PyBay 2024

Deploying Python Apps On Kubernetes

PyBay 2024

Senthil Kumaran

AWS EKS , CPython , Kubernetes


python™
kubernetes

https://github.com/orsenthil

In the Beginning ...

Borg, Omega, and Kubernetes

LESSONS LEARNED FROM THREE CONTAINER-MANAGEMENT SYSTEMS OVER A DECADE

BRENDAN BURNS,
BRIAN GRANT,
DAVID OPPENHEIMER,
ERIC BREWER, AND
JOHN WILKES,
GOOGLE INC.

Though widespread interest in software containers is a relatively recent phenomenon, at Google we have been managing Linux containers at scale for more than ten years and built three different container-management systems in that time. Each system was heavily influenced by its predecessors, even though they were developed for different reasons. This article describes the lessons we've learned from developing and operating them.

Read the Paper

there was

Search Engine in Python

A tiny search engine in python following the guide https://www.alexmolas.com/2024/02/05/a-search-engine-in-80-lines.html

Provide an index of links to crawl.

```
cat > feeds.txt <<EOF
http://bair.berkeley.edu/blog/feed.xml
http://benanne.github.io/feed.xml
https://simonwillison.net/atom/entries/
https://blog.bytebytego.com/feed
https://eli.thegreenplace.net/feeds/all.atom.xml
EOF
```

https://github.com/orsenthil/search

Crawl the feeds.txt

```
python crawler.py --feed-path feeds.txt
```

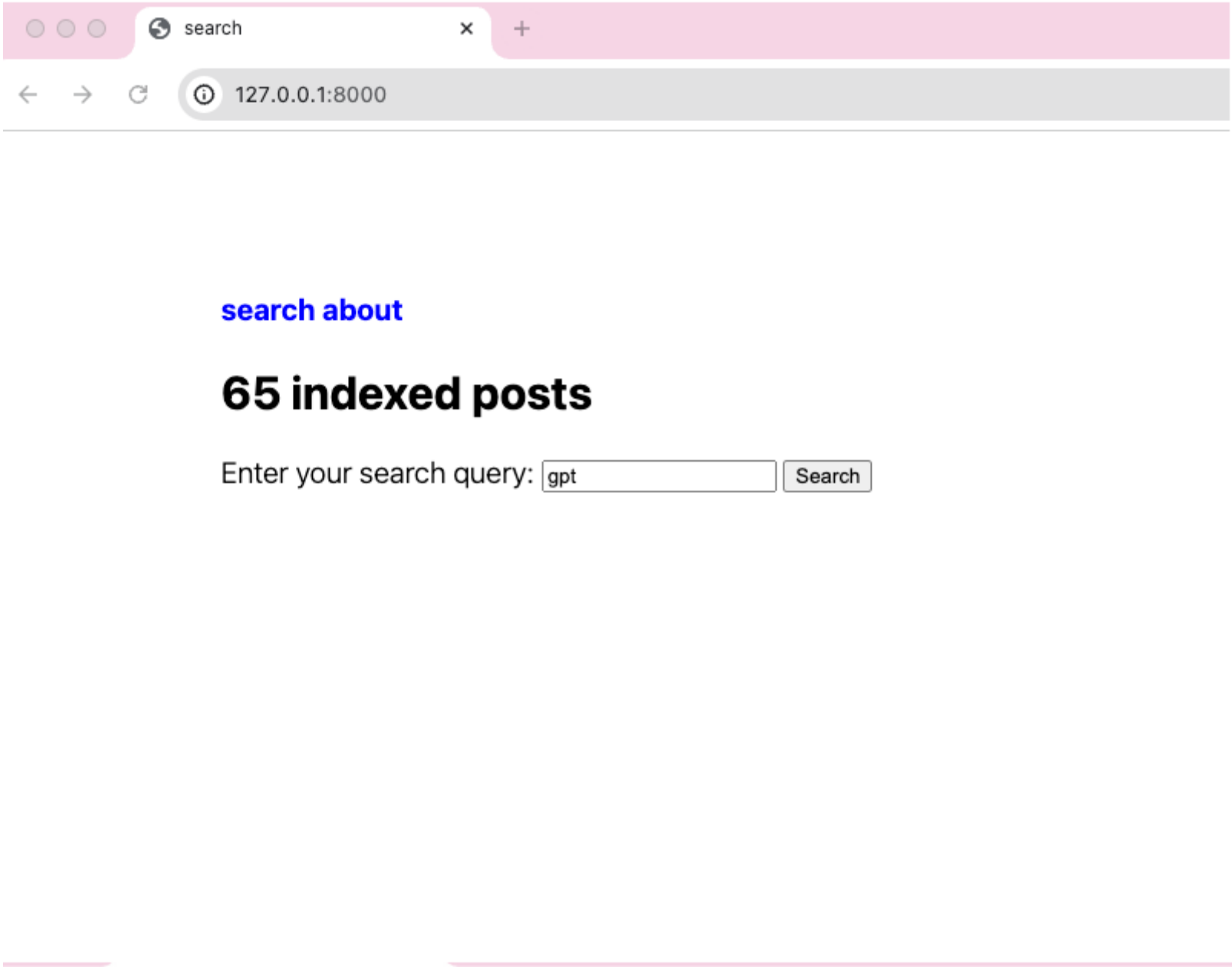
This will create a file output.parquet, which is the parquet format

