

NAT Instance Configuration

Enabling Internet Access for Private Subnets

ORACLE WHITE PAPER | JANUARY 2018





Table of Contents

Introduction	3
Assumptions	4
Basic NAT Configuration	4
Architecture	5
Configuration	6
Terraform	10
NAT HA and Advanced Concepts	11
HA Architecture	11
Required Resources and Creation Order	12
HA Configuration	14
Conclusion	14
Appendices	15
Appendix 1: Example notify.sh	15
Appendix 2: Example keepalived.conf	16



Introduction

Oracle Cloud Infrastructure enables you to create a virtual cloud network (VCN) that functions as an extended data center in the cloud. The virtual networking primitives offered by the platform give you full flexibility to build a network that meets complex enterprise requirements. You can use any address range for your VCN, segment it into subnets, and configure security lists and route tables. You can connect your VCN to your on-premises network through a dynamic routing gateway (DRG), either through IPsec connections over the internet or through FastConnect over private, dedicated connections.

One of the most common network design requirements is to secure private instances so that they are not accessible from the internet but are accessible only from the on-premises network or bastion hosts in public subnets. You can achieve this requirement by launching the instances in a private subnet or by choosing not to assign a public IP address at launch. However, these backend instances might need access to the internet for specific purposes, such as software updates or CRL verification. You can choose to route this traffic to your on-premises network through your internet gateway, but that might add unwanted latency or cost.

In 1993, the first document about Network Address Translation (NAT) was published. NAT was conceived as a way to reuse address space to prevent IP address exhaustion, but was widely adopted as a way for people to connect their private networks to the public internet.

Currently, NAT is most commonly used as a form of IP masquerade, which allows users to hide an entire IP space behind a single external address. In effect, NAT provide an additional firewall because no unauthorized traffic can enter the private network.

With the recent enhancements to the Oracle Cloud Infrastructure virtual networking platform, you can now enable outbound internet access from private instances by using NAT instances. This white paper describes recommended steps for setting up a NAT instance in your VCN and configuring your private subnet to route internet requests through it.



Assumptions

This white paper is intended for users who want to configure an instance that operates as a NAT gateway for the private network.

To perform a deployment as described in this paper, you should have a solid familiarity with the following items:

- Linux command line
- Provisioning cloud infrastructure via Terraform, a popular and free lightweight deployment tool (<https://www.terraform.io/intro/index.html>)
- Basic understanding of networking protocols

You should also be familiar with the fundamentals of the Oracle Cloud Infrastructure. For information, go to <https://docs.us-phoenix-1.oraclecloud.com/>. If this is the first time that you have used the platform, we recommend specifically the tutorial at <https://docs.us-phoenix-1.oraclecloud.com/Content/GSG/Reference/overviewworkflow.htm>.

