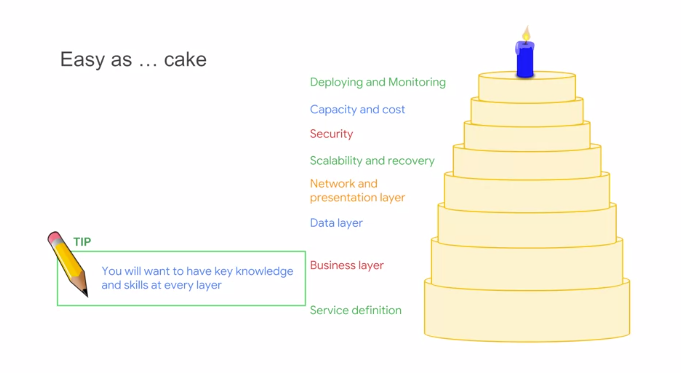
**Designing and Implementing**

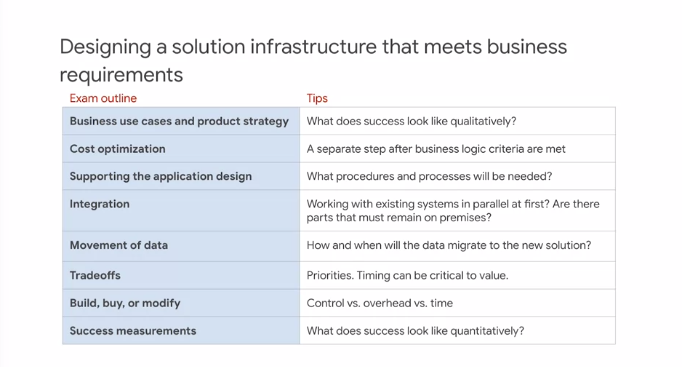
This module covers designing and implementing infrastructure solutions. Design can get complicated. Do you have an approach to design? It's easy to confuse elements if you don't use an organized method. Do you have favorite design elements? For example, do you find most of your designs start with VMs? You'll want to overcome these biases by understanding the infrastructure services available and when to select them. Today you'll be learning about and preparing for the Professional Cloud Architect exam. A lot of that has to do with design. Before you can design a solution, you need to understand the building blocks, the underlying services, and technologies that make up solutions in Google Cloud.

****

Here's a tip, use a layered model like this one. It'll help you organize your thinking about each exam question, so that you'll more easily recognize and focus on what's important. Professional Cloud Architects often use layered models to organize or separate solution designs. It makes it much easier to deal with the complexity and to make sure there are no dropouts in the design. This model comes from our design and process class. Sometimes people ask if the order of the layers is significant in this diagram well, the order is significant. You start with service definition at the bottom and move up the layers until you're planning the rollout and maintenance of the solution. So, each layer has dependencies on the one beneath it. Sometimes people also ask if the size of the layers in this diagram is meaningful.

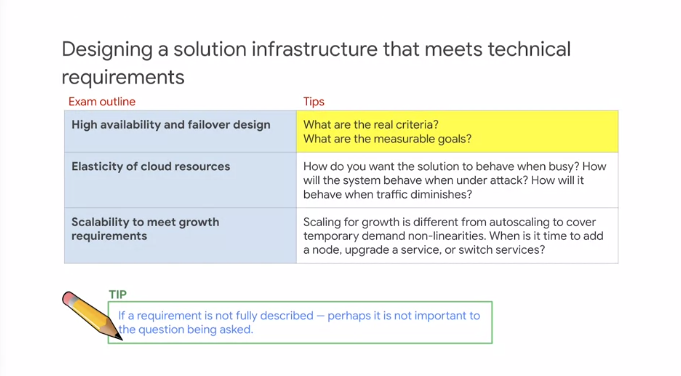
For example, a security more important than costs because the layer is bigger in the diagram and the answer to that is no. Each of the layers is equally important to your design. During the exam consider which layers are required as part of the answer. Next consider which layers are involved in the background. This can be a very useful way to surface exactly what skills are required by the exam question and help focus your time and attention during the exam. If you can rule out a layer there will be more time to focus on what is important in the question. Experienced Cloud Architects have assemblies that are familiar to them. A good analogy is a chess expert who sees the board and combinations of pieces rather than individual moves. When they change out one element, they may already know the consequences to security throughput reliability and so forth. This familiarity takes time and practice to develop. Until you develop your own familiar assemblies it's recommended that you use the model to consider changes in design from each layer and iterate to make sure you've covered everything. It's better to be careful with design and followed a structure approach than to accidentally introduce a design flaw in your thinking and maybe get an exam question wrong.

**Designing a solution infrastructure that meets business requirements**

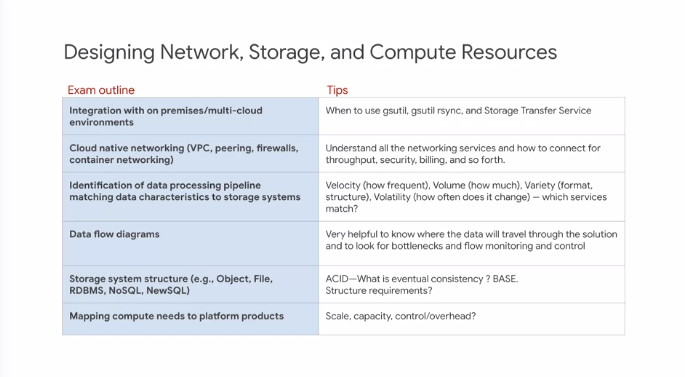
****

This class follows the exam guide. So whenever you see a slide like this, a blue column contains items directly from the exam guide and the white column contains tips and advice directly relevant to each outlined item. You can read through these yourself. I'm going to highlight and discuss one or two of these per slide. When we speak about business requirements, we're asking the question," What are the customer's needs and expectations?" Questions on the exam are realistic, so on a job, these discussions would likely be with a business stakeholder, and you'd need to be prepared to answer these questions and their concerns. You'll notice that the first and last items in the list have to do with determining the criteria for success and deciding how to measure that. It's very important to be explicit about exactly what you're trying to achieve. These items are often stated qualitatively at the beginning and are measurable and quantitative at the end. It's extremely difficult to optimize for success criteria and minimize cost at the same time. For this reason, it's recommended that you divide the activity into function first and cost optimization second. Another suggestion is to keep existing systems working in parallel for a period of time and verify that the new system is behaving as expected. Be on the lookout for these slides, with a blue column and a white column. They're intended to explicate for you what you see on the exam guide and how to prepare. Identify the context in which the solution is being employed, think about what the business is trying to accomplish. That will help distinguish between solutions that might seem equally correct. There are no universal remedies for all situations. Different design patterns have their best uses and their limitations. In every design, there will always be issues that need to be considered. For the exam, you need to know not only design patterns but their context of use and especially what circumstances to apply one design instead of another. What's the priority or the blend of priorities? Good, fast, inexpensive? How will these priorities be measured? What are the measures of time, budget or qualities? How will you know if your design meets requirement if you can describe them and measure them in some way? One way to think about these decisions is to consider whether to build a solution from scratch yourself, purchase solution off the self or from a vendor, or customize existing products and services or trade off speed of implementation for perhaps fewer or more general features. If you build it yourself, you get absolute control over what the solution does and how it works. If you go to the experts who already have handled similar problems and have experienced creating and implementing similar solutions, it's often the fastest way to get the matter resolved. Adapting what you already have, which might be partially depreciated or already paid for is often fast, but you'll probably compromise on features. A great example of this is with our machine learning products. We have pre-trained models, like Cloud Vision API and Natural Language API, which are fast, because you can just grab them and use them. We have Cloud ML Engine and TensorFlow, which allow you to take complete control, and we have Cloud Auto ML, which allows you to customize and add to a pre-trained model.

**Designing a solution infrastructure that meets technical requirements**

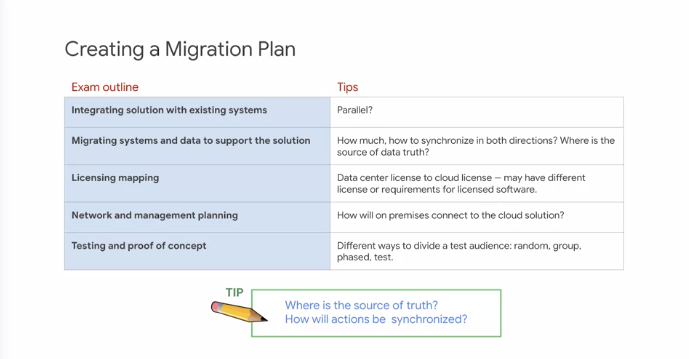


If a requirement is not described fully, perhaps it's not important to the question being asked. Notice that the exam guide highlights fail-over right next to elasticity. Why would that be the case? Well, consider this example. A client comes to you and says, "We're happy with our data center solution. We want to use the cloud, but only for fail-over and disaster recovery. Can you help?" So, you provide an IT solution in the cloud that gives them a disaster recovery scenario and fail-over capabilities. So, if elements of their existing solution go down, the cloud can provide an alternative. Some time goes by and the client comes back to you with an additional request. "The fail-over solution is working great, but we have another problem. Sometimes we get demand above our capacity and we don't want to invest in more data center capacity. Can you help us burst into the cloud so when we have extra demand, we can use the cloud for temporary extra capacity?" You modify some of the elements that you built for fail-over and you give them cloud bursting capability. So, there is the elasticity. Some more time goes by and the client isn't happy with their colo. They're paying for racks and they don't want to pay more and the equipment needs a refresh and they ask you for some help, "Can you help us migrate our application fully to the cloud so we can scale more dynamically to meet growth?" Of course you help them. Now, this is a very common adoption pattern. This is a natural and orderly progression of events. Now, you see why they're located together in the exam guide. For the exam, you need to think about what you would measure in the situation. You might not need to calculate anything, but knowing what you need to be measured will help you focus on what information is important in this question. Architectural principles. Compose simple services and evolve them, plan to avoid conditions that would lead to failure such as bottlenecks, consider the hardware or service limitations in the cloud, but also plan for resiliency and fast recovery when failure occurs. Consider the time value of the result of the solution. Here's an example. If a data processing job takes 3,000 years to run, the proposed method might not fit the business requirements and an alternate solution might be better. An old business saying, in business, all values are real-time values. Meaning that you might want to determine when a metric might change or how this will affect the value to the business. So, time value, what happens if the solution is delivered sooner or later? In a design class, we often say that all values are real time. In this case, constant value might not have any additional value if it's delivered earlier or later. Here's example two. This is a solution that has additional value if delivered sooner. So, the sooner it's delivered, the more value it provides. In example three, it's a solution that has diminishing value if it's delivered late. An example four has a solution that has no value if it's not delivered on time. One example would be an online ordering system tied to an event. If it's ready early, it provides no value because the customers are not ready to take orders and if it comes in a day late, then the customers will no longer be interested. Always take note of state information and consider the impact of storing and retrieving state information on scalability. All work in all systems can be divided up into job shops and assembly lines. In a job shop, a single server, service, or unit performs multiple steps. In an assembly line, each server, service, or unit performs one well-defined step and hands the work off to another worker for the subsequent steps. Food preparation can help illustrate these principles. Here's an assembly line kitchen. Each person is responsible for a specific action. One person places one kind of food on the plate, another puts on the sauce, a third places the garnish. On the one hand, there's not much to remember because each person individually just does one discrete thing over and over again. So, there isn't much state information. But what this model requires is coordination. If something happens and one person changes what they're doing, the line has to adjust. If a person in the middle takes a break, everyone further down the line stops working and the work queues up from the workers earlier in the line. However, this approach can really be worthwhile because each individual can focus on being really fast and efficient at their task. So, this model scales very well. In this instance, there is a single chef who's doing all of the steps, placing the food, adding the sauce, garnishing, so forth. The chef needs to remember what step was just performed and what step needs to happen next. They may need to change tools or fetch different resources. All of that information needs to be remembered for the work to proceed is called state information. Just exactly how much state information is required and where the state information is stored has a huge influence on scalability. just to recap, a job shop has to keep track of where it is in the process and that's the state information. The assembly line doesn't need to keep track of state. However, the units or microservices have to coordinate with each other. So, queuing and messaging becomes important as an alternative to keeping state. You can't get away from state in all cases. In those cases, you have to try to make state scalable and reliable and that leads to another common design pattern you'll want to know. Familiarize yourself with common design patterns, not just what they are, but how they work. It can come in handy to imagine the question in the context of a solution. For example, dividing the problem solution into front-end and back-end, stateless and stateful can be very helpful in surfacing the key design issues. There are resources, for example, architectures in our online documentation. The design shown is a general solution. It doesn't fit all cases, but is very common. First there's a scalable front-end with fail-over. Second are many small stateless servers for back-end scalability. Third, is the state information isolated and separated from the front and back-end.

