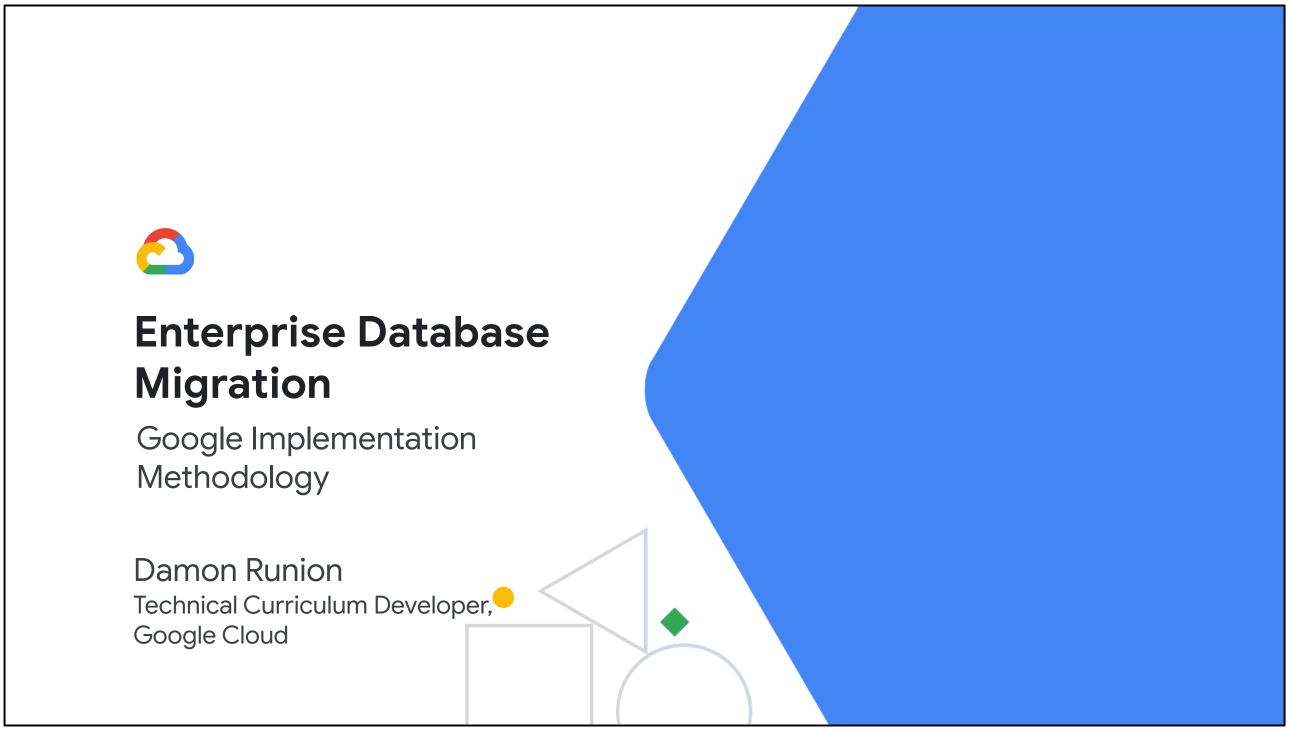


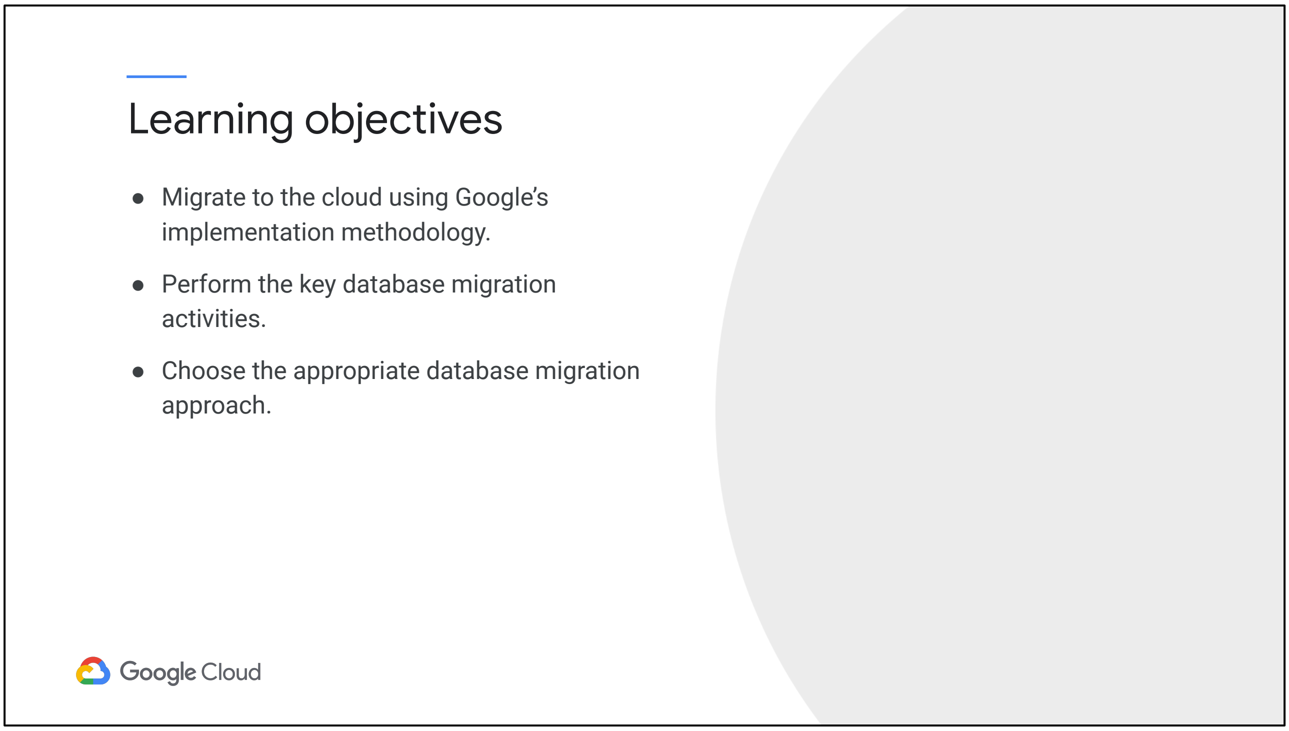
Enterprise Database Migration

Google Implementation Methodology

Damon Runion
Technical Curriculum Developer,
Google Cloud

Hi everyone, Damon here. Migrating to the cloud can go smoothly, or it can be filled with delays, bugs, downtime, and cost overruns. This is especially true when you are moving databases.

Over time, it's easy to lose track of database dependencies, and when you switch the old database off, unexpected things will fail. Careful planning can help mitigate this risk. Google has helped many enterprises move to the cloud and has developed an implementation methodology based on this experience.

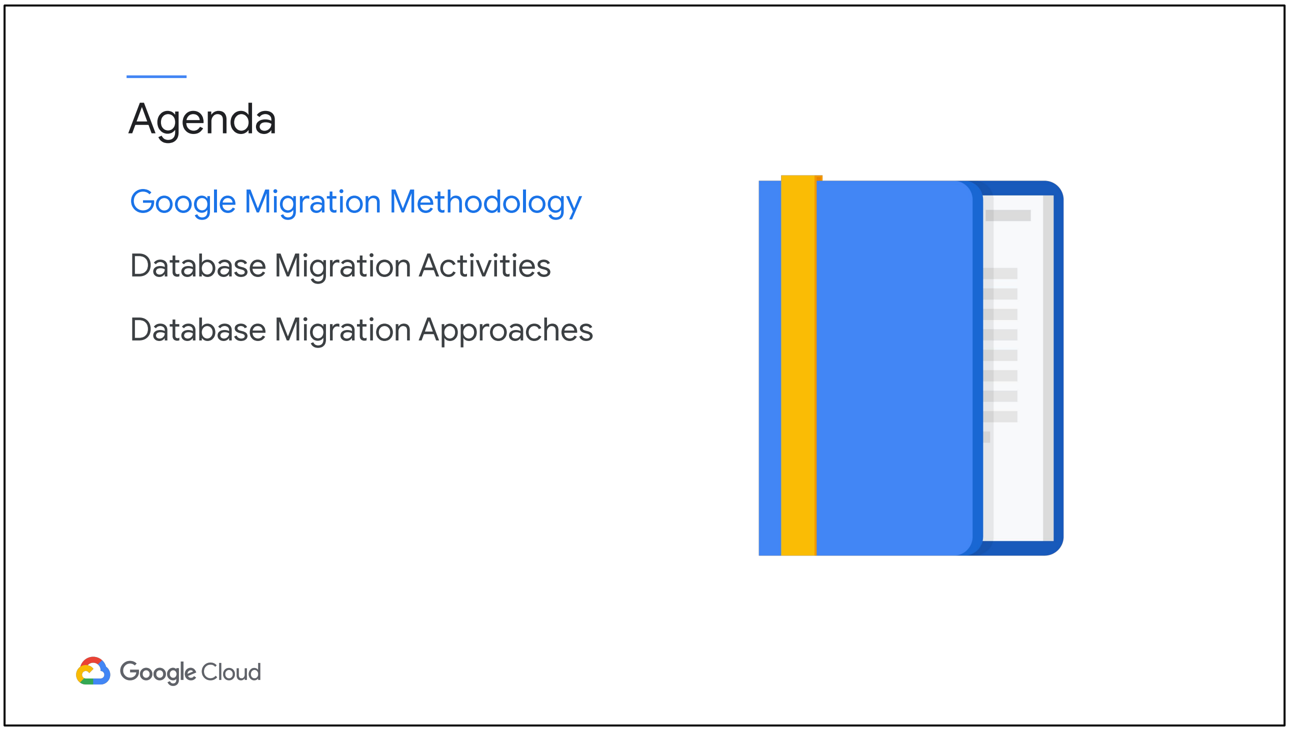


Learning objectives

- Migrate to the cloud using Google's implementation methodology.
- Perform the key database migration activities.
- Choose the appropriate database migration approach.

Google Cloud

- In this module, you learn to use the Google Cloud implementation methodology to migrate to the cloud.
- You learn what are the key database migration activities
- Lastly you learn how to choose the appropriate database migration approach.



Agenda

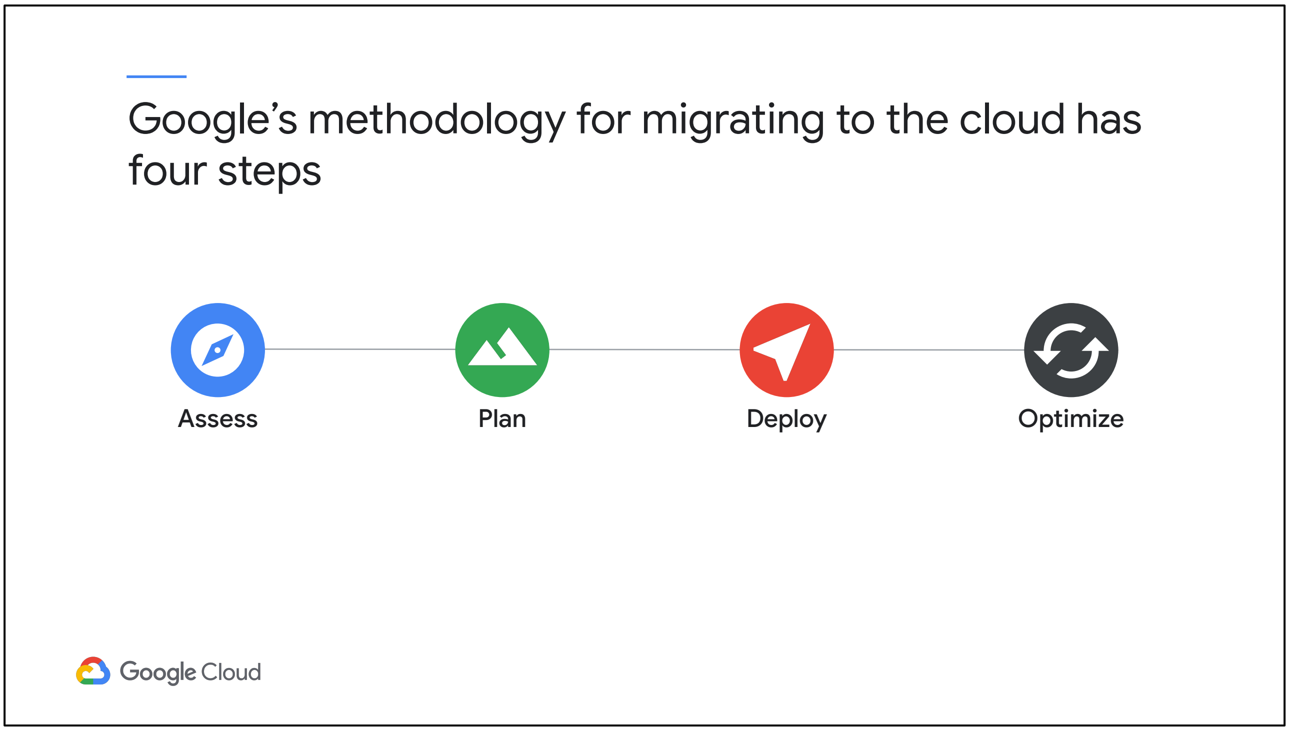
Google Migration Methodology

Database Migration Activities

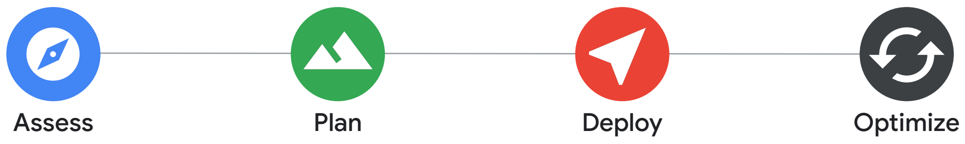
Database Migration Approaches

Google Cloud

Let's start by defining the Google migration methodology.

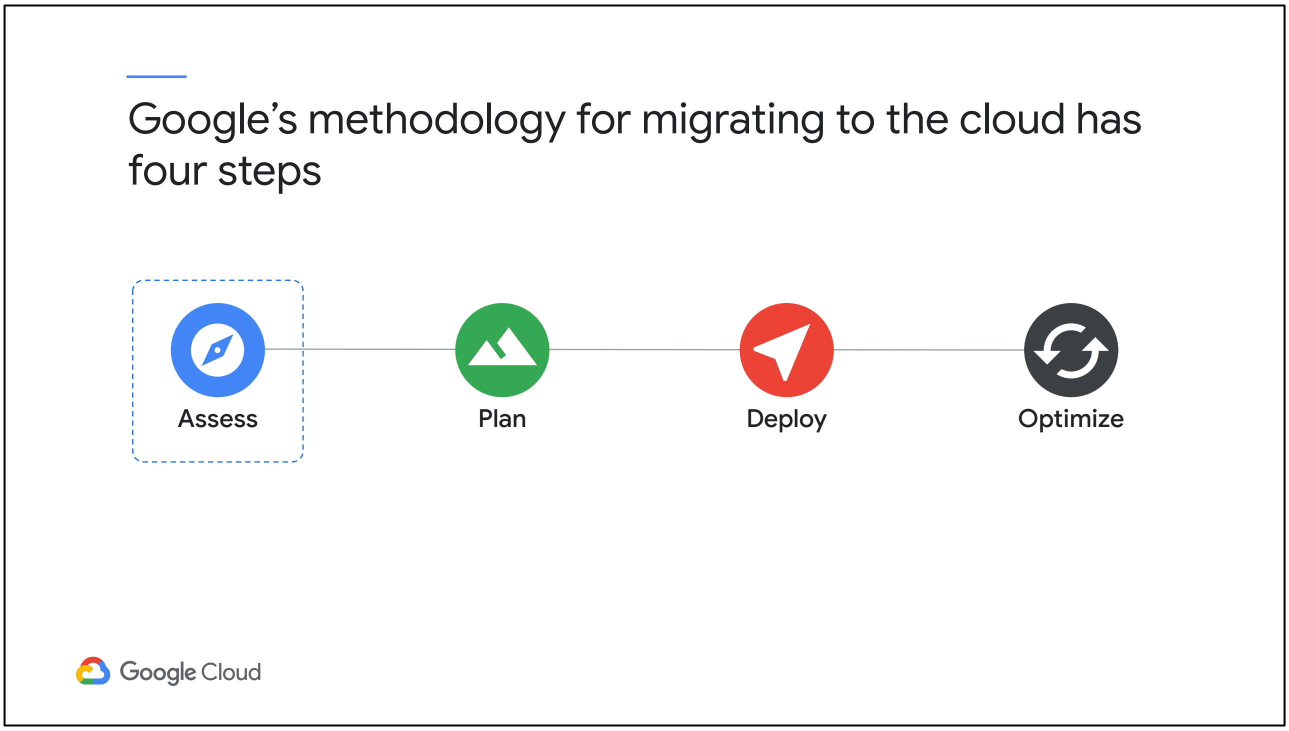


Google's methodology for migrating to the cloud has four steps

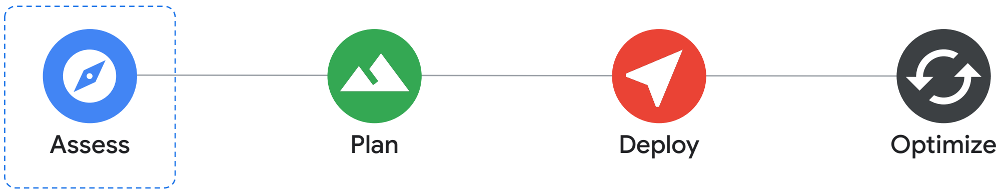


Google Cloud

Google's methodology for migrating to the cloud has four steps: Assess, plan, deploy, and optimize.



Google's methodology for migrating to the cloud has four steps

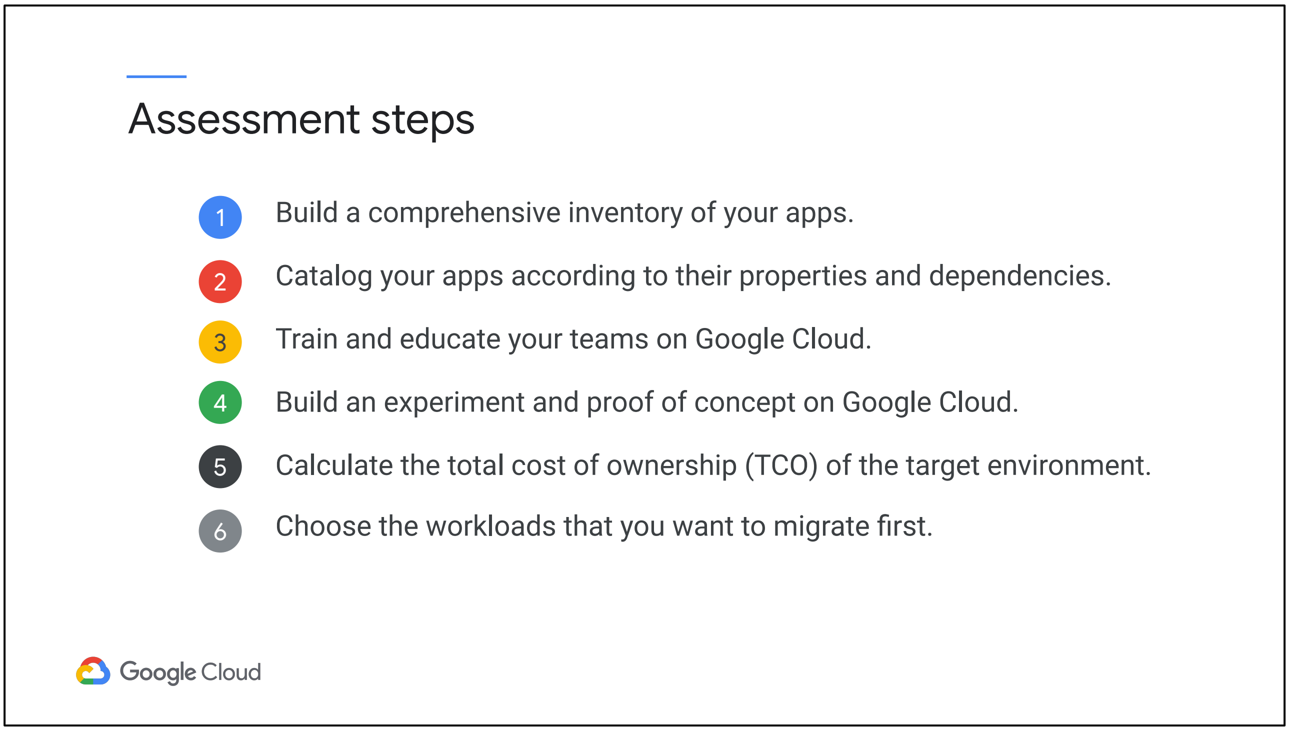


Google Cloud

The assessment phase is where you determine the requirements and dependencies to migrate your apps to Google Cloud.

The assessment phase is crucial for the success of your migration. You need to gain deep knowledge about the apps you want to migrate, their requirements, their dependencies, and your current environment. You need to understand your starting point to successfully plan and execute a Google Cloud migration.

