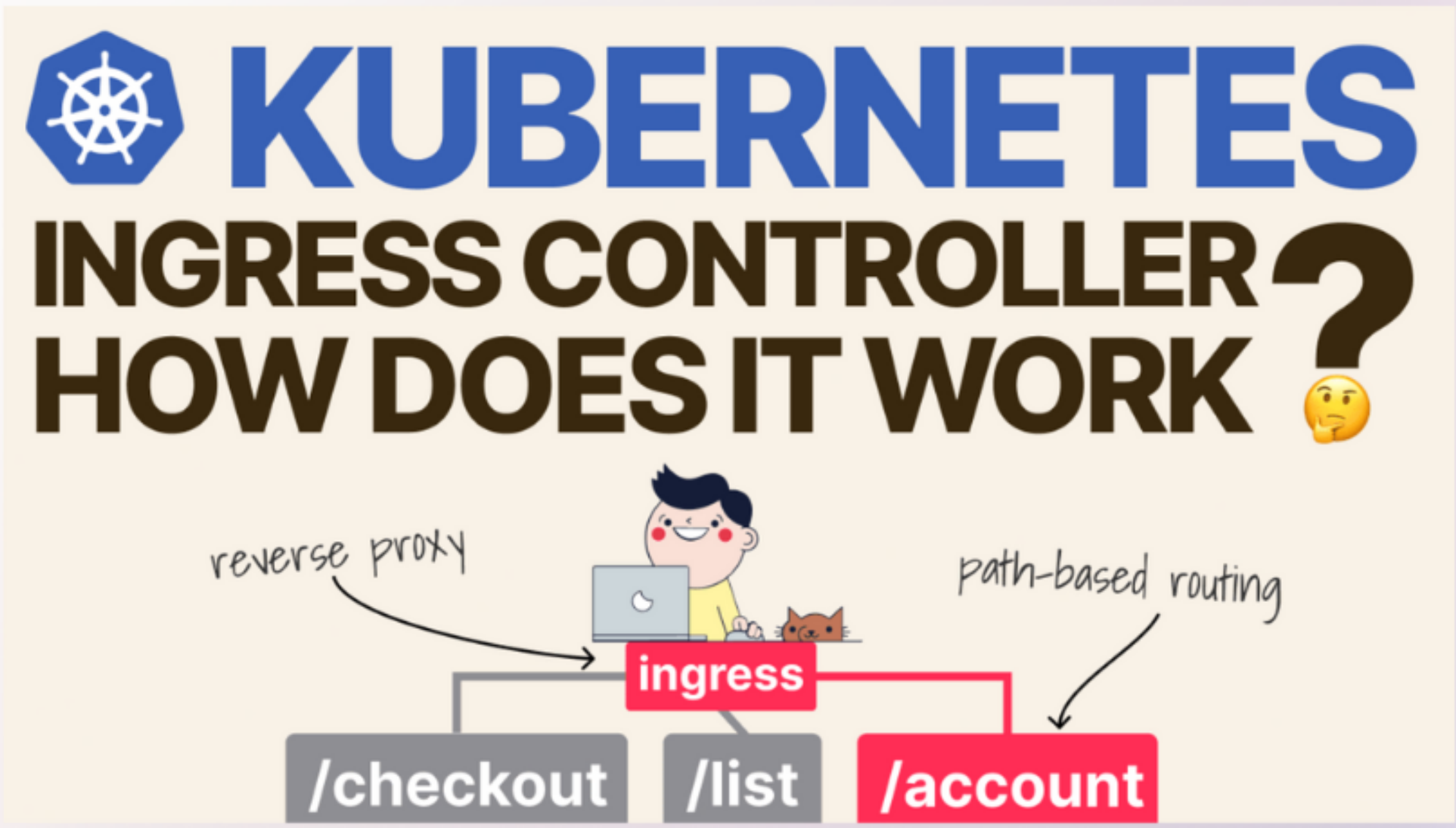
Daniele Polencic
@danielepolencic

How does the Ingress controller really work in Kubernetes?

