Building a Slackbot to manage your Raspberry Pi Kubernetes cluster

Kesi Soundararajan

Summer Intern

Introduction

Who am I?

Junior at the **University of Southern California**

Studying **Computer Science and Business Administration**

Tech enthusiast and tinkerer

Raspberry Pi hobbyist

VMware **Summer Intern**

Agenda

Context

What is Docker, Kubernetes, Slack, Raspberry Pi's?

The Plan

Specifications, concerns, etc.

Using Raspberry Pi's as a Kubernetes Cluster

How you can use this for your own project!

Integration with Slack

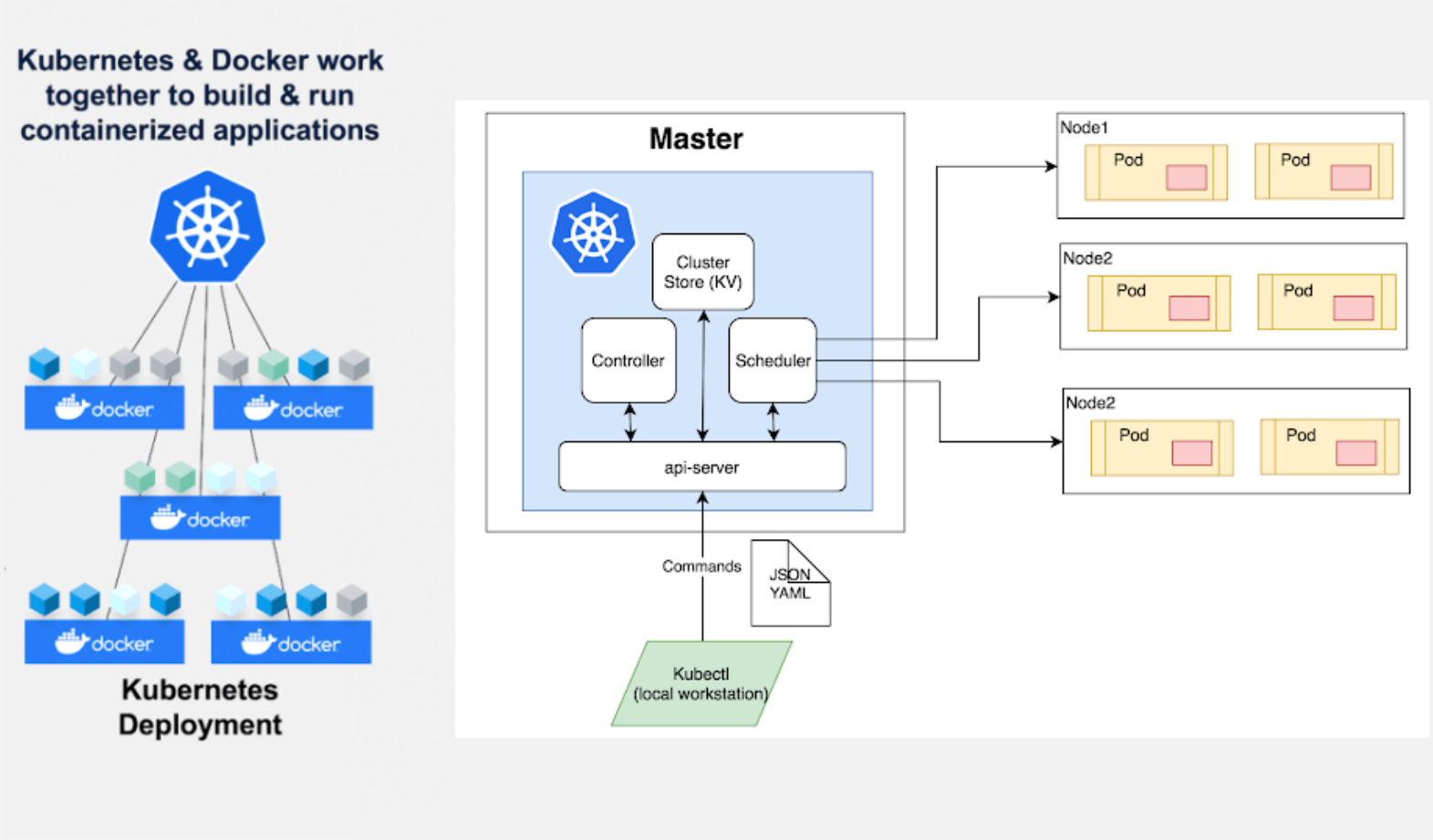
Using the Slack API to make something cool

Closing Remarks

Issues encountered, lessons learned

What is Docker and Kubernetes?

