Summary

In this chapter, you learned how to perform basic management tasks for working with Kubernetes clusters, nodes, pods, and services. The chapter also described how to list the contents of container image repositories. You learned how to pin services in the Cloud Console menu, view the status of Kubernetes clusters, and view image repository and image details using `gcloud` commands. This chapter also described how to modify and remove nodes and pods. You also saw the benefits of autoscaling in a real-world scenario.

Both Cloud Console and Cloud SDK, including Cloud Shell, can be used to add, remove, and modify nodes, pods, and services. They both can be used to review the contents of an image repository. Some of the most useful commands include `gcloud container clusters create` and `gcloud container clusters resize`. The `kubectl` command is used to modify Kubernetes resources such as deployments and pods.

Exam Essentials

Know how to view the status of a Kubernetes cluster. Use Cloud Console to list clusters and drill down into clusters to see details of the cluster, including node, pod, and container details. Know the `gcloud container clusters` command and its options.

Understand how to add, modify, and remove nodes. Use Cloud Console to modify nodes and know how to add and remove nodes by changing deployments. Use the `gcloud container clusters resize` command to add and remove nodes.

Understand how to add, modify, and remove pods. Use Cloud Console to modify pods and to add and remove pods by changing deployments. Use `kubectl get deployments` to list deployments, `kubectl scale deployment` to modify the number of deployments, and `kubectl autoscale deployment` to enable autoscaling.

Understand how to add, modify, and remove services. Use Cloud Console to modify services and add and remove services by changing deployments. Use `kubectl run` to start services and `kubectl expose deployment` to make a service accessible outside the cluster. Delete a service using the `kubectl delete service` command.

Know how to view Container Registry images and their details. Navigate the Container Registry pages in Cloud Console. Know the `gcloud container images list` and gcloud container images `describe` commands.

Review Questions

You can find the answers in the Appendix.

1. You are running several microservices in a Kubernetes cluster. You've noticed some performance degradation. After reviewing some logs, you begin to think the cluster may be improperly configured, and you open Cloud Console to investigate. How do you see the details of a specific cluster?
   - **A.** Type the cluster name into the search bar.
   - **B.** Click the cluster name.
   - **C.** Use the `gcloud cluster details` command.
   - **D.** None of the above.

2. You are viewing the details of a cluster in Cloud Console and want to see how many vCPUs are available in the cluster. Where would you look for that information?
   - **A.** Node Pools section of the Cluster Details page
   - **B.** Labels section of the Cluster Details page
   - **C.** Summary line of the Cluster Listing page
   - **D.** A and C

3. You have been assigned to help diagnose performance problems with applications running on several Kubernetes clusters. The first thing you want to do is understand, at a high level, the characteristics of the clusters. Which command should you use?
   - **A.** `gcloud container list`
   - **B.** `gcloud container clusters list`
   - **C.** `gcloud clusters list`
   - **D.** None of the above

4. When you first try to use the `kubectl` command, you get an error message indicating that the resource cannot be found or you cannot connect to the cluster. What command would you use to try to eliminate the error?
   - **A.** `gcloud container clusters access`
   - **B.** `gdcloud container clusters get-credentials`
   - **C.** `gcloud auth container`
   - **D.** `gcloud auth container clusters`

5. An engineer recently joined your team and is not aware of your team's standards for creating clusters and other Kubernetes objects. In particular, the engineer has not properly labeled several clusters. You want to modify the labels on the cluster from Cloud Console. How would you do it?
   - **A.** Click the Connect button.
   - **B.** Click the Deploy menu option.
   - **C.** Click the Edit menu option.
   - **D.** Type the new labels in the Labels section.

6. You receive a page in the middle of the night informing you that several services running on a Kubernetes cluster have high latency when responding to API requests. You review monitoring data and determine that there are not enough resources in the cluster to keep up with the load. You decide to add six more VMs to the cluster. What parameters will you need to specify when you issue the `cluster resize` command?
   A. Cluster size
   B. Cluster name
   C. Node pool name
   D. All of the above

7. You want to modify the number of pods in a cluster. What is the best way to do that?
   A. Modify pods directly
   B. Modify deployments
   C. Modify node pools directly
   D. Modify nodes

8. You want to see a list of deployments. Which option from the Kubernetes Engine navigation menu would you select?
   A. Clusters
   B. Storage
   C. Workloads
   D. Deployments

9. What actions are available from the Actions menu when viewing deployment details?
   A. Scale and Autoscale only
   B. Autoscale, Expose, and Rolling Update
   C. Add, Modify, and Delete
   D. None of the above

10. What is the command to list deployments from the command line?
    A. `gcloud container clusters list-deployments`
    B. `gcloud container clusters list`
    C. `kubectl get deployments`
    D. `kubectl deployments list`

11. What parameters of a deployment can be set in the Create Deployment page in Cloud Console?
    A. Container image
    B. Cluster name
    C. Application name
    D. All of the above

12. Where can you view a list of services when using Cloud Console?
    A. In the Deployment Details page
    B. In the Container Details page
    C. In the Cluster Details page
    D. None of the above

13. What `kubectl` command is used to add a service?
    A. `run`
    B. `start`
    C. `initiate`
    D. `deploy`

14. You are supporting machine learning engineers who are testing a series of classifiers. They have five classifiers, called ml-classifier-1, ml-classifier-2, etc. They have found that ml-classifier-3 is not functioning as expected and they would like it removed from the cluster. What would you do to delete a service called `ml-classifier-3`?
    A. Run the command `kubectl delete service ml-classifier-3`.
    B. Run the command `kubectl delete ml-classifier-3`.
    C. Run the command `gcloud service delete ml-classifier-3`.
    D. Run the command `gcloud container service delete ml-classifier-3`.

15. What service is responsible for managing container images?
    A. Kubernetes Engine
    B. Compute Engine
    C. Container Registry
    D. Container Engine

16. What command is used to list container images in the command line?
    A. `gcloud container images list`
    B. `gcloud container list images`
    C. `kubectl list container images`
    D. `kubectl container list images`

17. A data warehouse designer wants to deploy an extraction, transformation, and load process to Kubernetes. The designer provided you with a list of libraries that should be installed, including drivers for GPUs. You have a number of container images that you think may meet the requirements. How could you get a detailed description of each of those containers?
    A. Run the command `gcloud container images list details`.
    B. Run the command `gcloud container images describe`.
    C. Run the command `gcloud image describe`.
    D. Run the command `gcloud container describe`.

18. You have just created a deployment and want applications outside the cluster to have access to the services provided by the deployment. What do you need to do to the service?
    A. Give it a public IP address.
    B. Issue a `kubectl expose deployment` command.
    C. Issue a `gcloud expose deployment` command.
    D. Nothing, making it accessible must be done at the cluster level.

19. You have deployed an application to a Kubernetes cluster that processes sensor data from a fleet of delivery vehicles. The volume of incoming data depends on the number of vehicles making deliveries. The number of vehicles making deliveries is dependent on the number of customer orders. Customer orders are high during daytime hours, holiday seasons, and when major advertising campaigns are run. You want to make sure you have enough nodes running to handle the load, but you want to keep your costs down. How should you configure your Kubernetes cluster?
    A. Deploy as many nodes as your budget allows.
    B. Enable autoscaling.
    C. Monitor CPU, disk, and network utilization and add nodes as necessary.
    D. Write a script to run `gcloud` commands to add and remove nodes when peaks usually start and end, respectively.

20. When using Kubernetes Engine, which of the following might a cloud engineer need to configure?
    A. Nodes, pods, services, and clusters only
    B. Nodes, pods, services, clusters, and container images
    C. Nodes, pods, clusters, and container images only
    D. Pods, services, clusters, and container images only

Chapter

9

Computing with App Engine

THIS CHAPTER COVERS THE FOLLOWING OBJECTIVE OF THE GOOGLE ASSOCIATE CLOUD ENGINEER CERTIFICATION EXAM:

✓ **3.3 Deploying and implementing App Engine and Cloud Functions resources**

Official Google Cloud Certified: Associate Cloud Engineer: Study Guide
By Dan Sullivan

This chapter describes how to deploy App Engine Standard applications. We begin by reviewing the structure of an App Engine application and then examine how to specify an application configuration. Then, we will turn our attention to tuning App Engine applications through scaling and traffic splitting. We also discuss App Engine application versions along the way.

Google App Engine was originally designed to run applications in language-specific environments. Since the introduction of App Engine, Google has introduced App Engine Flexible, which can be used to deploy custom runtimes in containers. This chapter describes how to deploy applications to the original App Engine environment, known as App Engine Standard.

App Engine Components

App Engine Standard applications consist of four components:

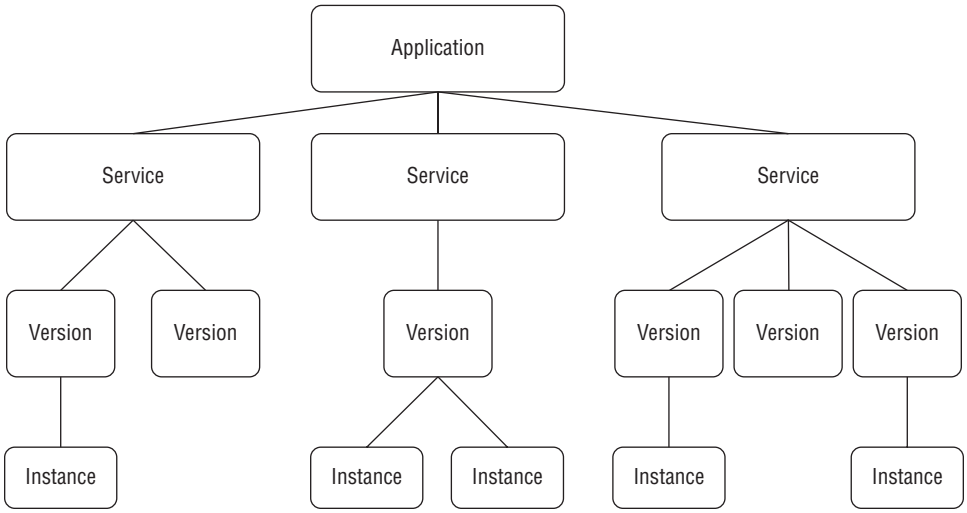
- Application
- Service
- Version
- Instance

An App Engine application is a high-level resource created in a project; that is, each project can have one App Engine application. All resources associated with an App Engine app are created in the region specified when the app is created.

Apps have at least one service, which is the code executed in the App Engine environment. Because multiple versions of an application's code base can exist, App Engine supports versioning of apps. A service can have multiple versions, and these are usually slightly different, with newer versions incorporating new features, bug fixes, and other changes relative to earlier versions. When a version executes, it creates an instance of the app.

Services are typically structured to perform a single function with complex applications made up of multiple services, known as *microservices*. One microservice may handle API requests for data access, while another microservice performs authentication and a third records data for billing purposes.

FIGURE 9.1 The component hierarchy of App Engine applications



Services are defined by their source code and their configuration file. The combination of those files constitutes a version of the app. If you slightly change the source code or configuration file, it creates another version. In this way, you can maintain multiple versions of your application at one time, which is especially helpful for testing new features on a small number of users before rolling the change out to all users. If bugs or other problems occur with a version, you can easily roll back to an early version. Another advantage of keeping multiple versions is that they allow you to migrate and split traffic, which we'll describe in more detail later in the chapter.

Deploying an App Engine Application

The Google Associate Cloud Engineer certification exam does not require engineers to write an application, but we are expected to know how to deploy one. In this section, you will download a Hello World example from Google and use it as a sample application that you will deploy. The app is written in Python, so you'll use the Python runtime in App Engine.

Deploying an App Using Cloud Shell and SDK

First, you will work in a terminal window using Cloud Shell, which you can start from the console by clicking the Cloud Shell icon. Make sure `gcloud` is configured to work with App Engine by using the following command:

```
gcloud components install app-engine-python
```

This will install or update the App Engine Python library as needed. If the library is up to date, you will receive a message saying that.

When you open Cloud Shell, you may have a directory named `python-docs-samples`. This contains a number of example applications, including the Hello World app we'll use. If you do not see this directory, you can download the Hello World app from Google using this:

```
git clone https://github.com/GoogleCloudPlatform/python-docs-samples
```

Next, change your working directory to the directory with the Hello World app, using the following:

```
cd python-docs-samples/appengine/standard/hello_world
```

If you list the files in the directory, you will see three files.

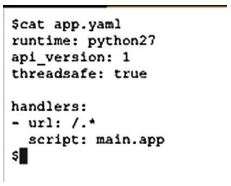
- `app.yaml`
- `main.py`
- `main_test.py`

Here you are primarily concerned with the `app.yaml` file. List the contents of this file using the following command:

```
cat app.yaml
```

This will show the configuration details, as shown in Figure 9.2.

FIGURE 9.2 The contents of an `app.yaml` file for a Python application



The app configuration file specifies the version of Python to use, the API version you are deploying, and a Python parameter called `threadsafe`, which is set to `true`. The last three lines specify the script to run, which in this case is `main.py`.

To deploy your app, you can use the following command:

```
gcloud app deploy app.yml
```

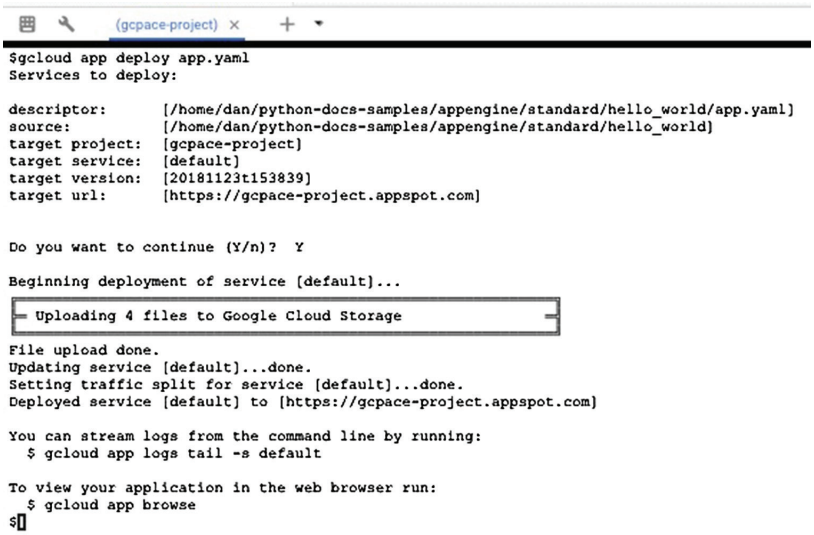
However, `app.yml` is the default, so if you are using that for the filename, you do not have to specify `app.yml` in the `deploy` command.

This command must be executed from the directory with the `app.yaml` file. The `gcloud app deploy` command has some optional parameters:

- `--version` to specify a custom version ID
- `--project` to specify the project ID to use for this app
- `--no-promote` to deploy the app without routing traffic to it

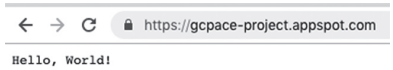
When you issue the `gcloud app deploy` command, you will see output such as in Figure 9.3.

FIGURE 9.3 The output of the `gcloud app deploy` command



You can see the output of the Hello World program by navigating in a browser to your project URL, such as `https://gcpace-project.appspot.com`. The project URL is the project name followed by `.appspot.com`. For example, Figure 9.4 shows the output.

FIGURE 9.4 The output of the Hello World app when running in App Engine Standard



You can also assign a custom domain if you would rather not use an `appspot.com` URL. You can do this from the Add New Custom domain function on the App Engine Settings page.

From the App Engine console, select Services from the left panel menu to see a listing of services, as in Figure 9.5.

FIGURE 9.5 A listing of services in the App Engine console

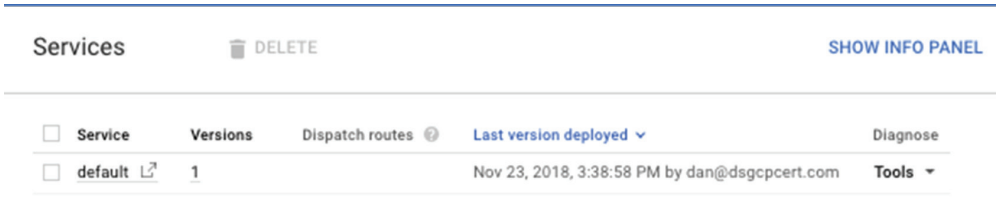


Figure 9.6 shows a list of versions. You can display this by selecting Versions from the left-panel menu.

FIGURE 9.6 A listing of versions in the App Engine console

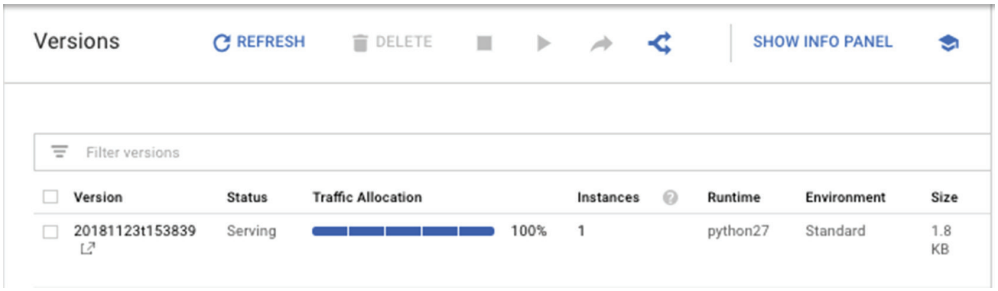
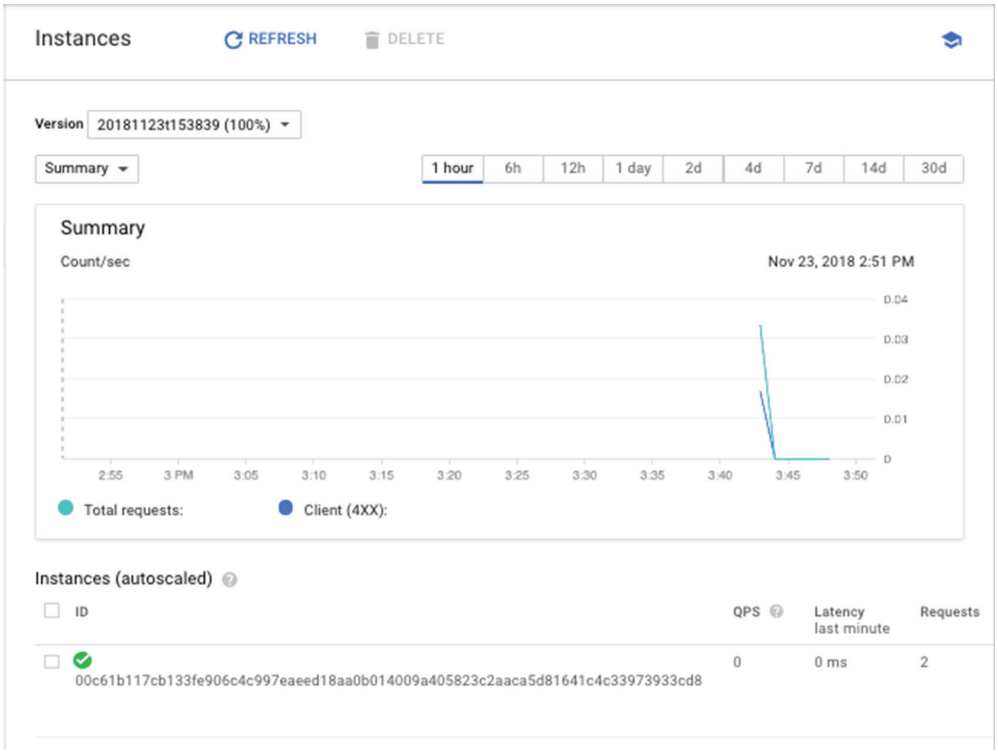


Figure 9.7 shows the instance performance details. You can display these details by selecting Instances from the left-panel menu. This information is useful for understanding the load on your application.

FIGURE 9.7 A listing of services in the App Engine console



You can stop serving versions using the `gcloud app versions stop` command and passing a list of versions to stop. For example, to stop serving versions named v1 and v2, use the following:

```
gcloud app versions stop v1 v2
```

You can also disable an entire application in the App Engine console, under Settings, by clicking the Disable App button.

Scaling App Engine Applications

Instances are created to execute an application on an App Engine managed server. App Engine can automatically add or remove instances as needed based on load. When instances are scaled based on load, they are called *dynamic* instances. These dynamic instances help optimize your costs by shutting down when demand is low.

Alternatively, you can configure your instances to be resident or running all the time. These are optimized for performance so users will wait less while an instance is started.

Your configuration determines whether an instance is resident or dynamic. If you configure autoscaling or basic scaling, then instances will be dynamic. If you configure manual scaling, then your instances will be resident.

To specify automatic scaling, add a section to `app.yaml` that includes the term `automatic_scaling` followed by key-value pairs of configuration options. These include the following:

- `target_cpu_utilization`
- `target_throughput_utilization`
- `max_concurrent_requests`
- `max_instances`
- `min_instances`
- `max_pending_latency`
- `min_pending_latency`

Target CPU Utilization Specifies the maximum CPU utilization that occurs before additional instances are started.

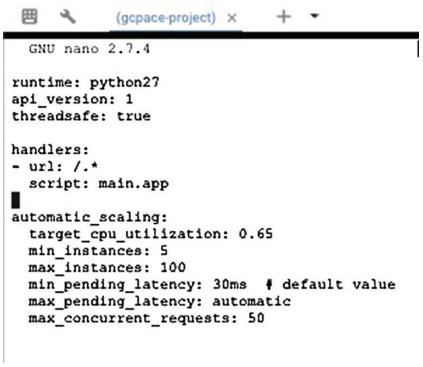
Target Throughput Utilization Specifies the maximum number of concurrent requests before additional instances are started. This is specified as a number between 0.5 and 0.95.

Maximum Concurrent Requests Specifies the max concurrent requests an instance can accept before starting a new instance. The default is 10; the max is 80.

Maximum and Minimum Instances Indicates the range of number of instances that can run for this application.

Maximum and Minimum Latency Indicates the maximum and minimum time a request will wait in the queue to be processed.

FIGURE 9.8 An example `app.yaml` for the Hello World app with autoscaling parameters



You can also use basic scaling to enable automatic scaling. The only parameters for basic scaling are `idle_timeout` and `max_instances`.

Figure 9.9 shows an example Hello World `app.yaml` file configured for basic scaling with a maximum of 10 instances and an `idle_timeout` of 20 minutes.

FIGURE 9.9 Example `app.yaml` using basic scaling

```
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /.*
  script: main.app

basic_scaling:
  max_instances: 10
  idle_timeout: 20m
```

If you prefer to use manual scaling because you need to control scaling, then specify the `manual_scaling` parameter and the number of instances to run. In the example in Figure 9.10, the Hello World app is configured to run with seven instances.

FIGURE 9.10 Example `app.yaml` using manual scaling

```
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /.*
  script: main.app

manual_scaling:
  instances: 7
```

Real World Scenario

Microservices vs. Monolithic Applications

Scalable applications are often written as collections of microservices. This has not always been the case. In the past, many applications were monolithic, or designed to include all functionality in a single compiled program or script. This may sound like a simpler, easy way to manage applications, but in practice it creates more problems than it solves.

- Any changes to the application require redeploying the entire application, which can take longer than deploying microservices. Developers tended to bundle changes before releasing them.
- If a bundled release had a bug in a feature change, then all feature changes would be rolled back when the monolithic application was rolled back.
- It was difficult to coordinate changes when teams of developers had to work with a single file or a small number of files of source code.

Microservices divide application code into single-function applications, allowing developers to change one service and roll it out without impacting other services. Source code management tools, like Git, make it easy for multiple developers to contribute components of a larger system by coordinating changes to source code files. This single-function code and the easy integration with other code promote more frequent updates and the ability to test new versions before rolling them out to all users at once.

Splitting Traffic between App Engine Versions

If you have more than one version of an application running, you can split traffic between the versions. App Engine provides three ways to split traffic: by IP address, by HTTP cookie, and by random selection. IP address splitting provides some stickiness, so a client is always routed to the same split, at least as long as the IP address does not change. HTTP cookies are useful when you want to assign users to versions. Random selection is useful when you want to evenly distribute workload.

When using IP address splitting, App Engine creates a hash, that is, a number generated based on an input string between 0 and 999, using the IP address of each version. This can create problems if users change IP address, such as if they start working with the app in the office and then switch to a network in a coffee shop. If state information is maintained in a version, it may not be available after an IP address change.

The preferred way to split traffic is with a cookie. When you use a cookie, the HTTP request header for a cookie named GOOGAPPUID contains a hash value between

0 and 999. With cookie splitting, a user will access the same version of the app even if the user's IP address changes. If there is no GOOGAPPUID cookie, then the traffic is routed randomly.

The command to split traffic is `gcloud app services set-traffic`. Here's an example:

```
gcloud app services set-traffic serv1 --splits v1=.4,v2=.6
```

This will split traffic with 40 percent going to version 1 of the service named serv1 and 60 percent going to version 2. If no service name is specified, then all services are split.

The `gcloud app services set-traffic` command takes the following parameters:

- `--migrate` indicates that App Engine should migrate traffic from the previous version to the new version.
- `--split-by` specifies how to split traffic using either IP or cookies. Possible values are `ip`, `cookie`, and `random`.

You can also migrate traffic from the console. Navigate to the Versions page and select the Migrate command.

Summary

App Engine Standard is a serverless platform for running applications in language-specific environments. As a cloud engineer, you are expected to know how to deploy and scale App Engine applications. App Engine applications consist of services, versions, and instances. You can have multiple versions running at one time. You can split traffic between versions and have all traffic automatically migrate to a new version. App Engine applications are configured through `app.yaml` configuration files. You can specify the language environment, scaling parameters, and other parameters to customize your deployment.

Exam Essentials

Know the structure of App Engine Standard applications. These consist of services, versions, and instances. Services usually provide a single function. Versions are different versions of code running in the App Engine environment. Instances are managed instances running the service.

Know how to deploy an App Engine app. This includes configuring the App Engine environment using the `app.yaml` file. Know that a project can have only one App Engine app at a time. Know how to use the `gcloud app deploy` command.

Know how to view the status of an application in the App Engine Console. This includes viewing a list of services, versions, and instances.

Understand the different scaling options. Three scaling options are autoscaling, basic scaling, and manual scaling. Only autoscaling and basic scaling are dynamic. Manual scaling creates resident instances. Autoscaling allows for more configuration options than basic scaling.

Know how to split traffic. Use the `gcloud app services set-traffic` command to split traffic. It takes a `--splits` parameter, which specifies the percent of traffic to route to each version.

Understand how to migrate traffic to a new version. You can migrate from the Versions page of the App Engine console or using the `--migrate` parameter with the `gcloud app services set-traffic` command.

Review Questions

You can find the answers in the Appendix.

1. You have designed a microservice that you want to deploy to production. Before it can be deployed, you have to review how you will manage the service lifecycle. The architect is particularly concerned about how you will deploy updates to the service with minimal disruption. What aspect of App Engine components would you use to minimize disruptions during updates to the service?
   - **A.** Services
   - **B.** Versions
   - **C.** Instance groups
   - **D.** Instances

2. You've just released an application running in App Engine Standard. You notice that there are peak demand periods in which you need up to 12 instances, but most of the time 5 instances are sufficient. What is the best way to ensure that you have enough instances to meet demand without spending more than you have to?
   - **A.** Configure your app for autoscaling and specify max instances of 12 and min instances of 5.
   - **B.** Configure your app for basic scaling and specify max instances of 12 and min instances of 5.
   - **C.** Create a cron job to add instances just prior to peak periods and remove instances after the peak period is over.
   - **D.** Configure your app for instance detection and do not specify a max or minimum number of instances.

3. In the hierarchy of App Engine components, what is the lowest-level component?
   - **A.** Application
   - **B.** Instance
   - **C.** Version
   - **D.** Service

4. What command should you use to deploy an App Engine app from the command line?
   - **A.** `gcloud components app deploy`
   - **B.** `gcloud app deploy`
   - **C.** `gcloud components instances deploy`
   - **D.** `gcloud app instance deploy`

5. You have deployed a Django 1.5 Python application to App Engine. This version of Django requires Python 3. For some reason, App Engine is trying to run the application using Python 2. What file would you check and possibly modify to ensure that Python 3 is used with this application?

   A. `app.config`

   B. `app.yaml`

   C. `services.yaml`

   D. `deploy.yaml`

6. You have several App Engine apps you plan to deploy from your project. What have you failed to account for in this design?

   A. App Engine only supports one app per project.

   B. App Engine only supports two apps per project.

   C. App Engine apps exist outside of projects.

   D. Nothing, this is a common pattern.

7. The latest version of your microservice code has been approved by your manager, but the product owner does not want the new features released until a press release is published. You'd like to get the code out but not expose it to customers. What is the best way to get the code out as soon as possible without exposing it to customers?

   A. Deploy with `gcloud app deploy --no-traffic.`

   B. Write a cron job to deploy after the press release is published.

   C. Deploy with `gcloud app deploy --no-promote.`

   D. Deploy as normal after the press release is published.

8. You have just deployed an app that hosts services that provide the current time in any time zone. The project containing the code is called `current-time-zone`, the service providing the user interface is called `time-zone-ui`, and the service performing the calculation is called `time-zone-calculate`. What is the URL where a user could find your service?

   A. `current-time-zone.appengine.com`

   B. `current-time-zone.appspot.com`

   C. `time-zone-ui.appspot.com`

   D. `time-zone-calculate.appspot.com`

9. You are concerned that as users make connections to your application, the performance will degrade. You want to make sure that more instances are added to your App Engine application when there are more than 20 concurrent requests. What parameter would you specify in `app.yaml`?