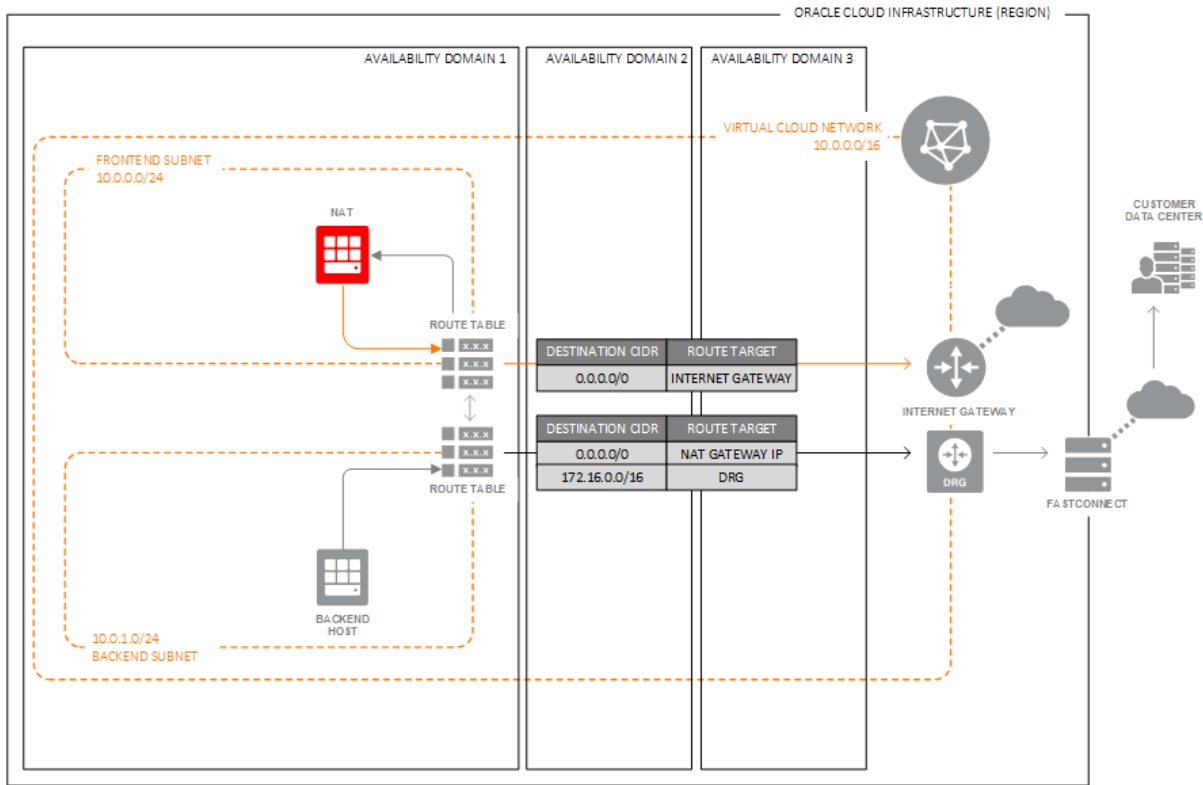
Basic NAT Configuration

The nature of Oracle Cloud Infrastructure Networking makes it unnecessary to physically separate subnets by using virtual interfaces. Because routing is done by underlying software-defined networking, and because security lists exist between the subnets in most of the cases, it's enough to use a single virtual NIC (VNIC). As a result, most of the examples in this paper use a single interface.

If you're using more than one VNIC, ensure that your routing tables correctly point to the gateway IP address in the subnet you're connected to, or disable reverse path filtering in the Linux kernel.

Architecture

The following diagram shows a high-level architecture of the proposed setup.



The diagram shows a VCN (NAT-VCN-1) with two subnets:

- **public** (10.0.0.0/24), with access to internet through an internet gateway. **NAT1** is an instance in the frontend subnet that will function as NAT for the private subnet.
- **private** (10.0.1.0/24), a private subnet with no access to the internet. **Backend Host** represents the instances in the private subnet.

NOTE: Private subnet policy forbids public IP address assignment from the VNIC, which provides an additional layer of security against external access.

Public Subnet Configuration

The route table for the public subnet has a route rule that configures the internet gateway as the route target for all traffic (0.0.0.0/0).

Its security list has an egress rule to allow traffic to all destinations. Ingress rules allow traffic from the backend subnet (and any other address ranges in the VCN).

Private Subnet Configuration

The route table for the private subnet has a route rule that configures the **NAT1** host as the route target for all traffic (0.0.0.0/0).

Its security list has an egress rule to allow traffic to all destinations. Ingress rules allow only specific address ranges (such as an on-premises network or any other backend subnets in the VCN).

If connection with on-premises network is used, then the route toward the enterprise should be set to the configured **DRG**.

With this configuration, when the backend instances initiate outbound internet requests, traffic is routed to the **NAT1** instance. The **NAT1** instance forwards the traffic to the internet through the internet gateway (after applying source NAT). The destination on the internet sees the traffic as sourced from the **NAT1** public IP address. When the NAT instance receives the response from the internet, it forwards the traffic to the backend instance (after applying destination NAT).

Configuration

This section shows an example of basic steps to create the network and a NAT instance.

1. In the Oracle Cloud Infrastructure Console, create a VCN without any resources. The VCN will have a default empty route table, a default security list, and DHCP options. In this example, the VCN is called NAT-VCN-1.
2. Open the VCN that you created. In the Resources panel, click **Security Lists** and then open the default security list. Click **Edit All Rules**.
3. Create an ingress rule. In the **Source CIDR** field, enter **10.0.1.0/24** (which will be your private subnet space), and select **All Protocols**. You can also create a rule allowing SSH protocol.



