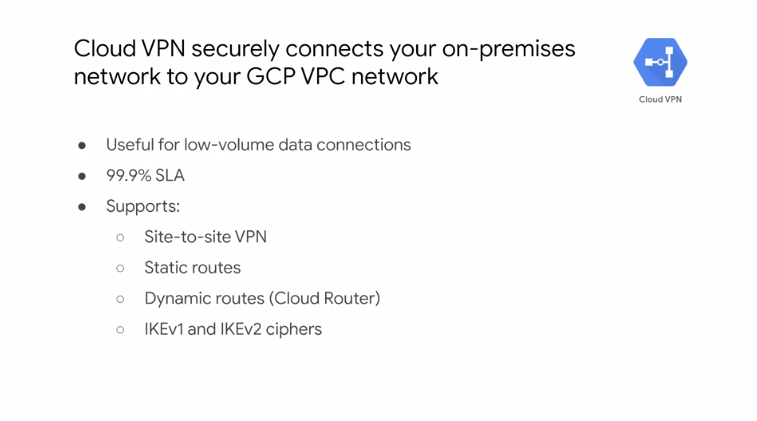
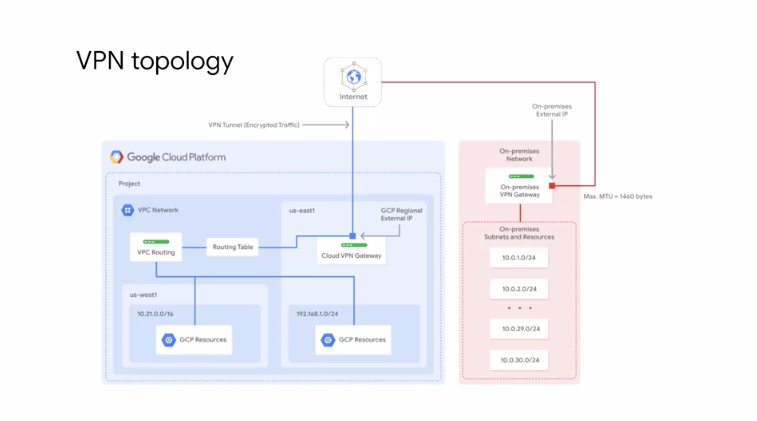
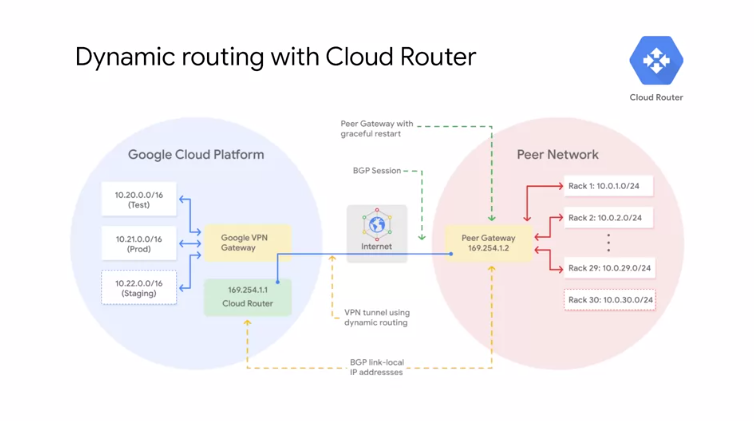
**Cloud VPN**

****

Cloud VPN securely connects your on-premise network to your GCP VPC network through an IPSec VPN tunnel. Traffic traveling between the two networks is encrypted by one VPN gateway. Then decrypted by the other VPN gateway. This protects your data as it travels over the public internet. That's why Cloud VPN is useful for low volume data connections. As a managed service Cloud VPN provides an SLA of 99.9 percent service availability and supports site to site VPN static and dynamic routes, and IKEv1 and IKEv2 ciphers. Cloud VPN doesn't support new cases where a client computers need to dial in to a VPN using client VPN software. Also, dynamic routes are configured with Cloud Router which we will cover briefly for more information about the SLA and these features see the links section of this video.

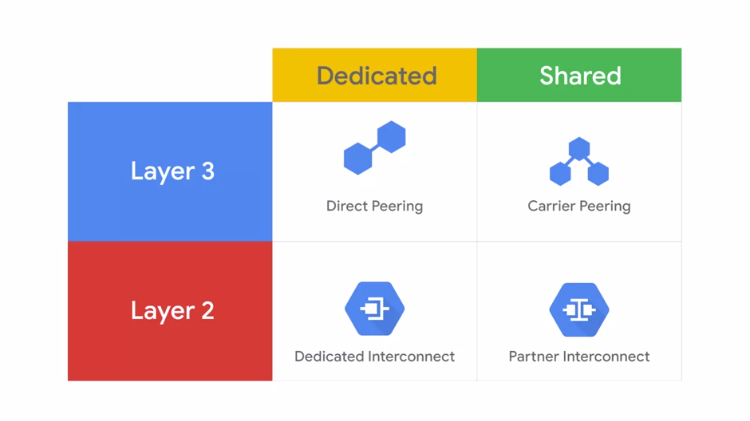


Let me walk through an example of Cloud VPN. This diagram shows a simple VPN connection between your VPC and on-premise network. Your VPC network has subnets in US-east one and US-west one. With GCP resources in each of those regions. These resources are able to communicate using their internal IP addresses because routing within a network is automatically configured, assuming that firewall rules allow the communication. Now, in order to connect to your on-premise network and its resources you need to configure your Cloud VPN gateway on-premise VPN gateway and to VPN tunnels. The Cloud VPN gateway is a regional resource that uses a regional external IP address. Your on-premise VPN gateway can be a physical device in your data center or a physical or software based VPN offering in another Cloud providers network. This VPN gateway also has an external IP address. A VPN tunnel then connects your VPN gateways and serves as the virtual medium through which encrypted traffic is passed. In order to create a connection between two VPN gateways you must establish two VPN tunnels. Each tunnel defines the connection from the perspective of its gateway and traffic can only pass when the pair of tunnels established. Now, one thing to remember when using Cloud VPN is that the maximum transmission unit or MTU for your on-premises VPN gateway cannot be greater than 1,460 bytes. This is because of the encryption and encapsulation of packets. For more information about this empty you consideration see the links section of this video.



I mentioned earlier that Cloud VPN supports both static and dynamic routes. In order to use dynamic routes you need to configure Cloud Router. Cloud Router can manage routes from Cloud VPN tunnel using border gateway protocol or BGP. This routing method allows for routes to be updated and exchanged without changing the tunnel configuration. For example, this diagram shows two different regional subnets in a VPC network namely tests and prod. The on-premise network has 29 subnets and the two networks are connected through Cloud VPN tunnels. Now, how would you handle adding new subnets? For example, how would you add a new staging subnet in the GCP network and a new on-premise 10.0.30.0/24 subnet to handle growing traffic in your data center? To automatically propagate network configuration changes the VPN tunnel uses Cloud Router to establish a BGP session between the VPC and the on-premise VPN gateway which must support BGP. The new subnets are then seamlessly advertised between networks. This means that instances in the new subnets can start sending and receiving traffic immediately as you will explore in the upcoming lab. To set up BGP an additional IP address has to be assigned to each end of the VPN tunnel. These two IP addresses must be link-local IP addresses. Belonging to the IP address range 169.254.0.0/16. These addresses are not part of IP address space of either network and are used exclusively for establishing a BGP session.

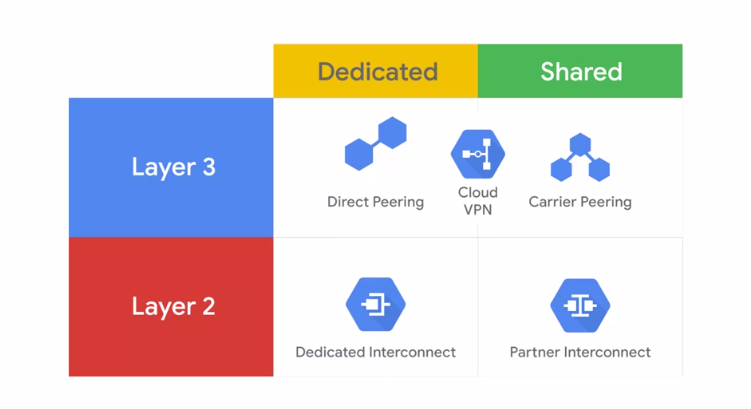
**Cloud Interconnect and Peering**



Next, let's talk about Cloud Interconnect and Peering services. There are different Cloud Interconnect and Peering services available to connect your infrastructure to Google's network. These services can be split into dedicated versus shared connections and layer two verses layer three connections. The services are Direct Peering, Carrier Peering, Dedicated Interconnect, and Partner Interconnect.

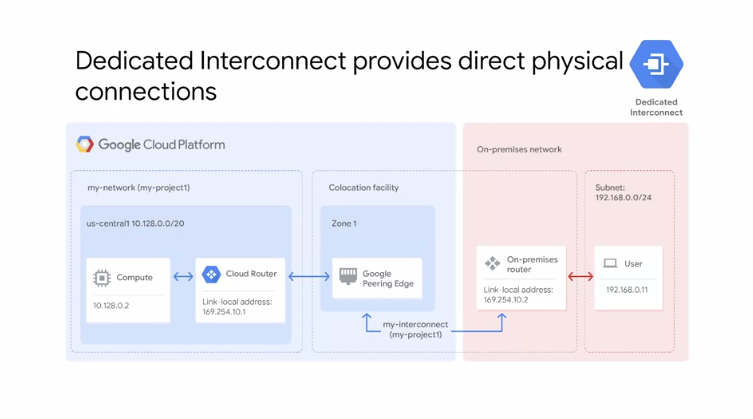
* Dedicated connections provide a direct connection to Google's network.
* But, shared connections provide a connection to Google's network through a partner.

Layer two connections use a VLAN that pipes directly into your GCP environment, providing connectivity to internal IP addresses in the RFC 1918 address space. Layer three connections provide access to G Suite services, YouTube and Google Cloud APIs using public IP addresses.



Now as I just explained earlier, Google also offers its own Virtual Private Network service called Cloud VPN. This service uses the public Internet but traffic is encrypted and provides access to internal IP addresses. That's why Cloud VPN is a useful addition to Direct Peering and Carrier Peering. Let me explain the Cloud Interconnect and Peering services separately first and then I'll provide some guidance on choosing the right connection.

