Why AWS SES?

* High volume, low cost email delivery
* The ability to interface with an API
* Sending rate limitations to prevent getting flagged as spam
* Daily statistics for delivery, failure, bounce backs, and complaint tracking

We use them because setting up mail delivery infrastructure is a real headache at our size, and components of it are either difficult or impossible when hosting on Amazon Web Services.

* SES is only operated out of a single AWS region: us-east-1. Any service running in other AWS regions are reliant on us-east-1's availability for email delivery. Also, sending email into SES from other regions incurs additional outbound data transfer fees at the standard AWS rates.
* SES gives no way to programmatically scrape stats about traffic. They publish graphs on the product's AWS console page but these stats aren't available via the [AmazonCloudWatch](http://www.quora.com/Amazon-CloudWatch) API.

The following table lists the AWS regions in which Amazon SES is available, and the corresponding endpoints for the Amazon SES API and SMTP interface.

| **Region name** | **Region** | **API (HTTPS) endpoint** | **SMTP endpoint** |
| --- | --- | --- | --- |
| US East (N. Virginia) Region | us-east-1 | email.us-east-1.amazonaws.com | email-smtp.us-east-1.amazonaws.com |
| US West (Oregon) Region | us-west-2 | email.us-west-2.amazonaws.com | email-smtp.us-west-2.amazonaws.com |
| EU (Ireland) Region | eu-west-1 | email.eu-west-1.amazonaws.com | email-smtp.eu-west-1.amazonaws.com |

**With sandbox access :-**

* Emails can be sent only to the Amazon SES mailbox simulator and to verified email addresses or domains.
* Emails can be sent only from verified email addresses or domains.
* You can send a maximum of 200 messages per 24-hour period.
* You can send a maximum of one message per second.

Amazon SES has endpoints in multiple AWS regions. You must request production access separately for each region you want to use.

**AMAZON SES PRICING**

If you are an Amazon EC2 user, you can get started with Amazon SES for free. You can send 62,000 messages per month to any recipient when you call Amazon SES from an Amazon EC2 instance directly or through AWS Elastic Beanstalk.

Note: Data transfer and attachment fees still apply.

Email messages are charged at $0.10 per thousand. Emails to the [Amazon SES Mailbox Simulator](http://docs.aws.amazon.com/ses/latest/DeveloperGuide/mailbox-simulator.html) are billed separately from emails to other recipients.

\*A message is defined as a single email communication sent to a single email address. A single email communication sent to multiple recipients is considered to be a unique message to each recipient.

**FINDINGS**

Sending limits are based on recipients rather than on messages. For example, an email that has 10 recipients counts as 10 against your quota. However, we do not recommend that you send an email to multiple recipients in one call to SendEmail because if the call to Amazon SES fails (for example, the request is improperly formatted), the entire email will be rejected and none of the recipients will get the intended email. We recommend that you call SendEmail once for every recipient.

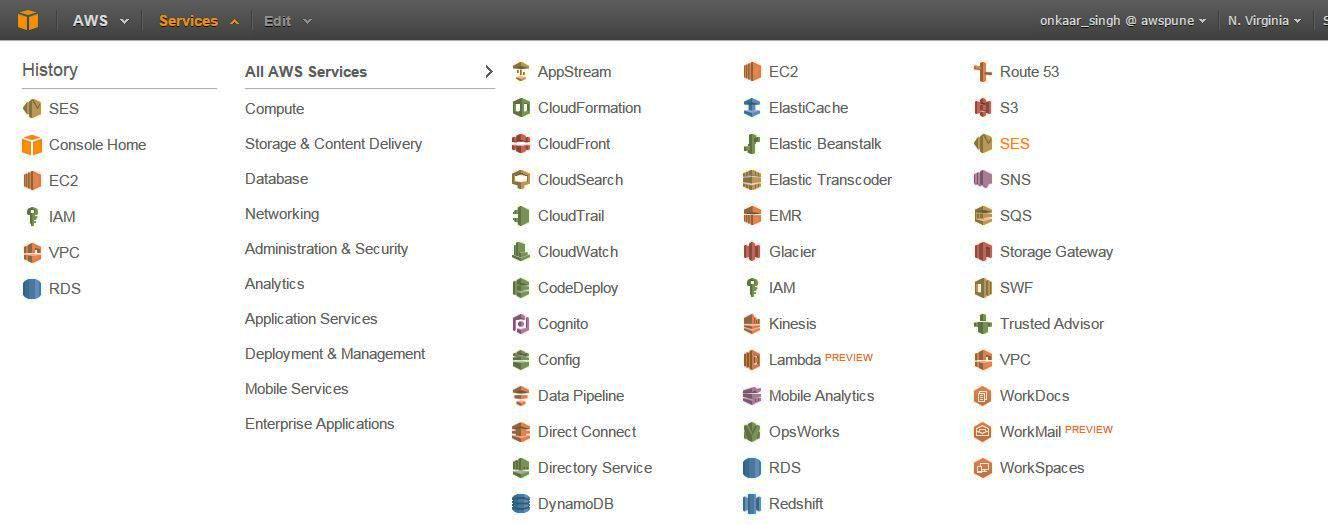
[Limits On SES](http://docs.aws.amazon.com/ses/latest/DeveloperGuide/limits.html)