The screenshot shows a configuration panel for a security rule. On the left, there is a red 'x' icon and a checkbox. The main configuration area includes a 'SOURCE CIDR' text input field containing '10.0.1.0/24', a 'Specified IP addresses: 10.0.1.0-10.0.1.255 (256 IP addresses)' label, and a '(more information)' link. To the right is an 'IP PROTOCOL' dropdown menu set to 'All Protocols' with a '(more information)' link below it. At the bottom left, the rule is identified as 'STATELESS' with a '(more information)' link, and a green note states 'Allows all traffic for all ports'.

4. Create an egress rule that allows **0.0.0.0/0** for **All Protocols**.
5. Click **Save Security List Rules**.

6. In the Resources panel, click **Internet Gateways**, and then click **Create Internet Gateway**.
7. Assign it the name **InternetGateway**, and then click **Create Internet Gateway**.
8. In the Resources panel, click **Route Tables**, and then open the default route table.
9. Create a rule in the default route table with the following values, and then click **Create**:
 - **Destination:** 0.0.0.0/0
 - **Target Type:** Internet Gateway Target
 - **Compartment:** Choose a compartment.
 - **Target Selection:** InternetGateway

Create Route Rule [help](#) [cancel](#)

Route Rule

DESTINATION CIDR BLOCK	TARGET TYPE	TARGET COMPARTMENT	TARGET INTERNET GATEWAY
0.0.0.0/0	Internet Gatew	Sandbox	InternetGateway

Specified IP addresses:
0.0.0.0-255.255.255.255
(4,294,967,296 IP
addresses)

Important: For a route rule that targets a Private IP, you must first enable "Skip Source/Destination Check" on the VNIC that the Private IP is assigned to.

[+](#)

Create

10. In the Resources panel, click **Subnets**, and then click **Create Subnet**.
11. Enter the following values, and then click **Create**:
 - **Name:** Public Subnet
 - **Availability Domain:** Choose from the list.
 - **CIDR Block:** 10.0.0.0/24
 - **Route Table:** Default Route Table NAT-VCN-1
 - **Subnet Access:** Public Subnet
 - **DHCP Options:** Default DHCP Options for NAT-VCN-1
 - **Security Lists:** Default Security for NAT-VCN-1

Before you create the private subnet, create your NAT instance. If you create the NAT instance first, you can choose the OCID (identifier) of the private IP assigned to your NAT instance as the route target in the subnet's route table.

12. From the **Compute** menu, click **Instances** and then click **Launch Instance**. Configure the instance as follows:

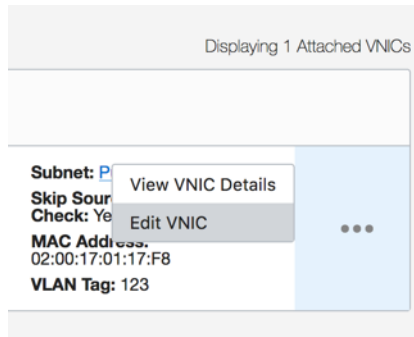
- **Name:** NAT1
- **Availability Domain:** Choose an Availability Domain that you created your subnet in.
- **Image:** Oracle-Linux-7.4-2017.08.24-1 or later
- **Shape:** VM.Standard1.2
- **Virtual Cloud Network:** NAT-VCN-1
- **Subnet:** Public Subnet
- **Private IP Address:** 10.0.0.2
- **Assign public IP address:** Select this check box.
- **SSH Keys:** Upload or paste your SSH key.
- Click **Show Advanced Options**, select the **Paste Cloud-Init Script** option, and paste the following text. Alternatively, you can download the file from https://github.com/oracle/terraform-provider-oci/blob/master/docs/examples/networking/nat/user_data.tpl and use the upload function.

