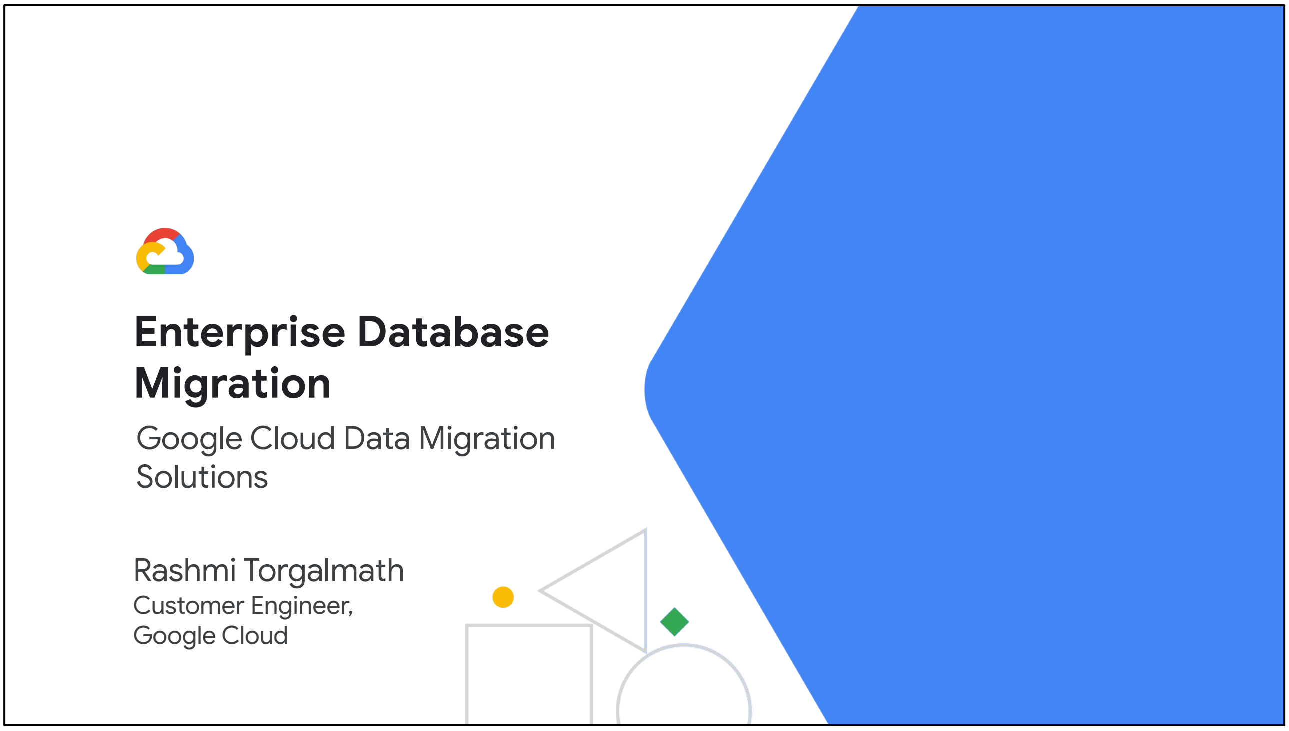


Enterprise Database Migration

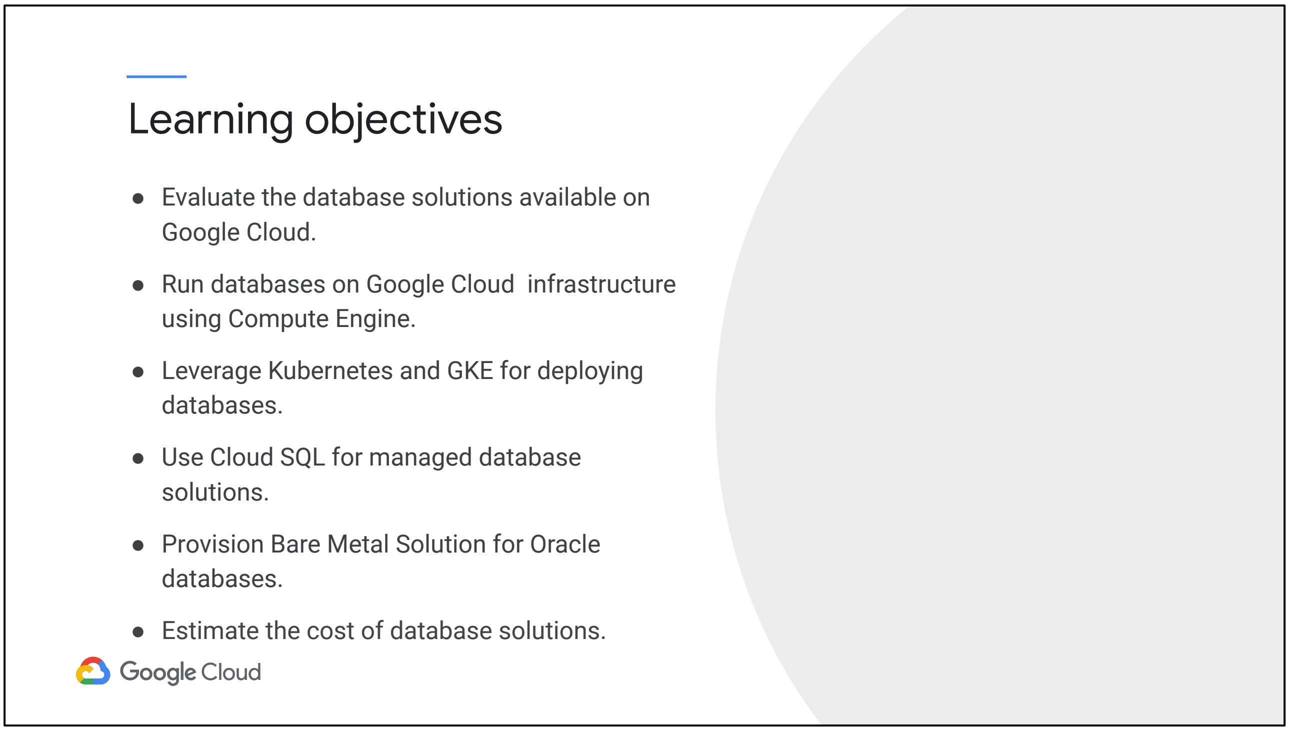
Google Cloud Data Migration Solutions

Rashmi Torgalmath
Customer Engineer,
Google Cloud

Hello, I am Rashmi Torgalmath and I'm a Customer Engineer at Google. Welcome back to Enterprise Database Migration.

There are many ways to run databases in the cloud, each with its own advantages.

In this module, you learn about the various solutions Google provides for running your databases.

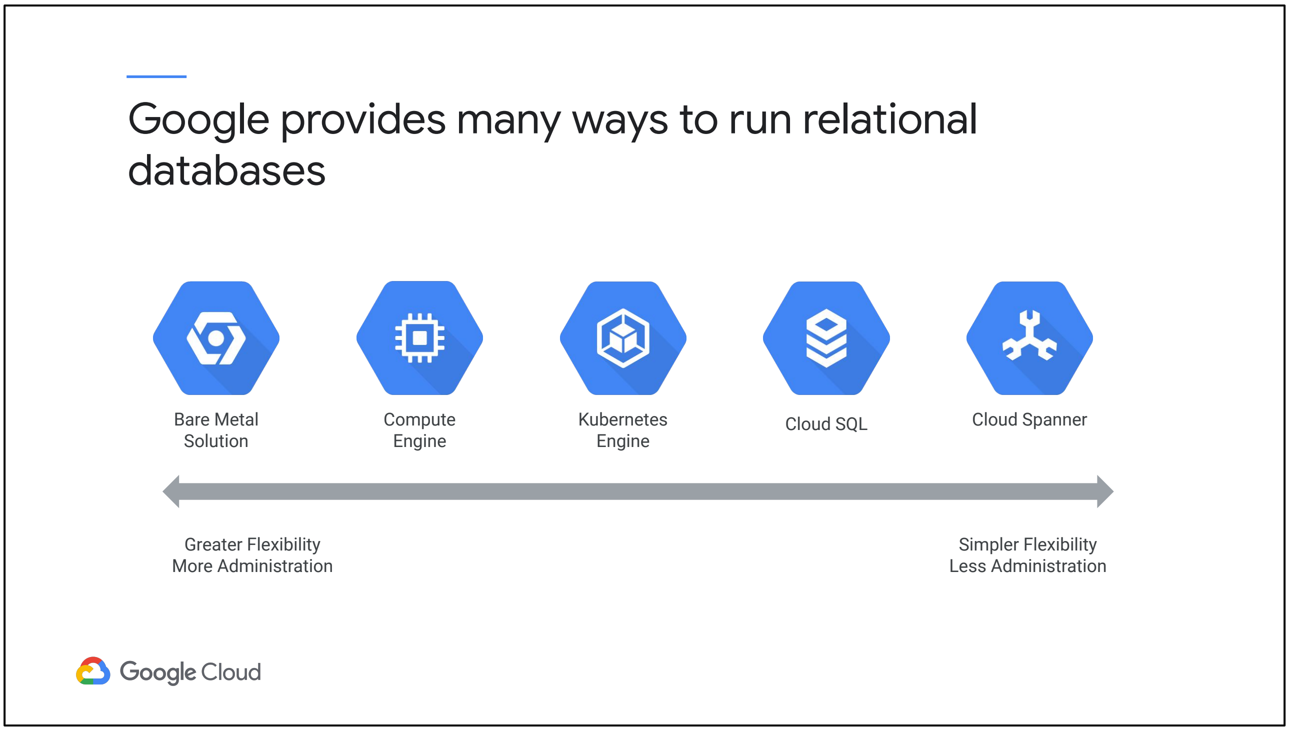


Learning objectives

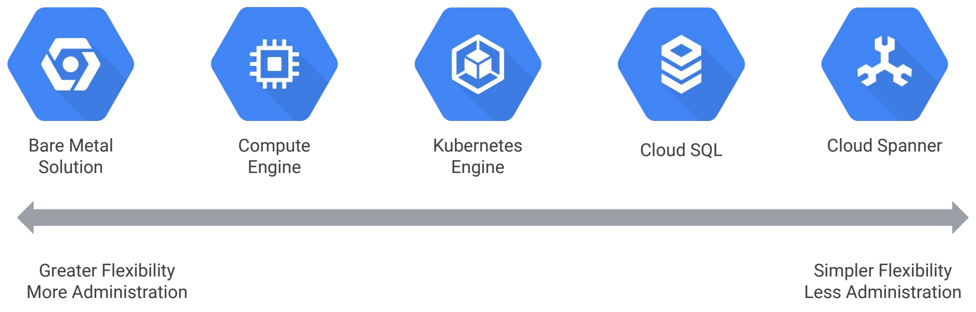
- Evaluate the database solutions available on Google Cloud.
- Run databases on Google Cloud infrastructure using Compute Engine.
- Leverage Kubernetes and GKE for deploying databases.
- Use Cloud SQL for managed database solutions.
- Provision Bare Metal Solution for Oracle databases.
- Estimate the cost of database solutions.

Google Cloud

- You get some hands-on experience using these database solutions and learn what you need to evaluate each one for your own use cases.
- You learn to use Compute Engine to run your databases on virtual machines using Google's infrastructure.
- You learn to configure databases to run in a Kubernetes cluster using GKE.
- You automate the creation and administration of databases using Cloud SQL.
- You learn how you can run Oracle databases using Google's Bare Metal Solution.
- And, you use the Google Cloud price calculator to estimate the cost of running a database using these various solutions.

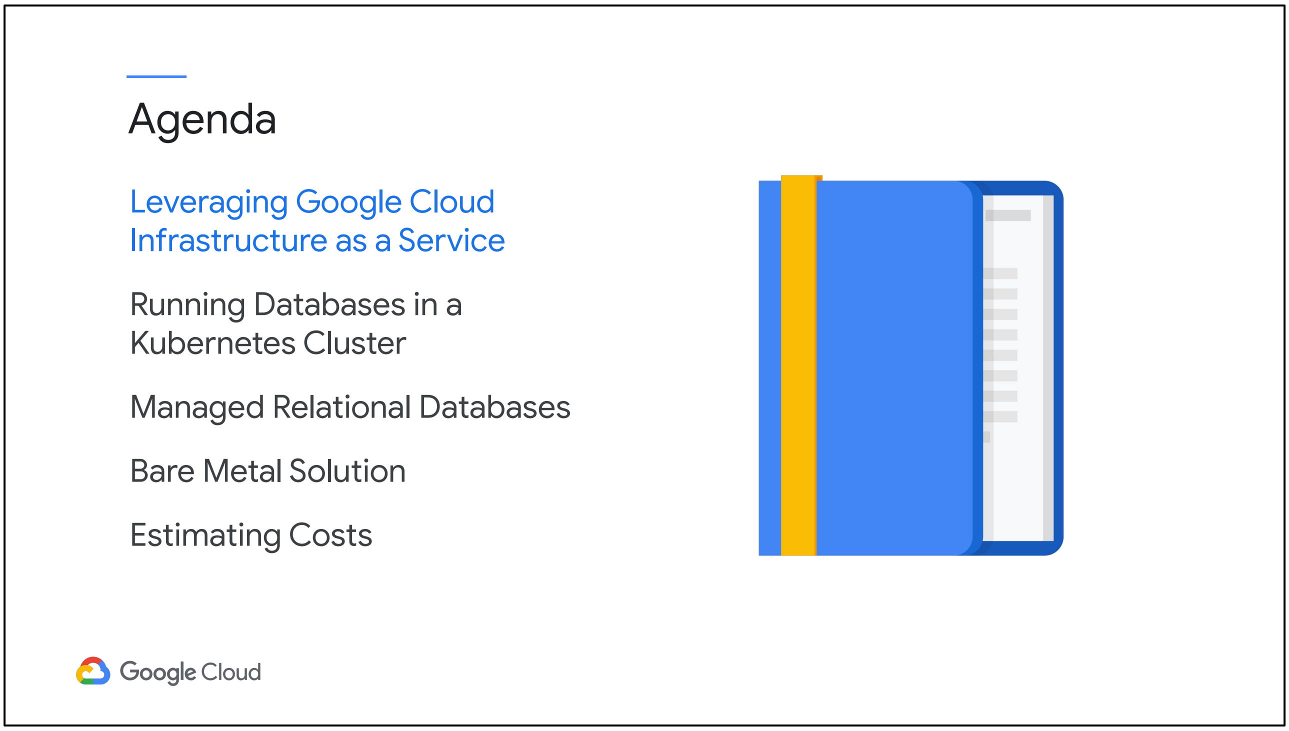


Google provides many ways to run relational databases


Bare Metal Solution
Compute Engine
Kubernetes Engine
Cloud SQL
Cloud Spanner
Greater Flexibility More Administration
Simpler Flexibility Less Administration

Google Cloud

Google provides many ways to run your relational database workloads. The one you choose depends on how much flexibility you need for your specific use cases. Managed databases are more automated and require less administration from you. Compute Engine and Kubernetes give you greater control over your deployment, but you have to do more work. With the Bare Metal Solution, you provision physical machines that you have complete control over.



Agenda

Leveraging Google Cloud Infrastructure as a Service

Running Databases in a Kubernetes Cluster

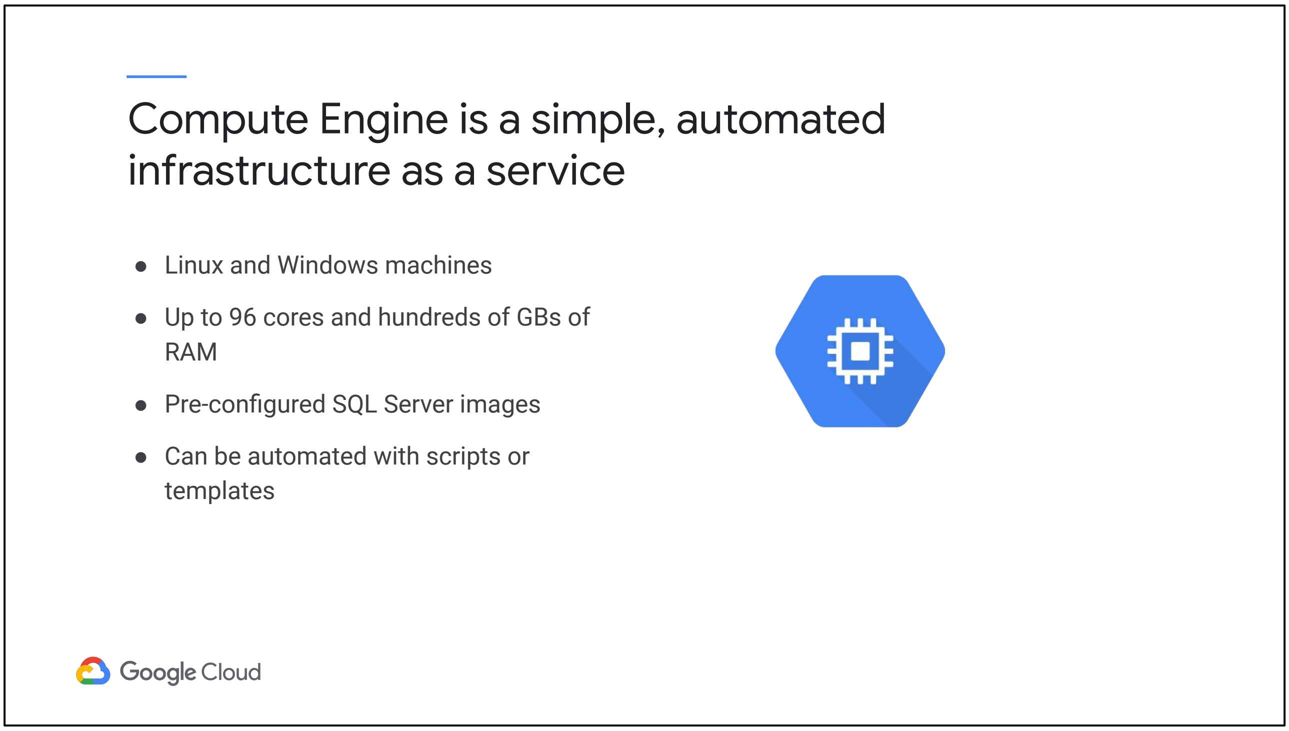
Managed Relational Databases

Bare Metal Solution

Estimating Costs

Google Cloud

Let's start by running databases on Google's infrastructure using Compute Engine.



Compute Engine is a simple, automated infrastructure as a service

- Linux and Windows machines
- Up to 96 cores and hundreds of GBs of RAM
- Pre-configured SQL Server images
- Can be automated with scripts or templates

Google Cloud

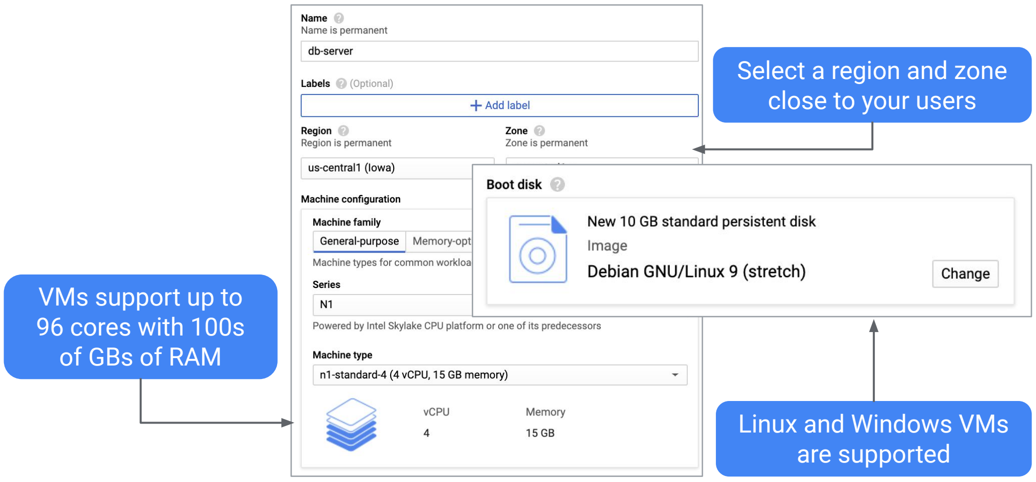
Compute Engine is the Google Cloud infrastructure-as-a-service product. You can run Linux or Windows virtual machines. Several different Linux distributions and Windows versions are supported.

Compute Engine supports very large VMs with up to 96 cores, hundreds of GBs of RAM, and hundreds of TBs of disk space.

For SQL Server users, Google maintains preconfigured virtual machine images with SQL Server already installed.

It is also easy to automate the creation of virtual machines using scripts, or Terraform, which you will see in this class.

Configure VMs to meet the database requirements


Name
Name is permanent
db-server
Labels (Optional)
Add label
Region
Region is permanent
us-central1 (Iowa)
Zone
Zone is permanent
Machine configuration
Machine family
General-purpose | Memory-opt
Machine types for common workloa
Series
N1
Powered by Intel Skylake CPU platform or one of its predecessors
Machine type
n1-standard-4 (4 vCPU, 15 GB memory)

vCPU	Memory
4	15 GB

Boot disk
New 10 GB standard persistent disk
Image
Debian GNU/Linux 9 (stretch)
Change

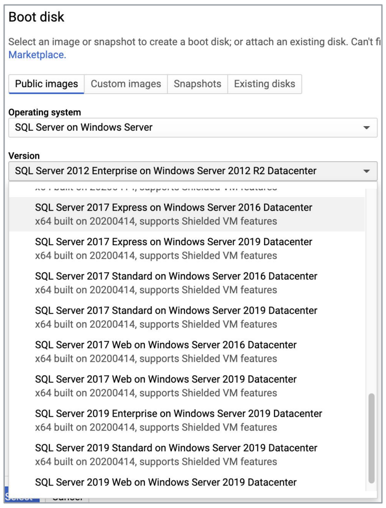
Select a region and zone close to your users
VMs support up to 96 cores with 100s of GBs of RAM
Linux and Windows VMs are supported

Google Cloud

- Configure your virtual machines as required to run your databases.
- Give the machine a name that describes its purpose.
  Specify the zone where you want the machine to live.
- Pick a boot disk. The boot disk is simply a persistent disk image that is configured with an operating system already installed.
- Finally, specify how much disk space is required for your machine.

Google provides pre-built SQL Server images

- SQL Server 2012 through 2019
- Express, Web, Standard, and Enterprise editions are supported.
- The license fee is included in the hourly rate:
  - BYOL is available through the Microsoft License Mobility for Google Cloud program.
- Images are built with VM Shielded security features:
  - Secure boot
  - Tamper-proof


Boot disk

Select an image or snapshot to create a boot disk; or attach an existing disk. Can't fi
Marketplace.

Operating system: SQL Server on Windows Server

Version: SQL Server 2012 Enterprise on Windows Server 2012 R2 Datacenter

SQL Server 2017 Express on Windows Server 2016 Datacenter
x64 built on 20200414, supports Shielded VM features

SQL Server 2017 Express on Windows Server 2019 Datacenter
x64 built on 20200414, supports Shielded VM features

SQL Server 2017 Standard on Windows Server 2016 Datacenter
x64 built on 20200414, supports Shielded VM features

SQL Server 2017 Standard on Windows Server 2019 Datacenter
x64 built on 20200414, supports Shielded VM features

SQL Server 2017 Web on Windows Server 2016 Datacenter
x64 built on 20200414, supports Shielded VM features

SQL Server 2017 Web on Windows Server 2019 Datacenter
x64 built on 20200414, supports Shielded VM features

SQL Server 2019 Enterprise on Windows Server 2019 Datacenter
x64 built on 20200414, supports Shielded VM features

SQL Server 2019 Standard on Windows Server 2019 Datacenter
x64 built on 20200414, supports Shielded VM features

SQL Server 2019 Web on Windows Server 2019 Datacenter

Google Cloud

If you want to run SQL Server, you can create a virtual machine and install SQL Server on it. Google makes it easier than that though. There are pre-built SQL Server images you can choose from.

