

Run a Cisco Spark Bot locally

Duration: 15 minutes

Objectives

As introduced earlier, Cisco Spark Bots are applications that invoke the Cisco Spark API under a Bot account identity.

In this lab, you will learn to run your own Cisco Spark Bot by taking several steps: create a Bot account, run a sample Cisco Spark bot on your local machine, expose your bot to the internet., and register two Webhooks in order to have Cisco Spark post notifications to your bot. Finally, you will interact with your bot:



MyAwesomeBot (bot) 12:12
Hi, I am the Hello World bot !
Type /hello to see me in action.



You 12:13
/hello



MyAwesomeBot (bot) 12:13
Hello Cisco DevNet Learning Labs

Pre-requisites

You will need a Cisco Spark user account to complete this lab. If you're not a Cisco Spark user yet, [click to sign up](#).

How to setup your own computer

Skip this section if you are using a machine provided by the event organizers.

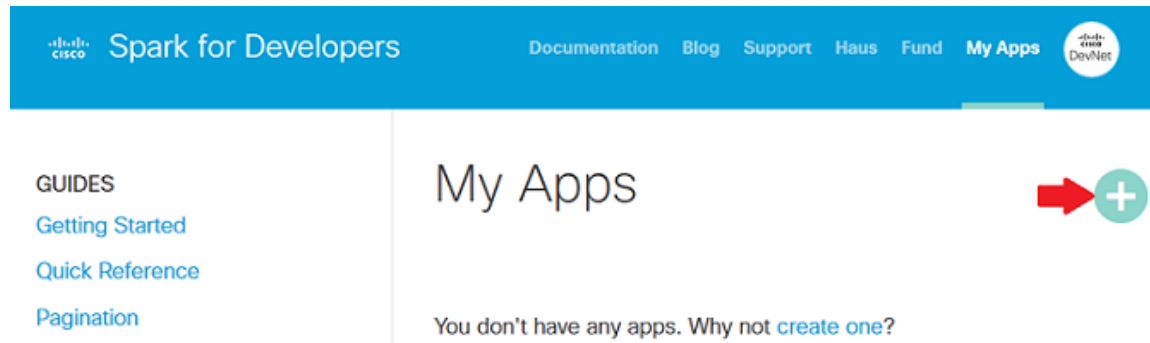
As we will run a bot sample, this requires a minimal NodeJS development environment on your local machine.

- NodeJS runtime: make sure you have both the "node" and "npm" runtimes installed locally on your machine. This lab has been tested with NodeJS v4.
- [optional] a NodeJS IDE: this lab does not require you to open any NodeJS files locally, but this will be mandatory if you opt for the "to go further" instructions.
- "git" command

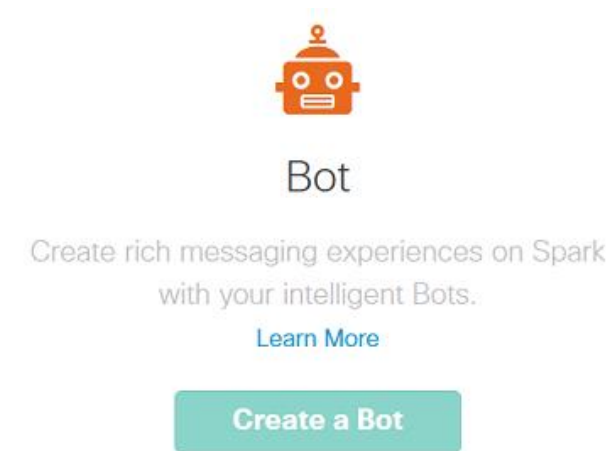
As we will visualize JSON payloads, make sure your Web browser embeds a JSON payload beautifier to make this experience more convenient. If you're using Chrome, [add json-viewer](#).

Step 1: Create a Bot account

Log in at "[Spark for Developers](#)" and open the "My Apps" menu.



Click on the "+" button, and choose "Create a Bot".



You will access the "New Bot" [creation form](#) below.

Fill in a name, a unique email identifier, and specify a publicly accessible image URL with a minimal resolution of 512x512 pixels. Feel free to pick this [image example](#) for the sake of this lab.

Note that you will not be authorized to pick the email "my-awesome-bot@sparkbot.io" as it is already reserved. Make sure to replace future mentions to the bot email you have just chosen.

