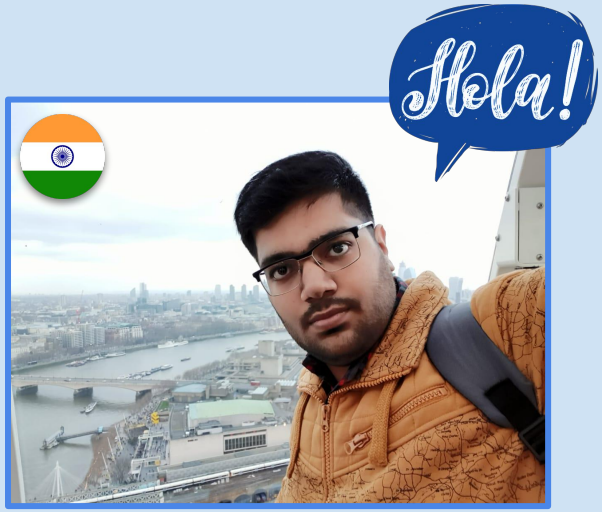

PYCON 22
COLOMBIA
Architecting and Running Distributed Python Applications with Ray on Kubernetes
Anmol Krishan Sachdeva
(@greatdevaks)
Hybrid Cloud Architect, Google
tinyurl.com/pycon-colombia-python-ray
RAY



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

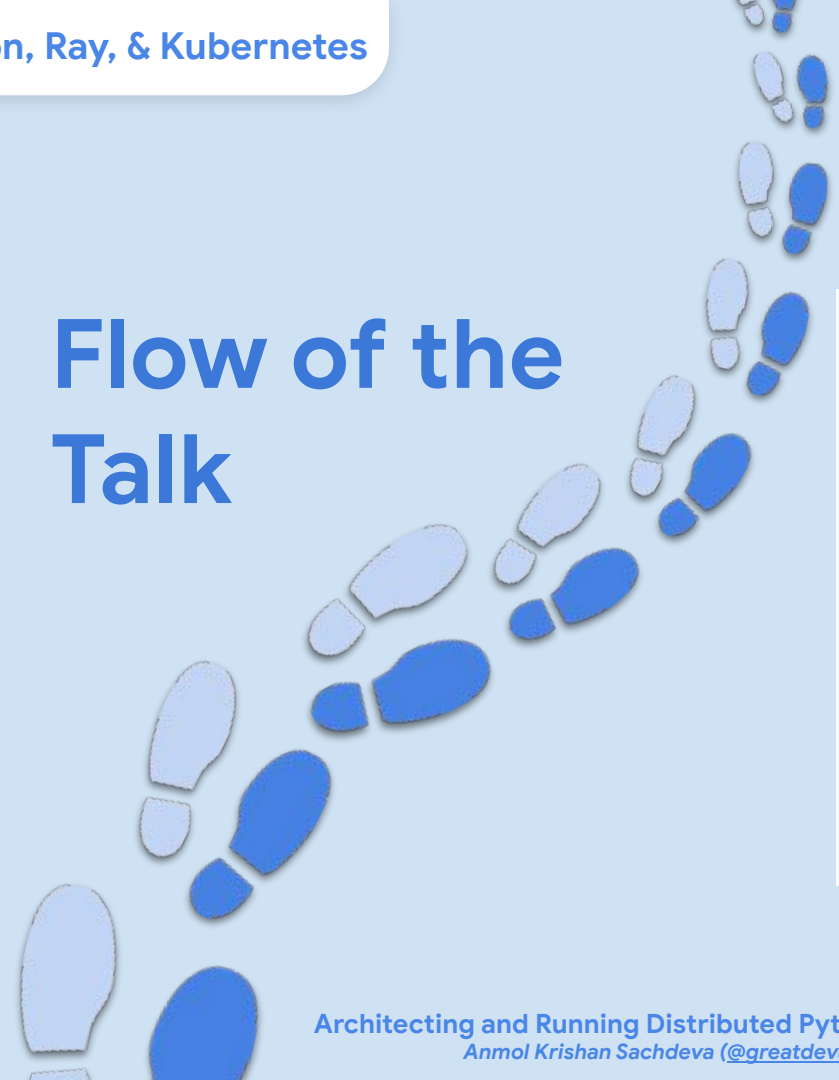
PYCON 22 COLOMBIA

- 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
- Internals and Architecture
- Building a Distributed Ray App
- Ray Autoscaler
- Deploying Ray App on K8s
- Q/A

