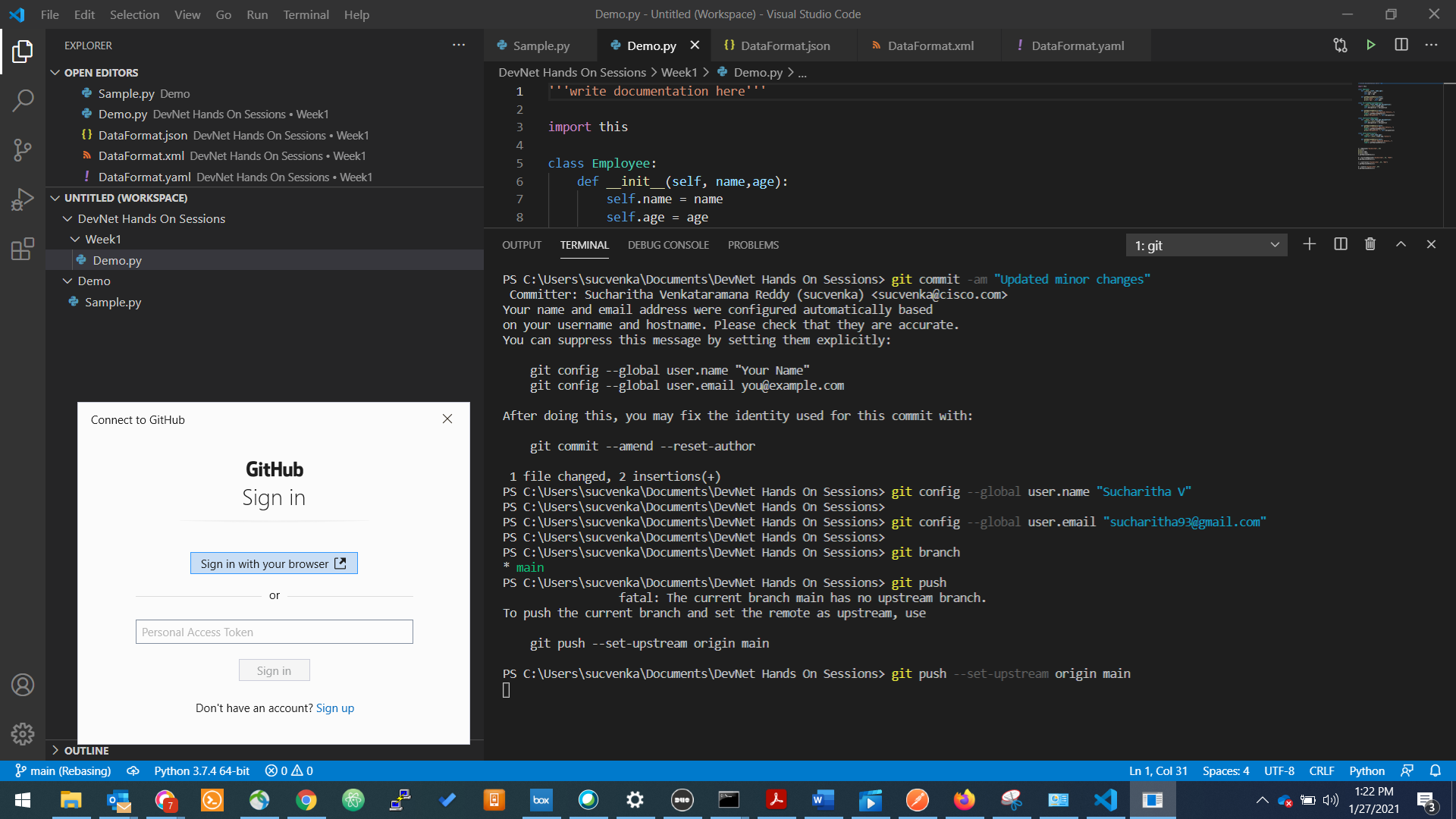
**Week 2 SVS DevNet Hands On**

***Git Refresher***



PS C:\Users\sucvenka\Documents\DevNet Hands On Sessions> git commit -am "Updated minor changes"

Committer: Sucharitha Venkataramana Reddy (sucvenka) <sucvenka@cisco.com>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 file changed, 2 insertions(+)

PS C:\Users\sucvenka\Documents\DevNet Hands On Sessions> git config --global user.name "Sucharitha V"

PS C:\Users\sucvenka\Documents\DevNet Hands On Sessions>

PS C:\Users\sucvenka\Documents\DevNet Hands On Sessions> git config --global user.email "sucharitha93@gmail.com"

PS C:\Users\sucvenka\Documents\DevNet Hands On Sessions>

PS C:\Users\sucvenka\Documents\DevNet Hands On Sessions> git push

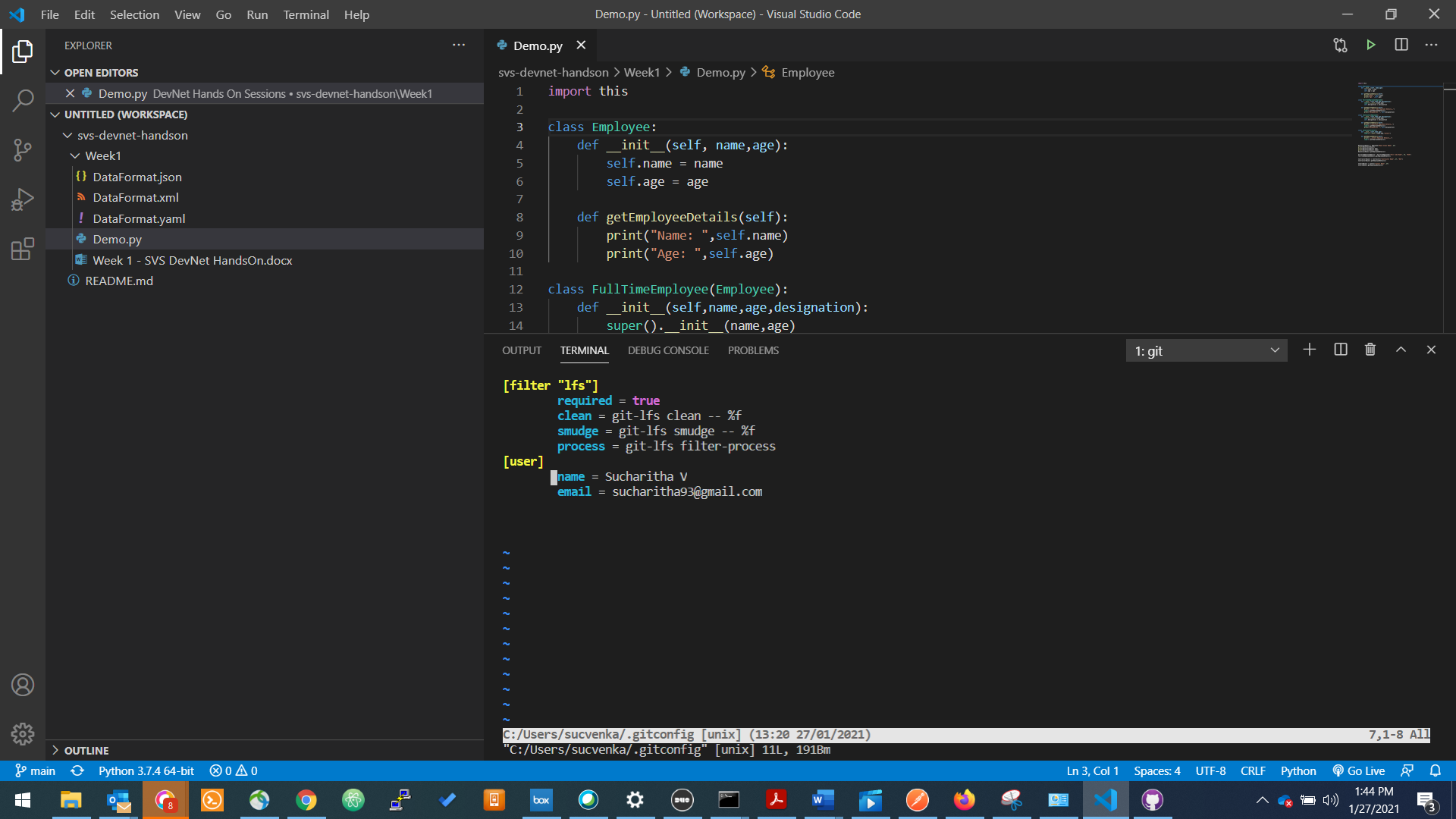
fatal: The current branch main has no upstream branch.

