Lab 6 Specification – Exploring SDK and Web Services
Due (via your git repo) no later than 2 p.m., Monday, 2nd Nov 2020.
50 points

Lab Goals

- Refresh our knowledge in using AWS
- Implement automation procedures using AWS SDK on the Cloud Platform.
- Implement web services on the Cloud Platform.

Summary

We will do a few hands-on exercises to automate creation, starting, stopping, terminating virtual machines in the Cloud Platform. In addition, we will implement web services on a Cloud Instance. During this implementation. we will learn how to publish and consume web services. We had started discussing the techniques, by which we can publish Cloud services through Web Services. It is the right time to understand how the web service setup works practically, and doing the basic networking to achieve this goal. Additionally, we will watch a video clip, which is a segment of a video series in lynda.com by David Gassner, to further solidify our understanding.

Learning Assignment

If not done so already, please read all of the relevant ”GitHub Guides”, available at https://guides.github.com/, which explains how to use many of the features that GitHub provides. In particular, please make sure that you have read guides such as ”Mastering Markdown” and ”Documenting Your Projects on GitHub”; each of them will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, you should also read

- **Servlets reading material in the class (lesson-4 and part-3) repository.**
- **If not done already, Basic Networking reading material in the class repository.**

Assignment Details

It is required for all students to follow the honor code. Some important points from the class honor code are outlined below for your reference:

1. Students are not allowed to share code files and/or other implementation details. It is acceptable to have a healthy discussion with your peers. However, this discussion should be limited to sharing ideas only.
2. Submitting a copy of the other’s program(s) is strictly not allowed. Please note that all work done during laboratory sessions will be an opportunity for students to learn, practice, and master the materials taught in this course. By doing the work individually, students maximize the learning and increase the chances to do well in other assessments such as lab assignments, skill tests, projects, etc.

At any duration during and/or after the lab session, students are recommended to team up with the Professor and/or the Technical Leader(s) to clarify if there is any confusion related to the items in the lab sheet and/or class materials.

Section 1: Web Services

This section is worth 15 points. The points breakdown is provided below:

- Task 1 = 15 points, a maximum of 3 points awarded for each question.

The term Web service (WS) is either:

- a service offered by an electronic device to another electronic device, communicating with each other via the World Wide Web, or
- a server running on a computer device, listening for requests at a particular port over a network, serving web documents (HTML, JSON, XML, images), and creating[clarification needed] web applications services, which serve in solving specific domain problems over the Web (WWW, Internet, HTTP) In a Web service a Web technology such as HTTP is used for transferring machine-readable file formats such as XML and JSON.

In practice, a Web service commonly provides an object-oriented Web-based interface to a database server, utilized for example by another Web server, or by a mobile app, that provides a user interface to the end-user. Many organizations that provide data in formatted HTML pages will also provide that data on their server as XML or JSON, often through a Web service to allow syndication, for example, Wikipedia's Export. Another application offered to the end-user may be a mashup, where a Web server consumes several Web services at different machines and compiles the content into one user interface. **(text taken from wiki)**

At the heart of modern web application architectures are web services, which drive communication on the web. Explore web services in depth in this tutorial. Watch more at http://www.lynda.com/OData-tutorials/....

This tutorial is a single movie from the Foundations of Programming: Web Services course presented by lynda.com author David Gassner. The complete course is 3 hours and 17 minutes and explores the history, types, and various implementations of web services—the standard method of communicating between applications and across the web. **text taken from YouTube video description**

In this section, we will watch a video and reflect on the points discussed in the clip. This reflection is instrumental to further advance our understanding of publishing services in the Cloud and in general to foster the learning from our recent discussions on how to publish cloud services using web services. To complete this part, it is required to do the following:

- **Task 1:** Watch this short video clip, by using the link below:

  `https://www.youtube.com/watch?v=u80uPzhFYvc`

  After watching the video, create a markdown file and name it as `video-reflection`. In this file, provide detailed answers to the questions provided below:

  1. What is a Web Service?
  2. In this video, the author talks about shared vocabulary. What is the shared vocabulary used in AXIS2 client server communication?
  3. The author discussed Ajax. Compare Ajax with servlets.
  4. What is a message format?
  5. What are the elements of the Web Service API?

