Assignment 1

1. Lab Assignments: Due Dates & Grading

The grading for lab assignments is up to 4 points of the total grading of the course: Lab assignments (4 points) and Exam (6 points). The partial grading and due date of Lab Assignments are:

LAB GRADING (4 points)		
Assignment 1 (2 points)	**AWS Architecture for Tagged Files**	
	Delivery	20th October 20:00 (*faitic*)
	Assessment	Monday, 23th October. 11:00 (*T110*)
Assignment 2 (2 points)	**Map Reduce on Distributed Data**	
	Delivery	24th November 20:00 (*faitic*)
	Assessment	Monday, 27th November. 11:00 (*T110*)
EXAM (6 points)		

2. AWS Architecture for tagged files

For this part of the assignment, you will create AWS architecture that stores documents (pdf files) tagged with terms automatically extracted from the document. The architecture can be accessed from a ClientApp (Figure 1) and from a Web Browser (Figure 2). The architecture should be composed of the following components:

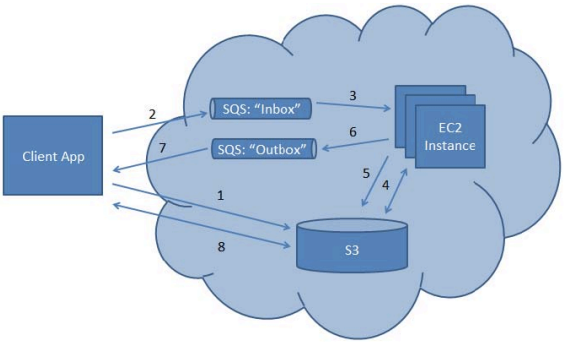

Figure 1 Running with a Java Client Application

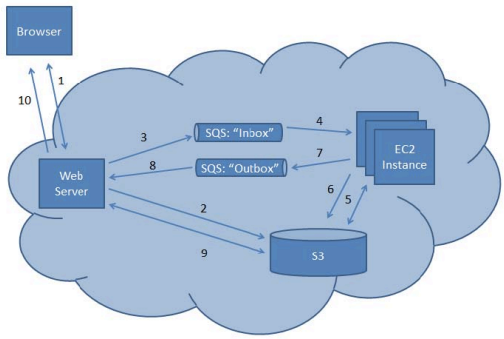

Figure 2 Running from a Web Front-end

- ***Client App***: Runs locally on your machine and communicate via message throughout a 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. The application allows the user to (i) upload a document (pdf file) to a bucket in S3 and automatically tagged it; and (ii) search the documents in the S3 bucket by tag and download them.

  - (i) The ClientApp uploads the pdf document D to S3, places the key to the S3 object D in the "inbox" queue, and waits for this message to be processed. Once D is processed (tag extracted), a message will be placed in the "outbox" queue with the extracted tag T. Once this message arrives, the ClientApp should show the tag T.
  - (ii) When the user enters a tag T to look up, the ClientApp put T 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 list of at most 3 files with the tag T. Once this message arrives, the Client App should show the list of files and download them.

- **Asynchronous Communication by *SQS request/response*** queues: ***SQS request queue ("inbox"),*** where requests are stored until they are processed; and ***SQS response queue ("outbox"),*** where responses are stored until the Client App retrieves the response. You can use message attributes of SQS to distinguish tagging and searching messages.

- ***S3 bucket***: The S3 bucket will contain the original documents and an index file that relates the documents with their tags.

- ***2 or more EC2 worker instances***: Each worker instance contains 2 applications: one for tagging (TaggingApp) and the other for searching (SerachingApp). These applications wait for appropriate messages (tagging or searching) to appear in the "inbox" queue, and when they do:
  - ***TaggingApp*** retrieves the document from S3, convert to text (use a library, e.g. Apache PDFBox®) and extract the more frequent word that will be considered the tag **T** of the file. Also TaggingApp maintains a **FileIndex** (also in S3) with the Key of the S3 document and the tag. Then, the application places the tag in the "outbox" queue and deletes the message from the "inbox" queue.
  - ***SearchingApp*** *consults the* ***FileIndex*** to obtain up to 3 documents (pdf files) with a tag ***T.*** Then, the application places the key of these S3 documents in the "outbox" queue, and delete the message from the "inbox" queue.

