Account Creation

Gmail : New account creation

(for people creating burner accounts only... otherwise feel free to use your normal e-mail account.)

1. Visit http://mail.google.com/mail/signup.

2. Fill out the form that appears. The fields should be as follows;

   - The name is arbitrary
   - Username should be “ficlopBurnerX”, where X is a number. **Case IS relevant**!

     Eg, ficlopBurner1, ficlopBurner2, etc.

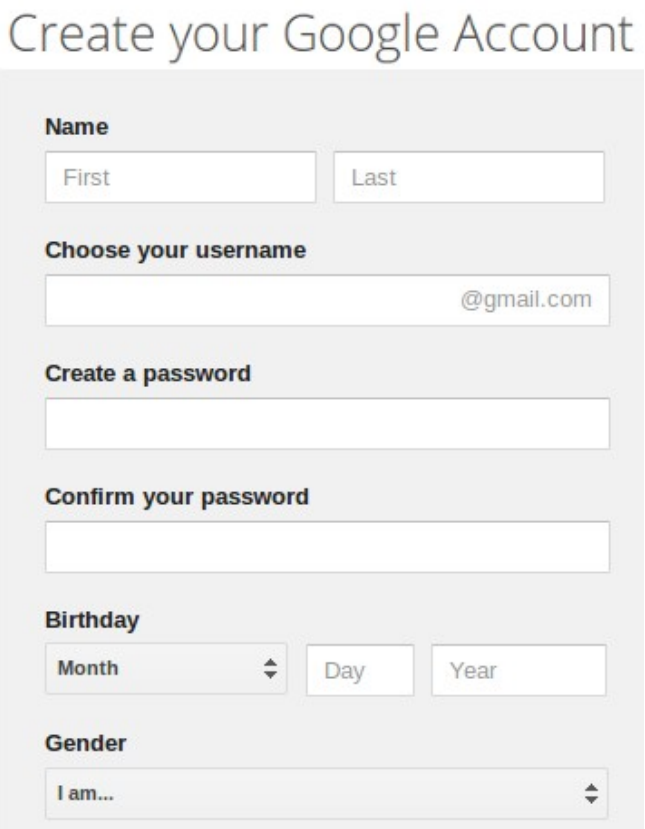
   - For password, use the same string as you used for username, but reverse the order of the words. Eg

     username: ficlopBurner1
     password: 1BurnerFiclop

   - Birthday and gender are irrelevant

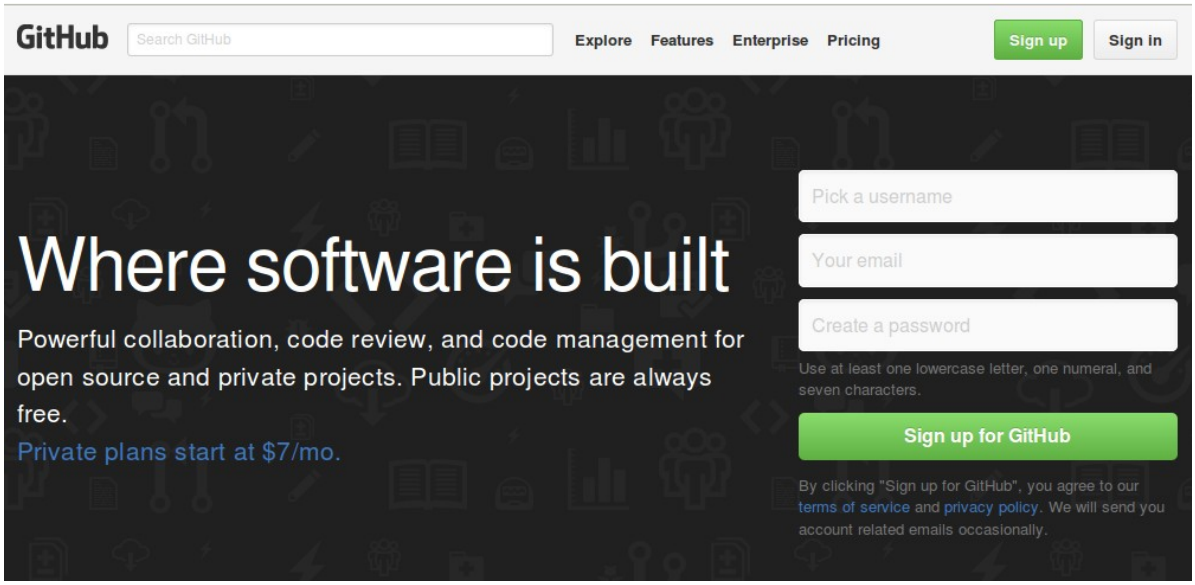
   - Leave all other fields blank

   - Complete the image captcha and hit “Next step”.

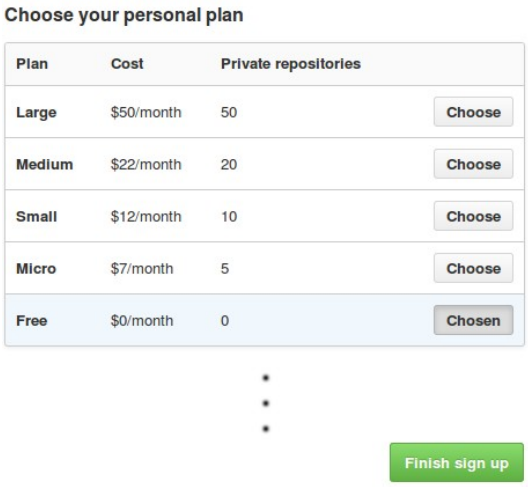


Github: New account creation

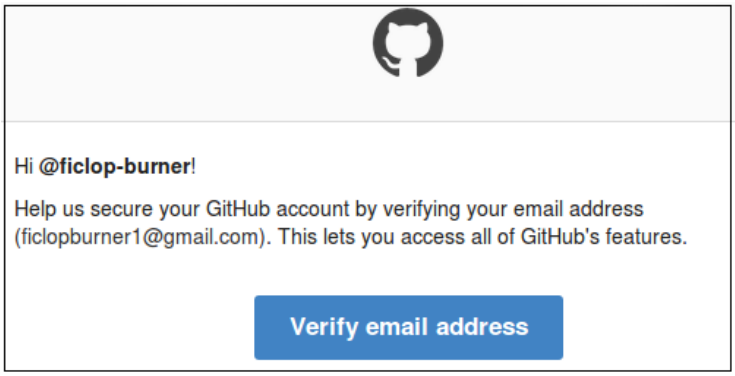
1. Visit https://github.com/. If you are not signed in as a different user, the page will appear as shown:Create a username, provide a valid e-mail address, and choose a password. Then hit “Sign up for Github”



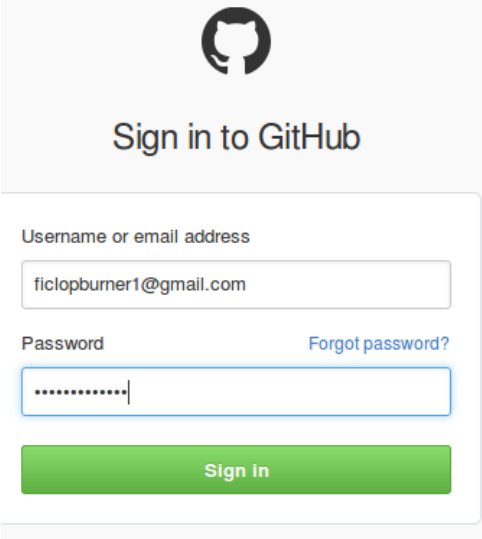
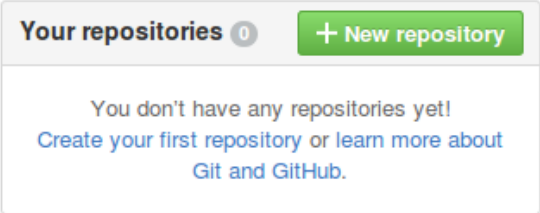
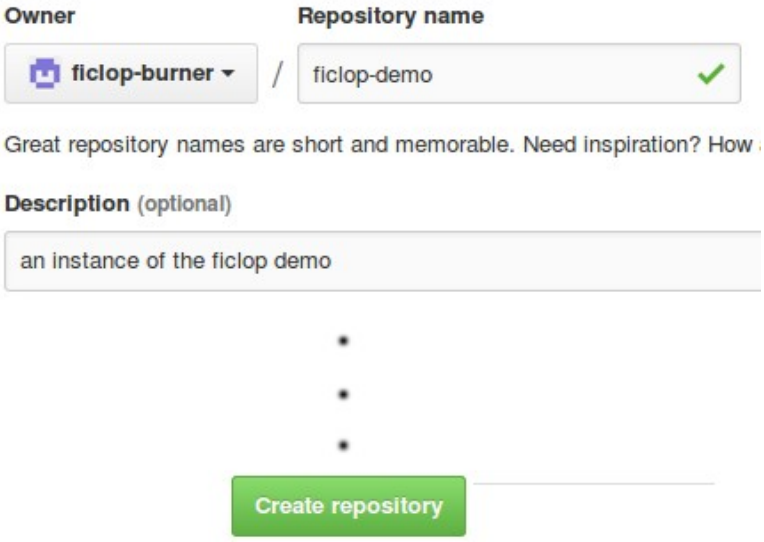
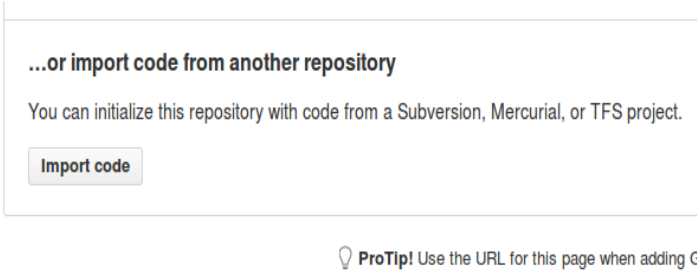
2. A page asking what payment plan you want to use will load. The free option is already chosen, so just click “Finish Sign up”.

