

think 2018

IBM



Lab Center – Hands-on Lab

Session 2473

Session Title IBM Connections Customizer : from Zero to Hero

Stefano Pogliani, IBM, stefano.pogliani@fr.ibm.com

Martti Garden, IBM, martti.garden@de.ibm.com

Padraig Edwards, IBM, padraig.edwards@ie.ibm.com

Martin Donnelly, IBM, marti_donnelly@ie.ibm.com

Table of Contents

Disclaimer	3
Introduction.....	6
Icons	6
Code, Solutions, Examples.....	6
Documentation	7
Lab 1 The typical “Hello World” script	8
What we want to achieve	8
Preparation Activities	9
The Greeting	9
The Description	9
IBM Connections page loading mechanism	10
Preparing the Environment	12
Creating your script	13
Creating your extension.....	19
Making your extension available	27
Activating the extension	28
Lab 2 Playing with CSS.....	31
What we want to achieve	31
Two different approaches	32
The CSS File	32
Using Javascript injection	33
Preparation Activities	33
Creating your script	35
Creating your extension.....	39
Making your extension available	46
Activating the extension	47
Using CSS Declaration	50
Preparation Activities	50
Making your extension available	58
Activating the extension	59
Heading.....	62

We Value Your Feedback!	62
Preparing the Environment	65
Subheading Level 2	65
Heading	66
We Value Your Feedback!	67

Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed “as is” without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts.

In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.”

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all**

warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

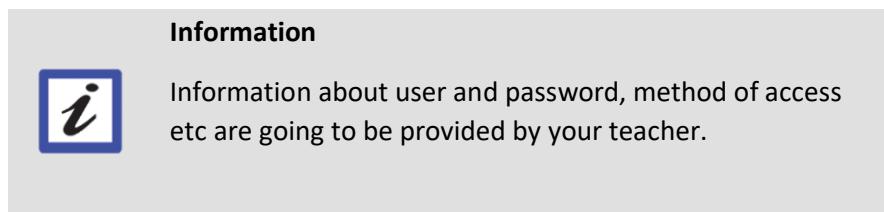
IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

© 2018 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Introduction

This workshop was prepared as step by step instruction. It should lead you through IBM Connections Customizer components and its capabilities. All included exercises are going to support you in creating your first IBM Connections Customizer scripts.



All exercises are fulfilled with screen shots for easier navigation. However, there may be some small differences based on your environment, browser version, language or user account used.

Icons

The following symbols appear in this document at places where additional guidance is available.

Icon	Purpose	Explanation
	Important!	This symbol calls attention to a particular step or command. For example, it might alert you to type a command carefully because it is case sensitive.
	Information	This symbol indicates information that might not be necessary to complete a step but is helpful or good to know.
	Trouble-shooting	This symbol indicates that you can fix a specific problem by completing the associated troubleshooting information.

Code, Solutions, Examples

A special Github repository has been created where solutions, examples and the code or the graphics to be used are made available to all the students. We will reference this Github repository throughout this document as "**Reference Repository**".

This Github repository is available at this address: <https://github.com/stefanopog/CustomizerThinkLab>

stefanopog / CustomizerThinkLab

Code Issues Pull requests Projects Wiki Insights Settings

IBM Connections Customizer Labs for Think2018

Add topics

6 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

stefanopog Create README.md Latest commit ddaa009 a minute ago

Docs Create README.md a minute ago

Lab1 ok 3 days ago

Lab2 ok 4 days ago

README.md Initial commit 4 days ago

README.md

CustomizerThinkLab

You may want to open a tab of your browser on this address in order to use the material. Feel free to clone this repository for better analyzing the code and the examples.

Documentation

The text of these exercises, together with other useful documentation, is stored in the Docs folder of the Github repository mentioned above.

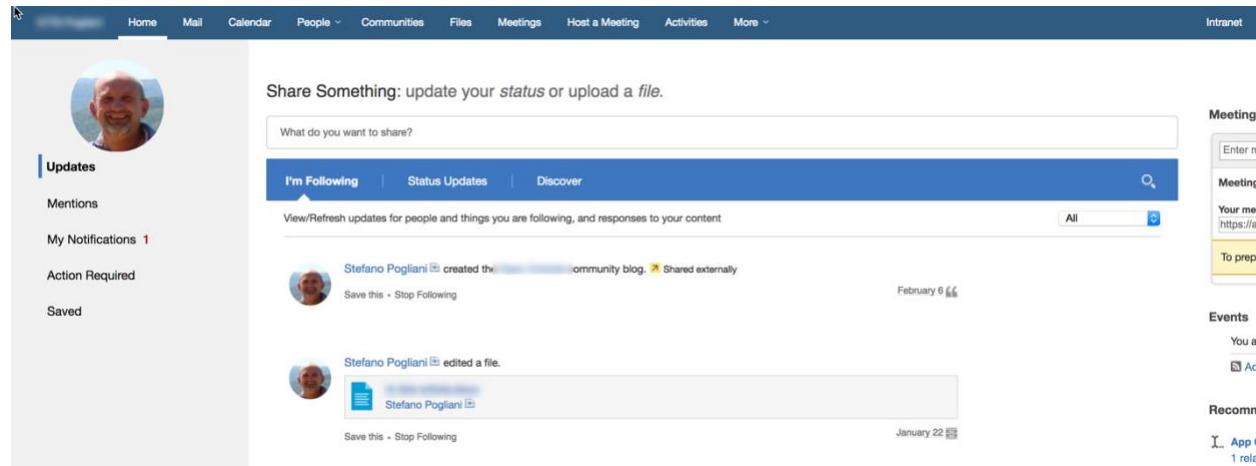
Lab 1 The typical “Hello World” script

This is a very simple exercise, very similar in the approach to the typical “Hello World” exercises you have certainly encountered in other labs.

The main goal we want to reach through this exercise is to guide you through the process of creating an IBM Connections Customizer extension; the extension itself is very simple because the focus will be on the process.

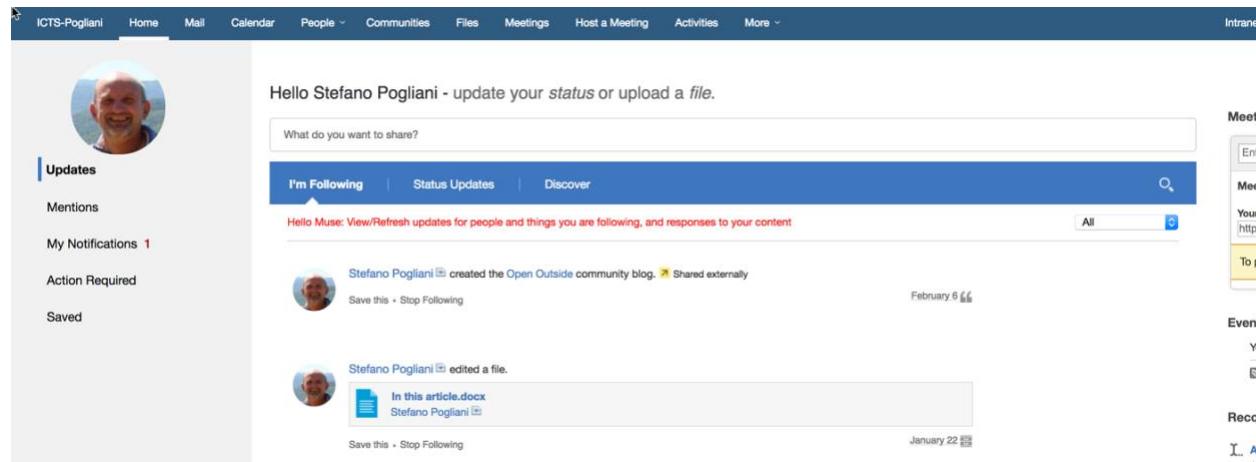
What we want to achieve

When accessing the traditional IBM Connections Cloud Homepage, you are likely to see something like this:



The screenshot shows the IBM Connections homepage with a dark blue header bar containing links for Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, and More. On the right side of the header, there is a vertical sidebar titled "Intranet" which includes sections for Meeting, Events, and Recommendations. The main content area features a user profile picture and a "Share Something" input field. Below this is a blue navigation bar with tabs for "I'm Following", "Status Updates", and "Discover". A search icon is located on the right of the bar. The main feed displays two items from Stefano Pogliani: one where he created a community blog and another where he edited a file. Each item includes a "Save this" link, a "Stop Following" link, and a timestamp (February 6 or January 22).

In our exercise, we will modify the look-and-feel of this page to be the following one:



The modified screenshot shows the same homepage layout but with a custom message. The "Share Something" string has been replaced by "Hello Stefano Pogliani - update your status or upload a file.". The rest of the interface, including the sidebar and activity feed, remains identical to the original screenshot.

So, we simply changed the “Share Something” string with the greeting “Hello Stefano Pogliani” and we changed the color of the font for the string below the blue header of the Activity Stream (and added a little text to it).

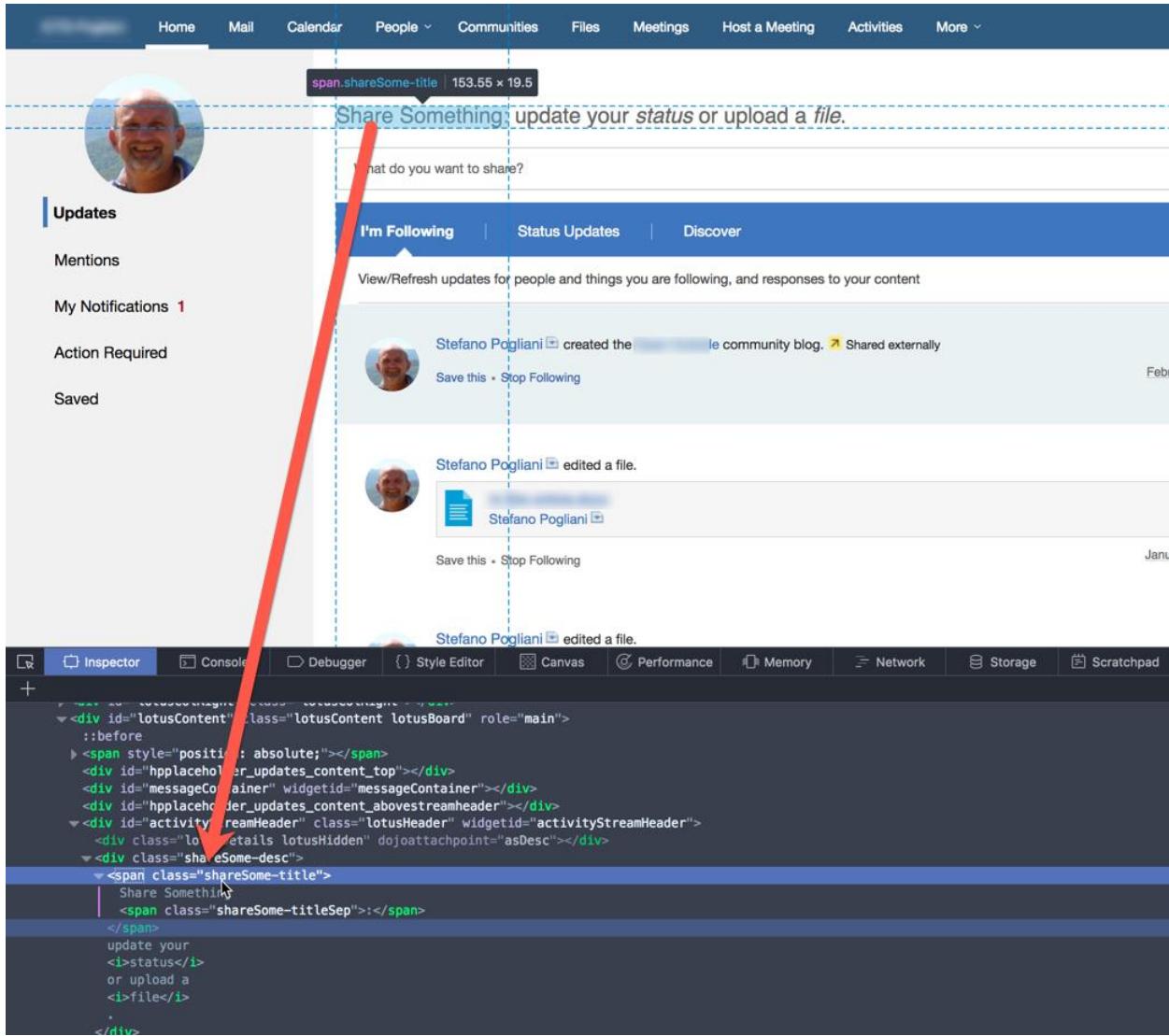
Preparation Activities

In order to do those modifications, we first need to understand which DOM elements we need to modify.

Our best friends here are the “Developer Tools” you find in your browser.

The Greeting

Which is the DOM element in the page holding the string “Share Something” ?



Using the Browser Developer Tools, we quickly find that the “Share Something” string is the content of a `` element whose class is “`shareSome-title`” .

The Description

The description string below the Blue Bar on the top of the Activity Stream is easily found also:

The screenshot shows the IBM Connections homepage with a user profile picture and navigation menu. The main area displays an activity stream with updates from Stefano Pogliani. A red arrow highlights the text "View/Refresh updates for people and things you are following, and responses to your content" in the UI, which corresponds to the highlighted `` element in the browser's DOM inspector.

```

<div id="activityStreamMain" class="lotusWidgetBody">
  <div class="lotusAccess" role="status" aria-live="assertive" aria-relevant="all" aria-atomic="true"></div>
  <div class="streamHeaderWrapper" dojoattachpoint="streamHeaderWrapper" style="width: auto;" aria-hidden="false"></div>
  <div class="filterArea" dojoattachpoint="pagingTop" style="margin-top: 0px;">
    <div class="filterAreaInner">
      <span dojoattachpoint="statusBottomNode"></span>
      <span class="lotusLeft" dojoattachpoint="placeholderHeaderLeft"></span>
      <div class="lotusLeft" dojoattachpoint="placeholderHeaderCenter" aria-label="View/Refresh updates for people and things you are following, and responses to your content">
        <span class="lotusRight"></span>
        <span id="asDesc" class="icStream-labelInner" dojoattachpoint="asDescription">
          View/Refresh updates for people and things you are following, and responses to your content
        </span>
      </div>
    </div>
  </div>

```

Using the Browser Developer Tools, we quickly find that the string is the content of a `` element whose id is “`asDesc`”.

IBM Connections page loading mechanism

IBM Connections Customizer injects into the page one or more scripts according to the definition of the extension(s). The IBM Connections Customizer scripts that are loaded, are declared as `<script>` elements at the bottom of the HTML code of the page that is rendered.

```

806
807
808
809 <script type="text/javascript">window.NREUM||(NREUM={});NREUM.info={"errorBeacon":"bam.nr-data.net","licenseKey":"0e69123577","agent":"","beacon":"bam.nr-data.net","appl
810
811
812
813 </div><script type='text/javascript' src='/files/customizer/'></script><script type='text/javascript' src='/files/customizer/'>
814 </div><script type='text/javascript' src='/files/customizer/'></script>
815

```

If the script will execute immediately, it is likely that it would execute before the page is fully built in your browser.

This means that we cannot assume that the HTML elements that we need to modify are already rendered when the script executes (during the load of the page). Thus, we need to implement a strategy

that would allow the real changes made by the script to happen once all the required elements in the page are there.

We think that it is fair enough to assume that we could start modifying the two **** elements introduced in the previous sections when the ActivityStream starts loading: at that point, the two **** elements are likely to be there:

The screenshot shows the IBM Connections interface with a red arrow highlighting the `<div>` element with the class `lotusStreamTopLoading` in the browser's developer tools DOM panel. This element is located within the `<div>` with the class `connectViews`.

