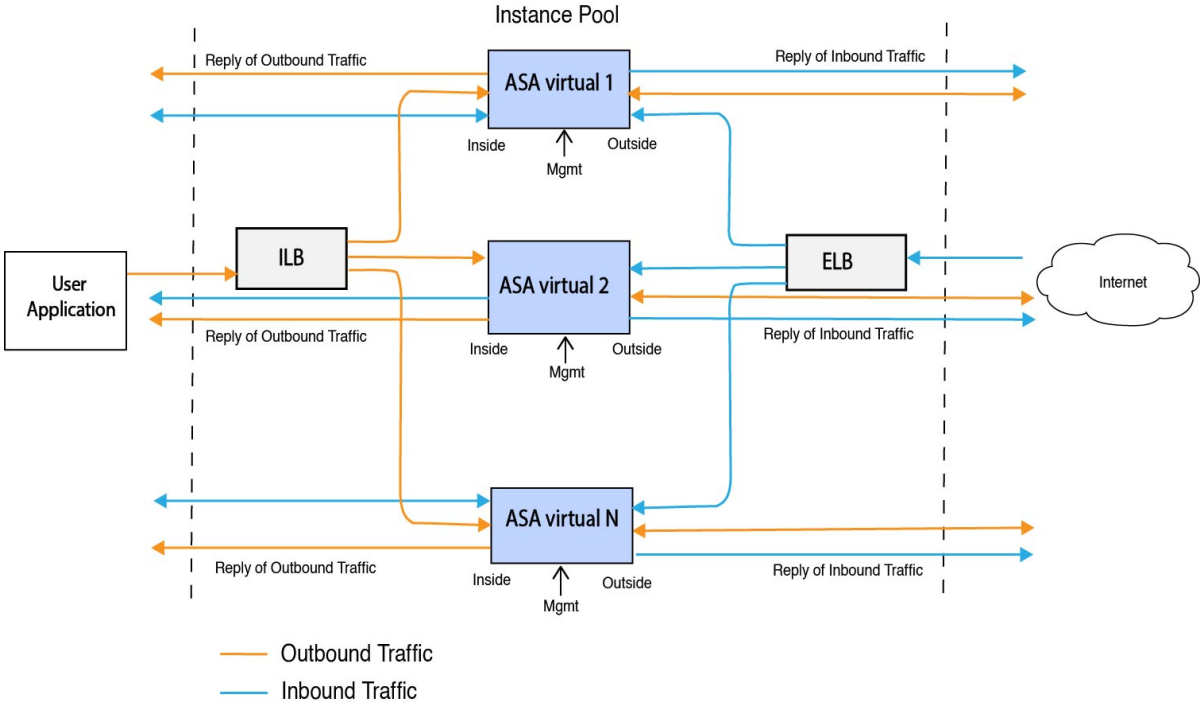
OCI에 ASA 가상 Auto Scale 솔루션 구축

Autoscale 사용 사례

ASA 가상의 사용 사례 – OCI Autoscale 솔루션은 사용 사례 다이어그램에 나와 있습니다. 인터넷 연결 로드 밸런서에는 리스너 및 대상 그룹 조합을 사용하여 활성화된 포트가 있는 공용 IP 주소가 있습니다.

그림 *1:* 사용 사례 다이어그램



포트 기반 분기는 네트워크 트래픽을 대상으로 구현할 수 있습니다. 이 작업은 NAT 규칙을 통해 수행할 수 있습니다. 이 컨피그레이션 예는 다음 섹션에서 설명합니다.

사전 요구 사항

권한 및 정책

다음은 솔루션을 구현하는 데 필요한 OCI 권한 및 정책입니다.

1. 사용자 및 그룹

참고 사용자 및 그룹을 생성하려면 OCI 사용자 또는 테넌시 관리자여야 합니다.

Oracle Cloud Infrastructure 사용자 계정 및 사용자 계정이 속한 그룹을 생성합니다. 사용자 계정이 있는 관련 그룹이 존재한다면 생성하지 않아도 됩니다. 사용자 및 그룹 생성 지침은 그룹 및 사용자 생성을 참고하십시오.

2. 그룹 정책

정책을 생성한 다음 그룹에 매핑해야 합니다. 정책을 생성하려면 **OCI** > **Identity & Security(ID** 및 보안**)** > **Policies(**정책**)** > **Create Policy(**정책 생성**)**로 이동합니다. 다음 정책을 생성하고 원하는 그룹에 추가합니다.

- <*Group_Name*> 그룹을 허용하여 컴파트먼트 <*Compartment_Name*>에서 메트릭 사용
- <*Group_Name*>그룹을 허용하여 컴파트먼트 <*Compartment_Name*>에서 알람 관리
- <*Group_Name*> 그룹을 허용하여 컴파트먼트 <*Compartment_Name*>에서 ons-topics 관리
- <*Group_Name*> 그룹을 허용하여 컴파트먼트 <*Compartment_Name*>에서 메트릭 검사
- <*Group_Name*> 그룹을 허용하여 컴파트먼트 <*Compartment_Name*>에서 메트릭 읽기
- <*Group_Name*> 그룹을 허용하여 컴파트먼트 <*Compartment_Name*>에서 태그 이름 공간 (tag-namespace) 사용
- <*Group_Name*> 그룹을 허용하여 컴파트먼트 <*Compartment_Name*>에서 로그 그룹 읽기
- <*Group_Name*>그룹을 허용하여 컴파트먼트 <*Compartment_Name*>에서 인스턴스 풀 (instance-pool) 사용
- <*Group_Name*> 그룹을 허용하여 테넌시에서 클라우드 쉘(cloud-shell) 사용
- <*Group_Name*> 그룹을 허용하여 테넌시에서 objectstorage-namespace 읽기
- <*Group_Name*> 그룹을 허용하여 테넌시에서 리포지토리 관리

참고 테넌시 레벨에서 정책을 생성할 수도 있습니다. 모든 권한을 제공하는 방법은 사용자가 결정합니다.

3. **Oracle Functions**에 대한 권한

   Oracle-Function이 다른 Oracle Cloud Infrastructure 리소스에 액세스할 수 있게 하려면, 동적 그룹에 기능을 포함한 다음 동적 그룹에 리소스에 대한 액세스 권한을 부여하는 정책을 만듭니다.

4. 동적 그룹 생성

   동적 그룹을 만들려면 **OCI** > **Identity & Security(ID** 및 보안**)** > **Dynamic Group(**동적 그룹**)** > **Create Dynamic Group(**동적 그룹 생성**)**으로 이동합니다.

   동적 그룹을 생성하는 동안 다음 규칙을 지정합니다.

   ```
   모든 {resource.type = 'fnfunc', resource.compartment.id = '<Your_Compartment_OCID>'}
   ```

   동적 그룹에 대한 자세한 내용은 다음을 참조하십시오.

   - https://docs.oracle.com/en-us/iaas/Content/Functions/Tasks/functionsaccessingociresources.htm
   - https://docs.oracle.com/en-us/iaas/Content/Identity/Tasks/managingdynamicgroups.htm

5. 동적 그룹에 대한 정책 생성

   정책을 추가하려면 **OCI** > **Identity & Security(ID** 및 보안**)** > **Policies(**정책**)** > **Create Policy(**정책 생성**)**로 이동합니다. 그룹에 다음 정책을 추가합니다.

   ```
   <Dynamic_Group_Name> 다이내믹 그룹을 허용하여 컴파트먼트<Compartment_OCID>에서 모든 리소스 관리
   ```

GitHub에서 파일 다운로드

ASA 가상 – OCI Autoscale 솔루션은 GitHub 리포지토리로서 전달됩니다. 리포지토리에서 파일을 가져오거나 다운로드할 수 있습니다.

