APPENDIX

CLOUD APPLICATION DEVELOPMENT

A

In the previous chapters our discussion was focused on research issues in cloud computing; now we examine computer clouds from the perspective of an application developer. This chapter presents a few recipes useful to assemble a cloud computing environment on a local system and to use basic cloud functions.

It is fair to assume that the population of application developers and cloud users is, and will continue to be, very diverse. Some cloud users have developed and run parallel applications on clusters or other types of systems for many years and expect an easy transition to the cloud. Others, are less experienced, but willing to learn and expect a smooth learning curve. Many view cloud computing as an opportunity to develop new businesses with minimum investment in computing equipment and human resources.

The questions we address are: How easy is it to use the cloud? How knowledgeable should an application developer be about networking and security? How easy is it to port an existing application to the cloud? How easy is it to develop a new cloud application?

The answers to these questions are different for the three cloud delivery models, SaaS, PaaS, and IaaS; the level of difficulty increases as we move towards the base of the cloud service pyramid as shown in Figure A.1. Recall that SaaS applications are designed for the end-users and are accessed over the web; in this case the user must be familiar with the API of a particular application. PaaS provides a set of tools and services designed to facilitate application coding and deploying, while IaaS provides the hardware and the software for servers, storage, networks, including operating systems and

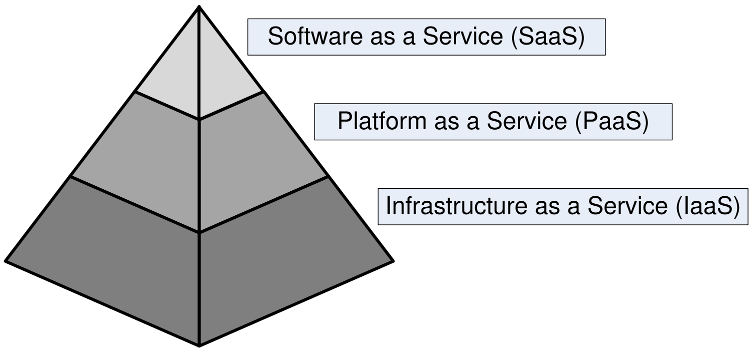

Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)

FIGURE A.1

A pyramid model of cloud computing paradigms; the infrastructure provides the basic resources, the platform adds an environment to facilitate the use of these resources, while software allows direct access to services.

Cloud Computing. DOI: 10.1016/B978-0-12-812810-7.00025-X

storage management software. We restrict our discussion to the IaaS cloud computing model and we concentrate on popular services offered by AWS.

Though the AWS services are well documented, the environment they provide for cloud computing requires some effort to benefit from the full spectrum of services offered. In this section we report on lessons learned from the experience of a group of students with a strong background in programming, networking, and operating systems; each one of them was asked to develop a cloud application for a problem of interest in their own research area. First, we discuss several issues related to cloud security, a major stumbling block for many cloud users; then we present a few recipes for the development of cloud applications, and finally we analyze several cloud applications developed by individuals in this group over a period of less than three months.

A.1 AWS EC2 INSTANCES

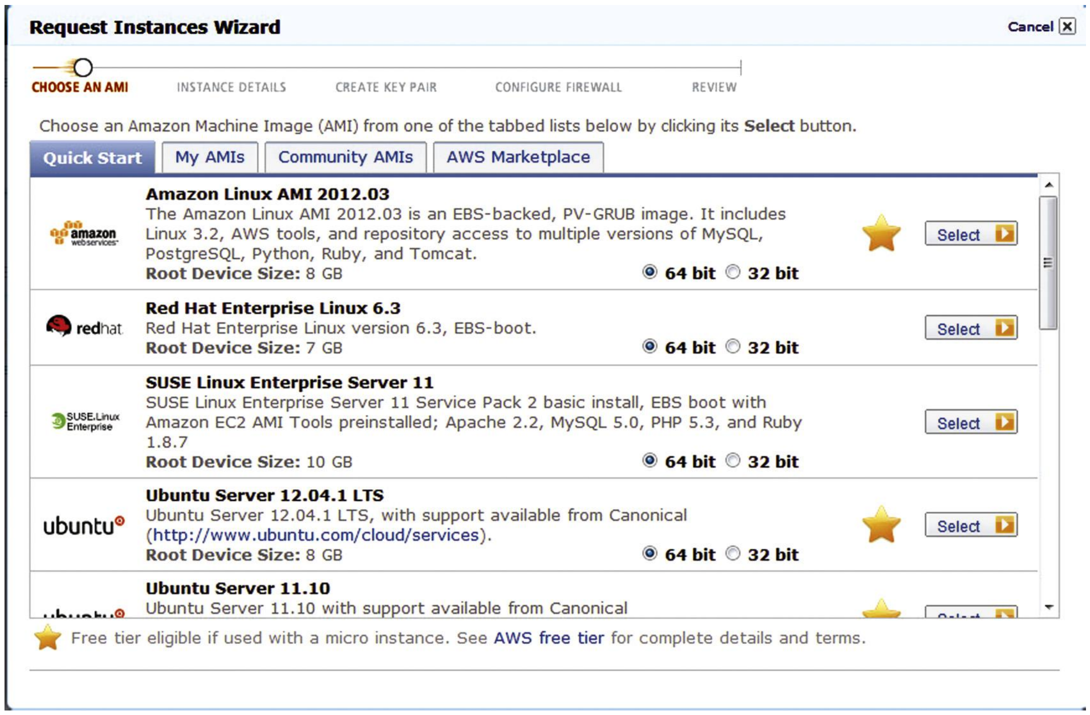
In spite of the wealth of information available from the providers of cloud services, the learning curve of an application developer is still relatively steep. The examples discussed in this chapter are designed to help overcome some of the hurdles faced when someone first attempts to use the AWS. Due to space limitations we have chosen to cover only a few of the very large number of combinations of services, operating systems, and programming environments supported by AWS.

Amazon Web Services are grouped in several categories: computing and networking, storage and content delivery, deployment and management, databases, and application services. In Sections 2.3 and 2.4 we mentioned that new services are continually added to AWS; the look and feel of the web pages changes in time. The screen shots reflect the state of the system at the time of the writing of the first edition of the book, second half of 2012. To access AWS one must first create an account at http://aws.amazon.com/. Once the account is created the Amazon Management Console (AMC) allows the user to select one of the service, e.g., EC2 and then start an instance.

Recall that an EC2 instance is a virtual server started in a region and availability zone selected by the user. Instances are grouped into a few classes and each class has a specific amount of resources such as CPU cycles, main memory, secondary storage, communication, and I/O bandwidth available to it. Several operating systems are supported by AWS including: Amazon Linux, Red Hat Enterprize Linux 6.3, SUSE Linux Enterprize Server 11, Ubuntu Server 12.04.1, as well as several version of Microsoft Windows, see Figure A.2.

The next step is to create an AMI (Amazon Machine Image) on one of the platforms supported by AWS and start an instance using the *RunInstance* API. An AMI is a unit of deployment, an environment including all information necessary to set up and boot an instance. If an application needs more than 20 instances then a special form must be filled out. The local instance persists in storage only for the duration of an instance. The data persist when an instance is started using the Amazon EBS (Elastic Block Storage) and then the instance can be restarted at a later time.

Once an instance is created the user can perform several actions; for example, connect to the instance, launch more instances identical to the current one, or create an EBS AMI. The user can also terminate, reboot, or stop the instance, see Figure A.3. The *Network & Security* panel allows the creation of *Security Groups, Elastic IP addresses, Placement Groups, Load Balancers* and *Key Pairs* (see the discussion in Section A.3), while the EBS panel allows the specification of volumes and the creation of snapshots.



Choose an Amazon Machine Image (AMI) from one of the tabbed lists below by clicking its **Select** button.

Amazon Linux AMI 2012.03
The Amazon Linux AMI 2012.03 is an EBS-backed, PV-GRUB image. It includes Linux 3.2, AWS tools, and repository access to multiple versions of MySQL, PostgreSQL, Python, Ruby, and Tomcat.
Root Device Size: 8 GB [✓ **64 bit**] **32 bit**

Red Hat Enterprise Linux 6.3
Red Hat Enterprise Linux version 6.3, EBS-boot.
Root Device Size: 7 GB [✓ **64 bit**] **32 bit**

SUSE Linux Enterprise Server 11
SUSE Linux Enterprise Server 11 Service Pack 2 basic install, EBS boot with Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.0, PHP 5.3, and Ruby 1.8.7
Root Device Size: 10 GB [✓ **64 bit**] **32 bit**

Ubuntu Server 12.04.1 LTS
Ubuntu Server 12.04.1 LTS, with support available from Canonical (http://www.ubuntu.com/cloud/services).
Root Device Size: 8 GB [✓ **64 bit**] **32 bit**

Ubuntu Server 11.10
Ubuntu Server 11.10 with support available from Canonical

Free tier eligible if used with a micro instance. See AWS free tier for complete details and terms.

