Review Questions

You can find the answers in the Appendix.

1. You have just opened the GCP console at `console.google.com`. You have authenticated with the user you want to use. What is one of the first things you should do before performing tasks on VMs?
   - **A.** Open Cloud Shell.
   - **B.** Verify you can SSH into a VM.
   - **C.** Verify that the selected project is the one you want to work with.
   - **D.** Review the list of running VMs.

2. What is a one-time task you will need to complete before using the console?
   - **A.** Set up billing
   - **B.** Create a project
   - **C.** Create a storage bucket
   - **D.** Specify a default zone

3. A colleague has asked for your assistance setting up a test environment in Google Cloud. They have never worked in GCP. You suggest starting with a single VM. Which of the following is the minimal set of information you will need?
   - **A.** A name for the VM and a machine type
   - **B.** A name for the VM, a machine type, a region, and a zone
   - **C.** A name for the VM, a machine type, a region, a zone, and a CIDR block
   - **D.** A name for the VM, a machine type, a region, a zone, and an IP address

4. An architect has suggested a particular machine type for your workload. You are in the console creating a VM and you don't see the machine type in the list of available machine types. What could be the reason for this?
   - **A.** You have selected the incorrect subnet.
   - **B.** That machine type is not available in the zone you specified.
   - **C.** You have chosen an incompatible operating system.
   - **D.** You have not specified a correct memory configuration.

5. Your manager asks for your help with understanding cloud computing costs. Your team runs dozens of VMs for three different applications. Two of the applications are for use by the marketing department and one is use by the finance department. Your manager wants a way to bill each department for the cost of the VMs used for their applications. What would you suggest to help solve this problem?
   - **A.** Access controls
   - **B.** Persistent disks
   - **C.** Labels and descriptions
   - **D.** Descriptions only

6. If you wanted to set the preemptible property using Cloud Console, in which section of the Create An Instance page would you find the option?
   - **A.** Availability Policy
   - **B.** Identity And API Access
   - **C.** Sole Tenancy
   - **D.** Networking

7. You need to set up a server with a high level of security. You want to be prepared in case of attacks on your server by someone trying to inject a rootkit (a kind of malware that can alter the operating system). Which option should you select when creating a VM?
   - **A.** Firewall
   - **B.** Shield VM
   - **C.** Project-wide SSH keys
   - **D.** Boot disk integrity check

8. All of the following parameters can be set when adding an additional disk through Google Cloud Console, except one. Which one?
   - **A.** Disk type
   - **B.** Encryption key management
   - **C.** Block size
   - **D.** Source image for the disk

9. You lead a team of cloud engineers who maintain cloud resources for several departments in your company. You've noticed a problem with configuration drift. Some machine configurations are no longer in the same state as they were when created. You can't find notes or documentation on how the changes were made or why. What practice would you implement to solve this problem?
   - **A.** Have all cloud engineers use only command-line interface in Cloud Shell.
   - **B.** Write scripts using gcloud commands to change configuration and store those scripts in a version control system.
   - **C.** Take notes when making changes to configuration and store them in Google Drive.
   - **D.** Limit privileges so only you can make changes so you will always know when and why configurations were changed.

10. When using the Cloud SDK command-line interface, which of the following is part of commands for administering resources in Compute Engine?
    - **A.** `gcloud compute instances`
    - **B.** `gcloud instances`
    - **C.** `gcloud instances compute`
    - **D.** None of the above

11. A newly hired cloud engineer is trying to understand what VMs are running in a particular project. How could the engineer get summary information on each VM running in a project?
    A. Execute the command `gcloud compute list`
    B. Execute the command `gcloud compute instances list`
    C. Execute the command `gcloud instances list`
    D. Execute the command `gcloud list instances`

12. When creating a VM using the command line, how should you specify labels for the VM?
    A. Use the `--labels` option with labels in the format of KEYS:VALUES.
    B. Use the `--labels` option with labels in the format of KEYS=VALUE.
    C. Use the `--labels` option with labels in the format of KEYS,VALUES.
    D. This is not possible in the command line.

13. In the boot disk advanced configuration, which operations can you specify when creating a new VM?
    A. Add a new disk, reformat an existing disk, attach an existing disk
    B. Add a new disk and reformat an existing disk
    C. Add a new disk and attach an existing disk
    D. Reformat an existing disk and attach an existing disk

14. You have acquired a 10 GB data set from a third-party research firm. A group of data scientists would like to access this data from their statistics programs written in R. R works well with Linux and Windows file systems, and the data scientists are familiar with file operations in R. The data scientists would each like to have their own dedicated VM with the data available in the VM's file system. What is a way to make this data readily available on a VM and minimize the steps the data scientists will have to take?
    A. Store the data in Cloud Storage.
    B. Create VMs using a source image created from a disk with the data on it.
    C. Store the data in Google Drive.
    D. Load the data into BigQuery.

15. The Network tab of the create VM form is where you would perform which of the following operations?
    A. Set the IP address of the VM
    B. Add a network interface to the VM
    C. Specify a default router
    D. Change firewall configuration rules

16. You want to create a VM using the `gcloud` command. What parameter would you include to specify the type of boot disk?
    - **A.** `boot-disk-type`
    - **B.** `boot-disk`
    - **C.** `disk-type`
    - **D.** `type-boot-disk`

17. Which of the following commands will create a VM with four CPUs that is named web-server-1?
    - **A.** `gcloud compute instances create --machine-type=n1-standard-4 web-server-1`
    - **B.** `gcloud compute instances create --cpus=4 web-server-1`
    - **C.** `gcloud compute instances create --machine-type=n1-standard-4 –instance-name web-server-1`
    - **D.** `gcloud compute instances create --machine-type=n1-4-cpu web-server-1`

18. Which of the following commands will stop a VM named web-server-1?
    - **A.** `gcloud compute instances halt web-server-1`
    - **B.** `gcloud compute instances --terminate web-server1`
    - **C.** `gcloud compute instances stop web-server-1`
    - **D.** `gcloud compute stop web-server-1`

19. You have just created an Ubuntu VM and want to log into the VM to install some software packages. Which network service would you use to access the VM?
    - **A.** FTP
    - **B.** SSH
    - **C.** RDP
    - **D.** `ipconfig`

20. Your management team is considering three different cloud providers. You have been asked to summarize billing and cost information to help the management team compare cost structures between clouds. Which of the following would you mention about the cost of VMs in GCP?
    - **A.** VMs are billed in 1-second increments, cost varies with the number of CPUs and amount of memory in a machine type, you can create custom machine types, preemptible VMs cost up to 80 percent less than standard VMs, and Google offers discounts for sustained usage.
    - **B.** VMs are billed in 1-second increments and VMs can run up to 24 hours before they will be be shut down.
    - **C.** Google offers discounts for sustained usage in only some regions, cost varies with the number of CPUs and amount of memory in a machine type, you can create custom machine types, preemptible VMs cost up to 80 percent less than standard VMs.
    - **D.** VMs are charged for a minimum of 1 hour of use and cost varies with the number of CPUs and amount of memory in a machine type.

Chapter 6

Managing Virtual Machines

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

✓ **4.1 Managing Compute Engine resources**

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

After creating virtual machines, a cloud engineer will need to work with both single instances of virtual machines (VMs) and groups of VMs that run the same configuration. The latter are called *instance groups* and are introduced in this chapter.

This chapter begins with a description of common management tasks and how to complete them in the console, followed by a description of how to complete them in Cloud Shell or with the Cloud SDK command line. Next, you will learn how to configure and manage instance groups. The chapter concludes with a discussion of guidelines for managing VMs.

Managing Single Virtual Machine Instances

We begin by discussing how to manage a single instance of a VM. By single instance, we mean one created by itself and not in an instance group or other type of cluster. Recall from previous chapters that there are three ways to work with instances: in Cloud Console, in Cloud Shell, and with the Cloud SDK command line. Both Cloud Shell and the Cloud SDK command lines make use of `gcloud` commands, so we will describe Cloud Shell and Cloud SDK together in this section.

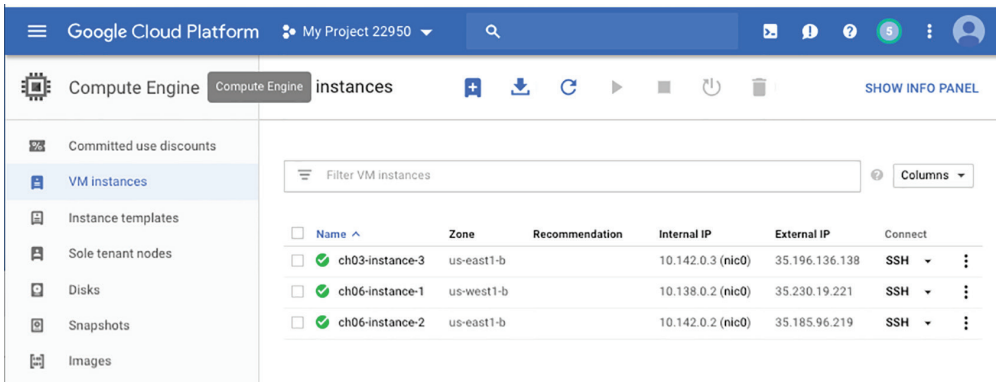
Managing Single Virtual Machine Instances in the Console

The basic VM management tasks that a cloud engineer should be familiar with are creating, stopping, and deleting instances. We covered creating instances in the previous chapter, so we'll focus on the other tasks here. You should also be familiar with listing VMs, attaching graphics processing units (GPUs) to VMs, and working with snapshots and images.

Starting, Stopping, and Deleting Instances

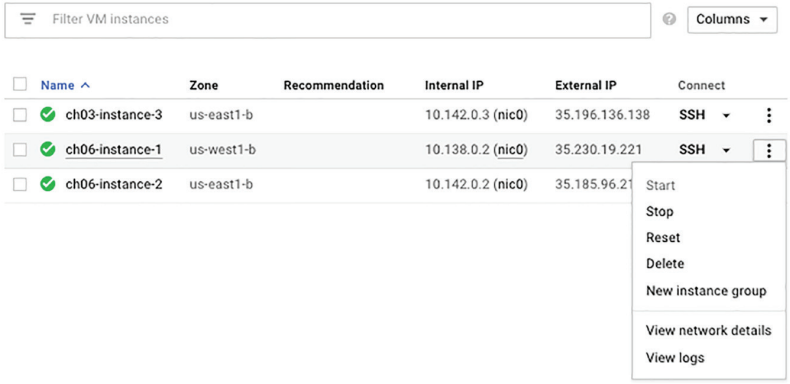
To start working, open the console and select Compute Engine. Then select VM instances. This will display a window such as in Figure 6.1, but with different VMs listed. In this example, there are three VMs.

FIGURE 6.1 The VM Instance panel in the Compute Engine section of Cloud Console



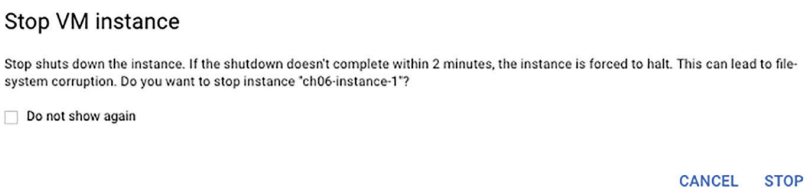
The three instances in Figure 6.1 are all running. You can stop the instances by clicking the three-dot icon on the right side of the line listing the VM attributes. This action displays a list of commands. Figure 6.2 shows the list of commands available for `instancech06-instance-1`.

FIGURE 6.2 The list of commands available from the console for changing the state of a VM



If you select Stop from the command menu, the instance will be stopped. When an instance is stopped, it is not consuming compute resources, so you will not be charged. The instance still exists and can be started again when you need it. Figure 6.3 shows a warning form that indicates you are about to stop a VM. You can click the dialog box in the lower left to suppress this message.

FIGURE 6.3 A warning message that may appear about stopping a VM

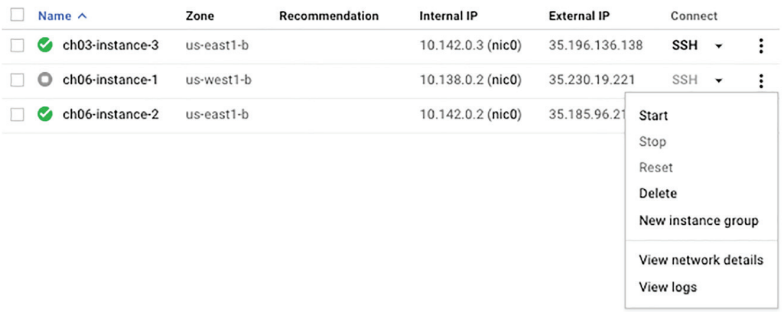


When you stop a VM, the green check mark on the left changes to a gray circle with a white square, and the SSH option is disabled, as shown in Figure 6.4.

FIGURE 6.4 When VMs are stopped the icon on the left changes, and SSH is no longer available.

To start a stopped VM, click the three-dot icon on the right to display the menu of available commands. Notice in Figure 6.5 that Start is now available, but Stop and Reset are not.

FIGURE 6.5 When VMs are stopped, Stop and Reset are no longer available, but Start is available as a command.

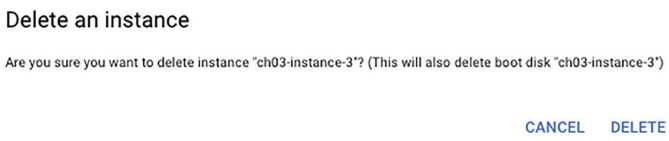


The Reset command restarts a VM. The properties of the VM will not change, but data in memory will be lost.

When a VM is restarted, the contents of memory are lost. If you need to preserve data between reboots or for use on other VMs, save the data to a persistent disk or Cloud Storage.

When you are done with an instance and no longer need it, you can delete it. Deleting a VM removes it from Cloud Console and releases resources, like the storage used to keep the VM image when stopped. Deleting an instance from Cloud Console will display a message such as in Figure 6.6.

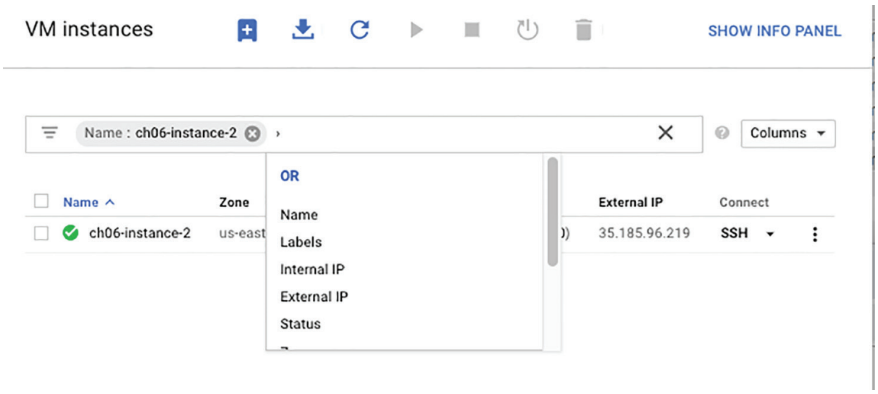
FIGURE 6.6 Deleting an instance from the console will display a warning message such as this.



Viewing Virtual Machine Inventory

The VM Instances page of Cloud Console will show a list of VMs, if any exist in the current project. If you have a large number of instances, it can help to filter the list to see only instances of interest. Do this using the Filter VM Instances box above the list of VMs, as shown in Figure 6.7.

FIGURE 6.7 List of instances filtered by search criteria



In this example, we have specified that we want to see only the instance named `ch06-instance-2`. In addition to specifying instance names, you can also filter by the following:

- Labels
- Internal IP
- External IP
- Status
- Zone
- Network

- Deletion protection
- Member of managed instance group
- Member of unmanaged instance group

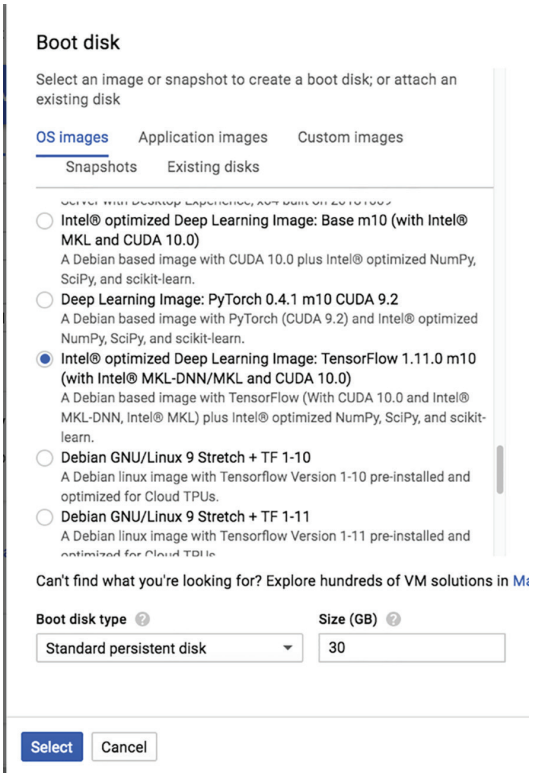
If you set multiple filter conditions, then all must be true for a VM to be listed unless you explicitly state the `OR` operator.

Attaching GPUs to an Instance

GPUs are used for math-intensive applications such as visualizations and machine learning. GPUs perform math calculations and allow some work to be off-loaded from the CPU to the GPU.

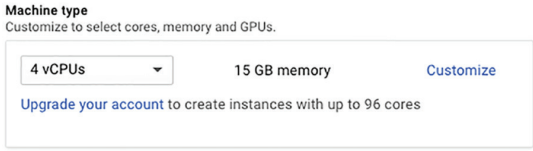
To add a GPU to an instance, you must start an instance in which GPU libraries have been installed or will be installed. For example, you can use one of the Google Cloud Platform (GCP) images that has GPU libraries installed, including the Deep Learning images, as shown in Figure 6.8. You must also verify that the instance will run in a zone that has GPUs available.

FIGURE 6.8 When attaching GPUs, it is best to use an image that has the necessary libraries installed. You can use a GCP-provided image or a custom image with the necessary libraries.



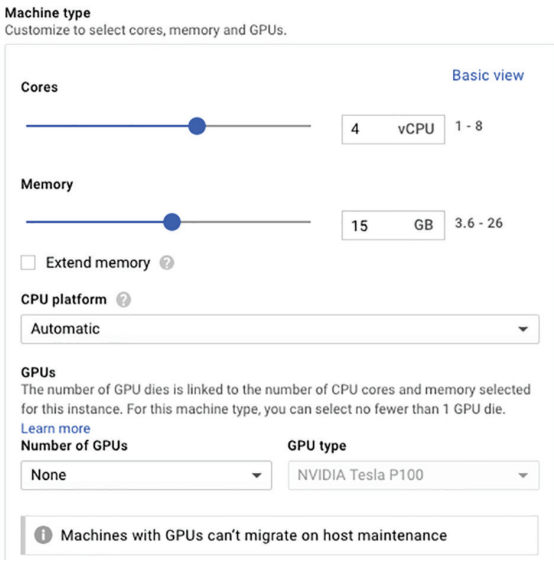
You will also need to customize the configuration for the machine type; Figure 6.9 shows the form.

FIGURE 6.9 The Cloud Console form for configuring machine type



Click Customize. This will expand the set of machine type parameters, as shown in Figure 6.10.

FIGURE 6.10 This form is used when creating a customized machine type.



Select the number of GPUs to attach. The options are None, 1, 2, or 4 (see Figure 6.11). Then select the GPU type, as shown in Figure 6.12.

FIGURE 6.11 Selecting the number of GPUs to attach to the VM

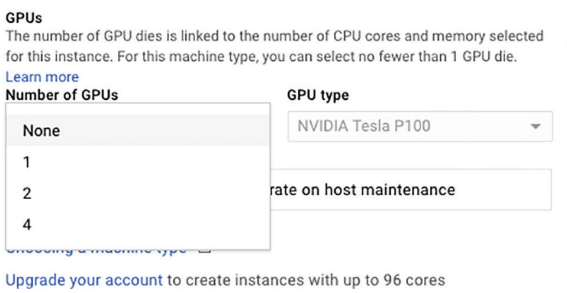
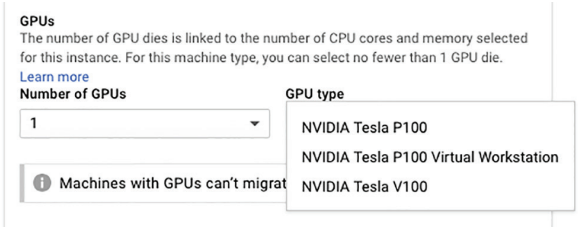


FIGURE 6.12 Selecting the type of GPUs to attach to the VM



There are some restrictions on the use of GPUs. The CPU must be compatible with the GPU selected. For example, if you are running a VM on a server with an Intel Skylake or later CPU, then you cannot use the Tesla K80 GPU. GPUs cannot be attached to shared memory machines. For the latest documentation on GPU restrictions and a list of zones with GPUs, see `https://cloud.google.com/compute/docs/gpus/`.

Also, if you add a GPU to a VM, you must set the instance to terminate during maintenance. This is set in the Availability Policies section of the VM configuration form (see Figure 6.13).

FIGURE 6.13 Recommended availability policies for VMs with attached GPUs

Working with Snapshots

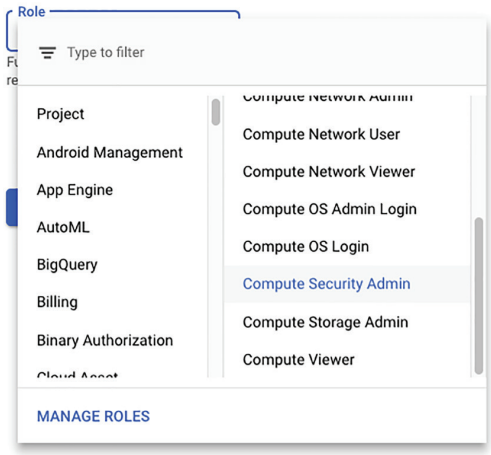
Snapshots are copies of data on a persistent disk. You use snapshots to save data on a disk so you can restore it. This is a convenient way to make multiple persistent disks with the same data.

When you first create a snapshot, GCP will make a full copy of the data on the persistent disk. The next time you create a snapshot from that disk, GCP will copy only the data that has changed since the last snapshot. This optimizes storage while keeping the snapshot up to date with the data that was on the disk the last time a snapshot operation occurred.

If you are running a database or other application that may buffer data in memory before writing to disk, be sure to flush disk buffers before you create the snapshot; otherwise, data in memory that should be written to disk may be lost. The way to flush the disk buffers will vary by application. For example, MySQL has a `FLUSH` statement.

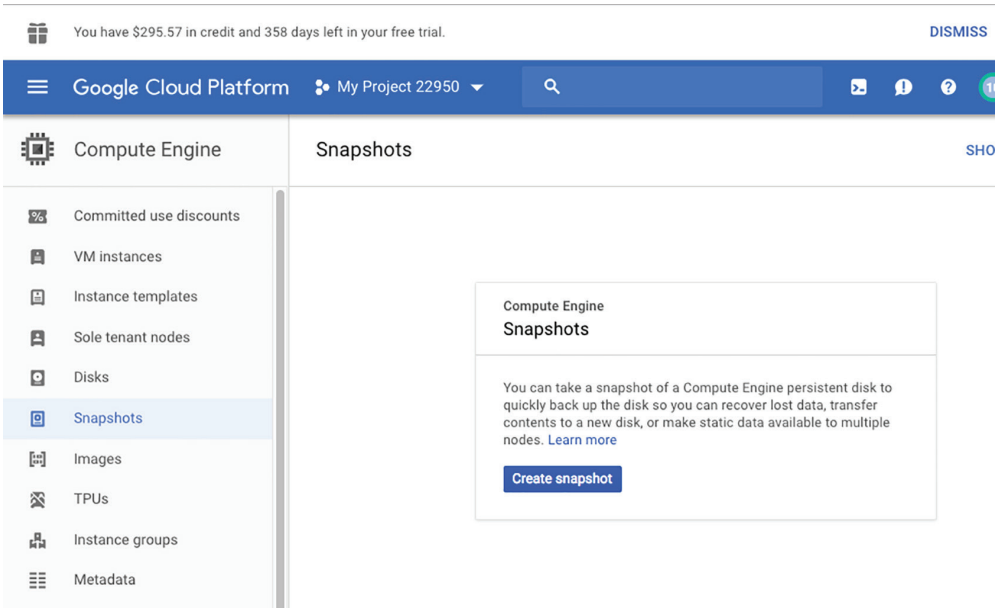
To work with snapshots, a user must be assigned the Compute Storage Admin role. Go to the Identity Access Management (IAM) page, select Roles, and then specify the email address of a user to be assigned the role. Select the role from the list of roles, as shown in Figure 6.14.

FIGURE 6.14 To work with snapshots, a user needs to have the Cloud Storage Admin role.



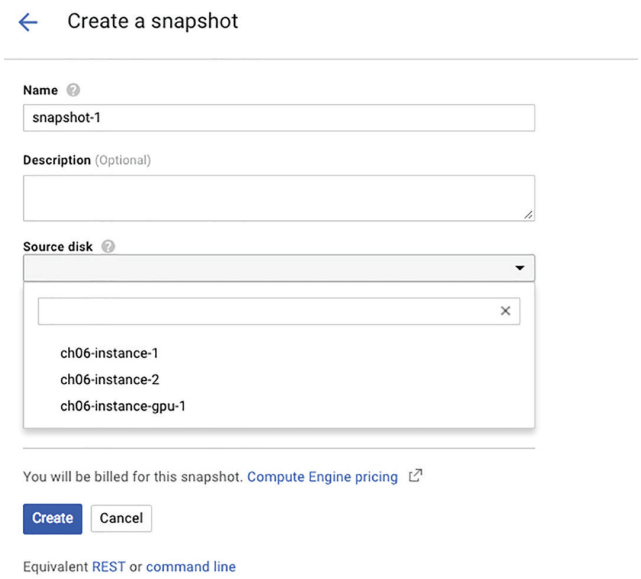
To create a snapshot from Cloud Console, display the Compute Engine options and select Snapshots from the left panel, as shown in Figure 6.15.

FIGURE 6.15 Creating a snapshot using Cloud Console



Then, click Create Snapshot to display the form in Figure 6.16. Specify and name and, optionally, a description. You can add labels to the snapshot as well. It is a good practice to label all resources with a consistent labeling convention. In the case of snapshots, the labels may indicate the type of data on the disk and the application that uses the data.

FIGURE 6.16 Form for creating a snapshot



If you are making a snapshot of a disk on a Windows server, check the Enable VSS box to create an application-consistent snapshot without having to shut down the instance.

