|  |
| --- |
| A picture containing drawing  Description automatically generated  Statement of Direction |
| OCI Compute Failover using OCI Functions  Ionut A. Vladu Senior OCI Specialist Oracle EMEA  September 2021, Version 1.0  Copyright © 2021, Oracle and/or its affiliates  Confidential – Oracle Restricted |



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

Table of contents

[Disclaimer 2](#_Toc86671149)

[Versioning 4](#_Toc86671150)

[Introduction 5](#_Toc86671151)

[Use case description: 5](#_Toc86671152)

[Setup 6](#_Toc86671153)

[Load Balancer Backend Set Configuration 6](#_Toc86671154)

[Deploy Failover Function 8](#_Toc86671155)

[Create Notification 9](#_Toc86671156)

[Create Alarm to trigger Function 10](#_Toc86671157)

[Test Failover 11](#_Toc86671158)

Versioning

|  |  |  |
| --- | --- | --- |
| Version | Author | Description |
| 1.0 | [Ionut.vladu@oracle.com](mailto:Ionut.vladu@oracle.com) | Create document |
|  |  |  |
|  |  |  |

Introduction

This document describes a use case of automatic OCI Compute failover in multiple Availability Domains using OCI Functions in order to maintain application accessible in case of issues.

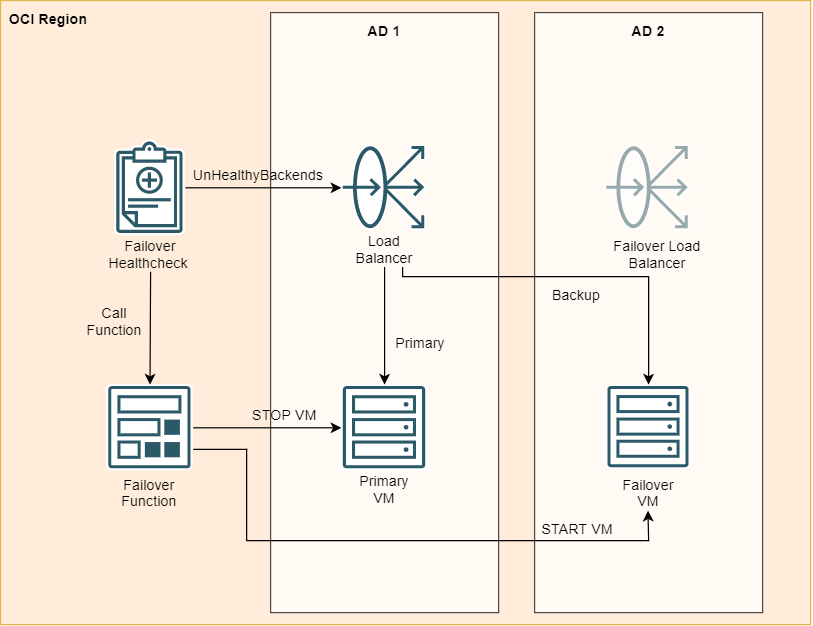
### Use case description:

We have an application running in OCI on Virtual Machines. The Virtual Machines are in a private subnet, so public access is made through a public Load Balancer. The following scenario was created to protect the application in case of issues.

We have a **Primary VM** that sits in Availability Domain 1 and a **Failover VM** in Availability Domain 2. All traffic is routed to the Primary VM as the Failover VM is marked as **Backup** in the Load Balancer and is not running.

In case the Primary VM becomes unhealthy, an alarm will trigger an OCI Function that will automatically START the Failover VM and STOP the Primary VM, if not already stopped.

The Load Balancer will route the traffic to the Backup site until the Primary VM will be healthy again. Once the Primary VM is healthy again, the Failover VM can be stopped.



