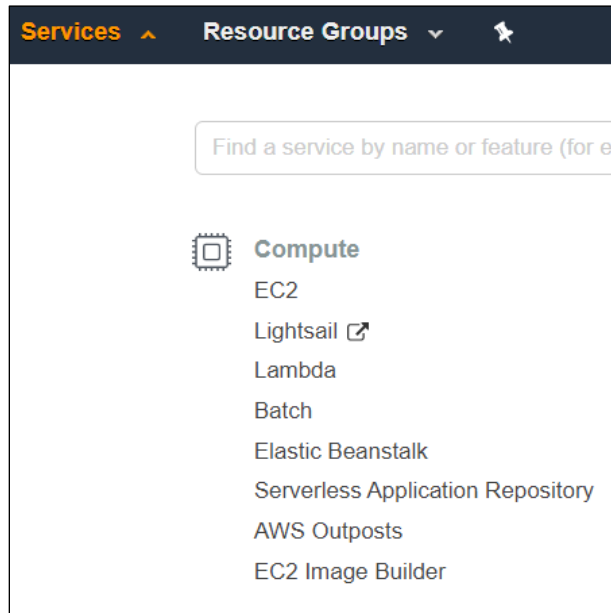


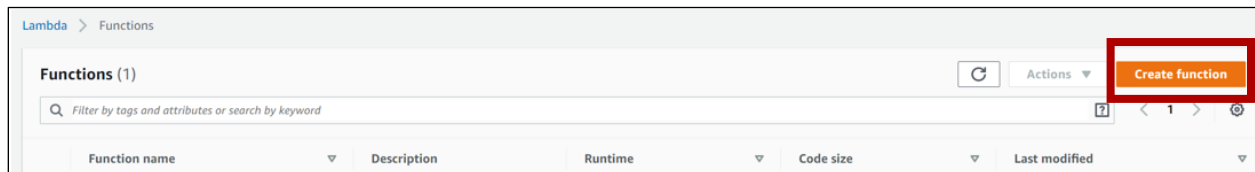
## CS 470 Module Four Assignment One Guide

### Part One – Creating and Testing a Lambda

1. Navigate to **AWS Lambda** by typing “Lambda” into the console search bar or selecting **Lambda** under **Compute**.



2. Click the orange **Create function** button in the upper-right corner.



3. Make sure **Author From Scratch** is selected.
4. Enter the function name “EchoFunction”.
5. Set the runtime to “Node.js 12.x”.
6. Leave the permissions option set to **Create a new role with basic Lambda permissions**.
7. Click the orange **Create Function** button in the lower-right corner.


Lambda > Functions > Create function

## Create function [Info](#)

Choose one of the following options to create your function.


**Author from scratch** [Info](#)

Start with a simple Hello World example.




**Use a blueprint** [Info](#)

Build a Lambda application from sample code and configuration presets for common use cases.



**Browse serverless app repository** [Info](#)

Deploy a sample Lambda application from the AWS Serverless Application Repository.



### Basic information

**Function name** [Info](#)  
Enter a name that describes the purpose of your function.

EchoFunction

Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
Choose the language to use to write your function.