Section 2: AWS SDK Development

This section is worth 15 points. The points breakdown is provided below:

- Task 2 = 15 points

In this section, we will setup the AWS SDK code to automate creation, starting, stopping, and terminating instances. The underlying principle that tried out here are the publishing software as a service using SDK. To complete this part, it is required to complete the tasks listed below:

1. **Task 2:**
   - Login to the AWS interface. Copy the AWS CLI by clicking on the Account Details button in the vocareum homepage. This includes access key, secret key, and session token.
   - Terminate all running instances. Delete all the current key-pairs, and the security groups.
   - Create a new instance. In this process, configure and create a new security group, and apply the required networking. For example, it is required to have all traffic with the source as anywhere and destination as anywhere applied. Finally, create/configure a new key pair and download the key pair. For example, a sample name for the key-pair is amohan-ec2-auto. Please change the file name to your name.
   - Since a new key-pair and security group are added, we are ready to terminate the running instance. At this point, we should have no running instances. The idea is to automate creation of virtual machines using the starter-code.
   - Edit the aws/src/cloud/Driver.java program. Between lines 14 and 18, fill out the string values with the corresponding values from your AWS.
   - Compile and Execute the program, using compile.sh and run.sh files. Windows users should use the exec.bat file. At this point, a new instance should be created in your AWS console.

- Comment lines between 24 and 32 in the Driver program. Uncomment line 34. Compile and Execute the program, using compile.sh and run.sh files. Windows users should use the exec.bat file. At this point, the new instance should be terminated.
- Comment line 34. Uncomment lines between 24 and 32 in the Driver program. In line number 25, edit the argument for createInstances. Change the argument from 1 to 3. Compile and Execute the program, using compile.sh and run.sh files. Windows users should use the exec.bat file. At this point, three new instances should be created in your AWS console. In your logs folder, there should be a new log file inserted with the logs of running the commands on the cloud instances. Copy the **log-output file** to the aws directory in the submission folder.
- Create three new methods in the VMProvisioner class. The methods names should be start(String instanceId), stop(String instanceId), and terminate(String instanceId). It is worth noting that the methods startInstance(), stopInstances(), and terminateInstances() start, stop, and terminate all instances respectively. Complete the implementation for the new methods start, stop, and terminate based on the implementation provided in the startInstance(), stopInstances(), and terminateInstances() methods. The main difference is that rather than start, stop, and terminate all instances. these new methods start, stop, and terminate the instance provided through the method argument. That is these new methods start, stop, and terminate only one instance per method call.
- Comment lines between 24 and 32 in the Driver program. Add the necessary method calls to the Driver program for the new methods created in the VMProvisioner class.
- Stop one of the three instance running using the new stop method. You may provide the instance id of any instance. Capture a screenshot of the AWS web interface console, clearly showing the instance state as stopped. Name the screenshot as **aws-1** and store the output file in the submission/aws folder.
- Comment the method call to the Stop method. Start the instance that was stopped in the previous step using the new start method. You should use the same instance id as the one used in the previous step. Capture a screenshot of the AWS web interface console, clearly showing the instance state as started and running. Name the screenshot as **aws-2** and store the output file in the submission/aws folder.
- Comment the method call to the Start method. Terminate the instance that was started in the previous step using the new terminate method. You should use the same instance id as the one used in the previous step. Capture a screenshot of the AWS web interface console, clearly showing the instance state as terminated. Name the screenshot as **aws-3** and store the output file in the submission/aws folder.
- Comment the method call to the Terminate method. Terminate the two remaining instance running using the terminateInstances method. Capture a screenshot of the AWS web interface console, clearly showing the instance state as terminated. Name the screenshot as **aws-4** and store the output file in the submission/aws folder. At this point, all instances should be terminated.