How do they work together to deploy your workload



Docker

Containerizes your application into packaged containers with all their dependencies included

Advantages:

- Highly Efficient
- Portable across platforms

Kubernetes

Open-source container orchestration tool

Deploys containers as pods to and manages deployment of pods to worker nodes

What is Slack?

Popular messaging service for teams to collaborate

Slack API

- Robust API for retrieving information from a Slack workspace
- Easy to build applications and bots around it and deploy to your workspace

What is a Raspberry Pi?

Not the desert

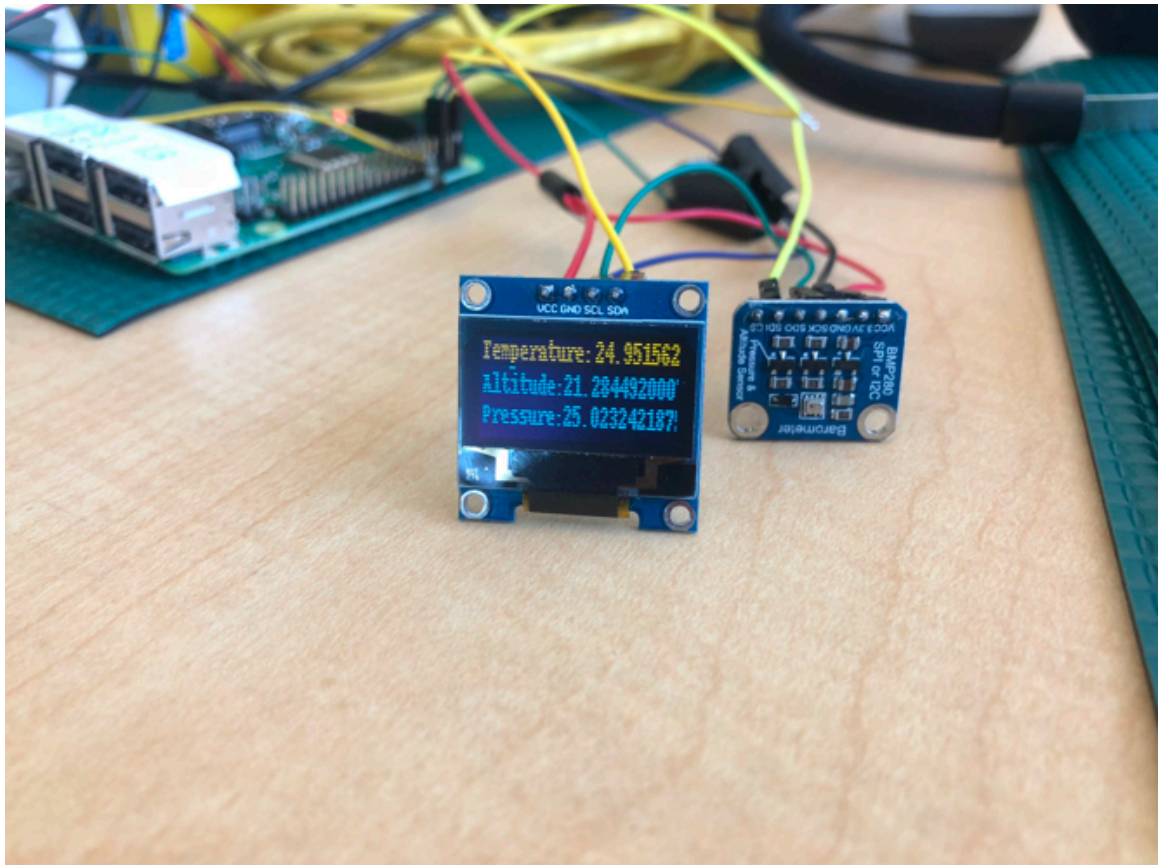
Low cost **microcontroller** the size of a credit card

Quad Core, ARM architecture based CPU with up to **8 gigs** of ram (Specs of high-end Raspberry Pi 4)

Linux based operating system, most common being Raspbian/Raspberry Pi OS (based on Debian meant for Raspberry Pi's)

GPIO pins for sensors and other hardware attachments!

Example Raspberry Pi Project with sensors!