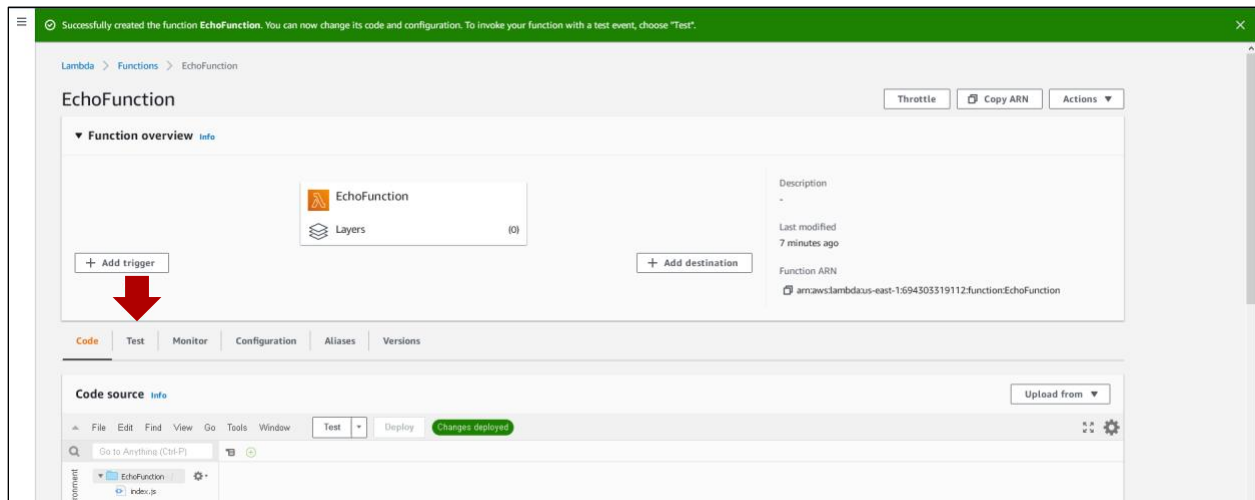
Node.js 12.x

**Permissions** [Info](#)  
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

► Choose or create an execution role

Cancel **Create function**

8. Congratulations! You have created your first serverless compute function – a bouncing baby Lambda! You should now see a screen like this:



- Click the **Test** tab in the left-hand side of the screen (see the red arrow).
- In the **Test event** box, enter an event name of "SimpleTestWithJsonData".
- Leave everything else the same and click the **Save changes** Button in the upper-right part of the screen.

