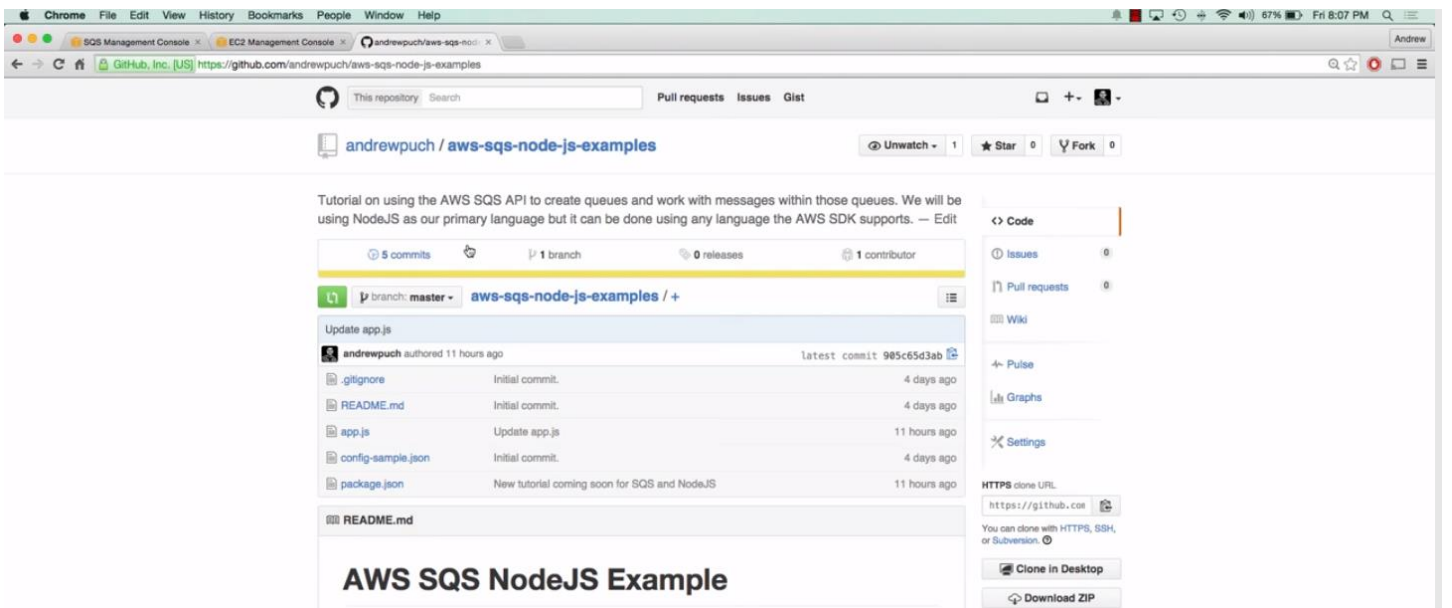
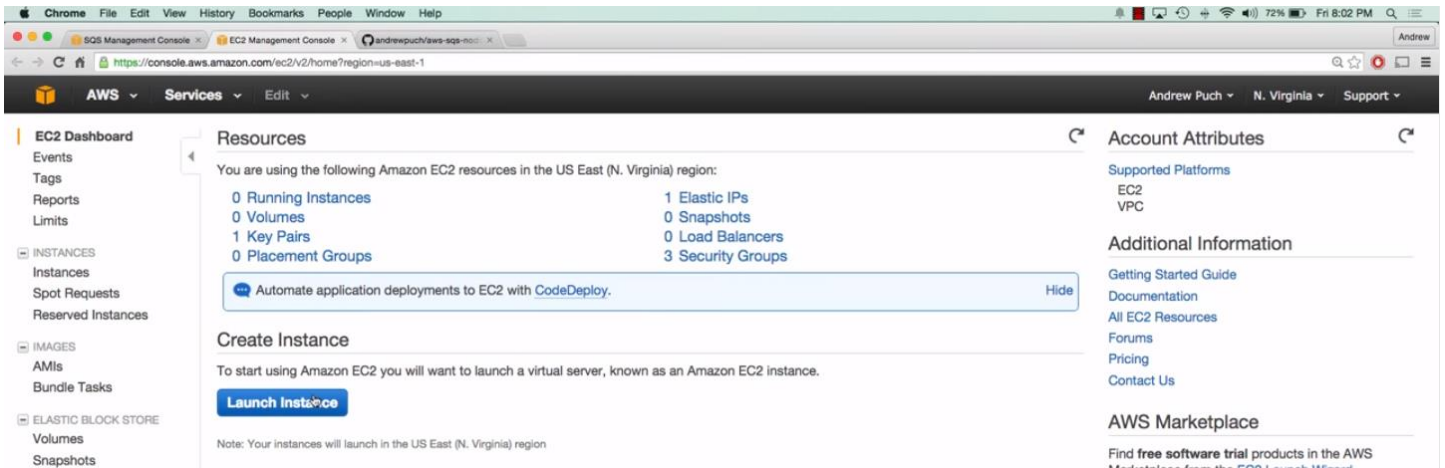
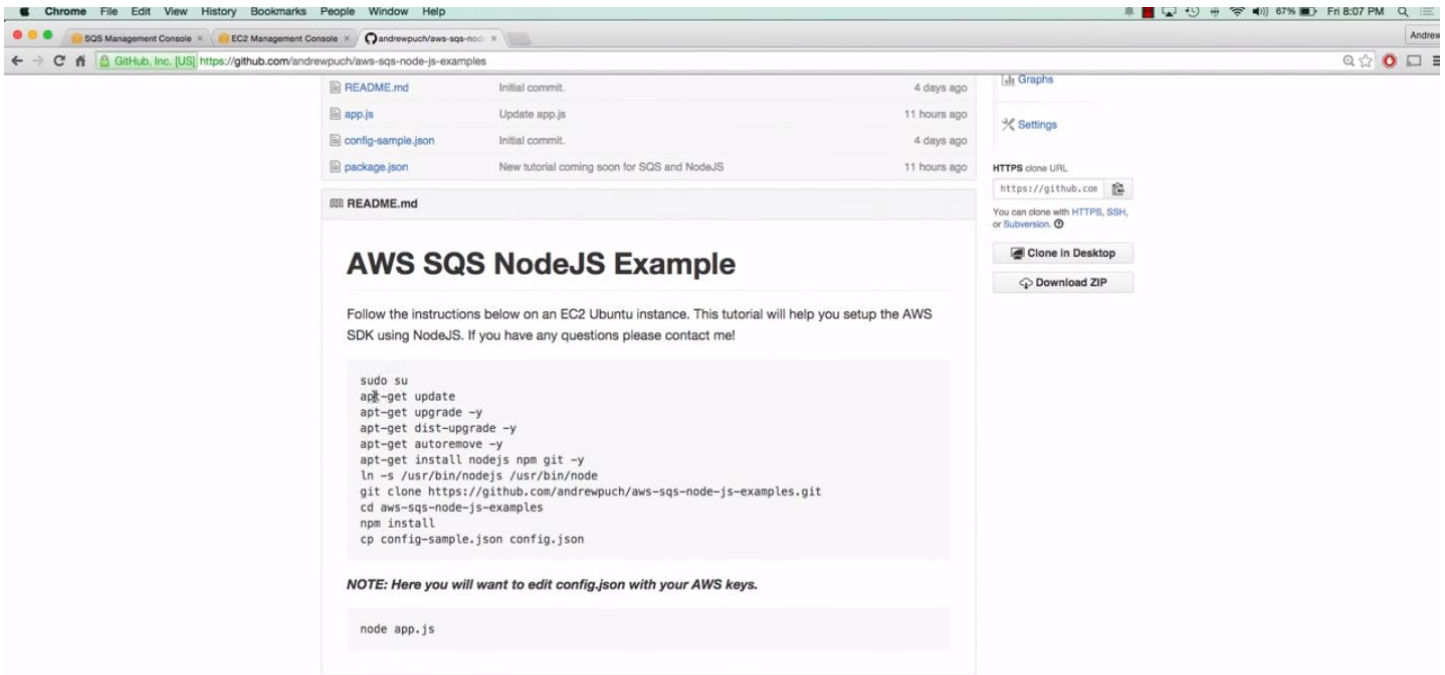


We will see a sample NodeJS service that creates a SQS queue, send messages to the queue, and process those messages.





Let us spin up an EC2 server that we will run our NodeJS app on.

