

# Introduction to Ray for Distributed Applications

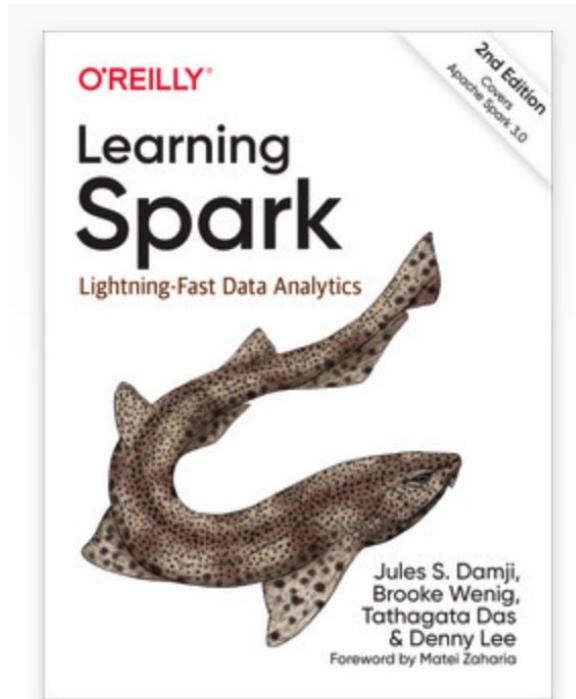
Jules Damji, Anyscale  
@2twitme

TAs: Stephanie Wang, Jiajun Yao, Sangbin Cho



# \$whoami (Jules)

- Lead Developer Advocate @Anyscale
- Senior Developer Advocate @Databricks
- Led Developer Advocacy @Hortonwork
- Held SWE positions:
  - + Sun Microsystems
  - + Netscape
  - + @Home
  - + Loudcloud/Opsware
  - + Verisign



# Anyscale

**Who we are:** Original creators of Ray, a unified framework for scalable computing

**What we do:** Scalable compute for AI And Python

**Why we do it:** Scaling is a necessity, scaling is hard; make distributed computing easy and simple for everyone

# Agenda

- Why & What's Ray & Ray Ecosystem
- Ray Architecture & Components
- Ray Core Design Patterns & APIs
- Modules [1 - 3]
- Closing Q & A with Committers
- Happy Hour 🍺 + Meetup

# Ray Summit Meetup

## Seacliff foyer

Meetup

### Ray Summit Meetup Community Talks

Monday, August 22  
6:00 PM - 8:00 PM

We are delighted to host an exclusive Ray Summit Meetup, hosted by Anyscale with Ray community talks, on the eve of the summit. Invited Ray community speakers will share how they use Ray to scale and solve challenging ML problems.

You don't have to be registered for the Ray Summit to attend. The meetup is free for the community. Join us for the Ray Summit Happy Hour from 5:00 ~ 6:00 p.m., followed immediately by the meetup.

Agenda (The times are not strict; they may vary slightly.)

- 5:00 ~ 6:00 p.m. Ray Summit Community Happy Hour (in Seacliff Foyer)
- 6:00 p.m. Welcome remarks, announcements, and agenda - Jules Damji, Anyscale
- 6:05 p.m. Talk 1: Ray + Arize: Close the ML infrastructure loop - Aparna Dhinakaran, Arize AI
- 6:35 p.m. Q & A
- 6:40 p.m. Talk 2: Maintaining long-running distributed Ray clusters - Jaehyun Sim, Ikigai Labs
- 7:20 p.m. Q & A
- 7:25 p.m. Talk 3: Large-scale distributed approximate nearest neighbor search with Ray - Daniel Acuna, Syracuse University
- 7:50 p.m. Q & A

Talk 1: Ray + Arize: Close the ML infrastructure loop Detecting, diagnosing, and resolving ML model performance can be difficult for even the most sophisticated ML engineers. As more machine learning models are deployed into production, it is imperative we have tools to monitor, troubleshoot, and explain model decisions. Join Aparna Dhinakaran, chief product officer at Arize AI, in a discussion on the state of commonly seen ML production monitoring challenges. Learn how to use ML Observability from training through production environments to find upstream model issues faster, monitor your models in real time at scale, and improve model interpretability and explainability.



**Daniel Acuna**

Associate Professor, Computer Science Department, University of Colorado



**Jaehyun Sim**

Director of Engineering, Ikigai Labs, Inc.



**Aparna Dhinakaran**

Chief Product Officer, Arize AI



**Jules Damji**

Lead Developer Advocate, Anyscale

# Why Ray

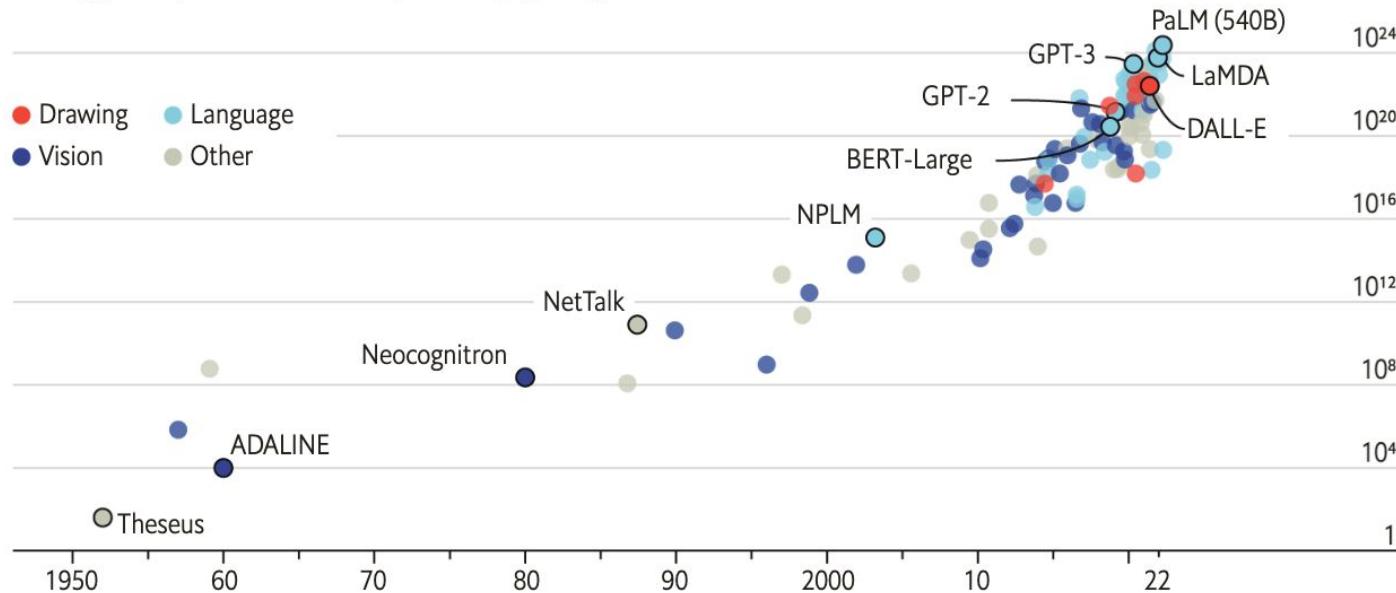


- Machine learning is pervasive
- Distributed computing is a necessity
- Python is the default language for DS/ML

