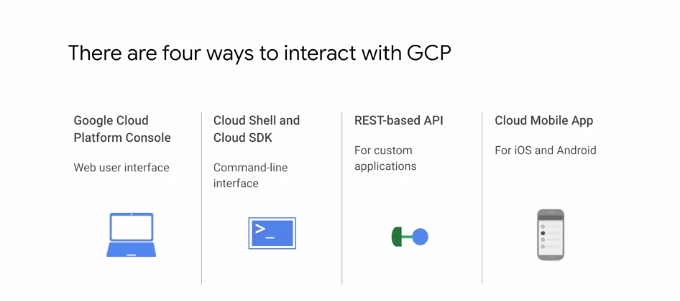
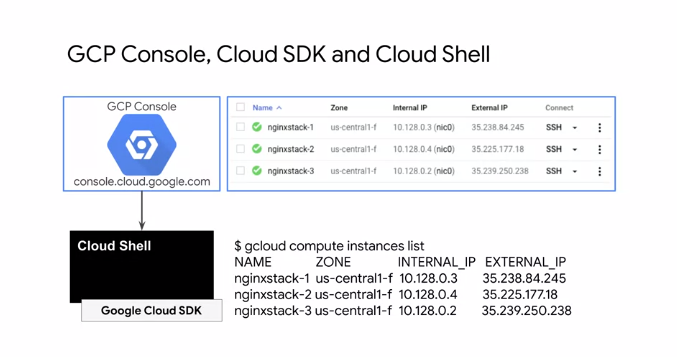
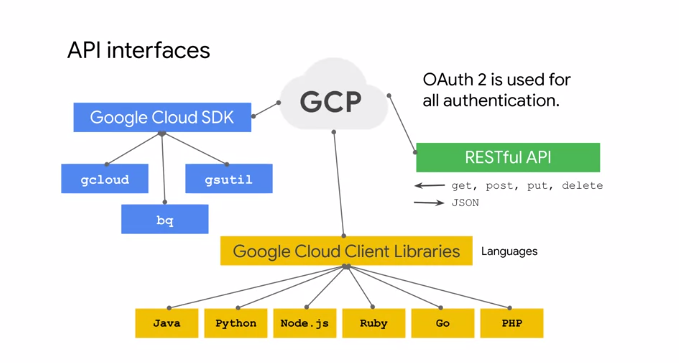
**Using GCP**

There are four ways you

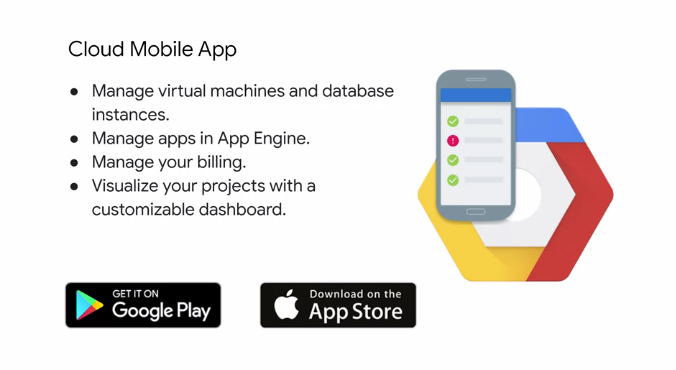
There are four ways you can interact with GCP, and we'll talk about each in turn. There's the Google Cloud Platform console or GCP console, Cloud Shell and the Cloud SDK, the APIs and the Cloud mobile app.



The GCP console provides a web-based graphical User Interface that you access through console.cloud.google.com For example, you can view your Virtual Machines and their details as shown on the top. If you prefer to work in a terminal window, the Cloud SDK provides the gcloud command line tool. For example, you can list your Virtual Machines and their details as shown on the bottom with the gcloud compute instances list command. GCP, also provides Cloud Shell which is a browser-based interactive shell environment for GCP that you can access from the GCP console. Cloud Shell is a temporary Virtual Machine with five gigabytes of persistent disk storage that has the Cloud SDK pre-installed. Throughout this course, you will apply what you learn in different labs. These labs will have instructions to use the GCP console such as on the navigation menu click ''Compute Engine > VM instances''. Let me dissect these instructions. First, within the GCP console, you will click on the icon with the three horizontal lines, which is the navigation menu as shown on the left. This opens a menu as shown on the right. All of the major products and services are listed on this menu. Then within the menu, hover over Compute Engine to open a sub-menu. Finally, click on VM instances on the sub-menu. You will get more comfortable with these instructions and the GCP console as you work on labs. Now, labs will also use command line instructions. You will enter these instructions either in Cloud Shell or an SSH terminal by simply copying and pasting them. In some cases, you will have to modify these commands, for example when choosing a globally unique name for a Cloud Storage bucket.

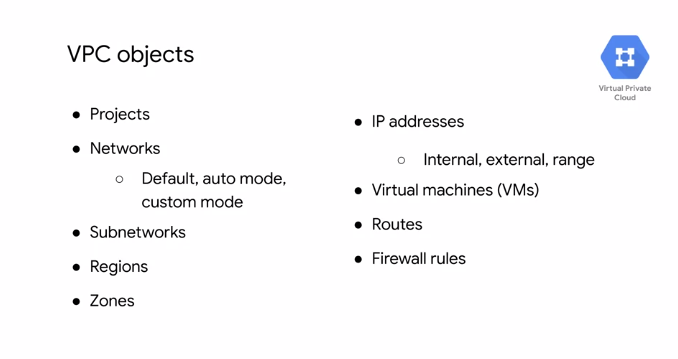


In addition to the Cloud SDK, you can also use Client Libraries that enable you to easily create and manage resources. GCP Client Libraries expose APIs for two main purposes. App APIs provide access to services, and they're optimized for supported languages such as Node. js or Python. Admin APIs offer functionality for resource management. For example, you can use Admin APIs if you want to build your own automated tools.



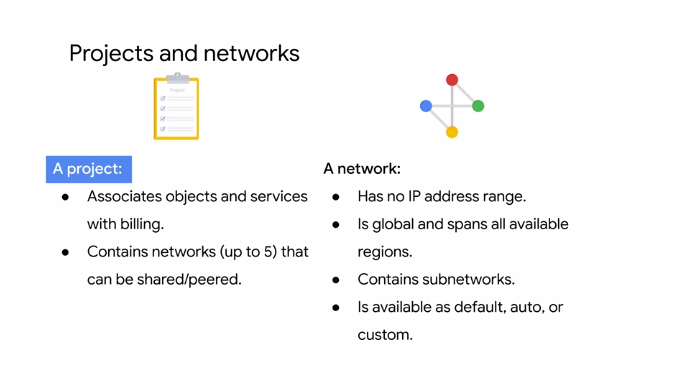
The Cloud mobile app is another way to interact with GCP. It allows you to manage GCP services from your Android or iOS device. For example, you can start, stop an SSH into Compute Engine instances, and see logs from each instance. You can also set up customizable graphs showing key metrics such as CPU usage, network usage, requests per seconds, and server errors. The app even offers alerts and incident management, and allows you to get up-to-date billing information for your projects, and get billing alerts for projects that are going over budget. You can download the Cloud mobile app from Google Play or from the App Store.

**Virtual Private Cloud**

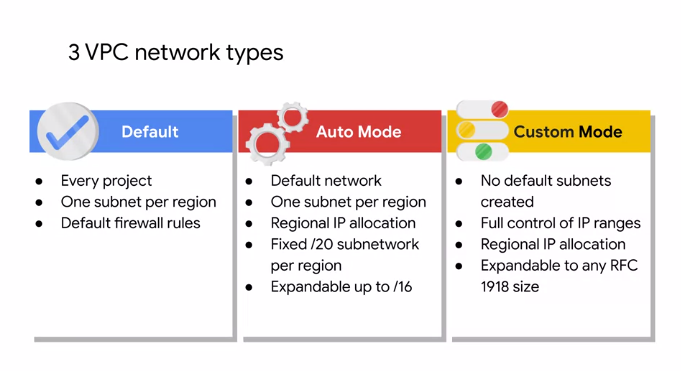
****

With GCP, you can provision your GCP resources, connect them to each other, and isolate them from each other in a Virtual Private Cloud. You can also define fine-grained network and policies within GCP and between GCP and On-premises or other public Clouds. Essentially, VPC is a comprehensive set of Google managed networking objects, which we will explore in detail throughout this module. Let me give you a high-level overview of these objects. Projects are going to encompass every single service that you use including networks. Networks come in three different flavors; default, auto mode, and custom mode. Subnetworks allow you to divide or segregate your environment. Regions in zones represents Google's datacenters and they provide continuous Data Protection and high availability. VPC provides IP addresses for internal and external use along with granular IP address range selections. As for virtual machines, in this module, we will focus on configuring VM instances from a networking perspective. We'll also go over routes and firewall routes.

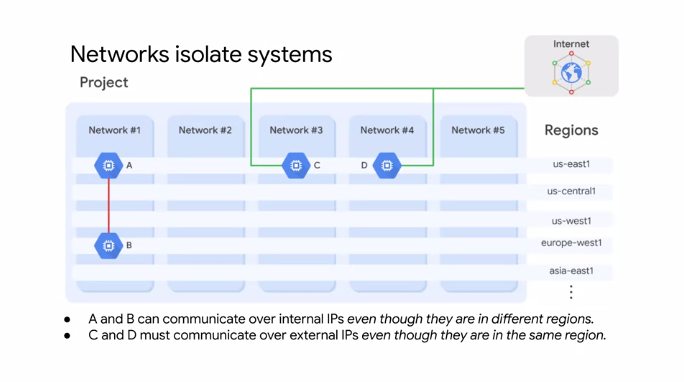
**Projects, networks, and subnetworks**

****

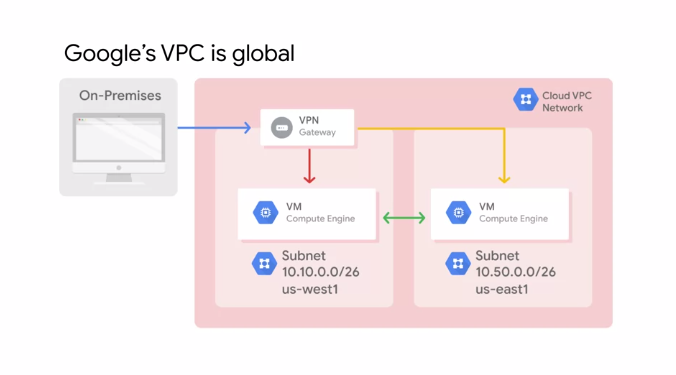
Let's start exploring the VPC objects by looking at projects, networks, and subnetworks. Projects are the key organizer of infrastructure resources in GCP. A project associates objects and services with billing. Now, it's unique that projects actually contain entire networks. The default quota for each project is five networks but you can simply request additional quota using the GCP console. These networks can be shared with other projects or they can be peered with networks in other projects. Both of which we'll cover later in the Architecting with Google Compute Engine Core series. These networks do not have IP ranges but are simply a construct of all of the individual IP addresses and services within that network. GCP networks are global spending all available regions across the world that I showed earlier. So you can have one network that later exists anywhere in the world, Asia, Europe, Americas, all simultaneously. Inside a network you can segregate your resources with regional subnetworks. I just mentioned that there are different types of networks: default, auto, and custom. Let's explore these types of networks in more detail.



