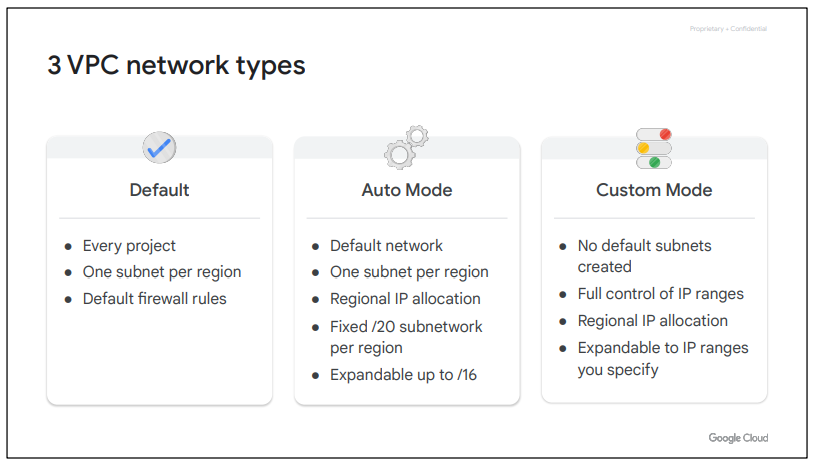
**Foundation**

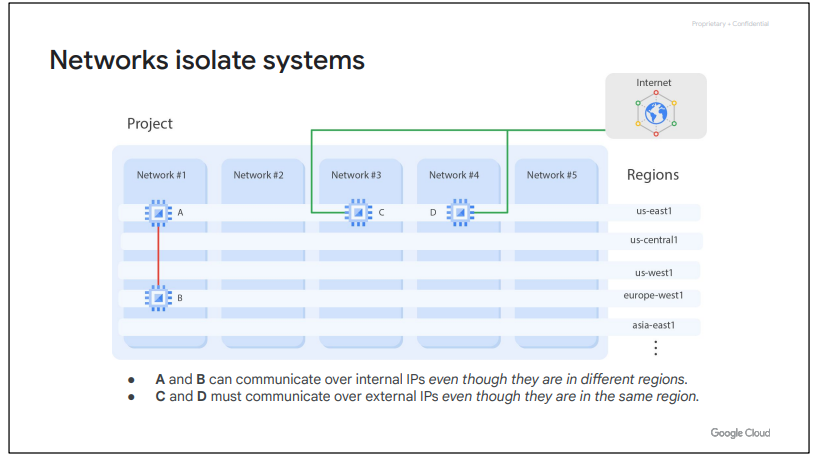
**Types of VPC Network**



Every project is provided with a default VPC network with preset subnets and firewall rules. Specifically, a subnet is allocated for each region with non-overlapping CIDR blocks and firewall rules that allow ingress traffic for ICMP, RDP, and SSH traffic from anywhere, as well as ingress traffic from within the default network for all protocols and ports.

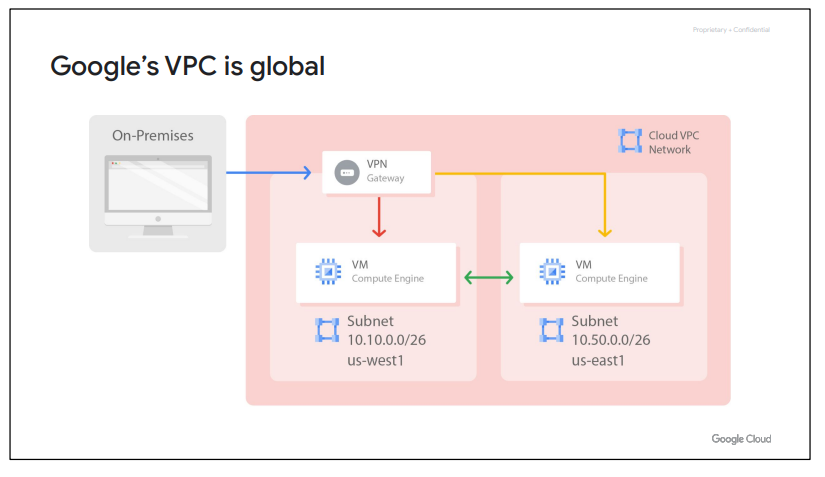
In **an auto mode** network, one subnet from each region is automatically created within it. The default network is actually an auto mode network. These automatically created subnets use a set of predefined IP ranges with a /20 mask that can be expanded to /16. All of these subnets fit within the 10.128.0.0/9 CIDR block. Therefore, as new Google Cloud regions become available, new subnets in those regions are automatically added to auto mode networks using an IP range from that block.

**A custom mode** network does not automatically create subnets. This type of network provides you with complete control over its subnets and IP ranges. You decide which subnets to create, in regions you choose, and using IP ranges you specify. These IP ranges cannot overlap between subnets of the same network.

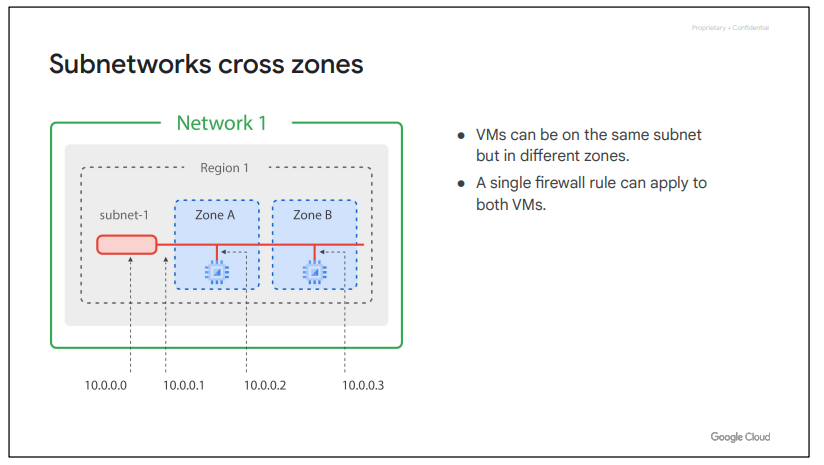


Each network contains separate virtual machines: A, B, C, and D. Because VMs A and B are in the same network, network 1, they can communicate using their internal IP addresses, even though they are in different regions. Essentially, your virtual machines, even if they exist in different locations across the world, take advantage of Google's global fiber network. Those virtual machines appear as though they're sitting in the same rack when it comes to a network configuration protocol.

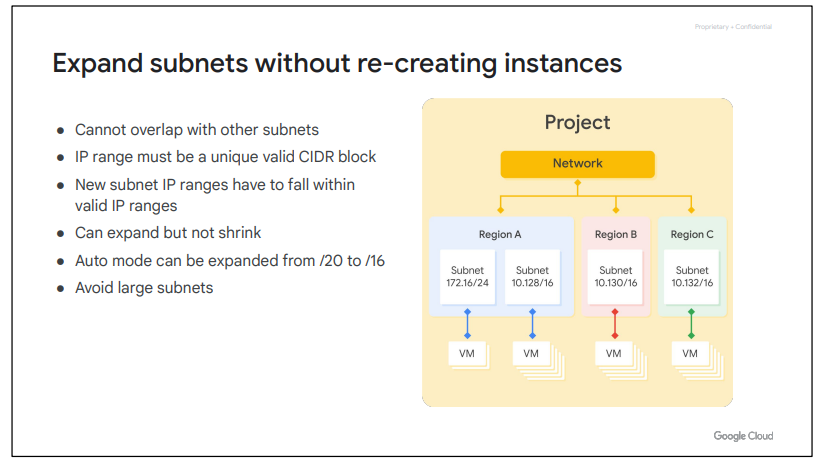
VMs C and D, however, are not in the same network. Therefore, by default, these VMs must communicate using their external IP addresses, even though they are in the same region.



Because VM instances within a VPC network can communicate privately on a global scale, a single VPN can securely connect your on-premises network to your Google Cloud network, as shown in this diagram.



Subnetworks work on a regional scale. Because a region contains several zones, subnetworks can cross zones. Now, even though the two virtual machines in this example are in different zones, they still communicate with each other using the same subnet IP address. This means that a single firewall rule can be applied to both VMs, even though they are in different zones.



Speaking of IP addresses of a subnet, Google Cloud VPCs let you increase the IP address space of any subnets without any workload shutdown or downtime.

This diagram illustrates a network with subnets that have different subnet masks, allowing for more instances in some subnets than others. This gives you flexibility and growth options to meet your needs, but there are some things to remember:

● The new subnet must not overlap with other subnets in the same VPC network in any region.

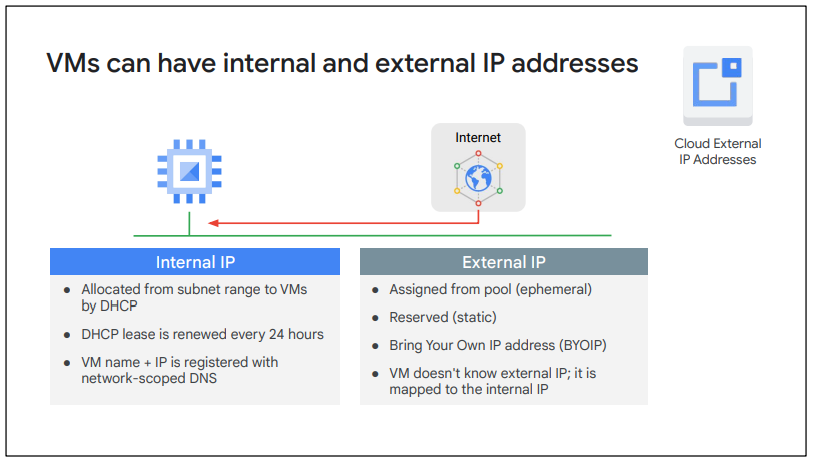
● Each IP range for all subnets in a VPC network must be a unique valid CIDR block.