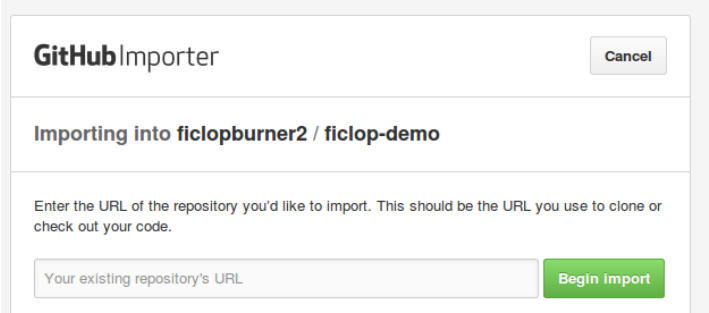
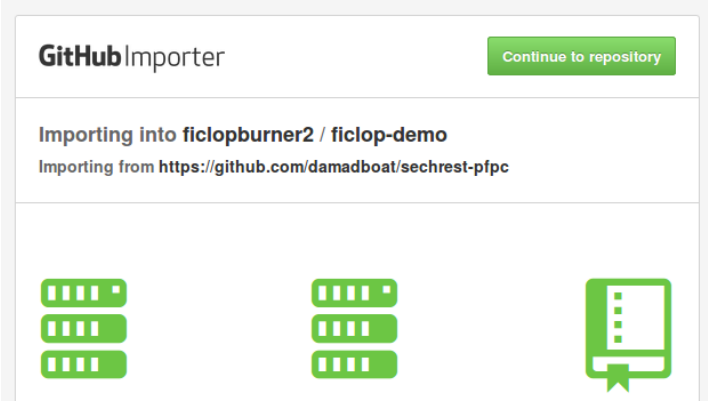


3. You should have received a verification e-mail from GitHub at the e-mail address you provided. Log in, open the e-mail from GitHub, and hit “Verify e-mail address”.

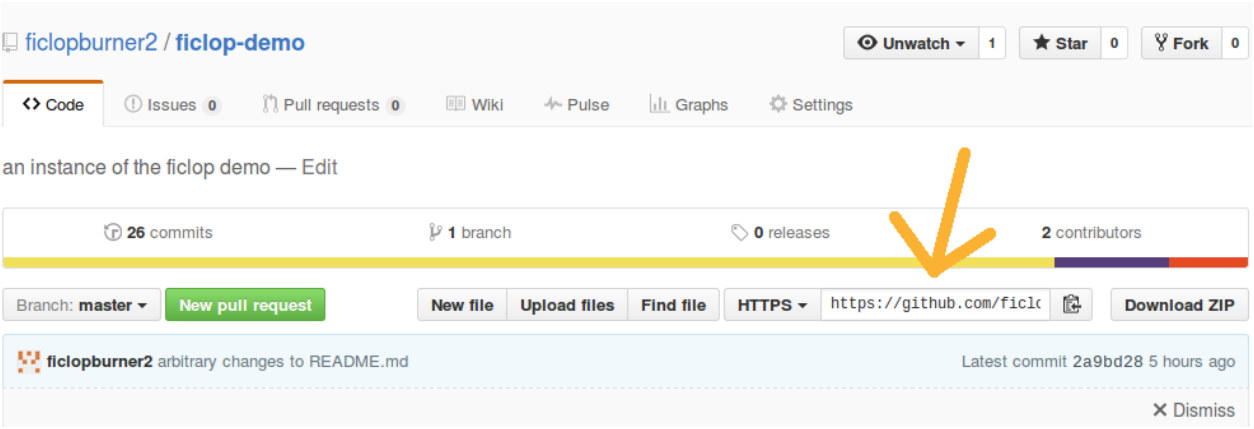


Github : Clone Ficlop repository

1. Visit https://github.com/. If necessary, log in.	
1. After you have logged in, you should see your account homepage. A new Github account will have no repositories. To add one Hit the "+ New repository" button, shown here.	
2. Give the new repository a name and a brief description. After that hit "Create repository"; we'll be cloning another repository to get you started, so we can skip repository initialization.	
3. Your new repository is currently empty. We're going to duplicate an existing repository to get you started. Hit the "Import code" button at the bottom of the page.	

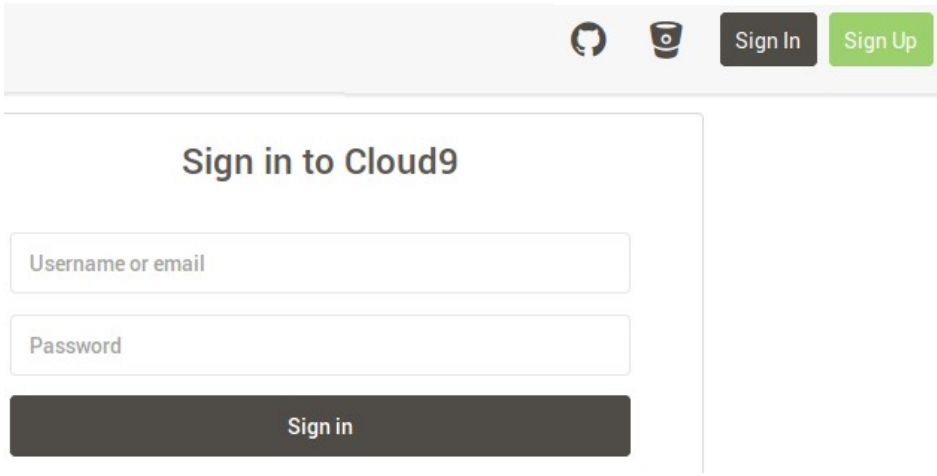
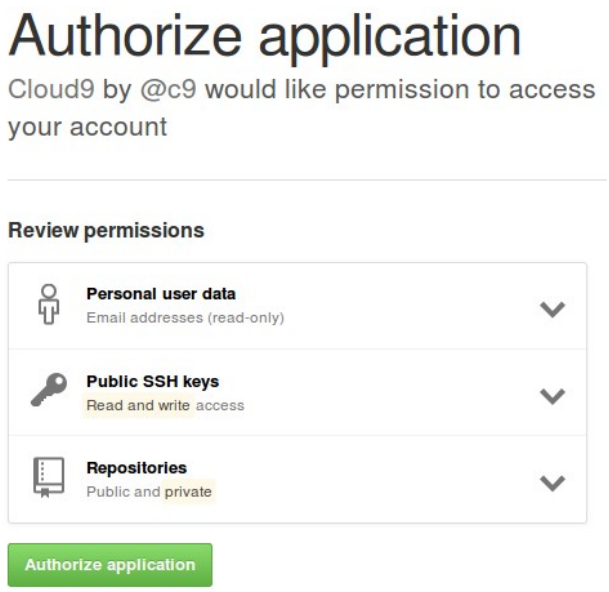
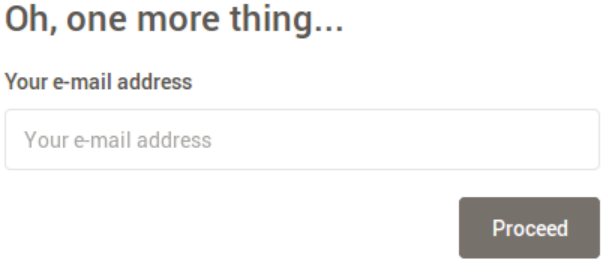
4. When the “GitHub Importer” page loads, fill in ~~https://github.com/ficlop-master/ficlop-demo-master~~ https://github.com/damadboat/sechrest-pfpc and click “Begin Import”.	
5. If the import process is successful, after a few moments the page will appear as shown. Click “Continue to repository” to see what was imported.	
6. If the import process was successful, your repository will appear as shown.	PUT IN A PICTURE SPACER SPACER

We have successfully created a GitHub account. Before we leave, however, please make note of the git repository address associated with this repository; this is important for some of the services we'll be using later. It is recoverable on the main page of your repository, as shown below.



