

Oracle JD Edwards EnterpriseOne: Scalable Elastic Deployment

Leveraging Container Technology and Oracle Cloud
Infrastructure Autoscaling for Compute Service

TECHNICAL BRIEF

November 19, 2019

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. 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, and timing of any features or functionality described for Oracle's products remain at the sole discretion of Oracle.

Table of Contents

Executive Summary	3
Overview: Configuring Containers in Oracle Cloud Infrastructure	5
The Value.....	5
How It Works.....	5
Overview: Configuring AutoScale Utility in Oracle Cloud Infrastructure	8
The Value.....	8
How It Works.....	9
Bringing Together Containers and Autoscaling Utility in Oracle Cloud Infrastructure.....	11
Patching Benefits of Autoscaling.....	11
Oracle Cloud Infrastructure Autoscaling.....	11
Testing Oracle Cloud Infrastructure Autoscaling with Containers	12
Testing Methodology, Metrics, and Results	13
Overview: Alternative Scaling of JD Edwards EnterpriseOne using Containers.....	17
Oracle Cloud Infrastructure Requirements for an Autoscale Utility	18
Conclusion.....	20
Appendix A: Test Configuration	21
Appendix B: Configuring Oracle Cloud Infrastructure Autoscaling	22

Creating an Instance Pool and Autoscaling Configuration.....	23
Preparing the Load Balancer for Autoscaling.....	30
Autoscaling Test ScreenShots.....	36
Appendix C: Configuring Oracle Cloud Infrastructure	
Autoscaling Utility for Patching	46

EXECUTIVE SUMMARY

In today's landscape, business requirements are always evolving. IT decision makers are looking for products that improve their employees' efficiency throughout the life cycle of ERP administration. The JD Edwards team has been investigating mechanisms to lower the ongoing total cost of ownership of your JD Edwards solution and deliver on the promises of simplified system administration, cloud automation, elasticity, and scalability.

These are some of the challenges that our customers face today:

- Your business experience includes peak processing times that require additional compute resources outside of the norms.
- System maintenance and patching and security operations across your expansive global footprint pose a challenge.
- You have divested businesses and need to quickly and easily align your systems to the changing business.

In response to these challenges, the JD Edwards team, along with the Oracle Cloud Infrastructure counterparts, has been automating processes to enable autonomous elasticity and scalability of your servers and their associated resources.

So What Is Elasticity?

Simply put, elasticity is your system's ability to balance your ever-changing processing requirements against the system resources needed to manage your workloads.

Why Is Elasticity Important?

You only pay for what you use! Rather than having servers and processing capacity that remain idle during your lower usage periods, you can have an infrastructure that can expand and contract based on your workloads to ensure peak performance during the periods you need higher processing power. Customers can realize significant cost savings by leveraging elasticity.

This technical brief describes how customers can leverage Oracle Cloud Infrastructure autoscaling for Compute Service with container technology to "right-size" their compute fleet and save infrastructure costs by scaling the workload tier dynamically and on demand.

The Oracle Cloud Infrastructure autoscaling enables automatic adjustment to the number of compute instances in an instance pool based on performance metrics such as CPU and memory utilization. Such automatic adjustment provides consistent performance to end users during periods of high demand and helps reduce the costs during periods of low demand. For more information on the Oracle Cloud Infrastructure feature on autoscaling, see [Oracle Cloud Infrastructure – Autoscaling](#).

For information about Oracle Optimized Solution for JD Edwards EnterpriseOne—a complete architecture that provides high-performance, high-availability, secure, and low-cost deployments— see [Implementing High Performance and Availability for JD Edwards EnterpriseOne](#).

This technical brief describes the mechanics of using the autoscaling feature in Oracle Cloud Infrastructure and how this technology is combined with the container technology and the elastic unit concept to provide a scalable versatile solution to a JD Edwards EnterpriseOne architecture.

Container technology is a method of packaging JD Edwards EnterpriseOne components as a unit so that the JD Edwards system can be run with its dependencies, isolated from other processes. The key thing to recognize with containers in Oracle Cloud Infrastructure is that they are designed to virtualize a single application by creating an isolation boundary at the application level rather than at the server level.

For more information, see:

[JD Edwards EnterpriseOne on Oracle Cloud](#)

[Deploying Containerized JD Edwards EnterpriseOne on Oracle Cloud Infrastructure](#)

OVERVIEW: CONFIGURING CONTAINERS IN ORACLE CLOUD INFRASTRUCTURE

Containers are open-source tools designed to make it easier to create, deploy, and initiate applications. Containers enable JD Edwards EnterpriseOne to implement an application, whether that be a JD Edwards EnterpriseOne logic, batch, or HTML component, with all the needed libraries and other dependencies, and deliver the application as a single package or image. Oracle has delivered many of these packaged images as standard repository images on Oracle Cloud Infrastructure. This section discusses the value of containers and provides an overview of how a container works.

The Value

Because all repository images in the container are initiated on top of the operating system, delivering these container images involves the sharing of this underlying operating system. Unlike a virtual machine (VM), containers enable applications to use the same operating system kernel providing a significant performance boost and reducing the size of the application.

You can quickly create VM images for any of the JD Edwards EnterpriseOne components such as logic, batch, and HTML servers. The container implementation goal is simple—make it easy to package and ship a code. In other words, the goal is to make the code as portable as possible and make that portability simple. The ability of containers to quickly initiate processes enables the creation of a complete VM image, including JD Edwards EnterpriseOne services in seconds instead of minutes.

For detailed information about using the container technology in the Oracle Cloud Infrastructure, see the technical brief [Deploying Containerized JD Edwards EnterpriseOne on Oracle Cloud Infrastructure](#).

How It Works

To a system architect or application developer who is already familiar with the overall structure of JD Edwards EnterpriseOne applications, the solution is quite simple, yet powerful. This document will introduce the concept of an elastic unit. Elastic units provide a supported JD Edwards EnterpriseOne architecture to minimize any changes to the JD Edwards EnterpriseOne configuration by defining an instance with a JD Edwards EnterpriseOne Enterprise Server and JD Edwards EnterpriseOne HTML Server. This elastic unit concept will serve as a starting point for more complex uses such as autoscaling and portability.

ELASTIC UNIT INSTANCE

The Oracle Cloud Infrastructure elastic unit is a single instance that encompasses a single JD Edwards EnterpriseOne Enterprise Server and EnterpriseOne HTML or JAS Server as its container components. The setup of an Oracle Cloud Infrastructure elastic unit is shown in the figure below.

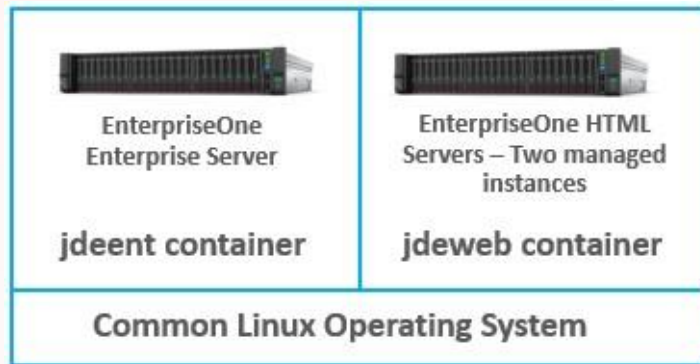


Figure: Oracle Cloud Infrastructure Elastic Unit Instance

As a unit, this JD Edwards EnterpriseOne and JAS Server pair services all JD Edwards EnterpriseOne external requests. The HTML server in the elastic unit receives the JD Edwards EnterpriseOne request and forwards that requested workload to the corresponding JD Edwards EnterpriseOne Enterprise Server in the elastic unit set.

The advantage of this design is that the need for any configurations using the JD Edwards EnterpriseOne Object Configuration Management (OCM) is eliminated. The HTML and Enterprise servers work as a unit—there is a 1-1 correspondence between the HTML Server and the JD Edwards EnterpriseOne Enterprise Server that will execute the JD Edwards EnterpriseOne requests

When the load parameters are set within the elastic unit, through a scaling mechanism the instance can be replicated for JD Edwards EnterpriseOne customers in increments of that load parameter. For example, a JD Edwards EnterpriseOne elastic unit can be configured to support 100 interactive users and if the user load increases beyond 100, then a second elastic unit can be initiated to service those users. The elastic unit also works in the reverse scenario. The JD Edwards system may have two elastic units, each capable of supporting 100 interactive users. A CPU threshold can be set so that if an elastic unit resource is not used, it can be removed from the JD Edwards EnterpriseOne architecture. Thus the model of elastic units works well for scaling up and down.

For each elastic unit, the machine name of the container for the JD Edwards EnterpriseOne Enterprise Server is “jdeent” and that of the container for the JD Edwards EnterpriseOne HTML Server is “jdeweb”, further simplifying any OCM configurations normally required if unique machine names are used. The only difference between replicated containers is the web tier port for which the JD Edwards EnterpriseOne requests are directed.

The JD Edwards EnterpriseOne web traffic either can be automatically directed to a load balancer, which manages the incoming network traffic to a set of back-end servers, or can be manually directed by users to the individual servers for servicing the JD Edwards EnterpriseOne requests. This configuration for directing network traffic is based on the customer’s decision during the JD Edwards EnterpriseOne architecture design implementation.

For the purpose of this document, an Oracle Cloud Infrastructure managed load balancer is illustrated. The benefit of an Oracle Cloud Infrastructure Load Balancer as a Service (LBaaS) is seamless updates to the load balancer that can be made through the autoscaling utility. This functionality makes the load balancer a hands-free solution for automating the direction of network traffic as JD Edwards EnterpriseOne servers are added and removed from the architecture.

The JD Edwards EnterpriseOne database server is notably missing as a container component within the elastic unit diagram; this is by design. The container technology performs well with many EnterpriseOne components such as the Enterprise server, HTML server, and AIS server to name a few. However, it does not perform well on a JD Edwards EnterpriseOne Database Server. Configuring a JD Edwards EnterpriseOne database container may be acceptable for testing and development purposes, but it is not recommended for use in a production mode and thus are excluded in the diagrams in this document.

Elastic units in a similar way can assist in patching. The simplest patching model, which is covered later in this document, consists of a simple operating system, software updates, security patches, and other non-JD Edwards EnterpriseOne updates that have few dependencies.

OVERVIEW: CONFIGURING AUTOSCALE UTILITY IN ORACLE CLOUD INFRASTRUCTURE

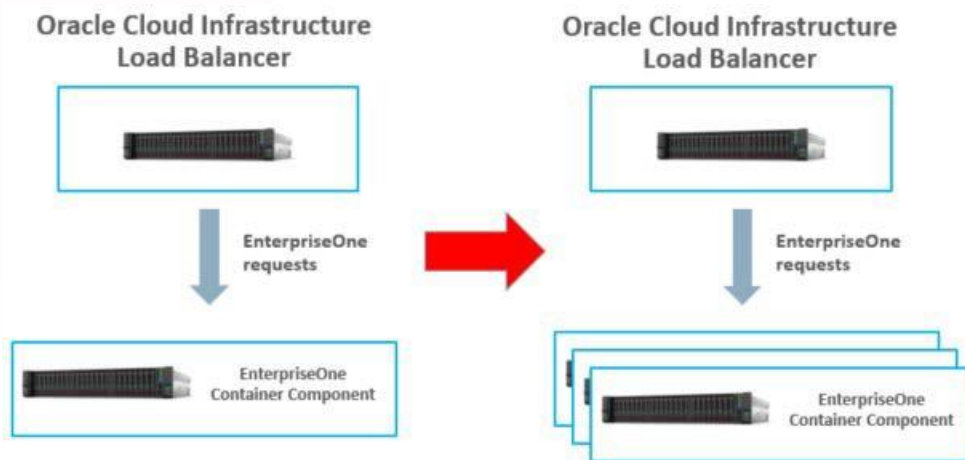
The Oracle Cloud Infrastructure autoscaling utility is an important addition to the Oracle Cloud Infrastructure framework, introducing the capability of launching VM images automatically based on CPU consumption or memory usage metrics. This new capability extends the possibilities of intelligence and automated scaling to yet a higher level.

Autoscaling allows adjustments to the number of compute instances in an instance pool based on the performance metrics of CPU or memory. The JD Edwards EnterpriseOne application scaling model can leverage this Oracle Cloud Infrastructure feature during periods of high usage when more hardware resources are required to meet business needs. The autoscaling feature also allows for an automatic scale-down which in turn reduces the JD Edwards EnterpriseOne footprint when limited resources are needed during periods of lower demand.

For more information on the Oracle Cloud Infrastructure autoscaling utility, see [Oracle Cloud Infrastructure – Autoscaling](#).

The Value

The Oracle Cloud Infrastructure autoscaling feature facilitates the replication of any of the JD Edwards EnterpriseOne container components based on the simple metrics of CPU or memory as depicted in the figure below:



To understand how these replicated compute instances function in this architecture, an Oracle Cloud Infrastructure Load Balancer is included in the requirements of scaling using this methodology.

The JD Edwards EnterpriseOne Database Server is not shown in the above diagram; the database is configured as a DBCS instance within Oracle Cloud Infrastructure. The DBCS instance is managed separately from the virtual machines in Oracle Cloud Infrastructure.

The Oracle Cloud Infrastructure LBaaS server also works with the Oracle Cloud Infrastructure autoscaling utility. When another JD Edwards EnterpriseOne component is added to the compute instances managed by the Oracle Cloud Infrastructure autoscaling utility, the utility also updates the Oracle Cloud Infrastructure LBaaS server with the added component.

For example, another JD Edwards EnterpriseOne HTML compute instance can be created. Furthermore, that JD Edwards EnterpriseOne web tier component and its port access will be controlled by an Oracle Cloud Infrastructure LBaaS device. The Oracle Cloud Infrastructure LBaaS server will automatically pick up the new port and provide service.

This automated mechanism eliminates the need for any manual configuration when a new resource that can provide JD Edwards EnterpriseOne services is available.

How It Works

Throughout the design process, you will make numerous decisions that will affect the behavior and ultimately determine the success of your solution. When considering autoscaling, the most important questions are:

- What is the maximum number of compute instances that will be allowed?
- What is the CPU or memory scale out threshold of the VM instance beyond which the autoscaling event is triggered?
- What is the threshold of CPU or memory to scale down or scale in compute instances?