This will configure the server firewall and kernel to provide NAT and routing service for other hosts on the network.

```
#cloud-config
write_files:
# Create file to be used when enabling ip forwarding
- path: /etc/sysctl.d/98-ip-forward.conf
  content: |
    net.ipv4.ip_forward = 1
runcmd:
# Run firewall commands to enable masquerading and port forwarding
# Enable ip forwarding by setting sysctl kernel parameter
- firewall-offline-cmd --direct --add-rule ipv4 nat POSTROUTING 0 -o ens3 -j MASQUERADE
- firewall-offline-cmd --direct --add-rule ipv4 filter FORWARD 0 -i ens3 -j ACCEPT
- /bin/systemctl restart firewalld
- sysctl -p /etc/sysctl.d/98-ip-forward.conf
```

13. Click **Launch Instance**.

- Click the name of your instance, and click **Attached VNICs**. Then click **Edit VNIC**.



- Select **Skip Source/Destination Check** and then click **Update VNIC**.

Note: This step is important. Without it, other instances can't send traffic through the NAT gateway because of security features enabled by default. Trying to configure the route target to the VNIC with a Source/Destination Check results in an error message.

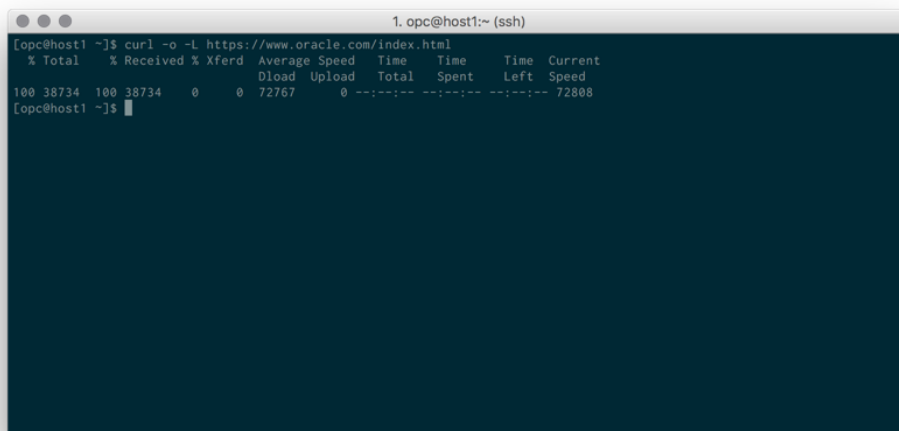
At this point you have created all required resources in the public subnet. The next steps guide you through the process of creating the private network.

- Go back to **Networking > Virtual Cloud Networks > NAT-VCN-1**.
- In the Resources panel, click **Security Lists**, and then click **Create Security List**.
- Enter the following values, and then click **Create Security List**:
 - For the name, enter **Security List for Private Subnet**.
 - Add an egress rule with **0.0.0.0/0** as the destination and **All Protocols** selected.
 - Configure ingress rules according to your needs and security policies. Be sure to add at least one rule that allows SSH traffic.
- In the Resources panel, click **Route Tables**, and then click **Create Route Table**.
- Enter the following values and then click **Create Route Table**:
 - Name:** Private Route
 - Destination:** 0.0.0.0/0
 - Target Type:** Private IP
 - Target Selection:** 10.0.0.2
- In the Resources panel, click **Subnets**, and then click **Create Subnet**.

22. Enter the following values and then click **Create**:

- **Name:** Private Subnet
- **Availability Domain:** Select an Availability Domain.
- **CIDR Block:** 10.0.1.0/24
- **Route Table:** Private Route
- **Subnet Access:** Private subnet
- **DHCP Options:** Default DHCP Options for NAT-VCN-1
- **Security Lists:** Security List for Private Subnet

Now you can launch the host instances by using any operating system in the private subnet. They can connect to the internet without a public IP address assigned, and no connections originated on the internet are possible directly to your server. To manage your private hosts, you need to use SSH to connect to the NAT instance first, or use a CPE VPN connection from your own network.



```
1. opc@host1:~ (ssh)
[opc@host1 ~]$ curl -o -L https://www.oracle.com/index.html
% Total % Received % Xferd Average Speed Time Time Time Current
         Dload Upload Total Spent Left Speed
100 38734 100 38734 0 0 72767 0 --:--:-- --:--:-- --:--:-- 72868
[opc@host1 ~]$
```