   A. `max_concurrent_requests`

   B. `target_throughput_utilization`

   C. `max_instances`

   D. `max_pending_latency`

10. What parameters can be configured with basic scaling?
    A. `max_instances` and `min_instances`
    B. `idle_timeout` and `min_instances`
    C. `idle_timeout` and `max_instances`
    D. `idle_timeout` and `target_throughput_utilization`

11. The `runtime` parameter in `app.yaml` is used to specify what?
    A. The script to execute
    B. The URL to access the application
    C. The language runtime environment
    D. The maximum time an application can run

12. What are the two kinds of instances available in App Engine Standard?
    A. Resident and dynamic
    B. Persistent and dynamic
    C. Stable and dynamic
    D. Resident and nonresident

13. You work for a startup, and costs are a major concern. You are willing to take a slight performance hit if it will save you money. How should you configure the scaling for your apps running in App Engine?
    A. Use dynamic instances by specifying autoscaling or basic scaling.
    B. Use resident instances by specifying autoscaling or basic scaling.
    C. Use dynamic instances by specifying manual scaling.
    D. Use resident instances by specifying manual scaling.

14. A team of developers has created an optimized version of a service. This should run 30 percent faster in most cases. They want to roll it out to all users immediately, but you are concerned that the substantial changes need to be released slowly in case there are significant bugs. What can you do to allocate some users to the new version without exposing all users to it?
    A. Issue the command `gcloud app services set-traffic.`
    B. Issue the command `gcloud instances services set-traffic.`
    C. Issue the command `gcloud app set-traffic.`
    D. Change the target IP address of the service for some clients.

15. What parameter to `gcloud app services set-traffic` is used to specify the method to use when splitting traffic?
    A. `--split-traffic`
    B. `--split-by`
    C. `--traffic-split`
    D. `--split-method`

16. What parameter to `gcloud app services set-traffic` is used to specify the percentage of traffic that should go to each instance?
    A. `--split-by`
    B. `--splits`
    C. `--split-percent`
    D. `--percent-split`

17. You have released a new version of a service. You have been waiting for approval from the product manager to start sending traffic to the new version. You get approval to route traffic to the new version. What parameter to `gcloud app services set-traffic` is used to specify that traffic should be moved to a newer version of the app?
    A. `--move-to-new`
    B. `--migrate-to-new`
    C. `--migrate`
    D. `--move`

18. The status of what components can be viewed in the App Engine console?
    A. Services only
    B. Versions only
    C. Instances and versions
    D. Services, versions, and instances

19. What are valid methods for splitting traffic?
    A. By IP address only
    B. By HTTP cookie only
    C. Randomly and by IP address only
    D. By IP address, HTTP cookies, and randomly

20. What is the name of the cookie used by App Engine when cookie-based splitting is used?
    A. GOOGID
    B. GOOGAPPUID
    C. APPUID
    D. UIDAPP

Chapter

10

Computing with Cloud Functions

THIS CHAPTER COVERS THE FOLLOWING OBJECTIVES OF THE GOOGLE ASSOCIATE CLOUD ENGINEER CERTIFICATION EXAM:

✓ **3.3 Deploying and implementing App Engine and Cloud Functions resources**

Official Google Cloud Certified: Associate Cloud Engineer: Study Guide
By Dan Sullivan

In this chapter, we describe the purpose of Cloud Functions as well as how to implement and deploy the functions. We will use examples of the functions written in Python. If you are unfamiliar with Python, that should not dissuade you from following along. The important details of Python functions will be explained. You will learn how to use the Cloud Console and `gcloud` commands to create and manage Cloud Functions.

This chapter covers Cloud Functions only. App Engine is covered in Chapter 9.

Introduction to Cloud Functions

Cloud Functions is a serverless compute service provided by Google Cloud Platform (GCP). Cloud Functions is similar to App Engine in that they are both serverless. A primary difference, though, is that App Engine supports multiple services organized into a single application, while Cloud Functions supports individual services that are managed and operate independently of other services.

App Engine is a good serverless option for web applications that have a front-end user interface running in one service, a set of APIs running in one or more other services, and business logic running in another service. The services together make up the application, so it makes sense to treat them as a single managed unit.

Not all computing requirements need multiple services. For example, your department may upload a daily data extract from a database, which is then loaded into an enterprise data warehouse. If the data extract files are loaded into Cloud Storage, then you could use a function to perform preprocessing, such as verifying the file is the right format and meets other business rules. If the file passes checks, a message is written to a Pub/Sub topic, a messaging service in GCP, which is read by the data warehouse load process. Cloud Functions allows developers to decouple the initial data quality check from the rest of the extraction, transformation, and load process.

There are limits to Cloud Functions. By default, the functions will time out after one minute, although you can set the timeout for as long as nine minutes.

Events, Triggers, and Functions

There are some terms you need to know before going any further into Cloud Functions:

- Events
- Triggers
- Functions

Events are a particular action that happens in Google Cloud, such as a file is uploaded to Cloud Storage or a message (called a *topic*) is written to a Pub/Sub message queue. There are different kinds of actions associated with each of the events. Currently, GCP supports events in five categories:

- Cloud Storage
- Cloud Pub/Sub
- HTTP
- Firebase
- Stackdriver Logging

Events in Cloud Storage include uploading, deleting, and archiving a file. Cloud Pub/Sub has an event for publishing a message. The HTTP type of event allows developers to invoke a function by making an HTTP request using POST, GET, PUT, DELETE, and OPTIONS calls. Firebase events are actions taken in the Firebase database, such as database triggers, remote configuration triggers, and authentication triggers. You can set up a function to respond to a change in Stackdriver Logging by forwarding log entries to a Pub/Sub topic and triggering a response from there.

For each of the Cloud Functions–enabled events that can occur, you can define a trigger. A *trigger* is a way of responding to an event.

Triggers have an associated *function*. The function is passed arguments with data about the event. The function executes in response to the event.

Runtime Environments

Functions run in their own environment. Each time a function is invoked, it is run in a separate instance from all other invocations. There is no way to share information between invocations of functions using only Cloud Functions. If you need to coordinate the updating of data, such as keeping a global count, or need to keep information about the state of functions, such as the name of the last event processed, then you should use a database, such as Cloud Datastore, or a file in Cloud Storage.

Google currently supports three runtime environments:

- Python 3
- Node.js 6
- Node.js 8

Let's walk through an example function. You want to record information about file uploads to a particular bucket in Cloud Storage. You can do this by writing a Python function that receives information about an event and then issues print commands to send a description of that data to a log file. Here is the Python code:

```
def cloud_storage_function_test(event_data, event_context):
print('Event ID: {}'.format(event_context.event_id))
      print('Event type: {}'.format(event_context.event_type))
      print('File: {}'.format(event_data['name']))
```

The first line begins the creation of a function called `cloud_storage_function_test`. It takes two arguments, `event_data` and `event_context`. These are Python data structures with information about the object of the event and about the event itself. The next three lines print the values of the `event_id`, `event_type`, and name of the file. Since this code will be run as a function, and not interactively, the output of a print statement will go to the function's log file.

Python functions should be saved in a file called `main.py`.

Real World Scenario

Making Documents Searchable

Litigation, or lawsuits, between businesses often involve reviewing a large volume of documents. Electronic documents may be in readily searchable formats, such as Microsoft Word documents or PDF files. Others may be scanned images of paper documents. In that case, the file needs to be preprocessed using an optical character recognition (OCR) program.

Functions can be used to automate the OCR process. When a file is uploaded, a Cloud Storage trigger fires and invokes a function. The function determines whether the file is in a searchable format or needs to be preprocessed by the OCR program. If the file does require OCR processing, the function writes the location of the file into a Pub/Sub topic.

A second function is bound to a new message event. When a file location is written in a message, the function calls the OCR program to scan the document and produce a searchable version of the file. That searchable version is written to a Cloud Storage bucket, where it can be indexed by the search tool along with other searchable files.

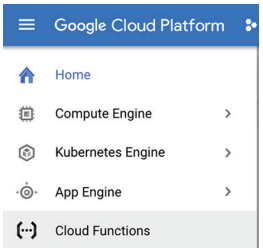
Cloud Functions Receiving Events from Cloud Storage

Cloud Storage is GCP's object storage. This service allows you to store files in containers known as *buckets*. We will go into more detail about Cloud Storage in Chapter 11, but for this chapter you just need to understand that Cloud Storage uses buckets to store files. When files are created, deleted, or archived, or their metadata changes, an event can invoke a function. Let's go through an example of deploying a function for Cloud Storage Events using Cloud Console and `gcloud` commands in Cloud SDK and Cloud Shell.

Deploying a Cloud Function for Cloud Storage Events Using Cloud Console

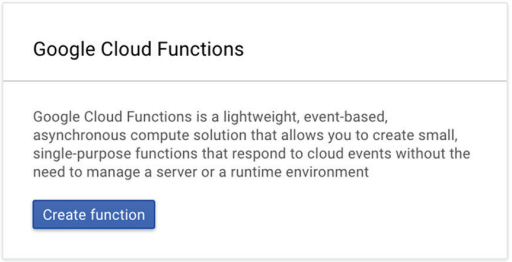
To create a function using Cloud Console, select the Cloud Function options from the vertical menu in the console, as in Figure 10.1.

FIGURE 10.1 Opening the Cloud Functions console



In the Cloud Functions console, you may be prompted to enable the Cloud Functions API if it is not already enabled. After the Cloud Functions API is enabled, you will have the option to create a new function, as shown in Figure 10.2.

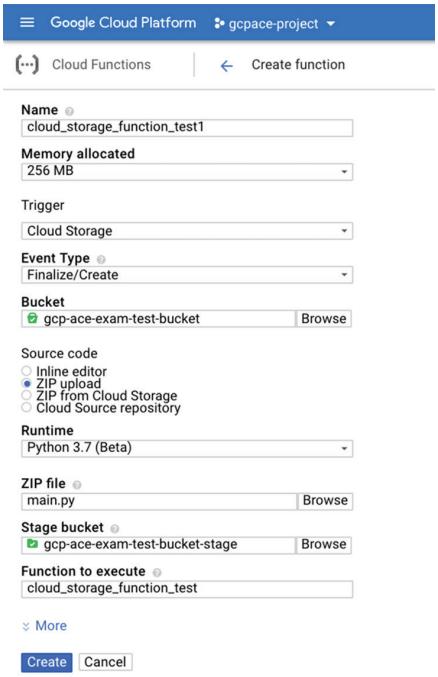
FIGURE 10.2 The prompt to create a new function in Cloud Console



When you create a new function in the console, a form such as in Figure 10.3 appears. In Figure 10.3, the options, which have been filled in, include:

- Function name
- Memory allocated for the function
- Trigger
- Event type
- Source of the function code
- Runtime
- Source code
- Python, Go or Node.js function to execute

FIGURE 10.3 Creating a function in the console



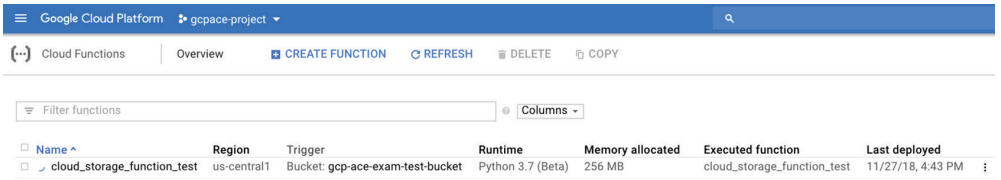
In the following example, we are uploading a file containing the function code. The contents of that file are as follows:

```
def cloud_storage_function_test(event_data, event_context):
  print('Event ID: {}'.format(event_context.event_id))
  print('Event type: {}'.format(event_context.event_type))
  print('File: {}'.format(event_data['name']))
```

The function name is the name GCP will use to refer to this function. Memory Allocated is the amount of memory that will be available to the function. Memory options range from 128MB to 2GB. Trigger is one of the defined triggers, such as HTTP, Cloud Pub/Sub, and Cloud Storage. There are several options for specifying where to find the source code, including uploading it, getting it from Cloud Storage or a Cloud Source repository, or entering the code in an editor. Runtime indicates which runtime to use to execute the code. The editor is where you can enter function code. Finally, the function to execute is the name of the function in the code that should run when the event occurs.

After a function is created, you will see a list of functions in the Cloud Functions console, such as in Figure 10.4.

FIGURE 10.4 List of functions in the console



Note that at the top of the list of functions there is the option to delete a function.

Deploying a Cloud Function for Cloud Storage Events Using gcloud Commands

The first step to using `gcloud` commands for Cloud Functions is to make sure you have the latest version of the commands installed. You can update standard `gcloud` commands using this:

```
gcloud components update
```

The Python commands are in beta at the time of writing, so you can ensure that they are installed with the following command:

```
gcloud components install beta
```

Let's assume you have created a Cloud Storage bucket called `gcp-ace-exam-test-bucket`. You can deploy a function using the `gcloud functions deploy` command. This command takes the name of a function as its argument. There are also three parameters you will need to pass in:

- `runtime`
- `trigger-resource`
- `trigger-event`

`runtime` indicates whether you are using Python 3.7 Node.js 6, or Node.js 8. `trigger-resources` indicates the bucket name associated with the trigger. `trigger-event` is the kind of event that will trigger the execution of the function. The possible options are as follows:

- `google.storage.object.finalize`
- `google.storage.object.delete`
- `google.storage.object.archive`
- `google.storage.object.metadataUpdate`

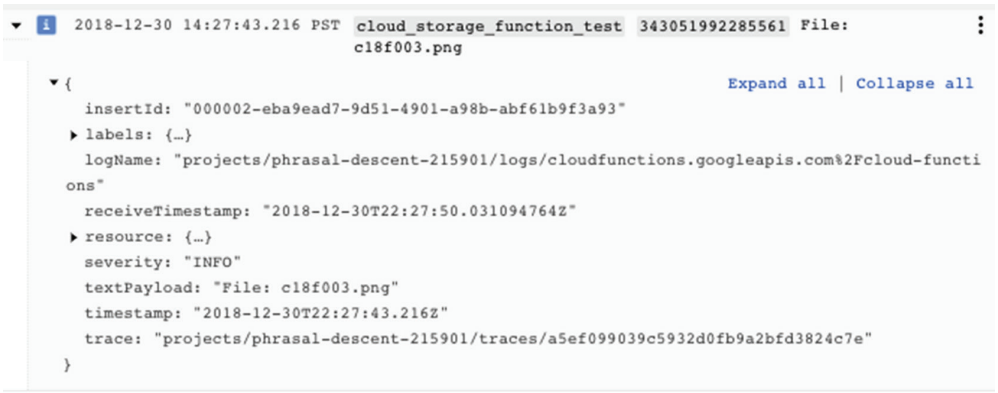
`finalize` is the term used to describe when a file is fully uploaded.

Whenever a new file is uploaded to the bucket called `gcp-ace-exam-test-bucket`, we want to execute the `cloud_storage_function_test`. We accomplish this by issuing the following command:

```
gcloud functions deploy cloud_storage_function_test \
        --runtime python37 \
        --trigger-resource gcp-ace-exam-test-bucket \
        --trigger-event google.storage.object.finalize
```

When you upload a file to the bucket, the function will execute and create a log message similar to what is shown in Figure 10.5.

FIGURE 10.5 Example log message generated by the `cloud_storage_function_test` function



When you are done with the function and want to delete it, you can use the `gcloud` function's `delete` command, like so:

```
gcloud functions delete cloud_storage_function_test
```

Cloud Functions Receiving Events from Pub/Sub

A function can be executed each time a message is written to a Pub/Sub topic. You can use Cloud Console or `gcloud` commands to deploy functions triggered by a Cloud Pub/Sub event.

Deploying a Cloud Function for Cloud Pub/Sub Events Using Cloud Console

Assume you are using a function similar to one used in the previous Cloud Storage example. This time we'll call the function `pub_sub_function_test`.

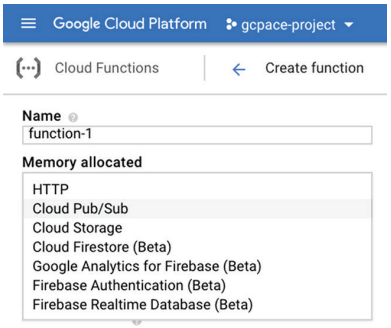
To create a function using Cloud Console, select the Cloud Function options from the vertical menu in the console. In the Cloud Functions console, you may be prompted to enable the Cloud Functions API if it is not already enabled. After the Cloud Functions API is enabled, you will have the option to create a new function. When creating a function, you will need to specify several parameters, including the cloud function name, memory allocated, event type, and source code. Here is the source code for `pub_sub_function_test`:

```
def pub_sub_function_test(event_data, event_context):
    import base64
print('Event ID: {}'.format(event_context.event_id))
      print('Event type: {}'.format(event_context.event_type))
      if 'name' in event_data:
      name = base64.b64decode(event_data['name']).decode('utf-8')
      print('Message name: {}'.format(event_data['name']))
```

This function prints the event ID and event type associated with the message. If the event data has a key-value pair with the key of `name`, then the function will also print the name in the message. Note that this function has an import statement and uses a function called `base64.b64decode`. This is because messages in Pub/Sub are encoded to allow for binary data in a place where text data is expected, and the `base64.b64decode` function is used to convert it to a more common text encoding called UTF-8.

The code is deployed in the same way as the previous Cloud Storage example with two exceptions. Instead of selecting a Cloud Storage trigger, choose Cloud Pub/Sub from the list of triggers, as shown in Figure 10.6. You can also specify the name of the Cloud Pub/Sub topic after specifying this is a Cloud Pub/Sub trigger. If the topic does not exist, it will be created.

FIGURE 10.6 Selecting a trigger from options in Cloud Console



Deploying a Cloud Function for Cloud Pub/Sub Events Using gcloud Commands

As with functions for Cloud Storage, if you are deploying Cloud Functions, it's a good idea to use the latest `gcloud` commands by issuing this:

```
gcloud components update
```

If you are using Python, you will want to install beta `gcloud` components as well:

```
gcloud components install beta
```

To deploy this function, you use the `gcloud functions deploy` command. When deploying a Cloud Pub/Sub function, you specify the name of the topic that will contain messages that will trigger the function. Like deploying for Cloud Storage, you have to specify the runtime environment you want to use. Here's an example:

```
gcloud functions deploy pub_sub_function_test --runtime python37 --trigger-topic
gcp-ace-exam-test-topic
```

You can delete this function using the `gcloud functions delete` command. Here's an example:

```
gcloud functions delete pub_sub_function_test
```

Summary

In this chapter, we worked with Cloud Functions and saw how to implement and deploy functions. We used examples of functions written in Python, but they could have been written in Node.js as well. Functions can be created using either the Google Cloud Console or

the command line. To use Cloud Functions, it is important to understand the relationship between events, triggers, and functions. Events are actions that happen in the cloud. Different services have different types of events. Triggers are how you indicate you want to execute a function when an event occurs. Functions refer to the code that is executed when an event occurs that has a trigger defined for it.

Exam Essentials

Know the relationship between events, triggers, and functions. Events are actions that happen, such as when a file is uploaded to Cloud Storage or a message is written to a Cloud Pub/Sub topic. Triggers are declarations that an action should be taken when an event occurs. Functions associated with triggers define what actions are taken when an event occurs.

Know when to use Cloud Functions versus App Engine applications. Cloud Functions is a service that supports single-purpose functions that respond to events in the cloud. App Engine is also a serverless computing option, but it is used to deploy multifunction applications, including those that users interact with directly.

Know the runtimes supported in Cloud Functions. Cloud Functions supports the following runtimes: Node.js 6, Node.js 8, and Python 3.

Know the parameters for defining a cloud function on a Cloud Storage event. Parameters for Cloud Storage include the following:

Cloud function name

Memory allocated for the function

Trigger

Event type

Source of the function code

Runtime

Source code

Name of the Python or Node.js function to execute

Know the parameters for defining a Cloud Function on a Cloud Pub/Sub event. Parameters for Pub/Sub include the following:

Cloud function name

Memory allocated for the function

Trigger

Topic

Source of the function code

Runtime

Source code

Name of the Python or Node.js function to execute

Know the `gcloud` commands for working with Cloud Functions. These include the following:

`gcloud` functions deploy

`gcloud` functions delete

Review Questions

You can find the answers in the Appendix.

1. A product manager is proposing a new application that will require several backend services, three business logic services, and access to relational databases. Each service will provide a single function, and it will require several of these services to complete a business task. Service execution time is dependent on the size of input and is expected to take up to 30 minutes in some cases. Which GCP product is a good serverless option for running this related service?
   - **A.** Cloud Functions
   - **B.** Compute Engine
   - **C.** App Engine
   - **D.** Cloud Storage

2. You have been asked to deploy a cloud function to reformat image files as soon as they are uploaded to Cloud Storage. You notice after a few hours that about 10 percent of the files are not processed correctly. After reviewing the files that failed, you realize they are all substantially larger than average. What could be the cause of the failures?
   - **A.** There is a syntax error in the function code.
   - **B.** The wrong runtime was selected.
   - **C.** The timeout is too low to allow enough time to process large files.
   - **D.** There is a permissions error on the Cloud Storage bucket containing the files.

3. When an action occurs in GCP, such as a file being written to Cloud Storage or a message being added to a Cloud Pub/Sub topic, that action is called what?
   - **A.** An incident
   - **B.** An event
   - **C.** A trigger
   - **D.** A log entry

4. All of the following generate events that can be triggered using Cloud Functions, except which one?
   - **A.** Cloud Storage
   - **B.** Cloud Pub/Sub
   - **C.** SSL
   - **D.** Firebase

5. Which runtimes are supported in Cloud Functions?
   A. Node.js 5, Node.js 6, and Node.js 8
   B. Node.js 8, Python, and Go
   C. Node.js 6, Node.js 8, and Python
   D. Node.js 8, Python, and Go

6. An HTTP trigger can be invoked by making a request using which of the following?
   A. GET only
   B. POST and GET only
   C. DELETE, POST, and GET
   D. DELETE, POST, REVISE, and GET

7. What types of events are available to Cloud Functions working with Cloud Storage?
   A. Upload or finalize and delete only
   B. Upload or finalize, delete, and list only
   C. Upload or finalize, delete, and metadata update only
   D. Upload or finalize, delete, metadata update, and archive

8. You are tasked with designing a function to execute in Cloud Functions. The function will need more than the default amount of memory and should be applied only when a finalize event occurs after a file is uploaded to Cloud Storage. The function should only apply its logic to files with a standard image file type. Which of the following required features cannot be specified in a parameter and must be implemented in the function code?
   A. Cloud function name
   B. Memory allocated for the function
   C. File type to apply the function to
   D. Event type

9. How much memory can be allocated to a Cloud Function?
   A. 128MB to 256MB
   B. 128MB to 512MB
   C. 128MB to 1GB
   D. 128MB to 2GB

10. How long can a cloud function run by default before timing out?
    A. 30 seconds
    B. 1 minute
    C. 9 minutes
    D. 20 minutes

11. You want to use the command line to manage Cloud Functions that will be written in Python. In addition to running the `gcloud components update` command, what command should you run to ensure you can work with Python functions?
    A. `gcloud component install`
    B. `gcloud components install beta`
    C. `gcloud components install python`
    D. `gcloud functions install beta`

12. You want to create a cloud function to transform audio files into different formats. The audio files will be uploaded into Cloud Storage. You want to start transformations as soon as the files finish uploading. Which trigger would you specify in the cloud function to cause it to execute after the file is uploaded?
    A. `google.storage.object.finalize`
    B. `google.storage.object.upload`
    C. `google.storage.object.archive`
    D. `google.storage.object.metadataUpdate`

13. You are defining a cloud function to write a record to a database when a file in Cloud Storage is archived. What parameters will you have to set when creating that function?
    A. `runtime` only
    B. `trigger-resource` only
    C. `runtime`, `trigger-resource`, `trigger-event` only
    D. `runtime`, `trigger-resource`, `trigger-event`, `file-type`

14. You'd like to stop using a cloud function and delete it from your project. Which command would you use from the command line to delete a cloud function?
    A. `gcloud functions delete`
    B. `gcloud components function delete`
    C. `gcloud components delete`
    D. `gcloud delete functions`

15. You have been asked to deploy a cloud function to work with Cloud Pub/Sub. As you review the Python code, you notice a reference to a Python function called `base64.b64decode`. Why would a decode function be required in a Pub/Sub cloud function?
    A. It's not required and should not be there.
    B. Messages in Pub/Sub topics are encoded to allow binary data to be used in places where text data is expected. Messages need to be decoded to access the data in the message.
    C. It is required to add padding characters to the end of the message to make all messages the same length.
    D. The decode function maps data from a dictionary data structure to a list data structure.

16. Which of these commands will deploy a Python cloud function called `pub_sub_function_test`?
    A. `gcloud functions deploy pub_sub_function_test`
    B. `gcloud functions deploy pub_sub_function_test --runtime python37`
    C. `gcloud functions deploy pub_sub_function_test --runtime python37 --trigger-topic gcp-ace-exam-test-topic`
    D. `gcloud functions deploy pub_sub_function_test --runtime python --trigger-topic gcp-ace-exam-test-topic`

17. When specifying a Cloud Storage cloud function, you have to specify an event type, such as `finalize`, `delete`, or `archive`. When specifying a Cloud Pub/Sub cloud function, you do not have to specify an event type. Why is this the case?
    A. Cloud Pub/Sub does not have triggers for event types.
    B. Cloud Pub/Sub has triggers on only one event type, when a message is published.
    C. Cloud Pub/Sub determines the correct event type by analyzing the function code.
    D. The statement in the question is incorrect; you do have to specify an event type with Cloud Pub/Sub functions.

18. Your company has a web application that allows job seekers to upload résumé files. Some files are in Microsoft Word, some are PDFs, and others are text files. You would like to store all résumés as PDFs. How could you do this in a way that minimizes the time between upload and conversion and with minimal amounts of coding?
    A. Write an App Engine application with multiple services to convert all documents to PDF.
    B. Implement a Cloud Function on Cloud Storage to execute on a finalize event. The function checks the file type, and if it is not PDF, the function calls a PDF converter function and writes the PDF version to the bucket that has the original.
    C. Add the names of all files to a Cloud Pub/Sub topic and have a batch job run at regular intervals to convert the original files to PDF.
    D. Implement a Cloud Function on Cloud Pub/Sub to execute on a finalize event. The function checks the file type, and if it is not PDF, the function calls a PDF converter function and writes the PDF version to the bucket that has the original.

19. What are options for uploading code to a cloud function?
    A. Inline editor
    B. Zip upload
    C. Cloud source repository
    D. All of the above

20. What type of trigger allows developers to use HTTP POST, GET, and PUT calls to invoke a cloud function?
    A. HTTP
    B. Webhook
    C. Cloud HTTP
    D. None of the above

Chapter 11

Planning Storage in the Cloud

THIS CHAPTER COVERS THE FOLLOWING OBJECTIVES OF THE GOOGLE ASSOCIATE CLOUD ENGINEER CERTIFICATION EXAM:

✓ **2.3 Planning and configuring data storage options**

Official Google Cloud Certified: Associate Cloud Engineer: Study Guide
By Dan Sullivan

As a cloud engineer, you will have to understand the various storage options provided in Google Cloud Platform (GCP). You will be expected to choose the appropriate option for a given use case while knowing the relative trade-offs, such as having access to SQL for a query language versus the ability to store and query petabytes of data streaming into your database.

Unlike most other chapters in the book, this chapter focuses more on storage concepts than on performing specific tasks in GCP. The material here will help you answer questions about choosing the best storage solution. Chapter 12 will provide details on deploying and implementing data solutions.

To choose between storage options, it helps to understand how storage solutions vary by:

- Time to access data
- Data model
- Other features, such as consistency, availability, and support for transactions

This chapter includes guidelines for choosing storage solutions for different kinds of requirements.

Types of Storage Systems

A main consideration when you choose a storage solution is the time in which the data must be accessed. At one extreme, data in an L1 cache on a CPU chip can be accessed in 0.5 nanoseconds (ns). At the other end of the spectrum some services can require hours to return data files. Most storage requirements fall between these extremes.

Nanoseconds, Milliseconds and Microseconds

Some storage systems operate at speeds as unfamiliar to us as what happens under an electron microscope. One second is an extremely long time when talking about the time it takes to access data in-memory or on disk. We measure time to access, or "latency," with three units of measure.

- Nanosecond (ns), which is 10^{-9} second
- Microsecond (μs), which is 10^{-6} second
- Millisecond (ms), which is 10^{-3} second

Note, the number 10^{-3} is in scientific notation and means 0.001 second. Similarly, 10^{-6} is the same as 0.000001, and 10^{-9} is the same as 0.000000001 second.

Another consideration is persistence. How durable is the data stored in a particular system? Caches offer the lowest latency for accessing data, but this type of volatile data exists only as long as power is supplied to memory. Shut down the server and away goes your data. Disk drives have higher durability rates, but they can fail. Redundancy helps here. By making copies of data and storing them on different servers, in different racks, in different zones, and in different regions, you reduce the risk of losing data due to hardware failures.

GCP has several storage services, including the following:

- A managed Redis cache service for caching
- Persistent disk storage for use with VMs
- Object storage for shared access to files across resources
- Archival storage for long-term, infrequent access requirements

Cache

A cache is an in-memory data store designed to provide applications with sub millisecond access to data. Its primary advantage over other storage systems is its low latency. Caches are limited in size by the amount of memory available, and if the machine hosting the cache shuts down, then the contents of the cache are lost. These are significant limitations, but in some use cases, the benefits of fast access to data outweigh the disadvantages.

MemoryStore

GCP offers Memorystore, a managed Redis service. Redis is a widely used open source cache. Since Memorystore is protocol-compatible with Redis, tools and applications written to work with Redis should work with Memorystore.