So, our strategy is going to be the following:

- Wait until the page will start loading the items in the ActivityStream
This happens once the `<div>` element whose class is “`loaderMain lotusHidden`” is rendered on the page.
In order to be sure we target that very element, we can say that the `<div>` element must be a child of the other `<div>` element whose class is “`lotusStreamTopLoading`”.
The “`dojo`” query for this is

```
dojo.query(".lotusStreamTopLoading div.loaderMain.lotusHidden")
```

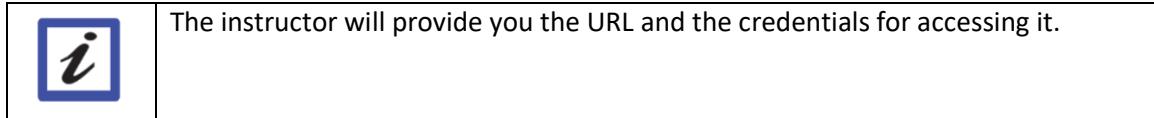


We could have used the “`id`” property of the child `<div>` element in order to be reasonably sure that we would have targeted the one we wanted. Unfortunately, you will see that the “`id`” ends with a “`_0`”. This implies that the value of the “`id`” property is calculated at runtime by IBM Connections and we cannot be sure there will not be one ending with “`_1`” or other strings.

- Once that element is rendered, we can modify the “textContent” property of the two elements and the font color of the second one.

Preparing the Environment

You have been invited as a Contributor to the Github repository associated to this organization.



You will arrive to a page similar to this one.

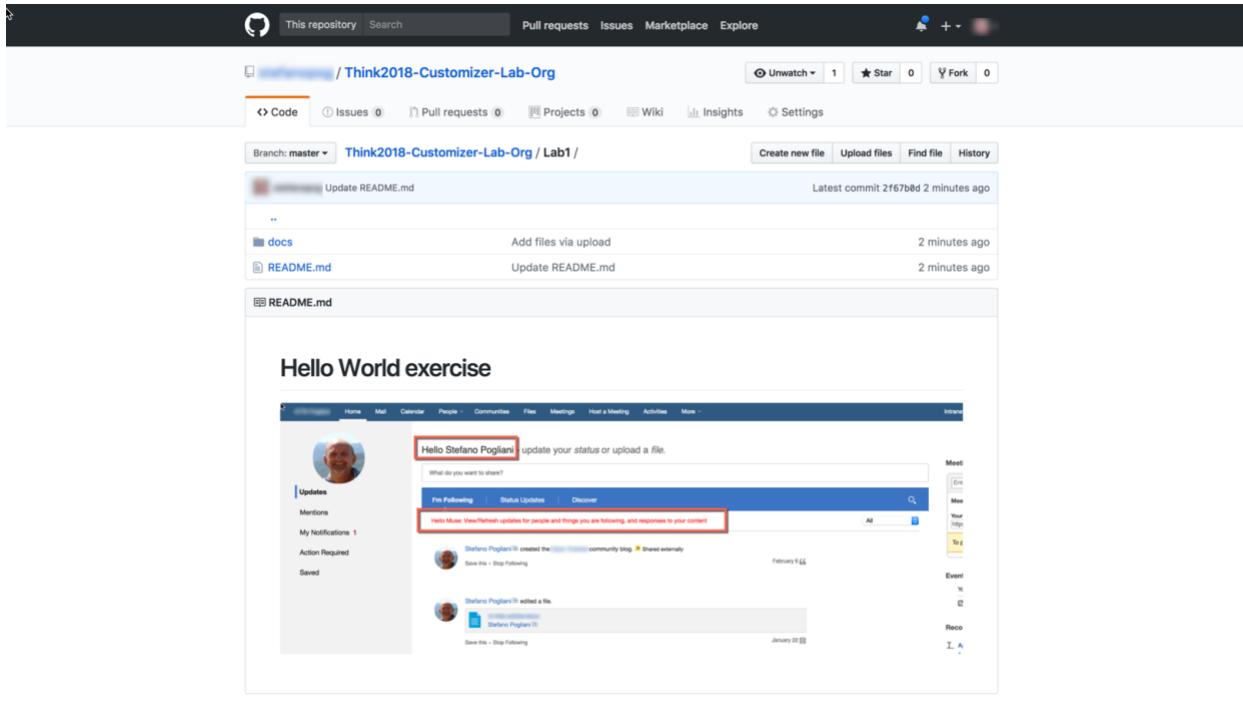
The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 11 commits, 1 branch, 0 releases, and 1 contributor. The latest commit was just now. The commit history includes:

- Update README.md by Lab1 (just now)
- Create README.md by Lab2 (9 minutes ago)
- Create README.md by Lab3 (8 minutes ago)
- Create README.md by Lab4 (8 minutes ago)
- Update README.md by README.md (10 minutes ago)

The README file contains the following content:

```
123456789
123456789 - Think2018 Customizer Lab Organization
All the scripts that will be activated for the Think2018 IBM Connections Customizer Organization
```

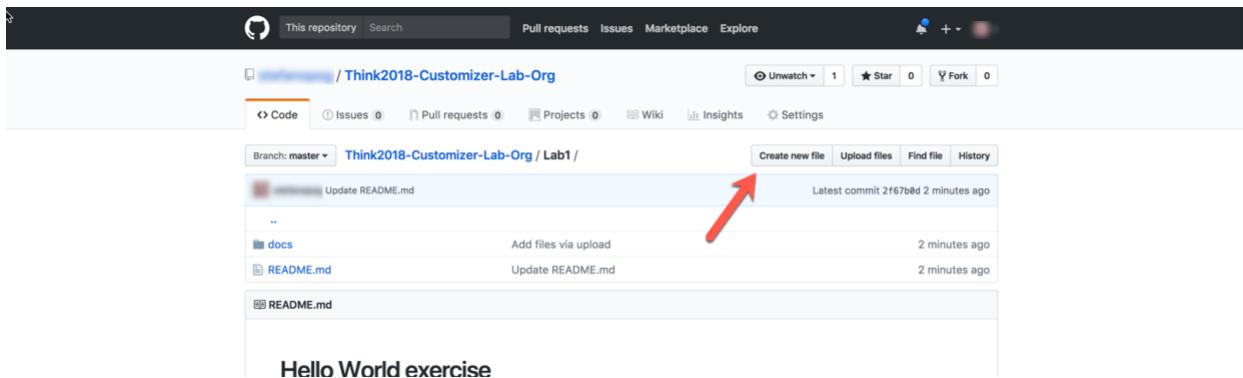
Click on the “Lab1” item and you will access this page:



Now you are ready to code!

Creating your script

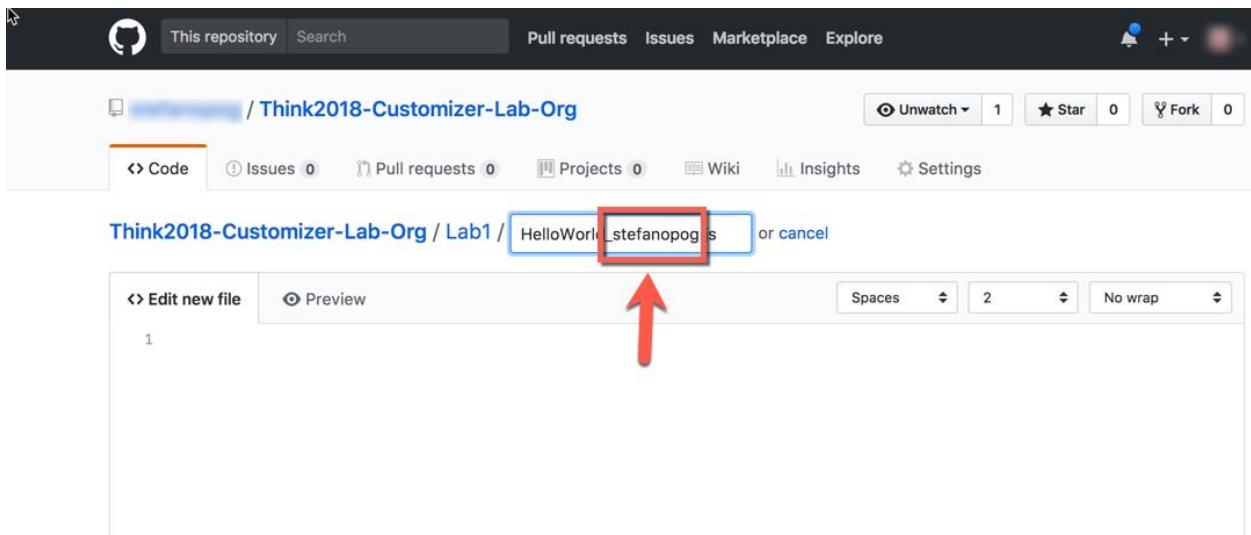
Create the Javascript file that will contain the code for your extension. Click on “Create File” as shown in the following image:



Now, you have to enter the name of the file. Since there are many people doing the exercise at the same time, we need to avoid conflict in filenames. For this reason, enter the following for the filename:

HelloWorld_<identifier>.js

<identifier> is a unique label that will be provided to you by the instructor (it is going to be the first part of the email address associated with your IBM Connections Cloud account for this lab).



Let's now proceed step-by-step in building our code together.

- Changing the Greeting.

As we [previously said](#), we need to modify the content of a `` element whose class is `"shareSome-title"`

The code that we need to use is the following:

```
//  
// Change the Greetings String  
// Note that the username of the currently Logged in user can be obtained by means of the  
// global variable "lconn.homepage.userName". This variable is defined by the  
// IBM Connections Home page  
//  
var thisUser = lconn.homepage.userName;  
dojo.query("span.shareSome-title")[0].textContent = "Hello " + thisUser + " -";
```

We use the “`dojo.query`” function to retrieve the `` element. Since the “`query`” returns an



In production code, we would certainly need to validate that the “`query`” actually returned the result that we were looking for.

array, we need to select the very first element of this array.

- Changing the Description.

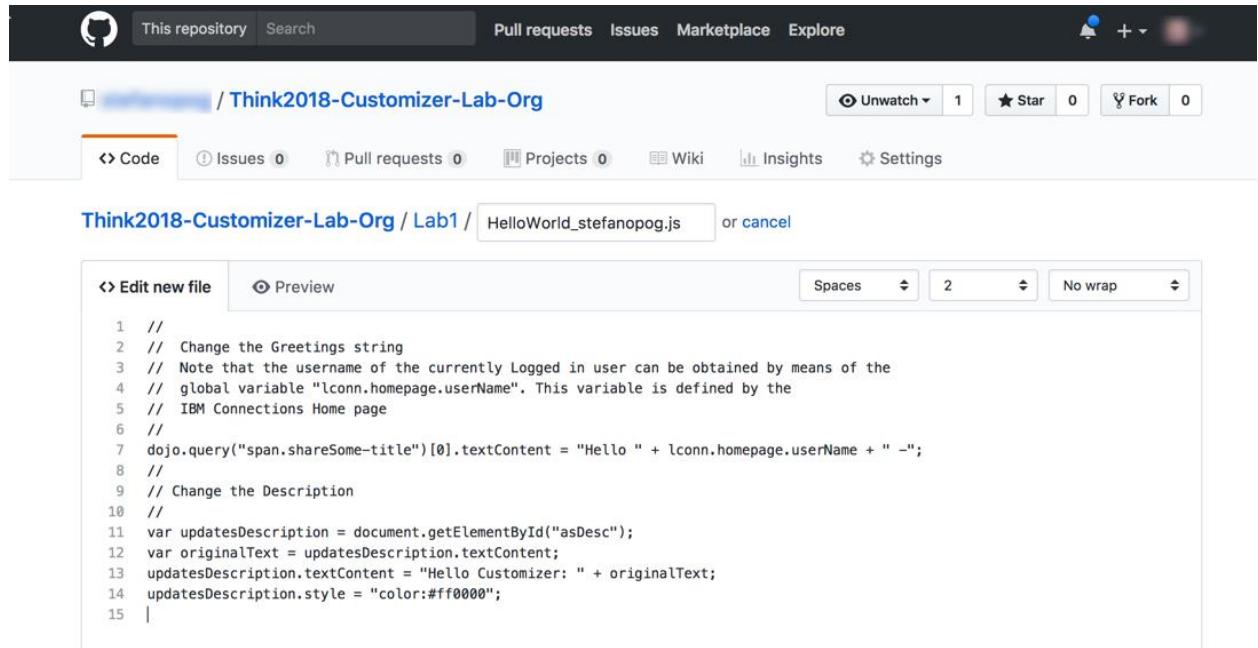
As we [previously said](#), we need to modify the content of a `` element whose id is `“asDesc”`.

The code that we need to use is the following:

```
//  
// Change the Description  
//  
var updatesDescription = document.getElementById("asDesc");  
var originalText = updatesDescription.textContent;  
updatesDescription.textContent = "Hello Customizer: " + originalText;  
updatesDescription.style = "color:#ff0000";
```

In this case we just used the “document.getElementById” function to retrieve the whose id is “asDesc”. We could have used the “dojo.byId” function.

- At this point our script looks like this:



The screenshot shows a GitHub repository interface. The top navigation bar includes links for "This repository", "Search", "Pull requests", "Issues", "Marketplace", and "Explore". On the right side of the header are icons for "Unwatch" (with a count of 1), "Star" (with a count of 0), "Fork" (with a count of 0), and a user profile icon. Below the header, the repository path "Think2018-Customizer-Lab-Org / Think2018-Customizer-Lab-Org" is displayed, along with tabs for "Code", "Issues 0", "Pull requests 0", "Projects 0", "Wiki", "Insights", and "Settings". A search bar at the top right contains the text "HelloWorld_stefanopog.js" and a "cancel" button. The main content area is a code editor with the following JavaScript code:

```
1 //  
2 // Change the Greetings string  
3 // Note that the username of the currently Logged in user can be obtained by means of the  
4 // global variable "lconn.homepage.userName". This variable is defined by the  
5 // IBM Connections Home page  
6 //  
7 dojo.query("span.shareSome-title")[0].textContent = "Hello " + lconn.homepage.userName + "-";  
8 //  
9 // Change the Description  
10 //  
11 var updatesDescription = document.getElementById("asDesc");  
12 var originalText = updatesDescription.textContent;  
13 updatesDescription.textContent = "Hello Customizer: " + originalText;  
14 updatesDescription.style = "color:#ff0000";  
15 |
```

We have now to make sure that the code we just entered would be executed when the elements we are modifying are properly rendered (as [mentioned previously](#)).

To implement this behavior, let's suppose that we have a function, called “waitFor”, which waits for the condition to be met and, then, will execute our code.

The way we would describe this in Javascript is by using the callback mechanism: the code we just entered becomes the body of a callback function executed by our “waitFor” once the condition is met.