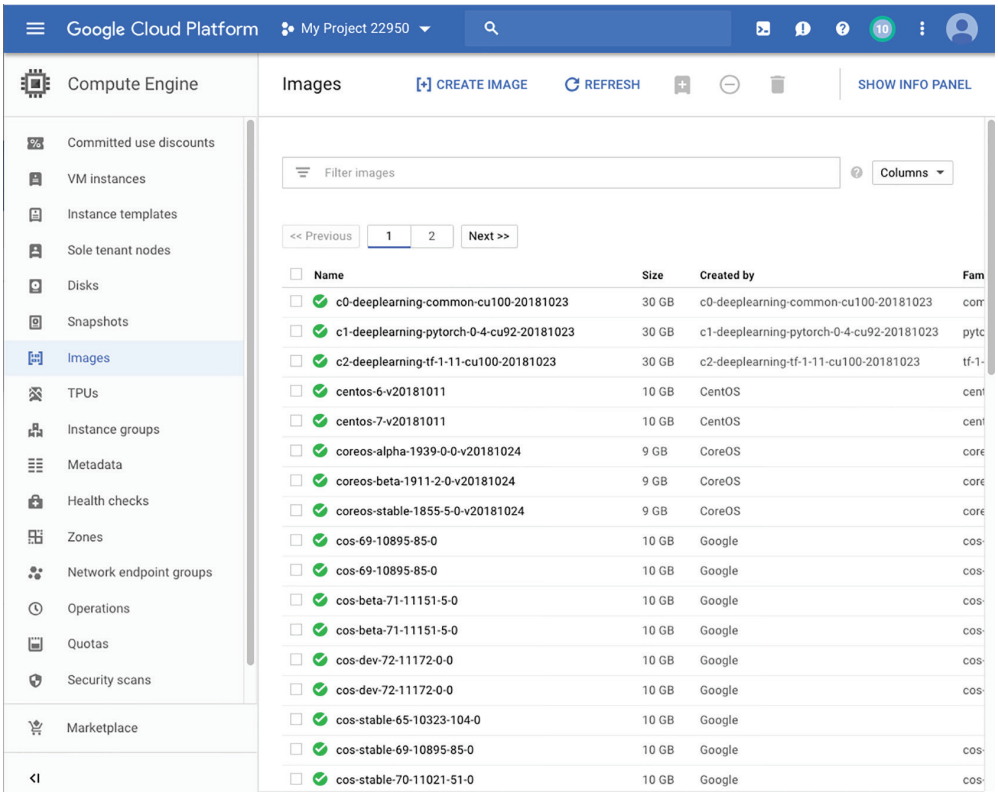
Working with Images

Images are similar to snapshots in that they are copies of disk contents. The difference is that snapshots are used to make data available on a disk, while images are used to create VMs. Images can be created from the following:

- Disk
- Snapshot
- Cloud storage file
- Another image

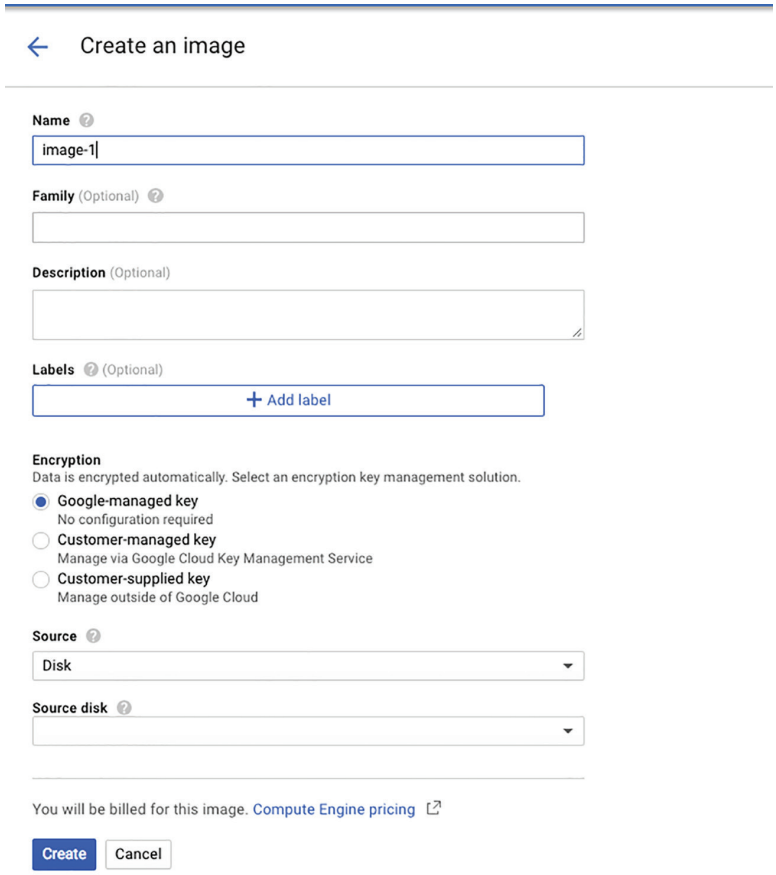
To create an image, choose the Image option from the Compute Engine page in Cloud Console, as shown in Figure 6.17. This lists available images.

FIGURE 6.17 Images available. From here, you can create additional images.



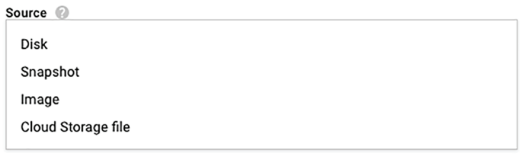
Select Create Image to show the form in Figure 6.18. This allows you to create a new image by specifying name, description, and labels. Images have an optional attribute called Family, which allows you to group images. When a family is specified, the latest, nondeprecated image in the family is used.

FIGURE 6.18 Cloud Console form for creating an image



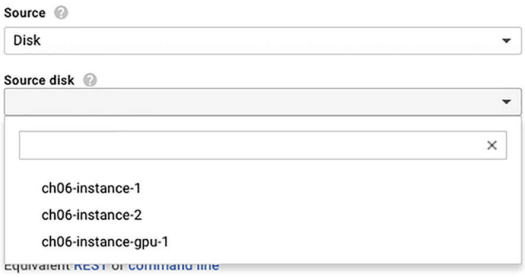
The form provides a drop-down list of options for the source of the image, as shown in Figure 6.19.

FIGURE 6.19 Options for the source of an image



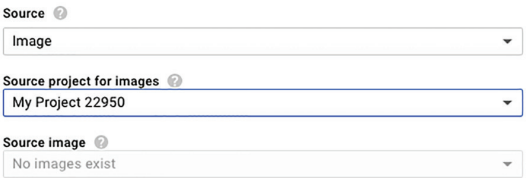
When Disk is selected as the source, you can choose from disks on VMs, as shown in Figure 6.20.

FIGURE 6.20 Options when using a disk as the source of an image



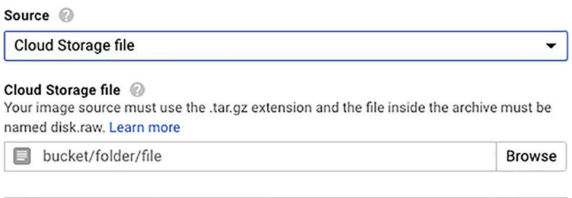
When you choose Image as the source type, you can choose an image from the current project or other projects (see Figure 6.21).

FIGURE 6.21 When using an image as a source, you can choose a source image from another project.



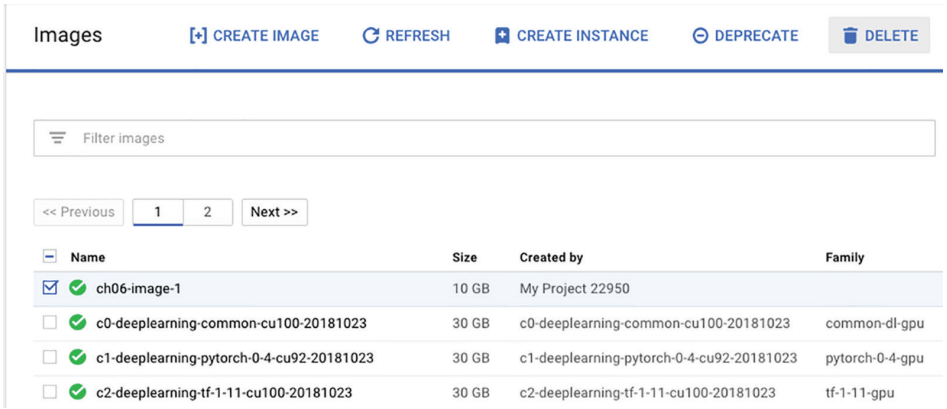
If you choose a Cloud Storage file as a source, you can browse your Cloud Storage bucket to find a file to use as the source (see Figure 6.22).

FIGURE 6.22 When using a Cloud Storage file as a source, you browse your storage buckets for a file.



After you have created an image, you can delete it or deprecate it by checking the box next to the image name and selecting Delete or Deprecate from the line of commands above the list, as shown in Figure 6.23. You can delete and deprecate only custom images, not GCP-supplied images.

FIGURE 6.23 The Delete and Deprecate commands are available when one of your custom images is selected.



Delete removes the image, while Deprecated marks the image as no longer supported and allows you to specify a replacement image to use going forward. Google's deprecated images are available for use but may not be patched for security flaws or other updates. Deprecation is a useful way of informing users of the image that it is no longer supported and that they should plan to test their applications with the newer, supported versions of the image. Eventually, deprecated images will no longer be available, and users of the deprecated images will need to use different versions.

After you have created an image, you can create an instance using that image by selecting the Create Instance option in the line of commands above the image listing.

In addition to managing VMs through the console, you can manage compute resources using the command line.

Managing a Single Virtual Machine Instance with Cloud Shell and the Command Line

In addition to managing VMs through the console, you can manage compute resources using the command line. The same commands can be used in Cloud Shell or in your local environment after you have installed Google Cloud SDK, which was covered in Chapter 5.

This section describes the most important commands for working with instances. Commands have their own specific sets of parameters; however, all `gcloud` commands support sets of flags. These are referred to as `gcloud`-wide flags, also known as `gcloud` global flags, and include the following:

- `--account` specifies a GCP account to use overriding the default account.
- `--configuration` uses a named configuration file that contains key-value pairs.
- `--flatten` generates separate key-value records when a key has multiple values.

- `--format` specifies an output format, such as a default (human readable) CSV, JSON, YAML, text, or other possible options.
- `--help` displays a detailed help message.
- `--project` specifies a GCP project to use, overriding the default project.
- `--quiet` disables interactive prompts and uses defaults.
- `--verbosity` specifies the level of detailed output messages. Options are `debug`, `info`, `warning`, and `error`.

Throughout this section, commands can take an optional `--zone` parameter. We assume a default zone was set when you ran `gcloud init`.

Starting Instances

To start an instance, use the `gcloud` command, specifying that you are working with a compute service and instances specifically. You also need to indicate that you will be starting an instance by specifying `start`, followed by the name of one or more instances.

The command syntax is as follows:

```
gcloud compute instances start  INSTANCE_NAMES
```

An example is as follows:

```
gcloud compute instances start  ch06-instance-1 ch06-instance-2
```

The `instance start` command also takes optional parameters. The `--async` parameter displays information about the `start` operation. The `--verbose` option in many Linux commands provides similar functionality. An example is as follows:

```
gcloud compute instances start  ch06-instance-1 ch06-instance-2 --async
```

GCP needs to know in which zone to create an instance. This can be specified with the `--zone` parameter as follows:

```
gcloud compute instances start  ch06-instance-1 ch06-instance-2 --zone us-central1-c
```

You can get a list of zones with the following command:

```
gcloud compute zones list
```

If no zone is specified, the command will prompt for one.

Stopping Instances

To stop an instance, use `gcloud compute instances` and specify `stop` followed by the name of one or more instances.

The command syntax is as follows:

```
gcloud compute instances stop  INSTANCE_NAMES
```

An example is as follows:

```
gcloud compute instances stop  ch06-instance-3 ch06-instance-4
```

Like the `instance start` command, the `stop` command takes optional parameters. The `--async` parameter causes information about the start operation to be displayed:

```
gcloud compute instances stop  ch06-instance-1 ch06-instance-2 -async
```

GCP needs to know which zone contains the instance to stop. This can be specified with the `--zone` parameter as follows:

```
gcloud compute instances stop ch06-instance-1 ch06-instance-2 --zone us-central1-c
```

You can get a list of zones with the following command:

```
gcloud compute zones list
```

Deleting Instances

When you are finished working with a VM, you can delete it with the `delete` command. Here's an example:

```
gcloud compute instances delete ch06-instance-1
```

The `delete` command takes the `--zone` parameter to specify where the VM to delete is located. Here's an example:

```
gcloud compute instances delete ch06-instance-1 --zone us-central2-b
```

When an instance is deleted, the disks on the VM may be deleted or saved by using the `--delete-disks` and `--keep-disks` parameters, respectively. You can specify `all` to keep all disks, `boot` to specify the partition of the root file system, and `data` to specify nonboot disks.

For example, the following command keeps all disks:

```
gcloud compute instances delete ch06-instance-1 --zone us-central2-b --keep-
disks=all
```

while the following deletes all nonboot disks:

```
gcloud compute instances delete ch06-instance-1 --zone us-central2-b --delete-
disks=data
```

Viewing VM Inventory

The command to view the set of VMs in your inventory is as follows:

```
gcloud compute instances list
```

This command takes an optional name of an instance. To list VMs in a particular zone, you can use the following:

```
gcloud compute instances list --filter="zone:ZONE"
```

where *`ZONE`* is the name of a zone. You can list multiple zones using a comma-separated list.

The `--limit` parameter is used to limit the number of VMs listed, and the `--sort-by` parameter is used to reorder the list of VMs by specifying a resource field. You can see the resource fields for a VM by running the following:

```
gcloud compute instances describe
```

Working with Snapshots

You can create a snapshot of a disk using the following command:

```
gcloud compute disks snapshot DISK_NAME --snapshot-names=NAME
```

where *`DISK_NAME`* is the name of a disk and *`NAME`* is the name of the snapshot. To view a list of snapshots, use the following:

```
gcloud compute snapshots list
```

For detailed information about a snapshot, use the following:

```
gcloud compute snapshots describe SNAPSHOT_NAME
```

where `SNAPSHOT_NAME` is the name of the snapshot to describe. To create a disk, use this:

```
gcloud compute disks create DISK_NAME --source-snapshot=SNAPSHOT_NAME
```

You can also specify the size of the disk and disk type using the `--size` and `--parameters`. Here's an example:

```
gcloud compute disks create ch06-disk-1 --source-snapshot=ch06-snapshot
--size=100 --type=pd-standard
```

This will create a 100GB disk using the `ch06-snapshot` using a standard persistent disk.

Working with Images

GCP provides a wide range of images to use when creating a VM; however, you may need to create a specialized image of your own. This can be done with the following command:

```
gcloud compute images create IMAGE_NAME
```

where *`IMAGE_NAME`* is the name given to the images. The source for the images is specified using one of the source parameters, which are as follows:

- `--source-disk`
- `--source-image`
- `--source-image-family`
- `--source-snapshot`
- `--source-uri`

The `source-disk`, `source-image`, and `source-snapshot` parameters are used to create an image using a disk, image, and snapshot, respectively. The `source-image-family` parameter uses the latest version of an image in the family. Families are groups of related

images, which are usually different versions of the same underlying image. The `source-uri` parameter allows you to specify an image using a web address.

An image can have a description and a set of labels. These are assigned using the `--description` and `--labels` parameters.

Here is an example of creating a new image from a disk:

```
gcloud compute images create ch06-image-1 --source-disk ch06-disk-1
```

You can also delete images when they are no longer needed using this:

```
gcloud compute images delete IMAGE_NAME
```

It is often helpful to store images on Cloud Storage. You can export an image to Cloud Storage with the following command:

```
gcloud compute images export --destination-uri DESTINATION_URI --image IMAGE_NAME
```

where *`DESTINATION_URI`* is the address of a Cloud Storage bucket to store the image.

Introduction to Instance Groups

Instance groups are sets of VMs that are managed as a single entity. Any `gcloud` or console command applied to an instance group is applied to all members of the instance group. Google provides two types of instance groups: managed and unmanaged.

Managed groups consist of groups of identical VMs. They are created using an instance template, which is a specification of a VM configuration, including machine type, boot disk image, zone, labels, and other properties of an instance. Managed instance groups can automatically scale the number of instances in a group and be used with load balancing to distribute workloads across the instance group. If an instance in a group crashes, it will be recreated automatically. Managed groups are the preferred type of instance group.

Unmanaged groups should be used only when you need to work with different configurations within different VMs within the group.

Creating and Removing Instance Groups and Templates

To create an instance group, you must first create an instance group template. To create an instance template, use the following command:

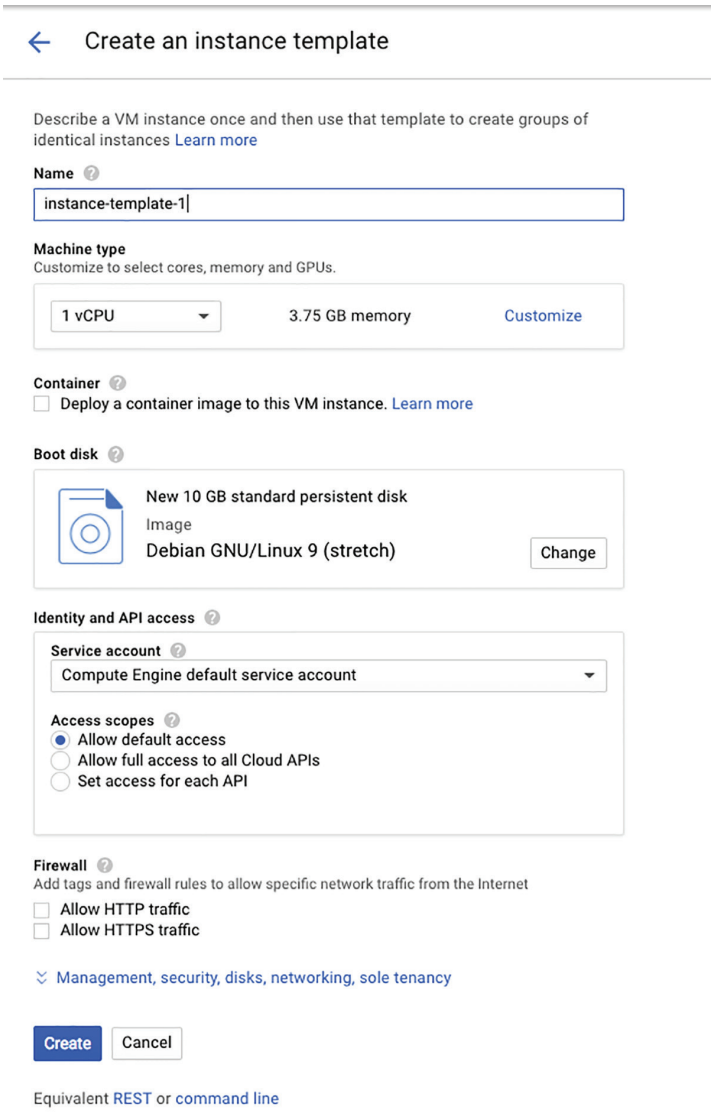
```
gcloud compute instance-templates create INSTANCE
```

You can specify an existing VM as the source of the instance template by using the `--source-instance` parameter (GCP will use a n1-standard1 image by default). Here's an example:

```
gcloud compute instance-templates create ch06-instance-template-1 --source-
instance=ch06-instance-1
```

Instance group templates can also be created in the console using the Instance Groups Template page, as shown in Figure 6.24.

FIGURE 6.24 Instance group templates can be created in the console using a form similar to the create instance form.

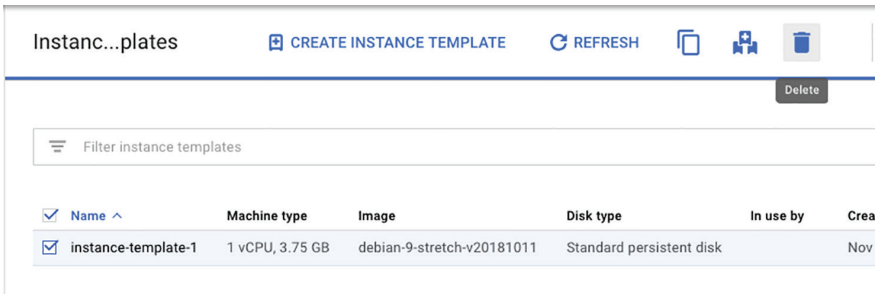


Instance groups can contain instances in a single zone or across a region. The first is called a *zonal* managed instance group, and the second is called a *regional* managed

instance group. Regional managed instance groups are recommended because that configuration spreads the workload across zones, increasing resiliency.

You can remove instance templates by deleting them from the Instance Group Template page in the console. Select the instance group template by checking the box in the list of templates and then delete it by clicking the delete icon, as shown in Figure 6.25.

FIGURE 6.25 Instance group templates can be deleted in the console.



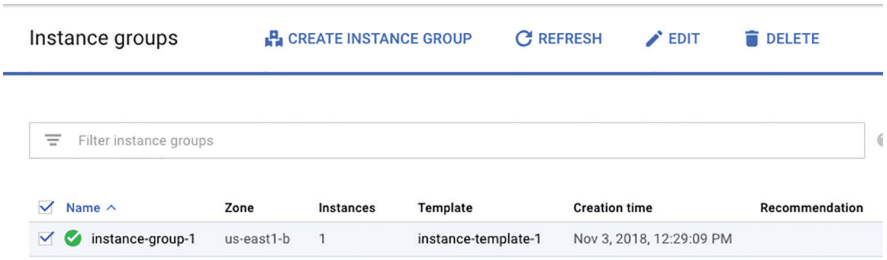
You can also delete an instance group template using the following command:

```
gcloud compute instance-templates delete NAME
```

where *INSTANCE-TEMPLATE-NAME* is the name of the template to delete.

To delete instance groups in the console, select the instance group to delete from the list of instance groups and click the delete icon, as shown in Figure 6.26.

FIGURE 6.26 The instance group can be deleted in the console.



Delete instance groups from the command line using the following:

```
gcloud compute instance-groups managed delete-instances NAME
```

where *INSTANCE-GROUP-NAME* is the name of the instance group to delete.

To list templates and instance groups, use the following:

```
gcloud compute instance-templates list
gcloud compute instance-groups managed list-instances
```

To list the instances in an instance group, use the following:

```
gcloud compute instance-groups managed list-instances INSTANCE-GROUP-NAME
```

Instance Groups Load Balancing and Autoscaling

To deploy a scalable, highly available application, you can run that application on a load-balanced set of instances. GCP offers a number of types of load balancing, and they all require use of an instance group.

In addition to load balancing, managed instance groups can be configured to autoscale. You can configure an autoscaling policy to trigger adding or removing instances based on CPU utilization, monitoring metric, load-balancing capacity, or queue-based workloads.

Real World Scenario

No More Peak Capacity Planning

Prior to the advent of the cloud, IT organizations often had to plan their hardware purchases around the maximum expected load. This is called *peak capacity planning*. If there is little variation in load, peak capacity planning is a sound approach. Businesses with highly variable workloads, such as retailers in the United States that have high demand during the last two months of the year, would have to support idle capacity for months out of the year. Cloud computing and autoscaling have eliminated the need for peak capacity planning. Additional servers are acquired in minutes, not weeks or months. When capacity is not needed, it is dropped. Instance groups automate the process of adding and removing VMs, allowing cloud engineers to finely tune when to add and when to remove VMs.

When autoscaling, ensure you leave enough time for VMs to boot up or shut down before triggering another change in the cluster configuration. If the time between checks is too small, you may find that a recently added VM is not fully started before another is added. This can lead to more VMs being added than are actually needed.

Guidelines for Managing Virtual Machines

Here are some guidelines for managing VMs:

- Use labels and descriptions. This will help you identify the purpose of an instance and also help when filtering lists of instances.

- Use managed instance groups to enable autoscaling and load balancing. These are key to deploying scalable and highly available services.
- Use GPUs for numeric-intensive processing, such as machine learning and high-performance computing. For some applications, GPUs can give greater performance benefit than adding another CPU.
- Use snapshots to save the state of a disk or to make copies. These can be saved in Cloud Storage and act as backups.
- Use preemptible instances for workloads that can tolerate disruption. This will reduce the cost of the instance by up to 80 percent.

Summary

In this chapter, you learned how to a managed single VM instances and instance groups. Single VM instances can be created, configured, stopped, started, and deleted using Cloud Console or using `gcloud` commands from Cloud Shell or your local machine if you have SDK installed.

Snapshots are copies of disks and are useful as backups and for copying data to other instances. Images are copies of disks that are in a format suitable for creating VMs.

The main command used to manage VMs is the `gcloud compute instances` command. `gcloud` uses a hierarchical structure to order the command elements. The command begins with `gcloud`, followed by a GCP resource, such as `compute` for Compute Engine, followed by an entity type such as `instances` or `snapshots`. An action is then specified, such as `create`, `delete`, `list`, or `describe`.

GPUs can be attached to instances that have GPU libraries installed in the operating system. GPUs are used for compute-intensive tasks, such as building machine learning models.

Instance groups are groups of instances that are managed together. Managed instance groups have instances that are the same. These groups support load balancing and autoscaling.

Exam Essentials

Understand how to navigate Cloud Console. Cloud Console is the graphical interface for working with GCP. You can create, configure, delete, and list VM instances from the Compute Engine area of the console.

Understand how to install Cloud SDK. Cloud SDK allows you to configure default environment variables, such as a preferred zone, and issue commands from the command line. If you use Cloud Shell, Cloud SDK is already installed.

Know how to create a VM in the console and at the command line. You can specify machine type, choose an image, and configure disks with the console. You can use commands at the command line to list and describe, and you can find the same information in the console. Understand when to use customized images and how to deprecate them. Images are copies of contents of a disk, and they are used to create VMs. Deprecated marks an image as no longer supported.

Understand why GPUs are used and how to attach them to a VM. GPUs are used for compute-intensive operations; a common use case for using GPUs is machine learning. It is best to use an image that has GPU libraries installed. Understand how to determine which locations have GPUs available, because there are some restrictions. The CPU must be compatible with the GPU selected, and GPUs cannot be attached to shared memory machines. Know how GPU costs are charged.

Understand images and snapshots. Snapshots save the contents of disks for backup and data-sharing purposes. Images save the operating system and related configurations so you can create identical copies of the instance.

Understand instance groups and instance group templates. Instance groups are sets of instances managed as a single entity. Instance group templates specify the configuration of an instance group and the instances in it. Managed instance groups support autoscaling and load balancing.

Review Questions

You can find the answers in the Appendix.

1. Which page in Google Cloud Console would you use to create a single instance of a VM?
   - **A.** Compute Engine
   - **B.** App Engine
   - **C.** Kubernetes Engine
   - **D.** Cloud Functions

2. You view a list of Linux VM instances in the console. All have public IP addresses assigned. You notice that the SSH option is disabled for one of the instances. Why might that be the case?
   - **A.** The instance is preemptible and therefore does not support SSH.
   - **B.** The instance is stopped.
   - **C.** The instance was configured with the No SSH option.
   - **D.** The SSH option is never disabled.

3. You have noticed unusually slow response time when issuing commands to a Linux server, and you decide to reboot the machine. Which command would you use in the console to reboot?
   - **A.** Reboot
   - **B.** Reset
   - **C.** Restart
   - **D.** Shutdown followed by Startup

4. In the console, you can filter the list of VM instances by which of the following?
   - **A.** Labels only
   - **B.** Member of managed instance group only
   - **C.** Labels, status, or members of managed instance group
   - **D.** Labels and status only

5. You will be building a number of machine learning models on an instance and attaching GPU to the instance. When you run your machine learning models they take an unusually long time to run. It appears that GPU is not being used. What could be the cause of this?
   - **A.** GPU libraries are not installed.
   - **B.** The operating system is based on Ubuntu.
   - **C.** You do not have at least eight CPUs in the instance.
   - **D.** There isn't enough persistent disk space available.

6. When you add a GPU to an instance, you must ensure that:
   A. The instance is set to terminate during maintenance.
   B. The instance is preemptible.
   C. The instance does not have nonboot disks attached.
   D. The instance is running Ubuntu 14.02 or later.

