Isaiah Narisma

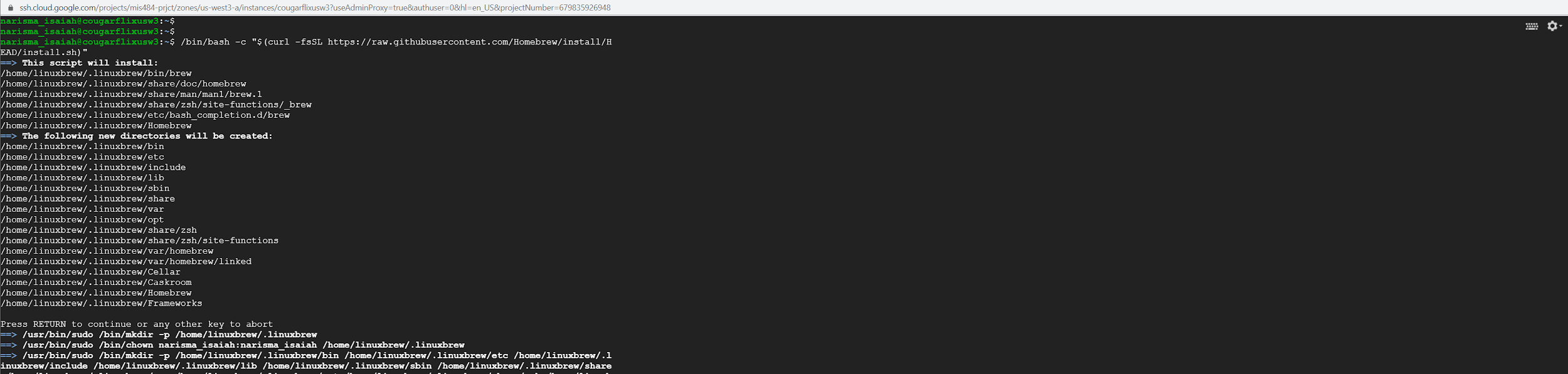
Professor Underhill

MIS 484-6 (01) 42896

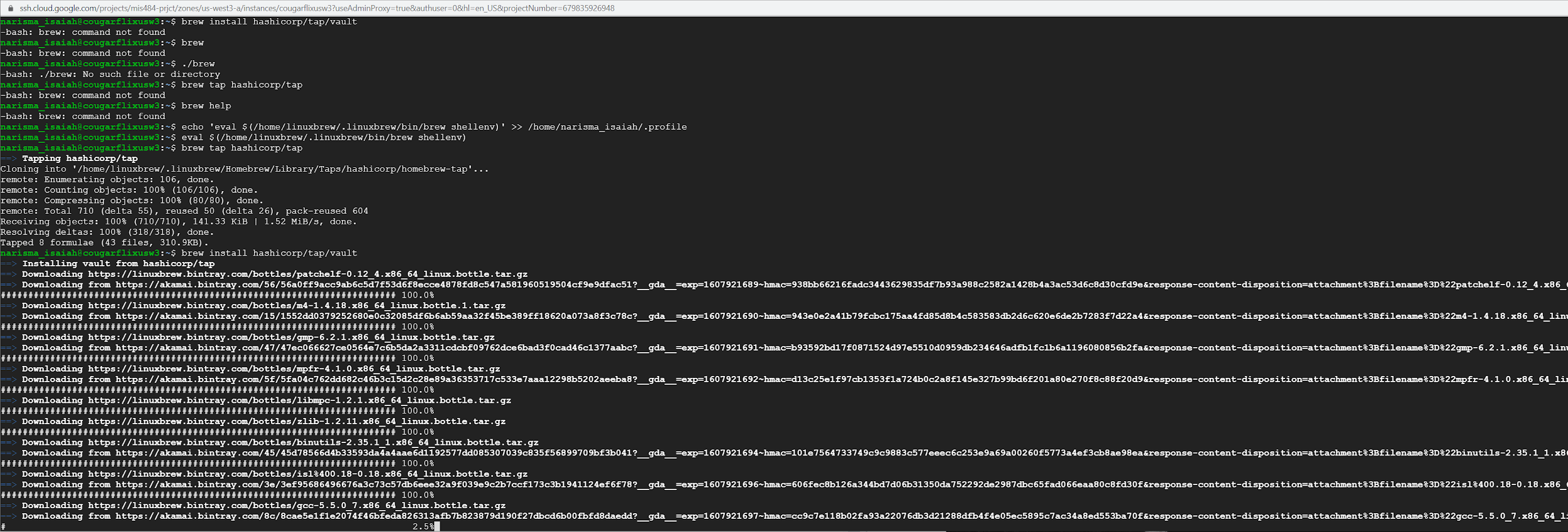
14 December, 2020

Cloud Security Deployment

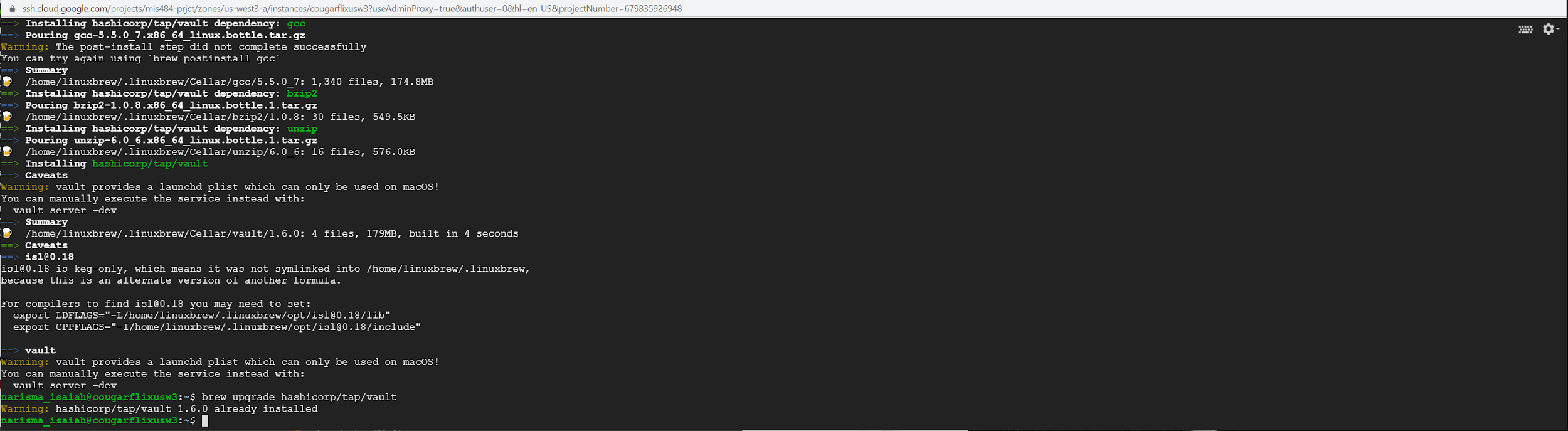
Installing Vault



To install vault, need to install Homebrew for Linux which we can do from the direct web page



Then from here once we have installed Homebrew on our instance, we can proceed to install vault. To install vault properly we can use the same line as before, except change `tap` to `install` and adding `/vault` at the end to access the homebrew formula. So the input would be `brew install hashicorp/tap/vault`.