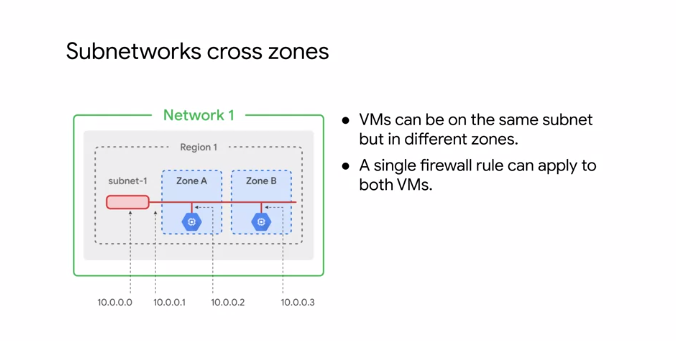
Every project is provided with a default VPC network with presets subnets and firewall rules. Specifically a subnet is allocated for each region with non-overlapping sider blocks and firewall rules that allow ingress traffic from ICMP, RDP, and SSH traffic from anywhere, as well as ingress traffic from within the default network for all protocols and ports. In an Auto Mode network, one subnets from each region is automatically created within it. The default network is actually an auto mode network. These automatically created subnets uses set of predefined IP ranges with a /20 mask that can be expanded to a /16. All of these subnets fit within the 10.128.0.0/9 cider block. Therefore, as new GCP regions become available, new subnets and dose regions are automatically added to automotive 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 within the RFC 1918 address space. These IP ranges cannot overlap between subnets of the same network. Now, you can convert an auto mode network to a custom mode network to take advantage of the control that custom mode networks provide. However, this conversion is one way. Meaning that custom mode networks cannot be changed to auto mode nodes. So carefully review the consideration for auto mode networks to help you decide which type of network meets your needs.



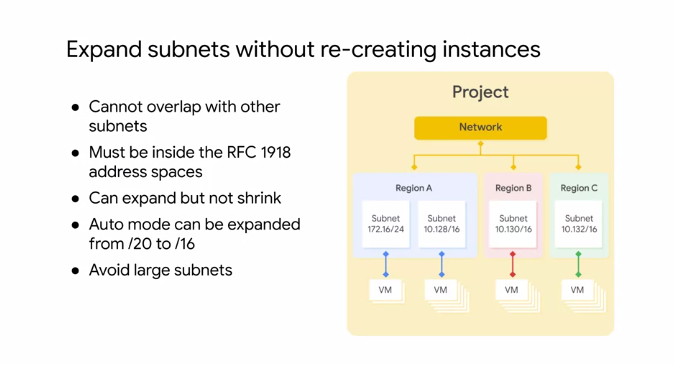
On this slide, we have an example of a project that contains five networks. All of these networks span multiple regions across the world as you can see on the right. Each network contains separate virtual machines: A, B, C, and D. Because VM's A and B are in the same network, Network 1, they can communicate using their internal IP address 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. VM C and D however are not in the same network. Therefore by default these VM's must communicate using their external IP addresses even though they are in the same region. The traffic between VM C and D isn't actually touching the public Internet but is going through the Google edge routers. This has different billing and security ramifications that we will explore later.



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 a GCP network as shown in this diagram. Even though the two VM instances are in separate regions, US-West 1 and US-East 1, they leverage Google's private network to communicate between each other and to an on-premises network through a VPN gateway. This reduces cost and network management complexity. I mentioned that subnetworks work on a regional scale. Because a region contains several zones, subnetworks can cross zones.

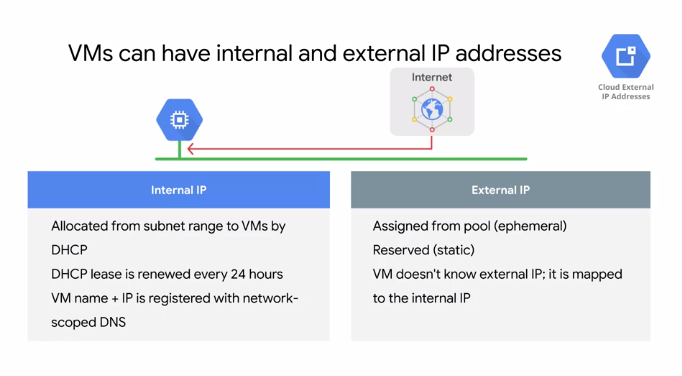


This slide has a region, Region 1 with two zones: zones A and B. Subnetworks can extend across these zones within the same region such as subnet-1. The subnet is simply an IP address range and you can use IP addresses within that range. Notice that the first and second addresses in the range 10.0.0.0 and 10.0.0.1 are reserved for the network and these subnets gateway respectively. This makes the first and second available addresses 10.0.0.2 and 10.0.0.3 which are assigned to the VM instances. The other reserved addresses in every subnets are the second-to-last address in the range and the last address which is reserved as the broadcast address. So to summarize, every subnet has four reserved IP addresses in its primary IP range. 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 VM's even though they are in different zones. Speaking of IP addresses of a subnet, Google Cloud VPC's let you increase the IP address space of any subnets without any workload shutdown or downtime.

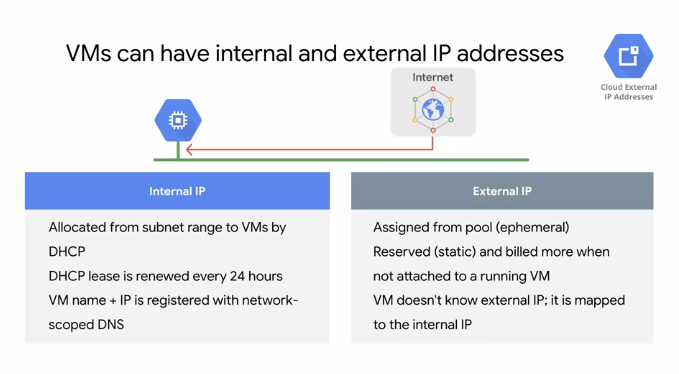


This diagram illustrates a network with subnets that have different subnet masks allowing for more instances in some subnets than others. This gives you flexibility and growth options to meet your needs but there are some things to remember. The new subnet must not overlap with other subnets in these same VPC network in any region. Also, the new subnets must stay inside the RFC 1918 address spaces. 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 IP range further. Also avoid creating large subnets. Overly large subnets are more likely to cause site arrange 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.

**IP addresses**



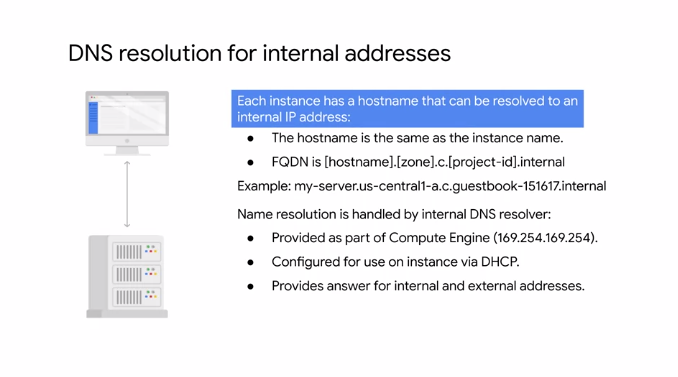
Now that we covered GCP networks at a high level, let's go deeper by exploring IP addresses. In GCP, each virtual machine can have two IP addresses assigned. One of them is an internal IP address, which is going to be assigned via DHCP internally. Every VM that starts up and any service that depends on virtual machines gets an internal IP address. Example of such services are App Engine and Kubernetes Engine, which are explored in other courses. When you create a VM in GCP, it's 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 you 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.



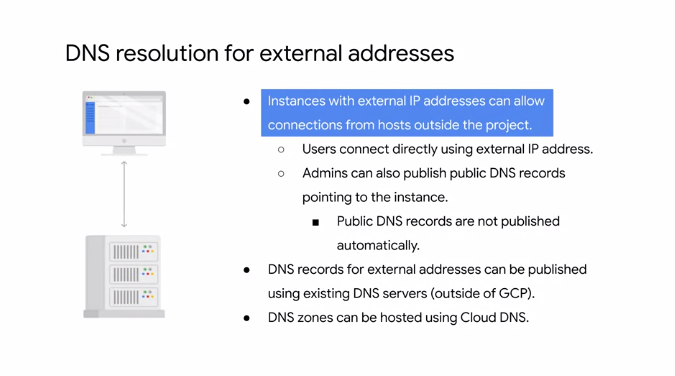
If you preserve a static external IP address and do not assign it to a resource such as a VM instance or a forwarding rule, you are charged at a higher rate than for static and ephemeral external IP addresses that are in use.

**Mapping IP addresses**

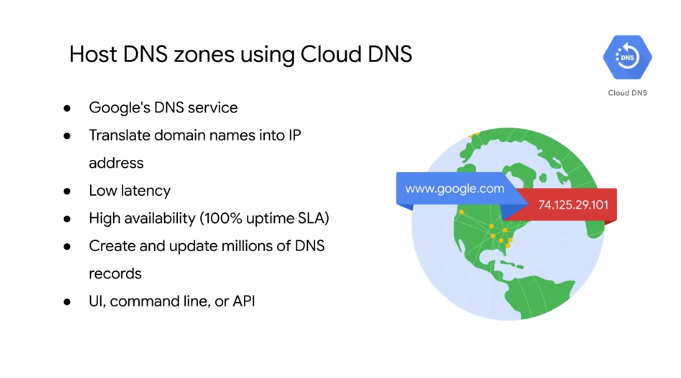
Regardless of whether you use an ephemeral or static IP address, the external address is unknown to the OS of the VM. The external IP address is mapped to the VMs internal address transparently by VPC. I'm illustrating this here by running if config within a VM in TCP, which only returns the internal IP address. Let's explore this further by looking at DNS resolution for both internal and external addresses.



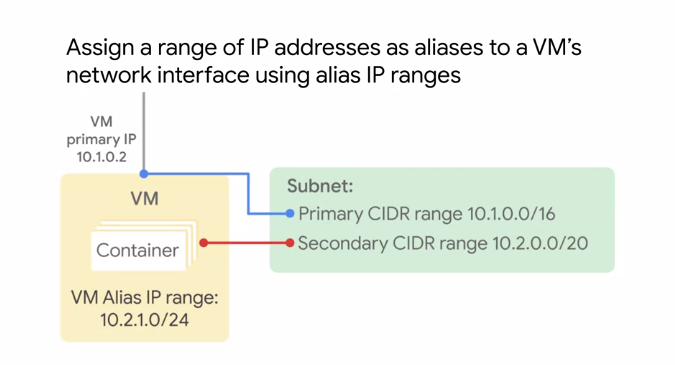
Let's start with internal addresses, each in Instance has a host name that can be resolved to an internal IP address. This hostname is the same as the instance name. There's also an internal fully qualified domain name or fqdn for an instance that uses the format shown on the slide. If you delete and recreate an instance, the internal IP address can change. This change can disrupt connections from other compute engine resources, which must obtain the new IP address before they can connect again. However, the DNS name always points to specific instance no matter what the internal IP address is. Each instance has a metadata server that also acts as a DNS resolver for that instance. The metadata server handles all DNS queries for local network resources and routes all other queries to Google's public DNS servers for public name resolution. I previously mentioned that an instance is not aware of any external IP address assigned to it. Instead, the network stores a lookup table that matches external IP addresses with the internal IP addresses of the relevant instances. For more information including how to setup your own resolve on instances see the link section of this video.



Now let's look at external addresses. Instances with external IP addresses can allow connections from hosts outside of the project. Users can do so directly using the external IP address. Public DNS records pointing to instances are not published automatically. However, admins can publish these using existing DNS servers. Domain and servers can be hosted on gcp using Cloud DNS. This is a managed service that's definitely worth considering, so let's explore them more detail. Cloud DNS is a scalable, reliable and managed authoritive domain name system or DNS service running on the same infrastructure as Google. Cloud DNS translate requests for domain names like google.com into IP addresses.

