SAMPLE CHAPTER

Google Cloud Platform IN ACTION

JJ Geewax

Foreword by Urs Hölzle

MANNING

Google Cloud Platform in Action

by JJ Geewax

Chapter 1

brief contents

1 What is "cloud"?

This chapter covers

- Overview of "the cloud"
- When and when not to use cloud hosting and what to expect
- Explanation of cloud pricing principles
- What it means to build an application for the cloud
- A walk-through of Google Cloud Platform

The term "cloud" has been used in many different contexts and it has many different definitions, so it makes sense to define the term—at least for this book.

> Cloud *is a collection of services that helps developers focus on their project rather than on the infrastructure that powers it.*

In more concrete terms, cloud services are things like Amazon Elastic Compute Cloud (EC2) or Google Compute Engine (GCE), which provide APIs to provision virtual servers, where customers pay per hour for the use of these servers.

In many ways, cloud is the next layer of abstraction in computer infrastructure, where computing, storage, analytics, networking, and more are all pushed higher up the computing stack. This structure takes the focus of the developer away from CPUs and RAM and toward APIs for higher-level operations such as storing or querying for data. Cloud services aim to solve your problem, not give you low-level tools for you to do so on your own. Further, cloud services are extremely flexible, with most requiring no provisioning or long-term contracts. Due to this, relying on these services allows you to scale up and down with no advanced notice or provisioning, while paying only for the resources you use in a given month.

1.1 What is Google Cloud Platform?

There are many cloud providers out there, including Google, Amazon, Microsoft, Rackspace, DigitalOcean, and more. With so many competitors in the space, each of these companies must have its own take on how to best serve customers. It turns out that although each provides many similar products, the implementation and details of how these products work tends to vary quite a bit.

Google Cloud Platform (often abbreviated as GCP) is a collection of products that allows the world to use some of Google's internal infrastructure. This collection includes many things that are common across all cloud providers, such as on-demand virtual machines via Google Compute Engine or object storage for storing files via Google Cloud Storage. It also includes APIs to some of the more advanced Google-built technology, like Bigtable, Cloud Datastore, or Kubernetes.

Although Google Cloud Platform is similar to other cloud providers, it has some differences that are worth mentioning. First, Google is "home" to some amazing people, who have created some incredible new technologies there and then shared them with the world through research papers. These include MapReduce (the research paper that spawned Hadoop and changed how we handle "Big Data"), Bigtable (the paper that spawned Apache HBase), and Spanner. With Google Cloud Platform, many of these technologies are no longer "only for Googlers."

Second, Google operates at such a scale that it has many economic advantages, which are passed on in the form of lower prices. Google owns immense physical infrastructure, which means it buys and builds custom hardware to support it, which means cheaper overall prices, often combined with improved performance. It's sort of like Costco letting you open up that 144-pack of potato chips and pay 1/144th the price for one bag.

1.2 Why cloud?

So why use cloud in the first place? First, cloud hosting offers a lot of flexibility, which is a great fit for situations where you don't know (or can't know) how much computing power you need. You won't have to overprovision to handle situations where you might need a lot of computing power in the morning and almost none overnight.

Second, cloud hosting comes with the maintenance built in for several products. This means that cloud hosting results in minimal extra work to host your systems compared to other options where you might need to manage your own databases, operating systems, and even your own hardware (in the case of a colocated hosting provider). If you don't want to (or can't) manage these types of things, cloud hosting is a great choice.

1.2.1 Why not cloud?

Obviously this book is focused on using Google Cloud Platform, so there's an assumption that cloud hosting is a good option for your company. It seems worthwhile, however, to devote a few words to why you might *not* want to use cloud hosting. And yes, there are times when cloud is not the best choice, even if it's often the cheapest of all the options.

Let's start with an extreme example: Google itself. Google's infrastructural footprint is exabytes of data, hundreds of thousands of CPUs, a relatively stable and growing overall workload. In addition, Google is a big target for attacks (for example, denial-of-service attacks) and government espionage and has the budget and expertise to build gigantic infrastructural footprints. All of these things together make Google a bad candidate for cloud hosting.

Figure 1.1 shows a visual representation of a usage and cost pattern that would be a bad fit for cloud hosting. Notice how the growth of computing needs (the bottom line) steadily increases, and the company is provisioning extra capacity regularly to stay ahead of its needs (the top, wavy line).

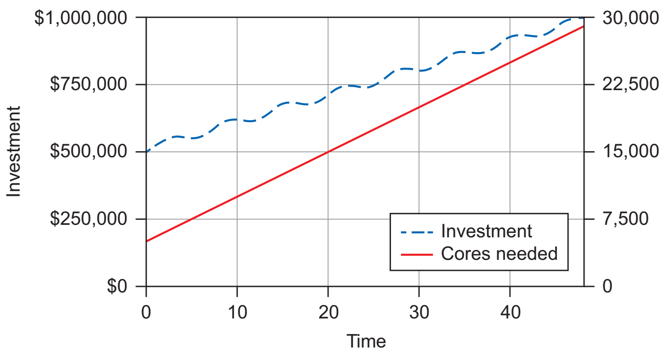


Figure 1.1 Steady growth in resource consumption

Compare this with figure 1.2, which shows a more typical company of the internet age, where growth is spiky and unpredictable and tends to drop without much notice. In this case, the company bought enough computing capacity (the top line) to handle a spike, which was needed up front, but then when traffic fell (the bottom line), it was stuck with quite a bit of excess capacity.

In short, if you have the expertise to run your own data centers (including the plans for disasters and other failures, and the recovery from those potential disasters), along with steady growing computing needs (measured in cores, storage, networking

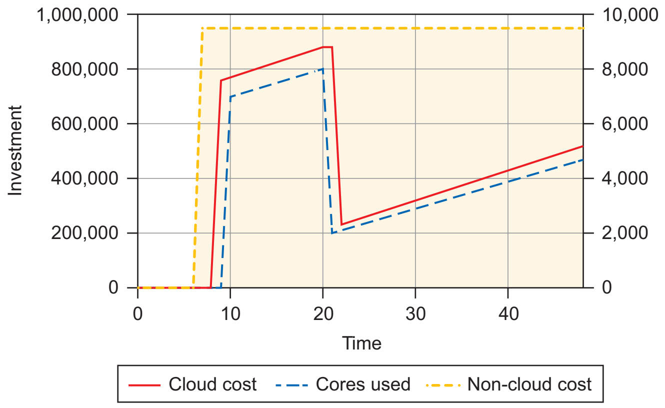


Figure 1.2 Unexpected pattern of resource consumption

consumption, and so on), cloud hosting might not be right for you. If you're anything like the typical company of today, where you don't know what you need today (and certainly don't know what you'll need several years from today), and don't have the expertise in your company to build out huge data centers to achieve the same economies of scale that large cloud providers can offer, cloud hosting is likely to be a good fit for you.

1.3 What to expect from cloud services

All of the discussion so far has been about cloud in the broader sense. Let's take a moment to look at some of the more specific things that you should expect from cloud services, particularly how cloud specifically differs from other hosting options.