● Also, the new subnet IP address ranges are regional internal IP addresses and have to fall within valid IP ranges. ○ Subnet ranges cannot match, be narrower, or be broader than a restricted range. ○ Subnet ranges cannot span a valid RFC range and a privately used public IP address range. ○ Subnet ranges cannot span multiple RFC ranges.

● The new network range must be larger than the original, which means the prefix length value must be a smaller number. In other words, you cannot undo an expansion.

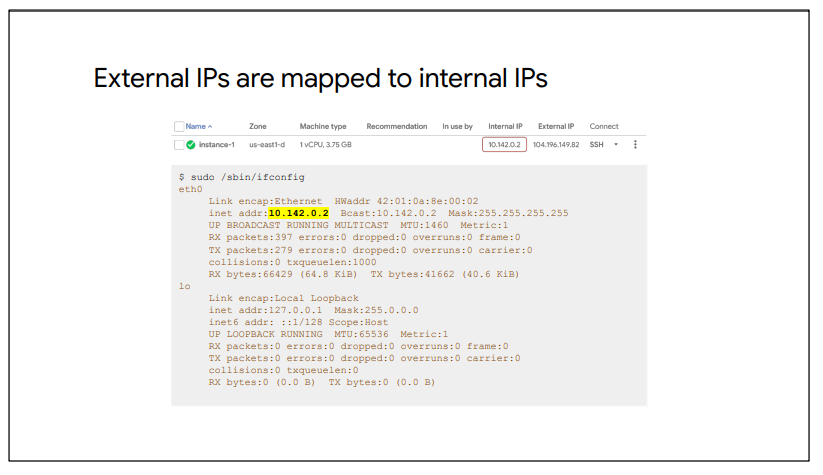
● Now, auto mode subnets start with a /20 IP range. They can be expanded to a /16 IP range, but no larger. Alternatively, you can convert the auto mode subnetwork to a custom mode subnetwork to increase the IP range further.

● Also, avoid creating large subnets. Overly large subnets are more likely to cause CIDR range collisions when using Multiple Network Interfaces and VPC Network Peering, or when configuring a VPN or other connections to an on-premises network. Therefore, do not scale your subnet beyond what you actually need.

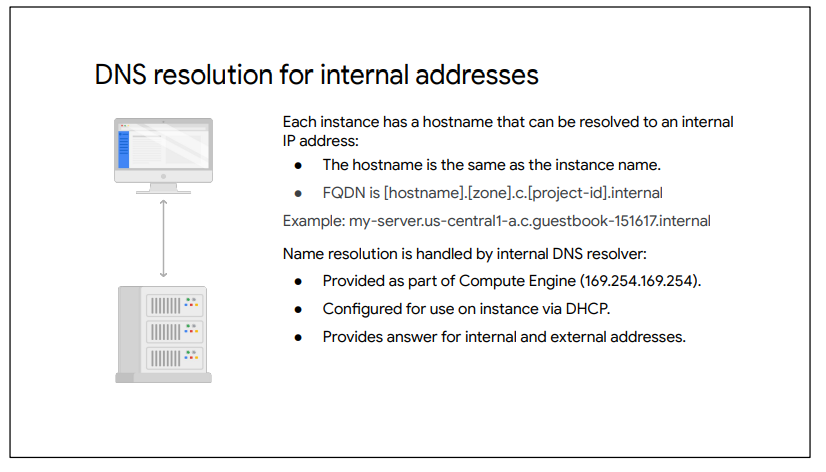


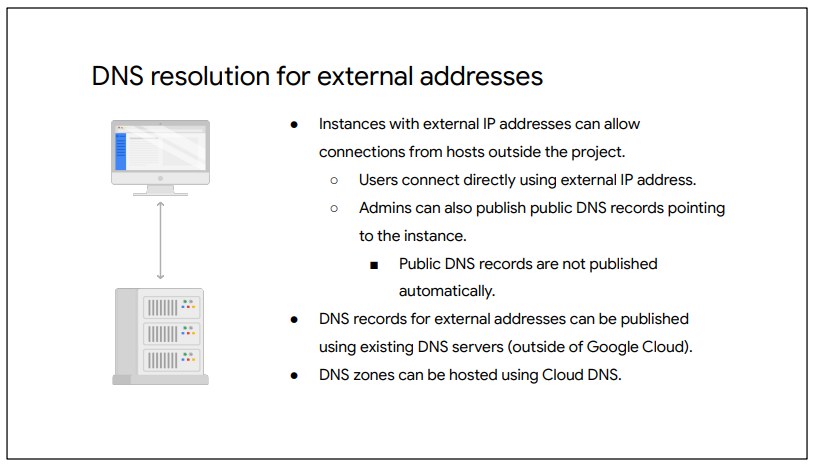
When you create a VM in Google Cloud, its symbolic name is registered with an internal DNS service that translates the name to the internal IP address. DNS is scoped to the network, so it can translate web URLs and VM names of hosts in the same network, but it can't translate host names from VMs in a different network.

The other IP address is the external IP address but this one is optional. You can assign an external IP address, if your device or your machine is externally facing. That external IP address can be assigned from a pool, making it ephemeral, or it can be assigned a reserved external IP address, making it static

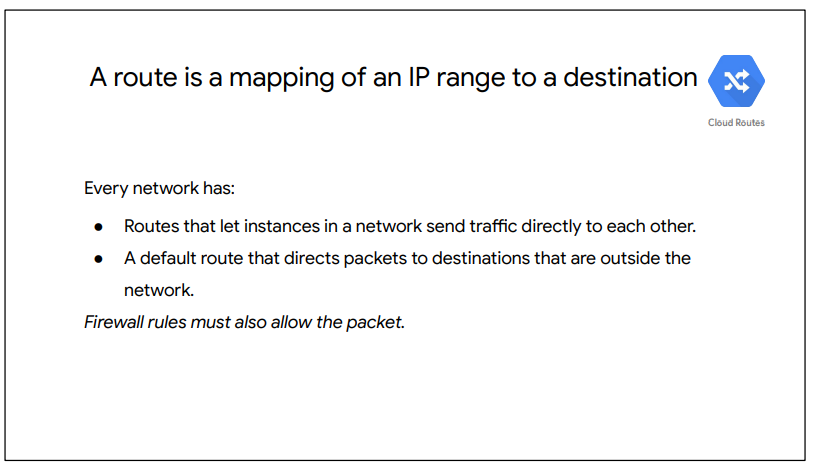


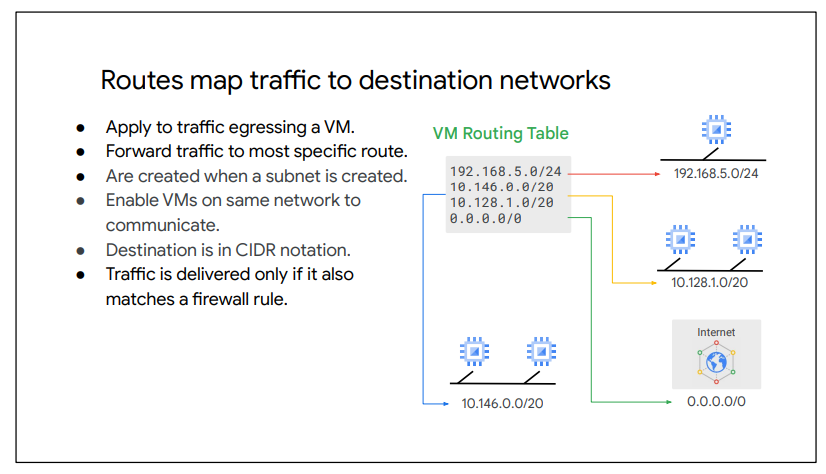
Regardless of whether you use an ephemeral or static IP address, the external address is unknown to the OS of the VM. The external IP address is mapped to the VM's internal address transparently by VPC. I am illustrating this here by running ifconfig within a VM in Google Cloud, which only returns the internal IP address.

