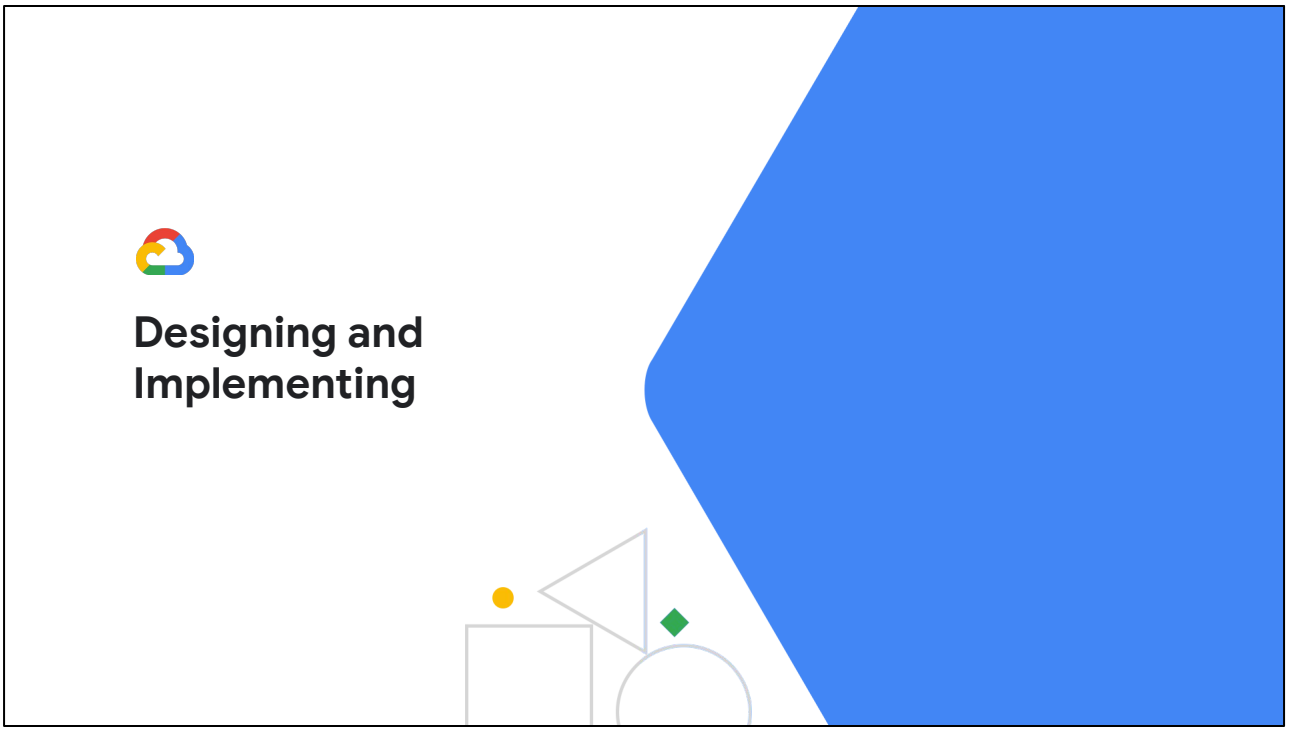

Designing and
Implementing

Course agenda

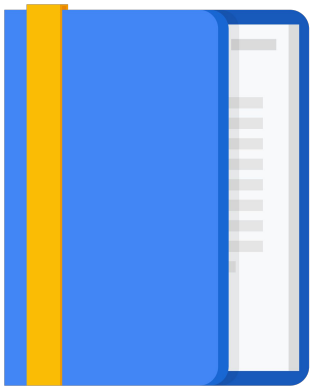
Module 1: Understanding the Professional Cloud Architect Certification

Module 2: Sample Case Studies for the Professional Cloud Architect Exam

Module 3: Designing and Implementing (Review and Preparation Tips)

Module 4: Optimizing and Operating (Review and Preparation Tips)

Module 5: Resources and Next Steps



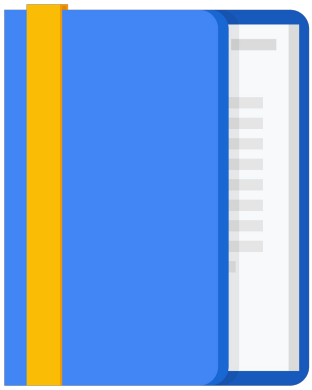
Google Cloud

Module agenda

Designing and Planning a Cloud Solution Architecture

Managing and Provisioning Solution Infrastructure

Containers and Google Kubernetes Engine



Google Cloud

Today you will be learning about and preparing for the Professional Cloud Architect exam.

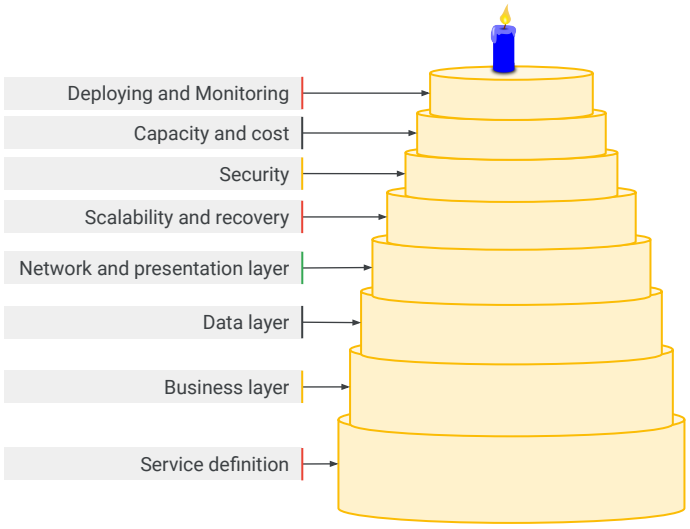
A lot of that has to do with design.

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.

Easy as … cake

TIP

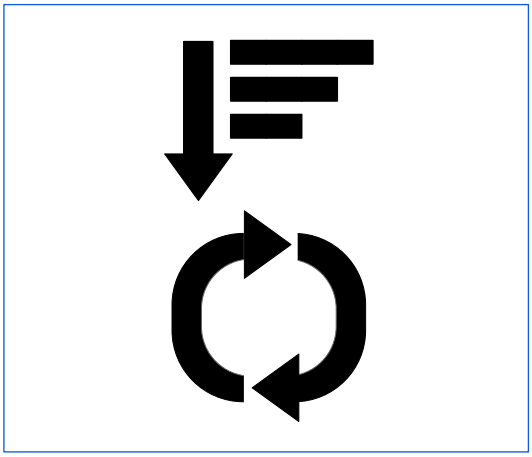
You will want to have key knowledge and skills at every layer.



Google Cloud

TIP: Use a layered model like this one. It will help you organize your thinking about each exam question so that you will more easily recognize and focus on what is important.

Divide and iterate



Google Cloud

TIP: 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 in 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 kind of familiarity takes time and practice to develop. Until you develop your own familiar assemblies, it is recommended that you use the model to consider changes in design from each layer, and iterate to make sure you've covered everything. It is better to be careful with design and follow a structured approach than to accidentally introduce a design flaw."

Divide and iterate. Use the layers to manage complexity.

It is too easy to get confused by the multiple qualities of multiple products.

For example, change from Cloud SQL to Datastore. Okay -- probably a data reason for that.

But what just happened to Security? What happened to your Monitoring plan? What

happened to your maintenance procedures? What happened to scalability and to recovery plans?

Designing a solution infrastructure that meets business requirements

Exam outline	Tips
Business use cases and product strategy	What does success look like qualitatively?
Cost optimization	A separate step after business logic criteria are met
Supporting the application design	What procedures and processes will be needed?
Integration	Working with existing systems in parallel at first? Are there parts that must remain on premises?
Movement of data	How and when will the data migrate to the new solution?
Tradeoffs	Priorities. Timing can be critical to value.
Build, buy, or modify	Control vs. overhead vs. time
Success measurements	What does success look like quantitatively?

Google Cloud

Business requirements. What are the customer's needs and expectations?

TIP: What is being asked usually going to be something that would be asked by a business stakeholder or that you would need to be able to answer on the job.

You will 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 is very important to be explicit about exactly what you are trying to achieve. These items are often stated qualitatively at the beginning and are measurable and quantitative at the end.

It is extremely difficult to optimize for success criteria and minimize costs at the same time. For this reason, it is recommended that you divide the activity into function, first, and cost optimization, second.

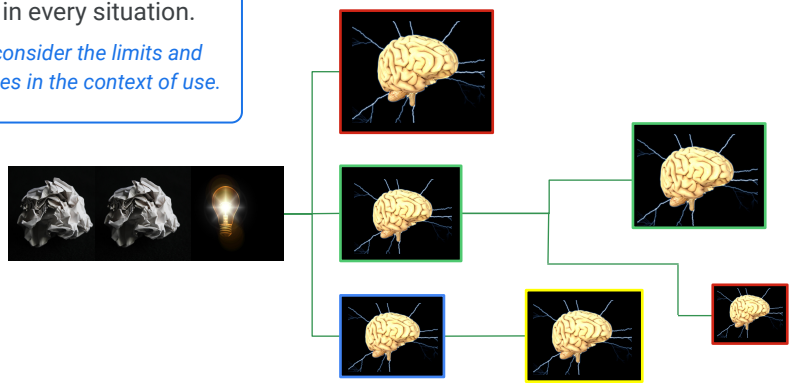
Another suggestion is to keep existing systems working in parallel for a period of time and verify that the new system is behaving as expected.

All solutions are contextual solutions

TIP

Even the best design patterns won't work in every situation.

You have to consider the limits and possible issues in the context of use.

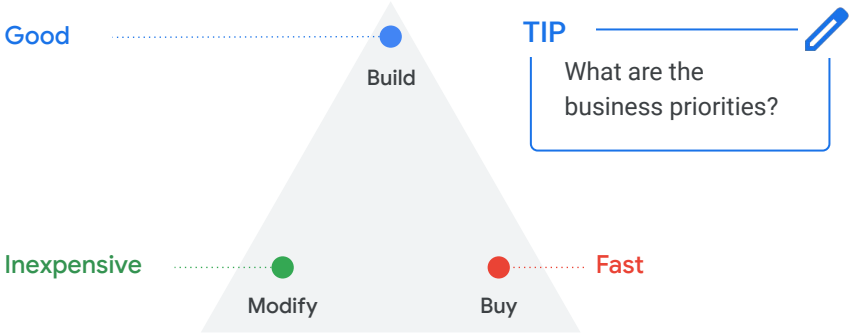


Google Cloud

TIP: 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. As you learn about design patterns, take note of their best uses and their limitations and issues so you can consider the design in a realistic context. 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.

Build, buy, or modify



Google Cloud

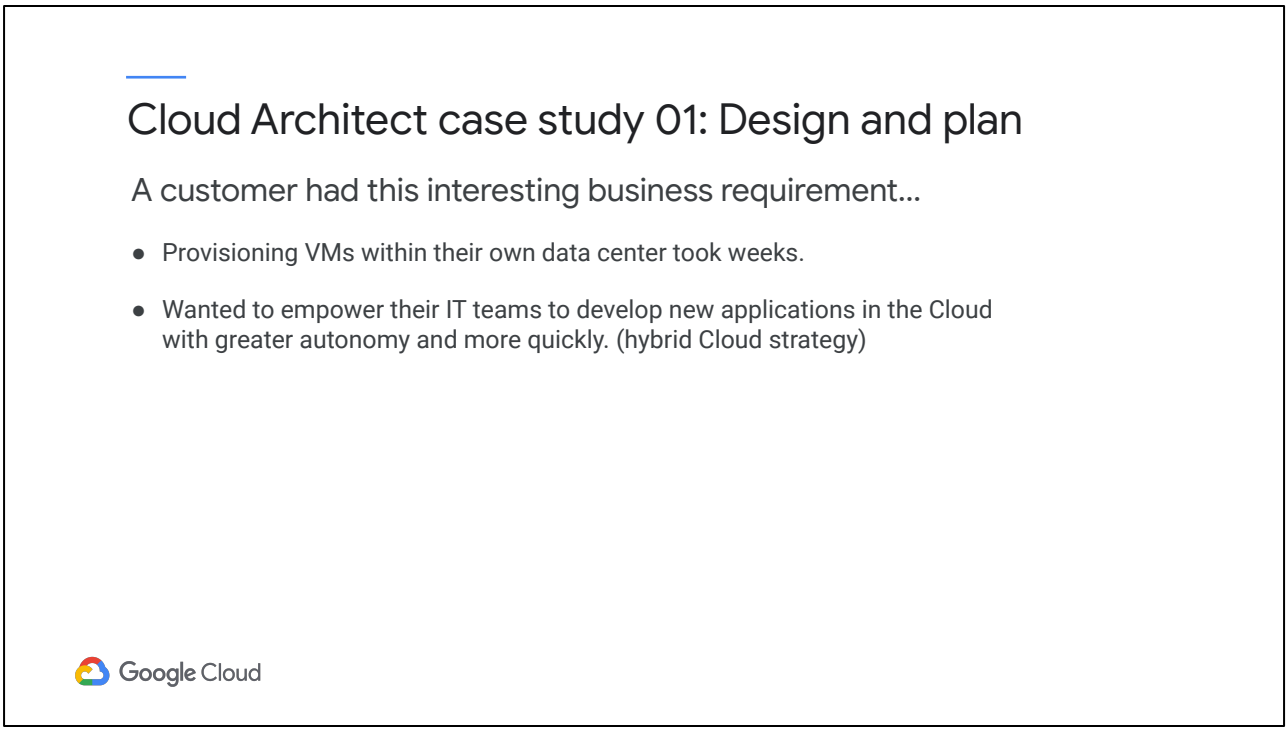
TIP: What is the priority or the blend of priorities -- Good, Fast, Cheap? And what are the qualities, time, budget by which those priorities will be measured?

Good-Fast-Cheap is one way to think about the decision to a) build a solution from scratch yourself, b) purchase a solution off-the-shelf or from a vendor, or c) customize existing products and services to trade-off speed for features.

"Good" in this case means control. If you build it yourself you get absolute control over what the solution does and how it works. But DIY (do-it-yourself) is rarely the fastest way to get a solution and it certainly is not cheap.

"Fast" means outsourcing. If you go to the experts who already have handled similar problems and have experience creating and implementing similar solutions, it is often the fastest way to get the matter resolved. But -- you have to give up control. And it might not be everything you had hoped for -- because you are basically giving the contractor/vendor the right to make tradeoffs for you. When you are short on time -- and the problem is imminent -- sometimes this is the most expedient way forwards.

"Cheap" means adapting what you already have (which might be partially depreciated, or already paid for). You might not get all the features you want. In some cases you might have to make up for deficiencies in the solution with manual methods -- transforming output of one part to provide input to another part through manual processes, for example.



This is a very common situation, where it isn't actually an application and a solution, but rather earlier in the business lifecycle moving to cloud, where the customer is interested in establishing standards and systems whereby their organizations can design and develop their own solutions as needed. The problem the customer came to us with was that their project teams were provisioning hardware, which was slow and arduous. A side effect of the slowness was that it stifled experimentation and innovation. Nobody wants to ask for hardware unless they know exactly what they need it for. That was the business requirement the customer expressed to us.

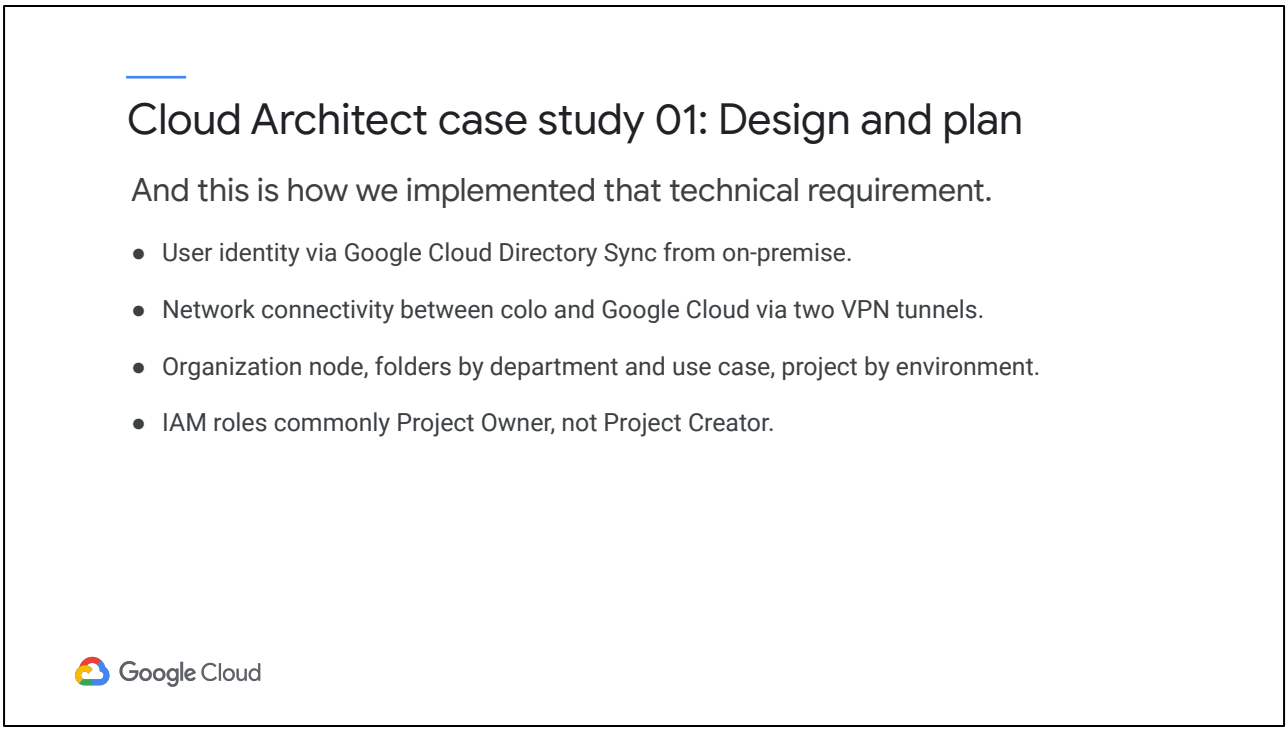
A customer had this interesting business requirement...

- Provisioning VMs within their own data center took weeks
- They wanted to empower their IT teams to develop new applications in the Cloud with greater autonomy and more quickly, using a hybrid Cloud strategy

We need a solution where the customer can provision projects, including run-time environments, at scale. And it needs to work very quickly. There is a review process. So this is not a "free-for-all". There is a human being or a group that will review certain parameters - a review gate. And basically, once a request has passed the review gate, it should only take a couple of minutes for the resources to be deployed.

We mapped that to technical requirements like this…

- A turn-key Google Cloud project for any developer team that requested it.
- Manually provisioned within 24 hours, following a brief review of the request.



