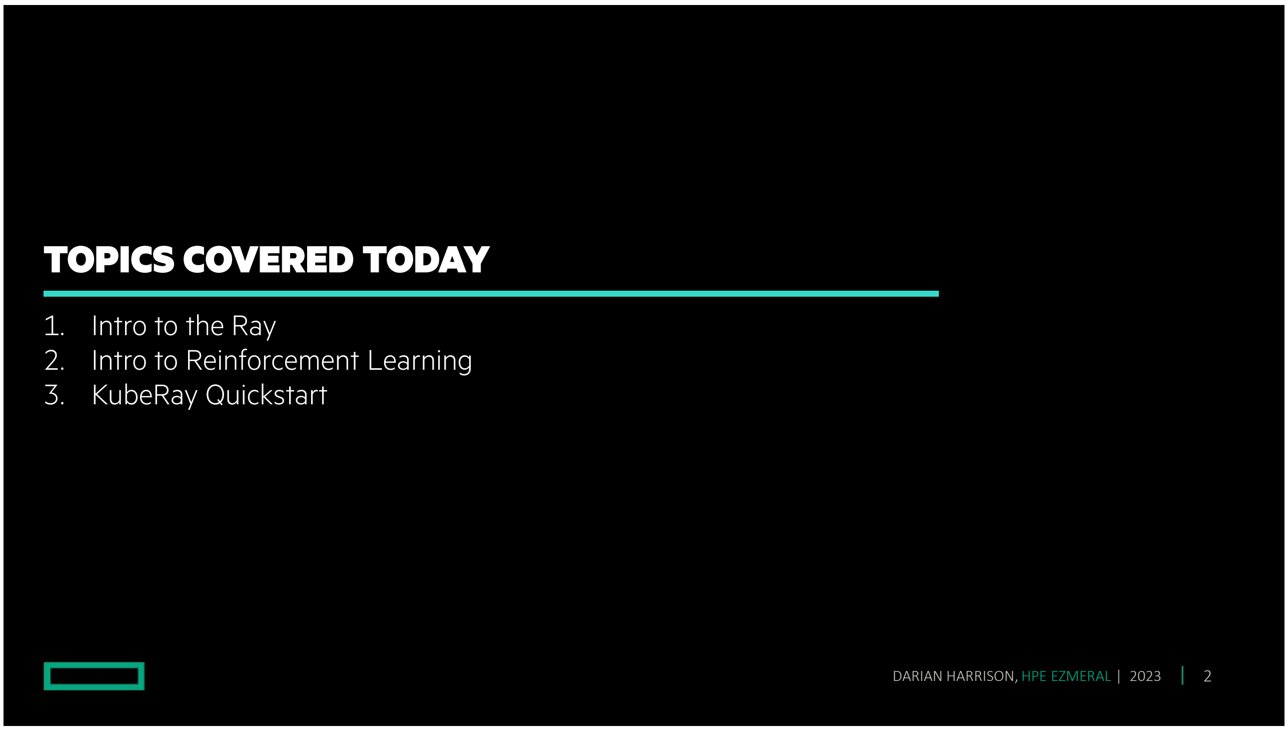
Hewlett Packard Enterprise

HPE EZMERAL

Ray

1



TOPICS COVERED TODAY

1. Intro to the Ray
2. Intro to Reinforcement Learning
3. KubeRay Quickstart

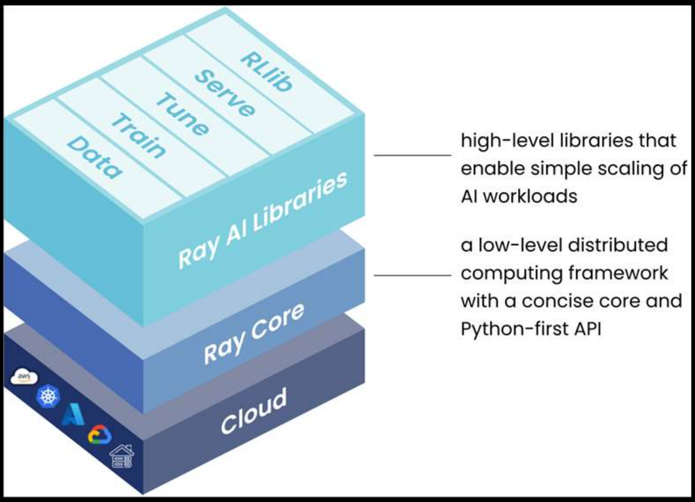
2

PHILOSOPHY

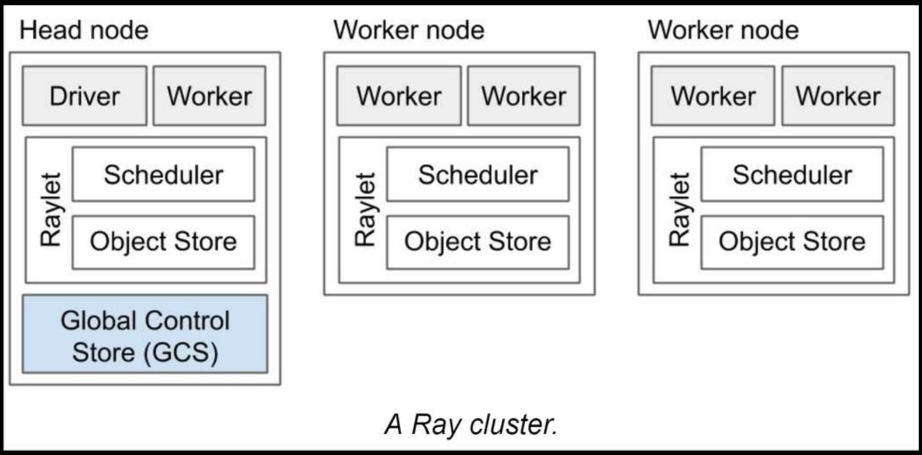
- Ray aims to provide a universal API for distributed computing.