7. You are using snapshots to save copies of a 100GB disk. You make a snapshot and then add 10GB of data. You create a second snapshot. How much storage is used in total for the two snapshots (assume no compression)?
   A. 210 GB, with 100GB for the first and 110GB for the second
   B. 110 GB, with 100GB for the first and 10GB for the second
   C. 110 GB, with 110 for the second (the first snapshot is deleted automatically)
   D. 221 GB, with 100GB for the first, 110GB for the second, plus 10 percent of the second snapshot (11 GB) for metadata overhead

8. You have decided to delegate the task of making backup snapshots to a member of your team. What role would you need to grant to your team member to create snapshots?
   A. Compute Image Admin
   B. Storage Admin
   C. Compute Snapshot Admin
   D. Compute Storage Admin

9. The source of an image may be:
   A. Only disks
   B. Snapshots or disks only
   C. Disks, snapshots, or another image
   D. Disks, snapshots, or any database export file

10. You have built images using Ubuntu 14.04 and now want users to start using Ubuntu 16.04. You don't want to just delete images based on Ubuntu 14.04, but you want users to know they should start using Ubuntu 16.04. What feature of images would you use to accomplish this?
    A. Redirection
    B. Deprecated
    C. Unsupported
    D. Migration

11. You want to generate a list of VMs in your inventory and have the results in JSON format. What command would you use?
    A. `gcloud compute instances list`
    B. `gcloud compute instances describe`
    C. `gcloud compute instances list --format json`
    D. `gcloud compute instances list --output json`

12. You would like to understand details of how GCP starts a virtual instance. Which optional parameter would you use when starting an instance to display those details?
    A. `--verbose`
    B. `--async`
    C. `--describe`
    D. `--details`

13. Which command will delete an instance named ch06-instance-3?
    A. `gcloud compute instances delete instance=ch06-instance-3`
    B. `gcloud compute instance stop ch06-instance-3`
    C. `gcloud compute instances delete ch06-instance-3`
    D. `gcloud compute delete ch06-instance-3`

14. You are about to delete an instance named ch06-instance-1 but want to keep its boot disk. You do not want to keep other attached disks. What `gcloud` command would you use?
    A. `gcloud compute instances delete ch06-instance-1 --keep-disks=boot`
    B. `gcloud compute instances delete ch06-instance-1 --save-disks=boot`
    C. `gcloud compute instances delete ch06-instance-1 --keep-disks=filesystem`
    D. `gcloud compute delete ch06-instance-1 --keep-disks=filesystem`

15. You want to view a list of fields you can use to sort a list of instances. What command would you use to see the field names?
    A. `gcloud compute instances list`
    B. `gcloud compute instances describe`
    C. `gcloud compute instances list --detailed`
    D. `gcloud compute instances describe --detailed`

16. You are deploying an application that will need to scale and be highly available. Which of these Compute Engine components will help achieve scalability and high availability?
    A. Preemptible instances
    B. Instance groups
    C. Cloud Storage
    D. GPUs

17. Before creating an instance group, you need to create what?
    A. Instances in the instance group
    B. Instance group template
    C. Boot disk image
    D. Source snapshot

18. How would you delete an instance group template using the command line?
    A. `gcloud compute instances instance-template delete`
    B. `glcoud compute instance-templates delete`
    C. `gcloud compute delete instance-template`
    D. `gcloud compute delete instance-templates`

19. What can be the basis for scaling up an instance group?
    A. CPU utilization and operating system updates
    B. Disk usage and CPU utilization only
    C. Network latency, load balancing capacity, and CPU utilization
    D. Disk usage and operating system updates only

20. An architect is moving a legacy application to Google Cloud and wants to minimize the changes to the existing architecture while administering the cluster as a single entity. The legacy application runs on a load-balanced cluster that runs nodes with two different configurations. The two configurations are required because of design decisions made several years ago. The load on the application is fairly consistent, so there is rarely a need to scale up or down. What GCP Compute Engine resource would you recommended using?
    A. Preemptible instances
    B. Unmanaged instance groups
    C. Managed instance groups
    D. GPUs

Chapter

7

Computing with Kubernetes

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

✓ **3.2 Deploying and implementing Kubernetes Engine resources**

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

This chapter introduces Kubernetes, a container orchestration system created and open sourced by Google. You will learn about the architecture of Kubernetes and the ways it manages workloads across nodes in a cluster. You will also learn how to manage Kubernetes resources with Cloud Console, Cloud Shell, and Cloud SDK. The chapter also covers how to deploy application pods (a Kubernetes structure) and monitor and log Kubernetes resources.

Introduction to Kubernetes Engine

Kubernetes Engine is Google Cloud Platform's (GCP's) managed Kubernetes service. With this service, GCP customers can create and maintain their own Kubernetes clusters without having to manage the Kubernetes platform.

Kubernetes runs containers on a cluster of virtual machines (VMs). It determines where to run containers, monitors the health of containers, and manages the full lifecycle of VM instances. This collection of tasks is known as *container orchestration*.

It may sound as if a Kubernetes cluster is similar to an instance group, which was discussed in Chapter 6. There are some similarities. Both are sets of VMs that can be managed as a group. Instance groups, however, are much more restricted. All VMs generally run the same image in an instance group. That is not the case with Kubernetes. Also, instance groups have no mechanism to support the deployment of containers. Containers offer a highly-portable, light-weight means of distributing and scaling your applications or workloads, like VMs, without replicating the guest OS. They can start and stop much faster (usually in seconds) and use fewer resources. You can think of a container as similar to shipping containers for applications and workloads. Like shipping containers that can ride on ships, trains, and trucks without reconfiguration, application containers can be moved from development laptops, to testing and production servers without reconfiguration. That would have to be done manually. Instance groups have some monitoring and can restart instances that fail, but Kubernetes has much more flexibility with regard to maintaining a cluster of servers.

Let's take a look at Kubernetes architecture, which consists of several objects and a set of controllers.

Keep in mind: when you use Kubernetes Engine, you will manage Kubernetes and your applications and workloads running in containers on the Kubernetes platform.

Kubernetes Cluster Architecture

A Kubernetes cluster consists of a cluster master and one or more nodes, which are the workers of the cluster. The cluster master controls the cluster and can be replicated and distributed for high-availability and fault tolerance.

The cluster master manages services provided by Kubernetes, such as the Kubernetes API, controllers, and schedulers. All interactions with the cluster are done through the master using the Kubernetes API. The cluster master issues the command that performs an action on a node. Users can also interact with a cluster using the `kubectl` command.

Nodes execute the workloads run on the cluster. Nodes are VMs that run containers configured to run an application. Nodes are primarily controlled by the cluster master, but some commands can be run manually. The nodes run an agent called *kubelet*, which is the service that communicates with the cluster master.

When you create a cluster, you can specify a machine type, which defaults to n1-standard-1 with 1 vCPU and 3.75GB of memory. These VMs run specialized operating systems optimized to run containers. Some of the memory and CPU is reserved for Kubernetes and so is not available to applications running on the node.

Kubernetes organizes processing into workloads. There are several organizing objects that make up the core functionality of how Kubernetes processes workloads.

Kubernetes Objects

Workloads are distributed across nodes in a Kubernetes cluster. To understand how work is distributed, it is important to understand some basic concepts, in particular the following:

- Pods
- Services
- Volumes
- Namespaces

Each of these objects contributes to the logical organization of workloads.

Pods

Pods are single instances of a running process in a cluster. Pods contain at least one container. They usually run a single container, but can run multiple containers. Multiple containers are used when two or more containers must share resources. Pods also use shared networking and storage across containers. Each pod gets a unique IP address and a set of ports. Containers connect to a port. Multiple containers in a pod connect to different ports and can talk to each other on localhost. This structure is designed to support running one instance of an application within the cluster as a pod. A pod allows its containers to behave as if they are running on an isolated VM, sharing common storage, one IP address, and a set of ports. By doing this, you can deploy multiple instance of the same application, or different instances of different applications on the same node or different nodes, without having to change their configuration.

Pods treat the multiple containers as a single entity for management purposes.

Pods are generally created in groups. Replicas are copies of pods and constitute a group of pods that are managed as a unit. Pods support autoscaling as well. Pods are considered ephemeral; that is, they are expected to terminate. If a pod is unhealthy—for example, if it is stuck in a waiting mode or crashing repeatedly—it is terminated. The mechanism that manages scaling and health monitoring is known as a *controller*.

You may notice that pods are similar to Compute Engine managed instance groups. A key difference is that pods are for executing applications in containers and may be placed on various nodes in the cluster, while managed instance groups all execute the same application code on

each of the nodes. Also, you typically manage instance groups yourself by executing commands in Cloud Console or through the command line. Pods are usually managed by a controller.

Services

Since pods are ephemeral and can be terminated by a controller, other services that depend on pods should not be tightly coupled to particular pods. For example, even though pods have unique IP addresses, applications should not depend on that IP address to reach an application. If the pod with that address is terminated and another is created, it may have another IP address. The IP address may be re-assigned to another pod running a different container.

Kubernetes provides a level of indirection between applications running in pods and other applications that call them: it is called a *service*. A service, in Kubernetes terminology, is an object that provides API endpoints with a stable IP address that allow applications to discover pods running a particular application. Services update when changes are made to pods, so they maintain an up-to-date list of pods running an application.

ReplicaSet

A ReplicaSet is a controller used by a deployment that ensures the correct number identical of pods are running. For example, if a pod is determined to be unhealthy, a controller will terminate that pod. The ReplicaSet will detect that not enough pods for that application or workload are running and will create another. ReplicaSets are also used to update and delete pods.

Deployment

Another important concept in Kubernetes is the deployment. Deployments are sets of identical pods. The members of the set may change as some pods are terminated and others are started, but they are all running the same application. The pods all run the same application because they are created using the same pod template.

A pod template is a definition of how to run a pod. The description of how to define the pod is called a *pod specification*. Kubernetes uses this definition to keep a pod in the state specified in the template. That is, if the specification has a minimum number of pods that should be in the deployment and the number falls below that, then additional pods will be added to the deployment by calling on a ReplicaSet.

StatefulSet

Deployments are well suited to stateless applications. Those are applications that do not need to keep track of their state. For example, an application that calls an API to perform a calculation on the input values does not need to keep track of previous calls or calculations. An application that calls that API may reach a different pod each time it makes a call. There are times, however, when it is advantageous to have a single pod respond to all calls for a client during a single session.

StatefulSets are like deployments, but they assign unique identifiers to pods. This enables Kubernetes to track which pod is used by which client and keep them together. StatefulSets are used when an application needs a unique network identifier or stable persistent storage.

Job

A job is an abstraction about a workload. Jobs create pods and run them until the application completes a workload. Job specifications are specified in a configuration file and include specifications about the container to use and what command to run.

Now that you're familiar with how Kubernetes is organized and how workloads are run, we'll cover how to deploy a Kubernetes cluster using Kubernetes Engine.

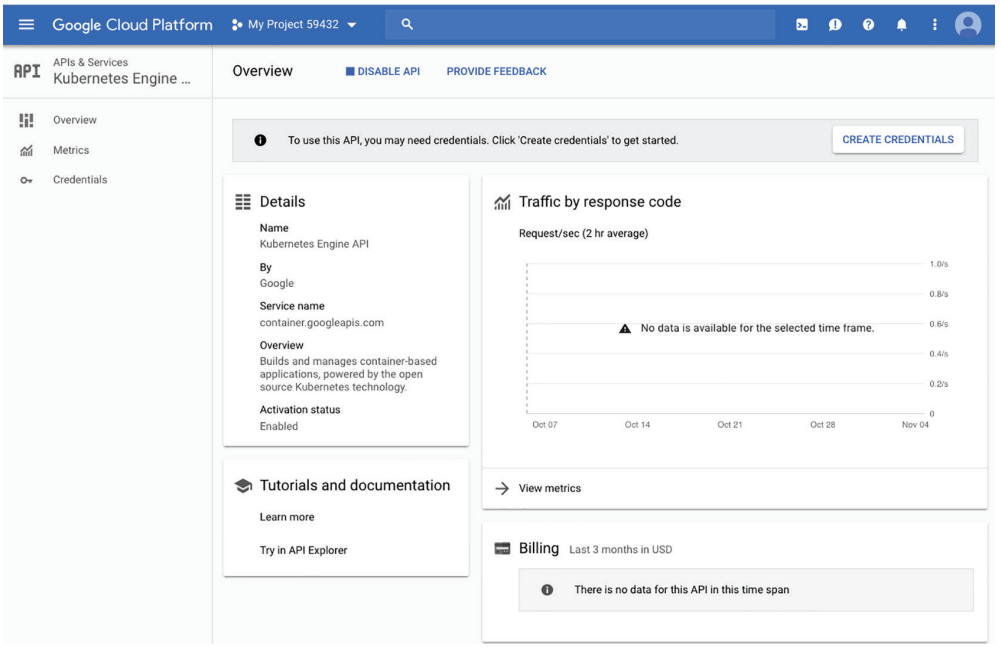
Deploying Kubernetes Clusters

Kubernetes clusters can be deployed using either Cloud Console or the command line in Cloud Shell, or your local environment if Cloud SDK is installed.

Deploying Kubernetes Clusters Using Cloud Console

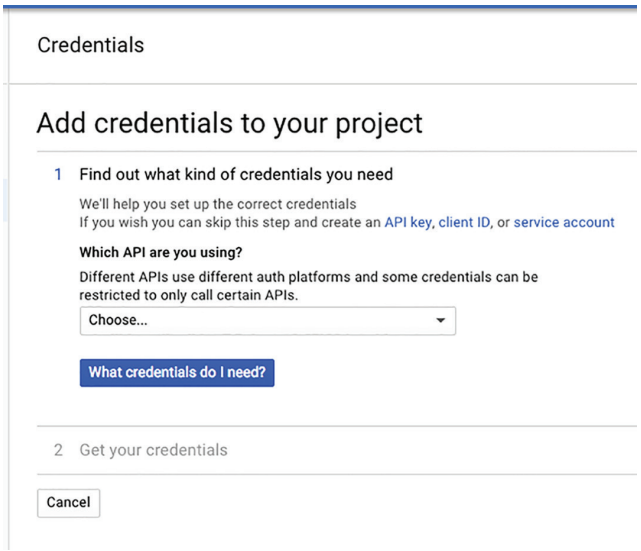
To use Kubernetes Engine, you will need to enable the Kubernetes Engine API. Once you have enabled the API, you can navigate to the Kubernetes Engine page in Cloud Console. Figure 7.1 shows an example of the Overview page.

FIGURE 7.1 The Overview page of the Kubernetes Engine section of Cloud Console



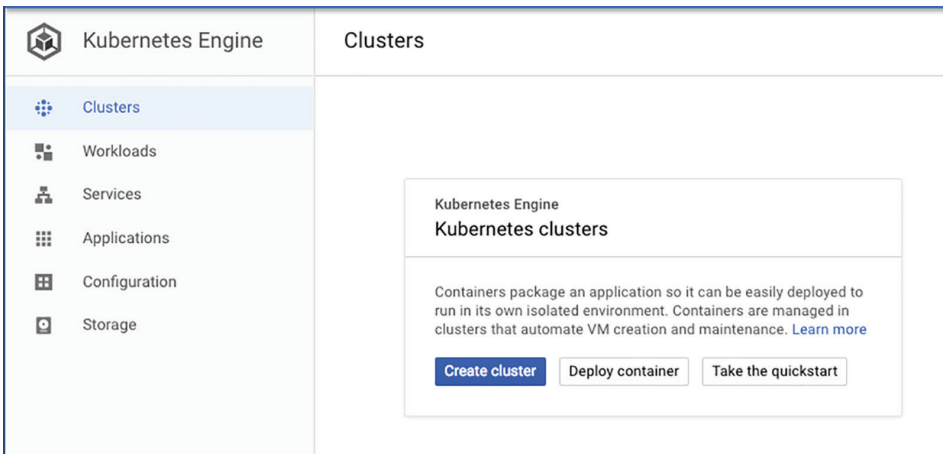
The first time you use Kubernetes Engine, you may need to create credentials. You can do this by clicking the Create Credentials button near the top of the Overview page. A form such as the one shown in Figure 7.2 will appear. You can specify which API you are using and then generate your credentials.

FIGURE 7.2 The form for creating credentials needed to use Kubernetes Engine



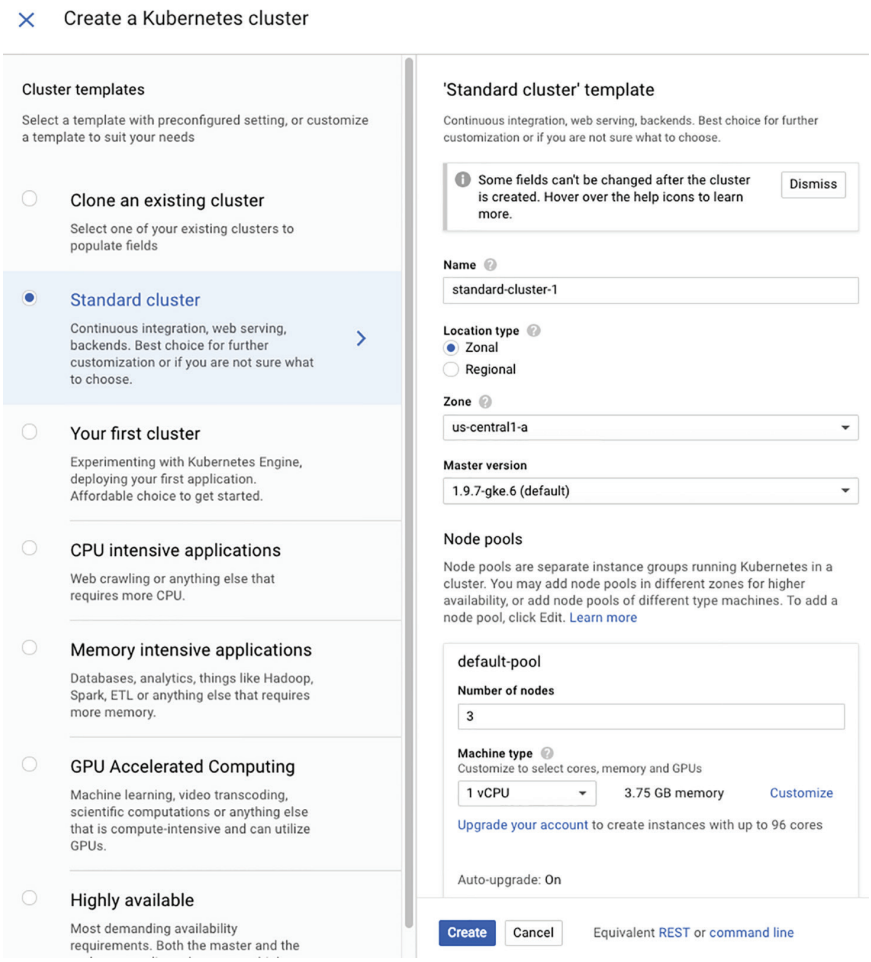
After creating credentials, if needed, you can create a cluster. Figure 7.3 shows the first page in the cluster creation step.

FIGURE 7.3 The first form for creating a Kubernetes cluster in Cloud Console



When you click Create Cluster, you will be presented with the option to choose from several templates, as shown in Figure 7.4. The templates vary in the number of vCPUs, memory, and use of GPUs. For example, the Standard Cluster template uses three nodes with one vCPU and 3.75 GB of memory, while the CPU Intensive template uses four vCPUs and 3.6GB of memory.

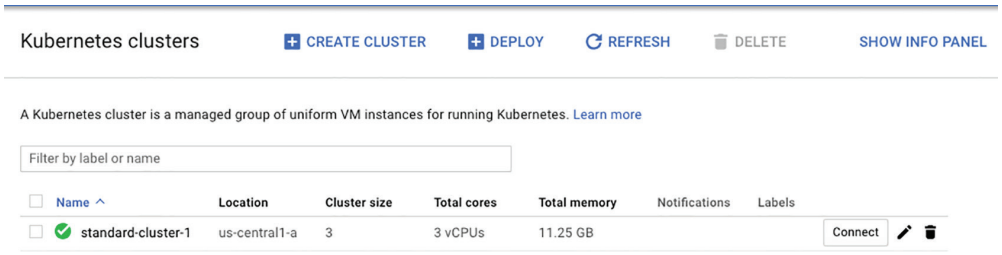
FIGURE 7.4 Templates for creating a Kubernetes cluster



You can modify the parameters provided in the template. For example, if you want to run VMs in different zones to improve availability, you can specify multiple node pools. Node pools are instance groups in a Kubernetes cluster. They're much like a Managed Instance Group but not the same.

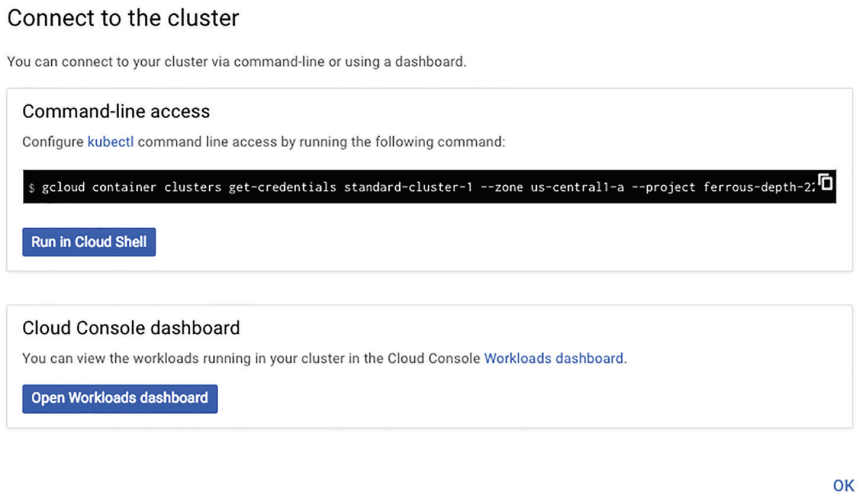
It can take a few minutes to create a cluster. When the cluster is created, it will appear in the list of clusters, as in Figure 7.5.

FIGURE 7.5 The cluster listing shows the number of instances, total cores, and total memory.



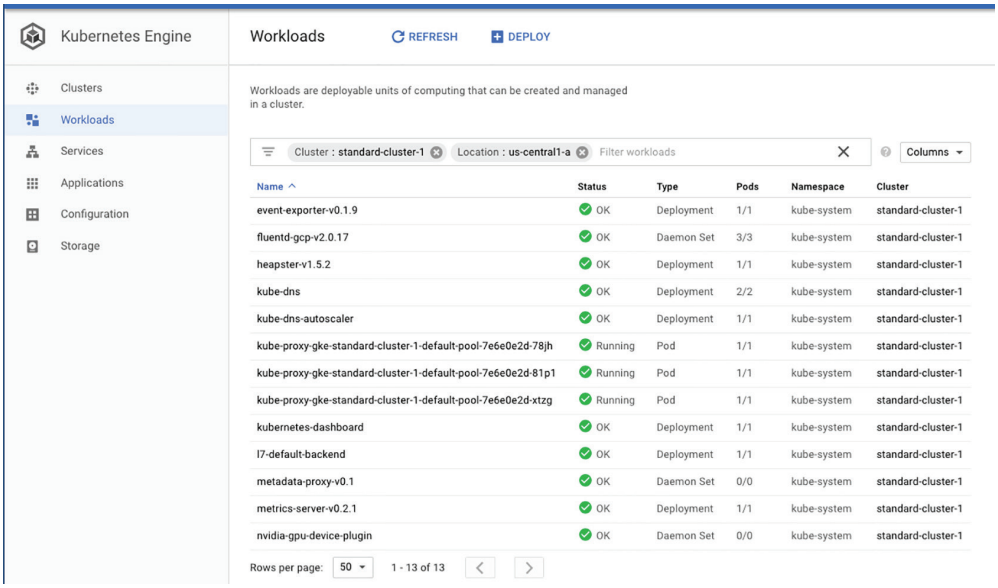
From the listing of clusters, you can edit, delete, and connect to a cluster. When you click Connect, you receive a `gcloud` command to connect to the cluster from the command line. You also have the option of viewing the Workloads page, as shown in Figure 7.6.

FIGURE 7.6 You can connect to the cluster either by using a `gcloud` command from the command line or by viewing the Workloads page.



Kubernetes runs a number of workloads to manage the cluster. You can view the currently running workloads in the Workloads page of the Kubernetes Engine section of Cloud Console. Figure 7.7 shows a subset of the workloads running on a newly started cluster.

FIGURE 7.7 The Workloads page lists currently running workloads.



Deploying Kubernetes Clusters Using Cloud Shell and Cloud SDK

Like other GCP services, Kubernetes Engine can be managed using the command line. The basic command for working with Kubernetes Engine is the following `gcloud` command:

```
gcloud beta container
```

Notice that Kubernetes Engine commands include the word *beta*. Google indicates a service that is not yet in general availability by including the word *alpha* or *beta* in the `gcloud` command. By the time you read this, Kubernetes Engine may be generally available, in which case the *beta* term will no longer be used.

This `gcloud` command has many parameters, including the following:

- Project
- Zone
- Machine type
- Image type
- Disk type
- Disk size
- Number of nodes

A basic command for creating a cluster looks like this:

```
gcloud container clusters create ch07-cluster --num-nodes=3 --region=us-central1
```

Commands for creating clusters can become quite long. For example, here is the command to create a cluster using the standard template:

```
gcloud beta container --project "ferrous-depth-220417" clusters create
"standard-cluster-2" --zone "us-central1-a" --username "admin"
--cluster-version "1.9.7-gke.6" --machine-type "n1-standard-1"
--image-type "COS" --disk-type "pd-standard" --disk-size "100" --scopes
"https://www.googleapis.com/auth/compute","https://www.googleapis.com/auth/
devstorage.read_only","https://www.googleapis.com/auth/logging.write",
"https://www.googleapis.com/auth/monitoring","https://www.googleapis.com/auth/
servicecontrol","https://www.googleapis.com/auth/service.management.readonly",
"https://www.googleapis.com/auth/trace.append" --num-nodes "3" --enable-cloud-
logging --enable-cloud-monitoring --network "projects/ferrous-depth-220417/
global/networks/default" --subnetwork "projects/ferrous-depth-220417/regions/
us-central1/subnetworks/default" --addons HorizontalPodAutoscaling,
HttpLoadBalancing,KubernetesDashboard --enable-autoupgrade --enable-autorepair
```

Rather than write this kind of command from scratch, you can use Cloud Console to select a template and then use the option to generate the equivalent command line from the Create Cluster form.

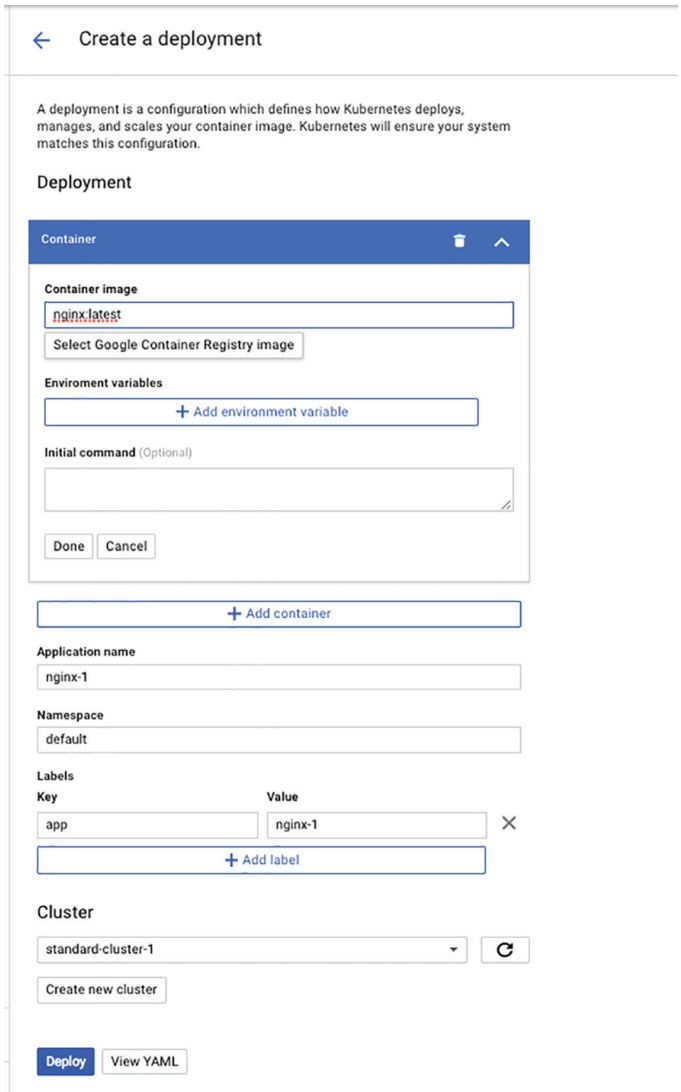
Deploying Application Pods

Now that you have created a cluster, let's deploy an application.

From the Cluster page of the Kubernetes Engine on Cloud Console, select Create Deployment. A form such as the one in Figure 7.8 appears. Within this form you can specify the following:

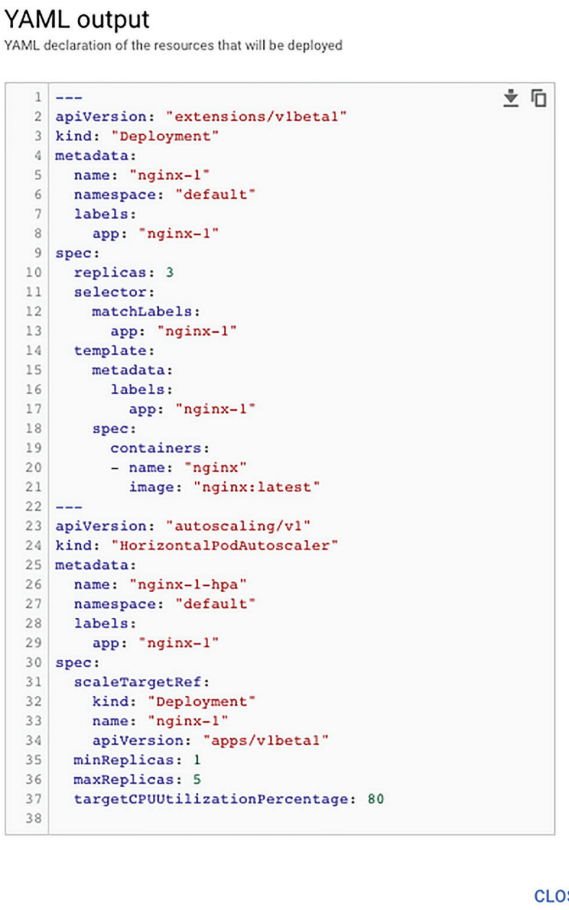
- Container image
- Environment variables
- Initial command
- Application name
- Labels
- Namespace
- Cluster to deploy to

FIGURE 7.8 The Create Deployment option provides a form to specify a container to run and an initial command to start the application running.



