First Generation

- Python 2.7
- PHP 5.5
- Go 1.9

Second Generation

- Java 8
- Python 3.7 (beta)
- PHP 7.2 (beta)
- Node.js 8 (beta) and 10 (beta)
- Go 1.11 (beta)

With the second-generation standard environment, developers can use any language extension, but in the first generation only a select set of whitelisted extensions and libraries are allowed. Network access is restricted in the first generation, but users have full network access in the second generation.

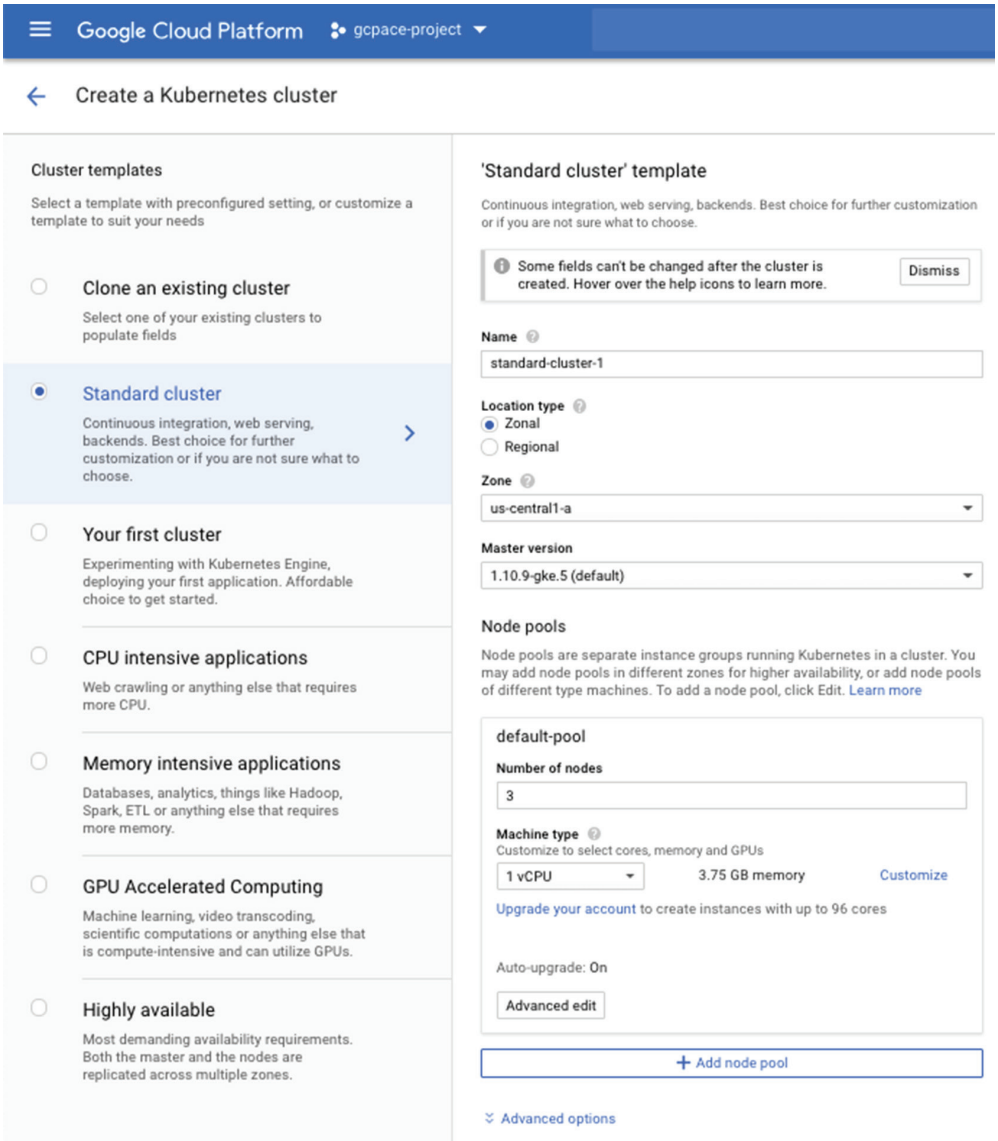
App Engine Flexible Environment

The App Engine flexible environment provides more options and control to developers who would like the benefits of a platform as a service (PaaS) like App Engine, but without the language and customization constraints of the App Engine standard environment (Figure 4.11).

The App Engine flexible environment uses containers as the basic building block abstraction. Users can customize their runtime environments by configuring a container. The flexible environment uses Docker containers, so developers familiar with Docker files can specify base operating system images, additional libraries and tools, and custom tools. It also has native support for Java 8, Eclipse Jetty 9, Python 2.7 and Python 3.6, Node.js, Ruby, PHP, .NET core, and Go.

In some ways, the App Engine flexible environment is similar to the Kubernetes Engine, which will be discussed in the next section. Both of these Google products can run customized Docker containers. The App Engine flexible environment provides a fully managed PaaS and is a good option when you can package your application and services into a small set of containers. These containers can then be autoscaled according to load. Kubernetes Engine, as we will see shortly, is designed to manage containers executing in a cluster that you control. With Kubernetes Engine you have control over your cluster but must monitor and manage that cluster using tools such as Stackdriver monitoring and autoscaling. With the App Engine flexible environment, the health of App Engine servers is monitored by Google and corrected as needed without any intervention on your part.

FIGURE 4.11 Interface to create a Kubernetes cluster in Kubernetes Engine



Use Cases for App Engine

The App Engine product is a good choice for a computing platform when you have little need to configure and control the underlying operating system or storage system. App Engine manages underlying VMs and containers and relieves developers and DevOps

professionals of some common system administration tasks, like patching and monitoring servers.

When to Use App Engine Standard Environment

The App Engine standard environment is designed for applications written in one of the supported languages. The standard environment provides a language-specific runtime that comes with its own constraints. The constraints are fewer in the second-generation App Engine standard environment.

If you are starting a new development effort and plan to use the App Engine standard environment, then it is best to choose second-generation instances. First-generation instances will continue to be supported, but that kind of instance should be used only for applications that already exist and were designed for that platform.

When to Use App Engine Flexible Environment

The App Engine flexible environment is well suited for applications that can be decomposed into services and where each service can be containerized. For example, one service could use a Django application to provide an application user interface, another could embed business logic for data storage, and another service could schedule batch processing of data uploaded through the application. If you need to install additional software or run commands during startup, you can specify those in the Dockerfile. For example, you could add a `run` command to a Dockerfile to run `apt-get update` to get the latest version of installed packages. Docker files are text files with commands for configuring a container, such as specifying a base image to start with and specifying package manager commands, like `apt-get` and `yum`, for installing packages.

The App Engine standard environment scales down to no running instances if there is no load, but this is not the case with the flexible environment. There will always be at least one container running with your service, and you will be charged for that time even if there is no load on the system.

Kubernetes Engine

Compute Engine allows you to create and manage VMs either individually or in groups called *instances groups*. Instance groups let you manage similar VMs as a single unit. This is helpful if you have a fleet of servers that all run the same software and have the same operational lifecycle. Modern software, however, is often built as a collection of services, sometimes referred to as *microservices*. Different services may require different configurations of VMs, but you still may want to manage the various instances as a single resource, or cluster. You can use Kubernetes Engine for that.

Kubernetes is an open source tool created by Google for administering clusters of virtual and bare-metal machines. (Kubernetes is sometimes abbreviated K8s.) Kubernetes is a container orchestration service that helps you. It allows you to do the following:

- Create clusters of VMs that run the Kubernetes orchestration software for containers
- Deploy containerized applications to the cluster
- Administer the cluster
- Specify policies, such as autoscaling
- Monitor cluster health

Kubernetes Engine is GCP's managed Kubernetes service. If you wanted, you could deploy a set of VMs, install Kubernetes on your VMs, and manage the Kubernetes platform yourself. With Kubernetes Engine you get the benefits of Kubernetes without the administrative overhead.

Kubernetes Functionality

Kubernetes is designed to support clusters that run a variety of applications. This is different from other cluster management platforms that provide a way to run one application over multiple servers. Spark, for example, is a big data analytics platform that runs Spark services on a cluster of servers. Spark is not a general-purpose cluster management platform like Kubernetes.

Kubernetes Engine provides the following functions:

- Load balancing across Compute Engine VMs that are deployed in a Kubernetes cluster
- Automatic scaling of nodes (VMs) in the cluster
- Automatic upgrading of cluster software as needed
- Node monitoring and health repair
- Logging
- Support for node pools, which are collections of nodes all with the same configuration

Kubernetes Cluster Architecture

A Kubernetes cluster includes a cluster master node and one or more worker nodes. These are referred to as the *master* and *nodes*, respectively.

The master node manages the cluster. Cluster services, such as the Kubernetes API server, resource controllers, and schedulers, run on the master. The Kubernetes API Server is the coordinator for all communications to the cluster. The master determines what containers and workloads are run on each node.

When a Kubernetes cluster is created from either Google Cloud Console or a command line, a number of nodes are created as well. These are Compute Engine VMs. The default

VM type is n1-standard-1 (1 vCPU and 3.75GB memory), but you can specify a different machine type when creating the cluster.

Kubernetes deploys containers in groups called *pods*. Containers within a single pod share storage and network resources. Containers within a pod share an IP address and port space. A pod is a logically single unit for providing a service. Containers are deployed and scaled as a unit.

It is important to note that some overhead is dedicated to running Kubernetes software on nodes. Some amount of CPU and memory is allocated for Kubernetes and therefore is not available for workload processing. Kubernetes Engine reserves memory resources as follows:

- 25 percent of the first 4GB of memory
- 20 percent of the next 4GB of memory, up to 8GB
- 10 percent of the next 8GB of memory, up to 16GB
- 6 percent of the next 112GB of memory, up to 128GB
- 2 percent of any memory above 128GB

CPU resources are reserved as follows:

- 6 percent of the first core
- 1 percent of the next core (up to two cores)
- 0.5 percent of the next two cores (up to four cores)
- 0.25 percent of any cores above four cores

Kubernetes High Availability

One way Kubernetes maintains cluster health is by shutting down pods that become starved for resources. Kubernetes supports something called *eviction policies* that set thresholds for resources. When a resource is consumed beyond the threshold, then Kubernetes will start shutting down pods.

Another way Kubernetes provides for high reliability is by running multiple identical pods. A group of running identical pods is called a *deployment*. The identical pods are referred to as *replicas*.

When deployments are rolled out, they can be in one of three states.

- Progressing, which means the deployment is in the process of performing a task
- Completed, which means the rollout of containers is complete and all pods are running the latest version of containers
- Failed, which indicates the deployment process encountered a problem it could not recover from

There are additional considerations when running stateful applications versus stateless applications. Those issues will be addressed in Chapter 7.

Kubernetes Engine Use Cases

Kubernetes Engine is a good choice for large-scale applications that require high availability and high reliability. Kubernetes Engine supports the concept of pods and deployment sets, which allow application developers and administrators to manage services as a logical unit. This can help if you have a set of services that support a user interface, another set that implements business logic, and a third set that provides backend services. Each of these different groups of services can have different lifecycles and scalability requirements. Kubernetes helps to manage these at levels of abstraction that make sense for users, developers, and DevOps professionals.

Cloud Functions

Cloud Functions is a serverless computing platform designed to run single-purpose pieces of code in response to events in the GCP environment. There is no need to provision or manage VMs, containers, or clusters when using Cloud Functions. Code that is written in Node.js 6, Node.js 8, or Python 3.7 can run in Cloud Functions.

Cloud Functions is not a general-purpose computing platform like Compute Engine or App Engine. Cloud Functions provides the "glue" between services that are otherwise independent.

For example, one service may create a file and upload it to Cloud Storage, and another service has to pick up those files and perform some processing on the file. Both of these services can be developed independently. There is no need for either to know about the other. However, you will need some way to detect that a new file has been written to Cloud Storage, and then the other application can begin processing it. We don't want to write applications in ways that make assumptions about other processes that may provide input or consume output. Services can change independently of each other. We should not have to keep track of dependencies between services if we can avoid it. Cloud Functions helps us avoid that situation.

Cloud Functions Execution Environment

GCP manages everything that is needed to execute your code in a secure, isolated environment. Of course, below the serverless abstraction, there are virtual and physical servers running your code, but you as a cloud engineer do not have to administer any of that infrastructure. Three key things to remember about Cloud Functions are the following:

- The functions execute in a secure, isolated execution environment.
- Compute resources scale as needed to run as many instances of Cloud Functions as needed without you having to do anything to control scaling.
- The execution of one function is independent of all others. The lifecycles of Cloud Functions are not dependent on each other.

There is an important corollary to these key points. That is, Cloud Functions may be running in multiple instances at one time. If two mobile app users uploaded an image file for processing at the same time, two different instances of Cloud Functions would execute at roughly the same time. You do not have to do anything to prevent conflicts between the two instances; they are independent.

Since each invocation of a Cloud Function runs in a separate instance, functions do not share memory or variables. In general, this means that Cloud Functions should be stateless. That means the function does not depend on the state of memory to compute its output. This is a reasonable constraint in many cases, but sometimes you can optimize processing if you can save state between invocations. Cloud Functions does offer some ways of doing this, which will be described in Chapter 11.

Cloud Functions Use Cases

Cloud Functions is well suited to short-running, event-based processing. If your workflows upload, modify, or otherwise alter files in Cloud Storage or use message queues to send work between services, then the Cloud Functions service is a good option for running code that starts the next step in processing. Some application areas that fit this pattern include the following:

- Internet of Things (IoT), in which a sensor or other device can send information about the state of a sensor. Depending on the values sent, Cloud Functions could trigger an alert or start processing data that was uploaded from the sensor.
- Mobile applications that, like IoT apps, send data to the cloud for processing
- Asynchronous workflows in which each step starts at some time after the previous steps completes, but there are no assumptions about when the processing steps will complete

Summary

GCP offers several computing options. The options vary in the level of control that you, as a user of GCP, have over the computing platform. Generally, with more control comes more responsibility and management overhead. Your objective when choosing a computing platform is to choose one that meets your requirements while minimizing DevOps overhead and cost.

Compute Engine is the GCP service that lets you provision VMs. You can choose from predefined configurations, or you can create a custom configuration with the best combination of virtual CPUs and memory for your needs. If you can tolerate some disruption in VM functioning, you can save a significant amount of money by using preemptible VMs.

App Engine is Google's PaaS offering. This is one of the serverless options. You provide application code and, in the case of the App Engine flexible environment, a specification for a

Docker container to run your application. The App Engine standard environment is appropriate for applications that can run in language-specific sandboxes.

Modern software applications are built on multiple services that may have different computing requirements and change on different lifecycles. Kubernetes Engine runs clusters of servers that can be used to run a variety of services while efficiently allocating work to servers as needed. Kubernetes Engine also provides monitoring, scaling, and remediation when something goes wrong with a VM in the cluster.

Loosely coupled applications may be strung together to implement complex workflows. Often, we want each component to be independent of others. In such cases, we often need to execute "glue" code that moves workload from one stage to another. Cloud Functions is the serverless computing option designed to meet this need.

Exam Essentials

Understand how images are used to create instances of VMs and how VMs are organized in projects. Instances run images, which contain operating systems, libraries, and other code. When you create an instance, you specify a project to contain the instance.

Know that GCP has multiple geographic regions and regions have one or more zones. VMs run in zones. A region is a geographical location, such as asia-east1, europe-west2, and us-east4. The zones within a region are linked by low-latency, high-bandwidth network connections.

Understand what preemptible VMs are and when they are appropriate to use. Also understand when not to use them. GCP offers an option called a preemptible VM for workloads that can be disrupted without creating problems.

Understand the difference between the App Engine standard and flexible environments. The standard environment runs a language-specific platform, and the App Engine flexible environment allows you to run custom containers.

Know that Kubernetes is a container orchestration platform. It also runs containers in a cluster.

Understand Kubernetes. It provides load balancing, automatic scaling, logging, and node health checks and repair.

Understand Cloud Functions. This service is used to run programs in response to events, such as file upload or a message being added to a queue.

Review Questions

You can find the answers in the Appendix.

1. You are deploying a Python web application to GCP. The application uses only custom code and basic Python libraries. You expect to have sporadic use of the application for the foreseeable future and want to minimize both the cost of running the application and the DevOps overhead of managing the application. Which computing service is the best option for running the application?
   - **A.** Compute Engine
   - **B.** App Engine standard environment
   - **C.** App Engine flexible environment
   - **D.** Kubernetes Engine

2. Your manager is concerned about the rate at which the department is spending on cloud services. You suggest that your team use preemptible VMs for all of the following except which one?
   - **A.** Database server
   - **B.** Batch processing with no fixed time requirement to complete
   - **C.** High-performance computing cluster
   - **D.** None of the above

3. What parameters need to be specified when creating a VM in Compute Engine?
   - **A.** Project and zone
   - **B.** Username and admin role
   - **C.** Billing account
   - **D.** Cloud Storage bucket

4. Your company has licensed a third-party software package that runs on Linux. You will run multiple instances of the software in a Docker container. Which of the following GCP services could you use to deploy this software package?
   - **A.** Compute Engine only
   - **B.** Kubernetes Engine only
   - **C.** Compute Engine, Kubernetes Engine, and the App Engine flexible environment only
   - **D.** Compute Engine, Kubernetes Engine, the App Engine flexible environment, or the App Engine standard environment

5. You can specify packages to install into a Docker container by including commands in which file?
   - **A.** `Docker.cfg`
   - **B.** `Dockerfile`
   - **C.** `Config.dck`
   - **D.** `install.cfg`

6. How much memory of a node does Kubernetes require as overhead?
   - **A.** 10GB to 20GB
   - **B.** 1GB to 2GB
   - **C.** 1.5GB
   - **D.** A scaled amount starting at 25 percent of memory and decreasing to 2 percent of marginal memory as the total amount of memory increases.

7. Your manager is making a presentation to executives in your company advocating that you start using Kubernetes Engine. You suggest that the manager highlight all the features Kubernetes provides to reduce the workload on DevOps engineers. You describe several features, including all of the following except which one?
   - **A.** Load balancing across Compute Engine VMs that are deployed in a Kubernetes cluster
   - **B.** Security scanning for vulnerabilities
   - **C.** Automatic scaling of nodes in the cluster
   - **D.** Automatic upgrading of cluster software as needed

8. Your company is about to release a new online service that builds on a new user interface experience driven by a set of services that will run on your servers. There is a separate set of services that manage authentication and authorization. A data store set of services keeps track of account information. All three sets of services must be highly reliable and scale to meet demand. Which of the GCP services is the best option for deploying this?
   - **A.** App Engine standard environment
   - **B.** Compute Engine
   - **C.** Cloud Functions
   - **D.** Kubernetes Engine

9. A mobile application uploads images for analysis, including identifying objects in the image and extracting text that may be embedded in the image. A third party has created the mobile application, and you have developed the image analysis service. You both agree to use Cloud Storage to store images. You want to keep the two services completely decoupled, but you need a way to invoke the image analysis as soon as an image is uploaded. How should this be done?
   - **A.** Change the mobile app to start a VM running the image analysis service and have that VM copy the file from storage into local storage on the VM. Have the image service run on the VM.
   - **B.** Write a function in Python that is invoked by Cloud Functions when a new image file is written to the Cloud Storage bucket that receives new images. The function should submit the URL of the uploaded file to the image analysis service. The image analysis service will then load the image from Cloud Storage, perform analysis, and generate results, which can be saved to Cloud Storage.
   - **C.** Have a Kubernetes cluster running continuously, with one pod dedicated to listing the contents of the upload bucket and detecting new files in Cloud Storage and another pod dedicated to running the image analysis software.

D. Have a Compute Engine VM running and continuously listing the contents of the upload bucket in Cloud Storage to detect new files. Another VM should be continually running the image analysis software.

10. Your team is developing a new pipeline to analyze a stream of data from sensors on manufacturing devices. The old pipeline occasionally corrupted data because parallel threads overwrote data written by other threads. You decide to use Cloud Functions as part of the pipeline. As a developer of a Cloud Function, what do you have to do to prevent multiple invocations of the function from interfering with each other?

A. Include a check in the code to ensure another invocation is not running at the same time.

B. Schedule each invocation to run in a separate process.

C. Schedule each invocation to run in a separate thread.

D. Nothing. GCP ensures that function invocations do not interfere with each other.

11. A client of yours processes personal and health information for hospitals. All health information needs to be protected according to government regulations. Your client wants to move their application to Google Cloud but wants to use the encryption library that they have used in the past. You suggest that all VMs running the application have the encryption library installed. Which kind of image would you use for that?

A. Custom image

B. Public image

C. CentOS 6 or 7

12. What is the lowest level of the resource hierarchy?

A. Folder

B. Project

C. File

D. VM instance

13. Your company is seeing a marked increase in the rate of customer growth in Europe. Latency is becoming an issue because your application is running in us-central1. You suggest deploying your services to a region in Europe. You have several choices. You should consider all of the following factors except which one?

A. Cost

B. Latency

C. Regulations

D. Reliability

14. What role gives users full control over Compute Engine instances?

A. Compute Manager role

B. Compute Admin role

C. Compute Manager role

D. Compute Security Admin

15. Which of the following are limitations of a preemptible VM?

A. Will be terminated within 24 hours.

B. May not always be available. Availability may vary across zones and regions.

C. Cannot migrate to a regular VM.

D. All of the above

16. Custom VMs can have up to how many vCPUs?

A. 16

B. 32

C. 64

D. 128

17. When using the App Engine standard environment, which of the following language's runtime is not supported?

A. Java

B. Python

C. C

D. Go

18. Kubernetes reserves CPU resources in percentage of cores available. The percentage is what range?

A. 1 percent to 10 percent

B. 0.25 percent to 6 percent

C. 0.25 percent to 2 percent

D. 10 percent to 12 percent

19. Kubernetes deployments can be in what states?

A. Progressing, stalled, completed

B. Progressing, completed, failed

C. Progressing, stalled, failed, completed

D. Progressing, stalled, running, failed, completed

20. A client has brought you in to help reduce their DevOps overhead. Engineers are spending too much time patching servers and optimizing server utilization. They want to move to serverless platforms as much as possible. Your client has heard of Cloud Functions and wants to use them as much as possible. You recommend all of the following types of applications except which one?

A. Long-running data warehouse data load procedures

B. IoT backend processing

C. Mobile application event processing

D. Asynchronous workflows

Chapter 5

Computing with Compute Engine Virtual Machines

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

✓ **1.3 Installing and configuring the command line interface (CLI), specifically Cloud SDK (e.g., setting the default project)**

✓ **2.2 Planning and configuring compute resources. Considerations include:**

- Selecting appropriate compute choices for a given workload (e.g., Compute Engine, Kubernetes Engine, App Engine)
- Using preemptible VMs and custom machine types as appropriate

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

In this chapter, you will learn about Google Cloud Console, a graphical user interface for working with Google Cloud Platform (GCP). You will learn how to install Google Cloud SDK and use it to create virtual machine instances and how to use Cloud Shell as an alternative to installing Google Cloud SDK locally.

