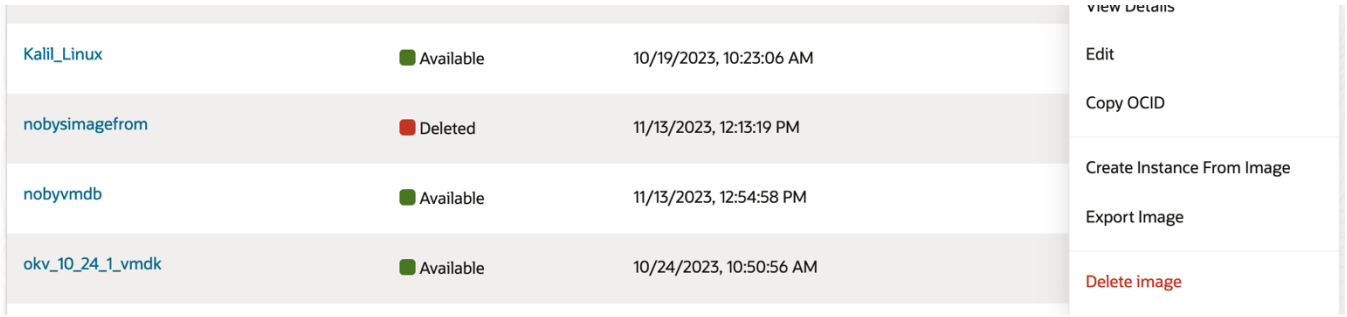

Figure 7

Complete the dialog box giving your image a name, choose a compartment, appropriate shape information (we chose 4 OCPU, 64GB of memory, 100GB boot volume, one public network interface but your needs may differ) and your ssh key, then select 'Create Instance' in the lower right. See Figures 7, 8, 9 as examples.

