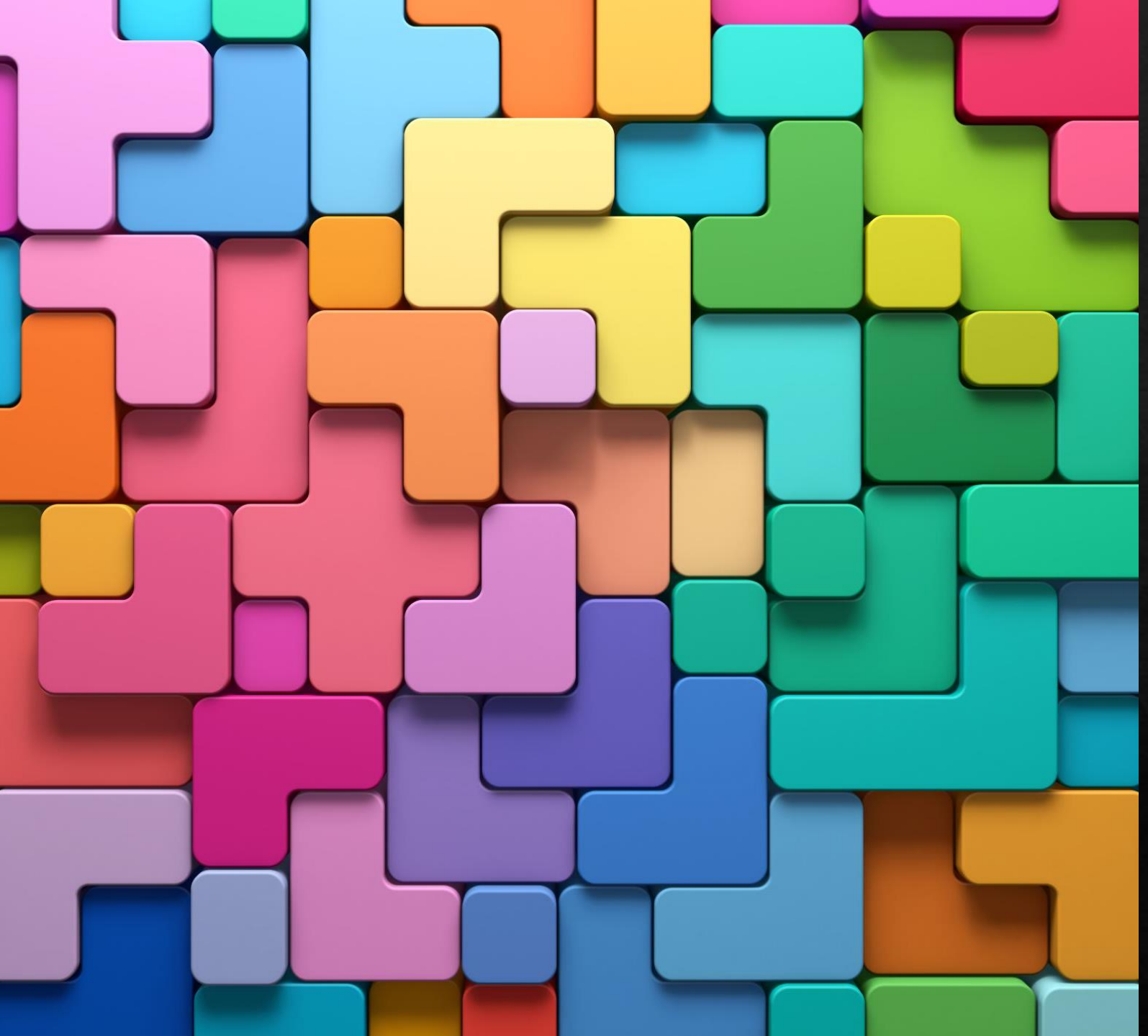


Raspberry Pi Automation with Ansible; and Feeding Sensor Data to Online Dashboards