OVERVIEW

- AI Libraries
- Core
- Clusters
- Deployment Platforms

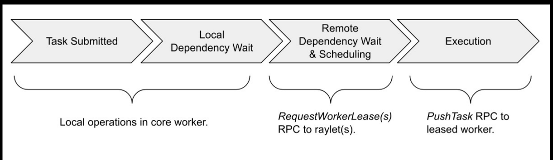

RLlib
Serve
Tune
Train
Data
Ray AI Libraries
Ray Core
Cloud
high-level libraries that enable simple scaling of AI workloads
a low-level distributed computing framework with a concise core and Python-first API

CLUSTER



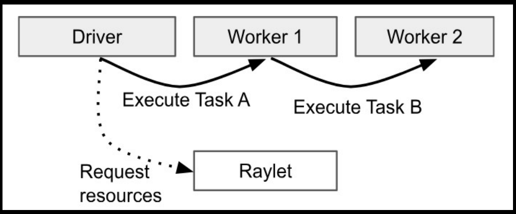
A Ray cluster.

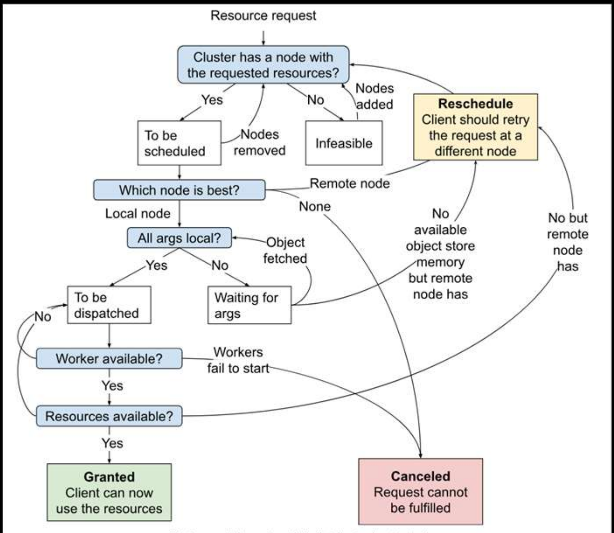
DISTRIBUTED TASK SCHEDULING AND EXECUTION



Task Submitted
Local Dependency Wait
Remote Dependency Wait & Scheduling
Execution
Local operations in core worker.
RequestWorkerLease(s) RPC to raylet(s).
PushTask RPC to leased worker.

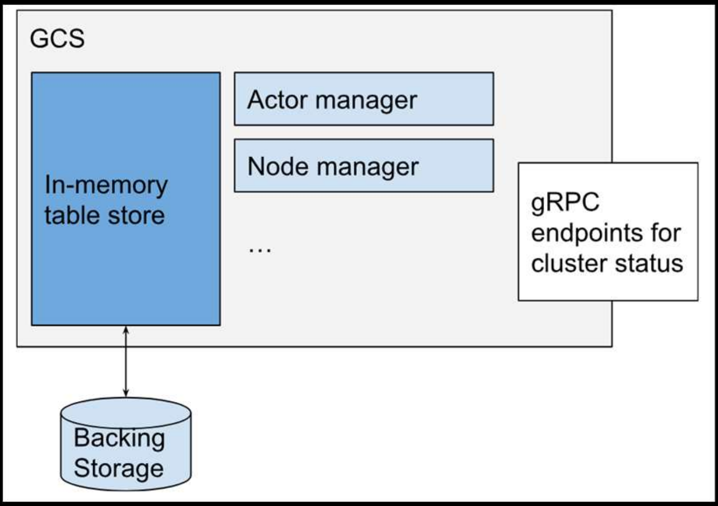
The scheduling workflow of a normal Ray task.