FIGURE A.2

The Instance menu allows the user to select from existing AMIs.

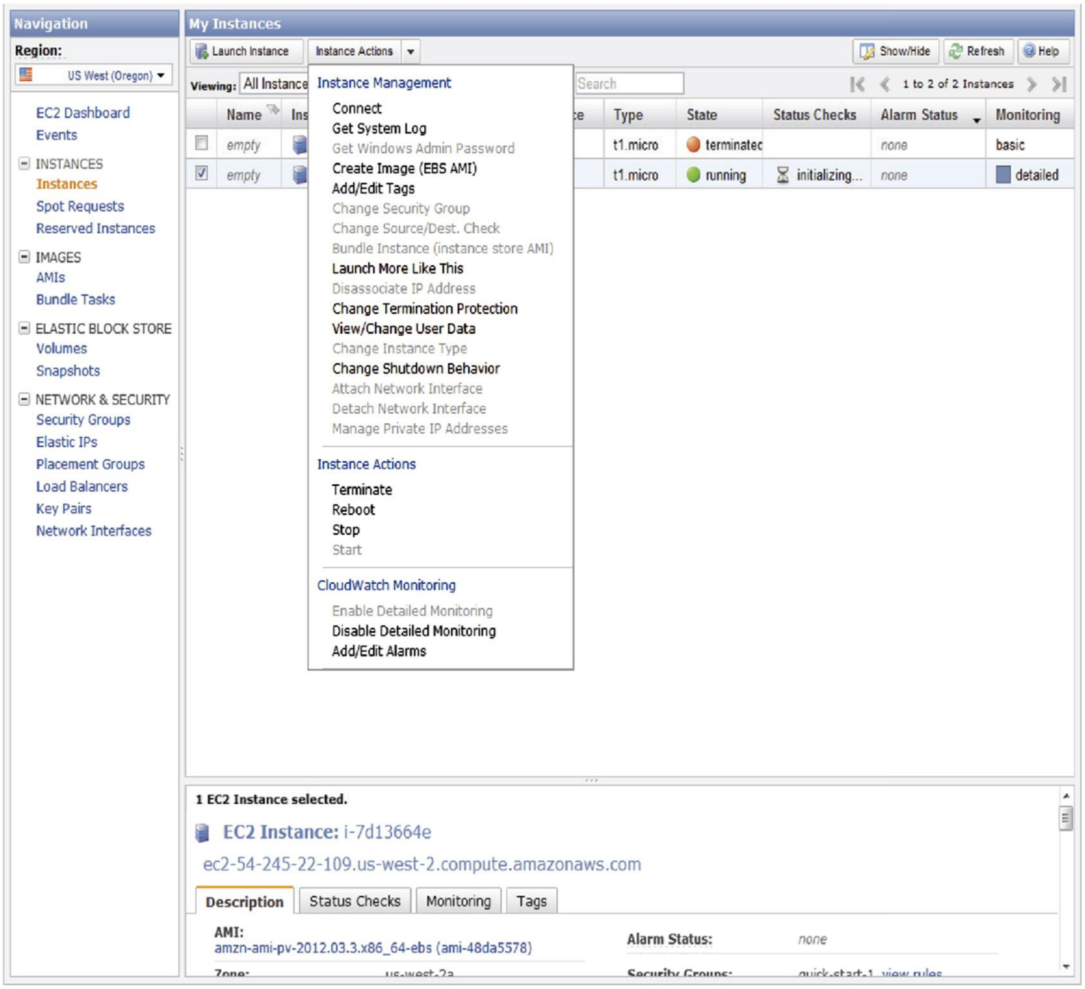
A.2 CONNECTING CLIENTS TO CLOUD INSTANCES THROUGH FIREWALLS

A firewall is a software system based on a set of rules for filtering network traffic; its function is to protect a computer in a local area network from unauthorized access. The first generation of firewalls, deployed in the late 1980s, carried out *packet filtering*; they discarded individual packets which did not match a set of acceptances rules. Such firewalls operated below the transport layer, and discarded packets based on the information in the headers of physical, data link, and transport layer protocols.

The second generation of firewalls operate at the transport layer and maintain the state of all connections passing through them. Unfortunately, this traffic filtering solution opened the possibility of *denial of service attacks*; a denial of service (DOS) attack targets a widely used network service and forces the operating system of the host to fill the connection tables with illegitimate entries. DOS attacks prevent legitimate access to the service.

The third generation of firewalls "understand" widely-used application layer protocols such as FTP, HTTP, TELNET, SSH, and DNS. These firewalls examine the header of application layer protocols and support *intrusion detection systems*.

Firewalls screen incoming traffic and, sometimes, filter outgoing traffic as well. A first filter encountered by the incoming traffic in a typical network is a firewall provided by the operating system of the



	Name	Ins	Type	State	Status Checks	Alarm Status	Monitoring
☐	empty		t1.micro	terminatec		none	basic
☑	empty		t1.micro	running	initializing...	none	detailed

1 EC2 Instance selected.

EC2 Instance: i-7d13664e

ec2-54-245-22-109.us-west-2.compute.amazonaws.com

AMI: amzn-ami-pv-2012.03.3.x86_64-ebs (ami-48da5578)

Alarm Status: none

FIGURE A.3

The *Instance Action* pull-down menu of the *Instances* panel of the *AWS Management Console* allows the user to interact with an instance, e.g., *Connect, Create an EBS AMI Image*, and so on.

router; the second filter is a firewall provided by the operating system running on the local computer, see Figure A.4.

Typically, the local area network (LAN) of an organization is connected to the Internet via a router; a router firewall often hides the true address of hosts in the local network using the network address translation (NAT) mechanism. The hosts behind a firewall are assigned addresses in a "private address range" and the router uses the NAT tables to filter the incoming traffic and translate external IP addresses to private ones. The mapping between the pair *(external address, external port)* and the

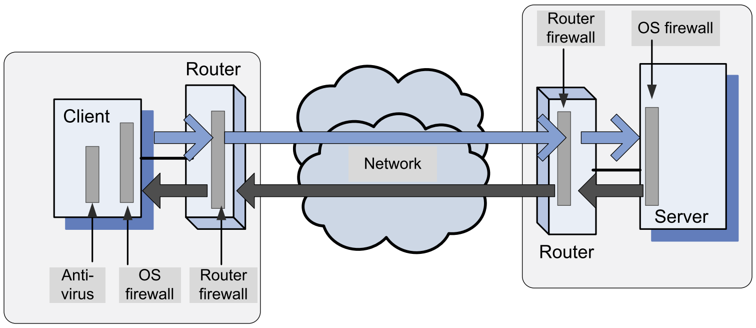


FIGURE A.4

Firewalls screen incoming and sometimes outgoing traffic. The first obstacle encountered by the inbound or outbound traffic is a router firewall, the next one is the firewall provided by the host operating system; sometimes, the antivirus software provides a third line of defense.

Table A.1 Firewall rule setting. The columns indicate if a feature is supported or not by an operating system: the second column – a single rule can be issued to accept/reject a default policy; the third and fourth columns – filtering based on IP destination and source address, respectively; the fifth and sixth columns – filtering based on TCP/UDP destination and source ports, respectively; the seventh and eights columns – filtering based on Ethernet MAC destination and source address, respectively; the ninth and tenth columns – inbound (ingress) and outbound (egress) firewalls, respectively.

Operating system	Def rule	IP dest addr	IP src addr	TCP/ UDP dest port	TCP/ UDP src port	Ether MAC dest	Ether MAC src	In-bound fwall	Out-bound fwall
Linux iptables	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OpenBSD	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Windows XP	No	No	Yes	Partial	No	No	No	Yes	No
Cisco Acces List	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Juniper Networks	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

(internal address, internal port) tuple carried by the network address translation function of the router firewall is called a *pinhole*.

If one tests a client-server application with the client and the server in the same local area network, the packets do not cross a router; once a client from a different LAN attempts to use the service, the packets may be discarded by the firewall of the router. The application may no longer work if the router is not properly configured.

The firewall support in several operating systems is discussed next. Table A.1 summarizes the options supported by different operating systems running on a host or on a router.

A *rule* specifies a filtering option at: (i) the network layer, when filtering is based on the destination/source IP address; (ii) the transport layer, when filtering is based on destination/source port number; (iii) the MAC layer, when filtering is based on the destination/source MAC address.