From a practical implementation perspective, that meant setting up the Organization node, and Folders for each organization, for different kinds of organizations and different kinds of projects. We implemented VPN tunnels to GCP so the company could reach the cloud resources without having to go through the public internet space unprotected. We also went through a process of identifying the proper IAM roles to assign to the various environments. And we needed to establish processes for Project Owners to be able to log into the environment. We needed to synchronize these roles and identities with the Active Directory service in the data center.

And this is how we implemented that technical requirement.

- User identity via Google Cloud Directory Sync (GCDS) from on-premise
- Network connectivity between colo and GCP via two VPN tunnels
- Organization node, folders by department and use case, project by environment
- IAM roles commonly Project Owner, not Project Creator.

We needed to set up processes for establishing new project creation and who would be assigned specific roles. So there was a kind of process and logic around this that needed to be established. What kind of data is going to be stored in each project? Is it confidential? We need to go through some basic classification. And who is going to pay for it? In this case there were multiple billing accounts and that needed to be decided.

Designing a solution infrastructure that meets technical requirements

Exam outline	Tips
High availability and failover design	What are the real criteria? What are the measurable goals?
Elasticity of cloud resources	How do you want the solution to behave when busy? How will the system behave when under attack? How will it behave when traffic diminishes?
Scalability to meet growth requirements	Scaling for growth is different from autoscaling to cover temporary demand non-linearities. When is it time to add a node, upgrade a service, or switch services?

TIP

If a requirement is not fully described -- perhaps it is not important to the question being asked.

Google Cloud

TIP: If a requirement is not described fully - perhaps it is not important to the question being asked.

Scaling, elasticity, and failover. What will you measure?

1. Begin simply. Iterate.

Compose simple services.
Build **up** to complexity.
Iterate.

2. Plan for failure.

Avoid bottlenecks.
Avoid single points of failure.

TIP

3. Measure stuff.

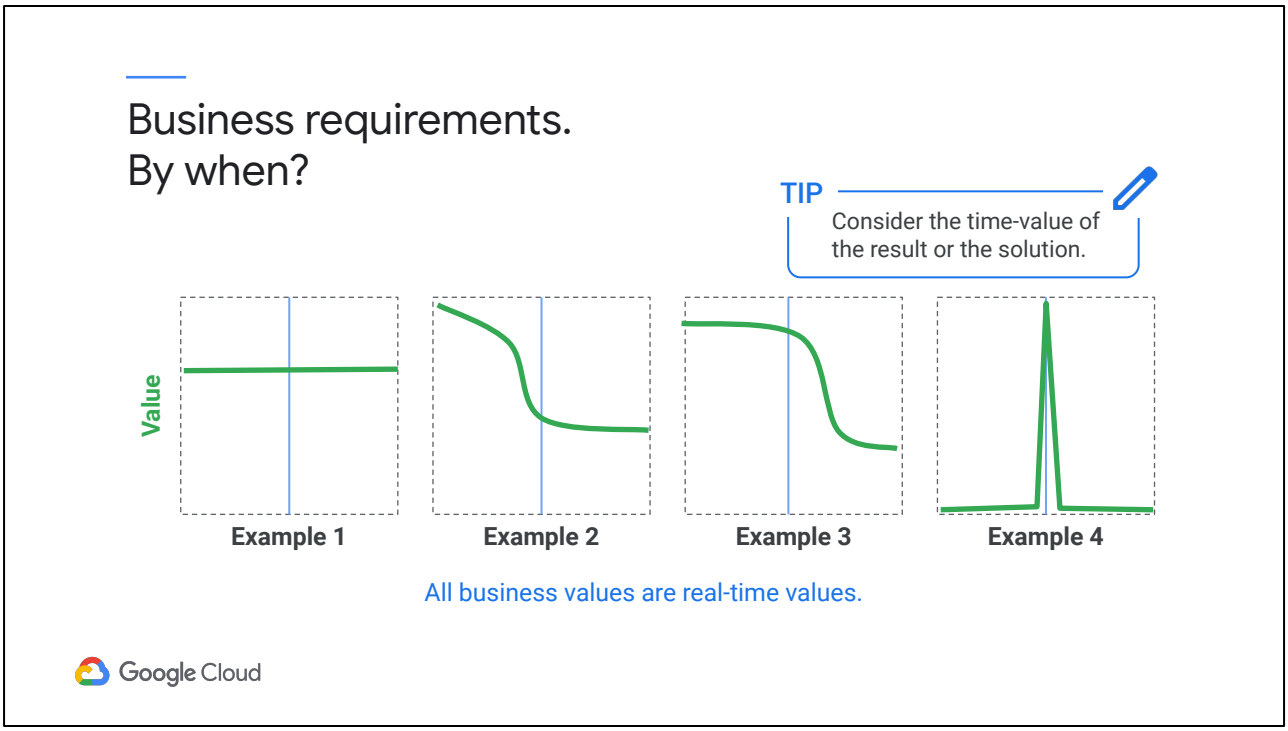
Consider the situation:
What are the limits?
How might the system fail?

Google Cloud

TIP: 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 would need to be measured will help you focus on what information is important in the 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.

High availability and failover design.



TIP: Consider the time-value of the result or the solution. ***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 and how this will affect value in the business.

Tradeoffs

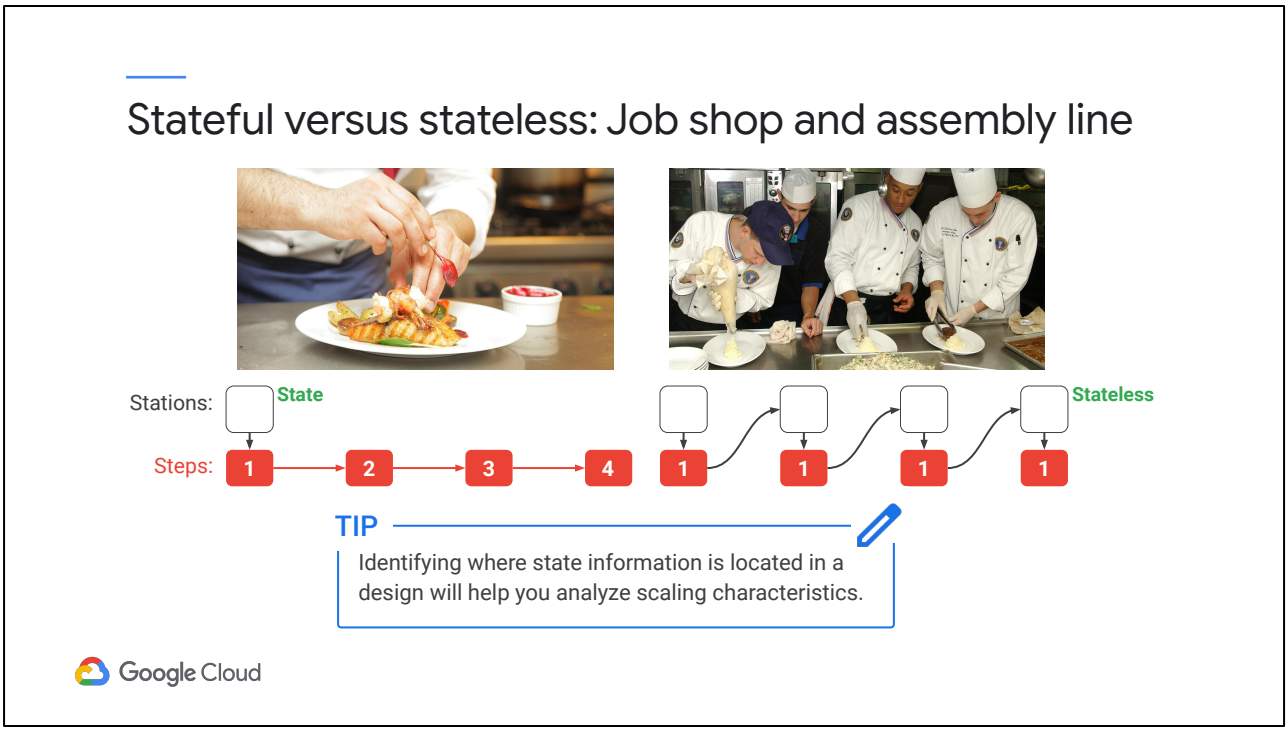
Time value. What happens if the solution is delivered sooner or later? "All values are real time values."

Example 1: A solution is of constant value whether delivered earlier or later.
An existing business process is replaced with a cloud-based process. The value is in cost savings. Whenever it is delivered it provides value. But because the business already expects the expense of the prior process, and that cost is built-in, the company does not value early or late delivery differently.

Example 2: A solution has additional value if delivered sooner. "The sooner the better".
A company wants to offer a new service. They know their competitors are working on something similar. The sooner they get to market the better because there is an unserved market. If they get to market first, they'll grab a larger share of the market. Otherwise, they will only get "their fair share".

Example 3: A solution has diminishing value if delivered late.
A new web service to provide instructions for a new product. If it is available during the initial offering of the product, it will have its greatest value because it will be used by the new purchasers of the product. If it is delivered a week or two later, it will still have value, but for fewer customers.

Example 4: A solution "falls off a cliff" if it is not delivered on time.
An order taking system that is connected to selling swag for a big sports event. If the system is delivered early, it has no additional value. If it is delivered after the sports event it has no value. It is only of value the weekend of the event.



TIP: Always watch for state and consider limits on scalability.

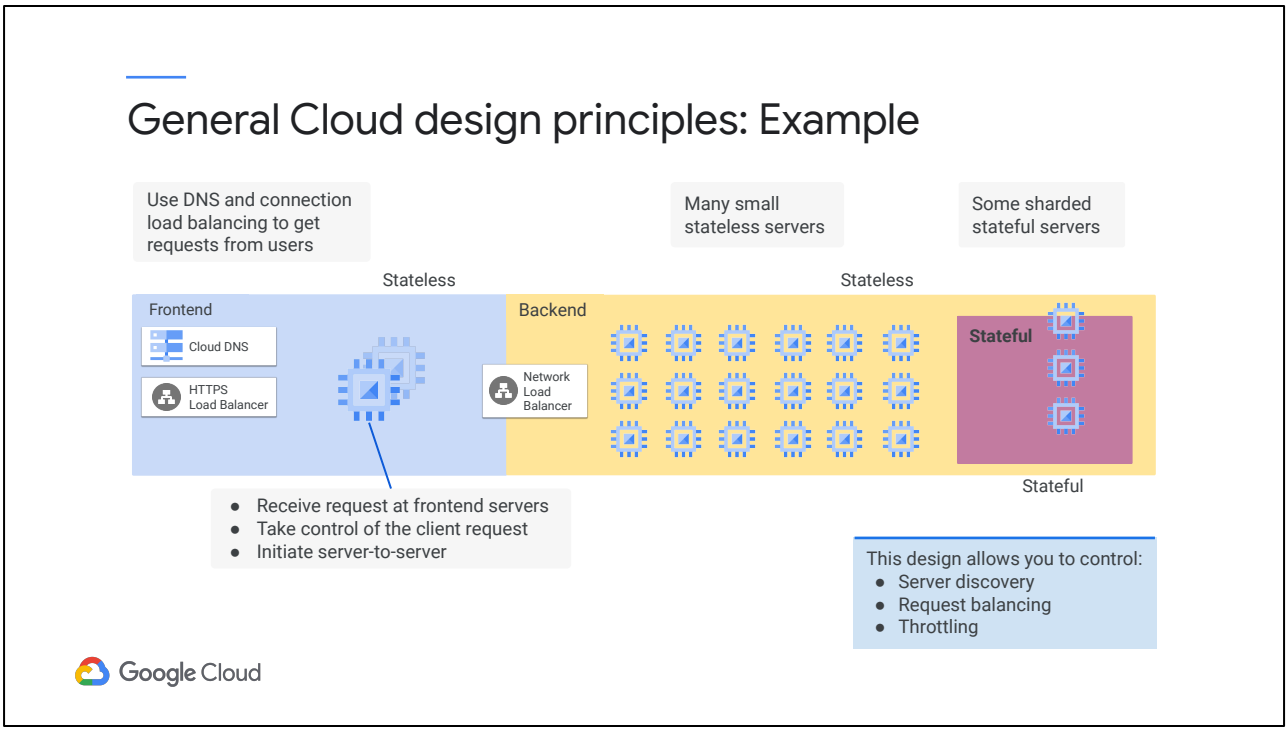
Analogy to explain state

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 step. These are the two extremes. There are variations where one station performs multiple steps as part of a conveyor belt of activities. If you think about the parts of your system in this way, it will highlight to you where work is handed off between stations and where multiple steps are performed within a single station. That will help you think about how to measure the work and how to adjust the relationships. For example, it is easier to queue up work between stations than within a single station. So there are natural breaks and tools. It makes it much easier to consider the operational performance characteristics of the overall system.

The job shop has to keep track of where it is in the process: STATE. The assembly line doesn't need to keep track of state. However, the units (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. And in those cases, you have to try to make state scalable and reliable.

One nice aspect of this approach is that it applies equally to human organizations and

to electronic systems. So you can define entire processes that include both automated and manual components.



TIP: 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. *Example: Dividing the problem/solution into "Frontend, Backend" and "Stateless, Stateful" can be very helpful in surfacing the key design issue.*

Resource for architectures: https://cloud.google.com/docs/tutorials#architecture

Scalable front end with fail-over.
Many small stateless servers for backend scalability.
A few stateful elements to support the scalable backend.

Business use case and product strategy

Designing network, storage, and compute resources

Exam outline	Tips
Integration with on premises/multi-cloud environments	When to use gsutil, gsutil rsync, and Storage Transfer Service
Cloud native networking (VPC, peering, firewalls, container networking)	Understand all the networking services and how to connect for throughput, security, billing, and so forth.
Identification of data processing pipeline matching data characteristics to storage systems	Velocity (how frequent), Volume (how much), Variety (format, structure), Volatility (how often does it change) -- which services match?
Data flow diagrams	Very helpful to know where the data will travel through the solution and to look for bottlenecks and flow monitoring and control
Storage system structure (e.g., Object, File, RDBMS, NoSQL, NewSQL)	ACID—What is eventual consistency ? BASE. Structure requirements?
Mapping compute needs to platform products	Scale, capacity, control/overhead?

Google Cloud

TIP: There is 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 the circumstance.

Storage Transfer Service: https://cloud.google.com/storage/transfer/

Cloud Transfer Appliance: https://cloud.google.com/transfer-appliance/

gsutil: https://cloud.google.com/storage/docs/quickstart-gsutil

gsutil sync: https://cloud.google.com/storage/docs/gsutil/commands/rsync

Which network, compute, storage resources are critical?

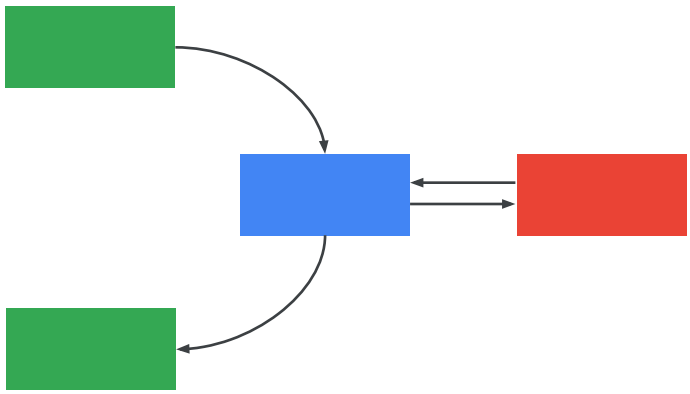
Google Cloud

TIP: Identifying bottlenecks is especially useful for questions involving building out from existing solutions. Example: The current system can only support X-number of users, and the goal is to support Y-number of users.

What is the bottleneck in the current design?

Is it bandwidth, gigabytes, queries per second? Where will the application reach limits?
This is often the factor that determines which solution is best in the circumstance.

Data flow diagrams

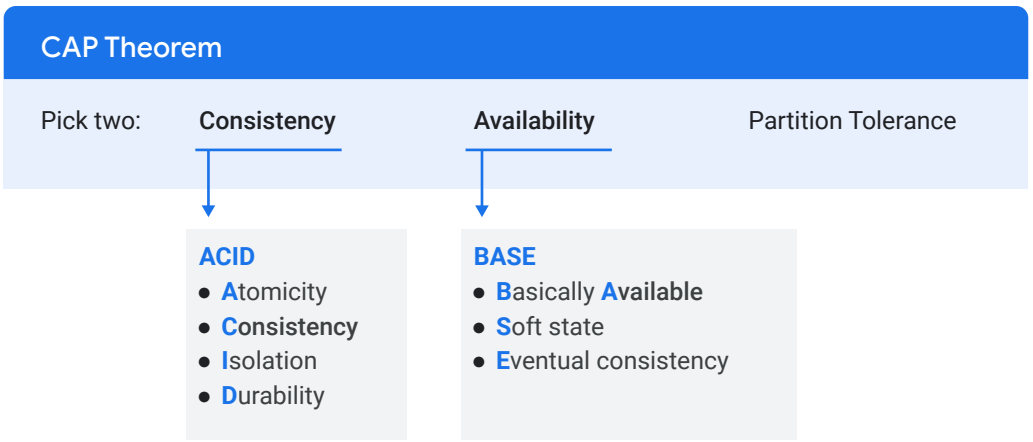


Google Cloud

TIP: 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 flows symmetrically, or that the capacities are symmetrical.

IOPS, Requests Per Second.
Can be useful in understanding workload, not just flow, but organizing you thinking around capacity.

What transaction qualities are required?



Google Cloud

TIP: ACID vs BASE is essential data knowledge that you will want to be very familiar with so that you can easily determine whether a particular data solution is compatible with the requirements identified in the case. Example: for a financial transaction, a service that provides only *eventual consistency* might be incompatible.

TIP: Did you know that in some cases an eventually consistent solution can be made strongly consistent for a specific limited use case?
https://cloud.google.com/datastore/docs/articles/balancing-strong-and-eventual-consistency-with-google-cloud-datastore/

In Datastore, there are only two APIs that provide a strongly consistent view for reading entity values and indexes: (1) the lookup by key method and (2) the ancestor query.

Database services provide a model of consistency. Consistency makes certain guarantees with respect to data transactions. And whatever guarantees are not made by the data service become the responsibility of the application code.

ACID - SQL databases such as MySQL and PostgreSQL.
BASE - NoSQL systems like Cloud BigTable

https://en.wikipedia.org/wiki/Consistency_(database_systems)

https://en.wikipedia.org/wiki/ACID

Atomicity - A transaction is either "all or nothing"

Consistency - Any transaction brings the database from one valid state to another.
Isolation - Transactions executed concurrently produce the same result as if executed sequentially.
Durability - Once a transaction is committed, the results are stable. Even in the event of a power loss the result is non-volatile.