When the system reaches the CPU or memory threshold, the autoscaling utility dynamically creates additional compute instances in near real time. As the load increases, upward scaling is initiated and the instances are scaled out. When the load decreases below a defined threshold, autoscaling automatically removes instances from the compute pool.

The autoscaling metric threshold is measured every minute. If three consecutive minutes pass and the threshold is met, an autoscaling event occurs (scale out or scale in). A cool-down period or the time between triggered events of approximately five minutes is automatically configured to allow the compute instances to stabilize before any new triggered event can occur. The period for stabilizing the environment is called the cool-down period and is configurable within the autoscaling utility. Five minutes is the default and minimum period for cool down, but a customer has the control to change this value.

The Oracle Cloud Infrastructure autoscaling function is invoked synchronously (in-line), meaning that it is executed outside of the currently running compute instance.

CHOOSING THE CPU OR MEMORY LIMIT BEYOND WHICH TO SCALE ANOTHER COMPUTE INSTANCE

The Oracle Cloud Infrastructure business model for scaling requires users to determine the right initial shape by selecting the CPU or memory threshold limits beyond which autoscaling is to be initiated (scale up and scale in) and setting the maximum number of autoscaling elastic units.

Option1: Choose an initial small shape and then scale horizontally.

Each shape should support 75–150 users. The granularity of this approach is to limit the cost and the number of resources.

Option 2: Choose an initial large shape and then scale horizontally.

This approach is preferable if the customer does not want scaling events to be executed frequently.

For scaling up, a starting CPU limit of 70–80% should be chosen. This is a typical limit on CPU usage beyond which performance of the application starts to degrade. For the testing results presented in this document, a CPU limit of 70% was used. For memory, a similar 70–80% limit should be set as the threshold for similar performance concerns.

For scaling down or in, a starting CPU limit of 1–5% should be used as a best practice. For JD Edwards EnterpriseOne Enterprise Servers, the consumption for an idle CPU is approximately 2%. The consumption of an idle JD Edwards EnterpriseOne HTML Server is appropriately 2% as well.

CHOOSING THE NUMBER OF COMPUTE INSTANCES

The recommended starting number of compute instances for autoscaling is three. This number depends largely on the customer requirements, but for initial configuration, three compute instances will ensure sufficient scalability during maintenance, periods of increased load.

The number of compute instances depends largely on the initial shape of the master compute instance in the pool to be replicated. If an initial small shape is chosen, more compute instances may be required to handle all the load and peak throughput requirements of a customer. If an initial large compute instance shape is chosen, fewer number of compute instances will have to be defined.

BRINGING TOGETHER CONTAINERS AND AUTOSCALING UTILITY IN ORACLE CLOUD INFRASTRUCTURE

The container toolset and the Oracle Cloud Infrastructure autoscaling utility are two separate technologies that can function independently of each other. However, when combined they can provide a superior model for scalability in the JD Edwards EnterpriseOne architecture.

A simple task is accomplished through Oracle Cloud Infrastructure autoscaling with containers—bringing up additional elastic units based on a single hardware resource threshold metric, namely CPU or memory consumption, as an input to trigger a series of autoscaling events.

Patching Benefits of Autoscaling

Autoscaling is the capability of initiating a manual scaling of resources using the autoscaling utility for patching the JD Edwards EnterpriseOne web tier components of the JD Edwards EnterpriseOne architecture.

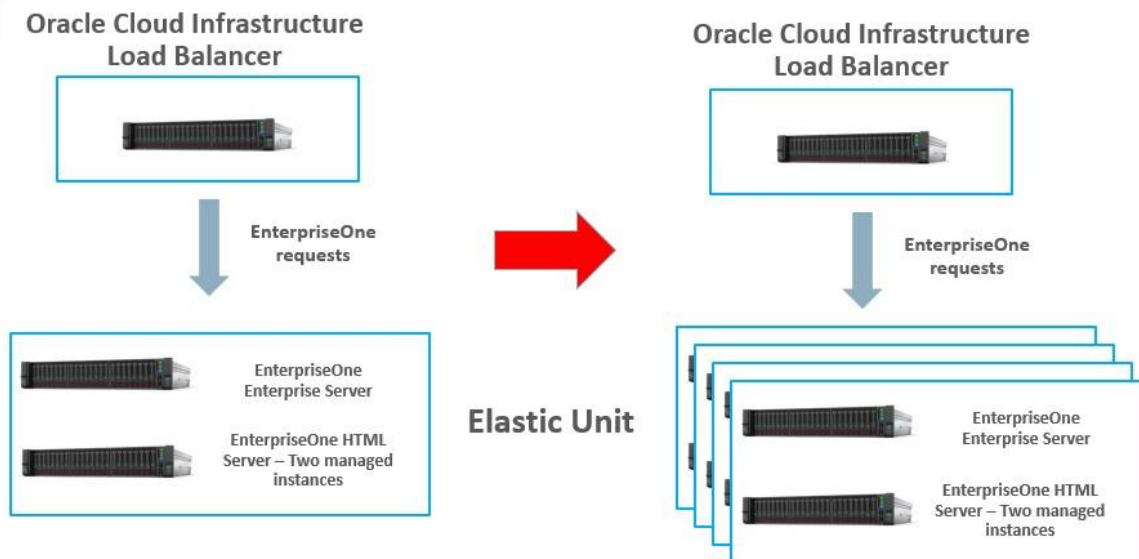
The goal is to minimize downtime for system maintenance and improve the deployment of infrastructure patches. This is beneficial especially for customers that have large footprints of servers across multiple time zones.

Patches can include operating system and other infrastructure patches, security patches, and particular database patches across each of the installed instances of JD Edwards EnterpriseOne.

The idea is to patch one of the JD Edwards EnterpriseOne component servers (enterprise, HTML, database, and so on) and deploy the patched server across their infrastructure, saving time and money.

Oracle Cloud Infrastructure Autoscaling

This section of the document describes the Oracle Cloud Infrastructure autoscaling process using containers and a JD Edwards EnterpriseOne elastic unit instance. For this example, both autoscaling and container technologies are combined to provide a solution to the customer.



Assume that a customer has a single elastic unit instance with two containers, a JD Edwards EnterpriseOne Enterprise Server and a HTML Server. This single JD Edwards EnterpriseOne environment would support an estimated 150 interactive and batch processes. The problem the customer is facing is that the hardware resources are sufficient for processes during the day, but are insufficient for processes during the night.

To solve this issue using the Oracle Cloud Infrastructure autoscaling utility, the elastic unit VM is put under the control of the Oracle Cloud Infrastructure autoscaling utility as shown in the figure above. The configuration of the VM controlled by the Oracle Cloud Infrastructure autoscaling utility is included in the appendix of this document.

The Oracle Cloud Infrastructure autoscaling process is as follows:

1. Oracle Cloud Infrastructure measures the per minute CPU or memory consumption of the elastic unit instance configured for autoscaling.
2. The threshold for scale up is set to 70% CPU. The threshold is achieved and is maintained for three minutes, after which the Oracle Cloud Infrastructure autoscaling utility event is triggered.
3. If a replication (scale up) event occurs, the VM is replicated and the new server or port for the WebLogic configuration is added to the load balancer (LBaaS) configuration. In a test environment, this process takes 5–10 minutes.

No OCM change is required; the load balancer is automatically configured by the Oracle Cloud Infrastructure autoscaling utility and will direct traffic to the new elastic unit when it is available.

The net effect is a scalable solution that meets a large number of customers' needs, which the former manual scaling process and configuration system could not meet. The new process of autoscaling transforms the older process into an intertwined and interdependent set of processes that shift tedious, time-consuming, and error-prone tasks from the user to a more autonomous JD Edwards EnterpriseOne system with a higher degree of intelligence. This is the path to digital transformation.

Testing Oracle Cloud Infrastructure Autoscaling with Containers

This section of the document covers the use cases, application behavior, and results of testing an Oracle Cloud Infrastructure autoscaling utility that is implemented with two JD Edwards EnterpriseOne component containers, the enterprise server and the HTML server, in a single elastic unit.

USE CASES

This investigation used a set of simple use cases that were designed to exercise and demonstrate certain aspects of launching additional elastic units using the Oracle Cloud Infrastructure autoscaling utility.

Because the CPU or memory threshold limit can be reached through load to either the JD Edwards EnterpriseOne Enterprise Server component (logic container) or the JD Edwards EnterpriseOne HTML Server component (WebLogic container), use cases were developed for each of the scenarios.

Both the JD Edwards EnterpriseOne HTML Server and Enterprise Server share common operating system resources. As a result, irrespective of the design of the container components, the impact of load on CPU utilization is the same for both the server components. The use cases presented in this document are:

1. Triggering scaling event on the JD Edwards EnterpriseOne HTML Server

User load is increased until CPU limit (threshold is 50%) is achieved on the EnterpriseOne HTML Server for three consecutive minutes.

In this use case, it is assumed that the CPU threshold is achieved triggering the Oracle Cloud Infrastructure autoscaling event to scale up as a direct result of JD Edwards EnterpriseOne HTML container utilization.

2. Triggering scaling event on the JD Edwards EnterpriseOne Enterprise Server

Batch processing is increased to achieve a specific CPU load on the JD Edwards EnterpriseOne Enterprise container. Increasing the batch load is only on the enterprise server container serves as a good use case for testing the enterprise server CPU resource consumption.

In this use case, it is assumed that the CPU threshold is achieved triggering the Oracle Cloud Infrastructure autoscaling event to scale up as a direct result of JD Edwards EnterpriseOne Enterprise Server container utilization.

Testing Methodology, Metrics, and Results

This section of the document describes the testing methodology and results of using the tools of Oracle Cloud Infrastructure autoscaling utility in an elastic unit configuration.

TESTING METHODOLOGY

The Oracle JD Edwards EnterpriseOne Day-in-the-Life (DIL) Kit was used in the testing of the Oracle Cloud Infrastructure autoscaling utility using containers to create the background load to the environment. The DIL Kit is a set of Oracle internally automated load testing scripts for generating various interactive loads in a JD Edwards EnterpriseOne architecture environment. The DIL Kit comprises 25 interactive applications across five functional modules, namely Customer Service Management, Finance Management, Human Capital Management, Supplier Relationship Management, and Supply Chain Management. Another component of the DIL Kit is the 1.2 TB foundational database that applications use, and which is designed to represent an average-sized JD Edwards EnterpriseOne customer.

Each of the use cases for testing consisted of the following steps and had a duration of an hour. These use cases were imitated concurrent to the background load of the DIL Kit to replicate a real customer. The following steps were performed in these use cases:

1. Logging in to the JD Edwards EnterpriseOne application as a specific JD Edwards EnterpriseOne role or user
2. Initiating the DIL Kit interactive and batch processes for a specific interactive and batch load—including increased numbers of interactive users—to target the CPU resources utilized by either the JD Edwards EnterpriseOne Enterprise Server or the JD Edwards EnterpriseOne HTML Server
3. Logging out of the JD Edwards EnterpriseOne application

For each test, a number of key metrics were collected for performance analysis.

TESTING METRICS (KEY PERFORMANCE INDICATORS)

Key performance indicators are the critical metrics that create the analytical basis for decision making. These metrics provide an indication as to whether the application is performing within the expected margins or whether the performance is below acceptable standards and may lead to customer dissatisfaction. The following key performance indicators were used in testing Oracle Cloud Infrastructure autoscaling under a managed set of containers:

END USER EXPERIENCE

It is generally considered that the end user experience defines the acceptable level of performance of an application. End user experience is the time between a button click within a browser and the moment when results are displayed on the browser or the next page is displayed to the user. Generally, a period between 0.10 seconds (100 ms) and 0.25 seconds (250 ms) can make a difference in end user experience.

The acceptable level of performance varies, but is generally considered to be between 3–7 seconds, with 3 seconds being the generally accepted norm. End user experience is also determined by the server performance and the browser rendering process. Selenium test automation scripts of specific JD Edwards EnterpriseOne applications form the basis of measuring end user experience metric.

SERVER SIDE METRICS

Server side metrics are good indicators of any problem with the processing of data that is transferred from a web server request made from the browser. Server side metrics include the time between the HTML request and the moment the response is initially returned to the web server. These metrics do not include the time the end user experiences due to browser rendering.

The common variables that impact the server side metrics are network latency, network bandwidth, and application and database performance when these requests are processed. The test group uses the Oracle Application Testing Suite (OATS) utility to measure the server side metrics.

OPERATING SYSTEM METRICS

The standard operating system metrics of CPU (processor), memory, disk I/O, and network throughput are used as an indicator of performance.

TESTING ORACLE CLOUD INFRASTRUCTURE AUTOSCALING UTILITY

For simplicity, the JD Edwards EnterpriseOne Purchase Order application (P4310) was used to measure the end user experience (including server side metrics and browser rendering) to provide the analytics to measure the baseline performance.

A baseline is the starting point used for comparisons. For testing the Oracle Cloud Infrastructure autoscaling utility, two baselines are needed, one with a single elastic unit and the other with dual elastic units. The performance of the application when supporting only interactive users and when supporting interactive users with concurrent batch processing was tested.

GOAL

The goal of the baseline comparison is to determine if there is any significant difference in the end user experience (performance) between a JD Edwards EnterpriseOne architecture with a single elastic unit configuration and an architecture with a dual elastic unit configuration after the replication of a new compute image is complete.

DISCUSSION

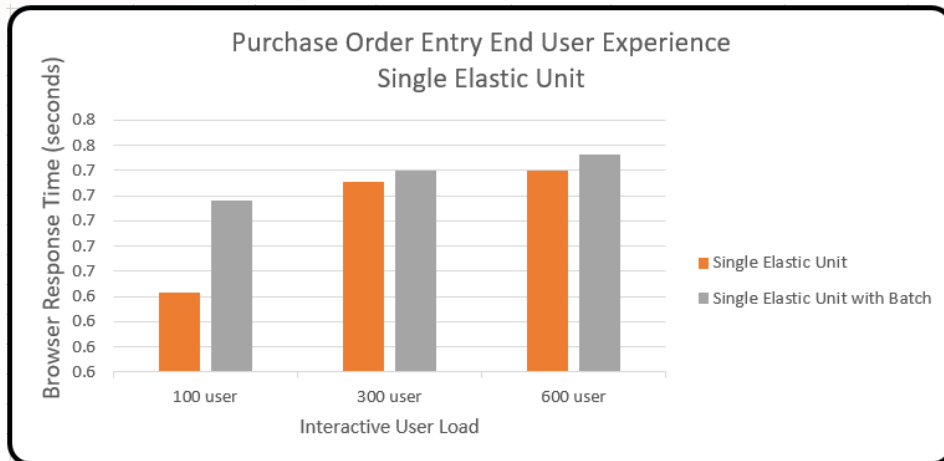
For this comparison, the DIL Kit was used and three load levels of interactive end user experience were measured. The levels of 100, 300, and 600 were chosen for this comparison.