Search the index

```
python main.py --data-path output.parquet
```

```
INFO:     Started server process [27449]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:51026 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:51041 - "GET /results/gpt HTTP/1.1" 200 OK
```

Search Interface



search about

Search Results - gpt

- https://simonwillison.net/2024/Jun/27/ai-worlds-fair/#atom-entries - Score: 3.2402747996700194
- https://simonwillison.net/2024/May/15/chatgpt-in-4o-mode/#atom-entries - Score: 3.177576264568573
- https://simonwillison.net/2024/May/28/weeknotes/#atom-entries - Score: 3.152774293286982
- http://bair.berkeley.edu/blog/2023/10/16/p3o/ - Score: 3.028567529711895
- https://simonwillison.net/2024/Apr/23/weeknotes/#atom-entries - Score: 2.9188686621396753

Namespaces
Containers
Pods

Deployments
Services
Ingress

Volumes
ConfigMaps
Secrets

```yaml
apiVersion: v1
kind: Pod
metadata:
  namespace: app-namespace
  name: curl-pod
spec:
  containers:
  - name: curl-container
    image: curlimages/curl
    command: ['sh', '-c', 'while true; do sleep 30; done;']
```

```
kubectl exec -it curl-pod --n app-namespace -- /bin/sh
```

Search Engine in Python

A tiny search engine in python following the guide https://www.alexmolas.com/2024/02/05/a-search-engine-in-80-lines.html

