Pod = conjunto de contenedores

Nodes = VM/fierro

Cluster = set of nodes

To intercept all http requests, we can have another container before it gets to the original service.

Dynatrace has observability and alerts.

Control plane: API, scheduler, controller, etcd

Control plane sees everything in Kubernetes. Everything in Kubernetes is an API.

Scheduler and Controller see how everything is working.

En dockerhub puedo encontrar un Docker con mongoDB que tenga todo listo para funcionar

In Dockerhub , we can find a Docker with MongoDB that has all the characteristics we need from a secure MongoDb implementation.

Everything in Kubernetes is an object.

Kubernetes allow to manage containers

Operators,

Everything is YAML, everything is text

ISTIO

Manages policies., it uses the concept of services mesh, is another Pod to intercept the requests, its name is Sidecar, and the technology is Envoy. It works like a proxy

Similar to ISTIO is Linkerd, but Linkerd is easier to use.

Kubernetes usually use a service mesh

Policies are in a YAML file

Fault injection is allowed in Istio to test the services.

OBSERVABILITY

Traces and spans

User to web to auth to azure. A trace is a set of spans.

In Kubernetes I don´t assign CPU and RAM, I define the request and the limit. For example, 4GB and 1 CPU the request, and the limit may be 8GB and 3 CPUs. That is assigned for every Pod.

Open telemetry

WORKSHOP

Microk8s

$sudo microk8s enable dns

$sudo microk8s enable dashboard

$sudo microk8s enable storage

$sudo microk8s kubectl get all --all-namespaces

microk8s kubectl -n kube-system describe secret $token

ca.crt: 1123 bytes

namespace: 11 bytes

token: eyJhbGciOiJSUzI1NiIsImtpZCI6IjZrcG1hSHo2M1pBcGYyeUZvYS1PaGxySElKUFNWa2xLNGtfWVFMTUpodEEifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJtaWNyb2s4cy1kYXNoYm9hcmQtdG9rZW4iLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiZGVmYXVsdCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImNlYjc1ZWVkLWQzNmMtNGI1NS1hYmM1LTFmZjY2NjU1MzY2ZSIsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDprdWJlLXN5c3RlbTpkZWZhdWx0In0.2lEAloQFtXigGDJsRp3fbMiYrUf8NiLwU0ceCZsZSFnC2uZGRTNFdFtyqU0DYKN9PvzLgq9OBFje5GKygIA4sC\_84MQs70bPndTiSRx7Glo9qkjmnf\_VMw53AVnbKmbdhd56lkON2yd94rLhJjeI3z\_WYbuRpc38ewOhLmpSfTOU64lReKu1UvpjfUiPF7VEuiu7k\_3DS\_q27V4pYm0KZmUr\_XwG5xYG83A65V6rZiLfdB7Be37GtWC1InVK3tzju2pfbtlruoABz5958Ihr5SR6IqNuAzxViV4\_NIn641R6df\_IarGqmfE6Ss3FZToax12jTiRc42mqI0hN25\_xUg

sudo microk8s dashboard-proxy

eyJhbGciOiJSUzI1NiIsImtpZCI6IjZrcG1hSHo2M1pBcGYyeUZvYS1PaGxySElKUFNWa2xLNGtfWVFMTUpodEEifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJtaWNyb2s4cy1kYXNoYm9hcmQtdG9rZW4iLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiZGVmYXVsdCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImNlYjc1ZWVkLWQzNmMtNGI1NS1hYmM1LTFmZjY2NjU1MzY2ZSIsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDprdWJlLXN5c3RlbTpkZWZhdWx0In0.2lEAloQFtXigGDJsRp3fbMiYrUf8NiLwU0ceCZsZSFnC2uZGRTNFdFtyqU0DYKN9PvzLgq9OBFje5GKygIA4sC\_84MQs70bPndTiSRx7Glo9qkjmnf\_VMw53AVnbKmbdhd56lkON2yd94rLhJjeI3z\_WYbuRpc38ewOhLmpSfTOU64lReKu1UvpjfUiPF7VEuiu7k\_3DS\_q27V4pYm0KZmUr\_XwG5xYG83A65V6rZiLfdB7Be37GtWC1InVK3tzju2pfbtlruoABz5958Ihr5SR6IqNuAzxViV4\_NIn641R6df\_IarGqmfE6Ss3FZToax12jTiRc42mqI0hN25\_xUg

One deployment may have one or more pods

One deployment needs a service, not directly to the pod

Secrets are encrypted, they are in the deployment and can be injected as an environment variable.

we are going to join to one cluster

we will join as masters.

Microk8s add-node

microk8s join 192.168.64.2:25000/226c55716719c700e1cf8579461fa82e/91ffa260d4bd

<https://github.com/kubernetes/dashboard/blob/master/docs/user/access-control/creating-sample-user.md>

USING VSCODE

Search in ChatGPT for “give a docker file for hello world when I call by http”

dockerfile

# Use the official Python image

FROM python:3.9-slim

# Set the working directory

WORKDIR /app

# Create a simple Hello World Python script

RUN echo "from http.server import BaseHTTPRequestHandler, HTTPServer\n\

class HelloWorldHandler(BaseHTTPRequestHandler):\n\

def do\_GET(self):\n\

self.send\_response(200)\n\

self.send\_header('Content-type', 'text/plain')\n\

self.end\_headers()\n\

self.wfile.write(b'Hello, World!')\n\

\n\

def run(server\_class=HTTPServer, handler\_class=HelloWorldHandler):\n\

server\_address = ('', 8000)\n\

httpd = server\_class(server\_address, handler\_class)\n\

print('Starting httpd...')\n\

httpd.serve\_forever()\n\

\n\

if \_\_name\_\_ == '\_\_main\_\_':\n\

run(HTTPServer, HelloWorldHandler)" > hello.py

# Expose port 8000

EXPOSE 8000

# Run the Python script

CMD ["python", "hello.py"]

Build docker image

docker build -t hello-world-node .

Run

docker run -p 8000:8000 hello-world-python

Open

<http://localhost:8000>