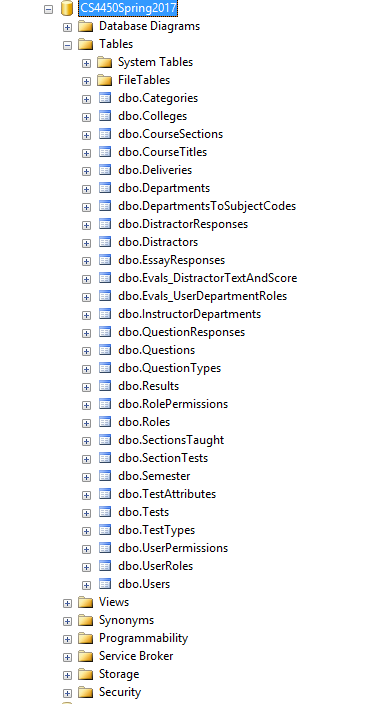
Before you begin setting up this API:

-----------------------------------------------------  
Speak with Rob Hilton about getting permission to access the CS4450Spring2017 database on Titan. At the time of writing this it should look like this:

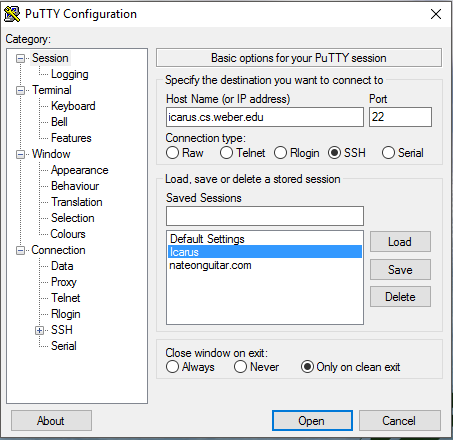
If it still exists you won’t have to generate the database all over again (the database creation scripts are included in the project if you need to rebuild this. It’s about 700,000 lines of code total, so reusing this will save you a lot of time)  
----------------------------------------------------------------------------------------------------------  
The API to handle database calls is written in PHP. It relies on a few packages that you will need to install using composer. Composer is a package manager similar to npm for JavaScript. If you want to learn about composer on your own, visit https://getcomposer.org/

