

Amazon Web Services IN ACTION

Andreas Wittig
Michael Wittig

FOREWORD BY Ben Whaley

SAMPLE CHAPTER



MANNING



*Amazon Web Services
in Action*

by Michael Wittig
and Andreas Wittig

Chapter 1

brief contents

PART 1	GETTING STARTED	1
1	■ What is Amazon Web Services? 3	
2	■ A simple example: WordPress in five minutes 34	
PART 2	BUILDING VIRTUAL INFRASTRUCTURE WITH SERVERS AND NETWORKING.....	51
3	■ Using virtual servers: EC2 53	
4	■ Programming your infrastructure: the command line, SDKs, and CloudFormation 91	
5	■ Automating deployment: CloudFormation, Elastic Beanstalk, and OpsWorks 124	
6	■ Securing your system: IAM, security groups, and VPC 152	
PART 3	STORING DATA IN THE CLOUD	183
7	■ Storing your objects: S3 and Glacier 185	
8	■ Storing your data on hard drives: EBS and instance store 204	

- 9 ■ Using a relational database service: RDS 225
- 10 ■ Programming for the NoSQL database service: DynamoDB 253

PART 4 ARCHITECTING ON AWS.....279

- 11 ■ Achieving high availability: availability zones, auto-scaling, and CloudWatch 281
- 12 ■ Decoupling your infrastructure: ELB and SQS 310
- 13 ■ Designing for fault-tolerance 331
- 14 ■ Scaling up and down: auto-scaling and CloudWatch 363

Part 1

Getting started

H ave you watched a blockbuster on Netflix, bought a gadget on [Amazon.com](#), or synced files with Dropbox today? If so, you've used Amazon Web Services (AWS) in the background. As of December 2014, AWS operated 1.4 million servers and therefore is a big player in the cloud computing market. The data centers of AWS are distributed throughout the United States, Europe, Asia, and South America. But the cloud doesn't consist of hardware and computing power alone. Software is part of every cloud platform and makes the difference for you as a customer. The information technology research firm Gartner has classified AWS as a leader in the Magic Quadrant for Cloud Infrastructure as a Service in 2015 for the fourth time. The speed and quality of innovation on the AWS platform is extremely high.

The first part of this book will guide you through your first steps with AWS and give you an idea of how you can use AWS to improve your IT infrastructure. Chapter 1 introduces cloud computing and AWS; you'll learn about key concepts and basics. Chapter 2 brings Amazon Web Service into action; you'll dive into a complex cloud infrastructure with ease.

1 *What is Amazon Web Services?*

This chapter covers

- Overview of Amazon Web Services
- Benefits of using Amazon Web Services
- Examples of what you can do with Amazon Web Services
- Creating and setting up an Amazon Web Services account

Amazon Web Services (AWS) is a platform of web services offering solutions for computing, storing, and networking, at different layers of abstraction. You can use these services to host web sites, run enterprise applications, and mine tremendous amounts of data. The term *web service* means services can be controlled via a web interface. The web interface can be used by machines or by humans via a graphical user interface. The most prominent services are EC2, which offers virtual servers, and S3, which offers storage capacity. Services on AWS work well together; you can use them to replicate your existing on-premises setup or design a new setup from scratch. Services are charged for on a pay-per-use pricing model.

As an AWS customer, you can choose among different data centers. AWS data centers are distributed in the United States, Europe, Asia, and South America. For example, you can start a virtual server in Japan in the same way you can start a virtual server in Ireland. This enables you to serve customers worldwide with a global infrastructure.

The map in figure 1.1 shows the data centers available to all customers.

Which hardware powers AWS?

AWS keeps secret the hardware used in its data centers. The scale at which AWS operates computing, networking, and storage hardware is tremendous. It probably uses commodity components to save money compared to hardware that charges extra for a brand name. Handling of hardware failure is built into real-world processes and software.¹

AWS also uses hardware especially developed for its use cases. A good example is the Xeon E5-2666 v3 CPU from Intel. This CPU is optimized to power virtual servers from the c4 family.

In more general terms, AWS is known as a *cloud computing platform*.

1.1 What is cloud computing?

Almost every IT solution is labeled with the term *cloud computing* or just *cloud* nowadays. A buzzword may help to sell, but it's hard to work with in a book.

Cloud computing, or the cloud, is a metaphor for supply and consumption of IT resources. The IT resources in the cloud aren't directly visible to the user; there are layers of abstraction in between. The level of abstraction offered by the cloud may vary from virtual hardware to complex distributed systems. Resources are available on demand in enormous quantities and paid for per use.



Figure 1.1 AWS data center locations

¹ Bernard Golden, "Amazon Web Services (AWS) Hardware," *For Dummies*, <http://mng.bz/k6lT>.

Here's a more official definition from the National Institute of Standards and Technology:

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

—The NIST Definition of Cloud Computing,
National Institute of Standards and Technology

Clouds are often divided into the following types:

- *Public*—A cloud managed by an organization and open to use by the general public
- *Private*—A cloud that virtualizes and shares the IT infrastructure within a single organization
- *Hybrid*—A mixture of a public and a private cloud

AWS is a public cloud. Cloud computing services also have several classifications:

- *Infrastructure as a service (IaaS)*—Offers fundamental resources like computing, storage, and networking capabilities, using virtual servers such as Amazon EC2, Google Compute Engine, and Microsoft Azure virtual machines
- *Platform as a service (PaaS)*—Provides platforms to deploy custom applications to the cloud, such as AWS Elastic Beanstalk, Google App Engine, and Heroku
- *Software as a service (SaaS)*—Combines infrastructure and software running in the cloud, including office applications like Amazon WorkSpaces, Google Apps for Work, and Microsoft Office 365

The AWS product portfolio contains IaaS, PaaS, and SaaS. Let's take a more concrete look at what you can do with AWS.

1.2 What can you do with AWS?

You can run any application on AWS by using one or a combination of services. The examples in this section will give you an idea of what you can do with AWS.

1.2.1 Hosting a web shop

John is CIO of a medium-sized e-commerce business. His goal is to provide his customers with a fast and reliable web shop. He decided to host the web shop on-premises, and three years ago he rented servers in a data center. A web server handles requests from customers, and a database stores product information and orders. John is evaluating how his company can take advantage of AWS by running the same setup on AWS, as shown in figure 1.2.

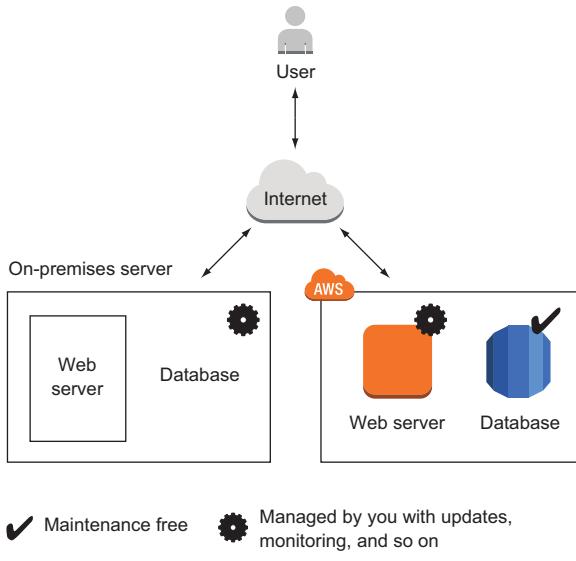


Figure 1.2 Running a web shop on-premises vs. on AWS

John realized that other options are available to improve his setup on AWS with additional services:

- The web shop consists of dynamic content (such as products and their prices) and static content (such as the company logo). By splitting dynamic and static content, John reduced the load for his web servers and improved performance by delivering the static content over a content delivery network (CDN).
- John uses maintenance-free services including a database, an object store, and a DNS system on AWS. This frees him from managing these parts of the system, decreases operational costs, and improves quality.
- The application running the web shop can be installed on virtual servers. John split the capacity of the old on-premises server into multiple smaller virtual servers at no extra cost. If one of these virtual servers fails, the load balancer will send customer requests to the other virtual servers. This setup improves the web shop's reliability.

Figure 1.3 shows how John enhanced the web shop setup with AWS.

John started a proof-of-concept project and found that his web application can be transferred to AWS and that services are available to help improve his setup.

1.2.2 **Running a Java EE application in your private network**

Maureen is a senior system architect in a global corporation. She wants to move parts of the business applications to AWS when the company's data-center contract expires in a few months, to reduce costs and gain flexibility. She found that it's possible to run enterprise applications on AWS.

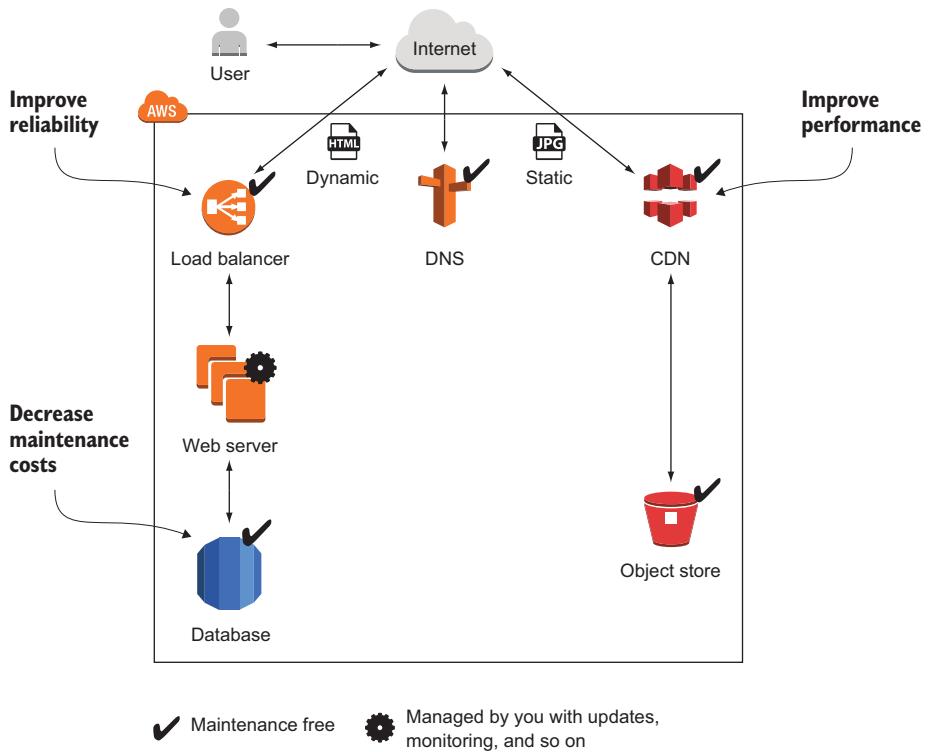


Figure 1.3 Running a web shop on AWS with CDN for better performance, a load balancer for high availability, and a managed database to decrease maintenance costs

To do so, she defines a virtual network in the cloud and connects it to the corporate network through a virtual private network (VPN) connection. The company can control access and protect mission-critical data by using subnets and control traffic between them with access-control lists. Maureen controls traffic to the internet using Network Address Translation (NAT) and firewalls. She installs application servers on virtual machines (VMs) to run the Java EE application. Maureen is also thinking about storing data in a SQL database service (such as Oracle Database Enterprise Edition or Microsoft SQL Server EE). Figure 1.4 illustrates Maureen's architecture.

