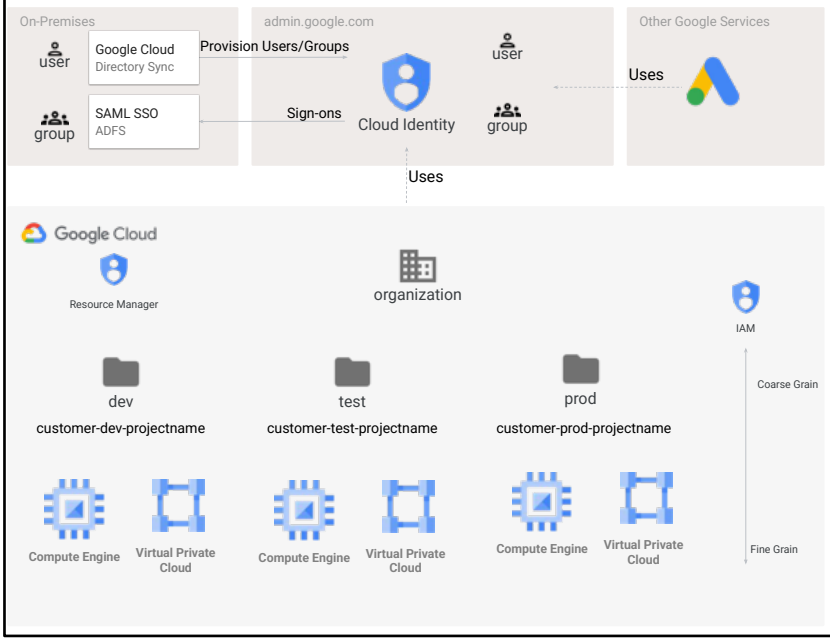
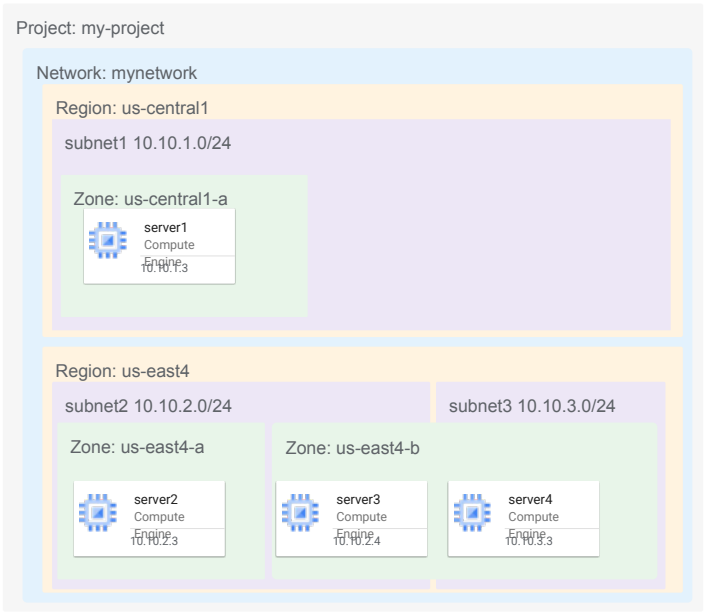
PCA Study Cards - IAM & Cloud Identity



- SSO and GCDS are mutually exclusive (although often used together)
- Resource Manager is where your hierarchy is defined
  - An organization is technically optional
  - Folders are optional as well
    - Can be nested
- IAM Permissions can be assigned at any level (org, folder, project, resource)
  - Lower generally is least privilege
- Three types of roles: Basic (primitive), Pre-defined, Custom
  - Basic (owner, editor, viewer) are generally limited to non-production or special cases
  - Pre-defined are most common
  - Custom have some limitations (not all permissions, limited number)
- Best practices
  - Assign to groups rather than user accounts
  - Assign lowest level practical
  - Assign fewest permissions possible to “get the job done”
  - Assigning at a higher level effects all current and future resources