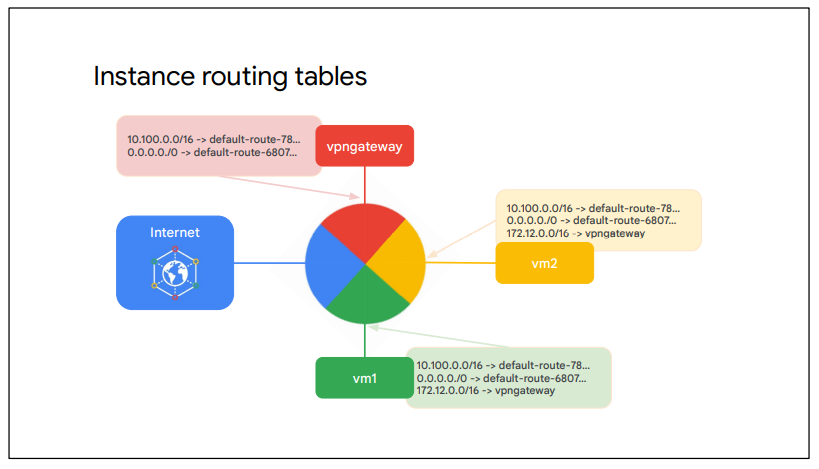




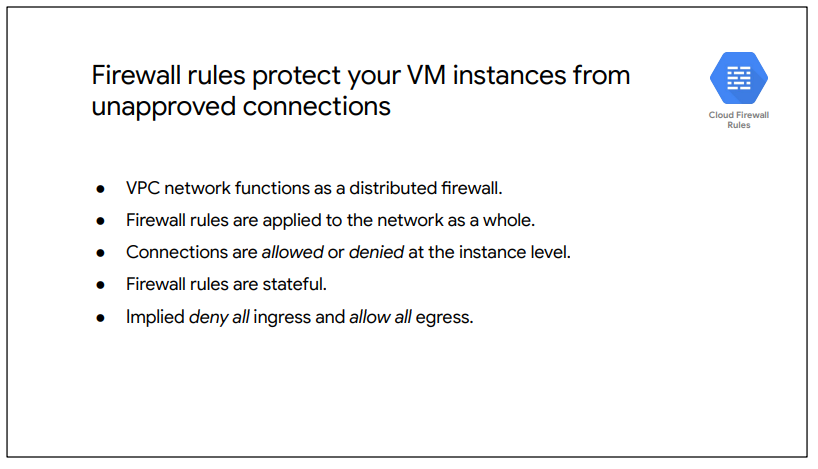
**Routes**





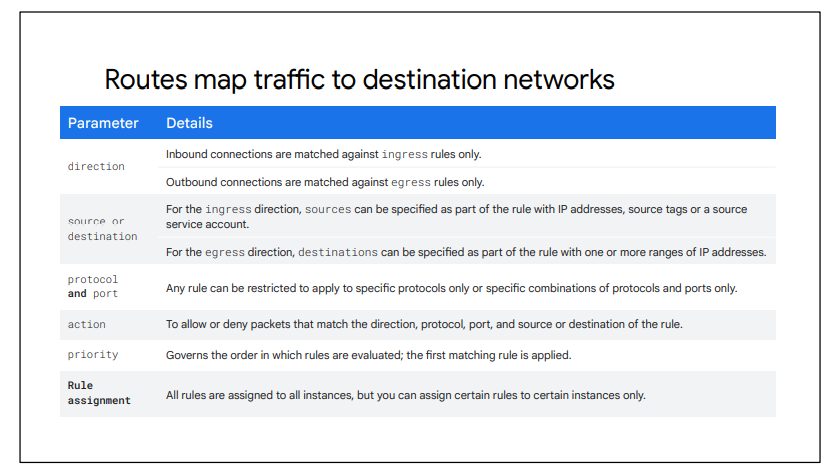


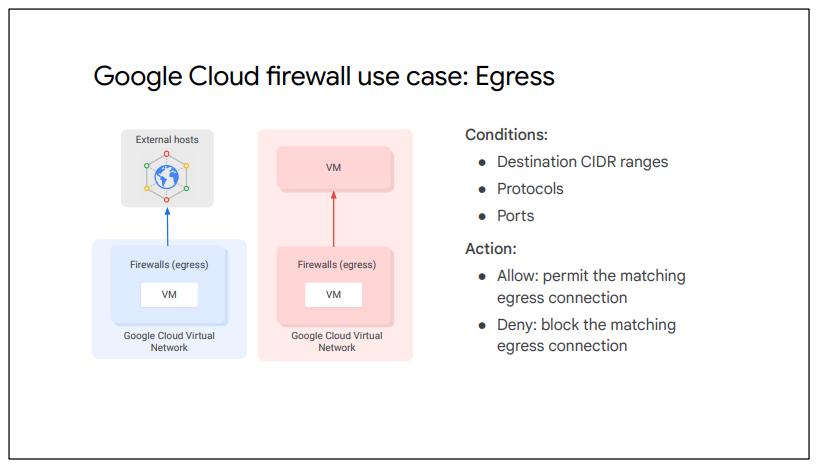
Each route in the Routes collection may apply to one or more instances. A route applies to an instance if the network and instance tags match. If the network matches and there are no instance tags specified, the route applies to all instances in that network. Compute Engine then uses the Routes collection to create individual read-only routing tables for each instance.

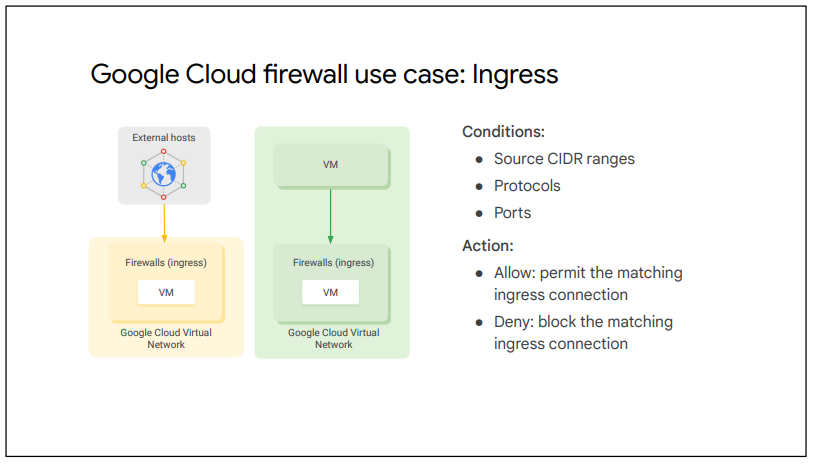


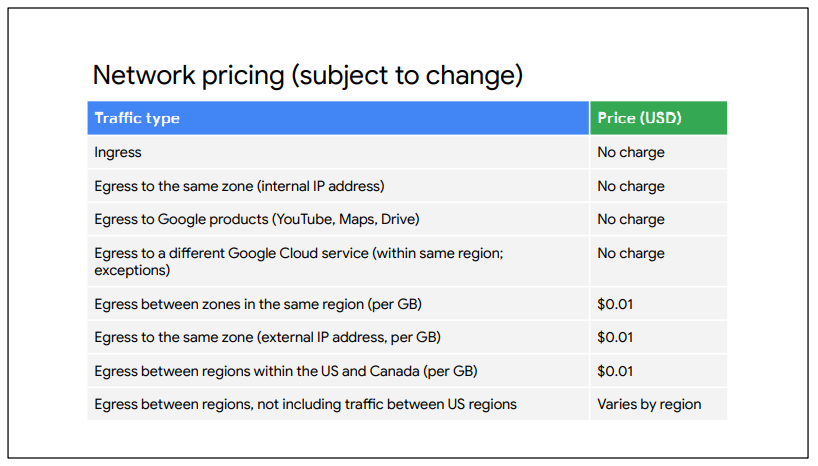
Google Cloud firewall rules are stateful. This means that if a connection is allowed between a source and a target or a target and a destination, all subsequent traffic in either direction will be allowed. In other words, firewall rules allow bidirectional communication once a session is established.

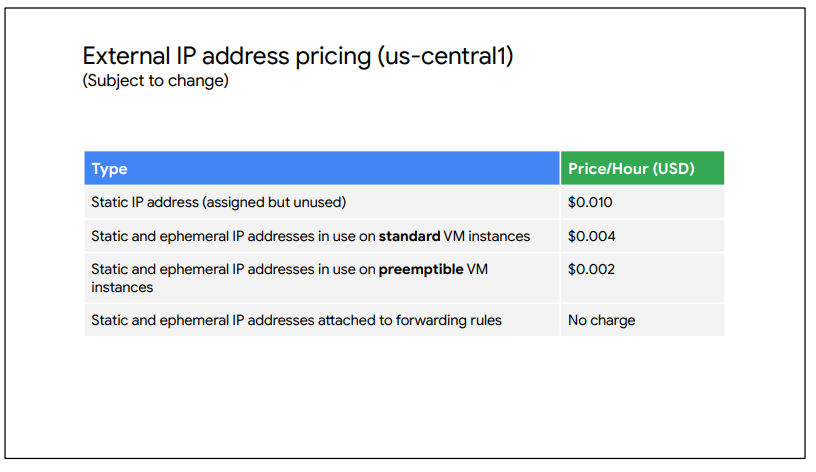
Also, if for some reason, all firewall rules in a network are deleted, there is still an implied "Deny all" ingress rule and an implied "Allow all" egress rule for the network.