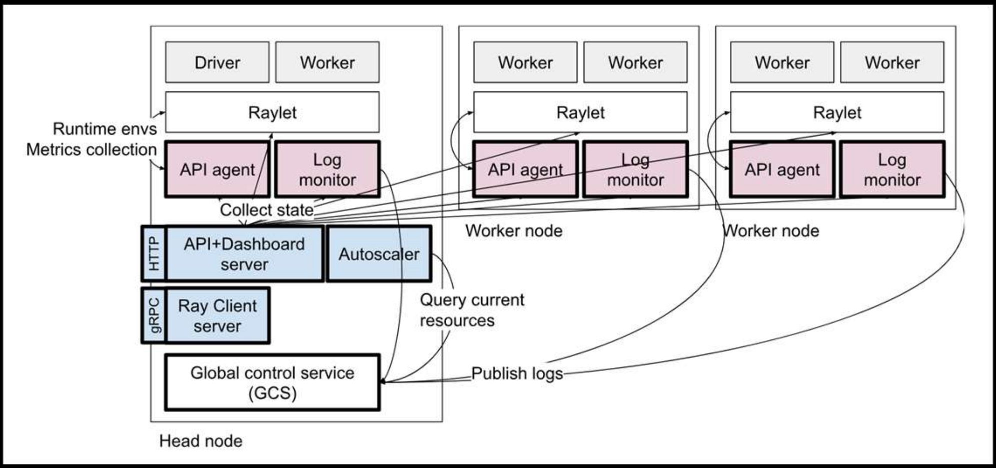


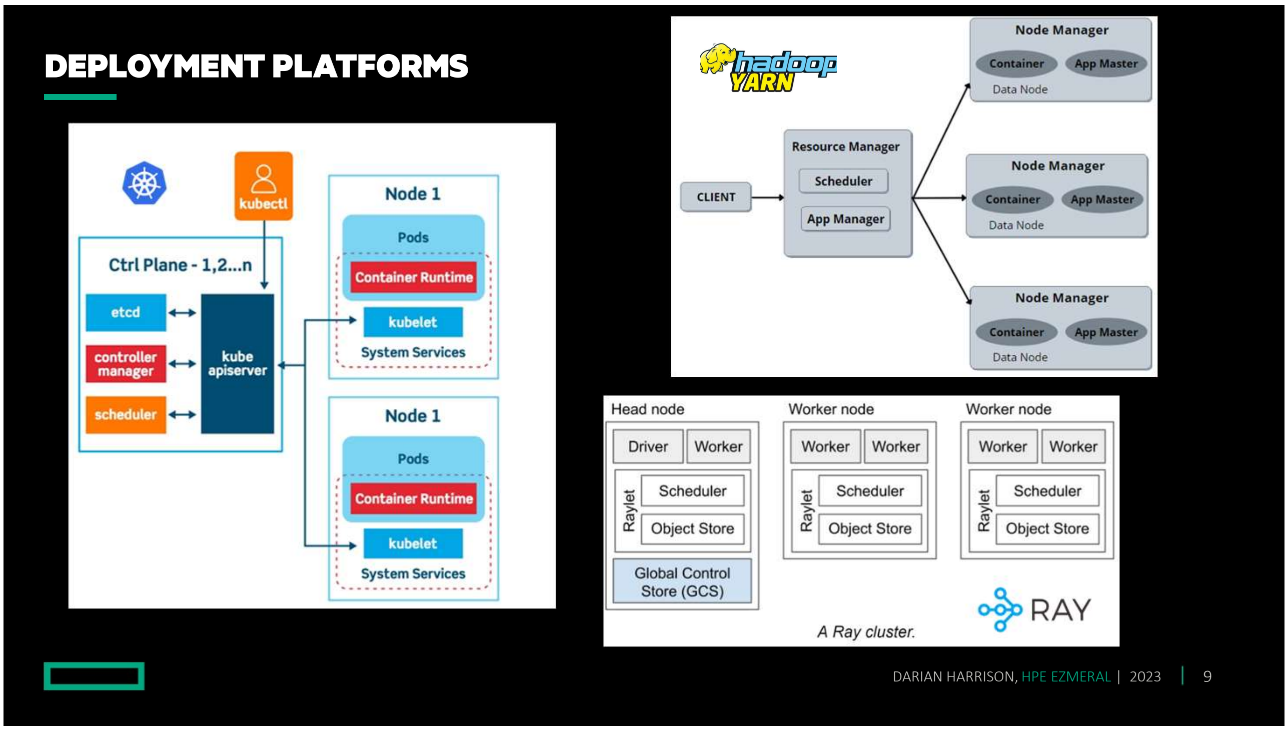
State machine of raylet distributed scheduling