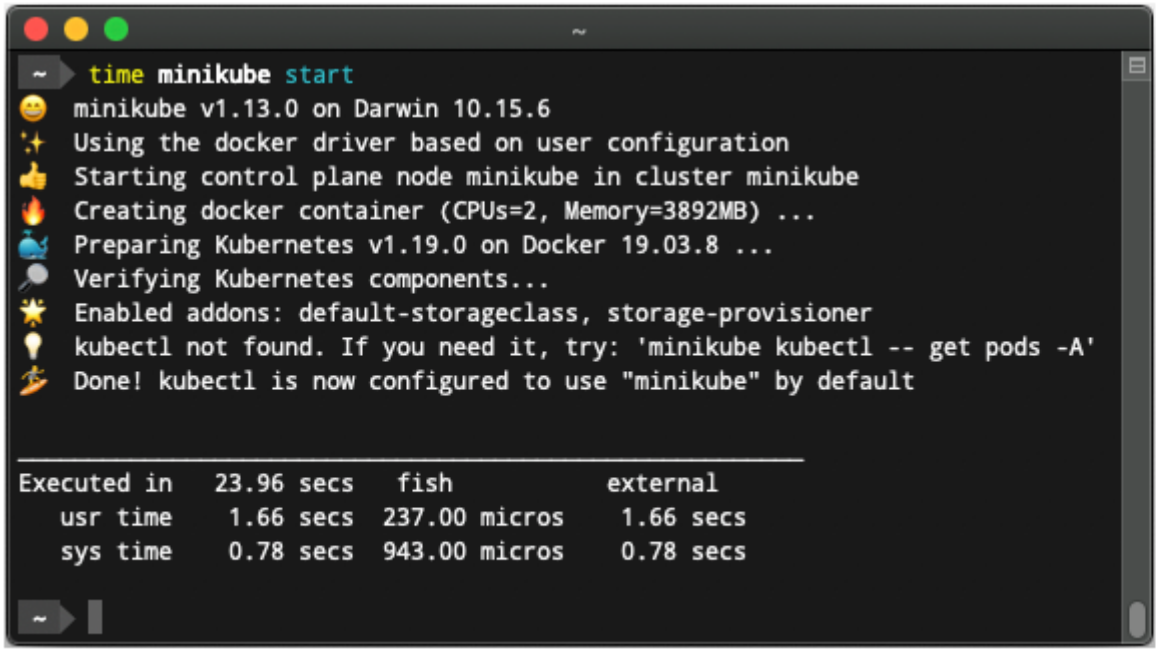
Provide an index of links to crawl.

```
cat > feeds.txt <<EOF
http://bair.berkeley.edu/blog/feed.xml
http://benanne.github.io/feed.xml
https://simonwillison.net/atom/entries/
https://blog.bytebytego.com/feed
https://eli.thegreenplace.net/feeds/all.atom.xml
EOF
```

