Deploy a Cisco Spark Bot

Duration: 20 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,
- deploy a nodejs code sample from Cloud9 development environment,
- and finally register two Webhooks to have your bot receive events as they happen in Cisco Spark rooms.

How to setup your own computer

In this learning lab, we will visualize JSON payloads. To make this experience more convenient, make sure your Web browser embeds a JSON payload beautifier. If you're using Chrome, click here to add json-viewer.

Pre-requisites

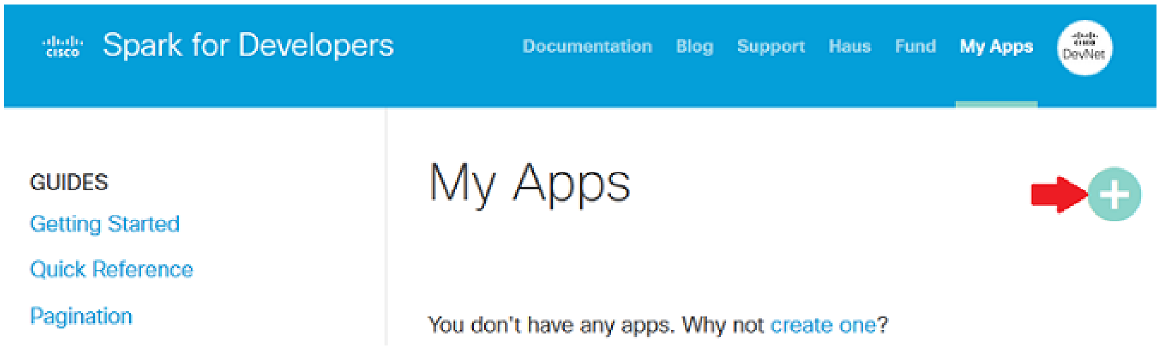
You will need a Cisco Spark user account to complete this lab. If you're not a Cisco Spark user yet, click to sign up.

Moreover, we will create a Cloud9 workspace from an existing nodejs bot, and register it on Cisco Spark. **Cloud9 by Amazon is a Cloud-based development environment** to edit, run and debug source code in various languages. Cloud9 free plan lets you work on an unlimited number of public workspaces, and one private workspace. Click to sign up if you are not already a Cloud9 user.

Note that the Cloud9 signup process requires credit card details, even though you won't be charged until you opt for a paying plan.

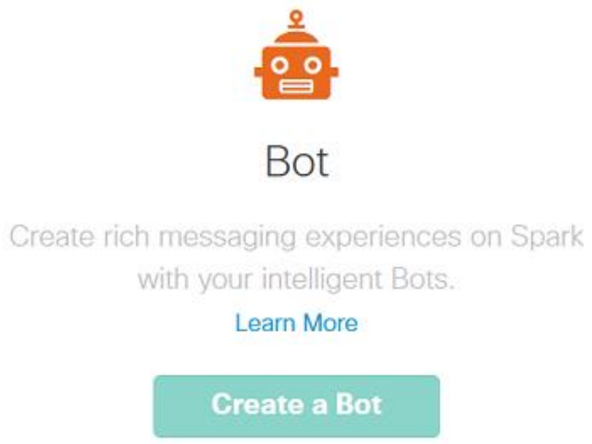
Step 1: Create a Bot account

Log in at "Spark for Developers" and open the "My Apps" menu.



You don't have any apps. Why not create one?