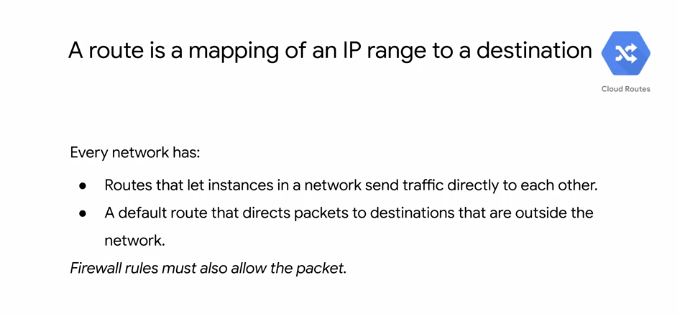


Cloud DNS uses Google's Global Network of any cast name servers to serve your DNS zones from a download locations around the world providing lower latency and high availability for your users. High availability is very important because if you can't look up a domain name the internet might as well be down. That's why gcp offers a 100% up-time service level agreement or SLA for domains configured in cloud DNS. For more information about this SLA see the link section of this video. Cloud DNS lets you create and update millions of DNS records without the burden of managing your own DNS service and software. Instead, you use a simple user interface, command line face or API. For more information about cloud DNS see the link section of this video.

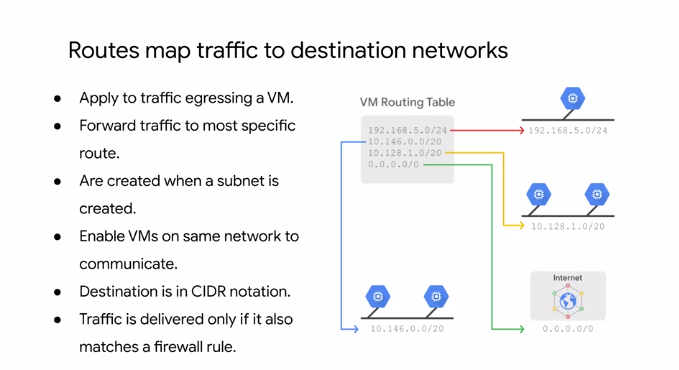


Another networking feature of gcp is alias IP ranges, alias IP ranges that you assign a range of internal addresses as an alias to Virtual machines network interface. This is useful if you have multiple services running on a VM and you want to assign a different IP address to each service. In essence, you can configure multiple IP addresses representing containers or applications hosted in a VM. Without having to define a separate network interface. You just draw the alias IP range from the local subnets primary or secondary side arranges. This diagram provides a basic illustration of primary and secondary site arranges and VM Alias IP ranges. For more information about alias IP ranges see the link section of this video.

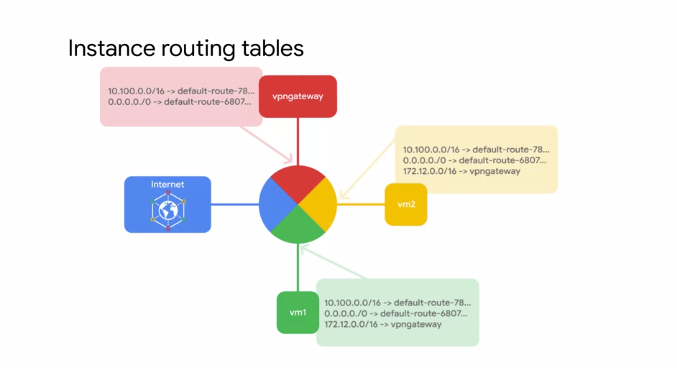
**Routes and firewall rules**

****

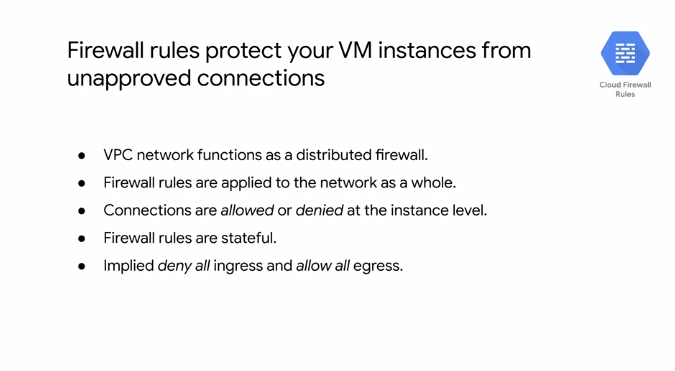
So far you've learned about projects, networks, subnetworks, and IP addresses. Let's use what you learned to understand how GCP routes traffic. By default, every network has routes that let instances in a network send traffic directly to each other even across subnets. In addition, every network has a default route that directs packets to destinations that are outside the network. Although these routes cover most of your normal routing needs, you can also create special routes that overwrite these routes. Just creating a route does not ensure that your packet will be received by the specified next top. Firewall rules must also allow the packet. The default network has preconfigured firewall rules that allow all instances in the network to talk with each other. Manually created networks do not have such rules, so you must create them as you will experience in the first lab. Routes match packets by destination IP addresses.



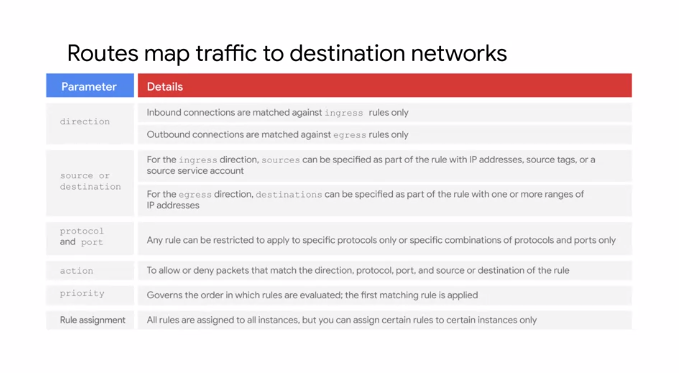
However, no traffic will flow without also matching a firewall rule. A route is created when a network is created, enabling traffic delivery from anywhere. Also, a route is created when a subset is created. This is what enables VM's on the same network to communicate.



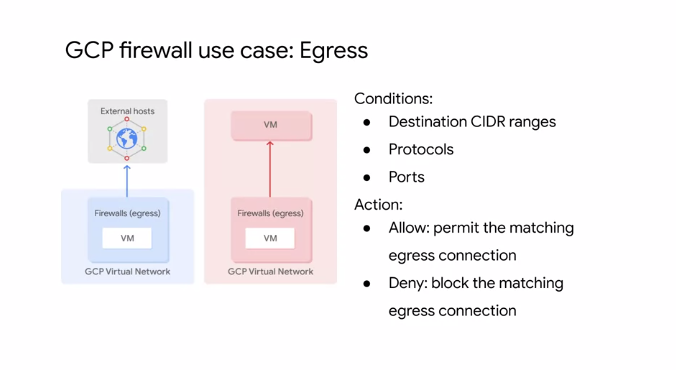
This slide shows a simplified routing table, but let's look at this in more detail. Each route in the routes collection may apply to one or more instances. A route applies to an instance if the network and instance tags match. If the network matches and there are no instance tags specified, the route applies to all instances in that network. Compute engine then uses the routes collection to create individual read-only routing tables for each instance. This diagram shows a massively scalable virtual router at the core of each network. Every virtual machine instance in the network is directly connected to this router, and all packets leaving a virtual machine instance are first handled at this layer before they are forwarded to the next hop. The virtual network router selects the next hop for a packet by consulting the routing table for that instance.



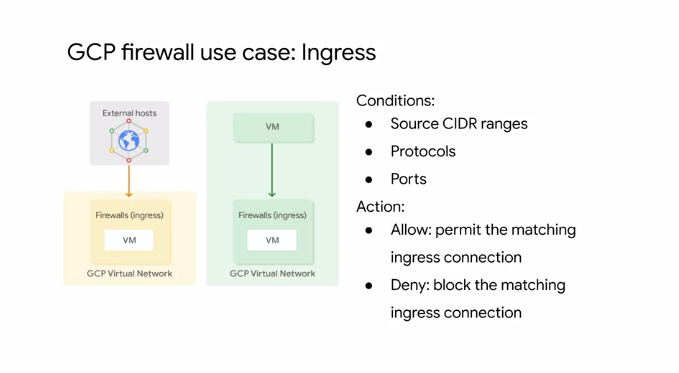
GCP firewall rules to protect you virtual machine instances from unapproved connections both inbound and outbound known as ingress and egress respectively. Essentially, every VPC network functions as a distributed firewall. Although firewall rules are applied to the network as a whole, connections are allowed or denied at the instance level. You can think of the firewall as existing not only between your instances and other networks, but between individual instances within the same network. GCP firewall rules are stateful. This means that if a connection is allowed between a source and a target or a target at 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. You can express your desired firewall configuration as a set of firewall rules.



Conceptually, a firewall rule is composed of the following parameters: the direction of the rule. Inbound connections are matched against ingress rules only, and outbound connections are matched against egress rules only. The source of the connection for ingress packets or the destination of the connection for egress packets. The protocol and port of the connection where any rule can be restricted to apply to specific protocols only or specific combinations of protocols imports only. The action of the rule which is to allow or deny packets that match the direction, protocol port and source or destination of the rule. The priority of the rule which governs the order in which rules are evaluated. The first matching rule is applied. The rule assignment. By default all rules are assigned to all instances but you can assign certain rules to certain instances only. For more information on firewall rule components, please refer to the links section of this video. Let's look at some GCP firewall or use cases for both egress and ingress.

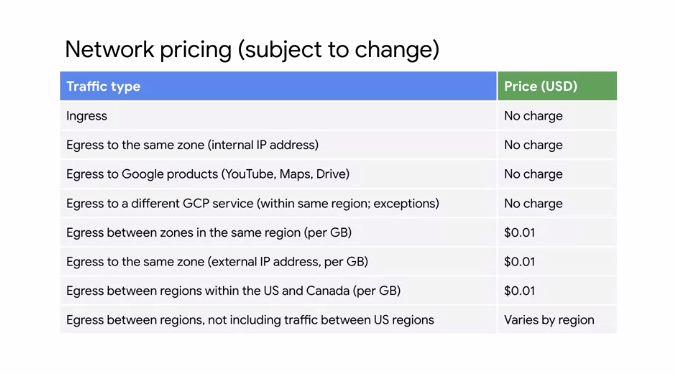


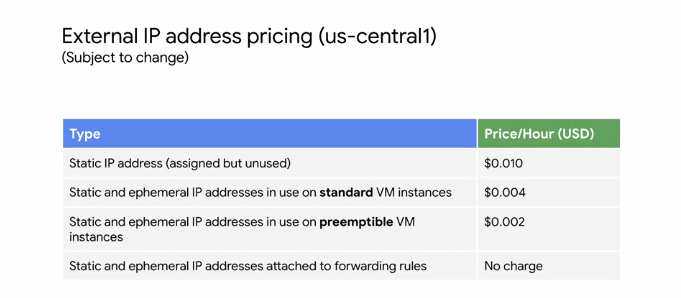
**Egress** firewall rules control outgoing connections originated inside your GCP network. Egress allow rules allow outbound connections that match specific protocol ports and IP addresses. Egress deny rules prevent instances from initiating connections that match non permitted port protocol and IP range combinations. For egress firewall rules, destinations to which a rule applies may be specified using IP CIDR ranges. Specifically, you can use the destination ranges to protect from undesired connections initiated by a VM instance towards an external host as shown on the left. You can also use destination ranges to prevent undesired connections from internal VM instances to specific GCP CIDR ranges. This is illustrated in the middle, where a VM in a specific subnet is shown attempting to connect inappropriately to another VM within the same network.