Terraform

Terraform is a tool for building, changing, and versioning infrastructure. It generates an execution plan from configuration files, describing what it will do to reach the necessary state, and then executes changes to build an infrastructure. For basic information about Terraform, see the following sites:

- <https://github.com/oracle/terraform-provider-oci>
- <https://community.oracle.com/community/oracle-cloud/cloud-infrastructure/blog/2017/02/15/terraform-and-oracle-bare-metal-cloud-services>

The NAT example configuration is located in the **examples** folder of the **terraform-provider-oci** Git repository at <https://github.com/oracle/terraform-provider-oci/tree/master/docs/examples/networking/nat>.

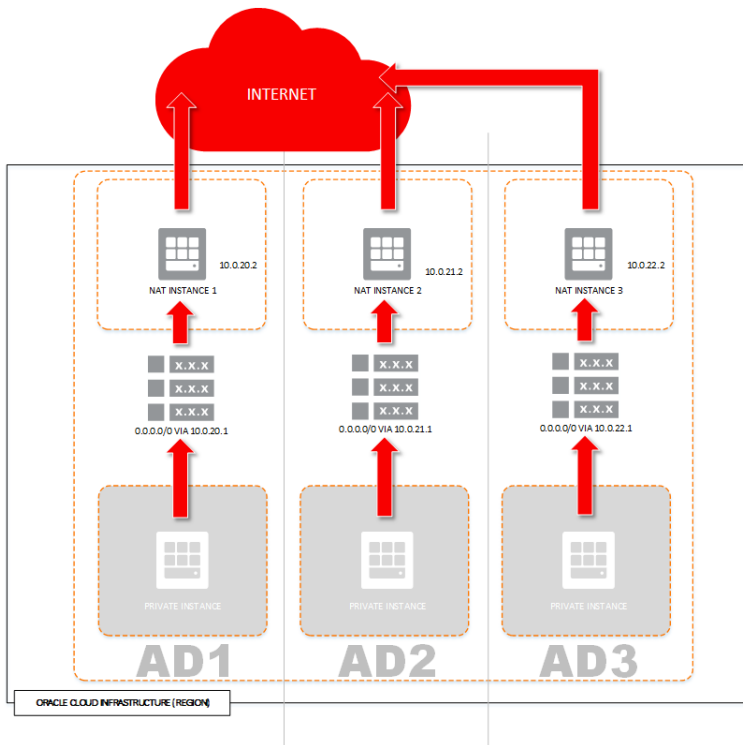
NAT HA and Advanced Concepts

When a deployment uses multiple Availability Domains, you must maintain some form of heartbeat and failover mechanism. In the typical non-cloud deployment, this can be achieved by using Layer 2 paths and IP failover. Because of the nature of the Oracle Cloud Infrastructure Networking, the best way to achieve redundancy is to maintain route-table configuration directly.

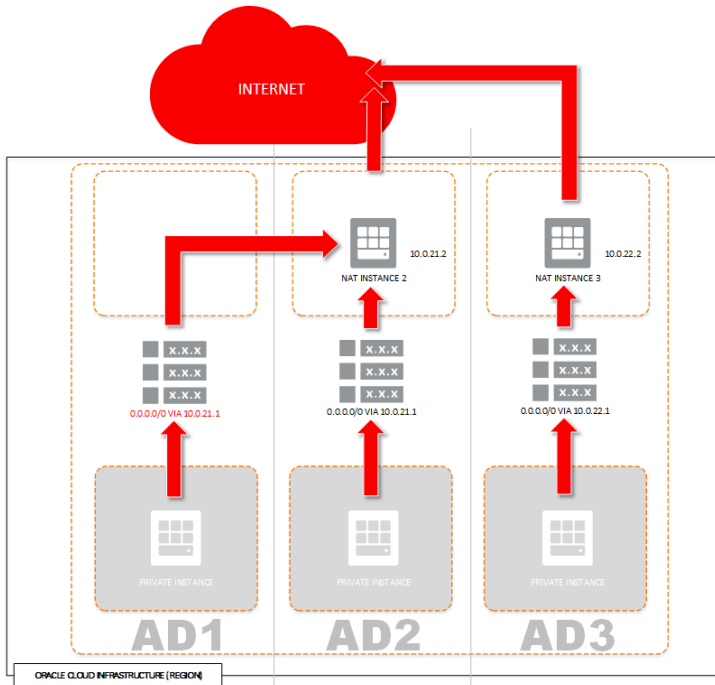
HA Architecture

The following diagram shows three Availability Domains with NAT instances and hosts in the private subnet.

