

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.

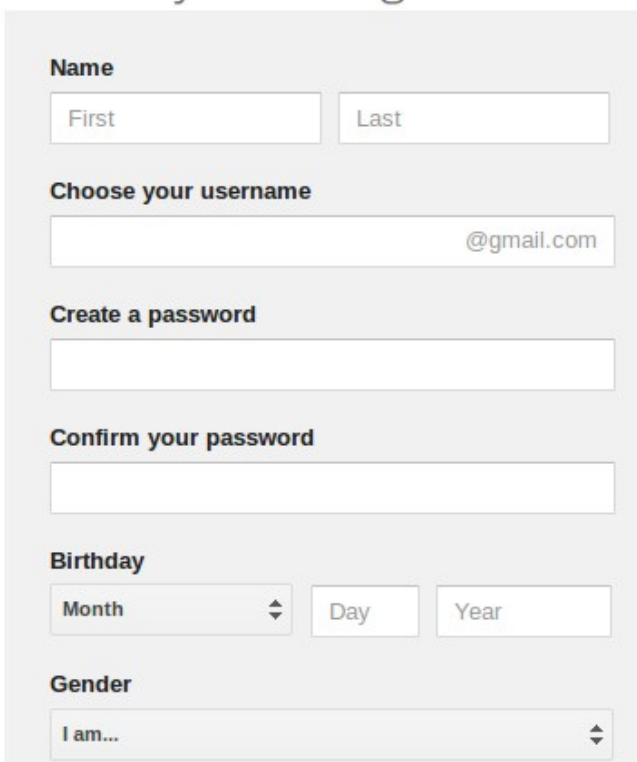
- For password, use the same string as you used for username, but reverse the order of the words. Eg

username: ficlopBurner1

password: 1BurnerFiclop

- Birthday and gender are irrelevant
- Leave all other fields blank
- Complete the image captcha and hit “Next step”.

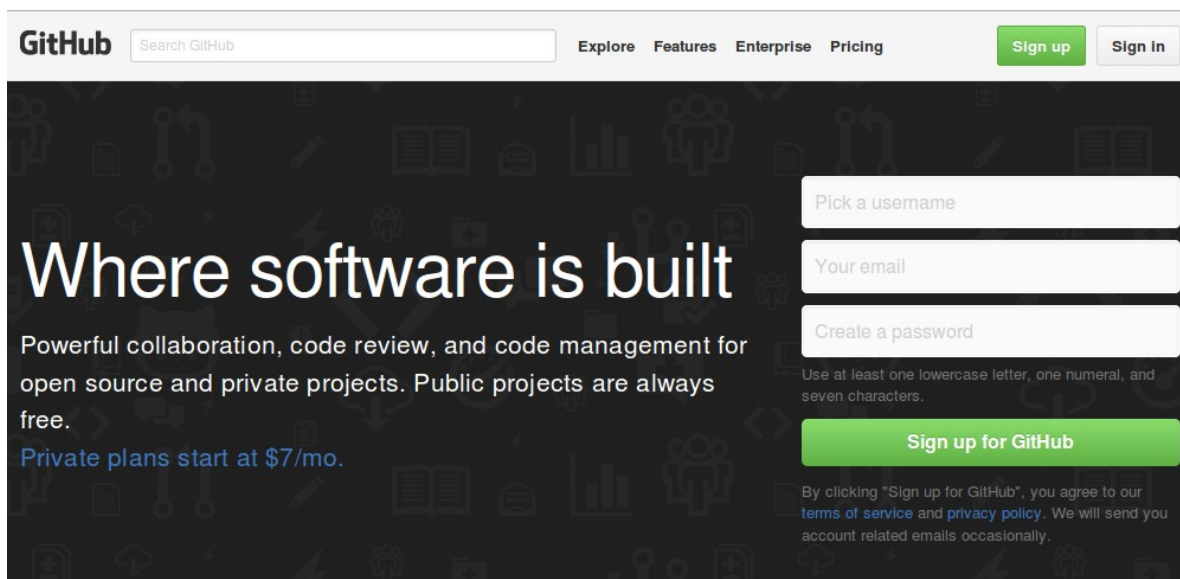
Create your Google Account



The screenshot shows the 'Create your Google Account' form. It includes fields for 'Name' (First and Last), 'Choose your username' (with a placeholder '@gmail.com'), 'Create a password', 'Confirm your password', 'Birthday' (Month, Day, Year), and 'Gender' (a dropdown menu with 'I am...' selected).

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”



GitHub

Search GitHub

Explore Features Enterprise Pricing

Sign up Sign in

Where software is built

Powerful collaboration, code review, and code management for open source and private projects. Public projects are always free.

Private plans start at \$7/mo.

Pick a username

Your email

Create a password

Use at least one lowercase letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We will send you account related emails occasionally.

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”.

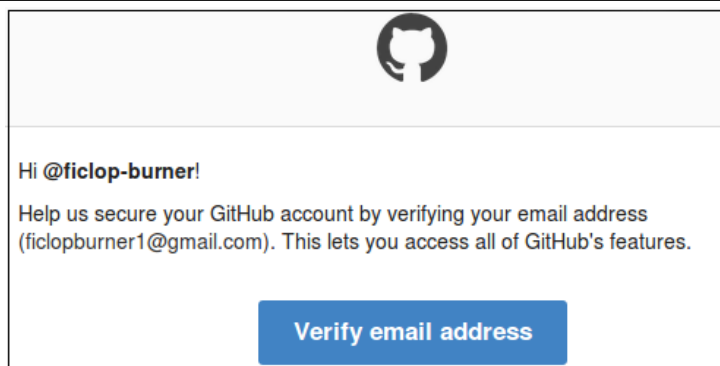
Choose your personal plan

Plan	Cost	Private repositories	
Large	\$50/month	50	Choose
Medium	\$22/month	20	Choose
Small	\$12/month	10	Choose
Micro	\$7/month	5	Choose
Free	\$0/month	0	Chosen

⋮

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

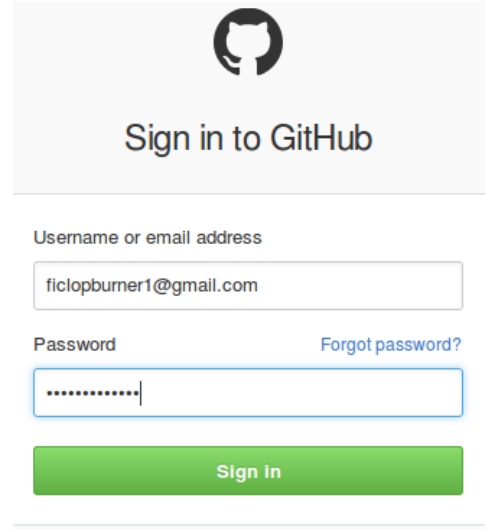
Hi @ficlop-burner!

Help us secure your GitHub account by verifying your email address (ficlopburner1@gmail.com). This lets you access all of GitHub's features.

Verify email address

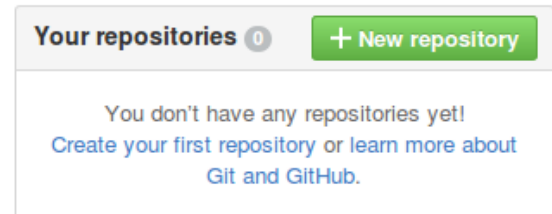
Github : Clone Ficlop repository from Ficlop Master

1. Visit <https://github.com/>. If necessary, log in.



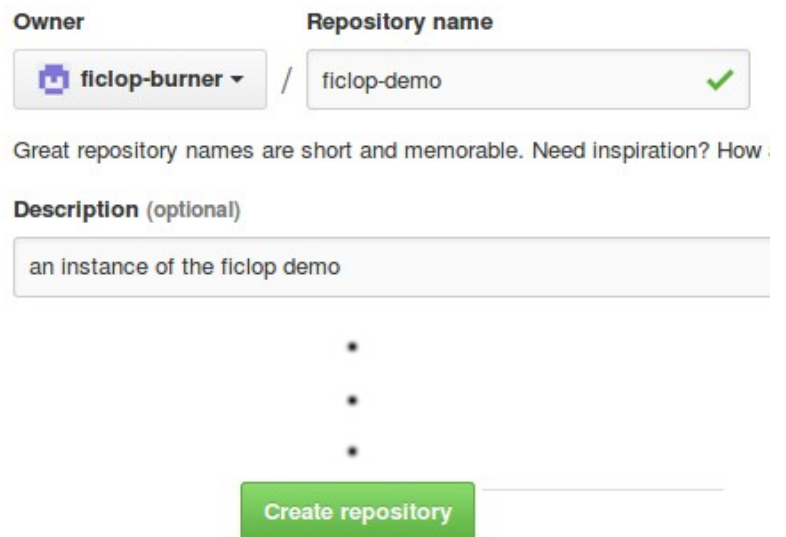
The image shows the GitHub sign-in page. At the top is the GitHub logo and the text "Sign in to GitHub". Below this are two input fields: "Username or email address" with the value "ficlopburner1@gmail.com" and "Password" with masked characters ".....". A link "Forgot password?" is next to the password field. At the bottom is a green "Sign in" button.

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.



The image shows the GitHub account homepage. At the top, it says "Your repositories 0" next to a green "+ New repository" button. Below this is a message: "You don't have any repositories yet! Create your first repository or learn more about Git and GitHub."

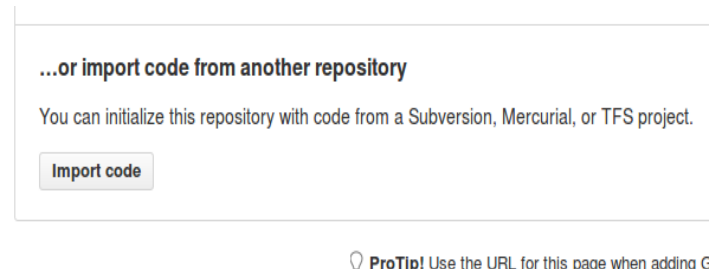
2. Give the new repository a name and a brief description. After that hit “Create repository”; we'll be cloning another repository to get you started, so we can skip repository initialization.



The image shows the GitHub "Create repository" page. It has two main sections: "Owner" and "Repository name". The "Owner" section shows a dropdown menu with "ficlop-burner" selected. The "Repository name" section shows a text input with "ficlop-demo" and a green checkmark. Below these is a message: "Great repository names are short and memorable. Need inspiration? How". Underneath is a "Description (optional)" section with a text input containing "an instance of the ficlop demo". At the bottom is a green "Create repository" button.

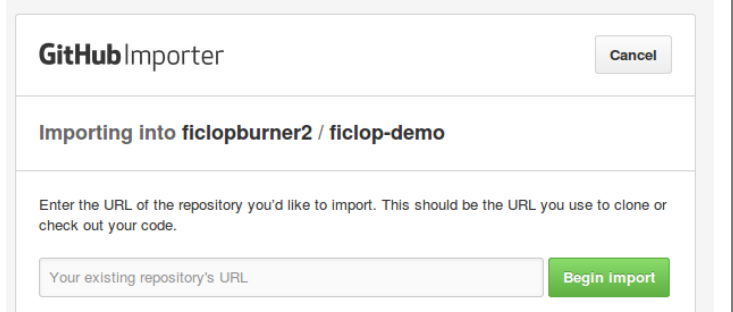
3. Your new repository is currently empty. We're going to duplicate an existing repository to get you started.

Hit the “Import code” button at the bottom of the page.

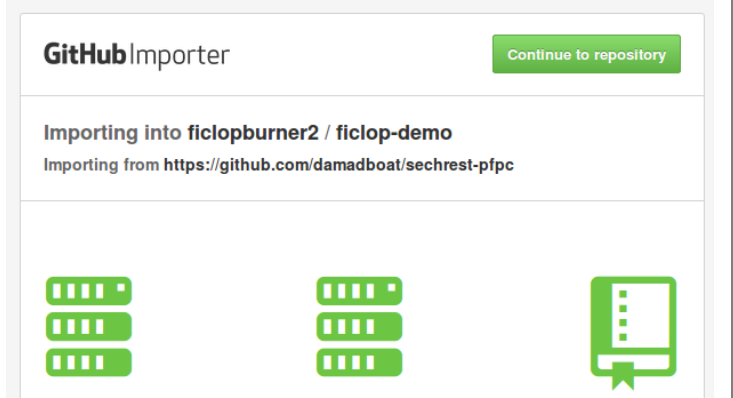


The image shows the GitHub "Import code" page. It has a heading "...or import code from another repository" and a subtext "You can initialize this repository with code from a Subversion, Mercurial, or TFS project." Below this is a button labeled "Import code". At the bottom right is a "ProTip!" icon and text: "Use the URL for this page when adding C".