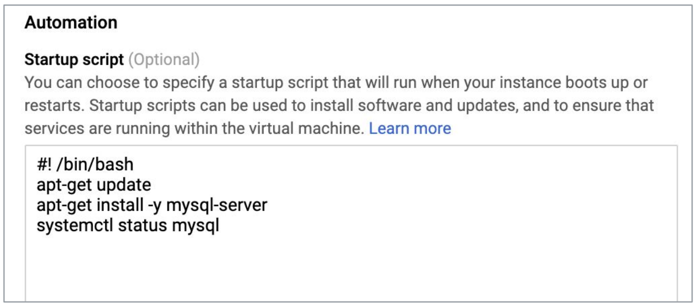
You will find multiple versions from SQL Server 2012 and up to the latest versions.

There are images configured to run the different SQL Server Editions: Express, Web, Standard, and Enterprise.

By default, SQL Server and Windows licenses are included in the hourly rate. However, you can bring your own license through the Microsoft License Mobility for Google Cloud program.

SQL Server images are also built using Google's Shielded VM security features. This ensures that the images have not been tampered with, and if they ever are, they will not boot.

Use a startup script to automate installation of VM components


Automation

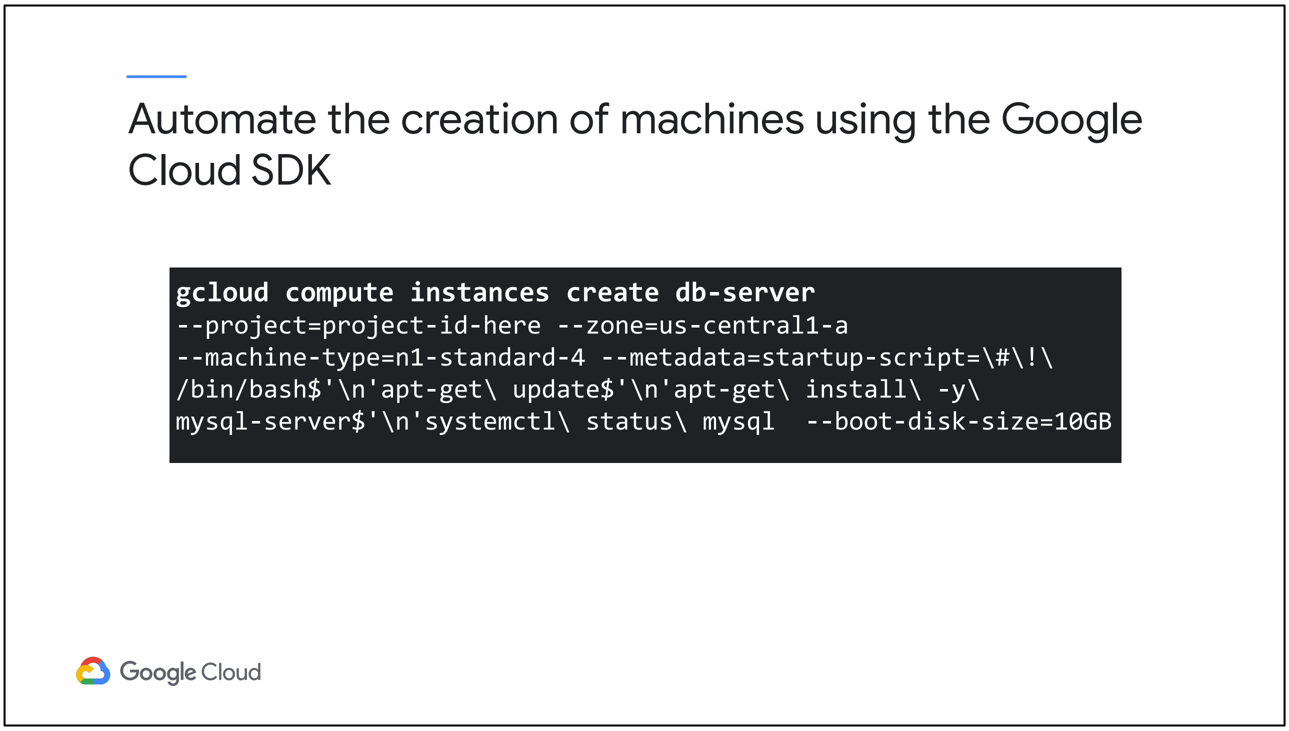
Startup script (Optional)
You can choose to specify a startup script that will run when your instance boots up or restarts. Startup scripts can be used to install software and updates, and to ensure that services are running within the virtual machine. Learn more

```
#! /bin/bash
apt-get update
apt-get install -y mysql-server
systemctl status mysql
```

Google Cloud

For automating the installation of components, you can include a startup script when configuring the machine. The script can be added to the machine configuration directly or you can refer to an external script stored in Cloud Storage.

For Linux machines, use Bash Shell scripts. For Windows, you can use batch, command, or Powershell scripts, whichever you prefer.



Automate the creation of machines using the Google Cloud SDK

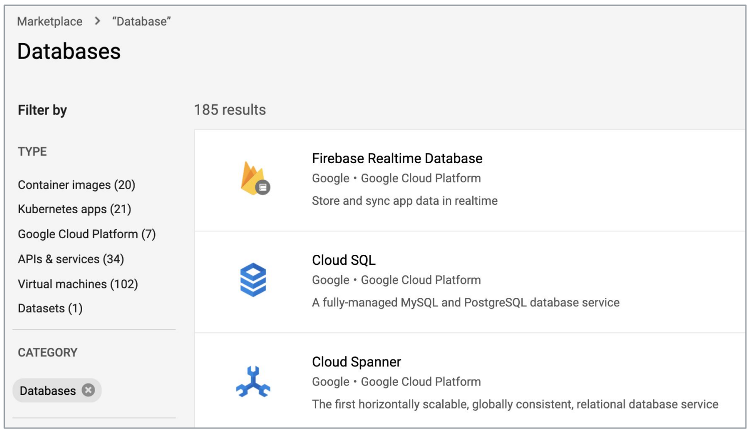
```
gcloud compute instances create db-server
--project=project-id-here --zone=us-central1-a
--machine-type=n1-standard-4 --metadata=startup-script=\#\!\
/bin/bash$'\n'apt-get\ update$'\n'apt-get\ install\ -y\
mysql-server$'\n'systemctl\ status\ mysql  --boot-disk-size=10GB
```

Google Cloud

Using the Google Cloud SDK, you can easily script the creation of your Compute Engine virtual machines and all other infrastructure resources like networks, firewall rules, instance templates, and instance groups.

To make scripting easy, you can use Google's web console to configure the resources and click the command line link at the bottom of the window. The SDK command for the resource as configured will be provided for you.

Marketplace provides preconfigured images for many different database types


Marketplace > "Database"

Databases

Filter by

185 results

TYPE

Container images (20)
Kubernetes apps (21)
Google Cloud Platform (7)
APIs & services (34)
Virtual machines (102)
Datasets (1)

CATEGORY

Databases

Firebase Realtime Database
Google • Google Cloud Platform
Store and sync app data in realtime

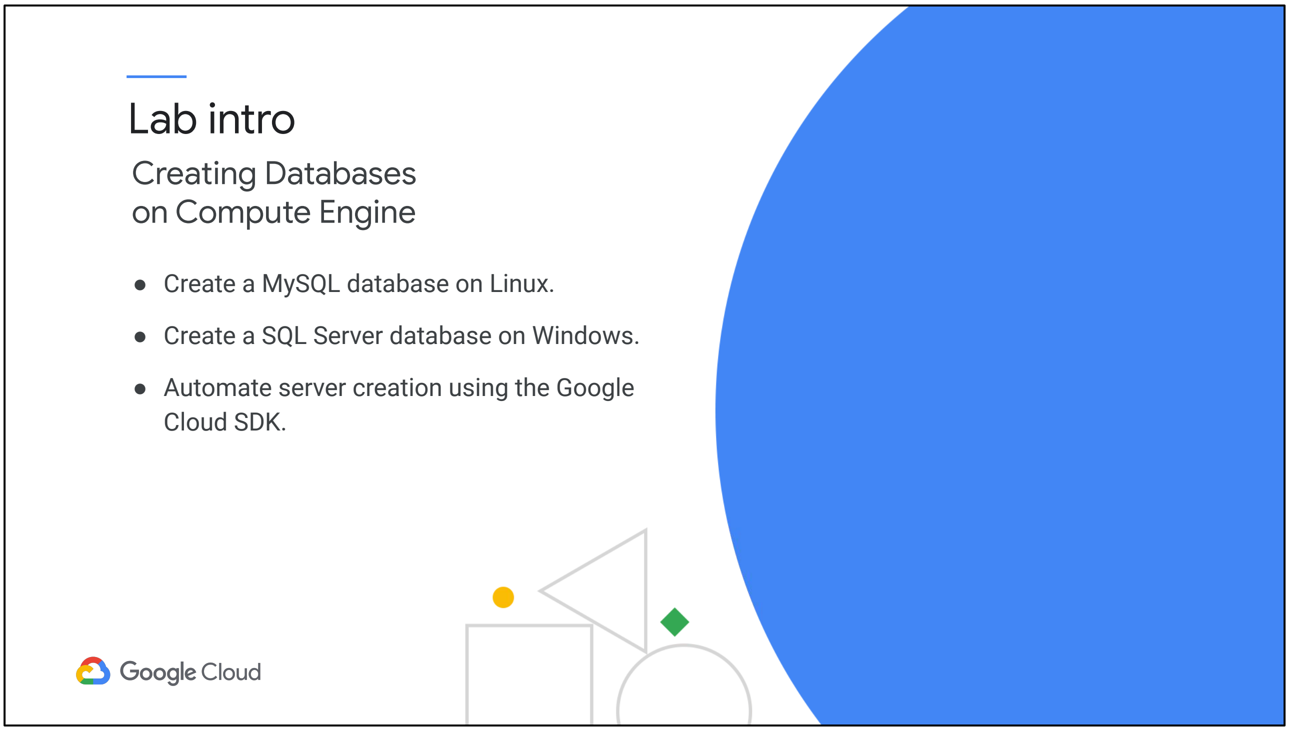
Cloud SQL
Google • Google Cloud Platform
A fully-managed MySQL and PostgreSQL database service

Cloud Spanner
Google • Google Cloud Platform
The first horizontally scalable, globally consistent, relational database service

Google Cloud

Another way to deploy virtual machines to Compute is through the Marketplace. Marketplace has many different preconfigured images for many different application types. Some of the images are provided by Google; others are provided by third parties.

Just search for what you are looking for. You will find many different databases, as well as many different software packages, pre-installed and configured.



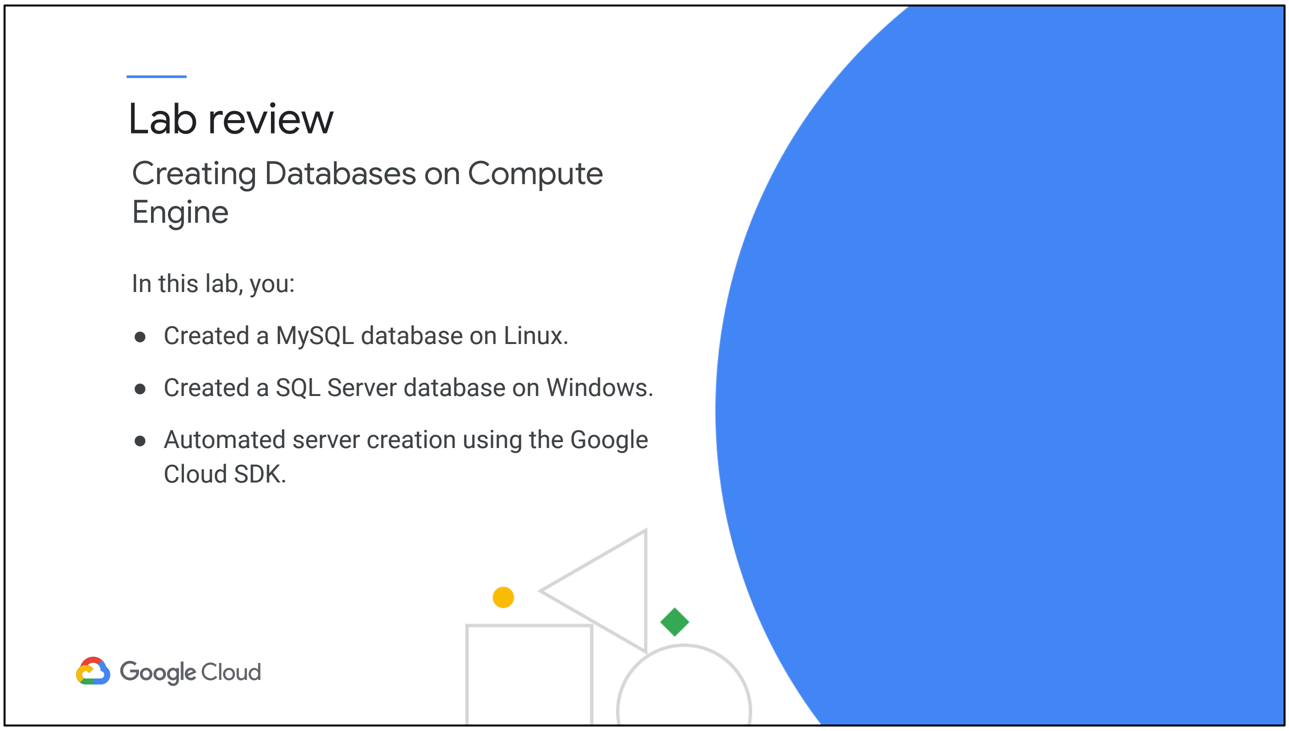
Lab intro

Creating Databases on Compute Engine

- Create a MySQL database on Linux.
- Create a SQL Server database on Windows.
- Automate server creation using the Google Cloud SDK.

Google Cloud

In this lab, you learn to create databases on Compute Engine. You create a MySQL database in Linux, a SQL Server database on Windows, and you use the Google Cloud SDK to automate the creation of your servers.



Lab review

Creating Databases on Compute Engine

In this lab, you:

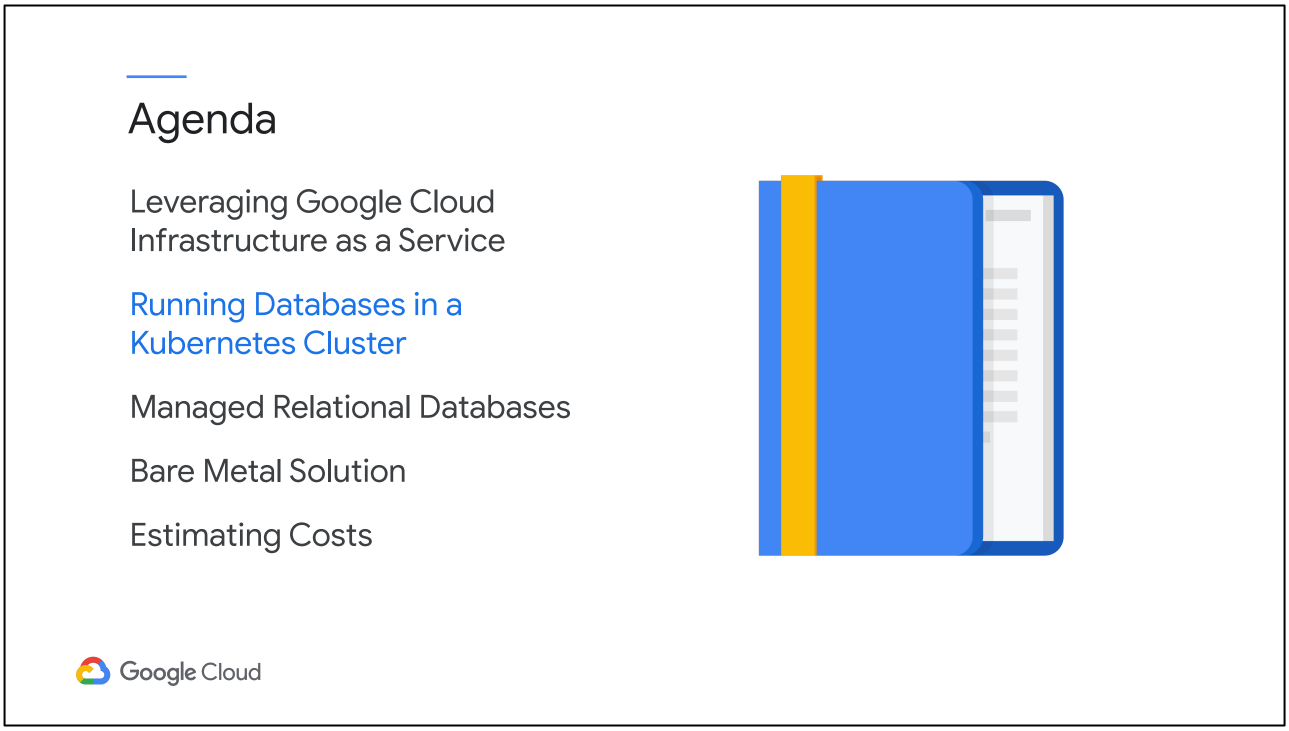
- Created a MySQL database on Linux.
- Created a SQL Server database on Windows.
- Automated server creation using the Google Cloud SDK.

Google Cloud

In this lab you: Created a MySQL database on Linux, Created a SQL Server database on Windows, and Automated server creation using the Google Cloud SDK.

The key takeaway is that there are almost no limitations when running databases using Compute Engine. You can configure your databases just as you would if you were running them in your own datacenter. The exception is Oracle. Oracle does not certify its database to run on virtual machines in Google Cloud. Later, you will learn how to use the Bare Metal Solution to run Oracle databases on Google Cloud.

In the next section, you will learn to run databases in a Kubernetes cluster.



Agenda

Leveraging Google Cloud Infrastructure as a Service

Running Databases in a Kubernetes Cluster

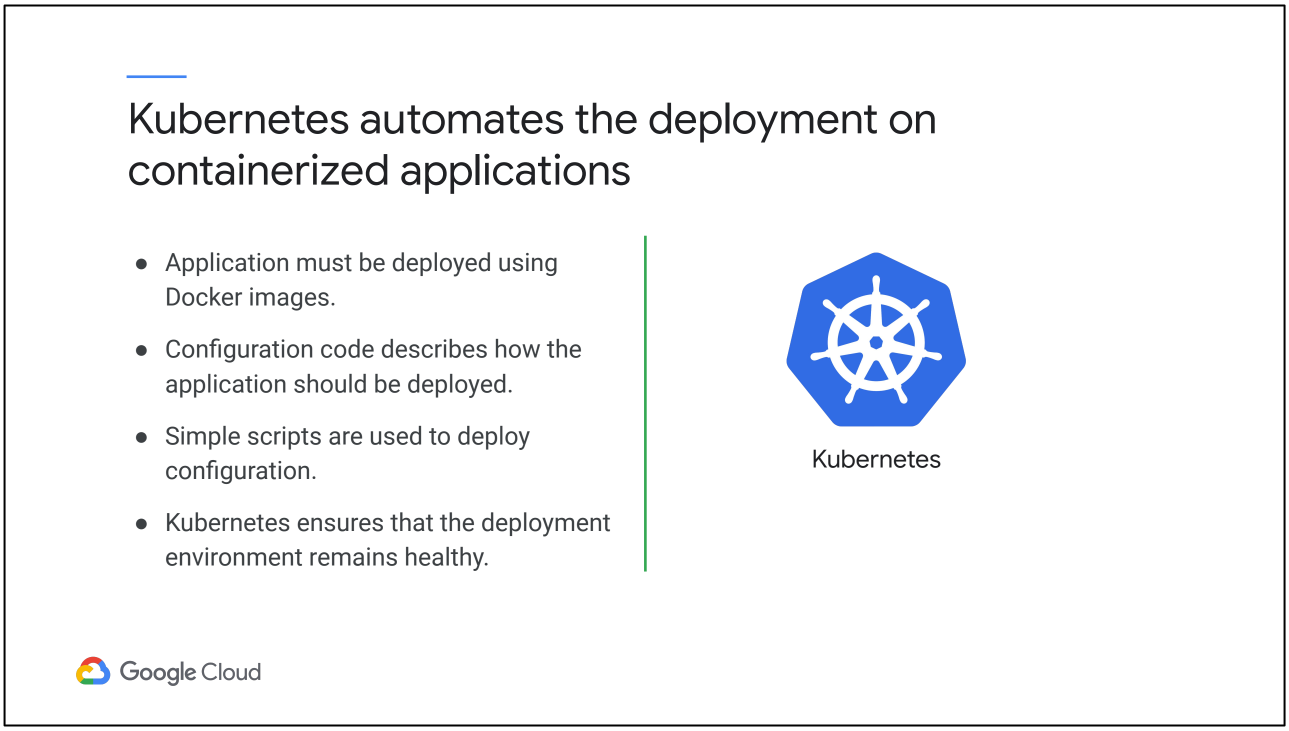
Managed Relational Databases

Bare Metal Solution

Estimating Costs

Google Cloud

Kubernetes is an open-source, cross-platform framework for running applications inside a cluster of shared resources. In this section, you learn to create a Kubernetes cluster on Google Cloud and configure a database to run inside that cluster.



Kubernetes automates the deployment on containerized applications

- Application must be deployed using Docker images.
- Configuration code describes how the application should be deployed.
- Simple scripts are used to deploy configuration.
- Kubernetes ensures that the deployment environment remains healthy.

Kubernetes

Google Cloud

Kubernetes automates the deployment of applications running in containers.

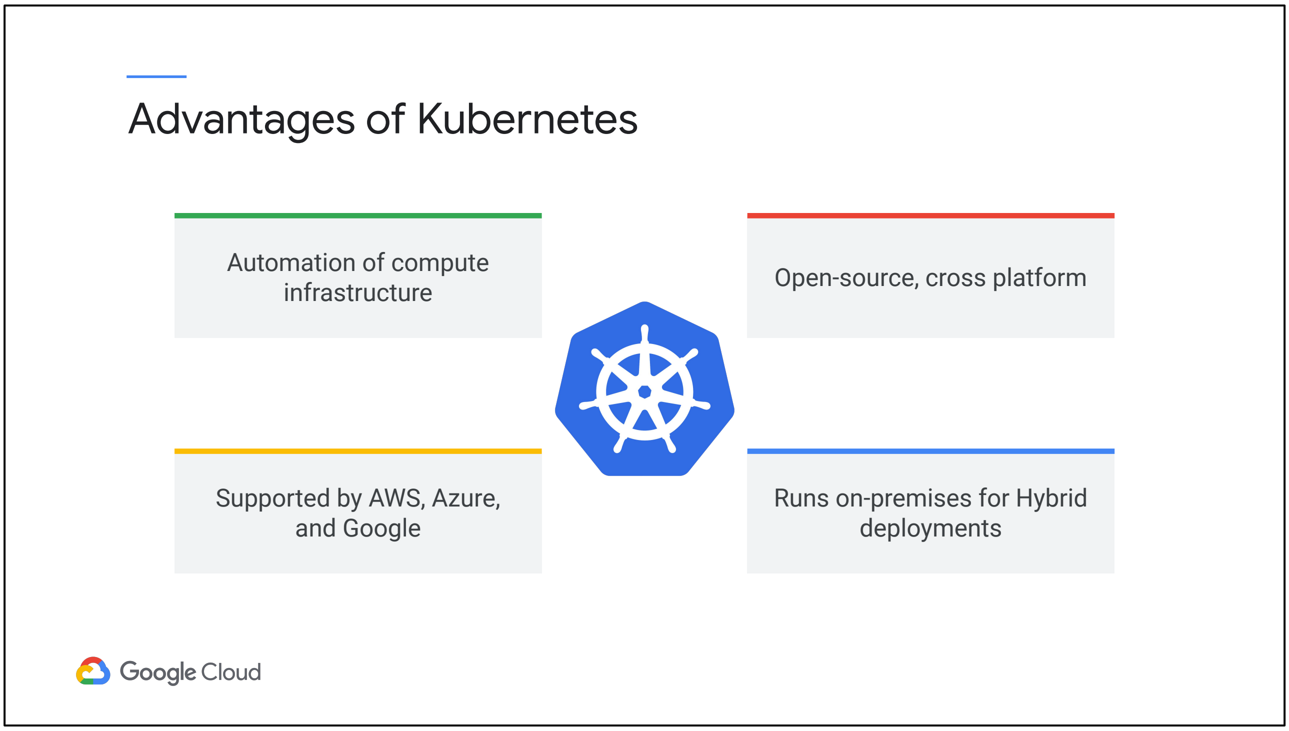
To use Kubernetes, you must first package your application inside a Docker image. In the case of a database, you can probably find a Docker image with the database software already configured. These images will be provided by the vendor or another third party and will be available for download over the internet.