Setup

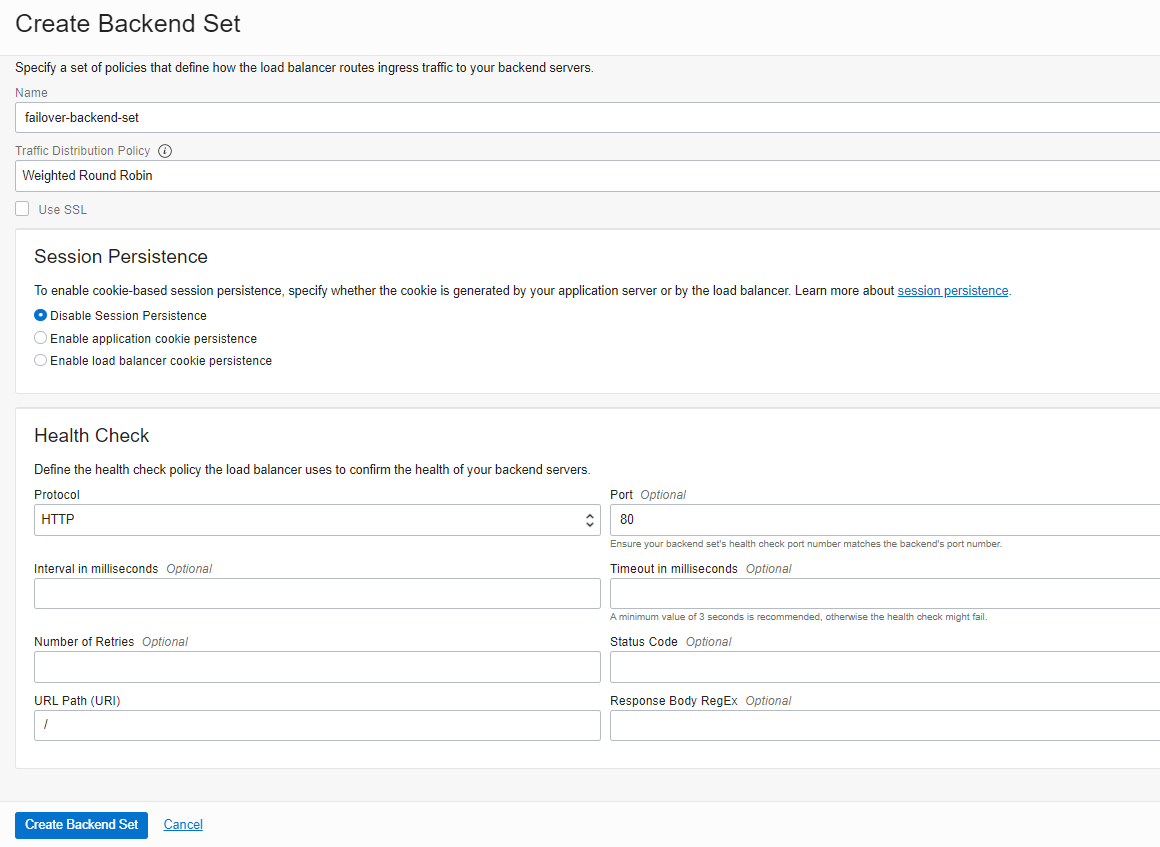
*Menu 🡪 Networking 🡪 Load Balancers*

### Load Balancer Backend Set Configuration

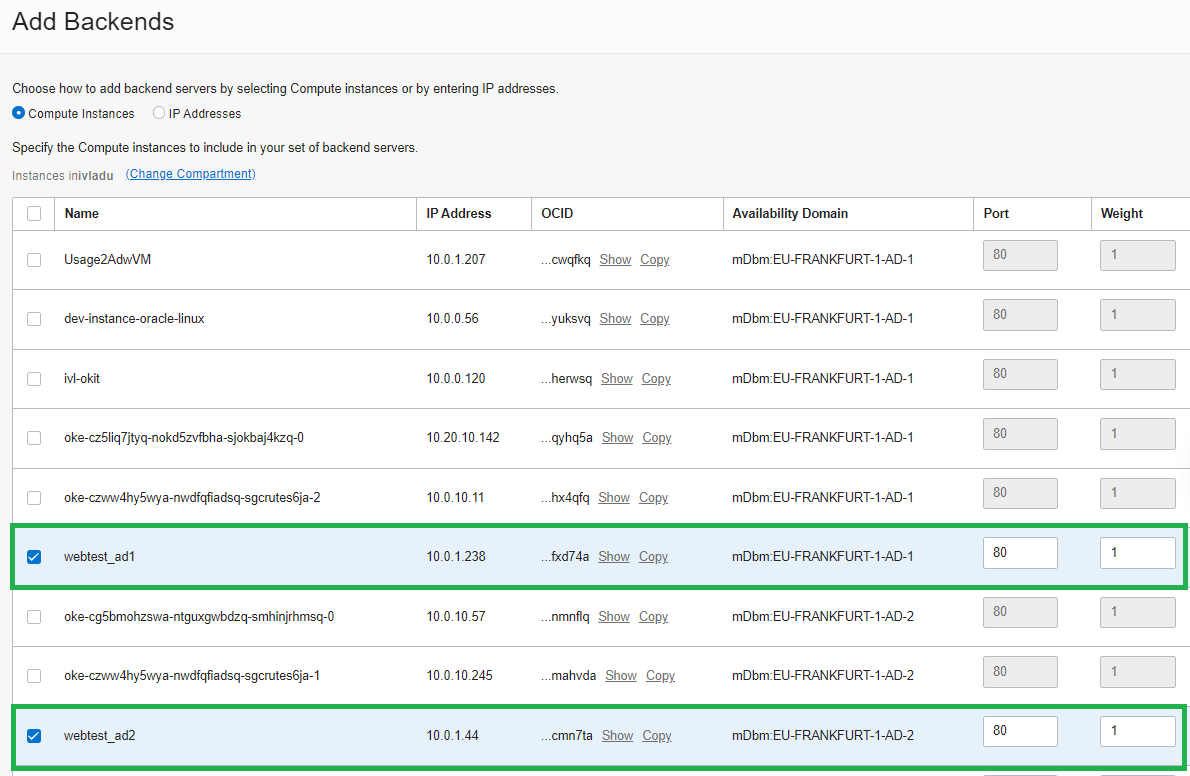
*Note: The creation of the Virtual Machines and the Load Balancer is