4. When the “GitHub Importer” page loads, fill in <https://github.com/damadboat/ficlop-demo-master> and click “Begin Import”.



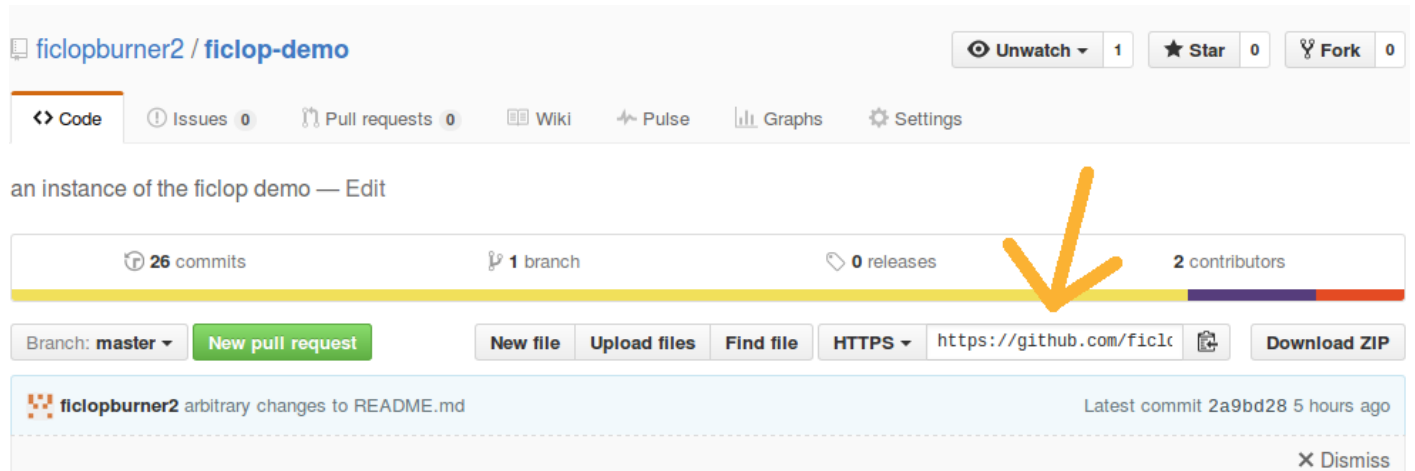
5. If the import process is successful, after a few moments the page will appear as shown. Click “Continue to repository” to see what was imported.



6. If the import process was successful, your repository will appear as shown below.

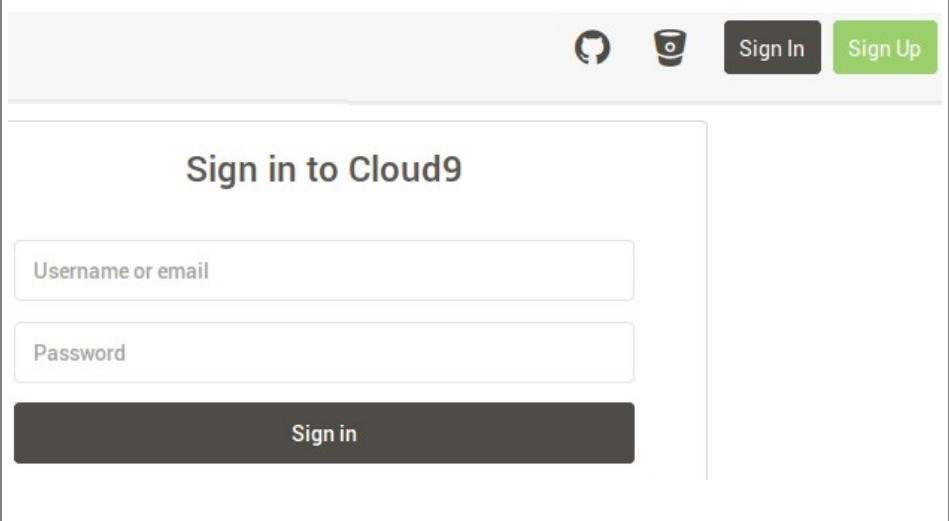
📁 firebase	making repo clean for documentation purposes.	10 minutes ago
📁 www	making repo clean for documentation purposes.	10 minutes ago
📄 LICENSE	one more time	20 days ago
📄 NOTICE	one more time	20 days ago
📄 README.md	making repo clean for documentation purposes.	10 minutes ago
📄 labbook.pdf	added labbook.pdf, removed old .odt files	8 days ago

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.

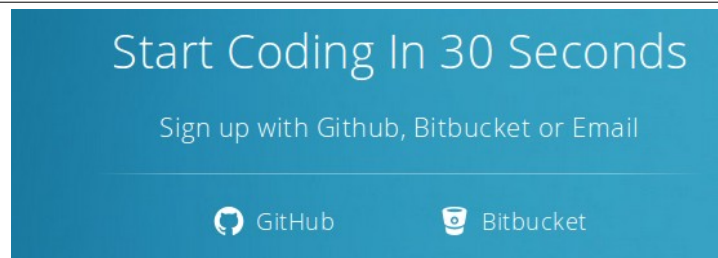


Cloud9: New Account Creation

1. Visit cloud9's login page (<https://c9.io/login>) and click "Sign Up".

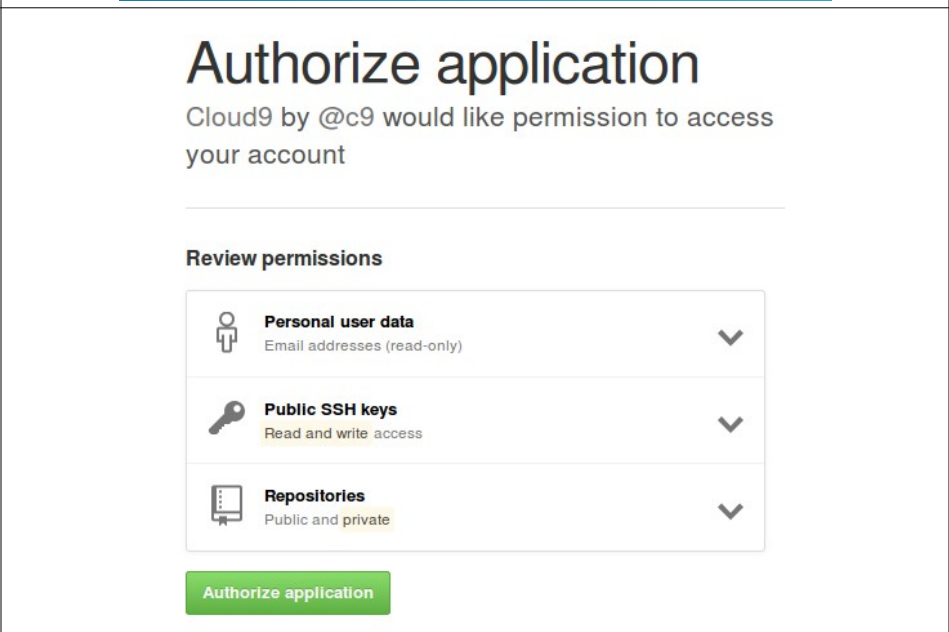
The image shows the Cloud9 login page. At the top right, there are icons for GitHub and Bitbucket, and two buttons: "Sign In" and "Sign Up". The main heading is "Sign in to Cloud9". Below it are two input fields: "Username or email" and "Password". At the bottom is a "Sign in" button.

2. After the page shown loads, click the button marked "GitHub".

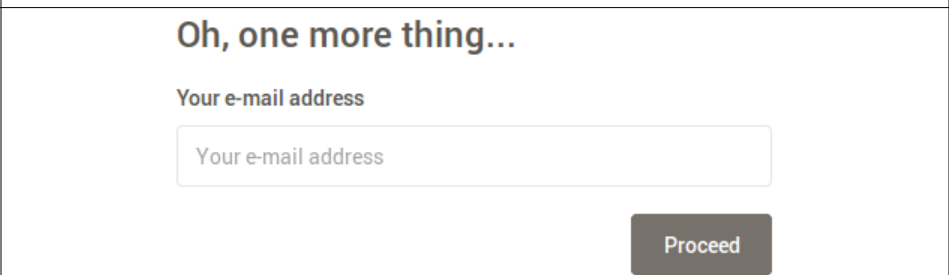
The image shows a blue banner with the text "Start Coding In 30 Seconds". Below it, it says "Sign up with Github, Bitbucket or Email". At the bottom, there are two buttons: "GitHub" and "Bitbucket".

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, the page will instead present a login form for github. After logging in, you should see the page shown.

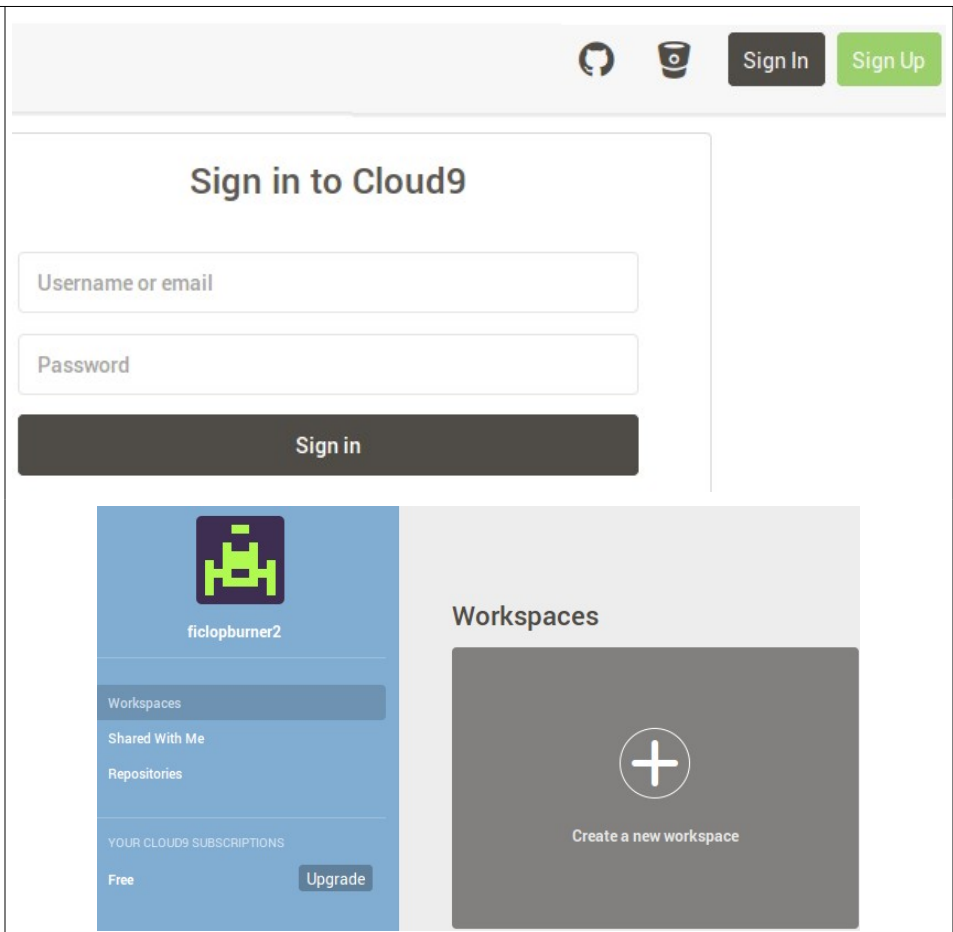
The image shows the "Authorize application" page. The heading is "Authorize application". Below it, it says "Cloud9 by @c9 would like permission to access your account". There is a section titled "Review permissions" with three items: "Personal user data" (Email addresses (read-only)), "Public SSH keys" (Read and write access), and "Repositories" (Public and private). At the bottom is a green button labeled "Authorize application".

4. After giving permission to Cloud9, Cloud9 may ask you to provide your e-mail address in a pop-up window.

The image shows a page titled "Oh, one more thing...". Below the title, it says "Your e-mail address". There is an input field for the email address. At the bottom right is a "Proceed" button.