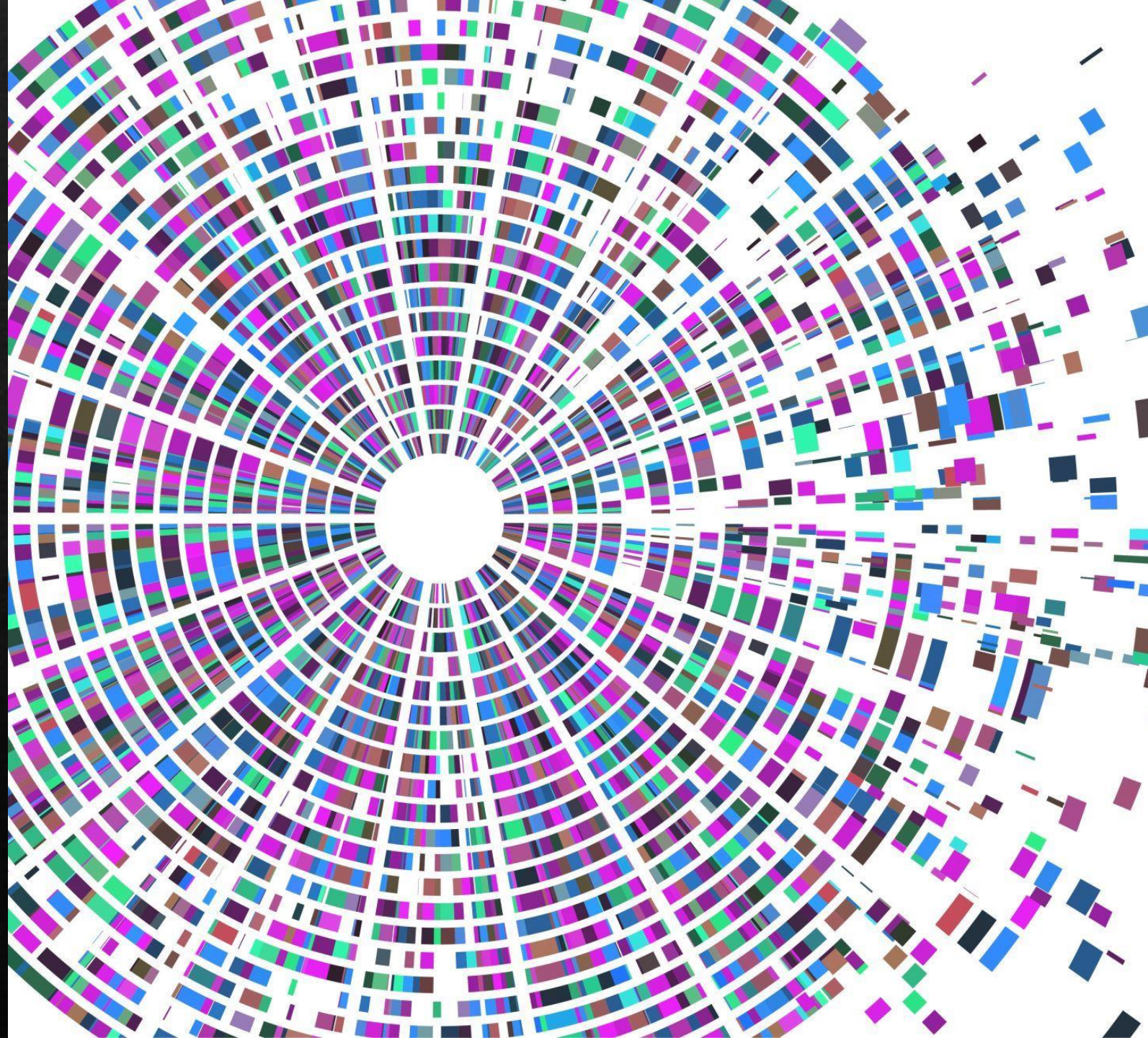
Aidan Parish

Internet of Things (101)

Introduction

I wanted to use the Raspberry Pi as a controller to automate patching and compliance in an environment that lacks a traditional domain controller; like an office space with a mix of different devices and operating systems. The solution that I present is using the Raspberry as an Ansible controller to automate patches and program installations on devices within the network (and much more) over SSH.