Python3 환경

make.py 파일은 복제된 리포지토리에 있습니다. 이 프로그램은 Oracle 함수와 템플릿 파일을 Zip 파일로 압축합니다. 이 파일을 대상 폴더에 복사합니다. 이러한 작업을 수행하려면 Python 3 환경이 구성되어야 합니다.

참고 이 python 스크립트는 Linux 환경에서만 사용할 수 있습니다.

인프라 구성

다음을 구성해야 합니다.

1. **VCN**

   ASA 가상 애플리케이션의 필요에 맞게 VCN을 만듭니다. 인터넷에 대한 경로가 연결된 서브넷이 하나 이상 있는 인터넷 게이트웨이를 이용해 VCN을 만듭니다.

   VCN 생성에 대한 자세한 내용은 https://docs.oracle.com/en-us/iaas/Content/GSG/Tasks/creatingnetwork.htm을 참조하십시오.

2. 애플리케이션 서브넷

   ASA 가상 애플리케이션의 필요에 맞게 서브넷을 만듭니다. 이 사용 사례에 맞는 솔루션을 구현하려면 ASA 가상 인스턴스에 3개의 서브넷이 필요합니다.

   서브넷 생성에 대한 자세한 내용은 https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/managingVCNs_topic-Overview_of_VCNs_and_Subnets.htm#을 참조하십시오.

3. 외부 서브넷

   서브넷은 인터넷 게이트웨이에 대한 기본 경로가 '0.0.0.0/0'이어야 합니다. 이 서브넷에는 Cisco ASA 가상 및 인터넷 연결 로드 밸런서의 외부 인터페이스가 포함되어 있습니다. 아웃바운드 트래픽에 NAT 게이트웨이가 추가되었는지 확인합니다.

   자세한 내용은 다음 문서를 참조하십시오.

   - https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/managingIGs.htm
   - https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/NATgateway.htm#To_create_a_NAT_gateway

4. 내부 서브넷

   NAT/인터넷 게이트웨이가 있거나 없는 애플리케이션 서브넷과 유사할 수 있습니다.

참고 ASA 가상 상태 프로브의 경우 포트 80을 통해 메타데이터 서버(169.254.169.254)에 연결할 수 있습니다.

5. 관리 서브넷

   관리 서브넷은 ASA 가상에 대한 SSH 액세스를 지원하기 위해 퍼블릭이어야 합니다.

6. 보안 그룹 - **ASA** 가상 인스턴스에 대한 네트워크 보안 그룹

   다음 요구 사항을 충족하는 ASA 가상 인스턴스에 대한 보안 그룹을 구성합니다.

   - (동일한 VCN에 있는) Oracle Functions는 ASA 가상의 관리 주소에 대한 SSH 연결을 수행합니다.
   - 관리 호스트는 ASA 가상 인스턴스에 대한 SSH 액세스가 필요할 수 있습니다.
   - ASA 가상은 라이선싱을 위해 CSSM/위성 서버와의 통신을 시작합니다.

7. 개체 스토리지 네임스페이스

   이 개체 스토리지 네임스페이스는 configuration.txt 파일이 있는 정적 웹사이트를 호스팅하는 데 사용합니다. configuration.txt 파일에 대해 사전 인증된 요청을 생성해야 합니다. 이 사전 인증된 URL은 템플릿 구축 중에 사용됩니다.

참고 업로드된 다음 컨피그레이션이 HTTP URL을 통해 ASA 가상 인스턴스에 액세스할 수 있는지 확인합니다.

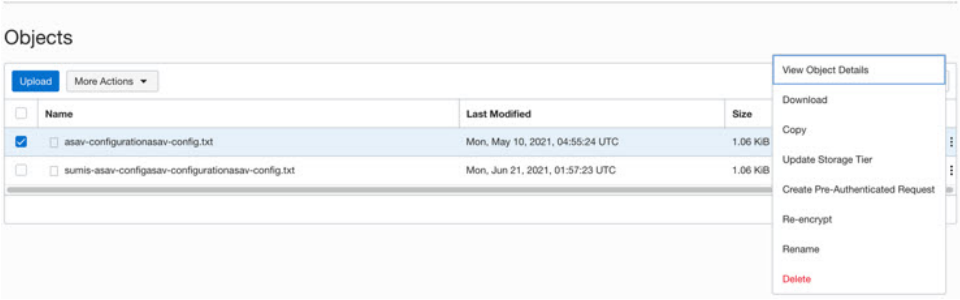
부팅된 ASA 가상은 다음 명령을 실행합니다. `$ copy /noconfirm <configuration.txt file's pre-authenticated request URL > disk0:Connfiguration.txt`

이 명령을 사용하면 configuration.txt 파일을 사용하여 ASA 가상 실행을 구성할 수 있습니다.

8. **configuration.txt** 파일 업로드

   ASA 가상 컨피그레이션 파일의 사전 인증된 요청 URL을 생성하려면 다음을 수행합니다.

   1. **Buckets(**버킷**)** > **Create Bucket(**버킷 생성**)**을 클릭합니다.
   2. **Upload(**업로드**)**를 클릭합니다.
   3. 구성 파일이 업로드되면, 아래 그림에서처럼 **Create Pre-Authenticated Request(**사전 인증된 요청 생성**)**를 선택합니다.



참고 이제 oracle-function에서 컨피그레이션 파일에 액세스할 수 있습니다.

네트워크 설정

1. **Inbound traffic(**인바운드 트래픽**)**

   configuration.txt의 <*Application VM IP*> 주소가 단계 2에 언급된 주소와 일치하는지 확인합니다.

2. **Outbound Traffic(**아웃 바운드 트래픽**)**

   - configuration.txt의 <*External Server IP*> 주소가 단계 2에 언급된 주소와 일치하는지 확인합니다.
   - 외부 VCN에 NAT 게이트웨이 하나가 있는지 확인합니다.
   - 아래 그림에서처럼 외부 VCN의 경로 테이블에 있는 동일한 <*External Server IP*> 주소가 NAT 게이트웨이를 대상으로 하는지 확인합니다.

Destination	Target Type	Target
0.0.0.0/0	Internet Gateway	outside-ig
8.8.8.8/32	NAT Gateway	nat-gw

비밀번호 암호화

참고 이 절차에 대한 자세한 내용은 Create and Secrets(자격 증명 모음 및 비밀번호 생성)을 참고하십시오.

ASA 가상의 비밀번호는 자동 확장 중에 사용하는 모든 ASA 가상 인스턴스를 구성하는 데 사용하며, ASA 가상 인스턴스의 CPU 사용량 데이터를 검색하는 데 사용합니다.

따라서 때때로 비밀번호를 저장하고 처리해야 합니다. 빈번한 변경 및 취약성 가능성 때문에, 비밀번호를 일반 텍스트 형식으로 수정하거나 저장할 수는 없습니다. 비밀번호는 반드시 암호화된 형식이어야 합니다.

암호화된 형식으로 비밀번호를 가져오려면 다음을 수행합니다.

단계 **1** Vault를 생성합니다.

OCI Vault는 마스터 암호화 키를 안전하게 생성하고 저장하는 서비스와, 이러한 서비스를 이용한 암호화 및 암호 해독 방법을 제공합니다. 따라서 (아직 생성하지 않았다면) Vault는 자동 확장 솔루션의 나머지 부분과 동일한 구획에 생성해야 합니다.

OCI > **Identity & Security(ID** 및 보안**)** > **Vault** > **Choose or Create New Vault(**새 **Vault** 선택 또는 생성**)**

.

단계 **2** 마스터 암호화 키를 생성합니다.