In this phase, you perform the following steps:



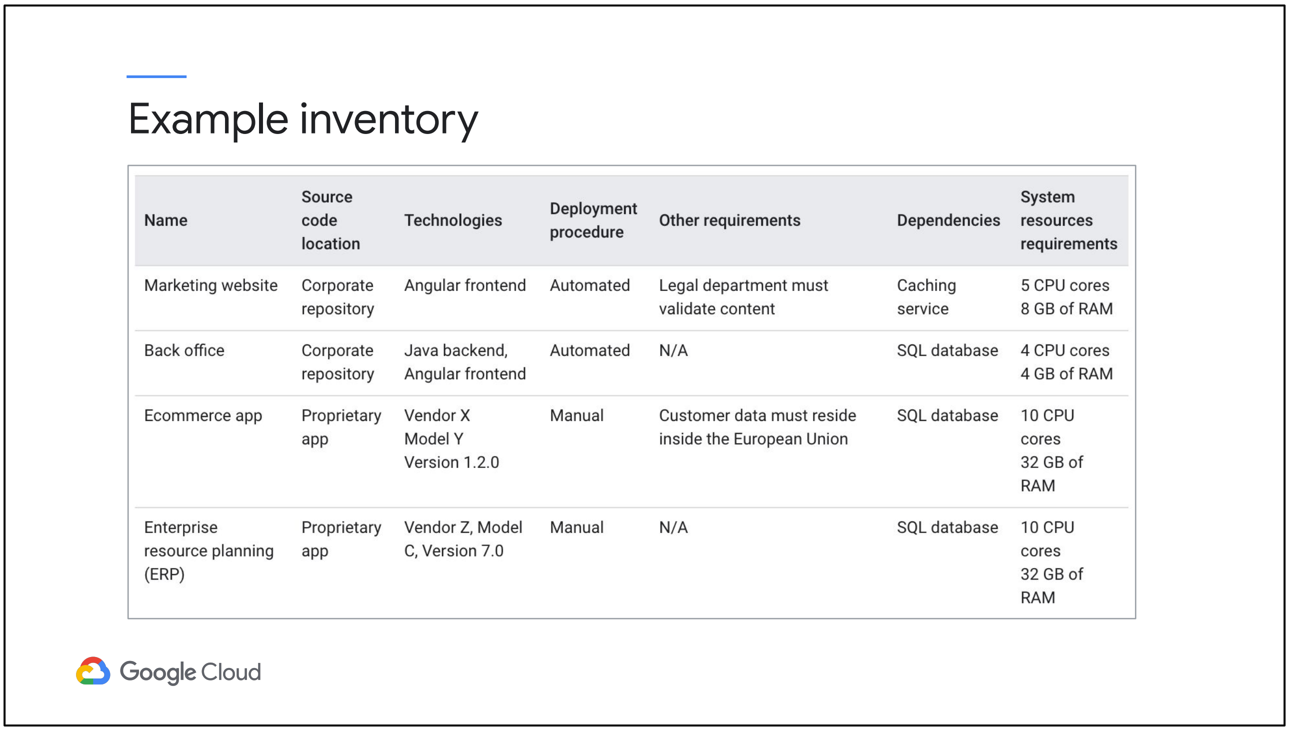
Assessment steps

1. Build a comprehensive inventory of your apps.
2. Catalog your apps according to their properties and dependencies.
3. Train and educate your teams on Google Cloud.
4. Build an experiment and proof of concept on Google Cloud.
5. Calculate the total cost of ownership (TCO) of the target environment.
6. Choose the workloads that you want to migrate first.

Google Cloud

Assessment steps include the following:

- First, Building a comprehensive inventory of your apps.
- Second, cataloging your apps according to their properties and dependencies.
- Third is training and educating your teams on Google Cloud.
- Fourth, building an experiment and proof of concept on Google Cloud.
- Fifth, calculating the total cost of ownership of the target environment.
- And finally, sixth choosing the workloads that you want to migrate first.



Example inventory

Name	Source code location	Technologies	Deployment procedure	Other requirements	Dependencies	System resources requirements
Marketing website	Corporate repository	Angular frontend	Automated	Legal department must validate content	Caching service	5 CPU cores 8 GB of RAM
Back office	Corporate repository	Java backend, Angular frontend	Automated	N/A	SQL database	4 CPU cores 4 GB of RAM
Ecommerce app	Proprietary app	Vendor X Model Y Version 1.2.0	Manual	Customer data must reside inside the European Union	SQL database	10 CPU cores 32 GB of RAM
Enterprise resource planning (ERP)	Proprietary app	Vendor Z, Model C, Version 7.0	Manual	N/A	SQL database	10 CPU cores 32 GB of RAM

Google Cloud

To scope your migration, you must first understand how many items, such as applications and hardware appliances, exist in your current environment, along with their dependencies.

Building the inventory is a non-trivial task that requires a significant effort, especially when you don't have any automatic cataloging system in place.

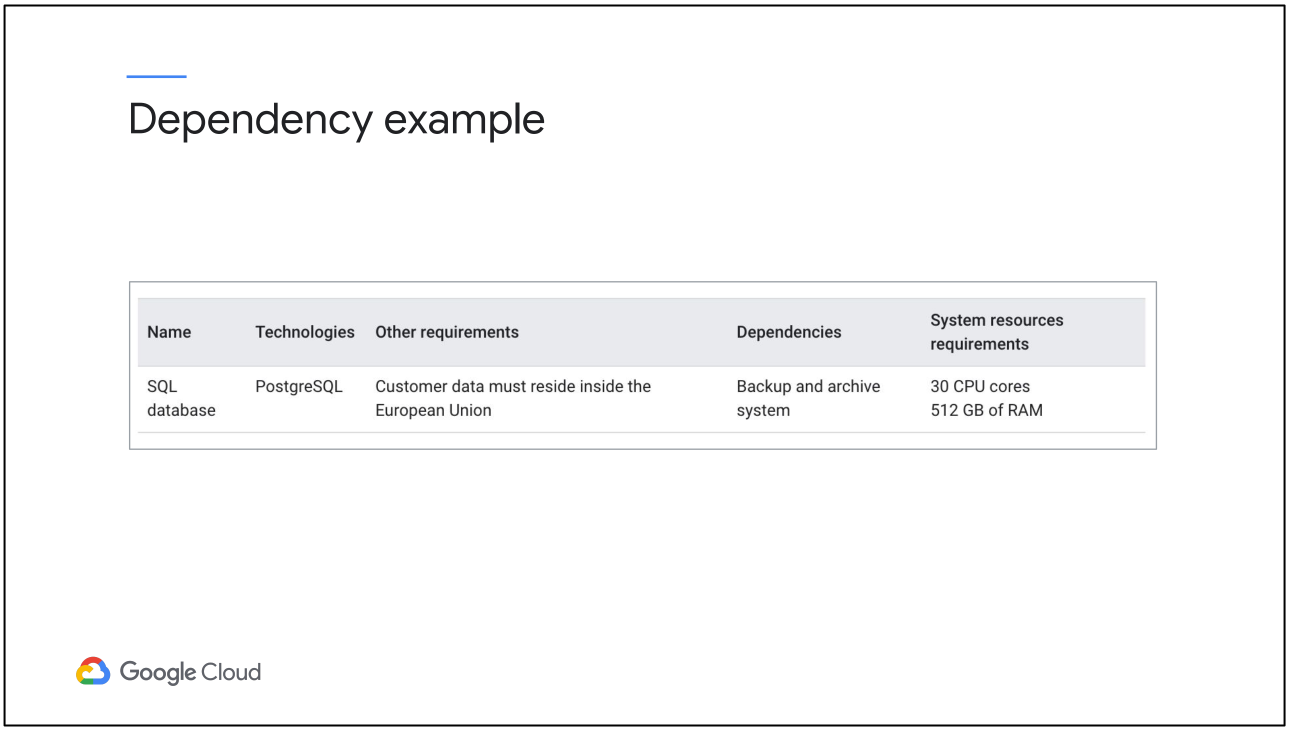
To have a comprehensive inventory, you need to use the expertise of the teams that are responsible for the design, deployment, and operation of each workload in your current environment, as well as the environment itself.

The inventory shouldn't be limited to only applications, but should at least contain the following:

- Dependencies of each app.
- Services supporting your app infrastructure.
- Servers, either virtual or physical.
- Physical appliances, such as network devices, firewalls, and other dedicated hardware.

Shown here is an example inventory.

In your projects, the fields may vary. But you get the idea. Show the things that are important for the success of your migration project.

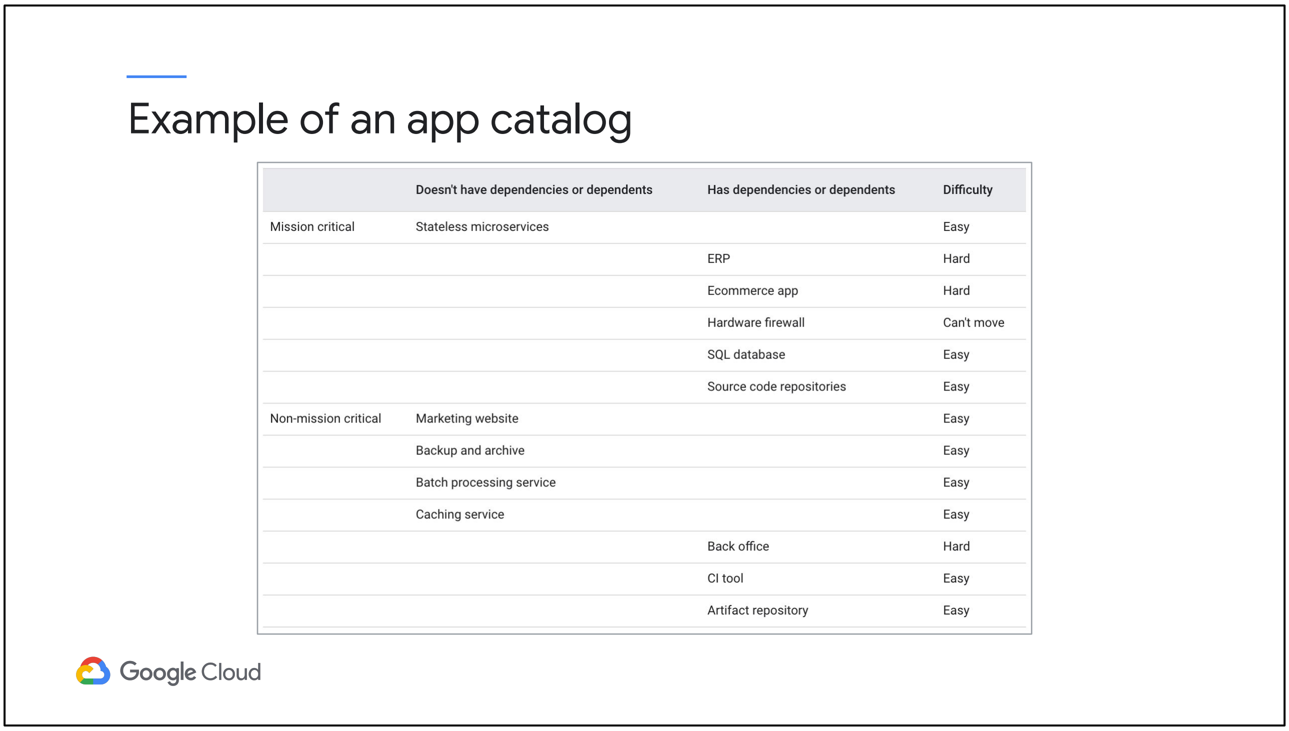


Dependency example

Name	Technologies	Other requirements	Dependencies	System resources requirements
SQL database	PostgreSQL	Customer data must reside inside the European Union	Backup and archive system	30 CPU cores 512 GB of RAM

Google Cloud

This table is an example of the dependencies of a database listed in the inventory. These dependencies are necessary for the database to correctly function. When you move the database, changes to the dependencies will also need to be made.



Example of an app catalog

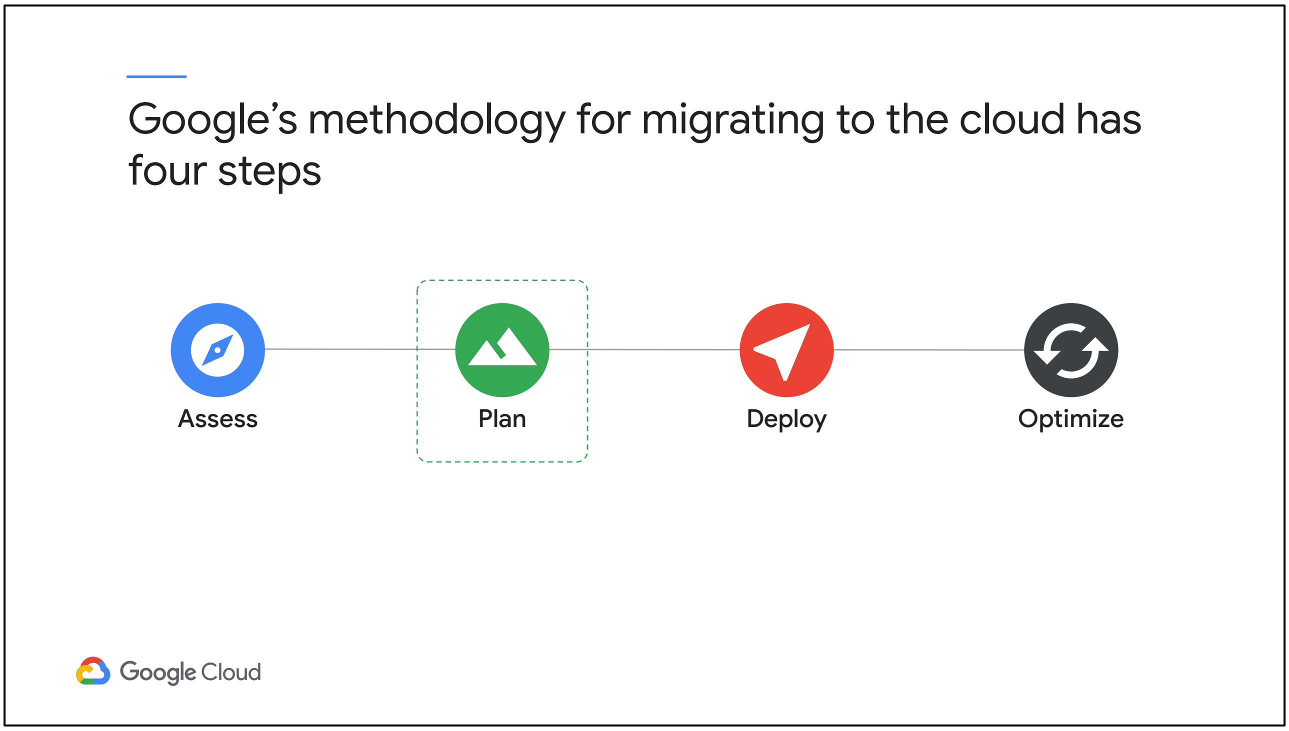
	Doesn't have dependencies or dependents	Has dependencies or dependents	Difficulty
Mission critical	Stateless microservices		Easy
		ERP	Hard
		Ecommerce app	Hard
		Hardware firewall	Can't move
		SQL database	Easy
		Source code repositories	Easy
Non-mission critical	Marketing website		Easy
	Backup and archive		Easy
	Batch processing service		Easy
	Caching service		Easy
		Back office	Hard
		CI tool	Easy
		Artifact repository	Easy

Google Cloud

This analysis is largely about identifying dependencies and dependents. It is important to catalog the applications that are dependent on the database. Care must be taken not to break those apps when the database is moved.

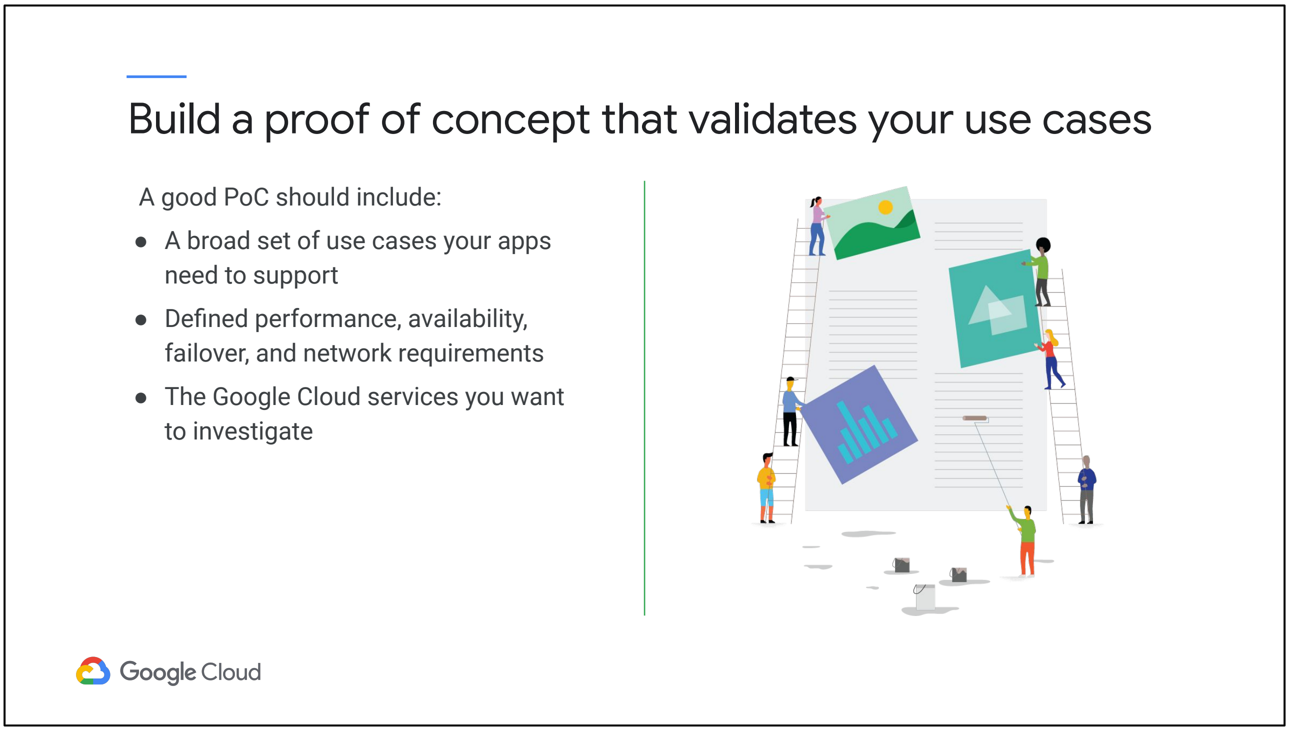
It is also important to identify the things that the database depends on, so the database continues to function correctly when it is moved.

Here is an example of an App Catalog.



Google's methodology for migrating to the cloud has four steps

Once you understand your current environment, you can start planning your migration.



Build a proof of concept that validates your use cases

A good PoC should include:

- A broad set of use cases your apps need to support
- Defined performance, availability, failover, and network requirements
- The Google Cloud services you want to investigate

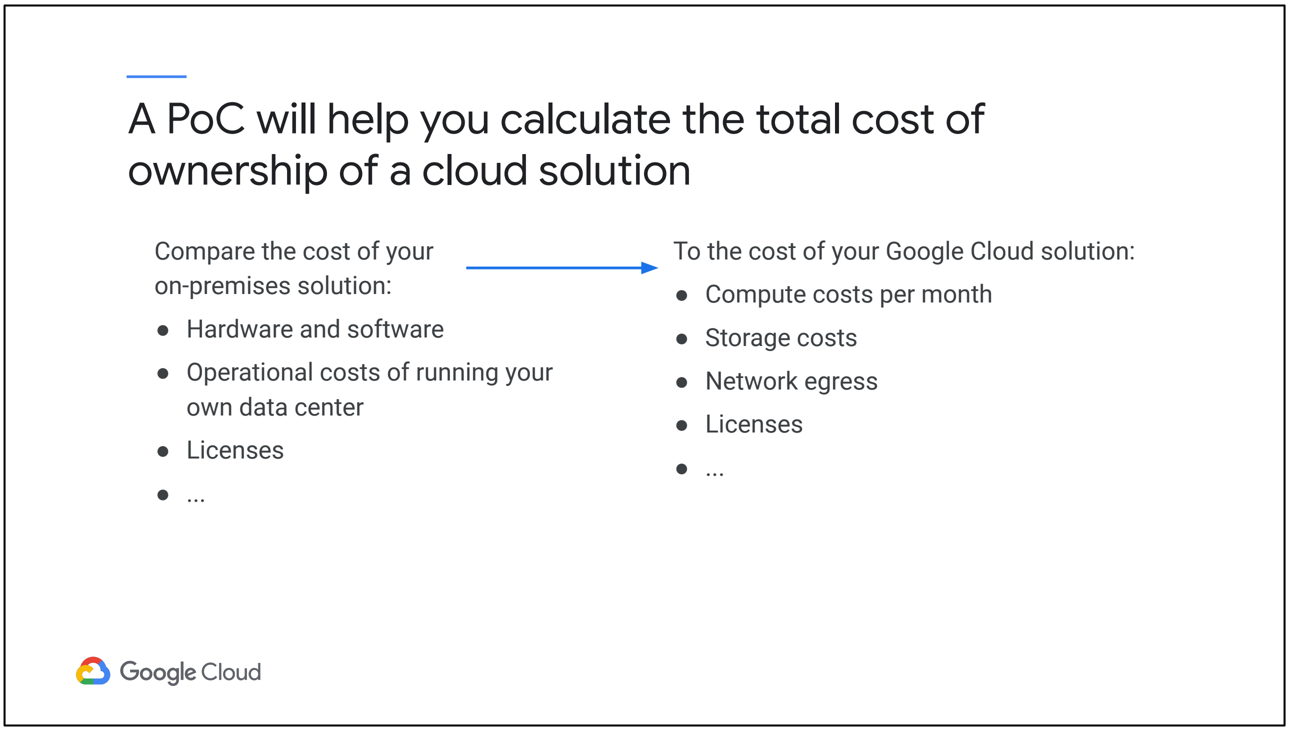
Google Cloud

To show the value and efficacy of Google Cloud, consider designing and developing one or more proofs of concept for each category of app in your app catalog. Experimentation and testing let you validate assumptions and demonstrate the value of cloud to business leaders.