Maureen has managed to connect the on-premises data center with a private network on AWS. Her team has already started to move the first enterprise application to the cloud.

1.2.3 Meeting legal and business data archival requirements

Greg is responsible for the IT infrastructure of a small law office. His primary goal is to store and archive all data in a reliable and durable way. He operates a file server to

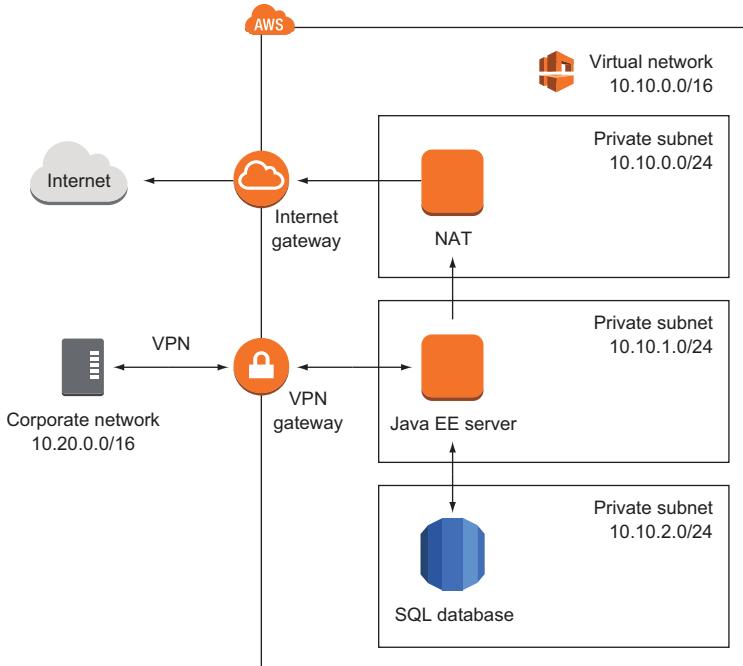


Figure 1.4 Running a Java EE application with enterprise networking on AWS

offer the possibility of sharing documents within the office. Storing all the data is a challenge for him:

- He needs to back up all files to prevent the loss of critical data. To do so, Greg copies the data from the file server to another network-attached storage, so he had to buy the hardware for the file server twice. The file server and the backup server are located close together, so he is failing to meet disaster-recovery requirements to recover from a fire or a break-in.
- To meet legal and business data archival requirements, Greg needs to store data for a long time. Storing data for 10 years or longer is tricky. Greg uses an expensive archive solution to do so.

To save money and increase data security, Greg decided to use AWS. He transferred data to a highly available object store. A storage gateway makes it unnecessary to buy and operate network-attached storage and a backup on-premises. A virtual tape deck takes over the task of archiving data for the required length of time. Figure 1.5 shows how Greg implemented this use case on AWS and compares it to the on-premises solution.

Greg is fine with the new solution to store and archive data on AWS because he was able to improve quality and he gained the possibility of scaling storage size.

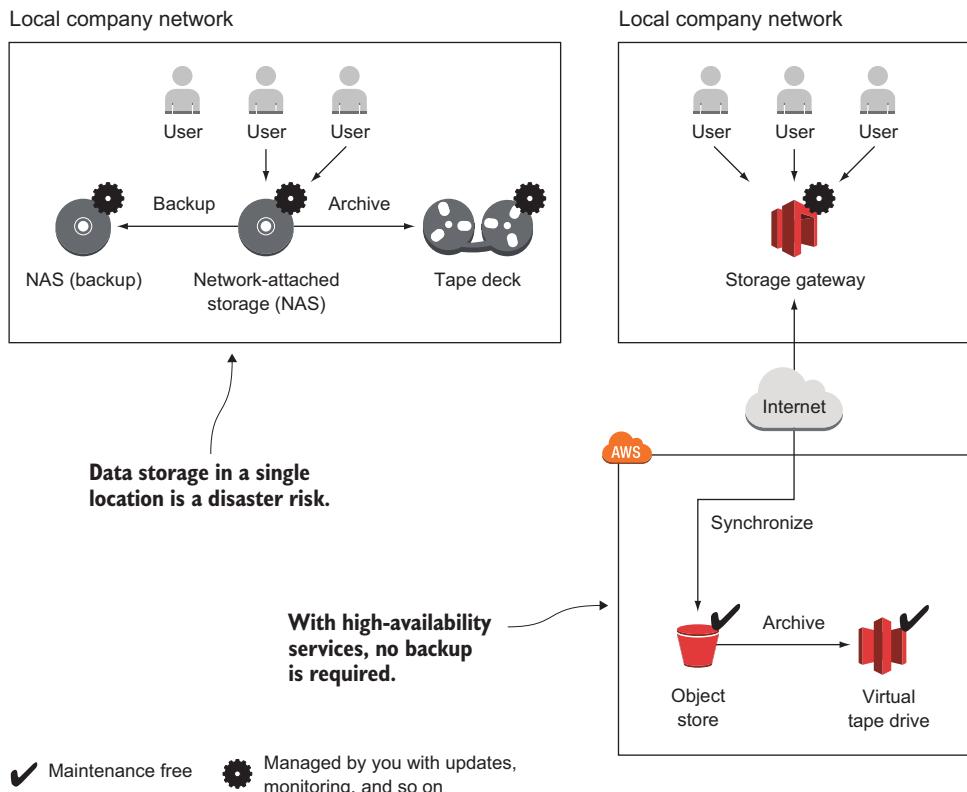


Figure 1.5 Backing up and archiving data on-premises and on AWS

1.2.4 Implementing a fault-tolerant system architecture

Alexa is a software engineer working for a fast-growing startup. She knows that Murphy's Law applies to IT infrastructure: anything that can go wrong, will go wrong. Alexa is working hard to build a fault-tolerant system to prevent outages from ruining the business. She knows that there are two type of services on AWS: fault-tolerant services and services that can be used in a fault-tolerant way. Alexa builds a system like the one shown in figure 1.6 with a fault-tolerant architecture. The database service is offered with replication and failover handling. Alexa uses virtual servers acting as web servers. These virtual servers aren't fault tolerant by default. But Alexa uses a load balancer and can launch multiple servers in different data centers to achieve fault tolerance.

So far, Alexa has protected the startup from major outages. Nevertheless, she and her team are always planning for failure.

You now have a broad idea of what you can do with AWS. Generally speaking, you can host any application on AWS. The next section explains the nine most important benefits AWS has to offer.

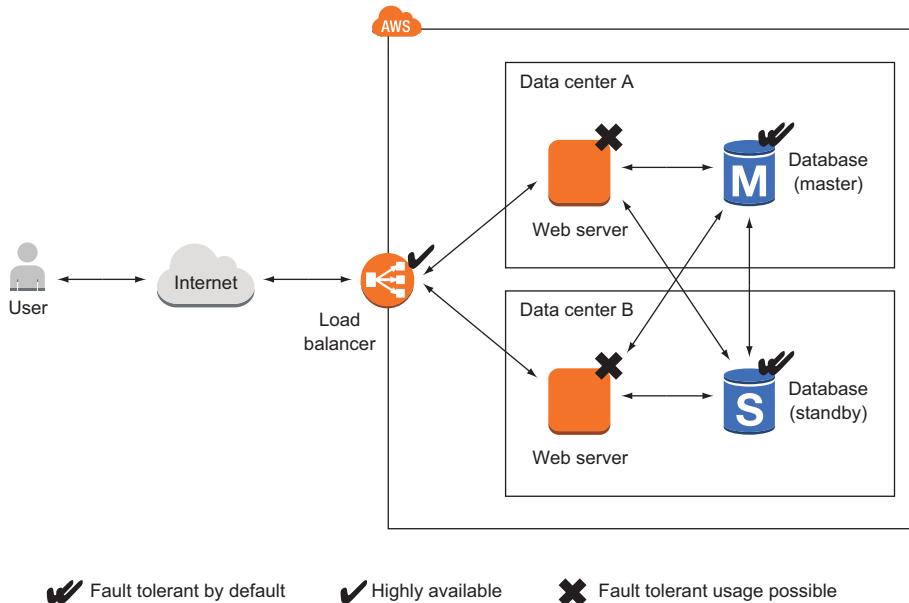


Figure 1.6 Building a fault-tolerant system on AWS

1.3 How you can benefit from using AWS

What's the most important advantage of using AWS? Cost savings, you might say. But saving money isn't the only advantage. Let's look at other ways you can benefit from using AWS.

1.3.1 Innovative and fast-growing platform

In 2014, AWS announced more than 500 new services and features during its yearly conference, re:Invent at Las Vegas. On top of that, new features and improvements are released every week. You can transform these new services and features into innovative solutions for your customers and thus achieve a competitive advantage.