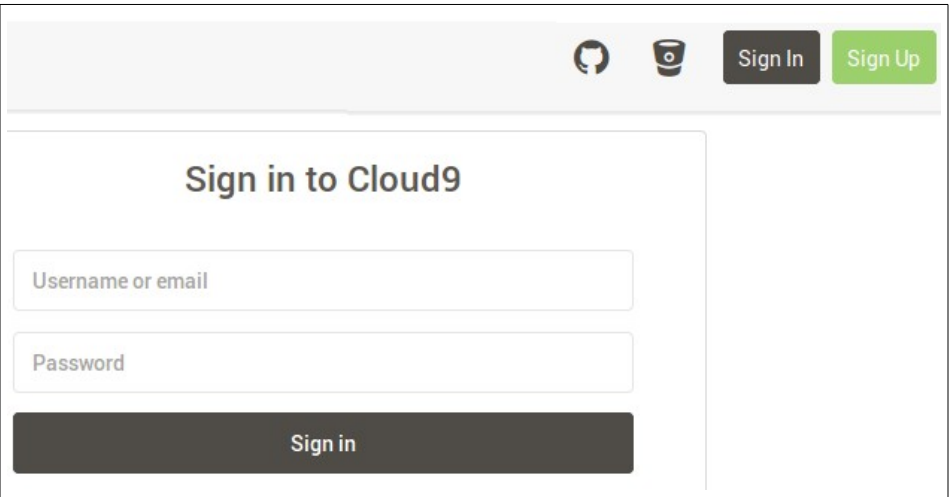
The git repository address is usually the same as the URL of your repository, with a “.git” extension added to the end.

Cloud9: New Account Creation

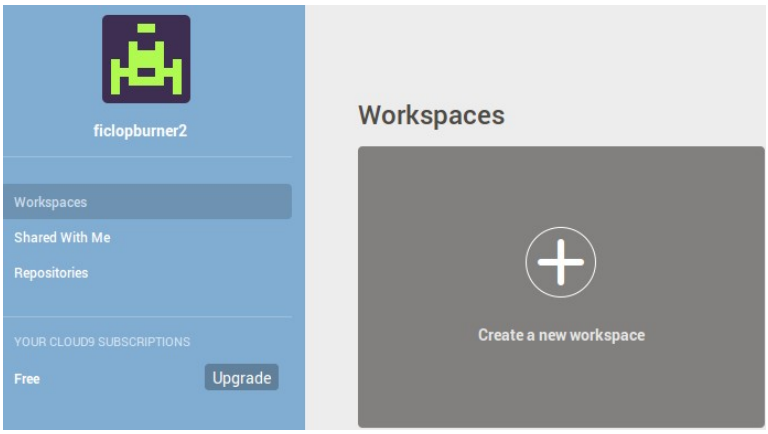
1. Visit cloud9's login page (https://c9.io/login) and click “Sign Up”.	
2. After the page shown loads, click the button marked “GitHub”.	
3. If you are logged into your Github account, Github will open a page asking you to authorize cloud9's permission request. Click “Authorize application.” If you are not logged into your Github account, login and try again.	
4. After giving permission to Cloud9, Cloud9 may ask you to provide your e-mail address in a pop-up window.	

Cloud9: Create a new project from GitHub code base

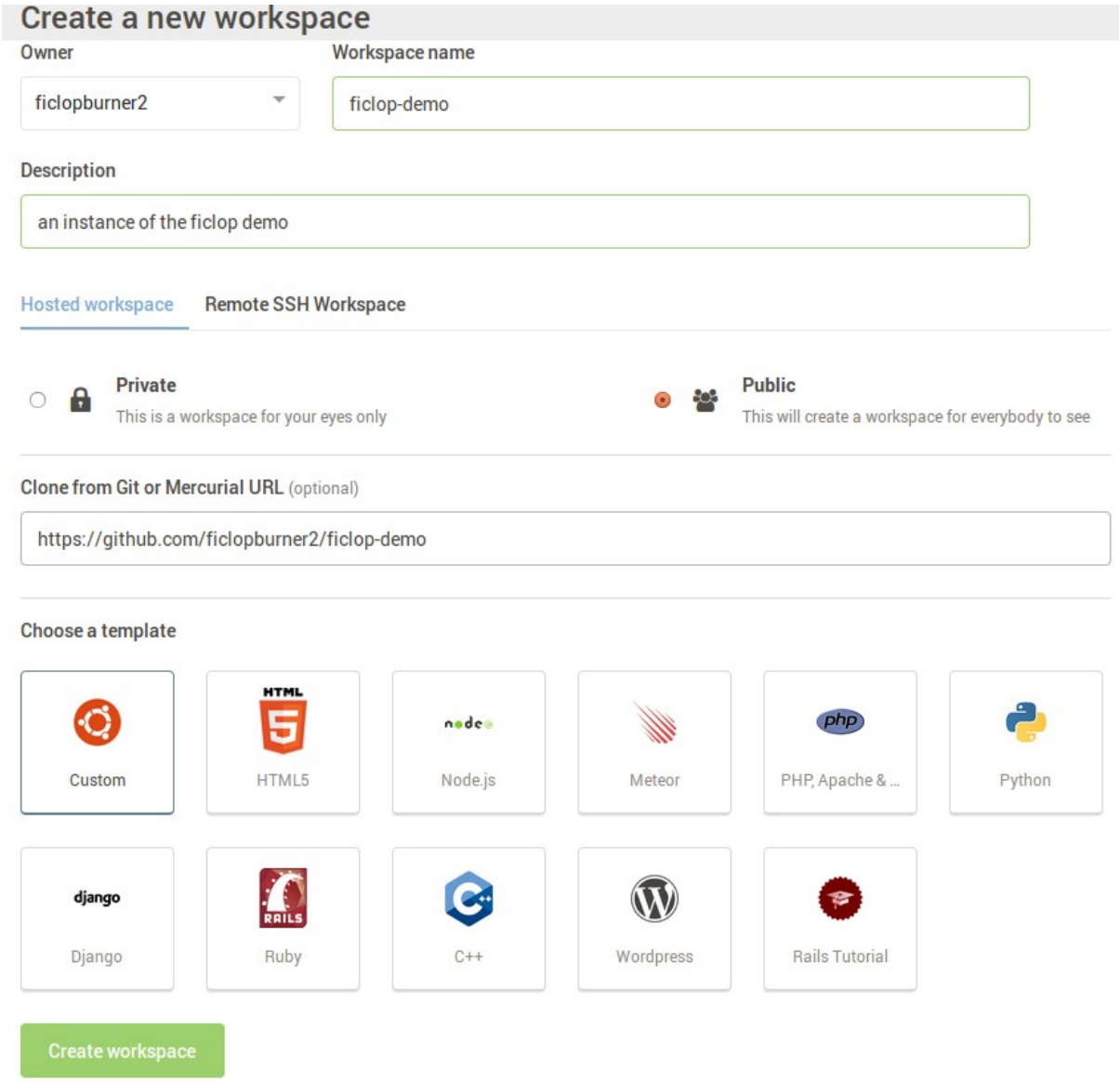
1. Visit cloud9's login page (https://c9.io/login) and sign in.