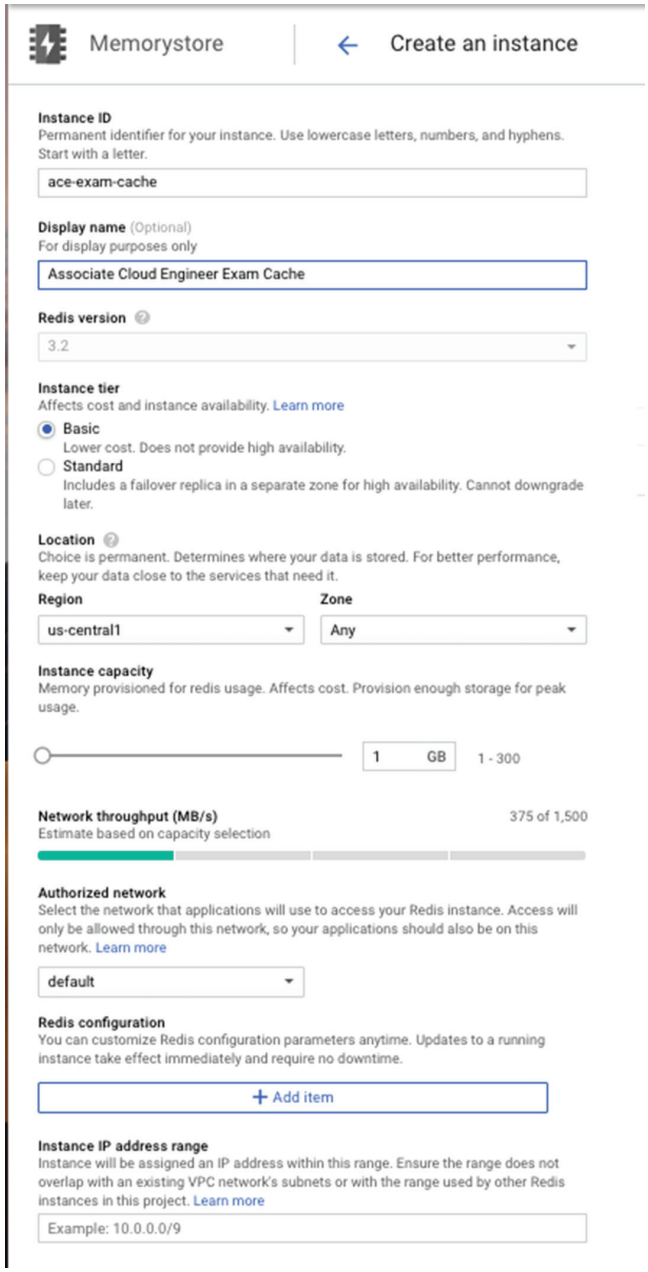
Caches are usually used with an application that cannot tolerate long latencies when retrieving data. For example, an application that has to read from a hard disk drive might have to wait 80 times longer than if the data were read from an in-memory cache. Application developers can use caches to store data that is retrieved from a database and then retrieved from the cache instead of the disk the next time that data is needed.

When you use Memorystore, you create instances that run Redis. The instance is configured with 1GB to 300GB of memory. It can also be configured for high availability, in which case Memorystore creates failover replicas.

Configuring Memorystore

Memorystore caches can be used with applications running in Compute Engine, App Engine, and Kubernetes Engine. Figure 11.1 shows the parameters used to configure Memorystore. You can navigate to this form by choosing Memorystore from the main console menu and then selecting the option to create a Redis instance.

FIGURE 11.1 Configuration parameters for a Memorystore cache



To configure a Redis cache in Memorystore, you will need to specify an instance ID, a display name, and a Redis version. Currently only Redis 3.2 is supported. You can choose to have a replica in a different zone for high availability by selecting the Standard instance tier. The Basic instance tier does not include a replica but costs less.

You will need to specify a region and zone along with the amount of memory to dedicate to your cache. The cache can be 1GB to 300GB in size. The Redis instance will be accessible from the default network unless you specify a different network. (See Chapters 14 and 15 for more on networks in GCP). The advanced options for Memorystore allow you to assign labels and define an IP range from which the IP address will be assigned.

Persistent Storage

In GCP, persistent disks provide durable block storage. Persistent disks can be attached to VMs in Google Compute Engine (GCE) and Google Kubernetes Engine (GKE). Since persistent disks are block storage devices, you can create file systems on these devices. Persistent disks are not directly attached to physical servers hosting your VMs but are network accessible. VMs can have locally attached solid-state drives (SSDs), but the data on those drives is lost when the VM is terminated. The data on persistent disks continues to exist after VMs are shut down and terminated. Persistent disks exist independently of virtual machines; local attached SSDs do not.

Features of Persistent Disks

Persistent disks are available in SSD and hard disk drive (HDD) configurations. SSDs are used when high throughput is important. SSDs provide consistent performance for both random access and sequential access patterns. HDDs have longer latencies but cost less, so HDDs are a good option when storing large amounts of data and performing batch operations that are less sensitive to disk latency than interactive applications. Hard drive–backed persistent disks can perform 0.75 read input output operations per second (IOPS) per gigabyte and 1.5 write IOPS per gigabyte, while network-attached SSDs can perform 30 read and write IOPS per gigabyte. Locally attached SSDs can achieve read IOPS rates between 266 and 453 per gigabyte and write IOPS rates between 186 and 240 per gigabyte.

Persistent disks can be mounted on multiple VMs to provide multireader storage. Snapshots of disks can be created in minutes, so additional copies of data on a disk can be distributed for use by other VMs. If a disk created from a snapshot is mounted to a single VM, it can support both read and write operations.

The size of persistent disks can be increased while mounted to a VM. If you do resize a disk, you may need to perform operating system commands to make that additional space accessible to the file system. Both SSD and HDD disks can be up to 64TB.

Persistent disks automatically encrypt data on the disk.

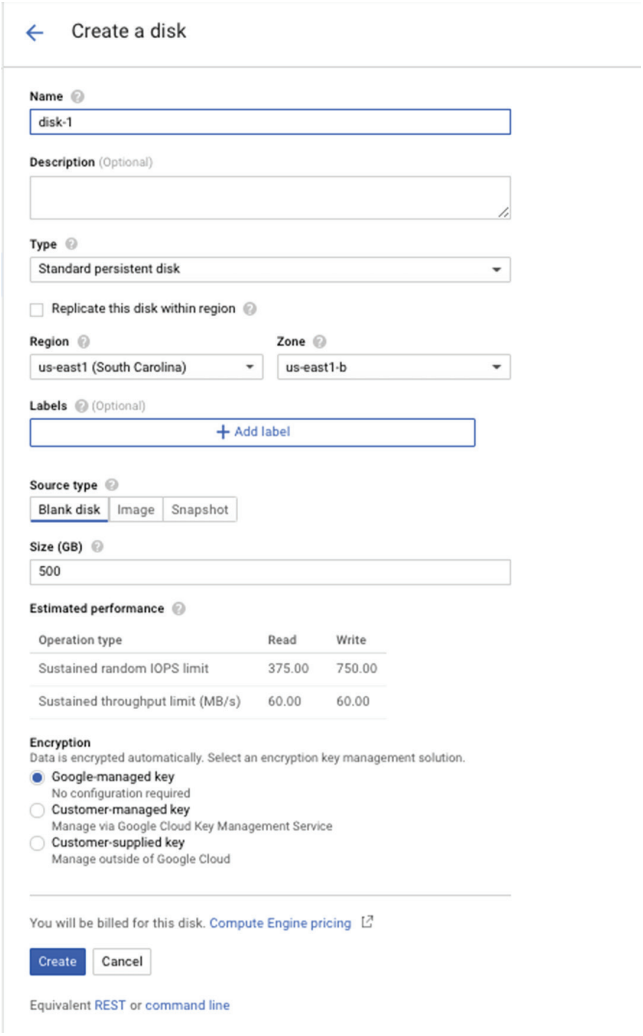
When planning your storage options, you should also consider whether you want your disks to be zonal or regional. Zonal disks store data across multiple physical drives in a single zone. If the zone becomes inaccessible, you will lose access to your disks. Alternatively, you

could use regional persistent disks, which replicate data blocks across two zones within a region but is more expensive than zonal storage.

Configuring Persistent Disks

You can create and configure persistent disks from the console by navigating to Compute Engine and selecting Disks. From the Disk page, click Create a Disk to display a form like that in Figure 11.2.

FIGURE 11.2 Form to create a persistent disk



You will need to provide a name for the disk, but the description is optional. There are two types of disk: standard, and SSD persistent disk. For higher availability, you can have a replica created within the region. You will need to specify a region and zone. Labels are optional, but recommended to help keep track of each disk's purpose.

Persistent disks can be created blank or from an image or snapshot. Use the image option if you want to create a persistent boot disk. Use a snapshot if you want to create a replica of another disk.

When you store data at rest in GCP, it is encrypted by default. When creating a disk, you can choose to have Google manage encryption keys, in which case no additional configuration is required. You could use GCP's Cloud Key Management Service to manage keys yourself and store them in GCP's key repository. Choose the customer-managed MKey option for this. You will need to specify the name of a key you have created in Cloud Key Management Service. If you create and manage keys using another key management system, then select customer-supplied SKey. You will have to enter the key into the form if you choose the customer-supplied key option.

Object Storage

Caches are used for storing relatively small amounts of data that must be accessible with submillisecond latency. Persistent storage devices can store up to 64TB on a single disk and provide up to hundreds of IOPS for read and write operations. When you need to store large volumes of data, that is, up to exabytes, and share it widely, object storage is a good option. GCP's object storage is Cloud Storage.

Features of Cloud Storage

Cloud Storage is an object storage system, which means files that are stored in the system are treated as atomic units—that is, you cannot operate on part of the file, such as reading only a section of the file. You can perform operations on an object, like creating or deleting it, but Cloud Storage does not provide functionality to manipulate subcomponents of a file. For example, there is no Cloud Storage command for overwriting a section of the file. Also, Cloud Storage does not support concurrency and locking. If multiple clients are writing to a file, then the last data written to the file is stored and persisted.

Cloud Storage is well suited for storing large volumes of data without requiring any consistent data structure. You can store different types of data in a bucket, which is the logical unit of organization in Cloud Storage. Buckets are resources within a project. It is important to remember that buckets share a global namespace, so each bucket name must be globally unique. We shouldn't be surprised if we can't name a bucket "mytestbucket" but it's not too difficult to find a unique filename, especially if you follow a bucket and object naming convention.

It is important to remember that object storage does not provide a file system. Buckets are analogous to directories in that they help organize objects into groups, but buckets are not true directories that support features such as subdirectories. Google does support an open source project called Cloud Storage Fuse, which provides a way to mount a bucket as a

file system on Linux and Mac operating systems. Using Cloud Storage Fuse, you can download and upload files to buckets using file system commands, but it does not provide full file system functionality. Cloud Storage Fuse has the same limitations as Cloud Storage. Its purpose is to make it more convenient to move data in and out of buckets when working in a Linux or Mac file system.

Cloud Storage provides four different classes of object storage: mult-regional, regional, nearline, and coldline.

Multiregional and Regional Storage

When you create a bucket, you specify a location to create the bucket. The bucket and its contents are stored in this location. You can store your data in a single region, known as a regional bucket, or multiple regions, not surprisingly known as multiregional buckets. Multiregional buckets provide more than 99.99 percent typical monthly availability with a 99.95 percent availability service level agreement (SLA). Data is replicated in multiple regions. Regional buckets have a 99.99 percent typical monthly availability and a 99.9 percent availability SLA. Regional buckets are redundant across zones.

Multiregional buckets are used when content needs to be stored in multiple regions to ensure acceptable times to access content. It also provides redundancy in case of zone-level failures. These benefits come with a higher cost, however. At the time of writing, multiregional storage in the United States costs $0.26/GB/month, while regional storage costs $0.20/GB/month. (You are not likely to be asked about specific prices on the Associate Cloud Engineer exam, but you should know the relative costs so that you can identify the lowest-cost solution that meets a set of requirements.)

Both regional and multiregional storage are used for frequently used data. If you have an application where users download and access files often, such as more than once per month, then it is most cost-effective to choose regional or multiregional. You choose between regional and multiregional based on the location of your users. If users are globally dispersed and require access to synchronized data, then multi-regional may provide better performance and availability.

What if your data is not actively used? For example, if you have files you need to store for seven years for compliance but don't expect to access, then you may want archival storage. Similarly, if you are storing files you need only for disaster recovery, then you may want a storage class designed for highly infrequent access, such as less than once per year. For these kinds of use cases, Google designed nearline and coldline storage classes.

A note on terminology: Google sometimes uses the term *georedundant*. Georedundant data is stored in at least two locations that are at least 100 miles apart. If your data is in multiregional locations, then it is georedundant.

Nearline and Coldline Storage

For infrequently accessed data, the nearline and coldline storage classes are good options. Nearline storage is designed for use cases in which you expect to access files less than once

per month. Coldline storage is designed, and priced, for files expected to be accessed once per year or less.

Nearline storage has a 99.95 percent typical monthly availability in multiregional locations and a 99.9 percent typical availability in regional locations. The SLAs for nearline are 99.9 percent in multiregional locations and 99.0 percent in regional locations. These lower SLAs come with a significantly lower cost: $0.10/GB/month. Before you start moving all your regional and multiregional data to nearline to save on costs, you should know that Google adds a data retrieval charge to nearline and coldline storage. The retrieval price for nearline storage is $0.01/GB. There is also a minimum 30-day storage duration for nearline storage.

Coldline storage has a 99.95 percent typical monthly availability in multiregional locations and a 99.9 percent typical availability in regional locations. The SLAs are 99.9 percent for multiregional locations and 99.0 percent for regional locations. Coldline also has the lowest cost per gigabyte at $0.07/GB/month. Remember, that is only the storage charge. Like nearline storage, coldline storage has access charges. Google expects data in coldline storage to be accessed once per year or less and have at least a 90-day minimum storage. The retrieval price for coldline storage is $0.05/GB.

It is more important to understand the relative cost relationships than the current prices. Prices can change, but the costs of each class relative to other classes of storage are more likely to stay the same. See Table 11.1 for a summary of features, costs and use cases for different storage types.

TABLE 11.1 Storage Services—Summary of Features

: :	**Regional**	**Multiregional**	**Nearline**	**Coldline**
Features	Object storage replicated across multiple zones	Object storage replicated across multiple regions	Object storage for access less than once per month	Object storage for access less than once per year
Storage cost	$0.20/GB/month	$0.26/GB/month	$0.10/GB/month	$0.07/GB/month
Access cost			$0.01/GB	$0.05/GB
Use case	Object storage shared across applications	Global access to shared objects	Older data in data lakes, backups	Document retention, compliance

Versioning and Object Lifecycle Management

Buckets in Cloud Storage can be configured to retain versions of objects when they are changed. When versioning is enabled on a bucket, a copy of an object is archived each time the object is overwritten or when it is deleted. The latest version of the object is known as the live version. Versioning is useful when you need to keep a history of changes to an object or want to mitigate the risk of accidentally deleting an object.

Cloud Storage also provides lifecycle management policies to automatically change an object's storage class or delete the object after a specified period. A lifecycle policy, sometimes called a configuration, is a set of rules. The rules include a condition and an action. If the condition is true, then the action is executed. Lifecycle management policies are applied to buckets and affect all objects in the bucket.

Conditions are often based on age. Once an object reaches a certain age, it can be deleted or moved to a lower-cost storage class. In addition to age, conditions can check the number of versions, whether the version is live, whether the object was created before a specific date, and whether the object is in a particular storage class.

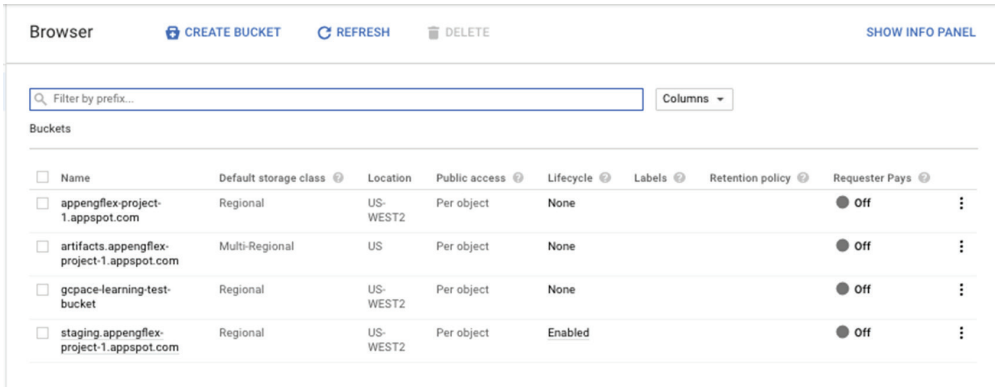
You can delete an object or change its storage class. Both unversioned and versioned objects can be deleted. If the live version of a file is deleted, then instead of actually deleting it, the object is archived. If an archived version of an object is deleted, the object is permanently deleted.

You can also change the storage class of an object using lifecycle management. There are restrictions on which classes can be assigned. Multiregional and regional storage objects can be changed to nearline or coldline. Nearline can be changed only to coldline.

Configuring Cloud Storage

You can create buckets in Cloud Storage using the console. From the main menu, navigate to Storage and select Create Bucket. This will display a form similar to Figure 11.3.

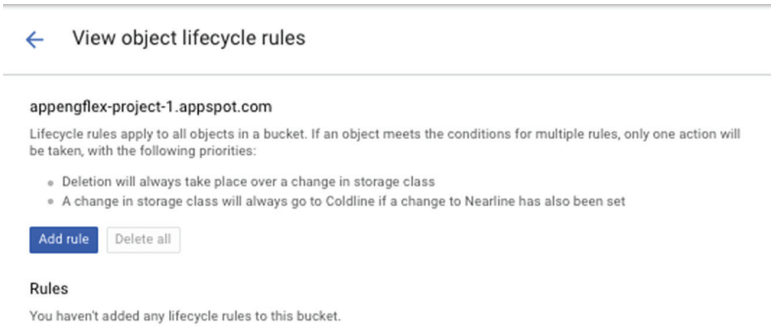
FIGURE 11.3 Form to create a storage bucket from the console. Advanced options are displayed.



When creating a bucket, you need to supply some basic information, including a bucket name and storage class. You can optionally add labels and choose either Google-managed keys or customer-managed keys for encryption. You can also set a retention policy to prevent changes to files or deleting files before the time you specify.

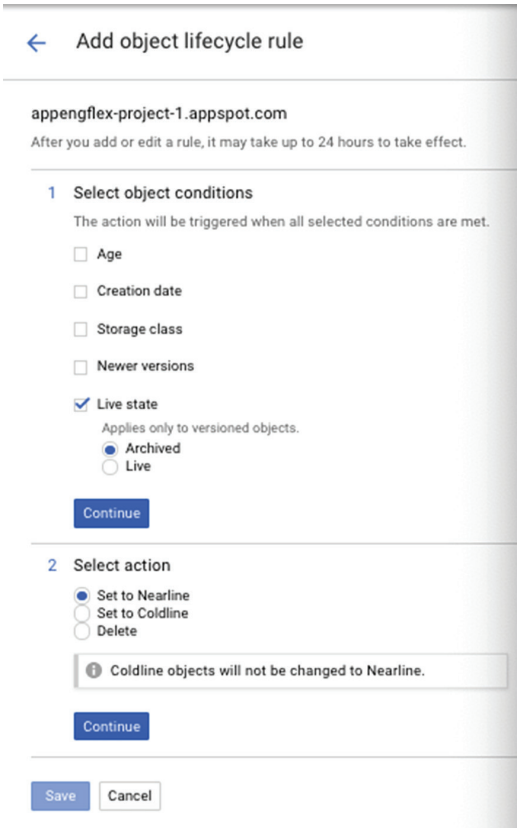
Once you have created a bucket, you define a lifecycle policy. From the Storage menu in the console, choose the Browse option, as shown in Figure 11.4.

FIGURE 11.4 The list of buckets includes a link to define or modify lifecycle policies.



Notice that the Lifecycle column indicates whether a lifecycle configuration is enabled. Choose a bucket to create or modify a lifecycle and click None or Enabled in the Lifecycle column. This will display a form such as in Figure 11.5.

FIGURE 11.5 When creating a lifecycle policy, click the Add Rule option to define a rule.



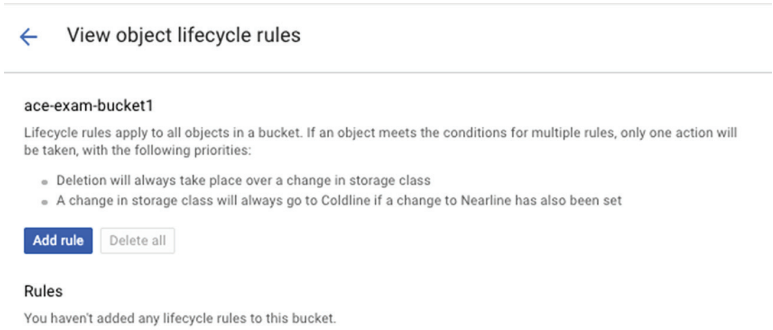
When you add a rule, you need to specify the object condition and the action. Condition options are Age, Creation Data, Storage Class, Newer Versions, and Live State. Live State applies to version objects, and you can set your condition to apply to either live or archived versions of an object. The action can be to set the storage class to either nearline or coldline.

Let's look at an example policy. From the Browser section of Cloud Storage in the console, you can see a list of buckets and their current lifecycle policies, as shown in Figure 11.6.

FIGURE 11.6 Listing of buckets in Cloud Storage Browser

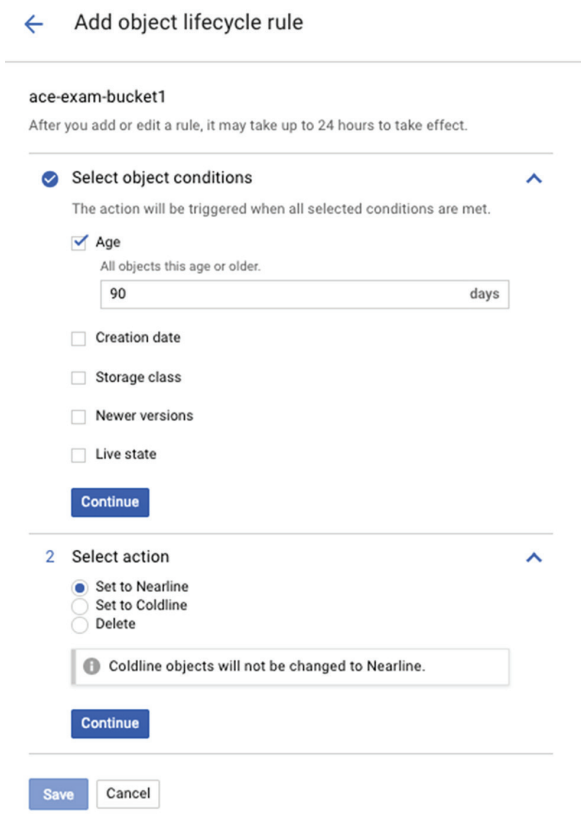
Click the policy status of a bucket to create a lifecycle rule (see Figure 11.7).

FIGURE 11.7 Form to add a lifecycle rule to a bucket



The Add Object Lifecycle Rule form appears as in Figure 11.8. In this form, you can specify the object conditions, such as Age and Storage Class, and action, such as Set To Nearline.

FIGURE 11.8 Add an object lifecycle rule to a bucket.



Storage Types When Planning a Storage Solution

When planning a storage solution, a factor to consider is the time required to access data.

Caches, like Memorystore, offer the fastest access time but are limited to the amount of memory available. Caches are volatile; when the server shuts down, the contents of the cache are lost. You should save the contents of the cache to persistent storage at regular intervals to enable recovery to the point in time when the contents of the cache were last saved.

Persistent storage is used for block storage devices, such as disks attached to VMs. GCP offers SSD and HDD drives. SSDs provide faster performance but cost more. HDDs are used when large volumes of data need to be stored in a file system but users of the data do not need the fastest access possible.

Object storage is used for storing large volumes of data for extended periods of time. Cloud Storage has both regional and multiregional storage classes and supports lifecycle management and versioning.

In addition to choosing an underlying storage system, you will also have to consider how data is stored and accessed. For this, it is important to understand the data models available and when to use them.

Storage Data Models

There are three broad categories of data models available in GCP: object, relational, and NoSQL. In addition, we will treat mobile optimized products like Cloud Firestore and Firebase as a fourth category, although these datastores use a NoSQL model. Their mobile supporting features are sufficiently important to warrant their own description.

Object: Cloud Storage

The object storage data model treats files as atomic objects. You cannot use object storage commands to read blocks of data or overwrite parts of the object. If you need to update an object, you must copy it to a server, make the change, and then copy the updated version back to the object storage system.

Object storage is used when you need to store large volumes of data and do not need fine-grained access to data within an object while it is in the object store. This data model is well suited for archived data, machine learning training data, and old Internet of Things (IoT) data that needs to be saved but is no longer actively analyzed.

Relational: Cloud SQL, Cloud Spanner, and BigQuery

Relational databases have been the primary data store for enterprises for decades. Relational databases support frequent queries and updates to data. They are used when it is important for users to have a consistent view of data. For example, if two users are reading data from a relational table at the same time, they will see the same data. This is not always the case with databases that may have inconsistencies between replicas of data, such as some NoSQL databases.

Relational databases, like Cloud SQL and Cloud Spanner, support database transactions. A transaction is a set of operations that is guaranteed to succeed or fail in its entirety—there is no chance that some operations are executed and others are not. For example, when a customer purchases a product, the count of the number of products available is decremented in the inventory table, and a record is added to a customer-purchased products table. With transactions, if the database fails after updating inventory but before

updating the customer-purchased products table, the database will roll back the partially executed transaction when the database restarts.

Cloud SQL and Cloud Spanner are used when data is structured and modeled for relational databases. Cloud SQL is a managed database service that provides MySQL and PostgreSQL databases. Cloud SQL is used for databases that do not need to scale horizontally, that is, by adding additional servers to a cluster. Cloud SQL databases scale vertically, that is, by running on servers with more memory and more CPU. Cloud Spanner is used when you have extremely large volumes of relational data or data that needs to be globally distributed while ensuring consistency and transaction integrity across all servers.

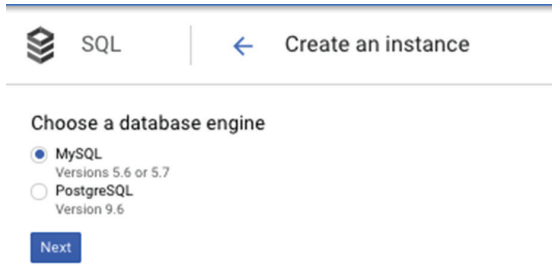
Large enterprises often use Cloud Spanner for applications like global supply chains and financial services applications, while Cloud SQL is often used for web applications, business intelligence, and ecommerce applications.

BigQuery is a service designed for a data warehouse and analytic applications. BigQuery is designed to store petabytes of data. BigQuery works with large numbers of rows and columns of data and is not suitable for transaction-oriented applications, such as ecommerce or support for interactive web applications.

Configuring Cloud SQL

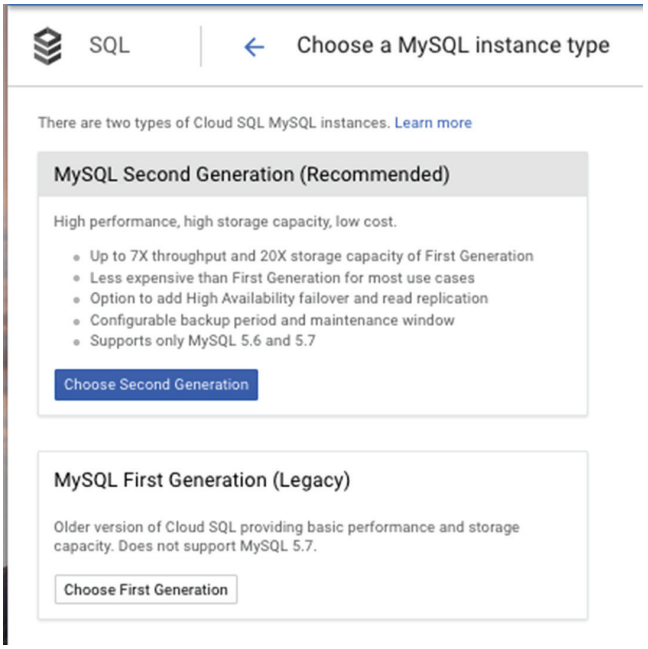
You can create a Cloud SQL instance by navigating to Cloud SQL in the main menu of the console and selecting Create Instance. You will be prompted to choose either a MySQL or PostgreSQL instance, as shown in Figure 11.9.

FIGURE 11.9 Cloud SQL provides both MySQL and PostgreSQL instances.



If you choose PostgreSQL, you are taken to the Configuration form. If you choose MySQL, you are prompted to choose either a First Generation or Second Generation MySQL instance (see Figure 11.10). Unless you need to use an older version of MySQL, a Second Generation instance is recommended. MySQL 2nd generation will provide greater capacity, optional high availability configurations, support for MySQL 5.7, and, in many cases, lower cost.

FIGURE 11.10 MySQL instances are available in First and Second Generation instances.



To configure a MySQL instance, you will need to specify a name, root password, region, and zone. The configuration options include the following:

- MySQL version.
- Connectivity, where you can specify whether to use a public or private IP address.
- Machine type. The default is a db-n1-standard-1 with 1 vCPU and 3.75GB of memory.
- Automatic backups.
- Failover replicas.
- Database flags. These are specific to MySQL and include the ability to set a database read-only flag and set the query cache size.
- Setting a maintenance time window.
- Labels.

