AWS Whitepaper

Running Oracle Hypervisors on Amazon EC2 Bare Metal

Running Oracle Hypervisors on Amazon EC2 Bare Metal: AWS Whitepaper

Table of Contents

Running Oracle Hypervisors on Amazon EC2 Bare Metal

Publication date: **August 5, 2021** (*Document history*)

Amazon Web Services (AWS) provides a wide range of operating environments to run your Oracle workloads, each offering different benefits depending on the use case and operating model. The operating environment you select can have a material impact on the cost of running the platform, especially the licensing of proprietary Oracle software. AWS provides Amazon EC2 bare metal instances with the capability for customers to bring their own hypervisor and opens the door for more flexible licensing arrangements. This whitepaper introduces the value proposition of running an Oracle Hypervisor on Amazon EC2 bare metal instances and shows how you can combine the agility and elasticity benefits of AWS with the cost efficiency of leveraging additional licensing optimizations.

All Oracle licensing policies and costs in this whitepaper are for informational purposes only and are based on the information available at the time of publication. AWS recommends that customers consult their own Oracle license agreement for more specific information.

Introduction

Customers running Oracle workloads on shared tenancy Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Relational Database Service (Amazon RDS) instances in AWS are subject to Oracle's Cloud Licensing policy. This differs considerably from the traditional on-premise Oracle license policy and can lead to a suboptimal cost model in certain circumstances.

Some customers choose to run their Oracle workloads on Amazon EC2 Dedicated Hosts so that they can adhere to the traditional core-based licensing model instead. This can yield a substantial licensing saving while still leveraging the power of the AWS Cloud platform.

AWS Deployment Option	SE License Included	Oracle Cloud Policy	Core-Based License	Unlimited License Agreement
RDS	✓	✓	✗	✓
EC2 Compute	✗	✓	✗	✓
VMware Cloud on AWS	✗	✓	✓	✓
Dedicated Hosts on EC2	✗	✓	✓	✓
EC2 Bare Metal Instances	✗	✓	✓	✓

Oracle licensing options for AWS Deployment patterns

To further optimize licensing costs, customers have the option to bring their own Hypervisor and run them on EC2 bare metal instances. In the case of third-party hypervisors such as VMWare ESx and Hyper-V this adds the capability to over-subscribe central processing unit (CPU) at the guest level in order to achieve higher consolidation density and make more efficient use of shared compute resources. This has been a highly effective strategy for those environments with relatively high proprietary licensing costs combined with dynamic workload levels.