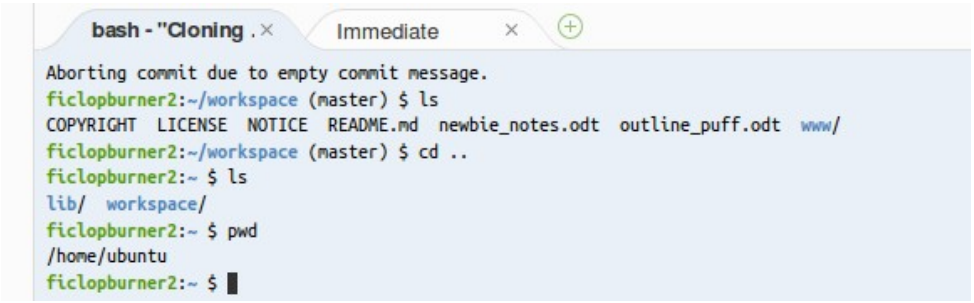


2. To create a new Cloud9 workspace with our GitHub project as a template, click “Create a new workspace.”

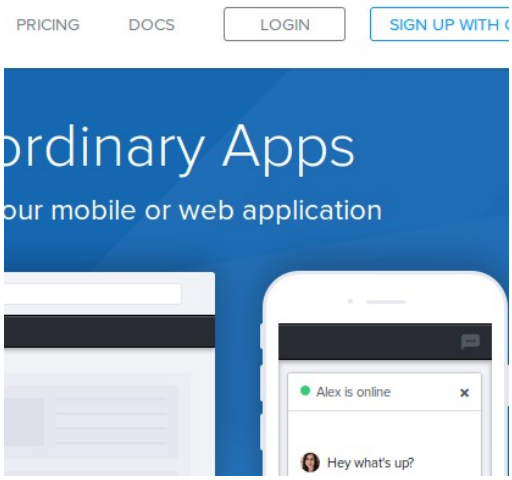
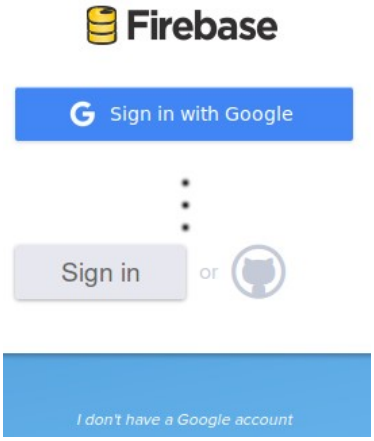
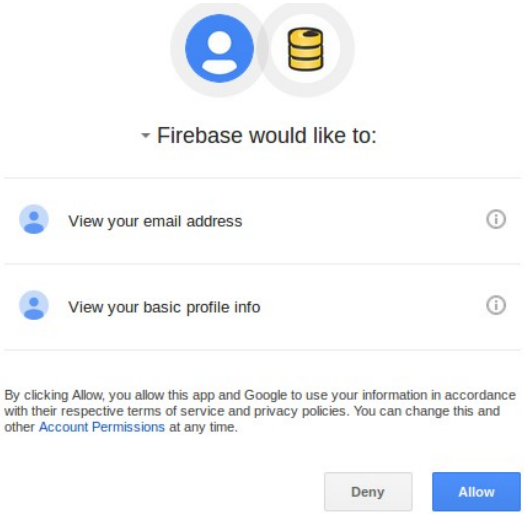


3. Fill out the form that loads as you see fit. In the “Clone from Git or Mercurial” field, specify the URL of the Github account that you created in the previous section.



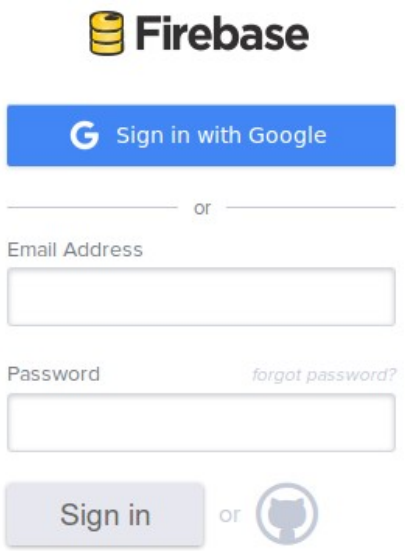
4. If the workspace creation process is successful, Cloud9 will open the newly created project. Shown here is the workspace navigator tab for that project. The directory structure should appear as it does in GitHub. 5. Also of interest is Cloud9's “bash” terminal, shown here. This allows any bash commands available on a typical Linux PC.	PUT IN A PICTURE SPACER SPACER SPACER 

Firebase: Account Creation

1. Visit https://www.firebase.com/ and click “Login”.	
2. If you have a Gmail account, click “Sign in with Google”. If not, click “I don't have a Google account.” This will prompt you to create an account with Google.	
3. Firebase may ask to recover some data from your gmail account. Click “Allow”	

Populating a Firebase Database

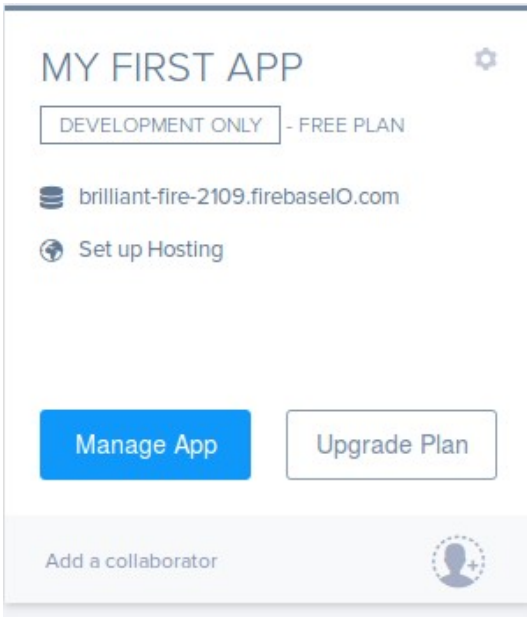
1. Visit https://www.firebase.com/login/ and provide your account credentials.



2. When you created your account, Firebase created a stub ~~application~~ database for you. This database is issued a randomly generated name and URL; in the example shown, the name is " brilliant-fire-2109", and is accessible from the following URL:

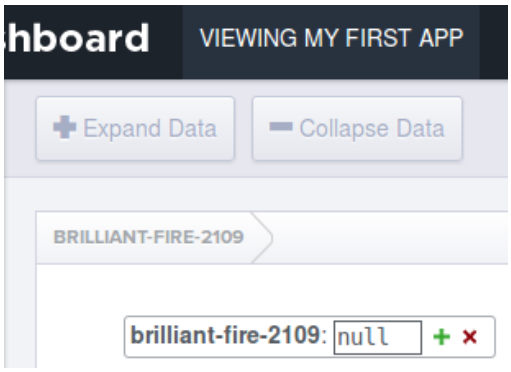
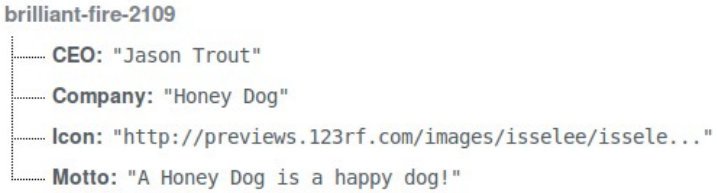
   "https://brilliant-fire-2109.firebaseio.com/"

   Click on "Manage App" to start adding data to it.



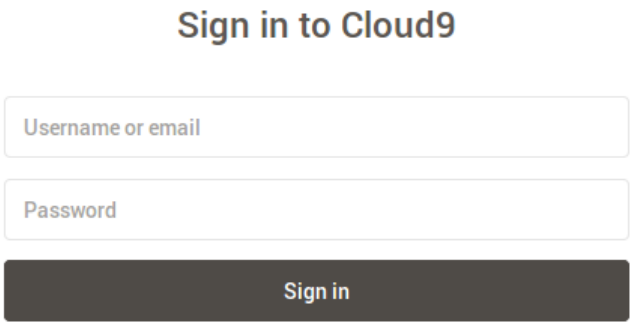
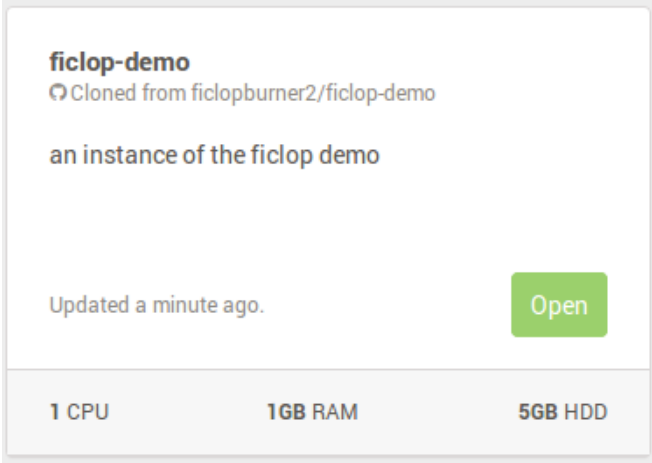
Firebase databases use the JSON format to store data. JSON data is stored as "Name: Value" pairs. The Name field is always a string. The Value field can be a number, a string, a boolean, an array, or an "object". Objects can contain anything the Value field can, including other objects.

See Wikipedia for more information on JSON. (https://en.wikipedia.org/wiki/JSON)

4. All Firebase databases have a root object matching the randomly generated name from the previous step. Mousing over this node reveals “Add” and “Remove” buttons, as shown. Pressing “Add” here will add a new entry to the root object. Click the “Add” button.	
5. Two fields, one for “name” and one for “value” will appear. Fill in the “name” field, but leave the “value” field alone.	
6. Now fill in the “value” field. Note that the “Add” button disappears. An entry with a value is, by definition, a non-object entry. To enter an object into the database, leave the “value” field blank.	
7. Add more data to the database until it has all the fields that the example shown has. Our Hello World example will read all these fields, so the database must include them for proper function. The CEO, Company, and Motto fields will be interpreted as strings, and can be anything. The Icon field is interpreted as a image URL; if you don't have a ready one hosted, we suggest a Google image search.	

Accessing Firebase data from app

This section assumes you have created a Cloud9 project. If you have not, refer to the "**Cloud9: Create a new project from GitHub code base**" section.