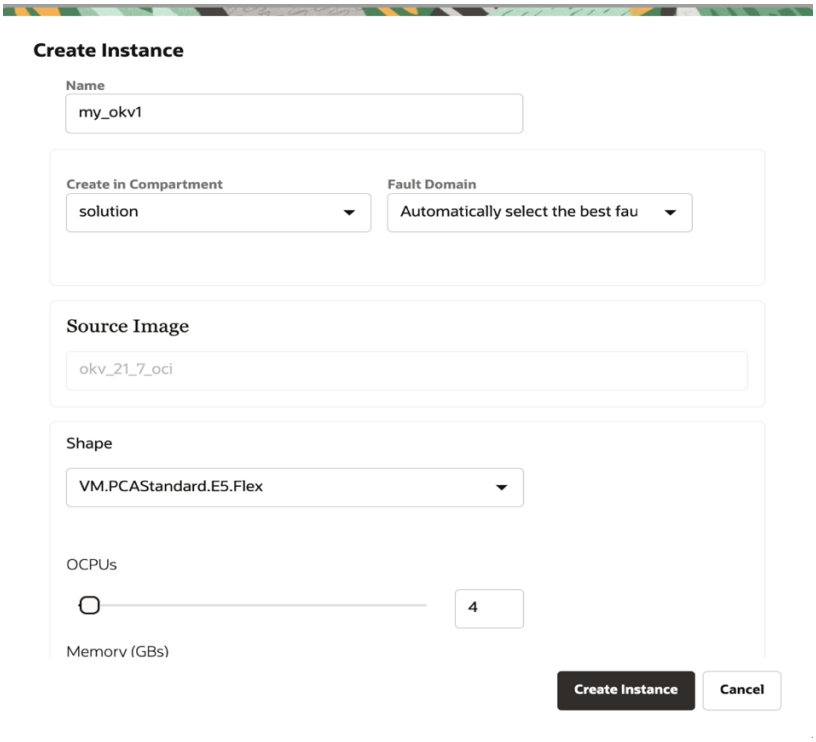

Figure 8

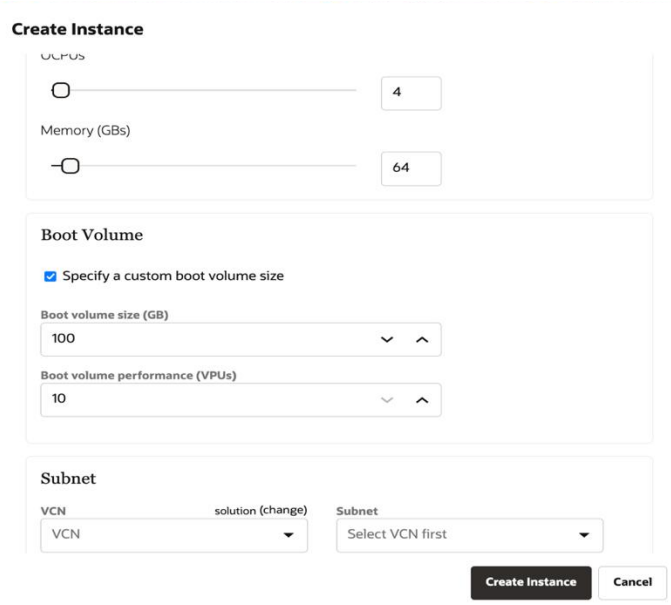

Figure 9

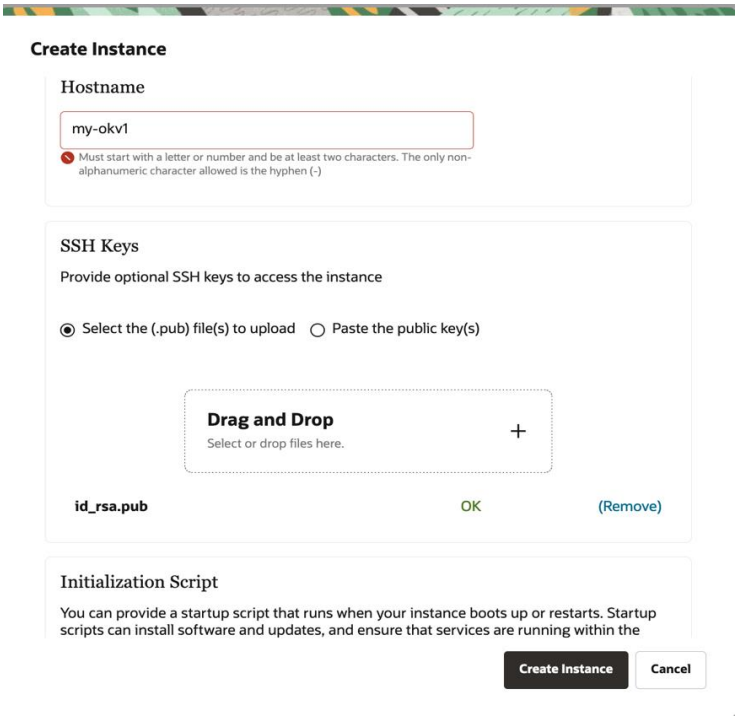

Figure 10

Once you select ‘Create Instance’, the system will create an instance using the custom image you have provided. Wait for this instance to boot. Once the instance is booted, log into the console and verify it is running. Once complete, repeat the process above to create an additional OKV node which will be used to create a high availability cluster.

ORACLE

Configuring your OKV Instances

Once the system has booted up, you must go through the post install steps to perform the initial configuration.

You will need:

The IP address or fqdn of a linux or Mac system to perform the configuration from. This can be any linux server on the same network as your OKV server(s) or a laptop. We will use c3bastion in this example.

Take note of the external and internal IP addresses assigned to the nodes you wish to add to your cluster. We will need both sets of addresses.

The IP address or fqdn of the OKV server(s) to be configured. We will use 10.122.56.38/172.20.0.33 and 10.122.56.29/172.20.0.21 in this example.

Server Initial Passwords

First, log into the the server to set the requisite temporary passwords.

my_laptop ~ $ ssh opc@10.122.56.29

Warning: Permanently added '10.122.56.29' (ED25519) to the list of known hosts.

Oracle Key Vault 21.7.0.0.0

DO NOT CHANGE ANY CONFIGURATIONS IN Oracle Key Vault Server APPLIANCE WITHOUT GUIDANCE

FROM ORACLE SUPPORT. ANY CHANGES SHOULD BE TRACEABLE TO APPROPRIATE SR REFERENCE.

****

****

Hello!

You are logged in as the 'opc' user.

'opc' is a temporary user used to set the root and support user passwords.

Once the passwords are set successfully, the 'opc' user will be deleted and

login to the Oracle Key Vault(OKV) instance using SSH will be turned off.

You can re-enable login to the OKV instance using SSH from the OKV management

console and login as the 'support' user.

Run the command below to set the root and support user passwords.

$ set_password

Next, login to the OKV management console to complete the post-install tasks.

****

****

[opc@okv0013978e4ef7 ~]$ set_password

Setting root password

Set root password:

Confirm:

Changing password for user root.

passwd: all authentication tokens updated successfully.

Successfully set the root password..

Do you wish to set the support user password at this time.

Enter 'y' or 'yes' to proceed: y

Set support user password:

Confirm:

Changing password for user support.

passwd: all authentication tokens updated successfully.

Successfully set the support user password..

Deleted 'opc' user..

You can re-enable login to the Oracle Key Vault instance using

SSH from the Oracle Key Vault management console.

Login as the 'support' user using the same ssh key as 'opc' user.