- ***Web front-end***: Now, in order to make your application consumable by others (in particular, people without your secret key pair), create a simple website as a front-end to your application. 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).

3. Assessment

- The basic assessment (**1,5 points**) will take into account:
  - Quality of design and programming.
  - Level of automatization of the architecture (launching, deployment and stopping).
  - Correct behaviour in simple and complex scenarios (more than one client)
- If you would like to go even further, also, improvements are encouraged (**0,5 points**).

4. Instruction for submission

Each student will submit the assignment to *faitic* by using a single ".zip" file, named "HereYourCompleteName.zip". This compressed file must have the following structure:

- *readme.txt*
- *Description.pdf*: a document no longer than 5 pages, which includes the following sections: (1) Client software design; (2) Service software design; (3) Tests; and (4) Improvements (if applicable). The document must be written in English.
- *ClientApp*: Directory with the source code of client software, a *build.xml* file (or equivalent) and a file *readme.txt* if necessary for compiling the code. For simplicity you can choose to include in this directory

the entire structure of the *Eclipse* project for the client software.

- I*nstanceCode:* Directory with the source code of service software, a *build.xml* file (or equivalent) and a file *readme.txt* if necessary for compiling the code. For simplicity you can choose to include in this directory the entire structure of the *Eclipse* project for the service software.
- *Web*: Directory with Web content to access service software.

5. Setting Up the Environment

Creating an AWS Account

- Access to AWS console. See **What Should I Try First?** and consult http://docs.aws.amazon.com/gettingstarted/latest/awsgsg-intro/gsg-aws-get-started-first.html)
- Do not forget to consult conditions for the Free Tier: https://aws.amazon.com/free/

Launching and instance

- Launch a Free Tier instance in EC2 via the AWS management console at: http://aws.amazon.com/console/
- Under EC2 tab, click "Launch Instance" and select an AMI from the Free Tier.
- Select to run 1 micro-instance (don't need to specify availability zone).
- Select defaults on "Advanced Instance Options" page and give your new instance a name by specifying a value of the "Name" key.
- Create a new secret-key pair, and download the private key to a location on your machine.
- 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:
  - SSH over port 22
  - FTP over port 21 (custom TCP rule)
  - HTTP over port 80
  - HTTPS over port 443
- Creating Billing Alerts.
- Review your details, and click "Launch". This will launch your first EC2 instance!

SSH and FTP into your EC2 instance

- You will need the public DNS name (or public IP) for your EC2 instance. To get it, 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.
- For details on how to SSH into the machine, go (http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/) and navigate to the "Connect to Linux/UNIX instance" page.
- For ssh into your machine:
  - Your key must not be publicly viewable for SSH to work. Use "chmod" if needed.
  - Log on as "ec2-user".

    ssh -i gxx.pem ec2-user@54.171.7.154
    (with the Public IP of the instance)
- For FTP into your machine, use a FTP client:
  - Specify public DNS or public IP address.
  - Enter "ec2-user" as user name
  - Specify "Private key file"

AWS SDK for Java (using Eclipse)

The AWS SDK for Java helps take the complexity out of coding by providing Java APIs for many AWS services including Amazon S3, Amazon EC2 and more. The single, downloadable package includes the AWS Java library, code samples, and documentation. If you use Eclipse EE, you can install "AWS Toolkit for Eclipse". To install "AWS Toolkit for Eclipse"; use http://aws.amazon.com/eclipse as the "Work with" site in Eclipse. After that, you can create a new "AWS Java Project" or "AWS Java Web Project". Also new perspectives are available: "Open AWS Management Perspective", "Show AWS Explorer View".

