

# Using Oracle Cloud Infrastructure Load Balancing for Your Highly Available WordPress Application

White Paper | April 2017 | Version 1.4





## 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 remains at the sole discretion of Oracle.

Questions or comments on this paper? Please email: [quickstart\\_us\\_grp@oracle.com](mailto:quickstart_us_grp@oracle.com).



ORACLE®



## Table of Contents

Disclaimer	1
Using Oracle Cloud Infrastructure Load Balancing for Your Highly Available WordPress Application	3
Assumptions	4
Target Audience	4
Introduction	5
Setting up the Network environment	6
Creating the Virtual Cloud Network	6
Creating the Security Lists	7
Create the Internet Gateway	8
Create the Route Tables	8
Configure the Route Tables	9
Create the Subnets	9
The Instances	10
Creating the Load Balancer	12
Load Balancer	12
Backend Set	13
Specify Backends	14
Creating a Listener	16
Modify the Security Lists	16
Configuring the Bastion Host and WordPress Instances	17
Testing the Load Balancer	23



Next Steps	23
Configuring Bastion SSH Pass-through	23



## Using Oracle Cloud Infrastructure Load Balancing for Your Highly Available WordPress Application

This white paper is designed as a reference guide for deploying the Oracle Cloud Infrastructure Load Balancing service on the Oracle Cloud Infrastructure platform.

To illustrate the functionality of Oracle Cloud Infrastructure Load Balancing, we will create a multi-node WordPress site that will be configured and fronted by Oracle Cloud Infrastructure Load Balancing. The environment will contain two WordPress web servers and a MySQL database server.

Oracle Cloud Infrastructure offers hourly metered Bare Metal and Virtual Machines and load balancing services. To support highly available deployments, Oracle Cloud Infrastructure builds regions with at least three Availability Domains (AD). Each AD is a fully independent data center with no fault domains shared across ADs. A two-tier application replicated across two ADs is highly available; a two-tier application replicated across three ADs increases data durability and ensures the reliability of the cluster and data set.



## Assumptions

Consumers of this white paper should –

- » Be familiar with the fundamentals of Oracle Cloud Infrastructure -
  - » <https://docs.us-phoenix-1.oraclecloud.com/>
  - » <https://docs.us-phoenix-1.oraclecloud.com/Content/General/Concepts/regions.htm>
- » The Oracle Cloud Infrastructure walkthrough is highly recommended if this is the first time you have used the platform –
  - » <https://docs.us-phoenix-1.oraclecloud.com/Content/GSG/Reference/overviewworkflow.htm>
- » Have a basic understanding of load balancing
- » Have a basic understanding of WordPress –
  - » <https://wordpress.org/>

## Target Audience

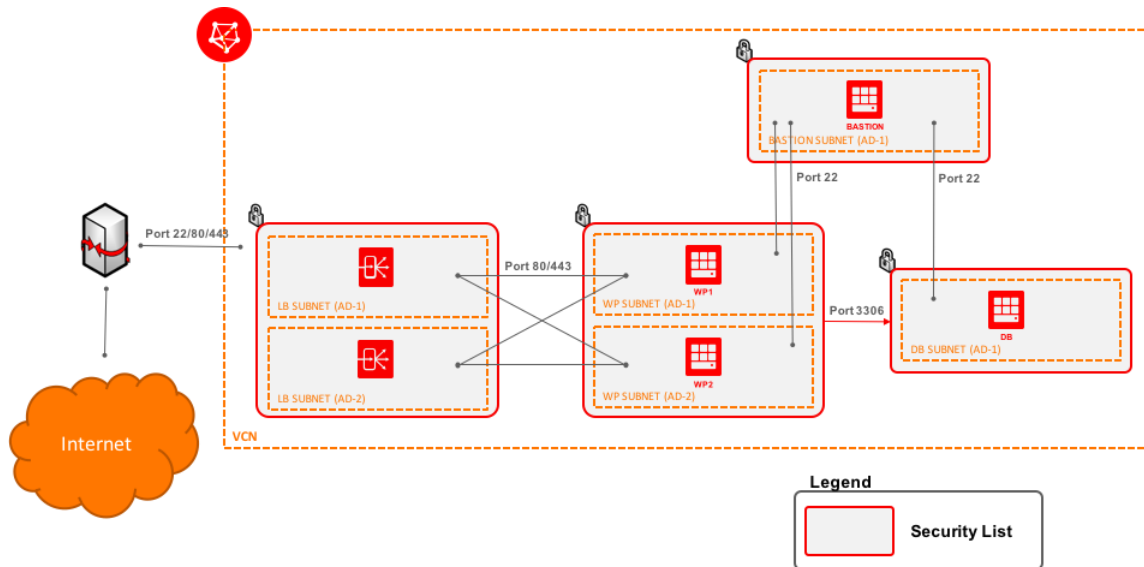
This white paper is targeted at customers who would like to get familiar with deploying highly available multi-tier applications in the Oracle Cloud Infrastructure.