1.3.1 Computing

You've already learned a little bit about how cloud computing is fundamentally different from virtual private, colocated, or on-premises hosting. Let's take a look at what you can expect if you decide to take the plunge into the world of cloud computing.

The first thing you'll notice is that provisioning your machine will be fast. Compared to colocated or on-premises hosting, it should be significantly faster. In real terms, the typical expected time from clicking the button to connecting via secure shell to the machine will be about a minute. If you're used to virtual private hosting, the provisioning time might be around the same, maybe slightly faster.

What's more interesting is what is missing in the process of turning on a cloud-hosted virtual machine (VM). If you turn on a VM right now, you might notice that there's no mention of payment. Compare that to your typical virtual private server (VPS), where you agree on a set price and purchase the VPS for a full year, making monthly payments (with your first payment immediately, and maybe a discount for up-front payment). Google doesn't mention payment at this time for a simple reason:

they don't know how long you'll keep that machine running, so there's no way to know how much to charge you. It can determine how much you owe only either at the end of the month or when you turn off the VM. See table 1.1 for a comparison.

Table 1.1 Hosting choice comparison

Hosting choice	Best if...	Kind of like...
Building your own data center	You have steady long-term needs at a large scale.	Purchasing a car
Using your own hardware in a colocation facility	You have steady long-term needs at a smaller scale.	Leasing a car
Using virtual private hosting	You have slowly changing needs.	Renting a car
Using cloud hosting	You have rapidly changing (or unknown) needs.	Taking an Uber

1.3.2 Storage

Storage, although not the most glamorous part of computing, is incredibly necessary. Imagine if you weren't able to save your data when you were done working on it? Cloud's take on storage follows the same pattern you've seen so far with computing, abstracting away the management of your physical resources. This might seem unimpressive, but the truth is that storing data is a complicated thing to do. For example, do you want your data to be edge-cached to speed up downloads for users on the internet? Are you optimizing for throughput or latency? Is it OK if the "time to first byte" is a few seconds? How available do you need the data to be? How many concurrent readers do you need to support?

The answers to these questions change what you build in significant ways, so much so that you might end up building entirely different products if you were the one building a storage service. Ultimately, the abstraction provided by a storage service gives you the ability to configure your storage mechanisms for various levels of performance, durability, availability, and cost.

But these systems come with a few trade-offs. First, the failure aspects of storing data typically disappear. You shouldn't ever get a notification or a phone call from someone saying that a hard drive failed and your data was lost. Next, with reduced-availability options, you might occasionally try to download your data and get an error telling you to try again later, but you'll be paying much less for storage of that class than any other. Finally, for virtual disks in the cloud, you'll notice that you have lots of choices about how you can store your data, both in capacity (measured in GB) and in performance (typically measured in input/output operations per second [IOPS]). Once again, like computing in the cloud, storing data on virtual disks in the cloud feels familiar.

On the other hand, some of the custom database services, like Cloud Datastore, might feel a bit foreign. These systems are in many ways completely unique to cloud hosting, relying on huge, shared, highly scalable systems built by and for Google. For example, Cloud Datastore is an adapted externalization of an internal storage system called Megastore, which was, until recently, the underlying storage system for many Google products, including Gmail. These hosted storage systems sometimes required you to integrate your own code with a proprietary API. This means that it'll become all the more important to keep a proper layer of abstraction between your code base and the storage layer. It still may make sense to rely on these hosted systems, particularly because all of the scaling is handled automatically.

1.3.3 Analytics (aka, Big Data)

Analytics, although not something typically considered "infrastructure," is a quickly growing area of hosting—though you might often see this area called "Big Data." Most companies are logging and storing almost everything, meaning the amount of data they have to analyze and use to draw new and interesting conclusions is growing faster and faster every day. This also means that to help make these enormous amounts of data more manageable, new and interesting open source projects are popping up, such as Apache Spark, HBase, and Hadoop.

As you might guess, many of the large companies that offer cloud hosting also use these systems, but what should you expect to see from cloud in the analytics and big data areas?

1.3.4 Networking

Having lots of different pieces of infrastructure running is great, but without a way for those pieces to talk to each other, your system isn't a single system—it's more of a pile of isolated systems. That's not a big help to anyone. Traditionally, we tend to take networking for granted as something that should work. For example, when you sign up for virtual private hosting and get access to your server, you tend to expect that it has a connection to the internet and that it will be fast enough.

In the world of cloud computing some of these assumptions remain unchanged. The interesting parts come up when you start developing the need for more advanced features, such as faster-than-normal network connections, advanced firewalling abilities (where you only allow certain IPs to talk to certain ports), load balancing (where requests come in and can be handled by any one of many machines), and SSL certificate management (where you want requests to be encrypted but don't want to manage the certificates for each individual virtual machine).

In short, networking on traditional hosting is typically hidden, so most people won't notice any differences, because there's usually nothing to notice. For those of you who do have a deep background in networking, most of the things you can do with your typical computing stack (such as configure VPNs, set up firewalls with `iptables`, and balance requests across servers using HAProxy) are all still possible. Google Cloud's networking features only act to simplify the common cases, where instead of running a separate VM with HAProxy, you can rely on Google's Cloud Load Balancer to route requests.

1.3.5 Pricing

In the technology industry, it's been commonplace to find a single set of metrics and latch on to those as the only factors in a decision-making process. Although many times that is a good heuristic in making the decision, it can take you further away from the market when estimating the total cost of infrastructure and comparing against the market price of the physical goods. Comparing only the dollar cost of buying the hardware from a vendor versus a cloud hosting provider is going to favor the vendor, but it's not an apples-to-apples comparison. So how do we make everything into apples?

When trying to compare costs of hosting infrastructure, one great metric to use is TCO, or total cost of ownership. This metric factors in not only the cost of purchasing the physical hardware but also ancillary costs such as human labor (like hardware administrators or security guards), utility costs (electricity or cooling), and one of the most important pieces—support and on-call staff who make sure that any software services running stay that way, at all hours of the night. Finally, TCO also includes the cost of building redundancy for your systems so that, for example, data is never lost due to a failure of a single hard drive. This cost is more than the cost of the extra drive—you need to not only configure your system, but also have the necessary knowledge to design the system for this configuration. In short, TCO is everything you pay for when buying hosting.

If you think more deeply about the situation, TCO for hosting will be close to the cost of goods sold for a virtual private hosting company. With cloud hosting providers, TCO is going to be much closer to what you pay. Due to the sheer scale of these cloud providers, and the need to build these tools and hire the ancillary labor anyway, they're able to reduce the TCO below traditional rates, and every reduction in TCO for a hosting company introduces more room for a larger profit margin.

1.4 Building an application for the cloud

So far this chapter has been mainly a discussion on what cloud is and what it means for developers looking to rely on it rather than traditional hosting options. Let's switch gears now and demonstrate how to deploy something meaningful using Google Cloud Platform.

