***Chapter 3*: Designing a Network on Oracle Cloud Infrastructure**

**Virtual Cloud Networks** (**VCNs**) provide network connectivity to all of the OCI services with a fully customizable private network in the Oracle cloud. Customers can bring in their own IP segment and assign it to their VCN. They can create their own network topology using guided workflows to connect the virtual subnets with virtual routers and set up firewall rules. Optionally, the VCN can be configured to have internet access and/or VPN access. Customers can then launch bare metal or virtual machine instances in VCN subnets, and the instances will be assigned a private IP address from that same subnet. Optionally, the public IP address can also be assigned to an instance that will enable communication with the internet. Customers can also use security lists (groups of firewall rules) that can be associated with an instance. A VCN is regional and does not span multiple regions. A VCN can span across multiple **Availability Domains** (**ADs**).

In this chapter, we're going to cover the following main topics:

* High-level architecture of VCNs
* VCN components
* Connectivity choices
* Load balancing
* VCN flow logs

**High level architecture of VCNs**

**VCN**s are implemented by creating an overlay network over the substrate network that is hosting the physical hosts. The overlay network provides address space separation between the substrate network and the customer visible VCN and provides the ability to place and migrate networking resources independent of the underlying network topology.

With the overlay network, a network packet-encapsulation mechanism is used to hide the overlay IP addresses from the underlying substrate network. The packet encapsulation is implemented on a smart chip installed on the compute nodes as well as the physical nodes that act as the border between the substrate network and the external networks.

The overlay network configuration is computed centrally by the VCN Control Plane and the packet-routing rules are delivered to the compute nodes. The compute nodes use these rules to determine the encapsulation and decapsulation action to take on every packet that goes in or out. In addition, these rules are also delivered to network devices that perform encapsulation and decapsulation.

However, as the fundamentals of a software-defined network, core network devices, including switches and routers, are unaware of the overlay networking. They do not directly participate in the rule's discovery, propagation, or management and they do not implement encapsulation and decapsulation.

OCI uses the subnet to divide each VCN for further segmentation. You can either specify these subnets as AD specific or they can be *regional* (*recommended*). If you create a subnet that is AD specific, then that address space doesn't span other ADs. However, if you specify that the subnet is regional, then that address space will span across all the available ADs in that region. As you can imagine, subnets are nothing but a contiguous range of IPs. You cannot have overlapping IP addresses in the subnets.

This is what has been illustrated in the following diagram, where you can see that subnets A-C are AD specific, whereas subnet D spans the entire region:

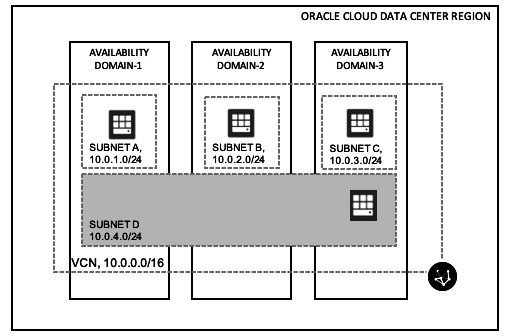


Figure 3.1 – VCN architecture

**VCN components**

While consuming Oracle Cloud in any form, the first thing that you need to set up is your virtual network through a VCN. This section details the various different components of the VCN.

**Subnets**

Instances placed onto each subnet automatically receive their network configuration from the subnet itself. However, you also have the option of manually specifying your own private IP address from the address scope of the subnet.

You can specify a subnet as either a private subnet or a public subnet:

* **Private subnet** – All the instances in this subnet get private IP addresses assigned to their attached VNICs.
* **Public subnet** – Instances placed in the public subnet not only get a public IP address assigned for external communication, but they also get a private IP address assigned to their VNICs.

**VNIC**

A **VNIC**, or **virtual network interface card**, is attached to an instance, and allows the instance to connect to a subnet within a VCN. This VNIC is responsible for deciding how this instance is going to be connected to the VCN and how it will send and receive network traffic. Whenever you launch an instance, the OCI Control Plane creates and attaches a VNIC to the instance. It is called a primary VNIC and you cannot remove this from the instance. Once this instance is up and running, you have the flexibility to add a secondary VNIC. You have the ability to remove it later, if you wish. You can assign this VNIC to the primary NIC's subnet, or you can choose a different subnet to attach it to.

You can see an illustration of this in the following diagram:

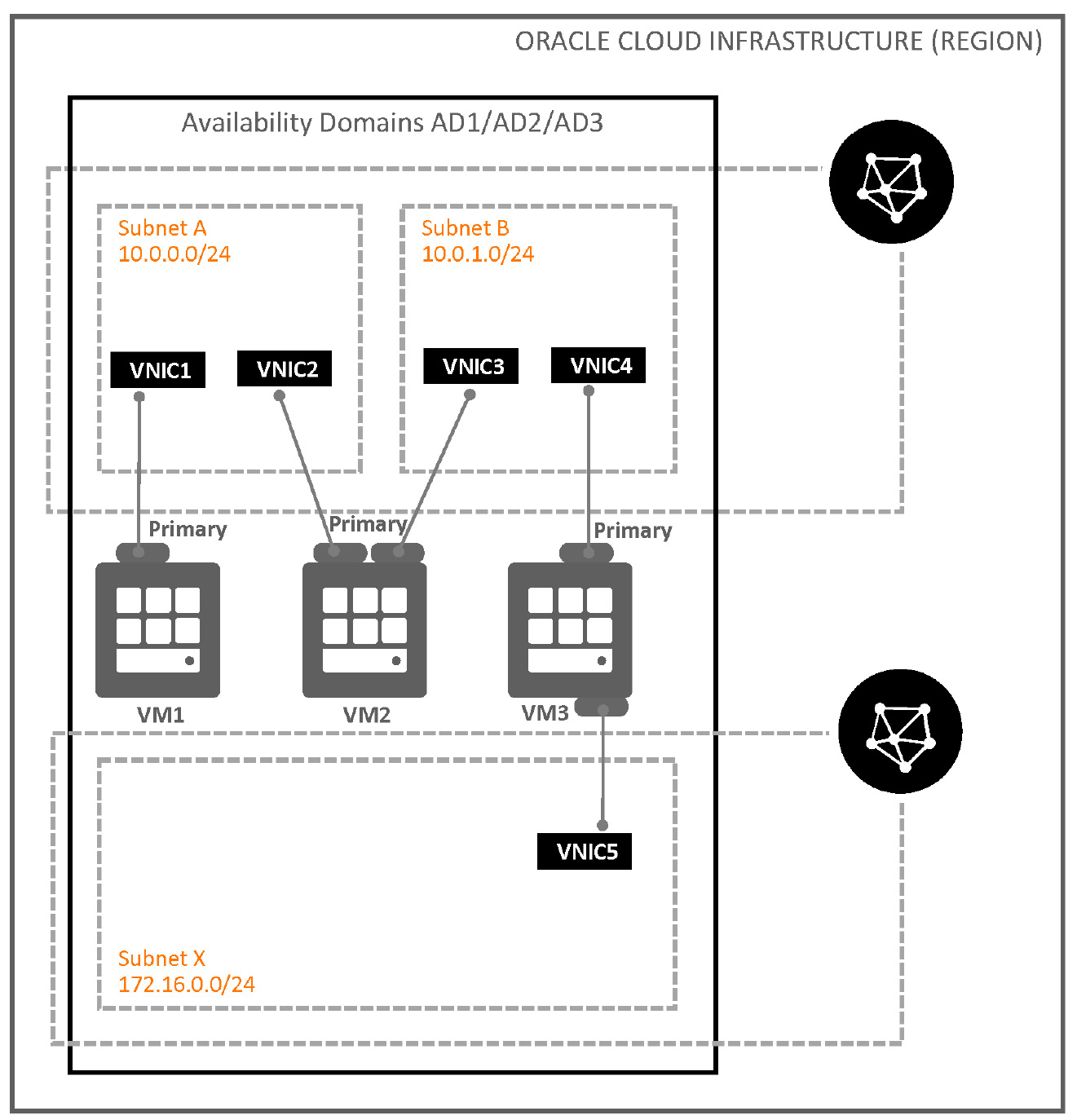


Figure 3.2 – Multiple-VNIC architecture

In the preceding diagram, you can see that you can have secondary VNICs connected to different subnets for different purposes.

**Multiple VNICs on bare metal instances**

Every bare metal instance has two physical NICs. But in the case of first-generation (X5 servers) bare metal hosts, only one physical NIC is active, but in second-generation (X7 servers) bare metal hosts, both of the NIC cards are set to active and each card provides 25 Gbps bandwidth. The first NIC card is set as the primary NIC card.

Each guest VM can get one or more secondary VNICs in the bring-your-own-hypervisor use case. In the case of **single-root I/O virtualization** (**SR-IOV**), **virtual functions** (**VFs**) are used by the hypervisor to provide network access to the guest VMs, where each VF can be configured with the VLAN tag and MAC address of a secondary VNIC. You can see an illustration of this in the following diagram:

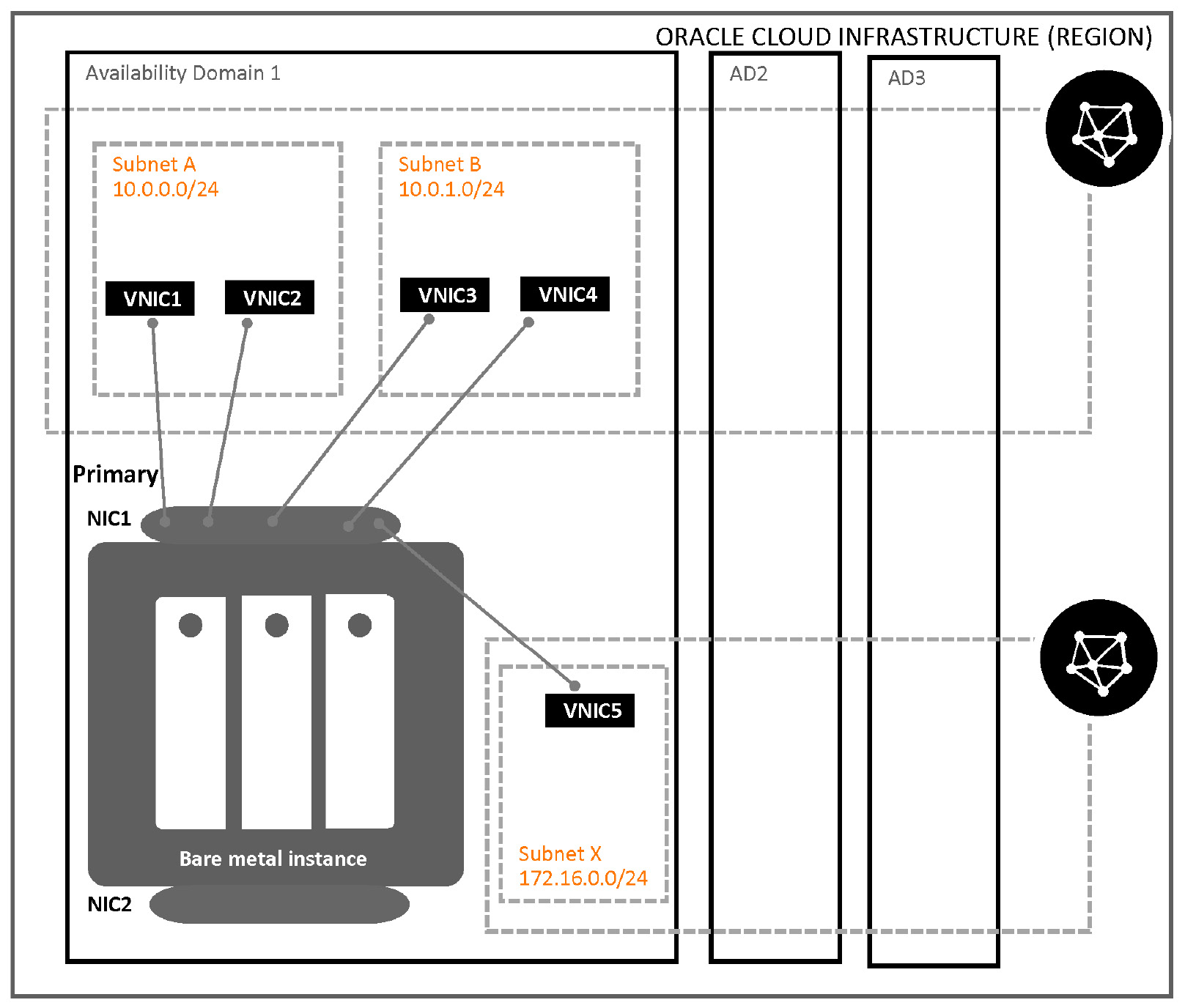


Figure 3.3 – Multiple VNICs on bare metal architecture

In the preceding diagram, you can see that a bare metal instance can have multiple VNICs and that you can use them to connect to different subnets for different purposes.

**Private IP address**

Each instance that you launch in OCI will have a private IP address assigned to it from the subnet IP pool. Although this is the primary IP address that gets assigned to it upon its creation, you can assign additional private IP addresses too. If you assign this instance to a public subnet and have specified that it should fetch a public IP address too, then the same NIC card will get a public IP address.

The following diagram shows an illustration of this:

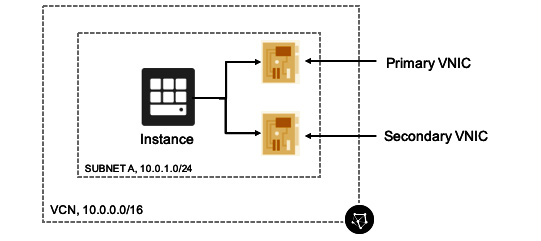


Figure 3.4 – Private IP on VNICs

In the preceding diagram, you can see that this instance has two different NIC cards and how its private IP address is assigned using the subnet's DHCP pool.

**Private IP as a route target**

The private IP address of an instance can be a route target as well. The main use case of this arrangement is when there needs to be a firewall appliance in the middle where every packet is checked at transit. So, your subnet's default gateway would be the private IP address of an instance other than the IP address of the internet gateway. You can see an illustration of this in the following diagram:

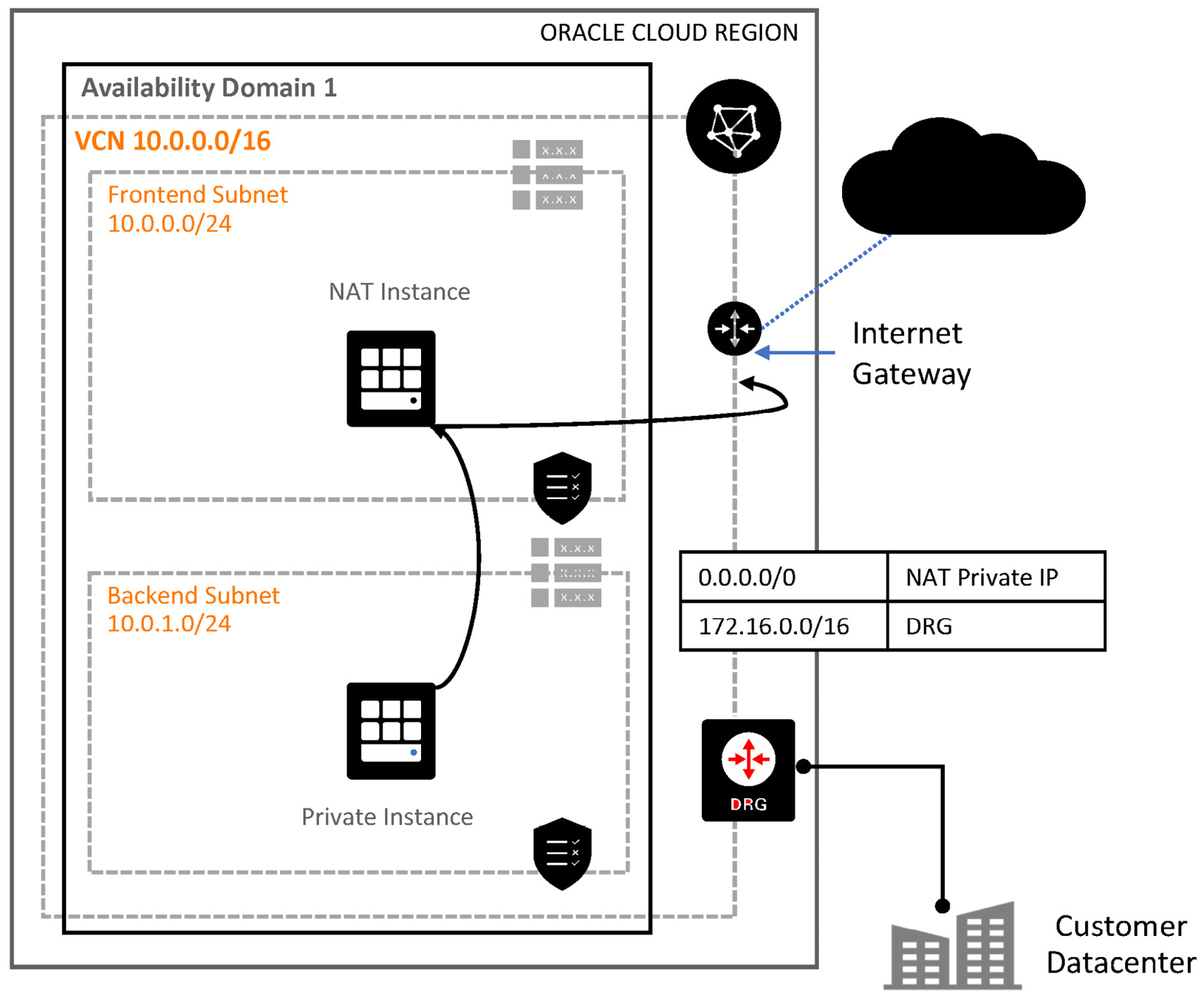


Figure 3.5 – Private IP as route target

In the preceding diagram, you can see how you can use a route rule to use a private IP as a route target and have a man in the middle to scan all of the traffic that an instance sends.

**Public IP address**

A public IP address is an IPv4 address that you provide to an instance so that it is reachable over the internet. You can add multiple public IP addresses to an instance's VNIC. You have the option to choose between the two available types of public IP address:

* **Ephemeral**: An ephemeral IP address is only available until the instance stays up. It's temporary in nature.
* **Reserved**: A reserved IP address is persistent in nature and doesn't depend on the instance life cycle. You also have the flexibility to detach this IP address from one instance and assign it to another instance.

**Internet gateway**

You need an internet gateway if you want your instances to talk to the internet and also receive traffic from internet. By design, you are bound to have only one internet gateway per VCN. To make this internet gateway effective, you need to assign a route rule to the route table of the VCN stating that this internet gateway is the next hop for the VCN. However, if you use a VCN creation wizard, then this will be done for you by the VCN itself.

You can see an illustration of this in the following diagram:

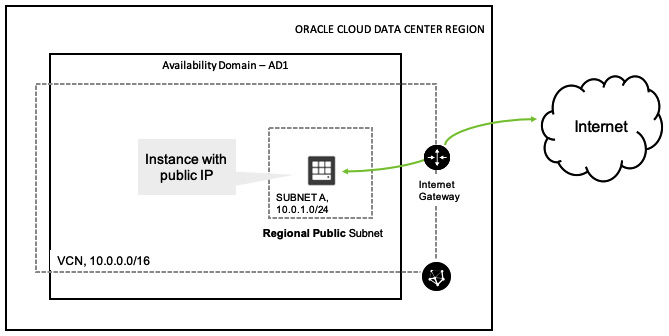


Figure 3.6 – Internet gateway

In the preceding diagram, you can see how you can use an internet gateway to send all traffic out of the VCN to the internet.

**Route table**

A route table, as the name suggests, is a table that has route rules, which allow instances to send traffic out of the VCN and handle traffic coming in to the VCN. You can either specify a destination CIDR or you can choose to have a private IP address as the next hop for the route rule.

At the time of the creation of the VCN, OCI will add a default route table, but you can choose to edit it any time afterwards. However, a route table is not always used, and one of the scenarios where it is not used is when the destination IP address falls under the same VCN CIDR block.

The route table must be updated to reflect your different VCN component types, such as NAT gateways, service gateways, or dynamic routing gateways.

You can see an illustration of the route table in the following diagram:

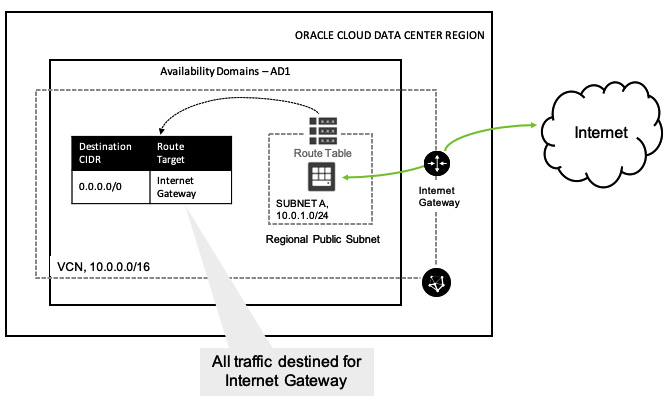


Figure 3.7 – Route table

In the preceding diagram, you can see how you should write a route rule in the route table to make the routing work. In this case, every packet for the outside world will be sent to the internet gateway and then it will go out to the internet.