excluded from this document as there is nothing specific to those steps.*

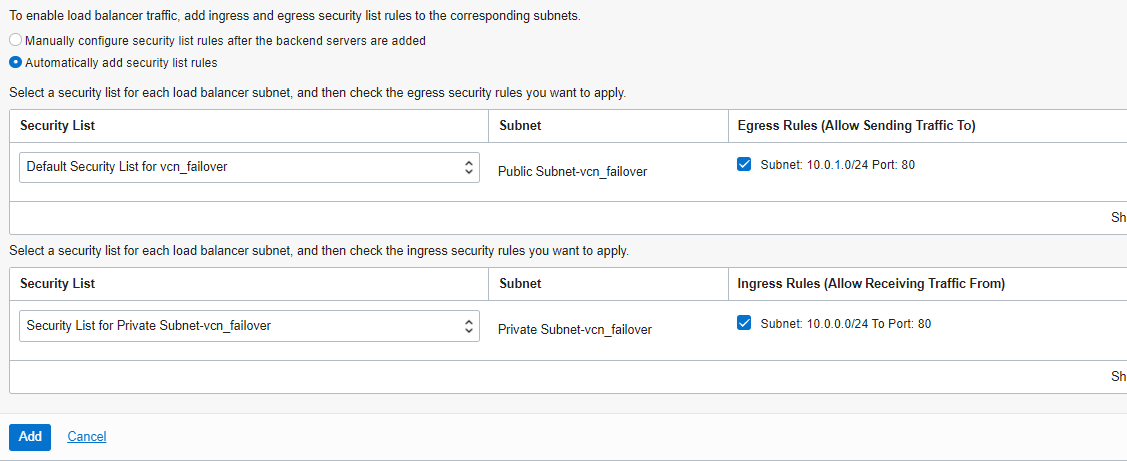
Once the Virtual Machines and the Load Balancer are created, we need to add the VMs to a **backend set** in the Load Balancer.