Test event

Format

Save changes

Invoke

Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

☒ New event  
☐ Saved event

Template

hello-world

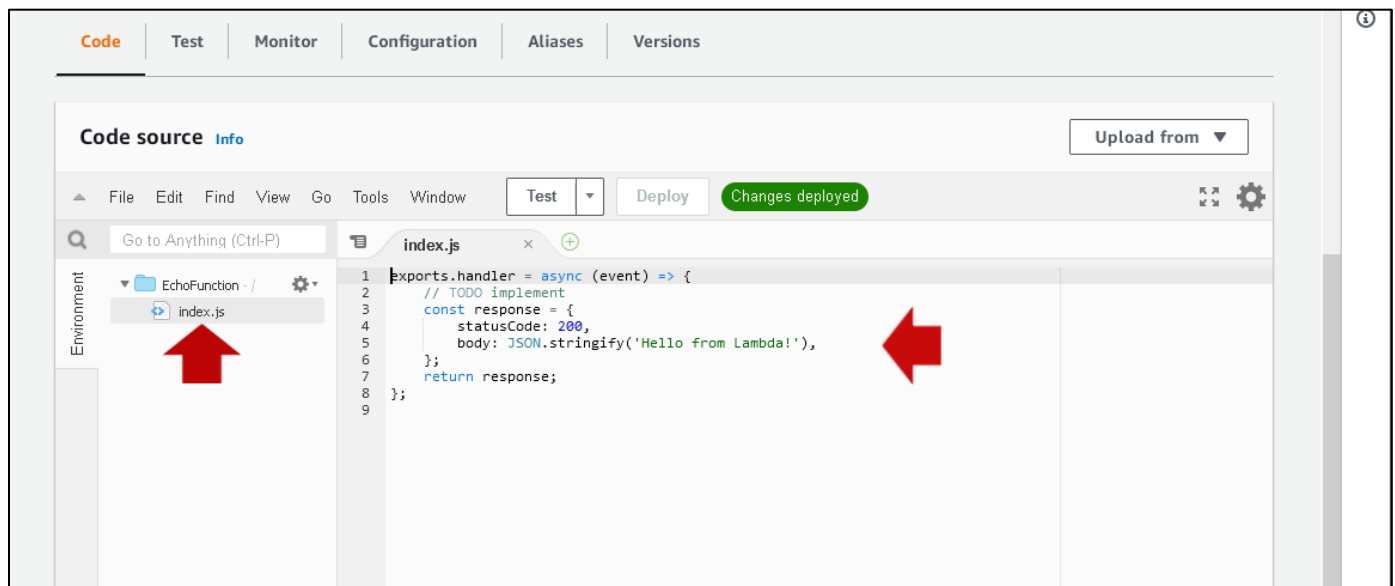
Name

SimpleTestWithJsonData

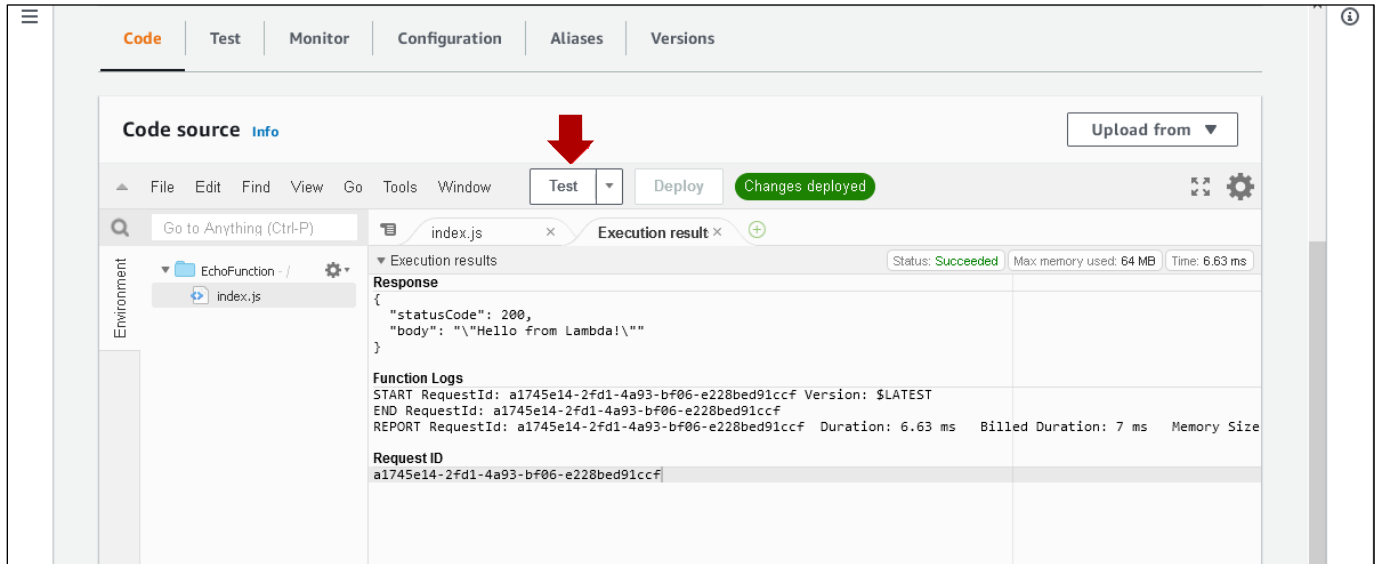
```

1 {
2   "key1": "value1",
3   "key2": "value2",
4   "key3": "value3"
5 }
  
```

12. You will still be looking at the Test tab in the Lambda console.
13. Now click on the Code tab and you will see the inline Lambda code editor in the box titled **Code source**. Double-left-click on the index.js symbol on the left side of the screen, and the code for the function will appear.



14. We will review the code in a moment, but for now, click the **Test** button on the editor.
15. Congratulations! You have now run your first Lambda! Your screen should look like this:

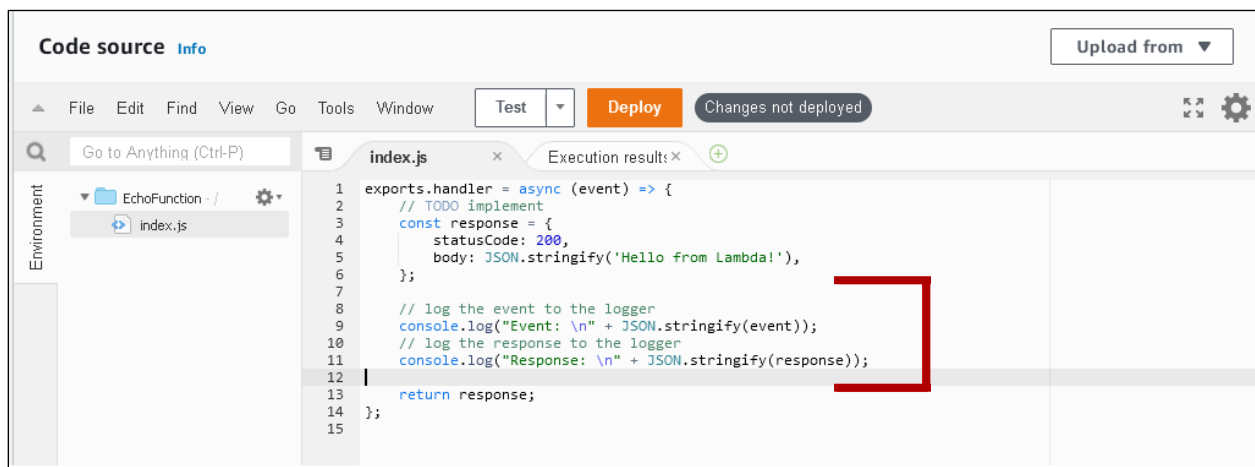


## Part Two – Creating an Echo Function

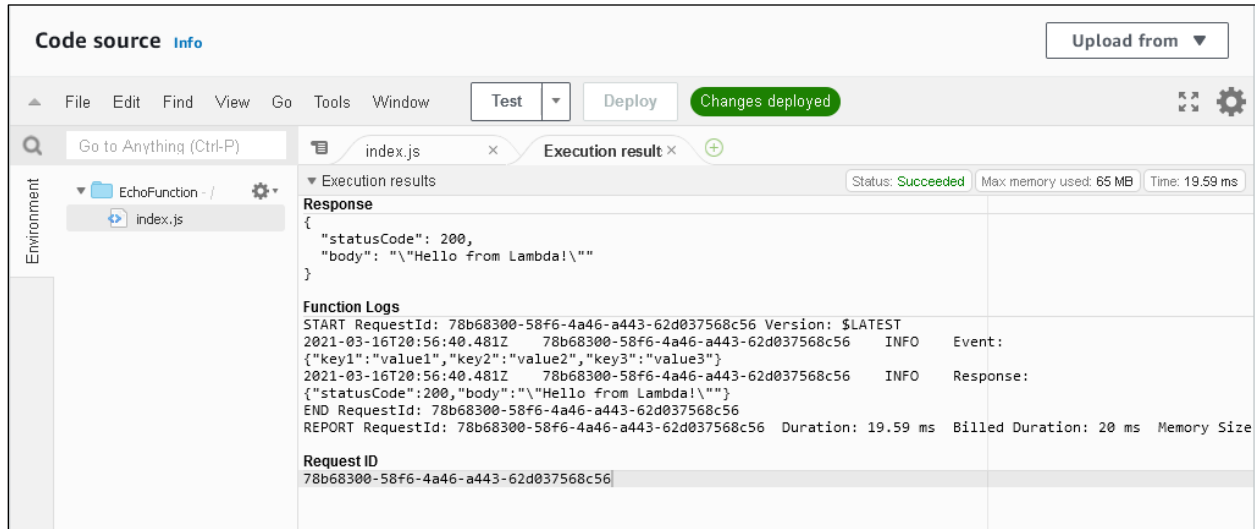
1. You are going to modify the code provided by AWS to perform a simple echo function.
2. Add the following code to index.js after the creation of the response object and before the return:

```
// log the event to the logger
console.log("Event: \n" + JSON.stringify(event));
// log the response to the logger
console.log("Response: \n" + JSON.stringify(response));
```