The testing of the Oracle Cloud Infrastructure autoscaling utility also involved the use of an Oracle Cloud Infrastructure LBaaS. The goal of the testing was to be sure that the end user experience was consistent throughout the autoscaling (scale up and scale in) process.

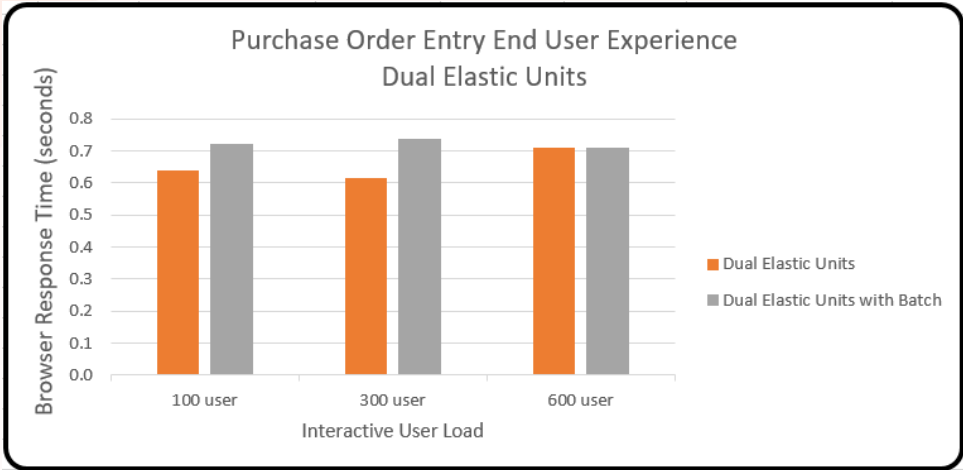
The scale-in threshold used in this testing was 50% CPU usage over a three-minute interval to trigger the scaling event. The testing was performed for an hour, during which a single elastic unit was scaled out to two elastic units and returned to a single elastic unit at the end of the test.

RESULTS

The figure below shows the baseline end user response times for interactive user levels of 100, 300, and 600 for a single elastic unit configuration.



The figure below shows the baseline end user response times for interactive user levels of 100, 300, and 600 for a dual elastic unit configuration.



The interactive user levels of 100 and 300 did not trigger any autoscaling event due to JD Edwards EnterpriseOne Enterprise Server load. However, a load of 600 users was sufficient to maintain a 50% CPU load for three consecutive minutes.

During the Oracle Cloud Infrastructure autoscaling event, a new elastic unit became available approximately 10 minutes after the event was triggered. The load balancer was updated and new requests were sent to both the HTML servers in their respective elastic unit containers.

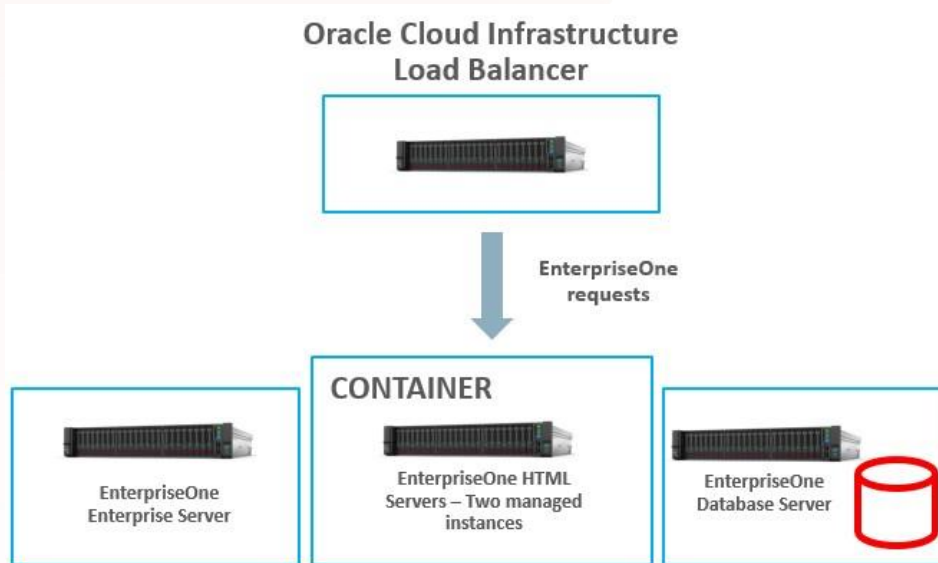
At the end of testing when the JD Edwards EnterpriseOne Enterprise Server and HTML Server became idle, another scale in event was triggered and the JD Edwards EnterpriseOne architecture returned to a single elastic unit configuration. The time interval between the triggered events (cool down setting) are configurable (the default is a minimum of five minutes) and in this test, the second triggered event occurred as expected.

A similar test with 100, 300, and 600 interactive users was performed with concurrent batches being processed by the JD Edwards EnterpriseOne Enterprise Server. In this case, the 50% threshold was triggered by the 300 and 600 interactive user levels due to the additional consumption of resources by the batch processing.

In the autoscaling events, a baseline end user response time ranging between 0.62 seconds (620 ms) and 0.75 seconds (750 ms), was observed. This represents a spread of 0.13 seconds (130 ms), which is an acceptable level of performance variance.

OVERVIEW: ALTERNATIVE SCALING OF JD EDWARDS ENTERPRISEONE USING CONTAINERS

An alternative to the Oracle Cloud Infrastructure autoscaling method is to use the JD Edwards EnterpriseOne HTML Server as the sole managed container within an elastic unit as depicted in the below figure.



The above figure shows a single instance managing a single HTML server with two managed instances. This configuration is a valid JD Edwards EnterpriseOne architecture and does not require any additional configuration if used with the Oracle Cloud Infrastructure autoscaling utility.

If the goal is to quickly replicate the JD Edwards EnterpriseOne system, instead of using the Oracle Cloud Infrastructure autoscaling utility with a full JD Edwards EnterpriseOne HTML Server VM, consider placing the JD Edwards EnterpriseOne HTML Server under the control of containers. This option enables quick start and almost immediate availability of the instance to users.

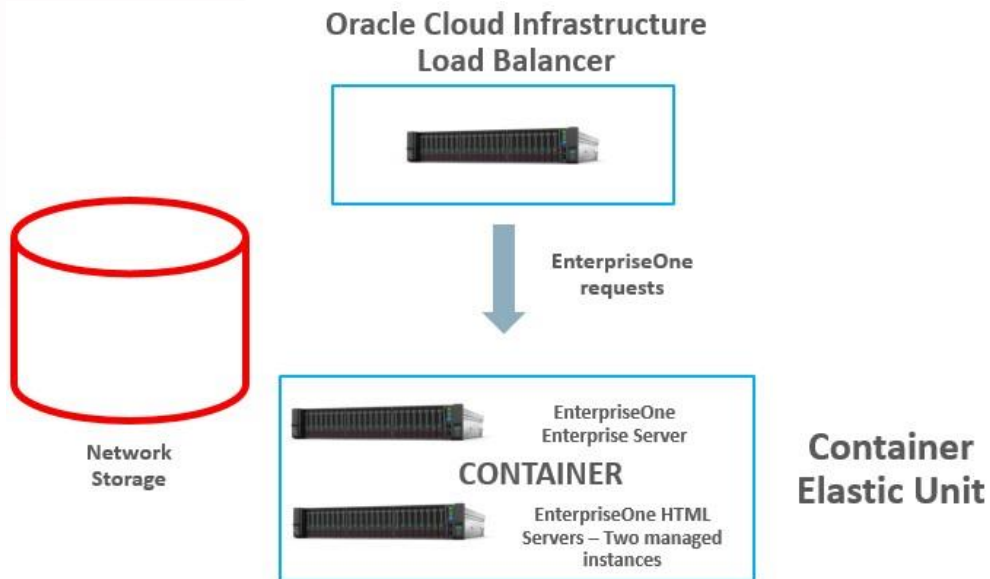
ORACLE CLOUD INFRASTRUCTURE REQUIREMENTS FOR AN AUTOSCALE UTILITY

The [Oracle Cloud Infrastructure Documentation](#) provides detailed information about the infrastructure requirements for autoscaling.

The high-level requirements can be summarized as follows:

- You have an instance pool and can optionally attach a load balancer to the instance pool. For steps to create an instance pool and attach a load balancer, see the section [Creating an Instance Pool](#) of the document.
- Monitoring is enabled on the instances in the instance pool. For steps to enable monitoring, see the section [Enabling Monitoring for Compute Instances](#) of the document.
- The instance pool supports the maximum number of instances that you want to scale to. This limit is determined by your tenancy's service limits. For more information, see the section [Service Limits](#).

Two additional Oracle Cloud Infrastructure components that are strongly recommended to be implemented with the Oracle Cloud Infrastructure autoscaling utility and the container utilities, are an Oracle Cloud Infrastructure load balancer and some form of network storage. Each of these two components are illustrated in the figure below and are subsequently described in detail.



Load Balancer

The function of the load balancer component of the Oracle Cloud Infrastructure is to automate the balancing of new HTML requests to the JD Edwards EnterpriseOne application after an autoscaling event.

Network Storage Component

The network storage component configuration is introduced as a central repository for any log files and other configurations that need to be saved prior to a scale down event. The network storage component is required because the Oracle Cloud Infrastructure autoscaling utility removes an elastic unit compute instance when scaling down, meaning any files added and changes made to that elastic unit is otherwise lost.

CONCLUSION

The Oracle Cloud Infrastructure autoscaling utility combined with a container utility is a powerful way to transform how you use your JD Edwards EnterpriseOne system to automate business processes and shift work from users to machines. The ability to launch additional elastic units from events within the Oracle Cloud Infrastructure environment itself vastly expands the possibilities for digital transformation.

The use cases discussed in this technical brief, along with the observed results and analyses, illustrate some effective and practical design patterns that can be followed as best practices. The choices about which events to scale out and scale in as elastic units are the design decisions that have the highest impact on the performance of your solution and the accessibility of JD Edwards EnterpriseOne architecture.

By automating processes using the Oracle Cloud Infrastructure autoscaling utility, you can shift work from users to machines, but in doing so you must also consider the effect of increased system resource consumption and the cost of those services on the Oracle Cloud Infrastructure platform.

When your business analysts have the Oracle Cloud Infrastructure autoscaling feature with elastic unit running efficiently, system administrators should observe and tune the JD Edwards EnterpriseOne system parameters, such as security kernels, call object kernels, and threads. It is equally important to measure performance metrics and test performance in a nonproduction environment during the design and test phase and to continuously monitor how the system behaves in the production environment. The success of your implementations can be achieved through a partnership among business analysts, system administrators, and system architects.

Using the Oracle Cloud Infrastructure autoscaling utility to expand the number of components enables you to patch any of the JD Edwards EnterpriseOne components within the autoscaling pool while the utility provides service to the original image. There will be a transition when moving users from the unpatched environment to the new environment, but the Oracle Cloud Infrastructure autoscaling utility can make this transition seamless.

The goal of this document has been to create an understanding through theoretical discussion as well as through observation, measurement, and analyses made as part of several use cases, so that you might implement the Oracle Cloud Infrastructure autoscaling utility with greater knowledge and foresight. As a result, you can ensure that your JD Edwards EnterpriseOne application is scalable and use it in new ways to move toward digital transformation.

APPENDIX A: TEST CONFIGURATION

The following section provides details of the test environment. The JD Edwards EnterpriseOne Enterprise Server and HTML Server are configured as containers within the same elastic unit instance.

JD Edwards EnterpriseOne Enterprise Server:

- Oracle Enterprise Linux 6
- Oracle Database12c (12.1.0.2) 32-bit Client
- 4 VCPUs x Intel Xeon CPU E5-2697 @ 2.90 GHz
- 12 GB RAM

Database Server:

- Oracle Enterprise Linux 6
- Oracle Database12c (12.1.0.2) Enterprise Edition 64 bit
- 8 VCPUs x Intel Xeon CPU E5-2697 @ 2.90GHz
- 30 GB RAM

HTML Server (HTML1, HTML2a, and HTML2b):

- Oracle Enterprise Linux 6
- 6 VCPUs x Intel Xeon CPU E5-2697 @ 2.90 GHz
- 16 GB RAM
- WebLogic Server 12c (12.1.3); Java JDK (1.8)
- Single Managed Instance (4 GB Heap Size)

AIS Server:

- Oracle Enterprise Linux 6
- 4 VCPUs x Intel Xeon CPU E5-2697 @ 2.90 GHz
- 16 GB RAM
- WebLogic Server 12c (12.1.3); Java JDK (1.8)
- Single Managed Instance (4 GB Heap Size)

Deployment Server:

- Windows 2012 R2 Enterprise Edition
- 2 VCPUs x Intel Xeon CPU E5-2697 @ 2.90 GHz
- 8 GB RAM

Server Manager Console:

- Oracle Enterprise Linux 6
- 1 VCPUs x Intel Xeon CPU E5-2697 @ 2.90 GHz
- 132 GB RAM

OATS Test Controller:

- Windows 2012 R2 Enterprise Edition
- 8 VCPUs x Intel Xeon CPU E5-2697 2.90 GHz
- 32 GB RAM
- Oracle Application Testing Suite 12.3.0.1.0.376

Software: JD Edwards EnterpriseOne Application 9.2 Update 2 with Tools Release 9.2.3.1

APPENDIX B: CONFIGURING ORACLE CLOUD INFRASTRUCTURE AUTOSCALING

This section describes the configuration steps for setting up the Oracle Cloud Infrastructure autoscaling utility. The container that is used in this process is Docker.

High Level Outline

Prerequisites—Elastic Unit and Load Balancer have been created and configured.

1. Save configured elastic unit instance as a custom image.
2. Create a model instance using the custom image with the desired shape and storage.
3. Create an instance configuration based on newly created instance.
4. Create an instance pool based on the instance configuration.
5. Create the autoscaling configuration and policy.
6. Use Oracle Cloud Infrastructure Console to monitor the instance pool and scaling activities.