**Dynamic routing gateway**

When you want to have connectivity for your on-premises equipment, you need to use the **dynamic routing gateway** (**DRG**). To establish the connectivity, you can either choose to connect to it via an IPSec VPN connection or a FastConnect connection. When you create a DRG and add it to the VCN, you need to create a new route rule to send the on-premises traffic via the DRG and not the internet gateway. You can only have one DRG per VCN.

You can see an illustration of this in the following diagram:

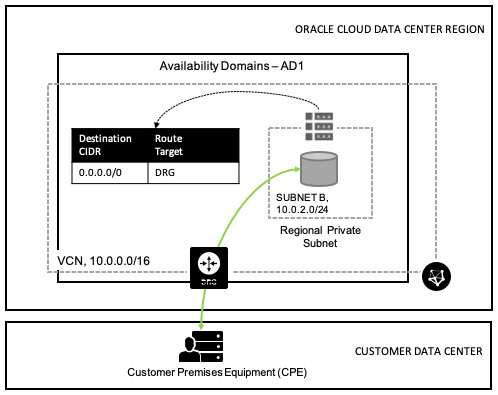


Figure 3.8 – Dynamic routing gateway

In the preceding diagram, you can see how you can use a DRG to connect your on-premises environment to the OCI. This is the same component that will be used to make either a VPN or FastConnect connection. We will discuss this later in the section on connectivity choice.

**NAT gateway**

Think about a use case where you don't want to expose your instance to the internet, but still want to download patches or updates. In this scenario, you just need to use an NAT gateway. It can only allow one-directional traffic—namely, outbound internet access. So, all of those private subnet instances can access the internet via an NAT gateway without having a public IP address.

You can see an illustration of this concept in the following diagram:

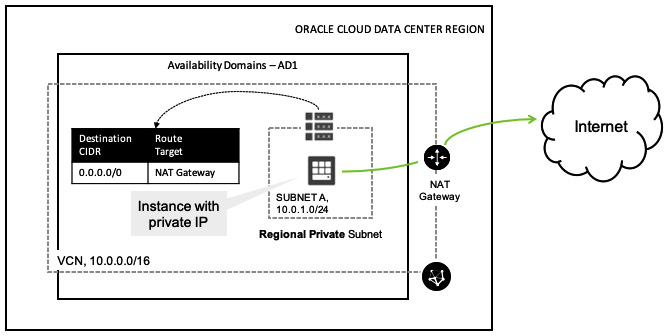


Figure 3.9 – NAT gateway

In the preceding diagram, you can see how you can use a NAT gateway to connect an instance that is connected to a private subnet to the public internet.

**Service gateway**

When you want to access public OCI resources, such as OCI object storage, you typically access them via the public internet. However, if you don't want your traffic to go through the public internet, you can use a **service gateway**.

This way, whenever your instance wants to access the object storage, it will never traverse the traffic through the internet, but will use the OCI network fabric to reach it instead.

You can see an illustration of the service gateway in the following diagram:

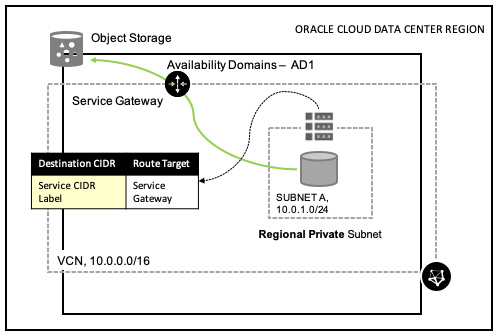


Figure 3.10 – Service gateway

In the preceding diagram, you can see how you can use a service gateway to access OCI object storage without getting routed via the public internet.

**Local peering (within region)**

So far, you have seen that VCNs are disjointed components in OCI. This means that one VCN cannot access other VCN resources. However, in cases where you want to run shared services within a region and host them separately on a different VCN, you need to use a **local peering gateway** (**LPG**) to connect these two VCNs together. This way, your shared resources will be available to the connected VCN's resources via a private IP address. As we are connecting two peering gateways to form the connection of two VCNs, you essentially cannot have overlapping IP addresses in the VCNs.

An illustration of the LPG is as follows:

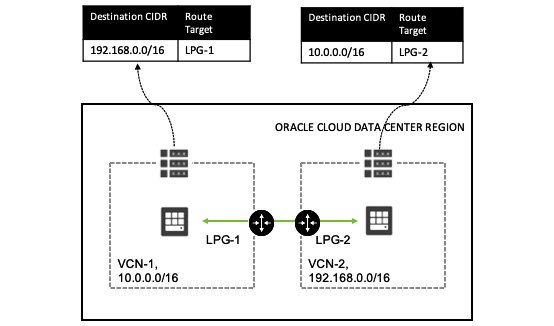


Figure 3.11 – Local peering gateway

In the preceding diagram, you can see how you can connect two VCNs in a single OCI region using a local peering gateway in the middle. You must also have proper routing rules in place to send the packets to the neighbor VCN via a particular LPG.

**Remote peering (across region)**

A local peering gateway is used to connect VCNs within a region inside a single tenancy. This means that it doesn't work when it comes to connecting VCNs across region, or perhaps connecting two different tenancies within a region as well. For this reason, you need to use a **remote peering gateway** (**RPG**). An RPG is implemented with a DRG. Similar to LPGs, you cannot have overlapping IP addresses for RPGs.

You can see an illustration of remote peering in the following diagram:

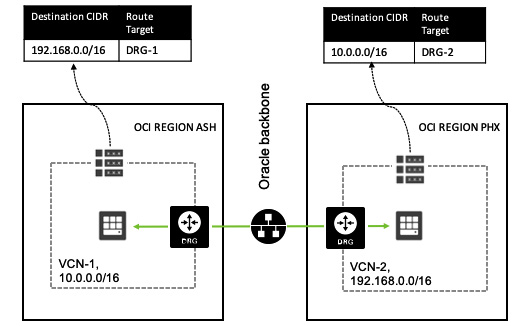


Figure 3.12 – Remote peering gateway

In the preceding diagram, you can see how you can use the Oracle backbone network to establish a remote peering connection between two VCNs located in two different regions.

**Security list**

The firewall is the most important function inside the virtual networking layer. OCI implements this in the form of security lists and the rules contained within them. A security list is associated with a subnet, which means that the rules will be applied to all the instances within a given subnet. You can create and assign a security list after the VCN creation if this is not done by default using the VCN creation wizard.

You can allow or restrict traffic to and from instances either within a VCN or outside of a VCN. Your specified security rules can either be stateful or stateless.

An illustration of this is as follows:

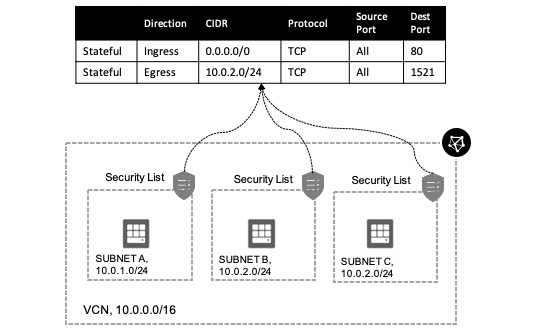


Figure 3.13 – Security list

In the preceding diagram, you can see how security rules are formed inside a security list that is connected to individual subnets.

**Network security group**

Using security lists and rules means that you have to choose resources and group them together to form a resource group on which you want to apply the security rules, as security rules are applied at subnet level. To minimize this scope, you need to use a **network security group** (**NSG**). Using an NSG, you can group resources that have similar characteristics.

You can apply NSG rules to a group of resources. Currently, this can be either compute instances, load balancers, or database instances.

As NSGs are nothing but resource groupings, you can use them as source or destination when writing firewall rules. Oracle's recommendation is to use the NSGs, which helps you to separate your application's security posture from the VCN.

You can see an illustration of this in the following diagram:

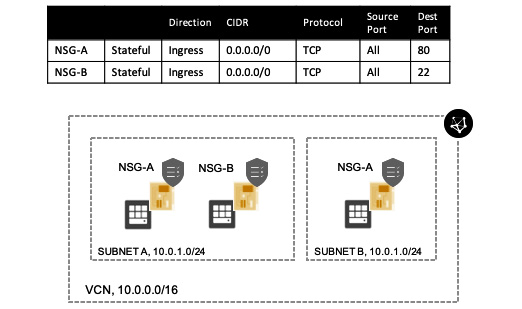


Figure 3.14 – NSG

The best solution is to use security lists and NSGs together; this way, you can create a union of the rules from both security lists and network security groups.

You can see an illustration of this combination in the following diagram:

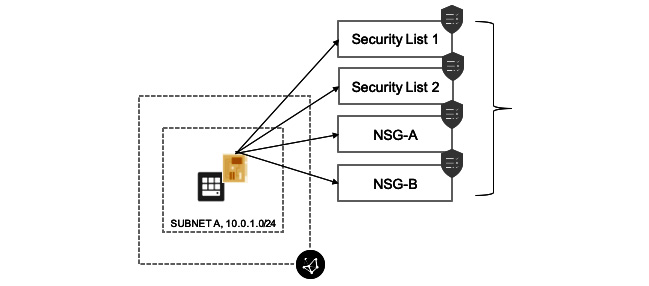


Figure 3.15 – Combination of an NSG and a security list

In the preceding diagram, you can see how we group together the NSG and security list to create a more robust security solution.

**Stateful and stateless security rules**

Stateful security rules track the response of the incoming traffic and whether matches allow the traffic, irrespective of the egress rules. The same approach is adopted when traffic is outbound from instances.