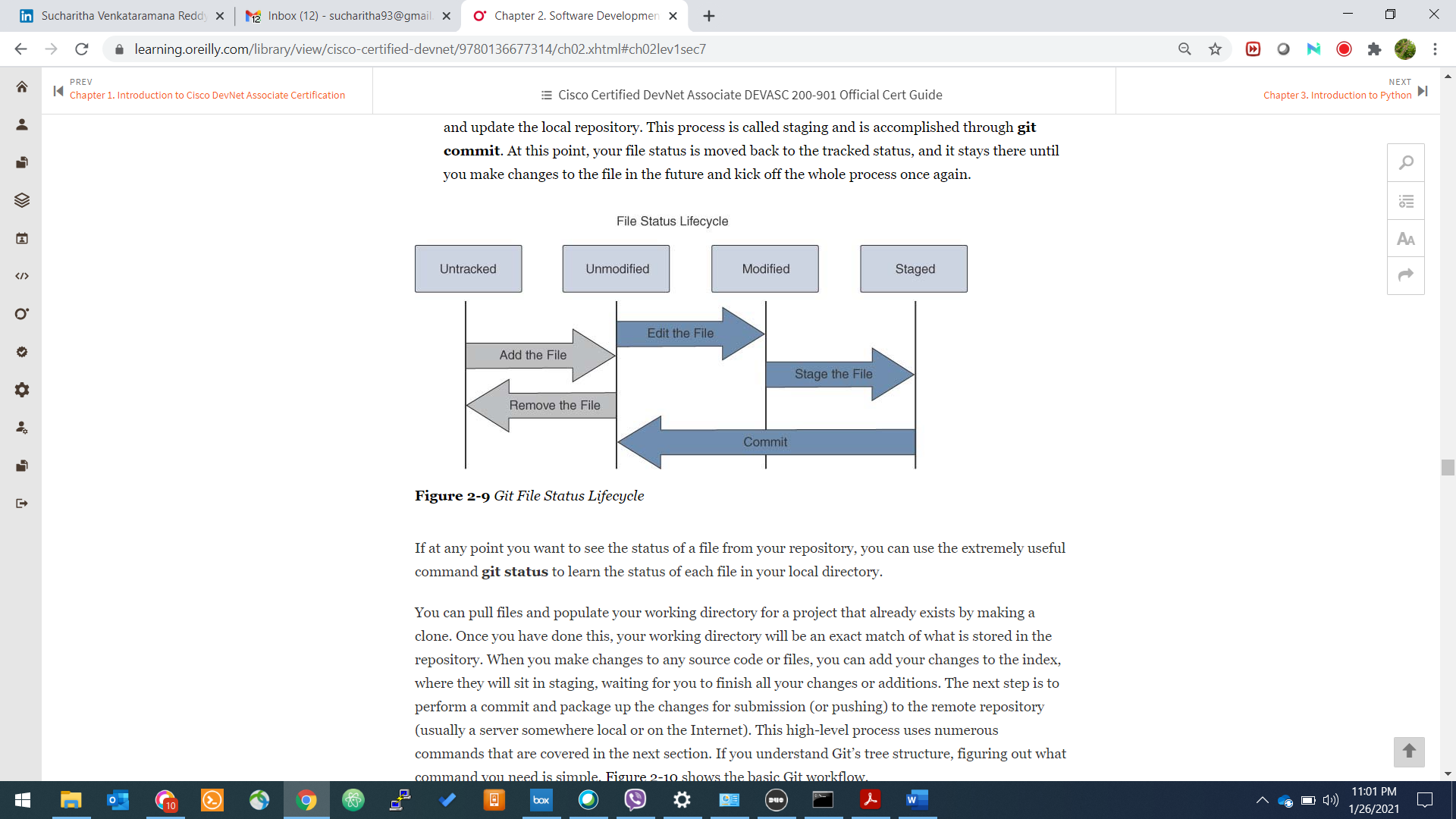
To push the current branch and set the remote as upstream, use

git push --set-upstream origin main

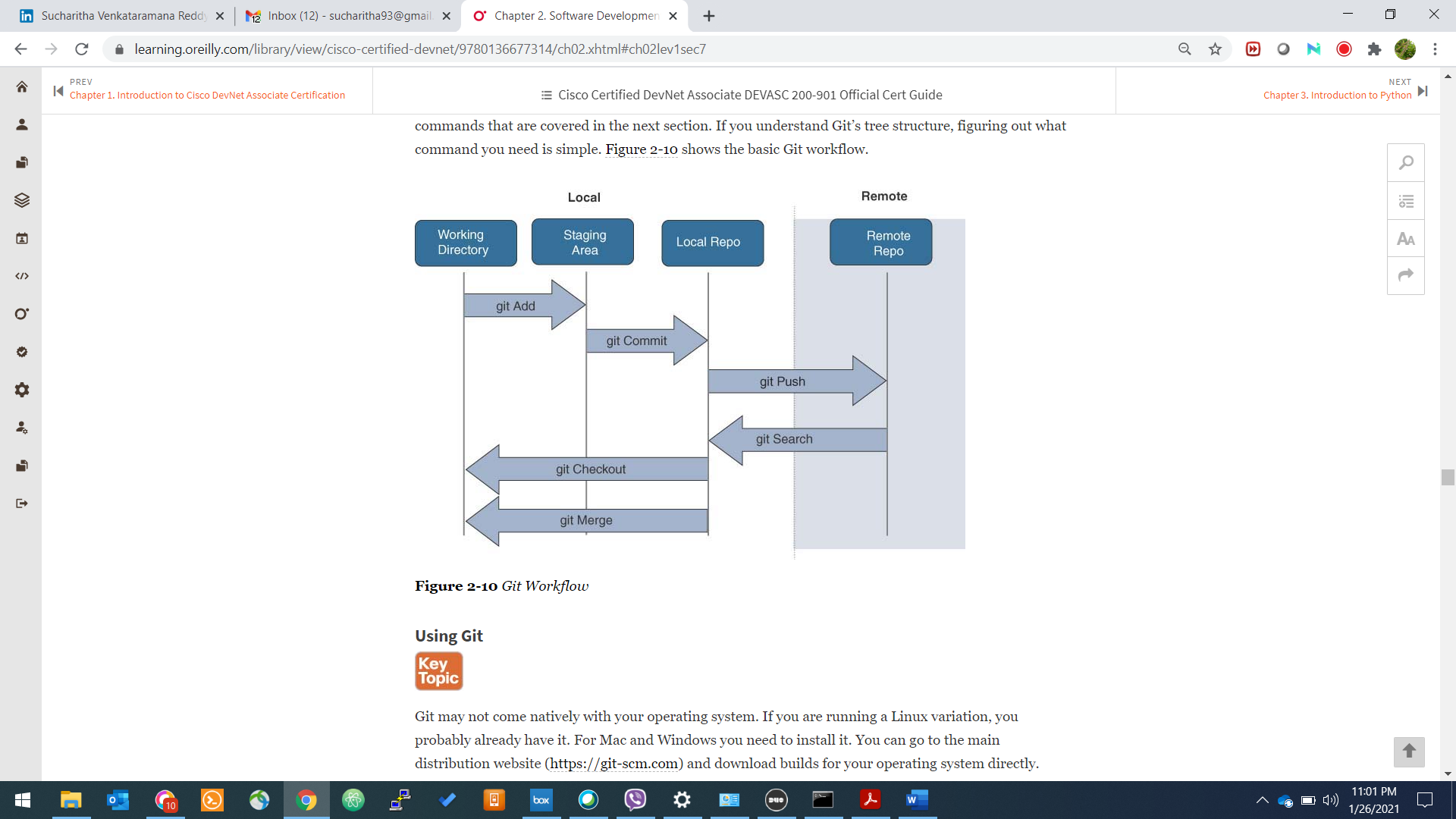
PS C:\Users\sucvenka\Documents\DevNet Hands On Sessions> git push --set-upstream origin main

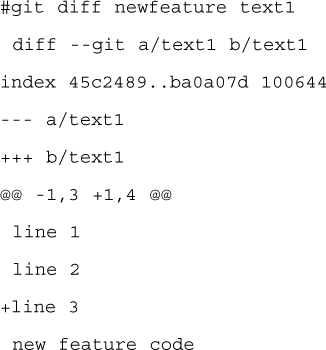
PS C:\Users\sucvenka\Documents\DevNet Hands On Sessions\svs-devnet-handson> git config --global -e





Git Workflow



[](https://learning.oreilly.com/library/view/cisco-certified-devnet/9780136677314/ch02.xhtml#ppg54-2a)

***API***

it is important that all applications should be able to communicate with other applications without depending on the underlying operating system and the programming languages. Web services are used to create such applications.

Web Services

A web service is a collection of standards and protocols that applications and systems use for exchanging data over the internet. A web service could be written in any programming language and is OS-independent.

For instance, an application built in PHP running on a Linux server can communicate with an Android application built using Java and running on an Android operating system.

What Is REST?

REST stands for Representational State Transfer. It is a stateless software architecture that provides many underlying characteristics and protocols that govern the behavior of clients and servers.

What Is Meant by RESTful?