https://github.com/orsenthil/search

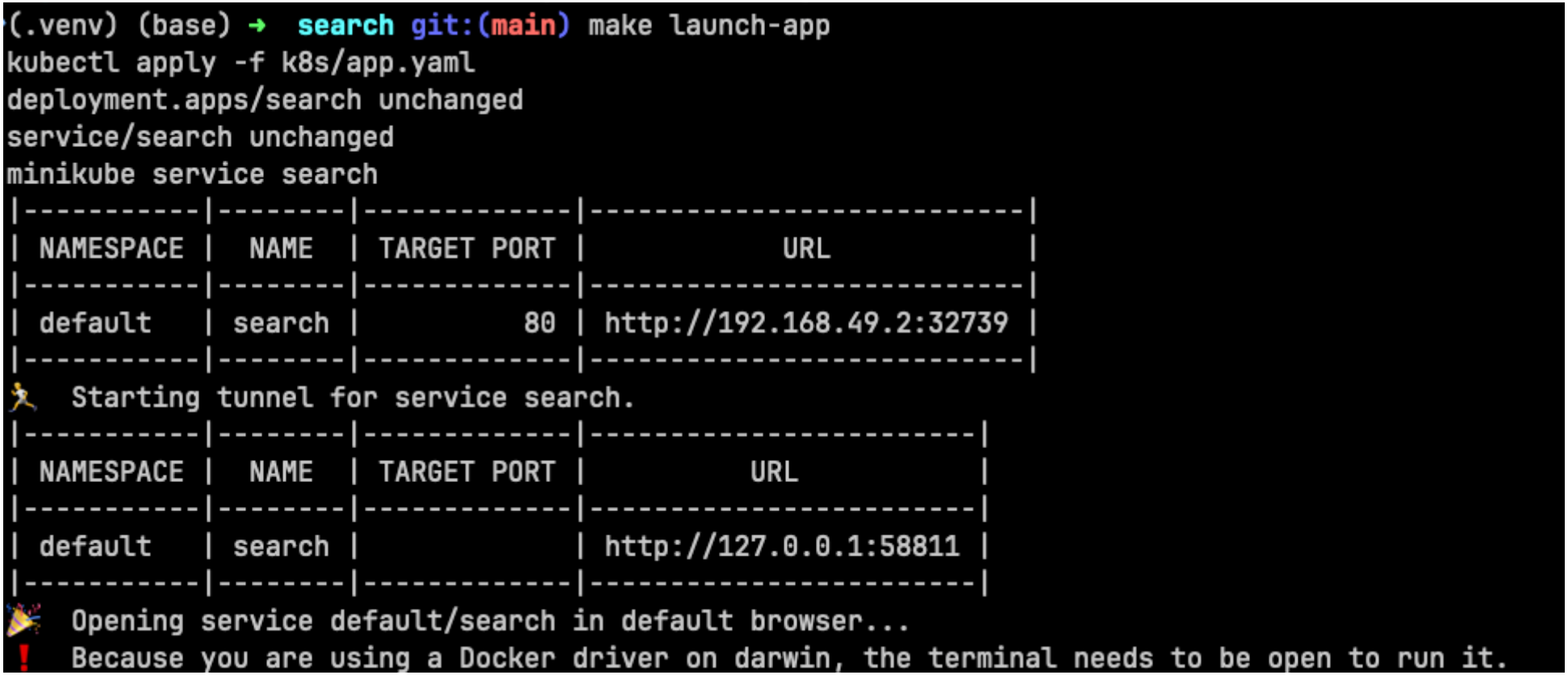
minikube

minikube implements a local Kubernetes cluster on macOS, Linux, and Windows. minikube's primary goals are to be the best tool for local Kubernetes application development and to support all Kubernetes features that fit.

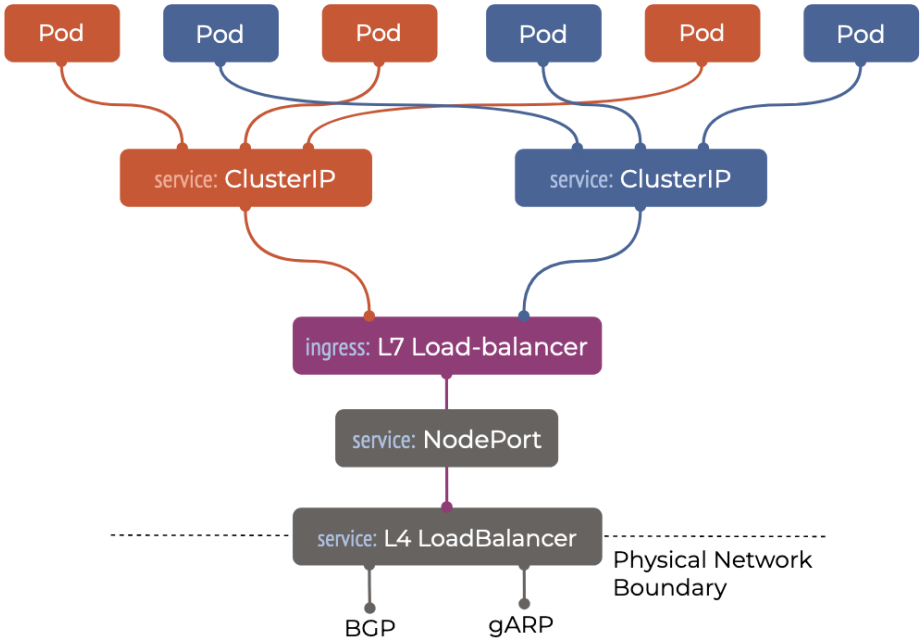


```yaml
# Kubernetes Deployment and Service configuration
apiVersion: apps/v1
kind: Deployment
metadata:
  name: search
  labels:
    app: search
spec:
  replicas: 1
  selector:
    matchLabels:
      app: search
  template:
    metadata:
      labels:
        app: search
    spec:
      containers:
      - name: search
        image: skumaran/search:latest
        ports:
        - containerPort: 80
        imagePullPolicy: Always
        resources:
          requests:
            memory: "64Mi"
            cpu: "250m"
          limits:
            memory: "128Mi"
            cpu: "500m"
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: search
spec:
  selector:
    app: search
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```


(.venv) (base) → search git:(main) make launch-app
kubectl apply -f k8s/app.yaml
deployment.apps/search unchanged
service/search unchanged
minikube service search
| NAMESPACE | NAME | TARGET PORT | URL |
| default | search | 80 | http://192.168.49.2:32739 |
Starting tunnel for service search.
| NAMESPACE | NAME | TARGET PORT | URL |
| default | search | | http://127.0.0.1:58811 |
Opening service default/search in default browser...
Because you are using a Docker driver on darwin, the terminal needs to be open to run it.

Kubernetes Networking Model



https://www.tkng.io/arch/

Flask