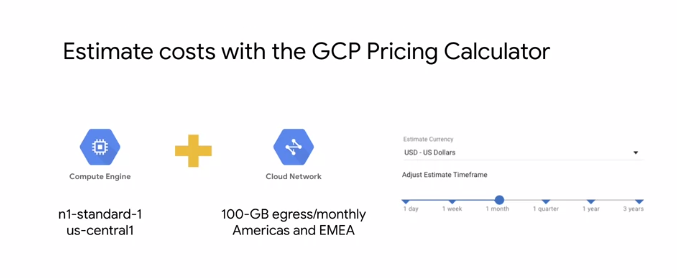
**Ingress** firewall rules protect against incoming connections to the instance from any source. Ingress allow rules allow specific protocol ports and IP ranges to connecting. The firewall prevents instances from receiving connections on non-permitted ports and protocols. Rules can be restricted to only affect particular sources. Source CIDR ranges can be used to protect an instance from undesired connections coming either from external networks or from GCP IP ranges. This diagram illustrates a VM receiving a connection from an external address, and another VM receiving a connection from a VM within the same network. You can control ingress connections from a VM instance by constructing inbound connection conditions using source CIDR ranges, protocols, or ports.

**Pricing**

****

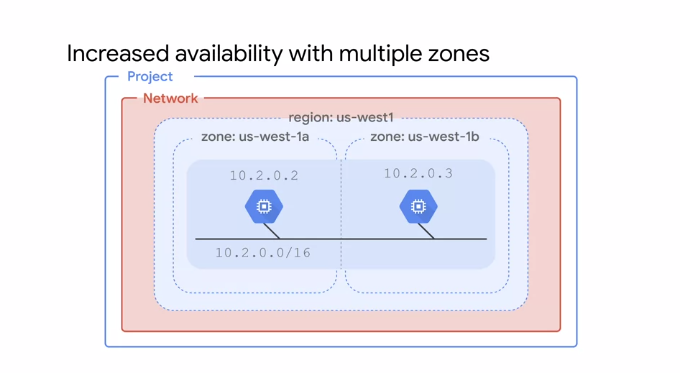
****

Before you apply what you just learned, let's talk about network pricing. It is important that you understand the circumstances in which you are built for GCP's network. This table, is from the Compute Engine documentation and it lists the price of each traffic type. First of all, egress or traffic coming into GCP's network is not charged, unless there is a resource, such as a load balancer that is processing egress traffic. Responses to request account as egress and are charged. The rest of this table, lists egress or traffic leaving a virtual machine. Egress traffic to the same zone, is not charged as long as that egress is through the internal IP address of an instance. Also egress traffic to Google products like YouTube, maps, drive, or traffic to a different GCP service within the same region, is not charged for. However, there is a charge for egress between zones in the same region, egress within a zone, if the traffic is through the external IP address of an instance, and egress between regions. As for the difference in egress traffic to the same zone, Compute Engine cannot determine the zone of a virtual machine through the external IP address. Therefore, this traffic is treated like egress between zones in the same region. Also there are some exceptions and pricing can always change. So refer to the documentation in the links section of these slides. Now, you are charged for static and ephemeral external IP addresses. This table, represents the external IP pricing for us-central1 as of this recording. You can see that if you reserve a static external IP address and do not assign it to a resource, such as a VM instance or a forwarding rule, you are charged at a higher rate and for static and ephemeral external IP addresses that are in use. Also external IP addresses on preemptible VMs, have a lower charge than for standard VM instances. Remember, pricing can always change, so please refer to the documentation link in the slides.

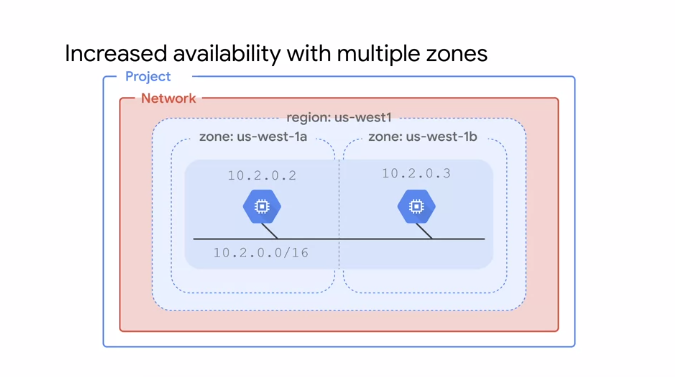


Also I recommend using the GCP pricing calculator to estimate the cost of a collection of resources, because each GCP service has its own pricing model. The pricing calculator is a web-based tool, that you use to specify the expected consumption of certain services and resources, and it then provides you with an estimated cost. For example, you can specify a specific instance type, in a specific region along with 100 gigabytes of monthly egress traffic to Americas and EMEA. The pricing calculator then returns the total estimated cost. You can adjust the currency and time frame to meet your needs, and when you finish, you can e-mail the estimate or save it to a specific URL for future reference. To use the pricing calculator today, refer to the link in the slides.

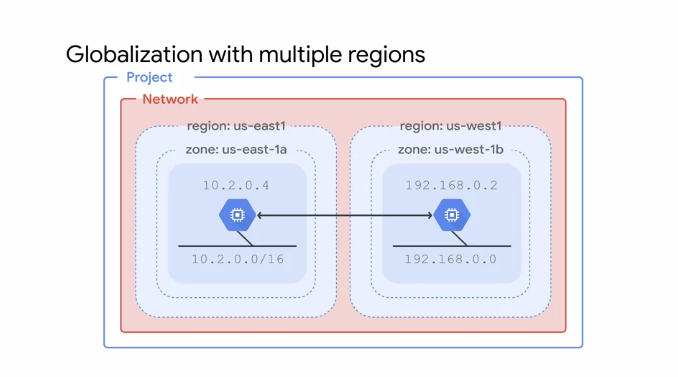
**Common Network designs**

****

Let's use what we have learned so far and look at common network designs. Now, common is a fairly relative term, while I could spend all day talking about network designs, I have picked a handful of designs that best relate to this module.

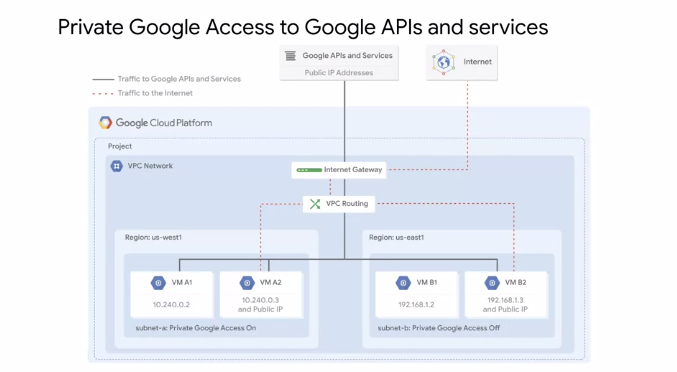


Let's start by looking at availability. If your application needs increased availability, you can place two virtual machines into multiple zones, but within the same subnet work as shown on this slide. Using a single sub-network allows you to to create a file a rule against the sub-network, in this case, 10.2.0.0/16. Therefore, by allocating VMs on a single subnet to separate zones, you get improved availability without additional security complexity. A regional managed instance group contains instances from multiple zones across the same region, which provides increased availability.



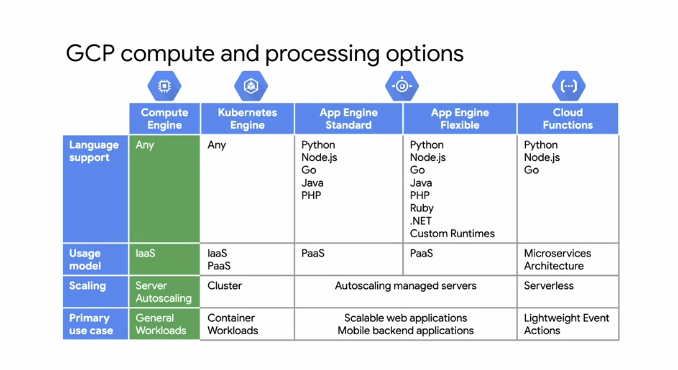
Next, let's look at globalization. In the previous design we placed resources in different zones in a single region, which provides isolation for many types of infrastructure, hardware and software failures.

Putting resources in different regions as shown on this slide provides an even higher degree of failure independence. This allows you to design robust systems with resources spread across different failure domains. When using a global load balancer like the HTTP load balancer, you can route traffic to the region that is closest to the user. This can result in better latency for users and lower network traffic costs for your project. We'll explore both managed instance groups and load balancers later in this course series. Now, as a general security best practice, I recommend only assigning internal IP addresses to your VM instances whenever possible. Cloud NAT is Google's managed network address translation service. It lets you provision your application instances without public IP addresses, while also allowing them to access the internet in a controlled and efficient manner. This means your private instances can access the internet for updates, patching, configuration management, and more. In this diagram Cloud NAT enables two private instances to access an update server on the Internet, which is referred to as outbound NAT. However, Cloud NAT does not Implement inbound NAT. In other words, hosts outside your VPC network cannot directly access any of the private instances behind the cloud NAT gateway. This helps you keep your VPC networks isolated and secure. Similarly, you should enable private Google access to allow VM instances that only have internal IP addresses to reach the external IP addresses of Google APIs and services. For example, if your private VM instance needs to access a cloud storage bucket, you need to enable private Google access. You enable private Google access on a subnet by subnet basis. As you can see in this diagram, subnet A has private Google access enabled and subnet B has it disabled. This allows VMA one to access Google APIs and services, even though it has no external IP address.

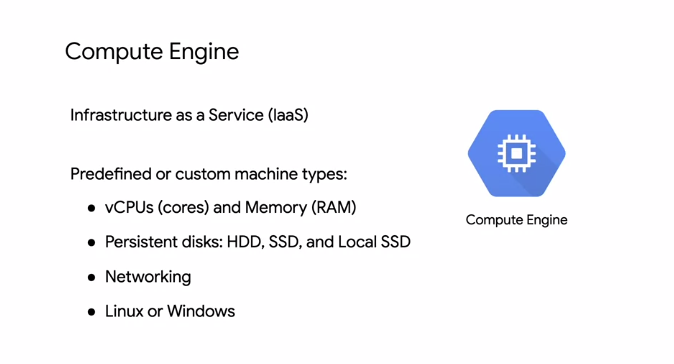


Private Google access has no effect on instances that have external IP addresses, that's why VMs A2 and B2 can access Google APIs and services. The only VM that can't access those APIs and services is VM B1. This VM has no public IP address and it is in a subnet where Google private access is disabled.