We should update vault to make sure that it is up to date. We can update vault by running run `brew upgrade hashicorp/tap/vault`

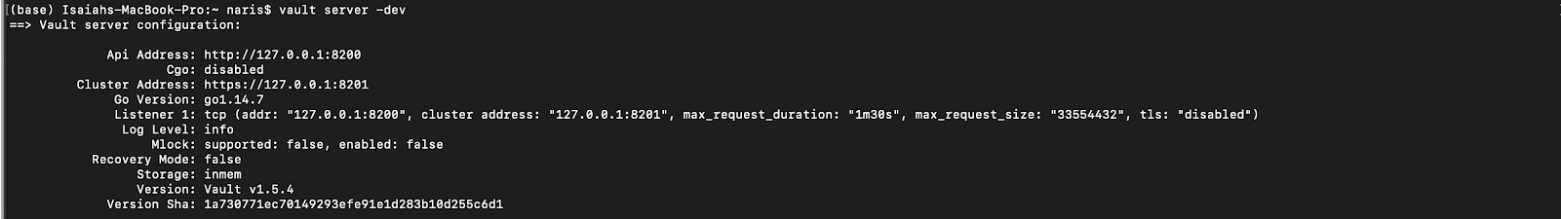


To confirm that we have vault installed, we can enter `vault` and should be returned with a string of commands related to vault that we can use

Now that we have vault successfully installed on our instance, we can begin starting our own server. The vault server is the only piece of the vault architecture that directly interacts with the data storage and back ends. All operations done in the vault server are done via a TLS connection. These `vault dev servers` are inherently not very secure as everything is stored in memory and that they are mostly useful for local development, testing, and exploration.

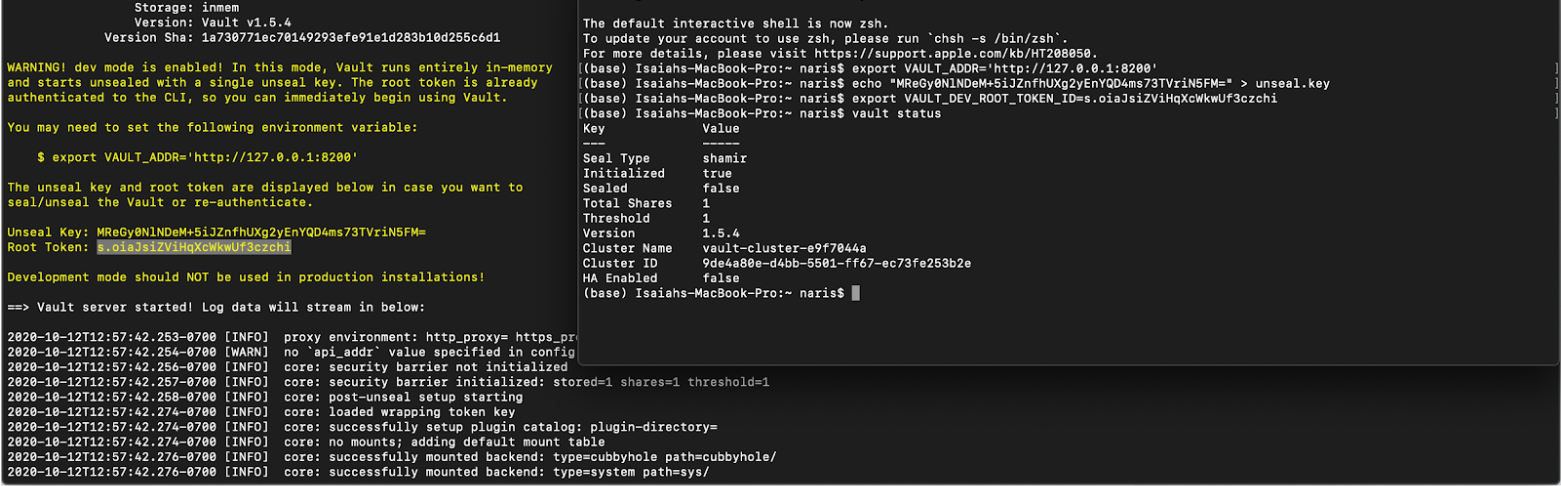
Starting our server

To start our local dev server, in the terminal enter `vault server -dev`.



Here we can see the configuration settings of our dev server where we can see that it listens to the local host address, stores data in memory, as well as shows unsealed keys and root tokens. It is strongly advised that you do not run dev servers in actual production.