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

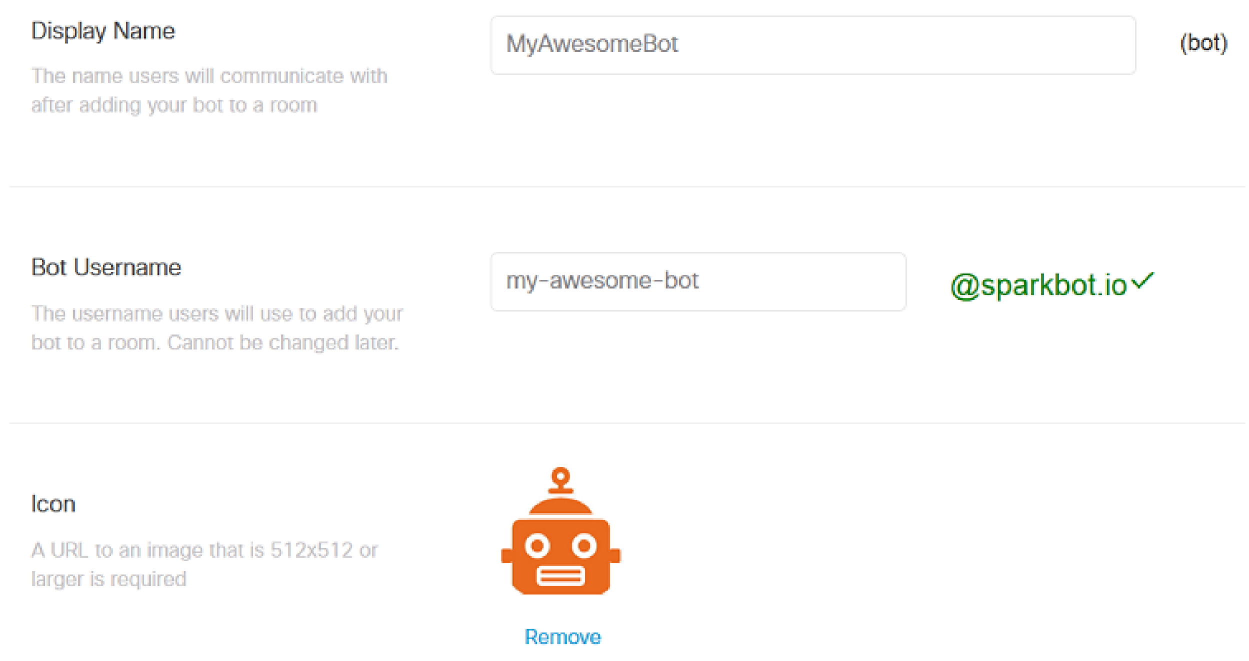


Create rich messaging experiences on Spark with your intelligent Bots.

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 chosen.



Display Name
The name users will communicate with after adding your bot to a room
MyAwesomeBot (bot)

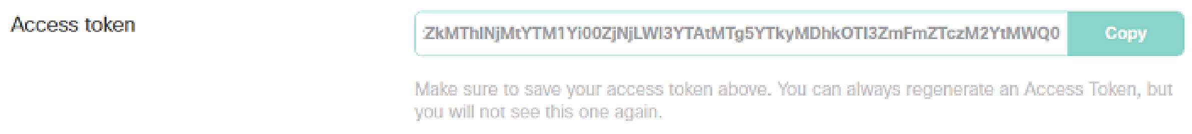
Bot Username
The username users will use to add your bot to a room. Cannot be changed later.
my-awesome-bot @sparkbot.io✓

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.

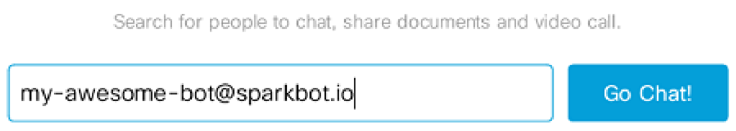
Note that a Cisco Spark Bot access token lasts 100 years. If you ever lose or reveal it, you can come back to this Bot details page and regenerate an access token. The previously issued token will be automatically deprecated.



Access token
ZkMThlNjMtYTM1Yi00ZjNjLWI3YTAtMTg5YTkyMDhkOTI3ZmFmZTczM2YtMWQ0 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: Deploy a bot on Cloud9

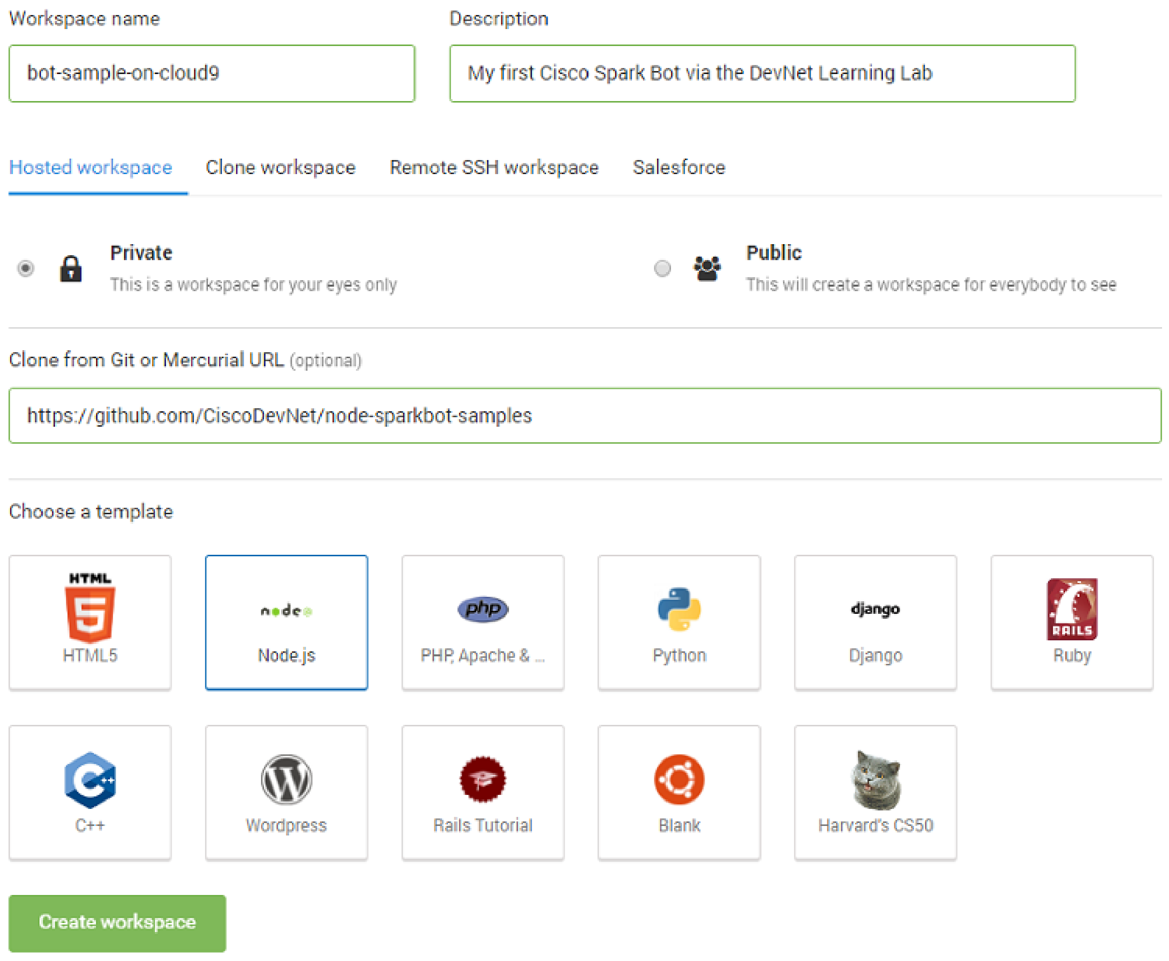
For the sake of this lab, we will clone an existing code sample, and leverage the Cloud9 platform to deploy our own version of a Cisco Spark bot.