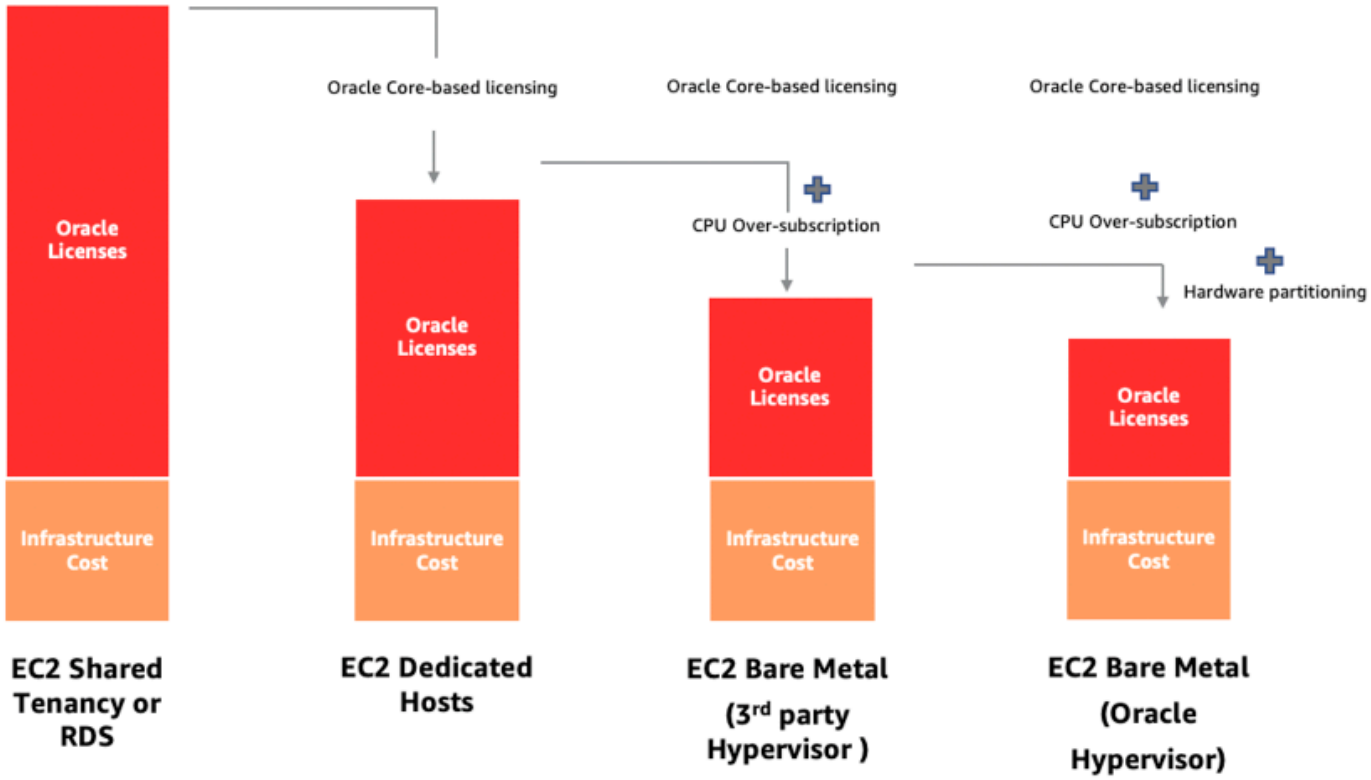
However, the use of a non-Oracle hypervisor in this way does not currently satisfy Oracle's hard partitioning requirements. Hardware partitioning is a mechanism to limit the number of CPU resources available to a server with the aim of reducing its licensing requirement. This technique has been useful in environments that separate a single large server into distinct smaller servers where each separated sub-system acts as a physically independent and self-contained server. Customers have been taking advantage of this technique to right-size their Oracle licenses and reduce their operating expenses.

In order to adhere to Oracle's hard partitioning requirements customers must run one of the approved technologies specified in the above partitioning policy. There are two options for customers running their Oracle workloads on AWS.

- Oracle VM Server
- Oracle Linux KVM

By running one of these two Oracle hypervisors on EC2 bare metal instances on AWS, customers can streamline their platform spend by combining various cost optimization mechanisms.

The following diagram shows relative cost savings achievable using these cost optimization techniques. This example is provided to show a conceptual idea rather than estimate the scale of the potential savings. Customers should consult their Oracle license agreement for their own business use case.



License optimization mechanisms

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Value Proposition

In some customer environments the proprietary software license cost can form the majority of the overall IT spend, and businesses have competitive and consumer pressure to reduce their operational expenditure.

Having the ability to optimize software licensing by adhering to hard partitioning policies while at the same time taking advantage of the operating model efficiencies of the AWS Cloud enables companies to streamline their IT operations and free up valuable resources.

AWS can help customers move away from their restrictive and punitive commercial arrangements towards more flexible and managed offerings. However, we understand that every customer use case is different and some customers are looking for transient or incremental migration approaches to the cloud. AWS continues to evangelise the modernisation approach but the ability to run Oracle Hypervisors on EC2 bare metal instances gives customers another option to choose in their cloud adoption journey.

Customers should keep in mind that there are alternative consolidation techniques that can be used to optimize license efficiency, for example, Oracle's Multi-tenant Database feature. However, the Oracle hypervisor on EC2 bare metal pattern is still useful for Weblogic VM server farms and for other lift and shift migrations where customers choose to minimize application and operating model changes during migration.