Outline with Details

Prerequisites

1. Prepare a master copy of an elastic unit. Ensure that this instance is set up to automatically start the containers and managed instances in the correct order. After the master copy is prepared and tested, clean up any logs that exist for the instance or set up the installation software so that the instance is not continuously replicated.
2. Prepare a load balancer to be used in your autoscaling environment. Here, you will determine a front end listener port and a back-end set to attach to the autoscaling configuration among other settings to facilitate the desired behavior during autoscaling.

Autoscaling Setup

1. Save the master elastic unit as a custom image. This custom image will be the source for accurate information about the OS and software state.
2. Bring up a model instance using the custom image with the exact shape and boot volume size desired for members of the autoscaling pool.
3. Using the model instance, create an instance configuration in your desired compartment.
4. Create an instance pool based on the new instance configuration. While defining the pool, you will choose how many instances to create initially in the pool and you will attach your load balancer.
5. Create an autoscaling configuration and policy in which you will define the following:
 - a. Cool Down Period: The time interval between autoscaling actions before evaluating the performance metric for further actions.
 - b. Performance Metric: Either CPU or memory utilization to trigger autoscaling actions.
 - c. Scaling Limits: The minimum number of instances, the maximum number of instances, and the initial number of instances in your autoscaling pool.
 - d. Scaling Rule: The threshold percentages to trigger actions to scale out (create new instances in the pool), or to scale in (terminate instances in the pool).
6. Use Oracle Cloud Infrastructure Console to monitor the instance pool and scaling activities.

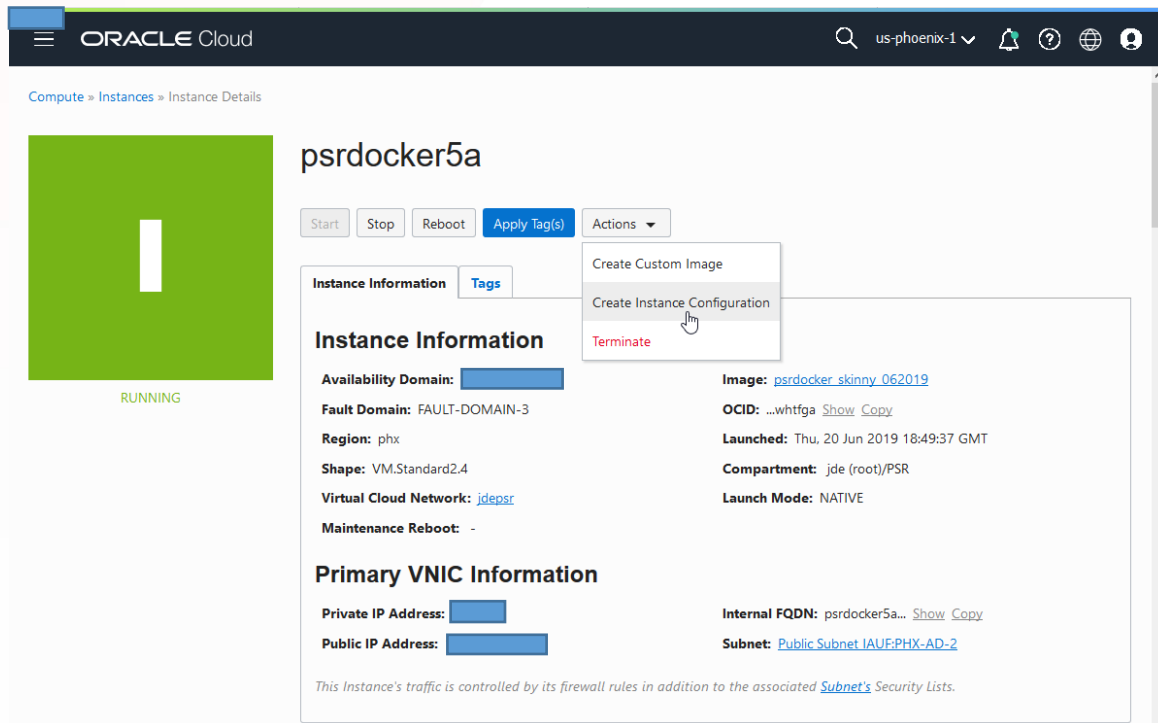
Creating an Instance Pool and Autoscaling Configuration

Assumptions and Prerequisites:

- Oracle Cloud Infrastructure Tenancy, Compartment, Security Lists, and other infrastructure are established.
- Elastic unit is prepared, tested, finalized, and saved as a final custom image.
- External RAC database is configured and tested with elastic unit.
- Final custom image in preferred state is used to bring up a model instance for the pool.

NOTE: These instructions are based on the current processes and user interface in Oracle Cloud Infrastructure. The process and features of the Oracle Cloud Infrastructure autoscaling utility along with the user interface could be changed and enhanced in the future.

1. Open the model instance to use as a base of the autoscaling pool.
2. Click **Create Instance Configuration** from the Actions drop-down list.



3. Select the desired compartment, give the instance configuration a meaningful name, and click the **Create Instance Configuration** button.

Create Instance Configuration from Instance [help](#) [cancel](#)

This will create a new Instance Configuration based on the Instance 'psrdocker5a'.

CREATE IN COMPARTMENT

PSR

jde (root)/PSR

INSTANCE CONFIGURATION NAME

psrdocker5-062019

Create Instance Configuration Cancel

- On the Instance Configurations page, click the newly created instance configuration to open the details screen.

ORACLE Cloud

US West (Phoenix)

Compute

Instance Configurations

Name	Created
psrdocker5-062019	Thu, Jun 20, 2019, 22:04:24 UTC

Showing 1 Item < Page 1 >

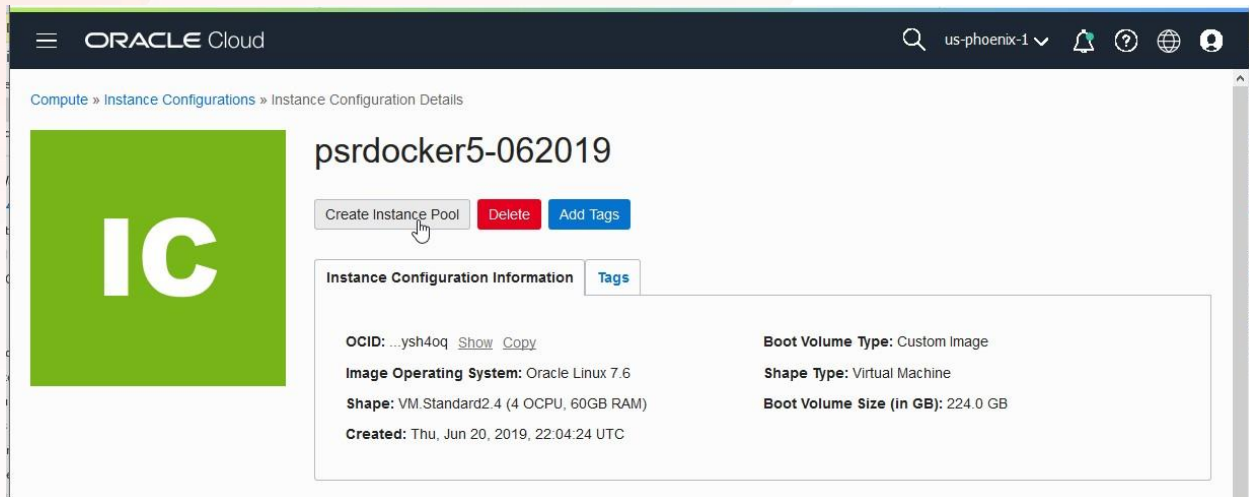
COMPARTMENT

PSR

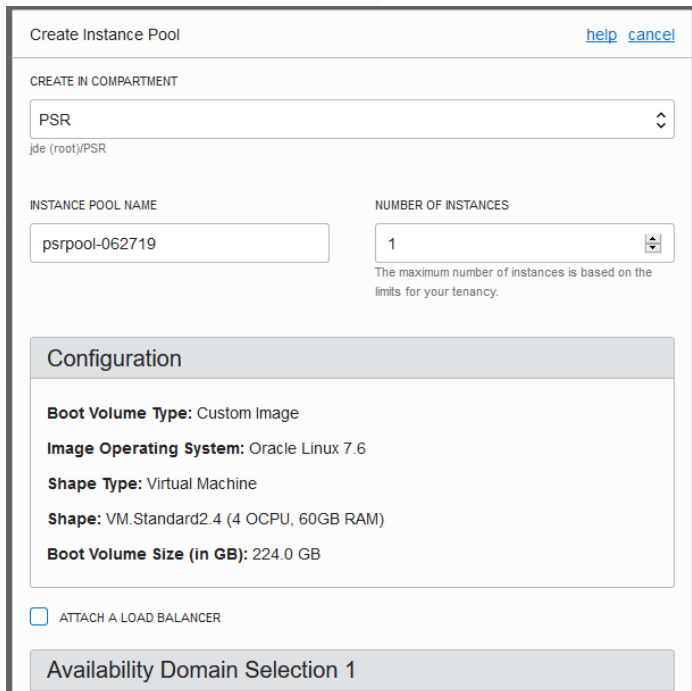
jde (root)/PSR

https://console.us-phoenix-1.oraclecloud.com/compute/instance-configs/ocid1.insta...onfiguration.oc1.phx.aaaaaaanc6pu6cf5qxtcqegj3yd3qcrs44vh77doiy5vagwhvbyysh4oq

- On the instance Configuration Details screen, click the **Create Instance Pool** button.

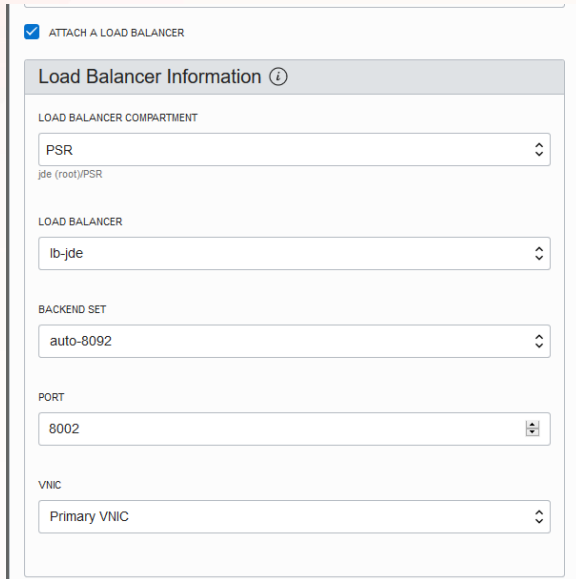


- In the Create Instance Pool form, select your compartment, enter a meaningful name for the pool, and select 1 for the initial number of instances in the pool. Click the **Attach a Load Balancer** check box to see the load balancer configuration fields and proceed to the next step.



- In the Load Balancer Information section, complete these fields:
 - Load Balancer Compartment:** Should be set to the same compartment in which you are working.
 - Load Balancer:** Use the drop-down list to select your load balancer.
 - Backend Set:** Select the back-end set you created for autoscaling.

Port: Enter the actual port on the back-end server that will be used to access the HTML server. This could be different from the port from which the listener will receive requests on the load balancer.



ATTACH A LOAD BALANCER

Load Balancer Information ⓘ

LOAD BALANCER COMPARTMENT
PSR
jde (root)/PSR

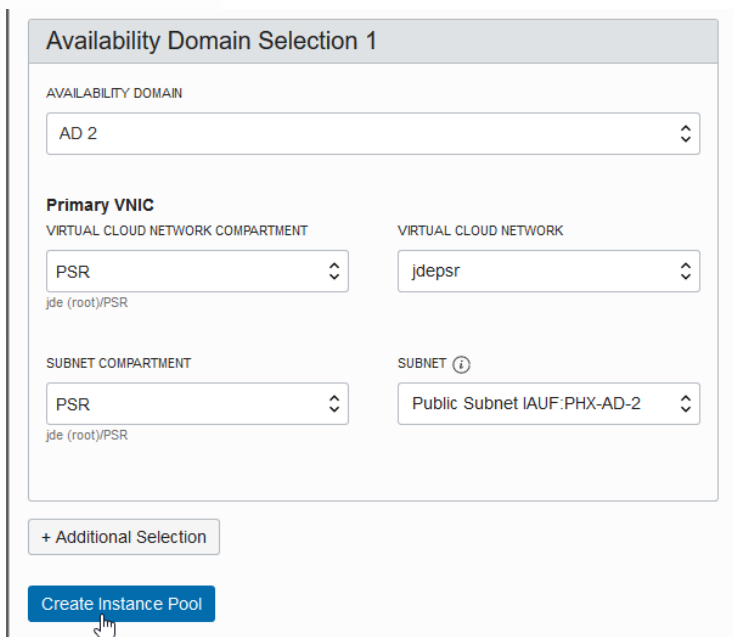
LOAD BALANCER
lb-jde

BACKEND SET
auto-8092

PORT
8002

VNIC
Primary VNIC

8. In the Availability Domain Selection 1 section, select your preferred Availability Domain, Virtual Cloud Network, Subnet, and the corresponding compartment for the applicable resources.
9. Confirm all the entered values and then click the **Create Instance Pool** button.



Availability Domain Selection 1

AVAILABILITY DOMAIN
AD 2

Primary VNIC

VIRTUAL CLOUD NETWORK COMPARTMENT
PSR
jde (root)/PSR

VIRTUAL CLOUD NETWORK
jdepsr

SUBNET COMPARTMENT
PSR
jde (root)/PSR

SUBNET ⓘ
Public Subnet IAUF:PHX-AD-2

+ Additional Selection

Create Instance Pool

10. The status is Provisioning because it creates the initial instance in the pool and attaches the instance as a backend component to the load balancer.

