2025

OCI-Terraform-Project File

OCI PROJECT – SECURE WEB APPLICATION DEPLOYMENT

ESRAA ALHARBI

Project Overview

This project involves designing and deploying a secure, available web application environment on Oracle Cloud Infrastructure (OCI. The solution adheres to OCI security best practices, uses Terraform for automation, and is implemented in a dedicated compartment with restricted access.

Table of Contents

Table of Figures

Architectural Diagram

The following diagram illustrates the overall architecture of the deployed environment on OCI. It shows the VCN, subnets (public and private), compute instances, file storage, load balancer, and security components such as WAF. This design ensures high availability, secure access, and optimized traffic distribution across backend servers.

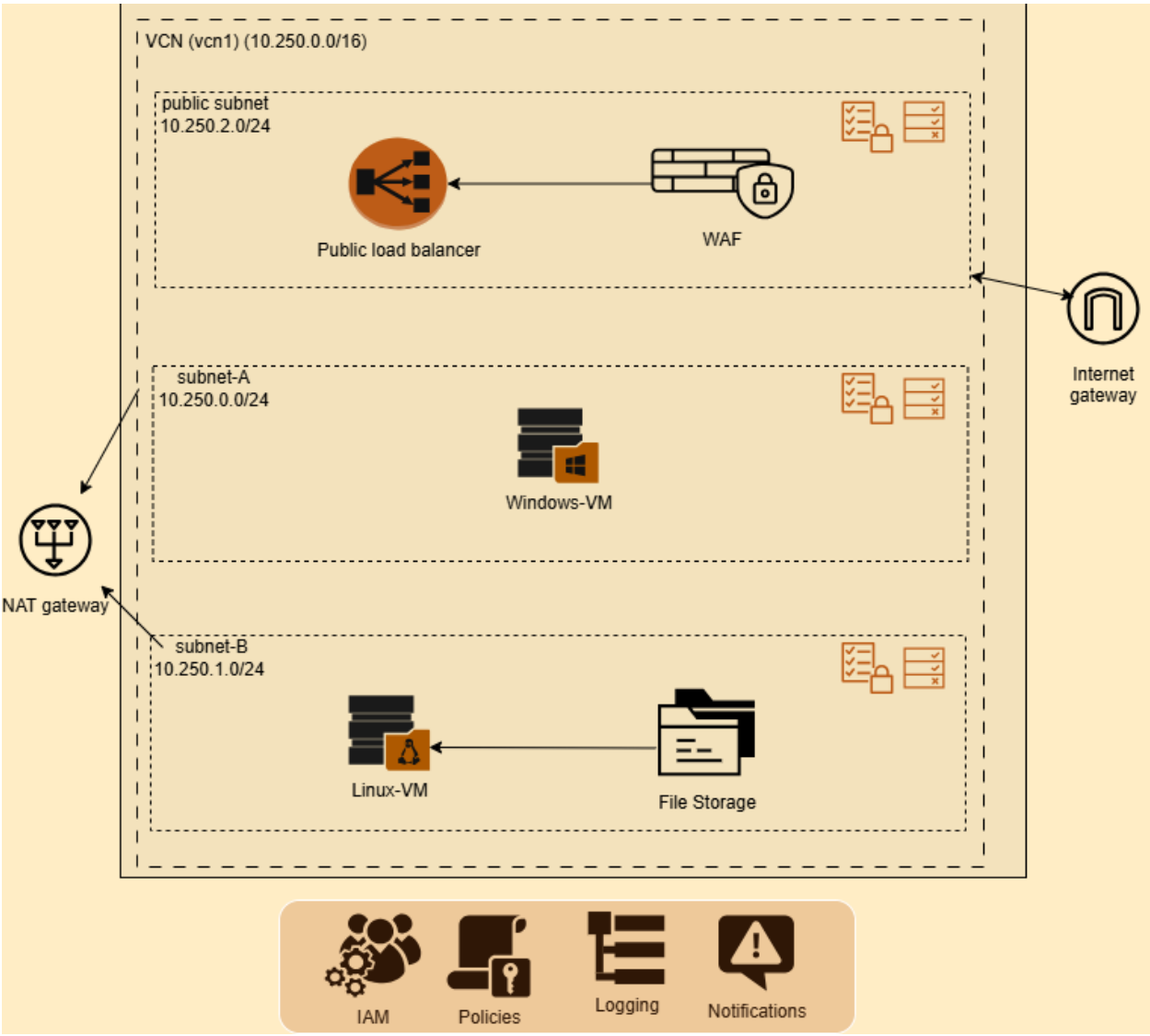


Figure 1:OCI Secure Web Application Architecture Diagram

Compartment

A dedicated compartment named **Training-Project** was created using Terraform to isolate resources and enforce access control.