Ansible



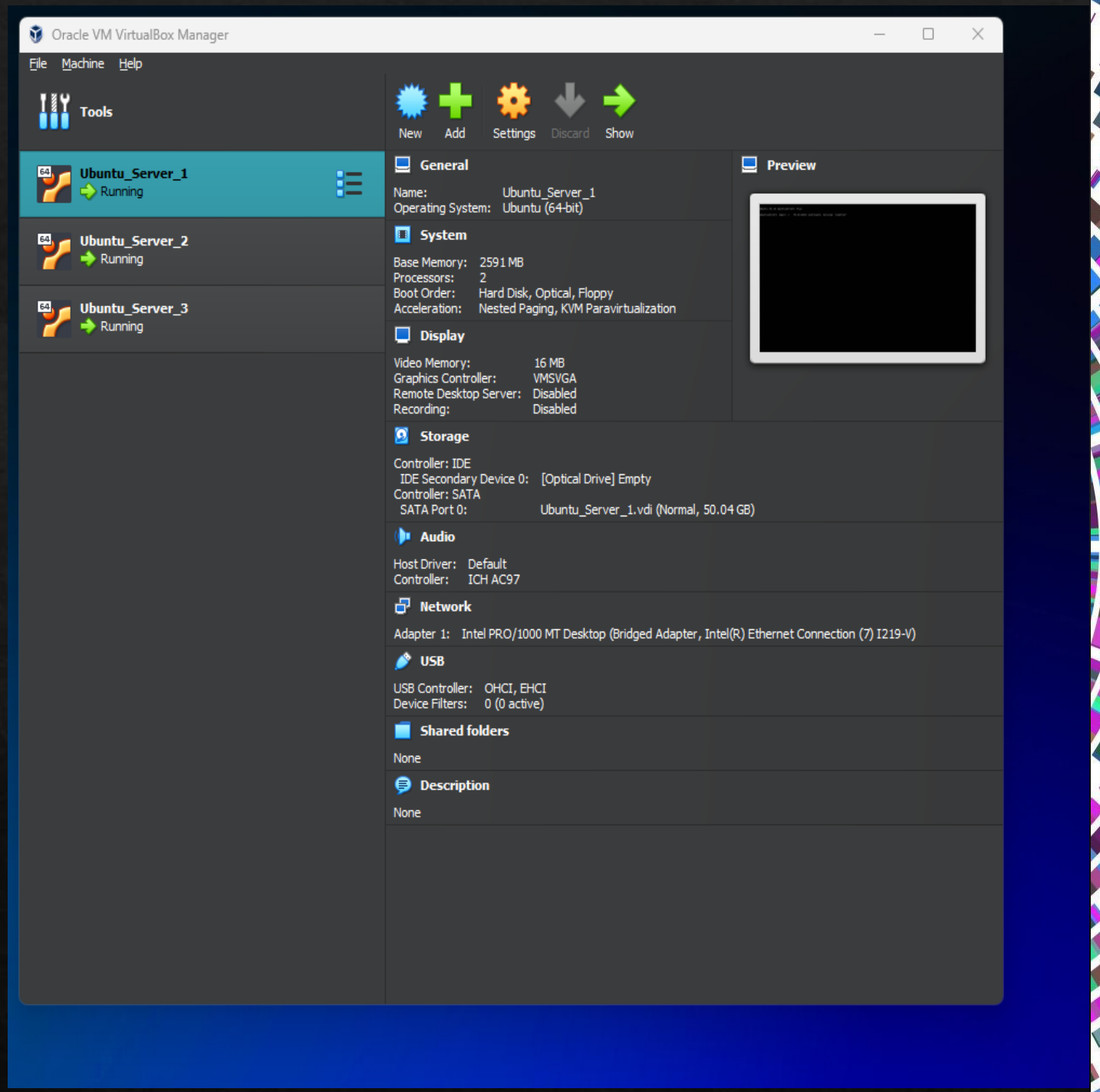
- Ansible is an automation platform that allows a controller device to send instructions (and automate workflows) to machines over the SSH (Secure Shell) protocol.

- Because Ansible works over SSH, you do not need a client agent on the devices that you are pushing automations to.

- Ansible works entirely over SSH, so all you need to do is ensure that the devices you are performing automations on have the appropriate public SSH key for the controller, and that all user passwords are passed along during the SSH connection. Best practice is to make a special user account on each device that you are automating specifically for Ansible.

- For this project, I made three separate Ubuntu server virtual machines.

- These are the servers that I will be performing automation tasks on to demonstrate the effectiveness of Ansible.