I had to find out for myself, so I built one from scratch in bash



7:48 PM · 16 Jan, 2023

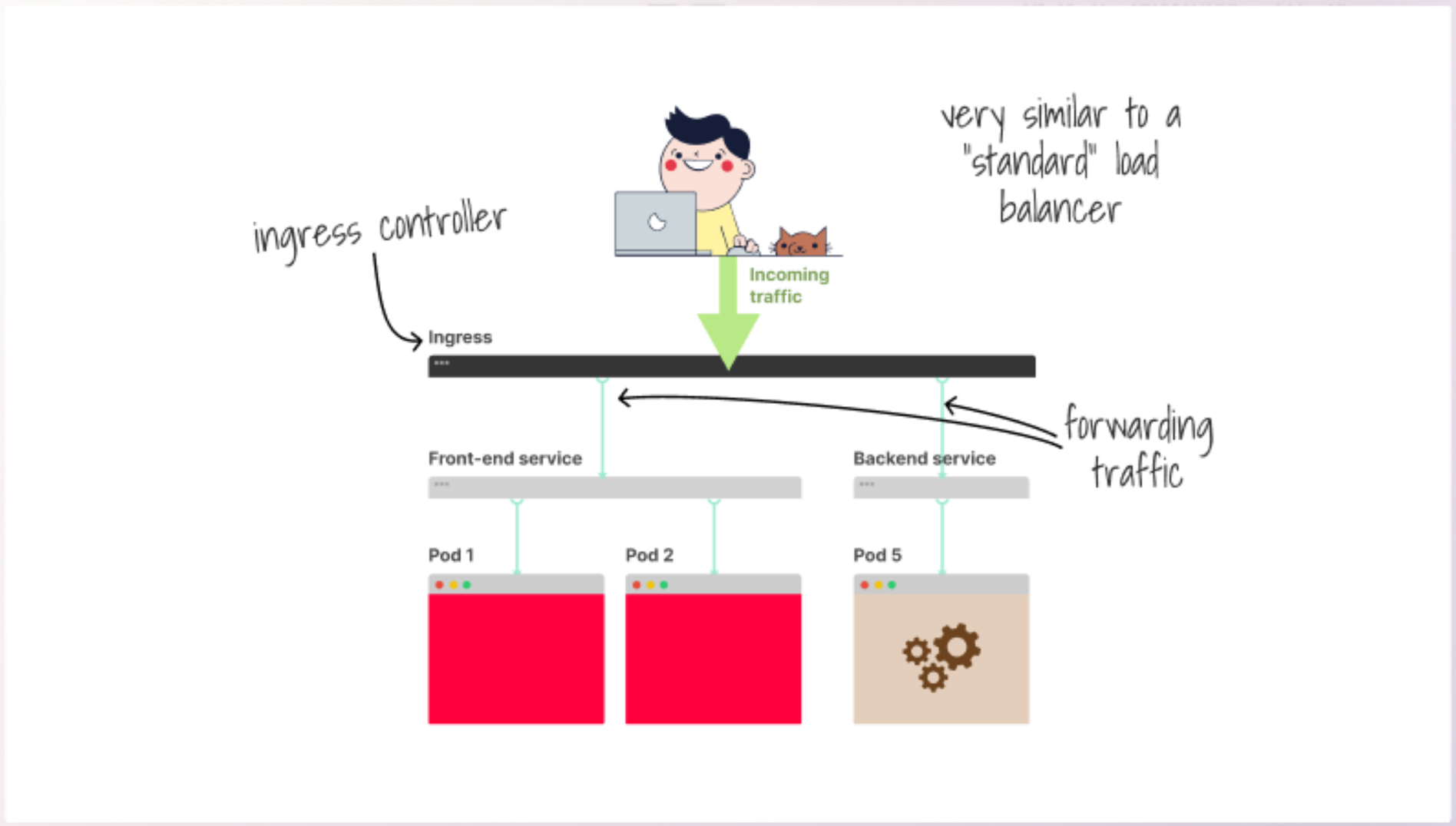
22 replies **213** shares **829** likes

Daniele Polencic
@danielepolencic

1/

Before diving into the code, here is a quick recap on how the ingress controller works

You can think of it as a router that forwards traffic to the correct pods



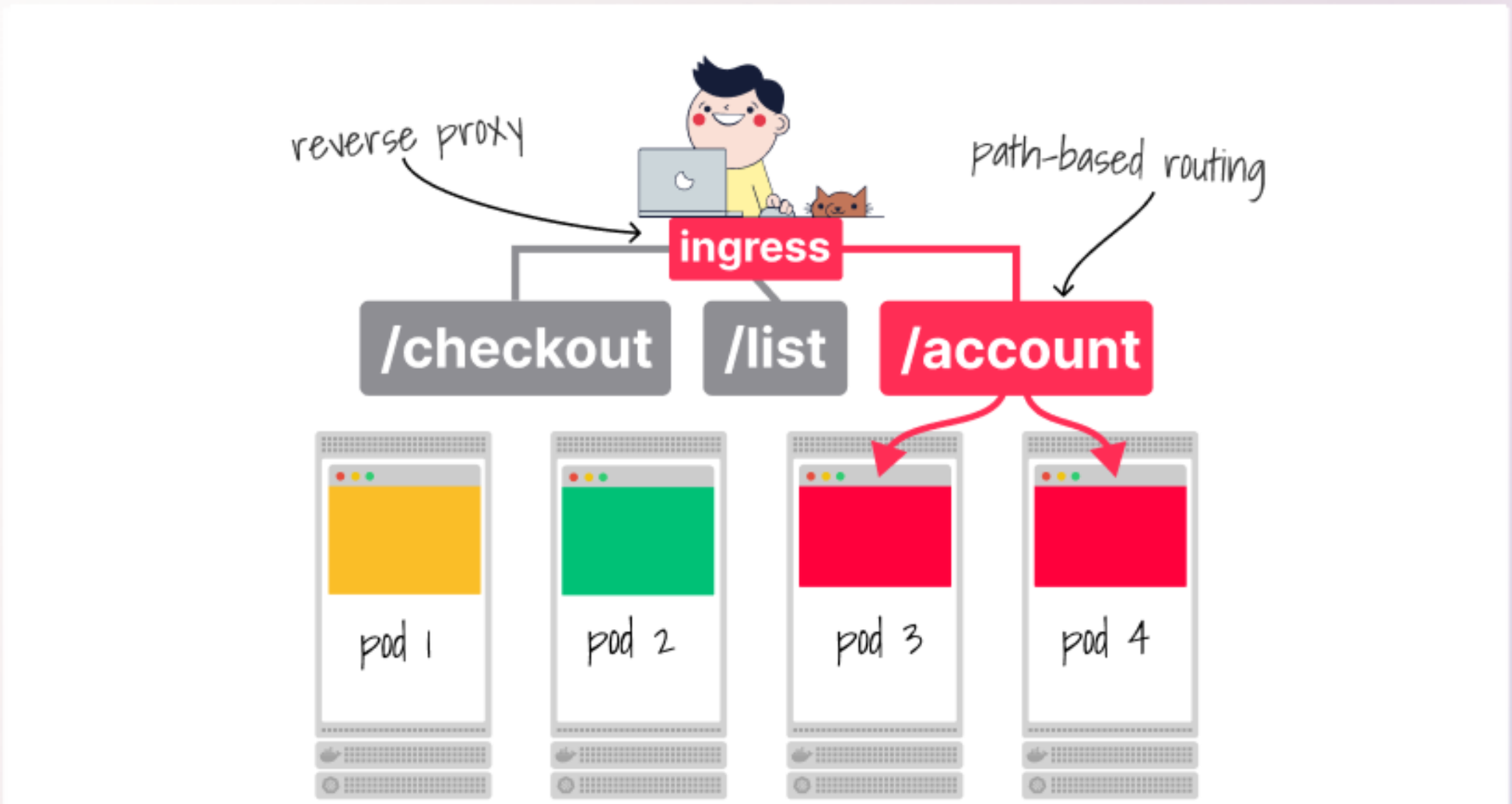
7:48 PM · 16 Jan, 2023

2 replies **2** shares **12** likes

Daniele Polencic
@danielepolencic

2/

More specifically, the ingress controller is a reverse proxy that works (mainly) on L7 and lets you route traffic based on domain names, paths, etc