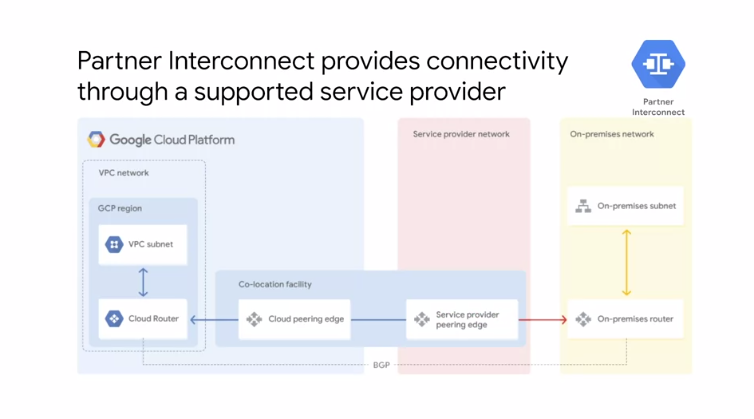
**Cloud Interconnect**

****

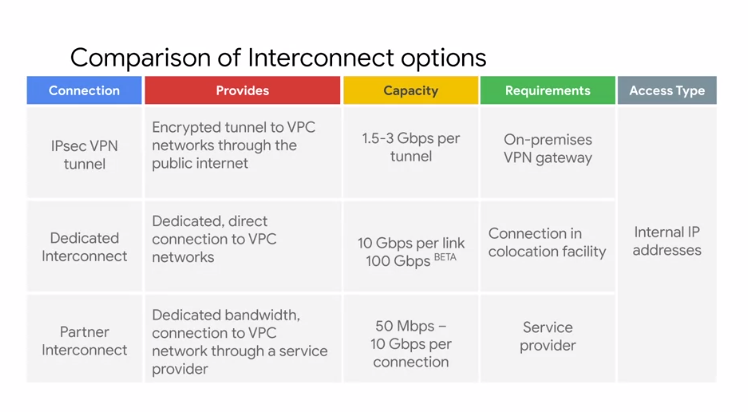
Dedicated Interconnect provides direct physical connections between your On-premise network and Google's network. This enables you to transfer a large amount of data between networks which can't be more cost-effective than purchasing additional bandwidth over the public Internet.



In order to use Dedicated Interconnect, you need to provision a cross-connect between the Google network and your own router in a common co-location facility, as shown in this diagram. To exchange routes between the networks, you configure a BGP session over the interconnect between the Cloud Router and the On-premise router. This will allow user traffic from the on-premise network to reach GCP resources on the VPC network and vice-versa. Dedicated Interconnect can be configured to offer a 99.9 percent or a 99.99 percent uptime SLA. See the Dedicated Interconnect documentation for details on how to achieve these SLAs. In order to use Dedicated Interconnect, your network must physically meet Google's network in a supported co-location facility. This map shows the location where you can create dedicated connections. For a full list of these locations, see the link section of this video. Now you might look at this map and say well I am nowhere near one of these locations. That's when you want to consider a Partner Interconnect.



Partner Interconnect provides connectivity between your on-premise network and your VPC network through a supported service provider. This is useful if your data center is in the physical location that cannot reach a Dedicated Interconnect co-location facility or if your data needs don't warrant a Dedicated Interconnect. In order to use Partner Interconnect, you work with the supported service provider to connect your VPC and on-premise networks. For a full list of providers, see the link section of this video. These service providers have existing physical connections to Google's network that they make available for their customers to use. After you establish connectivity with the service provider, you can request a Partner Interconnect connection from your service provider then establish a BGP session between your Cloud Router and On-premise Router to start passing traffic between your networks via the service providers network. Partner Interconnect can be configured to offer a 99.9 percent or 99.99 percent uptime SLA between Google and the service provider. See the Partner Interconnect documentation for details on how to achieve these SLAs.

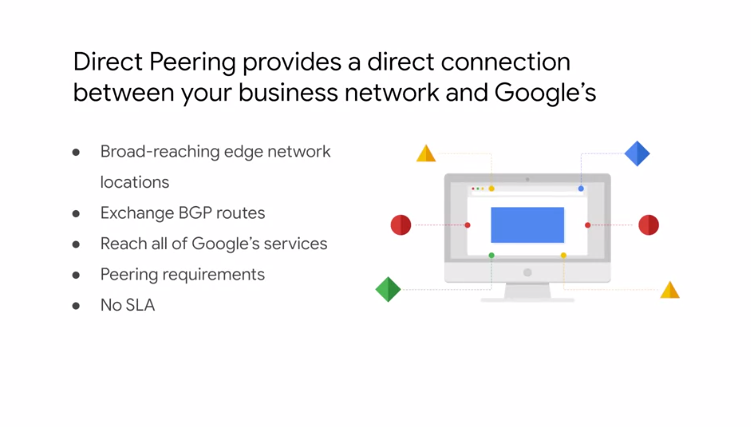


Let me compare the Interconnect options that we just discussed. All of these options provide internal IP address access between resources in your On-premise network and in your VPC network. The main differences are the connection capacity and the requirements for using a service.

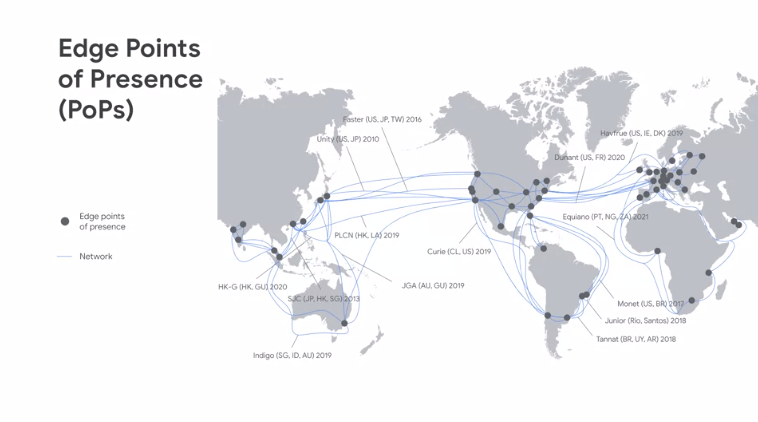
* The IPSec VPN tunnels that Cloud VPN offers have a capacity of 1.5 to 3 Gbps per tunnel and require VPN device on your On-premise network. The 1.5 Gbps capacity applies to the traffic that traverses the public Internet and the three Gbps capacity applies to the traffic that is traversing a Direct Peering link. You can configure multiple tunnels if you want to scale this capacity.
* Dedicated Interconnect has a capacity of 10 Gbps per link and requires you to have a connection in a Google supported co-location facility. You can have up to eight links to achieve multiples of 10 Gbps by 10 Gbps is the minimum capacity. As of this recording, there is a Beta feature that provides 100 Gbps per link with a maximum of two links. Keep in mind that features that are in beta are not covered by any SLA or deprecation policies and might be subject to backward-incompatible changes.
* Partner Interconnect has a capacity of 50 Mbps to 10 Gbps per connection and requirements depend on the service provider.

My recommendation is to start with VPN tunnels. When you need enterprise-grade connection to GCP, switch to Dedicated Interconnect or Partner Interconnect depending on your proximity to a co-location facility and your capacity requirements.

**Peering**

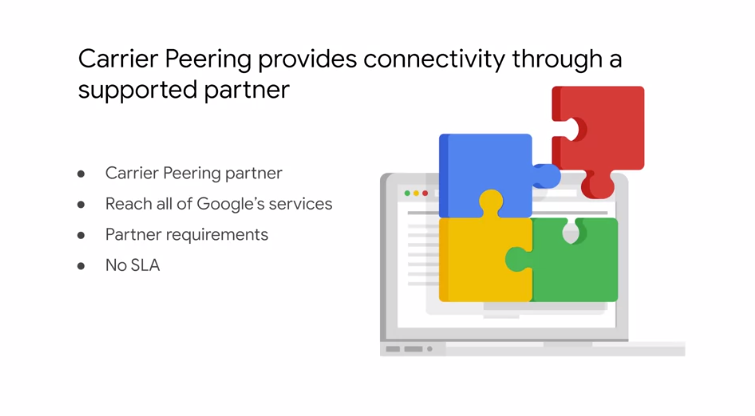
****

Let's talk about the Cloud Peering services which are Direct Peering and Carrier Peering. These services are useful when you require access to Google and Google Cloud properties. Google allows you to establish a direct peering connection between your business network and Google's. With this connection, you will be able to exchange internet traffic between your network and Google's at one of the Google's broad-reaching Edge network locations. Direct Peering with Google is done by exchanging BGP routes between Google and the peering entity. After a direct peering connection is in place, you can use it to reach all the Google's services including the full suite of Google Cloud Platform products. Unlike dedicated interconnect, direct peering does not have an SLA. In order to use direct peering, you need to satisfy the peering requirements in the links section of this video.

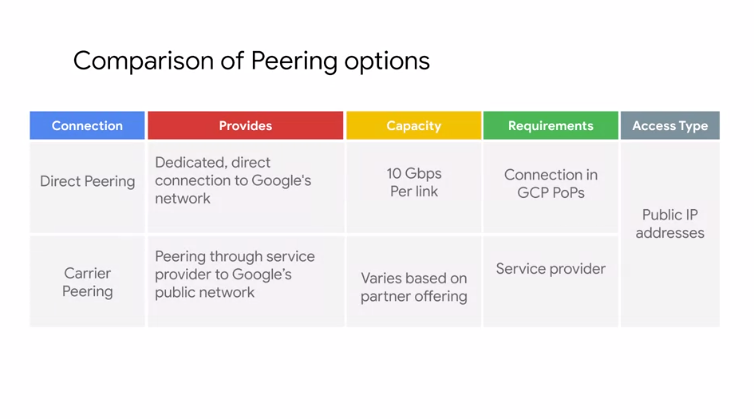


GCP's Edge points of presence or PoPs are where Google's network connects to the rest of the internet via peering. PoPs are present on over 90 internet exchanges and at over 100 interconnection facilities around the world. For more information about these exchange points and facilities, I recommend looking at Google's PeeringDB entries, which are linked below this video.

If you look at this map and say, "hey, I am nowhere near one of these locations," you will want to consider Carrier Peering.



If you require access to Google public infrastructure and cannot satisfy Google's peering requirements, you can connect via a carrier peering partner. Work directly with your service provider to get the connection you need and to understand the partners requirements. For a full list of available service providers, see the links section of this video. Now, just like direct peering, carrier peering also does not have an SLA.