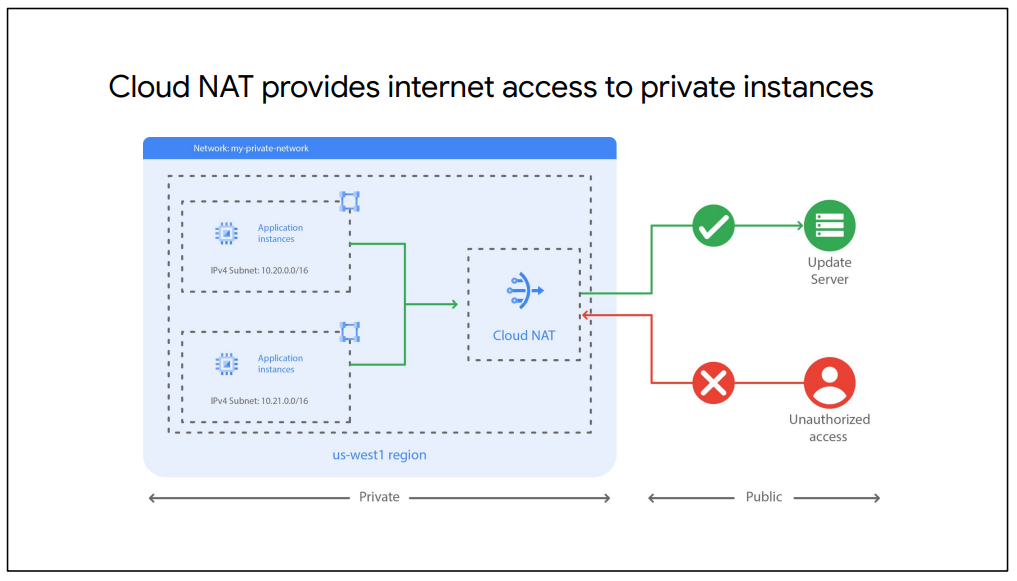












**NAT**

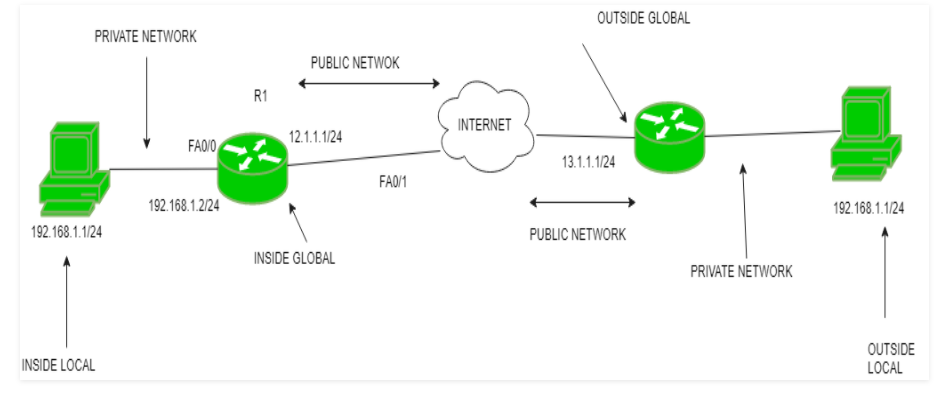
**Network Address Translation (NAT)** is a process in which one or more local IP address is translated into one or more Global IP address and vice versa in order to provide Internet access to the local hosts. Also, it does the translation of port numbers i.e. masks the port number of the host with another port number, in the packet that will be routed to the destination. It then makes the corresponding entries of IP address and port number in the NAT table. NAT generally operates on a router or firewall.

**Network Address Translation (NAT) working –**   
Generally, the border router is configured for NAT i.e the router which has one interface in the local (inside) network and one interface in the global (outside) network. When a packet traverse outside the local (inside) network, then NAT converts that local (private) IP address to a global (public) IP address. When a packet enters the local network, the global (public) IP address is converted to a local (private) IP address.

If NAT runs out of addresses, i.e., no address is left in the pool configured then the packets will be dropped and an Internet Control Message Protocol (ICMP) host unreachable packet to the destination is sent.

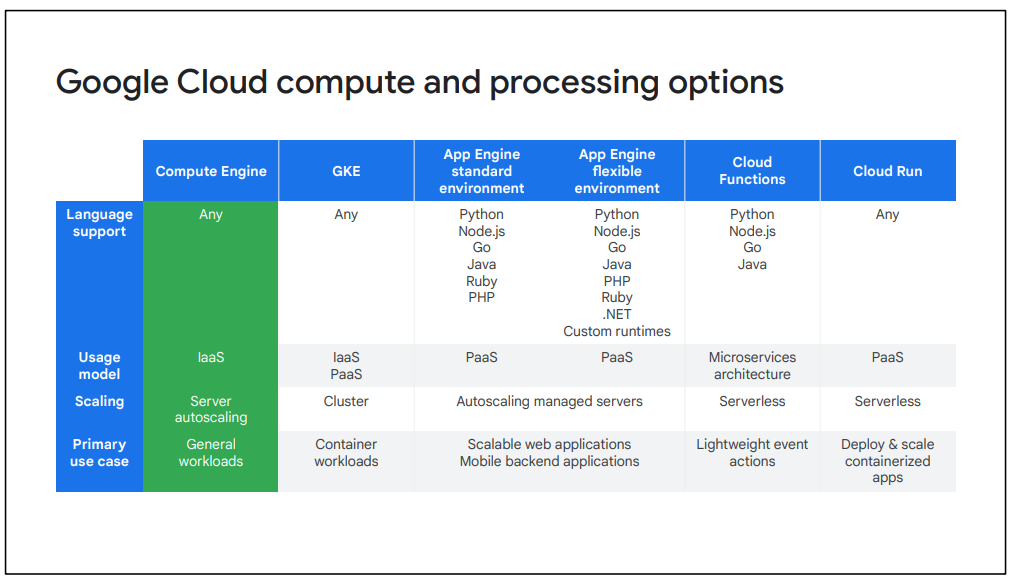
**Why mask port numbers ?**  
Suppose, in a network, two hosts A and B are connected. Now, both of them request for the same destination, on the same port number, say 1000, on the host side, at the same time. If NAT does only translation of IP addresses, then when their packets will arrive at the NAT, both of their IP addresses would be masked by the public IP address of the network and sent to the destination. Destination will send replies to the public IP address of the router. Thus, on receiving a reply, it will be unclear to NAT as to which reply belongs to which host (because source port numbers for both A and B are the same). Hence, to avoid such a problem, NAT masks the source port number as well and makes an entry in the NAT table.

* **Inside local address –** An IP address that is assigned to a host on the Inside (local) network. The address is probably not an IP address assigned by the service provider i.e., these are private IP addresses. This is the inside host seen from the inside network.
* **Inside global address –** IP address that represents one or more inside local IP addresses to the outside world. This is the inside host as seen from the outside network.
* **Outside local address –** This is the actual IP address of the destination host in the local network after translation.
* **Outside global address –** This is the outside host as seen from the outside network. It is the IP address of the outside destination host before translation.

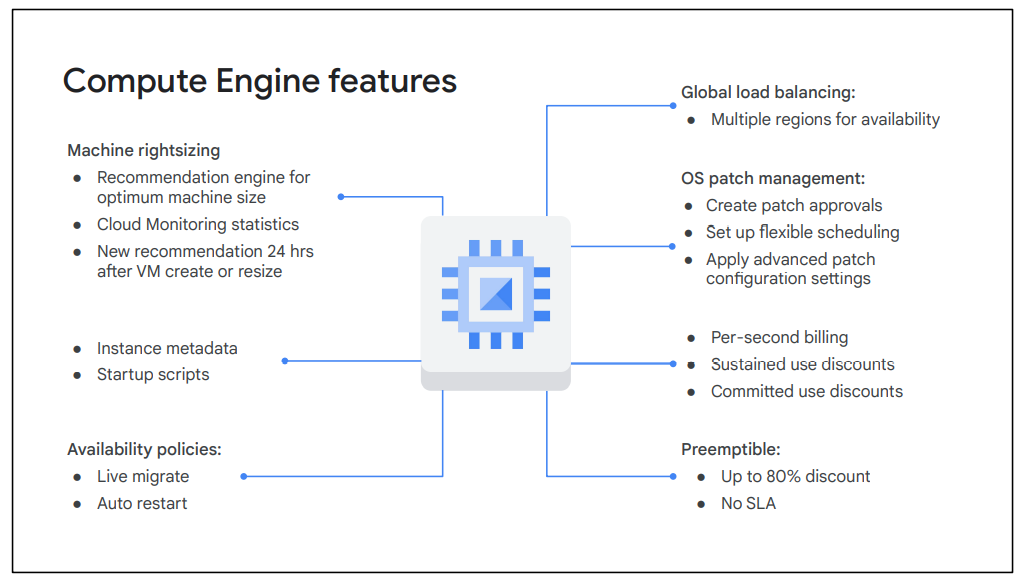


**2.**

VMs consists of a virtual CPU, some amount of memory, disk storage, and an IP address. Compute Engine is GCP’s service to create VMs; it is very flexible and offers many options, including some that can't exist in physical hardware.



* Infrastrucuture as a Service (Iaas)
* Predefined or custom machine types
  + vCPU(s) and Memory (RAM)
  + Storage
    - Zonal or Regional Persistent Disk (HDD or SSD)
    - Local SSD
    - Cloud Storage
  + Networking
  + Linux or Windows



Basically, SSDs are designed to give you a higher number of IOPS per dollar versus standard disks, which will give you a higher amount of capacity for your dollar.

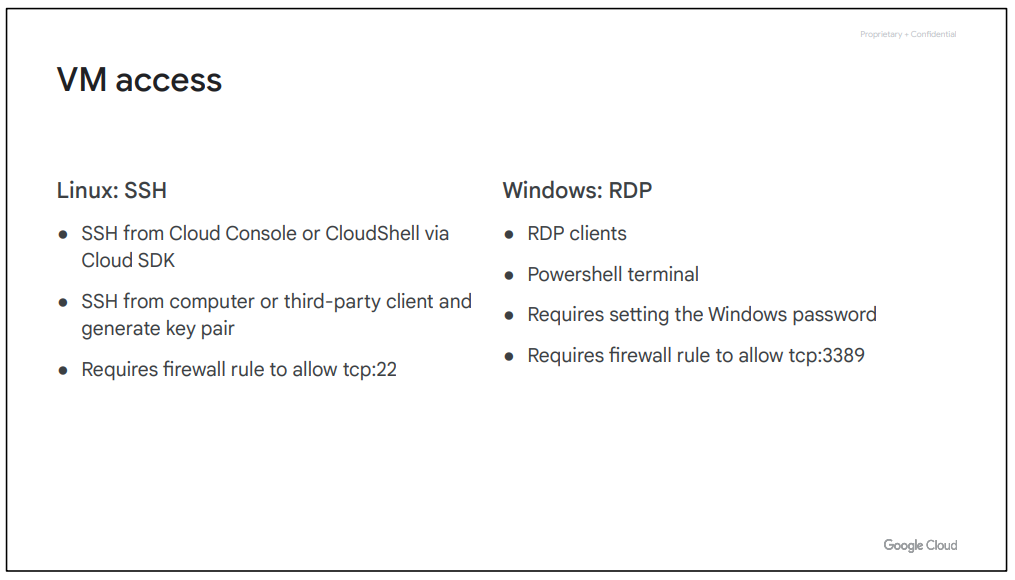
Local SSDs have even higher throughput and lower latency than SSD persistent disks, because they are attached to the physical hardware. However, the data that you store on local SSDs persists only until you stop or delete the instance. Typically, a local SSD is used as a swap disk.