```
resource "oci_identity_compartment" "Training-Project" {
  compartment_id = var.tenancy_ocid
  name=var.Training-Project_name
  description =var.Training-Project_description

 freeform_tags = {
    "Environment" = "Dev"
    "Owner"       = "Esra"
    "Project"     = "OCI Training"
  }

}
```

Figure 2:Compartment created via Terraform.

Networking

A **VCN Configuration:** Created a Virtual Cloud Network (**vcn1**) with CIDR block 10.250.0.0/16.

Subnets:

- Private Subnet A: 10.250.0.0/24
- Private Subnet B: 10.250.1.0/24
- Public Subnet (pubSN): 10.250.2.0/24

Gateways:

- NAT Gateway for private subnets
- Internet Gateway for public subnet

Security Lists: Configured custom security rules for each subnet.

All networking resources were provisioned using Terraform modules for consistency.

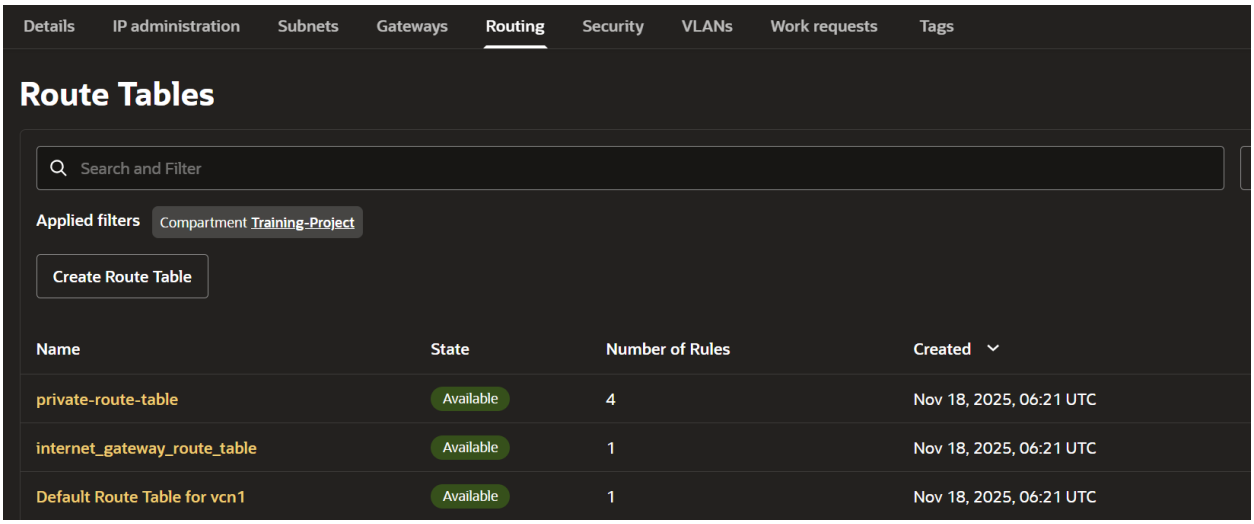


Figure 3: Routing Network

Compute

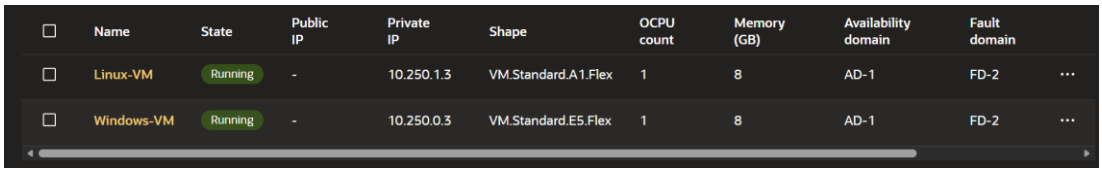
Windows VM:

- Subnet: Private Subnet A
- IP: 10.250.0.3
- Shape: VM.Standard.E5.Flex

Linux VM:

- Subnet: Private Subnet B
- IP: 10.250.1.3
- Shape: VM.Standard.A1.Flex
- SSH key configured

Both instances have **encryption in transit enabled** and were provisioned using Terraform.



Name	State	Public IP	Private IP	Shape	OCPU count	Memory (GB)	Availability domain	Fault domain
Linux-VM	Running	-	10.250.1.3	VM.Standard.A1.Flex	1	8	AD-1	FD-2
Windows-VM	Running	-	10.250.0.3	VM.Standard.E5.Flex	1	8	AD-1	FD-2

Figure 4:Compute Instances Overview

```
34    freeform_tags = {
35      APP_OWNER   = var.APP_OWNER
36      Environment = var.Environment
37    }

163     APP_OWNER="Esra"
164     Environment="Dev"
```

Figure 5:Added Free Tags to Instances.

Access to servers and Web Tier

Nginx web server

Installed **Nginx web server on Linux**, configured index.html with “Welcome” ,opened ports 80/443.

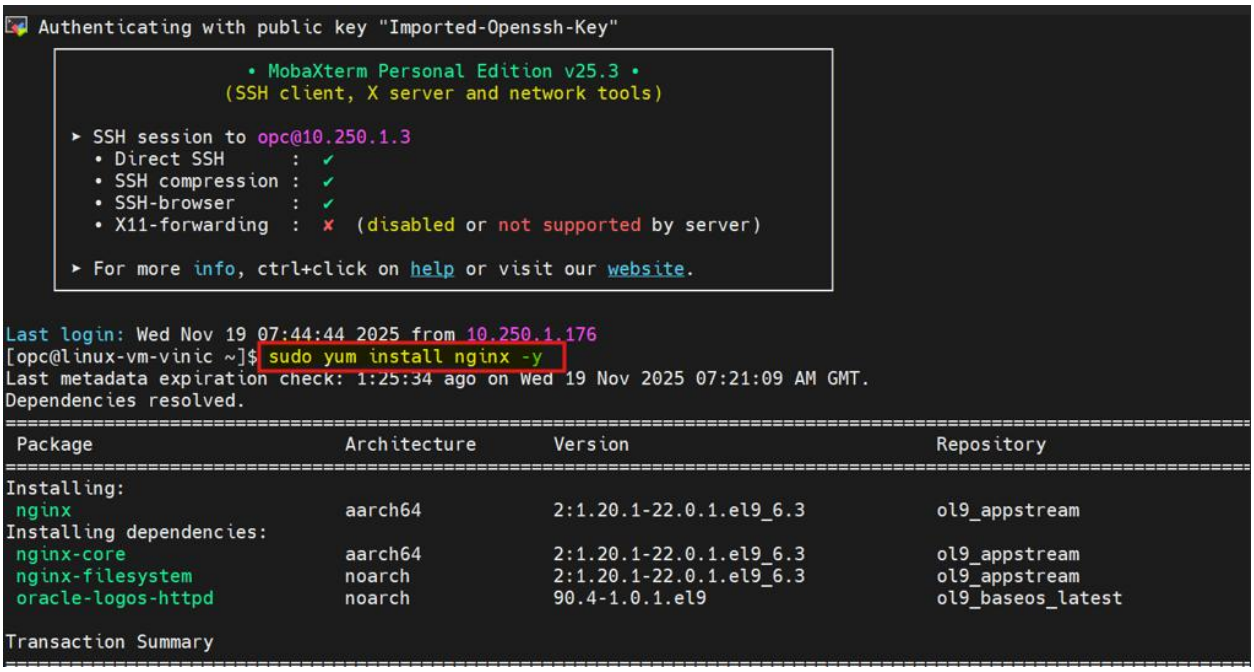


Figure 6:Installed Nginx on Linux

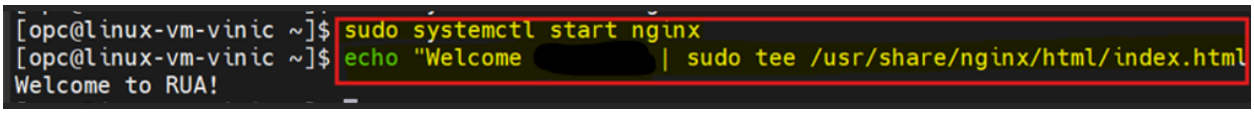


Figure 7:index.html commands

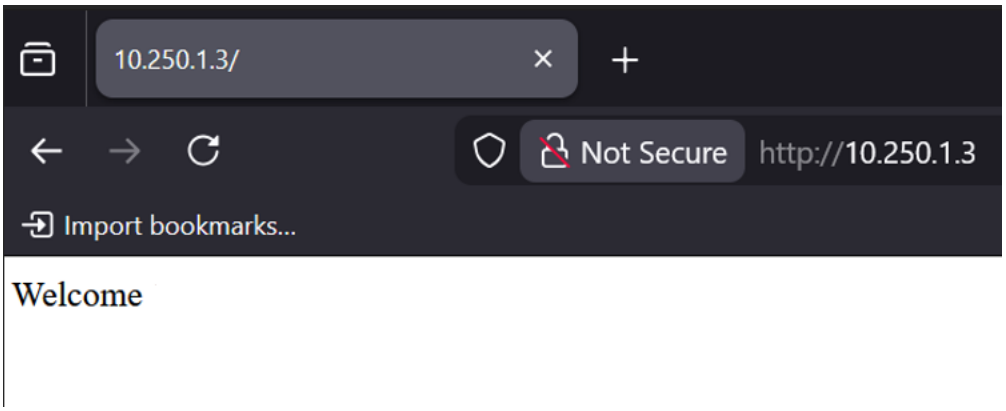


Figure 8:Web App Result.

IIS web server

On Windows VM Joined RUA domain, configured DNS, enabled RDP users, installed IIS web server, opened the port 80 and 443 in windows defender firewall, created new website called my Site and deployed “Welcome to RUA!” page.