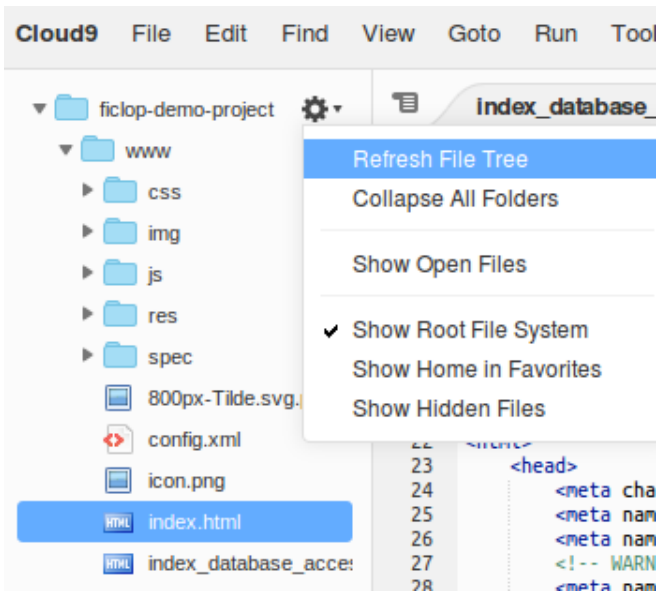
1. Visit https://c9.io/login, and log in if necessary.	
2. The project you created earlier should be visible on your cloud9 home page. Click on "Open" to open the Cloud9 workspace.	

3. The workspace has several .html files in the "www" directory to demonstrate various pieces of our app in isolation. To load one in particular we need to rename one of them to "index.html"

   Issue the following command in Cloud9's bash terminal

```
cp index_database_access.html index.html
```

4. Find index.html in the workspace's file tree and doubleclick it to open it. If it has not appeared, you may have to refresh the file tree display; click on the gear icon and select “Refresh File Tree”, as shown.



Index_database_access.html has 5 blocks that we will examine before continuing.

- This line pulls in the Firebase javascript library. For the documentation, and a primer, visit

  https://www.firebase.com/docs/web/quickstart.html

```
    <link rel="stylesheet" type="text/css" href="css/index.css" />
    <title>Ficlop Demo</title>
    <script src="https://cdn.firebase.com/js/client/2.4.0/firebase.js"></script>
</head>
<body unresolved onload="firebase_accesser()">
    <div>
```

- The <body> tag has added an “onload” attribute, so that our database access function is executed automatically.

```
</head>
<body unresolved onload="firebase_accesser()">
    <div>
        <p id="load1"> loading... </p>
```

- These lines define the html structure of the app. They have an id tag so that they can be changed after the database access function is called.

```
<body unresolved onload="firebase_accesser()">
    <div>
        <p id="load1"> loading... </p>
        <p id="load2"> loading... </p>
        <p id="load3"> loading... </p>
        <img id="img1" src="800px-Tilde.svg.png"/>

    </div>
    <script>
```

- These 3 functions are callback functions passed to the Firebase API. These functions are called when the database replies to a request for data that we previously initiate. Each function changes one of the tags mentioned in the previous item.

```
<script>
    function changeLoad1(databaseReply) {document.getElementById("load1").innerHTML = databaseReply.val();}
    function changeLoad2(databaseReply) {document.getElementById("load2").innerHTML = databaseReply.val();}
    function changeLoad3(databaseReply) {document.getElementById("load3").innerHTML = databaseReply.val();}
    function changeImg1(databaseReply)  {document.getElementById("img1") .src       = databaseReply.val();}

    function firebase_accesser() {
        var myFirebaseRef = new Firebase("https://brilliant-fire-2109.firebaseio.com/");
        myFirebaseRef.child("Company").on("value", changeLoad1);
```

- This is the database access function. After calling the Firebase constructor, it makes 4 database requests, one for each entry we put in our Firebase database. As an additional argument, it passes one of the callback functions mentioned in the previous item. When the database replies to each of our 4 requests, each callback function is called in turn.

```
function firebase_accesser() {
    var myFirebaseRef = new Firebase("https://brilliant-fire-2109.firebaseio.com/");
    myFirebaseRef.child("Company").on("value", changeLoad1);
    myFirebaseRef.child("Motto").on  ("value", changeLoad2);
    myFirebaseRef.child("CEO").on    ("value", changeLoad3);
    myFirebaseRef.child("Icon").on   ("value", changeImg1 );
}
```

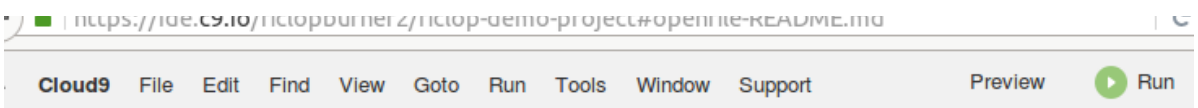
Now we will modify our app, view it, and record our changes to GitHub.

5. Make some trivial change to the app such that the produced app will reflect some change of yours.

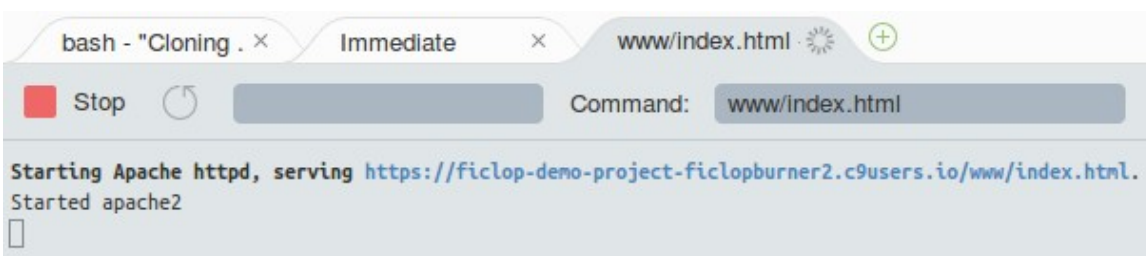
   This document assumes you add an extra text block at the beginning of the HTML section, as shown.

```
dy unresolved onload="firebase_accesser()">
 <div>
     Hello Firebase!
     <p id="load1"> loading... </p>
     <p id="load2"> loading... </p>
     <p id="load3"> loading... </p>
     <img id="img1" src="800px-Tilde.svg.png"/>

 </div>
 <script>
```

6. To open the app in the debugger, while viewing Index.html in the upper panel, hit the “Run” button on the top menu bar.



7. For an HTML file, clicking “Run” will start an apache service running our app as a webpage; this is visible in the lower panel.



Click on the hyperlink in the lower panel and select “Open” to view the app as a web page.

8. The app ought to look as shown. If not, try refreshing your webpage.	

The last thing we need to do is put our changes back into GitHub. GitHub is modified by means of the Git utility, a source control tool familiar to most Linux developers. See the “Git commands cheatsheet” in this document for the absolute bare essentials. Google can suggest many “full-blooded” Git tutorials for those interested in learning to use Git.

1. In this section, we created a new file (index.html), and modified it. To specify to Git that index.html has changed, issue the following command in Cloud9's bash terminal

   ```
   git add index.html
   ```

2. To commit any changes you've made to the local Git repository, issue the following command:

```
git commit -m “created index.html from index_database_access.html”
```

3. To push your changes from your local Git repository to GitHub, issue the following command.

   ```
   git push
   ```

   This command will require you to input your credentials to GitHub.

If successful, the bash terminal should appear as shown below

```
ficlopburner2:~/workspace/www (master) $ git add index.html
ficlopburner2:~/workspace/www (master) $ git commit -m "created index.html from index_database_access.html"
[master 8dd2184] created index.html from index_database_access.html
 1 file changed, 1 insertion(+), 1 deletion(-)
ficlopburner2:~/workspace/www (master) $ git push
Username for 'https://github.com': ficlopburner2
Password for 'https://ficlopburner2@github.com':
Counting objects: 11, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 1.91 KiB | 0 bytes/s, done.
Total 9 (delta 6), reused 0 (delta 0)
To https://github.com/ficlopburner2/ficlop-demo
   61bfa9d..8dd2184  master -> master
ficlopburner2:~/workspace/www (master) $
```

Git commands Cheatsheet

1. To specify to Git that a file has changed, use the following command in Cloud9's bash terminal:

   `git add {changed files}`

   You can use wildcards or regular expressions(?) when specifying files.

2. To view what changes a commit would make to the local Git repository, use the following command:

   `git status`

3. To cancel the effects of a `git add` on a particular file, use the following command:

   `git reset HEAD {files}`

4. To commit any changes you've made to the local Git repository, use the following command in Cloud9's bash terminal:

   `git commit –m “A message describing what changes there are in the commit you're about to add.”`

   Git requires some kind of message associated with each commit. You may skip the “-m” flag in the command, but if you do you must fill one in via 'nano' before Git will accept your commit.

5. To remove a file previously committed to the local repository, use the following command:

   `git rm --cached –r mydirectory` or `git rm --cached myfile`

   This command will NOT delete the file on the local drive. The following command will remove the previously commit file AND remove it from the local drive.