In normal conditions, traffic stays within the Availability Domain and is translated on the NAT gateway in the public subnet.



In the case of a NAT gateway failure, traffic should fail over to the NAT gateway in the other Availability Domain as shown in the following diagram:



Because NAT Instance 1 is not available, NAT Instance 2 takes over and modifies the route table of the private subnet in AD1. The default gateway is configured for a private IP of NAT Instance 2 as a route target, using the unique identifier of the private IP (OCID). This ensures that the private instance can still reach the internet even when an instance fails.

Required Resources and Creation Order

The following table summarizes the resources needed for the preceding setup. You can use it as a deployment checklist.

VCN	Dependencies		
Internet Gateway	Depends on the VCN		
Public Route Table AD1	Depends on the VCN		
Public Route Table AD2	Depends on the VCN		
Public Route Table AD3	Depends on the VCN		
Private Route Table AD1	Depends on the VCN		
Private Route Table AD2	Depends on the VCN		

VCN	Dependencies		
Private Route Table AD3	Depends on the VCN		
Public Security List	Depends on the VCN		
Private Security List	Depends on the VCN		
Public Subnet AD1	Depends on the VCN	Depends on the Route Table	Depends on the Security List
Public Subnet AD2	Depends on the VCN	Depends on the Route Table	Depends on the Security List
Public Subnet AD3	Depends on the VCN	Depends on the Route Table	Depends on the Security List
Private Subnet AD1	Depends on the VCN	Depends on the Route Table	Depends on the Security List
Private Subnet AD2	Depends on the VCN	Depends on the Route Table	Depends on the Security List
Private Subnet AD3	Depends on the VCN	Depends on the Route Table	Depends on the Security List
NAT Instance AD1	Depends on the Subnet		
NAT Instance AD2	Depends on the Subnet		
NAT Instance AD3	Depends on the Subnet		
Private Instance AD1	Depends on the Subnet		
Private Instance AD2	Depends on the Subnet		
Private Instance AD3	Depends on the Subnet		
Route Rules for Public subnets	Depends on the Internet Gateway		
Route Rules for Private subnets	NAT Instance		

Note that for route rule creation, the NAT instance must be launched first because the route to the target rule requires an existing private IP object that is attached to the NAT instance VNIC.



HA Configuration

For HA in this example, a keepalived daemon and a custom script that performs the failover are used. In addition, NAT instances must have an Oracle Cloud Infrastructure CLI, API key, user ID, and relevant policy to manage `virtual-network-family` within the compartment. For example:

```
Allow group NAT to manage virtual-network-family in compartment id  
aaa.compartment.ocid
```

Failover Script

The failover script processes the metadata information and performs route rule changes in the designated route table. The active node within the VRRP group points the route rule to its private IP OCID.

See Appendix 1 for the script example.

Keepalived.conf

For VRRP configuration, keepalived needs to be configured for three different instances. Each of the NAT instances will be the MASTER node for the Availability Domain, and the two remaining ones will be in the BACKUP state. Ensure that each VRRP instance has a different `virtual_router_id` parameter and that the MASTER node has a higher priority.

See Appendix 2 for the configuration example.

For each node, be sure to update the priority within the correct Availability Domain, initial state, and IP addresses of the nodes.

Conclusion

A NAT instance can be used to protect important resources in your cloud data center and provide services for the hosts located on the private subnets. Servers located in the private subnet are protected from being accessible from outside of your VCN, but can still reach out to the internet.