Connection to 10.122.56.29 closed.

my_laptop ~ $^D

Once this stage is complete, log in via the GUI using the root password provided in the previous step and complete the post install configuration; perform the user set up, system administrator setup, Time (NTP) and Domain Name System (DNS) setups. Then save this information.

Figure 11

Setup for Key, System, and Audit users

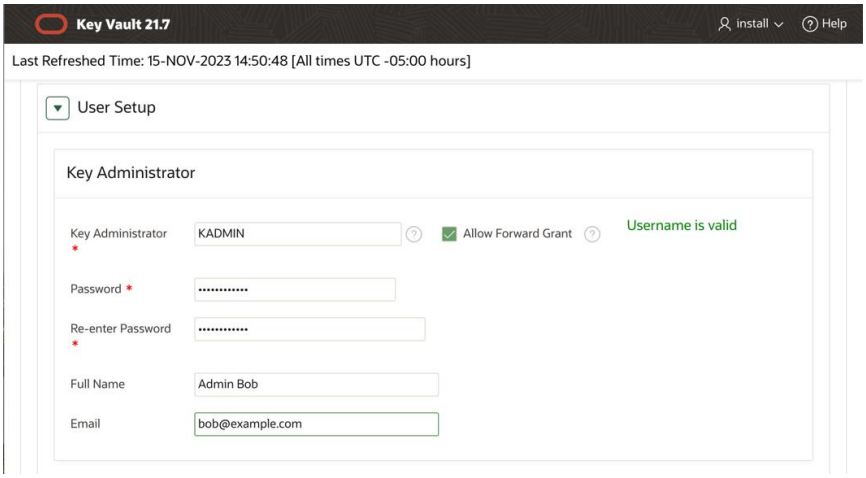

Figure 12

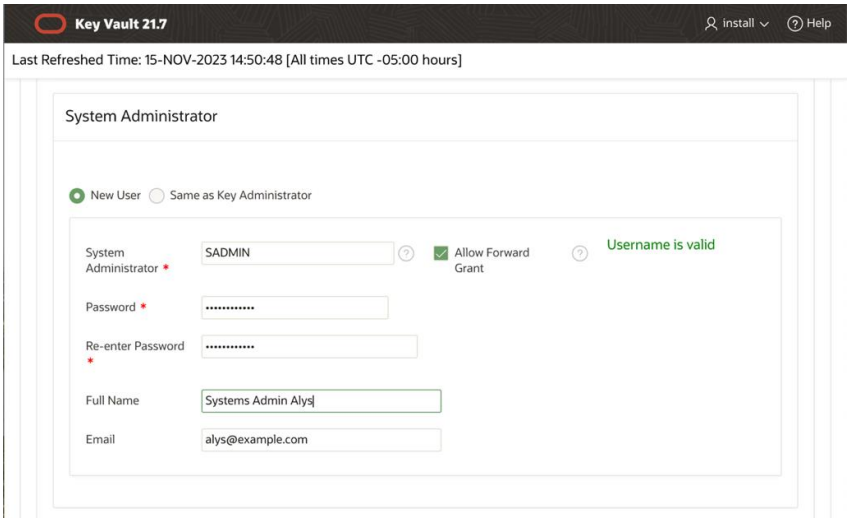

Figure 13

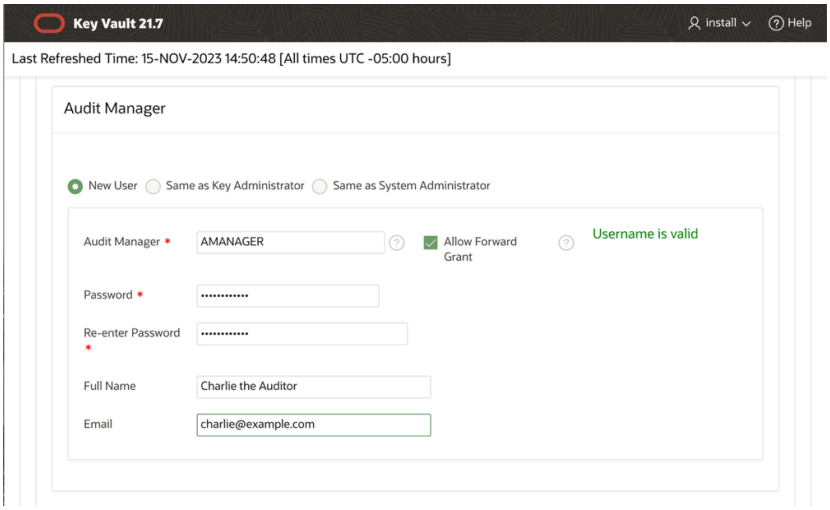

Figure 14

Specify the Recovery passphrase. Note: Do not lose the Recovery Passphraser. Store this in a secure locaton.