BASE is described under eventual consistency in Wikipedia:
https://en.wikipedia.org/wiki/Eventual_consistency

An eventually consistent system does not have Atomic transactions. So once a transaction has started to be committed, there are no guarantees until all the parts of the system have converged. This means that users requesting data may get "any" result (ie there are no guarantees), but in practice it means the user gets stale data. That is a guarantee that they will get SOME data, some result, but not necessarily the most current result.

https://pixabay.com/en/menu-yemekservisi-akula-1197654/

https://en.wikipedia.org/wiki/CAP_theorem

In database systems, ACID (most SQL systems) optimize for consistency and BASE (most NoSQL systems) optimize for availability.

Creating a migration plan

Exam outline	Tips
Integrating solution with existing systems	Parallel?
Migrating systems and data to support the solution	How much, how to synchronize in both directions? Where is the source of data truth?
Licensing mapping	Data center license to cloud license -- may have different license or requirements for licensed software.
Network and management planning	How will on premises connect to the cloud solution?
Testing and proof of concept	Different ways to divide a test audience: random, group, phased, test.

TIP

Where is the source of truth?

How will actions be synchronized?

Google Cloud

TIP: In migration to cloud, you are always starting with data and processing in a data center. So obviously, that is 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? When does that change?

Think outside the technical/theoretical box

Planning: Technical Solution

TIP

Doing: Practical Considerations

Google Cloud

TIP: Often an 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.

What resources will remain on premise?

What processes are necessary to integrate with existing systems in the cloud (in Google Cloud, 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?

Envisioning future solution improvements

Exam outline	Tips
Cloud and technology improvements	When the technology improves, how will the solution embrace the improvements?
Business needs evolution	When the business changes, how will the solution change? Most solutions don't just need to work once, they need to become self-sustaining.
Evangelism and advocacy	Who are the "gatekeepers" and what do they need to know to effectively do their jobs?

Google Cloud

TIP: If you are removing 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), will it 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 help stabilize the solutions? What could change or what might need to be measured and adjusted?

TIP: In some cases there may be multiple stakeholders. The best solution will solve for all interests or for the common interest.

Example: The CEO wants improved data analytics. The CTO wants lower costs. At first, the two interests might seem in conflict. However, moving to a different more efficient solution may solve both requirements.

How will the solution grow and change once implemented?

Operations

TIP

- Monitoring
- Logging
- Maintenance
- Procedures
- Human processes

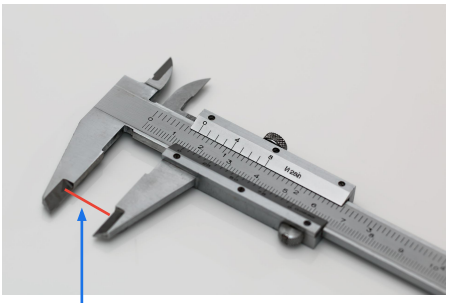


Figure out what to measure

Google Cloud

TIP: There is an old saying in IT: "Every 10x 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 to?
- What resources will remain on premise?
- What processes are necessary to integrate with existing systems in the cloud (in Google Cloud, 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?

Scenario #1

Question

Which application parts developed by separate project teams will communicate over RFC1918 addresses?

A. Single project, same VPC

B. Shared VPC, each project a service of the Shared VPC project

C. Parts communicate using HTTPS

D. Communicate over global load balancers, one per project

Google Cloud

Origin: CAPE

Scenario #1

Answer

Which application parts developed by separate project teams will communicate over RFC1918 addresses?

A. Single project, same VPC

B. Shared VPC, each project a service of the Shared VPC project

C. Parts communicate using HTTPS

D. Communicate over global load balancers, one per project

Scenario #1

Rationale

B - Each team has their own project but communicates securely over a single RFC1918 address space.

A - No separation.

C - Doesn't specify separate projects, therefore doesn't meet business requirements.

D - External IPs do not conform to address technical requirements.

https://cloud.google.com/vpc/docs/shared-vpc

https://cloud.google.com/vpc/docs/vpc-peering

Google Cloud

The specific configuration is called "Shared VPC, VPC Network Peering" and is described in the second link.

Scenario #2

Question

Which managed service solution would allow a company to support Apache Hadoop clusters to carry out real-time trend analysis?

A. Dataflow

B. BigQuery

C. A Hadoop cluster in Compute Engine

D. Dataproc

Google Cloud

Origin: Case study

Scenario #2

Answer

Which managed service solution would allow a company to support Apache Hadoop clusters to carry out real-time trend analysis?

A. Dataflow

B. BigQuery

C. A Hadoop cluster in Compute Engine

D. Dataproc

Google Cloud

Origin: Case study

Scenario #2

Rationale

D - A managed service that can run the Hadoop application.
Real-time trend analysis … refer to the case study to learn this is Hadoop/Spark.

A, B - Dataflow and BigQuery are both "serverless services", not managed services, and therefore won't meet the technical requirement to "Use managed services whenever possible."

C - A Hadoop cluster on Compute Engine is not a managed service.

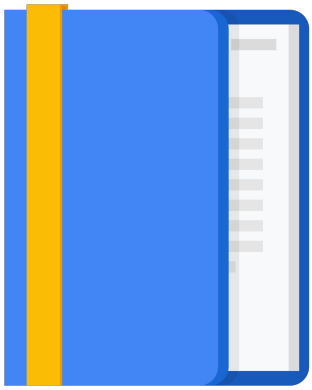
https://cloud.google.com/dataproc/docs/
https://cloud.google.com/solutions/migration/hadoop/hadoop-gcp-migration-overview
https://cloud.google.com/solutions/migration/hadoop/hadoop-gcp-migration-jobs
https://cloud.google.com/solutions/migration/hadoop/hadoop-gcp-migration-data

Module agenda

Designing and Planning a Cloud Solution Architecture

Managing and Provisioning Solution Infrastructure

Containers and Google Kubernetes Engine



Google Cloud

Configuring network topologies

Exam outline	Tips
Extending to on-premises (hybrid networking)	VPN. Interconnect.
Extending to a multi-cloud environment	Interoperation interface. Google Cloud's operations suite (Google Cloud + AWS)
Security	*Discussed in "Designing for security and compliance"*
Data protection	Encryption, access, CSEK, KMS

Google Cloud

TIP: External connections -- know your options; Internet, VPN, Cloud Router, and the various flavors of direct interconnects.

Network

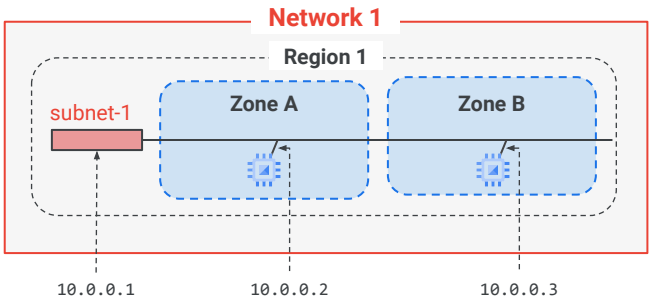
Google Cloud

TIP: Google cloud networking is not like other vendor network, 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 Google Cloud network.

https://cloud.google.com/docs/compare/data-centers/networking

Subnets extend across zones

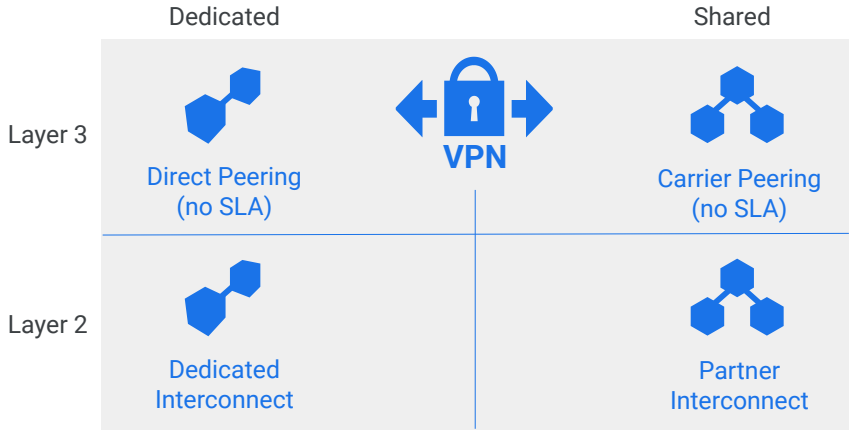
Subnetworks can extend across zones in the same region.
One VM and an alternate VM can be on the same subnet but in different zones.
A single firewall rule can apply to both VMs even though they are in different zones.



Google Cloud

TIP: Most confusing for cloud experts from other vendor's clouds. And yet it makes SO MUCH SENSE. A number of features of Google Cloud networks eliminate "make work", where methods from traditional IP have been inherited out of habit.

There are five ways to interconnect



https://cloud.google.com/interconnect/

Google Cloud

TIP: 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?

Protect data by practicing defense in depth

Edge protections

- VM access/IP addresses
- API access/endpoints

Network protections

- Firewall is the first line of defense

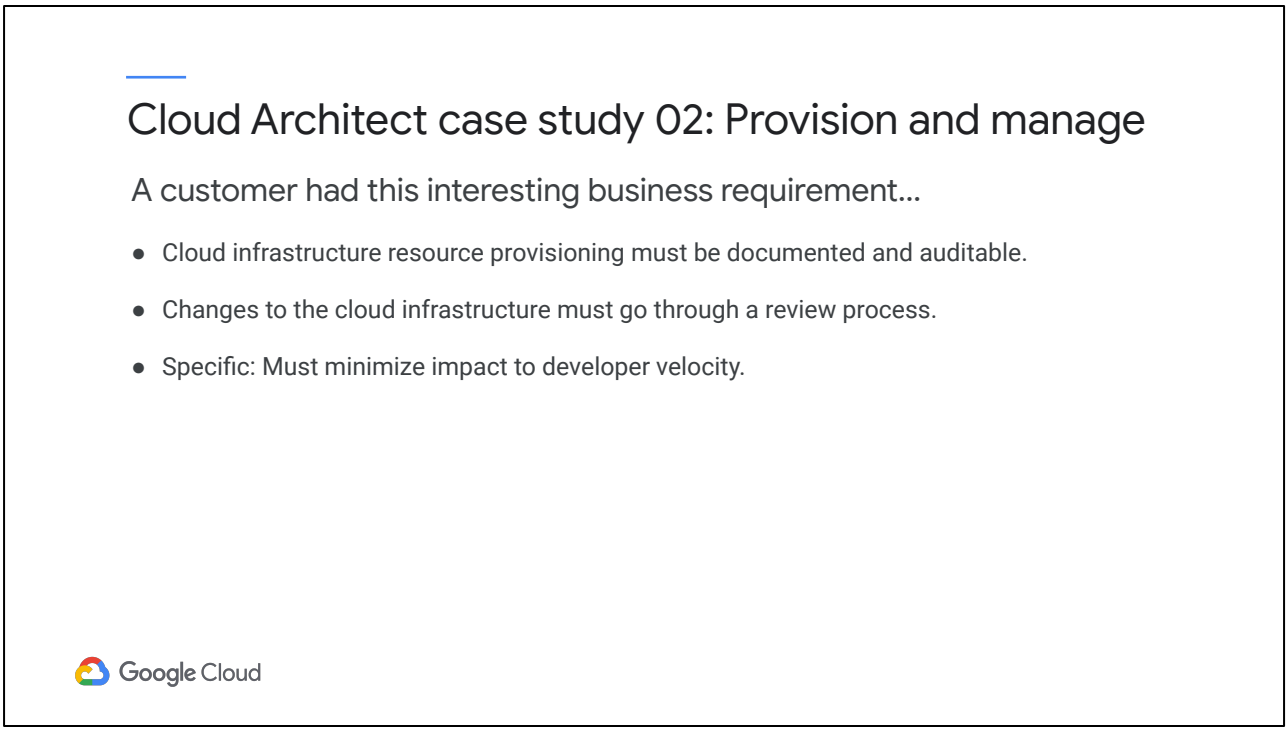
Infrastructure protections

- Isolation and controlled sharing of resources
- Server-side encryption and key management

Google Cloud

TIP: 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 storm the castle. In the exam, think "practical security" -- what is needed -- unless the requirement is compliance with a standard.

Even if you comply with a standard, that might not be sufficient to meet business security requirements. Sometimes standards are "floors" (minimum allowable) rather than "ceilings" (maximum possible).

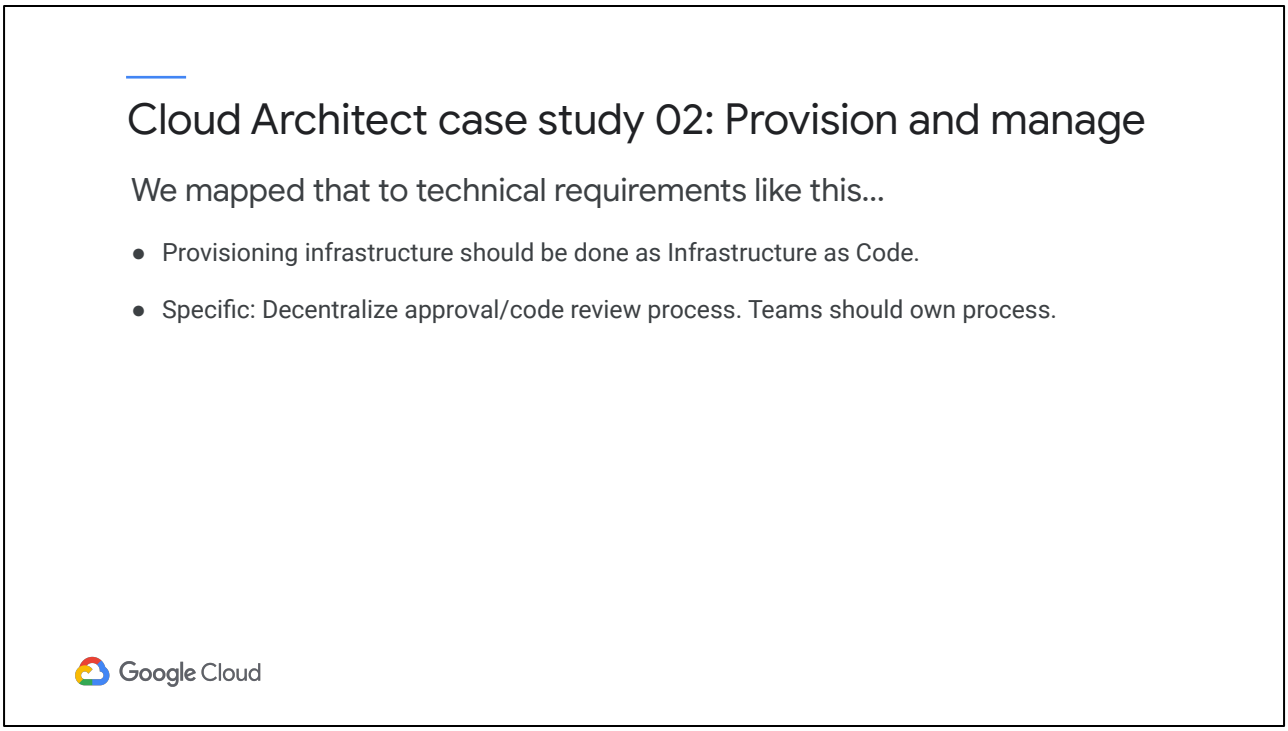


One thing I see with every large customer is that they have business requirements; they want to document resources. If I'm paying the bill, I want to know who is consuming resources, spinning up VMs for example. And I want to know that this is going through some sort of review process.

This customer wanted me to minimize the impact to the developer productivity. As soon as you start to add any sort of process, you slow things down. And they were very concerned that I not hurt their productivity. That was both a technical requirement and a business requirement.

A customer had this interesting business requirement...

- Cloud infrastructure resource provisioning must be documented and auditable
- Changes to the cloud infrastructure must go through a review process
- Specific: Must minimize impact to developer velocity

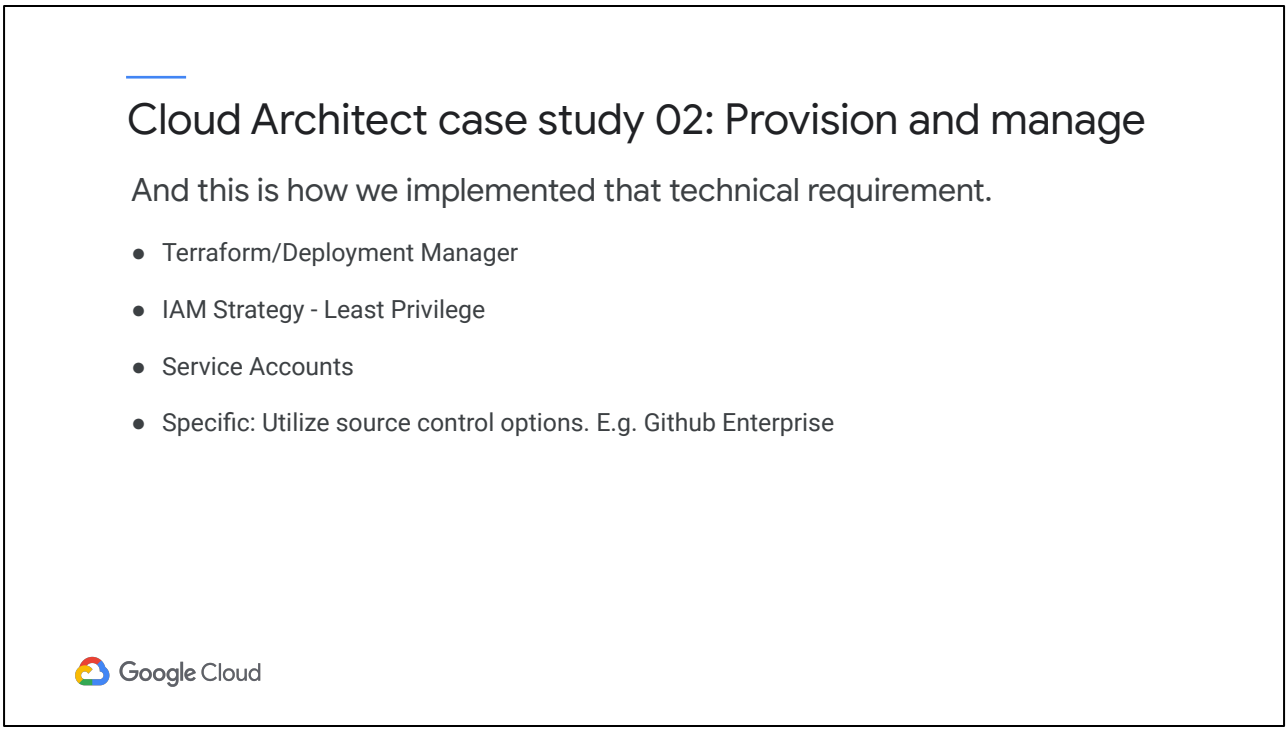


We immediately knew that part of the solution was "infrastructure as code". There are a lot of reasons to implement infrastructure as code. But forcing a solution into this implementation means you get the benefits of auditing, code review, and all those things that help satisfy the business requirement.