Appendices

Appendix 1: Example notify.sh

```
#!/bin/bash

STATUS=$1
AD=$2
RT_TABLE_ID=$3

CURL="/usr/bin/curl -s"
OCI=$( /usr/bin/oci)
MDS="http://169.254.169.254/opc/v1"
INSTANCE_MDS=$MDS"/instance"
VNIC_MDS=$MDS"/vnics"

PATH="/usr/libexec/keepalived"

PUB_SN_ID=$( $CURL "$INSTANCE_MDS/metadata/subnet_id")
PRIV_SN_ID=$( $CURL "$INSTANCE_MDS/metadata/private_subnet_id")
VNIC_ID=$( $CURL "$VNIC_MDS/0/vnicId")
echo $VNIC_MDS
echo $VNIC_ID

echo $STATUS > $PATH/$AD"_status.txt"

$OCI network private-ip list --subnet-id $PUB_SN_ID --vnic-id $VNIC_ID |
/usr/bin/python -c 'import sys, json; print
json.load(sys.stdin)["data"][0]["id"]' > $PATH"/private_ip.ocid"

case "$STATUS" in
  "master") echo "master"
    if [ ! -z $RT_TABLE_ID ]; then
      PRIVATE_IP_ID=$( /bin/cat $PATH"/private_ip.ocid")
      $OCI network route-table update --force --rt-id $RT_TABLE_ID --route-rules
' [{"cidrBlock": "0.0.0.0/0", "networkEntityId": "'$PRIVATE_IP_ID'"} ]'
      fi
    ;;
  "backup") echo "Backup Status"
    logger "Backup Status"
    exit 0
    ;;
  "stop") echo "Keepalived stopped"
    logger "Keepalived stopped"
    exit 0
    ;;
  "fault") echo "Keepalived fault!"
    logger "Keepalived fault!"
    exit 0
    ;;
  *) exit 1
    ;;
esac
```

Appendix 2: Example keepalived.conf

Replace `${variable}` with real values.

```
vrrp instance VI_1 {
    interface ens3
    state MASTER

    virtual_router_id 51
    priority ${priority_map_1}

    unicast_src_ip ${private_ip}
    unicast_peer {
        ${peer_ip}
        ${peer2_ip}
    }

    notify_master "/usr/libexec/keepalived/notify.sh master ad1 ${ad1_rt_id}"
    notify_backup "/usr/libexec/keepalived/notify.sh backup ad1 ${ad1_rt_id}"
    notify_fault "/usr/libexec/keepalived/notify.sh fault ad1 ${ad1_rt_id}"
    notify_stop "/usr/libexec/keepalived/notify.sh stop ad1 ${ad1_rt_id}"
}

vrrp_instance VI_2 {
    interface ens3
    state BACKUP

    virtual_router_id 52
    priority ${priority_map_2}

    unicast_src_ip ${private_ip}
    unicast_peer {
        ${peer_ip}
        ${peer2_ip}
    }

    notify_master "/usr/libexec/keepalived/notify.sh master ad2 ${ad2_rt_id}"
    notify_backup "/usr/libexec/keepalived/notify.sh backup ad2 ${ad2_rt_id}"
    notify_fault "/usr/libexec/keepalived/notify.sh fault ad2 ${ad2_rt_id}"
    notify_stop "/usr/libexec/keepalived/notify.sh stop ad2 ${ad2_rt_id}"
}

vrrp instance VI_3 {
    interface ens3
    state BACKUP

    virtual_router_id 53
    priority ${priority_map_3}

    unicast_src_ip ${private_ip}
    unicast_peer {
        ${peer_ip}
        ${peer2_ip}
    }

    notify_master "/usr/libexec/keepalived/notify.sh master ad3 ${ad3_rt_id}"
    notify_backup "/usr/libexec/keepalived/notify.sh backup ad3 ${ad3_rt_id}"
    notify_fault "/usr/libexec/keepalived/notify.sh fault ad3 ${ad3_rt_id}"
    notify_stop "/usr/libexec/keepalived/notify.sh stop ad3 ${ad3_rt_id}"
}
```








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
-  facebook.com/oracle
-  twitter.com/oracle
-  oracle.com

Integrated Cloud Applications & Platform Services

Copyright © 2018, 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. 0218

NAT Instance Configuration: Enabling Internet Access for Private Subnets
February 2018