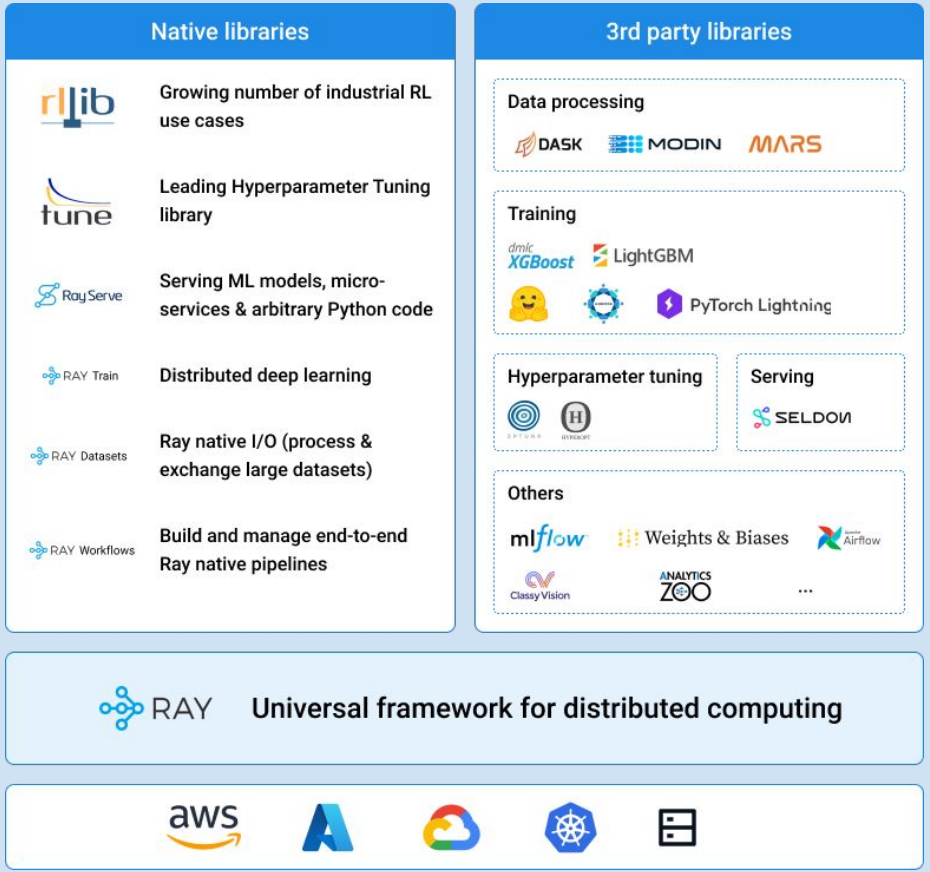
Ray Framework

- Created at RISELab at UC Berkeley
- A general purpose distributed computing and execution framework
- Python-first; parallelize Python programs flexibly
- Cluster Management capabilities
- 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