At high level :

```
//  
// wait for the required elements are rendered  
//  
waitFor(function() {  
    ... Our code here ....  
}, ".lotusStreamTopLoading div.loadMain.lotusHidden");
```

So, change the code in your editor to look like what is shown in the following picture:

```

1 waitFor(function(){
2     //
3     // Change the Greetings string
4     // Note that the username of the currently Logged in user can be obtained by means of the
5     // global variable "lconn.homepage.userName". This variable is defined by the
6     // IBM Connections Home page
7     //
8     dojo.query("span.shareSome-title")[0].textContent = "Hello " + lconn.homepage.userName + " -";
9     //
10    // Change the Description
11    //
12    var updatesDescription = document.getElementById("asDesc");
13    var originalText = updatesDescription.textContent;
14    updatesDescription.textContent = "Hello Customizer: " + originalText;
15    updatesDescription.style = "color:#ff0000";
16    updatesDescription.innerHTML = originalText;
17 }, ".lotusStreamTopLoading div.loaderMain.lotusHidden");

```

- Of course, we now need to provide the implementation of our “waitFor” function, right ?
You can copy the code here and paste before the first line of the script:

```

var waitFor = function(callback, elXPath, elxpathRoot, maxInter, waitTime) {
    if (!elXPath) var elXPathRoot = dojo.body();
    if (!maxInter) var maxInter = 10000; // number of intervals before expiring
    if (!waitTime) var waitTime = 1; // 1000=1 second
    if (!elXPath) return;

    var waitInter = 0; // current interval
    var intId = setInterval(function(){
        if ( ++waitInter < maxInter && !dojo.query(elXPath,elXPathRoot).length) return;
        clearInterval(intId);
        if ( waitInter >= maxInter ) {
            console.log("***** WAITFOR [" + elXPath + "] WATCH EXPIRED!!! interval " + waitInter + " (max:" + maxInter + ")");
        } else {
            console.log("***** WAITFOR [" + elXPath + "] WATCH TRIPPED AT interval " + waitInter + " (max:" + maxInter + ")");
            callback();
        }
    }, waitTime);
};

```

This code simply implements a polling mechanism using the Javascript “setInterval” function. We do not use the last 3 parameters of the function and the code defaults them to some value. What is to be noticed is the way in which “waitFor” uses the **callback** function (which is actually the code we wrote in the previous part of the exercise).

At the end, the code in your editor should look like this:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The file 'HelloWorld_stefanopog.js' is open. The code is as follows:

```

1  var waitFor = function(callback, elXPath, elXPathRoot, maxInter, waitTime) {
2      if (!elXPathRoot) var elXPathRoot = dojo.body();
3      if (!maxInter) var maxInter = 10000; // number of intervals before expiring
4      if (!waitTime) var waitTime = 1; // 1000=1 second
5      if (!elXPath) return;
6      var waitInter = 0; // current interval
7      var intId = setInterval(function(){
8          if ( ++waitInter < maxInter && !dojo.query(elXPath,elXPathRoot).length) return;
9
10         clearInterval(intId);
11         if (waitInter >= maxInter) {
12             console.log("**** WAITFOR [" + elXPath + "] WATCH EXPIRED!!! interval " + waitInter + " (max:" + maxInter);
13         } else {
14             console.log("**** WAITFOR [" + elXPath + "] WATCH TRIPPED AT interval " + waitInter + " (max:" + maxInter);
15             callback();
16         }
17     }, waitTime);
18 };
19 waitFor(function(){
20     //
21     // Change the Greetings string
22     // Note that the username of the currently Logged in user can be obtained by means of the
23     // global variable "lconn.homepage.userName". This variable is defined by the
24     // IBM Connections Home page
25     //
26     dojo.query("span.shareSome-title")[0].textContent = "Hello " + lconn.homepage.userName + " -";
27     //
28     // Change the Description
29     //
30     var updatesDescription = document.getElementById("asDesc");
31     var originalText = updatesDescription.textContent;
32     updatesDescription.textContent = "Hello Customizer: " + originalText;
33     updatesDescription.style = "color:#ff0000";
34 }, ".lotusStreamTopLoading div.loaderMain.lotusHidden");
35

```

- We are still missing one point though.

In the code we wrote we assumed that the **dojo toolkit** would be available in the page in order for your script to use it. We are confident that the IBM Connections page will load the **dojo toolkit** at a certain point, but we cannot assume that it will be loaded by the time we first use it (look at the line 8 in the previous picture: our “waitFor” function uses the **dojo toolkit** to poll the presence of the element...).

We certainly do not want our script to load the **dojo toolkit** another time.... So we need to wait for it to be ready on the page.

Fortunately the **dojo toolkit** provides a standard mechanism to deal with this point: the “dojo/domReady!” plugin (described here <https://dojotoolkit.org/reference-guide/1.10/dojo/domReady.html>).

So, we will simply need to wrap the code we wrote up until now in this way:

```

if (typeof(dojo) != "undefined") {
    require(["dojo/domReady!"], function () {
        try {
            ... all our previous code ....
        } catch(e) {
            alert('exception occurred in HelloWorld : ' + e);
        }
    });
}

```

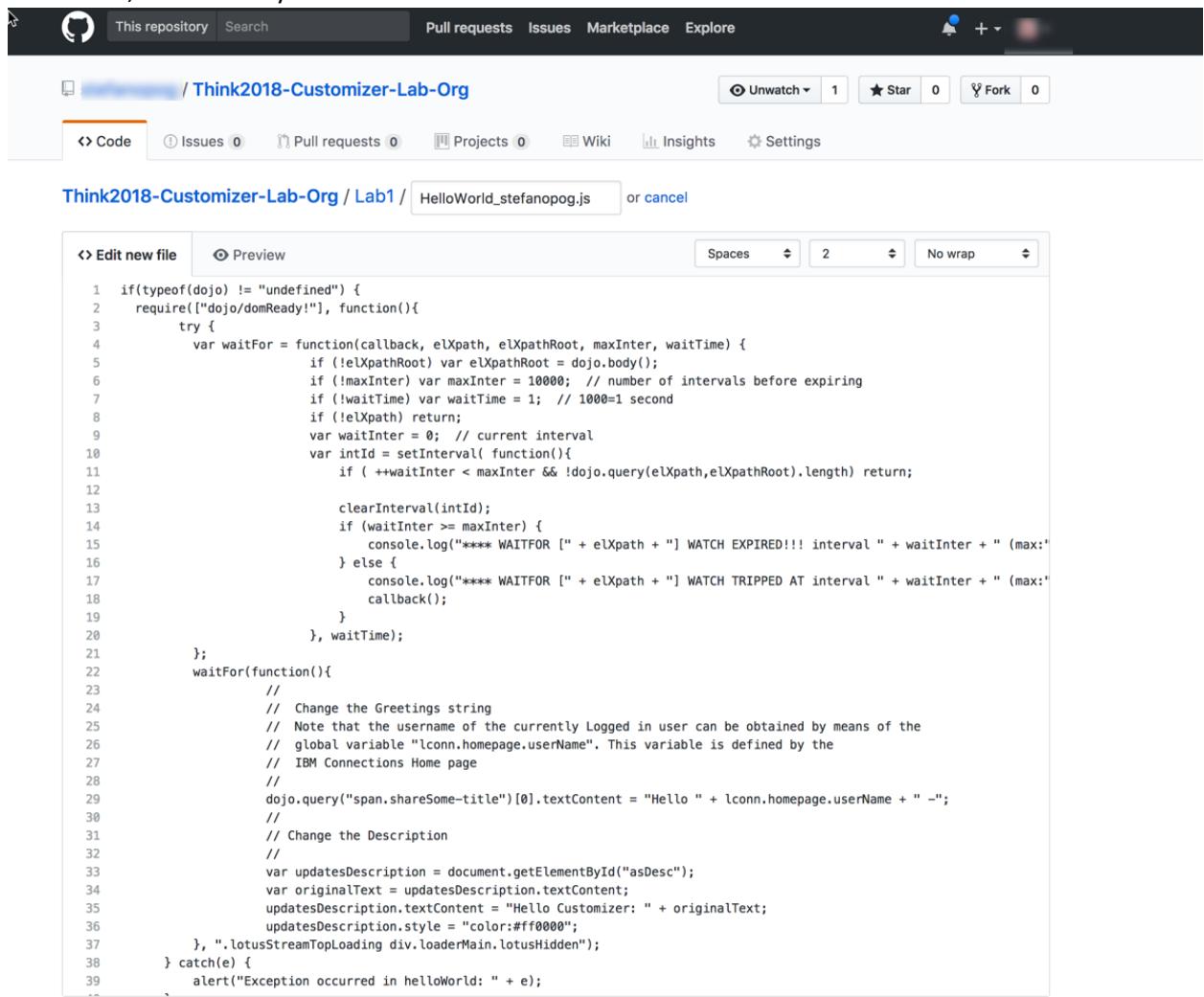
```

    }
});
}

```

We have introduced a “try... catch” statement to globally catch any exception we may encounter.

At the end, the code in your editor should look like this:

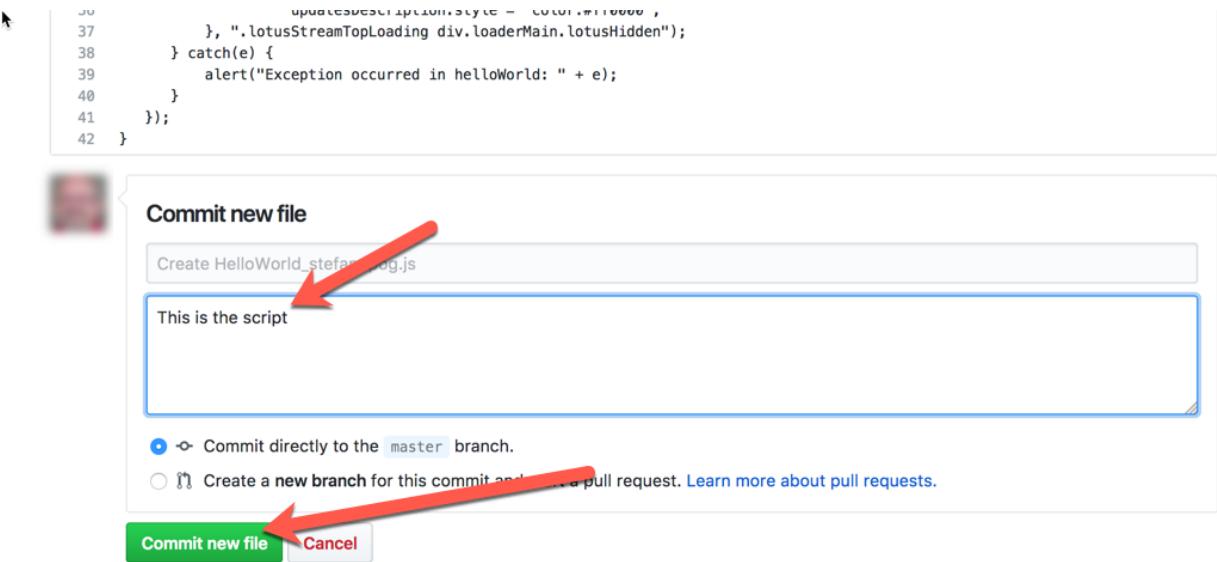


```

1  if(typeof(dojo) != "undefined") {
2      require(["dojo/domReady!"], function(){
3          try {
4              var waitFor = function(callback, elXPath, elXPathRoot, maxInter, waitTime) {
5                  if (!elXPathRoot) var elXPathRoot = dojo.body();
6                  if (!maxInter) var maxInter = 10000; // number of intervals before expiring
7                  if (!waitTime) var waitTime = 1; // 1000=1 second
8                  if (!elXPath) return;
9                  var waitInter = 0; // current interval
10                 var intId = setInterval( function(){
11                     if ( ++waitInter < maxInter && !dojo.query(elXPath,elXPathRoot).length) return;
12
13                     clearInterval(intId);
14                     if (waitInter >= maxInter) {
15                         console.log("**** WAITFOR [" + elXPath + "] WATCH EXPIRED!!! interval " + waitInter + " (max:"+
16                     } else {
17                         console.log("**** WAITFOR [" + elXPath + "] WATCH TRIPPED AT interval " + waitInter + " (max:"+
18                         callback();
19                     }
20                 }, waitTime);
21             };
22             waitFor(function(){
23                 //
24                 // Change the Greetings string
25                 // Note that the username of the currently Logged in user can be obtained by means of the
26                 // global variable "lconn.homepage.userName". This variable is defined by the
27                 // IBM Connections Home page
28                 //
29                 dojo.query("span.shareSome-title")[0].textContent = "Hello " + lconn.homepage.userName + " -";
30                 //
31                 // Change the Description
32                 //
33                 var updatesDescription = document.getElementById("asDesc");
34                 var originalText = updatesDescription.textContent;
35                 updatesDescription.textContent = "Hello Customizer: " + originalText;
36                 updatesDescription.style = "color:#ff0000";
37             }, ".lotusStreamTopLoading div.loaderMain.lotusHidden");
38         } catch(e) {
39             alert("Exception occurred in helloWorld: " + e);
        }
    }
}

```

- You can now commit your code to the repository. It is always good practice to add, as a comment, what you did.



- Your script is now safely recorded in the Github repository.

A screenshot of a GitHub repository page for `Think2018-Customizer-Lab-Org / Lab1`. The repository has 1 star, 0 forks, and 0 issues/pull requests/projects. The latest commit is `ffe185e just now`.

The commit details show the creation of a file named `HelloWorld_stefanopog.js`. A red arrow points from this file entry to the commit message. The commit message is `Create HelloWorld_stefanopog.js`.

Other files listed in the commit are `docs`, `README.md`, and another `README.md`.

Hello World exercise

A screenshot of an IBM Connections Intranet feed. A user named `Hello Stefano Poglian` has updated their status or uploaded a file. The status message is `Hello Muser: View/Refresh updates for people and things you are following, and responses to your content`. A red box highlights this message.

The feed also shows other activity items like `Updates`, `Mentions`, `My Notifications 1`, and `Action Required`.

Creating your extension

Whilst the extension can be directly created in the IBM Connections Cloud Administration panel, we suggest that you save the extension definition as a JSON file in the same Github repository.



In this way, the extension may be used in other situations and may become part of the overall documentation of the project

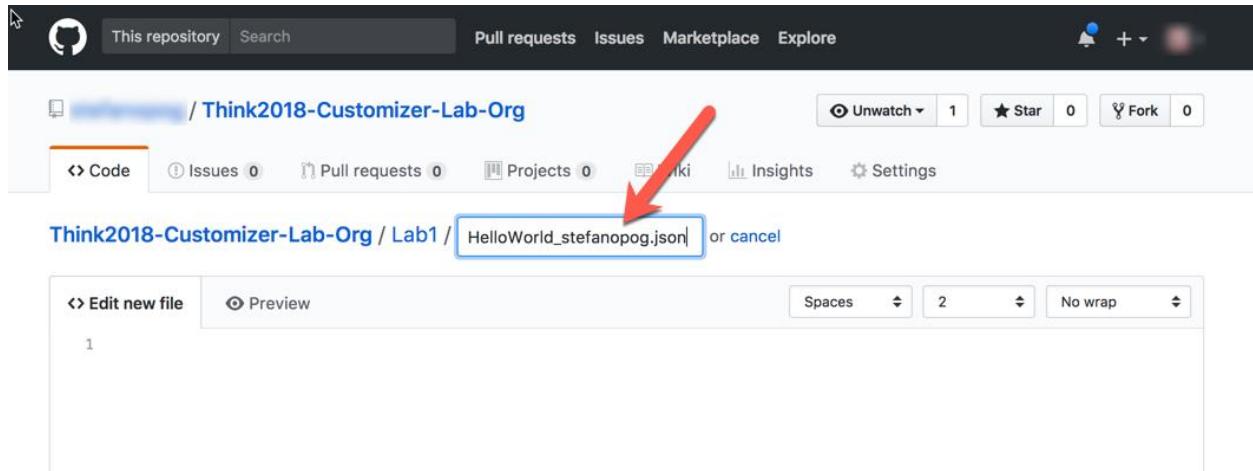
In order to create the extension, we need to create a new file inside our Github repository. The file will follow this naming convention

`HelloWorld_<identifier>.json`

Where `<identifier>` is a unique label that will be provided to you by the instructor (it is going to be the first part of the email address associated with your IBM Connections Cloud account for this lab). In this way, everybody will immediately understand that this is the extension definition associated with the `HelloWorld_<identifier>` script.