At a minimum, your PoC should include the following:

- A comprehensive list of the use cases that your apps support, including uncommon ones and corner cases.
- All the requirements for each use case, such as performance and scalability requirements, expected consistency guarantees, failover mechanisms, and network requirements.
- A potential list of technologies and products that you want to investigate and test.

You should design PoCs and experiments to validate all the use cases on the list. Each experiment should have a precise validity context, scope, expected outputs, and measurable business impact.



A PoC will help you calculate the total cost of ownership of a cloud solution

Compare the cost of your on-premises solution:

- Hardware and software
- Operational costs of running your own data center
- Licenses
- ...

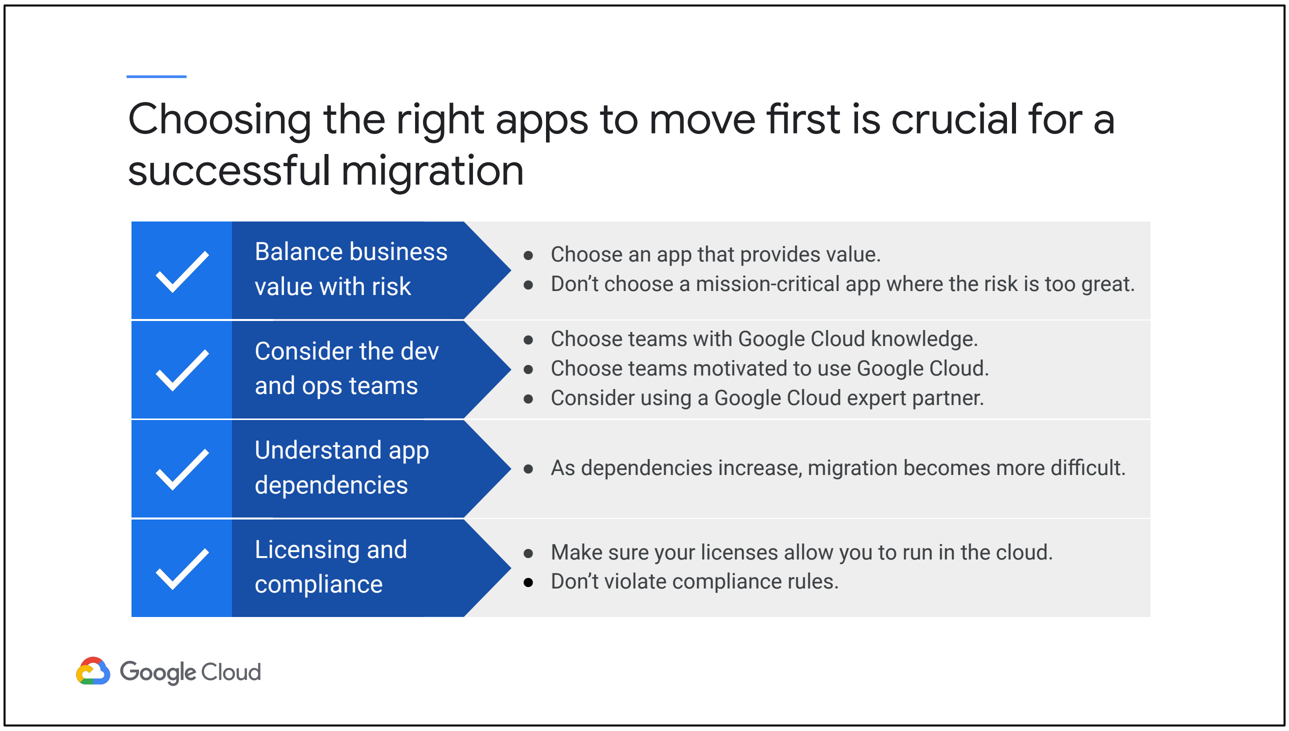
To the cost of your Google Cloud solution:

- Compute costs per month
- Storage costs
- Network egress
- Licenses
- ...

Google Cloud

A PoC will help you calculate the total cost of ownership of a cloud solution. When you have a clear view of the resources you need in the new environment, you can build a total cost of ownership model that lets you compare your costs on Google Cloud with the costs of your current environment.

When building this cost model, you should consider both the costs for hardware and software and also all the operational costs of running your own data center, such as power, cooling, maintenance, and other support services. Consider that it's also typically easier to reduce costs, thanks to the elastic scalability of Google Cloud resources, compared to a more rigid on-premises data center.



Choosing the right apps to move first is crucial for a successful migration

✓	Balance business value with risk	• Choose an app that provides value. • Don't choose a mission-critical app where the risk is too great.
✓	Consider the dev and ops teams	• Choose teams with Google Cloud knowledge. • Choose teams motivated to use Google Cloud. • Consider using a Google Cloud expert partner.
✓	Understand app dependencies	• As dependencies increase, migration becomes more difficult.
✓	Licensing and compliance	• Make sure your licenses allow you to run in the cloud. • Don't violate compliance rules.

Google Cloud

Now that you have an exhaustive view of your current environment, you need to complete your migration plan by choosing the initial order in which you want to migrate your apps. You can refine this order during the migration when you gain experience with Google Cloud and understand your environment and apps.

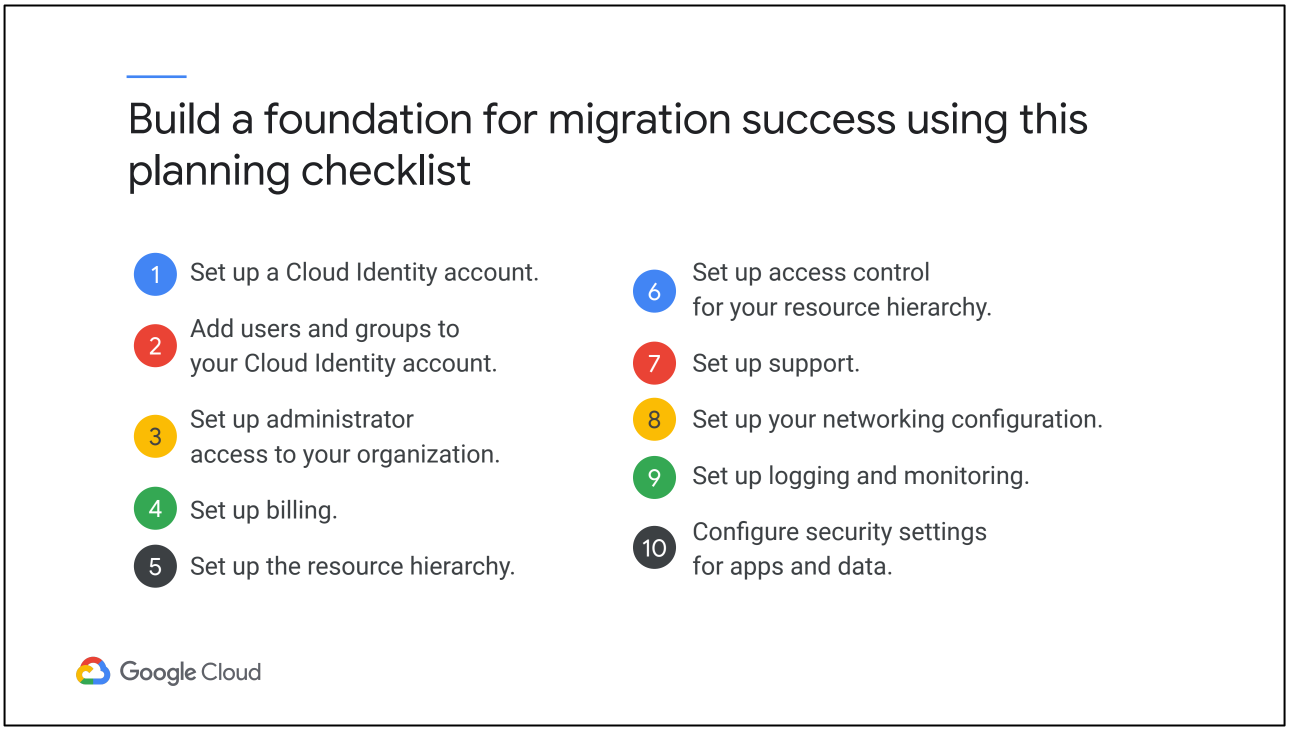
The apps you migrate first are the ones that let your teams build their knowledge and experience on Google Cloud. Greater cloud exposure and experience from your team can lower the risk of complications during the deployment phase of your migration and make subsequent migrations easier and quicker. For this reason, choosing the right first-movers is crucial for a successful migration. You can pick one app or put many apps from across your apps matrix in your first-mover list.
The task of identifying the first-movers is complex, but here are some criteria to guide you:

- Consider the business value of the app. Choose an app that provides value, but don't choose a mission-critical app where the risk is too great.

- Consider the teams responsible for development, deployment, and operations of the app. Choose teams with Google Cloud knowledge, and choose teams motivated to use Google Cloud. You may also want to consider using a Google Cloud expert partner to help.

- Make sure you understand the number, type, and scope of dependencies of the app. More dependencies can create a more difficult migration.

- Be careful to understand the compliance and licensing requirements of the app. Make sure your licenses allow you to run in the cloud, and don’t violate any compliance rules you are subject to.



Build a foundation for migration success using this planning checklist

1. Set up a Cloud Identity account.
2. Add users and groups to your Cloud Identity account.
3. Set up administrator access to your organization.
4. Set up billing.
5. Set up the resource hierarchy.
6. Set up access control for your resource hierarchy.
7. Set up support.
8. Set up your networking configuration.
9. Set up logging and monitoring.
10. Configure security settings for apps and data.

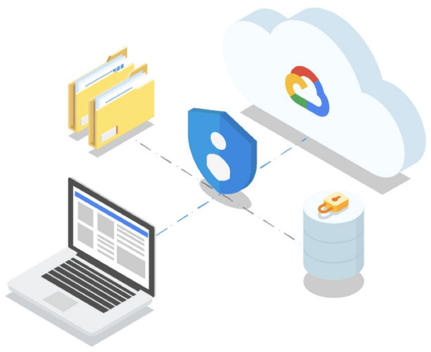
Google Cloud

When planning your migration to Google Cloud, you need to understand an array of topics and concepts related to cloud architecture. A poorly planned foundation can cause you to face delays, confusion, and downtime and can put the success of your cloud migration at risk.

Shown here is a checklist of organizational functions that need to be in place before you can start moving your applications and databases. If you put this foundation in place before your migration project, all you need to worry about is the migration. If you don't, you'll be tasked to add these foundational items as you go, making the migration harder and causing delays along the way.

Cloud Identity allows you to centrally manage users and groups who can access cloud resources

- Manage users and groups from the Google Admin website.
- You can synchronize users and groups with Active Directory or LDAP servers.
- You can also set up SSO with third-party identity providers.



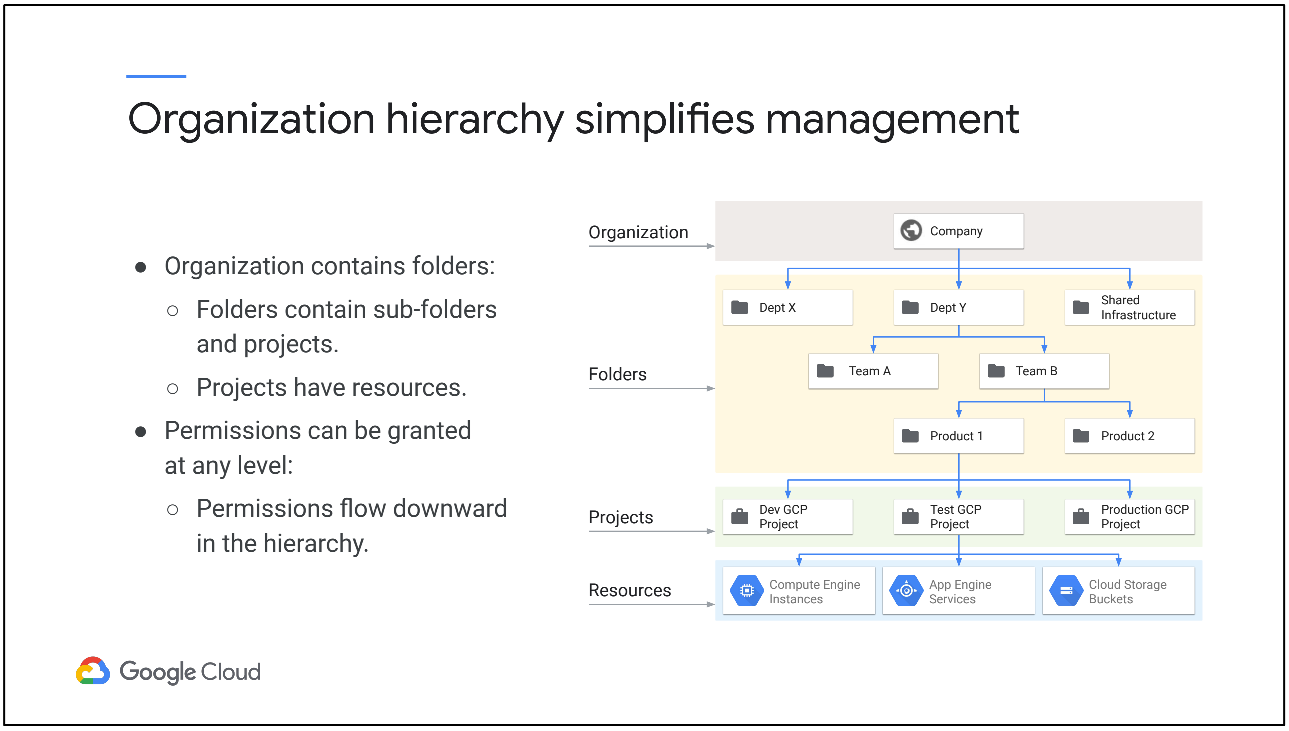
Google Cloud

Cloud Identity allows you to centrally manage users and groups who can access cloud resources.

You manage users and groups from the Google Admin website, and you can synchronize users and groups from your on-premises Active Directory or LDAP servers.

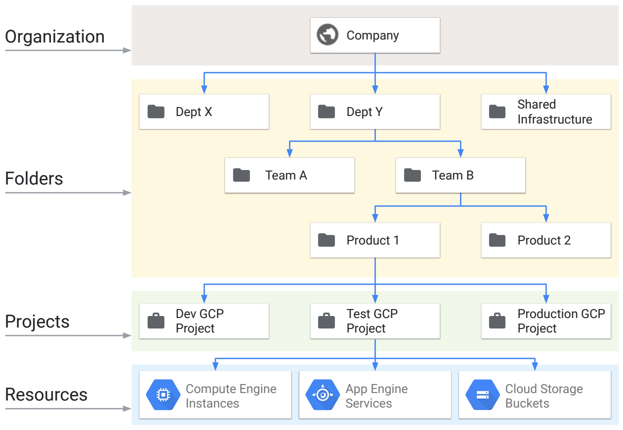
You can delegate the responsibility for identifying users to Google. Or you can easily set up a third-party single sign-on identity provider. The third party can be another cloud service like Microsoft Active Directory online, or it can be a service managed by you in your data center.

In any case, to use Google Cloud, your organization has to be set up under Cloud Identity. After that, you will automatically have a Google Cloud Organization set up.



Organization hierarchy simplifies management

- Organization contains folders:
    - Folders contain sub-folders and projects.
    - Projects have resources.
- Permissions can be granted at any level:
    - Permissions flow downward in the hierarchy.



Google Cloud

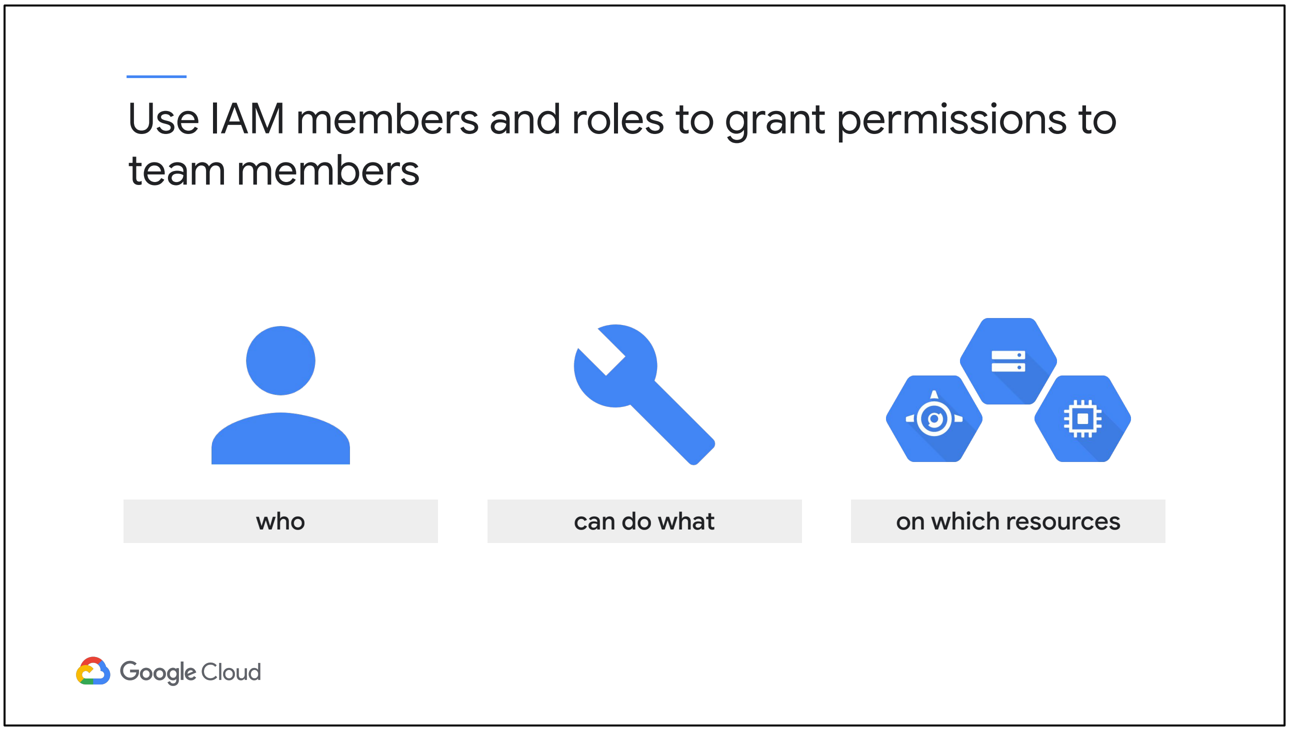
The Google Cloud Organization hierarchy can greatly simplify the management of resources, users, and groups.

At the top of the hierarchy is the Organization. Remember, this is created automatically when you set up your Cloud Identity account.

An Organization contains zero or more folders. Folders can have subfolders, which can have subfolders, and so on. Projects can be created within any folder or subfolder or directly below the Organization. Resources are created within projects.

Permissions can be granted at any level of the hierarchy, and those permissions flow downward. For example, you provide access to a Cloud Storage bucket, at the bucket level, the project level, the folder level, or the organization level. When a right is granted, it cannot be removed at a lower level.

You also need to identify the people who will manage the organizational hierarchy. There are roles like Organization Admin, Folder Creator, and Project Creator that have to be assigned to the right people who will manage this on behalf of the development teams.



