Jaspreet Kaur
Take Home
100285603

Answer1:

Messaging: It involves passing messages around. It is different from emails and text messages. It is intended for communication between software components instead of people.

SNS: Simple Notification Service is a push notification service that allows you to send an individual message or just fan out messages to a number of recipients. It can be used to send a push notification to a mobile phone or email to the user or messages to other services.

SQS: Simple Queue Service is a message queuing service. It is used to transmit messages between different components.

We can use SNS and SQS together to send messages to the application that require immediate notification of an event and can also be persisted in Amazon SQS queue for other applications to process at the later time.

Queue and Topic:

The queue is like a buffer. You can put and receive messages from this queue. The process that puts the message is called the producer and the process that takes the message is called consumer.

The topic is like a broadcasting station. You can publish messages to a topic and any other service can subscribe to this topic and can access these messages.

How does it work?

- First, you need to create a new SNS topic.
- Then you have to subscribe SQS Queue(create a new queue or existing Standard or FIFO) to the SNS topic. Currently, SNS is not compatible with FIFO so you have to subscribe standard SQS queue to an SNS topic.
- Then one of your micro services can act as a producer that will publish messages to your SNS topic. For example:

```
console.log("Loading function");

var AWS = require("aws-sdk");
```

```
exports.handler = function(event, context) {
    var eventText = JSON.stringify(event, null, 2);
    console.log("Received event:", eventText);
    var sns = new AWS.SNS();
    var params = {
        Message: eventText,
        Subject: "Test SNS From Lambda",
        TopicArn: "arn:aws:sns:us-west-2:123456789012:test-topic1"
    };
    sns.publish(params, context.done);
};
```

- You have already subscribed your SQS queue to your SNS topic so the published message will be pushed to the queue.
- To read and process published messages, you need two micro services (Consumer and actual work performer). Consumer microservice can read as many messages as possible from SQS and executes Work performer for each message.
- Once the task is finished successfully, the message will be removed from the queue. If there is any failure than the message will stay in SQS queue and will be handled by another available work performer microservice.

Note: Along with SQS queue, there will be a dead letter queue that will collect all the failed tasks. You can configure the failure time according to your application.

Answer2:

Following are the two alternative frameworks to the Serverless framework:

1) Squeezer Framework:

It is a framework designed to help developers to get a better architecture on serverless zero-administration compute platforms where the code runs on the top of microservices clouds like AWS Lambda, Azure Functions, IBM OpenWhisk & Google Functions.

It offers following things:

- Pay as you go: You can deploy your project for any time frame without worrying about the monthly cost.
- Intuitive Deployments: You can just deploy newly added features instead of deploying the whole project again and again.
- Smart Testing: You can just test the newly added features.
- Event Driven Ecosystem: You can add as many events you want to trigger your microservice's functions.
- Monolithic as Modular: You can share components between functions like you do with your traditional projects.

The difference from Serverless:

In Squeezer, we have to compile functions before deploying it on the cloud by using the following command:

```
sqz compile --cloud
```

Also, the commands are a little bit different from serverless. For example, if you want to invoke function locally and want to send string input then you have to use this command:

```
sqz invoke -f MyFunction -j '{"a":"b"}'
```

2) Zappa:

Zappa allows us to build and deploy serverless, event-driven Python applications on AWS Lambda plus API Gateway. It is not fully serverless. There is still a server but it has 40 ms lifecycle.

It offers:

- It is easy to use.
- In Zappa, each request is given its own HTTP server by Amazo API Gateway.
- It is cheaper than other deployment servers like Heroku.
- You can deploy serverless microservices with frameworks like Flask and Bottle, and for hosting larger web apps and CMSes with Django.

- It allows you to have trillion of events in your application.
- It also provides free SSL certificates, global app deployment, API access management, automatic security policy generation, precompiled C-extensions, auto keep-warms, oversized Lambda packages.

The difference from Serverless:

- It only supports Python language.
- It provides features like custom domain name with auto-renewing TLS certificates, keeping your lambda warm automatically, reading variables from S3, managing API key to secure endpoints.
- Zappa provides active community support. They have an active slack channel and a high number of stars on GitHub.