Cloud9: Create a new project from GitHub code base

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



The screenshot displays the Cloud9 web interface. At the top, there are navigation links for GitHub and AWS, along with 'Sign In' and 'Sign Up' buttons. The main section is titled 'Sign in to Cloud9' and contains a login form with fields for 'Username or email' and 'Password', followed by a 'Sign in' button. Below the login form, the dashboard is visible, featuring a user profile for 'ficlopburner2' with a green robot icon. The dashboard includes a sidebar with links to 'Workspaces', 'Shared With Me', and 'Repositories'. The main content area shows 'Workspaces' with a large grey box containing a plus sign and the text 'Create a new workspace'. At the bottom, there is a section for 'YOUR CLOUD9 SUBSCRIPTIONS' showing a 'Free' plan with an 'Upgrade' button.

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.

Create a new workspace

Owner

ficlopburner2

Workspace name

ficlop-demo

Description

an instance of the ficlop demo

Hosted workspace

Remote SSH Workspace



Private

This is a workspace for your eyes only



Public

This will create a workspace for everybody to see

Clone from Git or Mercurial URL (optional)

https://github.com/ficlopburner2/ficlop-demo

Choose a template



Custom



HTML5



Node.js



Meteor



PHP, Apache & ...



Python

django

Django



Ruby



C++



Wordpress

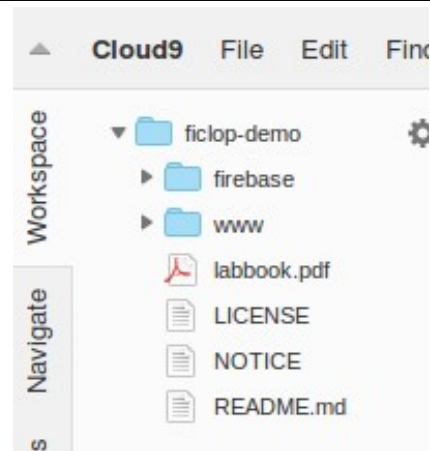


Rails Tutorial

Create workspace

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.

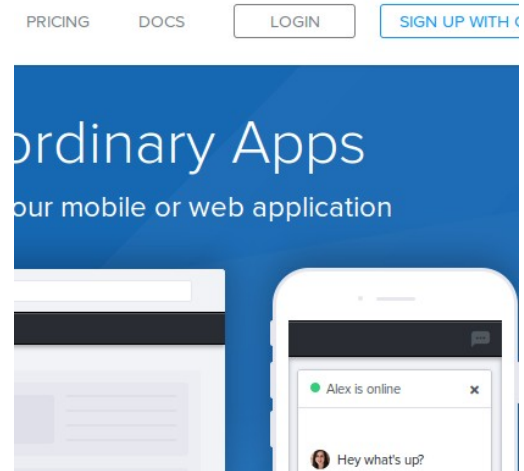
5. Also of interest is Cloud9's "bash" terminal, shown here. This allows any bash commands available on a typical Linux PC.



```
bash - "Cloning" . x Immediate x +
Aborting commit due to empty commit message.
ficlopburner2:~/workspace (master) $ ls
COPYRIGHT LICENSE NOTICE README.md newbie_notes.odt outline_puff.odt www/
ficlopburner2:~/workspace (master) $ cd ..
ficlopburner2:~ $ ls
lib/ workspace/
ficlopburner2:~ $ pwd
/home/ubuntu
ficlopburner2:~ $
```

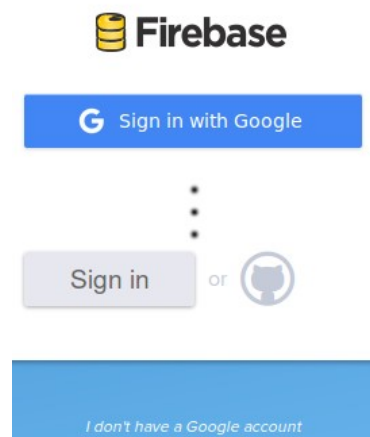

Firebase: Account Creation

1. Visit <https://www.firebase.com/> and click “Login”.

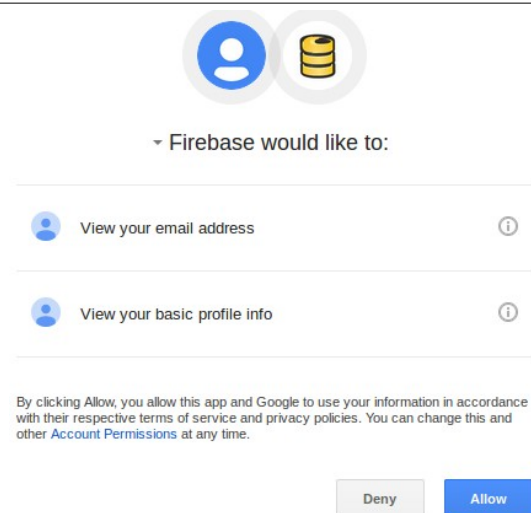


2. If you have a Gmail account, click “Sign in with Google”, and this will associate your firebase account with your g-mail account.

If you don't have a g-mail account, click “I don't have a Google account”, at the bottom of the page. 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 some stub resources 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.

3. A page showing the database contents will load; at present, there are no database entries, so this page is mostly blank.

The screenshot displays the Firebase web interface. At the top, the Firebase logo is visible. Below it is a 'Sign in with Google' button. A separator line with 'or' follows. There are input fields for 'Email Address' and 'Password', with a 'forgot password?' link next to the password field. Below these is a 'Sign in' button and a GitHub logo. The main content area is titled 'MY FIRST APP' with a settings gear icon. It shows 'DEVELOPMENT ONLY - FREE PLAN' and the app's URL 'brilliant-fire-2109.firebaseio.com'. There is a 'Set up Hosting' link. Two buttons, 'Manage App' and 'Upgrade Plan', are present. Below them is an 'Add a collaborator' section with a user icon. At the bottom, there's a 'hboard' section titled 'VIEWING MY FIRST APP'. It has 'Expand Data' and 'Collapse Data' buttons. A breadcrumb shows 'BRILLIANT-FIRE-2109'. At the bottom right, a field shows 'brilliant-fire-2109: null' with green and red icons.

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.

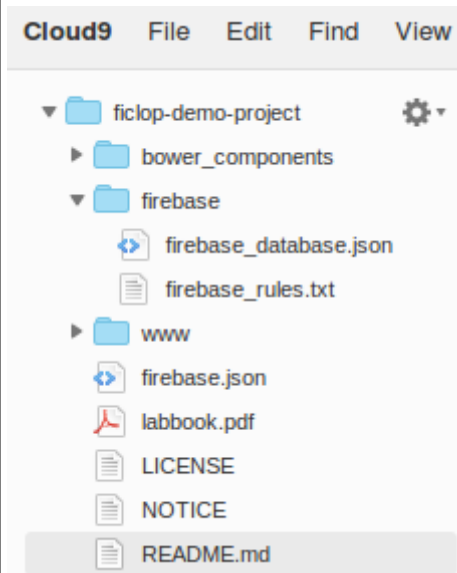
See Wikipedia for more information on JSON. (<https://en.wikipedia.org/wiki/JSON>)

Although Firebase allows you to enter JSON pairs by hand, it is much easier to populate the database by importing your data from an existing JSON document. We have provided a JSON document to get you started as part of the ficlop repository.

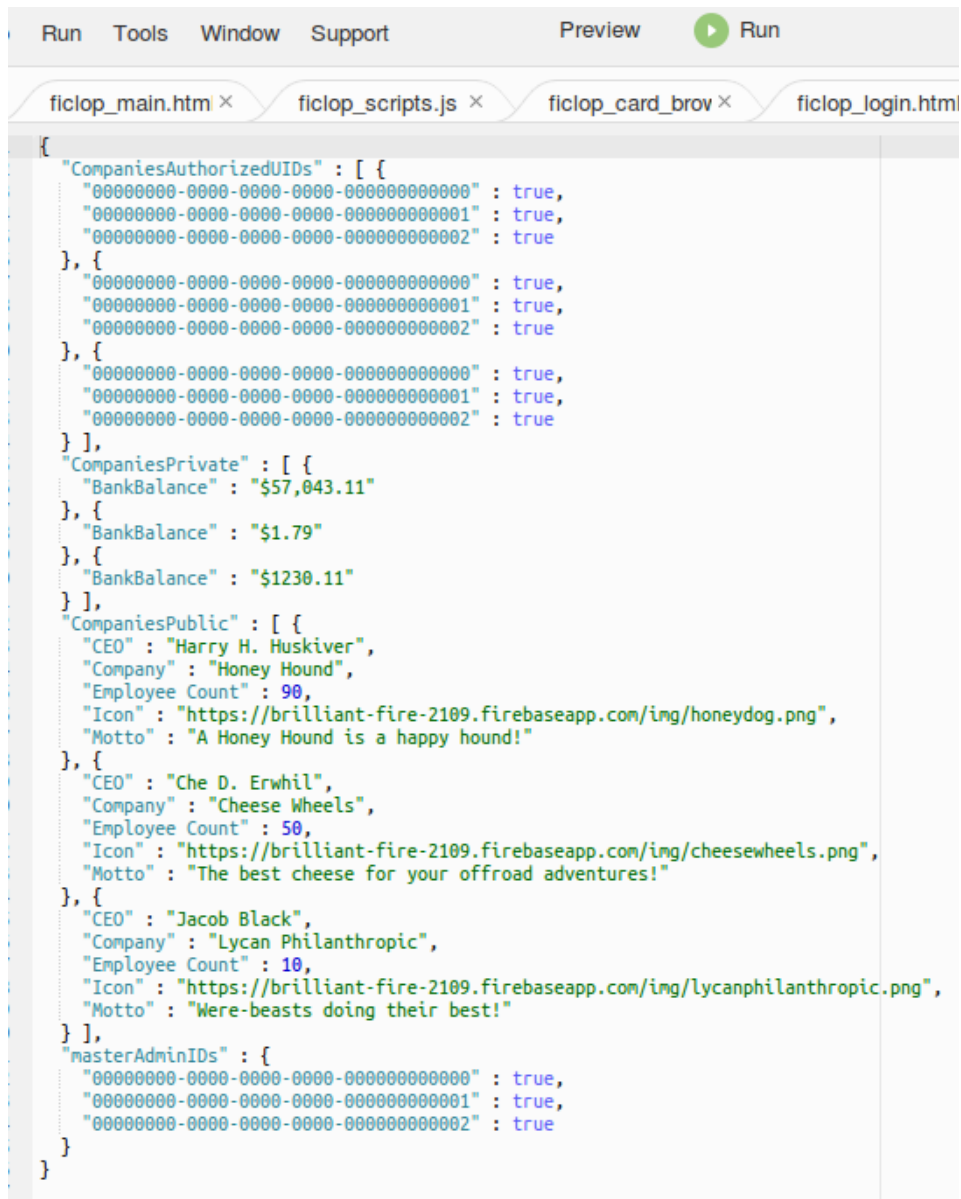
4. Visit https://c9.io/login , and log in if necessary.	<div><h3>Sign in to Cloud9</h3><div><input type="text" value="Username or email"/></div><div><input type="password" value="Password"/></div><div>Sign in</div></div>
5. The project you created earlier should be visible on your cloud9 home page. Click on “Open” to open the Cloud9 workspace.	<div><div>ficlop-demo Cloned from ficlopburner2/ficlop-demo an instance of the ficlop demo</div><div>Updated a minute ago.<div>Open</div></div><div><div>1 CPU</div><div>1GB RAM</div><div>5GB HDD</div></div></div>

6. The file you want is called “firebase_database.json”, which is under the “firebase” directory. It should appear in the workspace browser as shown.

Double click on the file entry to open the file.



The contents of firebase_database.json is shown below.