ORACLE Cloud us-phoenix-1

Compute » Instance Pools » Instance Pool Details » Work Requests

psrpool-062719

PROVISIONING

[Edit](#) [Start](#) [Stop](#) [Reboot](#) [Add Tags](#) [Actions](#)

[Instance Pool Information](#) [Tags](#)

OCID: ...7wffza [Show](#) [Copy](#) **Target Instance Count:** 1
Availability Domain: AD-2 **Instance Configuration:** [psrdocker5-062019](#)
Compartment: jde (root)/PSR **Created:** Thu, Jun 27, 2019, 20:19:04 UTC

Resources

- Created Instances (1)
- Load Balancers (1)
- Work Requests (2)**

Work Requests

Operation	Status	% Complete	Accepted	Started	Fin
Attach load balancer	In Progress	0	Thu, Jun 27, 2019, 20:21:35 UTC	Thu, Jun 27, 2019, 20:21:35 UTC	-
Create instances in pool	In Progress	80	Thu, Jun 27, 2019, 20:19:21 UTC	Thu, Jun 27, 2019, 20:19:21 UTC	-

Showing 2 items < Page 1 >

Note: Instances created in a pool are given an autogenerated unique host name based on the name of the pool.

ORACLE Cloud us-ph

Compute » Instance Pools » Instance Pool Details

psrpool-062719

RUNNING

[Edit](#) [Start](#) [Stop](#) [Reboot](#) [Add Tags](#) [Actions](#)

[Instance Pool Information](#) [Tags](#)

OCID: ...7wffza [Show](#) [Copy](#) **Target Instance Count:** 1
Availability Domain: AD-2 **Instance Configuration:** [psrdocker5-062019](#)
Compartment: jde (root)/PSR **Created:** Thu, Jun 27, 2019, 20:19:04 UTC

Resources

- Created Instances (1)**
- Load Balancers (1)
- Work Requests (2)

Created Instances

Name	Backend Health Status	Status	Availability Domain	Fault Domain	Instance Configuration
inst-qsd88-psrpool-062719	Unknown	Running	AD-2	FD-2	psrdocker5-062019

- From the Instance Pool Details form, select **Create Autoscaling Configuration** from the Actions drop-down menu.

The screenshot shows the Oracle Cloud console interface. At the top, the Oracle Cloud logo and navigation icons are visible. The breadcrumb trail indicates the current page is 'Compute > Instance Pools > Instance Pool Details'. The main content area displays the details for an instance pool named 'psrpool-062719'. A large green square with a white 'P' and the word 'RUNNING' below it indicates the pool's status. A toolbar contains buttons for 'Edit', 'Start', 'Stop', 'Reboot', 'Add Tags', and an 'Actions' dropdown menu. The 'Actions' menu is open, showing 'Create Autoscaling Configuration' and 'Terminate'. Below the toolbar, the 'Instance Pool Information' tab is active, showing details such as OCID, Availability Domain, Compartment, Target Instance Count, Instance Configuration, and Created date. At the bottom, there is a 'Resources' section with a link to 'Created Instances (1)' and a table with columns for 'Backend Health', 'Availability', 'Fault', and 'Instance'. The URL at the bottom of the browser window is partially visible.

- Enter or confirm the following values in the Create Autoscaling Configuration form.

Autoscaling Configuration Compartment: Your compartment should be populated by default.

Autoscaling Configuration Name: Keep the default unique name or enter a meaningful name of your choice.

Cooldown in Seconds: This field is populated with 300 seconds (five minutes) by default which is the minimum value allowed. The cool-down period is the amount of time between scaling actions or decisions as to scale out or scale in the number of instances.

Instance Pool: The fields in this section should automatically be populated with your compartment and instance pool name.

Create Autoscaling Configuration
[help](#) [cancel](#)

AUTOSCALING CONFIGURATION COMPARTMENT

PSR

jde (root)/PSR

AUTOSCALING CONFIGURATION NAME

psrautoscale-0627

COOLDOWN IN SECONDS ⓘ

300

The minimum value is 300 seconds, which is also the default value.

Instance Pool

INSTANCE POOL COMPARTMENT

PSR

jde (root)/PSR

INSTANCE POOL

psrpool-062719

i

Ensure that you [enable monitoring](#) to allow the instances in this instance pool to emit metrics. Autoscaling relies on these metrics to take autoscaling actions.

13. In the Autoscaling Policy section, complete these fields:

Autoscaling Policy Name: Keep the default unique name or provide a meaningful name of your choice.

Performance Metric: Select CPU or Memory Utilization.

For this testing, CPU was used.

Scaling Limits:

- Minimum Instances = 1
- Maximum Instances = 3
- Initial Instances = 1

Scaling Rule:

- Scale-out Operator = Greater than (>)
- Threshold Percentage=50
- Number of Instances to Add = 1,
- Scale-in Operator = Less than (<)
- Threshold Percentage = 1
- Number of Instances to Remove = 1

Autoscaling Policy

AUTOSCALING POLICY NAME

PERFORMANCE METRIC ⓘ

Scaling Limits

MINIMUM NUMBER OF INSTANCES	MAXIMUM NUMBER OF INSTANCES	INITIAL NUMBER OF INSTANCES
<input type="text" value="1"/>	<input type="text" value="3"/> <small>The maximum number of instances is based on the limits for your tenancy.</small>	<input type="text" value="1"/>

Scaling Rule

SCALE-OUT OPERATOR	THRESHOLD PERCENTAGE ⓘ	NUMBER OF INSTANCES TO ADD
<input type="text" value="Greater than (>)"/>	<input type="text" value="50"/>	<input type="text" value="1"/>
SCALE-IN OPERATOR	THRESHOLD PERCENTAGE ⓘ	NUMBER OF INSTANCES TO REMOVE
<input type="text" value="Less than (<)"/>	<input type="text" value="1"/>	<input type="text" value="1"/>

14. Optionally, fill out the Tags section and click the **Create** button.

Preparing the Load Balancer for Autoscaling

Assumptions and Prerequisites:

The load balancer has been created and configured for standard EnterpriseOne access.

1. On the Load Balancer Details screen, click the **Create Backend Set** button to create a new backend set. This is the back-end set that is associated with the autoscaling pool.

ORACLE Cloud

Networking » Load Balancers » Load Balancer Details » Backend Sets

lb-jde

Terminate Add Tag(s)

Load Balancer Information Tags

Load Balancer Information

OCID: ...6h65kq [Show](#) [Copy](#)

Created: Tue, 28 May 2019 09:23:15 GMT

Shape: 100Mbps

IP Address: 129.146.149.151 (Public)

Virtual Cloud Network: [jlepsi](#)

Subnet (1 of 2): [Public Subnet IAUF-PHX-AD-3](#)

Subnet (2 of 2): [Public Subnet IAUF-PHX-AD-2](#)

Traffic between this load balancer and its backend servers is subject to the governing security lists.

[Learn more about load balancers and security lists.](#)

Overall Health

OK

Backend Sets Health

0 Critical

0 Warning

1 Unknown

2 OK

Resources

Metrics

Backend Sets (3)

Path Route Sets (0)

Rule Sets (0)

Listeners (3)

Backend Sets

Create Backend Set

Name	Traffic Distribution Policy	Number of Backends	Health
8003	Weighted Round Robin	2	OK
8082	Weighted Round Robin	1	OK

Terms of Use and Privacy Cookie Preferences

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

2. Enter the information to configure the back-end set.

Name: Provide a meaningful name.

Traffic Distribution Policy: Select your preferred policy.

Use SSL: Check if the back-end server is going to be accessed through HTTPS and then provide certificate information (this should be already uploaded to the load balancer if SSL has been implemented).

Use Session Persistence: Check whether the sessions are sticky and are on the same backend set.

Cookie Name: Select a specific cookie for session persistence or enter * to match any cookie name.

Health Check:

- Protocol = TCP
- Port=Port

This is the port that is used for the backend access.

- Number of retries=2

Leave the other fields blank and they will be populated with the default values.

Create Backend Set [help](#) [cancel](#)

Specify a set of policies that define how the load balancer routes ingress traffic to your backend servers.

NAME

TRAFFIC DISTRIBUTION POLICY ⓘ

USE SSL
 USE SESSION PERSISTENCE

COOKIE NAME ⓘ

Specify "*" to match any cookie name.

DISABLE FALLBACK
Disable fallback to other servers when the original server is unavailable.

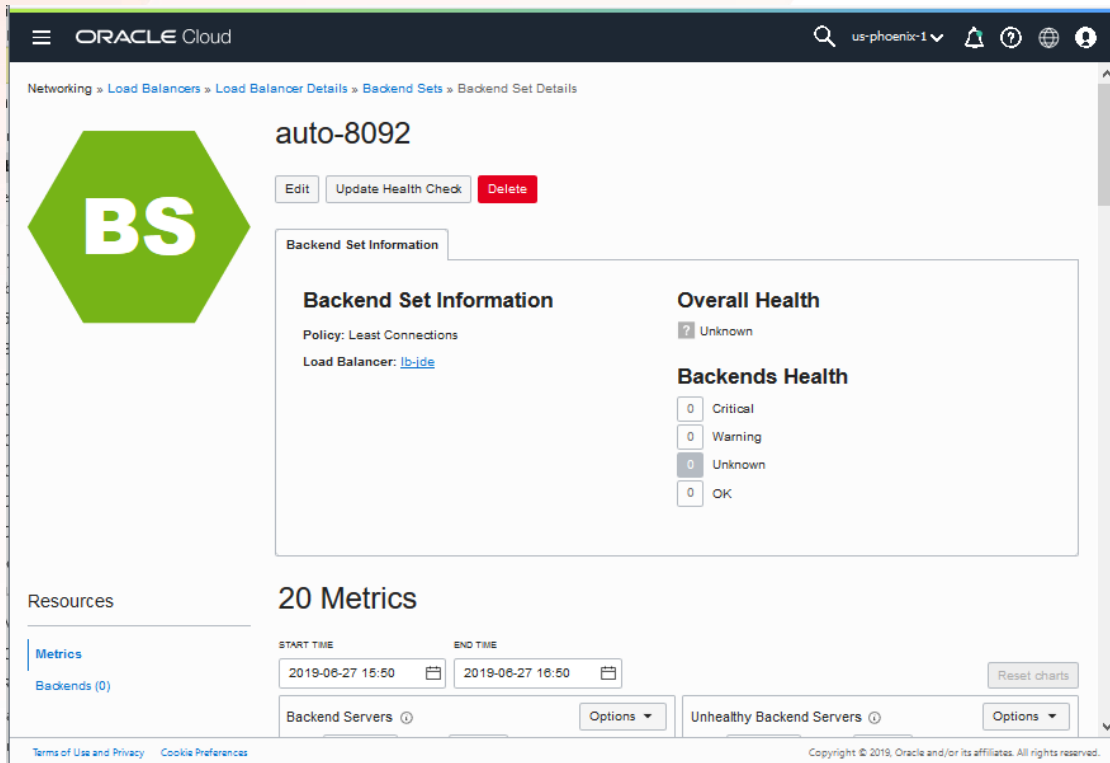
Health Check

Define the health check policy the load balancer uses to confirm the health of your backend servers.

PROTOCOL PORT OPTIONAL

INTERVAL IN MS OPTIONAL TIMEOUT IN MS OPTIONAL NUMBER OF RETRIES OPTIONAL

- When the work request has been completed, open the backend set and view the details. Notice that no backends are associated to the backend set at this time.



4. Return to the load balancer details screen and click **Listeners** under Resources, and then click the **Create Listener** button.

The screenshot shows the Oracle Cloud console interface for a Load Balancer named 'lb-jde'. The page is titled 'Networking > Load Balancers > Load Balancer Details > Listeners'. On the left, there is a green hexagonal icon with 'LB' and the word 'ACTIVE' below it. Below the icon is a 'Resources' sidebar with links for Metrics, Backend Sets (3), Path Route Sets (0), Rule Sets (0), and Listeners (3). The main content area is divided into two sections: 'Load Balancer Information' and 'Overall Health'. The 'Load Balancer Information' section includes fields for OCID, creation time, shape, IP address, Virtual Cloud Network, and subnets. The 'Overall Health' section shows a green checkmark and 'OK'. Below this is the 'Backend Sets Health' section, which shows a summary of health status: 0 Critical, 0 Warning, 1 Unknown, and 2 OK. At the bottom, there is a 'Listeners' table with a 'Create Listener' button above it. The table has columns for Name, Protocol, Port, Backend Set, Path Route Set, Hostnames, and Use SSL. Two listeners are listed: one with Name '8082', Protocol 'HTTP', Port '8082', Backend Set '8082', Path Route Set, Hostnames, and Use SSL 'No'; and another with Name 'lb-20190528-1449-listener', Protocol 'HTTP', Port '8002', Backend Set '8003', Path Route Set, Hostnames 'jdedemo.au.oraclecloud.com', and Use SSL 'No'.

- To create a new listener port and associate it with the new backend set created in the previous steps, complete these fields:

Name: Provide a meaningful name.

HOSTNAMES: Select the hostnames that are associated with your public SSL certificate (if applicable).

Protocol: Select the protocol used to access this port (HTTP in this case).

Port: Enter the front-end port for the listener to receive requests on the load balancer.

Use SSL: Check if the listener port can be accessed using SSL and then provide certificate information (this should be already uploaded to the load balancer if the SSL has been implemented).

Backend Set: Select the back-end set created previously to route requests from this listener.

Idle Timeout in Seconds: Leave this field blank so that it is automatically populated with the default value of 60 seconds.

Create Listener [help](#) [cancel](#)

To allow your load balancer to accept ingress traffic, specify the protocol and port for your public IP address.

NAME

HOSTNAMES *OPTIONAL*

PROTOCOL PORT USE SSL

BACKEND SET

IDLE TIMEOUT IN SECONDS *OPTIONAL*

The default timeout for HTTP is 60 seconds.

