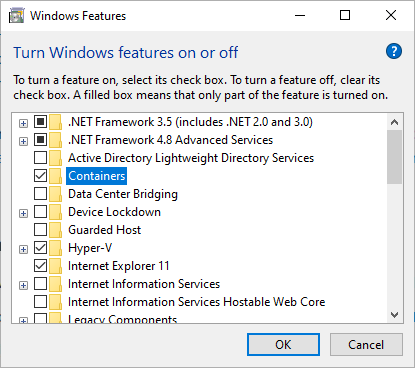
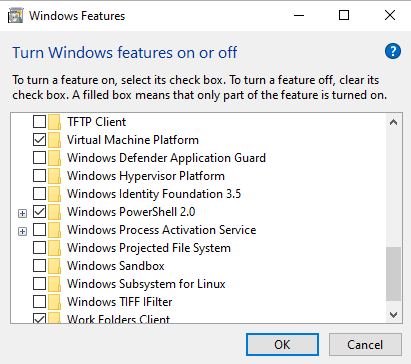
****

**Build instructions for running OMH on FHIR on a Windows 10 desktop/laptop computer**

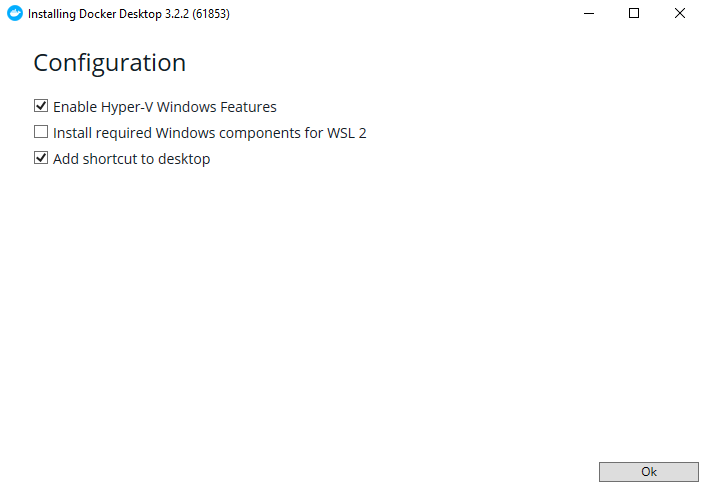
1. Login as administrator
2. Ensure that these features **ARE** enabled on the computer!!! Can do that in the control panel 🡪 Programs 🡪 Turn Windows features on and off. If you turn it on, reboot.

* Containers
* Hyper-V
* Virtual Machine Platform
* **DO NOT TURN ON**: Windows Subsystem for Linux (because it’s **WSL** and we’re not going to use it because it makes Docker flakey)

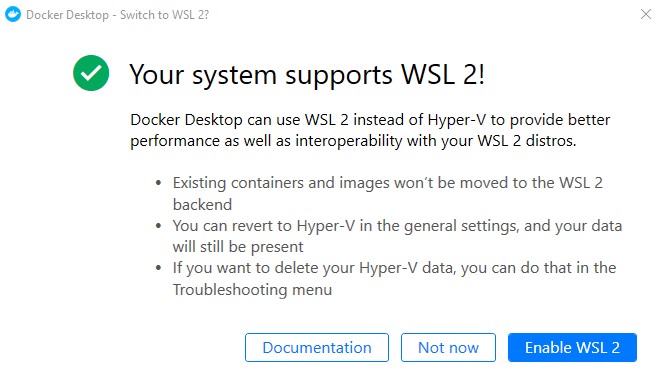




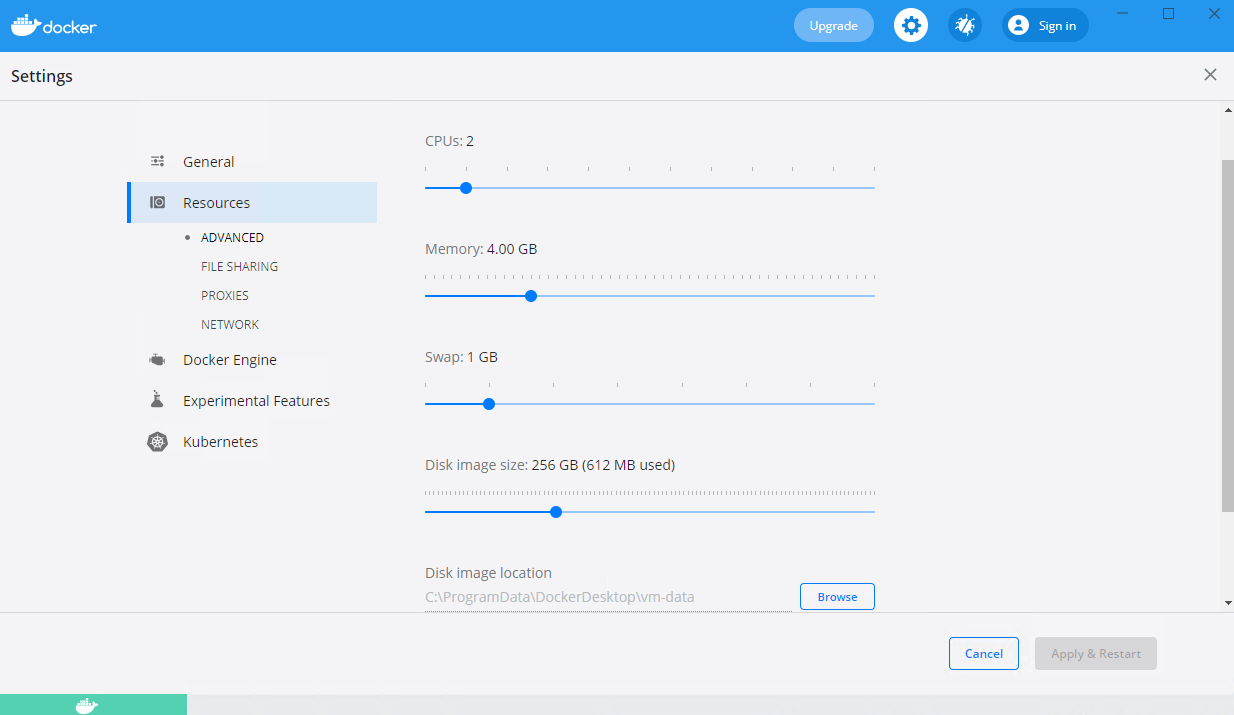
1. Copy package to C:/Users/<<USER>>/Documents/OMH-on-FHIR
2. Download the Stable version of Docker Desktop for Windows <https://docs.docker.com/docker-for-windows/install/>
3. In the Download folder, right-click on the file and Run as Administrator and do the full install.
4. When you see the configuration screen, **don’t do WSL because it’s buggy**!:



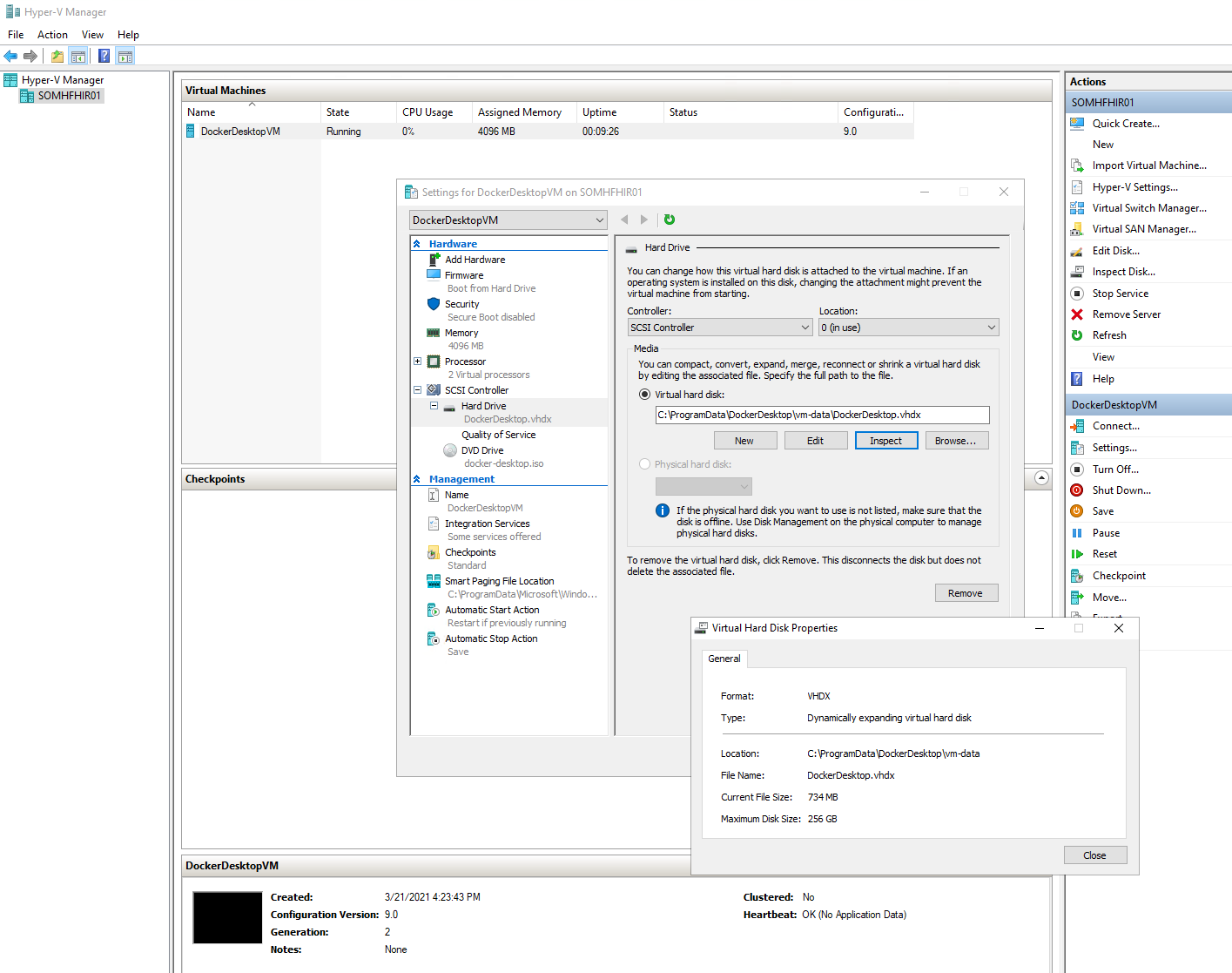
1. You will get a message and click **NOT NOW**!!!!!:



1. Reboot again. Wait about 5 minutes to allow docker to start up. It will usually give messages when it is starting and when it has finished starting up. And there will be a little whale in the System Tray. If there isn’t, go to Task Manager and stop all of the Docker processes EXCEPT FOR DOCKER SERVICE!!!. Then double-click on the Docker Desktop icon.
2. Increase the Memory and Disk Image Size under Settings (gear) 🡪 Resources. Then click Apply & Restart



1. You can confirm that this worked in Windows Powershell by typing: **virtmgmt.msc**  Click on the DockerDesktopVM and on the right click on Settings… and you can click on the HardDrive and then Inspect to see that it worked:

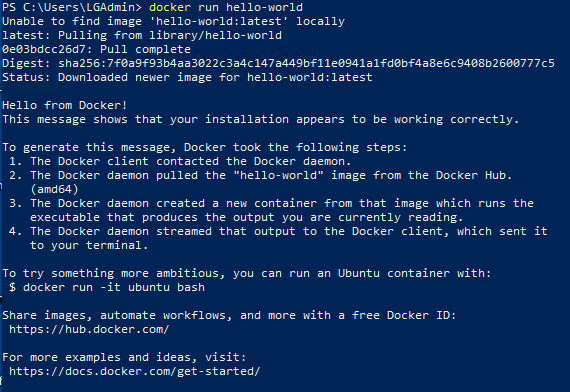


1. NOTE: I can’t seem to create/use a daemon.json file to specify “local” logging which is important to limit the size of the log files. So instead, in the .YML docker-compose file I have added:

logging:

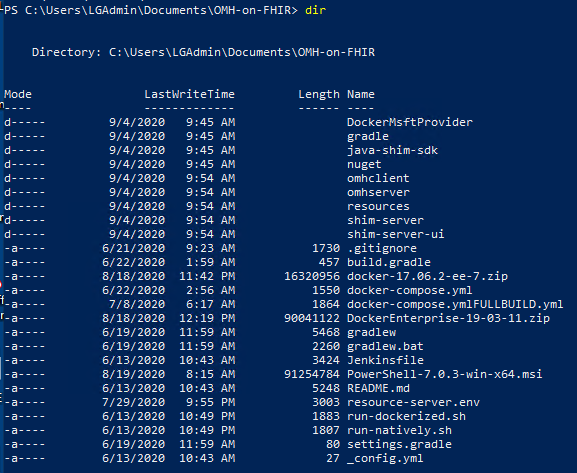
driver: "json-file"

1. Start Windows **Powershell** as an Administrator. To confirm that Docker is loaded and running properly, type: docker run hello-world



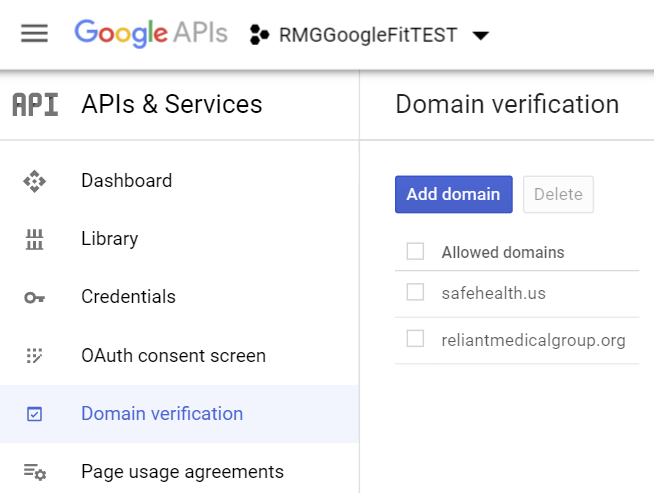
If docker wasn’t recognized, run Docker Desktop.