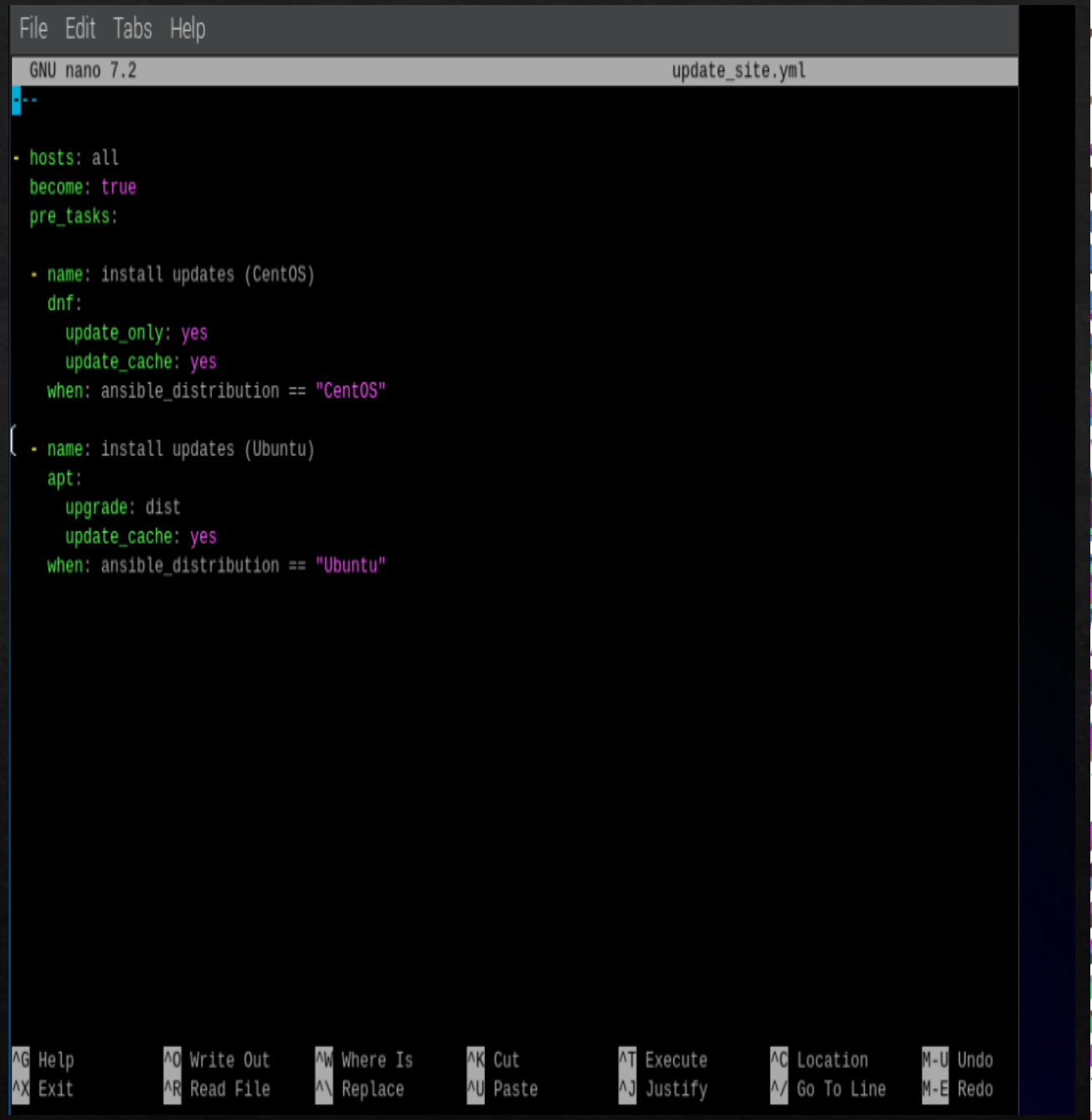


- While Ansible can be run solely from the command line with specific one-off commands, the best way to use Ansible is to make purpose-built playbook files. These playbook files allow you to perform complex tasks, or multiple tasks at once.