You can see an illustration of this in the following diagram:

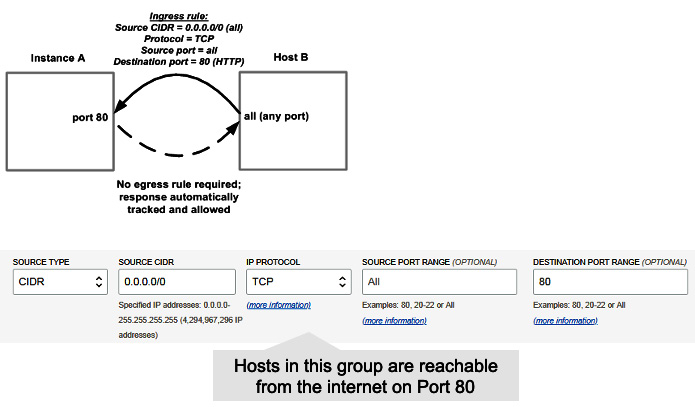


Figure 3.16 – Stateful security rules

Stateless rules work in exactly the opposite way. This means that without a matching egress rule, ingress traffic is not processed. So your incoming traffic will not be allowed by default if you don't have a matching egress rule.

Two main use cases of stateless rules are *load balancing* and *big data*. You can see an illustration of stateless rules in the following diagram:

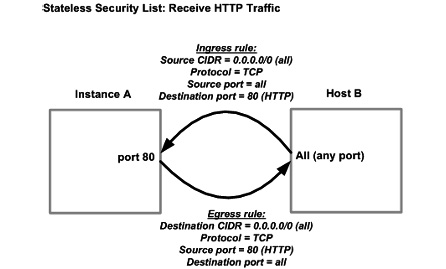


Figure 3.17 – Stateless security rules

In the preceding diagram, you can see that instance A only responds to an incoming packet at port 80, but host B responds to any port for the incoming requests.

**Default VCN components**

VCN has some default components:

* Default route table
* Default security list
* Default set of DHCP options

These components can't be removed, but their contents can be changed.

As the name implies, the DHCP options' job is to provide an IP addresses to the instances automatically at the time of boot up. An illustration of the default VCN components is shown in the following diagram:

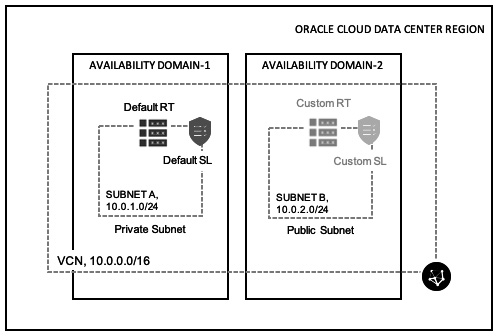


Figure 3.18 – Default VCN components

The preceding diagram shows the default components of the VCN that get created as part of the VCN creation wizard.

**Reviewing the VCN components**

Let's take a recap of the VCN components and their details:

* Subnets can have one route table and multiple security lists associated with it. The route table defines what can be routed out of the VCN.
* Private subnets are recommended to have individual route tables to control the flow of traffic outside of the VCN.
* All hosts within a VCN can route to all other hosts in a VCN (no local route rule required).
* Security lists manage connectivity from north to south (incoming/outgoing VCN traffic) and from east to west (internal VCN traffic between multiple subnets).
* OCI follows a white list model (you must manually specify white listed traffic flows); by default, things are locked down.
* Instances cannot communicate with other instances in the same subnet until you permit them to!
* Oracle's recommendation is to use the NSGs, which help you to separate your application's security posture from the VCN.

You can see a top-down view of the VCN in the following diagram:

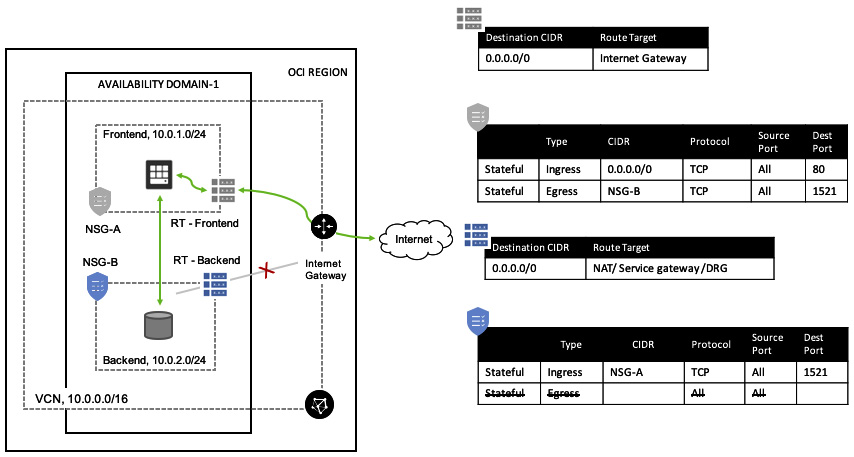


Figure 3.19 – Overall VCN

In the preceding picture, you can see a top-down view of the VCN, showing all of the aforementioned components. It is connected to the internet by an internet gateway, it has the VCN construct and the subnet, a security list, and rules, as well as routing rules. In the next section, you will see how you can connect to OCI.

**Connection choices**

You need to connect to OCI resources in order to access them. Without a connection in place, you cannot connect to your workloads. You can connect to your OCI resources to and from OCI using three different methods:

* Connecting through the public internet
* Connecting through a VPN
* Connecting through FastConnect

Let's discuss them in the next section.

**Connecting through the public internet**

Accessing an OCI instance over the public internet is pretty straightforward, and doesn't require much effort. You would need to go through the following steps to get internet access to and from the OCI instance:

1. Create a VCN and provide a CIDR range.
2. Create an internet gateway.
3. Create a route rule with traffic to the internet gateway (for all IP addresses, **0.0.0.0/0**).
4. Create security list rules to allow traffic.
5. Make sure that each instance's firewall allows the traffic as well.
6. Create a public subnet within a specific AD with the route table and security list.
7. Create an instance with a public IP address within the subnet.

You can see an illustration of this in the following diagram:

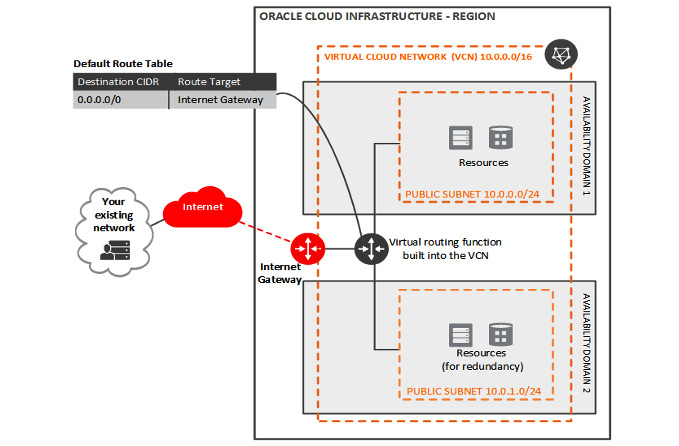


Figure 3.20 – Accessing an OCI instance over the public internet

VCN has provided a workflow to make this simple. Let's go through the following steps to create an OCI instance and access it over the public internet:

1. Sign in to the OCI console.
2. Open the navigation menu. Click **Networking** and select **Virtual Cloud Networks**.
3. Check whether you are in the correct compartment in the **Compartment** list.
4. Click **Start VCN Wizard**.
5. Select **VCN with Internet Connectivity**, and then click **Start VCN Wizard**.

Enter the values for the following fields:

-**VCN Name**

-**Compartment**

-**VCN CIDR Block**

-**Public Subnet CIDR Block**

-**Private Subnet CIDR Block**

Accept the default values for any other fields. You can see a sample screenshot of the workflow in the following image:

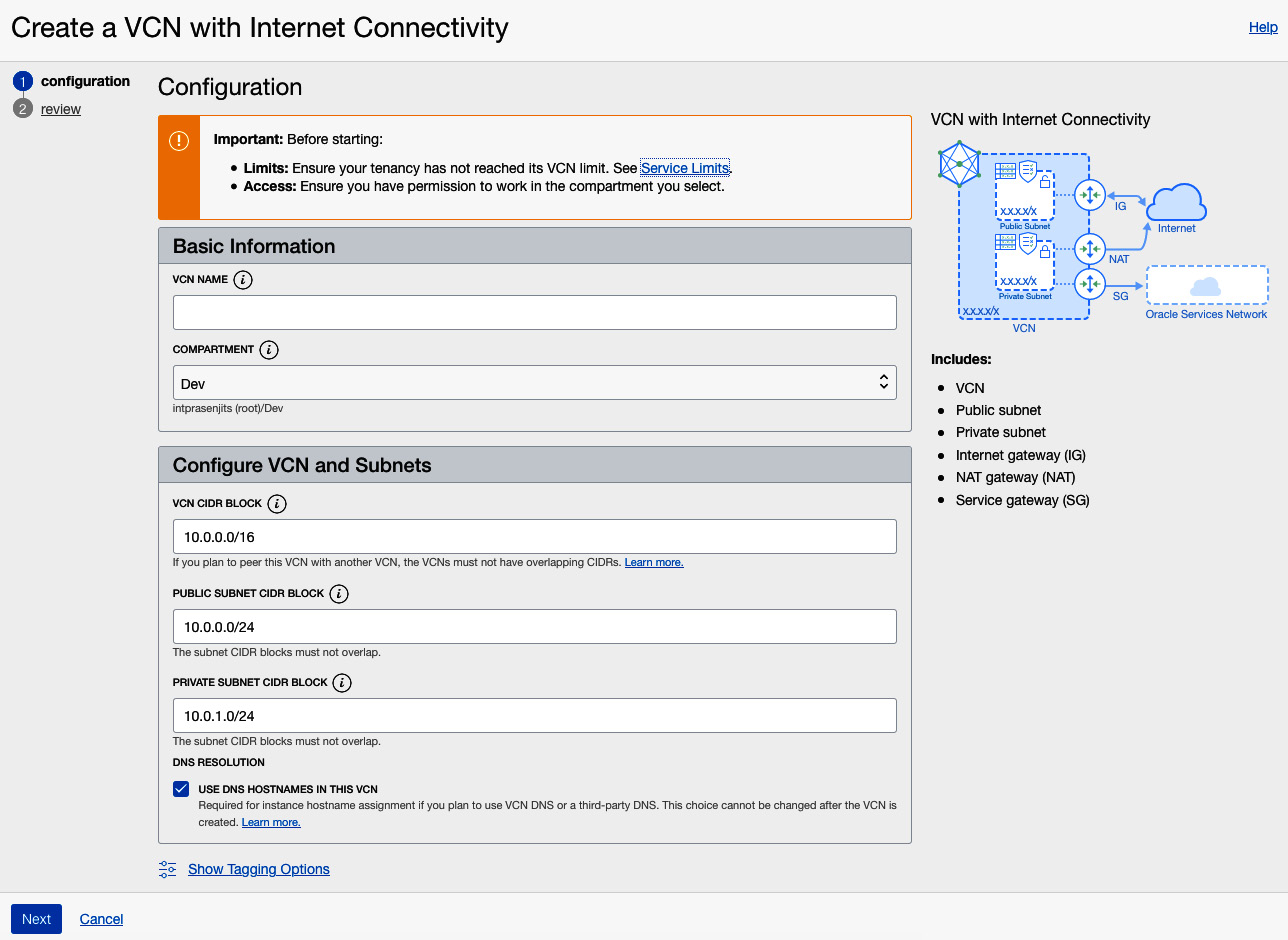


Figure 3.21 – VCN creation wizard

1. Click **Next**.
2. Click **Create** to start the workflow.
3. Once the process completes, click **View Virtual Cloud Network**.