1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

T2 instances are VPC-only. Your T2 instance will launch into your VPC. [Learn more](#) about T2 and VPC.

| | Family | Type | vCPUs | Memory (GiB) | Instance Storage (GB) | EBS-Optimized Available | Network Performance |
|-------------------------------------|-----------------|--------------------------------|-------|--------------|-----------------------|-------------------------|---------------------|
| <input checked="" type="checkbox"/> | General purpose | t2.micro Free tier eligible | 1 | 1 | EBS only | - | Low to Moderate |
| <input type="checkbox"/> | General purpose | t2.small | 1 | 2 | EBS only | - | Low to Moderate |
| <input type="checkbox"/> | General purpose | t2.medium | 2 | 4 | EBS only | - | Low to Moderate |
| <input type="checkbox"/> | General purpose | t2.large | 2 | 8 | EBS only | - | Low to Moderate |
| <input type="checkbox"/> | General purpose | m4.large | 2 | 8 | EBS only | Yes | Moderate |
| <input type="checkbox"/> | General purpose | m4.xlarge | 4 | 16 | EBS only | Yes | High |
| <input type="checkbox"/> | General purpose | m4.2xlarge | 8 | 32 | EBS only | Yes | High |

Cancel

Previous

Review and Launch

Next: Configure Instance Details

1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances1

Purchasing option☐ Request Spot Instances

Networkvpc-89af74ec (172.30.0.0/16) Create new VPC

Subnetsubnet-b9151091(172.30.0.0/24) | public-1a | us- Create new subnet
251 IP Addresses available

Auto-assign Public IP☒ Use subnet setting (Enable)

IAM roleNone Create new IAM role

Shutdown behaviorStop

Enable termination protection☐ Protect against accidental termination

Monitoring☐ Enable CloudWatch detailed monitoring
Additional charges apply.

Tenancy☒ Shared tenancy (multi-tenant hardware)
Additional charges will apply for dedicated tenancy.

Network interfaces

| Device | Network Interface | Subnet | Primary IP | Secondary IP addresses |
|--------|-------------------|--------|------------|------------------------|
|--------|-------------------|--------|------------|------------------------|

Cancel

Previous

Review and Launch

Next: Add Storage

ChromeFileEditViewHistoryBookmarksPeopleWindowHelp

SQS Management ConsoleEC2 Management Consoleandrewpuch/aws-sqs-ro...

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

Andrew PuchN. VirginiaSupport

1. Choose AMI2. Choose Instance Type3. Configure Instance4. Add Storage5. Tag Instance6. Configure Security Group7. Review

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

| Type | Device | Snapshot | Size (GiB) | Volume Type | IOPS | Delete on Termination | Encrypted |
|------|-----------|---------------|------------|-----------------------|-----------|-------------------------------------|---------------|
| Root | /dev/sda1 | snap-4e3c6d2b | 8 | General Purpose (SSD) | 24 / 3000 | <input checked="" type="checkbox"/> | Not Encrypted |

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

ChromeFileEditViewHistoryBookmarksPeopleWindowHelp

SQS Management ConsoleEC2 Management Consoleandrewpuch/aws-sqs-ro...

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

Andrew PuchN. VirginiaSupport

1. Choose AMI2. Choose Instance Type3. Configure Instance4. Add Storage5. Tag Instance6. Configure Security Group7. Review

Step 5: Tag Instance

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon EC2 resources.

Key (127 characters maximum)

Value (255 characters maximum)

Name

AWS SQS Example

Create Tag (Up to 10 tags maximum)

ChromeFileEditViewHistoryBookmarksPeopleWindowHelp

SQS Management ConsoleEC2 Management Consoleandrewpuch/aws-sqs-ro...

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

Andrew PuchN. VirginiaSupport

1. Choose AMI2. Choose Instance Type3. Configure Instance4. Add Storage5. Tag Instance6. Configure Security Group7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:

- Create a new security group
- Select an existing security group

Filter VPC security groups

| Security Group ID | Name | Description | Actions |
|-------------------------------------------------|-----------|----------------------------|-----------------------------|
| <input type="checkbox"/> sg-5e6f1d3b | default | default VPC security group | Copy to new |
| <input checked="" type="checkbox"/> sg-d8bd5ebc | Wide Open | Wide Open | Copy to new |

Inbound rules for sg-d8bd5ebc (Selected security groups: sg-d8bd5ebc)

| Type | Protocol | Port Range | Source |
|-------------|----------|------------|-----------|
| All traffic | All | All | 0.0.0.0/0 |

Cancel

Previous

Review and Launch

ChromeFileEditViewHistoryBookmarksPeopleWindowHelp

SGS Management ConsoleEC2 Management Consoleandrewpuch/aws-sgs-rol

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

Andrew PuchN. VirginiaSupport

1. Choose AMI2. Choose Instance Type3. Configure Instance4. Add Storage5. Tag Instance6. Configure Security Group7. Review

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

⚠️

Improve your instances' security. Your security group, **Wide Open**, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only. You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

AMI Details

Free tier eligible

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-d05e75b8

Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).
Root Device Type: ebs Virtualization type: hvm

Instance Type

ECUs

vCPUs

Memory (GiB)

Instance Storage (GB)

EBS-Optimized Available

Network Performance

| | | | | | | |
|----------|----------|---|---|----------|---|-----------------|
| t2.micro | Variable | 1 | 1 | EBS only | - | Low to Moderate |
|----------|----------|---|---|----------|---|-----------------|

Security Groups

Security Group ID

Name

Description

| | | |
|-------------|-----------|-----------|
| sg-d8bd5ebc | Wide Open | Wide Open |
|-------------|-----------|-----------|

All selected security groups inbound rules

| Security Group ID | Type | Protocol | Port Range | Source |
|-------------------|------|----------|------------|--------|
|-------------------|------|----------|------------|--------|

CancelPreviousLaunch

ChromeFileEditViewHistoryBookmarksPeopleWindowHelp

SGS Management ConsoleEC2 Management Consoleandrewpuch/aws-sgs-rol

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

Andrew PuchN. VirginiaSupport

1. Choose AMI2. Choose Instance Type3. Configure Instance4. Add Storage5. Tag Instance6. Configure Security Group7. Review

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

⚠️

Improve your instances' security. Your security group, **Wide Open**, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only. You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

AMI Details

Free tier eligible

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-d05e75b8

Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).
Root Device Type: ebs Virtualization type: hvm

Instance Type

ECUs

vCPUs

Memory (GiB)

Instance Storage (GB)

EBS-Optimized Available

Network Performance

| | | | | | | |
|----------|----------|---|---|----------|---|-----------------|
| t2.micro | Variable | 1 | 1 | EBS only | - | Low to Moderate |
|----------|----------|---|---|----------|---|-----------------|

Security Groups

Security Group ID

Name

Description

| | | |
|-------------|-----------|-----------|
| sg-d8bd5ebc | Wide Open | Wide Open |
|-------------|-----------|-----------|

All selected security groups inbound rules

| Security Group ID | Type | Protocol | Port Range | Source |
|-------------------|------|----------|------------|--------|
|-------------------|------|----------|------------|--------|

CancelPreviousLaunch

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair

Select a key pair

awstutorialseries

☒ I acknowledge that I have access to the selected private key file (awstutorialseries.pem), and that without this file, I won't be able to log into my instance.

CancelLaunch Instances

Chrome File Edit View History Bookmarks People Window Help

SQS Management Console EC2 Management Console andrewpuch/aw-sqs-ec2

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#instances:

AWS Services Edit

Andrew Puch N. Virginia Support

EC2 Dashboard Events Tags Reports Limits

INSTANCES

Instances

Spot Requests

Reserved Instances

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

NETWORK & SECURITY

Security Groups

Elastic IPs

Placement Groups

Load Balancers

Key Pairs

Network Interfaces

AUTO SCALING

Launch Configurations

Auto Scaling Groups

Launch Instance Connect Actions

Filter by tags and attributes or search by keyword

1 to 1 of 1

| Name | Instance ID | Instance Type | Availability Zone | Instance State | Status Checks | Alarm Status | Public DNS | Public IP | Key Name |
|--------------|-------------|---------------|-------------------|----------------|----------------|--------------|--------------------------|----------------|-------------------|
| AWS SQS E... | i-555f60fc | t2.micro | us-east-1a | running | 2/2 checks ... | None | ec2-54-208-245-163.co... | 54.208.245.163 | awstutorialseries |

Instance: i-555f60fc (AWS SQS Example) Public DNS: ec2-54-208-245-163.compute-1.amazonaws.com