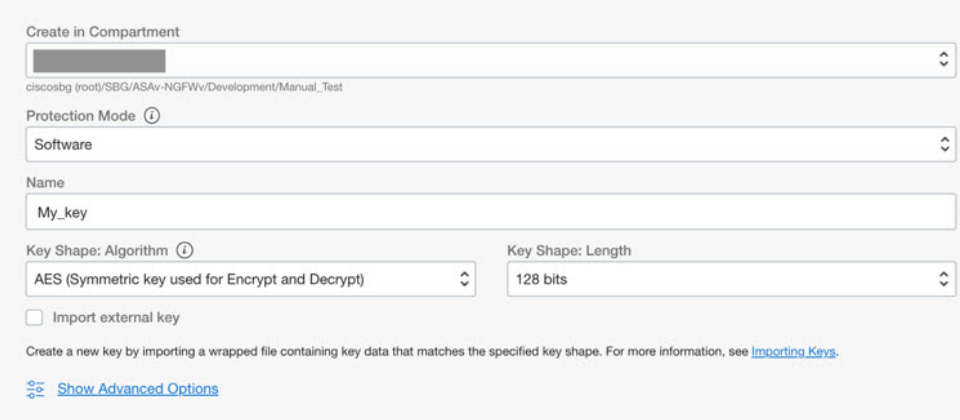
일반 텍스트 비밀번호를 암호화하려면 마스터 암호화 키 하나가 필요합니다.

OCI > **Identity & Security(ID** 및 보안**)** > **Vault** > **Choose or Create Key(**키 선택 또는 생성**)**으로 이동합니다.

임의 비트 길이의 알고리즘에서 키를 선택합니다.

1. AES – 128, 192, 256
2. RSA – 2048, 3072, 4096
3. ECDSA – 256, 384, 521

그림 ***2:*** 새 키 생성



단계 **3** 암호화된 비밀번호를 만듭니다.

1. **OCI** > **Open CloudShell(CloudShell** 열기**)(OCI Cloud Terminal)**로 이동합니다.
2. <*Password*>를 자신의 비밀번호로 교체하여 다음 명령을 실행합니다.

```
echo -n '<Password>' | base64
```

3. 선택한 Vault에서 암호화 엔드포인트 및 마스터 암호화 키 OCID를 복사합니다. 다음 값을 교체하고 암호화 명령을 실행합니다.
   - 사용자 키의 OCID가 포함된 KEY_OCID

- 사용자 Vault의 암호화 엔드포인트 URL이 포함된 Cryptographic_Endpoint_URL
- 비밀번호와 사용자의 비밀번호

암호화 명령

```
oci kms crypto encrypt --key-id Key_OCID --endpoint
Cryptographic_Endpoint_URL --plaintext <base64-value-of-password>
```

4. 위 명령의 출력에서 암호문을 복사하고 필요한 경우 사용합니다.

ASA 컨피그레이션 파일 준비

애플리케이션이 구축되었거나 구축 계획을 사용할 수 있는지 확인합니다.

단계 1 구축하기 전에 다음 입력 매개변수를 수집합니다.

매개변수	데이터 유형	설명
tenancy_ocid	문자열	계정이 속한 테넌시의 OCID입니다. 테넌시 OCID를 찾는 방법은 여기를 참조하십시오. 테넌시 OCID는 `ocid1.tenancy.oc1..<unique_ID>` 같은 형식을 취합니다.
compartment_id	문자열	리소스를 생성할 구획의 OCID입니다. 예: `ocid1.compartment.oc1..<unique_ID>`
compartment_name	문자열	구획의 이름
region	문자열	리소스를 생성할 지역의 고유 식별자입니다. 예: `us-phoenix-1, us-ashburn-1`

매개변수	데이터 유형	설명
lb_size	문자열	외부 및 내부 로드 밸런서의 사전 프로비저닝된 총 대역폭(인그레스 + 이그레스)을 결정하는 템플릿입니다. 지원되는 값: 100Mbps, 10Mbps, 10Mbps-Micro, 400Mbps, 8000Mbps 예: 100Mbps
availability_domain	쉼표로 구분된 값	예: Tpeb:PHX-AD-1 참고 Cloud Shell에서 **oci iam availability-domain list** 명령을 실행하여 가용성 도메인 이름을 가져옵니다.
min_and_max_instance_count	쉼표로 구분된 값	인스턴스 풀에서 유지할 최소 및 최대 인스턴스 수입니다. 예: 1,5
autoscale_group_prefix	문자열	템플릿을 사용하여 생성되는 모든 리소스의 이름을 지정하는 데 사용하는 접두사입니다. 예를 들어 리소스 접두사가 'autoscale'로 지정된 경우, 모든 리소스의 이름은 autoscale_resource1, autoscale_resource2 등으로 지정됩니다.
asav_config_file_url	URL	ASA 가상을 구성하는 데 사용할 개체 스토리지에 업로드된 구성 파일의 URL입니다. 참고 컨피그레이션 파일의 사전 인증된 요청 URL이 제공되어야 합니다. 예: https://objectstorage.<region-name>.oraclecloud.com/<object-storage-name>/oci-asav-configuration.txt
mgmt_subnet_ocid	문자열	사용할 관리 서브넷의 OCID입니다.
inside_subnet_ocid	문자열	사용할 내부 서브넷의 OCID입니다.
outside_subnet_ocid	문자열	사용할 외부 서브넷의 OCID입니다.

매개변수	데이터 유형	설명
mgmt_nsg_ocid	문자열	사용할 관리 서브넷 네트워크 보안 그룹의 OCID입니다.
inside_nsg_ocid	문자열	사용할 내부 서브넷 네트워크 보안 그룹의 OCID입니다.
outside_nsg_ocid	문자열	사용할 외부 서브넷 네트워크 보안 그룹의 OCID입니다.
elb_listener_port	쉼표로 구분된 값	외부 로드 밸런서 리스너의 통신 포트 목록입니다. 예: 80
ilb_listener_port	쉼표로 구분된 값	내부 로드 밸런서 리스너의 통신 포트 목록입니다. 예: 80
health-check-port	문자열	상태 확인을 실행할 로드 밸런서의 백엔드 서버 포트입니다. 예: 8080
instance_shape	문자열	생성할 인스턴스의 셰이프입니다. 셰이프는 인스턴스에 할당되는 CPU 수, 메모리 양 및 기타 리소스를 결정합니다. 지원되는 셰이프: "VM.Standard2.4" 및 "VM.Standard2.8"
lb_bs_policy	문자열	내부 및 외부 로드 밸런서의 백엔드 집합에 사용할 로드 밸런서 정책입니다. 로드 밸런서 정책의 작동 방식에 대한 자세한 내용은 여기를 참조하십시오. 지원되는 값: "ROUND_ROBIN", "LEAST_CONNECTIONS", "IP_HASH"

매개변수	데이터 유형	설명
image_name	문자열	인스턴스 구성을 생성하는 데 사용하는 마켓플레이스 이미지의 이름입니다. 기본값: "Cisco ASAv(ASA 가상 방화벽)" 참고 맞춤형 이미지를 구축하고 싶은 사용자는 custom_image_ocid 매개변수를 구성해야 합니다.
image_version	문자열	사용할 OCI Marketplace에서 사용할 수 있는 ASA 가상 이미지의 버전입니다. 현재 9.15.1.15 및 9.16.1 버전을 사용할 수 있습니다. 기본값: "Cisco ASAv(ASA 가상 방화벽)"
scaling_thresholds	쉼표로 구분된 값	축소 및 확장에 사용할 CPU 사용량 임계값입니다. 축소 및 확장 임계값을 쉼표로 구분된 입력으로 지정하십시오. 예: 15,50 여기서 15는 축소 임계값이고 50은 확장 임계값입니다.
custom_image_ocid	문자열	마켓플레이스 이미지를 사용하지 않을 경우 인스턴스 컨피그레이션을 생성하는 데 사용할 맞춤형 이미지의 OCID입니다. 참고 custom_image_ocid는 선택적 매개변수입니다.
asav_password	문자열	암호화된 형식의 ASA 가상용 암호로, 구성을 위한 ASA 가상로의 SSH에 사용합니다. 비밀번호를 암호화하는 방법에 대한 지침은 구성 가이드나 여기를 참조하십시오.
cryptographic_endpoint	문자열	암호화 엔드포인트는 비밀번호 해독에 사용하는 URL입니다. Vault에서 찾을 수 있습니다.