**Designing Network, Storage, and Compute Resources**

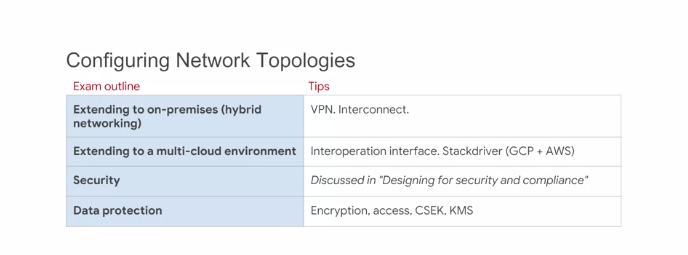
There's usually more than one way to accomplish the same function in a system using different cloud technologies. However, the technologies make a difference in the overall system and its qualities. It helps to narrow down your options first based on what could work, and then decide which service is best in a circumstance. Identifying bottlenecks is especially useful for questions involving building out from existing solutions. For example, the current system can support X number of users, and the goal is to support Y number of users. What's the bottleneck in the current design? Is it bandwidths, gigabytes, queries per second? Where will the application hit its limits? This is often the factor that determines which solution is best in the circumstance. You need to be able to read data flow diagrams and to clearly think about the progression of data through a system. Don't assume that data-flow symmetrically, or that the capacities are symmetrical. Acid versus base is essential data knowledge that you'll want to be very familiar with, so that you can easily determine whether a particular data solution is compatible with requirements identified in the case. For example, for a financial transaction, a service that provides only eventual consistently might be incompatible. Did you know that in some cases and eventually, consistent solution can be made strongly consistent for a specific limited use case? In Cloud Datastore, there are only two APIs that provide a strongly consistent view for reading entity values and indexes. First, the lookup by key method and second, the ancestor query. Database services provide a model of consistency. Consistency makes certain guarantees with respect to data transactions. Whatever guarantees are not made by the data service becomes a responsibility of the application code. You should know the definition and significance of atomicity, consistency, isolation, and durability.

**Creating a Migration Plan**



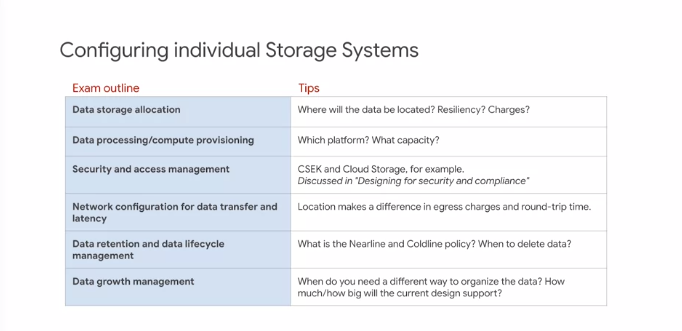
In migration to cloud, you're always starting with data and processing in a data center. So, obviously that's the source of truth, and the cloud version is just a copy. But as you go through the migration plan, what happens to the truth? Eventually, you want the cloud version to be either equal and related to the data center version or more likely to replace it and allow the data center resources to be repurposed. Where is the source of truth? And when does that change? Often and ambiguous technical situation like which data storage service to select or which compute platform to choose is easily resolved if you think about it in practical terms and go beyond theory. Look for practical clues in the case question not just technical clues. We often talk about implementing complete solutions in the cloud. That certainly simplifies the design, but realistically there often elements that cannot be migrated from the data center. What resources will remain on-prem? What processes are necessary to integrate with existing systems in the cloud in GCP or in other clouds and in the data center? How do you prove to yourself that the solution will work and isn't just a good theory? Do you need to run a proof of concept? If you're moving something from a design for efficiency, for cost reduction, consider the consequences at all the layers. For example, if you reduce capacity of a solution because there are fewer users, would also reduce or remove the availability or disaster recovery features. After a solution is designed and implemented, it goes through a period of stabilization prior to the completion of delivery and hand-off. What do you need to include in the design to stabilize the solutions? What could change or what might need to be measured and adjusted? In some cases. There may be multiple stakeholders. The best solution will solve for all interests or for the common interest. For example, a CEO wants to improve data analytics, the CTO wants lower cost. At first the two interests might seem in conflict, however, moving to a different more efficient solution may solve both requirements. There's an old saying in IT, "Every 10 X in growth, something breaks." This is because the assumptions that went into the original design are no longer true. Quite often scaling up changes the nature of the problem and requires revisiting assumptions. So, in exam questions if you notice rapid growth or other indicators of large-scale change, start thinking about what might have broken or what might be ready to break. What processes will you need to define, to keep the solution operating? Monitoring, logging, metrics, to manage too. Often the design isn't done at hand-off. In practice, most cloud architects are required to stay on a project for a period after delivery to ensure that the solution continues to work and is maintainable.

**Managing and provisioning solution infrastructure**

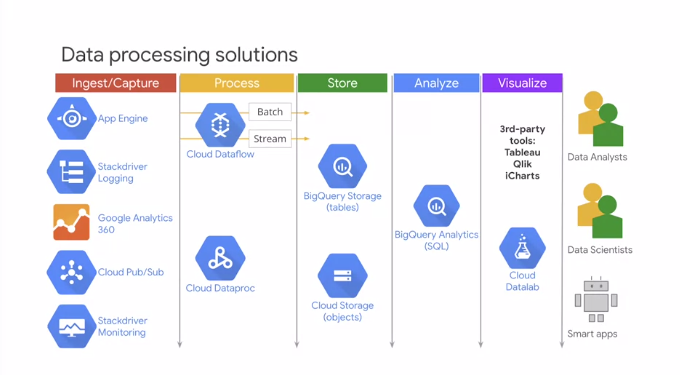


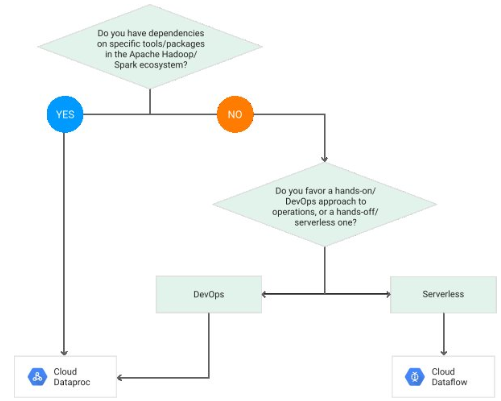
Managing and provisioning solution infrastructure. If you think about it, managing and provisioning are both about capacity and demand and choosing the right infrastructure components to support and adapt to the demand. External connections, know your options. Internet, VPN, Cloud Router and the various flavors of direct interconnects. Google Cloud Networking is not like other vendor networks. Not like traditional IP networks and not like other SDA networks. That's networking in the cloud and you need to know how you might handle migrating an existing data center network into a GCP network. Most confusing for cloud experts from other vendors clouds and yet it makes so much sense. A number of features of Google Cloud Networks eliminate the extra work where methods from traditional IP have been inherited out of habit. Subnetworks can extend across zones in the same region. One VM and an alternate VM can be on the same subnet but in different zones. A single firewall rule can apply to both VM's even though they're in different zones. This makes it much easier to design and implement resilient or high-availability solutions. Know your options. When would you choose which method? Have you considered backup connections? What happens if an interconnect goes down? How will you perform maintenance? Does the application need to stay up? You can create multiple VPNs for higher bandwidth or for alternate paths as backups. But, if the Internet goes down and you have one Internet provider, there's no alternative. Security in the field is practical. The goal is to raise the cost of violating the security above the value of the data. A good model of this is a castle. The bigger and higher the castle walls, the greater the treasure inside or conversely, if you have great treasure inside, you want to raise the castle walls just beyond where it would be worthwhile for someone to store in the castle. In the exam, think practical security. What's needed? Unless the requirement is compliance with the standard. Even if you comply with the standard, that might not be sufficient to meet business security requirements. Sometimes, standards are floors, the minimum allowable rather than ceilings, the maximum possible.

**Configuring individual Storage Systems**

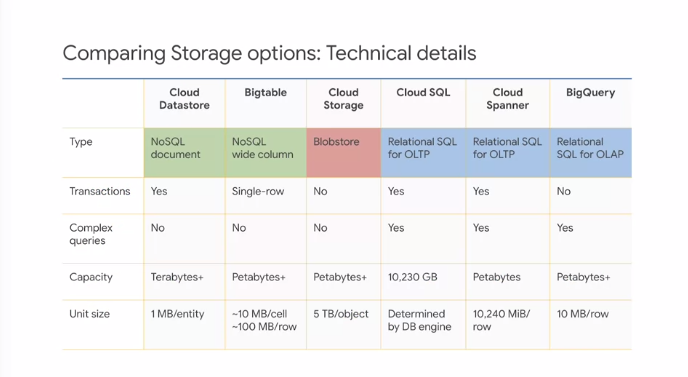
****

Dig into the details in each storage or data option. Learning the difference between BigQuery and Cloud Bigtable and which one is a managed service and which one is serverless is not something you want to be doing during the exam.