Description Status Checks Monitoring Tags

| | | | |
|-----------------------|------------------------------|-------------------|----------------------------------------------------------|
| Instance ID | i-555f60fc | Public DNS | ec2-54-208-245-163.compute-1.amazonaws.com |
| Instance state | running | Public IP | 54.208.245.163 |
| Instance type | t2.micro | Elastic IP | - |
| Private DNS | ip-172-30-0-187.ec2.internal | Availability zone | us-east-1a |
| Private IPs | 172.30.0.187 | Security groups | Wide Open. view rules |
| Secondary private IPs | - | Scheduled events | No scheduled events |
| VPC ID | vpc-89af74ec | AMI ID | ubuntu-trusty-14.04-amd64-server-20150325 (ami-d05e75b8) |
| Subnet ID | subnet-b9151091 | Platform | - |
| Network interfaces | eth0 | IAM role | - |
| Source/dest. check | True | Key pair name | awstutorialseries |

```

Andrews-MacBook-Air:awstutorialseries andrewpuch$ ssh -i awstutorialseries.pem ubuntu@54.208.245.163
The authenticity of host '54.208.245.163 (54.208.245.163)' can't be established.
RSA key fingerprint is 00:9e:95:c4:8f:26:db:78:92:c9:d9:b3:cc:d3:89:71.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.208.245.163' (RSA) to the list of known hosts.
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-48-generic x86_64)

* Documentation:  https://help.ubuntu.com/

System information as of Fri Jun 26 17:03:42 UTC 2015

System load: 0.08      Memory usage: 5%      Processes:      83
Usage of /:  9.8% of 7.74GB  Swap usage:  0%      Users logged in: 0

Graph this data and manage this system at:
https://landscape.canonical.com/

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

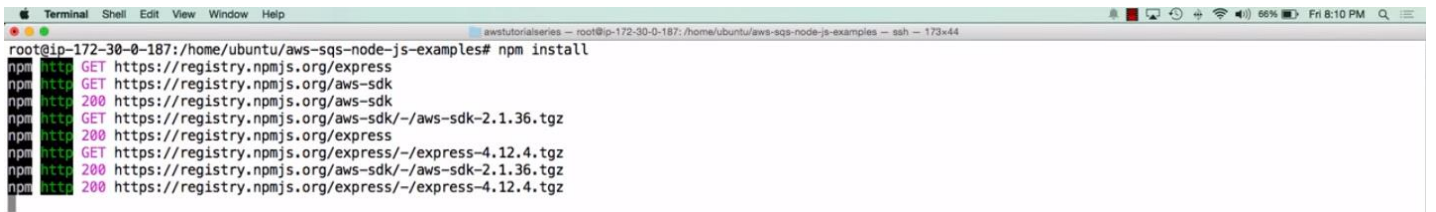
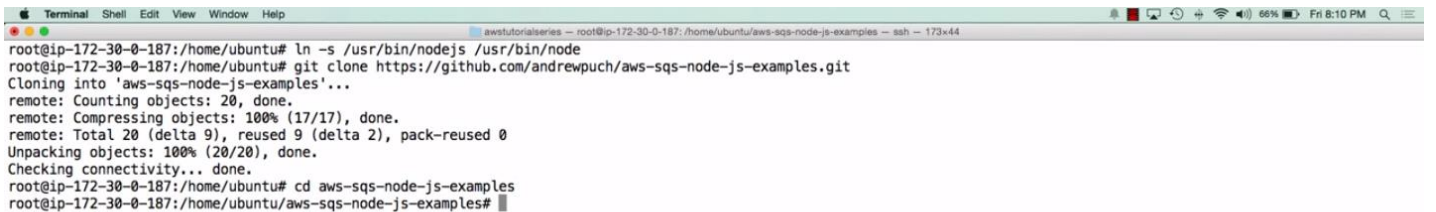
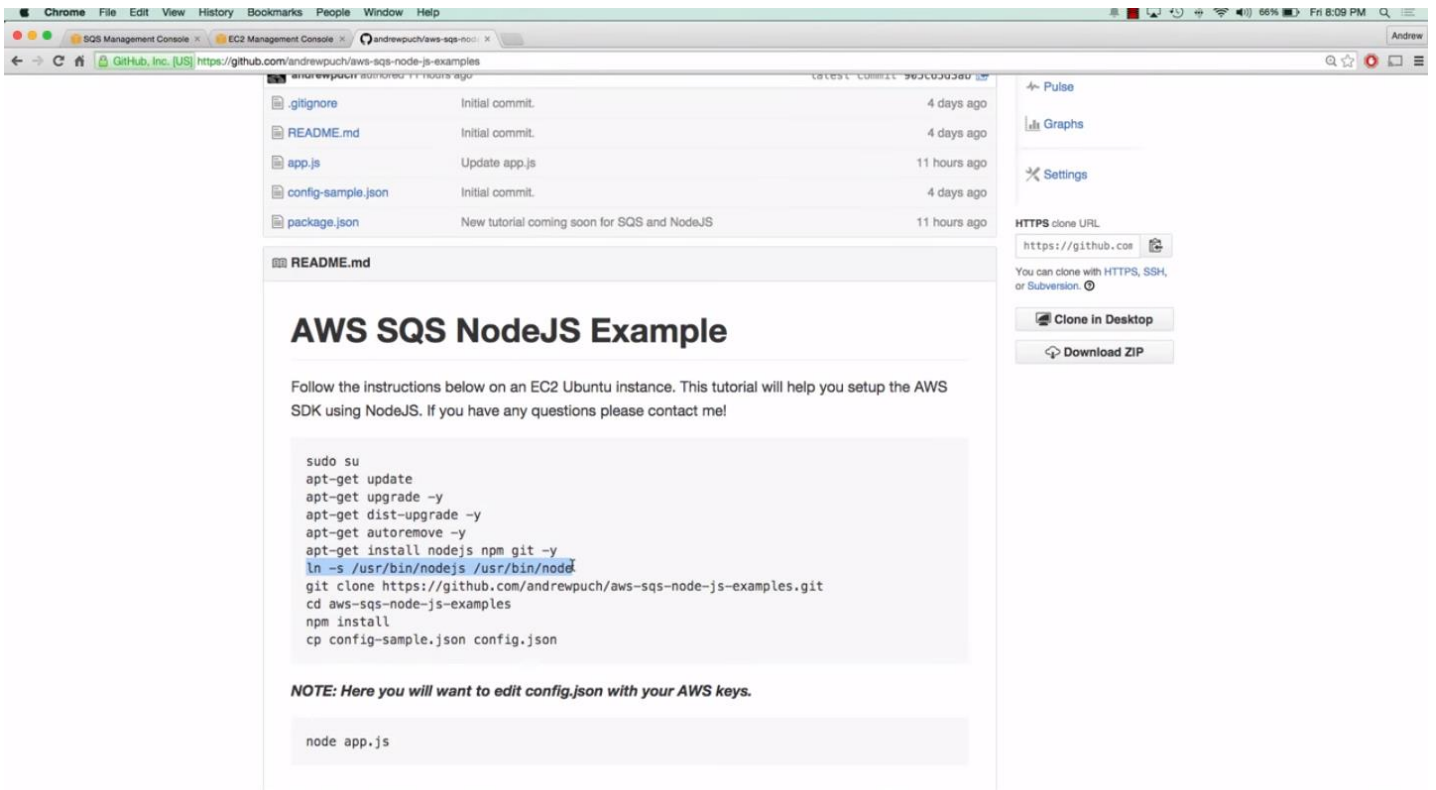
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

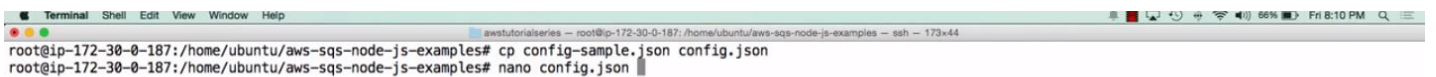
ubuntu@ip-172-30-0-187:~$