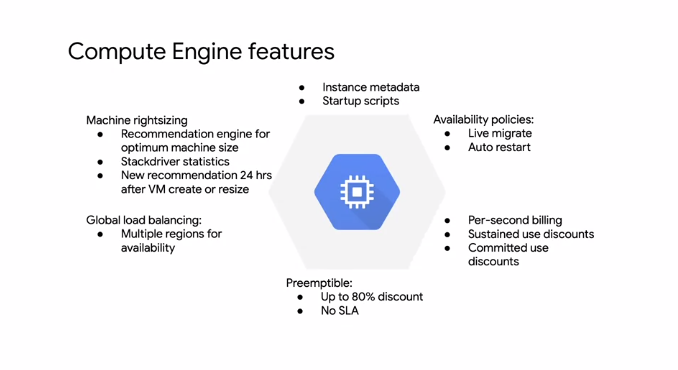
**Compute engine**

****

As I mentioned in the introduction to the course, there is a spectrum of different options in GCP for compute and processing. We will focus on the traditional virtual machine instances. Now the difference is Compute Engine gives you the utmost inflexibility. Run whatever language you want, it's your virtual machine. This is purely an Infrastructure as a Service or IaaS model. You have a VM and an operating system and you can choose how to manage it and how to handle aspects such as autoscaling, where you'll configure the rules about adding more virtual machines in specific situations. Autoscaling will be covered in a later course of this series. The primary work case of Compute Engine is any general workload, especially an enterprise application that was designed to run on a server infrastructure. This makes Compute Engine very portable and easy to run in the Cloud. Other services like Google Kubernetes Engine, which consist of containers workloads may not be as easily transferable as what you're used to find On-premises.



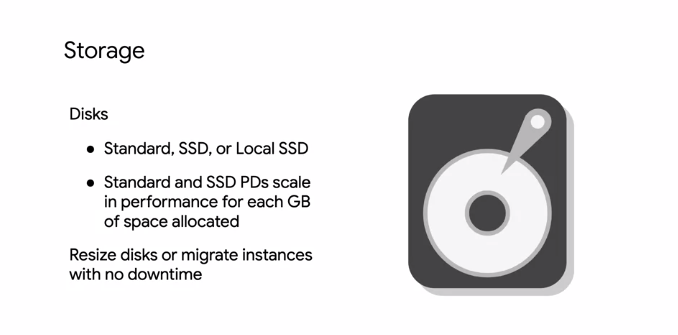
So what it is Compute Engine? As it's heard, it's physical servers that you're used to running inside the GCP environment with a number of different configurations. Both predefined and custom machine types allow you to choose how much memory and how much CPU you want. You chose the type of disk you want, what do you want to just use standard hard drives, SSDs, local SSDs, or a mix. You can even configure the networking interfaces and run a combination of Linux and Windows machines.



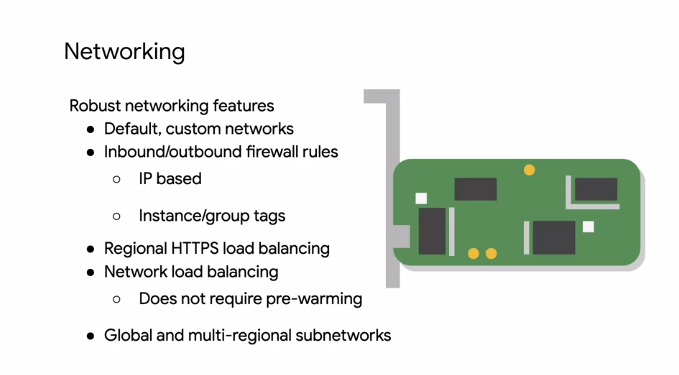
Several different features will be covered throughout this module such as machine rightsizing, startup scripts, metadata, availability policies, and pressing, and usage discounts. Let's start by looking at the compute options.



Compute Engine provides several different machine types that we'll discuss later in this module. If those machines don't meet your needs, you can also customize your own machine. Your choice of CPU will affect your network throughput. Specifically, your network will scale at two gigabits per second for each CPU core, except for instances with two and four virtual CPUs which receive up to 10 gigabits per second of bandwidth. As of this recording, there's a theoretical maximum throughput of 32 gigabits per second for an instance with 16 or more CPUs, and 100 gigabits per second maximum throughput for specific instances that have T4 of V100 GPUs attached. When you're migrating from an on-premises setup, you're used to physical cores which have hyper-threading. On Compute Engine, each virtual CPU or vCPU is implemented as a single hardware hyper-thread on one of the available CPU platforms. For an up-to-date list of all the available CPU platforms, refer to the links section of this video.

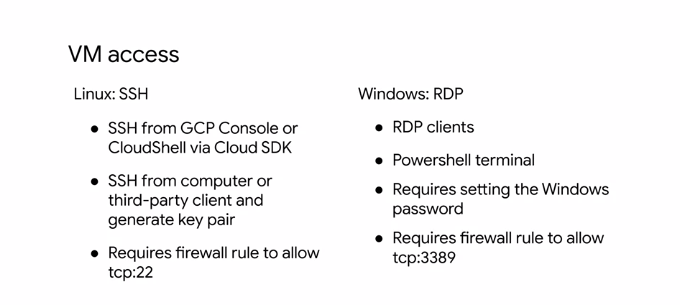


After you pick your compute options, you want to choose your disk. You have three options, standard, SSD, or local SSD. So basically, do you want the standard spinning hard disk drives or HDDs, or flash memory solid state drives SSDs. Both of these options provide the same amount of capacity in terms of disk size when choosing a persistent disk. Therefore, the question really is about performance versus cost because there is a different pricing structure. 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're 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 just like you would do if you want to create a RAM disc. But if you need more capacity, you can store those on a local SSD. You can create instances with up to eight separate 375 gigabytes local SSD partitions for total of three terabytes of local SSD space for each instance. Standard and non-local SSD disks can be sized up to 64 terabytes for each instance. The performance of these disks scales with each gigabyte of space allocated. As for networking, we've already seen network and features applied to Compute Engine in the previous modules lab. We looked at the different types of networks and created firewall rules using IP addresses and network tags.

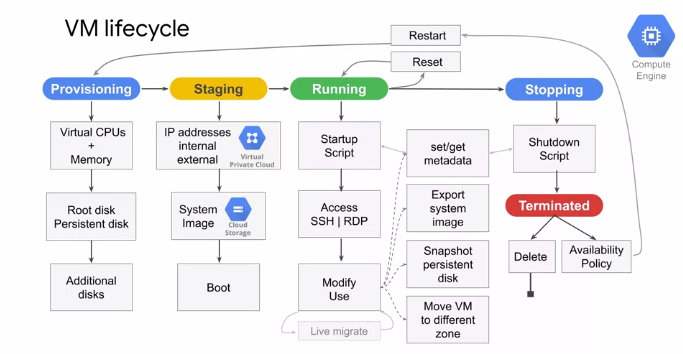


You'll also notice that you can do regional HTTPS load balancing and network load balancing. This doesn't require any pre-warming because a load balancer isn't a hardware device that needs to analyze your traffic. A load balancer is essentially a set of traffic engineering rules that are coming into the Google network. VPC is applying the rules destined for your IP address subnet range. We'll learn more about load balancers in a later course of the architecting with Google Compute Engine series.

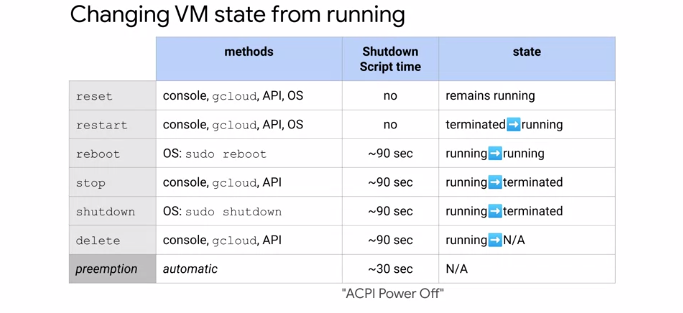
**VM access and lifecycle**

****

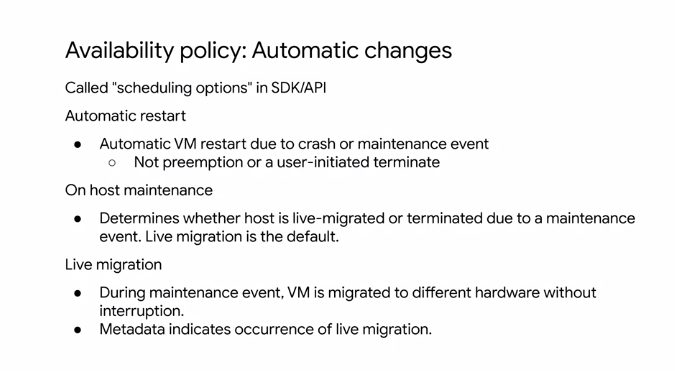
For accessing a VM, the creator of an instance has four root privileges on that instance. On a Linux instance, the creator has SSH capability and can use the GCP Console to grant SSH capability to other users. On a Windows instance, the creator can use the GCP Console to generate a username and password. After that, anyone who knows the username and password can connect to the instance using a remote desktop protocol or RDP client. I listed the required firewall rules for both SSH and RDP here but you don't need to define these if you're using the default network that we covered in the previous module. For more information on SSH key management and creating passwords for Windows instances, refer to the links section of this video.



The life cycle of a VM is represented by different statuses. I will cover this life cycle on a high level but I recommend returning to this diagram as a reference. 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 disk are being reserved for the instance but the instance itself isn't running yet. Next, the instant 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 GCP to perform maintenance that is integral to keeping the infrastructure protected and reliable without interrupting any of your VMs. While you're instance is running, you can also move your VM to a different zone. Take a snapshot of the VMs 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 instance which would bring it back to its provision state or delete it. You also have the option to reset a VM which is similar to pressing the reset button on your computer. This action wipes the memory content of the machine and resets the virtual machine to its initial state. The instance remains in the running state throughout the reset. There are different ways you can change a VM state from running.



Some methods involve the GCP Console and the GCloud command while others are performed from the OS such as for a reboot and shut down. It's important to know that if you're restarting, rebooting, stopping, or even deleting an instance, the shutdown process will take about 90 seconds. For a preemptible VM, if the instance is not stopped after 30 seconds, Compute Engine sends an ACPI G3 mechanical off signal to the operating system. Remember that when writing shutdown scripts for preemptible VMs.



As I mentioned previously, Compute Engine can live migrate your virtual machine to another host due to a maintenance event to prevent your applications from experiencing disruptions. A VMs availability policy determines how they instance behaves in such an event. The default maintenance behavior for instances is to live migrate, but you can change the behavior to terminate your instance during maintenance events instead. If your VM is terminated due to a crash or other maintenance event, your instance automatically restarts by default but this can also be changed. These availability policies can be configured both during the instance creation and while an instance is running by configuring the automatic restart and on host maintenance options. For more information on live migration, refer to the link section of this video. When a VM is terminated, you do not pay for memory and CPU resources.

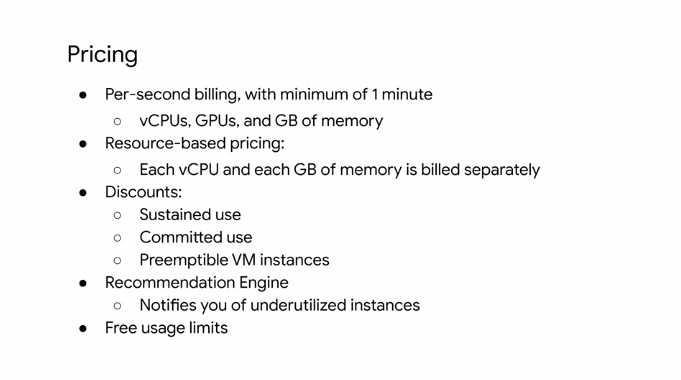


However, you are charged for any attached disks and reserved IP addresses. In the terminated state, you can perform any of the actions listed here such as changing the machine type, but you cannot change the image of a stopped VM. Also, not all of the actions listed here require you to stop a virtual machine. For example, VM availability policies can be changed while the VM is running as discussed previously.