# Blessings of scale ...

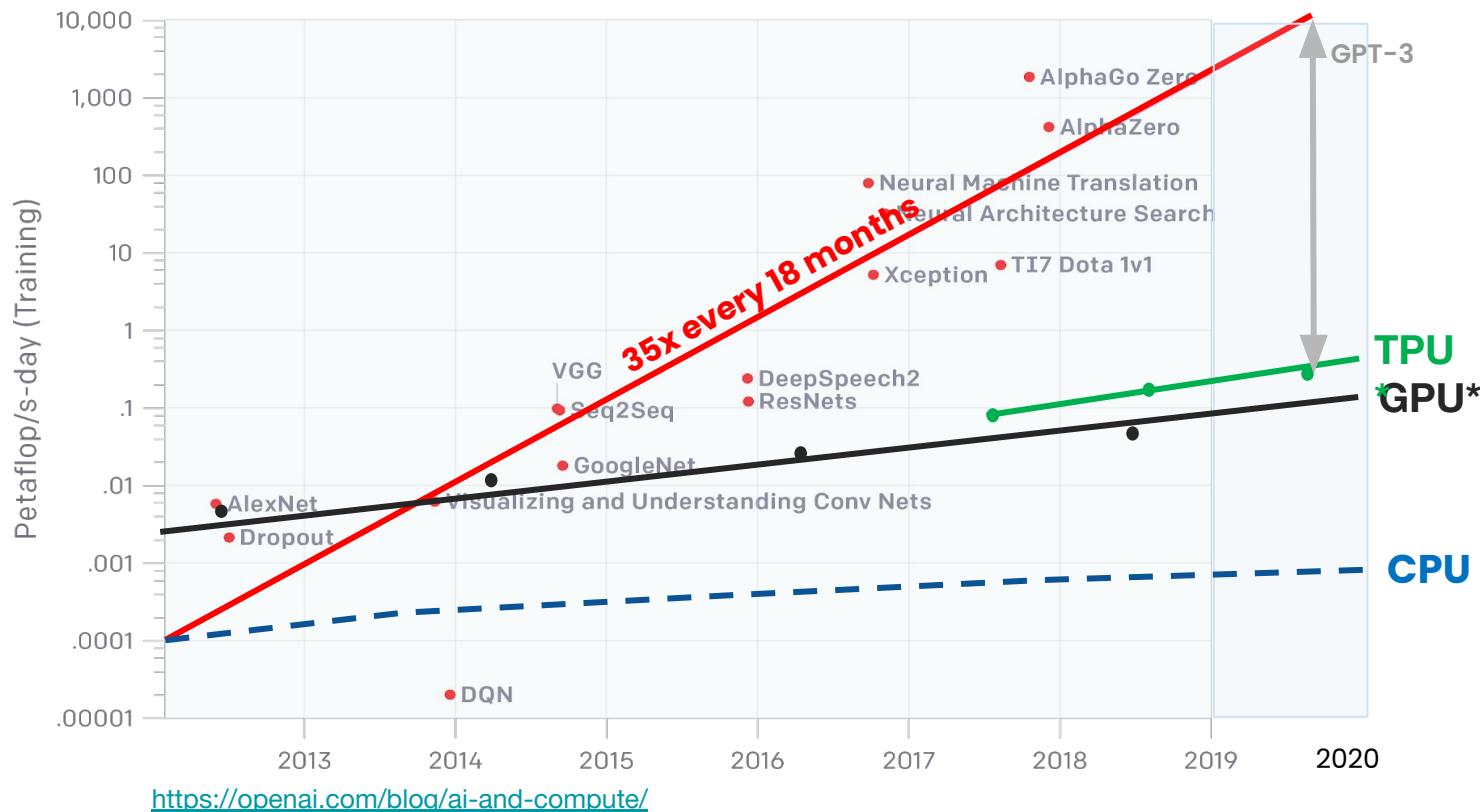
## The blessings of scale

AI training runs, estimated computing resources used  
Floating-point operations, selected systems, by type, log scale

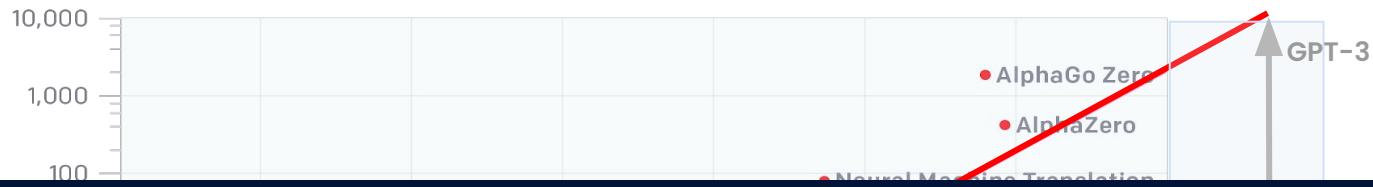


Sources: "Compute trends across three eras of machine learning", by J. Sevilla et al., arXiv, 2022; Our World in Data

# Compute - supply demand problem



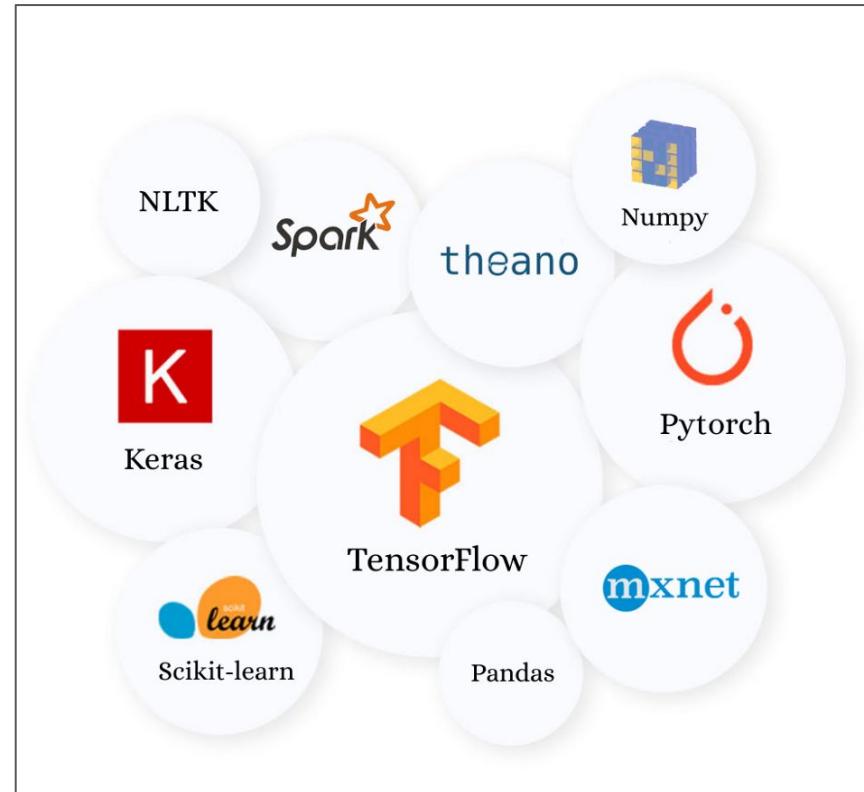
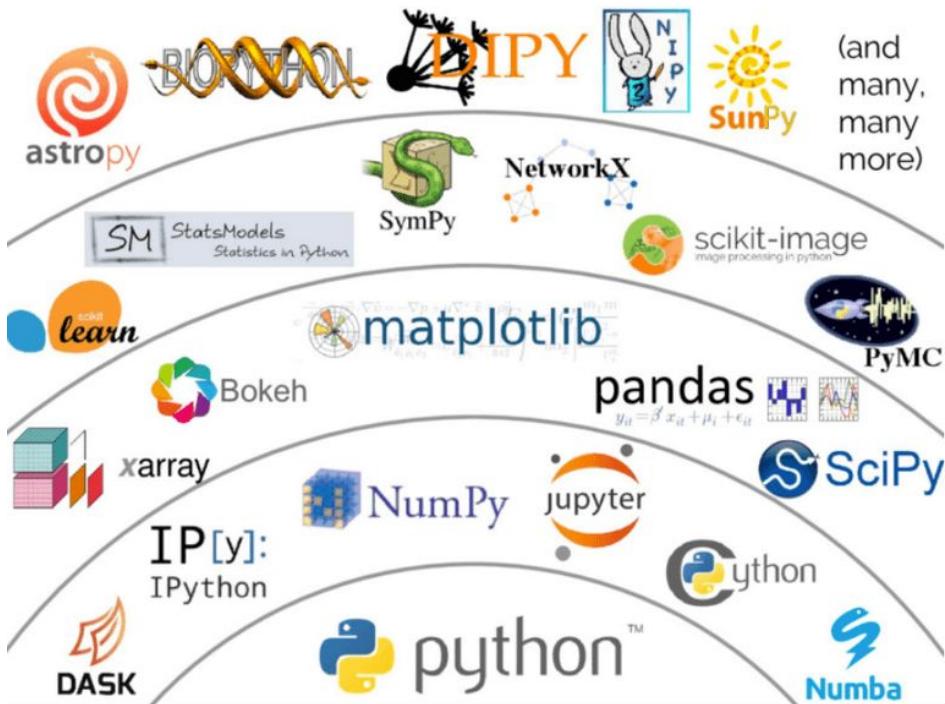
# Specialized hardware is not enough