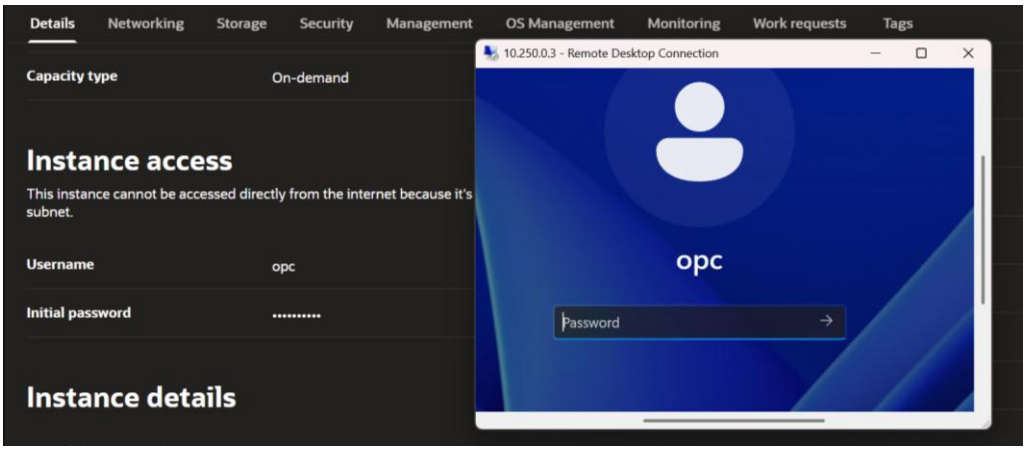


Figure 9: Accessed the Windows Instance

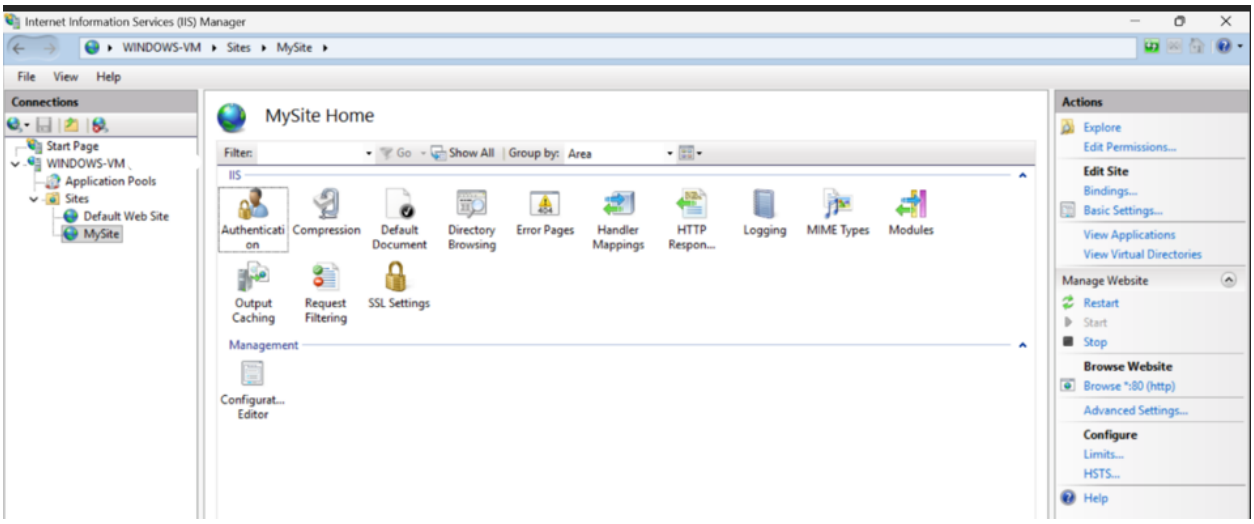


Figure 10:Installed IIS on Windows

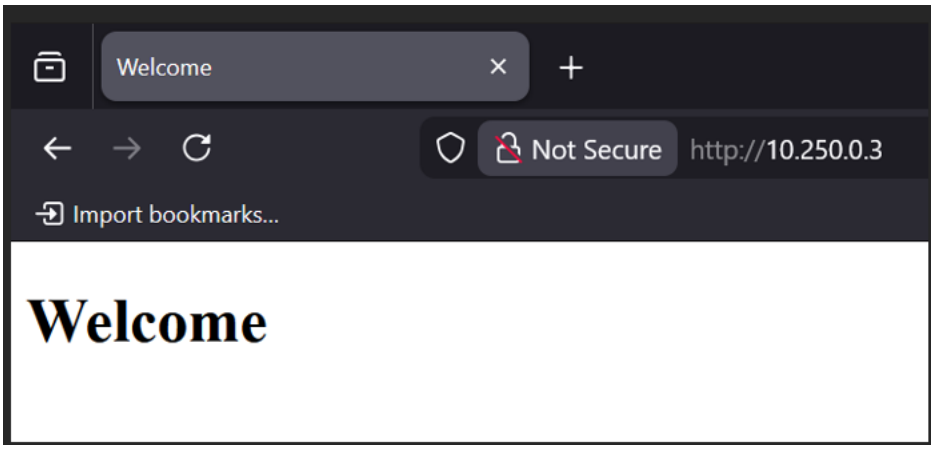


Figure 11:Web App Result.

File Storage

Created File Storage in Private Subnet B.

Configured policies and security lists, installed NFS on Linux VM,created directory to mounted storage at 10.250.1.238.

Placed the web content under mounted folder, Updated Nginx root directory to point to mounted storage and adjusted permissions.