Sign up at https://c9.io.

Create a new **private** Workspace. Make sure to create it as private so that the access token - we will set later - does not get disclosed.

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

Make sure to **pick the Node.js template**, and click "Create workspace".



Workspace name: bot-sample-on-cloud9

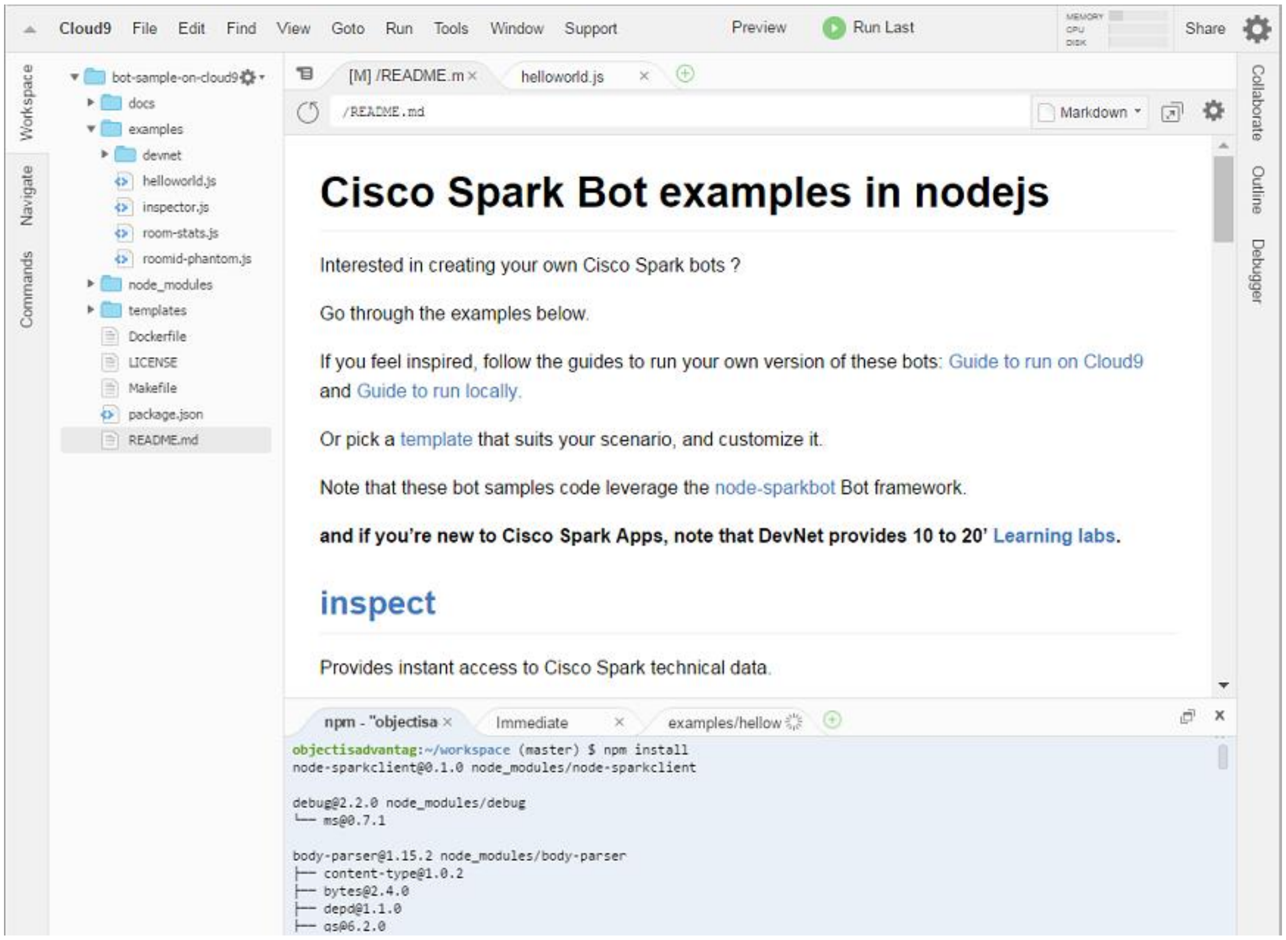
Description: My first Cisco Spark Bot via the DevNet Learning Lab