## No way out but to distribute!



# Python data science ecosystem



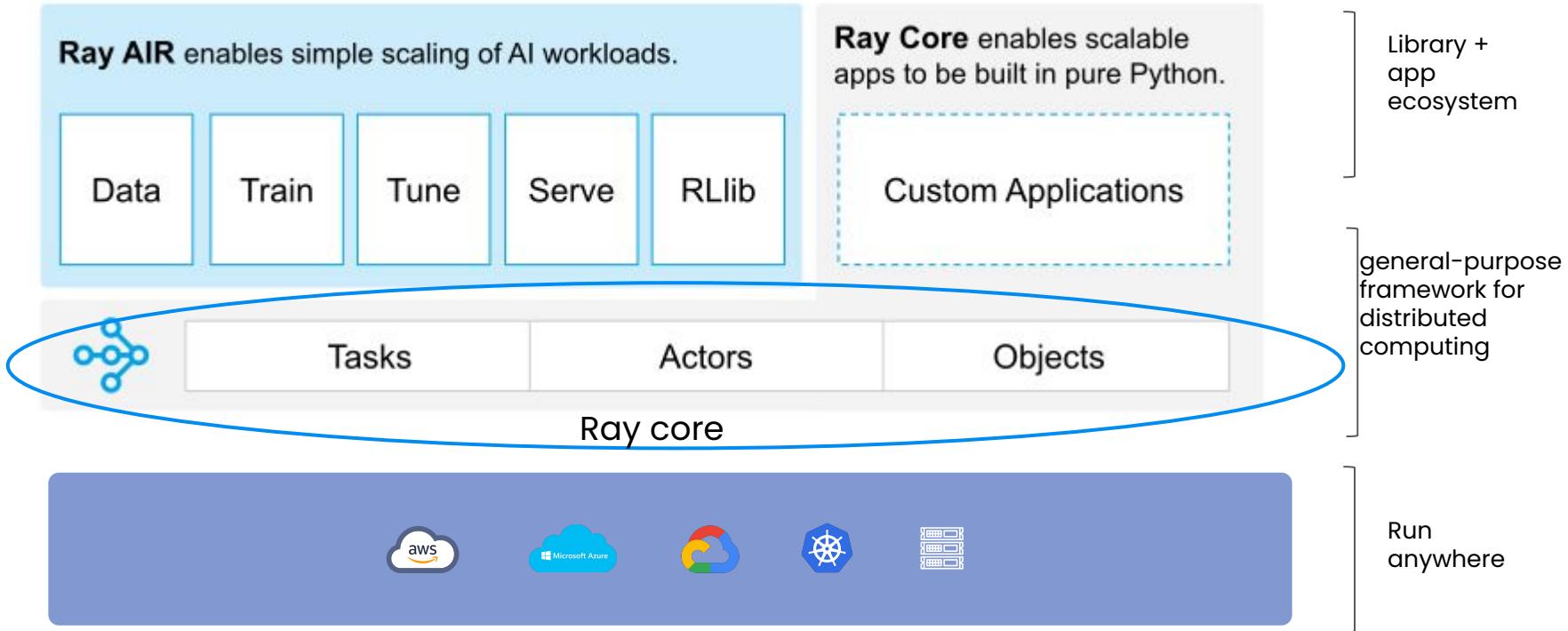
# What is Ray



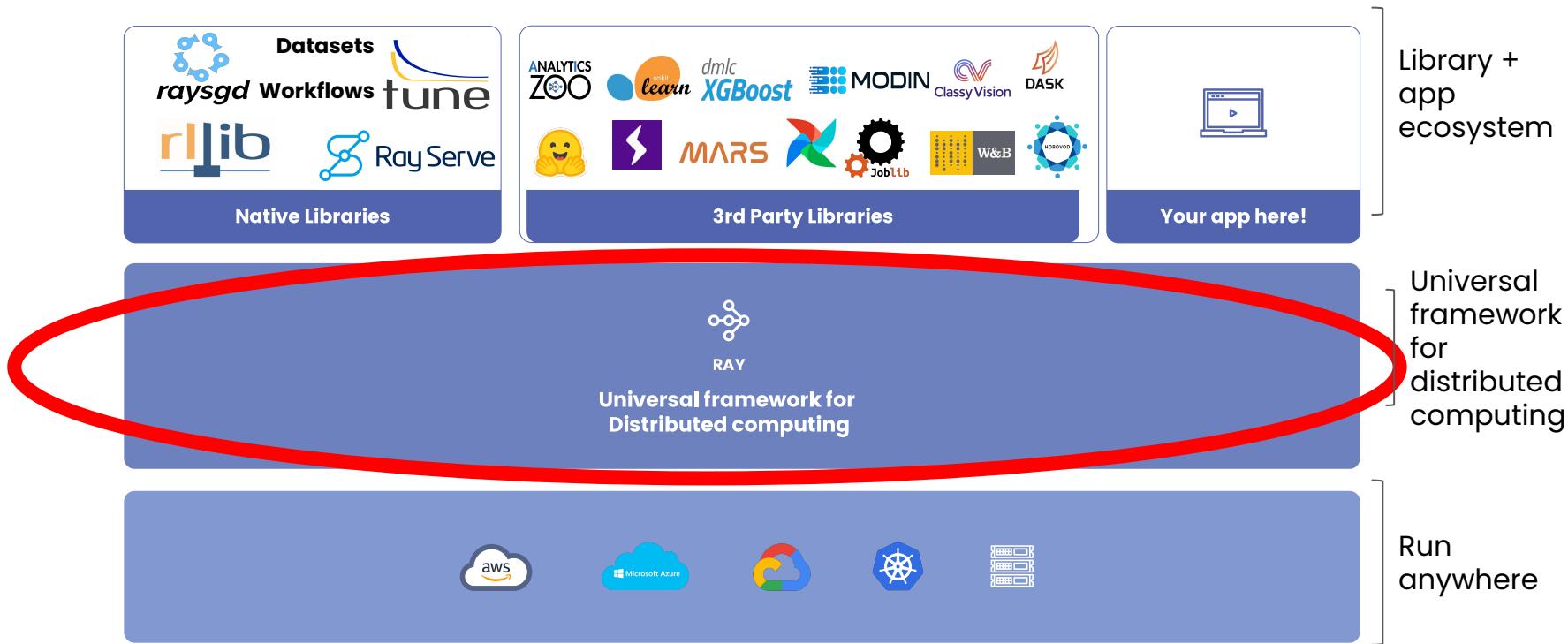
- A simple/general-purpose library for distributed computing
- An ecosystem of Python libraries (for scaling ML and more)
- Runs on laptop, public cloud, K8S, on-premise

A layered cake of functionality and capabilities for scaling ML workloads

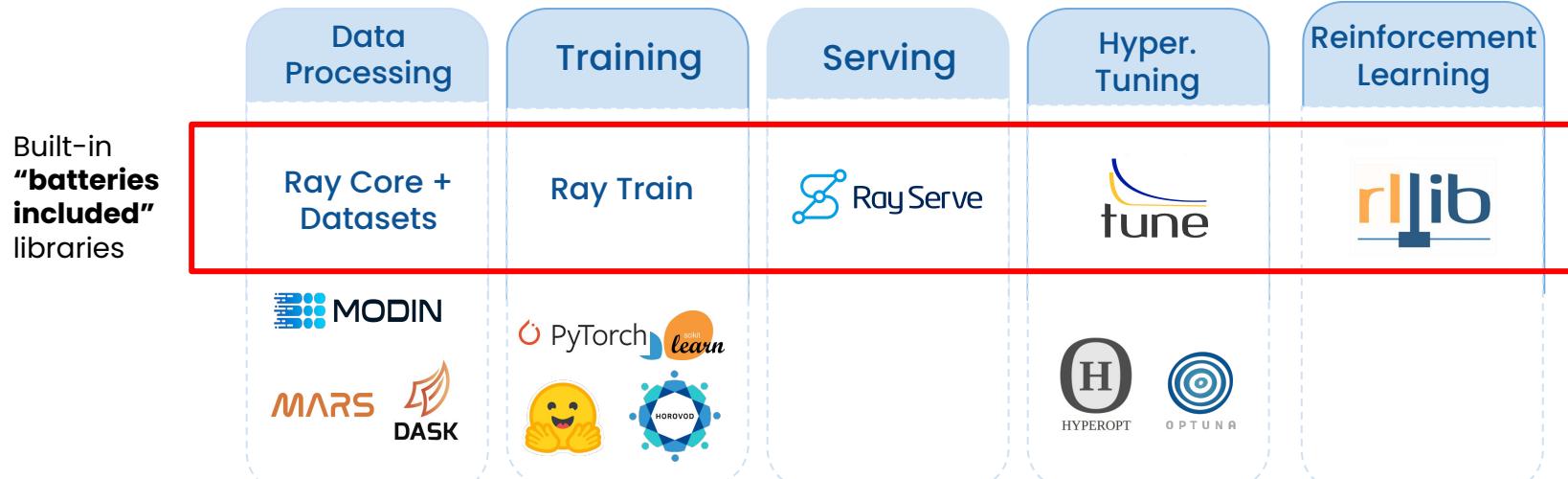
# A Layered Cake and Ecosystem



# A Layered Cake and Ecosystem

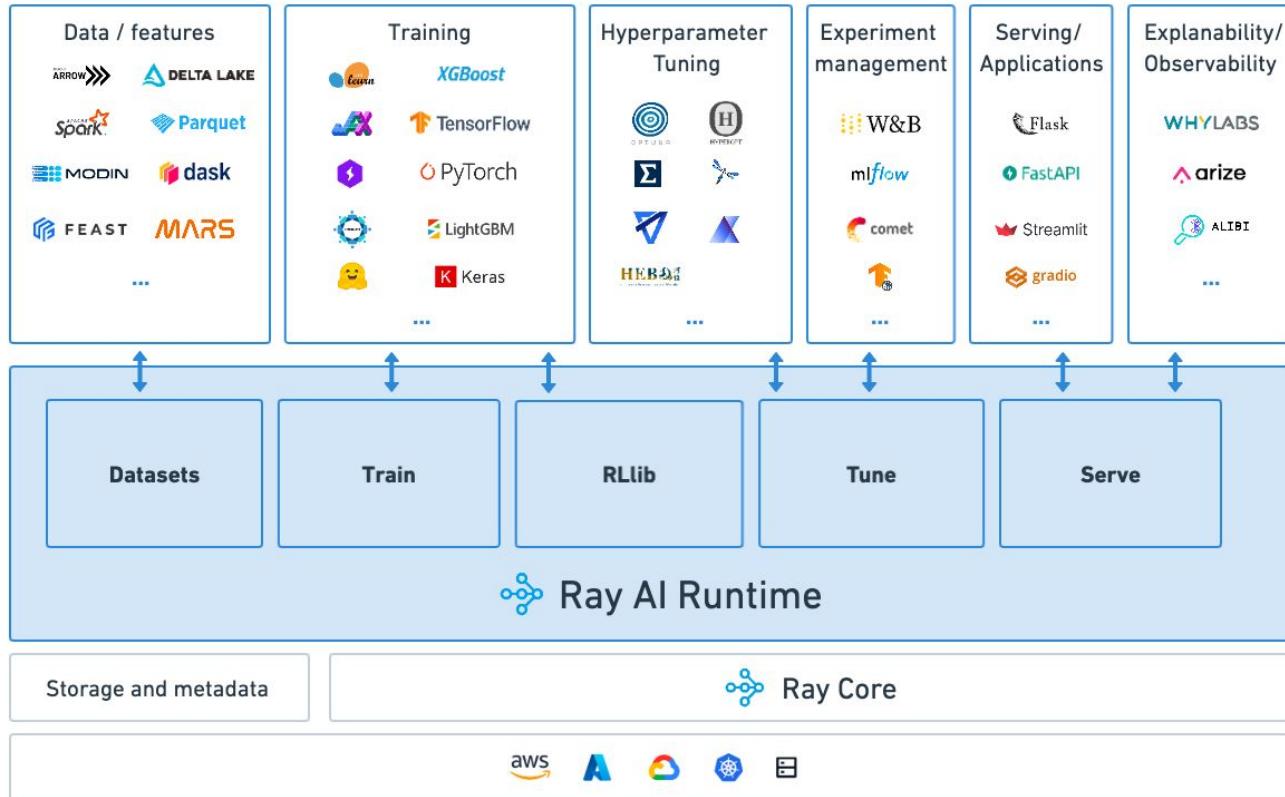


# Rich ecosystem of scaling ML workloads



Only use the libraries you need!