The customer has dozens of development teams but only one Cloud Engineer. So the solution needed to be distributed. Each team is actually responsible for owning their own process.

We mapped that to technical requirements like this…

- Provisioning infrastructure should be done as Infrastructure as Code
- Specific: Decentralize approval/code review process. Teams should own process.



And this is how we implemented that technical requirement.

- Terraform/Deployment Manager
- IAM Strategy - Least Privilege
- Service Accounts
- Specific: Utilize source control options, such as Github Enterprise

They use Github Enterprise with a code owners file. The file is a text file that limits who can actually perform actions in the repository.

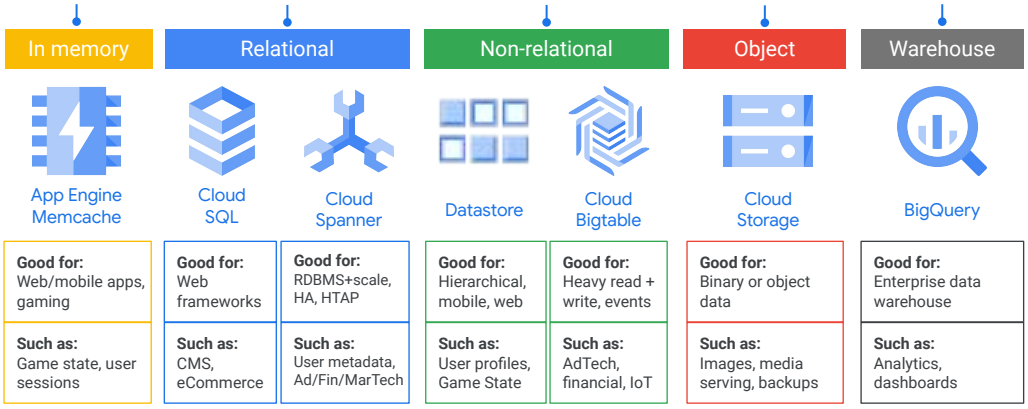
Configuring individual storage systems

Exam outline	Tips
Data storage allocation	Where will the data be located? Resiliency? Charges?
Data processing/compute provisioning	Which platform? What capacity?
Security and access management	CSEK and Cloud Storage, for example. *Discussed in "Designing for security and compliance"*
Network configuration for data transfer and latency	Location makes a difference in egress charges and round-trip time.
Data retention and data lifecycle management	What is the Nearline and Coldline policy? When to delete data?
Data growth management	When do you need a different way to organize the data? How much/how big will the current design support?

Google Cloud

TIP: Dig in to 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.

Comparing storage and database



Google Cloud

Comparing storage options: Technical details

	Datastore	Bigtable	Cloud Storage	Cloud SQL	Cloud Spanner	BigQuery
Type	NoSQL document	NoSQL wide column	Blobstore	Relational SQL for OLTP	Relational SQL for OLTP	Relational SQL for OLAP
Transactions	Yes	Single-row	No	Yes	Yes	No
Complex queries	Yes	No	No	Yes	Yes	Yes
Capacity	Terabytes+	Petabytes+	Petabytes+	10,230 GB	Petabytes	Petabytes+
Unit size	1 MB/entity	~10 MB/cell ~100 MB/row	5 TB/object	Determined by DB engine	10,240 MiB/ row	10 MB/row

Google Cloud

- Cloud Bigtable is not a relational database; it does not support SQL queries or joins, nor does it support multi-row transactions. Also, it is not a good solution for small amounts of data (< 1 TB).
- If you need full SQL support for an online transaction processing (OLTP) system, consider Cloud SQL and Cloud Spanner. Cloud Spanner is particularly suited for databases larger than about 2 TB and databases that will be written to by global clients.
- If you need to store immutable blobs larger than 10 MB, such as large images 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 Datastore.
- If you need interactive querying in an online analytical processing (OLAP) system, consider BigQuery.

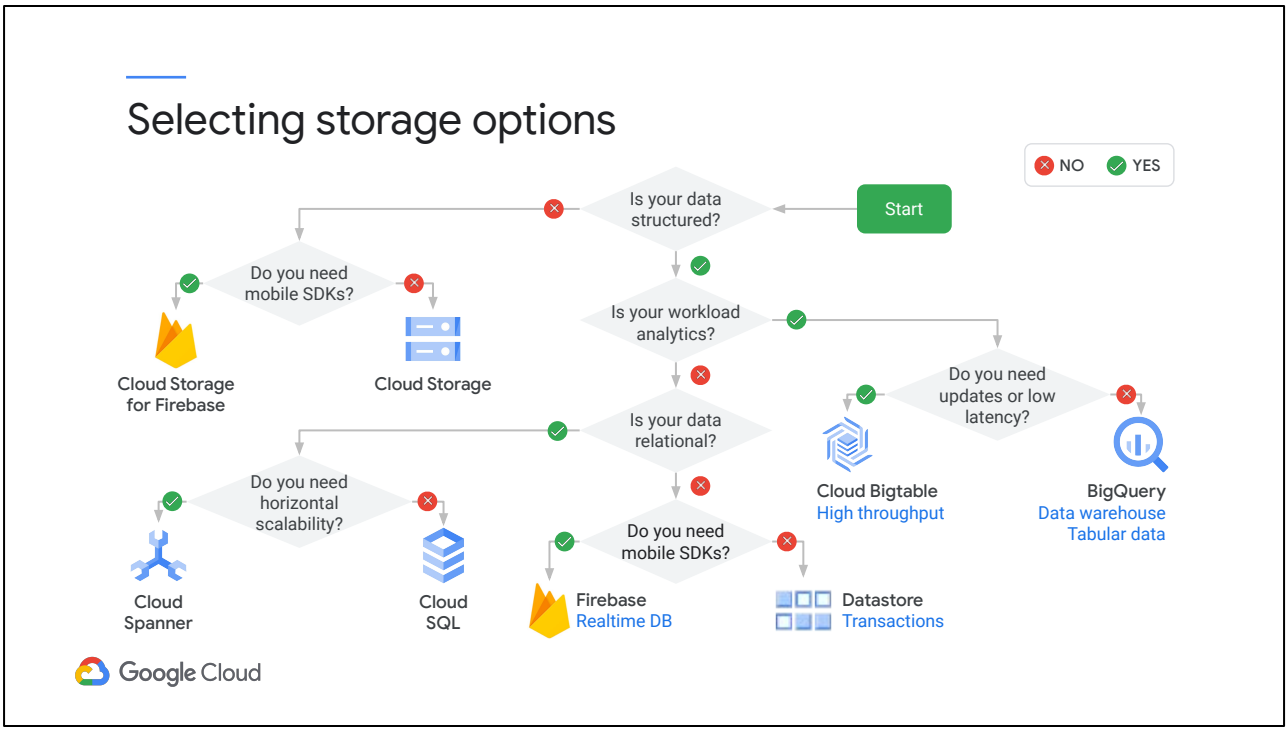
Comparing storage options: Use cases

	Datastore	Bigtable	Cloud Storage	Cloud SQL	Cloud Spanner	BigQuery
Type	NoSQL document	NoSQL wide column	Blobstore	Relational SQL for OLTP	Relational SQL for OLTP	Relational SQL for OLAP
Best for	Getting started, App Engine applications	"Flat" data, Heavy read/write, events, analytical data	Structured and unstructured binary or object data	Web frameworks, existing applications	Large-scale database applications (> ~2 TB)	Interactive querying, offline analytics
Use cases	Getting started, App Engine applications	AdTech, Financial and IoT data	Images, large media files, backups	User credentials, customer orders	Whenever high I/O, global consistency is needed	Data warehousing

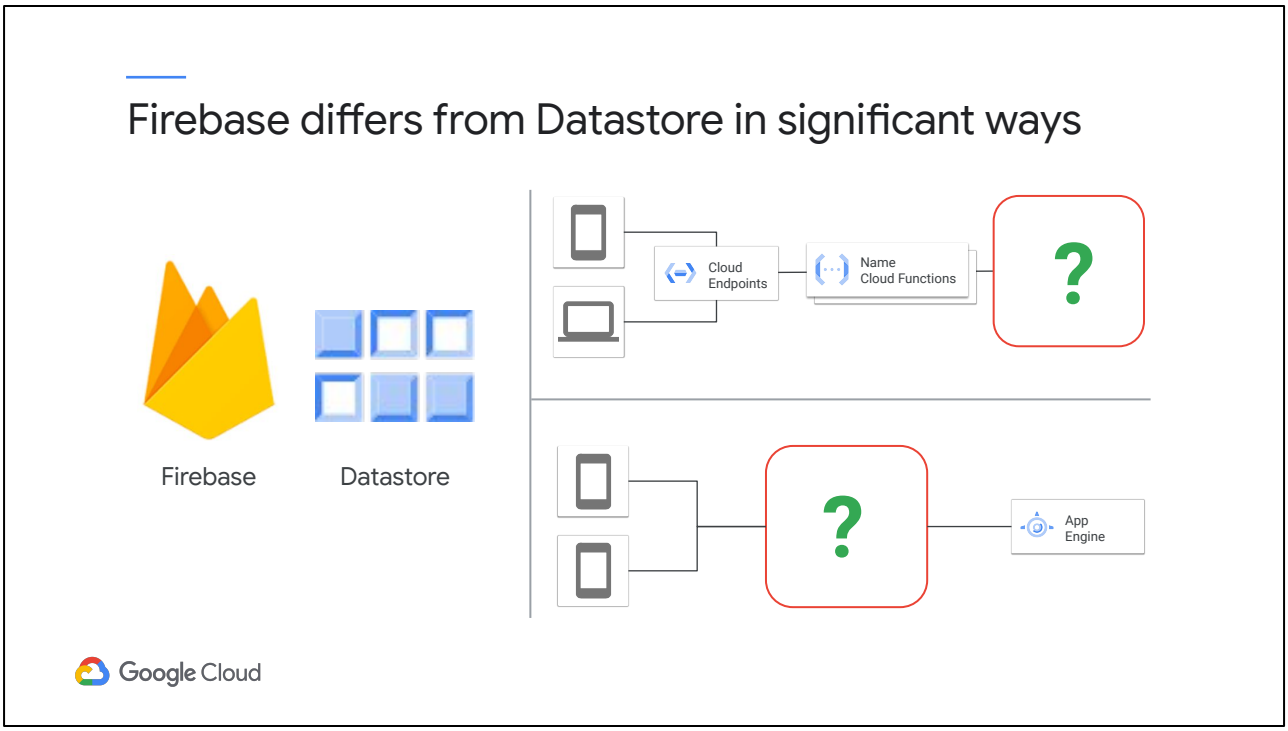
Google Cloud

Google Cloud 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 service 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.
- Datastore provides a nearly infinitely scalable, schemaless solution.
- If you want a disk, you can mount Persistent Disk as a block store that can be used by Compute Engine.
- For just pure data and blobs, Cloud Storage can deliver what you need.
- Cloud Bigtable offers companies a fast, fully managed, infinitely scalable NoSQL database service ideal for web, mobile, and IoT applications.



TIP:
BigQuery is recommended as a data warehouse.
BigQuery is the default storage for tabular data.
Use Cloud BigTable if you need transactions.
Use Cloud BigTable if you want low-latency/high-throughput.



Firebase differs from Datastore in many significant ways. Firebase is a mobile platform that provides features beyond storage including authentication, notifications, and real-time synchronization to clients. Datastore is a NoSQL Database. Cloud Bigtable can scale to massive amounts of data. Cloud Bigtable queries are more sophisticated than Firebase queries.

https://cloud.google.com/docs/tutorials#marketing

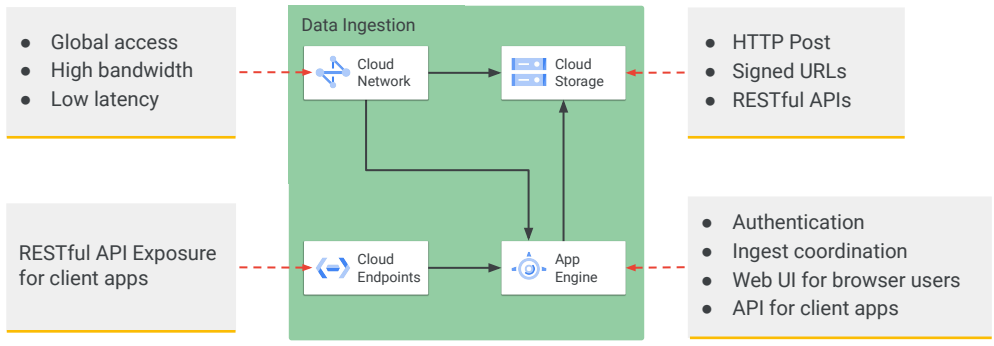
https://cloud.google.com/solutions/building-scalable-web-apps-with-cloud-datastore

https://cloud.google.com/solutions/mobile/mobile-app-backend-services

Solution:
Top one is a scalable web app with mobile support. Cloud functions drive a Cloud Bigtable backend.
Bottom one is a mobile client app. Mobile clients interact with Firebase, and App Engine provides backend processing.

Ingesting data into your service

These are the key components for ingesting data into Google Cloud and the features they contribute to data ingestion.



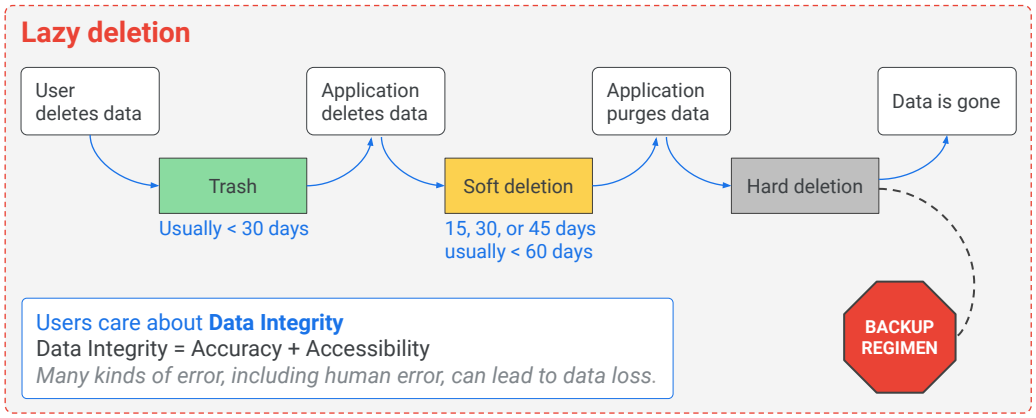
Google Cloud

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. And upload is provided for client applications via RESTful APIs exposed by Cloud Endpoints.

The goal of backup isn't archive, it's restore



Google Cloud

Example of lazy deletion. Number of layers of "soft deletion" and delay between steps should be tuned to the application and business requirements.

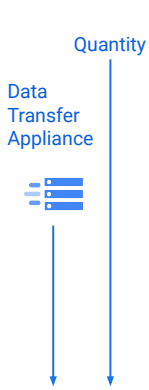
Data integrity is the quality of the data remaining accurate and accessible.

If the data loses accuracy or accessibility, it is no longer reliable to the system (and the users), and therefore has lost integrity.

Many kinds of failures, including human error, can lead to data loss.

Strategy is defense in depth.

How long does data transfer take?



	1 Mbps	10 Mbps	100 Mbps	1 Gbps	10 Gbps	100 Gbps
1 GB	3 hrs	18 mins	2 mins	11 secs	1 sec	0.1 sec
10 GB	30 hrs	3 hrs	18 mins	2 mins	11 secs	1 sec
100 GB	12 days	30 hrs	3 hrs	18 mins	2 mins	11 secs
1 TB	124 days	12 days	30 hrs	3 hrs	18 mins	2 mins
10 TB	3 years	124 days	12 days	30 hrs	3 hrs	18 mins
100 TB	34 years	3 years	124 days	12 days	30 hrs	3 hrs
1 PB	340 years	34 years	3 years	124 days	12 days	30 hrs
10 PB	3,404 years	340 years	34 years	3 years	124 days	12 days
100 PB	34,048 years	3,404 years	340 years	34 years	3 years	124 days

Google Cloud

How long does it take to transfer data online?

The left side of the table are closer to physical speeds, the right side of the table is closer to online speeds.

Therefore, it is much faster to accumulate data online and work with it and transfer it online than to collect data physically and then transfer it.

https://cloud.google.com/data-transfer/

Cloud Storage is extremely versatile

Backup and archive

Replace/decommission infrastructure

Content storage and delivery

TIP

Bucket classes, access control, object versioning, encryption keys, object lifecycle

Google Cloud

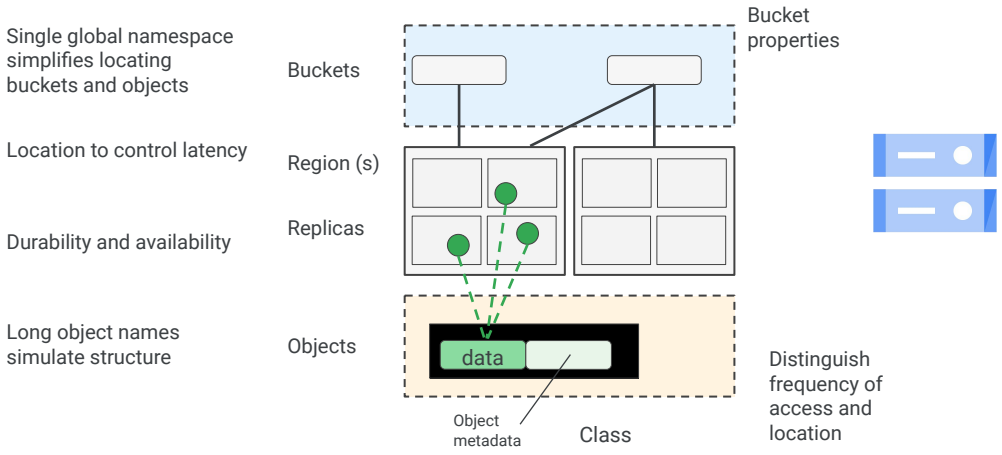
https://cloud.google.com/products/data-transfer/

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

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. Dataproc is a managed service for Hadoop clusters - useful for processing data and returning it to Cloud Storage or BigQuery.

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.

How does Cloud Storage work?