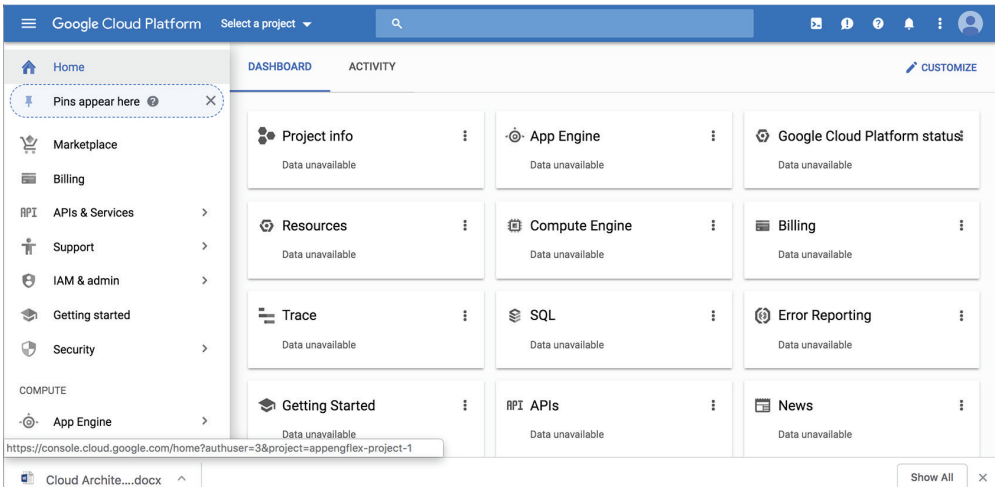
Creating and Configuring Virtual Machines with the Console

Let's create a VM in Compute Engine. We have three options for doing this: we can use Google Cloud Console, Google Cloud SDK, or Google Cloud Shell. Let's start with the console.

Google Cloud Console is a web-based graphical user interface for creating, configuring, and managing resources in Google Cloud. In this chapter, we will use it to create a VM.

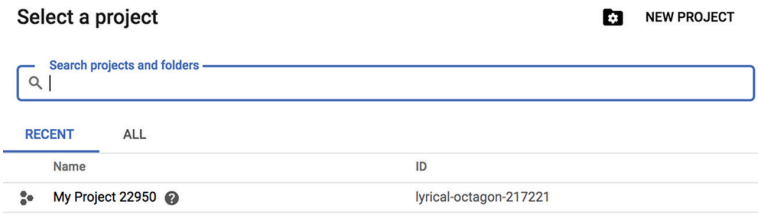
To open the console, navigate in your browser to `https://console.cloud.google.com` and log in. Figure 5.1 shows an example of the main form in the console.

FIGURE 5.1 The main starting form of Google Cloud Console



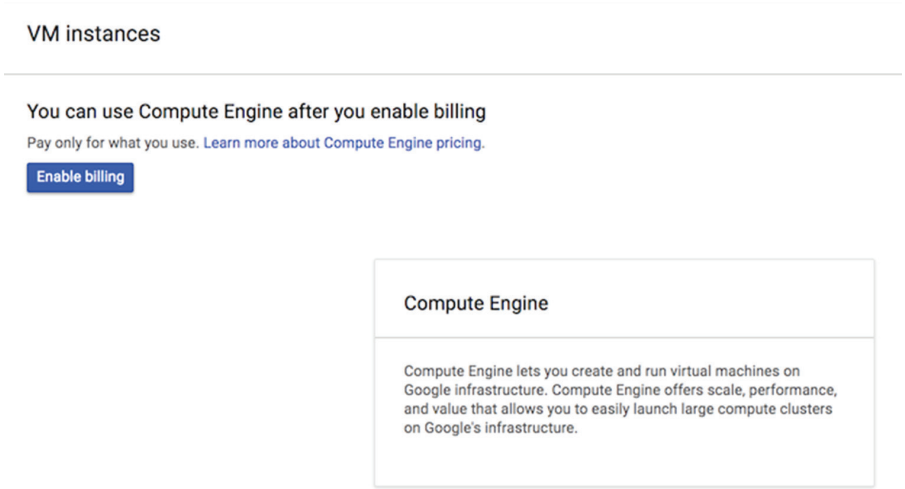
In the upper-left section of the form, click the Select A Project option to display the existing projects. You can also create a new project from this form, which is shown in Figure 5.2.

FIGURE 5.2 The Project form lets you choose the project to work with when creating VMs. You can also create a new project here.



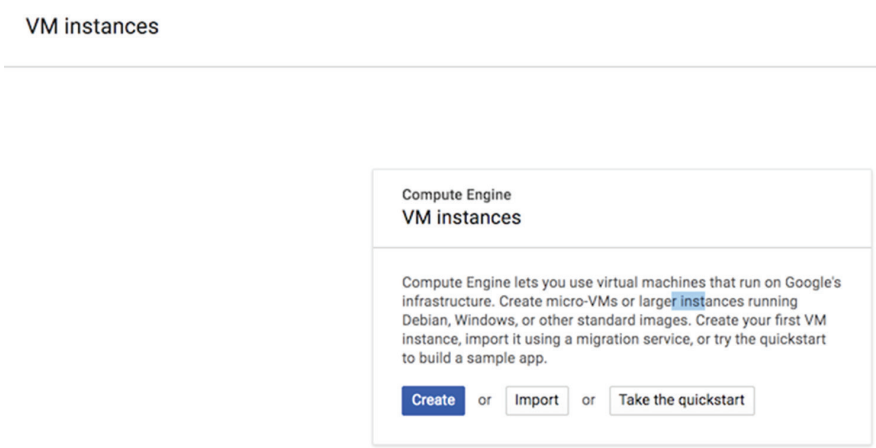
After you select an existing project or create a new project, you can return to the main console panel. The first time you try to work a VM you will have to create a billing account if one has not already been created. Figure 5.3 shows a message and button on the main panel for creating a billing account.

FIGURE 5.3 When a billing account does not exist for a project, you will be given the option to create a billing account when you try to create a VM.



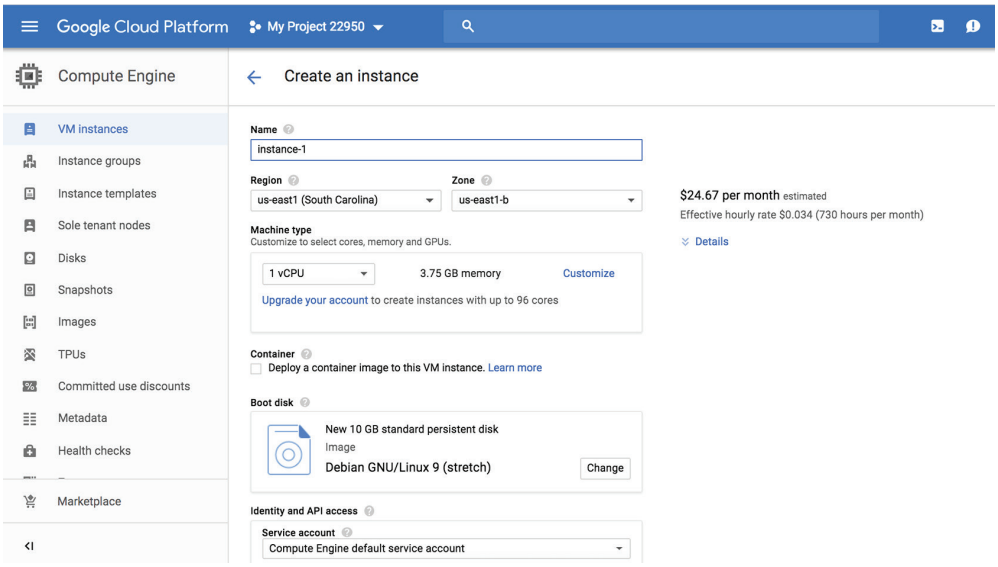
Click Enable Billing and fill in the billing information, such as name, address, and credit card. Once billing is enabled, you will return to the main panel (see Figure 5.4).

FIGURE 5.4 The starting panel for creating a VM



Click the Create button in the dialog box to bring up a VM configuration, as shown in Figure 5.5.

FIGURE 5.5 Part of the main configuration form for creating VMs in Compute Engine



Main Virtual Machine Configuration Details

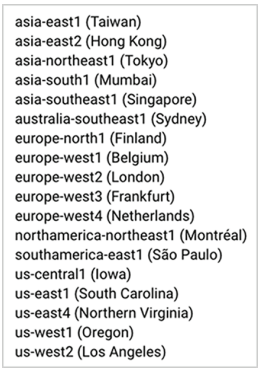
Within the console, you can specify all the needed details about the configuration of the VM that you are creating, including the following:

- Name of the VM
- Region and zone where the VM will run
- Machine type, which determines the number of CPUs and the amount of memory in the VM
- Boot disk, which includes the operating system the VM will run

You can choose the name of your VM. This is primarily for your use. Google Cloud uses other identifiers internally to manage VMs.

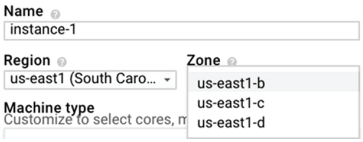
You will need to specify a region. Regions are major geographical areas. A partial list of regions is shown in Figure 5.6.

FIGURE 5.6 A partial list of regions providing Compute Engine services



After you select a region, you can select a zone. Remember, a zone is a data center–like facility within a region. Figure 5.7 shows an example list of zones available in the us-east-1 region.

FIGURE 5.7 A list of zones within the us-east-1 region

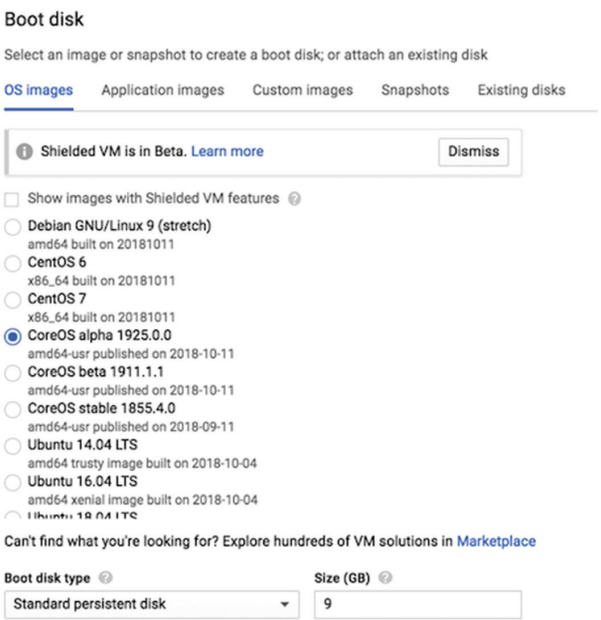


After you specify a region and zone, Google Cloud can determine the VMs available in that zone. Not all zones have the same availability. Figure 5.8 shows an example listing of machine types available in the us-east1-b zone.

FIGURE 5.8 A list of machine types available in the us-east1-b zone

The Boot Disk Option section lists a default configuration. Clicking the Change button brings up the Boot Disk Option dialog, as shown in Figure 5.9.

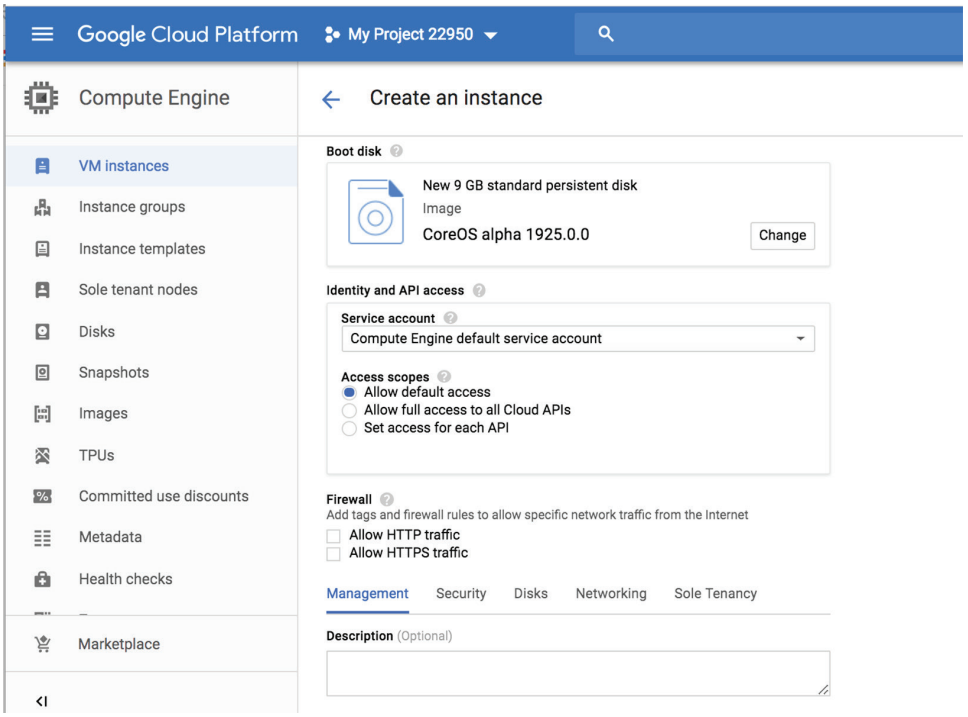
FIGURE 5.9 Dialog for configuring the boot disk of the VM



Here you can choose the operating system you want to use. You can also choose the boot disk type, which can be either Standard Persistent Disk or SSD Persistent Disk. You can also specify the size of the disk.

Following the Boot Disk section is the Identity and API Access section. Here you can specify a service account for the VM and set the scope of API access. If you want the processes running on this VM to use only some APIs, you can use these options to limit the VM's access to specific APIs.

FIGURE 5.10 Identity and API Access and Firewall configurations



In the next section, you can select if you want the VM to accept HTTP or HTTPS traffic.

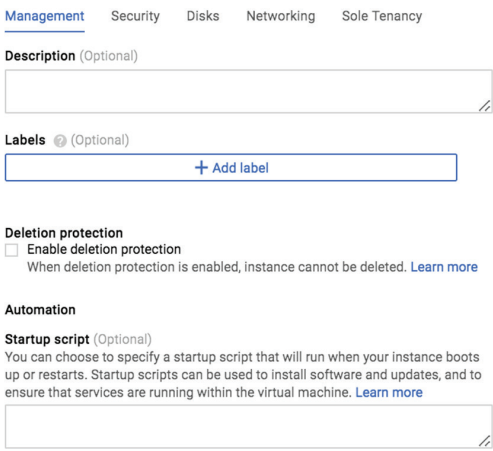
Additional Configuration Details

Click Management, Security, Disks, Networking, and Sole Tenancy to expose additional configuration options.

Management Tab

The Management tab of the form (Figure 5.11) provides a space where you can describe the VM and its use. You can also create labels, which are key-value pairs. You can assign any label you like. Labels and a general description are often used to help manage your VMs and understand how they are being used. Labels are particularly important when your number of servers grows. It is a best practice to include a description and labels for all VMs.

FIGURE 5.11 The first part of the Management tab of the VM creation form



If you want to force an extra confirmation before deleting an instance, you can select the deletion protection option. If someone tries to delete the instance, the operation will fail.

You can specify a startup script to run when the instance starts. Copy the contents of the startup script to the script text box. For example, you could paste a bash or Python script directly into the text box.

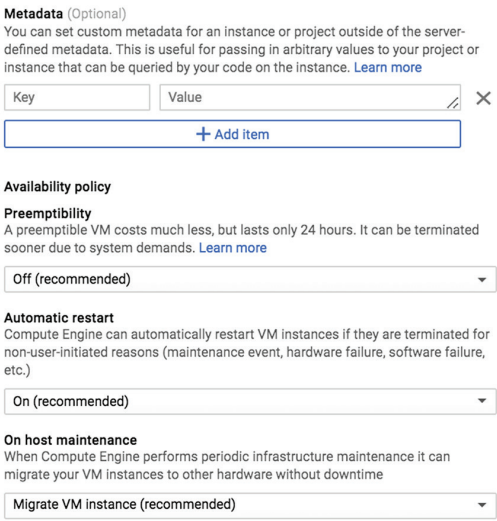
The Metadata section allows you to specify key-value pairs associated with the instance. These values are stored in a metadata server, which is available for querying using the Compute Engine API. Metadata tags are especially useful if you have a common script you want to run on startup or shutdown but want the behavior of the script to vary according to some metadata values.

The Availability Policy sets three parameters.

- Preemptibility, which when enabled allows Google to shut down the server with a 30-second notice. In return, the cost of a preemptible server is much lower than that of a nonpreemptible server.

- Automatic restart, which indicates if the server stops because of a hardware failure, maintenance event, or some other non-user–controlled event
- On host maintenance, which indicates whether the virtual server should be migrated to another physical server when a maintenance event occurs

FIGURE 5.12 The second part of the Management tab of the VM creation form

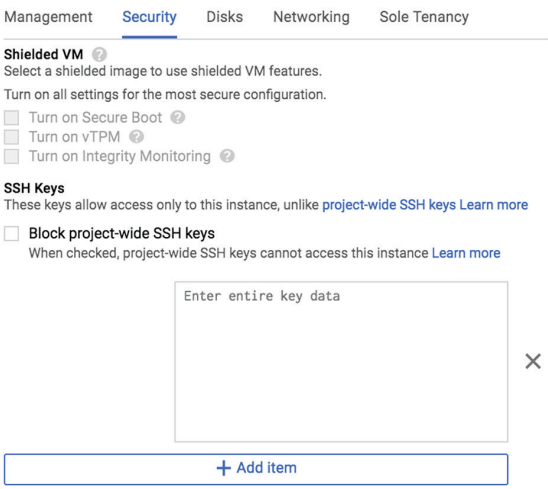


In the Security section, you can specify if you want to use Shielded VMs and Secure Shell (SSH) keys.

Shielded VMs are configured to have additional security mechanisms that you can choose to run. These include the following:

- Secure Boot, which ensures that only authenticated operating system software runs on the VM. It does this by checking the digital signatures of the software. If a signature check fails, the boot process will halt.
- Virtual Trusted Platform Module (vTPM), which is a virtualized version of a trusted platform module (TPM). A TPM is a specialized computer chip designed to protect security resources, like keys and certificates.
- Integrity Monitoring, which uses a known good baseline of boot measurements to compare to recent boot measurements. If the check fails, then there is some difference between the baseline measurement and the current measurements.

FIGURE 5.13 Additional security controls can be placed on VMs.

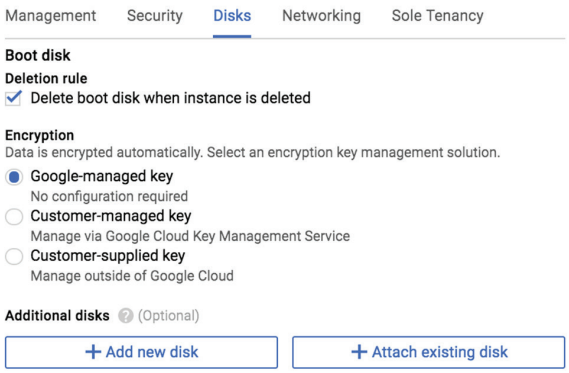


GCP supports the concept of project-wide SSH keys, which are used to give users project-wide access to VMs. You can block that behavior at the VM if you use project-wide SSH keys and do not want all project users to have access to this machine.

The next advanced tab is the Boot Disk tab. Here you can specify whether the boot disk should be deleted when the instance is deleted. You can also select how you would like to manage encryption keys for the boot disk. By default, Google manages those keys.

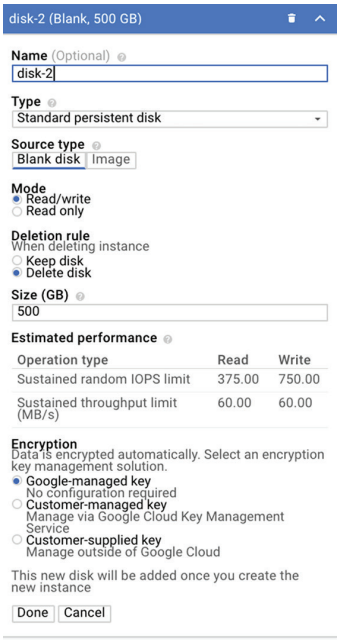
Within the Boot Disk configuration tab, you also have the option of adding a new disk or attaching an existing disk. Figure 5.14 shows the tab for adding a new disk.

FIGURE 5.14 Boot disk advanced configuration



When adding an existing disk, the dialog form appears, as in Figure 5.15. Note that the Disk drop-down has a list of existing disks you can choose from. You can make the disk read-only or read/write. You can also indicate if you want the disk deleted when the instance is deleted. Using an existing disk in read-only mode is a good way of replicating reference data across multiple instances of VMs.

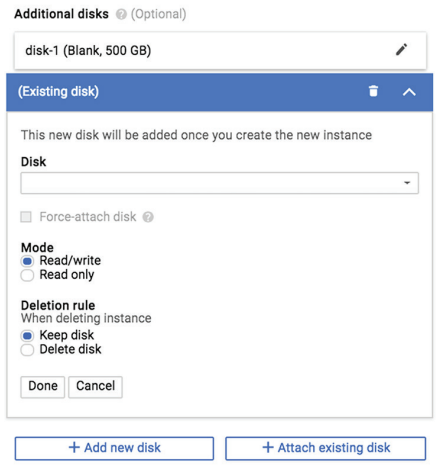
FIGURE 5.15 Dialog for adding an existing disk to a VM



You can also add a new disk using the dialog shown in Figure 5.16. When adding a new disk, you need to provide the following information:

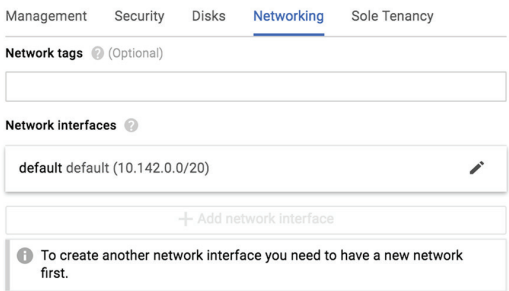
- Name of the disk
- Disk type, either standard or SSD Persistent disk
- Source image, if this is not a blank disk
- Indication of whether to delete the disk when the instance is deleted
- Size in gigabytes
- How the encryption keys will be managed

FIGURE 5.16 Dialog for adding a new disk to a VM



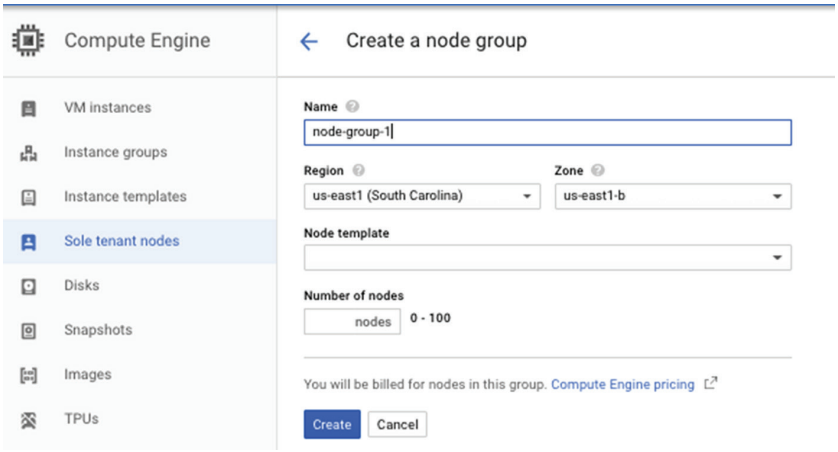
On the Networking tab, you can see the network interface information, including the IP address of the VM. If you have multiple networks, you have the option of adding another network interface to that other network. This use of dual network interfaces can be useful if you are running some type of proxy or server that acts as a control for flow of some traffic between the networks. In addition, you can also add network tags in this dialog (see Figure 5.17).