1. Create a Backend Set

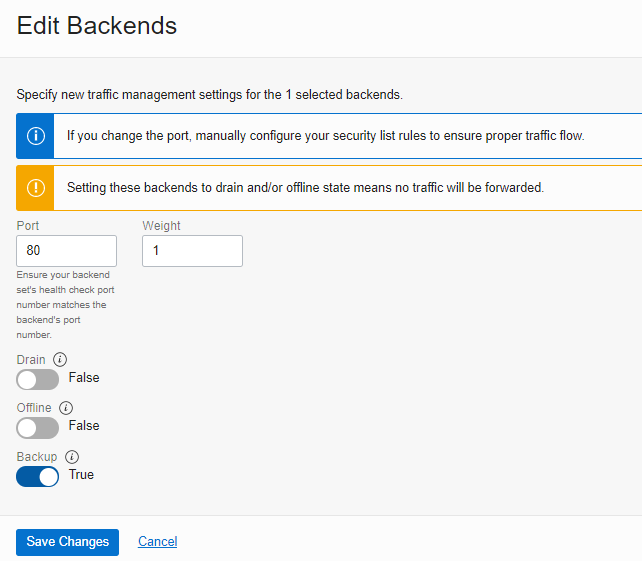


1. Add Backends
   1. Select the two VMs and check “Automatically add security list rules” and click **ADD**





1. Select the **Failover VM,** click on **Actions** and then **Edit** to mark it as **Backup**



Now the **Primary VM** and the **Failover VM** should be in the same backend set, but the **Failover** should be marked as **Backup**.

The Load Balancer will only send traffic to the **Primary**, as long as it is healthy, and will only switch and send traffic to the **Failover** if **Primary** is unhealthy.

Once **Primary** is healthy again, the Load Balancer will switch back and send traffic only to **Primary**.

