Architecting and Running Distributed Python Applications with KubeRay

Anmol Krishan Sachdeva
(@greatdevaks)
Hybrid Cloud Architect, Google

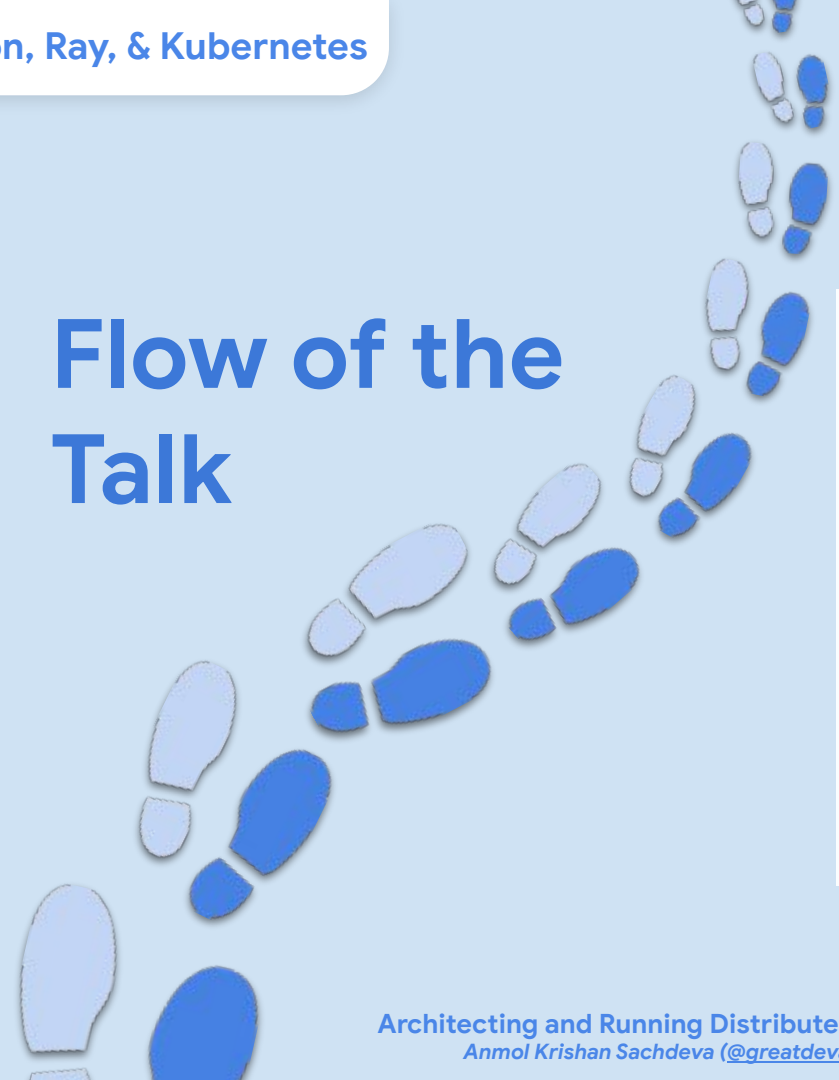
tinyurl.com/pygeekle-python-kuberay

Anmol Krishan Sachdeva
Hybrid Cloud Architect, Google
MSc Advanced Computing
University of Bristol, United Kingdom
LinkedIn: greatdevaks
Twitter: @greatdevaks

- International Tech Speaker
  - KubeCon, PyCon*, EuroPython, GeoPython, Geekle, etc.
- Distinguished Guest Lecturer and Tech Panelist
- Conference Organizer
  - EuroPython, GeoPython, PyCon*, etc.
- Represented India at reputed International Hackathons
- Deep Learning Researcher
- Publications at International Journals
- ALL STACK DEVELOPER
- Mentor

Disclaimer

The content and the views presented during the talk/session are the author's own and not of the organizations/companies they are associated with.