3. Your code should look like this:



4. Click **Deploy**.
5. Click **Test** again.
6. Your execution results will now look like this:



**Code source** [Info](#) Upload from ▼

File Edit Find View Go Tools Window Test ▼ Deploy Changes deployed

Go to Anything (Ctrl-P) index.js Execution result x +

Environment EchoFunction - / index.js

**Execution results** Status: Succeeded Max memory used: 65 MB Time: 19.59 ms

**Response**

```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```

**Function Logs**

```
START RequestId: 78b68300-58f6-4a46-a443-62d037568c56 Version: $LATEST
2021-03-16T20:56:40.481Z 78b68300-58f6-4a46-a443-62d037568c56 INFO Event:
{"key1":"value1","key2":"value2","key3":"value3"}
2021-03-16T20:56:40.481Z 78b68300-58f6-4a46-a443-62d037568c56 INFO Response:
{"statusCode":200,"body":"\"Hello from Lambda!\""
}
END RequestId: 78b68300-58f6-4a46-a443-62d037568c56
REPORT RequestId: 78b68300-58f6-4a46-a443-62d037568c56 Duration: 19.59 ms Billed Duration: 20 ms Memory Size
```

**Request ID**  
78b68300-58f6-4a46-a443-62d037568c56

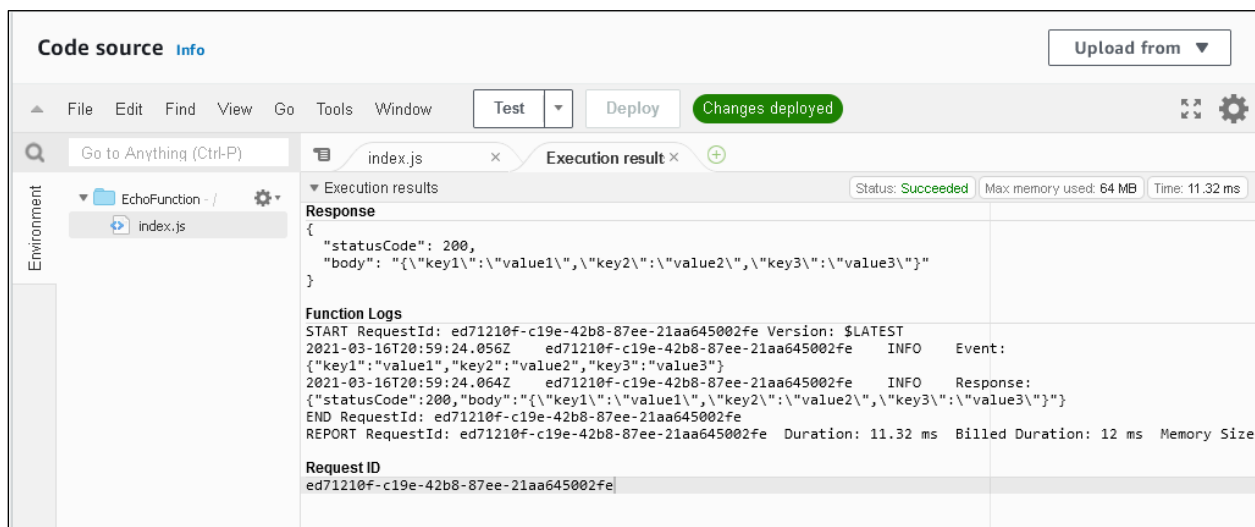
7. The values you passed in through the test event “SimpleTestWithJsonData” are now logged as the event object, and your response is the response object being created.
8. We want the response to echo what was passed in, so modify the code to do so. This will change the following code from:

```
// create some text to send back to the client
body: JSON.stringify('Hello from Lambda!'),
```

To:

```
// create some text to send back to the client
body: JSON.stringify(event),
```

9. Make sure to click **Save**.
10. Now click **Test** again and your execution results should be the following:



**Code source** [Info](#) Upload from ▼

File Edit Find View Go Tools Window Test ▼ Deploy Changes deployed

Go to Anything (Ctrl-P) index.js Execution result x +

Environment EchoFunction - / index.js

**Execution results** Status: Succeeded Max memory used: 64 MB Time: 11.32 ms

**Response**

```
{
  "statusCode": 200,
  "body": "\"{\\\"key1\\\":\\\"value1\\\",\\\"key2\\\":\\\"value2\\\",\\\"key3\\\":\\\"value3\\\"}\""
}
```

**Function Logs**

