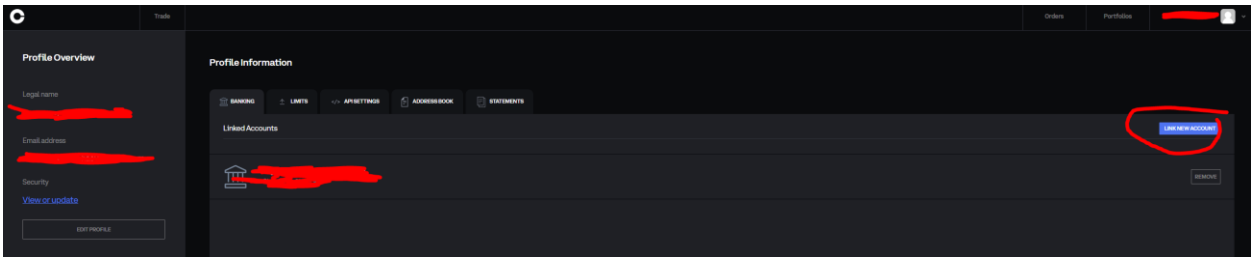
How To Set Up Recurring Buys Using Google Cloud VM and Coinbase Pro

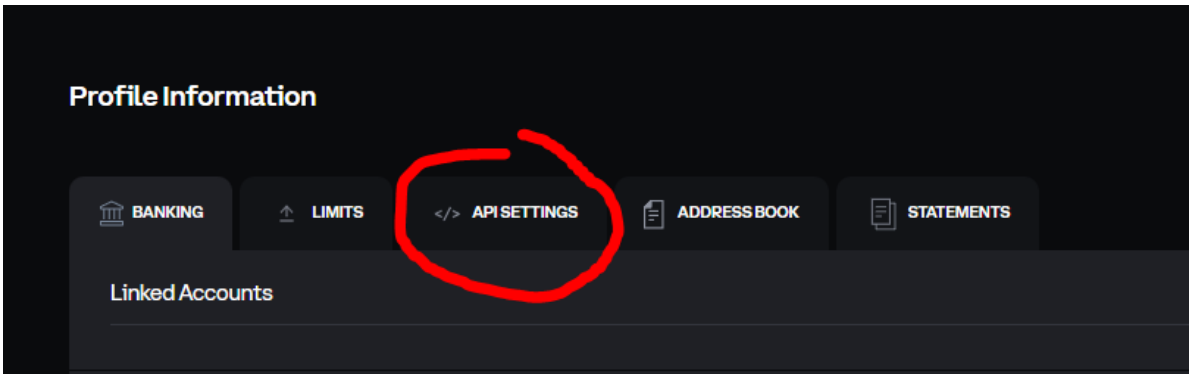
Create a Coinbase Pro account

1. Navigate to https://pro.coinbase.com/
2. In the top right corner click the sign up button
    a. I believe if you are an existing Coinbase customer you can use the sign in button and log in with your normal Coinbase info
3. After you have gone through all of the steps to sign in or to create an account, you should link a bank account if you don't already have one. This can be done by clicking in the top right corner where it says your name, and hitting the "Banking" button

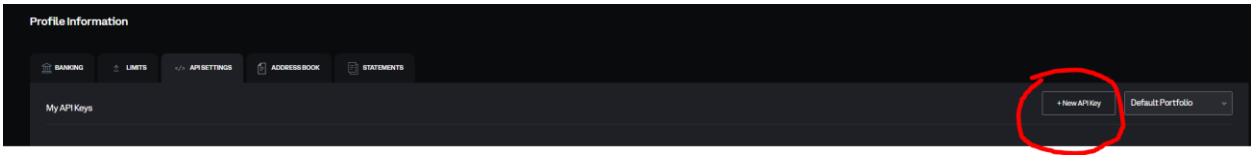
4. When you get to banking page, click the "Link New Account" button and follow through the steps to link your bank account\
    a. Use the "Bank account" option, not the "Wire transfers" option



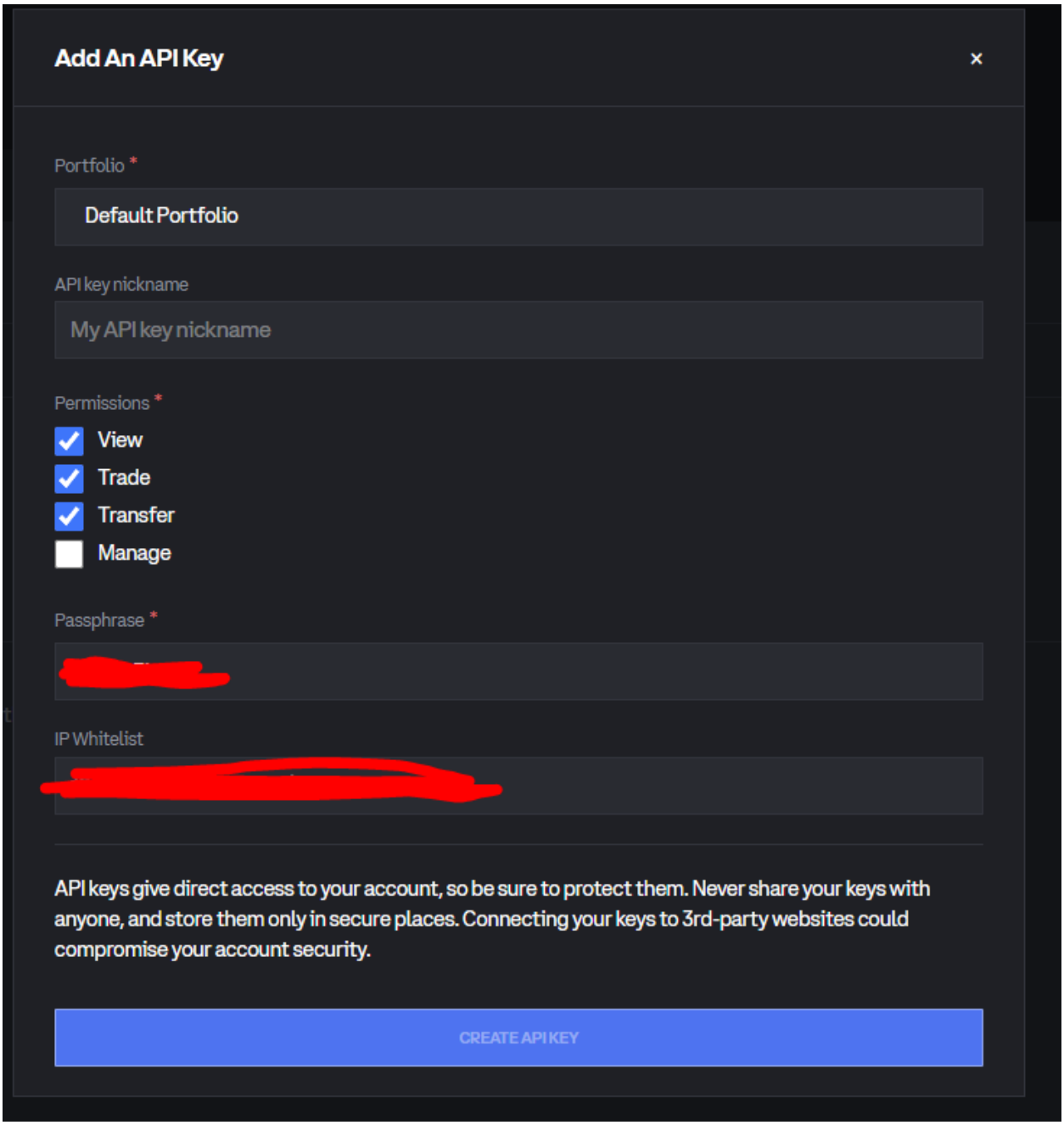
5. When your bank account is attached, click on the "API Settings" tab



