|  |
| --- |
| A picture containing drawing  Description automatically generated  Statement of Direction |
| OCI Compute Failover using OCI Functions  Ionut A. Vladu Senior Domain Specialist Cloud Engineer - OCI 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](#_Toc86318031)

[Versioning 3](#_Toc86318032)

[Introduction 5](#_Toc86318033)

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.

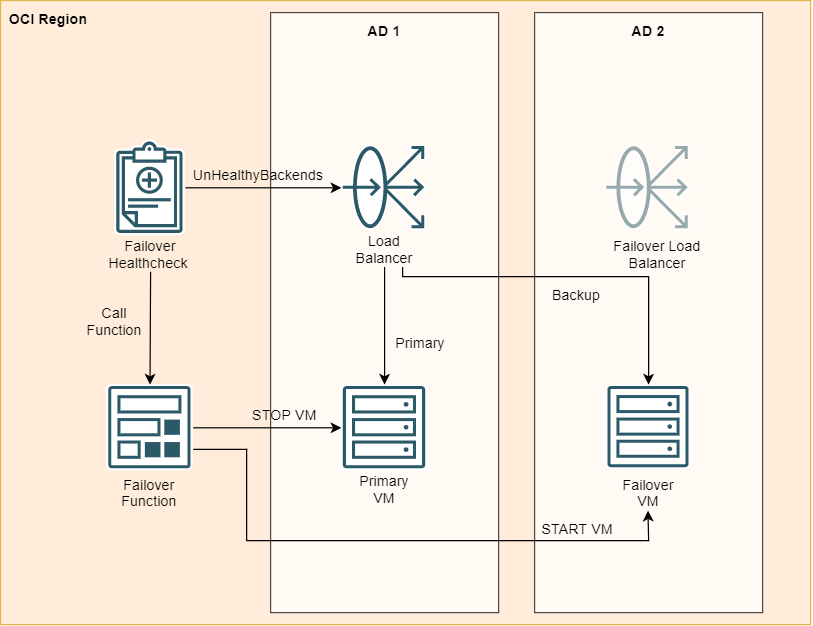
**Use case description:**

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.

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.



LB with 2 backends – VM1 as primary and VM2 as backup

VM2 will be stopped and in critical state

Create alarm on LB UnhealthyBackEnds

If alarm -> Start VM2

LB will automatically switch traffic to VM2 as long as VM1 is unhealthy

2 VMs – one Primary STARTED, one Failover STOPPED

1 LB sending traffic to both

Failover VM set as backup in the LB Backend Set so traffic will be sent only to primary (as long as healthy)

1 Alarm on the LB Backend Set for Unhealthy Backends (fire if there are 2 or more unhealthy backends). The failover one would be in critical state anyway, because the VM is STOPPED

Call Function (and send notif as well on another channel)

The FN will check the state of the primary VM, and if it’s not STOPPED, it will STOP it

Then, the FN will check the state of the failover VM, and if it’s not STARTED, it will START it.

When the PRIMARY VM will be started will become healthy again, the LB will automatically switch the traffic back to it, so you can turn off the failover.

This way of doing the failover will have a downtime of aprox 3-5 minutes (the time the LB notices that the node is not healthy, the time the alarm gets fired (gets fired with 2 min delay because the condition must be maintained for 1min minimum) and the time the fn is called and the vm is started + the time the LB sees the new instance is OK)

Having the failover VM started, would have no downtime at all.