Use IAM members and roles to grant permissions to team members

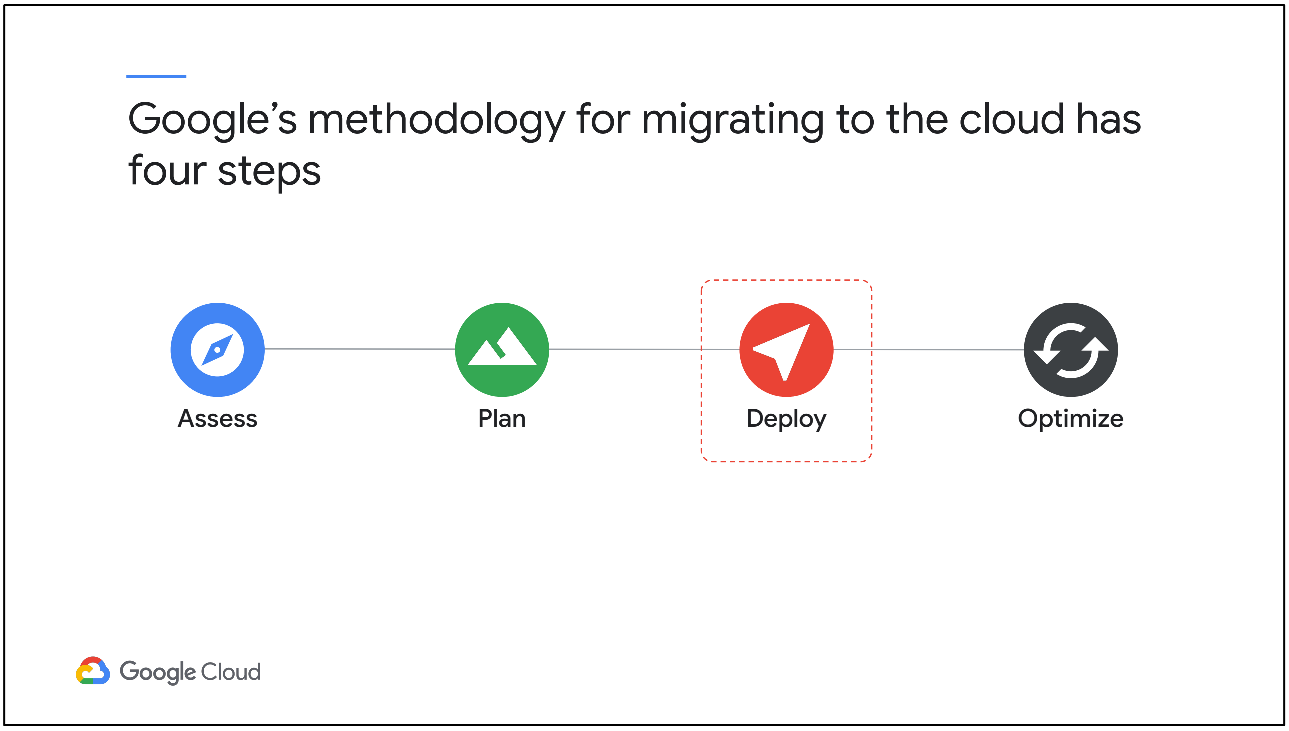
who	can do what	on which resources

Google Cloud

Identity and Access Management is the service that is used to grant permission to team members. IAM controls who can do what to which resources.

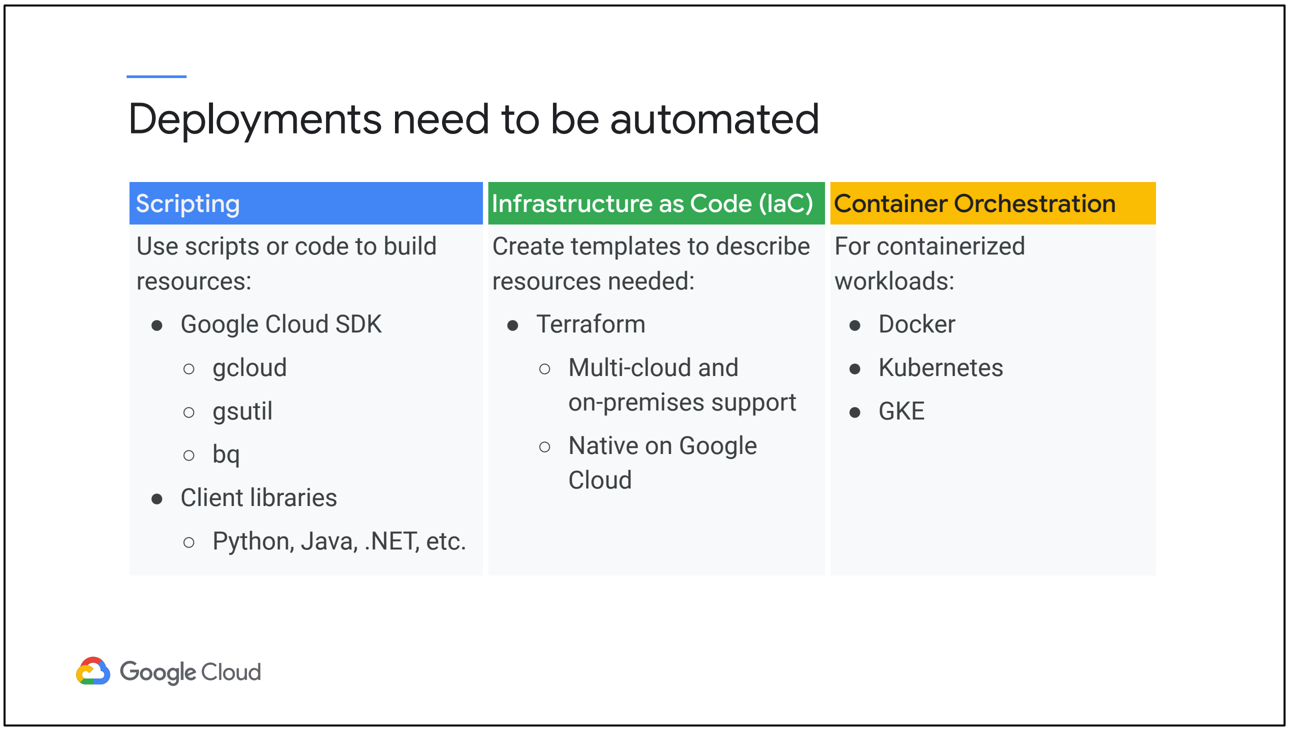
Your users are added as Cloud Identity accounts and authenticate to Google Cloud with their login. Add members to groups to simplify management of permissions. Groups are managed by Google or by using your third-party SSO provider.

Users or groups are assigned one or more roles. A role is simply a list of API functions. If the function is in the list, it can be called by members of that group. Again, the assignment of roles to members can be done at the Organization, Folder, Project, or Resource level, whichever is most appropriate.



Google’s methodology for migrating to the cloud has four steps

Once you have a plan, it's time to start deploying resources to the cloud.



Deployments need to be automated

Scripting	Infrastructure as Code (IaC)	Container Orchestration
Use scripts or code to build resources: ● Google Cloud SDK ○ gcloud ○ gsutil ○ bq ● Client libraries ○ Python, Java, .NET, etc.	Create templates to describe resources needed: ● Terraform ○ Multi-cloud and on-premises support ○ Native on Google Cloud	For containerized workloads: ● Docker ● Kubernetes ● GKE

Google Cloud

A key aspect of making your deployments successful is automation. If you were only going to create a Google Cloud resource once, the fastest and easiest way would be to manually create it with the web console. In real projects, things aren't that simple though. You make mistakes along the way, change your mind as to how you are architecting your solution, new versions of things come along and you want to move your resources from one project to another project. If you're creating your resources manually, your projects will take much longer, maintenance will be harder, change will be more difficult, and your projects will be more prone to human error.

Almost every Google Cloud resource can be created manually using the web console. The console is a great way to see how things work and to do experiments. However, every resource can also be created using code, and there are different ways to write that code.

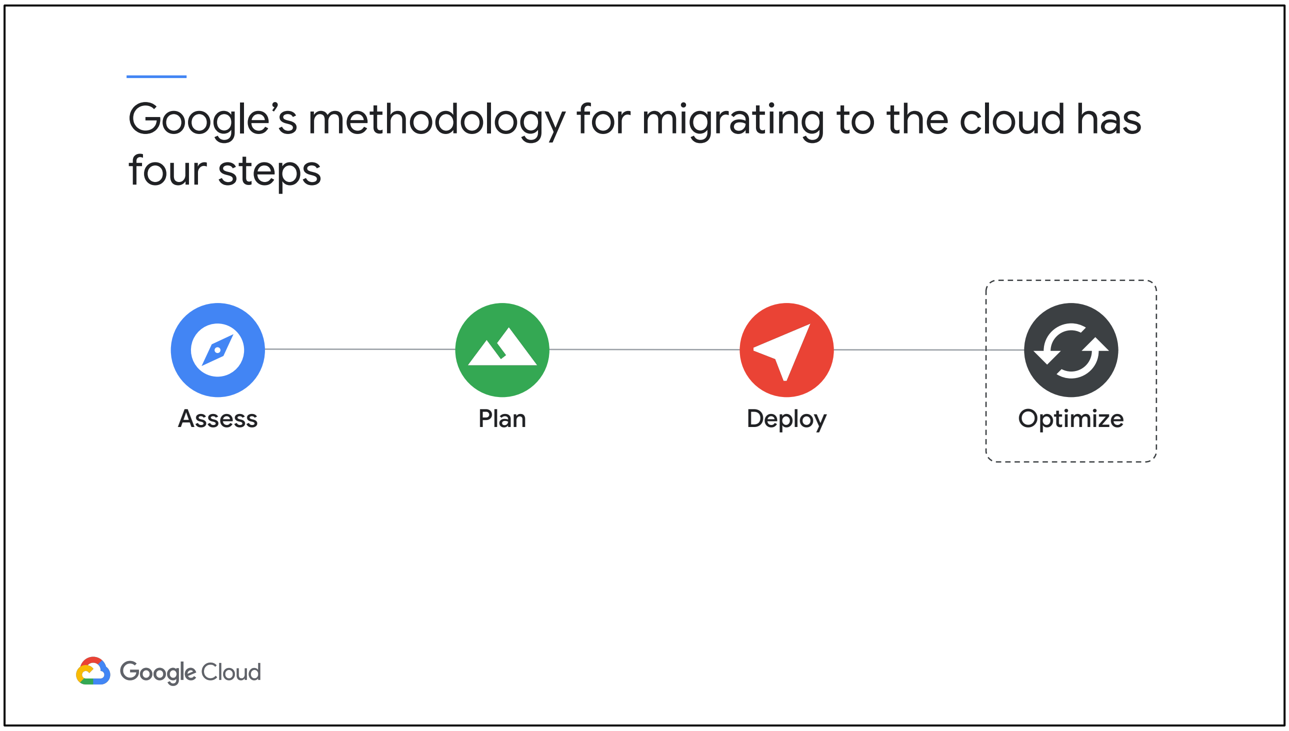
- One way to automate resource creation is through scripting. You saw earlier in the course how you could use the Google Cloud SDK and some simple CLI commands to create virtual machines and Cloud SQL databases. All resources can be created with similar commands. You could easily string together multiple commands inside a code file and run them as a Shell script. You could also download a Google Cloud client library for your favorite programming language and write a program using it to create resources. Many languages are supported, including Java, Python, .NET, and Node.js.

- You can also use infrastructure-as-code tools to automate resource creation.

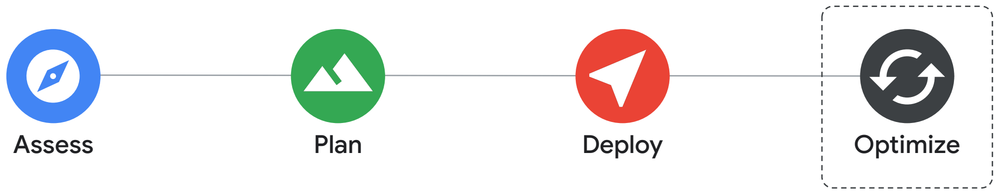
- With IaC tools, you define templates which describe what resources you want to create in a project, then with a simple command, you can create, delete, or update your resources based on the templates. This is similar to programming a Shell script, but uses a more declarative approach to describe what you want. The advantage of Terraform is that it works with any cloud provider and also works in private clouds. Often, companies already have experience using Terraform and want to leverage that prior knowledge when migrating to Google Cloud. Terraform is considered a first-class citizen on Google Cloud. It is even installed by default in Google Cloud Shell.

- You also saw earlier in the course how Kubernetes works. Kubernetes is a great way to automate the deployment of your applications, databases, and resources. In a way, it is similar to Terraform. You create a template that describes your applications and use simple commands to deploy them. There is even a Terraform provider that can be used in place of the Kubernetes templates and kubectl CLI.

Whichever tool you use (scripting, IaC, or Kubernetes), the important thing is you're not doing things manually; instead, you are automating as much as possible. In many cases, you will end up with a combination of these tools.

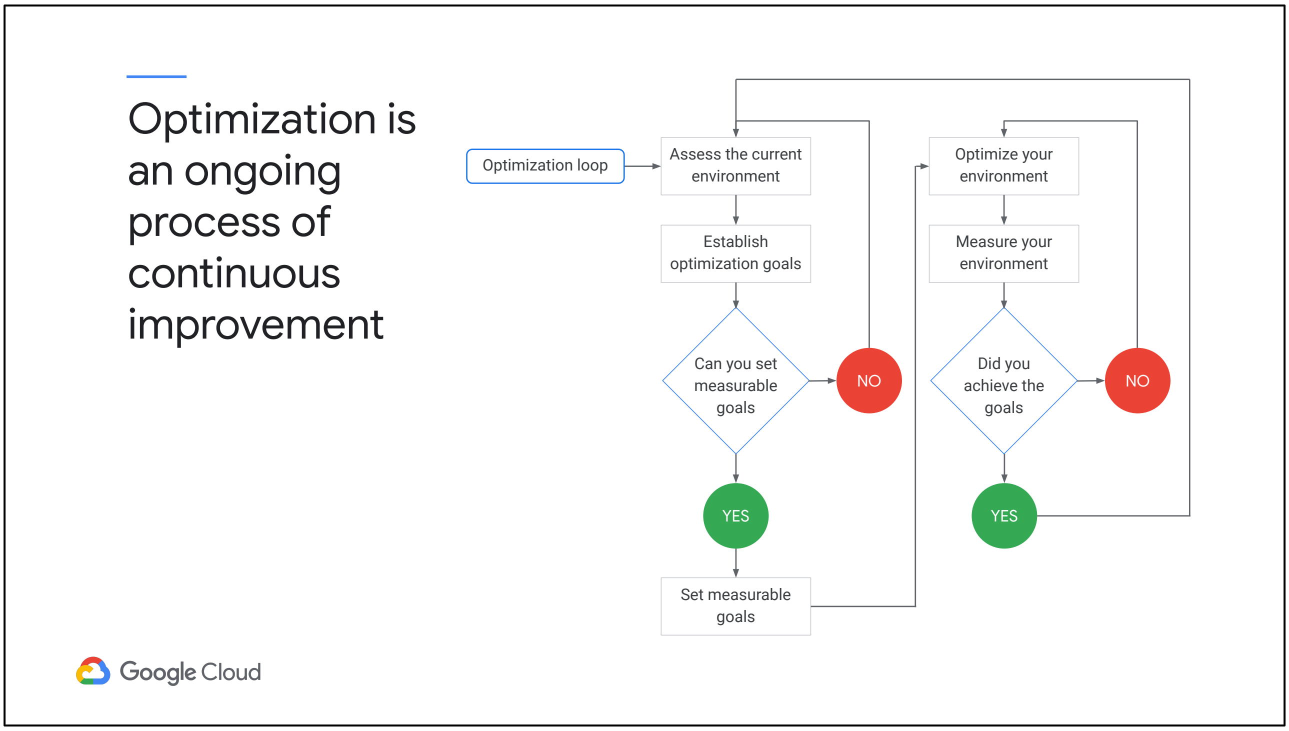


Google's methodology for migrating to the cloud has four steps

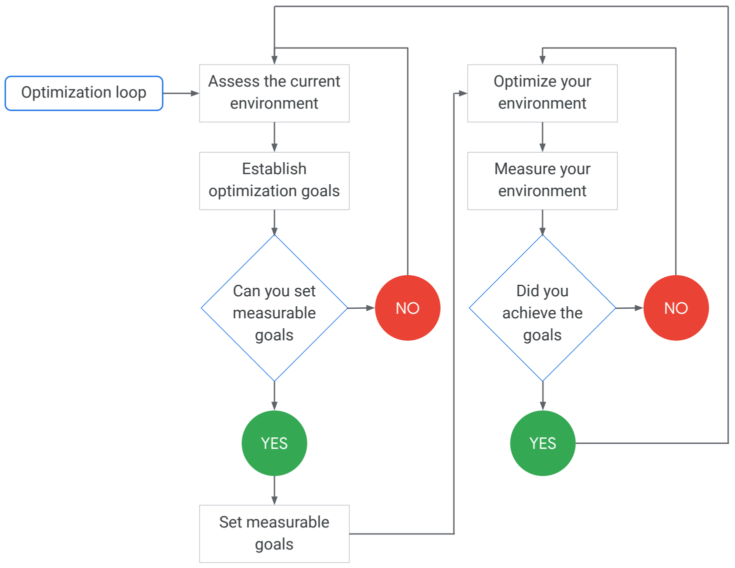


Google Cloud

After you've done your initial deployment, the optimization phase begins. Optimization is an ongoing process of continuous improvement.



Optimization is an ongoing process of continuous improvement

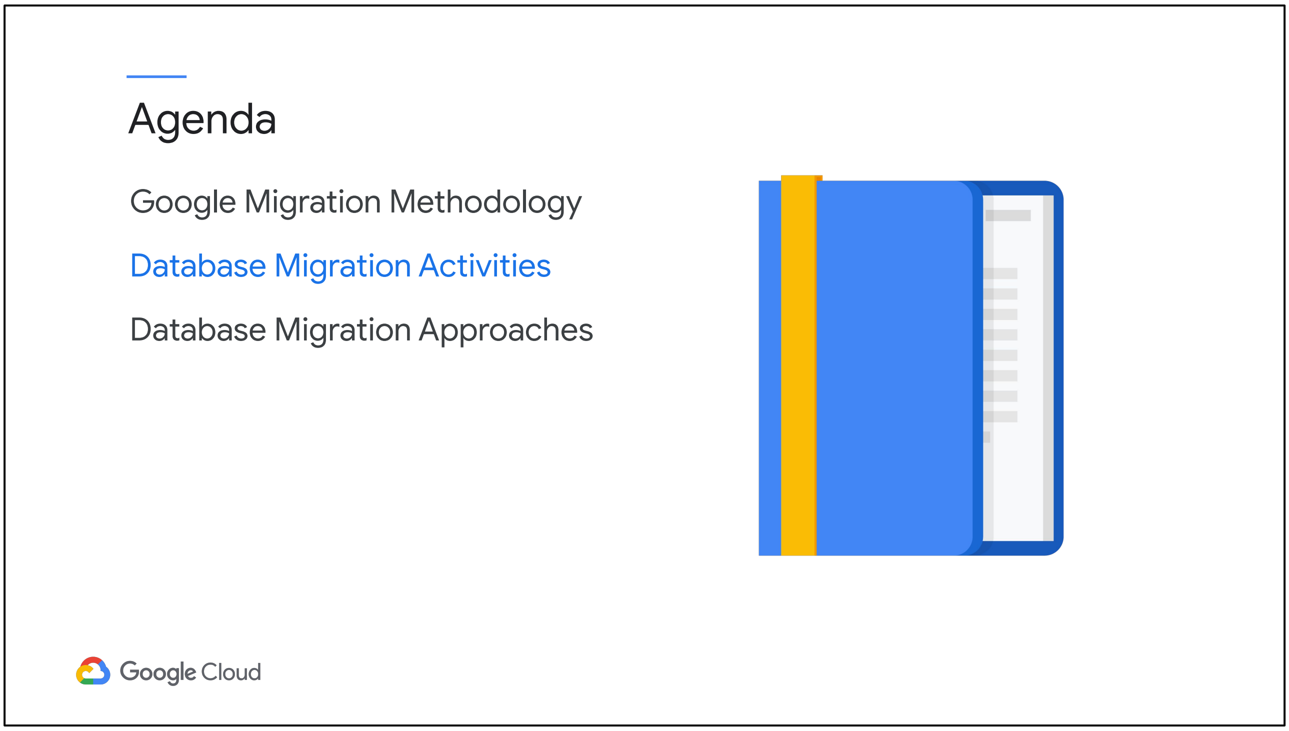

Optimization loop
Assess the current environment
Establish optimization goals
Can you set measurable goals
NO
YES
Set measurable goals
Optimize your environment
Measure your environment
Did you achieve the goals
NO
YES

Google Cloud

You optimize your environment as it evolves. To avoid uncontrolled and duplicative efforts, you can set measurable optimization goals and stop when you meet these goals. After that, you can always set new and more ambitious goals, but consider that optimization has a cost in terms of resources, time, effort, and skills.

This diagram shows the optimization loop.

Asses the current environment.
Set your optimization goals.
Optimize your environment.
If you meet your goals, reassess the environment. If not, try again.