After you have the Docker image, you need a configuration file, in YAML format, that describes how you want the image to be deployed in the cluster. This is perhaps the hardest part of learning Kubernetes. Luckily, configuration is standard, and you will find completed examples that you can adapt for your specific workloads.

After you have the configuration code, you can use a few simple CLI commands to deploy, manage, and delete your deployments.

You use the CLI to upload your configuration to the cluster. The configuration describes what you want, and Kubernetes ensures that this is what you get. If your deployment becomes out of sync with the configuration, Kubernetes fixes it. If the configuration is changed, Kubernetes changes the deployment to match the new configuration.

After you deploy your applications, Kubernetes ensures that they remain healthy. If they become unhealthy, Kubernetes fixes them.



Advantages of Kubernetes

Automation of compute infrastructure

Open-source, cross platform

Supported by AWS, Azure, and Google

Runs on-premises for Hybrid deployments

Google Cloud

People new to Kubernetes, and even people who use Kubernetes, might ask why you would run a database in a Kubernetes cluster. There are many advantages to Kubernetes.

- Automation may be the most important reason to use Kubernetes. It is important for deployments to be reproducible and testable. After the initial configuration is complete, removing, updating, and deleting deployments can be done with simple scripts. This allows you to move your deployments to different environments, like development, testing and production, quickly and easily.

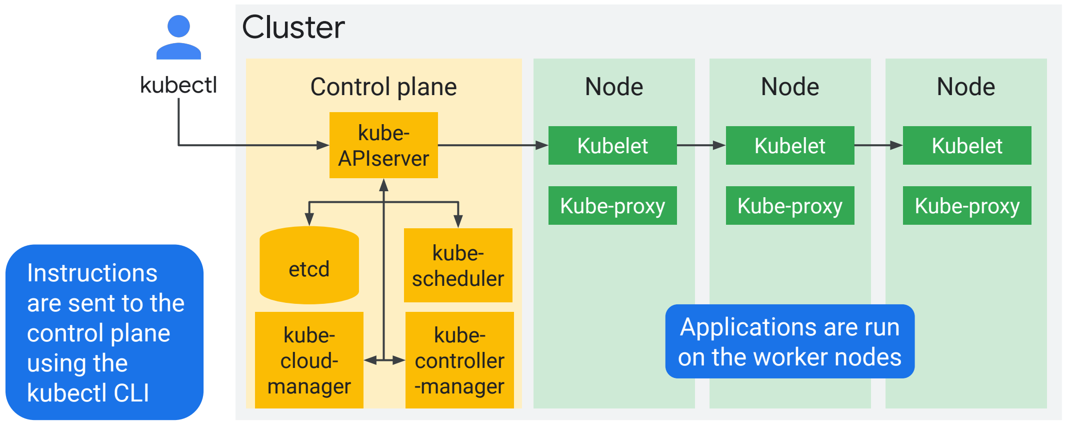
- Kubernetes is also open-source and cross-platform. Kubernetes is an extremely popular open-source project managed by the Cloud Native Foundation with many contributors. It is supported by Linux and Windows, and can be used in many different public and private clouds.

- Kubernetes is supported by all the major cloud providers. AWS has a service called EKS (Elastic Kubernetes Service). Microsoft has Azure Kubernetes service (AKS), and Google has GKE (Google Kubernetes Engine). Kubernetes is also supported by private clouds like Red Hat's Openshift, and is supported by automation tools like Pivotal Cloud Foundry.

- Kubernetes clusters can also be run on-premises. You can install Kubernetes yourself on a cluster of machines running in your own data center. Kubernetes

- configurations and scripts are also the same regardless of where the cluster is running; this allows you to have one tool and one set of scripts and deploy your software to any environment. Many customers want this type of hybrid deployment scenario provided by Kubernetes. You can configure the software once and deploy it to any cloud or even on-premises with little or no change to the configuration.

Kubernetes clusters consist of control plane and worker nodes



Google Cloud

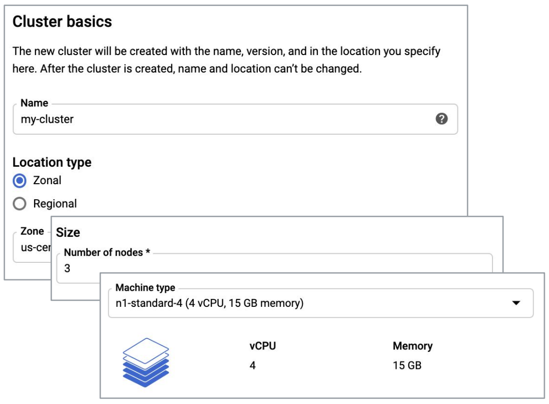
To use Kubernetes, you first need a Cluster. A cluster requires one or more control planes and some number of worker nodes.

Commands and configuration are sent to the control plane. The control plane stores the configuration and deploys the applications to the worker nodes. The control plane then monitors the applications to ensure that they are running as expected. If something fails, the control plane fixes it.

When you use Google Kubernetes Engine, Google provides the control plane as a service. They ensure that the control plane is highly available and scalable.

Google Kubernetes Engine (GKE) automates the creation of Kubernetes clusters

- Specify cluster location, number of nodes, and the size of each node.
- Nodes are really just Compute Engine virtual machines with Kubernetes installed.
- The control plane is provided automatically by Google.



Cluster basics

The new cluster will be created with the name, version, and in the location you specify here. After the cluster is created, name and location can't be changed.

Name: my-cluster

Location type: [✓ Zonal] Regional

Zone: us-cer

Size

Number of nodes *: 3

Machine type: n1-standard-4 (4 vCPU, 15 GB memory)

vCPU	Memory
4	15 GB

Google Cloud

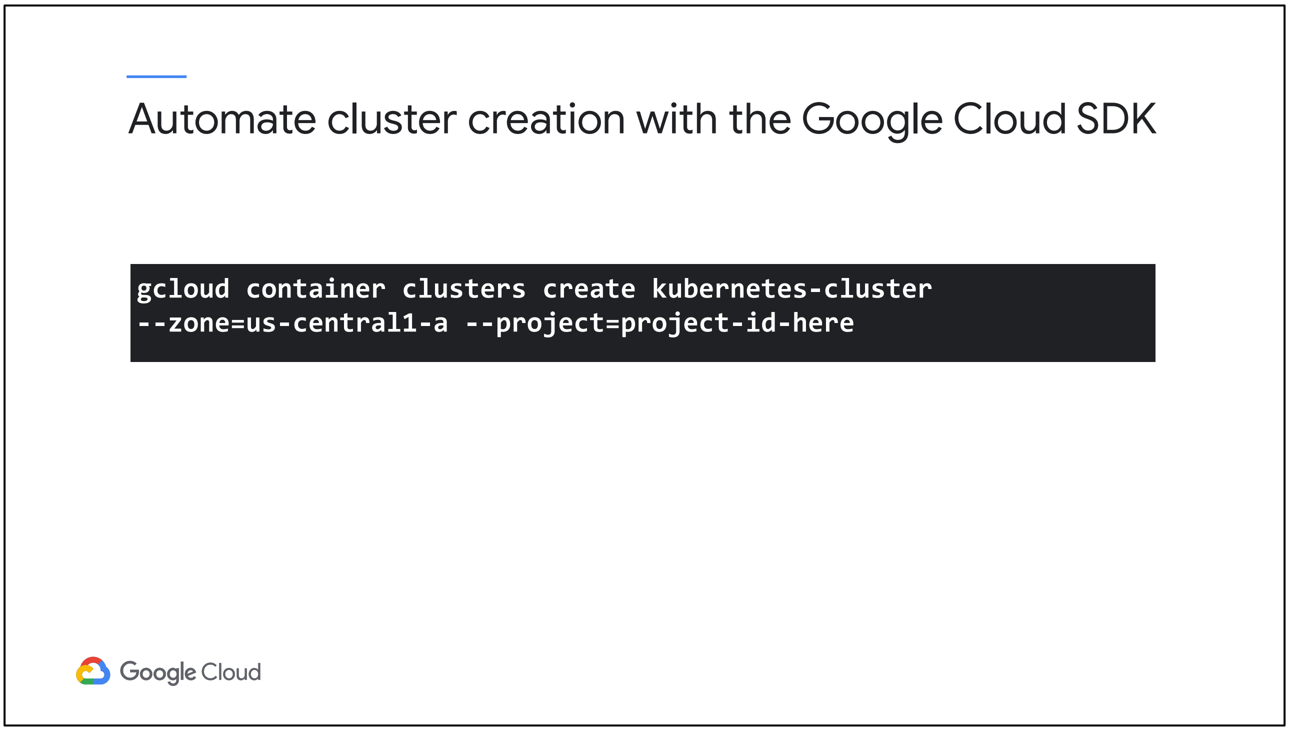
Use Google Kubernetes Engine to create clusters on Google Cloud.

You specify the location, number of nodes, and the size of each node.

The nodes themselves are really just preconfigured virtual machines running in Compute Engine and managed by the GKE service.

The control plane node is automatically provided by Google; you just configure the workers.

There are many additional options you can set. The details are really beyond the scope of this course though.



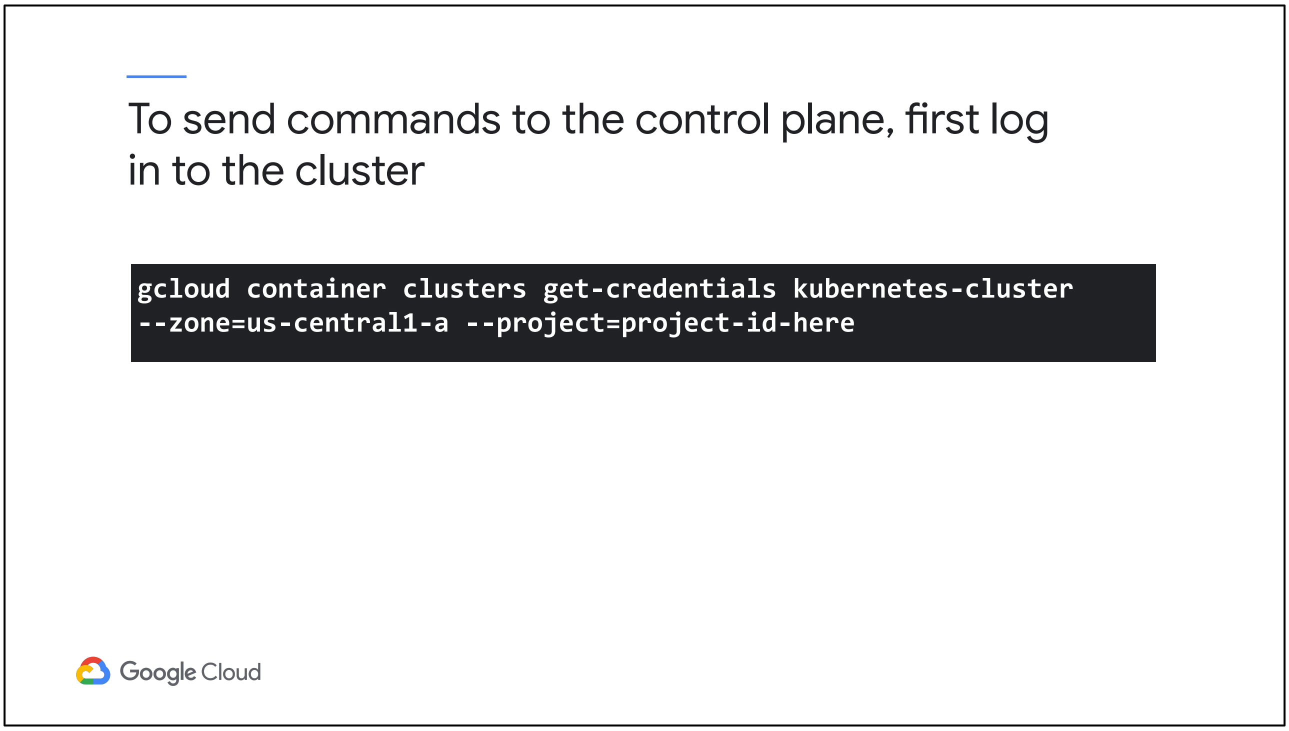
Automate cluster creation with the Google Cloud SDK

```
gcloud container clusters create kubernetes-cluster
--zone=us-central1-a --project=project-id-here
```

Google Cloud

Like all other Google Cloud resources, you can easily script the creation of the cluster using the Google Cloud SDK and the CLI.

The commands for creating a cluster are simple: "gcloud container clusters create," followed by the name of the cluster and parameters. A simple example is shown on this slide. Note, there are defaults for almost all parameters. So, in the command here, the zone and Google Cloud project are specified. For everything else, machine size, number of nodes, security settings, etc., the defaults are being used.



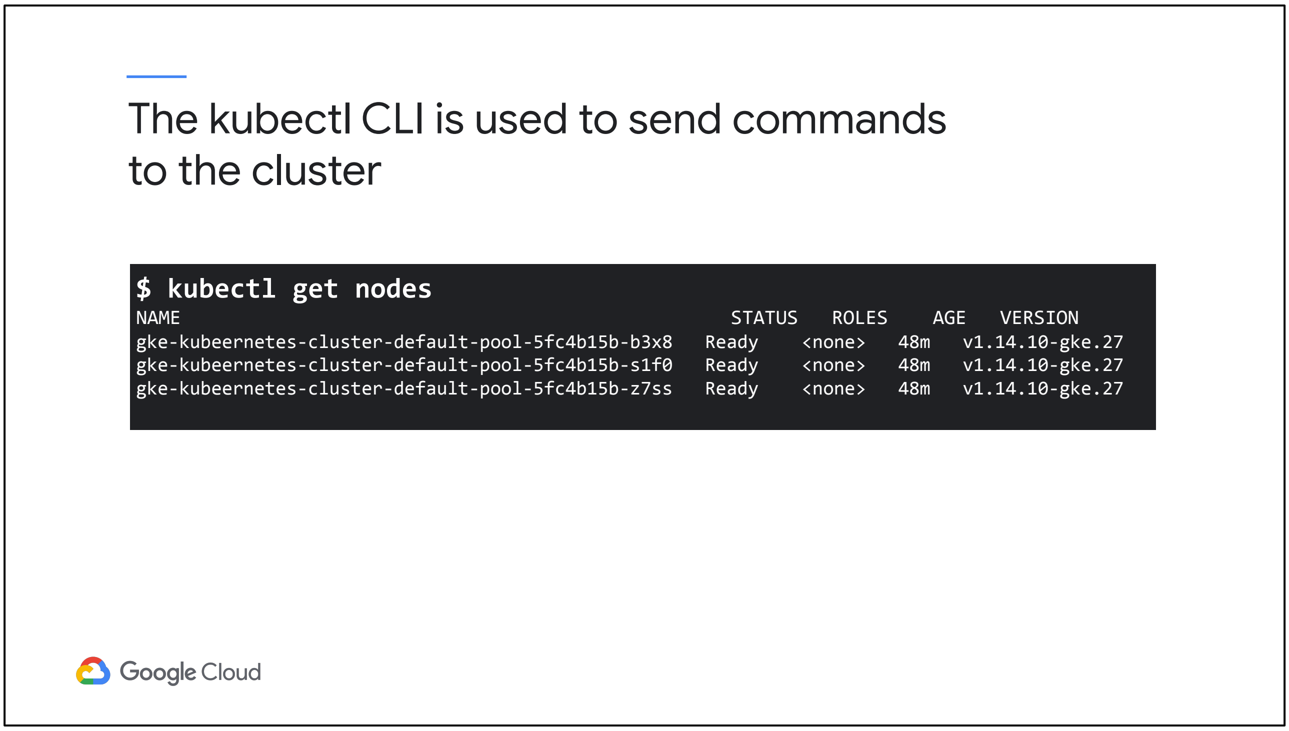
To send commands to the control plane, first log in to the cluster

```
gcloud container clusters get-credentials kubernetes-cluster
--zone=us-central1-a --project=project-id-here
```

Google Cloud

After your cluster is created, you have to connect to it in order to send commands to it. Notice that the command here is the same as the previous command, but instead of "create," we are calling "get-credentials."

You could have multiple clusters, so you have to connect to the right one before you start sending commands to it.



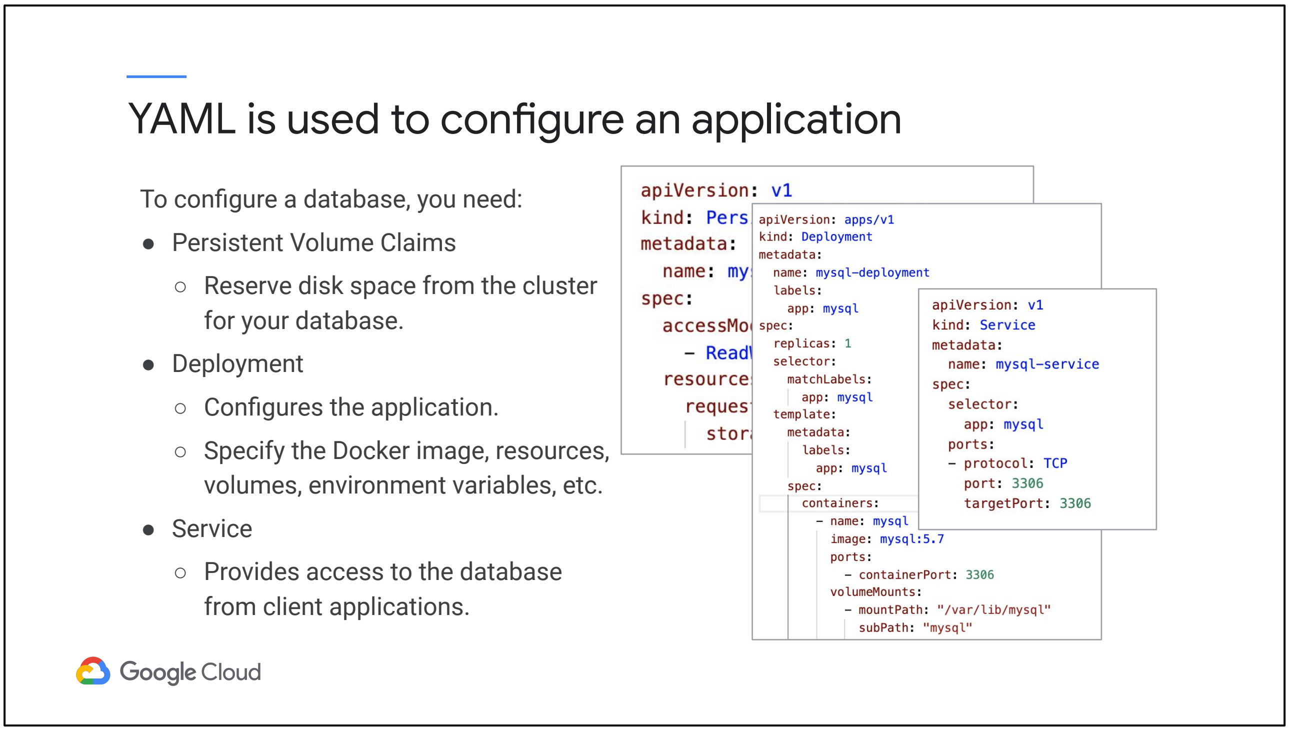
The kubectl CLI is used to send commands to the cluster

```
$ kubectl get nodes
NAME                                                 STATUS    ROLES    AGE    VERSION
gke-kubeernetes-cluster-default-pool-5fc4b15b-b3x8   Ready     <none>   48m    v1.14.10-gke.27
gke-kubeernetes-cluster-default-pool-5fc4b15b-s1f0   Ready     <none>   48m    v1.14.10-gke.27
gke-kubeernetes-cluster-default-pool-5fc4b15b-z7ss   Ready     <none>   48m    v1.14.10-gke.27
```

Google Cloud

After you're connected to your cluster, you use the "kubectl" CLI to send commands to it. This CLI is included with the Google Cloud SDK and already installed in Google Cloud Shell.

You could also download this CLI from the Cloud Native Foundation, which is responsible for maintaining it.



YAML is used to configure an application

To configure a database, you need:

- Persistent Volume Claims
  - Reserve disk space from the cluster for your database.
- Deployment
  - Configures the application.
  - Specify the Docker image, resources, volumes, environment variables, etc.
- Service
  - Provides access to the database from client applications.

```
apiVersion: v1
kind: Pers
metadata:
  name: my
spec:
  accessMo
    - Read
  resource
    request
      stor
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql:5.7
          ports:
            - containerPort: 3306
          volumeMounts:
            - mountPath: "/var/lib/mysql"
              subPath: "mysql"
```

```
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
spec:
  selector:
    app: mysql
  ports:
  - protocol: TCP
    port: 3306
    targetPort: 3306
```

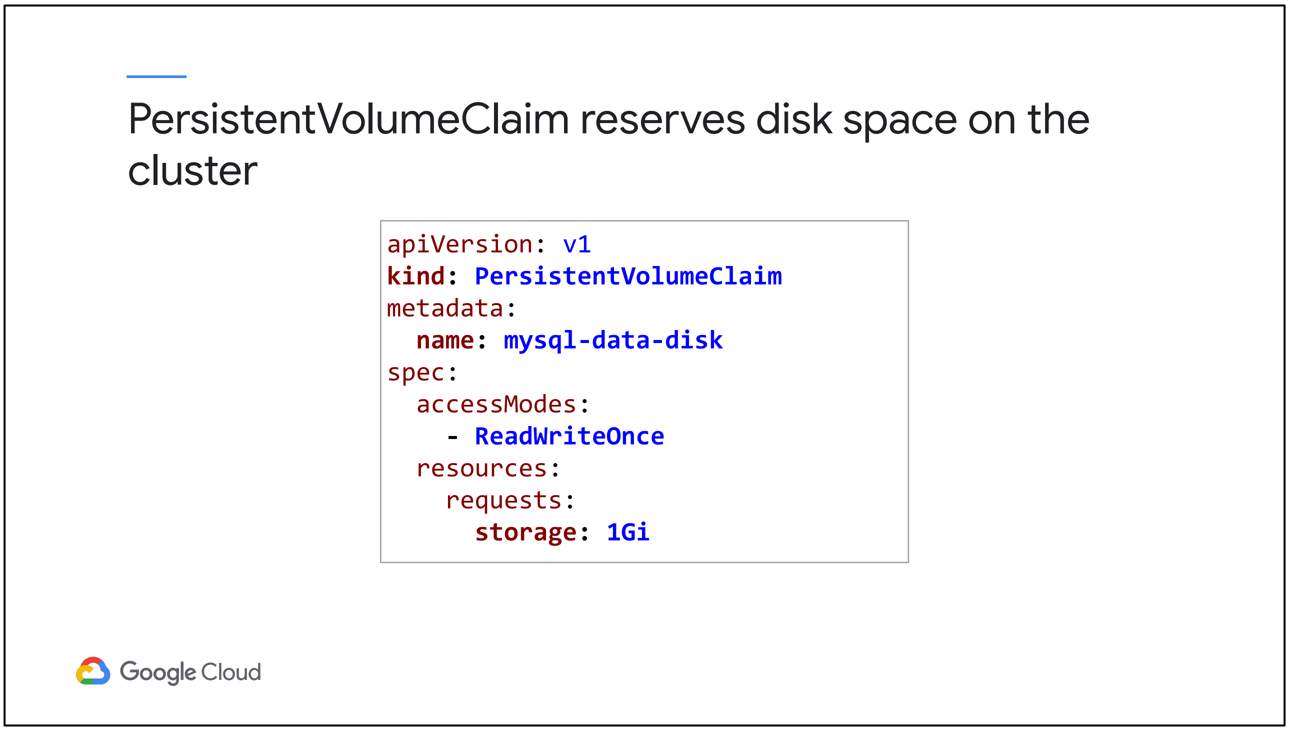
Google Cloud

YAML is used to configure an application to run in a Kubernetes cluster.