7:48 PM · 16 Jan, 2023

1 reply **6** likes

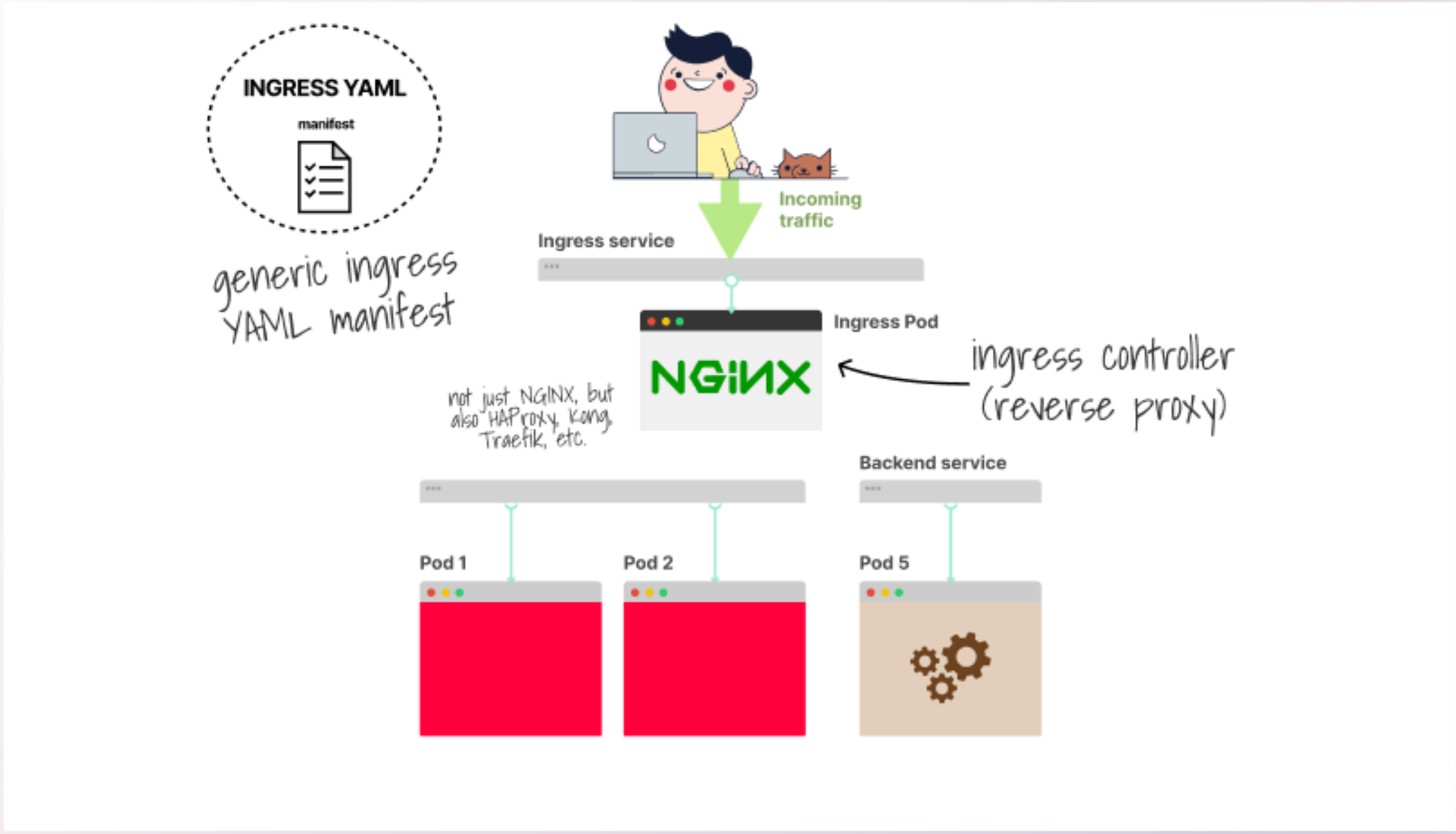
Daniele Polencic
@danielepolencic

3/

Kubernetes doesn't come with one by default

So you have to install and configure an Ingress controller of choice in your cluster

But it provides a universal manifest (YAML) definition



7:48 PM · 16 Jan, 2023

1 reply **1** shares **3** likes

Daniele Polencic
@danielepolencic

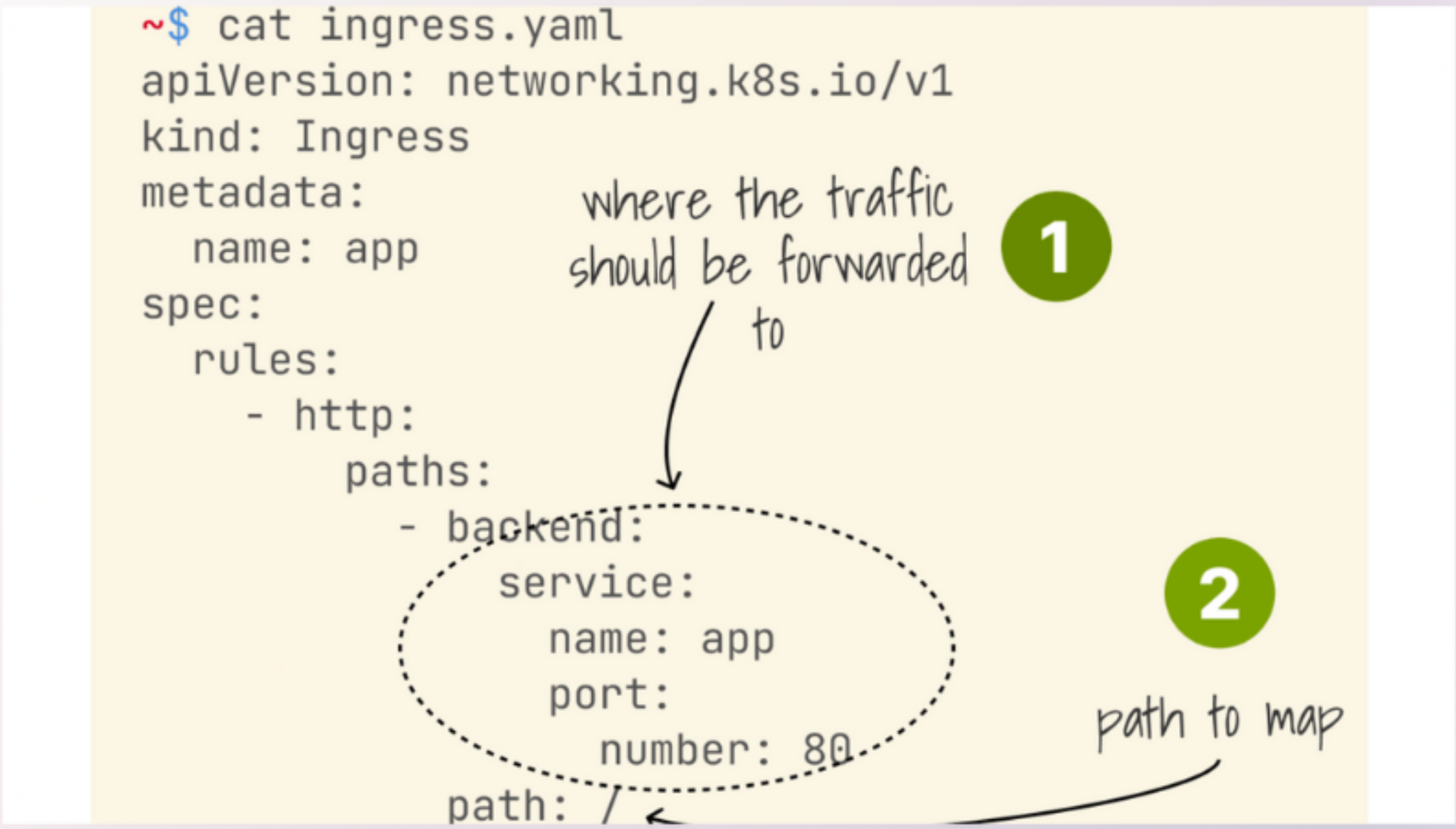
4/

The exact YAML definition is expected to work regardless of what Ingress controller you use