매개변수	데이터 유형	설명
master_encryption_key_id	문자열	비밀번호가 암호화된 키의 OCID입니다. Vault에서 찾을 수 있습니다.
Profile Name(프로필 이름)		OCI에서의 사용자 프로파일 이름입니다. 사용자의 프로파일 섹션에서 확인할 수 있습니다. 예: oracleidentitycloudservice/<user>@<mail>.com
개체 스토리지 네임스페이스		테넌시를 만들 때 생성되는 고유한 식별자입니다. 이 값은 **OCI** > **Administration(**관리**)** > **Tenancy Details(**테넌시 세부 정보**)**에서 확인할 수 있습니다.
권한 부여 토큰		Oracle-Functions를 OCI 컨테이너 레지스트리에 푸시할 권한을 부여하는 docker 로그인의 비밀번호로 사용됩니다. 토큰을 얻으려면 **OCI** > **Identity(ID)** > **Users(**사용자**)** > **User Details(**사용자 세부 정보**)** > **Auth Tokens(**인증 토큰**)** > **Generate Token(**토큰 생성**)**으로 이동합니다.

단계 **2** 로드 밸런서 상태 프로브 및 액세스 정책에 대한 개체, 라이선싱 및 NAT 규칙을 구성합니다.

```
! Default route via outside
route outside 0.0.0.0 0.0.0.0 <Outside Subnet gateway> 2

! Health Check Configuration
object network metadata-server
host 169.254.169.254
object service health-check-port
service tcp destination eq <health-check-port>
object service http-port
service tcp destination eq <traffic port>
route inside 169.254.169.254 255.255.255.255 <Inside Subnet GW> 1

! Health check NAT
nat (outside,inside) source static any interface destination static interface metadata-server service
health-check-port http-port
nat (inside,outside) source static any interface destination static interface metadata-server service
health-check-port http-port

! Outbound NAT
object network inside-subnet
subnet <Inside Subnet> <Inside Subnet Gateway>
object network external-server
host <External Server IP>
nat (inside,outside) source static inside-subnet interface destination static interface external-server


! Inbound NAT
object network outside-subnet
```

```
subnet <Outside Subnet> <Outside Subnet GW>
object network http-server-80
host <Application VM IP>
nat (outside,inside) source static outside-subnet interface destination static interface http-server-80


!
dns domain-lookup outside
DNS server-group DefaultDNS

! License Configuration
call-home
profile license
destination transport-method http
destination address http <URL>
debug menu license 25 production
license smart
feature tier standard
throughput level <Entitlement>
licence smart register idtoken <License token> force
!
```

이러한 상태 프로브 연결 및 데이터 플레인 구성은 액세스 정책에서 허용되어야 합니다.

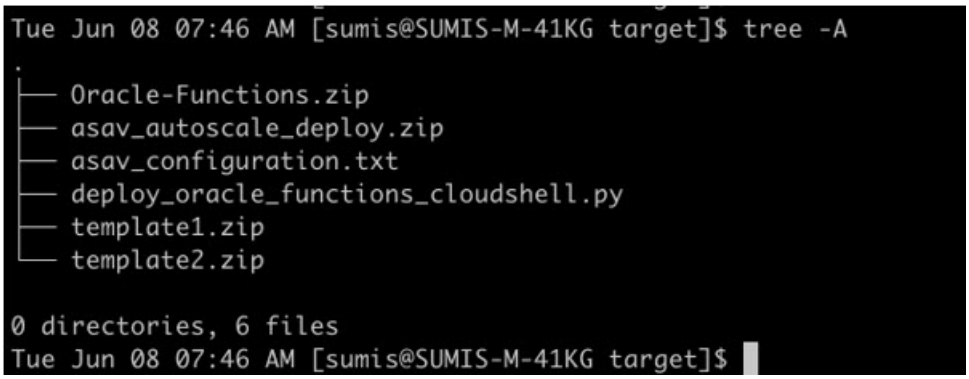
단계 **3** 컨피그레이션 세부 정보를 이용해 *configuration.txt* 파일을 업데이트합니다.

단계 **4** *configuration.txt* 파일을 사용자가 생성한 개체 스토리지 공간에 업로드하고, 업로드된 파일에 대한 사전 인증된 요청을 생성합니다.

참고 configuration.txt의 사전 인증된 요청 URL을 스택 구축에 사용하는지 확인합니다.

단계 **5** Zip 파일을 생성합니다.

make.py 파일은 복제된 리포지토리에 있습니다. `python3 make.py build` 명령을 실행하여 zip 파일을 생성합니다. 대상 폴더에는 다음 파일이 있습니다.



참고 클라우드 셸을 사용하여 자동 확장 솔루션을 구축하는 경우 `python3 make.py build`를 실행하기 전에 *easy_deploy/deployment_parameters.json* 파일을 업데이트합니다. 업데이트에 대해서는 단계 1 및 Oracle Functions 구축을 참조하십시오.

OCI에 자동 확장 구축

구축을 위한 사전 요구 사항 단계가 완료되면 OCI 스택 생성을 시작합니다. 수동 구축을 수행하거나 Cloud Shell을 사용하여 자동 확장 구축을 수행할 수 있습니다. 사용자 버전에 맞는 구축 스크립트 및 템플릿은 GitHub 리포지토리에서 제공됩니다.

수동 구축

엔드 투 엔드 Autoscale 솔루션 구축은 Terraform 템플릿 1 스택 구축 , Oracle Functions 구축, Terraform 템플릿-2 구축이라는 3가지 단계로 구성됩니다.