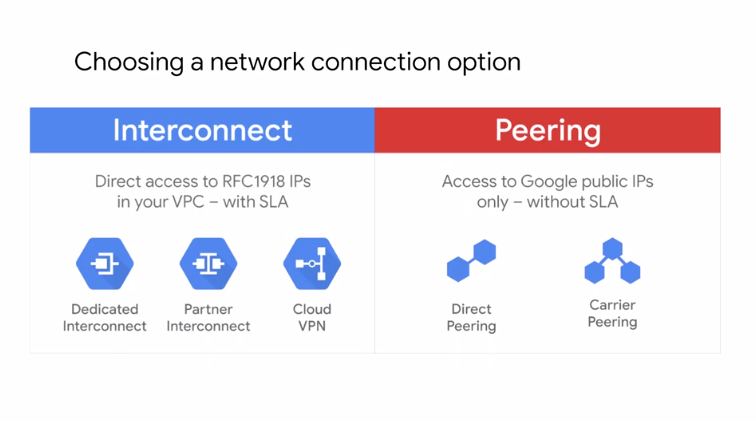


Let me compare the peering options that we just discussed. All of these options provide public IP address access to all of Google's services. The main differences are capacity and the requirements for using a service. Direct peering has a capacity of 10Gbps per link and requires you to have a connection in a GCP Edge point of presence. Carrier peering's capacity and requirements vary depending on the service provider that you work with.

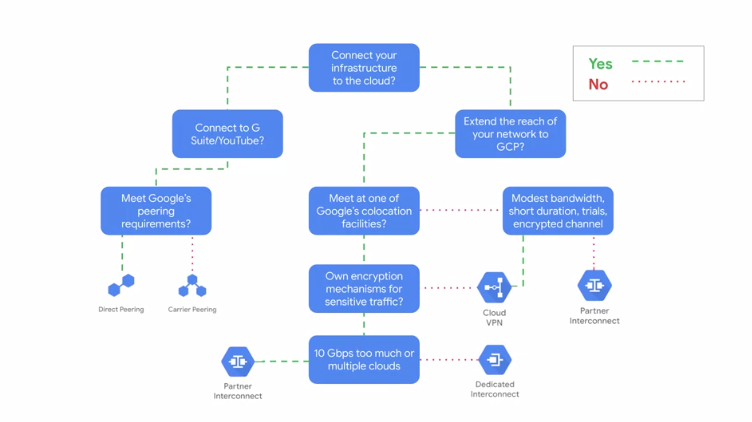
**Choosing a connection**



Now that we have discussed all the different connections services, let me help you determine which service best meets your hybrid connectivity needs. I started this lesson by introducing the five different ways to connect your infrastructure to GCP. I split these sources into dedicated versus shared connections and Layer 2 versus Layer 3 connections.



Another way to organize these sources is by interconnect services and by peering services. Interconnect services provide direct access to RFC1918 IP addresses in your VPC with an SLA. Peering services in contrast offer access to Google public IP addresses only without an SLA. Another way to choose the right service that meets your needs is with a flow diagram.



Let me walk you through this diagram from the top using the assumptions that you want to extend your infrastructure to the Cloud.

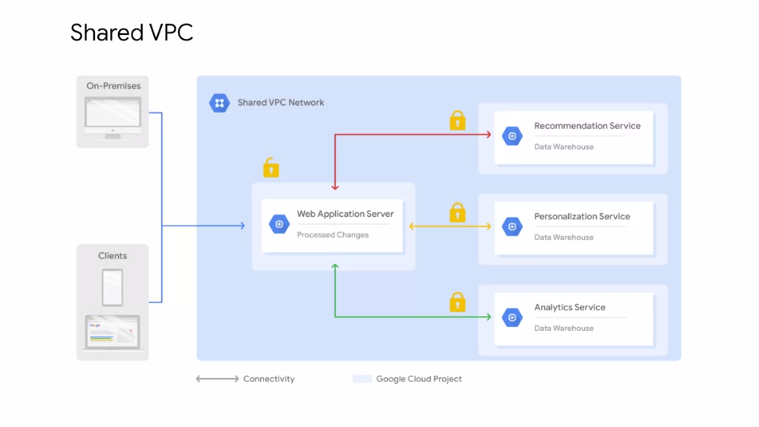
* Ask yourself whether you need to extend your network for G Suite services, YouTube or Google Cloud APIs.
* If you do, choose one of the peering services.
  + If you can meet Google's direct peering requirements, choose Direct Peering.
  + Otherwise, choose Carrier Peering.
* If you don't need to extend your network for G Suite services or Google Cloud APIs, but want to extend the reach of your network to GCP, you want to pick one of the interconnect services.
  + If you cannot meet Google at one of its co-location facilities, choose Cloud VPN or Partner Interconnect. This choice will depend on your bandwidth and encryption requirements along with the purpose of the connection.
  + Specifically, if you have modest bandwidth needs, will use the connection for short durations and trials, and require an encrypted channel, choose Cloud VPN. Otherwise, choose Partner Interconnect.
  + If you can meet Google at one of its co-location facilities you, might jump to Dedicated Interconnect. However, if you cannot provide your own encryption mechanism for sensitive traffic, feel that a 10 Gbps connections is too big or won't access to multiple clouds, you will want to consider Cloud VPN or Partner Interconnect instead.

**Shared VPC and VPC Networks**

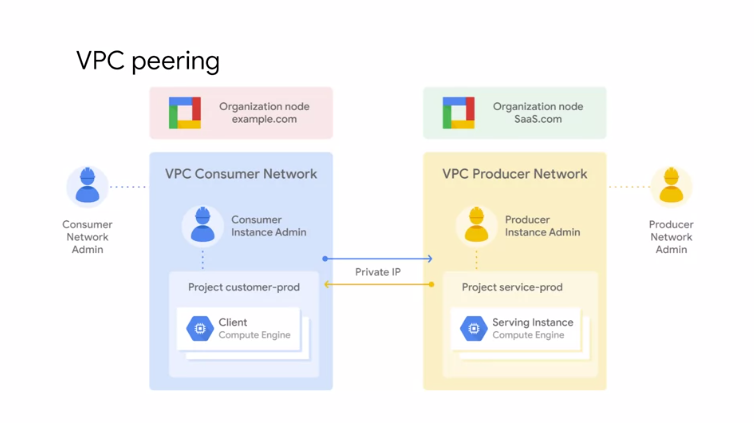
Let's move our attention from hybrid connectivity to shared VPC networks. In the simplest Cloud environment, a single project might have one VPC network, spanning many regions with VM instances hosting very large and complicated applications. However, many organizations commonly deploy multiple isolated projects with multiple VPC networks and sublets. In this lesson, we are going to cover two configurations for sharing VPC networks across GCP projects.

First, we will go over shared VPC which allows you to share a network across several projects in your GCP organization. Then, we will go over VPC Network Peering which allows you to configure private communication across projects in same or different organizations.

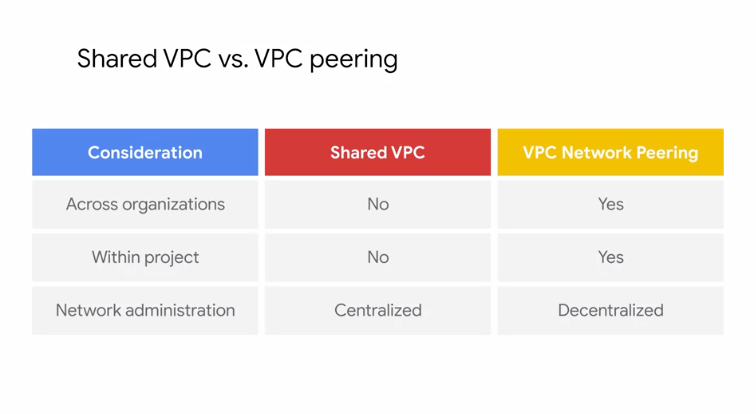
**Shared VPC** allows an organization to connect resources from multiple projects to a common VPC network. This allows the resources to communicate with each other securely, and efficiently using internal IPs from that network.



For example, in this diagram, there is one network that belongs to the web application servers project. This network is shared with three other projects. Namely, the recommendation service, the personalization service, and the analytics service. Each of those service projects has instances that are in the same network as the web application server, and allow for private communication to that server using the internal IP addresses. The web application server communicates with clients and on-premises, using the server's external IP address. The backend services in contrast cannot be reached externally because they only communicate using internal IP addresses. When you shared VPC, you designate a project as a **host project**, and attach one or more other service projects to it. In this case, the web application servers project is the host project. The three other projects are the service projects. The overall VPC network is called the **shared VPC network**.



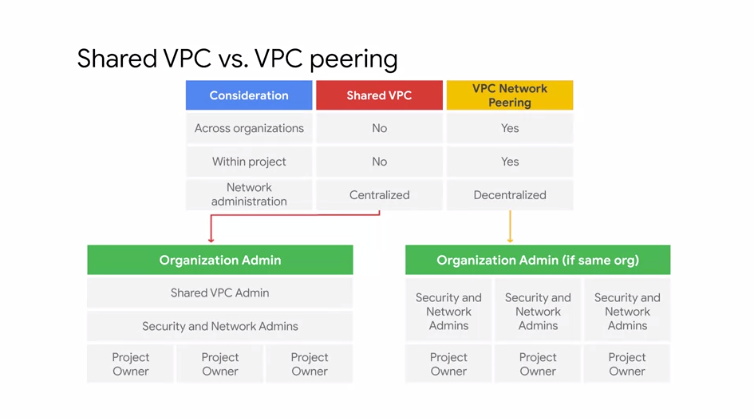
**VPC Network Peering** in contrast, allows private RFC 1918 connectivity across two VPC networks, regardless of whether they belong to the same project, or the same organization. Now, remember that each VPC network will have firewall rules that define what traffic is allowed or denied between the networks. For example, in this diagram, there are two organizations that represent a consumer and a producer respectively. Each organization has its own organization node. VPC network, virtual machine instances, network admin, and instance admin. In order for VPC Network Peering to be established successfully, the producer network admin needs to peer the producer network with the consumer network. The consumer network admin needs to peer the consumer network with the producer network. When both peering connections are created, the VPC Network Peering session becomes active and routes are exchanged. This allows the virtual machine instances to communicate privately using their internal IP addresses. VPC Network Peering is a decentralized or distributed approach to multiproject networking. Because each VPC network, may remain under the control of separate administrator groups, and maintains its own global firewall, and routing tables. Historically, such projects would consider external IP addresses or VPNs to facilitate private communication between VPC networks. However, VPC Network Peering does not incur the network latency, security, and cost drawbacks that are present when using external IP addresses or VPNs.



Now that we've talked about shared VPC, and VPC Network Peering, let me compare both of these configurations to help you decide which is appropriate for a given situation.

If you want to configure a private communication between VPC networks in different organizations, you have to use VPC Network Peering. Shared VPC only works within the same organization.

