LESSON 5

In lesson 5 you are to create a new function that will update Slack after a new image (with emojis) is created.

1. UPDATING SLACK

You have done all this work and your serverless pipeline can now create new images. It is time to write the function that will update Slack. The good news is that we don't need to send an image to slack to display. We can just send a URL and Slack will display it for us. However, your S3 files are private and protected. Sending a URL to the file will not work. So, to get around this problem, you will create a **signed** url that will allow anyone with that url to view the file. This will also work for displaying the url in slack.

- Make a copy of the Lesson 4 folder and name it Lesson 5
- In the Lesson 5 folder open **slackupdate.js**
- Copy the following implementation to this file

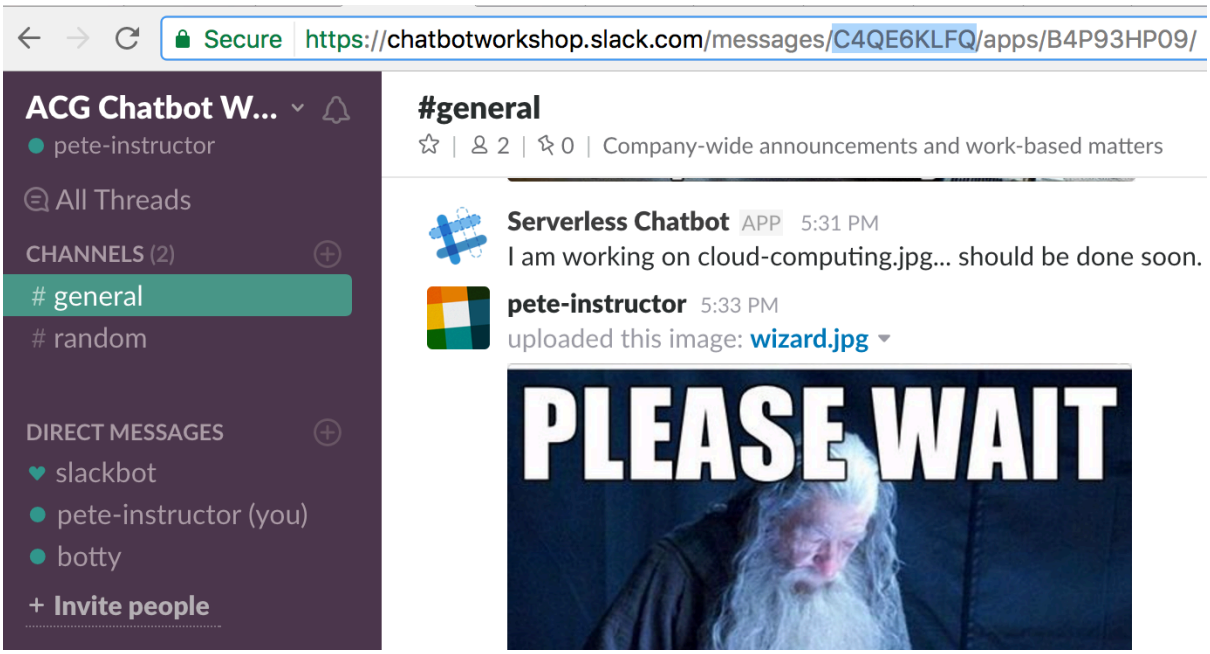
```
'use strict';

const aws = require('aws-sdk');
const https = require('https');
const qs = require('querystring');
const s3 = new aws.S3();

const getSignedUrl = function(bucket, key) {
    return new Promise((resolve, reject) => {
        const params = {Bucket: bucket, Key: key, Expires: 604800};
        var url = s3.getSignedUrl('getObject', params);
        resolve(url);
    });
};

const writeToSlack = function(url) {
  return new Promise((resolve, reject) => {
    const response = {
        token: process.env.BOT_ACCESS_TOKEN,
        channel: process.env.CHANNEL_ID,
        text: url
      }

      const slackurl = process.env.POST_MESSAGE_URL + qs.stringify(response);

      https.get(slackurl, (res) => {
        const statusCode = res.statusCode;
        resolve();
      })
  });
}

module.exports.execute = (event, context, callback) => {
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, " "));
```

```
  getSignedUrl(bucket, key)
    .then((url) => writeToSlack(url))
    .then(() => callback(null))
    .catch((err) => callback(err))
};
```

2. CHANNEL ID

Previously when you posted a message back to Slack, you had the Channel ID from the incoming message generated by Slack. Unfortunately, right now, we don't have this information. So, you are going to include your **General** Channel ID as as environment variable. This means that all links will be automatically sent to **General**.