Hosted workspace: Private (This is a workspace for your eyes only) [✓]

Clone from Git or Mercurial URL (optional): https://github.com/CiscoDevNet/node-sparkbot-samples

Choose a template: Node.js [✓]

After a minute, your workspace will be ready to use. Congrats, you now have at hand a fully functional nodejs development environment with your own copy of a Cisco Spark Bot!



Cisco Spark Bot examples in nodejs

Interested in creating your own Cisco Spark bots ?

Go through the examples below.

If you feel inspired, follow the guides to run your own version of these bots: Guide to run on Cloud9 and Guide to run locally.

Or pick a template that suits your scenario, and customize it.

Note that these bot samples code leverage the node-sparkbot Bot framework.

and if you're new to Cisco Spark Apps, note that DevNet provides 10 to 20' Learning labs.

inspect

Provides instant access to Cisco Spark technical data.

```
objectisadvantag:~/workspace (master) $ npm install
node-sparkclient@0.1.0 node_modules/node-sparkclient

debug@2.2.0 node_modules/debug
└── ms@0.7.1

body-parser@1.15.2 node_modules/body-parser
├── content-type@1.0.2
├── bytes@2.4.0
├── depd@1.1.0
├── qs@6.2.0
```

Go to the shell terminal at the bottom of the IDE, and type "npm install". The nodejs libraries used by our code sample get downloaded to your Cloud9 workspace.

In the File Explorer, look for the "helloworld.js" file in the "examples" directory. Double click the file to make it the current editable source code in your workspace.

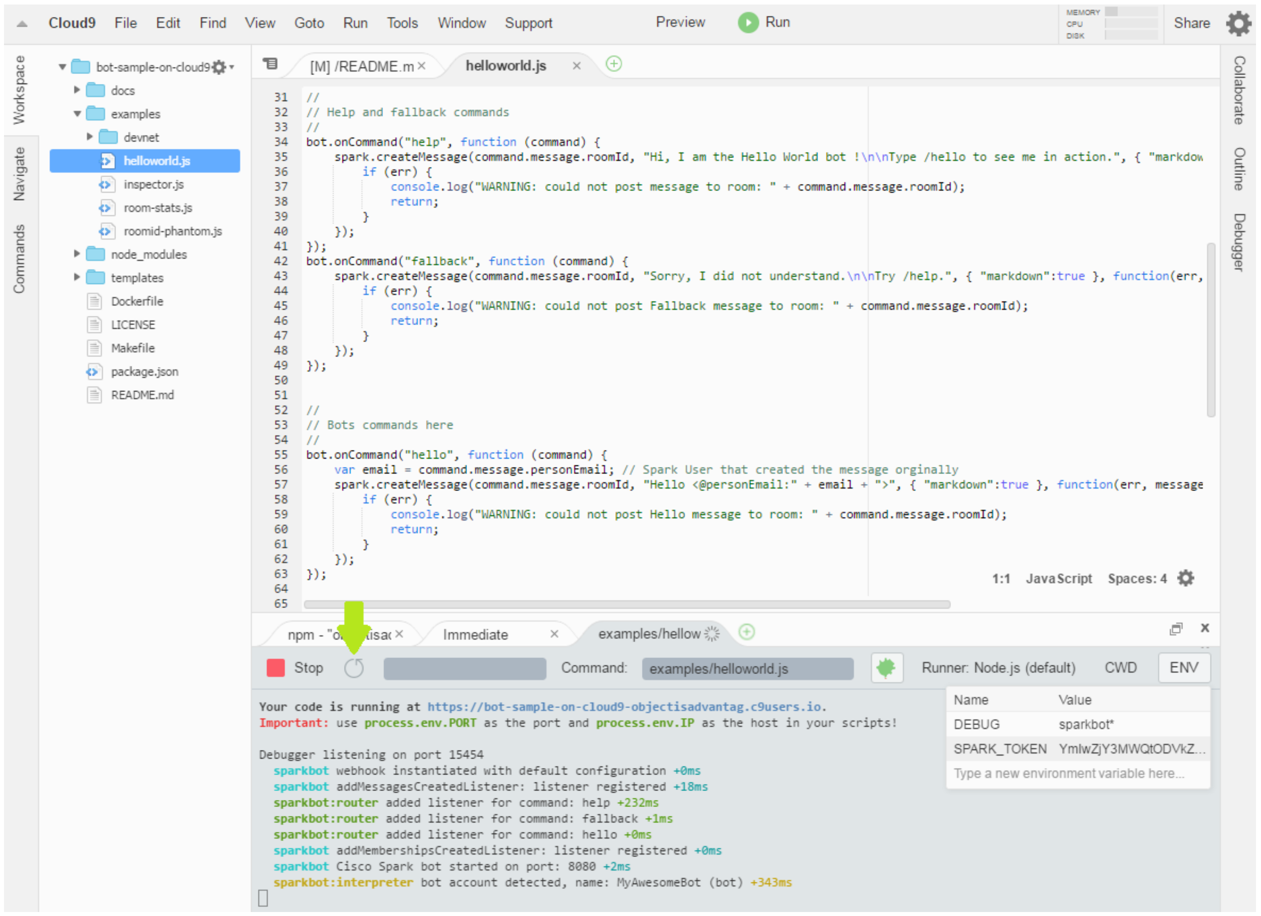
Now click the "Run" button in the top menu to launch an instance of your code. Run