FIGURE 5.17 Dialog for network configuration of a VM



If you need to ensure that your VMs run on a server only with your other VMs, then you can specify sole tenancy. The Sole Tenancy tab allows you to specify labels regarding sole tenancy for the server (see Figure 5.18).

FIGURE 5.18 Sole Tenancy configuration form



Creating and Configuring Virtual Machines with Cloud SDK

A second way to create and configure VMs is with Google Cloud SDK, which provides a command-line interface. To use Cloud SDK, you will first need to install it on your local device.

Installing Cloud SDK

You have three options for interacting with Google Cloud resources:

- Using a command-line interface
- Using a RESTful interface
- Using the Cloud Shell

Before using either of the first two options from your local system, you will need to install Cloud SDK on your machine. Cloud Console is a graphical user interface you can access through a browser at `https://console.cloud.google.com`.

Cloud SDK can be installed on Linux, Windows, or Mac computers.

Installing Cloud SDK on Linux

If you are using Linux, you can install Cloud SDK using your operating system's package manager. Ubuntu and other Debian distributions use `apt-get` to install packages. Red Hat Enterprise, CentOS, and other Linux distributions use `yum`. For instructions on using `apt-get`, see `https://cloud.google.com/sdk/docs/quickstart-debian-ubuntu`. For instructions on installing on Red Hat Enterprise or CentOS, see `https://cloud.google.com/sdk/docs/quickstart-redhat-centos.Installing`.

Cloud SDK on Mac OS

Instructions for installing on a Mac and the installation file for Cloud SDK are available at `https://cloud.google.com/sdk/docs/quickstart-macos`. The first step is to verify that you have Python 2.7 installed. There are two versions of Cloud SDK, one for 32-bit macOS and one for 64-bit macOS.

Installing Cloud SDK on Windows

To install Cloud SDK on a Windows platform, you will need to download the appropriate installer. You can find instructions at `https://cloud.google.com/sdk/docs/quickstart-windows`.

Example Installation on Ubuntu Linux

The first step in installing Cloud SDK is to get the appropriate version of the package for your operating system. The following commands are for installing Cloud SDK on Ubuntu. See `https://cloud.google.com/sdk/docs/quickstart-debian-ubuntu` for any updates to this procedure.

You need to identify which version of the operating system you are using because the Google naming convention for Cloud SDK references the operating system name. The following command creates an environment variable with the name of the Cloud SDK package for the current operating system:

```
export CLOUD_SDK_REPO="cloud-sdk-$(lsb_release -c -s)"
```

Note that if you receive an error message that the `lsb_release` command is not found, you can install it with the following commands:

```
sudo apt-get update
sudo apt-get install lsb-core
```

You can see the value of the variable `CLOUD_SDK_REPO` using the following command:

```
echo $CLOUD_SDK_REPO
```

This will display a value such as `cloud-sdk-bionic`. Bionic is the code name for Ubuntu 18.04.

Next, you need to specify where to find Cloud SDK. We do this by adding the URL of the Cloud SDK package to the /etc/apt/sources.list.d/google-cloud-sdk.list file. Now apt-get will know where to find the package.

```
echo "deb http://packages.cloud.google.com/apt $CLOUD_SDK_REPO main" | sudo tee
-a /etc/apt/sources.list.d/google-cloud-sdk.list
```

You also need to import the GCP public key, which you do with this command:

```
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
```

Finally, you need to update the apt-get package list and then use apt-get to install Cloud SDK.

```
sudo apt-get update && sudo apt-get install google-cloud-sdk
```

Now Cloud SDK is installed and you can execute commands using it. The first step is to initialize Cloud SDK using the gcloud init command, as shown here:

```
gcloud init
```

When you receive an authentication link, copy it into your browser. You are prompted to authenticate with Google when you go to that URL. Next, a response code appears in your browser. Copy that to your terminal window and paste it in response to the prompt that should appear.

Next, you are prompted to enter a project. If projects already exist in your account, they will be listed. You also have the option of creating a new project at this point. The project you select or create will be the default project used when issuing commands through Cloud SDK.

Creating a Virtual Machine with Cloud SDK

To create a VM from the command line, you will use the gcloud command. You use this command for many cloud management tasks, including the following:

- Compute Engine
- Cloud SQL instances
- Kubernetes Engine
- Cloud Dataproc
- Cloud DNS
- Cloud Deployment

The gcloud command is organized into a hierarchy of groups, such as the compute group for Compute Engine commands. We'll discuss other groups in later chapters; the focus here is on Compute Engine.

A typical gcloud command starts with the group, as shown here:

```
gcloud compute
```

A subgroup is used in Compute Engine commands to indicate what type of compute resource you are working with. To create an instance, you use this command:

```
gcloud compute instances
```

And the action you want to take is to create an instance, so you would use this:

```
gcloud compute instances create ace-instance-1, ace-instance-2
```

If you do not specify additional parameters, such as a zone, Google Cloud will use your information from your default project. You can view your project information using the following gcloud command:

```
gcloud compute project-info describe
```

To create a VM in the us-central1-a zone, add the zone parameter like this:

```
gcloud compute instances create ace-instance-1 ace-instance-2 --zone
us-central1-a
```

You can list the VMs you've create using this:

```
gcloud compute instances list
```

Here are commonly used parameters with the create instance command:

- `--boot-disk-size` is the size of the boot disk for a new disk. Disk size may be between 10GB and 2TB.
- `--boot-disk-type` is the type of disk. Use `gcloud compute disk-types list` for a list of disk types available in the zone the VM is created in.
- `--labels` is the list of key-value pairs in the format of KEY=VALUE.
- `--machine-type` is the type of machine to use. If not specified, it uses n1-standard-1. Use `gcloud compute machine-types list` to view a list of machine types available in the zone you are using.
- `--preemptible`, if included, specifies that the VM will be preemptible.

For additional parameters, see the `glcoud compute instance create` documentation at `https://cloud.google.com/sdk/gcloud/reference/compute/instances/create`.

To create a standard VM with 8 CPUs and 30GB of memory, you can specify n1-standard-8 as the machine type.

```
gcloud compute instances create ace-instance-n1s8 --machine-type=n1-standard-8
```

If you want to make this instance preemptible, you add the preemptible parameter:

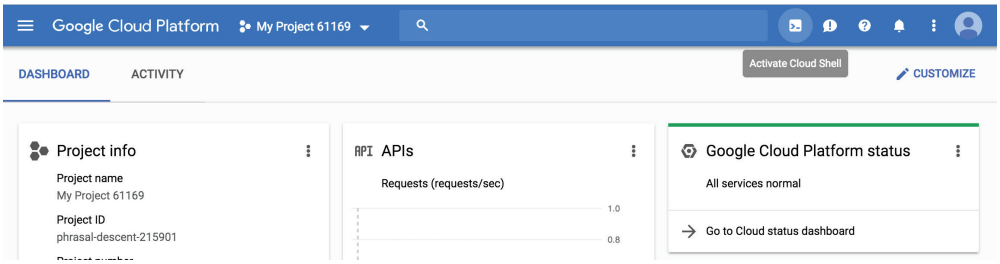
```
gcloud compute instances create --machine-type=n1-standard-8 --preemptible ace-
instance-1
```

Creating a Virtual Machine with Cloud Shell

An alternative to running gcloud commands locally is to run them in a cloud instance. Cloud Shell provides this capability. To use Cloud Shell, start it from Cloud

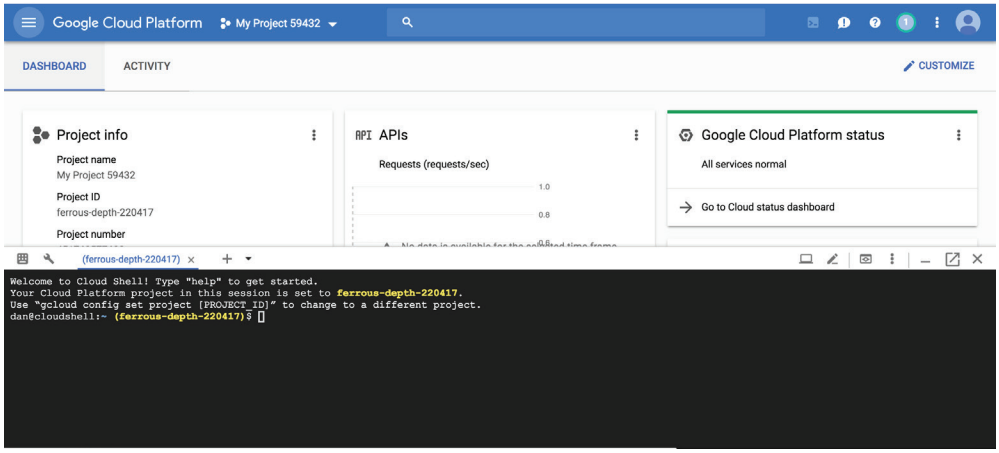
Console by clicking the shell icon in the upper-right corner of the browser, as shown in Figure 5.19.

FIGURE 5.19 Cloud Shell is activated through Cloud Console.



Cloud Shell provides a Linux command line, as shown in Figure 5.20, and Cloud SDK is already installed. All `gcloud` commands that you can enter on your local device with Cloud SDK installed can be used in Cloud Shell.

FIGURE 5.20 Cloud Shell opens a command-line window in the browser.



Basic Virtual Machine Management

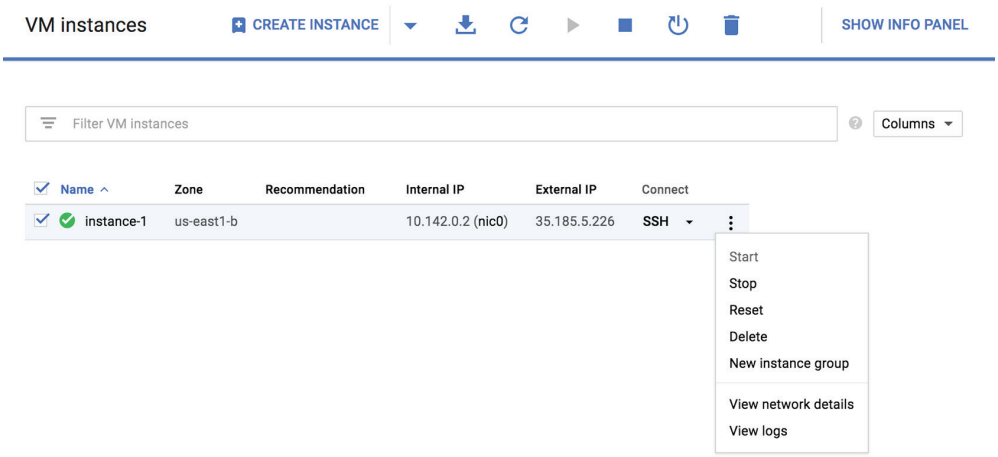
When VMs are running, you can perform basic management tasks by using the console or using `gcloud` commands.

Starting and Stopping Instances

In the console you view a list of instances by selecting Compute Engine and then VM Instances from the left-side panel of the console. You can then select a VM to operate on

and list command options by clicking the three dot icons on the right. Figure 5.21 shows an example.

FIGURE 5.21 Basic operations on VMs can be performed using a pop-up menu in the console.



Note that you can start a stopped instance using the `start` command that is enabled in the pop-up for stopped instances.

You can also use `gcloud` to stop an instance with the following command, where `INSTANCE-NAME` is the name of the instance:

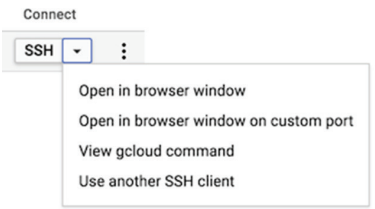
```
gcloud compute instances stop INSTANCE-NAME
```

Network Access to Virtual Machines

As a cloud engineer, you will sometimes need to log into a VM to perform some administration tasks. The most common way is to use SSH when logging into a Linux server or Remote Desktop Protocol (RDP) when logging into a Windows server.

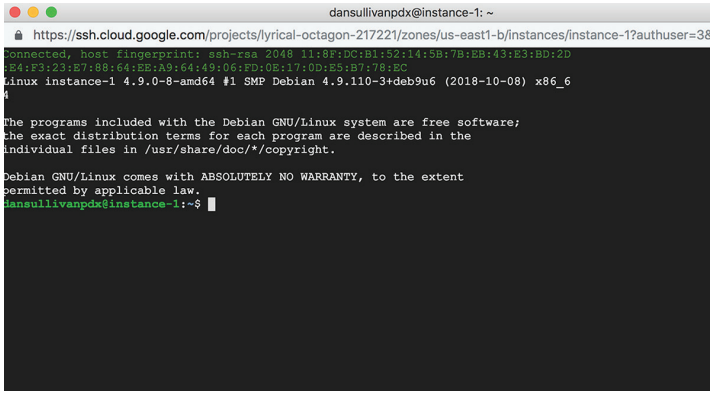
Figure 5.22 shows the set of options for using SSH from the console. This list of options appears when you click the SSH button associated with a VM.

FIGURE 5.22 From the console, you can start an SSH session to log into a Linux server.



Choosing the Open In Browser Window option will open a new browser window and display a terminal window for accessing the command line on the server.

FIGURE 5.23 A terminal window opens in a new browser window when using Cloud Shell.



Monitoring a Virtual Machine

While your VM is running you can monitor CPU, disk, and network load by viewing the Monitoring page in the VM Instance Details page.

To access monitoring information in the console, select a VM instance from the VM Instance page by clicking the name of the VM you want to monitor. This will show the Details page of the VM. Select the Monitoring option near the top of the page to view monitoring details.

Figures 5.24, 5.25, and 5.26 show the information displayed about CPU, network utilization, and disk operations.

FIGURE 5.24 The Monitoring tab of the VM Instance Details page shows CPU utilization.

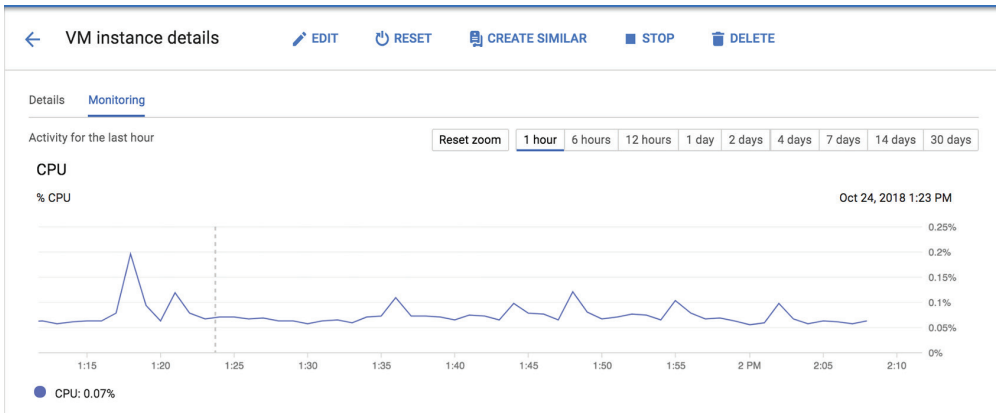


FIGURE 5.25 The Monitoring tab of the VM Instance Details page also shows network utilization.

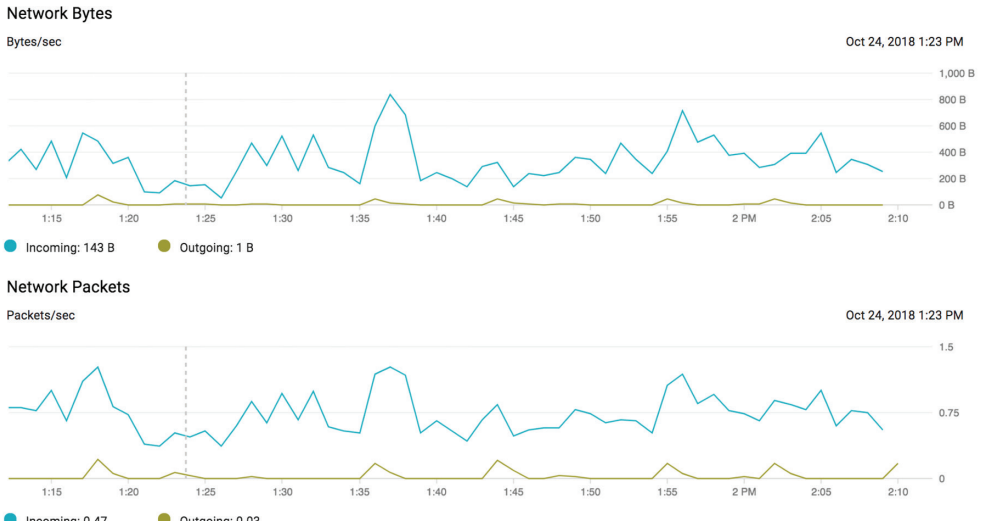
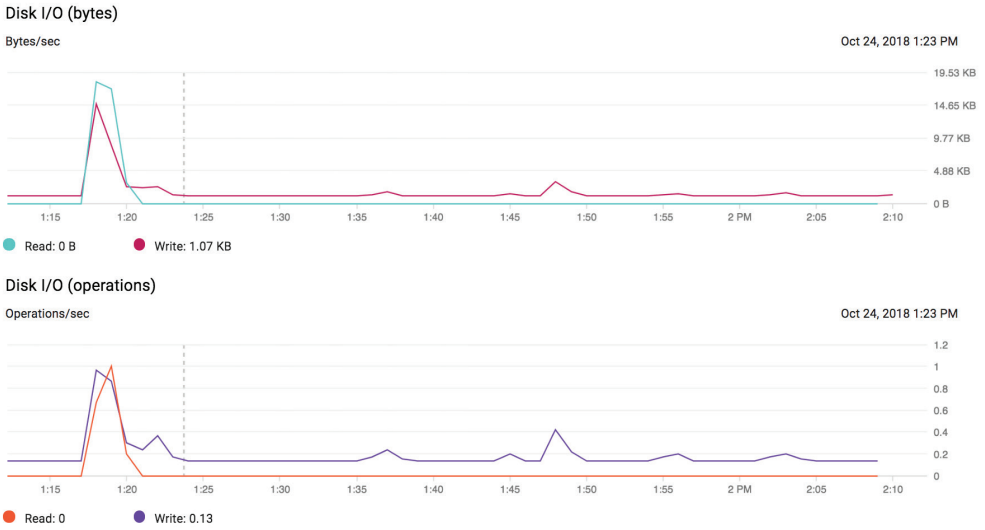


FIGURE 5.26 Disk utilization is included in the Monitoring tab of the VM Instance Details page.



Cost of Virtual Machines

Part of the basic management of a VM is tracking the costs of the instances you are running. If you want to track costs automatically, you can enable Cloud Platform billing and setup Billing Export. This will produce daily reports on the usage and cost of VMs.

The following are the most important things to remember about VM costs:

- VMs are billed in 1-second increments.
- The cost is based on machine type. The more CPUs and memory used, the higher the cost.
- Google offers discounts for sustained usage.
- VMs are charged for a minimum of 1 minute of use.
- Preemptible VMs can save you up to 80 percent of the cost of a VM.

Guidelines for Planning, Deploying, and Managing Virtual Machines

Consider the following guidelines to help with streamlining your work with VMs. These guidelines apply to working with a small number of VMs. Later chapters will provide additional guidelines for working with clusters and instance groups, which are sets of similarly configured VMs.

- Choose a machine type with the fewest CPUs and the smallest amount of memory that still meets your requirements, including peak capacity. This will minimize the cost of the VM.
- Use the console for ad hoc administration of VMs. Use scripts with `gcloud` commands for tasks that will be repeated.
- Use startup scripts to perform software updates and other tasks that should be performed on startup.
- If you make many modifications to a machine image, consider saving it and using it with new instances rather than running the same set of modifications on every new instance.
- If you can tolerate unplanned disruptions, use preemptible VMs to reduce cost.
- Use SSH or RDP to access a VM to perform operating system–level tasks.
- Use Cloud Console, Cloud Shell, or Cloud SDK to perform VM-level tasks.

Summary

Google Cloud Console is a web-based graphical user interface for managing GCP resources. Cloud SDK is a command-line package that allows engineers to manage cloud resources from the command line of their local device. Cloud Shell is a web-based terminal interface to VMs. Cloud SDK is installed in Cloud Shell.

When creating a VM, you have to specify a number of parameters, including a name for the VM, a region and zone where the VM will run, a machine type that specifies the number of vCPUs and the amount of memory, and a boot disk that includes an operating system.

`gcloud` is the top-level command of the hierarchical command structure in Cloud SDK.

Common tasks when managing VMs are starting and stopping instances, using SSH to access a terminal on the VM, monitoring, and tracking the cost of the VM.

Exam Essentials

Understand how to use Cloud Console and Cloud SDK to create, start, and stop VMs. Parameters that you will need to provide when creating a VM include name, machine type, region, zone, and boot disk. Understand the need to create a VM in a project.

Know how to configure a preemptible VM using Cloud Console and the `gcloud` commands. Know when to use a preemptible VM and when not to. Know that preemptible VMs cost up to 80 percent less than nonpreemptible VMs.

Know the purpose of advanced options, including Shielded VMs and advanced boot disk configurations. Know that advanced options provide additional security. Understand the kinds of protections provided.

Know how to use `gcloud` compute instance commands to list, start, and stop VMs. Know the structure of `gcloud` commands. `gcloud` commands start with `gcloud` followed by a service, such as `compute`, followed by a resource type, such as `instances`, followed by a command or verb, like `create`, `list`, or `describe`.

Understand how to monitor a VM. Know where to find CPU utilization, network monitoring, and disk monitoring in the VM Instances pages of the console. Know the difference between listing and describing instances with a `gcloud` command.

Know the factors that determine the cost of a VM. Know that Google charges by the second with a 1-minute minimum. Understand that the costs of a machine type may be different in different locations. Know that cost is based on the number of vCPUs and memory.

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.