- Configure AWS Toolkit in Eclipse:
  - 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".
  - Copy these into Eclipse under Preferences->AWS Toolkit (this ensures that every new AWS Java project you create has the keys already configured in the *AwsCredentials.properties* file).
- Look at, and run, the AWS java samples in Eclipse:
  - File -> New -> Project: Select "AWS Java Project", and click next.
  - Choose a project name, and select all of the Samples/Apps available.
  - Look at the code for all of these samples to get a sense for how they work.
  - You can run these samples as well.
  - 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.
  - 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.
- Run a simple Java program:
  - First, write a simple Java application that includes some AWS API calls ("Hello World" style), on an EC2 instance. In order to see how to do this, you should reference the AWS SDK.
  - Create jar file of this simple app from Eclipse
  - Using an FTP client, upload this JAR file onto your EC2 machine.
  - SSH into your EC2 machine, navigate to the directory with the jar file, and execute the command "java -jar <name-of-jar-file>".[1]
  - Make sure that your application ran correctly by verifying the output.

6. Development structure and working order

The recommended structure during the development phase of this assignment is shown in Figure 3. Note how the code that is later to be run on your EC2 instance is run on your local computer during this development phase (saves a lot of time). Once all the code is working as intended the EC2 instance Code can be moved to the Amazon Cloud, and the final structure will look like Figure 4.

[1] 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).

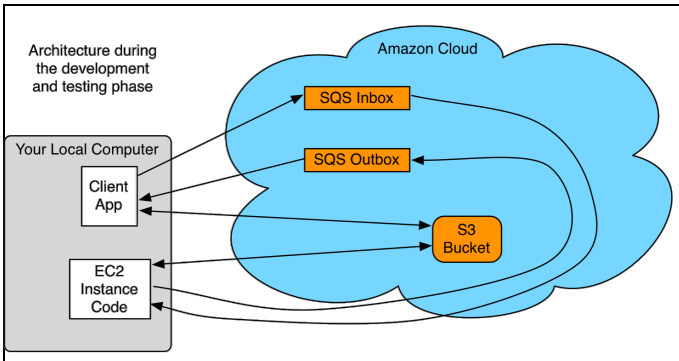

Figure 3 Development

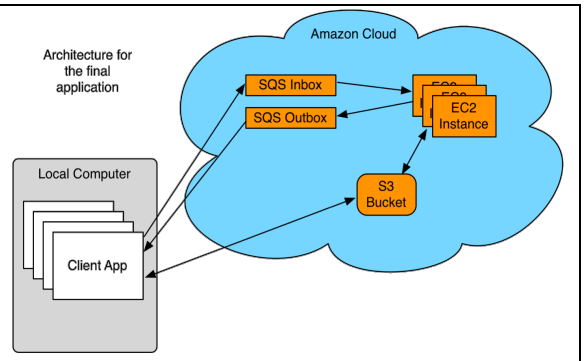

Figure 4 Final Application

We suggest the following working order:

- Study some "AWS java sample code"
  https://docs.aws.amazon.com/java-sdk/latest/developer-guide/java-dg-samples.html
- Create your own S3-bucket and do the following actions: Put an object in the bucket and get and object from the bucket;
- Create your own SQS-queue and do the following actions: Put a message in the queue; Read a message from the queue; Delete a message in the queue; Change the "VisibilityTimeout" for the queue; Enable "Long Polling" for the queue.
- Start developing the Client App on your computer.
- Start developing the EC2 instance-code. This is the code that is to be run on the EC2 instance in the AWS Cloud but should be running on your local computer during development (Figure 3). Once everything works as intended this will be moved to an EC2 instance (Figure 4).
- Export the working EC2 instance-code to an EC2 instance.
- Make the script run on start-up
  - Log into your modified EC2 instance (created in the previous step)
  - Open /etc/rc.local using an editor of your choice.
  - After the line touch /var/lock/subsys/local add the following line:
    java -jar /home/ec2-user/RunnableJAR-file.jar &
  - Now the JAR-file should execute on startup of the instance
- Create an AMI-image from your modified EC2 instance
  - Go online to your AWS Management Console
  - Go to the EC2 Dashboard
  - Select your modified EC2 instance
  - Choose Actions -> Image -> Create
  - After a while your new AMI-image should be visible under the AMIs-menu
- Launch a new Instance from your newly created AMI-image
  - Select your newly created AMI-image
  - Choose Action -> Launch
  - Launch one more and you have your 2+ instances.
- Full Test
- Please, after the assessment of this assignment, do not forget to:
  - Terminate all the EC2 instances.
  - Delete your SQS-queues
  - Delete your S3-bucket
  - Destroy your AMI-image
  - Destroy your credentials