In this section, you learned how to create a VCN and its components. Let's create an instance now and connect it to this newly created VCN.

**Creating an OCI compute instance on a public subnet**

To create an OCI compute instance in the private subnet, follow these steps:

1. In the console, under the navigation menu, select **Compute**, and then select **Instances**.
2. Click **Create Instance**.
3. Provide a name for the instance—for example, **Private-Instance**.
4. Select the compartment in which you want to place the instance.
5. Choose the availability domain in which you want to place the instance.
6. Select a shape for the virtual machine—for example, **VM.Standard.E3.Flex**.

You can see a sample screenshot of the workflow in the following image:

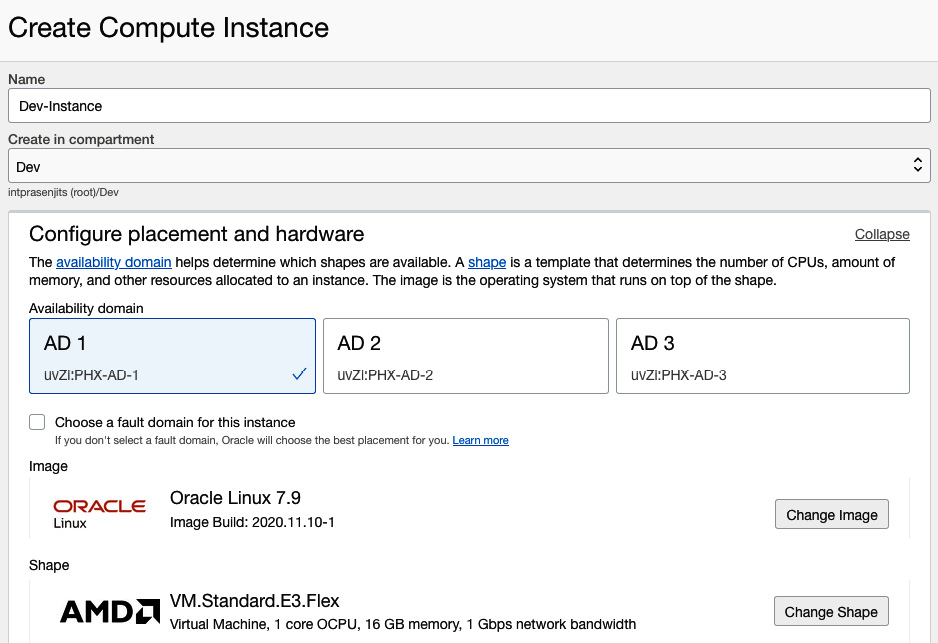


Figure 3.22 – Instance creation wizard

1. In the **Configure networking** section, select the compartment in which your VCN resides. This is typically the same compartment in which you're deploying this VM.
2. Select the **VCN**.
3. Select the compartment in which the subnet resides.
4. Select the public subnet.
5. Make sure that you have the **Assign a public IPv4 address** option selected.
6. In the **Add SSH keys** section, let OCI create an SSH key pair, and save the private key and the public key. You can see a sample screenshot of the workflow in the following image:

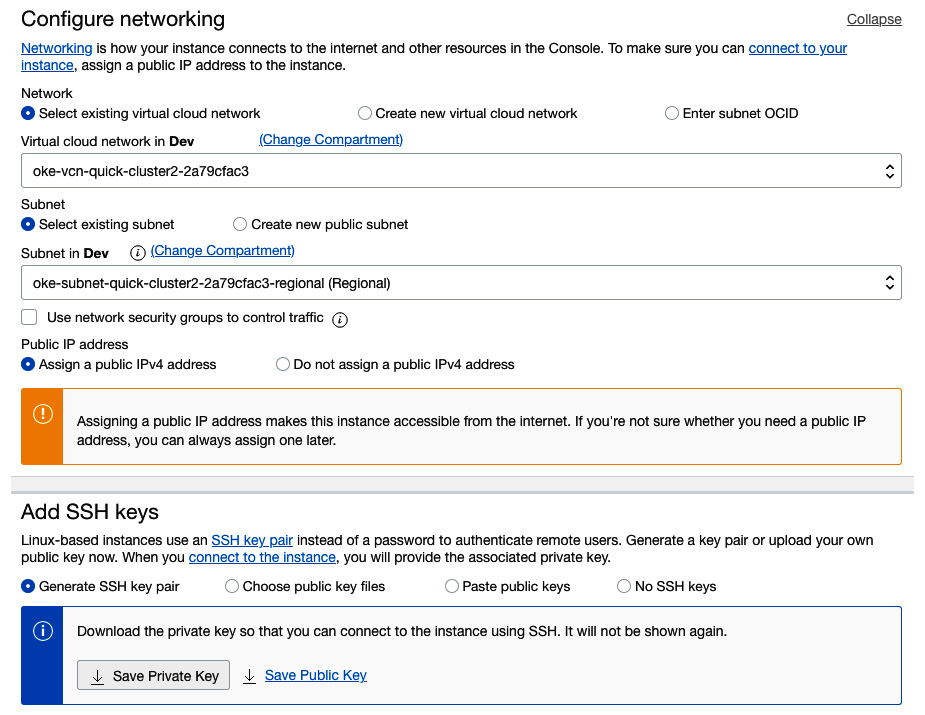


Figure 3.23 – Instance creation wizard with networking selection

1. Click **Create**.

After the instance is created, the instance details page is displayed. Make a note of the public IP address.

In this section, you learned how to create an instance and connect it to the public-facing VCN subnet. Now let's look at how you can connect to it using SSH.

**Accessing the instance over the public internet**

From your local environment, use SSH to connect to the OCI instance on the public subnet (for example, **JumpHost**). But before you do this, you need to fix the SSH private key file permission. Let's run the following command to fix the permission and then connect to it using SSH:

chmod 400 <path-of-the-ssh-private-key>

ssh -i <path-of-the-ssh-private-key> opc@<public-ip-address>

**Connecting through a VPN**

The OCI VPN provides you with an IPsec connection to create a secure network tunnel between OCI and the on-premises environment. However, the network speed is not guaranteed, and depends on the internet speed.

As an architecture with no single point of failure, OCI provides you with physically redundant VPN tunnels. This means that OCI will provision two different circuits in two different availability domains (where more than one AD exists) to make a redundant VPN connection.

There are some limitations to this as well.

* An OCI VPN can only support an **Internet Key Exchange v1** (**IKEv1**) using a shared secret.
* Currently, only static routes are supported (**Border Gateway Protocol** (**BGP**) is not supported). When you create an IPsec connection, static routes are added. These static routes can't be modified after an IPsec connection has been created.

You can see an illustration of this in the following diagram:

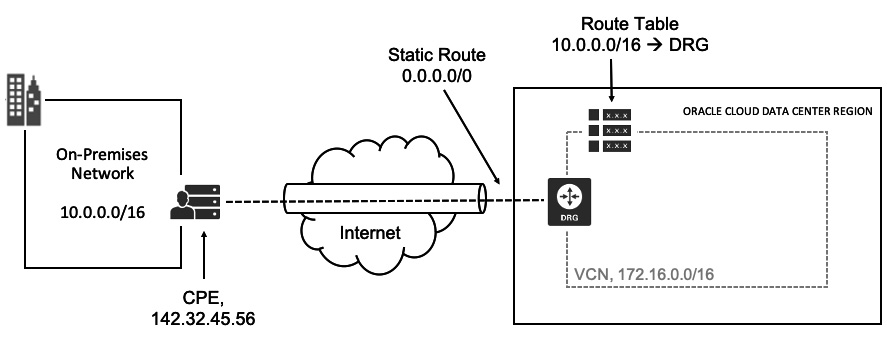


Figure 3.24 – OCI VPN setup

Customers can use a single **Customer Premises Equipment** (**CPE**) or a dual-head CPE for redundancy. You can see an illustration of the single-head CPE for the VPN connection in the following diagram:

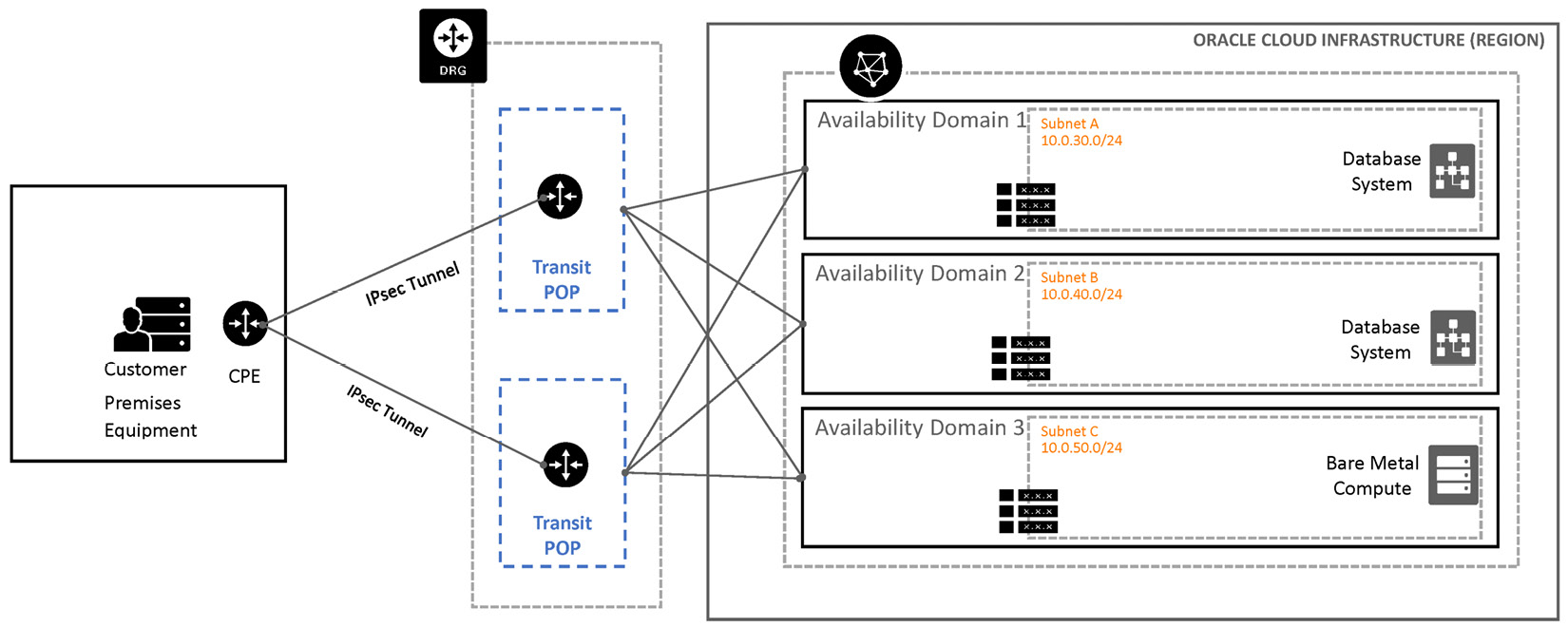


Figure 3.25 – VPN setup using single-head CPE

Additionally, you can configure two CPEs to create a **highly available** (**HA**) deployment in your on-premises network. The following diagram shows the recommended HA VPN deployment with three configured tunnels per CPE. This is illustrated in the following diagram:

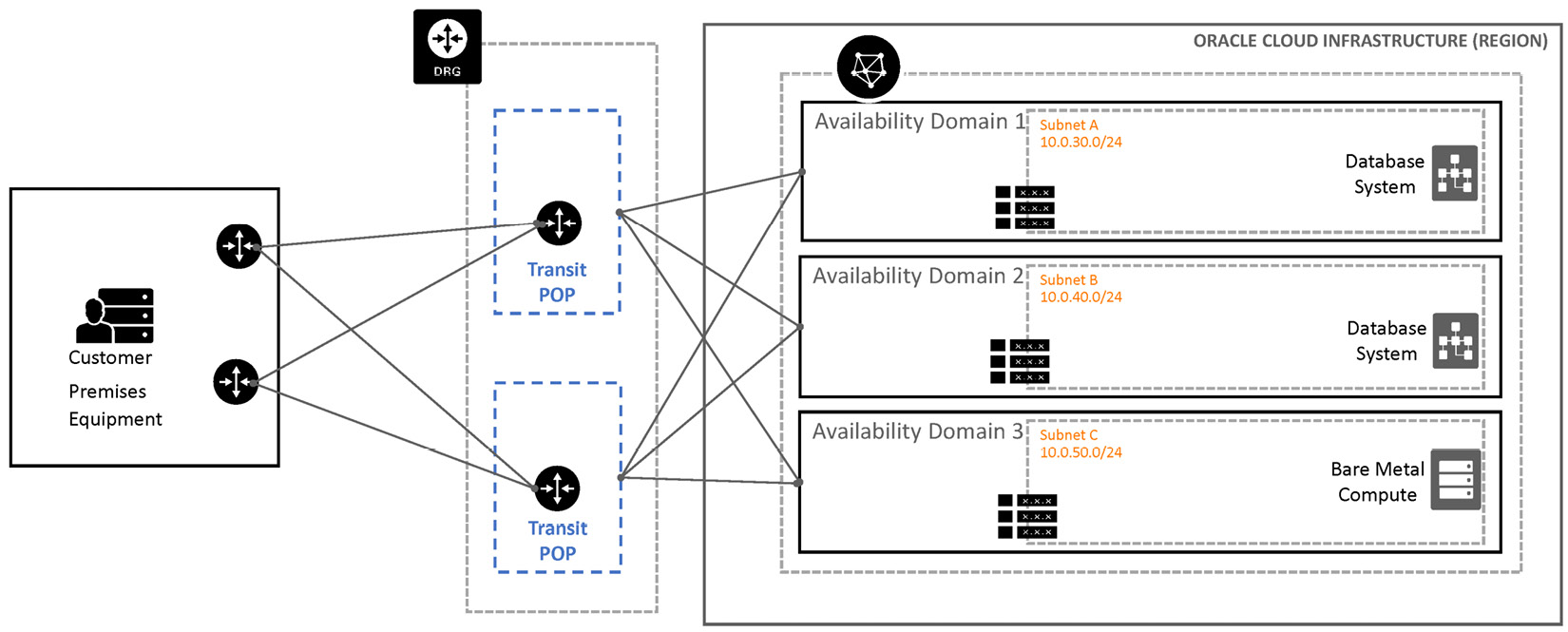


Figure 3.26 – VPN setup using multi-head CPE

The customer can use private subnets as well by using a *bastion host* connected to a public and a private subnet (multiple NICs). You can use a private subnet with a VPN connection to on-premises subnets as well. You can see an illustration of this in the following diagram:

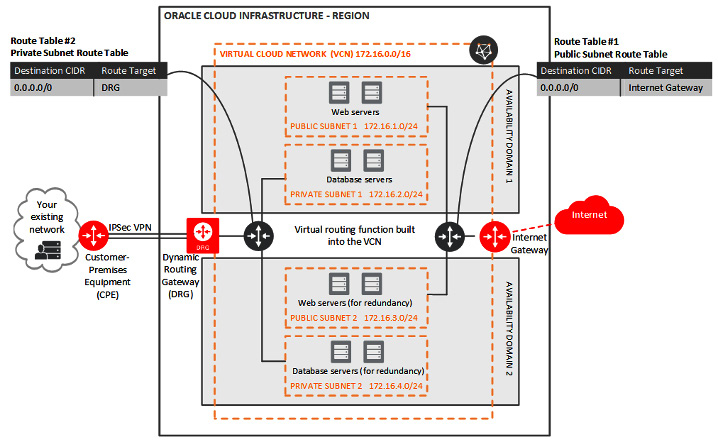


Figure 3.27 – VPN with a public and private subnet

In the preceding diagram, you can see how a customer can establish a VPN connection and then use a private and public subnet to differentiate the workload connectivity option.

**Connecting through FastConnect**

As you have noticed, the IPsec VPN connection doesn't provide guaranteed bandwidth, and depends on the public internet speed. If you want a dedicated and high-bandwidth connection, you need to choose a FastConnect connection. Customers can connect to OCI directly or via pre-integrated network partners. OCI supports port speeds of 1 Gbps and 10 Gbps increments. It doesn't have any charges for inbound/outbound data transfer and uses the BGP protocol for traffic routing. You can see an illustration of the FastConnect connection in the following image:

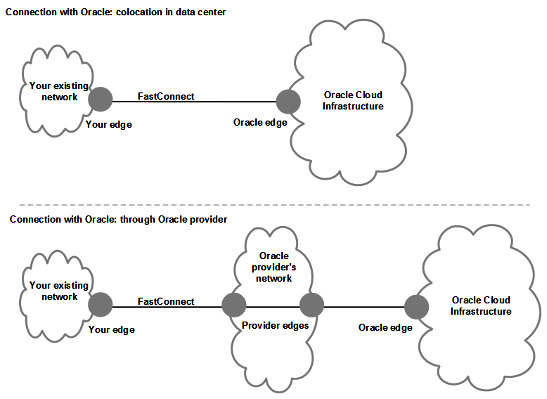


Figure 3.28 – FastConnect scenarios

The preceding image shows the connection options using either a co-located datacenter or a provider connection in the middle.

Let's now discuss the virtual circuit, as this is something that you will create to make the connection.

**Virtual circuit**

The virtual circuit is the key component of establishing the single and logical connection between the customer's on-premises network (typically using an edge router) and the DRG deployed on OCI. To establish this virtual circuit, you need information from the customer, Oracle, and a service provider. You can create more than one virtual circuit to segregate different organization functions. This segregation also provides redundancy. FastConnect uses the BGP routing protocol to establish this connection.

**FastConnect using scenarios**

Customers can choose between two kinds of peering for FastConnect connection.

**Private peering**:

* Extension of the on-premises network to the OCI VCN
* Communication across the connection with private IP addresses

**Public peering**:

* To access public OCI services, such as object storage, OCI console, or APIs over dedicated FastConnect connection
* Doesn't use DRG

You can see an illustration of the private and public peering for FastConnect connection in the following image:

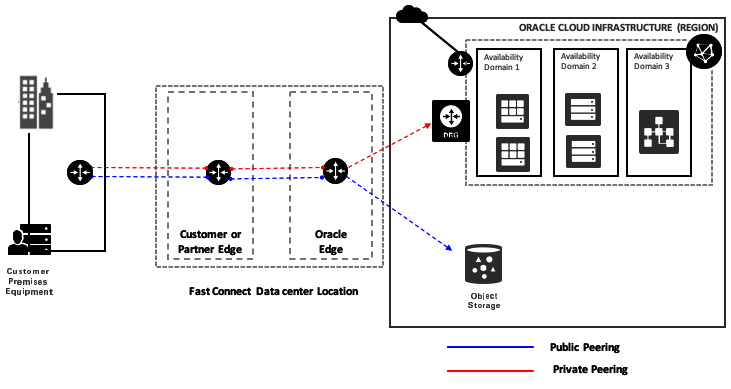


Figure 3.29 – FastConnect private and public peering

In this image, you can see that you can use public peering to access more public-facing OCI services, such as OCI object storage, and use private peering to access the other resources, such as instances, via a DRG.

In this section, you have learned about the different ways to connect to the OCI. In the next section, we will discuss how you load balance those incoming connections to your backend servers.

**Load balancer**

A load balancer's responsibility is to balance the network traffic between the clients and the backend. However, they do a number of other things as well:

* Service discovery
* Health checks
* Algorithms

Let's look at the benefits of using a load balancer:

* **Fault tolerance and HA** – You can use different algorithms to load balance the backend servers to provide fault-tolerant and highly available web services.
* **Scale** – You can add more backend resources when the load increases.
* **Naming abstraction** – You can use a load balancer and provide FQDN out of the load balancer listener IP address without knowing the instance's private IP address.

These are the generic benefits and use cases of any load balancer. Let's look at what the OCI load balancer does.

OCI provides this load balancer as a managed service offering for you to provide resilient web service experience to your client. It supports both private and public load balancers and also supports TCP, HTTP/1.0, HTTP/1.1, HTTP/2, and WebSocket protocols. For security, you can choose between SSL termination, end-to-end SSL, and SSL tunneling. Apart from standard layer-3/4 features, OCI also supports layer-7 capabilities, such as session persistence and content-based routing.

