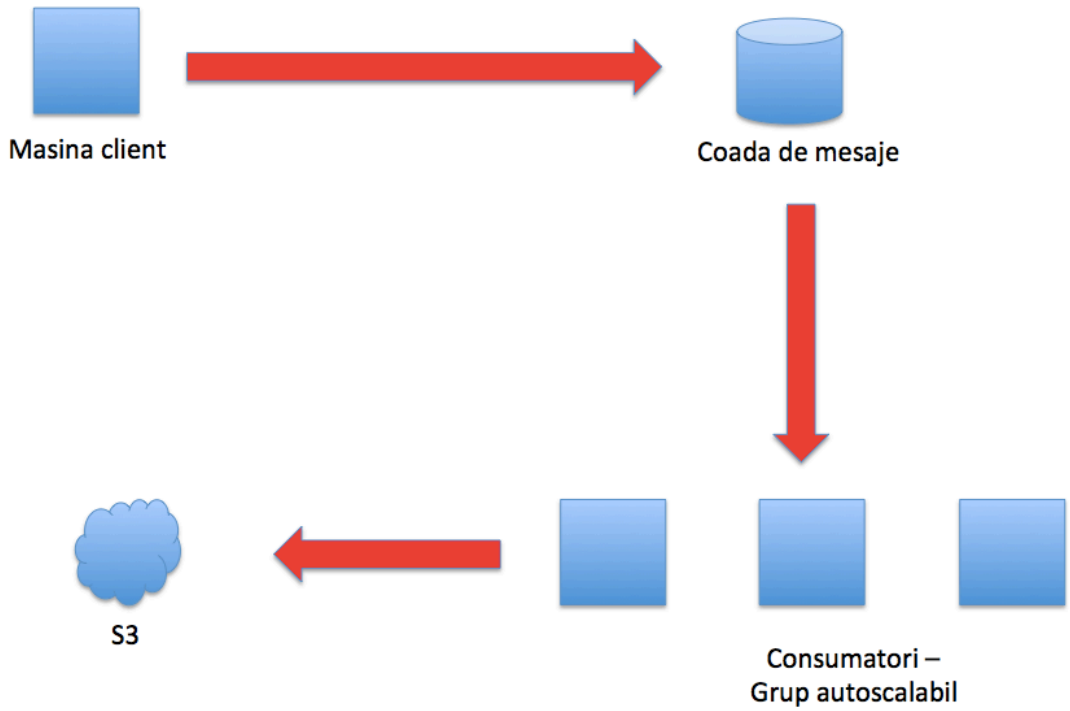
Introducere

Scopul este de a vedea cum se lucreaza cu o coada SQS, sa utilizati tool-ul din linia de comanda pentru a crea un grup de auto scalare si familiarizarea cu API-ul java oferit de amazon.

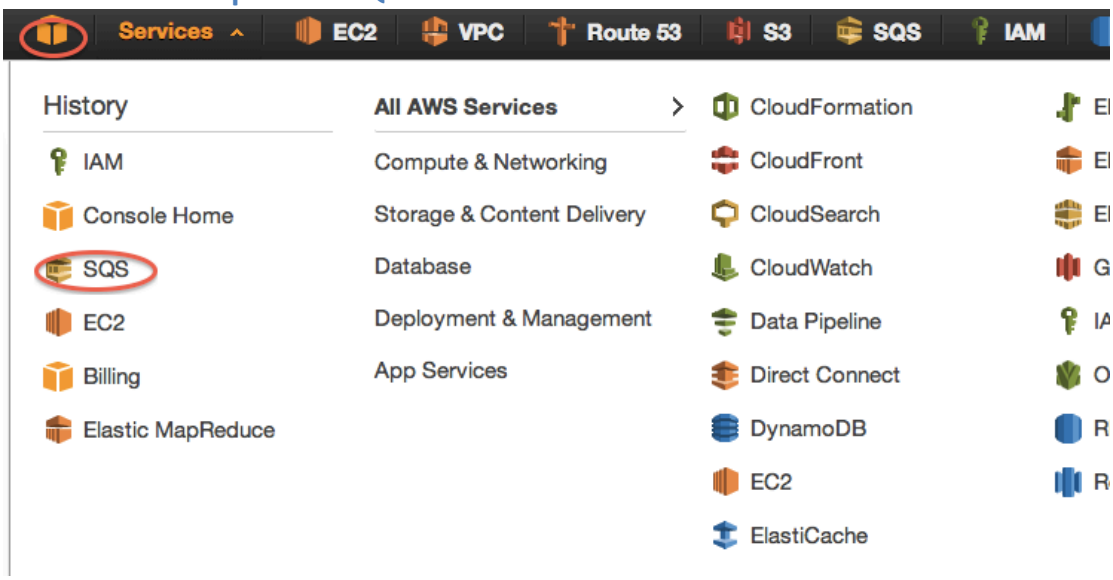
Pe parcursul laboratorului vom crea urmatoarea infrasctructura:



