Amazon Web Services (AWS) Assignment

Challenges in Cloud and Mobile Computing (COMS E6998-7)
Due: 11/02/2011 (palindrome!), before class

Overview

For this assignment, you will build a basic application using Amazon's cloud services. You will become familiar with how to launch and access virtual machines using EC2, and learn how to execute code on these machines. You will learn how to use Amazon's Simple Queue Service (SQS), as well as Amazon's Simple Storage Service (S3). And, as an extra challenge, you can create a public facing web page for your application!

There are several ways of doing this assignment, including a variety of languages you can use, and a variety of ways of interacting with AWS. For example, in order to interact with AWS, you can use the AWS management console, command-line tools, SOAP APIs, REST APIs, as well as libraries in other languages. However, I will only provide nice detailed instructions on how to do this assignment in one particular way (you are welcome to do it another way, but I will be unable to help you along the way if you run into issues). Specifically, I will give guidance on how to use the AWS management console to create AWS resources, and how to use the AWS SDK for Java to interact with these resources (both from within AWS, and outside of AWS).

In order to do the assignment in the way I will outline, which uses the AWS management console for various purposes, you will have to set up an AWS account. To set up an account, you will have to provide a credit card number, but this credit card will not be charged because everything we will do in this assignment falls under the "free usage tier" of AWS (http://aws.amazon.com/free/). If you are not comfortable providing a credit card number, you are very welcome to do the assignment without using the AWS management console, and instead using other means (eg, REST APIs, command-line tools, or AWS SDK for Java) which simply require an access key id/secret access key pair (which I can provide to you). Let me know ASAP if you would like to do the assignment in this way, and I will provide you with the keys.

For a great overview of AWS's offerings, and how to use AWS to build reliable and scalable web services, read this whitepaper: http://d36cz9buwru1tt.cloudfront.net/AWS_Web_Hosting_Best_Practices.pdf

Grading + What to turn in

By the due date, I will require that everyone email me (avnermay@cs.columbia.edu) a zipped folder with all of their code. Additionally, I will ask that, as soon as possible (but no later than the due date) everyone sign up for a 10 minute slot to come to CEPSR 6LW5 (the Social and Mobile Lab) to demo their application for me. Please sign up for your 10 minute slot here (http://www.doodle.com/f8fwuhebf8r88w87).

This assignment has a "setup section", and then three parts: Part 1 must function properly when you demo the assignment to me to get full credit for the assignment. If it does not, you can still get some partial credit if you turn in some code which makes sense. Part 2 is bonus. Part 3 is "bonus++" (aka, just for fun), and those who do it will get the opportunity to demo their application to the class at the end of the semester!!

Important Tips

- Remember to only launch micro instances with the free AMI ("Basic 32-bit Amazon Linux AMI 2011.09 (AMI Id: ami-7f418316)"). All other instance types cost money! (for more info on exactly what is free, see http://aws.amazon.com/free/)
- When you are entirely finished with this assignment, and have already demoed it, make sure to terminate all of your EC2 instances, and release all of your AWS resources. If you leave them on, you will eventually begin to be charged for their use (after 12 month "free usage tier" period is over). Note that once you terminate an instance, everything that was stored on that instance is lost.
- For various parts of this assignment, you will be able to test both client and server parts of your application on your local machine, to avoid the hassle of redeploying your server-side changes constantly onto your EC2 machines.

Part 0: Setup + Learning how to use AWS

1. Sign up for an AWS account (http://aws.amazon.com/)
    a. As I mentioned, you will have to provide credit card info
    b. Additionally, you will provide a phone number, and will receive an automated call to confirm your identify (you will type into the phone a 4-digit code they show you on the website)
    c. Within a few minutes you should receive an email confirming that your account has been set up.
2. Launch a 32-bit linux micro-instance in EC2 via the AWS management console
    a. The AWS management console is a central location where you can create, release, interact with, and monitor your AWS resources. It has a nice GUI, and helpful wizards for performing common tasks (eg, launching EC2 instances, creating SQS queues, creating buckets in S3). Take some time to look around and see what is available.
    b. Helpful resource for this step: http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/
    c. Go to AWS management console (http://aws.amazon.com/console/).
    d. Under EC2 tab, click "Launch Instance"
    e. Select "Basic 32-bit Amazon Linux AMI 2011.09 (AMI Id: ami-7f418316)" (the star next to this item specifies it is free)
    f. Select to run 1 micro-instance (don't need to specify availability zone)
    g. Select defaults on "Advanced Instance Options" page
    h. Give your new instance a name by specifying a value of the "Name" key.
    i. Create a new secret-key pair, and download the private key to a location on your machine.
    j. Configure your security group: Security groups offer a way to restrict the incoming traffic to a machine, by port, protocol, and IP address. When configuring your security group, ensure you allow access to the machine via the ports below:
        i. SSH over port 22
        ii. FTP over port 21 (custom TCP rule)
        iii. HTTP over port 80
        iv. HTTPS over port 443
    k. Review your details, and click "Launch". This will launch your first EC2 instance!
3. SSH into your machine