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org / Lab1'. At the top right, there is a navigation bar with links for 'Unwatch', 'Star', 'Fork', and 'Settings'. Below the navigation bar, there are buttons for 'Create new file', 'Upload files', 'Find file', and 'History'. A red arrow points to the 'Create new file' button. The main area of the page displays a list of files: 'docs' (Add files via upload, an hour ago), 'HelloWorld_stefanopog.js' (Create HelloWorld_stefanopog.js, just now), and 'README.md' (Update README.md, an hour ago). Below this list, there is a section titled 'Hello World exercise' which includes a screenshot of the IBM Connections interface showing a status update from 'Hello Muse'.

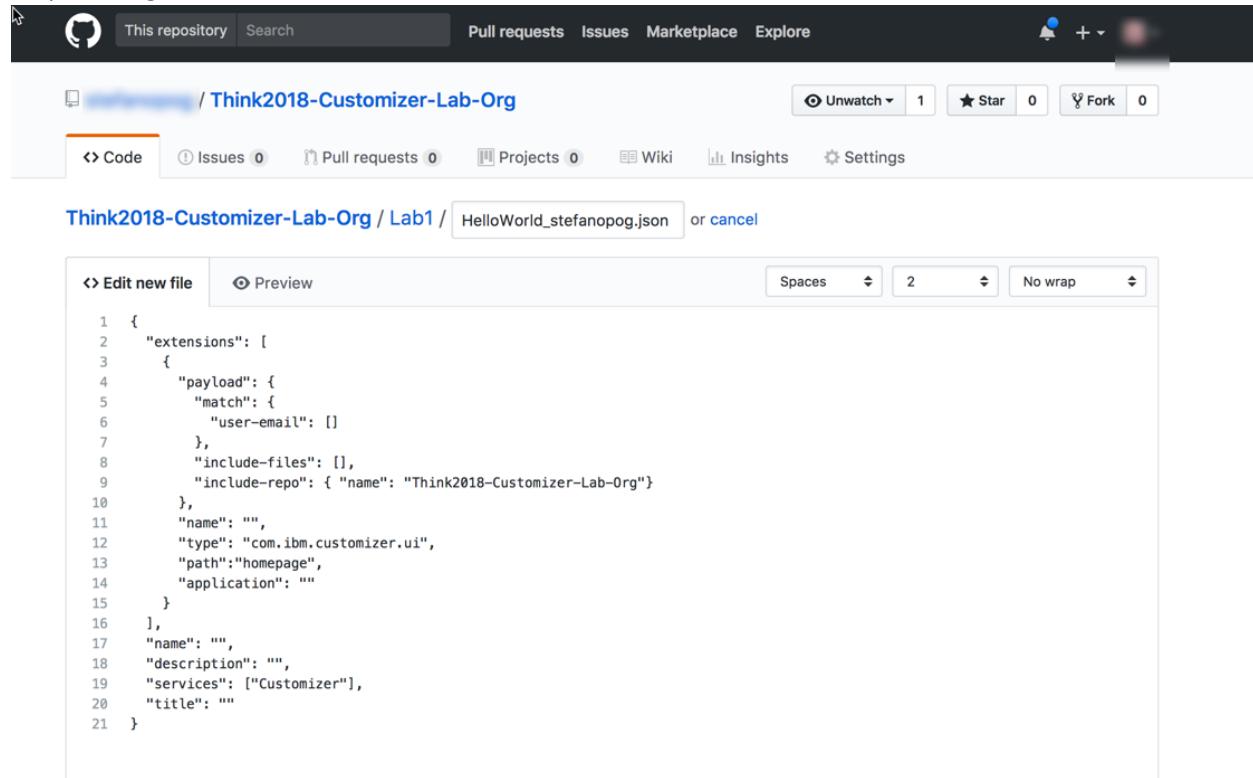
And, then,



Let's first copy this empty template inside our new json file (a copy of this template is available as the Lab1/emptyTemplate.json under "**Reference Repository**"):

```
{  
  "extensions": [  
    {  
      "payload": {  
        "match": {  
          "user-email": []  
        },  
        "include-files": [],  
        "include-repo": {  
          "name": "Think2018-Customizer-Lab-Org"  
        }  
      },  
      "name": "",  
      "type": "com.ibm.customizer.ui",  
      "path": "homepage",  
      "application": ""  
    }  
  ],  
  "name": "",  
  "description": "",  
  "services": ["Customizer"],  
  "title": ""  
}
```

So you will get this result:



The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 1 unwatched star and 0 forks. The 'Code' tab is selected, showing a file named 'HelloWorld_stefanopog.json'. The code content is as follows:

```
1  {
2    "extensions": [
3      {
4        "payload": {
5          "match": {
6            "user-email": []
7          },
8          "include-files": [],
9          "include-repo": { "name": "Think2018-Customizer-Lab-Org" }
10         },
11        "name": "",
12        "type": "com.ibm.customizer.ui",
13        "path": "homepage",
14        "application": ""
15      }
16    ],
17    "name": "",
18    "description": "",
19    "services": ["Customizer"],
20    "title": ""
21 }
```

Some of the values of the JSON descriptor are already provided. A complete documentation for the attributes is available at this url :

<https://github.com/ibmcnxdev/customizer/blob/master/docs/IBMConnectionsCustomizer.pdf>

- “include-repo”: (line 8)
This MUST be the name of the Github repository that we are using. It is the “root” directory that the IBM Connections Customizer engine will use in order to retrieve the scripts that the extension uses
- “type”: (line 13)
This is a string whose value “com.ibm.customizer.ui” will tell IBM Connections Cloud which type of IBM Connections Customizer service will be used by the extension
- “path”: (line 14)
The value of “homepage” will instruct the IBM Connections Customizer engine to add the scripts referenced by the extension only to the “homepage” service of IBM Connections
- “services”: (line 19)
This is a string whose value “Customizer” will tell IBM Connections Cloud which type extension it needs to register in the Application Registry.

Let's now fill the other values that are required:

- “user-email”: (line 5)
We can restrict the injection of the scripts associated to the extension only to users whose email address will match the string.
Since we do not want to interfere with other people working at the same time on the same IBM

Connections Cloud organization, we need to provide a value for this attribute. The value needs to be the email address you have been provided by the instructor to log into IBM Connections Cloud (enclosed in double-quotes). So something like :

```
"user-email": ["stefanopog@think2018-Customizer.com"]
```



In real life, you may not need to use this attribute if the extension you are creating will be used by **all the people** in your organization.
See the [documentation](#) for discovering all the possibilities associated to the "match" clause.

- "include-files": (line 7)

This is where you will actually reference the script that you created earlier in the exercise. The path to reference the script is relative to the Github repository you are using (see the "include-repo" clause described above).

So in your situation you need to provide the following value :

```
"include-files": ["Lab1>HelloWorld_<identifier>.js"]
```

- "name": (line 12)

this is the name of the Extension.

You can use the following:

```
"name": "HelloWorld Ext Lab1 <identifier>"
```

- "name": (line 17)

This is the name of the Application we are going to declare to IBM Connections. Typically, an "Application" can declare one or more extensions (in our example, we have one extension only associated with the application).

This name **must be unique** for the IBM Connections Organization, since it is the name that will be recognized by the IBM Connections Application registry. You can use the following name:

```
"name": "HelloWorld App Lab1 <identifier>"
```

- "application": (line 15)

This entry (which is within the "extension") actually needs to reference the correct "Application". So, the value you will provide for this attribute **MUST BE the same** as the value you provided for the "name" attribute of the Application (line 17). You can use the following name:

```
"application": "HelloWorld App Lab1 <identifier>"
```

- "description": (line 18)

This is a free text description of what the Application (and its Extensions) provide. You can use the following:

"description": "Hello World Homepage Customization from <identifier>"

- "title": (line 20)

This is the title that will appear in the IBM Connections Cloud application registry. You can use the following title:

"title": "<identifier> Hello World"

Once the Organization admin will use your JSON file to declare the extension, this is how it will appear in the IBM Connections Cloud Application Registry:

The screenshot shows the IBM Connections Cloud Administration interface. On the left, there's a sidebar with various settings like Personal, My Account Settings, User Accounts, Organization Account Settings, Partitions, Subscriptions, Announcements, Internal Apps, Order History, and Organization Extensions. The Organization Extensions section is currently selected. In the main area, there's a heading 'Info: To create classic Organization Extensions, click here'. Below it, there's a 'New App' button and a list of apps. One specific app is highlighted with a red box and labeled 'stefanopog Hello World'. Red arrows point to the 'title' and 'description' fields of this app. The 'description' field contains the text 'Service: Customizer Hello World Homepage Customization from stefanopog'. Other apps in the list include 'Digital Communication Sample', 'Guest Model Package', 'Mobile User Activity', 'CRMCustomizer', and 'Mobile User Activity'. The top navigation bar includes Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, More, Intranet, Intranet2, Admin, and other icons.

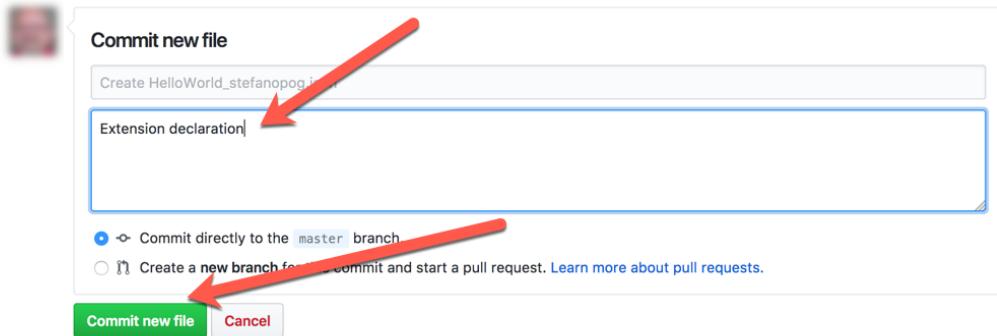
At this point, the JSON file you are editing in Github will look like this:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'HelloWorld_stefanopog.json'. The page displays a single commit from 'stefanopog' with the message 'Update HelloWorld_stefanopog.json'. The commit was made a minute ago with the SHA '8ab9cee'. There is one contributor listed. The code editor shows the JSON content:

```
1  {
2    "extensions": [
3      {
4        "payload": {
5          "match": {
6            "user-email": ["stefanopog@think2018-Customizer.com"]
7          },
8          "include-files": ["Lab1/HelloWorld_stefanopog.js"],
9          "include-repo": { "name": "Think2018-Customizer-Lab-Org" }
10        },
11        "name": "HelloWorld Ext Lab1 stefanopog",
12        "type": "com.ibm.customizer.ui",
13        "path": "homepage",
14        "application": "HelloWorld App Lab1 stefanopog"
15      }
16    ],
17    "name": "HelloWorld App Lab1 stefanopog",
18    "description": "Hello World Homepage Customization from stefanopog",
19    "services": ["Customizer"],
20    "title": "stefanopog Hello World"
21 }
```

You can Commit the change to the Github repository:

```
16 ],
17 "name": "HelloWorld App Lab1 stefanopog",
18 "description": "Hello World Homepage Customization from stefanopog",
19 "services": ["Customizer"],
20 "title": "stefanopog Hello World"
21 }
```



And your Repository will look like this:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 1 unwatched star and 0 forks. The 'Code' tab is selected. A red arrow points to the file 'HelloWorld_stefanopog.json' in the file list.

File List:

- ..
- docs
- HelloWorld_stefanopog.json
- HelloWorld_stefanopog.json (highlighted by a red arrow)
- README.md

File Details for 'HelloWorld_stefanopog.json':

File	Action	Time
docs	Add files via upload	3 hours ago
HelloWorld_stefanopog.json	Create HelloWorld_stefanopog.json	2 hours ago
HelloWorld_stefanopog.json (highlighted)	Create HelloWorld_stefanopog.json	just now
README.md	Update README.md	3 hours ago

The screenshot shows the IBM Internal Intranet homepage. A status update from 'Hello Muse' is highlighted with a red box. The update reads: 'Hello Muse: View/Refresh updates for people and things you are following, and responses to your content.'

Making your extension available

At this point, everything is ready to be put in production. This is what the [documentation](#) states:

- Share your repo with IBM – [add "ibmcndev" as a collaborator](#)
- IBM ([ibmcnxdev](#)) then creates a fork of your repository under [github.com/ibmcnxdev](#) and grants you read access by default.
- You can continue to work on your extension using your original repo for your source code activity, but once you are ready to deliver to IBM Cloud you must issue a [pull request](#) to IBM.
- IBM merges your pull request once acceptance criteria are met.
- Upon merge, the repo files are automatically pushed to IBM Customizer via a [webhook](#).
- Rinse & repeat starting at Step (c) for extension updates.

Steps **a.** and **b.** have already been done for you before you started the exercise.

Now, you need to execute **Step c.**

Go back to the root directory of your Github repository and click on the “Pull Request” button:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. At the top, there are tabs for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. Below the tabs, there's a summary bar with metrics: 13 commits, 1 branch, 0 releases, and 1 contributor. A red arrow points to the 'New pull request' button, which is located in the middle of the summary bar. Below this bar, there's a list of files: 'Create HelloWorld_stefanopog.json', 'Lab1', 'Lab2', 'Lab3', 'Lab4', 'README.md', and another 'README.md'. The 'README.md' file has a note indicating it was updated 7 minutes ago. At the bottom of the page, there are buttons for 'Create new file', 'Upload files', 'Find file', and 'Clone or download'.

Activating the extension

Now your extension has been validated. The validation process (from **IBMCNDEV**), if successful, will make the files you created in this exercise available to IBM Connections Cloud and to IBM Connections Customizer.

The only thing that is missing is to actually define the Extension in the IBM Connections Cloud Application Registry. This is the step that informs IBM Connections Customizer of what to do with the scripts that you just created.

The IBM Connections Cloud organization administrator needs to be given the JSON file which contains the extension (the one you created [here](#)).

He will go to the IBM Connections Application Registry and will declare a new extension:

The screenshot shows the IBM Connections Cloud Administration interface. On the left, there's a sidebar with sections like 'Personal' (My Account Settings, User Accounts, Organization Account Settings, Partitions, Subscriptions, Announcements, Internal Apps, Order History), 'Organization Extensions' (Connections Mobile App Management, Chat and Meetings, Apps), and 'System Settings' (Security, Theme, IBM SmartCloud Notes, Profile Customization, Directory Options, File Download History). The main area has tabs 'Apps' and 'All Apps'. Below them is a 'New App +' button with a plus sign. A red arrow points to this button. To the right, there are several app cards, including 'Guest Model Package', 'IBM User Activity', and 'IBM System Configuration'.

A new screen is shown:

The screenshot shows the 'App Editor' screen. The sidebar remains the same. The main area has tabs 'Back to Apps' and 'Code Editor'. The 'Code Editor' tab is selected. It contains a 'Warning: Import will overwrite editor contents.' message and two buttons: 'Cancel' and 'Save'. A red arrow points upwards from the 'Save' button towards the JSON code area. Another red arrow points upwards from the JSON code area towards the 'Save' button. The JSON code is as follows:

```
1 {  
2   "name": "",  
3   "title": "",  
4   "description": "",  
5   "services": false,  
6   "extensions": false  
7 }
```

The IBM Connections Cloud organization admin will replace the content pointed by the right arrow (1) with the content of the JSON file you created and, then, **saves** (2) the extension.

You can now go to the homepage and see the final result:

The screenshot shows a Microsoft SharePoint intranet homepage. At the top, there's a navigation bar with links for Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, More, and Intranet. On the left, there's a sidebar with sections for Updates, Mentions, My Notifications (1), Action Required, and Saved. The main content area has a header "Hello Stefano Pogliani - update your status or upload a file." Below it is a search bar and tabs for I'm Following, Status Updates, and Discover. A red box highlights a message: "Hello Muse: View/Refresh updates for people and things you are following, and responses to your content". There are two items listed under "I'm Following": 1) "Stefano Pogliani created the [redacted] community blog." (Shared externally) on February 6. 2) "Stefano Pogliani edited a file." on January 22. To the right, there are vertical columns for Meet, Event, and Reco.