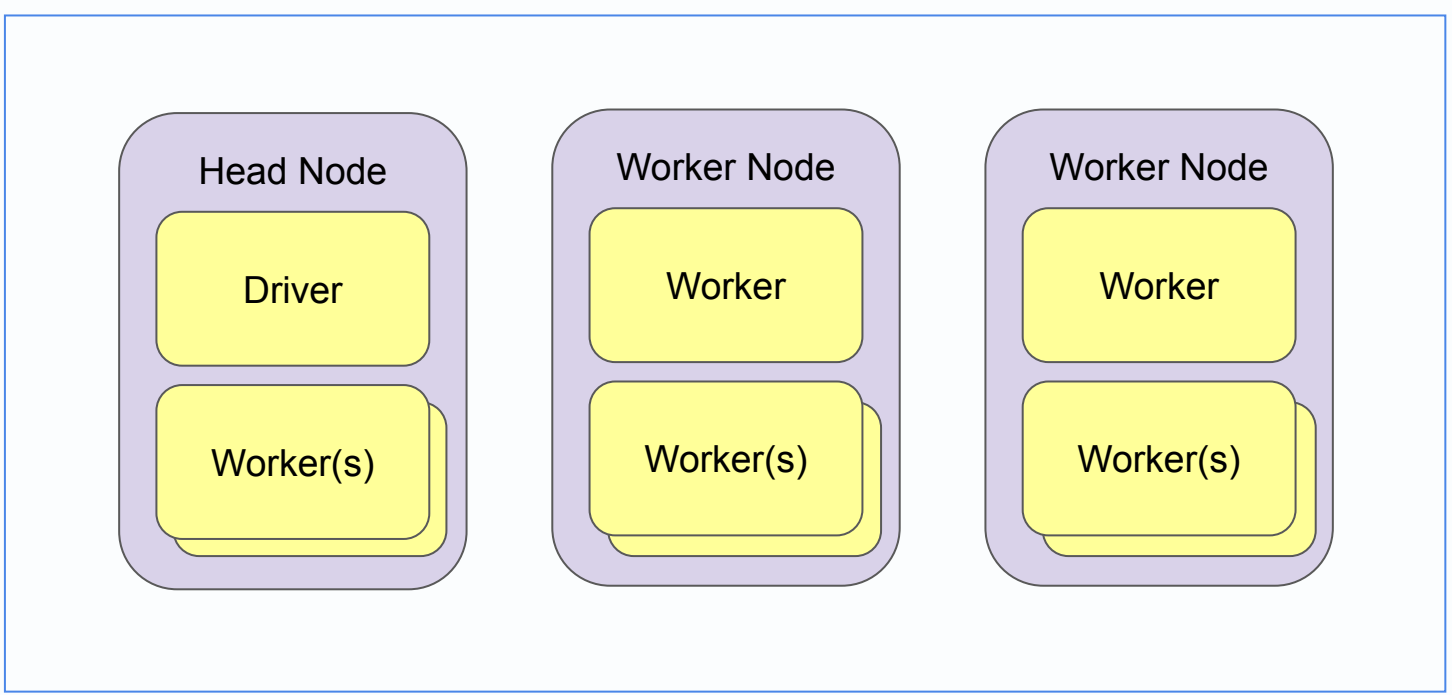
Why Ray?

- Single server is not enough for many distributed tasks
- Parallelizing existing Python code may require complete rewrite
- Fault Tolerance and High Availability of tasks
  - Multiple machines?
  - Multiple datacenters?
- Tasks, if decomposed to run in parallel, can speed computation up
- Python interpreter is effectively single threaded and ineffective for distributed computing