Figure 11.11 shows the configuration form for MySQL second-generation, and Figure 11.12 shows the PostgreSQL configuration form.

FIGURE 11.11 Configuration form for a MySQL Second Generation instance

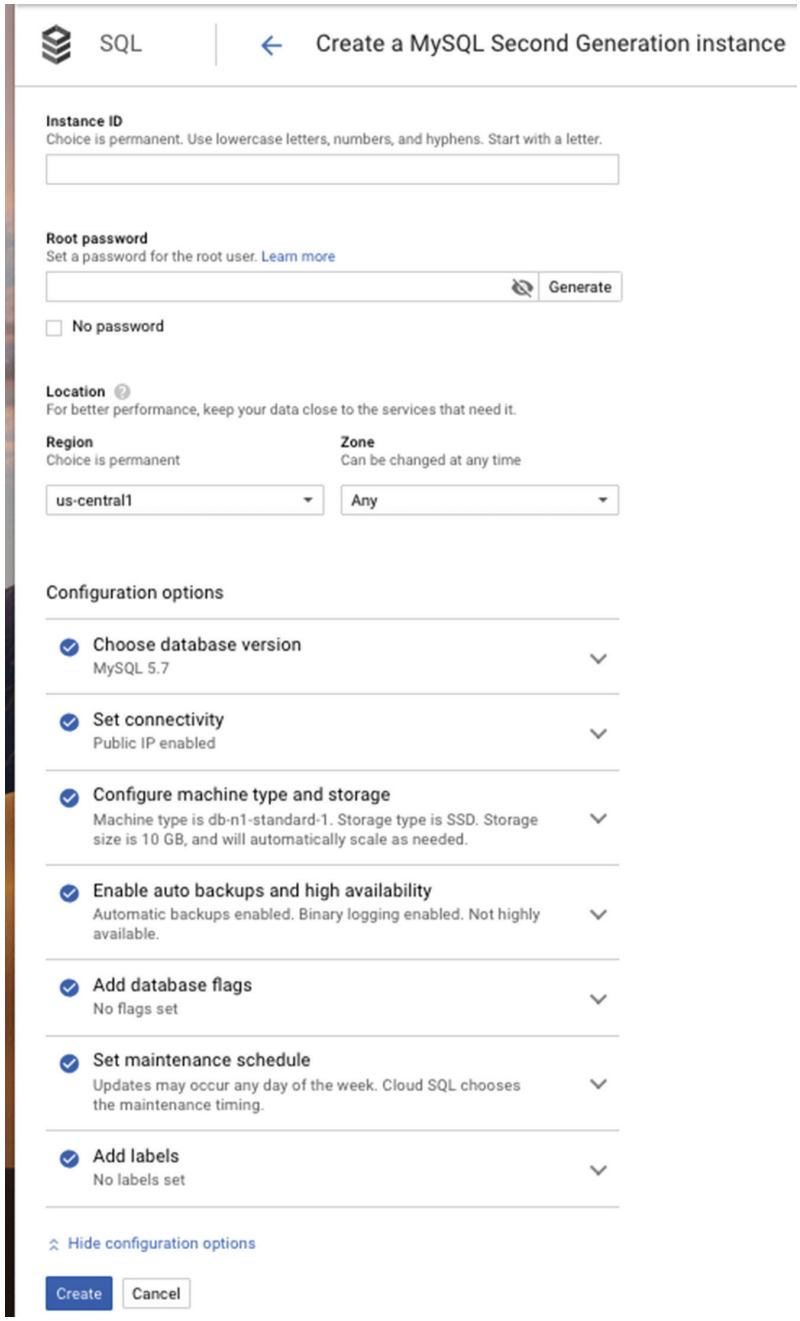
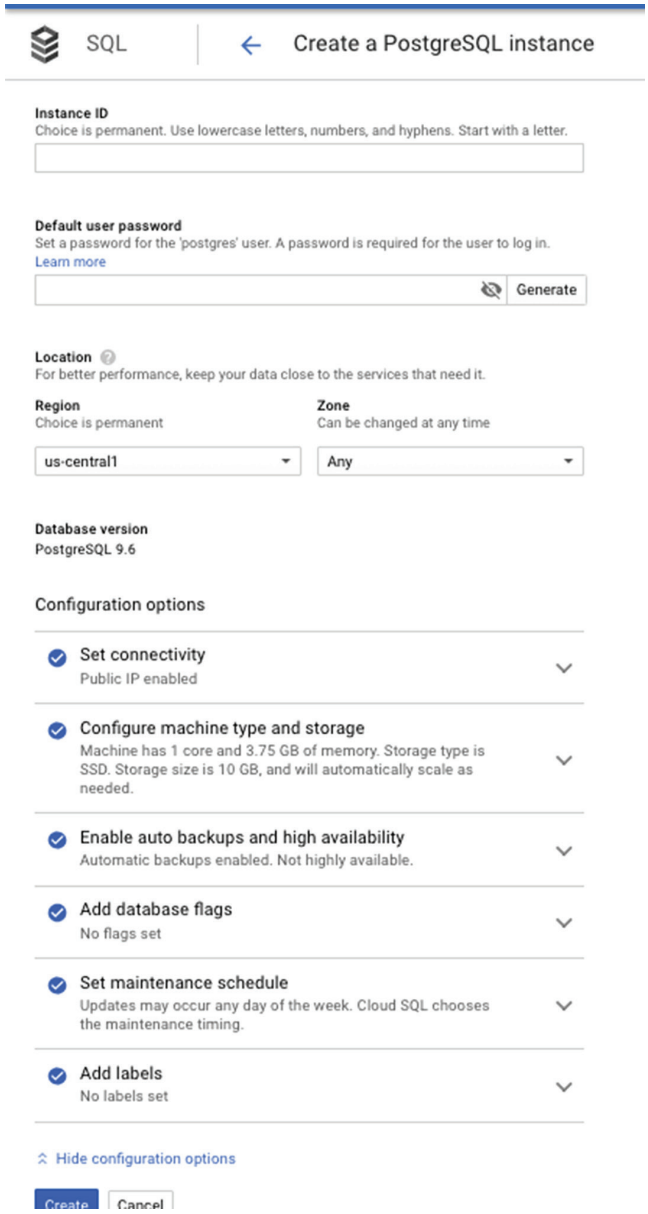


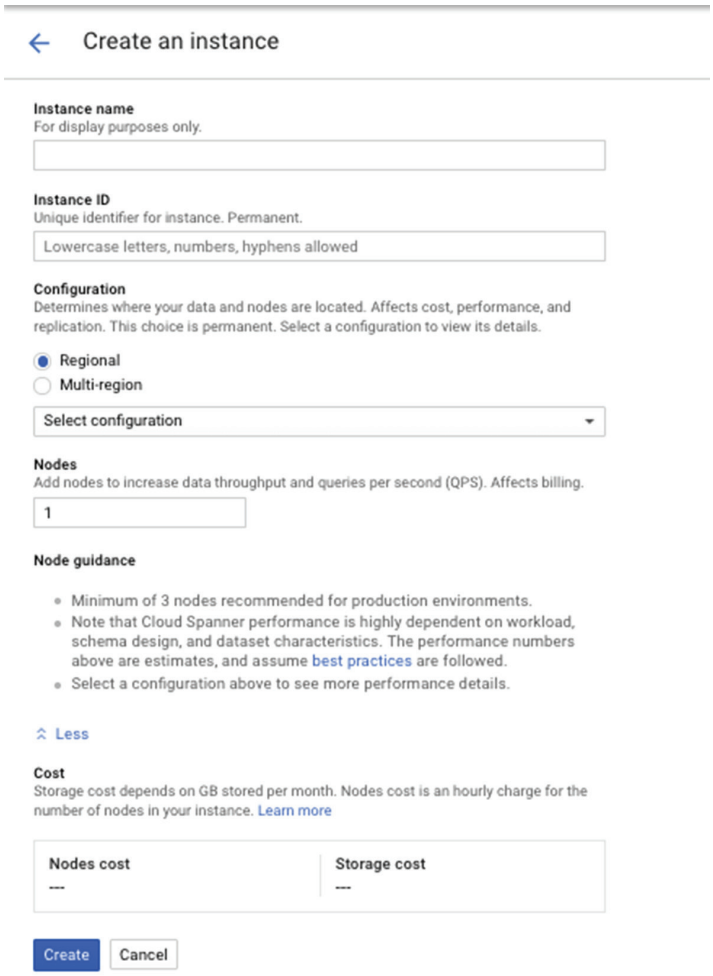
FIGURE 11.12 Configuration form for a PostgreSQL instance



Configuring Cloud Spanner

If you need to create a global, consistent database with support for transactions, then you should consider Cloud Spanner. Given the advanced nature of Spanner, its configuration is surprisingly simple. In the console, navigate to Cloud Spanner and select Create Instance to display a form like Figure 11.13.

FIGURE 11.13 The Cloud Spanner configuration form in Cloud Console



You need to provide an instance name, instance ID, and number of nodes.

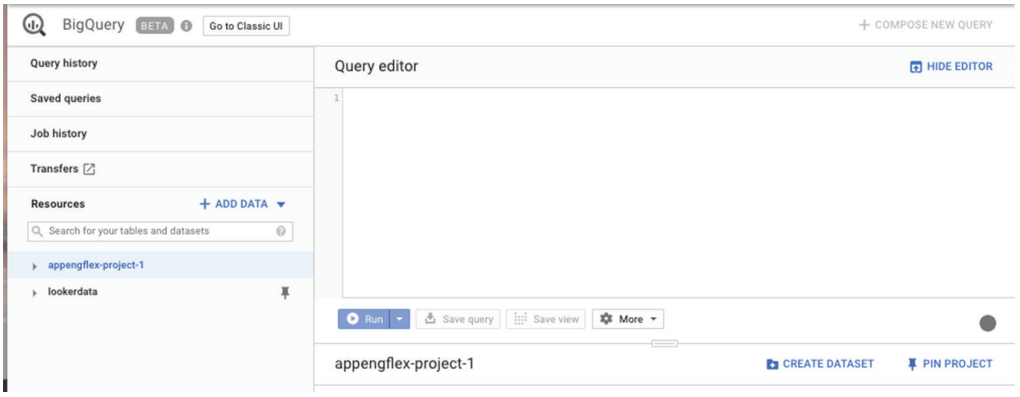
You will also have to choose either a regional or multiregional configuration to determine where nodes and data are located. This will determine cost and replication storage location. If you select regional, you will choose from the list of available regions, such as us-west1, asia-east1, and europe-north1.

It should be noted that Cloud Spanner is significantly more expensive than Cloud SQL or other database options. A single regional node located in us-central1 costs $0.90 per hour, while a single multiregional node in nam3 costs $3 per hour. A single multiregional node in nam-eur-asia1 costs $9 per hour.

Configuring BigQuery

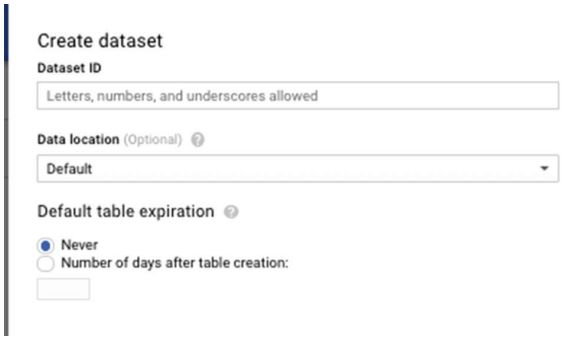
BigQuery is a managed analytics service, which provides storage plus query, statistical, and machine learning analysis tools. BigQuery does not require you to configure instances. Instead, when you first navigate to BigQuery from the console menu, you will see a form such as in Figure 11.14.

FIGURE 11.14 BigQuery user interface for creating and querying data



The first task for using BigQuery is to create a data set to hold data. You do this by clicking Create Dataset to display the form shown in Figure 11.15.

FIGURE 11.15 Form to create a dataset in BigQuery



When creating a data set, you will have to specify a name and select a region in which to store it. Not all regions support BigQuery. Currently you have a choice of nine locations across the United States, Europe, and Asia.

In Chapter 12, we will discuss how to load and query data in BigQuery and other GCP databases.

NoSQL: Datastore, Cloud Firestore, and Bigtable

NoSQL databases do not use the relational model and do not require a fixed structure or schema. Database schemas define what kinds of attributes can be stored. When no fixed schema is required, developers have the option to store different attributes in different records. GCP has three NoSQL options:

- Cloud Datastore
- Cloud Firestore
- Cloud Bigtable

Datastore Features

Datastore is a document database. That does not mean it is used to store documents like spreadsheets or text files, but the data in the database is organized into a structure called a document. Documents are made up of sets of key-value pairs. A simple example is as follows:

```
{
book : "ACE Exam Guide",
        chapter: 11,
    length: 20,
    topic: "storage"
  }
```

This example describes the characteristics of a chapter in a book. There are four keys or properties in this example: book, chapter, length, and storage. This set of key-value pairs is called an entity in Datastore terminology. Entities often have properties in common, but since Datastore is a schemaless database, there is no requirement that all entities have the same set of properties. Here's an example:

```
{
book : "ACE Exam Guide",
        Chapter: 11,
    topic: "computing",
    number_of_figures: 8
  }
```

Datastore is a managed database, so users of the service do not need to manage servers or install database software. Datastore automatically partitions data and scales up or down as demand warrants.

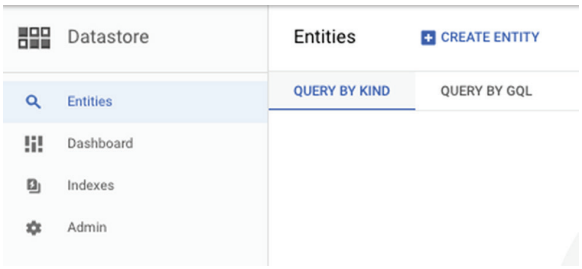
Datastore is used for nonanalytic, nonrelational storage needs. It is a good choice for product catalogs, which have many types of products with varying characteristics or properties. It is also a good choice for storing user profiles associated with an application.

Datastore has some features in common with relational databases, such as support for transactions and indexes to improve query performance. The main difference is that Datastore does not require a fixed schema or structure and does not support relational operations, such as joining tables, or computing aggregates, such as sums and counts.

Configuring Datastore

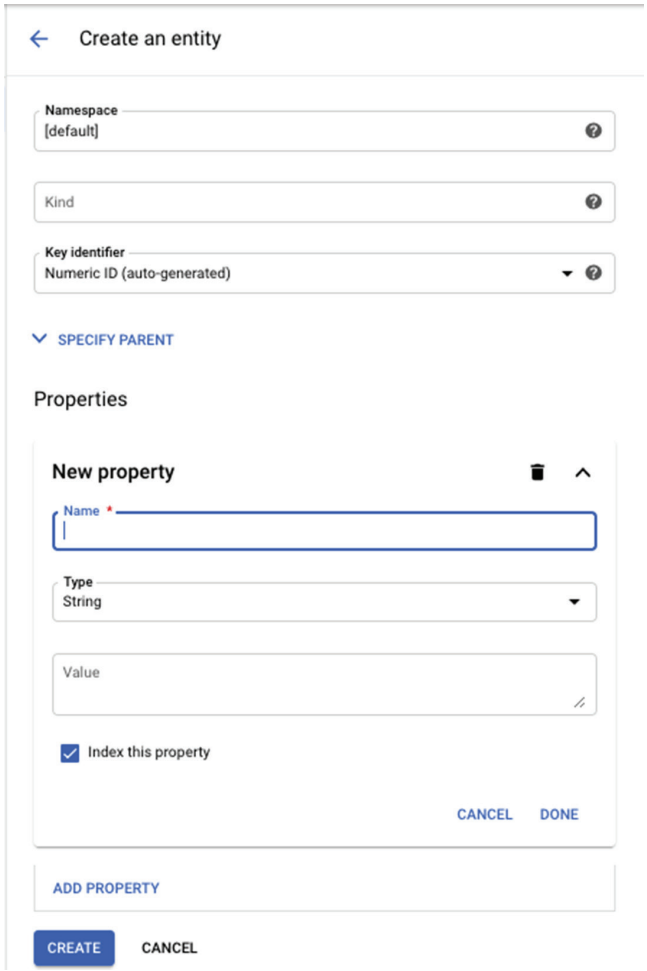
Datastore, like BigQuery, is a managed database service that does not require you to specify node configurations. Instead, you can work from the console to add entities to the database. Figure 11.16 shows the initial form that appears when you first navigate to Datastore in Cloud Console.

FIGURE 11.16 The Datastore user interface allows you to create and query data.



Select Create Entity to display a form to add data in a document data structure, as shown in Figure 11.17.

FIGURE 11.17 Adding entities to Datastore



When creating an entity, you specify a namespace, which is a way to group entities much like schemas group tables in a relational database. You will need to specify a kind, which is analogous to a table in a relational database. Each entity requires a key, which can be an autogenerated numeric key or a custom-defined key.

Next, you will add one or more properties that have a names, types, and values. Types include string, date and time, Boolean, and other structured types like arrays.

Additional details on loading and querying data in Datastore are in Chapter 12.

Cloud Firestore Features

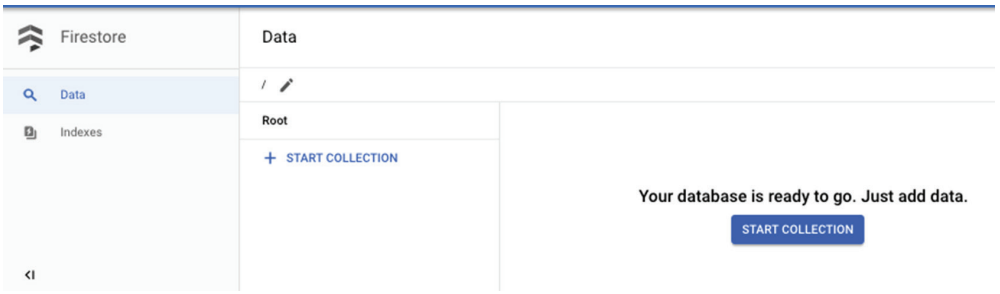
Cloud Firestore is a managed NoSQL database that uses the document data model. This is similar to Datastore, and in fact, Datastore databases can use the newer Cloud Firestore storage system. One advantage of Cloud Firestore is that it is designed for storing, synchronizing, and querying data across distributed applications, like mobile apps. Apps can be automatically updated in close to real time when data is changed on the backend.

Cloud Firestore supports transactions and provides multiregional replication.

Configuring Cloud Firestore

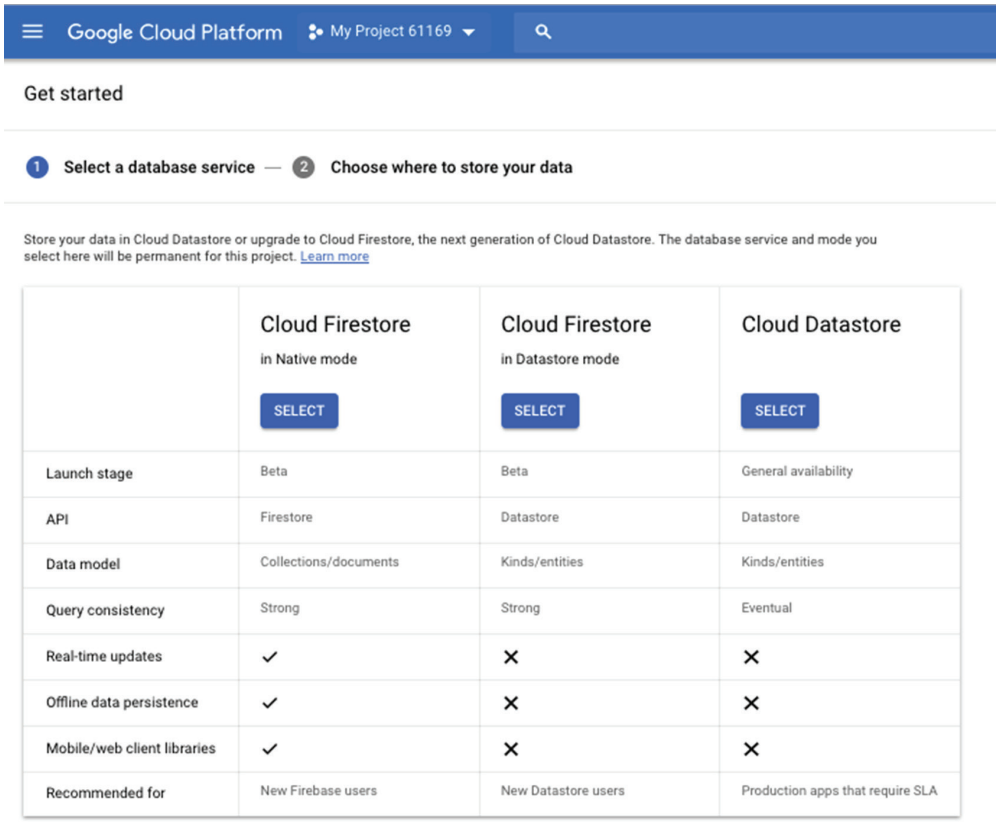
Cloud Firestore is a managed database service that does not require you to configure instances. You do, however, have to choose a data storage system. The options include using Datastore, using Firestore in Datastore mode (which uses the Datastore storage system,) or using Firestore in native mode. New Firestore users should use Firestore in native mode (see Figure 11.18).

FIGURE 11.18 Firestore can be configured to use Datastore's backend storage system or its newer native storage system.



After selecting the storage system, you will be prompted to select a location for the database, as shown in Figure 11.19.

FIGURE 11.19 Selecting a location for a Firebase database



Firestore will create a database, which may take a few minutes. When the database is ready, you will see a display such as in Figure 11.20.

In Chapter 12, we will look into loading and querying data from Firestore.

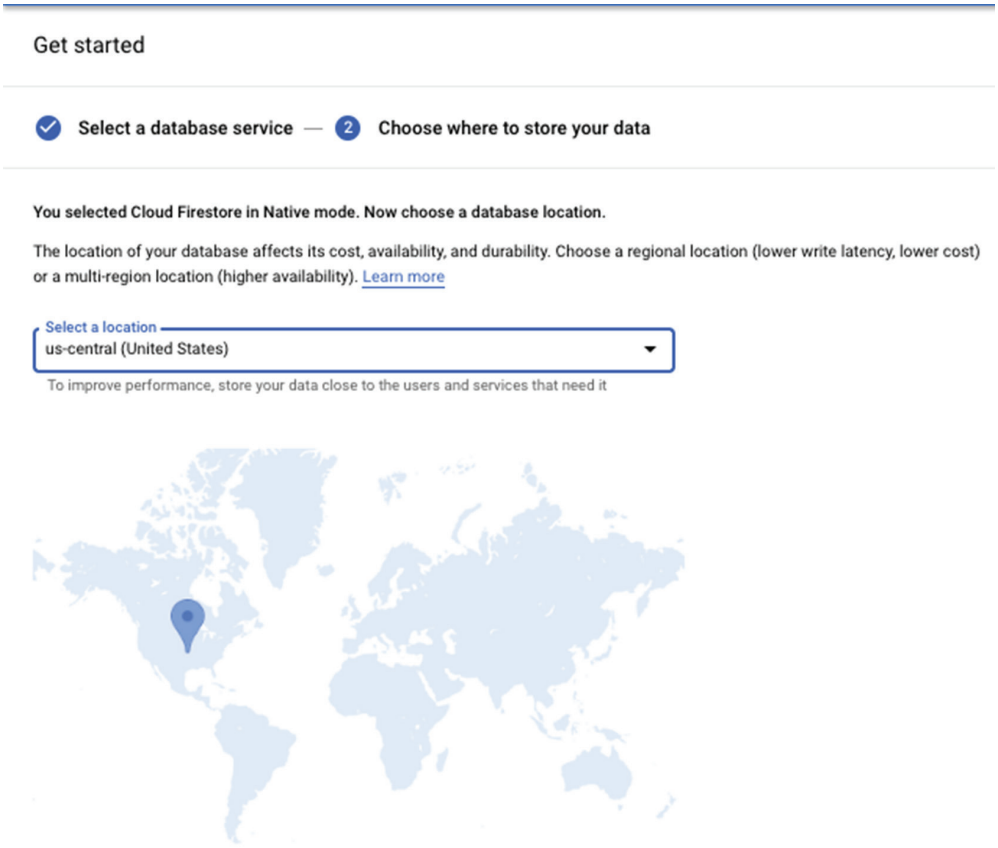
Bigtable Features

Bigtable is another NoSQL database, but unlike Datastore, it is a wide-column database, not a document database. Wide-column databases, as the name implies, store tables that can have a large number of columns. Not all rows need to use all columns, so in that way it is like Datastore—neither require a fixed schema to structure the data.

Bigtable is designed for petabyte-scale databases. Both operational databases, like storing IoT data, and analytic processing, like data science applications, can effectively use Bigtable. This database is designed to provide consistent, low-millisecond latency. Bigtable runs in clusters and scales horizontally.

Bigtable is designed for applications with high data volumes and a high-velocity ingest of data. Time series, IoT, and financial applications all fall into this category.

FIGURE 11.20 Firestore database ready for use



Configuring Bigtable

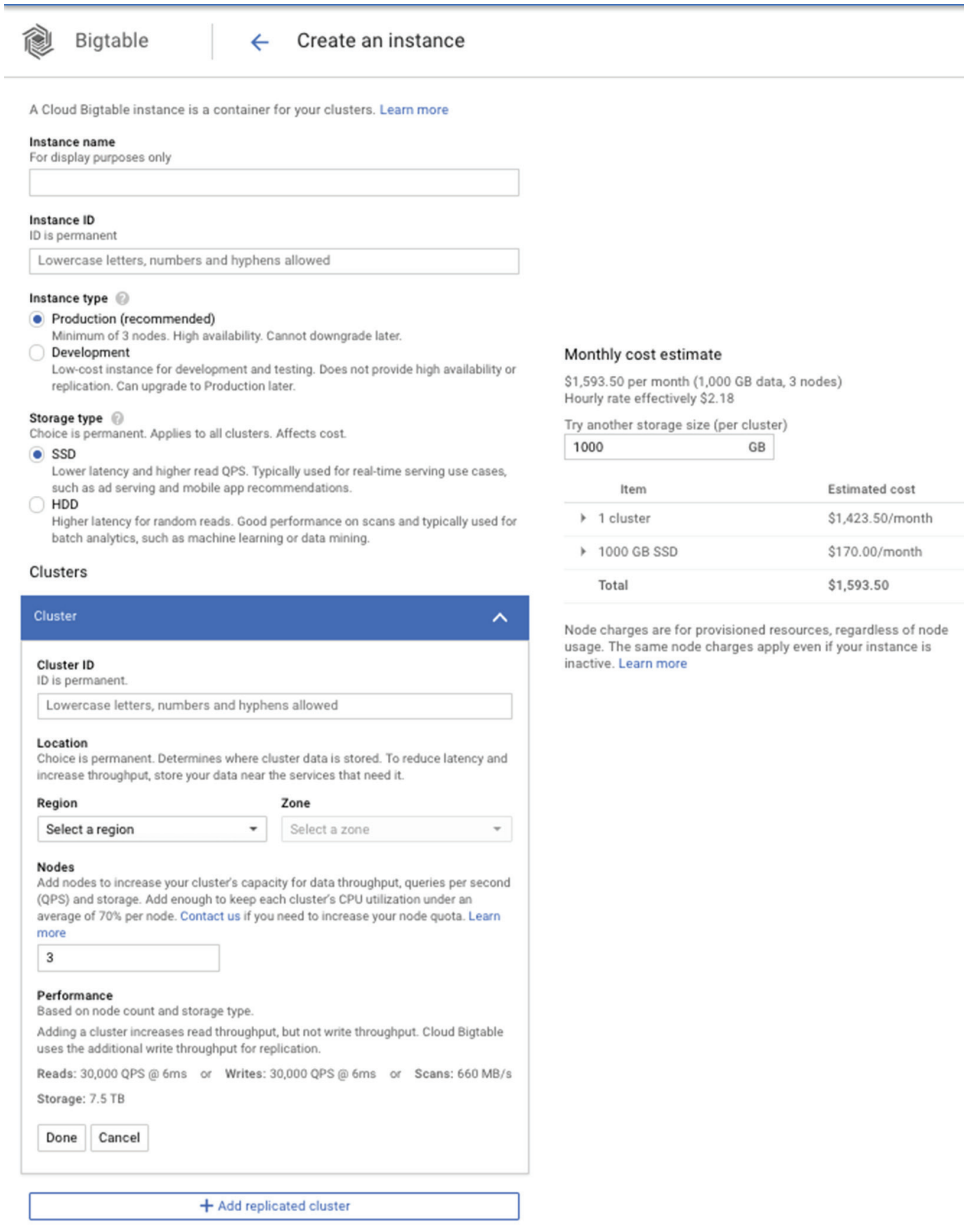
From Cloud Console, navigate to Bigtable and click Create Instance. This will display a form such as shown in Figure 11.21.

In this form, you will need to provide an instance name and an instance ID. Next, choose between production or development mode. Production clusters have a minimum of three nodes and provide for high availability. Development mode uses low-cost instances without replication or high availability. You will also need to choose either SSD or HDD for persistent disks used by the database.

Bigtable can support multiple clusters. For each cluster you will need to specify a cluster ID, a region and zone location, and the number of nodes in the cluster. The cluster can be replicated to improve availability.

In Chapter 12, we will describe how to load and query data in Bigtable.