## Introduction

In this white paper, we will create a small environment in Oracle Cloud Infrastructure to get you comfortable with the features of Oracle Cloud Infrastructure Load Balancing. We will build a WordPress environment consisting of two web servers and one database server. We will also leverage a bastion host to access these systems to ensure that they are not exposed to attack from the Internet while we build and secure them.

The environment used in this white paper consists of a single Virtual Cloud Network (VCN) with six subnets in a single compartment. Of those, three will be accessible via the Internet and will include Oracle Cloud Infrastructure Load Balancing and the bastion host while the remaining subnets will be private and contain the WordPress web and database instances. Security List rules will be configured to allow the public subnets to communicate with the Internet and with the private subnets, while the private subnets will only be allowed to communicate with the public subnets on specific ports (Port 80 in our case).

The diagram below is a high-level view into how we will be configuring the environment:




Confidential – Oracle Internal/Restricted/Highly Restricted 5

In this white paper we will set up a VCN across multiple Availability Domains (AD), set up a bastion host to allow for access to the environment from the Internet, set up the application tier for WordPress across two ADs, set up the database tier for WordPress, configure the WordPress application, set up the load balancer to front end the WordPress application and test that the WordPress application and load balancing are functioning as expected.

The white paper includes:

- A Virtual Cloud Network
- Route Tables
- Internet Gateways
- Private subnets (for the Web Servers)

- 
- Public subnets (for the Load Balancers)
  - Security Lists
  - Installing the instances (from a high level)
  - Load Balancer with its individual components
  - Configuration of the WordPress websites and database

## Setting up the Network environment

A VCN is your customizable private network in Oracle Cloud Infrastructure. Just like a traditional data center network, a VCN provides you with complete control over your network environment. This includes assigning your own private IP address space, creating subnets, creating route tables and configuring stateful firewalls. In the first step, we create the Virtual Cloud Network that will be used throughout this white paper. Subsequent section will cover the creation of the components within the VCN.

### Creating the Virtual Cloud Network

The Create Virtual Cloud Network wizard gives us two different options. The first will create only the VCN and no other components, while the second option will create the VCN, an Internet gateway, the default route table and three subnets (one in each Availability Domain). While the second option simplifies the creation of the environment, it does have some drawbacks. For example, all of the resources will use default names, which could cause confusion in environments with multiple VCNs. Also, as our WordPress environment is more complex than the default, 3 subnet/1 security list environment created by the wizard, we will need to manually create several more components. For that reason, we will create the VCN and each of its components manually. This will allow us to both be very prescriptive with the naming of each of the components and will also ensure that only the appropriate security rules are created to ensure the best security for your environment.

For simplicity, in this white paper, a single 10.0.0.0/16 network will be used. Create a new Virtual Cloud Network and make sure that you create only the VCN and not the related resources (as mentioned above you will create the rest manually):

- **Name:** Production WordPress Network
- **CIDR block:** 10.0.0.0/16



Create Virtual Cloud Network

[help](#)
[cancel](#)

NAME OPTIONAL

Production WordPress Network

☒ CREATE VIRTUAL CLOUD NETWORK ONLY
   
☐ CREATE VIRTUAL CLOUD NETWORK PLUS RELATED RESOURCES

Creates a Virtual Cloud Network only. You'll still need to set up at least one Subnet, Gateway, and Route Rule to have a working Virtual Cloud Network.

CIDR BLOCK

10.0.0.0/16

CIDR range must fall within 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16  
 Specified IP addresses: 10.0.0.0-10.0.0.255 (65,536 IP addresses)

Create Virtual Cloud Network

Once the VCN is created, our configuration will be similar to the one pictured below:

Create Virtual Cloud Network



Production WordPress Network  
OCID: ...wboeja Show Copy

CIDR Block: 10.0.0.0/16

Default Route Table: [Default Route Table for Production WordPress Network](#)

Created: Mon, 30 Jan 2017 17:58:19 GMT

...

AVAILABLE

### Best Practice:

*Manually create your VCN and all the related components (internet gateway, route table, security lists, etc). This will allow you to be very prescriptive in your naming of those components. This will make your life considerably easier as your Oracle Cloud Infrastructure environment grows. Although you can use the wizard, you will find that the names are tied to the name given to the VCN initially and that you will spend time “cleaning up” some of the other components. Only use the full wizard for test environments. Do not use it for production environments.*

You may notice that even though you selected the ‘Create Virtual Cloud Network only’ option in the wizard, it created a route table and a security list. These can be safely ignored, as we will not be using them in this white paper.

## Creating the Security Lists

As shown in the high-level diagram, four security lists are used. This section will only create the security lists themselves and not the security list rules (this will be done later).