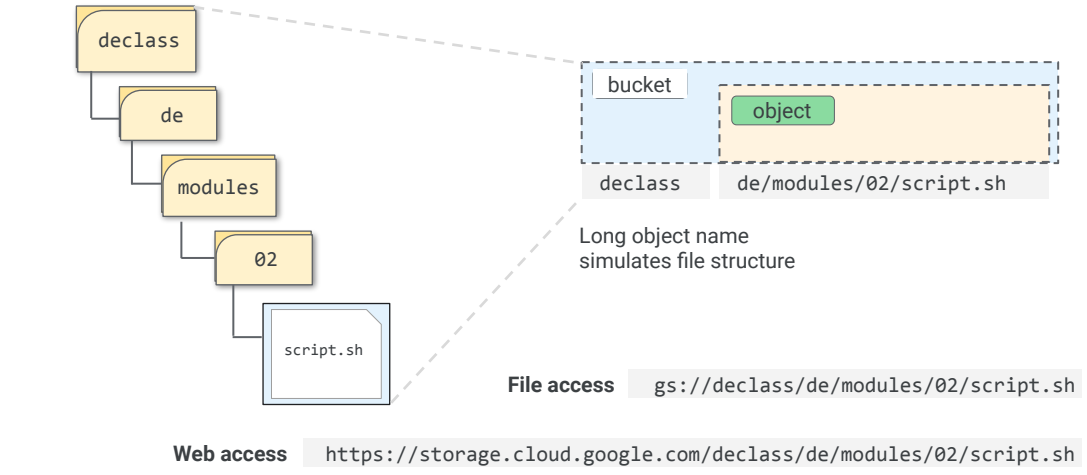
Google Cloud

Cloud Storage is an object store. A lot of its amazing properties have to do with the fact of what's missing -- that it is not a file system.

A global namespace for buckets simplifies locating a specific bucket. Buckets are either multi-regional, dual-regional, or regional. Objects within a bucket are replicated across zones.

That gives the objects durability and availability that are not possible in a physical storage device like a disk. Objects have several properties, among them is "class" which declares the frequency of anticipated access of the object.
This allows the Cloud Storage service to distinguish between types of objects and internally manage them to service levels.

Cloud Storage simulates a file system



Google Cloud

https://storage.cloud.google.com uses TLS (HTTPS) to transport your data which protects credentials as well as data in transit.

Avoiding use of sensitive information as part of bucket or object names.

Choosing among Google Cloud Storage classes

Storage Class	Minimum duration	Availability SLA	Typical monthly availability	Use cases	Name for APIs and gsutil
Standard Storage	None	Multi-region 99.95% Dual-region 99.95% Region 99.9%	>99.99% availability in multi-regions and dual-regions; 99.99% in regions	Access data frequently ("hot" data) and/or store for brief periods • Serve website content • Stream videos • Interactive workloads • Mobile and gaming apps	STANDARD
Nearline Storage	30 days	Multi-region 99.9% Dual-region 99.9% Region 99.0%	99.95% availability in multi-regions and dual-regions; 99.9% in regions	Read/modify data ≤ once per month • Data backup • Serve long-tail multimedia content	NEARLINE
Coldline Storage	90 days			Read/modify data no more than once a quarter	COLDLINE
Archive Storage	365 days	None		Read/modify data < once a year • Cold data storage • Disaster recovery	ARCHIVE

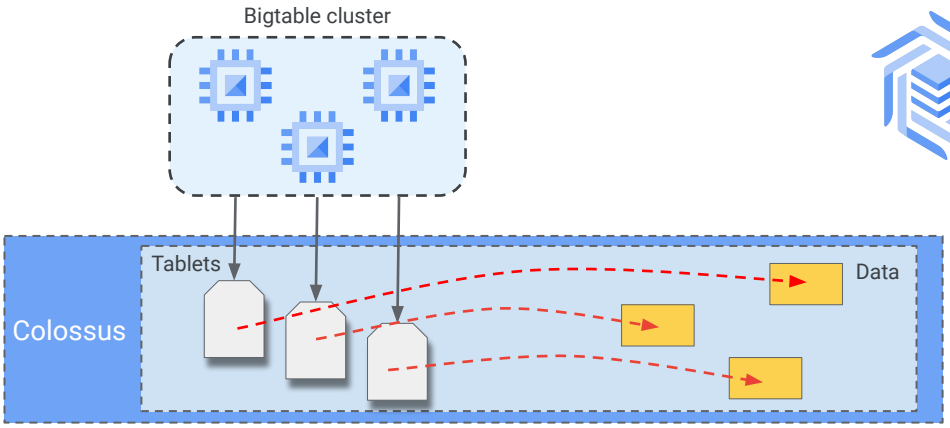
Google Cloud

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 stored per month, and egress and data transfer charges may apply. See the Cloud Storage pricing page for more information on class-specific pricing. [https://cloud.google.com/storage/pricing]

For more information Cloud Storage classes, see:
https://cloud.google.com/storage/docs/storage-classes

How does Cloud Bigtable work?



Google Cloud

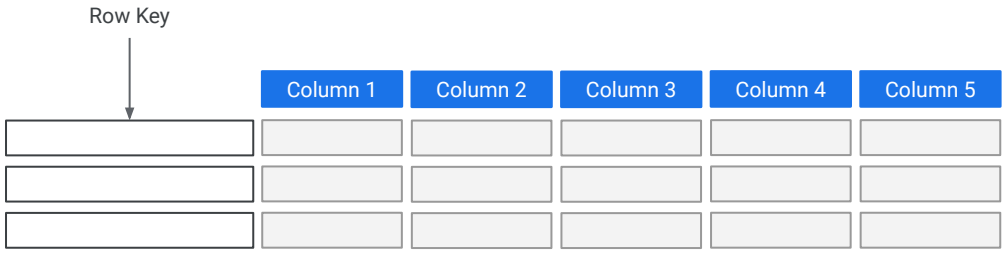
Cloud Bigtable stores data in a file system called Colossus. Colossus also contains data structures called Tablets that are used to identify and manage the data. And metadata about the Tablets is what is stored on the VMs in the Bigtable cluster itself.

This design provides amazing qualities to Cloud Bigtable. It has three levels of operation. It can manipulate the actual data. It can manipulate the Tablets that point to and describe the data. Or it can manipulate the metadata that points to the Tablets. Rebalancing tablets from one node to another is very fast, because only the pointers are updated.

Cloud Bigtable is a learning system. It detects "hot spots" where a lot of activity is going through a single Tablet and splits the Tablet in two. It can also rebalance the processing by moving the pointer to a Tablet to a different VM in the cluster. So its best use case is with big data -- above 300 GB -- and very fast access but constant use over a longer period of time. This gives Cloud Bigtable a chance to learn about the traffic pattern and rebalance the Tablets and the processing.

When a node is lost in the cluster, no data is lost. And recovery is fast because only the metadata needs to be copied to the replacement node. Colossus provides better durability than the default 3 replicas provided by HDFS.

Cloud Bigtable design idea is "simplify for speed"



The Row Key is the index.
And you get only one.

Google Cloud

Cloud Bigtable stores data in tables. And to begin with, it is just a table with rows and columns. However, unlike other table-based data systems like spreadsheets and SQL databases, Cloud Bigtable has only one index.

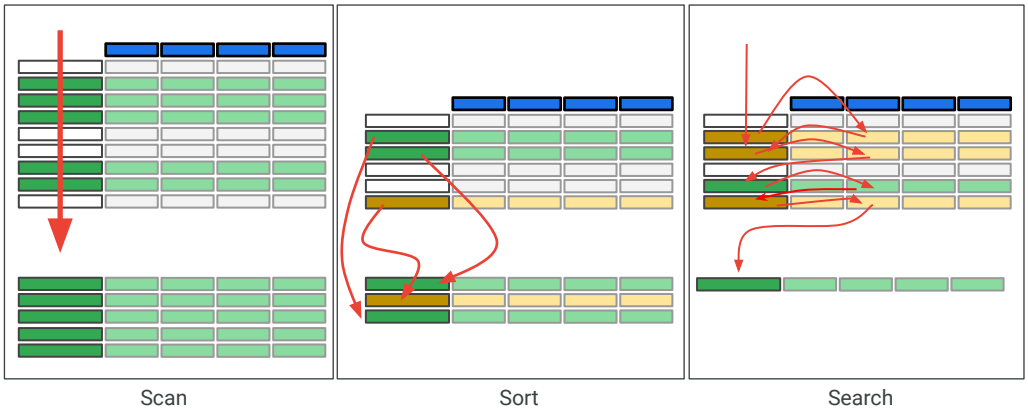
That index is called the Row Key. There are no alternate indexes or secondary indexes. And when data is entered, it is organized lexicographically by the Row Key.

The design principle of Cloud Bigtable is speed through simplification. If you take a traditional table, and simplify the controls and operations you allow yourself to perform on it then you can optimize for those specific tasks.

It is the same idea behind RISC (Reduced Instruction Set Computing). Simplify the operations. And when you don't have to account for variations, you can make those that remain very fast.

In Cloud Bigtable, the first thing we must abandon in our design is SQL. This is a standard of all the operations a database can perform. And to speed things up we will drop most of them and build up from a minimal set of operations. That is why Cloud Bigtable is called a NoSQL database.

But speed depends on your data and Row Key



Google Cloud

The green items are the results you want to produce from the query. In the best case you are going to scan the Row Key one time, from the top-down. And you will find all the data you want to retrieve in adjacent and contiguous rows. You might have to skip some rows. But the query takes a single scan through the index from top-down to collect the result set.

The second instance is sorting. You are still only looking at the Row Key. In this case the yellow line contains data that you want, but it is out of order. You can collect the data in a single scan, but the solution set will be disorderly. So you have to take the extra step of sorting the intermediate results to get the final results. Now think about this. What does the additional sorting operation do to timing? It introduces a couple of variables. If the solution set is only a few rows, then the sorting operation will be quick. But if the solution set is huge, the sorting will take more time. The size of the solution set becomes a factor in timing. The orderliness of the original data is another factor. If most of the rows are already in order, there will be less manipulation required than if there are many rows out of order. The orderliness of the original data becomes a factor in timing. So introducing sorting means that the time it takes to produce the result is much more variable than scanning.

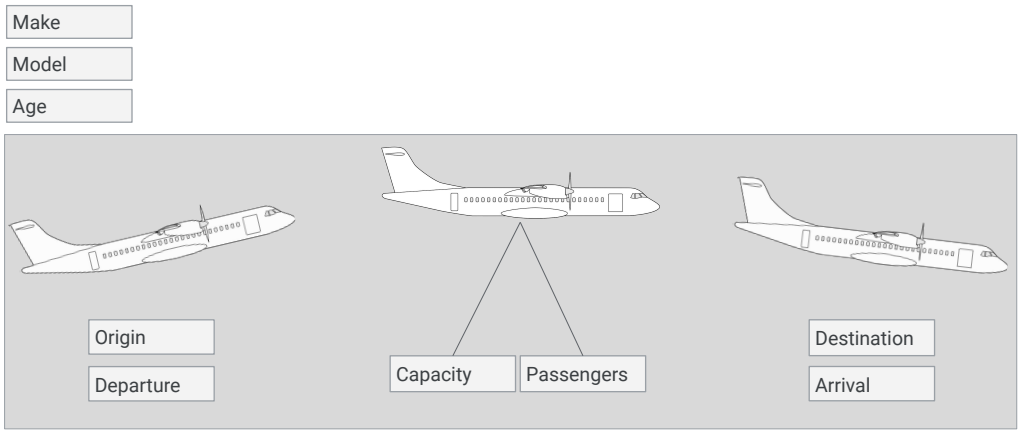
The third instance is searching. In this case, one of the columns contains critical data. You can't tell whether a row is a member of the solution set or not without examining the data contained in the critical column. The Row Key is no longer sufficient. So now you are bouncing back and forth between Row Key and column contents. There are many approaches to searching. You could divide it up into multiple steps, one scan

through the Row Keys and subsequent scans through the columns, and then perhaps a final sort to get the data in the order you want. And it gets much more complicated if there are multiple columns containing critical information. And it gets more complicated if the conditions of solution set membership involve logic such as a value in one column AND a value in another column, or a value in one column OR a value in another column. However, any algorithm or strategy you use to produce the result is going to be slower and more variable than scanning or sorting.

What is the lesson from this exploration? That to get the best performance with the design of the Cloud Bigtable service, you need to get your data in order first, if possible, and you need to select or construct a Row Key that minimizes sorting and searching and turns your most common queries into scans.

Not all data and not all queries are good use cases for the efficiency that the Cloud Bigtable service offers. But when it is a good match, Cloud Bigtable is so consistently fast that it is magical.

Flights of the world: Reviewing the data



Google Cloud

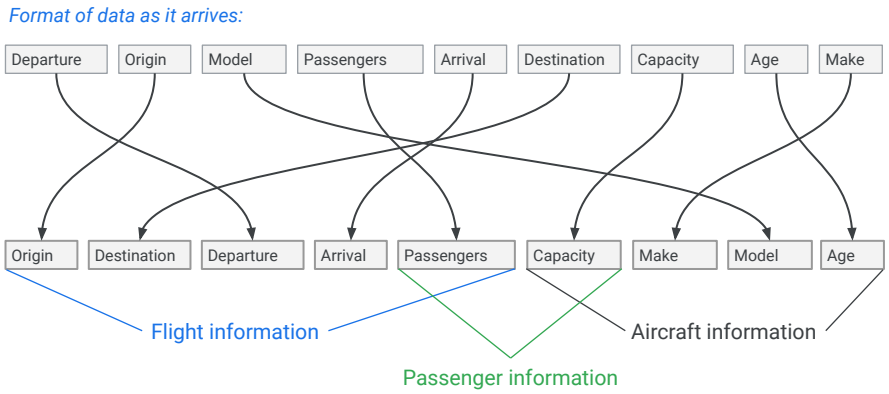
Each entry records the occurrence of one flight.

The data include city of origin and the date and time of departure, and destination city and date and time of arrival.

Each airplane has a maximum capacity, and related to this is the number of passengers that were actually aboard each flight.

Finally, there is information about the aircraft itself, including the manufacturer, called the make, the model number, and the current age of the aircraft at the time of the flight.

Flights of the world: Transforming the data



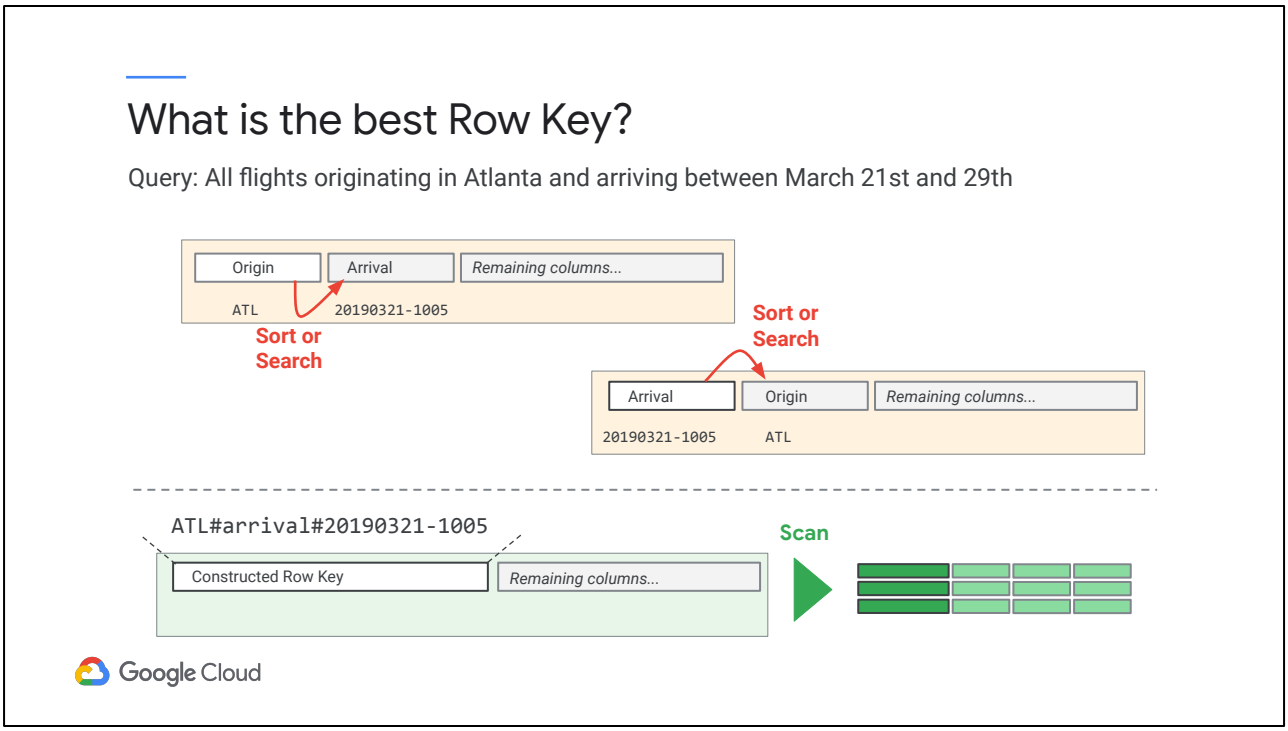
Google Cloud

Often the data arrives in a pre-selected order that might not be optimal for the purposes of the application.

In this case, transforming the data before storing it makes sense.

In the example, the fields are sorted into Flight Information and Aircraft Information.

The number of passengers on a flight and the capacity of the flight to hold passengers were moved to the end, and beginning of the groups so that passenger-related information would be adjacent.



In this example, the Row Key will be defined for the most common use case. The Query is to find all flights originating from the Atlanta airport and arriving between March 21st and 29th. The airport where the flight originates is in the Origin field. And the date when the aircraft landed is listed in the Arrival field.

If you use Origin as the Row Key, you will be able to pull out all flights from Atlanta -- but the Arrival field will not necessarily be in order. So that means searching through the column to produce the solution set.

If you use the Arrival field as the Row Key, it will be easy to pull out all flights between March 21st and 29th, but the airport of origin won't be organized. So you will be searching through the arrival column to produce the solution set.

In the third example, a Row Key has been constructed from information extracted from the Origin field and the Arrival field -- creating a constructed Row Key. Because the data is organized lexicographically by the Row Key, all the Atlanta flights will appear in a group, and sorted by date of arrival. Using this Row Key you can generate the solution set with only a scan.

In this example, the data was transformed when it arrived. So constructing a Row Key during the transformation process is straightforward.

Cloud Bigtable schema organization

Column Families

Row Key	Flight_Information					Aircraft_Information			
	Origin	Destination	Departure	Arrival	Passengers	Capacity	Make	Model	Age
ATL#arrival#20190321-1121	ATL	LON	20190321-0311	20190321-1121	158	162	B	737	18
ATL#arrival#20190321-1201	ATL	MEX	20190321-0821	20190321-1201	187	189	B	737	8
ATL#arrival#20190321-1716	ATL	YVR	20190321-1014	20190321-1716	201	259	B	757	23

Google Cloud

Cloud Bigtable also provide Column Families. By accessing the Column Family, you can pull some of the data you need without pulling all of the data from the row or having to search for it and assemble it.
This makes access more efficient.

https://cloud.google.com/bigtable/docs/schema-design

When to use something else...