**Networking**

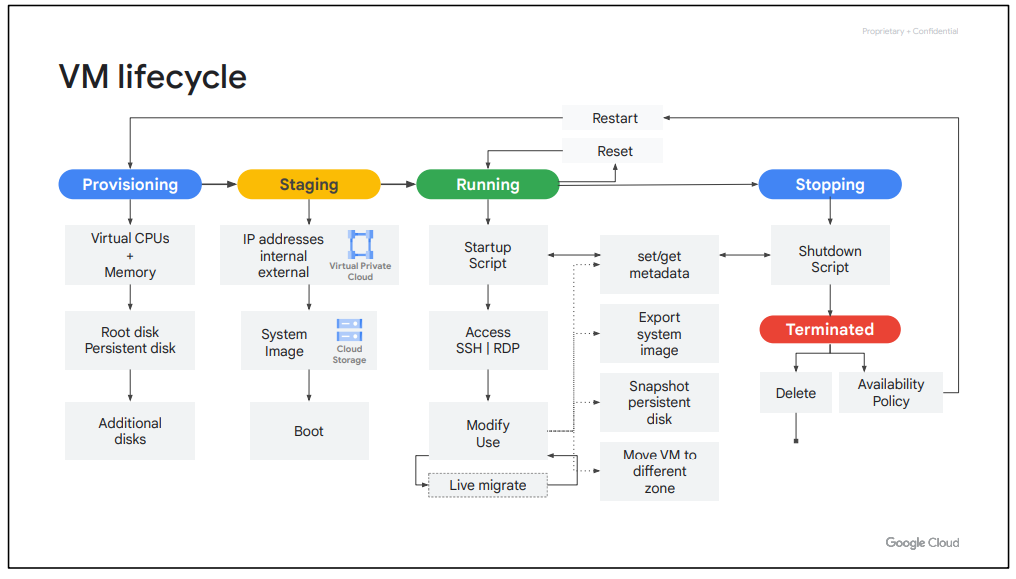
Robust networking features:

* Default, custom networks
* Inbound/Outbound firewall rules
  + IP based
  + Instance/group based
* Regional HTTPS load balancing
* Network load balancing
  + Doesn’t require pre-warming
* Global and multi regional subnetworks

You’ll also notice that you can do regional HTTPS load balancing and network load balancing. This doesn’t require any pre-warming because a load balancer isn't a hardware device that needs to analyze your traffic. A load balancer is essentially a set of traffic engineering rules that are coming into the Google network, and VPC is applying your rules destined for your IP address subnet range.



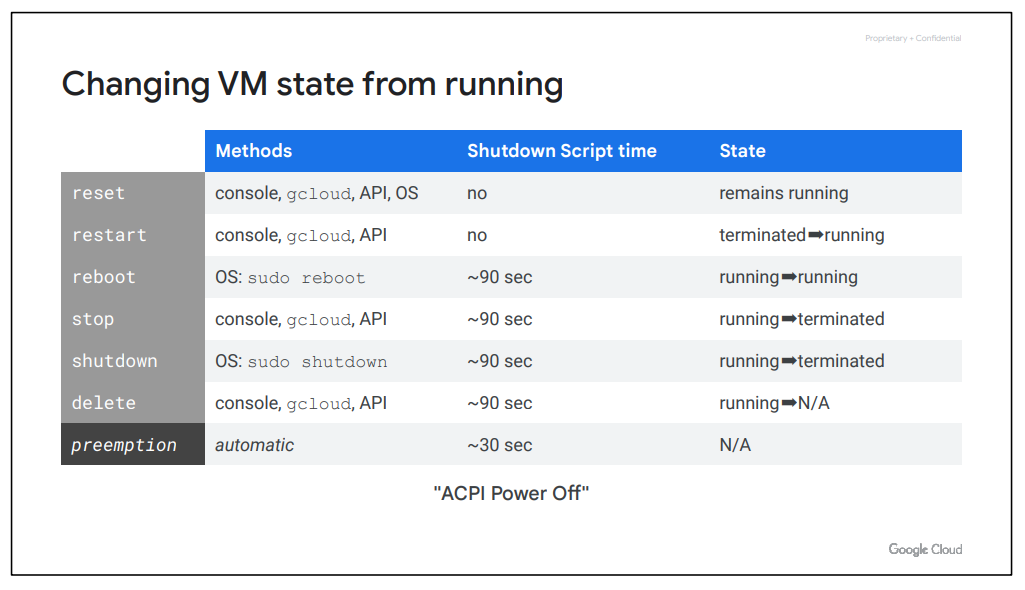
**VM Lifecycle**

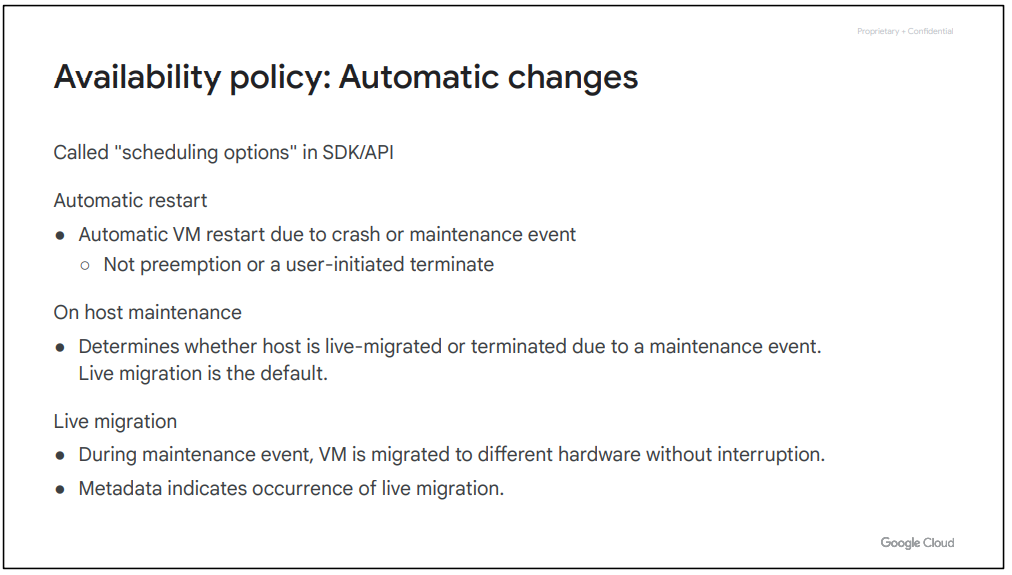


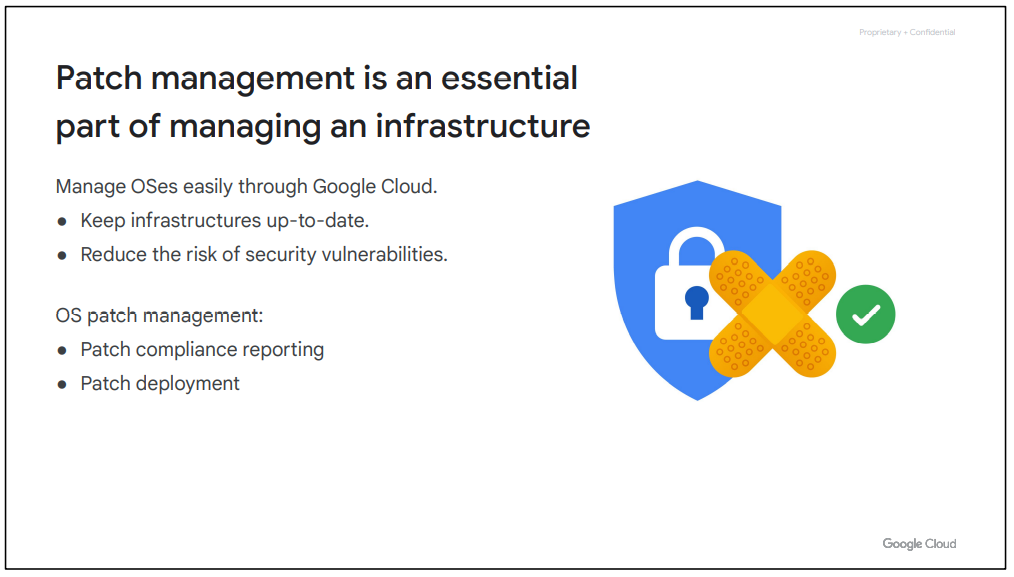
When you define all the properties of an instance and click Create, the instance enters the provisioning state. Here the resources such as CPU, memory, and disks are being reserved for the instance, but the instance itself isn’t running yet. Next, the instance moves to the staging state where resources have been acquired and the instance is prepared for launch. Specifically, in this state, Compute Engine is adding IP addresses, booting up the system image, and booting up the system.

After the instance starts running, it will go through pre-configured startup scripts and enable SSH or RDP access. Now, you can do several things while your instance is running. For example, you can live migrate your virtual machine to another host in the same zone instead of requiring your instance to be rebooted. This allows Google Cloud to perform maintenance that is integral to keeping the infrastructure protected and reliable, without interrupting any of your VMs. While your instance is running, you can also move your VM to a different zone, take a snapshot of the VM’s persistent disk, export the system image, or reconfigure metadata. We will explore some of these tasks in later labs.

Some actions require you to stop your virtual machine; for example, if you want to upgrade your machine by adding more CPU. When the instance enters this state, it will go through pre-configured shutdown scripts and end in the terminated state. From this state, you can choose to either restart the instance, which would bring it back to its provisioning state, or delete it







When you provision a premium image, there is a cost associated with the image. This cost includes both the usage of the OS but also the patch management of the OS. Using Google Cloud, we can easily manage the patching of your OSes.