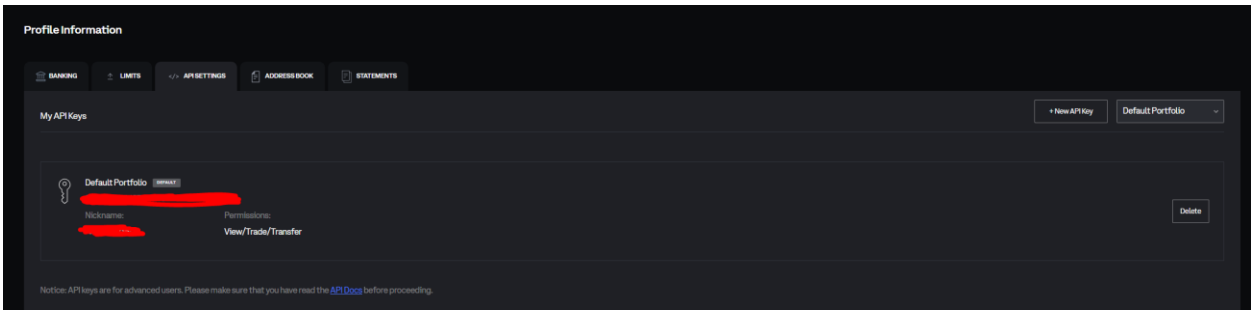
6. Click the “+ New API Key” button”



7. You can set the portfolio name to whatever you’d like, or leave it at the default name, and you can also give the key a nickname but this is not necessary. Make sure you check the boxes next to “View”, “Trade”, and “Transfer”.
    a. Copy and paste the Passphrase somewhere temporarily, because this is the last time you’ll be able to see it and you’ll need it later
    b. Click “Create API Key” at the bottom. You made need to enter your 2FA code after you hit the create button
    c. Copy the “API Secret” somewhere temporarily as well, as you’ll need this later and won’t be able to get it again

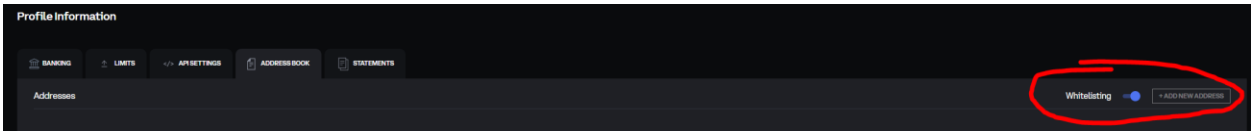


8. If you were successful, you should see your API key details. Do not share these details with anyone. KEEP THE KEY DETAILS PRIVATE

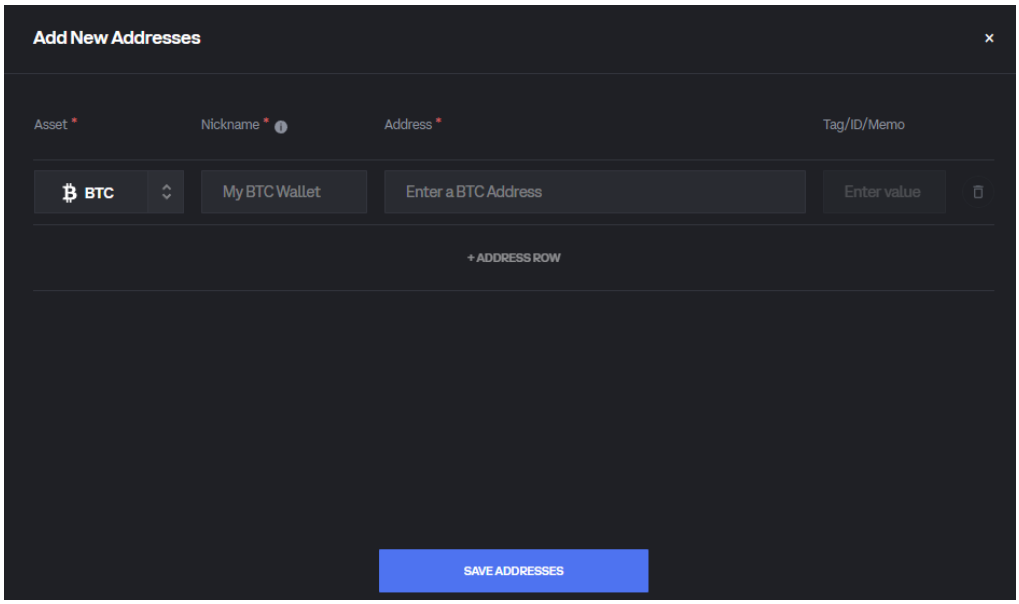


9. Click the “Address Book” tab

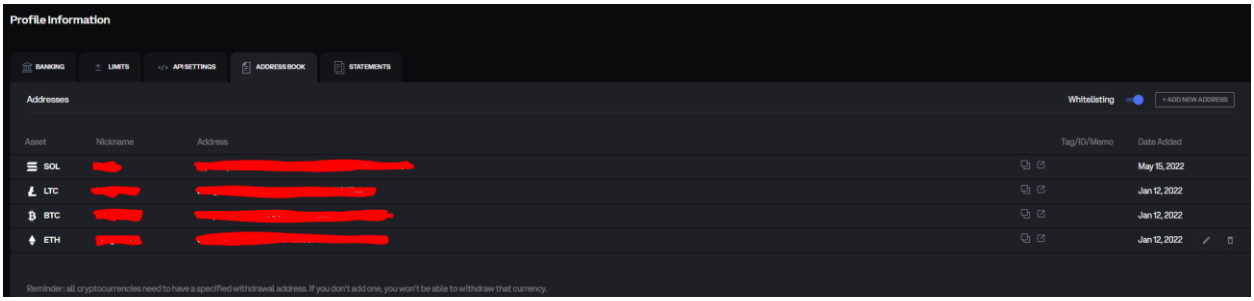
10. Turn on Whitelisting, and click the “+ Add New Address” button
    a. Whitelisting will make it so you can only send crypto to the specified address in your whitelist. This will protect you if someone gets into your coinbase account or into your VM (we’ll get to that later). New addresses require a 48hr seasoning period, so it will make it more difficult for someone to take all your crypto.