BMP 280 sensor (temperature, pressure, altitude)

OLED display

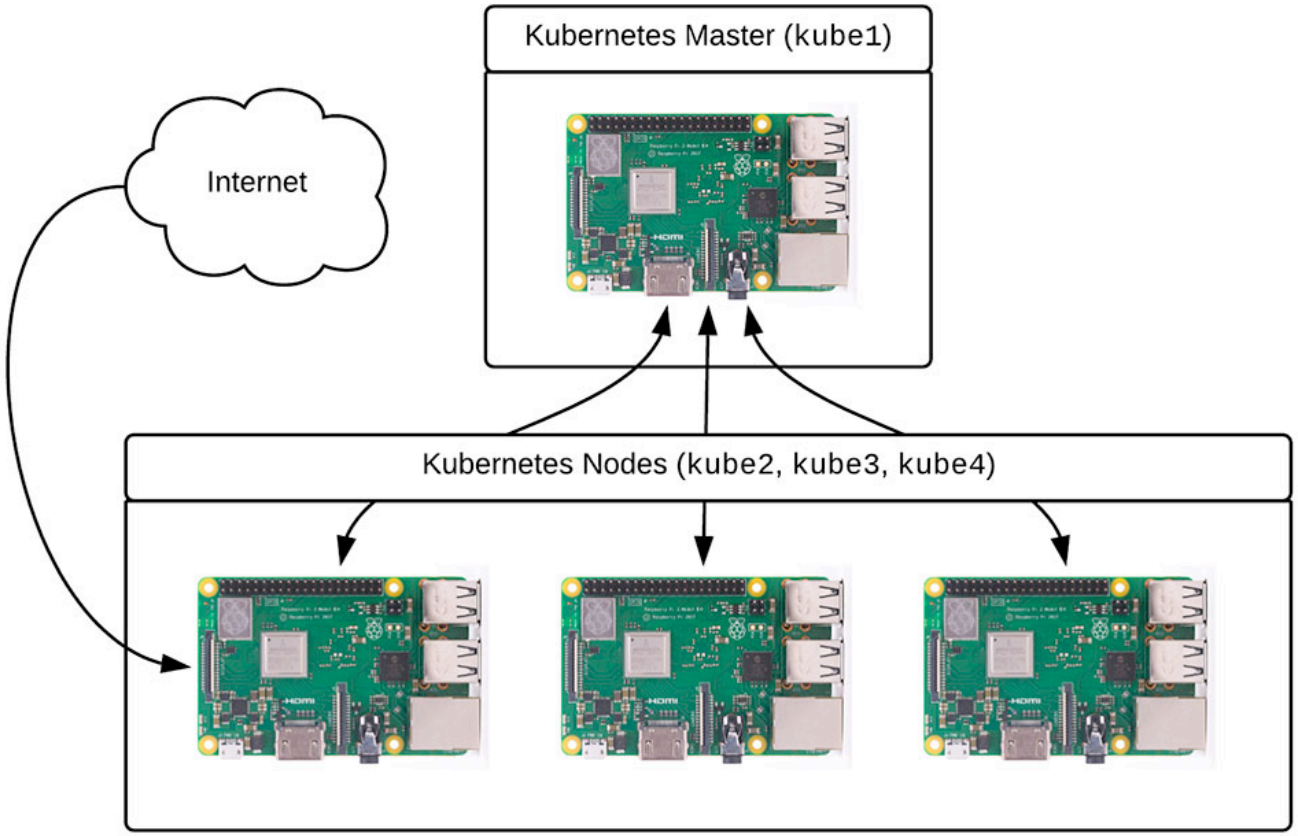
Python App to display sensor data on the OLED

Using Raspberry Pi's as Kubernetes Worker Nodes to deploy an app

The Plan:

Use Raspberry Pi's as nodes in a Kuberenetes cluster

Use Kubernetes cluster to deploy our sensor app!



Steps to build a Raspberry Pi Kubernetes cluster

Install Kubernetes and assign Pi as master node, connect other Pi's to master node as workers

Use Docker to containerize app

Deploy app as Kubernetes pod on our worker node using a YAML file

Setting up your Raspberry Pi for Kubernetes

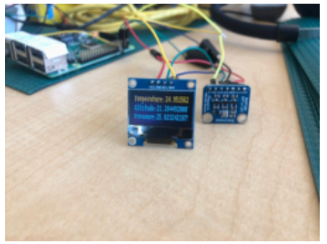
1. Flash Raspbian/Raspberry Pi OS onto an SD card

