LESSON 6

In lesson 6 you will learn how to share your app with other teams.

1. MAKING THINGS RIGHT

Your Slack Bot is coupled to your Slack team. That's ok while you've been learning but what if you wanted to make it installable by anyone? After all, it would be fun to ship our bot and make it available for anyone to use. You are going to create an install Lambda function and a DynamoDB table that will allow users to install and registers their bots. Our DynamoDB table will maintain a list of registration and various bots. And, we'll grow it to contain other information too.

- Make a coupy of the Lesson 5 folder and name it **Lesson 6**
- In the Lesson 6 folder create a new file called **install.js**
- Add the following to **serverless.yml**

```
functions:
  install:
    handler: install.endpoint
    events:
      - http:
          path: install
          method: get
```

- You are going to add a DynamoDB table that will contain information about each team that has installed the bot. So, you need to specify a DynamoDB table in the **resource** section of the **serverless.yml** file and the Serverless Framework will create it for you. Add the following to the bottom of your **serverless.yml**.


```
resources:
  Resources:
    TeamsDynamoDbTable:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: ${self:service}-${self:provider.stage}-teams
        AttributeDefinitions:
          - AttributeName: team_id
            AttributeType: S
        KeySchema:
          - AttributeName: team_id
            KeyType: HASH
        ProvisionedThroughput:
          ReadCapacityUnits: 1
          WriteCapacityUnits: 1
```

- Finally, add the following to the **iamRoleStatements** section

```
    - Effect: Allow
      Action:
        - dynamodb:GetItem
        - dynamodb:PutItem
      Resource:
        - arn:aws:dynamodb:*:*:table/${self:service}-${self:provider.stage}-teams
```

Remember to save your serverless.yml file once you are finished with it.

2. INSTALL FUNCTION

You have an install function in **install.js** so open it up and copy the following implementation to it. In a nutshell, this function will be invoked by Slack. It will then go out and get an OAuth token with the additional information about the team. Finally, it will store this information in to a DynamoDB table.

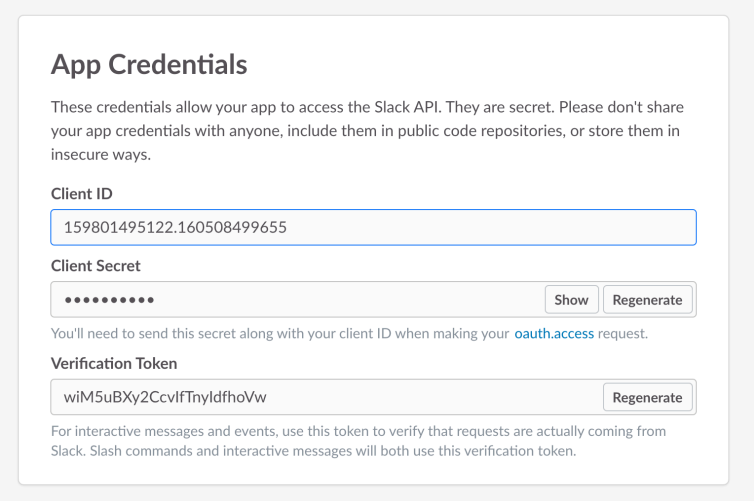
```
'use strict';

const aws = require('aws-sdk');
const qs = require('querystring');
const request = require('request');

const db = new aws.DynamoDB.DocumentClient();

const extractCode = function(event) {
    return new Promise((resolve, reject) => {
        if (event.queryStringParameters && event.queryStringParameters.code) {
            return resolve(event.queryStringParameters.code);
        }

        reject('Code not provided');
    });
};

const getOAuthToken = function(code) {
    return new Promise((resolve, reject) => {
        if (code === null) { return reject('Could not provided'); }

        const params = {
            client_id: process.env.CLIENT_ID,
            client_secret: process.env.CLIENT_SECRET,
            code
        }

        var url = process.env.SLACK_OAUTH + qs.stringify(params);

        request.get(url, (err, res, body) => {
            if (err || res.statusCode !== 200) {
                reject(err);
            } else {
                resolve(body);
            }
        })
    });
};

const saveToDynamo = function(response) {
    return new Promise((resolve, reject) => {
        const params = {
```

```
            TableName: process.env.TABLE_NAME,
            Item: JSON.parse(response)
        }

        db.put(params, (err, data) =>{
            if (err) {
                reject(err);
            } else {
                resolve();
            }
        })
    });
}

const successResponse = function() {
    return {
        statusCode: 200
    }
}

const errorResponse = function() {
    return {
        statusCode: 302
    }
}

module.exports.endpoint = (event, context, callback) => {
    extractCode(event)
        .then((code) => getOAuthToken(code))
        .then((response) => saveToDynamo(response))
        .then(() => callback(null, successResponse()))
        .catch(() => callback(null, errorResponse()))
};
```