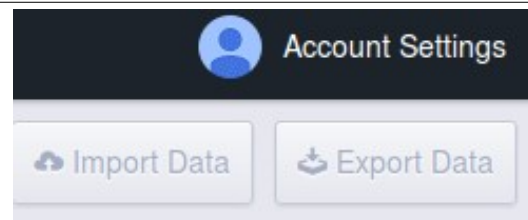


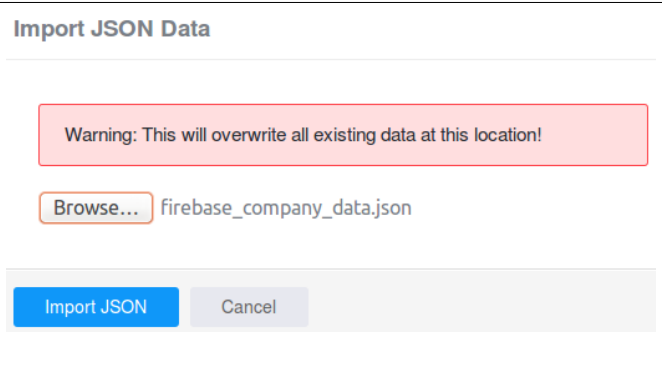
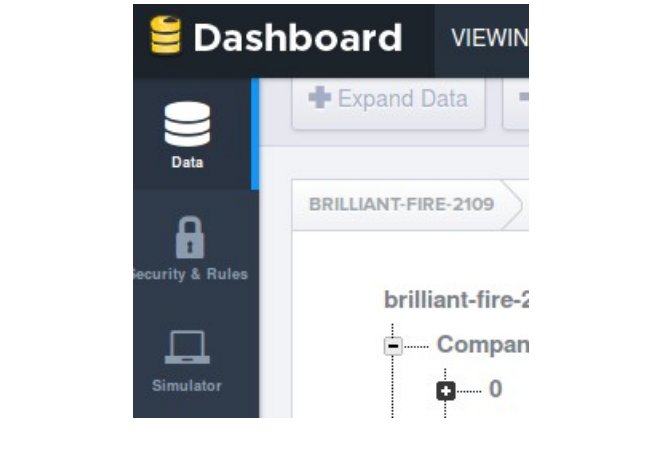
```
{
  "CompaniesAuthorizedUIDs" : [ {
    "00000000-0000-0000-0000-000000000000" : true,
    "00000000-0000-0000-0000-000000000001" : true,
    "00000000-0000-0000-0000-000000000002" : true
  }, {
    "00000000-0000-0000-0000-000000000000" : true,
    "00000000-0000-0000-0000-000000000001" : true,
    "00000000-0000-0000-0000-000000000002" : true
  }, {
    "00000000-0000-0000-0000-000000000000" : true,
    "00000000-0000-0000-0000-000000000001" : true,
    "00000000-0000-0000-0000-000000000002" : true
  } ],
  "CompaniesPrivate" : [ {
    "BankBalance" : "$57,043.11"
  }, {
    "BankBalance" : "$1.79"
  }, {
    "BankBalance" : "$1230.11"
  } ],
  "CompaniesPublic" : [ {
    "CEO" : "Harry H. Huskiver",
    "Company" : "Honey Hound",
    "Employee Count" : 90,
    "Icon" : "https://brilliant-fire-2109.firebaseio.com/img/honeydog.png",
    "Motto" : "A Honey Hound is a happy hound!"
  }, {
    "CEO" : "Che D. Erwhil",
    "Company" : "Cheese Wheels",
    "Employee Count" : 50,
    "Icon" : "https://brilliant-fire-2109.firebaseio.com/img/cheesewheels.png",
    "Motto" : "The best cheese for your offroad adventures!"
  }, {
    "CEO" : "Jacob Black",
    "Company" : "Lycan Philanthropic",
    "Employee Count" : 10,
    "Icon" : "https://brilliant-fire-2109.firebaseio.com/img/lycanphilanthropic.png",
    "Motto" : "Were-beasts doing their best!"
  } ],
  "masterAdminIDs" : {
    "00000000-0000-0000-0000-000000000000" : true,
    "00000000-0000-0000-0000-000000000001" : true,
    "00000000-0000-0000-0000-000000000002" : true
  }
}
```

If you want, customize some of the string fields in this file (in particular, under the “CompaniesPublic” section) so you can observe the changes in your generated app.

7. Firebase can only take an import from a local file on your computer, so we have to download firebase_database.json in order to import it. Right click on firebase_database.json's icon in the workspace navigator and click “Download”.

8. After the download finishes, return to your Firebase page and click “Import Data”.



<p>9. Use the “Browse” button to locate and specify “firebase_database.json”. Then hit “Import JSON”.</p> <p>This process can take a while. If it seems to get stuck, try refreshing the page a couple times.</p>	
<p>10. After Firebase completes loading the JSON document, the data from the text file will appear in the web page. This page is always available under the “Data” tab when you are looking at your project dashboard.</p>	

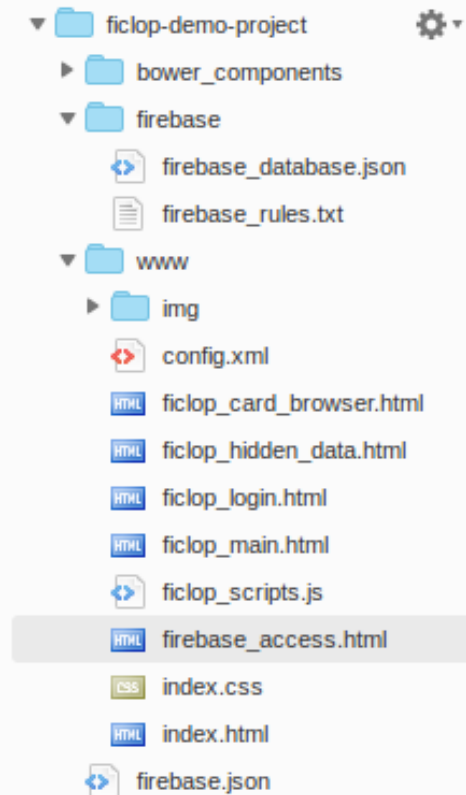
For the moment, this database is totally unsecured; anybody can read or write anything they want from it. We'll cover how to go about securing it in the “**Firestore: Adding User Authorization**” section. In the next section, we'll show you how to read data from this (unsecured) database.

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.

<p>1. Visit https://c9.io/login, and log in if necessary.</p>	<div><h3>Sign in to Cloud9</h3><div><input type="text" value="Username or email"/></div><div><input type="password" value="Password"/></div><div>Sign in</div></div>
<p>2. The project you created earlier should be visible on your cloud9 home page. Click on “Open” to open the Cloud9 workspace.</p>	<div><div><div>ficlop-demo Cloned from ficlopburner2/ficlop-demo an instance of the ficlop demo Updated a minute ago.</div><div>Open</div></div><div><div>1 CPU</div><div>1GB RAM</div><div>5GB HDD</div></div></div>

- The file “firebase_access.html” is a simple demo of how to access an unsecured firebase. Find firebase_access.html in the workspace's file tree and double click it to open it.



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

<ul style="list-style-type: none"> This line pulls in the Firebase javascript library. For the documentation, and a primer, visit https://www.firebase.com/docs/web/quickstart.html 	<pre> <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> <p id="load1"> loading... </p> </div> </pre>
<ul style="list-style-type: none"> The <body> tag has added an “onload” attribute, so that our database access function is executed automatically. 	<pre> </head> <body unresolved onload="firebase_accesser()"> <div> <p id="load1"> loading... </p> <p id="load2"> loading... </p> </div> </pre>
<ul style="list-style-type: none"> 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. 	<pre> <body unresolved onload="firebase_accesser()"> <div> <p id="load1"> loading... </p> <p id="load2"> loading... </p> <p id="load3"> loading... </p> </div> </script> </pre>

- 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/");
```

- 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("CompaniesPublic/0/Company").on("value", changeLoad1);
  myFirebaseRef.child("CompaniesPublic/0/Motto").on("value", changeLoad2);
  myFirebaseRef.child("CompaniesPublic/0/CEO").on("value", changeLoad3);
  myFirebaseRef.child("CompaniesPublic/0/Icon").on("value", changeImg1);
}
</script>
```

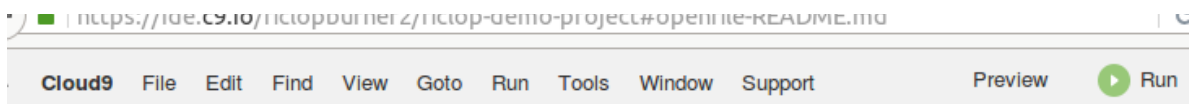
Now we will modify our app, view it, and record our changes to GitHub.

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

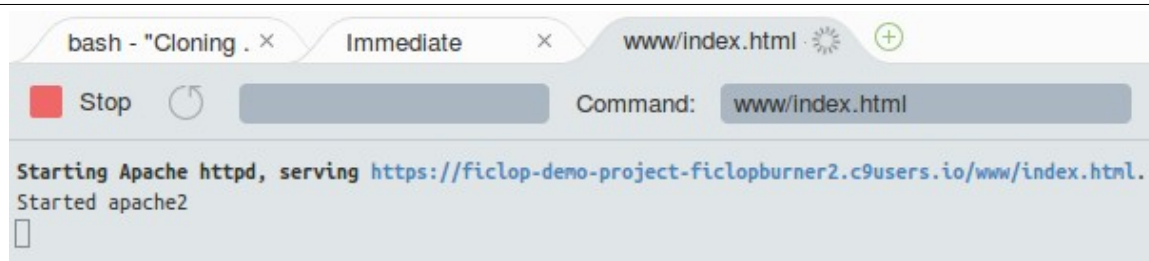
This document assumes you add an extra text block at the beginning of the HTML section, as shown.

```
---
<div unresolved onload="firebase_accesser()">
<div>
  Hello Firebase!
  <p id="load1"> loading... </p>
  <p id="load2"> loading... </p>
  <p id="load3"> loading... </p>
  
</div>
</script>
```

- To open the app in the debugger, while viewing `firebase_access.html` in the file viewer, hit the “Run” button on the top menu bar.

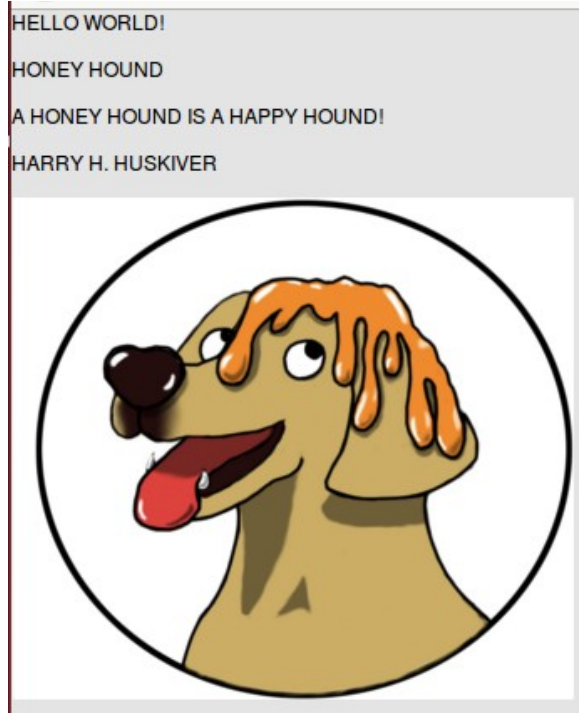


- For an HTML file, clicking “Run” will start an apache service running our app as a webpage; this is visible in the lower panel.