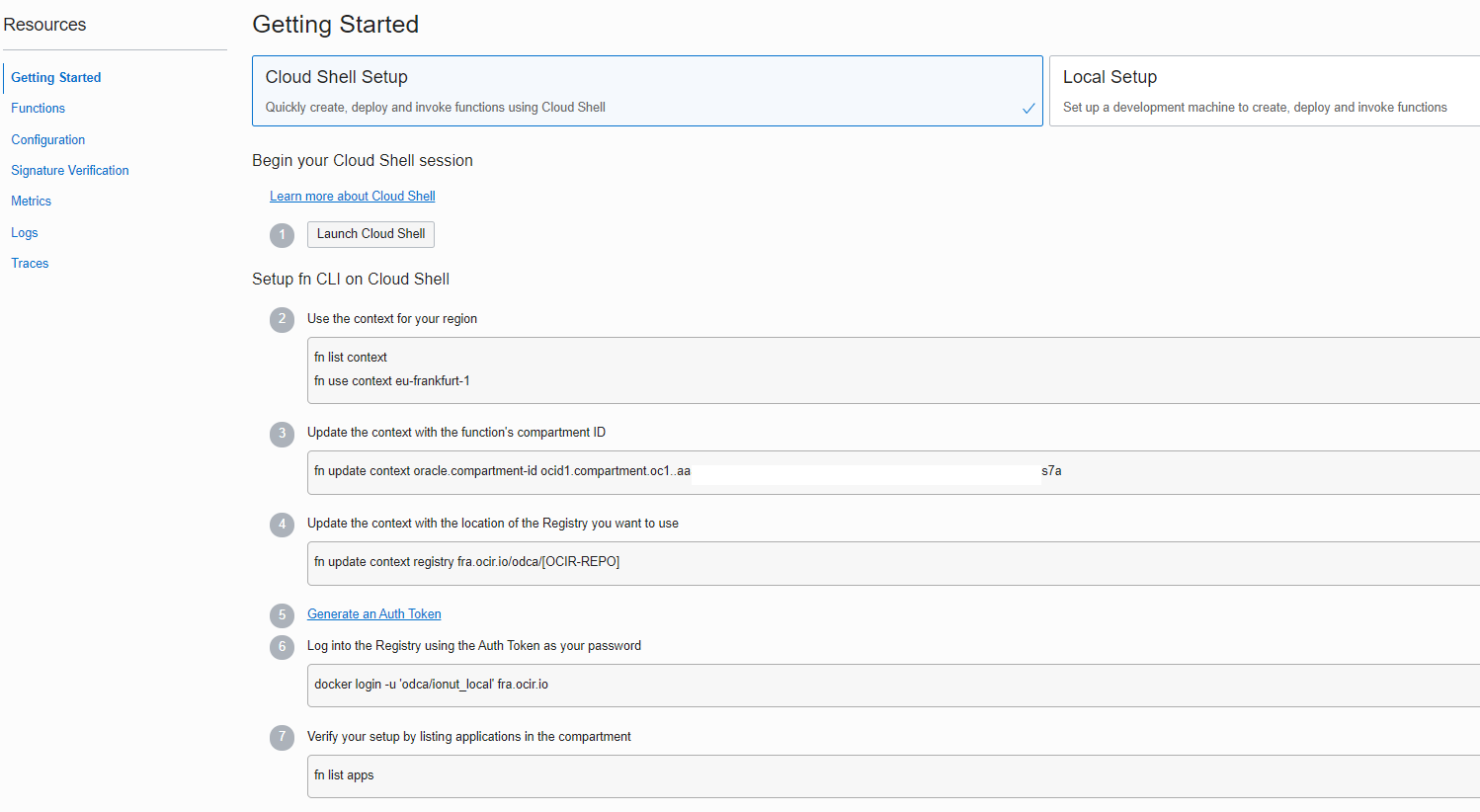


### Deploy Failover Function

*Menu 🡪 Developer Services 🡪 Applications (under Functions)*

Before deploying the OCI Function, make sure you have the right policies. [Check Step 4 in Section A: Create Policy for group and service.](https://docs.oracle.com/en-us/iaas/Content/Functions/Tasks/functionsquickstartcloudshell.htm)

1. Create a new **Application** under Developer Services -> Functions
   1. It is recommended to have the application in the same VCN as the compute VMs, but in a Public Subnet
2. Follow the first 7 steps from the Getting Started Guide using Cloud Shell



1. Download the source code for the Failover Function and navigate to the Function’s folder

*git clone* [*https://github.com/iavladu/oci-compute-failover.git*](https://github.com/iavladu/oci-compute-failover.git)