```

We SSH into the EC2 server and follow all the commands shown below



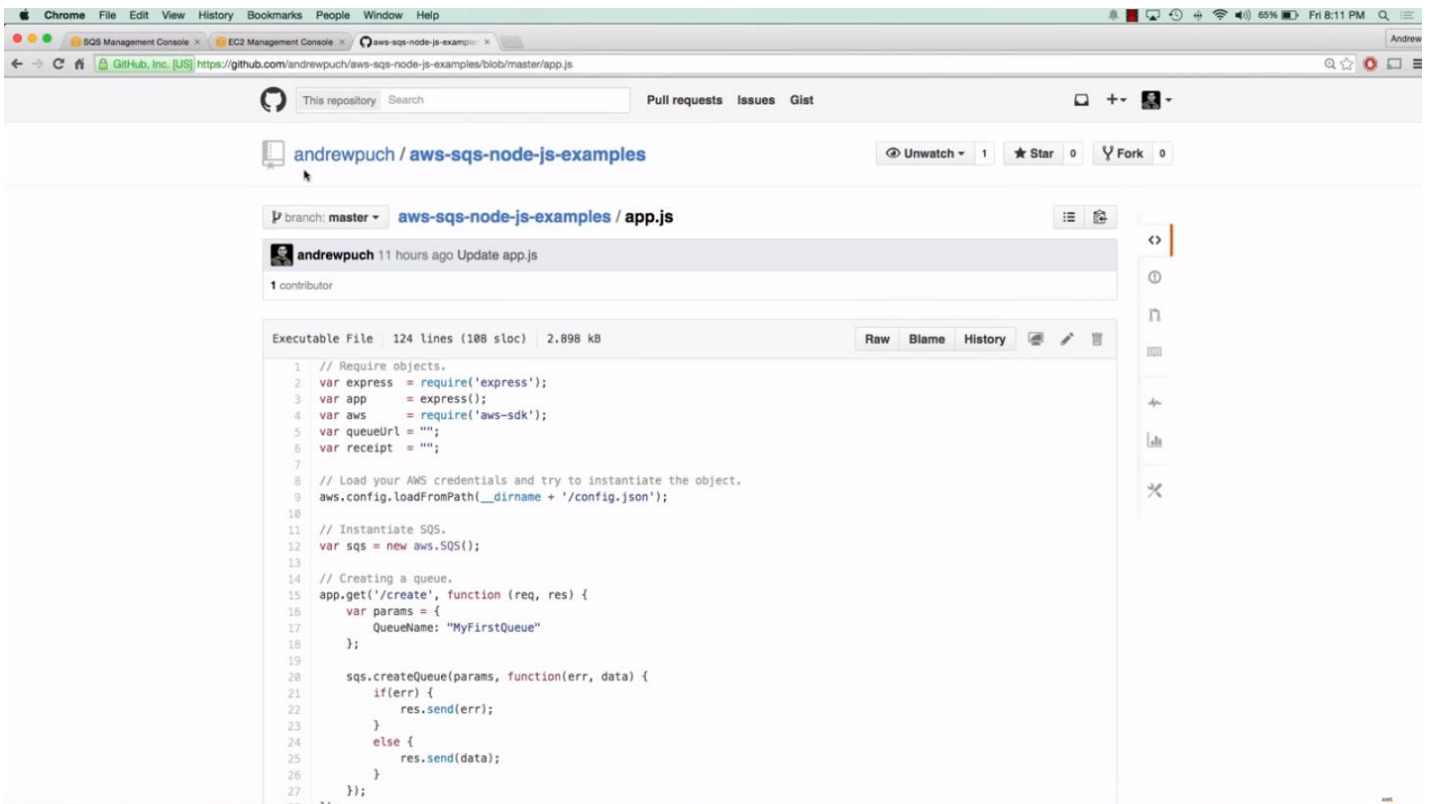
We then run the **npm install** command to install all the dependencies and the AWS SDK.



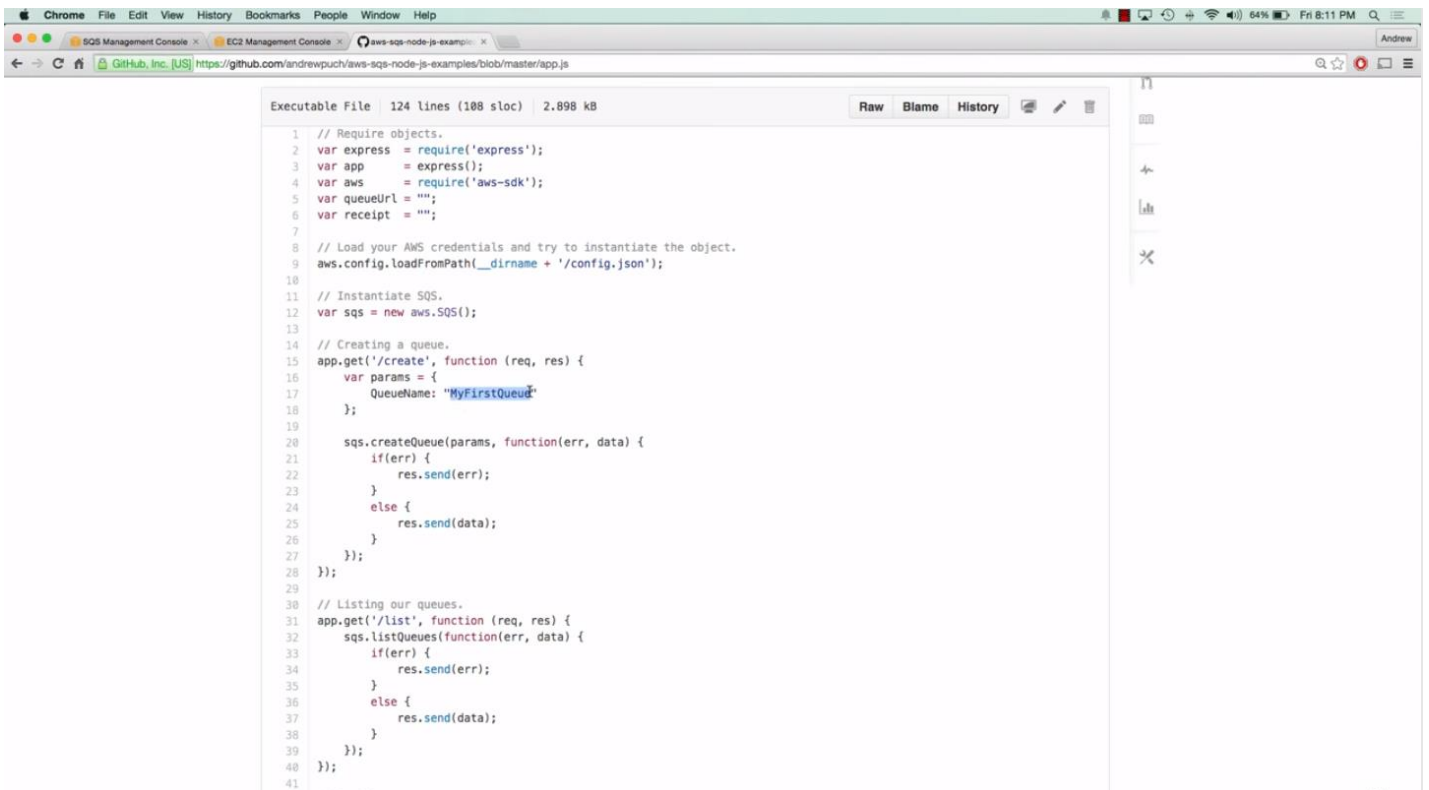
Copy the config-sample file provided into your config.json file and you need to put in your AWS access and secret keys in the file as above.



We now start the NodeJS app



```
1 // Require objects.
2 var express = require('express');
3 var app = express();
4 var aws = require('aws-sdk');
5 var queueUrl = "";
6 var receipt = "";
7
8 // Load your AWS credentials and try to instantiate the object.
9 aws.config.loadFromPath(__dirname + '/config.json');
10
11 // Instantiate SQS.
12 var sqs = new aws.SQS();
13
14 // Creating a queue.
15 app.get('/create', function (req, res) {
16   var params = {
17     QueueName: "MyFirstQueue"
18   };
19
20   sqs.createQueue(params, function(err, data) {
21     if(err) {
22       res.send(err);
23     }
24     else {
25       res.send(data);
26     }
27   });
28 });
```

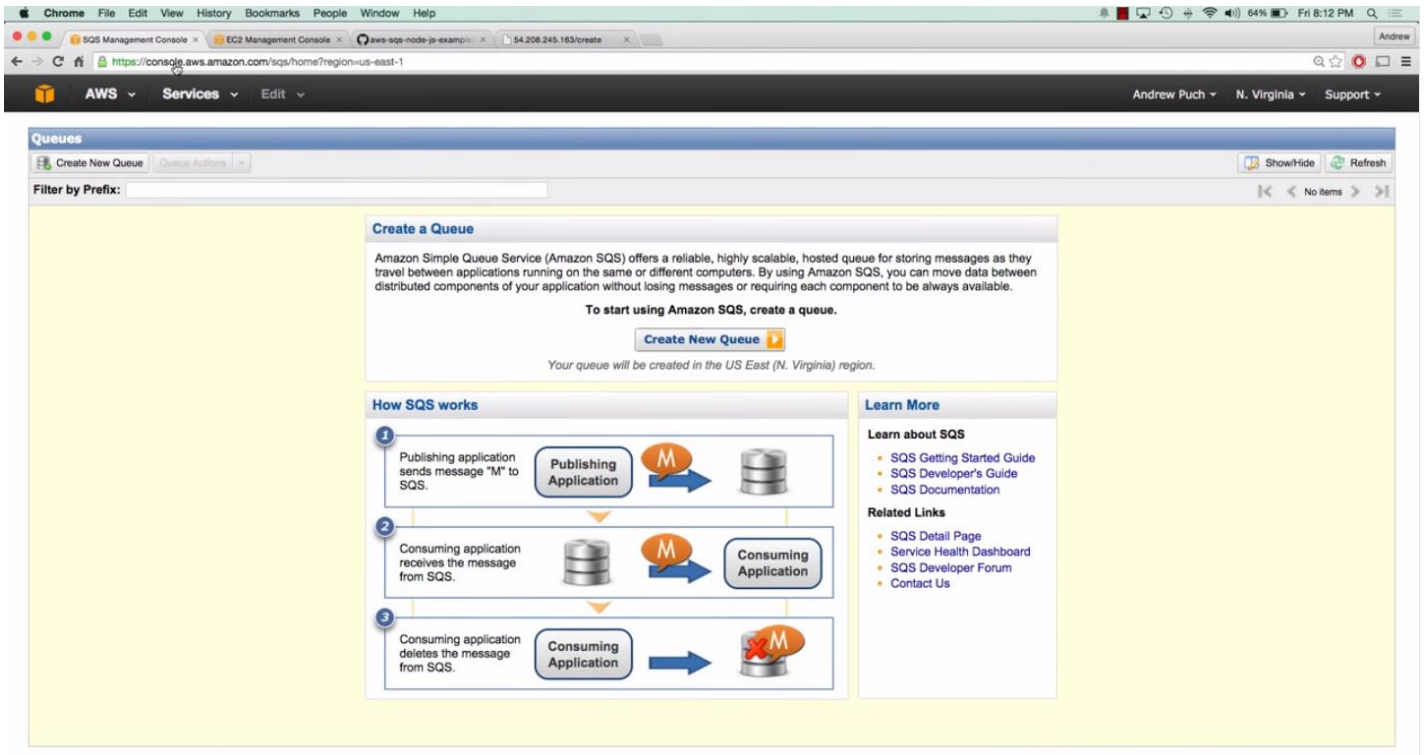


```
1 // Require objects.
2 var express = require('express');
3 var app = express();
4 var aws = require('aws-sdk');
5 var queueUrl = "";
6 var receipt = "";
7
8 // Load your AWS credentials and try to instantiate the object.
9 aws.config.loadFromPath(__dirname + '/config.json');
10
11 // Instantiate SQS.
12 var sqs = new aws.SQS();
13
14 // Creating a queue.
15 app.get('/create', function (req, res) {
16   var params = {
17     QueueName: "MyFirstQueue"
18   };
19
20   sqs.createQueue(params, function(err, data) {
21     if(err) {
22       res.send(err);
23     }
24     else {
25       res.send(data);
26     }
27   });
28 });
29
30 // Listing our queues.
31 app.get('/list', function (req, res) {
32   sqs.listQueues(function(err, data) {
33     if(err) {
34       res.send(err);
35     }
36     else {
37       res.send(data);
38     }
39   });
40 });
41
```

