2.1.3 Controlling remote servers with a cloud API

The API is to a cloud what the dashboard and controls are to a car. You have tremendous power under that hood, but you need the dials and readouts to know what the vehicle is doing. You need the steering wheel, accelerator, and brake to control it. Remember, you'd never drive fast if you didn't have good brakes.

When you have a cloud, you need a way to access it. The highest-level clouds—those offering Software as a Service (SaaS) applications—offer a browser-based web interface. Lower-level clouds—those offering Infrastructure as a Service (IaaS)—also need a way to access applications. Each type of cloud must provide some kind of API that can be used to provision resources, configure and control them, and release them when they're no longer needed.

An API is necessary to engage the service of a cloud provider. It's a way for the vendor to expose service features and potentially enable competitive differentiation. For example, Amazon's EC2 API is a SOAP- and HTTP Query-based API used to send proprietary commands to create, store, provision, and manage Amazon Machine Images (AMIs). Sun's Project Kenai Cloud API specification is a Representational State Transfer (REST)-ful API for creating and managing cloud resources, including compute, storage, and networking components.

> **REST ARCHITECTURE AND RESTFUL APIS** Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems, such as the World Wide Web. The REST architectural style was developed in parallel with the HTTP protocol. The largest-known implementation of a system conforming to the REST architectural style is the World Wide Web. In fact, REST can be considered a post hoc description of the features of the web that made the web successful. REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of *representations* of *resources*. A resource can be any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource. Conforming to the REST constraints is referred to as being *RESTful*.

Because your cloud applications will be the lifeblood of your company, you'll want to ensure that only authorized parties can access your applications. If an application was running in your company's secure data center protected by layers of physical and logical security you'd be certain that no unauthorized person could access it. Here, because everything having to do with your application and the server it runs on is by definition accessible over the internet, the approach Amazon and others take to security is to issue X.509 public key pairs initially and then require a key on every API call. This ensures that the caller has the credentials to access the infrastructure.

To understand a cloud API—for which there isn't yet an accepted standard—it's best to look at Amazon's cloud API as the default standard as they're the leaders. Table 2.2 outlines some of the basic definitions and operations central to the Amazon cloud API.

Table 2.2 Basic terms and operations of the Amazon EC2 API

Term	Description
AMI	An Amazon Machine Image is an encrypted and signed machine image suitable to run in a virtual server environment. For example, it may contain Linux, Apache, MySQL, or PHP, as well as the application of the AMI's owner. AMIs can be public (provided by Amazon), private (custom designed by its creator), paid (purchased from a third party), or shared (created by the community for free). AMIs can be stored in Amazon's Simple Storage Service (S3).
Instance	The result of launching an AMI is a running system called an *instance*. When an instance terminates, the data on that instance vanishes. For all intents and purposes, an Instance is identical to a traditional host computer.
Standard flow	1. Use a standard AMI by customizing an existing one. 2. Bundle the AMI, and get an AMI ID to enable launching as many instances of the AMI as needed. 3. Launch one or more instances of this AMI. 4. Administer and use the running instance(s).
Connecting	From a web browser, go to http://<hostname>, where <hostname> is your instance's public hostname. If you want to connect to a just-launched public AMI that hasn't been modified, run the `ec2-get-console-output` command. The result in either case enables you to log in as root and exercise full control over this instance, just like any host computer you could walk up to in a data center.

We've barely scratched the surface of all the concepts and corresponding API calls that exist in Amazon's API. Documentation is available at http://docs.amazonwebservices.com. APIs also cover these areas:

- Using instance addressing
- Using network security
- Using regions and availability zones
- Using Amazon Elastic Block Store (EBS)
- Using auto scaling, elastic load balancing, and Amazon CloudWatch
- Using public data sets
- Using Amazon's Virtual Private Cloud

We'll revisit select aspects of the cloud API at various points throughout the book. Let's leave this now and talk about the next important layer in what it takes to set up and use a cloud: cloud storage.