Open another terminal window, and copy and paste the export vault address so that our vault client can talk with our dev server.



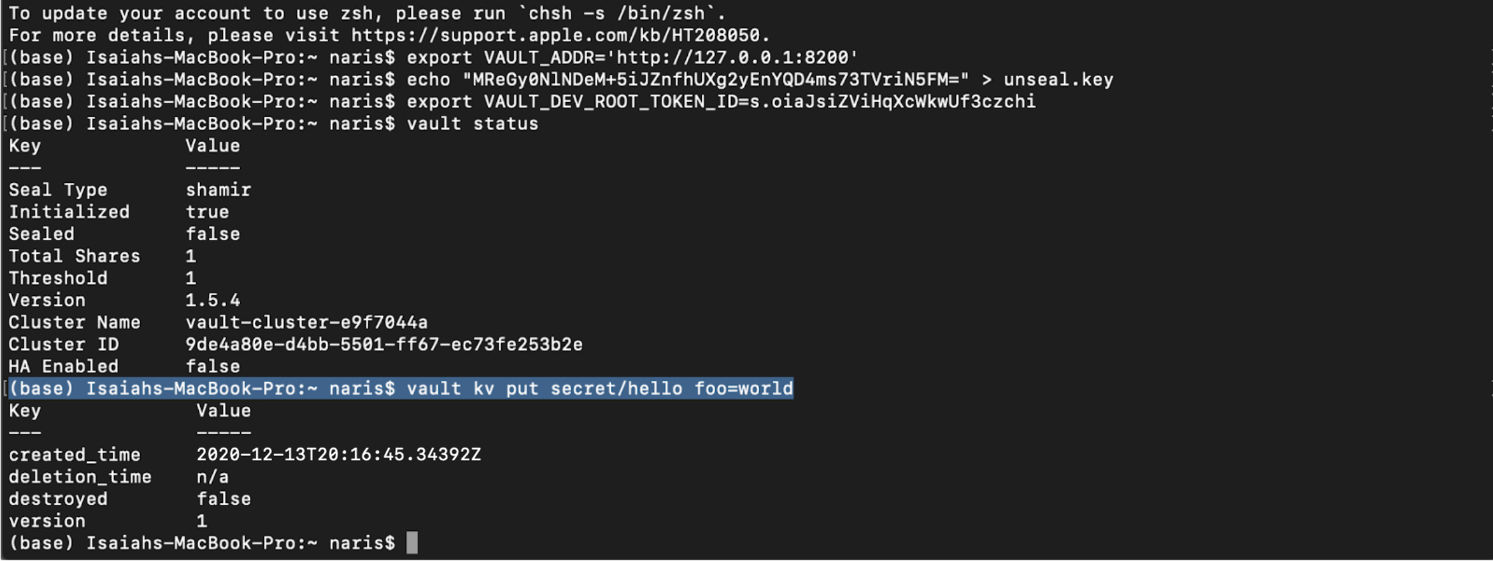
We should follow these steps:

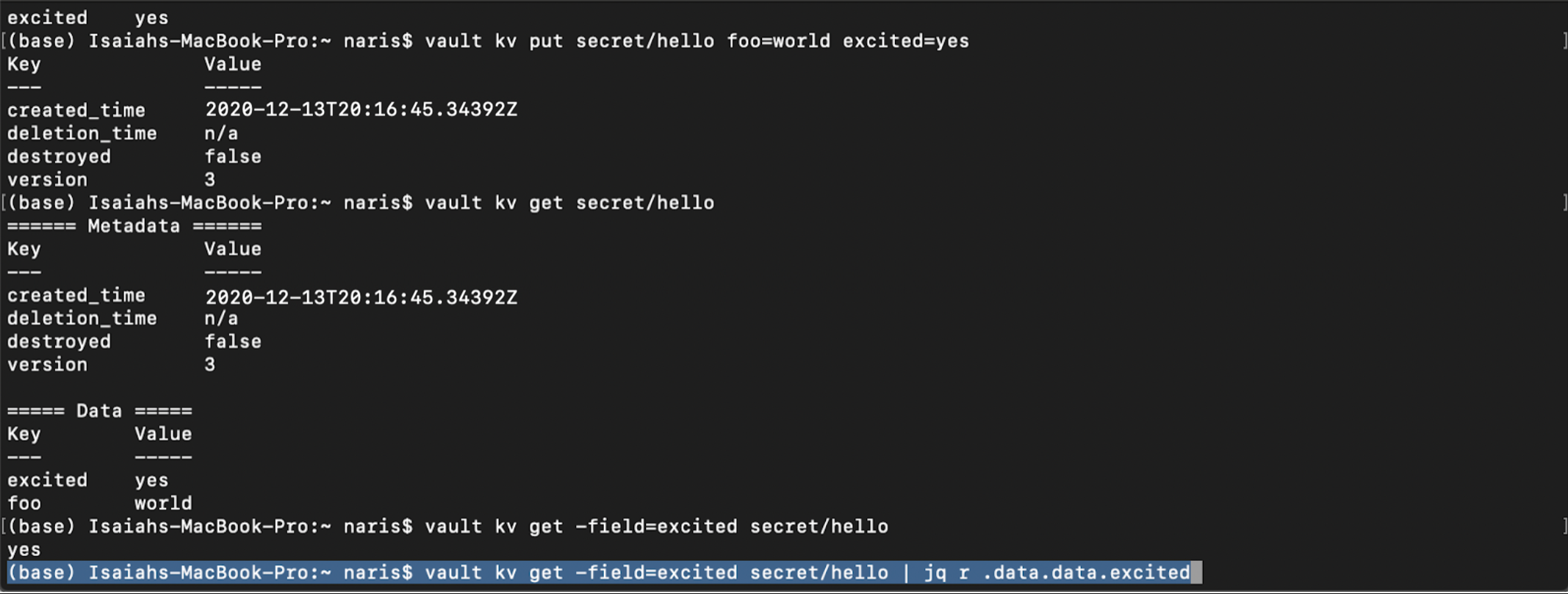
1. export vault\_addr

2. echo our unseal key into an `unseal.key` file

3. export our vault\_dev token

4. enter `vault\_status` to confirm that we ran it successfully





Reflection

Vault Real-World Application

The main problem that vault solves is the challenge of `secret management` which is essentially just credential management at its core. A credential is any piece of information whether that be a code, a string of keys, a hidden key, a passphrase, etc. that allows a user to have authorization, authentication, and access to otherwise inaccessible information.

At my job we work with credentials and IAM on a daily basis, figuring out who has the keys to what, how they are using them, what they are using with them, and how they are being stored. We have multiple keys for multiple different applications that we need to keep track and manage, and I see that vault will make it easier to centralize everything.

In a typical workflow environment utilizing encryption keys and authorizations, many of the keys are sprawled out across various sections of the project. Whether that be in the source code, in the configuration settings, or even embedded in a version control system. The challenge then becomes keeping track of who has access to what, do this AWS engineer have access to GitHub assets? These are the questions that HashiCorp vault aims to solve.

If there has been a compromise with one of these authentication tools, the next challenge would be how to be able to rotate between authentication measures? All of these measures are spread out all over the place and we don't really know who has access to what. This sort of scenario is known in the industry as `secret sprawl` where we have authentication measures sprawled throughout various systems in our systems architecture.

Where I would find vault security incredibly useful would be in applications where the security posture of your firm is paramount in the company. This way, with the ephemeral keys they are always changing, and being able to rotate authentication measures makes it that much more difficult for would-be hackers to gain access.

I really like this approach for the vault because it allows do you centralization of keys, and if anything were to happen we can securely lock down all access to everything in our infrastructure to better improve our Security posture.