FIGURE 11.21 Configuration form for Bigtable



Real World Scenario

The Need for Multiple Databases

Healthcare organizations and medical facilities store and manage a wide range of data about patients, their treatments, and the outcomes. A patient's medical records include demographic information, such as name, address, age, and so on. Medical records also store detailed information about medical conditions and diagnoses as well as treatment, such as drugs prescribed and procedures performed. This kind of data is highly structured. Transaction support and strong consistency are required. Relational databases, like Cloud SQL, are a good solution for this kind of application.

The medical data stored in transactional, relational databases is valuable for analyzing patterns in treatments and recovery. For example, data scientists could use medical records to identify patterns associated with re-admission to the hospital. However, transactional relational databases are not suited for analytics. A better option is to use BigQuery and build a data warehouse with data structured in ways that make it easier to analyze data. Data from the transactional system is extracted, transformed, and loaded into a Bigtable data set.

Choosing a Storage Solution: Guidelines to Consider

GCP offers multiple storage solutions. As a cloud engineer, you may have to help plan and implement storage solutions for a wide range of applications. The different storage solutions lend themselves to different use cases, and in many enterprise applications, you will find that you need two or more different storage products to support the full range of application requirements. Here are several factors to keep in mind when choosing storage solutions:

Read and Write Patterns Some applications, such as accounting and retail sales applications, read and write data frequently. There are also frequent updates in these applications. They are best served by a storage solution such as Cloud SQL if the data is structured; however, if you need a global database that supports relational read/write operations, then Cloud Spanner is a better choice. If you are writing data at consistently high rates and in large volumes, consider Bigtable. If you are writing files and then downloading them in their entirety, Cloud Storage is a good option.

Consistency Consistency ensures that a user reading data from the database will get the same data no matter which server in a cluster responds to the request. If you need strong consistency, which is always reading the latest data, then Cloud SQL and Cloud Spanner

are good options. Datastore can be configured for strong consistency, but IO operations will take longer than if a less strict consistency configuration is used. Datastore is a good option if your data is unstructured; otherwise, consider one of the relational databases. NoSQL databases offer at least eventual consistency, which means some replicas may not be in sync for a short period of time. During those periods it is possible to read stale data. If your application can tolerate that, then you may find that less strict consistency requirements can lead to faster read and write operations.

Transaction Support If you need to perform atomic transactions in your application, use a database that supports them. You may be able to implement transaction support in your application, but that code can be difficult to develop and maintain. The relational databases, Cloud SQL and Spanner, and Datastore provide transaction support.

Cost The cost of using a particular storage system will depend on the amount of data stored, the amount of data retrieved or scanned, and per-unit charges of the storage system. If you are using a storage service in which you provision VMs, you will have to account for that cost as well.

Latency Latency is the time between the start of an operation, like a request to read a row of data from a database, to the time it completes. Bigtable provides consistently low-millisecond operations. Spanner can have longer latencies, but with those longer latencies you get a globally consistent, scalable database.

In general, choosing a data store is about making tradeoffs. In an ideal world, we could have a low-cost, globally scalable, low-latency, strongly consistent database. We don't live in an ideal world. We have to give up one or more of those characteristics.

In the next chapter, you will learn how to use each of the storage solutions described here, with an emphasis on loading and querying data.

Summary

When planning cloud storage, consider the types of storage systems and types of data models. The storage systems provide the hardware and basic organizational structure used for storing data. The data models organize data into logical structures that determine how data is stored and queried within a database.

The main storage systems available in GCP are Memorystore, a managed cache service, and persistent disks, which are network-accessible disks for VMs in Compute Engine and Kubernetes Engine. Cloud Storage is GCP's object storage system.

The primary data models are object, relational, and NoSQL. NoSQL databases in GCP are further subdivided into document and wide-column databases. Cloud Storage uses an object data model. Cloud SQL and Cloud Spanner use relational databases for transaction processing applications. BigQuery uses a relational model for data warehouse and analytic applications. Datastore and Firebase are document databases. Bigtable is a wide-column table.

When choosing data storage systems, consider read and write patterns, consistency requirements, transaction support, cost, and latency.

Exam Essentials

Know the major storage system types, including caches, persistent disks, and object storage. Caches are used to improve application performance by reducing the need to read from databases on disk. Caches are limited by the amount of available memory. Persistent disks are network devices that are attached to VMs. Persistent disks may be attached to multiple VMs in read-only mode. Object storage is used for storing files for shared access and long-term storage.

Know the major kinds of data models. Relational databases are used for transaction processing systems that require transaction support and strong consistency. Cloud SQL and Cloud Spanner are relational databases used for transaction processing applications. BigQuery uses a relational model but is designed for data warehouses and analytics. The object model is an alternative to a file system model. Objects, stored as files, are treated as atomic units. NoSQL data models include document data models and wide-column models. Datastore and Firebase are document model databases. Bigtable is a wide-column database.

Know the four storage classes in Cloud Storage. Regional, multiregional, nearline, and coldline are the four storage classes. Multiregional class replicates data across regions. Regional storage replicates data across zones. Nearline is designed for infrequent access, less than once per month. Coldline storage is designed for archival storage, with files being accessed less than once per year. Both nearline and coldline storage incur retrieval charges in addition to charges based on the size of data.

Know that cloud applications may require more than one kind of data store. For example, an application may need a cache to reduce latency when querying data in Cloud SQL, object storage for the long-term storage of data files, and BigQuery for data warehousing reporting and analysis.

Know that you can apply lifecycle configurations on Cloud Storage buckets. Lifecycles are used to delete files and change storage class. Know that regional and multiregional class can be changed to nearline or coldline. Nearline storage can change to coldline. Regional class storage cannot be changed to multiregional, and multiregional cannot be changed to regional.

Know the characteristics of different data stores that help you determine which is the best option for your requirements. Read and write patterns, consistency requirements, transaction support, cost, and latency are often factors.

Review Questions

You can find the answers in the Appendix.

1. You are tasked with defining lifecycle configurations on buckets in Cloud Storage. You need to consider all possible options for transitioning from one storage class to another. All of the following transitions are allowed except for one. Which one is that?
   - **A.** Nearline to coldline
   - **B.** Regional to nearline
   - **C.** Multiregional to coldline
   - **D.** Regional to multiregional

2. Your manager has asked for your help in reducing Cloud Storage charges. You know that some of the files stored in Cloud Storage are rarely accessed. What kind of storage would you recommend for those files?
   - **A.** Nearline
   - **B.** Regional
   - **C.** Coldline
   - **D.** Multiregional

3. You are working with a startup developing analytics software for IoT data. You have to be able to ingest large volumes of data consistently and store it for several months. The startup has several applications that will need to query this data. Volumes are expected to grow to petabyte volumes. Which database should you use?
   - **A.** Cloud Spanner
   - **B.** Bigtable
   - **C.** BigQuery
   - **D.** Datastore

4. A software developer on your team is asking for your help improving the query performance of a database application. The developer is using a Cloud SQL MySQL, Second Generation instance. Which options would you recommend?
   - **A.** Memorystore and SSD persistent disks
   - **B.** Memorystore and HDD persistent disks
   - **C.** Datastore and SSD persistent disks
   - **D.** Datastore and HDD persistent disks

5. You are creating a set of persistent disks to store data for exploratory data analysis. The disks will be mounted on a virtual machine in the us-west2-a zone. The data is historical data retrieved from Cloud Storage. The data analysts do not need peak performance and are more concerned about cost than performance. The data will be stored in a local relational database. Which type of storage would you recommend?
   A. SSDs
   B. HDDs
   C. Datastore
   D. Bigtable

6. Which of the following statements about Cloud Storage is not true?
   A. Cloud Storage buckets can have retention periods.
   B. Lifecycle configurations can be used to change storage class from regional to multiregional.
   C. Cloud Storage does not provide block-level access to data within files stored in buckets.
   D. Cloud Storage is designed for high durability.

7. When using versioning on a bucket, what is the latest version of the object called?
   A. Live version
   B. Top version
   C. Active version
   D. Safe version

8. A product manager has asked for your advice on which database services might be options for a new application. Transactions and support for tabular data are important. Ideally, the database would support common query tools. What databases would you recommend the product manager consider?
   A. BigQuery and Spanner
   B. Cloud SQL and Spanner
   C. Cloud SQL and Bigtable
   D. Bigtable and Spanner

9. The Cloud SQL service provides fully managed relational databases. What two types of databases are available in Cloud SQL?
   A. SQL Server and MySQL
   B. SQL Server and PostgreSQL
   C. PostgreSQL and MySQL
   D. MySQL and Oracle

10. Which of the following Cloud Spanner configurations would have the highest hourly cost?
   A. Located in us-central1
   B. Located in nam3

C. Located in us-west1-a

D. Located in nam-eur-asia1

11. Which of the following are database services that do not require you to specify configuration information for VMs?

A. BigQuery only

B. Datastore only

C. Firebase and Datastore

D. BigQuery, Datastore, and Firebase

12. What kind of data model is used by Datastore?

A. Relational

B. Document

C. Wide-column

D. Graph

13. You have been tasked with creating a data warehouse for your company. It must support tens of petabytes of data and use SQL for a query language. Which managed database service would you choose?

A. BigQuery

B. Bigtable

C. Cloud SQL

D. SQL Server

14. A team of mobile developers is developing a new application. It will require synchronizing data between mobile devices and a backend database. Which database service would you recommend?

A. BigQuery

B. Firestore

C. Spanner

D. Bigtable

15. A product manager is considering a new set of features for an application that will require additional storage. What features of storage would you suggest the product manager consider?

A. Read and write patterns only

B. Cost only

C. Consistency and cost only

D. None, they are all relevant considerations.

16. What is the maximum size of a Memorystore cache?
    - **A.** 100GB
    - **B.** 300GB
    - **C.** 400GB
    - **D.** 50GB

17. Once a bucket has its storage class set to coldline, what are other storage classes it can transition to?
    - **A.** Regional
    - **B.** Nearline
    - **C.** Multi-regional
    - **D.** None of the above

18. Before you can start storing data in BigQuery, what must you create?
    - **A.** A data set
    - **B.** A bucket
    - **C.** A persistent disk
    - **D.** An entity

19. What features can you configure when running a Second Generation MySQL database in Cloud SQL?
    - **A.** Machine type
    - **B.** Maintenance windows
    - **C.** Failover replicas
    - **D.** All of the above

20. A colleague is wondering why some storage charges are so high. They explain that they have moved all their storage to nearline and coldline storage. They routinely access most of the object on any given day. What is one possible reason the storage costs are higher than expected?
    - **A.** Nearline and coldline incur access charges.
    - **B.** Transfer charges.
    - **C.** Multiregional coldline is more expensive.
    - **D.** Regional coldline is more expensive.

Chapter 12

Deploying Storage in Google Cloud Platform

THIS CHAPTER COVERS THE FOLLOWING OBJECTIVES OF THE GOOGLE ASSOCIATE CLOUD ENGINEER CERTIFICATION EXAM:

- ✓ 3.4 Deploying and implementing data solutions
- ✓ 4.4 Managing data solutions

Official Google Cloud Certified: Associate Cloud Engineer: Study Guide
By Dan Sullivan

In this chapter, we will discuss how to create data storage systems in several Google Cloud Platform (GCP) products, including Cloud SQL, Cloud Datastore, BigQuery, Bigtable, Cloud Spanner, Cloud Pub/Sub, Cloud Dataproc, and Cloud Storage. You will learn how to create databases, buckets, and other basic data structures as well as how to perform key management tasks, such as backing up data and checking the status of jobs.

Deploying and Managing Cloud SQL

Cloud SQL is a managed relational database service. In this section, you will learn how to do the following:

- Create a database instance
- Connect to the instance
- Create a database
- Load data into the database
- Query the database
- Back up the database

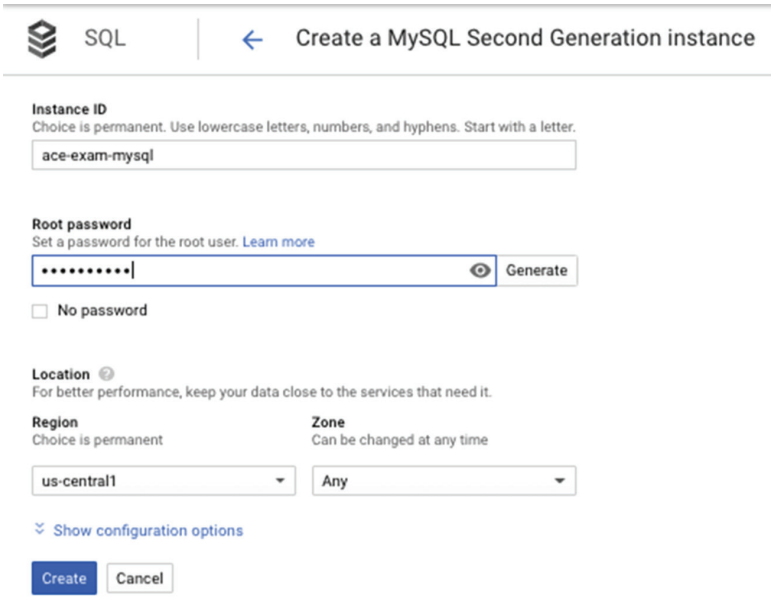
We will use a MySQL instance in this section, but the following procedures are similar for PostgreSQL.

Creating and Connecting to a MySQL Instance

We described how to create and configure a MySQL instance in Chapter 11, but will review the steps here.

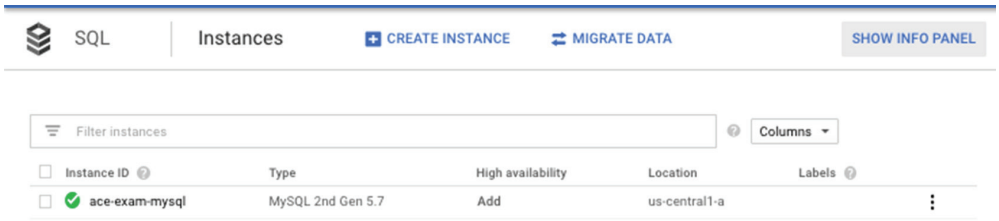
From the console, navigate to SQL and click Create Instance. Choose MySQL and then select Second Generation Instance type. This will lead to a form such as in Figure 12.1.

FIGURE 12.1 Creating a MySQL instance



After a few minutes, the instance is created; the MySQL list of instances will look similar to Figure 12.2.

FIGURE 12.2 A listing of MySQL instances



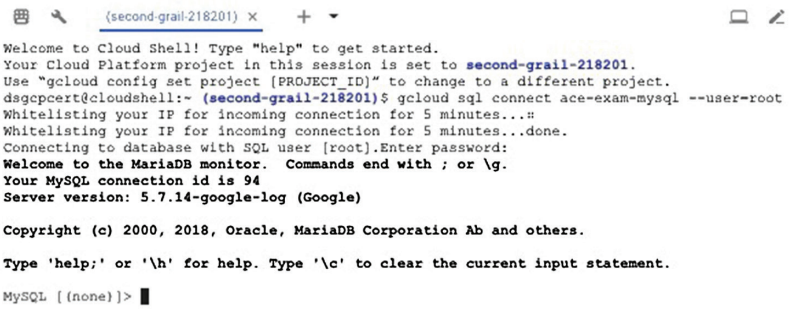
After the database is created, you can connect by starting Cloud Shell and using the `gcloud sql connect` command. This command takes the name of the instance to connect to and optionally a username and password. It is a good practice to not specify a password in the command line. Instead, you will be prompted for it, and it will not be displayed as you type. You may see a message about whitelisting your IP address; this is a security measure and will allow you to connect to the instance from Cloud Shell.

To connect to the instance called `ace-exam-mysql`, use the following command:

```
gcloud sql connect ace-exam-mysql -user=root
```

This opens a command-line prompt to the MySQL instance, as shown in Figure 12.3.

FIGURE 12.3 Command-line prompt to work with MySQL after connecting using `gcloud sql connect`



Creating a Database, Loading Data, and Querying Data

In the MySQL command-line environment, you use MySQL commands, not `gcloud` commands. MySQL uses standard SQL, so the command to create a database is `CREATE DATABASE`. You indicate the database to work with (there may be many in a single instance) by using the `USE` command. For example, to create a database and set it as the default database to work with, use this:

```
CREATE DATABASE ace_exam_book;
USE ace_exam_book
```

You can then create a table using `CREATE TABLE`. Data is inserted using the `INSERT` command. For example, the following commands create a table called `books` and inserts two rows:

```
CREATE TABLE books (title VARCHAR(255), num_chapters INT, entity_id INT NOT NULL
\AUTO_INCREMENT, PRIMARY KEY (entity_id));INSERT INTO books (title,num_chapters)
VALUES ('ACE Exam Study Guide', 18);
INSERT INTO books (title,num_chapters) VALUES ('Architecture Exam Study Guide',
18);
```

To query the table, you use the `SELECT` command. Here's an example:

```
SELECT * from books;
```

This will list all the rows in the table, as shown in Figure 12.4.

FIGURE 12.4 Listing the contents of a table in MySQL

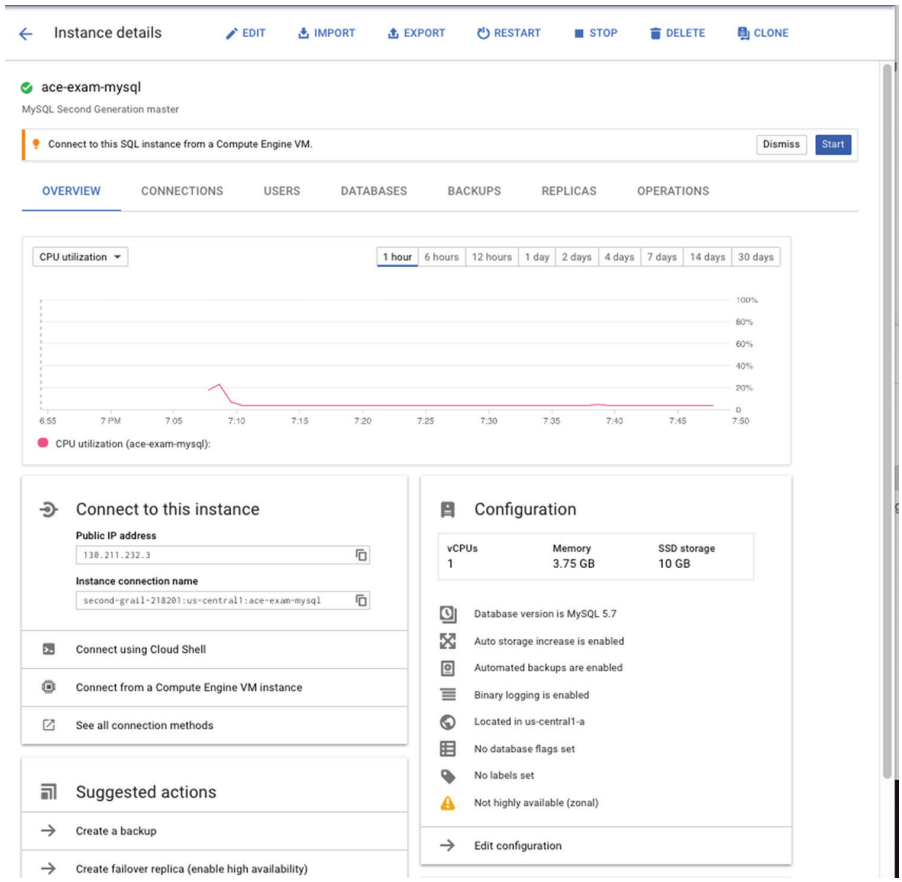
```
MySQL [ace_exam_book]> SELECT * FROM books;
+-------------------------------+--------------+-----------+
| title                         | num_chapters | entity_id |
+-------------------------------+--------------+-----------+
| ACE Exam Study Guide          |           18 |         1 |
| Architecture Exam Study Guide |           14 |         2 |
+-------------------------------+--------------+-----------+
2 rows in set (0.04 sec)

MySQL [ace_exam_book]>
```

Backing Up MySQL in Cloud SQL

Cloud SQL enables both on-demand and automatic backups.

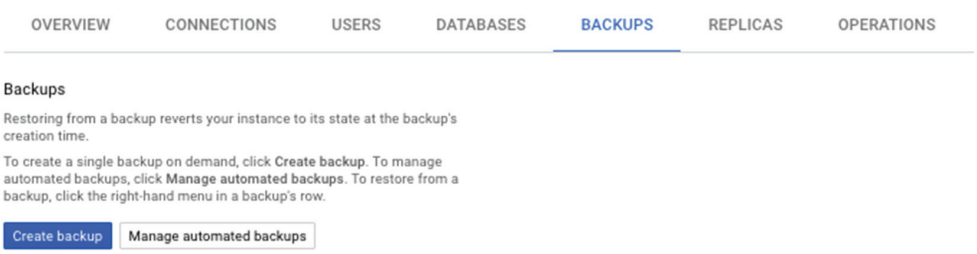
To create an on-demand backup, click the name of the instance on the Instances page on the console. This will display the Instance Details page (see Figure 12.5).

FIGURE 12.5 A MySQL Instance Details page



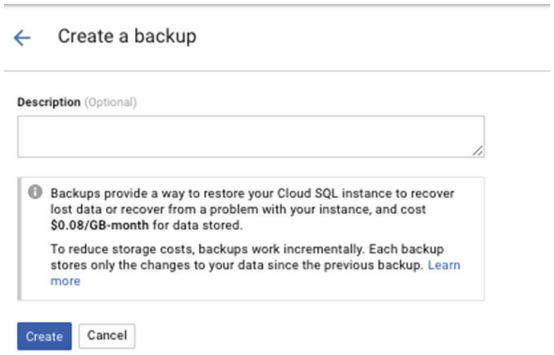
Click the Backups tab to display the Create Backup option (see Figure 12.6).

FIGURE 12.6 Form used to click Create Backup



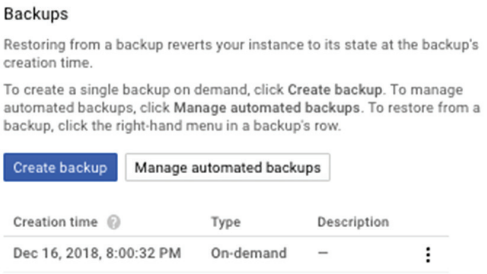
Clicking Create Backup opens a form like that shown in Figure 12.7.

FIGURE 12.7 Assign a description to a backup and create it.



Fill in the optional description and click Create. When the backup is complete, it will appear in the list of backups, as shown in Figure 12.8.

FIGURE 12.8 Listing of backups available for this instance



You can also create a backup using the `gcloud sql backups` command, which has this form:

```
gcloud sql backups create --async --instance [INSTANCE_NAME]
```

Here, *`[INSTANCE_NAME]`* is the name, such as `ace-exam-mysql` and the `--async` parameter is optional.

To create an on-demand backup for the `ace-exam-mysql` instance, use the following command:

```
gcloud sql backups create --async --instance ace-exam-mysql
```

You can also have Cloud SQL automatically create backups.

From the console, navigate to the Cloud SQL Instance page, click the name of the instance, and then click Edit Instance. Open the Enabled Auto Backups section and fill in the details of when to create the backups (see Figure 12.9). You must specify a time range for when automatic backups should occur. You can also enable binary logging, which is needed for more advanced features, such as point-in-time recovery.

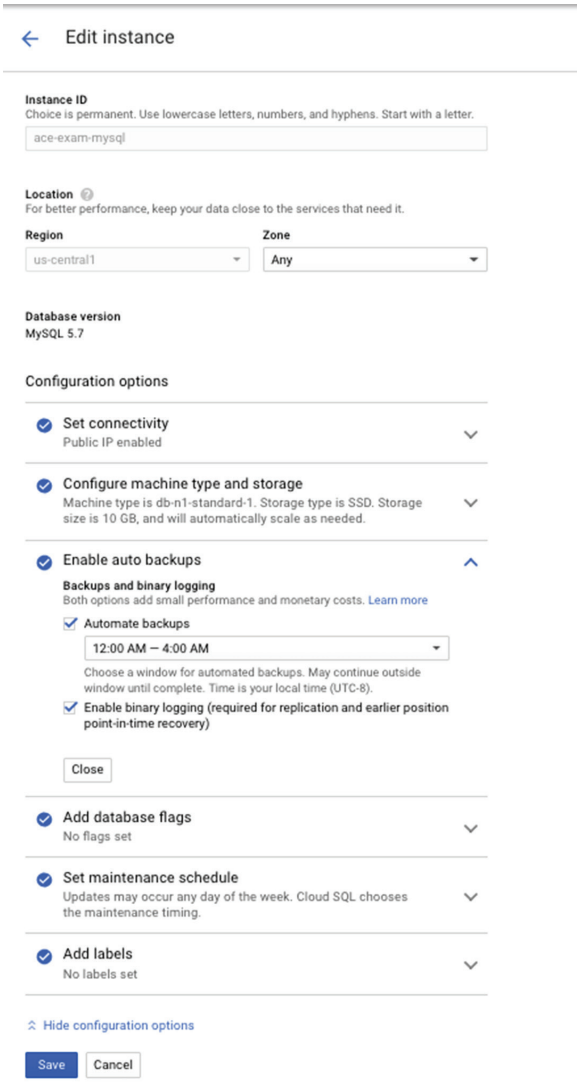
To enable automatic backups from the command line, use the `gcloud` command:

```
gcloud sql instances patch [INSTANCE_NAME] -backup-start-time [HH:MM]
```

For this example instance, you could run automatic backups at 1:00 a.m. with the following command:

```
gcloud sql instances patch ace-exam-mysql -backup-start-time 01:00
```

FIGURE 12.9 Enabling automatic backups in Cloud Console



Deploying and Managing Datastore

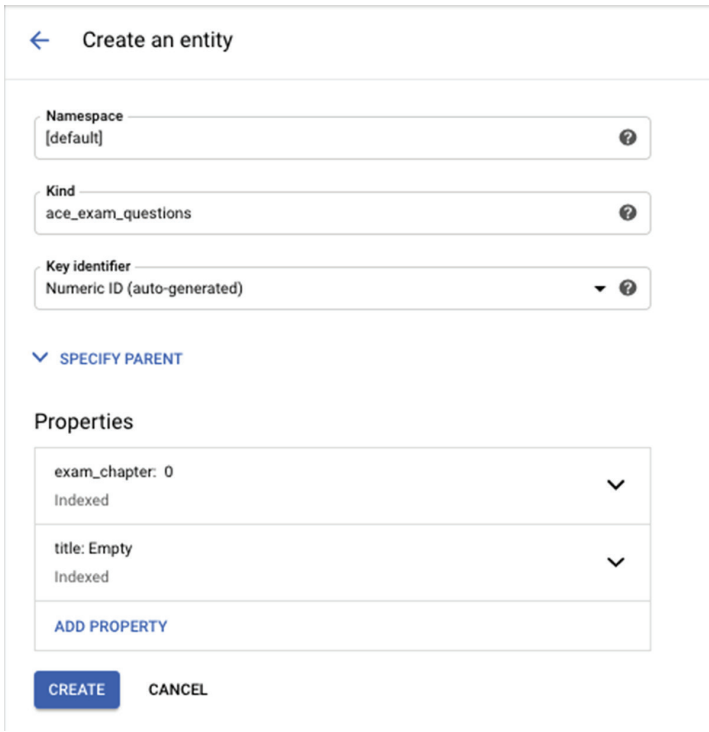
Chapter 11 described how to initialize a Datastore document database. Now, you will see how to create entities and add properties to a document database. You'll also review backup and restore operations.

Adding Data to a Datastore Database

You add data to a Datastore database using the Entities option in the Datastore section of the console. The Entities data structure is analogous to a schema in relational databases.

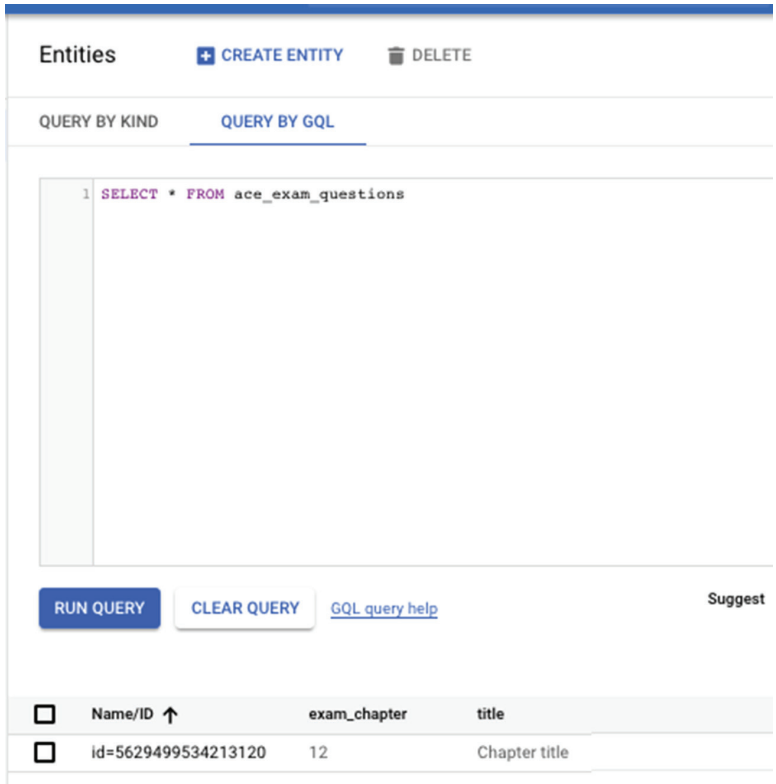
You create an entity by clicking Create Entity and filling in the form that appears. Here you will need to fill in Kind, which is analogous to a table in a relational database, and Properties, as shown in Figure 12.10.