Unlike other cloud providers' models of using two different load balancers for network load balancing and application load balancing, OCI provides a single load balancer to achieve both of these objectives. For guaranteed and prewarm bandwidth requirements, OCI also provides different shapes of load balancer, such as 10 Mbps, 100 Mbps, 400 Mbps, and 8 Gbps. You have full flexibility to switch to a different shape at any time without any disruption.

In the next section, let's discuss the different formats of the available load balancers. We will also discuss the layer 3/4 load balancer and how it is different to the layer 7 load balancer, which uses cookie-based session persistence, virtual host-based routing, and so on. We will also discuss how to secure the connection using SSL termination.

**Public load balancer**

As you can tell from its name, the public load balancer is for public use, which means that you can use the public load balancer to receive traffic using a public IP address before sending it to the backend servers. As it's a regional service, you can either provision a regional subnet or provide multiple subnets from different ADs.

Using the principle of fault tolerance, OCI creates two different load balancer instances and deploys them in multiple availability domains. This way, it can switch to the other load balancer instance in case a disaster strikes, such as when one AD becomes unavailable.

An illustration of the public load balancer is shown in the following figure:

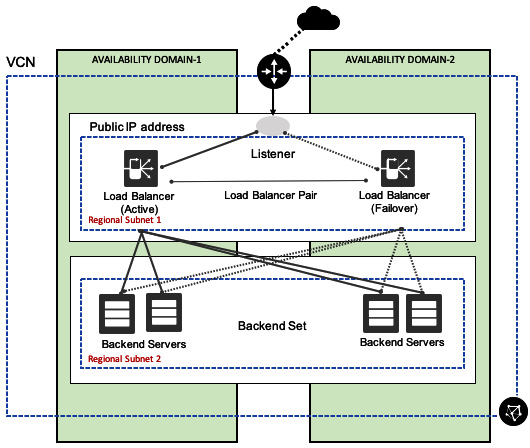


Figure 3.30 – Public load balancer

In the preceding image, you can see that there are two load balancer heads that have been placed on a single public regional subnet, which is in turn load balancing incoming traffic to four different backend servers that are connected to another public regional subnet.

**Private load balancer**

A private load balancer balances the load between the instances that are hosted in the private subnet. This means that you can only access this load balancer from the same VCN on which it is hosted. Using the principle of fault tolerance, OCI creates two different load balancer instances, and deploys them in multiple ADs. This way, it can switch to the other load balancer instance in case a disaster strikes, such as when one AD becomes unavailable.

An illustration of the private load balancer is shown in the following figure:

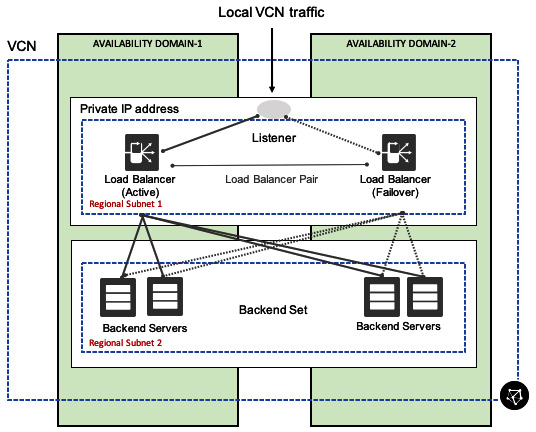


Figure 3.31 – Private load balancer

From the preceding image, you can see that there are two load balancer heads that have been placed on a single private regional subnet, which is in turn load balancing incoming traffic to four different backend servers that are connected to another private regional subnet.

**Load balancing policies**

There are three load balancing policies that OCI supports:

* **Round robin**: This is the default algorithm, and sends the incoming traffic randomly to the backend server set. Once it sends the first round of packets, it repeats the same order again.
* **IP hash**: The incoming IP address will be used to create a hash to send the traffic to the same backend server.
* **Least connection**: This algorithm chooses the backend server that has the fewest active connections to send the incoming traffic to.

While layer-3/4 load balancing happens over the policy algorithm and weight, HTTP load balancing considers cookie-based persistence.

**Health check**

The health check performs the availability check of the backend server. You can activate the health check for the following:

* Backends
* Backend set
* Overall load balancer

A load balancer IP can have up to 16 listeners (port numbers). Each listener has a backend set that can have 1 to *N* backend servers. The health API provides a four-state health status (ok, warning, critical, and unknown). The health status is updated every three minutes.

The health API displays the health of a load balancer instance in relation to its backends. Customers can utilize the health API to build their own notification and monitoring systems or integrate it with their existing systems.

**SSL handling**

Certificates are the main requirement for using SSL on the OCI load balancer. Certificates will contain the public key, private key, and CA portion of it, but you can upload and attach the certificate bundle with the listener after it is created. However, for the best user experience, you should upload it to the portal before you create the listener or backend set.

You need to make sure that the OCI load balancer does not have the capability to create any SSL certificates at all. This means that you must have it created beforehand. You can either use a CA to generate the certificates or you can use self-sign certificates.

Let's look at the three different types of SSL handling of an OCI load balancer:

* **SSL termination** – In this mode, your connection is not encrypted between the load balancer and the backend set, as the termination of the SSL connection happens at the load balancer layer.
* **SSL tunneling** – In this mode, you will have an SSL connection only between the load balancer and the backend servers.
* **End-to-end SSL** – As you can tell from the name, this is the most efficient way of load balancing as it not only encrypts the traffic from your client to the load balancer, but also from the load balancer to the backend servers.

**Session persistence**

Session persistence sends the incoming packets from a single originating source to a single backend server. For URLs where you rely on caching, such as in logging sessions or shopping carts, having a single backend server responding to the same originating client can help boost performance. For this, session persistence is the key feature.

OCI load balancer supports cookie-based session persistence. This works at layer 7 (HTTP). However, session persistence doesn't work if the client doesn't accept cookies.

You can configure cookie-based session persistence either on the OCI load balancer level or at the backend set level. You need to specify two parameters while configuring this. These are as follows:

* **Cookie name**: Either this can match the exact name of the cookie or it can match all to establish session persistence.
* **Fallback**: The fallback setting decides what the load balancer should do in the case that the original backend server that the session persistence was originally established with is not available.

**Request routing – virtual hostnames and path routing**

The request routing feature allows users to route traffic based on certain request parameters:

* Hostname (HTTP requests): **virtual hostnames**
* The HTTP(s) request's path: **path routing**

Let's look at both of these in detail.

**Virtual hostnames**

A virtual hostname is another way to provide an identifier to any of the *OCI LB listeners*. So, you can set different hostnames for different applications that get served by your backend set. A virtual hostname only operates at layer 7 (HTTP and HTTPS) and does not support layer 4 (TCP). Enterprises use a single load balancer to host multiple apps, with each app identified by hostname. There are three advantages to this approach:

* **Single associated IP address** – This makes the process of network-ACL configuration simpler.
* **Single bandwidth/shape** – Multiplexing many apps in a single shape/size provides customers with the flexibility of better managing the aggregate bandwidth demands, and also improves overall utilization.
* **Shared backend set definition** – This enables the administrative simplification of managing the set of backends under a single resource.

Without virtual hostnames, customers have to instantiate different load balancers for each application. This would lead to administrative pain points related to IP address management and network/ACL configuration.

The OCI load balancer service supports three matching variants for virtual hosts' hostnames in the following order:

1. Exact matching (for example, **login.example.com**)
2. Longest wildcard starting with asterisk (**\*.example.com**)
3. Longest wildcard ending with asterisk (**login.example.\***)

**Path routing**

Typically, applications have different routing path sets for various different modules (for example, **/login**, **/app**, or **/static**), and requests for each of these endpoints need to be routed to different backend sets. Without path-based routing, different endpoints would need to be represented by different port numbers.

If the customer wants to differentiate traffic between two backend sets, their options would be either using different listeners for each backend set, which requires a different port for each listener, or different load balancers for each set of traffic.

Neither of these workarounds are suitable as they require clients to hit different endpoints, either a different port or a different load balancer altogether.

You need to use *path routing* for these scenarios. The load balancer matches this path routing string with an incoming URL path to decide which backend set it has to send the request to. A *path route rule* has four different pattern-matching types:

* **EXACT MATCH** – This rule applies when an exact match is found for the path string:

^<path\_string>$

* **FORCE\_LONGEST\_PREFIX\_MATCH** – This rule applies when a longest match of the beginning string is found at the URI path:

<path\_string>.\*

* **PREFIX\_MATCH** – This rule applies when a given string matches the beginning portion of the incoming URI path:

^<path\_string>.\*

* **SUFFIX\_MATCH** – This rule applies when a given string matches the ending portion of the incoming URI path:

.\*<path\_string>$

All of these route rules are only applied at layer 7, which is HTTP and HTTPS. You have no control over layer 4, which consists of the TCP packets using these rules. The maximum number of path rules that you can have is 20, while you can have 1 path route set per listener.

You have seen how the OCI load balancer handles the incoming traffic and provides various options to handle and secure the connections. In the next section, we will discuss how these traffic flow help you to diagnose connectivity issues and how important flow logs are.

**VCN flow logs**

VCN flow logs keep detailed records of every flow that passes through your VCN and present this data for analysis in the OCI Logging service. The data includes information about the source and destination of the traffic, along with the quantity of traffic and the *permit* or *deny* action taken, based on your network security rules. You can use this information for network monitoring, troubleshooting, and compliance. Through integration with the logging service, you can view, search, and retrieve log files.

Customers can use the *service connector* to create the object storage bucket and archive the data, as well as use the Oracle Streaming service to deliver the data to a third-party log analysis tool of their choice.

At a high level, the following steps show how flow logs will collect data:

1. Flow logs will send data every minute to OCI Logging for VNICs.
2. OCI Logging will index flow the log data every 10 mins and serve it to the customers.
3. Data sent to the logging service will be indexed and then shown via the logging UI.
4. Users can search to show all IP addresses, and these addresses can be exported once via CSV or JSON.
5. Data is retained for 30 days by default, and customers can choose to extend this period to up to 6 months.
6. If customers need to retain data for archival purposes, they must use a service connector to use object storage or streaming:

**Object storage** – The service connector will read data from indexed logs and then export the flow log data every 10 mins or every 10 GBs, whichever comes first. It uses OCI Object Storage to store this flow log.

**Streaming** – Integration with the streaming service will use the same 1-minute interval. Flow logs will send data every minute to the OCI Logging service, and then the customer can set up a connector using the *service connector* to connect it to the streaming service.