An API that has following features is known as RESTful API:

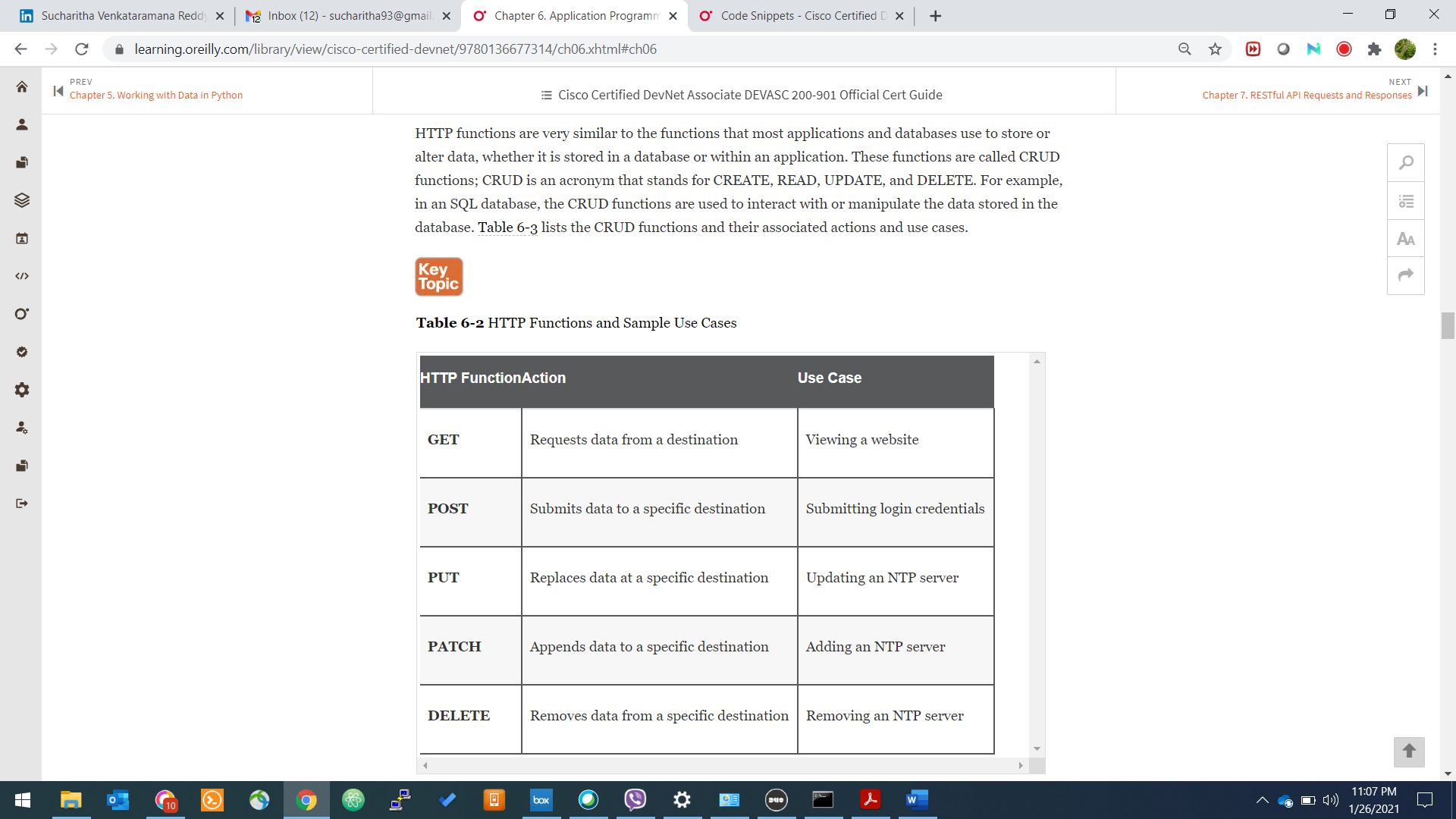
Client-server architecture: The client is the front-end and the server is the back-end of the service. It is important to note that both of these entities are independent of each other.

Stateless: No data should be stored on the server during the processing of the request transfer. The state of the session should be saved at the client’s end.

Cacheable: The client should have the ability to store responses in a cache. This greatly improves the performance of the API.

What Is a REST API?

A RESTful API (also known as a RESTful web service) is a web service implemented using HTTP protocol and the principles of REST. It is a collection of resources that employ HTTP methods (GET, PUT, POST, DELETE).



Popular RESTful API request formats:

* REST
* XML-RPC
* SOAP

Popular RESTful API response formats:

* REST
* XML-RPC
* SOAP
* JSON
* PHP

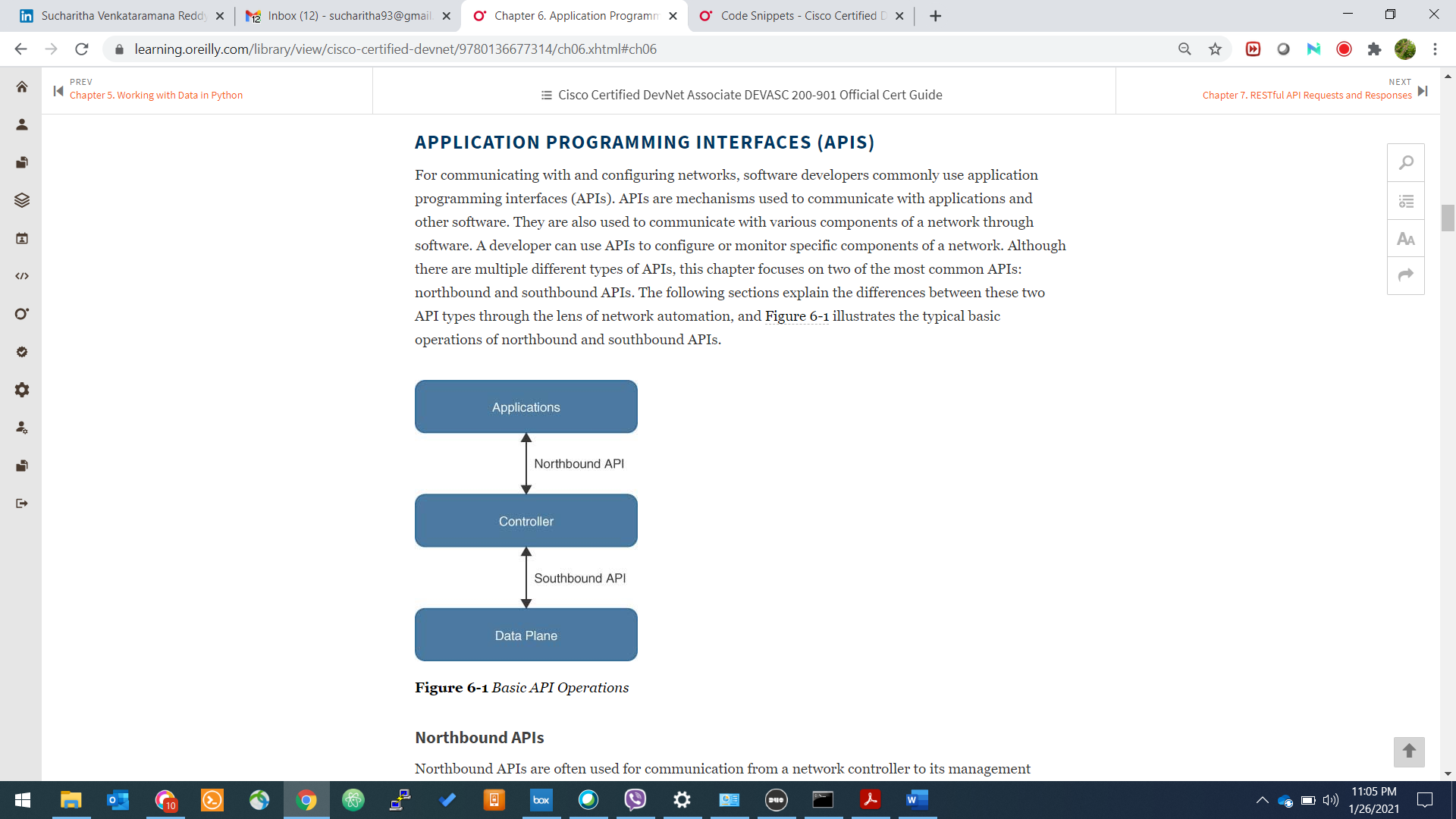
Why REST API?

A RESTful API is used to make applications distributed and independent over the internet with the aim of enhancing the performance, scalability, simplicity, modifiability, visibility, portability, and reliability of the application.