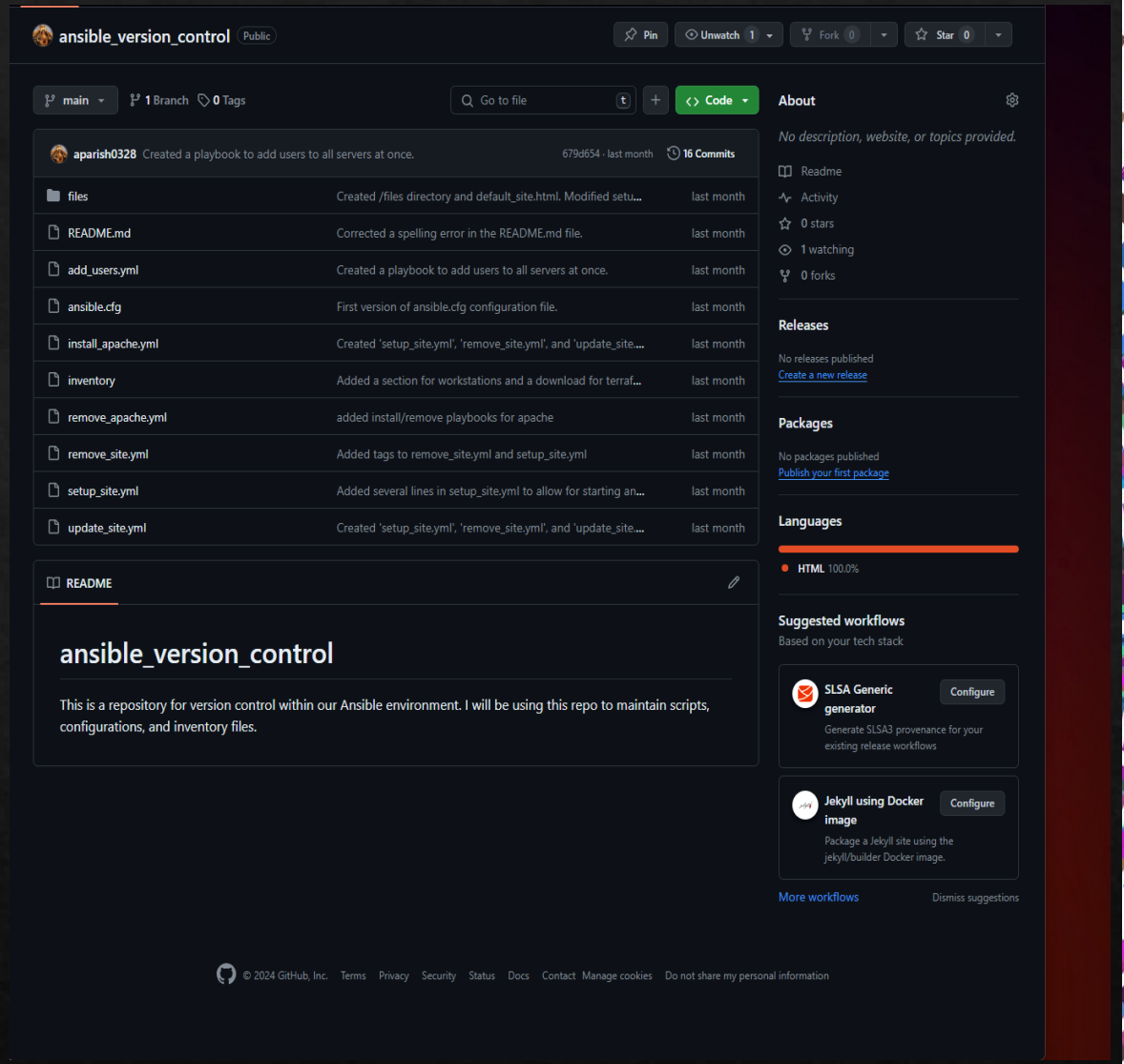
- These playbooks typically take the form of .yml or .yaml files.

- To run a playbook, you will also need an inventory list to run the playbook against. This inventory list is all of the devices that you want to perform automation tasks against.

- This example playbook pushes a simple update and install to all servers in the inventory.