Display Name

The name users will communicate with after adding your bot to a room

MyAwesomeBot

(bot)

Bot Username

The username users will use to add your bot to a room. Cannot be changed later.

my-awesome-bot

@sparkbot.io ✓

Icon

A URL to an image that is 512x512 or larger is required



Remove

Click “Add Bot” to get your Cisco Spark Bot created.

Your Bot's access token is displayed. Paste it in a safe place as it won't show up again, and we will use it in steps 2 and 3 of this lab.

Note that a Cisco Spark Bot access token lasts 100 years. If you ever lose or reveal it, you can come back to this Bot details page and regenerate an access token. The previously issued token will be automatically deprecated.

Access token

ZkMThNjMtYTM1Yi00ZjNjLWI3YTAuMTg5YTkyMDhkOTI3ZmFmZTczM2YtMWQ0

Copy

Make sure to save your access token above. You can always regenerate an Access Token, but you will not see this one again.

Your bot can now be added to any Cisco Spark Room by specifying its email: my-awesome-bot@sparkbot.io in our example.

Go to your Cisco Spark client, and create a new room with your Bot as a participant.

Search for people to chat, share documents and video call.

my-awesome-bot@sparkbot.io

Go Chat!

Even though you can chat with your bot, you won't see him answer ... as we haven't connected it yet to any custom code logic. We'll work on that in the next steps.

Step 2: Run a bot on your local machine

For the sake of this lab, we will clone an existing code sample, and configure it to run under the Cisco Spark identity of the Bot we just created.

This bot consists...

Fill in the "Clone from Git" field with the URL of our code sample: <https://github.com/CiscoDevNet/node-sparkbot-samples>.

Run the commands below to clone the [github project](#), get a local copy of the provided example, install the project dependencies and launch your integration.

Make sure your local environment meet all the pre-requisities. The git, NodeJS and npm commands are required to copy and launch the example on your local machine. If your environment does not provide the "make" command, you can paste the run section of the Makefile as an alternative.

```
> git clone https://github.com/CiscoDevNet/node-sparkbot-samples
> cd node-sparkbot-samples
> npm install
> DEBUG=sparkbot*,samples* SPARK_TOKEN=YOUR_BOT_TOKEN_HERE node examples/hello
world.js
...
```

- **DEBUG** with "sparkbot*" as a value: this variable activates logs,
- **SPARK_TOKEN** with your bot's token value: paste the token generated in step 1.

Check your bot health check is now live at <http://localhost:8080>.

Congrats, you now have your own copy of a Cisco Spark Bot!

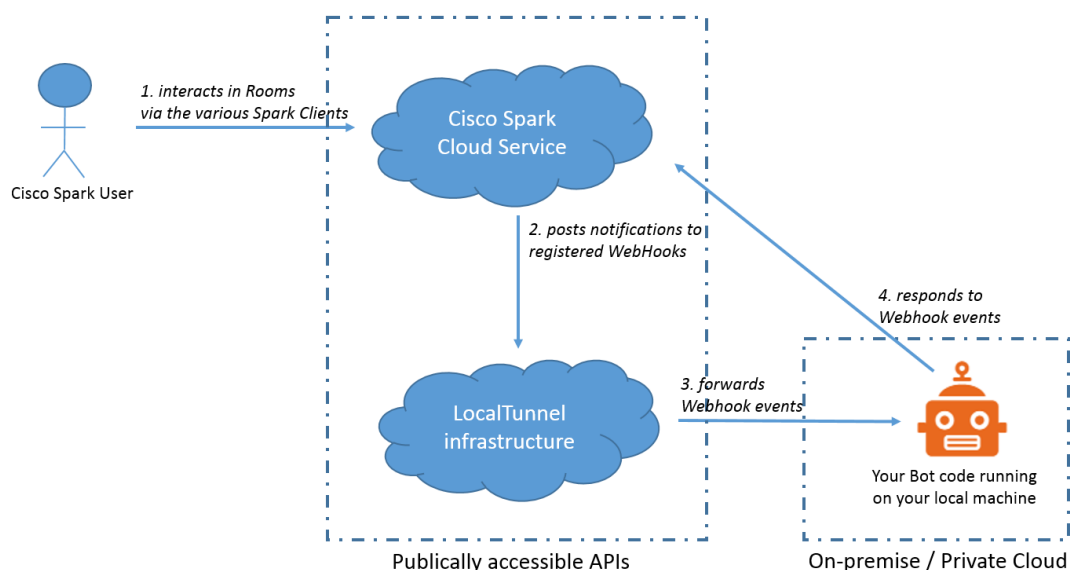
Step 3: Expose your Bot on the internet

Cisco Spark requires your Bot to expose a public endpoint, in order to send it real-time notifications about what's happening in the rooms your bot has joined.