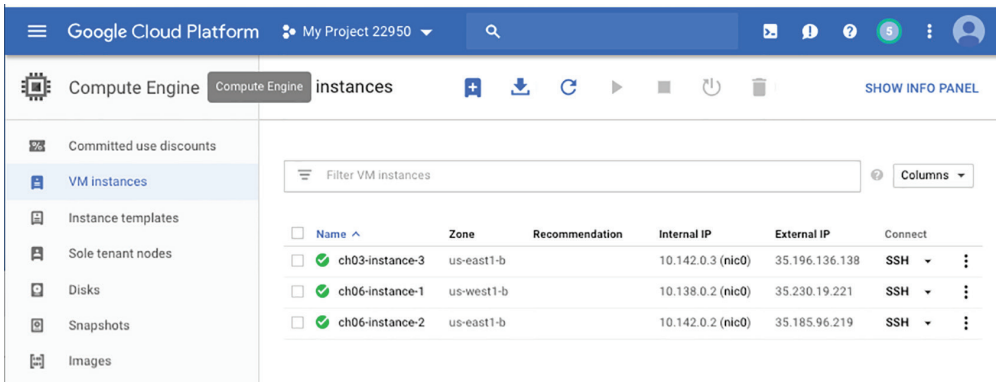
Managing Single Virtual Machine Instances in the Console

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

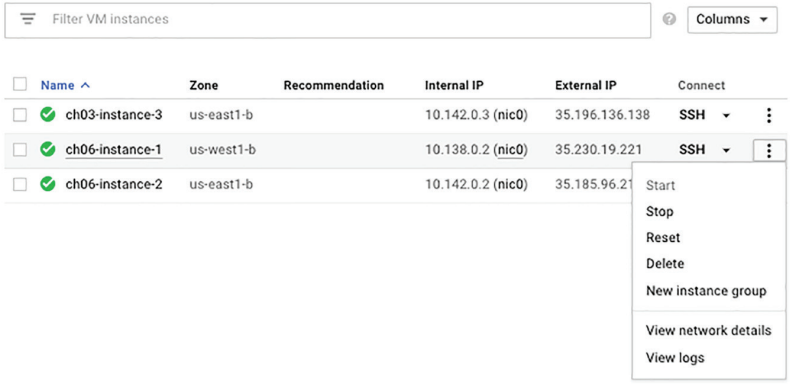
To start working, open the console and select Compute Engine. Then select VM instances. This will display a window such as in Figure 6.1, but with different VMs listed. In this example, there are three VMs.

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



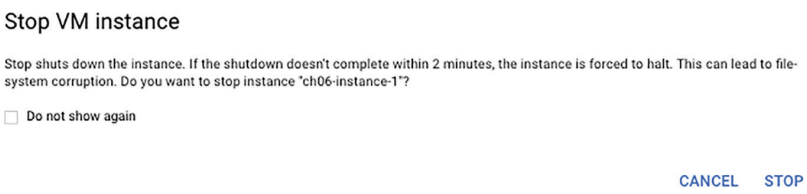
The three instances in Figure 6.1 are all running. You can stop the instances by clicking the three-dot icon on the right side of the line listing the VM attributes. This action displays a list of commands. Figure 6.2 shows the list of commands available for `instancech06-instance-1`.

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

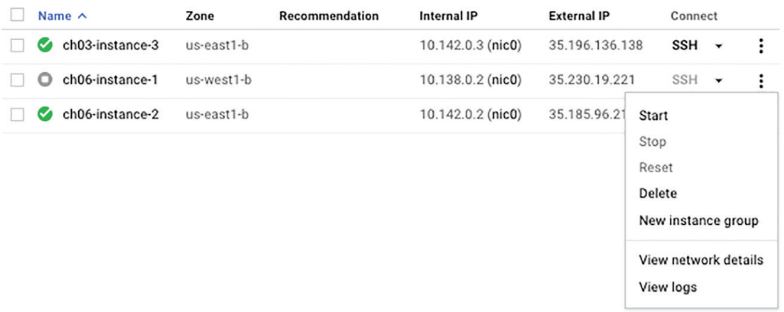


When you stop a VM, the green check mark on the left changes to a gray circle with a white square, and the SSH option is disabled, as shown in Figure 6.4.

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.

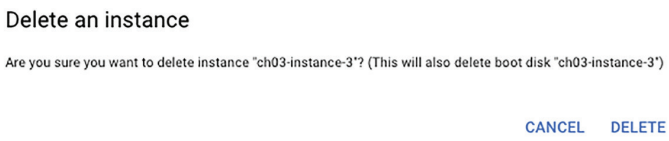


The Reset command restarts a VM. The properties of the VM will not change, but data in memory will be lost.

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.

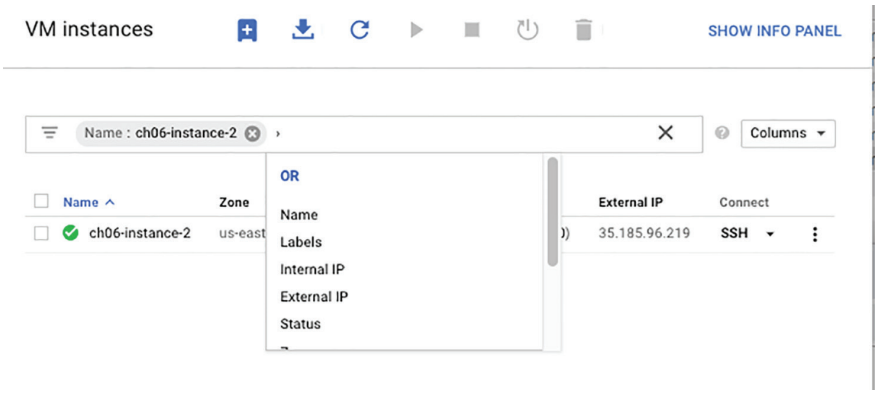
FIGURE 6.6 Deleting an instance from the console will display a warning message such as this.



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

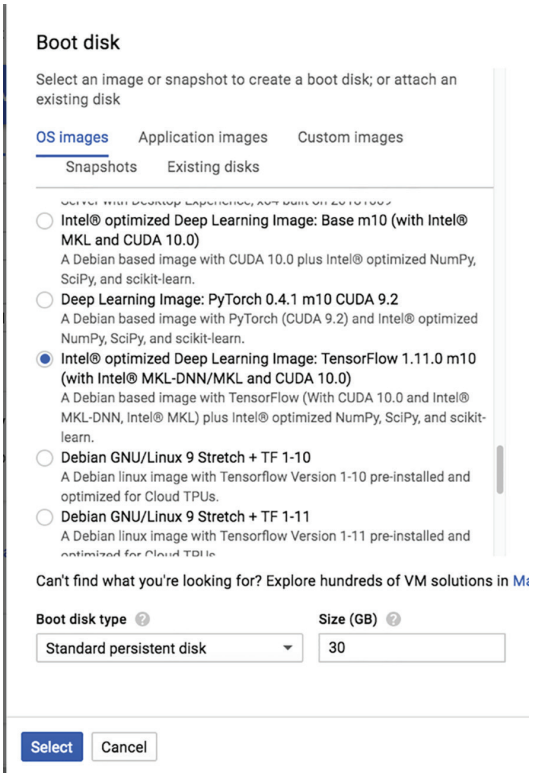
If you set multiple filter conditions, then all must be true for a VM to be listed unless you explicitly state the `OR` operator.

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.

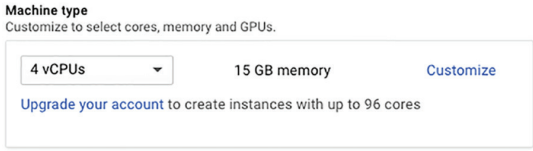
To add a GPU to an instance, you must start an instance in which GPU libraries have been installed or will be installed. For example, you can use one of the Google Cloud Platform (GCP) images that has GPU libraries installed, including the Deep Learning images, as shown in Figure 6.8. You must also verify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



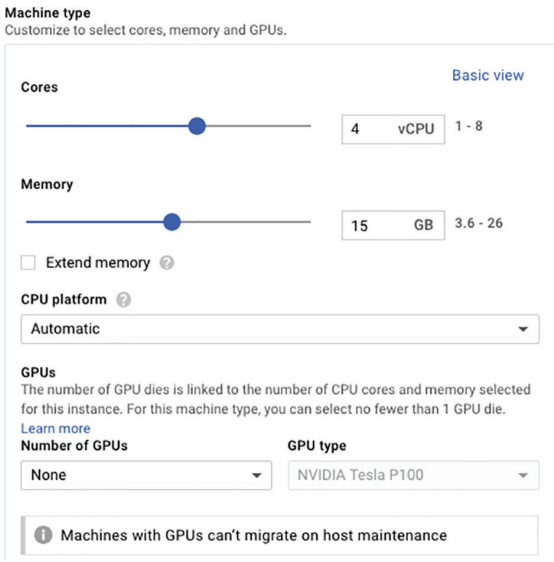
You will also need to customize the configuration for the machine type; Figure 6.9 shows the form.

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

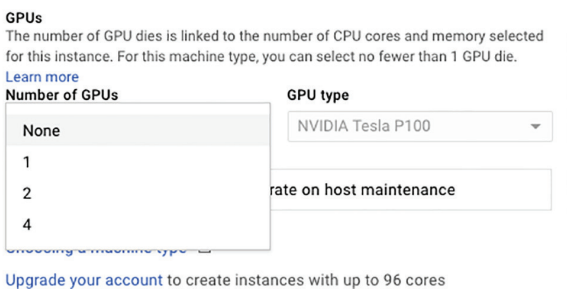
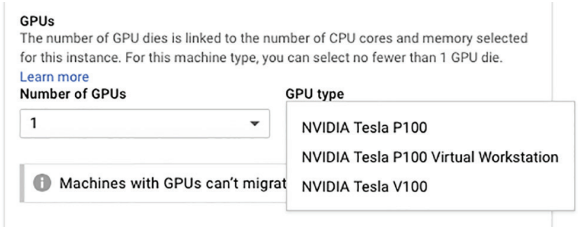


FIGURE 6.12 Selecting the type of GPUs to attach to the VM



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

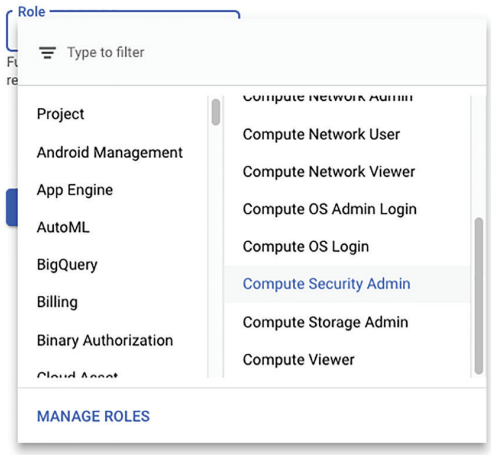
Snapshots are copies of data on a persistent disk. You use snapshots to save data on a disk so you can restore it. This is a convenient way to make multiple persistent disks with the same data.

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.

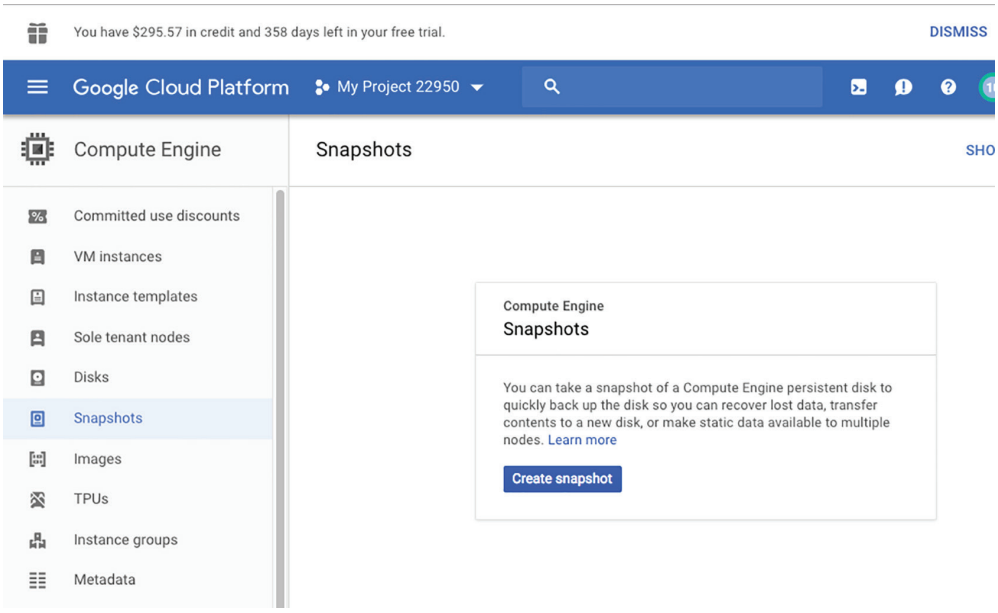
To work with snapshots, a user must be assigned the Compute Storage Admin role. Go to the Identity Access Management (IAM) page, select Roles, and then specify the email address of a user to be assigned the role. Select the role from the list of roles, as shown in Figure 6.14.

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



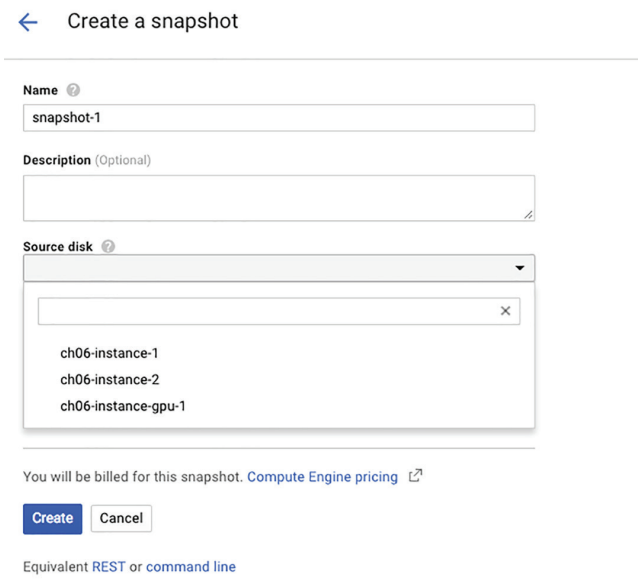
To create a snapshot from Cloud Console, display the Compute Engine options and select Snapshots from the left panel, as shown in Figure 6.15.

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.

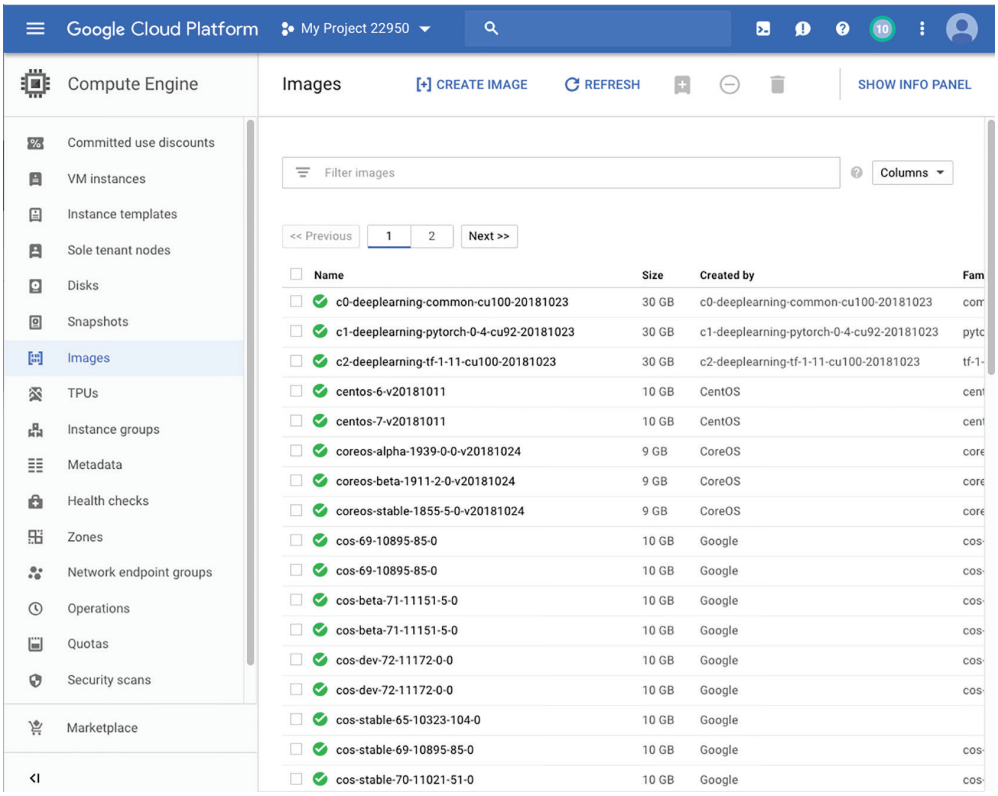
Working with Images

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

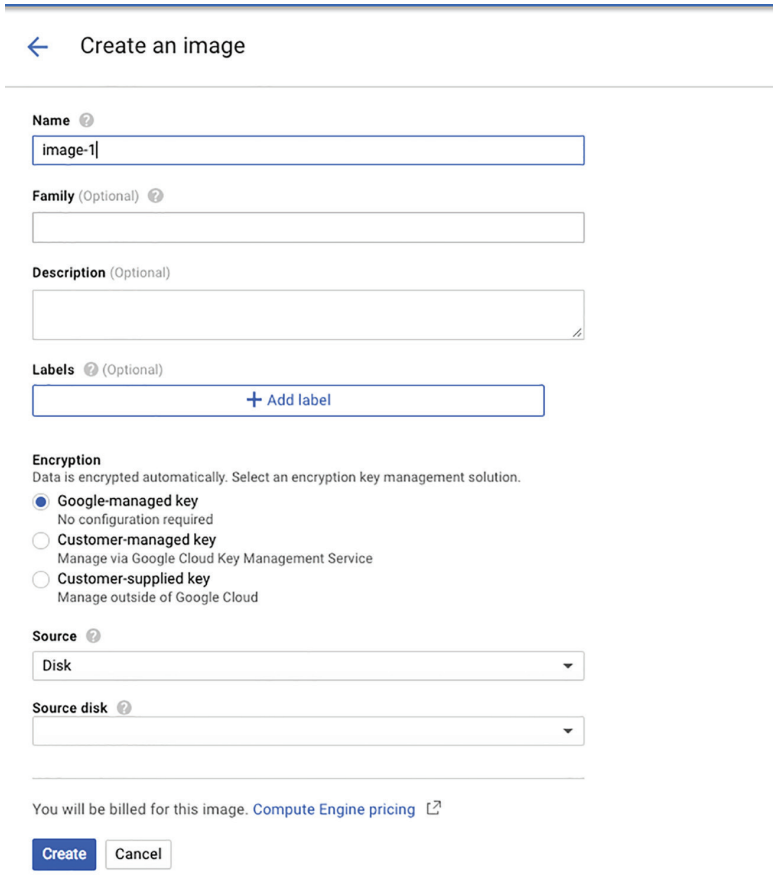
To create an image, choose the Image option from the Compute Engine page in Cloud Console, as shown in Figure 6.17. This lists available images.

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



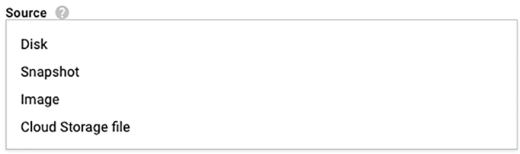
Select Create Image to show the form in Figure 6.18. This allows you to create a new image by specifying name, description, and labels. Images have an optional attribute called Family, which allows you to group images. When a family is specified, the latest, nondeprecated image in the family is used.

FIGURE 6.18 Cloud Console form for creating an image



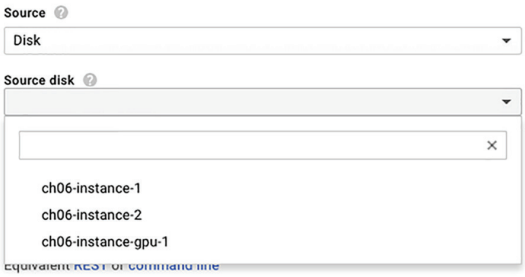
The form provides a drop-down list of options for the source of the image, as shown in Figure 6.19.

FIGURE 6.19 Options for the source of an image



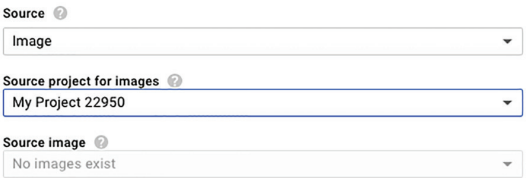
When Disk is selected as the source, you can choose from disks on VMs, as shown in Figure 6.20.

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



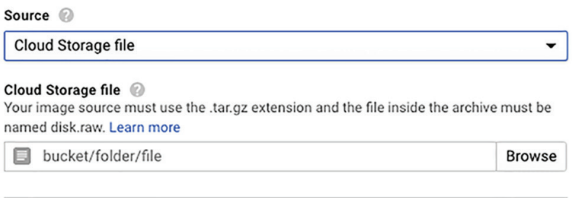
When you choose Image as the source type, you can choose an image from the current project or other projects (see Figure 6.21).

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



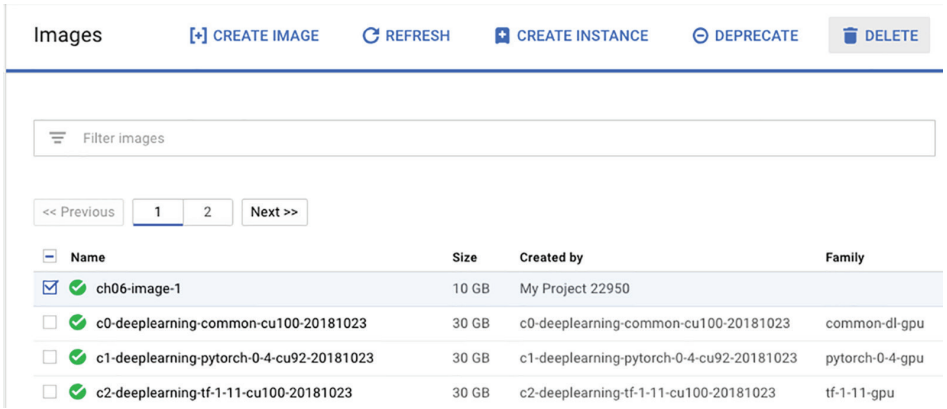
If you choose a Cloud Storage file as a source, you can browse your Cloud Storage bucket to find a file to use as the source (see Figure 6.22).

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:

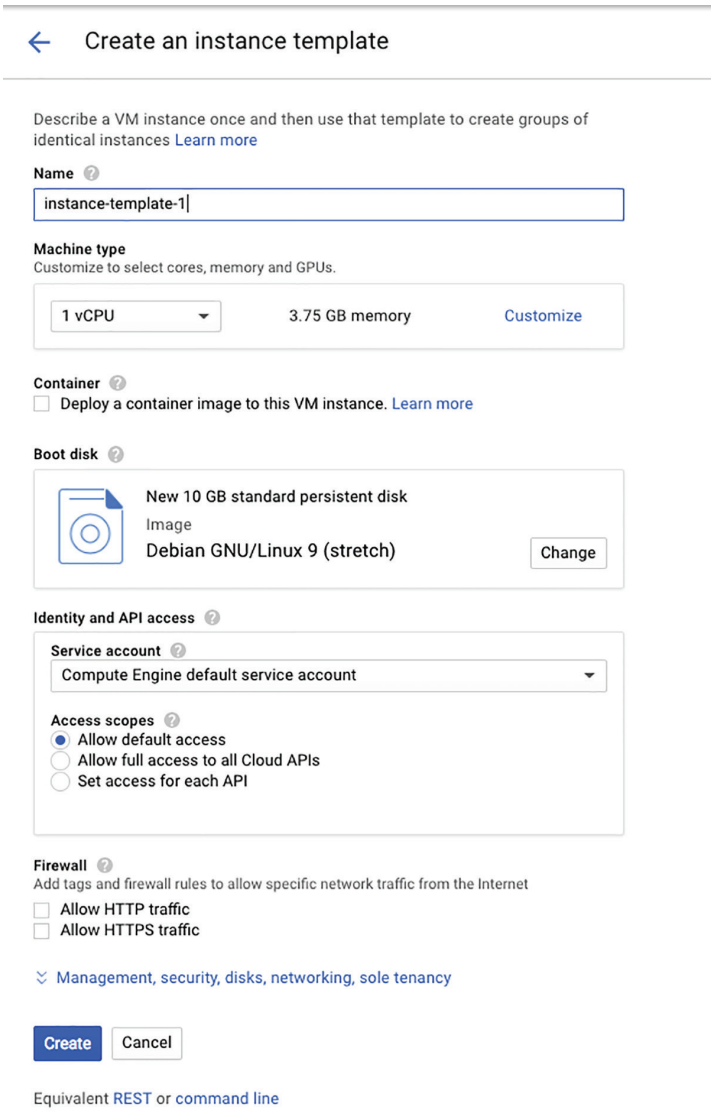
```
gcloud compute instance-templates create INSTANCE
```

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.