Google Cloud

PCA Study Cards - networking basics



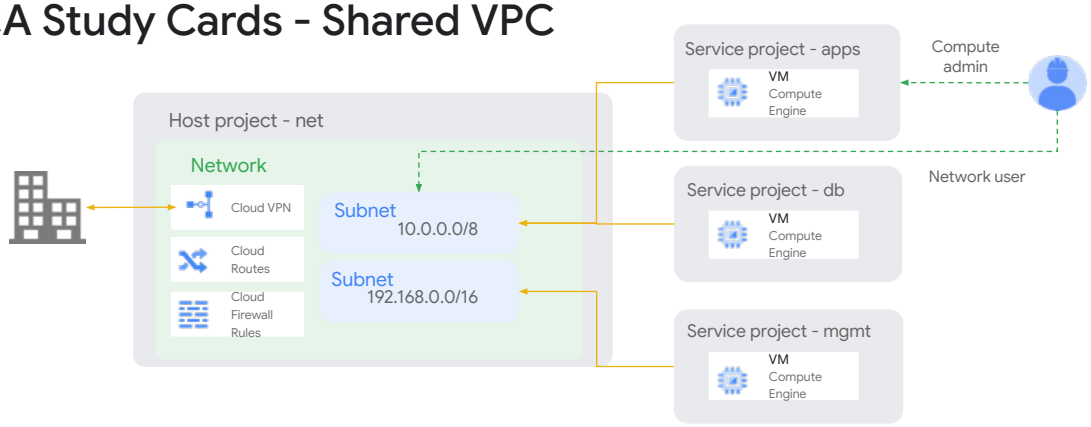
- A VPC belongs to 1 project
- A VPC can be present in every region across GCP (and is in the default configuration)
- No additional configuration is required for servers to communicate globally (VPNs or routers)
- A subnet crosses zones within a region, but cannot cross regional boundaries
- Implied Firewall Rules (65535):
  - Allow all egress traffic
  - Deny all ingress traffic
- Default rules
  - Allow SSH, ICMP, RDP
  - Block SMTP Traffic
- Lower the number of firewall rule the higher the priority (1 > 10)
- Components of a firewall rule:
  - Direction (ingress / egress)
  - Priority (0 to 65535)
  - Action (Allow / Deny)
  - Enforcement Status
  - Target
  - Source
  - Protocol
  - Log (1 or 0)

Google Cloud

https://cloud.google.com/vpc/docs/firewalls

```
gcloud compute networks create mynetwork --project=lclarkin-network
--subnet-mode=custom --mtu=1460 --bgp-routing-mode=global

gcloud compute networks subnets create subnet1 --project=lclarkin-network
--range=10.10.1.0/24 --network=mynetwork --region=us-central1
--enable-private-ip-google-access

gcloud compute networks subnets create NAME --project=lclarkin-network
--range=IP_RANGE --network=mynetwork --region=REGION

gcloud compute networks subnets create subnet2 --project=lclarkin-network
--range=10.10.2.0/24 --network=mynetwork --region=us-east4

gcloud compute networks subnets create subnet3 --project=lclarkin-network
--range=10.10.3.0/24 --network=mynetwork --region=us-east4
```

PCA Study Cards - Shared VPC



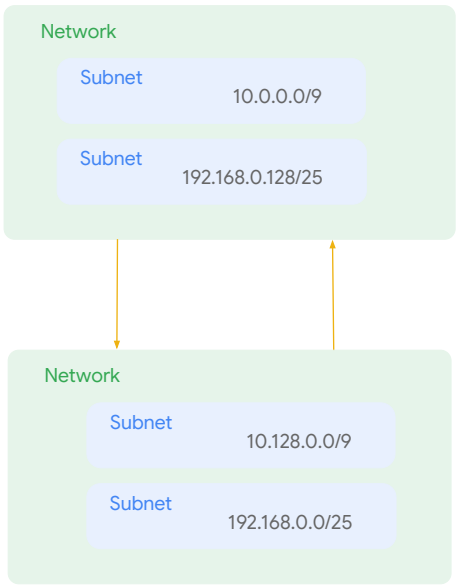
- ➔ Shared VPC is the most common way to share networks. Allows you the flexibility of having many projects (good for security / billings / etc) without the overhead of managing a lot of VPCs.
- ➔ Allows you to setup a robust network in the host project and share subnet(s) with service projects.
- ➔ Allows good security segmentation as admins on compute nodes don't need to admin network functions (only need user permissions).
- ➔ Connectivity to other networks (VPN and interconnects) and firewall rules can be centrally managed in the host project.
- ➔ Host and service projects ***must*** belong to the same GCP organization