**Compute options**

Now that you have completed the lab, let's dive deeper into the compute options that are available to you in GCP, by focusing on CPU and memory. You have three options for creating and configuring a VM. You can use the GCP console as you did in the previous lab, the Cloud Shell command-line, or the RESTful API. If you'd like to automate and process very complex configurations, you might want to programmatically configure these through the RESTful API by defining all the different options for your environment. If you plan on using the command line or RESTful API, I recommend that you first configure the instance through the GCP Console, and then ask Compute Engine for the equivalent REST request or command-line as I showed you in my demo earlier. This way you avoid any typos and get drop-down lists of all the available CPU and memory options. Speaking of CPU memory options, let's look at all the different machine types that are currently available. A machine type specifies a particular collection of virtual hardware resources available to a VM instance, including the system memory size, vCPU count, and maximum persistent disk capability. GCP offers several machine types that can be grouped into two categories. Predefined machine types. These have fixed collection of resources, are managed by Compute Engine, and are available in multiple different classes. Each class has a predefined ratio of gigabytes of memory per Virtual CPU. These are the standard machine types, high memory, high CPU, memory optimized, Compute optimize, and shared-core machine types. There also the custom machine types. These lets you specify the number of virtual CPUs, and the amount of memory for your instance. Let's explore each of these Machine types. But remember that these Machine types and the available options can change. Standard machine types are suitable for tasks that have a balance of CPU and memory needs. Standard machine types have 3.75 gigabytes of memory per virtual CPU. The virtual CPU configurations come in different intervals from 1vCPU all the way to 96 vCPUs as shown on this table. Each of these machines supports a maximum of 128 persistent disks with a total persistent disk size of 64 terabytes, which is also the case for the high memory, high CPU, memory optimized, and compute optimized machine types. High memory machine types are ideal for tasks that require more memory relative to vCPUs. High memory machine types have 6.5 gigabytes of system memory per vCPU. Similar to the stent machine types, the vCPU configurations come in different intervals from 2vCPUs all the way to 96vCPUs, as shown on this table. High CPU machine types are ideal for tasks that require more vCPUs relative to memory. High CPU machine types have 0.9 gigabytes of memory per vCPU. Memory optimized machine types are ideal for tasks that require intensive use of memory with higher memory to vCPU ratios than high memory machine types. These machine types are perfectly suited for in-memory databases and in-memory analytics such as SAP HANA and business warehouse workloads, genomic analysis, and SQL Analysis Services. Memory optimized machine types have more than 14 gigabytes of memory per vCPU. These Machines come in four configurations as shown in this table, with only the n1-megamem-96 supporting a local SSD, as of this recording. Compute optimized machine types are ideal for compute intensive workloads. These machine types are for the highest performance per core on Compute Engine. Built on the latest generation Intel scalable processors, the casket lake, C2 machine types offer up to 3.8 gigahertz sustained all-core turbo, and provide full transparency into the architecture of the underlying server platforms, enabling advanced performance tuning. C2 machine types offer much more computing power, run on a newer platform, and are generally more robust for compute intensive workloads than the n1 high CPU machine types. Shared-core machine types provide one virtual CPU that is allowed to run for a portion of the time on a single hardware hyper-thread on the host CPU running your instance. Shared-core instances can be more cost effective for running small non resource intensive applications than other machine types. There are only two shared-core machine types to choose from, they're the f1-micro and the g1-small. The f1-micro machine types offer bursting capabilities that allow instances to use additional physical CPU for short periods of time. Bursting happens automatically when your instance requires more physical CPU than you originally allocated. During these spikes, your instance will opportunistically take advantage of available physical CPU in bursts. Note that bursts are not permanent and are only possible periodically. For up-to-date information about all of these machine types, see the link section of this video. If none of the predefined machine types match your needs, you can independently specify the number of vCPUs and the amount of memory for your instance. Custom machine types are ideal for the following scenarios; when you have workloads that are not a good fit for the predefined machine types that are available to you, or when you have workloads that require more processing power or more memory but you don't need all of the upgrades that are provided by the next larger predefined machine type. It cost slightly more to use a custom machine type than equivalent predefined machine type. There are still some limitations in the amount of memory and vCPUs you can select. Only machine types with one virtual CPU or an even number of virtual CPUs can be created. Memory must be between 0.9 gigabytes and 6.5 gigabytes per virtual CPU by default. The total memory of the instance must be a multiple of 256 megabytes. By default, a custom machine can have up to 6.5 gigabytes of memory per virtual CPU. However, this might not be enough memory for your workload. So at an additional cost, you can get more memory per virtual CPU beyond the 6.5 gigabytes limit. This is referred to as extended memory, and you can learn more about this in the links section of this video. The first thing you want to consider when choosing a region and zone is the geographical location in which you want to run your resources. This map shows the current and planned GCP regions and the number of zones. For up-to-date information on the available regions and zones, see the documentation linked for this video. Each zone supports a combination of Ivy Bridge, Sandy Bridge, Haswell, Broadwell, and Skylake platforms. When you create an instance in the zone, your instance will use the default processes supported in that zone. For example, if you create an instance in the US-central1 a zone, your instance will use the Sandy Bridge processor.

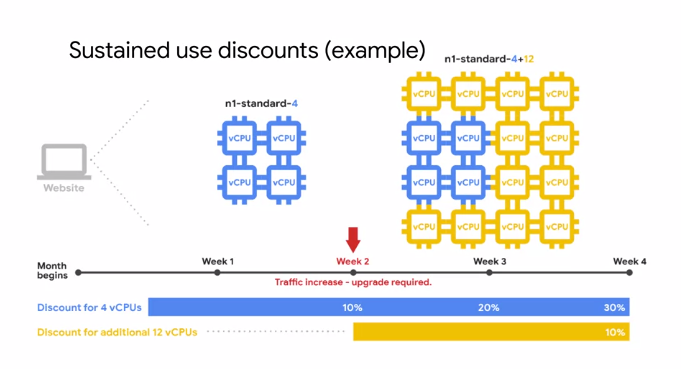
**Compute pricing**

****

GCP offers a variety of different options to keep the prices low for compute engine resources. All of these vCPUs, GPUs and gigabyte of memory are charged a minimum of one minute. For example if you've run your virtual machine for 30 seconds, you will bill it for one minute of usage. After one minute, instances are charged in one second increments. Compute Engine uses a resource-based pricing model where each virtual CPU and each gigabyte of memory on Compute Engine is built separately rather than as part of a single machine type. You still create instances using predefined machine types, but your bill reports them as individual vCPUs and memory used. There are several discounts available but the disk on types cannot be combined. There are research-based pricing which allows Compute Engine to apply sustained use discounts to all of your predefined machine types usage in a region collectively rather than to individual machine types.

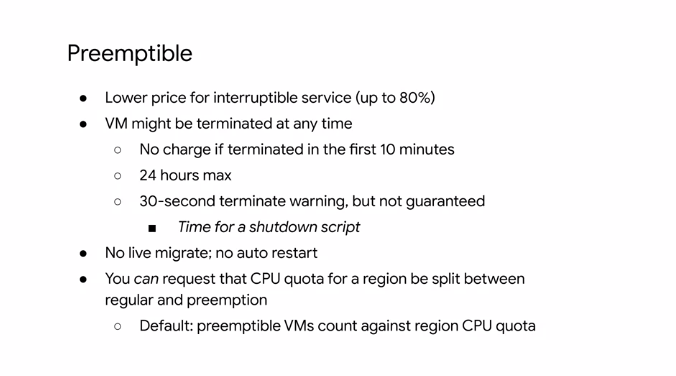


If you're workload is stable and predictable, you can purchase a specific amount of vCPU and memory for a discount off of normal prices in return for committing to a usage term of one or three years. The discount is up to 57 percent for most machine types or custom machine types. The discount is up to 70 percent for memory optimized machine types. A preemptible VM is instance that you can create and run at much lower price than normal instances. However, Compute Engine might terminate or preempt these instances if it requires access to those resources for other tasks. Preemptible instances are access Compute Engine capacity, so their availability varies with usage. The ability to customize the amount of memory in CPU through custom machine types allows for further pricing customization. Speaking of sizing your Machine, Compute Engine provides VM sizing recommendations to help you optimize the resource use of your virtual machine instances. When you create a new instance, recommendations for the new instance will appear 24 hours after the instance has been created. Compute Engine also has Free usage limits, for the exact terms, please refer to the links section of this video. Sustained use discounts are automatic discounts that you get for running specific Compute Engine resources, be it CPUs, memory, and GPU devices for a significant portion of the billing month. For example when you run one of these resources for more than 25 percent of a month, Compute Engine automatically gives you a discount for every incremental minute you use for that instance. The discount increases with usage, and you can get up to 30 percent net discount for instances that run the entire month. This table shown on this slide describes the discount you get at each usage level of a VM instance. To take advantage of the full 30 percent discount, create your VM instances on the first day of the month because discounts reset at the beginning of each month. The graph on this slide demonstrates how your effective discount increases with use. For example, if you use a virtual machine for 50 percent of the month, you get an effective discount of 10 percent. If you use it for 75 percent of the month, you get an effective discount of 20 percent, and if you use it for a 100 percent of the month, you get an effective discount of 30 percent. You can also use the GCP pricing calculator to estimate your sustained use discount for any arbitrary workload. For the calculator, see the links section of this video. Compute Engine calculates sustained use discounts based on vCPU and memory usage across each region, and separately for each of the following categories, pre-defined machine types and custom machine types.

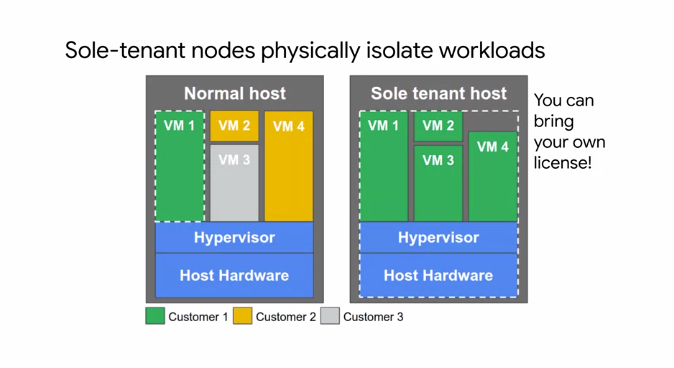


Let's go through an example where you have two instances that are in the same region but have different machine types, and run at different times of the month. Compute Engine breaks down the number of vCPUs and amount of memory used across all instances that use predefined machine types, and combines the resources to qualify for the largest sustained usage discounts possible. As shown on this slide, you run the following two instances in the US Central one region during a month. For the first half of the month, you run an n1-standard-4 instance with four virtual CPUs, and 15 gigabytes of memory. For the second half of the month, you run a larger n1-standard-16 instance with 16 virtual CPUs, and 60 gigabytes of memory. In this scenario, Compute Engine reorganizes these machine types into individual vCPUs and memory resources, it combines their usage to create the following resources as shown on the bottom. Four virtual CPUs and 15 gigabytes of memory for a full month, and then 12 virtual CPUs and 45 gigabytes of memory for half of the month.