The number of attendees to the re:Invent conference grew from 9,000 in 2013 to 13,500 in 2014.² AWS counts more than 1 million businesses and government agencies among its customers, and in its Q1 2014 results discussion, the company said it will continue to hire more talent to grow even further.³ You can expect even more new features and services in the coming years.

² Greg Bensinger, "Amazon Conference Showcases Another Side of the Retailer's Business," *Digits*, Nov. 12, 2014, <http://mng.bz/hTBo>.

³ "Amazon.com's Management Discusses Q1 2014 Results - Earnings Call Transcript," *Seeking Alpha*, April 24, 2014, <http://mng.bz/60qX>.

1.3.2 Services solve common problems

As you've learned, AWS is a platform of services. Common problems such as load balancing, queuing, sending email, and storing files are solved for you by services. You don't need to reinvent the wheel. It's your job to pick the right services to build complex systems. Then you can let AWS manage those services while you focus on your customers.

1.3.3 Enabling automation

Because AWS has an API, you can automate everything: you can write code to create networks, start virtual server clusters, or deploy a relational database. Automation increases reliability and improves efficiency.

The more dependencies your system has, the more complex it gets. A human can quickly lose perspective, whereas a computer can cope with graphs of any size. You should concentrate on tasks a human is good at—describing a system—while the computer figures out how to resolve all those dependencies to create the system. Setting up an environment in the cloud based on your blueprints can be automated with the help of infrastructure as code, covered in chapter 4.

1.3.4 Flexible capacity (scalability)

Flexible capacity frees you from planning. You can scale from one server to thousands of servers. Your storage can grow from gigabytes to petabytes. You no longer need to predict your future capacity needs for the coming months and years.

If you run a web shop, you have seasonal traffic patterns, as shown in figure 1.7. Think about day versus night, and weekday versus weekend or holiday. Wouldn't it be nice if you could add capacity when traffic grows and remove capacity when traffic shrinks? That's exactly what flexible capacity is about. You can start new servers within minutes and throw them away a few hours after that.

The cloud has almost no capacity constraints. You no longer need to think about rack space, switches, and power supplies—you can add as many servers as you like. If your data volume grows, you can always add new storage capacity.

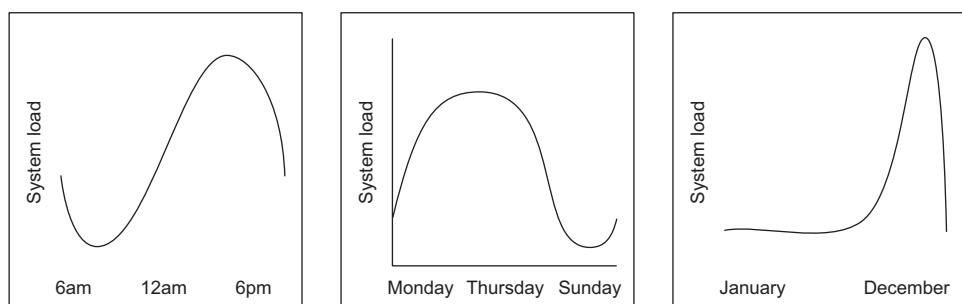


Figure 1.7 Seasonal traffic patterns for a web shop

Flexible capacity also means you can shut down unused systems. In one of our last projects, the test environment only ran from 7:00 a.m. to 8:00 p.m. on weekdays, allowing us to save 60%.

1.3.5 *Built for failure (reliability)*

Most AWS services are fault-tolerant or highly available. If you use those services, you get reliability for free. AWS supports you as you build systems in a reliable way. It provides everything you need to create your own fault-tolerant systems.

1.3.6 *Reducing time to market*

In AWS, you request a new virtual server, and a few minutes later that virtual server is booted and ready to use. The same is true with any other AWS service available. You can use them all on demand. This allows you to adapt your infrastructure to new requirements very quickly.

Your development process will be faster because of the shorter feedback loops. You can eliminate constraints such as the number of test environments available; if you need one more test environment, you can create it for a few hours.

1.3.7 *Benefiting from economies of scale*

At the time of writing, the charges for using AWS have been reduced 42 times since 2008:

- In December 2014, charges for outbound data transfer were lowered by up to 43%.
- In November 2014, charges for using the search service were lowered by 50%.
- In March 2014, charges for using a virtual server were lowered by up to 40%.

As of December 2014, AWS operated 1.4 million servers. All processes related to operations must be optimized to operate at that scale. The bigger AWS gets, the lower the prices will be.

1.3.8 *Worldwide*

You can deploy your applications as close to your customers as possible. AWS has data centers in the following locations:

- United States (northern Virginia, northern California, Oregon)
- Europe (Germany, Ireland)
- Asia (Japan, Singapore)
- Australia
- South America (Brazil)

With AWS, you can run your business all over the world.

1.3.9 *Professional partner*

AWS is compliant with the following:

- *ISO 27001*—A worldwide information security standard certified by an independent and accredited certification body

- *FedRAMP & DoD CSM*—Ensures secure cloud computing for the U.S. Federal Government and the U.S. Department of Defense
- *PCI DSS Level 1*—A data security standard (DSS) for the payment card industry (*PCI*) to protect cardholders data
- *ISO 9001*—A standardized quality management approach used worldwide and certified by an independent and accredited certification body

If you're still not convinced that AWS is a professional partner, you should know that Airbnb, Amazon, Intuit, NASA, Nasdaq, Netflix, SoundCloud, and many more are running serious workloads on AWS.

The cost benefit is elaborated in more detail in the next section.

1.4 How much does it cost?

A bill from AWS is similar to an electric bill. Services are billed based on usage. You pay for the hours a virtual server was running, the used storage from the object store (in gigabytes), or the number of running load balancers. Services are invoiced on a monthly basis. The pricing for each service is publicly available; if you want to calculate the monthly cost of a planned setup, you can use the AWS Simple Monthly Calculator (<http://aws.amazon.com/calculator>).

1.4.1 Free Tier

You can use some AWS services for free during the first 12 months after you sign up. The idea behind the Free Tier is to enable you to experiment with AWS and get some experience. Here is what's included in the Free Tier:

- 750 hours (roughly a month) of a small virtual server running Linux or Windows. This means you can run one virtual server the whole month or you can run 750 virtual servers for one hour.
- 750 hours (or roughly a month) of a load balancer.
- Object store with 5 GB of storage.
- Small database with 20 GB of storage, including backup.

If you exceed the limits of the Free Tier, you start paying for the resources you consume without further notice. You'll receive a bill at the end of the month. We'll show you how to monitor your costs before you begin using AWS. If your Free Tier ends after one year, you pay for all resources you use.

You get some additional benefits, as detailed at <http://aws.amazon.com/free>. This book will use the Free Tier as much as possible and will clearly state when additional resources are required that aren't covered by the Free Tier.

1.4.2 Billing example

As mentioned earlier, you can be billed in several ways:

- *Based on hours of usage*—If you use a server for 61 minutes, that's usually counted as 2 hours.

- *Based on traffic*—Traffic can be measured in gigabytes or in number of requests.
- *Based on storage usage*—Usage can be either provisioned capacity (for example, 50 GB volume no matter how much you use) or real usage (such as 2.3 GB used).

Remember the web shop example from section 1.2? Figure 1.8 shows the web shop and adds information about how each part is billed.

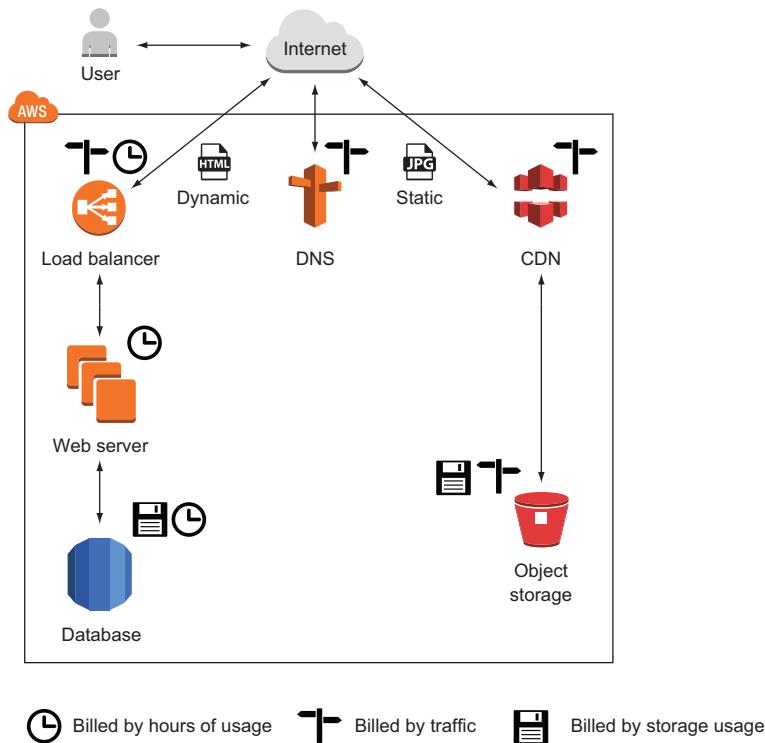


Figure 1.8 Web shop billing example

Let's assume your web shop started successfully in January, and you decided to run a marketing campaign to increase sales for the next month. Lucky you: you were able to increase the number of visitors of your web shop fivefold in February. As you already know, you have to pay for AWS based on usage. Table 1.1 shows your bills for January and February. The number of visitors increased from 100,000 to 500,000, and your monthly bill increased from 142.37 USD to 538.09 USD, which is a 3.7-fold increase. Because your web shop had to handle more traffic, you had to pay more for services, such as the CDN, the web servers, and the database. Other services, like the storage of static files, didn't experience more usage, so the price stayed the same.

With AWS, you can achieve a linear relationship between traffic and costs. And other opportunities await you with this pricing model.