- Open your Slack in the web browser
- Click on the **General** channel
- Look at the URL and copy the channel ID which should be between **messages** and **apps** in the URL. The ID should be about 9 characters long and include numbers and letters.



- Open **serverless.yml** for Lesson 5
- Update the environment variables of the **slackupdate** function to include the following information

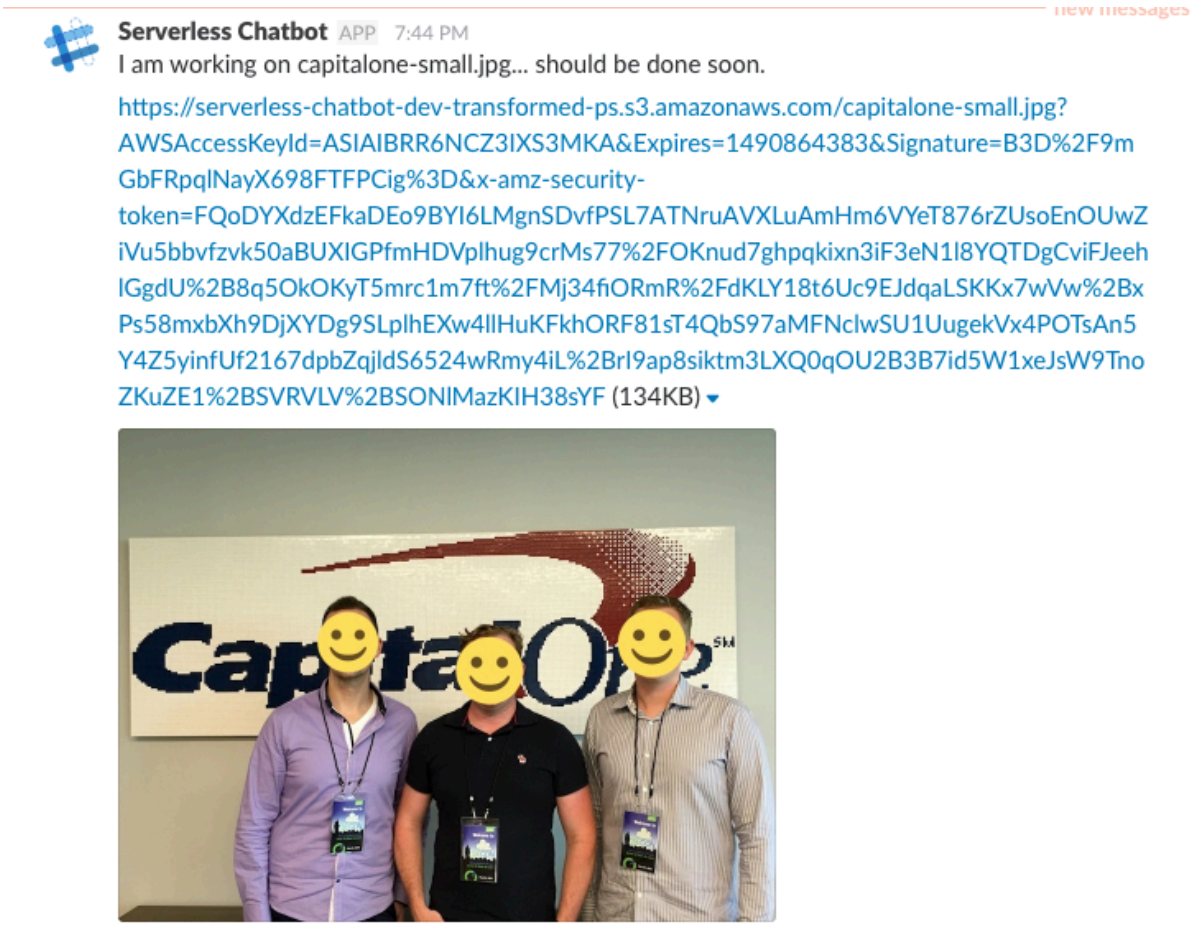
```
  slackupdate:
    handler: slackupdate.execute
    environment:
      POST_MESSAGE_URL: 'https://slack.com/api/chat.postMessage?'
      BOT_ACCESS_TOKEN: 'BOT ACCESS TOKEN FROM PREVIOUS LESSONS'
      CHANNEL_ID: 'CHANNEL ID YOU JUST COPIED'
```