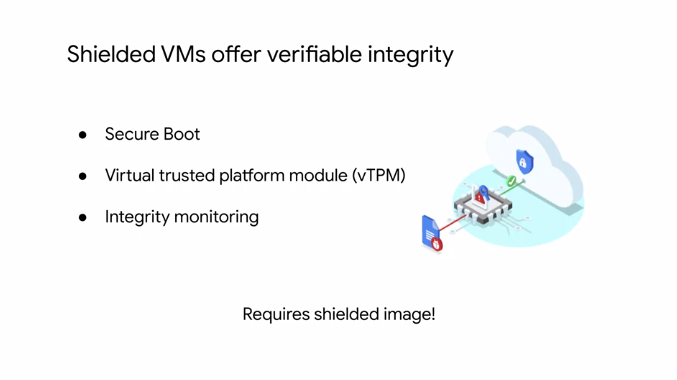
**Special compute configuration**

****

As I mentioned earlier a **preemptible VM** is an instance that you can create and run at much lower prices than normal instances. See whether you can make your application function completely on **preemptible VM**s, because an 80 percent discount is a significant investment in your application. Now, just to reiterate, these VMs might be preempted at any time, and there is no charge if that happens within the first 10 minutes. Also, preemptible VMs are only going to live for up to 24 hours, and you only get a 30-second notification before the machine is preempted. It's also worth noting that there are no life migrations, no automatic resorts in preemptible VMs. But something that we will highlight is that you can actually create monitoring and load balances then can startup new preemptible VMs in case of a failure. In other words, there are external ways to keep restarting preemptible VMs if you need to. One major use case of preemptible VMs is running a batch processing job. If some of those instances terminate during processing, the job slows down but does not completely stop. Therefore preemptible instances complete your batch processing tasks without placing additional workload on your existing instances and without requiring you to pay full price for additional normal instances. If you have workloads that require physical isolation from other workloads, or virtual machines in order to meet compliance requirements, you want to consider sole-tenant nodes.

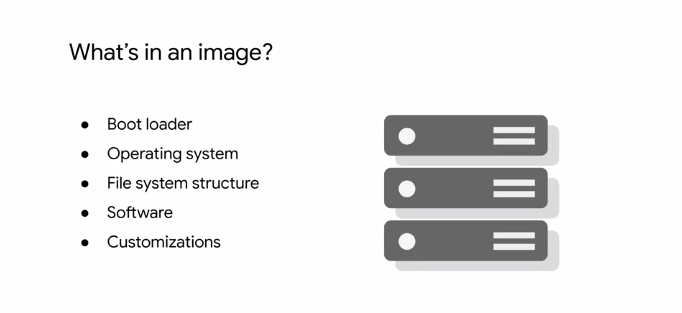


A sole-tenant node is a physical Compute Engine server that is dedicated to hosting VM instances only for your specific project. Use sole-tenant nodes to keep your instances physically separated from instances in other projects, or to group your instances together in the same host hardware. For example if you have a payment processing workload that needs to be isolated to meet the compliance requirements. The diagram on the left shows a normal host with multiple VM instances from multiple customers. A sole-tenant node as shown on the right, also has multiple VM instances, but they all belong to the same project. As of this recording, the only available node type can accommodate VM instances up to 96 V CPUs and 624 gigabytes of memory. You can also fill the node with multiple smaller VM instances of various sizes including custom machine types and instances with extended memory. Also if you have existing operating system licenses, you can bring them to Compute Engine using sole-tenant nodes while minimizing Physical Core usage with the in-place restart feature. To learn how to create nodes and place your instances on those nodes, see the links section of this video.

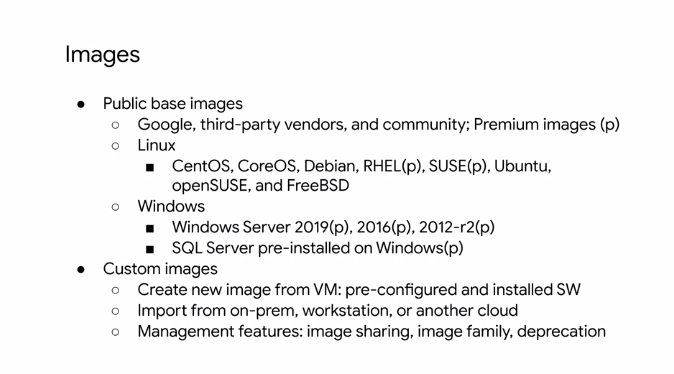


Another Compute option is to create **shielded VMs**. Shielded VMs offer verifiable integrity of your VM instances. So you can be confident that you're instances haven't been compromised by boot or kernel level of malware or rootkits. Shielded VMs verifiable integrity is achieved through the use of secure boot, Virtual Trusted Platform Module or VTPM enabled measured boot and integrity monitoring. Shield VMs is the first offering in the shielded Cloud initiative. The shielded Cloud initiative is meant to provide an even more secure foundation for all of GCP by providing verifiable integrity, and offering features like VTPM shielding or ceiling that help prevent data exfiltration. In order to use the shielded VM features, you need to select a shielded image. We'll learn about images in the next section.

**Images**

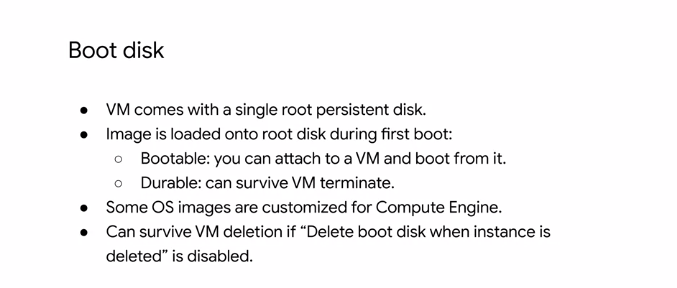
****

Next, let's focus on images. 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.

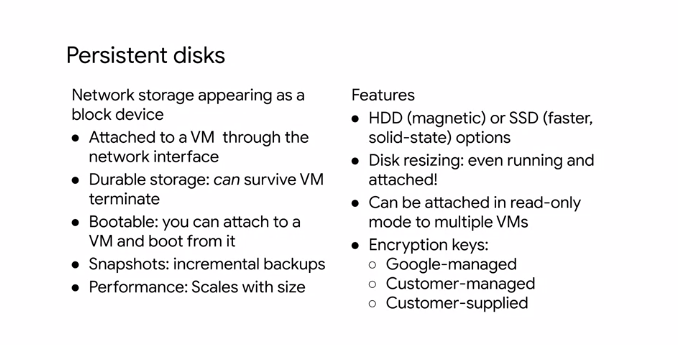


You can select either a **public** or **custom** image. As you saw in the previous lab, you can choose from both Linux and Windows images. Some of these images are premium images as indicated in parentheses with a p. These images will have per second charges after a one-minute minimum, with the exception of SQL Server images, which are charged per minute after a 10-minute minimum. Premium image prices vary with the machine type. However, these prices are global and do not vary by region or zone. You can also use custom images. For example, you can create and use a custom image by pre installing software that's been authorized for your particular organization. You also have the option of importing images from your on-premises or workstation, or from another cloud provider. This is a no cost service that is as simple as installing an agent. I highly recommend that you look at it. You can also share custom images with anybody in your project or among other projects too.

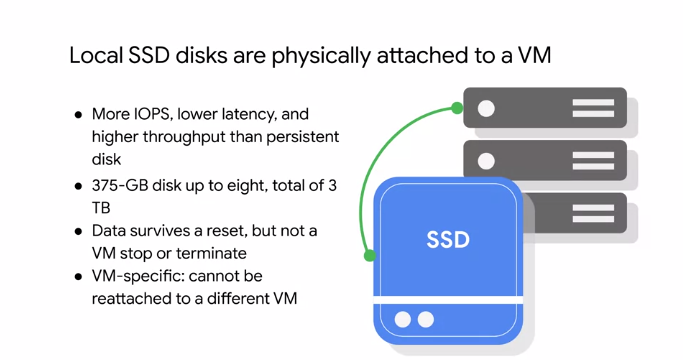
**Disk options**

****

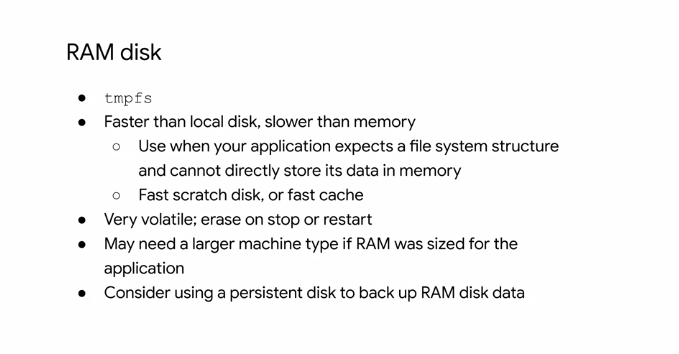
At this point, you've chosen an operating system, but that operating system is going to be included as part of some kind of disk. So let's look at the disk options. Every single VM comes with a single root persistent disk because you're choosing a base image to have that loaded on. This image is bootable and that you can attach it to VM and boot from it. It's durable and that it can survive, if the VM terminates. To have a boot disks survive a VM deletion, you need to disable the delete boot disk when instance is deleted option in the instances properties. As I discussed earlier, there are different types of disks. Let's explore these in more detail.



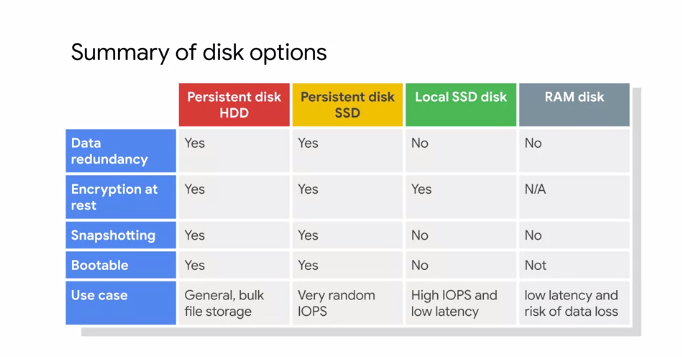
The first is that we create, is what we call a persistent disk. That means it's going to be attached to the VM through the network interface. Even though it's persistent, it's not physically attached to the machine. The separation of disk and compute, allows a disk to survive if the VM terminates. You can also perform snapshots of these disks which are incremental backups that we'll discuss later. The choice between HDD and SSD disk comes down to cost and performance. To learn more about the disk performance and how it scales with disk size, see the links section of this video. Another cool feature of persistent disks, is that you can dynamically resize them, even while they are running and attached to a VM. You can also attach a disk in read only mode to multiple VMs. This allows you to share static data between multiple instances, which is cheaper than replicating your data to unique disks for individual instances. By default, computer engine encrypts all data at rest. GCP handles and manages this encryption for you, without any additional actions on your part. However, if you wanted to control and manage this encryption yourself, you can either use Cloud key management service, to create and manage key encryption keys, which is known as customer managed encryption keys. Or you can create and manage your own key encryption keys known as customer supplied encryption keys.



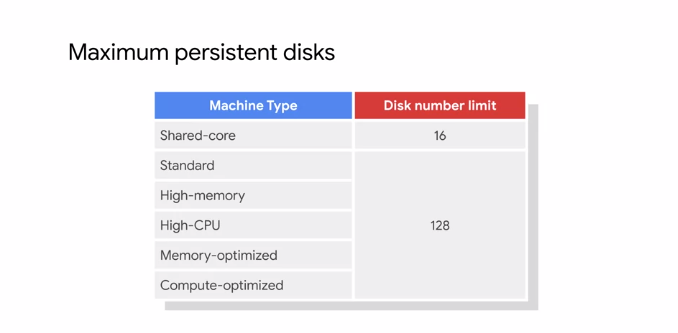
Now, local SSDs are different from persistent disks and that they're physically attached to the virtual machine. Therefore, these disk are ephemeral, but provide very high IOPS. For up to date numbers, I recommend referring to the documentation. Currently, you can attach up to eight local SSD disks with 375 gigabytes each, resulting in a total of three terabytes. Data on these disks will survive a reset, but not a VM stop or terminate. Because these disks can't be reattached to a different VM. You also have the option of using a RAM disk. You can simply use TM PFS if you want to store data in memory.