1.4.1 What is a cloud application?

In many ways, an application built for the cloud is like any other. The primary difference is in the assumptions made about the application's architecture. For example, in a traditional application, we tend to deploy things such as binaries running on particular servers (for example, running a MySQL database on one server and Apache with `mod_php` on another). Rather than thinking in terms of which servers handle which things, a typical cloud application relies on hosted or managed services whenever possible. In many cases it relies on containers the way a traditional application would rely on servers. By operating this way, a cloud application is often much more flexible and able to grow and shrink, depending on the customer demand throughout the day.

Let's take a moment to look at an example of a cloud application and how it might differ from the more traditional applications that you might already be familiar with.

1.4.2 Example: serving photos

If you've ever built a toy project that allows users to upload their photos (for example, a Facebook clone that stores a profile photo), you're probably familiar with dealing with uploaded data and storing it. When you first started, you probably made the age-old mistake of adding a `BINARY` or `VARBINARY` column to your database, calling it `profile_photo`, and shoving any uploaded data into that column.

If that's a bit too technical, try thinking about it from an architectural standpoint. The old way of doing this was to store the image data in your relational database, and then whenever someone wanted to see the profile photo, you'd retrieve it from the database and return it through your web server, as shown in figure 1.3.

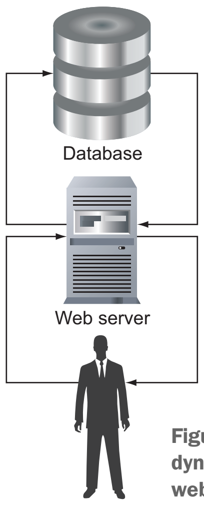


Figure 1.3 Serving photos dynamically through your web server

In case it wasn't clear, this is bad for a variety of reasons. First, storing binary data in your database is inefficient. It does work for transactional support, which profile photos probably don't need. Second, and most important, by storing the binary data of a photo in your database, you're putting extra load on the database itself, but not using it for the things it's good at, like joining relational data together.

In short, if you don't need transactional semantics on your photo (which here, we don't), it makes more sense to put the photo somewhere on a disk and then use the static serving capabilities of your web server to deliver those bytes, as shown in figure 1.4. This leaves the database out completely, so it's free to do more important work.

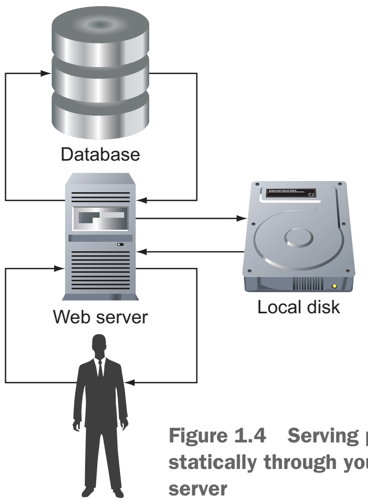


Figure 1.4 Serving photos statically through your web server

This structure is a huge improvement and probably performs quite well for most use cases, but it doesn't illustrate anything special about the cloud. Let's take it a step further and consider geography for a moment. In your current deployment, you have a single web server living somewhere inside a data center, serving a photo it has stored locally on its disk. For simplicity, let's assume this server lives somewhere in the central United States. This means that if someone nearby (for example, in New York) requests that photo, they'll get a relatively zippy response. But what if someone far away, like in Japan, requests the photo? The only way to get it is to send a request from Japan to the United States, and then the server needs to ship all the bytes from the United States back to Japan.

This transaction could take on the order of hundreds of milliseconds, which might not seem like a lot, but imagine you start requesting lots of photos on a single page. Those hundreds of milliseconds start adding up. What can you do about this? Most of you might already know the answer is edge caching, or relying on a content distribution network. The idea of these services is that you give them copies of your data (in this case, the photos), and they store those copies in lots of different geographical locations. Then, instead of sending a URL to the image on your single server, you send a URL pointing to this content distribution provider, and it returns the photo using the closest available server. So where does cloud come in?

Instead of optimizing your existing storage setup, the goal of cloud hosting is to provide managed services that solve the problem from start to finish. Instead of storing the photo locally and then optimizing that configuration by using a content delivery network (CDN), you'd use a managed storage service, which handles content distribution automatically—exactly what Google Cloud Storage does.

In this case, when someone uploads a photo to your server, you'd resize it and edit it however you want, and then forward the final image along to Google Cloud Storage, using its API client to ship the bytes securely. See figure 1.5. After that, all you'd do is refer to the photo using the Cloud Storage URL, and all of the problems from before are taken care of.

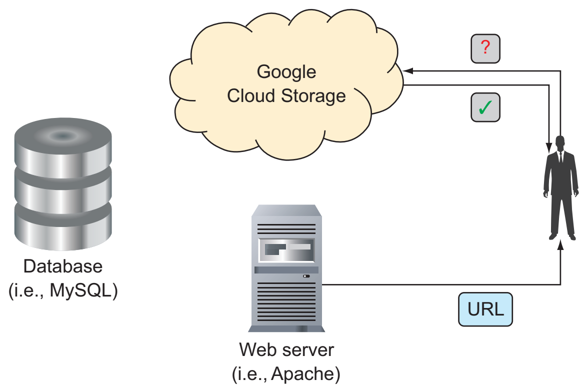


Figure 1.5 Serving photos statically through Google Cloud Storage

This is only one example, but the theme you should take away from this is that cloud is more than a different way of managing computing resources. It's also about using managed or hosted services via simple APIs to do complex things, meaning you think less about the physical computers.

More complex examples are, naturally, more difficult to explain quickly, so next let's introduce a few specific examples of companies or projects you might build or work on. We'll use these later to explore some of the interesting ways that cloud infrastructure attempts to solve the common problems found with these projects.

1.4.3 Example projects

Let's explore a few concrete examples of projects you might work on.

TO-DO LIST

If you've ever researched a new web development framework, you've probably seen this example paraded around, showcasing the speed at which you can do something real. ("Look how easy it is to make a to-do list app with our framework!") To-Do List is nothing more than an application that allows users to create lists, add items to the lists, and mark them as complete.

Throughout this book, we rely on this example to illustrate how you might use Google Cloud for your personal projects, which quite often involve storing and retrieving data and serving either API or web requests to users. You'll notice that the focus of this example is building something "real," but it won't cover all of the edge cases (and there may be many) or any of the more advanced or enterprise-grade features. In short, the To-Do List is a useful demonstration of doing something real, but incredibly simple, with cloud infrastructure.

INSTASNAP

InstaSnap is going to be our typical example of "the next big thing" in the start-up world. This application allows users to take photos or videos, share them on a "timeline" (akin to the Instagram or Facebook timeline), and have them self-destruct (akin to the SnapChat expiration).

The wrench thrown in with InstaSnap is that although in the early days most of the focus was on building the application, the current focus is on scaling the application to handle hundreds of thousands of requests every single second. Additionally, all of these photos and videos, though small on their own, add up to enormous amounts of data. In addition, celebrities have started using the system, meaning it's becoming more and more common for thousands of people to request the same photos at the same time. We'll rely on this example to demonstrate how cloud infrastructure can be used to achieve stability even in the face of an incredible number of requests. We also may use this example when pointing out some of the more advanced features provided by cloud infrastructure.