Customers can gain insight into their network traffic utilization by consulting flow logs for the subnets in their VCNs.

A flow log file is a list of flow records. Each record indicates the number of TCP or UDP packets that were seen during a collection window for a particular tuple of source and destination IP addresses and ports. It also includes time information, along with the protocol code, byte count, and whether the packets were accepted or rejected due to security rules.

To enable flow logs on a subnet, the customer goes to the central OCI Logging UI and selects the subnet as the resource they wish to monitor. Successfully enabling flow logging on a subnet will result in log files being stored in their chosen bucket and being viewed via the UI.

Each instance in a VCN has one or more VNIC The networking service uses security lists to determine what traffic is allowed through a given VNIC. When you attach a VNIC to a subnet, it will be subjected to all the security lists associated with that subnet.

**Configuring VCN flow logs**

VCN flow configuration is done through the OCI Logging service. Let's configure the VCN flow logs:

1. Log in to the OCI console.
2. Open the menu. Select **Logging** and then select **Log Groups**.
3. Click on **Create Log Group**.
4. Provide a **Name** and **Description** and click on **Create**.

It will take you to the details of the Log Group. From here, select **Logs** from the **Resources** section. Here, we need to **Enable Service Log**.

1. Click on **Enable Service Log**.
2. Select **Virtual Cloud Network (subnets)** as the **Service**.
3. Select the subnet that you want to monitor from the **Resource** dropdown.
4. Under the **Configure Log** section, select **Flow Logs (All records)** under the **Log Category**.
5. Provide a **Log Name** and click on **Enable Log**. You can see an example of this in the following screenshot:

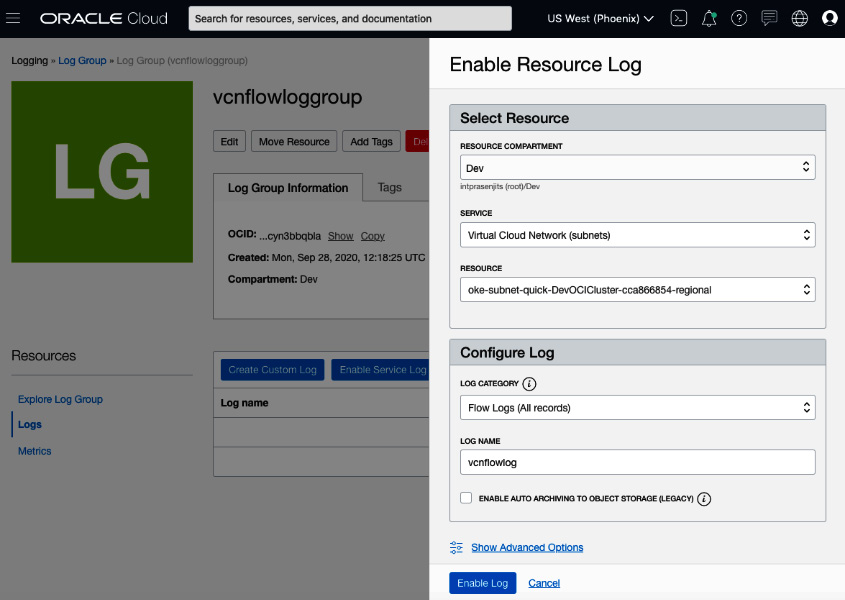


Figure 3.32 – Enable VCN flow logs

1. After five minutes or so, you will see that the VCN flow logs have started pouring in to the **Explore Log** section. You can see an example of this in the following image:

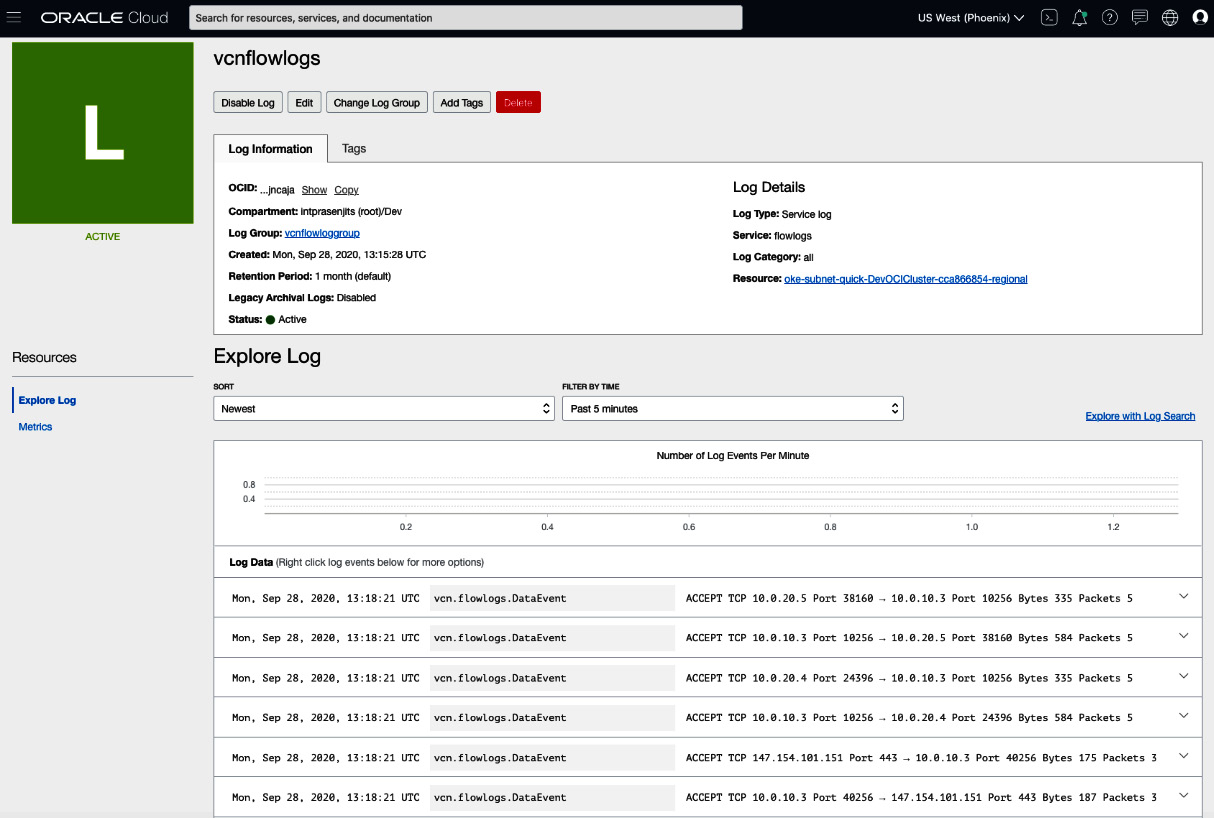


Figure 3.33 – VCN flow log data

1. From here, you can click on the **Explore with Log Search** button to go to the **Search** section under **Logging**.
2. Here, you can filter the logs by IP or by action taken on the packer flow.
3. You can click on the **Visualize** tab and create a visualization using **INTERVAL** and **GROUP BY**. You can see an example of this in the following screenshot:

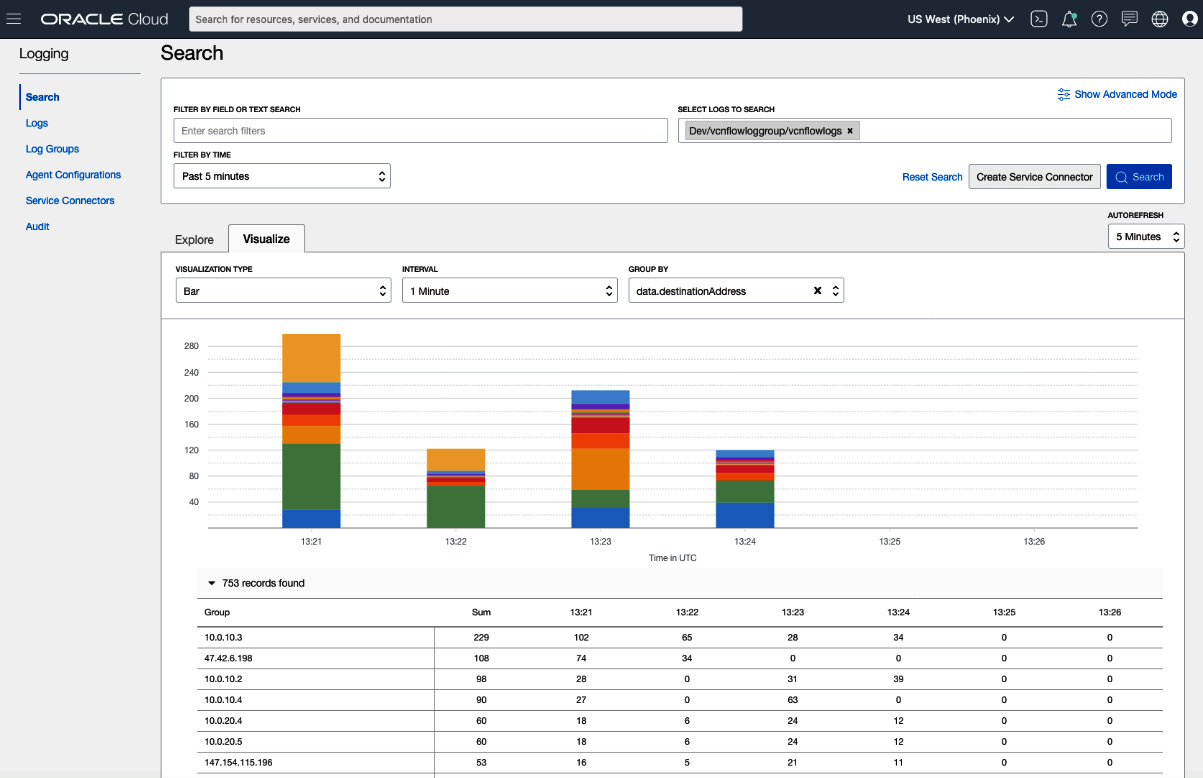


Figure 3.34 – VCN flow log data visualization

So, you can see that by using VCN flow log data, you can find anomalies easily and visualize what is happening in your VCN.

**Summary**

In this chapter, you learned the foundation of OCI–namely, VCN. We explained the overall VCN architecture, along with the VCN components and different connection options. You also learned about the concepts of the OCI load balancer and how you can use the VCN flow log for troubleshooting and finding anomalies.

In the next chapter, you will see the various compute choices that you have in OCI and how you can leverage them in your use cases.