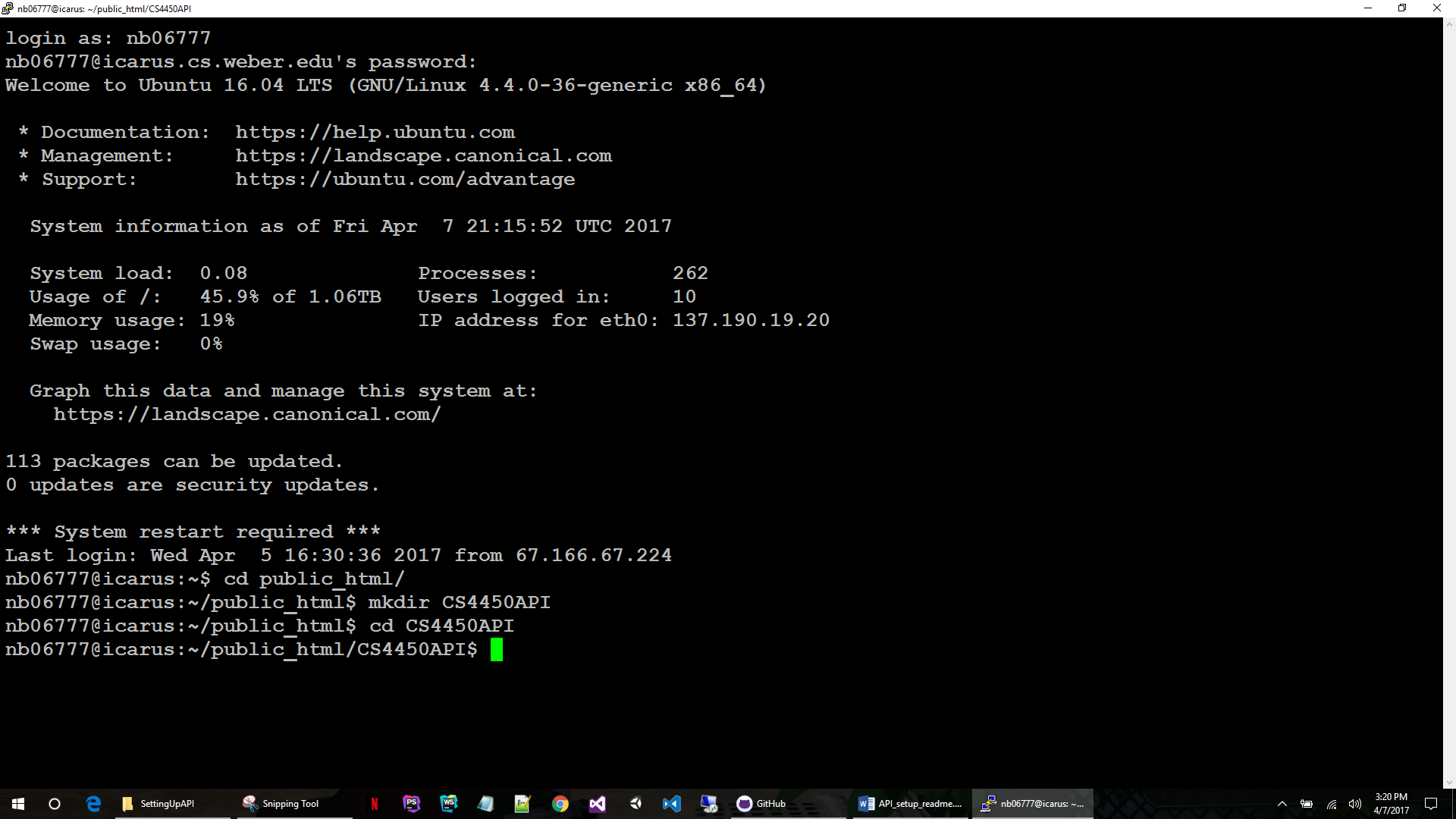


You do not need to learn extra about composer, this readme will explain the parts you need to use composer to get the libraries that the API relies on.

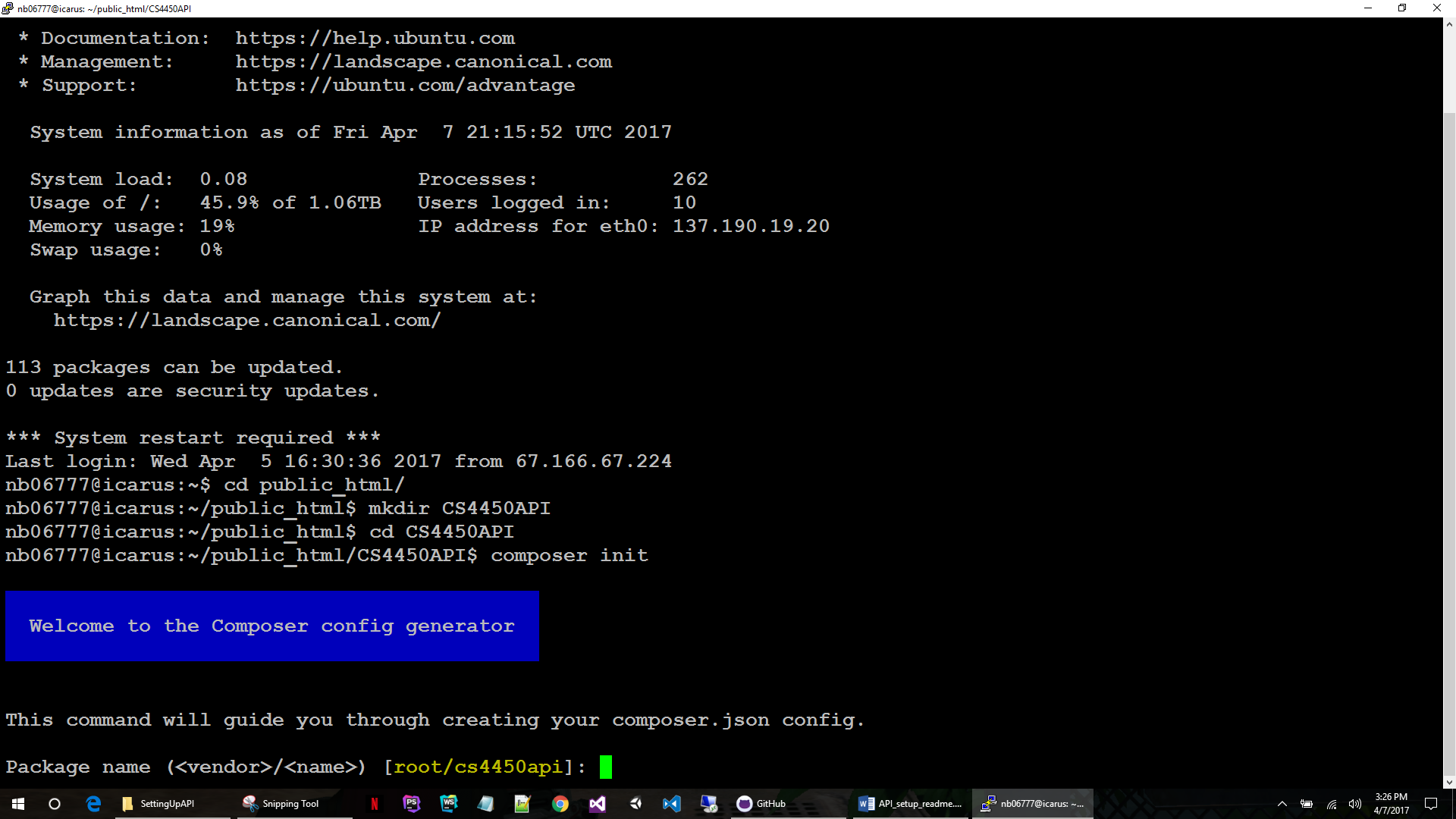
Composer will instantiate a PSR-4 autoloader so you don't need to "require" anything but the file that runs the autoloader (require is like C++'s #include or Java's import statement). The packages that will be autoloaded are called “firebase/php-jwt” and “nikic/fast-route”. Icarus has composer installed, so that is a viable option for hosting this API, but you can use any PHP and composer enabled server to host this. We will go through setting it up on Icarus as that is where we know it will work the way it is supposed to.