For a database, you need to configure a disk space. This is done with a persistent volume claim.

Then, you configure a deployment. The deployment, among other things, specifies what Docker image you want to run.

Lastly, you configure a service. The service provides access to the database from the client applications that need to connect to it.

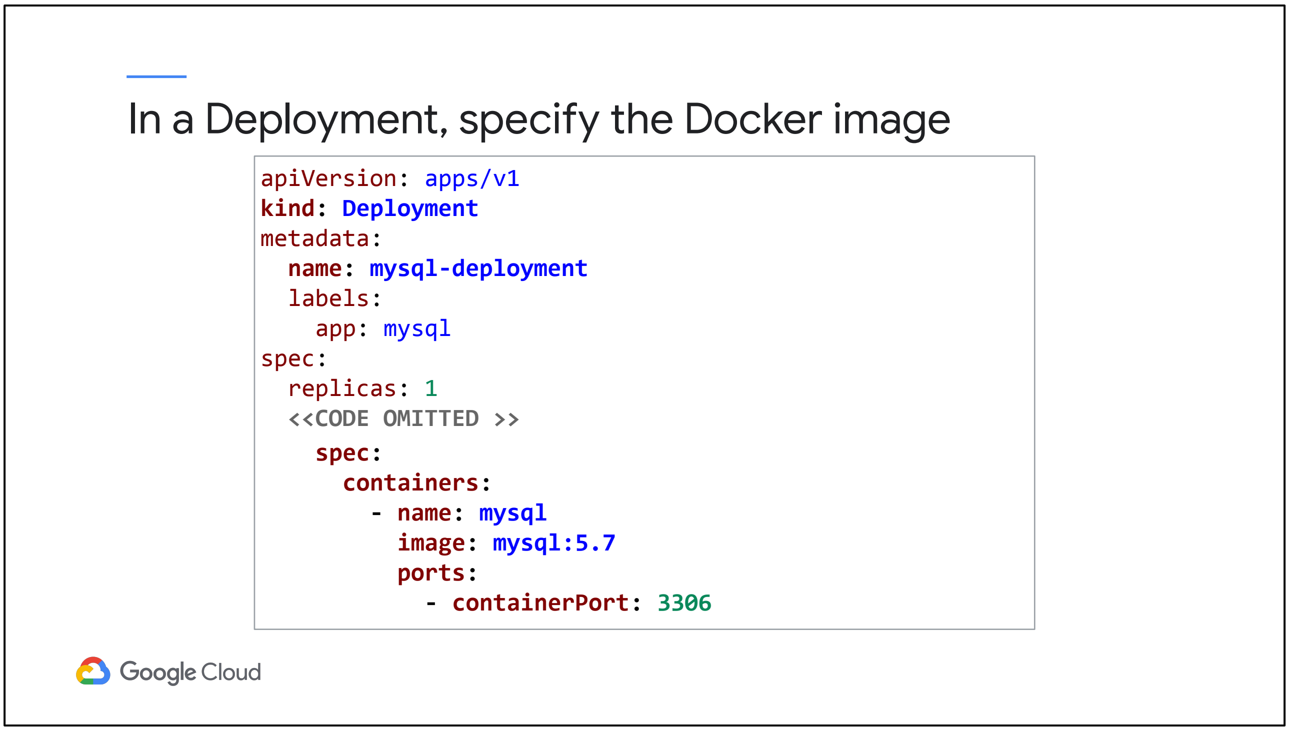


PersistentVolumeClaim reserves disk space on the cluster

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-data-disk
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Google Cloud

In the example shown here, a PersistentVolumeClaim is being configured. This reserves disk space on the cluster that will be used by your database. Note the name, "mysql-data-disk." The access mode makes it a read-write disk used by one deployment. In this case, just 1 GB of disk space is being allocated.

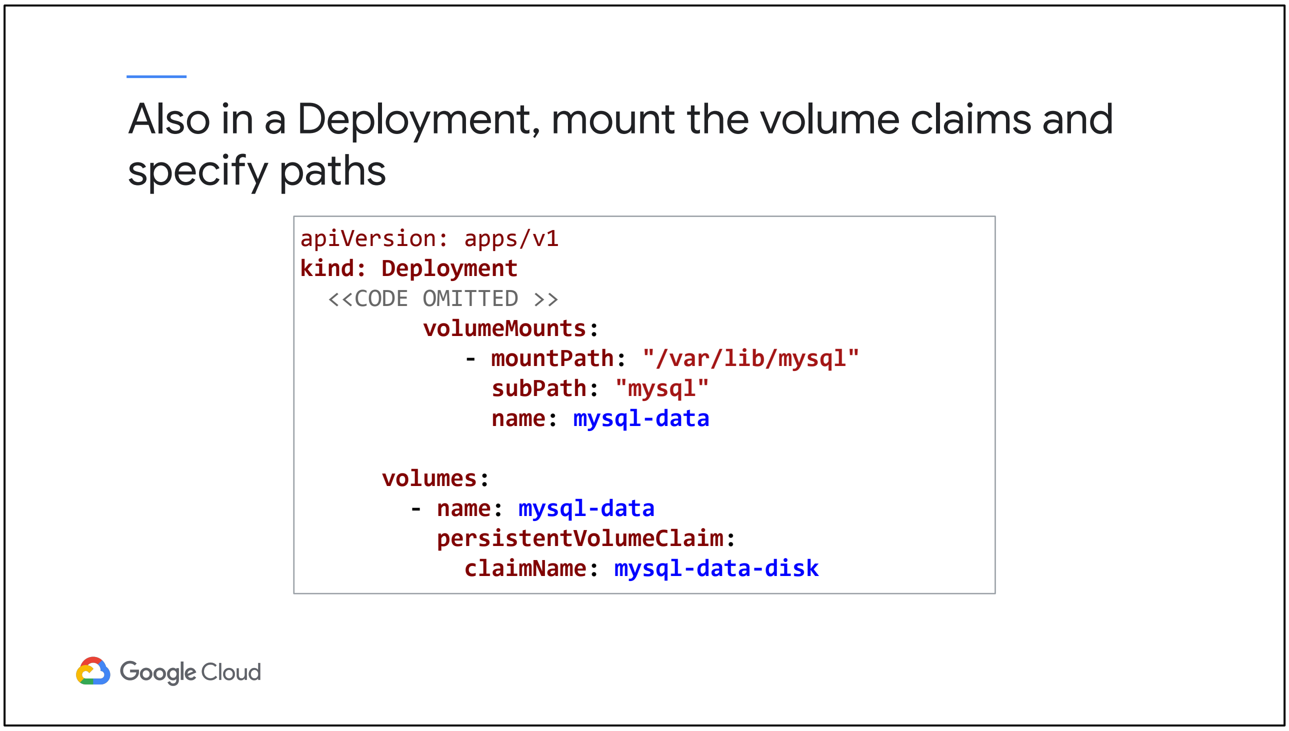


In a Deployment, specify the Docker image

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  labels:
    app: mysql
spec:
  replicas: 1
  <<CODE OMITTED >>
    spec:
      containers:
        - name: mysql
          image: mysql:5.7
          ports:
            - containerPort: 3306
```

Google Cloud

Shown here is a portion of the Deployment configuration. Notice the image property near the bottom. In this case, the MySQL version 5.7 Docker image will be downloaded and run in the cluster. Also, notice the containerPort property. This is the port that MySQL runs on by default, 3306.



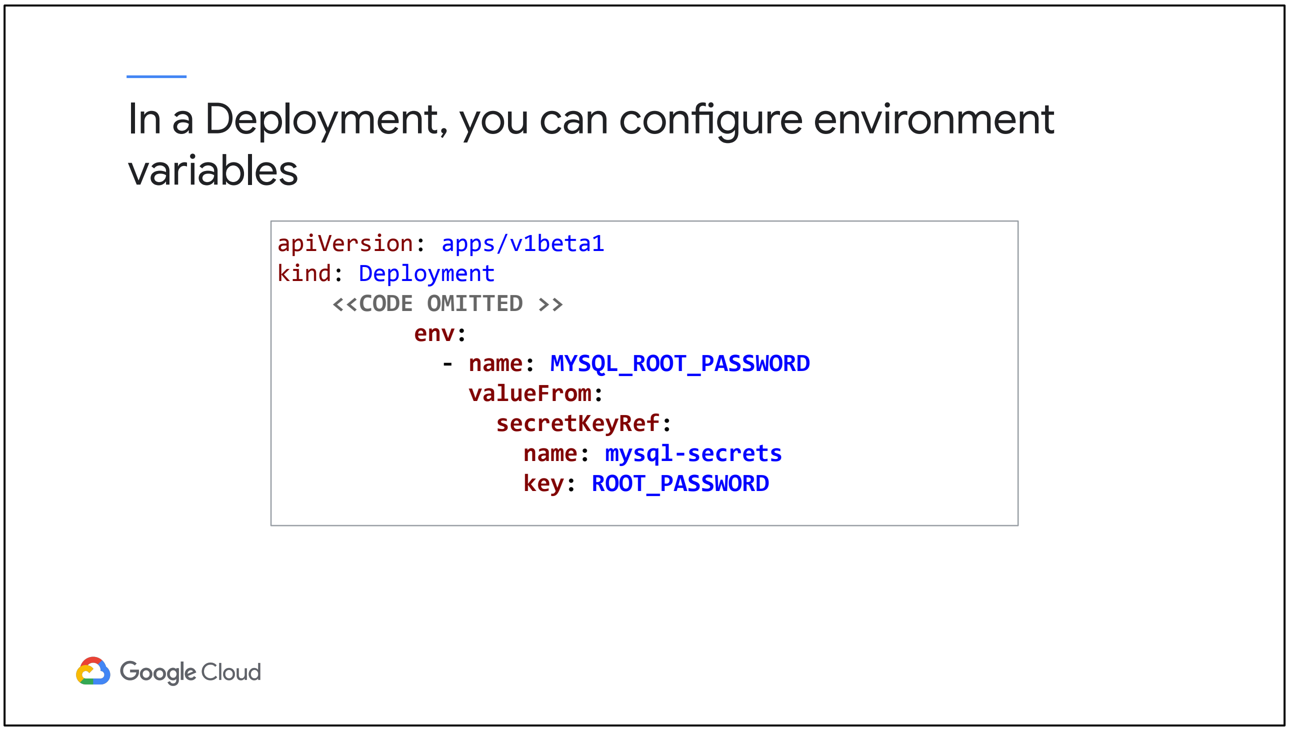
Also in a Deployment, mount the volume claims and specify paths

```yaml
apiVersion: apps/v1
kind: Deployment
  <<CODE OMITTED >>
         volumeMounts:
            - mountPath: "/var/lib/mysql"
              subPath: "mysql"
              name: mysql-data

      volumes:
        - name: mysql-data
          persistentVolumeClaim:
            claimName: mysql-data-disk
```

Google Cloud

Shown here is more of the Deployment configuration. Notice that in the Volumes section, the persistentVolumeClaim, "mysql-data-disk," that you saw a minute ago, is being attached as a volume called "mysql-data." Then above that, that volume is being mounted. The mountPath, "/var/lib/mysql," is where your MySQL database will write its data.

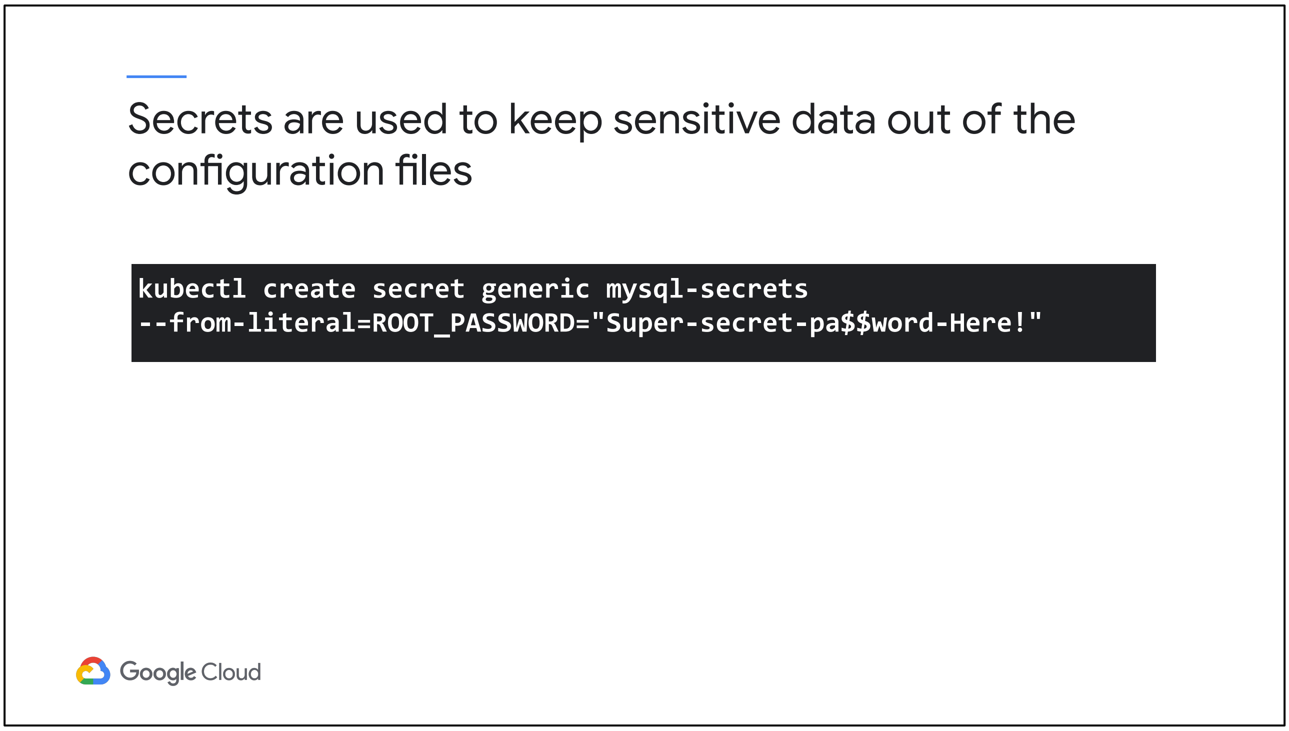


In a Deployment, you can configure environment variables

```yaml
apiVersion: apps/v1beta1
kind: Deployment
    <<CODE OMITTED >>
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql-secrets
                  key: ROOT_PASSWORD
```

Google Cloud

Here is yet more of the Deployment configuration. Notice that an environment variable is being created for the password for the MySQL root user. The value of the password is stored separately as a Kubernetes secret.



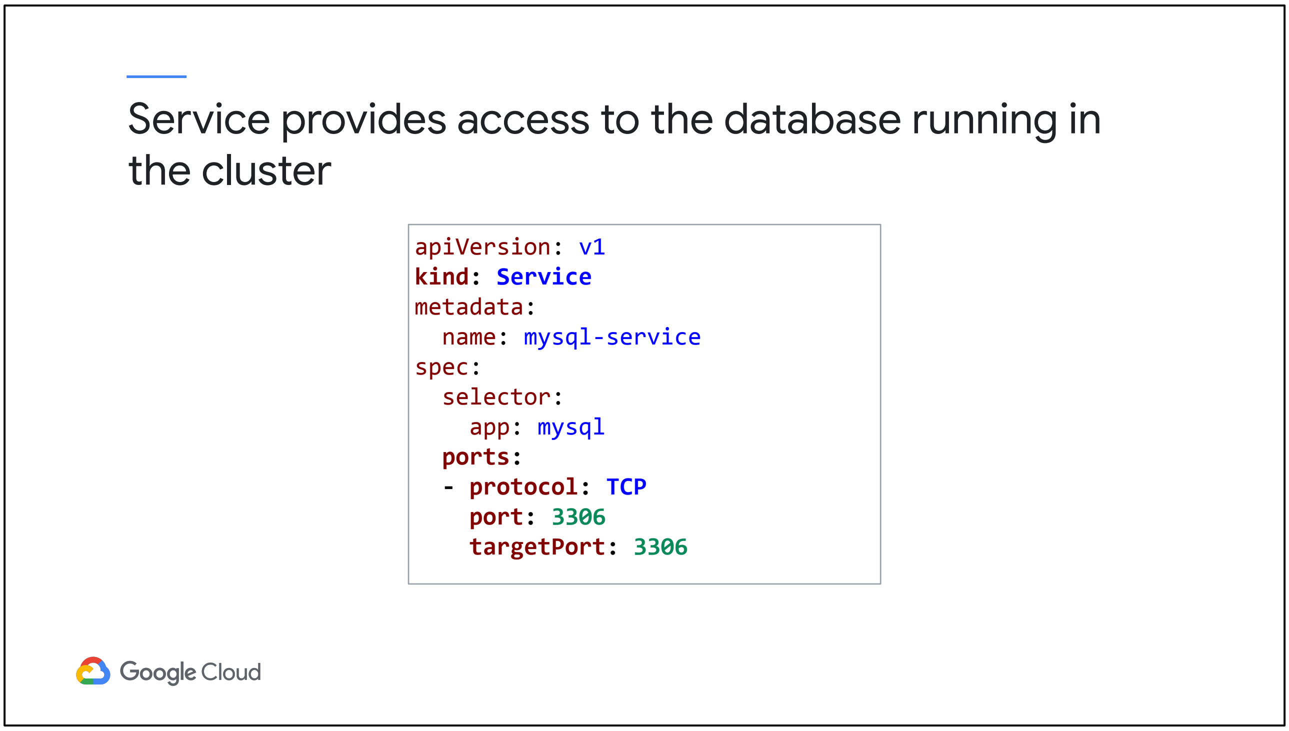
Secrets are used to keep sensitive data out of the configuration files

```
kubectl create secret generic mysql-secrets
--from-literal=ROOT_PASSWORD="Super-secret-pa$$word-Here!"
```

Google Cloud

Here the secret is being set for the root user password.

It is a best practice to keep sensitive data out of the configuration files. That's why the password is set separately.



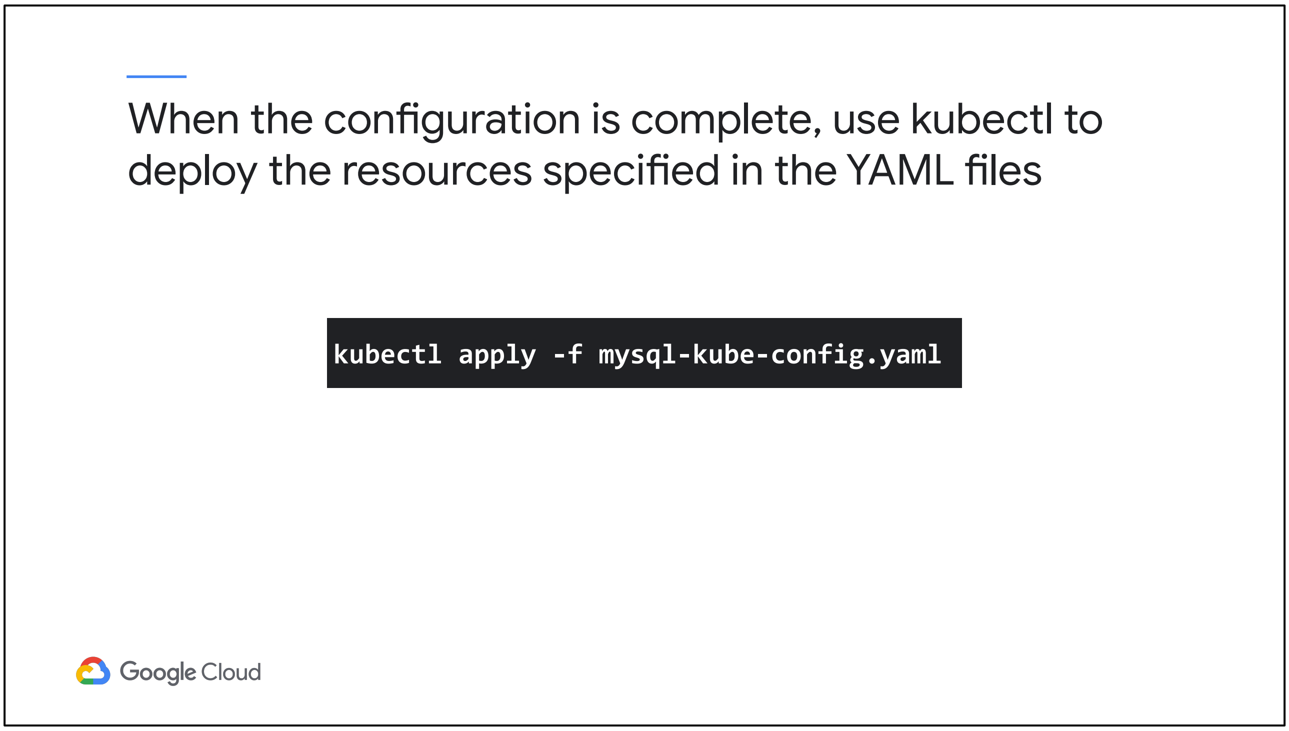
Service provides access to the database running in the cluster

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
spec:
  selector:
    app: mysql
  ports:
  - protocol: TCP
    port: 3306
    targetPort: 3306
```

Google Cloud

At this point, you've seen the configuration for the deployment and the volume. The last thing to configure is a service. Services provide access to your programs that are running in the cluster. Clients don't connect directly to the application in the cluster: they go through the service.

Each service is assigned an IP address. Requests are made to the service, which proxies the request to the deployed application; in this case, that application will be the MySQL database configured previously.



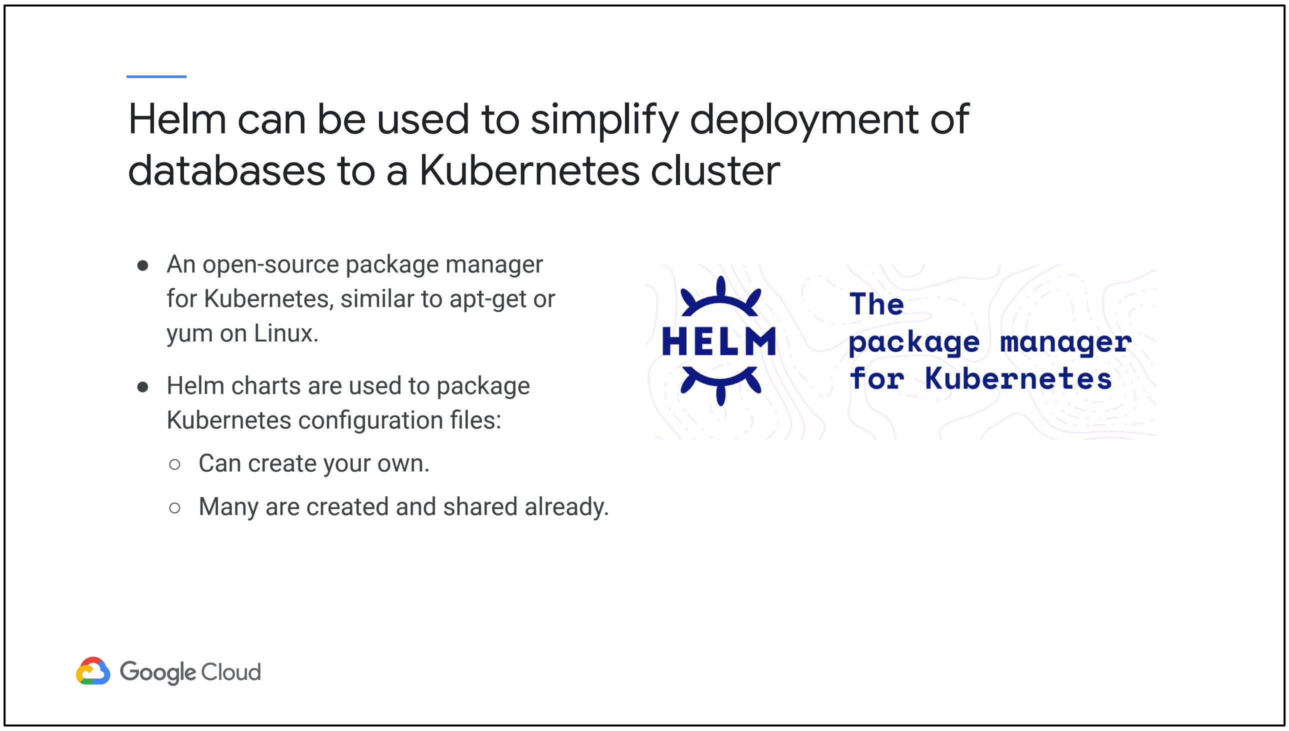
When the configuration is complete, use kubectl to deploy the resources specified in the YAML files

```
kubectl apply -f mysql-kube-config.yaml
```

Google Cloud

You can store the configuration code in one or more files. When you're ready to deploy the application, use the command "kubectl apply" and specify the configuration file as shown here.