Create the four security lists:

- Production WordPress - Bastion (Public)
- Production WordPress - DB (Private)
- Production WordPress - LB (Public)
- Production WordPress - Web (Private)

Again, we want to use prescriptive names to ensure that identifying the correct security list is easy when these are associated with the subnets we will create later.

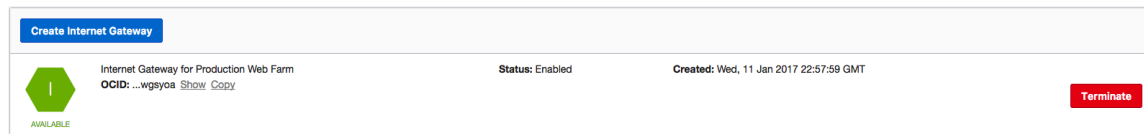
---

*As a best practice, always be as prescriptive in your naming of Oracle Cloud Infrastructure components as you can. This will make it easier in the future when you have to re-visit an environment.*

---

## Create the Internet Gateway

Next, we will create an Internet Gateway to allow subnets, such as the load balancer subnets, to communicate with the Internet. Create an Internet Gateway name 'Internet Gateway for Production Web Farm'.



## Create the Route Tables

Next, we will create the route tables for each of the different type of subnets we will create. These route tables will be used to define which subnets can route to the Internet.






We will be creating four new route tables. Two of them will be public and exposed to the Internet on specific ports, these route tables will be used for the load balancer and bastion subnets. The other two route tables will be private and will only be exposed to the public subnets on specific ports.

Create the following route tables:

- Production WordPress - Bastion (Public)
- Production WordPress - DB (Private)
- Production WordPress - LB (Public)
- Production WordPress - Web (Private)

Once the Route Tables are created, our configuration will be similar to the one pictured below:



Create Route Table					
 AVAILABLE	<a href="#">Default Route Table for Production WordPress Network</a> OCID: ...imvk2q <a href="#">Show</a> <a href="#">Copy</a>	<div>0</div> RULES	Compartment: Whitepapers	Created: Mon, 30 Jan 2017 17:58:19 GMT	...
 AVAILABLE	<a href="#">Production WordPress - Bastion (Public)</a> OCID: ...nlec2q <a href="#">Show</a> <a href="#">Copy</a>	<div>0</div> RULES	Compartment: Whitepapers	Created: Mon, 30 Jan 2017 18:09:42 GMT	...
 AVAILABLE	<a href="#">Production WordPress - DB (Private)</a> OCID: ...ltgzzq <a href="#">Show</a> <a href="#">Copy</a>	<div>0</div> RULES	Compartment: Whitepapers	Created: Mon, 30 Jan 2017 18:09:26 GMT	...
 AVAILABLE	<a href="#">Production WordPress - LB (Public)</a> OCID: ...3756pa <a href="#">Show</a> <a href="#">Copy</a>	<div>0</div> RULES	Compartment: Whitepapers	Created: Mon, 30 Jan 2017 18:09:52 GMT	...
 AVAILABLE	<a href="#">Production WordPress - Web (Private)</a> OCID: ...adaaxq <a href="#">Show</a> <a href="#">Copy</a>	<div>0</div> RULES	Compartment: Whitepapers	Created: Mon, 30 Jan 2017 18:09:33 GMT	...

## Configure the Route Tables

The route table rules will define which subnets can communicate with the Internet. During production, only the load balancer and bastion subnets will be allowed to do that; however, during the initial setup of the WordPress environment, you may want to add a rule for their subnets to allow them to communicate with the Internet to download images and updates.

Create the following rule for each of the subnets that will need access to the Internet:

CIDR Block	Target
0.0.0.0/0	Internet Gateway for Production Web Farm

## Create the Subnets







The final network components we need to create are the subnets. Six subnets are going to be required: two for the load balancer, one for the bastion host, two for the WordPress web servers, and one for the WordPress database. For fault tolerance, the two load balancer subnets and two web subnets should be created in different Availability Domains.

Create the following subnets:

Name	Availability Domain	CIDR Block	Route Table	Security Lists
Production WordPress - Web (Private) - PHX-AD-1	PHX-AD-1	10.0.1.0/24	Production WordPress - Web (Private)	Production WordPress - Web (Private)
Production WordPress - Web (Private) - PHX-AD-2	PHX-AD-2	10.0.2.0/24	Production WordPress - Web (Private)	Production WordPress - Web (Private)

Name	Availability Domain	CIDR Block	Route Table	Security Lists
<b>Production WordPress - DB (Private) - PHX-AD-1</b>	PHX-AD-1	10.0.10.0/24	Production WordPress - DB (Private)	Production WordPress - DB (Private)
<b>Production WordPress - LB (Public) - PHX-AD-1</b>	PHX-AD-1	10.0.20.0/24	Production WordPress - LB (Public)	Production WordPress - LB (Public)
<b>Production WordPress - LB (Public) - PHX-AD-2</b>	PHX-AD-2	10.0.21.0/24	Production WordPress - LB (Public)	Production WordPress - LB (Public)
<b>Production WordPress - Bastion (Public) - PHX-AD-1</b>	PHX-AD-1	10.0.30.0/24	Production WordPress - Bastion (Public)	Production WordPress - Bastion (Public)