E*EXCHANGE

E*Exchange is our example of more grown-up application development that tends to come with growing from a small or mid-sized company into a larger, more mature, more heavily capitalized company, which means audits, Sarbanes-Oxley, and all the other (potentially scary) requirements. To make things more complicated, E*Exchange is an application for trading stocks in the United States, and, therefore, will act as an example of applications operating in more highly regulated industries, such as finance.

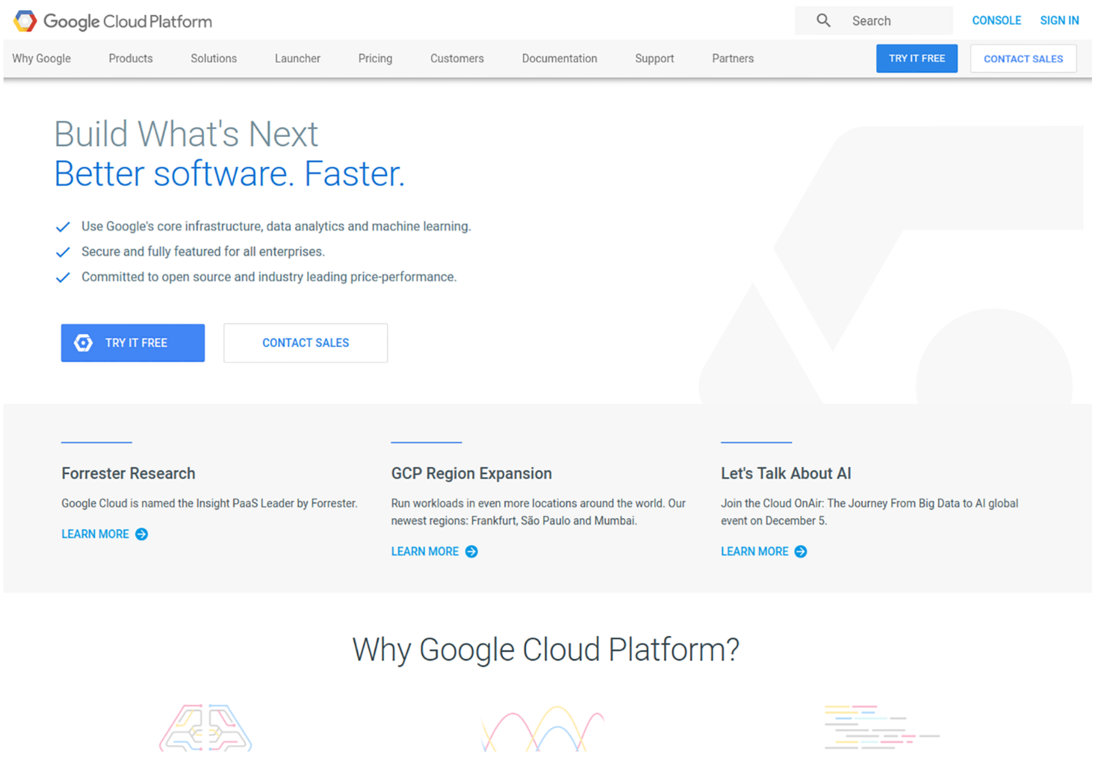
E*Exchange comes up whenever we explore several of the many enterprise-grade features of cloud infrastructure, as well as some of the concerns about using shared services, particularly with regard to security and access control. Hopefully these examples will help you bridge the gap between cool features that seem fun—or boring features that seem useless—and real-life use cases of these features, including how you can rely on cloud infrastructure to do some (or most) of the heavy lifting.

1.5 Getting started with Google Cloud Platform

Now that you've learned a bit about cloud in general, and what Google Cloud Platform can do more specifically, let's begin exploring GCP.

1.5.1 Signing up for GCP

Before you can start using any of Google's Cloud services, you first need to sign up for an account. If you already have a Google account (such as a Gmail account), you can use that to log in, but you'll still need to sign up specifically for a cloud account. If you've already signed up for Google Cloud Platform (see figure 1.6), feel free to skip



Build What's Next
Better software. Faster.

- Use Google's core infrastructure, data analytics and machine learning.
- Secure and fully featured for all enterprises.
- Committed to open source and industry leading price-performance.

Forrester Research
Google Cloud is named the Insight PaaS Leader by Forrester.
LEARN MORE

GCP Region Expansion
Run workloads in even more locations around the world. Our newest regions: Frankfurt, São Paulo and Mumbai.
LEARN MORE

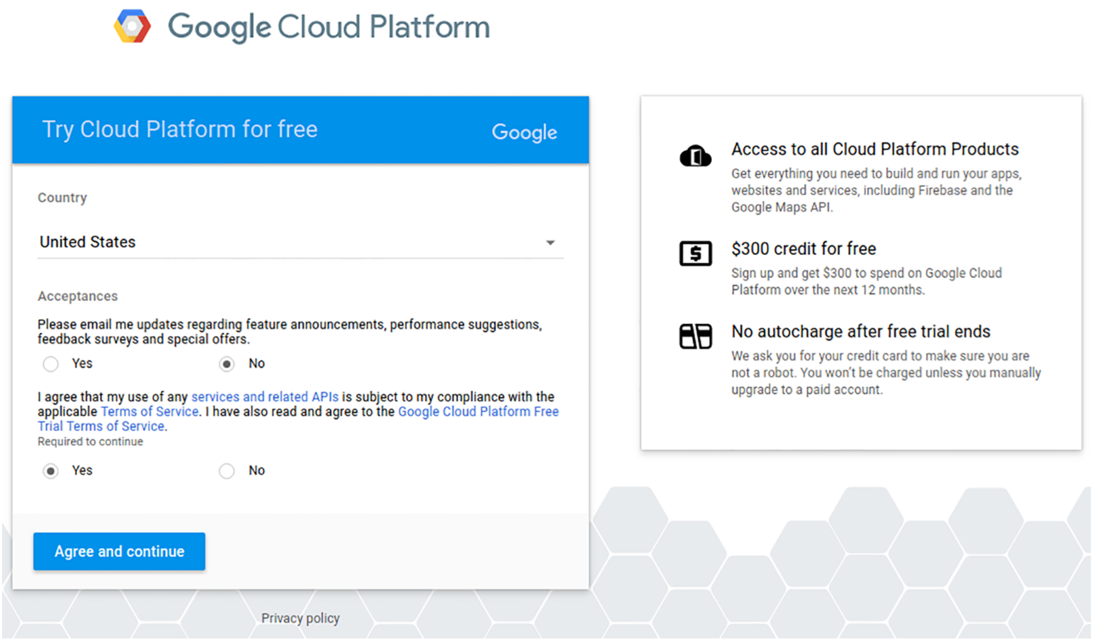
Let's Talk About AI
Join the Cloud OnAir: The Journey From Big Data to AI global event on December 5.
LEARN MORE

Why Google Cloud Platform?