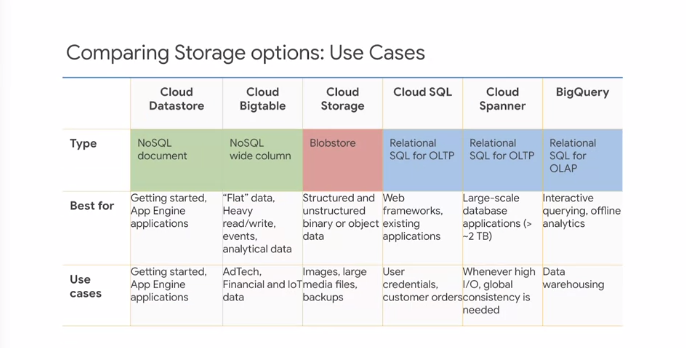


BigQuery is an inexpensive datastore for tabular data. It's cost comparable with cloud storage, so it makes sense to ingest into BigQuery and leaves the data there. Ingesting the data depends on where it's coming from. Cloud logs, gap, can directly be ingested into BQ. From pub sub, you have an API. In the most general case, you can use Dataflow and write code to ingest data in batch or stream. You could also use open source tools like Spark or Hadoop to do the processing in which case you'd use Cloud Dataproc. The analysis itself is done by BigQuery. Results can be visualized in iPython notebook like in datalab or CoLab or in third-party tools. So, BigQuery's role is in both storage and in analysis. In other words, it's a data warehousing solution.



Cloud Bigtable is not a relational database. It does not support SQL queries or joins nor does it support multi-row transactions. Also, it's not a good solution for small amounts of data like less than one terabyte.

* If you need full SQL support for an online transaction processing system, that's OLTP, consider Cloud SQL and Cloud Spanner.
  + Cloud Spanner is particularly suited for databases larger than about two terabytes and databases that will be written to by global clients.
* If you need to store immutable blobs larger than 10 megabytes such as large integers or movies, consider cloud storage.
* If you need to store highly structured objects or if you require support for acid transactions and SQL-like queries, consider Cloud Datastore.
* If you need interactive querying in an online analytical processing or OLAP system, consider BigQuery.

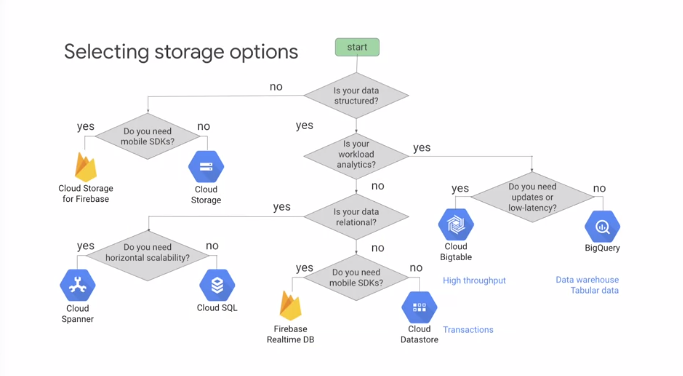


You should probably commit this table to memory and be able to use it backwards. For example, if the exam question contains data warehouse, you should be thinking BigQuery is a candidate. GoogleCloud Platform delivers various storage service offerings which remove much of the burden of building and managing storage and infrastructure. Like Google's other cloud services, storage services free you to focus on doing what you do best and differentiating at the application or a services layer. Google's storage offerings range across the spectrum.

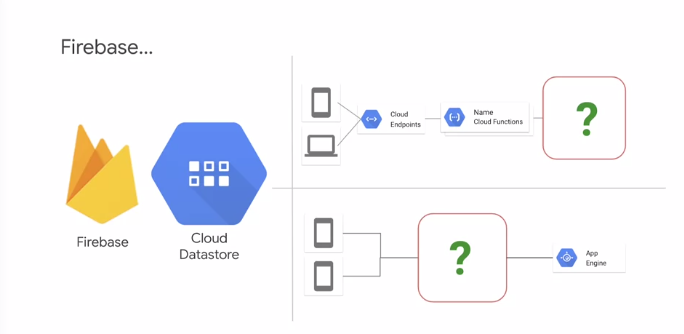
You can use different types of storage in the same project. Cloud SQL gives you fully managed MySQL so you have relational DB and a more traditional approach to queries, cloud Datastore provides a nearly infinitely scalable schema-less solution.

If you want a disk, you can mount persistent disk as a block store that can be used by a Compute Engine or just pure data and blobs, use cloud storage when that will deliver what you need.

Cloud Bigtable offers companies a fast fully managed infinitely scalable NoSQL database service, ideal for web mobile and IOT applications.

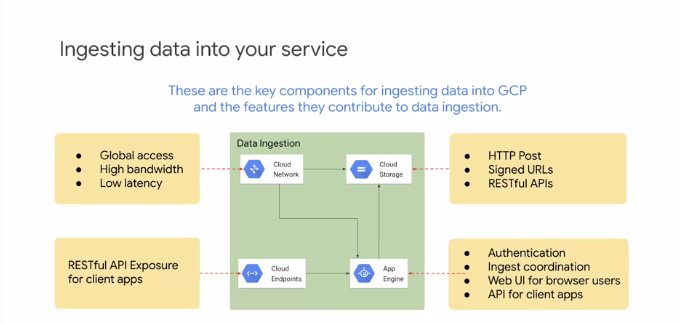


BigQuery is recommended as a data warehouse. BigQuery is the default storage solution for tabular data. Use CloudSQL if you need transactions, and use Cloud Bigtable if you want low latency and high throughput. Firebase differs from Cloud Datastore in many significant ways.

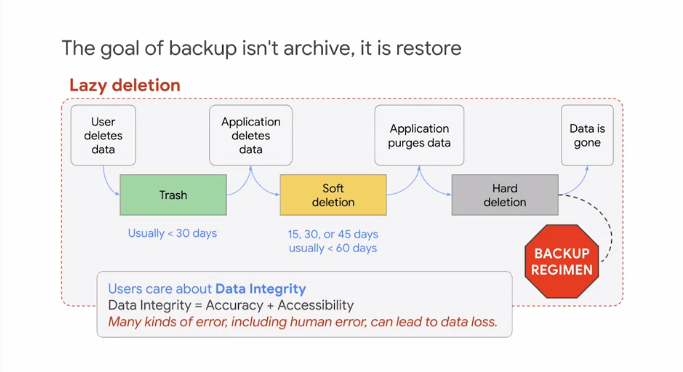


Firebase is a mobile platform that provides features beyond storage including authentication, notifications, and real-time synchronization of clients. Cloud Datastore is a NoSQL database. Cloud Bigtable can scale to massive amounts of data, Cloud Bigtable queries can be more sophisticated than Firebase queries. The top one is a scalable web app with mobile support. Cloud Functions drive a Cloud Bigtable back-end and the bottom one is a mobile client app. Mobile clients interact with Firebase, and App Engine provides back-end processing.

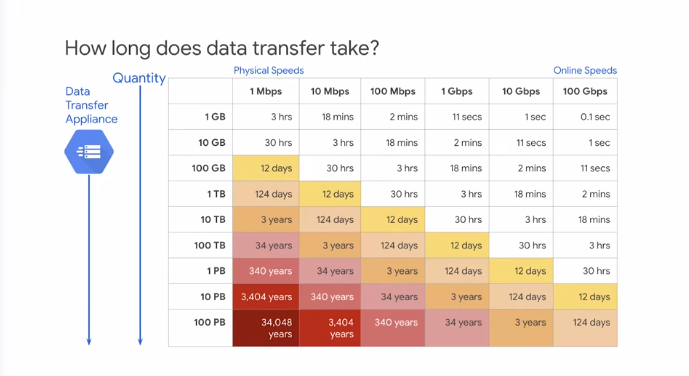
**Data transfer**



Global Google Cloud Network Integration with Google Cloud Storage, enables high bandwidth upload from anywhere. App Engine provides authentication and ingest coordination, and management between the network and storage. Upload is exposed to browsers through a web UI implemented on App Engine. Upload is provided for client applications via RESTful APIs exposed by Cloud Endpoints.



This is an example of a lazy deletion design. There are a number of layers of soft deletion, and delay between steps that should be tuned to the application and its business requirements. Data integrity is the quality of the data remaining accurate and accessible. If the data loses accessibility or accuracy, it's no longer reliable to the system, and the users, and therefore has lost integrity. Many kinds of failure including human error can lead to data loss. Strategy is defense in depth.

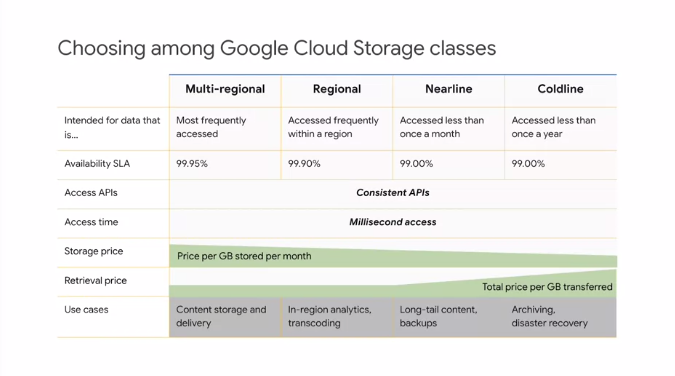


How long does it take to transfer data online? The left side of the table are closer to physical speeds, and the right side of the table is closer to online speeds. Therefore, it's much faster to accumulate data online and work with it and transfer it online, than to collect the data physically and then transfer it.

**Cloud Storage**

****

Cloud Storage, backup is often the easiest and first cloud application for many businesses. Archive is a natural step. And once the utility and value are understood, a common follow-on step is to reduce the cost and overhead of maintaining offline archive storage such as tape libraries by directly loading them into archival storage.

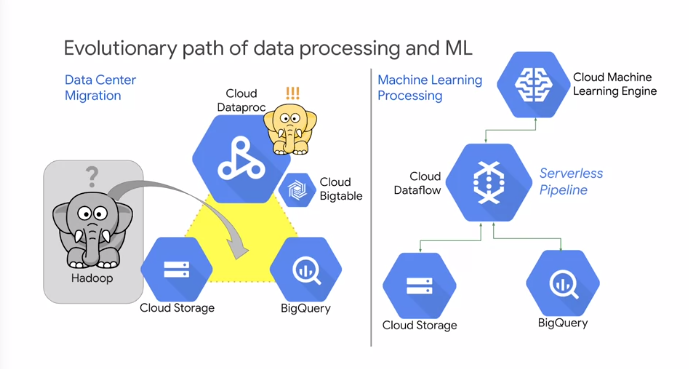


Cloud Storage offers four storage classes. Customers can associate each of their buckets with the storage class most appropriate for its use case. All of the storage classes are accessed in analogous ways using the Cloud Storage API, and all offer millisecond access times. All storage classes incur a cost per gigabyte of data storage per month, and egress and data transfer charges may apply. In addition to those charges, nearline storage also incurs an access fee per gigabyte of data read, and coldline storage incurs a higher fee per gigabyte of data read. See the Cloud Storage pricing page for more information.