When should you choose disk storage?

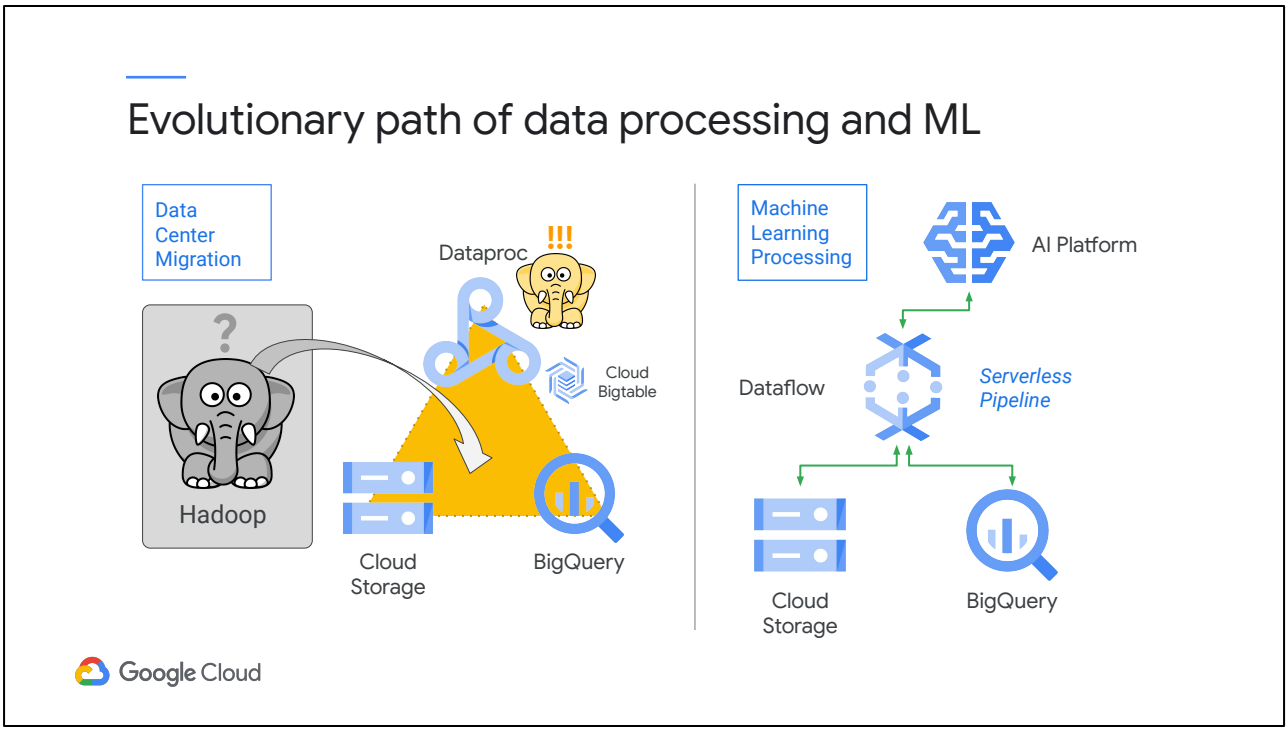
When should you choose mobile storage?

Google Cloud

TIP: Fast, temporary, data -- maybe a local disk.

https://cloud.google.com/compute/docs/disks/

Standard and Regional HDDs and SSDs are now supported. Regional versions have zonal redundancy.



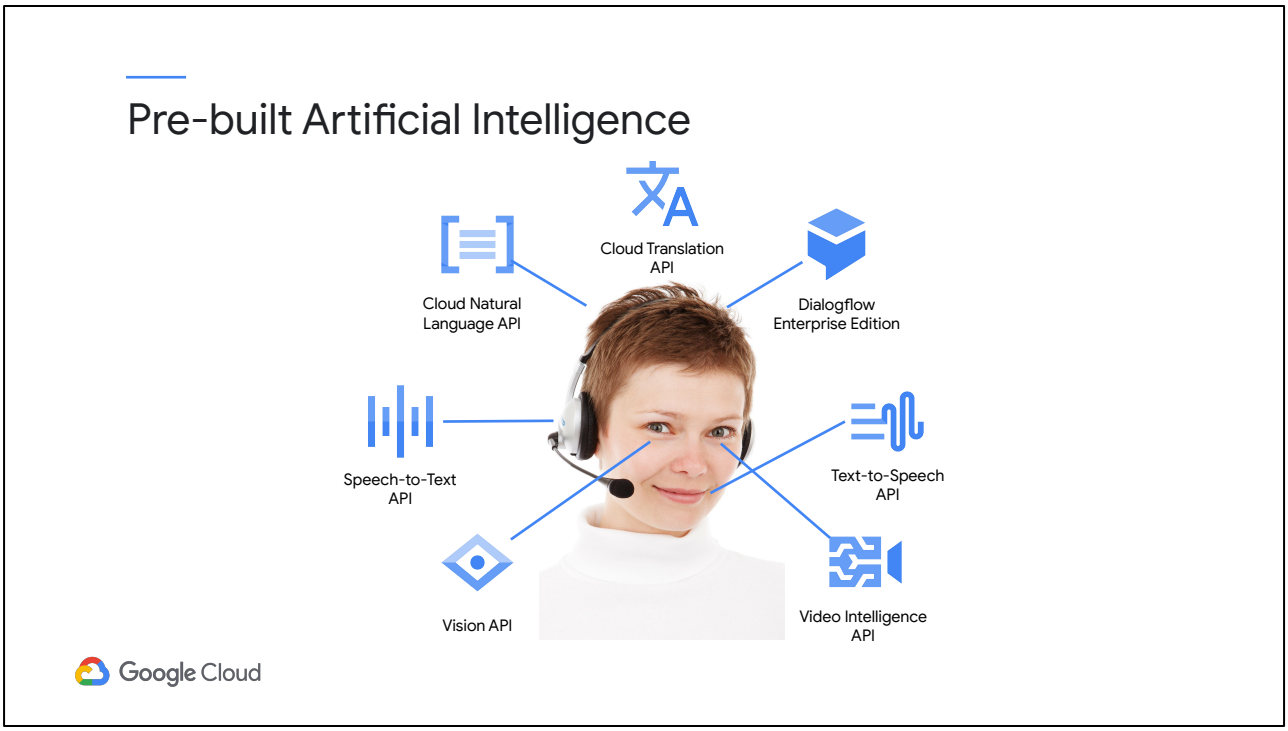
Data Center Migration.
BigQuery provides a front end for analysis and a back end that can read from several sources including BigQuery tables, but also CSV files in Cloud Storage. Dataproc is a managed service for Hadoop clusters - useful for processing data and returning it to Cloud Storage or BigQuery.

The first step is migration from Data Center processing to Cloud Data processing. BigQuery replaces many tools and custom applications in the data center, while Dataproc replaces Hadoop. Cloud BigTable is a "drop in" replacement for HBase. Machine Learning is available from Dataproc using APIs such as Natural Language Processing (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.

https://cloud.google.com/products/data-transfer/



TIP: Machine Learning is more of a focus of the Data Engineering track rather than the Cloud Architect track, but it is still part of the infrastructure a Cloud Architect might use to define solutions. So you should be familiar with the services and what they do.

Pre-built models are offered as services. In many cases these building blocks can be used to create the application you want without the expense or complexity of creating your own models.

The Speech-to-Text API converts audio to text for data processing. The Cloud Natural Language API recognizes parts of speech called entities and sentiment. The Cloud Translation API converts text in one language to another.

Dialogflow Enterprise Edition is used to build chatbots to conduct conversations. The Text-to-Speech API converts text into high quality voice audio. Vision API is for working with and recognizing content in still images. And the Video Intelligence API is for recognizing motion and action in video.

AutoML builds custom AI models

AutoML Natural Language

AutoML Translation

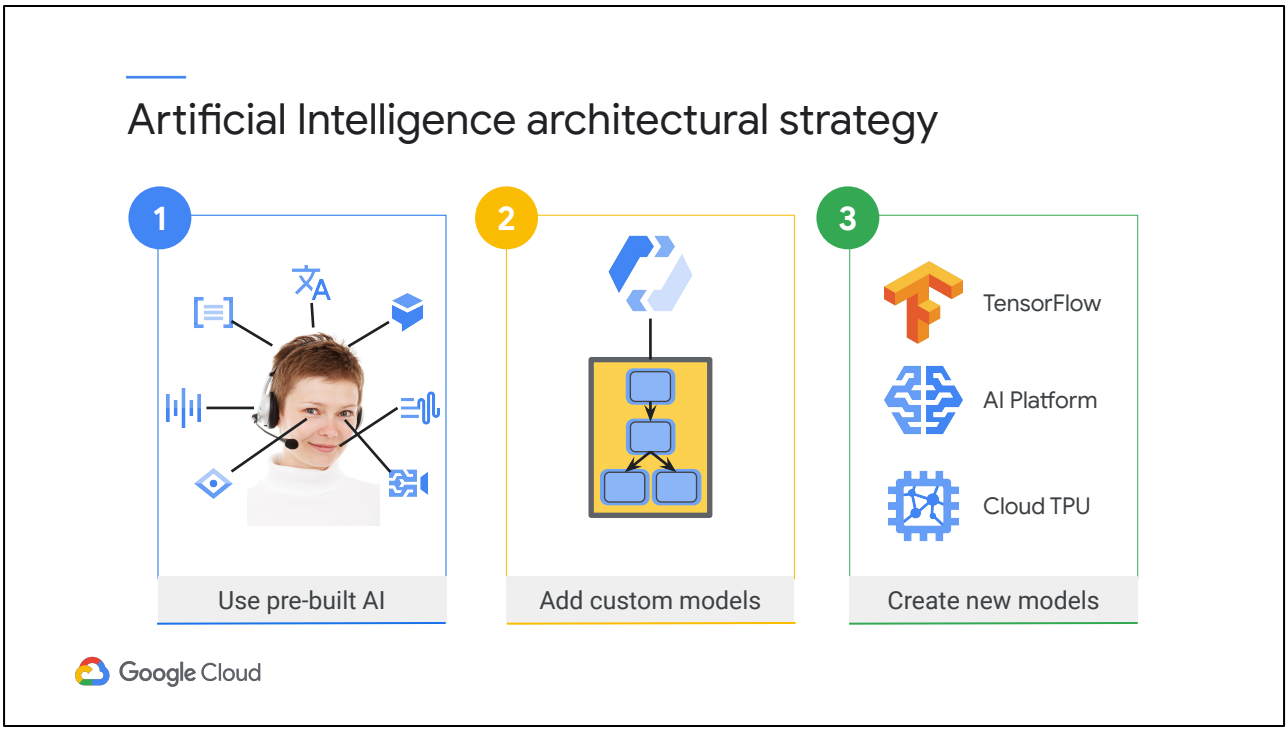
AutoML Vision

Google Cloud

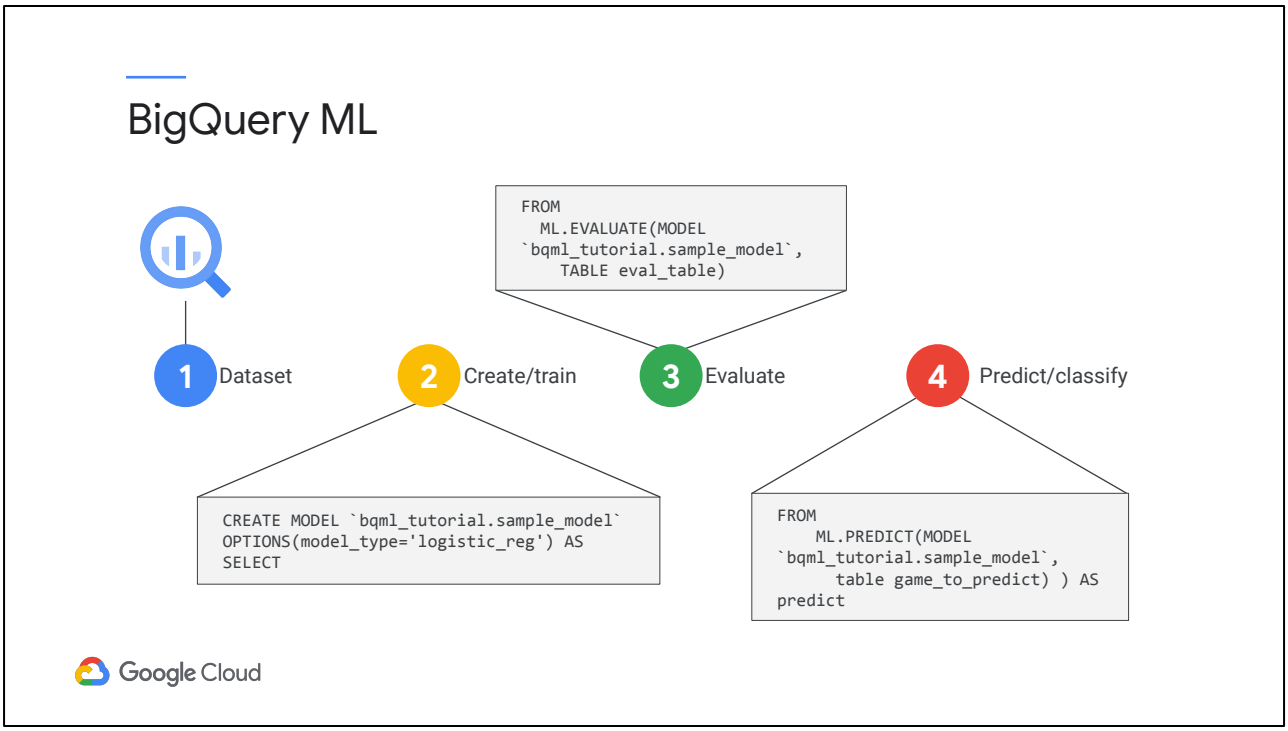
AutoML Natural Language classifies English content into custom categories.

AutoML Translation enables models that translate into words and phrases specific to your domain.

AutoML Vision classifies images according to your defined labels.



The recommended strategy is to first, use the pre-built artificial intelligence services. Next, you can use AutoML to produce Custom Models which can be used with the pre-built services or on their own. Remember that you can divide a problem into specialized parts and use multiple Custom Models together. Finally, if you discover you need more advanced features, you can use the Machine Learning and Artificial Intelligence services to create new models.



There are two key benefits of BigQuery ML. First, it works on your data that is already in BigQuery storage -- in your Data Warehouse or Data Lake.

So the data does not need to be copied out of BigQuery or reformatted. This improves efficiency in many ways, and in some cases resolves privacy concerns.

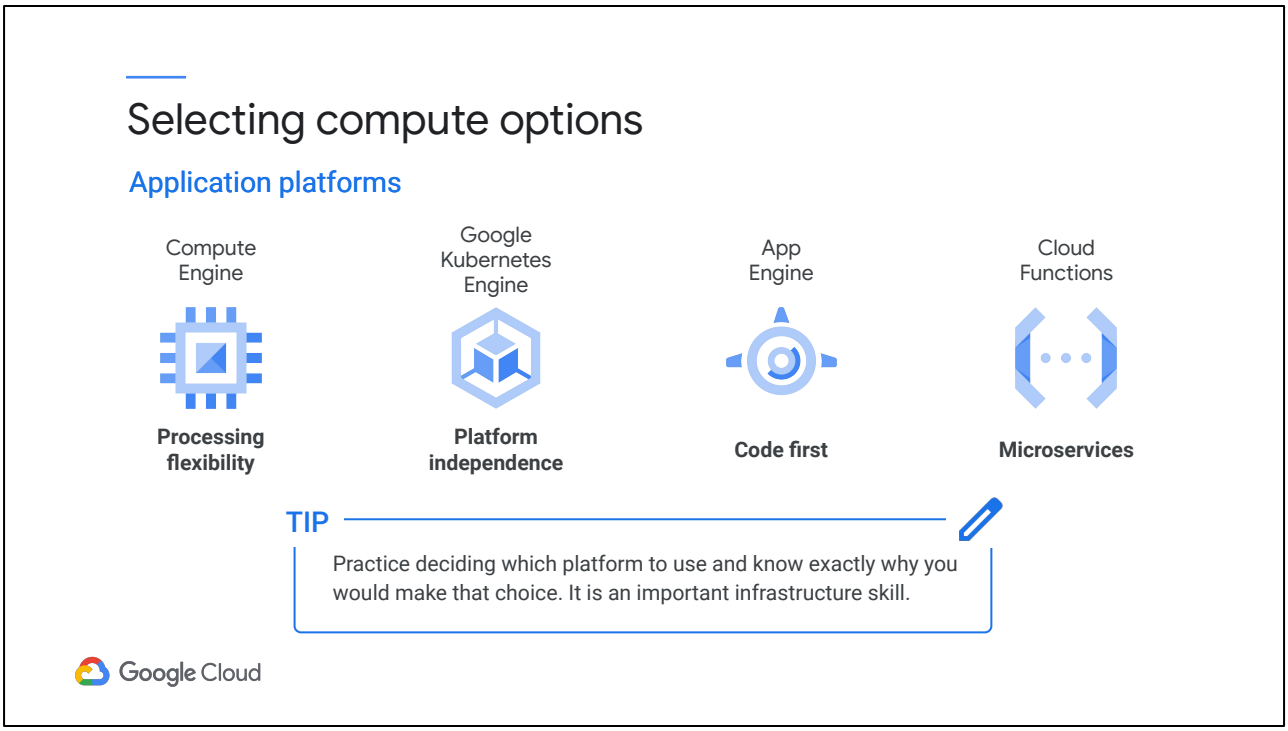
Second, BigQuery ML uses SQL commands. That means BigQuery users can learn to create their own models by extending their use of a language they already know.

Works with full models (not Custom Models like AutoML).

The locality of your model is the same as your dataset. So that is defined when the dataset is created.