Once you have specified a deployment, you can display the corresponding YAML specification, which can be saved and used to create deployments from the command line. Figure 7.9 shows an example deployment YAML file. The output is always displayed in YAML format.

FIGURE 7.9 YAML specification for a Kubernetes deployment



In addition to installing Cloud SDK, you will need to install the Kubernetes command-line tool `kubectl` to work with clusters from the command line. You can do this with the following command:

```
gcloud components install kubectl
```

If the Cloud SDK Manager is disabled, you may receive an error when running `gcloud components install kubectl`. If that occurs, you can use the component manager, following the instructions at `https://cloud.google.com/sdk/install`.

The Cloud SDK component manager works only if you don't install SDK through another package manager. If you want to use the component manager, you can install it using one of these methods:

```
https://cloud.google.com/sdk/downloads#versioned

https://cloud.google.com/sdk/downloads#interactive
```

Additional packages are available in our deb and yum repos; all the same components are available, and you just need to use your existing package manager to install them.

```
https://cloud.google.com/sdk/downloads#apt-get
https://cloud.google.com/sdk/downloads#yum
```

You can then use `kubectl` to run a Docker image on a cluster by using the `kubectl run` command. Here's an example:

```
kubectl run ch07-app-deploy --image=ch07-app --port=8080
```

This will run a Docker image called `ch07-app` and make its network accessible on port 8080. If after some time you'd like to scale up the number of replicas in the deployment, you can use the `kubectl scale` command:

```
kubectl scale deployment ch07-app-deploy --replicas=5
```

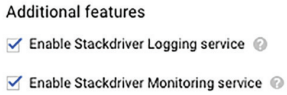
This example would create five replicas.

Monitoring Kubernetes

Stackdriver is GCP's comprehensive monitoring, logging, and alerting product. It can be used to monitor Kubernetes clusters.

When creating a cluster, be sure to enable Stackdriver monitoring and logging by selecting Advanced Options in the Create Cluster form in Cloud Console. Under Additional Features, choose Enable Logging Service and Enable Monitoring Service, as shown in Figure 7.10.

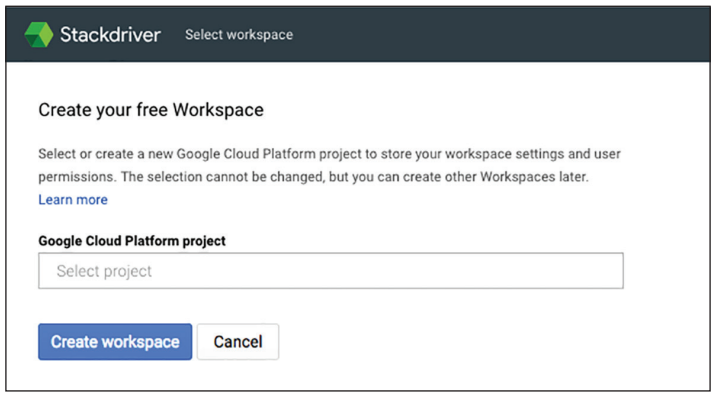
FIGURE 7.10 Expanding the Advanced Options in the Create Cluster dialog will show two check boxes for enabling Stackdriver logging and monitoring.



To set up Stackdriver from Cloud Console, select Stackdriver from the top-level menu on the left. Initially, you will need to create a workspace in your project by selecting a new workspace and launching monitoring when prompted (see Figure 7.11). Once a workspace is created, you can monitor your GCP resources, including Kubernetes clusters.

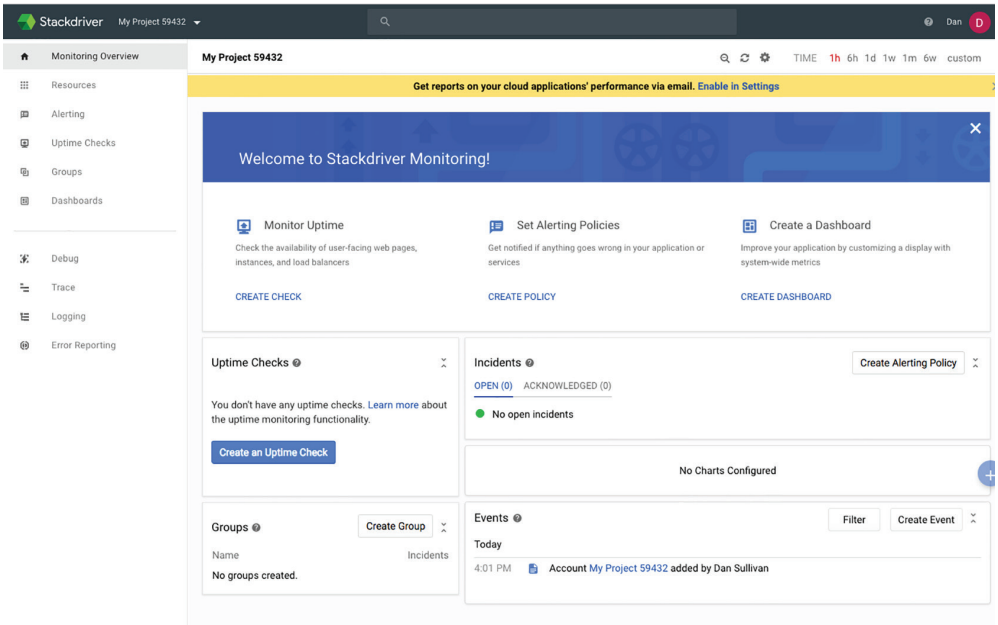
Workspaces are resources for monitoring and can support up to 100 monitored projects. Workspaces contain dashboards, alerting policies, group definitions, and notification checks.

FIGURE 7.11 An initial dialog box to create a workspace in Stackdriver



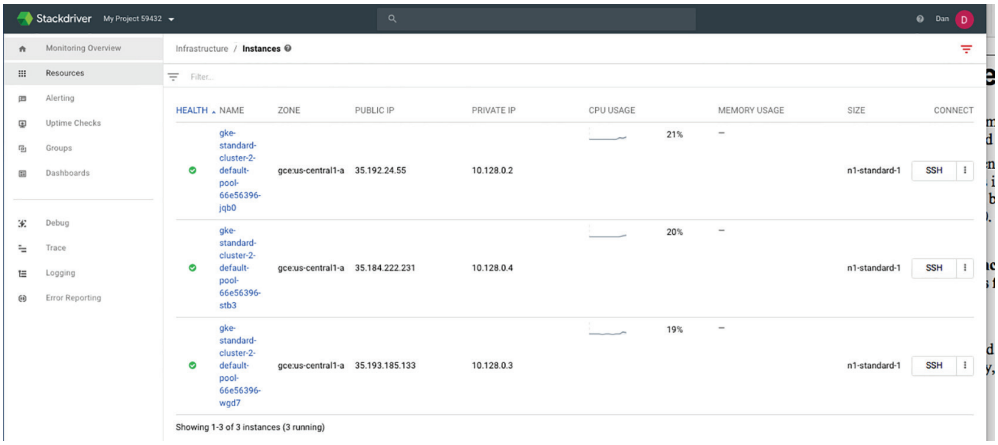
After you create a workspace, open Stackdriver, and it displays the Monitoring Overview page, shown in Figure 7.12.

FIGURE 7.12 The Stackdriver Monitoring Overview page



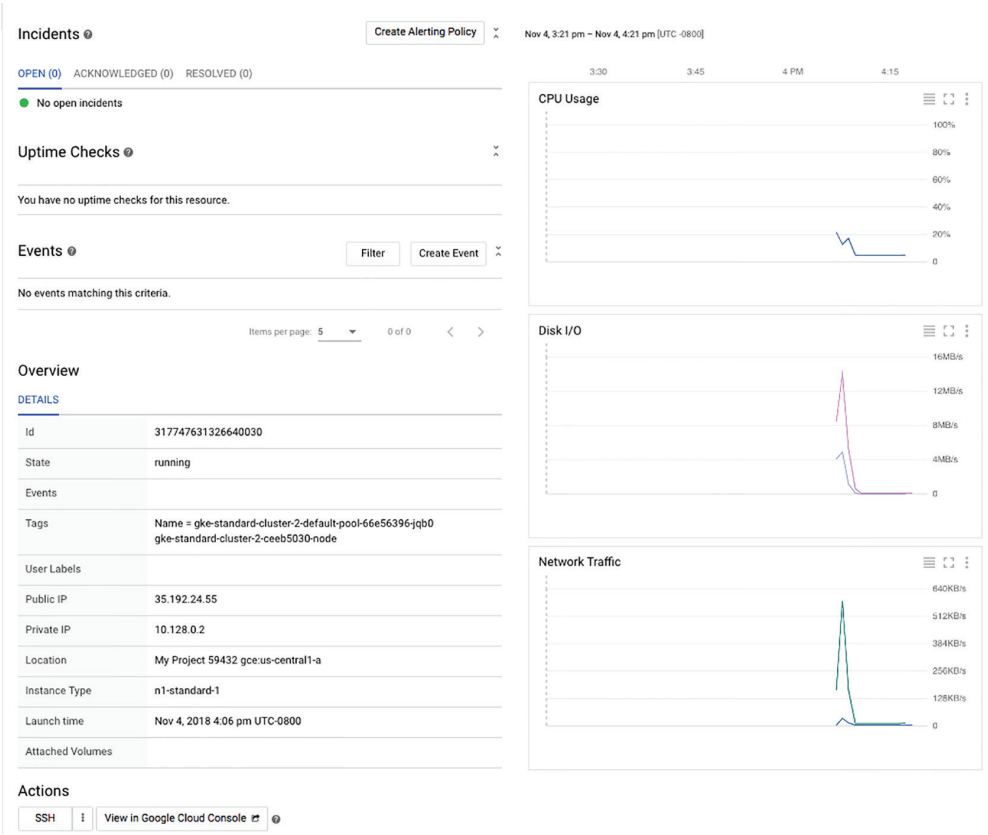
From the Overview Page, click Resources and select Instances to list the instances in your cluster. This displays a list such as in Figure 7.13.

FIGURE 7.13 List of instances in a Kubernetes cluster



Click the names of any of the instances to show a detailed page of monitoring information, as shown in Figure 7.14.

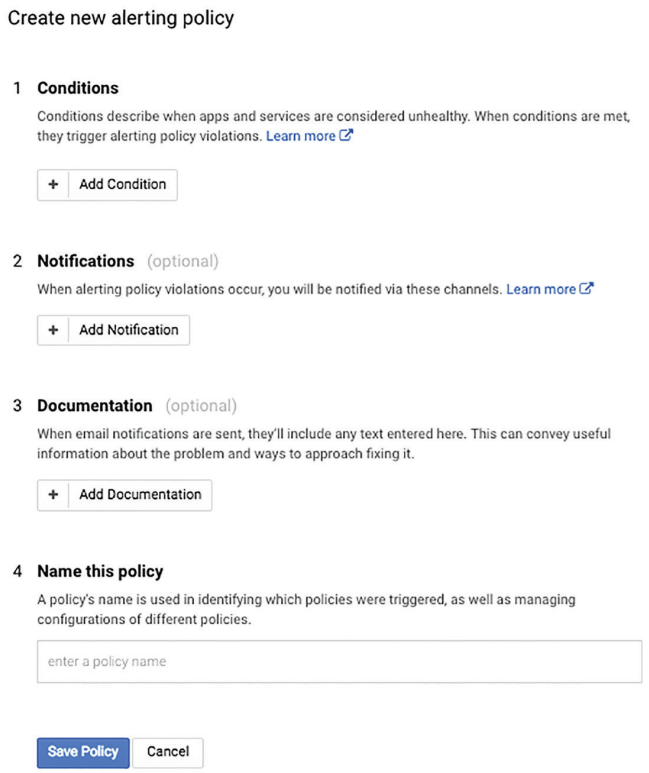
FIGURE 7.14 A typical detailed monitoring page of an instance running in a Kubernetes cluster



From the Details page, you can view an overview of details about the instance and view CPU usage, disk IO, and network traffic. You can also create alerting policies to notify you if some condition, such as high CPU utilization, occurs on the instance. When you create alerts, they can be applied to an individual instance in the cluster or to all instances in the cluster.

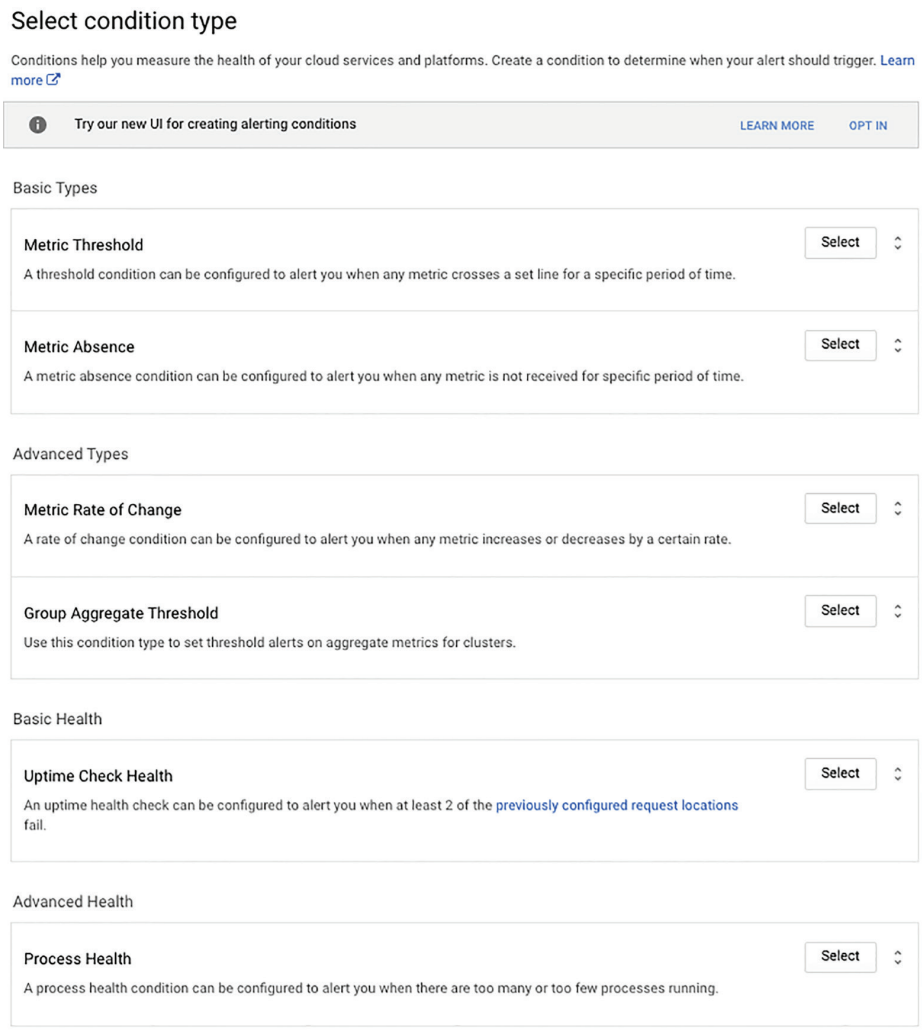
In the detail Stackdriver page, create an alert by clicking the Create Alerting Policy button. This displays a dialog such as in Figure 7.15, from which you can create conditions, notifications, and documentation. You can also name the policy.

FIGURE 7.15 When creating an alerting policy, this form allows you to specify components of the policy.



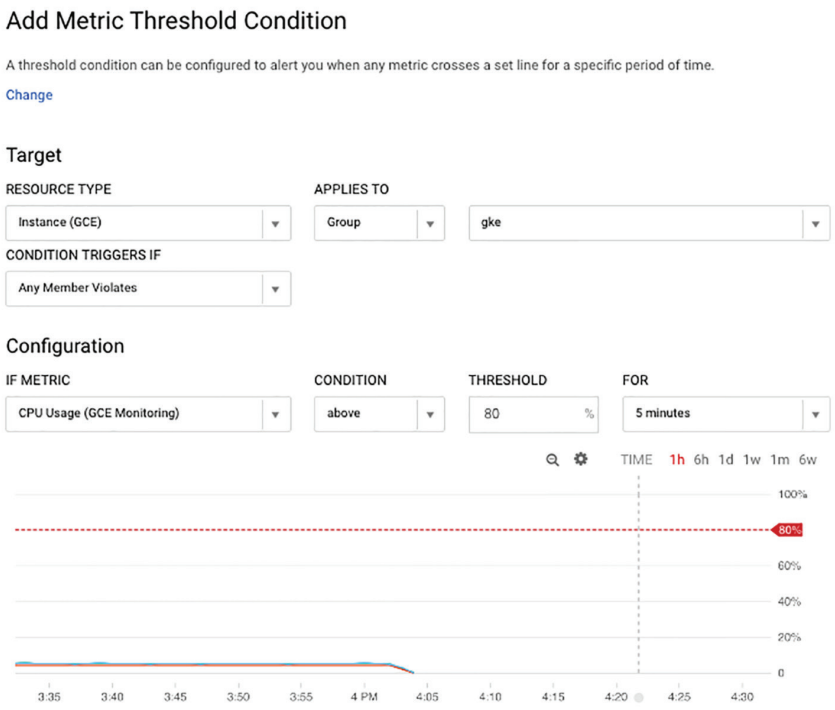
When you add a condition, a form such as the one shown in Figure 7.16 appears.

FIGURE 7.16 Stackdriver supports a number of condition types.



Select Metric Threshold to display a form like Figure 7.17, which shows how to specify an alert on CPU utilization over 80 percent for 5 minutes.

FIGURE 7.17 Stackdriver metric threshold conditions are based on a set of monitored resources, such as CPU utilization.

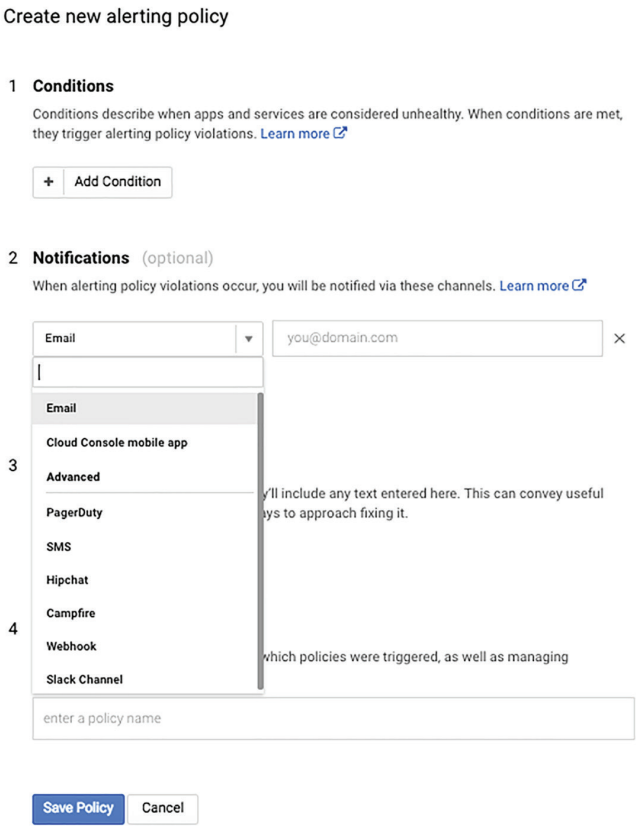


Stackdriver will need to know how to notify you if an alert is triggered. You can specify your choice of notification channels in the Create New Alerting Policy form, as shown in Figure 7.18. Channels include email, webhooks, and SMS text messaging as well as third-party tools such as PagerDuty, Campfire, and Slack.

Stackdriver supports more advanced alerting as well, including process health, uptime checks, group aggregate thresholds, and metric rates of change.

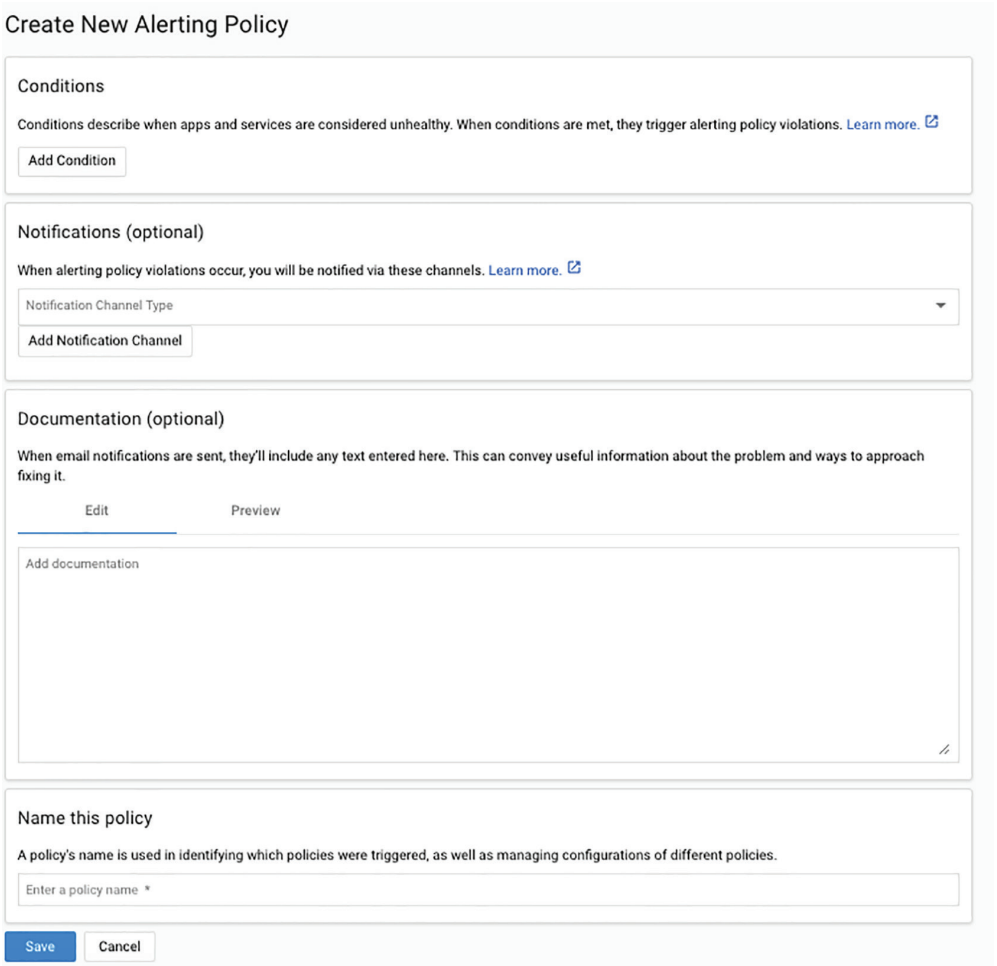
Let's walk through an example of creating a policy to monitor CPU utilization.

FIGURE 7.18 Stackdriver supports a number of condition types.



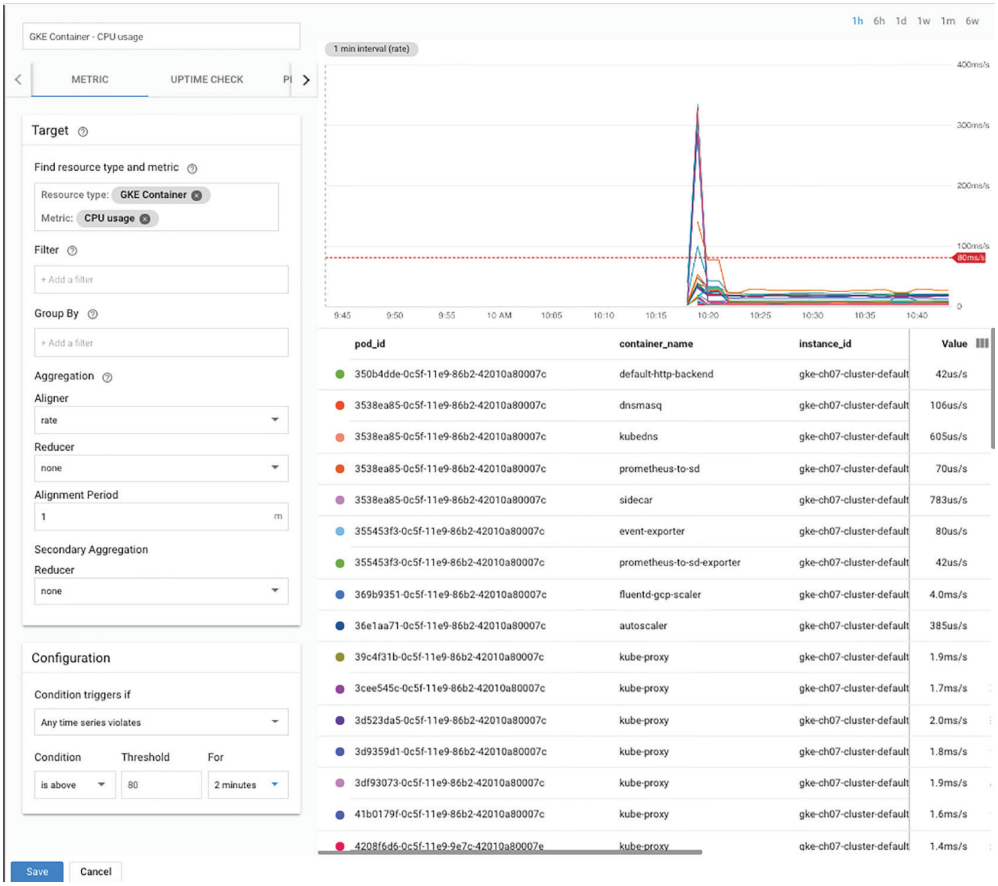
For more details on monitoring, see Chapter 18. To create a policy to monitor CPU utilization, navigate to the monitoring page in Stackdriver and click Create Policy. This will display the form to create a policy, which is a four-step process: create a condition, specify a notification channel, add a description, and name the policy. (See Figure 7.19.)

FIGURE 7.19 Creating a policy to monitor CPU utilization



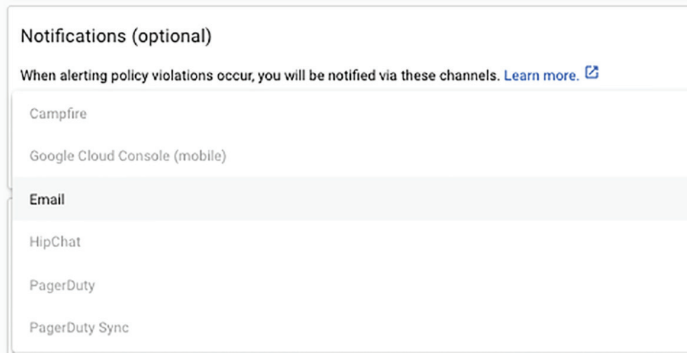
Click Add Condition to display a form like that shown in Figure 7.20.