This ends Lab 1.
Congratulations!

Lab 2 Playing with CSS

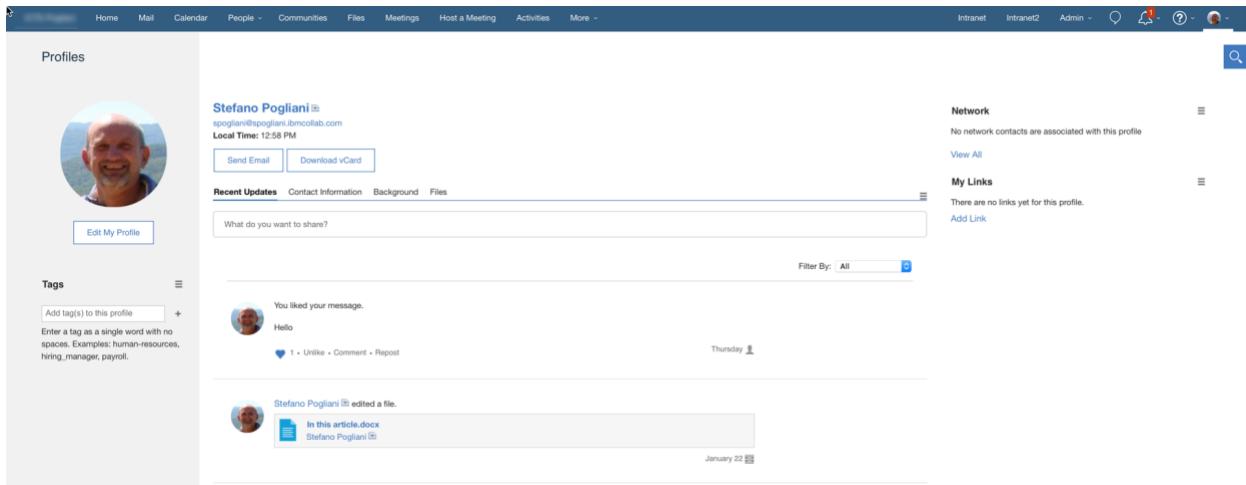
In the previous exercise we played with some DOM Elements using IBM Connections Customizer. We just saw how, programmatically, we could modify the style of an HTML element (when we colored the "Description" in red).

What if we would like to apply extensive changes to the CSS of one or more IBM Connections pages? Probably using Javascript code to wrap CSS directives is going to be tedious and error-prone. It would be much easier if we could compile all of our style changes into a CSS file and have IBM Connections Customizer apply that style with a single command.

This is the goal of this second exercise. Once again, the extension is really simple by itself, and the focus is really on showing how a CSS file can be used by an IBM Connections Customizer extension.

What we want to achieve

When accessing the traditional IBM Connections Cloud Profile page, you are likely to see something similar to this:

A screenshot of the IBM Connections Cloud Profile page for Stefano Pogliani. The top navigation bar includes Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, More, Intranet, Intranet2, Admin, and a search bar. The main profile area shows a large circular profile picture of Stefano Pogliani, his name, email (spogliani@spogliani.ibmcollab.com), and the local time (12:58 PM). Buttons for 'Send Email' and 'Download vCard' are present. Below this are tabs for Recent Updates, Contact Information, Background, and Files. A text input field asks 'What do you want to share?'. The 'Recent Updates' section shows a message from Stefano: 'You liked your message.' followed by 'Hello'. There is a 'Filter By: All' dropdown. The 'Network' section indicates no network contacts are associated with this profile. The 'My Links' section shows a message: 'There are no links yet for this profile.' with a 'Add Link' button. On the left, there is a 'Tags' sidebar with a text input for adding tags and a note about entering single words with no spaces. The bottom of the page shows a file edit history entry: 'Stefano Pogliani edited a file. In this article.docx' on January 22.

In our exercise, we will modify the look-and-feel of this page to be the following one:

We are not going to modify any functionality from the standard Profile page, but simply changing the way in which the standard information is shown.

Two different approaches

We will present two ways to this usecase:

1. The first one makes use some Javascript code which will inject the CSS file containing the stylesheet we want to apply
2. The second uses a new functionality, recently introduced in IBM Connections Customizer, which allows the developer to directly reference CSS files in the Extension definition.

We thought that, whilst the second one is, by far, the quickest and the most efficient from a performance perspective, the first would be a useful exercise that would allow students to further understand the way in which IBM Connections Customizer works.

The CSS File

Both approaches would require the definition of the CSS file.

	<p>The objective here is not to explain the Stylesheet itself. You may experiment with the CSS file which is provided as an example in order to better understand the way the CSS directives have been applied in order to produce the desired result.</p>
---	--

The CSS file that we will be using, and the associated image used as a background are already available under the Lab2 directory of the Github repository associated with the IBM Connections Cloud organization we are using.

Please do not touch nor modify those two files as they will be used by the code we are going to write.

stefanopog / Think2018-Customizer-Lab-Org

Code Issues Pull requests Projects Wiki Insights Settings

Branch: master Think2018-Customizer-Lab-Org / Lab2 /

Create new file Upload files Find file History

Latest commit c68524f 14 seconds ago

		ok	14 seconds ago
	docs		
	README.md	ok	14 seconds ago
	profileImageBackground.jpeg	ok	2 minutes ago
	profilesCustomization.css	ok	2 minutes ago
	README.md		

Profiles Customization exercise



For your own convenience, the CSS file and the associated background image are also available under the Lab2 directory of the “**Reference Repository**”

Using Javascript injection

We have seen [in our first Lab](#) how the IBM Connections Customizer injection engine works. So, we know that if we create a Javascript script, a reference to that script will be placed at the end of the HTML code that displays the page.

If the script will contain executable statements, they will then be executed immediately as the HTML finishes loading.

Preparation Activities

You have been invited as a Contributor to the Github repository associated to this organization.



The instructor will provide you the URL and the credentials for accessing it.

You will arrive to a page similar to this one.

This screenshot shows a GitHub repository page for 'Exercise'. The repository has 11 commits, 1 branch, 0 releases, and 1 contributor. The latest commit was just now. The repository contains files like 'Lab1', 'Lab2', 'Lab3', 'Lab4', and 'README.md'. Below the repository details, there is a large text area containing the number '123456789' and the text '123456789 - Think2018 Customizer Lab Organization'. It also mentions 'All the scripts that will be activated for the Think2018 IBM Connections Customizer Organization'.

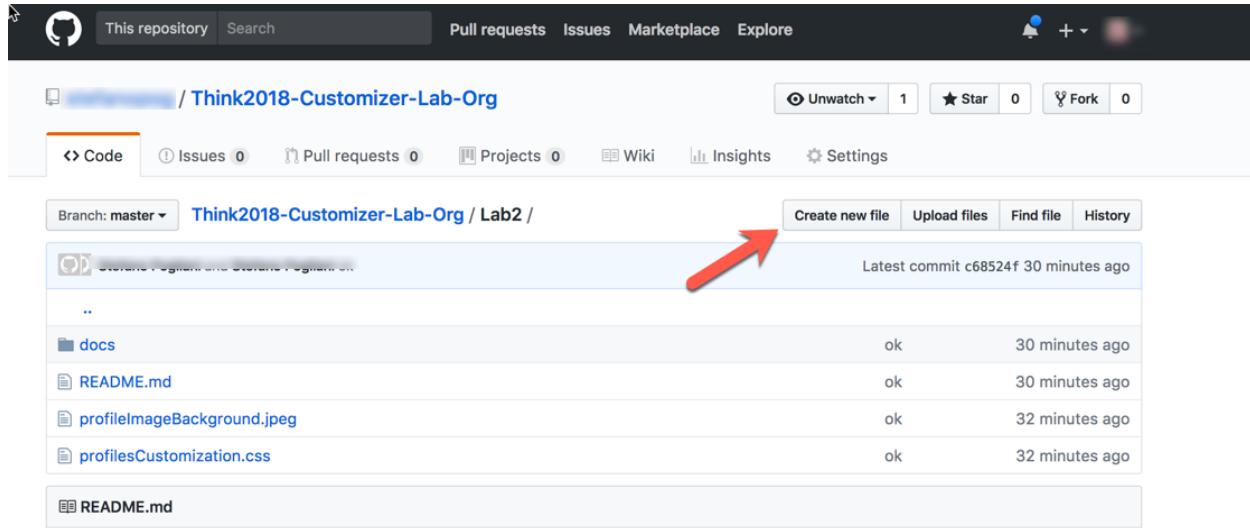
Click on the “Lab2” item and you will access this page:

This screenshot shows the 'Lab2' directory within the 'Exercise' repository. The directory contains files such as 'docs', 'README.md', 'profileImageBackground.jpeg', and 'profilesCustomization.css'. The latest commit was c68524f, made 28 minutes ago. Below the directory listing, there is a large text area titled 'Profiles Customization exercise' containing a screenshot of an IBM Connections profile page for 'Stefano Pogliani'.

Now you are ready to code!

Creating your script

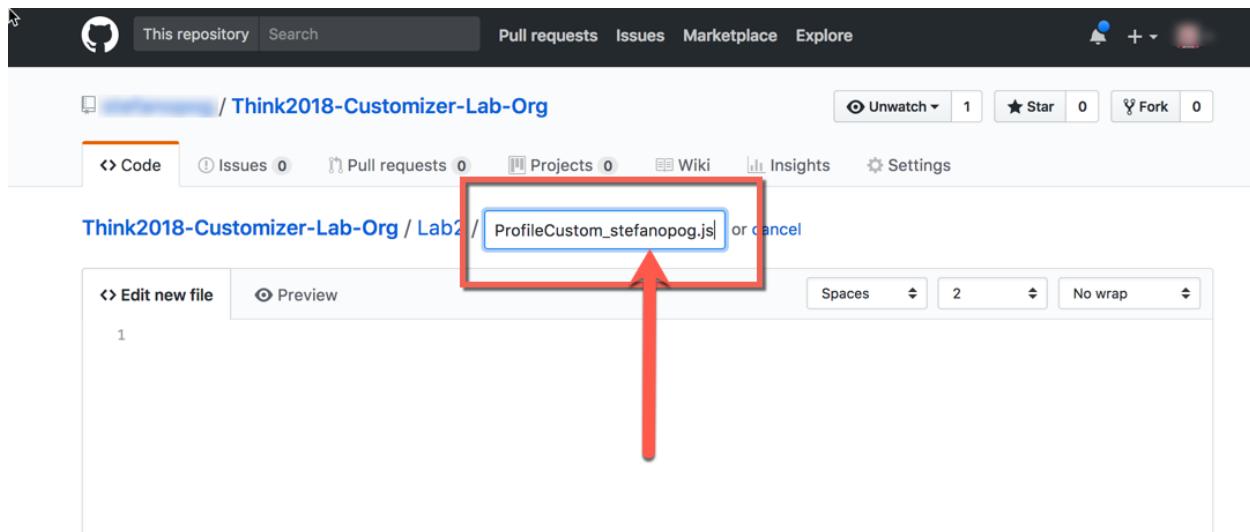
Create the Javascript file that will contain the code for your extension. Click on “Create File” as shown in the following image:



Now, you have to enter the name of the file. Since there are many people doing the exercise at the same time, we need to avoid conflict in filenames. For this reason, enter the following for the filename:

ProfileCustom_<identifier>.js

<identifier> is a unique label that will be provided to you by the instructor (it is going to be the first part of the email address associated with your IBM Connections Cloud account for this lab).



Let's now proceed step-by-step in building our code together.

- The script will need to upload a CSS file.

The CSS will need to be placed in the <head> element of the HTML page so that the browser will load it immediately and will apply it to the page.

The dojo way to do it is the following:

```
dojo.place('your HTML element', dojo.doc.head, 'last');
```

where:

- “your HTML element” will be the <link> element referencing the script.
In our case it will be :

```
'<link rel="stylesheet" type="text/css" src="/files/customizer/Lab2/profilesCustomization.css">'
```

Note the path for the “src” qualifier: “/files/customizer/Lab2” is the URL where the artefacts we create in our Github repository will be stored after the validation phase.



	<p>Note the use of the file “profileCustomization.css”. As previously stated this file has already been placed in your Directory. It contains the CSS classes that will build the new User Interface for the Profiles page. Feel free to explore it in order to understand how the CSS classes have been used, but please do not modify it.</p>
--	---

- dojo.doc.head is the handle to the <head> element provided by dojo
- “last” informs the dojo.place to place the new element as the last child of the <head> element

The final form of our statement will be:

```
dojo.place('<link rel="stylesheet" type="text/css" src="/files/customizer/Lab2/profilesCustomization.css">', dojo.doc.head, 'last');
```

- At this point, your script looks like this:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Lab2'. The code editor contains the following line of JavaScript:

```
1 dojo.place('<link rel="stylesheet" type="text/css" src="/files/customizer/Lab2/profilesCustomization.css">', dojo.doc.head, 'last')
```

- We are still missing one point though.

In the code we wrote we assumed that the dojo toolkit would be available for the page in order for your script to use it. We are confident that the IBM Connections page will load the dojo toolkit at a certain point, but we cannot assume that it will be loaded by the time we first use it. We certainly do not want our script to load the dojo toolkit another time.... So we need to wait for it to be ready on the page.

Fortunately the dojo toolkit provides a standard mechanism to deal with this point: the "dojo/domReady!" plugin (described here <https://dojotoolkit.org/reference-guide/1.10/dojo/domReady.html>).

So, we will simply need to wrap the code we wrote up until now in this way:

```
if (typeof(dojo) != "undefined") {
    require(["dojo/domReady!"], function () {
        try {
            ... all our previous code ....
        } catch(e) {
            alert('exception occurred in HelloWorld : ' + e);
        }
    });
}
```

We have introduced a “try... catch” statement to globally catch any exception we may encounter.

At the end, the code in your editor should look like this:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 1 pull request, 0 issues, 0 pull requests, 0 projects, 0 wiki pages, 0 insights, and 0 settings. The code editor displays the following JavaScript code:

```
1 if (typeof (dojo) != "undefined") {
2     require(["dojo/domReady!"], function () {
3         try {
4             dojo.place(
5                 '<link rel="stylesheet" type="text/css" href="/files/customizer/Lab2/profilesCustomization.css"/></link>',
6                 dojo.doc.head,
7                 'last'
8             );
9         } catch (e) {
10             alert('exception occurred in HelloWorld : ' + e);
11         }
12     });
13 }
```

- You can now commit your code to the repository. It is always good practice to add, as a comment, what you did.



The dialog box is titled 'Commit new file' and contains a text input field with the placeholder 'Create ProfileCustom_stefanopog.js'. Below the input field is a text area containing the text 'This is the script'. At the bottom of the dialog are two radio button options: one selected for 'Commit directly to the master branch.' and another for 'Create a new branch for this commit and start a pull request.' There are also 'Commit new file' and 'Cancel' buttons.

- Your script is now safely recorded in the Github repository.

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 1 star and 0 forks. The 'Code' tab is selected, showing a list of files under the 'master' branch. One file, 'ProfileCustom_stefanopog.js', is highlighted with a red arrow pointing to it. The commit history shows several other files like 'docs', 'README.md', 'profileImageBackground.jpeg', and 'profilesCustomization.css' were also updated at the same time. Below the code view, there's a preview of an IBM Connections profile page with a blue header and a user profile picture.

Creating your extension

Whilst the extension can be directly created in the IBM Connections Cloud Administration panel, we suggest that you save the extension definition as a JSON file in the same Github repository.