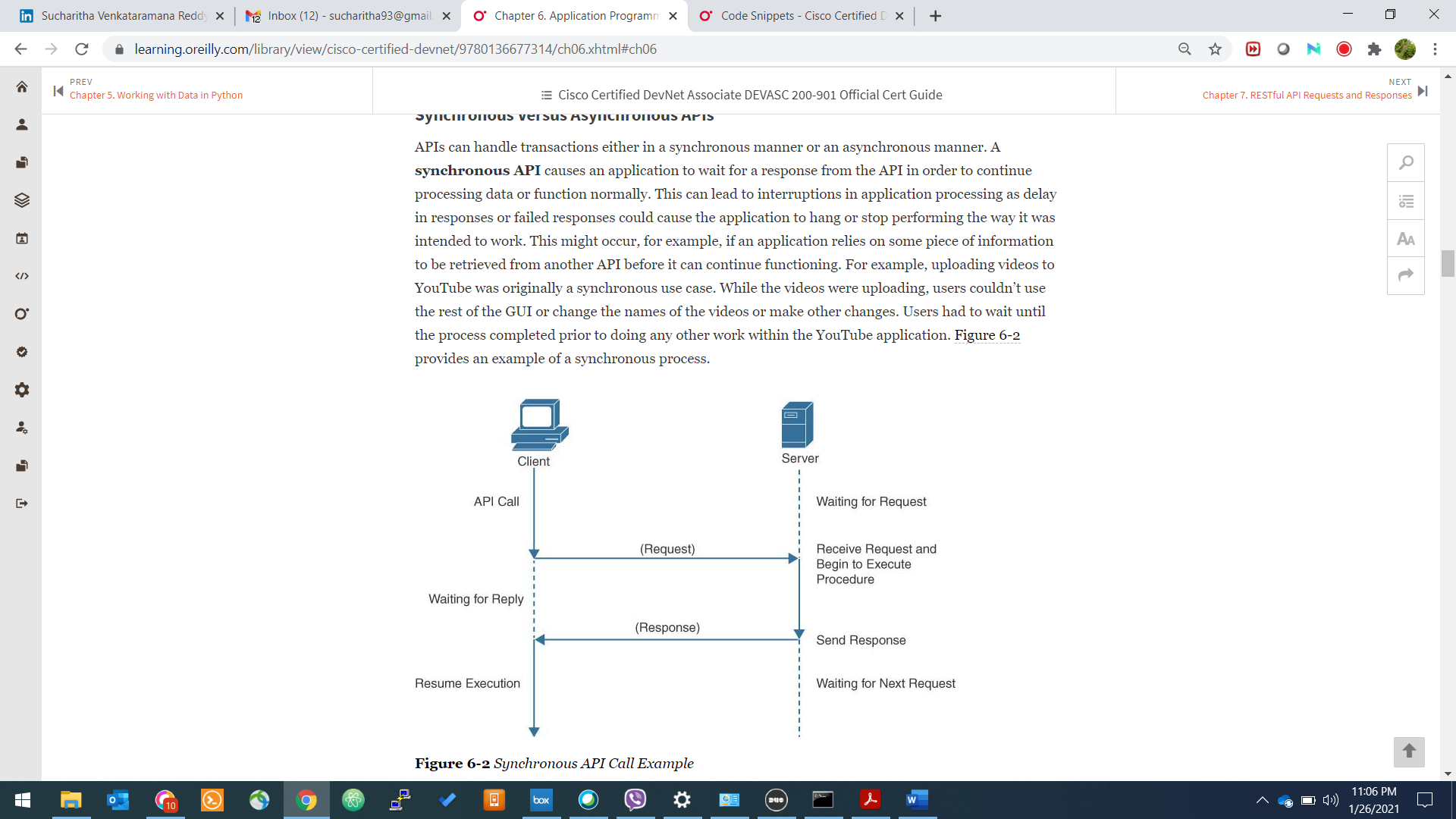
REST Constraints:

* Client / Server
* Stateless
* Cache
* Uniform Interface
* Layered System
* Code On Demand

