Amazon Web Service APIs

Work with an AWS API/SDK and programmatically manage resources on AWS.

APIs and SDK access for AWS

The Amazon Web Services SDK includes different kinds of API packages which developers can use to create and manage resources and applications running on AWS. This allows developers to programmatically automate management tasks on AWS. Almost all of the functionality in AWS is accessible via CLI-tools, APIs or SDK. We will run down some of the most popular AWS tools now:

Amazon Command-Line Interface API Tools

For those of you who prefer to use bash scripts to run cloud jobs, the Amazon CLI tools allow for command-line manipulation of various facets of AWS. The following CLI-tools are available from AWS:

- EC2-API tools: The EC2 API tools allow you to create, manage and terminate EC2 instances.
- ELB-API tools: The ELB API tools allow you to create, manage and terminate Amazon Elastic Load Balancers.
- Amazon Cloudwatch Command-line tools: The command line tools for Cloudwatch allow you to monitor resources in AWS and set alarms.

You should find most the CLI API tools already installed in the AMIs that we provide for the project, if not, you can always download and install the CLI tools yourself.

Students who plan on using the CLI tools within bash/shell scripts should:

1. Download and install the CLI tools into thier instance
2. Add the path of the cli tools into **`$HOME/.bashrc`**
3. Setup their AWS credentials within the **`$HOME/.bashrc`** file, as detailed below.

Most of the above API tools require you to specify your AWS Credentials (Either you AWS Access Key ID/Secret Key or X.509 Certificate Keypairs). Please follow the instructions of each of the APIs to setup access and provide the required credentials in the AMI in the **`$HOME/.bashrc`** file. Look at the **`$HOME/.bashrc`** file in the course AMIs for more info.

AWS SDK for Java

The AWS SDK is available for multiple languages, including, Java, .NET, node.js, Ruby etc. The AWS SDK for Java has been developed by Amazon and is a fully featured SDK, which includes:

- **The AWS Java Library:** Java packages, classes and methods (i.e. an API) that allow you to program AWS using Java. The API hides much of the complexity that is otherwise involved in using a

REST/SOAP-based HTTP interface including authentication, request retries and error handling.

- **Code Samples:** Code samples that demonstrate the use of an API to perform various tasks on AWS.
- **Eclipse support:** Includes an Eclipse plugin that allows developers to create Java apps that work with AWS from within the Eclipse IDE.

Amazon EC2 API

You are already quite familiar with Amazon EC2 through the AWS Management Console. The AWS SDK exposes the EC2 management functionality accorded by the AWS Management Console and the command line tools via programming interfaces in Java. All of the EC2 related functionality is encapsulated in the **`com.amazonaws.services.ec2`** package. You can use the EC2 classes and methods to:

- Create EC2 Keypairs
- Create Security Groups for Instances and open ports (also known as *authorize security group ingress*)
- Create, launch, stop, reboot and terminate instances on EC2.

For example in the EC2 API, the following java code snippet will launch an instance:

Walkthrough

Launching an Instance using the AWS SDK for Java

```
//Load the Properties File with AWS Credentials
Properties properties = new Properties();
properties.load(EC2CWTest.class.getResourceAsStream("/AwsCredentials.properties"));

bawsc = new BasicAWSCredentials(properties.getProperty("accessKey"),
properties.getProperty("secretKey"));

//Create an Amazon EC2 Client
AmazonEC2Client ec2 = new AmazonEC2Client(bawsc);

//Create Instance Request
RunInstancesRequest runInstancesRequest = new RunInstancesRequest();

//Configure Instance Request
runInstancesRequest.withImageId("ami-3b44d352")
.withInstanceType("t1.micro")
.withMinCount(1)
.withMaxCount(1)
.withKeyName("project1_test")
.withSecurityGroups("MySecurityGroup");

//Launch Instance
RunInstancesResult runInstancesResult = ec2.runInstances(runInstancesRequest);

//Return the Object Reference of the Instance just Launched
Instance instance=runInstancesResult.getReservation().getInstances().get(0);
```

Walkthrough

Listing all running instances using the AWS SDK for Java

```
//Load the Properties File with AWS Credentials
Properties properties = new Properties();
properties.load(EC2CWTest.class.getResourceAsStream("/AwsCredentials.properties"));

bawsc = new BasicAWSCredentials(properties.getProperty("accessKey"),
properties.getProperty("secretKey"));

//Launch an EC2 Client
amazonEC2Client = new AmazonEC2Client(bawsc);

//Obtain a list of Reservations
List<Reservation> reservations =
amazonEC2Client.describeInstances().getReservations();
```

```
    int reservationCount = reservations.size();

    for(int i = 0; i < reservationCount; i++) {
        List<Instance> instances = reservations.get(i).getInstances();

        int instanceCount = instances.size();

        //Print the instance IDs of every instance in the reservation.
        for(int j = 0; j < instanceCount; j++) {
            Instance instance = instances.get(j);

            if(instance.getState().getName().equals("running")) {
                System.out.println(instance.getInstanceId());
            }
        }
    }
```

The snippets above assume that you have an **`AwsCredentials.properties`** file with your AWS keys and have already created an EC2 keypair and security group.

In addition, the Amazon EC2 API provides classes such as DescribeInstanceStatusRequest to make requests to check an instance's status:

- Running
- Pending
- Shutting Down, etc...

For more information on the Amazon EC2 API, please refer to the AWS SDK for Java Documentation and the AWS Java API Reference

Amazon CloudWatch

Amazon CloudWatch enables developers to monitor various facets of their AWS resources. Developers can use it to collect and track metrics from various AWS resources that are running on the AWS Cloud. Using APIs, CloudWatch also allows you to programmatically retrieve monitoring data, which can be used to track resources, spot trends and take automated action based on the state of your cloud resources on AWS. CloudWatch also allows you to set **alarms**, which constitute a set of instructions to be followed in case of an event that is tracked by CloudWatch.

CloudWatch can be used to monitor various types of AWS resources including:

- EC2 instances
- EBS volumes
- EMR job flows etc
- ELB Loads

For EC2 instances, CloudWatch allows you to monitor cpu, memory and disk utilization.

For more information on CloudWatch please refer to the documentation.

AWS SDK for Python (boto)

For python-inclined developers, AWS supports a third-party API called boto, which can be used to make API requests to AWS from within python. To install boto on the AMI, just follow the steps outlined in the Getting Started Guide.