```python
from flask import Flask
app = Flask(__name__)


@app.route("/")
def hello():
    return "hello from flask!"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80)
```

https://github.com/JasonHaley/hello-python

Dockerfile

```
FROM python:3.11

RUN mkdir /app
WORKDIR /app
ADD . /app
RUN pip install -r requirements.txt

EXPOSE 80

CMD ["python", "/app/main.py"]
```

```yaml
---
apiVersion: v1
kind: Namespace
metadata:
  name: python-namespace
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: python-namespace
  name: python-app
spec:
  selector:
    matchLabels:
      app: hello-python
  replicas: 4
  template:
    metadata:
      labels:
        app: hello-python
    spec:
      containers:
      - name: hello-python
        image: skumaran/hello-python:latest
        imagePullPolicy: Always
        resources:
          limits:
            cpu: "1"
            memory: "512Mi"
        ports:
        - containerPort: 80
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: hello-python-service
spec:
  selector:
    app: hello-python
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 80
  type: LoadBalancer
```

https://github.com/mukulmantosh/django-kubernetes

Gunicorn as serving layer for Django
Psycopg 2 as Adapter for PostGres DataBase
Nginx as frontend Proxy

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-cm
  namespace: django-app
data:
  DB_HOST: "postgres-service"
  DB_USERNAME: "sampleuser"
  DB_PASSWORD: "sampleuser123"
  DB_NAME: "django-app"
  DB_PORT: "5432"
  STATIC_ROOT: "/data/static"
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: app-service
  namespace: django-app
  labels:
    app: app-svc
spec:
  ports:
    - port: 8000
      targetPort: 8000
  selector:
    app: django-application
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: django-app-deploy
  namespace: django-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: django-application
  template:
    metadata:
      labels:
        app: django-application
    spec:
      volumes:
        - name: staticfiles
          persistentVolumeClaim:
            claimName: staticfiles-pvc
      containers:
        - image: skumaran/django-kubernetes:1.0
          imagePullPolicy: Always
          name: django-app-container
          envFrom:
            - configMapRef:
                name: app-cm
          ports:
            - containerPort: 8000
          resources:
            limits:
              cpu: "1"
              memory: "512Mi"
          volumeMounts:
            - mountPath: "/data/static"
              name: staticfiles
```

```
name: staticfiles
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: staticfiles-pv
  labels:
    type: local
    app: django-staticfiles
spec:
  storageClassName: manual
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/data/static"
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: staticfiles-pvc
  namespace: django-app
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  volumeName: staticfiles-pv
```