In this way, the extension may be used in other situations and may become part of the overall documentation of the project

In order to create the extension, we need to create a new file inside our Github repository. The file will follow this naming convention

`ProfileCustom_<identifier>.json`

Where <identifier> is a unique label that will be provided to you by the instructor (it is going to be the first part of the email address associated with your IBM Connections Cloud account for this lab). In this way, everybody will immediately understand that this is the extension definition associated with the `ProfileCustom_<identifier>` script.

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Lab2'. The 'Code' tab is selected. A red arrow points to the 'Create new file' button in the top right corner of the code editor area.

Branch: master / Think2018-Customizer-Lab-Org / Lab2 /

Create new file Upload files Find file History

Latest commit 937d9fc just now

ProfileCustom_stefanopog.js	Create ProfileCustom_stefanopog.js	just now
docs	ok	an hour ago
ProfileCustom_stefanopog.js	Create ProfileCustom_stefanopog.js	just now
README.md	ok	an hour ago
profileImageBackground.jpeg	ok	an hour ago
profilesCustomization.css	ok	an hour ago
README.md		

Profiles Customization exercise

Stefano Pogliani Local Time: 12:46 PM

Recent Updates Contact Information Background Files

Tags Add logo to this profile

Recent Updates What do you want to share?

Network No network contacts are associated with this profile

And, then,

The screenshot shows a GitHub code editor with a new file named 'ofileCustom_stefanopog.json' being created. A red arrow points to the file name input field.

ofileCustom_stefanopog.json or cancel

Edit new file Preview Spaces 2 No wrap

```
1
```

Let's first copy this empty template inside our new json file (a copy of this template is available as the Lab2/emptyTemplate.json under "Reference Repository"):

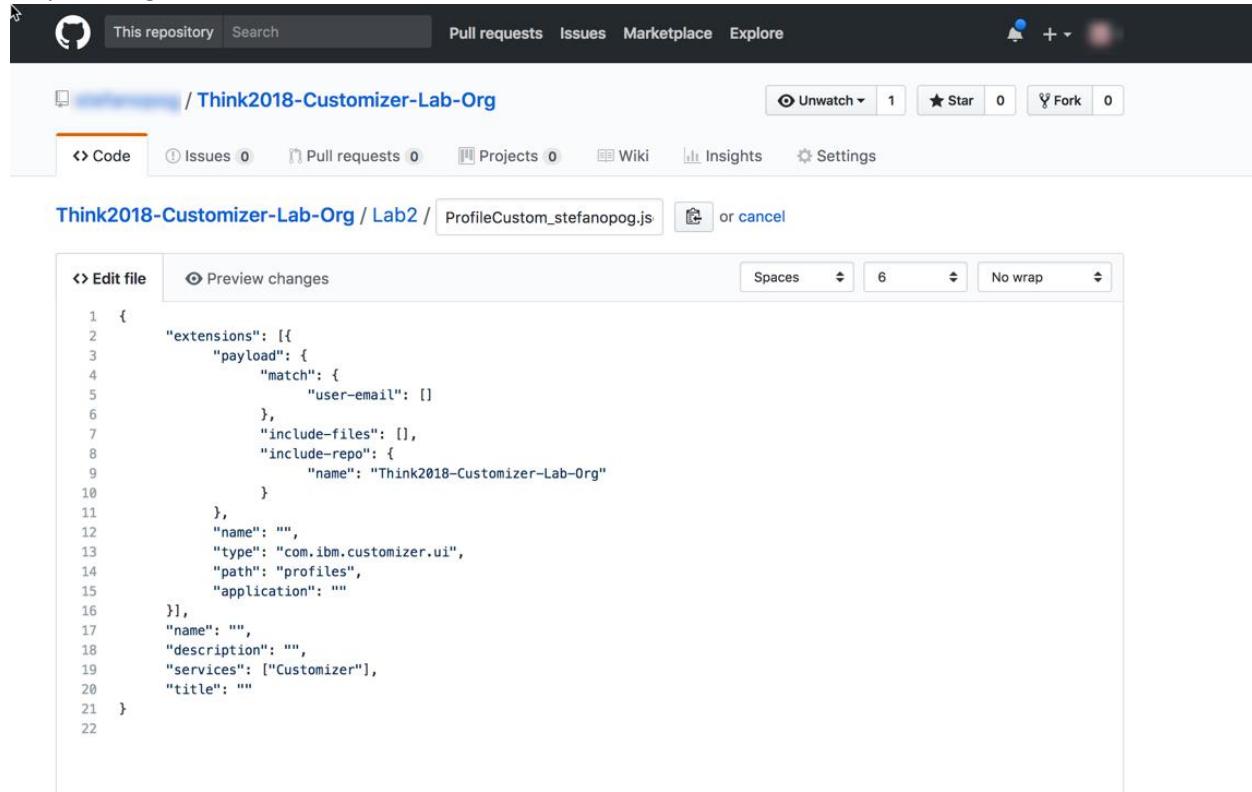
```
{
  "extensions": [{}
```

```

"payload": {
    "match": {
        "user-email": []
    },
    "include-files": [],
    "include-repo": {
        "name": "Think2018-Customizer-Lab-Org"
    }
},
"name": "",
"type": "com.ibm.customizer.ui",
"path": "profiles",
"application": ""
},
"name": "",
"description": "",
"services": ["Customizer"],
"title": ""
}
}

```

So you will get this result:



```

1  {
2      "extensions": [
3          "payload": {
4              "match": {
5                  "user-email": []
6              },
7              "include-files": [],
8              "include-repo": {
9                  "name": "Think2018-Customizer-Lab-Org"
10             }
11            },
12            "name": "",
13            "type": "com.ibm.customizer.ui",
14            "path": "profiles",
15            "application": ""
16        ],
17        "name": "",
18        "description": "",
19        "services": ["Customizer"],
20        "title": ""
21    }
22

```

Some of the values of the JSON descriptor are already provided. A complete documentation for the attributes is available at this URL :

<https://github.com/ibmcnxdev/customizer/blob/master/docs/IBMConnectionsCustomizer.pdf>

- “include-repo”: (line 8)

This MUST be the name of the Github repository that we are using. It is the “root” directory that the IBM Connections Customizer engine will use in order to retrieve the scripts that the extension uses

- “type”: (line 13)
This is a string whose value “com.ibm.customizer.ui” will tell IBM Connections Cloud which type of IBM Connections Customizer service will be used by the extension
- “path”: (line 14)
The value of “profiles” will instruct the IBM Connections Customizer engine to add the scripts referenced by the extension only to the “profiles” service of IBM Connections
- “services”: (line 19)
This is a string whose value “Customizer” will tell IBM Connections Cloud which type extension it needs to register in the Application Registry.

Let's now fill the other values that are required:

- “user-email”: (line 5)
We can restrict the injection of the scripts associated to the extension only to users whose email address will match the string.
Since we do not want to interfere with other people working at the same time on the same IBM Connections Cloud organization, we need to provide a value for this attribute.
The value needs to be the email address you have been provided by the instructor to log into IBM Connections Cloud (enclosed in double-quotes). So, something like:

```
"user-email": ["stefanopog@think2018-Customizer.com"]
```



In real life, you may not need to use this attribute if the extension you are creating will be used by **all the people** in your organization.
See the [documentation](#) for discovering all the possibilities associated to the “match” clause.

- “include-files”: (line 7)
This is where you will actually reference the script that you created earlier in the exercise.
The path to reference the script is relative to the Github repository you are using (see the “include-repo” clause described above).
So in your situation you need to provide the following value :

```
"include-files": ["Lab2/ProfileCustom_<identifier>.js"]
```

- “name”: (line 12)
this is the name of the Extension.
You can use the following:

```
"name": "Profile Ext Lab2 <identifier>"
```

- “name”: (line 17)
This is the name of the Application we are going to declare to IBM Connections. Typically, an “Application” can declare one or more extensions (in our example, we have one extension only associated with the application).

This name must be unique for the IBM Connections Organization, since it is the name that will be recognized by the IBM Connections Application registry. You can use the following name:

"name": "Profile App Lab2 <identifier>"

- "application": (line 14)

This entry (which is within the "extension") actually needs to reference the correct "Application". So, the value you will provide for this attribute MUST BE the same as the value you provided for the "name" attribute of the Application (line 17). You can use the following name:

"application": "Profile App Lab2 <identifier>"

- "description": (line 18)

This is a free text description of what the Application (and its Extensions) provide. You can use the following:

"description": "Profile Customization from <identifier>"

- "title": (line 20)

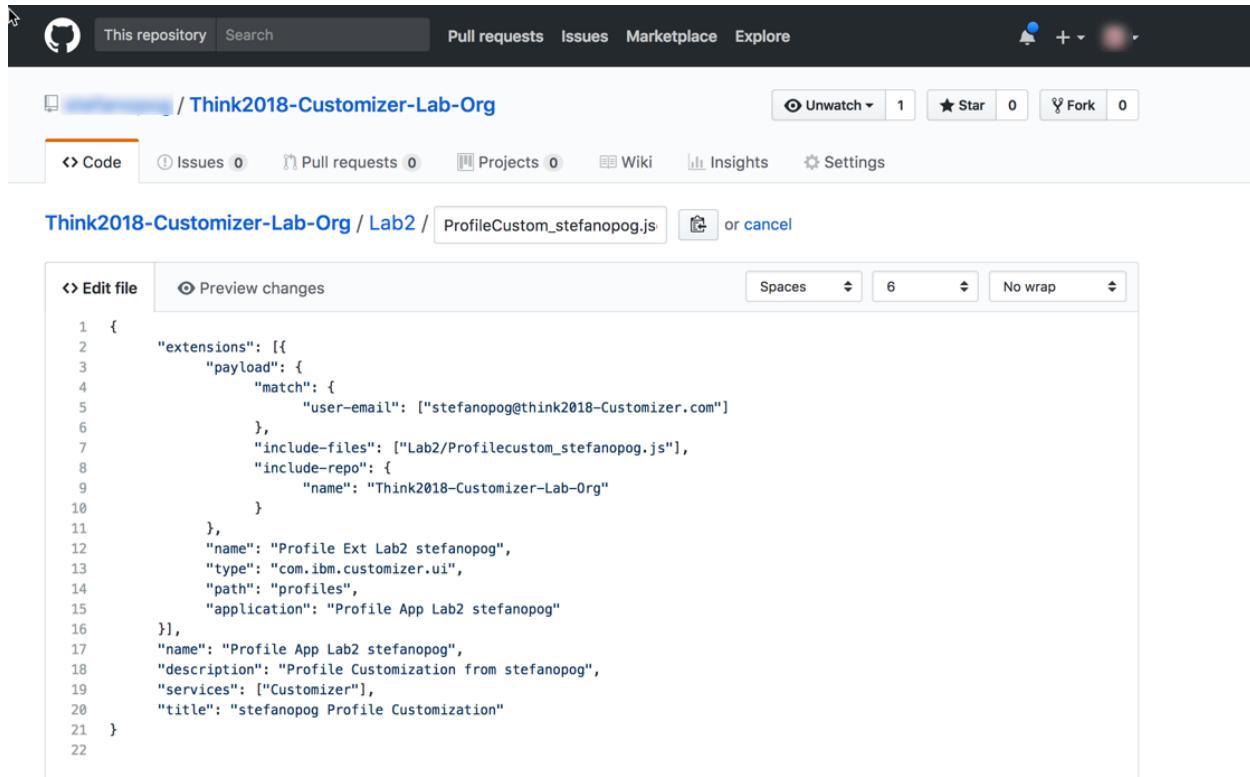
This is the title that will appear in the IBM Connections Cloud application registry. You can use the following title:

"title": "<identifier> Profile Customization"

Once the Organization admin will use your JSON file to declare the extension, this is how it will appear in the IBM Connections Cloud Application Registry:

The screenshot shows the IBM Connections Application Registry interface. On the left, there's a navigation sidebar with sections like Personal, Organization Extensions (which is currently selected), and System Settings. The main area displays a grid of application cards. One card, titled 'stefanopog Profile Customizati...' with the subtitle 'Service: Customizer', has a red arrow pointing to its title. Another red arrow points to the subtitle 'Profile Customization from stefanopog'. The card also indicates it is 'Enabled'.

At this point, the JSON file you are editing in Github will look like this:

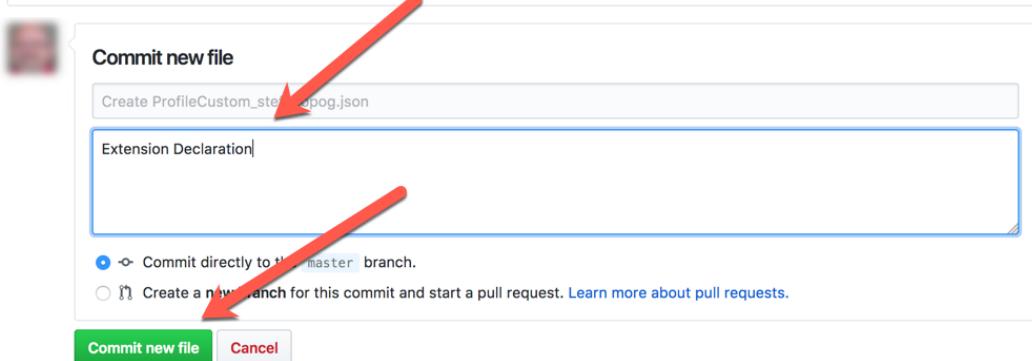


The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with links for 'This repository', 'Search', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. On the right side of the header, there are icons for a bell (notifications), a plus sign (create), and a user profile. Below the header, the repository path 'Think2018-Customizer-Lab-Org' is displayed, along with buttons for 'Unwatch' (1), 'Star' (0), 'Fork' (0). The main content area shows a file named 'ProfileCustom_stefanopog.js' being edited. The code in the editor is as follows:

```
1  {
2      "extensions": [
3          {
4              "payload": {
5                  "match": {
6                      "user-email": ["stefanopog@think2018-Customizer.com"]
7                  },
8                  "include-files": ["Lab2/Profilecustom_stefanopog.js"],
9                  "include-repo": {
10                      "name": "Think2018-Customizer-Lab-Org"
11                  }
12              },
13              "name": "Profile Ext Lab2 stefanopog",
14              "type": "com.ibm.customizer.ui",
15              "path": "profiles",
16              "application": "Profile App Lab2 stefanopog"
17          },
18          {
19              "name": "Profile App Lab2 stefanopog",
20              "description": "Profile Customization from stefanopog",
21              "services": ["Customizer"],
22              "title": "stefanopog Profile Customization"
23          }
24      ]
25 }
```

You can Commit the change to the Github repository:

```
15     "application": "Profile App Lab2 stefanopog"
16   },
17   "name": "Profile App Lab2 stefanopog",
18   "description": "Profile Customization from stefanopog",
19   "services": ["Customizer"],
20   "title": "stefanopog Profile Customization"
21 }
```



And your Repository will look like this:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 0 issues, 0 pull requests, 0 projects, 0 wiki pages, 0 insights, and 0 settings. The branch is 'master'. The file 'ProfileCustom_stefanopog.json' is highlighted with a red arrow. The file was created just now and is located in the 'docs' folder.

File	Content	Last Modified
ProfileCustom_stefanopog.json	Create ProfileCustom_stefanopog.json	just now
ProfileCustom_stefanopog.js	ok	an hour ago
docs	ok	2 hours ago
README.md	ok	2 hours ago
profileImageBackground.jpeg	ok	3 hours ago
profilesCustomization.css	ok	3 hours ago

Profiles Customization exercise

The screenshot shows the 'Profiles Customization exercise' interface. It displays a preview of a customized profile for 'Stefano Poglani' with a local time of 12:46 PM. The interface includes sections for Tags, Recent Updates, and Network.