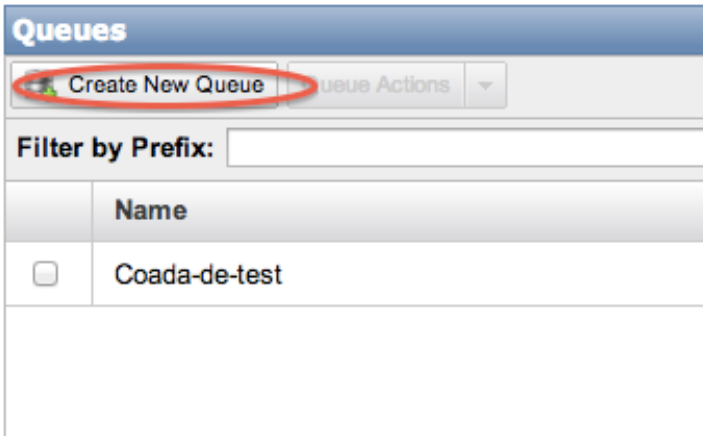
Partea1: Coada SQS

Vom crea o coada folosind consola web:
https://fmi-unibuc-bigdata.signin.aws.amazon.com/console

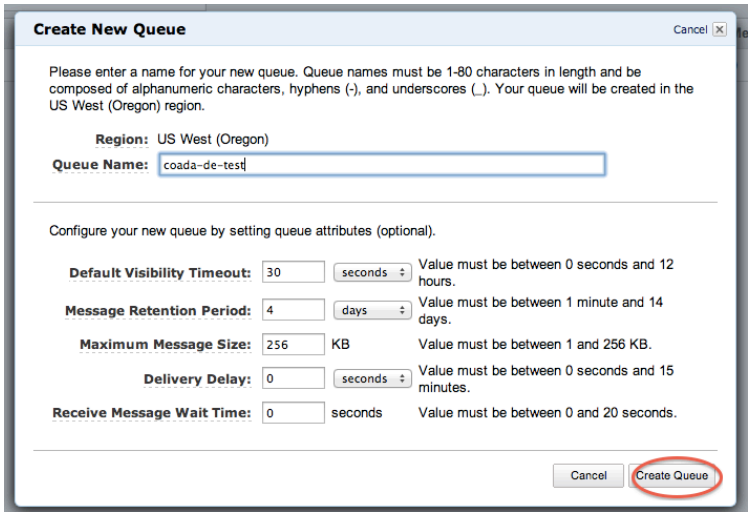
Pas1: Accesati aplicatia SQS



Pas2: Crearea unei cozi

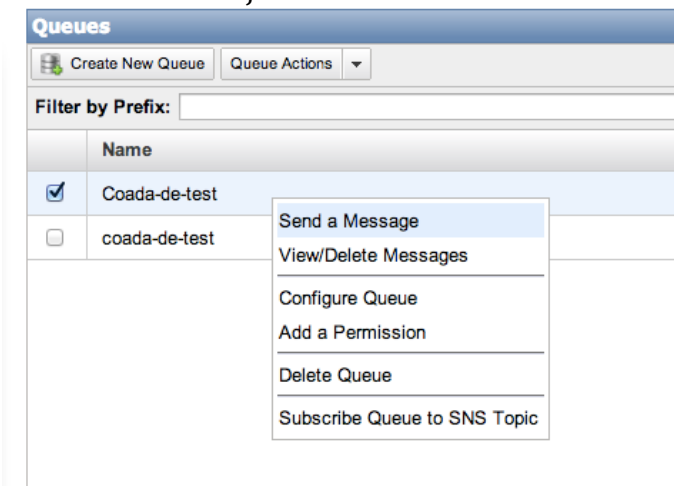


Veti completa formularul urmator (numele cozii):