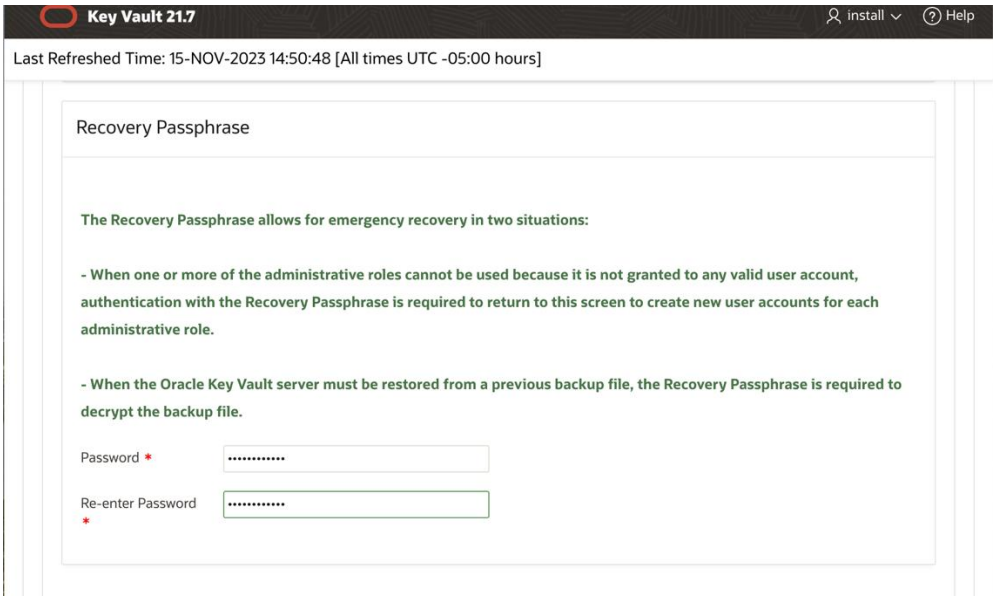

Figure 15

Lastly, set up the NTP and DNS servers.

NOTE: On the C3, These should always be set to the default for the C3 chassis. Please only specify a single NTP server, and a single DNS server. The IP address to be used for both is the same, 169.254.169.254

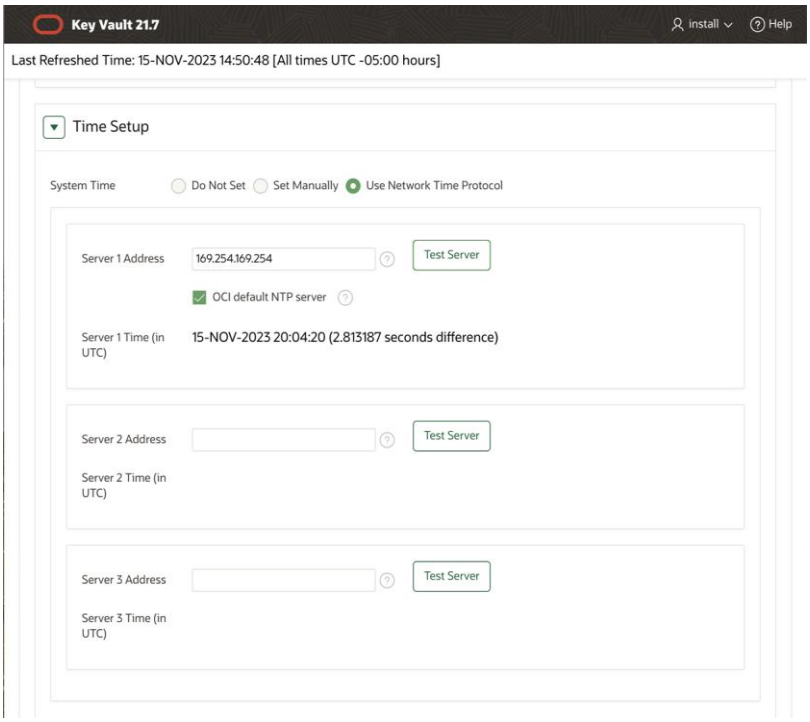

Figure 16

Here you can see we have used the single NTP server, 169.254.169.254 and tested it to confirm operation.

Use the same IP address for the lone DNS server. DNS and NTP are both provided redundantly inside the C3 and thus should only have the single entry.

