Using this project with IFTTT

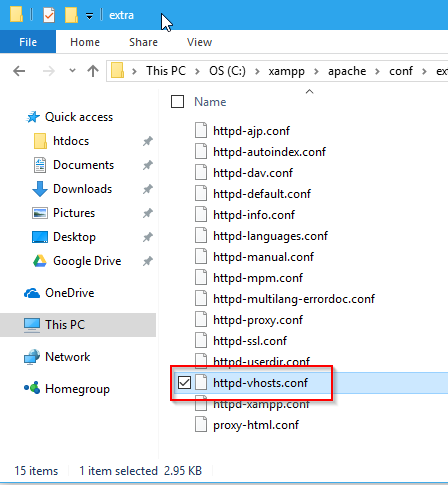
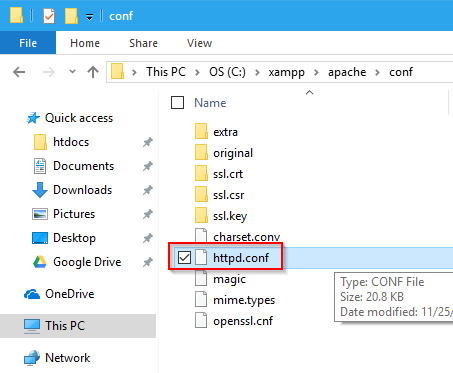
Connecting your TCP Lights to IFTTT with this project is a cake-walk. It will require these few things:

1. This project setup and controlling your lights (I’m assuming you’ll use Apache)
2. A static IP and or a IP service like http://freeddns.noip.com
3. A trigger device - Google Home / Alexa (I’m using a Google Home)
4. A IFTTT account

# Configuring Apache

Assuming you have this project already up and running on Apache or with XAMPP or something similar – most of the work is complete. By default apache runs on the standard port 80. I suggest setting it up to run on a non-standard port when integrating with IFTTT – but this is up to you. To run on a non standard port you’ll need to open / edit two files:

1. Apache/conf/httpd.conf
2. Apache/conf/extra/httpd-vhosts.conf



Within the httpd.conf file near line 58 you’ll see:

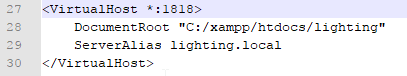
Listen 80 – this indicated to Apache that it should listen for requests on port 80. What we are going to do is get it to listen on another port. To do so, just add a new line and type “Listen {portNumber}” For instance: “Listen 1818”



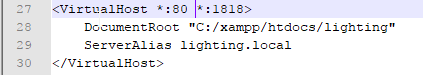
Save this.

Next open up the httpd-vhosts.conf file within the conf/extra folder.

What you’ll want to do is add a Virtual host entry to this file connecting you’re the TCP Web App to port we just setup to listen on. That can be done like so:



You can also make Apache listen on more than one port for a virtual host, IE configure Port 80 (standard) and the one you setup.



<VirtualHost \*:80 \*:1818>

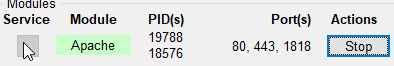
DocumentRoot "C:/xampp/htdocs/lighting"

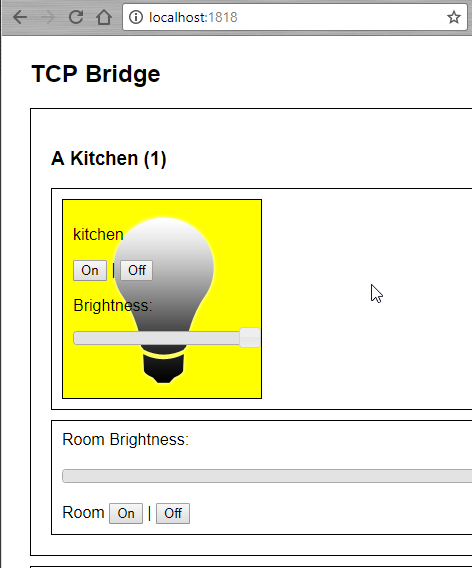
ServerAlias lighting.local

</VirtualHost>

In the example above, the Document Root is the path to where you have the project files. The Server Alias is what you refer to the project as in your local browser. This should be in your host file pointing back to 127.0.0.1