The same line with "delete" instead of "apply" would remove everything that was configured from the cluster.



Helm can be used to simplify deployment of databases to a Kubernetes cluster

- An open-source package manager for Kubernetes, similar to apt-get or yum on Linux.
- Helm charts are used to package Kubernetes configuration files:
  - Can create your own.
  - Many are created and shared already.

HELM

The package manager for Kubernetes

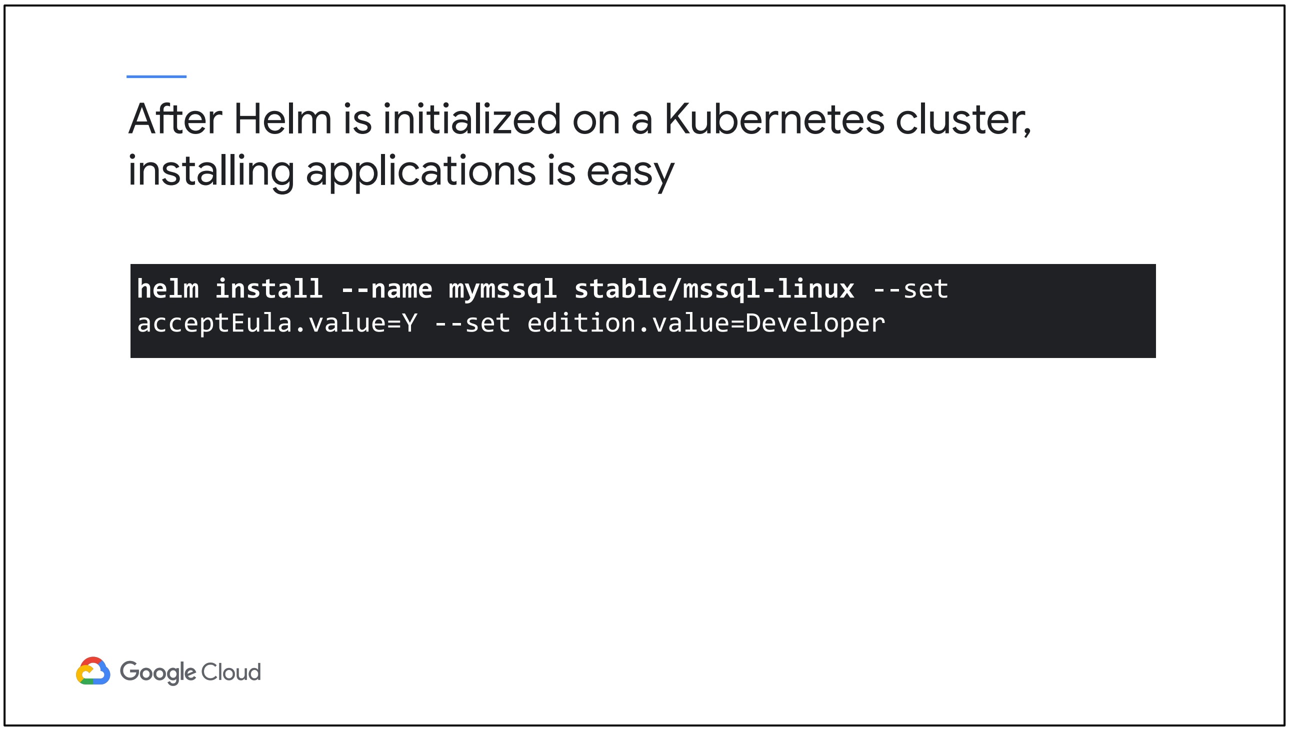
Google Cloud

If you are new to Kubernetes, you might be overwhelmed by all the configuration. I know I was when I was first learning it. Fortunately, there is an easier and arguably better way to deploy a database into a cluster.

Helm is an open-source package manager for Kubernetes, similar to apt-get in Linux or Chocolatey on Windows. You can configure Helm to run on your cluster, then automate deployments using Helm Charts.

Helm Charts provide preconfigured deployments for hundreds of applications, including many databases like MySQL, SQL Server, and PostgreSQL.

A huge community provides the Charts. You could also create your own Charts.



After Helm is initialized on a Kubernetes cluster, installing applications is easy

```
helm install --name mymssql stable/mssql-linux --set
acceptEula.value=Y --set edition.value=Developer
```

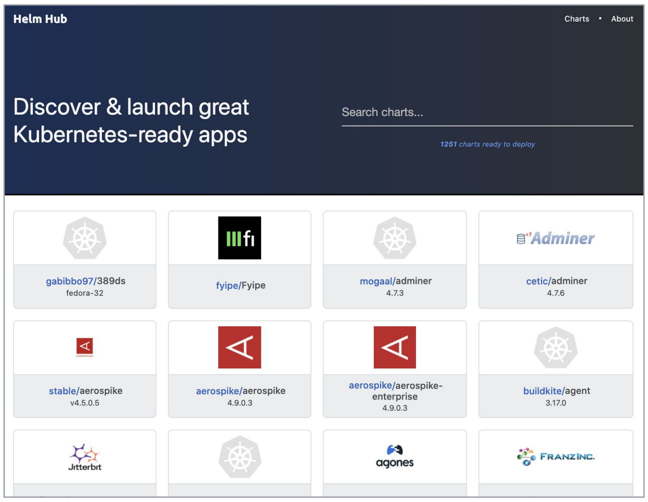
Google Cloud

You first run "helm init" to initialize Helm on a Kubernetes Cluster. Helm is already installed in Google Cloud Shell, so if you're using it, you don't even have to install Helm itself.

After Helm is initialized, you simply use the "helm install" command while specifying the name of the Chart and any parameters that are required.

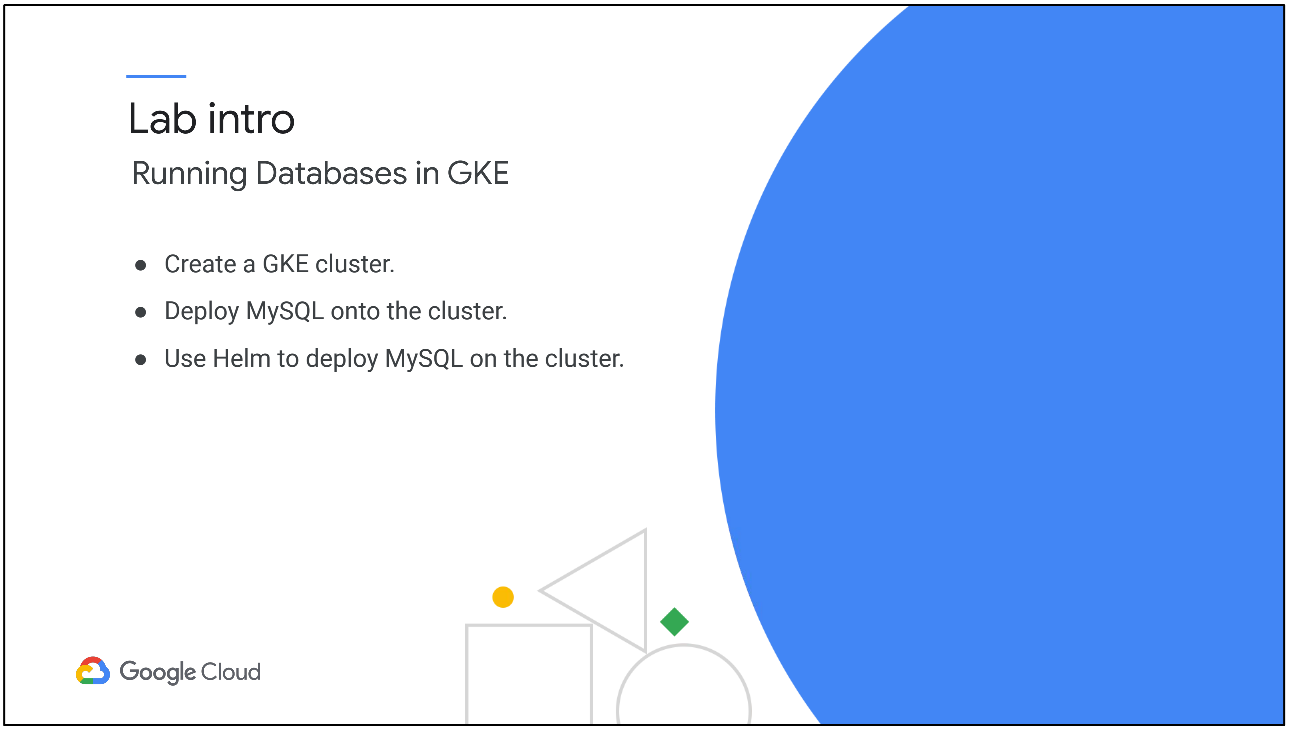
In the example above, MySQL is being deployed. This eliminates the need for you to write your configuration files.

Search for Helm charts at Helm Hub



Google Cloud

You can search for Charts on Helm Hub website.



Lab intro

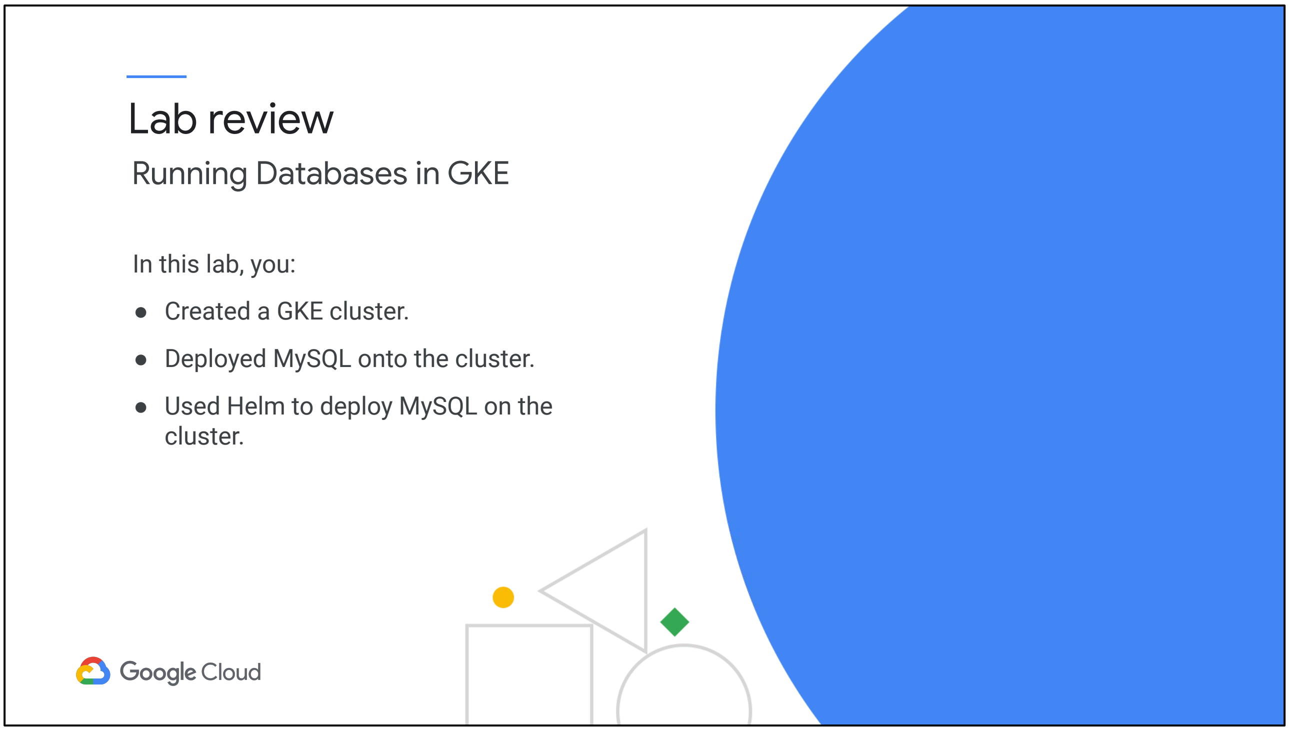
Running Databases in GKE

- Create a GKE cluster.
- Deploy MySQL onto the cluster.
- Use Helm to deploy MySQL on the cluster.

Google Cloud

In this lab, you will run databases in a Google Kubernetes Engine cluster. First, you will need to create the cluster. Then, you will configure and deploy MySQL into the cluster.

After you do the configuration manually, you see how to automate the deployment using Helm.



Lab review

Running Databases in GKE

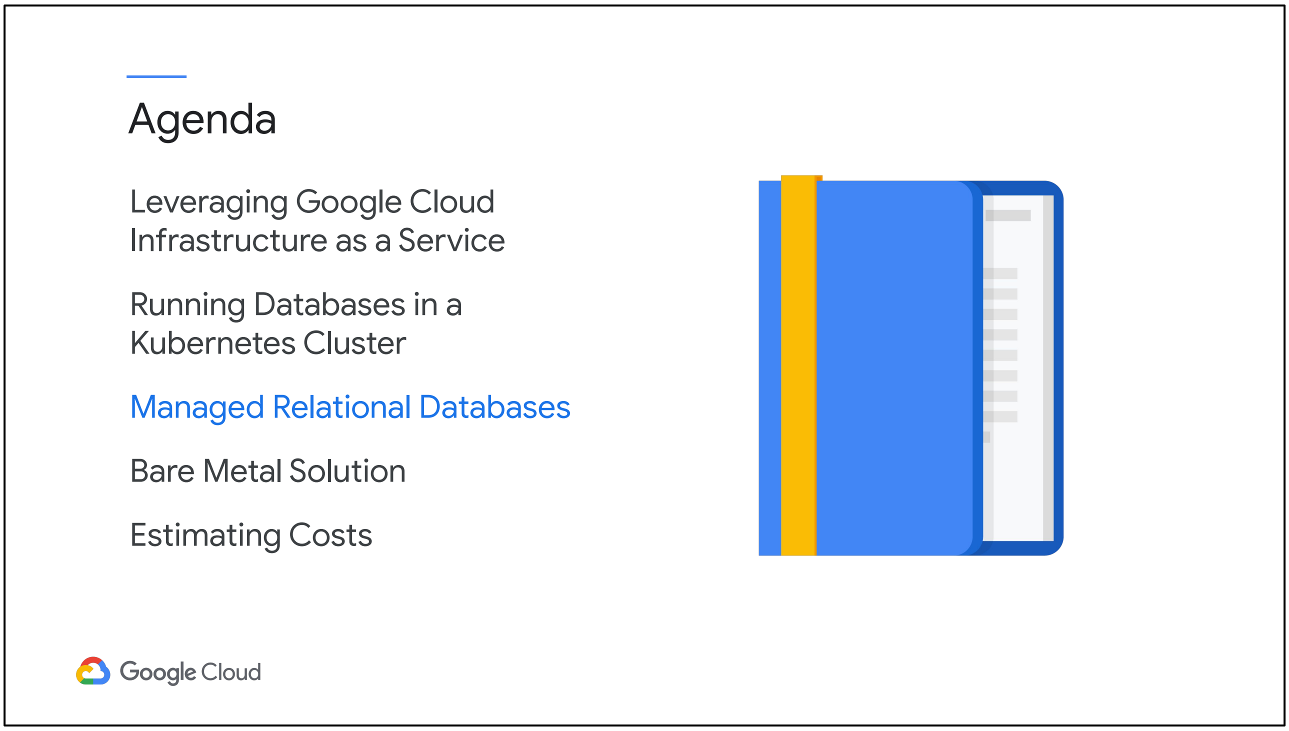
In this lab, you:

- Created a GKE cluster.
- Deployed MySQL onto the cluster.
- Used Helm to deploy MySQL on the cluster.

Google Cloud

In this lab, you created a GKE cluster, deployed MySQL onto the cluster, and used Helm to deploy MySQL on the cluster.

There are many advantages of using Kubernetes. These included, automated resource creation, application deployment and health monitoring. Kubernetes is also ideal if you want a cross-cloud or hybrid cloud environment. In this lab you saw how to easily run databases in a Kubernetes cluster. Next, you will learn to use Google managed database services.



Agenda

Leveraging Google Cloud Infrastructure as a Service

Running Databases in a Kubernetes Cluster

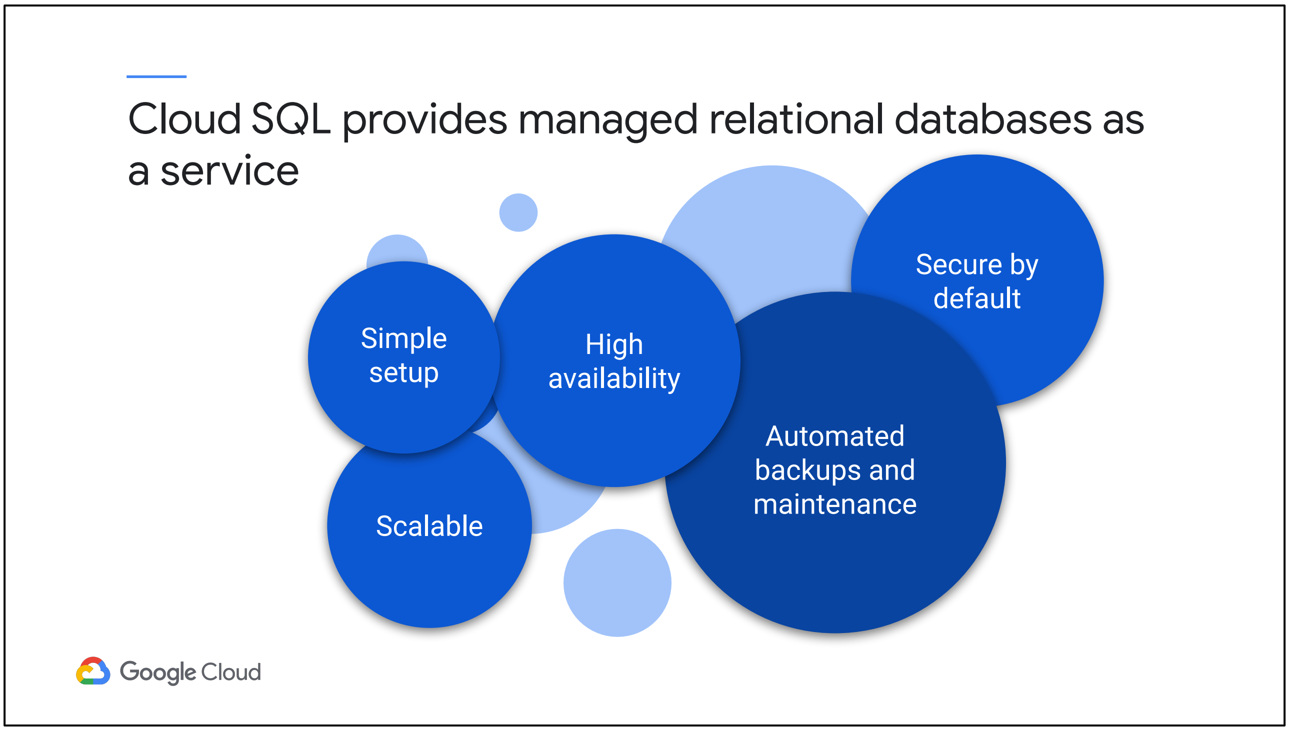
Managed Relational Databases

Bare Metal Solution

Estimating Costs

Google Cloud

Google Cloud provides a couple of managed database-as-a-service products that can simplify and automate the creation and administration of your relational databases.



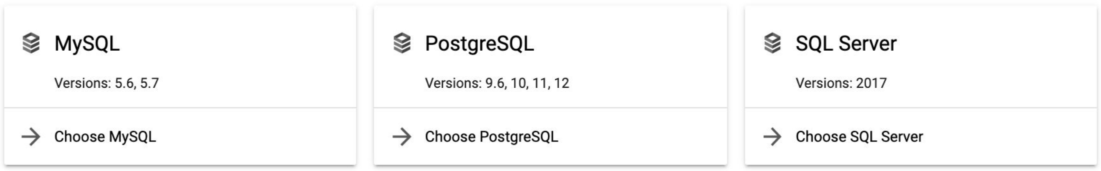
Cloud SQL provides managed relational databases as a service

Simple setup
High availability
Secure by default
Automated backups and maintenance
Scalable
Google Cloud

Cloud SQL is the first managed relational database service. Cloud SQL databases are easy to set up, either in the console or with a command line command.

- Cloud SQL databases are scalable up to 96 cores and about 30 TB of disk space.
- Setup is simple and can be done from the command line or console.
- You can configure Cloud SQL to automatically create a failover database in another zone.
- Backups and maintenance tasks, like installing patches, are automatically taken care of for you.
- Cloud SQL databases are also secure by default. A firewall allows access to the database only from the project where the database is created. You can add clients from other networks as required for your specific use case.

Cloud SQL supports MySQL, PostgreSQL, and SQL Server


MySQL
Versions: 5.6, 5.7
Choose MySQL

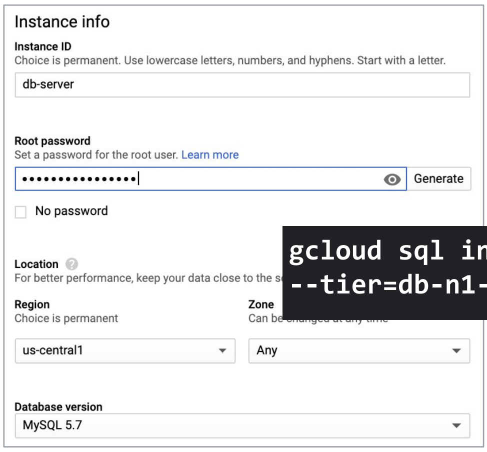
PostgreSQL
Versions: 9.6, 10, 11, 12
Choose PostgreSQL

SQL Server
Versions: 2017
Choose SQL Server

Cloud SQL supports MySQL, PostgreSQL, and SQL Server. You choose your preferred database when creating a Cloud SQL instance. Support for various features varies slightly depending on the database you choose. If you choose SQL Server, you will also be charged for the license.

See the product documentation for more details.

Create the database in the Console or use the CLI



Instance info

Instance ID
Choice is permanent. Use lowercase letters, numbers, and hyphens. Start with a letter.
db-server

Root password
Set a password for the root user. Learn more
••••••••••••••••

Location
For better performance, keep your data close to the s

Region
Choice is permanent
us-central1

Zone
Can be changed at any time
Any

Database version
MySQL 5.7

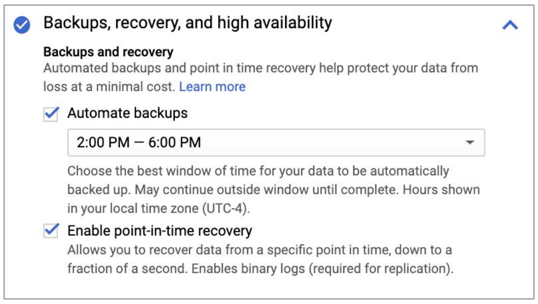
```
gcloud sql instances create db-server
--tier=db-n1-standard-2 --region=us-central1
```

Google Cloud

When creating the database, you can use the Google Cloud Console or a simple CLI command as shown here. See the documentation for a complete list of parameters for your preferred database.

Schedule backups as needed

- Backups are done on a schedule you choose.
  - Note: Cloud SQL backups can only be restored to another Cloud SQL instance.
- Point-in-time recovery allows you to restore a database to any time before the database became corrupted.



Backups, recovery, and high availability

Backups and recovery

Automated backups and point in time recovery help protect your data from loss at a minimal cost. Learn more

✓ Automate backups

2:00 PM — 6:00 PM

Choose the best window of time for your data to be automatically backed up. May continue outside window until complete. Hours shown in your local time zone (UTC-4).