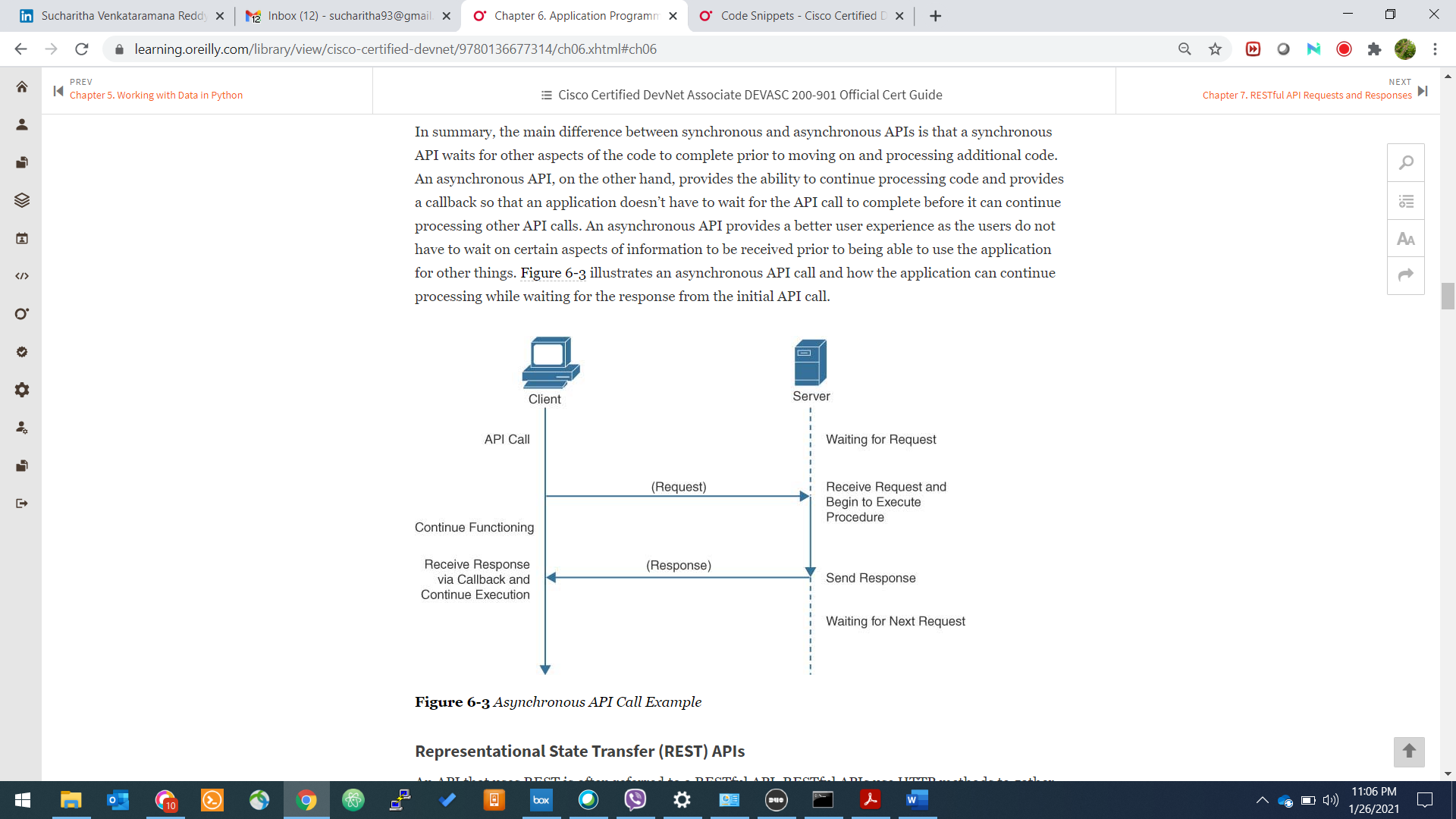
Basic API Operations in our context



Synchronous Calls

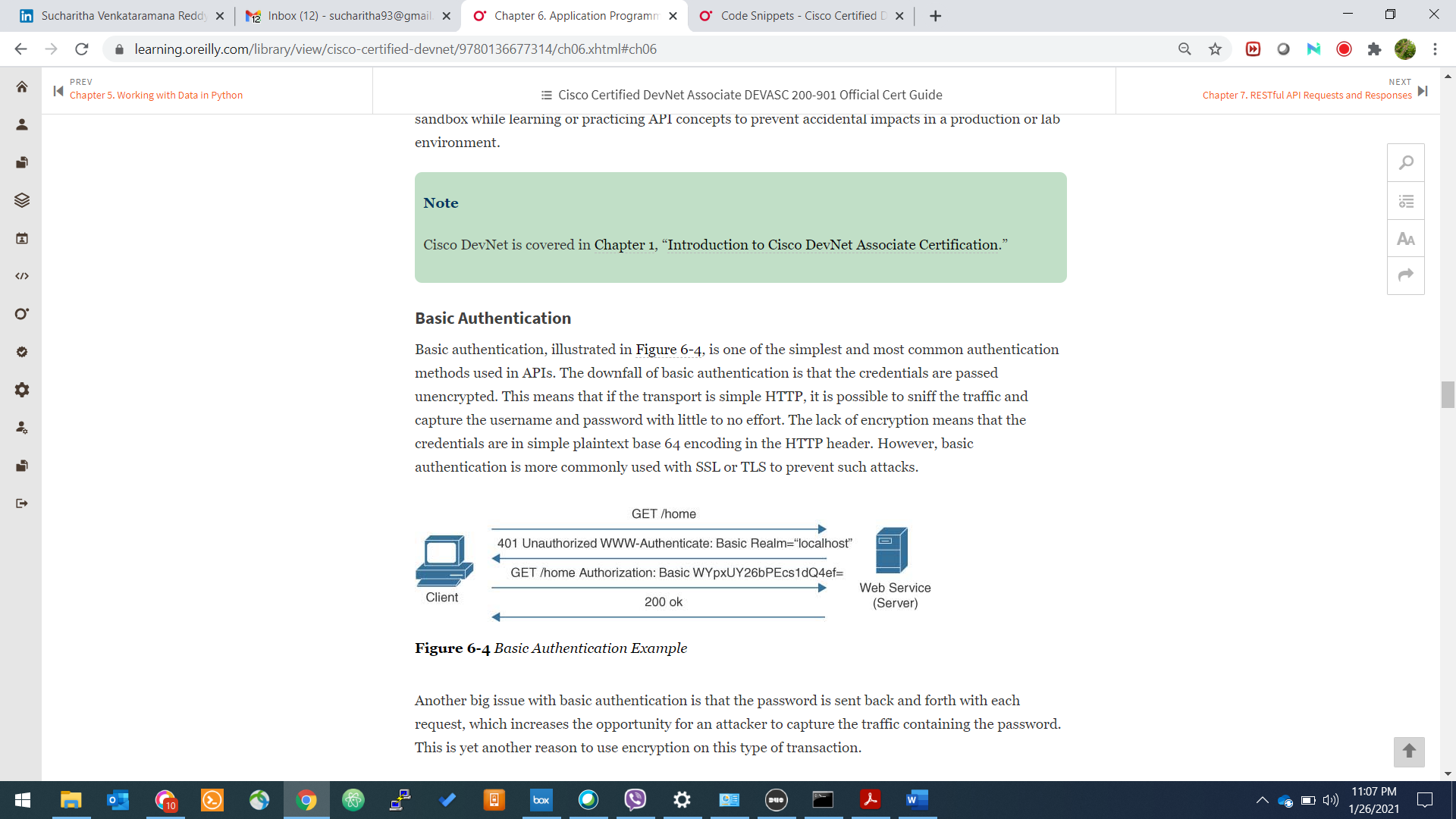


Asynchronous Calls



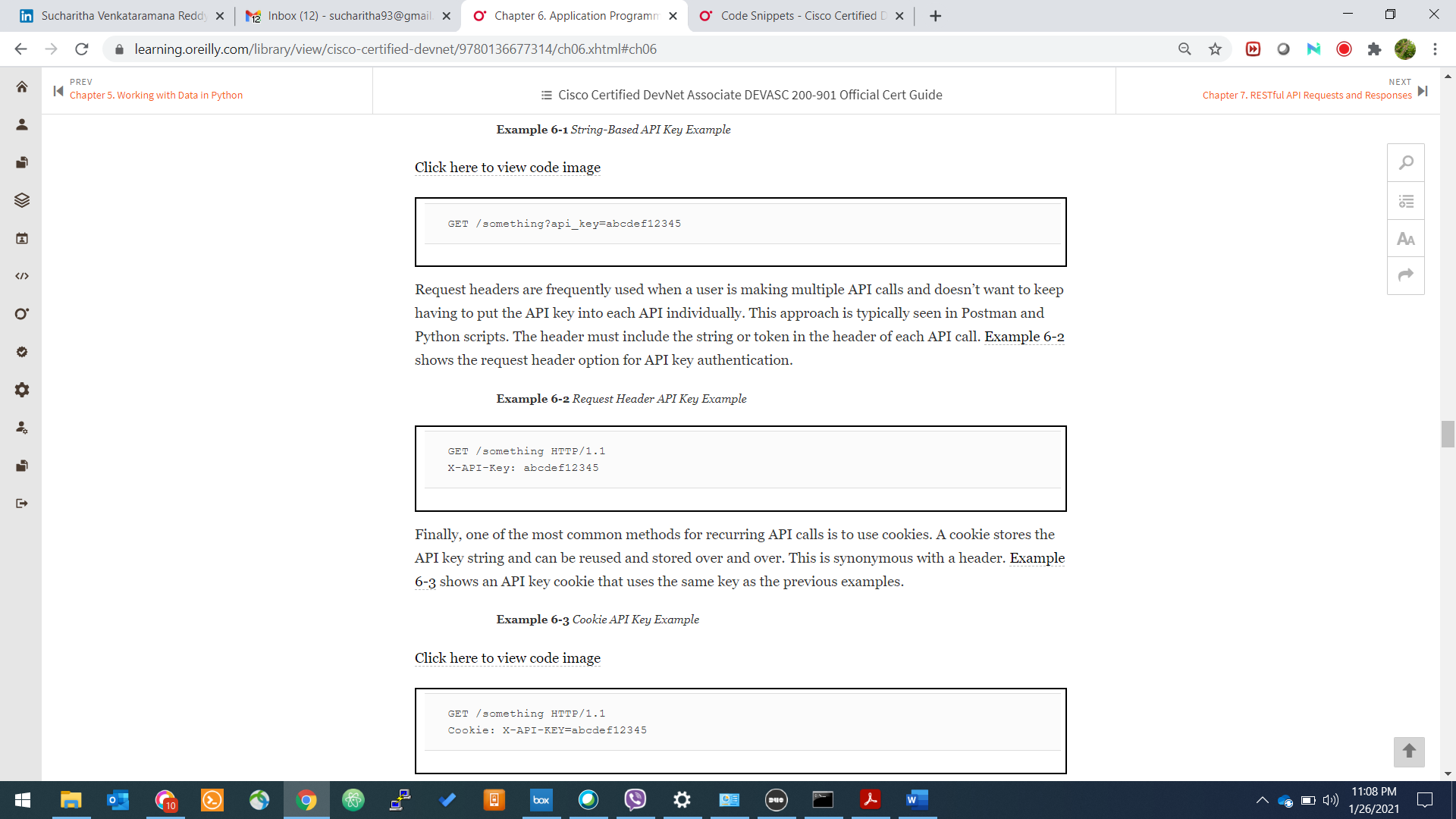
Authentication Mechanisms :

* Basic Authentication

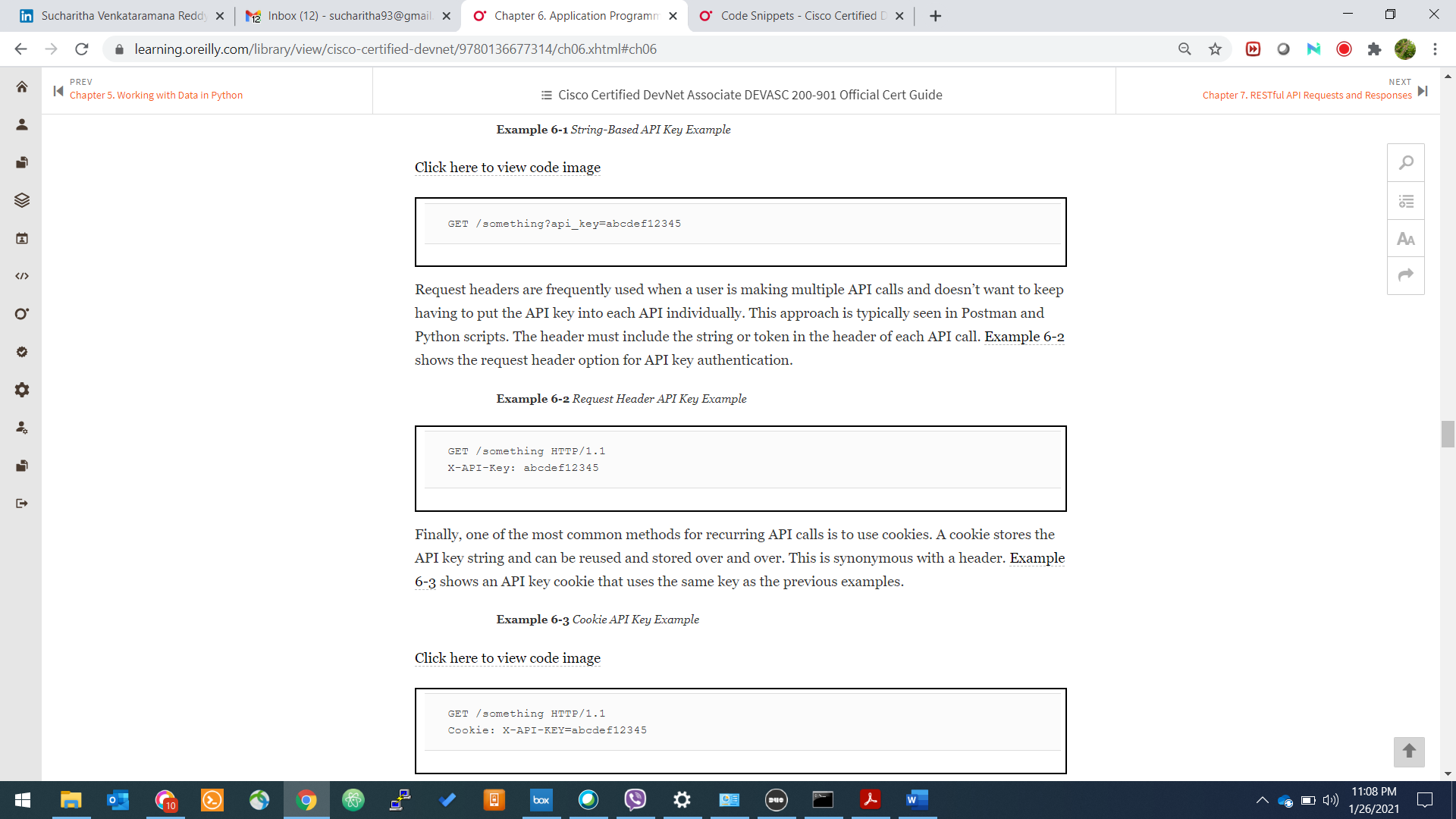


Issues:

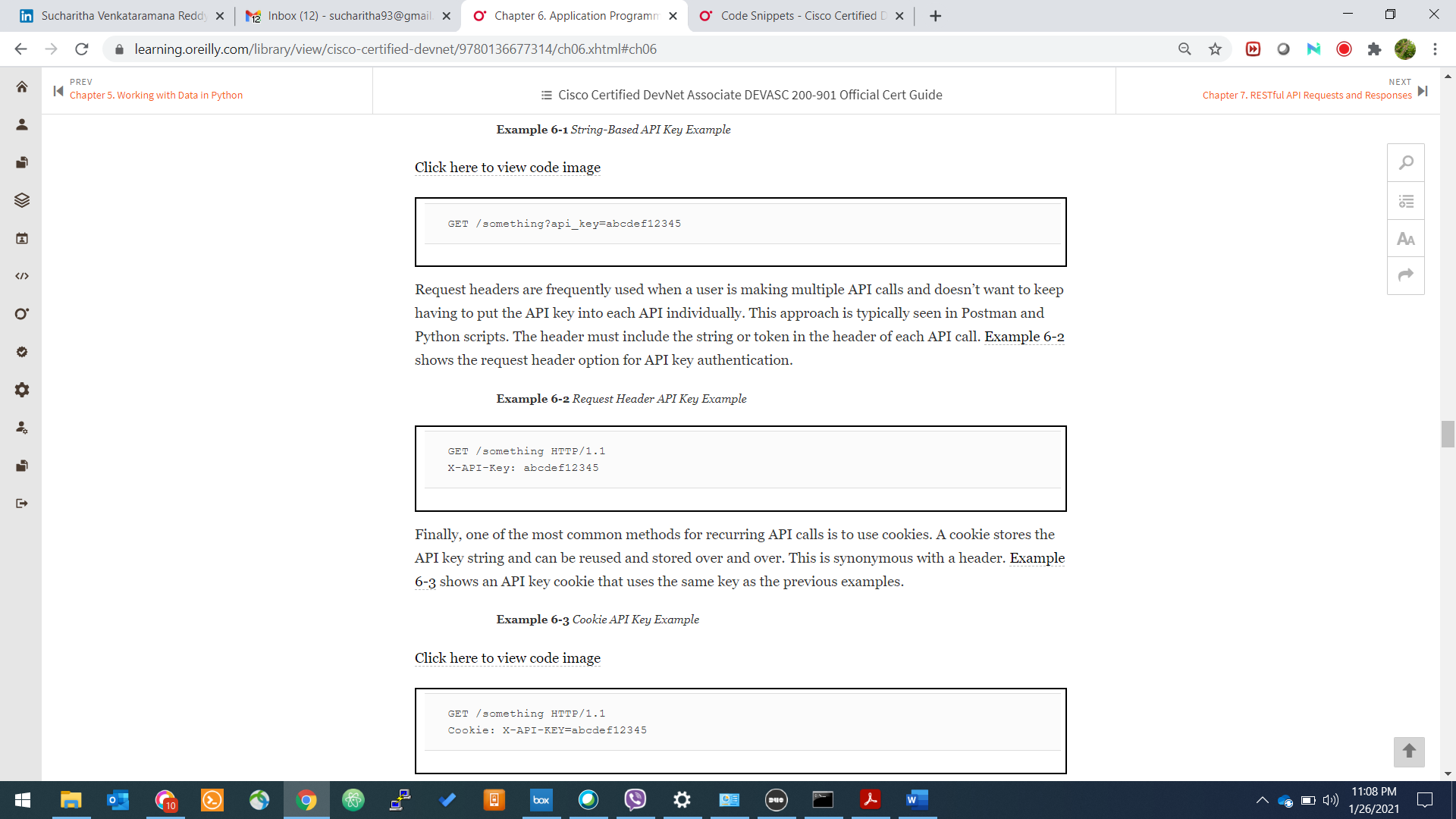
* + Username & Password are sent across in every request & response
  + The credentials are shared unencrypted and is more prone to attack
* API Key
  + String