# Ray AI Runtime (AIR) is a scalable runtime for end-to-end ML applications



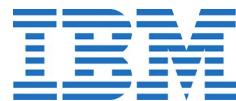
# Ray 2.0 & Ray AIR sessions

- Introduction Ray AI Runtime
- State of Ray Serve in 2.0
- Shuffling 100TB with Ray Datasets
- Ray Observability: Present & future
- Many others in Ray Deep Dives track ...

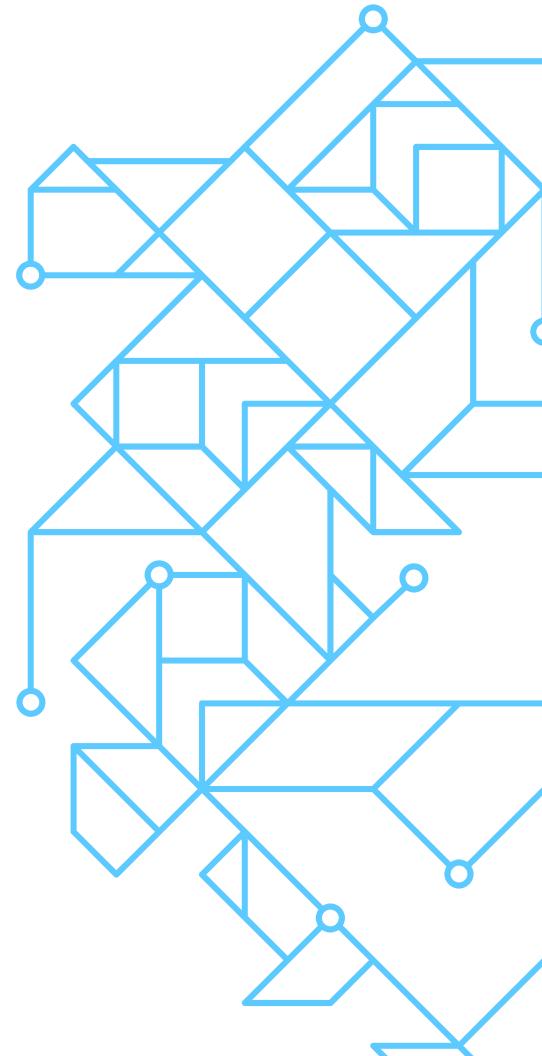
# Who's using it



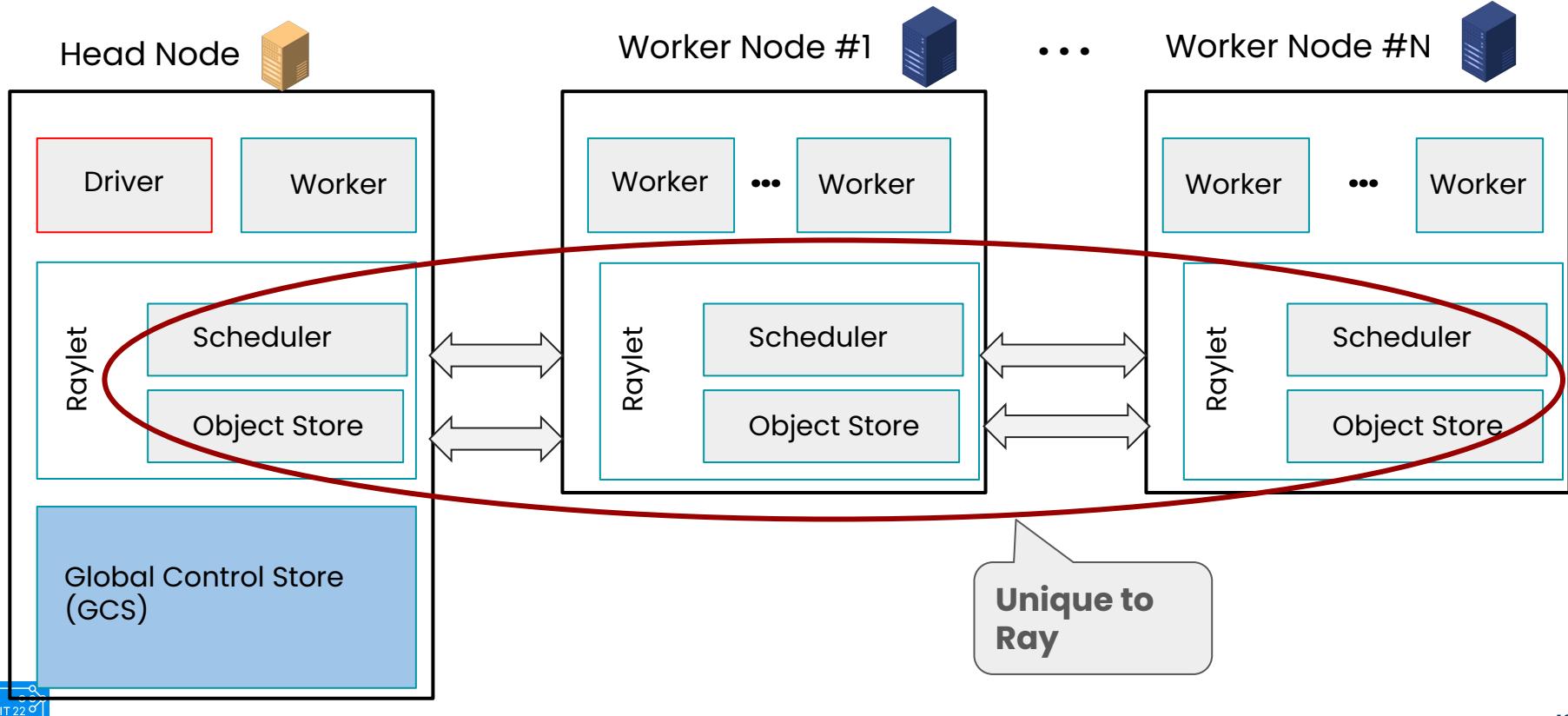
McKinsey  
& Company

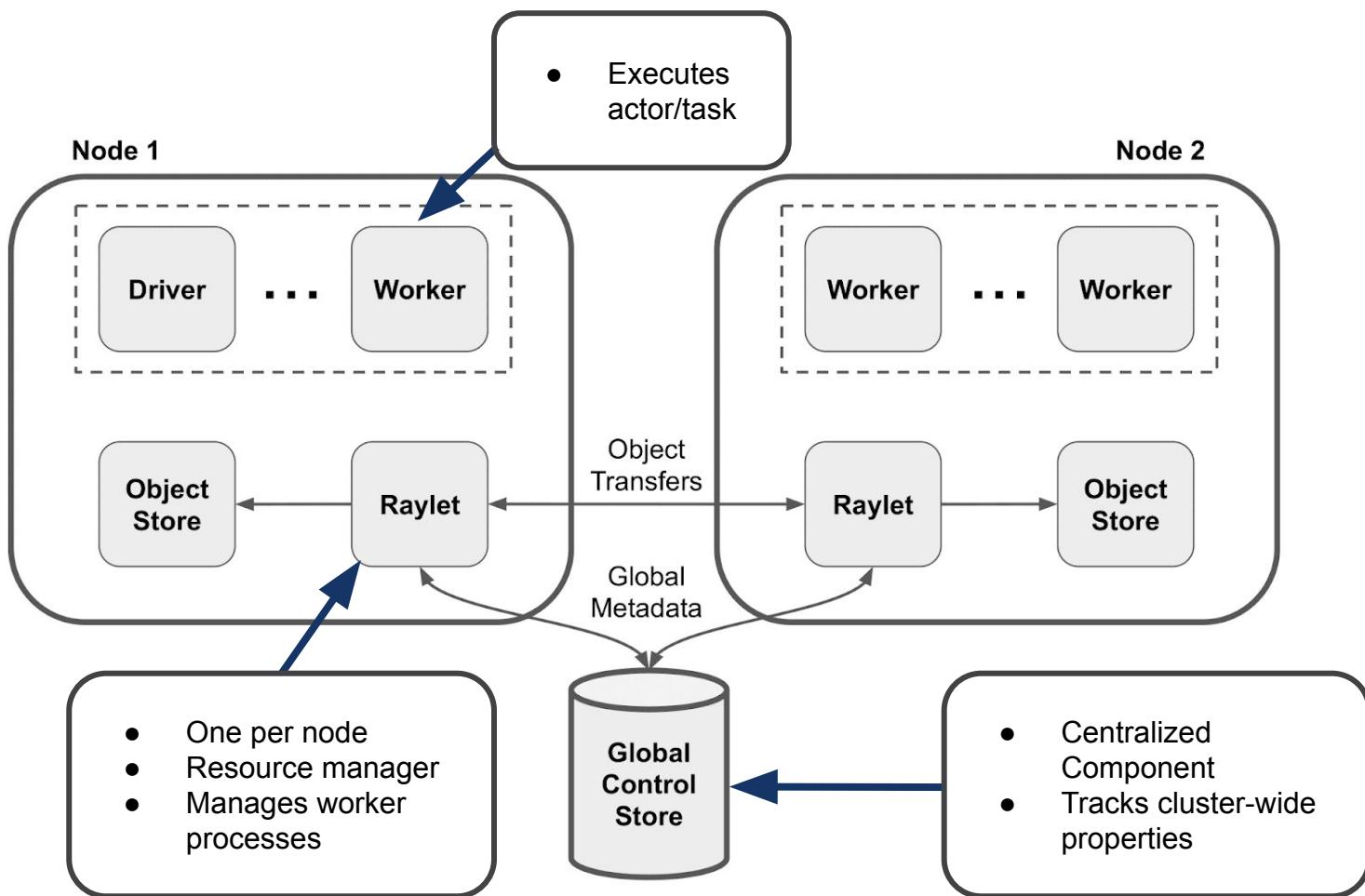


# Ray Architecture & Components



# An anatomy of a Ray cluster



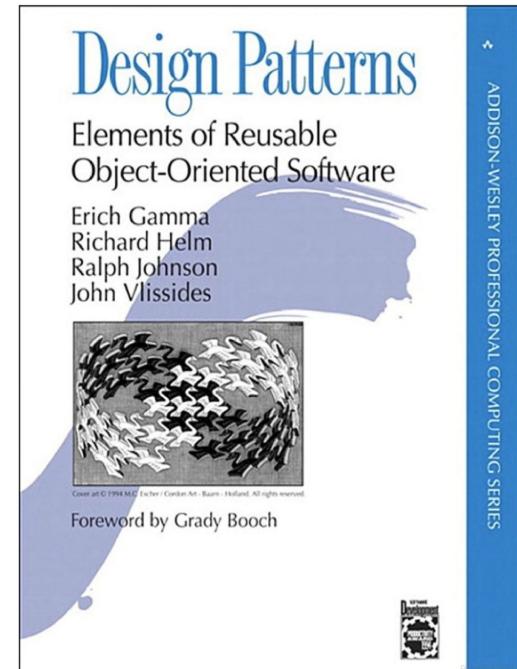


# Ray distributed design patterns & APIs



# Ray Basic Design Patterns

- Ray Parallel Tasks
  - + Functions as stateless units of execution
  - + Functions distributed across the cluster as tasks
- Ray Objects as Futures
  - + Distributed (immutable objects) store in the cluster
  - + Fetched when materialized
  - + Enable massive asynchronous parallelism
- Ray Actors
  - + Stateful service on a cluster
  - + Enable Message passing



1. [Patterns for Parallel Programming](#)
2. [Ray Design Patterns](#)
3. [Ray Distributed Library Integration Patterns](#)

# Python → Ray APIs



```
def f(x):
    # do something with
    x:
        y = ...
    return y
```

Task

```
@ray.remote
def f(x):
    # do something with
    x:
        Y = ...
    return y
```

Distributed

f()  
Node

f()  
Node

```
class Cls():
    def
    __init__(self, x):
    def f(self, a):
        ...
    def g(self, a):
        ...
```

Actor

```
@ray.remote
class Cls():
    def __init__(self,
    x):
        def f(self, a):
            ...
        def g(self, a):