FIGURE 7.20 Adding a condition to a policy



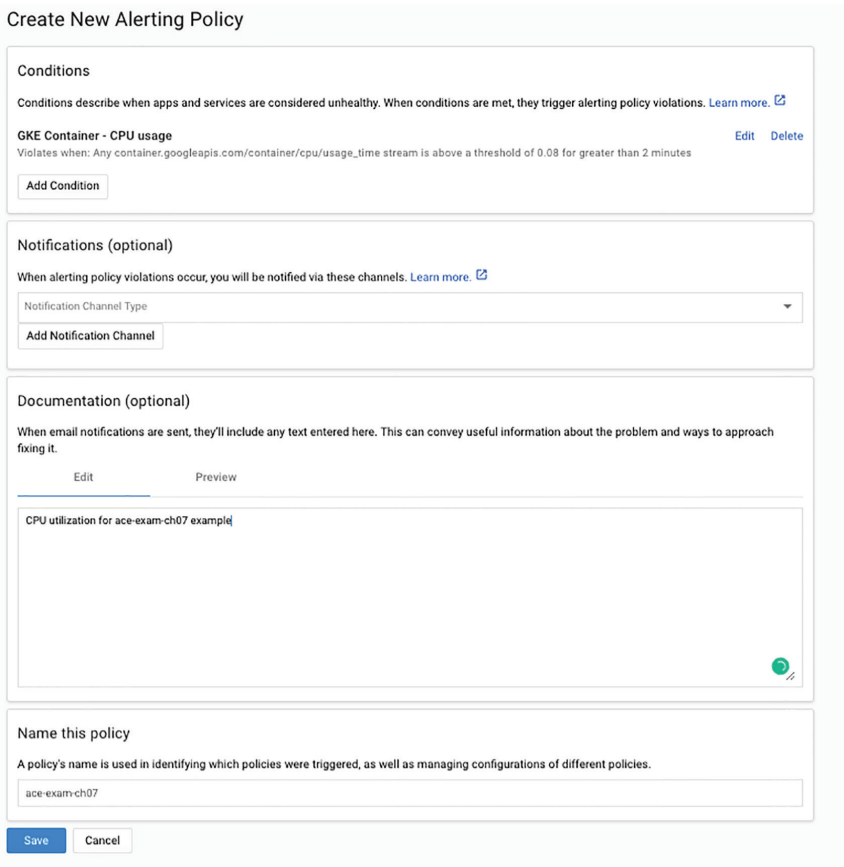
In the Filter parameter, enter **GKE Container** and **CPU Usage**. In the Configuration section, specify 80 percent as the threshold and 2 minutes as the time period. Save the condition. This will return to the Create Policy form. In the Notification parameter, select Email from the drop-down list, as shown in Figure 7.21.

FIGURE 7.21 Choosing a notification channel



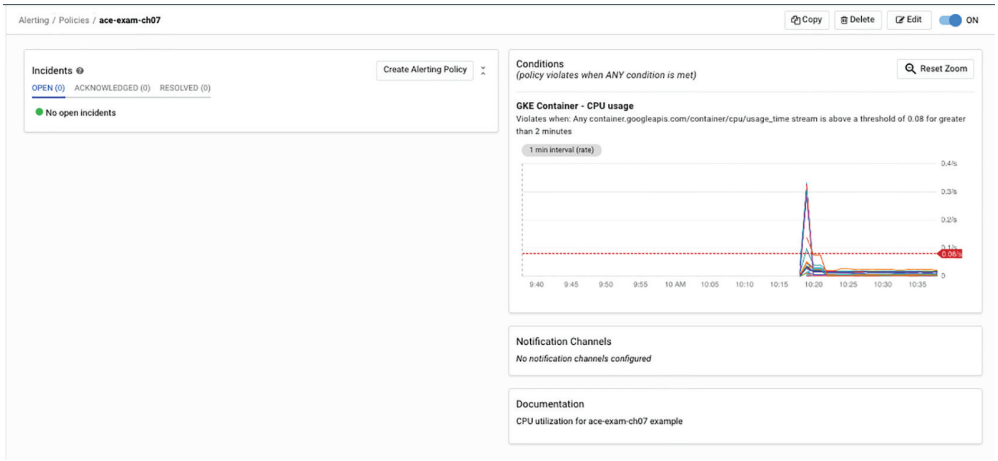
Add a description and policy name, as shown in Figure 7.22.

FIGURE 7.22 A completed policy creation form



Save the policy specification to display a monitoring summary, as shown in Figure 7.23.

FIGURE 7.23 Monitoring results of policy on CPU usage



Summary

Kubernetes Engine is a container orchestration system for deploying applications to run in clusters. Kubernetes is architected with a single cluster manager and worker nodes.

Kubernetes uses the concept of pods are instances running a container. It is possible to run multiple containers in a pod, but that occurs less frequently than single-container pods. ReplicaSets are controllers for ensuring that the correct number of pods are running. Deployments are sets of identical pods. StatefulSets are a type of deployment used for stateful applications.

Kubernetes clusters can be deployed through Cloud Console or by using `gcloud` commands. You deploy applications by bundling the application in a container and using the console or the `kubectl` command to create a deployment that runs the application on the cluster.

Stackdriver is used to monitor instances in clusters. You can create alerts and have notifications delivered to a variety of channels.

Exam Essentials

Understand that Kubernetes is a container orchestration system. Kubernetes Engine is a GCP product that provides Kubernetes to GCP customers. Kubernetes manages containers that run in a set of VM instances.

Understand that Kubernetes uses a single cluster master that controls nodes that execute workloads. Kubernetes uses the master to coordinate execution and monitor the health of pods. If there is a problem with a pod, the master can correct the problem and reschedule the disrupted job.

Be able to describe pods. Pods are single instances of a running process, services provide a level of indirection between pods and clients calling services in the pods, a ReplicaSet is a kind of controller that ensures that the right number of pods are running, and a deployment is a set of identical pods.

Kubernetes can be deployed using Cloud Console or using `gcloud` commands. `gcloud` commands manipulate the Kubernetes Engine service, while `kubectl` commands are used to manage the internal state of clusters from the command line. The base command for working with Kubernetes Engine is `gcloud container`. Note that `gcloud` and `kubectl` have different command syntaxes. `kubectl` commands specify a verb and then a resource, as in `kubectl scale deployment ...`, while `gcloud` specifies a resource before the verb, as in `gcloud container clusters create`. Deployments are created using Cloud Console or at the command line using a YAML specification.

Deployments are sets of identical pods. StatefulSets are a type of deployment used for stateful applications. Kubernetes is monitored using Stackdriver. Stackdriver can be configured to generate alerts and notify you on a variety of channels. To monitor the state of a cluster, you can create a policy that monitors a metric, like CPU utilization, and have notifications sent to email or other channels.

Review Questions

You can find the answers in the Appendix.

1. A new engineer is asking for clarification about when it is best to use Kubernetes and when to use instance groups. You point out that Kubernetes uses instance groups. What purpose do instance groups play in a Kubernetes cluster?
   - **A.** They monitor the health of instances.
   - **B.** They create pods and deployments.
   - **C.** They create sets of VMs that can be managed as a unit.
   - **D.** They create alerts and notification channels.

2. What kinds of instances are required to have a Kubernetes cluster?
   - **A.** A cluster master and nodes to execute workloads.
   - **B.** A cluster master, nodes to execute workloads, and Stackdriver nodes to monitor node health.
   - **C.** Kubernetes nodes; all instances are the same.
   - **D.** Instances with at least four vCPUs.

3. What is a pod in Kubernetes?
   - **A.** A set of containers
   - **B.** Application code deployed in a Kubernetes cluster
   - **C.** A single instance of a running process in a cluster
   - **D.** A controller that manages communication between clients and Kubernetes services

4. You have developed an application that calls a service running in a Kubernetes cluster. The service runs in pods that can be terminated if they are unhealthy and replaced with other pods that might have a different IP address. How should you code your application to ensure it functions properly in this situation?
   - **A.** Query Kubernetes for a list of IP addresses of pods running the service you use.
   - **B.** Communicate with Kubernetes services so applications do not have to be coupled to specific pods.
   - **C.** Query Kubernetes for a list of pods running the service you use.
   - **D.** Use a `gcloud` command to get the IP addresses needed.

5. You have noticed that an application's performance has degraded significantly. You have recently made some configuration changes to resources in your Kubernetes cluster and suspect that those changes have alerted the number of pods running in the cluster. Where would you look for details on the number of pods that should be running?
   - **A.** Deployments
   - **B.** Stackdriver
   - **C.** ReplicaSet
   - **D.** Jobs

6. You are deploying a high availability application in Kubernetes Engine. You want to maintain availability even if there is a major network outage in a data center. What feature of Kubernetes Engine would you employ?
   A. Multiple instance groups
   B. Multizone/region cluster
   C. Regional deployments
   D. Load balancing

7. You want to write a script to deploy a Kubernetes cluster with GPUs. You have deployed clusters before, but you are not sure about all the required parameters. You need to deploy this script as quickly as possible. What is one way to develop this script quickly?
   A. Use the GPU template in the Kubernetes Engine cloud console to generate the `gcloud` command to create the cluster
   B. Search the Web for a script
   C. Review the documentation on gcloud parameters for adding GPUs
   D. Use an existing script and add parameters for attaching GPUs

8. What `gcloud` command will create a cluster named `ch07-cluster-1` with four nodes?
   A. `gcloud beta container clusters create ch07-cluster-1 --num-nodes=4`
   B. `gcloud container beta clusters create ch07-cluster-1 --num-nodes=4`
   C. `gcloud container clusters create ch07-cluster-1 --num-nodes=4`
   D. `gcloud beta container clusters create ch07-cluster-1 4`

9. When using Create Deployment from Cloud Console, which of the following cannot be specified for a deployment?
   A. Container image
   B. Application name
   C. Time to live (TTL)
   D. Initial command

10. Deployment configuration files created in Cloud Console use what type of file format?
    A. CSV
    B. YAML
    C. TSV
    D. JSON

11. What command is used to run a Docker image on a cluster?
    A. `gcloud container run`
    B. `gcloud beta container run`
    C. `kubectl run`
    D. `kubectl beta run`

12. What command would you use to have 10 replicas of a deployment named `ch07-app-deploy`?
    A. `kubectl upgrade deployment ch07-app-deploy --replicas=5`
    B. `gcloud containers deployment ch07-app-deploy --replicas=5`
    C. `kubectl scale deployment ch07-app-deploy --replicas=10`
    D. `kubectl scale deployment ch07-app-deploy --pods=5`

13. Stackdriver is used for what operations on Kubernetes clusters?
    A. Notifications only
    B. Monitoring and notifications only
    C. Logging only
    D. Notifications, monitoring, and logging

14. Before monitoring a Kubernetes cluster, what must you create with Stackdriver?
    A. Log
    B. Workspace
    C. Pod
    D. ReplicaSet

15. What kind of information is provided in the Details page about an instance in Stackdriver?
    A. CPU usage only
    B. Network traffic only
    C. Disk I/O, CPU usage, and network traffic
    D. CPU usage and disk I/O

16. When creating an alerting policy, what can be specified?
    A. Conditions, notifications, and time to live
    B. Conditions, notifications, and documentation
    C. Conditions only
    D. Conditions, documentation, and time to live

17. Your development team needs to be notified if there is a problem with applications running on several Kubernetes clusters. Different team members prefer different notification methods in addition to Stackdriver alerting. What is the most efficient way to send notifications and meet your team's requests?
    A. Set up SMS text messaging, Slack, and email notifications on an alert.
    B. Create a separate alert for each notification channel.
    C. Create alerts with email notifications and have those notification emails forwarded to other notification systems.
    D. Use a single third-party notification mechanism.

18. A new engineer is trying to set up alerts for a Kubernetes cluster. The engineer seems to be creating a large number of alerts and you are concerned this is not the most efficient way and will lead to more maintenance work than required. You explain that a more efficient way is to create alerts and apply them to what?
    - **A.** One instance only
    - **B.** An instance or entire group
    - **C.** A group only
    - **D.** A pod

19. You are attempting to execute commands to initiate a deployment on a Kubernetes cluster. The commands are not having any effect. You suspect that a Kubernetes component is not functioning correctly. What component could be the problem?
    - **A.** The Kubernetes API
    - **B.** A StatefulSet
    - **C.** Cloud SDK `gcloud` commands
    - **D.** ReplicaSet

20. You have deployed an application to a Kubernetes cluster. You have noticed that several pods are starved for resources for a period of time and the pods are shut down. When resources are available, new instantiations of those pods are created. Clients are still able to connect to pods even though the new pods have different IP addresses from the pods that were terminated. What Kubernetes component makes this possible?
    - **A.** Services
    - **B.** ReplicaSet
    - **C.** Alerts
    - **D.** StatefulSet

Chapter

8

Managing Kubernetes Clusters

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

✓ **4.2 Managing Kubernetes Engine resources**

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

This chapter describes how to perform basic Kubernetes management tasks, including the following:

- Viewing the status of Kubernetes clusters
- Viewing image repositories and image details
- Adding, modifying, and removing nodes
- Adding, modifying, and removing pods
- Adding, modifying, and removing services

You'll see how to perform each of these tasks using Google Cloud Console and Cloud SDK, which you can use locally on your development machines, on GCP virtual machines, and by using Cloud Shell.

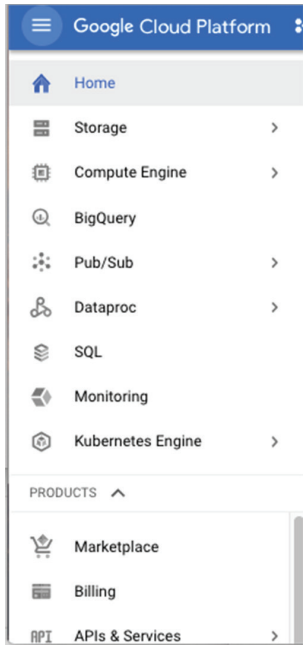
Viewing the Status of a Kubernetes Cluster

Assuming you have created a cluster using the steps outlined in Chapter 7, you can view the status of a Kubernetes cluster using either Google Cloud Console or the `gcloud` commands.

Viewing the Status of Kubernetes Clusters Using Cloud Console

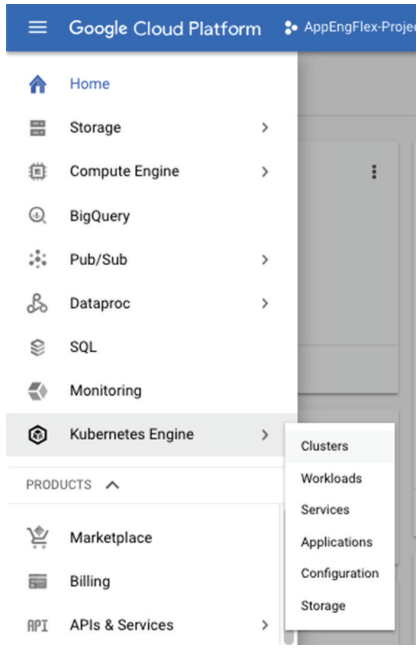
Starting from the Cloud Console home page, open the navigation menu by clicking the three stacked lines icon in the upper-left corner. This displays the list of GCP services, as shown in Figure 8.1.

FIGURE 8.1 Navigation menu in Google Cloud Console



Select Kubernetes Engine from the lists of services, as shown in Figure 8.2.

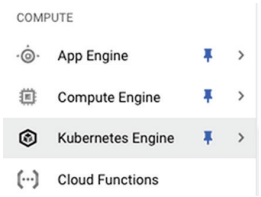
FIGURE 8.2 Selecting Kubernetes Engine from the navigation menu



Pinning Services to the Top of the Navigation Menu

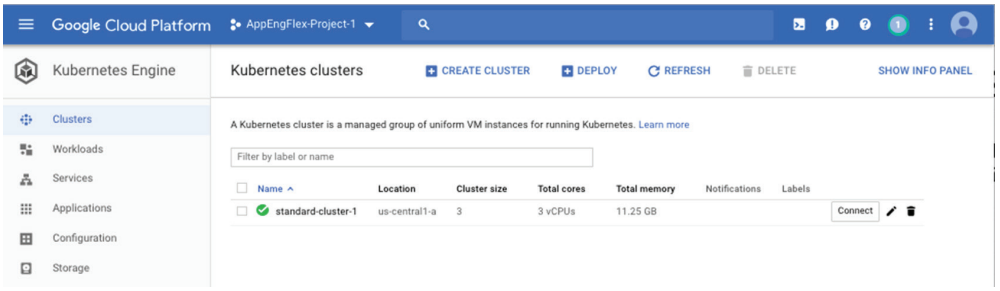
In Figure 8.2, Kubernetes Engine has been *"pinned"* so it is displayed at the top. You can pin any service in the navigation menu by mousing over the product and clicking the pin icon that appears, as in Figure 8.3. In that figure, Compute Engine and Kubernetes Engine are already pinned, and Cloud Functions can be pinned by clicking the gray pin icon.

FIGURE 8.3 Pinning a service to the top of the navigation menu



After clicking Kubernetes Engine in the navigation menu, you will see a list of running clusters, as in Figure 8.4, which shows a single cluster called `standard-cluster-1`.

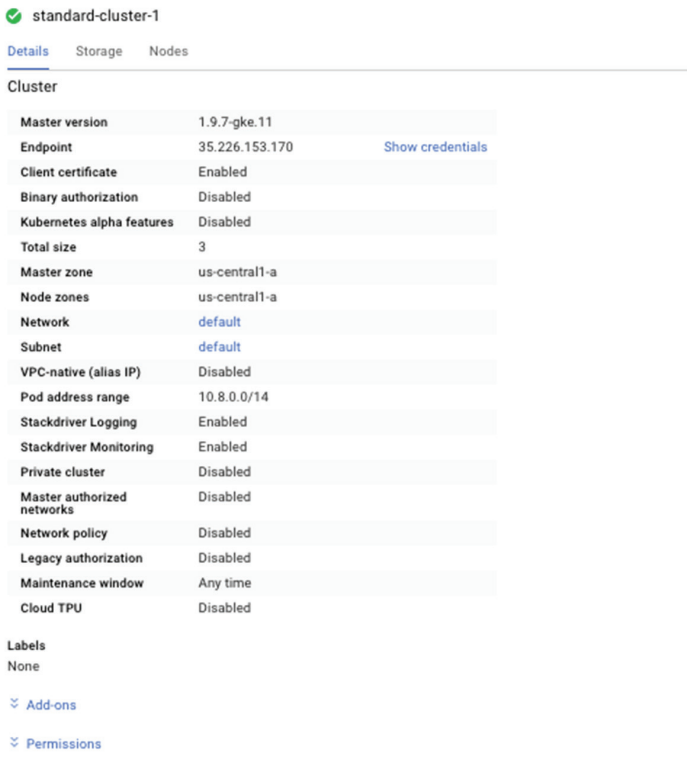
FIGURE 8.4 Example list of clusters in Kubernetes Engine



Mouse over the name of the cluster to highlight it, as in Figure 8.5, and click the name to display details of the cluster, as in Figure 8.6.

FIGURE 8.5 Click the name of a cluster to display its details.

FIGURE 8.6 The first part of the cluster Details page describes the configuration of the cluster.



Clicking the Add-ons and Permissions links displays information like that shown in Figure 8.7. The Add-ons section displays the status of optional add-on features of a cluster. The Permissions section shows which GCP service APIs are enabled for the cluster.

FIGURE 8.7 Add-on and permission details for a cluster

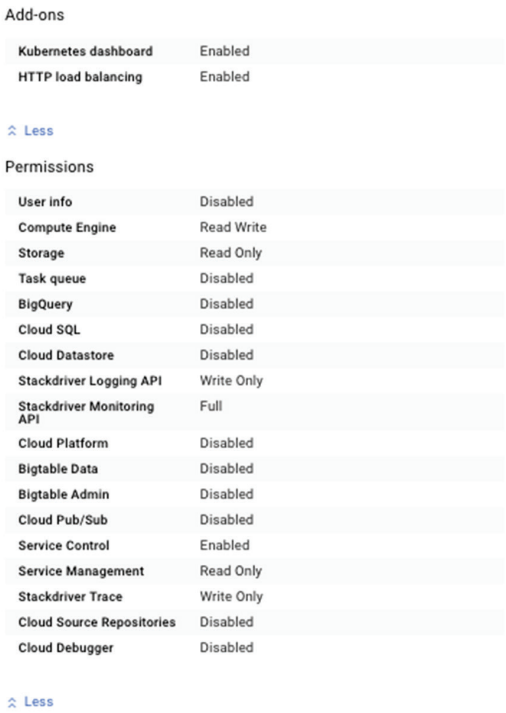


Figure 8.8 shows example details of node pools, which are separate instance groups running in a Kubernetes clusters. The details in this section include the node image running on the nodes, the machine type, the total number of vCPUs (listed as Total Cores), the disk type, and whether the nodes are preemptible.

Below the name of the cluster is a horizontal list of three options: Details, Storage, and Nodes. So far, we have described the contents of the Details page. Click Storage to display information such as in Figure 8.9, which displays persistent volumes and the storage classes used by the cluster.

This cluster does not have persistent volumes but uses standard storage. Persistent volumes are durable disks that are managed by Kubernetes and implemented using Compute Engine persistent disks. A storage class is a type of storage with a set of policies specifying quality of service, backup policy, and a provisioner (which is a service that implements the storage).

FIGURE 8.8 Details about node pools in the cluster

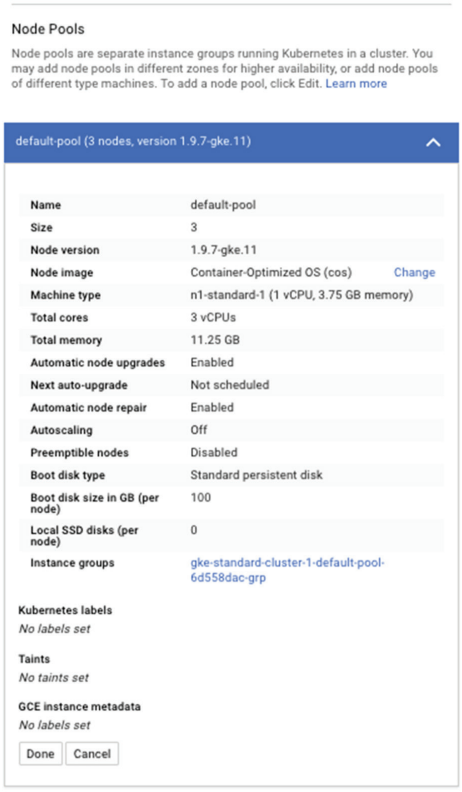
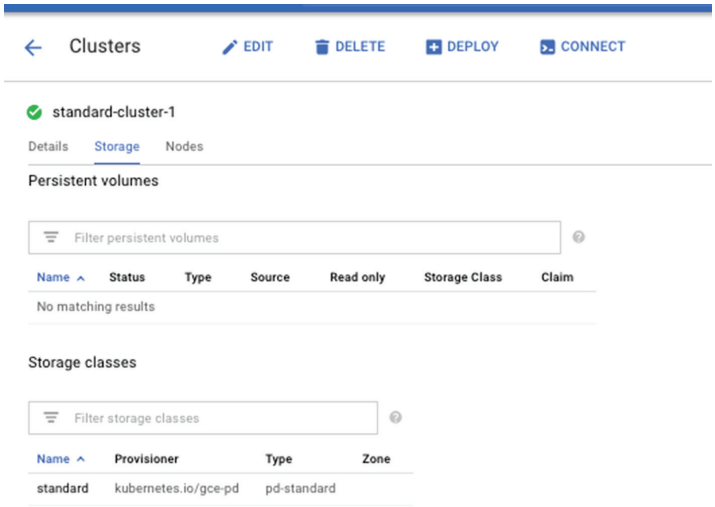
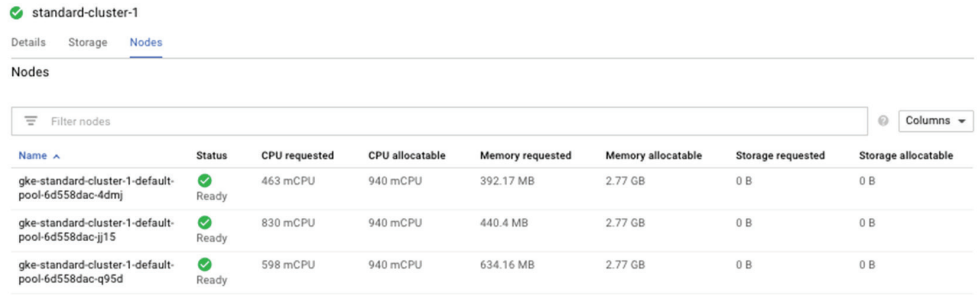


FIGURE 8.9 Storage information about a cluster



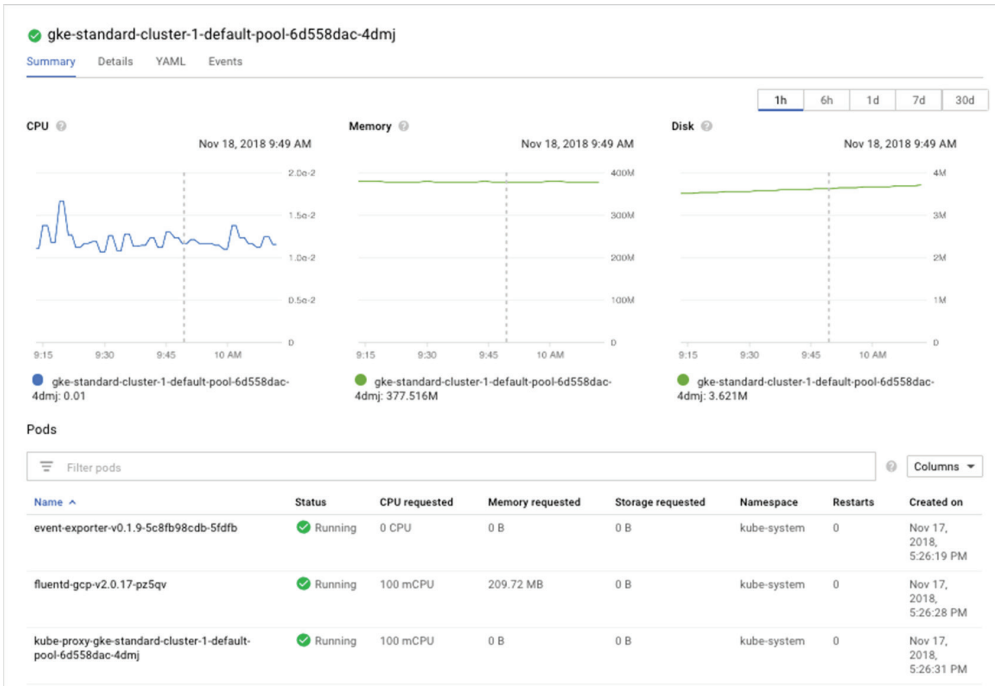
Under the Nodes option of the cluster status menu, you can see a list of nodes or VMs running in the cluster, as shown in Figure 8.10. The nodes list shows basic configuration information.

FIGURE 8.10 Listing of nodes in the cluster



Click the name of one of the nodes to see detailed status information such as in Figure 8.11. The node details include CPU utilization, memory consumption, and disk IO. There is also a list of pods running on the node.

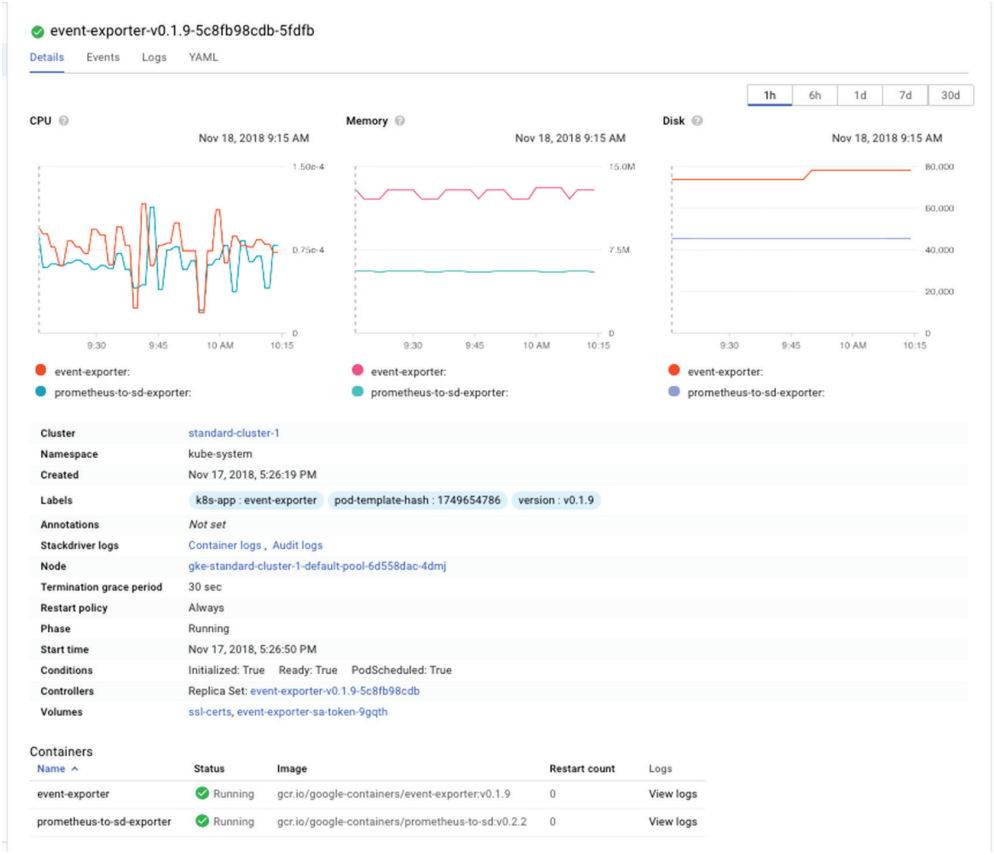
FIGURE 8.11 Example details of a node running in a Kubernetes cluster



Click the name of a pod to see its details. The pod display is similar to the node display with CPU, memory, and disk statistics. Configuration details include when the pod was created, labels assigned, links to logs, and status (which is shown as Running in Figure 8.12).