Section 3: Web Services

This section is worth 20 points. The points breakdown is provided below:

- Task 3 = 10 points
- Task 4 = 10 points

In this section, we will publish and consume web services. To complete this part, it is required to complete the prerequisite in the tasks tried out in class Monday, Oct 26th. The release notes related to the steps implemented in the class to develop a web service for year of birth calculator program is provided in the starter-code repository. If all the set up is done correctly, then you should be able to publish the service on a new cloud instance, and try out the Java and Python version of the client.

1. **Task 3:**
   - Create a copy of dob-service folder in the web-service/server directory and name the new folder as calc-service.
   - Rename the dob folder as calc folder.
   - Rename the DobService class in src/com/web directory to CalculatorService.
   - Remove getYear() method, and introduce new methods in the CalculatorService namely, add(int arg0, int arg1), sub(int arg0, int arg1), multiply(int arg0, int arg1), and divide(int arg0, int arg1). All these method should return strings, sample output:

     Addition Result: res

     Subtraction Result: res

     Multiplication Result: res

     Division Result: res

     Here res represents the result of the calculation from each of these methods.
   - Edit the services.xml file in the META-INF directory to change the reference to the project from DobService to CalculatorService. Also remove the operation named getYear and introduce the operation named add, subtract, multiply, and divide. Please note: the operation names should match the method names in the service.
   - Follow the release notes, similar to the DobService, publish the new calculator service to the tomcat server in the cloud instance.
   - Open the WSDL file for the CalculatorService and save the file as **CalculatorService.wsdl** in the submission/webservice directory.
2. **Task 4:**
   - Create a copy of dob-client folder in the web-service/client/linux(windows) directory and name the new folder as calc-client.

- Follow the release notes, similar to the DobClient, consume the new calculator service in the cloud instance. Edit the soaptest.py file by providing the correct reference to the WSDL file. Make changes to input two numbers in the soaptest.py file. Also, print the result by invoking all the four methods, add, sub, multiply, and divide. At this point, the calc-client in python should be fully functional. Capture a screenshot of the results from executing the client on your laptop. Name this file as pyclient and store the file in the submission/webservice directory.
- Follow the release notes, similar to the DobClient, consume the new calculator service in the cloud instance. Rename the DobDriver program in the Driver directory to CalculatorDriver. Edit the CalculatorDriver program to scan user-input for two numbers. Update the request to pass two arguments, that is by using setArgs0, and setArgs1. Remove the current response and instead create four responses to invoke the four methods, add, sub, multiply, and divide respectively. Print out the final result in the Driver program. At this point, the calc-client in Java should be fully functional. Capture a screenshot of the results from executing the client on your laptop. Name this file as javaclient and store the file in the submission/webservice directory.

Submission Details

For this assignment, please submit the following to your GitHub lab repository.

1. **video-reflection** markdown file.
2. upload of **CalculatorService.wsdl** file.
3. Copy of the **log-output file**.
4. upload of **aws-1** file.
5. upload of **aws-2** file.
6. upload of **aws-3** file.
7. upload of **aws-4** file.
8. upload of **javaclient** file.
9. upload of **pyclient** file.
10. It is recommended to upload a readme file, with the details that you would like the Professor to know while grading the work. For example, it may be reflection of your experience in the lab by highlighting some of the challenges faced and a brief mention of how you had addressed those challenges while implementing this lab. The readme file may also include a brief mention of any details that one should know about executing your program and what to expect during the execution.
11. It is highly important, for you to meet the honor code standards provided by the college and to ensure that the submission is made before the deadline. The honor code policy can be accessed through the course syllabus. Make sure to add the statement ”This work is mine unless otherwise cited.” in all your deliverables such as source code and PDF files.

Grading Rubric

1. Details including the points breakdown are provided in the individual sections above.
2. If a student needs any clarification on their lab credits, it is strongly recommended to talk to the Professor. The lab credits may be changed if deemed appropriate.

Wish you
all the best