            ...
            ...
```

Distributed

Cls  
Node

Cls()  
Node

```
import numpy as np
a= np.arange(1, 10e6)
b = a * 2
```

Distributed  
immutable  
object



```
import numpy as np
a = np.arange(1, 10e6)
obj_a = ray.put(a)
b = ray.get(obj_a) * 2
```

Distributed

a  
Node

a  
Node

# Function → Task

```
@ray.remote
def read_array(file):
    # read ndarray "a"
    # from "file"
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

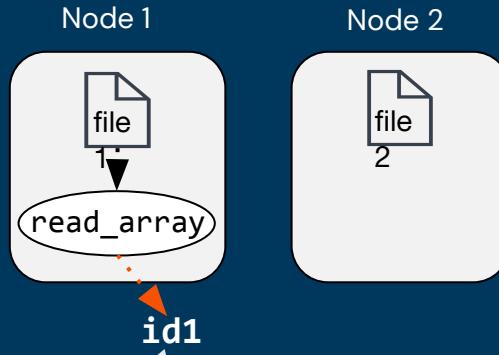
# Class → Actor

```
@ray.remote(num_gpus=1)
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value
```

```
c = Counter.remote()
id4 = c.inc.remote()
id5 = c.inc.remote()
```

# Task API

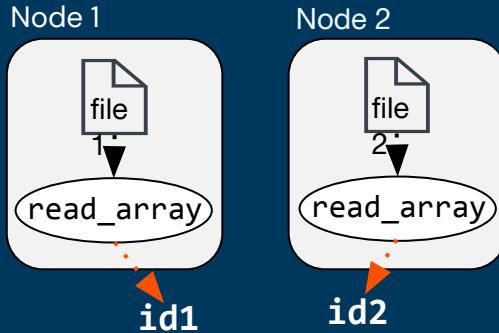
```
@ray.remote  
def read_array(file):  
    # read ndarray "a"  
    # from "file"  
    return a  
  
@ray.remote  
def add(a, b):  
    return np.add(a, b)  
  
id1 = read_array.remote(file1)  
id2 = read_array.remote(file2)  
id = add.remote(id1, id2)  
sum = ray.get(id)
```



Return **id1** (future) immediately,  
before read\_array() finishes

# Task API

```
@ray.remote  
def read_array(file):  
    # read ndarray "a"  
    # from "file"  
    return a  
  
@ray.remote  
def add(a, b):  
    return np.add(a, b)  
  
id1 = read_array.remote(file1)  
id2 = read_array.remote(file2)  
id = add.remote(id1, id2)  
sum = ray.get(id)
```



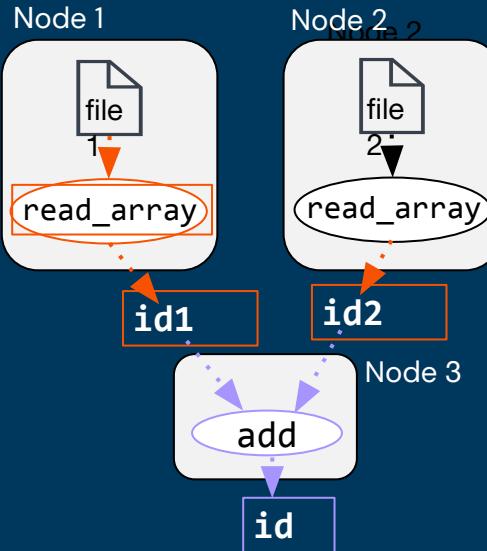
Dynamic task graph:  
build at runtime

# Task API

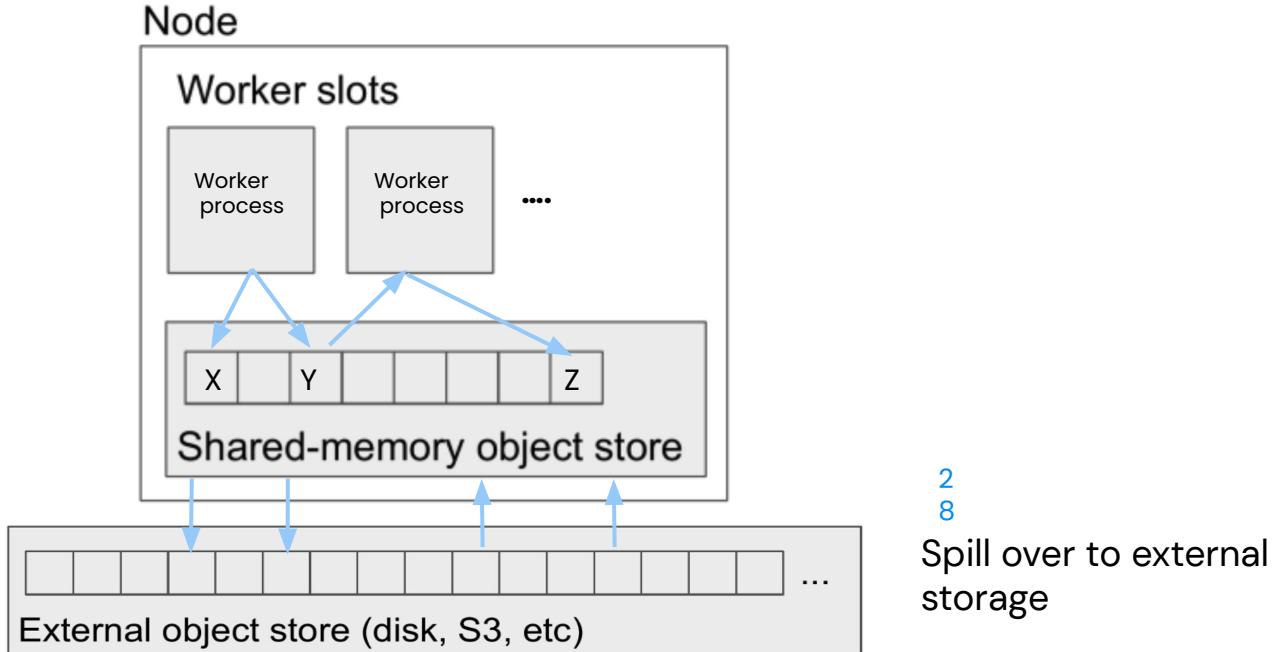
```
@ray.remote
def read_array(file):
    # read ndarray "a"
    # from "file"
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