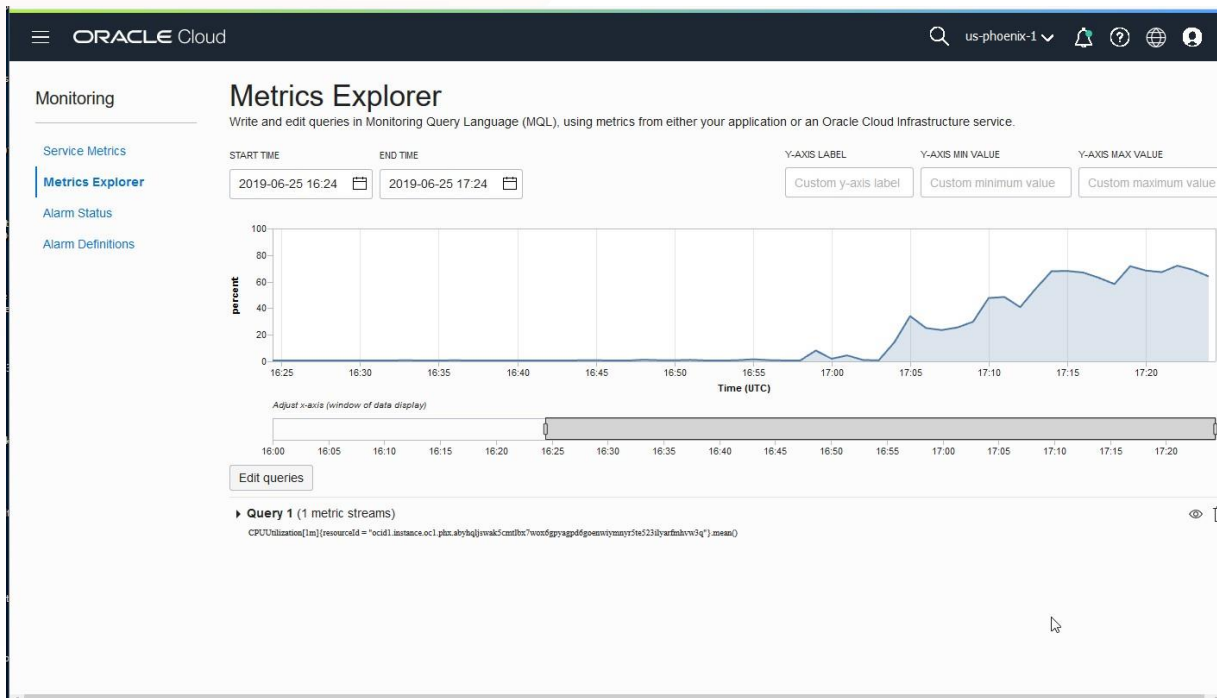
There are no path route sets for this load balancer. [Create a path route set.](#)

AUTOSCALING TEST SCREENSHOTS

The following screens were captured during a test of the autoscaling utility using JD Edwards EnterpriseOne elastic units.

Testing began with an instance pool that can have a minimum of one instance and a maximum of three instances, and has an autoscaling configuration for triggering a scaling event when a threshold of 50% CPU is reached. There is a five-minute cooling period between scaling actions before the threshold is reevaluated on the instances.

In the image below, you can see the CPU utilization of the single-pool instance as it reached the specified threshold.



After the CPU threshold was maintained for the defined period, a scaling event was triggered and a new instance was created in the pool. The details of the new instance are shown in the screenshot below.

Information
If the instance pool has been in the scaling state for an extended period of time it may be because the number of instances requested has exceeded your tenancy's service limits for that shape and availability domain. For more information, see [Service Limits](#). You can check your tenancy's service limits [here](#).

psrpool-062119

Edit Start Stop Reboot Add Tags Actions

Instance Pool Information Tags

OCID: ...ajbhkq Show Copy
Availability Domain: AD-2
Compartment: jde (root)/PSR

Target Instance Count: 2
Instance Configuration: [psrdocker5-062019](#)
Autoscaling Configuration: [psrdocker-autoscale-062119](#)
Created: Fri, Jun 21, 2019, 18:02:29 UTC

Resources

- Created Instances (2)
- Load Balancers (1)
- Work Requests (1)

Created Instances

Name	Backend Health Status	Status	Availability Domain	Fault Domain	Instance Configuration	Created
inst-lbcsy-psrpool-062119	Ok	Running	AD-2	FD-3	psrdocker5-062019	Fri, Jun 21, 2019, 21:31:28 UTC
inst-p0l7o-psrpool-062119	Unavailable	Provisioning	AD-2	FD-1	psrdocker5-062019	Tue, Jun 25, 2019, 17:19:36 UTC

In the following screenshot, the individual work requests to create the additional instance and to attach the load balancer are displayed as successful.

Instance Pool Information Tags

OCID: ...ajbhkq Show Copy
Availability Domain: AD-2
Compartment: jde (root)/PSR

Target Instance Count: 2
Instance Configuration: [psrdocker5-062019](#)
Autoscaling Configuration: [psrdocker-autoscale-062119](#)
Created: Fri, Jun 21, 2019, 18:02:29 UTC

Resources

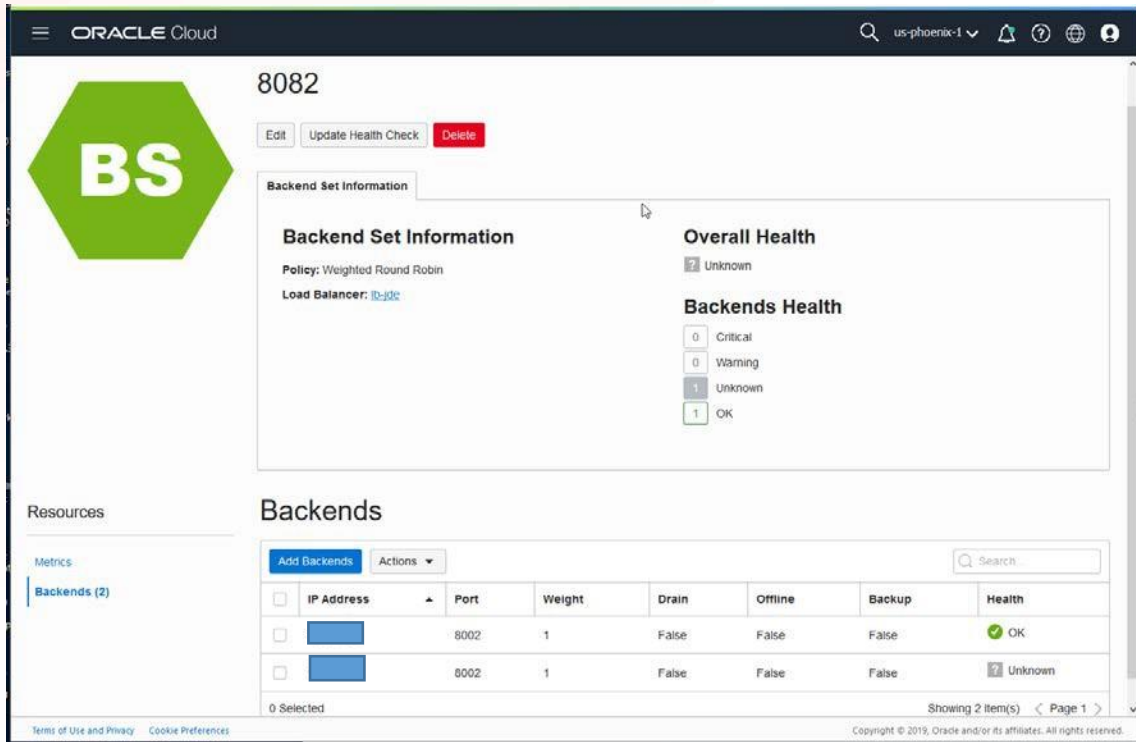
- Created Instances (2)
- Load Balancers (1)
- Work Requests (2)

Work Requests

Operation	Status	% Complete	Accepted	Started	Finished
Attach load balancer	Succeeded	100	Tue, Jun 25, 2019, 17:21:21 UTC	Tue, Jun 25, 2019, 17:21:21 UTC	Tue, Jun 25, 2019, 17:22:06 UTC
Create instances in pool	Succeeded	100	Tue, Jun 25, 2019, 17:19:19 UTC	Tue, Jun 25, 2019, 17:19:19 UTC	Tue, Jun 25, 2019, 17:22:15 UTC

Showing 2 Items < Page 1 >

This screenshot shows that a new instance was added to the backend definition of the load balancer. When the health check of the new backend instance is reported as successful, inbound traffic is routed to the new instance.



After the cool-down period expires, the increasing load triggers an additional scaling event. This screenshot shows that the resulting third instance has been added to the pool and the load balancer.

ORACLE Cloud us-phoenix-1

Compute » Instance Pools » Instance Pool Details » Work Requests

psrpool-062119

P RUNNING

Edit Start Stop Reboot Add Tags Actions

Instance Pool Information Tags

OCID: [ojbhkq](#) Show Copy
 Availability Domain: AD-2
 Compartment: jde (root)PSR

Target Instance Count: 3
 Instance Configuration: [psrdocker5-062019](#)
 Autoscaling Configuration: [psrdocker-autoscale-062119](#)
 Created: Fri, Jun 21, 2019, 18:02:29 UTC

Resources

- Created Instances (3)
- Load Balancers (1)
- Work Requests (4)**

Work Requests

Operation	Status	% Complete	Accepted	Started	Finished
Attach load balancer	Succeeded	100	Tue, Jun 25, 2019, 17:30:40 UTC	Tue, Jun 25, 2019, 17:30:40 UTC	Tue, Jun 25, 2019, 17:31:29 UTC
Create instances in pool	Succeeded	100	Tue, Jun 25, 2019, 17:28:25 UTC	Tue, Jun 25, 2019, 17:28:25 UTC	Tue, Jun 25, 2019, 17:31:53 UTC
Attach load balancer	Succeeded	100	Tue, Jun 25, 2019, 17:21:21 UTC	Tue, Jun 25, 2019, 17:21:21 UTC	Tue, Jun 25, 2019, 17:22:06 UTC
Create instances in pool	Succeeded	100	Tue, Jun 25, 2019, 17:19:19 UTC	Tue, Jun 25, 2019, 17:19:19 UTC	Tue, Jun 25, 2019, 17:22:15 UTC

Showing 4 items < Page 1 >

Terms of Use and Privacy Cookie Preferences Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The following screenshot shows that the third instance has been added to the backend set.

ORACLE Cloud us-phoenix-1

8082

BS

Edit Update Health Check Delete

Backend Set Information

Policy: Weighted Round Robin
 Load Balancer: [lb-8082](#)

Overall Health
 Warning

Backends Health

- 0 Critical
- 0 Warning
- 1 Unknown
- 2 OK

Resources

- Metrics
- Backends (3)**

Backends

	IP Address	Port	Weight	Drain	Offline	Backup	Health
<input type="checkbox"/>	[redacted]	8082	1	False	False	False	OK
<input type="checkbox"/>	[redacted]	8082	1	False	False	False	OK
<input type="checkbox"/>	[redacted]	8082	1	False	False	False	Unknown

Terms of Use and Privacy Cookie Preferences Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The following screenshot shows the resulting pool of three instances which are all functional and are processing requests from the load balancer.

psrpool-062119

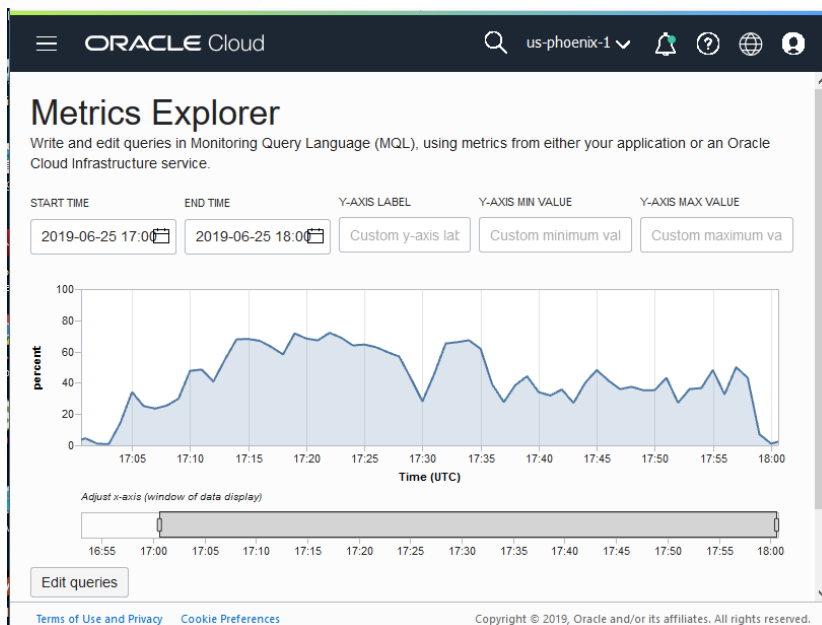
OCID: [...apbtkq](#) Show Copy
 Availability Domain: AD-2
 Compartment: [jde \(root\)PSR](#)

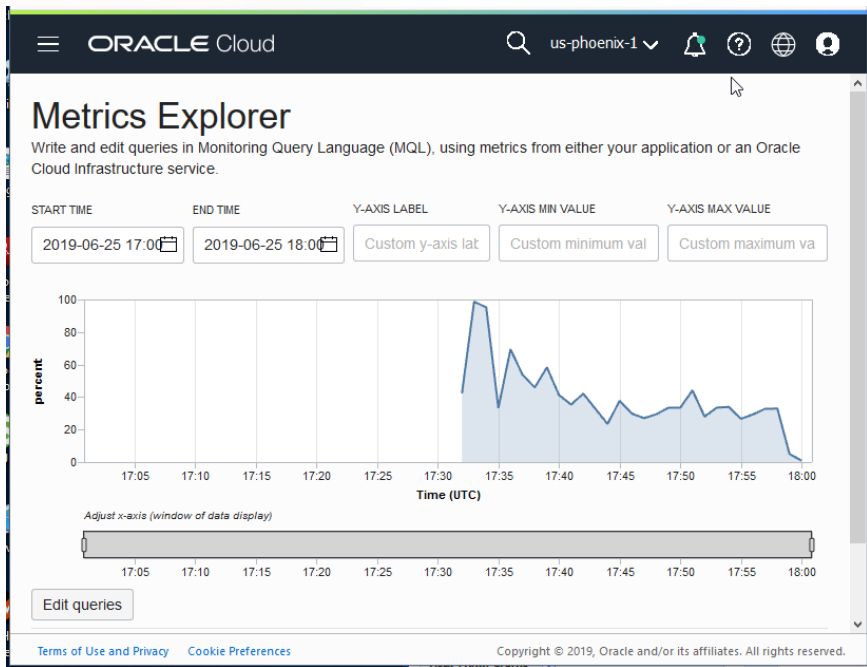
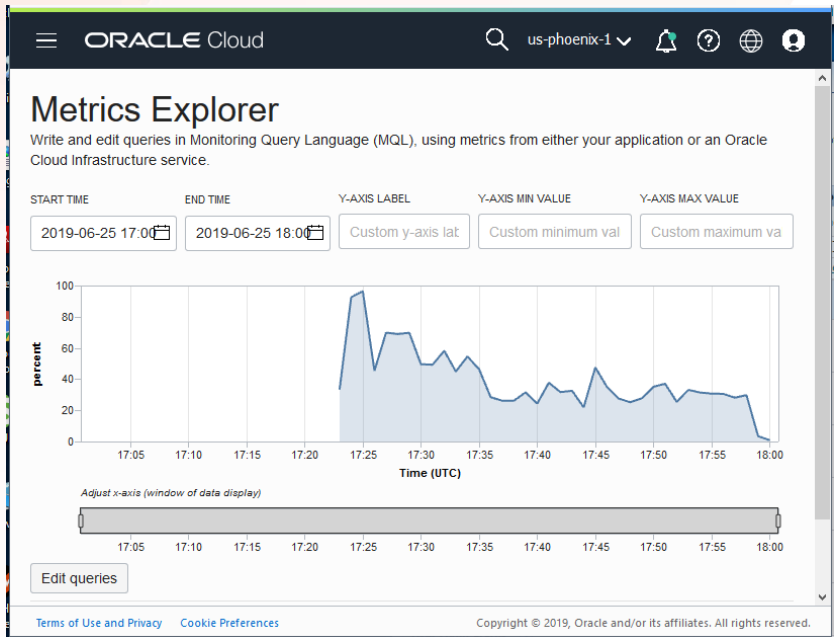
Target Instance Count: 3
 Instance Configuration: [psrdocker5-062019](#)
 Autoscaling Configuration: [psrdocker.autoscale-062119](#)
 Created: Fri, Jun 21, 2019, 18:02:29 UTC