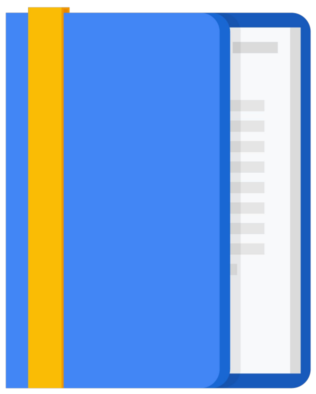


Agenda

Google Migration Methodology

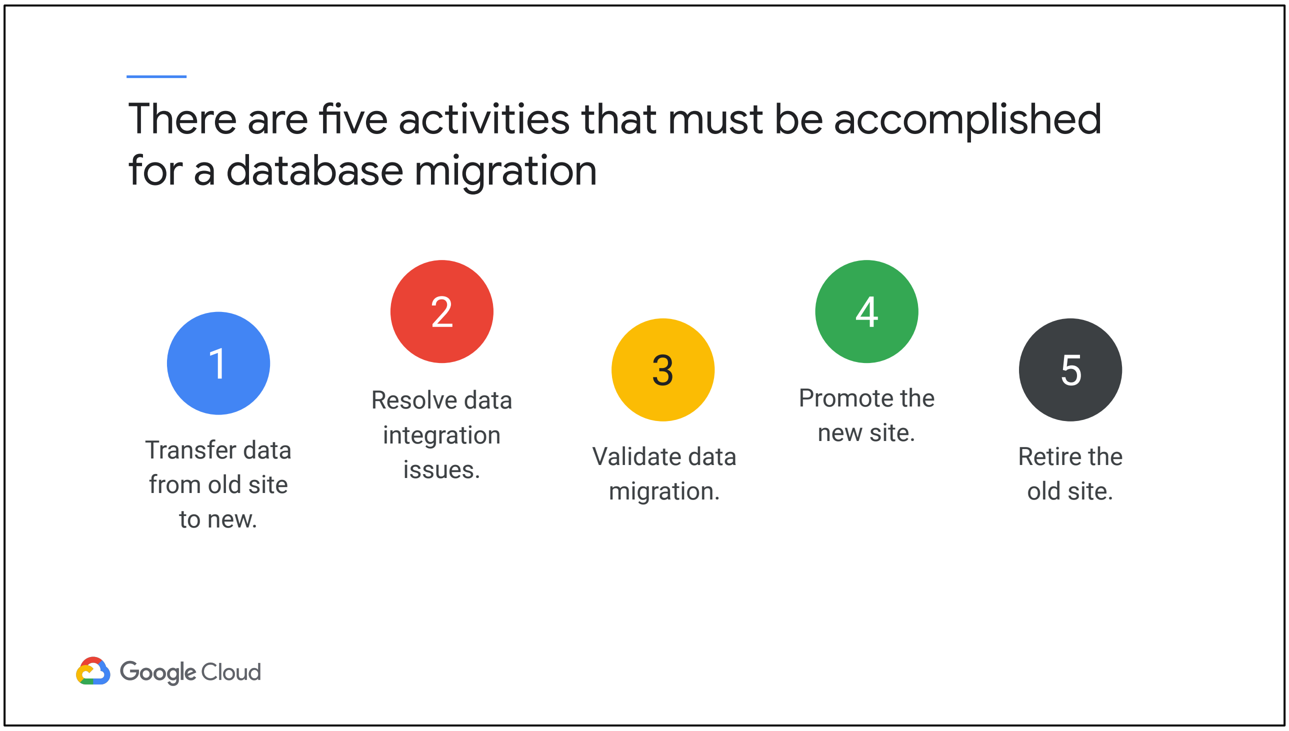
Database Migration Activities

Database Migration Approaches



Google Cloud

Several activities have to be performed to migrate a database to Google Cloud. Let's talk about those now.



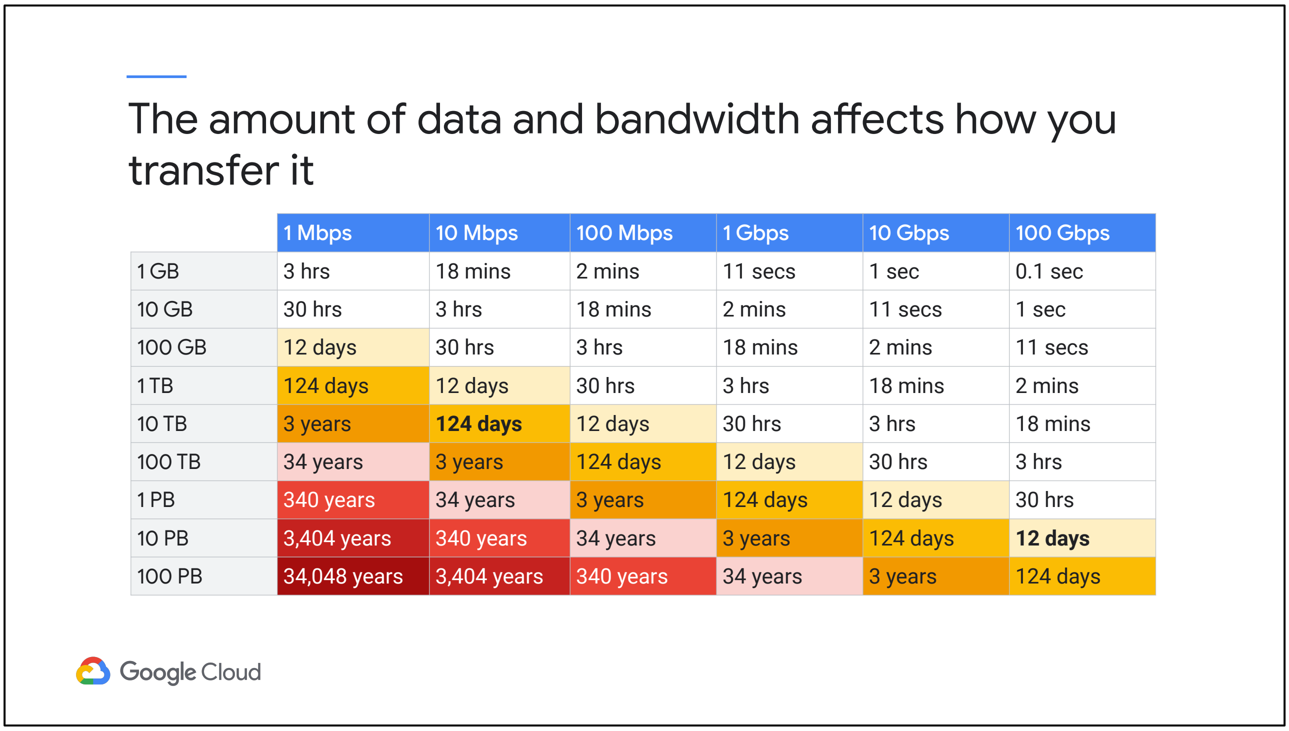
There are five activities that must be accomplished for a database migration

1. Transfer data from old site to new.
2. Resolve data integration issues.
3. Validate data migration.
4. Promote the new site.
5. Retire the old site.

Google Cloud

There are five steps required to migrate a database:

- First, transfer the data from the old site to Google Cloud.
- Second, resolve any data integration issues you might have.
- Third, validate the migration.
- Fourth, promote the new database.
- And fifth, retire the old database.



The amount of data and bandwidth affects how you transfer it

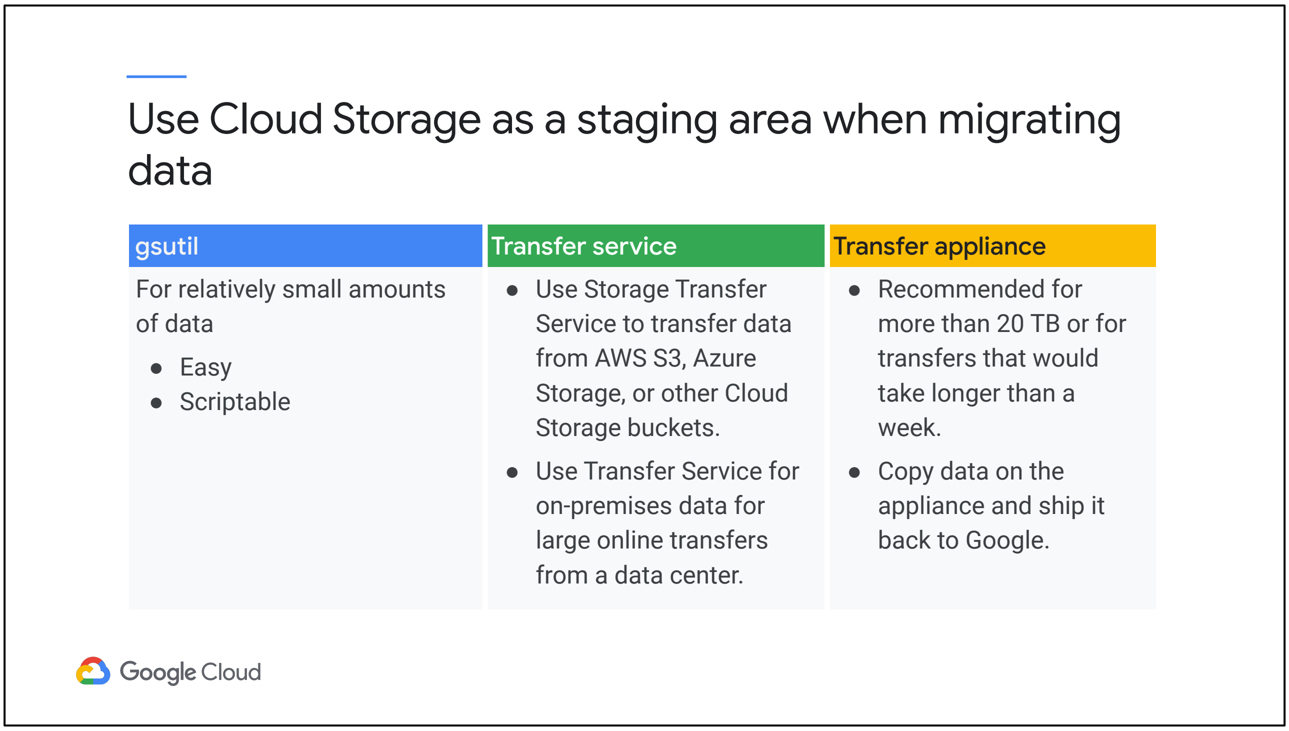
	1 Mbps	10 Mbps	100 Mbps	1 Gbps	10 Gbps	100 Gbps
1 GB	3 hrs	18 mins	2 mins	11 secs	1 sec	0.1 sec
10 GB	30 hrs	3 hrs	18 mins	2 mins	11 secs	1 sec
100 GB	12 days	30 hrs	3 hrs	18 mins	2 mins	11 secs
1 TB	124 days	12 days	30 hrs	3 hrs	18 mins	2 mins
10 TB	3 years	**124 days**	12 days	30 hrs	3 hrs	18 mins
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

Transferring the data can be easy or hard, depending on the amount of data you have. If you only have a few GBs, you don't have a problem. But if you have TBs of data, you had better have a fast network connection to get that initial data transferred quickly.

The table here shows the theoretical data transfer time required for given amounts of data based on your bandwidth. If you have many terabytes of data, you can consider setting up a high-speed Cloud Interconnect connection between your data center and Google. Interconnect connections are sold in 10-Gb increments, and you can have a maximum of 200 Gb/sec bandwidth.

Even with a connection that fast, if you have petabytes of data, it's going to take too long. If your data transfer job will take too long over the internet, you can order a transfer appliance.



Use Cloud Storage as a staging area when migrating data

gsutil	Transfer service	Transfer appliance
For relatively small amounts of data • Easy • Scriptable	• Use Storage Transfer Service to transfer data from AWS S3, Azure Storage, or other Cloud Storage buckets. • Use Transfer Service for on-premises data for large online transfers from a data center.	• Recommended for more than 20 TB or for transfers that would take longer than a week. • Copy data on the appliance and ship it back to Google.

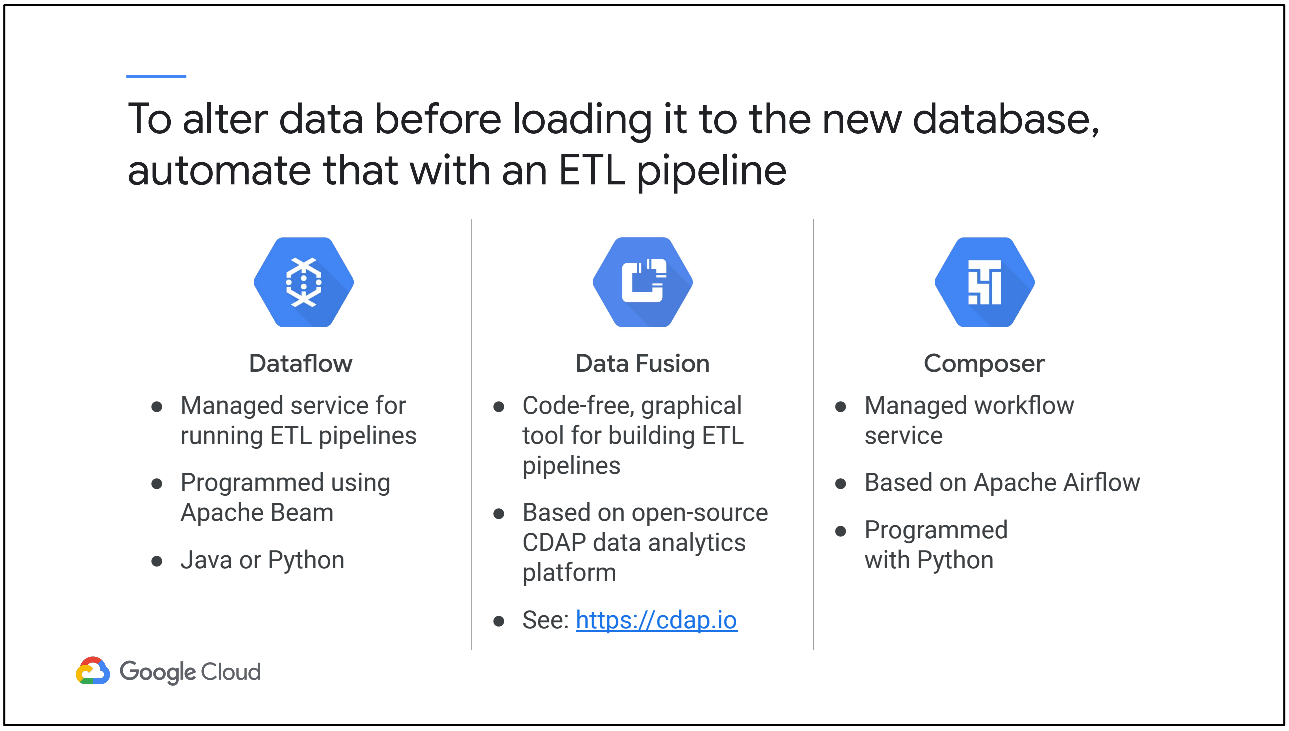
Google Cloud

When migrating large amounts of data to Google Cloud, consider using Cloud Storage as a staging area. After the data is in storage, you can now move it around at Google speed. Cloud Storage is also relatively inexpensive compared to other storage locations.

- The simplest way to copy data into storage is using the **gsutil** command line interface. You can make this part of your automation scripts. If you are familiar with basic Linux file system commands, you will easily pick up gsutil commands.

- There are also a few more automated data **transfer services**. Storage Transfer Service can be used to transfer data over the internet from an AWS S3 bucket, Azure Storage, another Cloud Storage bucket, or from an on-premises file server. Transfer jobs are simple to create and can be run on a recurring schedule if needed.

- A **Transfer Appliance** is a physical storage device that you copy your data to. Google sends you the appliance, and you mount it on your network. You copy the data, encrypting it with an encryption key you generate. Send it back to Google, and then you can copy the data into a Cloud Storage bucket and use your key to decrypt it. You keep the encryption key, so if your appliance is lost or stolen in transit, no one will have access to the unencrypted data.



To alter data before loading it to the new database, automate that with an ETL pipeline

Dataflow

- Managed service for running ETL pipelines
- Programmed using Apache Beam
- Java or Python

Data Fusion

- Code-free, graphical tool for building ETL pipelines
- Based on open-source CDAP data analytics platform
- See: https://cdap.io

Composer

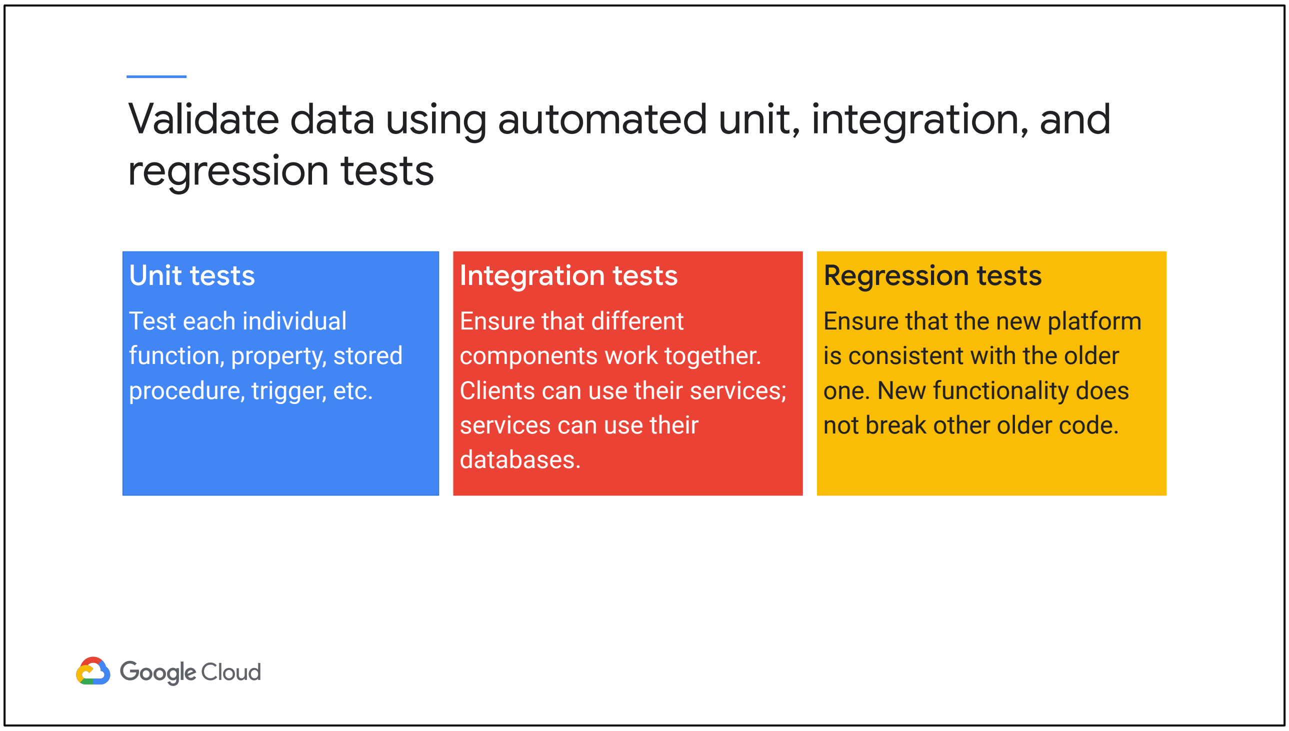
- Managed workflow service
- Based on Apache Airflow
- Programmed with Python

Google Cloud

Often, customers want to alter their database structure or move to a different database when migrating to the Cloud.

If data needs to be altered prior to being loaded to the new database, you can automate that with an ETL pipeline. Google Cloud provides multiple tools for doing that. These are Dataflow, Data Fusion, and Cloud Composer. There are also third-party tools, like Striim, or vendor-specific tools from Microsoft or Oracle.

Later in the course, these tools are discussed in more detail.



Validate data using automated unit, integration, and regression tests

Unit tests
Test each individual function, property, stored procedure, trigger, etc.

Integration tests
Ensure that different components work together. Clients can use their services; services can use their databases.

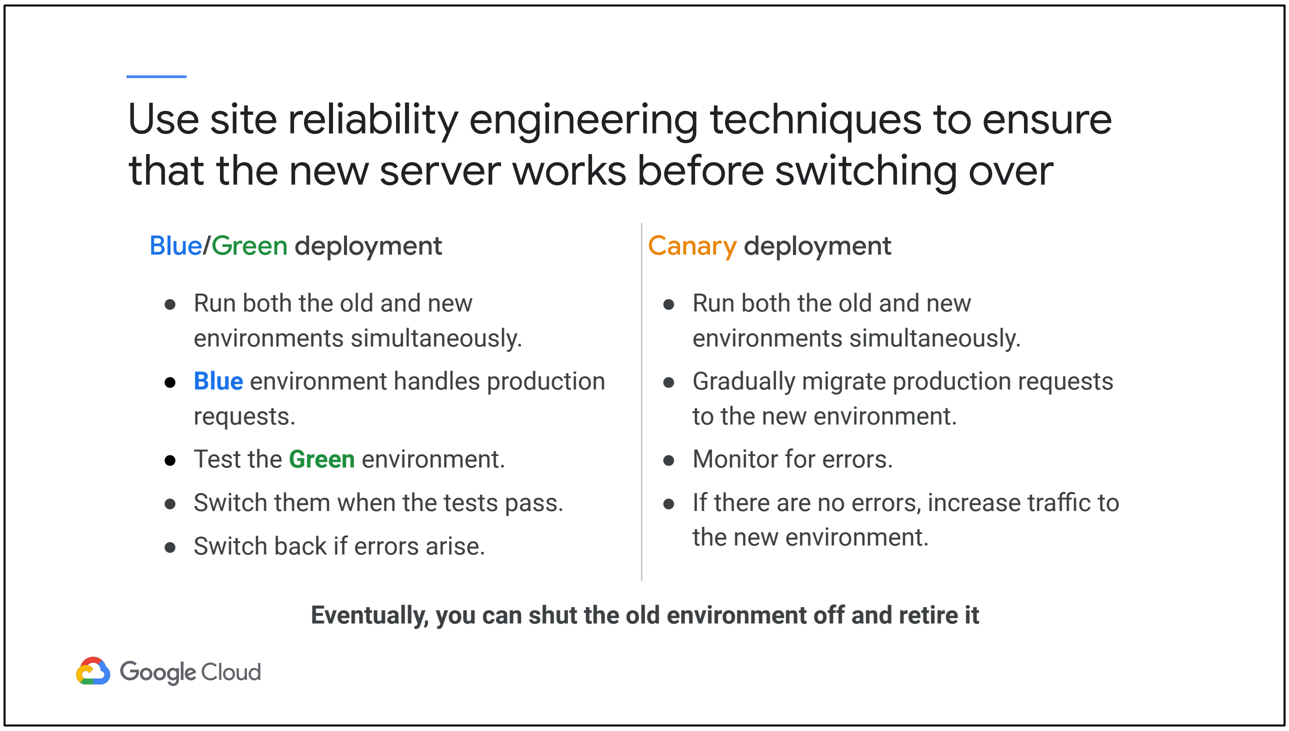
Regression tests
Ensure that the new platform is consistent with the older one. New functionality does not break other older code.