Google Cloud

[SHOW?] Explain that Service Project is NOT any special kind of a project. It's just a project without a VPC and some VPC resources need to be shared from a Host Project
Explain privileges: Compute Admin in Service Project + Network User in Host project. Otherwise, a VPC can theoretically be created in a Service Project

- Recommended reading: go through the whole document explaining how to create, use and assign privileges in Shared VPC: https://cloud.google.com/vpc/docs/provisioning-shared-vpc

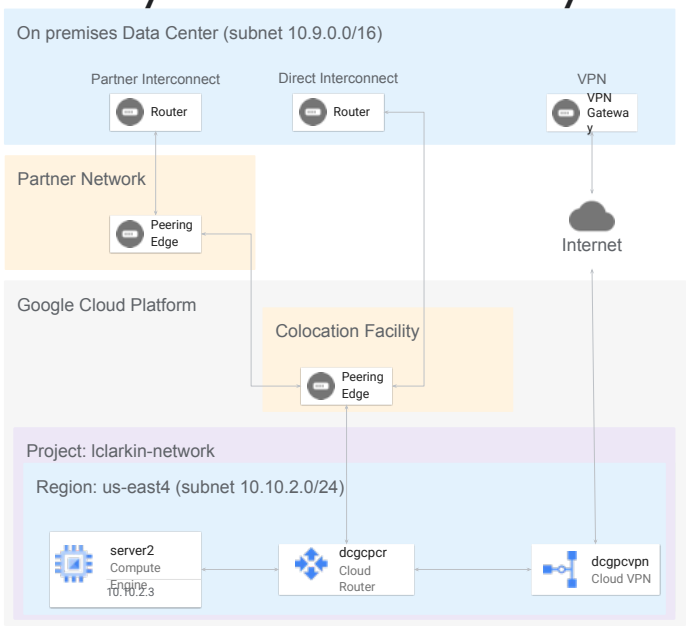
PCA Study Cards - VPC Peering



- ➔ Peering works both within and between GCP organizations
- ➔ When setting up the peering you determine which subnet(s) to publish routes to
- ➔ Administrators on both sides must configure the peering in order for it to work
- ➔ The peering between the networks is **not** transitive, so traffic will not route to any other networks peered
- ➔ Links between the networks are high throughput and very low latency (unlike connecting via a VPN)
- ➔ IP Networks ***cannot*** overlap

Note: Starting to see peering as part of the solution for GCP Products: Apigee X and Datastream configurations both require peering as part of the setup

PCA Study Cards - Connectivity



- Speeds (*)
  - VPN up to 3 gbps
  - Partner up to 50 gbps
  - Dedicated up to 100 gbps
- VPN Connections always go over the internet
  - The connection is encrypted using IPSec
  - There are pre-shared keys exchanged to facilitate
  - Must have a public IP address
- Interconnects (both direct and partner) are always to GCP, not Google
  - Consumer services / Workspace still go over the internet
- You should ***never*** have only 1 connection into GCP
  - Connections should be in two separate regions (not zones)
  - You can use a different solution to backup the primary (primary interconnect, vpn as backup)
- IP Address ranges cannot overlap in any of the architectures

* Can stack some of these solution for higher speeds

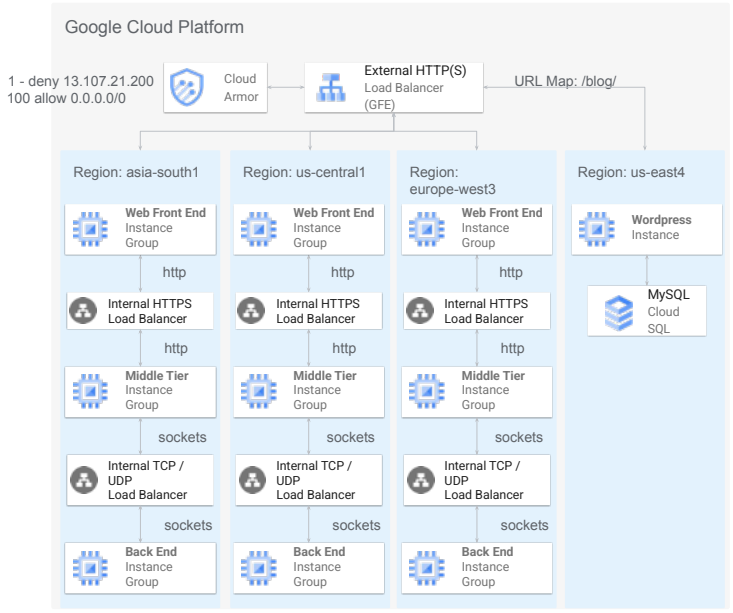
Google Cloud

Dedicated
https://cloud.google.com/network-connectivity/docs/interconnect/concepts/dedicated-overview