Our configuration should look something similar to the image below:

Create Subnet					
 AVAILABLE	Production WordPress - Bastion (Public) - PHX-AD-1 OCID: ...mihxqz <a href="#">Show</a> <a href="#">Copy</a>	CIDR Block: 10.0.30.0/24 Virtual Router MAC Address: 00:00:17:09:9C:0B	Availability Domain: KKL:PHX-AD-1	Route Table: <a href="#">Production WordPress - Bastion (Public)</a> Security Lists: <a href="#">Production WordPress - Bastion (Public)</a>	DHCP Options: <a href="#">Default DHCP Options for Production WordPress Network</a> <a href="#">Terminate</a>
 AVAILABLE	Production WordPress - DB (Private) - PHX-AD-1 OCID: ...5fctya <a href="#">Show</a> <a href="#">Copy</a>	CIDR Block: 10.0.10.0/24 Virtual Router MAC Address: 00:00:17:09:9C:0B	Availability Domain: KKL:PHX-AD-1	Route Table: <a href="#">Production WordPress - DB (Private)</a> Security Lists: <a href="#">Production WordPress - DB (Private)</a>	DHCP Options: <a href="#">Default DHCP Options for Production WordPress Network</a> <a href="#">Terminate</a>
 AVAILABLE	Production WordPress - LB (Public) - PHX-AD-1 OCID: ...6c7hqq <a href="#">Show</a> <a href="#">Copy</a>	CIDR Block: 10.0.20.0/24 Virtual Router MAC Address: 00:00:17:09:9C:0B	Availability Domain: KKL:PHX-AD-1	Route Table: <a href="#">Production WordPress - LB (Public)</a> Security Lists: <a href="#">Production WordPress - LB (Public)</a>	DHCP Options: <a href="#">Default DHCP Options for Production WordPress Network</a> <a href="#">Terminate</a>
 AVAILABLE	Production WordPress - Web (Private) - PHX-AD-1 OCID: ...ev5zqq <a href="#">Show</a> <a href="#">Copy</a>	CIDR Block: 10.0.1.0/24 Virtual Router MAC Address: 00:00:17:09:9C:0B	Availability Domain: KKL:PHX-AD-1	Route Table: <a href="#">Production WordPress - Web (Private)</a> Security Lists: <a href="#">Production WordPress - Web (Private)</a>	DHCP Options: <a href="#">Default DHCP Options for Production WordPress Network</a> <a href="#">Terminate</a>
 AVAILABLE	Production WordPress - LB (Public) - PHX-AD-2 OCID: ...kzuuvq <a href="#">Show</a> <a href="#">Copy</a>	CIDR Block: 10.0.21.0/24 Virtual Router MAC Address: 00:00:17:09:9C:0B	Availability Domain: KKL:PHX-AD-2	Route Table: <a href="#">Production WordPress - LB (Public)</a> Security Lists: <a href="#">Production WordPress - LB (Public)</a>	DHCP Options: <a href="#">Default DHCP Options for Production WordPress Network</a> <a href="#">Terminate</a>
 AVAILABLE	Production WordPress - Web (Private) - PHX-AD-2 OCID: ...tnj2la <a href="#">Show</a> <a href="#">Copy</a>	CIDR Block: 10.0.2.0/24 Virtual Router MAC Address: 00:00:17:09:9C:0B	Availability Domain: KKL:PHX-AD-2	Route Table: <a href="#">Production WordPress - Web (Private)</a> Security Lists: <a href="#">Production WordPress - Web (Private)</a>	DHCP Options: <a href="#">Default DHCP Options for Production WordPress Network</a> <a href="#">Terminate</a>

## The Instances

Our environment requires four instances. Three will be used for the WordPress site itself, the fourth will be used as a bastion host.

We will need to create several key pairs for the instances. It is a best practice to create a key pair for each type of the instances (bastion, web, and db). Create three sets of key pairs using the instructions on the Oracle Cloud Infrastructure documentation page <https://docs.us-phoenix-1.oraclecloud.com/Content/Compute/Tasks/managingkeypairs.htm>. Use the public keys in the instance creation while keeping the private keys secure. We will use the private keys to log in to the instance through the bastion host.