Oracle License Portability to AWS

Subject to the terms and conditions of the specific license agreement, Oracle licenses may be portable to AWS.

Licenses that can be migrated can include:

- Processor Licenses (Server licenses based on CPUs)
- Enterprise License Agreements (ELA)
- Unlimited License Agreements (ULA)
- Perpetual Unlimited License Agreements (PULA)
- Business Process Outsourcing (BPO) licenses
- Oracle Partner Network (OPN) licenses
- Named User Plus Licenses (Server licenses based on CPUs and users)

Additional conditions or limitations (including possible costs) may be applicable for licenses that are ported to AWS. Check your specific license agreement for additional details and limitations.

Oracle licensing applies similarly to Oracle Database on Amazon RDS and on Amazon EC2 with the exception that hourly licensing is available only on Amazon RDS.

Hypervisor Choice

To comply with the hard partitioning requirements laid out in the license policy for partitioned environments customers have the choice between 2 different hypervisors to bring to AWS:

- Oracle VM Server (OVM Server), or
- Oracle Linux KVM (OL KVM)

Both of these hypervisors also have associated configuration requirements that must be implemented to comply with the policy.

As of 31 March 2021, Premier Support for OVM Server has ended. Oracle is offering Extended Support for OVM Server until March 31, 2024 for an additional fee. Extended Support will not include support for new hardware or new bug fixes. After 2024, OVM Server will be covered by Sustaining Support. For more information about support for Oracle Hypervisors, see Support Policies.

Customers should keep this in mind when selecting an Oracle hypervisor to host their workloads.

The following table shows a comparison of the two hypervisors.

Table - Oracle Hypervisor comparison

Attribute	Oracle VM Server	Oracle Linux KVM
Hypervisor	Xen	KVM
Support Level	Sustaining Support (or paid Extended Support)	Premier Support
New Deployments	No	Yes
Bug fixes	No	Yes
Management	Oracle VM Manager (OVMM)	Oracle Linux Virtualization Manager (+KVM ecosystem)