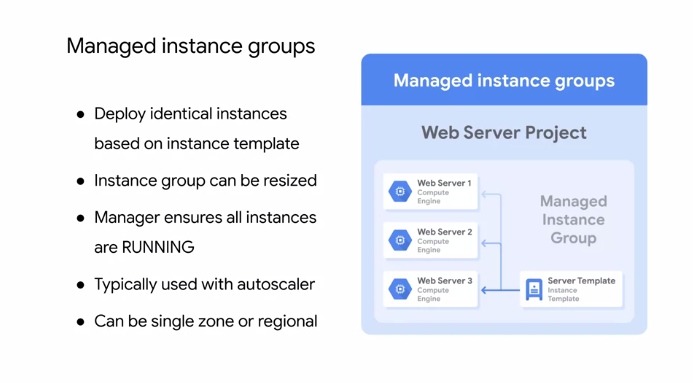
Somewhat similarly, if you want to configure private communication between VPC networks in the same project, you have to use VPC Network Peering. This doesn't mean that the networks need to be in the same project, but they can be. Shared. VPC, only works across projects.



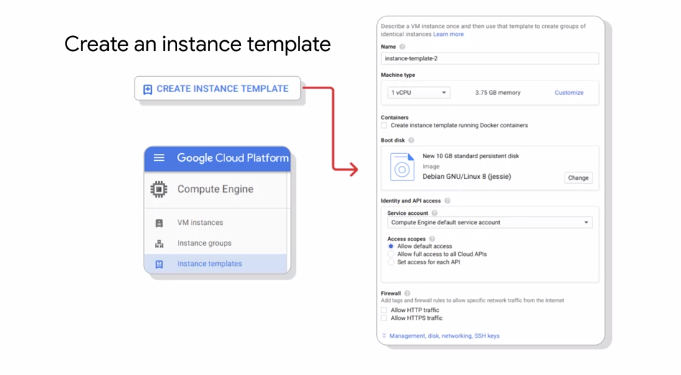
In my opinion, the biggest difference between the two configurations is the network administration models. Shared VPC is a centralized approach to multi-project networking because security and network policy occurs in a single designated VPC network.

In contrast, VPC Network Peering is a decentralized approach because each VPC network can remain under the control of separate administrator groups, and maintains its own global firewall, and routing tables. If you want to learn more about shared VPC and VPC Peering, I recommend the networking in Google Cloud Platform course, which you can find in the links section of this video.

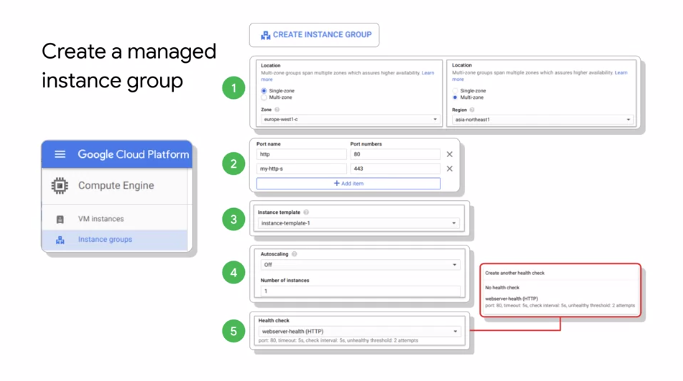
**Managed Instance Groups**

****

A managed instance group is a collection of identical virtual machine instances that you control as a single entity using an **instance template.** You can easily update all the instances in the group by specifying a new template in a rolling update. Also, when your application requires additional compute resources, managed instance groups can automatically scale the number of instances in the group. Managed instance groups can work with load balancing services to distribute network traffic to old instances in the group. If an instance in the groups stops, crashes, or is deleted by an action other than the instance groups command, the managed instance group automatically recreates the instance so it can resume its processing tasks. The recreated instance uses the same name and the same instance template as the previous instance. Managed instance groups can automatically identify and recreate unhealthy instances in a group to ensure that all the instances are running optimally. Regional managed instance groups are generally recommended over zonal managed instance groups because they allow you to spread the application load across multiple zones instead of confining your application to a single zone or you're having to manage multiple instance groups across different zones. This replication protects against zonal failures and unforeseen scenarios where an entire group of instances in a single zone malfunctions. If that happens, your application can continue serving traffic from instances running in another zone of the same region.

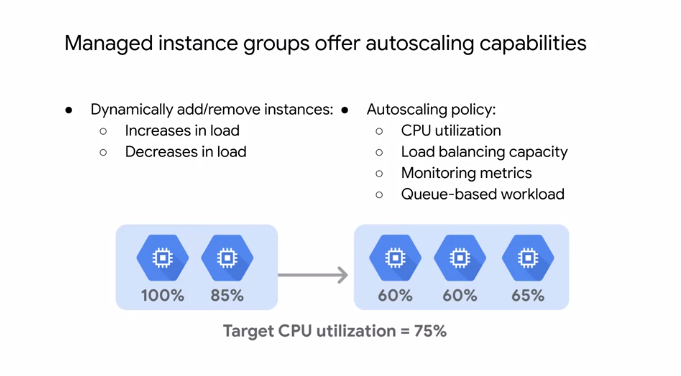


In order to create a managed instance group, you first need to create an **instance template**. Next, you are going to create a managed instance group of end specific instances. The instance group manager then automatically populates the instance group based on the instance template. You can easily create instance templates using the GCP console. The instance template dialogue looks and works exactly like creating an instance, except that the choices are recorded so that they can be repeated.



When you create an instance group, you define the specific rules for the instance group. First, decide whether the instance group is going to be single or multi zoned and where those locations will be. Second, choose the ports that you are going to allow and load balance across. Third, select the instance template that you want to use. Fourth, decide whether you want to auto-scale and under what circumstances. Finally, consider creating a health check to determine which instances are healthy and should receive traffic. Essentially, you're still creating virtual machines, but you're applying more rules to that instance group.

**Autoscaling and health checks**

****

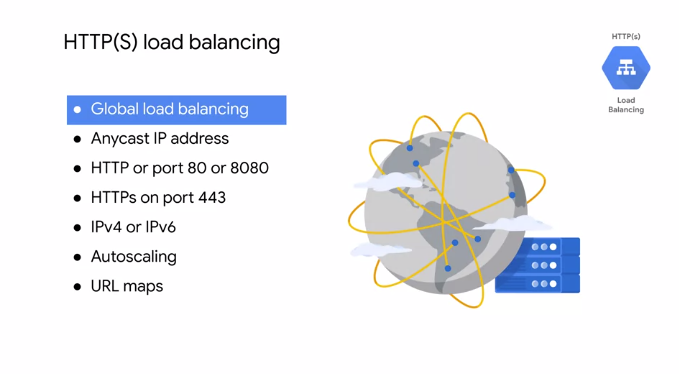
Let me provide more details on the autoscaling and health checks of the managed instance group. As I mentioned earlier, managed instance groups offer autoscaling capabilities that allow you to automatically add or remove instances from a managed instance group based on increase or decrease in load.

Autoscaling helps your applications gracefully handle increase in traffic and reduces cost when the need for resource is lower. You just define the autoscaling policy, and the autoscaler performs automatic scaling based on the measured load. Applicable autoscaling policies include scaling based on CPU utilization, load balancing capacity, or monitoring metrics, or by a queue-based workload like Cloud Pub/Sub.

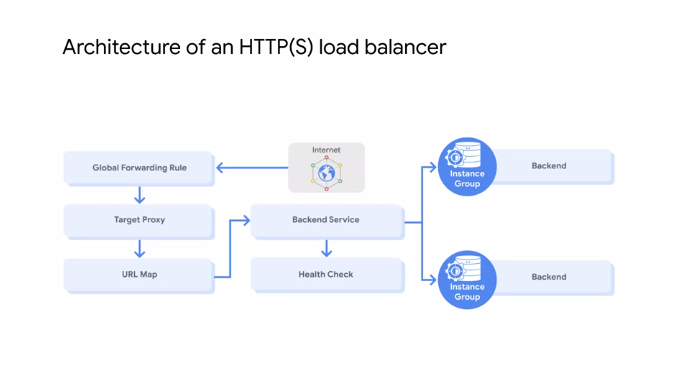
For example, let's assume you have two instances that are at 100 percent and 85 percent CPU utilization as shown on this slide. If your target CPU utilization is 75 percent, the autoscaler will add another instance to spread out the CPU load and stay below the 75 percent target CPU utilization. Similarly, if the overall load is much lower than the target, the autoscaler will remove instances as long as that keeps the overall utilization below the target.

Now, you might ask yourself, how do I monitor the utilization of my instance group? When you click on an instance group or even an individual virtual machine, a graph is presented. By default, you will see the CPU utilization over the past hour. But you can't change the timeframe and visualize other metrics like disk and network usage. These graphs are very useful for monitoring your instances, utilization, and for determining how best to configure your autoscaling policy to meet changing demands. If you monitor the utilization of your virtual machine instances and Stackdriver monitoring, you can even set up alerts through several notification channels. For more information on autoscaling, see the links section of this video. Another important configuration for a managed instance group and load balancer is a health check. A health check is very similar to an Uptime check in Stackdriver. You just define a protocol, port, and health criteria as shown in the screenshot. Based on this configuration, GCP computes a health state for each instance. The health criteria defines how often to check whether an instance is healthy. That's the check interval. How long to wait for a response? That's the timeout. How many successful attempts are decisive? That's the healthy threshold. How many failed attempts are decisive? That the unhealthy threshold. In the example on this slide, the health check would have to fill twice over a total of 15 seconds before an instance is considered unhealthy.

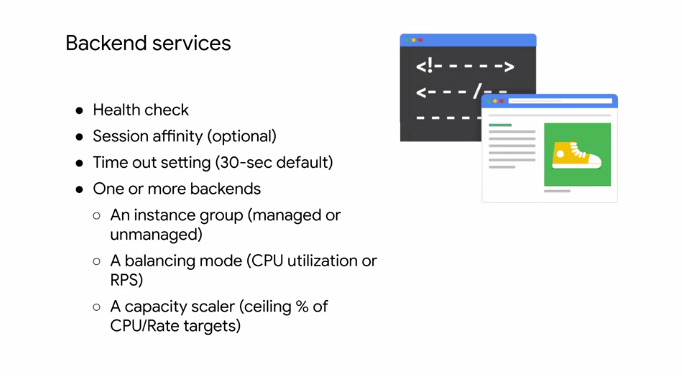
**Overview of HTTP(S) load balancing**

****

Now, let's talk about HTTPS Load Balancing which acts at layer seven of the OSI model. This is the application layer which deals with the actual content of each message allowing for routing decisions based on the URL. GCP HTTPS load balancing provides global load balancing for HTTPS requests destined for your instances. This means that your applications are available to your customers at a single any cast IP address, which simplifies your DNS setup. HTTPS load balancing balances HTTP and HTTPS traffic across multiple backend instances and across multiple regions. HTTP requests are load balanced on port 80 or 8080, and HTTPS requests are load balanced on port 443. This load balancers supports both IPv4 and IPv6 clients, is scalable, requires no pre-warming, and enables content-based and cross-regional load balancing. You can configure your own maps that route some URLs to one set of instances and route other URLs to other instances. Requests are generally routed to the instance group that is closest to the user. If the closest instance group does not have sufficient capacity, the request is sent to the next closest instance group that does have the capacity. You will get to explore most of these benefits in the first lab of the module.