The Cloud9 console displays an error, as we have not specified the Cisco Spark identity under which our code should run. Now comes time to configure our bot.

Click the ENV button at the bottom right of the Cloud9 terminal, and create two variables:

- 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.

Finally, click on the wheel to restart your bot, and check your logs look similar to the snapshot below.



```
//
// Help and fallback commands
//
bot.onCommand("help", function (command) {
    spark.createMessage(command.message.roomId, "Hi, I am the Hello World bot !\n\nType /hello to see me in action.", { "markdow
        if (err) {
            console.log("WARNING: could not post message to room: " + command.message.roomId);
            return;
        }
    });
});
bot.onCommand("fallback", function (command) {
    spark.createMessage(command.message.roomId, "Sorry, I did not understand.\n\nTry /help.", { "markdown":true }, function(err,
        if (err) {
            console.log("WARNING: could not post Fallback message to room: " + command.message.roomId);
            return;
        }
    });
});


//
// Bots commands here
//
bot.onCommand("hello", function (command) {
    var email = command.message.personEmail; // Spark User that created the message orginally
    spark.createMessage(command.message.roomId, "Hello <@personEmail:" + email + ">", { "markdown":true }, function(err, message
        if (err) {
            console.log("WARNING: could not post Hello message to room: " + command.message.roomId);
            return;
        }
    });
});
```

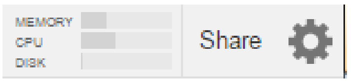
```
Your code is running at https://bot-sample-on-cloud9-objectisadvantag.c9users.io.
Important: use process.env.PORT as the port and process.env.IP as the host in your scripts!

Debugger listening on port 15454
  sparkbot webhook instantiated with default configuration +0ms
  sparkbot addMessagesCreatedListener: listener registered +18ms
  sparkbot:router added listener for command: help +232ms
  sparkbot:router added listener for command: fallback +1ms
  sparkbot:router added listener for command: hello +0ms
  sparkbot addMembershipsCreatedListener: listener registered +0ms
  sparkbot Cisco Spark bot started on port: 8080 +2ms
  sparkbot:interpreter bot account detected, name: MyAwesomeBot (bot) +343ms
```

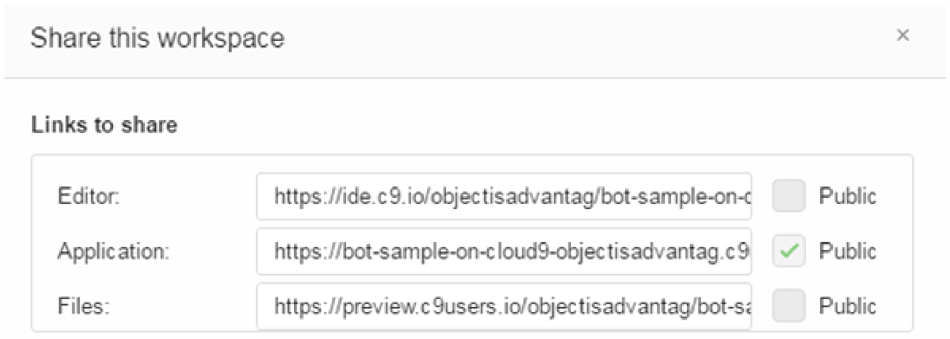
Name	Value
DEBUG	sparkbot*
SPARK_TOKEN	YmlwZjY3MWQtODVkZ...
Type a new environment variable here...	

As we created our Cloud9 workspace as private, we will now need to take an extra step to make our bot accessible from Cisco Spark.

Click on the "Share" link in the upper right corner, close to your avatar.



As the popup shows up, click the "Public" checkbox facing the "Application" links to share.