Making your extension available

At this point, everything is ready to be put in production. This is what the [documentation](#) states:

- Share your repo with IBM – [add "ibmcnxddev" as a collaborator](#)
- IBM ([ibmcnxddev](#)) then creates a fork of your repository under [github.com/ibmcnxddev](#) and grants you read access by default.
- You can continue to work on your extension using your original repo for your source code activity, but once you are ready to deliver to IBM Cloud you must issue a [pull request](#) to IBM.
- IBM merges your pull request once acceptance criteria are met.
- Upon merge, the repo files are automatically pushed to IBM Customizer via a webhook.
- Rinse & repeat starting at Step (c) for extension updates.

Steps **a.** and **b.** have already been done for you before you started the exercise.

Now, you need to execute Step **c.**

Go back to the root directory of your Github repository and click on the “Pull Request” button:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. At the top, there are tabs for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. Below these, there are statistics: 13 commits, 1 branch, 0 releases, and 1 contributor. A red arrow points to the 'New pull request' button, which is located below the commit count. The main area displays a list of recent commits, including 'Create HelloWorld_stefanopog.json' by 'Lab1' (7 minutes ago), 'Create README.md' by 'Lab2' (3 hours ago), 'Create README.md' by 'Lab3' (3 hours ago), 'Create README.md' by 'Lab4' (3 hours ago), and 'Update README.md' by 'README.md' (3 hours ago). There is also a section for 'README.md'.

Activating the extension

Now your extension has been validated. The validation process (from **IBMCNDEV**), if successful, will make the files you created in this exercise available to IBM Connections Cloud and to IBM Connections Customizer.

The only thing that is missing is to actually define the Extension in the IBM Connections Cloud Application Registry. This is the step that informs IBM Connections Customizer of what to do with the scripts that you just created.

The IBM Connections Cloud organization administrator needs to be given the JSON file which contains the extension (the one you created [here](#)).

He will go to the IBM Connections Application Registry and will declare a new extension:

The screenshot shows the IBM Connections Cloud Administration interface. On the left, there's a sidebar with sections like 'Personal', 'User Accounts', 'Organization Account Settings', etc. Under 'Organization Extensions', there are 'Connections Mobile App Management', 'Chat and Meetings', and 'Apps'. The main area has a heading 'Info: To create classic Organization Extensions, click here'. Below it are tabs 'Apps' and 'All Apps'. A prominent red arrow points to the 'New App +' button. The right side shows a grid of app cards.

A new screen is shown:

The screenshot shows the 'App Editor' screen. The sidebar includes 'Organization Extensions' with 'Apps' selected. The main area has tabs 'Back to Apps' and 'Code Editor' (which is selected). It displays a 'App Editor' title and a warning message: 'Check out the help and examples section for working with apps, if you need some guidance.' Below that is a 'Warning: Import will overwrite editor contents.' message. At the bottom are 'Cancel' and 'Save' buttons. A large red arrow points to the JSON code area on the right, and another red arrow points to the 'Save' button.

```

1 {
2   "name": "",
3   "title": "",
4   "description": "",
5   "services": false,
6   "extensions": false
7 }

```

The IBM Connections Cloud organization admin will replace the content pointed by the right arrow (1) with the content of the JSON file you created and, then, saves (2) the extension.

You can now go to the homepage and see the final result:

The screenshot shows a user profile page for Stefano Pogliani. At the top, there's a navigation bar with links like Home, Mail, Calendar, People, Communities, Files, Meetings, Host a Meeting, Activities, More, Intranet, Intranet2, Admin, and a search bar. Below the header is a decorative banner featuring a network of icons like a gear, a person, and a document.

The main content area starts with a circular profile picture of Stefano Pogliani, followed by his name "Stefano Pogliani" and the local time "12:46 PM". There are buttons for "Edit My Profile" and "Send Email/Download vCard".

Below the profile are tabs for "Recent Updates", "Contact Information", "Background", and "Files".

The "Recent Updates" section contains the following items:

- A message from Stefano: "You liked your message." (Hello) - Thursday, January 22
- A file edit: "Stefano Pogliani edited a file." (In this article.docx) - Stefano Pogliani - January 22
- A file edit: "Stefano Pogliani edited a file." - Stefano Pogliani - January 22

The "Network" section indicates "No network contacts are associated with this profile" and has a "View All" link.

The "My Links" section says "There are no links yet for this profile" and has a "Add Link" button.

Using CSS Declaration

What we have been describing in the previous section ([Using Javascript injection](#)) is the approach which has Javascript to load the CSS file.

Recently, the IBM Connections Customizer engine has been improved in order to directly accept the declaration of CSS files within the Extension declaration. Roughly, you will be able to use a direct reference to the CSS files you may want to apply to a given IBM Connections page from the “include-files” clause of the Extension declaration.



Obviously, the rendering of the “customized” IBM Connections page will be much quicker and effective

Let's see together how we can modify what we have previously done in order to make a proper use of this new functionality.

Preparation Activities

What we need to do is, simply, to modify the Extension declaration in order to reference the CSS file instead of the Javascript file.

Go to the Github directory associated with the second Lab (the one we just worked on in the [first part of this lab](#)).

The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with links for 'This repository', 'Search', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, the repository name 'Think2018-Customizer-Lab-Org' is displayed, followed by the branch 'master'. A red arrow points to the 'ProfileCustom_stefanopog.json' file in the list of files. The list includes other files like 'docs', 'ProfileCustom_stefanopog.js', 'ProfileCustom_stefanopog.json', 'README.md', 'profileImageBackground.jpeg', and 'profilesCustomization.css'. At the bottom of the repository view, there's a section titled 'Profiles Customization exercise' with a preview of an IBM Connections profile page.

Click on the JSON Extension file your previously created:

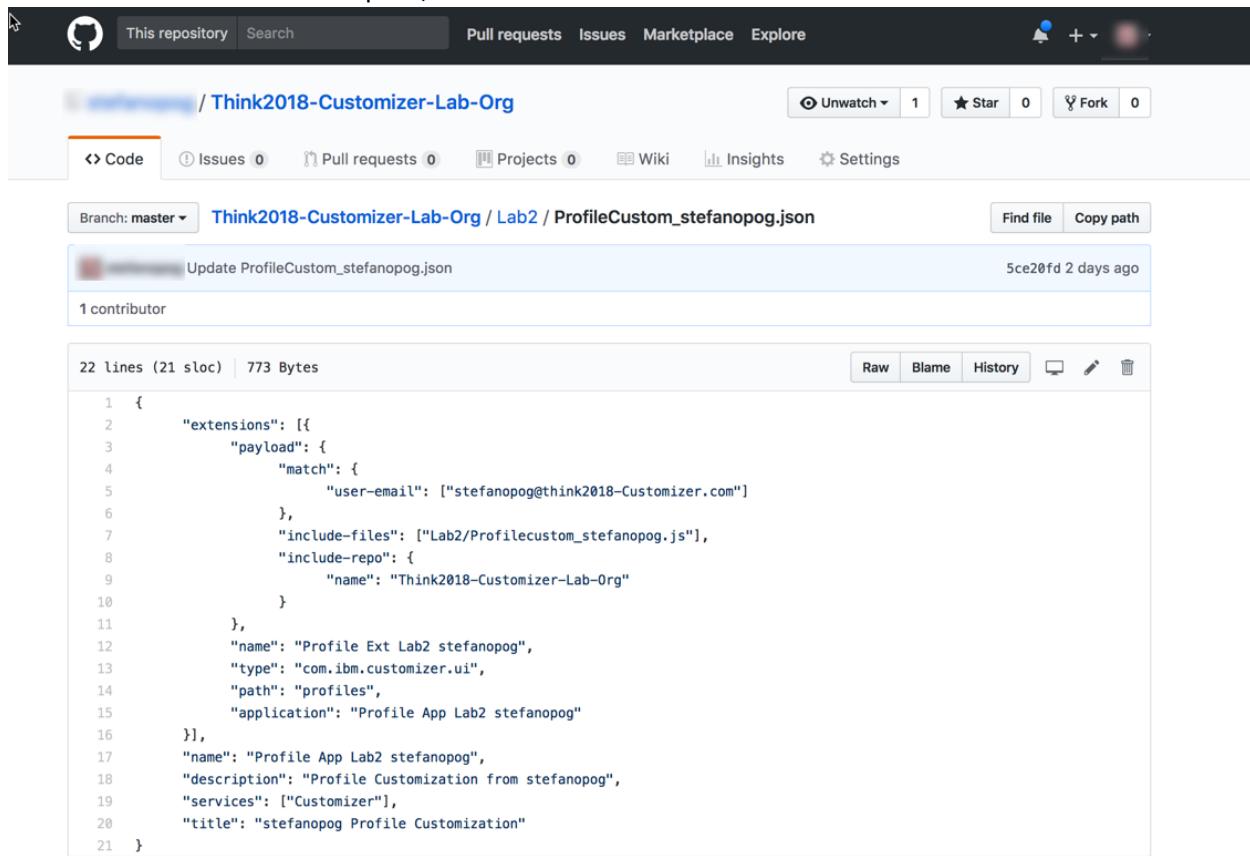
The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Lab2'. The commit history is displayed, with a red arrow pointing to the commit for 'ProfileCustom_stefanopog.json'. The commit message is 'Update ProfileCustom_stefanopog.json'.

File	Message	Date
docs	ok	2 days ago
ProfileCustom_stefanopog.js	Create ProfileCustom_stefanopog.js	2 days ago
ProfileCustom_stefanopog.json	Update ProfileCustom_stefanopog.json	2 days ago
README.md	ok	2 days ago
profileImageBackground.jpeg	ok	2 days ago
profilesCustomization.css	ok	2 days ago

Profiles Customization exercise

The screenshot shows a user profile page with a custom profile background featuring a network diagram. The user's name is 'Stefano Pogliani'.

When the “view” on this file opens,



The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The file 'ProfileCustom_stefanopog.json' is selected. The code editor displays the JSON content:

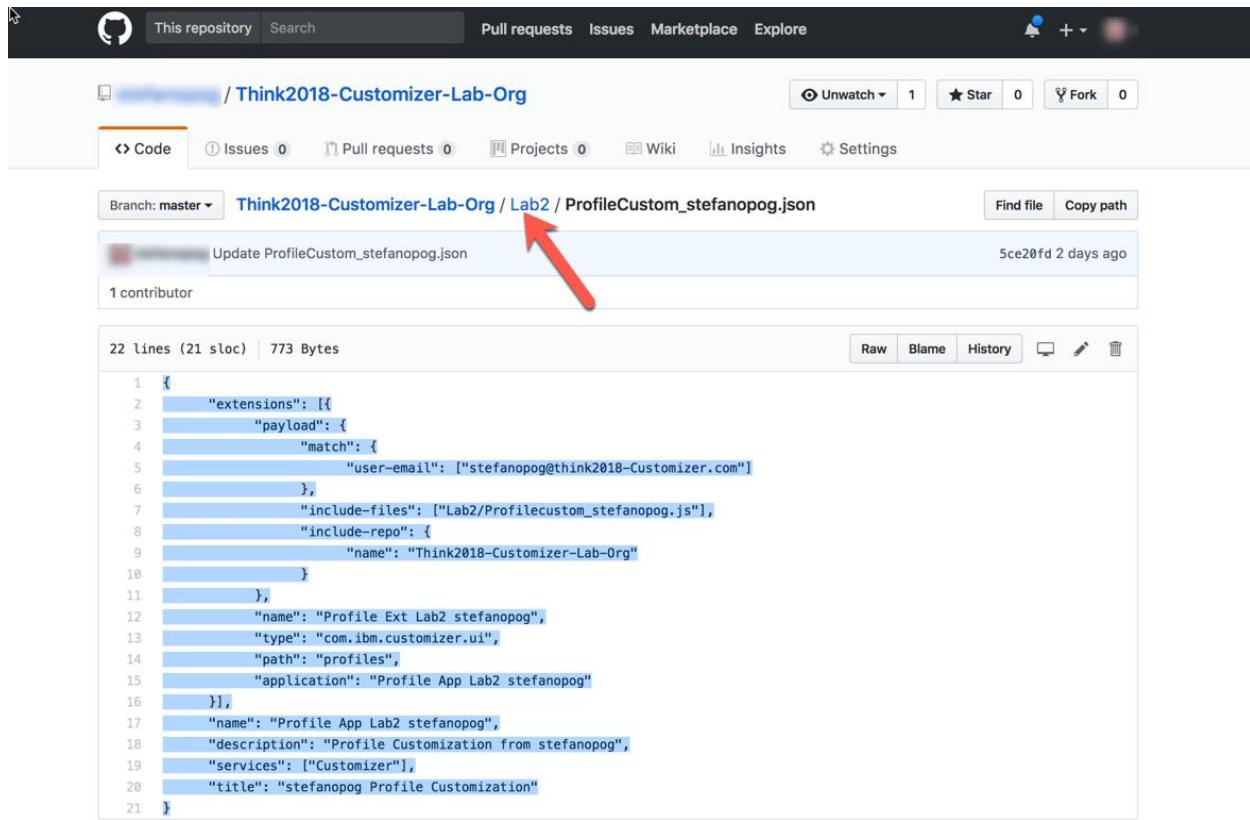
```
1  {
2      "extensions": [
3          "payload": {
4              "match": {
5                  "user-email": ["stefanopog@think2018-Customizer.com"]
6              },
7              "include-files": ["Lab2/Profilecustom_stefanopog.js"],
8              "include-repo": {
9                  "name": "Think2018-Customizer-Lab-Org"
10             }
11            },
12            "name": "Profile Ext Lab2 stefanopog",
13            "type": "com.ibm.customizer.ui",
14            "path": "profiles",
15            "application": "Profile App Lab2 stefanopog"
16        ],
17        "name": "Profile App Lab2 stefanopog",
18        "description": "Profile Customization from stefanopog",
19        "services": ["Customizer"],
20        "title": "stefanopog Profile Customization"
21    }
```

by means of the mouse, select the whole content of the file and, then, issue CTRL-C to copy the selection into the computer clipboard:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository has 1 unwatched star and 0 forks. The 'Code' tab is selected, showing a file named 'ProfileCustom_stefanopog.json'. The file was last updated 2 days ago by a contributor named 'stefanopog'. The code content is a JSON object with 22 lines and 21 SLOC, totaling 773 bytes. The JSON structure defines an 'extensions' array containing a single object. This object has a 'payload' key which contains a 'match' key with a 'user-email' value of 'stefanopog@think2018-Customizer.com'. It also includes 'include-files', 'include-repo', and 'name' keys. Another part of the object specifies a profile named 'Profile Ext Lab2 stefanopog' with type 'com.ibm.customizer.ui', path 'profiles', and application 'Profile App Lab2 stefanopog'. The final part of the object specifies a profile named 'Profile App Lab2 stefanopog' with description 'Profile Customization from stefanopog', services 'Customizer', and title 'stefanopog Profile Customization'.

```
1 {
2     "extensions": [
3         {
4             "payload": {
5                 "match": {
6                     "user-email": ["stefanopog@think2018-Customizer.com"]
7                 },
8                 "include-files": ["Lab2/Profilecustom_stefanopog.js"],
9                 "include-repo": {
10                     "name": "Think2018-Customizer-Lab-Org"
11                 }
12             },
13             "name": "Profile Ext Lab2 stefanopog",
14             "type": "com.ibm.customizer.ui",
15             "path": "profiles",
16             "application": "Profile App Lab2 stefanopog"
17         },
18         {
19             "name": "Profile App Lab2 stefanopog",
20             "description": "Profile Customization from stefanopog",
21             "services": ["Customizer"],
22             "title": "stefanopog Profile Customization"
23         }
24 }
```

Now go back to the Lab2 Directory clicking on the “Lab2” link shown in the picture below:



The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The breadcrumb navigation bar at the top right shows the path: Think2018-Customizer-Lab-Org / Lab2 / ProfileCustom_stefanopog.json. A red arrow points to the 'ProfileCustom_stefanopog.json' part of the path. The main content area displays the JSON file's code.

```
1 {
2     "extensions": [
3         {
4             "payload": {
5                 "match": {
6                     "user-email": ["stefanopog@think2018-Customizer.com"]
7                 },
8                 "include-files": ["Lab2/Profilecustom_stefanopog.js"],
9                 "include-repo": {
10                     "name": "Think2018-Customizer-Lab-Org"
11                 }
12             },
13             "name": "Profile Ext Lab2 stefanopog",
14             "type": "com.ibm.customizer.ui",
15             "path": "profiles",
16             "application": "Profile App Lab2 stefanopog"
17         },
18         {
19             "name": "Profile App Lab2 stefanopog",
20             "description": "Profile Customization from stefanopog",
21             "services": ["Customizer"],
22             "title": "stefanopog Profile Customization"
23         }
24 }
```

Now create a new file in the Lab2 Directory as shown here:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The repository name is 'Lab2 /'. The 'Code' tab is selected. The file list shows several files and folders, all updated 2 days ago. A red arrow points to the 'Create new file' button in the top right corner of the file list area.