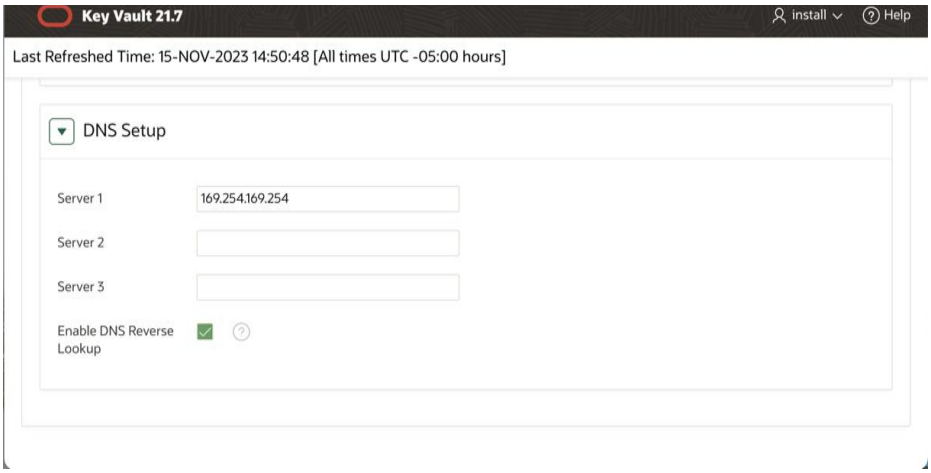

Figure 17

Once these steps are completed, select the 'SAVE' button at the upper right of the page and repeat for any additional servers you will be adding to a cluster.

You must then log out, log back in as the Systems Admin user and start the REST services.

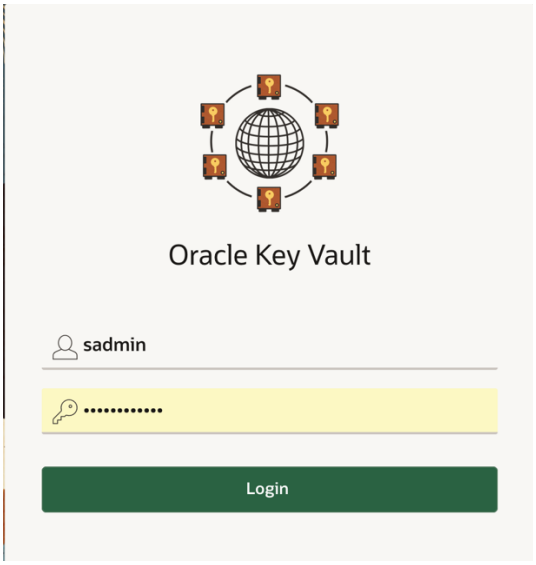

Figure 18

Select 'RESTful Services' from the System tab, check the 'Enable' box and Save the setting.

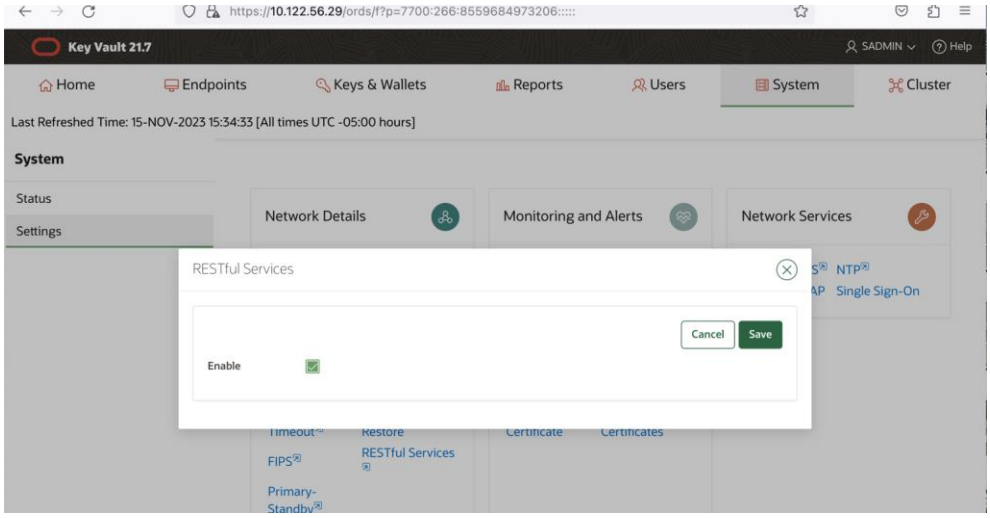

Figure 19

Once complete, you may complete the configuration using the RESTful service interface.

Server Configuration

On the server you will be using to perform the configuration download the RESTful OKV service package as follows:

NOTE: You must do this once per server you are configuring as the download includes certificates for the OKV server.