```
START RequestId: ed71210f-c19e-42b8-87ee-21aa645002fe Version: $LATEST
2021-03-16T20:59:24.056Z ed71210f-c19e-42b8-87ee-21aa645002fe INFO Event:
{"key1":"value1","key2":"value2","key3":"value3"}
2021-03-16T20:59:24.064Z ed71210f-c19e-42b8-87ee-21aa645002fe INFO Response:
{"statusCode":200,"body":"\"{\\\"key1\\\":\\\"value1\\\",\\\"key2\\\":\\\"value2\\\",\\\"key3\\\":\\\"value3\\\"}\""
}
END RequestId: ed71210f-c19e-42b8-87ee-21aa645002fe
REPORT RequestId: ed71210f-c19e-42b8-87ee-21aa645002fe Duration: 11.32 ms Billed Duration: 12 ms Memory Size
```

**Request ID**  
ed71210f-c19e-42b8-87ee-21aa645002fe

## Part Three – Enhancing the Echo Function

1. You can now echo your results, at least for that JSON data that was passed in. Let's do something different.
2. First, create a new test class by clicking the down arrow next to **Test** and selecting **Configure test event**.
3. Select **Create New Test Event** and name it "EchoWithQuery", with the following JSON body:

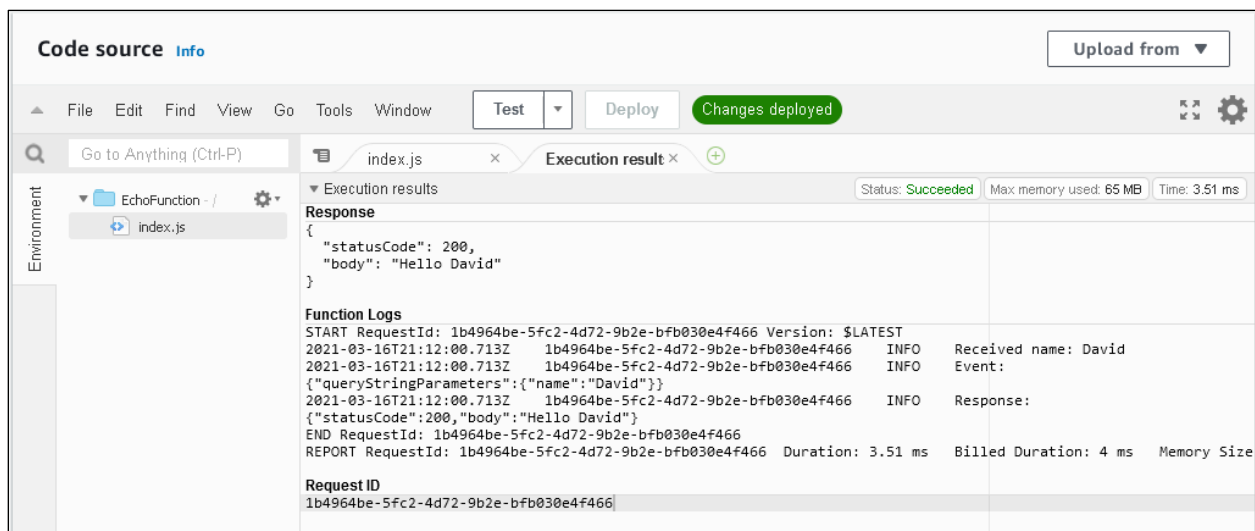
```
{
  "queryStringParameters": {
    "name": "David"
  }
}
```

4. Click **Create**.
5. Now replace the code from the TODO to the log statements with the following:

```
var name = 'unknown';
if (event.queryStringParameters && event.queryStringParameters.name) {
  console.log("Received name: " + event.queryStringParameters.name);
  name = event.queryStringParameters.name;
}

const response = {
  statusCode: 200,
  body: "Hello " + name,
};
```

6. Click **Deploy**.
7. Run your EchoWithQuery test and you should see the following execution results:



The screenshot shows the AWS Lambda console interface. At the top, there's a 'Code source' link and an 'Upload from' dropdown. Below that is a toolbar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test', and 'Deploy' buttons. A green badge indicates 'Changes deployed'. The main area is divided into two panes. The left pane shows the 'Environment' section with a folder 'EchoFunction - /' and a file 'index.js'. The right pane shows the 'Execution result' for the 'index.js' file. It displays the 'Response' as a JSON object: { "statusCode": 200, "body": "Hello David" }. Below the response, there are 'Function Logs' showing the execution details, including the request ID, version, and the log output: 'Received name: David' and 'Event: { "queryStringParameters": { "name": "David" } }'. The logs also show the response: { "statusCode": 200, "body": "Hello David" }. At the bottom, there's a 'Request ID' field with the value '1b4964be-5fc2-4d72-9b2e-bfb030e4f466'.