- There is a change you need to make to **serverless.yml** too. You need to add a few environment variables referred to in the code above. Set the following to **serverless.yml**.

```
functions:
  install:
    handler: install.endpoint
    events:
      - http:
          path: install
          method: get
    environment:
      CLIENT_ID: '159801495122.160508499655'
      CLIENT_SECRET: b3f939ef9af2a6c5cc848001f6bac147
      SLACK_OAUTH: 'https://slack.com/api/oauth.access?'
      TABLE_NAME: ${self:service}-${self:provider.stage}-teams
```

You will need to replace the Client ID and the Client Secret with your own ID and secret. You will find these in your Slack API settings if you click **Basic Information** and then scroll down to **App Credentials**.



3. DEPLOYMENT

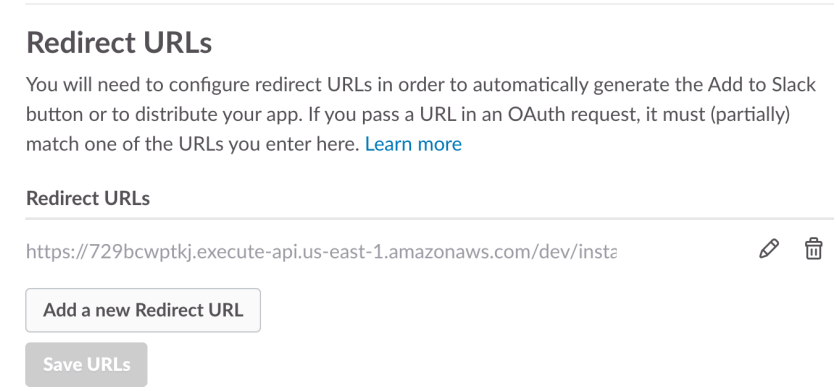
Having done everything in the previous steps, deploy your bot by issues **serverless deploy** from the terminal. Note the new URL (it ends with /dev/install) you will get after the deployment is finished. You will need it in the next step.

```
Serverless: Packaging service...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading service .zip file to S3 (1.86 MB)...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.....................................
Serverless: Stack update finished...
Serverless: Removing old service versions...
Service Information
service: serverless-chatbot
stage: dev
region: us-east-1
api keys:
  None
endpoints:
  GET - https://729bcwptkj.execute-api.us-east-1.amazonaws.com/dev/install
  POST - https://729bcwptkj.execute-api.us-east-1.amazonaws.com/dev/echo
functions:
  install: serverless-chatbot-dev-install
  hello: serverless-chatbot-dev-hello
  faceswap: serverless-chatbot-dev-faceswap
  slackupdate: serverless-chatbot-dev-slackupdate
```

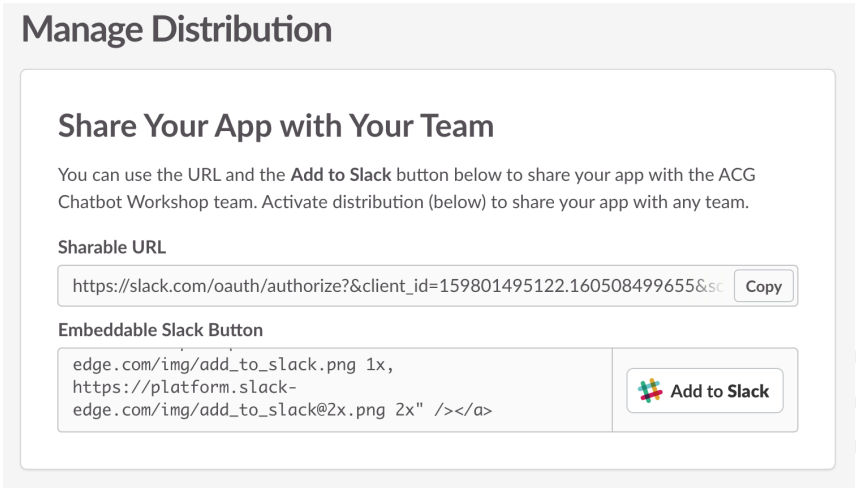
4. GENERATING INSTALL LINK

You can finally generate an install link and an install button to share with others.