In Linux or Unix systems the firewall can be configured only by someone with a *root* access using the *sudo* command. The firewall is controlled by a kernel data structure, the *iptables*. The *iptables* command is used to set up, maintain, and inspect the tables of the *IPv4* packet filter rules in the Linux kernel. Several tables may be defined; each table contains a number of built-in chains and may also contain user-defined chains.

A *chain* is a list of rules which can match a set of packets: the *INPUT* rule controls all incoming connections; the *FORWARD* rule controls all packets passing through this host; and the *OUTPUT* rule controls all outgoing connections from the host. A *rule* specifies what to do with a packet that matches: *Accept* – let the packet pass; *Drop* – discharge the packet; *Queue* – pass the packet to the user space; *Return* – stop traversing this chain and resume processing at the head of the next chain. For complete information on the *iptables* see http://linux.die.net/man/8/iptables.

To get the status of the firewall, specify the L (List) action of the *iptables* command

```
 sudo iptables -L
```

As a result of this command the status of the *INPUT, FORWARD,* and *OUTPUT* chains will be displayed.

To change the default behavior for the entire chain, specify the action P (Policy), the chain name, and target name; e.g., to allow all outgoing traffic to pass unfiltered use

```
sudo iptables -P OUTPUT ACCEPT
```

To add a new security rule specify: the action, A (add), the chain, the transport protocol, *tcp* or *udp*, and the target ports as in

```
sudo iptables -A INPUT -p -tcp -dport ssh -j ACCEPT
sudo iptables -A OUTPUT -p -udp -dport 4321 -j ACCEPT
sudo iptables -A FORWARD -p -tcp -dport 80 -j DROP
```

To delete a specific security rule from a chain, set the action D (Delete) and specify the chain name and the rule number for that chain; the top rule in a chain has number 1:

```
sudo iptables -D INPUT 1
sudo iptables -D OUTPUT 1
sudo iptables -D FORWARD 1
```

By default the Linux virtual machines on Amazon's EC2 accept all incoming connections.

The ability to access that virtual machine will be permanently lost when a user accesses an EC2 virtual machine using ssh and then issues the following command

```
sudo iptables -P INPUT DROP.
```

The access to a Windows firewall is provided by a GUI accessed as follows:

```
Control Panel -> System & Security -> Windows Firewall -> Advanced Settings
```

The default behavior for incoming and/or outgoing connections can be displayed and changed from the window *Windows Firewall with Advanced Security on Local Computer*.

The access to the Windows XP firewall is provided by a GUI accessed by selecting *Windows Firewall* in the *Control Panel*. If the status is *ON*, incoming traffic is blocked by default, and a list of Exceptions (as noted on the *Exceptions* tab) define the connections allowed. The user can only define exceptions for: *tcp* on a given port, *udp* on a given port, and a specific program. *Windows XP* does not provide any control over outgoing connections.

Antivirus software running on a local host may provide an additional line of defense. For example, the Avast antivirus software (see www.avast.com) supports several real-time shields. The *Avast network shield* monitors all incoming traffic; it also blocks access to known malicious websites. The *Avast web shield* scans the HTTP traffic and monitors all web browsing activities. The antivirus also provides statistics related to its monitoring activities.

A.3 SECURITY RULES FOR APPLICATION- AND TRANSPORT-LAYER PROTOCOLS IN EC2

A client must know the IP address of a virtual machine in the cloud, to be able to connect to it. Domain Name Service (DNS) is used to map human-friendly names of computer systems to IP addresses in the Internet or in private networks. DNS is a hierarchical distributed database and plays a role reminiscent of an Internet phone book.

In 2010 Amazon announced a DNS service called Route 53 to route users to AWS services and to infrastructure outside of AWS. A network of DNS servers scattered across the globe, which enables customers to gain reliable access to AWS and place strict controls over who can manage their DNS system by allowing integration with AWS Identity and Access Management (IAM).

For several reasons, including security and the ability of the infrastructure to scale up, the IP addresses of instances visible to the outside world are mapped internally to private IP addresses. A virtual machine running under Amazon's EC2 has several IP addresses:

1. *EC2 Private IP Address:* The internal address of an instance; it is only used for routing within the EC2 cloud.
2. *EC2 Public IP Address:* Network traffic originating outside the AWS network must use either the public IP address or the elastic IP address of the instance. The public IP address is translated using the Network Address Translation (NAT) to the private IP address when an instance is launched and it is valid until the instance is terminated. Traffic to the public address is forwarded to the private IP address of the instance.
3. *EC2 Elastic IP Address:* The IP address allocated to an account and used by traffic originated from outside AWS. NAT is used to map an elastic IP address to the private IP address. Elastic IP addresses allow the cloud user to mask instance or availability zone failures by programmatically re-mapping a public IP addresses to any instance associated with the user's account. This allows fast recovery after a system failure; for example, rather than waiting for a cloud maintenance team to reconfigure or replace the failing host, or waiting for DNS to propagate the new public IP to all of the customers of a web service hosted by EC2, the web service provider can re-map the elastic IP address to a replacement instance. Amazon charges a fee for unallocated Elastic IP addresses.

To control access to a user VMs AWS uses *security groups*. A VM instance belongs to one, and only one, security group which can only be defined before the instance is launched. Once an instance is running, the security group the instance belongs to cannot be changed. However, more than one instance can belong to a single security group.

Security group rules control inbound traffic to the instance and have no effect on outbound traffic from the instance. The inbound traffic to an instance, either from outside the cloud or from other instances running on the cloud, is blocked, unless a rule stating otherwise is added to the security group of the instance. For example, assume a client running on instance A in the security group Σ_A is to connect to a server on instance B listening on TCP port P, where B is in security group Σ_B. A new rule must be added to security group Σ_B to allow connections to port P; to accept responses from server B a new rule must be added to security group Σ_A.

The following steps allow the user to add a security rule:

1. Sign in to the AWS Management Console at http://aws.amazon.com using your Email address and password and select EC2 service.
2. Use the *EC2 Request Instance Wizard* to specify the instance type, whether it should be monitored, and specify a key/value pair for the instance to help organize and search, see Figure A.6.
3. Provide a name for the key pair, then on the left hand side panel choose *Security Groups* under *Network & Security*, select the desired security group and click on the *Inbound* tab to enter the desired rule, see Figure A.5.

To allocate an elastic IP address use the *Elastic IPs* tab of the *Network & Security* left hand side panel.

On Linux or Unix systems the port numbers below 1024 can only be assigned by the *root*. The plain ASCII file called *services* maps friendly textual names for Internet services to their assigned port numbers and protocol types as in the following example:

```
netstat 15/tcp
ftp     21/udp
ssh     22/tcp
telnet  23/tcp
http    80/tcp
```

A.4 HOW TO LAUNCH AN EC2 LINUX INSTANCE AND CONNECT TO IT

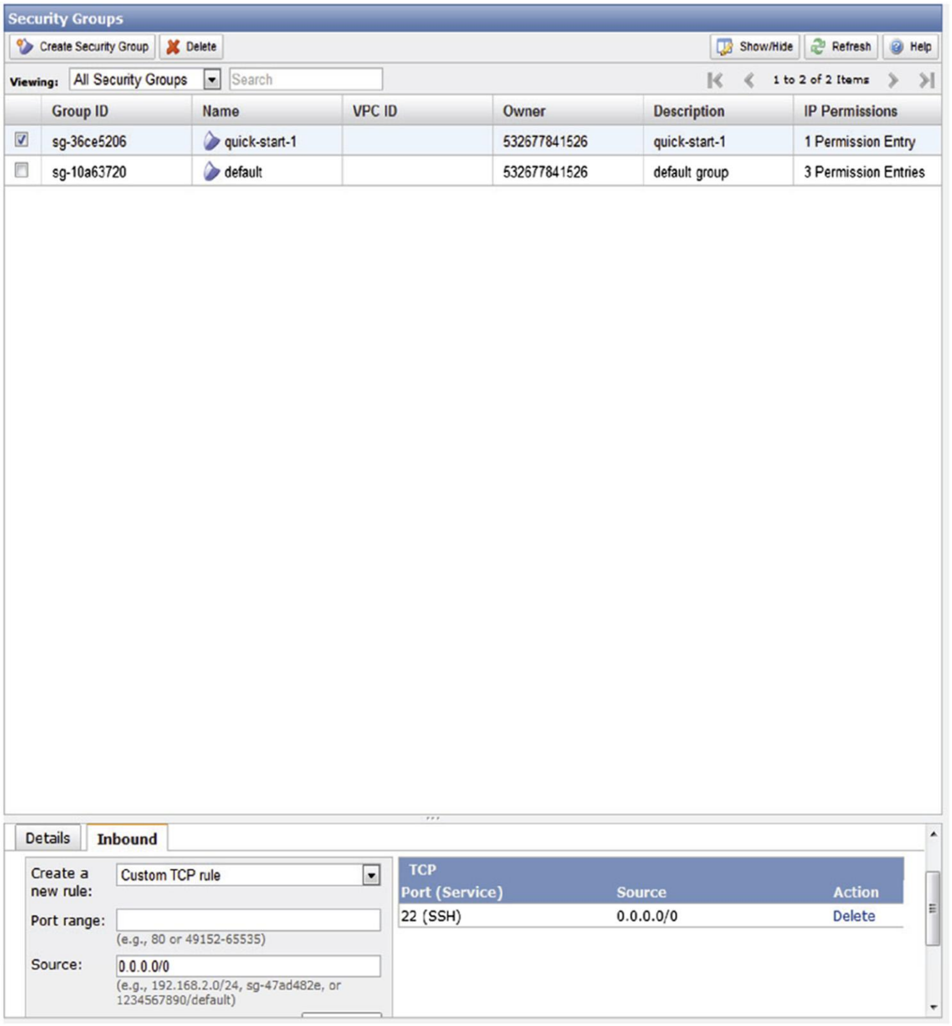
This section gives a step-by-step process to launch an EC2 Linux instance from a Linux platform.