Figure 1.6 Google Cloud Platform

ahead. First, navigate to https://cloud.google.com, and click the button that reads "Try it free!" This will take you through a typical Google sign-in process. If you don't have a Google account yet, follow the sign-up process to create one.

If you're eligible for the free trial, you'll see a page prompting you to enter your billing information. The free trial, shown in figure 1.7, gives you $300 to spend on Google Cloud over a period of 12 months, which should be more than enough time to explore all the things in this book. Additionally, some of the products on Google Cloud Platform have a free tier of usage. Either way, all the exercises in this book will remind you to turn off any resources after the exercise is finished.



Country

United States

Acceptances

Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers.

Yes [✓ No]

I agree that my use of any services and related APIs is subject to my compliance with the applicable Terms of Service. I have also read and agree to the Google Cloud Platform Free Trial Terms of Service.
Required to continue

[✓ Yes] No

Access to all Cloud Platform Products
Get everything you need to build and run your apps, websites and services, including Firebase and the Google Maps API.

$300 credit for free
Sign up and get $300 to spend on Google Cloud Platform over the next 12 months.

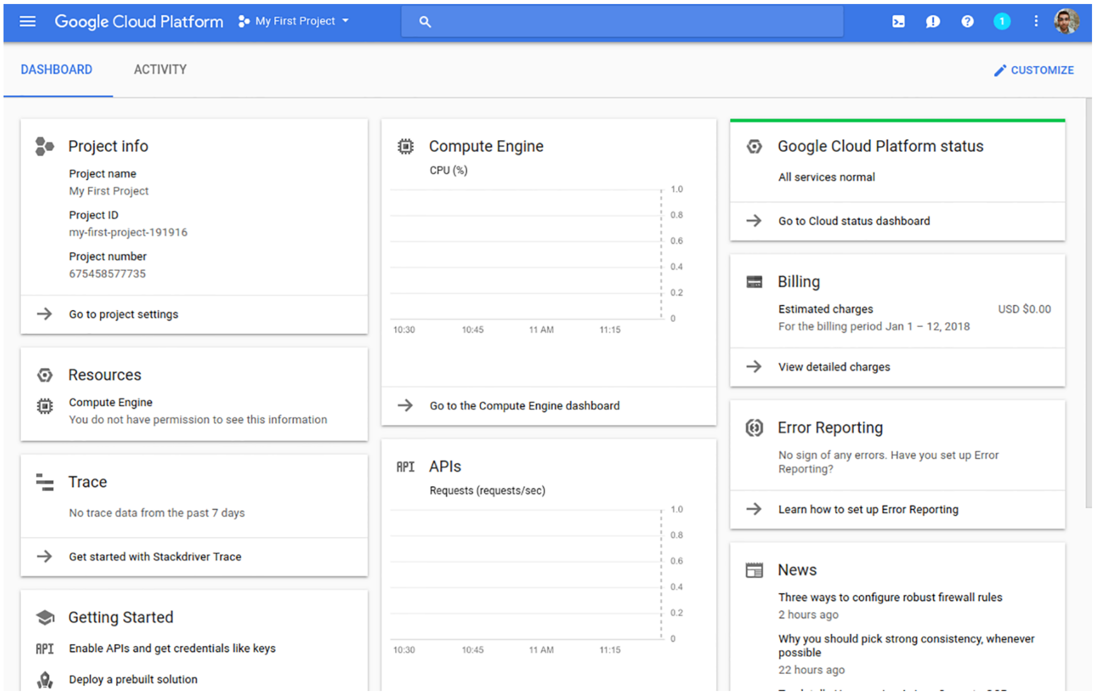
No autocharge after free trial ends
We ask you for your credit card to make sure you are not a robot. You won't be charged unless you manually upgrade to a paid account.

Figure 1.7 Google Cloud Platform free trial

1.5.2 Exploring the console

After you've signed up, you are automatically taken to the Cloud Console, shown in figure 1.8, and a new project is automatically created for you. You can think of a project like a container for your work, where the resources in a single project are isolated from those in all the other projects out there.

On the left side of the page are categories that correspond to all the different services that Google Cloud Platform offers (for example, Compute, Networking, Big Data, and Storage), as well as other project-specific configuration sections (such as authentication, project permissions, and billing). Feel free to poke around in the console to familiarize yourself with where things live. We'll come back to all of these



Project info
Project name
My First Project
Project ID
my-first-project-191916
Project number
675458577735
Go to project settings

Resources
Compute Engine
You do not have permission to see this information

Trace
No trace data from the past 7 days
Get started with Stackdriver Trace

Getting Started
Enable APIs and get credentials like keys
Deploy a prebuilt solution

Compute Engine
CPU (%)
Go to the Compute Engine dashboard

APIs
Requests (requests/sec)

Google Cloud Platform status
All services normal
Go to Cloud status dashboard

Billing
Estimated charges USD $0.00
For the billing period Jan 1 – 12, 2018
View detailed charges

Error Reporting
No sign of any errors. Have you set up Error Reporting?
Learn how to set up Error Reporting

News
Three ways to configure robust firewall rules
2 hours ago
Why you should pick strong consistency, whenever possible
22 hours ago

Figure 1.8 Google Cloud Console

things later as we explore each of these areas. Before we go any further, let's take a moment to look a bit closer at a concept that we threw out there: projects.

1.5.3 Understanding projects

When we first signed up for Google Cloud Platform, we learned that a new project is created automatically, and that projects have something to do with isolation, but what does this mean? And what are projects anyway? Projects are primarily a container for all the resources we create. For example, if we create a new VM, it will be "owned" by the parent project. Further, this ownership spills over into billing—any charges incurred for resources are charged to the project. This means that the bill for the new VM we mentioned is sent to the person responsible for billing on the parent project. (In our examples, this will be you!)

In addition to acting as the owner of resources, projects also act as a way of isolating things from one another, sort of like having a workspace for a specific purpose. This isolation applies primarily to security, to ensure that someone with access to one project doesn't have access to resources in another project unless specifically granted access. For example, if you create new service account credentials (which we'll do later) inside one project, say `project-a`, those credentials have access to resources only inside `project-a` unless you explicitly grant more access.

On the flip side, if you act as yourself (for example, `you@gmail.com`) when running commands (which you'll try in the next section), those commands can access anything that you have access to inside the Cloud Console, which includes *all* of the projects you've created, as well as ones that others have shared with you. This is one of the reasons why you'll see much of the code we write often explicitly specifies project IDs: you might have access to lots of different projects, so we have to clarify which one we want to own the thing we're creating or which project should get the bill for usage charges. In general, imagine you're a freelancer building websites and want to keep the work you do for different clients separate from one another. You'd probably have one project for each of the websites you build, both for billing purposes (one bill per website) and to keep each website securely isolated from the others. This setup also makes it easy to grant access to each client if they want to take ownership over their website or edit something themselves.

Now that we've gotten that out of the way, let's get back into the swing of things and look at how to get started with the Google Cloud software development kit (SDK).