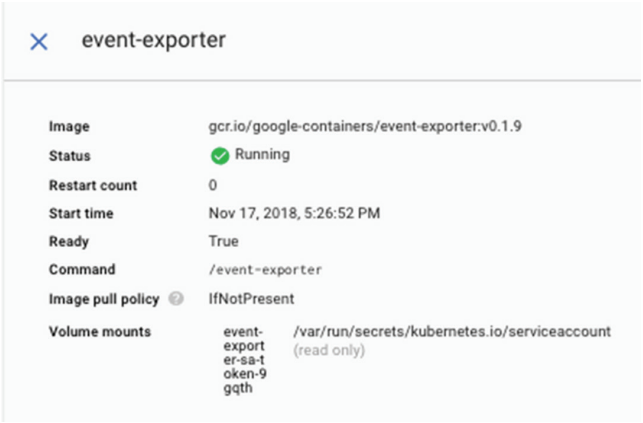
Other possible statuses are Pending, which indicates the pod is downloading images; Succeeded, which indicates the pod terminated successfully; Failed, which indicates at least one container failed; and Unknown, which means the master cannot reach the node and status cannot be determined.

FIGURE 8.12 Pod status display, with status as Running



At the bottom of the pod display is a list of containers running. Click the name of a container to see its details. Figure 8.13 shows the details of the container named `event-exporter`. Information includes the status, the start time, the command that is running, and the volumes mounted.

FIGURE 8.13 Details of a container running in a pod



Using Cloud Console, you can list all clusters and view details of their configuration and status. You can then drill down into each node, pod, and container to view their details.

Viewing the Status of Kubernetes Clusters Using Cloud SDK and Cloud Shell

You can also use the command line to view the status of a cluster. The `gcloud container cluster` list command is used to show those details.

To list the names and basic information of all clusters, use this command:

```
gcloud container clusters list
```

This produces the output shown in Figure 8.14.

FIGURE 8.14 Example output from the `gcloud container clusters` list command

Why Don't Commands Start with `gcloud kubernetes`?

`gcloud` commands start with the word `gcloud` followed by the name of the service, for example, `gcloud compute` for Compute Engine commands and `gcloud sql` for Cloud SQL commands. You might expect the Kubernetes Engine commands to start with `gcloud kubernetes`, but the service was originally called Google Container Engine. In November 2017, Google renamed the service Kubernetes Engine, but the `gcloud` commands remained the same.

To view the details of a cluster, use the `gcloud container clusters describe` command. You will need to pass in the name of a zone or region using the `--zone` or `--region` parameter. For example, to describe a cluster named `standard-cluster-1` located in the `us-central1-a` zone, you would use this command:

```
gcloud container clusters describe --zone us-central1-a standard-cluster-1
```

This will display details like those shown in Figure 8.15 and Figure 8.16. Note that the `describe` command also displays authentication information such as client certificate, username, and password. That information is not shown in the figures.

FIGURE 8.15 Part 1 of the information displayed by the `gcloud container clusters describe` command

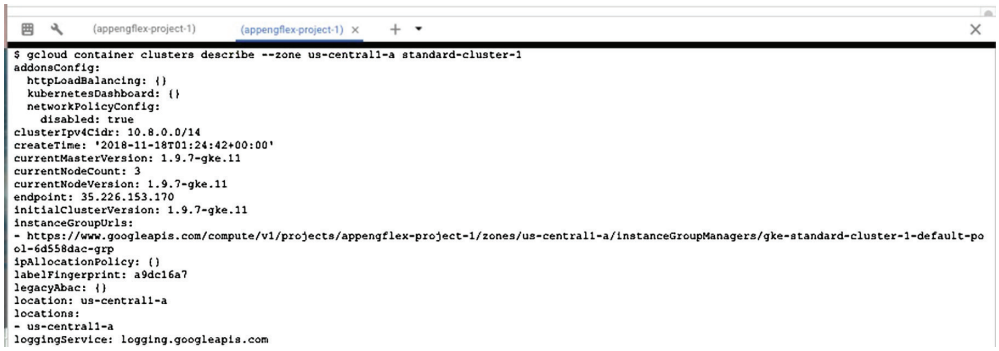


FIGURE 8.16 Part 2 of the information displayed by the `gcloud container clusters describe` command

```
masterAuthorizedNetworksConfig: {}
monitoringService: monitoring.googleapis.com
name: standard-cluster-1
network: default
networkConfig:
  network: projects/appengflex-project-1/global/networks/default
  subnetwork: projects/appengflex-project-1/regions/us-central1/subnetworks/default
networkPolicy: {}
nodeConfig:
  diskSizeGb: 100
  diskType: pd-standard
  imageType: COS
  machineType: n1-standard-1
  oauthScopes:
  - https://www.googleapis.com/auth/compute
  - https://www.googleapis.com/auth/devstorage.read_only
  - https://www.googleapis.com/auth/logging.write
  - https://www.googleapis.com/auth/monitoring
  - https://www.googleapis.com/auth/servicecontrol
  - https://www.googleapis.com/auth/service.management.readonly
  - https://www.googleapis.com/auth/trace.append
  serviceAccount: default
nodeIpv4CidrSize: 24
nodePools:
- autoscaling: {}
  config:
    diskSizeGb: 100
    diskType: pd-standard
    imageType: COS
    machineType: n1-standard-1
    oauthScopes:
    - https://www.googleapis.com/auth/compute
    - https://www.googleapis.com/auth/devstorage.read_only
    - https://www.googleapis.com/auth/logging.write
    - https://www.googleapis.com/auth/monitoring
    - https://www.googleapis.com/auth/servicecontrol
    - https://www.googleapis.com/auth/service.management.readonly
    - https://www.googleapis.com/auth/trace.append
    serviceAccount: default
  initialNodeCount: 3
  instanceGroupUrls:
  - https://www.googleapis.com/compute/v1/projects/appengflex-project-1/zones/us-central1-a/instanceGroupManagers/gke-standard-cluster-1-default-
pool-6d558dac-grp
  management:
    autoRepair: true
    autoUpgrade: true
  name: default-pool
  selfLink: https://container.googleapis.com/v1/projects/appengflex-project-1/zones/us-central1-a/clusters/standard-cluster-1/nodePools/default-p
ool
  status: RUNNING
  version: 1.9.7-gke.11
selfLink: https://container.googleapis.com/v1/projects/appengflex-project-1/zones/us-central1-a/clusters/standard-cluster-1
servicesIpv4Cidr: 10.11.240.0/20
status: RUNNING
subnetwork: default
zone: us-central1-a
```

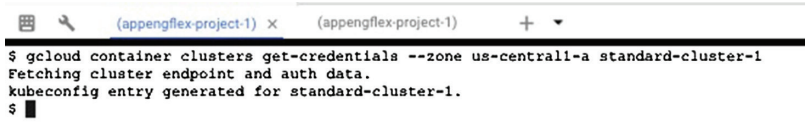
To list information about nodes and pods, use the `kubectl` command.

First, you need to ensure you have a properly configured `kubeconfig` file, which contains information on how to communicate with the cluster API. Run the command `gcloud container clusters get-credentials` with the name of a zone or region and the name of a cluster. Here's an example:

```
gcloud container clusters get-credentials --zone us-central1-a standard-cluster-1
```

This will configure the `kubeconfig` file on a cluster named `standard-cluster-1` in the `use-central1-a` zone. Figure 8.17 shows an example output of that command, which includes the status of fetching and setting authentication data.

FIGURE 8.17 Example output of the `get-credentials` command

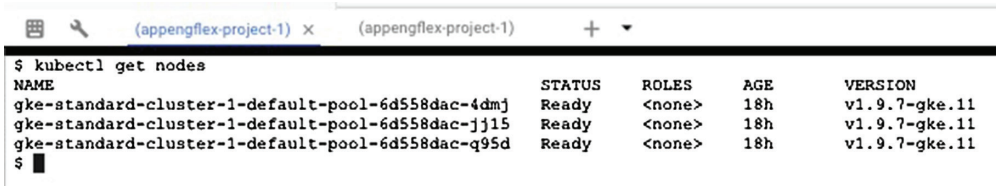


You can list the nodes in a cluster using the following:

```
kubectl get nodes
```

This produces output such as in Figure 8.18, which shows the status of three nodes.

FIGURE 8.18 Example output of the `kubectl get nodes` command

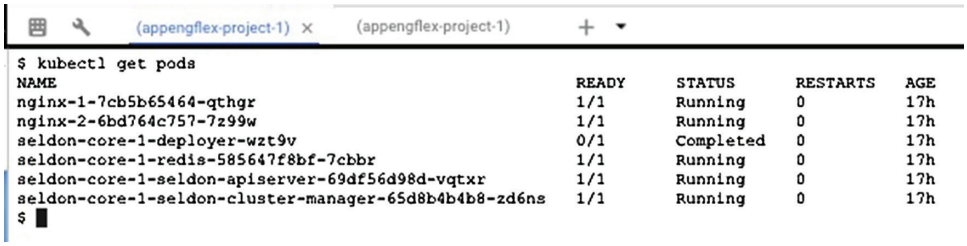


Similarly, to list pods, use the following command:

```
kubectl get pods
```

This produces output such as in Figure 8.19, which lists pods and their status.

FIGURE 8.19 Example output of the `kubectl get pods` command

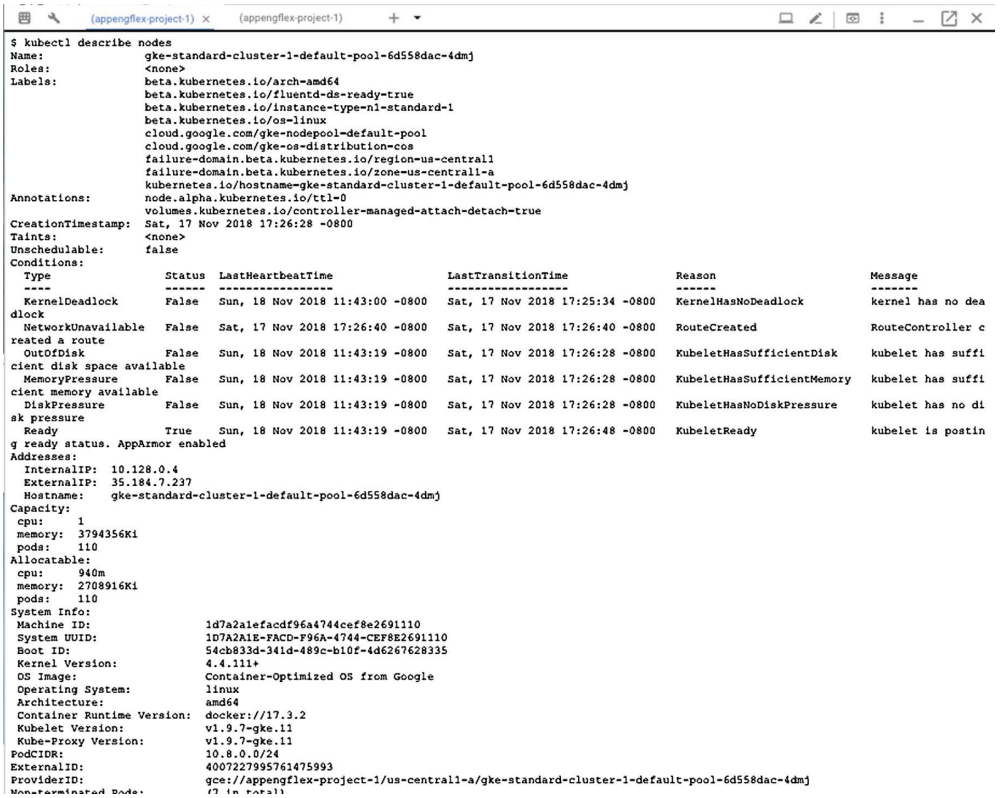


For more details about nodes and pods, use these commands:

```
kubectl describe nodes
kubectl describe pods
```

Figures 8.20 and 8.21 show partial listings of the results. Note that the `kubectl describe pods` command also includes information about containers, such as name, labels, conditions, network addresses, and system information.

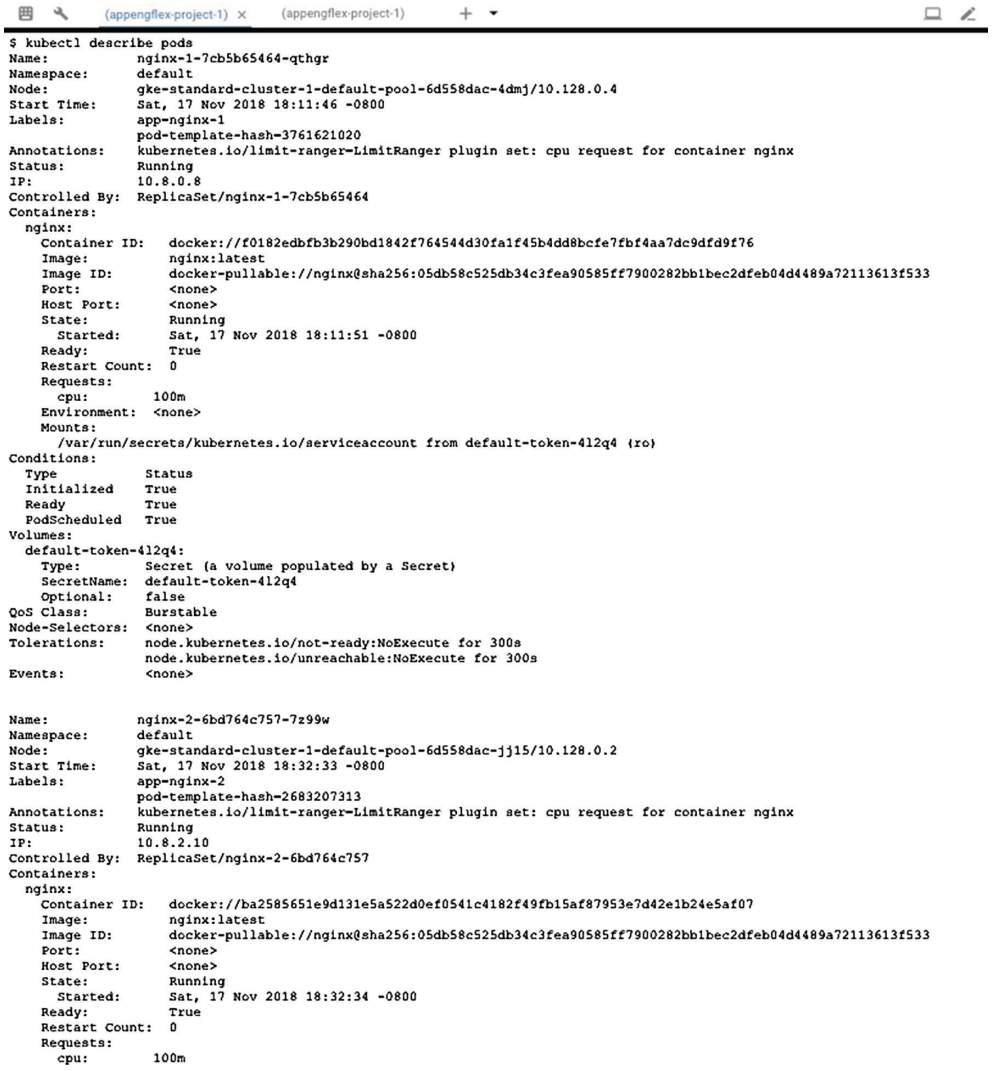
FIGURE 8.20 Partial listing of the details shown by the `kubectl describe nodes` command



```
(appengflex-project-1) ×   (appengflex-project-1)

$ kubectl describe nodes
Name:               gke-standard-cluster-1-default-pool-6d558dac-4dmj
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/fluentd-ds-ready=true
                    beta.kubernetes.io/instance-type=n1-standard-1
                    beta.kubernetes.io/os=linux
                    cloud.google.com/gke-nodepool=default-pool
                    cloud.google.com/gke-os-distribution=cos
                    failure-domain.beta.kubernetes.io/region=us-central1
                    failure-domain.beta.kubernetes.io/zone=us-central1-a
                    kubernetes.io/hostname=gke-standard-cluster-1-default-pool-6d558dac-4dmj
Annotations:        node.alpha.kubernetes.io/ttl=0
                    volumes.kubernetes.io/controller-managed-attach-detach=true
CreationTimestamp:  Sat, 17 Nov 2018 17:26:28 -0800
Taints:             <none>
Unschedulable:      false
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----                 ------  -----------------                 ------------------                ------                       -------
  KernelDeadlock       False   Sun, 18 Nov 2018 11:43:00 -0800   Sat, 17 Nov 2018 17:25:34 -0800   KernelHasNoDeadlock          kernel has no dea
dlock
  NetworkUnavailable   False   Sat, 17 Nov 2018 17:26:40 -0800   Sat, 17 Nov 2018 17:26:40 -0800   RouteCreated                 RouteController c
reated a route
  OutOfDisk            False   Sun, 18 Nov 2018 11:43:19 -0800   Sat, 17 Nov 2018 17:26:28 -0800   KubeletHasSufficientDisk     kubelet has suffi
cient disk space available
  MemoryPressure       False   Sun, 18 Nov 2018 11:43:19 -0800   Sat, 17 Nov 2018 17:26:28 -0800   KubeletHasSufficientMemory   kubelet has suffi
cient memory available
  DiskPressure         False   Sun, 18 Nov 2018 11:43:19 -0800   Sat, 17 Nov 2018 17:26:28 -0800   KubeletHasNoDiskPressure     kubelet has no di
sk pressure
  Ready                True    Sun, 18 Nov 2018 11:43:19 -0800   Sat, 17 Nov 2018 17:26:48 -0800   KubeletReady                 kubelet is postin
g ready status. AppArmor enabled
Addresses:
  InternalIP:  10.128.0.4
  ExternalIP:  35.184.7.237
  Hostname:    gke-standard-cluster-1-default-pool-6d558dac-4dmj
Capacity:
 cpu:     1
 memory:  3794356Ki
 pods:    110
Allocatable:
 cpu:     940m
 memory:  2708916Ki
 pods:    110
System Info:
 Machine ID:                 1d7a2a1efacdf96a4744cef8e2691110
 System UUID:                1D7A2A1E-FACD-F96A-4744-CEF8E2691110
 Boot ID:                    54cb833d-341d-489c-b10f-4d6267628335
 Kernel Version:             4.4.111+
 OS Image:                   Container-Optimized OS from Google
 Operating System:           linux
 Architecture:               amd64
 Container Runtime Version:  docker://17.3.2
 Kubelet Version:            v1.9.7-gke.11
 Kube-Proxy Version:         v1.9.7-gke.11
PodCIDR:                     10.8.0.0/24
ExternalID:                  4007227995761475993
ProviderID:                  gce://appengflex-project-1/us-central1-a/gke-standard-cluster-1-default-pool-6d558dac-4dmj
Non-terminated Pods:         (7 in total)
```

FIGURE 8.21 Partial listing of the details shown by the `kubectl describe pods` command



To view the status of clusters from the command line, use the `gcloud container` commands, but to get information about Kubernetes managed objects, like nodes, pods, and containers, use the `kubectl` command.

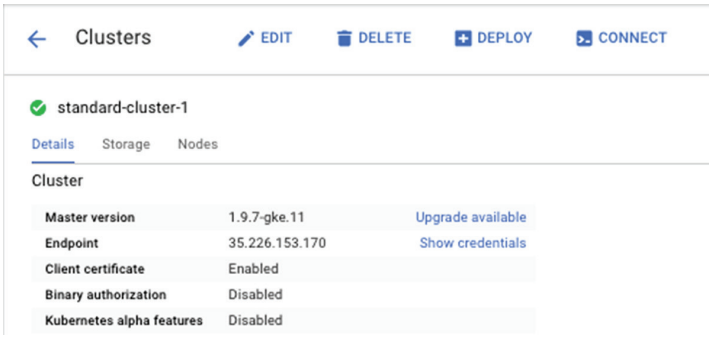
Adding, Modifying, and Removing Nodes

You can add, modify, and remove nodes from a cluster using either Cloud Console or Cloud SDK in your local environment, on a GCP virtual machine, or in Cloud Shell.

Adding, Modifying, and Removing Nodes with Cloud Console

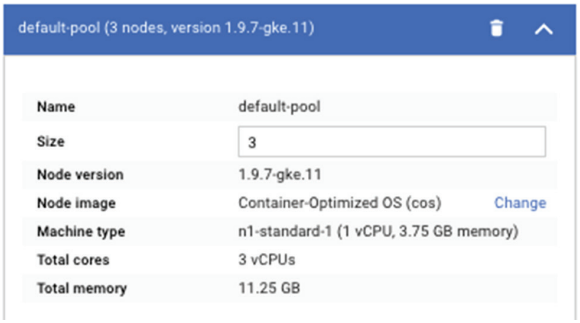
From Cloud Console, navigate to the Kubernetes Engine page and display a list of clusters. Click the name of a cluster to display its details, as in Figure 8.22. Note the Edit option near the top of the screen. Click that to open an Edit form.

FIGURE 8.22 Details of a cluster in Cloud Console



Scroll down to the Node Pools section, which lists the name, size, node image, machine type, and other information about the cluster. The size parameter is optional. In the example shown in Figure 8.23, the cluster has three nodes.

FIGURE 8.23 Details of a node pool in Cloud Console



To add nodes, increase the size to the number of nodes you would like. To remove nodes, decrease the size to the number of nodes you'd like to have.

Adding, Modifying, and Removing with Cloud SDK and Cloud Shell

The command to add or modify nodes is `gcloud container clusters resize`. The command takes three parameters, as shown here:

- `cluster name`
- `node pool name`
- `cluster size`

For example, assume you have a cluster named `standard-cluster-1` running a node pool called `default-pool`. To increase the size of the cluster from 3 to 5, use this command:

```
gcloud container clusters resize standard-cluster-1 --node-pool default-pool
--size 5 --region=us-central1
```

Once a cluster has been created, you can modify it using the `gcloud container clusters update` command. For example, to enable autoscaling, use the `update` command to specify the maximum and minimum number of nodes. The command to update a cluster named `standard-cluster-1` running in a node pool called `default-pool` is as follows:

```
gcloud container clusters update standard-cluster-1 --enable-autoscaling
--min-nodes 1 \
--max-nodes 5 --zone us-central1-a --node-pool default-pool
```

Real World Scenario

Keeping Up with Demand with Autoscaling

Often it is difficult to predict demand on a service. Even if there are regular patterns, such as large batch jobs run during nonbusiness hours, there can be variation in when those peak loads run. Rather than keep manually changing the number of vCPUs in a cluster, enable autoscaling to automatically add or remove nodes as needed based on demand. Autoscaling can be enabled when creating clusters with either Cloud Console or `gcloud`. This approach is more resilient to unexpected spikes and shifts in long-term patterns of peak use. It will also help optimize the cost of your cluster by not running too many servers when not needed. It will also help maintain performance by having sufficient nodes to meet demand.

Adding, Modifying, and Removing Pods

You can add, modify, and remove pods from a cluster using either Cloud Console or Cloud SDK in your local environment, on a GCP VM, or in Cloud Shell.

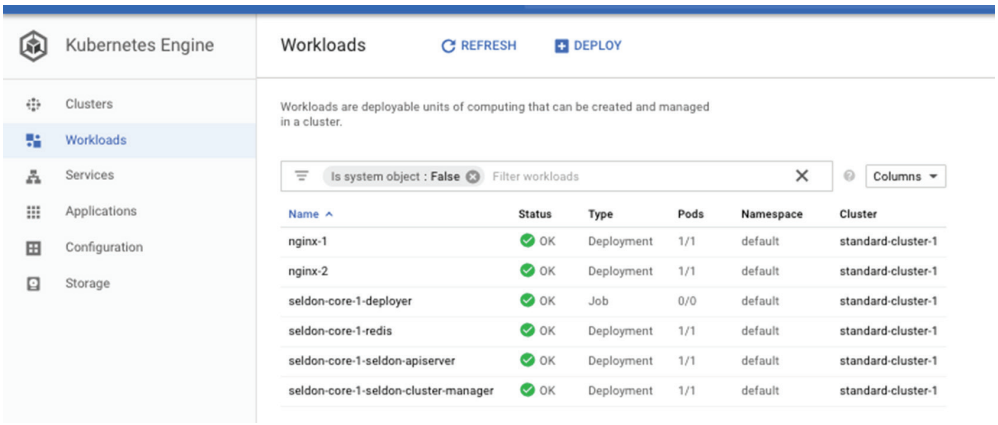
It is considered a best practice to not manipulate pods directly. Kubernetes will maintain the number of pods specified for a deployment. If you would like to change the number of pods, you should change the deployment configuration.

Adding, Modifying, and Removing Pods with Cloud Console

Pods are managed through deployments. A deployment includes a configuration parameter called *replicas*, which are the number of pods running the application specified in the deployment. This section describes how to use Cloud Console to change the number of replicas, which will in turn change the number of pods.

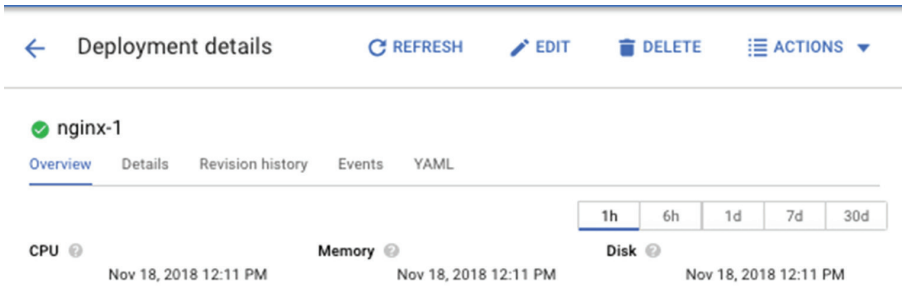
From Cloud Console, select the Workloads options from the navigation menu on the left. This displays a list of deployments, as in Figure 8.24.

FIGURE 8.24 List of deployments in a cluster



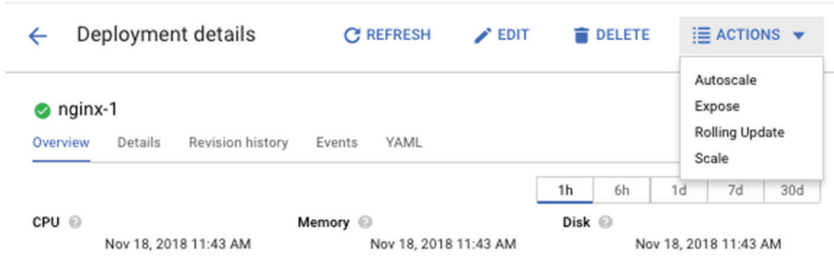
Click the name of the deployment you want to modify; a form is displayed with details such as in Figure 8.25. Note the Actions option in the top horizontal menu.

FIGURE 8.25 Multiple forms contain details of a deployment and include a menu of actions you can perform on the deployment.



Click Actions to list the options, which are Autoscale, Expose, Rolling Update, and Scale, as shown in Figure 8.26.

FIGURE 8.26 List of actions available for deployments

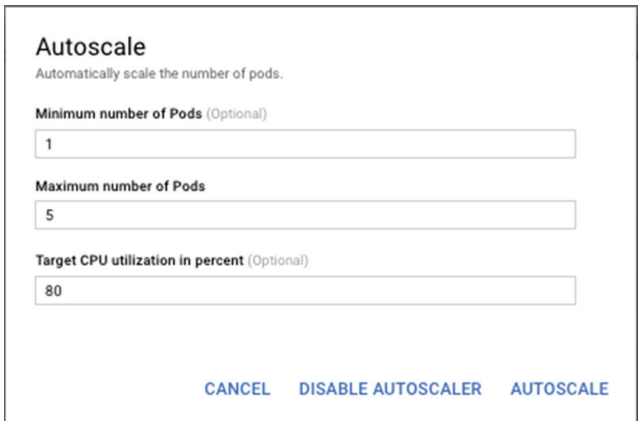


Select Scale to display a dialog that allows you to set a new size for the workload, as shown in Figure 8.27. In this example, the number of replicas has been changed to 2.