Google Cloud

The third activity is to validate the data migration. This can be done with thorough automated testing.

- **Unit tests** are used to verify each individual function, property, stored procedure, trigger, etc.
- **Integration tests** ensure that different components work together. Clients can use services; services can use their databases.
- **Regression tests** ensure that the new platform is consistent with the older one. New functionality does not break other older code.

For lower-level tests, use an automated test framework that works with your favorite programming language. There are many automated, third-party testing tools that you can use for system, security, and black box testing.



Use site reliability engineering techniques to ensure that the new server works before switching over

Blue/Green deployment

- Run both the old and new environments simultaneously.
- **Blue** environment handles production requests.
- Test the **Green** environment.
- Switch them when the tests pass.
- Switch back if errors arise.

Canary deployment

- Run both the old and new environments simultaneously.
- Gradually migrate production requests to the new environment.
- Monitor for errors.
- If there are no errors, increase traffic to the new environment.

Eventually, you can shut the old environment off and retire it

Google Cloud

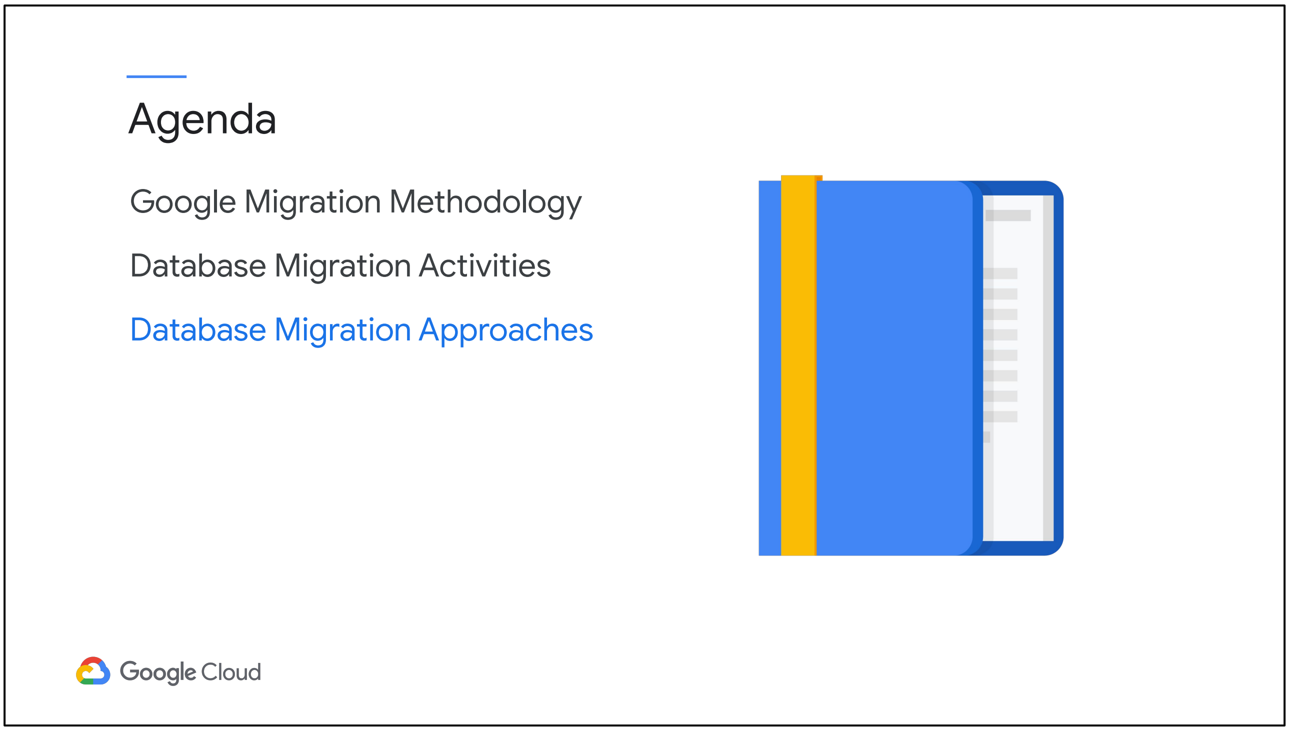
The last two activities in the database migration process are to promote the new database and retire the old one.

You can use site reliability engineering techniques to ensure that the new server works before the switchover.

- **Blue/Green deployments** can mitigate the risks of that switchover. With a Blue/Green deployment, you run both environments at the same time. You test the new one before migrating workloads on it. When you're ready, switch the traffic. Thus, the blue becomes the green, and vice versa. If something fails, you can quickly switch them back. If not, you can eventually turn off the old environment.

- With a **Canary deployment**, you run both the old and new environments simultaneously. Gradually migrate production requests to the new environment and monitor for errors.
  If there are no errors, increase traffic to the new environment until it eventually takes over completely.

At some point, you're safe to retire the old environment.



Agenda

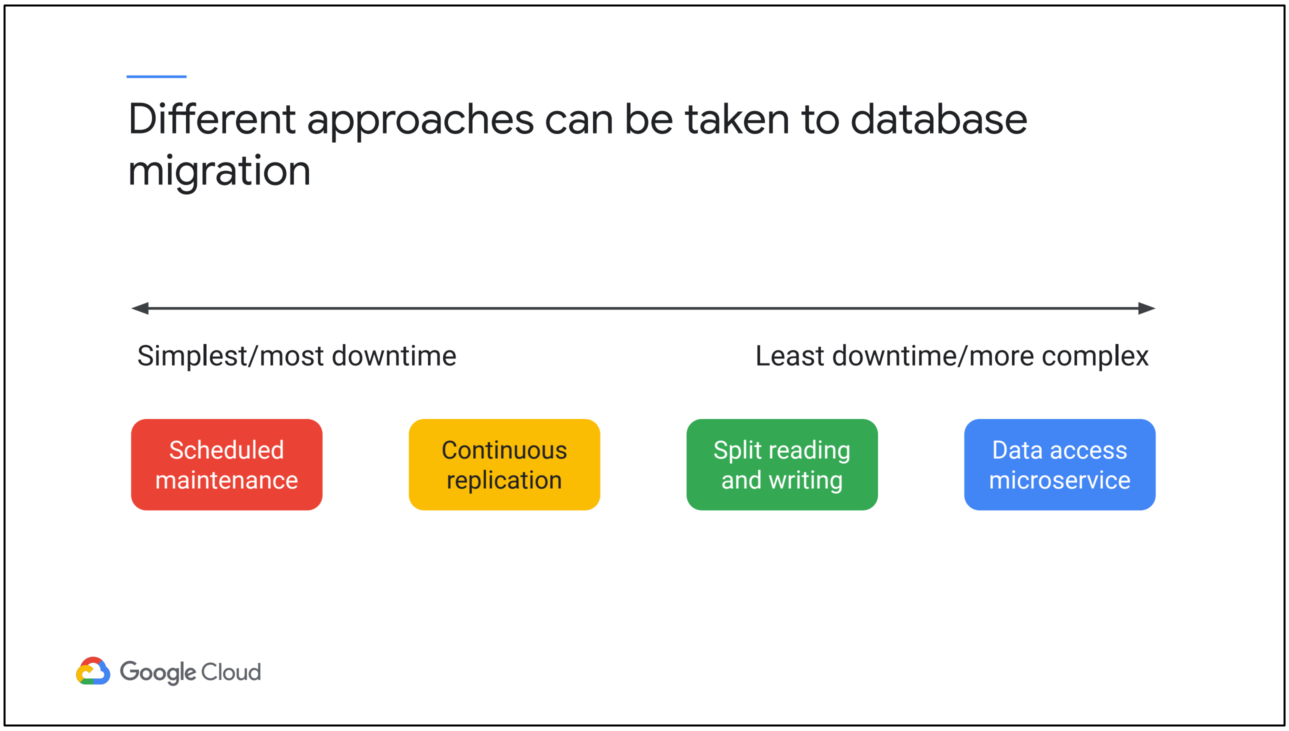
Google Migration Methodology

Database Migration Activities

Database Migration Approaches

Google Cloud

There are some different approaches you can take to database migration. Let's talk about those now.

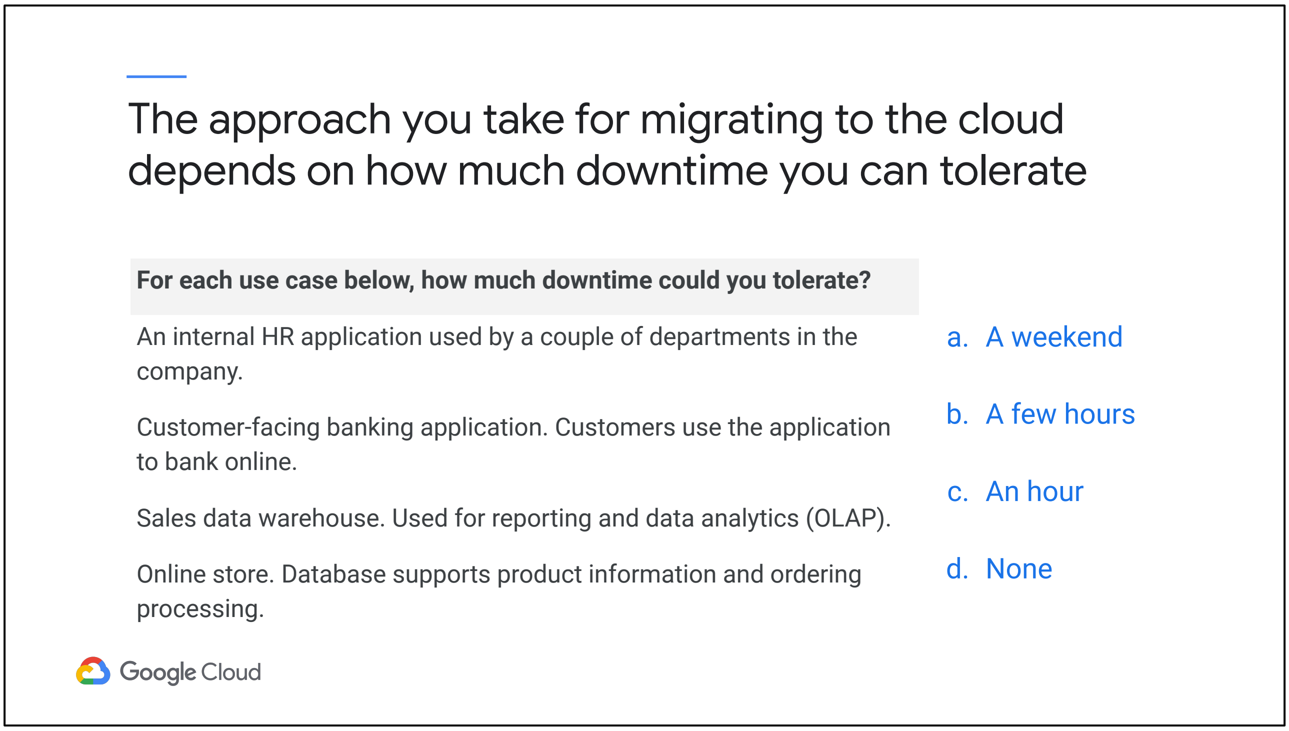


Different approaches can be taken to database migration

Simplest/most downtime
Least downtime/more complex
Scheduled maintenance
Continuous replication
Split reading and writing
Data access microservice

Google Cloud

Broadly speaking, there are four approaches you can take when migrating a database. They are: scheduled maintenance, continuous replication, split reading and writing, and data access microservice. The one you choose largely depends on the amount of downtime you can tolerate and the complexity of your current environment.



The approach you take for migrating to the cloud depends on how much downtime you can tolerate

For each use case below, how much downtime could you tolerate?

An internal HR application used by a couple of departments in the company.

Customer-facing banking application. Customers use the application to bank online.

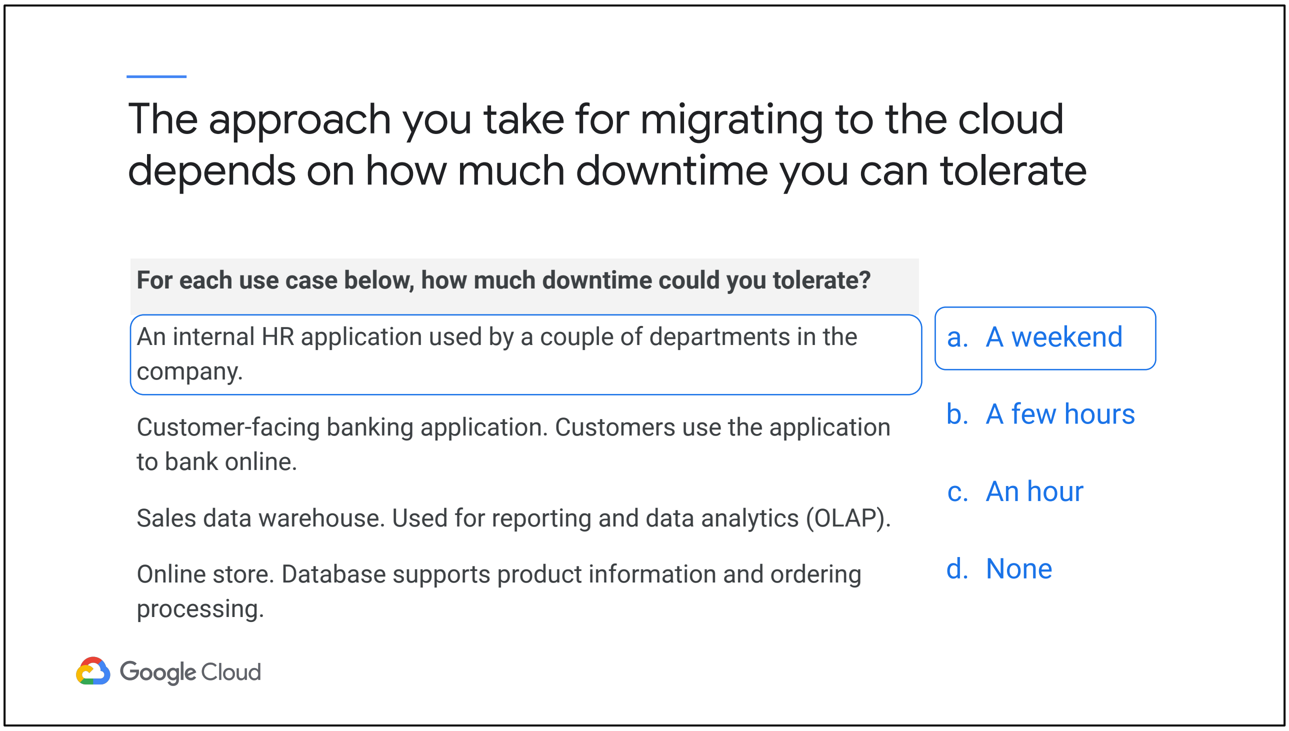
Sales data warehouse. Used for reporting and data analytics (OLAP).

Online store. Database supports product information and ordering processing.

a. A weekend
b. A few hours
c. An hour
d. None

Google Cloud

Here are four different use cases. For each one on the left, choose the amount of downtime you think you could tolerate on the right. Pause the video and take a couple minutes to do this.

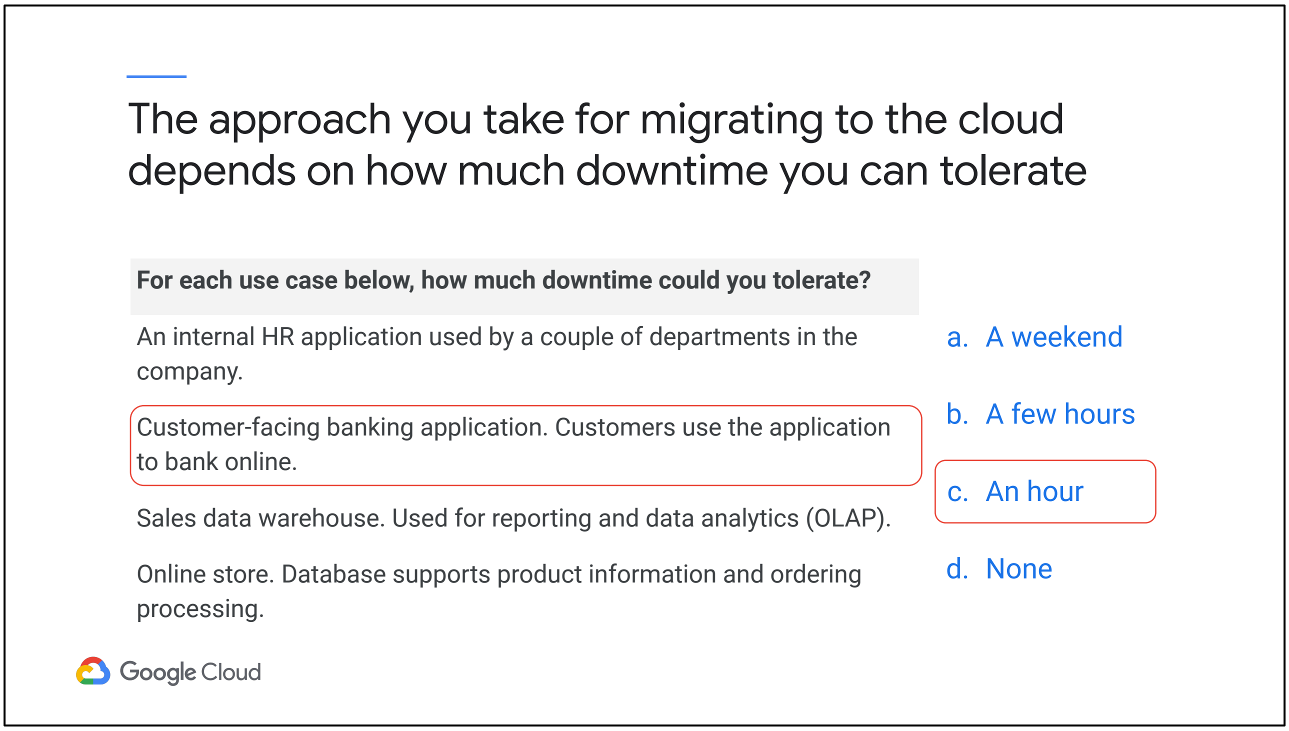


The approach you take for migrating to the cloud depends on how much downtime you can tolerate

For each use case below, how much downtime could you tolerate?	
An internal HR application used by a couple of departments in the company.	a. A weekend
Customer-facing banking application. Customers use the application to bank online.	b. A few hours
Sales data warehouse. Used for reporting and data analytics (OLAP).	c. An hour
Online store. Database supports product information and ordering processing.	d. None

Google Cloud

With so little information, maybe you can't be sure of the answers. But in the first case, the application is internal to the company without a large number of users. It would probably be OK if the application was down for a weekend.

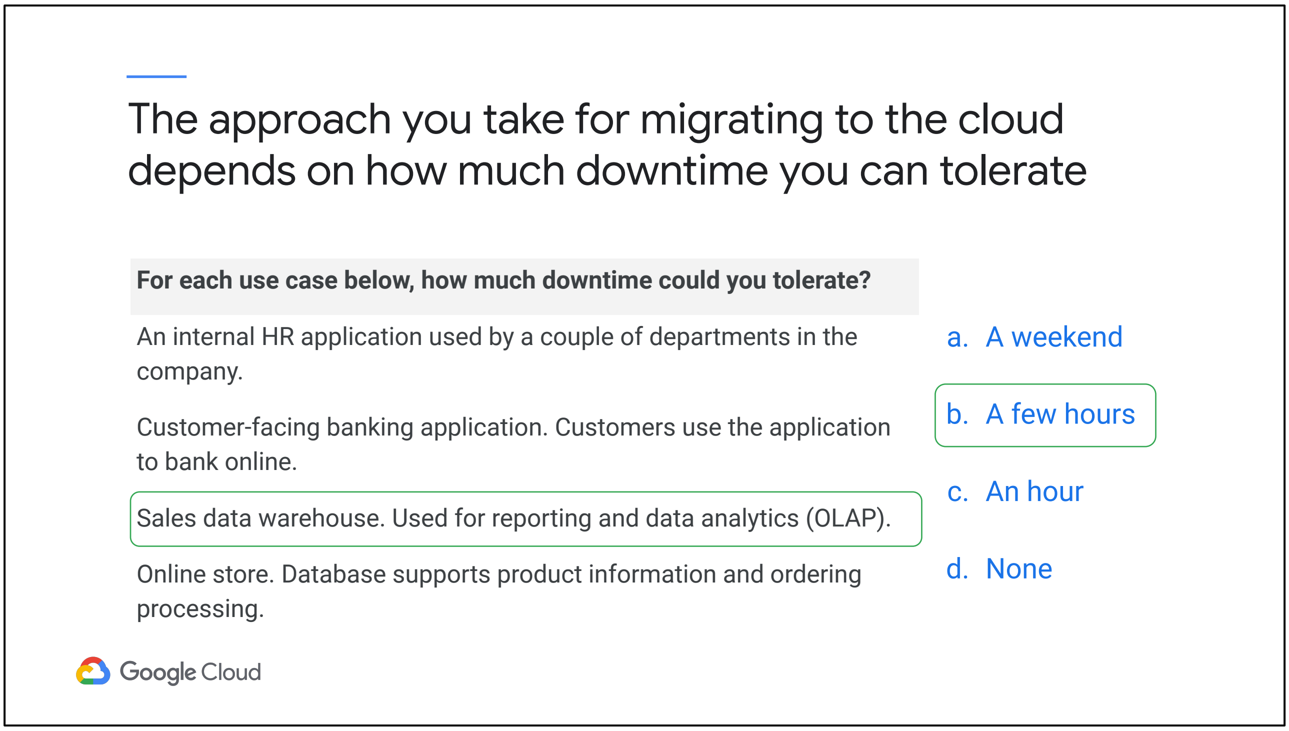


The approach you take for migrating to the cloud depends on how much downtime you can tolerate

For each use case below, how much downtime could you tolerate?	
An internal HR application used by a couple of departments in the company.	a. A weekend
Customer-facing banking application. Customers use the application to bank online.	b. A few hours
Sales data warehouse. Used for reporting and data analytics (OLAP).	c. An hour
Online store. Database supports product information and ordering processing.	d. None

Google Cloud

The second case describes a customer-facing, critical application. You might be able to tell users that the application is unavailable for an hour or so. The business might tell you that there can't be any downtime though.