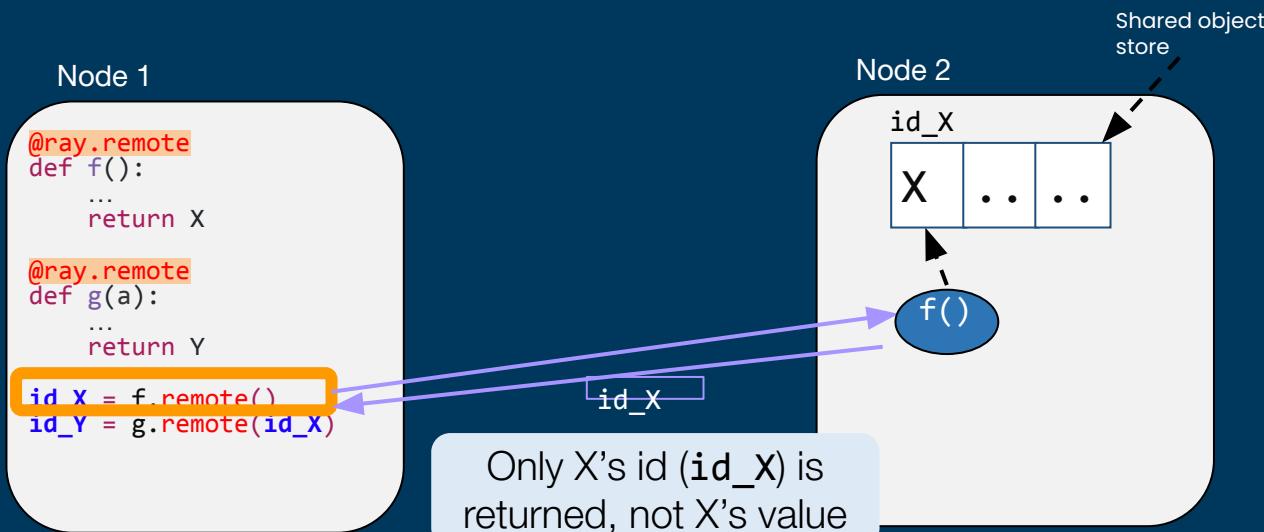
id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id) → ray.get() block until
                           result available
```



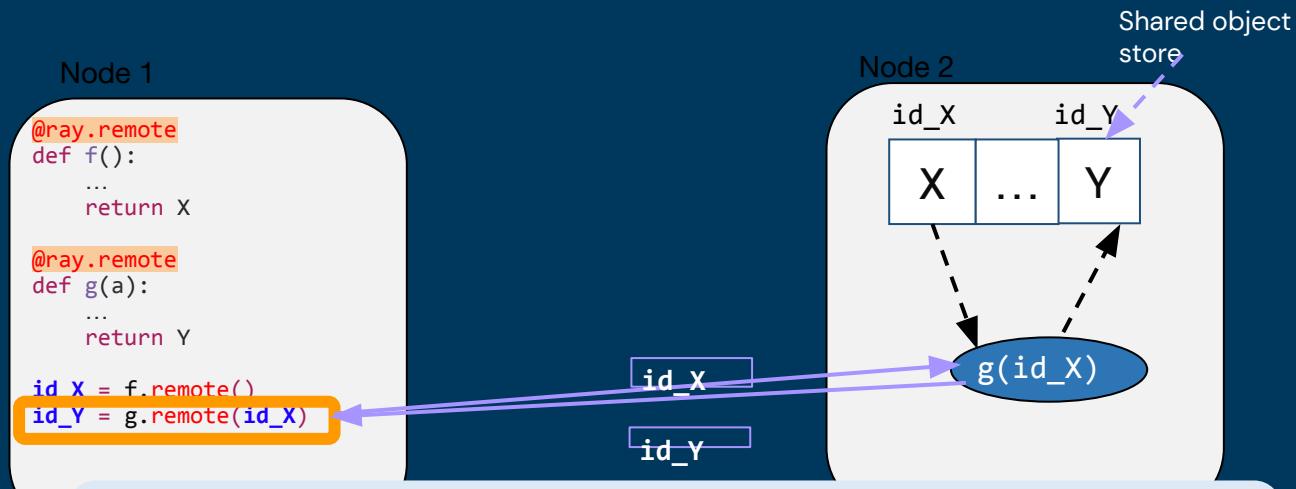
# Distributed Immutable object store



# Distributed object store

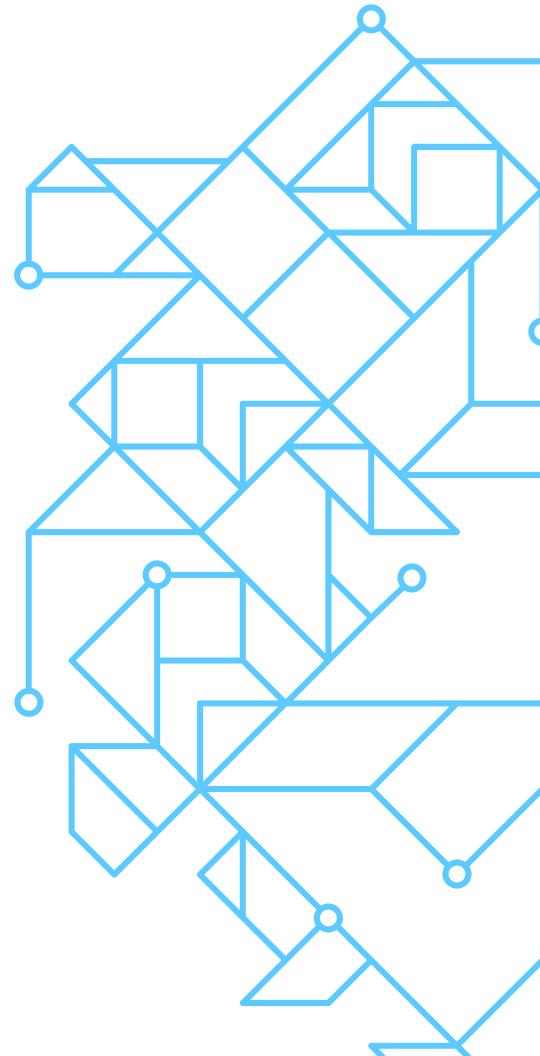


# Distributed object store



`g(id_X)` is scheduled on same node, so `X` is never transferred

# Examples of Distributed Applications with Ray



# Distributed Applications with Ray

## ML Libraries

- Ray AI Runtime
  - Ray Data, Tune, Train, Serve, RLlib
- Distributed scikit-learn/Joblib
- Distributed XGBoost on Ray
- Dask on Ray
- Modin on Ray

All using Ray design patterns

## Monitoring Services

- WhyLabs
- Arize AI
- W & B

All using Ray design patterns

## ML Platforms & Integrations

- Merlin (Shopify)
- Zero Copy (IBM)
- TorchX
- MLflow, Comet
- AirFlow
- HuggingFace
- Pycaret
- Ludwig AI

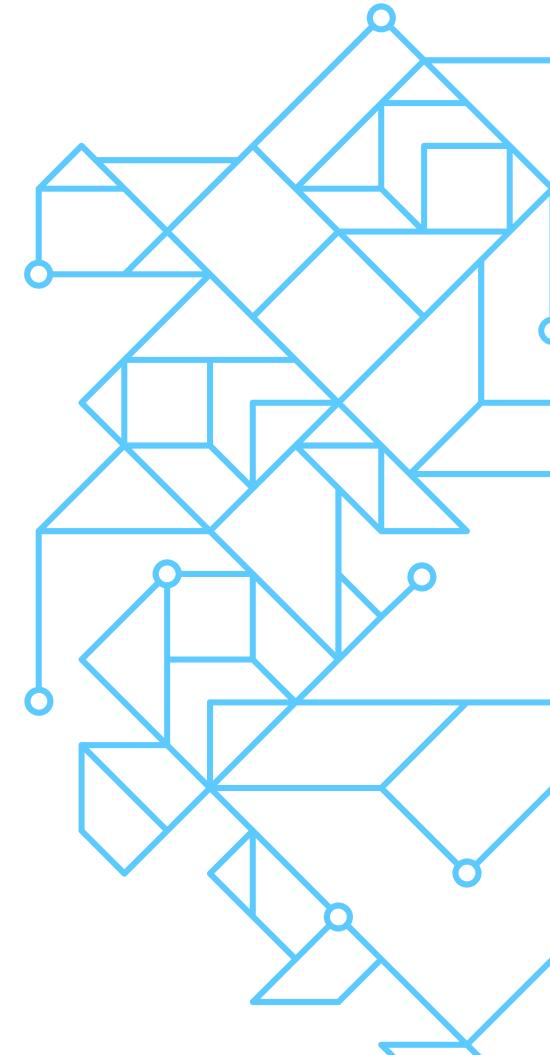
All using Ray design patterns

Ray Ecosystem: <https://docs.ray.io/en/latest/ray-overview/ray-libraries.html>

# Key Takeaways

- Distributed computing is a necessity & norm
- Ray's vision: make distributed computing simple
  - + Don't have to be distributed programming expert
- Build your own disruptive apps & libraries with Ray
- Scale your ML workloads with Ray libraries (Ray AIR)

Let's go with



# Anyscale User/Password

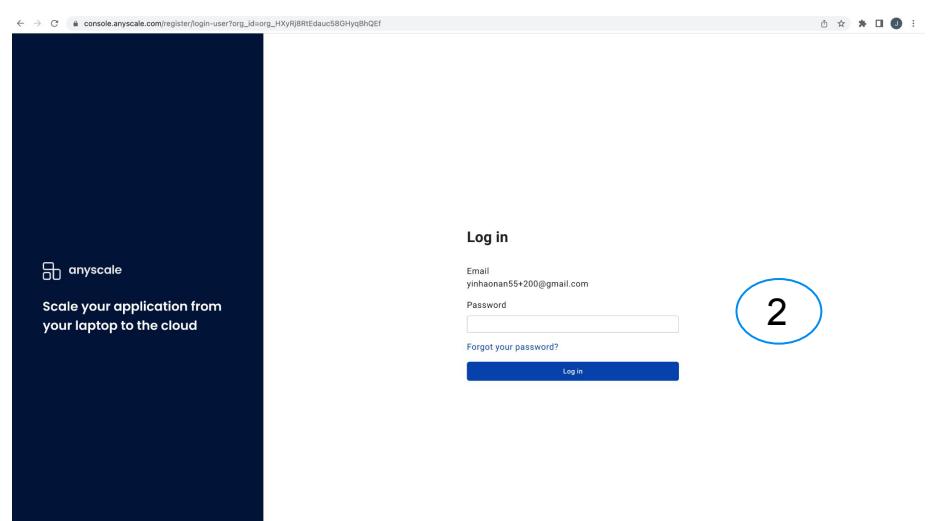
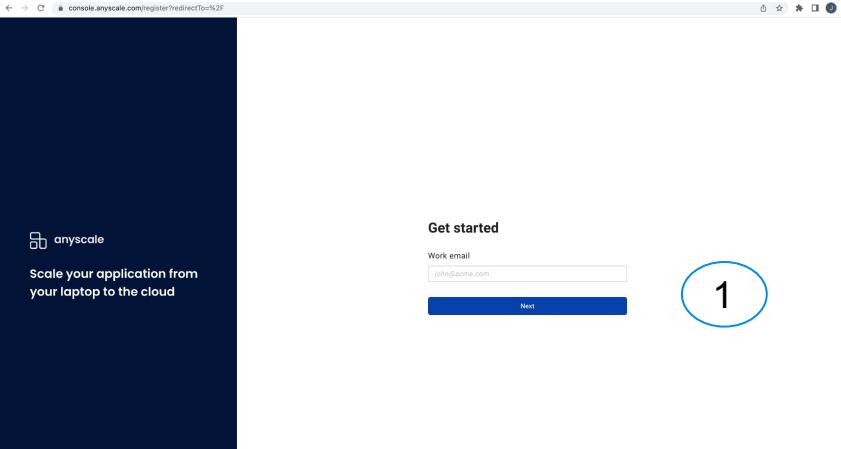


<https://bit.ly/rsummit2022-class-logins>

- Choose any line from spreadsheet under your class name: <your class name here>
- In column “Account” switch to “Claimed”
- For example, Username/password: [yinhaonan55+520@gmail.com](mailto:yinhaonan55+520@gmail.com)/[tutorialpassword520](#)

# Your Anyscale Cluster

- Console: <http://console.anyscale.com/>
- User name: <[username@gmail.com](mailto:username@gmail.com)>
- Password : password

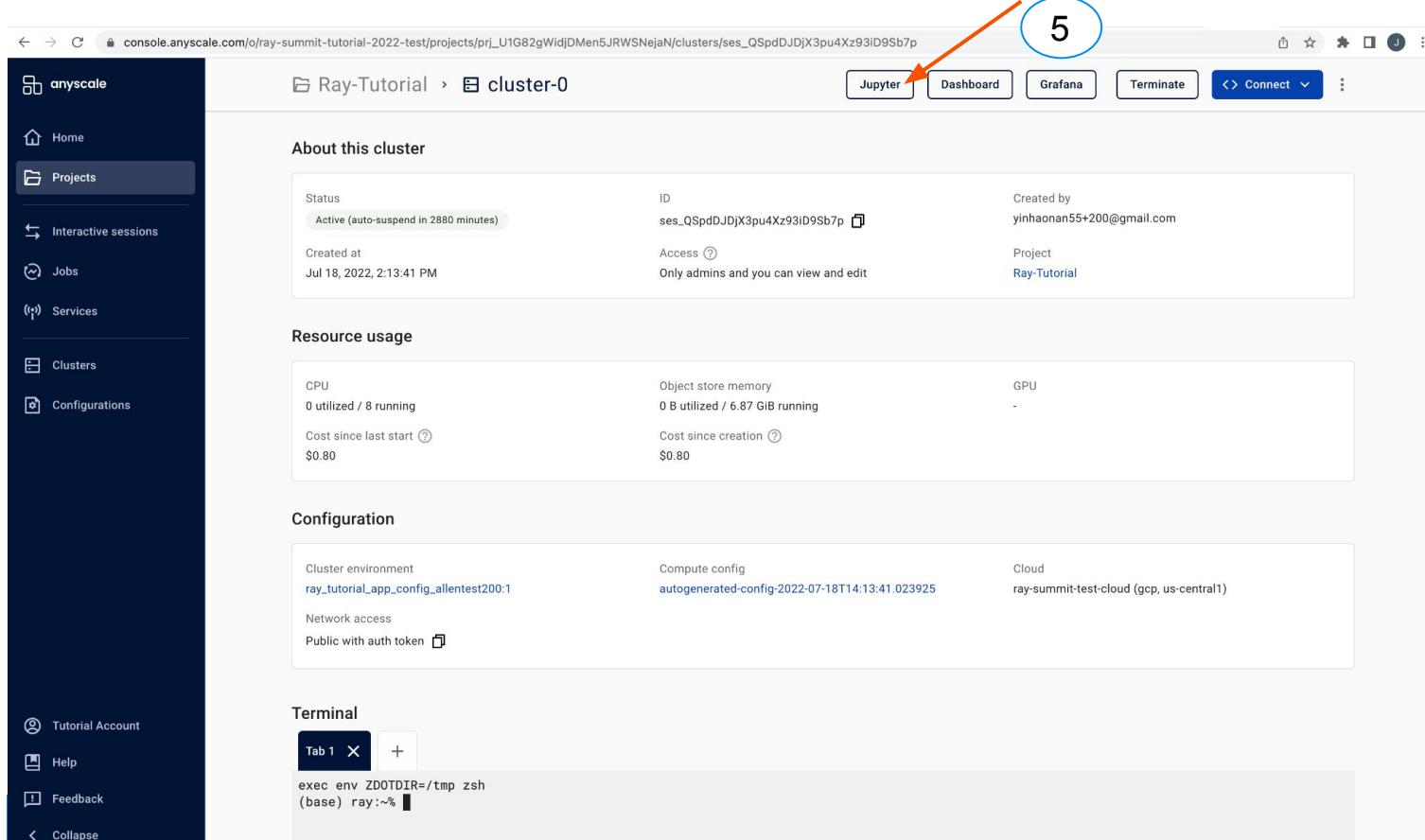