A. Launch an instance

1. From the *AWS management console*, select EC2 and, once signed in, go to *Launch Instance Tab*.
2. To determine the processor architecture when you want to match the instance with the hardware enter the command

   ```
   uname -m
   ```

   and choose an appropriate Amazon Linux AMI by pressing *Select*.
3. Choose *Instance Details* to control the number, size, and other settings for instances.
4. To learn how the system works, press *Continue* to select the default settings.



	Group ID	Name	VPC ID	Owner	Description	IP Permissions
☑	sg-36ce5206	quick-start-1		532677841526	quick-start-1	1 Permission Entry
☐	sg-10a63720	default		532677841526	default group	3 Permission Entries

Create a new rule: Custom TCP rule

Port range:

(e.g., 80 or 49152-65535)

Source: 0.0.0.0/0

(e.g., 192.168.2.0/24, sg-47ad482e, or 1234567890/default)

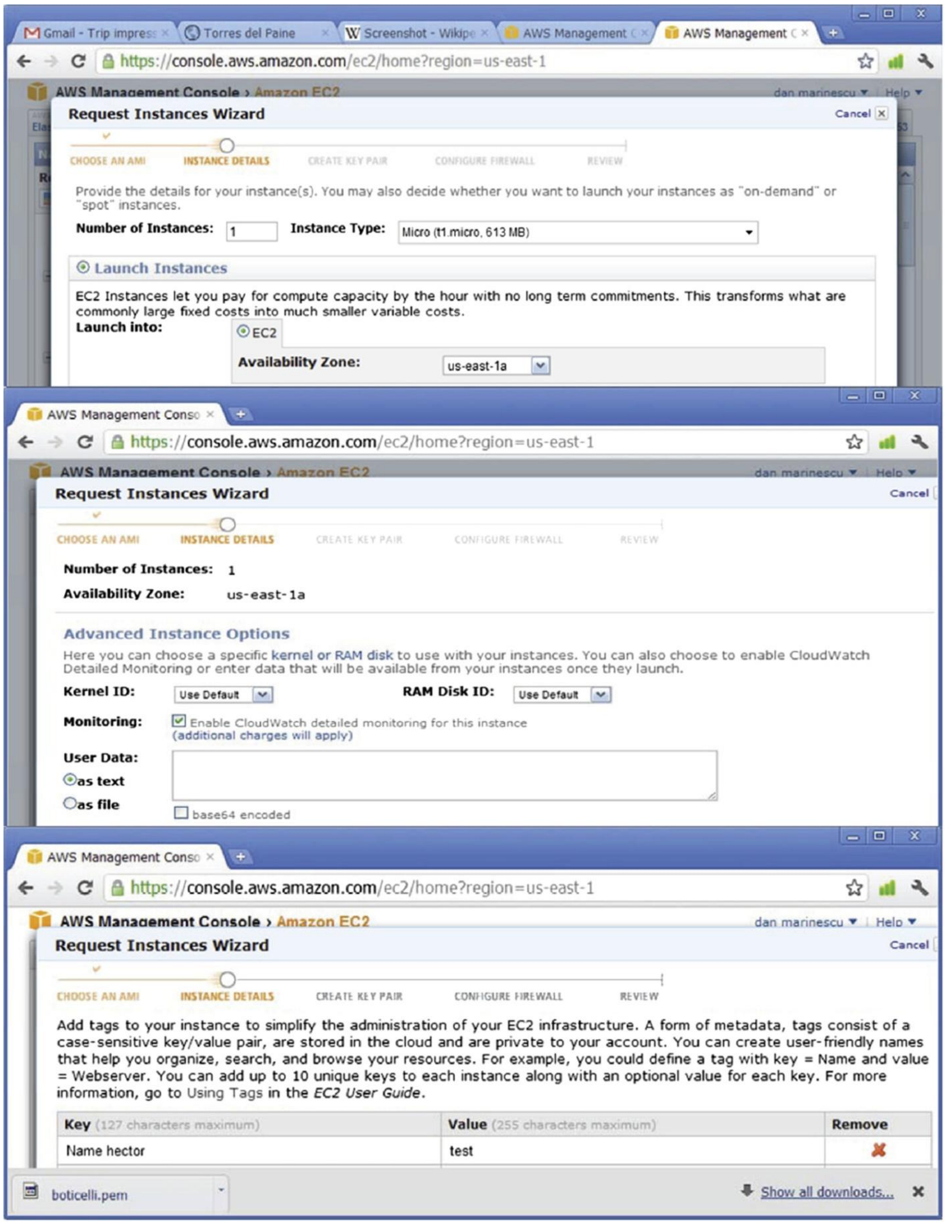
TCP Port (Service)	Source	Action
22 (SSH)	0.0.0.0/0	Delete

FIGURE A.5

AWS security. Choose *Security Groups* under *Network & Security*, select the desired security group and click on the *Inbound* tab to enter the desired rule.

5. Define the instances security, as discussed in Section A.3: in the *Create Key Pair* page enter a name for the pair and then press *Create and Download Key Pair*.
6. The key pair file downloaded in the previous step is a *.pem* file and it must be hidden to prevent unauthorized access; if the file is in the directory *awcdir/dada.pem* enter the commands

```
cd awcdir
chmod 400 dada.pem
```



Request Instances Wizard

Provide the details for your instance(s). You may also decide whether you want to launch your instances as "on-demand" or "spot" instances.

Number of Instances: 1 **Instance Type:** Micro (t1.micro, 613 MB)

Launch Instances

EC2 Instances let you pay for compute capacity by the hour with no long term commitments. This transforms what are commonly large fixed costs into much smaller variable costs.

Launch into: EC2

Availability Zone: us-east-1a

Request Instances Wizard

Number of Instances: 1

Availability Zone: us-east-1a

Advanced Instance Options

Here you can choose a specific kernel or RAM disk to use with your instances. You can also choose to enable CloudWatch Detailed Monitoring or enter data that will be available from your instances once they launch.

Kernel ID: Use Default **RAM Disk ID:** Use Default

Monitoring: ✓ Enable CloudWatch detailed monitoring for this instance (additional charges will apply)

User Data:

● **as text**

○ **as file**

☐ base64 encoded

Request Instances Wizard

Add tags to your instance to simplify the administration of your EC2 infrastructure. A form of metadata, tags consist of a case-sensitive key/value pair, are stored in the cloud and are private to your account. You can create user-friendly names that help you organize, search, and browse your resources. For example, you could define a tag with key = Name and value = Webserver. You can add up to 10 unique keys to each instance along with an optional value for each key. For more information, go to Using Tags in the *EC2 User Guide*.

Key (127 characters maximum)	**Value** (255 characters maximum)	**Remove**
Name hector	test	✗

FIGURE A.6

EC2 Request Instances Wizard is used to: (A) specify the number and type of instances and the zone; (B) specify the kernelId, the RAM diskId, and enable the *CloudWatch* service to monitor the EC2 instance; (C) add tags to the instance; a tag is stored in the cloud and consists of a case-sensitive key/value pair private to the account.