FIGURE 12.10 Adding data to a Datastore entity



After creating entities, you can query the document database using GQL, a query language similar to SQL. Figure 12.11 shows an example query using the `SELECT` command.

FIGURE 12.11 Query data store using GGL, a SQL-like query language



Backing Up Datastore

To back up a Datastore database, you need to create a Cloud Storage bucket to hold a backup file and grant appropriate permissions to users performing backup.

You can create a bucket for backups using the `gsutil` command.

```
gsutil mb gs://[BUCKET_NAME]/
```

Here, *[BUCKET_NAME]* is the name, such as `ace_exam_backups`. In our example, we use `ace_exam_backups` and create that bucket using the following:

```
gsutil mb gs://ace_exam_backups/
```

Users creating backups need the `datastore.databases.export` permission. If you are importing data, you will need `datastore.databases.import`. The Cloud Datastore Import Export Admin role has both permissions; see Chapter 17 for details on assigning roles to users.

The user with the Cloud Datastore Import Export Admin role can now make a backup using the following command:

```
gcloud –namespaces='[NAMESPACE]' gs://[BUCKET_NAME}
```

In this example, the command to create a backup is as follows:

```
gcloud datastore export –namespaces='(default)' gs://ace_exam_backups
```

To import a backup file, use the `gcloud datastore import` command:

```
gcloud datastore import gs://[BUCKET]/[PATH]/[FILE].overall_export_metadata
```

In our example, you can import using this:

```
gcloud datastore import gs://ace_exam_backups/[FILE].overall_export_metadata
```

Here, *[FILE]* is the filename assigned by the export process.

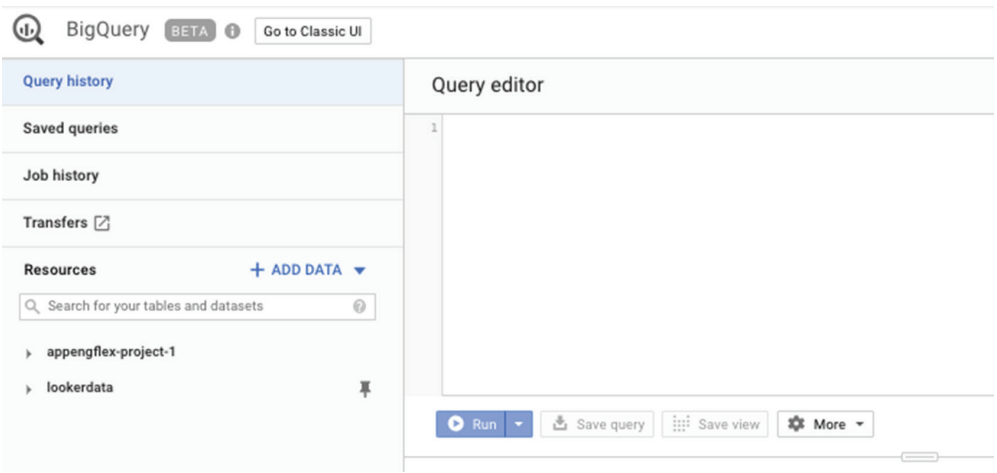
Deploying and Managing BigQuery

BigQuery is a fully managed database service, so Google takes care of backups and other basic administrative tasks. As a Cloud Engineer, you still have some administrative tasks when working with BigQuery. Two of those tasks are estimating the cost of a query and checking on the status of a job.

Estimating the Cost of Queries in BigQuery

In the console, choose BigQuery from the main navigation menu to display the BigQuery query interface, as shown in Figure 12.12.

FIGURE 12.12 The BigQuery user interface. Note that this is a beta version of the new interface; older versions will look different.



In this form you can enter a query in the Query Editor, such as a query about names and genders in the usa_1910_2013 table, as shown in Figure 12.13.

FIGURE 12.13 Example query with estimated amount of data scanned

Notice in the lower-right corner that BigQuery provides an estimate of how much data will be scanned. You can also use the command line to get this estimate by using the `bq` command with the `--dry-run` option.

```
bq --location=[LOCATION] query --use_legacy_sql=false --dry_run [SQL_QUERY]
```

Here, *[Location]* is the location in which you created the data set you are querying, and *[SQL_QUERY]* is the SQL query you are estimating.

You can use this number with the Pricing Calculator to estimate the cost. The Pricing Calculator is available at `https://cloud.google.com/products/calculator/`. After selecting BigQuery, navigate to the On-Demand tab, enter the name of the table you are querying, set the amount of storage to 0, and then enter the size of the query in the Queries line of the Queries Pricing section. Be sure to use the same size unit as displayed in the BigQuery console. In our example, the unit of measure is megabytes. When you click Add To Estimate, the Pricing Calculator will display the cost (see Figure 12.14).

Viewing Jobs in BigQuery

Jobs in BigQuery are processes used to load, export, copy, and query data. Jobs are automatically created when you start any of these operations.

To view the status of jobs, navigate to the BigQuery console and click Job History in the menu to the left. This will display a list of jobs and their status. Notice, in Figure 12.15, that the top job in the list has a green bar, indicating that the job completed successfully. This is an example of an expanded view of a job entry. Below that is a single-line summary of a job that failed. The failure is indicated by the red icon next to the job description.

FIGURE 12.14 Using the Pricing Calculator to estimate the cost of a query

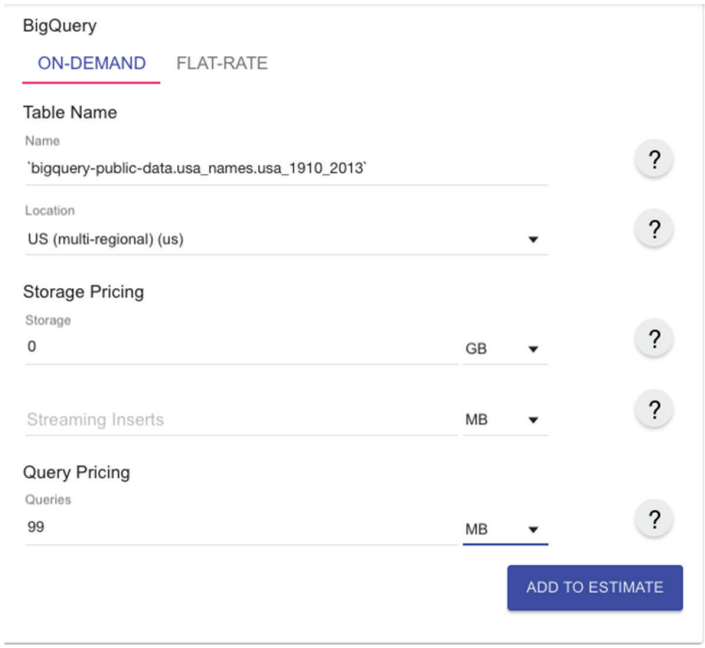
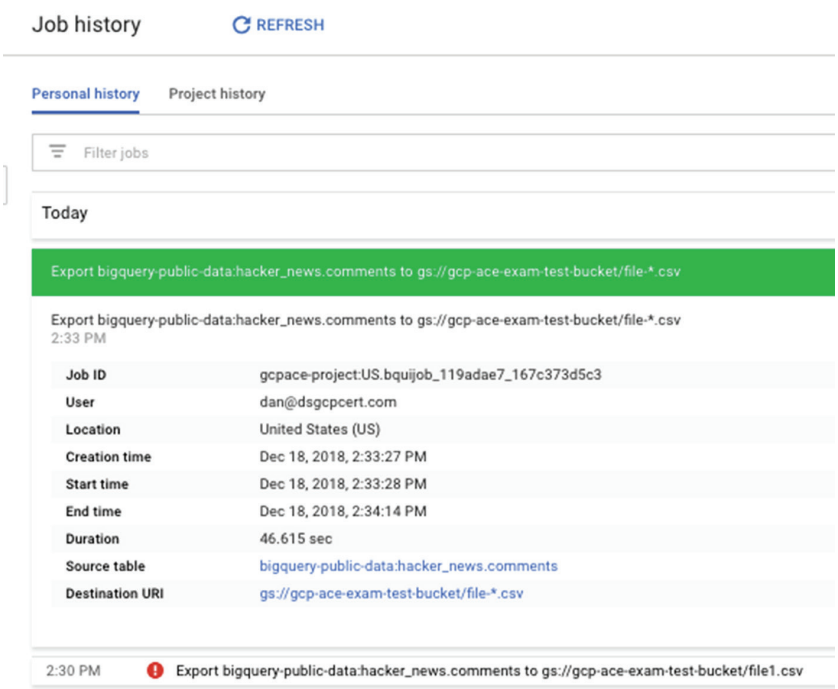


FIGURE 12.15 A listing of job statuses in BigQuery



You could also view the status of a BigQuery job by using the `bq show` command. For example, to show the results of the successful export job shown in Figure 12.15, you could use this command:

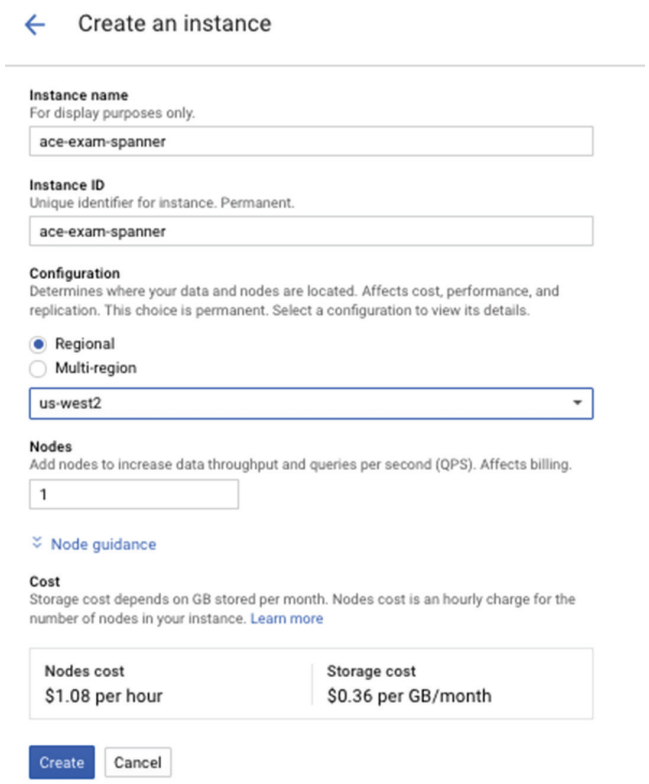
```
bq --location=US show -j gcpace-project:US.bquijob_119adae7_167c373d5c3
```

Deploying and Managing Cloud Spanner

Now, let's turn our attention to Cloud Spanner, the global relational database. In this section, you will create a database, define a schema, insert some data, and then query it.

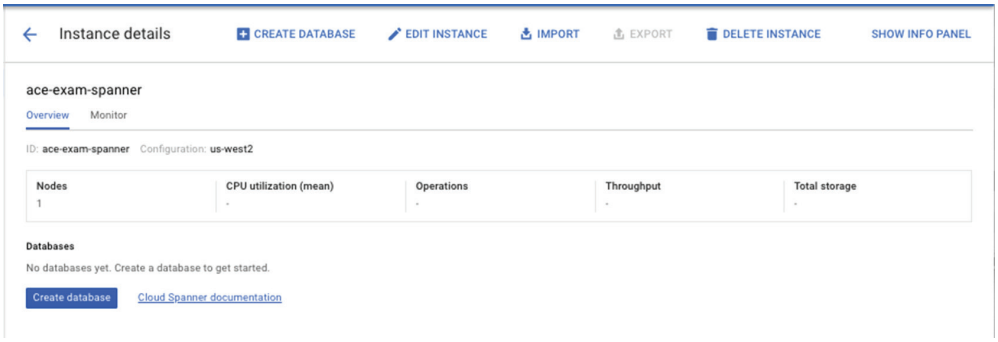
First, you will create a Cloud Spanner instance. Navigate to the Cloud Spanner form in the console and select Create Instance. This will display a form as shown in Figure 12.16.

FIGURE 12.16 Create a Cloud Spanner instance.



Next, you need to create a database in the instance. Select Create Database at the top of the Instance Details page, as shown in Figure 12.17.

FIGURE 12.17 Create a database within a Cloud Spanner instance.



When creating a database, you will need to use the SQL data definition language (DDL) to define the structure of tables. SQL DDL is the set of SQL commands for creating tables, indexes, and other data structures (see Table 12.1). In the example in Figure 12.18, you use a Singers table definition provided by Google in the Cloud Spanner Quickstart (`https://cloud.google.com/spanner/docs/quickstart-console`).

TABLE 12.1 SQL data definition commands

Command	Description
`CREATE TABLE`	Creates a table with columns and data types specified
`CREATE INDEX`	Creates an index on the specified column(s)
`ALTER TABLE`	Changes table structure
`DROP TABLE`	Removes the table from the database schema
`DROP INDEX`	Removes the index from the database schema

After executing the `CREATE TABLE` command, you will see a listing of the table structure, as in Figure 12.19.

FIGURE 12.18 Create a table within the database.

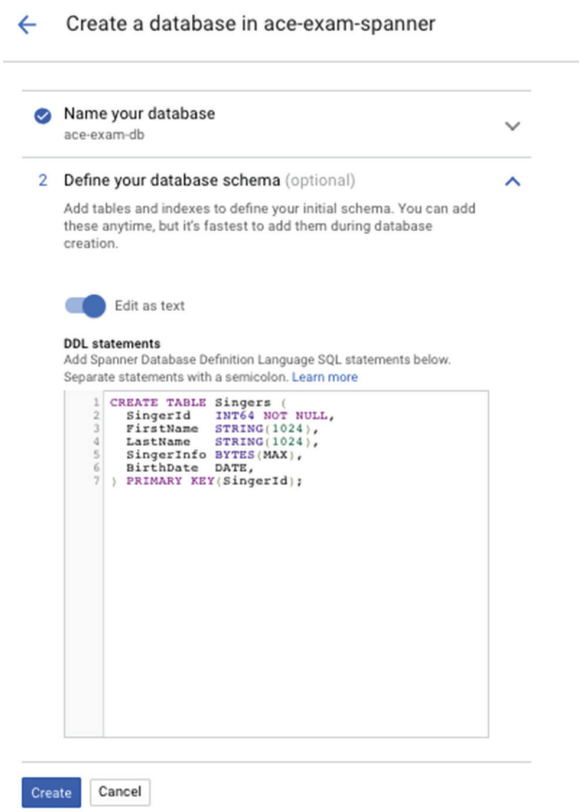
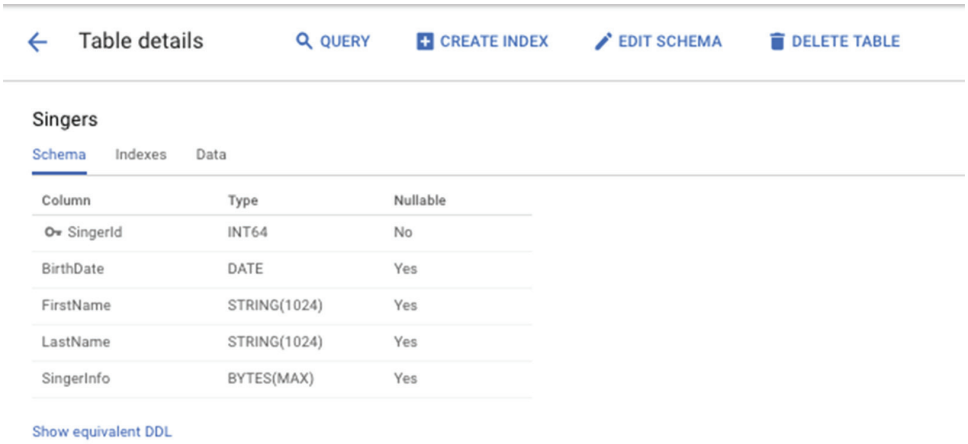
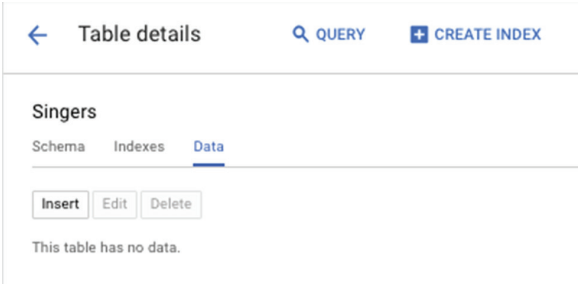


FIGURE 12.19 List of table columns in the table



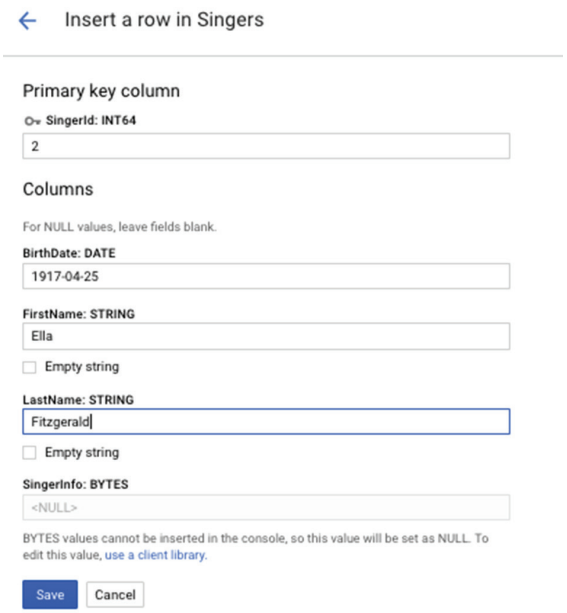
To add data to the table, select the Data table in the Table Details page, as shown in Figure 12.20.

FIGURE 12.20 Select the Data tab to insert data into the table.



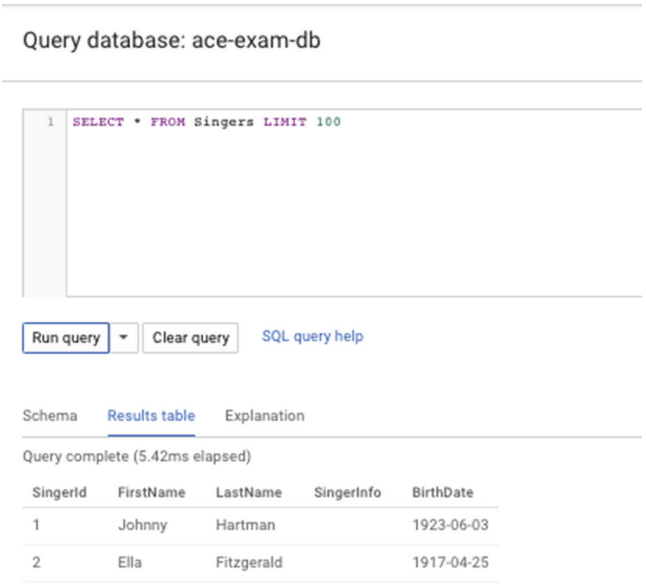
When you add a row, you will see a form like the one in Figure 12.21, which shows the columns in the table. In this example, the columns are SingerID, BirthDate, FirstName, LastName, and SingerInfo.

FIGURE 12.21 Data entered into the table



Finally, you can execute a query by selecting the query from the Table Details page (as shown in Figure 12.22).

FIGURE 12.22 Query a table from the Query form.



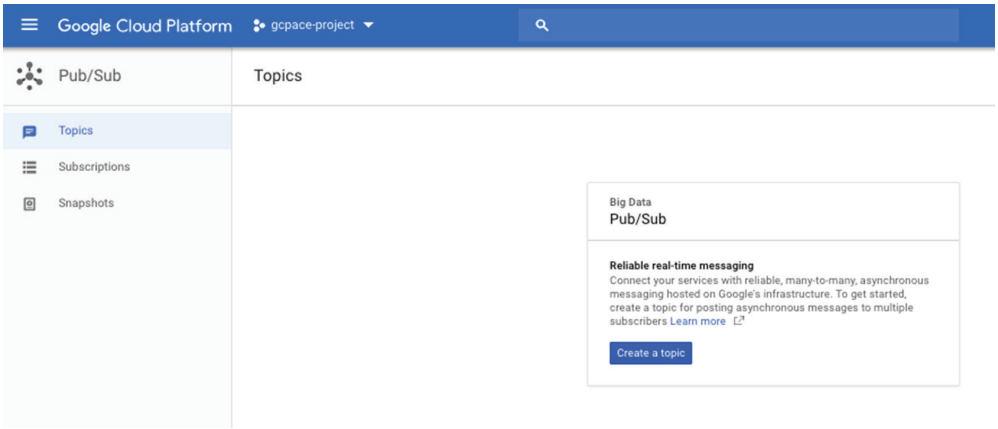
Cloud Spanner is a managed database service, so you will not have to patch, backup, or perform other basic data administration tasks. Your tasks, and those of data modelers and software engineers, will focus on design tables and queries.

Deploying and Managing Cloud Pub/Sub

There are two tasks required to deploy a Pub/Sub message queue: creating a topic and creating a subscription. A topic is a structure where applications can send messages. Pub/Sub receives the messages and keeps them until they are read by an application. Applications read messages by using a subscription.

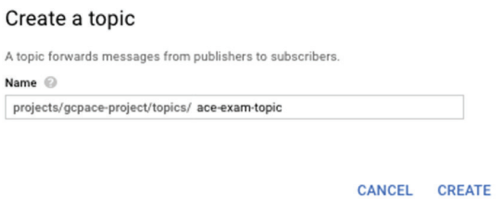
The first step for working with Pub/Sub is to navigate to the Pub/Sub page in Cloud Console. The first time you use Pub/Sub, the form will be similar to Figure 12.23.

FIGURE 12.23 Create a Pub/Sub topic.



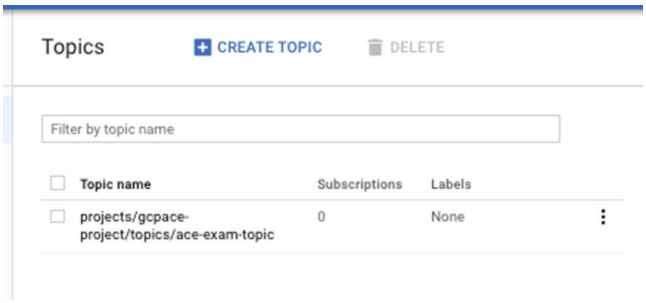
When you click Create a Topic, you will be prompted for a name for the topic, as in Figure 12.24.

FIGURE 12.24 Name a topic.



You will see a list of topics displayed in the Topics page after creating the first topic, as shown in Figure 12.25.

FIGURE 12.25 List of topics



To create a subscription to a topic, click the three-dot icon at the end of the topic summary line in the listing. The menu that appears includes a Create Subscription option

(see Figure 12.26). Click Create Subscription to create a subscription to that topic. This will display a form like that shown in Figure 12.27.

FIGURE 12.26 Creating a subscription to a topic

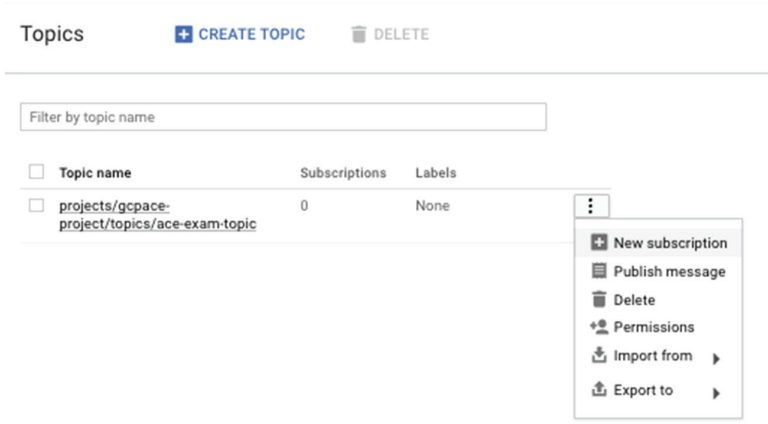
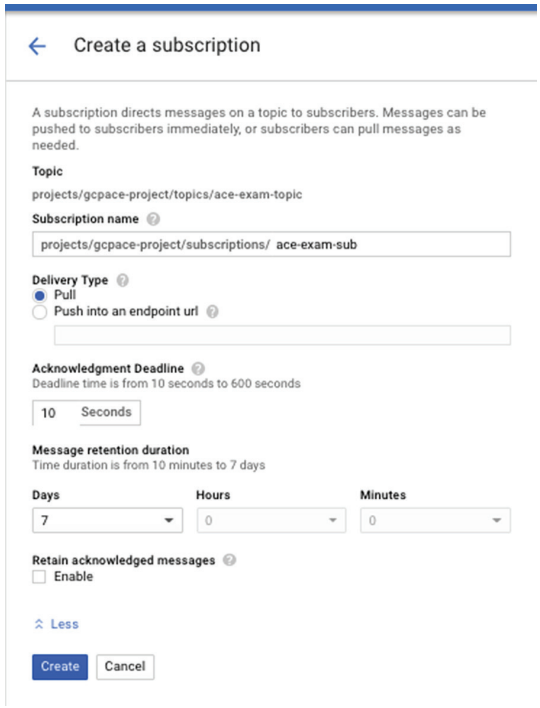


FIGURE 12.27 The form for creating a subscription



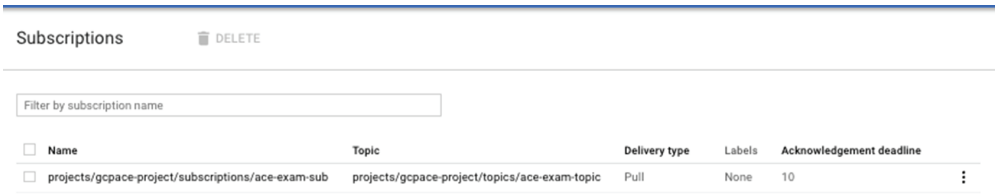
To create a subscription, specify a subscription name and delivery type. Subscriptions can be *pulled*, in which the application reads from a topic, or *pushed*, in which the subscription writes messages to an endpoint. If you want to use a push subscription, you will need to specify the URL of an endpoint to receive the message.

Once a message is read, the application reading the message acknowledges receiving the message. Pub/Sub will wait the period of time specified in the Acknowledgment Deadline parameter. The time to wait can range from 10 to 600 seconds.

You can also specify a retention period, which is the length of time to keep a message that cannot be delivered. After the retention period passes, messages are deleted from the topic.

When you complete creating a subscription, you will see a list of subscriptions like that shown in Figure 12.28.

FIGURE 12.28 A list of subscriptions



In addition to using the console, you can use `gcloud` commands to create topics and subscriptions. The commands to create topics and subscriptions are as follows:

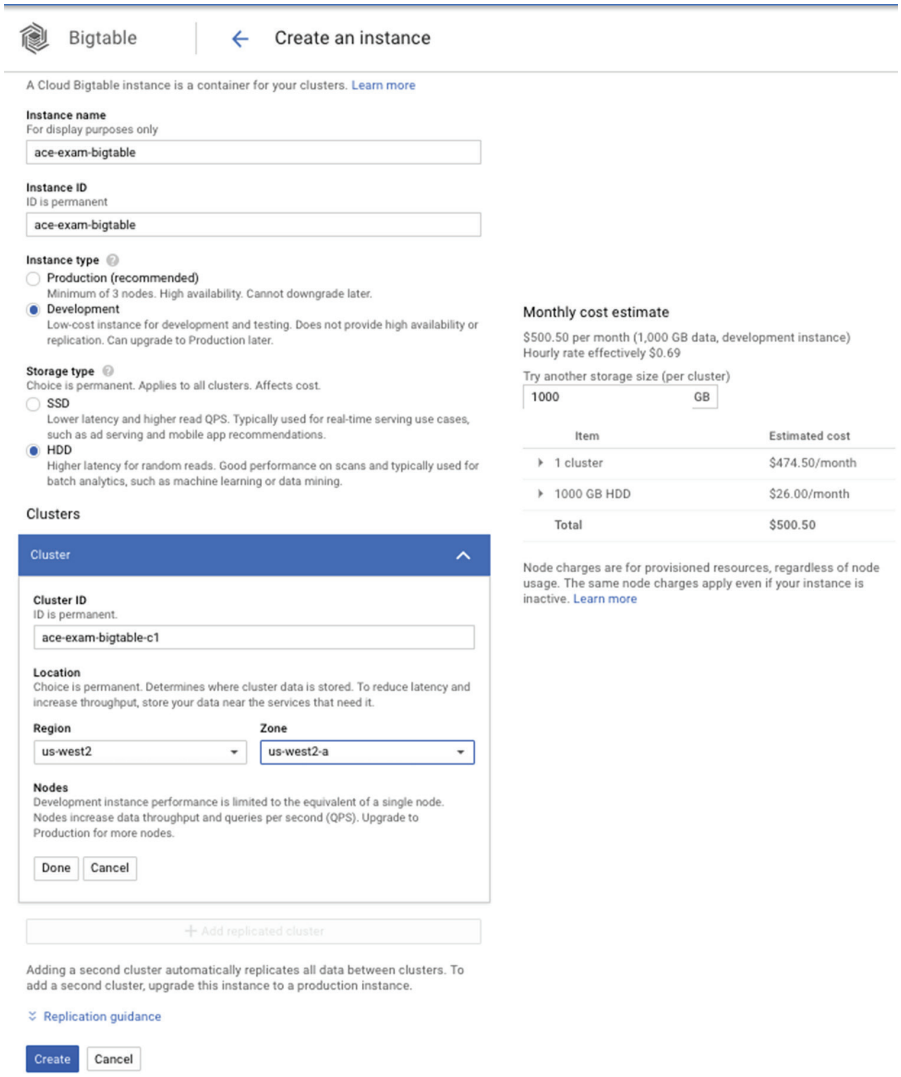
```
gcloud pubsub topics create [TOPIC-NAME]
gcloud pubsub subscriptions create [SUBSCRIPTION-NAME] --topic [TOPIC-NAME]
```

Deploying and Managing Cloud Bigtable

As a Cloud Engineer, you may need to create a Bigtable cluster, or set of servers running Bigtable services, as well as create tables, add data, and query that data.