Save these changes. If you’re using XAMPP, start and stop Apache and you should see that it is running on the ports your specified:

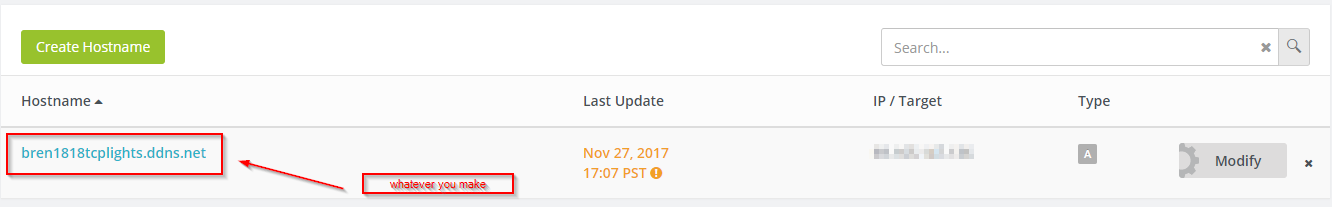


Now, go to your favorite browser and ensure that the application loads up. Go to localhost:{port} if you are using port 80 you can omit the port number.  
 

## Configuring DDNS

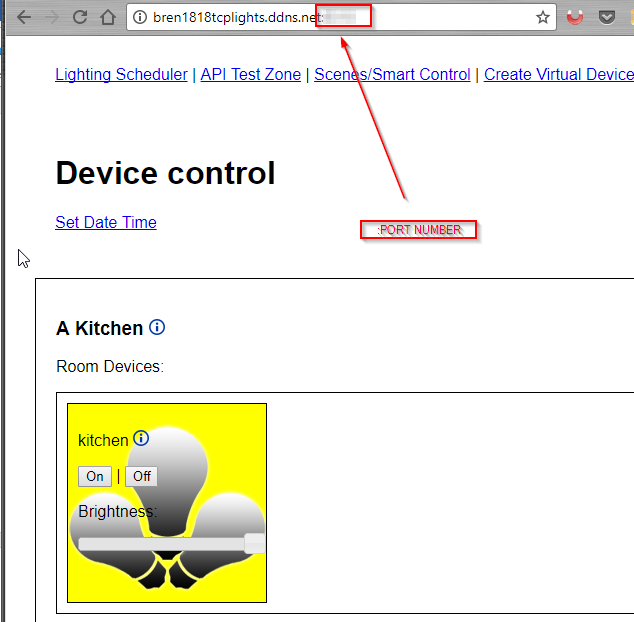
If you don’t have a static IP, or even if you do, using a service like: <http://freeddns.noip.com> is great because you can tie your hard-to-remember potentially always changing IP address into a static easy-to-remember one. Sign up for a free account and make something you can remember IE:

{yourname}tcplights.ddns.net



I suggest you also install a client to auto-update your IP if it changes.

Make sure the address works by going to it in your browser. \*\* Remember to include the :PORTNUMBER if you’re using something other than port 80



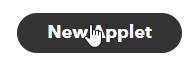
If this is working, onto to the next step.

## Configuring IFTTT

What?! We’re already at the IFTTT setup? No way it can’t be this easy – can it?

Well yes it can be, this is the last major step, and its perhaps the most difficult, but it is also easy to do.

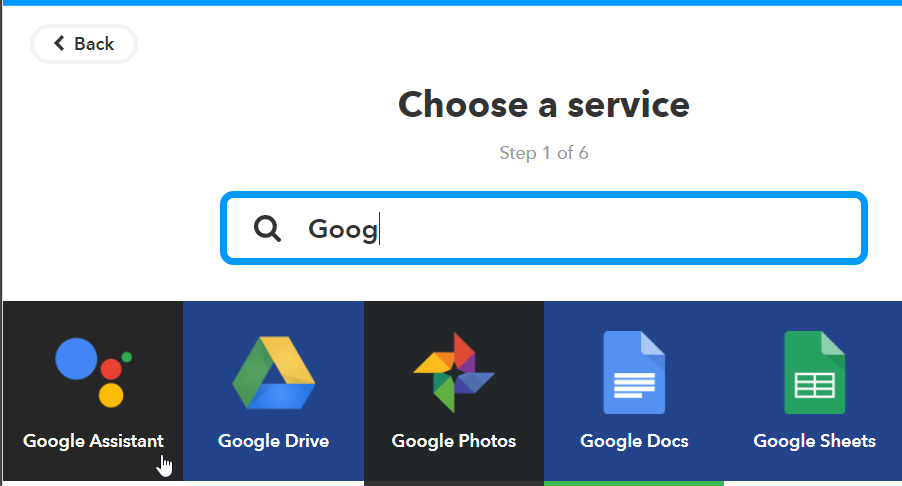
Open up and create an account with IFTTT if you don’t already have one. Click on “My Applets”:

 and then  “New Applet”.

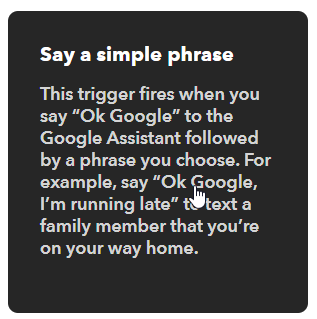
Click on the “+this”



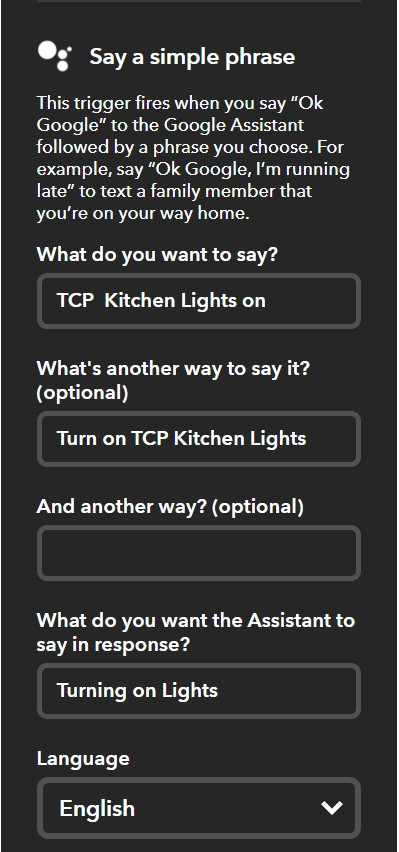
This is where you’ll want to select your Trigger – IE a Google Home or something else. I will write up using a Google Home. Select “Google Assistant”.



Choose: “Say a Simple Phrase”

 (\*\*Note as you get comfortable with the next couple steps, you can use other options or try experimenting)

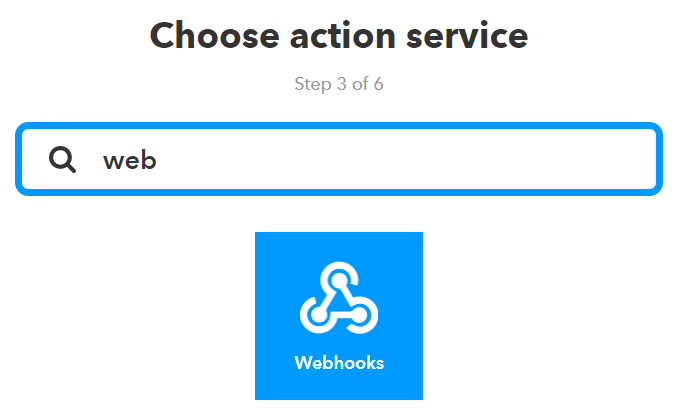
Configure the Simple Phrase as you see fit. For example, an action I’d like is to be able to say “*Hey Google Turn on the TCP Kitchen Lights*”.



Once you’ve configured the trigger, the next step is to configure the “that”.