GLOBAL CONTROL SERVICE

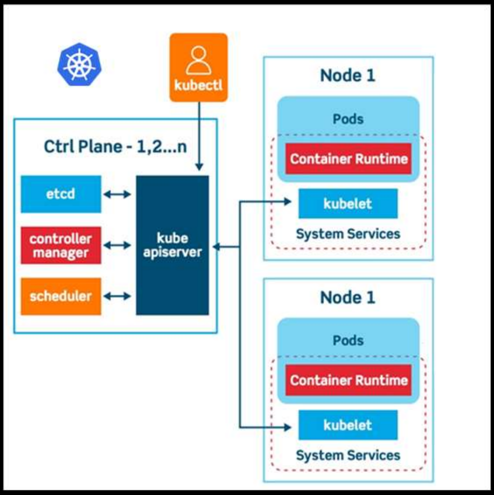


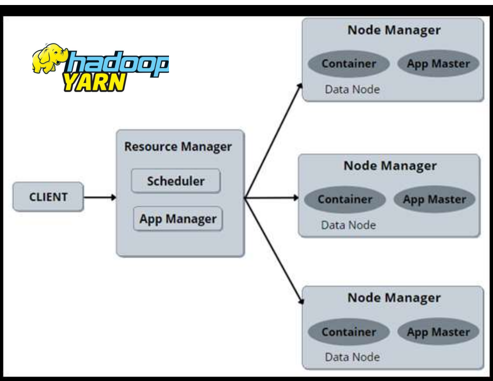
AUXILIARY SERVICES

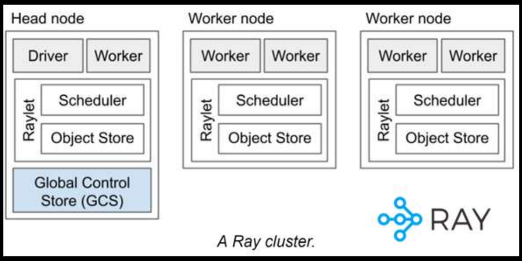




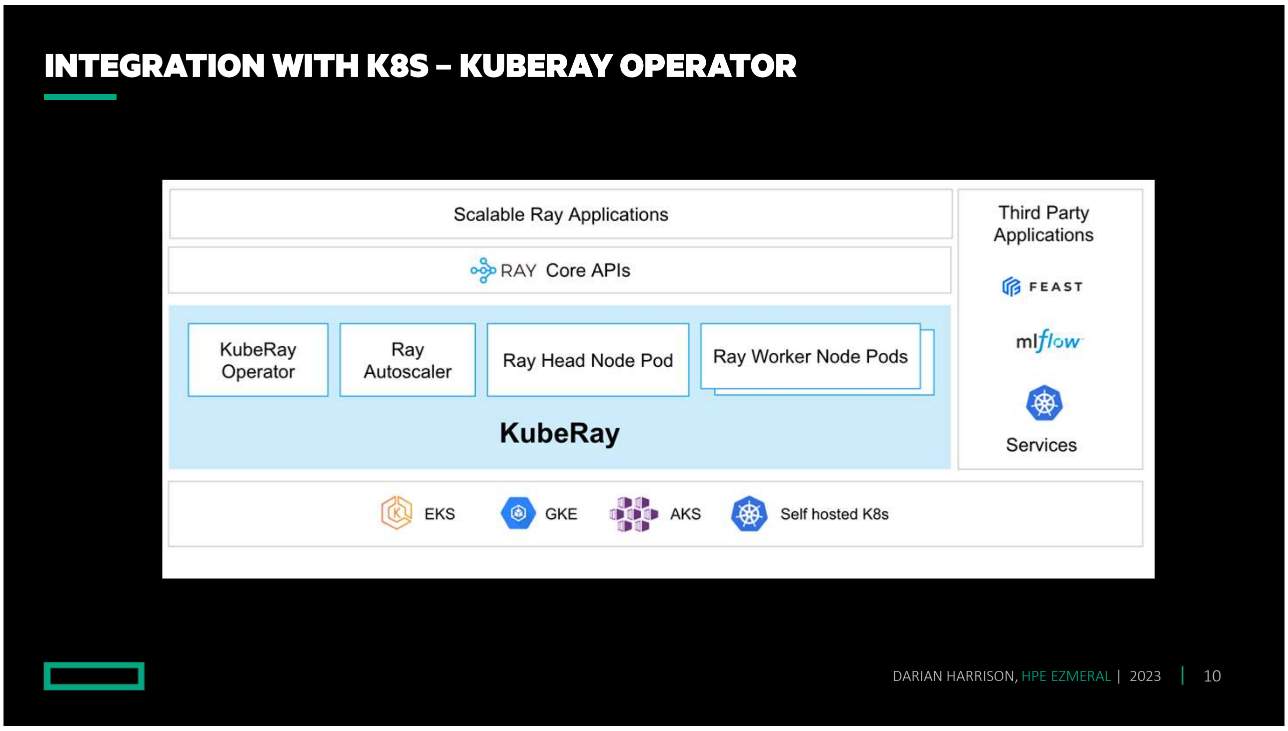
DEPLOYMENT PLATFORMS



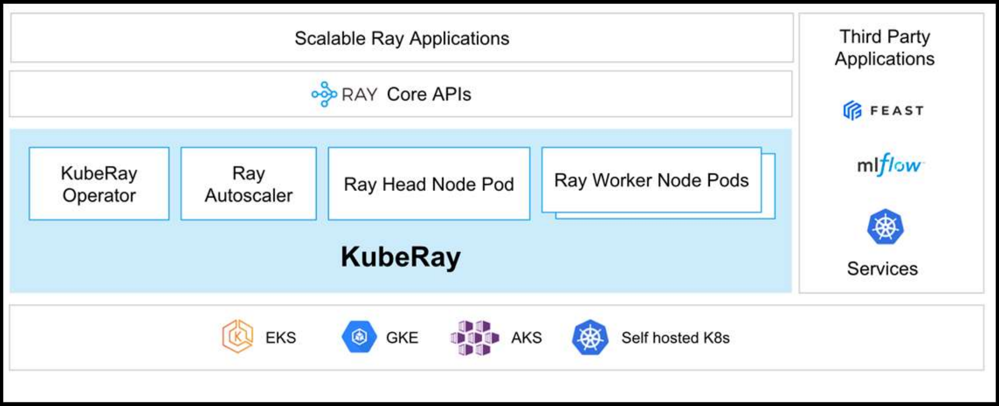


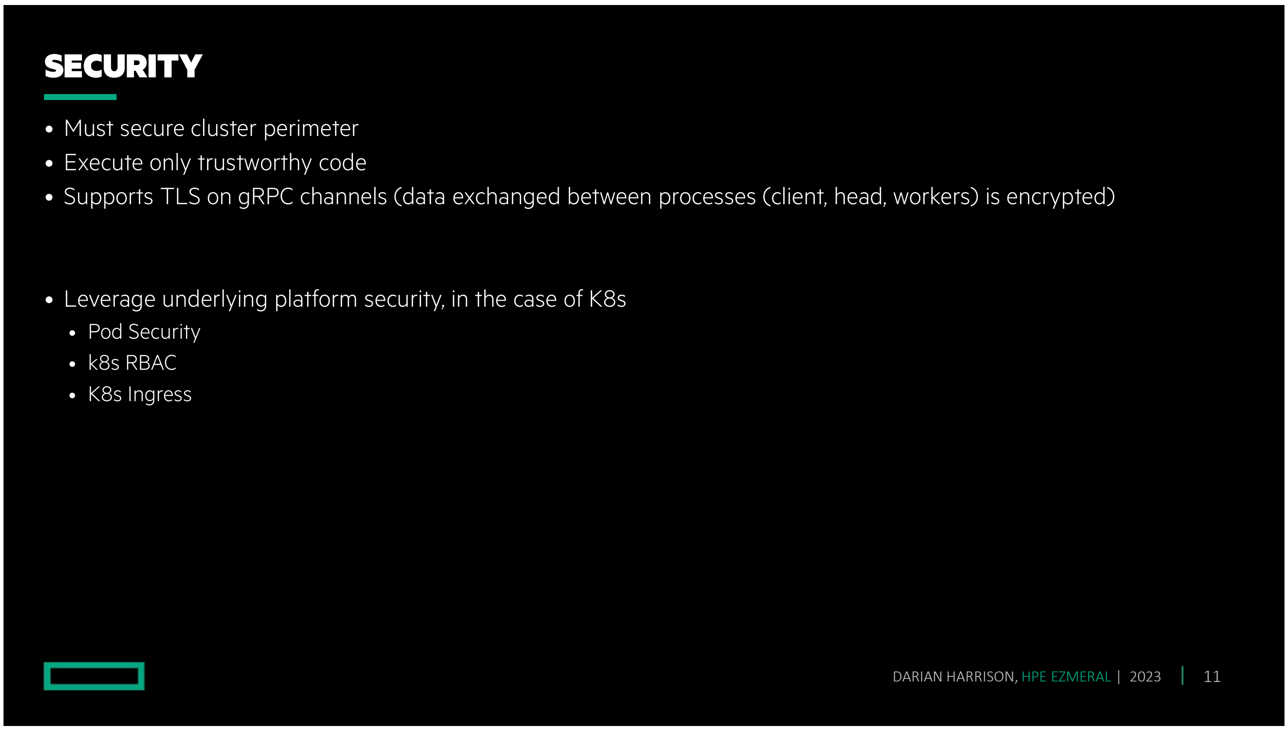


A Ray cluster.