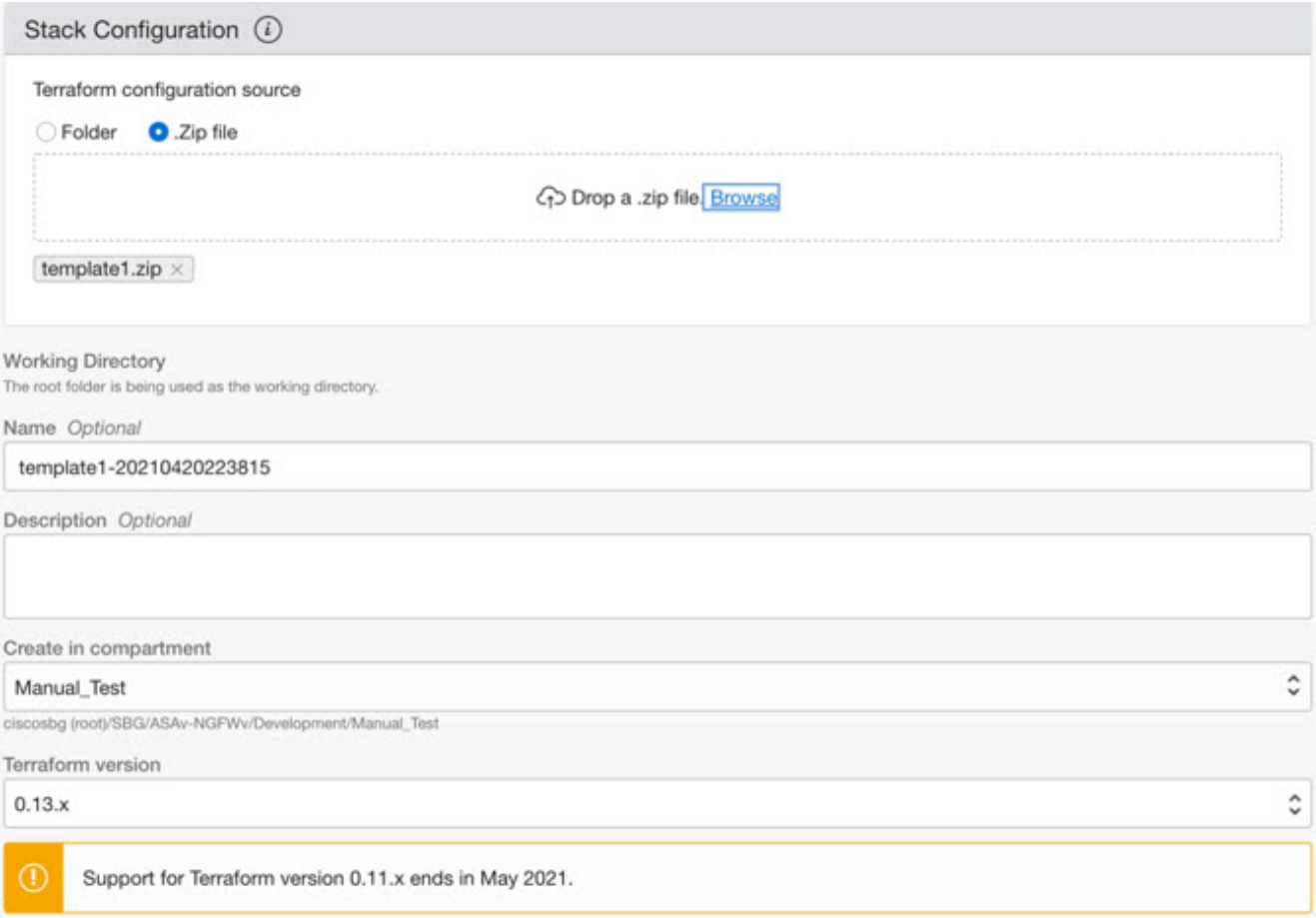
Terraform 템플릿 1 스택 구축

단계 **1** OCI 포털에 로그인합니다.

화면의 우측 상단에 지역이 표시됩니다. 원하는 지역에 있는지 정기적으로 확인합니다.

단계 **2** **Developer Service(**개발자 서비스**)** > **Resource Manager(**리소스 관리자**)** > **Stack(**스택**)** > **Create Stack(**스택 생성**)**을 선택합니다.

My Configuration(내 구성**)**을 선택하고, 아래 그림에서처럼 대상 폴더에 있는 *Terraform template1.zip* 팔을일 Terraform 컨피그레이션 소스로 선택합니다.



단계 **3** **Transform version(**변형 버전**)** 드롭다운 목록에서 0.13.x 또는 0.14.x를 선택합니다.

단계 **4** 다음 단계에서 단계 1에 수집된 모든 세부 정보를 입력합니다.

참고 유효한 입력 매개변수를 입력하십시오. 입력하지 않으면 다음 단계에서 스택 구축이 실패할 수 있습니다.

단계 **5** 다음 단계에서 **Terraform Actions(Terraform** 작업**)** > **Apply(**적용**)**를 클릭합니다.

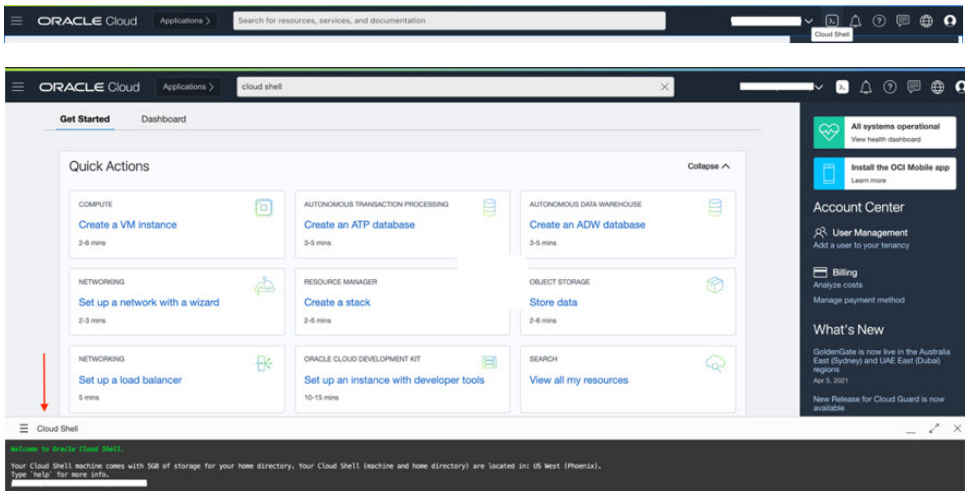
구축을 성공적으로 완료되면 Oracle Functions 구축을 진행합니다.

Oracle Functions 구축

참고 이 단계는 *Terraform* 템플릿 *1*을 구축한 후에만 수행해야 합니다.

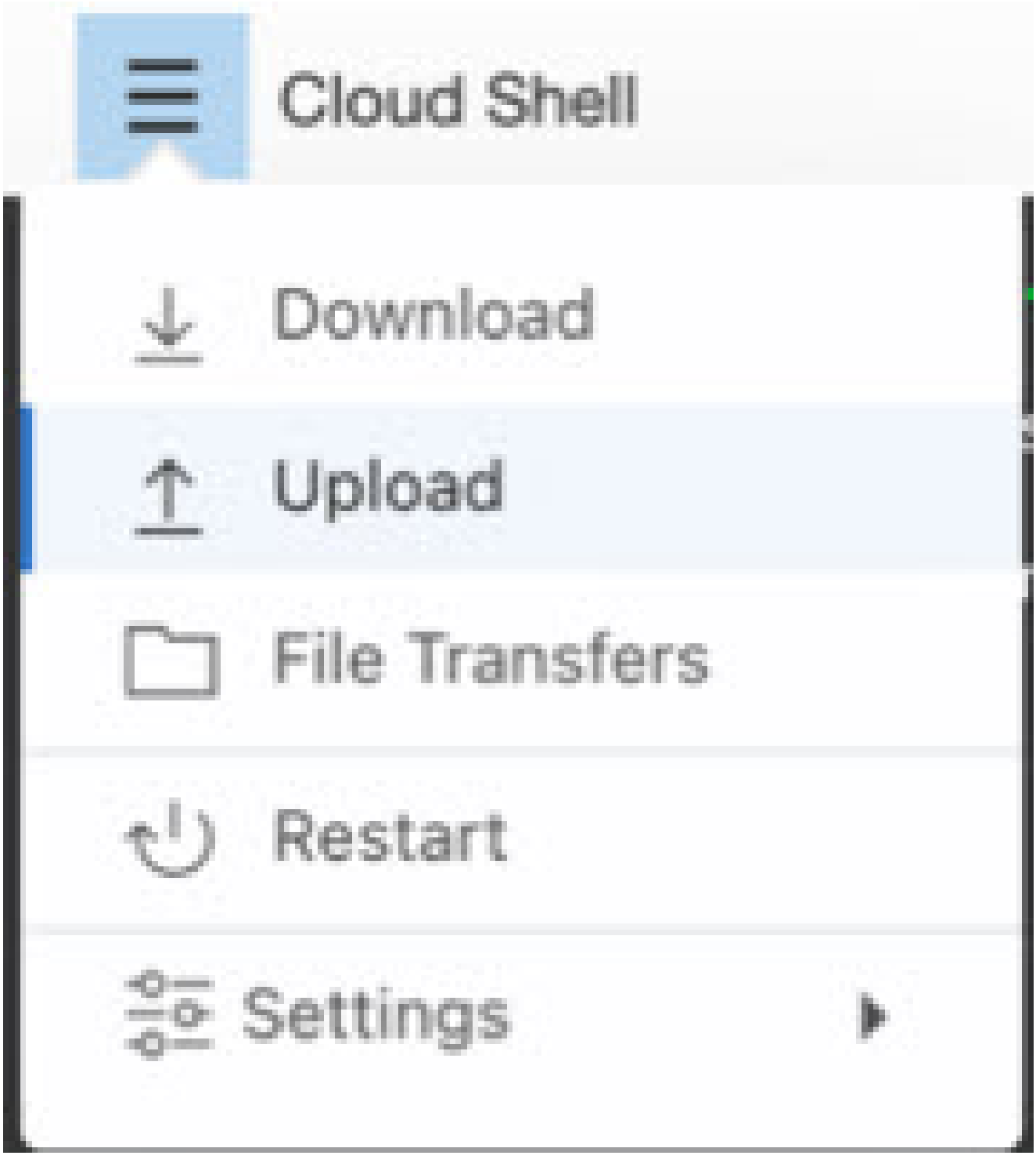
OCI에서 Oracle Functions는 OCI 컨테이너 레지스트리에 저장되는 Docker 이미지로 업로드됩니다. 구축할 때 Oracle Functions를 (Terraform 템플릿 1에서 생성된) OCI 애플리케이션 중 하나에 푸시해야 합니다.