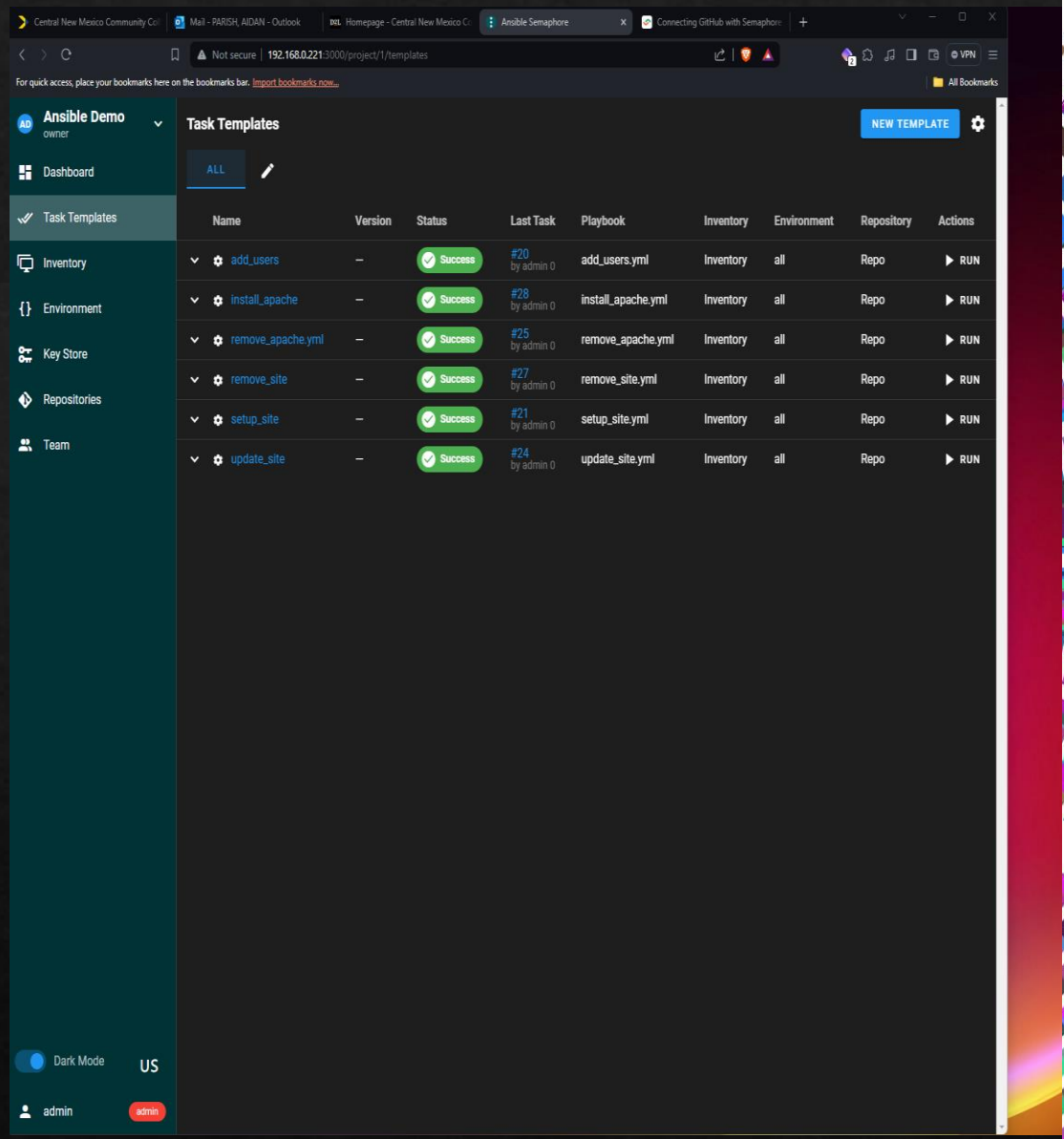


- After playbooks are made, they can be entirely automated using scheduling programs, such as Linux's crontab.
- When maintaining automation workflows, version control is very important. It is critical to ensure that Ansible playbooks and tasks are the most current version that they can be. To maintain proper version control, I have made a Github repository to house all of my playbooks.
- I can clone my repository into the ansible folder on the Ansible controller (this is running on my Raspberry Pi) to create parity between the two directories. I won't go to in depth on proper Github etiquette, but the basics of it are:
- To add to the Git repository, you use Git Commit/Git Push.
- To take from the Git repository, you use Git Pull.



My Github Repository can be found at: https://github.com/aparish0328/ansible_version_control

- There are a few Web UIs that are designed to provide a visual frontend for Ansible. One of them is AWX, and this is usually the one that you will see. However, I opted for a simpler and more streamlined web UI. It is also important to note that I was having difficulty getting AWX to work on Raspberry Pi but was able to make Ansible Semaphore to work relatively easily.



- Ansible Semaphore is the Web UI that I chose to use. It's lightweight compared to AWX and can be run in a Docker container; which is how I have it running. I actually used Portainer to manage this container and others. For Semaphore to work, you also need a MySQL Database running concurrently. I also have this containerized. Portainer allows me to keep my containers organized and to start and stop them easily.

- Semaphore addresses playbooks as 'Tasks.' In order to use your playbooks, you need to import them from an online Git repository; another reason why I made a version control Github Repository.

Feeding Sensor Data to an Online Dashboard

- For the physical aspect of my project, I have made a Python program that captures environment variables such as Ambient Room Temperature, Relative Humidity, as well as internal system variables such as CPU temperature and Ram Usage (using the output of the command free).

- The program then sends this information to my online Adafruit Dashboard every 30 seconds.