Flow of the Talk

- Ray Framework
- Ray Clusters
- KubeRay
- Ray Autoscaler
- Cluster Walkthrough
- Ray Jobs

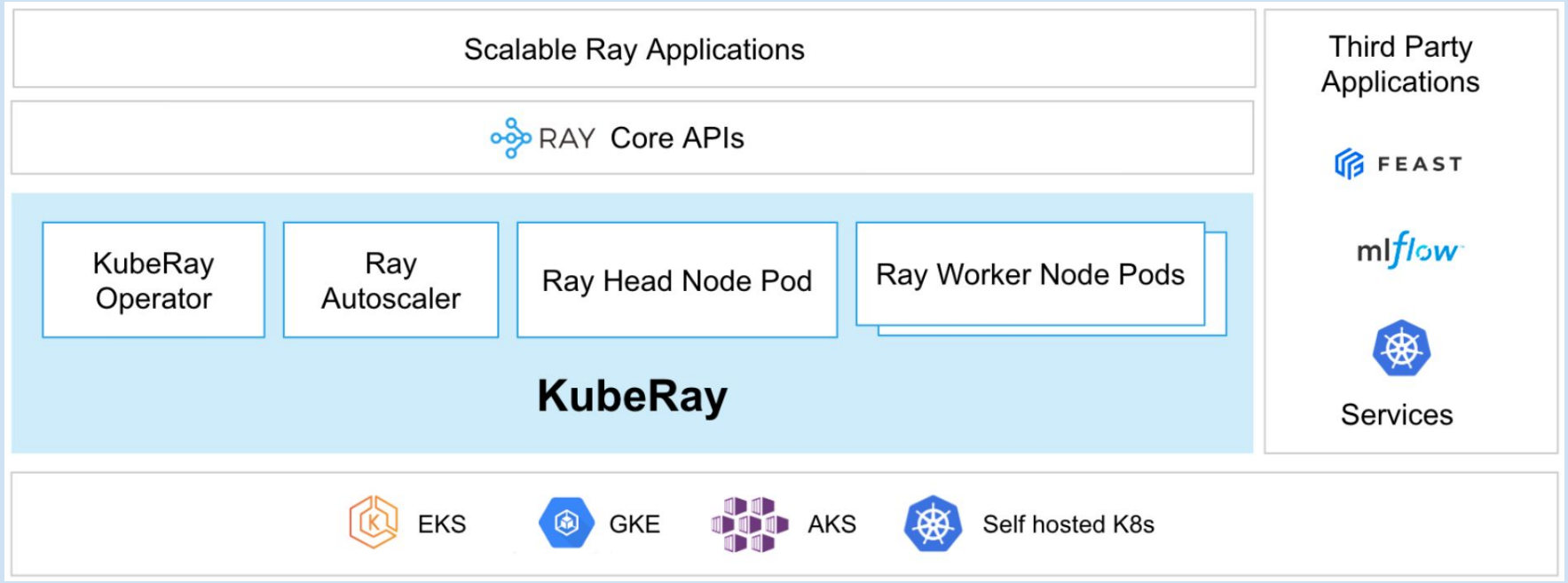
Ray Framework

- Created at RISELab at UC Berkeley
- A general purpose distributed computing and execution framework
- Python-first; parallelize Python programs flexibly, without rewrites
  - Parallelization => More Speed
- Cluster Management capabilities; multi-datacenter support
- Great integrations with popular Data Science tools
- Supports high performance and heterogeneous workloads
  - Some tasks may require GPU and some work well with CPU
- Dynamic execution that works amazingly with task dependencies

The Big Picture: Ray Cluster

PyCon Colombia Talk Part 1

Ray on Kubernetes: KubeRay



Deploying KubeRay

```
# This creates the KubeRay operator and all of the resources it needs.
! kubectl create -k "github.com/ray-project/kuberay/ray-operator/config/default?ref=v0.3.0&timeout=90s"

# Note that we must use "kubectl create" in the above command. "kubectl apply" will not work due to
https://github.com/ray-project/kuberay/issues/271

# You may alternatively clone the KubeRay GitHub repo and deploy the operator's configuration from your local
file system.
```

Verify KubeRay Operator is Running

```
! kubectl get deployments -n ray-system

# NAME           READY   UP-TO-DATE   AVAILABLE   AGE
# ray-operator   1/1     1            1           40s

! kubectl -n ray-system get pod --selector=app.kubernetes.io/component=kuberay-operator

# NAME                              READY   STATUS    RESTARTS   AGE
# ray-operator-75dbbf8587-5lrvn   1/1     Running   0          31s
```