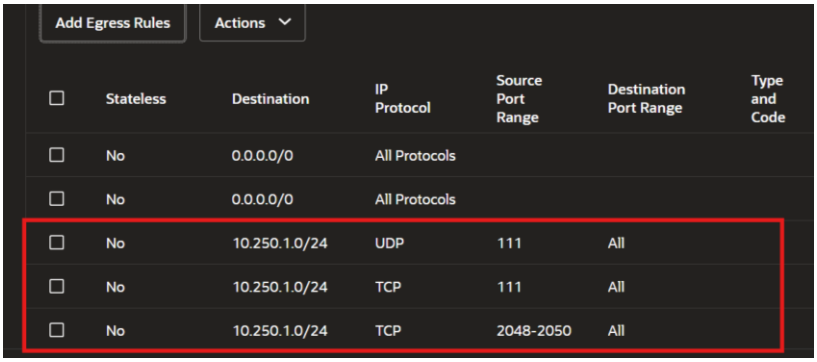
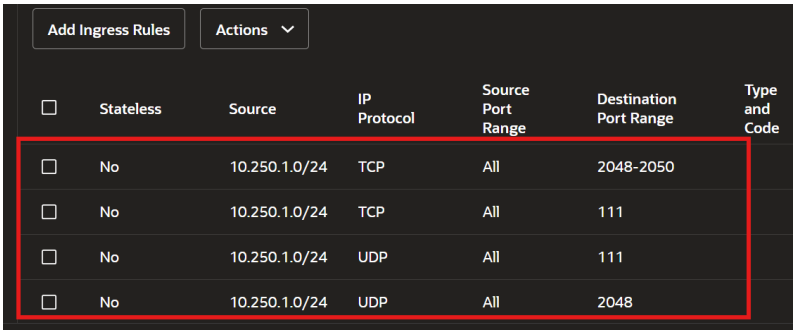


Figure 12:Configured Egress Security Lists.



Add Ingress Rules | Actions

	Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code
☐	No	10.250.1.0/24	TCP	All	2048-2050	
☐	No	10.250.1.0/24	TCP	All	111	
☐	No	10.250.1.0/24	UDP	All	111	
☐	No	10.250.1.0/24	UDP	All	2048	

Figure 13:Configured Ingress Security Lists.

```
[opc@linux-vm-vinic ~]$ sudo mkdir -p /mnt/FileSystemTraining
[opc@linux-vm-vinic ~]$ sudo mount 10.250.1.238:/FileSystemTraining /mnt/FileSystemTrainin
Created symlink /run/systemd/system/remote-fs.target.wants/rpc-statd.service → /usr/lib/sy
.
[opc@linux-vm-vinic ~]$ df -h
Filesystem                      Size  Used Avail Use% Mounted on
devtmpfs                        4.0M     0  4.0M   0% /dev
tmpfs                           3.4G     0  3.4G   0% /dev/shm
tmpfs                           1.4G  111M  1.3G   9% /run
efivarfs                        256K   14K  243K   6% /sys/firmware/efi/efivars
/dev/mapper/ocivolume-root       30G  6.7G   23G  23% /
/dev/sda2                       2.0G  392M  1.6G  20% /boot
/dev/mapper/ocivolume-oled       15G  799M   15G   6% /var/oled
/dev/sda1                       100M  7.5M   93M   8% /boot/efi
tmpfs                           689M     0  689M   0% /run/user/989
tmpfs                           689M     0  689M   0% /run/user/1000
10.250.1.238:/FileSystemTraining  8.0E     0  8.0E   0% /mnt/FileSystemTraining
[opc@linux-vm-vinic ~]$ ^C
```

Figure 14: Mounted File Storage to Linux VM

```
[opc@linux-vm-vinic ~]$ sudo cp /usr/share/nginx/html/index.html /mnt/FileSystemTraining/
[opc@linux-vm-vinic ~]$ ls -l /mnt/FileSystemTraining/
total 8
-rw-r--r--. 1 root root 16 Nov 26 09:13 index.html
```