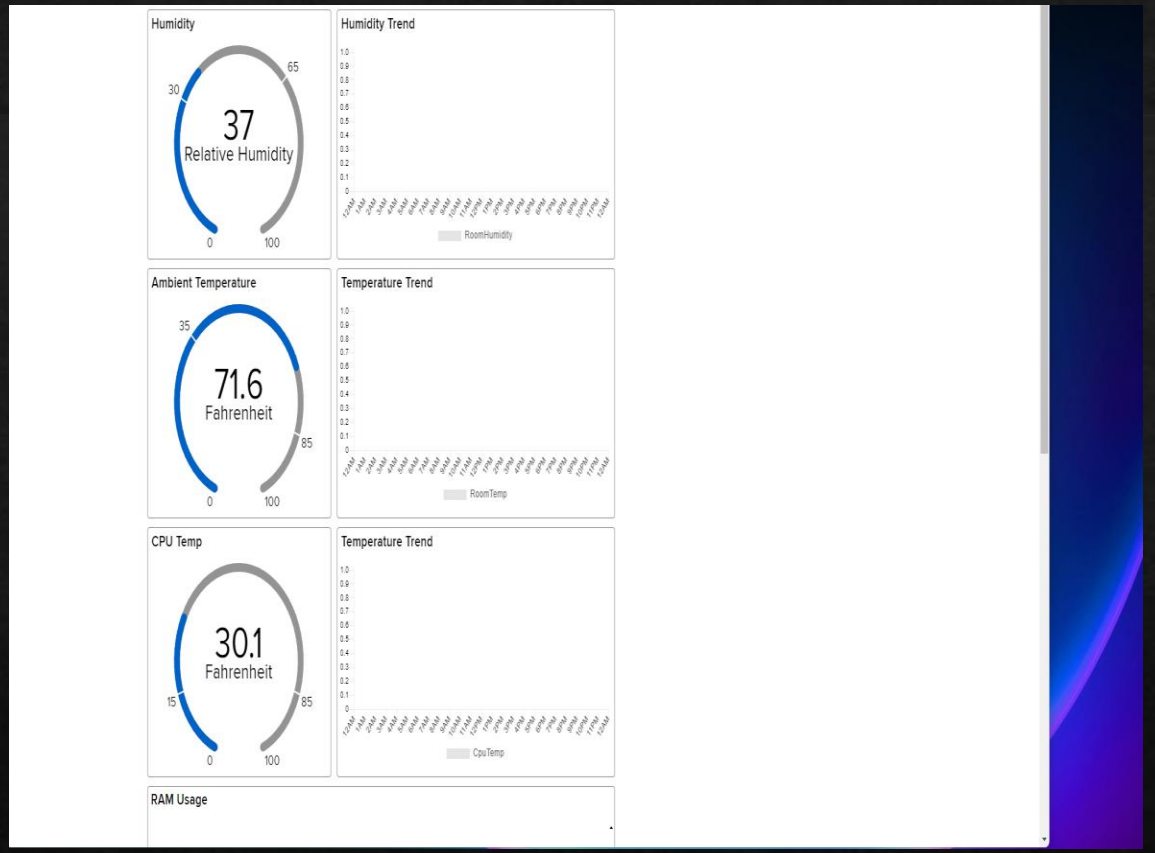
```
import time
import adafruit_dht
import board
from Adafruit_IO import Client
from subprocess import check_output
from re import findall


dht = adafruit_dht.DHT11(board.D4)
aio = Client('[your_username_here]','[your_key_here]')

def get_temp():
    temp = check_output(["vcgencmd","measure_temp"]).decode("UTF-8")
    return(findall("\d+\.\d+",temp)[0])

while True:
    try:
        ambTemp = (dht.temperature * 9/5) + 32
        smallTemp = round(ambTemp, 2)
        humidity = dht.humidity

        humfeed = aio.feeds('roomhumidity')
        aio.send_data(humfeed.key, humidity)

        tempfeed = aio.feeds('roomtemp')
        aio.send_data(tempfeed.key, smallTemp)

        cpu_temp = get_temp()
        cpufeed = aio.feeds('cputemp')
        aio.send_data(cpufeed.key, cpu_temp)

        RAM = check_output(["free"]).decode("UTF-8")
        ramfeed = aio.feeds('ram')
        aio.send_data(ramfeed.key, RAM)

    except RuntimeError as e:
        print("Reading failure: ", e.args)
    time.sleep(30)
```

Physical Sensor Setup

Power is delivered through the 3V3 pin, while data is transferred through GPIO pin 4.