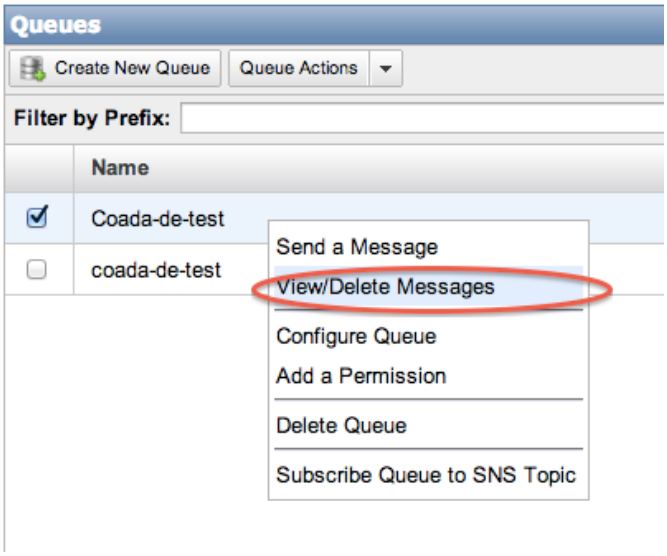


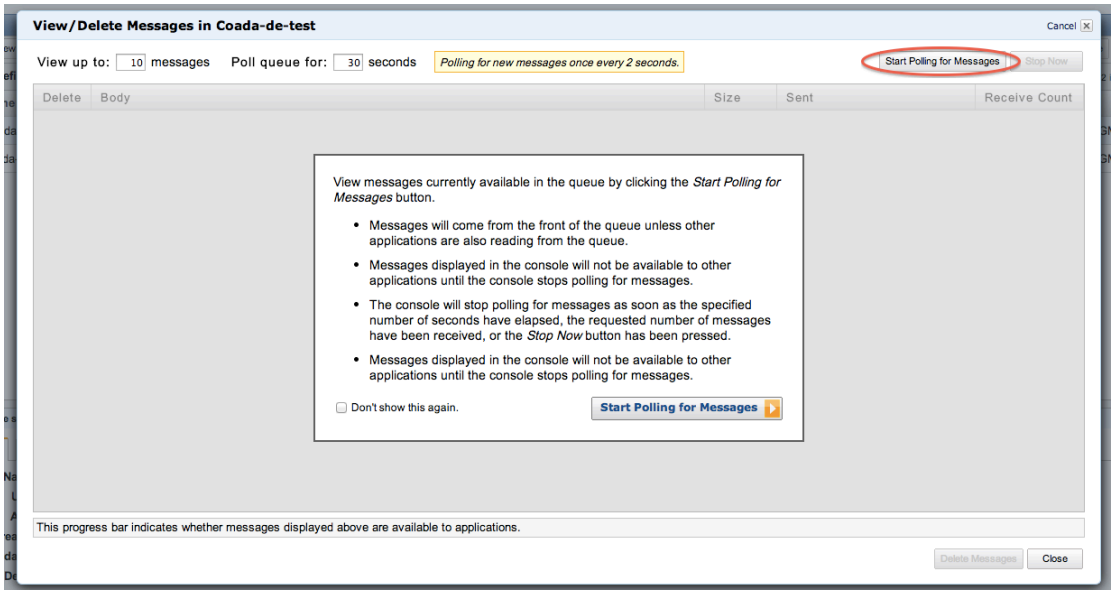
Pas3: Testarea cozii

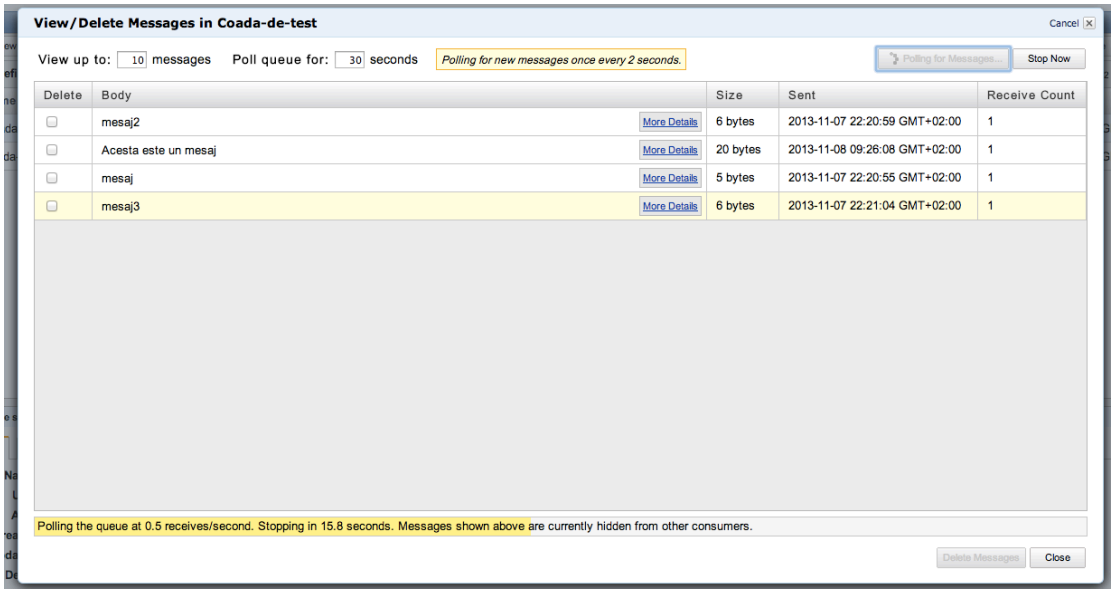
Trimiteti un mesaj in coada folosind consolva web.