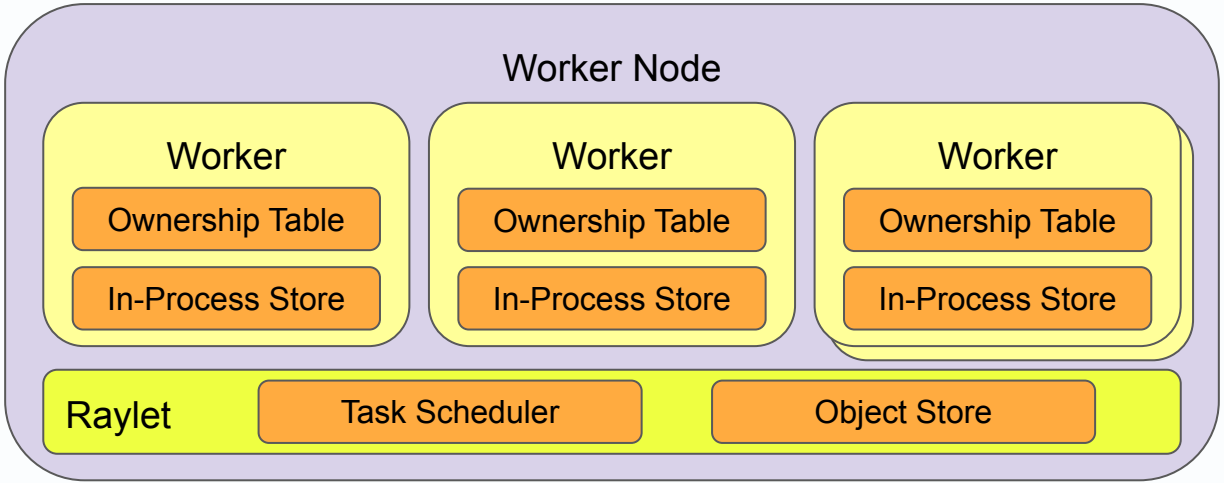
The Power and Beauty of Ray



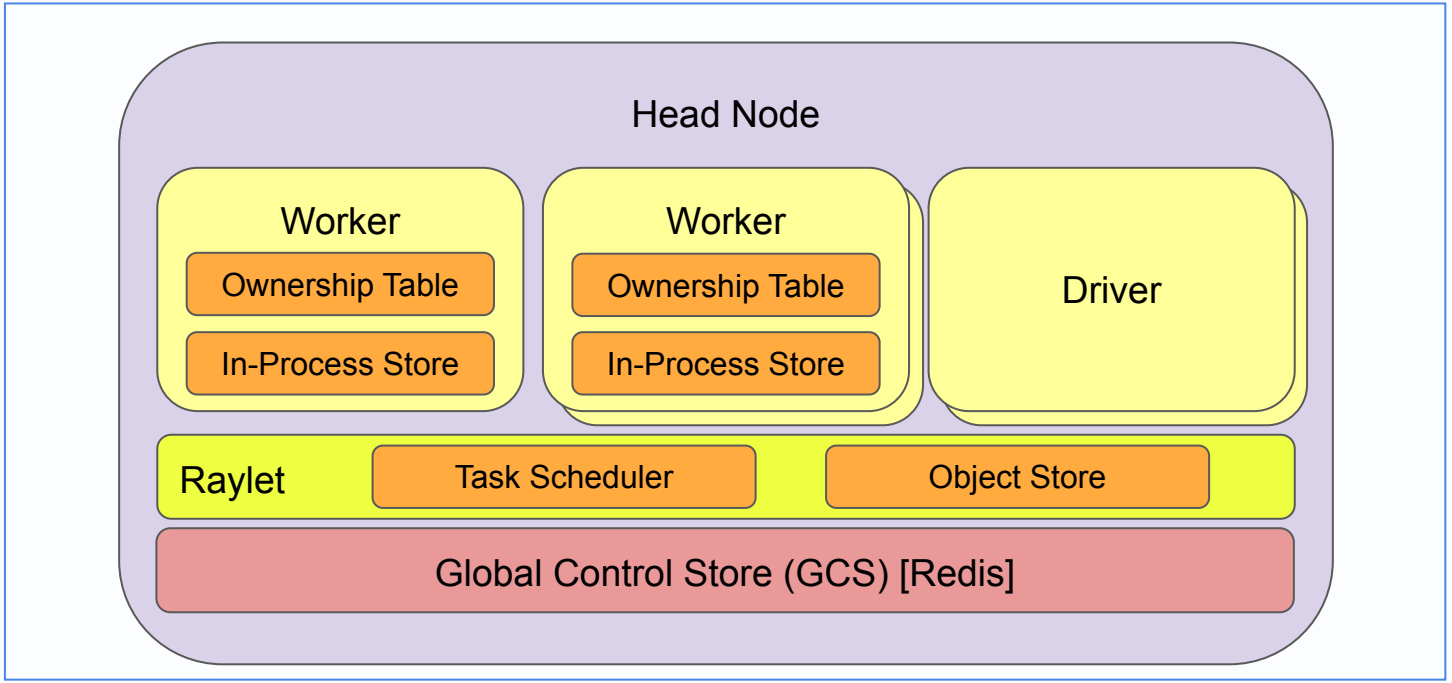
The Big Picture: Ray Cluster

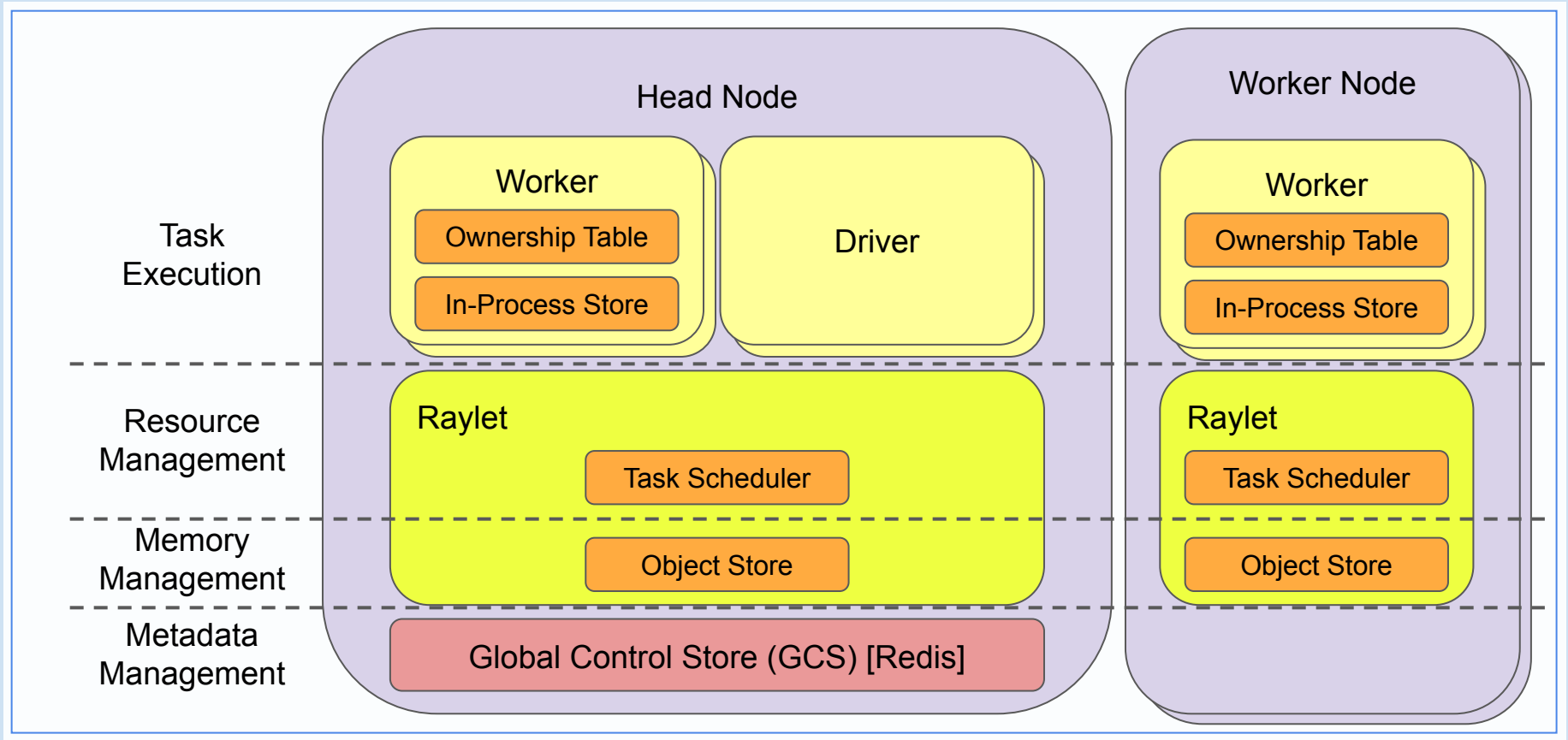


Inside a Worker Node



Inside a Head Node

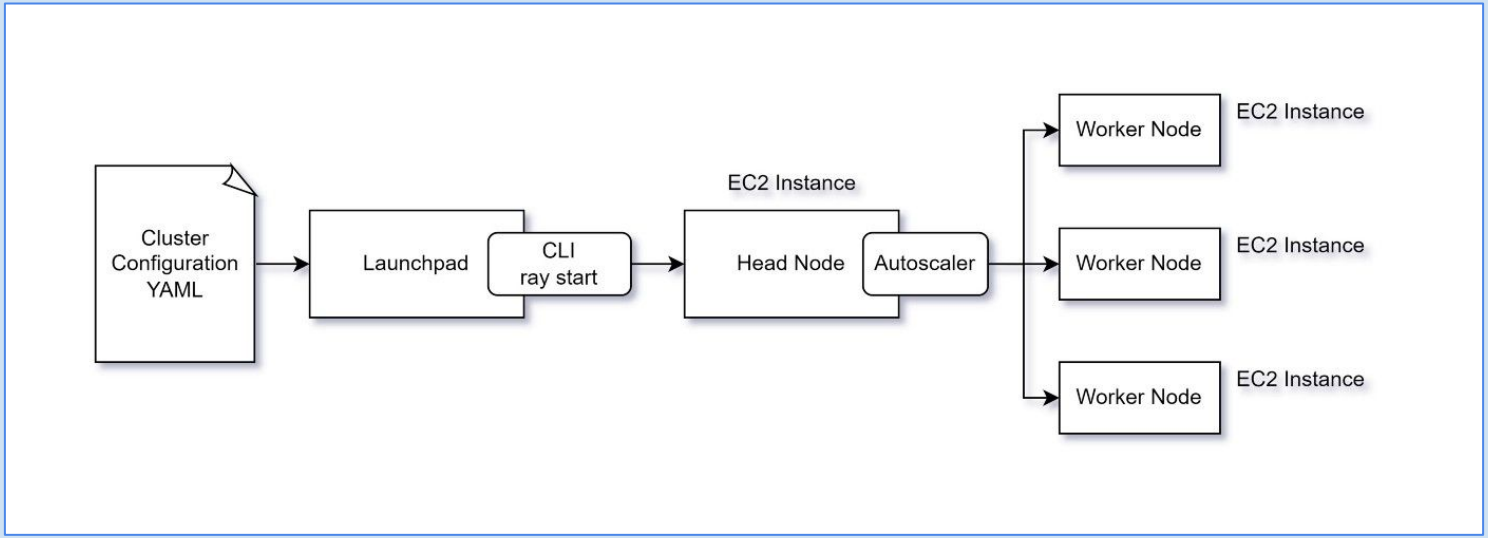



Task Execution
Resource Management
Memory Management
Metadata Management
Head Node
Worker
Ownership Table
In-Process Store
Driver
Raylet
Task Scheduler
Object Store
Global Control Store (GCS) [Redis]
Worker Node
Worker
Ownership Table
In-Process Store