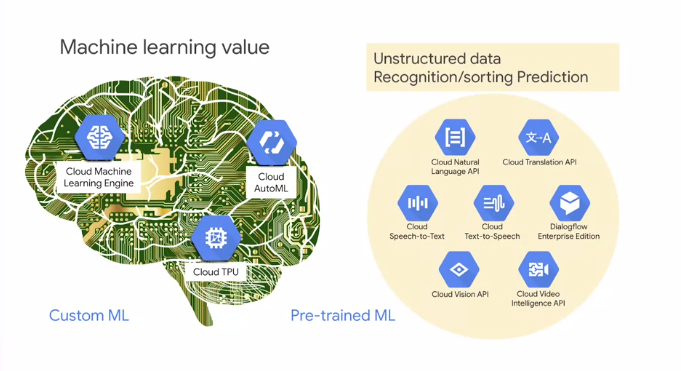


First, temporary data, maybe you want to use a local disk. Standard and regional HDDs and SSDs are now supported. Regional versions have zonal redundancy.

**Data processing to Machine Learning**

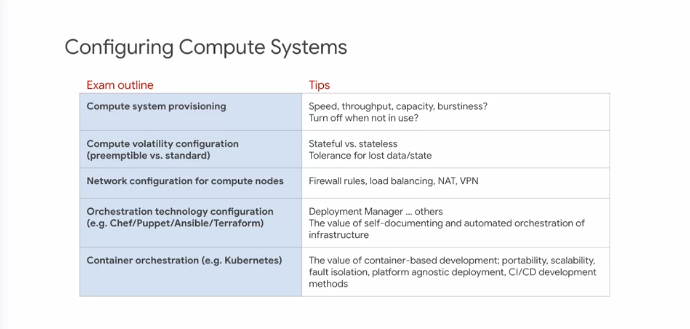
****

Data Center Migration. BigQuery provides a front-end for analysis and a back-end that can read from several sources including BigQuery tables, but also CSV files in Cloud Storage. Cloud Dataproc is a managed service for Hadoop clusters, useful for processing data and returning it to Cloud Storage or BigQuery. The first step is migration from Data Center processing to Cloud Data processing. BigQuery replaces many tools and custom applications in data center, while Cloud Dataproc replaces Hadoop. Cloud Bigtable is a drop in replacement for Hbase. Machine learning is available from Cloud Dataproc using APIs, such as natural language processing or NLP. When ready, the business can move from cluster-based managed service to a serverless service, and access the full benefits of machine learning. Machine learning provides value through tagging of unstructured data, which makes it useful for specific purposes. Machine learning can also be used to recognize items and for prediction.

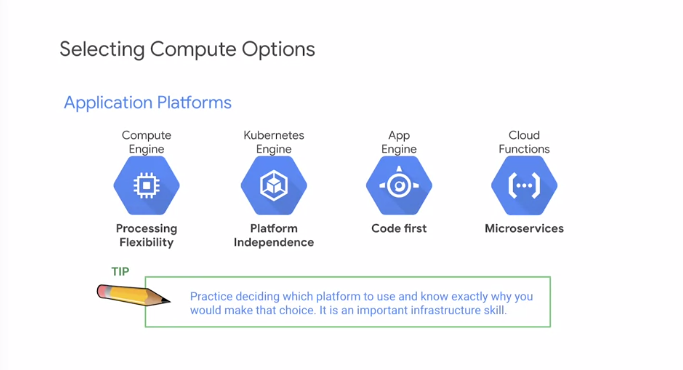


Machine learning is more of a focus of the data engineering track rather than the cloud architect track. But it's still part of the infrastructure of a cloud architect, and it might be used for finding solutions. So, you should be familiar with the services and what they do.

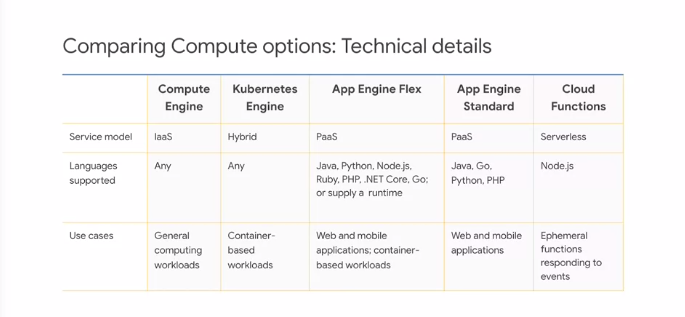
**Configuring Compute Systems**

****

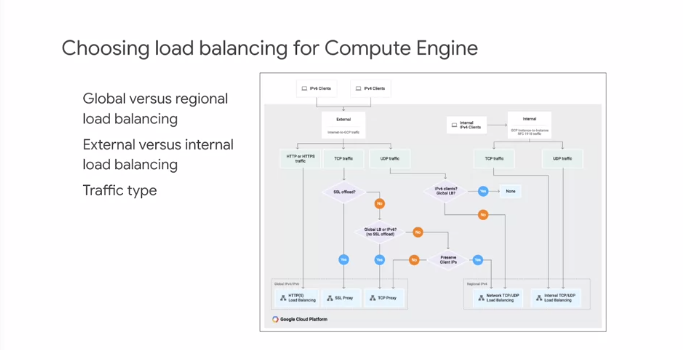
We've already covered many of these items. One thing to consider in the design is whether you can create an application that's tolerant of some amount of lost data or state and part of the system can simply drop some data or can store data externally and recover from drops. Then if that part is isolated, you can consider using Preemptible VMs for lower cost. By the way, do you remember what the maximum time is that you would have control of a Preemptible VM? It's 24 hours. If a VM hasn't been requested by then, it'll be pulled from service anyway. Development environments and disaster recovery are often good applications for creating infrastructure through automation technologies such as Deployment Manager or Terraform. In the development environment case, you can generate a clone of the production infrastructure solution or use by the development team. So the test team needs an environment, deploy another copy. Quality control needs and environment, another copy. Auditing and compliance test backup and recovery, create more deployments on demand.



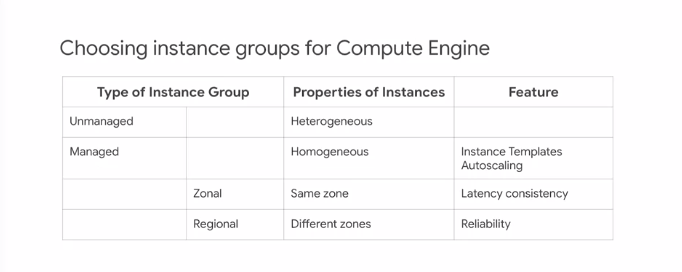
There's no perfect solution for all circumstances. But there are usually several options that can work and one that is optimal for a specific situation. In general, if your use case is application focused, App Engine is the fastest way to get something up and running with the least infrastructure overhead. Kubernetes is more complicated and places more responsibilities on the developer and the support staff. Fewer items in Kubernetes or automated when compared with App Engine. On the other hand, it offers controls over automation and Kubernetes can make the application platform independent which is easier for development and migration of support for different environments. Compute Engine is a VM. The software installation and maintenance gives far greater control and performance efficiency at the expense of IT overhead. Finally, Cloud functions provide a micro-services approach which means creating small stateless elements, very fast and scalable. But you have to be diligent about isolating state and there's additional programming overhead involved. You can also implement micro-services solutions on Kubernetes.



This is another one of those tables that you should know backwards and forwards. For example, if a solution involves Node.js, which compute service should immediately become a candidate for part of the solution? App Engine Flex and Cloud Functions both support Node.js.

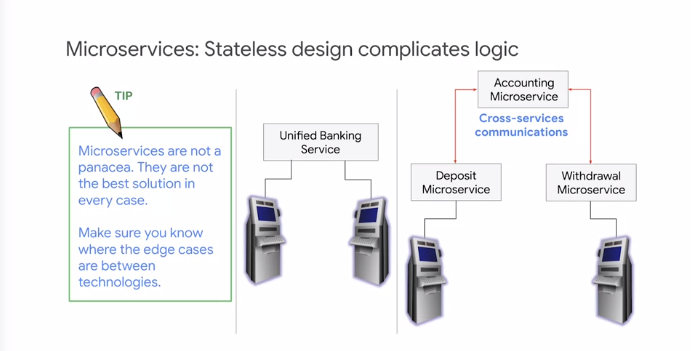


First of all, there's no such thing as a load balancer in Google Cloud. Allow me to explain. In Google Cloud, there is no load balancer because the function of distributing traffic is handled by the software defined network. So there are several kinds of load balancing. But these are just features that are part of the network, not physical devices. Load balancing services are distinguished by the kind of traffic they direct. By whether they're intended to balance traffic from one server to another inside the Google Cloud or if they're intended to direct data arriving from the Internet. Also load balancing can be global or applied to a specific region. Make sure you understand the basics of how geo-distributed and load balancing works. How can everyone go to the homepage for google.com and get great response time as if from a local server?

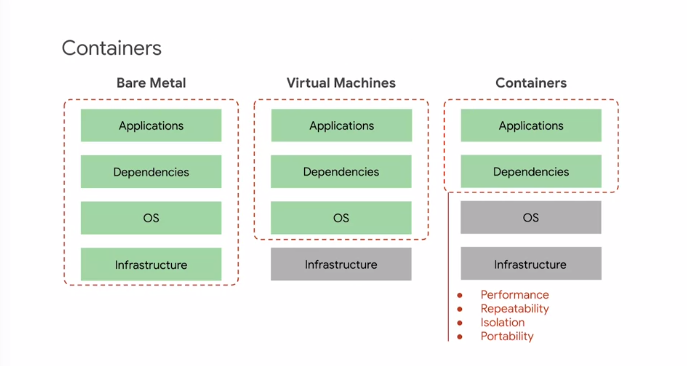


Unmanaged instance groups collect different kinds of instances. Usually, this is done for management of lift and shift existing designs and it's not recommended because it does not make the best use of the features available in cloud. Managed instance groups are all the same kinds of instances meaning that the type can be defined by an instance template and auto-scaling is available. Zonal managed instance groups keep all the instances in the same zone which is useful to provide consistent network location when the instances must communicate with similar latency and avoid zone to zone transfer. Regional managed instance groups distribute the instance and multiple zones within the region increasing reliability. Instance groups should be managed instance groups to make effective use of the cloud.