The **/create** endpoint is going to create a SQS queue called MyFirstQueue,


```
Chrome File Edit View History Bookmarks People Window Help
SQS Management Console x EC2 Management Console x aws-sqs-node-js-examp... x 54.208.245.163/create x
54.208.245.163/create
{
  "ResponseMetadata": {
    "RequestId": "5a8c22bf-b8df-5087-99e6-6cb832c17268"
  },
  "QueueUrl": "https://sqs.us-east-1.amazonaws.com/52346937018/MyFirstQueue"
}
```

We got a response back when we call the **/create** endpoint and we now have a SQS queue URL returned for us to us



We can now refresh our SQS dashboard in AWS to see our queue listed

The screenshot shows the AWS Management Console for the SQS service, specifically the 'Queues' page. The page has a 'Filter by Prefix' input field and a 'Refresh' button. Below the input field, there is a table with the following data:

| Name | Messages Available | Messages In Flight | Created |
|--------------|--------------------|--------------------|-------------------------------|
| MyFirstQueue | 0 | 0 | 2015-06-26 13:11:58 GMT-04:00 |

This dashboard will let you know the messages that are available in your queue, messages in flight

Chrome File Edit View History Bookmarks People Window Help

SQS Management Console x EC2 Management Console x aws-sqs-node-js-examp... x 54.208.245.163/create x

https://console.aws.amazon.com/sqs/home?region=us-east-1#queue-browser:prefix=

AWS Services Edit Andrew Puch N. Virginia Support

Queues

Create New Queue Queue Actions Show/Hide Refresh

Filter by Prefix:

| Name | Messages Available | Messages in Flight | Created |
|--------------------------------------------------|--------------------|--------------------|-------------------------------|
| <input checked="" type="checkbox"/> MyFirstQueue | 0 | 0 | 2015-06-26 13:11:58 GMT-04:00 |

1 SQS Queue selected.

Details Permissions Redrive Policy

Name: MyFirstQueue
URL: https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue
ARN: arn:aws:sqs:us-east-1:523469370318:MyFirstQueue
Created: 2015-06-26 13:11:58 GMT-04:00
Last Updated: 2015-06-26 13:11:58 GMT-04:00
Delivery Delay: 0 seconds

Default Visibility Timeout: 30 seconds
Message Retention Period: 4 days
Maximum Message Size: 256 KB
Receive Message Wait Time: 0 seconds
Messages Available (Visible): 0
Messages in Flight (Not Visible): 0
Messages Delayed: 0

Chrome File Edit View History Bookmarks People Window Help

SQS Management Console x EC2 Management Console x aws-sqs-node-js-examp... x 54.208.245.163/create x

https://console.aws.amazon.com/sqs/home?region=us-east-1#send-message:selected=https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue;noRefresh=true;prefix=

AWS Services Edit Andrew Puch N. Virginia Support

Queues

Create New Queue Queue Actions Show/Hide Refresh

Filter by Prefix:

| Name | Messages Available | Messages in Flight | Created |
|--------------------------------------------------|--------------------|--------------------|-------------------------------|
| <input checked="" type="checkbox"/> MyFirstQueue | 0 | 0 | 2015-06-26 13:11:58 GMT-04:00 |

1 SQS Queue selected.

Details Permissions Redrive Policy

Name: MyFirstQueue
URL: https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue
ARN: arn:aws:sqs:us-east-1:523469370318:MyFirstQueue
Created: 2015-06-26 13:11:58 GMT-04:00
Last Updated: 2015-06-26 13:11:58 GMT-04:00
Delivery Delay: 0 seconds

Maximum Message Size: 256 KB
Receive Message Wait Time: 0 seconds
Messages Available (Visible): 0
Messages in Flight (Not Visible): 0
Messages Delayed: 0