7. Configure the firewall; go to the page *Configure Firewall*, select the option *Create a New Security Group* and provide a *Group Name*. Normally one uses *ssh* to communicate with the instance; the default port for communication is port 8080 and one can change the port and other rules by creating a new rule.
8. Press *Continue* and examine the review page which gives a summary of the instance.
9. Press *Launch* and examine the confirmation page and then press *Close* to end the examination of the confirmation page.
10. Press the *Instances* tab on the navigation pane to view the instance.
11. Look for your *Public DNS* name. As by default some details of the instance are hidden, click on the *Show/Hide* tab on the top of the console and select *Public DNS*.
12. Record the *Public DNS* as *PublicDNSname*; it is needed to connect to the instance from the Linux terminal.
13. Use the *ElasticIP* panel to assign an elastic IP address if a permanent IP address is required.

B. Connect to the instance using ssh and the tcp transport protocol.

1. Add a rule to the *iptables* to allow ssh traffic using the tcp protocol. Without this step either an*access denied* or a *permission denied* error message appears when trying to connect to the instance.

```
sudo iptables -A iptables  -p  -tcp  -dport  ssh  -j  ACCEPT
```

2. Enter the Linux command

```
ssh -i abc.pem ec2-user@PublicDNSname
```

If you get the prompt *You want to continue connecting?* respond *Yes*; a warning that the DNS name was added to the list of known hosts will appear.

3. An icon of the *Amazon Linux AMI* will be displayed.

C. Gain root access to the instance

By default the user does not have *root* access to the instance thus, cannot install any software. Once connected to the EC2 instance use the following command to gain *root* privileges

```
sudo -i
```

Then use *yum* install commands to install software, e.g., *gcc* to compile C programs on the cloud.

D. Run the service *ServiceName*

If the instance runs under *Linux* or *Unix* the service is terminated when the *ssh* connection is closed; to avoid the early termination use the command

```
nohup ServiceName
```

To run the service in the background and redirect *stdout* and *stderr* to files *p.out* and *p.err*, respectively, execute the command

```
nohup ServiceName > p.out 2 > p.err &
```

A.5 HOW TO USE S3 IN JAVA

The Java API for Amazon Web Services is provided by the AWS SDK. A software development kit (SDK) is a set of software tools for the creation of applications in a specific software environment. Java Development Kit (JDK) is an SDK for Java developers available from Oracle.

JDK includes a set of programming tools such as: *javac*, the Java compiler which converts Java source code into Java bytecode; *java*, the loader for Java applications, it can interpret the class files generated by the Java compiler; *javadoc* the documentation generator; *jar*, the archiver for class libraries; *jdb*, the debugger; *JConsole*, the monitoring and management console; *jstat*, JVM statistics monitoring; *jps*, JVM process status tool; *jinfo*, the utility to get configuration information from a running Java process; *jrunscript*, the command-line script shell for Java; *appletviewer* tool to debug Java applets without a web browser; and *idlj*, the IDL-to-Java compiler. The *Java Runtime Environment* is also a component of the JDK consisting of a Java Virtual Machine (JVM) and libraries.

Create an S3 client. S3 access is handled by the class *AmazonS3Client* instantiated with the account credentials of the AWS user

```
AmazonS3Client s3 = new AmazonS3Client(
new BasicAWSCredentials("your_access_key", "your_secret_key"));
```

The access and the secret keys can be found on the user's AWS account home page as mentioned in Section A.3.

Buckets. An *S3 bucket* is analogous to a file folder or directory and it is used to store *S3 Objects*. Bucket names must be *globally unique* hence, it is advisable to check first if the name exists

```
s3.doesBucketExist("bucket_name");
```

This function returns "true" if the name exists and "false" otherwise. Buckets can be created and deleted either directly from the AWS Management Console or programmatically as follows:

```
s3.createBucket("bucket_name");
s3.deleteBucket("bucket_name");
```

S3 objects. An *S3 object* stores the actual data and it is indexed by a key string. A single key points to only one S3 object in one bucket. Key names do not have to be globally unique, but if an existing key is assigned to a new object, then the original object indexed by the key is lost. To upload an object in a bucket one can use the *AWS Management Console,* or programmatically a file $local\_file\_name$ can be uploaded from the local machine to the bucket $bucket\_name$ under the key key using

```
File f = new File("local_file_name");
s3.putObject("bucket_name", "key", f);
```

A versioning feature for the objects in S3 was made available recently; it allows to preserve, retrieve, and restore every version of an S3 object. To avoid problems when uploading large files, e.g., the drop of the connection, use the *.initiateMultipartUpload()* with an API described at the *AmazonS3Client*. To access this object with key key from the bucket $bucket\_name$ use:

```
S3Object myFile = s3.getObject("bucket_name", "key");
```

To read this file, you must use the S3Object's *InputStream*:

```
InputStream in = myFile.getObjectContent();
```

The *InputStream* can be accessed using *Scanner, BufferedReader* or any other method supported. Amazon recommends closing the stream as early as possible, as the content is not buffered and it is streamed directly from the S3; an open *InputStream* means an open connection to S3. For example, the following code will read an entire object and print the contents to the screen:

```
AmazonS3Client s3 = new AmazonS3Client(
    new BasicAWSCredentials("access_key", "secret_key"));
    InputStream input = s3.getObject("bucket_name", "key")
       .getObjectContent();
    Scanner in = new Scanner(input);
    while (in.hasNextLine())
        {
         System.out.println(in.nextLine());
        }
in.close();
input.close();
```

Batch Upload/Download. Batch upload requires repeated calls of *s3.putObject()* while iterating over local files.

To view the keys of all objects in a specific bucket use

```
 ObjectListing listing = s3.listObjects("bucket_name");
```

Object Listing supports several useful methods including *getObjectSummaries(). S3ObjectSummary* encapsulates most of an S3 object properties (excluding the actual data), including the key to access the object directly,

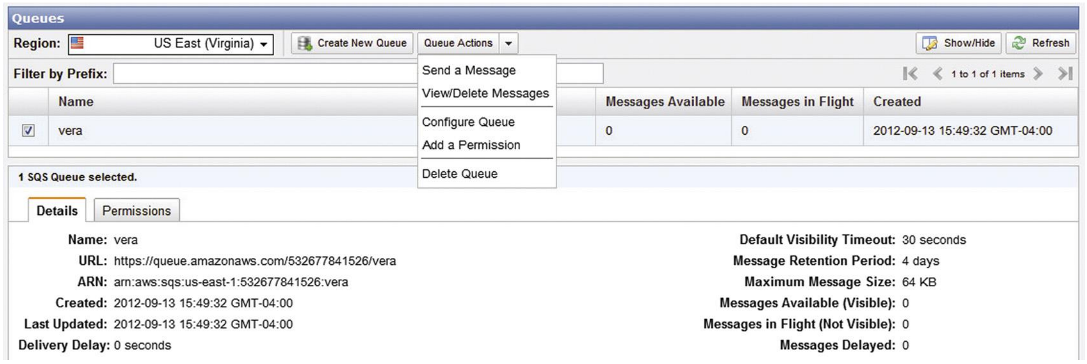
```
List<S3ObjectSummary> summaries = listing.getObjectSummaries();
```

For example, the following code will create a list of all keys used in a particular bucket and all of the keys will be available in string form in *List < String > allKeys*:

```
AmazonS3Client s3 = new AmazonS3Client(
    new BasicAWSCredentials("access_key", "secret_key"));
    List<String> allKeys = new ArrayList<String>();
    ObjectListing listing = s3.listObjects("bucket_name");
    for (S3ObjectSummary summary:listing.getObjectSummaries())
      {
        allKeys.add(summary.getKey());
      }
```

Note that if the bucket contains a very large number of objects then *s3.listObjects()* will return a truncated list. Use the following command to test if the list is truncated one could use *listing.isTruncated()*; to get the next batch of objects

```
 s3.listNextBatchOfObjects(listing)};
```



	Name	Messages Available	Messages in Flight	Created
☑	vera	0	0	2012-09-13 15:49:32 GMT-04:00

1 SQS Queue selected.

Name:	vera	Default Visibility Timeout:	30 seconds
URL:	https://queue.amazonaws.com/532677841526/vera	Message Retention Period:	4 days
ARN:	arn:aws:sqs:us-east-1:532677841526:vera	Maximum Message Size:	64 KB
Created:	2012-09-13 15:49:32 GMT-04:00	Messages Available (Visible):	0
Last Updated:	2012-09-13 15:49:32 GMT-04:00	Messages in Flight (Not Visible):	0
Delivery Delay:	0 seconds	Messages Delayed:	0

FIGURE A.7

Queue actions in SQS.