The approach you take for migrating to the cloud depends on how much downtime you can tolerate

For each use case below, how much downtime could you tolerate?

An internal HR application used by a couple of departments in the company.

Customer-facing banking application. Customers use the application to bank online.

[✓ Sales data warehouse. Used for reporting and data analytics (OLAP).]

Online store. Database supports product information and ordering processing.

a. A weekend

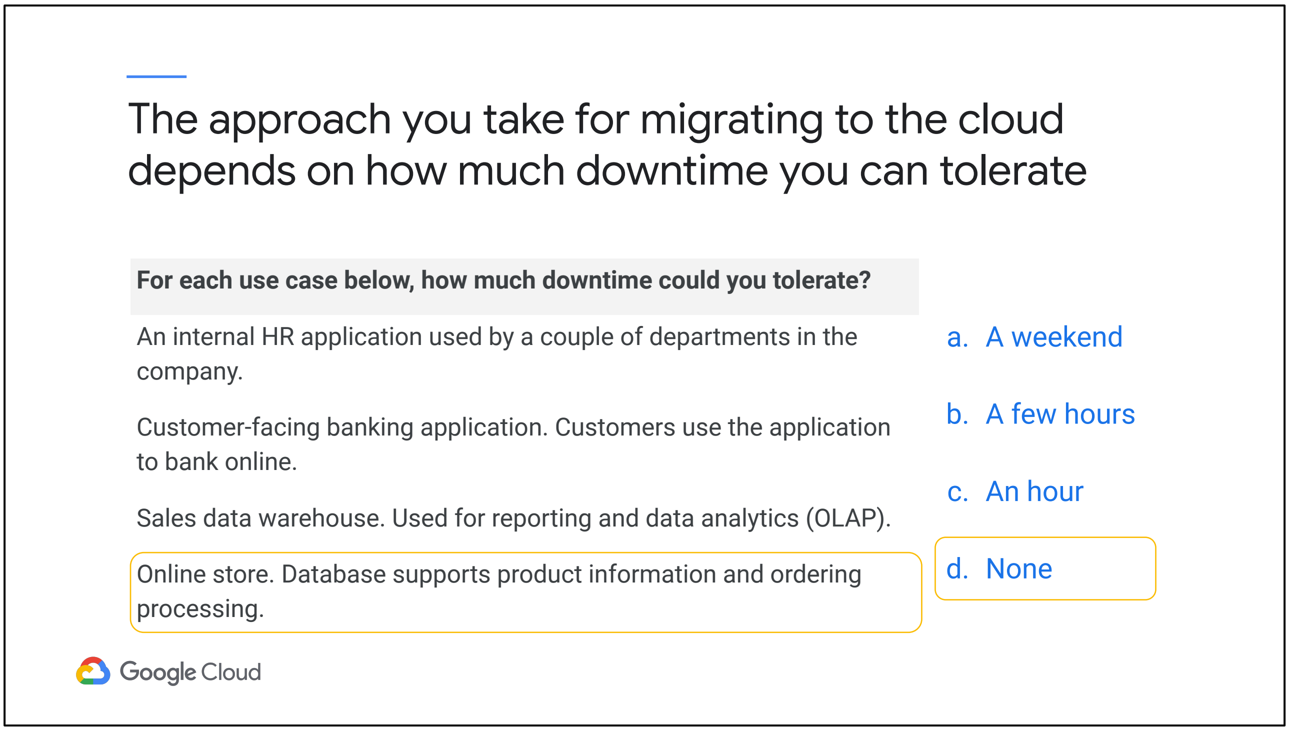
[✓ b. A few hours]

c. An hour

d. None

Google Cloud

The third case describes a data warehouse used for reporting and analytics. I'll assume users are internal, and we could function with a few hours of downtime—maybe even a weekend.



The approach you take for migrating to the cloud depends on how much downtime you can tolerate

For each use case below, how much downtime could you tolerate?

An internal HR application used by a couple of departments in the company.

Customer-facing banking application. Customers use the application to bank online.

Sales data warehouse. Used for reporting and data analytics (OLAP).

Online store. Database supports product information and ordering processing.

a. A weekend

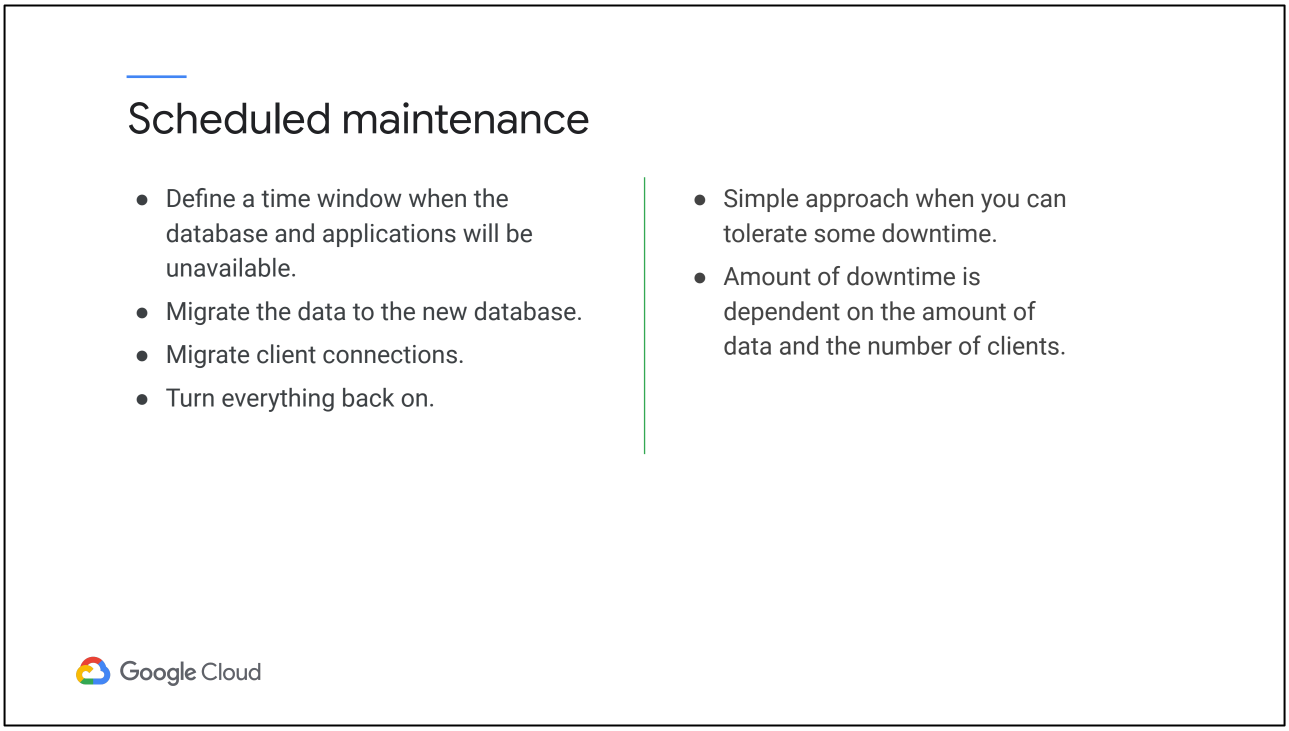
b. A few hours

c. An hour

d. None

Google Cloud

The last case describes a database that is critical to sales in our online store. If I'm running the business, I don't want to lose orders. I want as close to no downtime as possible.



Scheduled maintenance

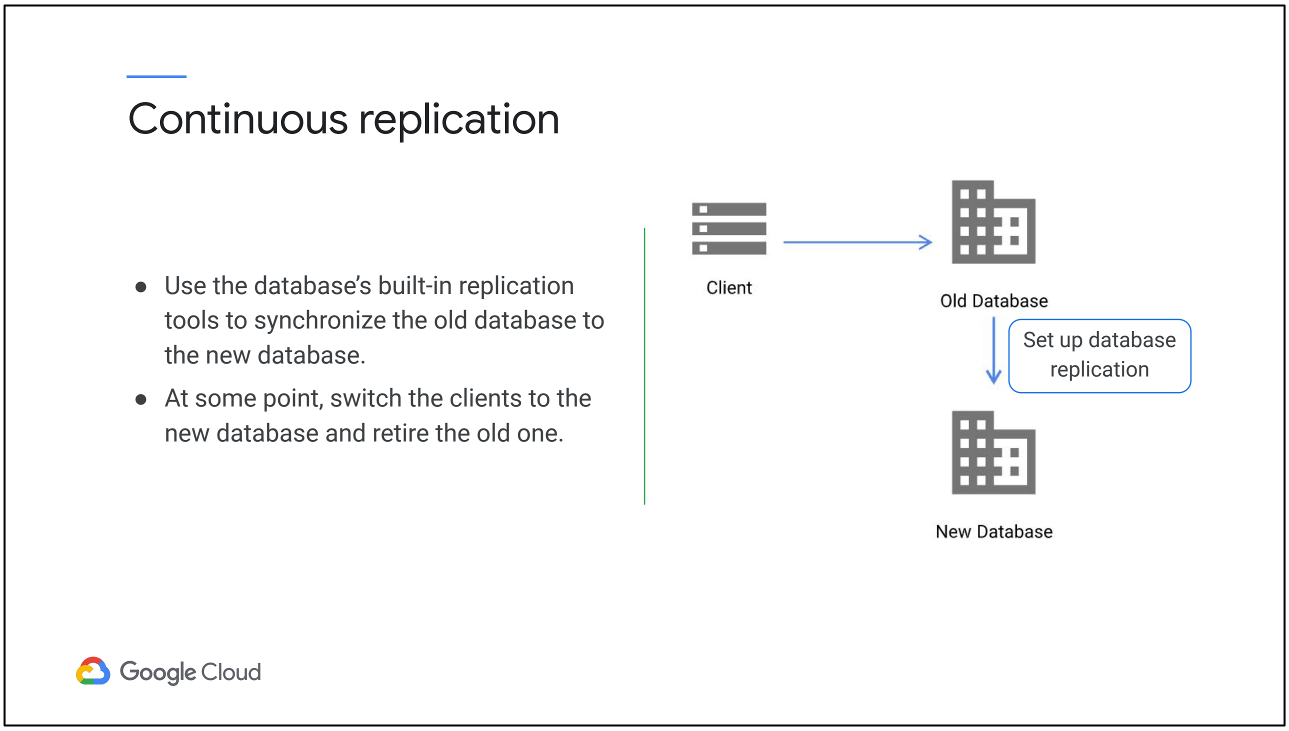
- Define a time window when the database and applications will be unavailable.
- Migrate the data to the new database.
- Migrate client connections.
- Turn everything back on.

- Simple approach when you can tolerate some downtime.
- Amount of downtime is dependent on the amount of data and the number of clients.

Google Cloud

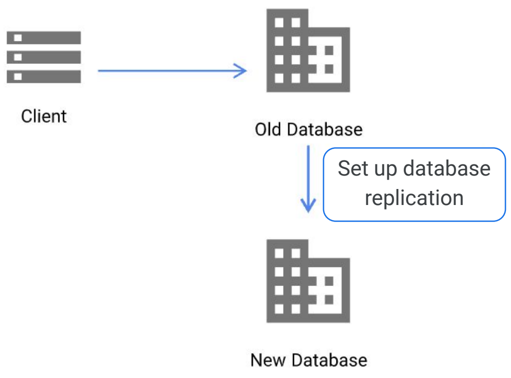
Use scheduled maintenance if you can tolerate some downtime.
Define a time window when the database and applications will be unavailable. Migrate the data to the new database, then migrate client connections. Lastly, turn everything back on.

This approach is simple. The length of the window will be dependent on the amount of data you need to transfer. When the database is very large, you can do an initial migration of historical data. Then when the time comes to do the migration, you only need to worry about new data added since the initial transfer.



Continuous replication

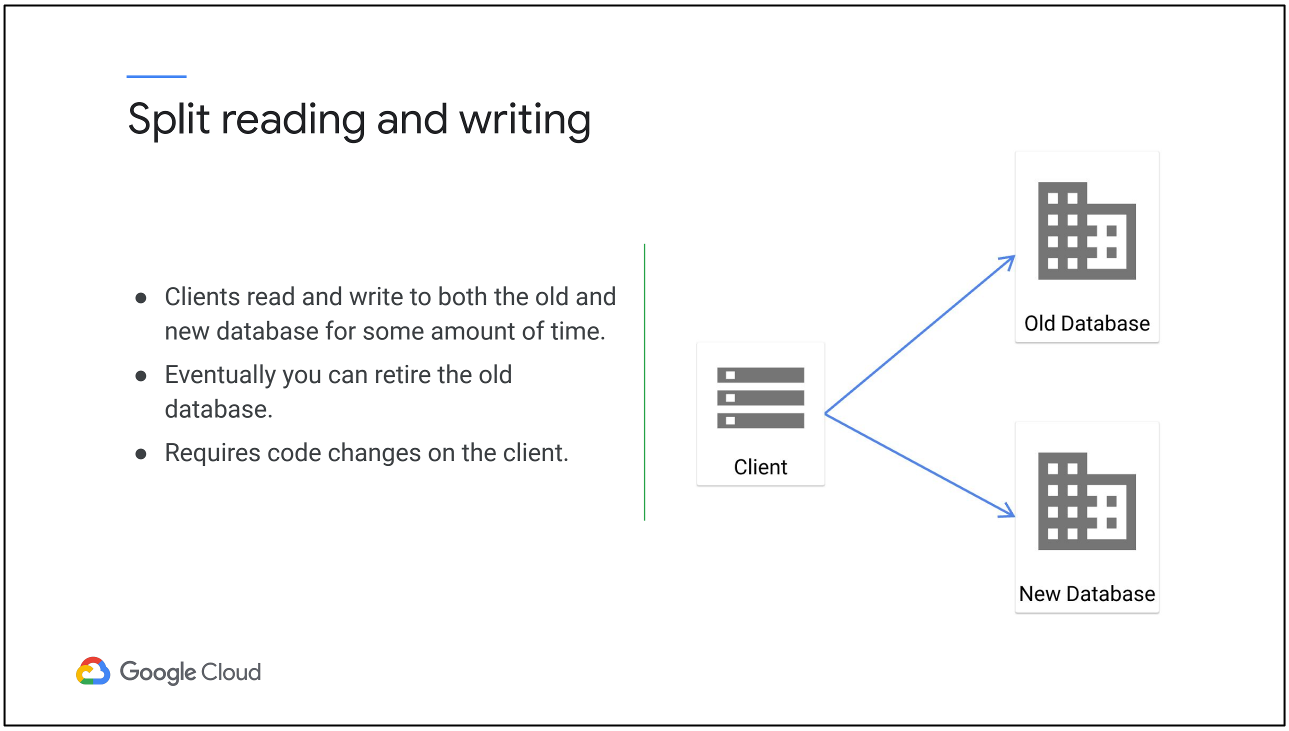
- Use the database's built-in replication tools to synchronize the old database to the new database.
- At some point, switch the clients to the new database and retire the old one.



Google Cloud

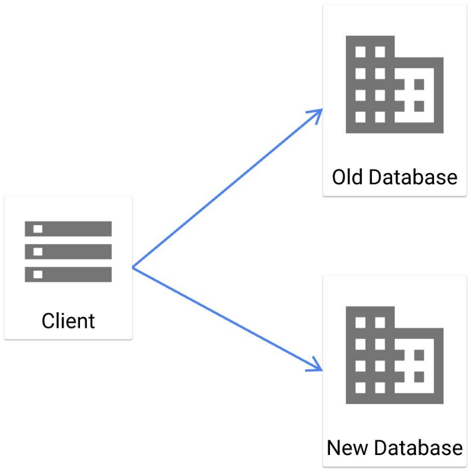
Continuous replication uses your database's built-in replication tools to synchronize the old database to the new database. This is relatively simple to set up and can be done by your database administrator. There are also third-party tools like Striim that will automate this process.

Eventually, you will move the client connections from the old database to the new one. Then you can turn off the replication and retire the old site.



Split reading and writing

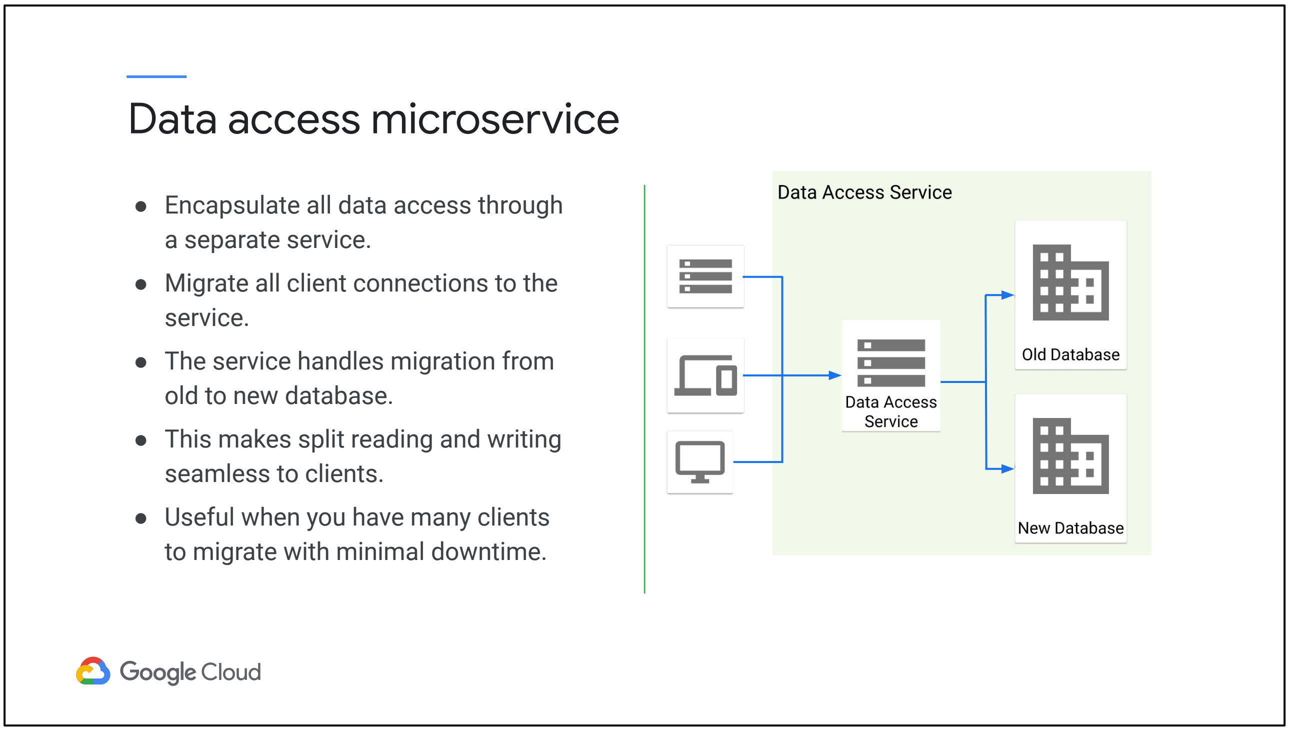
- Clients read and write to both the old and new database for some amount of time.
- Eventually you can retire the old database.
- Requires code changes on the client.



Google Cloud

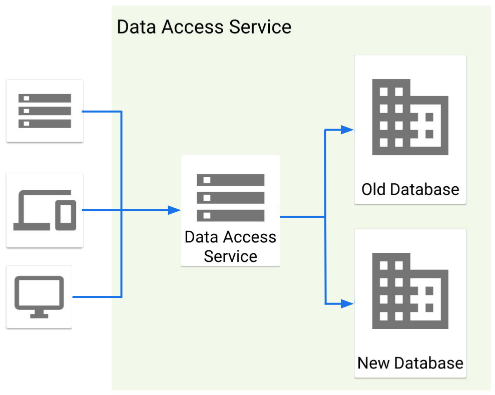
With split reading and writing, the clients read and write to both the old and new databases for some amount of time. Eventually, you can retire the old database.