2. Set static hostname and IP address for your Pi's

3. Install Kubernetes and Docker on your Pi's

Scripts and in-depth instruction for these steps can be found at

https://blogs.vmware.com/code/2019/06/14/kubernetes-raspberry-pi-cluster/



Kubernetes on a Raspberry Pi Cluster

Kripa Sitaraman posted June 14th, 2019

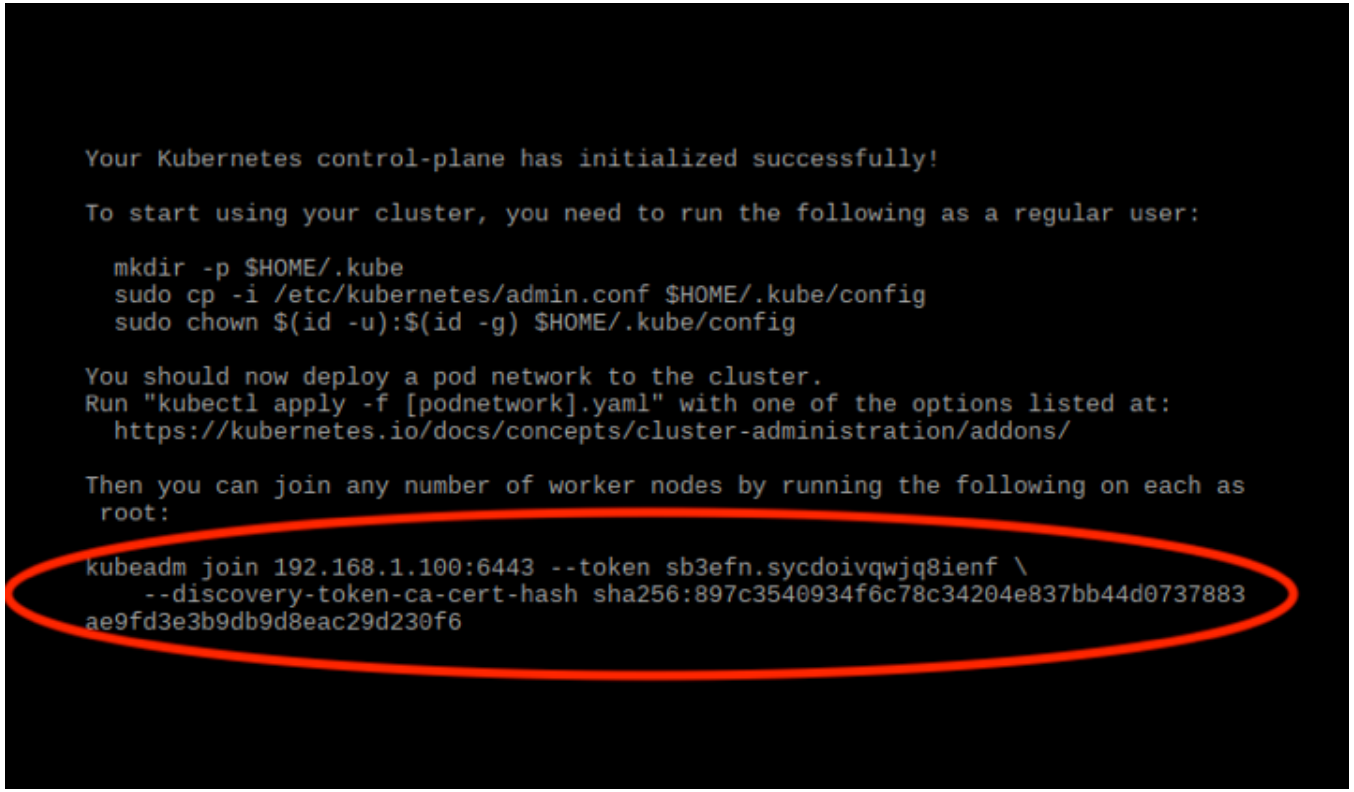
Kubernetes on a Raspberry Pi Cluster By Kesi Soundararajan I. Introduction Kubernetes is an open source, container orchestration tool that

Initializing your cluster and joining your worker nodes

With all the prerequisites in place, we can use our master node to initialize a Kubernetes cluster.

```
1 $ sudo kubeadm init
```

After initializing the cluster, there will be a token created for worker nodes to join the cluster like below:



Installing a container Network

```
1 $ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
```

Your nodes will not be in the “Ready” state until Weave is correctly installed and launched. If the above command does not work as expected, troubleshooting can be done with the $ weave stop and $ weave launch commands, and checking if Weave shows up in the list of cluster internal pods with $ sudo kubectl get po –all-namespaces

```
pi@masterpi:~ $ sudo kubectl get nodes --kube
NAME         STATUS   ROLES    AGE    VERSION
masterpi     Ready    master   2d3h   v1.14.3
workerpi     Ready    <none>   2d2h   v1.14.3
```