   `git rm –r mydirectory` or `git rm myfile`

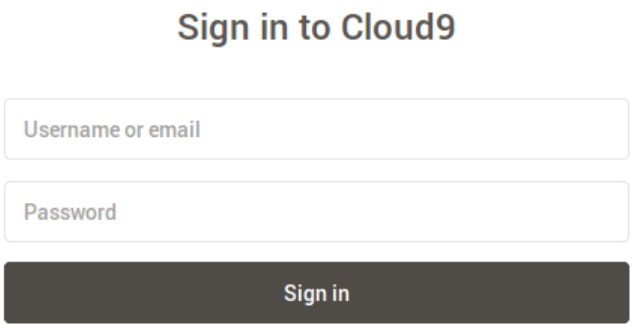
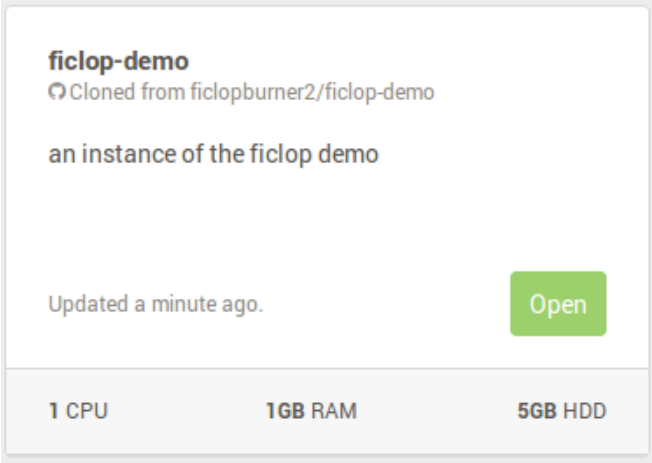
6. To push the changes in your local Git repository out to GitHub, use the following command in Cloud9's bash terminal:

   `git push`

   This command requires you to provide your GitHub credentials.

Hosting webapp on Firebase

In addition to database services, Firebase offers hosting services as well. This section describes how to host your app, making it available to end users.

1. Visit https://c9.io/login, and log in if necessary.	
2. Open the Ficlop demo project we've been working with.	

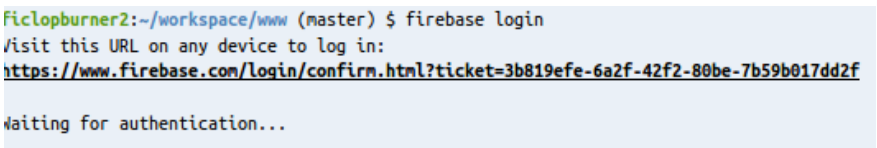
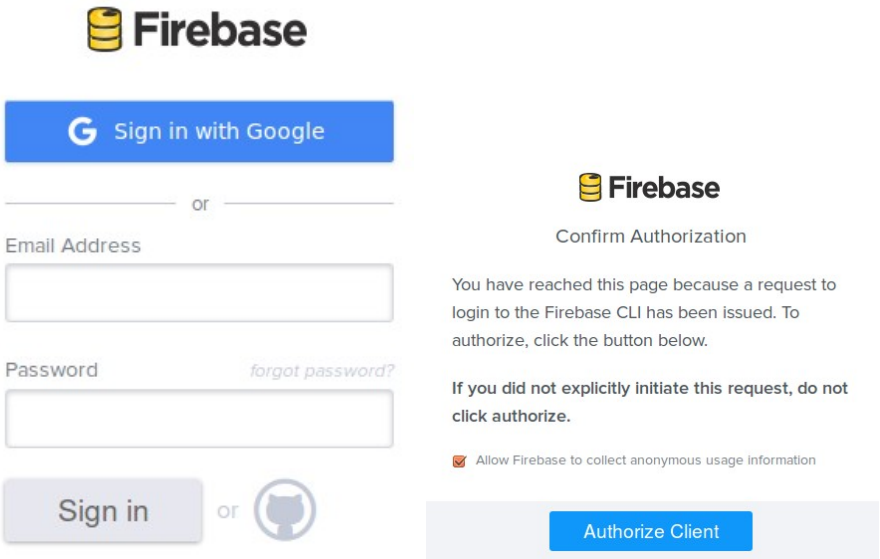
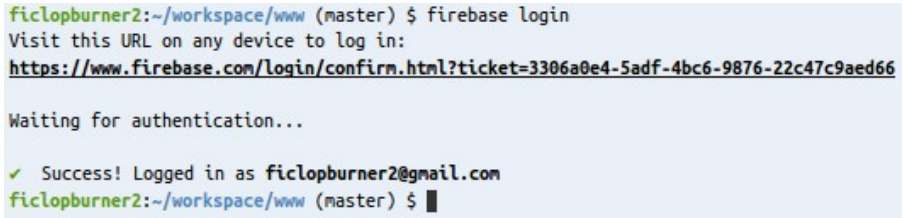
3. Firebase offers command-line utilities for Ubuntu systems. We can utilize these tools from the Cloud9 bash terminal.

   To install the Firebase tools, issue the following command:

   ```
   npm install -g firebase-tools
   ```

4. Firebase-tools must be associated with your Firebase account in order to function properly. You can log into your Firebase account by issuing the following command:

   ```
   firebase login
   ```

5. This command will prompt you to open an authentication page in a seperate tab, as shown. Click on the link and select “Open”.	
6. Provide your Firebase account credentials if necessary. If you are already logged in, this isn't necessary. After providing your credentials, Firebase will ask you to authorize the request we made with `firebase login`. Click “Authorize Client.”	
7. After authorizing the Cloud9 machine, return to Cloud9. It should appear as shown.	

8. Now we must configure firebase-tools to the specifics of our app. To begin configuring, issue the following commands.

   `cd /home/ubuntu/workspace`

   `firebase init`

9. `firebase init` will ask a series of questions. Provide the following answers:

   - For "**What Firebase do you want to use?**", specify the app that Firebase created for you when you created an account with them. In the example shown, this is "brilliant-fire-2109".
   - For "**What directory should be the public root?**" type in "www".

   This should be all that is required. The output should be as shown below.

```
ficlopburner2:~/workspace (master) $ firebase init
⚠  Initializing in a directory with 10 files

? What Firebase do you want to use? brilliant-fire-2109
? What directory should be the public root? www
Firebase initialized, configuration written to firebase.json
ficlopburner2:~/workspace (master) $
```

10. We may now deploy our app. Issue the following command.

    `firebase deploy`

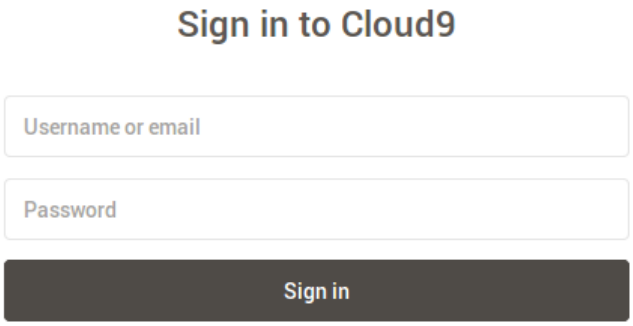
    The output will be as shown.

```
ficlopburner2:~/workspace/www (master) $ firebase deploy

=== Deploying to 'brilliant-fire-2109'...

i  deploying hosting
i  preparing www directory for upload...
✔  58 files uploaded successfully

✔  Deploy complete!

URL: https://brilliant-fire-2109.firebaseapp.com
Dashboard: https://brilliant-fire-2109.firebaseio.com

Visit the URL above or run firebase open
ficlopburner2:~/workspace/www (master) $
```

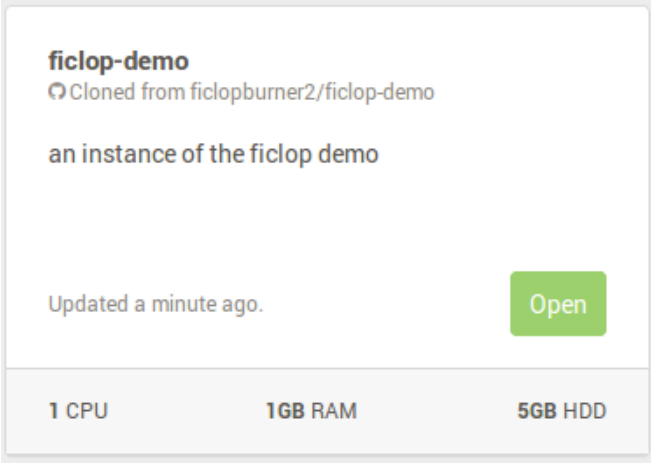
Building an app with Polymer components

This section assumes you have created a Cloud9 project. If you have not, refer to the "**Cloud9: Create a new project from GitHub code base**" section.

1. Visit https://c9.io/login, and log in if necessary.



2. The project you created earlier should be visible on your cloud9 home page. Click on "Open" to open the Cloud9 workspace.



3. Polymer elements are distributed using a tool called "bower". We must install bower on the PC running our Cloud9 workspace to get them.