Name	Backend Health Status	Status	Availability Domain	Fault Domain	Instance Configuration	Created
inst-grnc1-psrpool-062119	✔ OK	Running	AD-2	FD-2	psrdocker5-062019	Tue, Jun 25, 2019, 17:28:40 UTC
inst-ibvty-psrpool-062119	✔ OK	Running	AD-2	FD-3	psrdocker5-062019	Fri, Jun 21, 2019, 21:31:28 UTC
inst-p0i2o-psrpool-062119	✔ OK	Running	AD-2	FD-1	psrdocker5-062019	Tue, Jun 25, 2019, 17:19:36 UTC

Showing 3 items

The following screenshots show the CPU metrics from each of the instances in the pool during the load test timeline.





When the CPU consumption in an instance is less than the minimum CPU threshold (less than 1%), a scaling event is triggered and the instance is terminated from the pool and is removed from the back end of the load balancer. The following screenshots show the work requests that are in progress for the scale-down event.

ORACLE Cloud us-phenix-1

Information

If the instance pool has been in the scaling state for an extended period of time it may be because the number of instances requested has exceeded your tenancy's service limits for that shape and availability domain. For more information, see [Service Limits](#). You can check your tenancy's service limits [here](#).

psrpool-062119

SCALING

Edit Start Stop Reboot Add Tags Actions

Instance Pool Information Tags

OCID: ...ajbhkq Show Copy **Target Instance Count:** 2
 Availability Domain: AD-2 **Instance Configuration:** psrpool-062119
 Compartment: jde (root)/PSR **Autoscaling Configuration:** psrpool-autoscale-062119
Created: Fri, Jun 21, 2019, 18:02:29 UTC

Resources

- Created instances (3)
- Load Balancers (1)
- Work Requests (6)

Work Requests

Operation	Status	% Complete	Accepted	Started	Finished
<a>Detach load balancer	In Progress	0	Tue, Jun 25, 2019, 18:09:44 UTC	Tue, Jun 25, 2019, 18:09:44 UTC	-
<a>Terminate instances in pool	In Progress	20	Tue, Jun 25, 2019, 18:09:19 UTC	Tue, Jun 25, 2019, 18:09:19 UTC	-
<a>Attach load balancer	Succeeded	100	Tue, Jun 25, 2019, 17:30:40 UTC	Tue, Jun 25, 2019, 17:30:40 UTC	Tue, Jun 25, 2019, 17:31:29 UTC
<a>Create instances in pool	Succeeded	100	Tue, Jun 25, 2019, 17:28:25 UTC	Tue, Jun 25, 2019, 17:28:25 UTC	Tue, Jun 25, 2019, 17:31:53 UTC

ORACLE Cloud us-phenix-1

Compute > Instance Pools > Instance Pool Details > Work Requests > Work Requests Details

WR

IN PROGRESS

Terminate instances in pool

Work Requests Information

20% Complete

OCID: ...4o4dp Show Copy **Accepted:** Tue, Jun 25, 2019, 18:09:19 UTC
 Compartment: jde (root)/PSR **Started:** Tue, Jun 25, 2019, 18:09:19 UTC
Finished: -

Resources

- Log Messages (3)
- Error Messages (0)
- Associated Resources (1)

Log Messages

Message	Timestamp
Deleting 1 load balancer backends	Tue, Jun 25, 2019, 18:09:42 UTC
Selecting instance ocid1.instance.oc1.phx.ab7h9jx7b5eyk4r4lyvdgo7a4pgxmnevz526k3rpkemghiyoi7zh7q to terminate	Tue, Jun 25, 2019, 18:09:29 UTC
Terminating 1 instances from instance pool ocid1.instancepool.oc1.phx.aaaaaaaapna4ief56fwy5i7trpee56vie4b0hbojlm4dzbfafw4ugajbnkq	Tue, Jun 25, 2019, 18:09:19 UTC

Showing 3 item(s) < Page 1 >

Terms of Use and Privacy Cookie Preferences Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

psrpool-062119

Instance Pool Information

- OCID: apbhkq
- Availability Domain: AD-2
- Compartment: jde (root)PSR
- Target Instance Count: 2
- Instance Configuration: psrdocker5-062019
- Autoscaling Configuration: psrdocker-autoscale-062119
- Created: Fri, Jun 21, 2019, 18:02:29 UTC

Created Instances

Name	Backend Health Status	Status	Availability Domain	Fault Domain	Instance Configuration	Created
inst-pdnc1-psrpool-062119	Ok	Running	AD-2	FD-2	psrdocker5-062019	Tue, Jun 25, 2019, 17:28:40 UTC
inst-libvay-psrpool-062119	Ok	Running	AD-2	FD-3	psrdocker5-062019	Fri, Jun 21, 2019, 21:31:28 UTC
inst-pd2o-psrpool-062119	Unavailable	Terminating	AD-2	FD-1	psrdocker5-062019	Tue, Jun 25, 2019, 17:19:36 UTC

After another cool-down period, another instance in which CPU consumption was lower than the threshold value was terminated resulting in a single-instance pool with a single back end on the load balancer. The following screenshots show the progress of the work requests for the scale-down event and the resulting pool of a single back end server.

psrpool-062119

Instance Pool Information

- OCID: apbhkq
- Availability Domain: AD-2
- Compartment: jde (root)PSR
- Target Instance Count: 1
- Instance Configuration: psrdocker5-062019
- Autoscaling Configuration: psrdocker-autoscale-062119
- Created: Fri, Jun 21, 2019, 18:02:29 UTC

Work Requests

Operation	Status	% Complete	Accepted	Started	Finished
Detach load balancer	In Progress	0	Tue, Jun 25, 2019, 18:25:56 UTC	Tue, Jun 25, 2019, 18:25:56 UTC	-
Terminate instances in pool	In Progress	20	Tue, Jun 25, 2019, 18:25:24 UTC	Tue, Jun 25, 2019, 18:25:25 UTC	-
Detach load balancer	Succeeded	100	Tue, Jun 25, 2019, 18:09:44 UTC	Tue, Jun 25, 2019, 18:09:44 UTC	Tue, Jun 25, 2019, 18:13:20 UTC
Terminate instances in pool	Succeeded	100	Tue, Jun 25, 2019, 18:09:19 UTC	Tue, Jun 25, 2019, 18:09:19 UTC	Tue, Jun 25, 2019, 18:15:04 UTC
Attach load balancer	Succeeded	100	Tue, Jun 25, 2019, 17:30:40 UTC	Tue, Jun 25, 2019, 17:30:40 UTC	Tue, Jun 25, 2019, 17:31:29 UTC

ORACLE Cloud us-phoenix-1

Compute » Instance Pools » Instance Pool Details

psrpool-062119

P RUNNING

Edit Start Stop Reboot Add Tags Actions

Instance Pool Information Tags

OCID: [ajphkq](#) Show Copy
 Availability Domain: AD-2
 Compartment: jde (root)PSR

Target Instance Count: 1
 Instance Configuration: [psrlocker5-062019](#)
 Autoscaling Configuration: [psrlocker-autoscale-062119](#)
 Created: Fri, Jun 21, 2019, 18:02:29 UTC

Resources

- Created Instances (2)
- Load Balancers (1)
- Work Requests (8)

Created Instances

Name	Backend Health Status	Status	Availability Domain	Fault Domain	Instance Configuration	Created
inst-pcncj-psrpool-062119	OK	Running	AD-2	FD-2	psrlocker5-062019	Tue, Jun 25, 2019, 17:28:40 UTC
inst-tbvsy-psrpool-062119	Unavailable	Terminated	AD-2	FD-3	psrlocker5-062019	Fri, Jun 21, 2019, 21:31:28 UTC

Showing 2 items

Terms of Use and Privacy Cookie Preferences Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

ORACLE Cloud us-phoenix-1

Networking » Load Balancers » Load Balancer Details » Backend Sets » Backend Set Details » Backends

8082

BS

Edit Update Health Check Delete

Backend Set Information

Policy: Weighted Round Robin
 Load Balancer: [lb-jde](#)

Overall Health
 OK

Backends Health

- 0 Critical
- 0 Warning
- 0 Unknown
- 1 OK

Resources

- Metrics
- Backends (1)

Backends

Add Backends Actions Search...

<input type="checkbox"/>	IP Address	Port	Weight	Drain	Offline	Backup	Health
<input type="checkbox"/>	 	8082	1	False	False	False	OK

0 Selected Showing 1 Item(s) < Page 1 >

Terms of Use and Privacy Cookie Preferences Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The screenshot shows the Oracle Cloud console for an Instance Pool named 'psrpool-062119'. The pool is in a 'RUNNING' state. Below the pool information, there is a 'Work Requests' table with the following data:

Operation	Status	% Complete	Accepted	Started	Finished
Detach load balancer	Succeeded	100	Tue, Jun 25, 2019, 18:25:56 UTC	Tue, Jun 25, 2019, 18:25:56 UTC	Tue, Jun 25, 2019, 18:29:32 UTC
Terminate instances in pool	Succeeded	100	Tue, Jun 25, 2019, 18:25:24 UTC	Tue, Jun 25, 2019, 18:25:25 UTC	Tue, Jun 25, 2019, 18:31:35 UTC
Detach load balancer	Succeeded	100	Tue, Jun 25, 2019, 18:09:44 UTC	Tue, Jun 25, 2019, 18:09:44 UTC	Tue, Jun 25, 2019, 18:13:20 UTC
Terminate instances in pool	Succeeded	100	Tue, Jun 25, 2019, 18:09:19 UTC	Tue, Jun 25, 2019, 18:09:19 UTC	Tue, Jun 25, 2019, 18:15:04 UTC
Attach load balancer	Succeeded	100	Tue, Jun 25, 2019, 17:30:40 UTC	Tue, Jun 25, 2019, 17:30:40 UTC	Tue, Jun 25, 2019, 17:31:29 UTC
Create instances in pool	Succeeded	100	Tue, Jun 25, 2019, 17:28:25 UTC	Tue, Jun 25, 2019, 17:28:25 UTC	Tue, Jun 25, 2019, 17:31:53 UTC
Attach load balancer	Succeeded	100	Tue, Jun 25, 2019, 17:21:21 UTC	Tue, Jun 25, 2019, 17:21:21 UTC	Tue, Jun 25, 2019, 17:22:06 UTC
Create instances in pool	Succeeded	100	Tue, Jun 25, 2019, 17:19:19 UTC	Tue, Jun 25, 2019, 17:19:19 UTC	Tue, Jun 25, 2019, 17:22:15 UTC

Following the scaling test, the instances from the pool were still visible as terminated instances for a short time in the Oracle Cloud Infrastructure console. After a scale-down event, the metrics can be viewed and captured in the duration when the terminated images are visible. The terminated images are shown in the following screenshot:

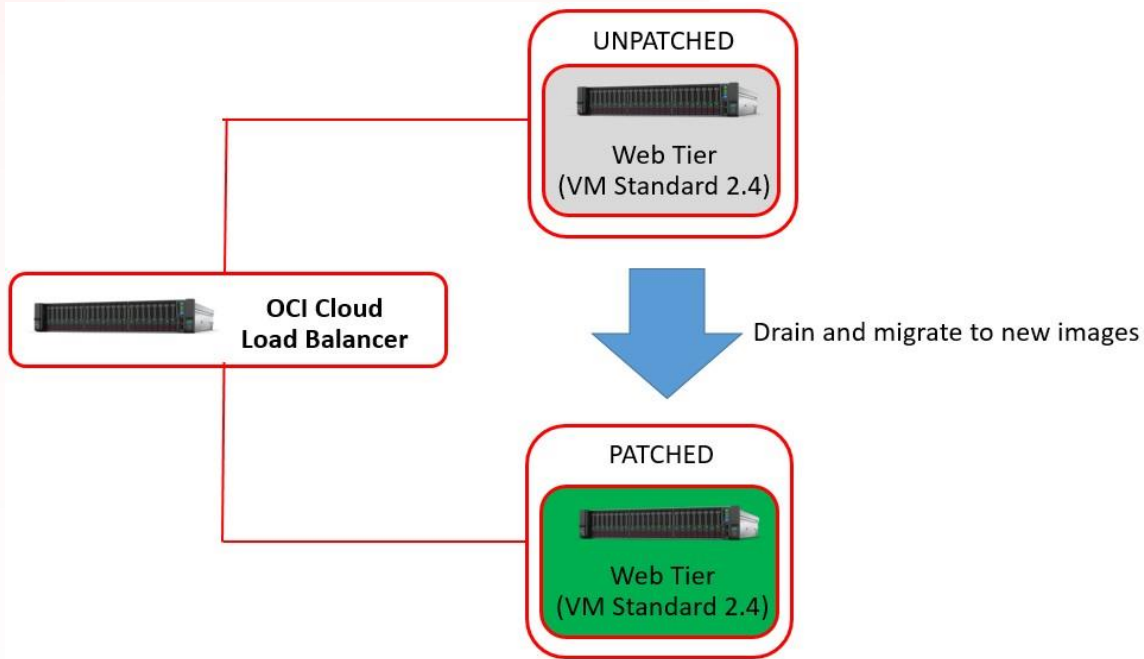
Instances in PSR Compartment

The screenshot shows a list of instances in the PSR compartment. The instances are sorted by 'Created Date (Desc)'. The list includes:

- Instance 1:** `inst-p02c-psrpool-062119`, Shape: VM.Standard2.4, Region: phx, Availability Domain: [redacted], Fault Domain: FAULT-DOMAIN-2, Created: Tue, 25 Jun 2019 17:28:39 GMT, Status: RUNNING.
- Instance 2:** `inst-p02c-psrpool-062119`, Shape: VM.Standard2.4, Region: phx, Availability Domain: [redacted], Fault Domain: FAULT-DOMAIN-1, Created: Tue, 25 Jun 2019 17:19:35 GMT, Status: TERMINATED.
- Instance 3:** `psrpool2`, Shape: VM.Standard2.4, Region: phx, Availability Domain: [redacted], Fault Domain: FAULT-DOMAIN-2, Created: Mon, 24 Jun 2019 18:50:42 GMT, Status: RUNNING.
- Instance 4:** `inst-lbvy-psrpool-062119`, Shape: VM.Standard2.4, Region: phx, Availability Domain: [redacted], Fault Domain: FAULT-DOMAIN-3, Created: Fri, 21 Jun 2019 21:31:27 GMT, Status: TERMINATED.