Share this workspace

Links to share

Editor:	https://ide.c9.io/objectisadvantag/bot-sample-on-c	Public
Application:	https://bot-sample-on-cloud9-objectisadvantag.c9	✓ Public
Files:	https://preview.c9users.io/objectisadvantag/bot-sa	Public

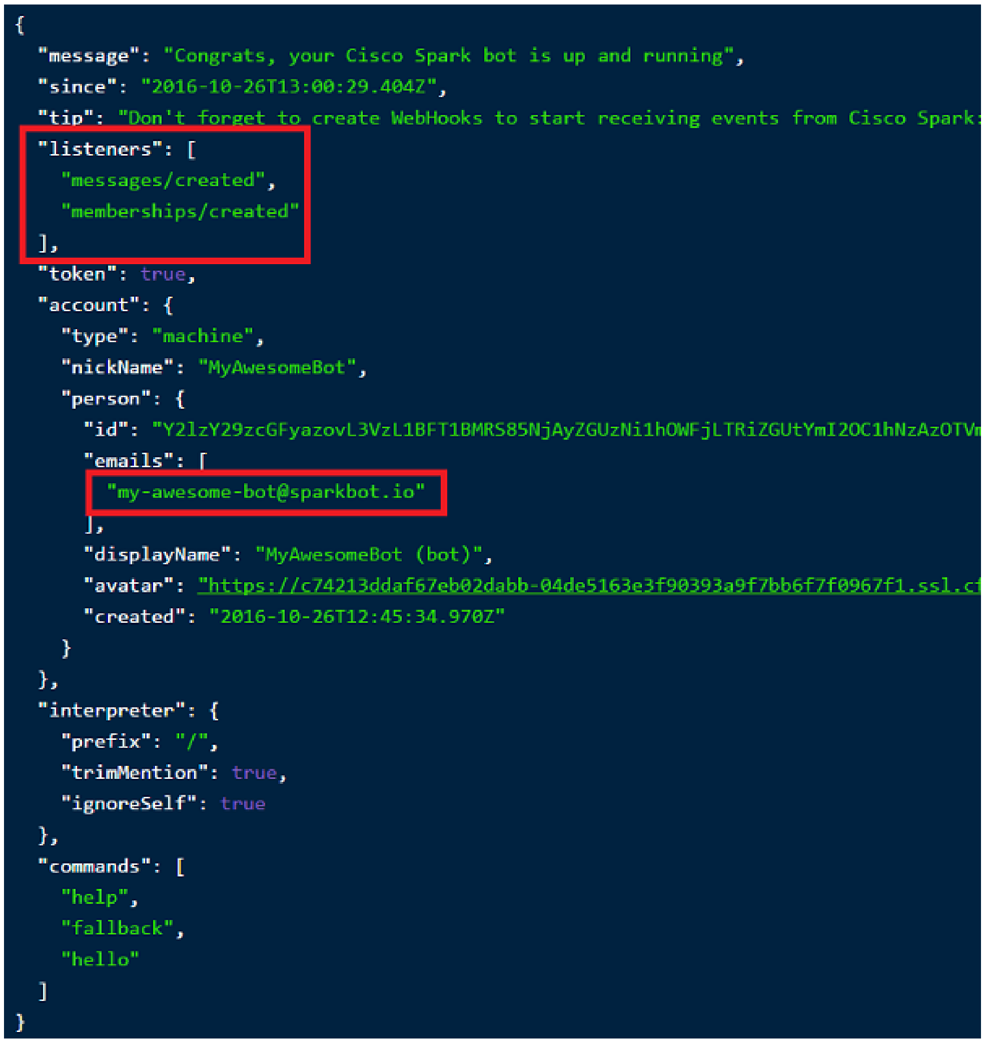
Step 3: Create Webhook events

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

Look for your bot public URL in the Cloud9 terminal. It appears run after the mention "Your code is running at ", and is formatted as: https://replace-with-your-cloud9-project-name.c9users.io/.

Open your bot public URL.

Note that the snapshot below leverages the Chrome plugin “JSON viewer” as mentioned in the pre-requisites.



```
{
  "message": "Congrats, your Cisco Spark bot is up and running",
  "since": "2016-10-26T13:00:29.404Z",
  "tip": "Don't forget to create WebHooks to start receiving events from Cisco Spark:
  "listeners": [
    "messages/created",
    "memberships/created"
  ],
  "token": true,
  "account": {
    "type": "machine",
    "nickName": "MyAwesomeBot",
    "person": {
      "id": "Y2lzY29zcGFyazovL3VzL1BFT1BMRS85NjAyZGUzNi1hOWFjLTRiZGUtYmI2OC1hNzAzOTVm
      "emails": [
        "my-awesome-bot@sparkbot.io"
      ],
      "displayName": "MyAwesomeBot (bot)",
      "avatar": "https://c74213ddaf67eb02dabb-04de5163e3f90393a9f7bb6f7f0967f1.ssl.cf
      "created": "2016-10-26T12:45:34.970Z"
    }
  },
  "interpreter": {
    "prefix": "/",
    "trimMention": true,
    "ignoreSelf": true
  },
  "commands": [
    "help",
    "fallback",
    "hello"
  ]
}
```

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 Cloud9, your bot will return a 200 OK response, with extra configuration and deployment data.

If you look attentively to the JSON data above, 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.

Our next task is to create Cisco Spark Webhooks so that we start receiving these events.