a. In order to SSH into your machine, you will need the public DNS name for your EC2 instance. To get the name, go to the "Instances" tab in the AWS management console and select your newly launched instance. The details section should appear at the bottom of the screen, and the public DNS name will be listed there.
b. For details on how to SSH into the machine, go here (http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/) and navigate to the "Connect to Linux/UNIX instance" page.
    i. You will need the private key you downloaded above. If you are a windows user, you will have to use PuttyGen to convert this "*.pem" file to a "*.ppk" file, and then use Putty to connect to the machine (specifying this .ppk file in the SSH->Auth tab).
    ii. Log on as "ec2-user"

4. FTP into your machine
    a. Using an FTP client (eg, WinSCP), connect to the EC2 machine
        i. Specify public DNS name
        ii. Enter "ec2-user" as user name
        iii. Specify "Private key file"
    b. You can now easily explore your EC2 machine's file system, copy files to it, move files around, delete files, etc.
5. Set-up Eclipse on your personal machine
    a. Install Eclipse EE version:
        i. If you do not already have Eclipse, download it here (http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/indigosr1). Note that I believe you need the "EE" version of Eclipse for some of the "AWS Toolkit for Eclipse" features to install properly.
    b. Install "AWS Toolkit for Eclipse"
        i. This video is helpful: http://d36cz9buwru1tt.cloudfront.net/videos/eclipse-java-sdk-video.html
        ii. Go to Help->Install new software.
        iii. Enter "http://aws.amazon.com/eclipse" as the "Work with" site.
        iv. Select all the AWS toolkit items to install, and click through all of the prompts (accept license agreements, etc)
        v. Restart Eclipse after the AWS toolkit has been installed.
        vi. After you restart Eclipse, you should notice various things which are different in Eclipse:
            1. You can create a new "AWS Java Project" or "AWS Java Web Project"
            2. Click on the little orange cube icon in the icon list at the top of the screen, to see what else is available (for example, click "Open AWS Management Perspective", and "Show AWS Explorer View")
            3. Play around with these options a little.
        vii. Configure AWS Toolkit in Eclipse
            1. Find your AWS access key id and secret access key by going to http://aws.amazon.com/, clicking on the "Account" tab, and choosing "Security Credentials".
            2. Copy these into Eclipse under Window->Preferences->AWS Toolkit (this ensures that every new AWS Java project you create has the keys already configured in the AwsCredentials.properties file).

6. Look at, and run, the AWS java samples in Eclipse
    a. File -> New -> Project: Select "AWS Java Project", and click next.
    b. Choose a project name, and select all of the Samples/Apps available.
    c. Look at the code for all of these samples to get a sense for how they work.
    d. You can run these samples as well.
    e. Notice that the Java API calls get translated into Https requests to the AWS servers, which appear in red in the console window when you run any of the Java AWS code. It is interesting to look at what goes out over the wire.
    f. Notice that if you put a breakpoint in some places in these samples, and then go to the AWS management console and refresh it, you will be able to see how these samples are creating and using AWS resources.
7. Run a simple Java program, which performs some AWS API calls, on an EC2 instance
    a. First, write a simple Java application which includes some AWS API calls ("Hello World" style). In order to see how to do this, you should reference the AWS SDK.
    b. Create jar file of this simple app from Eclipse
        i. File->Export
        ii. Under "Java", Select "Runnable JAR File".
        iii. Under "Launch configuration", select the Java app you just wrote.
        iv. Select the export destination (including name of output file).
        v. Select "Extract required libraries into generated JAR".
    c. Using an FTP client, upload this JAR file onto your EC2 machine.
    d. SSH into your EC2 machine, navigate to the directory with the jar file, and execute the command "java -jar <name-of-jar-file>"
        i. Note: conveniently, the JRE (Java Runtime Environment) is already installed on the Linux image you are using, so there is no need to install it. Furthermore, the directory containing the "java" executable is by default in the "PATH" on the machine (so you can simply invoke "java" to execute this program).
    e. Make sure that your application ran correctly by verifying the output.

**Helpful resource*: Documentation for the AWS SDK for Java:
http://docs.amazonwebservices.com/AWSJavaSDK/latest/javadoc/

Part 1: Creating a basic application with SQS, EC2, and S3.