Use the following properties to create the instances (the shape we used for this white paper is a recommendation and you can scale it up or down as you deem fit):

Name	Image	Shape	Availability Domain	Virtual Cloud Network	Subnet
<b>WP-1</b>	Oracle-Linux-7.x	VM.Standard1.2	PHX-AD-1	Production WordPress Network	Production WordPress - Web (Private) - PHX-AD-1
<b>WP-2</b>	Oracle-Linux-7.x	VM.Standard1.2	PHX-AD-2	Production WordPress Network	Production WordPress - Web (Private) - PHX-AD-2
<b>WP-DB</b>	Oracle-Linux-7.x	VM.Standard1.2	PHX-AD-1	Production WordPress Network	Production WordPress - DB (Private) - PHX-AD-1
<b>Bastion-1</b>	Oracle-Linux-7.x	VM.Standard1.2	PHX-AD-1	Production WordPress Network	Production WordPress - Bastion (Public) - PHX-AD-1

Use the following table to note the public and RFC1918 IP addresses for each instance –

Instance	Public IP	RFC1918 IP
<b>WP-1</b>		
<b>WP-2</b>		
<b>WP-DB</b>		
<b>Bastion-1</b>		

---

*NOTE: All inter-instance communication should be across the RFC1918 address of the instances, not the public IP. Using the public IP adds latency to the connection and limits the bandwidth. Using the RFC1918 IP supports access to the full network bandwidth and the lowest possible latency. Also, the only instance that will be accessible via the public IP address will be the bastion host. The rest will be blocked via security lists.*

---

We are not going to worry about the configuration of the instances yet (we will do that at the end of the walkthrough). We will, however, need at least the two web instances to be up and running before we complete the load balancer configuration.

## Creating the Load Balancer

Now that you have the network environment and the instances set up, you can create the load balancer. The load balancer is actually made up of several components:

- Load Balancer
- Backend Set
- Backends
- Listener
- Security lists

### Load Balancer

The first step is to create the load balancer itself. Using the wizard in the console, create a Load Balancer with the following properties:

Name	Shape	Virtual Cloud Network	Subnet (1 of 2)	Subnet (2 of 2)
WordPress	100Mbps	Production WordPress Network	Production WordPress – LB (Public) - PHX-AD-1	Production WordPress – LB (Public) - PHX-AD-2

Create Load Balancer

[help](#)
[cancel](#)

If your VCN or subnets are in a different compartment than your load balancer, [click here](#) to enable compartment selection for those resources.

NAME

WordPress

SHAPE

100Mbps

Network Information

Specify the VCN in which the load balancer accepts incoming traffic.

VIRTUAL CLOUD NETWORK

Production WordPress Network

The Load Balancing Service uses two subnets to ensure high availability for your public IP address. You can use the address as a front-end for incoming traffic and to load balance that traffic across all subnets within the VCN.

SUBNET (1 OF 2)

Production WordPress - LB (Public) - PHX-AD-1

SUBNET (2 OF 2)

Production WordPress - LB (Public) - PHX-AD-2

The load balancer shape will determine the amount of traffic the load balancer can handle (100Mbps, 400Mbps, or 8000Mbps).

Once the load balancer is created, take note of its public IP address. We will need to configure WordPress with that IP address to ensure that WordPress functions properly behind a load balancer.

Load balancer	Public IP
WordPress	

## Backend Set

Next, create the backend set. The backend set is a logical container that will contain one or more compute instances. The load balancer will direct traffic to each of these instances based on the type of policy associated with the backend set for a specific protocol (in our case, Weighted Round Robin).

Create a Backend Set with the following properties:

Name	Policy	Use SSL	Protocol	Port	URL Path (URI)
WPBackendSet_80	Weighted Round Robin	(not checked)	HTTP	80	/

Create Backend Set

[help](#)
[cancel](#)

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

NAME

WPBackendSet\_80

POLICY

Weighted Round Robin

USE SSL

Health Check

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

PROTOCOL

HTTP

PORT

80

INTERVAL IN MS (Optional)

TIMEOUT IN MS (Optional)

NUMBER OF RETRIES (Optional)

URL PATH (URI)

/

STATUS CODE (Optional)

RESPONSE BODY REGEX (Optional)

## Specify Backends

We now need to associate the two web servers that we created earlier with the backend set. This can either be done by using the internal IP addresses of the instances or by using the Instance OCIDs (instructions on how to find the OCIDs for our created instances are shown below). If we use the Instance OCIDs, the wizard will automatically create the security rules for us between the load balancer (public) subnets and the web (private) subnets. If the IP address is used, we will need to create the security rules manually. For our white paper, we will use the instance OCIDs.