---

postgres database used with django

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-secret-credentials
  namespace: django-app
  labels:
    app: postgres
data:
  POSTGRES_DB: "django-app"
  POSTGRES_USER: "sampleuser"
  POSTGRES_PASSWORD: "sampleuser123"
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
  namespace: django-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgresdb
  template:
    metadata:
      labels:
        app: postgresdb
    spec:
      containers:
        - name: postgresdb
          image: postgres:16.0
          ports:
            - containerPort: 5432
          envFrom:
            - configMapRef:
                name: db-secret-credentials
          volumeMounts:
            - mountPath: /var/lib/postgres/data
              name: db-data
      volumes:
        - name: db-data
          persistentVolumeClaim:
            claimName: postgres-pvc
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: postgres-pv
  labels:
    type: local
    app: postgres
spec:
  storageClassName: manual
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/data/db"
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-pvc
  namespace: django-app
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  volumeName: postgres-pv
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
  namespace: django-app
  labels:
    app: postgres-svc
spec:
  type: NodePort
  ports:
    - port: 5432
      targetPort: 5432
      nodePort: 30004
  selector:
    app: postgresdb
```

---

kubernetes jobs

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: django-db-migrations
  namespace: django-app
spec:
  ttlSecondsAfterFinished: 100
  activeDeadlineSeconds: 60

  template:
    spec:
      containers:
        - name: migration-container
          image: skumaran/django-kubernetes:1.0
          command: ['python', 'manage.py', 'migrate']
          imagePullPolicy: Always
          envFrom:
            - configMapRef:
                name: app-cm
          ports:
            - containerPort: 8000
      restartPolicy: OnFailure
  backoffLimit: 15
```

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: django-staticfiles
  namespace: django-app
spec:
  ttlSecondsAfterFinished: 100
  activeDeadlineSeconds: 60

  template:
    spec:
      volumes:
        - name: staticfiles
          persistentVolumeClaim:
            claimName: staticfiles-pvc
      containers:
        - name: staticfiles-container
          image: skumaran/django-kubernetes:1.0
          command: ['python', 'manage.py', 'collectstatic', '--noinput']
          imagePullPolicy: Always
          envFrom:
            - configMapRef:
                name: app-cm
          ports:
            - containerPort: 8000
          volumeMounts:
            - mountPath: "/data/static"
              name: staticfiles
      restartPolicy: OnFailure
  backoffLimit: 3
```

Ingress

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  namespace: django-app
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx-service
                port:
                  number: 80
```

http://k8s-djangoap-nginxing-ad74b811e3-1604513598.us-west-2.elb.amazonaws.com/

django

View release notes for Django 4.2

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Point your Domain to CNAME of the ALB
Get a Certificate
Create an Ingress with HTTPS

```yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: orsenthil@gmail.com
    privateKeySecretRef:
      name: letsencrypt-prod-account-key
    solvers:
    - http01:
        ingress:
          class: alb
```