# Your Anyscale Cluster

The screenshot shows the Anyscale console interface. On the left, a dark sidebar menu lists various options: Home, Projects, Interactive sessions, Jobs, Services, Clusters (which is highlighted with a blue circle and labeled '3'), and Configurations. At the bottom of the sidebar are links for Tutorial Account, Help, Feedback, and Collapse. The main content area is titled 'Clusters' and displays a table of existing clusters. The table has columns for Name, Status, Active resources, Cost, Cluster environment, Project, Cloud, Created by, and Created at. One cluster, named 'cluster-0', is listed with a status of 'Terminated', no active resources, a cost of '\$0.80', and details about its environment and creators. At the top of the cluster table, there are buttons for '+ Create', 'Start', 'Terminate' (which is circled in blue and labeled '4'), and 'Archive'. There are also filters for 'Search names', 'Cluster status', 'Created by', and 'Include archived'.

Name	Status	Active resources	Cost	Cluster environment	Project	Cloud	Created by	Created at
cluster-0	Terminated	None	\$0.80	ray_tutorial_app_config_allentest200:1	Ray-Tutorial	ray-summit-test-cloud (GCP)	Me	7/18

# Your Anyscale Cluster



The screenshot shows the Anyscale Cluster console interface. A red arrow points from a blue circle containing the number 5 to the "Jupyter" button in the top navigation bar. The "Jupyter" button is highlighted with a black border.

console.anyscale.com/o/ray-summit-tutorial-2022-test/projects/prj\_U1G82gWidjDMen5JRWSNejaN/clusters/ses\_QSpdDJDX3pu4Xz93iD9Sb7p

Ray-Tutorial > cluster-0

**Jupyter** **Dashboard** **Grafana** **Terminate** **Connect** :

### About this cluster

Status	ID	Created by
Active (auto-suspend in 2880 minutes)	ses_QSpdDJDX3pu4Xz93iD9Sb7p	yinhanan55+200@gmail.com
Created at	Access	Project
Jul 18, 2022, 2:13:41 PM	Only admins and you can view and edit	Ray-Tutorial

### Resource usage

CPU	Object store memory	GPU
0 utilized / 8 running	0 B utilized / 6.87 GiB running	-
Cost since last start	Cost since creation	
\$0.80	\$0.80	

### Configuration

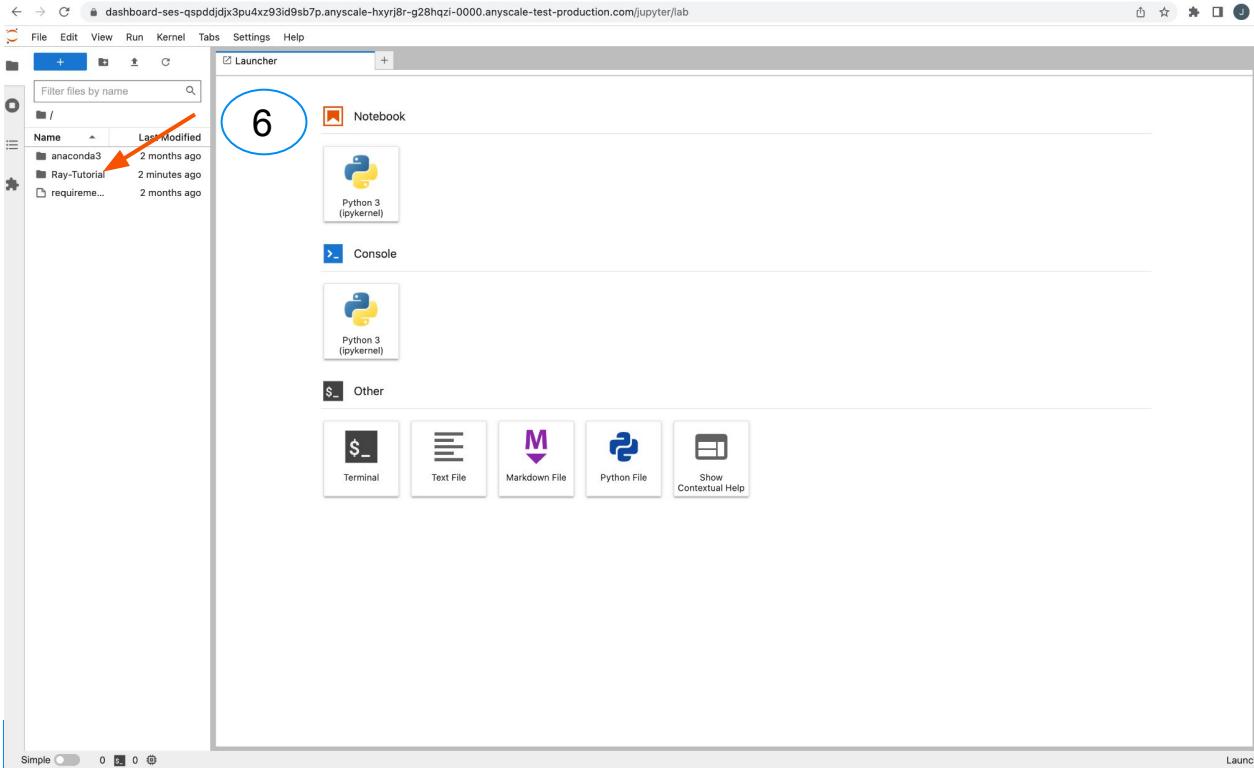
Cluster environment	Compute config	Cloud
ray_tutorial_app_config_alltest200:1	autogenerated-config-2022-07-18T14:13:41.023925	ray-summit-test-cloud (gcp, us-central1)
Network access		
Public with auth token		

### Terminal

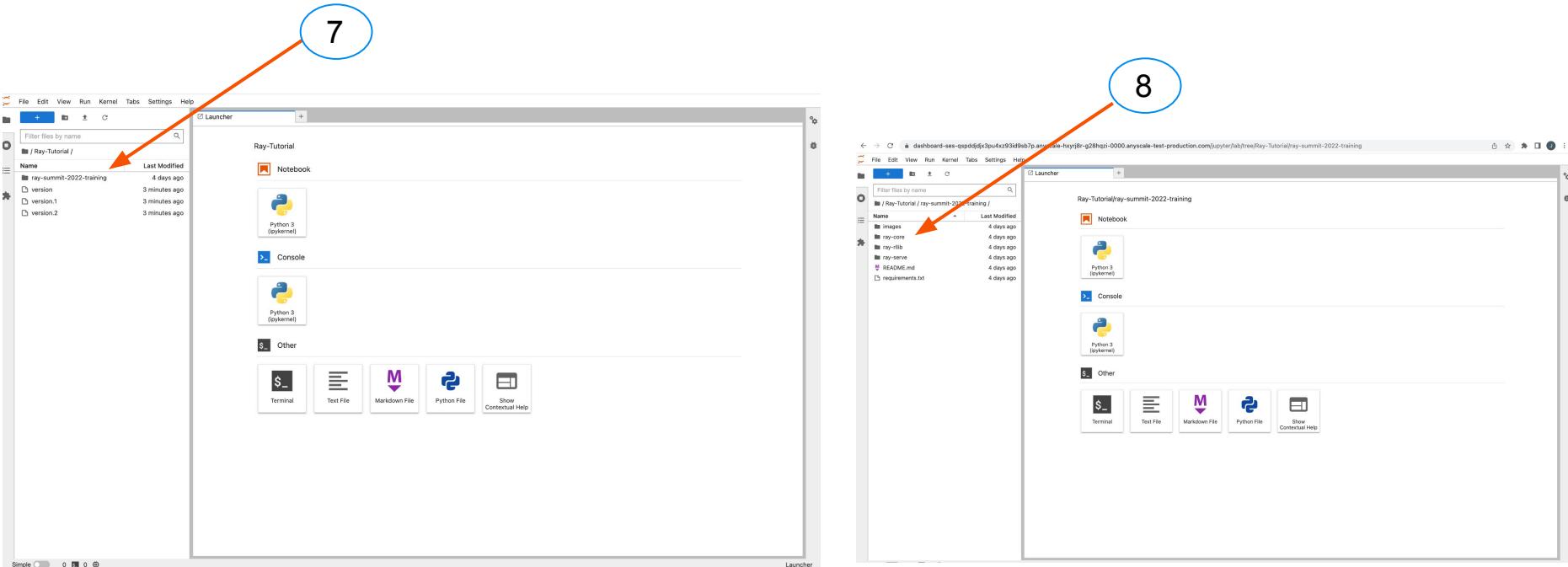
Tab 1 **+ *[redacted]***