**Two ways to use Amazon SES services:-**

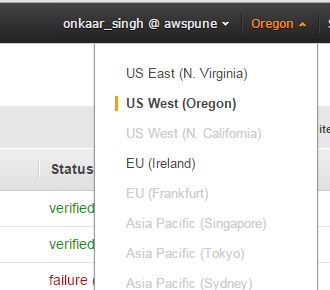
* 1. Using the SES java API.
  2. Using Amazon SES SMTP interface.

Steps to use SES services:-

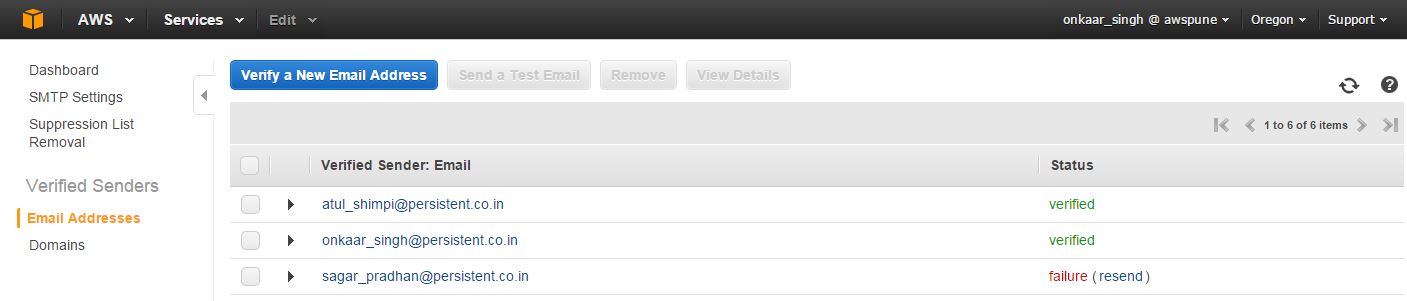
1. Login into your AWS account.
2. Select SES from services.



1. SES services is available in only one of the three regions



1. Verify the sender’s email address.



Now that the sender’s email has been verified we can use either of the above two methods to implement SES.

1. Using Amazon SES API to send email:

File Name:- AmazonSESSample.java

import java.io.IOException;

import com.amazonaws.ClientConfiguration;

import com.amazonaws.services.simpleemail.\*;

import com.amazonaws.services.simpleemail.model.\*;

import com.amazonaws.auth.AWSCredentials;

import com.amazonaws.auth.BasicAWSCredentials;

import com.amazonaws.regions.\*;

public class AmazonSESSample {

static final String FROM = "onkaar\_singh@persistent.co.in"; // Replace with your "From" address. This address must be verified.

static final String TO = "onkaar\_singh@persistent.co.in"; // Replace with a "To" address. If you have not yet requested

// production access, this address must be verified.

static final String BODY = "This email was sent through Amazon SES by using the AWS SDK for Java.";

static final String SUBJECT = "Amazon SES test (AWS SDK for Java)";

private static String accessKey = "AKIAIPSGWELRAGX527QQ";

private static String secretKey = "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";

static AWSCredentials credentials()

{

return new BasicAWSCredentials(accessKey, secretKey);

}

public static void main(String[] args) throws IOException {

final ClientConfiguration clientConfiguration;

final String PROXY\_HOST="ptproxy.persistent.co.in";

final int PROXY\_PORT=8080;

clientConfiguration = new ClientConfiguration();

clientConfiguration.setProxyHost(PROXY\_HOST);

clientConfiguration.setProxyPort(PROXY\_PORT);

AWSCredentials credential = AmazonSESSample.credentials();

// Construct an object to contain the recipient address.

Destination destination = new Destination().withToAddresses(new String[]{TO});

// Create the subject and body of the message.

Content subject = new Content().withData(SUBJECT);

Content textBody = new Content().withData(BODY);

Body body = new Body().withText(textBody);

// Create a message with the specified subject and body.

Message message = new Message().withSubject(subject).withBody(body);

// Assemble the email.