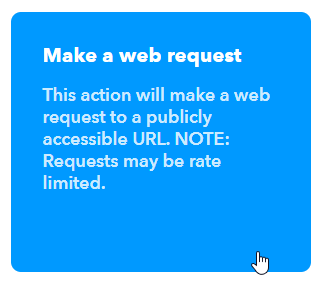


Click on the “+that”. We want to use the “Web Hooks” to talk back to our server

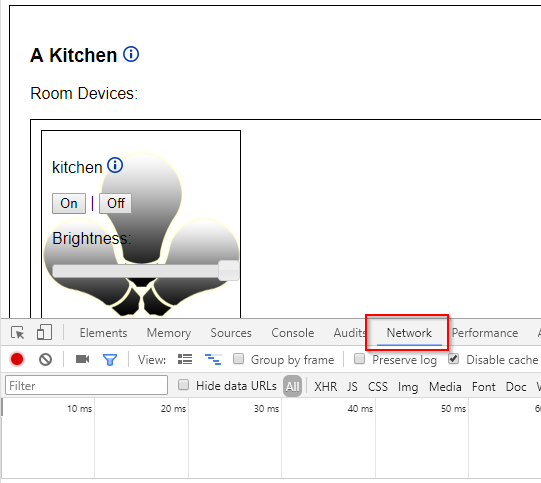


This is probably the most difficult part of the process, but it is still very simple.

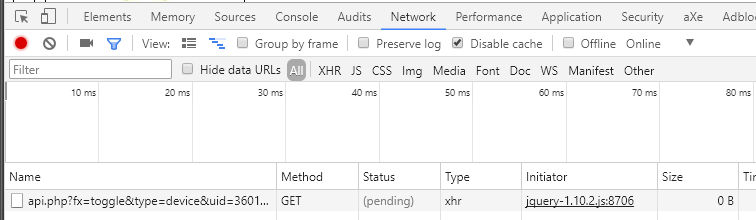
Webhooks currently only has one option (“Make a web request”). Select it.



All we really need to figure out is the URL. Going back to the browser where your web-application open with the name & port you’ve setup press F12 to open up the developer tools. (F12 works pretty much universally across browsers). I use chrome as my browser of choice, so hopefully you can follow along. Open up the “Network sub tab”

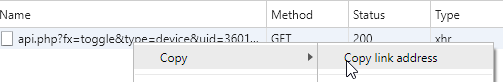


And then perform the action you want to happen. For me, since I want the Kitchen lights to turn on, I’ll click the “On” button.



You should see a line of text appear in the network tab. This is the API call back that I’ve setup for the project.

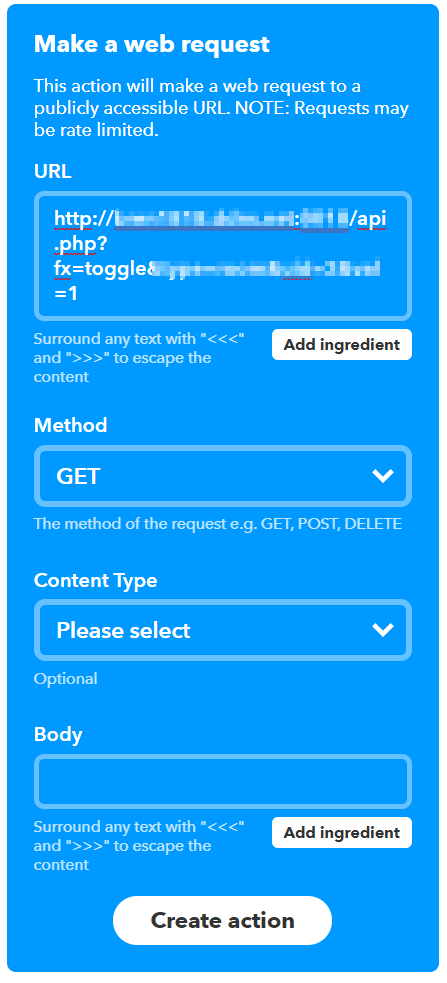
Right click and choose copy > copy link address



This should give you a string like:

http://{yourname}tcplights.ddns.net:{portNumber}/api.php?fx=toggle&type=device&uid=360uniquedevice5508&val=1

Paste this into the “URL” Box in the webhook section of IFTTT. Leave the Method as GET and click “Create action”.



If you just created an “On” Action, I suggest you create another corresponding “Off” action to match.

## Final Step

Test away! Try it out, talk to your Device and see if turning your lights works with voice commands. Explore the API actions I have setup via the developer tools and come up with your own ideas!

Be sure to keep security in mind when opening up your house to the “Internet of things”.

Enjoy your no-longer-dumb lights.