Figure 15:Placed the web content under mounted folder.

```
[opc@linux-vm-vinic ~]$ sudo -i
[root@linux-vm-vinic ~]# cd /etc/nginx
[root@linux-vm-vinic nginx]# ls
conf.d        fastcgi.conf.default    koi-utf     mime.types.default  scgi_params          uwsgi_param
default.d     fastcgi_params          koi-win     nginx.conf          scgi_params.default  win-utf
fastcgi.conf  fastcgi_params.default  mime.types  nginx.conf.default  uwsgi_params
[root@linux-vm-vinic nginx]#
[root@linux-vm-vinic nginx]# vim nginx.conf
[root@linux-vm-vinic nginx]# systemctl reload nginx.service
```

Figure 16: Accessed Nginx Configuration File.

```
include             /etc/nginx/mime.types;
default_type        application/octet-stream;

# Load modular configuration files from the /etc/nginx/conf.d directory.
# See http://nginx.org/en/docs/ngx_core_module.html#include
# for more information.
include /etc/nginx/conf.d/*.conf;

server {
    listen       80;
    listen       [::]:80;
    server_name  ;
    root     /mnt/FileSystemTraining;
    index index.html;
    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
```

Figure 17:Updated Nginx root directory.

Load Balancer

Deployed Public Load Balancer (public Ip: 80.225.64.77) in pubSN with backend set (Linux & Windows VMs) on port 80.

Configured:

- WAF Policy for security
- Weighted Round Robin load balancing.
- Health checks
- TLS using certificate.

Figure 18:WAF policy.

Figure 19:Published IIS on load balancer.

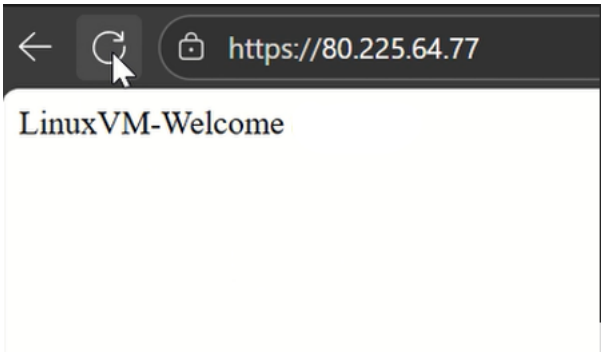


Figure 20:Published Nginx on load balancer.

Security

Applied Geo-Restriction via WAF to allow only KSA traffic:

!i_contains(['SA'], connection.source.geo.countryCode)

No public IPs assigned to VMs for added security.
Implemented **(IAM)** policies:

Policy 1:

Allow group OCI_Administrators to manage all-resources in compartment Training-Project

Policy 2:

Allow group OCI_Administrators to manage file-family in compartment Training-Project.

Figure 21: Access Control Rule for KSA Only.

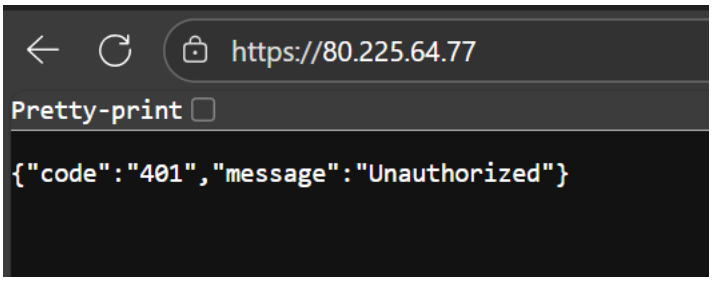


Figure 22: Load Balancer Access from Other Countries.

Monitoring

Enabled logging and created two alarms:

- **Compute Alarm:** Triggers when CpuUtilization > 80% for 1 minute.
- **Load Balancer Alarm:** Triggers when UnHealthyBackendServers > 1 for 1 minute.