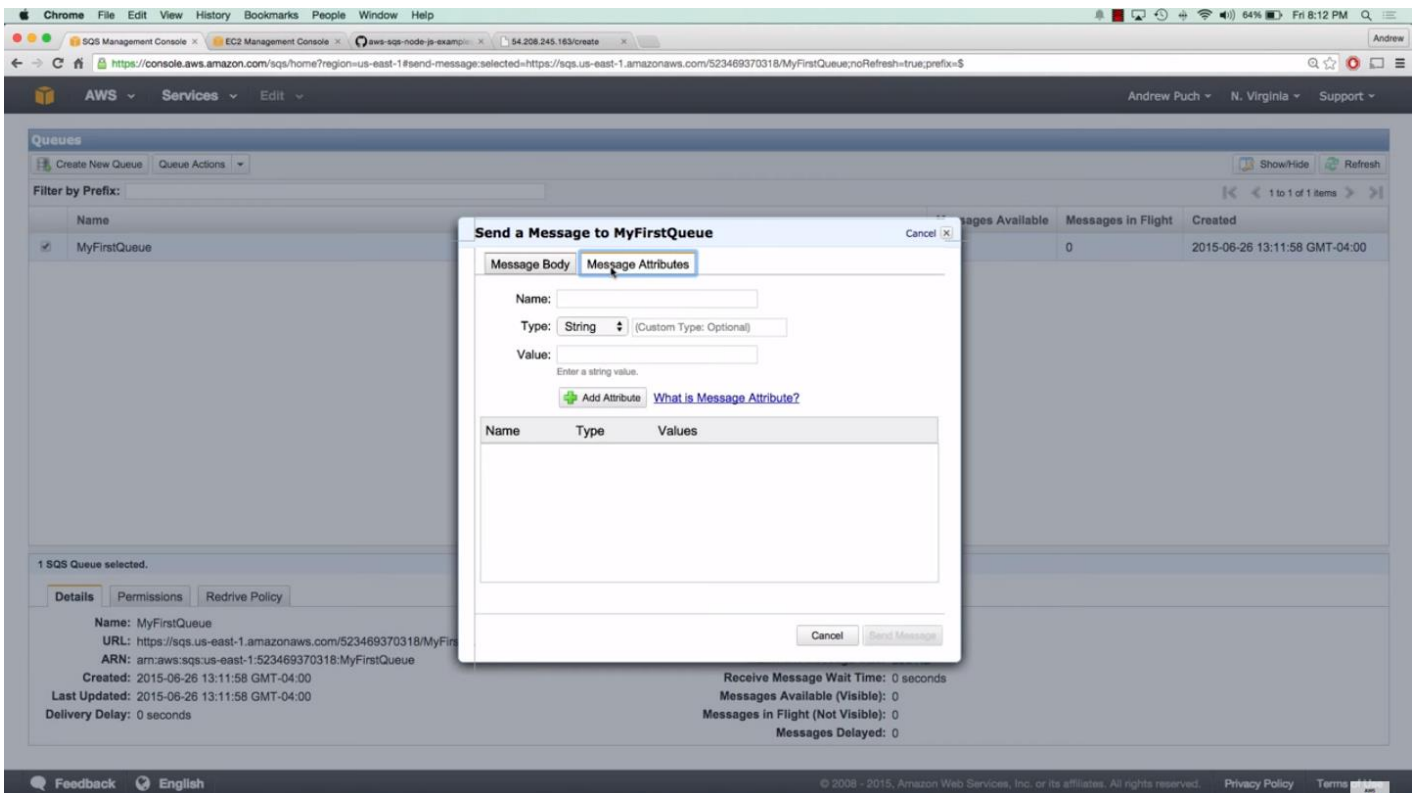
Send a Message to MyFirstQueue

Message Body Message Attributes

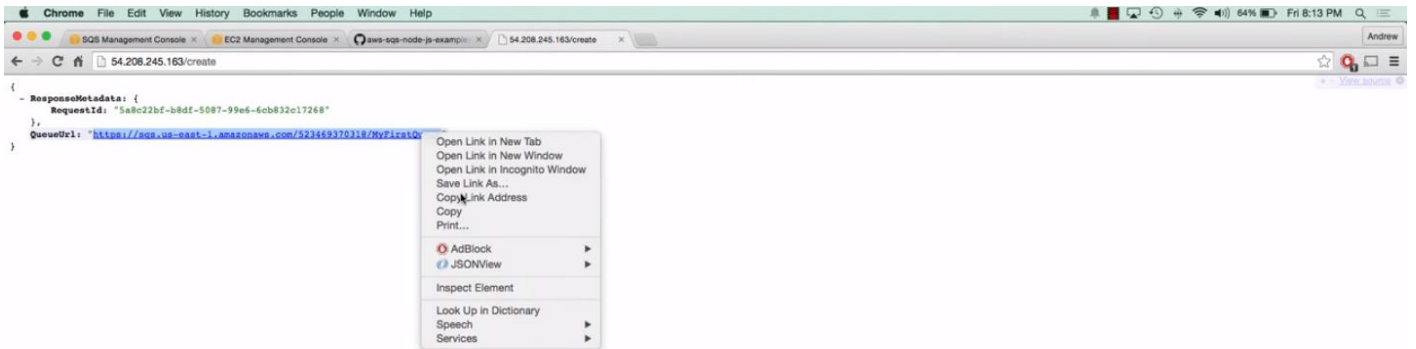
Enter the text of a message you want to send.

☐ Delay delivery of this message by 0 seconds (up to 15 minutes).

Cancel Send Message



You can do a bunch of different things with message attributes like selecting the type of attribute that you want to send, SQS will validate that the type is always correct.



Copy the SQS queue URL so that we can use it in our app.js file



```
Terminal Shell Edit View Window Help
aws-tutorial-series - root@ip-172-30-0-187: /home/ubuntu/aws-sqs-node-js-examples - ssh - 173x44
GNU nano 2.2.6 File: app.js

// Require objects.
var express = require('express');
var app = express();
var aws = require('aws-sdk');
var queueUrl = "";
var receipt = "";

// Load your AWS credentials and try to instantiate the object.
aws.config.loadFromPath(__dirname + '/config.json');

// Instantiate SQS.
var sqs = new aws.SQS();

// Creating a queue.
app.get('/create', function (req, res) {
  var params = {
    QueueName: "MyFirstQueue"
  };

  sqs.createQueue(params, function(err, data) {
    if(err) {
      res.send(err);
    }
    else {
      res.send(data);
    }
  });
});

// Listing our queues.
app.get('/list', function (req, res) {
  sqs.listQueues(function(err, data) {
    if(err) {
      res.send(err);
    }
    else {
      res.send(data);
    }
  });
});

Get Help      WriteOut      Read File      [ Read 123 lines
Exit          Justify          Where Is      Prev Page
Cut Text      Cur Pos
UnCut Text    To Spell
```

```
Terminal Shell Edit View Window Help
aws-tutorial-series - root@ip-172-30-0-187: /home/ubuntu/aws-sqs-node-js-examples - ssh - 173x44
GNU nano 2.2.6 File: app.js Modified

// Require objects.
var express = require('express');
var app = express();
var aws = require('aws-sdk');
var queueUrl = "https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue";
var receipt = "";

// Load your AWS credentials and try to instantiate the object.
aws.config.loadFromPath(__dirname + '/config.json');

// Instantiate SQS.
var sqs = new aws.SQS();

// Creating a queue.
app.get('/create', function (req, res) {
  var params = {
    QueueName: "MyFirstQueue"
  };

  sqs.createQueue(params, function(err, data) {
    if(err) {
      res.send(err);
    }
    else {
      res.send(data);
    }
  });
});

// Listing our queues.
app.get('/list', function (req, res) {
  sqs.listQueues(function(err, data) {
    if(err) {
      res.send(err);
    }
    else {
      res.send(data);
    }
  });
});

Get Help      WriteOut      Read File      Prev Page
Exit          Justify          Where Is      Next Page
Cut Text      Cur Pos
UnCut Text    To Spell
```

Save and close and restart the app

```
Terminal Shell Edit View Window Help
aws-tutorial-series - root@ip-172-30-0-187: /home/ubuntu/aws-sqs-node-js-examples - ssh - 173x44
root@ip-172-30-0-187:/home/ubuntu/aws-sqs-node-js-examples# node app.js
AWS SQS example app listening at http://0.0.0.0:80
^Croot@ip-172-30-0-187:/home/ubuntu/aws-sqs-node-js-examples# nano app.js
root@ip-172-30-0-187:/home/ubuntu/aws-sqs-node-js-examples# node app.js
AWS SQS example app listening at http://0.0.0.0:80
```



```
Chrome File Edit View History Bookmarks People Window Help
SQS Management Console EC2 Management Console aws-sqs-node-js-examples 54.208.245.163/create
54.208.245.163/create
{
  "ResponseMetadata": {
    "RequestId": "5a8c22bf-b8df-5087-99e6-6cb832c17268"
  },
  "QueueUrl": "https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue"
}
```

We are now going to send a message to that queue from our NodeJS app

```
Chrome File Edit View History Bookmarks People Window Help
SQS Management Console EC2 Management Console aws-sqs-node-js-examples 54.208.245.163/create
54.208.245.163/create
https://github.com/andrewpuch/aws-sqs-node-js-examples/blob/master/app.js
28 });
29
30 // Listing our queues.
31 app.get('/list', function (req, res) {
32   sqs.listQueues(function(err, data) {
33     if(err) {
34       res.send(err);
35     }
36     else {
37       res.send(data);
38     }
39   });
40 });
41
42 // Sending a message.
43 // NOTE: Here we need to populate the queue url you want to send to.
44 // That variable is indicated at the top of app.js.
45 app.get('/send', function (req, res) {
46   var params = {
47     MessageBody: 'Hello world!',
48     QueueUrl: queueUrl,
49     DelaySeconds: 0
50   };
51
52   sqs.sendMessage(params, function(err, data) {
53     if(err) {
54       res.send(err);
55     }
56     else {
57       res.send(data);
58     }
59   });
60 });
61
62 // Receive a message.
63 // NOTE: This is a great long polling example. You would want to perform
64 // this action on some sort of job server so that you can process these
65 // records. In this example I'm just showing you how to make the call.
66 // It will then put the message "in flight" and I won't be able to
67 // reach that message again until that visibility timeout is done.
68 app.get('/receive', function (req, res) {
69   var params = {
70     QueueUrl: queueUrl,
71     VisibilityTimeout: 600 // 10 min wait time for anyone else to process.
72   };
73 });
```

The **/send** endpoint is going to send a string to the queue as below

```
Chrome File Edit View History Bookmarks People Window Help
SQS Management Console EC2 Management Console aws-sqs-node-js-examples 54.208.245.163/send
54.208.245.163/send
{
  "ResponseMetadata": {
    "RequestId": "7de77c14-5b85-5b3c-b1b4-3213d12a108c"
  },
  "MD5OfMessageBody": "86fb269d190d2c85f6e04680eca42a20",
  "MessageId": "3d4eb777-b6cd-4b56-9abf-15079a2a3069"
}
```

We call the **/send** endpoint and get a valid response back with a messageId

Queues

Create New Queue Queue Actions Show/Hide Refresh

Filter by Prefix:

| Name | Messages Available | Messages in Flight | Created |
|--------------|--------------------|--------------------|-------------------------------|
| MyFirstQueue | 1 | 0 | 2015-06-26 13:11:58 GMT-04:00 |

1 SQS Queue selected.

Details Permissions Redrive Policy

Name: MyFirstQueue
URL: https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue
ARN: arn:aws:sqs:us-east-1:523469370318:MyFirstQueue
Created: 2015-06-26 13:11:58 GMT-04:00
Last Updated: 2015-06-26 13:11:58 GMT-04:00
Delivery Delay: 0 seconds