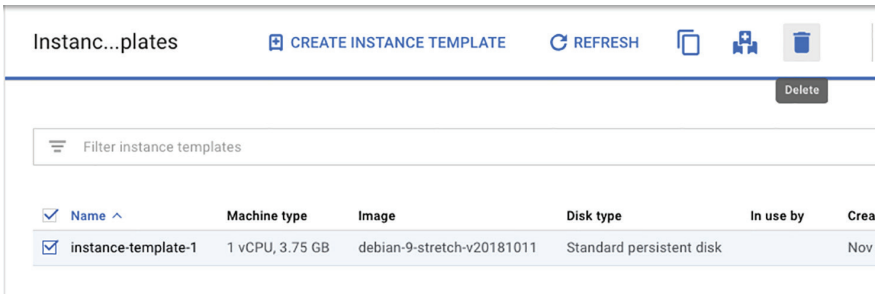


Instance groups can contain instances in a single zone or across a region. The first is called a *zonal* managed instance group, and the second is called a *regional* managed

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.



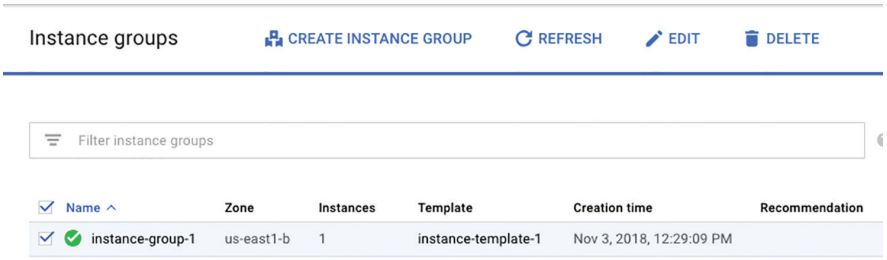
You can also delete an instance group template using the following command:

```
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.

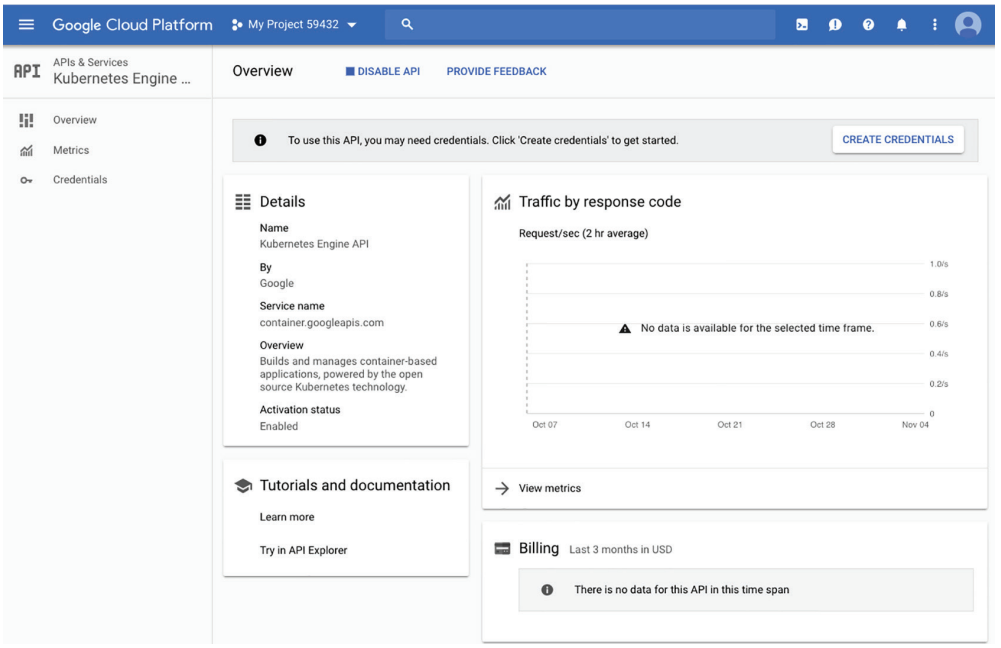
Deploying Kubernetes Clusters

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

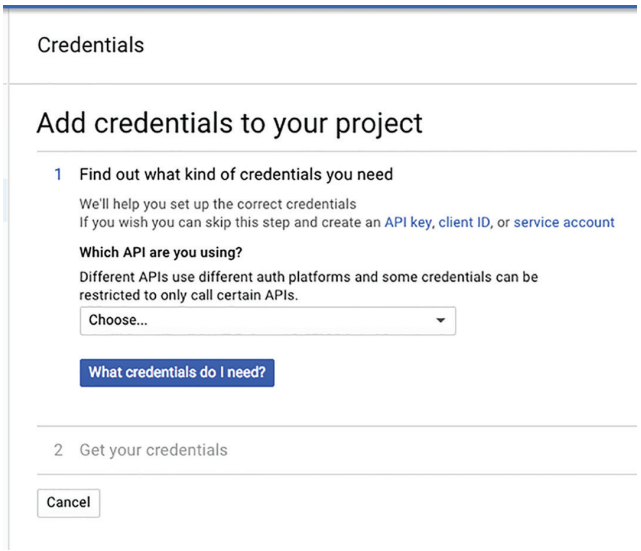
To use Kubernetes Engine, you will need to enable the Kubernetes Engine API. Once you have enabled the API, you can navigate to the Kubernetes Engine page in Cloud Console. Figure 7.1 shows an example of the Overview page.

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



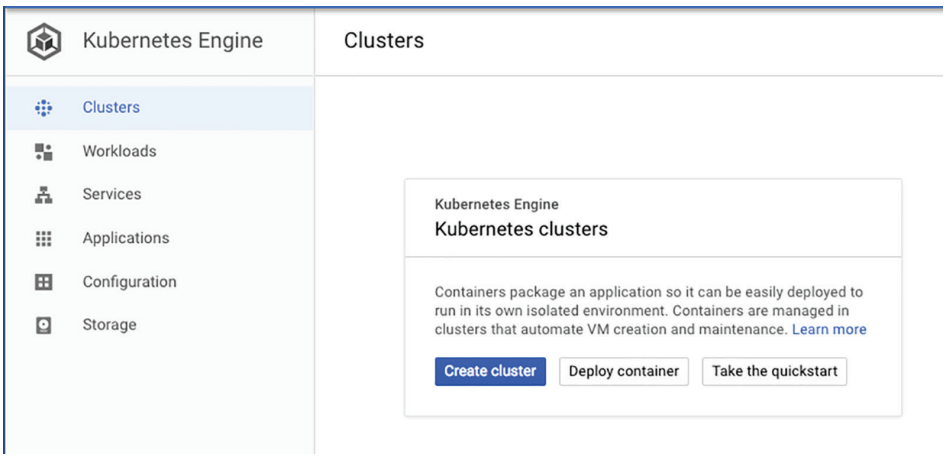
The first time you use Kubernetes Engine, you may need to create credentials. You can do this by clicking the Create Credentials button near the top of the Overview page. A form such as the one shown in Figure 7.2 will appear. You can specify which API you are using and then generate your credentials.

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



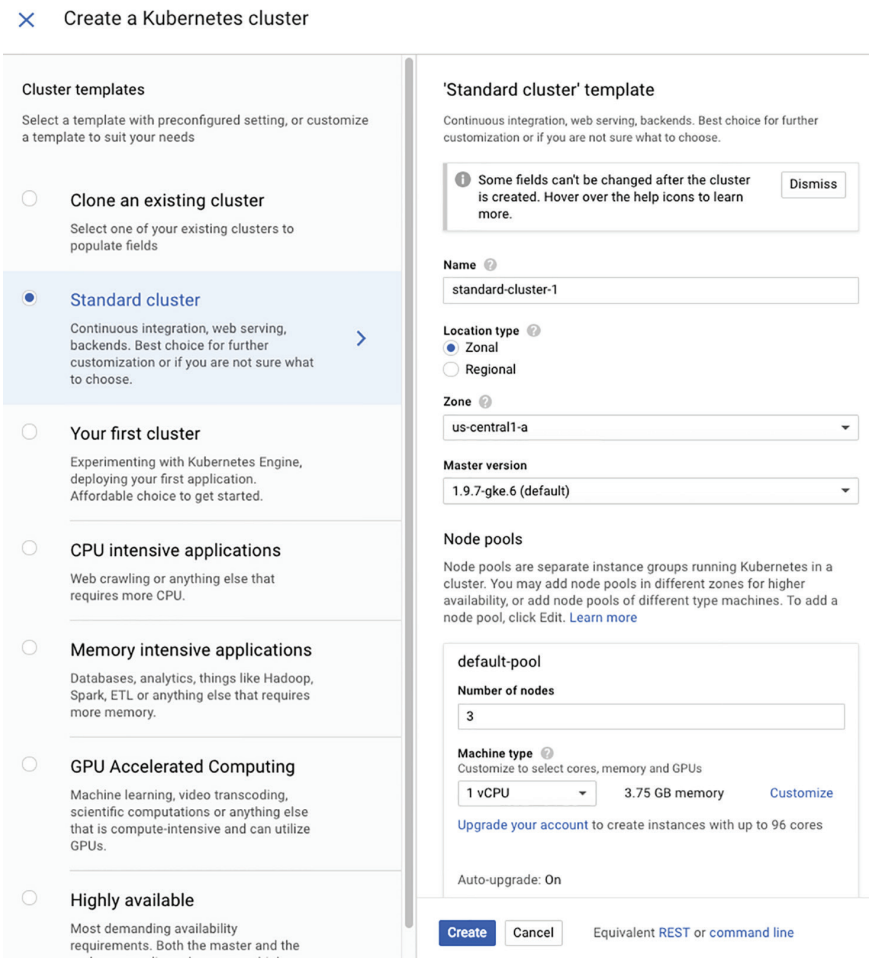
After creating credentials, if needed, you can create a cluster. Figure 7.3 shows the first page in the cluster creation step.

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.

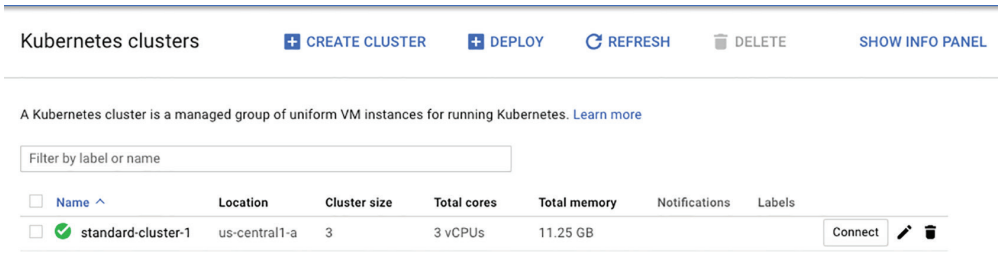
FIGURE 7.4 Templates for creating a Kubernetes cluster



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.

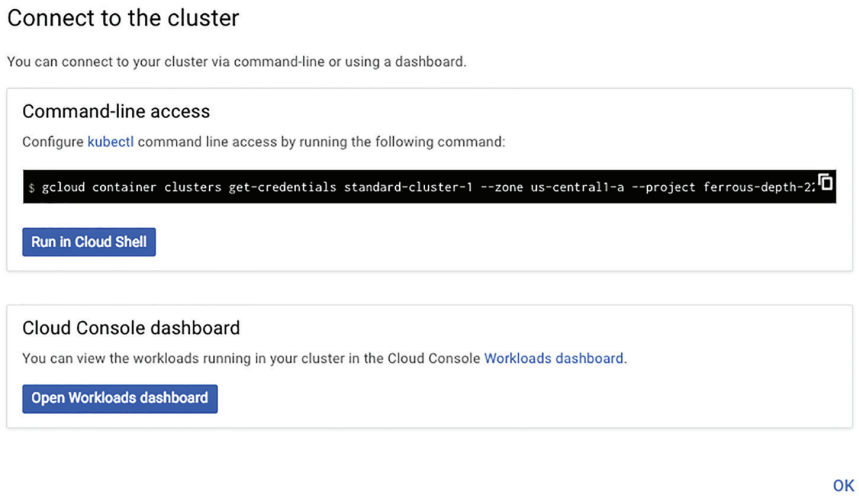
It can take a few minutes to create a cluster. When the cluster is created, it will appear in the list of clusters, as in Figure 7.5.

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



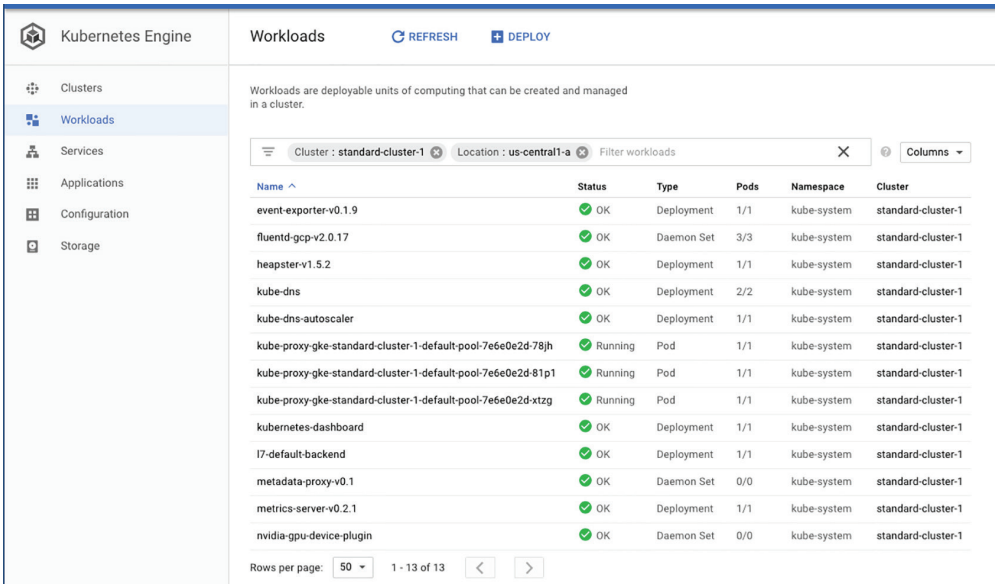
From the listing of clusters, you can edit, delete, and connect to a cluster. When you click Connect, you receive a `gcloud` command to connect to the cluster from the command line. You also have the option of viewing the Workloads page, as shown in Figure 7.6.

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.

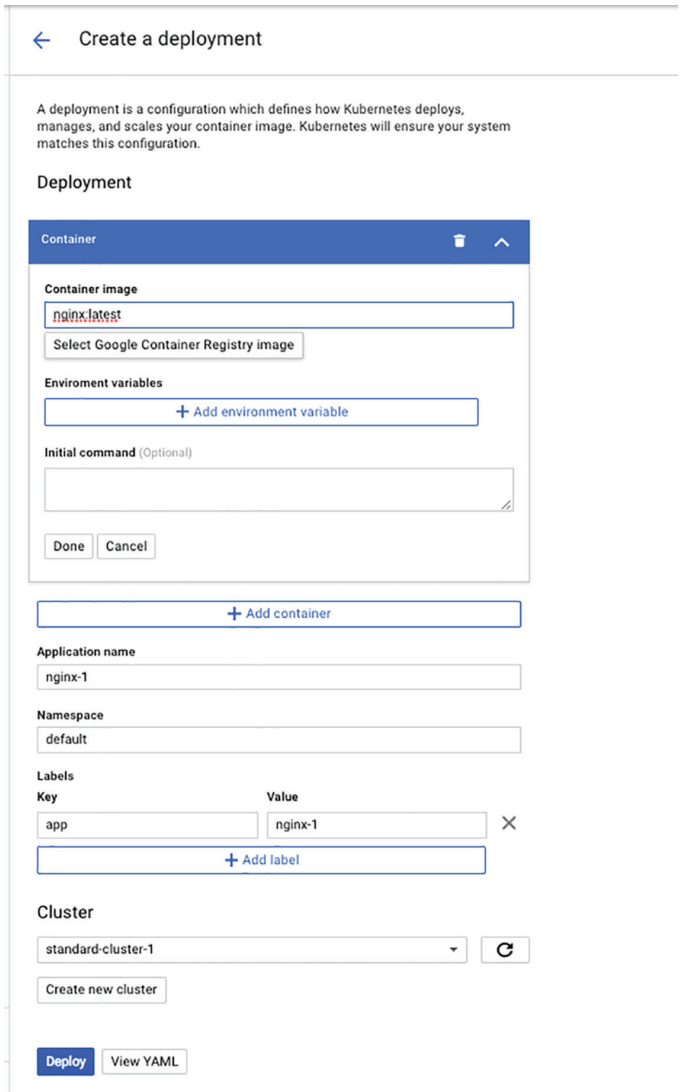
Deploying Application Pods

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:

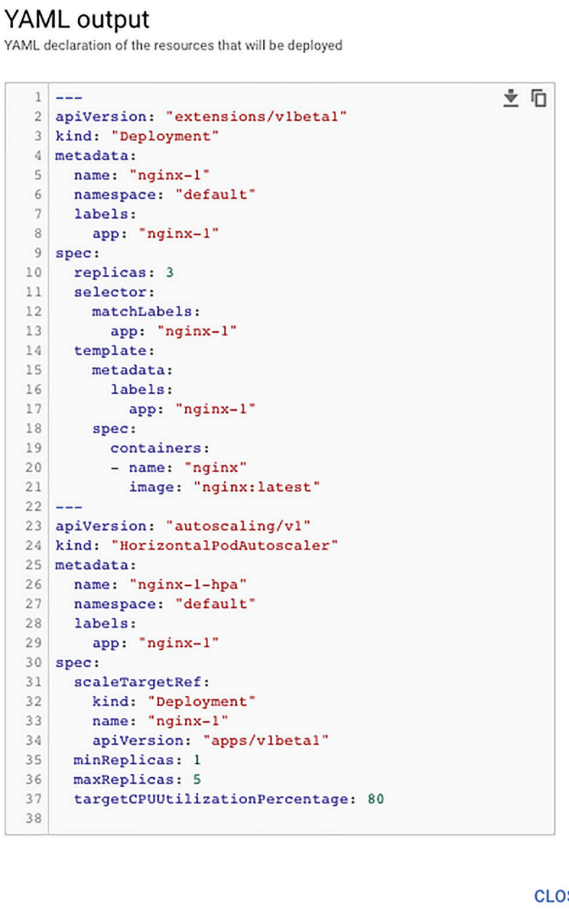
- Container image
- Environment variables
- Initial command
- Application name
- Labels
- Namespace
- Cluster to deploy to

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
```

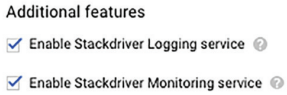
This example would create five replicas.

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.

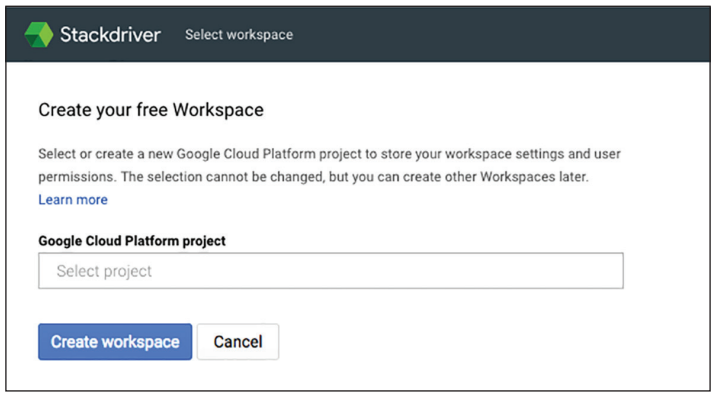
FIGURE 7.10 Expanding the Advanced Options in the Create Cluster dialog will show two check boxes for enabling Stackdriver logging and monitoring.



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.

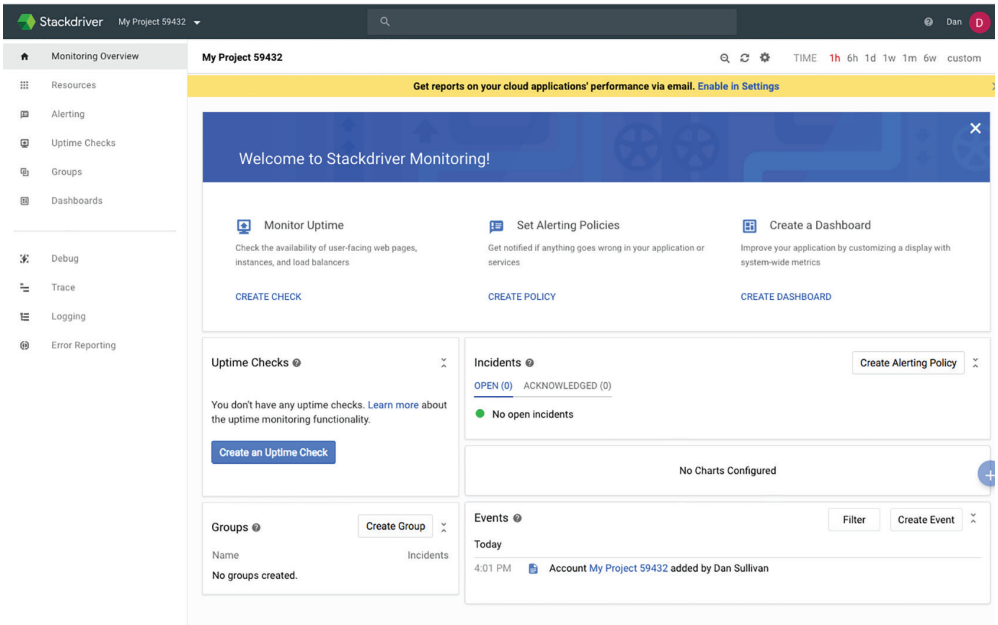
Workspaces are resources for monitoring and can support up to 100 monitored projects. Workspaces contain dashboards, alerting policies, group definitions, and notification checks.