In this section, we will therefore create a tunnel between your machine and the internet so that the port your bot is currently listening at, will now be publically accessible. [LocalTunnel](#) and [ngrok](#) are two popular tunneling technologies, with freemium plans for developers.

Note that we chose localtunnel for its ease of installation, but also because the same subdomain can be reused among several launch of the tool, which greatly simplifying the webhook registration we'll setup in the next section. NGrok provides subdomain preservation as a paying feature.

Here is the global architecture we'll setup. This section focusses on the communication labelled 3 in the figure below, which ensures our bot receives events as Cisco Spark posts them to a publically accessible endpoint.



Type “npm install –g localtunnel” on your local machine to install localtunnel globally.

An “lt” command is now available at the command line.

Choose a subdomain name, and create a tunnel from your machine port 8080 to the internet, by typing the command “lt –s myawesomebot –p 8080”.

Note that if the subdomain is already in use, localtunnel will return an error message. Simply opt for another subdomain.

Here is a sumup of the commands run in this section.

TODO: add commands

Your Bot is now exposed live on the internet, and can be reached **through HTTP and HTTPS** at the following address: <https://<your-subdomain>.localtunnel.me/>

Hit your bot public URL on localtunnel, and check its health check responds successfully. We'll dig into more details concerning the JSON payload that is displayed to you.

Note that the snapshot below leverages the Chrome plugin "[JSON viewer](#)".

TODO: add snapshot

Step 4: Create Webhook events

In this section, we will create REST Webhooks so that our bot starts receiving notifications from Cisco Spark.

For the sake of this lab, we embed a health-check in our code sample.

In other terms, when hitting your bot Root URL on localhost or on localtunnel, your bot returns a 200 OK response, with extra configuration and deployment information.

If you look attentively to the JSON data below, you will see your bot is running under the Cisco Spark Identity of the Bot Account you created earlier. Moreover, the “listeners” section lists the events your bot can take action on.

TODO add snapshot

Our next task is to create Webhooks.

Open the [list of Webhooks](#) supported by Cisco Spark, and look for the event entries needed by your bot:

- Messages / Created: a new message got posted into a room
- Memberships / Created: someone joined a room that you are in

Click on the Webhooks entry in the API Reference section on the left, and select the “Post” method. This will drive you to the [Create a Webhook](#) form.

Fill in the fields as shown in the snapshot below:

- Authorization header: change the access token to your bot’s, and *do NOT remove the “Bearer ” prefix*,
- targetUrl: paste the public URL of your bot on Cloud9,
- resource and event: make sure to fill in the “messages” and “created” values, as it is only default placeholders you see in the form,
- last fields “Secret” and “Filter” are optional. Leave them blank.

Create a Webhook


Test Mode ☒

Creates a webhook.





POST `https://api.ciscospark.com/v1/webhooks`

Request Headers

Content-type `application/json; charset=utf-8`

Authorization  `Bearer MTQwMjdiOWYyY2Q0Ni00ODM4LTlhODYtOGJlZTBjOW`

Request Parameters

Name	Type	Your values		Required
name	string	 <code>Webhooks for my bot on Cloud9</code>	(i)	✓
targetUrl	string	 <code>https://bot-sample-on-cloud9-obji</code>	(i)	✓
resource	string	 <code>messages</code>	(i)	✓
event	string	 <code>created</code>	(i)	✓

Request

```
{
  "name": "Webhooks for my bot on Cloud9",
  "targetUrl": "https://bot-sample-on-cloud9-objectisad vantag.c9user",
  "resource": "messages",
  "event": "created"
}
```

Response

200 / success

```
{
  "id": "Y2l2Y29zcGFyazovL3VzL1dFQkhPT0svNTdkNmQ1NTQthjkyZC0wM2RkLWJ",
  "name": "Webhooks for my bot on Cloud9",
  "targetUrl": "https://bot-sample-on-cloud9-objectisad vantag.c9user",
  "resource": "messages",
  "event": "created",
  "orgId": "Y2l2Y29zcGFyazovL3VzL1dFQkhPT0svNTdkNmQ1NTQthjkyZC0wM2RkLWJ",
  "createdBy": "Y2l2Y29zcGFyazovL3VzL1dFQkhPT0svNTdkNmQ1NTQthjkyZC0wM2RkLWJ",
  "appId": "Y2l2Y29zcGFyazovL3VzL1dFQkhPT0svNTdkNmQ1NTQthjkyZC0wM2RkLWJ",
  "ownedBy": "creator",
  "created": "2016-10-25T13:26:46.719Z"
}
```

Then click “Run” and check the response in the right panel.