In the Edit Backends wizard, add the two instances with the following properties:

Instance OCID	Port	Weight
OCID of WP-1	80	50
OCID of WP-2	80	50

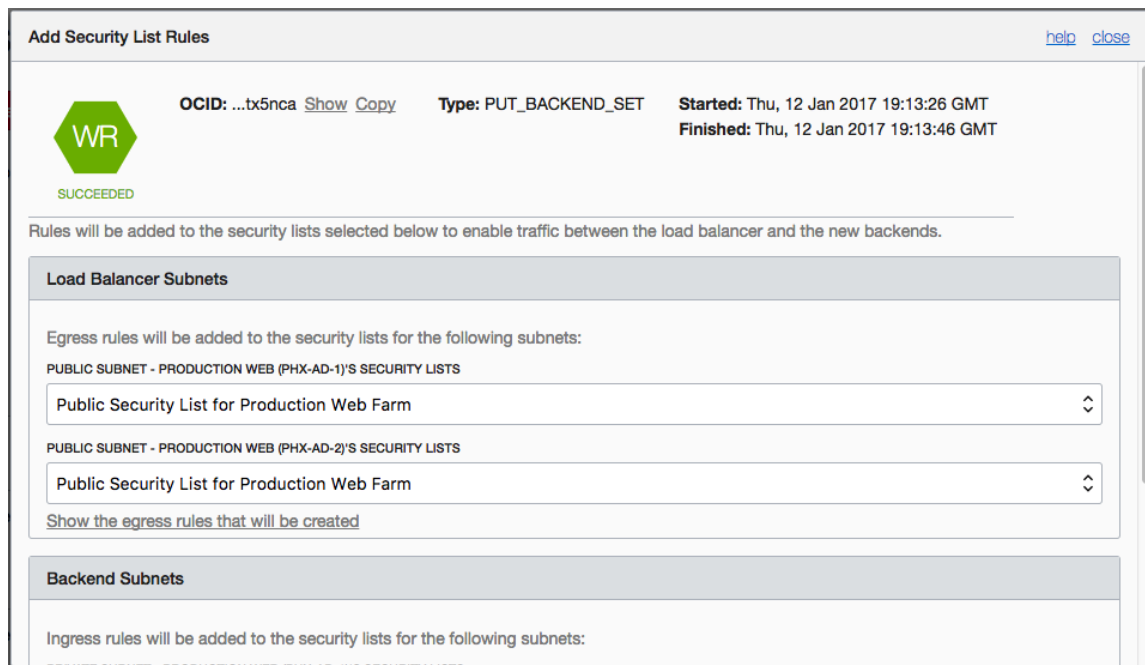
*Note that the weight specified in the edit backends wizard is not a percentage. It is simply a measure of how much of the the traffic will be directed to that instance based on the total weight associated with the load balancer. In our example, each instance has a weight of 50 out of a total of 100 (in essence, 50% of the traffic). However, we would get the same result if we assigned a weight of 1 to each of the instances (1 out of 2 will direct 50% of the traffic to that instance).*



## Finding the OCIDs of your instances:

1. In the Edit Backends Wizard, click the [View Instances](#) link. This will open another tab in your browser and navigate to the Compute Instances section of the OBMC Portal (<https://console.us-az-phoenix-1.oracleiaas.com/#!/a/compute/instances>).
2. Find an instance for which you need the OCID and click the **Copy** link.
3. Go back to the previous tab (Edit Backends) and paste the OCID in the Instance OCID field in the Wizard.
4. Repeat steps 2 and 3 for all other Instances that are going to be added to the Load Balancer Backend.

The Edit Backends Wizard will automatically create the required Security Rules in the appropriate subnets (the icon in the top left of the wizard will be orange while these are being created). However, that option will not be available until after the Backends are configured. Once they are (the icon in the top left of the wizard will change to green), we will be able to review the rules that are being created and complete the wizard:



The following ingress stateful rules are automatically created on the Production WordPress – Web (Private) Security List:

**Source:** 10.0.20.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 80  
**Allows:** TCP traffic for ports: 80 HTTP

**Source:** 10.0.21.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 80

**Allows:** TCP traffic for ports: 80 HTTP

The following stateful rules are automatically created on the Production WordPress – LB (Public) Security List:

**Destination:** 10.0.2.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 80  
**Allows:** TCP traffic for ports: 80 HTTP

**Destination:** 10.0.1.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 80  
**Allows:** TCP traffic for ports: 80 HTTP

### Creating a Listener

The Listener is an endpoint that will listen for incoming traffic on the floating IP address of the Load Balancer. Since we are only using port 80 in our white paper, create a listener with the following properties:

Name	Protocol	Port	SSL	Backend Set
WebserverListener_80	HTTP	80	(unchecked)	WPBackendSet_80

---

*If you enable SSL on your installation of WordPress, you will need to create a second listener to listen on port 443 and direct traffic to a port 443 enabled backend set.*

---

Create Listener

[help](#) [cancel](#)

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

NAME

WPLListener\_80

PROTOCOL

HTTP

?

PORT

80

?

USE SSL

☐

BACKEND SET

WPBackendSet

Create

### Modify the Security Lists

The final step in the configuration is to create an Ingress rule on the Production WordPress – LB (Public) security list to allow HTTP traffic on port 80 from the Internet:

**Source:** 0.0.0.0/0  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 80  
**Allows:** TCP traffic for ports: 80 HTTP

---

*Note that a source of 0.0.0.0/0 implies that any address on the Internet will be allowed to connect to port 80. If this is a private system, we can limit access by entering the IP range of the clients that are allowed to connect.*

---

## Configuring the Bastion Host and WordPress Instances

Now that we have the load balancer configured, we can configure WordPress. We will access the web and database tiers through the bastion host. Before we do that, we have to add some security list rules.