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



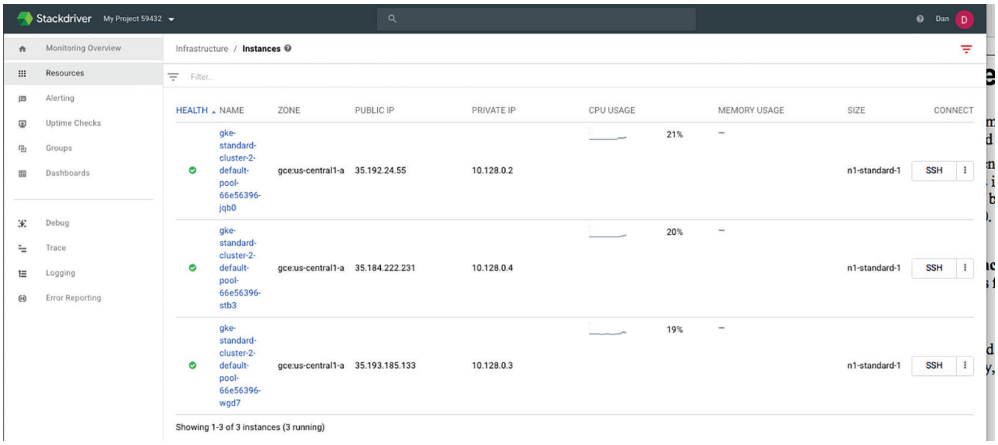
After you create a workspace, open Stackdriver, and it displays the Monitoring Overview page, shown in Figure 7.12.

FIGURE 7.12 The Stackdriver Monitoring Overview page



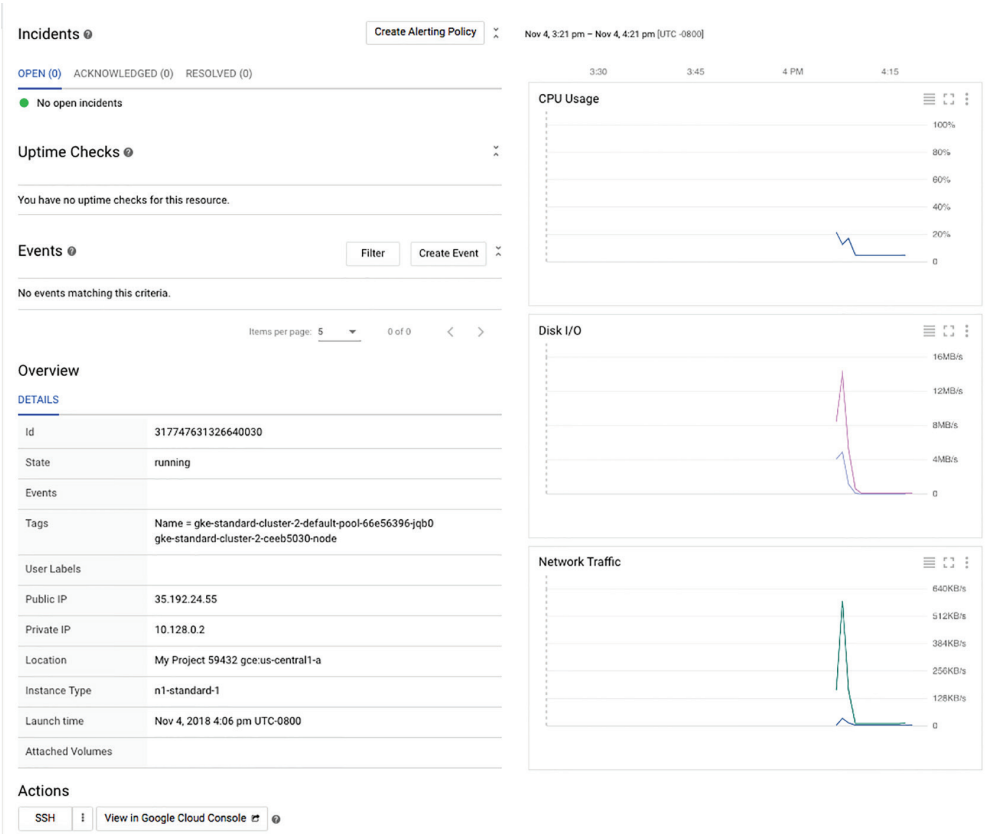
From the Overview Page, click Resources and select Instances to list the instances in your cluster. This displays a list such as in Figure 7.13.

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.

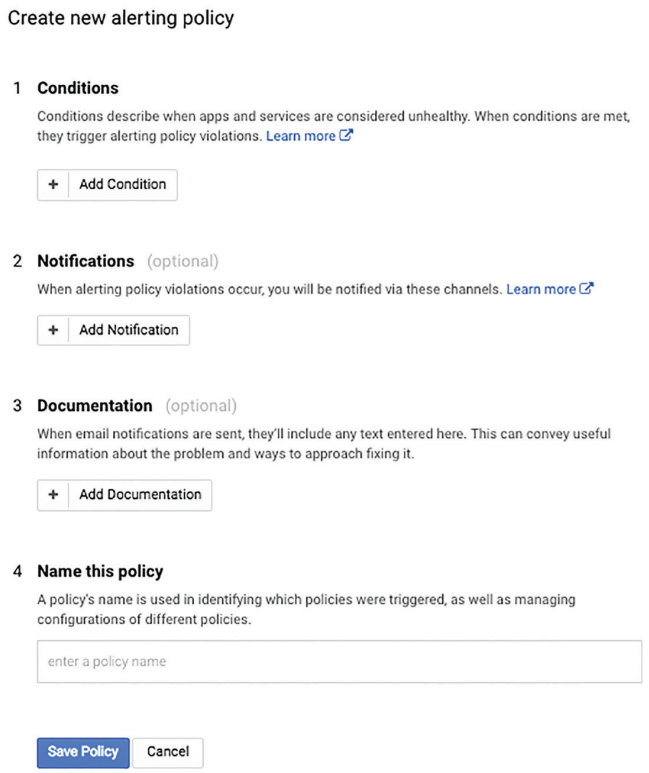
FIGURE 7.14 A typical detailed monitoring page of an instance running in a Kubernetes cluster



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.

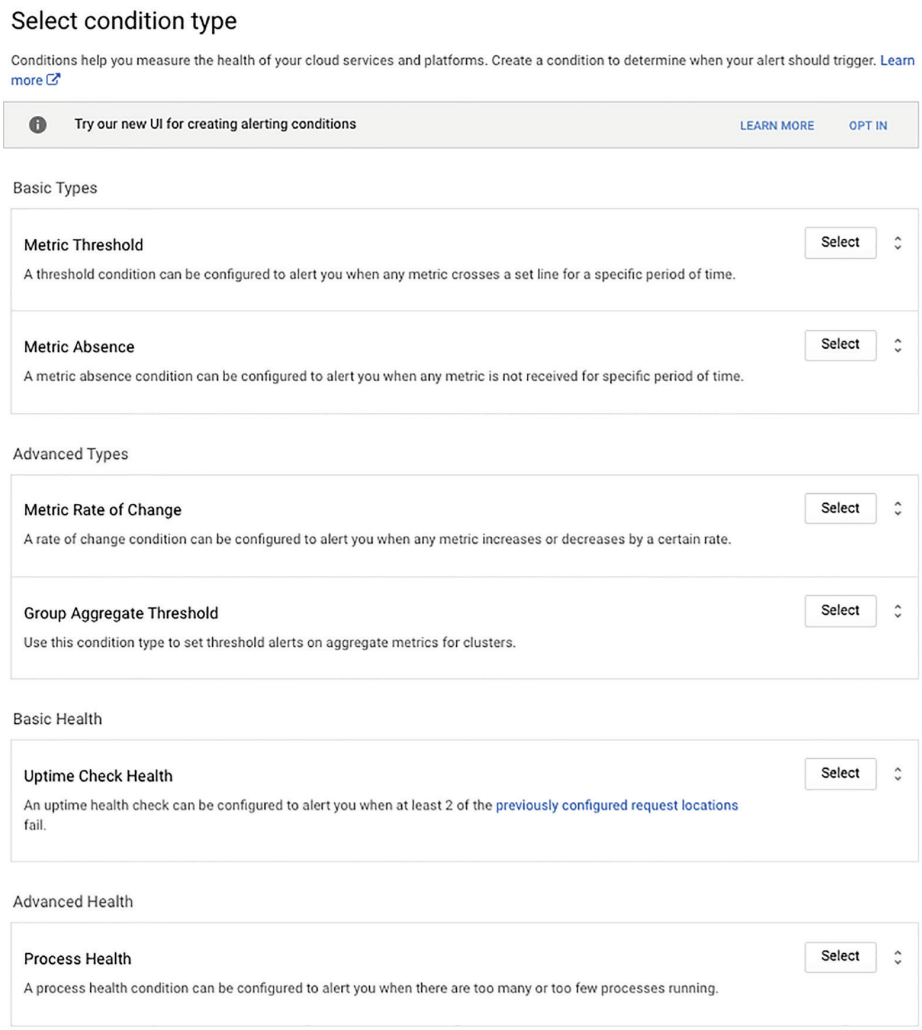
In the detail Stackdriver page, create an alert by clicking the Create Alerting Policy button. This displays a dialog such as in Figure 7.15, from which you can create conditions, notifications, and documentation. You can also name the policy.

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



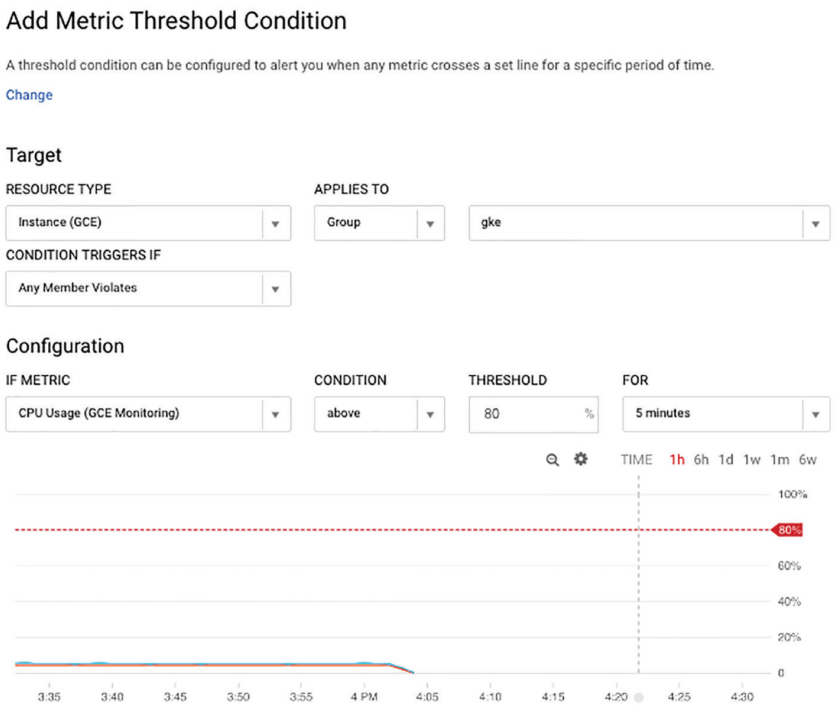
When you add a condition, a form such as the one shown in Figure 7.16 appears.

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.

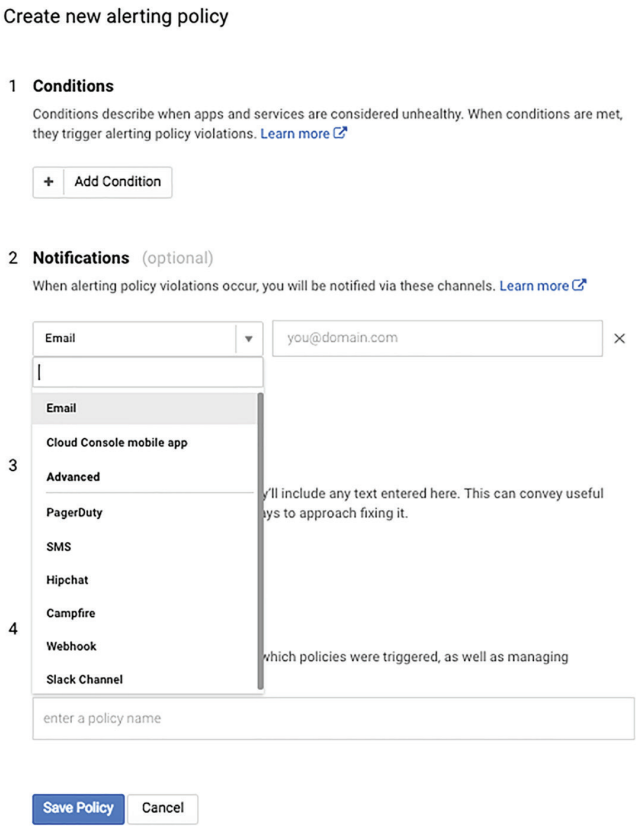


Stackdriver will need to know how to notify you if an alert is triggered. You can specify your choice of notification channels in the Create New Alerting Policy form, as shown in Figure 7.18. Channels include email, webhooks, and SMS text messaging as well as third-party tools such as PagerDuty, Campfire,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

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

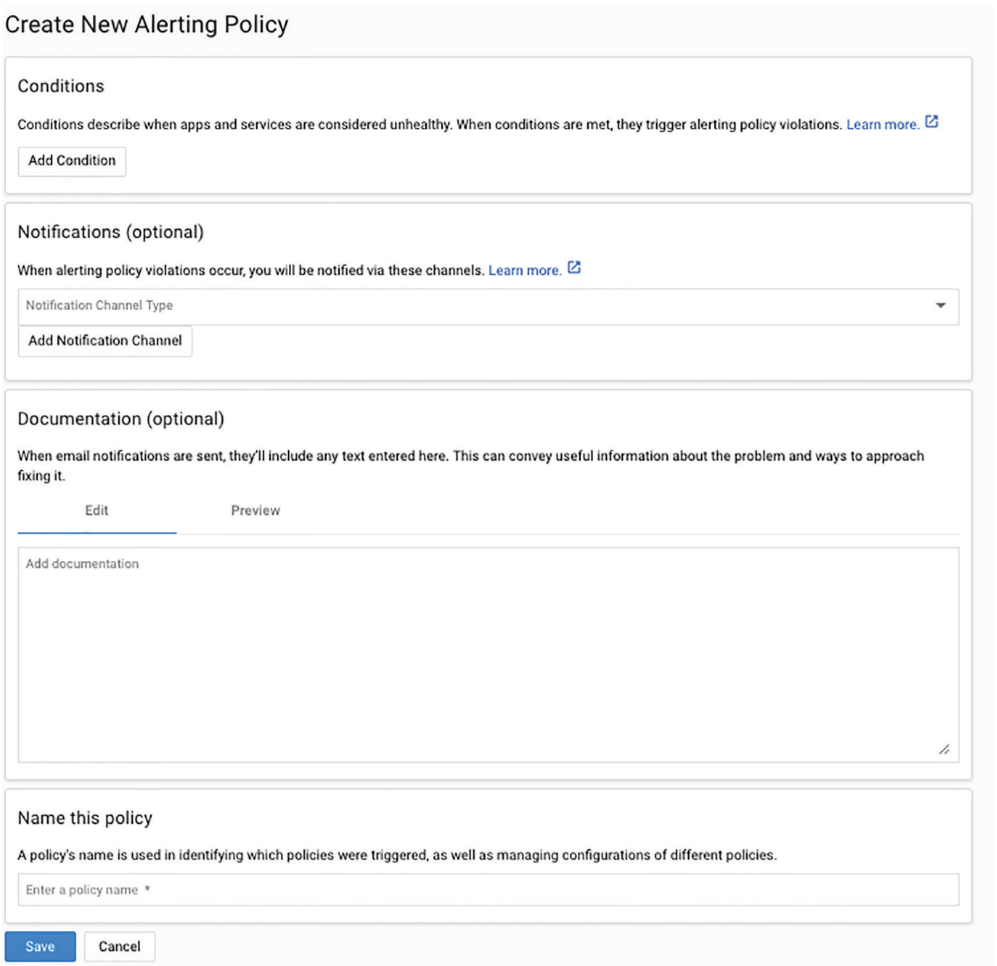
Let's walk through an example of creating a policy to monitor CPU utilization.

FIGURE 7.18 Stackdriver supports a number of condition types.



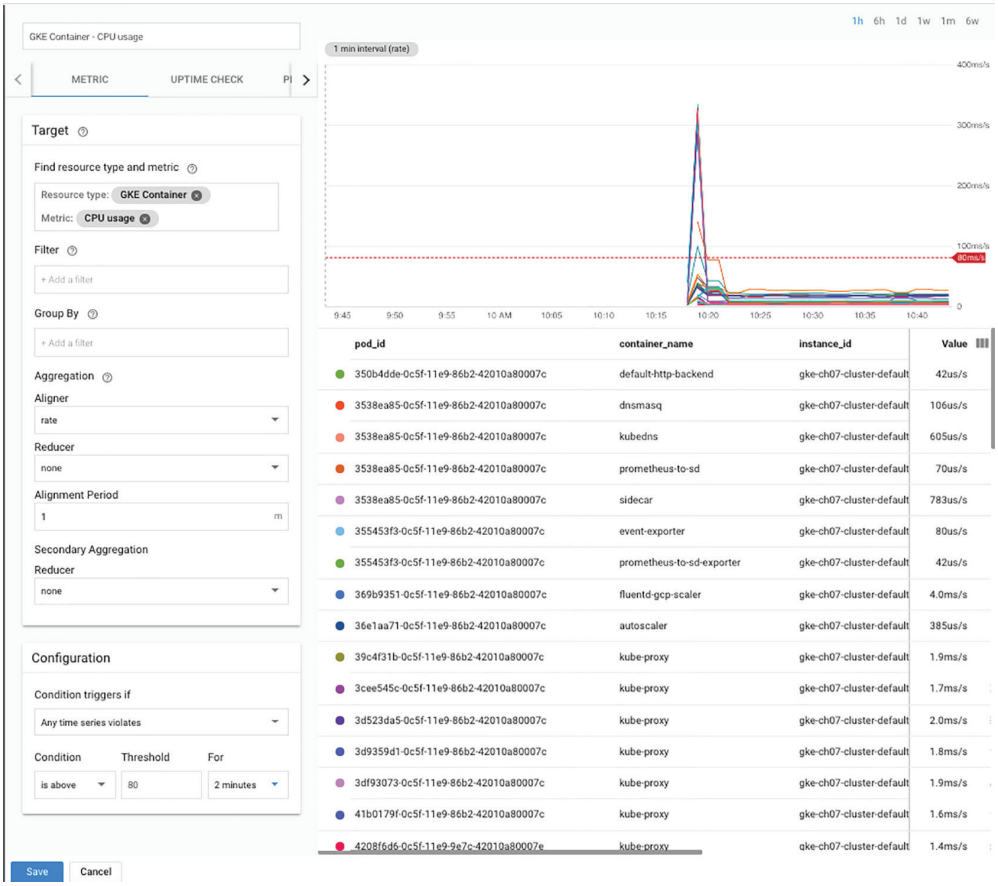
For more details on monitoring, see Chapter 18. To create a policy to monitor CPU utilization, navigate to the monitoring page in Stackdriver and click Create Policy. This will display the form to create a policy, which is a four-step process: create a condition, specify a notification channel, add a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

FIGURE 7.19 Creating a policy to monitor CPU utilization



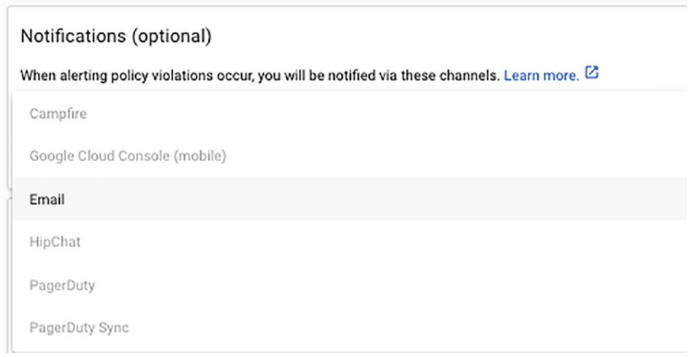
Click Add Condition to display a form like that shown in Figure 7.20.

FIGURE 7.20 Adding a condition to a policy



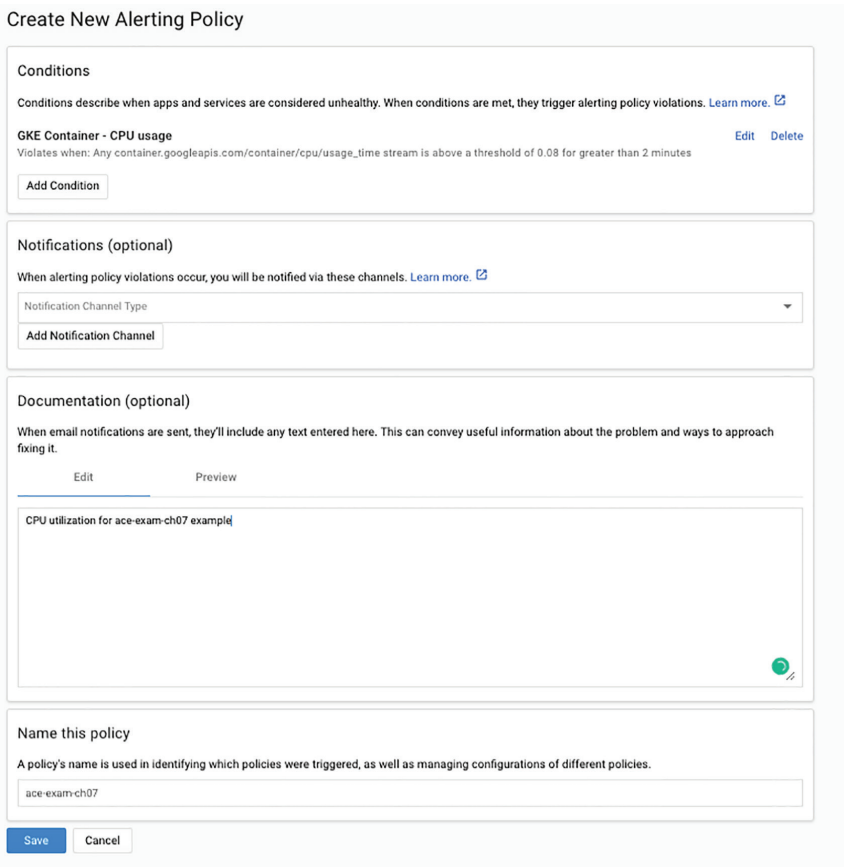
In the Filter parameter, enter **GKE Container** and **CPU Usage**. In the Configuration section, specify 80 percent as the threshold and 2 minutes as the time period. Save the condition. This will return to the Create Policy form. In the Notification parameter, select Email from the drop-down list,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

FIGURE 7.21 Choosing a notification channel



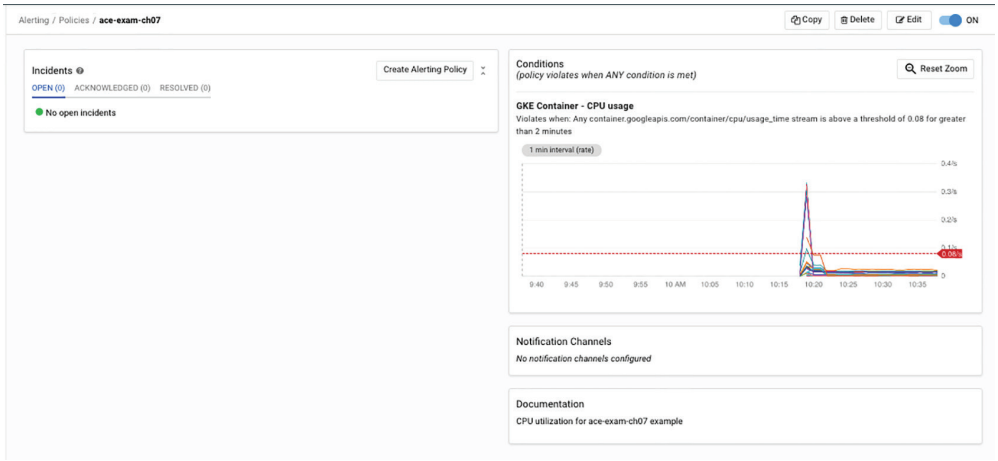
Add a description and policy name, as shown in Figure 7.22.