To account for a large number of objects in the bucket, the previous example becomes

```
AmazonS3Client s3 = new AmazonS3Client(
new BasicAWSCredentials("access_key", "secret_key"));
List<String> allKeys = new ArrayList<String>();
ObjectListing listing = s3.listObjects("bucket_name");
while (true)
  {
   for (S3ObjectSummary summary :
       listing.getObjectSummaries())
       {
        allKeys.add(summary.getKey());
       }
    if (!listing.isTruncated())
       {
        break;
       }
   listing = s3.listNextBatchOfObjects(listing);
  }
```

A.6 HOW TO MANAGE AWS SQS SERVICES IN C#

Recall from Section 2.3 that SQS is a system for supporting automated workflows. Multiple components can communicate with messages sent and received via SQS. An example showing the use of message queues is presented in Section 7.6. Figure A.7 shows the actions available for a given queue in *SQS*.

The following steps can be used to create a queue, send a message, receive a message, delete a message, delete the queue in C#:

1. Authenticate an SQS connection

```
NameValueCollection appConfig =
        ConfigurationManager.AppSettings;
    AmazonSQS sqs = AWSClientFactory.CreateAmazonSQSClient
        (appConfig["AWSAccessKey"], appConfig["AWSSecretKey"]);
```

2. Create a queue

```
CreateQueueRequest sqsRequest = new CreateQueueRequest();
sqsRequest.QueueName = "MyQueue";
CreateQueueResponse createQueueResponse =
    sqs.CreateQueue(sqsRequest);
String myQueueUrl;
myQueueUrl = createQueueResponse.CreateQueueResult.QueueUrl;
```

3. Send a message

```
SendMessageRequest sendMessageRequest =
    new SendMessageRequest();
sendMessageRequest.QueueUrl =
    myQueueUrl; //URL from initial queue
sendMessageRequest.MessageBody = "This is my message text.";
sqs.SendMessage(sendMessageRequest);
```

4. Receive a message

```
ReceiveMessageRequest receiveMessageRequest =
    new ReceiveMessageRequest();
receiveMessageRequest.QueueUrl = myQueueUrl;
ReceiveMessageResponse receiveMessageResponse =
    sqs.ReceiveMessage(receiveMessageRequest);
```

5. Delete a message

```
DeleteMessageRequest deleteRequest =
    new DeleteMessageRequest();
deleteRequest.QueueUrl = myQueueUrl;
deleteRequest.ReceiptHandle = messageRecieptHandle;
DeleteMessageResponse DelMsgResponse =
    sqs.DeleteMessage(deleteRequest);
```

6. Delete a queue

```
DeleteQueueRequest sqsDelRequest = new DeleteQueueRequest();
sqsDelRequest.QueueUrl =
    createQueueResponse.CreateQueueResult.QueueUrl;
DeleteQueueResponse delQueueResponse =
sqs.DeleteQueue(sqsDelRequest);
```

A.7 HOW TO INSTALL SNS ON UBUNTU 10.04

SNS, the Simple Notification Service, is a web service for: monitoring applications, workflow systems, time-sensitive information updates, mobile applications, and other event-driven applications which require a simple and efficient mechanism for message delivery. SNS "pushes" messages to clients, rather than requiring a user to periodically poll a mailbox or another site for messages.

SNS is based on the publish-subscribe paradigm; it allows a user to define the topics, the transport protocol used (HTTP/HTTPS, Email, SMS, SQS), and the end-point (URL, Email address, phone number, SQS queue) for notifications to be delivered.

Ubuntu is an open source operating system for personal computers based on Debian Linux distribution. The desktop version of Ubuntu[1] supports Intel x86 32-bit and 64-bit architectures.

SNS supports the following actions:

- Add/Remove Permission
- Confirm Subscription
- Create/Delete Topic
- Get/Set Topic Attributes
- List Subscriptions/Topics/Subscriptions By Topic
- Publish/Subscribe/Unsubscribe

The site http://awsdocs.s3.amazonaws.com/SNS/latest/sns-qrc.pdf provides detailed information about each one of these actions.

The following steps must be taken to install an SNS client:

1. Install Java in the *root* directory and then execute the commands

```
deb http://archive.canonical.com/lucid partner
update
install sun-java6-jdk
```

   Then change the default Java settings

```
update-alternatives -config java
```

2. Download the *SNS* client, unzip the file and change permissions

```
wget http://sns-public-resources.s3.amazonaws.com/
       SimpleNotificationServiceCli-2010-03-31.zip
chmod 775 /root/ SimpleNotificationServiceCli-1.0.2.3/bin
```

3. Start the AWS management console and go to *Security Credentials*. Check the *Access Key ID* and the *Secret Access Key* and create a text file */root/credential.txt* with the following content:

```
AWSAccessKeyId= your_Access_Key_ID
AWSSecretKey= your_Secret_Access_Key
```

4. Edit the *.bashrc* file and add

```
export AWS_SNS_HOME=~/SimpleNotificationServiceCli-1.0.2.3/
export AWS_CREDENTIAL_FILE=$HOME/credential.txt
export PATH=$AWS_SNS_HOME/bin
export JAVA_HOME=/usr/lib/jvm/java-6-sun/
```

[1]Ubuntu is an African humanist philosophy; "ubuntu" is a word in the Bantu language of South Africa meaning "humanity towards others."

5. Reboot the system
6. Enter on the command line

   ```
   sns.cmd
   ```

   If the installation was successful the list of *SNS* commands will be displayed.

A.8 HOW TO CREATE AN EC2 PLACEMENT GROUP AND USE MPI

An *EC2 Placement Group*, is a logical grouping of instances which allows the creation of a virtual cluster. When several instances are launched as an *EC2 Placement Group* the virtual cluster has a high bandwidth interconnect system suitable for network-bound applications. The cluster computing instances require an HVM (Hardware Virtual Machine) ECB-based machine image, while other instances use a PVM (Paravirtual Machine) image. Such clusters are particularly useful for high performance computing when most applications are communication intensive.

Once a placement group is created, MPI can be used for communication among the instances in the placement group. MPI is a de-facto standard for parallel applications using message passing, designed to ensure high performance, scalability, and portability; it is a language-independent "message-passing application programmer interface, together with a protocol and the semantic specifications for how its features must behave in any implementation" [206]. MPI supports point-to-point, as well as collective communication; it is widely used by parallel programs based on the SPMD (Same Program Multiple Data) paradigm.

The following *C* code [206] illustrates the startup of MPI communication for a process group, *MPI_COM_PROCESS_GROUP* consisting of *nprocesses*; each process is identified by its *rank*. The runtime environment *mpirun* or *mpiexec* spawns multiple copies of the program, with the total number of copies determining the number of process ranks in *MPI_COM_PROCESS_GROUP*.

```
#include <mpi.h>
#include <stdio.h>
#include <string.h>
#define TAG 0
#define BUFSIZE 128

int main(int argc, char *argv[])
{
  char idstr[32];
  char buff[BUFSIZE];
  int nprocesses;
  int my_processId;
  int i;
  MPI_Status stat;
  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COM_PROCESS_GROUP,&nprocesses);
  MPI_Comm_rank(MPI_COM_PROCESS_GROUP,&my_processId);
```

MPI_SEND and *MPI_RECEIVE* are blocking send and blocking receive, respectively; their syntax is:

```
int MPI_Send(void *buf, int count, MPI_Datatype datatype,
             int dest, int tag,MPI_Comm comm)
int MPI_Recv(void *buf, int count, MPI_Datatype datatype,
           int source, int tag, MPI_Comm comm, MPI_Status *status)
```

with

buf	–	initial address of send buffer (choice)
count	–	number of elements in send buffer (nonnegative integer)
datatype	–	data type of each send buffer element (handle)
dest	–	rank of destination (integer)
tag	–	message tag (integer)
comm	–	communicator (handle).

Once started, every process other than the coordinator, the process with $rank = 0$, sends a message to the entire group and then receives a message from each of the other members of the process group.

```
 if(my_processId == 0)
 {
   printf("%d: We have %d processes\n", my_processId, nprocesses);
   for(i=1;i<nprocesses;i++)
   {
     sprintf(buff, "Hello %d! ", i);
     MPI_Send(buff, BUFSIZE, MPI_CHAR, i, TAG, MPI_COMM_PROCESS_GROUP);
   }
   for(i=1;i<nprocesses;i++)
   {
     MPI_Recv(buff, BUFSIZE, MPI_CHAR, i, TAG, MPI_COMM_PROCESS_GROUP, &stat);
     printf("%d: %s\n", my_processId, buff);
   }
 }
 else
 {
   /* receive from rank 0: */
   MPI_Recv(buff, BUFSIZE, MPI_CHAR, 0, TAG, MPI_COMM_PROCESS_GROUP, &stat);
   sprintf(idstr, "Processor %d ", my_processId);
   strncat(buff, idstr, BUFSIZE-1);
   strncat(buff, "reporting for duty\n", BUFSIZE-1);
   /* send to rank 0: */
   MPI_Send(buff, BUFSIZE, MPI_CHAR, 0, TAG, MPI_COM_PROCESS_GROUP);
 }

 MPI_Finalize();
 return 0;
}
```

An example of cloud computing using the MPI is described in [167]. An example of MPI use on EC2 is at http://rc.fas.harvard.edu/faq/amazonec2.