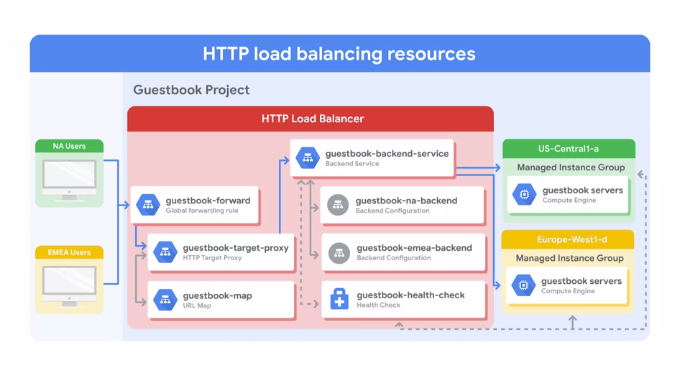


Let me walk through the complete architecture of an HTTPS load balancer by using this diagram. A Global Forwarding Rule direct incoming requests from the Internet to a target HTTP proxy. The target HTTP proxy checks each request against a URL map to determine the appropriate backend service for the request. For example, you can send requests for www.example.com slash audio to one backend service, which contains instances configured to deliver audio files, and the request for www.example.com slash video to another backend service which contains instances configured to deliver video files.



The backend service directs each request to an appropriate backend based on solving capacity zone and instance held of its attached backends. The backend services contain a health check, session affinity, a timeout setting, and one or more backends. A health check pulls instances attached to the backend service at configured intervals. Instances that pass the health check are allowed to receive new requests. Unhealthy instances are not sent requests until they are healthy again. Normally, HTTPS load balancing uses a round robin algorithm to distribute requests among available instances. This can be overridden with **session affinity**. Session affinity attempts to send all requests from the same client to the same Virtual Machine Instance. Backend services also have a timeout setting, which is set to 30 seconds by default. This is the amount of time the backend service will wait on the backend before considering the request a failure. This is a fixed timeout not an idle timeout. If you require longer lived connections, set this value appropriately. The backends themselves contain an instance group, a balancing mode, and a capacity scalar. An instance group contains Virtual Machine Instances. The instance group may be a managed instance group with or without autoscaling or an unmanaged instance group. A balancing mode tells the load balancing system how to determine when the backend is at full usage. If older backends for the backend service in a region are at the full usage, new requests are automatically routed to the nearest region that can still handle requests. The balancing mode can be based on CPU utilization or requests per second. A capacity setting is an additional control that interacts with the balancing mode setting. For example, if you normally want your instances to operate at a maximum of 80 percent CPU utilization, you would set your balancing mode to 80 percent CPU utilization and your capacity to 100 percent. If you want to cut instance utilization in half, you could leave the balancing mode at 80 percent CPU utilization and set capacity to 50 percent. Now, any changes to your backend services are not instantaneous. So don't be surprised if it takes several minutes for your changes to propagate throughout the network.

**Example: HTTP load balancer**

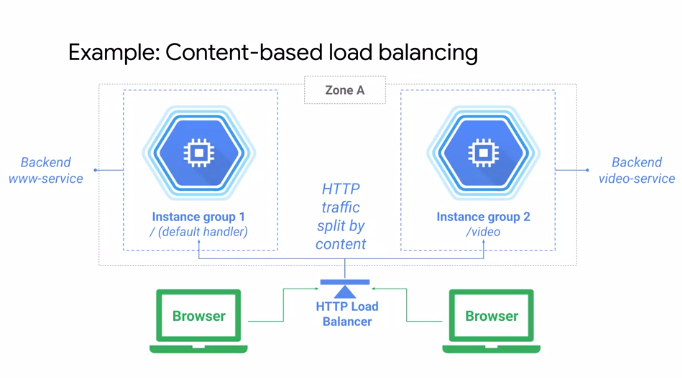
****

Let me work through an HTTP load balancer in action. The project on this slide has single global IP address, but users enter the Google Cloud network from two different locations, one in North America, and one in EMEA.

First, the global forwarding rule directs incoming requests to the target HTTP proxy. The proxy checks the URL map to determine the appropriate back-end service for the request. In this case, we're serving a guestbook application with only one back-end service.

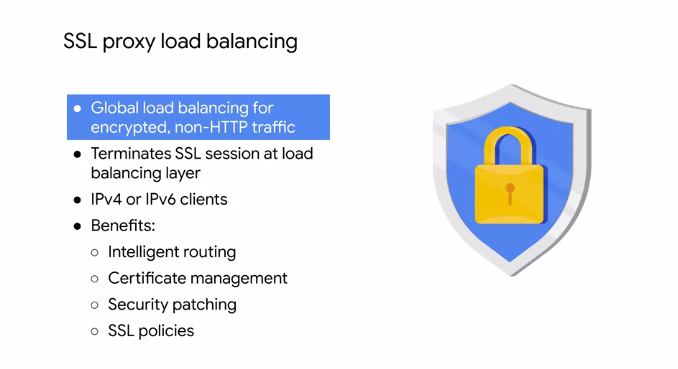
The back-end service has two back-ends, one in US Central 1-a, and one in Europe West 1-d. Each of those back-ends consist of a managed instance group. Now, when a user request comes in, the load balancing service determines the approximate origin of the request from the source IP address. The load balancing service also knows the locations of the instances owned by the back-end service, their overall capacity and their overall current usage. Therefore, if the instances closest to the user has available capacity, the request is forwarded to that closest set of instances.

In our example, traffic from the user in North America, would be forwarded to the managed instance group in US Central 1-a, and the traffic from the user in EMEA would be forwarded to the managed instance group in Europe West 1-d. If there are several users in each region, the incoming requests to the given region are distributed evenly across all available back-end services, and instances in that region. If there are no healthy instances, with available capacity in a given region, the load balancer instead sends the request to the next closest region with available capacity. Therefore, traffic from the EMEA user, could be forwarded to the US Central 1-a back-end, if the Europe West 1-d back-end does not have capacity, or has no healthy instances as determined by the health checker. This is referred to as cross-region load balancing.

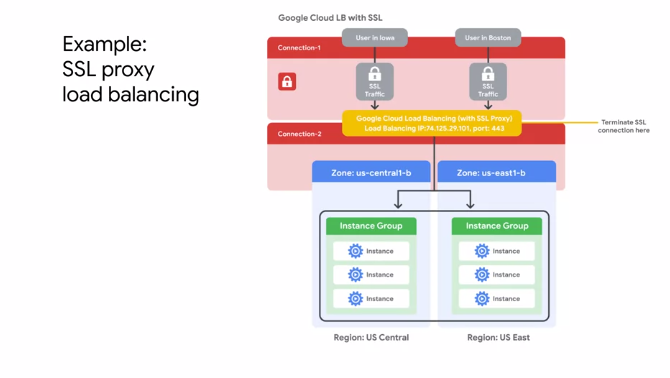


Another example, of an HTTPS load balancer, is a content-based load balancer. In this case there are two separate back-end services that handle either web or video traffic. The traffic is split by the load balancer based on the URL header as specified in the URL map, if the user's navigating to slash video, the traffic is sent to the back-end video service and if the user is navigating anywhere else, the traffic is sent to the web service back-end. All of that is achieved with a single global IP address.

**SSL proxy load balancing**

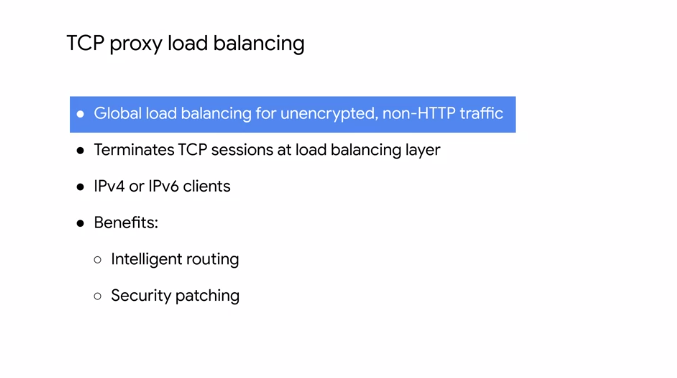
****

Let's talk about SSL proxy and TCP proxy load balancing. SSL proxy is a global load balancing service for encrypted, non-HTTP traffic. This load balancer terminates user SSL connections at the load balancing layer, then balances the connections across your instances using the SSL or TCP protocols. These instances can be in multiple regions, and the load balancer automatically directs traffic to the closest region that has capacity. SSL proxy load balancing supports both IPv4 and IPv6 addresses for client traffic and provides intelligent routing, certificate management, security patching, and SSL policies. Intelligent routing means that this load balancer can route requests to backend locations where there is capacity. From a certificate management perspective, you only need to update your customer-facing certificate in one place when you need to switch those certificates. Also, you can reduce the management overhead for your virtual machine instances by using self-signed certificates on your instances. In addition, if vulnerabilities arise in the SSL or TCP stack, GCP will apply patches at the load balancer automatically in order to keep your instances safe. For the full list of ports supported by SSL proxy load balancing and other benefits, please refer to the link section of this video.

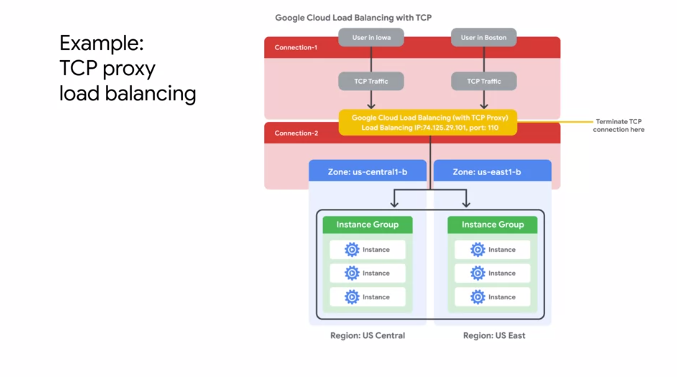


This network diagram illustrates SSL proxy load balancing. In this example, traffic from users in Iowa and Boston is terminated at the global load balancing layer. From there, a separate connection established to the closest backend instance. In other words, the user in Boston would reach the US East region, and the user in Iowa would reach the US Central region, if there's enough capacity. Now, the traffic between the proxy and the backend can use SSL or TCP. I recommend using SSL.