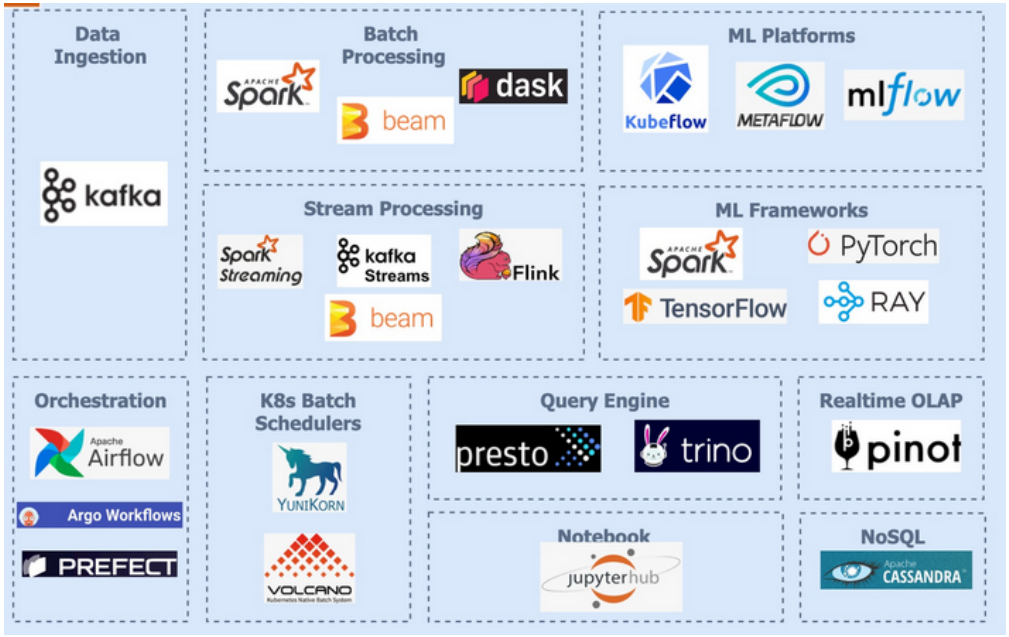
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress-tls
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'
    alb.ingress.kubernetes.io/ssl-redirect: '443'
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
spec:
  tls:
  - hosts:
    - django.learntosolveit.com
    secretName: nginx-tls-cert
  rules:
  - host: django.learntosolveit.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx-service
            port:
              number: 80
```

You have working starter project Django app.
Add Features
Rinse and Repeat

Handle Scale using Horizontal AutoScaler

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: my-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: django-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
    name: cpu
    targetAverageUtilization: 80
  behavior:
    scaleUp:
    stabilizationWindowSeconds: 60
    policies:
    - type: Percent
      value: 10
      periodSeconds: 60
    scaleDown:
    stabilizationWindowSeconds: 60
    policies:
    - type: Percent
```

```
      value: 10
      periodSeconds: 60
```



https://awslabs.github.io/data-on-eks/docs/introduction/intro

---

Verify Node Pools with GPU and Nvidia Device Plugin

Verify the Karpenter autosclaer Nodepools

```
kubectl get nodepools
```

```
NAME                NODECLASS
g5-gpu-karpenter    g5-gpu-karpenter
x86-cpu-karpenter   x86-cpu-karpenter
```

Verify the NVIDIA Device plugin

```
kubectl get pods -n nvidia-device-plugin
```

```
NAME                                                              READY   STATUS    RESTARTS   AGE
nvidia-device-plugin-gpu-feature-discovery-b4clk                  1/1     Running   0          3h13m
nvidia-device-plugin-node-feature-discovery-master-568b49722ldt   1/1     Running   0          9h
nvidia-device-plugin-node-feature-discovery-worker-clk9b          1/1     Running   0          3h13m
nvidia-device-plugin-node-feature-discovery-worker-cwg28          1/1     Running   0          9h
nvidia-device-plugin-node-feature-discovery-worker-ng52l          1/1     Running   0          9h
nvidia-device-plugin-p56jj                                        1/1     Running   0          3h13m
```

Verify Kuberay Operator which is used to create Ray Clusters

```
kubectl get pods -n kuberay-operator
```

```
NAME                                READY   STATUS    RESTARTS   AGE
kuberay-operator-7894df98dc-447pm   1/1     Running   0          9h
```

Mistral 7B Inference Model from HuggingFace

https://github.com/awslabs/data-on-eks/tree/main/gen-ai/inference/vllm-rayserve-gpu

---

Run the following command to verify the services:

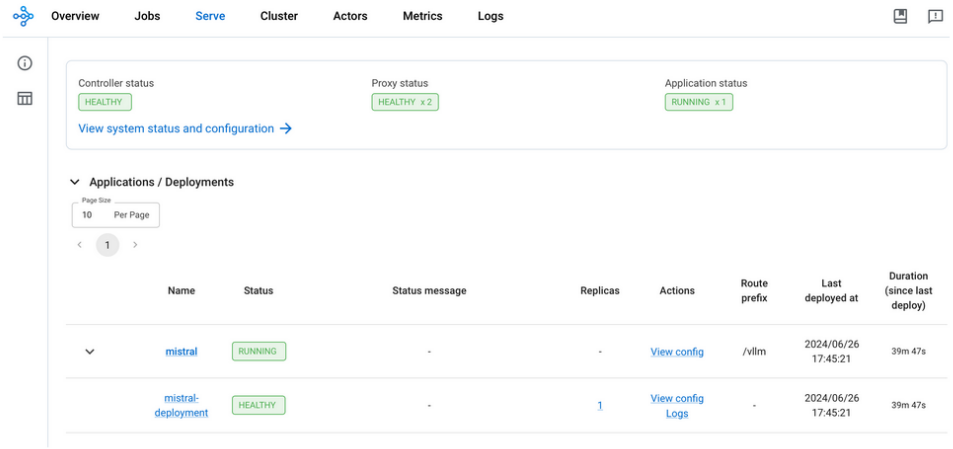
```
kubectl get svc -n rayserve-vllm

NAME             TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                                        AGE
vllm             ClusterIP   172.20.208.16    <none>        6379/TCP,8265/TCP,10001/TCP,8000/TCP,8080/TCP  48m
vllm-head-svc    ClusterIP   172.20.239.237   <none>        6379/TCP,8265/TCP,10001/TCP,8000/TCP,8080/TCP  37m
vllm-serve-svc   ClusterIP   172.20.196.195   <none>        8000/TCP                                       37m
```

To access the Ray dashboard, you can port-forward the relevant port to your local machine:

```
kubectl -n rayserve-vllm port-forward svc/vllm 8265:8265
```

You can then access the web UI at http://localhost:8265, which displays the deployment of jobs and actors within the Ray ecosystem.



Testing Mistral-7b Chat Model

Now it's time to test the Mistral-7B chat model. We'll use a Python client script to send prompts to the RayServe inference endpoint and verify the outputs generated by the model. The script reads prompts from a `prompts.txt` file and writes the responses to a `results.txt` file in the same location. It also logs the response times and token lengths for each response.

First, execute a port forward to the `vllm-serve-svc` Service using kubectl:

```
kubectl -n rayserve-vllm port-forward svc/vllm-serve-svc 8000:8000
```

Prompt: [INST] Explain the theory of relativity.
Response: [INST] Explain the theory of relativity. [/INST] The theory of relativity, developed by Albert Einstein, is a fundamental theory in physics that describes the relationship between space and time, and how matter and energy interact within that framework. It is actually composed of two parts: the Special Theory of Relativity, published in 1905, and the General Theory of Relativity, published in 1915.

The Special Theory of Relativity is based on two postulates: the first one states that the laws of physics are the same in all inertial frames of reference (frames that are not accelerating); the second one asserts that the speed of light in a vacuum is the same for all observers, regardless of their motion or the source of the light.

From these two postulates, several counter-intuitive consequences follow. For example, the length of an object contracts when it is in motion relative to an observer, and time dilation occurs, meaning that a moving clock appears to tick slower than a stationary one. These phenomena have been confirmed by numerous experiments.

The General Theory of Relativity is a theory of gravitation, which extended the Special Theory of Relativity by incorporating gravity into the fabric of spacetime. In this theory, mass causes a distortion or curvature in spacetime, which is felt as a gravitational force. This is in contrast to the Newtonian view of gravity as a force acting at a distance between two masses.

One of the most famous predictions of General Relativity is the bending of light by gravity, which was first observed during a solar eclipse in 1919. The theory has been extremely successful in explaining various phenomena, such as the precession of Mercury's orbit, the gravitational redshift of light, and the existence of black holes and gravitational waves.

In summary, the theory of relativity is a groundbreaking theory in physics that fundamentally changed our understanding of space, time, and matter. It has been incredibly successful in making accurate predictions about the natural world and has stood the test of time through numerous experiments and observations.

http://github.com/orsenthil

That's all folks!