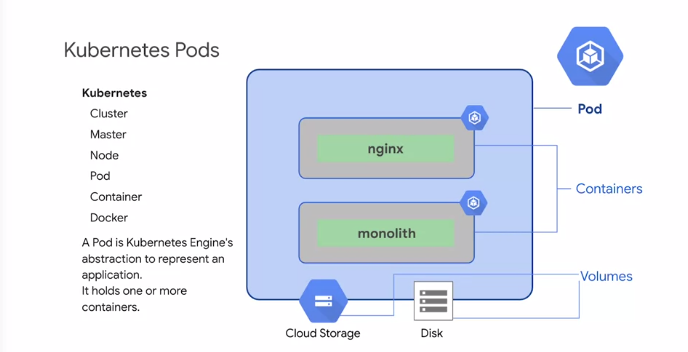
**Microservices, Containers, Data Processing, and IoT**

****

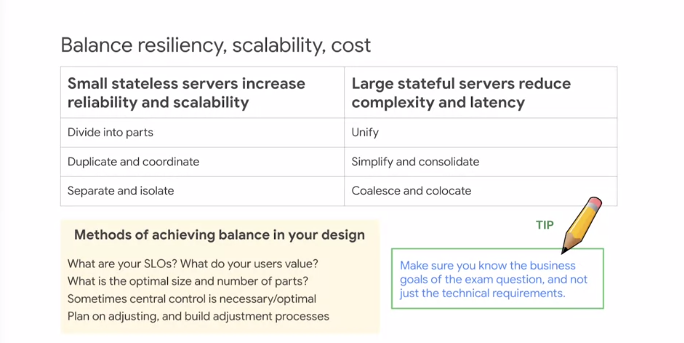
Microservices is not a panacea, it doesn't fit all cases. You can implement a microservices solution in App Engine cloud functions and using Node.js and Kubernetes. The platforms have overlapping coverage. Do you know when you might choose one platform over another for microservices solution? Coordinating a transaction across stateless microservices is tricky. You have to store the state externally and retrieve and use it in each function. Microservices architectures are commonly used and implemented in Cloud Functions or in App Engine.



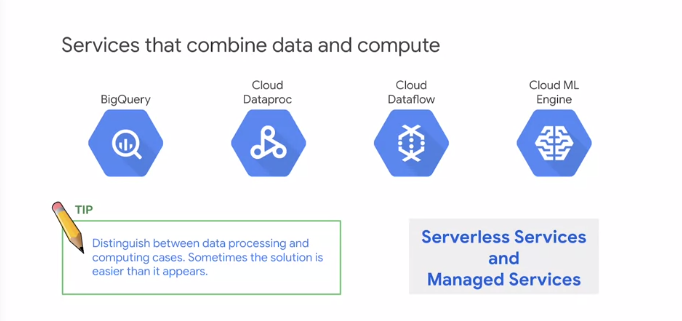
Containers bundle application code and dependencies into a single unit, abstracting the application from the infrastructure.



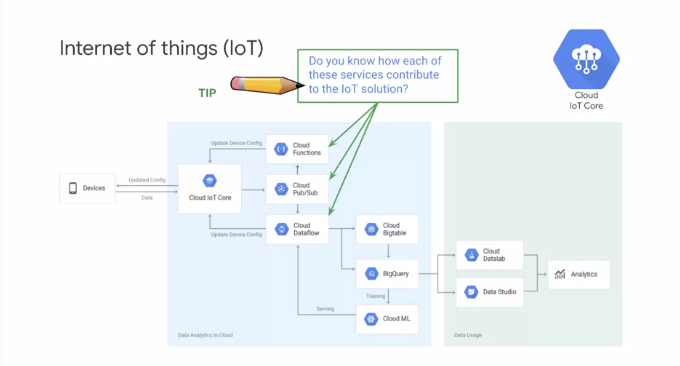
Make sure you know the difference between containers and VMs. Make sure you know the basics of Kubernetes. When might you use a different pod? How does code maintenance and migration to a new version work? Understand AB testing and deployment. A pod is Kubernetes engine abstraction to represent an application. It holds one or more containers. The containers in the pod share a single IP address and a single namespace. A pod can share other items such as access to storage. Any data access mattered on a pod called a volume is available to all containers in the pod. Containers that are part of the same pod are guaranteed to be scheduled together on the same VM and can share state via local volumes. Persistent volumes using persistent disks in Compute Engine survive instance and container restarts. How do you balance resiliency, scalability, and cost?



Well, small stateless servers are more scalable and reliable closer to a microservices designed. Of course, you have to coordinate all the messaging and state storage and retrieval. So the cost is increased complexity. Large stateful servers reduce application complexity. They allow you to combine activities and organize them making the application more manageable. However, this comes at a cost of decreased scalability and a greater chance that if something goes wrong it will take down the service. What you want to do is blend the approaches where it makes sense to the business. This is another case where what the client wants is what's most important to the design and on an exam it means being sensitive to and looking out for those trade-offs.



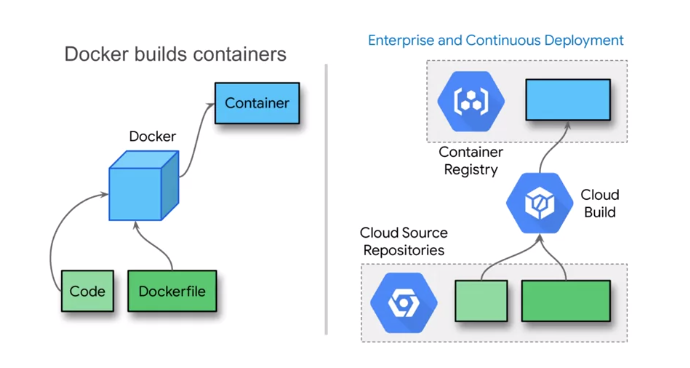
Services, managed services, and serverless services, knowing the differences between them will help distinguish solutions that meet all the requirements. A managed service gives you visibility to servers but limited control. You give up control for automation. It's great for popular use cases and eliminating overhead. Serverless services completely hide all servers. Generally, it's more fast scalable and efficient than you could create on your own. The key trade off their proprietary. To be fair, all services are sometimes called managed services even if they're really serverless services.



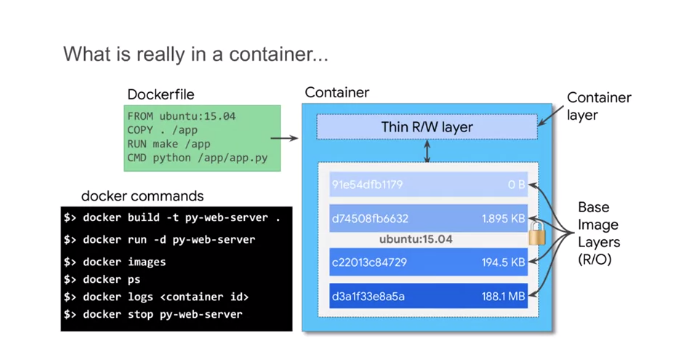
I mentioned previously that there are common assemblies of services that work together. If you review solutions and diagrams, you'll start to recognize some. There's a great resource for becoming familiar with a variety of solution architectures online. Look for it at cloud.google.com/solutions. The core assembly here is IoT Core, Cloud Functions, Cloud Pub Sub, and Cloud Dataflow. The IoT core device manager. Google Cloud IoT Core provides a fully managed service for device registration, authentication, authorization, metadata and configuration. Cloud Pub Sub. Google Cloud Pub Sub provides a globally durable message ingestion service. Cloud Pub Sub can act like a shock absorber and rate level or for incoming data. Cloud Pub Sub scales to handle data spikes that can occur when swarms of devices respond to events in the physical world. It buffers these spikes to help isolate them from applications monitoring their data. Cloud Dataflow. Google Cloud Dataflow provides the open Apache beam programming model as a managed service for processing data in multiple ways including batch, extract transform load ETL patterns, and continuous streaming patterns. Cloud Dataflow performs well with high volume data processing. Cloud Functions. IoT events and data can be sent to the cloud at a high rate and need to be processed quickly. Cloud functions allow you to write custom logic that can be applied to each event as it arrives. This can be used to trigger alerts, filter invalid data, or invoke other APIs. Cloud Functions can operate on each published event individually.

**Experiment: Containers and GKE video (Like/Dislike) ?**

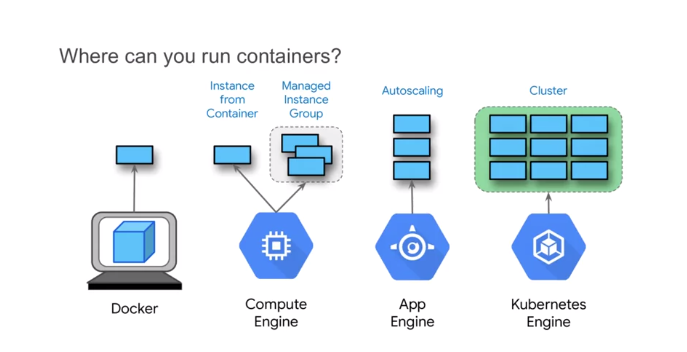
Between 2017 and 2018, the number of organizations using containers for software development had to deploy their services doubled. The trend shows no signs of slowing. For this reason, container knowledge and skill with Kubernetes is increasing the importance for the job of a Cloud architect. Of course, if you need more of these skills for the job, you will also need them to prepare for the exam. Docker is software that builds containers.



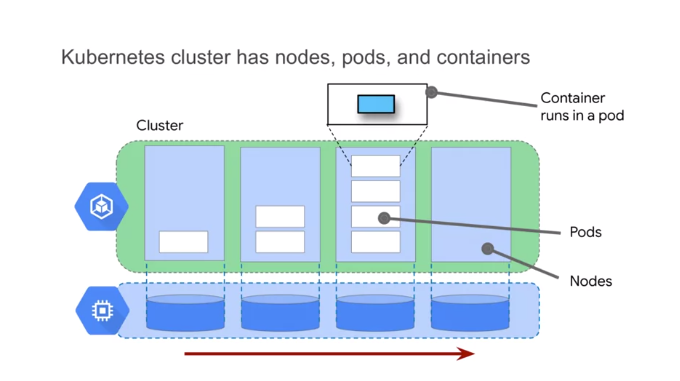
User apply application code and instructions called a Docker file, and Docker follows the instructions and assembles the code and dependencies into the container. Containers can be run much as an application can run. However, it is a self-contained environment that can run on many platforms. Google Cloud offers a service called Cloud Build which functions similarly to Docker. It accepts code and configuration and builds containers. Cloud Build offers many features and services that are geared towards professional development. It is designed to fit into a continuous development, continuous deployment workflow. It is designed to scale and to handle many application developers working on, and continuously updating a live global service. If you had 100 developers sharing source files, you would need a system for managing them, for tracking them, versioning them, and enforcing a checking review and approval process. Cloud Source Repositories is a cloud-based solution. If you were deploying hundreds of containers, you would not be in keeping it to yourself. One of the reasons to use containers is to share them with others. So you need a way to manage and share them. This is the purpose of container registry. Container registry has various integrations with continuous integration, continuous deployment services. A Docker container is an image built-in layers. Each layer is created by an instruction in the Docker file.