SendEmailRequest request = new SendEmailRequest().withSource(FROM).withDestination(destination).withMessage(message);

try

{

System.out.println("Attempting to send an email through Amazon SES by using the AWS SDK for Java...");

// Instantiate an Amazon SES client, which will make the service call. The service call requires your AWS credentials.

// Because we're not providing an argument when instantiating the client, the SDK will attempt to find your AWS credentials

// using the default credential provider chain. The first place the chain looks for the credentials is in environment variables

// AWS\_ACCESS\_KEY\_ID and AWS\_SECRET\_KEY.

// For more information, see http://docs.aws.amazon.com/AWSSdkDocsJava/latest/DeveloperGuide/credentials.html

AmazonSimpleEmailServiceClient client = new AmazonSimpleEmailServiceClient(credential,clientConfiguration);

System.out.println("Client created");

// Choose the AWS region of the Amazon SES endpoint you want to connect to. Note that your production

// access status, sending limits, and Amazon SES identity-related settings are specific to a given

// AWS region, so be sure to select an AWS region in which you set up Amazon SES. Here, we are using

// the US West (Oregon) region. Examples of other regions that Amazon SES supports are US\_EAST\_1

// and EU\_WEST\_1. For a complete list, see http://docs.aws.amazon.com/ses/latest/DeveloperGuide/regions.html

Region REGION = Region.getRegion(Regions.US\_WEST\_2);

System.out.println("Region set");

client.setRegion(REGION);

// Send the email.

client.sendEmail(request);

System.out.println("Email sent!");

}

catch (Exception ex)

{

System.out.println("The email was not sent.");

System.out.println("Error message: " + ex.getMessage());

}

}

}

1. Using Amazon SES SMTP interface to send emails

File Name:- AmazonSESSample.java

import java.util.Properties;

import javax.mail.\*;

import javax.mail.internet.\*;

public class AmazonSESSample {

static final String FROM = "onkaar\_singh@persistent.co.in"; // Replace with your "From" address. This address must be verified.

static final String TO = "onkaar\_singh@persistent.co.in"; // Replace with a "To" address. If you have not yet requested

// production access, this address must be verified.

static final String BODY = "Hi Onkaar, \n I am sending this email to my self. \n Thanks, \n Onkaar";

static final String SUBJECT = "Sending Email through Amazon SES smtp";

// Supply your SMTP credentials below. Note that your SMTP credentials are different from your AWS credentials.

static final String SMTP\_USERNAME = "AKIAIFSCQPTMRJD3747A"; // Replace with your SMTP username.

static final String SMTP\_PASSWORD = "AtJLISfWhFRhUT2NLqkZ2F7mT+irDlnFdX56Lq6YwntM"; // Replace with your SMTP password.

// Amazon SES SMTP host name. This example uses the us-west-2 region.

static final String HOST = "email-smtp.us-west-2.amazonaws.com";

// Port we will connect to on the Amazon SES SMTP endpoint. We are choosing port 25 because we will use

// STARTTLS to encrypt the connection.

static final int PORT = 25; //what about ssl port like 465 or 587 ?

public static void main(String[] args) throws Exception {

// Create a Properties object to contain connection configuration information.

Properties props = System.getProperties();

props.put("mail.transport.protocol", "smtp");

props.put("mail.smtp.port", PORT);

// Set properties indicating that we want to use STARTTLS to encrypt the connection.

// The SMTP session will begin on an unencrypted connection, and then the client

// will issue a STARTTLS command to upgrade to an encrypted connection.

props.put("mail.smtp.auth", "true");

props.put("mail.smtp.starttls.enable", "true");

props.put("mail.smtp.starttls.required", "true");

// Create a Session object to represent a mail session with the specified properties.

Session session = Session.getDefaultInstance(props);

// Create a message with the specified information.

MimeMessage msg = new MimeMessage(session);

msg.setFrom(new InternetAddress(FROM));

msg.setRecipient(Message.RecipientType.TO, new InternetAddress(TO));

msg.setSubject(SUBJECT);

msg.setContent(BODY,"text/plain");

// Create a transport.

Transport transport = session.getTransport();

// Send the message.

try

{

System.out.println("Attempting to send an email through the Amazon SES SMTP interface...");

// Connect to Amazon SES using the SMTP username and password you specified above.

transport.connect(HOST, SMTP\_USERNAME, SMTP\_PASSWORD);

System.out.println("Connecting...!");

// Send the email.

transport.sendMessage(msg, msg.getAllRecipients());

System.out.println("Email sent!");

}

catch (Exception ex) {

System.out.println("The email was not sent.");

System.out.println("Error message: " + ex.getMessage());

}

finally

{

// Close and terminate the connection.

transport.close();

}

}

}

To use the second method we need to create SMTP credentials which will be valid for all regions.