✓ Enable point-in-time recovery

Allows you to recover data from a specific point in time, down to a fraction of a second. Enables binary logs (required for replication).

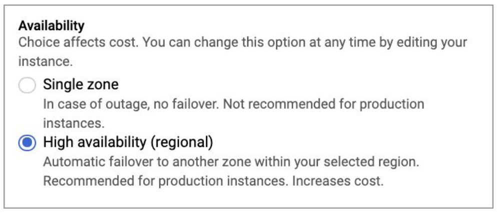
Google Cloud

When configuring a database, you can enable automatic backups and specify when you want them to run. Backups are done by creating snapshots of the persistent disk being used by the database server. Be aware that these backups can only be restored to another Cloud SQL instance. If you wanted to restore a backup on a server other than one running in Cloud SQL, you would run a traditional SQL backup command.

Backups are done daily. You can enable point-in-time recovery. If you enable this feature, in the event of a disaster, you can restore the database back to any point in time before the disaster.

Add a failover replica for high availability

- Creates a second database server in another zone in the region where the main was created.
- Uses Google Cloud regional persistent disks to keep the data synchronized.
- Will automatically switch to the failover if the main becomes unavailable.


Availability
Choice affects cost. You can change this option at any time by editing your instance.

- ○ Single zone
  In case of outage, no failover. Not recommended for production instances.
- [✓ High availability (regional)]
  Automatic failover to another zone within your selected region. Recommended for production instances. Increases cost.

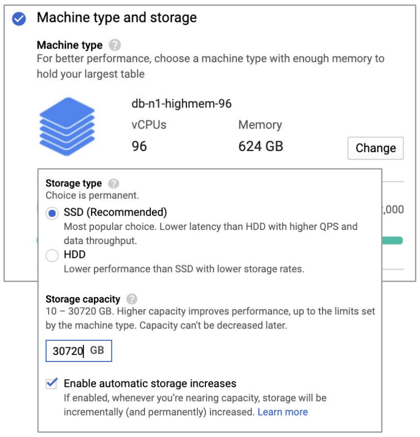
Google Cloud

If your database was corrupted and you had to recover from a backup, you would be down for the amount of time required to run the restore. It shouldn't take very long for the restore to run, but you would first have to discover the problem and determine that you needed to run the restore. That might take considerably longer.

If you can't tolerate much downtime, you can enable high availability when deploying your database. This creates a failover database in another zone. If the main database becomes unavailable, Cloud SQL automatically switches to the failover until the main comes back online. It takes only about a minute for the system to perform the failover operation, so downtime is minimized.

Scale Cloud SQL vertically to increase capacity

- Start small and scale up as needed:
  - Up to 96 vCPUs and 624 GB RAM
  - Up to about 30 TB storage
- Network throughput automatically increases as you add CPUs.
- Disk performance automatically increases as the disk becomes larger:
  - Use SSD storage for faster disks.
  - Use HDD storage to save money.



Machine type and storage

Machine type
For better performance, choose a machine type with enough memory to hold your largest table

db-n1-highmem-96
vCPUs 96
Memory 624 GB

Storage type
Choice is permanent.
● SSD (Recommended)
Most popular choice. Lower latency than HDD with higher QPS and data throughput.
○ HDD
Lower performance than SSD with lower storage rates.

Storage capacity
10 – 30720 GB. Higher capacity improves performance, up to the limits set by the machine type. Capacity can't be decreased later.

30720 GB

✓ Enable automatic storage increases
If enabled, whenever you're nearing capacity, storage will be incrementally (and permanently) increased. Learn more

Google Cloud

When configuring a database, start small and scale up as needed by adding machine resources.

Cloud SQL supports up to 96 vCPUS and hundreds of GBs of RAM. Disk size ranges from 10 GB to about 30 TB.

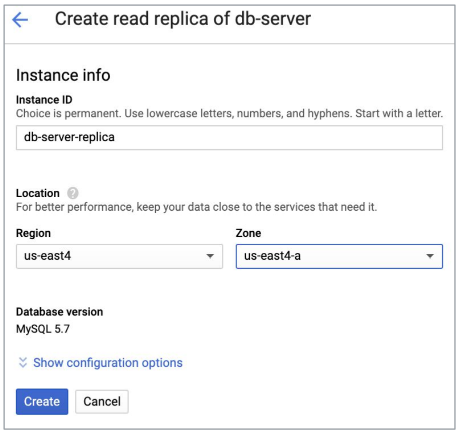
As you increase the number of CPUs, the amount of network throughput automatically increases.

When configuring disk storage, you can choose standard or SSD drives. SSD drives are much faster, but also significantly more expensive.

If you enable automatic storage increases, the service will allocate more disk space for you as needed.

Add read replicas to support high-volume reads

- Read replicas can be in different regions to support users around the globe.
- Read replicas have different IP addresses and connection names than the main.
    - Configure client applications to use the closest replica when running SELECT queries.



Create read replica of db-server

Instance info

Instance ID
Choice is permanent. Use lowercase letters, numbers, and hyphens. Start with a letter.
db-server-replica

Location
For better performance, keep your data close to the services that need it.

Region us-east4

Zone us-east4-a

Database version
MySQL 5.7

Google Cloud

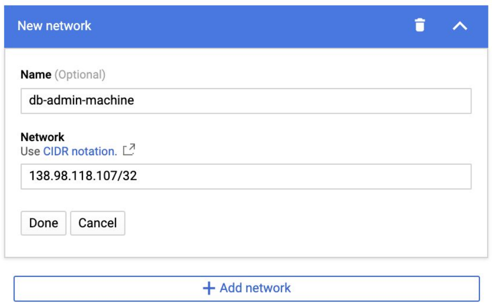
For users who need to support high volumes of reads, you can automate the creation of read replicas.

Read replicas can even be in multiple regions around the world. The replicas are automatically synchronized with the main.

Each read replica will be allocated a different IP address. When configuring your applications, send all the writes to the main and send the reads to the replica closest to the user.

Firewall blocks access to the Cloud SQL database

- By default, there is no access to the database from outside the project.
- You can authorize networks as needed using CIDR addressing.



New network

Name (Optional)

db-admin-machine

Network

Use CIDR notation.

138.98.118.107/32

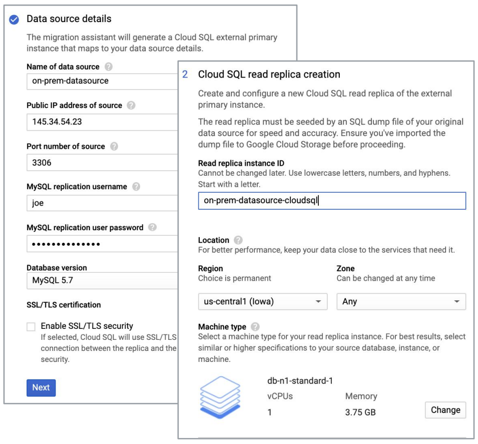
Google Cloud

As we said earlier, a firewall blocks access to the database. By default, there is no access to the database from outside the project.

You can authorize networks as needed using CIDR addressing.

Cloud SQL Migration Assistant can help you migrate a database into Cloud SQL

- On-premises to Google Cloud SQL
- Other cloud provider to Cloud SQL
- One Google Cloud project to another
- Currently, only supports MySQL

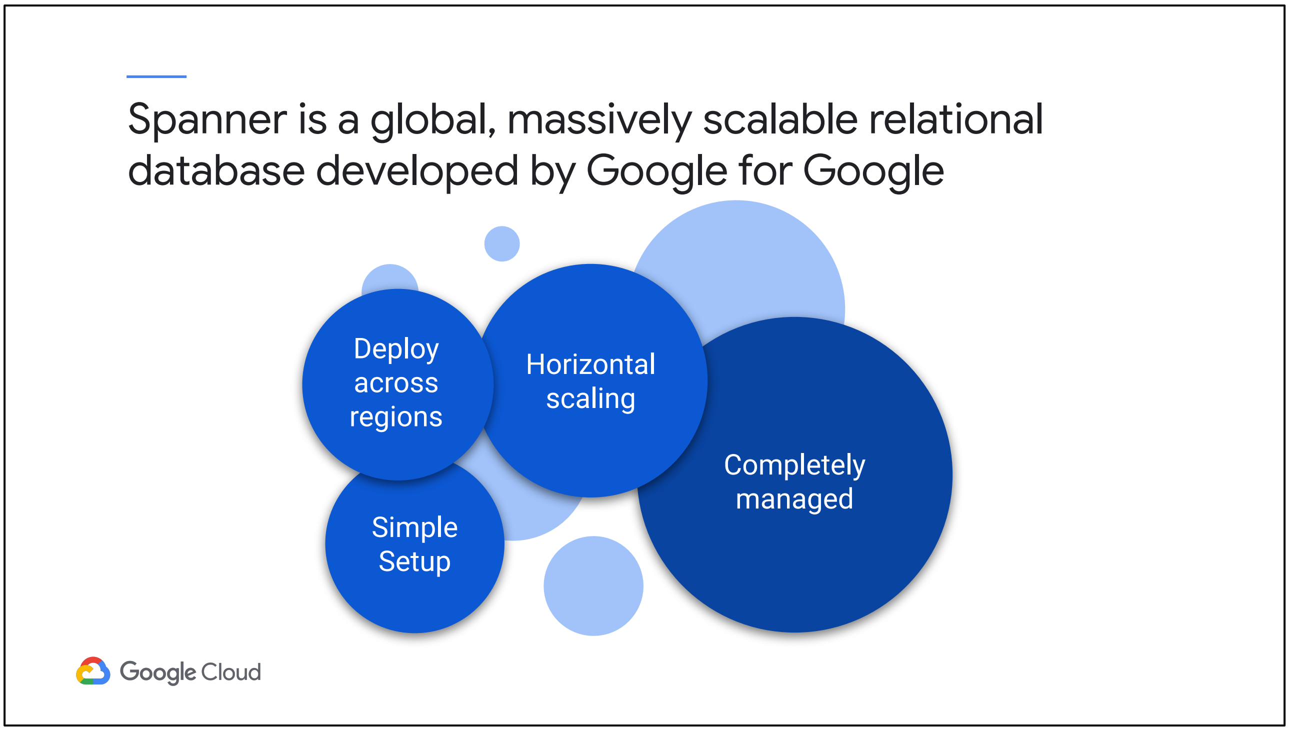

Data source details
The migration assistant will generate a Cloud SQL external primary instance that maps to your data source details.
Name of data source: on-prem-datasource
Public IP address of source: 145.34.54.23
Port number of source: 3306
MySQL replication username: joe
MySQL replication user password: ••••••••••••••
Database version: MySQL 5.7
SSL/TLS certification
☐ Enable SSL/TLS security
If selected, Cloud SQL will use SSL/TLS connection between the replica and the security.

2 Cloud SQL read replica creation
Create and configure a new Cloud SQL read replica of the external primary instance.
The read replica must be seeded by an SQL dump file of your original data source for speed and accuracy. Ensure you've imported the dump file to Google Cloud Storage before proceeding.
Read replica instance ID
Cannot be changed later. Use lowercase letters, numbers, and hyphens. Start with a letter.
on-prem-datasource-cloudsql
Location
For better performance, keep your data close to the services that need it.
Region — Choice is permanent: us-central1 (Iowa)
Zone — Can be changed at any time: Any
Machine type
Select a machine type for your read replica instance. For best results, select similar or higher specifications to your source database, instance, or machine.
db-n1-standard-1
vCPUs: 1
Memory: 3.75 GB

Google Cloud

Google also provides the Cloud SQL Migration Assistant, which can help you migrate an on-premises database or a database running in another cloud, or from one Cloud SQL database to another one in a different project.

This is a fairly new feature and at the time of this writing is only supported for MySQL.



Spanner is a global, massively scalable relational database developed by Google for Google

Deploy across regions
Horizontal scaling
Completely managed
Simple Setup
Google Cloud

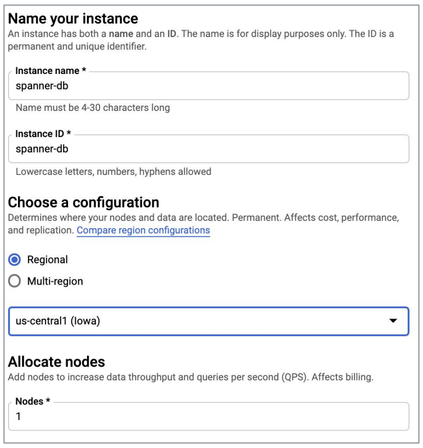
Spanner is a global, massively scalable relational database developed by Google for Google. It is the second choice we discuss for managed relational database solutions in Google Cloud.

- Spanner is completely managed. You don't have to worry about patches, backups, or any other administrative chores.
- Setup is simple. You just answer a couple of questions and the database is created in a few minutes.
- Spanner provides options for cross-region deployments.
- Spanner is massively scalable. You scale out horizontally by adding nodes. Each node can accommodate approximately 2 TB of data, and you could have 1000s of nodes, assuming of course, that you can afford it.

Creating a Spanner database is easy

1. Name your instance.
2. Specify a location.
3. Specify the number of nodes.

```
gcloud spanner instances create spanner-db
--config=regional-us-central1 --nodes=1
--description="Spanner Database"
```



Name your instance

An instance has both a **name** and an **ID**. The name is for display purposes only. The ID is a permanent and unique identifier.

Instance name *: spanner-db

Name must be 4-30 characters long

Instance ID *: spanner-db

Lowercase letters, numbers, hyphens allowed

Choose a configuration

Determines where your nodes and data are located. Permanent. Affects cost, performance, and replication. Compare region configurations

[✓ Regional] Multi-region

us-central1 (Iowa)

Allocate nodes

Add nodes to increase data throughput and queries per second (QPS). Affects billing.

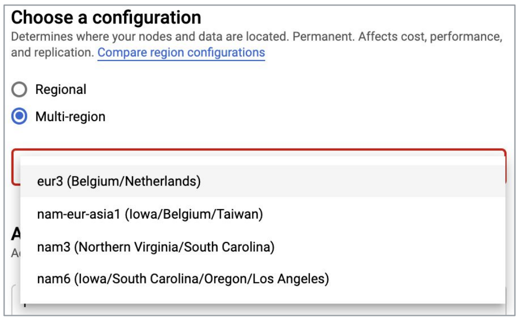
Nodes *: 1

Google Cloud

To create a Spanner database, you only have to specify a name, a location, and the number of nodes. Like all other resources, you can use the Google Cloud Console or the Command Line Interface.

Deployments can be regional or multi-regional

- Regional deployments have all nodes and data in a single region:
  - 99.99% availability SLA
- Multi-regional deployments replicate nodes and data in more than one region:
  - 99.999% availability SLA
  - Can be multiple regions in Europe or the US.
  - Can be multiple regions around the world (US, Europe, and Asia).



Choose a configuration

Determines where your nodes and data are located. Permanent. Affects cost, performance, and replication. Compare region configurations

Regional [✓ Multi-region]

- eur3 (Belgium/Netherlands)
- nam-eur-asia1 (Iowa/Belgium/Taiwan)
- nam3 (Northern Virginia/South Carolina)
- nam6 (Iowa/South Carolina/Oregon/Los Angeles)

Google Cloud

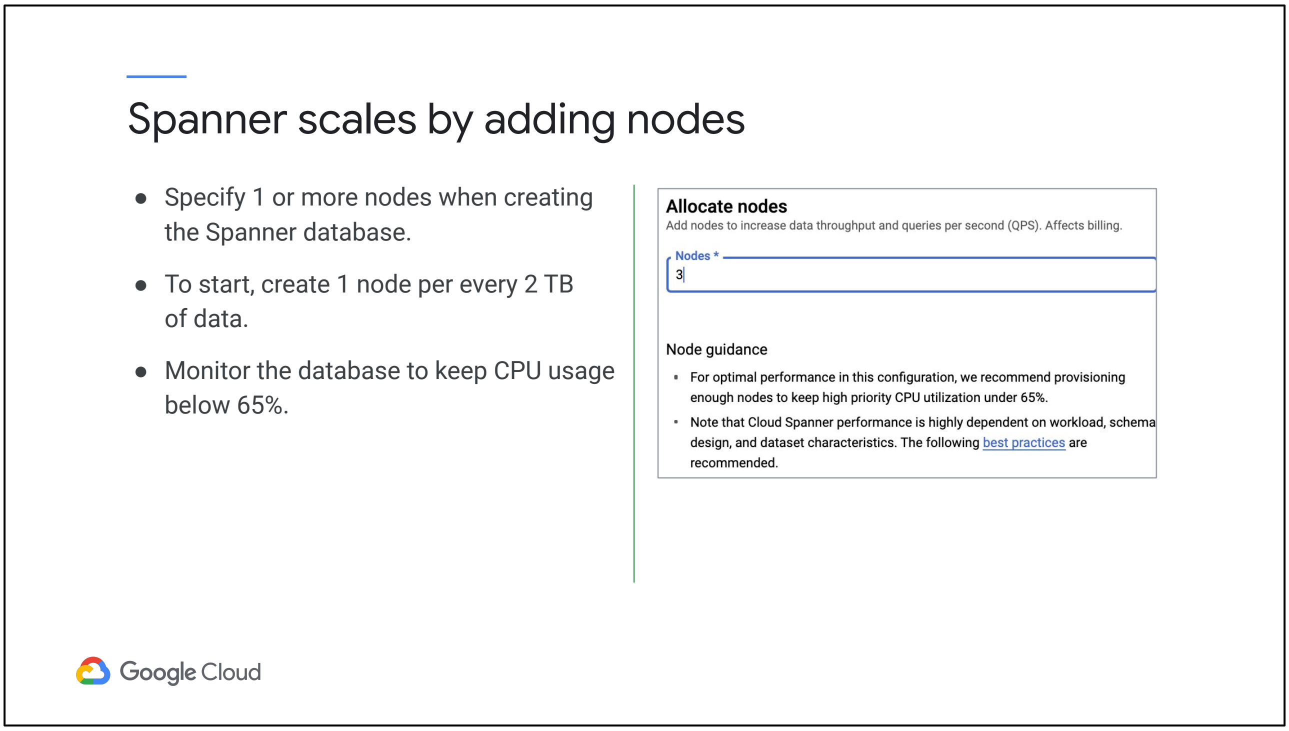
When specifying location, you can choose a regional or multi-regional deployment.

Regional deployments have all nodes and data in a single region and offer a 99.99% availability SLA.

Multi-regional deployments replicate nodes and data in more than one region. These offer a 99.999% availability SLA.

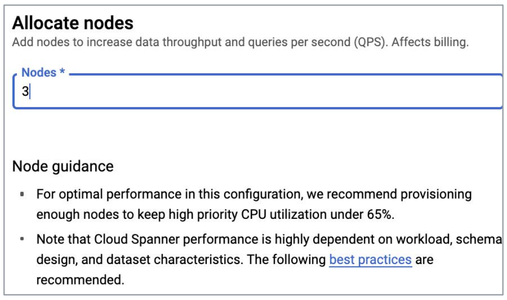
You can choose multiple regions in Europe or the US. This would provide 5 9s of availability on a single continent. As an example, a large bank that operates only in the United States or only in Europe might choose this type of deployment in order to have extremely high availability for their customers.

If a company had users all over the world, they could choose a deployment replicated across the US, Europe, and Asia. This would provide low latency and high performance to users everywhere.



Spanner scales by adding nodes

- Specify 1 or more nodes when creating the Spanner database.
- To start, create 1 node per every 2 TB of data.
- Monitor the database to keep CPU usage below 65%.



Allocate nodes

Add nodes to increase data throughput and queries per second (QPS). Affects billing.

Nodes *

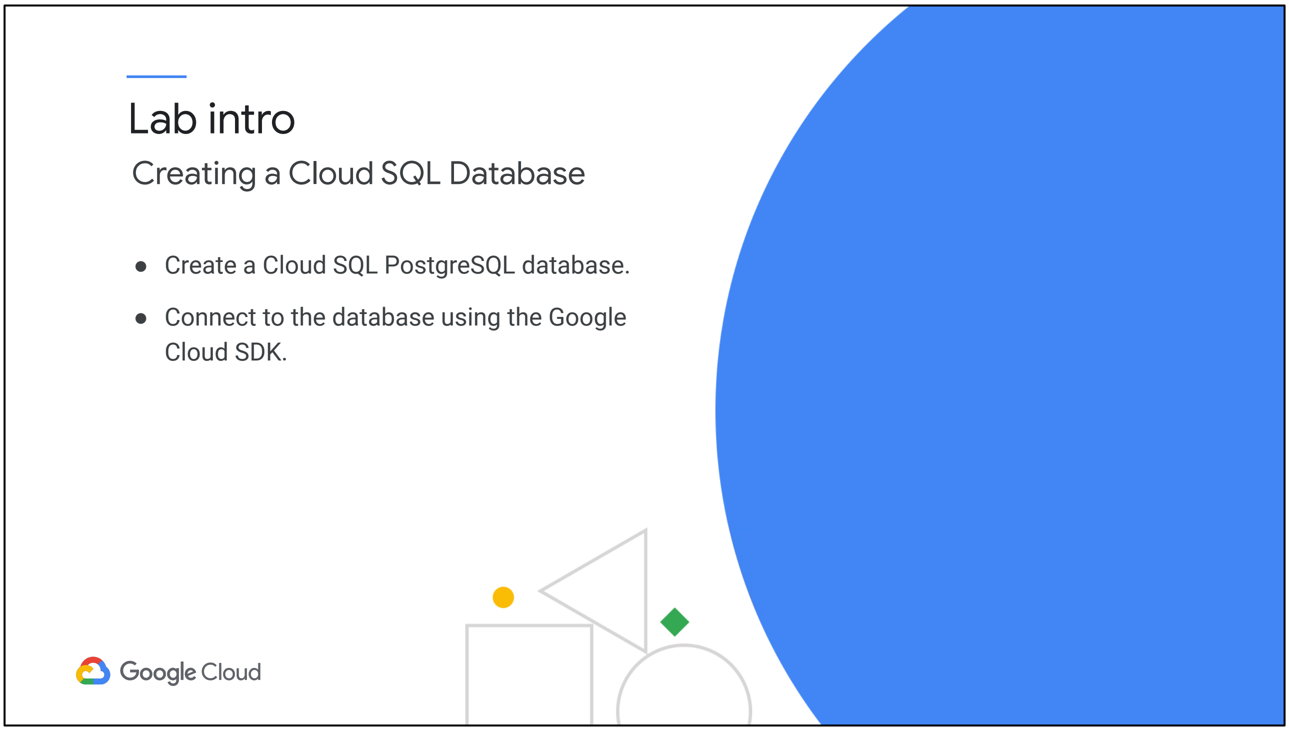
3

Node guidance

- For optimal performance in this configuration, we recommend provisioning enough nodes to keep high priority CPU utilization under 65%.
- Note that Cloud Spanner performance is highly dependent on workload, schema design, and dataset characteristics. The following best practices are recommended.

Google Cloud

A Spanner database can operate with a single node. Over time as your application and database grows, you can add more nodes. Each node can handle approximately 2 TB of data. Google recommends that you also monitor the database and keep CPU usage below 65%. Add more nodes if your database exceeds that.