- In the Slack API click **Oauth & Permissions**.
- Scroll to **Redirect URLs** and add a new URL. This should be your new **install** URL from the previous step. Don’t forget to save!

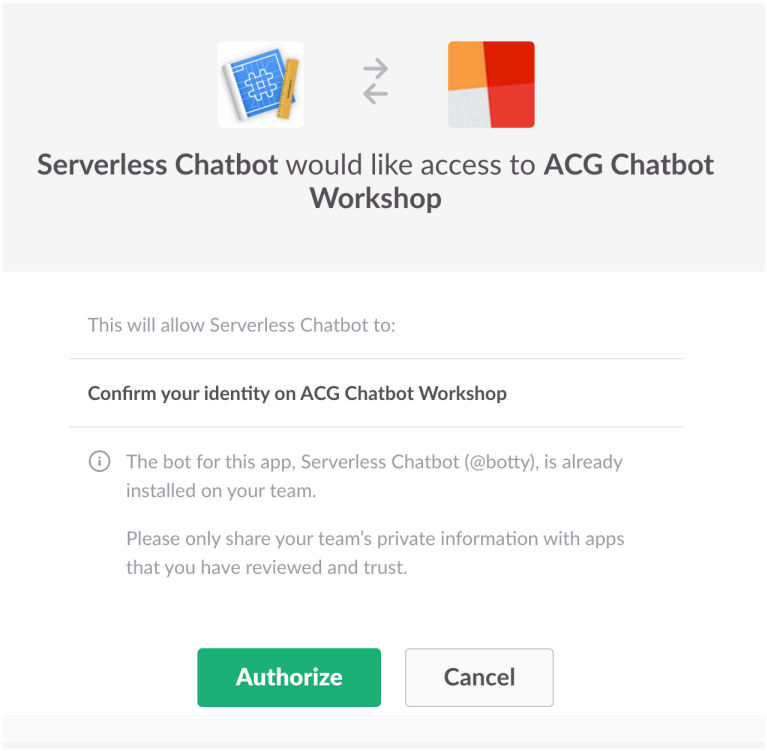


- Click **Manage Distribution** on the left.
- You should see a **sharable URL** and code for an **Embeddable Slack Button**



5. TESTING

Let's test the installation. Grab the **Sharable URL,** paste it in to your browser's address bar, and hit enter. You should see something like this.