INTEGRATION WITH K8S – KUBERAY OPERATOR



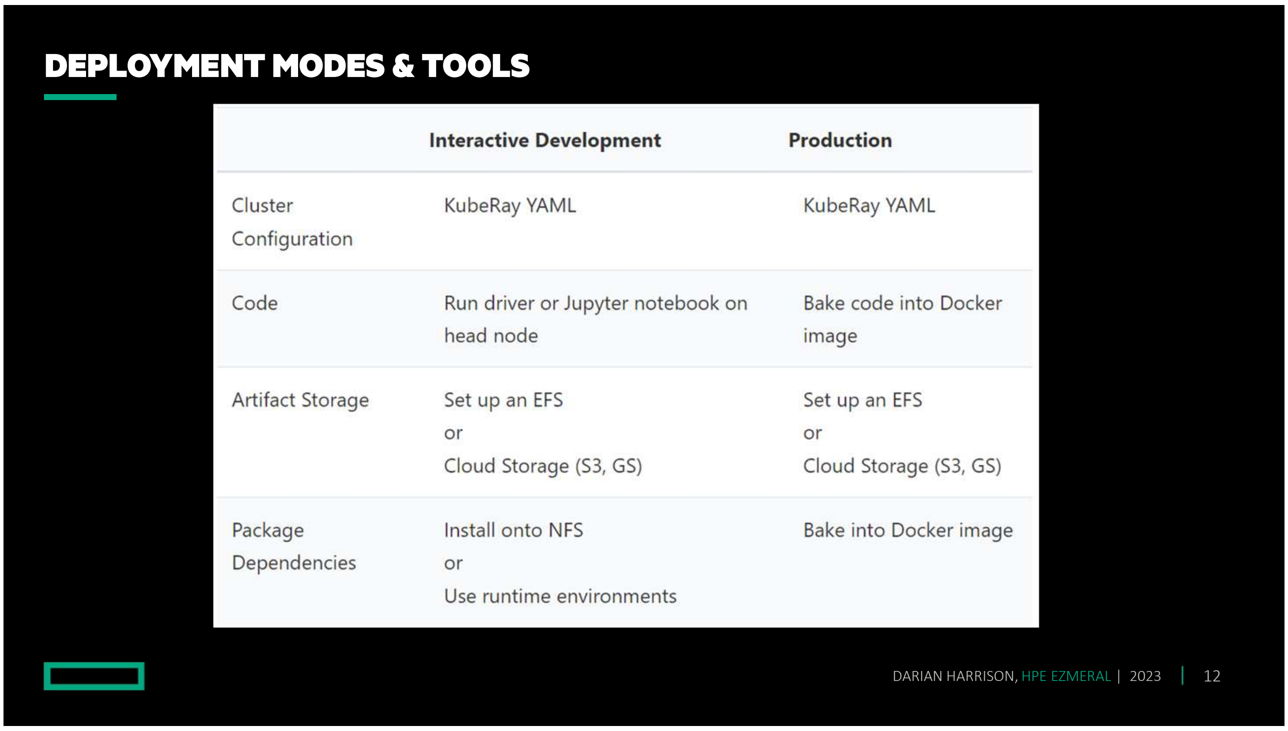


SECURITY

- Must secure cluster perimeter
- Execute only trustworthy code
- Supports TLS on gRPC channels (data exchanged between processes (client, head, workers) is encrypted)

- Leverage underlying platform security, in the case of K8s
  - Pod Security
  - k8s RBAC
  - K8s Ingress