**TCP proxy load balancing**

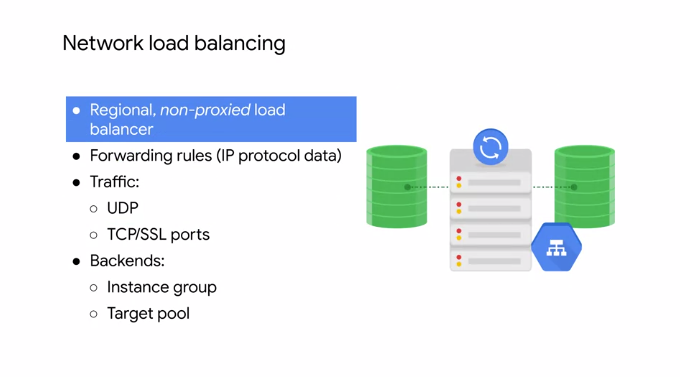
****

TCP proxy is a global load balancing service for unencrypted non-HTTP traffic. This load balancer terminates your customers TCP sessions at the load balancing layer, then forwards the traffic to your virtual machine instances using TCP or SSO. These instances can be in multiple regions and the load balancer automatically directs traffic to the closest region that has capacity. TCP proxy load balancing supports both IPv4 and IPv6 addresses for Client Traffic. Similar to SSL proxy load balancer, the TCP proxy load balancer provides intelligent routing and security patching. For the fullest reports supported by TCP proxy load balancing and other benefits, please refer to the links section of this video. This network diagram illustrates TCP proxy load balancing.

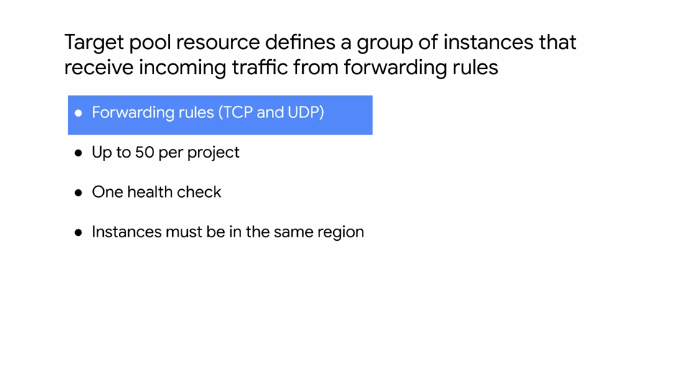


In this example, traffic from users in Iowa and Boston is terminated at the Global Load Balancing layer. From there, a separate connection is established to the closest backend instance. As in the SSL proxy load balancing example, the users in Boston would reach the US East region and the user in Iowa which reach the US central region, if there's enough capacity. Now the traffic between the proxy and the backends can use SSL or TCP and I also recommend using SSL here.

**Network load balancing**

****

Next let's talk about network load balancing which is a regional load balancing service. Network load balancing is a regional non proxied load balancing service. In other words, all traffic is passed through the load balancer instead of being proxied and traffic can only be balanced between virtual machine instances that are in the same region unlike a global load balancer. This load balancing service uses forwarding rules to balance the load on your systems based on incoming IP protocol data such as address, port, and protocol type. You can use it to load balance UDP traffic and to load balance TCP NSSL traffic on ports that are not supported by the TCP proxy and SSL proxy load balancers. The back ends of a network load balancer can be a template-based instance group or target pooled resource. But what is the target pool resource?

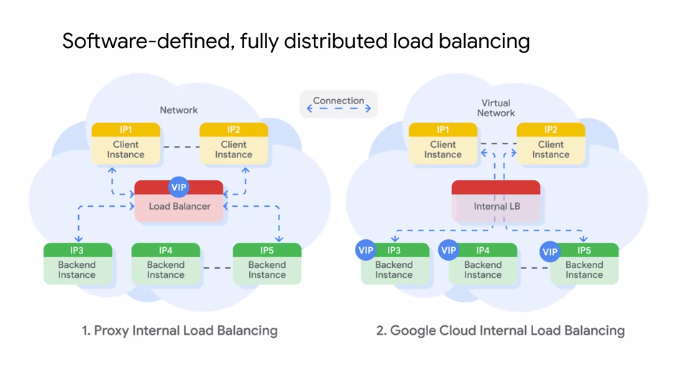


A target pool resource defines a group of instances that receive incoming traffic from forwarding rules. When a forwarding rule direct traffic to a target pool, the load balancer picks an instance from these target pools based on hash of the source IP and port, and the destination IP and port. These target pools can only be used with forwarding rules that handled TCP and UDP traffic. Now each project can have up to 50 target pools and each target pool can have only one health check. Also, all the instances of a target pool must be in the same region which is the same limitation as for the network load balancer.

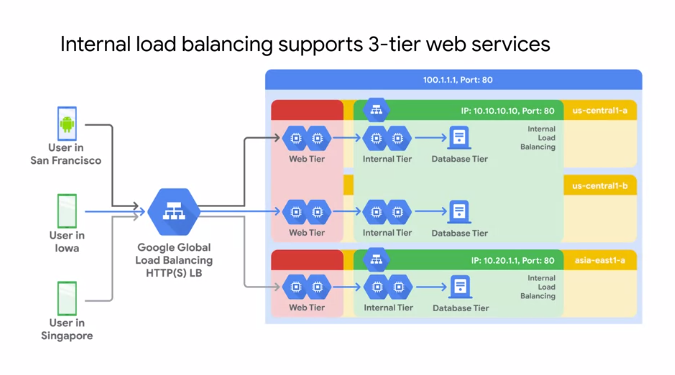
**Internal load balancing**

****

Next, let's talk about internal load balancing. Internal load balancing is a regional, private load balancing service for TCP and UDP based traffic. In other words, this load balancer enables you to run and scale your services behind a private load balancing IP address. This means that it is only accessible through the internal IP address of virtual machine instances that are in the same region. Therefore, use internal load balancing to configure an internal load balancing IP address, to act as the front end to your private backend instances. Because you don't need a public IP address for your load balanced service, your internal client requests to stay internal to your VPC network and the region. This often results in lower latency, because all your load balanced traffic will stay within Google's network, making your configuration much simpler. Let's talk more about the benefit of using a software-defined internal load balancing service.

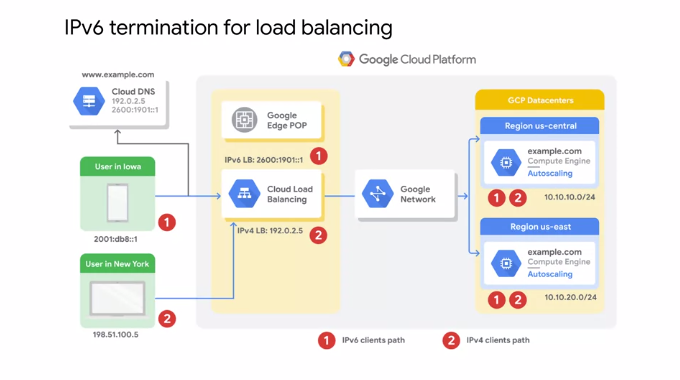


GCP internal load balancing is not based on a device or a virtual machine instance. Instead, it is a software-defined, fully distributed load balancing solution. In the traditional proxy model of internal load balancing as shown on the left, you configure an internal IP address on a load balancing device or instances, and your client instance connects to this IP address. Traffic coming to the IP address is terminated at the load balancer, and the load balancer selects a backend to establish a new connection to. Essentially, there are two connections. One between the client and the load balancer, and the one between the load balancer and the backend. GCP internal load balancing distributes client instance requests to the backend using a different approach, as shown on the right. It uses lightweight load balancing built on top of Andromeda, Google's network virtualization stack, to provide software-defined load balancing that directly delivers the traffic from the client instance to a backend instance. To learn more about Andromeda, I recommend the blog that is in the links section of this video.

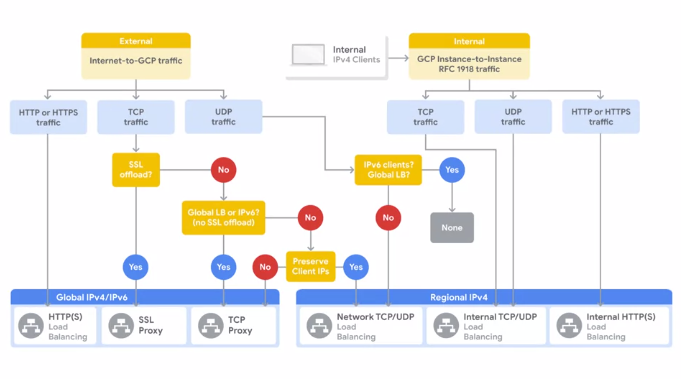


Now, internal load balancing enables you to support use-cases such as the traditional treaty or web service. In this example, the web tier uses an external HTTPS load balancer, that provides a single global IP address for users in San Francisco, Iowa and Singapore, and so on. The backends of this load balancer are located in the US-Central1 and Asia-East-1 region, because this is a global load balancer. These backends then access an internal load balancer in each region as the application or internal tier. The backends of this internal tier are located in US-Central1-A, US-Central1-B, and Asia-East1-B. The last tier is the database tier in each of these zones. The benefit of this three-tier approach is that neither the database tier nor the application tier is exposed externally. The simplified security and network pricing.

**Choosing a load balancer**

****

Now, that we have discussed all the different load balancing services within GCP, let me help you determine which load balancer best meets your need. One differentiator between the different GCP load balancers is the support for IPv6 clients. Only the HTTPS, SSL proxy, and TCP proxy load balancing services support IPV6 clients. IPv6 termination for these load balancers enables you to handle IPv6 requests from your users and proxy them over IPv4 to your backend. For example, in this diagram, there is a website, www.example.com, that is translated by Cloud DNS to both an IPv4 and IPv6 address. This allows a desktop user in New York and a mobile user in Iowa to access the load balancer through the IPv4 and IPv6 addresses respectively. But how does the traffic get to the backends and their IPv4 addresses? Well, the load balancer acts as a reverse proxy, terminates the IPv6 client connection and places the request into an IPv4 connection to a backend. On the reverse path, the load balancer receives the IPv4 response from the backend and places it into the IPv6 connection back to the original client. In other words, configuring IPv6 termination for your load balancers, lets your backend instances appear as IPv6 applications to your IPv6 clients.