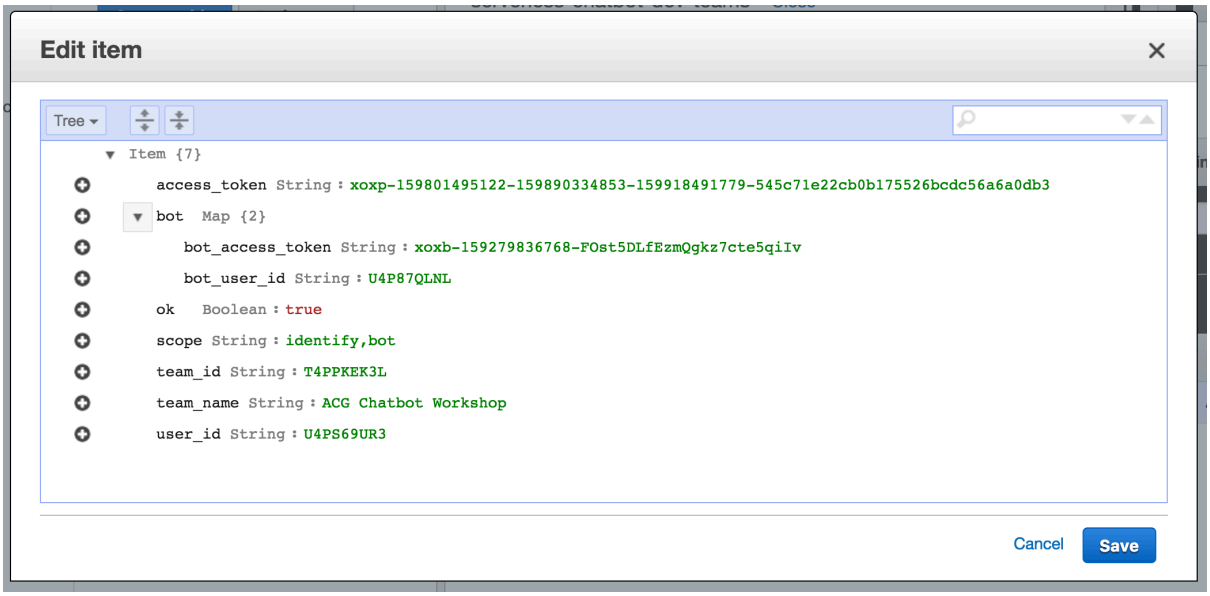


- You already have the bot installed but click **Authorize** anyway.
- You will see a blank screen because we haven't set up a redirect URL but if you inspect the network tab of your browser you should see a 200 response.

6. DYNAMODB

Let's check out DynamoDB to see if the response from Slack was saved correctly.

- In the AWS console click on **DynamoDB**
- Click **Tables**
- Select **serverless-chatbot-dev-teams**
- Select **Items**
- Click on the item in the grid to see it



7. HARDCODED VALUES

We'd love to ship your bot now but we cannot. You have hardcoded values, including the bot_access_token, in your **serverless.yml**. Your challenge is to modify the other functions you have to read from the database to get the relevant information, rather than relying on hardcoded environment variables.

If you look at **handler.js** it refers to the BOT_ACCESS_TOKEN environment variable. But, you can now change it to look up the bot access token in DynamoDB based on the team. You can get the team out of the request that slack sends to our Lambda function. Let's try it now.

- Open **handler.js** and add the following new function

```
const getBotAccessToken = function(team) {
    return new Promise((resolve, reject) => {

        var params = {
            TableName: process.env.TEAMS_TABLE,
            Key: {
                "team_id" : {
                    S: team
                }
            }
        }

        db.getItem(params, (err, data) =>{
            if (err) {
                reject(err);
            } else {
                resolve(data.Item.bot.M.bot_access_token.S);
            }
        });
    });
}
```

- Change the start of the **downloadFileToSystem** to this (note the **accessToken**)

```
const downloadFileToSystem = function(accessToken, path, filename) {
    var file = fs.createWriteStream(process.env.TEMP_FOLDER + filename);

    const options = {
        hostname: process.env.SLACK_HOSTNAME,
        path: path,
        headers: {
            authorization: 'Bearer ' + accessToken
        }
    };
```

- Change the start of the **updateStatusInSlack** to (note the **accessToken**)

```
const updateStatusInSlack = function(accessToken, filename, channel) {
  return new Promise((resolve, reject) => {

    const response = {
        token: accessToken,
        channel: channel,
        text: 'I am working on ' + filename + '... should be done soon.'
      }
```

- Change the endpoint function to

```
module.exports.endpoint = (event, context, callback) => {
  const request = JSON.parse(event.body);

  if (request.event.type && request.event.type === 'message' &&
      request.event.subtype && request.event.subtype === 'file_share') {

    const path = request.event.file.url_private_download;
    const filename = request.event.file.name;
    const channel = request.event.channel;
    var accessToken = '';

    getBotAccessToken(request.team_id)
        .then((token) => {accessToken = token; return downloadFileToSystem(accessToken, path, filename)})
        .then(() => uploadToBucket(filename))
        .then(() => updateStatusInSlack(accessToken, filename, channel))
        .then(() => callback(null, {statusCode: 200}))
        .catch(() => callback(null, {statusCode: 500}));

      return;
  }

  callback(null, {statusCode: 200});
};
```

- In **serverless.yml** remove **BOT_ACCESS_TOKEN** from the **hello** function.

- Redeploy your bot by typing **serverless deploy** from the terminal and upload a new image in Slack. Everything should work as before.

ADVANCED QUESTIONS

1. You still have Slack specific variables encoded as environment variables for certain functions. Modify those functions to use DynamoDB rather than using environment variables.