FIGURE 8.27 Set the number of replicas for a deployment.

You can also have Kubernetes automatically add and remove replicas (and pods) depending on need by specifying autoscaling. Choose Autoscaling from the menu to display the form shown in Figure 8.28. You can specify a minimum and maximum number of replicas to run here.

FIGURE 8.28 Enable autoscaling to automatically add and remove replicas as needed depending on load.



The Action menu also provides options to expose a service on a port, as shown in Figure 8.29, and to specify parameters to control rolling updates to deployed code, as shown in Figure 8.30. The parameters include the minimum seconds to wait before considering the pod updated, the maximum number of pods above target size allowed, and the maximum number of unavailable pods.

FIGURE 8.29 Form to expose services running on pods

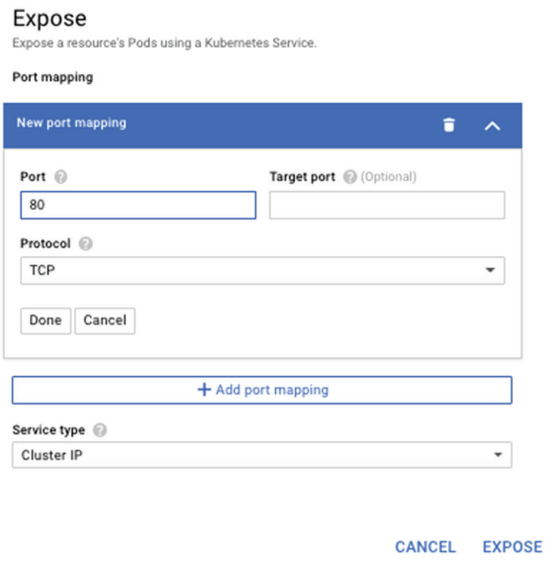
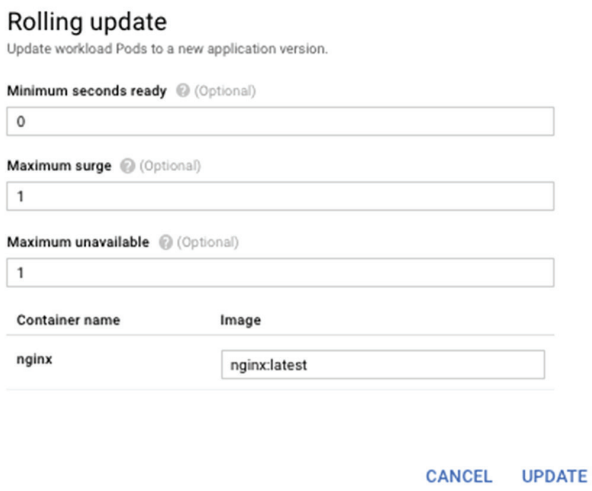


FIGURE 8.30 Form to specify parameters for rolling updates of code running in pods



Adding, Modifying, and Removing Pods with Cloud SDK and Cloud Shell

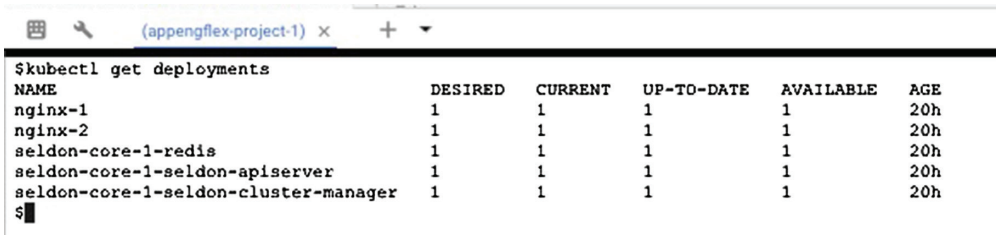
Working with pods in Cloud SDK and Cloud Shell is done by working with deployments; deployments were explained earlier in the section "Adding, Modifying, and Removing Pods with Cloud Console." You can use the `kubectl` command to work with deployments.

To list deployments, use the following command:

```
kubectl get deployments
```

This will produce a list of deployments such as in Figure 8.31.

FIGURE 8.31 A list of deployments on the command line



To add and remove pods, change the configuration of deployments using the `kubectl scale deployment` command. For this command, you have to specify the deployment name

and number of replicas. For example, to set the number of replicas to 5 for a deployment named nginx-1, use this:

```
kubectl scale deployment nginx-1 --replicas 5
```

To have Kubernetes manage the number of pods based on load, use the `autoscale` command. The following command will add or remove pods as needed to meet demand based on CPU utilization. If CPU usage exceeds 80 percent, up to 10 additional pods or replicas will be added. The deployment will always have at least one pod or replica.

```
kubectl autoscale deployment nginx-1 --max 10 --min 1 --cpu-percent 80
```

To remove a deployment, use the `delete deployment` command like so:

```
kubectl delete deployment nginx-1
```

Adding, Modifying, and Removing Services

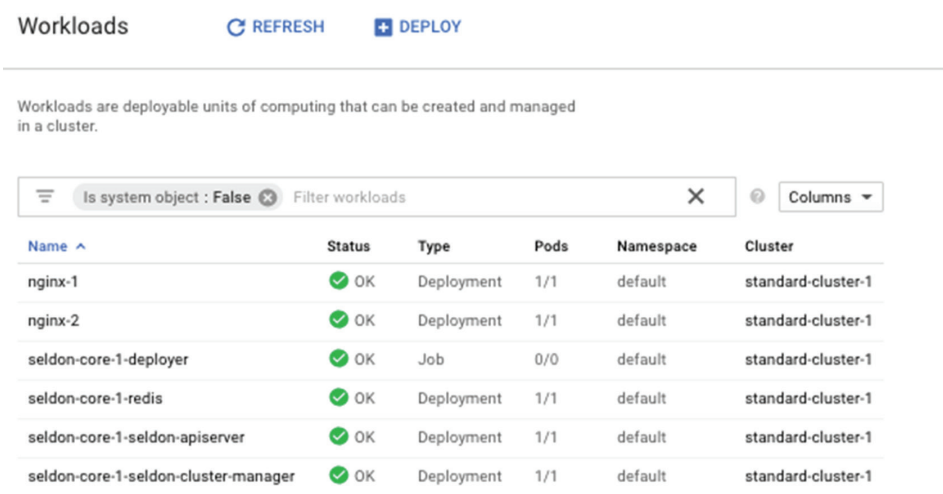
You can add, modify, and remove services from a cluster using either Cloud Console or Cloud SDK in your local environment, on a GCP VM, or in Cloud Shell.

A service is an abstraction that groups a set of pods as a single resource.

Adding, Modifying, and Removing Services with Cloud Console

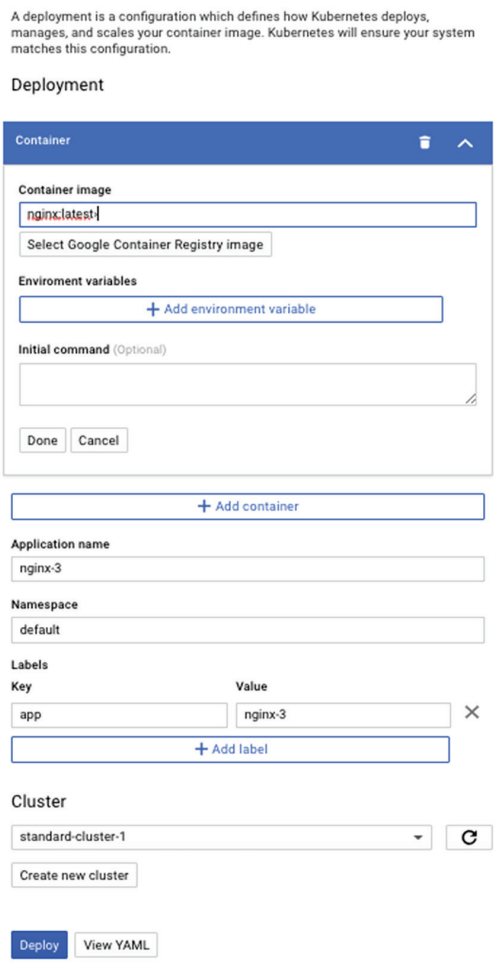
Services are added through deployments. In Cloud Console, select the Workloads option from the navigation menu to display a list of deployments, as in Figure 8.32. Note the Deploy option in the horizontal menu at the top of the page.

FIGURE 8.32 List of deployments along with a Deploy command to create new services



Click Deploy to show the deployment form, as in Figure 8.33.

FIGURE 8.33 Form to specify a new deployment for a service



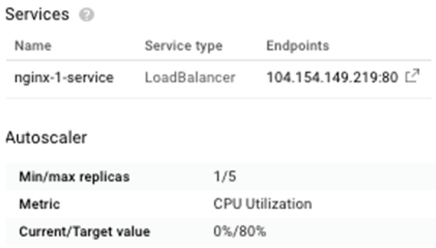
In the Container Image parameter, you can specify the name of an image or select one from the Google Container Repository. To specify a name directly, specify a path to the image using a URL such as this:

```
gcr.io/google-samples/hello-app:2.0
```

You can specify labels, the initial command to run, and a name for your application.

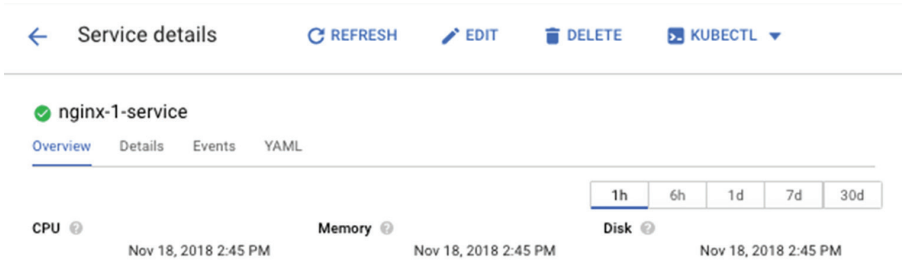
When you click the name of a deployment, like those listed earlier in Figure 8.32, you will see details of that deployment, including a list of services, like that shown in Figure 8.34.

FIGURE 8.34 Details of a service running in a deployment



Clicking the name of a service opens the Detail form of the service, which includes a Delete option in the horizontal menu, as shown in Figure 8.35.

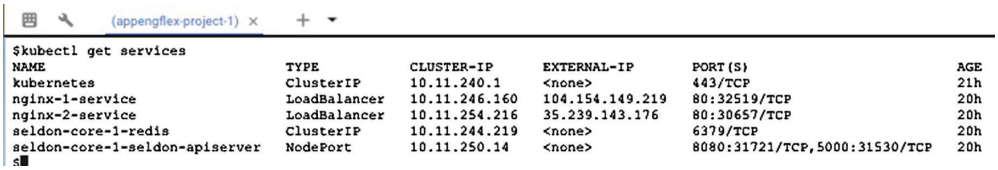
FIGURE 8.35 Navigate to the Service Details page to delete a service using the Delete option in the horizontal menu.



Adding, Modifying, and Removing Services with Cloud SDK and Cloud Shell

Use the `kubectl get services` command to list services. Figure 8.36 shows an example listing.

FIGURE 8.36 A list of services displayed by a `kubectl get services` command



To add a service, use the `kubectl run` command to start a service. For example, to add a service called `hello-server` using the sample application by the same name provided by Google, use the following command:

```
kubectl run hello-server --image=gcr.io/google/samples/hello-app:1.0 --port 8080
```

This command will download and start running the image found at the path `gcr.io/google-samples/` called `hello-app`, version 1. It will be accessible on port 8080. Services need to be exposed to be accessible to resources outside the cluster. This can be set using the `expose` command, as shown here:

```
kubectl expose deployment hello-server --type="LoadBalancer"
```

This command exposes the services by having a load balancer act as the endpoint for outside resources to contact the service.

To remove a service, use the `delete service` command, as shown here:

```
kubectl delete service hello-server
```

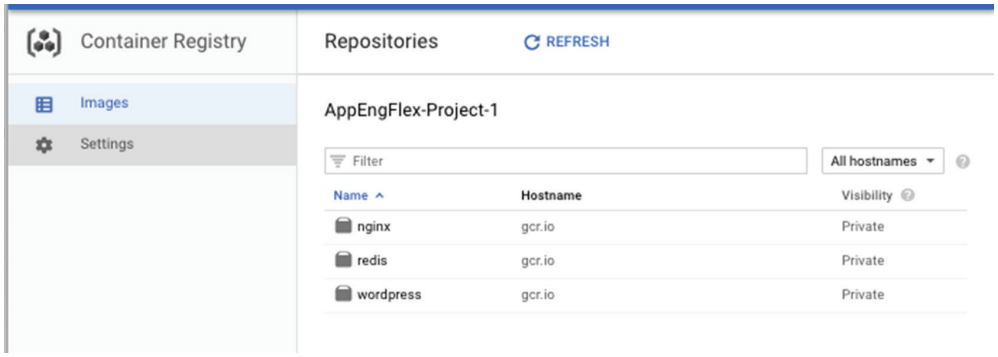
Viewing the Image Repository and Image Details

Container Registry is a GCP service for storing container images. Once you have created a registry and pushed images to it, you can view the contents of the registry and image details using Cloud Console and Cloud SDK and Cloud Shell.

Viewing the Image Repository and Image Details with Cloud Console

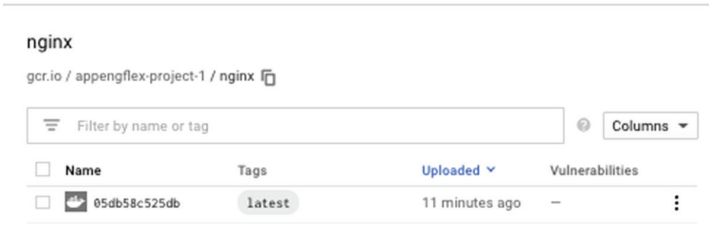
In Cloud Console, select Container Registry from the navigation menu to display the contents of a registry. Figure 8.37 shows an example listing with three images for Nginx, Redis, and WordPress.

FIGURE 8.37 A listing of images in a Container Registry



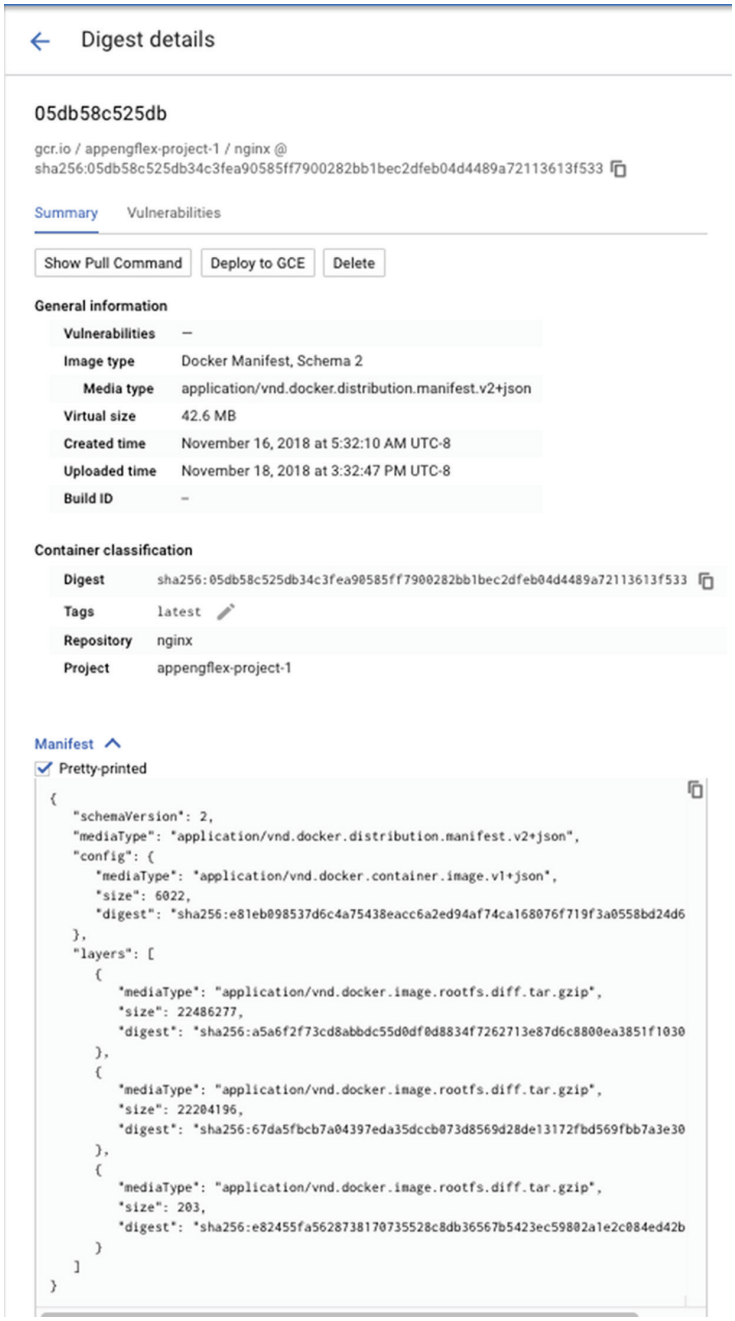
To see the details of an image, click the image name. For example, Figure 8.38 shows a listing for the Nginx image. This listing will list one entry for each version of the image. Since there is only one version of the image, there is only one listed.

FIGURE 8.38 A list of versions for an image



To see the details of that version, click the version name. This displays a listing such as in Figure 8.39, which includes the image type, size, and time created.

FIGURE 8.39 Details of a version of an image



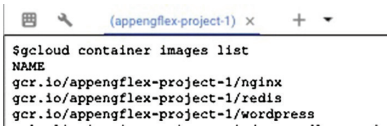
Viewing the Image Repository and Image Details with Cloud SDK and Cloud Shell

From the command line, you work with images in a registry using `gcloud container images` commands. For example, to list the contents of a registry, use this:

```
gcloud container images list
```

This command produces a list of images, such as in Figure 8.40. You can also list Google containers using `gcloud container images list --repository gcr.io/google-containers`.

FIGURE 8.40 List of images in a container repository



To view the details of an image, use the `describe` command and pass in the name of the image as an argument. For example, the following command:

```
gcloud container images describe gcr.io/appengflex-project-1/nginx
```

will produce an output list such as that shown in Figure 8.41. You can also describe a Google image with a command such as `gcloud container images describe gcr.io/google-containers/toolbox`.

FIGURE 8.41 A listing of image details produced by the `describe image` command

Kubernetes Engine makes use of container images stored in a Container Repository. The contents of the Container Repository can be viewed in summary and in detail using both Cloud Console and the command-line Cloud SDK, including in Cloud Shell.

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:

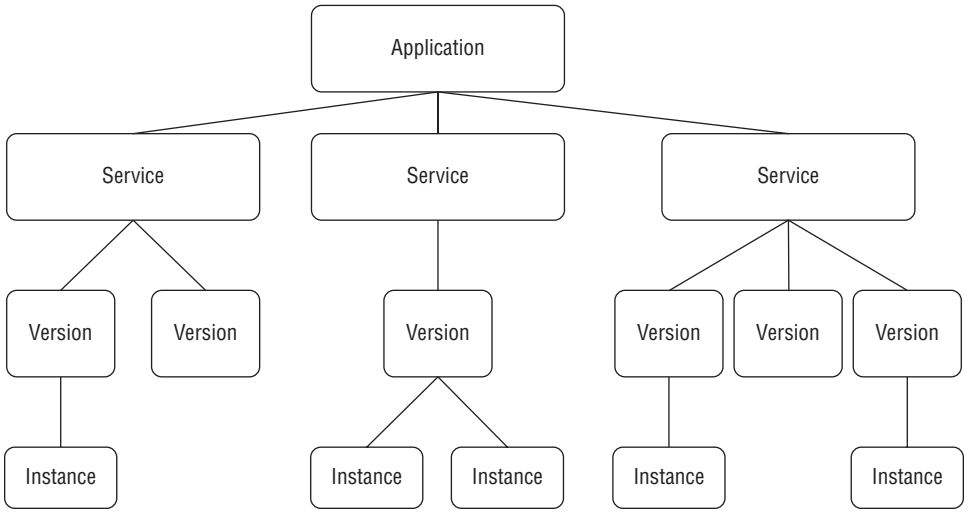
- Application
- Service
- Version
- Instance

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.

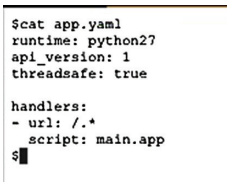
- `app.yaml`
- `main.py`
- `main_test.py`

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
```

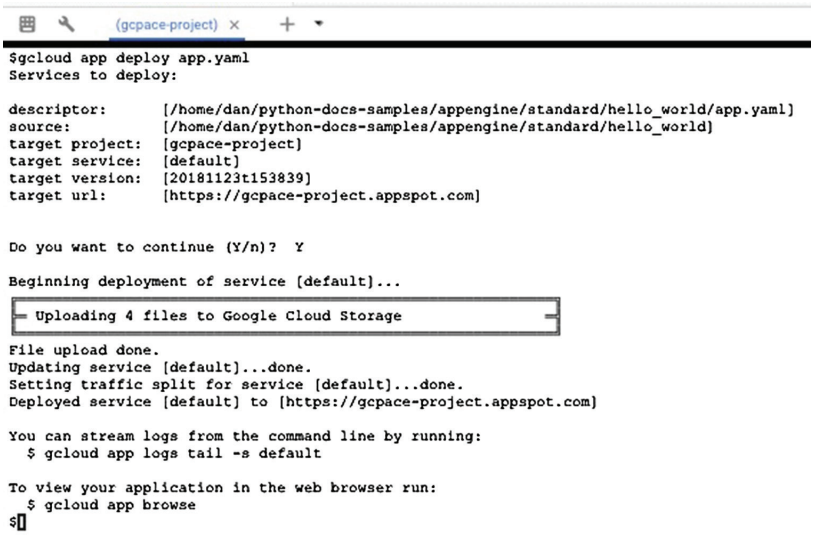
However, `app.yml` is the default, so if you are using that for the filename, you do not have to specify `app.yml` in the `deploy` command.

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

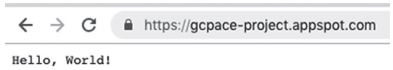
When you issue the `gcloud app deploy` command, you will see output such as in Figure 9.3.

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

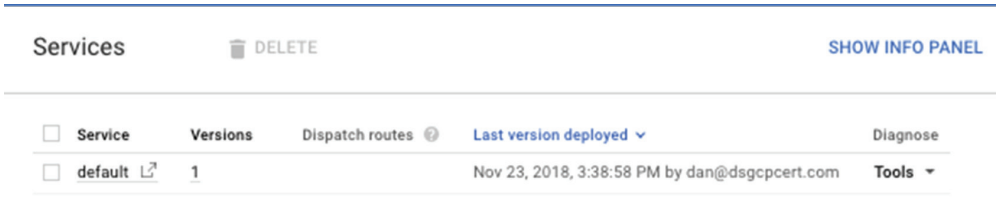


Figure 9.6 shows a list of versions. You can display this by selecting Versions from the left-panel menu.

FIGURE 9.6 A listing of versions in the App Engine console

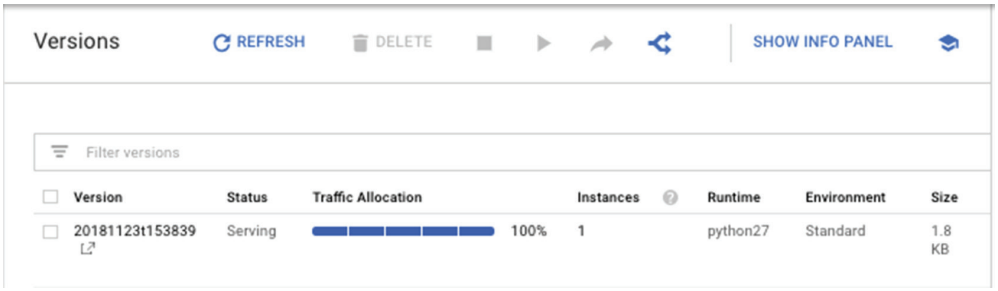
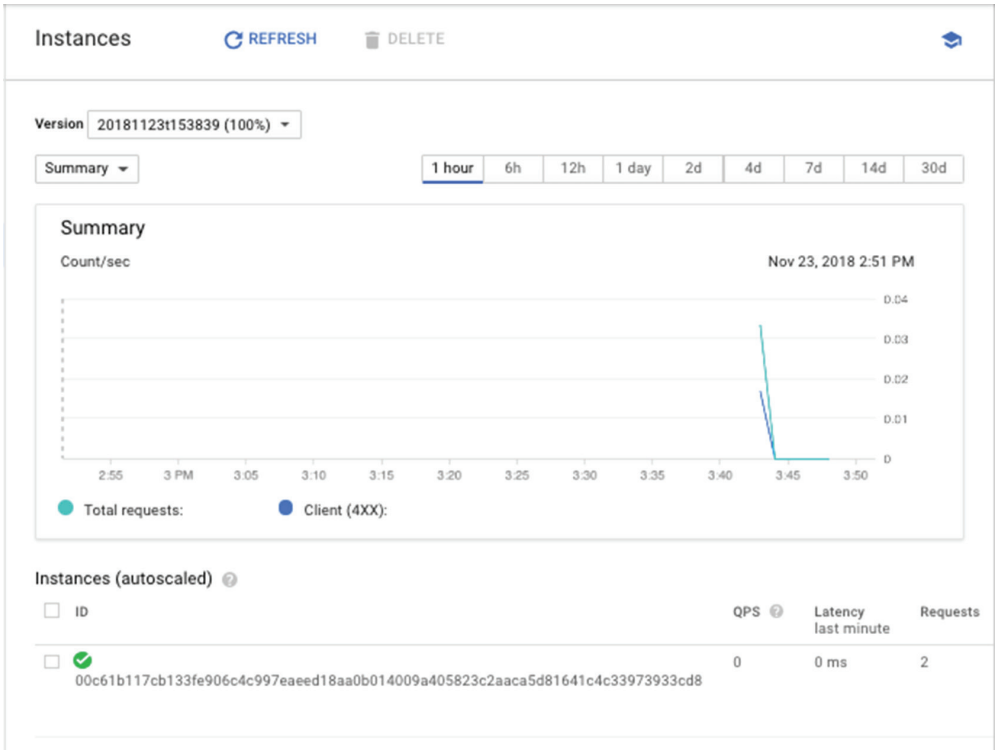


Figure 9.7 shows the instance performance details. You can display these details by selecting Instances from the left-panel menu. This information is useful for understanding the load on your application.

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.

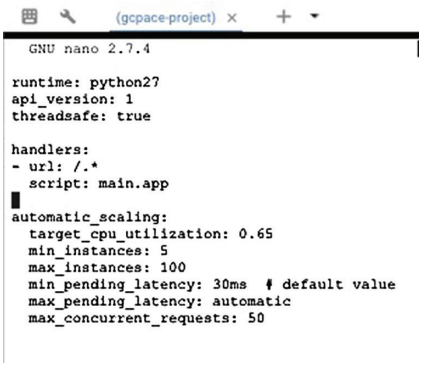
Target Throughput Utilization Specifies the maximum number of concurrent requests before additional instances are started. This is specified as a number between 0.5 and 0.95.

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.

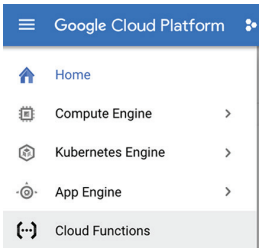
Cloud Functions Receiving Events from Cloud Storage

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

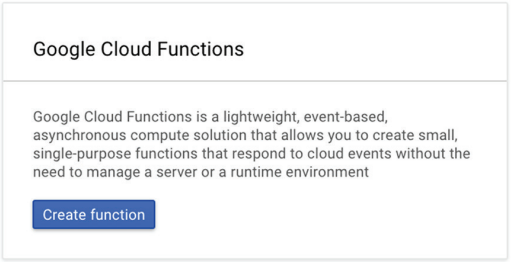
To create a function using Cloud Console, select the Cloud Function options from the vertical menu in the console, as in Figure 10.1.

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.

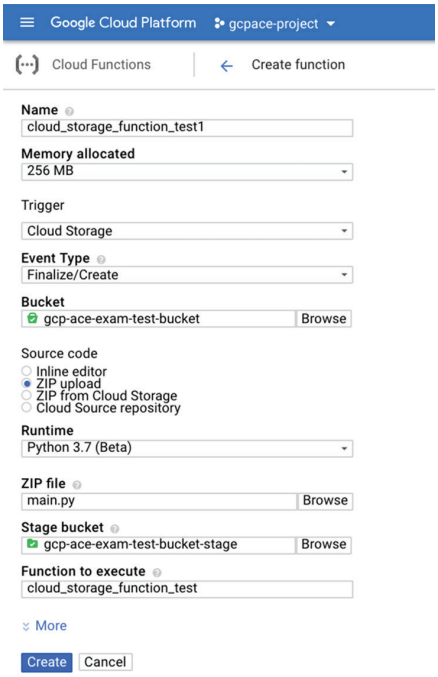
FIGURE 10.2 The prompt to create a new function in Cloud Console



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



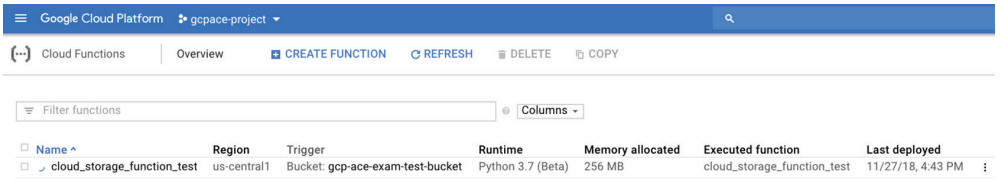
In the following example, we are uploading a file containing the function code. The contents of that file are as follows:

```
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`

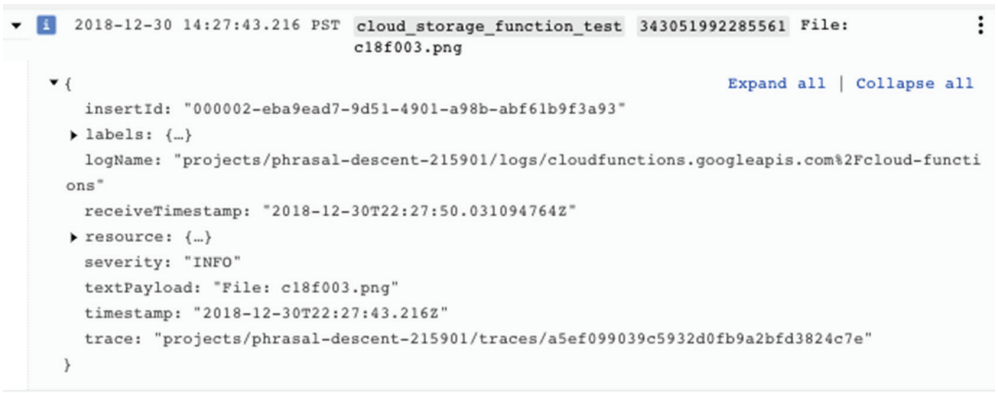
`finalize` is the term used to describe when a file is fully uploaded.