SSH into Icarus using this information and sign in using your username and password.

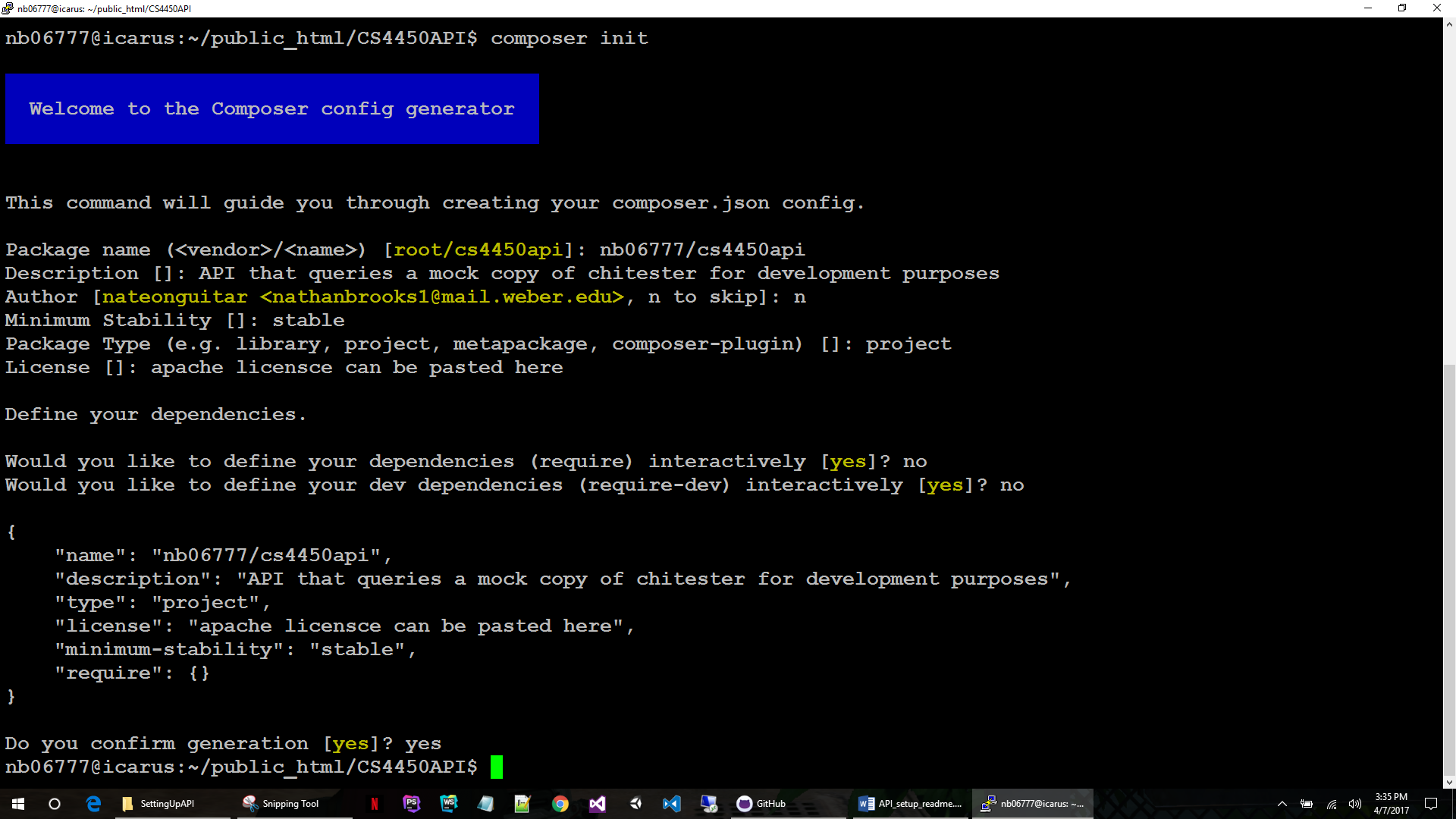
  
  
Every CS student has a ~xx01234 directory, you will automatically be placed in your directory when you log in. Inside this directory public\_html directory, CD into it. Make a new directory here and call it whatever you wish, we will call ours CS4450API.