1. Now type: cd C:/Users/<<USER>>/Documents/OMH-on-FHIR (or whatever you named the folder). You can type dir to confirm that you have been successful so far:



Do the following to run the application:

1. Make sure that you own a domain that you have the login to manage. It must have an email account. For instance, I created the [LGarber@SafeHealth.us](mailto:LGarber@SafeHealth.us)
2. Set up a Google G-Suite account so we can access Google Fit. <https://gsuite.google.com/landing/partners/referral/trial.html> Here’s the account we’re using: [RMGGFITOMHonFHIR@gmail.com](mailto:RMGGFITOMHonFHIR@gmail.com) and SAFEHealth.us ([LGarber@SafeHealth.us](mailto:LGarber@SafeHealth.us) ).  **Note that you must own/control the domain name for the G-suite account, the App’s home page, App Privacy Policy, App Terms of Service, and the App’s redirect URLs! Google has you add TXT or CNAME records to those domains to verify them before giving Production Public OAuth Consent Screen verification approval!**
3. Get OAuth Credentials by following these instructions: <https://developers.google.com/fit/rest/v1/get-started>
4. Go to the [Google API Console](https://console.developers.google.com/flows/enableapi?apiid=fitness).
5. Make sure you are logged in as the domain owner (e.g. we used [LGarber@SafeHealth.us](mailto:LGarber@SafeHealth.us)). Create a Project and name it RMGGoogleFitDEV, RMGGoogleFitTEST or RMGGoogleFitPROD for example (with that same domain as the organization and location), and click **Create**
6. Click **Domain verification** from the menu on the left, and then click **Add domain** button. You should enter the domains that carry the App’s Home Page, Privacy Policy, Terms of Service, and Callback redirects. Follow the instructions to configure verification that you control those domains (using TXT or CNAME). So you will end up, for example:



1. Click **Credentials** from the menu on the left, and the **+Create Credentials** from the top, then select **OAuth Client ID**.
2. Under **Application type**, select **Web application**.
3. Give it a name (e.g. GoogleFitOMHFHIRTEST)
4. Under **Authorized JavaScript origins**, enter the base URL of the websitesite from which browser requests will originate (for example https://developers.google.com is the URL used by the OAuth Playground). **THIS DOESN’T SEEM TO NEED TO BE POPULATED**
5. Under **Authorized redirect URI**, enter the URL of the site where responses will be handled (for example https://developers.google.com/oauthplayground is the URL used by the OAuth Playground).

For our DEV environment (on my laptop):

<http://localhost>

<http://localhost:8989>

<http://localhost:8989/omhonfhir/activity>

<http://localhost:8989/omhonfhir/login>

<http://localhost:8989/omhonfhir/callback>

<http://localhost:8989/omhonfhir/launch>

<http://localhost:8084/authorize/googlefit/callback>

<http://localhost:8083/authorize/googlefit/callback>

<http://localhost:8888/mdata/authorize/googlefit/callback>

Or for our TEST environment:

[http://STOMHFHIR01.FHS.com](http://localhost)

[http://STOMHFHIR01.FHS.com:8989](http://localhost:8989)

[http://STOMHFHIR01.FHS.com:8989/omhonfhir/activity](http://localhost:8989/omhonfhir/activity)

[http://STOMHFHIR01.FHS.com:8989/omhonfhir/login](http://localhost:8989/omhonfhir/login)

[http://STOMHFHIR01.FHS.com:8989/omhonfhir/callback](http://localhost:8989/omhonfhir/callback)

[http://STOMHFHIR01.FHS.com:8989/omhonfhir/launch](http://localhost:8989/omhonfhir/launch)

[http://STOMHFHIR01.FHS.com:8084/authorize/googlefit/callback](http://localhost:8084/authorize/googlefit/callback)

[http://STOMHFHIR01.FHS.com:8083/authorize/googlefit/callback](http://localhost:8083/authorize/googlefit/callback)

[http://STOMHFHIR01.FHS.com:8888/mdata/authorize/googlefit/callback](http://localhost:8888/mdata/authorize/googlefit/callback)

Or for our PROD environment:

<http://SOMHFHIR01.FHS.com>

<http://SOMHFHIR01.FHS.com:8989>

<http://SOMHFHIR01.FHS.com:8989/omhonfhir/activity>

<http://SOMHFHIR01.FHS.com:8989/omhonfhir/login>

<http://SOMHFHIR01.FHS.com:8989/omhonfhir/callback>

<http://SOMHFHIR01.FHS.com:8989/omhonfhir/launch>

<http://SOMHFHIR01.FHS.com:8084/authorize/googlefit/callback>

<http://SOMHFHIR01.FHS.com:8083/authorize/googlefit/callback>

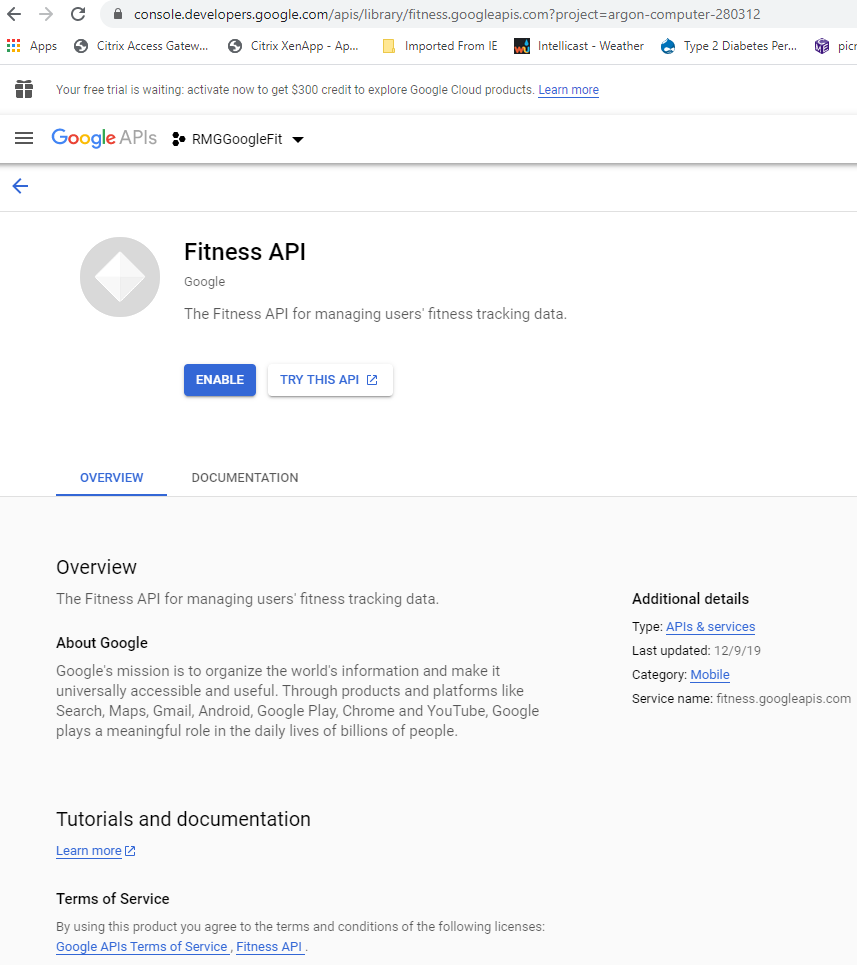
<http://SOMHFHIR01.FHS.com:8888/mdata/authorize/googlefit/callback>

1. Click **Create**. Your new OAuth 2.0 Client ID and secret appear in the list of IDs for your project. An OAuth 2.0 Client ID is a string of characters, something like this:

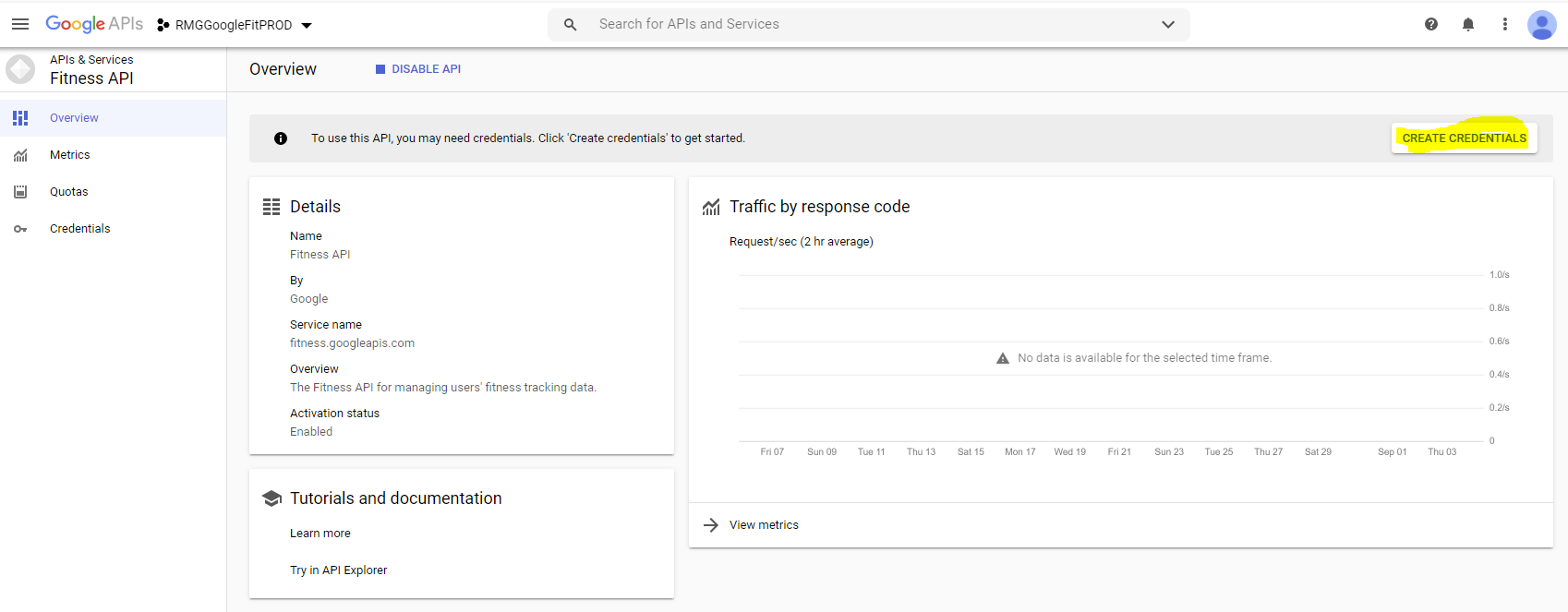
780816631155-gbvyo1o7r2pn95qc4ei9d61io4uh48hl.apps.googleusercontent.com

And a Client Secret looks like this: Ls0-3M1dgx-BTMybU\_1QTwNt

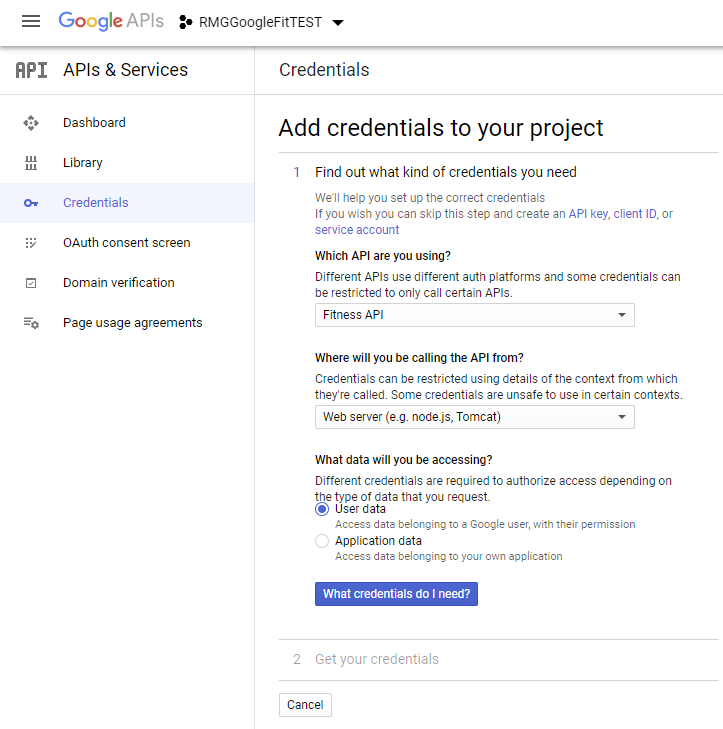
1. Click on API Library (left-hand menu) and search for the Fitness API. Click to **Enable** it.



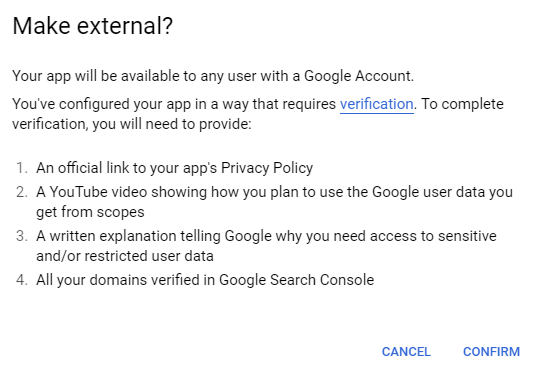
1. Then go back into the Library (click the GoogleAPIs logo and then Library) to find it again and then click “Manage”
2. Click the “Create Credentials” button:



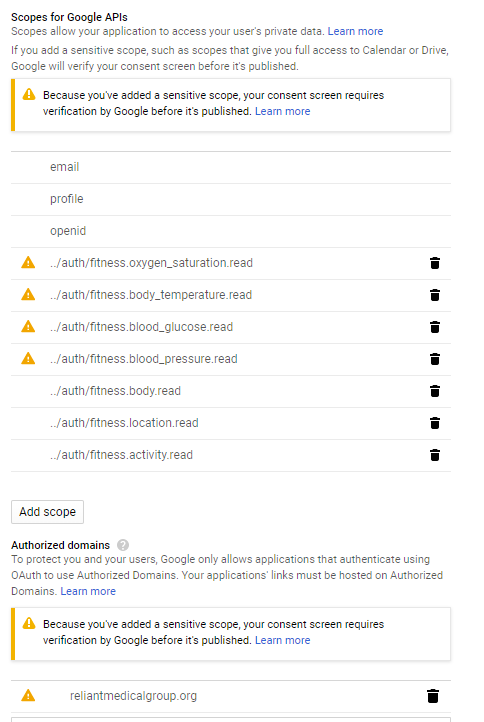
1. Select Fitness API and Web server from the dropdown menu and click “User data”:



1. Click on “What credentials do I need?” and “Set up Consent Screen” (or it might be the “Configure Consent Screen” button)
   1. Click Internal if you’re just testing with users who have the same email address as your domain.
   2. However, eventually you’ll need to make it External. Clicking on that gives you:



* 1. For external, the application type is Public. Give it an application name like Reliant Medical Group MyChart TEST
  2. Upload a logo
  3. Provide a support email (e.g. lgarber@safehealth.us)
  4. Now you have to add Scopes (i.e. the data elements that you want to download from GoogleFit). Click on the Add Scope button, select the ones you desire, and then click Add. Notice that the ones that had a lock next to them now trigger an alert that Google needs to verify everything before posting:



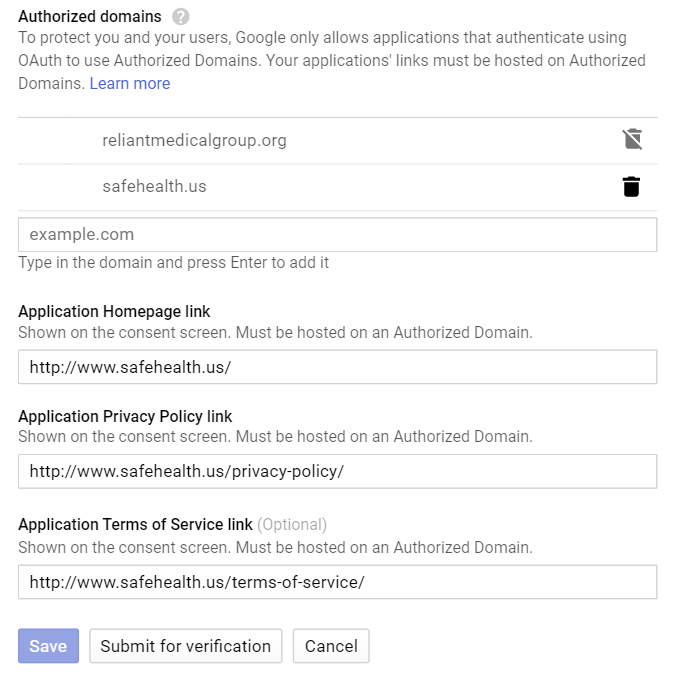
If for some reason the Fitness API scopes don’t appear, you can manually enter these (or others from <https://developers.google.com/identity/protocols/oauth2/scopes?hl=en_US#fitness>):

<https://www.googleapis.com/auth/fitness.body.read>

<https://www.googleapis.com/auth/fitness.location.read>

<https://www.googleapis.com/auth/fitness.activity.read>

* 1. So for now I unchecked those 4 scopes.
  2. You also need to provide a domain that will host the application as well as privacy and terms of service links:

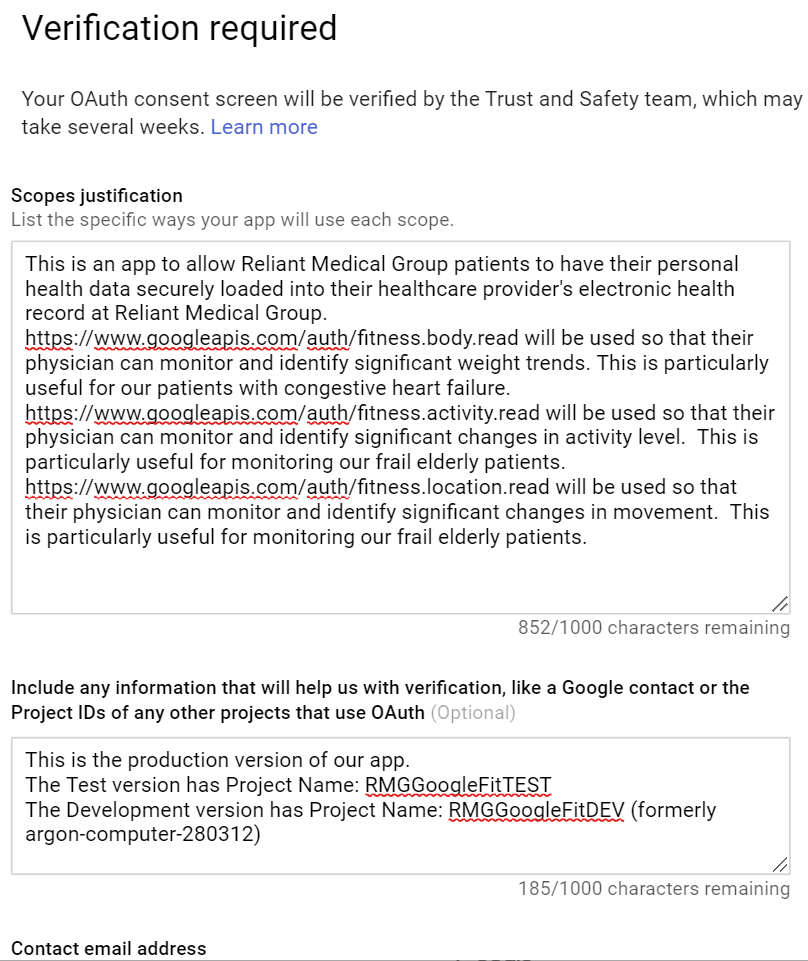


<http://reliantmedicalgroup.org>

<https://mychart.reliantmedicalgroup.org/MyChart/Authentication/Login?mode=stdfile&option=hlthprivacy>

<https://mychart.reliantmedicalgroup.org/MyChart/Authentication/Login?mode=stdfile&option=termsandconditions>

* 1. Click Submit for verification. A Verification required screen will pop up:



* 1. For the Scopes justification:

This is an app to allow Reliant Medical Group patients to have their personal health data securely loaded into their healthcare provider's electronic health record at Reliant Medical Group.

https://www.googleapis.com/auth/fitness.body.read will be used so that their physician can monitor and identify significant weight trends. This is particularly useful for our patients with congestive heart failure.

https://www.googleapis.com/auth/fitness.activity.read will be used so that their physician can monitor and identify significant changes in activity level. This is particularly useful for monitoring our frail elderly patients.

https://www.googleapis.com/auth/fitness.location.read will be used so that their physician can monitor and identify significant changes in movement. This is particularly useful for monitoring our frail elderly patients.

* 1. For the extra information:

This app is now ready for production after having successfully been tested in our DEV and Test environments.

I have created separate environments/projects and have physical servers for the Test and Production environments:

Project: RMGGoogleFitDEV

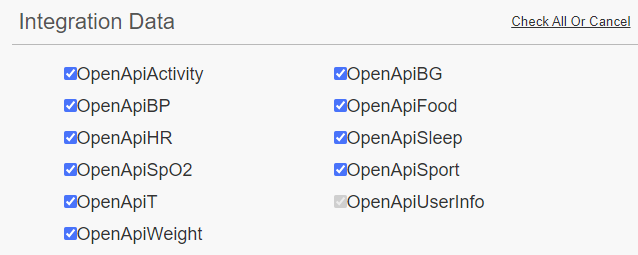
Project: RMGGoogleFitTEST

Project: RMGGoogleFitPROD

Please proceed with verification of RMGGoogleFitPROD for production. Thank you

Provide an email for contact address

1. Similarly, get credentials from iHealth (<https://developer.ihealthlabs.com/dev_documentation_OpenApiDoc.htm>)
   1. Create an account on the developer site
   2. Click on MyApplications and click REGISTER button
   3. Name (e.g. Reliant Medical Group MyChart TEST)
   4. App Website: (e.g. [www.ReliantMedicalGroup.org](http://www.ReliantMedicalGroup.org))
   5. Categories: Health & Fitness
   6. Description: Reliant Medical Group patients can use this to securely share your health measurements with your healthcare team. Be sure to talk to your healthcare provider first to enable this feature for you.
   7. Check all for Integration Data:



* 1. Redirect: <http://localhost:8888/mdata/authorize/ihealth/callback> or <http://STOMHFHIR01.FHS.com:8888/mdata/authorize/ihealth/callback>

Or <http://SOMHFHIR01.FHS.com:8888/mdata/authorize/ihealth/callback>

* 1. Auth Removal: <http://localhost:8084/deauthorize/ihealth> or <http://STOMHFHIR01.FHS.com:8084/deauthorize/ihealth> or <http://SOMHFHIR01.FHS.com:8084/deauthorize/ihealth>
  2. Select “Pull Model”
  3. Disable the Pub/Sub
  4. Press the FINISH button

1. Similarly, get credentials from Withings (<https://developer.withings.com/oauth2/>):
   1. Create a Withings account
   2. Register your application (<https://account.withings.com/partner/add_oauth2>)
   3. Name: Reliant Medical Group MyChart TEST
   4. Description: Reliant Medical Group patients can use this to securely share your health measurements with your healthcare team. Be sure to talk to your healthcare provider first to enable this feature for you.
   5. Callback URL: <http://STOMHFHIR01.FHS.com:8888/mdata/authorize/withings/callback>
   6. NOTE: You need to keep the Environment as DEV because when you switch to PROD you need https and the port has to be 80 or 443 which is not how GTRI built the OMH-on-FHIR client.
   7. Logo: e.g. RMG MyChart Icon.png
2. Using Notepad, update resource-server.env file to store in C:/Users/<<USER>>/Documents/OMH-on-FHIR with environment variables to configure the Shimmer server. Remove the #’s to uncomment lines we need to configure and enter correct values where there are placeholders.

| **Variable** | **Description** |
| --- | --- |
| OPENMHEALTH\_SHIMMER\_DATA\_PROVIDER\_REDIRECT\_BASE\_URL | Base URL for Shimmer to use for OAuth redirects |
| OPENMHEALTH\_SHIM\_FITBIT\_CLIENT\_ID | Client ID for Shimmer to use for FitBit Authentication |
| OPENMHEALTH\_SHIM\_FITBIT\_CLIENT\_SECRET | Client Secret for Shimmer to use for FitBit authentication |
| OPENMHEALTH\_SHIM\_GOOGLEFIT\_CLIENT\_ID | Client ID for Shimmer to use for Google Fit authentication |
| OPENMHEALTH\_SHIM\_GOOGLEFIT\_CLIENT\_SECRET | Client secret for Shimmer to use for Google Fit authentication |

Place the file in the root directory of the project (i.e. C:/Users/97/Documents/OMH-on-FHIR ).

Also need to update "C:\Users\97\Documents\OMH-on-FHIR\omhclient\app\js\env.js" which uses the FHIRAPIBase for the Shimmer **Authentication** redirect (which it does before authorization):

(function (window) {

    //following environment variable pattern described here: https://www.jvandemo.com/how-to-configure-your-angularjs-application-using-environment-variables/

    window.\_\_env = window.\_\_env || {};

    window.\_\_env.baseUrl = '/omhonfhir/';

    window.\_\_env.fitbitShim = 'fitbit';

    window.\_\_env.googleFitShim = 'googlefit';

    window.\_\_env.omhOnFhirClientId = '93651a15-4664-486e-8661-eca7ebc21bda';

    window.\_\_env.omhOnFhirScope = 'patient/\*.read launch'; //to force provider login use the following scopes 'openid profile'

    window.\_\_env.omhOnFhirRedirectUri = 'https://apps.hdap.gatech.edu/omhonfhir/login';

    window.\_\_env.omhOnFhirAPIBase = 'https://apps.hdap.gatech.edu/mdata';

    window.\_\_env.omhOnFhirAPIShimmerAuth = '/shimmerAuthentication';

}(this));

So it looks like this:

(function (window) {

//following environment variable pattern described here: https://www.jvandemo.com/how-to-configure-your-angularjs-application-using-environment-variables/

window.\_\_env = window.\_\_env || {};

window.\_\_env.baseUrl = '/omhonfhir/';

window.\_\_env.fitbitShim = 'fitbit';

window.\_\_env.googleFitShim = 'googlefit';

window.\_\_env.iHealthShim = 'ihealth';

window.\_\_env.withingsShim = 'withings';

window.\_\_env.omhOnFhirClientId = '93651a15-4664-486e-8661-eca7ebc21bda';

window.\_\_env.omhOnFhirScope = 'patient/\*.read launch'; //to force provider login use the following scopes 'openid profile'

window.\_\_env.omhOnFhirRedirectUri = 'http://STOMHFHIR01.FHS.com:8989/omhonfhir/activity'; //'http://rmgomhclient:8989/omhonfhir/login'; //'https://apps.hdap.gatech.edu/omhonfhir/login';

window.\_\_env.omhOnFhirAPIBase = 'http://STOMHFHIR01.FHS.com:8888/mdata'; //shimmerAuthentication Redirect URL Base //'https://apps.hdap.gatech.edu/mdata';

window.\_\_env.omhOnFhirAPIShimmerAuth = '/shimmerAuthentication';

window.\_\_env.CheckORDforOMHURL = 'http://sesscriptnp01.fhs.com/Interconnect-WebServiceHostPOC/api/Garber/2020/Informatics/Utility/CheckORDforOMH';

window.\_\_env.FileShimmerIDforOMHURL = 'http://sesscriptnp01.fhs.com/Interconnect-WebServiceHostPOC/api/Garber/2020/Informatics/Utility/FileShimmerIDforOMH';

}(this));

1. Create a postgres.env file with environment variables to configure the Postgres database.

| **Variable** | **Description** |
| --- | --- |
| POSTGRES\_DB | The name of the database to create/use = mdata |
| POSTGRES\_USER | The name of the database user = mdata |
| POSTGRES\_PASSWORD | The password for the database user = mdata11! |

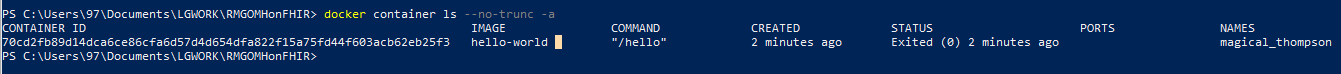
Store it in C:/Users/97/Documents/OMH-on-FHIR/omhserver/

1. Create a omh-server.env file with environment variables to configure the OMH on Fhir web service.

| **Variable** | **Description** |
| --- | --- |
| SHIMMER\_SERVER\_URL | The URL to the Shimmer resource server |
| SHIMMER\_REDIRECT\_URL | The redirect URL to pass to the Shimmer API. It contains the URL to the mdata-app /authorize/fitbit/callback endpoint that handles successful user authentication. Shimmer only redirects to this URL after successful authentication. |
| OMH\_ON\_FHIR\_CALLBACK | The URL to the OMH on FHIR UI application to use after successful Shimmer authentication. |
| OMH\_ON\_FHIR\_LOGIN | The URL to the user interface that handles login to Fitbit and Google fit. |