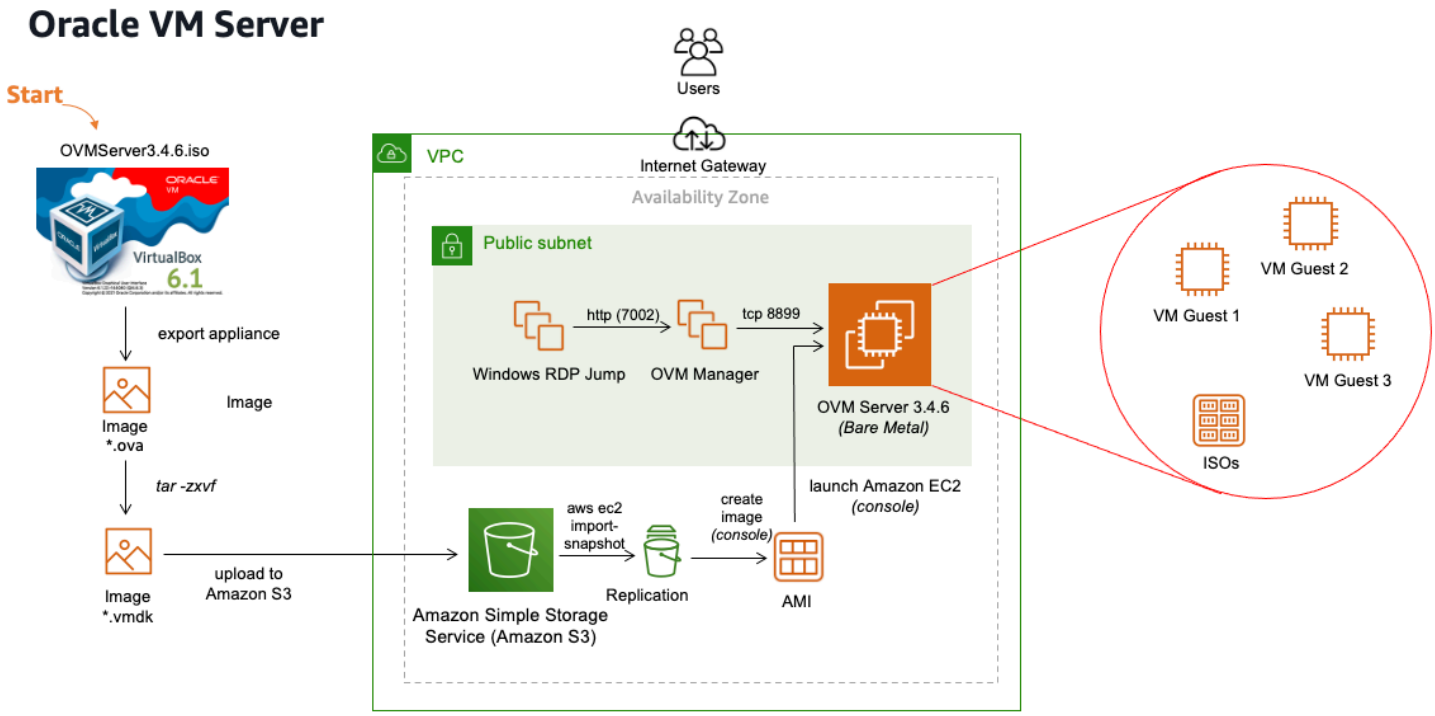
Topics

- Oracle VM Server
- Oracle Linux KVM

Oracle VM Server

Oracle VM Server is based on the Xen hypervisor and works in conjunction with OVM Manager (OVMM) to provide a hosting and management platform for running virtual machines. This software is free to download and distribute.

The high-level steps to deploy OVM Server onto Amazon EC2 bare metal instances is shown in the diagram below.



Oracle VM Server deployment on EC2 Bare Metal

To deploy an operating system or hypervisor to an EC2 bare metal instance you need an Amazon Machine Image (AMI). You can obtain an AMI for OVM Server in one of the following ways:

- Download an AMI from the AWS Marketplace (if one exists)
- Download an AMI from the Community AMI repository (if one exists)
- Create your own AMI

At the time of publication there are no publicly available AMIs in either the AWS Marketplace or Community repository for a recent version of OVM Server.

To create your own AMI for Oracle VM Server use the following procedure:

1. Download Oracle VM Server 3.4.6 from the official download site (in ISO format).
2. Download a recent version of Virtualbox onto your laptop or PC.
3. Create a VM Guest in Virtualbox using the downloaded ISO file in Step 1.
4. Verify that the Elastic Network Adaptor (ENA) and NVMe drivers are included in the OVM Server kernel.
5. Shutdown the OVM Server guest VM.
6. Export the OVM Server guest VM using Open Virtualisation Format v1 (`*.ova`).
7. Extract the `*.vmdk` file from the `*.ova` file from Step 6 above (using `tar` or `zip`).
8. Upload the `*.vmdk` file to an Amazon S3 bucket.
9. Import the `*.vmdk` file to an EC2 snapshot using the following command.

```
# aws ec2 import-snapshot --description "OVM Server 3.4.6 from VMDK" \
--disk-container file://VMDK.json

{ "Description": "OVMServer VMDK", "Format": "VMDK", "UserBucket": { "S3Bucket": "my-s3-bucket", "S3Key": "OVMServer-disk001.vmdk" } }
```

10. Using the AWS Management Console, create an AMI (image) from the snapshot imported in Step 9.

You will now have an Amazon Machine Image which contains OVM Server 3.4.6 that can be used to launch EC2 instances. This image will boot on traditional shared tenancy instances, dedicated hosts, and bare metal instances.

To launch this AMI onto an EC2 bare metal instance, choose a *.metal instance type in the AWS console or CLI (for example, `m5.metal`).