The critical fields in that file are:

① The path

② The backend



7:48 PM · 16 Jan, 2023

1 reply 3 likes

Daniele Polencic
@danielepolencic

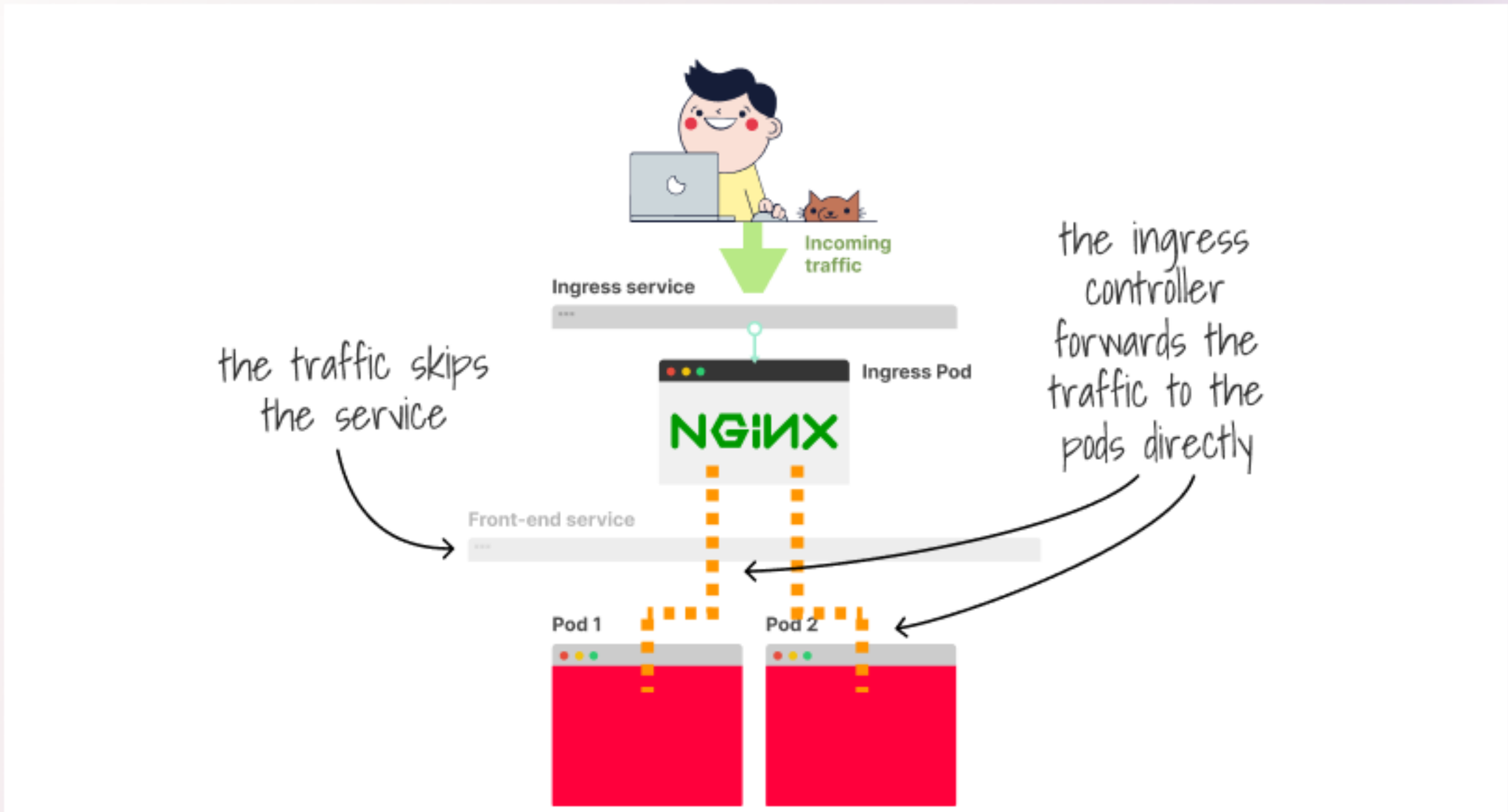
5/

The backend describes which service should receive the forwarded traffic

But, funny enough, the traffic never reaches it

This is because the controller uses endpoints to route the traffic

What is an endpoint?