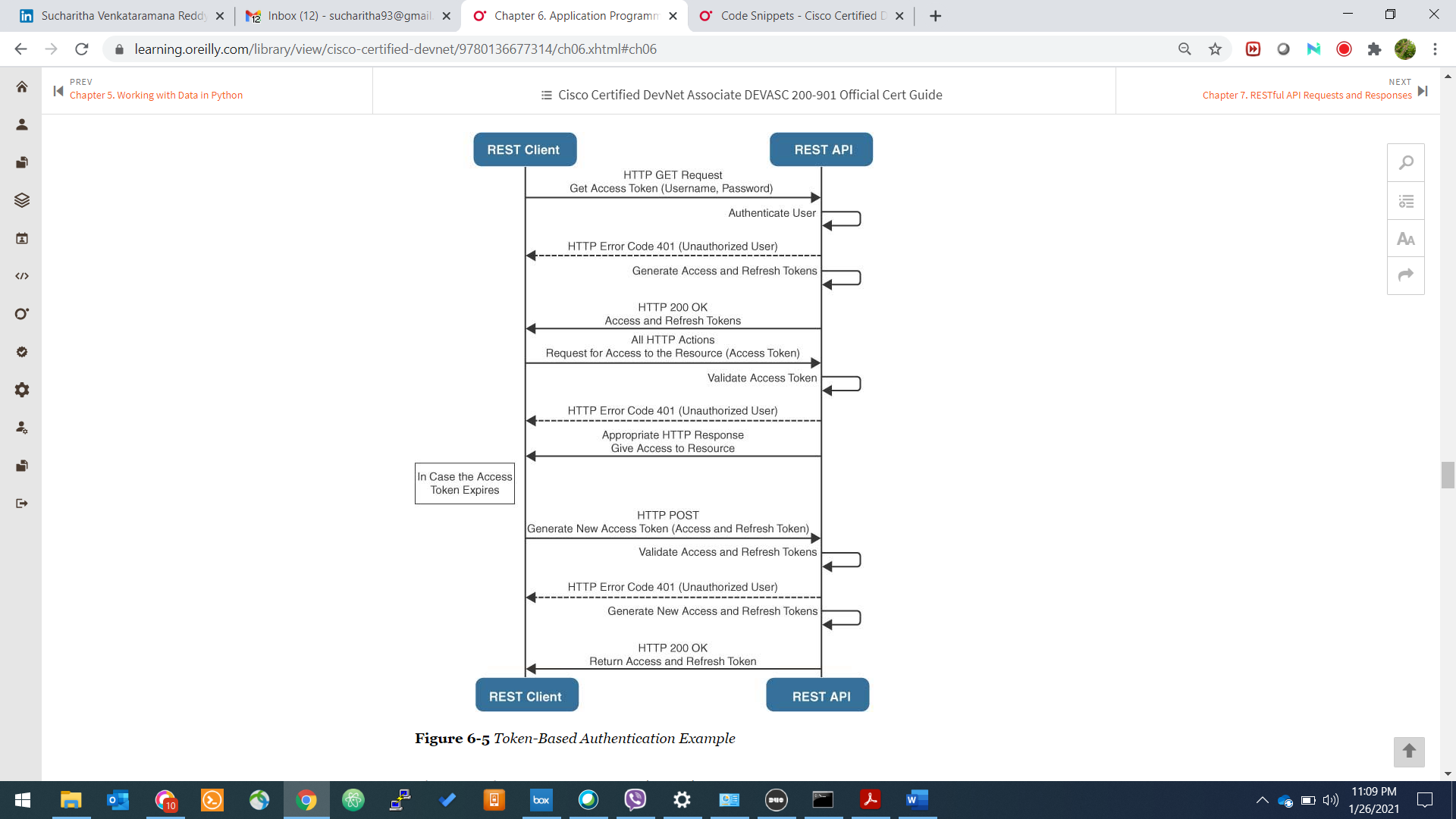
* + Request Header



* + Cookie

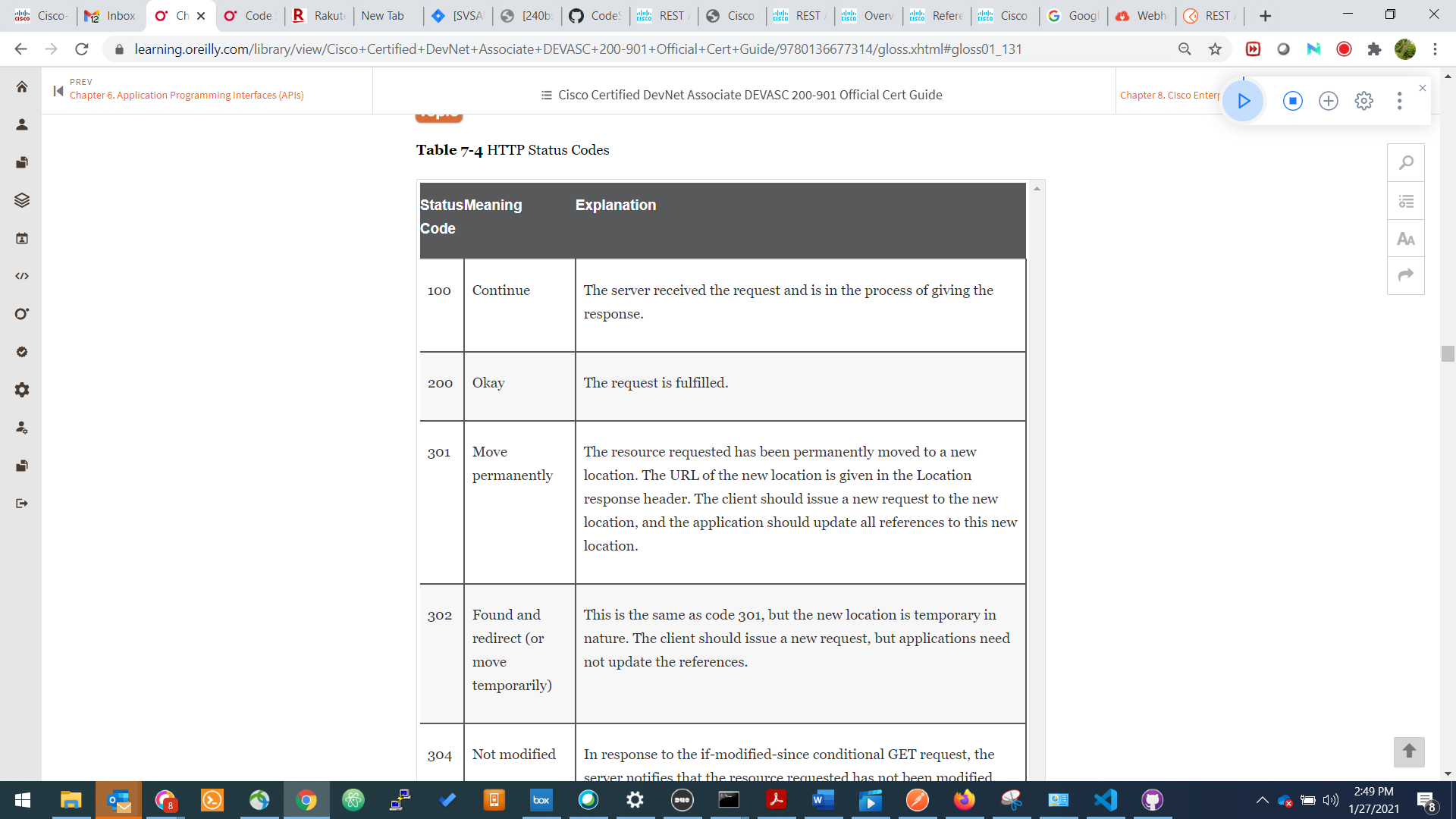


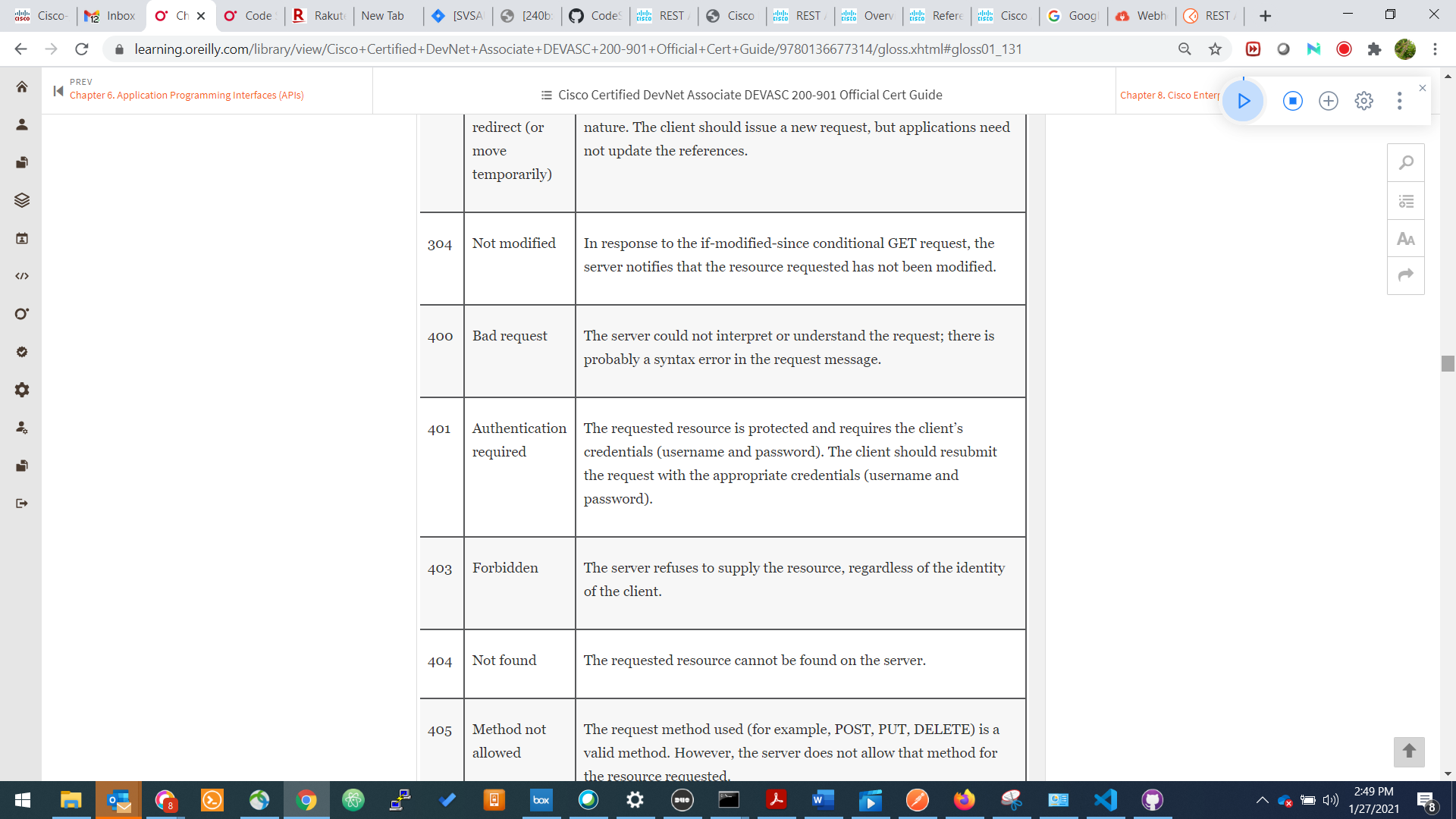
* Token Based

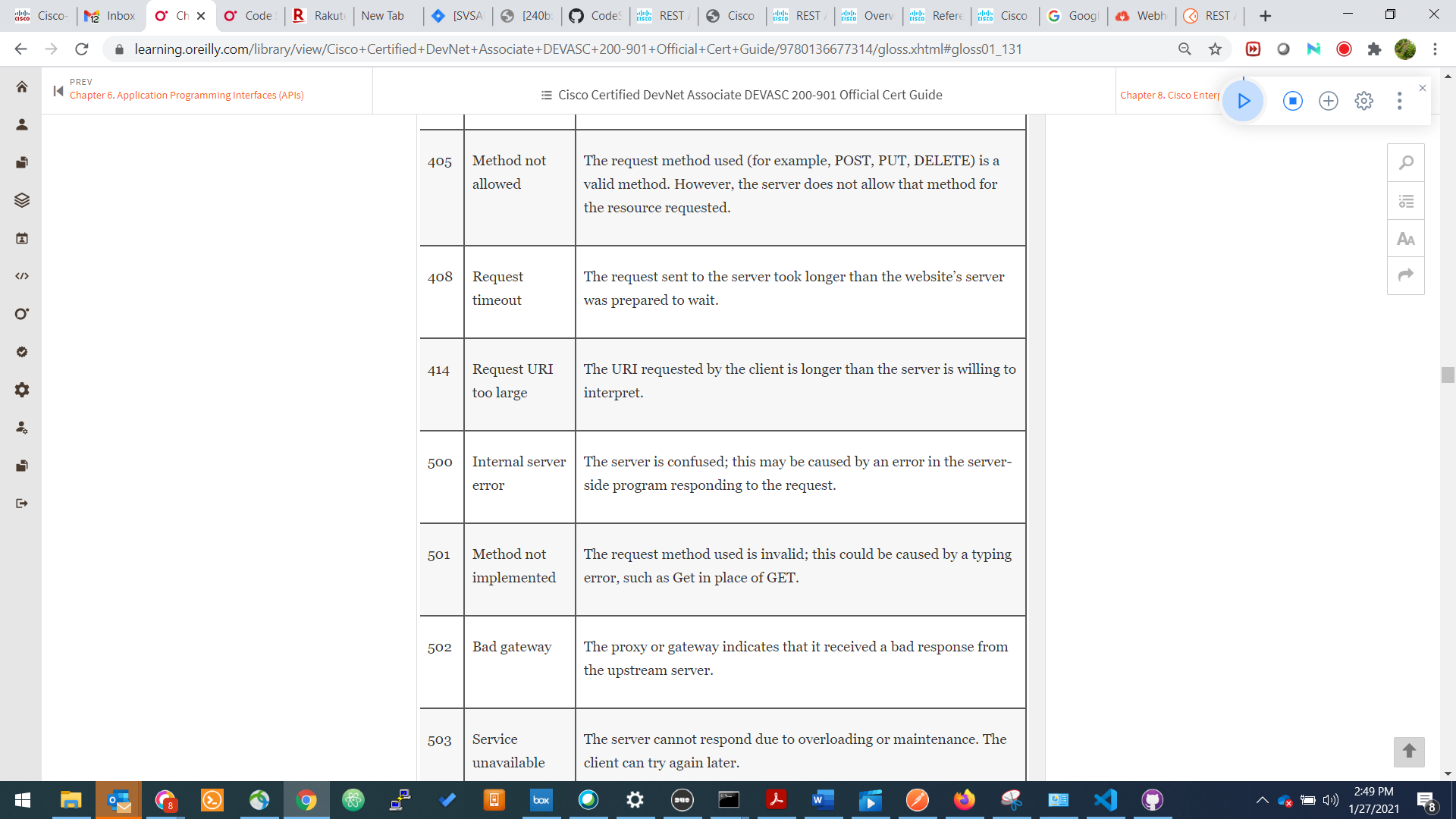


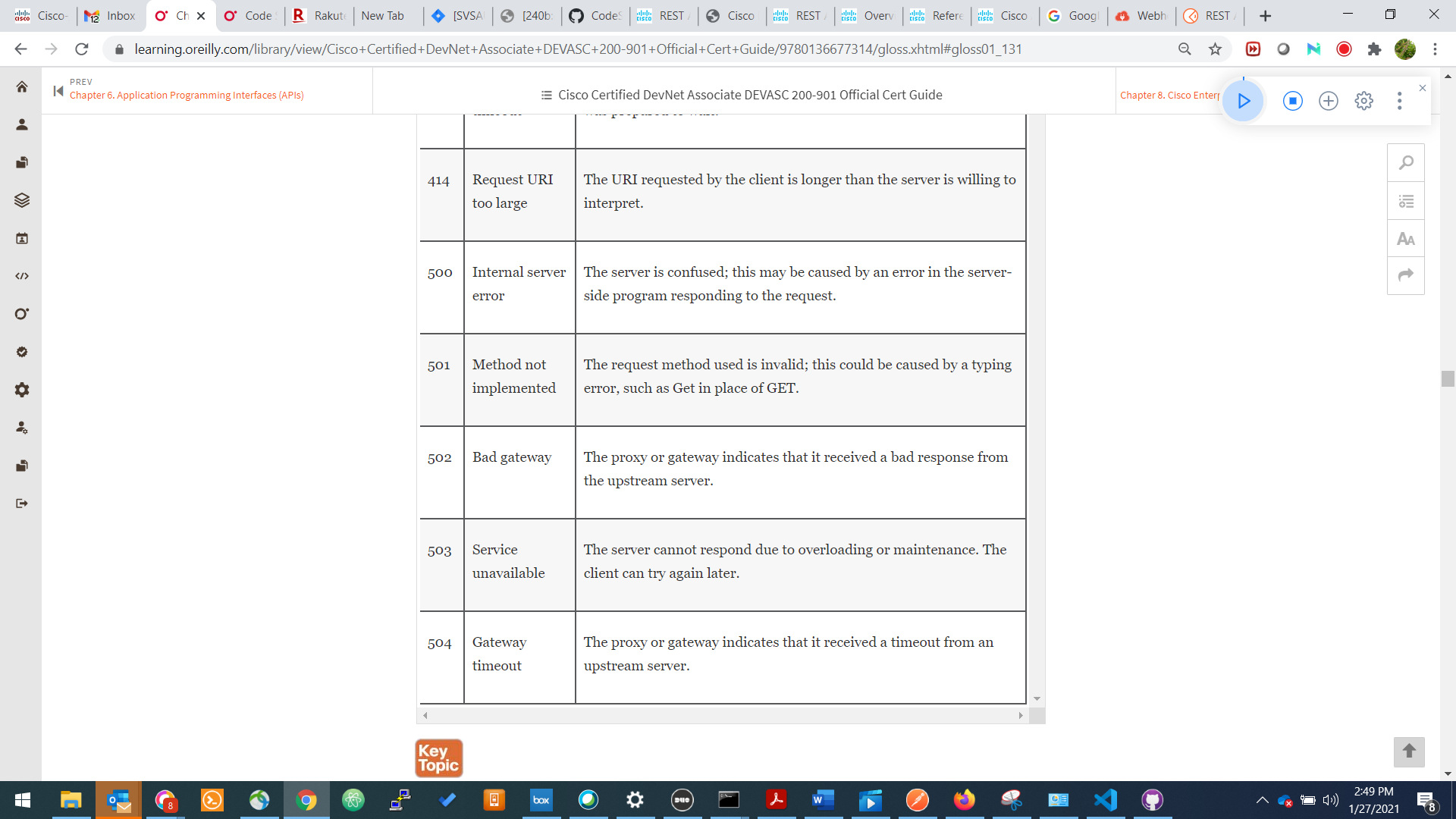
HTTP response status codes indicate whether a specific [HTTP](https://developer.mozilla.org/en-US/docs/Web/HTTP) request has been successfully completed. Responses are grouped in five classes:

1. Informational responses (100–199)
2. Successful responses (200–299)
3. Redirects (300–399)
4. Client errors (400–499)
5. Server errors (500–599)









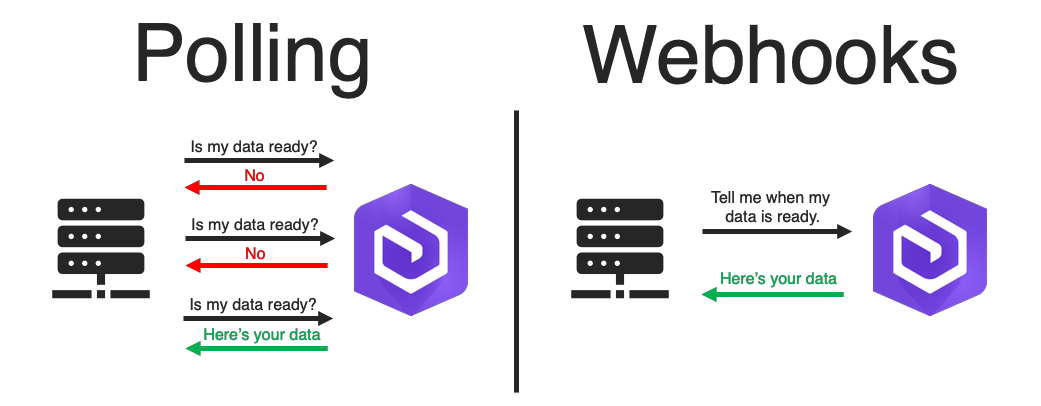
**REST API Versioning**

Versioning is a crucial part of API design. It gives developers the ability to improve an API without breaking the client’s applications when new updates are rolled out. Four strategies are commonly employed with API versioning:

* **URI path versioning:** In this strategy, the version number of the API is included in the URL path.
* **Query parameter versioning:** In this strategy, the version number is sent as a query parameter in the URL.
* **Custom headers:** REST APIs are versioned by providing custom headers with the version number included as an attribute. The main difference between this approach and the two previous ones is that it doesn’t clutter the URI with versioning information.
* **Content negotiation:** This strategy allows you to version a single resource representation instead of versioning an entire API, which means it gives you more granular control over versioning. Another advantage of this approach is that it doesn’t require you to implement URI routing rules, which are introduced by versioning through the URI path.

***Webhooks***

* User defined HTTP Callbacks
* Referred to as reverse API
* Uses Validation token



***Postman Demo***