단계 **1** OCI Cloud Shell을 엽니다.



단계 **2** *deploy_oracle_functions_cloudshell.py* 및 *Oracle-Functions.zip*을 업로드합니다.

Cloud Shell의 햄버거 메뉴에서 **Upload(**업로드**)**를 선택합니다.



단계 **3** **ls** 명령을 사용하여 파일을 확인합니다.

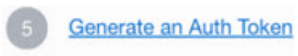
단계 **4** `python3 Deploy_Oracle_Functions.py -h`를 실행합니다. `deploy_oracle_functions_cloudshell.py` 스크립트에는 아래 그림에서처럼 help 인수를 사용하여 세부 정보를 찾을 수 있는 몇 가지 입력 매개변수가 필요합니다.

```
$ python3 Deploy_Oracle_Functions.py -h
usage: Deploy_Oracle_Functions.py [-h] -a  -r  -p  -c  -o  -t

*** Script to deploy Oracle Function for OCI ASAv Autoscale Solution ***

Instruction to find values of required arguments:
Application Name: Name of Application created by first Terraform Template
Region Identifier: OCI -> Administration -> Region Management
Profile Name: OCI -> Profile
Compartment OCID: OCI -> Identity -> Compartment -> Compartment Details
Object Storage Namespace: OCI -> Administration -> Tenancy Details
Authorization Token: OCI -> Identity -> Users -> User Details -> Auth Tokens -> Generate Token

optional arguments:
  -h, --help  show this help message and exit
  -a          Name of Application in OCI to which functions will be deployed
  -r          Region Identifier
  -p          Profile Name of User
  -c          Compartment OCID
  -o          Object Storage Namespace
  -t          Authorization Token for Docker Login (*Please Put in Quotes)
```

스크립트를 실행하려면 다음 인수를 전달해야 합니다.

표 *1:* 인수 및 세부 정보

인수	세부 사항
애플리케이션 이름	Terraform 템플릿 1 구축에 의해 생성된 OCI 애플리케이션의 이름입니다. 값은 템플릿 1에 제공된 "**autoscale_group_prefix**"와 "**_application**" 접미사를 결합하여 얻습니다.
지역 식별자	지역 식별자는 다양한 지역에 대해 OCI에서 고정된 지역 코드워드입니다. 예: 'us-phoenix-1'(Phoenix) 또는 "ap-melbourne-1"(Melbourne). 지역 식별자가 포함된 모든 지역의 목록을 가져오려면 **OCI** > **Administration(**관리**)** > **Region Management(**지역 관리**)**로 이동합니다.
Profile Name(프로필 이름**)**	OCI에서의 단순 사용자 프로파일 이름입니다. 예: *oracleidentitycloudservice/<user>@<mail>.com* 이름은 사용자의 프로파일 섹션에서 확인할 수 있습니다.

인수	세부 사항
구획 **OCID**	구획의 OCID(Oracle Cloud Identifier)입니다. 사용자가 OCI 애플리케이션을 보유한 구획 OCID입니다. **OCI** > **Identity(ID)** > **Compartment(**구획**)** > **Compartment Details(**구획 상세 정보**)**로 이동합니다.
개체 스토리지 네임스페이스	테넌시를 만들 때 생성되는 고유한 식별자입니다. **OCI** > **Administration(**관리**)** > **Tenancy Details(**테넌시 세부 정보**)**로 이동합니다.
권한 부여 토큰	Oracle-Functions를 OCI 컨테이너 레지스트리에 푸시할 권한을 부여하는 docker 로그인의 비밀번호로 사용됩니다. 구축 스크립트에서 따옴표로 묶인 토큰을 지정합니다. **OCI** > **Identity(ID)** > **Users(**사용자**)** > **User Details(**사용자 세부 정보**)** > **Auth Tokens(**인증 토큰**)** > **Generate Token(**토큰 생성**)**으로 이동합니다. 어떤 이유로든 User Details(사용자 세부사항)가 표시되지 않는다면 **Developer services(**개발자 서비스**)** > **Functions(**기능**)**를 클릭합니다. Terraform 템플릿 1에서 생성한 애플리케이션으로 이동합니다. **Getting Started(**시작하기**)**를 클릭하고 Cloud Shell Setup(클라우드 쉘 설정)을 선택하면 아래에서처럼 인증 토큰을 생성할 수 있는 링크가 표시됩니다. 

단계 **5** 유효한 입력 인수를 전달하여 `python3 Deploy_Oracle_Functions.py` 명령을 실행합니다. 모든 기능이 구축될 때까지 시간이 오래 걸릴 수 있습니다. 구축이 끝나면 파일을 제거하고 Cloud Shell을 닫아도 됩니다.

Terraform 템플릿-2 구축

템플릿 2는 알람, 기능 호출을 위한 ONS 항목 같은 알람 생성 관련 리소스를 구축합니다. 템플릿 2 구축은 Terraform 템플릿-1 구축과 유사합니다.

단계 **1** OCI 포털에 로그인합니다.

화면의 우측 상단에 지역이 표시됩니다. 원하는 지역에 있는지 정기적으로 확인합니다.

단계 **2** **Developer Service(**개발자 서비스**)** > **Resource Manager(**리소스 관리자**)** > **Stack(**스택**)** > **Create Stack(**스택 생성**)**을 선택합니다.