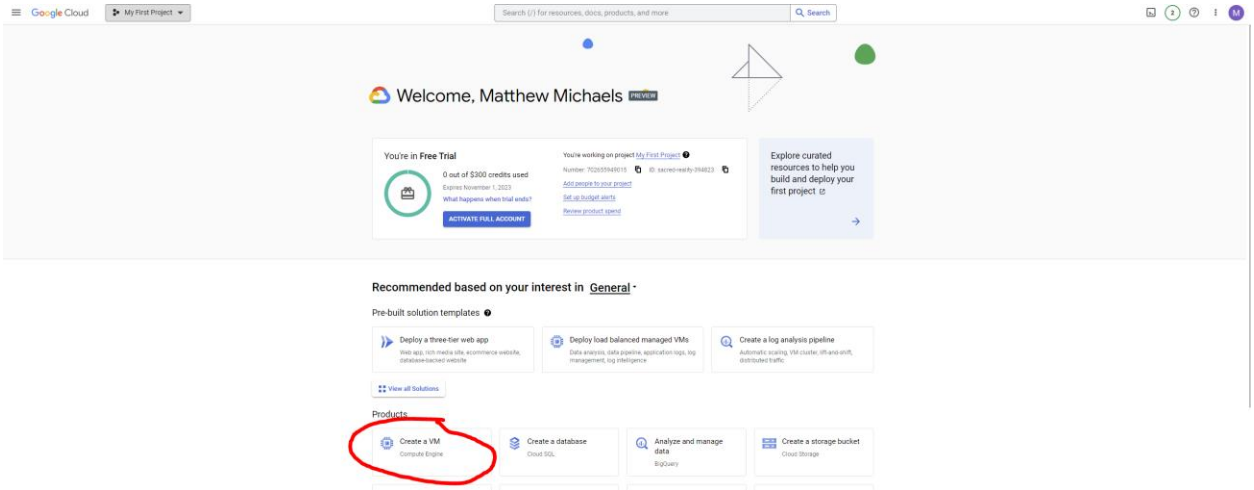
11. Select the asset you want to add an address for, add a nickname, and then put in the receive address for your wallet where you’d like to keep your coins. Repeat this for all of the assets you’d like to set up a recurring buy, and then click the “Save Addresses” button



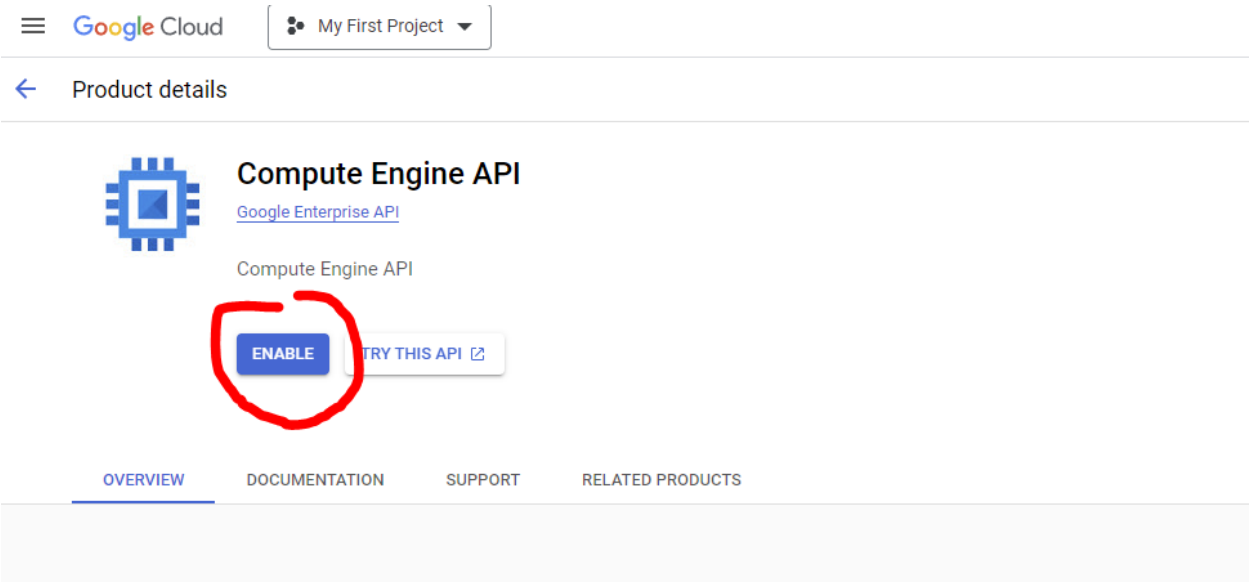
12. When you finish adding your addresses you should see a list like this, that has your nicknames, shows the assets, and the addresses



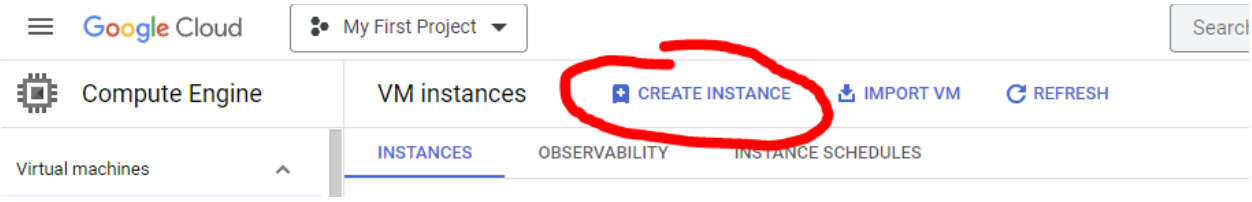
13. Navigate to https://cloud.google.com/compute to begin setting up your VM
14. Click on the "Try Compute Engine Free" button and log in with your google account
    a. Google gives more than enough credits for our purposes that will let you do this for free for 90 days. After that the cost varies based on how often you do your recurring buys. Doing a recurring buy twice a month costs about $1
15. Follow the steps to create your compute account
    a. You will have to enter a credit or debit card here, but you will not be charged at the end of your trial unless you manually upgrade your account
16. When you finish setting up your account, you'll come to the follow screen. Click on the "Create a VM" button



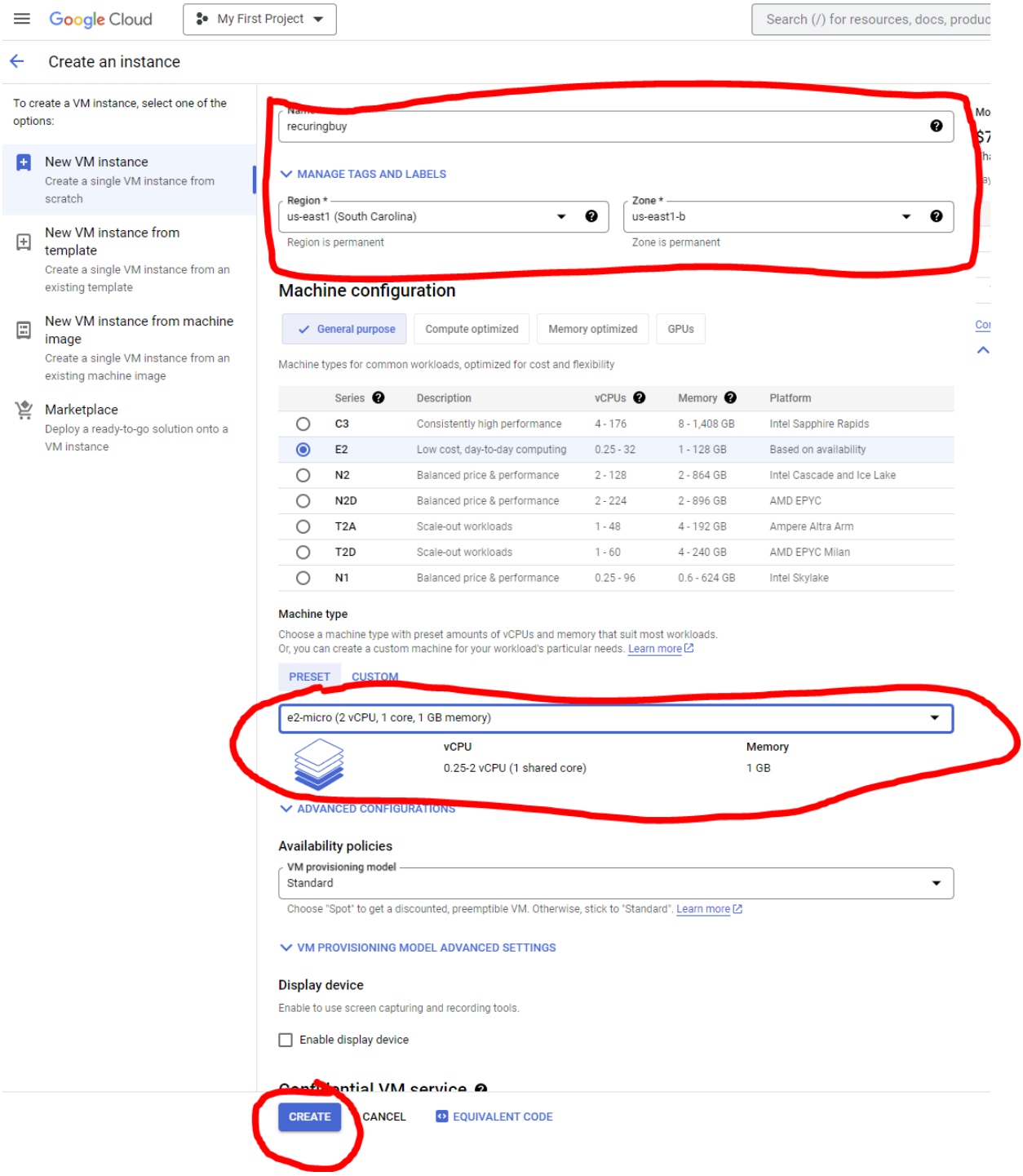
17. Click the "Enable" button if this pops up for you
    a. If you have created a VM with google before you may not see this page
    b. After you click "Enable" it may take a few minutes before it finishes