7:48 PM · 16 Jan, 2023

1 reply **5** likes

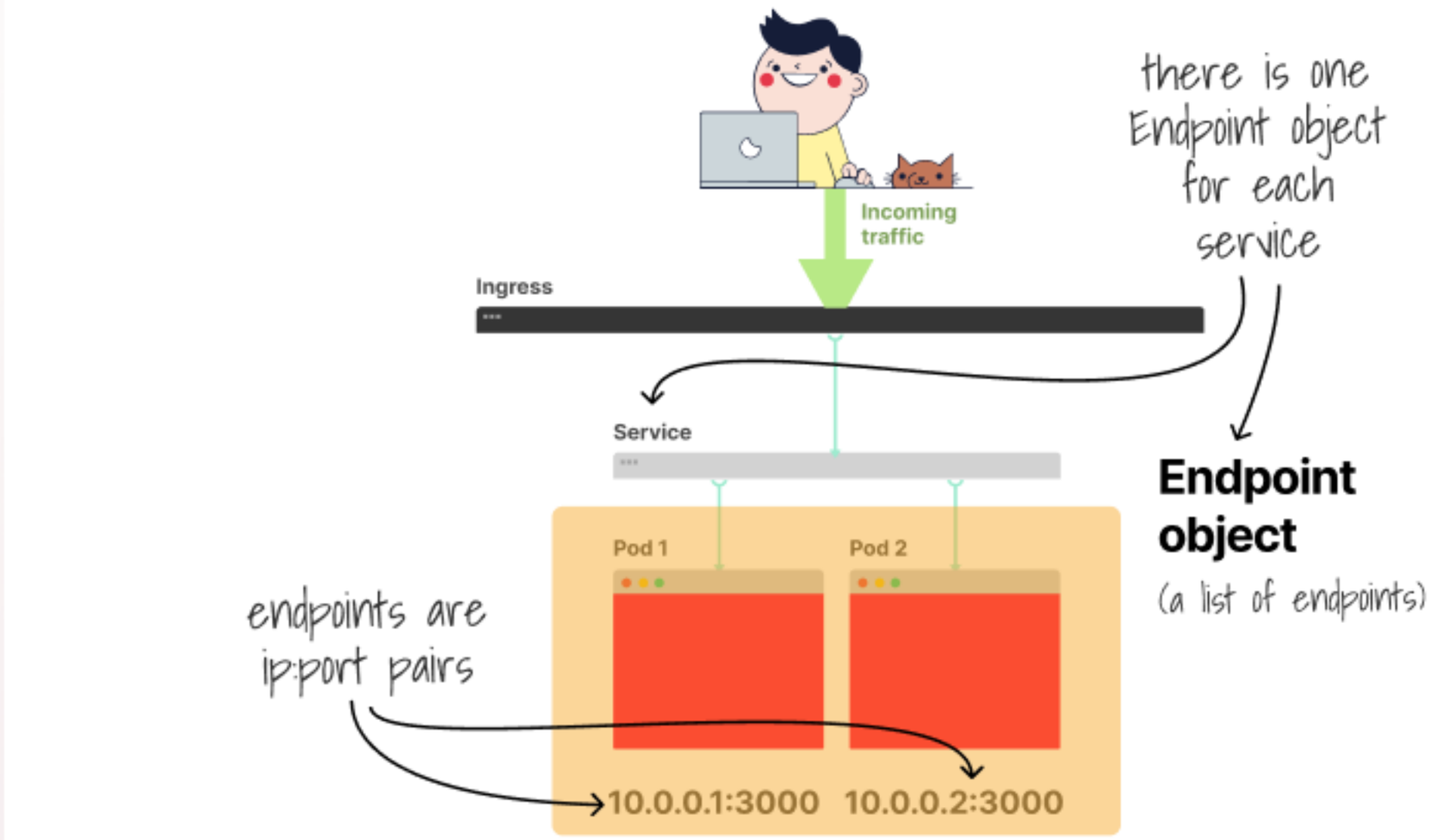
Daniele Polencic
@danielepolencic

6/

When you create a Service, Kubernetes creates a companion Endpoint object

The Endpoint object contains a list of endpoints (ip:port pair)

The IP and ports belong to the Pod



7:48 PM · 16 Jan, 2023

1 reply **1** shares **5** likes

Daniele Polencic
@danielepolencic

7/

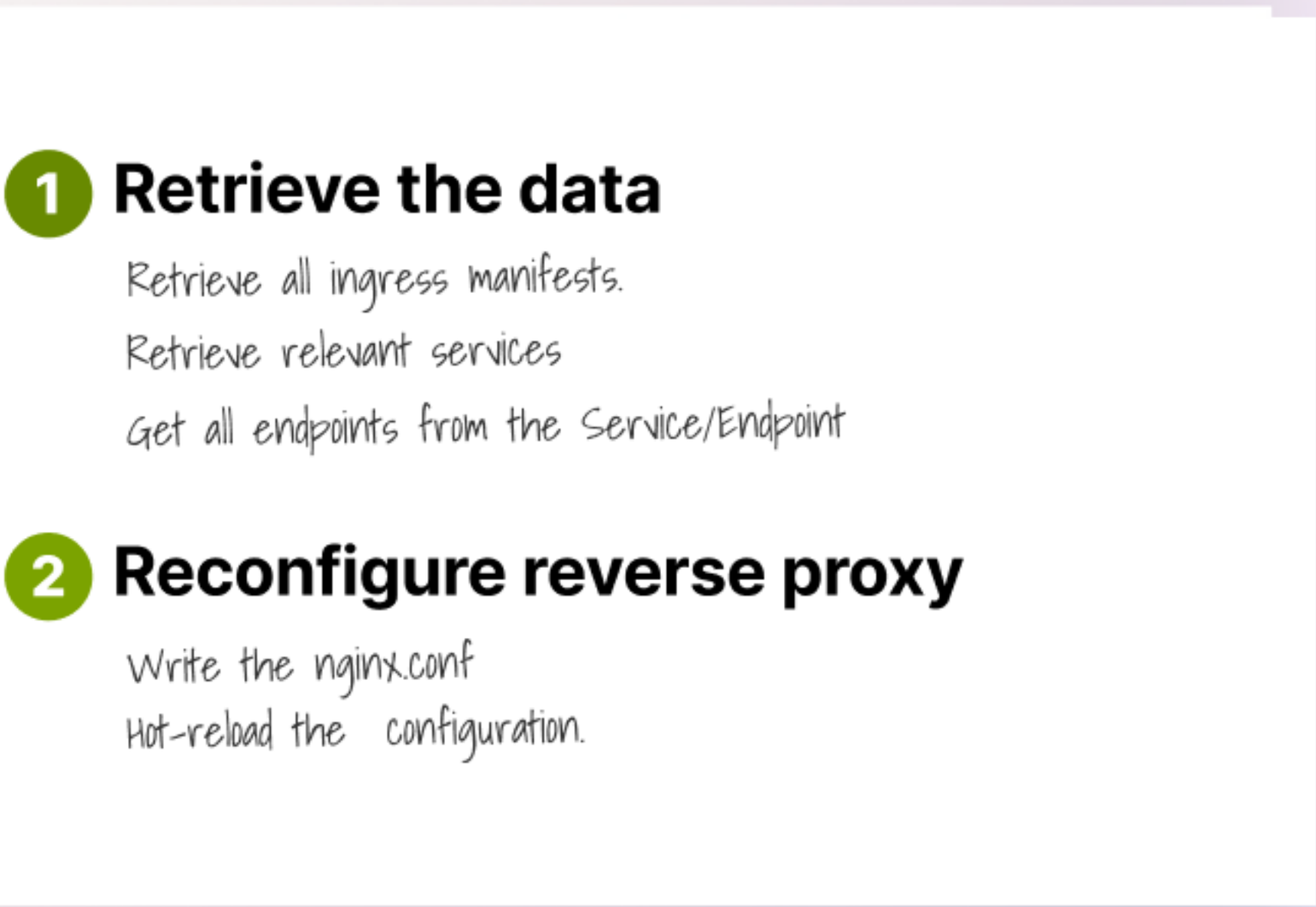
Enough, theory

How does this work in practice if you want to build your own controller?