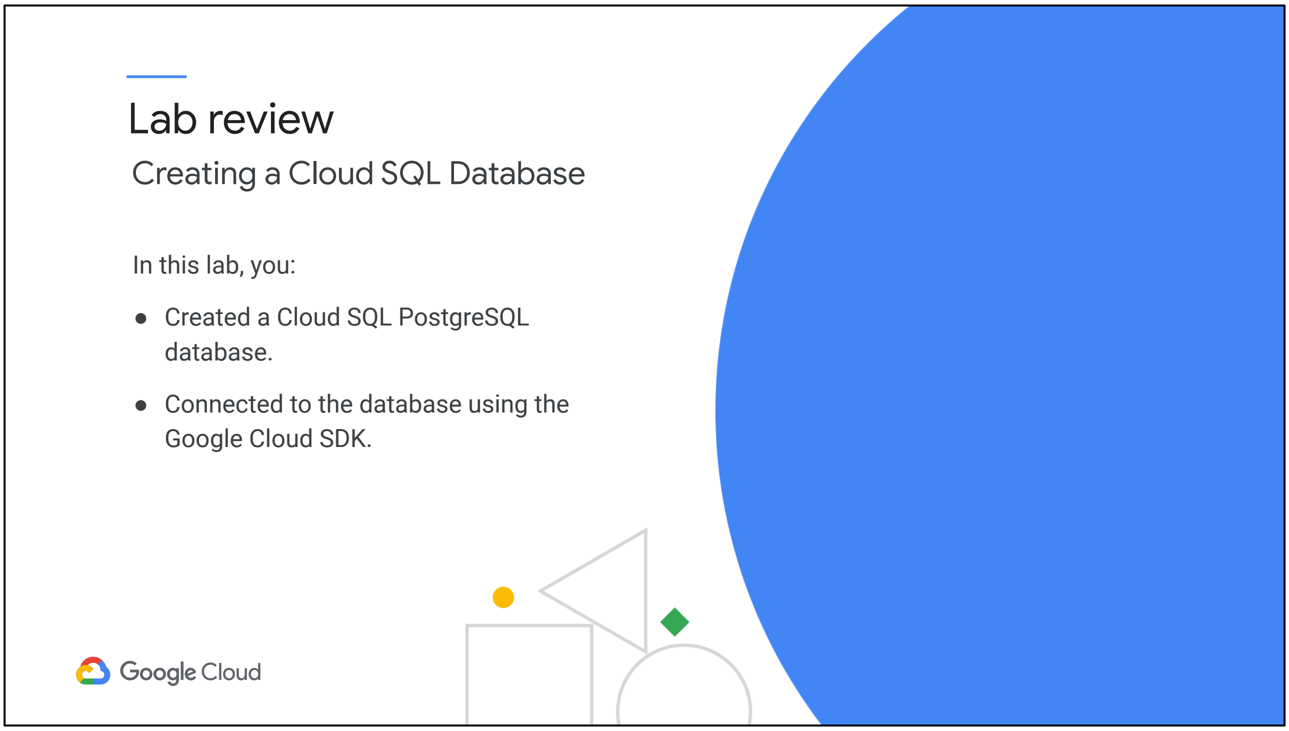
Lab intro

Creating a Cloud SQL Database

- Create a Cloud SQL PostgreSQL database.
- Connect to the database using the Google Cloud SDK.

Google Cloud

In this lab, you create a PostgreSQL database using Cloud SQL. You also see how to connect to it from a client machine using the Google Cloud SDK.



Lab review

Creating a Cloud SQL Database

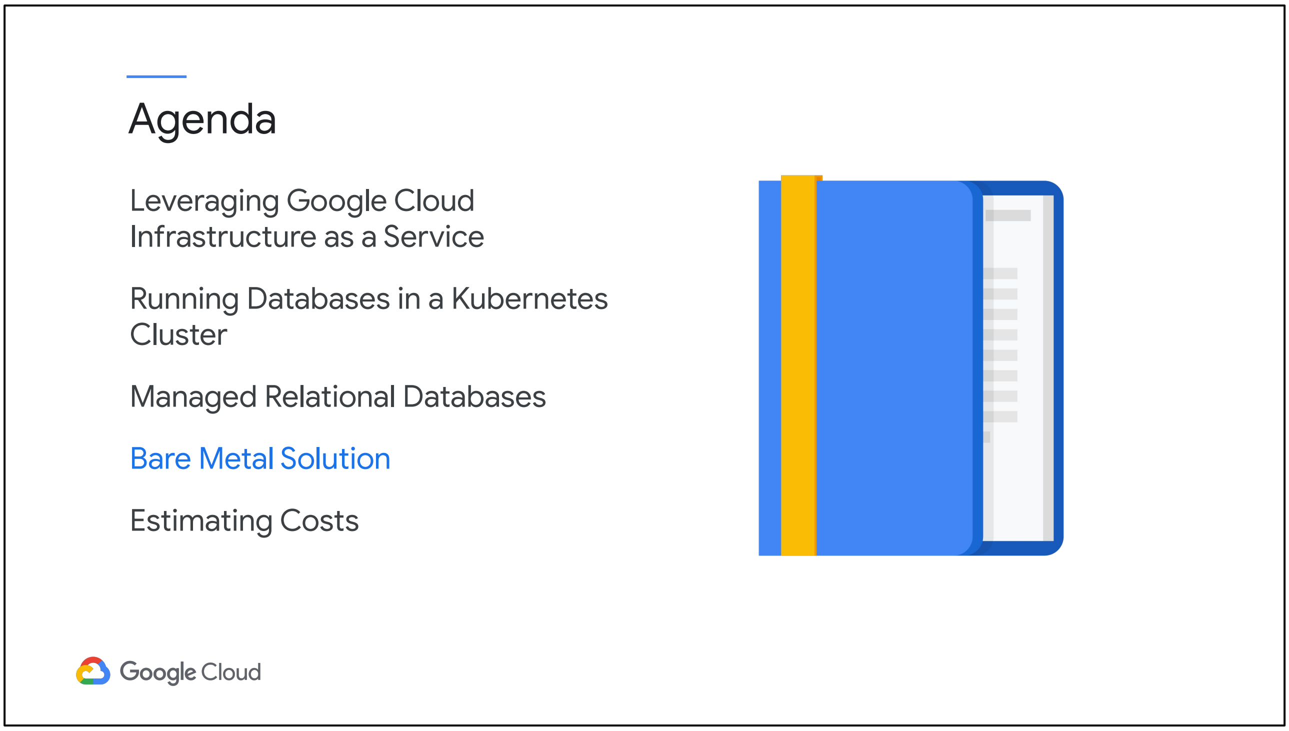
In this lab, you:

- Created a Cloud SQL PostgreSQL database.
- Connected to the database using the Google Cloud SDK.

In this lab, you:
Created a Cloud SQL PostgreSQL database.
Connected to the database using the Google Cloud SDK.

Cloud SQL provides a more managed database solution compared to one with Compute Engine or Kubernetes Engine. While there are some limitations it automate backups and other administrative chores. It also allows you to easily create failover and read replicas.

Cloud SQL supports MySQL, PostgreSQL and SQL Server. Next, you will learn how to use Bare Metal Solution to run Oracle.



Agenda

Leveraging Google Cloud Infrastructure as a Service

Running Databases in a Kubernetes Cluster

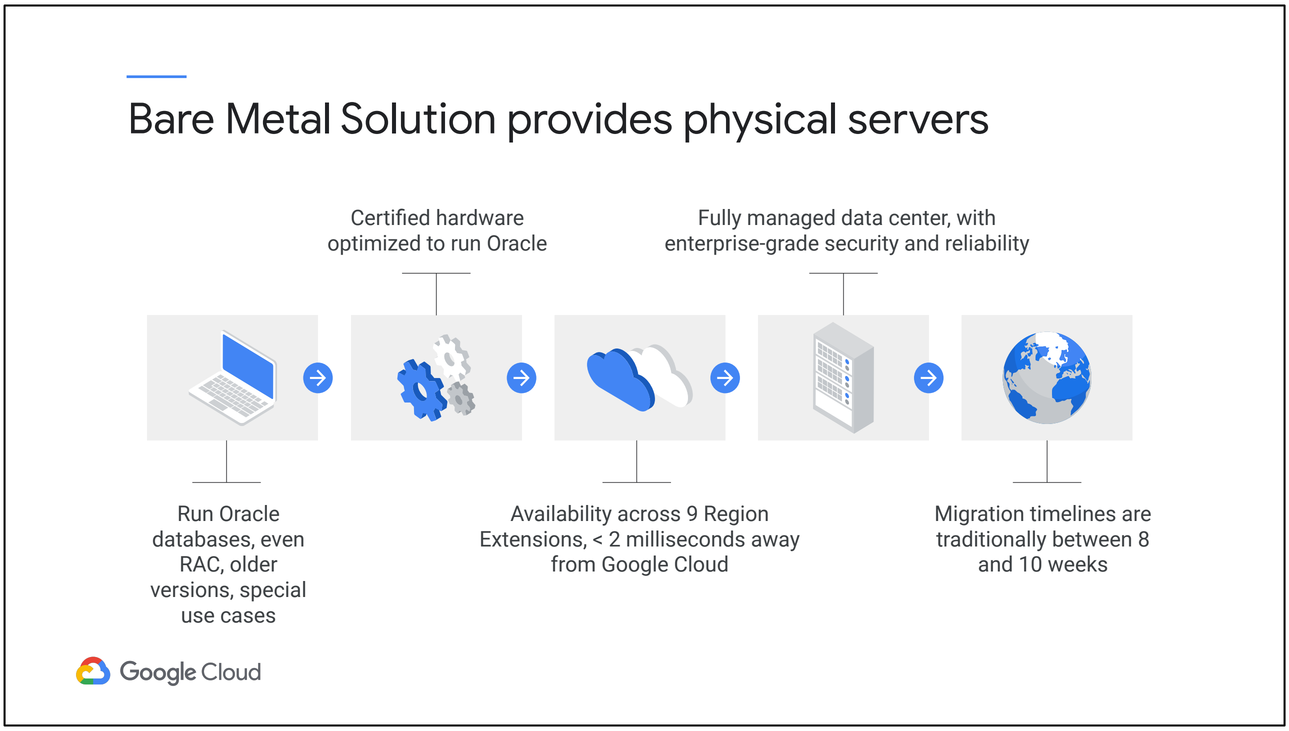
Managed Relational Databases

Bare Metal Solution

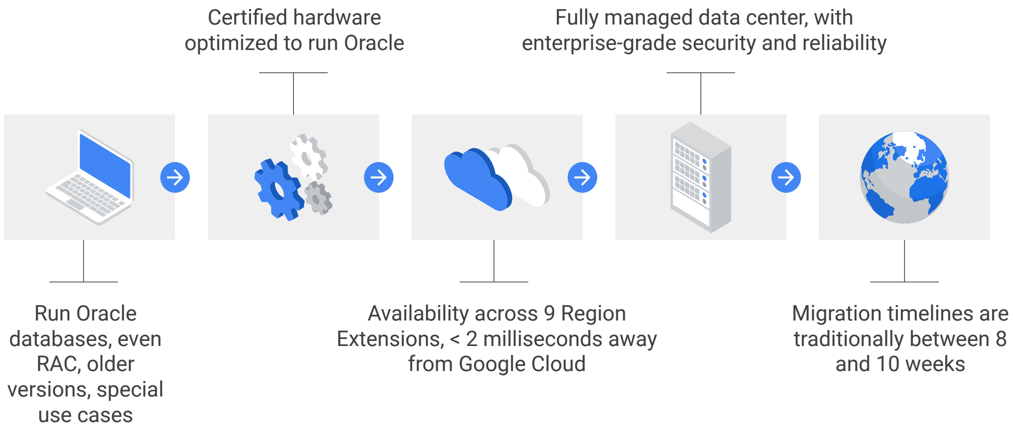
Estimating Costs

Google Cloud

In theory, an Oracle database would run on a Compute Engine virtual machine. However, Oracle has not approved Google Cloud as an official platform, so that is not a viable option. It would not be consistent with Oracle's license agreement and it would not be supported. There is a workaround though for customers who want to take advantage of Google Cloud for their applications and still keep their Oracle databases: it is the Google Cloud Bare Metal Solution.



Bare Metal Solution provides physical servers


Certified hardware optimized to run Oracle
Fully managed data center, with enterprise-grade security and reliability
Run Oracle databases, even RAC, older versions, special use cases
Availability across 9 Region Extensions, < 2 milliseconds away from Google Cloud
Migration timelines are traditionally between 8 and 10 weeks

Google Cloud

As the name implies, Bare Metal Solution provides physical servers. There are no limitations. You connect to these machines as if they were in your own data center, and you can configure them any way you like.

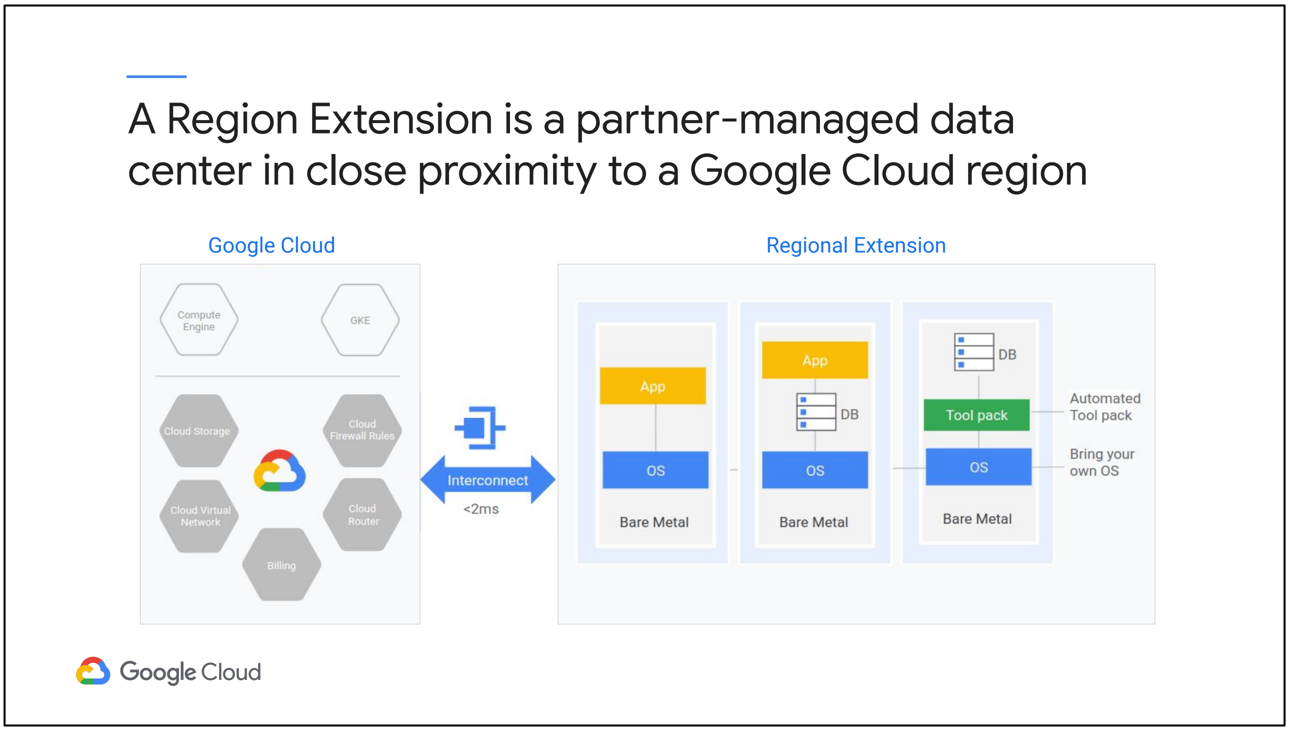
- For Oracle databases, you can use any version you like, even older versions that are not supported by managed services like Amazon's RDS. All Oracle features are also supported; most importantly, Real Application Clusters (RAC), which is ***not*** supported by managed cloud database services running Oracle.

- The hardware that is provisioned for you is certified and optimized to run Oracle, so you won't have any license and support issues. Also, for managed cloud services, Oracle charges for licenses by the virtual CPU. There are two vCPUs per CPU core. When running on physical hardware, you pay by the physical core. This is a significant savings on the license cost compared to managed cloud-based solutions.

- The bare metal servers actually run in a partner's data center that is geographically close to a Google region. Google guarantees less than 2 ms latency between the partner data center and Google. At the time of this writing, Bare Metal Solution is available in 9 regions.

- The data centers where the Bare Metal Solution servers run are enterprise-grade in terms of security and reliability and are backed with

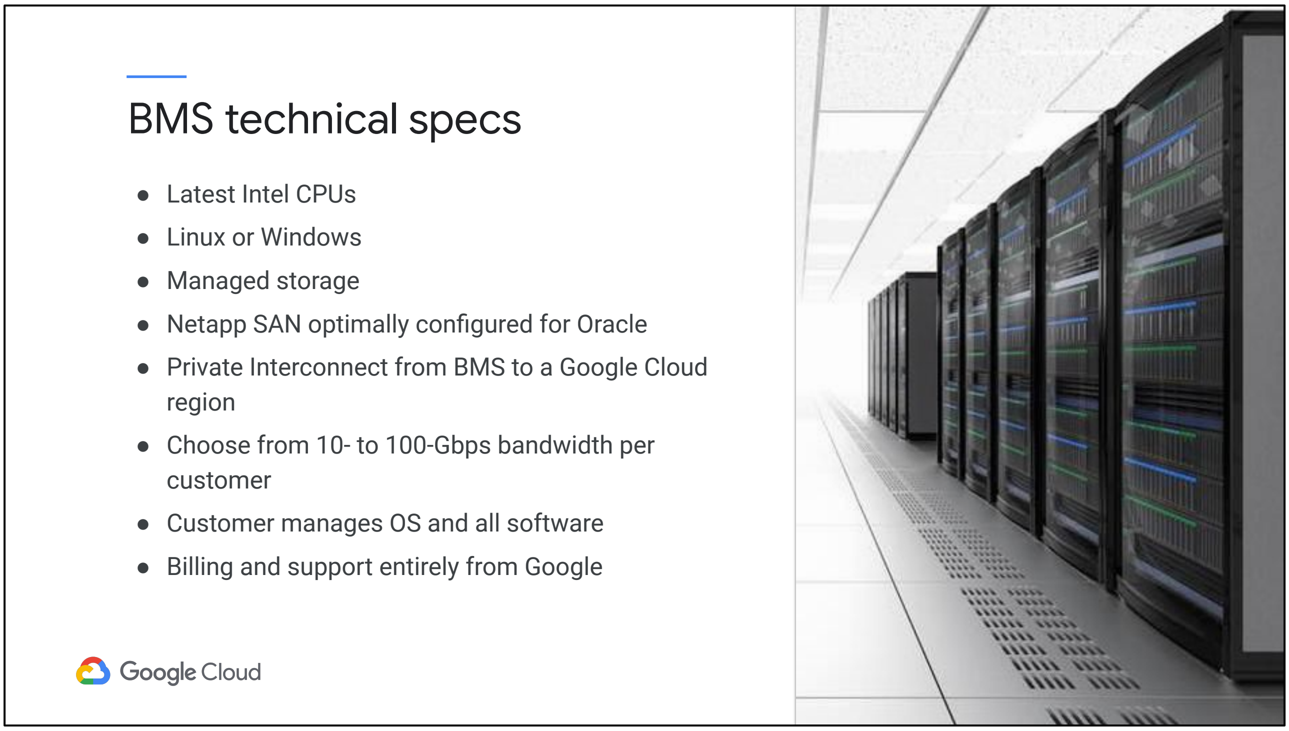
- Google's support.
- When you use Bare Metal Solution, the servers take some time to be provisioned, but migration times traditionally can be completed in only 8 to 10 weeks.



A Region Extension is a partner-managed data center in close proximity to a Google Cloud region

The partner data centers where the Bare Metal Solution servers run are called regional extensions. They are connected to Google via a high-speed interconnect and are geographically close for a very fast connection.

The partner provisions the servers. A network is created by Google that provides access to your machines. You connect to your servers via a Bastion host and configure them any way you like.

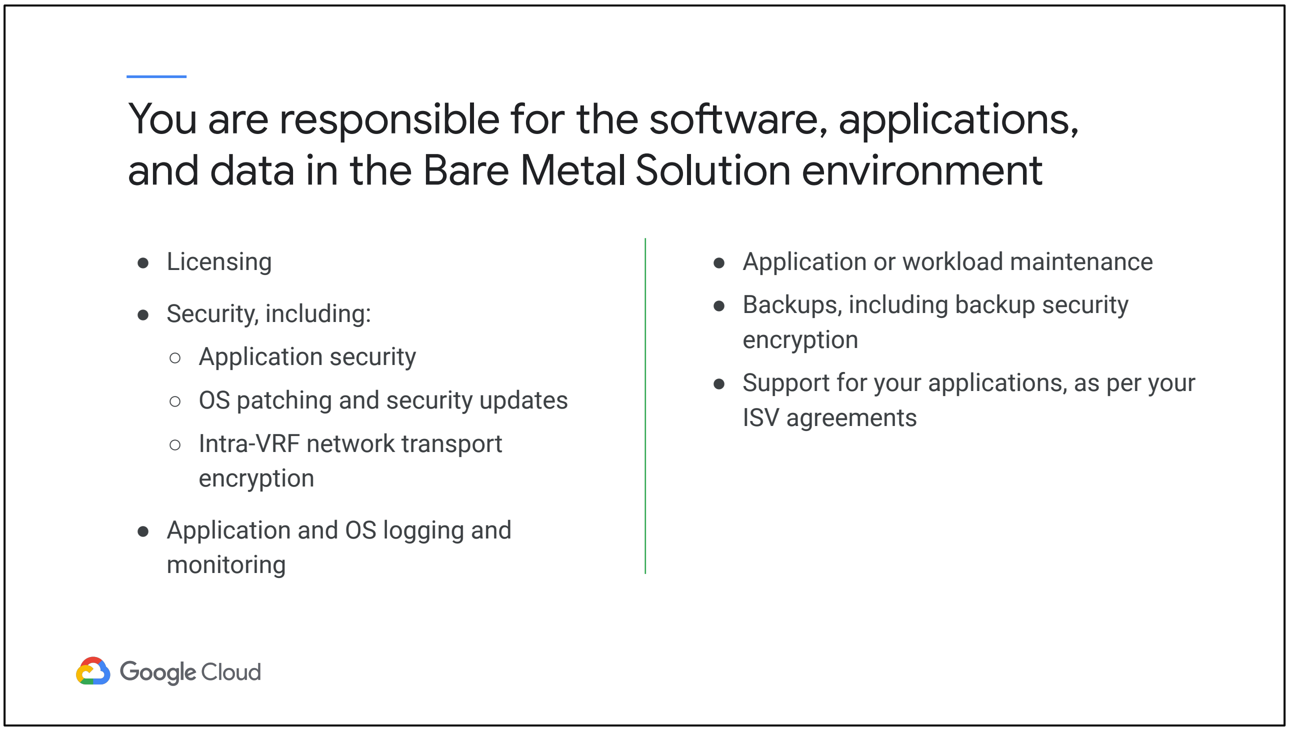


BMS technical specs

- Latest Intel CPUs
- Linux or Windows
- Managed storage
- Netapp SAN optimally configured for Oracle
- Private Interconnect from BMS to a Google Cloud region
- Choose from 10- to 100-Gbps bandwidth per customer
- Customer manages OS and all software
- Billing and support entirely from Google

Google Cloud

- Bare Metal Solution servers use the latest Intel processors.
- Depending on your preference, either Windows or Linux is installed on the servers for you,
- Storage is also provisioned for you…
- along with a Netapp SAN if desired.
- A high-speed link is set up.
- You can choose the amount of bandwidth required for your workloads.
- The customer manages the machines after the initial setup. You can configure those machines any way you like.
- And finally Google handles all the billing and support.