Table 1.1 How an AWS bill changes if the number of web shop visitors increases

Service	January usage	February usage	February charge	Increase
Visits to website	100,000	500,000		
CDN	26 M requests + 25 GB traffic	131 M requests + 125 GB traffic	113.31 USD	90.64 USD
Static files	50 GB used storage	50 GB used storage	1.50 USD	0.00 USD
Load balancer	748 hours + 50 GB traffic	748 hours + 250 GB traffic	20.30 USD	1.60 USD
Web servers	1 server = 748 hours	4 servers = 2,992 hours	204.96 USD	153.72 USD
Database (748 hours)	Small server + 20 GB storage	Large server + 20 GB storage	170.66 USD	128.10 USD
Traffic (outgoing traffic to internet)	51 GB	255 GB	22.86 USD	18.46 USD
DNS	2 M requests	10 M requests	4.50 USD	3.20 USD
Total cost			538.09 USD	395.72 USD

1.4.3 Pay-per-use opportunities

The AWS pay-per-use pricing model creates new opportunities. You no longer need to make upfront investments in infrastructure. You can start servers on demand and only pay per hour of usage; and you can stop using those servers whenever you like and no longer have to pay for them. You don't need to make an upfront commitment regarding how much storage you'll use.

A big server costs exactly as much as two smaller ones with the same capacity. Thus you can divide your systems into smaller parts, because the cost is the same. This makes fault tolerance affordable not only for big companies but also for smaller budgets.

1.5 Comparing alternatives

AWS isn't the only cloud computing provider. Microsoft and Google have cloud offerings as well.

OpenStack is different because it's open source and developed by more than 200 companies including IBM, HP, and Rackspace. Each of these companies uses OpenStack to operate its own cloud offerings, sometimes with closed source add-ons. You could run your own cloud based on OpenStack, but you would lose most of the benefits outlined in section 1.3.

Comparing cloud providers isn't easy, because open standards are mostly missing. Functionality like virtual networks and message queuing are realized differently. If you know what features you need, you can compare the details and make your decision.

Otherwise, AWS is your best bet because the chances are highest that you'll find a solution for your problem.

Following are some common features of cloud providers:

- Virtual servers (Linux and Windows)
- Object store
- Load balancer
- Message queuing
- Graphical user interface
- Command-line interface

The more interesting question is, how do cloud providers differ? Table 1.2 compares AWS, Azure, Google Cloud Platform, and OpenStack.

Table 1.2 Differences between AWS, Microsoft Azure, Google Cloud Platform, and OpenStack

	AWS	Azure	Google Cloud Platform	OpenStack
Number of services	Most	Many	Enough	Few
Number of locations (multiple data centers per location)	9	13	3	Yes (depends on the OpenStack provider)
Compliance	Common standards (ISO 27001, HIPAA, FedRAMP, SOC), IT Grundsatz (Germany), G-Cloud (UK)	Common standards (ISO 27001, HIPAA, FedRAMP, SOC), ISO 27018 (cloud privacy), G-Cloud (UK)	Common standards (ISO 27001, HIPAA, FedRAMP, SOC)	Yes (depends on the OpenStack provider)
SDK languages	Android, Browsers (JavaScript), iOS, Java, .NET, Node.js (JavaScript), PHP, Python, Ruby, Go	Android, iOS, Java, .NET, Node.js (JavaScript), PHP, Python, Ruby	Java, Browsers (JavaScript), .NET, PHP, Python	-
Integration into development process	Medium, not linked to specific ecosystems	High, linked to the Microsoft ecosystem (for example, .NET development)	High, linked to the Google ecosystem (for example, Android)	-
Block-level storage (attached via network)	Yes	Yes (can be used by multiple virtual servers simultaneously)	No	Yes (can be used by multiple virtual servers simultaneously)
Relational database	Yes (MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server)	Yes (Azure SQL Database, Microsoft SQL Server)	Yes (MySQL)	Yes (depends on the OpenStack provider)
NoSQL database	Yes (proprietary)	Yes (proprietary)	Yes (proprietary)	No
DNS	Yes	No	Yes	No

Table 1.2 Differences between AWS, Microsoft Azure, Google Cloud Platform, and OpenStack (continued)

	AWS	Azure	Google Cloud Platform	OpenStack
Virtual network	Yes	Yes	No	Yes
Pub/sub messaging	Yes (proprietary, JMS library available)	Yes (proprietary)	Yes (proprietary)	No
Machine-learning tools	Yes	Yes	Yes	No
Deployment tools	Yes	Yes	Yes	No
On-premises data-center integration	Yes	Yes	Yes	No

In our opinion, AWS is the most mature cloud platform available at the moment.

1.6 Exploring AWS services

Hardware for computing, storing, and networking is the foundation of the AWS cloud. AWS runs software services on top of the hardware to provide the cloud, as shown in figure 1.9. A web interface, the API, acts as an interface between AWS services and your applications.

You can manage services by sending requests to the API manually via a GUI or programmatically via a SDK. To do so, you can use a tool like the Management Console, a web-based user interface, or a command-line tool. Virtual servers have a peculiarity: you can connect to virtual servers through SSH, for example, and gain administrator

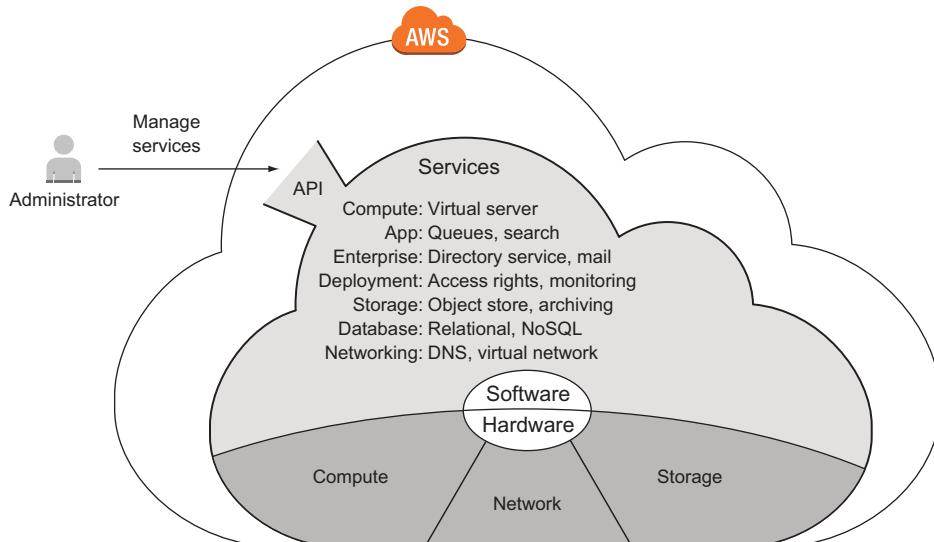


Figure 1.9 The AWS cloud is composed of hardware and software services accessible via an API.

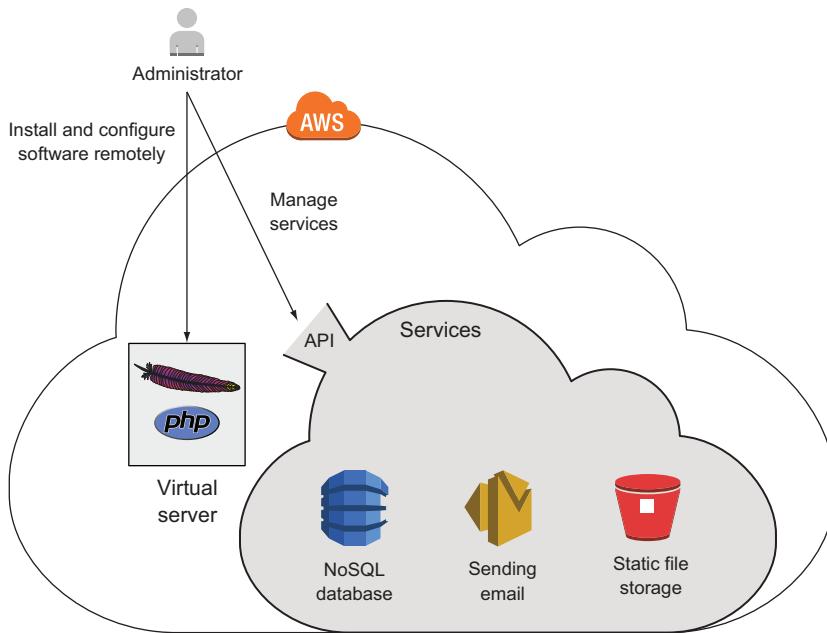


Figure 1.10 Managing a custom application running on a virtual server and dependent services

access. This means you can install any software you like on a virtual server. Other services, like the NoSQL database service, offer their features through an API and hide everything that's going on behind the scenes. Figure 1.10 shows an administrator installing a custom PHP web application on a virtual server and managing dependent services such as a NoSQL database used by the PHP web application.

Users send HTTP requests to a virtual server. A web server is installed on this virtual server along with a custom PHP web application. The web application needs to talk to AWS services in order to answer HTTP requests from users. For example, the web application needs to query data from a NoSQL database, store static files, and send email. Communication between the web application and AWS services is handled by the API, as figure 1.11 shows.

The number of different services available can be scary at the outset. The following categorization of AWS services will help you to find your way through the jungle:

- *Compute services* offer computing power and memory. You can start virtual servers and use them to run your applications.
- *App services* offer solutions for common use cases like message queues, topics, and searching large amounts of data to integrate into your applications.

- *Enterprise services* offer independent solutions such as mail servers and directory services.
- *Deployment and administration services* work on top of the services mentioned so far. They help you grant and revoke access to cloud resources, monitor your virtual servers, and deploy applications.
- *Storage* is needed to collect, persist, and archive data. AWS offers different storage options: an object store or a network-attached storage solution for use with virtual servers.
- *Database storage* has some advantages over simple storage solutions when you need to manage structured data. AWS offers solutions for relational and NoSQL databases.
- *Networking services* are an elementary part of AWS. You can define private networks and use a well-integrated DNS.

Be aware that we cover only the most important categories and services here. Other services are available, and you can also run your own applications on AWS.