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.

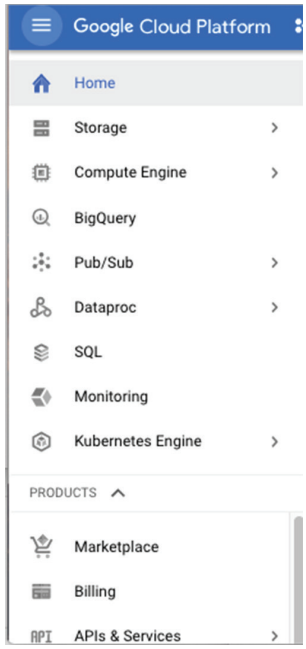
Viewing the Status of a Kubernetes Cluster

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

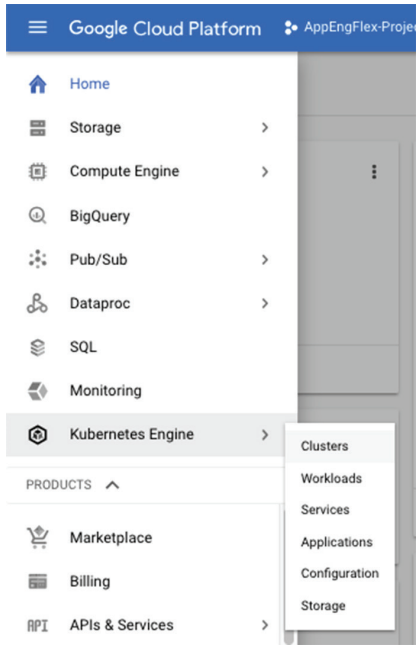
Starting from the Cloud Console home page, open the navigation menu by clicking the three stacked lines icon in the upper-left corner. This displays the list of GCP services, as shown in Figure 8.1.

FIGURE 8.1 Navigation menu in Google Cloud Console



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

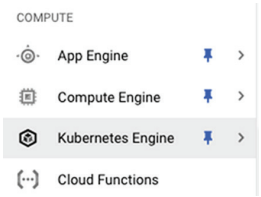
FIGURE 8.2 Selecting Kubernetes Engine from the navigation menu



Pinning Services to the Top of the Navigation Menu

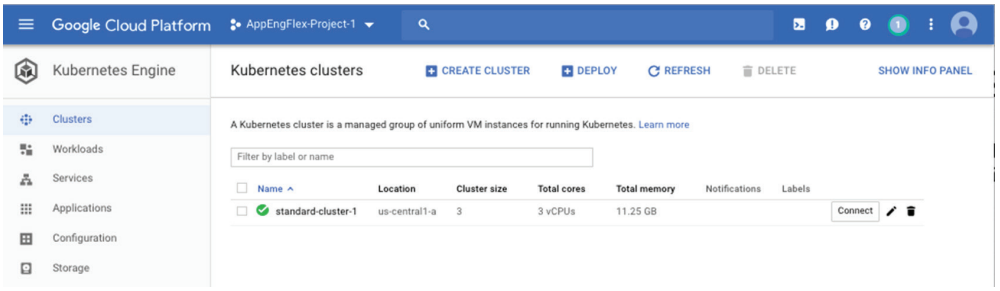
In Figure 8.2, Kubernetes Engine has been *"pinned"* so it is displayed at the top. You can pin any service in the navigation menu by mousing over the product and clicking the pin icon that appears, as in Figure 8.3. In that figure, Compute Engine and Kubernetes Engine are already pinned, and Cloud Functions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

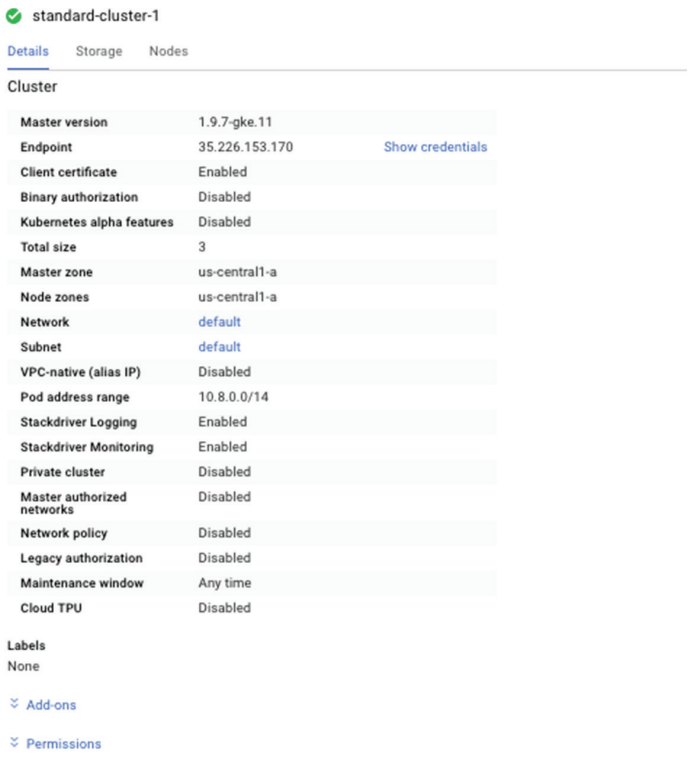
FIGURE 8.4 Example list of clusters in Kubernetes Engine



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

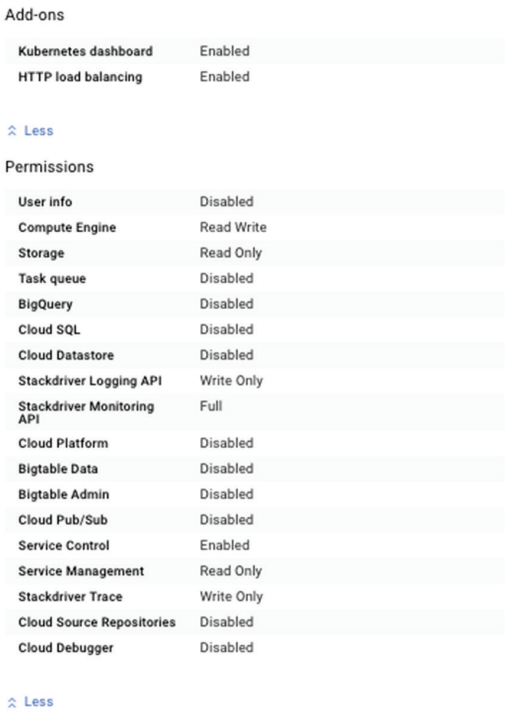


Figure 8.8 shows example details of node pools, which are separate instance groups running in a Kubernetes clusters. The details in this section include the node image running on the nodes, the machine type, the total number of vCPUs (listed as Total Cores), the disk type, and whether the nodes are preemptible.

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

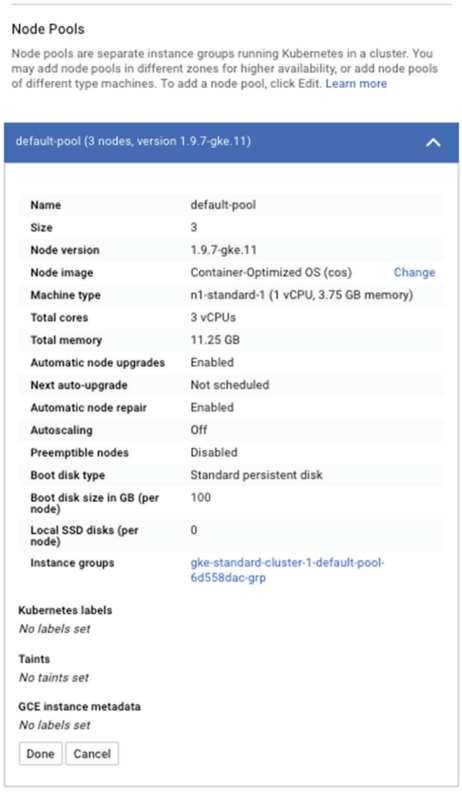
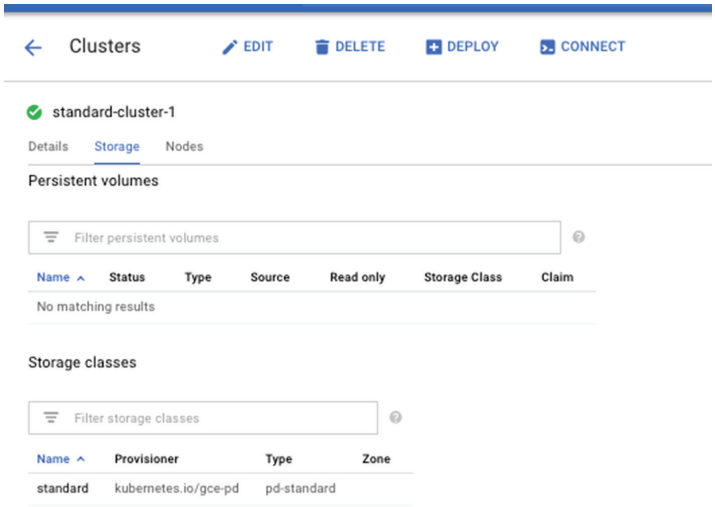
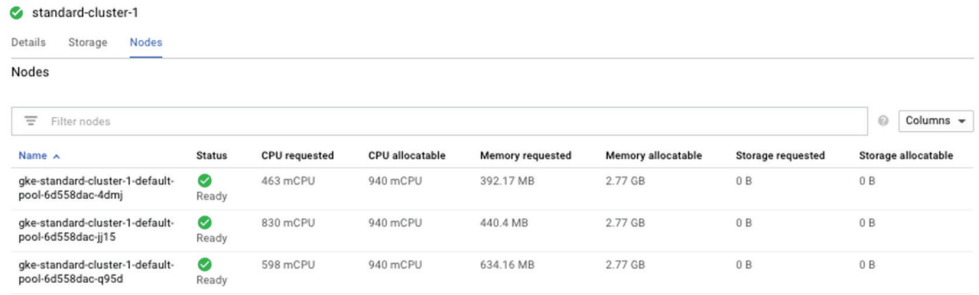


FIGURE 8.9 Storage information about a cluster



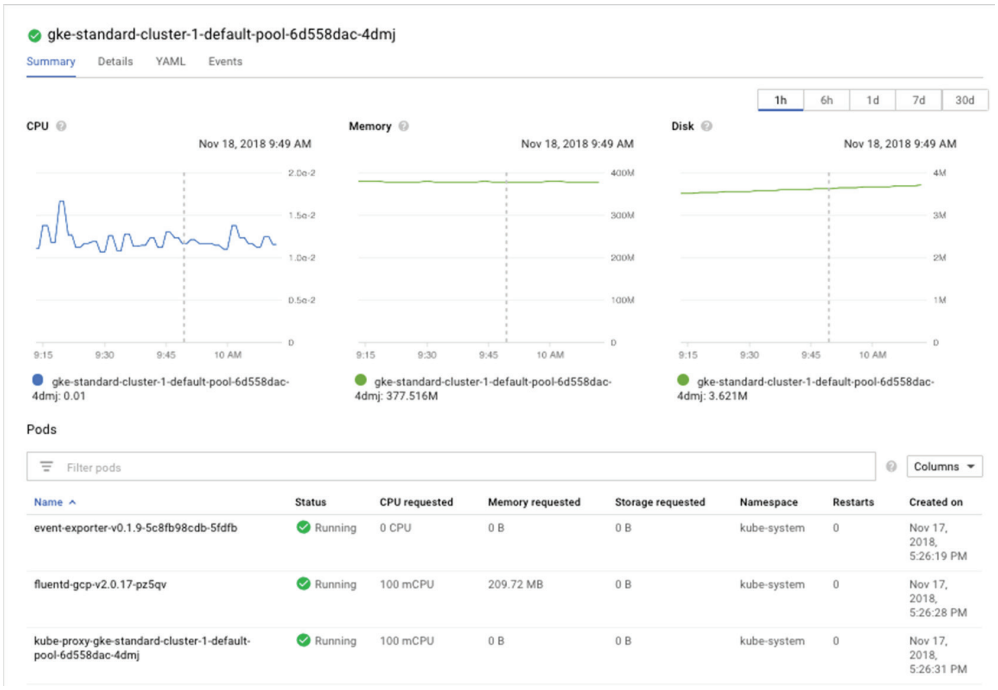
Under the Nodes option of the cluster status menu, you can see a list of nodes or VMs running in the cluster, as shown in Figure 8.10. The nodes list shows basic configuration information.

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.

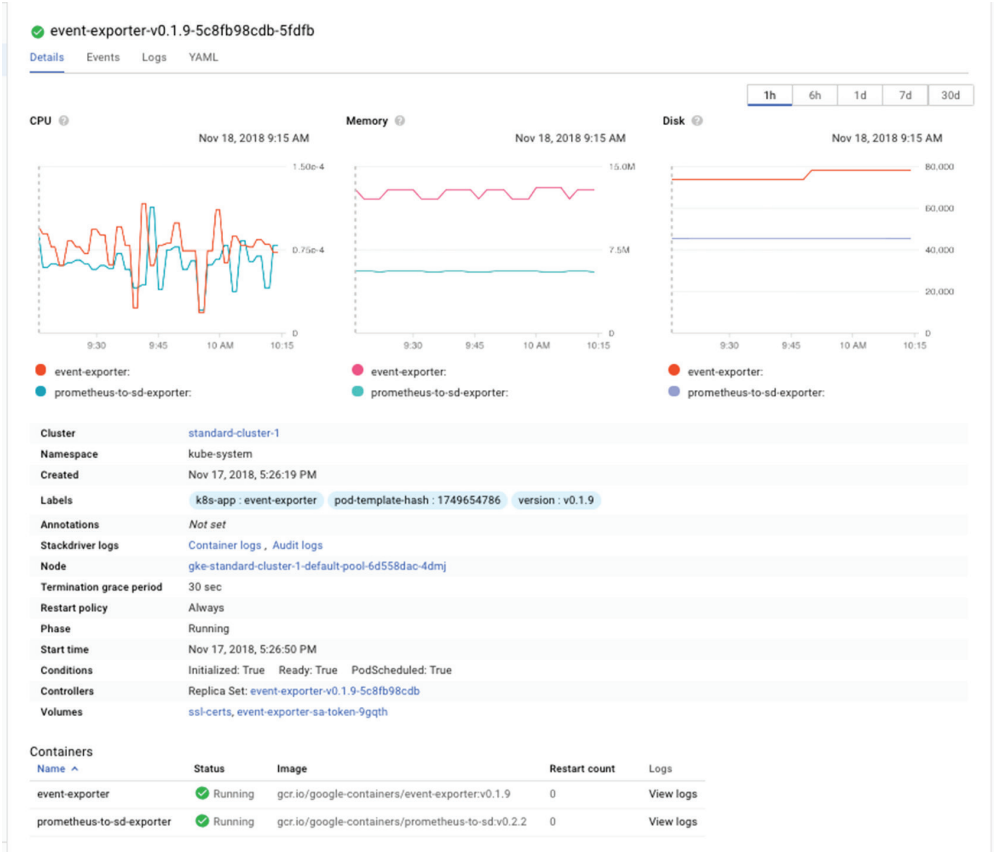
FIGURE 8.11 Example details of a node running in a Kubernetes cluster



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).

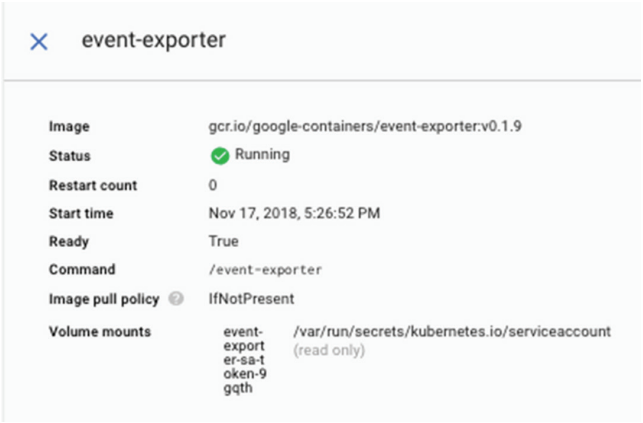
Other possible statuses are Pending, which indicates the pod is downloading images; Succeeded, which indicates the pod terminated successfully; Failed, which indicates at least one container failed; and Unknown, which means the master cannot reach the node and status cannot be determined.

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

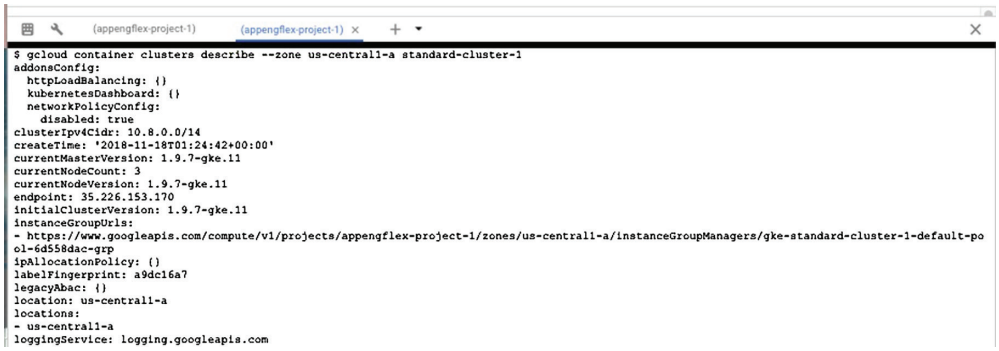


FIGURE 8.16 Part 2 of the information displayed by the `gcloud container clusters describe` command

```
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
```

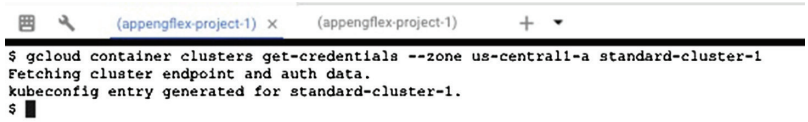
To list information about nodes and pods, use the `kubectl` command.

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

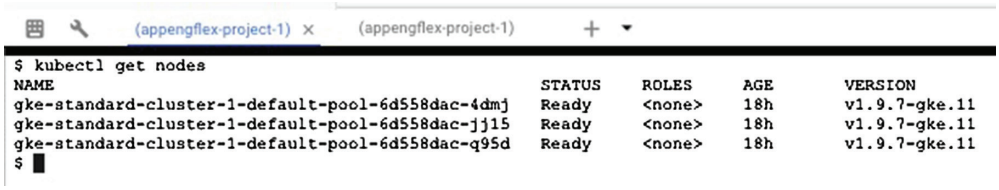


You can list the nodes in a cluster using the following:

```
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

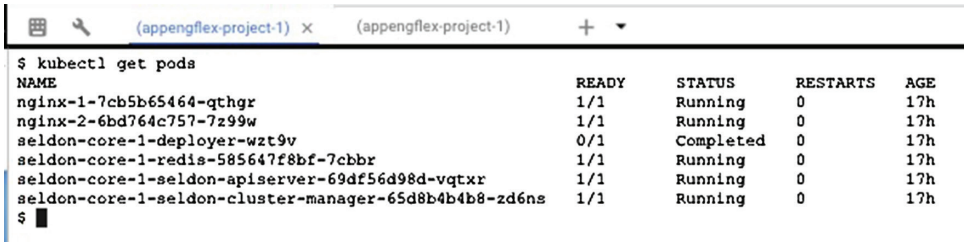


Similarly, to list pods, use the following command:

```
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

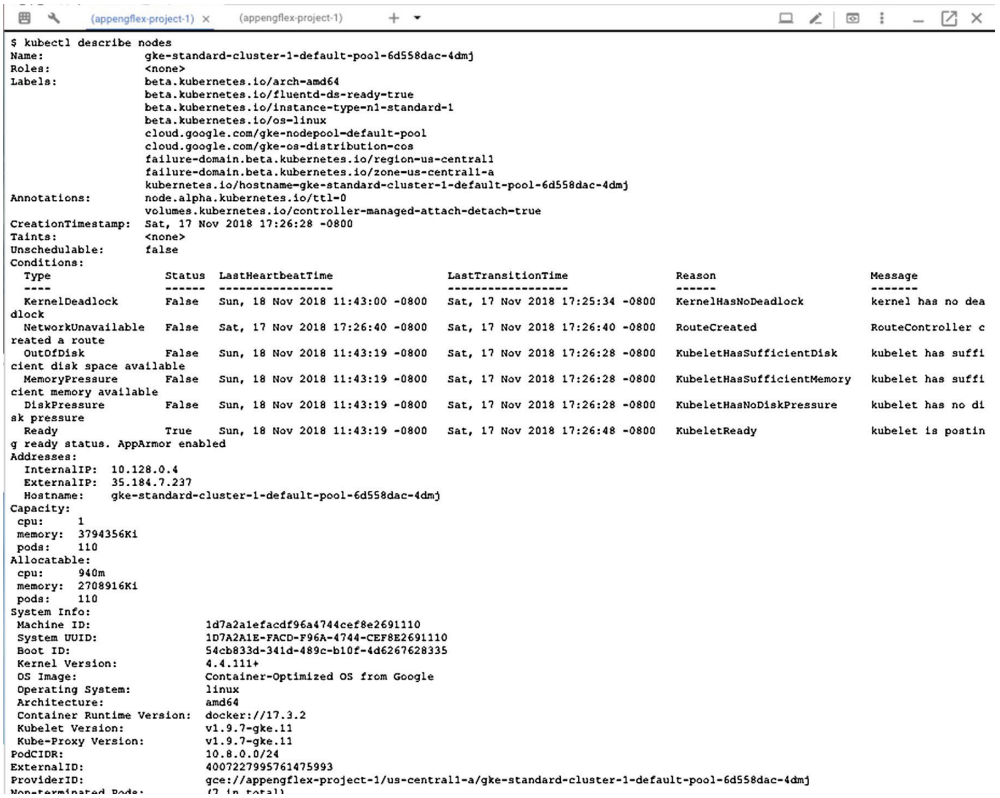


For more details about nodes and pods, use these commands:

```
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.

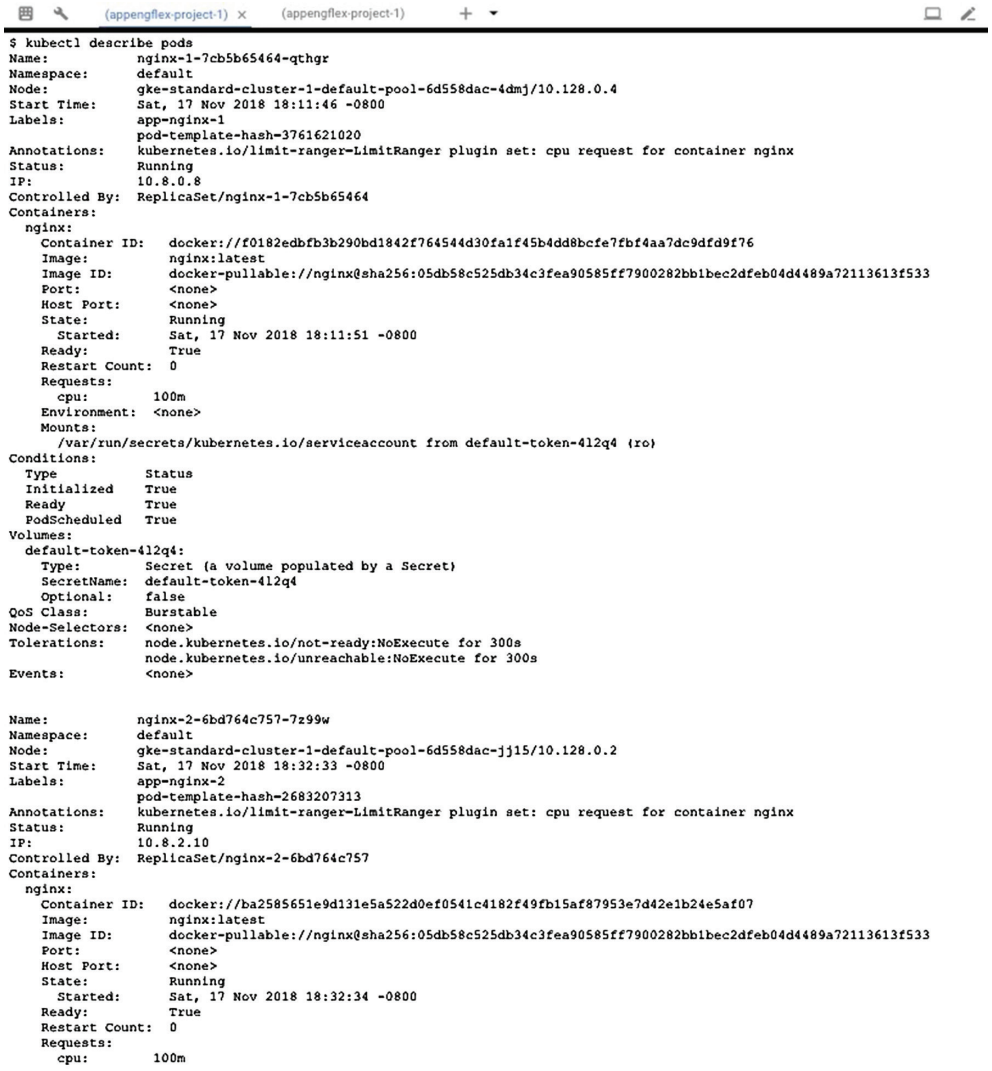
FIGURE 8.20 Partial listing of the details shown by the `kubectl describe nodes` command



```
(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.