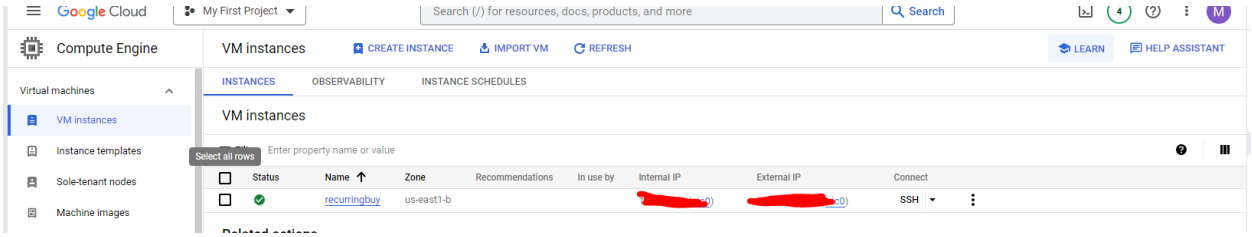
18. Click the “Create Instance” button



19. On the next screen, set your name, set your region to the closest area to where you live, leave zone at whatever it selects, and change your machine type to e2-micro. When you have your settings correct,
    a. You can choose a more powerful machine if you’d like, but it will cost more money and for our purposes it isn’t needed

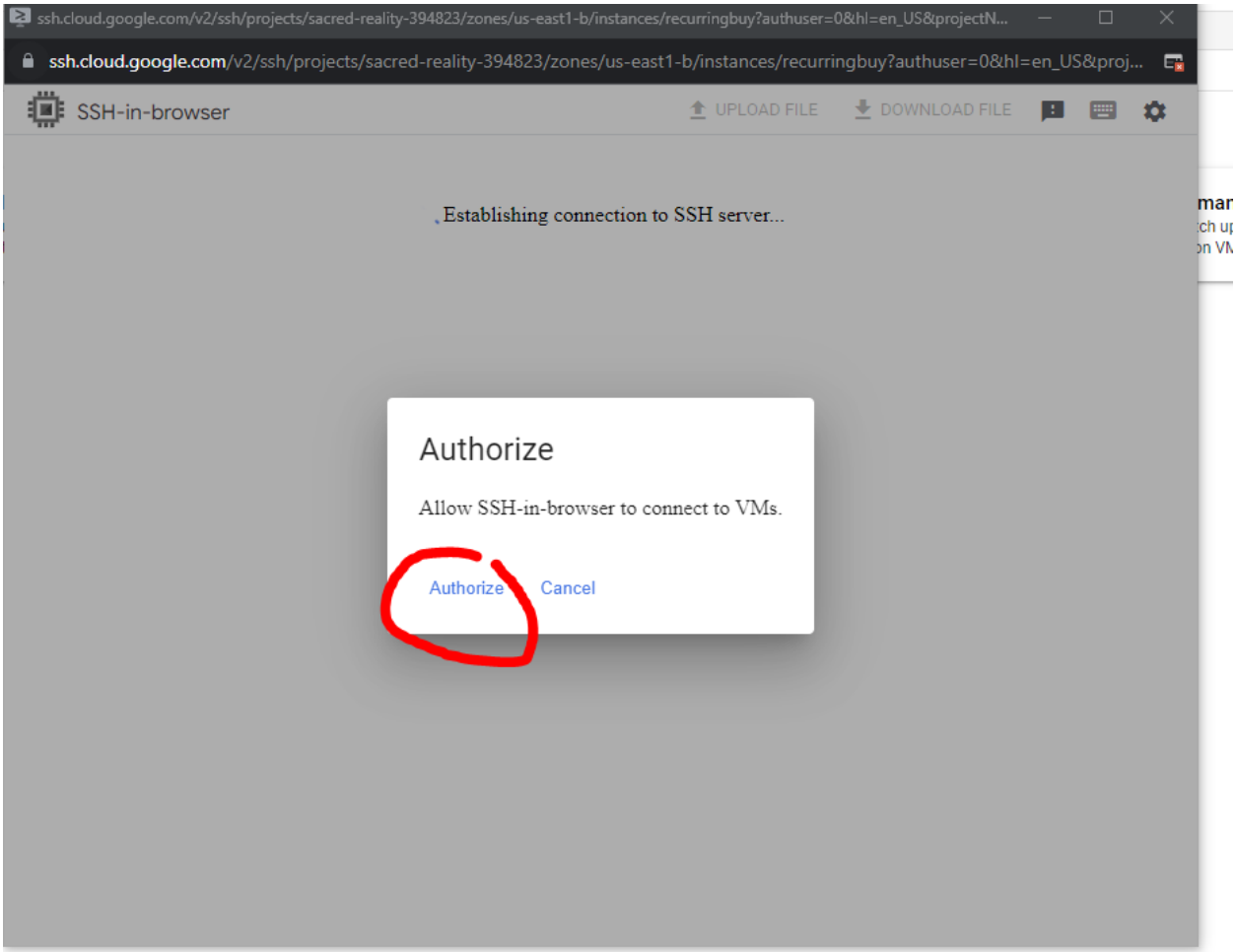


20. After you hit “Create” you should be brought to this screen where you can see your VM that you just created. Leave this page open