3. DEPLOY

Let's deploy and see what it looks like now. Type **serverless deploy** from the Lesson 5 folder in your terminal. Wait for the deployment to finish.

4. TEST

Upload an image to a Slack channel and wait for a minute or two. Do you see a giant new (signed) URL appear out of nowhere and the image?

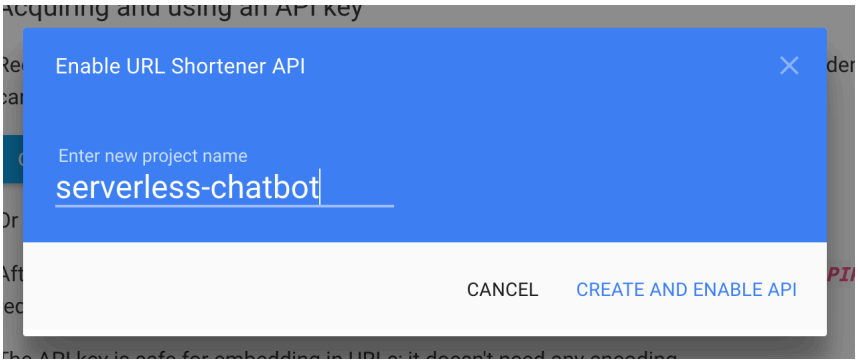


That is a pretty scary URL and it doesn't look super good. How could we make it a bit prettier?

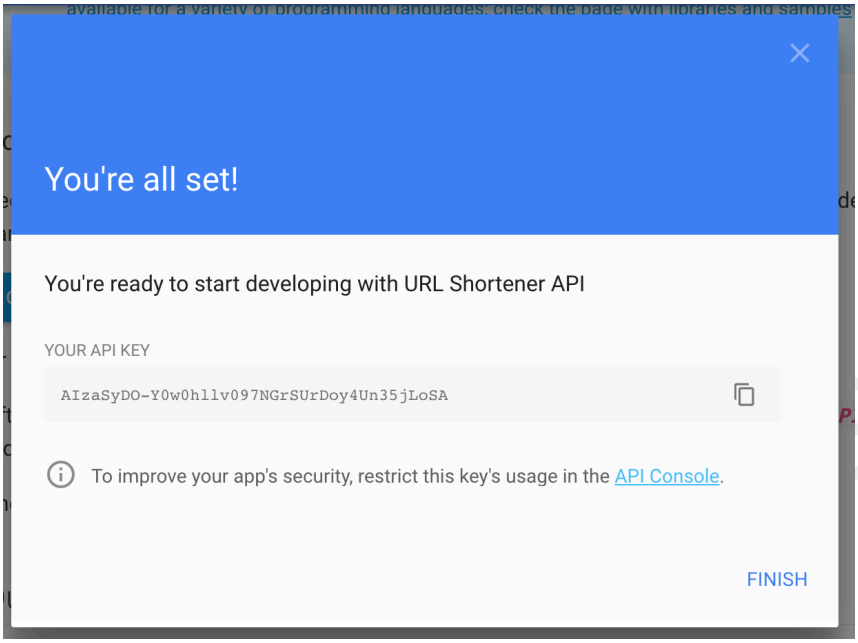
5. SHORTEN IT

Let's use the Google Shortener API. There is a little bit of a set up but it's quick.