Now that we've looked at AWS services in detail, it's time for you to learn how to interact with those services.

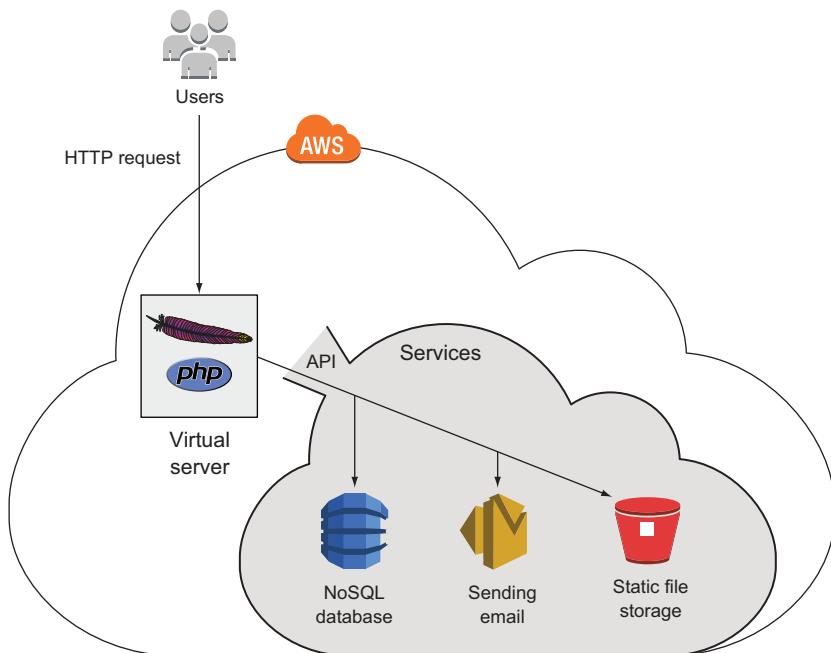


Figure 1.11 Handling an HTTP request with a custom web application using additional AWS services

1.7 Interacting with AWS

When you interact with AWS to configure or use services, you make calls to the API. The API is the entry point to AWS, as figure 1.12 demonstrates.

Next, we'll give you an overview of the tools available to make calls to the AWS API. You can compare the ability of these tools to automate your daily tasks.

1.7.1 Management Console

You can use the web-based Management Console to interact with AWS. You can manually control AWS with this convenient GUI, which runs in every modern web browser (Chrome, Firefox, Safari ≥ 5, IE ≥ 9); see figure 1.13.

If you're experimenting with AWS, the Management Console is the best place to start. It helps you to gain an overview of the different services and achieve success quickly. The Management Console is also a good way to set up a cloud infrastructure for development and testing.

1.7.2 Command-line interface

You can start a virtual server, create storage, and send email from the command line. With the command-line interface (CLI), you can control everything on AWS; see figure 1.14.

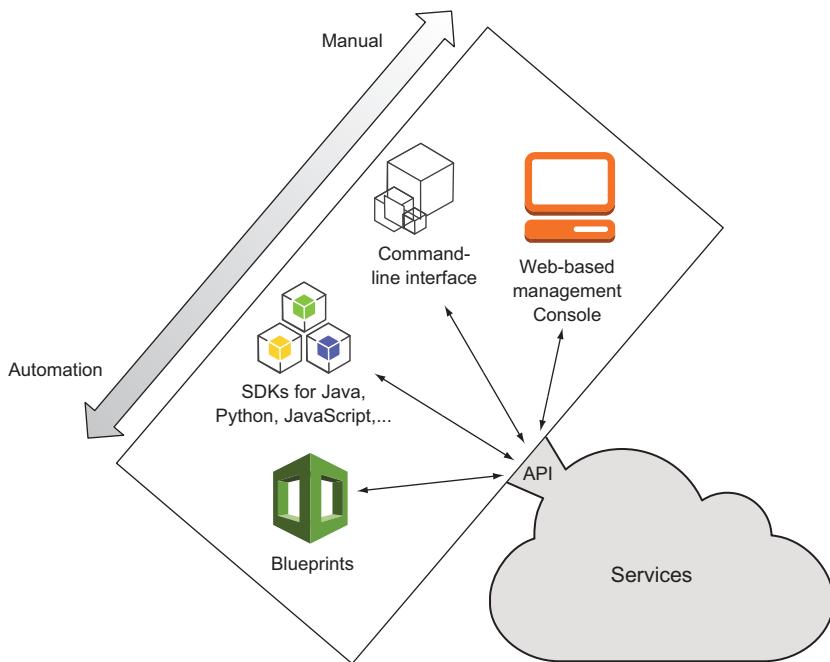


Figure 1.12 Tools to interact with the AWS API

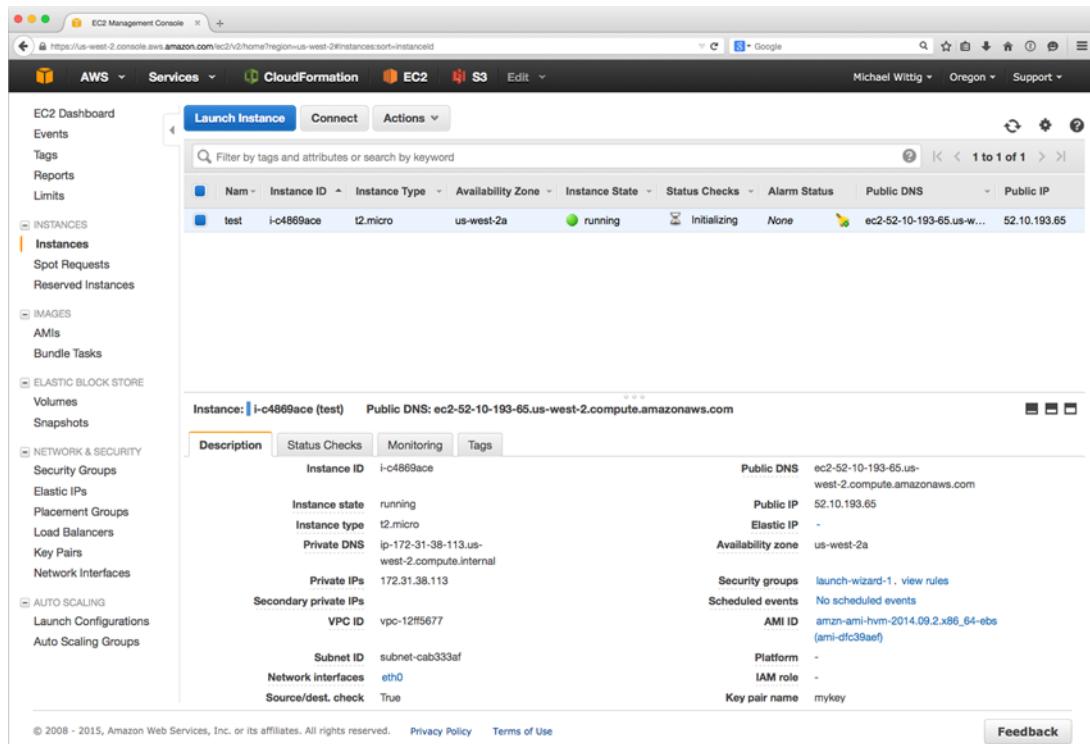


Figure 1.13 Management Console

```
michael ~ bash ~ 92x39
Last login: Fri Feb 20 09:32:45 on ttys000
mwittig:~ michael$ aws cloudwatch list-metrics --namespac "AWS/EC2" --max-items 3
{
    "Metrics": [
        {
            "Namespace": "AWS/EC2",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-ed62dc0b"
                }
            ],
            "MetricName": "StatusCheckFailed_Instance"
        },
        {
            "Namespace": "AWS/EC2",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-ed62dc0b"
                }
            ],
            "MetricName": "StatusCheckFailed"
        },
        {
            "Namespace": "AWS/EC2",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-8a02beec"
                }
            ],
            "MetricName": "CPUUtilization"
        }
    ],
    "NextToken": "None___3"
}
mwittig:~ michael$
```

Figure 1.14 Command-line interface

The CLI is typically used to automate tasks on AWS. If you want to automate parts of your infrastructure with the help of a continuous integration server like Jenkins, the CLI is the right tool for the job. The CLI offers a convenient way to access the API and combine multiple calls into a script.

You can even begin to automate your infrastructure with scripts by chaining multiple CLI calls together. The CLI is available for Windows, Mac, and Linux, and there's also a PowerShell version available.

1.7.3 SDKs

Sometimes you need to call AWS from within your application. With SDKs, you can use your favorite programming language to integrate AWS into your application logic. AWS provides SDKs for the following:

- Android
- Browsers (JavaScript)
- iOS
- Java
- .NET
- Node.js (JavaScript)
- PHP
- Python
- Ruby
- Go

SDKs are typically used to integrate AWS services into applications. If you're doing software development and want to integrate an AWS service like a NoSQL database or a push-notification service, an SDK is the right choice for the job. Some services, such as queues and topics, must be used with an SDK in your application.

1.7.4 Blueprints

A *blueprint* is a description of your system containing all services and dependencies. The blueprint doesn't say anything about the necessary steps or the order to achieve the described system. Figure 1.15 shows how a blueprint is transferred into a running system.

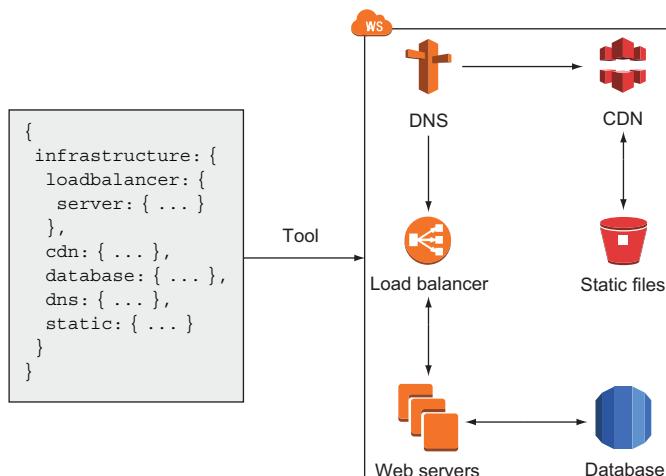


Figure 1.15 Infrastructure automation with blueprints

Consider using blueprints if you have to control many or complex environments. Blueprints will help you to automate the configuration of your infrastructure in the cloud. You can use blueprints to set up virtual networks and launch different servers into that network, for example.