The most intuitive interface for administering OVM Server and its guest VMs is Oracle VM Manager (OLMM), a separate product distributed by Oracle. OVMM talks to OVM Server over HTTP and allows administrators to create and manage VM guests using their web browser across one or more OVM Server hypervisors. OVMM is available as a separately downloadable software image from Oracle.

Use the following procedure to install OVMM and use it to create a virtual machine guest on the OVM Server.

1. Download OVM Manager (OVMM) in ISO file format.
2. Launch a standard EC2 instance using a recent RHEL or Amazon Linux AMI.
3. Install OVMM onto the EC2 instance created in Step 2 above (mount ISO from Step 1.)
4. Open TCP Port 7002 incoming to the OVMM instance from your laptop or bastion server (RDP Jump host).
5. Open TCP Port 8899 outgoing from the OVM Server metal instance, and incoming to the OVMM instance.
6. Log in to the OVMM console (`https://<hostname>:7002/ovm/console`)
7. Choose **Discover Servers**, and type the **private IP** of the OVM Server metal instance, and type the **Agent password** you chose when installing OVM Server
8. Create a **Server Pool** in OVMM and add the discovered OVM Server to it. **Do not create a Cluster pool**.
9. Add an additional **gp2 EBS volume** to the OVM Server metal instance. **Do not partition this volume.**
10. In the OVMM Console using the previous EBS Volume, create a new **Repository**.

OVMM is now ready to create guest virtual machines that are hosted on the remote Oracle VM server, as shown in the previous architecture diagram.

Use the following procedure, to create a guest VM:

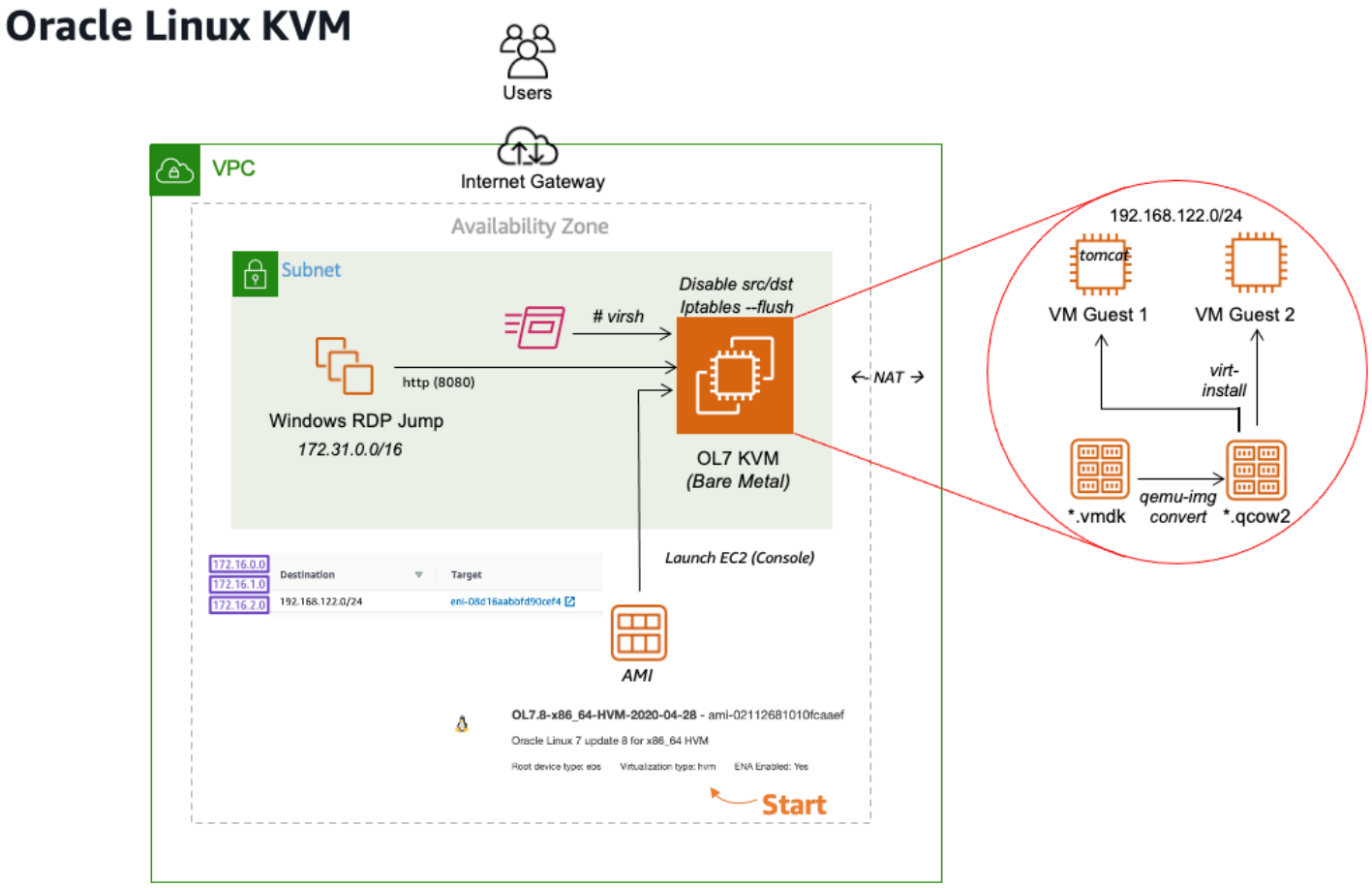
1. Create a VM guest in the OVMM Console using the previously created Repository (use the default network created by OVM Server).
2. In the OVMM Console, add a CDROM and virtual disk to the VM guest.
3. Download a publicly available ISO for an operating system image (for example, Centos7) onto a directory on the OVM Server instance.
4. Using the OVMM Console, import the ISO image into the Repository (using HTTP).
5. Edit the VM guest's CDROM device to point to the ISO image from Step 4.
6. Boot the VM Guest using the ISO, and install the operating system.
7. Verify that you can connect to the VM Guest from the metal instance hypervisor host.