There are two parts:

① Retrieving data from Kubernetes

② Reconfiguring the reverse proxy



7:48 PM · 16 Jan, 2023

1 reply **1** shares **3** likes

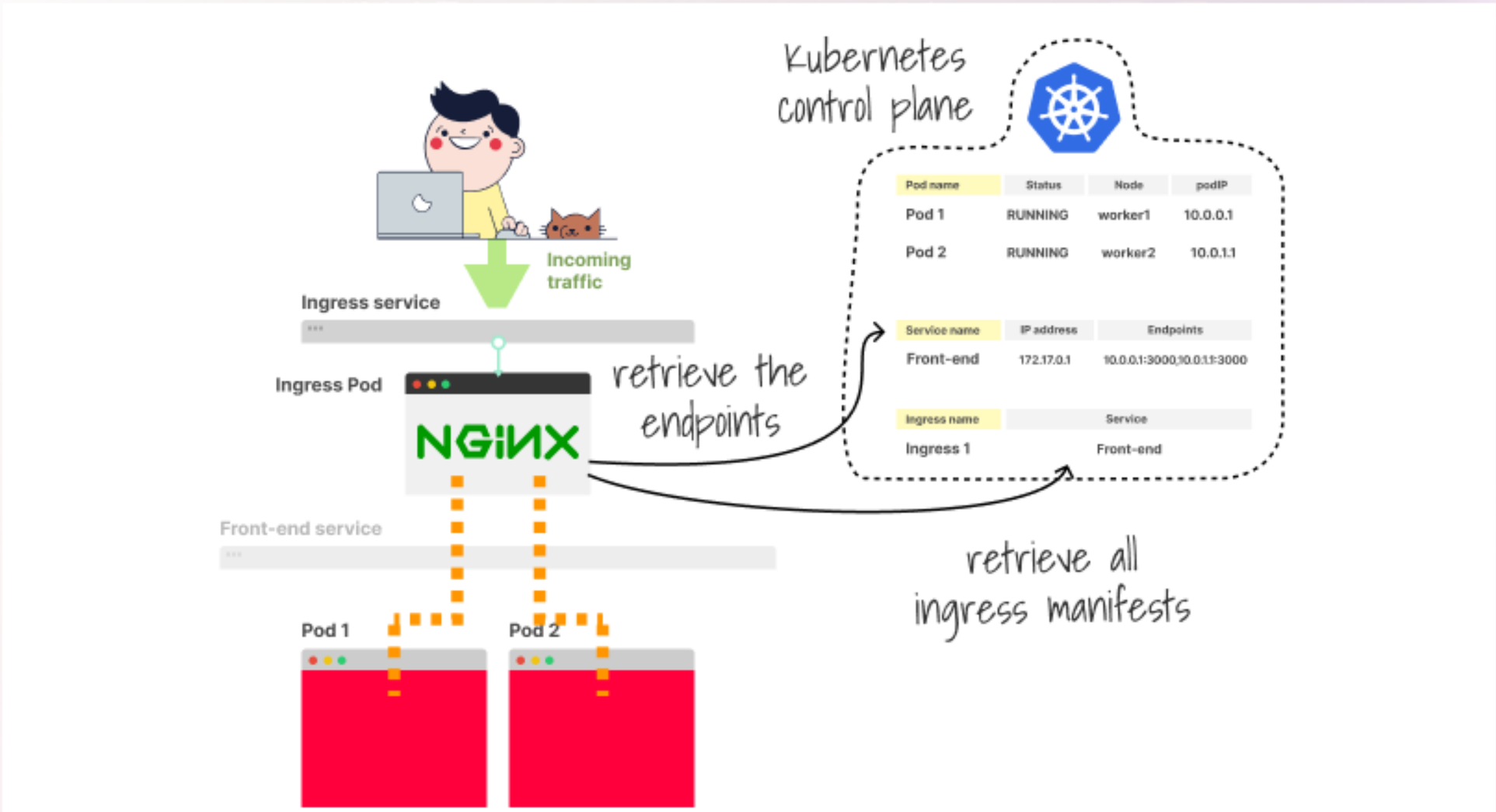
Daniele Polencic
@danielepolencic

8/

In 1), the controller has to watch for changes to Ingress manifests and endpoints

If an ingress YAML is created, the controller should be configured

The same happens when the service changes (e.g. a new Pod is added)