   To install bower, run the following command in the Cloud9 Bash terminal.

```
npm install -g bower
```

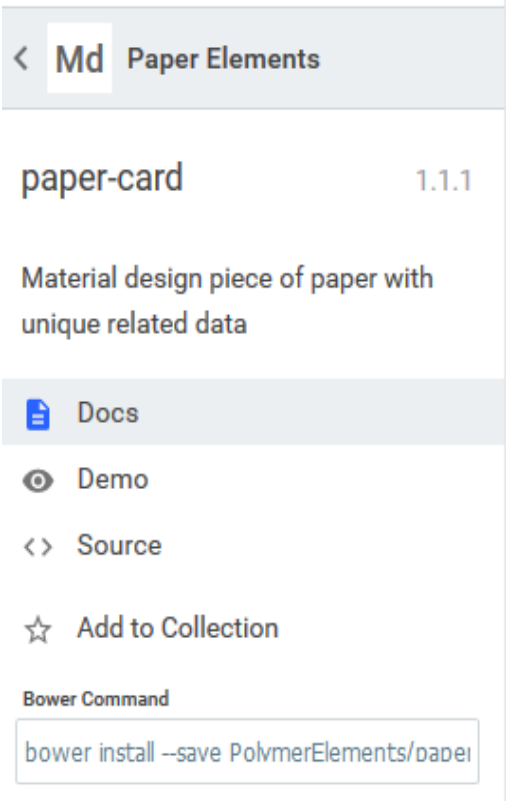
   The results in the terminal are shown below

```
ficlopburner2:~/workspace/www (master) $ npm install -g bower
/home/ubuntu/.nvm/versions/node/v4.2.4/bin/bower -> /home/ubuntu/.nvm/versions/node/v4.2.4/lib/node_mod
bower@1.7.7 /home/ubuntu/.nvm/versions/node/v4.2.4/lib/node_modules/bower
ficlopburner2:~/workspace/www (master) $ bower

Usage:

    bower <command> [<args>] [<options>]
```

4. One of the Polymer elements we'll be using with this app is the “paper-card” element. It is documented in the Polymer catalog, at https://elements.polymer-project.org/elements/paper-card. Of immediate interest on this page is the bower command used to install the component on the left, as shown.

   Issue the following commands to install the “paper-card” Polymer element.

   ```
   cd /home/ubuntu/workspace

   bower install --save
   PolymerElements/paper-card
   ```



5. We'll also be using the “paper-button” element. As with the paper-card, it is documented in the Polymer catalog, at https://elements.polymer-project.org/elements/paper-button.

   Download paper-button the same way you downloaded paper-card.

   These bower commands will create a directory called “bower_components” in /home/ubuntu/workspace. After downloading “paper-button” and “paper-card”, the directory should appear as shown below.

```
ficlopburner2:~/workspace (master) $ cd bower_components/
ficlopburner2:~/workspace/bower_components (master) $ ls
font-roboto/                    iron-icon/                  marked-element/     paper-styles/
iron-a11y-keys-behavior/        iron-icons/                 paper-behaviors/    polymer/
iron-autogrow-textarea/         iron-iconset-svg/           paper-button/       prism/
iron-behaviors/                 iron-image/                 paper-card/         prism-element/
iron-checked-element-behavior/  iron-input/                 paper-icon-button/  webcomponentsjs/
iron-demo-helpers/              iron-meta/                  paper-input/
iron-flex-layout/               iron-validatable-behavior/  paper-material/
iron-form-element-behavior/     marked/                     paper-ripple/
ficlopburner2:~/workspace/bower_components (master) $
```

Polymer and Phonegap-Build do not play well together right out of the box; our best guess is that Phonegap-Build barfs when handling repositories that contain multiple files called “index.html”, and Polymer elements downloaded via bower contain these by default. We have found a workaround for this problem which we will be describing in this section, but the following rules must be observed for the workaround to be effective.

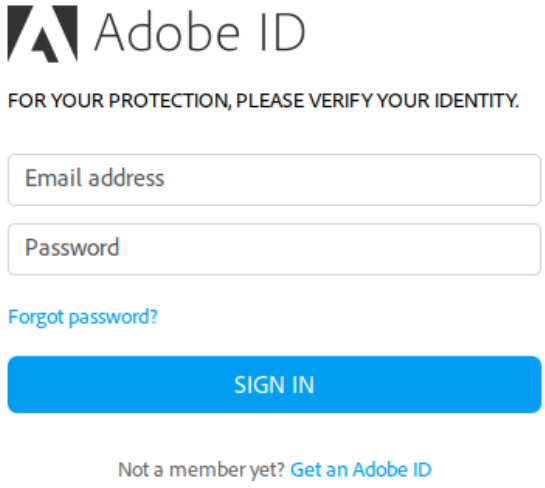
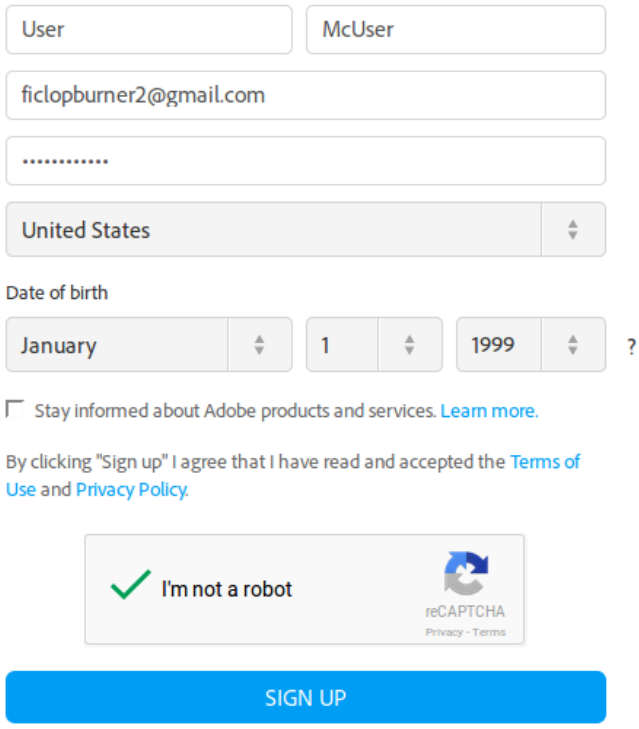
- Polymer components **cannot** go into Git.
- No files other than the outermost wrapper for your file can be called “index.html”.
- The wrapper “index.html” must be **vulvanized** using the vulcanize command line utility.

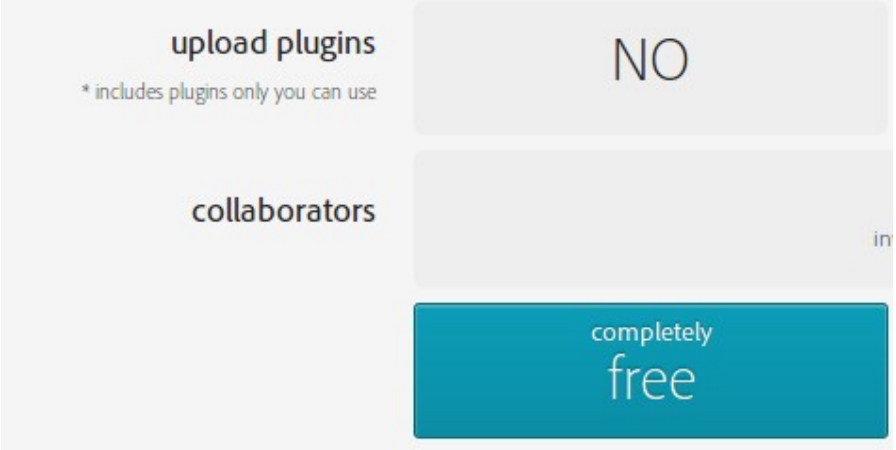
6. By default, anything under the “workspace” directory is under Git source control. Git can be set to ignore certain files or directories using “.gitignore” files. Issue the following commands to ensure the bower_components directory is not uploaded to git.
 	
`cd /home/ubuntu/workspace`
 	
`echo “bower_components” >> .gitignore`	

7.	
8.	