[root@c3bastion tmp]# mkdir /tmp/okv

root@c3bastion tmp]# cd /tmp/okv

root#c3bastion okv]# curl -Ok --tlsv1.2 **https://10.122.56.16:5695/okvrestclipackage.zip**

```
                 % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                Dload  Upload   Total   Spent    Left  Speed
100  2740  100  2740    0     0     78      0  0:00:35  0:00:34  0:00:01   741
```

[root@c3bastion okv]# unzip okvrestclipackage.zip

Archive: okvrestclipackage.zip

  creating: lib/

  creating: bin/

 inflating: bin/okv

 inflating: bin/okv.bat

  creating: conf/

 inflating: conf/okvrestcli.ini

 inflating: conf/okvrestcli_logging.properties

 inflating: lib/okvrestcli.jar

[root@scasg03bast okv]# cd bin

Edit bin/okv:

Remove the "#" from the beginning of the third line, save and exit:

```
************************************************************
#!/bin/bash
export OKV_RESTCLI_DIR=$(dirname "${0}")/..
#export OKV_RESTCLI_CONFIG=$OKV_RESTCLI_DIR/conf/okvrestcli.ini
if [ -z "$JAVA_HOME" ]
then
  echo "JAVA_HOME environment variable is not set."
  exit 1
fi

if [ -z "$OKV_RESTCLI_CONFIG" ]
then
  echo "OKV_RESTCLI_CONFIG environment variable is not set."
  exit 1
fi

export OKV_RESTCLI_JAR=$OKV_RESTCLI_DIR/lib/okvrestcli.jar
$JAVA_HOME/bin/java -jar $OKV_RESTCLI_JAR "$@"

************************************************************
```

Edit conf/okvrestcli.ini:

Remove the "#" sign from the beginning of lines 3 .. 6, add in the private IP address of the first server, add in the username and delete the line that starts with "password"

```
**********************************************************

#Provide absolute path for log_property, okv_client_config properties
[Default]
#log_property=./conf/okvrestcli_logging.properties
#server=172.20.0.21
#okv_client_config=./conf/okvclient.ora
#user=sadmin
#password=[user password]

**********************************************************

#Provide absolute path for log_property, okv_client_config properties
[Default]
log_property=./conf/okvrestcli_logging.properties
server=<IP_address of OKV01>
okv_client_config=./conf/okvclient.ora
user=<name of an OKV-administrator with the SYSADMIN privilege>
client_wallet = .

**********************************************************
```

JAVA_HOME needs to be set for OKV REST command to work:

```
$ java -version
openjdk version "1.8.0_372"
OpenJDK Runtime Environment (build 1.8.0_372-b07)
OpenJDK 64-Bit Server VM (build 25.372-b07, mixed mode)
```

OpenJDK is not supported; the Linux program “namei” follows symbolic links and helps to confirm where OpenJDK is installed:

```
$ which java
/usr/bin/java

$ namei /usr/bin/java | grep " l "
 l java -> /etc/alternatives/java
   l java -> /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.372.b07-1.el7_9.x86_64/jre/bin/java
```

Oracle Java can be downloaded with script-friendly commands (Oracle Java 17 is the current long-term release):

```
[root@c3bastion okv]#  wget https://download.oracle.com/java/17/latest/jdk-17_linux-x64_bin.rpm
<content redacted for brevity>
Saving to: ‘jdk-17_linux-x64_bin.rpm’

100%[=======================================================================================
=======================>] 182,170,753 22.3MB/s   in 4.9s

2023-11-14 10:21:48 (35.5 MB/s) - ‘jdk-17_linux-x64_bin.rpm’ saved [182170753/182170753]

[root@c3bastion okv]# yum localinstall ./jdk-17_linux-x64_bin.rpm
Loaded plugins: ulninfo
Examining ./jdk-17_linux-x64_bin.rpm: 2000:jdk-17-17.0.9-11.x86_64
Marking ./jdk-17_linux-x64_bin.rpm to be installed
<content redacted for brevity>
Installed:
  jdk-17.x86_64 2000:17.0.9-11
Complete!
[root@c3bastion okv]# namei /usr/bin/java | grep " l "
 l java -> /etc/alternatives/java
   l java -> /usr/lib/jvm/jdk-17-oracle-x64/bin/java
[root@c3bastion okv]#
```

Eventually, confirm that “alternatives” has been updated by the java installation process:

```
$ namei /usr/bin/java | grep " l "
 l java -> /etc/alternatives/java
   l java -> /usr/lib/jvm/jdk-17-oracle-x64/bin/java
```

That output gives us JAVA_HOME:

```
[root@c3bastion okv]# export JAVA_HOME=/usr/lib/jvm/jdk-17-oracle-x64
```

In order to simplify the deployment process, store the password of the OKV administrator with the SYSADMIN privilege in a wallet:

```
$ okv admin client-wallet add --client-wallet . --wallet-user <name of an OKV-
administrator with the SYSADMIN privilege>


Password: <type password of an OKV-administrator with the SYSADMIN privilege>
{
  "result" : "Success"
}
```