1 reply 2 likes

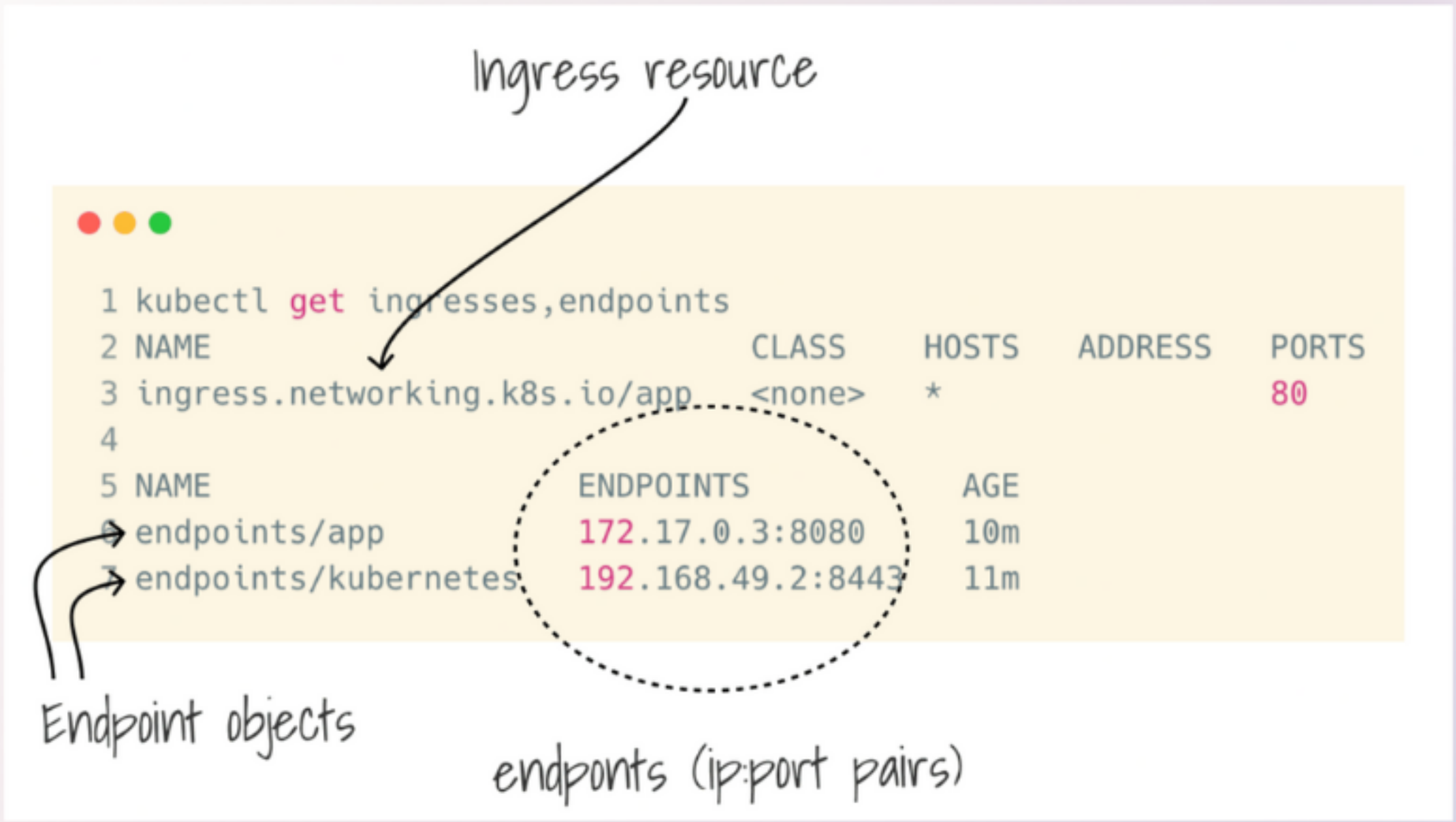
Daniele Polencic
@danielepolencic

9/

In practice, this could be as simple as `kubectl get ingresses` and `kubectl get endpoints `

With this data, you have the following:

- The path of the ingress manifest
- All the endpoints that should receive traffic



7:48 PM · 16 Jan, 2023

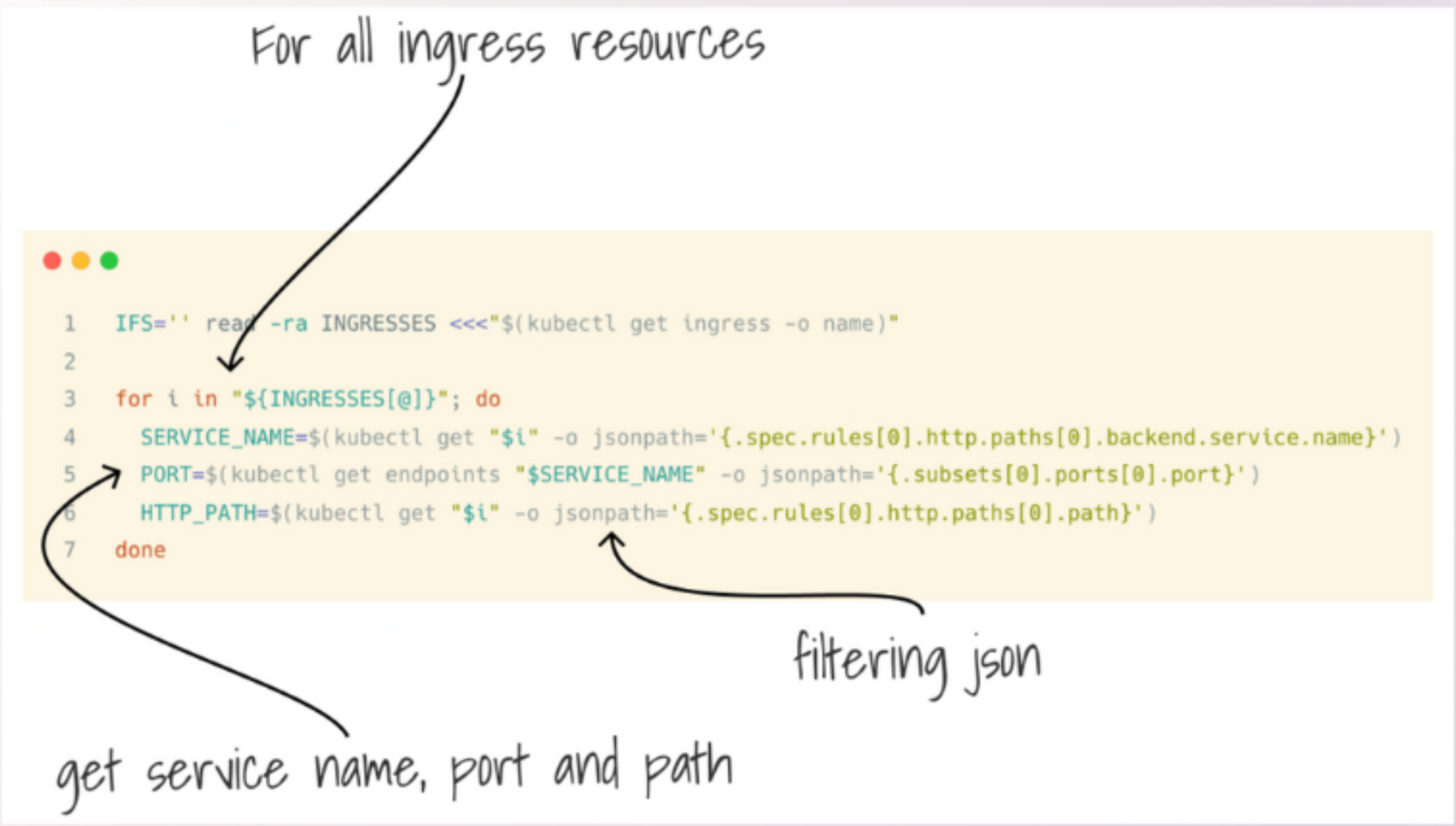
1 reply **1** likes

Daniele Polencic
@danielepolencic

10/

With `kubectl get ingresses`, you can get all the ingress manifest and loop through them

I used `-o jsonpath` to filter the rules and retrieve: the path, and the backend service



7:48 PM · 16 Jan, 2023

1 reply **2** likes

Daniele Polencic
@danielepolencic

11/

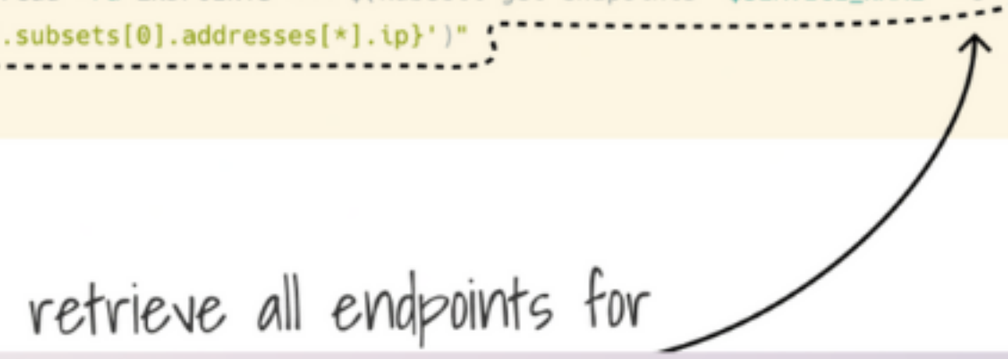
With `kubectl get endpoint `, you can retrieve all the endpoints (ip:port pair) for a service