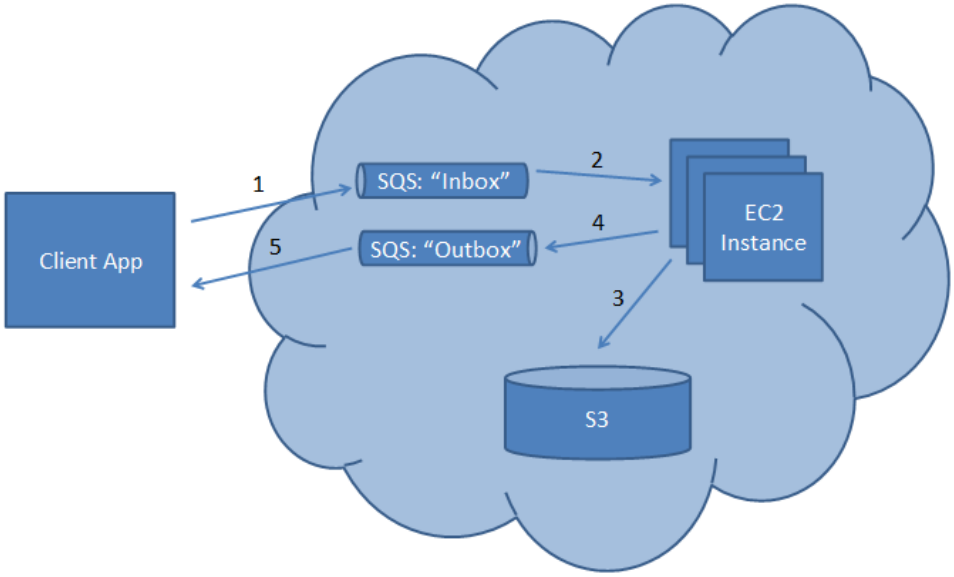
For this part of the assignment, you will create an AWS application which calculates the min, max, product, sum, and sum of squares of a list of integers. The application will consist of various components:

- Client App: Runs locally on your machine. User inputs a comma-separated list of integers, and this is sent as a message into the SQS request queue ("inbox"). The application will then wait until a response to this message appears in the SQS response queue ("outbox"), and then delete this message from the "outbox" queue. (Note: it is important that the client only pay attention to the message which is the response to the request it sent. You should think of a clever way to do this.). It should print out the response in the format "Min:a Max:b Product:c Sum:d Sum-of-Squares:e". (Note: sum of squares of {a,b,c} = a^2 + b^2 + c^2).
- SQS request queue ("inbox"): This is where requests are stored until they are processed.
- SQS response queue ("outbox"): This is where responses are stored until the user sees the response to his message.
- 2+ EC2 worker instances: These instances read from the "inbox" queue, process the requests, log the request/response info as a text file in S3 bucket, send a response message to the

"outbox" queue, and delete the message from the "inbox" queue once they have finished processing it.

- S3 bucket: The S3 bucket will contain a log of all the requests/responses which the system has processed. This information should at least include: MessageIds and bodies of both request and reply messages, timestamp of when request was processed, and some identifier (eg, IP address and/or machine name) of the EC2 machine which processed the job. Each request/reply pair should be logged in their own text files.

Architectural diagram of Part 1

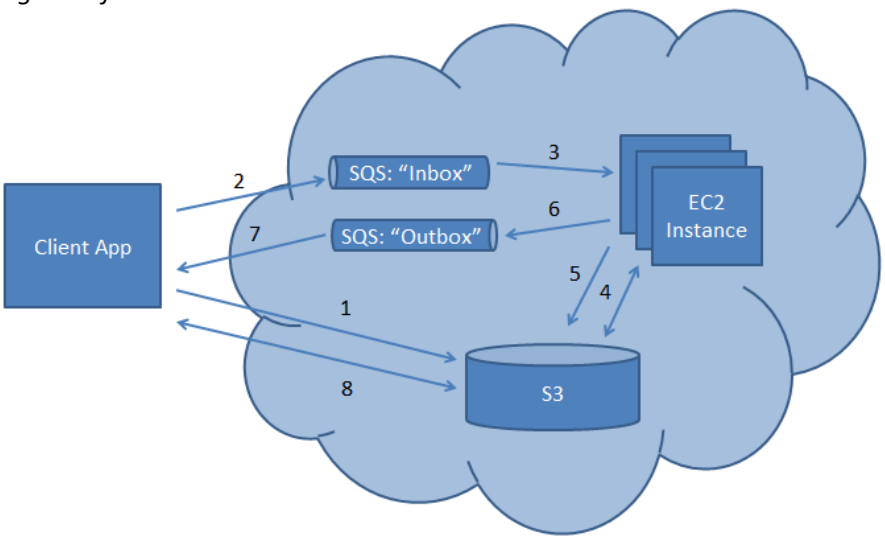


Part 2: Creating an image-processing application with SQS, EC2, and S3 (bonus)