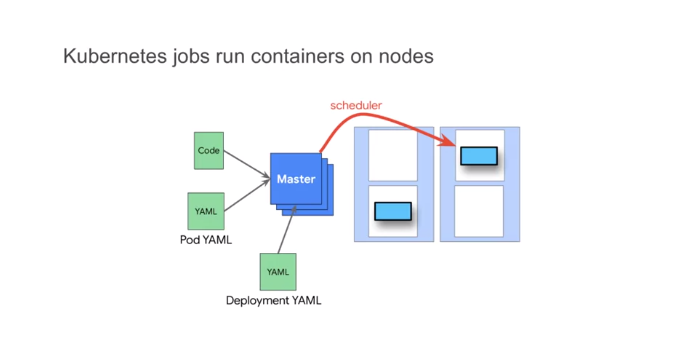
All the layers except for the top one are air locked. The thin read-write layer at the top is where you can make changes to a running container. For example, if you needed to change a file, those changes would be written here. The layer designed inside of a container isolates functions. This is what makes the container stable and portable. Here are a few of the common Docker commands. The docker build command creates the container image. The docker run command runs the container. There are other Docker commands that can help you list images, check the status of a running container, work with logs or stop a running container.



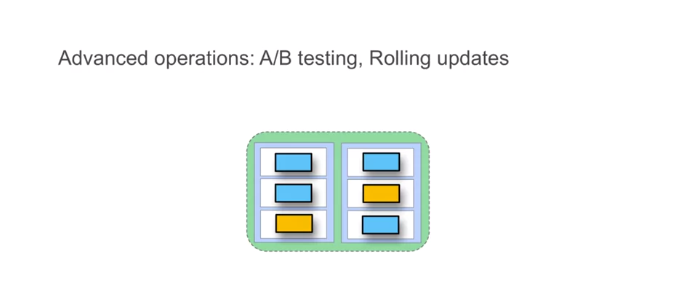
You can run a container in Docker itself, as you saw with the docker run command. You can also run containers using Compute Engine. Compute Engine gives you the alternative to start up a virtual machine from a container, rather than from an OS Image Boot Disk. You also have this option when creating an instance template, which means you can create managed instance groups from containers. App Engine supports containers as custom runtimes. The main difference between the App Engine standard environment and the App Engine flexible environment is that flexible hosts applications in Docker containers. It creates Docker containers and persists them in Container Registry. A Container Orchestrator is a full service for managing, running and monitoring containers. Both App Engine flexible environment and Google Kubernetes engine are container orchestrators. Kubernetes is an open standard software. So you can run a Kubernetes cluster in your data center. Google Kubernetes engine provides Kubernetes as a managed service.



A Kubernetes cluster is composed of nodes, which are a unit of hardware resources. Nodes in GKA are implemented as VMs in Compute Engine. Each node has pods. Pods are resource management units. A pod is how Kubernetes controls and manages resources needed by applications and how it executes code. Pods also give the system fine grain control over scaling. Each pod host, manages, and runs one or more containers. The containers in a pod share networking and storage. So typically, there is one container per pod, unless the containers hold closely related applications. For example, a second container might contain a logging system for the application in the first container. A pod can be moved from one node to another without reconfiguring or rebuilding anything. This design enables advanced controls and operations that gives systems built on Kubernetes unique qualities.

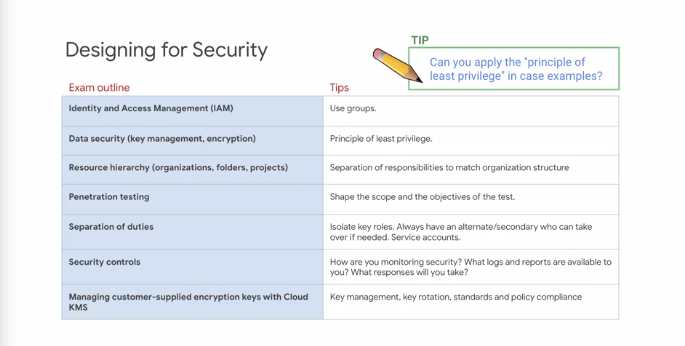


Each cluster has a master node that determines what happens on the cluster. There are usually at least three of them for availability, and they can be located across zones. A Kubernetes job makes changes to the cluster. For example, a pod YAML file provides the information to start up and run a pod on a node. If for some reason a pod stops running or a node is lost, the pod will not automatically be replaced. The deployment YAML tells Kubernetes how many pods you want running. So the Kubernetes deployment is what keeps a number of pods running. The deployment YAML also defines a replica set, which has how many copies of a container you want running. The Kubernetes scheduler determines on which node and in which pod the replica containers are to be run. One of the advanced things that Kubernetes deployments allow you to do is roll out software to some pods and not others. So you can actually keep version in production on most of the pods and try out version B with a sample group and other pods.

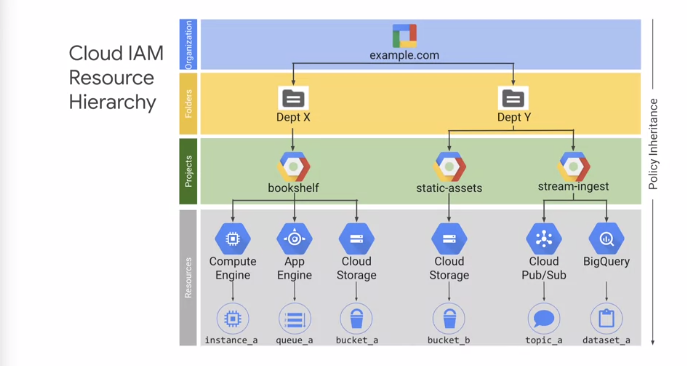


This is called A/B testing, and it is great because you can test the new software in a real production environment without risking the integrity of the entire service. Another thing you can do with deployments is a rolling update. Basically, you load up the new software in a replacement pod, switch the load to the new pod, and turn down the old one. This allows you to perform a controlled and gradual roll out of the new software across the service. If something goes wrong, you can detect the problem and roll back to the previous software. Really, if you are going to run an enterprise production service you will need these kinds of operations. That is one major reason to adopt Kubernetes. There are a number of subjects that were not covered in this brief overview. For example, how containers running in the same pod can share resources, how containers running in different pods can communicate, and how networking is handled between a node's IP and the applications. These subjects and more are covered in the course, getting started with Google Kubernetes engine or you can find more information in the online documentation.

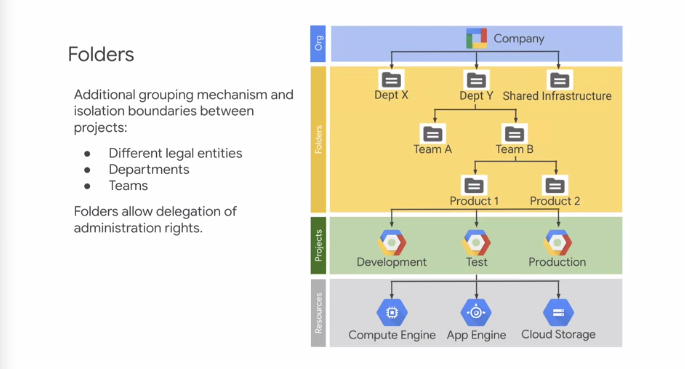
**Designing for Security**

****

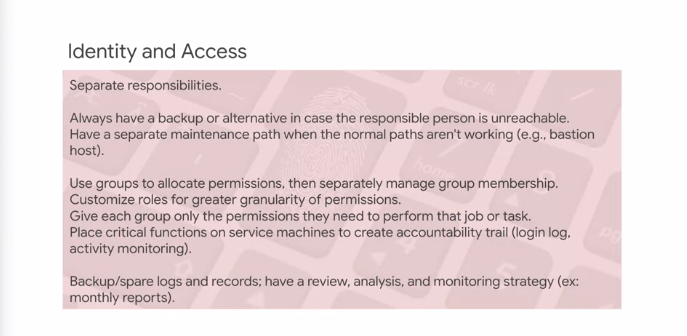
One key to securing access is to request and established groups that represent roles. Then apply the permissions to the groups, and allow the people in the organization who manage identity to assign membership to the groups. This creates a clean interface between permission management on the cloud side, and group membership on the personnel IT side. Another key to security, is to craft security permissions. The standard roles are defined for the most common use cases, but you might want to derive more granular and restricted roles by customizing them. Service accounts are a great way to separate system components, and established secure communications between components. A bastion host is a way to leverage a service account. For risky and uncommon actions, make the user admin startup and log into bastion host. From there they can borrow the service account assigned to the host to perform restricted functions. One benefit is that the login process generates logs for accountability.



A policy is set on a resorts, and each policy contains a set of rules enroll members. Resources inherit policies from parents. So, a policy can be set on a resource for example, a service, and another policy can be set on a parents such as a project that contains that service. The final policy is the union of the parent policy and the resource policy. What happens when these two policies are in conflict? What if the policy on the resource only gives access to a single cloud storage bucket, and restricts access to all other buckets? However, at the project level, a rule exists that grants access to all buckets in the project. Which rule wins? The more restrictive rule on the resource or the moral general role on the project? If the parent policy is less restrictive, it overrides a more restrictive resource policy. So, in this case, the project policy wins.

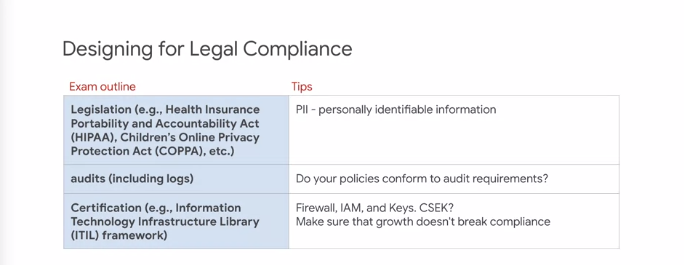


Folders map well to organization structure. It's a way to isolate organizations or users or products while still having them share billing and corporate resources.

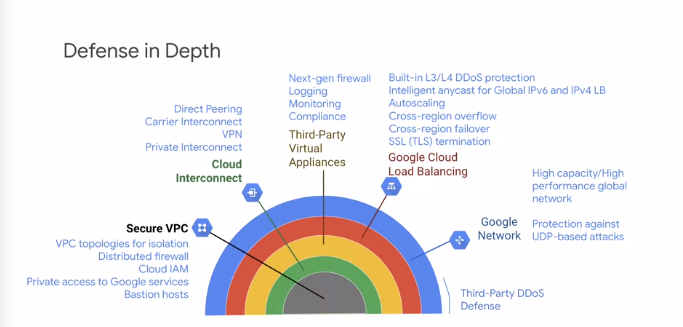


Commit a security checklist to memory. Sometimes just running down a list will rapidly identify a solution.