Package our application into docker

Creating a Dockerfile

The first step in turning our script into a fully independent containerized Docker image is making a Dockerfile. A Dockerfile essentially lays out the dependencies and parent image to include when packaging our app. It should be created with no file extension and must be put in the same directory as your Python code. For this project, our Dockerfile looks like so:

```
1  FROM arm32v7/python:3-slim-stretch
2
3  RUN apt-get update -y
4
5  RUN apt-get install -y build-essenti
6
7  RUN pip install Pillow
8
9  RUN pip install Adafruit_SSD1306
10
11 RUN pip3 install adafruit-circuitpyt
12
13 ADD BMPDisp.py /
14
15 CMD [ "python", "./BMPDisp.py"]
```

Build your Docker image and upload to Dockerhub

Building a Docker Image

Our Dockerfile is ready to go, now it's time to build it!

```
$ sudo docker build -t [buildname] .
```

Running this should make Docker start building an image, following all the steps we laid out in the Dockerfile. To make sure the Docker image has built correctly, try running it through Docker with:

```
$ sudo docker run --privileged [build
```

Uploading to the Dockerhub repository

Make a Dockerhub account, and then login with:

```
$ docker login
```

Since we have built the image locally, running $ docker images should show the image ID that we can use to tag it and push it onto the repository

```
$ docker tag [image id] [username]/[r

$ docker push [username]/[repository]
```

Deploy workload to Kubernetes

Creating a YAML file

IV. Deploying your workload to Kubernetes

Kubernetes uses YAML files to generate pods and assign workloads to nodes. For the purposes of this project, we want to show that our script can run on individual nodes as specified by our YAML file. On the master, create a blank YAML file and copy and paste the contents below into that file.

```
apiVersion: v1
kind: Pod
metadata:
      name: bmp-disp
spec:
        nodeName: [name of the node you want to run the image on]
        containers:
        - name: bmp-disp
           image: vmwarecode/bmp280:v1
           imagePullPolicy: IfNotPresent
           securityContext:
                 capabilities:
                       add: ["SYS_ADMIN"]
                 privileged: true
                 allowPrivilegeEscalation: true
```

Deploying a pod using our YAML file

To actually create the pod, run

```
1 $ kubectl create -f [filename].yaml
```

To check if the pod is running, use

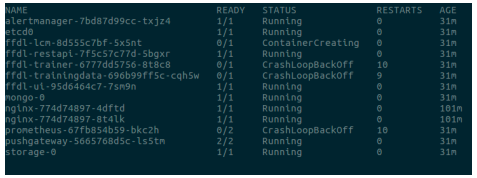
```
1 $ kubectl get pods
```

If the pod is up and running, you should see our script running on the worker node you specified in the YAML file!

```
NAME                                READY   STATUS              RESTARTS   AGE
alertmanager-7bd87d99cc-txjz4       1/1     Running             0          31m
etcd0                               1/1     Running             0          31m
ffdl-lcm-8d555c7bf-5x5nt            0/1     ContainerCreating   0          31m
ffdl-restapi-7f5c57c77d-5bgxr       1/1     Running             0          31m
ffdl-trainer-6777dd5756-8t8c8       0/1     CrashLoopBackOff    10         31m
ffdl-trainingdata-696b99ff5c-cqh5w  0/1     CrashLoopBackOff    9          31m
ffdl-ui-95d6464c7-7sm9n             1/1     Running             0          31m
mongo-0                             1/1     Running             0          31m
nginx-774d74897-4dftd               1/1     Running             0          101m
nginx-774d74897-8t4lk               1/1     Running             0          101m
prometheus-67fb854b59-bkc2h         0/2     CrashLoopBackOff    10         31m
pushgateway-5665768d5c-ls5tm        2/2     Running             0          31m
storage-0                           1/1     Running             0          31m
```