Store it in C:/Users/97/Documents/OMH-on-FHIR/omhserver/

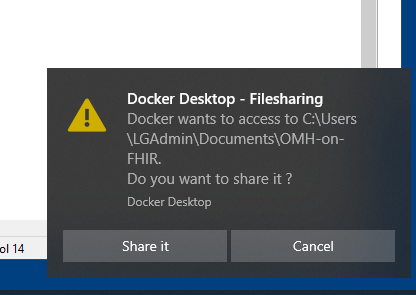
1. Show that there currently aren’t any docker containers running (other than the hello-world that exited already) by typing in powershell: **docker container ls --no-trunc -a**



1. Now type: cd C:/Users/97/Documents/LGWORK/RMGOMHonFHIR (or whatever you named the folder).
2. Now build the containers by typing in powershell:

**docker-compose -f docker-compose.yml up --force-recreate**

**If this pops up in the bottom-right corner of the screen, click “Share it”**



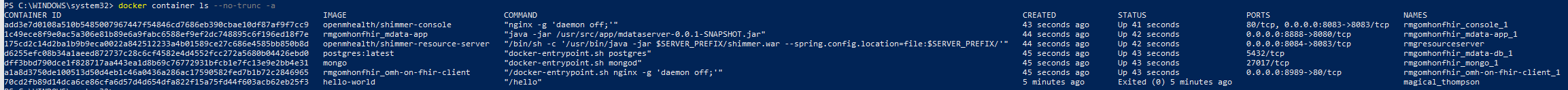
1. This takes a few minutes to build all of the docker containers. You don’t get a command prompt when it’s done because this window displays the log messages for all of the containers now, however the last messages when done building are:

rmgresourceserver | 2020-08-27 13:03:06.002 INFO 1 --- [ main] o.e.jetty.server.AbstractConnector : Started ServerConnector@7995092a{HTTP/1.1,[http/1.1]}{0.0.0.0:8083}

rmgresourceserver | 2020-08-27 13:03:06.005 INFO 1 --- [ main] .s.b.c.e.j.JettyEmbeddedServletContainer : Jetty started on port(s) 8083 (http/1.1)

rmgresourceserver | 2020-08-27 13:03:06.011 INFO 1 --- [ main] org.openmhealth.shimmer.Application : Started Application in 10.293 seconds (JVM running for 11.994)

1. Open a powershell as administrator. Now type: cd C:/Users/97/Documents/LGWORK/RMGOMHonFHIR (or whatever you named the folder).
2. Now to see the containers you built, type: **docker container ls --no-trunc -a**



1. Going forward (but not now), any edits you make need to be recompiled. **But remember that deleting the mongo and mdata (postgres) containers will remove all of the OAuth2 authorizations/Access Tokens/Refresh Tokens for every patient so you would need to sign them up again (unless you’ve done the backups as described further below).**
   1. Whenever you make an edit to env.js (or any javascript or html routines), you will also need to do the following (after stopping/deleting the old containers):
      1. CD to the omhclient folder (e.g. C:\Users\97\Documents\LGWORK\OMH-on-FHIR\omhclient) that you need to rebuild that container using (and do NOT forget that last period which tells it that the dockerfile is in this folder!!!):

PS C:\Users\97\Documents\LGWORK\OMH-on-FHIR\omhclient> **docker build --no-cache .**

* + 1. After that, Now build the containers by typing in powershell:

**docker-compose -f docker-compose.yml up --force-recreate**

* + 1. NOTE: SOMETIMES INTELLIJ DOESN’T RECOGNIZE CHANGED FILES!!! So you need to disable 'safe writes'. Please try disabling Use "safe write" in File | Settings | Appearance & Behavior | System Settings
    2. **BUT IntelliJ sometimes GETS WORSE!!!! If you still don’t get Javascript changes updating, Enable safe write, close IntelliJ. Reopen IntelliJ and then disable safewrite again!**
    3. **LASTLY, if it still doesn’t seem to work, CLEAR THE BROWSER CACHE!!!**
    4. **Also, if you ever get on the browser a “White Label” error, or in a logfile get this error:**

**initdb: error: could not create directory "/var/lib/postgresql/data/pg\_wal": No space left on device**

**It means you have filled up the allotted virtual disk space in Docker. This shouldn’t happen if you followed the instructions in the beginning! You can get a rough idea on how much storage can be reclaimed by typing:**

**docker system df**

**Or more details by typing:**

**docker system df --verbose**

**You need to run this command to get rid of old unused Docker images**

**docker image prune**

**If that didn’t clean up enough, MAKE SURE THAT ALL CONTAINERS ARE RUNNING OR RESTARTING (BECAUSE THE FOLLOWING COMMAND WILL DELETE STOPPED CONTAINERS AND EVERYTHING ASSOCIATED WITH THEM)!!! By typing**

**docker container ls --no-trunc -a**

**Then enter:**

**docker system prune -a**

**If that wasn’t enough, try:**

**docker system prune -a –volumes**

**If that still isn’t enough, stop and remove the containers that are easy to do (everything but NOT MONGO and NOT POSTGRES because those have the database that you’d need to recover (see backup discussion below)!!!!!!:**

**REPOSITORY TAG IMAGE ID CREATED SIZE SHARED SIZE UNIQUE SIZE CONTAINERS**

**rmgomhclient latest 173a68125124 4 hours ago 270.8MB 0B 270.8MB 1**

**rmgmdataserver latest 058d6243c032 4 hours ago 696.7MB 0B 696.7MB 1**

**rmgresourceserver latest 009929659c7d 2 months ago 114.1MB 0B 114.1MB 1**

**openmhealth/shimmer-console latest 4b0c63d8a6d2 3 years ago 16.39MB 0B 16.39MB 1**

**docker stop <<container ID>>**

**docker container rm <<container ID>>**

**After all 4 are stopped and removed, then rebuild them in this order:**

* + - * 1. **resource-server**
        2. **mdata-app15**
        3. **omh-on-fhir-client35**
        4. **console**

**Using (for example for the 3rd one):**

**PS C:\Users\LGAdmin\Documents\OMH-on-FHIR> docker-compose -f docker-compose.yml up -d --build omh-on-fhir-client35**

**If that doesn’t work you may need to rebuild all of the containers as above but this may kill the databases SO BACK THEM UP FIRST!!!!. But THIS WILL ALSO BE A GOOD TIME TO INCREASE DOCKER’S STORAGE. So we’ll do this in 3 steps:**

1. **Stop automated queries for data (For Reliant, that means we suspend the Epic SCHEDQ jobs)**
2. **Backup Mongo and Postgres databases**
3. **Stop containers**
4. **Review Docker’s virtual hard drive and expand it (using virtmgmt.msc if you can’t view the Docker Windows Dashboard – see the install instructions above)**
5. **Restart the containers**
6. **Restore Mongo and Postgres databases**
7. **Restart the Epic SCHEDQ jobs**

**First, Stop any automated queries for data**

**Here’s how to backup the databases (DO ALL OF THESE AT LEAST TO SAVE THE BACKUPS BECAUSE I WAS ONLY ABLE TO RESTORE USING ONE TECHNIQUE FOR EACH DATABASE!!!):**

**TO BACKUP POSTGRESQL MDATA DATABASE:**

**At an elevated powershell command prompt, enter these lines:**

cd "C:\Program Files\PostgreSQL\13\bin"

.\pg\_dump -U mdata mdata > mdata\_2021-03-21.sql

**When it asks for the password, enter:**

mdata11!