A blueprint removes much of the burden from you because you no longer need to worry about dependencies during system creation—the blueprint automates the entire process. You’ll learn more about automating your infrastructure in chapter 4.

It’s time to get started creating your AWS account and exploring AWS practice after all that theory.

1.8 **Creating an AWS account**

Before you can start using AWS, you need to create an account. An AWS account is a basket for all the resources you own. You can attach multiple users to an account if multiple humans need access to the account; by default, your account will have one root user. To create an account, you need the following:

- A telephone number to validate your identity
- A credit card to pay your bills

Using an old account?

You can use your existing AWS account while working on the examples in this book. In this case, your usage may not be covered by the Free Tier, and you may have to pay for your usage.

Also, if you created your existing AWS account before December 4, 2013, you should create a new one: there are legacy issues that may cause trouble when you try our examples.

1.8.1 **Signing up**

The sign-up process consists of five steps:

- 1 Provide your login credentials.
- 2 Provide your contact information.
- 3 Provide your payment details.
- 4 Verify your identity.
- 5 Choose your support plan.

Point your favorite modern web browser to <https://aws.amazon.com>, and click the Create a Free Account / Create an AWS Account button.

1. PROVIDING YOUR LOGIN CREDENTIALS

The Sign Up page, shown in figure 1.16, gives you two choices. You can either create an account using your Amazon.com account or create an account from scratch. If you create the account from scratch, follow along. Otherwise, skip to step 5.

Fill in your email address, and select I Am a New User. Go on to the next step to create your login credentials. We advise you to choose a strong password to prevent misuse

The screenshot shows the 'Sign In or Create an AWS Account' page. At the top, it says 'My e-mail address is:' followed by an input field. Below that, there are two radio button options: 'I am a new user.' (selected) and 'I am a returning user and my password is:' followed by another input field. A yellow 'Sign in using our secure server' button is present. At the bottom, there are links for 'Forgot your password?' and 'Has your e-mail address changed?'. The background features a light gray gradient.

Figure 1.16 Creating an AWS account: Sign Up page

of your account. We suggest a password with 16 characters, numbers, and symbols. If someone gets access to your account, they can destroy your systems or steal your data.

2. PROVIDING YOUR CONTACT INFORMATION

The next step, as shown in figure 1.17, is to provide your contact information. Fill in all the required fields, and continue.

The screenshot shows the 'Contact Information' page. It includes fields for 'Full Name*', 'Company Name', 'Country*' (set to United States), 'Address*', 'City*', 'State / Province or Region*', 'Postal Code*', 'Phone Number*', and 'Security Check' (with a CAPTCHA image showing '8TGATC'). Below the CAPTCHA is a link 'Refresh Image'. There is also a text field 'Please type the characters as shown above' and a corresponding input field. At the bottom, there is an 'AWS Customer Agreement' checkbox and a note: 'Check here to indicate that you have read and agree to the terms of the AWS Customer Agreement'. A yellow 'Create Account and Continue' button is at the very bottom.

Figure 1.17 Creating an AWS account: providing your contact information

The screenshot shows the 'Payment Information' step of the AWS account creation process. At the top, there's a navigation bar with five tabs: 'Contact Information' (with a checkmark), 'Payment Information' (highlighted in orange), 'Identity Verification', 'Support Plan', and 'Confirmation'. Below the tabs, the title 'Payment Information' is centered. A descriptive text box says: 'Please enter your payment information below. You will be able to try a broad set of AWS products for free via the Free Usage Tier. We will only bill your credit card for usage that is not covered by our Free Usage Tier.' A table provides details about the AWS Free Usage Tier: it's free for 1 year, includes 750hrs/month* for Compute (Amazon EC2), 5GB for Storage (Amazon S3), and 750hrs/month* for Database (Amazon RDS). There's a link to 'View full offer details ». Below the table, there are fields for 'Credit Card Number' (a large input field) and 'Expiration Date' (two dropdown menus for month and year). A field for 'Cardholder's Name' is also present. Underneath these fields, a section titled 'Choose Your Billing Address' asks to select the address associated with the credit card. It offers two options: a radio button for 'Use my contact address' (selected) and another for 'Use a new address'. A small note indicates that the billing address must be a valid US address. At the bottom right is a large yellow 'Continue' button.

Figure 1.18 Creating an AWS account: providing your payment details

3. PROVIDE YOUR PAYMENT DETAILS

Now the screen shown in figure 1.18 asks for your payment information. AWS supports MasterCard and Visa. You can set your preferred payment currency later, if you don't want to pay your bills in USD; supported currencies are EUR, GBP, CHF, AUD, and some others.

4. VERIFYING YOUR IDENTITY

The next step is to verify your identity. Figure 1.19 shows the first step of the process.

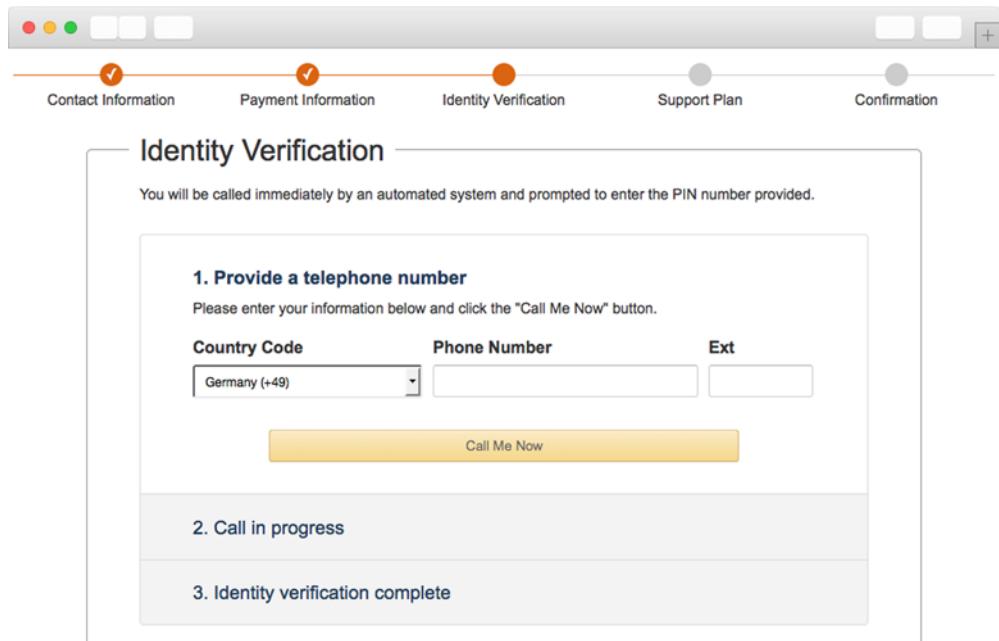


Figure 1.19 Creating an AWS account: verifying your identity (1 of 2)

After you complete the first part, you'll receive a call from AWS. A robot voice will ask you for your PIN, which will be like the one shown in figure 1.20. Your identity will be verified, and you can continue with the last step.

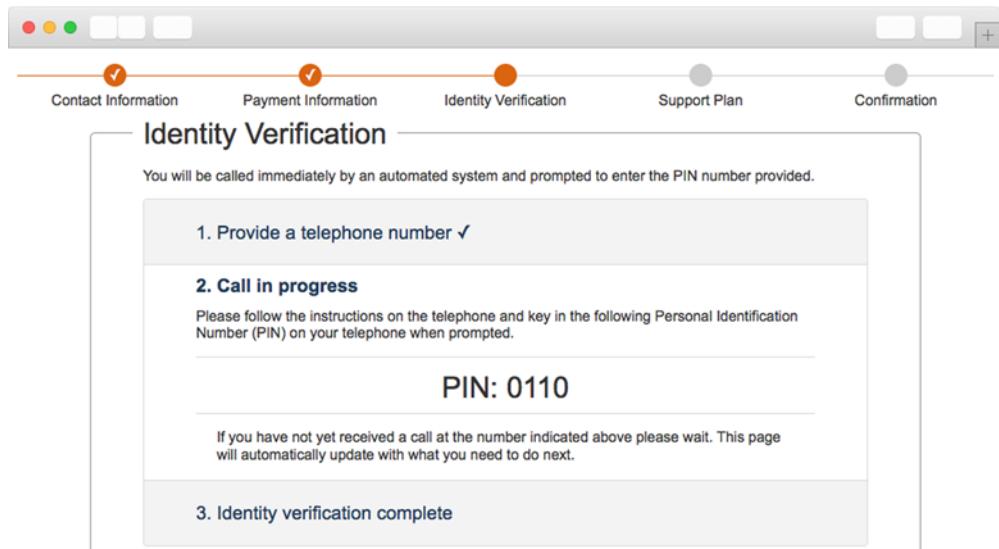


Figure 1.20 Creating an AWS account: verifying your identity (2 of 2)