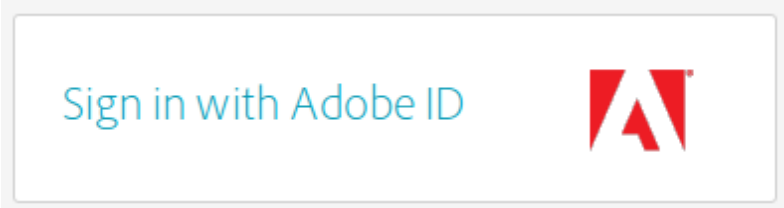
Phonegap-Build: New Account Creation

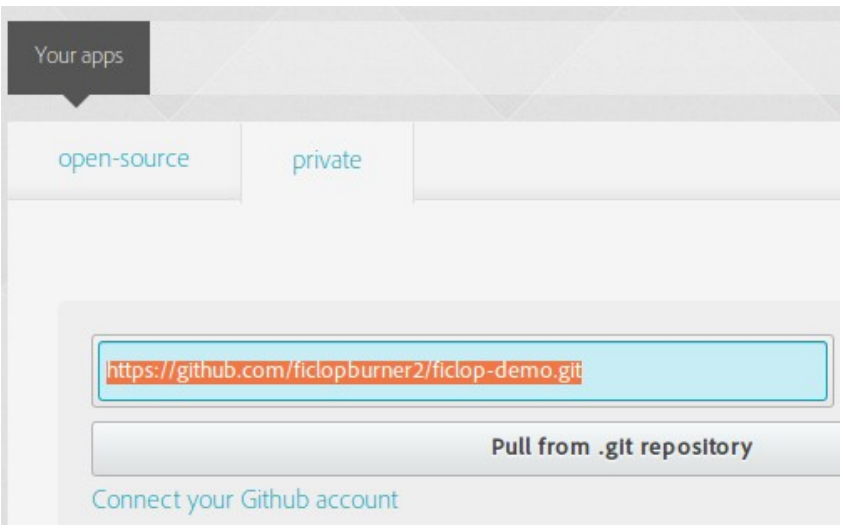
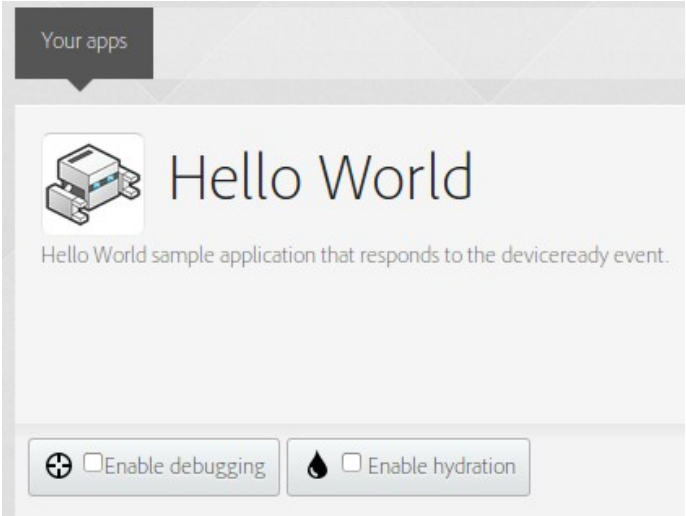
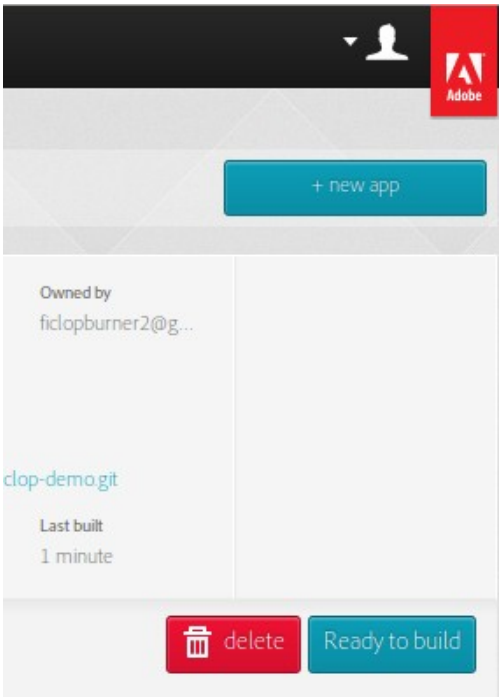
Phonegap-build is an Adobe service, and we need an Adobe account before we can use it.

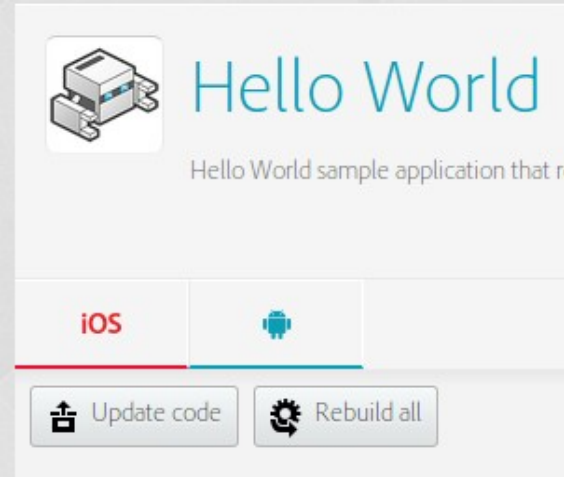
1. Visit the Adobe Account creation page at *https://accounts.adobe.com/* and click “Get an Adobe ID”.	
2. Fill out the form and click “Sign Up”.	
3. Adobe requires you to confirm the e-mail address you provided before using Phonegap build. You should receive the e-mail momentarily after clicking “Sign Up”.	

4. Visit Phonegap-Build's page (https://build.phonegap.com/) and click “Sign up”.	
5. The page that loads asks you to pick a payment plan for your account. For our purposes, the free account is sufficient.	

Phonegap-Build: Building our App

1. Visit *https://build.phonegap.com/* and click “Sign in”.	
2. When the page loads, click on “Sign in with Adobe ID”	
3. When the page loads, fill in your Adobe account credentials.	

4. You should see your “Apps” page with no apps present, as shown. To download your app to Phonegap-Build, fill in the git repository associated with your GitHub account and click “Pull from .git repository”. Note that this is NOT the same thing as the URL of your GitHub account. See the end of the GitHub section for more details. 5.	
6. If the download was successful, you'll see a new “Hello World” application in your apps page, as shown.	
1. Your apps page also has a “Ready to Build” button that will kick off a build. Click it. This button only appears if the app has never been built before. Subsequent builds use a slightly different presentation.	

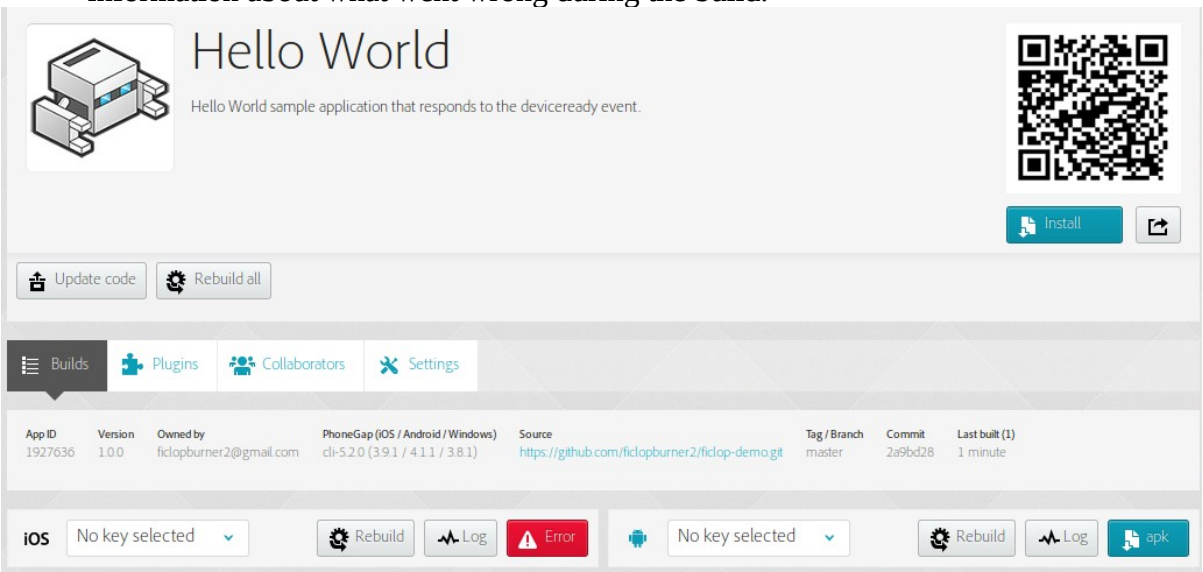
2. When your app starts building, status bars for iOS and Android will appear, and show a grey “In progress” animation; when the build succeeds or fails, they will stabilize; red indicates a failed build, blue a successful one, as shown. Once at least one build has been completed, the “Update code” and “Rebuild all” buttons will appear. • “Update code” will re-download your app code from GitHub; this must be done manually when you make changes to GitHub. • “Rebuild All” will attempt to build your app for all supported phone operating systems (iOS and Android).	
3. Once your app has been built at least once, you can get more detailed controls by clicking on the app name.	

4. The detailed app screen is shown below. This page has several useful features.
   - The settings tab allows you to customize the outermost features of the app (like name and description), and also allows you to delete the app (as we're only permitted one app, this is useful for sanity testing).

   - Along the bottom are three buttons. The first two are the “Rebuild” and “Log” buttons. These two buttons will attempt another build, and view the compiler log for the last build, respectively.

     If the build was successful, the third button will be blue, and will queue your app to be downloaded. The downloaded file can be run as a native iOS or Android application.

     If the build was not successful, the third button will be red, and will give you more information about what went wrong during the build.



5. Visit https://build.phonegap.com/apps on your phone and click the download link to download the app directly to your phone.