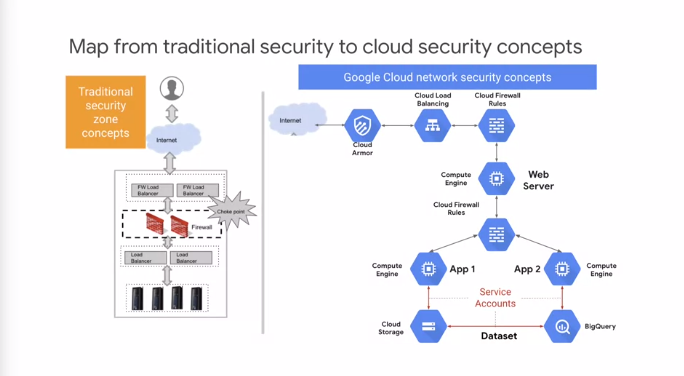
**Designing for Legal Compliance**

****

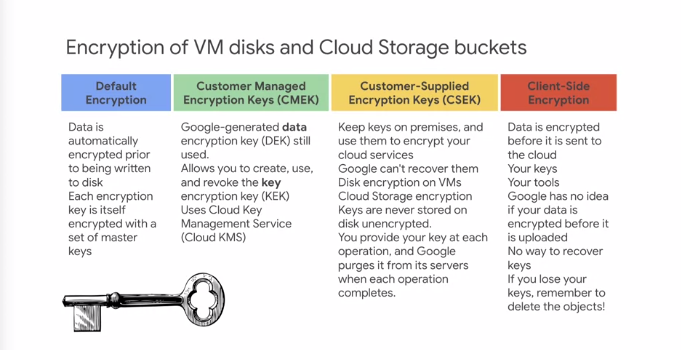
What are the two most common compliance areas? Privacy regulations such as HIPAA and GPDR, and commercial and live business standards such as PCI DSS.



Google Network has layers of protection. Each layer protects and compliments the next internal layer. The main thing to know is that Google handles security up to a point, after that, the security is up to you. So, you need to know where your responsibilities begin. Secure VPC; identify optimal VPC topology, deploy distributed firewalls, control access with IAM permissions, avoid adding public IPs to instances. Access Google services internally. Cloud Interconnect, connect securely to on-prem or other cloud deployments, Private Interconnect, Carrier Interconnect, Direct Peering, VPN. Third-Party Virtual Appliances, enhance VPC security with third party appliances, Next-gen firewalls, IDS slash IPS, that's intrusion detection. Logging, monitoring, scale third-party appliances using internal load balancing so you don't create choke points in your VPC. Global Cloud Load Balancing provides edge protection and global infrastructure protection for IPv4 and IPv6. Layer three and layer four, denial of service protections. Anycast IP even if backends are in multiple regions to absorb a tax for resiliency, auto-scaling, cross-region, overflow, and cross-region failover. Google network; high-capacity, high-performance, software defined network, virtualization global networks with subnets, organizations, folders, cross-project networking, peering. Third-Party DDoS, you can complement the infrastructure with additional security from third-party providers. Here's some key concepts: Cloud Armor, Cloud Load Balancing, Cloud Firewall Rules, Service Accounts, separation into front-end and back-end, isolation of resources using separate service accounts between devices.

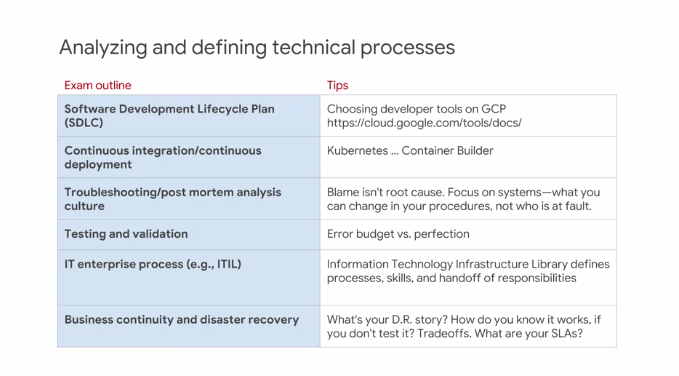


Because of pervasive availability of firewall rules, you don't have to install a router in the network at a particular location to get firewall protection. That means you can layer the firewalls as shown in this example, because of pervasive support for Service Accounts you can lock down connections between components. When faced with a security question on an exam or in practice, determine which of the specific technologies or services is being discussed: Authentication, encryption for example, then determine exactly what the goals are for sufficient security. Is it deterrence? Is it meeting a standard for compliance? Is the goal to eliminate a particular risk or vulnerability? This will help you define a scope of a solution whether it's on an exam or in a real-world application.



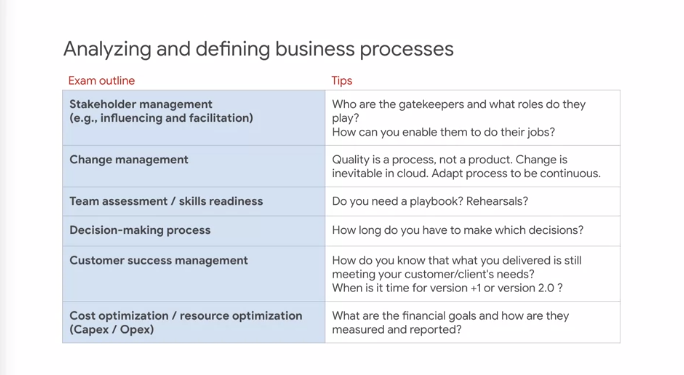
GCP provides several encryption options. Customer Managed encryption keys CMEK, using Cloud KMS. When you use Cloud Dataproc, cluster and job data is stored on persistent disks associated with the Compute Engine VMs in your cluster, and in a Cloud Storage bucket. The persistent disk and bucket data is encrypted using a Google-generated data encryption key called a DEK and a key encryption key called a KEK. The CMEK feature allows you to create use and revoke the key encryption key, the KEK. Google still controls the data encryption key or the DEK. Default encryption, encryption at rest uses the key management system KMS to generate KEKs and DEKs. The Key Management Service KMS allows you to generate AES-256 keys. You can use these values off Cloud. The service also handles key rotation and when a file is destroyed there is a 24-hour delay before final deletion.

**Analyzing and defining technical processes**

****

There are a number of items to review on this slide. Two of them are related and important for you to know. The first is the concept of an error budget. Some companies arbitrarily make a goal of 100 percent like 100 percent up-time but that isn't realistic. We exist in a quantum mechanical universe where random events happen. So, 100 percent isn't really physically possible most of the time. So, at Google, we understand that some amount of time will be spent in error. What we do is identify and manage that time. So, if you have 99 percent up-time, then you have an error budget of one percent. If you find that you've not used all one percent of the error budget during that period, then it's time to do some things that are potentially disruptive. That leads me to a second point and that is about backup and disaster recovery. We often say people don't care about backup, they care about restore. But how do you know that the restore processes are working if you don't try them? If you did a backup last year and you've been consistently generating backups since that time, what are the chances the restore process will work? That's a good thing to do with that extra error budget. It's handy to have a testing checklist in mind to help you consider all options. Consider the questions you're trying to answer with testing. Will the solution support the number of users? Will it handle peak traffic? Is latency acceptable? And so forth. The test environment should resemble production as closely as possible. If you can, test on a part of the production service during a low-use time such as at night. That's called a Dark Launch. If you can't do that, test in pre-production using a synthetic workload that closely resembles a real workload. The results could be misleading if the workload is not designed well. The pricing calculator is very handy for comparing different configurations and identifying cost-effective alternatives. The pricing calculator can be used with BigQuery to estimate the cost of a query before you submit it. The basic advice for optimizing VM cost is use the right size VM and the right resources, customize if necessary. You can use what-if scenarios to see how changing the design can influence cost. For example, which is more cost effective, four machines with eight CPUs or eight machines with four CPUs or 32 machines with one CPU? The GCP console gives you price estimates in the interface when you're configuring the instance. Preemptible VMs can be a great way to scale out. The important thing to remember is that the application has to be designed to handle the loss of any of the preemptible workers at any time. There are committed use discounts and sustained use discounts. Sustained use is when you use the same kind of instances in the same location, and an automatic discount kicks in. Committed use discounts is where you reserve resources and commit to using them in advance at a discounted rate. Discounting algorithms are subject to change. Please see current discounting details in the online documentation. Optimizing disk cost has to do with two factors, size and performance. If you over-allocate disk, you'll be paying for storage capacity that you're not using. It's a much better idea to offload data to cloud storage so you're paying for what you use, rather than holding disk capacity that might not be used. Disk performance can be complicated but there are four factors to consider. The frequency of reads, the size of reads, the frequency of rights, and the size of rights. Generally, smaller more frequent reads and writes are less performance than longer and less frequent ones. Remember that read and write performance are usually not symmetric. Also, consider using a cash if the usage pattern involves a lot of repeated reads. Egress is free, networking cost are similar per GCP product but are build per product. So, you need to view the pricing documentation per product for the details. Example, cloud storage has standard egress cost but there are also separate charges for data migration and for cloud storage operations. Here's an example, egress between regions might be one cent per gigabyte, egress to the internet. The first terabyte to the world destinations might be 12 cents per gigabyte. For the example, I wouldn't expect to know the exact cost of egress from zone to zone or zone to Internet. But I would anticipate the need to know what activities are charged and generally, which actions are more or less expensive than other actions. For example, in a disaster recovery scenario, you'd want to recognize that the improved isolation of storing data in a separate region will be more expensive than just storing the data in a different zone in the same region.