As your Cisco Spark API call completes successfully, you will see a green “200/success” displayed, and your Webhook will be assigned a unique identifier (check the “id” field). This webhook is fired every time a new message is added to a Cisco Spark room your bot is a member of.

Now, let's create our second Webhook, in order to receive an event every time our bot is added to a room.

On the same form, modify the value of the "resource" field: replace “messages” with “memberships”.

Click “Run” again, and check your second Webhook got successfully created, with the “200 / success” message.

Step 5: Test your Cisco Spark Bot

In previous steps, you deployed a Cisco Spark bot sample on Cloud9, launched a new instance under the identity of a Bot Account you created, and create Webhooks to start receiving events at your bot public URL on Cloud9. Now comes time to chat with your bot.

Reach to the Cisco Spark room you created in step 1.

Enter `/hello`.

Check your bot's response!



Mention your bot in Group rooms

As introduced in previous labs, Cisco Spark Bots MUST be mentioned in “Group” rooms in order to be notified.

Open the “[Create a Room](#)” form.

Fill in the form with a title. Do NOT modify the token in the Authorization header, so that the room gets created under your personal Cisco Spark identity.

Check the result is a "200/success", and the room type is set to “Group”. As you're the only participant in the room for now, Cisco Spark created a "Group" room to let you add participants later.

Create a Room

Test Mode ☒

Creates a room. The authenticated user is automatically added as a member of the room. See the Memberships API to learn how to add more people to the room.

POST <https://api.ciscospark.com/v1/rooms>

Request Headers

Content-type application/json; charset=utf-8

Authorization Bearer MmNkYzUwYzAtZDU1MC00MzdhlTg5ZC

Request Parameters

Name	Type	Your values	Required
title	string	<u>Test room for my first bot</u>	<input checked="" type="checkbox"/>
roomId	string	<u>Y2lzY29zcGFyazovL3VzL1</u>	<input type="checkbox"/>

Request

```
{
  "title": "Test room for my first bot"
}
```

Response

200 / success

```
{
  "id": "Y2lzY29zcGFyazovL3VzL1JPT00vNGY0NDc4YzUwYzAtZDU1MC00MzdhlTg5ZC",
  "title": "Test room for my first bot",
  "type": "group",
  "isLocked": false,
  "lastActivity": "2016-10-26T13:44:25.977Z",
  "creatorId": "Y2lzY29zcGFyazovL3VzL1BFT1BMRSA",
  "created": "2016-10-26T13:44:25.932Z"
}
```

Now, open your favorite Cisco Spark client, and reach to the newly created room.

Invite your bot to the conversation by adding it as a participant, with the email address you created in step 1.

Type /hello.

You do not get any answer as your bot is not mentioned.

Now mention your bot... and watch for its answer.

 MyAwesomeBot (bot) 12:19
Hi, I am the Hello World bot!
Type /hello to see me in action.
Note that this is a 'Group' room. I will wake up only when mentioned.

 You 12:19
/hello

 You 12:22
MyAwesomeBot /hello

 MyAwesomeBot (bot) 12:23
Hello Cisco DevNet Learning Labs

To go further

What about extending your bot!

If you have a fully functional development environment at your disposal, we suggest you start by adding a few breakpoints in your bot main file "helloworld.js", and restart your bot in debug mode.

Then, you'll want to add some commands of your own. If you're looking for some inspiration, take a look at these [examples](#).

Note that for the sake of this lab, we leveraged a nodejs bot framework that hides most of the complexity of listening to messages and taking action as commands are fired. Note that several other nodejs Bot frameworks exist for Cisco Spark, from basic to advanced. As an example, [Nick Marus's flint framework](#) embeds interesting features such as automated creation of the Webhooks required by your bot... Certainly worth a try...