Configuring compute systems

Exam outline	Tips
Compute system provisioning	Speed, throughput, capacity, burstiness? Turn off when not in use?
Compute volatility configuration (preemptible versus standard)	Stateful versus stateless Tolerance for lost data/state
Network configuration for compute nodes	Firewall rules, load balancing, NAT, VPN
Orchestration technology configuration (e.g. Chef/Puppet/Ansible/Terraform)	Deployment Manager … others The value of self-documenting and automated orchestration of infrastructure
Container orchestration (e.g. Kubernetes)	The value of container-based development; portability, scalability, fault isolation, platform independent deployment, CI/CD development methods



TIP: There is 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 support staff. Fewer items in Kubernetes are 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 or support of 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. And finally, Cloud Functions provides a microservices approach which means creating small stateless elements. Very fast and scalable. But you have to be diligent about isolating state. And there is additional programming overhead involved.

https://cloud.google.com/compute/docs/

https://cloud.google.com/kubernetes-engine/docs/

https://cloud.google.com/appengine/docs/

https://cloud.google.com/appengine/docs/the-appengine-environments

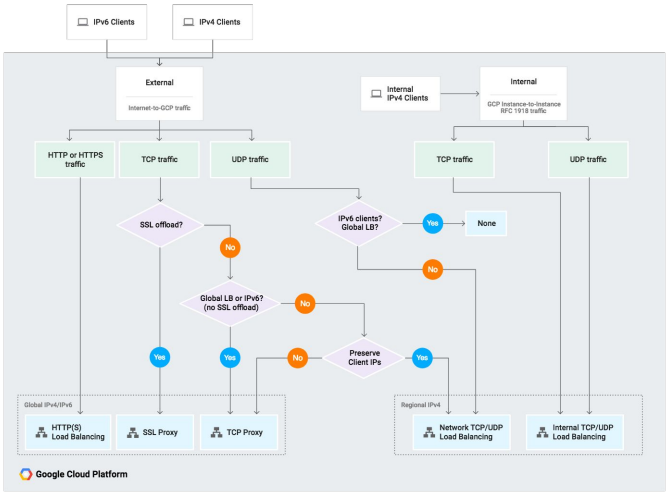
https://cloud.google.com/functions/docs/

Comparing compute options: Technical details

	Compute Engine	Google Kubernetes Engine	App Engine flexible environment	App Engine standard environment	Cloud Functions
Service model	IaaS	Hybrid	PaaS	PaaS	Serverless
Languages supported	Any	Any	Java, Python, Node.js, Ruby, PHP, .NET Core, Go; or supply a runtime	Java, Go, Python, PHP	Node.js
Use cases	General computing workloads	Container-based workloads	Web and mobile applications; container-based workloads	Web and mobile applications	Ephemeral functions responding to events

Choosing load balancing for Compute Engine

- Global versus regional load balancing
- External versus internal load balancing
- Traffic type



Google Cloud

https://cloud.google.com/load-balancing/docs/

https://cloud.google.com/load-balancing/docs/choosing-load-balancer

Choosing instance groups for Compute Engine

Type of Instance Group		Properties of Instances	Feature
Unmanaged		Heterogeneous	
Managed		Homogeneous	Instance Templates Autoscaling
	Zonal	Same zone	Latency consistency
	Regional	Different zones	Reliability

Google Cloud

Unmanaged Instance Groups collect different kinds of instances. Usually this is done for management of "lift and shift" existing designs, and it is not recommended because it does not make the best use of the features available in cloud.

Managed Instance Groups are all the same kind of instance, meaning that the type can be defined by an Instance Template and Autoscaling 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 instances in multiple zones within the region, increasing reliability.

https://cloud.google.com/compute/docs/instance-groups/

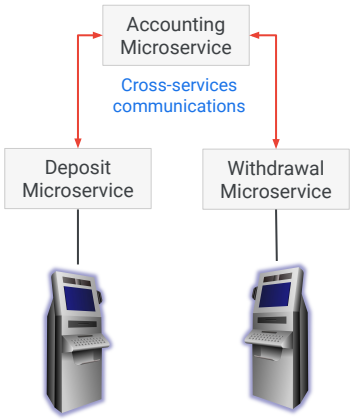
https://cloud.google.com/compute/docs/autoscaler/

Microservices: Stateless design complicates logic

TIP

Microservices are not a panacea. They are not the best solution in every case.

Make sure you know where the edge cases are between technologies.



Google Cloud

TIP: Microservices is not a panacea! It does not fit in all cases. You can implement a microservices solution in App Engine, Cloud Functions, and using Node.js in Kubernetes. The platforms have overlapping coverage. Do you know when you might choose one platform over another for a microservices solution?

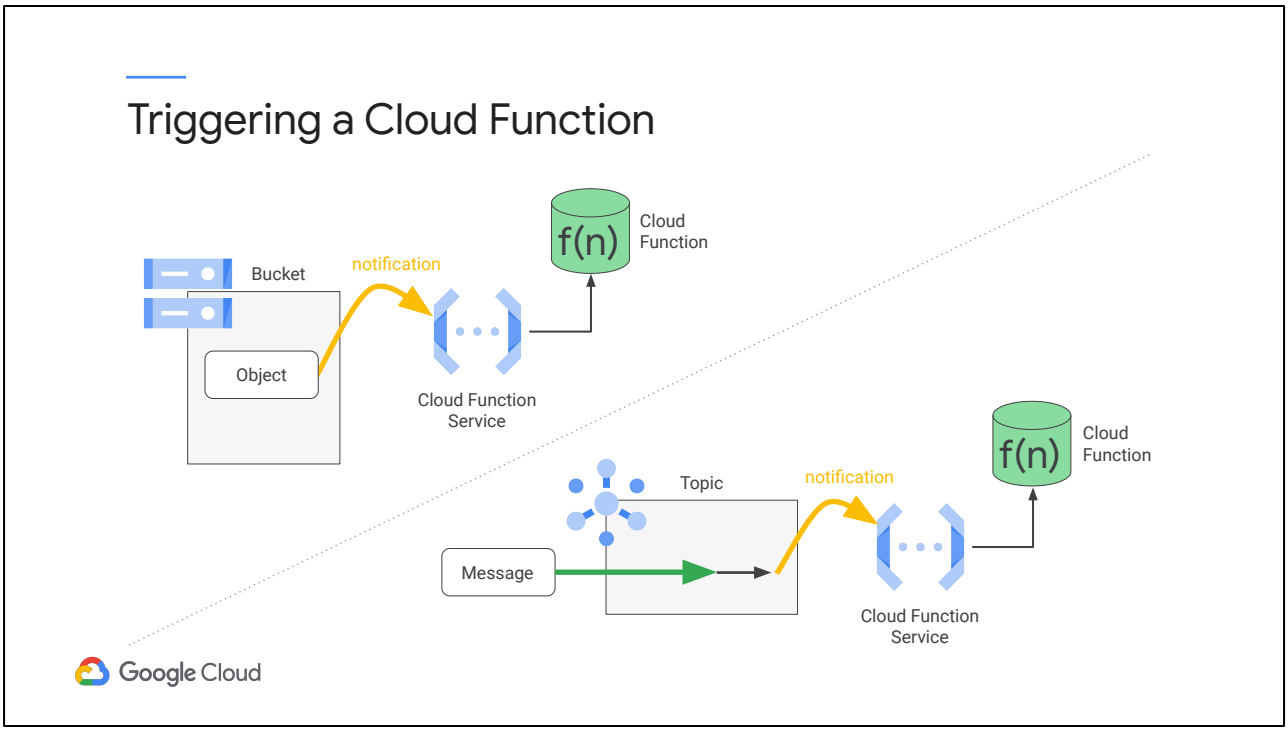
https://cloud.google.com/appengine/docs/standard/python/microservices-on-app-engine

https://cloud.google.com/community/tutorials/developing-services-with-k8s

https://cloud.google.com/functions/

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 implemented in Cloud Functions or in App Engine.



A Cloud Function is a serverless, stateless, execution environment for application code. You deploy your code to the Cloud Functions service and set it up to be triggered by a class of events. Mobile application developers use the HTTP (web) event. but you can also use events that are associated with Cloud Storage or Pub/Sub.

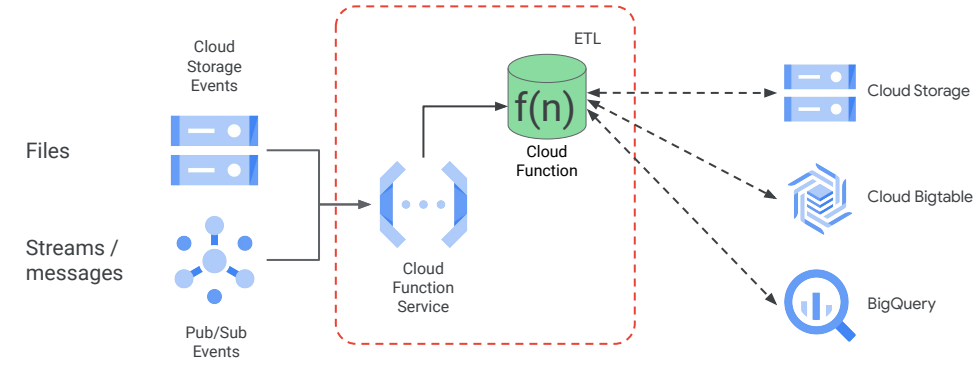
Cloud Functions can be triggered by A user-defined HTTP callback (a webhook), which is commonly used for mobile and web development.

However, Cloud Functions can also be triggered by Pub/Sub messages and by events occuring in Cloud Storage.

How it works:

Authentication
Send watch request
Sync notification event
add, update, remove object
Notification
Waits for acknowledgement
If the app is unreachable for 20 seconds, the notification is retired.
If the app is reachable, but does not acknowledge, then exponential backoff 30 seconds after fail up to max 90 minutes for up to 7 days

Example of using Cloud Functions for data ingress



Google Cloud

When the event occurs, it triggers the Cloud Function to run. Each time an event occurs and the function is run, it is a fresh instance without history. For example, if you wanted to create a Cloud Function that counts the number of times it is called, it would have to store that counter information externally, such as in Cloud Storage. When you deploy a Cloud Function, you can specify requirements so that common libraries are loaded into the environment. Because Cloud Functions are lightweight and stateless, you can construct microservices applications that are highly scalable.

In the illustration, the Cloud Function uses APIs to work with common data storage components. For example, it might extract metadata from image files uploaded to Cloud Storage and save the metadata in BigQuery for analysis.

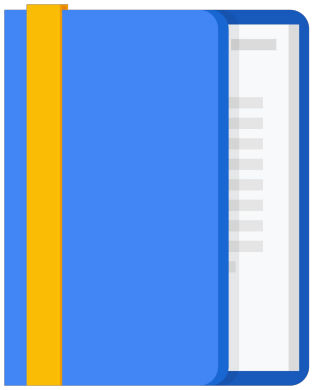
Cloud Functions has integration with Google Cloud's operations suite so you can monitor your application.

Module agenda

Designing and Planning a Cloud Solution Architecture

Managing and Provisioning Solution Infrastructure

Containers and Google Kubernetes Engine

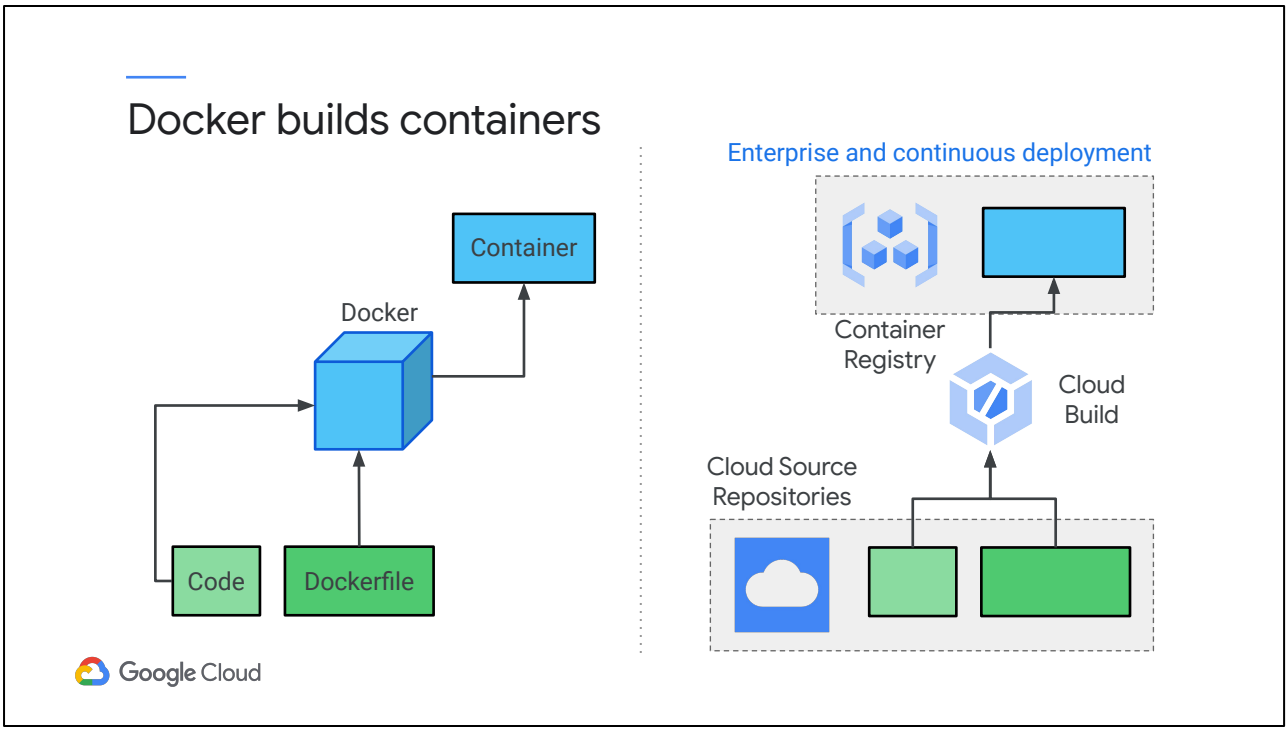


Google Cloud

Between 2017 and 2018 the number of organizations using containers for software development and to deploy their services approximately doubled.

And the trend shows no signs of slowing. For this reason, container knowledge and skill with Kubernetes is increasingly important for the job of a Cloud Architect.

And, of course, if you need more of these skills for the job, you will also need them to prepare for the exam.



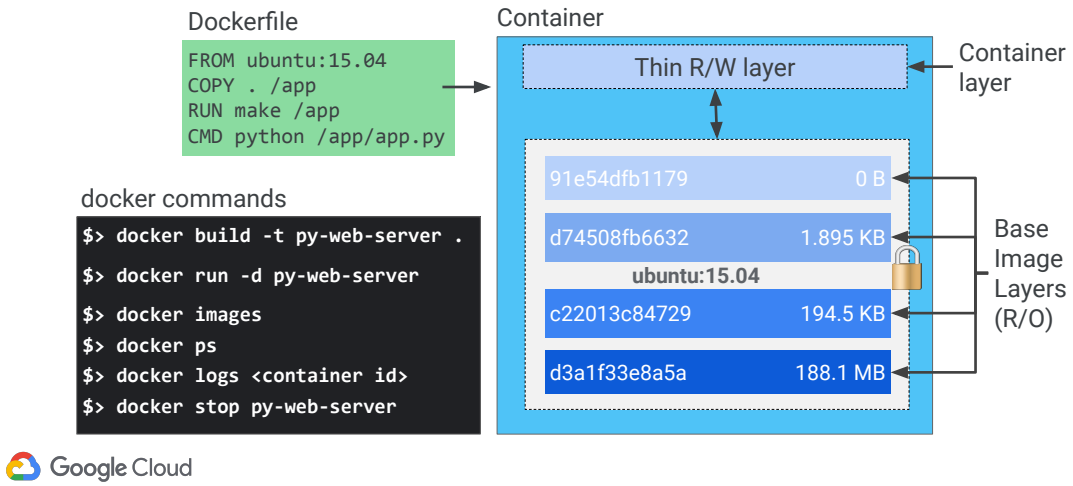
Docker is software that builds containers. You supply application code, and instructions, called a Dockerfile, and Docker follows the instructions and assembles the code and dependencies into the container. Containers can be "run", much as an application can run. However, it is a self-contained environment 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. And it is designed to scale to handle many application developers working on and continuously updating a live global service.

If you had a hundred developers sharing source files, you would need a system for managing them, for tracking them, versioning them, and enforcing a check-in, review, and approval process. Cloud Source Repositories is a cloud-based solution.

If you were deploying hundreds of containers you would not being 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. And this is the purpose of Container Registry. Container Registry has various integrations with Continuous Integration / Continuous Deployment services.

What is really in a container...



A docker container is an image built in layers. Each layer is created by an instruction in the dockerfile.

All the layers except for the top one are 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 layered design inside of a container isolates functions. This is what makes the container stable and portable.

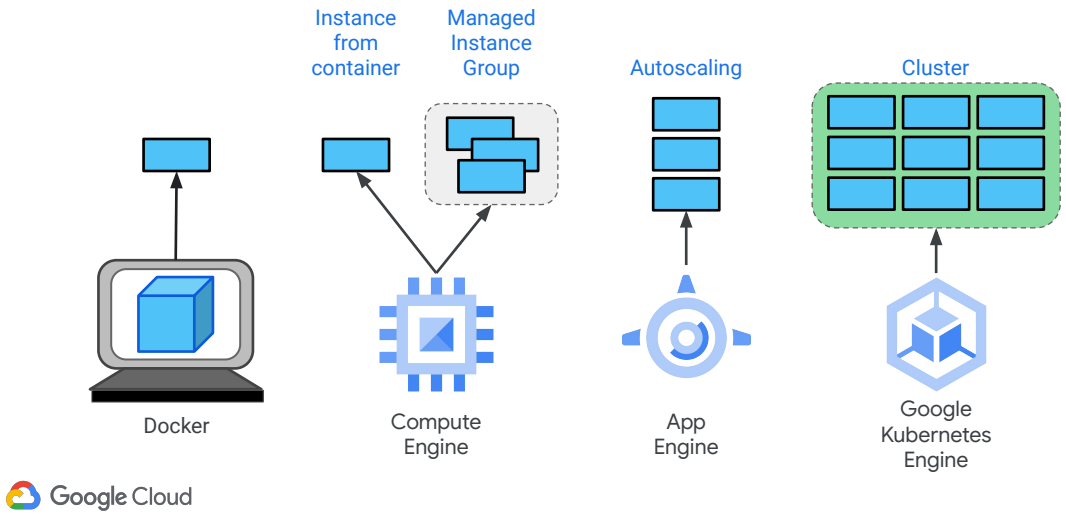
Here are a few of the common docker commands.

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.

Where can you run containers?



You can run a container in Docker itself, as you saw with the "docker run" command.