For this part of the assignment, you will create an AWS application which resizes (or in some other way processes) an image which a user uploads to the cloud. It should be composed of the following components:

- Client App: Uploads images to bucket in S3, places the "key" to this uploaded image in the "inbox" queue, and waits for this message to be processed. Once this message is processed, a message will be placed in the "outbox" queue with the "key" to the resized image. Once this message arrives, the Client App should download the resized image from S3. (once again, make sure to correlate between requests and replies, as in Part 1)
- SQS request/response queues: same idea as part 1.
- 2+ EC2 worker instances: These instances wait for messages to appear in the "inbox" queue, and when they do, they retrieve the referenced image file from S3, perform some image-processing on the image, place the resulting image in S3, place the "key" to the "result" image file in the "outbox" queue, and delete the message from the "inbox" queue.
    - With regard to how to perform image processing: See Linux "convert" command, which you can install via the command "sudo yum install ImageMagick" on your EC2 machine. Pick some function of "convert" which you like, and have your EC2 machines perform this function on the uploaded images. Suggestions include: resize, paint, etc. If you're feeling ambitious, you could even have the user specify what function they would like to perform.
- S3 bucket: This will contain the original and processed images.

Architectural diagram of Part 2

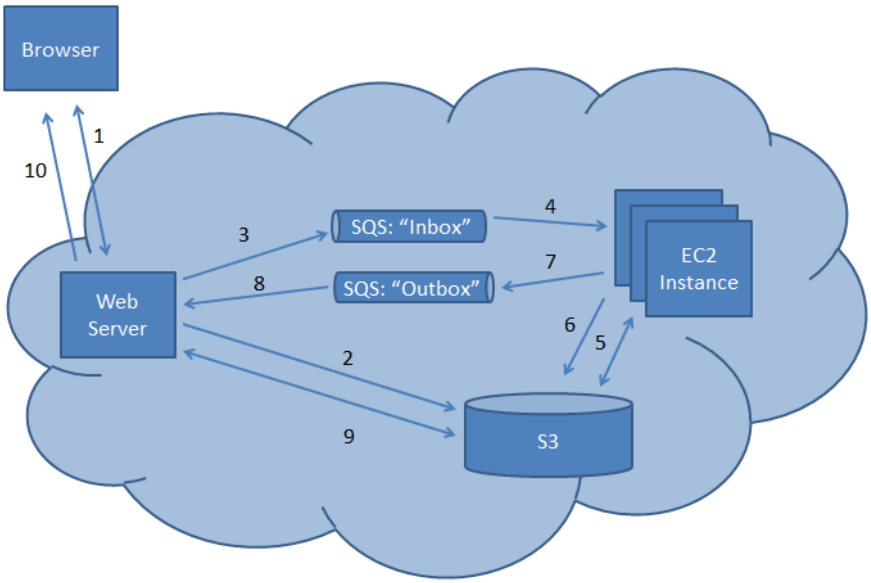


Part 3: Create a web front-end for the application in part 2 (bonus++)

Now, in order to make your awesome image-processing application consumable by others (in particular, people without your secret key pair), as well as easier to use in general, create a simple website as a front-end to your application in part 2. To do this, you will have to create an EC2 instance to serve as your web server (you will have to install Apache and Tomcat on it). People who do this will get a chance to demo their application to the class! (Feel free to work on this part even past the due date)

If you would like to go even further: Feel free to play around with what you have done up to this point, and get creative. See if you can use other parts of AWS (for example, Auto-scaling, elastic load balancer, EBS, RDS…). You can try out different architectures/designs for the application. You can even have your application do something other than image processing. In other words, do whatever you want, build something cool, and try to use other services offered by AWS. Have fun!

Architectural diagram of Part 3



Resources

- AWS developer resources:
  - http://aws.amazon.com/documentation/
  - http://aws.amazon.com/articles
  - http://aws.amazon.com/java/
  - There are lots of helpful resources on the AWS web page…
- Documentation for AWS SDK for Java: http://docs.amazonwebservices.com/AWSJavaSDK/latest/javadoc/
- Whitepaper on AWS and scalable/reliable cloud architectures: http://d36cz9buwru1tt.cloudfront.net/AWS_Web_Hosting_Best_Practices.pdf
- Article on image-processing cloud application (with various different architectures, using SQS and S3): http://sqs-public-images.s3.amazonaws.com/Building_Scalabale_EC2_applications_with_SQS2.pdf

Questions?

- Email avnermay@cs.columbia.edu if you have any questions about this assignment.
- Also, remember that my office hours are Tuesdays from 5:30-6:30 pm.