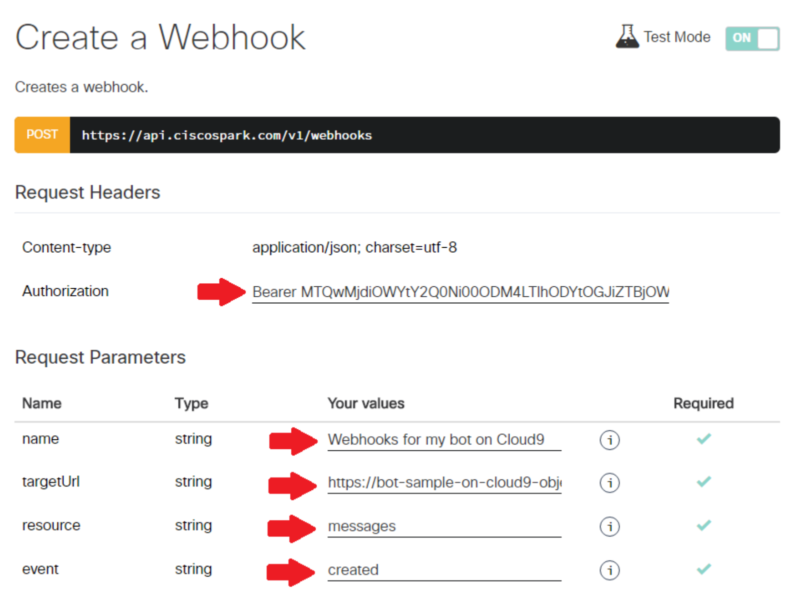
Open the list of Webhooks supported by Cisco Spark, and look for the event entries needed by your bot:

- 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.



Creates a webhook.

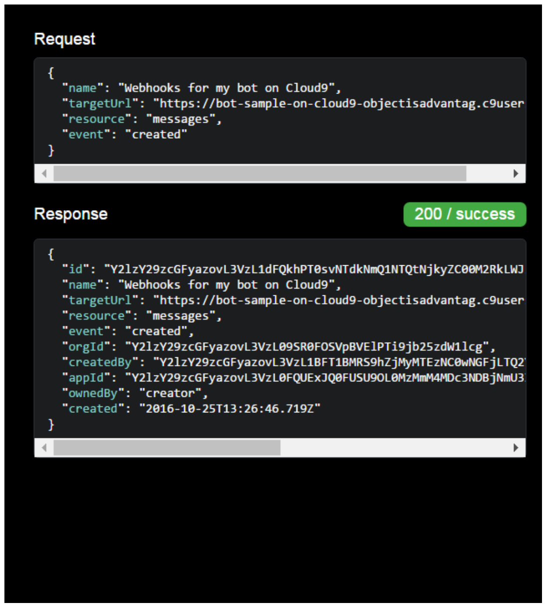
POST `https://api.ciscospark.com/v1/webhooks`

Request Headers

Content-type	application/json; charset=utf-8
Authorization	Bearer MTQwMjdiOWYtY2Q0Ni00ODM4LTlhODYtOGJiZTBjOW

Request Parameters

Name	Type	Your values	Required
name	string	Webhooks for my bot on Cloud9	✓
targetUrl	string	https://bot-sample-on-cloud9-obj	✓
resource	string	messages	✓
event	string	created	✓