APPENDIX C: CONFIGURING ORACLE CLOUD INFRASTRUCTURE AUTOSCALING UTILITY FOR PATCHING

This section of the document describes a methodology for patching the JD Edwards EnterpriseOne architecture on the Oracle Cloud Infrastructure. A simplified example of a web tier patching process is illustrated in the figure below:



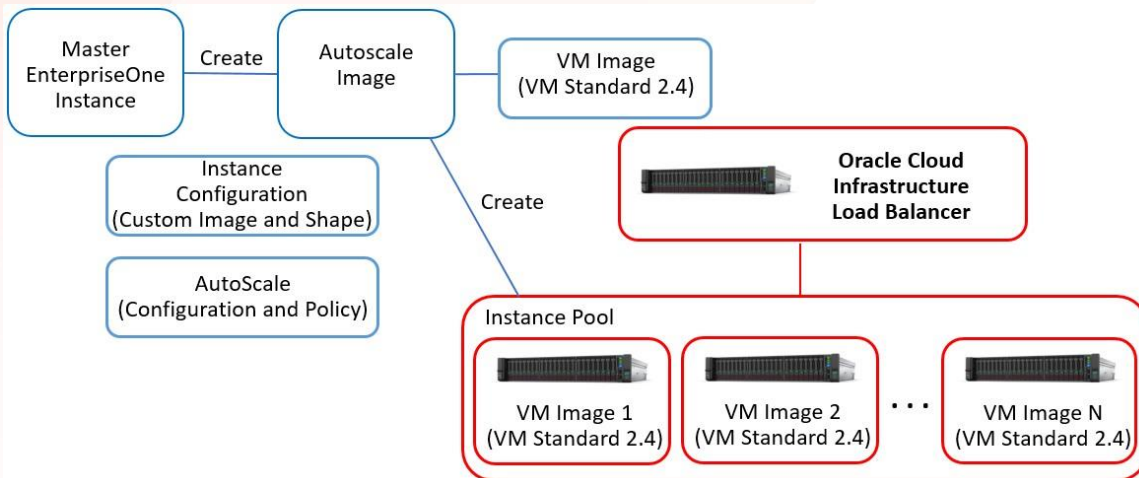
In the above figure, a web tier is shown which has an older version of software or an older security patch. A new patched web tier VM image is introduced into the architecture and the drain feature of the load balancer is initiated to move any new connections from the original unpatched web tier component to the patched web tier component.

The unpatched web tier component can then be taken offline and deprecated. The advantage of this model is that it includes a “back out” process to this methodology. In addition to the forward process of moving users from an unpatched web tier to a patched web tier, the reverse process can also be used to back out users from a patched version back to an unpatched version.

A back out plan is an important contingency plan of any patching methodology. A back out plan is critical to ensure that an alternative solution can be used if the new system or patched environment does not deliver services as expected or does not pass the post-implementation test.

Applying a Simple Model to an Instance Pool

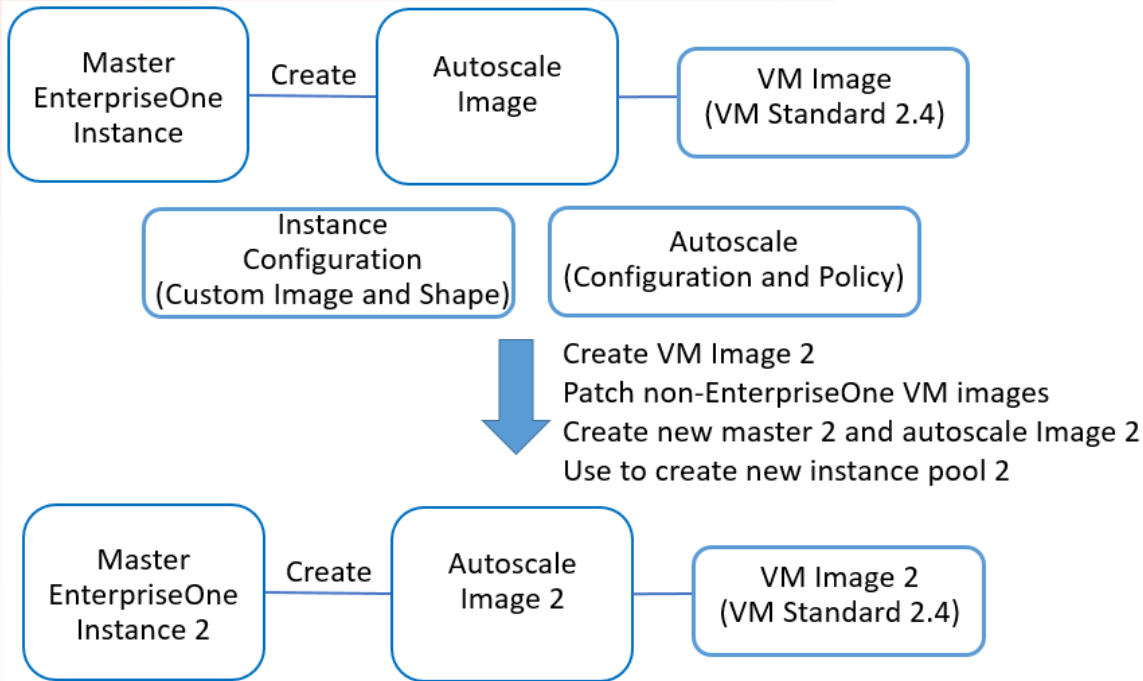
The diagram below illustrates the components that are involved in Oracle Cloud Infrastructure for applying a patch to an instance pool using the patching methodology explained above.



An instance pool is a requirement for autoscaling; it enables you to provision and create multiple compute instances based on the same autoscale image within the same region. VM Standard 2.4 refers to the shape of the Oracle Cloud VM image. A shape in Oracle Cloud has a unique set of processing and memory allocations as well as disk I/O configurations.

When you create the first instance pool, you must first define the master instance, autoscale image, and VM image shape. The master instance and autoscale images are templates that the instance pool configuration uses to create the VM images in the instance pool. You must also define a configuration and policy for the autoscaling utility of the Oracle Cloud Infrastructure.

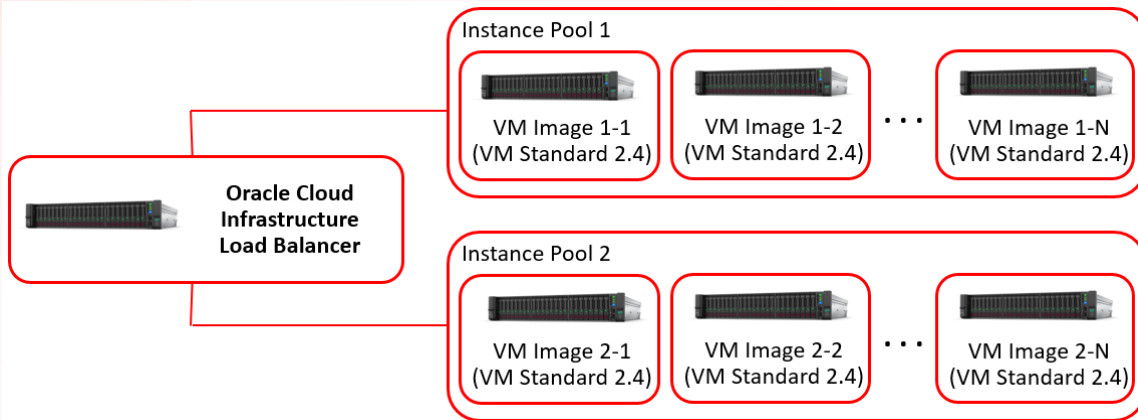
The autoscale configuration includes the minimum and maximum number of VM shapes in the instance pool. The autoscale policy defines the CPU and memory thresholds that the Oracle Cloud Infrastructure autoscaling utility uses to initiate a new VM image in the instance pool or deprecate and possibly remove a VM image from the instance pool. The next figure will describe applying the simplified model described above for patching purposes.



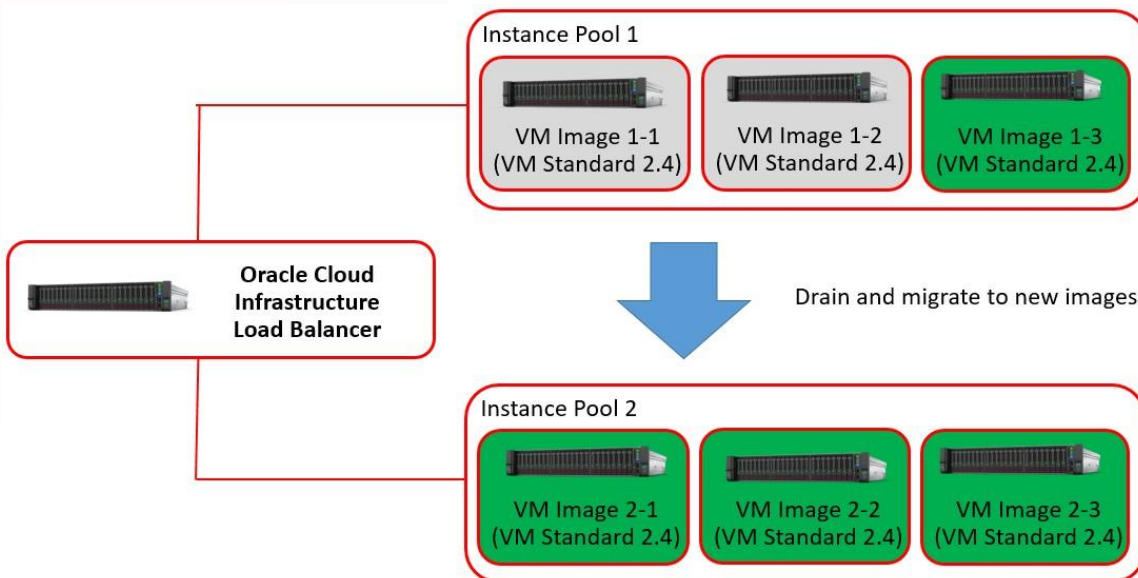
Patching non-EnterpriseOne VM image uses the same components of the simplified model. The process of patching involves taking the VM image (VM Standard 2.4) created from the original master EnterpriseOne image using the instance configuration and autoscale configuration and policy, patching it, and then the patched VM image is used to create a new master EnterpriseOne instance 2.

From the master EnterpriseOne instance 2, the autoscale image 2 is initiated and this new patched version of the image is what is propagated to the larger EnterpriseOne architecture implementation. In essence, only a single patched VM image is required and using the Oracle Cloud autoscale utility, the new patched software on the new VM image (VM Image 2 in the above diagram) is what is implemented over time to the entire Cloud architecture.

More than one set of these components can be created using the same or a unique instance configuration. In the below diagram Instance Pool 1 could be a patched version of the operating system, say recent security fixes, that supports the database. The instance pool can be a patched version of the operating system that updates the version of other third party software.



With the master instance 2, autoscale image 2, and VM image 2 defined, a new instance pool (instance pool 2) is created and attached to the same backend set of the Oracle Cloud Infrastructure Load Balancer.



The load balancer now can manage each of the VM images within each of the instance pools (instance pool 1 and instance pool 2) for patching. In the above diagram, three VM images exist in each instance pool.

Two of the VM images in instance pool 1 in the above diagram have been manually configured to “drain” (VM image 1-1 and VM image 1-2, which are grayed out). As a result of this configuration, the processes of these two VM images are moved so that they can be taken offline for patching. This means that all new requests to JD Edwards EnterpriseOne will no longer go to these VM images, but to the three VM images in instance pool 2 and the remaining “undrained” instance (VM image 1-3) in instance pool 1 that are still in an active state.

The marking of VM images to be in a “drain” state is still a manual process for the cloud administrator and it may take time to move all the processes off the VM images that have to be patched.

The advantages of this patching methodology are:

- You only patch the *initial* VM image defined by the master instance, autoscale image, and VM shape.
- The autoscaling utility handles the creation of the load balancer and updates the component with the configuration of each of the active VM shapes within the instance pool.

ORACLE CORPORATION

Worldwide Headquarters

500 Oracle Parkway, Redwood Shores, CA 94065 USA

Worldwide Inquiries

TELE + 1.650.506.7000 + 1.800.ORACLE1

FAX + 1.650.506.7200

oracle.com

CONNECT WITH US

Call +1.800.ORACLE1 or visit oracle.com. Outside North America, find your local office at oracle.com/contact.



blogs.oracle.com/oracle



facebook.com/oracle



twitter.com/oracle

Integrated Cloud Applications & Platform Services

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 1119

Oracle JD Edwards EnterpriseOne: Scalable Elastic Deployment

November, 2019



Oracle is committed to developing practices and products that help protect the environment

The Oracle logo, consisting of the word "ORACLE" in a bold, white, sans-serif font, set against a solid red rectangular background.