Verificati existenta mesajelor din coada:







Partea 2: Folosirea librariei java pentru a trimite mesaje

Vom folosi librarie java si vom crea 2 aplicatii:

1. O aplicatie producator: care pune mesajele scrise la consola in coada
2. O aplicatie consumator: un loop care citeste si afiseaza mesajele.

Cele 2 aplicatii constituie un sistem rudimentar de chat.

Pas1: Descarcarea librariei

http://aws.amazon.com/sdkforjava/

Dupa dezarhivare libraria se afla in folderul bin/aws-java-sdk-1.6.4.jar

Pas 2: Aplicatia producator

Creati un nou proiect eclipse si includeti libraria descarcata. Scrieti un program care ruleaza un loop infinit in care citeste o linie de la tastatura si o adauga ca un mesaj in coada SQS.

Exemplu de cod (incomplet):

Autentificare

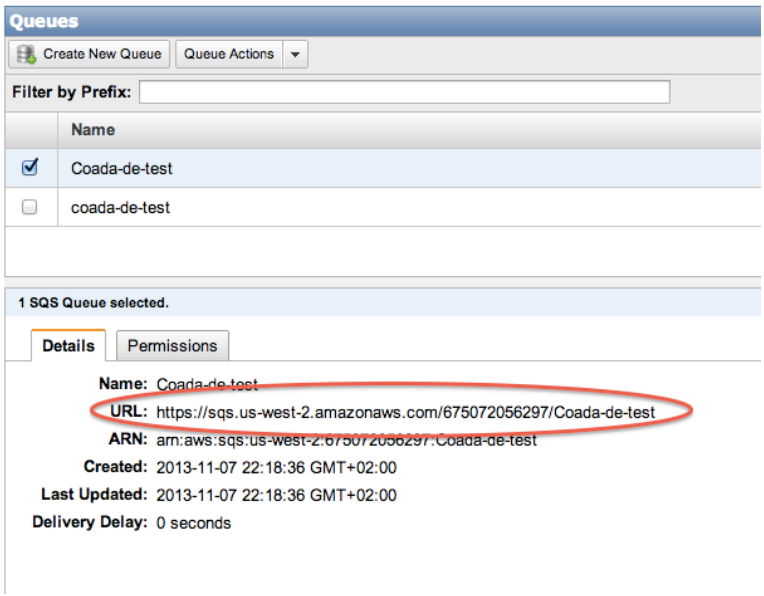
```
String accessKey = “abcdefgh…”;
String secretKey = “secret-hbdf6tsf6”;
AWSCredentials cred = new BasicAWSCredentials(
   accessKey, secretKey
);
```

Veti primi o lista cu cheile pentru fiecare utilizator.

Trimiterea unui mesaj

```
AmazonSQSClient sqs = new AmazonSQSClient(cred);
SendMessageRequest request = new  SendMessageRequest(“adresa-coada”, “mesaj1”);
SendMessageResult result = sqs.sendMessage(request);
```

Adresa cozii o gasiti in consola web:



Pas 3: Aplicatia consumator

Veti avea nevoie din nou de codul de autentificare iar citirea mesajelor din coada se face intr-un mod asemanator cu scrierea.
Gasiti documentatie si exemple la:
http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/sqs/AmazonSQSClient.html

Pas 4: Impachetarea aplicatiilor

Exportati aplicatiile ca “runnable jar”. Rulati-le in paralel (in 2 console diferite).
Verificati functionarea sistemului de chat.

Partea 3: Folosirea librariei pentru a accesa S3

Modificati aplicatia consumator sa scrie mesajele intr-un fisier din s3, si sa astepte 10 secunde dupa ce a scris fiecare mesaj.

Pentru a scrie in S3 folositi clasa AmazonS3Client:
AmazonS3Client s3 = new AmazonS3Client(cred);

Folositi aceeasi instanta de credentiale pe care o folosete si AmazonSQSClient;

Scrierea de mesaje in S3 se face asemanator cu trimiterea de mesaje in coada.
s3.putObject(new PutObjectRequest(“bucket-name”, “cheie”, continut)).
Gasiti documentatie si exemple la:
http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/s3/AmazonS3Client.html

Partea 4: Rularea aplicatiilor in cloud

Creati o masina EC2. Copiati si rulati pe ea pachetul cu aplicatia consumator creata la pasul anterior. Rulati producatorul local si scrieti mesaje. Verificati ca mesajele sunt salvate in s3. Deschideti inca o masina de consumator. Verificati ca ambele masini consuma mesaje din coada.

Partea 5: Autoscaling group pentru consumatori

Daca ai ajuns la partea 5, cheama-ma pentru detalii si explicatii.