A.9 STARCLUSTER – A CLUSTER COMPUTING TOOLKIT FOR EC2

StarCluster, http://star.mit.edu/cluster/, is an open source cluster-computing toolkit for EC2. The system assigns user-friendly names to the nodes of the virtual cluster and the cluster is so configured to allow *ssh* from any node of the cluster to any other nodes. It allows to attach EBS volumes to the cluster for persistent storage and provides and API for executing OS commands such as copying files. StarCluster supports dynamically cluster reconfiguration, as well as lunching spot instances to reduce the service costs.

StarCluster AMIs consist of several scientific libraries:

1. OpenMPI – for writing parallel applications.
2. Automatically Tuned Linear Algebra Software (ATLAS) – optimized for Amazon EC2 larger instances, see http://math-atlas.sourceforge.net/.
3. NumPy/SciPy compiled against the optimized ATLAS install. SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering. NumPy is a set of tools for integrating C/C++ and Fortran code useful linear algebra, Fourier transform, and random number capabilities, see https://www.scipy.org/scipylib/.
4. IPython – interactive parallel computing in Python, see https://ipython.org/.

Several other important features of the StarCluster are: (1) Support for starting/stopping EBS-backed clusters on EC2. (2) Elastic Load Balancing – using Sun Grid Engine queue statistics. (3) Support for specifying instance types on a per-node basis. (4) A number of commands for EC2 and S3 operations including the ones in Table A.2.

A.10 AN ALTERNATIVE SETTING OF AN MPI VIRTUAL CLUSTER

An alternative setting up for an MPI cluster is described in [312]. The instances used are *cc2.8xlarge* with 2x Intel Xeon E5-2670 processors and $\approx$ 60 GB of RAM per node and spot instances rather than reserved ones were used thus, saving about 90% of the cost. The VM virtualization layer used by the *cc2.8xlarge* instances is thinner than the one of other instances.

Setting user's instances involves several steps:

1. Select a VM image and an instance type.
2. Define a placement group.
3. Configure the storage.
4. Define the VM tags to manage the instances.
5. Create of a key pair.
6. Setup the security group.
7. Launch the instances.

The nest phase is the configuration of the virtual cluster (VC). All instances of the virtual cluster are booted with the same operating system and share a 10 Gbps Ethernet subnet. The public IP address of the booted instances is available from the EC2 Dashboard by selecting their entry under the "Instances" page. The preliminary steps for the VC configurations are:

- Create nodes aliases – add their internal IP addresses to */etc/hosts* as *node1, node2, node3, node4*.
- Create aliases for files to be transferred from the master (*node1*) to workers *node2, node3, node4*.
- Enable password-less ssh between instances.

Table A.2 StartCluster commands for EC2 and S3 operations.

Command	Function
listinstances	List all running EC2 instances
listspots	List all EC2 spot instance requests
listimages	List all registered EC2 images (AMIs)
listpublic	List all public StarCluster images on EC2
listkeypairs	List all EC2 keypairs
createkey	Create a new Amazon EC2 keypair
removekey	Remove a keypair from Amazon EC2
s3image	Create a new instance-store (S3) AMI from a running EC2 instance
ebsimage	Create a new EBS image (AMI) from a running EC2 instance
removeimage	Deregister an EC2 image (AMI)
createvolume	Create a new EBS volume for use with StarCluster
listvolumes	List all EBS volumes
resizevolume	Resize an existing EBS volume
removevolume	Delete one or more EBS volumes
spothistory	Show spot instance pricing history stats
showconsole	Show console output for an EC2 instance
listregions	List all EC2 regions
listzones	List all EC2 availability zones in the current region
listbuckets	List all S3 buckets
showbucket	Show all files in an S3 bucket

There are two options to install and lunch MPI in every node, the first for *OpenMPI* and the second for *mpich*, both use the *yum* package.

```
sudo yum install openmpi-devel
sudo yum install mpich-devel
```

Once the GCC compiler, the MPI runtime, and the mpicc wrapper are available in every node the following commands must be added to the *.bashrc* file on all compute nodes

```
export PATH=/usr/lib64/openmpi/bin:$PATH
export LD_LIBRARY_PATH=/usr/lib64/openmpi/lib
```

The following command runs a program “application.xx” available on every node

```
mpirun -np 32 -hostfile ~/nodefile ~/application.xx
```

with the hostfile called “nodefile” compatible with OpenMPI created as follows:

```
$ cat ~/nodefile
node1 slots=16
node2 slots=16
node3 slots=16
node4 slots=16
```

A 64-way MPI job is created. Each node has 16 cores and 16 hyperthreads.

A.11 HOW TO INSTALL HADOOP ON ECLIPSE ON A WINDOWS SYSTEM

Eclipse (http://www.eclipse.org/) is a software development environment. Eclipse consists of an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java and can be used to develop applications in Java and, by means of various plug-ins, in C, C++, Perl, PHP, Python, R, Ruby, and several other languages. The IDE is often called Eclipse CDT for C/C++, Eclipse JDT for Java, and Eclipse PDT for PHP.

The software packages used are:

- Apache Hadoop is a software framework that supports data-intensive distributed applications under a free license. *Hadoop* was inspired by Google's MapReduce; see Section 7.5 for a discussion of MapReduce and Section 7.6 for an application using *Hadoop*.
- *Cygwin* is a Unix-like environment for Microsoft Windows. It is open source software, released under the GNU General Public License version 2. *Cygwin* consists of: (1) a dynamic-link library (DLL) as an API compatibility layer providing a substantial part of the POSIX API functionality; and (2) an extensive collection of software tools and applications that provide a Unix-like look and feel.

A. Pre-requisites

- *Java 1.6*; set JAVA_Home = path where *JDK* is installed.
- *Eclipse Europa 3.3.2*
  Note: the *hadoop* plugin was specially designed for Europa and newer releases of *Eclipse* might have some issues with *hadoop* plugin.

B. SSH Installation

1. Install *cygwin* using the installer downloaded from http://www.cygwin.com. From the *Select Packages* window select the *openssh* and *openssl* under Net.
   Note: Create a desktop icon when asked during installation.
2. Display the "Environment Variables" panel

```
Computer -> System Properties -> Advanced System Settings
                                        -> Environment Variables
```

   Click on the variable named *Path* and press *Edit*; append the following value to the path variable

```
    ;c:\cygwin\bin;c:\cygwin\usr\bin
```

3. Configure the *ssh daemon* using *cygwin*. Left click on the *cygwin* icon on desktop and click "Run as Administrator". Type in the command window of *cygwin*

```
   ssh-host-config.
```

4. Answer "Yes" when prompted *sshd should be installed as a service*; answer "No" to all other questions.

5. Start the *cygwin* service by navigating to

   ```
   Control Panel -> Administrative Tools -> Services
   ```

   Look for *cygwin sshd* and start the service.
6. Open *cygwin* command prompt and execute the following command to generate keys

   ```
   ssh-keygen
   ```

7. When prompted for filenames and pass phrases press ENTER to accept default values. After the command has finished generating keys, enter the following command to change into your *.ssh* directory:

   ```
   cd ~/.ssh
   ```

8. Check if the keys were indeed generated

   ```
   ls -l
   ```

9. The two files *id_rsa.pub* and *id_rsa* with recent creation dates contain authorization keys.
10. To register the new authorization keys, enter the following command (Note: the sharply-angled double brackets are very important)

    ```
    cat id_rsa.pub >> authorized_keys
    ```

11. Check if the keys were set up correctly

    ```
    ssh localhost
    ```

12. Since it is a new *ssh* installation, you will be warned that authenticity of the host could not be established and will be asked whether you really want to connect. Answer YES and press ENTER. You should see the *cygwin* prompt again, which means that you have successfully connected.
13. Now execute again the command:

    ```
    ssh localhost
    ```

    this time no prompt should appear.