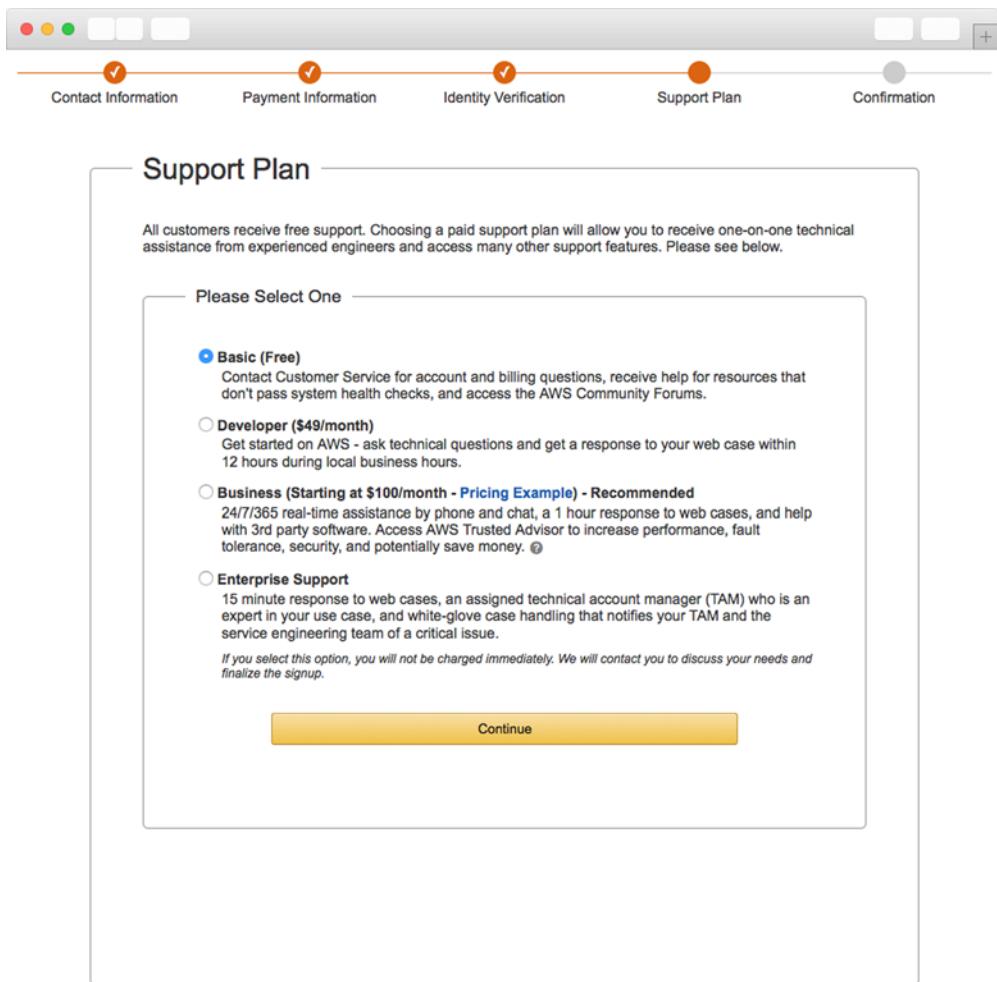


Figure 1.21 Creating an AWS account: choosing your support plan

5. CHOOSING YOUR SUPPORT PLAN

The last step is to choose a support plan; see figure 1.21. In this case, select the Basic plan, which is free. If you later create an AWS account for your business, we recommend the Business support plan. You can even switch support plans later.

High five! You're done. Now you can log in to your account with the AWS Management Console.

1.8.2 Signing In

You have an AWS account and are ready to sign in to the AWS Management Console at <https://console.aws.amazon.com>. As mentioned earlier, the Management Console is a web-based tool you can use to control AWS resources. The Management Console

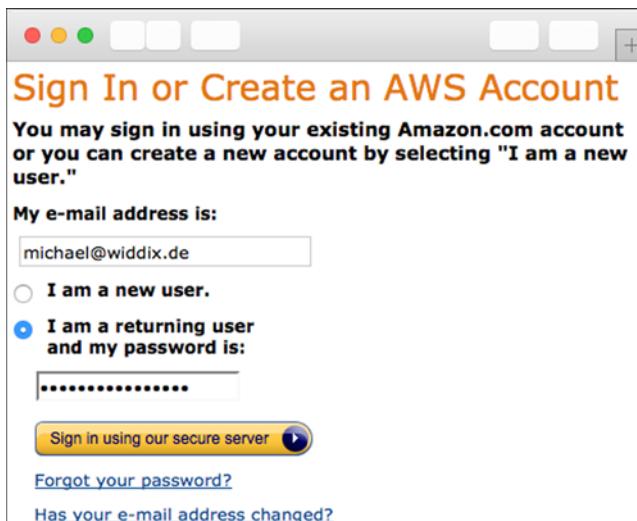


Figure 1.22 Sign in to the Management Console.

uses the AWS API to make most of the functionality available to you. Figure 1.22 shows the Sign In page.

Enter your login credentials and click Sign In Using Our Secure Server to see the Management Console, shown in figure 1.23.

The screenshot shows the AWS Management Console homepage. The top navigation bar includes "AWS", "Services", "Edit", and user information "Michael Wittig · Oregon · Support". The main content area is titled "Amazon Web Services" and lists several service categories:

- Compute**: EC2, Lambda
- Storage & Content Delivery**: S3, Storage Gateway, Glacier, CloudFront
- Database**: RDS, DynamoDB, ElastiCache, Redshift
- Networking**: VPC, Direct Connect, Route 53
- Administration & Security**: Directory Service, Identity & Access Management, Trusted Advisor, CloudTrail, Config, CloudWatch
- Deployment & Management**: Elastic Beanstalk, OpsWorks, CloudFormation, CodeDeploy
- Analytics**: EMR, Kinesis, Data Pipeline
- Application Services**: SQS, SWF, AppStream, Elastic Transcoder, SES, CloudSearch
- Mobile Services**: Cognito, Mobile Analytics, SNS
- Enterprise Applications**: WorkSpaces, WorkDocs, WorkMail
- Additional Resources**: Getting Started, AWS Console Mobile App, AWS Marketplace, Service Health

At the bottom, there's a note: "All services operating normally. Updated: Feb 11 2015 10:11:00 GMT+0100".

Figure 1.23 AWS Management Console

The most important part is the navigation bar at the top; see figure 1.24. It consists of six sections:

- *AWS*—Gives you a fast overview of all resources in your account.
- *Services*—Provides access to all AWS services.
- *Custom section (Edit)*—Click Edit and drag-and-drop important services here to personalize the navigation bar.
- *Your name*—Lets you access billing information and your account, and also lets you sign out.
- *Your region*—Lets you choose your region. You'll learn about regions in section 3.5. You don't need to change anything here now.
- *Support*—Gives you access to forums, documentation, and a ticket system.

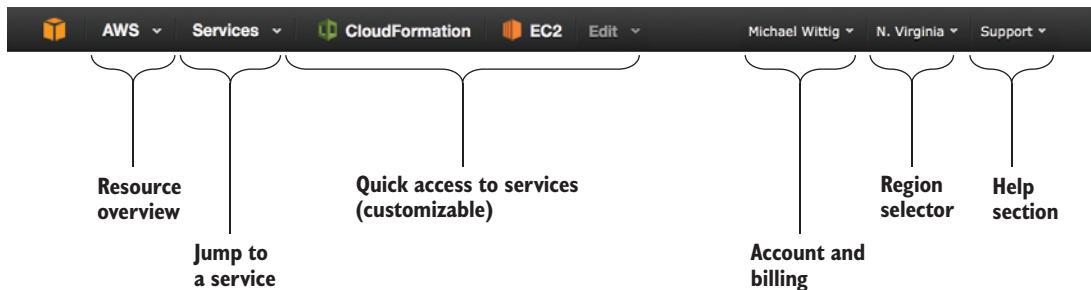


Figure 1.24 AWS Management Console navigation bar

Next, you'll create a key pair so you can connect to your virtual servers.

1.8.3 Creating a key pair

To access a virtual server in AWS, you need a *key pair* consisting of a private key and a public key. The public key will be uploaded to AWS and inserted into the virtual server. The private key is yours; it's like your password, but much more secure. Protect your private key as if it's a password. It's your secret, so don't lose it—you can't retrieve it.

To access a Linux server, you use the SSH protocol; you'll authenticate with the help of your key pair instead of a password during login. You access a Windows server via Remote Desktop Protocol (RDP); you'll need your key pair to decrypt the administrator password before you can log in.

The following steps will guide you to the dashboard of the EC2 service, which offers virtual servers, and where you can obtain a key pair:

- 1 Open the AWS Management Console at <https://console.aws.amazon.com>.
- 2 Click Services in the navigation bar, find the EC2 service, and click it.
- 3 Your browser shows the EC2 Management Console.

The EC2 Management Console, shown in figure 1.25, is split into three columns. The first column is the EC2 navigation bar; because EC2 is one of the oldest services, it has many

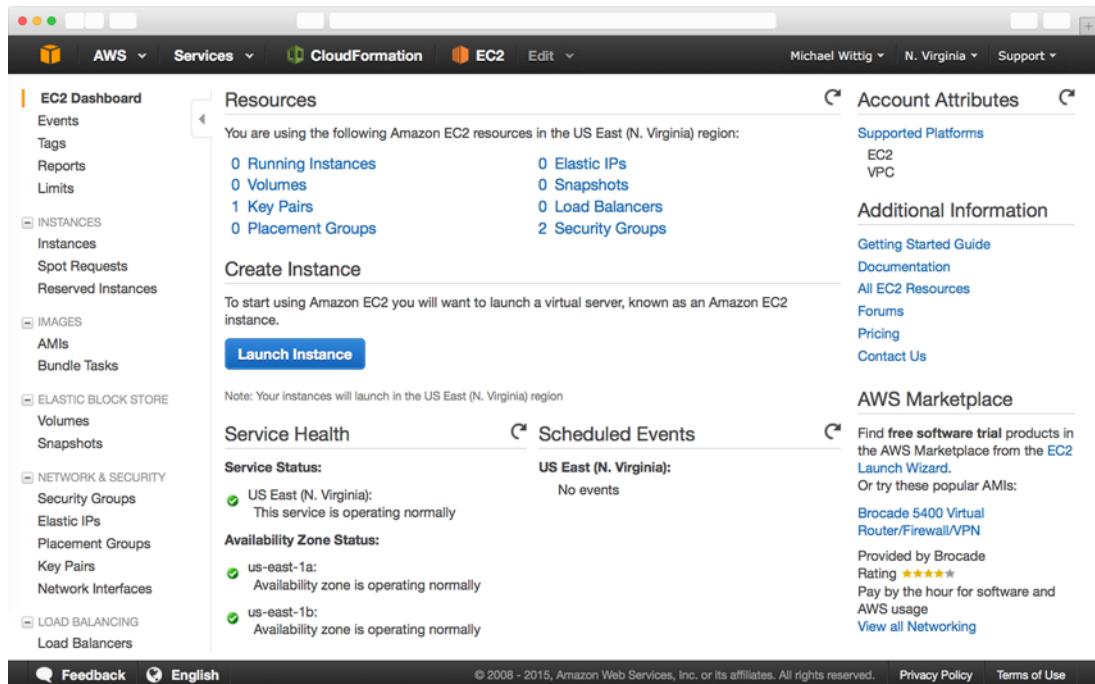


Figure 1.25 EC2 Management Console

features that you can access via the navigation bar. The second column gives you a brief overview of all your EC2 resources. The third column provides additional information.

