Homework 3: Bitcoin Data Ingestion

1. Use docker to both build and run the bitcoind binary:
    a. Write a Dockerfile that builds the bitcoind binary.
    b. Use docker-compose to start up the bitcoind binary on your machine.
    c. Specify volume mount points in your docker-compose file so that the bitcoind downloaded data is being stored on your host machine rather than inside the guest container, which enables the bitcoind instance inside the container to start where it left off on its next run.
        i. The bitcoind binary has a way to specify a data directory for the download.
2. Run a full data sync of the bitcoind server on a modal.
    a. Use a modal volume to store the bitcoin data so that whenever you boot up the bitcoind server, the sync resumes where it left off.
        i. The bitcoind server should sync the entire chain from scratch and then continue running to add more blocks as they arrive.
        ii. Use the bitcoin-cli to verify that new blocks are being added every few months.
    b. This step will take up to 4 days to fully sync.
    c. Be sure to run your modal script in a detached form so that disconnections of your laptop to the modal infrastructure do not terminate the job.
        i. See the `--detach` flag on the `modal run` command:
            1. https://modal.com/docs/reference/cli/run
    d. As the sync is happening over a few days, be sure to periodically check it to make sure it's making progress.
        i. One method:
            1. You can use the `modal volume` command on the data directory to see if new data is in fact by added every few minutes:
            2. https://modal.com/docs/guide/volumes
        ii. Another method:
            1. Use getblock RPC call to find out what's the last block downloaded.
            2. There may be other calls to get the last downloaded block directly:
                a. https://developer.bitcoin.org/reference/rpc/index.html
                b. Check:
                    i. getblockchaininfo
                    ii. getblockcount
3. Run RPC calls from your laptop to your bitcoind server on modal.
    iii. Security: need to make sure that if RPC bitcoind server is exposed to the public internet, that it is protected with a username/password and https/ssl over the network.

iv. Many ways of doing this:
    1. Read the section on web endpoints:
        a. https://modal.com/docs/guide/webhooks
    2. Alternative approach is to use tunnels:
        a. https://modal.com/docs/guide/tunnels
    3. Write a modal function for example called "`def getblock(num: int)`" that does a local http request (http://127.0.0.1:8332/rpc") to the bitcoind server. In others, for each bitcoind RPC call, write a modal python function that invokes that RPC call to the server running locally.
        a. This method gives you auth and network encryption through the modal CLI out of the box.