1.5.4 Installing the SDK

After you get comfortable with the Google Cloud Console, you'll want to install the Google Cloud SDK. The SDK is a suite of tools for building software that uses Google Cloud, as well as tools for managing your production resources. In general, anything you can do using the Cloud Console can be done with the Cloud SDK, `gcloud`. To install the SDK, go to https://cloud.google.com/sdk/, and follow the instructions for your platform. For example, on a typical Linux distribution, you'd run this code:

```
$ export CLOUD_SDK_REPO="cloud-sdk-$(lsb_release -c -s)"
$ echo "deb http://packages.cloud.google.com/apt $CLOUD_SDK_REPO main" | \
  sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list
$ curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo \
  apt-key add -
$ sudo apt-get update && sudo apt-get install google-cloud-sdk
```

Feel free to install anything that looks interesting to you—you can always add or remove components later on. For each exercise that we go through, we always start by reminding you that you may need to install extra components of the Cloud SDK. You also may be occasionally prompted to upgrade components as they become available. For example, here's what you'll see when it's time to upgrade:

```
Updates are available for some Cloud SDK components. To install
them, please run:
 $ gcloud components update
```

As you can see, upgrading components is pretty simple: run `gcloud components update`, and the SDK handles everything. After you have everything installed, you have to tell the SDK who you are by logging in. Google made this easy by connecting your terminal and your browser:

```
$ gcloud auth login
Your browser has been opened to visit:

    [A long link is here]

    Created new window in existing browser session.
```

You should see a normal Google login and authorization screen asking you to grant the Google Cloud SDK access to your cloud resources. Now when you run future `gcloud` commands, you can talk to Google Cloud Platform APIs as yourself. After you click Allow, the window should automatically close, and the prompt should update to look like this:

```
$ gcloud auth login
Your browser has been opened to visit:

    [A long link is here]


Created new window in existing browser session.
WARNING: `gcloud auth login` no longer writes application default credentials.
If you need to use ADC, see:
  gcloud auth application-default --help

You are now logged in as [your-email-here@gmail.com].
Your current project is [your-project-id-here].  You can change this setting
     by running:
  $ gcloud config set project PROJECT_ID
```

You're now authenticated and ready to use the Cloud SDK as yourself. But what about that warning message? It says that even though you're logged in and all the `gcloud` commands you run will be authenticated as you, any code that you write may not be. You can make any code you write in the future automatically handle authentication by using application default credentials. You can get these using the `gcloud auth` subcommand once again:

```
$ gcloud auth application-default login
Your browser has been opened to visit:

    [Another long link is here]

Created new window in existing browser session.

Credentials saved to file:
     [/home/jjg/.config/gcloud/application_default_credentials.json]

These credentials will be used by any library that requests
Application Default Credentials.
```

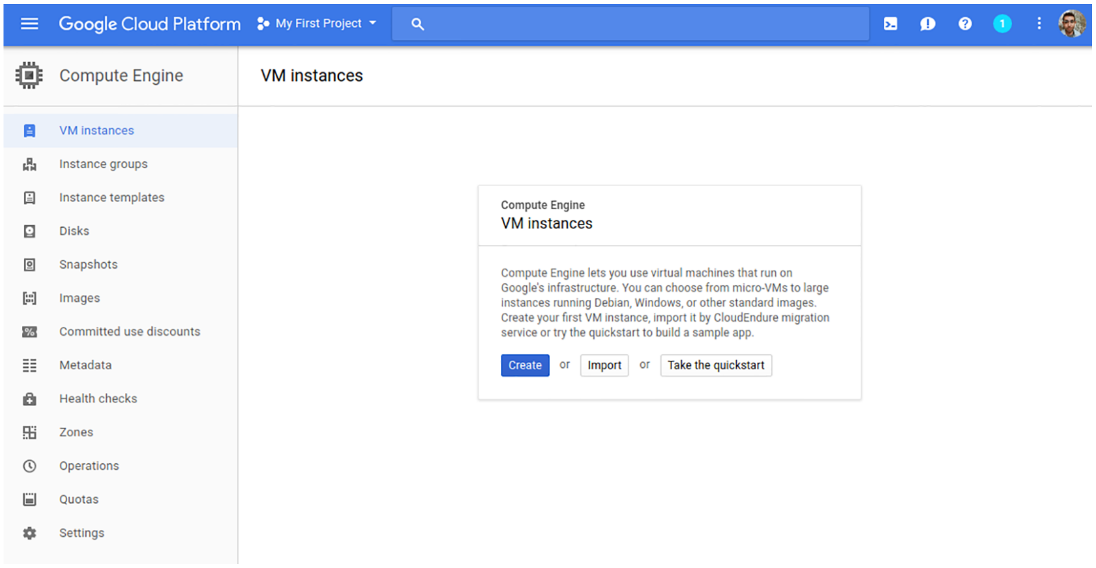
Now that we have dealt with all of the authentication pieces, let's look at how to interact with Google Cloud Platform APIs.

1.6 Interacting with GCP

Now that you've signed up and played with the console, and your local environment is all set up, it might be a good idea to try a quick practice task in each of the different ways you can interact with GCP. Let's start by launching a virtual machine in the cloud and then writing a script to terminate the virtual machine in JavaScript.

1.6.1 In the browser: the Cloud Console

Let's start by navigating to the Google Compute Engine area of the console: click the Compute section to expand it, and then click the Compute Engine link that appears. The first time you click this link, Google initializes Compute Engine for you, which should take a few seconds. Once that's complete, you should see a Create button, which brings you to a page, shown in figure 1.9, where you can configure your virtual machine.