*cd oci-compute-failover*

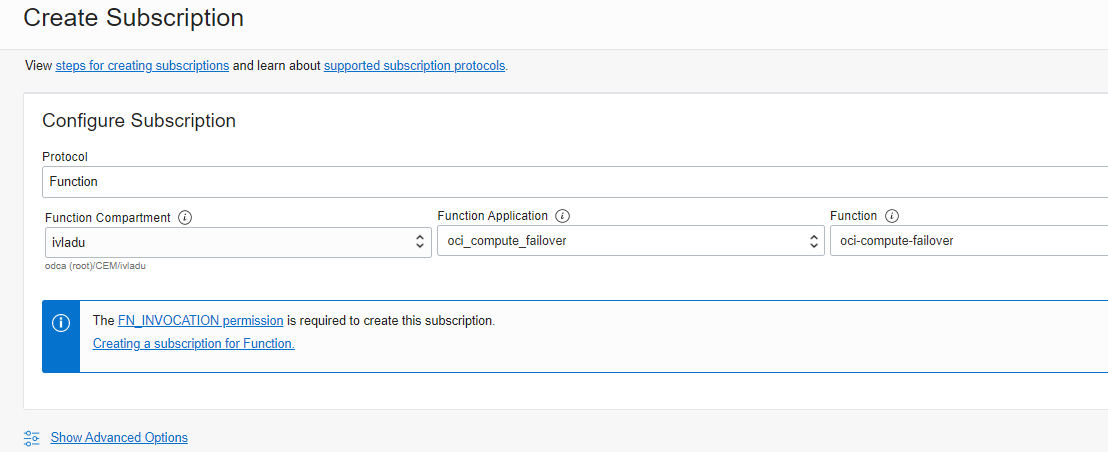
1. Deploy the Failover Function to your newly created Function

*fn -v deploy --app <your application name>*

### Create Notification

*Menu 🡪 Developer Services 🡪 Notifications (under Application Integration)*

1. Create Topic
2. Create Subscription
   1. Select **Function** protocol and the failover application and function

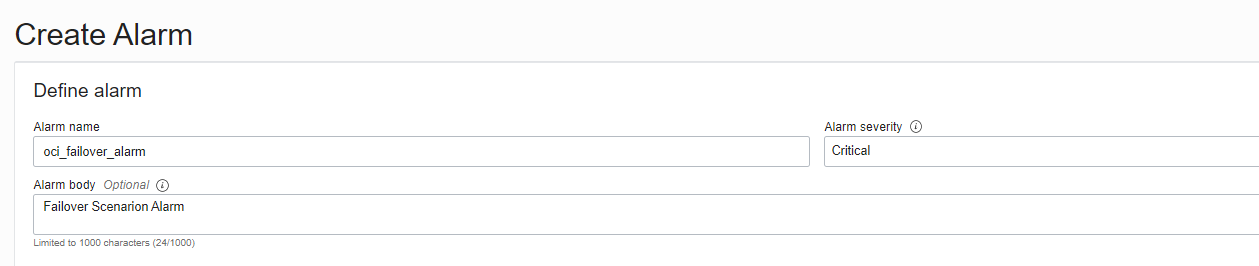


1. Additional subscriptions for email, slack or other channels can be created as well to get notified when the alarm will be triggered

### Create Alarm to trigger Function

*Menu 🡪 Observability & Management 🡪 Alarm Definition (under Monitoring)*

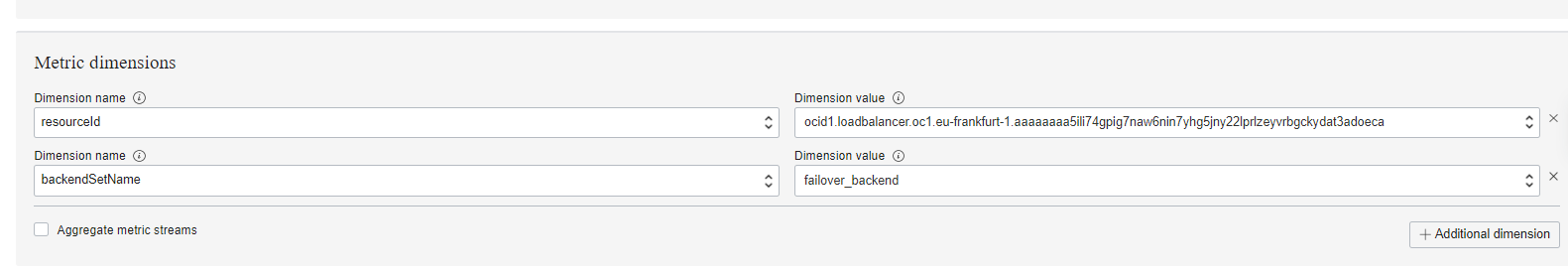
1. Create Alarm
   1. Give it a name, a body and select the Severity – Critical is recommended