Obviously, this requires code changes on the client. You would only do this when you are migrating to different types of databases; for example, if you are migrating from Oracle to Spanner. If you are migrating from Oracle on-premises to Oracle on Google Cloud Bare Metal Solution, continuous replication would make more sense.



Data access microservice

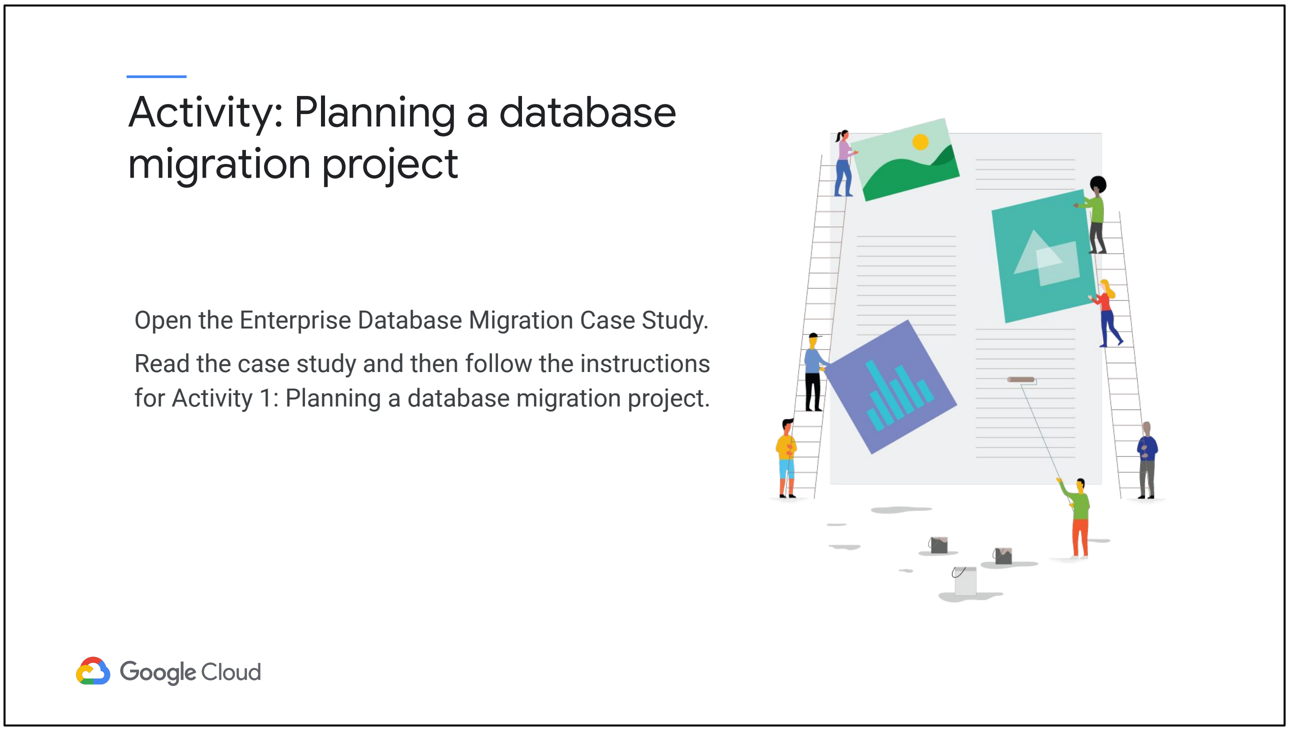
- Encapsulate all data access through a separate service.
- Migrate all client connections to the service.
- The service handles migration from old to new database.
- This makes split reading and writing seamless to clients.
- Useful when you have many clients to migrate with minimal downtime.



Google Cloud

If you have a large number of clients, split reading and writing is impractical. It would require too many code changes. In that case, you can create a data access microservice.

All data access is encapsulated or hidden behind the service. First, migrate all client connections to the service; the service then handles migration from the old to the new database. Essentially, this makes split reading and writing seamless to clients. This is a complex approach, but it is useful when you have many clients to migrate and can tolerate minimal or no downtime.



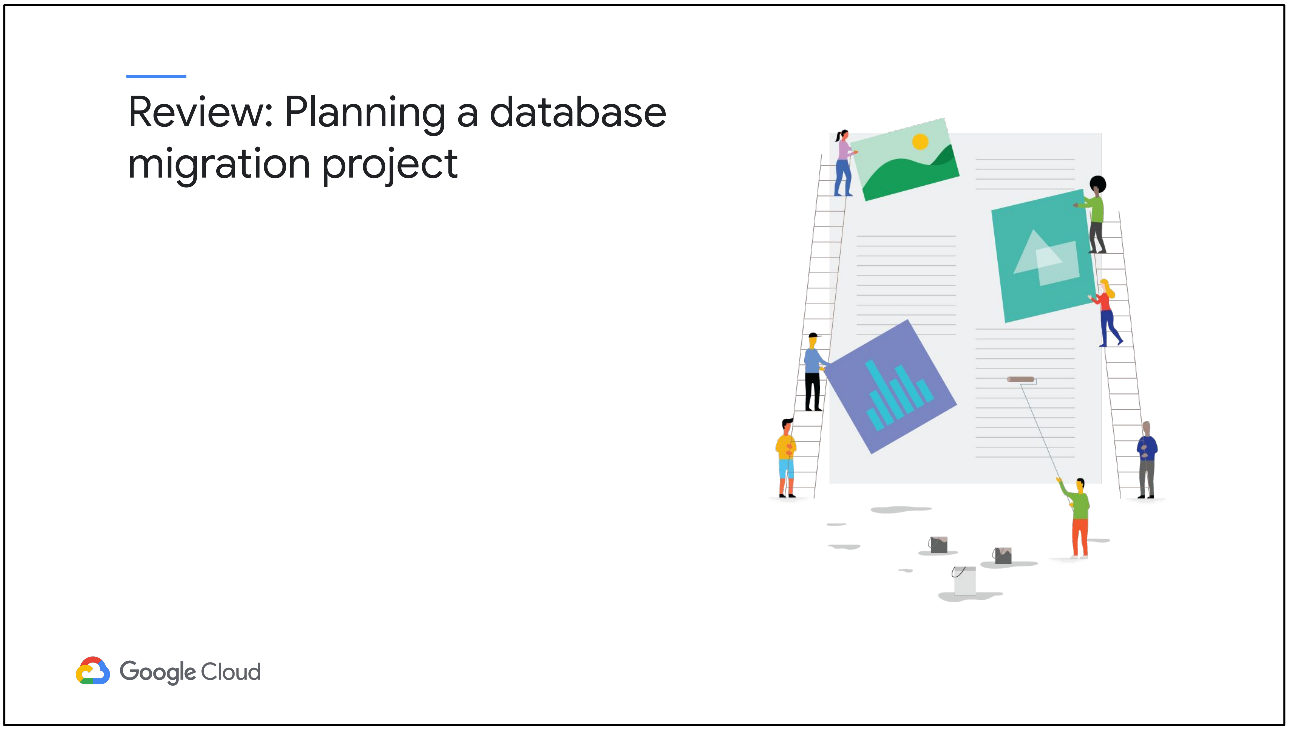
Activity: Planning a database migration project

Open the Enterprise Database Migration Case Study.

Read the case study and then follow the instructions for Activity 1: Planning a database migration project.

Google Cloud

In this activity, you analyze a case study and perform some of the planning tasks we covered in this module. The Enterprise Database Migration case study is available for download from the course outline. The case study represents a fictitious enterprise customer who wants to move to the cloud. Like all large organizations, it has a mix of many different applications and databases of varying importance. Review the case study and then follow the instructions for Activity 1.

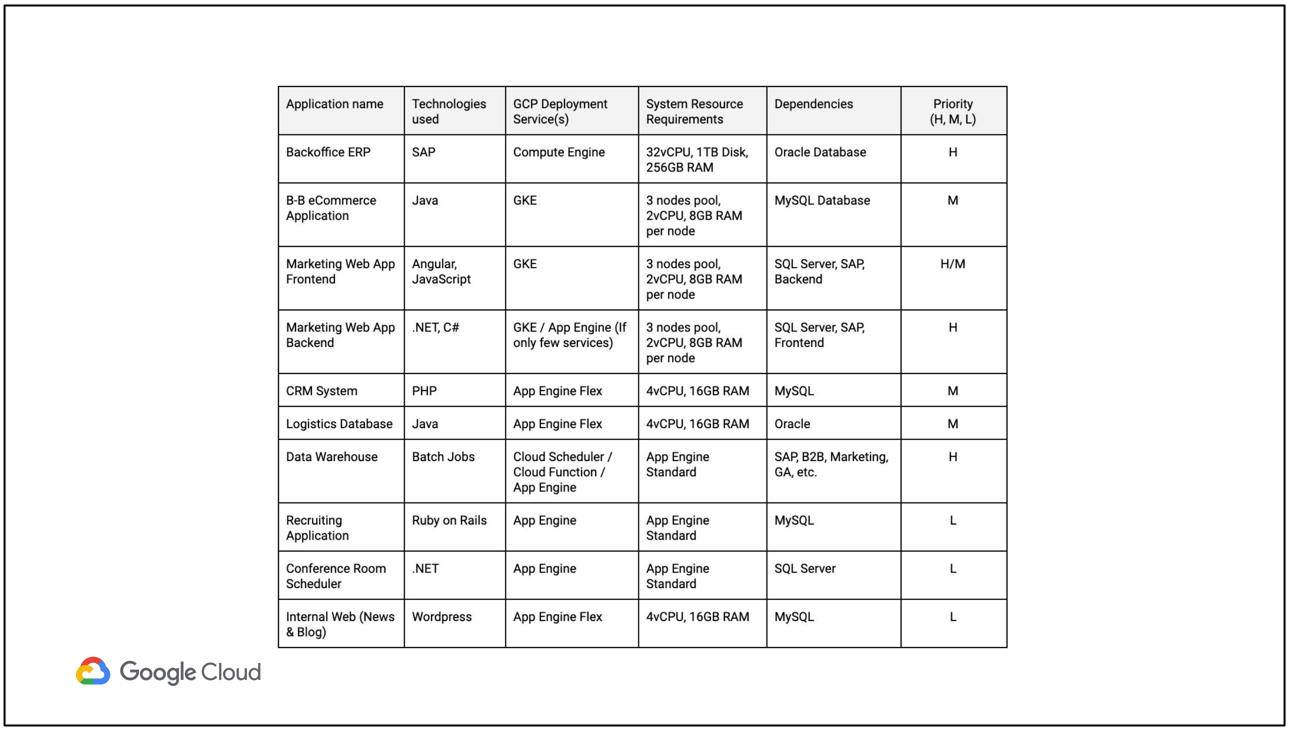


Review: Planning a database migration project

Google Cloud

In this activity, you analyzed a case study and performed some of the planning tasks we covered in this module. The case study is a fictitious enterprise customer who wants to move to the cloud. Like all large organizations, it has a mix of many different applications and databases of varying importance.

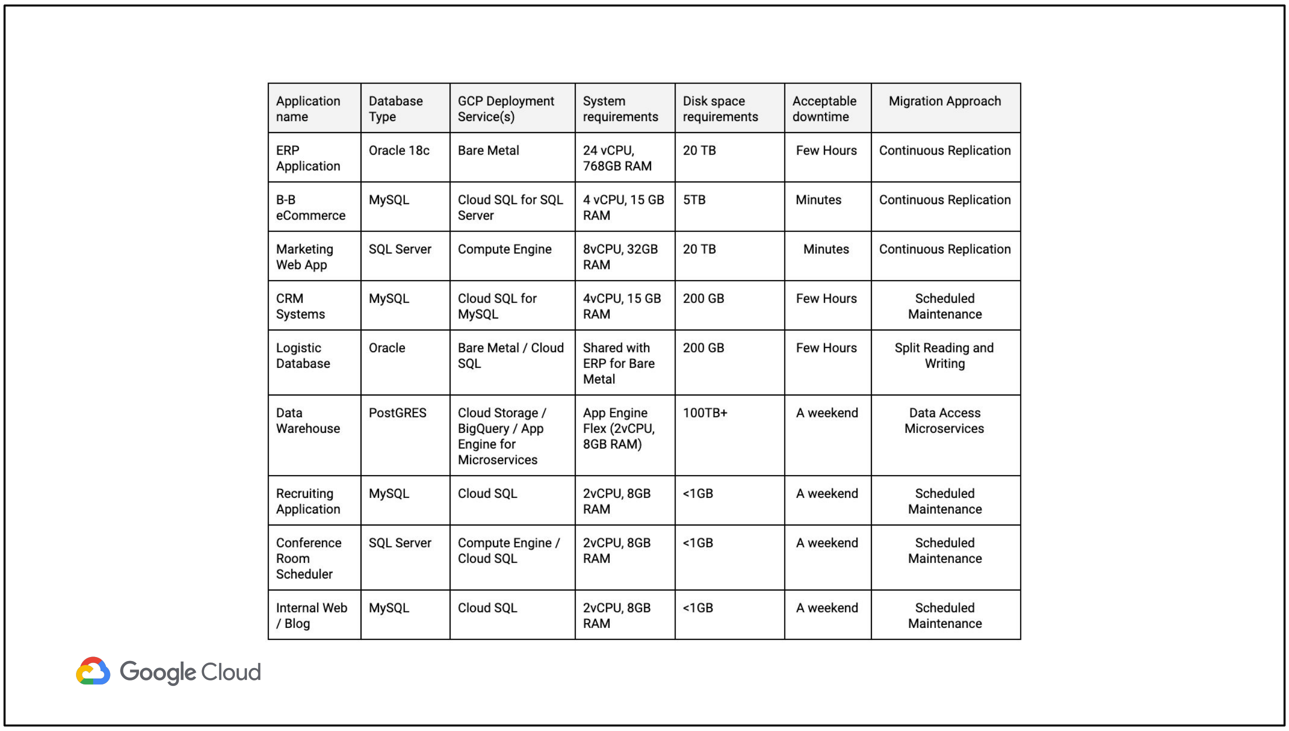
In activity one you were asked to create an application Inventory, outline database dependencies and build an app catalog.



Application name	Technologies used	GCP Deployment Service(s)	System Resource Requirements	Dependencies	Priority (H, M, L)
Backoffice ERP	SAP	Compute Engine	32vCPU, 1TB Disk, 256GB RAM	Oracle Database	H
B-B eCommerce Application	Java	GKE	3 nodes pool, 2vCPU, 8GB RAM per node	MySQL Database	M
Marketing Web App Frontend	Angular, JavaScript	GKE	3 nodes pool, 2vCPU, 8GB RAM per node	SQL Server, SAP, Backend	H/M
Marketing Web App Backend	.NET, C#	GKE / App Engine (If only few services)	3 nodes pool, 2vCPU, 8GB RAM per node	SQL Server, SAP, Frontend	H
CRM System	PHP	App Engine Flex	4vCPU, 16GB RAM	MySQL	M
Logistics Database	Java	App Engine Flex	4vCPU, 16GB RAM	Oracle	M
Data Warehouse	Batch Jobs	Cloud Scheduler / Cloud Function / App Engine	App Engine Standard	SAP, B2B, Marketing, GA, etc.	H
Recruiting Application	Ruby on Rails	App Engine	App Engine Standard	MySQL	L
Conference Room Scheduler	.NET	App Engine	App Engine Standard	SQL Server	L
Internal Web (News & Blog)	Wordpress	App Engine Flex	4vCPU, 16GB RAM	MySQL	L

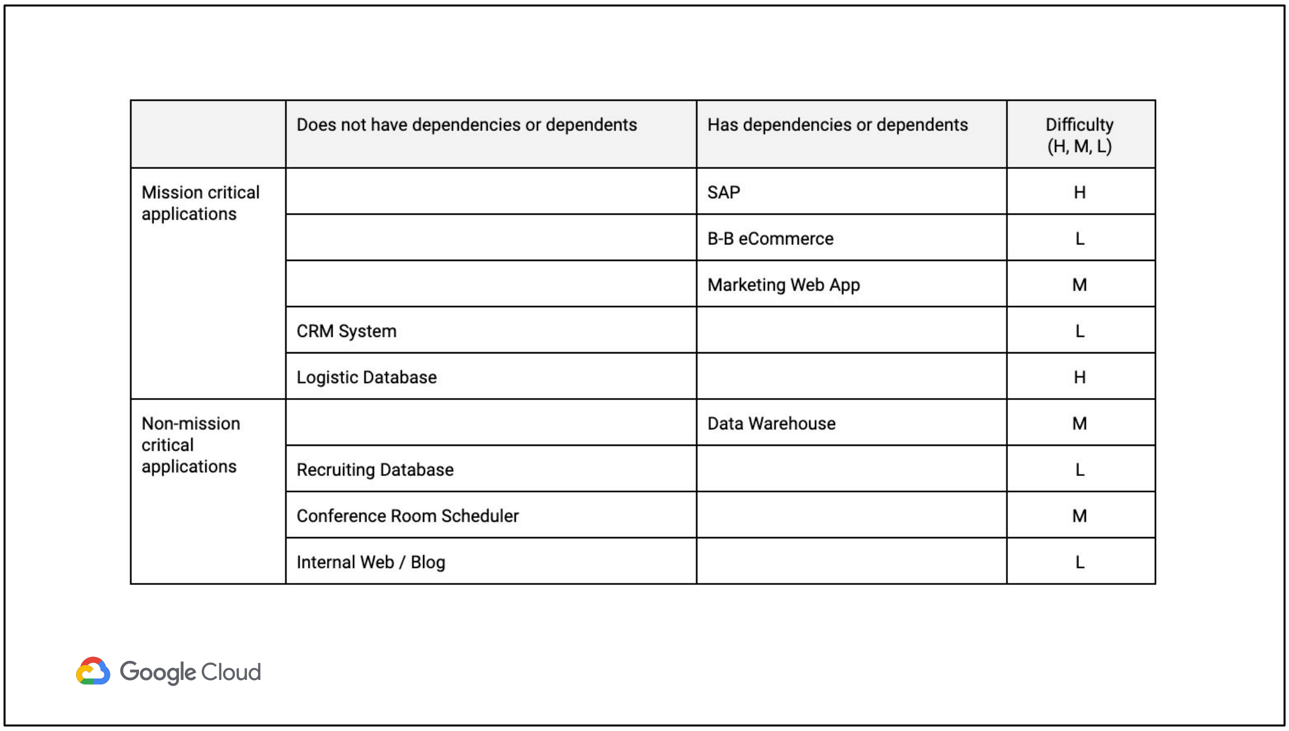
Google Cloud

Here is an example application inventory that was done by a prior student. You may pause the video to look at it in detail.



Application name	Database Type	GCP Deployment Service(s)	System requirements	Disk space requirements	Acceptable downtime	Migration Approach
ERP Application	Oracle 18c	Bare Metal	24 vCPU, 768GB RAM	20 TB	Few Hours	Continuous Replication
B-B eCommerce	MySQL	Cloud SQL for SQL Server	4 vCPU, 15 GB RAM	5TB	Minutes	Continuous Replication
Marketing Web App	SQL Server	Compute Engine	8vCPU, 32GB RAM	20 TB	Minutes	Continuous Replication
CRM Systems	MySQL	Cloud SQL for MySQL	4vCPU, 15 GB RAM	200 GB	Few Hours	Scheduled Maintenance
Logistic Database	Oracle	Bare Metal / Cloud SQL	Shared with ERP for Bare Metal	200 GB	Few Hours	Split Reading and Writing
Data Warehouse	PostGRES	Cloud Storage / BigQuery / App Engine for Microservices	App Engine Flex (2vCPU, 8GB RAM)	100TB+	A weekend	Data Access Microservices
Recruiting Application	MySQL	Cloud SQL	2vCPU, 8GB RAM	<1GB	A weekend	Scheduled Maintenance
Conference Room Scheduler	SQL Server	Compute Engine / Cloud SQL	2vCPU, 8GB RAM	<1GB	A weekend	Scheduled Maintenance
Internal Web / Blog	MySQL	Cloud SQL	2vCPU, 8GB RAM	<1GB	A weekend	Scheduled Maintenance

Here are the database dependencies. You may pause the video to look at it in detail.



	Does not have dependencies or dependents	Has dependencies or dependents	Difficulty (H, M, L)
Mission critical applications		SAP	H
		B-B eCommerce	L
		Marketing Web App	M
	CRM System		L
	Logistic Database		H
Non-mission critical applications		Data Warehouse	M
	Recruiting Database		L
	Conference Room Scheduler		M
	Internal Web / Blog		L

Google Cloud

Lastly, here is the completed App Catalog. You may pause the video to look at it in detail.

Module review

Google Cloud

In this module, you learned how to migrate to the cloud using Google's implementation methodology. Google's methodology consists of 4 steps: Assess, Plan, Deploy and Optimize.

You also learned what the key database migration activities are. They are:

Transfer data from old site to new.
Resolve data integration issues.
Validate data migration.
Promote the new site.
And finally, retire the old site.

In the final section of the module you learned how to choose the appropriate database migration approach: scheduled maintenance, continuous replication, split reading and writing, or data access microservice.

Google Cloud

Google Cloud