Request

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

Response

```
{
  "id": "Y2lzY29zcGFyazovL3VzL1dFQkhPT0svNTdkNmQ1NTQtNjkyZC00M2RkLWJ
  "name": "Webhooks for my bot on Cloud9",
  "targetUrl": "https://bot-sample-on-cloud9-objectisadvantag.c9user
  "resource": "messages",
  "event": "created",
  "orgId": "Y2lzY29zcGFyazovL3VzL09SR0FOSVpBVElPTi9jb25zdW1lcg",
  "createdBy": "Y2lzY29zcGFyazovL3VzL1BFT1BMRS9hZjMyMTEzNC0wNGFjLTQ2
  "appId": "Y2lzY29zcGFyazovL3VzL0FQUExJQ0FUSU9OL0MzMmM4MDc3NDBjNmU3
  "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.

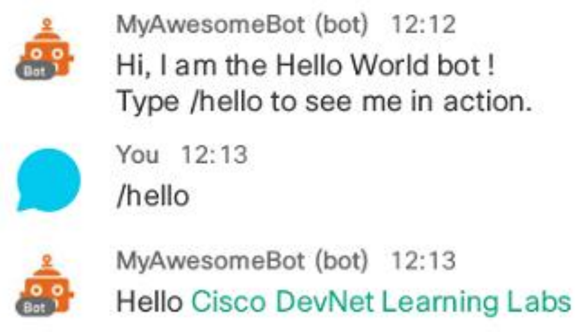
Step 4: Test your Cisco Spark Bot

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!



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

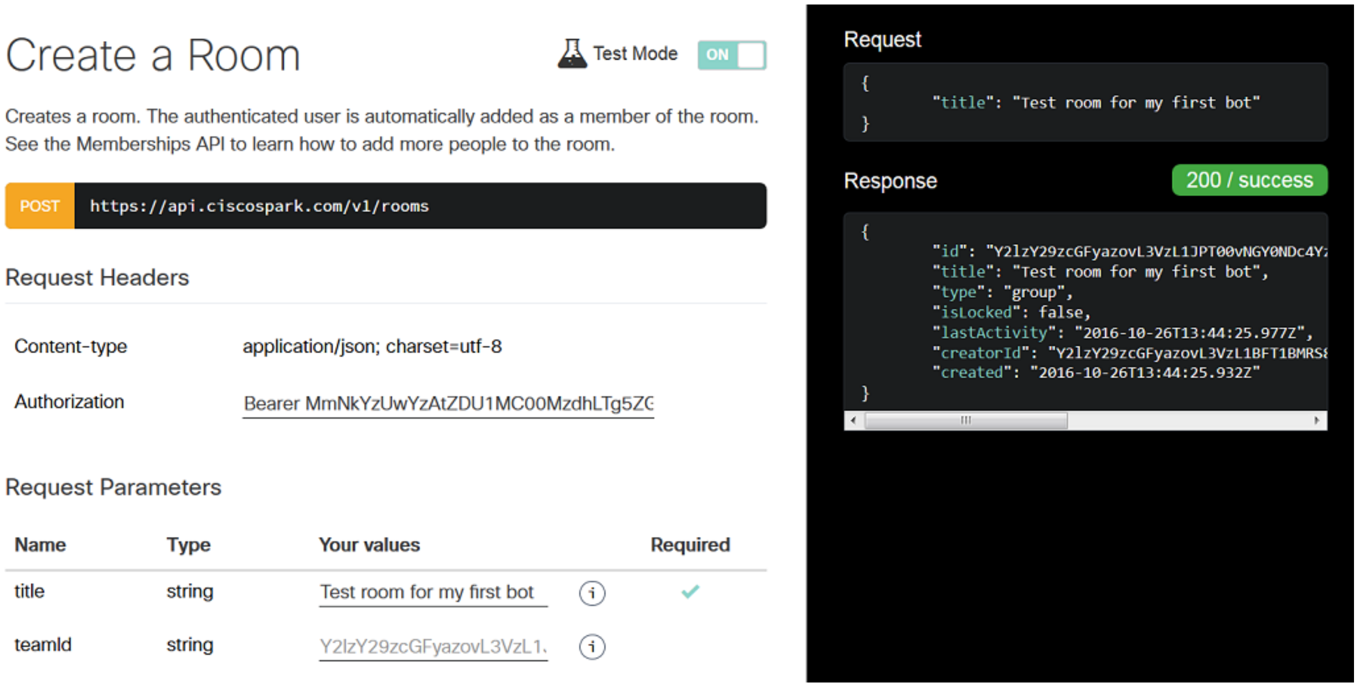
Mention your bot in Group rooms

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 ON

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 MmNkYzUwYzAtZDU1MC00MzdhLTg5ZG

Request Parameters

Name	Type	Your values	Required
title	string	Test room for my first bot	✓
teamId	string	Y2lzY29zcGFyazovL3VzL1.	

Request

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

Response 200 / success

```
{
        "id": "Y2lzY29zcGFyazovL3VzL1JPT00vNGY0NDc4Yz
        "title": "Test room for my first bot",
        "type": "group",
        "isLocked": false,
        "lastActivity": "2016-10-26T13:44:25.977Z",
        "creatorId": "Y2lzY29zcGFyazovL3VzL1BFT1BMRS8
        "created": "2016-10-26T13:44:25.932Z"
}
```

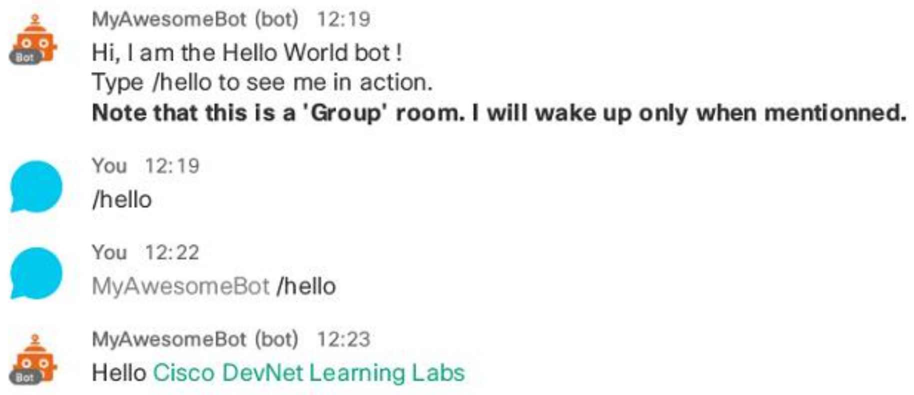
Now, open your favorite Cisco Spark client, and reach to the newly created room.

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 mentionned.

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!

As you code is running on Cloud9, 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…