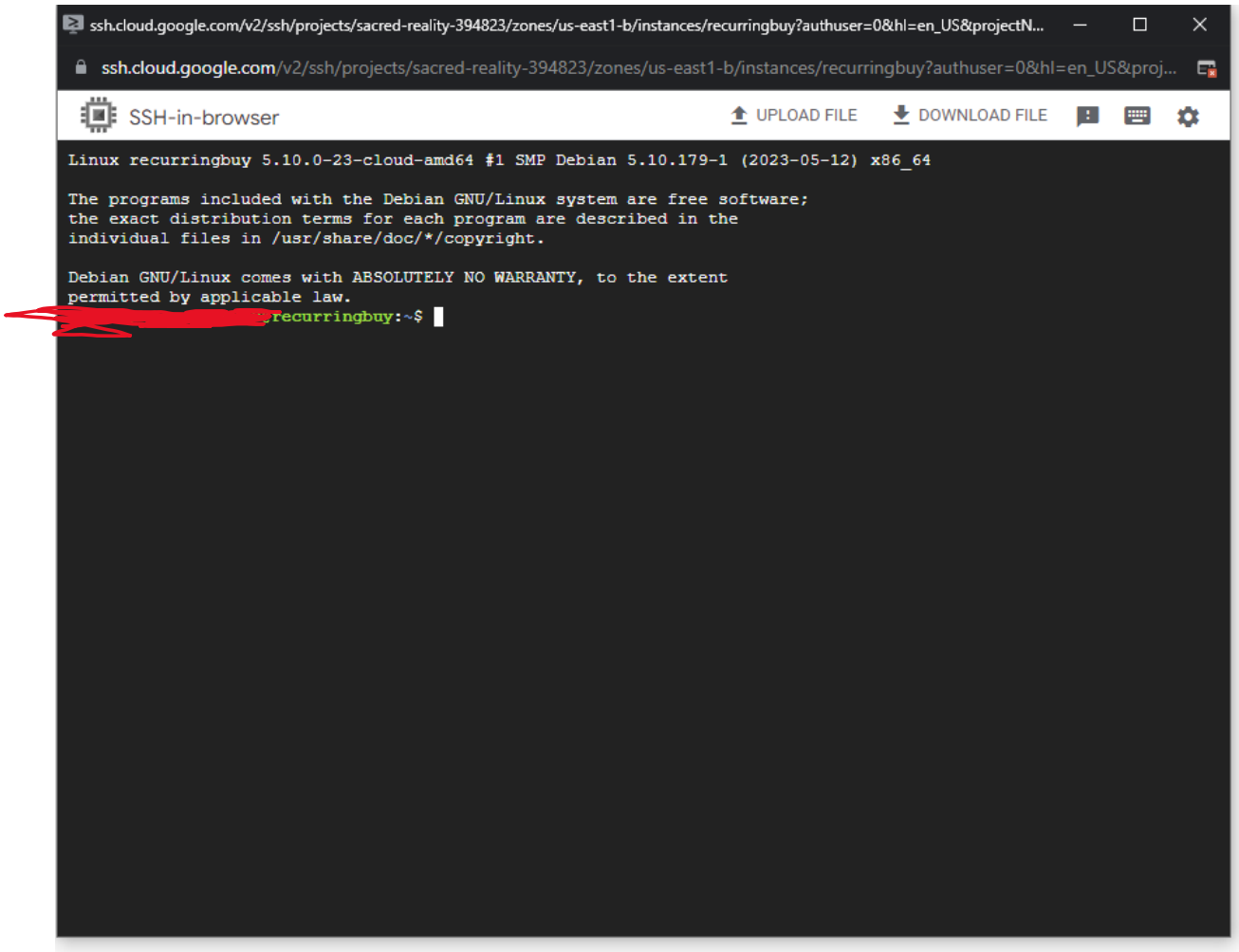


21. Click on the “SSH” button, and a window should pop up

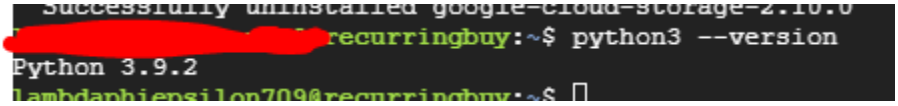
22. If you are prompted to authorize in browser SSH, click the “Authorize” button



23. If everything is successful you should have a terminal open that is connected to your VM



24. Paste the following into your terminal and hit enter:
    a. sudo apt update
       sudo apt install python3 python3-dev python3-venv
    b. Say yes to any prompts that may appear during the install
25. Paste the following into your terminal and hit enter:
    a. sudo apt-get install wget
       wget https://bootstrap.pypa.io/get-pip.py
       sudo python3 get-pip.py
    b. Say yes to any prompts that may appear during the install
26. When the installs complete you type the command “python3 –version” into your terminal and hit enter. If python was successfully installed you should see some version number appear



27. Paste the following into your terminal and hit enter:
    a. pip3 install cbpro

28. Leave your terminal open, and navigate to https://drive.google.com/drive/folders/1_nsutLQjL0BPQPuze_1vq8Q7kFrdHeOl?usp=sharing, and download the three python files located in that google drive folder
    a. The files are called bankTransferGeneric.py and coinBuyGeneric.py, and coinTransferGeneric.py
29. When the files are downloaded on to your PC, open them in your text editor of choice. They should look like this:

```python
import cbpro
from cbpro.authenticated_client import AuthenticatedClient

api_key = ""
api_secret = ""
api_passphrase = ""

class bankTransfer:

    client = AuthenticatedClient("","","")

    def __init__(self,key,secret,passphrase) -> cbpro.AuthenticatedClient:
        self.client =  cbpro.AuthenticatedClient(key,secret,passphrase)

    def getBankPaymentId(self):
        method_id = ""
        paymentMethods = self.client.get_payment_methods()
        for method in paymentMethods:
            type = method.get('type',None)
            currency = method.get('currency',None)
            if currency == 'USD' and type == 'ach_bank_account':
                method_id = method.get('id',None)
                break
        return method_id

    def deposit(self, amount, currency, method):
        self.client.deposit(amount,currency,method)

if __name__ == "__main__":

    RB = bankTransfer(api_key,api_secret,api_passphrase)
    bankID = RB.getBankPaymentId()
    RB.deposit(300,'USD',bankID)

    print("Deposited 300 into coinbase pro account")
```

```python
import cbpro
from cbpro.authenticated_client import AuthenticatedClient

api_key = ""
api_secret = ""
api_passphrase = ""

class coinBuy:

    client = AuthenticatedClient("","","")

    def __init__(self,key,secret,passphrase) -> cbpro.AuthenticatedClient:
        self.client =  cbpro.AuthenticatedClient(key,secret,passphrase)

    def buy(self, pair, quantity):
        response = self.client.place_market_order(product_id=pair,side='buy',funds=quantity)

    def withdrawCrypto(self,amount,coin,address):
        amount = round(float(amount),8)
        response = self.client.crypto_withdraw(amount,coin,address)
        print("{}: {}".format(coin,response))

    def getAvailableBalance(self,coin):

        available = ""

        accounts = self.client.get_accounts()
        for account in accounts:
            currency = account.get('currency',None)
            if currency == coin:
                available = account.get('available',None)
                break

        return available

if __name__ == "__main__":

    RB = coinBuy(api_key,api_secret,api_passphrase)

    #Buy BTC
    RB.buy('BTC-USD',50)
    print("Bought BTC")

    #Buy ETH
    RB.buy('ETH-USD',50)
    print("Bought ETH")

    #Buy LTC
    RB.buy('LTC-USD',50)
```

```python
import cbpro
from cbpro.authenticated_client import AuthenticatedClient

api_key = ""
api_secret = ""
api_passphrase = ""

class coinBuy:

    client = AuthenticatedClient("","","")

    def __init__(self,key,secret,passphrase) -> cbpro.AuthenticatedClient:
        self.client =  cbpro.AuthenticatedClient(key,secret,passphrase)

    def buy(self, pair, quantity):
        response = self.client.place_market_order(product_id=pair,side='buy',funds=quantity)

    def withdrawCrypto(self,amount,coin,address):
        amount = round(float(amount),8)
        response = self.client.crypto_withdraw(amount,coin,address)
        print("{}: {}".format(coin,response))

    def getAvailableBalance(self,coin):

        available = ""

        accounts = self.client.get_accounts()
        for account in accounts:
            currency = account.get('currency',None)
            if currency == coin:
                available = account.get('available',None)
                break

        return available

if __name__ == "__main__":

    BTC_address = ''
    ETH_address = ''
    LTC_address = ''

    RB = coinBuy(api_key,api_secret,api_passphrase)

    #Transfer BTC
    balance = RB.getAvailableBalance('BTC')
    print("BTC balance: " + balance)
    RB.withdrawCrypto(balance,'BTC',BTC_address)
    print("Transfered BTC")
```