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 cheat sheet” section 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 modified one file (firebase_access.html). To specify to Git that firebase_access.html has changed, issue the following command in Cloud9's bash terminal

```
git add firebase_access.html
```

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

```
git commit -m "added hello world to firebase_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 Cheat Sheet

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.

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.

Sign in to Cloud9

Sign in

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

ficlop-demo

Cloned from ficlopburner2/ficlop-demo

an instance of the ficlop demo

Updated a minute ago.

Open

1 CPU

1GB RAM

5GB HDD

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

Usage:

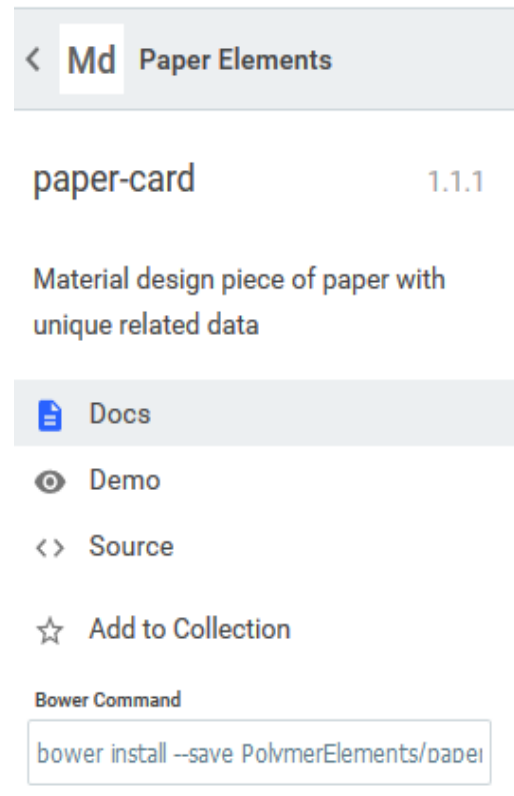
```
bower <command> [<args>] [<options>]
```

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>. Although much of what is on this page is useful for working with Polymer components, of immediate interest 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" and "paper-input" elements. 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-image/      paper-input/  
iron-a11y-keys-behavior/  iron-input/      paper-material/  
iron-autogrow-textarea/  iron-meta/       paper-ripple/  
iron-behaviors/        iron-validatable-behavior/  paper-styles/  
iron-checked-element-behavior/  paper-behaviors/  polymer/  
iron-flex-layout/      paper-button/     webcomponentsjs/  
iron-form-element-behavior/  paper-card/
```

6. By default, anything under the "workspace" directory is under Git source control. This will turn out to be a problem later in the build process. 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
```

The full demo app makes use of all the “ficlop” files in the www directory; we’ll be going over some of the code in them now.

- “ficlop_main.html” is the main file for the ficlop demo, our app with polymer components. Shown below is the block importing the polymer components we need; these are done with a <link> elements, as shown.

```
<!-- import all the Polymer components we're using -->
<script src="../../bower_components/webcomponentsjs/webcomponents-lite.js"></script>
<link rel="import" href="../../bower_components/iron-input/iron-input.html">
<link rel="import" href="../../bower_components/paper-button/paper-button.html">
<link rel="import" href="../../bower_components/paper-card/paper-card.html">
<link rel="import" href="../../bower_components/paper-input/paper-input.html">
```

The ficlop demo is entirely driven from Javascript, so its' body kicks off a loader script via onload attribute, and little else.

```
<body unresolved onload="loadLogin()">
  ERROR: Script Kickoff Failure!
</body>
```

- “ficlop_scripts.js” is the workhorse of the demo, giving functionality to all the buttons and swapping content in and out via DOM modification. Shown below is the “loadPage” function, which allows HTML snippets from other files to be pulled into the DOM.

```
//http://stackoverflow.com/questions/18930361/how-to-load-another-html-file-u
function loadPage(href)
{
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.open("GET", href, false); //JET some debuggers claim calling this
  xmlhttp.send();
  return xmlhttp.responseText;
}
```

We’ll be referencing code from ficlop_scripts.js later in this section, so keep it open.

- “ficlop_login.html”, “ficlop_card_browser.html”, and “ficlop_hidden_data.html” are all HTML snippets designed to be used with the loadPage function.

7. Now let's run the app. As we did with “firebase_access.html”, hit the “Run” button on the top menu bar while viewing firebase_access.html in the file viewer, then click the link that the apache server provides you in the open panel. The app should appear as shown.

Ficlop Polymer Demo

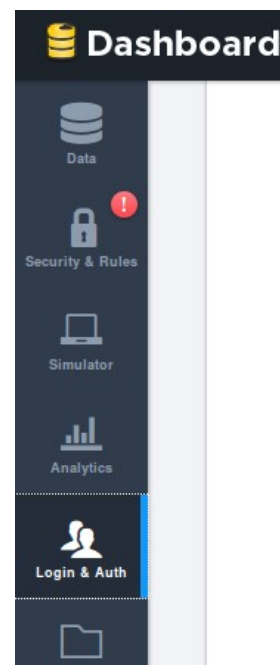
user email

password

LOG IN

8. The demo features some simple user authentication, so we'll need to create an account authorized to log into the demo. For that, visit your Firebase account and go to the “Login & Auth” tab, as shown.

If you need a reminder on how to get to your Firebase account, refer to the **Populating a Firebase Database** section.



9. At the bottom of the “Login & Auth” page is an empty table of “Registered Users”, as shown below. Hit the “Add User” button on the right.

Registered Users [refresh list](#)

Add User

Email ▾

Created ▾

User UID

Search for user...

no registered email/password users

10. A dialog will open for you to fill in an e-mail address and password, as shown.

By default there is no authentication of these e-mail addresses, so fill in whatever you want.

11. The Registered Users table will now contain an entry for the user you just made, as well as assigning a User UID for them.

Note where to find this User UID, because you will need it later.

12. Using the e-mail and password you just created, you can now log into the app. Fill in the fields you made for your user and click “LOG IN”.

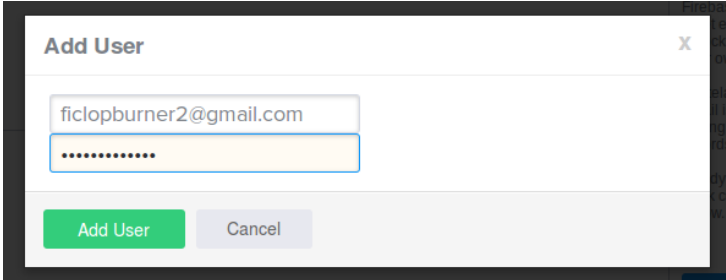
For the purposes of this tutorial, the password field is made readable, but it is a trivial change to make it display as the more familiar dots.

13. If you logged in successfully, the app should appear as shown.

The app's purpose is to display data about a (fictional) company in a “baseball card” format. You can view each company page by pressing the scrolling buttons at the bottom of the page. There's also an “Update” button on each page that lets us write data to the database.

Some of the data shown, like the company name and logo, would be public knowledge for a real company. Other data, like the company's bank balance, really shouldn't be public information. Also, while a company's employee count may be public knowledge, it probably shouldn't be editable by the general public.

The next section will show you how to secure your data on Firebase.



A dialog box titled "Add User" with a close button (X) in the top right corner. It contains two input fields: the first is for an email address, with "ficlopburner2@gmail.com" entered; the second is for a password, shown as a series of dots. Below the input fields are two buttons: a green "Add User" button and a grey "Cancel" button.

Email ▾	Created ▾	User UID
ficlopburner2@gmail.com	Mar 07, 2016	82d3384b-732d-431f-bbcb-6cf816bfa71d

Ficlop Polymer Demo

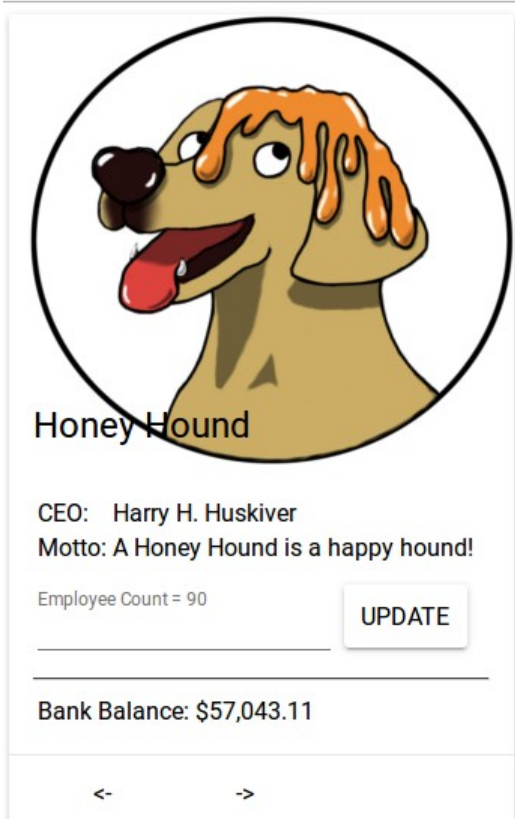
user email

ficlopburner2@gmail.com

password

2BurnerFiclop

LOG IN



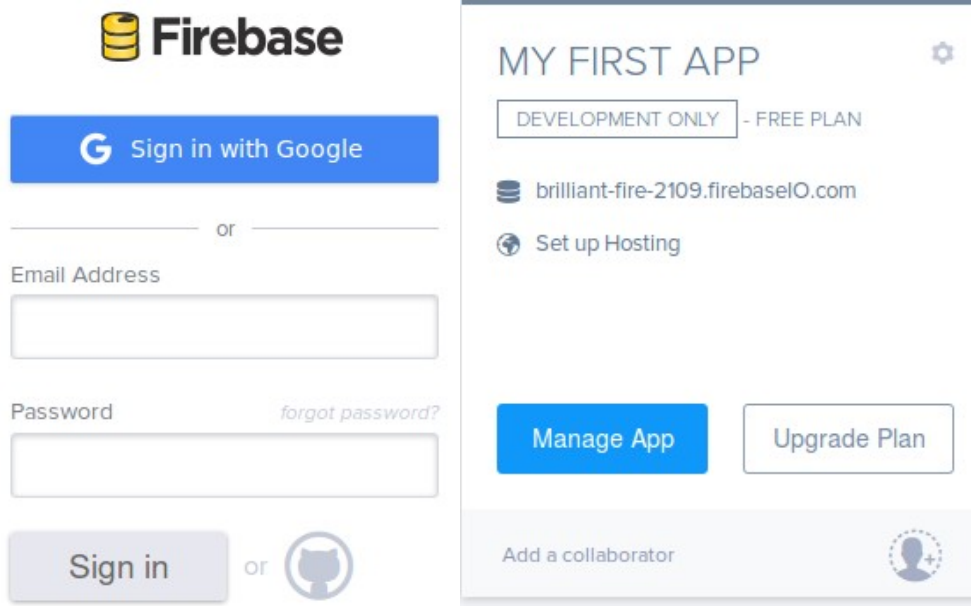
A screenshot of a mobile app interface showing a company card for "Honey Hound". The card features a circular logo with a cartoon illustration of a golden retriever's head with orange honey dripping over its eyes and ears. Below the logo, the company name "Honey Hound" is displayed. Further down, the CEO is listed as "Harry H. Huskiver" and the motto is "A Honey Hound is a happy hound!". There is a text input field for "Employee Count" with the value "90" and an "UPDATE" button to its right. At the bottom, the "Bank Balance" is shown as "\$57,043.11". Navigation arrows "<- >-" are at the very bottom.

Firestore: Adding user authorization

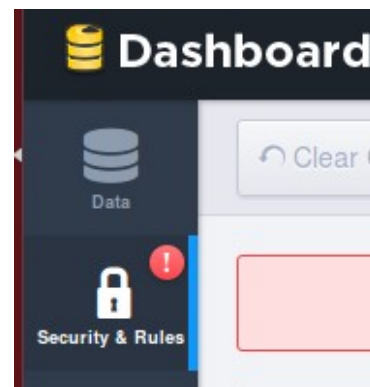
This section builds on the app we started in the “**Building an app with Polymer components**” section. If you have not completed that section, please do so before starting on this section.

Firestore can enforce user authorization for read/write access on anything it stores, at an arbitrary granularity. It does this through a “Security Rules” file associated with your Firestore account. This section covers writing/modifying this file.

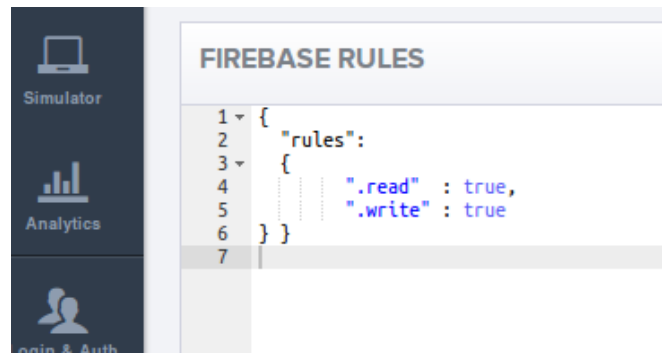
1. Log into your firestore account (<https://www.firebaseio.com/login/>) and click on your project's “Manage App” button.



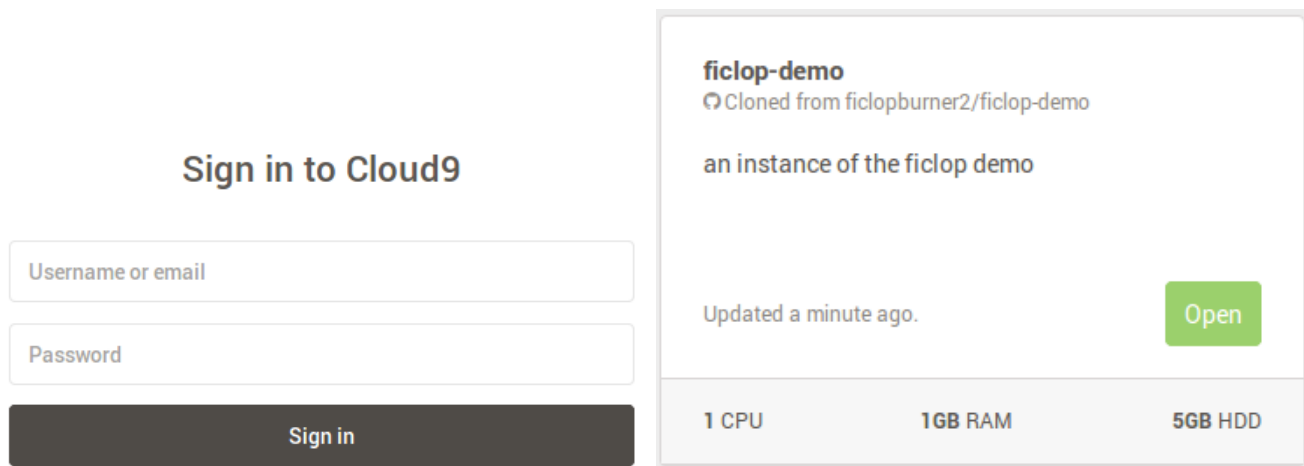
2. The security rules file is located, appropriately, under the “Security & Rules” tab of your dashboard, as shown.



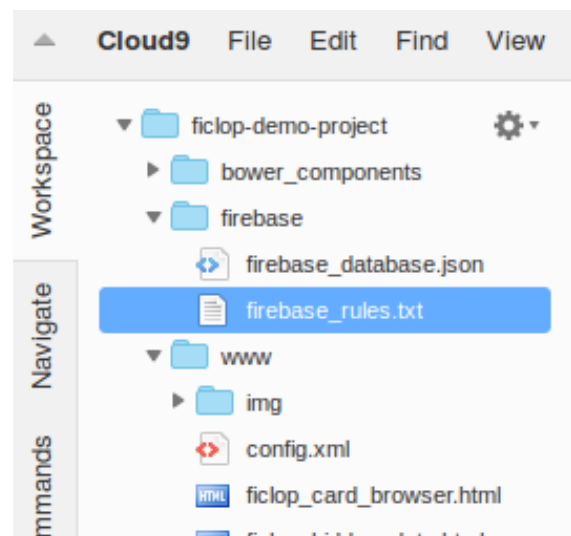
3. Opening this tab shows the default security rules file, as shown. As written, file always returns true when an app reads or writes; this means that anyone can read or write anything they want from the database.



4. As with the firebase data itself, this demo includes a firebase security rules file to get you started. Log into your cloud9 account and open your ficlop project to get it



5. Locate "firebase_rules.txt" in the workspace file tree and double click it.



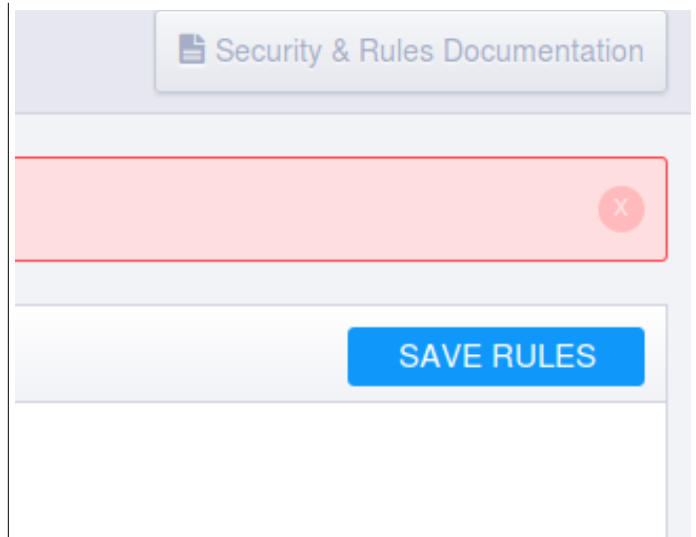
6. Copy/Paste the entire contents of “firebase_rules.txt” into the file editor for the “Security & Rules” tab. After doing so, the file should appear as shown below in the Security & Rules editor.

FIREBASE RULES