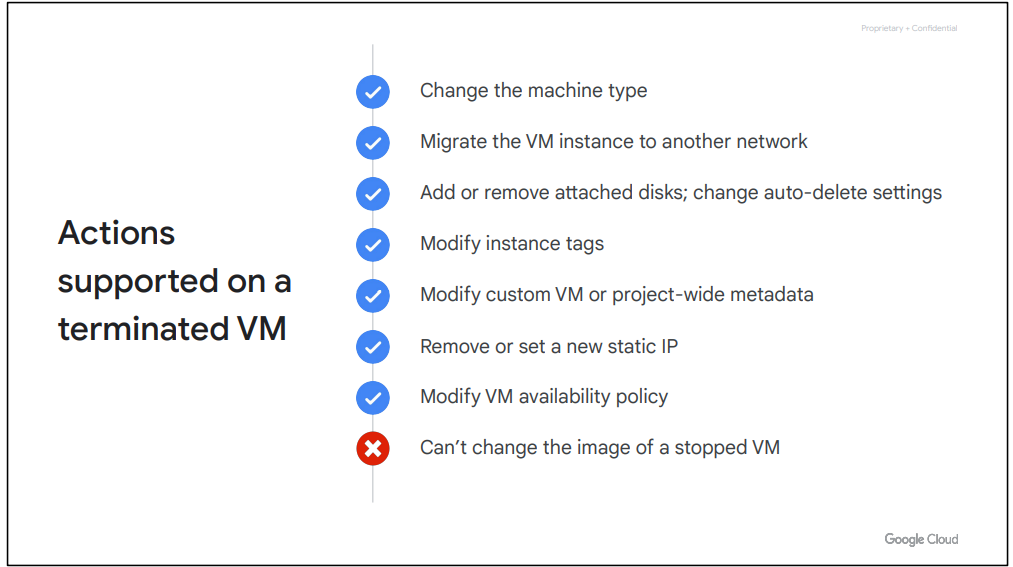
Managing patches effectively is a great way to keep your infrastructure up-to-date and reduce the risk of security vulnerabilities. But without the right tools, patching can be daunting and labor intensive.

Use OS patch management to apply operating system patches across a set of Compute Engine VM instances. Long-running VMs require periodic system updates to protect against defects and vulnerabilities.

Several tasks that need to be performed while patch management:

* *Create patch approvals* You can select what patches to apply to your system from the full set of updates available for the specific operating system.
* *Set up flexible scheduling* You can choose when to run patch updates (one-time and recurring schedules)
* *Apply advanced patch configuration settings*  You can customize your patches by adding configurations such as pre and post patching scripts.

\*\* When a VM is terminated, you do not pay for memory and CPU resources. However, you are charged for any attached disks and reserved IP addresses.



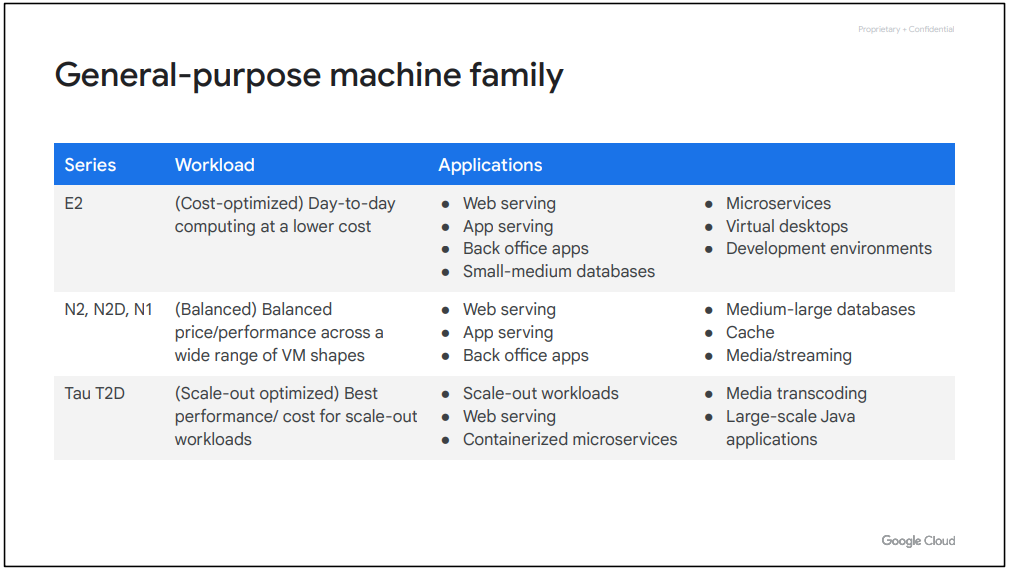
There are four Compute Engine machine families.

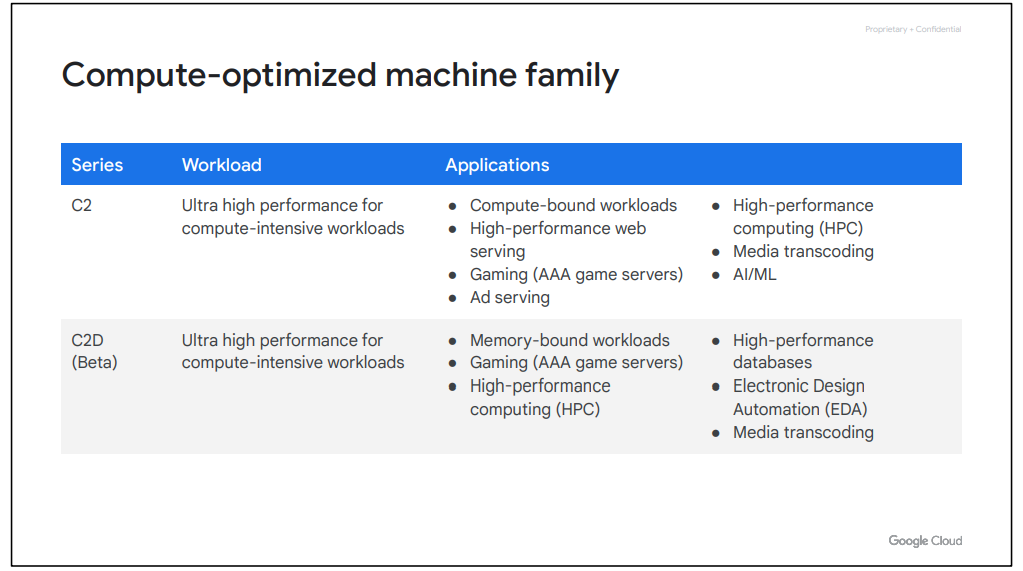
● General-purpose

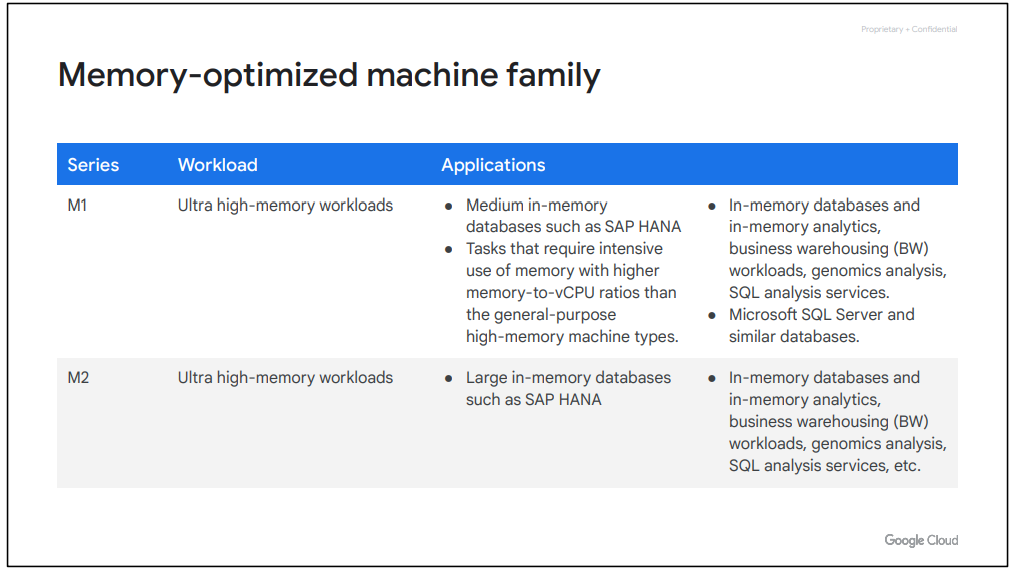
● Compute-optimized

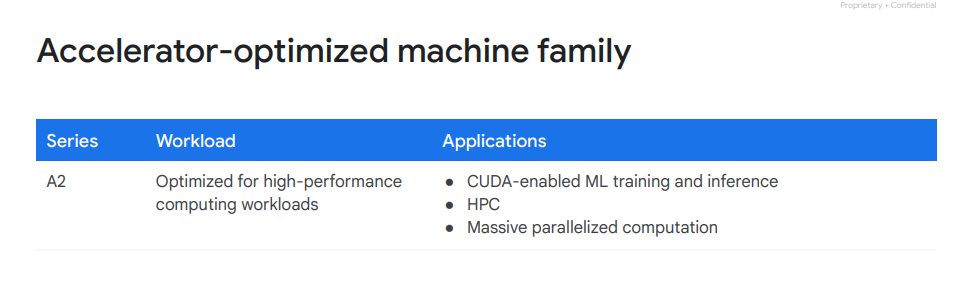
● Memory-optimized

● Accelerator-optimized









**Pricing**

* Per Second billing, with minimum of 1 minute
  + vCPUs, GPUs, and GBs of memory
* Resource based pricing
  + Each vCPU and each GB of memory is billed separately.
* Discounts:
  + Sustained use
  + Committed use
  + Preemptible VM instances
* Recommendation engine
  + Notifies you of underutilized resources
* Free Usage limits