11

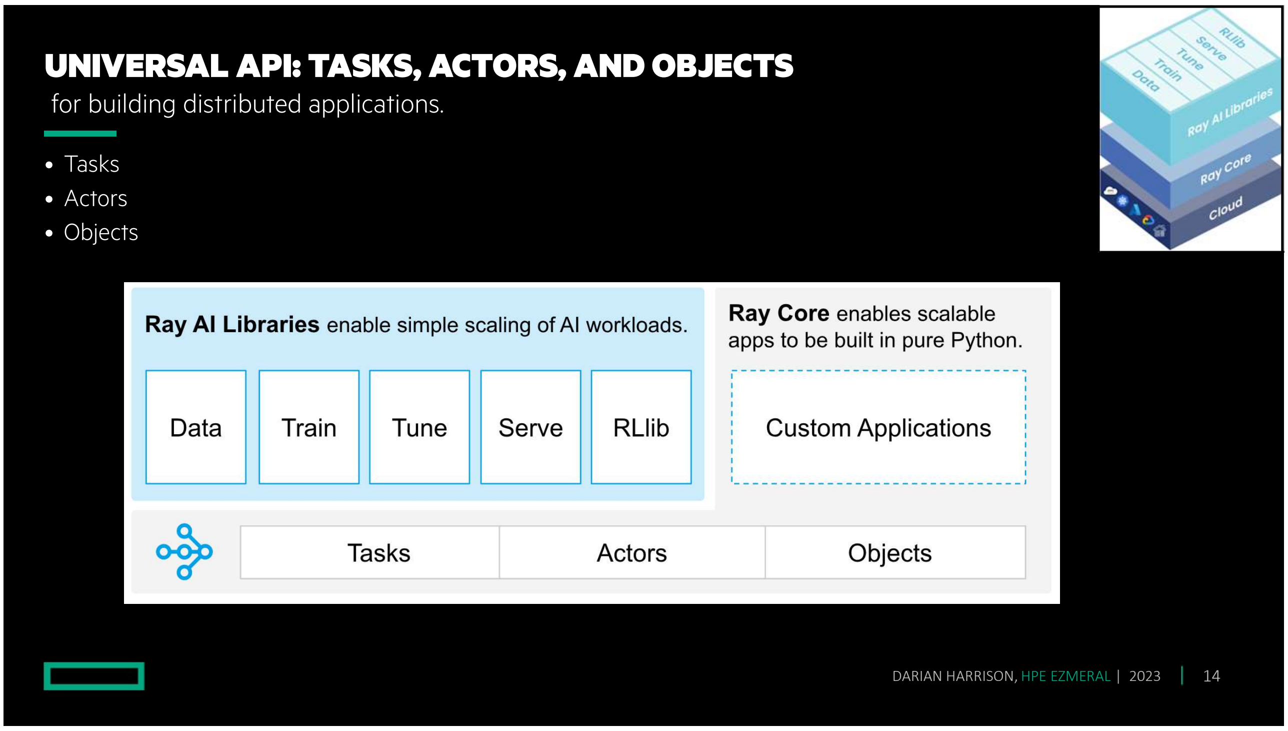


DEPLOYMENT MODES & TOOLS

	Interactive Development	Production
Cluster Configuration	KubeRay YAML	KubeRay YAML
Code	Run driver or Jupyter notebook on head node	Bake code into Docker image
Artifact Storage	Set up an EFS or Cloud Storage (S3, GS)	Set up an EFS or Cloud Storage (S3, GS)
Package Dependencies	Install onto NFS or Use runtime environments	Bake into Docker image

12

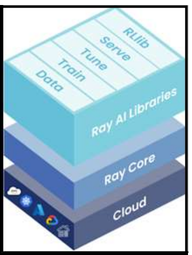
RAY CORE

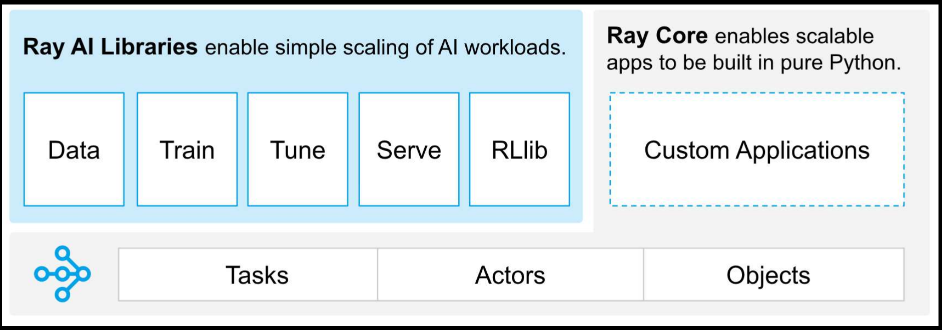


UNIVERSAL API: TASKS, ACTORS, AND OBJECTS

for building distributed applications.

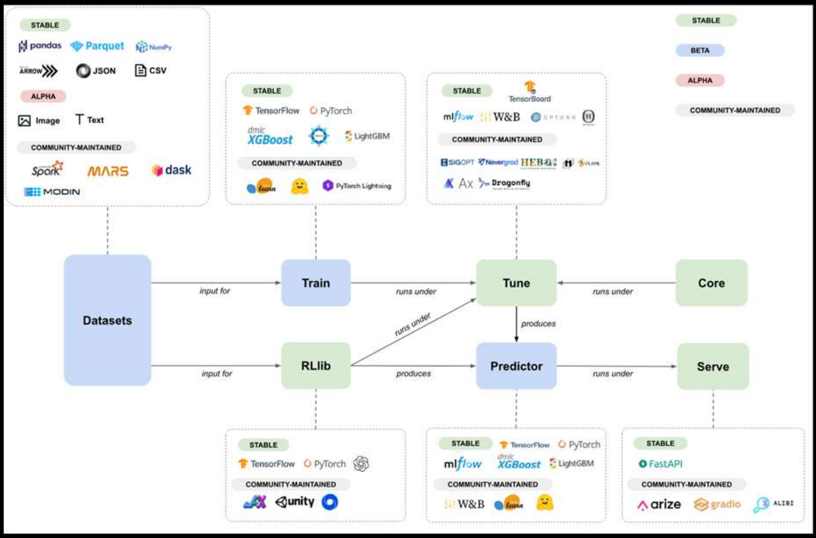
- Tasks
- Actors
- Objects

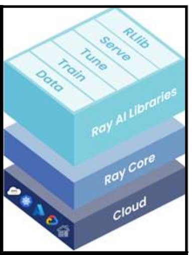



Ray AI Libraries enable simple scaling of AI workloads.
Data
Train
Tune
Serve
RLlib
Ray Core enables scalable apps to be built in pure Python.
Custom Applications
Tasks
Actors
Objects

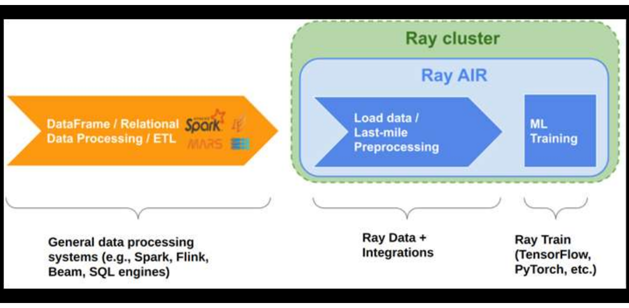
RAY AI LIBRARIES

OVERVIEW

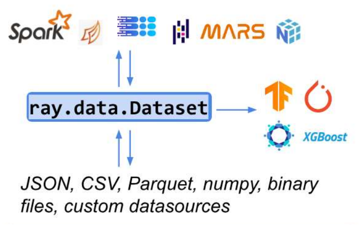


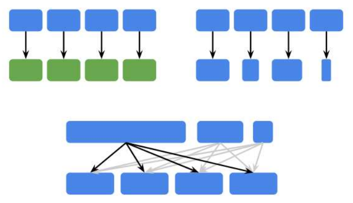


RAY DATA



Ray cluster
Ray AIR
DataFrame / Relational Data Processing / ETL
Load data / Last-mile Preprocessing
ML Training
General data processing systems (e.g., Spark, Flink, Beam, SQL engines)
Ray Data + Integrations
Ray Train (TensorFlow, PyTorch, etc.)