Partner
https://cloud.google.com/network-connectivity/docs/interconnect/concepts/partner-overview
https://cloud.google.com/network-connectivity/docs/interconnect/concepts/service-providers#north-america

Cloud VPN
https://cloud.google.com/network-connectivity/docs/vpn/concepts/overview

PCA Study Cards - Load Balancing



- External HTTP(S) Load Balancer
  - Global Service (*)
  - Traffic to "closest" endpoint
  - Single Anycast IP Address
  - Can be used for workloads on-premises or other clouds
- URL Map apply to both Internal and External HTTP(s) Load balancers
  - Directs to different backends
  - Based on a fragment of the url or host names
- Cloud Armor
  - rules to protect vulnerable backend services from OWASP Top 10 attacks like SQL Injection and cross site scripting
  - Allow / Deny lists for IP addresses and regions
  - Like Firewall rules, lower the number higher the priority (1>10)
  - Named IP list are 3rd party maintained list for malicious IP addresses
- Additional items to remember:
  - Health Checks on backends
  - Firewall rules
  - SSL Proxy (not shown) is for non-http traffic

* requires premium network tier

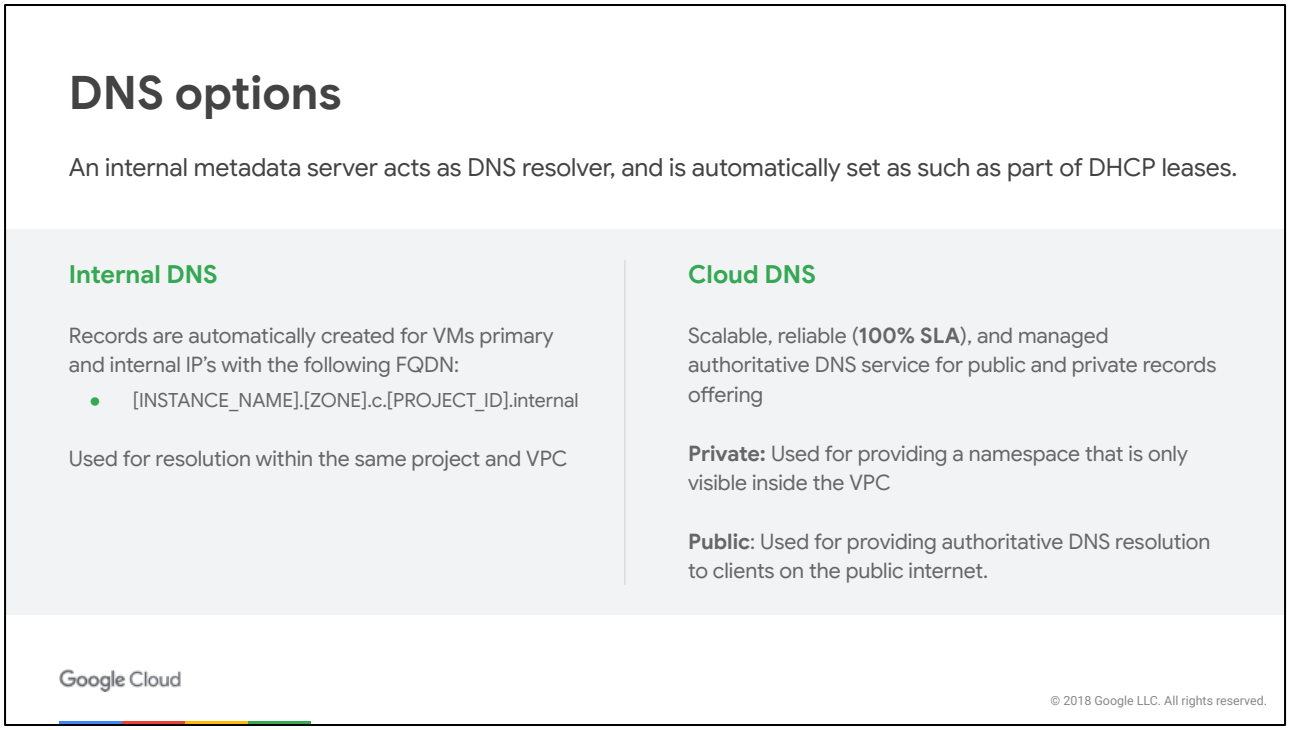
Google Cloud

Usually LBs are layered: external HTTPS -> Interal HTTPS -> Internal TCP/UDP

3rd party provider list for CA
https://cloud.google.com/armor/docs/armor-named-ip#ip-list-providers

Most common Organization Policies

Policy Constraint	Description
compute.vmExternalIpAccess	A list of project/zone/instance names that are allowed to have external IP addresses and deny all others. Attempts to create any other VMs with an external IP address will fail.
compute.trustedImageProjects	A list of projects that contain trusted images that can be used as the basis for a VM and deny all others. Attempting to instantiate a VM with an image from another project is denied.
compute.skipDefaultNetworkCreation	Disables the creation of default VPC when creating a project. The default VPC uses auto mode subnetworks and includes default firewall rules which are often incompatible with production deployments.
iam.disableServiceAccountKeyCreation	This boolean constraint disables the creation of service account external keys where this constraint is set to `True`.
compute.restrictVpcPeering	This list constraint defines the set of VPC networks that are allowed to be peered with the VPC networks belonging to this project, folder, or organization.
serviceuser.services	This list constraint defines the set of services and their APIs that can be enabled on this resource and below. By default, all services are allowed.