To create a Bigtable instance, navigate to the Bigtable console and click Create Instance. This will display a form like that shown in Figure 12.29. (See Chapter 11 for additional details on creating a Bigtable instance.)

FIGURE 12.29 Creating a Bigtable instance



Much of the work you will do with Bigtable is done at the command line.

To create a table, open a Cloud Shell browser and install the `cbt` command. Unlike relational databases, Bigtable is a NoSQL database and does not use the SQL command. Instead, the `cbt` command has subcommands to create tables, insert data, and query tables (see Table 12.2).

TABLE 12.2 cbt commands

Command	Description
`createtable`	Creates a table
`createfamily`	Creates a column family
`read`	Reads and displays rows
`ls`	Lists tables and columns

To configure `cbt` in Cloud Shell, enter these commands:

```
gcloud components update
gcloud components install cbt
```

Bigtable requires an environment variable called `instance` to be set by including it in a `.cbt` configuration file called `.cbtrc`, which is kept in the home directory.

For example, to set the instance to `ace-exam-bigtable`, enter this command at the command-line prompt:

```
echo instance = ace-exam-bigtable >> ~/.cbtrc
```

Now `cbt` commands will operate on that instance. To create a table, issue a command such as this:

```
cbt createtable ace-exam-bt-table
```

The `ls` command lists tables. Here's an example:

```
cbt ls
```

This will display a list of all tables. Tables contain columns, but Bigtable also has a concept of column families. To create a column family called `colfam1`, use the following command:

```
cbt createfamily ace-exam-bt-table colfam1
```

To set a value of the cell with the column `colfam1` in a row called `row1`, use the following command:

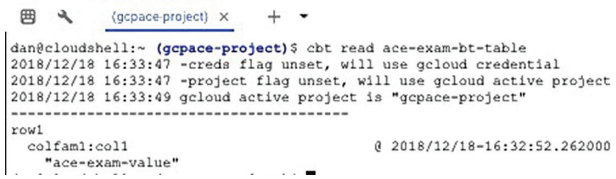
```
cbt set ace-exam-bt-table row1 colfam1:col1=ace-exam-value
```

To display the contents of a table, use a `read` command such as this:

```
cbt read ace-exam-bt-table
```

The `read` command will generate output such as that shown in Figure 12.30.

FIGURE 12.30 Displaying table contents using the `cbt read` command

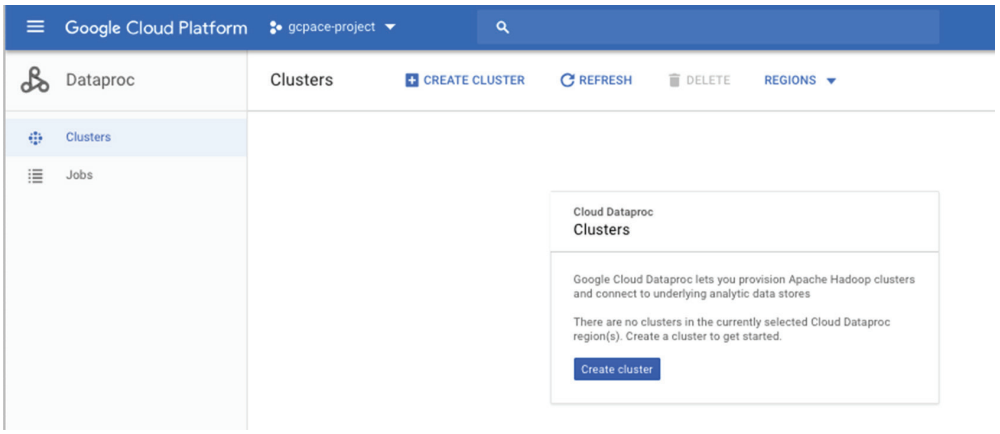


Deploying and Managing Cloud Dataproc

Cloud Dataproc is Google's managed Apache Spark and Apache Hadoop service. Both Spark and Hadoop are designed for "big data" applications. Spark supports analysis and machine learning, while Hadoop is well suited to batch, big data applications. As a Cloud Engineer, you should be familiar with creating a Dataproc cluster and submitting jobs to run in the cluster.

To create a cluster, navigate to the Dataproc part of Cloud Console (see Figure 12.31).

FIGURE 12.31 Dataproc console page

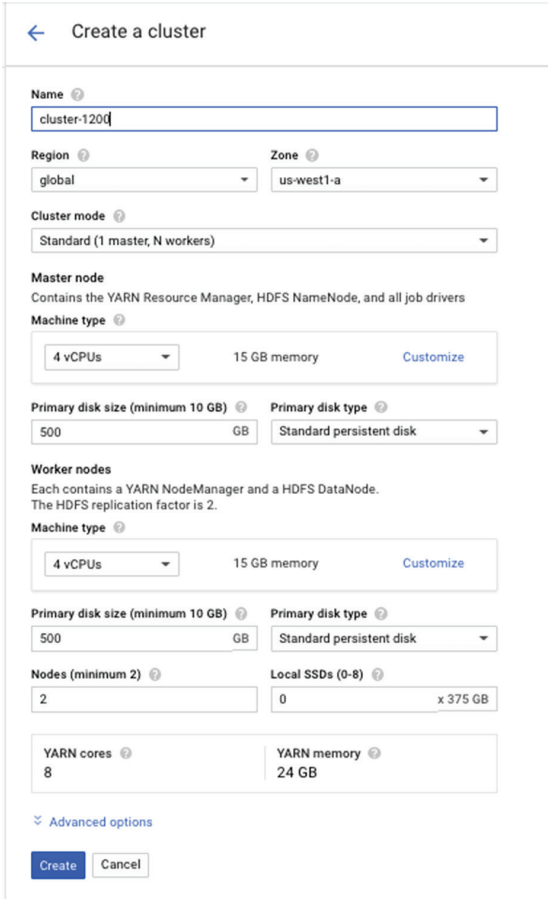


Create a Dataproc cluster by filling in the Create Cluster form. You will need to specify the name of the cluster and a region and zone. You'll also need to specify the cluster mode,

which can be single node, standard, or high availability. Single node is useful for development. Standard has only one master node, so if it fails, the cluster becomes inaccessible. The high availability mode uses three masters.

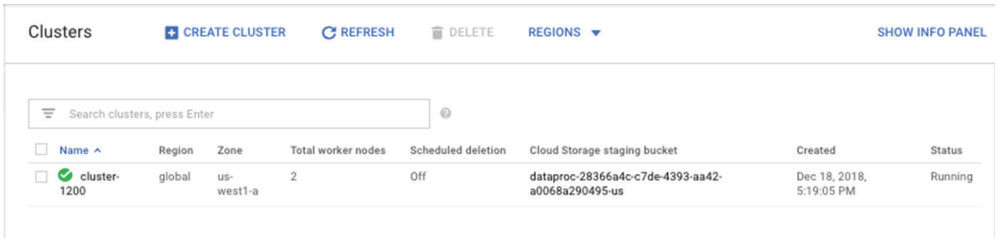
You will also need to specify machine configuration information for the master nodes and the worker nodes. You'll specify CPUs, memory, and disk information. The cluster mode determines the number of master nodes, but you can choose the number of worker nodes. If you choose to expand the list of advanced options, you can indicate you'd like to use preemptible VMs and specify a number of preemptible VMs to run (see Figure 12.32).

FIGURE 12.32 Create a Dataproc cluster.



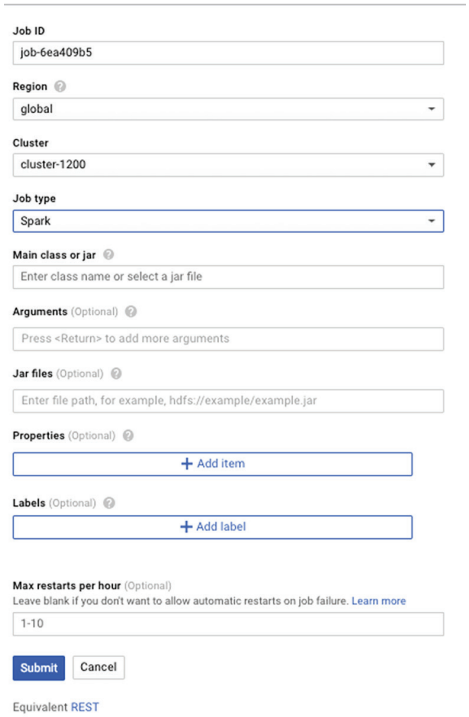
After you create a cluster, it will appear in the list of clusters, as in Figure 12.33.

FIGURE 12.33 Listing of Dataproc clusters



When the cluster is running, you can submit jobs using the Jobs form shown in Figure 12.34.

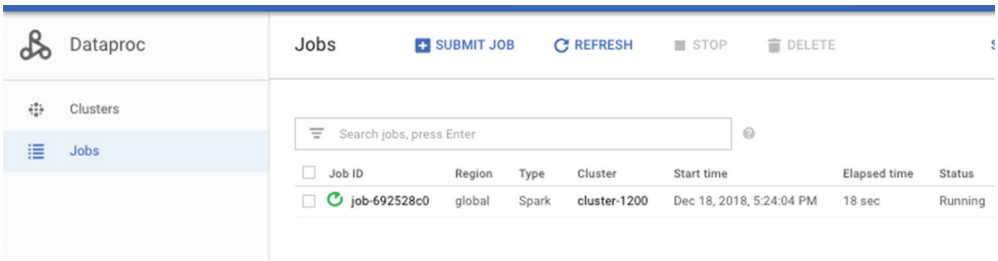
FIGURE 12.34 Submit a job from the Cluster Details page.



You will need to specify the cluster to run the job and the type of job, which can be either Spark, PySpark, SparkR, Hive, Spark SQL, Pig, or Hadoop. The JAR files are the Java programs that will be executed, and the Main Class or JAR is the name of the

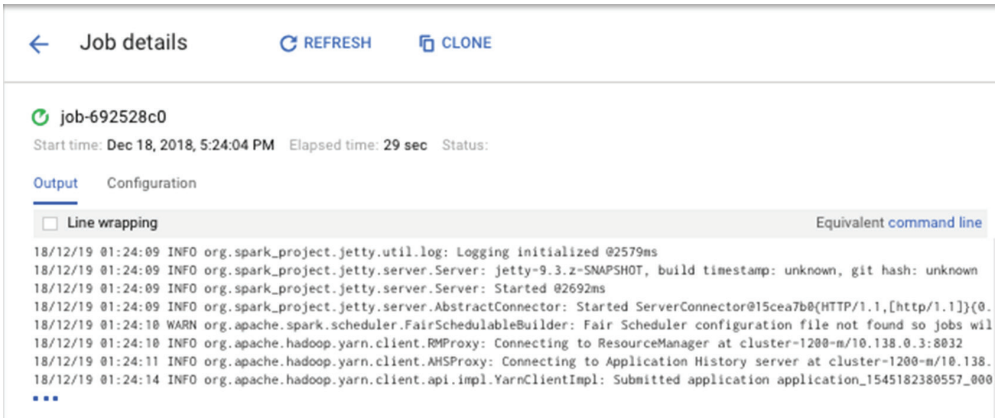
function or method that should be invoked to start the job. If you choose PySpark, you will submit a Python program; if you submit SparkR, you will submit an R program. When running Hive or SparkSQL, you will submit query files. You can also pass in optional arguments. Once the job is running, you will see it in the jobs listing page, as shown in Figure 12.35.

FIGURE 12.35 Listing of jobs



Double-clicking Jobs ID in the listing will display details of the job log (see Figure 12.36).

FIGURE 12.36 Logging detail of a running job



In addition to using the console, you can create a cluster using the `gcloud dataproc clusters` command. Here's an example:

```
gcloud dataproc clusters create cluster-bc3d --zone us-west2-a
```

This will create a default cluster in the us-west2-a zone. You can also specify additional parameters for machine types, disk configurations, and other cluster characteristics.

You use the `gcloud dataproc jobs` command to submit jobs from the command line. Here's an example:

```
gcloud dataproc jobs submit spark --cluster cluster-bc3d  --jar ace_exam_jar.jar
```

This will submit a job running the `ace_exam_jar.jar` program on the `cluster-bc3d` cluster.

Real World Scenario

Spark for Machine Learning

Retailers collect large volumes of data about shoppers' purchases, and this is especially helpful for understanding customers' preferences and interests. The transaction processing systems that collect much of this data are not designed to analyze large volumes of data. For example, if retailers wanted to recommend products to customers based on their interests, they could build machine learning models trained on their sales data. Spark has a machine learning library, called MLlib, that is designed for just this kind of problem. Engineers can export data from transaction processing systems, load it into Spark, and then apply a variety of machine learning algorithms, such as clustering and collaborative filtering, for recommendations. The output of these models includes products that are likely to be of interest to particular customers. It's applications like these that drive the adoption of Spark and other analytics platforms.

Managing Cloud Storage

In Chapter 11, you saw how to use lifecycle management policies to automatically change a bucket's storage class. For example, you could create a policy to change a regional storage class bucket to a nearline bucket after 90 days. There may be times, however, when you would like to manually change a bucket's storage class. In those cases, you can use the `gsutil rewrite` command and specify the `-s` flag. Here's an example:

```
gsutil rewrite -s [STORAGE_CLASS] gs://[PATH_TO_OBJECT]
```

Here, *`[STORAGE_CLASS]`* is the new storage class. It can be `multi_regional`, `regional`, `nearline`, or `coldline`.

It is not possible to change a bucket's storage class in the console.

Another common task with Cloud Storage is moving objects between buckets. You can do this using the `gsutil mv` command. The form of the command is as follows:

```
gsutil mv gs://[SOURCE_BUCKET_NAME]/[SOURCE_OBJECT_NAME] \
gs://[DESTINATION_BUCKET_NAME]/[DESTINATION_OBJECT_NAME]
```

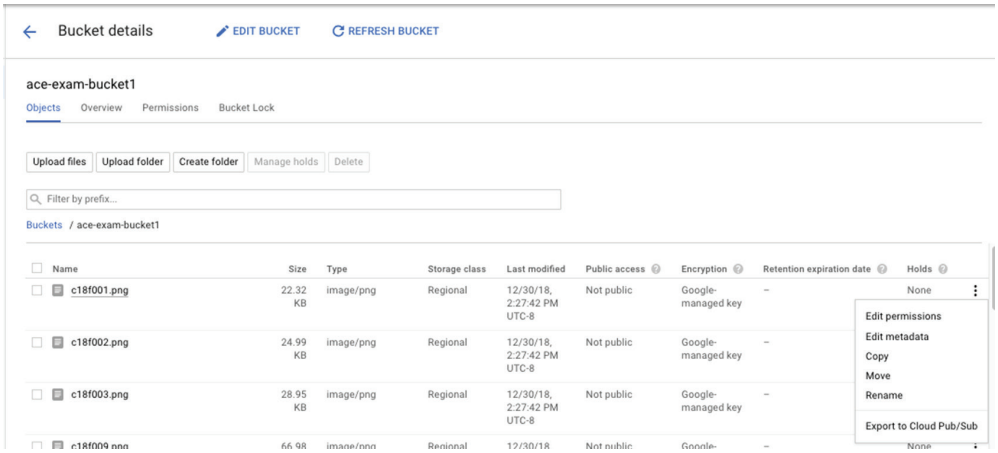
Here, *[SOURCE_BUCKET_NAME]* and *[SOURCE_OBJECT_NAME]* are the original bucket name and filename, and *[DESTINATION_BUCKET_NAME]* and *[DESTINATION_OBJECT_NAME]* are the target bucket and filename, respectively.

The `move` command can also be used to rename an object, similar to the `mv` command in Linux. For an object in Cloud Storage, you can use this command:

```
gsutil mv gs://[BUCKET_NAME]/[OLD_OBJECT_NAME] gs://[BUCKET_NAME]/
[NEW_OBJECT_NAME]
```

You can also use the console to move and rename objects in a bucket. Navigate to the Cloud Storage section of the console and browse to a bucket, as shown in Figure 12.37. Click the three-dot icon at the end of the object description. This displays a list of operations, including renaming and moving.

FIGURE 12.37 Renaming and moving an object from the console



Summary

In this chapter, you learned how to perform basic deployment and management tasks for a number of GCP services, including Cloud SQL, Cloud Datastore, BigQuery, Bigtable, Cloud Spanner, Cloud Pub/Sub, Cloud Dataproc, and Cloud Storage. You have seen how to use the console and command-line tools. While `gcloud` is often used, several of the services have their own command-line tools. There was some discussion of how to create database

structures, insert data, and query that data in the various database services. We also discussed basic Cloud Storage management operations, such as moving and renaming objects.

Exam Essentials

Understand how to initialize Cloud SQL and Cloud Spanner. Cloud SQL and Cloud Spanner are the two managed relational databases for transaction processing systems. BigQuery is relational but designed for data warehouses and analytics. Understand the need to create databases and tables. Know that SQL is used to query these databases.

Understand how to initialize Cloud Datastore and Cloud Bigtable. These are two NoSQL offerings. You can add small amounts of data to Cloud Datastore through the console and query it with a SQL-like language called GQL. Cloud Bigtable is a wide-column database that does not support SQL. Bigtable is managed with the `cbt` command-line tool.

Know how to export data from BigQuery, estimate the cost of a query, and monitor jobs in BigQuery. BigQuery is designed to work with petabyte-scale data warehouses. SQL is used to query data. Know how to export data using the console. Understand that the `bq` command line, not `gcloud`, is the tool for working with BigQuery from the command line.

Know how to convert Cloud Storage bucket storage classes. Lifecycle policies can change storage classes of buckets when events occur, such as a period of time passes. Know that `gsutil rewrite` is used to change the storage class of a bucket interactively. Know how to use the console and the command line to move and rename objects.

Understand that Pub/Sub is a message queue. Applications write data to topics, and applications receive messages through subscriptions to topics. Subscriptions can be push or pull. Unread messages have a retention period after which they are deleted.

Understand that Cloud Dataproc is a managed Spark and Hadoop service. These platforms are used for big data analytics, machine learning, and large-scale batch jobs, such as large volume extraction, transformation, and load operations. Spark is a good option for analyzing transaction data, but data must be loaded into Spark from its source system.

Know the four command-line tools: `gcloud`, `gsutil`, `bq`, and `cbt`. `gcloud` is used for most products but not all. `gsutil` is used to work with Cloud Storage from the command line. If you want to work with BigQuery from the command line, you need to use `bq`. To work with Bigtable, you use the `cbt` command.

Review Questions

You can find the answers in the Appendix.

1. Cloud SQL is a fully managed relational database service, but database administrators still have to perform some tasks. Which of the following tasks do Cloud SQL users need to perform?
   - **A.** Applying security patches
   - **B.** Performing regularly scheduled backups
   - **C.** Creating databases
   - **D.** Tuning the operating system to optimize Cloud SQL performance

2. Which of the following commands is used to create a backup of a Cloud SQL database?
   - **A.** `gcloud sql backups create`
   - **B.** `gsutil sql backups create`
   - **C.** `gcloud sql create backups`
   - **D.** `gcloud sql backups export`

3. Which of the following commands will run an automatic backup at 3:00 a.m. on an instance called `ace-exam-mysql`?
   - **A.** `gcloud sql instances patch ace-exam-mysql --backup-start-time 03:00`
   - **B.** `gcloud sql databases patch ace-exam-mysql --backup-start-time 03:00`
   - **C.** `cbt sql instances patch ace-exam-mysql --backup-start-time 03:00`
   - **D.** `bq gcloud sql instances patch ace-exam-mysql --backup-start-time 03:00`

4. What is the query language used by Datastore?
   - **A.** SQL
   - **B.** MDX
   - **C.** GQL
   - **D.** DataFrames

5. What is the correct command-line structure to export data from Datastore?
   - **A.** `gcloud datastore export '[NAMESPACE]' gs://[BUCKET_NAME]`
   - **B.** `gcloud datastore export gs://[BUCKET_NAME]`
   - **C.** `gcloud datastore export --namespaces='[NAMESPACE]' gs://[BUCKET_NAME]`
   - **D.** `gcloud datastore dump --namespaces='[NAMESPACE]' gs://[BUCKET_NAME]`

6. When you enter a query into the BigQuery query form, BigQuery analyzes the query and displays an estimate of what metric?
   - **A.** Time required to enter the query
   - **B.** Cost of the query
   - **C.** Amount of data scanned
   - **D.** Number of bytes passed between servers in the BigQuery cluster

7. You want to get an estimate of the volume of data scanned by BigQuery from the command line. Which option shows the command structure you should use?
   - **A.** `gcloud BigQuery query estimate [SQL_QUERY]`
   - **B.** `bq --location=[LOCATION] query --use_legacy_sql=false --dry_run [SQL_QUERY]`
   - **C.** `gsutil --location=[LOCATION] query --use_legacy_sql=false --dry_run [SQL_QUERY]`
   - **D.** `cbt BigQuery query estimate [SQL_QUERY]`

8. You are using Cloud Console and want to check on some jobs running in BigQuery. You navigate to the BigQuery part of the console. Which menu item would you click to view jobs?
   - **A.** Job History.
   - **B.** Active Jobs.
   - **C.** My Jobs.
   - **D.** You can't view job status in the console; you have to use `bq` on the command line.

9. You want to estimate the cost of running a BigQuery query. What two services within Google Cloud Platform will you need to use?
   - **A.** BigQuery and Billing
   - **B.** Billing and Pricing Calculator
   - **C.** BigQuery and Pricing Calculator
   - **D.** Billing and Pricing Calculator

10. You have just created a Cloud Spanner instance. You have been tasked with creating a way to store data about a product catalog. What is the next step after creating a Cloud Spanner instance that you would perform to enable you to load data?
    - **A.** Run `gcloud spanner update-security-patches`.
    - **B.** Create a database within the instance.
    - **C.** Create tables to hold the data.
    - **D.** Use the Cloud Spanner console to import data into tables created with the instance.

11. You have created a Cloud Spanner instance and database. According to Google best practices, how often should you update VM packages using `apt-get`?
    A. Every 24 hours.
    B. Every 7 days.
    C. Every 30 days.
    D. Never, Cloud Spanner is a managed service.

12. Your software team is developing a distributed application and wants to send messages from one application to another. Once the consuming application reads a message, it should be deleted. You want your system to be robust to failure, so messages should be available for at least three days before they are discarded. Which GCP service is best designed to support this use case?
    A. Bigtable
    B. Dataproc
    C. Cloud Pub/Sub
    D. Cloud Spanner

13. Your manager asks you to set up a bare-bones Pub/Sub system as a sandbox for new developers to learn about messaging systems. What are the two resources within Pub/Sub you will need to create?
    A. Topics and tables
    B. Topics and databases
    C. Topics and subscriptions
    D. Tables and subscriptions

14. Your company is launching an IoT service and will receive large volumes of streaming data. You have to store this data in Bigtable. You want to explore the Bigtable environment from the command line. What command would you run to ensure you have command-line tools installed?
    A. `apt-get install bigtable-tools`
    B. `apt-get install cbt`
    C. `gcloud components install cbt`
    D. `gcloud components install bigtable-tools`

15. You need to create a table called `iot-ingest-data` in Bigtable. What command would you use?
    A. `cbt createtable iot-ingest-data`
    B. `gcloud bigtable tables create ace-exam-bt-table`
    C. `gcloud bigtable create tables ace-exam-bt-table`
    D. `gcloud create ace-exam-bt-table`

16. Cloud Dataproc is a managed service for which two big data platforms?
    A. Spark and Cassandra
    B. Spark and Hadoop
    C. Hadoop and Cassandra
    D. Spark and TensorFlow

17. Your department has been asked to analyze large batches of data every night. The jobs will run for about three to four hours. You want to shut down resources as soon as the analysis is done, so you decide to write a script to create a Dataproc cluster every night at midnight. What command would you use to create a cluster called `spark-nightly-analysis` in the `us-west2-a zone`?
    A. `bq dataproc clusters create spark-nightly-analysis --zone us-west2-a`
    B. `gcloud dataproc clusters create spark-nightly-analysis --zone us-west2-a`
    C. `gcloud dataproc clusters spark-nightly-analysis --zone us-west2-a`
    D. None of the above

18. You have a number of buckets containing old data that is hardly ever used. You don't want to delete it, but you want to minimize the cost of storing it. You decide to change the storage class to coldline for each of those buckets. What is the command structure that you would use?
    A. `gcloud rewrite -s [STORAGE_CLASS] gs://[PATH_TO_OBJECT]`
    B. `gsutil rewrite -s [STORAGE_CLASS] gs://[PATH_TO_OBJECT]`
    C. `cbt rewrite -s [STORAGE_CLASS] gs://[PATH_TO_OBJECT]`
    D. `bq rewrite -s [STORAGE_CLASS] gs://[PATH_TO_OBJECT]`

19. You want to rename an object stored in a bucket. What command structure would you use?
    A. `gsutil cp gs://[BUCKET_NAME]/[OLD_OBJECT_NAME] gs://[BUCKET_NAME]/[NEW_OBJECT_NAME]`
    B. `gsutil mv gs://[BUCKET_NAME]/[OLD_OBJECT_NAME] gs://[BUCKET_NAME]/[NEW_OBJECT_NAME]`
    C. `gsutil mv gs://[OLD_OBJECT_NAME] gs://[NEW_OBJECT_NAME]`
    D. `gcloud mv gs://[OLD_OBJECT_NAME] gs://[NEW_OBJECT_NAME]`

20. An executive in your company emails you asking about creating a recommendation system that will help sell more products. The executive has heard there are some GCP solutions that may be good fits for this problem. What GCP service would you recommend the executive look into?
    A. Cloud Dataproc, especially Spark and its machine learning library
    B. Cloud Dataproc, especially Hadoop
    C. Cloud Spanner, which is a global relational database that can hold a lot of data
    D. Cloud SQL, because SQL is a powerful query language

Chapter 13

Loading Data into Storage

THIS CHAPTER COVERS THE FOLLOWING OBJECTIVES OF THE GOOGLE ASSOCIATE CLOUD ENGINEER CERTIFICATION EXAM:

✓ **3.4 Deploying and implementing data solutions**

Official Google Cloud Certified: Associate Cloud Engineer: Study Guide
By Dan Sullivan

In this chapter, we will delve into the details of loading and moving data into various storage and processing systems in Google Cloud Platform. We'll start by explaining how to load and move data in Cloud Storage using the console and the command line.

The bulk of the chapter will describe how to import and export data into data storage and analysis services, including Cloud SQL, Cloud Datastore, BigQuery, Cloud Spanner, Cloud Bigtable, and Cloud Dataproc. The chapter wraps up with a look into streaming data into Cloud Pub/Sub.

Loading and Moving Data to Cloud Storage

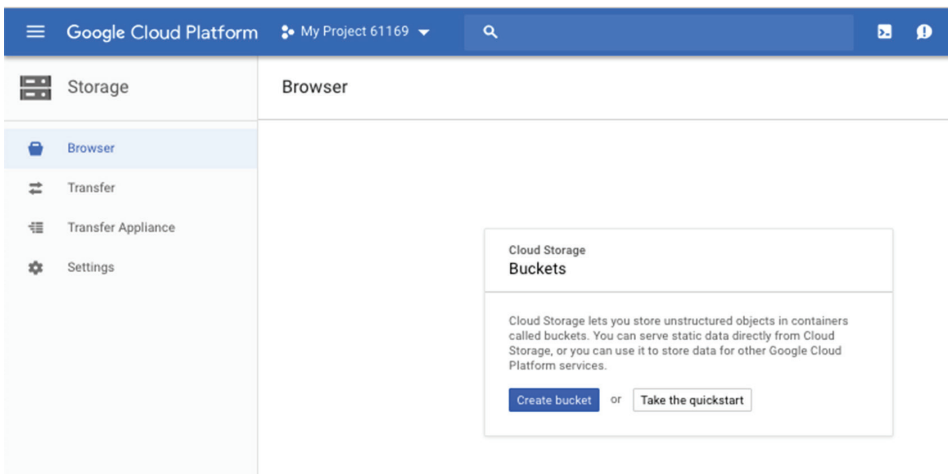
Cloud Storage is used for a variety of storage use cases, including long-term storage and archiving, file transfers, and data sharing. This section describes how to create storage buckets, load data into storage buckets, and move objects between storage buckets.

Loading and Moving Data to Cloud Storage Using the Console

Loading data into Cloud Storage is a common task and easily done using Cloud Console.

Navigate to the Cloud Storage page of Cloud Console. You will see either a list of existing buckets or an option to create a new bucket (see Figure 13.1).

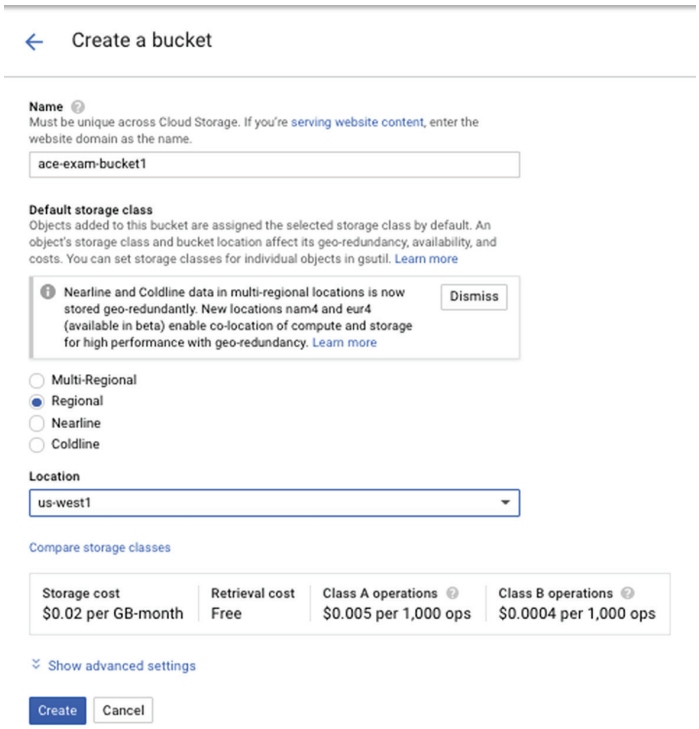
FIGURE 13.1 The first step in loading data into Cloud Storage is to create a bucket.