30. If you don't still have the webpage open, navigate back to your Coinbase Pro account, and open up the API tab
31. Above your nickname of your API key will be a string of numbers and letter; this is your API key. If you click on this it will be copied to your keyboard

32. Copy your API key and paste it in between the two paratheses at the top of each file on the line that says api_key
33. Repeat step 32, but paste in your API secret and API passphrase that you saved off earlier and paste those in their appropriate spots in each document
34. When you're finished it should look something like this in each document
    a. Yours will have your actual strings of numbers and letters instead of the gibberish in the screenshot

```
api_key = "123456asda123"
api_secret = "12sdd51ew"
api_passphrase = "763244ad13"
```

35. Once you have successfully added your key, secret, and passphrase to each file, navigate to bankTransferGeneric.py
36. At the bottom of the document, look for the line that says "RB.deposit(300,'USD',bankID)"
    a. This line controls how much money gets deposited into your coinbase pro account each time this file is run
37. Change the "300" to whatever dollar amount you want deposited into your account each time this is run
    a. I'd suggest setting it to the total dollar amount you want to invest each month
38. Open up coinTransferGeneric.py, and scroll down to line 38 where it has a blank spot for a bitcoin address

```
38      BTC_address = ''
39      ETH_address = ''
40      LTC_address = ''
41
```

39. Here you want to fill in your wallet receive address in between the apostrophes if want to do recurring buys of BTC, ETH, or LTC
    a. If you want to buy a crypto not listed here, copy paste one of these lines, and change the name to fit whatever crypto you want to trade. For example, XRP_address, BNB_address, etc.
40. Once you have a receive address set up for each crypto you'd like to trade scroll down to line 45. Lines 45-48 are what transfer the crypto you bought, in this case Bitcoin, to the wallet address you provided
    a. Lines 51-54 and 57-60 would transfer your ETH or LTC if you had any. You can delete or modify these lines as you see fit

```
45      balance = RB.getAvailableBalance('BTC')
46      print("BTC balance: " + balance)
47      RB.withdrawCrypto(balance,'BTC',BTC_address)
48      print("Transfered BTC")
49
```

41. If you are trading a crypto that is not already set up, you can copy paste lines 45-48 (or one of the other existing blocks for buying and transferring and set it up for your crypto
    a. Here is an example of what trading XRP would look like

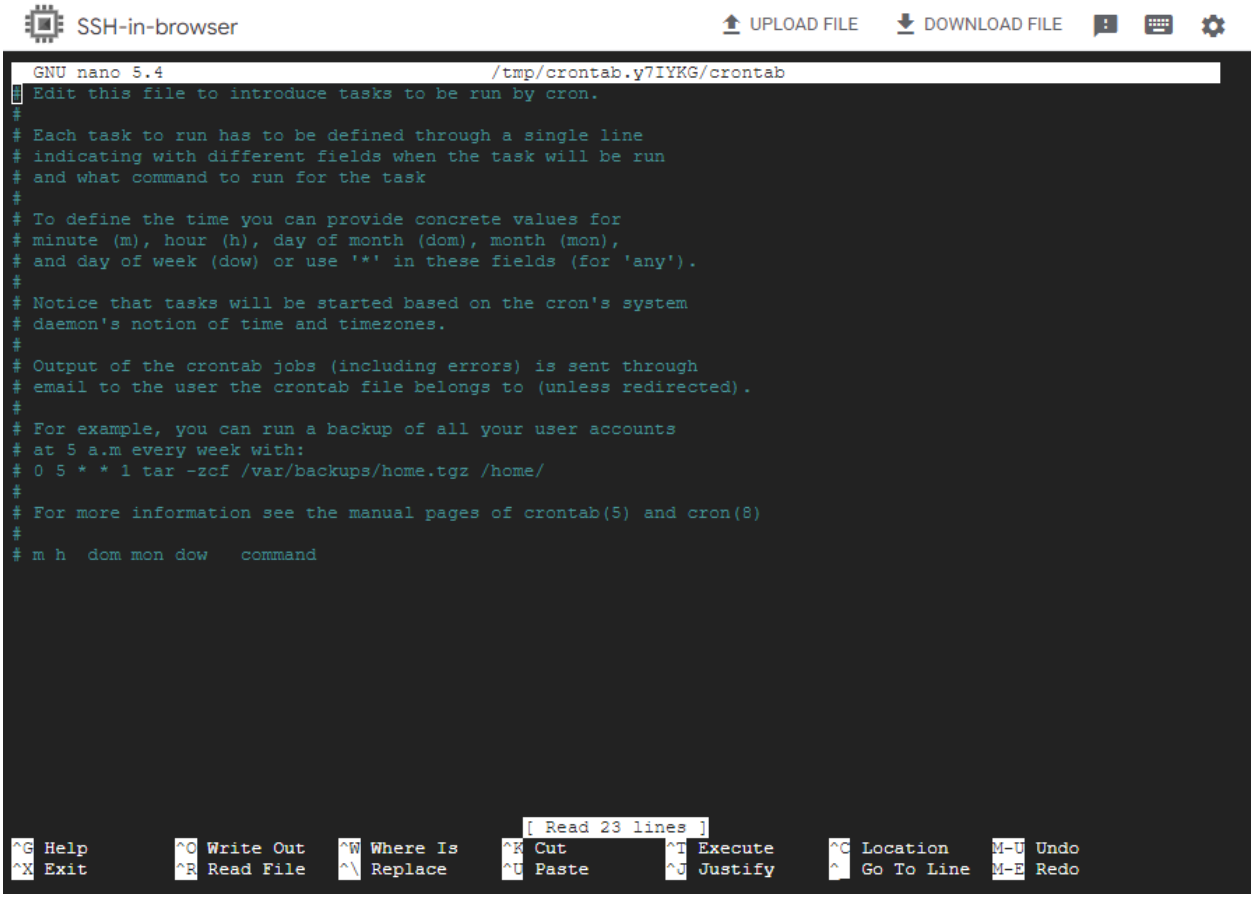
```
#Transfer XRP
balance = RB.getAvailableBalance('XRP')
print("XRP balance: " + balance)
RB.withdrawCrypto(balance,'XRP',XRP_address)
print("Transfered XRP")
```

42. Once you have all of the cryptos set up that you plan on buying, navigate to the coinBuyGeneric.py file
43. Scroll down to line 41. Here you can see a buy of Bitcoin. You can update this line or the others ones that buy crypto similarly how you modified them in coinTransferGeneric.py to be set up to purchase the crypto of your choice
    a. The "50" on line 41 controls the dollar amount of the purchase that will occur each time this script is run. Make sure to update this value to reflect the size of the purchase of each crypto that you would like to make every time the script is ran
44. Once you have finished updating all of the files, make sure you save them all
45. Navigate back to your SSH terminal, and click the "Upload File" button