Raylet
Task Scheduler
Object Store

Ray Autoscaling

- Ray supports highly elastic workloads which are most efficient on an autoscaling cluster
- An Autoscaler
  - Attempts to launch/terminate/restart worker nodes
  - Considers the resource demands of the cluster
  - Ensures that workloads have sufficient resources to run
  - Minimizes idle resources and performs load-based autoscaling
  - Uses a Binpacking Algorithm to binpack the user demands
  - Maximizes utilization and minimizes costs

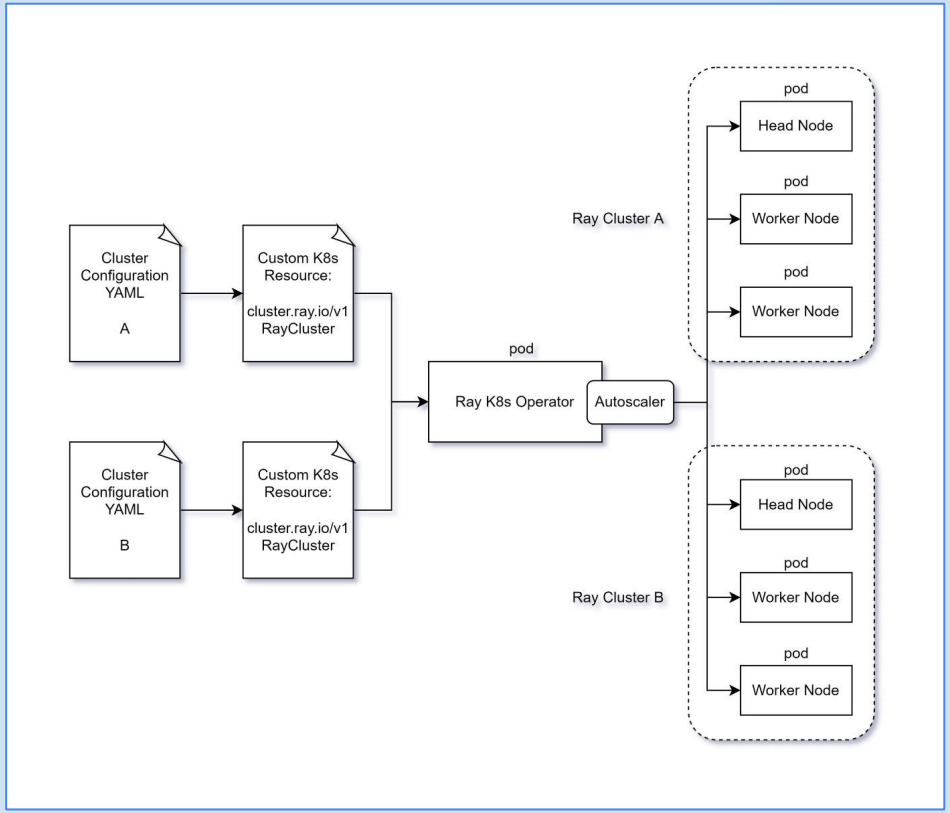
Ray Cluster: Cluster Launcher



Source: Medium Blog: Scaling Applications with Kubernetes on Ray by Vishnu Deva

Ray Cluster

Kubernetes Operator



Source: Medium Blog: Scaling Applications with Kubernetes on Ray by Vishnu Deva

PYCON 22 COLOMBIA

Te veo el próximo año

tinyurl.com/pycon-colombia-python-ray

#PyConColombia
#Ray #Kubernetes
#PyCon #Python
@greatdevaks

Anmol Krishan Sachdeva

Hybrid Cloud Architect, Google

MSc Advanced Computing, University of Bristol, UK

LinkedIn | Twitter *(@greatdevaks)*