Add the following rules to the Production WordPress - Bastion (Public) security list:

**Type:** Ingress (stateful)  
**Source:** 0.0.0.0/0  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 22  
**Allows:** TCP traffic for ports: 22 SSH Remote Login Protocol

**Type:** Egress (stateful)  
**Destination:** 10.0.1.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 22  
**Allows:** TCP traffic for ports: 22 SSH Remote Login Protocol

**Type:** Egress (stateful)  
**Destination:** 10.0.2.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 22  
**Allows:** TCP traffic for ports: 22 SSH Remote Login Protocol

**Type:** Egress (stateful)  
**Destination:** 10.0.10.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 22  
**Allows:** TCP traffic for ports: 22 SSH Remote Login Protocol

**Type:** Egress (stateful)  
**Destination:** 0.0.0.0/0  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** All  
**Allows:** TCP traffic for ports: all

Add the following rules to the Production WordPress - Web (Private) security list:

**Type:** Ingress (stateful)

**Source:** 10.0.30.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 22  
**Allows:** TCP traffic for ports: 22 SSH Remote Login Protocol

**Type:** Egress (stateful)  
**Destination:** 10.0.10.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 3306  
**Allows:** TCP traffic for ports: 3306

**Type:** Egress (stateful)  
**Destination:** 0.0.0.0/0  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** All  
**Allows:** TCP traffic for ports: all

Add the following rules to the Production WordPress - DB (Private) security list:

**Type:** Ingress (stateful)  
**Source:** 10.0.30.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 22  
**Allows:** TCP traffic for ports: 22 SSH Remote Login Protocol

**Type:** Ingress (stateful)  
**Source:** 10.0.1.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 3306  
**Allows:** TCP traffic for ports: 3306

**Type:** Ingress (stateful)  
**Source:** 10.0.2.0/24  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** 3306  
**Allows:** TCP traffic for ports: 3306

**Type:** Egress (stateful)  
**Destination:** 0.0.0.0/0  
**IP Protocol:** TCP  
**Source Port Range:** All  
**Destination Port Range:** All  
**Allows:** TCP traffic for ports: all

---

*Rather than going through the theory behind a bastion host and its installation/configuration here, refer to Scott Lowe's blog post found at <http://blog.scottlowe.org/2015/11/21/using-ssh-bastion-host/>.*

---

For the database instance:

1. SSH into the instance through the bastion instance (see the "Configuring bastion SSH pass-through" section to set up the ssh tunnel through the bastion):



```
$ ssh instance_name
```

2. Update the instance:

```
$ sudo yum update -y
```

3. Enable yum for mysql:

```
$ sudo vi /etc/yum.repos.d/public-yum-ol7.repo
```

4. Locate the 'ol7\_MySQL56' option and change 'enabled' to a value of 1.
5. Install the web server, php, tools and php libraries (in case we need to do some testing locally, although we will not enable the web server on this instance):

```
$ sudo yum install mod_ssl openssl httpd php php-mysql unzip mlocate wget php-gd php-xml -y
```

6. Open up the operating system firewall to allow the web instances to communicate with the database instance. We limit communication on the web port to the VCN subnet.

```
$ sudo firewall-cmd --permanent --zone=public \
--add-rich-rule='rule family="ipv4" source address="10.0.1.0/24" \
port protocol="tcp" port="3306" accept'

$ sudo firewall-cmd --permanent --zone=public \
--add-rich-rule='rule family="ipv4" source address="10.0.2.0/24" \
port protocol="tcp" port="3306" accept'

$ sudo firewall-cmd --reload
```

7. Start and enable the mysql server:


```
$ sudo systemctl start mysqld  
$ sudo systemctl enable mysqld
```

8. Secure the mysql server:

```
$ sudo mysql_secure_installation  
  
Set Password = Y  
Enter a_strong_password  
Reenter a_strong_password  
Remove Anonymous user? = Y  
Disallow root login remotely? = Y  
Remove Test Database? = Y  
Reload Privileges? = Y
```

9. Create the WordPress database and account user:

```
$ sudo mysql -u root -p  
  
CREATE DATABASE wordpress;  
GRANT ALL PRIVILEGES ON wordpress.* TO wordpress@ip_address_of_wp-1_instance;  
GRANT ALL PRIVILEGES ON wordpress.* TO wordpress@ip_address_of_wp-2_instance;  
SET PASSWORD FOR wordpress@ ip_address_of_wp-1_instance =  
PASSWORD('use_a_strong_password');  
SET PASSWORD FOR wordpress@ ip_address_of_wp-2_instance =  
PASSWORD('use_a_strong_password');  
FLUSH PRIVILEGES;  
exit
```