C. Download *hadoop*

1. Download *hadoop 0.20.1* and place in a directory such as

   ```
   C:\Java
   ```

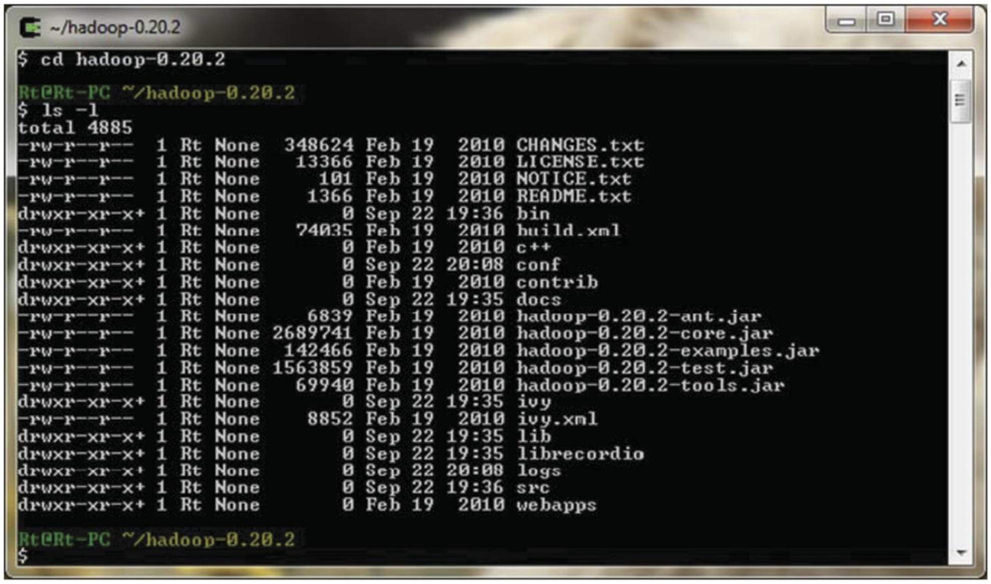
2. Open the *cygwin* command prompt and execute

   ```
   cd
   ```

3. Enable the home directory folder to be shown in the Windows Explorer window

   ```
   explorer
   ```

4. Open another Windows Explorer window and navigate to the folder that contains the downloaded *hadoop* archive.
5. Copy the *hadoop* archive into the home directory folder.



```
$ cd hadoop-0.20.2

Rt@Rt-PC ~/hadoop-0.20.2
$ ls -l
total 4885
-rw-r--r--  1 Rt None  348624 Feb 19  2010 CHANGES.txt
-rw-r--r--  1 Rt None   13366 Feb 19  2010 LICENSE.txt
-rw-r--r--  1 Rt None     101 Feb 19  2010 NOTICE.txt
-rw-r--r--  1 Rt None    1366 Feb 19  2010 README.txt
drwxr-xr-x+ 1 Rt None       0 Sep 22 19:36 bin
-rw-r--r--  1 Rt None   74035 Feb 19  2010 build.xml
drwxr-xr-x+ 1 Rt None       0 Feb 19  2010 c++
drwxr-xr-x+ 1 Rt None       0 Sep 22 20:08 conf
drwxr-xr-x+ 1 Rt None       0 Feb 19  2010 contrib
drwxr-xr-x+ 1 Rt None       0 Sep 22 19:35 docs
-rw-r--r--  1 Rt None    6839 Feb 19  2010 hadoop-0.20.2-ant.jar
-rw-r--r--  1 Rt None 2689741 Feb 19  2010 hadoop-0.20.2-core.jar
-rw-r--r--  1 Rt None  142466 Feb 19  2010 hadoop-0.20.2-examples.jar
-rw-r--r--  1 Rt None 1563859 Feb 19  2010 hadoop-0.20.2-test.jar
-rw-r--r--  1 Rt None   69940 Feb 19  2010 hadoop-0.20.2-tools.jar
drwxr-xr-x+ 1 Rt None       0 Sep 22 19:35 ivy
-rw-r--r--  1 Rt None    8852 Feb 19  2010 ivy.xml
drwxr-xr-x+ 1 Rt None       0 Sep 22 19:35 lib
drwxr-xr-x+ 1 Rt None       0 Sep 22 19:35 librecordio
drwxr-xr-x+ 1 Rt None       0 Sep 22 20:08 logs
drwxr-xr-x+ 1 Rt None       0 Sep 22 19:36 src
drwxr-xr-x+ 1 Rt None       0 Feb 19  2010 webapps

Rt@Rt-PC ~/hadoop-0.20.2
$
```

FIGURE A.8

The result of unpacking *hadoop*.

D. Unpack *hadoop*

1. Open a new *cygwin* window and execute

```
tar -xzf hadoop-0.20.1.tar.gz
```

2. List the contents of the home directory

```
ls -l
```

A newly created directory called *hadoop-0.20.1* should be seen. Execute

```
cd hadoop-0.20.1
ls -l
```

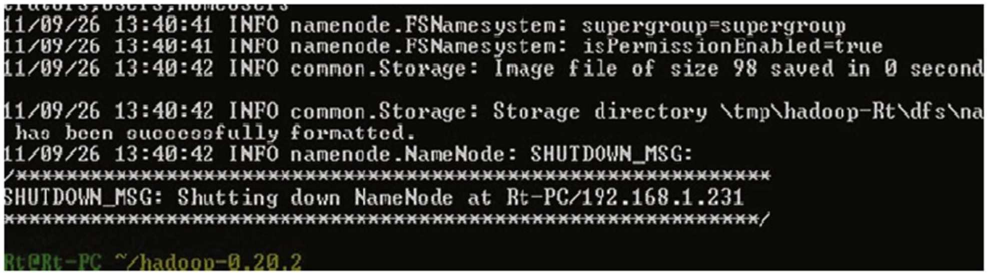
The files listed in Figure A.8 should be seen.

E. Set properties in configuration file

1. Open a new *cygwin* window and execute the following commands

```
cd hadoop-0.20.1
cd conf
explorer
```

2. The last command will cause the Explorer window for the *conf* directory to pop up. Minimize it for now or move it to the side.



```
11/09/26 13:40:41 INFO namenode.FSNamesystem: supergroup=supergroup
11/09/26 13:40:41 INFO namenode.FSNamesystem: isPermissionEnabled=true
11/09/26 13:40:42 INFO common.Storage: Image file of size 98 saved in 0 second

11/09/26 13:40:42 INFO common.Storage: Storage directory \tmp\hadoop-Rt\dfs\na
 has been successfully formatted.
11/09/26 13:40:42 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at Rt-PC/192.168.1.231
************************************************************/

Rt@Rt-PC ~/hadoop-0.20.2
```

FIGURE A.9

The creation of HDFS.

3. Launch *Eclipse* or a text editor such as *Notepad* ++ and navigate to the *conf* directory and open the file *hadoop*-site to insert the following lines between $< configuration >$ and $< /configuration >$ tags.

```
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9100</value>
</property>
<property>
<name>mapred.job.tracker</name>
<value>localhost:9101</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
```

F. Format the Namenode

Format the *namenode* to create a Hadoop Distributed File System (HDFS). Open a new *cygwin* window and execute the following commands:

```
cd hadoop-0.20.1
mkdir logs
bin/hadoop namenode -format
```

When the formatting of the *namenode* is finished, the message in Figure A.9 appears.

A.12 EXERCISES AND PROBLEMS

Problem 1. Establish an account to AWS. Use the AWS management console to launch an EC2 instance and connect to it.

Problem 2. Launch three EC2 instances; the computations carried out by the three instances should consist of two phases and the second phase should be started only after all instances have finished the first stage. Design a protocol and use *Simple Queue Service (SQS)* to implement the barrier synchronization after the first phase.

Problem 3. Use the *Zookeeper* to implement the coordination model in Problem 2.

Problem 4. Use the *Simple Workflow Service (SWF)* to implement the coordination model in Problem 2. Compare the three methods.

Problem 5. Upload several (10–20) large image files to an S3 bucket. Start an instance which retrieves the images from the S3 bucket and compute the retrieval time. Use the *ElastiCache* service and compare the retrieval time for the two cases.

Problem 6. Numerical simulations are ideal applications for cloud computing. Output data analysis of a simulation experiment requires the computation of confidence intervals for the mean for the quantity of interest [296]. This implies that one must run multiple batches of simulation, compute the average value of the quantity of interest for each batch, and then calculate say 95% confidence intervals for the mean. Use the *CloudFormation* service to carry out a simulation using multiple cloud instances which store partial results in S3 and then another instance computes the confidence interval for the mean.

Problem 7. Run an application which takes advantage of the *Autoscaling* service.

Problem 8. Use the *Elastic Beanstalk* service to run an application and compare it with the case when the *Autoscaling* service was used.

Problem 9. Design a cloud service and a testing environment; use the *Elastic Beanstalk* service to support automatic scaling up and down and use the *Elastic Load Balancer* to distribute the incoming service request to different instances of the application.

This page intentionally left blank