- Go to https://developers.google.com/url-shortener/v1/getting_started
- Scroll down until you see **Acquiring and using an API key**
- Click on the button called **Get A Key**
- Create a new project called **serverless-chatbot**



- Click **Create and Enable API**
- Copy your API Key somewhere safe and click **Finish**



- Open **serverless.yml** and add a new environment variable to the **slackupdate** function called **SHORTENER_API_KEY**. Set it to your API Key.
- Then add another variable called **SHORTENER_API_URL** and set it to https://www.googleapis.com/urlshortener/v1/url

```
  environment:
    POST_MESSAGE_URL: 'https://slack.com/api/chat.postMessage?'
    BOT_ACCESS_TOKEN: 'xoxb-159279836768-FOst5DLfEzmQgkz7cte5qiIv'
    CHANNEL_ID: 'C4QE6KLFQ'
    SHORTENER_API_KEY: 'AIzaSyDO-Y0w0hllv097NGrSUrDoy4Un35jLoSA'
    SHORTENER_API_URL: 'https://www.googleapis.com/urlshortener/v1/url?'
```

6. POST IT

Google requires you to POST to it to get a short url. You can do it using the native HTTP functionality in nodejs but it isn't very user friendly. So, you are going to install the **request** module. It'll greatly simplify your life.

- Open your terminal to Lesson 5
- Type **npm install request --save** and hit enter
- After the installation finishes you can check that the dependency was added by looking at package.json

7. MORE IMPLEMENTATION

Let's update the implementation of **slackupdate.js** to issue a request to Google to get a short URL.

- At the top of the **slackupdate.js** file add the following line. It should go just under **use strict**.

```
const request = require('request');
```

- Add the following function under the **getSignedUrl** function

```
const getShortUrl = function(url) {
    return new Promise((resolve, reject) => {
        var req = {
            uri: process.env.SHORTENER_API_URL + qs.stringify({key: process.env.SHORTENER_API_KEY}),
            method: 'POST',
            json: true,
            body: {
                longUrl: url
            }
        }

        request(req, (err, res, body) => {
            if (err && res.statusCode !== 200) {
                reject(err);
            } else {
                resolve(body.id);
            }
        });
    });
}
```

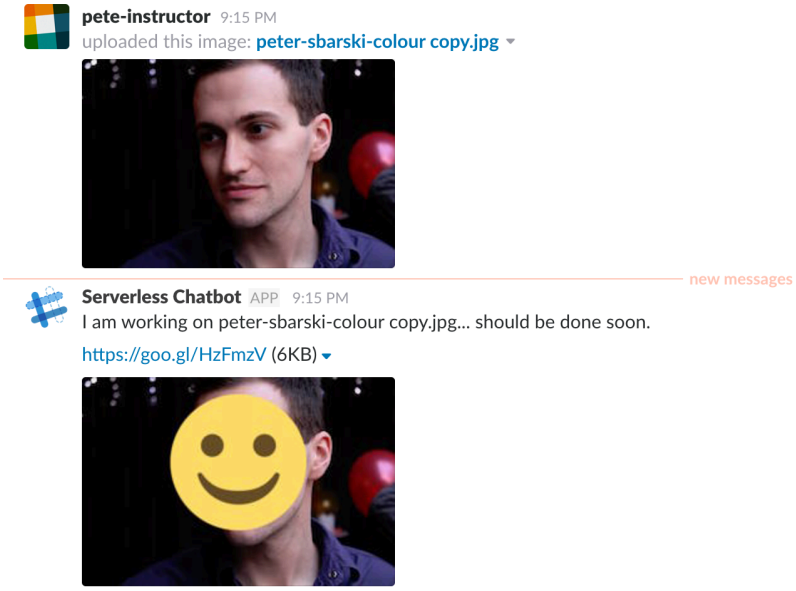
- Finally, modify the **execute** function to be

```
module.exports.execute = (event, context, callback) => {
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, " "));

  getSignedUrl(bucket, key)
    .then((url) => getShortUrl(url))
    .then((url) => writeToSlack(url))
    .then(() => callback(null))
    .catch((err) => callback(err))
};
```

8. DEPLOY AND TEST AGAIN

You can now deploy and test again in Slack.

- Run **serverless deploy** from the terminal (make sure to be in the Lesson 5 folder)
- Upload a test image to Slack and see if you get another image with an emoji appear in the General.



What a journey! You've now built your serverless chatbot that also does image processing. How cool is that?!

ADVANCED QUESTIONS

1. The getSignedUrl function has an expiry timeout. It means that after a period of time the images will not be accessible. Change the implementation of the slackupdate function to make the URL **public** instead of using getSignedUrl.