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`.

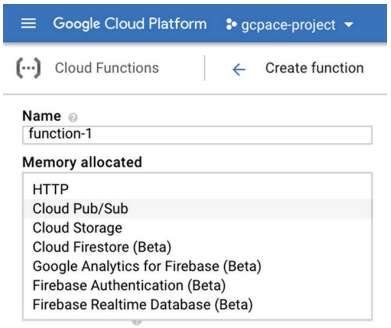
To create a function using Cloud Console, select the Cloud Function options from the vertical menu in the console. In the Cloud Functions console, you may be prompted to enable the Cloud Functions API if it is not already enabled. After the Cloud Functions API is enabled, you will have the option to 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.

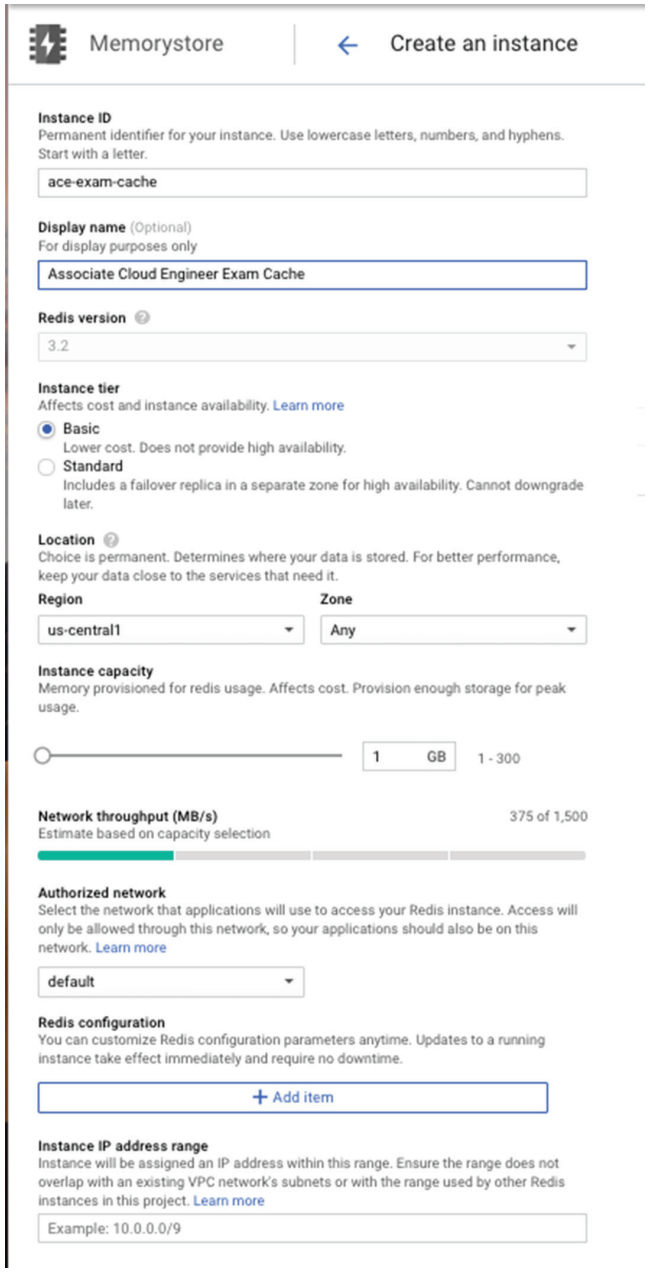
Caches are usually used with an application that cannot tolerate long latencies when retrieving data. For example, an application that has to read from a hard disk drive might have to wait 80 times longer than if the data were read from an in-memory cache. Application developers can use caches to store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

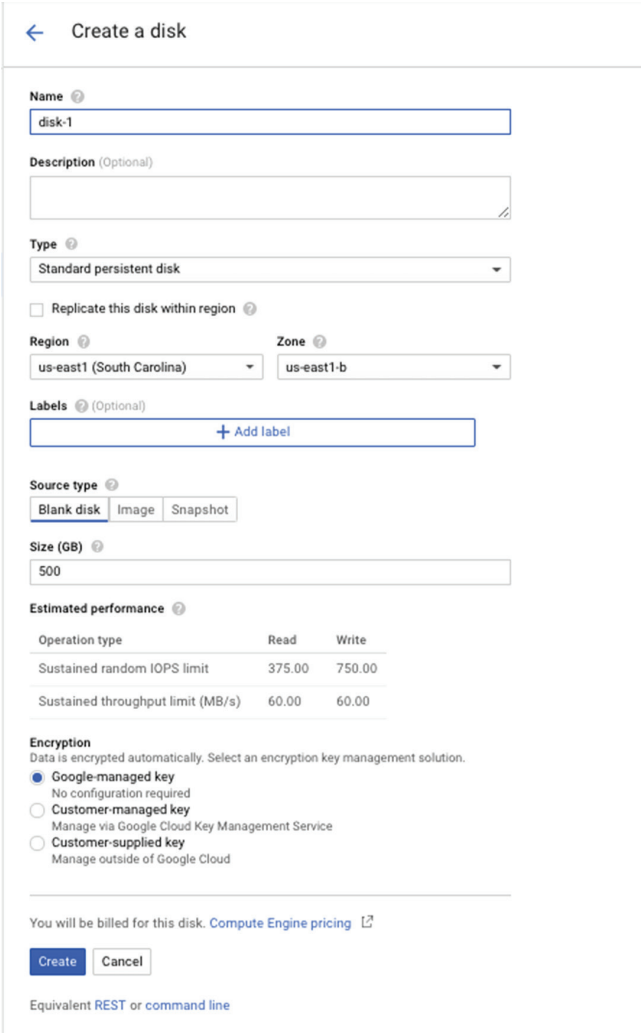
When planning your storage options, you should also consider whether you want your disks to be zonal or regional. Zonal disks store data across multiple physical drives in a single zone. If the zone becomes inaccessible, you will lose access to your disks. Alternatively, you

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.

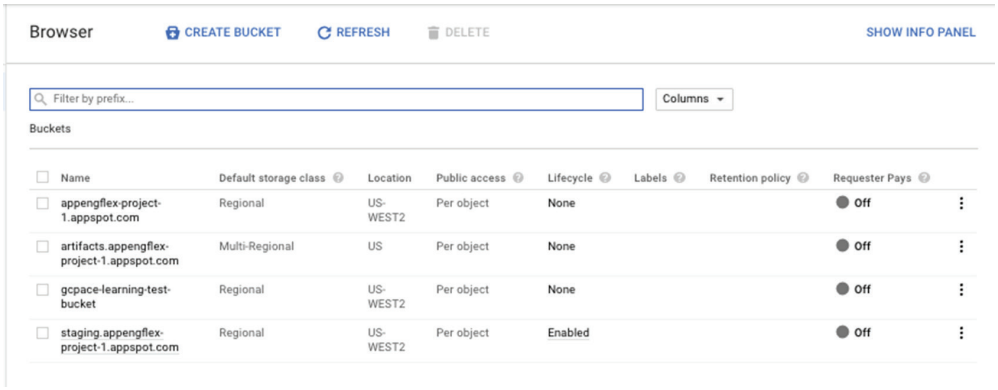
You can delete an object or change its storage class. Both unversioned and versioned objects can be deleted. If the live version of a file is deleted, then instead of actually deleting it, the object is archived. If an archived version of an object is deleted, the object is permanently deleted.

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.

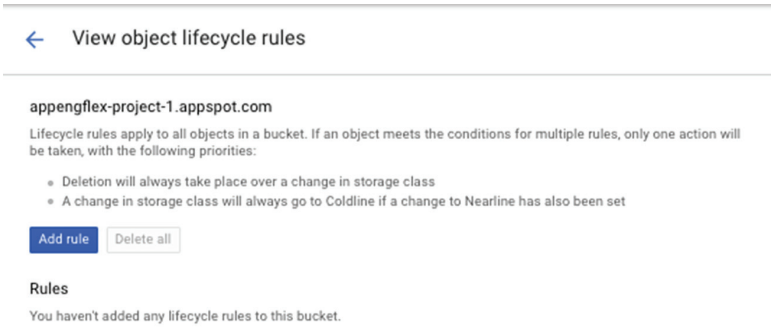
FIGURE 11.3 Form to create a storage bucket from the console. Advanced options are displayed.



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.

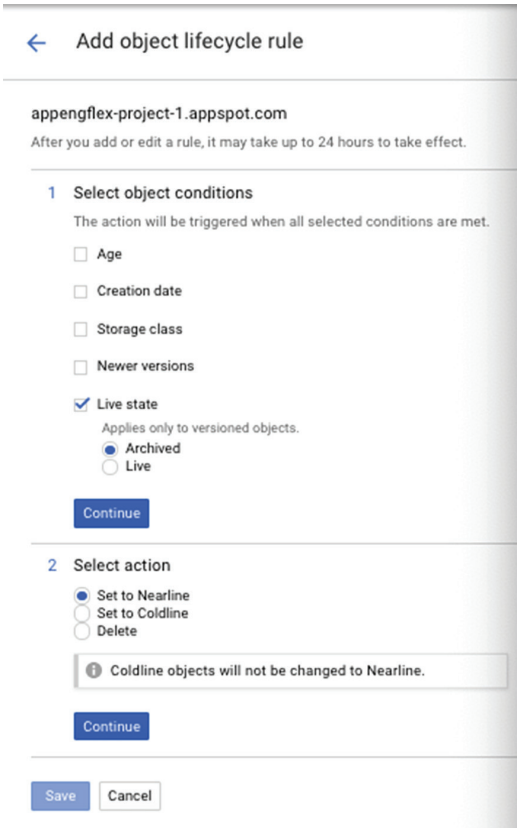
Once you have created a bucket, you define a lifecycle policy. From the Storage menu in the console, choose the Browse option, as shown in Figure 11.4.

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.



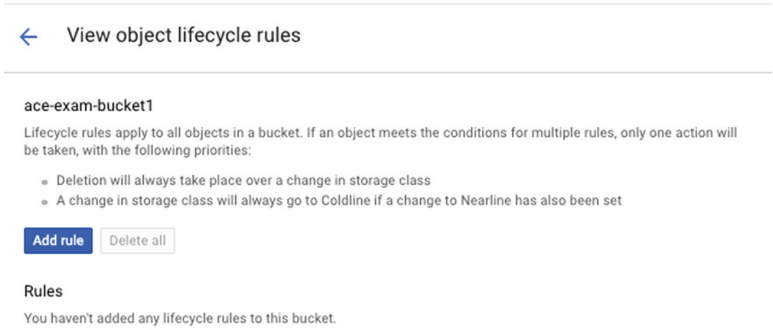
When you add a rule, you need to specify the object condition and the action. Condition options are Age, Creation Data, Storage Class, Newer Versions, and Live State. Live State applies to version objects, and you can set your condition to apply to either live or archived versions of an object. The action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

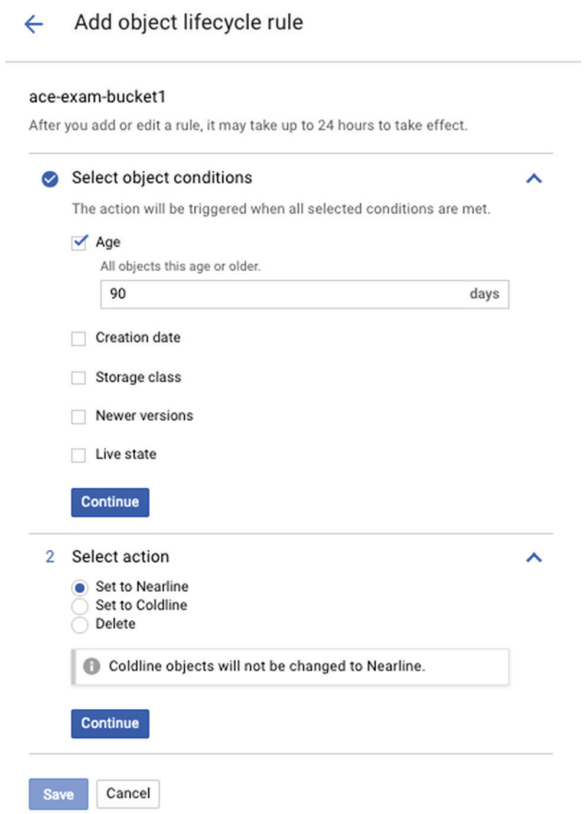
Click the policy status of a bucket to create a lifecycle rule (see Figure 11.7).

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.

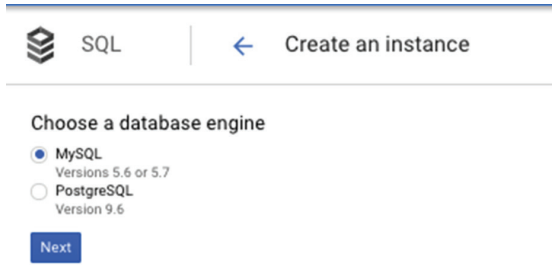
Large enterprises often use Cloud Spanner for applications like global supply chains and financial services applications, while Cloud SQL is often used for web applications, business intelligence, and ecommerce applications.

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

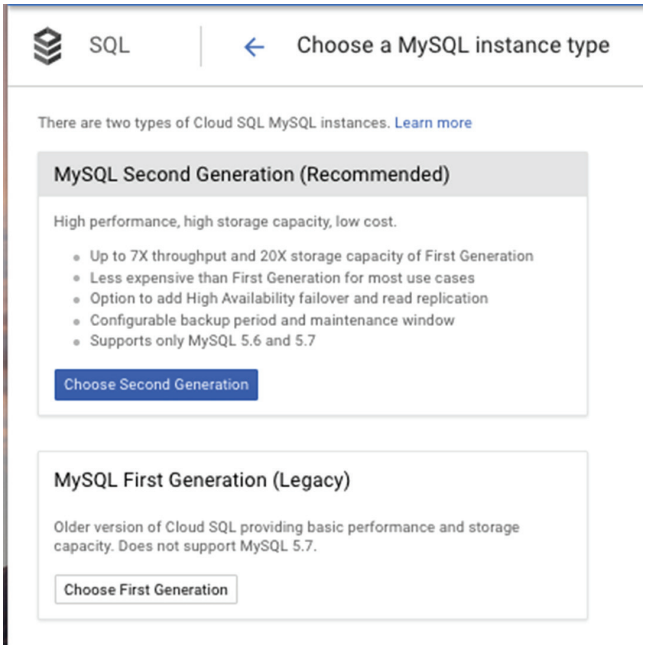
You can create a Cloud SQL instance by navigating to Cloud SQL in the main menu of the console and selecting Create Instance. You will be prompted to choose either a MySQL or PostgreSQL instance, as shown in Figure 11.9.

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

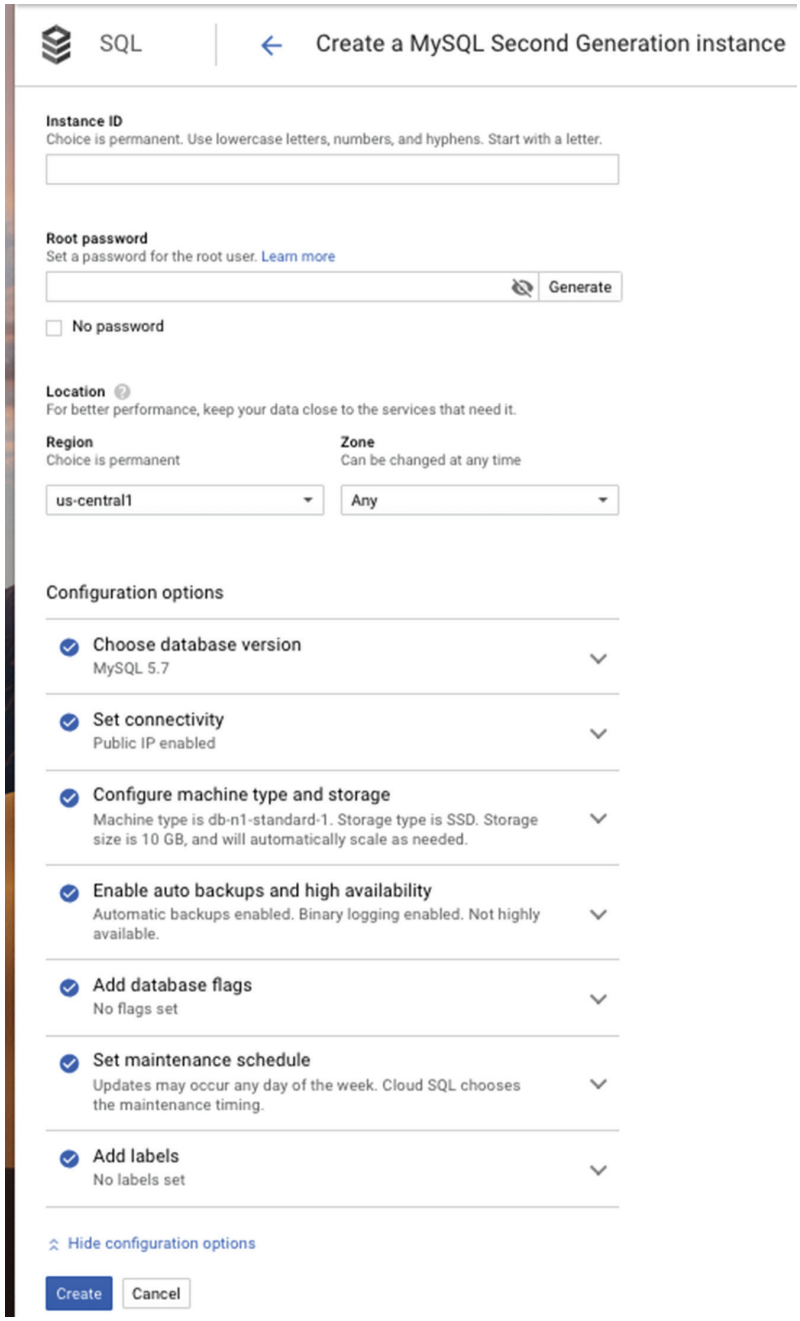
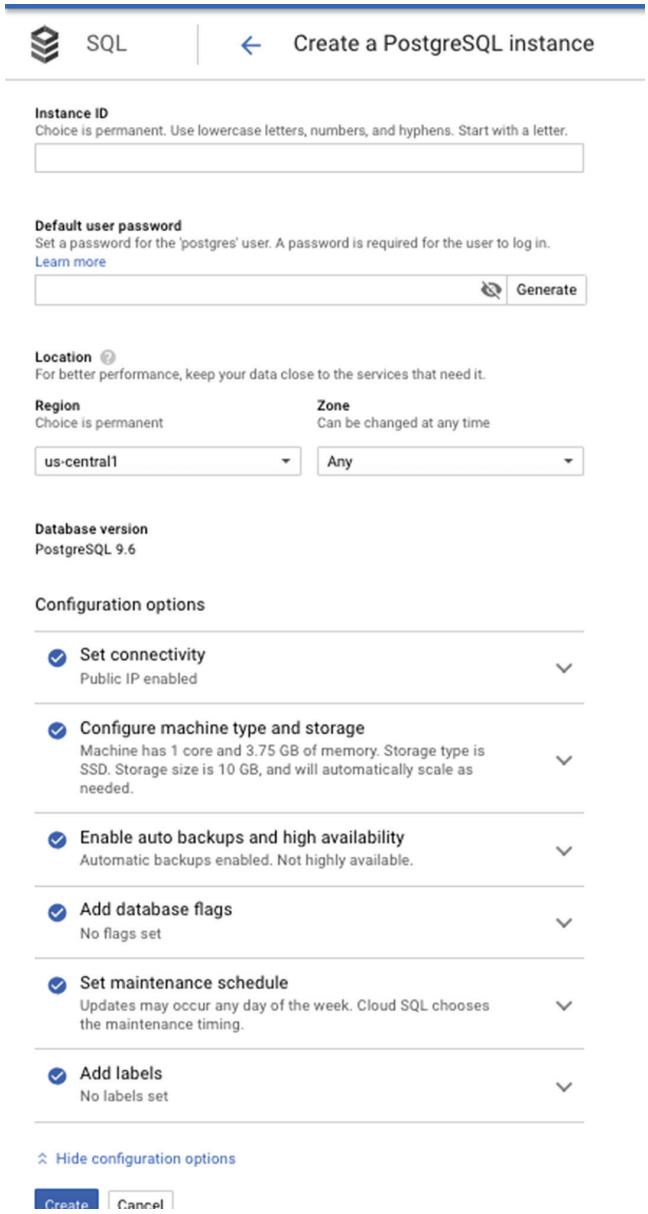


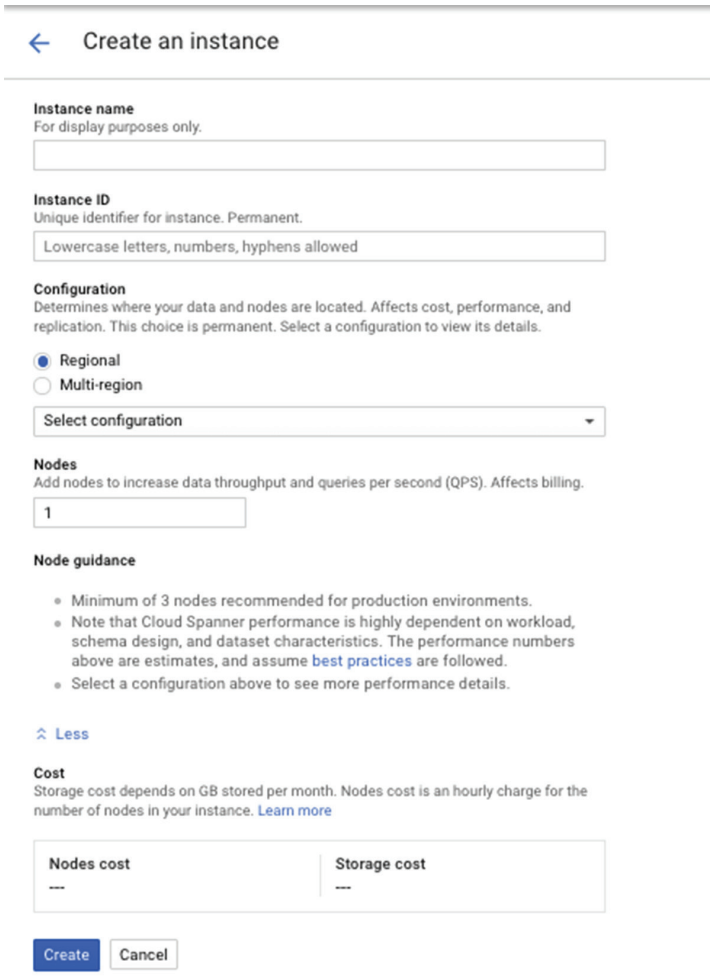
FIGURE 11.12 Configuration form for a PostgreSQL instance



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.

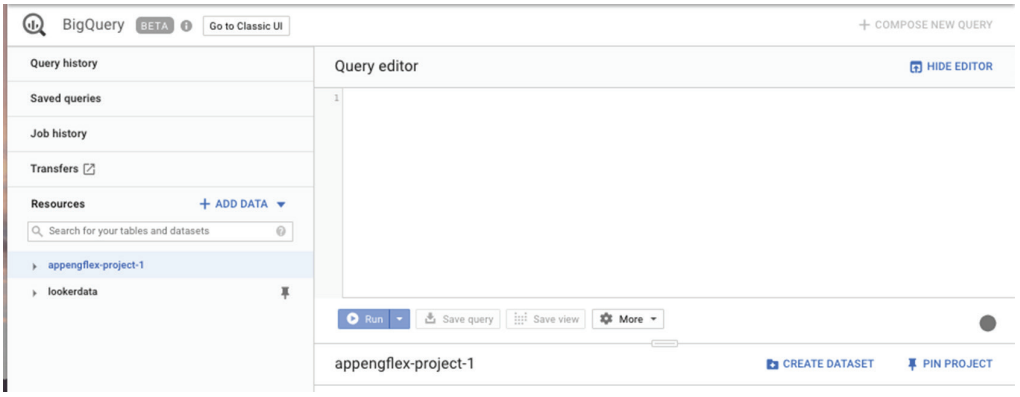
You will also have to choose either a regional or multiregional configuration to determine where nodes and data are located. This will determine cost and replication storage location. If you select regional, you will choose from the list of available regions, such as us-west1, asia-east1, and europe-north1.

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

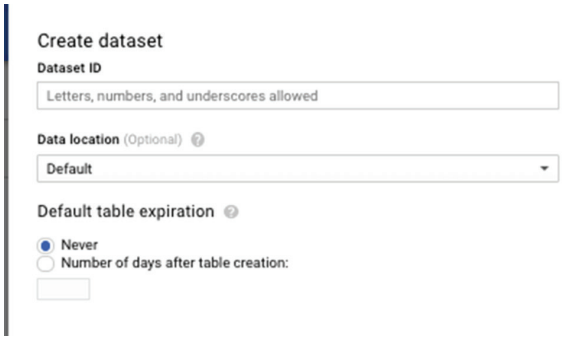
BigQuery is a managed analytics service, which provides storage plus query, statistical, and machine learning analysis tools. BigQuery does not require you to configure instances. Instead, when you first navigate to BigQuery from the console menu, you will see a form such as in Figure 11.14.

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.