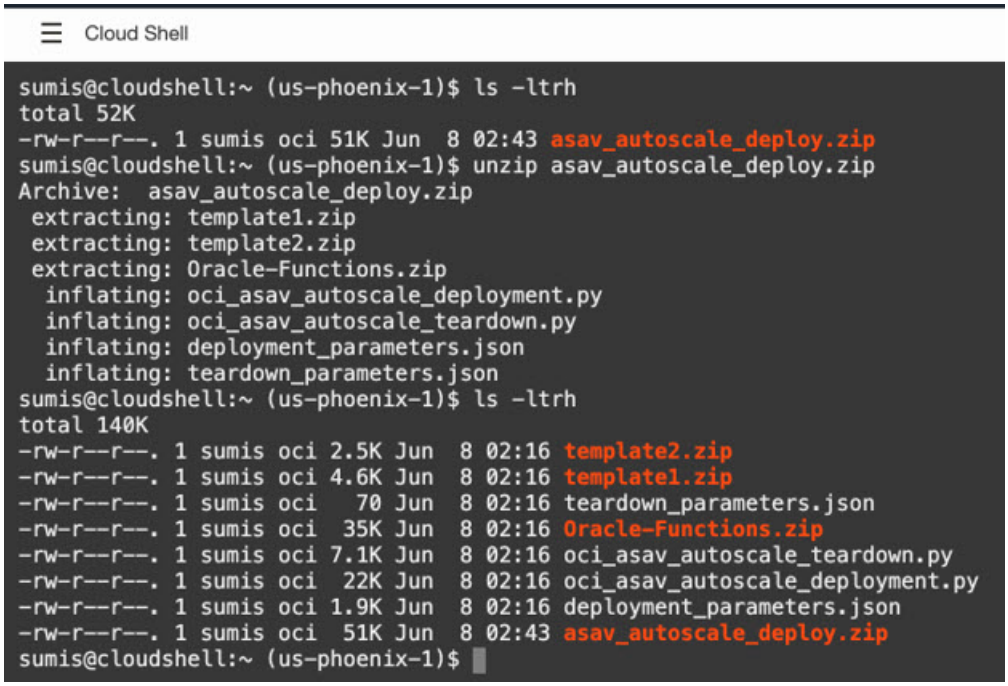
대상 폴더의 *Terraform template2.zip*을 Terraform 컨피그레이션의 소스로 선택합니다.

단계 **3** 다음 단계에서 **Terraform Actions(Terraform** 작업**)** > **Apply(**적용**)**를 클릭합니다.

Cloud Shell을 사용하여 자동 확장 구축

구축 오버헤드를 방지하기 위해 간편한 엔드 투 엔드 구축 스크립트를 호출하여 자동 확장 솔루션(terraform template1, template2 및 oracle functions)을 구축할 수 있습니다.

단계 **1** 대상 폴더의 *asav_autoscale_deploy.zip* 파일을 클라우드 쉘에 업로드하고 파일의 압축을 풉니다.



단계 **2** `python3 make.py` 빌드 명령을 실행하기 전에 *deployment_parameters.json*에서 입력 매개변수를 업데이트했는지 확인합니다.

단계 **3** 자동 확장 솔루션 구축을 시작하려면 클라우드 쉘에서 `python3 oci_asav_autoscale_deployment.py` 명령을 실행합니다.

솔루션 구축이 끝나려면 약 10~15분 정도 걸립니다.

솔루션 구축 중에 오류가 발생하면 오류 로그가 저장됩니다.

구축 검증

모든 리소스가 구축되고 Oracle Functions가 알람 및 이벤트와 연결되어 있는지 확인합니다. 기본적으로 인스턴스 풀은 최소 및 최대 인스턴스 수가 0입니다. OCI UI에서 원하는 최소 및 최대 개수로 인스턴스 풀을 수정할 수 있습니다. 이렇게 하면 새 ASA 가상 인스턴스가 트리거됩니다.

인스턴스를 하나만 실행하고 관련 워크플로우를 확인한 다음 동작을 검증하여, 예상대로 작동하는지 확인하는 것이 좋습니다. 이 검증을 게시하면 ASA 가상의 실제 요구 사항을 구축할 수 있습니다.

참고 OCI 확장 정책 때문에 제거되지 않도록, ASA 가상 인스턴스의 최소 수를 **Scale-In protected(**축소 보호**)**로 지정합니다.

자동 확장 업그레이드

Autoscale 스택 업그레이드

이 릴리스에서는 업그레이드가 지원되지 않습니다. 스택을 재구축해야 합니다.

ASA 가상 **VM** 업그레이드

이 릴리스에서는 ASA 가상 VM 업그레이드가 지원되지 않습니다. 필요한 ASA 가상 이미지를 사용하여 스택을 재구축해야 합니다.

인스턴스 풀

1. 인스턴스 풀의 최소 및 최대 인스턴스 수를 변경하려면 다음을 수행합니다.

   Developer Services(개발자 서비스**)** > **Function(**기능**)** > **Application Name(created by Terraform Template 1)(**애플리케이션 이름**, Terraform** 템플릿 **1**에서 생성**)** > **Configuration(**구성**)**을 클릭합니다.

   min_instance_count 및 max_instance_count를 각각 변경합니다.

2. 인스턴스 삭제/종료는 축소와는 다릅니다. 축소 작업이 아니라 외부 작업 때문에 인스턴스 풀의 인스턴스가 삭제/종료된 경우, 인스턴스 풀은 복구를 위해 새 인스턴스를 자동으로 시작합니다.

3. Max_instance _count는 확장 작업에 대한 임계값 제한을 정의하지만, UI를 통해 인스턴스 풀의 인스턴스 수를 변경하면 이 제한을 초과할 수 있습니다. UI의 인스턴스 수가 OCI 애플리케이션에 설정된 max_instance_count보다 작은지 확인합니다. 그렇지 않다면 적절하게 임계값을 늘리십시오.

4. 애플리케이션에서 바로 인스턴스 풀의 인스턴스 수를 줄이면, 프로그래밍 방식으로 설정된 정리 작업이 수행되지 않습니다. 백엔드가 두 로드 밸런서 모두에서 드레인되거나 제거되지 않으므로, ASA 가상에 라이선스가 있다면 백엔드가 손실됩니다.

5. 몇 가지 이유 때문에, ASA 가상 인스턴스가 비정상적으로 응답하지 않고 일정 기간 SSH를 통해 연결할 수 없다면 인스턴스가 인스턴스 풀에서 강제로 제거되며 라이선스가 손실될 수 있습니다.

Oracle Functions

- Oracle Functions는 실제로는 docker 이미지입니다. 이러한 이미지는 OCI 컨테이너 레지스트리의 루트 디렉토리에 저장됩니다. 이러한 이미지는 삭제하면 안 됩니다. Autoscale 솔루션에서 사용하는 기능도 삭제되기 때문입니다.
- Terraform 템플릿 1에서 생성한 OCI 애플리케이션에는 Oracle Functions가 정상적으로 작동하는 데 필요한 중요한 환경 변수가 포함되어 있습니다. 반드시 필요한 경우가 아니면 이러한 환경 변수의 값이나 형식을 변경해선 안 됩니다. 변경 사항은 새 인스턴스에만 반영됩니다.

로드 밸런서 백엔드 집합

OCI에서, 인스턴스 풀에 대한 로드 밸런서 연결은 ASA 가상에서의 관리 인터페이스로 구성된 기본 인터페이스를 사용하는 경우에만 지원됩니다. 따라서 내부 인터페이스는 내부 로드 밸런서의 백엔드 집합에 연결되고 외부 인터페이스는 외부 로드 밸런서의 백엔드 집합에 연결됩니다. 이러한 IP는 자동으로 백엔드 집합에 추가되거나 집합에서 제거되지 않습니다. Autoscale 솔루션은 두 작업을 모두 프로그래밍 방식으로 처리합니다. 그러나 외부 활동, 유지 보수 또는 문제 해결을 위해 작업을 수동으로 수행해야 할 수도 있습니다.

요구 사항에 따라 리스너 및 백엔드 집합을 사용하여 로드 밸런서에서 추가 포트를 열 수 있습니다. 향후 인스턴스 IP는 백엔드 집합에 자동으로 추가되지만, 이미 존재하는 인스턴스 IP는 수동으로 추가해야 합니다.

로드 밸런서에 리스너 추가

포트를 로드 밸런서에 리스너로 추가하려면 **OCI** > **Networking(**네트워킹**)** > **Load Balancer(**로드 밸런서**)** > **Listener(**리스너**)** > **Create Listener(**리스터 생성**)**로 이동합니다.