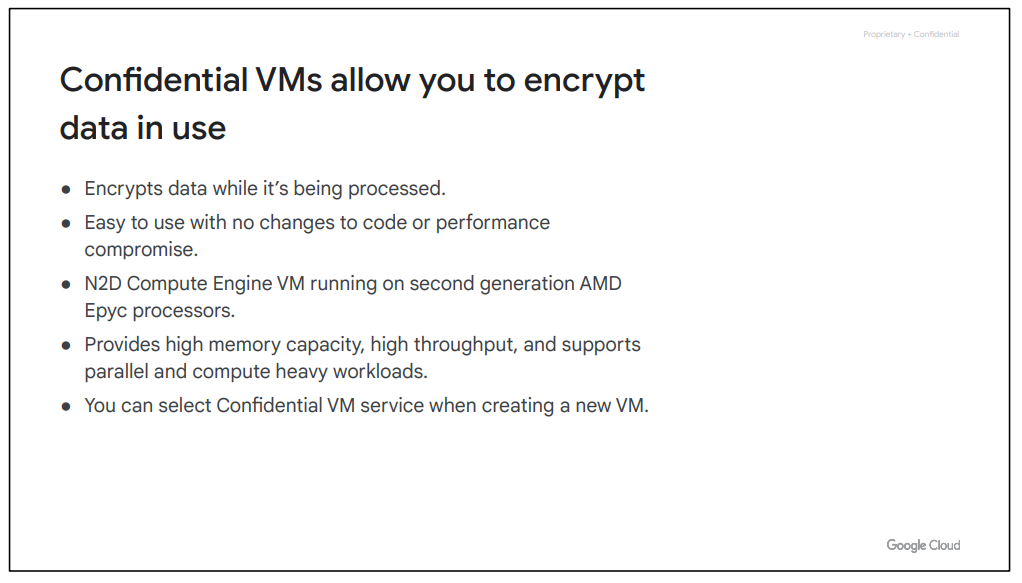
Compute Engine calculates sustained use discounts based on vCPU and memory usage across each region and separately for each of the following categories:

● Predefined machine types

● Custom machine type

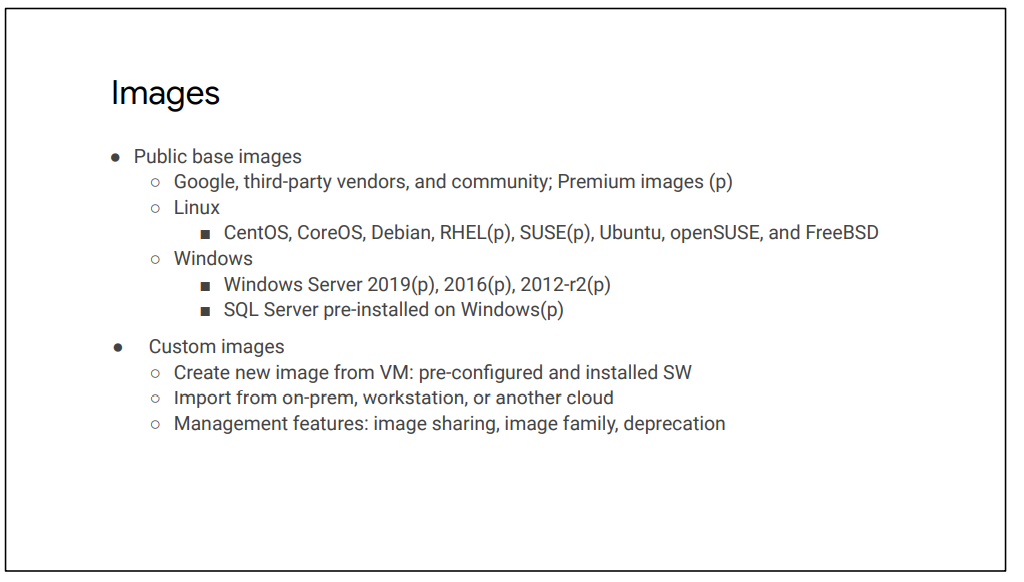
**Preemptible**

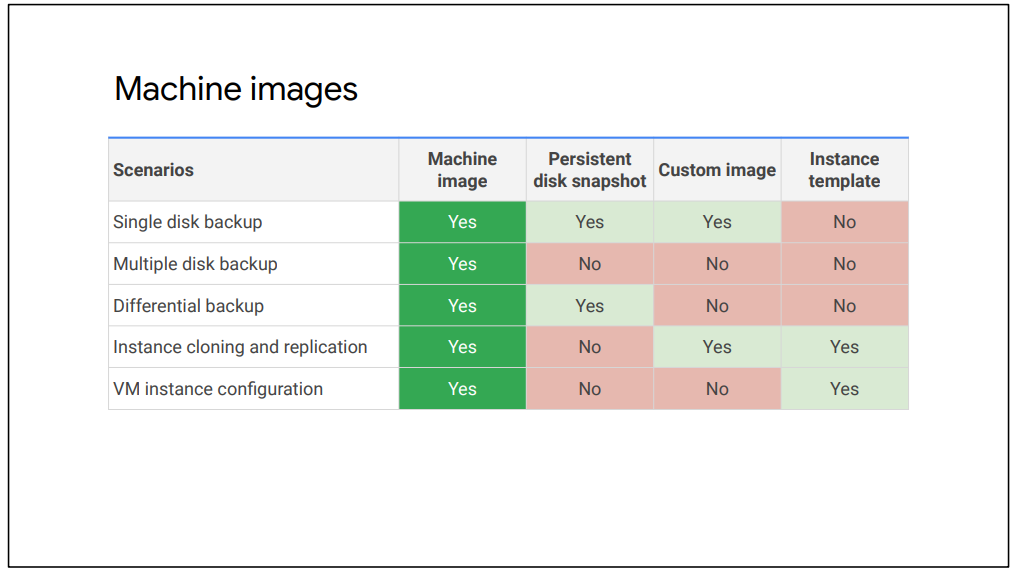
* Lower price for interruptible service (upto 80%)
* VM might be terminated at any time
  + No charge if terminated in the first minute
  + 24 hrs max
  + 30-sec terminate warning but not guaranteed
    - Time for a shutdown script
* No live migrate, no auto restart
* You can request the CPU quota for a region be split between regular and preemption
  + Default : preemptible VMs count against region CPU quota