When you create a bucket, you are prompted to specify the storage class of the bucket and the region to store it in, as shown in Figure 13.2.

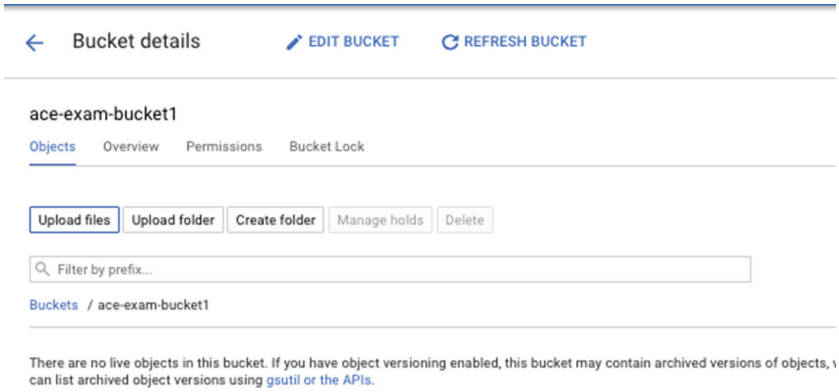
For the exam, remember that buckets are regional resources. Buckets are replicated across zones in the region.

FIGURE 13.2 Defining a regional bucket in us-west1



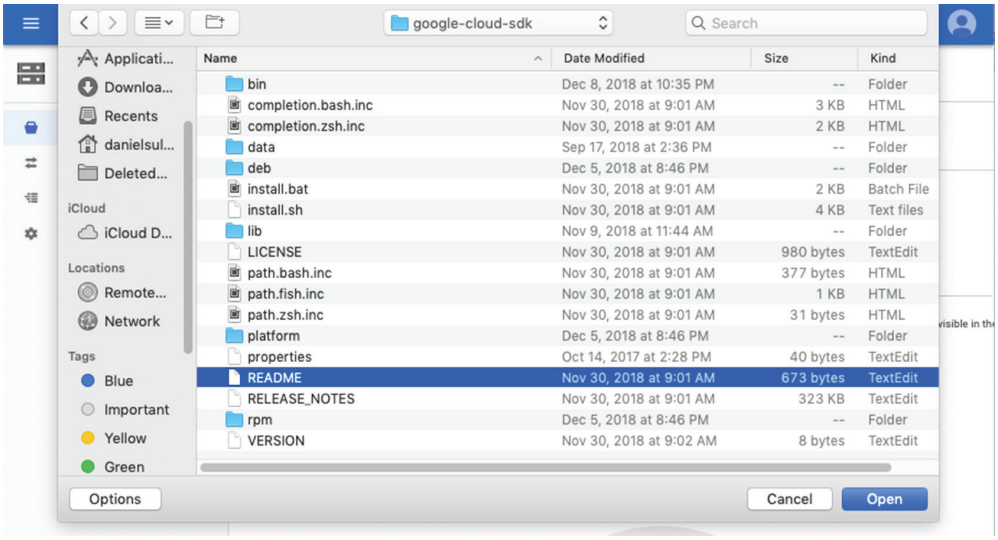
After you create a bucket, you see the Bucket Details page (see Figure 13.3). From here, you can upload individual files or folders.

FIGURE 13.3 From Bucket Details page, you can upload files and folders.



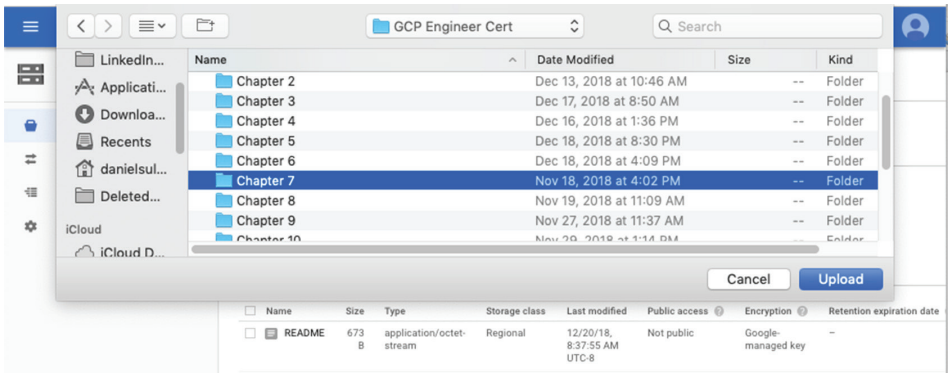
When you upload a file, you are prompted to do so using your client devices file system. Figure 13.4 shows a macOS Finder window for uploading files. You would see something similar on a Windows or Linux operating system.

FIGURE 13.4 Choosing the File Upload option prompts for a file using the client device file system tools.



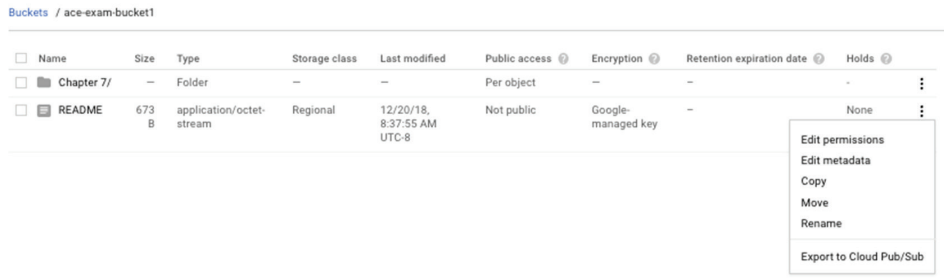
When you upload a folder, you are also prompted by your local operating system tools (see Figure 13.5).

FIGURE 13.5 Choosing the Folder Upload option works similarly to File Upload; you are prompted for a folder using the client device file system tools.



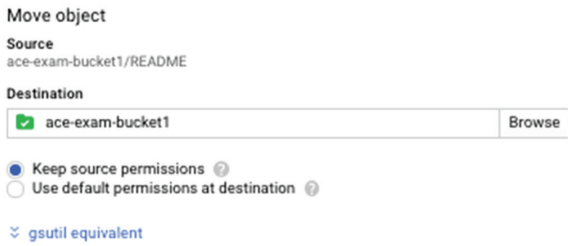
It's easy to move objects between buckets. Just click the three-dot option at the end of a line about an object to display a list of operations, which includes Move. Selecting Move will open a dialog form like that shown in Figure 13.6.

FIGURE 13.6 Objects can be moved by using the `move` command in the Operations menu.



When moving an object, you are prompted for a destination bucket and folder, as shown in Figure 13.7.

FIGURE 13.7 When moving an object in the console, you will be prompted for a destination bucket and folder.



You can move objects using the Move operation in the pop-up menu, but you cannot move a folder this way. Move is not an option in the pop-up-menu when selecting a folder.

Loading and Moving Data to Cloud Storage Using the Command Line

Loading and moving data can be done in the command line using the `gsutil` command.

To create a bucket, use the `gsutil mb` command; "mb" is short for "make bucket."

```
gsutil mb gs://[BUCKET_NAME]/
```

Keep in mind that bucket names must be globally unique. To create a bucket named `ace-exam-bucket1`, you would use the following command:

```
gsutil mb gs://ace-exam-bucket1/
```

To upload a file from your local device or a GCP virtual machine (VM), you can use the `gsutil cp` command to copy files. The command is as follows:

```
gsutil cp [LOCAL_OBJECT_LOCATION] gs://[DESTINATION_BUCKET_NAME]/
```

For example, to copy a file called `README.txt` from `/home/mydir` to the bucket `ace-exam-bucket1`, you'd execute the following command from your client device command line:

```
gsutil cp /home/mydir/README.txt gs://ace-exam-bucket1/
```

Similarly, if you'd like download a copy of your data from a Cloud Storage bucket to a directory on a VM, you could log into the VM using SSH and issue a command such as this:

```
gsutil cp gs://ace-exam-bucket1/README.txt /home/mydir/
```

In this example, the source object is on Cloud Storage, and the target file is on the VM from which you are running the command.

The `gsutil` tool has a `move` command; its structure is as follows:

```
gsutil mv gs://[SOURCE_BUCKET_NAME]/[SOURCE_OBJECT_NAME] \
         gs://[DESTINATION_BUCKET_NAME]/[DESTINATION_OBJECT_NAME]
```

To move the `README.txt` file from `ace-exam-bucket1` to `ace-exam-bucket2` and keep the same filename, you'd use this command:

```
gsutil mv gs://ace-exam-bucket1/README.txt  gs://ace-exam-bucket2/
```

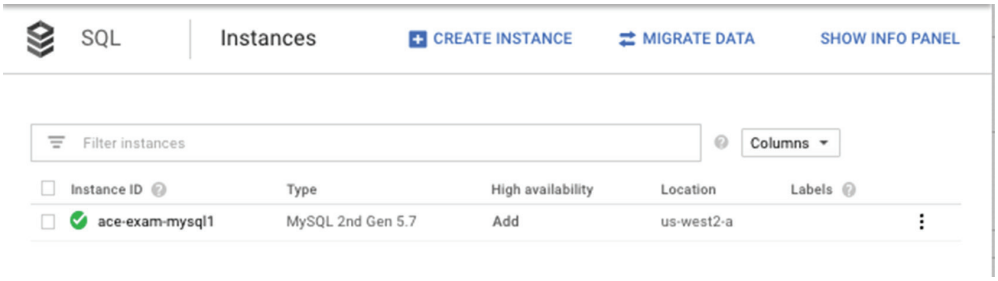
Importing and Exporting Data

As a Cloud Engineer, you may need to perform bulk data operations, such as importing and exporting data from databases. These operations are done with command-line tools and sometimes the console. We will not look into how to programmatically insert data into databases; that is more of an application developer and database administrator task.

Importing and Exporting Data: Cloud SQL

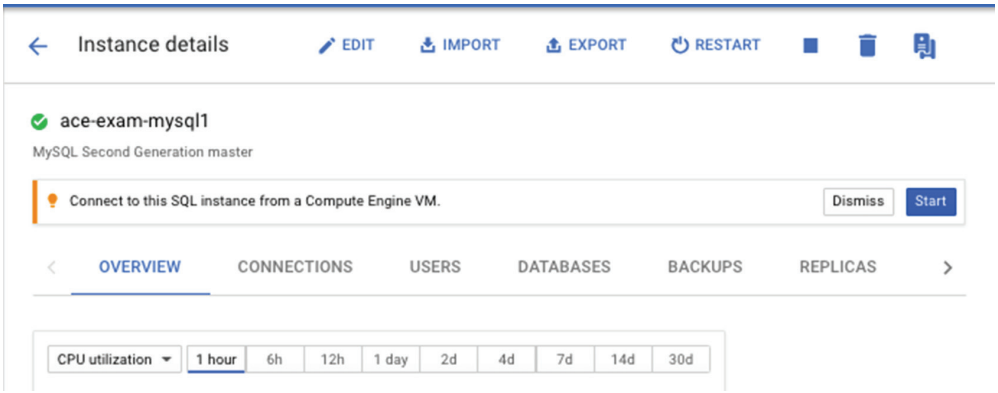
To export a Cloud SQL database using the console, navigate to the Cloud SQL page of the console to list database instances, as in Figure 13.8.

FIGURE 13.8 Listing of database instances on the Cloud SQL page of the console



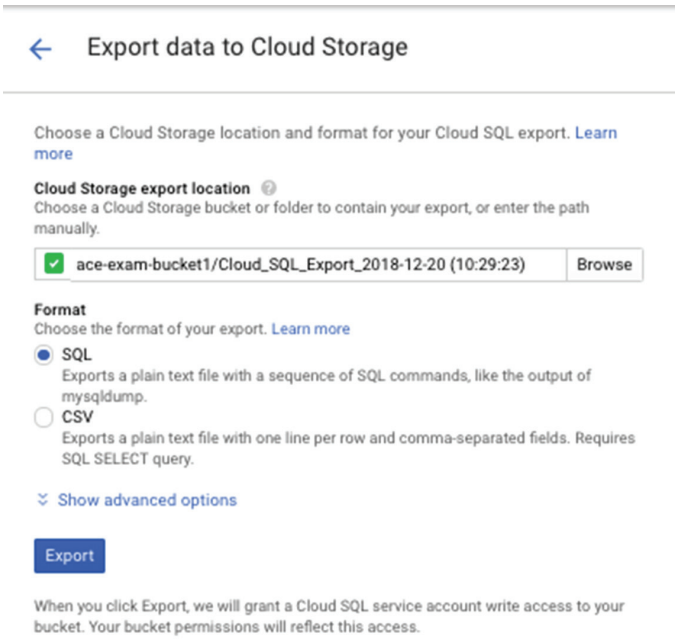
Open the Instance Detail page by double-clicking the name of the instance (see Figure 13.9).

FIGURE 13.9 The Instance Detail page has Import and Export tabs.



Select the Export tab to show the export database dialog. You will need to specify a bucket to store the backup file (see Figure 13.10).

FIGURE 13.10 Exporting a database requires a bucket to store the export file and a file format specification.

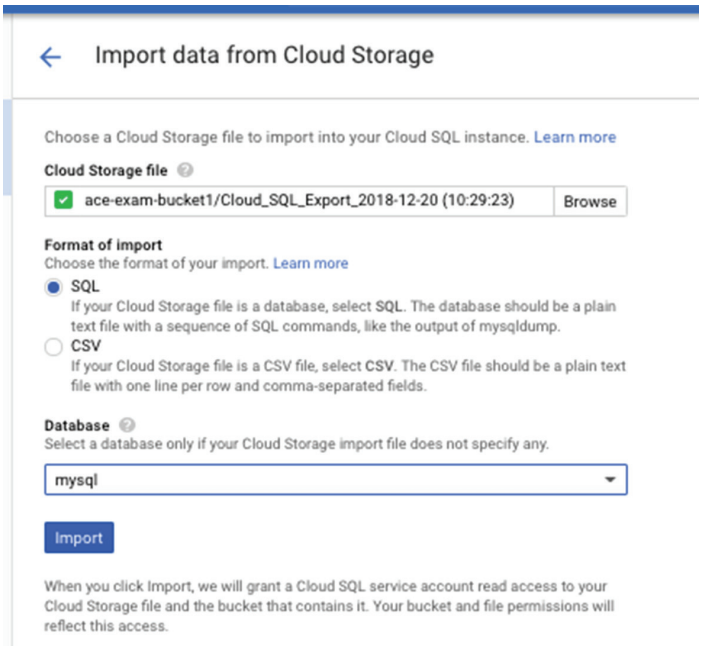


You will also need to choose SQL or CSV output. The SQL output is useful if you plan to import the data to another relational database. CSV is a good choice if you need to move this data into a nonrelational database.

After you create an export file, you can import it.

Follow the same instructions as for exporting, but choose the Import tab instead of the Export tab. This will show a form like that in Figure 13.11. Specify the source file, the file format, and the database to import the data to.

FIGURE 13.11 Importing a database requires a path to the bucket and object storing the export file, a file format specification, and a target database within the instance.



You can also create, import, and export a database using the command line. Use the `gsutil` command to create a bucket, such as this:

```
gsutil mb gs://ace-exam-bucket1/
```

You need to ensure that the service account can write to the bucket, so get the name of the service account by describing the instance with the following command:

```
gcloud sql instances describe [INSTANCE_NAME]
```

In this example, this command would be as follows:

```
gcloud sql instances describe ace-exam-mysql1
```

This will produce a detailed listing about the instance, including the service account email. See Figure 13.12 for an example of the output.

FIGURE 13.12 Details about a database instance generated by the `gcloud sql instances describe` command

```
$
$gcloud sql instances describe ace-exam-mysql1
backendType: SECOND_GEN
connectionName: phrasal-descent-215901:us-west2:ace-exam-mysql1
databaseVersion: MYSQL_5_7
etag: b37b029e41e738e0028eba63f8e37f3035c234006b770ef599c8289fcc69ea6d
gceZone: us-west2-a
instanceType: CLOUD_SQL_INSTANCE
ipAddresses:
- ipAddress: 35.235.110.75
  type: PRIMARY
kind: sql#instance
name: ace-exam-mysql1
project: phrasal-descent-215901
region: us-west2
selfLink: https://www.googleapis.com/sql/v1beta4/projects/phrasal-descent-215901/instances/ace-exam-mysql1
serverCaCert:
  cert: |-
    -----BEGIN CERTIFICATE-----
    MIIDITCCAgmgAwIBAgIBADANBgkqhkiG9w0BAQsFADBIMSMwIQYDVQQDExpHb29n
    bGUgQ2xvdWQgU1FMIFNlcnZlciBDQTEUMBIGA1UEChMLR29vZ2xlLCBJbmMxCzAJ
    BgNVBAYTAlVTMB4XDTE4MTIyMDE4MjQyNVoXDTI4MTIxNzE4MjUyNVowSDEjMCEG
    A1UEAxMaR29vZ2xlIENsb3VkIFNRTCBTZXJ2ZXIgQ0ExFDASBgNVBAoTC0dvb2ds
    ZSwgSW5jMQswCQYDVQQGEwJVU2CCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
    ggEBAKFcYN3KQFgGgrAVEmUkpu8vKLXt6B+QeTNgN1bZj7KX439N8iHP+vIOy4WY
    irsOWUqimiEWqrR/xzqb4L1qtc8GxJMLFqfrEs+L7P/9GExYg0q8SO4To2vUvBNJ
    RQHwrp9Yxo4cfVAwFnkFPGyrGopgNqHqxiGZWYVdObE88P7VV6S9FDkwqf3Wglqx
    EfseqkpU5ucCyE8RGAgwMpd3/tEAQc+ir/eN9vg6J3KL96LeE5TCnn5WB9oTRUp8
    QfBvhk2HZ/fSUcE4Ez+5JCUlkkso0dXGIMX3/km+1Qys0LADwvGpP0pHsgrPl/Wg
    YKR1U3XtW+jkGlMTT4JmLfAlOF8CAwEAAaMWMBQwEgYDVR0TAQH/BAgwBgEB/wIB
    ADANBgkqhkiG9w0BAQsFAAOCAQEAcEvQIPhCAh7XVEdriQxvIiAM+lJyzlUA4Kms
    QoPDzMryvjZVs2tad4f6es4H/Lu7az5mGR6kv3K2wBVjpLiJfUJbC7fRfCHM7rXp
    wWvCeeLiGiFFM8GhycvUA3i/yN26UAo5glxIQv01IHPmHrrQsTMORbeFQ0O0HcTl
    9zBKOvH80lp6X1eYGtv/h2wk0ZyMsUYsEf7hLg50lSOs66/LLSxfhU0TxDhnlGE6
    WD9WJxDZmlRKT/81IO0gd6Y70gSrnxx8ZPw4Mi64QIqRF1s+1l5uVul1avhwN3U7
    3vhVKlqWDWXzJDGRyLU9qN/A9CYN5aDMvEsGxAQRgubOWHSrTA==
    -----END CERTIFICATE-----
  certSerialNumber: '0'
  commonName: C=US,O=Google\, Inc,CN=Google Cloud SQL Server CA
  createTime: '2018-12-20T18:24:25.497000+00:00'
  expirationTime: '2028-12-17T18:25:25.497000+00:00'
  instance: ace-exam-mysql1
  kind: sql#sslCert
  sha1Fingerprint: 7f5880321db93d5c4cfe952ef3566e2c5e12b6a6
serviceAccountEmailAddress: tnkknzut25bezoq72bjbfmo5hu@speckle-umbrella-30.iam.gserviceaccount.com
settings:
  activationPolicy: ALWAYS
  backupConfiguration:
    binaryLogEnabled: true
    enabled: true
    kind: sql#backupConfiguration
    replicationLogArchivingEnabled: false
    startTime: 06:00
  dataDiskSizeGb: '10'
  dataDiskType: PD_SSD
  ipConfiguration:
    ipv4Enabled: true
  kind: sql#settings
  maintenanceWindow:
    day: 0
    hour: 0
    kind: sql#maintenanceWindow
  pricingPlan: PER_USE
  replicationType: SYNCHRONOUS
  settingsVersion: '1'
  storageAutoResize: true
  storageAutoResizeLimit: '0'
  tier: db-n1-standard-1
state: RUNNABLE
```

With that you can use the `gsutil acl ch` command to change the access controls on the bucket to allow the service account to access the bucket. That command is as follows:

```
gsutil acl ch -u [SERVICE_ACCOUNT_ADDRESS]:W gs://[BUCKET_NAME]
```

The `gsutil acl ch` command changes access permissions. The `-u` parameter specifies the user. The `:W` option indicates that the user should have write access to the bucket. In our example, the command would be as follows:

```
gsutil acl ch -u tnkknzut25bezoq72bjbfmo5hu@spe-umbra-30.iam.gserviceaccount.com /
:W gs://ace-exam-bucket1
```

Now that the service account has write access to the bucket, you can create an export of a database using this command:

```
gcloud sql export sql [INSTANCE_NAME] gs://[BUCKET_NAME]/[FILE_NAME] \
                            --database=[DATABASE_NAME]
```

For example, the following command will export the MySQL database to a SQL dump file written to the `ace-exam-bucket1` bucket:

```
gcloud sql export sql ace-exam-mysql1 \
                  gs://ace-exam-buckete1/ace-exam-mysqlexport.sql \
                   --database=mysql
```

If you prefer to export to a CSV file, you would change `sql` to `csv` in the above command. Here's an example:

```
gcloud sql export csv ace-exam-mysql1 gs://ace-exam-buckete1/ace-exam-mysql-export.csv \
                            --database=mysql
```

Importing to a database uses a similarly structured command.

```
gcloud sql import sql [INSTANCE_NAME] gs://[BUCKET_NAME]/[IMPORT_FILE_NAME] \
                            --database=[DATABASE_NAME]
```

Using the example database, bucket, and export file, you can import the file using this command:

```
gcloud sql import sql ace-exam-mysql1 gs://ace-exam-buckete1/ace-exam-mysql-export.sql \
                            --database=mysql
```

Importing and Exporting Data: Cloud Datastore

Importing and exporting data from Datastore is done through the command line. Datastore uses a namespace data structure to group entities that are exported. You will need to specify the name of the namespace used by the entities you are exporting. The default namespace is simply `(default)`.

The Cloud Datastore `export` command is as follows:

```
gcloud datastore export --namespaces="(default)" gs://${BUCKET}
```

You can export to a bucket called `ace-exam-datastore1` using this command:

```
gcloud datastore export --namespaces="(default)" gs://ace-exam-datastore1
```

The Cloud Datastore `import` command is as follows:

```
gcloud datastore import gs://${BUCKET}/[PATH]/[FILE].overall_export_metadata
```

The export process will create a folder named `ace-exam-datastore1` using the data and time of the export. The folder will contain a metadata file and a folder containing the exported data. The metadata filename will use the same date and time used for the containing folder. The data folder will be named after the namespace of the exported Datastore database. An example import command is as follows:

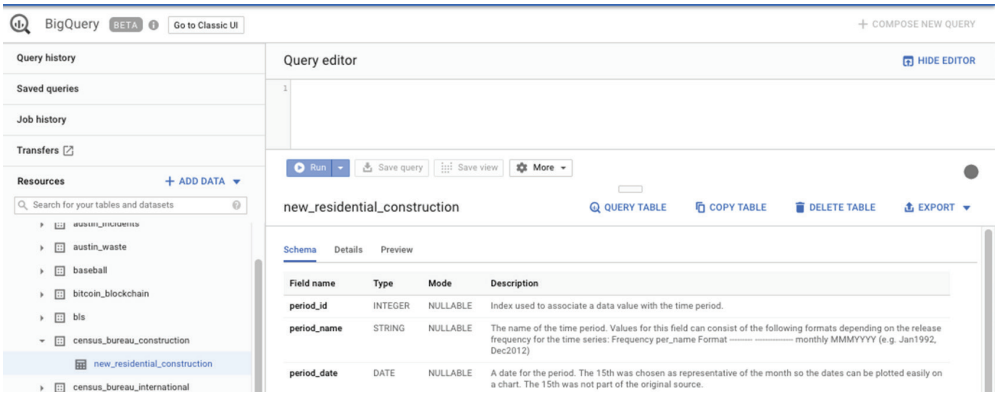
```
gcloud datastore import gs://ace-exam-datastore1/2018-12-20T19:13:55_64324/
2018-12-20T19:13:55_64324.overall_export_metadata
```

Importing and Exporting Data: BigQuery

BigQuery users can export and import tables using Cloud Console and the command line.

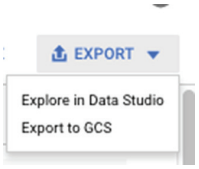
To export a table using the console, navigate to the BigQuery console interface. Under Resources, open the data set containing the table you want to export. Click the table name to list the table description, as shown in Figure 13.13. Notice the Export option in the upper right.

FIGURE 13.13 Detailed list of a BigQuery table



At the far right, click Export to display a list of two export locations: Google Cloud Storage or Data Studio, which is an analysis tool in GCP (see Figure 13.14).

FIGURE 13.14 Choosing a target location for a BigQuery export



Selecting Cloud Storage displays a form like the one shown in Figure 13.15. Enter the bucket name to store the export file. Choose a file format. The options are CSV, Avro, and JSON. Choose a compression type. The options are None or Gzip for CSV and "deflate" and "snappy" for Avro.

File Formats

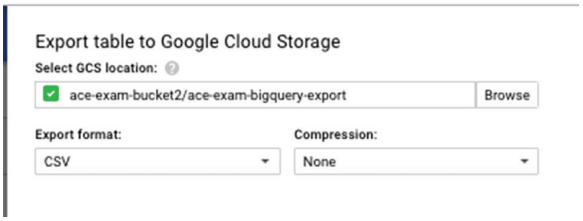
BigQuery offers several export file options. CSV, short for "comma-separated values," is a human-readable format suitable for small data sets that will be imported into tools that only support the CSV format. CSV is not optimized for storage, so it does not compress or use a more efficient encoding than text. It's not the best option when exporting large data sets.

JSON is also a human-readable format that has similar advantages and disadvantages to CSV. One difference is that JSON includes schema information with each record, while CSV uses an optional header row with column names at the beginning of the file to describe the schema.

Gzip is a widely used lossless compression utility.

Avro is a compact binary format that supports complex data structures. When data is saved in the Avro format, a schema is written to the file along with data. Schemas are defined in JSON. Avro is a good option for large data sets, especially when importing data into other applications that read the Avro format, including Apache Spark, which is available as a managed service in Cloud Dataproc. Avro files can be compressed using either the deflate or snappy utilities. Deflate produces smaller compressed files, but snappy is faster.

FIGURE 13.15 Specifying the output parameters for a BigQuery export operation



To export data from the command line, use the bq `extract` command. The structure is as follows:

```
bq extract --destination_format [FORMAT] --compression [COMPRESSION_TYPE]
--field_delimiter [DELIMITER] --print_header [BOOLEAN] [PROJECT_ID]:[DATASET].
[TABLE] gs://[BUCKET]/[FILENAME]
```

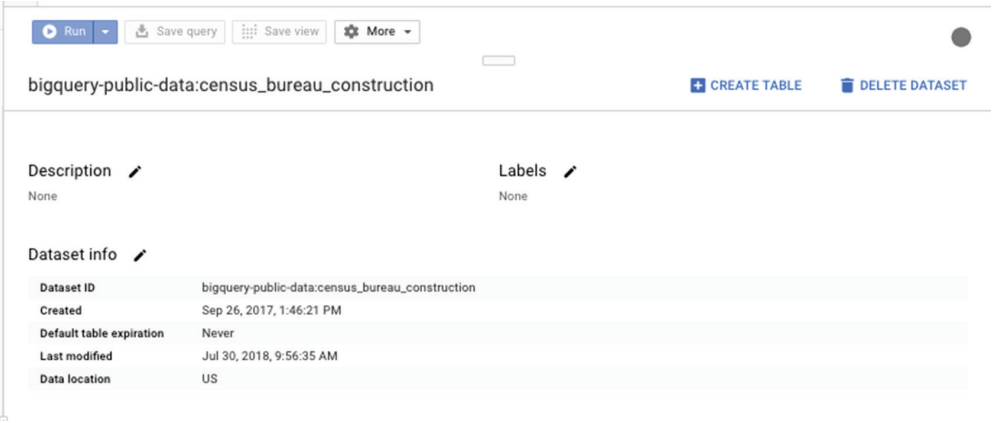
Here's an example:

```
bq extract --destination_format CSV --compression GZIP 'mydataset.mytable'
gs://example-bucket/myfile.zip
```

Remember, the command-line tool for working with BigQuery is `bq`, not `gcloud`.

To import data into BigQuery, navigate to the BigQuery console page and select a data set you'd like to import data into. Click a data set and then select the Create Table tab, as shown in Figure 13.16.

FIGURE 13.16 When viewing a data set, you have the option to create a table.



The Create Table form takes several parameters, including an optional source table, a destination project, the dataset name, the table type, and the table name (see Figure 13.17).