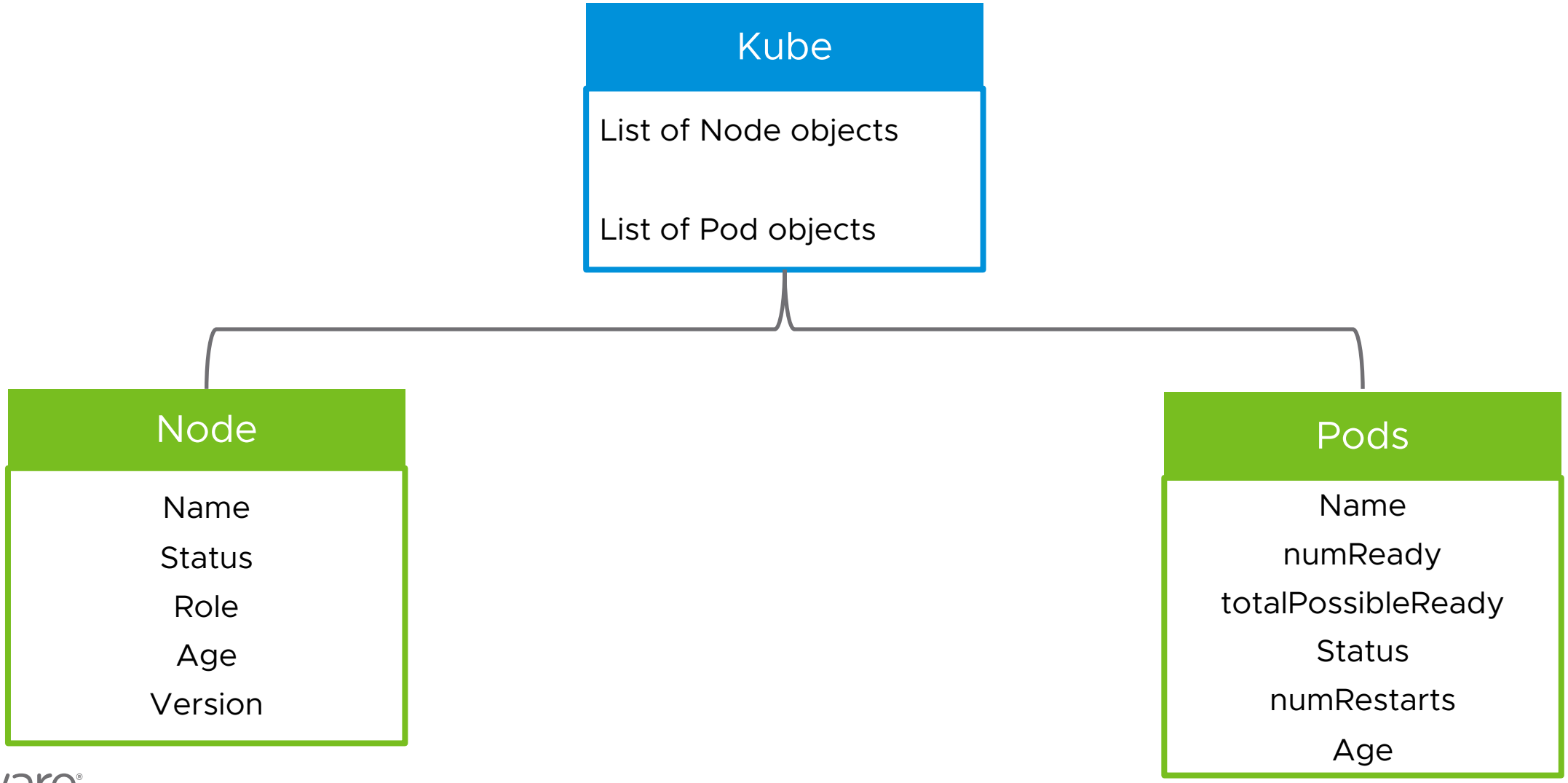
Integrating with the Slack API

Specifications

Slack API	Language	Library	Functionality
slack api	C++	Slacking - Lazy modern C++ people also loves Slack !	
Allows us to read messages and post in our slack channel when @'d	Familiar with the language, allows us to access functionality of the Slack API	Lightweight library that allows us to write C++ code and parse what the Slack API returns (JSON C++ library built in)	Bot should be able to output information about nodes and pods