gcp.resourceLocations	BETA: This list constraint defines the set of locations where location-based GCP resources can be created. Policies for this constraint can specify multi-regions such as asia and europe, regions such as us-east1 or europe-west1, or individual zones such as europe-west1-b as allowed or denied locations.
sql.restrictPublicIp	This boolean constraint restricts configuring Public IP on Cloud SQL instances where this constraint is set to True. This constraint is not retroactive, Cloud SQL instances with existing Public IP access will still work even after this constraint is enforced. By default, Public IP access is allowed to Cloud SQL instances.
sql.disableDefaultEncryptionCreation	BETA: Restrict default Google-managed encryption on Cloud SQL instances
compute.requireShieldedVm	This boolean constraint, when set to True, requires that all new Compute Engine VM instances use Shielded disk images with Secure Boot, vTPM, and Integrity Monitoring options enabled. Secure Boot can be disabled after creation, if desired. Shielded VM features add verifiable integrity and exfiltration resistance to your VMs.
compute.restrictSharedVpcHostProjects	Restrict Shared VPC Host Projects This list constraint defines the set of Shared VPC host projects that projects at or below this resource can attach to. By default, a project can attach to any host project in the same organization, thereby becoming a service project.
iam.allowedPolicyMemberDomains	This list constraint defines the set of members that can be added to Cloud IAM policies. By default, all user identities are allowed to be added to Cloud IAM policies. The allowed/denied list must specify one or more Cloud Identity or G Suite customer IDs. If this constraint is active, only identities in the allowed list will be eligible to be added to Cloud IAM policies.



DNS on a very high level!

We basically have 2 options of DNS in GCP:

- Default, internal DNS one, which is injected into every VM metadata: so each VM automatically gets DNS records from the whole VPC to a local table and can communicate with other VMs in this VPC based on automatically created FQDNs (which you can see on the left)
    - [SHOW] alternatively, you can give VMs alternative names when you create them (Advanced -> Networking) - but you need to remember that those names are not automatically managed is such a way,
- And you can supplement it with Cloud DNS, where you'd want a more robust service allowing you to use DNS across VPCs and with on-premises, also using some custom DNS records which you can create.