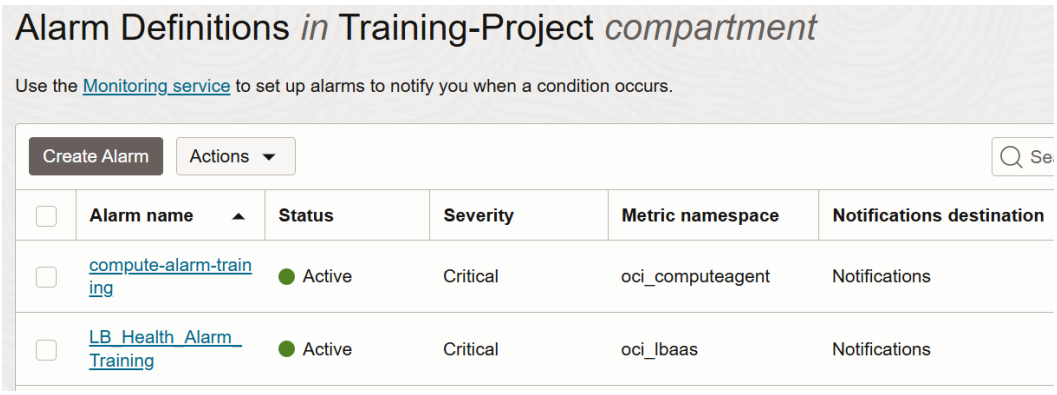


Figure 23:List of alarms.

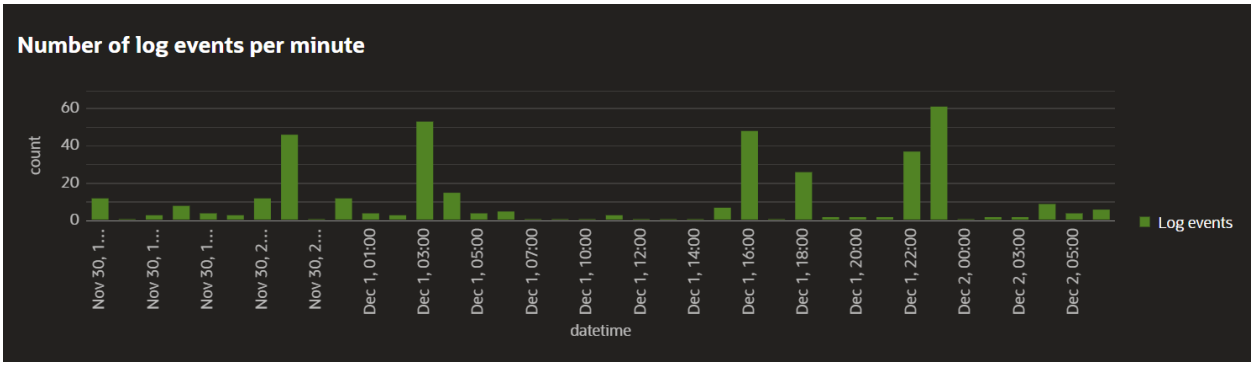


Figure 24:Access logs.

datetime	type	data.message
Dec 02, 2025, 07:59:04	loadbalancer.error	{"data":{"errorLog":{"errorDetails":"Client 64.23.208.64 sent plain HTTP request to HTTPS port","type":"sslClient"},"tir
Dec 02, 2025, 07:59:02	loadbalancer.error	{"data":{"errorLog":{"errorDetails":"Client 64.23.208.64 sent plain HTTP request to HTTPS port","type":"sslClient"},"tir
Dec 02, 2025, 07:17:40	loadbalancer.error	{"data":{"errorLog":{"errorDetails":"Client 192.159.99.101 sent plain HTTP request to HTTPS port","type":"sslClient"},
Dec 02, 2025, 06:18:19	loadbalancer.error	{"data":{"errorLog":{"errorDetails":"healthStatus: Healthy, backendSetName: bs_lb_Training, backend: 10.250.0.3:80, (
Dec 02, 2025, 06:18:17	loadbalancer.error	{"data":{"errorLog":{"errorDetails":"healthStatus: Healthy, backendSetName: bs_lb_Training, backend: 10.250.0.3:80, (
Dec 02, 2025, 02:20:34	loadbalancer.error	{"data":{"errorLog":{"errorDetails":"Client 85.11.183.6 sent plain HTTP request to HTTPS port","type":"sslClient"},"tim
Dec 02, 2025, 00:30:10	loadbalancer.error	{"data":{"errorLog":{"errorDetails":"SSL handshake failed for 128.203.201.208","type":"ssl"},"timestamp":"2025-12-0

Figure 25:Error logs.