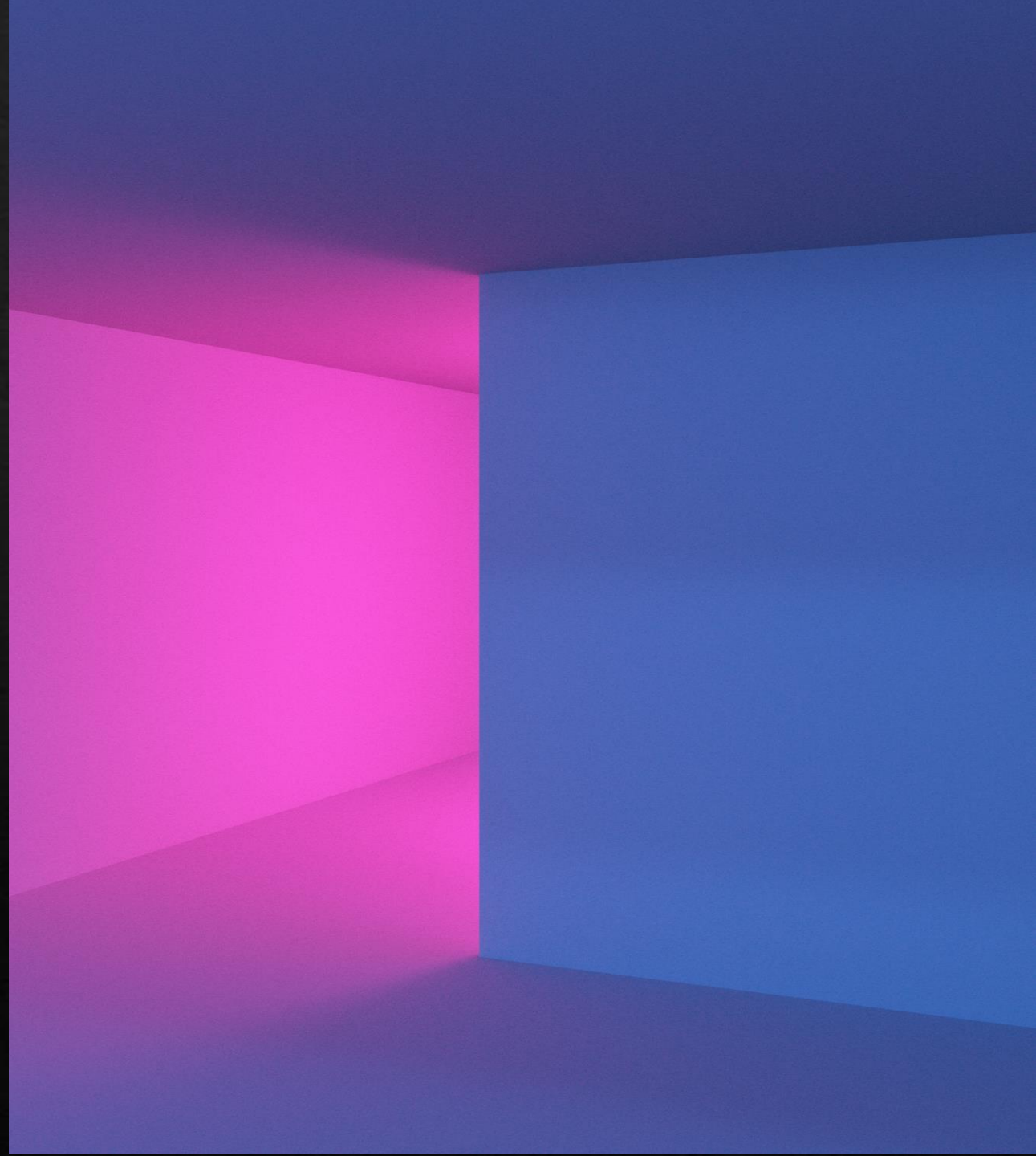
- After collecting the live data on Adafruit.io, I made a workflow that displays all of the raw data visually in the form of graphs.

- The picture to the right shows the online dashboard on Adafruit.io



Conclusion

- Ansible allows for the automation of endpoints without the need for client agents on these devices. Using SSH, Ansible is secure and easy to setup.
- Web UIs are useful tools to make using platforms like Ansible more intuitive to use with a visual interface.
- Version control is crucial for maintaining a good code base and can be done with Github or a similar Git repository (Gitlab, for example).
- Python can be used to address physical components. A very practical use for python is read sensor data (both external and internal) and feed them to online dashboards using webhooks, APIs, MQTT, etc.



References

- Guide on Using Adafruit.io: https://www.jeremymorgan.com/tutorials/raspberry-pi/how-to-iot-adafruit-raspberrypi/

- Docker Image for Ansible Semaphore: https://github.com/ChristianLempa/boilerplates/blob/main/docker-compose/ansiblesemaphore/docker-compose.yaml

- Video on Ansible Semaphore: https://www.youtube.com/watch?v=NyOSoLn5T5U&t=242s

- This Playlist taught me everything I needed to know about Ansible: https://youtube.com/playlist?list=PLT98CRl2KxKEUHie1m24-wkyHpEsa4Y70&si=XoQIMslxo_lXsE_-

- My Github Repository for Ansible Version Control: https://github.com/aparish0328/ansible_version_control