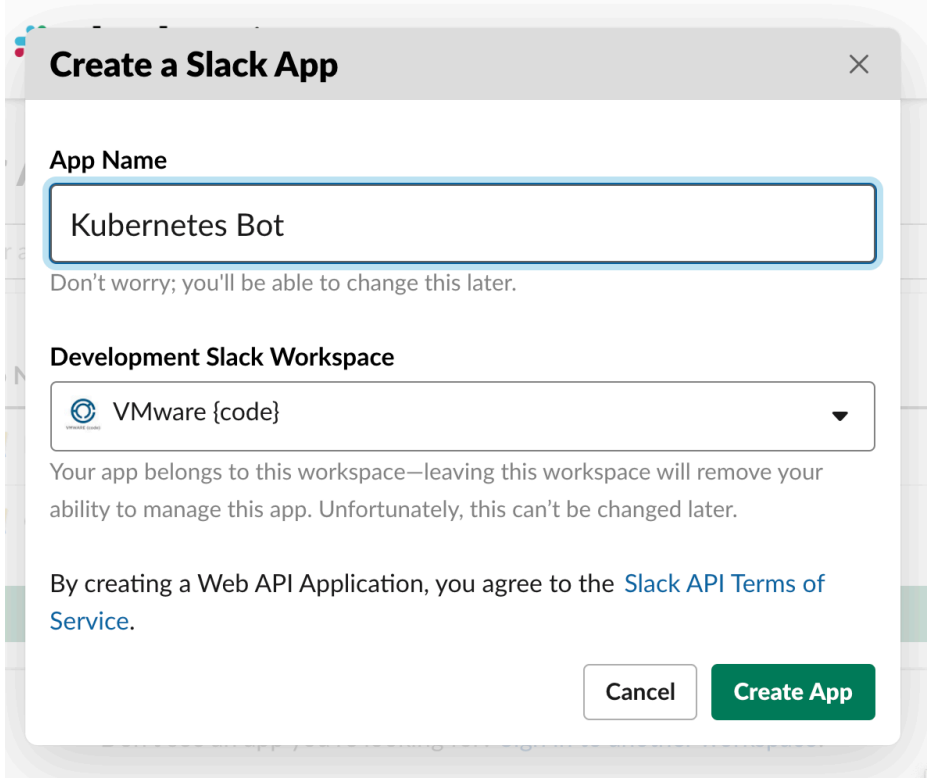
Getting information about Kubernetes

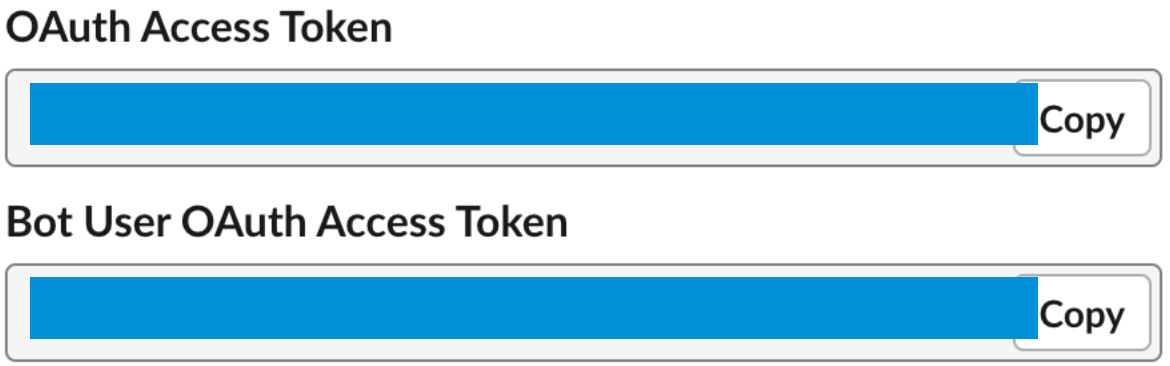
Kubectl get nodes/pods -> C++



Setting up the Slack API

Getting your token and Creating an Slack instance






```
void CreateSlackInstance(string botToken) {
    auto& slack = slack::create(botToken);
}
```

Setting up the Slack API

Setting up Permissions

Bot Token Scopes

Scopes that govern what your app can access.

OAuth Scope	Description
channels:history	View messages and other content in public channels that kubernetesbot has been added to
chat:write	Send messages as @kubernetesbot
groups:history	View messages and other content in private channels that kubernetesbot has been added to
im:history	View messages and other content in direct messages that kubernetesbot has been added to
incoming-webhook	Post messages to specific channels in Slack
mpim:history	View messages and other content in group direct messages that kubernetesbot has been added to