TL;DR / Purpose of the slide:

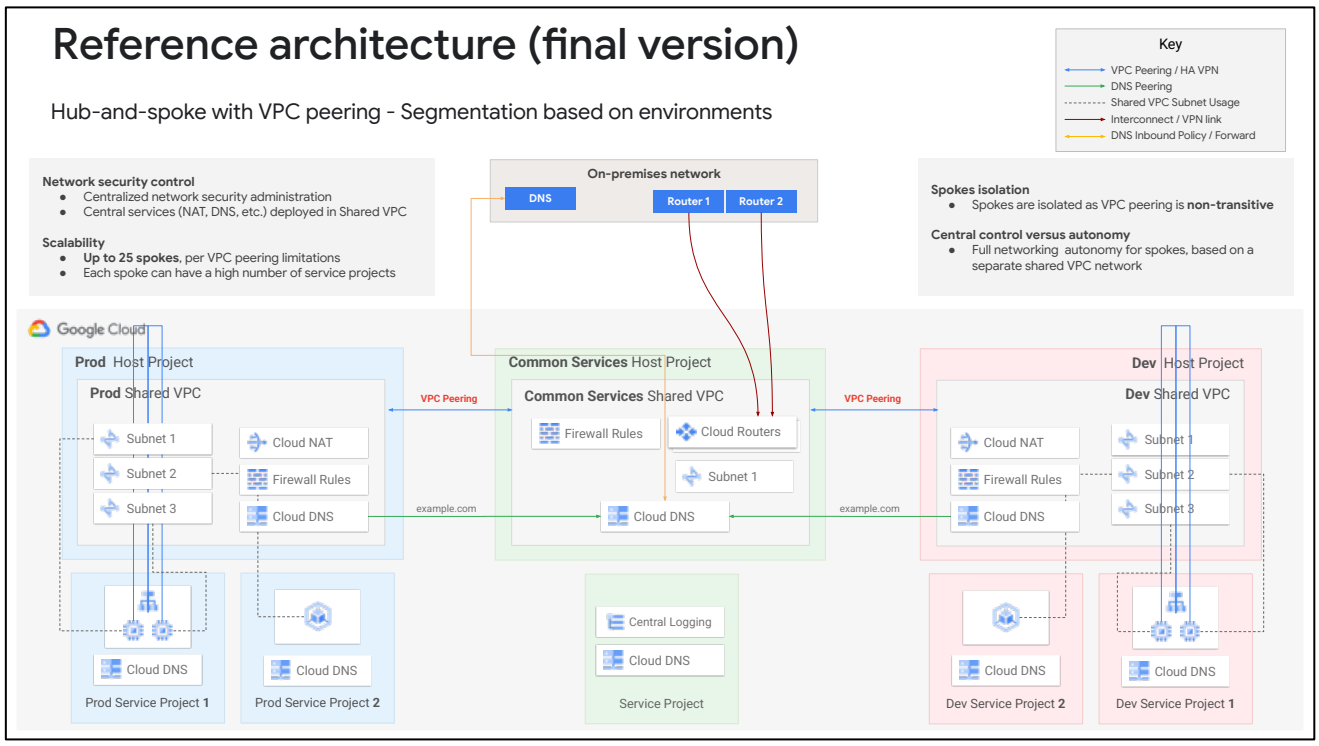
- Review **DNS options** in GCP

Key points:

- **Metadata server**
  - Every VM instance has a metadata server used for querying instance info, e.g: name, ID, startup/shutdown scripts, custom metadata, service account
  - The metadata server also acts as the **DNS resolver** for both **internal and external resolutions** (i.e resolving hostnames in the public internet)
  - It is set on VM's as part of default **DHCP leases**.
  - **Overriding** it is **possible** by **customizing DHCP** configuration, however it is **not a common pattern**
- **Internal DNS**
  - **Records** are automatically created for **VM's primary IP** (not Alias IP's), e.g:
    - instance1.europe-west1b.c.projectx.internal
  - Used **within the same VPC**
- **Cloud DNS**
  - Managed DNS service. 100% SLA
  - **Public** as an **authoritative DNS** resolution accessible from the public internet
  - **Private** for **DNS resolution** inside a **VPC**
- In the next slides, we will discuss **Cloud DNS Private** in details

Probing questions (optional):

- Would you like to use your own custom DNS names for VM resources, or would you like to use the default internal DNS names? (i.e [instance_name].[zone].c.[project_id].internal)



from DNS perspective, there two common patterns which I'd like you to be aware of:

- First one is to differentiate between on-premises resources and GCP ones by using different subdomains (for example you'd use corp.example.com subdomain for on-premises and gcp.example.com for GCP). And then you would configure DNS in such a way that name resolution for GCP resources would be offloaded to Cloud Dns, and name resolution of on-premises resources would be offloaded to on-premises servers. Technically it's done using mechanisms called Inbound and Outbound Forwarding, but we would have to spend too much time to explain how they work.
- And the second pattern is to centralize DNS name resolution within GCP by configuring that it should happen in some specific, central GCP project. You would configure it using so-called DNS peering, which should not be mistaken for VPC Peering!).

And this is how you would set it up in a more common version, where you have not one, but multiple Shared VPCs: usually one for common services and 2 or more additional ones for your workloads, connected using hub-and-spoke setup.
This setup is very similar to the previous one, but it's more complex because of multiple VPCs in GCP. And just like we said it's a best practice to centralize DNS resolution, we might use the mechanisms we have seen in the slide between on-premises and this

central VPC for Common Services, but we would add DNS Peering from those Dev and Prod spokes, just because they should not resolve everything on their own, but rather - redirect resolution to the central VPC, which can handle it using the connectivity we set up to on-premises DNS Servers.

Those spokes are connected through VPC Peering or Cloud VPN, and on top of that, we create DNS Peering from Spoke networks to the Hub network, plus we set up Cloud DNS in this Hub network -> either by creating local DNS zones, or creating DNS Forwarding which forwards DNS requests to on-premises DNS Servers.

TL;DR / Purpose of the slide:

- **Hub-and-spoke with VPC peering** for **environments segmentation**

Key points:

- **Use case**
  - hub-and-spoke model
  - Each spoke represents a **larger network segment**, e.g.environments (prod, test, dev), business units, etc. We will need a **handful of spokes.**
  - Spokes shouldn't be able to communicate with each other.
  - Within each spoke, it should be possible to separate connectivity between workloads.
- **High-level**
  - In this design, the hub is a **Shared VPC** and is **VPC peered** to the spokes, which are in turn **Shared VPCs** as well.
  - Making use of **Shared VPC** helps keeping the design scalable and simple.
  - **VPC peering** is useful for **lower latency**, fits the spokes **scale requirements,** and helps **isolating connectivity between the spokes.**
- **Network connectivity**
  - **Hub**
    - Common services host project hosts a **VPC** that is **connected with on-premise** (VPN / interconnect)
    - Centralizes **administration** of **network services** for connectivity **between spokes and on-premise**, such as FW, DNS, routes, etc.
    - We could also have a **security appliance** in the hub for **east-west** and/or **north-south** communication, depending on the

        - need.
    - **Spokes**
        - Spokes are connected to the hub with **VPC peering** to ensure **low latency**, and **minimal management overhead**.
        - **Separation between spokes** is ensured due to **non-transitivity**.
        - **# of spokes**: As spokes represent larger network segments, the **upper limit of 25 peered networks**/spokes is **sufficient**.
        - **Each spoke** is essentially a **Shared VPC host project** with **network administration autonomy**: FW rules, NAT, DNS, routes, etc.
        - To allow a **path between spokes and on-prem, custom routes exchange** is configured between the hub and each spoke.
        - Controlling **network access between workloads** within each spoke is done **based on VPC firewall,** which we will discuss in a later slide.
- **DNS topology**
    - Similar to the hub-and-spoke model we have shown in the DNS topology section
    - **Google Cloud to on-prem**: **DNS forwarding zone**
    - **On-prem to Google Cloud: Inbound server-policy** allows access to inbound nameservers from on-prem
    - **Spokes to hub**: **DNS peering** to allow making use of the forwarding zone and querying on-prem
    - **Hub to spokes**: **DNS peering** to allow on-prem query spokes, and not the hub only
- **Additional design considerations** regarding **VPC Peering**
    - Communication between **on-prem** and **products** (e.g: Cloud SQL Private IP, GKE private masters) is not possible when **using VPC peering** due to **non-transitivity**.
        - **Workaround** by going through a **proxy** in the environment VPC. See link.
    - When using **VPC Peering**, some **limits are shared** between the VPCs. E.g: forwarding rules

Probing questions (optional):

- None