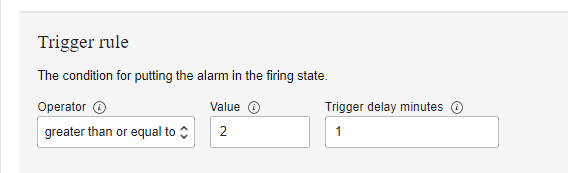
* 1. Under Metric Description
     1. Select the compartment
     2. The **oci\_lbaas** Metric Namespace
     3. The **UnHealthyBackendServers** Metric Name



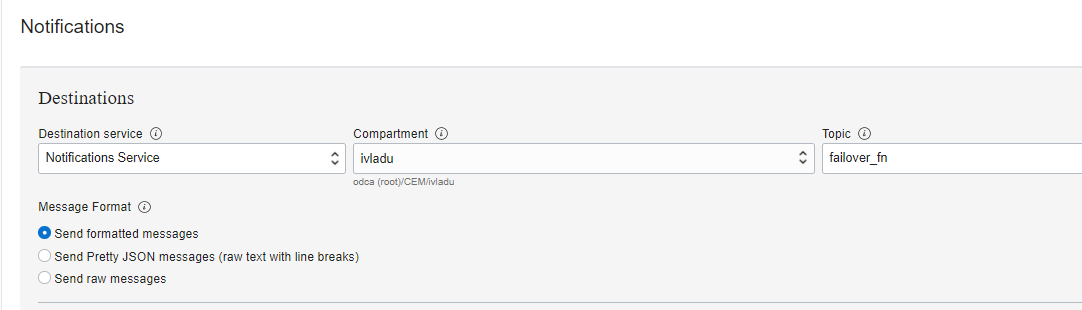
* 1. Under Metric Dimensions
     1. Select **resourceId** as Dimension Name and the **Load Balancer OCID** as Dimension Value
     2. Add Additional dimension
     3. Select **backendSetName** as Dimension Name and the **Backend Set Name** as Dimension Value



* 1. Under Trigger rule
     1. Select **greater than or equal to** as Operator and **2** as Value



* 1. Select the Topic for failover



* 1. Save alarm

Test Failover

All traffic should go to the Primary VM under normal circumstances. In order to test the failover, there are two ways:

1. Stop the Primary VM
   1. If the Primary VM is shut down, the Alarm will be triggered, and the Failover Function will be invoked
   2. The Function will start the Failover VM and will not attempt to stop the Primary VM, as the status is not RUNNING
   3. Once the Failover VM is up and running, the Load Balancer will automatically send traffic to this machine instead
   4. If in the meantime, the Primary VM becomes healthy again, the Load Balancer will switch back to it and traffic will only go to the Primary and the Failover VM can be stopped
2. Stop the webserver on the Primary VM (or the services that is subject to the health check), but not the VM itself
   1. If the health check on the Load Balancer (in this case, HTTP) becomes unhealthy, the alarm will be triggered, and the Failover Function will be invoked
   2. The Function will start the Failover VM and will also stop the Primary VM
   3. Once the Failover VM is up and running, the Load Balancer will automatically send traffic to this machine instead
   4. If in the meantime, the Primary VM becomes healthy again, the Load Balancer will switch back to it and traffic will only go to the Primary and the Failover VM can be stopped

*Note:*

*This way of doing the failover will have a downtime of approximative 3-5 minutes*

* *The Load Balancer notices that the node is not healthy*
* *The alarm gets fired with 1 min delay because the condition must be maintained for 1 min minimum*
* *The Function is called to start the Failover VM*
* *The Failover VM is started*
* *The Load Balancer sees the new instance as healthy*

*If a failover with almost zero downtime is needed, then the Failover VM should be up-and-running, and the Load Balancer will switch traffic automatically between primary and backup.*