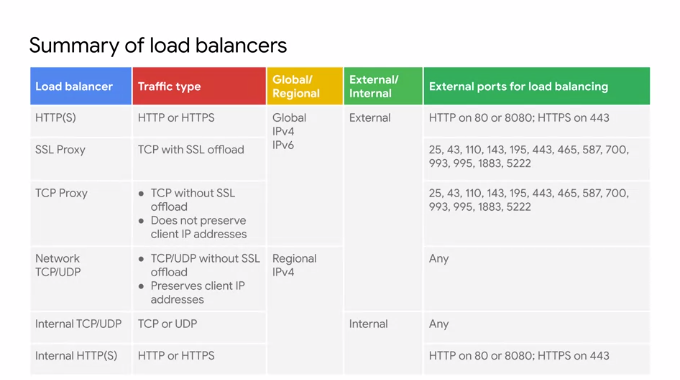


Now, in order to decide which load balancer best suits your implementation of GCP, consider the following aspects of Cloud load balancing. Global versus regional load balancing, external versus internal load balancing, and the traffic type. If you need an external load balancing service, start on the top left of this flowchart.

First, choose the type of traffic that your load balancer must handle.

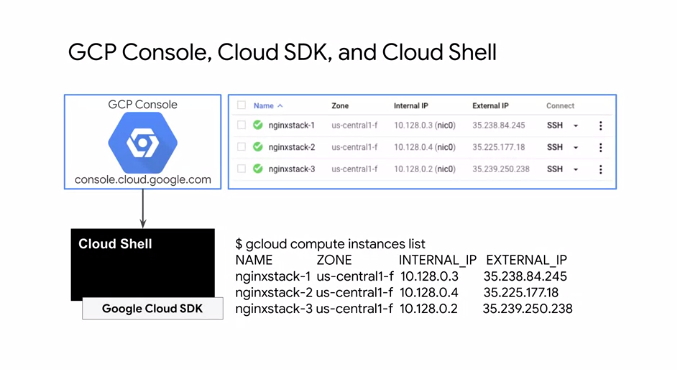
* If that is HTTP or HTTPS traffic, I recommend using the HTTPS load balancing service as a layer seven load balancer.
* Otherwise, use the TCP and UDP traffic paths of this flowchart to determine whether the SSL proxy, TCP proxy, or network load balancing service meets your needs.
* If you need an internal load balancing service, you have the internal load balancing service available and it supports both TCP and UDP traffic.

As I mentioned at the beginning of this module, there's actually another internal load balancer for HTTPS traffic but it's in beta as of this recording. The sixth load balancer is for HTTP or HTTPS traffic and it's regional meaning for IPv4 clients.

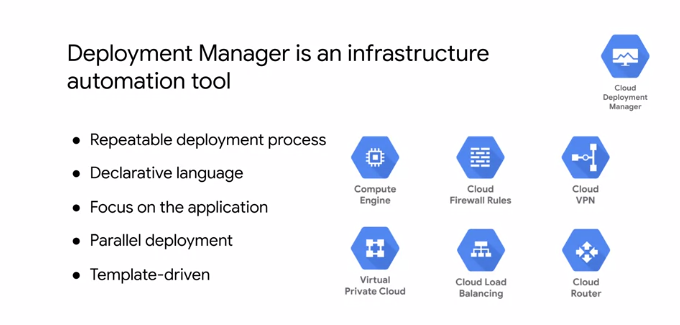


If you prefer a table over a flowchart, I recommend this summary table. This table helps you identify the right load balancer based on the traffic type, the distribution of your backend global or regional, and the type of IP addresses of your backend external or internal. This table also lists the available ports for load balancing and highlights that only the Global Load Balancers support both IPv4 and IPV6 clients.

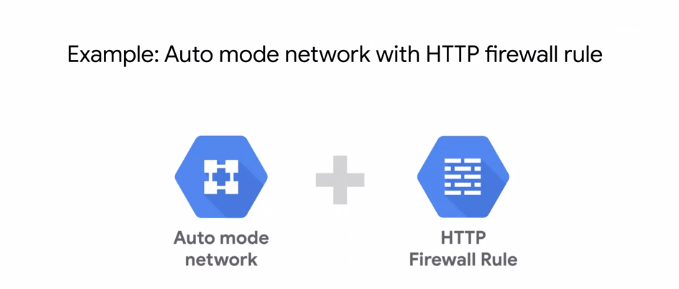
**Deployment Manager**

****

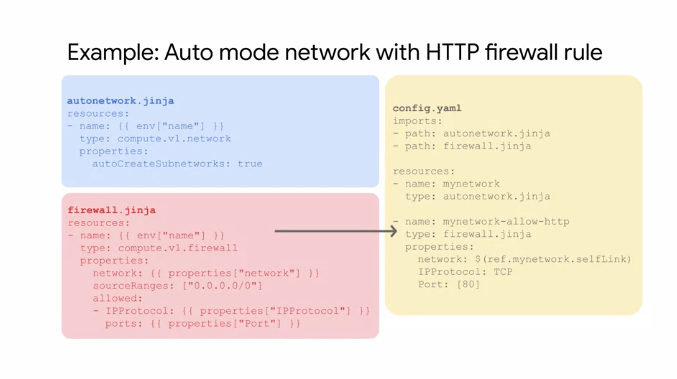
So far you've been creating GCP resources using the GCP Console and Cloud Shell. I recommend the GCP Console when you are new to using a service or if you prefer a UI. Cloud Shell works best when you are comfortable using a specific service and you want to quickly create resources using the command line. Deployment Manager takes this one step further.



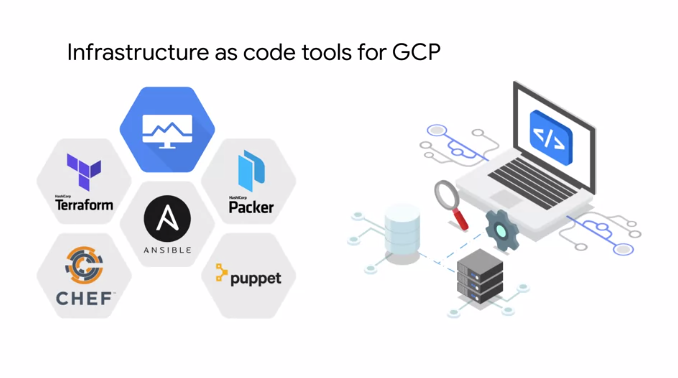
Deployment Manager is an infrastructure deployment service that automates the creation and management of GCP resources for you. You just specify all the resources needed for your application in a declarative format and deploy your configuration. This deployment can be repeated over and over with consistent results and you can delete a whole deployment with one command or click. The benefit of a declarative approach is that it allows you to specify what the configuration should be and let the system figure out the steps to take. Instead of deploying each resource separately, you specify the set of resources which compose the application or service, allowing you to focus on the application. Unlike Cloud Shell, Deployment Manager will deploy resources in parallel. You can even abstract parts of your configuration into individual building blocks or templates that can be used for other configurations. Deployment Manager uses the underlying APIs of each GCP service to deploy your resources. This enables you to deploy almost everything we have seen so far from instances, instance templates and groups to VPC networks, firewall rules, VPN tunnels, Cloud routers, and load balancers. For a full list of supported resource types, see the link section of this video.



Before you get into the lab, let me walk you through a quick example that shows how Deployment Manager can be used to set up an auto mode network with an HTTP firewall rule. I could put this whole deployment into one single configuration. However, it's useful to parameterize your configuration with templates. Specifically, we're going to create one template for the auto mode network and one for the firewall rule. Therefore, if we want to create either of these resources somewhere else later on, we can use those templates.

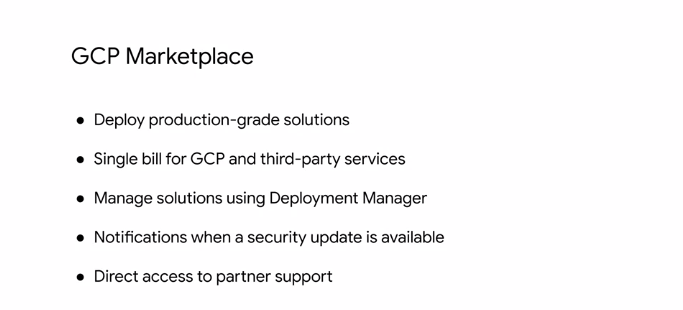


Let's start with the auto mode network template which we can write in Jinja2 or Python. Now, each resource must contain a name, type, and properties. For the name, I'm using an invariant variable to get the name from the top-level configuration which makes this template more flexible. For the type, I'm defining the API for a VPC network which is compute.v1.network. You can find all supported types in the documentation or query them within Cloud Shell as you will explore in the upcoming lab. By definition, an auto mode network automatically creates a subnetwork in each region. Therefore, I am setting the auto-create subnetworks property to true. Next, let's write the template for the HTTP firewall rule. For the name, I'm again using an invariant variable to get the name from the top-level configuration. For the type, I'm defining the API for a firewall rule which is compute.v1.firewall. The properties section contains the network I want to apply this firewall rule to the source IP ranges, and the protocols, and ports that are allowed. Except for the source IP ranges, I'm defining these properties as template properties. I will provide the exact properties from the top-level configuration, which makes this firewall rule extremely flexible. Essentially, I can use this firewall rule template for any network and any protocol and port combination. Next, let's write the top-level configuration in YAML syntax. I start by importing the templates that I want to use in this configuration, which are autonetwork.jinja and firewall.jinja. Then I define the auto mode network by giving it the name mynetwork and leveraging the auto network.jinja template. I could create more auto mode networks in this configuration with other names or simply reuse this template in other configurations later on. Now I define the firewall rule by giving it a name, leveraging the firewall.jinja template, referencing my network, and defining the IP protocol and port. I can easily add other ports such as 443 for HTTPS or 22 for SSH traffic. Using the self link reference for the network name ensures that the VPC network is created before the firewalled rule. This is very important because Deployment Manager creates all the resources in parallel unless you use references. You would get an error without the reference because you cannot create a firewall rule for a non-existing network.