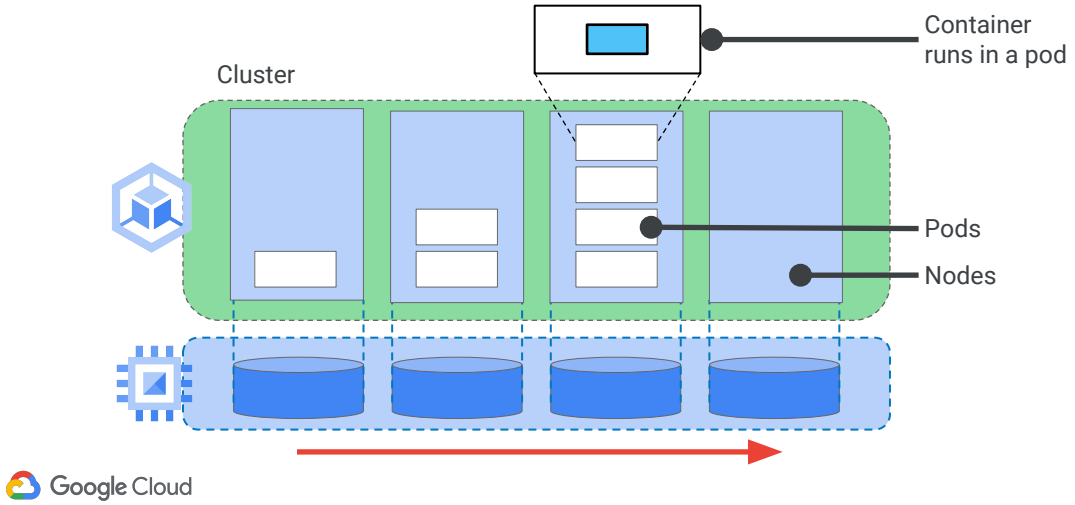
You can also run containers using Compute Engine. Compute Engine gives you the alternative to start up a virtual machine from a container rather than from an OS Image boot disk. You also have this option when creating an instance template, which means you can create Managed Instance Groups from containers.

App Engine supports containers as custom runtimes. The main difference between the App Engine standard environment and the App Engine flexible environment is that the flexible environment 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 open standard software, so you can run a Kubernetes cluster in your data center. Google Kubernetes Engine provides Kubernetes as a managed service.

Kubernetes cluster has nodes, pods, and containers



Google Cloud

A Kubernetes cluster is composed of nodes, which are a unit of hardware resources. Nodes in GKE 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.

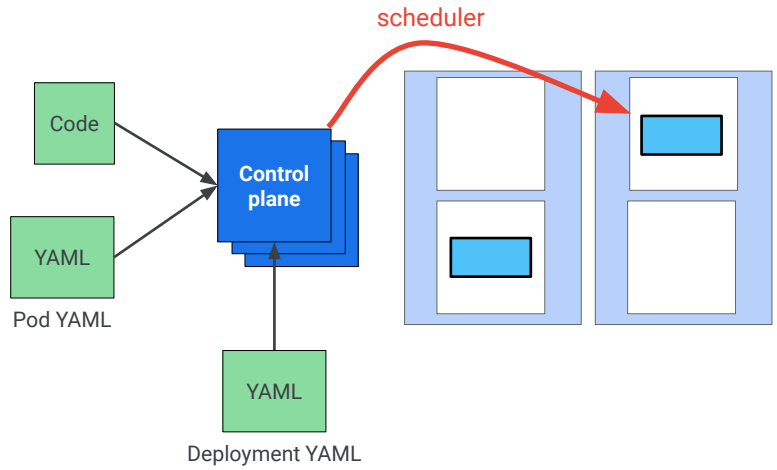
Each pod hosts, manages, and runs one or more containers. The containers in a pod share networking and storage.

So typically, there is one container per pod, unless the containers hold closely related applications. For example, a second container might contain the 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.

Kubernetes jobs run containers on nodes

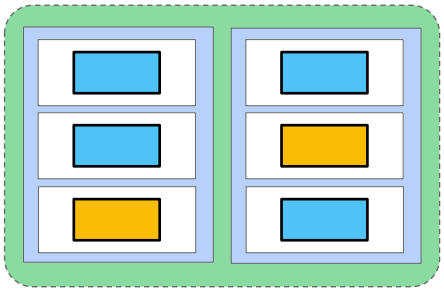


Google Cloud

Each cluster has a control plane 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 is 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.

Advanced operations: A/B testing, rolling updates



Google Cloud

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 A in production on most of the pods and try out version B with a sample group in other pods. This is called A/B testing and it is great because you can test the new software in the 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. And 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.

Balance resiliency, scalability, cost

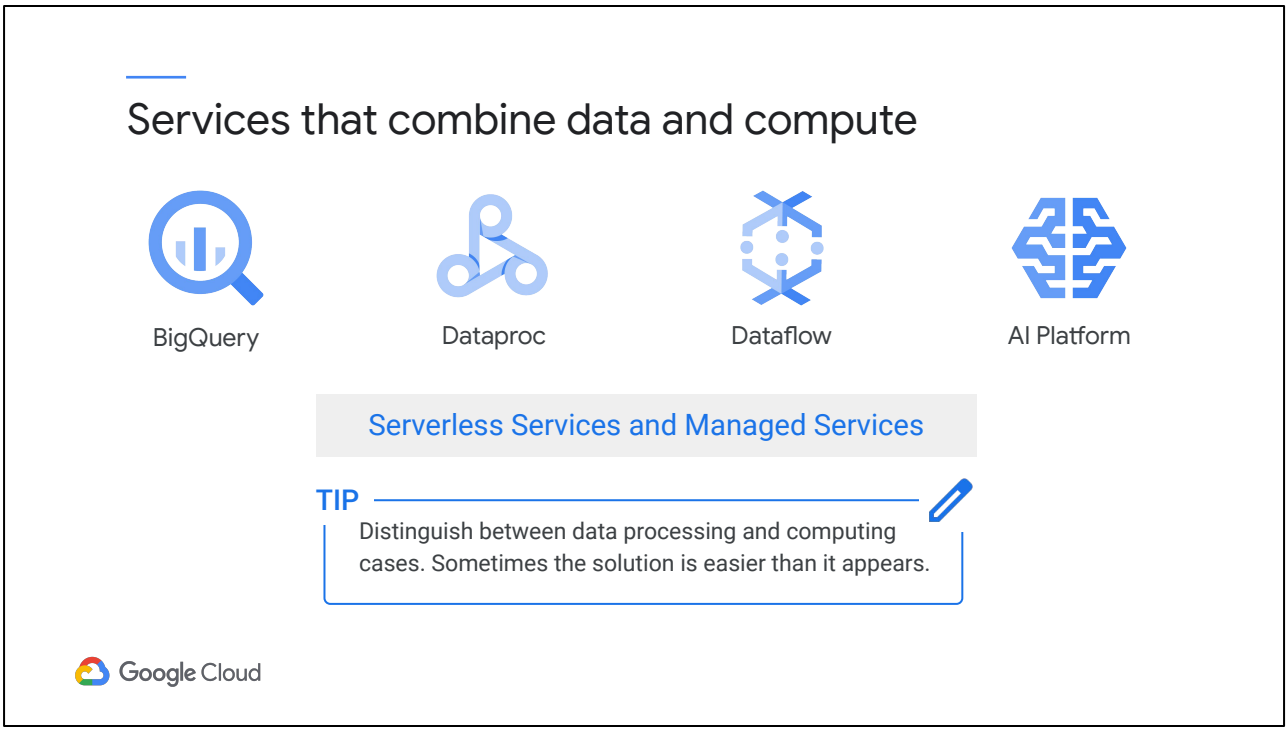
Small stateless servers increase reliability and scalability	Large stateful servers reduce complexity and latency
Divide into parts	Unify
Duplicate and coordinate	Simplify and consolidate
Separate and isolate	Coalesce and colocate

Methods of achieving balance in your design

- What are your SLOs? What do your users value?
- What is the optimal size and number of parts?
- Sometimes central control is necessary/optimal
- Plan on adjusting, and build adjustment processes

TIP

Make sure you know the business goals of the exam question, and not just the technical requirements.



TIP: Services, Managed Services, and Serverless Services -- knowing the difference 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. Great for popular use-cases and eliminating overhead.

Serverless Services completely hide all servers. Generally, more fast, scalable, and efficient than you could create on your own.
The key tradeoff: they are proprietary.

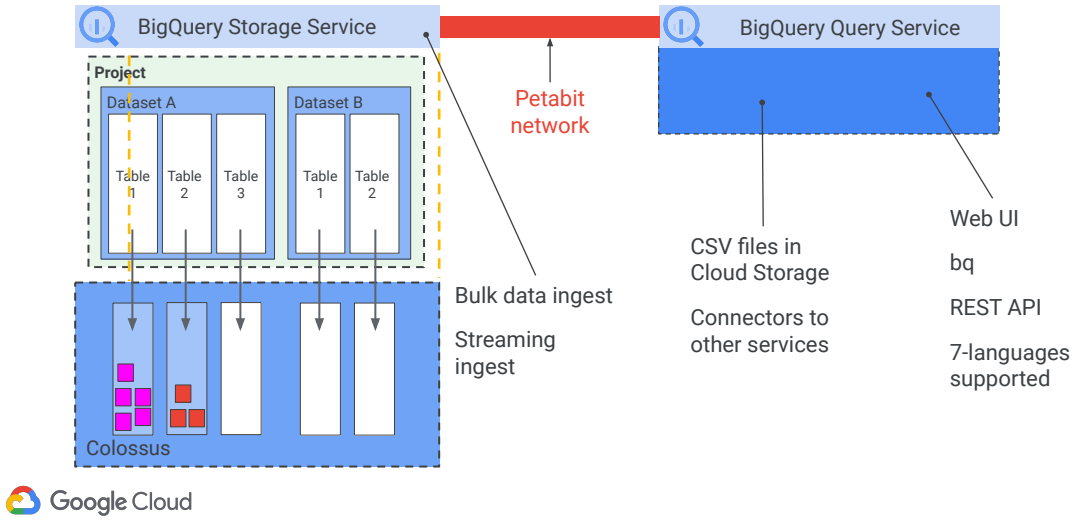
https://cloud.google.com/bigquery/docs/

https://cloud.google.com/dataproc/docs/

https://cloud.google.com/dataflow/docs/

https://cloud.google.com/ml-engine/docs/tensorflow/technical-overview

BigQuery is a cloud data warehouse service



Google Cloud

BigQuery is a data warehouse service. It is a "serverless" service, meaning that it is fully managed. So users do not have visibility or control over individual servers or clusters of servers. BigQuery runs data processing jobs that can load, export, copy or query data.

BigQuery has two parts, a storage service and a query service, which work together. They are connected by Google's high speed internal network.

The storage service manages the data. Data is contained within a project in datasets in tables. The tables are stored as highly compressed columns in Google's Colossus file system which provides durability and availability.

BigQuery Storage Service automatically shards and shuffles data in the underlying file system to provide a very high level of service at huge scales. The sharding occurs automatically and provides the advantages of data distribution while completely concealed from you at the Dataset and Table level.

The storage service supports bulk data ingest and streaming ingest. So it can work with huge amounts of data and also real-time data streams.

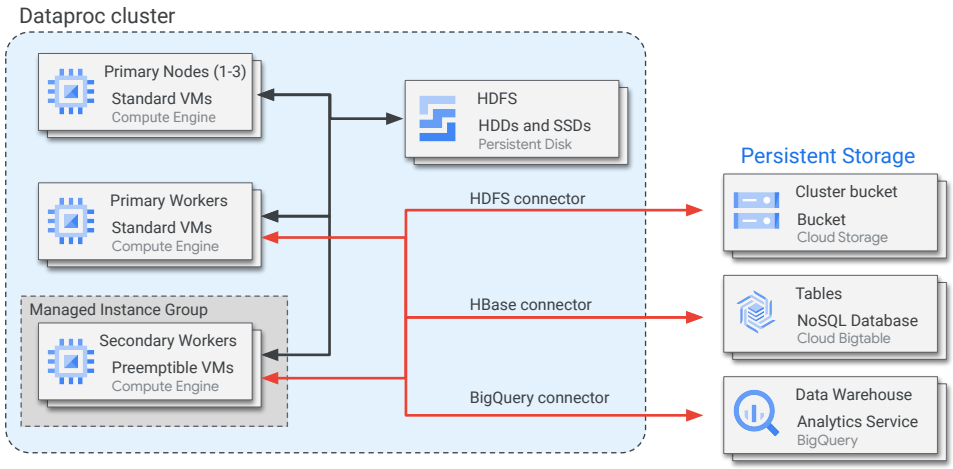
The query service runs interactive or batch queries that are submitted through console, the BigQuery web UI, the bq command line tool, or via REST API. The REST API is supported for seven programming languages.

There are connectors to other services such as Dataproc which simplify creating complex workflows between BigQuery and other Google Cloud data processing services. The query service can also run query jobs on data contained in other locations, such as tables in CSV files hosted in Cloud Storage.

BigQuery is most efficient when working with data contained in its own storage service. The storage service and the query service work together to internally organize the data to make queries efficient over huge datasets of Terabytes and Petabytes in size.

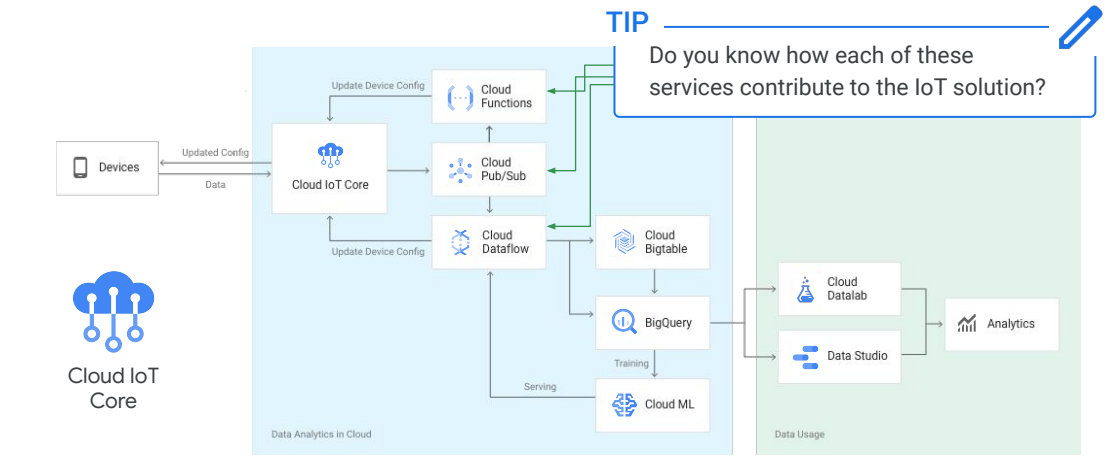
The most important control over resource consumption and costs is writing a query that controls the amount of data processed. In general, this is done with SELECT by choosing subsets of data at the start of a job rather than by using LIMIT which only omits data from the final results at the end of a job.

Dataproc does processing and transformation



Google Cloud

Example: Internet of things (IoT)



Google Cloud

TIP: There are common "assemblies" of services that work together. If you review solutions and diagrams you will start to recognize them.
This is a great resource for becoming familiar with a variety of solution architectures: https://cloud.google.com/solutions/

IoT Core Device Manager
IoT Core provides a fully managed service for managing devices. This includes registration, authentication, and authorization inside the Google Cloud resource hierarchy as well as device metadata stored in the cloud, and the ability to send device configuration from the service to devices.

Pub/Sub
Pub/Sub provides a globally durable message ingestion service. By creating topics for streams or channels, you can enable different components of your application to subscribe to specific streams of data without needing to construct subscriber-specific channels on each device. Pub/Sub also natively connects to other Google Cloud services, helping you to connect ingestion, data pipelines, and storage systems.

Pub/Sub can act like a shock absorber and rate leveller for both incoming data streams and application architecture changes. Many devices have limited ability to store and retry sending telemetry data. Pub/Sub scales to handle data spikes that can occur when swarms of devices respond to events in the physical world, and buffers these spikes to help isolate them from applications monitoring the data.

Dataflow
Dataflow provides the open Apache Beam programming model as a managed service for processing data in multiple ways, including batch operations, extract-transform-load (ETL) patterns, and continuous, streaming computation. Dataflow can be particularly useful for managing the high-volume data processing pipelines required for IoT scenarios. Dataflow is also designed to integrate seamlessly with the other Google Cloud services you choose for your pipeline.

Cloud Functions
IoT events and data can be sent to the cloud at a high rate and need to be processed quickly. For many IoT applications, the decision to place the device into the physical environment is made in order to provide faster access to data. For example, produce exposed to high temperatures during shipping can be flagged and disposed of immediately.

Being able to process and act on this information quickly is key. Cloud Functions allows 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.

https://cloud.google.com/solutions/iot-overview

https://cloud.google.com/pubsub/docs/overview

Scenario #1

Question

Which network feature could help a company meet its goals to expand service to Asia while reducing latency?

A. Cloud NAT

B. Network TCP/UDP

C. Cloud Router

D. Cloud CDN

Scenario #1

Answer

Which network feature could help a company meet its goals to expand service to Asia while reducing latency?

A. Cloud NAT

B. Network TCP/UDP

C. Cloud Router

D. Cloud CDN

Google Cloud

Scenario #1

Rationale

D - Cloud CDN will enable a company to expand its online presence with a single IP address and global reach leveraging Google's global network.

A - Cloud NAT enables you to provision your application instances without public IP addresses, while also allowing them to access the internet, but it won't reduce the latency associated with web and video content delivery, with global scale and reach.

B - Network load balancing is internal only. It can help scale the service but not necessarily expand its reach.

C - Cloud Router discovers changed network topology using BGP. That might be valuable if they open offices in Asia, but that doesn't directly expand the services.

https://cloud.google.com/cdn/docs/
https://cloud.google.com/cdn/docs/overview
https://cloud.google.com/cdn/docs/caching

Scenario #2

Question

How can you minimize the cost of storing security video files that are processed repeatedly for 30 days?

A. Standard Storage, then move to Coldline Storage or Archive Storage after 30 days.

B. Nearline Storage, then move to Coldline Storage after 30 days.

C. Standard Storage, then move to Nearline Storage after 30 days.

D. Keep the files in Standard Storage.

Origin: CAPE

Scenario #2

Answer

How can you minimize the cost of storing security video files that are processed repeatedly for 30 days?

A. Standard Storage, then move to Coldline Storage or Archive Storage after 30 days.

B. Nearline Storage, then move to Coldline Storage after 30 days.

C. Standard Storage, then move to Nearline Storage after 30 days.

D. Keep the files in Standard Storage.

Scenario #2

Rationale

A - Standard Storage for lowest access costs over the 30 days, then Coldline Storage or Archive Storage because it is unlikely to be read after the 30 days.

B - Using Nearline Storage over the 30 days won't be cost effective because the data is accessed too frequently. There is also a 30 day minimum storage duration.

C - Moving from Standard Storage to Nearline Storage after the 30 days isn't as cost effective as Coldline Storage or Archive Storage if the data is not going to be accessed that frequently.

D - Keeping the data in Standard Storage is the least cost effective option if it is not going to be accessed frequently after 30 days.

https://cloud.google.com/storage/docs/storage-classes

A Challenge Lab has minimal instructions. It explains the circumstance and the expected results -- you have to figure out how to implement them.

This is a timed lab.
The lab will expire after 45 minutes.
The lab *can* be completed in 30 minutes.


Google Cloud