Even in this case, I used `-o jsonpath` to filter those down and save them in a bash array

```
IFS='' read -ra INGRESSES <<<"$(kubectl get ingress -o name)"

for i in "${INGRESSES[@]}"; do
  SERVICE_NAME=$(kubectl get "$i" -o jsonpath='{.spec.rules[0].http.paths[0].backend.service.name}')
  PORT=$(kubectl get endpoints "$SERVICE_NAME" -o jsonpath='{.subsets[0].ports[0].port}')
  HTTP_PATH=$(kubectl get "$i" -o jsonpath='{.spec.rules[0].http.paths[0].path}')

  IFS='' read -ra ENDPOINTS <<<"$(kubectl get endpoints "$SERVICE_NAME" -o
jsonpath='{.subsets[0].addresses[*].ip}')"
done
```



7:48 PM · 16 Jan, 2023

1 reply **4** likes

Daniele Polencic
@danielepolencic

12/

At this point, you can use the data to reconfigure the ingress controller

In my experiment, I used Nginx, so I just wrote a template for the nginx.conf and hot-reloaded the server

template

```
upstream app {
    server <INSERT ENDPOINTS HERE>;
}


server {

    listen 80;
    location <INSERT PATH HERE> {
        proxy_pass http://app;
    }

}
```

rendered template

```
upstream app {
    server 172.17.0.3:8080;
}


server {

    listen 80;
    location / {
        proxy_pass http://app;
    }

}
```

7:48 PM · 16 Jan, 2023

1 reply **1** likes

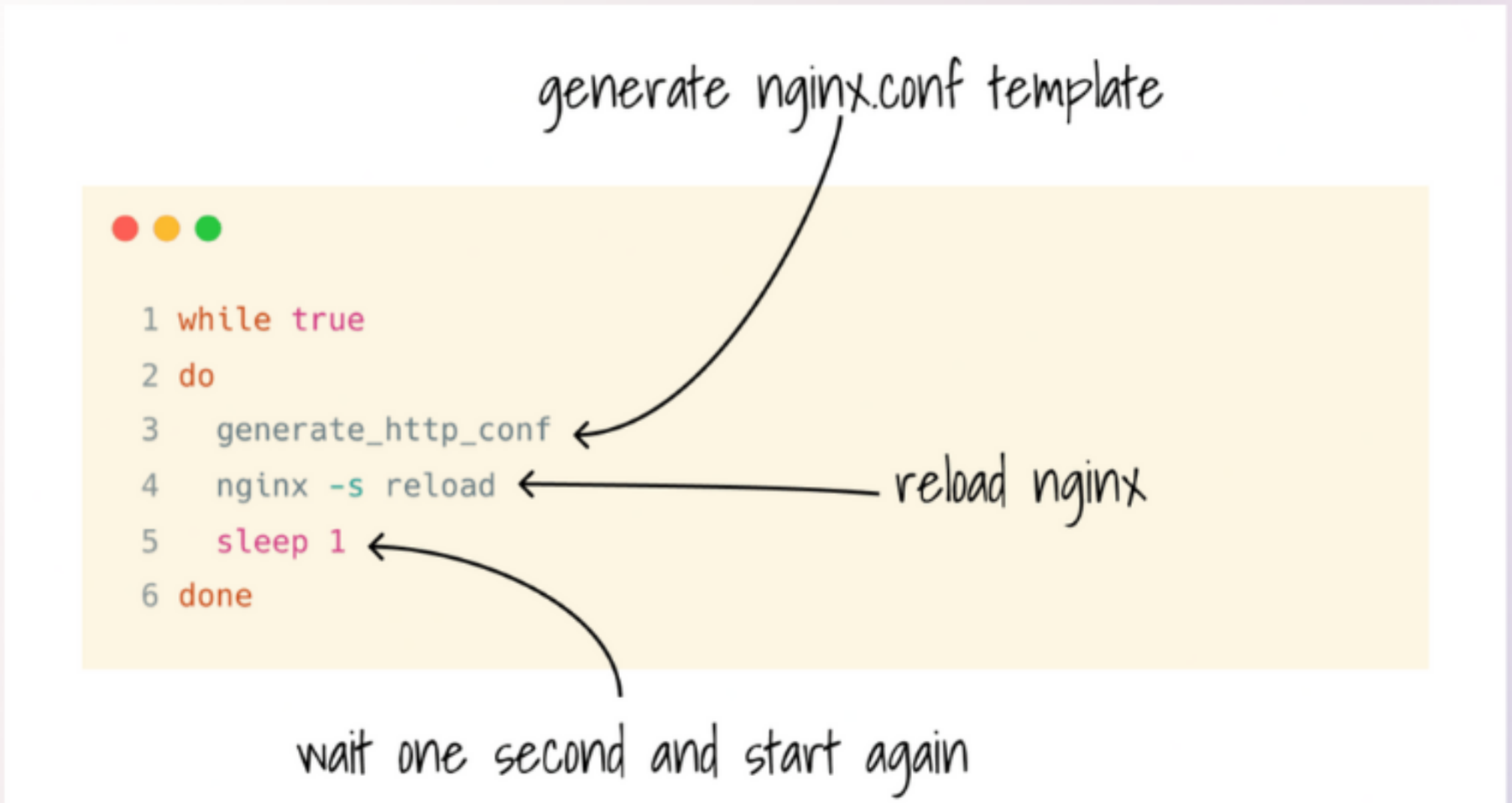
Daniele Polencic
@danielepolencic

13/

In my script, I didn't bother with detecting changes

I decided to recreate the `nginx.conf` in full every second

But you can already imagine extending this to more complex scenarios



7:49 PM · 16 Jan, 2023

1 reply **1** likes

Daniele Polencic
@danielepolencic

14/

The last step was to package the script as a container and set up the proper RBAC rule so that I could consume the API endpoints from the API server

And here it is — it worked!

7:49 PM · 16 Jan, 2023

1 reply **1** likes

Daniele Polencic
@danielepolencic

15/

If you want to play with the code, you can find it here:
github.com/learnk8s/bash-...

I plan to write a longer form article on this; if you are interested, you can sign up for the Learnk8s newsletter here: learnk8s.io/newsletter

7:49 PM · 16 Jan, 2023

1 reply **10** likes

Daniele Polencic
@danielepolencic

16/

And if you are unsure what ingress controllers are out there, at @learnk8s we have put together a handy comparison:

docs.google.com/spreadsheets/d...

7:49 PM · 16 Jan, 2023

1 reply **2** shares **14** likes

Daniele Polencic
@danielepolencic

17/

And finally, if you've enjoyed this thread, you might also like the Kubernetes workshops that we run at Learnk8s learnk8s.io/training or this collection of past Twitter threads

Until next time!

7:49 PM · 16 Jan, 2023

3 replies **1** shares **12** likes