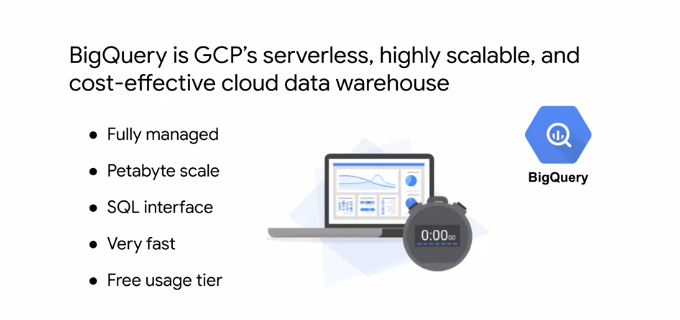
Now there are other infrastructure automation tools in addition to Deployment Manager that you can use in GCP. You can also use Terraform, CHEF, Puppet, Ansible, or Packer. All of these tools allow you to treat your infrastructure like software, which helps you decrease costs, reduce risk, and deploy faster by capturing infrastructure as code. You might recognize some of these tools because they work across many Cloud service providers. I recommend that you provision and manage resources on Google Cloud with the tools you already know. That's why in the upcoming lab, you'll have the choice of using Deployment Manager or Terraform to automate the deployment of Infrastructure. For more information on each of these tools, see the link section of this video.

**GCP Marketplace**

****

Let's learn a little more about GCP Marketplace. GCP marketplace lets you quickly deploy functional software packages that run on GCP. Essentially, GCP marketplace offers production grade solutions from third-party vendors who have already created their own deployment configurations based on Deployment Manager. These solutions are built together with all of your projects GCP services. If you already have a license for a third party service, you might be able to use a Bring Your Own License solution. You can deploy a software package now and scale that deployment later when your applications require additional capacity. GCP even updates the images of these software packages to fix critical issues and vulnerabilities but doesn't update software that you have already deployed. You even get direct access to partner support.

**BigQuery**

****

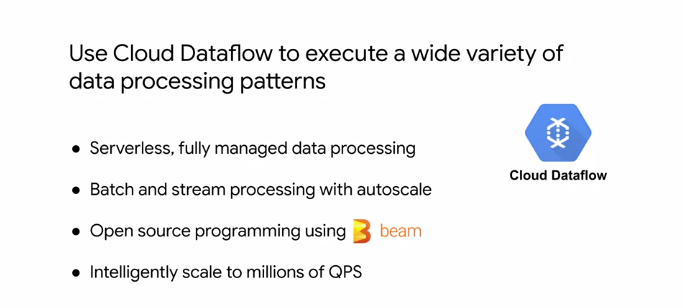
**BigQuery** is GCP's serverless, highly scalable, and cost effective Cloud data warehouse. It's a petabyte scale data warehouse that allows for super-fast queries using the processing power of Google's infrastructure. Because there's no infrastructure for you to manage, you can focus on uncovering meaningful insights using familiar SQL, without the need for the database administrator. BigQuery is used by all types of organizations, and there's a free usage tier to help you get started. For more information, see the links section of this video.

You can access BigQuery by using the GCP console, by using the command line tool, or by making calls to the BigQuery REST API, using the variety of client libraries such as Java,.NET or Python. There are also several third-party tools that you can use to interact with BigQuery, such as visualizing the data, or loading the data.

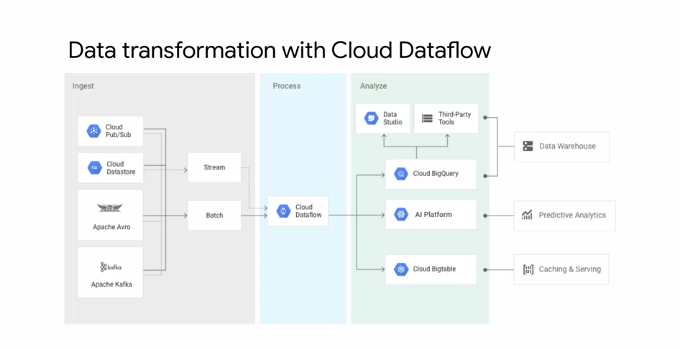


Here's an example of a query on the table with over 100 billion rows. This query processes over 4.1 terabyte, but takes less than a minute to execute. The same query would take hours if not days through a serial execution.

**Cloud Dataflow**

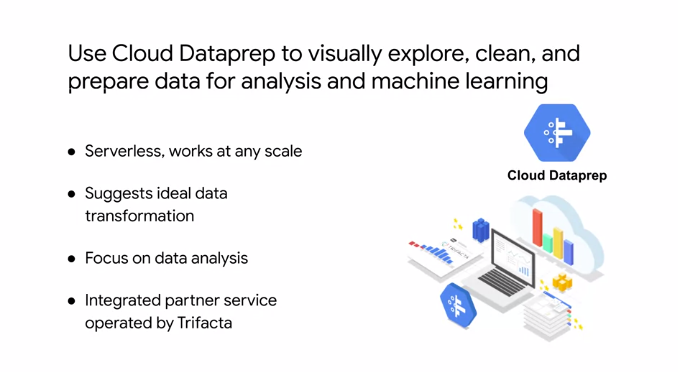
****

Let's learn a little bit about Cloud Dataflow. Cloud Dataflow is a managed service for executing a wide variety of data processing patterns. It is essentially a fully managed service for transforming and enriching data in stream and batch modes with equal reliability and expressiveness. With Cloud Dataflow, a lot of the complexity of infrastructure setup and maintenance is handled for you. It's build on Google Cloud Infrastructure and auto-scaled to meet the demands of your data pipeline, allowing you to intelligently scale to millions of queries per second. Cloud Dataflow supports fast, simplified pipeline development via expressive SQL, Java, and Python APIs in the Apache Beam SDK which provides a rich set of windowing, and session analysis primitives as well as an ecosystem of source and sync connectors. Cloud Dataflow, is also tightly coupled with other GCP services like Stackdriver, so you can set a priority alerts and notifications to monitor your pipeline and the quality of data coming in and out.



This diagram shows some example use cases of Cloud Dataflow. As I just mentioned, Cloud Dataflow processes stream and batch data. This data could come from other GCP services like Cloud Datastore or Cloud Pub Sub which is Google's messaging and publishing service. The data could also be ingested from third party services like Apache Avro and Apache Kafka. After you transform the data with Cloud Dataflow, you can analyze it in BigQuery, AI platform, or even Cloud Bigtable. Using Data Studio, you can even build real-time dashboards for IoT devices

**Cloud Dataprep**

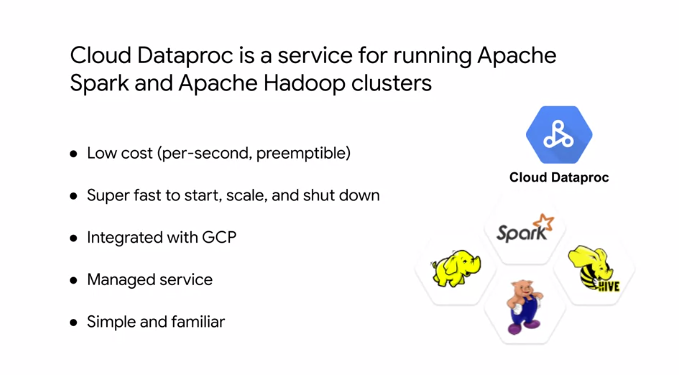
****

Let's learn a little bit about Cloud Dataprep. Cloud Dataprep is an Intelligent Data Service for visually exploring, cleaning, and preparing structured and unstructured data for analysis reporting and Machine Learning. Because Cloud Dataprep is serverless and works at any skill, there's no infrastructure to deploy are manage. Your next ideal data transformation is suggested and predicted with each UI input so you don't have to write code. With automatic schema, data types, possible joins and anomaly detection, you can skip time-consuming Data Profiling and focus on Data Analysis. Cloud Dataprep is an integrated partner service operated by Trifacta and based on their industry leading Data Preparation Solution Trifacta Wrangler. Google works closely with Trifacta to provide a seamless user experience that removes the need for upfront software installation, separate licensing costs or ongoing operational overhead. Cloud Dataprep is fully-managed and scales on demand to meet your growing data preparation needs so you can stay focused on analysis.



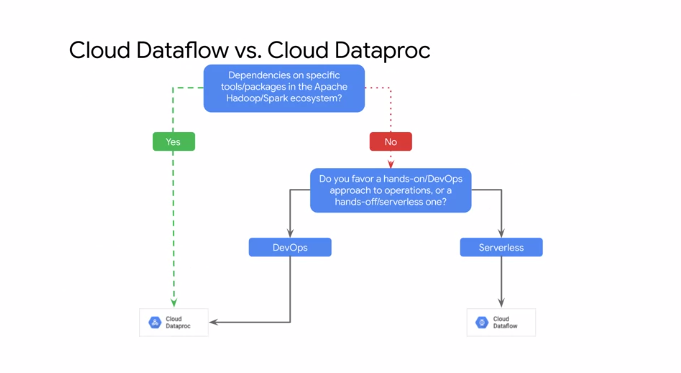
Here's an example of a Cloud Dataprep architecture. As you can see, Cloud Dataprep can be leveraged to prepare raw data from BigQuery, Cloud Storage, or a file upload before ingesting it into a transformational pipeline like Cloud Data flow. The refined data can then be exported to BigQuery or Cloud Storage for analysis and machine learning.

**Cloud Dataproc**

****

Let's learn a little bit about Cloud Dataproc. Cloud Dataproc is a fast easy to use fully managed Cloud service for running Apache Spark and Apache Hadoop clusters in a simpler way. You only pay for the resources you use with per second billing. If you leverage preemptible instances in your cluster, you can reduce your costs even further.

Without using Cloud Dataproc, it can take from five to 30 minutes to create Spark and Hadoop clusters On-premise or through other infrastructure as a service providers. Cloud Dataproc clusters are quick to start, scale, and shut down with each of these operations taking 90 seconds or less on average. This means you can spend less time waiting for clusters and more hands-on time working with your data. Cloud Dataproc has built-in integration with other GCP services such as BigQuery, Cloud Storage, Cloud Bigtable, Stackdriver Logging, and Stackdriver monitoring. This provides you with the complete data platform rather than just a Spark or Hadoop cluster. As a managed service, you can create clusters quickly, manage them easily, and save money by turning clusters off when you don't need them. With less time and money spent on administration, you can focus on your jobs and your data. If you're already using Spark, Hadoop, Pig or Hive you don't even need to learn new tools or API's is to use Cloud Dataproc. This makes it easy to move existing projects into Cloud Dataproc without redevelopment.



Now, Cloud Dataproc and Cloud Dataflow can both be used for data processing, and there's overlap in their batch and streaming capabilities. So how do you decide which product is a better fit for your environment? But first ask yourself whether you have dependencies on specific tools or packages in the Apache Hadoop or Spark ecosystem. If that's the case, you'll obviously want to use Cloud Dataproc. If not, ask yourself whether you prefer a hands-on or DevOps approach to Operations or a hands-off or serverless approach. If you opt for the DevOps approach, you want to use Cloud Dataproc. Otherwise, use Cloud Dataflow.