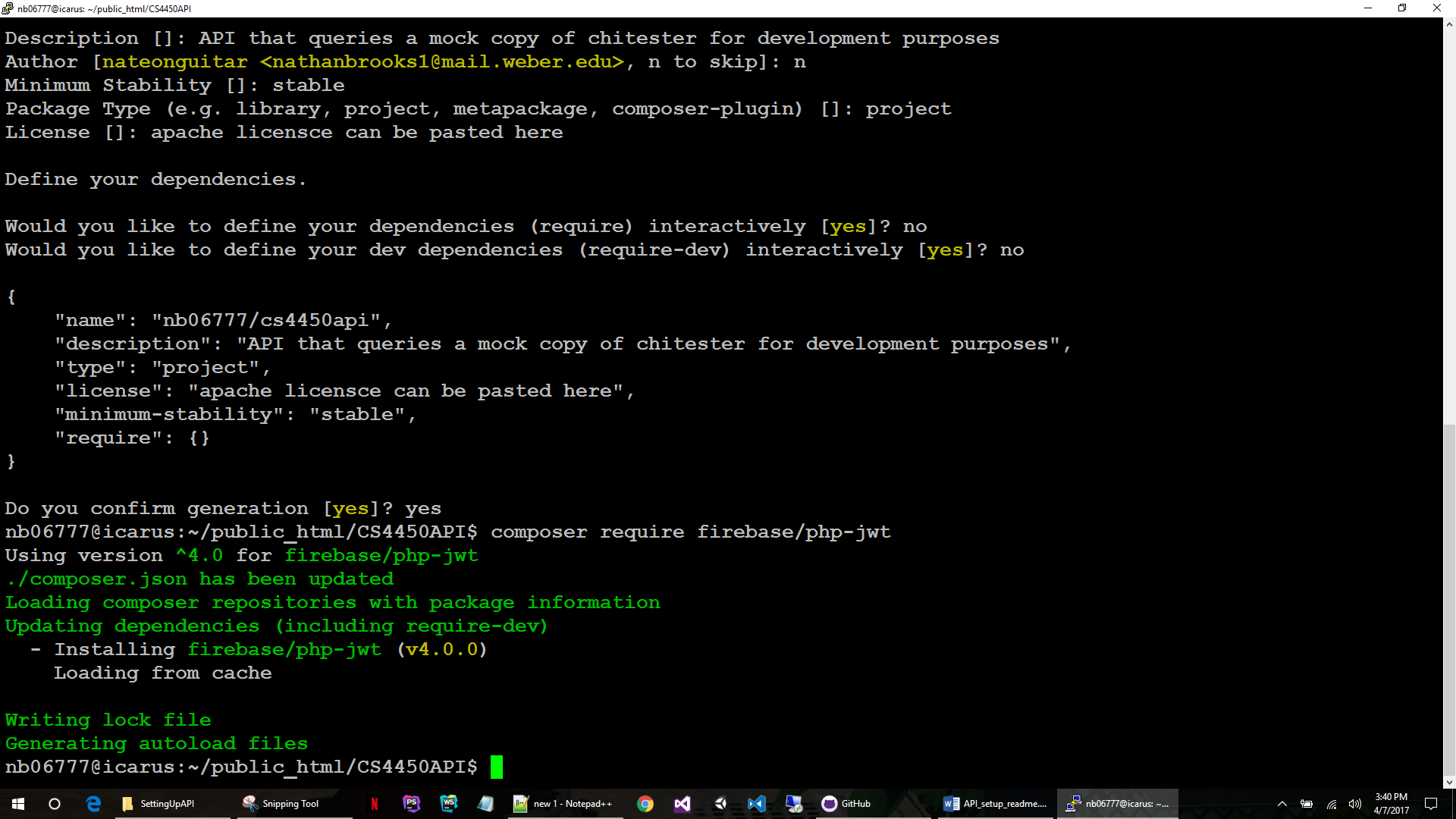


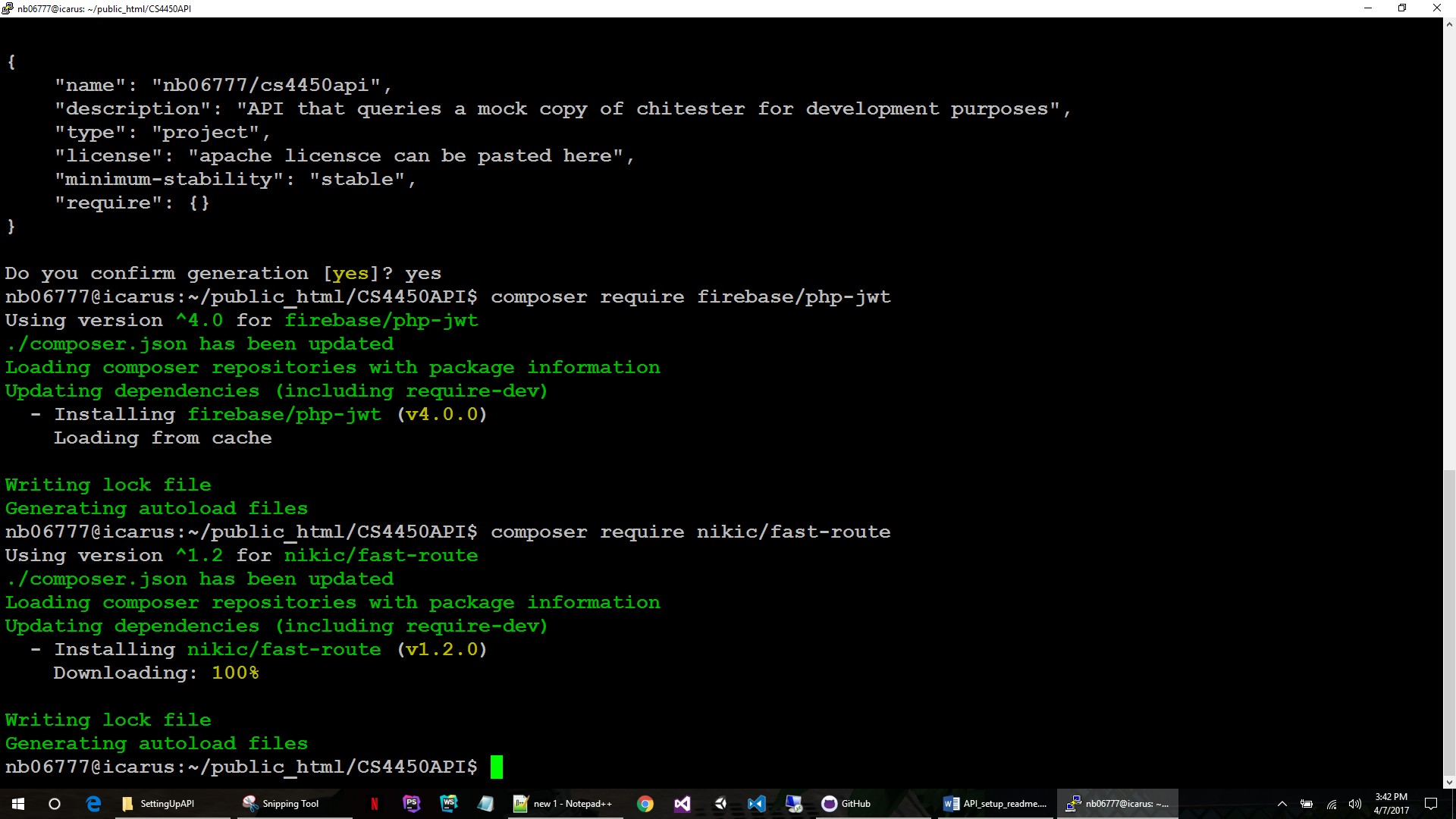
Run “composer init”