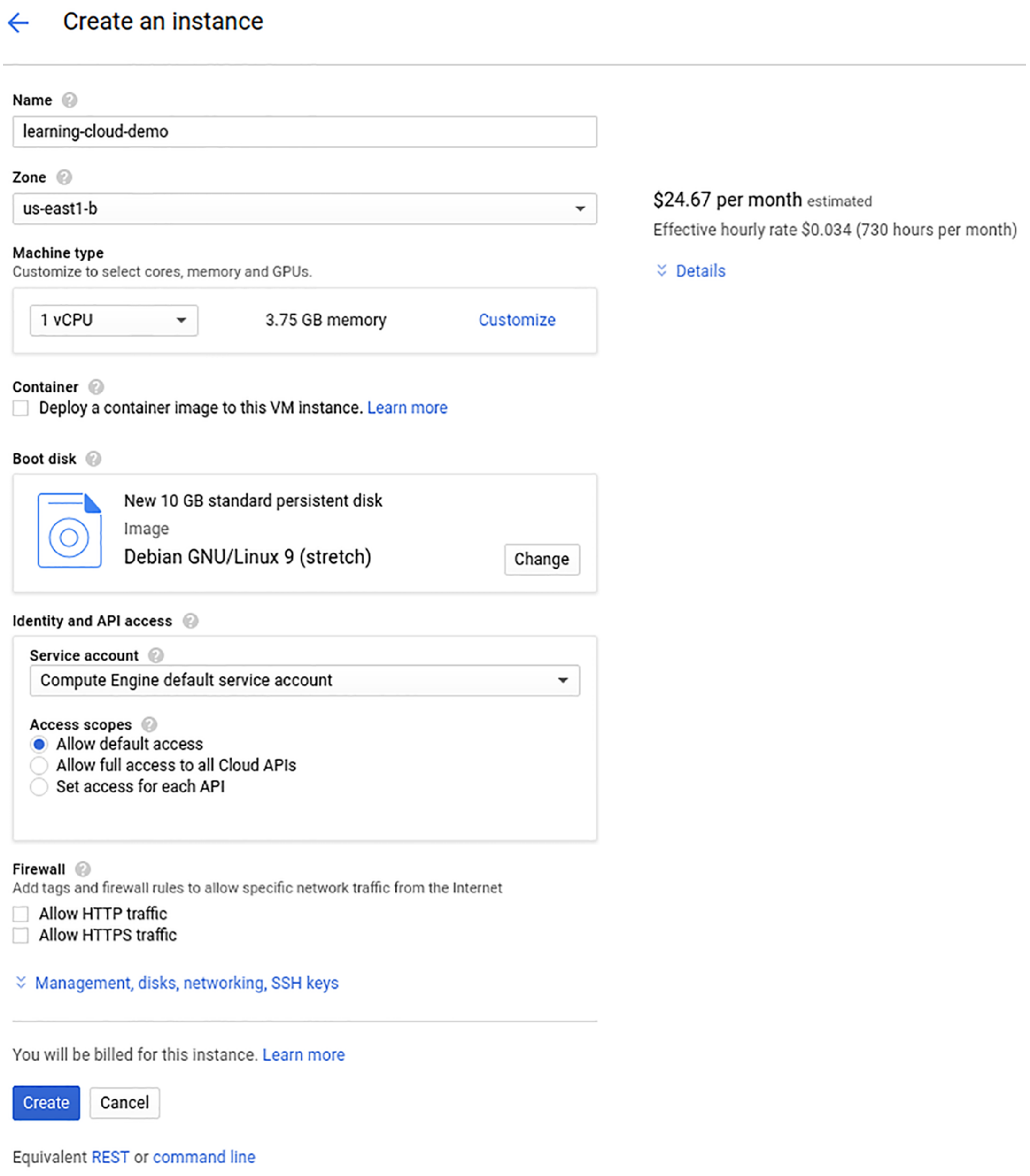


Compute Engine lets you use virtual machines that run on Google's infrastructure. You can choose from micro-VMs to large instances running Debian, Windows, or other standard images. Create your first VM instance, import it by CloudEndure migration service or try the quickstart to build a sample app.

Figure 1.9 Google Cloud Console, where you can create a new virtual machine

On the next page, a form (figure 1.10) lets you configure all the details of your instance, so let's take a moment to look at what all of the options are.

First there is the instance Name. The name of your virtual machine will be unique inside your project. For example, if you try to create "instance-1" while you already have an instance with that same name, you'll get an error saying that name is already taken. You can name your machines anything you want, so let's name our instance "learning-cloud-demo." Below that is the Zone field, which represents where the machine should live geographically. Google has data centers all over the place, so you



Create an instance

Name
learning-cloud-demo

Zone
us-east1-b

Machine type
Customize to select cores, memory and GPUs.
1 vCPU 3.75 GB memory Customize

Container
☐ Deploy a container image to this VM instance. Learn more

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

Identity and API access
Service account
Compute Engine default service account

Access scopes
● Allow default access
○ Allow full access to all Cloud APIs
○ Set access for each API

Firewall
Add tags and firewall rules to allow specific network traffic from the Internet
☐ Allow HTTP traffic
☐ Allow HTTPS traffic

Management, disks, networking, SSH keys

You will be billed for this instance. Learn more

Create Cancel

Equivalent REST or command line

$24.67 per month estimated
Effective hourly rate $0.034 (730 hours per month)
Details

Figure 1.10 Form where you define your virtual machine

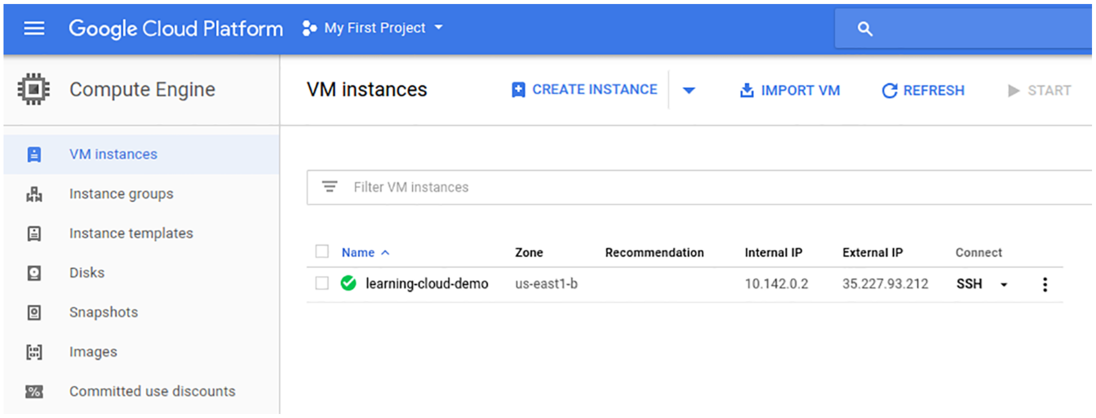
can choose from several options of where you want your instance to live. For now, let's put our instance in `us-central1-b` (which is in Iowa).

Next is the Machine Type field, where you can choose how powerful you want your cloud instances to be. Google has lots of different sizing options, ranging from f1-micro (which is a small, not powerful machine) all the way up to n1-highcpu-32 (which is a 32-core machine), or a n1-highmem-32 (which is a 32-core machine with 208 GB of RAM). As you can see, you have quite a few options, but because we’re testing things out, let’s leave the machine type as n1-standard-1, which is a single-core machine with about 4 GB of RAM.

Many, many more knobs let you configure your machine further, but for now, let’s launch this n1-standard-1 machine to test things out. To start the virtual machine, click Create and wait a few seconds.

TESTING OUT YOUR INSTANCE

After your machine is created, you should see a green checkmark in the list of instances in the console. But what can you do with this now? You might notice in the Connect column a button that says “SSH” in the cell. See figure 1.11.



Name	Zone	Recommendation	Internal IP	External IP	Connect
✓ learning-cloud-demo	us-east1-b		10.142.0.2	35.227.93.212	SSH

Figure 1.11 The listing of your VM instances

If you click this button, a new window will pop up, and after waiting a few seconds, you should see a terminal. This terminal is running on your new virtual machine, so feel free to play around—typing `top` or `cat /etc/issue` or anything else that you’re curious about.