Cluster Configuration

Once the initial OKV software is installed and configured on the server, convert the stand alone OKV into a candidate node:

```
[root@c3bastion okv]# bin/okv cluster node create --cluster-name OCEAN11 --cluster-subgroup
WEST_COAST --node-name OKV04
{
  "result" : "Success",
  "value" : {
    "requestId" : "26032"
  }
}
[root@c3bastion okv]#
[root@scasg03bast okv]# bin/okv cluster node status --pairing-request-id 26032
{
  "result" : "Success",
  "value" : {
    "status" : "IN-PROGRESS"
  }
}
```

After several minutes, this will change to SUCCEEDED, much like this:

```
[root@c3bastion okv]# bin/okv cluster node status --pairing-request-id 26032
{
  "result" : "Success",
  "value" : {
    "status" : "SUCCEEDED"
  }
}
[root@c3bastion okv]#
```

Once this step is completed, the node should show up in the cluster management and monitoring tab.

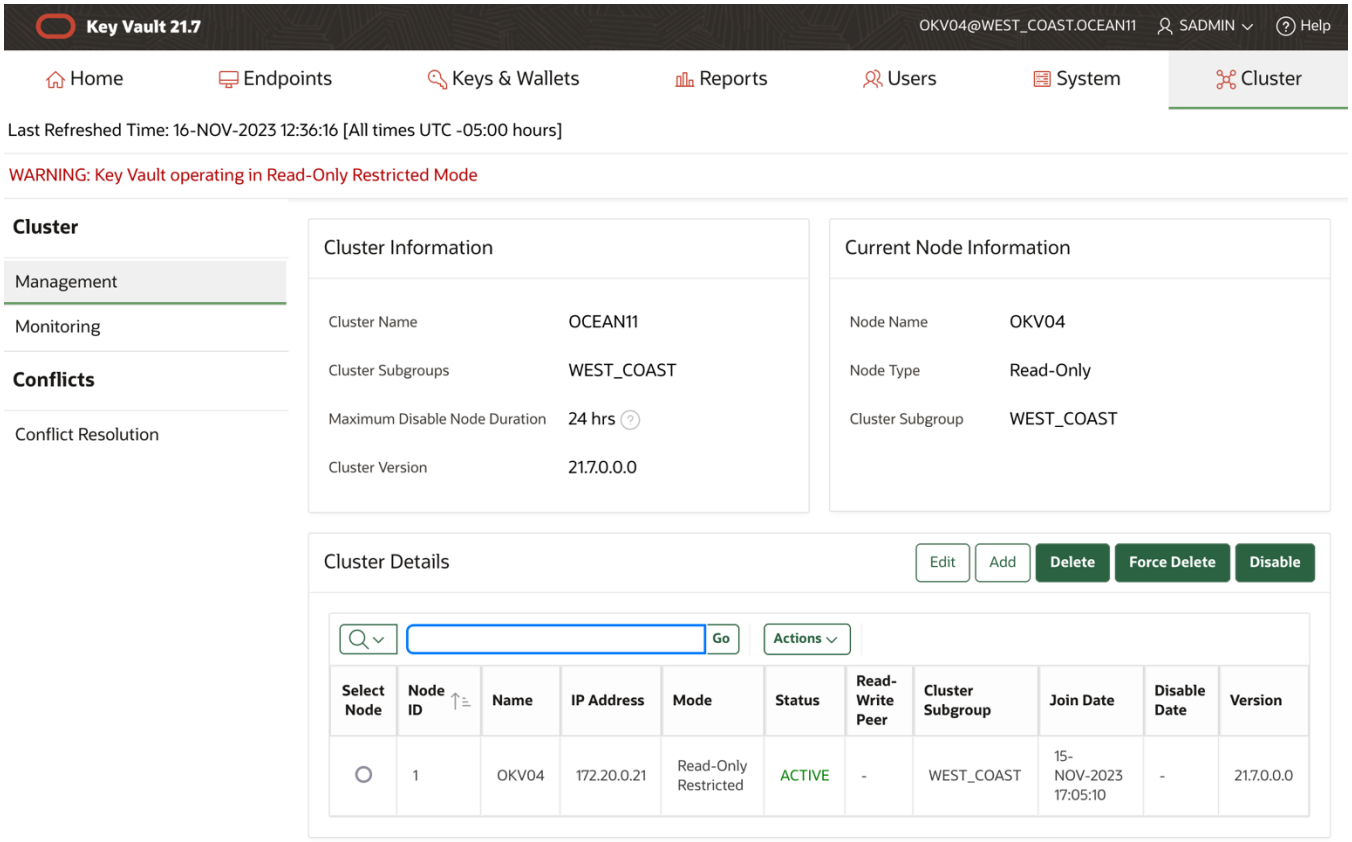

Figure 20

The next command, which adds the 2nd stand-alone OKV server to first to build a read-write pair, asks for a unique nodeID, Before adding a node, confirm which nodeID has already been taken:

$ okv cluster info get | jq -r '.value.nodes[].nodeID'

$ okv cluster node add --candidate-node-ip-address 172.20.0.33 --candidate-node-user sadmin --cluster-subgroup WEST_COAST --mode READ-WRITE --node-id 2 --node-name OKV06
Recovery Passphrase: (of first OKV node)
Candidate Node Password: <password of an OKV-administrator with the SYSADMIN privilege>


```
{
 "result" : "Success",
 "value" : {
  "requestId" : "3060"
 }
}
```


Monitor the process on the first node:

[root@c3bastion okv]$ bin/okv cluster node status --pairing-steps TRUE --node-name OKV04


```
{
 "result" : "Success",
 "value" : {
  "stages" : [ {
   "step1" : "Open transport channel with the candidate node",
   "status" : "COMPLETED"
  }, {
   "step2" : "Verify the candidate node details",
   "status" : "COMPLETED"
  }, {
   "step4" : "Generate the controller node details",
   "status" : "COMPLETED"
  }, {
   "step5" : "Generate backup of the controller node for cloning",
   "status" : "COMPLETED"
  }, {
   "step6" : "Send clone bundle to the candidate node",
   "status" : ""
  }, {
   "step7" : "Enable data replication (downstream mining configuration) to the candidate node",
   "status" : ""
  }, {
```

```
    "step8" : "Enable data replication to other cluster nodes",
    "status" : ""
   }, {
    "step9" : "The candidate node successfully joins the cluster",
    "status" : ""
   } ]
 }
}
```


And check on the second node:

[root@c3bastion okv]# bin/okv cluster node status --pairing-steps TRUE --candidate-node-ip-address 172.20.0.21 --candidate-node-user sadmin

Candidate Node Password:


```
{
 "result" : "Success",
 "value" : {
  "stages" : [ {
   "step1" : "Send node details to the controller node",
   "status" : "COMPLETED"
  }, {
   "step2" : "Receive clone bundle from the controller node",
   "status" : "COMPLETED"
  }, {
   "step3" : "Restore backup on the candidate node",
   "status" : "COMPLETED"
  }, {
   "step4" : "Update credentials of the candidate node",
   "status" : "COMPLETED"
  }, {
   "step5" : "Tune the database on the candidate node",
   "status" : "COMPLETED"
  }, {
   "step6" : "Setup network configuration on the candidate node",
   "status" : "COMPLETED"
  }, {
```

```
 "step7" : "Enable data replication (downstream mining configuration) on the candidate node",
     "status" : "COMPLETED"
   }, {
     "step8" : "Enable data replication on the candidate node",
     "status" : "COMPLETED"
   } ]
 }
}
[root@c3bastion okv]#
```


When complete, the command will show ‘No pairing status”:


```
[root@c3bastion okv]# bin/okv cluster node status --pairing-steps TRUE --node-name OKV10
{
  "result" : "Failure",
  "message" : "No pairing status"
}
[root@c3bastion okv]#
```


The first 2-node OKV read-write pair is ready to be used:


```
[root@c3util okv]$ bin/okv cluster info ge
{
  "result" : "Success",
  "value" : {
    "clusterName" : "OCEAN11",
    "clusterSubgroups" : [ "WEST_COAST" ],
    "clusterVersion" : "21.7.0.0.0",
    "maximumDisableNodeDuration" : "24 hrs",
    "nodes" : [ {
      "nodeName" : "OKV04",
      "nodeID" : "1",
      "ipAddress" : "172.20.0.33",
      "mode" : "Read-Write",
      "status" : "ACTIVE",
```

```
  "readWritePeer" : "OKV05",
    "clusterSubgroup" : "WEST_COAST",
    "joinDate" : "2023-11-16 20:53:25",
    "disableDate" : "",
    "version" : "21.7.0.0.0"
  }, {
    "nodeName" : "OKV05",
    "nodeID" : "2",
    "ipAddress" : "172.20.0.21",
    "mode" : "Read-Write",
    "status" : "ACTIVE",
    "readWritePeer" : "OKV04",
    "clusterSubgroup" : "WEST_COAST",
    "joinDate" : "2023-11-16 21:02:24",
    "disableDate" : "",
    "version" : "21.7.0.0.0"
  } ]
 }
}
[root@c3bastion okv]#
```


Congratulations, you now have an operation Oracle Key Vault system. You may begin to use your key vault as outlined in the **Oracle Key Vault documentation**.

NOTE: For additional use cases with Oracle Key Vault (OKV), please refer to the OKV official documentation.

Oracle Key Vault 21.8 Official Documentation
Oracle Key Vault Use Cases