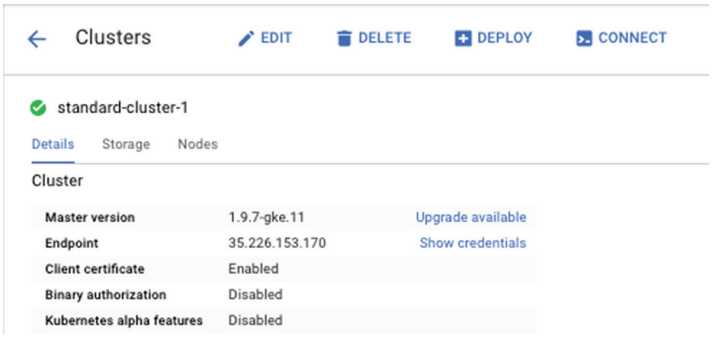
Adding, Modifying, and Removing Nodes

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

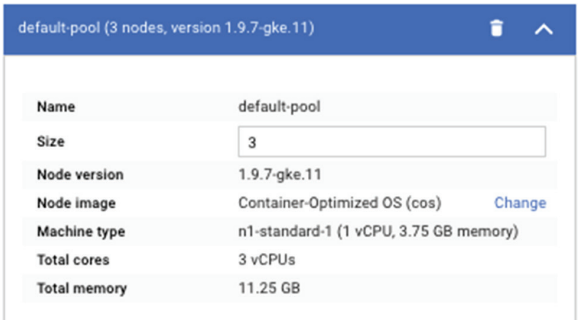
From Cloud Console, navigate to the Kubernetes Engine page and display a list of clusters. Click the name of a cluster to display its details, as in Figure 8.22. Note the Edit option near the top of the screen. Click that to open an Edit form.

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.

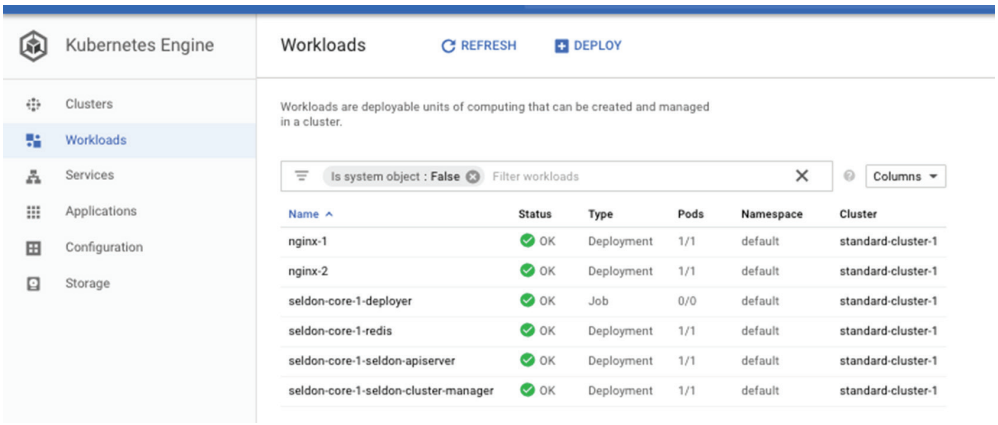
It is considered a best practice to not manipulate pods directly. Kubernetes will maintain the number of pods specified for a deployment. If you would like to change the number of pods, you should change the deployment configuration.

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.

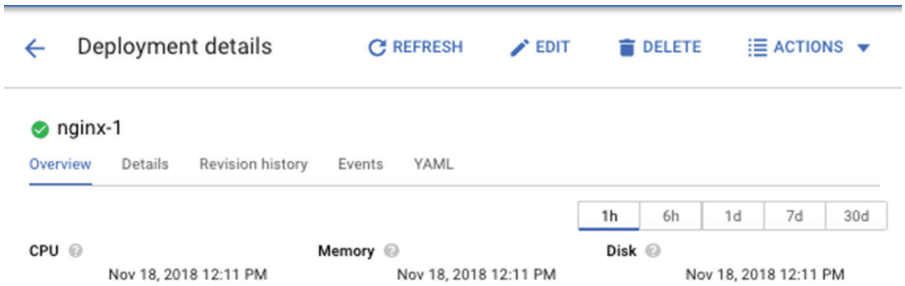
From Cloud Console, select the Workloads options from the navigation menu on the left. This displays a list of deployments, as in Figure 8.24.

FIGURE 8.24 List of deployments in a cluster



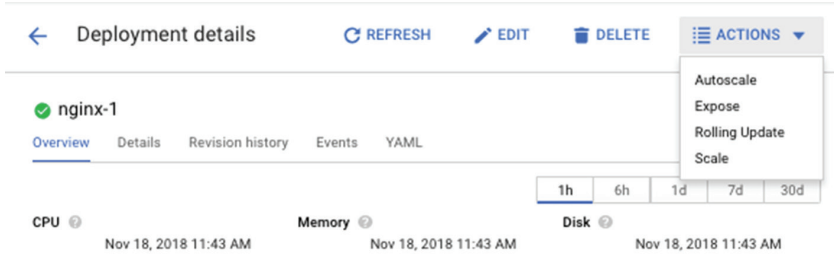
Click the name of the deployment you want to modify; a form is displayed with details such as in Figure 8.25. Note the Actions option in the top horizontal menu.

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

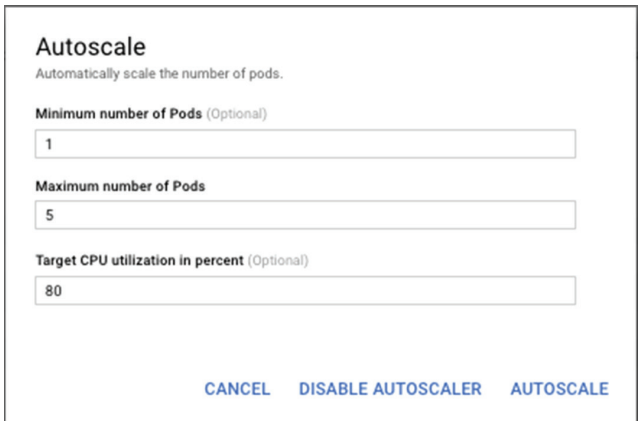


Select Scale to display a dialog that allows you to set a new size for the workload, as shown in Figure 8.27. In this example, the number of replicas has been changed to 2.

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

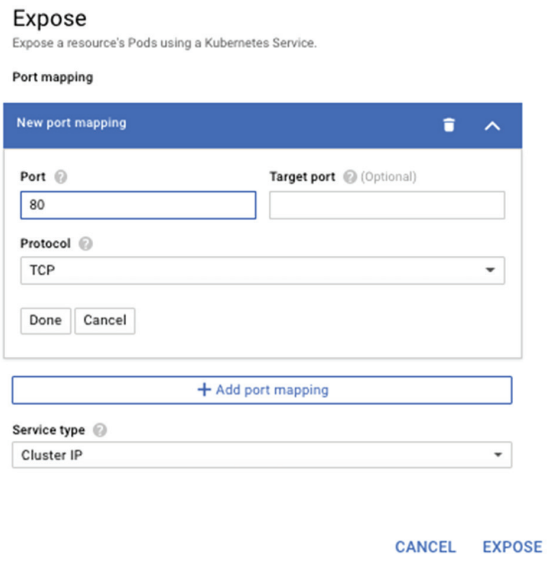
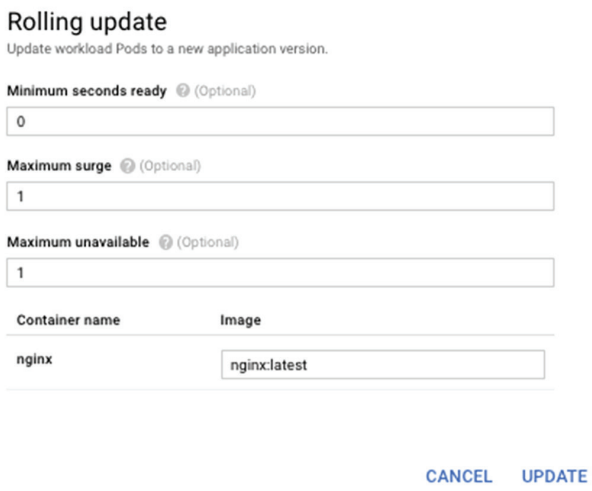


FIGURE 8.30 Form to specify parameters for rolling updates of code running in pods



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

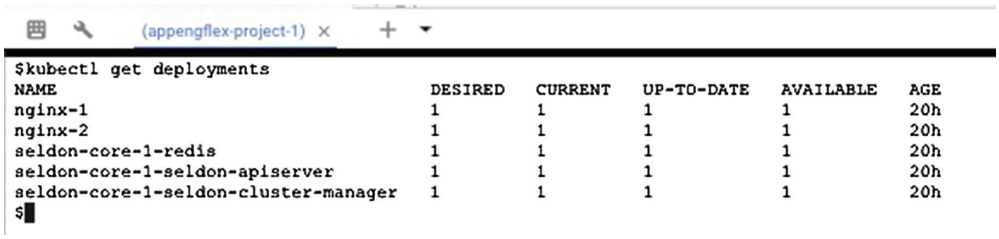
Working with pods in Cloud SDK and Cloud Shell is done by working with deployments; deployments were explained earlier in the section "Adding, Modifying, and Removing Pods with Cloud Console." You can use the `kubectl` command to work with deployments.

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

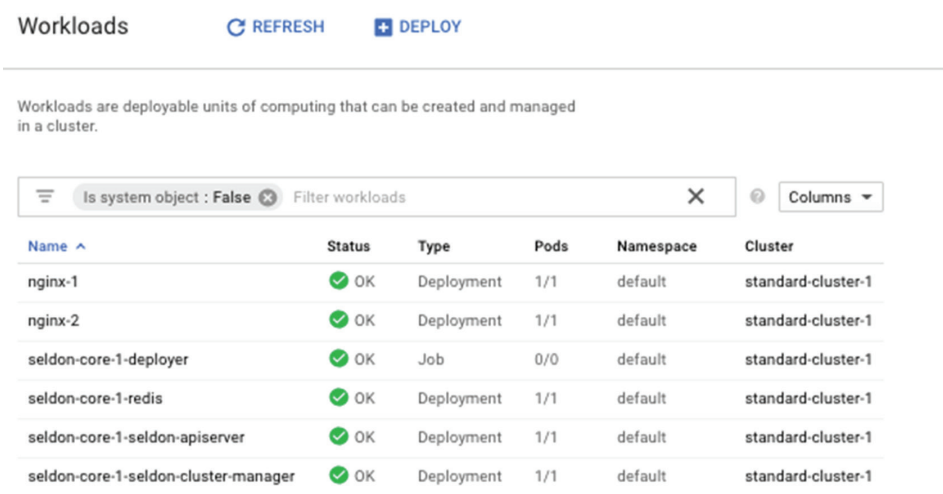
You can add, modify, and remove services from a cluster using either Cloud Console or Cloud SDK in your local environment, on a GCP VM, or in Cloud Shell.

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

Adding, Modifying, and Removing Services with Cloud Console

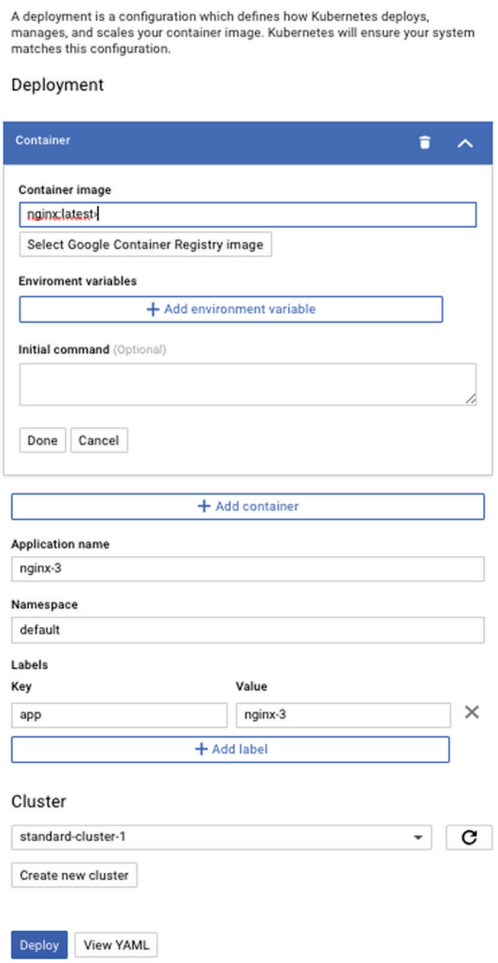
Services are added through deployments. In Cloud Console, select the Workloads option from the navigation menu to display a list of deployments, as in Figure 8.32. Note the Deploy option in the horizontal menu at the top of the page.

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



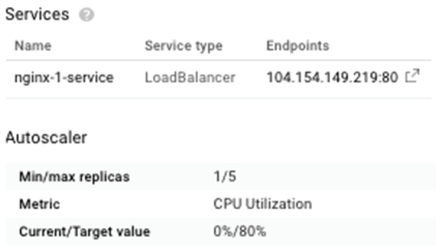
In the Container Image parameter, you can specify the name of an image or select one from the Google Container Repository. To specify a name directly, specify a path to the image using a URL such as this:

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

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

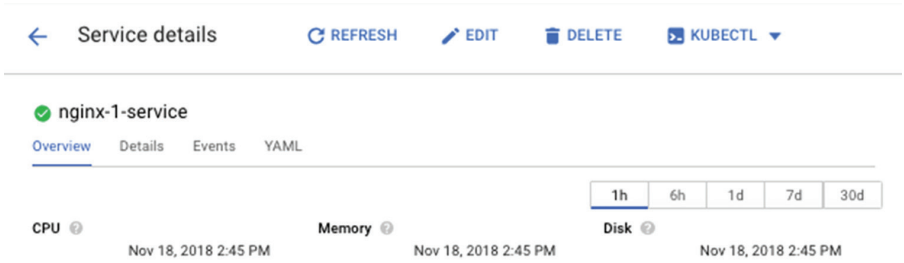
When you click the name of a deployment, like those listed earlier in Figure 8.32, you will see details of that deployment, including a list of services, like that shown in Figure 8.34.

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.

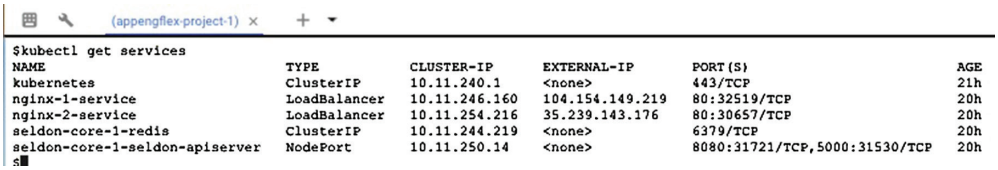
FIGURE 8.35 Navigate to the Service Details page to delete a service using the Delete option in the horizontal menu.



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
```

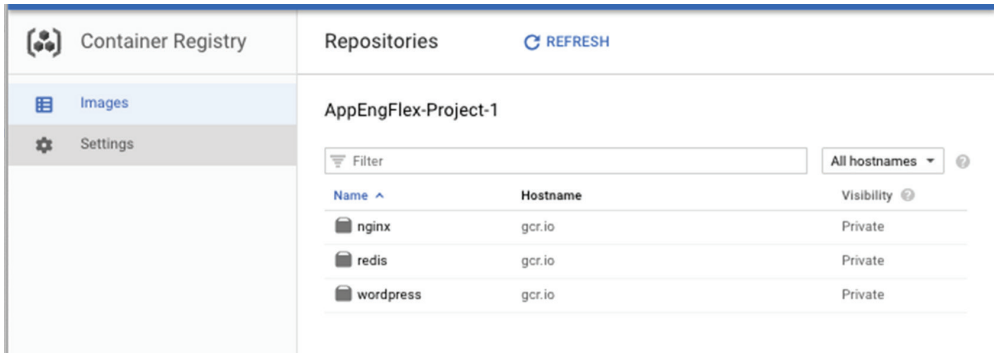
Viewing the Image Repository and Image Details

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

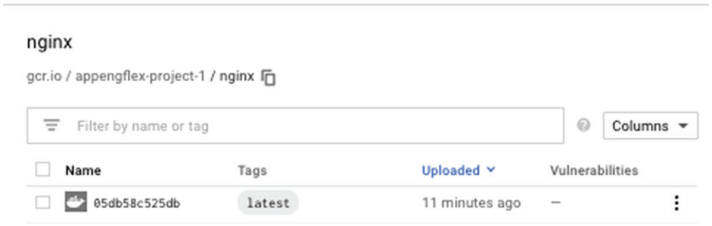
In Cloud Console, select Container Registry from the navigation menu to display the contents of a registry. Figure 8.37 shows an example listing with three images for Nginx, Redis, and WordPress.

FIGURE 8.37 A listing of images in a Container Registry



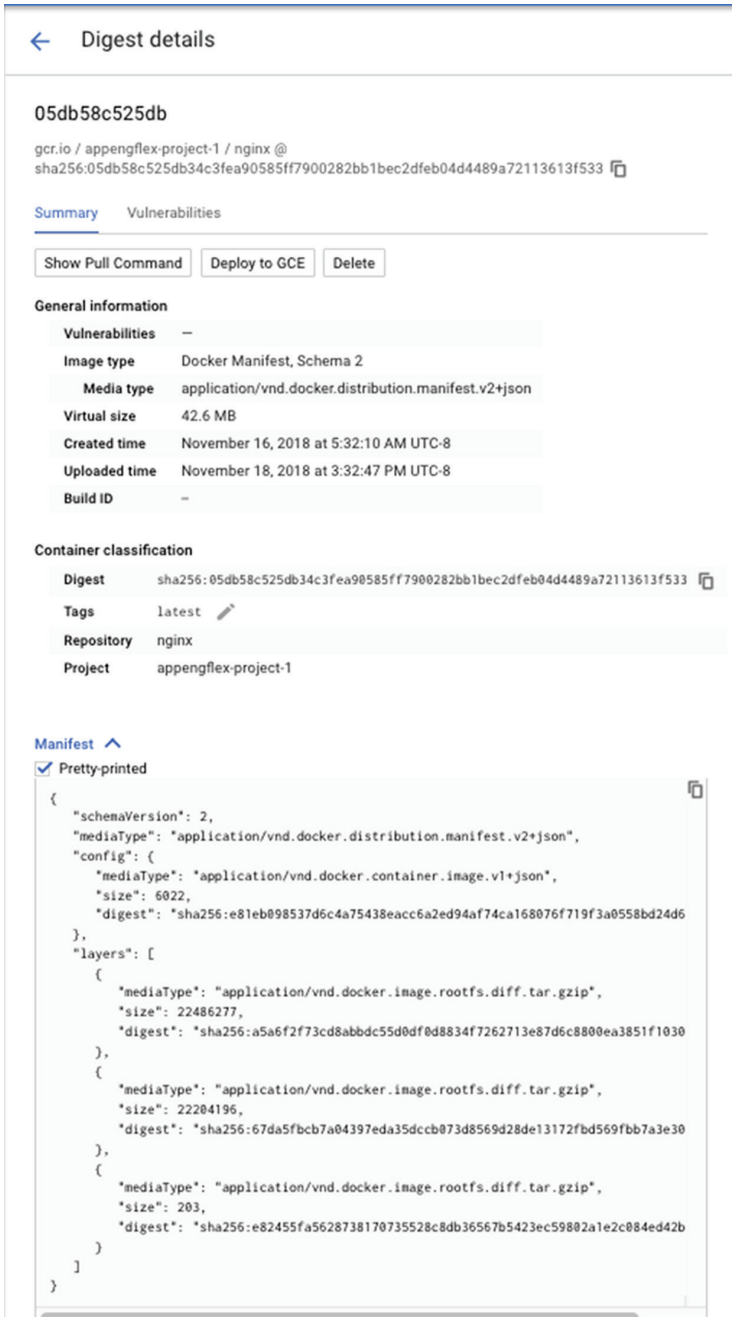
To see the details of an image, click the image name. For example, Figure 8.38 shows a listing for the Nginx image. This listing will list one entry for each version of the image. Since there is only one version of the image, there is only one listed.

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



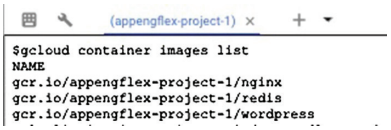
Viewing the Image Repository and Image Details with Cloud SDK and Cloud Shell

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.