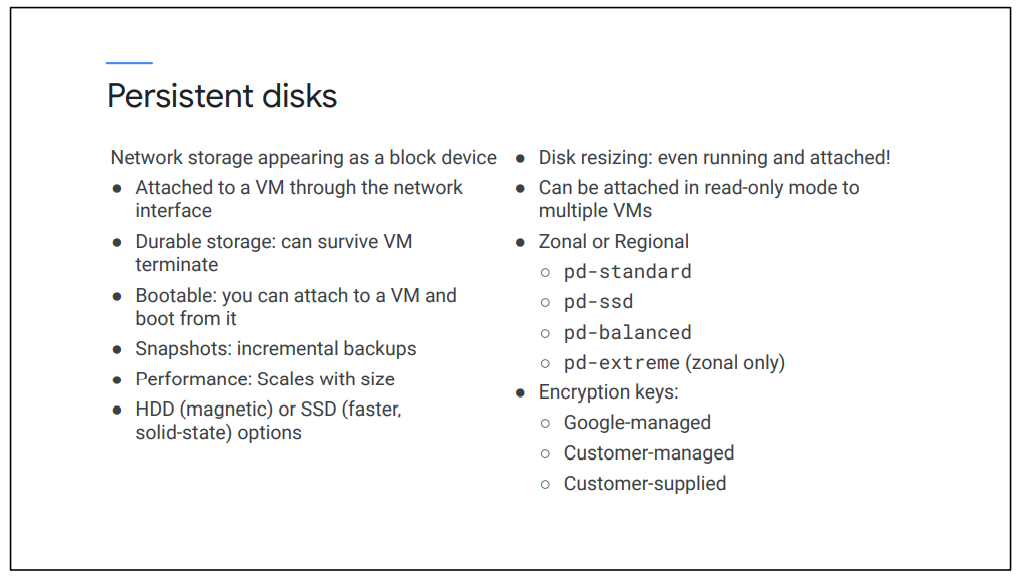


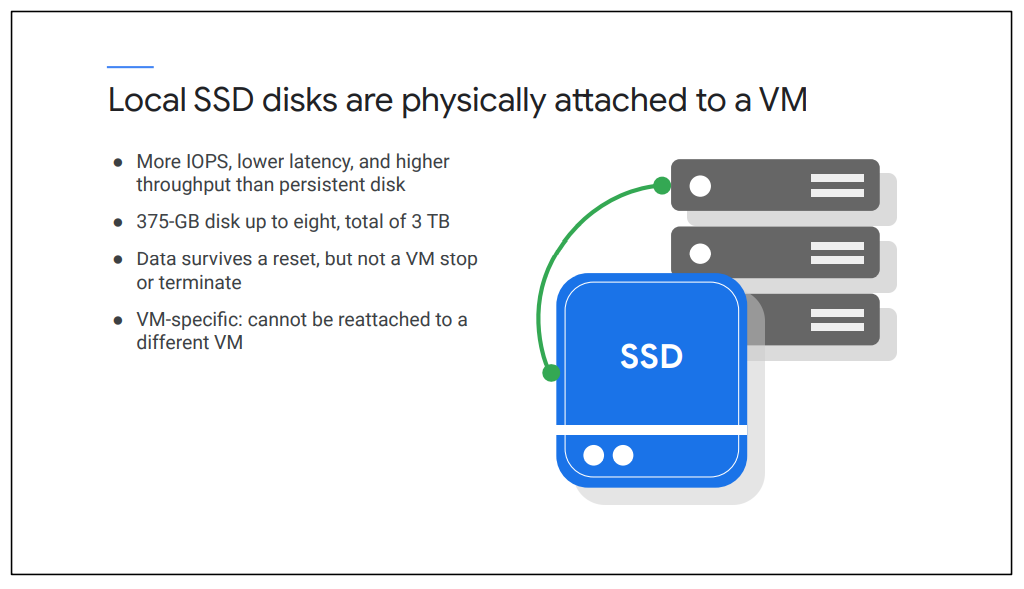
**Image :**

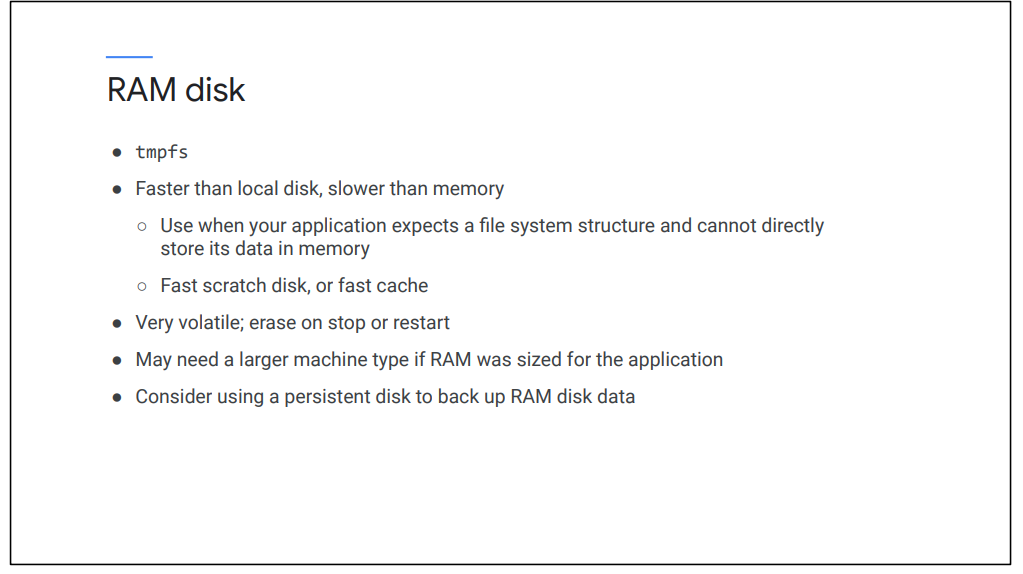
When creating a virtual machine, you can choose the boot disk image. This image includes the boot loader, the operating system, the file system structure, any pre-configured software, and any other customizations.

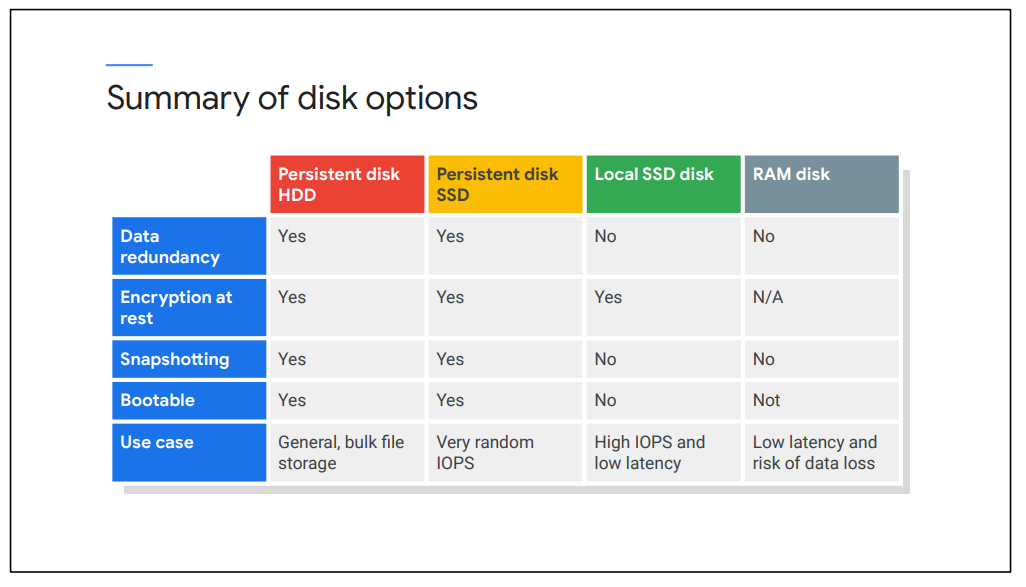


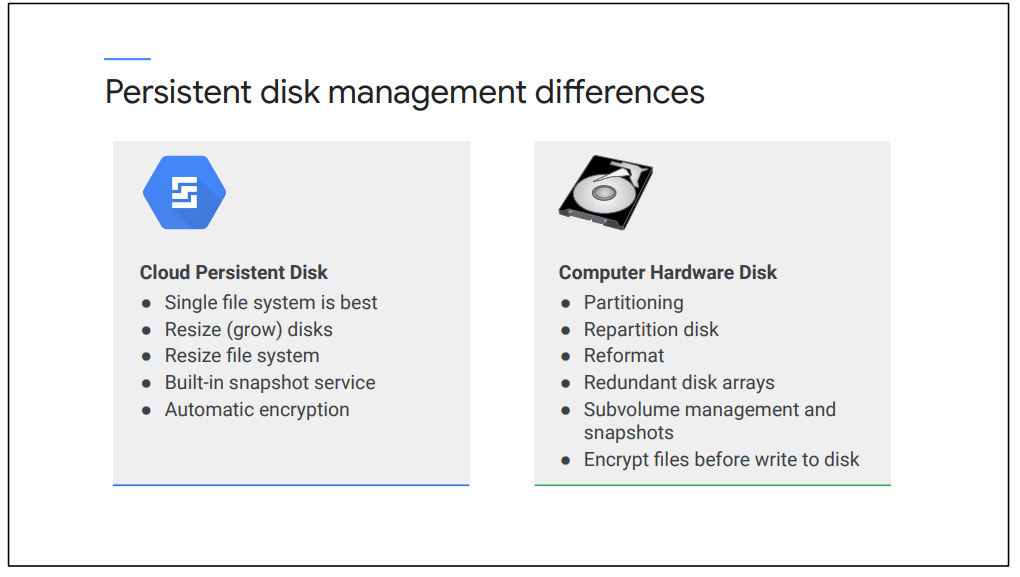












**Common Actions :**

**Metadata and scripts**

The metadata server is particularly useful in combination with startup and shutdown scripts, because you can use the metadata server to programmatically get unique information about an instance, without additional authorization. For example, you can write a startup script that gets the metadata key/value pair for an instance's external IP address and use that IP address in your script to set up a database. Because the default metadata keys are the same on every instance, you can reuse your script without having to update it for each instance.

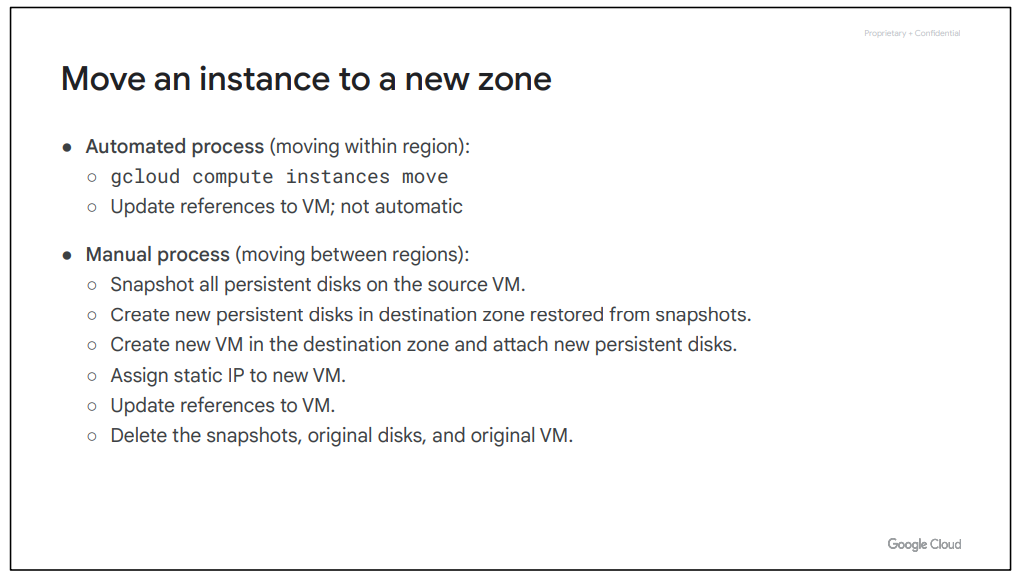
**Moving an instance to a new zone**

Another common action is to move an instance to a new zone. For example, you might do so for geographical reasons or because a zone is being deprecated.

You can move a VM even if one of the following scenarios applies:

● The VM instance is in a TERMINATED state.

● The VM instance is a Shielded VM that uses UEFI firmware



**Snapshot : Backup Critical Data**

Snapshots have many use cases. For example, they can be used to backup critical data into a durable storage solution to meet application, availability, and recovery requirements. These snapshots are stored in Cloud Storage, which is covered later.

**Snapshot : Migrate data between zones**

Snapshots can also be used to migrate data between zones. We just discussed this when going over the manual process of moving an instance between two regions, but this can also be used to simply transfer data from one zone to another. For example, you might want to minimize latency by migrating data to a drive that can be locally attached in the zone where it is used.