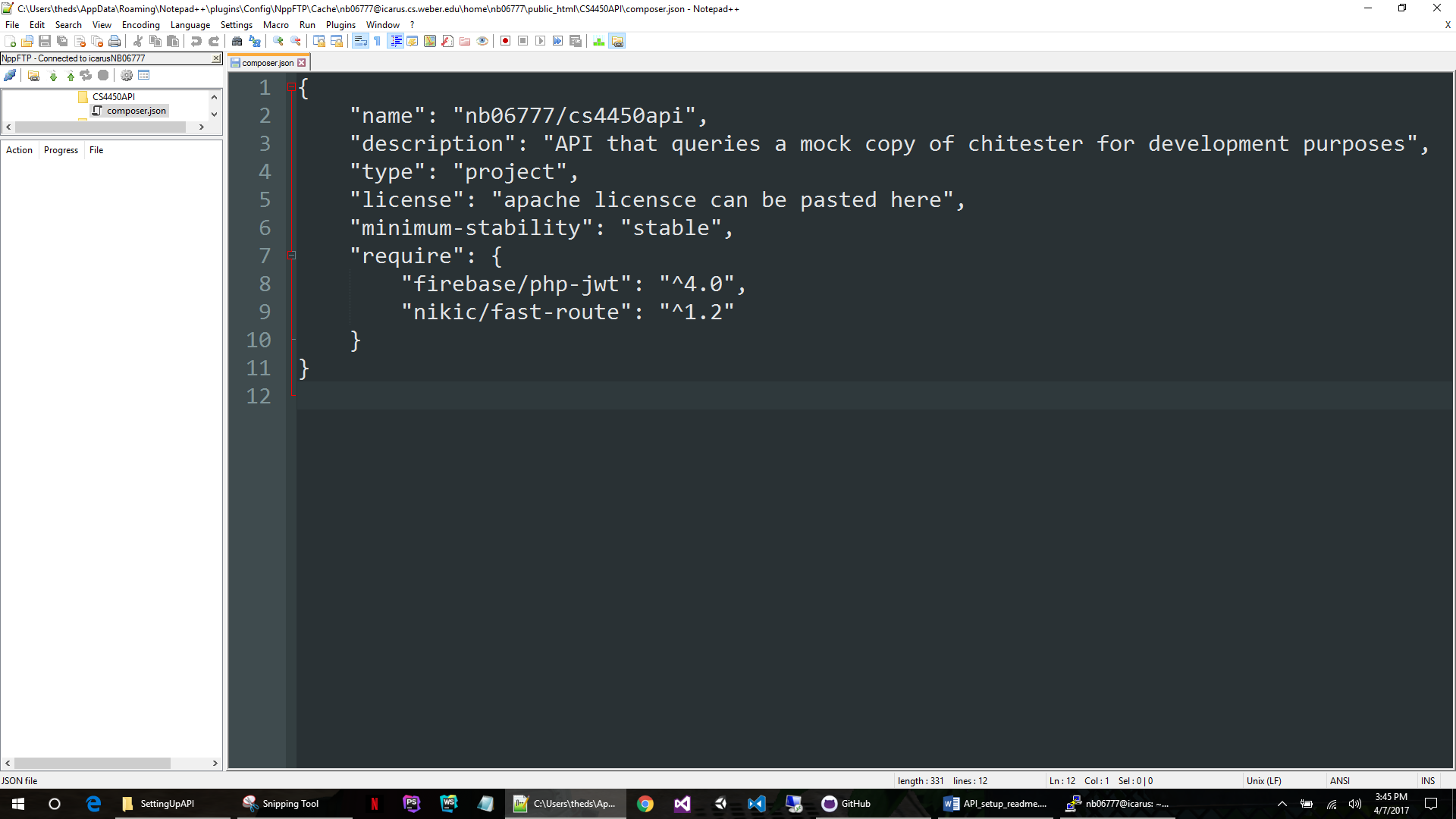


* Give it
  + a creator name / project name,
  + a description,
  + an author name if you wish,
  + a minimum stability
    - (stable is recommended, we don’t want to rely on anything that’s not stable),
  + tell composer that this will be a “project”,
  + paste in a license agreement, or come up with one yourself
    - (you can also just put in a single character to skip this)
  + say “no” to defining dependencies interactively
    - (we are only importing 2, it’s just easier to do it manually)
  + Say “no” again, same reason
  + Say “yes” to confirm generating this json file that composer will rely on for autoloading and importing
    - (we will modify it manually in a minute)