Default Visibility Timeout: 30 seconds
Message Retention Period: 4 days
Maximum Message Size: 256 KB
Receive Message Wait Time: 0 seconds
Messages Available (Visible): 1
Messages in Flight (Not Visible): 0
Messages Delayed: 0

Feedback English © 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms

We now have 1 message available for processing in our queue

```
56     else {
57         res.send(data);
58     }
59     });
60 });
61
62 // Receive a message.
63 // NOTE: This is a great long polling example. You would want to perform
64 // this action on some sort of job server so that you can process these
65 // records. In this example I'm just showing you how to make the call.
66 // It will then put the message "in flight" and I won't be able to
67 // reach that message again until that visibility timeout is done.
68 app.get('/receive', function (req, res) {
69     var params = {
70         QueueUrl: queueUrl,
71         VisibilityTimeout: 600 // 10 min wait time for anyone else to process.
72     };
73
74     sqs.receiveMessage(params, function(err, data) {
75         if(err) {
76             res.send(err);
77         }
78         else {
79             res.send(data);
80         }
81     });
82 });
83
84 // Deleting a message.
85 app.get('/delete', function (req, res) {
86     var params = {
87         QueueUrl: queueUrl,
88         ReceiptHandle: receipt
89     };
90
91     sqs.deleteMessage(params, function(err, data) {
92         if(err) {
93             res.send(err);
94         }
95         else {
96             res.send(data);
97         }
98     });
99 });
```

Next, we can now receive that message from the queue. We will call the **/receive** endpoint that will use the queueUrl to get the messages for us. We use the **VisibilityTimeout** value to tell the SQS queue that it is going to take us 600 seconds to process that request after getting it from the queue, the message status will change to **in flight** during the time when we have retrieved the message and its being processed.

```
{
  "ResponseMetadata": {
    "RequestId": "b9e659a-ee9f-567b-b39e-ab9d6474ed03"
  },
  "Messages": [
    {
      "MessageId": "3d4eb777-b6cd-4b56-9abf-15079a2a3069",
      "ReceiptHandle": "AQEBakhvq5mgL109Qf0y83f7eTzA8FocW85XDfT7mXb+5bYxW1Duhac9J00BuVAM2/8LLG1a+KpTBRLOCHdTWDOQv0cKp1t0W1W5C74APw1kfc0cGhLYR0U0V/jvD00n9V5I4+TNYOXDI1eNoFj+QbD4C+HDe18d3kLoMoHD4XPAbc7oY1trONY6Szhz12IWj1aMh5Jz8e5ShL8gK3eCVk47ExGRakeB8a7vyp",
      "MD5OfBody": "86fb269d19d2c85f6e0468ceca42a20",
      "Body": "Hello world!"
    }
  ]
}
```

We call the **/receive** endpoint and got a message that has a body content of ‘Hello world!’,

The screenshot shows the AWS Management Console interface for the 'Queues' section. A table lists the queue 'MyFirstQueue' with 0 messages available and 1 message in flight. Below the table, the details for 'MyFirstQueue' are shown, including its name, URL, ARN, and other metadata.

| Name | Messages Available | Messages in Flight | Created |
|--------------|--------------------|--------------------|-------------------------------|
| MyFirstQueue | 0 | 1 | 2015-06-26 13:11:58 GMT-04:00 |

1 SQS Queue selected.

Details Permissions Redrive Policy

Name: MyFirstQueue
URL: https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue
ARN: arn:aws:sqs:us-east-1:523469370318:MyFirstQueue
Created: 2015-06-26 13:11:58 GMT-04:00
Last Updated: 2015-06-26 13:11:58 GMT-04:00
Delivery Delay: 0 seconds

Default Visibility Timeout: 30 seconds
Message Retention Period: 4 days
Maximum Message Size: 256 KB
Receive Message Wait Time: 0 seconds
Messages Available (Visible): 0
Messages in Flight (Not Visible): 1
Messages Delayed: 0

We also see that it is in flight. This is very useful if you have some job scheduling process where you are constantly processing messages that arrive in a queue.

The screenshot shows a Chrome browser window displaying the response from the /receive endpoint. A context menu is open over the 'ReceiptHandle' value, with the 'Copy' option highlighted.

```
{
  "ResponseMetadata": {
    "RequestId": "b9e659a-ee9f-567b-b39e-ab9d6474ed03"
  },
  "Messages": [
    {
      "MessageId": "3d4eb777-b6cd-4b56-9abf-15079a2a3069",
      "ReceiptHandle": "AQEBakhvq5mgL109Qf0y83f7eTzA8FocW85XDfT7mXb+5bYxW1Duhac9J00BuVAM2/8LLG1a+KpTBRLOCHdTWDOQv0cKp1t0W1W5C74APw1kfc0cGhLYR0U0V/jvD00n9V5I4+TNYOXDI1eNoFj+QbD4C+HDe18d3kLoMoHD4XPAbc7oY1trONY6Szhz12IWj1aMh5Jz8e5ShL8gK3eCVk47ExGRakeB8a7vyp",
      "MD5OfBody": "86fb269d19d2c85f6e0468ceca42a20",
      "Body": "Hello world!"
    }
  ]
}
```

The screenshot shows the source code of the /receive endpoint response. The 'ReceiptHandle' value is highlighted.

```
{
  "ResponseMetadata": {
    "RequestId": "b9e659a-ee9f-567b-b39e-ab9d6474ed03"
  },
  "Messages": [
    {
      "MessageId": "3d4eb777-b6cd-4b56-9abf-15079a2a3069",
      "ReceiptHandle": "AQEBakhvq5mgL109Qf0y83f7eTzA8FocW85XDfT7mXb+5bYxW1Duhac9J00BuVAM2/8LLG1a+KpTBRLOCHdTWDOQv0cKp1t0W1W5C74APw1kfc0cGhLYR0U0V/jvD00n9V5I4+TNYOXDI1eNoFj+QbD4C+HDe18d3kLoMoHD4XPAbc7oY1trONY6Szhz12IWj1aMh5Jz8e5ShL8gK3eCVk47ExGRakeB8a7vyp",
      "MD5OfBody": "86fb269d19d2c85f6e0468ceca42a20",
      "Body": "Hello world!"
    }
  ]
}
```

We copy out the **ReceiptHandle** value from the message /receive endpoint response

```
Terminal Shell Edit View Window Help
awstutorialseries -- root@ip-172-30-0-187: /home/ubuntu/aws-sqs-node-js-examples -- ssh -- 173x44
root@ip-172-30-0-187: /home/ubuntu/aws-sqs-node-js-examples# nano app.js
```

```
GNU nano 2.2.6 File: app.js

// Require objects.
var express = require('express');
var app = express();
var aws = require('aws-sdk');
var queueUrl = "https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue";
var receipt = "";

// Load your AWS credentials and try to instantiate the object.
aws.config.loadFromPath(__dirname + '/config.json');

// Instantiate SQS.
var sqs = new aws.SQS();

// Creating a queue.
app.get('/create', function (req, res) {
  var params = {
    QueueName: "MyFirstQueue"
  };

  sqs.createQueue(params, function(err, data) {
    if(err) {
      res.send(err);
    }
    else {
      res.send(data);
    }
  });
});

// Listing our queues.
app.get('/list', function (req, res) {
  sqs.listQueues(function(err, data) {
    if(err) {
      res.send(err);
    }
    else {
      res.send(data);
    }
  });
});

[ Read 123 lines ]
Get Help WriteOut Read File Prev Page Cut Text Cur Pos
Exit Justify Where Is Next Page UnCut Text To Spell
```

```
Terminal Shell Edit View Window Help
awstutorialseries -- root@ip-172-30-0-187: /home/ubuntu/aws-sqs-node-js-examples -- ssh -- 173x44
GNU nano 2.2.6 File: app.js Modified

// Require objects.
var express = require('express');
var app = express();
var aws = require('aws-sdk');
var queueUrl = "https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue";
$N3IJ/Wox8PxtUtnf+k7MDEpdvk/4Bw==;

// Load your AWS credentials and try to instantiate the object.
aws.config.loadFromPath(__dirname + '/config.json');

// Instantiate SQS.
var sqs = new aws.SQS();

// Creating a queue.
app.get('/create', function (req, res) {
  var params = {
    QueueName: "MyFirstQueue"
  };

  sqs.createQueue(params, function(err, data) {
    if(err) {
      res.send(err);
    }
    else {
      res.send(data);
    }
  });
});

// Listing our queues.
app.get('/list', function (req, res) {
  sqs.listQueues(function(err, data) {
    if(err) {
      res.send(err);
    }
    else {
      res.send(data);
    }
  });
});

Get Help WriteOut Read File Prev Page Cut Text Cur Pos
Exit Justify Where Is Next Page UnCut Text To Spell
```