```
1 {  
2   "rules":  
3   {  
4     "CompaniesPrivate" :  
5     {  
6       "$CompanyPrivateIndex" :  
7       {  
8         ".read": "root.child('CompaniesAuthorizedUIDs').child($CompanyPrivateIndex).ch  
9         ".write": "root.child('CompaniesAuthorizedUIDs').child($CompanyPrivateIndex).chi  
10      } },  
11    "CompaniesPublic" :  
12    {  
13      ".read" : true,  
14      "$CompanyPublicIndex" :  
15      {  
16        ".write": "root.child('CompaniesAuthorizedUIDs').child($CompanyPublicIndex).chi  
17      } },  
18    "CompaniesAuthorizedUIDs" :  
19    {  
20      ".read" : true,  
21      ".write" : true  
22    } }  
23  } }  
24 }  
25 }
```

7. To commit the changes to the Security Rules file, hit the “Save Rules” button, as shown. If the rules file is syntactically correct, you’ll get a green “Rules Saved” message.

With this security rules file in place, only authorized users will be able to access a company's sensitive information. However, as of yet we have not defined what an “authorized user” is. The next steps will describe how to add authorized users.



The security rules file that Ficlop uses allows approved users (only) read access to data considered sensitive (like “Bank Balance”), but permits free read access to data considered public (like company name and logo). Approved users may write anywhere in the database they like. Steps 6-9 explain how `firebase_rules.txt` makes this approved/unapproved user distinction. Actual functional steps resume at step 10.

For a tutorial on Firebase Security Rules, go to <https://www.firebase.com/docs/security/quickstart.html>

8. At their most basic, security rules files define a set of sub-nodes (which must overlap with actual sub-nodes in the database to have any effect), and then define boolean statements which determines whether or not a read or write will be allowed.

Shown here is the default Firebase security file. It defines a set of subnodes (the “rules” sub-node stands in for the root node), and two (trivially simple) boolean statements; one for read, and one for write. This security file permits any user to read or write any node in the database.

FIREBASE RULES

```
1 {  
2   "rules":  
3   {  
4     ".read" : true,  
5     ".write" : true  
6   } }  
7
```

9. For the boolean statements of the rules file to be useful, they have to be able to determine things about the user's read/write operation, and about the state of the database.

Shown here is one of the security rules used in firebase_rules.txt.