**Analyzing and defining business processes**

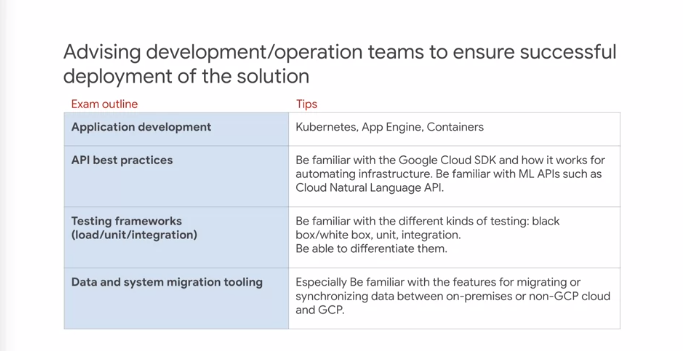
****

Analyzing and defining business processes is covered in our design and process courses. Let's expand on a couple of these issues. In the change management outline item, there's a tip that says, quality is a process not a product. As a working cloud architect, you'll almost never have a job where you design and implement the technical solution and then you're done. Instead, you'll be required to stay on the project for a period after implementation and launch, to make sure that solution continues to run and stabilizes. Anticipating that, you'll want to develop process checks, and operational knobs to ensure that the solution can be monitored and adjusted during the stabilization period. So, the business processes might look like generating reports, and it might include weekly or monthly meetings to analyze the reports. It might include procedures that explains to administrators how to take action. Another item is customer success management. Only in the past few years has customer success been broken out from support. The difference is that support is there to make sure a solution continues to operate as it was designed and built. But what if the business circumstances change, or the technology changes, and the current solution is starting to drift off from the actual business needs? Customer success is about making sure that the solution continues to evolve, and remains effective and efficient for the current requirements, and uses the latest and most efficient technologies and methods. If you can exchange a server oriented architecture for a serverless service, you no longer have to be concerned about instance overhead, just the SLAs of the service. There are different hardware, CPU architecture supported in different zones. For example Sandy Bridge Aswell, Broad Well, Ivy Bridge, Skylight and so forth. There are a lot of benchmark comparisons online, and there open source CPU measurement tools that you can use. It suggested that you test your application and workload in different zones to see what differences the hardware in the zone might make. There are new persistent disk features and options released periodically. So, check the documentation online for the latest figures and details. People often assume that a persistent disk is just a hard disk, when it has different features and capabilities. Consider potential I/O burst, if you've planned on I/O based on an average, and the actual disk usage is bursty, the disk could be under provision for dealing with the bursts. Persistent disk performance scales with the size of the disk. So, if you trade up to a larger disk in your design, revisit the performance to avoid overcapacity. If you trade disk size down, check for under capacity. Potential I/Os may be constrained by CPU. An n1-standard-4 can drive a PD-SSD at capacity, and an n1-standard-16 can drive a local SSD at capacity. There are open source disk measurement tools available, and it's recommended that you run tests on your application at various sizes of data and loads, to understand not just capacity, but how the solution handle stress and overload conditions. In general, internal IPs are faster than external IPs. Even VM to VM in the same zone over external IPs can be about one gigabit per second versus 8.5 gigabits per second for internal IPs. Here's another tip. Does a default instance in Google Compute Engine know the difference between an internal and an external IP? No, it doesn't. Standard instances have one interface, and all traffic arrives on that interface. So, that means that external IP isn't added to the hardware interface to the local IP. For sizing network capacity, consider potential I/O burst. If you've planned on I/Os based on an average, and the actual traffic is bursty, it could be underprovision. Network capacity scales with the number of cores. So, if you change out the number of cores and the VM's in your design, revisit the network capacity to make sure the design does not over or under provision. What are some of the factors you should consider when estimating workload? Most common of course are the characteristics of communication and messaging, how often a request or transactions or operations performed, and how big is the payload of the request. Other factors include state changes and methods that divide work into parts, such as sharding, distributing work to multiple workers, such as pipelines, or aggregate work for efficiency, such as batching.

**Developing procedures to test resilience**

In this section, we'll discuss developing testing procedures. You can't test everything, so you need to consider what items can act as indicators. How do you prove that the solution is working properly? How do you know if the solution is highly available or scalable? This design illustrates that in some cases, technical design is not sufficient, but must be followed up with human procedures. The original design was intended to handle a maximum of 1000 queries per second. It was originally a functional design. However, over time, the load has grown. While the system is still operational, if one of the front end servers were to fail and the combined traffic was taken up by the other front end, the total would be 1,200 queries per second, and would be above capacity. For this reason, resiliency requires identifying key metrics, in this case total load, and periodically adjusting capacity to stay ahead of growth. On the other side, if load were diminishing consistently over time, there could be cost savings in downsizing the front end servers. A common misunderstanding is forgetting that auto-scaling follows the law of diminishing returns. Imagine that you're considering the target CPU utilization for the entire group of VMs, the percentage utilization that an additional VM contributes depends on the size of the group. The fourth VM added to a group offers 25 percent increase in capacity to the group. The tenth VM attitude group only offers 10 percent more capacity, even though the VMs are the same size. In this example, removing one VM doesn't get close enough to the target of 75 percent. Removing a second VM would exceed the target. The auto-scaler behaves conservatively, so it will shut down one VM rather than two VMs. It would prefer under utilization over running out of resource when it's needed. Now, here's a tip. Consider using Stackdriver custom metrics for auto-scaling. The reason is that CPU utilization is rarely a good measure of customer experience. A custom metric can enable auto-scaling on a more meaningful value. For example, a game service might scale with the number of players, which might be more directly related to application performance than something like CPU utilization.

**Advising development operation teams**



Most of the items in the exam outline have been covered already in another context.

The first rule of testing is that you can't test everything so you need to make some decisions. Unit testing focuses on individual functional units, for example, exercising an API. In some development environments, it's common for the original software developer to provide a testing application that exercises the API and validates that it's working as expected. Integration testing has to do with putting parts together and testing them as an assembly. Sometimes the individual parts can pass unit test because each is working as designed, but when the units are assembled they may not be compatible. You can also discover timing issues called race conditions during integration testing. One good piece of advice is to create a launch checklist. In this example, there are dependencies, capacities, single points of failure, security and access, and a phased roll out plan. With all those items to be checked and some of them being very complex, it's easy to see the value of using an organized approach to ensuring that everything's ready. General advice about release management? Well, automate everything you can. Also, instead of creating a process with a resource bottleneck which can slow down release, consider implementing a self-service approach. Let the lead developers or the product managers perform the release using the tools. Reliability and consistency are the keys to making release work well. Also, implement access control over critical release features and processes. For example, a team lead or a tech lead might be a member of the release group, and have special access.

When we think about capacity planning for launch, it's common to create a moon shot event where everything has to come together perfectly at a single moment for the launch to succeed. Consider instead using a phased approach, by launching first to a smaller market. The service can generate feedback and even warn of issues that might not scale in subsequent phases.

A classic example of this was when the first Pokemon Go game was launched. It was launched first in Japan. The game was so popular that it had scaling issues because the demand was much greater than the anticipated demand for which the service was designed. Fortunately, launches in Europe, the US and other locations were separated by a few days. Staging the launch gave the team the time needed to understand the scaling issue and redesign and reimplement the service before its second launch. I'm pretty sure you know this already. There are three ways to interact with Google Cloud, first is GCP Console. Second is the tiny virtual machine that's started up inside Console called Cloud Shell. One thing that makes Cloud Shell useful is it's authorized in the project, and it has the Cloud SDK tools including gcloud, gsutil and bq installed. You can also install the Cloud SDK outside of Google Cloud on a local computer or VM.

**Ensuring solution and operations reliability**

Ensuring solution and operations reliability. How do you ensure that a solution is reliable? Part of it occurs in the design. Making sure that common changes like increased traffic are handled in elastic ways. However, part of it is also in planning to monitor the service and to notice and respond to unplanned events. Some of those activities require human intelligence. For this reason, operations reliability spans both the technical and the procedural domains. Site reliability or SRE, is Google's approach to DevOps. It's a very comprehensive system that involves changing the culture about how maintenance occurs. One central idea is the division of aspects of operations into separate layers for clarity. Here's a tip, you ought to know something about each of these layers and most importantly, you should be able to distinguish between the layers. For example, monitoring is not incident response. They're related. Do you know what features relates them? It's alerts. A Stackdriver alert is triggered by monitoring and begins incident response, which is composed mainly of procedures. Qualities are often where our goals start, but figuring out how to measure them quantitatively enables data-driven operations. It can be difficult to figure out exactly what to measure because sometimes what's easily measured is not a good indicator of customer interests. Speaking of alerts, at Google, we have the concept of alerting for the right reason. Often, alerts are designed to signify some metric passing some limit. But the question is whether that metric or trigger is something the customer cares about or not. We need to alert on some technical measures. But if there's something that is directly causing the customer frustration and upset, that should also be an alert or perhaps replace a more technical alert. Make sure you know the difference between blackbox monitoring and whitebox monitoring. Blackbox monitoring and whitebox monitoring are frequently misunderstood. In the cloud architect contexts, the difference has to do with the assumptions you can make when designing your monitoring framework. In blackbox monitoring, you're not supposed to know or think about the inner workings of the application. All you can see is the user interface or the API interface. So, the only assumptions you're allowed to make have to do with these interactions. Blackbox monitoring is very good for validating user experience. You end up monitoring things like latency between request and response. In whitebox monitoring, the application is assumed to be known to you. The inner workings of the application are transparent. So, you can use that special knowledge when defining the test. A good example would be if you knew that under certain conditions a critical resource will get oversubscribed and you've designed the system from resiliency. In this case, you might flood the interface to trigger the state as if the service was under attack to see if the resiliency worked as expected. That's whitebox monitoring, where the tests can be focused on inner workings and not just the UI. In practice of course, you need both kinds. Here's an example, CPU utilization may or may not indicate user satisfaction. Round-trip delay or frequency of request errors might be a better measure of the user's experience. What metrics are you using? Can you define metrics that relate directly to user experience and service objectives? What are the watermarks or alert levels at which human processes are engaged? How are you setting those values? When do they need to be revisited and updated? How do you know they're related to important events? Know how to use trace and debug. Examples of other tools that Stackdriver replaces. Note that it's not just a collection of alternate tools that's the issue, but how you use them together. The individual tools are not integrated or designed to work together. So, a lot of manual procedures and translation massaging of data are required to use them together. With Stackdriver, the integration is by design. So, that work disappears. Stackdriver is also multi-cloud, able to manage projects across GCP and AWS. Another useful idea is that people don't plan to fail, they fail to plan. Another way of saying this is, the only time we have to prepare for emergencies is before they happen. Once the emergency is occurring, it's too late to prepare. You can design a great technical solution, but if it doesn't include human processes, then it might not be adaptive and resilient. Easy buttons are tools and processes that automate common actions. A playbook is a list of what to do when. So, here's a general rule; for every alert you should have a play in the playbook. What are the differences between a dashboard, an alert, and incident response? A dashboard is a display for monitoring a system. It's commonly tailored to the application. An alert occurs when a condition is met such as a metric crossing above a particular value for a given duration. The alert is the notification and alert could just be a warning or it could be a notification of an incident that needs to be handled immediately. Incident response consists of the steps you would take when a problem occurs. This might be written up in a playbook. Find a lab such as Quick Labs lab that uses logging and trace and debug to identify and solve an application problem. This will give you a sense of the value and how these components work together. There's a lab like this in the architect in GCP infrastructure class. Google's approach focuses on transparency, on involving the customer in the solution and blamelessness. Assigning blame establishes root cause with a person or an organization instead of getting to the real technical or procedural issue so that it can be fixed. If blame has been assigned, there's a high likelihood that the process has been prematurely suspended without really addressing the problem. What are the people supposed to do? What decisions or actions are they supposed to make or take? Are these documented? As mentioned, the metrics are not sufficient without the meeting to review the metrics, to evaluate them and make decisions and take actions. In those cases where timing is critical, you'll want to playbook and easy buttons supporting automation to increase the speed and consistency of incident response. Here's another tip, when something goes wrong with the cloud resource, give yourself or your team a limited period of time to solve it. For example, if a VM starts behaving incorrectly, see if it's something that's easily fixed. Then spare the VM to the side and replace it. Perform your diagnostics and debugging after the instance is replaced.