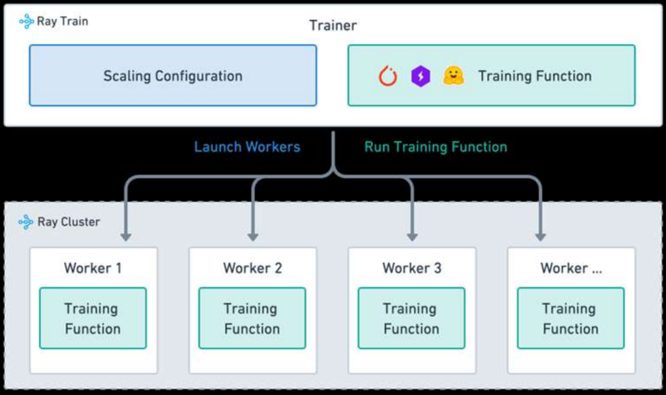
```
ds = ray.data.read_parquet(...)
# Pass datasets to tasks and actors
func.remote(ds)
# Split datasets into shards
shards = ds.split(xgboost.num_workers,
    locality_hints=xgboost.workers)
# Distributed training on datasets
for i, s in enumerate(shards):
    xgboost.workers[i].train.remote(s)
```

Standard way to load and exchange data in Ray

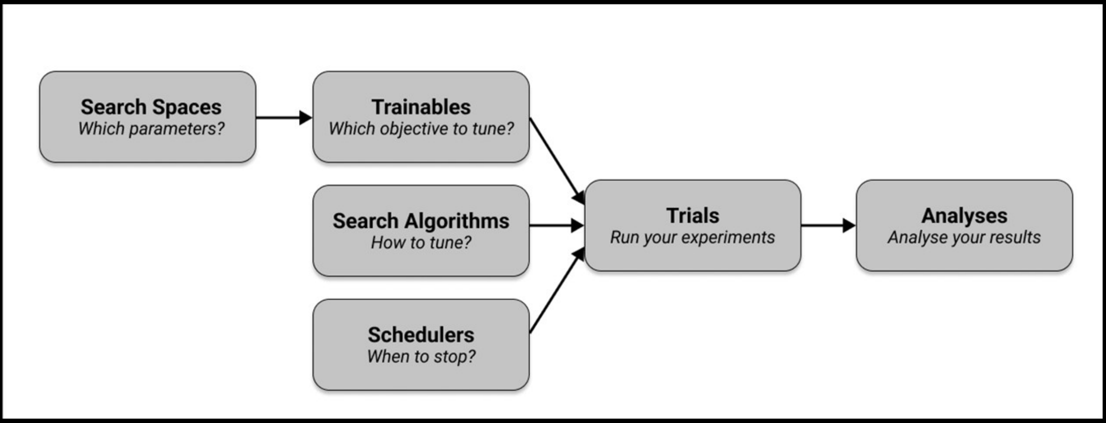
Basic distributed ops: map, filter, and repartition