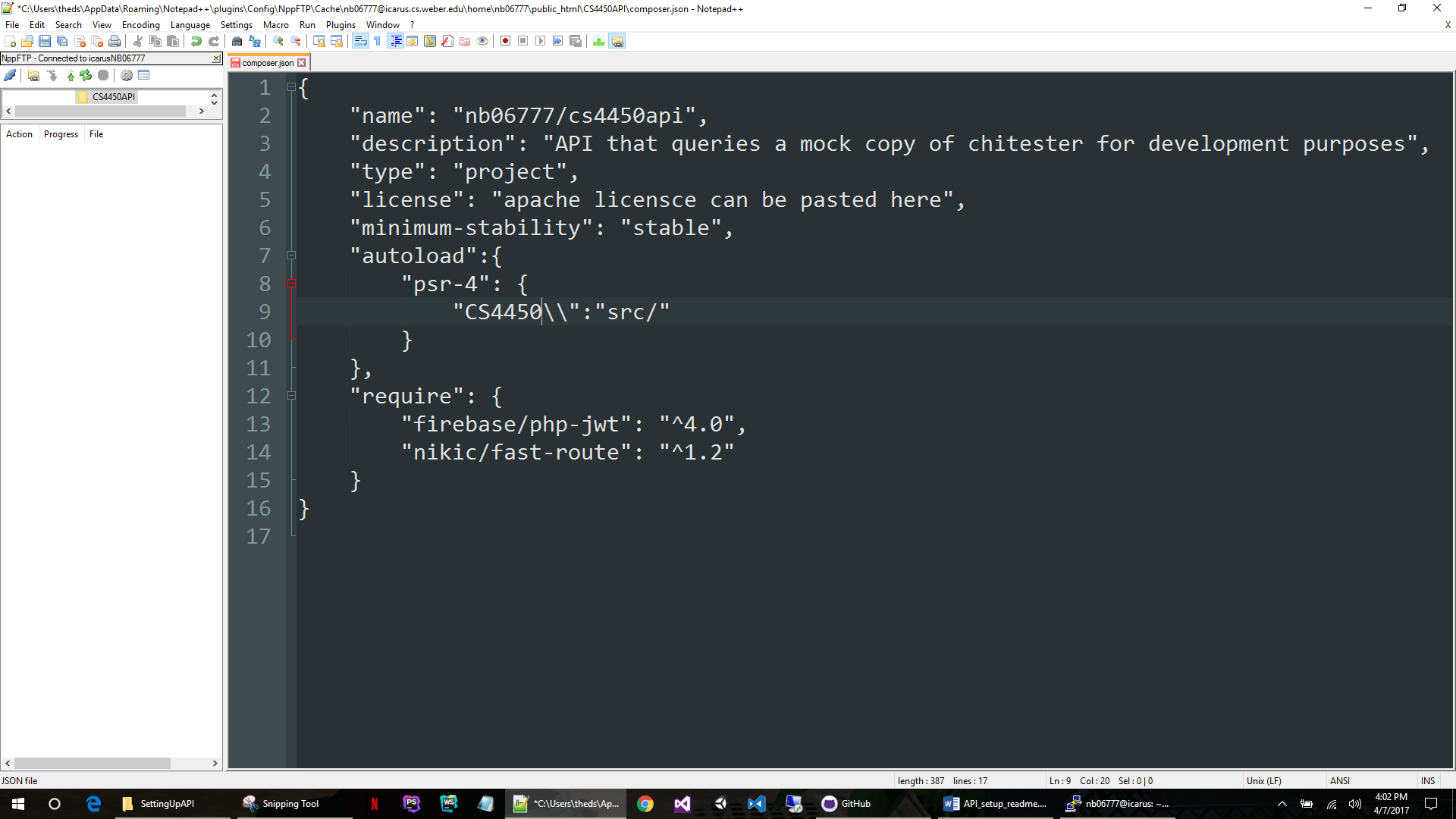


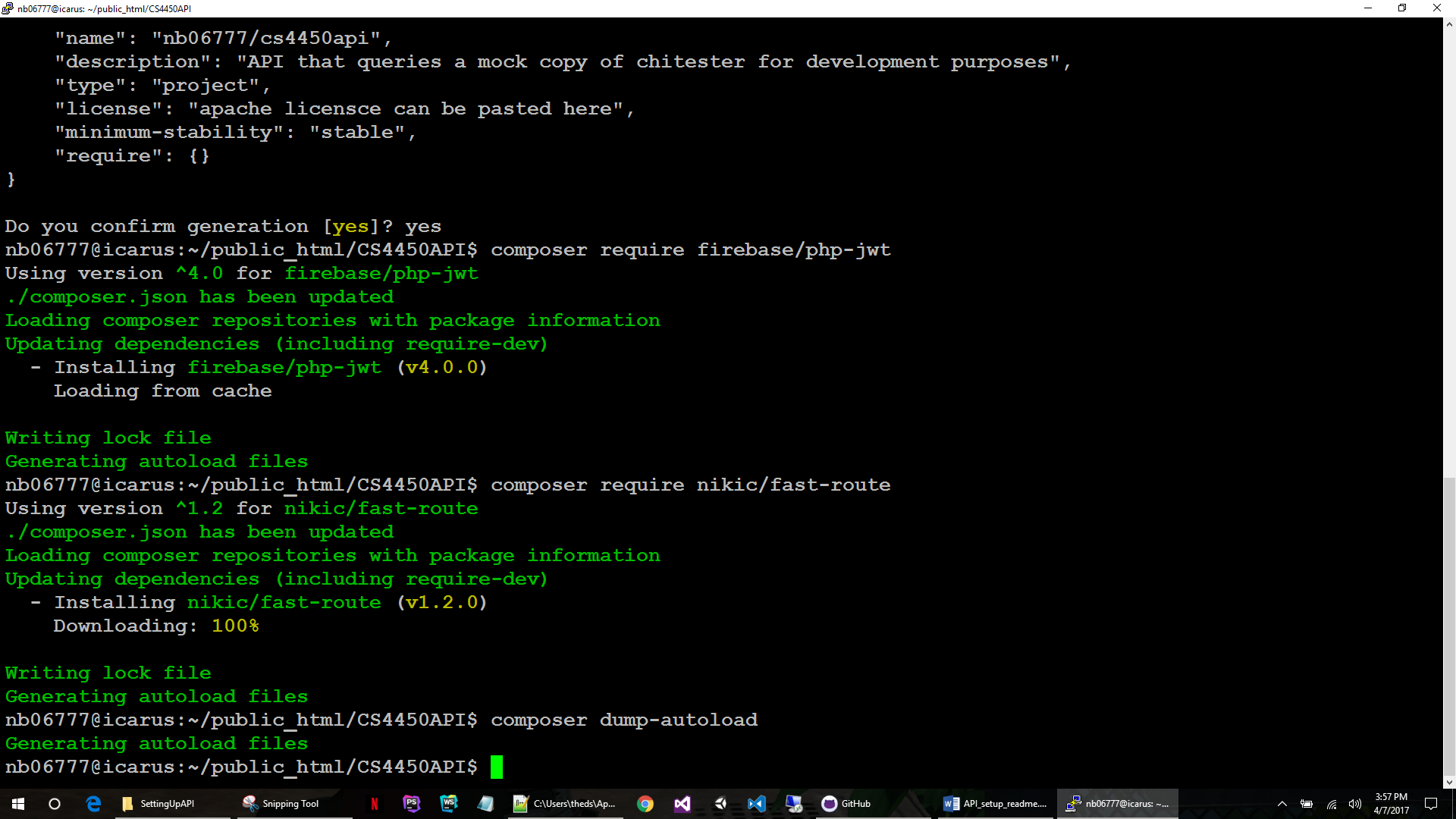
Once this is created we want to tell composer to download the libraries we need from the repository (I believe it will use <https://packagist.org/> by default, you can search for other useful packages there)  
  
type: composer require firebase/php-jwt  
(this might take a minute to install, let it run)  
  
  
  
type: composer require nikic/fast-route   
(again, this might take a minute to install, let it run)



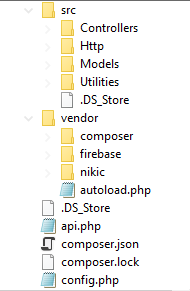
Now open the composer.json file that was created by composer for editing. You can edit it using something like vim or you can download it, modify it, and reupload. As long as it is modified correctly the method does not matter.  
  


We need to tell it what kind of autoloader to use. This information MUST go before the “require” line or it won’t work.  
  
we will insert a new object as “autoload” and tell to use a “psr-4” autoloader, then give it a namespace, we will use CS4450 and tell it what directory to include in this namespace. It’s a little confusing, but if you look at the example picture you will see that we are giving the namespace “CS4450” to anything in the “src” directory.

The lines that were added were from 7 to 11, don’t forget the comma at the end of line 11.  


Save this file back to Icarus and go back to the command prompt.   
  
type: composer dump-autoload  
  
This will tell composer to rebuild dependency files to reflect the changes we made to composer.json  
  
  
Now it’s time to upload all of the API files from the project in to the correct directories. You can zip it, upload it, and unzip it, or just move files one at a time (there’s not that many). The structure will look like this:

Create an “src” directory and start moving files over.  
  
  
  
You may not need the .DS\_Store file, I don’t remember making it. I would try without first.  
  
  
  
----------------------------------------------------------------------------------------------------------



If you would like to see Josh Jensen explain the composer setup process, visit <https://www.youtube.com/watch?v=EukLS5fdXCQ>  
Hopefully it’s still live. It is a little on the low quality side and he takes a long time to explain. Hopefully with these two tutorials combined you can get this working.