Deploying a Ray Cluster

```
# Deploy a sample Ray Cluster CR from the KubeRay repo:
! kubectl apply -f https://raw.githubusercontent.com/ray-project/kuberay/release-0.3/ray-operator/config/samples/ray-cluster.autoscaler.yaml

# This Ray cluster is named `raycluster-autoscaler` because it has optional Ray Autoscaler support enabled.
```

```
! kubectl get raycluster

# NAME                    AGE
# raycluster-autoscaler   XXs

# View the pods in the Ray cluster named "raycluster-autoscaler"
! kubectl get pods --selector=ray.io/cluster=raycluster-autoscaler

# NAME                                          READY   STATUS    RESTARTS   AGE
# raycluster-autoscaler-head-xxxxx              2/2     Running   0          XXs
# raycluster-autoscaler-worker-small-group-yyyyy   1/1     Running   0          XXs
```

Ray Cluster Autoscaler Configuration

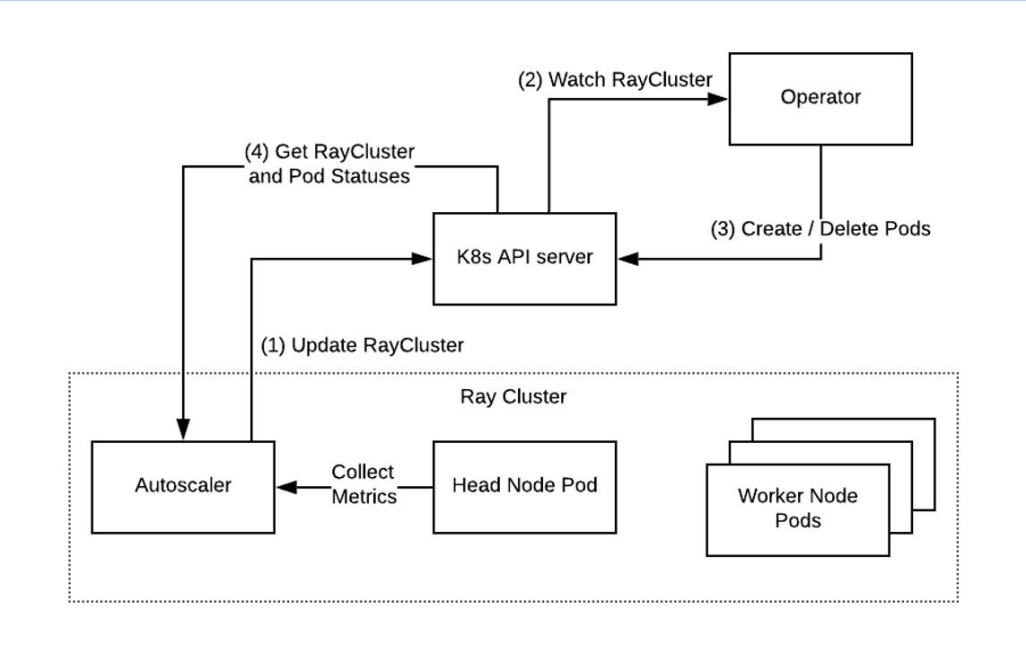
Ray Job Execution

```
# Substitute your output from the last cell in place of "raycluster-autoscaler-head-xxxxx"

! kubectl exec raycluster-autoscaler-head-xxxxx -it -c ray-head -- python -c "import ray; ray.init()"
# 2022-08-10 11:23:17,093 INFO worker.py:1312 -- Connecting to existing Ray cluster at address: <IP
address>:6379...
# 2022-08-10 11:23:17,097 INFO worker.py:1490 -- Connected to Ray cluster.
```

Ray Job Submission

```
! kubectl get service raycluster-autoscaler-head-svc

# NAME                              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)                       AGE
# raycluster-autoscaler-head-svc    ClusterIP   10.96.114.20   <none>        6379/TCP,8265/TCP,10001/TCP   XXs

# Execute this in a separate shell.
! kubectl port-forward service/raycluster-autoscaler-head-svc 8265:8265

# The following job's logs will show the Ray cluster's total resource capacity, including 3 CPUs.

! ray job submit --address http://localhost:8265 -- python -c "import ray; ray.init();
print(ray.cluster_resources())"
```

Autoscaling



Thanks Everyone !

tinyurl.com/pygeekle-python-kuberay

#Geekle #PyGeekle
#Ray #Kubernetes
#PythonSummit
#KubeRay
@greatdevaks

Anmol Krishan Sachdeva

Hybrid Cloud Architect, Google
MSc Advanced Computing, University of Bristol, UK
LinkedIn | Twitter *(@greatdevaks)*