```
"CompaniesPrivate" :  
{  
  "$CompanyPrivateIndex" :  
  {  
    ".read": "root.child('CompaniesAuthorizedUIDs').child($CompanyPrivateIndex).child(auth.uid).exists()  
    ".write": "root.child('CompaniesAuthorizedUIDs').child($CompanyPrivateIndex).child(auth.uid).exists()  
  } },  
  ...  
}
```

- “root” refers to the root node in our database.
- “.child” accesses a sub-node of a node previously referenced, by name.
“root.child('CompaniesAuthorizedUIDs')” refers to the CompaniesAuthorizedUIDs sub-node in the database
- Dollar (\$) signs indicate variables that hold a piece of a node's path in the database.
- “auth” is a structure that holds information about the user's security credentials. “auth.uid” refers to a unique number that each registered user is assigned when they are added to Firebase.
- “.exists()” returns true if the previously referenced node is present in the database, false otherwise.

A vague description of the effect these rules will have is, if there exists in “CompaniesAuthorizedUIDs” a sub-node whose name matches the UID of the user trying to read/write, that user will be granted read/write permissions to certain fields under the “CompaniesPrivate” database node.

10. When used in a rule, a \$ variable holds a piece of a node's path; which piece depends on where the \$ variable is referenced outside of the rule.

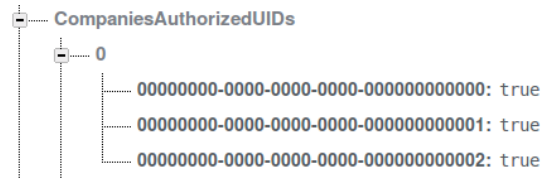
Shown below on the left is the database, and on the right is our security rules file. When “\$CompaniesPrivateIndex” is parsed in a security rule, because it is referenced under “CompaniesPrivate”, it's range of possible values are 0, 1, or 2; these are the three possible paths directly under “CompaniesPrivate”.

Which value is used when the security rule is assessed depends on what node the user is trying to read/write. When a user is trying to read/write “CompaniesPrivate/0” or one of it's sub-nodes, \$CompaniesPrivateIndex is equivalent 0, and similarly they are 1 or 2 if they access “CompaniesPrivate/1” or “CompaniesPrivate/2”, respectively.



11. Revisiting our boolean statement from step 7, the effect can be better stated as “if the user is trying to access 'CompaniesPrivate/0', and there exists a node under 'CompaniesAuthorizedUIDs/0' that matches the user's UID, the user is granted access; otherwise they are not.”, and similar rules for “CompaniesPrivate/0” and “CompaniesPrivate/1.”

Shown here is the database entry for “CompaniesAuthorizedUIDs/0”; the 32-digit numbers are UIDs, although they don't code to any actual registered users.

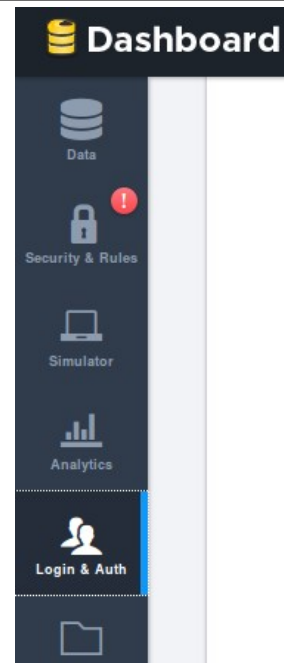


Now we're ready to add authenticated users to the database.

12. A user's UID is listed in the “Registered Users” table in the “Login & Auth” tab of your Dashboard, shown here. In the “**Building an app with Polymer components**” section, we already added a user to log into the app, as shown below.

Email ▾	Created ▾	User UID
ficlopburner2@gmail.com	Mar 07, 2016	82d3384b-732d-431f-bbcb-6cf816bfa71d

If you don't have a user, follow the steps in “**Building an app with Polymer components**” and create one.



13. Under the “Login & Auth” tab, copy the user's UID to your clipboard. Then, under the data tab, mouse over the “CompaniesAuthorizedUIDs/0” element; you should get a green + and a red -.

Click on the green + to add a new node to the database. A dialog will appear, as shown.



14. Paste the users UID to the “Name” field of the new node dialog. For Value, fill in “true”, without quotes (the value field is not actually used, but the key must have a value). Then hit Add.

After adding the node “CompaniesAuthorizedUIDs/0” should appear as shown.

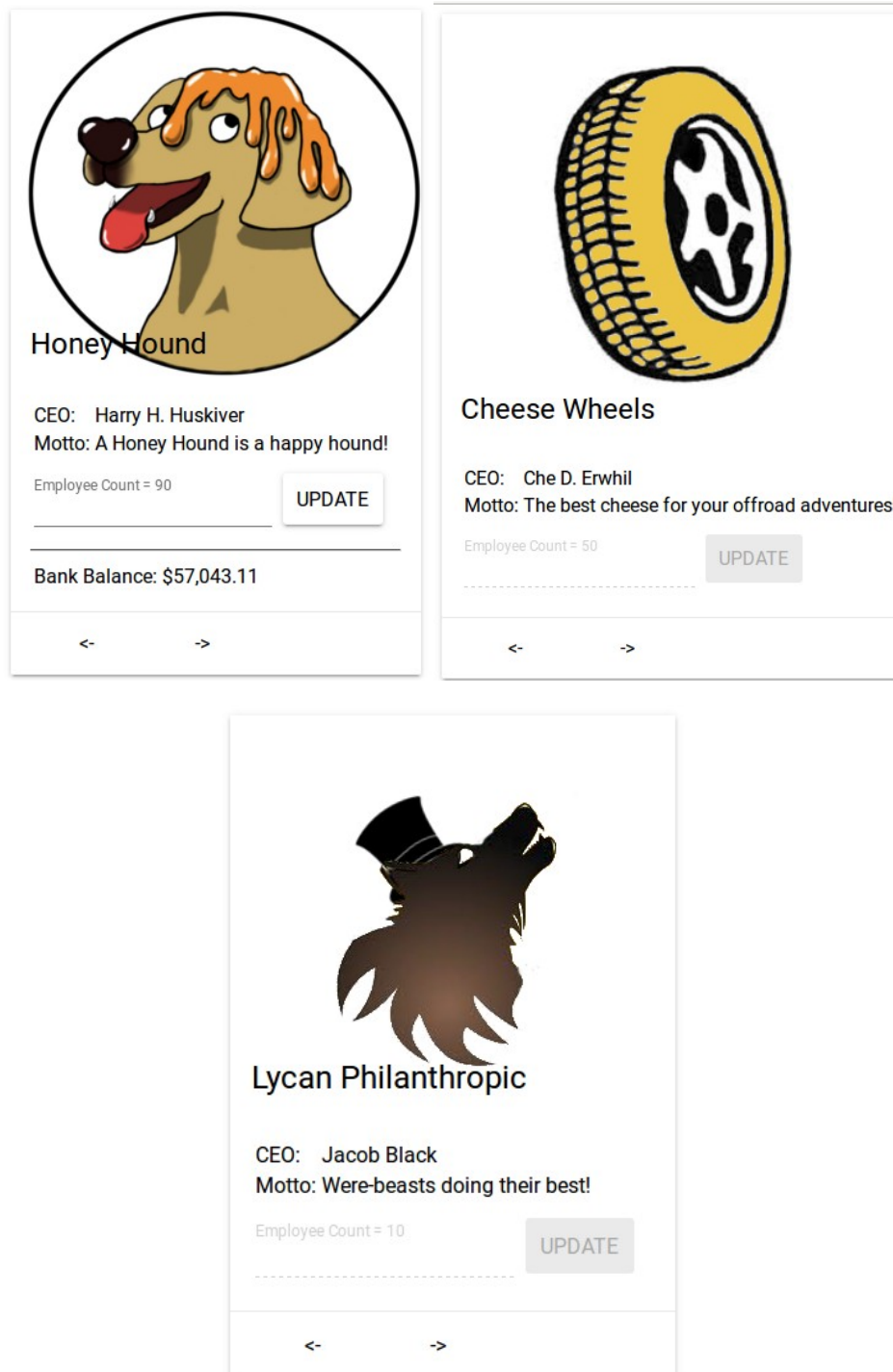


For the purposes of this demo, adding users by hand is perfectly reasonable. A more full-featured app would automate this process by means of a “App registration” process, but that goes beyond the scope of this demo.

15. With this user's UID added to the list of approved UIDs, when we log into this app as this user, we'll see the sensitive information for the approved company (Honey Hounds), but not for unapproved companies (Cheese Wheels or Lycan Philanthropic).

To test this, open your cloud9 project and run “ficlop_main.html”. Refer to the steps in the “**Accessing Firebase data from our app**” section if you need a refresher

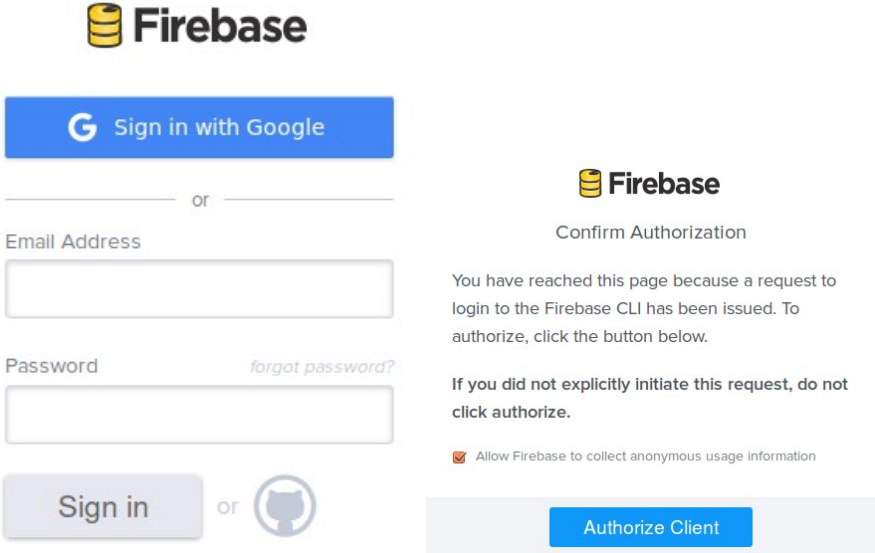
The three company cards should appear as shown;



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 via web browser.

1. Visit https://c9.io/login , and log in if necessary.	<div><h3>Sign in to Cloud9</h3><div><input type="text" value="Username or email"/></div><div><input type="password" value="Password"/></div><div>Sign in</div></div>
2. Open the Ficlop demo project we've been working with.	<div><div><div>ficlop-demo Cloned from ficlopburner2/ficlop-demo an instance of the ficlop demo Updated a minute ago. 1 CPU 1GB RAM 5GB HDD</div><div>Open</div></div></div>
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: <code>npm install -g firebase-tools</code>	
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: <code>firebase login</code>	

<p>5. This command will prompt you to open an authentication page in a separate tab, as shown.</p> <p>Click on the link and select “Open”.</p>	<pre> ficlopburner2:~/workspace/www (master) \$ firebase login /visit this URL on any device to log in: https://www.firebase.com/login/confirm.html?ticket=3b819efe-6a2f-42f2-80be-7b59b017dd2f Waiting for authentication... </pre>
<p>6. Provide your Firebase account credentials if necessary. If you are already logged in, this isn't necessary.</p> <p>After providing your credentials, Firebase will ask you to authorize the request we made with <code>firebase login</code>. Click “Authorize Client.”</p>	
<p>7. After authorizing the Cloud9 machine, return to Cloud9. It should appear as shown.</p>	<pre> ficlopburner2:~/workspace/www (master) \$ firebase login Visit this URL on any device to log in: https://www.firebase.com/login/confirm.html?ticket=3306a0e4-5adf-4bc6-9876-22c47c9aed66 Waiting for authentication... ✓ Success! Logged in as ficlopburner2@gmail.com ficlopburner2:~/workspace/www (master) \$ █ </pre>

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. The last step is to deploy the app, but there is a minor workaround that we must do for the deploy to work. We have instructed Firebase to upload the contents of the “www” directory, but the polymer elements directory (“bower_components”) is not available from that directory. “bower_components” must stay where it is to ensure that Phonegap does not pull in unvulcanized polymer components and break the build.

To work around this problem, we'll create a temporary symbolic link for Firebase deploy's benefit, and delete it afterwards. To do this, issue the following commands;

```
cd /home/ubuntu/workspace/www  
ln -s ../bower_components bower_compoonents  
firebase deploy  
rm ../bower_components
```

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

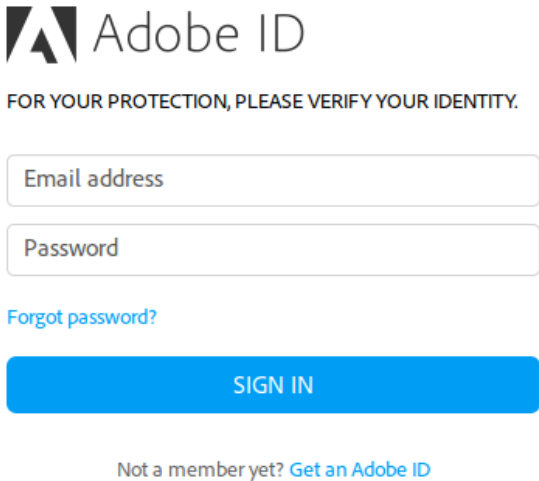
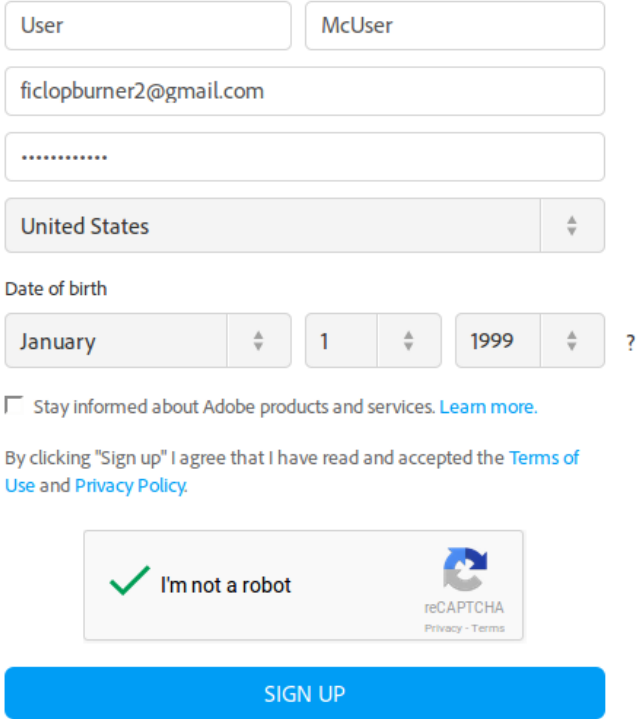
✓ Deploy complete!

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

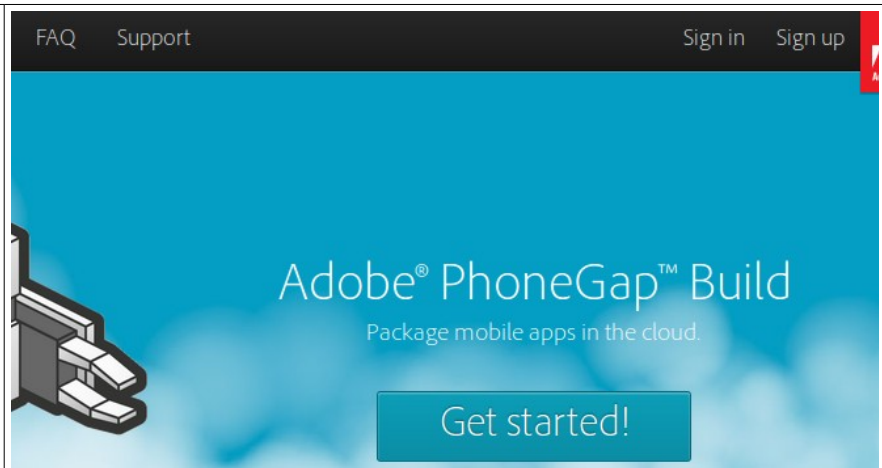
Visit the URL above or run firebase open
ficlopburner2:~/workspace/www (master) $
```

Phonegap-Build: New Account Creation

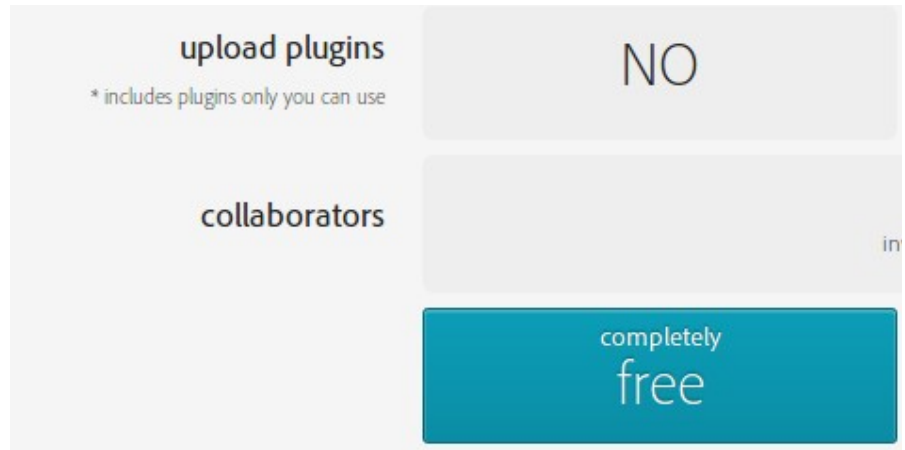
Phonegap-build is an Adobe service, and we need an Adobe account before we can use it.