Seamless interop with Ray tasks, actors, and libraries

RAY TRAIN

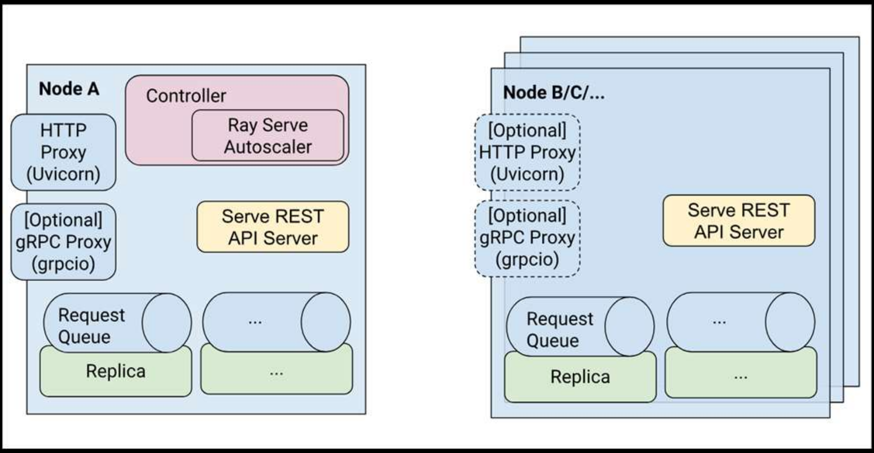


RAY TUNE



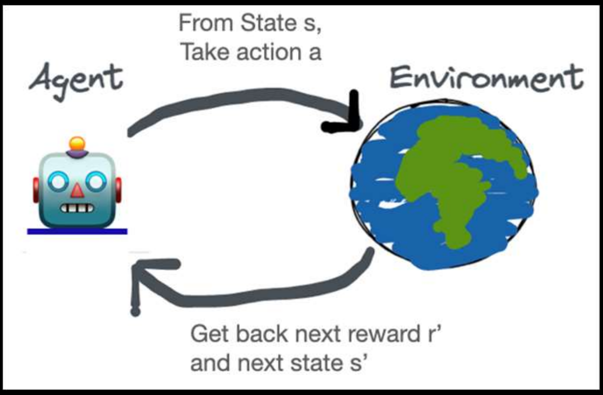
19

RAY SERVE

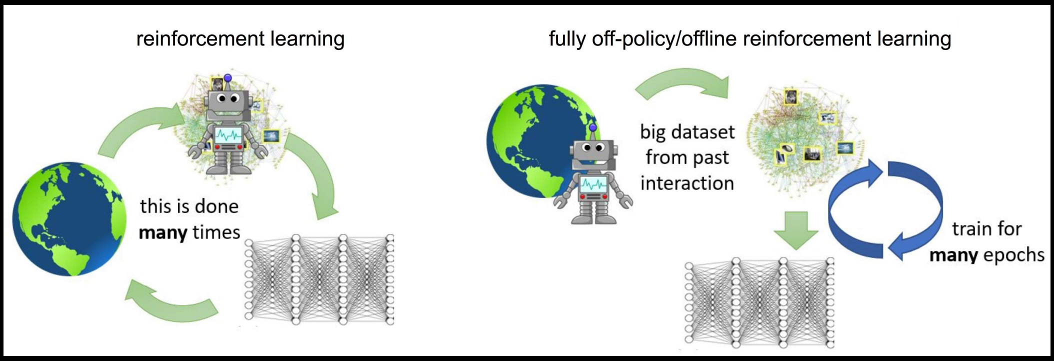


20

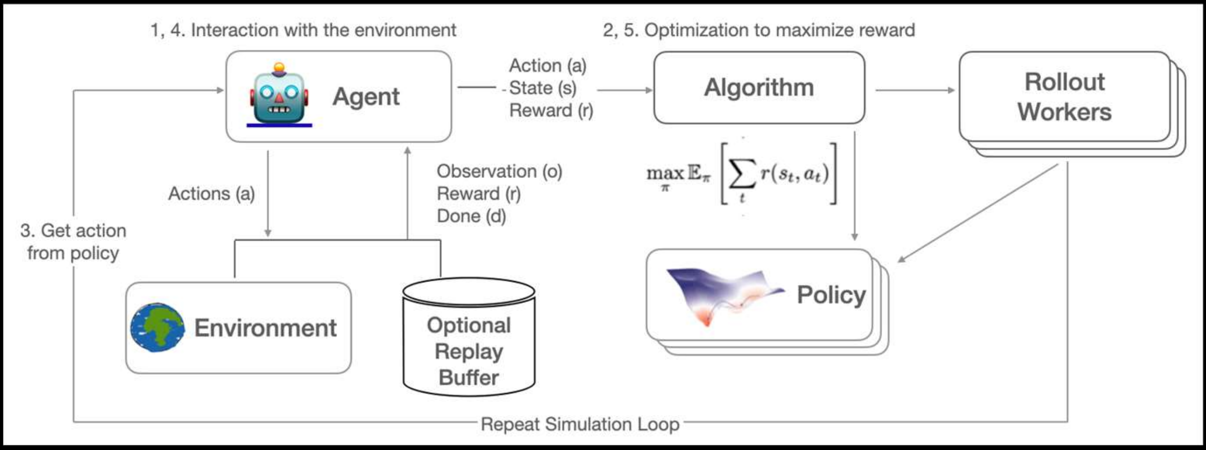
REINFORCEMENT LEARNING



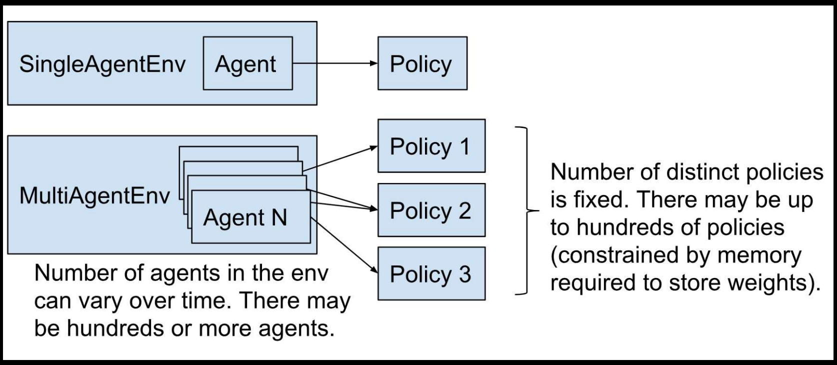
DEEP REINFORCEMENT LEARNING



RAY RLLIB - CONCEPTS



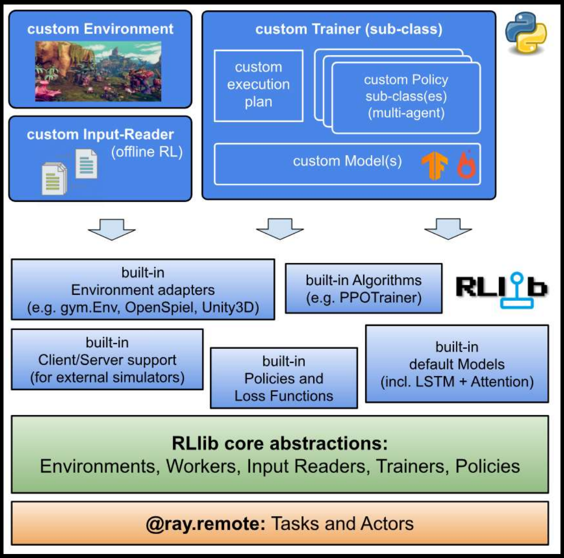
MULTI-AGENT MULTI-MODEL EXAMPLE



Number of agents in the env can vary over time. There may be hundreds or more agents.

Number of distinct policies is fixed. There may be up to hundreds of policies (constrained by memory required to store weights).

RLLIB CORE ABSTRACTIONS


custom Environment
custom Input-Reader
(offline RL)
custom Trainer (sub-class)
custom execution plan
custom Policy sub-class(es) (multi-agent)
custom Model(s)
built-in Environment adapters (e.g. gym.Env, OpenSpiel, Unity3D)
built-in Algorithms (e.g. PPOTrainer)
RLlib
built-in Client/Server support (for external simulators)
built-in Policies and Loss Functions
built-in default Models (incl. LSTM + Attention)
RLlib core abstractions:
Environments, Workers, Input Readers, Trainers, Policies
@ray.remote: Tasks and Actors

DEMO

REFERENCES

- https://docs.ray.io/en/latest/index.html