To adhere to the licensing requirement for hard partitioning with OVM Server you must bind vCPUs to physical CPU threads or cores.

Oracle Linux KVM

Since Oracle has announced the deprecation of Oracle VM Server, customers are being encouraged to move to Oracle Linux KVM as a long-term strategic hypervisor to host their workloads. Linux Kernel-based Virtual Machine (KVM) is being adopted as an industry standard hypervisor for x86 platforms. The kernel component of KVM is included in mainline Linux, as of v2.6.20 and the userspace component of KVM is included in mainline QEMU, as of v1.3.

Because KVM is included in recent Linux versions, the steps to deploy Oracle Linux KVM onto EC2 Bare Metal are much simpler than those for OVM Server.



Oracle Linux KVM deployment process on EC2 Bare Metal

When deploying onto EC2 bare metal, the key advantages of running Oracle Linux KVM are as follows:

- There are public AMIs available for Oracle Enterprise Linux 7 with built-in KVM support, which mitigates the need to create your own using Virtualbox.
- There is no need to install a remote management component on a separate server to create guest virtual machines, because the virsh CLI can be installed locally on the OL KVM host.

These benefits simplify the hypervisor architecture and optimize for speed of deployment.

KVM is supported on Oracle Enterprise Linux (OEL) 7 and OEL 8 systems using either RHCK or any Unbreakable Enterprise Kernel (UEK) since UEK Release 4. You can find publicly accessible AMIs for recent versions of Oracle Linux 7 and 8 in both the AWS Marketplace and community AMI repository.