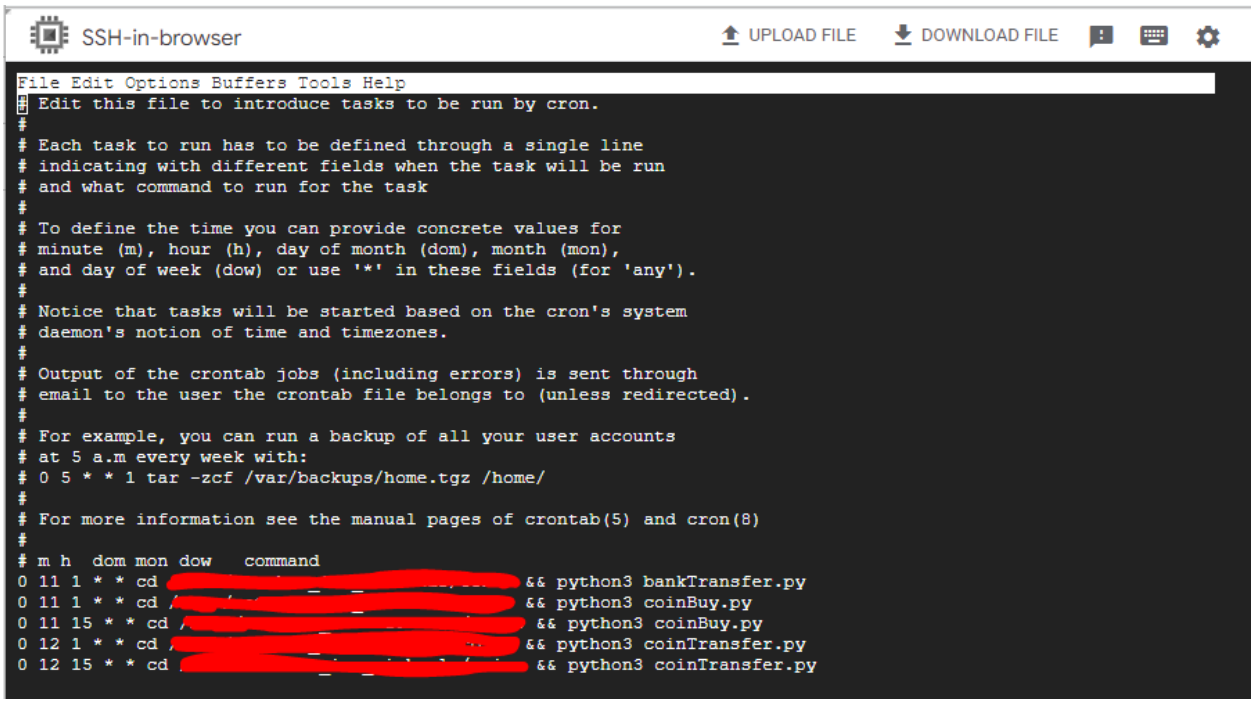
46. Select the files you just modified, and click the open button, and then the upload files button
    a. If it has been a bit since you accessed the terminal, it may give you an error. If this is the case you may need to close and reopen your SSH terminal winow
47. Now that your files are uploaded, we are going to set up our VM to automatically run these scripts on a set schedule using cron jobs
    a. I have mine set up to run at 6am, and 7am EST on the 1st and 15th of every month, and I will walk you how to set it up the same way
    b. If you are interested in a different schedule this is a great link that talks about cron jobs and should allow you to configure it how you'd like. https://phoenixnap.com/kb/set-up-cron-job-linux
48. Type the following command into your terminal, and then hit enter:
    a. Pwd

b. Copy and paste the result of that command, that will be necessary for a following step

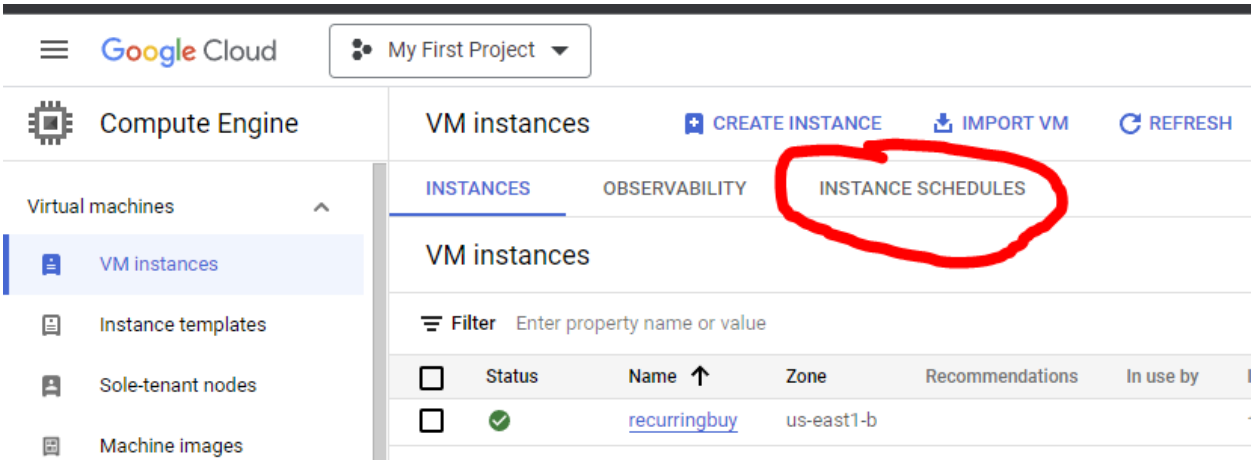
49. Type the following command into your terminal, and then hit enter:
    a. Crontab -e
    b. If this is the first time setting up a cron job you'll get a prompt asking which editor you'd like to use. If you get this, chose option 1
50. If the command worked, you should get to the following screen