**OR EVEN BETTER we can Backup all tables and globals for Postgres in a human-readable format using (this works if there were no weird characters, although ignore the errors on the restore):**

cd "C:\Program Files\PostgreSQL\13\bin"

docker exec -t omh-on-fhir\_mdata-db\_1 pg\_dumpall -c -U mdata > mdata\_backup\_2021-03-21.sql

**and then you’ll be able to restore it later using:**

cd "C:\Program Files\PostgreSQL\13\bin"

cat mdata\_backup\_2021-03-21.sql | docker exec -i omh-on-fhir\_mdata-db\_1 psql -U mdata -d mdata

**OR (ALTHOUGH THIS DOESN’T SEEM TO WORK) we can Backup Postgres mdata in case there were unreadable characters (which makes a compressed format) using:**

cd "C:\Program Files\PostgreSQL\13\bin"

docker exec -ti omh-on-fhir\_mdata-db\_1 bash -c 'pg\_dump -Fc -U mdata > mdata\_cleanbackup\_2021-03-21.sql'

docker cp omh-on-fhir\_mdata-db\_1:/mdata\_cleanbackup\_2021-03-21.sql mdata\_cleanbackup\_2021-03-21.sql

**and then restore it using:**

cd "C:\Program Files\PostgreSQL\13\bin"

docker cp mdata\_backup\_2021-03-21.sql omh-on-fhir\_mdata-db\_1:/mdata\_cleanbackup\_2021-03-21.sql

docker exec -ti omh-on-fhir\_mdata-db\_1 pg\_restore -U mdata -c -d mdata mdata\_cleanbackup\_2021-03-21.sql

**To BACKUP MONGO DB (DO ALL OF THESE AT LEAST TO SAVE THE BACKUPS BECAUSE I WAS ONLY ABLE TO RESTORE USING ONE TECHNIQUE FOR EACH DATABASE!!!):**

**At an elevated powershell command prompt, enter these lines to create a dump folder with all of the tables:**

cd "C:\Program Files\MongoDB\Tools\100\bin"

.\mongodump

**And you can restore the database later using:**

cd "C:\Program Files\MongoDB\Tools\100\bin"

.\mongorestore --nsInclude='omh\_dsu.accessParameters' --nsFrom='omh\_dsu.accessParameters' --nsTo='omh\_dsu.accessParameters'

.\mongorestore --nsInclude='omh\_dsu.authorizationRequestParameters' --nsFrom='omh\_dsu.authorizationRequestParameters' --nsTo='omh\_dsu.authorizationRequestParameters'

**While this approach seemed better, IT DOES NOT WORK!!!!!!!!, at an elevated powershell command prompt, enter these lines to create a dump folder with all of the tables:**

cd "C:\Program Files\MongoDB\Tools\100\bin"

docker exec omh-on-fhir\_mongo\_1 sh -c 'mongodump --archive' > mongoDB\_backup\_2021-03-21.dump

**And you can restore using:**

cd "C:\Program Files\MongoDB\Tools\100\bin"

docker exec -i omh-on-fhir\_mongo\_1 sh -c 'mongorestore --archive' < mongoDB\_backup\_2021-03-21.dump

**Now you have to stop and remove all of the containers by typing:**

cd "C:\Users\LGAdmin\Documents\OMH-on-FHIR"

docker-compose -f docker-compose.yml down -v

**Do whatever setting changes you need to do to docker (e.g. expanding the disk space as shown in the very beginning of this document).**

**Now restart all the container by typing:**

cd "C:\Users\LGAdmin\Documents\OMH-on-FHIR"

docker-compose -f docker-compose.yml up --force-recreate

**Now restore both the Postgres and Mongo databases using the AQUA options above.**

**Now restart any automated query Jobs that were suspended. AND WE’RE ALIVE AGAIN!!! 😊**

**This is the nuclear option to free up space where you lose everything if you didn’t backup!!!:**

**docker rmi -f $(docker images -a -q)**

* + 1. **TO REBUILD A SINGLE CONTAINER (e.g. the FHIR Client USING THE NAME LISTED IN THE YML FILE, NOT THE CONTAINER NAME), make changes and then type into powershell:**

**PS C:\Users\LGAdmin\Documents\OMH-on-FHIR> docker-compose -f docker-compose.yml up -d --build omh-on-fhir-client35**

* + 1. **This can also be useful for debugging to display the log file for a container (e.g. omh-on-fhir\_mongo\_1 to follow the last 5000 lines (50 pages) which you want to do because the file is ginormous! You can stop following it by typing Ctrl-C which gives you the command prompt back):**

**docker logs omh-on-fhir\_mongo\_1 -f –-tail 5000**

**THIS DOESN’T WORK: You can also delete the log file using (for e.g. omh-on-fhir\_mongo\_1, ALTHOUGH BE CAREFUL BECAUSE THIS CAN CORRUPT THE FILE IF TRYING TO WRITE TO IT, SO BEST TO PAUSE ANY AUTOMATED QUERIES FIRST):**

**echo "" > $(docker inspect --format='{{.LogPath}}' omh-on-fhir\_mongo\_1)**

**THIS WON’T HELP YOU EITHER: But you can find where log files are stored using:**

**docker exec -it XXX /bin/sh where XXX is the container ID**

**su –**

**cd /**

**find -name "\*.log"**

**To get info on the size of files in a directory, type (which shows the file size in the first column in bytes):**

**ls -lash**

**To see the size of directories, type:**

du -bsh \*

**And you can view what is in files using cat or nano, but you’ll need to install nano in the container. You do that with:**

su –

apt-get update

apt-get install sudo -y

* 1. Whenever you make a change to java files, it appears that you have to rebuild using gradle. This seems best done using IntelliJ IDEA. Use the “Terminal” at the bottom and in your root directory, type **gradlew build**



This may require being done on the development server, which then requires moving "H:\OMH-on-FHIR\shim-server\docker\shimmer.war" to the live server and then doing, for instance to rebuild the rmgresourceserver:

**PS C:\Users\LGAdmin\Documents\OMH-on-FHIR> docker-compose -f docker-compose.yml up -d --build resource-server**

1. **If you ever have an individual patient that stops synching,** you can go to the patient’s EHR and simply re-connect them. HOWEVER, if that doesn’t work, you need to:
   1. Find the patient’s shimmer ID based on their Medical Record Number by looking into the Postrgres database
      1. First, make sure that there’s not an instance of PostGres running on the computer outside of a container by using the Task Manager and stopping any background PostGres instances. Then Open the Postgres SQL Shell (psql) app (which you have to have downloaded and installed). Then enter (taking the default if you don’t see an entry):

**Server [localhost]:**

**Database [postgres]: mdata**

**Port [5432]:**

**Username [postgres]: mdata**

**Password for user mdata: mdata11!**

**mdata=# select \* from application\_user;**

**ehr\_id | shim\_key | shimmer\_id**

**--------+----------+------------**

* 1. Start MongoDB Compass (which you also need to have downloaded), and connect to **mongodb://localhost:27017**
  2. Open **omh\_dsu**
  3. Open **accessParameters**
  4. Find the record with the right shimkey and the username that matches the Shimmer ID. Click on the trash can and delete that record. Now you will be able to reconnect them from the EHR.