Use the following procedure, to deploy Oracle Linux KVM onto an EC2 Bare Metal instance:

1. Locate an AMI of a recent version of OEL 7 or OEL 8.
2. Launch an EC2 bare metal instance type (`*.metal`) using the AMI from Step 1.
3. Install additional packages to provide tools to deploy and administer VM guests:

```
# yum -y install virt-manager libvirt libvirt-python python-virtinst libvirt-client
virt-install virt-viewer qemu-kvm qemu-img
```

At this point KVM is installed and ready to host new virtual machine guests. There is a very mature ecosystem consisting of open source and proprietary tools that can be used to create and manage VM guests on KVM hosts. For an example of how broad the selection is in this ecosystem, see Management Tools.

The simplest tools to interface with KVM for the purposes of deploying guest VMs are the CLI tools `virt-install` and `virsh`. Virtual machines can be created and bootstrapped by downloading a VM template in qcow2 format from the many available public repositories. Alternatively you can download a VM image in `*.vmdk` format and convert it to `qcow2` format using `qemu-img`:

```
# qemu-img convert -O qcow2 /VMs/ConvertMe.vmdk /VMs/ConvertMe.qcow2
```

To create a guest VM on your KVM Bare Metal instance, run virt-install using the downloaded qcow2 file:

```
# virt-install --name Centos7 --memory 2048 \
```

```
--vcpus 2 --disk /VMs/Centos7.qcow2 \
--import --os-variant rhel7 --network default
```

The new VM guest will be connected to the *default* `libvirt` network which is a private network that facilitates network connectivity via Network Address Translation (NAT) between guest and host. The KVM host also runs a DHCP server which automatically assigns an IP address from this default network pool to the guest VM on boot (if the guest OS is configured accordingly).

Oracle Linux KVM also has its own configuration requirements to comply with the hard partitioning licensing policy and you must bind vCPUs to physical CPU threads or cores.

Conclusion

At AWS we are passionate about helping customers remove waste and solve challenging problems with elegant and cost effective solutions. Modern businesses have a variety of compelling pressures to minimise their IT operational expenditure and are constantly looking for ways to do more with less. The ability to run Oracle Hypervisors on EC2 bare metal instances in AWS opens the door to some additional license optimization techniques that can help you realise substantial savings in your IT budgets. CPU-based licensing, CPU over-subscription, and hard partitioning can all individually reduce your license requirements for proprietary Oracle software. By running an Oracle hypervisor on EC2 bare metal you can combine all three and gain the aggregate cost savings.

When this deployment pattern is combined with the power of the AWS control plane and integrated with the full suite of native AWS services such as Amazon Virtual Private Cloud (Amazon VPC), Elastic Block Store, and AWS CloudFormation, you can build a very secure and resilient operating environment for your Oracle workloads, with a very low total cost of ownership (TCO).

While this pattern may not be the default recommendation in an AWS migration strategy, it still provides a useful option for certain use cases and circumstances. This whitepaper is intended to inform customers of the possibilities rather than promote a particular migration approach. Your specific situation should be analyzed on a case by case basis to determine the migration approach that best fits your timelines, risk appetite, and AWS experience.

Contributors

Contributors to this document include:

- Matt McClernon, Data Transformation Architect, Amazon Web Services

Further reading

For more information related to running Oracle workloads on AWS, refer to the following resources:

Oracle licensing on AWS

- Licensing Oracle Software in the Cloud Computing Environment
- Running Oracle on AWS FAQs
- Third Party Guidance for Oracle Licensing on AWS
- Oracle Cost Traps and How to Overcome Them with AWS Solutions

Oracle Database on AWS:

- Advanced Architectures for Oracle Database on Amazon EC2
- Strategies for Migrating Oracle Database to AWS
- Best Practices for Running Oracle Database on AWS

Oracle on AWS

- Oracle and AWS
- Amazon RDS for Oracle

RDS Oracle Hands on Labs

- Amazon RDS Oracle Immersion Day

AWS service details

- Cloud Products
- AWS Architecture Center
- AWS Documentation
- AWS Whitepapers & Guides

AWS pricing information:

- AWS Pricing
- AWS Pricing Calculator

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Minor update	Fixed formatting of command line examples.	May 6, 2022
Initial publication	Whitepaper first published.	August 5, 2021

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.