In our app.js file, we then change the receipt value to the value we got for the ReceiptHandle


```
Terminal Shell Edit View Window Help
aws-tutorial-series - root@ip-172-30-0-187: /home/ubuntu/aws-sqs-node-js-examples - ssh - 173x44
root@ip-172-30-0-187:/home/ubuntu/aws-sqs-node-js-examples# nano app.js
root@ip-172-30-0-187:/home/ubuntu/aws-sqs-node-js-examples# node app.js
AWS SQS example app listening at http://0.0.0.0:80
```

We restart our NodeJS app again

```
Chrome File Edit View History Bookmarks People Window Help
SQS Management Console x EC2 Management Console x aws-sqs-node-js-examp... x 54.208.245.163/receive x view-source:54.208.245.163/... x
https://github.com/andrewpuch/aws-sqs-node-js-examples/blob/master/app.js
74 sqs.receiveMessage(params, function(err, data) {
75   if(err) {
76     res.send(err);
77   }
78   else {
79     res.send(data);
80   }
81 });
82 });
83
84 // Deleting a message.
85 app.get('/delete', function (req, res) {
86   var params = {
87     QueueUrl: queueUrl,
88     ReceiptHandle: receipt
89   };
90
91   sqs.deleteMessage(params, function(err, data) {
92     if(err) {
93       res.send(err);
94     }
95     else {
96       res.send(data);
97     }
98   });
99 });
100
101 // Purging the entire queue.
102 app.get('/purge', function (req, res) {
103   var params = {
104     QueueUrl: queueUrl
105   };
106
107   sqs.purgeQueue(params, function(err, data) {
108     if(err) {
109       res.send(err);
110     }
111     else {
112       res.send(data);
113     }
114   });
115 });
116
117 // Start server.
118 var server = app.listen(80, function () {
```

Now we can call the **/delete** endpoint to remove that particular message from our queue once we finish processing it, we need to pass the receipt value back to identify the message to be deleted from the queue

```
Chrome File Edit View History Bookmarks People Window Help
SQS Management Console x EC2 Management Console x aws-sqs-node-js-examp... x 54.208.245.163/delete x view-source:54.208.245.163/... x
54.208.245.163/delete
{
  "ResponseMetadata": {
    "RequestId": "edd1a81a-bfc3-5743-8doc-e01d3487a8db"
  }
}
```

We call the **/delete** endpoint along with the receipt value and get a valid RequestId back in the response

Chrome File Edit View History Bookmarks People Window Help

SQS Management Console EC2 Management Console aws-sqs-node-js-exampl... 54.208.245.163/delete view-source:54.208.245.163/delete

https://console.aws.amazon.com/sqs/home?region=us-east-1#queue-browser:selected=https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue;prefix=

AWS Services Edit Andrew Puch N. Virginia Support

Queues

Create New Queue Queue Actions Show/Hide Refresh

Filter by Prefix:

| Name | Messages Available | Messages in Flight | Created |
|--------------------------------------------------|--------------------|--------------------|-------------------------------|
| <input checked="" type="checkbox"/> MyFirstQueue | 0 | 0 | 2015-06-26 13:11:58 GMT-04:00 |

1 SQS Queue selected.

Details Permissions Redrive Policy

Name: MyFirstQueue
URL: https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue
ARN: arn:aws:sqs:us-east-1:523469370318:MyFirstQueue
Created: 2015-06-26 13:11:58 GMT-04:00
Last Updated: 2015-06-26 13:11:58 GMT-04:00
Delivery Delay: 0 seconds

Default Visibility Timeout: 30 seconds
Message Retention Period: 4 days
Maximum Message Size: 256 KB
Receive Message Wait Time: 0 seconds
Messages Available (Visible): 0
Messages in Flight (Not Visible): 0
Messages Delayed: 0

Feedback English © 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

The message is now deleted from the queue

Chrome File Edit View History Bookmarks People Window Help

SQS Management Console EC2 Management Console aws-sqs-node-js-exampl... 54.208.245.163/send view-source:54.208.245.163/send

```
{
  "ResponseMetadata": {
    "RequestId": "c172051b-f6d4-57a1-973f-4b1c7d8a267e"
  },
  "SendMessageBody": "86fb269d19d2c85f6e0468ceca42a20",
  "MessageId": "e0bae19a-cd18-4600-be69-3ee7da5acc54"
}
```

We can also send in a bunch of messages into the SQS queue by calling the /send endpoint multiple times in this demo

The screenshot shows the AWS Management Console interface for an SQS queue. At the top, there's a navigation bar with 'AWS Services' and 'Edit' options. Below that, a 'Queues' section has a 'Create New Queue' button and a 'Queue Actions' dropdown. A table lists the queue 'MyFirstQueue' with 10 messages available and 0 in flight. Below the table, a 'Details' tab is selected, showing the queue's name, URL, ARN, creation time, last update time, and delivery delay. On the right, a summary of queue settings is displayed, including a 30-second visibility timeout, 4-day message retention, 256 KB maximum message size, 0-second receive message wait time, and 10 messages available.

| Name | Messages Available | Messages in Flight | Created |
|--------------|--------------------|--------------------|-------------------------------|
| MyFirstQueue | 10 | 0 | 2015-06-26 13:11:58 GMT-04:00 |

1 SQS Queue selected.

Details | Permissions | Redrive Policy

Name: MyFirstQueue
URL: https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue
ARN: arn:aws:sqs:us-east-1:523469370318:MyFirstQueue
Created: 2015-06-26 13:11:58 GMT-04:00
Last Updated: 2015-06-26 13:11:58 GMT-04:00
Delivery Delay: 0 seconds

Default Visibility Timeout: 30 seconds
Message Retention Period: 4 days
Maximum Message Size: 256 KB
Receive Message Wait Time: 0 seconds
Messages Available (Visible): 10
Messages in Flight (Not Visible): 0
Messages Delayed: 0

We have 10 messages in the queue now. Let us see how we can process these messages quickly

The screenshot shows a code editor with JavaScript code for an AWS SQS example application. The code includes a function to delete a message from the queue and a function to purge the entire queue. The purge function is highlighted with a yellow background. The code also shows the server listening on port 80 and logging the AWS SQS example app listening at http://%s:%s, host, port).

```
87 QueueUrl: queueUrl,
88 ReceiptHandle: receipt
89 };
90
91 sqs.deleteMessage(params, function(err, data) {
92   if(err) {
93     res.send(err);
94   }
95   else {
96     res.send(data);
97   }
98 });
99 });
100
101 // Purging the entire queue.
102 app.get('/purge', function (req, res) {
103   var params = {
104     QueueUrl: queueUrl
105   };
106
107   sqs.purgeQueue(params, function(err, data) {
108     if(err) {
109       res.send(err);
110     }
111     else {
112       res.send(data);
113     }
114   });
115 });
116
117 // Start server.
118 var server = app.listen(80, function () {
119   var host = server.address().address;
120   var port = server.address().port;
121
122   console.log('AWS SQS example app listening at http://%s:%s', host, port);
123 });
```

We can remove all the messages from the queue by calling the **/purge** endpoint of our app using the queueUrl value for the specific queue as shown above

The screenshot shows a web browser with the URL '54.208.245.163/purge'. The response is a JSON object with 'ResponseMetadata' and 'RequestId' fields. The 'RequestId' is '106e185b-57a5-580f-b04c-5450d0ddae20'.

```
{
  "ResponseMetadata": {
    "RequestId": "106e185b-57a5-580f-b04c-5450d0ddae20"
  }
}
```

We call the **/purge** endpoint and get a valid RequestId back saying it succeeded

Chrome File Edit View History Bookmarks People Window Help

SQS Management Console EC2 Management Console aws-sqs-node-js-examp... 54.208.245.183/purge view-source:54.208.245.183/purge

https://console.aws.amazon.com/sqs/home?region=us-east-1#queue-browser:selected=https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue;prefix=

AWS Services Edit Andrew Puch N. Virginia Support

Queues

Create New Queue Queue Actions Show/Hide Refresh

Filter by Prefix:

| Name | Messages Available | Messages in Flight | Created |
|--------------------------------------------------|--------------------|--------------------|-------------------------------|
| <input checked="" type="checkbox"/> MyFirstQueue | 0 | 0 | 2015-06-26 13:11:58 GMT-04:00 |

1 SQS Queue selected.

Details Permissions Redrive Policy

Name: MyFirstQueue
URL: https://sqs.us-east-1.amazonaws.com/523469370318/MyFirstQueue
ARN: arn:aws:sqs:us-east-1:523469370318:MyFirstQueue
Created: 2015-06-26 13:11:58 GMT-04:00
Last Updated: 2015-06-26 13:17:44 GMT-04:00
Delivery Delay: 0 seconds

Default Visibility Timeout: 30 seconds
Message Retention Period: 4 days
Maximum Message Size: 256 KB
Receive Message Wait Time: 0 seconds
Messages Available (Visible): 0
Messages in Flight (Not Visible): 0
Messages Delayed: 0

Feedback English

© 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

The messages are gone.