1.6.2 *On the command line: gcloud*

Now that you’ve created an instance in the console, you might be curious how the Cloud SDK comes into play. As mentioned earlier, anything that you can do in the Cloud Console can also be done using the `gcloud` command, so let’s put that to the test by looking at the list of your instances, and then connecting to the instance like you did with the SSH button. Let’s start by listing the instances. To do this, type `gcloud compute instances list`. You should see output that looks something like the following snippet:

```
$ gcloud compute instances list
NAME                ZONE          MACHINE_TYPE  PREEMPTIBLE INTERNAL_IP
     EXTERNAL_IP   STATUS
learning-cloud-demo us-central1-b n1-standard-1             10.240.0.2
     104.154.94.41 RUNNING
```

Cool, right? There's your instance that you created, as it appears in the console.

Connecting to your instance

Now that you can see your instance, you probably are curious about how to connect to it like we did with the SSH button. Type `gcloud compute ssh learning-cloud-demo` and choose the zone where you created the machine (`us-central1-b`). You should be connected to your machine via SSH:

```
$ gcloud compute ssh learning-cloud-demo
For the following instances:
 - [learning-cloud-demo]
choose a zone:
 [1] asia-east1-c
 [2] asia-east1-a
 [3] asia-east1-b
 [4] europe-west1-c
 [5] europe-west1-d
 [6] europe-west1-b
 [7] us-central1-f
 [8] us-central1-c
 [9] us-central1-b
 [10] us-central1-a
 [11] us-east1-c
 [12] us-east1-b
 [13] us-east1-d
Please enter your numeric choice:  9

Updated [https://www.googleapis.com/compute/v1/projects/glass-arcade-111313].
Warning: Permanently added '104.154.94.41' (ECDSA) to the list of known hosts.
Linux learning-cloud-demo 3.16.0-0.bpo.4-amd64 #1 SMP Debian 3.16.7-ckt11-
     1+deb8u3~bpo70+1 (2015-08-08) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
jjg@learning-cloud-demo:~$
```

Under the hood, Google is using the credentials it obtained when you ran `gcloud auth login`, generating a new public/private key pair, securely putting the new public key onto the virtual machine, and then using the private key generated to connect to the machine. This means that you don't have to worry about key pairs when connecting. As long as you have access to your Google account, you can always access your virtual machines!

1.6.3 In your own code: google-cloud-*

Now that we've created an instance inside the Cloud Console, then connected to that instance from the command line using the Cloud SDK, let's explore the last way you can interact with your resources: in your own code. What we'll do in this section is write a small Node.js script that connects and terminates your instance. This has the fun side effect of turning off your machine so you don't waste any money during your free trial! To start, if you don't have Node.js installed, you can do that by going to https://nodejs.org and downloading the latest version. You can test that all of this worked by running the `node` command with the `--version` flag:

```
$ node --version
v7.7.1
```

After this, install the Google Cloud client library for Node.js. You can do this with the `npm` command:

```
$ sudo npm install --save @google-cloud/compute@0.7.1
```

Now it's time to start writing some code that connects to your cloud resources. To start, let's try to list the instances currently running. Put the following code into a script called `script.js`, and then run it using `node script.js`.

Listing 1.1 Showing all VMs (script.js)

```
const gce = require('@google-cloud/compute')({
  projectId: 'your-project-id'                    <- Make sure to change
});                                                  this to your project ID!
const zone = gce.zone('us-central1-b');

console.log('Getting your VMs...');

zone.getVMs().then((data) => {
  data[0].forEach((vm) => {
    console.log('Found a VM called', vm.name);
  });
  console.log('Done.');
});
```

If you run this script, the output should look something like the following:

```
$ node script.js
Getting your VMs...
Found a VM called learning-cloud-demo
Done.
```

Now that we know how to list the VMs in a given zone, let's try turning off the VM using our script. To do this, update your code to look like this.

Listing 1.2 Showing and stopping all VMs

```
const gce = require('@google-cloud/compute')({
  projectId: 'your-project-id'
});
const zone = gce.zone('us-central1-b');

console.log('Getting your VMs...');

zone.getVMs().then((data) => {
  data[0].forEach((vm) => {
    console.log('Found a VM called', vm.name);
    console.log('Stopping', vm.name, '...');
    vm.stop((err, operation) => {
      operation.on('complete', (err) => {
        console.log('Stopped', vm.name);
      });
    });
  });
});
```

This script might take a bit longer to run, but when it's complete, the output should look something like the following:

```
$ node script.js
Getting your VMs...
Found a VM called learning-cloud-demo
Stopping learning-cloud-demo ...
Stopped learning-cloud-demo
```

The virtual machine we started in the UI is in a "stopped" state and can be restarted later. Now that we've played with virtual machines and managed them with all of the tools available (the Cloud Console, the Cloud SDK, and your own code), let's keep the ball rolling by learning how to deploy a real application using Google Compute Engine.

Summary

- *Cloud* has become a buzzword, but for this book it's a collection of services that abstract away computer infrastructure.
- Cloud is a good fit if you don't want to manage your own servers or data centers and your needs change often or you don't know them.
- Cloud is a *bad* fit if your usage is steady over long periods of time.
- When in doubt, if you need tools for GCP, start at http://cloud.google.com.

CLOUD

Google Cloud Platform IN ACTION

JJ Geewax

Thousands of developers worldwide trust Google Cloud Platform, and for good reason. With GCP, you can host your applications on the same infrastructure that powers Search, Maps, and the other Google tools you use daily. You get rock-solid reliability, an incredible array of prebuilt services, and a cost-effective, pay-only-for-what-you-use model. This book gets you started.

Google Cloud Platform in Action teaches you how to deploy scalable cloud applications on GCP. Author and Google software engineer JJ Geewax is your guide as you try everything from hosting a simple WordPress web app to commanding cloud-based AI services for computer vision and natural language processing. Along the way, you'll discover how to maximize cloud-based data storage, roll out serverless applications with Cloud Functions, and manage containers with Kubernetes. Broad, deep, and complete, this authoritative book has everything you need.

What's Inside

- The many varieties of cloud storage and computing
- How to make cost-effective choices
- Hands-on code examples
- Cloud-based machine learning

Written for intermediate developers. No prior cloud or GCP experience required.

JJ Geewax is a software engineer at Google, focusing on Google Cloud Platform and API design.

"Demonstrates how to use GCP in practice while also explaining how things work under the hood."

—From the Foreword by Urs Hölzle, SVP, Technical Infrastructure, Google

"Provides powerful insight into Google Cloud, with great worked examples."

—Max Hemingway DXC Technology

"A great asset when migrating to Google Cloud, not only for developers, but for architects and management too."

—Michał Ambroziewicz, Netsprint

"As an Azure user, I got great insights into Google Cloud and a comparison of both providers. A must-read."

—Grzegorz Bernas Antaris Consulting

Free eBook
See first page

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit manning.com/books/google-cloud-platform-in-action

ISBN-13: 978-1-61729-352-8
ISBN-10: 1-61729-352-0
55999
9 781617 293528

MANNING $59.99 / Can $79.99 [INCLUDING eBOOK]