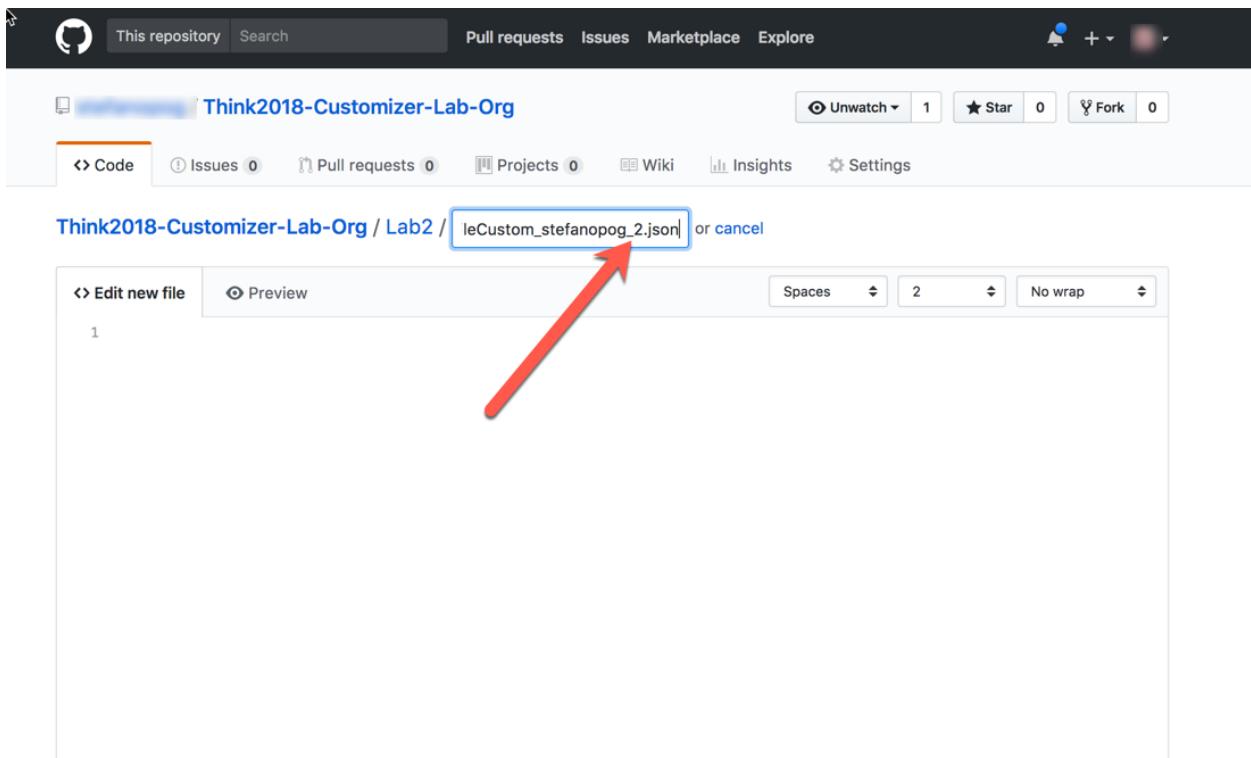
File/Folder	Last Commit
Update ProfileCustom_stefanopog.json	Latest commit 5ce20fd 2 days ago
..	
docs	ok 2 days ago
ProfileCustom_stefanopog.js	Create ProfileCustom_stefanopog.js 2 days ago
ProfileCustom_stefanopog.json	Update ProfileCustom_stefanopog.json 2 days ago
README.md	ok 2 days ago
profileImageBackground.jpeg	ok 2 days ago
profilesCustomization.css	ok 2 days ago

Profiles Customization exercise

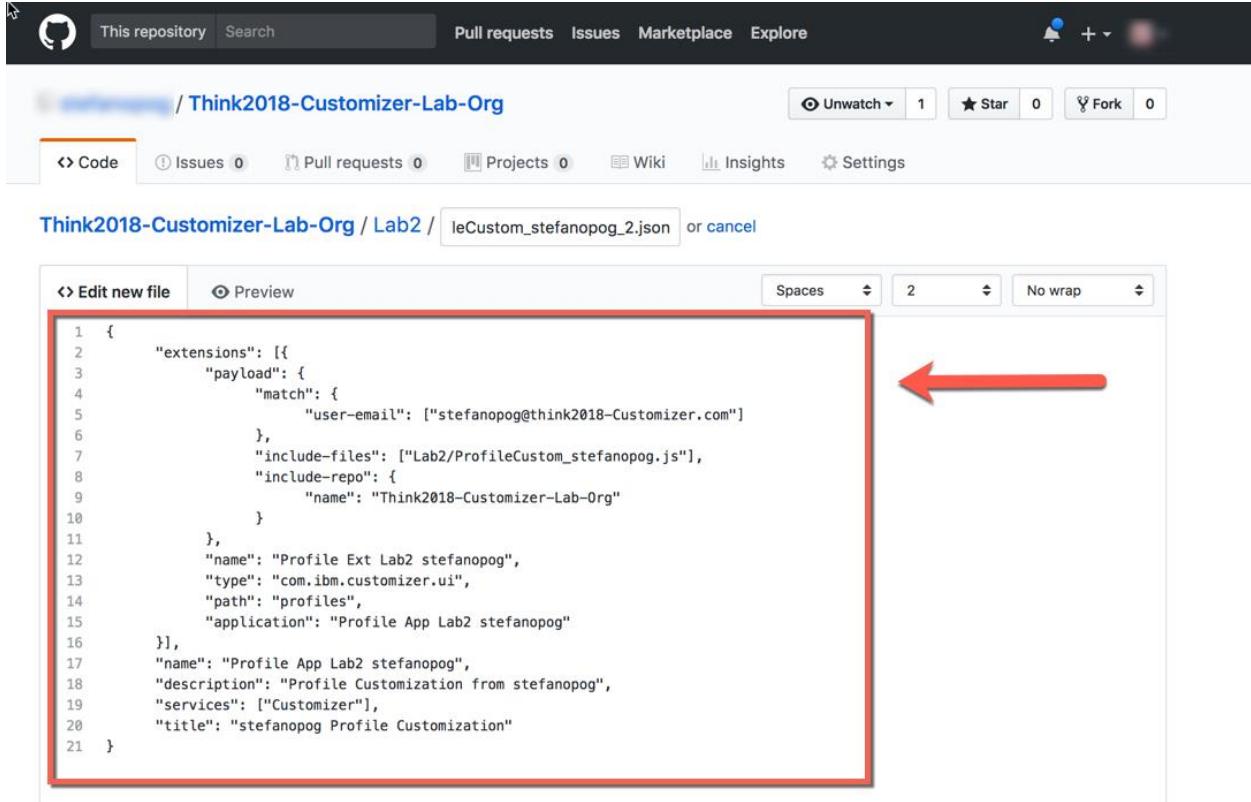
Give the new file the following name:

ProfilCustom_<identifier>_2.json

Where <identifier> is a unique label that will be provided to you by the instructor (it is going to be the first part of the email address associated with your IBM Connections Cloud account for this lab). Note that we append a “_2” suffix to the file name in order to distinguish it from the other one.



Now, using CTRL-V paste in the canvas what you previously copied to the clipboard:



The only modifications you need to perform are:

- **at line 7.** You will change:

```
"include-files": ["Lab2/ProfileCustom_stefanopog.js"]
```

 with

```
"include-files": ["Lab2/profilesCustomization.css"]
```
- **at line 17.** You will change:

```
"name": "Profile App Lab2 <identifier>"
```

 with

```
"name": "Profile App 2 Lab2 <identifier>"
```
- **at line 15.** You will change:

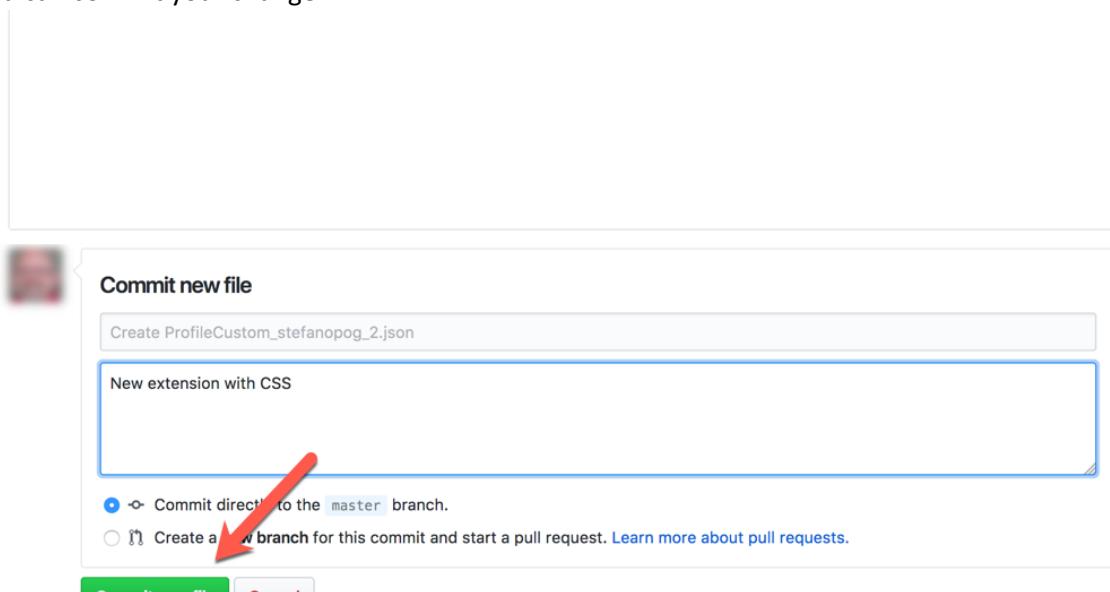
```
"application": "Profile App Lab2 <identifier>"
```

 with

```
"application": "Profile App 2 Lab2 <identifier>"
```

Those changes created a new extension (we added a **2** to the name in order to distinguish it from the one we previously created) which loads the CSS file directly.

Then you can commit your change:



And your Repository will look like this:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. The 'Code' tab is selected. A red arrow points to the file 'ProfileCustom_stefanopog_2.json' in the list of files. The file was created just now. Below the file list, there is a preview of a 'Profiles Customization exercise' page.

File	Action	Time
ProfileCustom_stefanopog_2.json	Create ProfileCustom_stefanopog_2.json	just now
ProfileCustom_stefanopog.js	Create ProfileCustom_stefanopog.js	2 days ago
ProfileCustom_stefanopog.json	Update ProfileCustom_stefanopog.json	2 days ago
ProfileCustom_stefanopog_2.json	Create ProfileCustom_stefanopog_2.json	just now
README.md	ok	2 days ago
profileImageBackground.jpeg	ok	2 days ago
profilesCustomization.css	ok	2 days ago

Profiles Customization exercise

Making your extension available

At this point, everything is ready to be put in production. This is what the [documentation](#) states:

- Share your repo with IBM – [add "ibmcnxddev" as a collaborator](#)
- IBM ([ibmcnxdev](#)) then creates a fork of your repository under [github.com/ibmcnxdev](#) and grants you read access by default.
- You can continue to work on your extension using your original repo for your source code activity, but once you are ready to deliver to IBM Cloud you must issue a [pull request](#) to IBM.
- IBM merges your pull request once acceptance criteria are met.
- Upon merge, the repo files are automatically pushed to IBM Customizer via a webhook.
- Rinse & repeat starting at Step (c) for extension updates.

Steps **a.** and **b.** have already been done for you before you started the exercise.

Now, you need to execute **Step c.**

Go back to the root directory of your Github repository and click on the “Pull Request” button:

The screenshot shows a GitHub repository page for 'Think2018-Customizer-Lab-Org'. At the top, there are navigation links: This repository, Search, Pull requests, Issues, Marketplace, Explore. Below the header, the repository name is displayed. On the right side, there are buttons for Unwatch (1), Star (0), Fork (0). The main content area shows repository statistics: 13 commits, 1 branch, 0 releases, 1 contributor. A red arrow points to the 'New pull request' button, which is located in the middle of the stats row. Below the stats, there's a dropdown for the branch ('Branch: master') and the 'New pull request' button itself. Further down, a list of recent commits is shown, including 'Create HelloWorld_stefanopog.json', 'Create README.md', 'Create README.md', 'Create README.md', and 'Update README.md'. The latest commit was made 7 minutes ago.

Activating the extension

Now your extension has been validated. The validation process (from **IBMCNDEV**), if successful, will make the files you created in this exercise available to IBM Connections Cloud and to IBM Connections Customizer.

The only thing that is missing is to actually define the Extension in the IBM Connections Cloud Application Registry. This is the step that informs IBM Connections Customizer of what to do with the scripts that you just created.

The IBM Connections Cloud organization administrator needs to be given the JSON file which contains the extension (the one you created [here](#)).

He will go to the IBM Connections Application Registry and will declare a new extension:

Info: To create classic Organization Extensions, [click here](#)

Apps All Apps

New App +

A new screen is shown:

Info: To create classic Organization Extensions, [click here](#)

Back to Apps

Code Editor

App Editor

Check out the help and examples section for working with apps, if you need some guidance.

Warning: Import will overwrite editor contents.

Cancel Save

```

1 {
2   "name": "",
3   "title": "",
4   "description": "",
5   "services": false,
6   "extensions": false
7 }
```

The IBM Connections Cloud organization admin will replace the content pointed by the right arrow (1) with the content of the JSON file you created and, then, **saves** (2) the extension.

You can now go to the homepage and see the final result:

Home Mail Calendar People Communities Files Meetings Host a Meeting Activities More Intranet Intranet2 Admin ?

Stefano Pogliani

Local Time: 12:46 PM

Edit My Profile

Send Email Download vCard

Recent Updates Contact Information Background Files

Tags

Add tag(s) to this profile +

Enter a tag as a single word with no spaces. Examples: human-resources, hiring, manager, payroll.

Recent Updates

What do you want to share?

You liked your message.
Hello Thursday

Stefano Pogliani edited a file.
In this article.docx Stefano Pogliani January 22

Stefano Pogliani edited a file.

Network

No network contacts are associated with this profile

[View All](#)

My Links

There are no links yet for this profile.

[Add Link](#)

This ends Lab 2.
Congratulations!

We Value Your Feedback!

- Don't forget to submit your Think 2018 session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.
- Access the Think 2018 agenda tool to quickly submit your surveys from your smartphone, laptop or conference kiosk.