백엔드 집합에 백엔드 등록

ASA 가상 인스턴스를 로드 밸런서에 추가하려면 ASA 가상 인스턴스 외부 인터페이스 IP를 외부 로드 밸런서의 백엔드 집합에서 백엔드로 구성해야 합니다. 내부 인터페이스 IP는 내부 로드 밸런서의 백엔드 집합에서 백엔드로 구성해야 합니다. 사용 중인 포트가 리스너에 추가되었는지 확인합니다.

OCI에서 자동 확장 구성 삭제

Terraform을 사용하여 구축된 스택한 OCI의 Resource Manager를 사용하여 동일한 방식으로 삭제할 수 있습니다. 스택을 삭제하면 스택에서 생성된 모든 리소스가 제거되고 이러한 리소스와 연결된 모든 정보가 영구적으로 제거됩니다.

참고 스택을 삭제할 때는 인스턴스 풀의 최소 인스턴스 수)을 0으로 설정한 다음 인스턴스가 종료될 때까지 대기하는 것이 좋습니다. 이렇게 하면 모든 인스턴스를 제거하고 잔여물을 남기지 않을 수 있습니다.

수동 삭제를 수행하거나 Cloud Shell을 사용하여 자동 확장 삭제 을 사용할 수 있습니다.

수동 삭제

엔드 투 엔드 Autoscale 솔루션 삭제는 Terraform 템플릿 2 스택 삭제, Oracle-Functions 삭제, Terraform 템플릿 1 스택 삭제 라는 3가지 단계로 구성됩니다.

Terraform 템플릿 2 스택 삭제

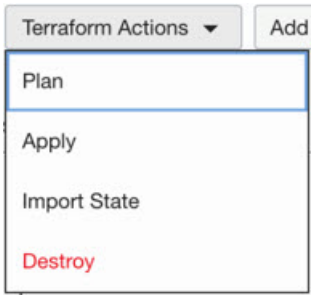
자동 확장 구성을 삭제하려면 Terraform 템플릿 2 스택 삭제부터 시작해야 합니다.

단계 **1** OCI 포털에 로그인합니다.

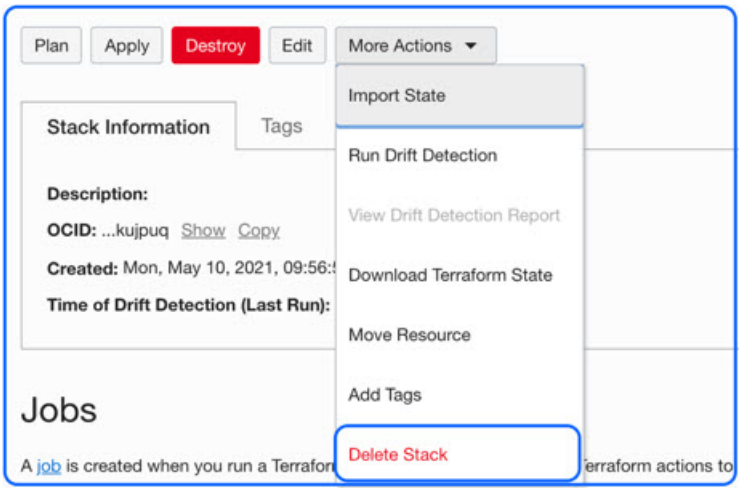
화면의 우측 상단에 지역이 표시됩니다. 원하는 지역에 있는지 정기적으로 확인합니다.

단계 **2** **Developer Services(**개발자 서비스**)** > **Resource Manager(**리소스 관리자**)** > **Stack(**스택**)**을 선택합니다.

단계 **3** Terraform 템플릿 2에서 생성한 스택을 선택한 다음, 아래 그램에서처럼 **Terraform Actions(Terraform** 작업**)** 드롭다운 메뉴에서 **Destroy(**삭제**)**를 클릭합니다.



Destroy Job(삭제 작업)이 생성됩니다. 리소스를 하나씩 제거하기 때문에 시간이 조금 걸립니다. 삭제 작업이 완료되면 아래 그림에서처럼 스택을 삭제할 수 있습니다.



단계 **4** 계속해서 Oracle 기능을 삭제합니다.

Oracle-Functions 삭제

Oracle-Function 구축은 Terraform 템플릿 스택 구축의 일부가 아니며, 클라우드 쉘을 사용하여 별도로 업로드됩니다. 따라서 Terraform 스택 삭제에서는 이 구축 삭제를 지원하지 않습니다. Terraform 템플릿 1에서 생성한 OCI 애플리케이션 내의 모든 Oracle-Functions를 삭제해야 합니다.

단계 **1** OCI 포털에 로그인합니다.

화면의 우측 상단에 지역이 표시됩니다. 원하는 지역에 있는지 정기적으로 확인합니다.

단계 **2** **Developer Services(**개발자 서비스**)** > **Functions(**기능**)**를 선택합니다. 템플릿 1 스택에서 생성된 애플리케이션 이름을 선택합니다.

단계 **3** 이 애플리케이션에서 각 기능을 방문하여 삭제합니다.

Terraform 템플릿 1 스택 삭제

참고 템플릿 1 스택 삭제는 모든 Oracle-Functions를 삭제한 후에만 성공합니다.

Terraform 템플릿 2 삭제와 동일합니다.

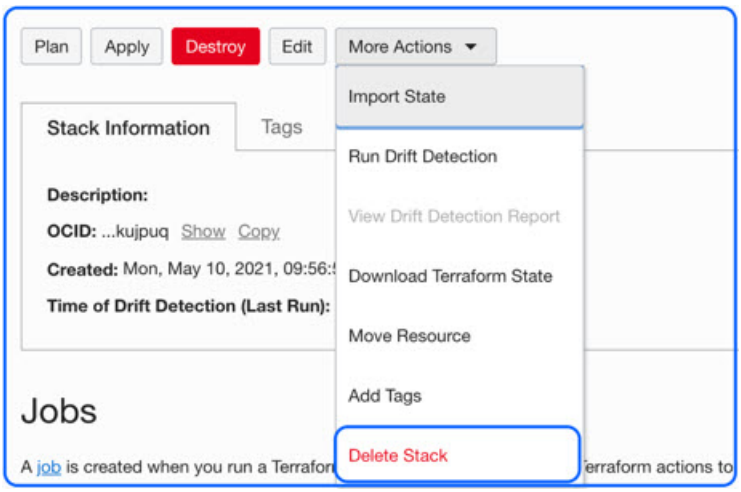
단계 **1** OCI 포털에 로그인합니다.

화면의 우측 상단에 지역이 표시됩니다. 원하는 지역에 있는지 정기적으로 확인합니다.

단계 **2** **Developer Services(**개발자 서비스**)** > **Resource Manager(**리소스 관리자**)** > **Stack(**스택**)**을 선택합니다.

단계 **3** Terraform 템플릿 2에서 생성한 스택을 선택한 다음 **Terraform Actions(Terraform** 작업**)**드롭다운 메뉴에서 **Destroy(**삭제**)**를 클릭합니다. Destroy Job(삭제 작업)이 생성됩니다. 리소스를 하나씩 제거하기 때문에 시간이 조금 걸립니다.

단계 **4** 제거 작업이 완료되면 아래 그림과 같이 **More Actions(**추가 작업**)** 드롭다운 메뉴에서 스택을 삭제할 수 있습니다.



Terraform 템플릿 1 스택을 삭제한 후에는 모든 리소스가 삭제되고 어떤 유형의 잔여물도 없는지 확인해야 합니다.

Cloud Shell을 사용하여 자동 확장 삭제

사용자는 스크립트를 사용하여 클라우드 쉘에서 `python3 oci_asav_autoscale_teardown.py` 명령을 실행하여 스택과 oracle 기능을 삭제할 수 있습니다. 스택을 수동으로 구축하는 경우 stack1 및 stack2의 stack ID를 업데이트하고, *teardown_parameters.json* 파일에서 애플리케이션 ID를 업데이트합니다.