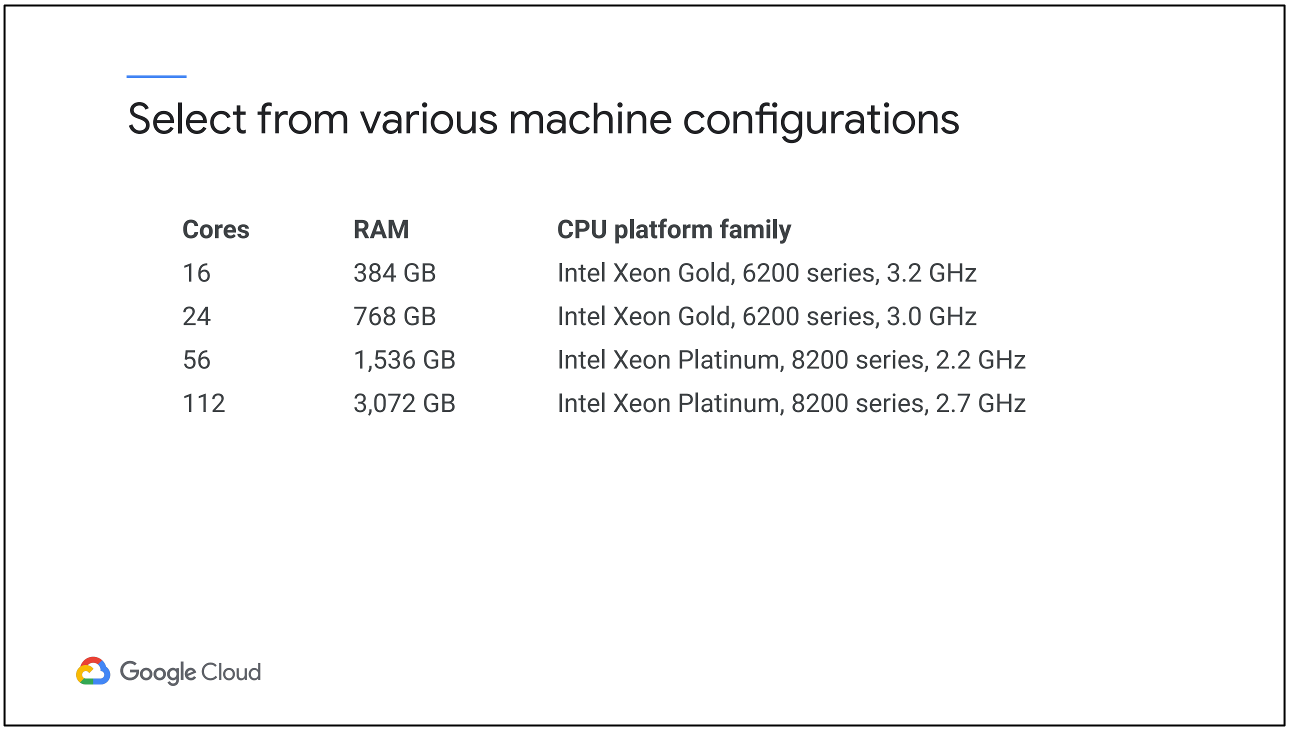
You are responsible for the software, applications, and data in the Bare Metal Solution environment

- Licensing
- Security, including:
  - Application security
  - OS patching and security updates
  - Intra-VRF network transport encryption
- Application and OS logging and monitoring
- Application or workload maintenance
- Backups, including backup security encryption
- Support for your applications, as per your ISV agreements

Google Cloud

- Customers are responsible for the software, applications, and data in the Bare Metal Solution environment.
  This includes licensing, security, and patch management. Customers can integrate Google Cloud Operations services for monitoring and logging.

- Customers are also responsible for administering the database, including backups and encryption.

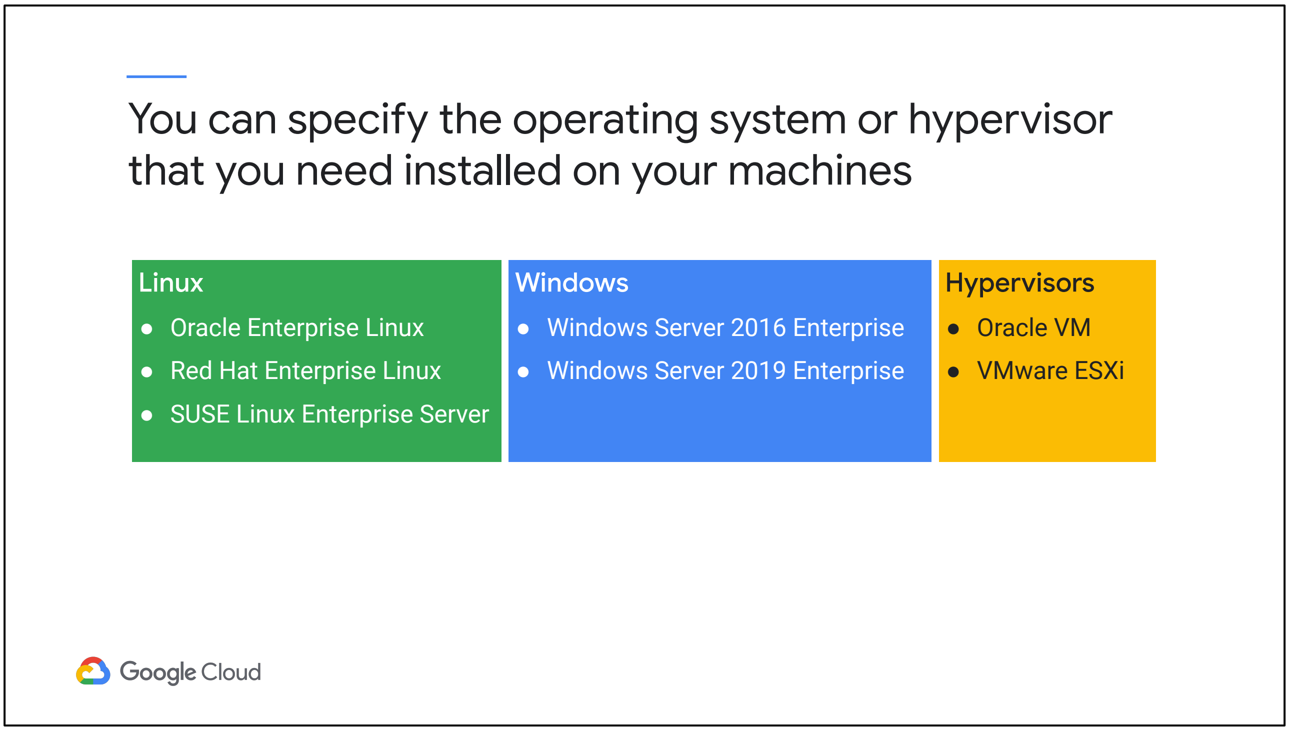


Select from various machine configurations

Cores	RAM	CPU platform family
16	384 GB	Intel Xeon Gold, 6200 series, 3.2 GHz
24	768 GB	Intel Xeon Gold, 6200 series, 3.0 GHz
56	1,536 GB	Intel Xeon Platinum, 8200 series, 2.2 GHz
112	3,072 GB	Intel Xeon Platinum, 8200 series, 2.7 GHz

Google Cloud

This chart shows the current server configurations available when setting up Bare Metal Solution. Choose the server configuration and number of machines that best fits your needs. Visit the Google Cloud website for latest information on configurations as options are subject to change.



You can specify the operating system or hypervisor that you need installed on your machines

Linux
- Oracle Enterprise Linux
- Red Hat Enterprise Linux
- SUSE Linux Enterprise Server

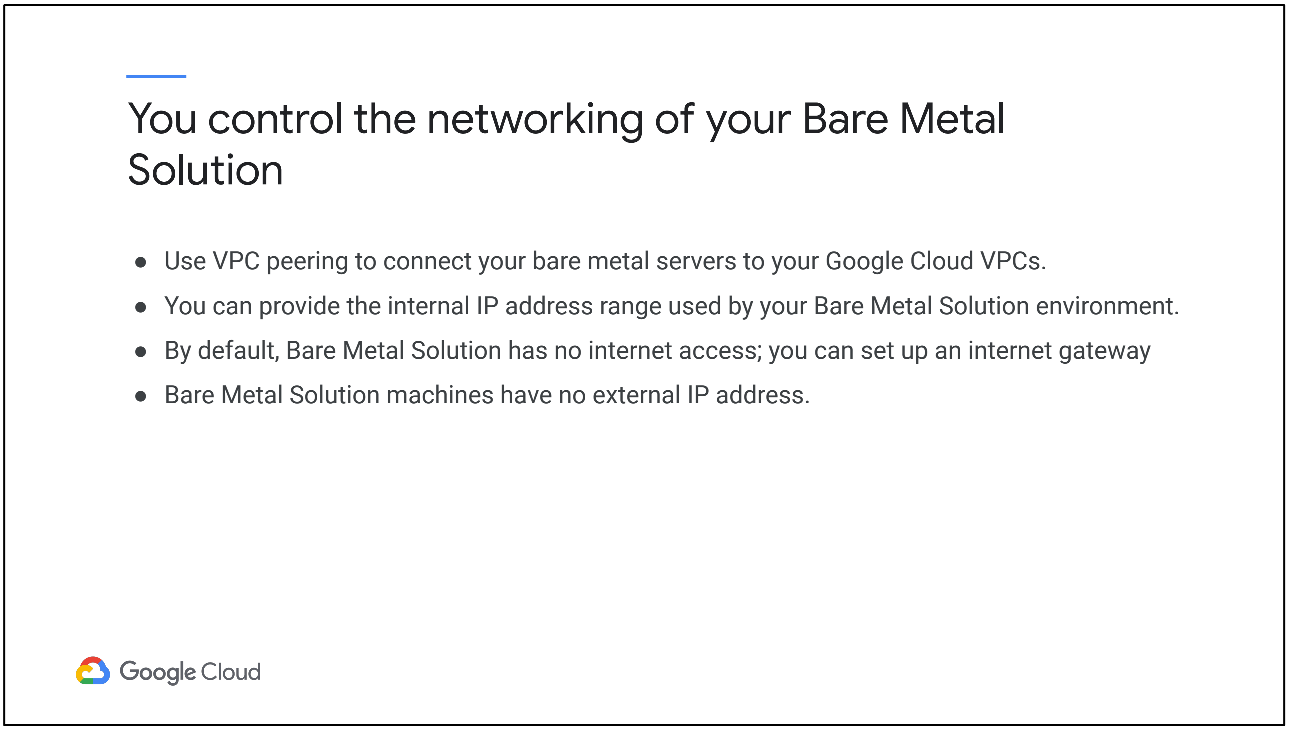
Windows
- Windows Server 2016 Enterprise
- Windows Server 2019 Enterprise

Hypervisors
- Oracle VM
- VMware ESXi

Google Cloud

When you place an order, you can request the operating system you want installed on your servers.

- For Linux, you can choose Oracle Enterprise Linux, Red Hat Enterprise Linux, or SUSE Linux.
- If you prefer Windows, you can choose Windows Server 2016 or 2019 Enterprise edition.
- You can alternatively have a hypervisor installed for virtualization. You can choose either Oracle VM or VMware ESXi.



You control the networking of your Bare Metal Solution

- Use VPC peering to connect your bare metal servers to your Google Cloud VPCs.
- You can provide the internal IP address range used by your Bare Metal Solution environment.
- By default, Bare Metal Solution has no internet access; you can set up an internet gateway
- Bare Metal Solution machines have no external IP address.

Google Cloud

You also have control over the networking of your Bare Metal Solution.

You use VPC peering to connect your Google Cloud VPCs to your servers.

When placing the order, you can provide the internal IP address ranges you want in your Bare Metal Solution environment.

By default, Bare Metal Solution has no internet access; however, you can set up an internet gateway.

Bare Metal Solution machines have no external IP addresses. All access to the machines would be through a Bastion host or a jump server.

Agenda

Leveraging Google Cloud Infrastructure as a Service

Running Databases in a Kubernetes Cluster

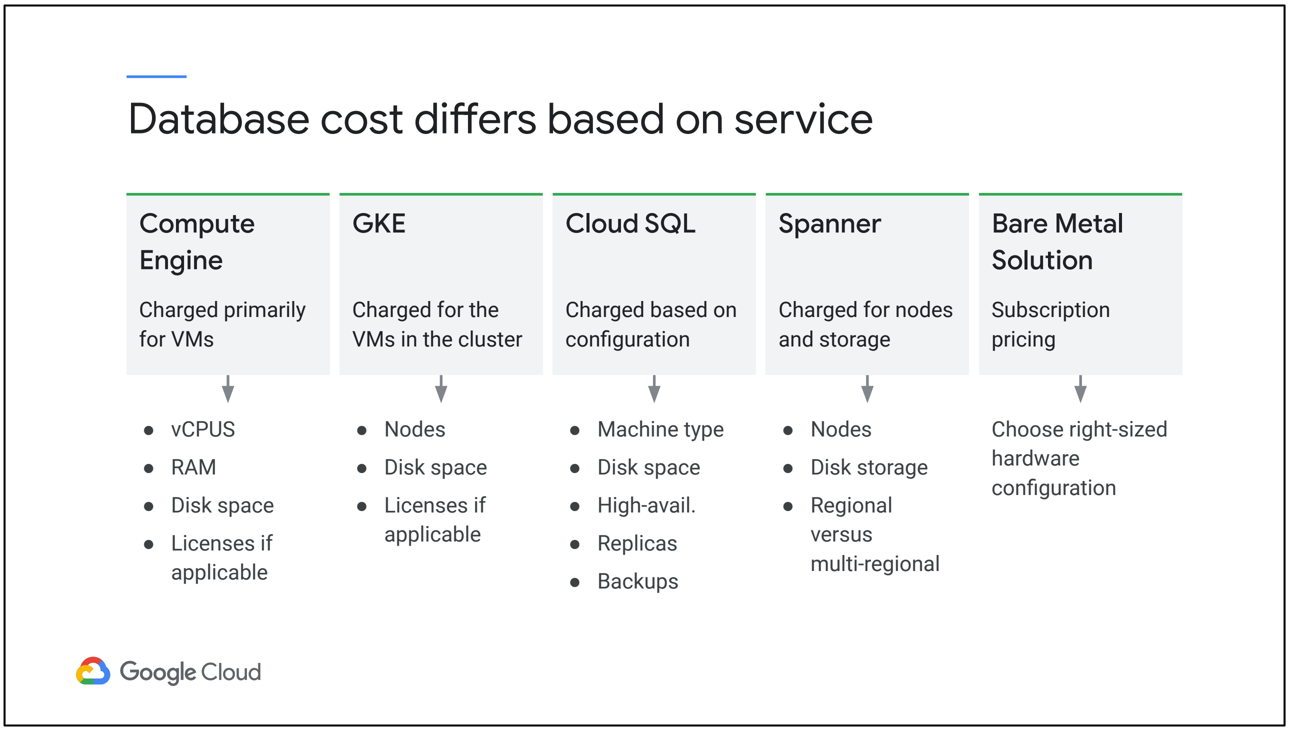
Managed Relational Databases

Bare Metal Solution

Estimating Costs

Google Cloud

In this module, you have seen multiple ways to run your databases in Google Cloud. These include Compute Engine, Kubernetes Engine, Cloud SQL, and Bare Metal Solution. Understanding the costs of various solutions is helpful for determining which one is appropriate to use.



Database cost differs based on service

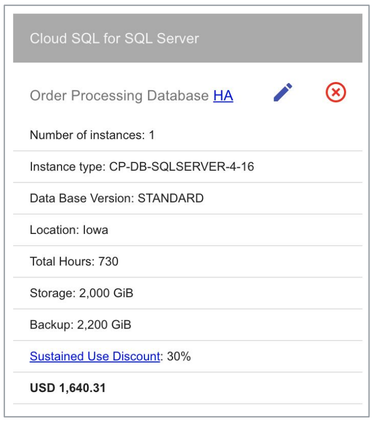
Compute Engine	GKE	Cloud SQL	Spanner	Bare Metal Solution
Charged primarily for VMs	Charged for the VMs in the cluster	Charged based on configuration	Charged for nodes and storage	Subscription pricing
• vCPUS • RAM • Disk space • Licenses if applicable	• Nodes • Disk space • Licenses if applicable	• Machine type • Disk space • High-avail. • Replicas • Backups	• Nodes • Disk storage • Regional versus multi-regional	Choose right-sized hardware configuration

Google Cloud

- When using **Compute Engine**, your primary charges are for vCPUs, RAM, and disk space. You also need to pay for licenses if you are using licensed software.
- With **GKE**, you pay for the virtual machines and disk space used by the cluster.
- When using **Cloud SQL**, you configure a machine type. Larger machines cost more. Of course, you also pay for the disk space used, and if using SQL Service, the license. There are additional charges if you configure the database for high availability, and if you create read replicas. Backups are stored in Cloud Storage, so you also pay a little for those.
- With Google Cloud **Spanner**, you pay for the nodes and the amount of storage space used. The charge increases if you deploy your database to multiple regions.
- Finally, with **Bare Metal Solution**, you choose the hardware configuration that best matches your needs. You can also configure the amount of storage required and the amount of bandwidth required between your Google project and the regional extension.

Use the Google Cloud Pricing Calculator to create initial cost estimates

Balance cost, administrative overhead, and required features when evaluating which service to use.



Cloud SQL for SQL Server

Order Processing Database HA

Number of instances: 1

Instance type: CP-DB-SQLSERVER-4-16

Data Base Version: STANDARD

Location: Iowa

Total Hours: 730

Storage: 2,000 GiB

Backup: 2,200 GiB

Sustained Use Discount: 30%

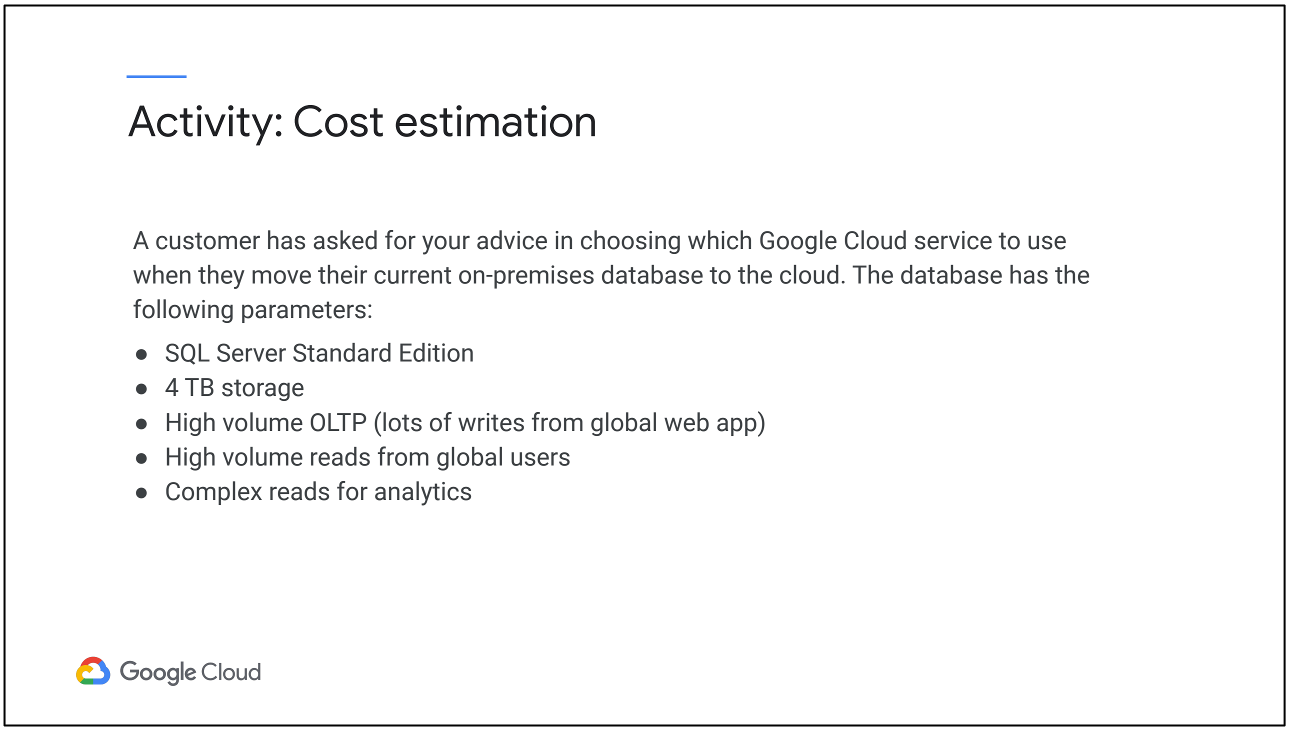
USD 1,640.31

Google Cloud

You can use the Google Cloud Pricing Calculator to estimate the costs for various solutions. Use it make make comparisons between the various services. Of course, this is just the price Google charges. Make sure you estimate the total cost of ownership for various choices.

Balance cost, administrative overhead, and required features when evaluating which service to use. A completely managed solution like Spanner might initially look expensive, but when you add in the administrative overhead of other solutions, it may be more cost-effective than you expected.

You will find the price calculator on the Google Cloud website.



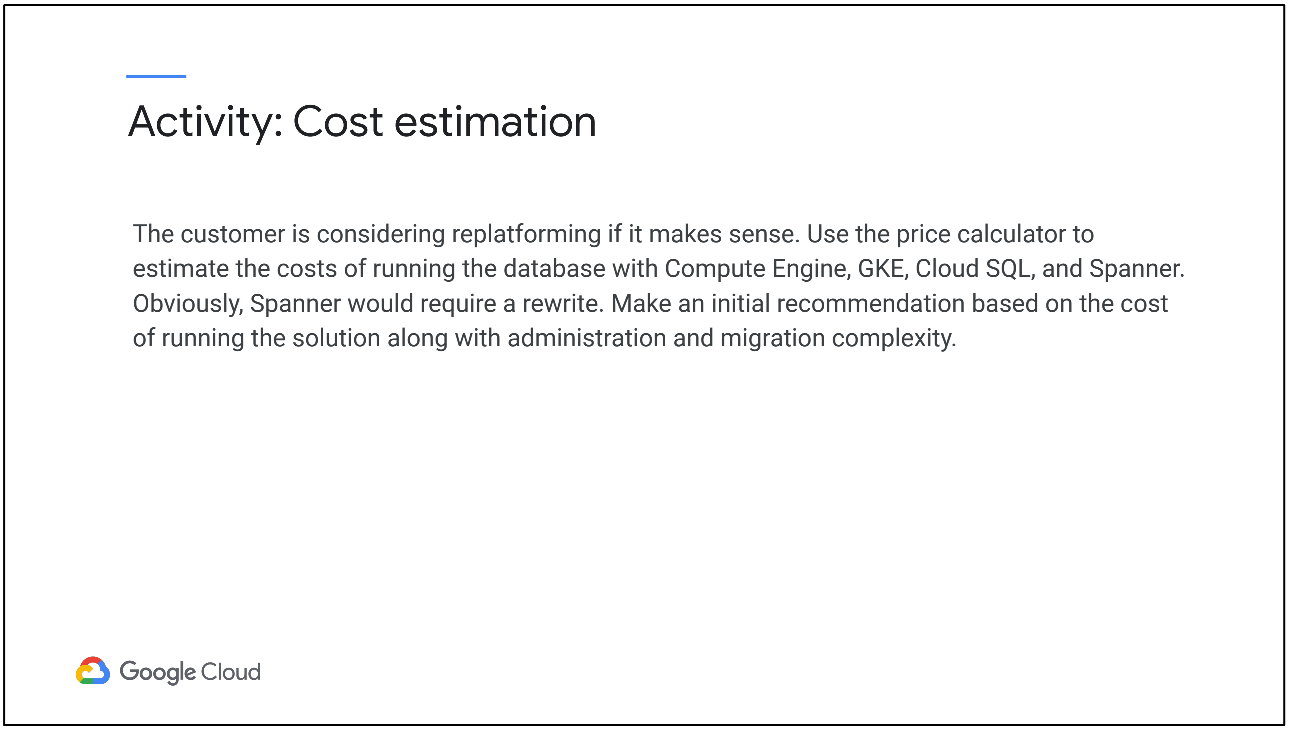
Activity: Cost estimation

A customer has asked for your advice in choosing which Google Cloud service to use when they move their current on-premises database to the cloud. The database has the following parameters:

- SQL Server Standard Edition
- 4 TB storage
- High volume OLTP (lots of writes from global web app)
- High volume reads from global users
- Complex reads for analytics

Google Cloud

Next is an activity to apply what you just learned. For this activity consider a fictitious customer and their existing systems. The customer is particularly interested in their database deployments with specifications provided on the screen and in the case study.



Activity: Cost estimation

The customer is considering replatforming if it makes sense. Use the price calculator to estimate the costs of running the database with Compute Engine, GKE, Cloud SQL, and Spanner. Obviously, Spanner would require a rewrite. Make an initial recommendation based on the cost of running the solution along with administration and migration complexity.

Google Cloud

Use the pricing calculator to estimate the cost of running the database using the various services. When doing the estimates, some values are hard to predict and are almost a guess, but do your best. This activity will give you a good idea of how you are being charged for each service. Doing a comparison like this for any real-world projects you work on is highly recommended.

Module review

Google Cloud

In this module, you:

Evaluated the database solutions available on Google Cloud.
Ran databases on Google Cloud infrastructure using Compute Engine.
Leveraged Kubernetes and GKE for deploying databases.
Used Cloud SQL for managed database solutions.
Learned about provisioning Bare Metal Solution for Oracle databases.
And you Estimated the cost of database solutions.

Google Cloud

Google Cloud