For each of the web instances:

1. SSH into the instance through the bastion instance (see the “Configuring bastion SSH pass-through” section to set up the ssh tunnel through the bastion):

```
$ ssh Instance_name
```

2. Patch the instance:

```
$ sudo yum update -y
```

3. Install the web server, php, tools, and php libraries:

```
$ sudo yum install mod_ssl openssl httpd -y  
$ sudo yum install php php-mysql -y  
$ sudo yum install unzip mlocate wget -y  
$ sudo yum install php-gd php-xml -y
```

4. Open the operating system firewall to allow the load balancer to communicate with the web instance.  
We limit communication on the web port to the VCN subnet.

```
$ sudo firewall-cmd --permanent --zone=public \  
--add-rich-rule='rule family="ipv4" source address="10.0.20.0/24" \  
port protocol="tcp" port="80" accept'  
  
$ sudo firewall-cmd --permanent --zone=public \  
--add-rich-rule='rule family="ipv4" source address="10.0.21.0/24" \  
port protocol="tcp" port="80" accept'  
  
$ sudo firewall-cmd --reload
```

5. Start and enable the web server:

```
$ sudo systemctl start httpd
$ sudo systemctl enable httpd
```

7. Download the latest version of WordPress:

```
$ cd ~
$ sudo wget https://wordpress.org/latest.zip
```

8. Unzip WordPress:

```
$ ls -lrt
$ sudo unzip latest.zip
$ cd wordpress
$ cd /var/www/html
$ rm -rf *
$ cp ~/wordpress/* /var/www/html -R
```

9. Configure WordPress to connect to the database:

```
$ cd /var/www/html
$ sudo cp wp-config-sample.php wp-config.php
$ sudo vi wp-config.php

Modify the lines below:

define('DB_NAME', 'wordpress');
define('DB_USER', 'wordpress');
define('DB_PASSWORD', 'use_a_strong_password');
define('DB_HOST', 'internal_ip_address_of_database_instance');

Add the lines below to the end of the wp-config.php file:

define( 'WP_SITEURL', 'http://ip_address_of_load_balancer' );
define( 'WP_HOME', 'http://ip_address_of_load_balancer' );
```

10. Complete WordPress configuration through a browser by navigating to `http://ip_address_of_load_balancer/`.

---

*These instructions for setting up WordPress on a single Oracle Linux instance were modified for a multi-node installation from <http://blog.simplespider.com/2015/05/install-wordpress-on-oracle-linux-7.html>.*

---



## Testing the Load Balancer

Now that the load balancer is fully configured, you can test its functionality by navigating to the Floating IP address on a web browser. If the load balancer has been configured properly, the WordPress site will appear. The extra security rules that we created to configure the web and database instances can now be removed.

You can test that the load balancer is balancing the communication across the web servers by temporarily stopping the web server on one of the web instances:

```
$ sudo systemctl stop httpd
```

The WordPress site should continue to function even though one of the instances has been temporarily disabled.

Make sure to restart the HTTP server on the instance after you complete the test:

```
$ sudo systemctl start httpd
```

Note that it will take a short period for the load balancer to detect that the web instance can no longer be communicated with on port 80. This is controlled by the Timeout and Number of retries properties on the load balancer health check.

## Next Steps

Now that we have the load balancer created, you try adding more web instances and see how the load balancer handles them. Also, enable SSL on the web instances and add port 443 to the load balancer. Remember to enable port 443 into the web instances firewall.

## Configuring Bastion SSH Pass-through

To simplify access to the web and db instances, we will use an SSH tunnel through the bastion host. This will allow us to connect to the backend instances through the bastion host with a single command. To do this, we will use the SSH config file on our local client to make using the bastion instance completely transparent.

Edit the `~/.ssh/config` file and add this section. Repeat this for the second web instance and the db instance:

```
Host instance_name
    Hostname <private IP of instance_name>
    User opc
    IdentityFile <private key for instance_name>
    ProxyCommand ssh -i <private key for bastion> opc@<bastion public IP> -W
    %h:%p %r
```

**Oracle Corporation, World Headquarters**

500 Oracle Parkway  
Redwood Shores, CA 94065, USA

**Worldwide Inquiries**

Phone: +1.650.506.7000  
Fax: +1.650.506.7200

## CONNECT WITH US



[blogs.oracle.com/oracle](https://blogs.oracle.com/oracle)



[facebook.com/oracle](https://facebook.com/oracle)



[twitter.com/oracle](https://twitter.com/oracle)



[oracle.com](https://oracle.com)

**Integrated Cloud Applications & Platform Services**

Copyright © 2017, 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. 0116

Using Oracle Cloud Infrastructure Load Balancing for Your Highly Available WordPress Application  
April 2017

Author: Barry Shilmover ([barry.shilmover@oracle.com](mailto:barry.shilmover@oracle.com))



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