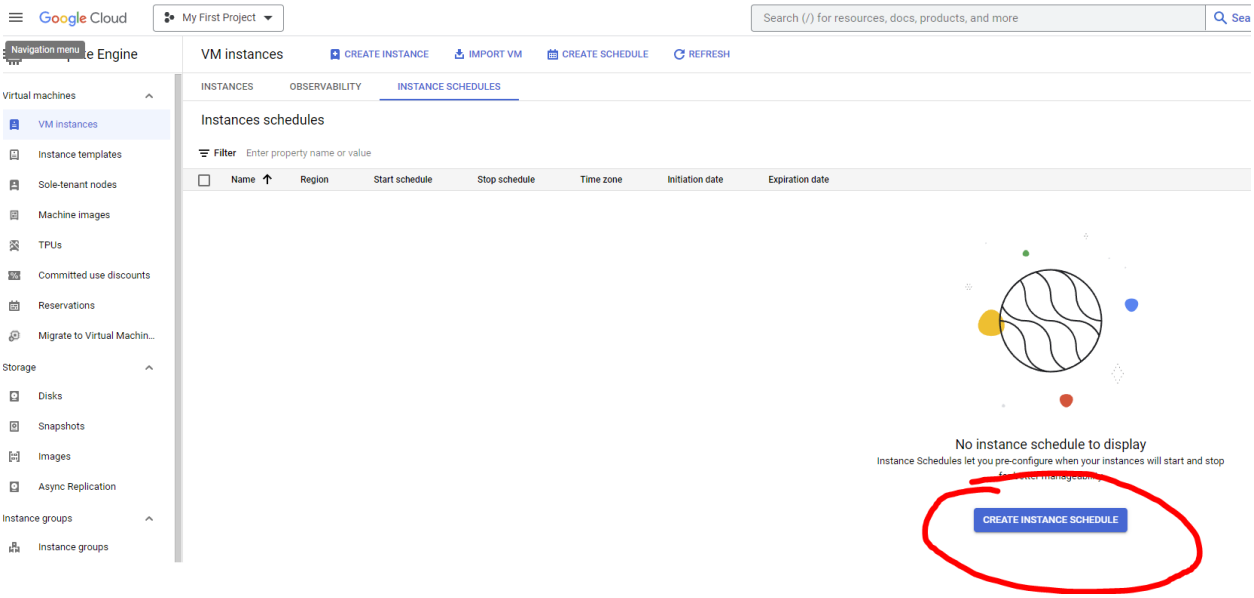
51. Use the arrow key to scroll down as far as possible
52. Look at the below screenshot and type in the same information into your window to make it match the screenshot
    a. My font color is different because I used a different editor
    b. Where the red lines are is where you will put the output of the "pwd" command that you ran earlier
    c. My files are called bankTransfer.py, coinBuy.py, and coinTransfer.py, but you should put whatever your name is. If you haven't renamed the files, they'll be bankTransferGeneric.py, coinBuyGeneric.py, and coinTransferGeneric.py.



53. If you follow this exactly, your scripts will run as follows
    a. bankTransfer.py will run on 1st of every month at 6am EST
        i. It takes 7 days for the money to settle, so this money that is deposited is really for the following month
        ii. You may need to transfer in the first month of money and wait a week before initiating all of these scripts or else your first groups of purchases may fail
    b. coinBuy.py will run on the 1st and 15th of every month at 6am
    c. coinTransfer.py will run on the 1st and 15th of every month at 7am
        i. This script runs an hour after coinBuy.py so that the purchase order has time to go through. If you run them back to back sometimes your transfer will not work, and the crypto will sit in your coinbase account until the next time the script is ran
54. When you have added all of the commands into your cron job file, exit the editor
    a. If you used the editor I suggested, press control + x
55. You can now exit the SSH terminal
56. Navigate back to your compute engine window that shows your VM instances, and click the “Instance Schedules” button at the top



57. Click the “Create Instance Schedule” button



58. Give your schedule a name, add a description, set the region to the same region as your VM, set a start time of 5am, and a stop time of 8am, set the time zone, to the time zone of you and your machine, and set the frequency to monthly and the days to the 1st and the 15th
    a. If you chose to do your recurring buys on a different schedule, adjust the frequency accordingly
    b. You can probably move the hours in closer to the actual script start times, abut I chose to give a lot of time between my machine starting and stopping and when my scripts run

Create a new schedule

Use CRON expression

Name *
on-off-schedule

Name is permanent

Description
turns the machine on at 5am and off at 8am

Region *
us-east1 (South Carolina)

Start time
05:00 AM

Stop time
08:00 AM

Time zone
Eastern Daylight Time (EDT)

Initiate date EST

If left empty, schedule will take effect immediately

End date EST

If left empty, schedule will run indefinitely until it is deleted.

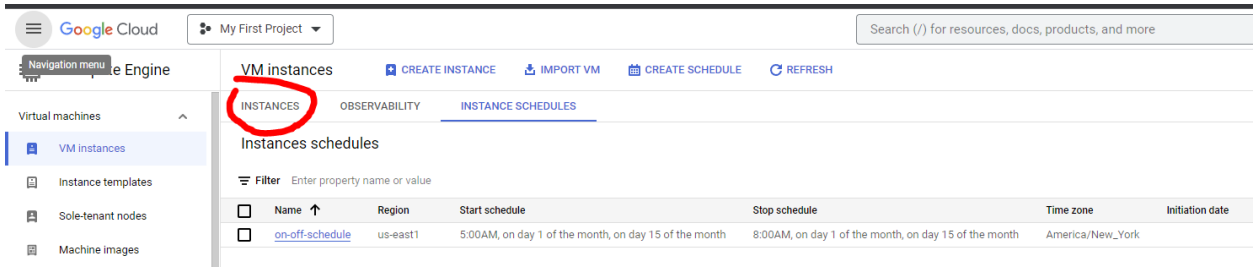
Frequency *
Repeat monthly

Day(s) of the month
1 and 15

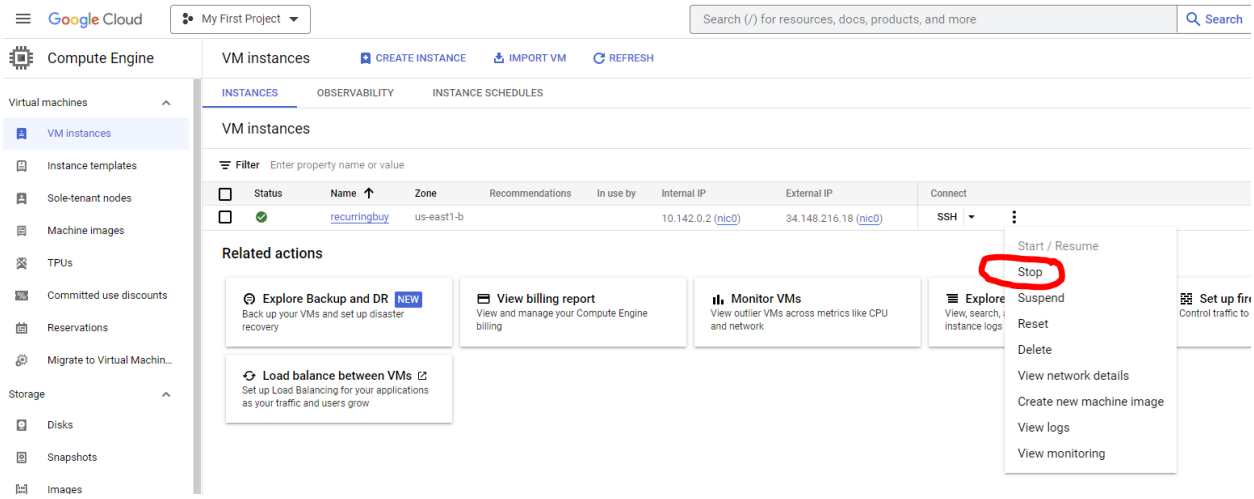
You cannot edit a schedule after it is created

SUBMIT CANCEL

59. When the schedule is set how you'd like, click "Submit". You should now see your schedule
60. Navigate back to the "Instances" tab



61. Click the 3 dots, and select "stop". This will turn off your VM.
    a. The instance schedule that you set up will now automatically turn on and off the machine for you to save you money. The more time your machine is on, the more it will cost you to run.



62. Sit back, relax, and forget about crypto. These scripts will automatically run and will buy, and transfer crypto to your wallet for you

If you'd like to give me a tip as a thank you for this guide, a few of my crypto addresses are below. If you don't want to, that's perfectly fine too 😊. I hope this serves you well

BTC: bc1qxaxx6ds52p3q8jn8qg08hs985adp0t4h0vq97z
LTC: 0xC3A0bF234225cf576772cd614616e207E81D7032
ETH: ltc1qpt73v9l3pln3cntmgaetql3rcfrq9l9lf5ykr3