Cloud Computing

Amazon Web Services – Assignment 3

5/3/2013
C08502846
Seamus Barcoe

Table of Contents

Introduction

The purpose of this assignment is to design and implement a system for the creation and management of Amazon EC2 instances in an academic environment. The system is intended to mimic a system that would be used in an academic environment to create virtual machine instances on-demand for users in a given class.

Using AWS ECC2 service and other AWS Services

The services that I will be using in this project are as follows:

DynamoDB

This is a fast, fully managed NoSQL database service which makes it simple and cost-effective to store and retrieve any amount of data. All data items are stored on Solid State Drives which means faster data access. The drives are replicated across 3 Availability Zones for high availability and durability.

Amazon Elastic Compute Cloud (Amazon EC2)

This is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.

Amazon Simple Email Service

Amazon Simple Email Service (Amazon SES) provides a highly scalable and cost-effective solution for bulk and transactional email-sending. JavaMail provides a standard and easy-to-use API for sending mail from Java applications. The AWS SDK for Java brings these two together with an AWS JavaMail provider, which gives developers the power of Amazon SES, combined with the ease of use and standard interface of the JavaMail API.

Research

In order to carry out this assignment I researched the following topics:

How to store Data

Data will be stored in a database using Amazon Web Service DynamoDB. I will setup a table called Students and pass in each students info into this table. The following code sets up our table:

static AmazonDynamoDBClient dynamoDB;

```
String tableName = "Students";
            // Create a table with a primary hash key named 'name', which holds a
string
            CreateTableRequest createTableRequest = new
CreateTableRequest().withTableName(tableName)
                    .withKeySchema(new
KeySchemaElement().withAttributeName("name").withKeyType(KeyType.HASH))
                    .withAttributeDefinitions(new
AttributeDefinition().withAttributeName("name").withAttributeType(ScalarAttributeT
ype.S))
                    .withProvisionedThroughput(new
ProvisionedThroughput().withReadCapacityUnits(10L).withWriteCapacityUnits(10L));
                TableDescription createdTableDescription =
dynamoDB.createTable(createTableRequest).getTableDescription();
                System.out.println("Created Table: " + createdTableDescription);
```

How to retrieve data

To retrieve data from the students.csv file I did the following(see CSVFileReader.java):

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.StringTokenizer;


public class CSVFileReader
{
	String fileName;

	 ArrayList <String>storeValues = new ArrayList<String>();
	public CSVFileReader(String FileName)
	{
		this.fileName=FileName;
	}

	public void ReadFile()
	{
		try {
```

```
                        //storeValues.clear();//just in case this is the second call of the ReadFile
Method./
                        BufferedReader br = new BufferedReader( new FileReader(fileName));

                        StringTokenizer st = null;
                        int lineNumber = 0, tokenNumber = 0;

                        while( (fileName = br.readLine()) != null)
                        {
                                lineNumber++;
                                //System.out.println(fileName);
                                storeValues.add(fileName);
                                //break comma separated line using ","
                                st = new StringTokenizer(fileName, ",");

                                //reset token number
                                tokenNumber = 0;

                        }
                } catch (FileNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

        //mutators and accesors
        public void setFileName(String newFileName)
        {
                this.fileName=newFileName;
        }
        public String getFileName()
        {
                return fileName;
        }
        public ArrayList getFileValues()
        {
                return this.storeValues;
        }
        public void displayArrayList()
        {
                for(int x=0;x<this.storeValues.size();x++)
                {
                        System.out.println(storeValues.get(x));

                }
        }

}
```

I will enter in the data manually. The following code is a sample of how to enter data into our Students table.

Firstly I created the table:

```
String tableName = "Students";
            // Create a table with a primary hash key named 'name', which holds a 
string
            CreateTableRequest createTableRequest = new 
CreateTableRequest().withTableName(tableName)
                    .withKeySchema(new 
KeySchemaElement().withAttributeName("name").withKeyType(KeyType.HASH))
                    .withAttributeDefinitions(new 
AttributeDefinition().withAttributeName("name").withAttributeType(ScalarAttributeT
ype.S))
                    .withProvisionedThroughput(new 
ProvisionedThroughput().withReadCapacityUnits(10L).withWriteCapacityUnits(10L));
                TableDescription createdTableDescription = 
dynamoDB.createTable(createTableRequest).getTableDescription();
                System.out.println("Created Table: " + createdTableDescription);
```

Then I created a map which holds our data:

```
private static Map<String, AttributeValue> newItem(String name, String studentid, 
String username, String email, String surname, String firstname, String mi, String 
programme, int stage, String rstat)
    {
        Map<String, AttributeValue> item = new HashMap<String, AttributeValue>();

        item.put("name", new AttributeValue(name));
        item.put("studentid", new AttributeValue(studentid));
        item.put("username", new AttributeValue(username));
        item.put("email", new AttributeValue(email));
        item.put("surname", new AttributeValue(surname));
        item.put("firstname", new AttributeValue(firstname));
        item.put("mi", new AttributeValue(mi));
        item.put("programme", new AttributeValue(programme));
        item.put("stage", new AttributeValue().withN(Integer.toString(stage)));
        item.put("rstat", new AttributeValue(rstat));

        return item;
    }
```

Finally I added items to the table(2 examples):

```
// Add an item
            Map<String, AttributeValue> item = newItem("name1","c76000001", 
"i.smith", "ian.smith@nodit.ie", "smith", "ian", "J", "DT228", 2, "EL");
            PutItemRequest putItemRequest = new PutItemRequest(tableName, item);
            PutItemResult putItemResult = dynamoDB.putItem(putItemRequest);
            System.out.println("Result: " + putItemResult);


            // Add another item
            item = newItem("name2","c76000002", "j.kennedy",
"john.kennedy@nodit.ie", "kennedy", "john", "f", "DT228", 2, "RE");
            putItemRequest = new PutItemRequest(tableName, item);
            putItemResult = dynamoDB.putItem(putItemRequest);
            System.out.println("Result: " + putItemResult);
```

How to create EC2 Virtual Machines on EC2 system

In order to create Instances we do the following:

```
      // CREATE EC2 INSTANCES
             RunInstancesRequest runInstancesRequest = new RunInstancesRequest()
                 .withInstanceType("t1.micro")
                 .withImageId("ami-3bec7952")
                 .withMinCount(1)
                 .withMaxCount(1)
                 .withSecurityGroupIds("dt228-3-cloud")
                 .withKeyName("test.key")
                 .withAdditionalInfo("Ian Smith");

             RunInstancesResult runInstances =
ec2.runInstances(runInstancesRequest);
```

I then tagged the instances like so:

```
// TAG EC2 INSTANCES
             List<Instance> instances =
runInstances.getReservation().getInstances();
             int idx = 1;
             for (Instance instance : instances)
             {
               CreateTagsRequest createTagsRequest = new CreateTagsRequest();
               createTagsRequest.withResources(instance.getInstanceId()) //
                   .withTags(new Tag("Student 1", "Ian Smith" + idx));
               ec2.createTags(createTagsRequest);

               idx++;
             }
```

How to communicate – SNS / SES

In order to communicate to the students I first initialized the appropriate variables:

```
private static final String TO = "barcoe4@hotmail.com";
private static final String FROM = "barcoe4@hotmail.com";
private static final String BODY = "Hello World!";
private static final String SUBJECT = "Hello World!";
```

I then Setup JavaMail to use the Amazon Simple Email Service by specifying the "aws" protocol:

```
Properties props = new Properties();
props.setProperty("mail.transport.protocol", "aws");

props.setProperty("mail.aws.user", credentials.getAWSAccessKeyId());
props.setProperty("mail.aws.password", credentials.getAWSSecretKey());

Session session = Session.getInstance(props);
```

I then created the message like so:

```
// Create a new Message
Message msg = new MimeMessage(session);
msg.setFrom(new InternetAddress(FROM));
msg.addRecipient(Message.RecipientType.TO, new InternetAddress(TO));
msg.setSubject(SUBJECT);
msg.setText(BODY);
msg.saveChanges();

// Reuse one Transport object for sending all your messages
// for better performance
Transport t = new AWSJavaMailTransport(session, null);
t.connect();
t.sendMessage(msg, null);

// Close your transport when you're completely done sending
// all your messages
t.close();
```

How a user access virtual machines

In order for the students to access the virtual machines they will need to use Putty. Putty is an SSH and telnet client, developed for Windows platform. It is open source software so it is free for students to use. Students will need the Host Name or IP Address and the port number they are trying to connect to. They also may need to configure the correct Tunnel settings in Putty.

Completed system

This completed system must consist of the following: It must be driven by data in students.csv. This is a table which holds all the relevant information required to identify each student uniquely. Each student will have a virtual machine created for them. The machine will have Ubuntu OS installed and have a type T1 computer specs.

Once a machine is created for each user, they will be notified using AWS SES and SNS messaging system. This will give them the details and instructions that they need in order to use their new virtual machine.

Conclusion

I learned a lot about Amazon Web Services throughout this assignment. I learned that there are endless possibilities when it comes to setting up and creating your own working environment. It is easy to maintain, secure and affordable as you only pay for what you use.

There is a lot of help on amazon’s website: http://docs.aws.amazon.com. They explain things very well and go through different aspects of AWS step by step with good samples included.