Follow these steps to create a new key pair:

- 1 Click Key Pairs in the navigation bar under Network & Security.
- 2 Click the Create Key Pair button on the page shown in figure 1.26.
- 3 Name the Key Pair `mykey`. If you choose another name, you must replace the name in all the following examples!

During key-pair creation, you downloaded a file called `mykey.pem`. You must now prepare that key for future use. Depending on your operating system, you may need to do things differently, so please read the section that fits your OS.

Using your own key pair

It's also possible to upload the public key part from an existing key pair to AWS. Doing so has two advantages:

- You can reuse an existing key pair.
- You can be sure that only you know the private key part of the key pair. If you use the Create Key Pair button, AWS knows (at least briefly) your private key.

We decided against that approach in this case because it's less convenient to implement in a book.

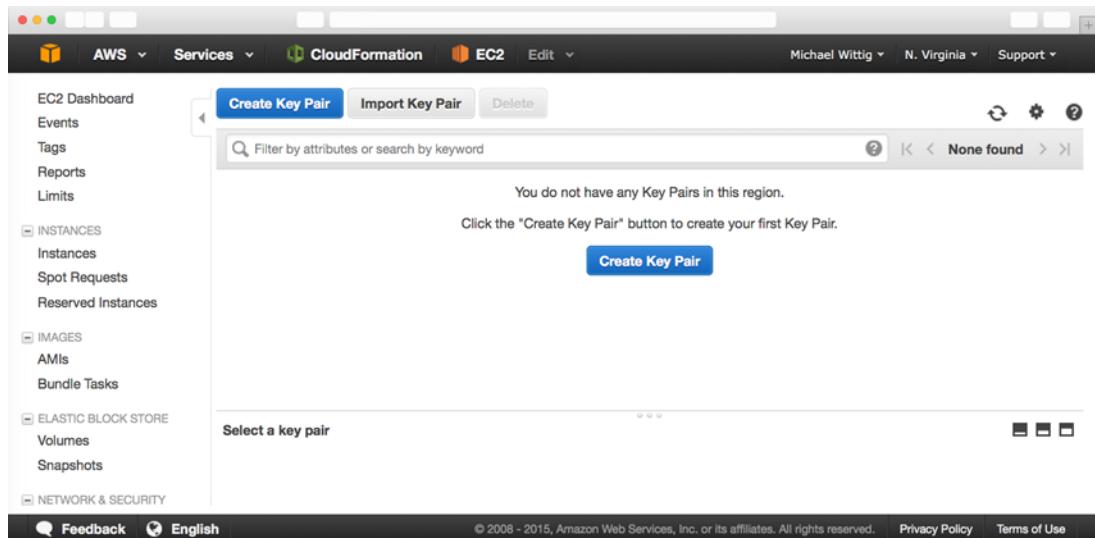


Figure 1.26 EC2 Management Console key pairs

LINUX AND MAC OS X

The only thing you need to do is change the access rights of mykey.pem so that only you can read the file. To do so, run `chmod 400 mykey.pem` in the terminal. You'll learn about how to use your key when you need to log in to a virtual server for the first time in this book.

WINDOWS

Windows doesn't ship a SSH client, so you need to download the PuTTY installer for Windows from <http://mng.bz/A1bY> and install PuTTY. PuTTY comes with a tool called PuTTYgen that can convert the mykey.pem file into a mykey.ppk file, which you'll need:

- 1 Run the application PuTTYgen. The screen shown in figure 1.27 opens.
- 2 Select SSH-2 RSA under Type of Key to Generate.
- 3 Click Load.
- 4 Because PuTTYgen displays only *.ppk files, you need to switch the file extension of the File Name field to All Files.
- 5 Select the mykey.pem file, and click Open.
- 6 Confirm the dialog box.
- 7 Change Key Comment to mykey.
- 8 Click Save Private Key. Ignore the warning about saving the key without a passphrase.

Your .pem file is now converted to the .ppk format needed by PuTTY. You'll learn how to use your key when you need to log in to a virtual server for the first time in this book.

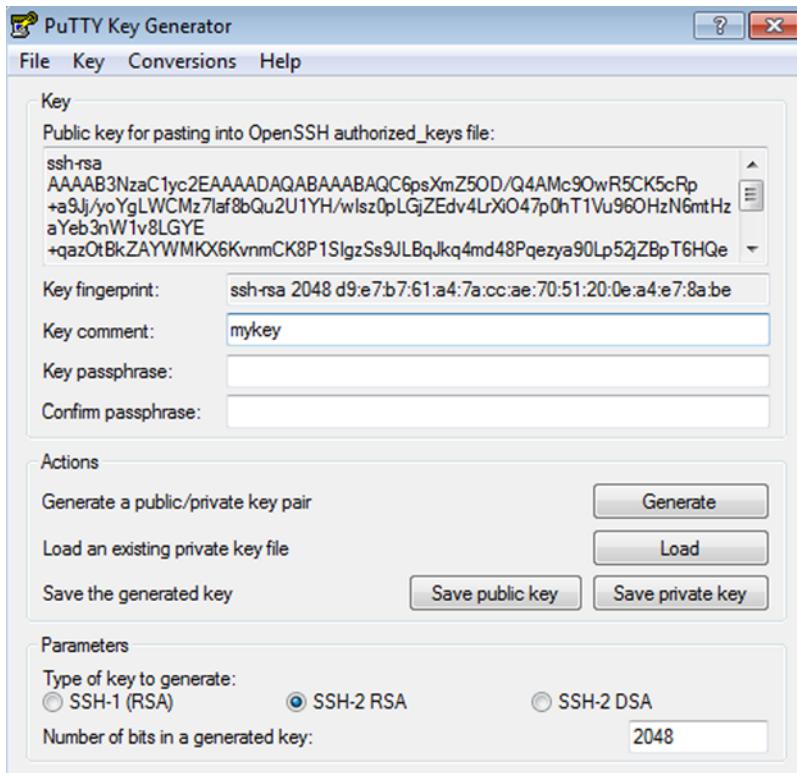


Figure 1.27 PuTTYgen allows you to convert the downloaded .pem file into the .pk file format needed by PuTTY.

1.8.4 Creating a billing alarm

Before you use your AWS account in the next chapter, we advise you to create a billing alarm. If you exceed the Free Tier, an email is sent to you. The book warns you whenever an example isn't covered by the Free Tier. Please make sure that you carefully follow the cleanup steps after each example. To make sure you haven't missed something during cleanup, please create a billing alarm as advised by AWS: <http://mng.bz/M7Sj>.

1.9 Summary

- Amazon Web Services (AWS) is a platform of web services offering solutions for computing, storing, and networking that work well together.
- Cost savings aren't the only benefit of using AWS. You'll also profit from an innovative and fast-growing platform with flexible capacity, fault-tolerant services, and a worldwide infrastructure.
- Any use case can be implemented on AWS, whether it's a widely used web application or a specialized enterprise application with an advanced networking setup.

- You can interact with AWS in many different ways. You can control the different services by using the web-based GUI; use code to manage AWS programmatically from the command line or SDKs; or use blueprints to set up, modify, or delete your infrastructure on AWS.
- Pay-per-use is the pricing model for AWS services. Computing power, storage, and networking services are billed similarly to electricity.
- Creating an AWS account is easy. Now you know how to set up a key pair so you can log in to virtual servers for later use.

Amazon Web Services IN ACTION

Andreas and Michael Wittig

Physical data centers require lots of equipment and take time and resources to manage. If you need a data center, but don't want to build your own, Amazon Web Services may be your solution. Whether you're analyzing real-time data, building software as a service, or running an e-commerce site, AWS offers you a reliable cloud-based platform with services that scale.

Amazon Web Services in Action introduces you to computing, storing, and networking in the AWS cloud. You'll start with an overview of cloud computing and then begin setting up your account. You'll learn how to automate your infrastructure by programmatically calling the AWS API to control every part of AWS. Next, you'll learn options and techniques for storing your data. You'll also learn how to isolate your systems using private networks to increase security. Finally, this book teaches you how to design for high availability and fault tolerance.

What's Inside

- Overview of cloud concepts and patterns
- Deploy applications on AWS
- Integrate Amazon's pre-built services
- Manage servers on EC2 for cost-effectiveness

Written for developers and DevOps engineers moving distributed applications to the AWS platform.

Andreas Wittig and **Michael Wittig** are software engineers and consultants focused on AWS and web development.

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

“A confident, practical guide through the maze of the industry’s leading cloud platform.”

—From the Foreword
by Ben Whaley

“Fantastic introduction to cloud basics with excellent real-world examples.”

—Rambabu Posa, GL Assessment

“A very thorough and practical guide to everything AWS ... highly recommended.”

—Scott M. King, Amazon

“Cuts through the vast expanse of official documentation and gives you what you need to make AWS work now!”

—Carm Vecchio, Computer Science Corporation (CSC)

Free eBook
SEE INSERT

ISBN 13: 978-1-61729-288-0
ISBN 10: 1-61729-288-5

5 4 9 9 9



9 7 8 1 6 1 7 2 9 2 8 8 0



MANNING

\$49.99 / Can \$57.99 [INCLUDING eBook]