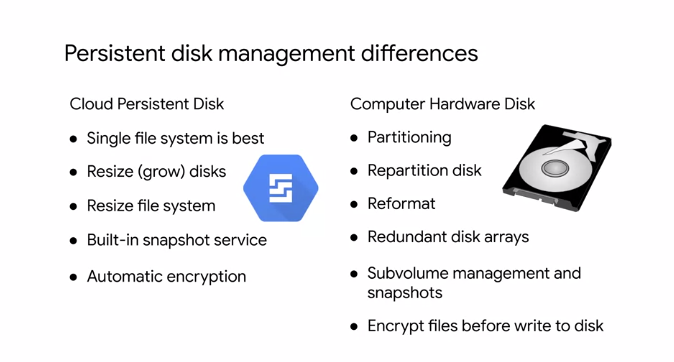
This will be the fastest type of performance available if you need small data structures. I recommend a high memory virtual machine if you need to take advantage of such features, along with a persistent disk to backup the RAM disk data.



In summary, you have several different disk options. Persistent disk can be rebooted and snapshotted, but local SSDs and RAM disks are ephemeral. I recommend choosing a persistent HDD disk when you don't need performance but just need capacity. If you have high-performance needs, start looking at the SSD options. The persistent disk offer data redundancy because the data on each persistent disk is distributed across several physical disks. Local SSDs provide even higher performance but without the data redundancy. Finally, RAM disks are very volatile, but they provide the highest performance.

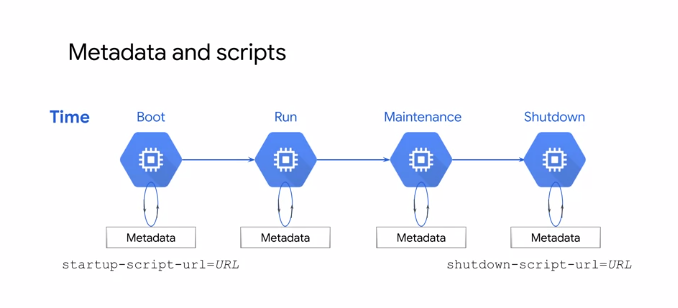


Now, just as there is a limit on how many local SSDs you can attach to VM, there's also a limit on how many persistent disks you can attach to VM. As illustrated in this table, this limit depends on the machine type. For the shared core machine type, you can attach up to 16 disks. For the standard high CPU memory optimized and compute-optimized machine types, you can attach up to 128 disks. So you can create massive amounts of capacity for a single host. Now, remember that little nuance when I told you about how throughput is limited by the number of cores that you have. That throughput also shares the same bandwidth with disk IO. So if you plan on having a large amount of disk IO throughput, you will also compete with any network egress or ingress throughput. So remember that, especially if you'll be increasing the number of drives attached to a virtual machine.

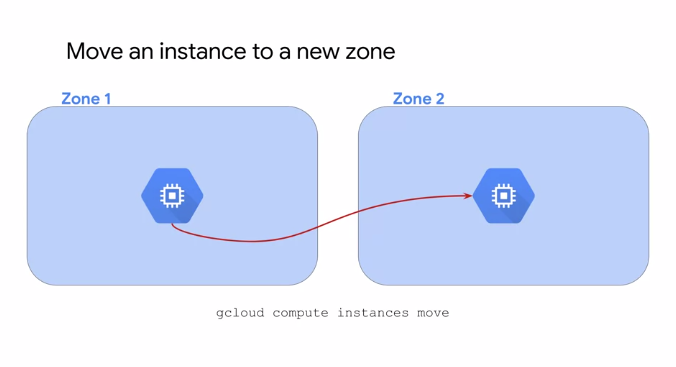


There are many differences between a physical hard disk in a computer and a persistent disk, which is essentially a Virtual Network device. First of all, if you remember with normal computer hardware disks, you have to partition them. Essentially, you have a drive and you're carving up a section for the operating system to get its own capacity. If you want to grow it, you have to repartition it and if you want to make changes you might even have to reformat. If you want redundancy, you might create a redundant disk array and if you want encryption, you need to encrypt files before writing them to the disk. With Cloud persistent disks, things are very different because all that management is handled for you on the back-end. You can simply grow disks and resize a file system because disks are Virtual Network devices. Redundancy and snapshots services are built-in and disks are automatically encrypted. You can even use your own keys and that will ensure that no party can get to the data except you.

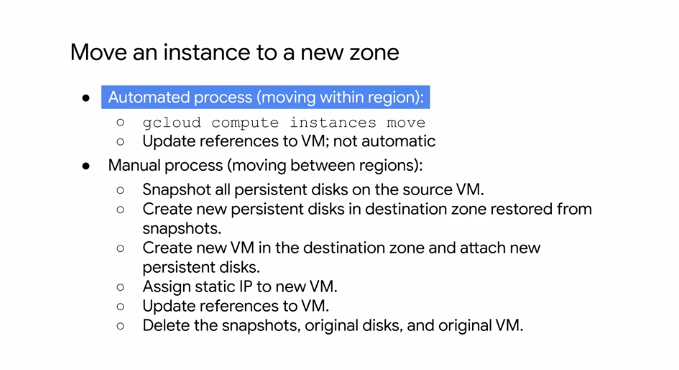
**Common Compute Engine actions**

****

Now that we have covered all the different compute image and disk options, let's look at some common actions that you can perform with Compute Engine. Every VM instance stores its metadata on a metadata server. 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 new script to setup 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, this helps you create less brittle code for your applications. Storing and retrieving instance metadata is a very common Compute Engine action. I recommend storing these startup and shutdown scripts in Cloud Storage as you will explore in the upcoming lab of this module.



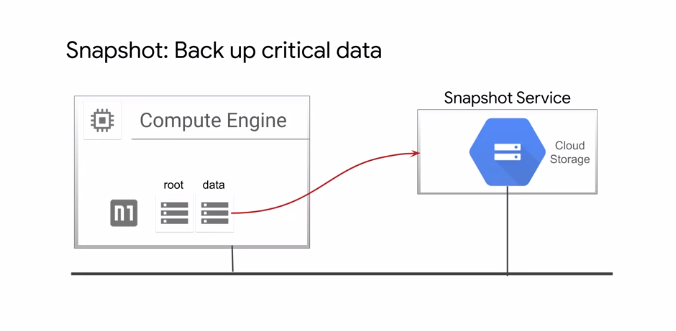
Another common action is to move an instance to a new zone. For example, you might do so for geographical reasons or because a zone is being deprecated.



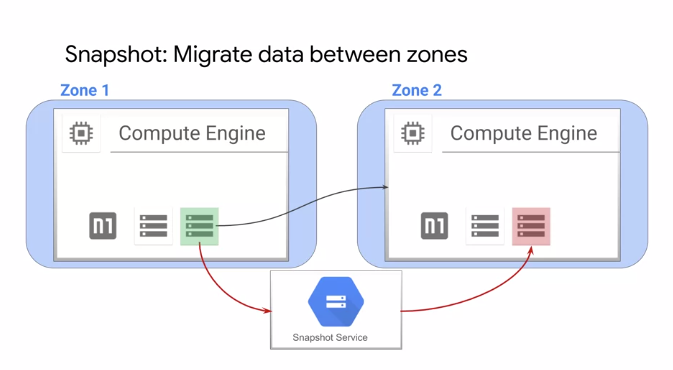
If you move your instance within the same region, you can automate the move by using the gcloud compute instances move command. If we move your instance to a different region, you need to manually do so by following the process outlined here.

This involves

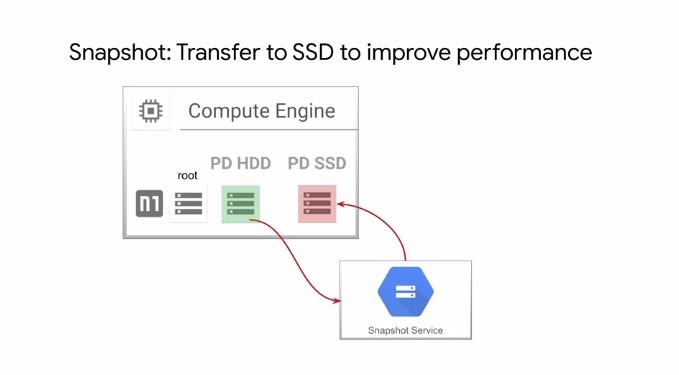
* making a snapshot of all persistent disks and
* creating new disks in the destination zone from that snapshot.
* Next, you create a new VM in the destination zone and attach the new persistent disks,
* assign a static IP, and update any references to the VM.
* Finally, you delete the original VM, its disks and the snapshot.



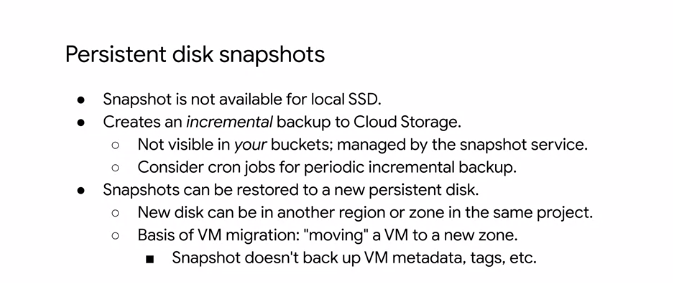
Speaking of snapshots, let's take a closer look at these. Snapshots have many use cases. For example, they can be used to backup critical data into a durable storage solution to meet application, availability, and recovery requirements. These snapshots are stored in Cloud Storage, which is covered later.



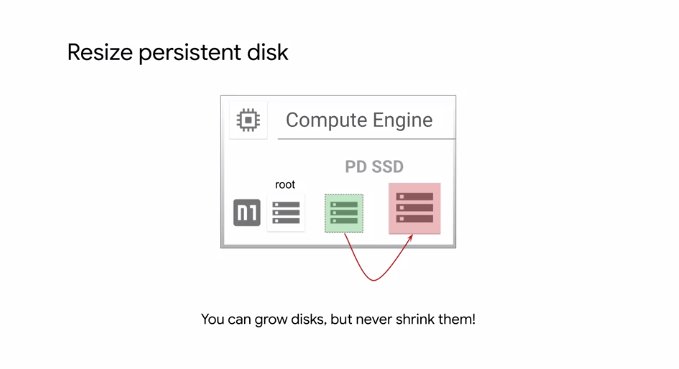
Snapshots can also be used to micro data between zones. I just discussed this when going over the manual process of moving an instance between two regions, but this can also be used to simply transfer data from one zone to another. For example, you might want to minimize latency by migrating data to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



Which brings me to another snapshot use case of transferring data to a different disk type. For example, if you want to improve disk performance, you could use a snapshot to transfer data from a standard ECD persistent disk to a SSD persistent disk.



Now that I've covered some of these snapshot use cases, let's explore the concept of a disk snapshot. First of all, this slide is titled persistent disk snapshots because snapshots are available only to persistent disks and not to local SSDs. Snapshots are different from public images and custom images which are used primarily to create instances or configure instance templates, in that snapshots are useful for periodic backup of the data on your persistent disks. Snapshots are incremental and automatically compressed, so you can create regular snapshots on a persistent disk faster and at a much lower cost than if you regularly created a full image of the disk. As we saw with the previous examples, snapshots can be restored to a new persistent disk, allowing for a move to a new zone. To create a persistent disk snapshot, see the link section of this video.



Another common Compute Engine action is to resize your persistent disk. The added benefit of increasing storage capacity is to improve I/O performance. This can be achieved while the disk is attached to a running VM without having to create a snapshot. Now, while you can grow disk and size, you can never shrink them. So keep this in mind.