Calling the Slack API for information

Setting up Permissions

C++ Code → Slack API Call → JSON response

```
auto getResponse = slack::post(
        "conversations.history",
        {
            {"channel", CHANNEL_ID},
            {"limit", "1"}
        }
    );

if (getResponse["ok"] == "false") {
    return "something went wrong";
}
```


```
{
    "ok": true,
    "messages": [
        {
            "type": "message",
            "user": "U012AB3CDE",
            "text": "I find you punny and would like to smell your no
            "ts": "1512085950.000216"
        },
        {
            "type": "message",
            "user": "U061F7AUR",
            "text": "Isn't this whether dreadful? <https://badpuns.ex
            "attachments": [
                {
                    "service_name": "Leg end nary a laugh, Ink.",
                    "text": "This is likely a pun about the weather."
                    "fallback": "We're withholding a pun from you",
                    "thumb_url": "https://badpuns.example.com/puns/12
                    "thumb_width": 1920,
                    "thumb_height": 700,
                    "id": 1
                }
            ],
            "ts": "1512085950.218404"
```

The Result!

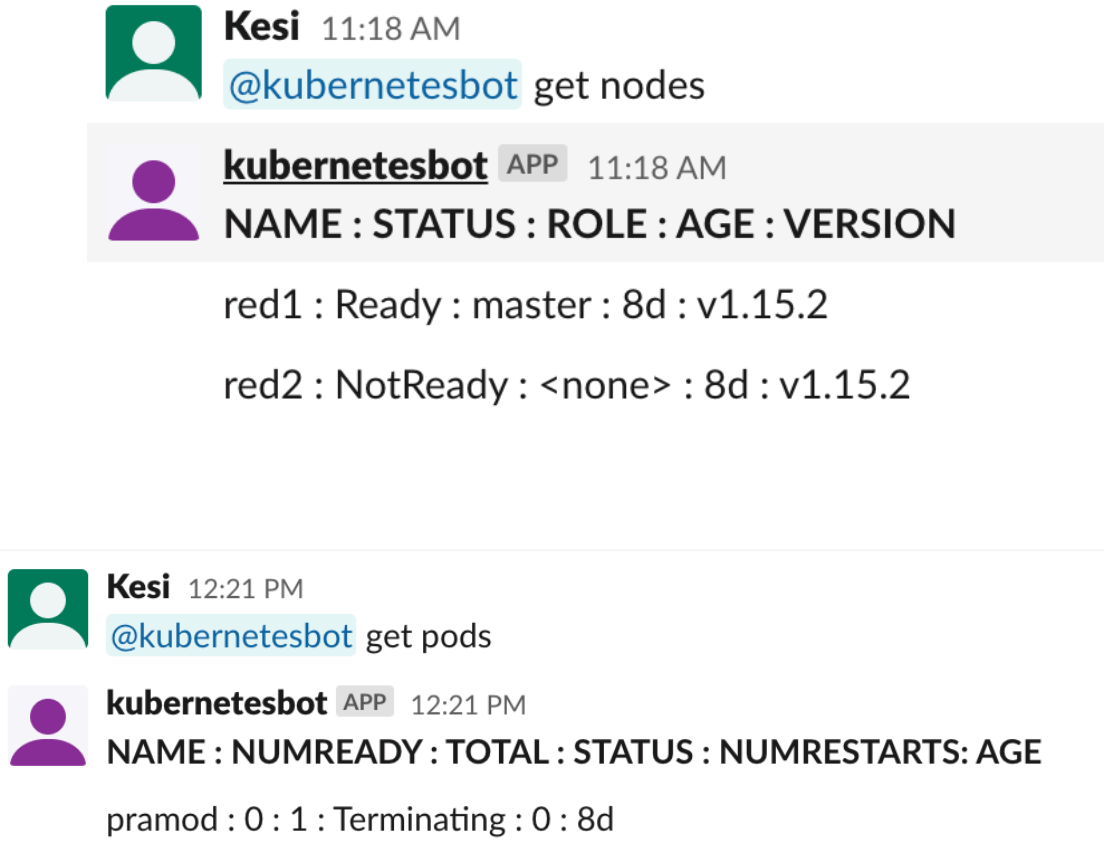
What our bot can do when we put this all together

Written in C++

Uses the Slack API to get information from a Raspberry Pi Kubernetes cluster and sends messages about them back via Slack

Alerts you if node or pod goes down in real time

Some fun chatbot features that you can personalize with a rules file



Closing Remarks

Issues Encountered, Links to learn more

Using C++ with the Slack API

Made things more difficult, better libraries for slack built in Python and Ruby

Github Slackbot Project:

https://github.com/kesisoundusc/kubernetesslackbot

Building a Kubernetes Raspberry Pi Cluster Blog Post:

https://blogs.vmware.com/code/2019/06/14/kubernetes-raspberry-pi-cluster/