<p>1. Visit the Adobe Account creation page at https://accounts.adobe.com/ and click “Get an Adobe ID”.</p>	
<p>2. Fill out the form and click “Sign Up”.</p>	
<p>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”.</p>	

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 for Android

This section assumes you have built and configured an app using Polymer components. If you have not, refer to the “Building an app with Polymer components” section.

Phonegap-Build and Polymer do not play well together right out of the box; Phonegap-Build requires a single “index.html” entry point to build properly, and Polymer elements do not respect that requirement. There is a workaround, but we must follow these 3 rules.

- 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.

We describe the steps for vulcanizing our app next.

<p>1. Visit https://c9.io/login, and log in if necessary.</p>	<div><h3>Sign in to Cloud9</h3><div><input type="text" value="Username or email"/></div><div><input type="password" value="Password"/></div><div>Sign in</div></div>
<p>2. The project you created earlier should be visible on your cloud9 home page. Click on “Open” to open the Cloud9 workspace.</p>	<div><div><div>ficlop-demo Cloned from ficlopburner2/ficlop-demo an instance of the ficlop demo Updated a minute ago.<div>Open</div></div><div>1 CPU1GB RAM5GB HDD</div></div></div>

3. Like bower, vulcanize is available using npm. Issue the following command in cloud9's bash window.

```
npm install vulcanize
```

It should appear as shown.

```
ficlopburner3:~/workspace (master) $ npm install vulcanize
vulcanize@1.14.7 node_modules/vulcanize
├─ path-posix@1.0.0
├─ es6-promise@2.3.0
├─ nopt@3.0.6 (abbrev@1.0.7)
├─ dom5@1.3.1 (clone@1.0.2, parse5@1.5.1)
├─ hydrolysis@1.22.0 (path-is-absolute@1.0.0, estraverse@3.1.0, espree@2.2.5, doctrine
└─ update-notifier@0.6.1 (is-npm@1.0.0, semver-diff@2.1.0, chalk@1.1.1, boxen@0.3.1, c
ficlopburner3:~/workspace (master) $ ls
LICENSE NOTICE README.md firebase/ labbook.pdf node_modules/ www/
ficlopburner3:~/workspace (master) $ vulcanize
vulcanize: Reduce an HTML file and its dependent HTML Imports into one file
```

4. The main entry point of our app is in “ficlop_main.html”, and we want to create “index.html”. Issue the following command to create a vulcanized index.html.

```
vulcanize ficlop_main.html > index.html
```

If successful, there will be no output.

5. When we have finished configuring Phonegap-Build, it will be able to pull our source tree directly from our Github account. This means that we must upload the vulcanized “index.html” to Github. Issue the following commands to push the new index.html to Github.

```
git add index.html
```

```
git commit -m "vulcanized index.html"
```

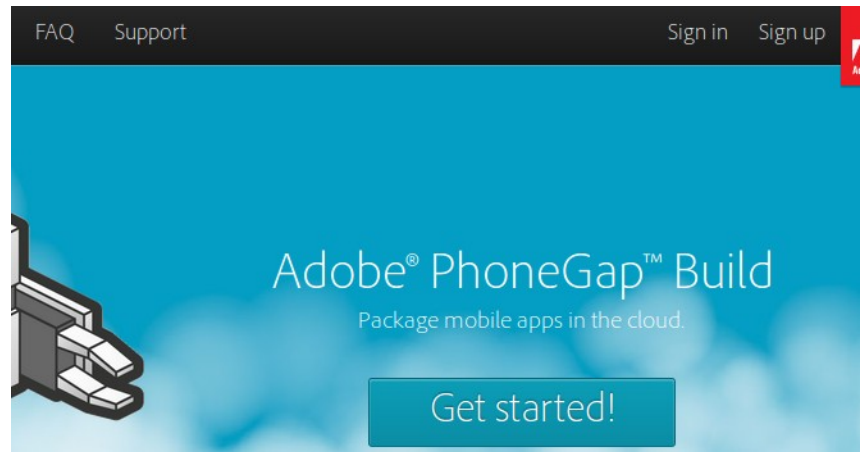
```
git push
```

Git push will require you to give your github credentials. If the Github commands are successful, it should appear as shown below. Refer to the Git commands cheat, or online tutorials, for more information.

```
ficlopburner2:~/workspace/www (master) $ git add index.html
ficlopburner2:~/workspace/www (master) $ git commit -m "uploading vulcanized ficlop_main.html"
[master b07951d] uploading vulcanized ficlop_main.html
2 files changed, 3 insertions(+), 9 deletions(-)
ficlopburner2:~/workspace/www (master) $ git push
Username for 'https://github.com': ficlopburner2@gmail.com
Password for 'https://ficlopburner2@gmail.com@github.com':
Counting objects: 14, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 830 bytes | 0 bytes/s, done.
Total 9 (delta 7), reused 0 (delta 0)
To https://github.com/ficlopburner2/ficlop-demo
cbf3ef2..b07951d master -> master
ficlopburner2:~/workspace/www (master) $
```

A vulcanized index.html is now available to Phonegap-Build. Now we can build our app for Android.

6. Visit <https://build.phonegap.com/> and click “Sign in”.



7. When the page loads, click on “Sign in with Adobe ID”

Sign in with Adobe ID



8. When the page loads, fill in your Adobe account credentials.

Adobe ID

SIGN IN TO CONTINUE

Bd PhoneGap Build

ficlopburner2@gmail.com

Password

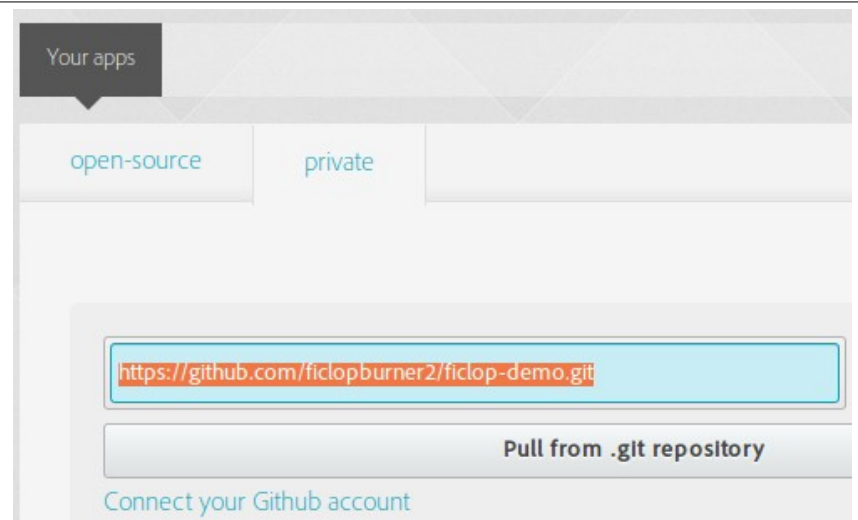
☒ Stay signed in
Uncheck on public devices.

[Forgot password?](#)

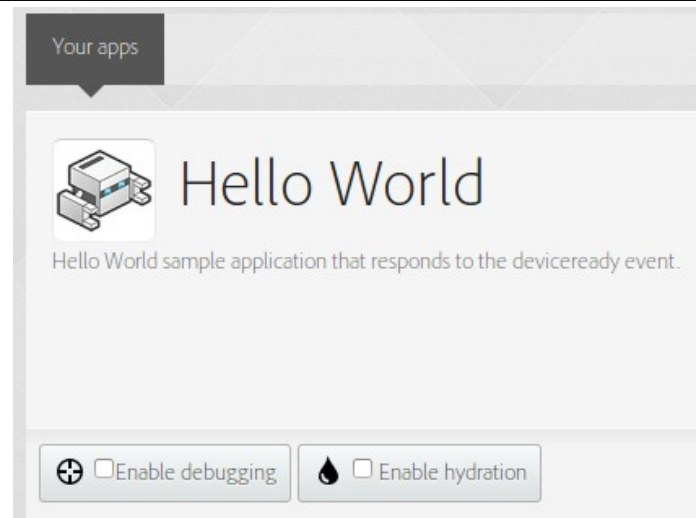
SIGN IN

9. 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.

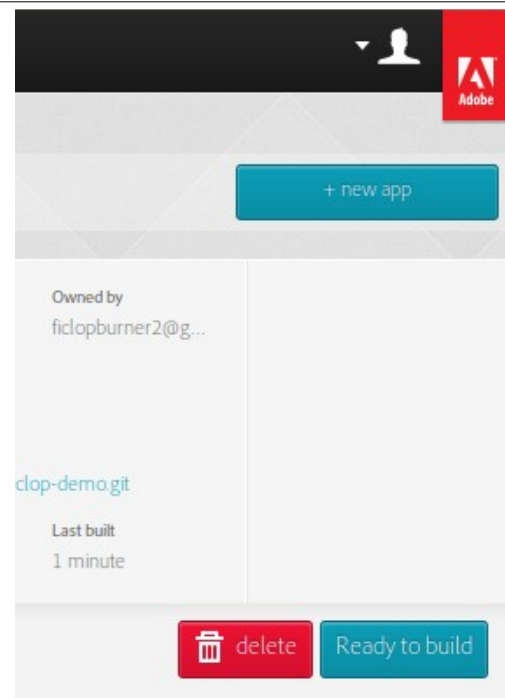


10. If the download was successful, you'll see a new “Hello World” application in your apps page, as shown.



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).

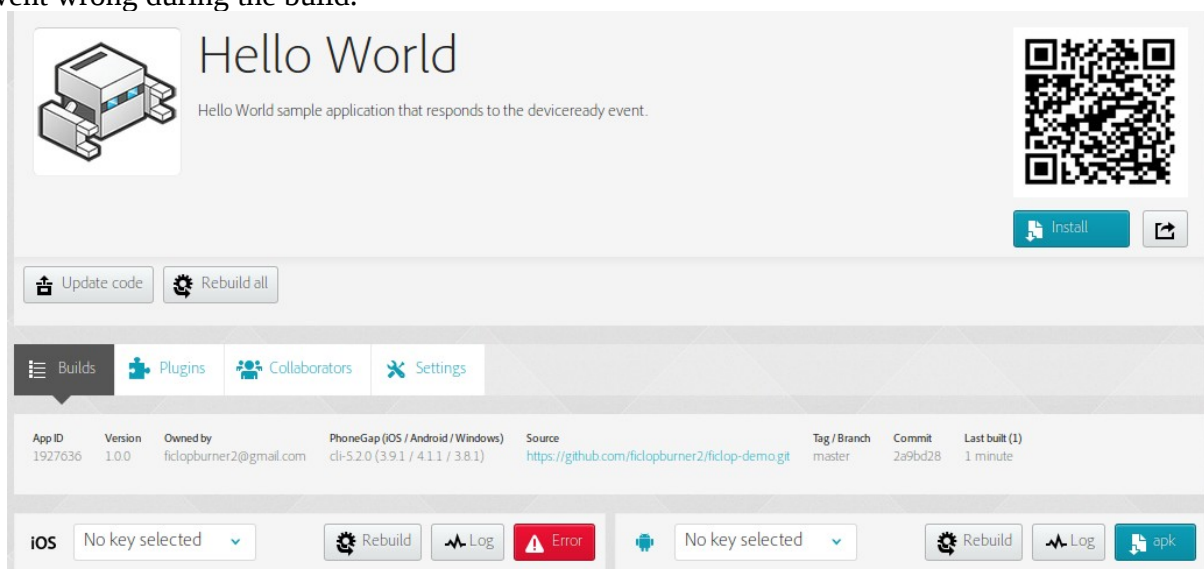
- Once your app has been built at least once, you can get more detailed controls by clicking on the app name.



- 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.



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

Debugging Tips

Cloud9

- If you're using Firefox to view your app, get the Firebug plugin to view console messages and such.
- Remember that your browser will cache the last version of your webpage, even if you're hosting it in the debugger

Firebase Security Rules

- Your primary tool for debugging is the “Simulator” tab, available under your Firebase dashboard.

Phonegap-Build

- The most common problem that is specific to Phonegap-build is a featureless blank screen on the phone app. This is caused by multiple files named “index.html” in the Github account sourced by phonegap build. Double check that you have not violated the rules discussed at the beginning of the “**Phonegap-Build: Building our App for Android**” section.