```
exec env ZDOTDIR=/tmp zsh
(base) ray:~%
```

# Your Anyscale Cluster



# Your Anyscale Cluster



# Thank you.

Tell us what you think...

<https://bit.ly/ray-core-summit2022>



# Tell us what you think...

<https://bit.ly/ray-core-summit2022>

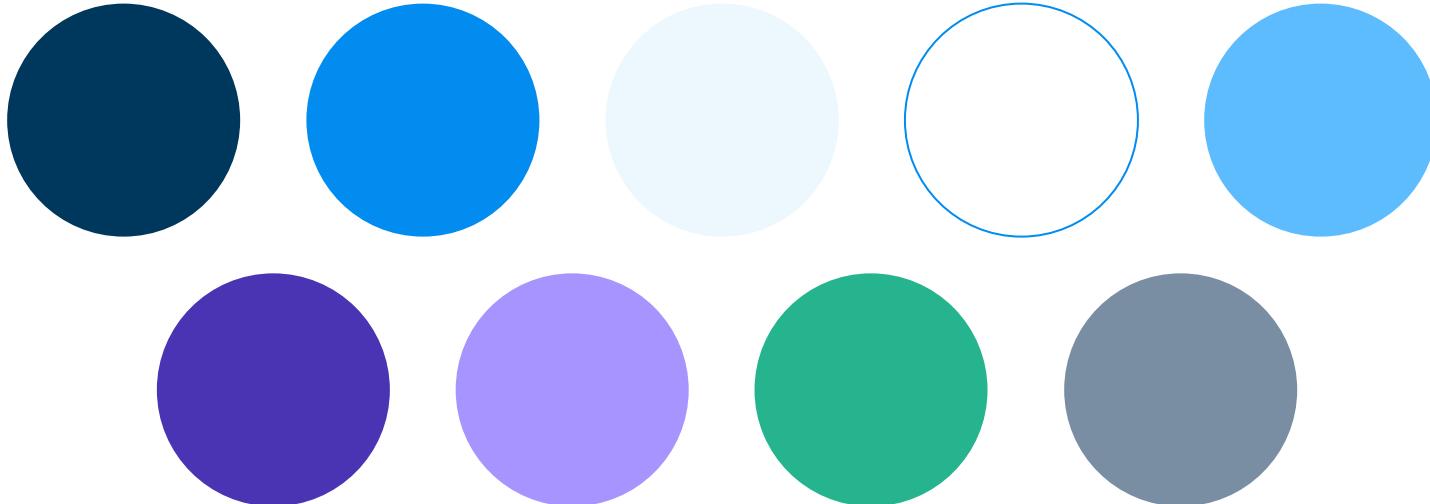


# Here is the Title Slide

Firstname Lastname, Company



# Colors



# Here is a basic Dark Slide

# Here is a Basic Light Slide

# How about a slide with 2 options?

## Here is an info card

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua Ut enim ad minim veniam, quis nostrud exercitation

## Here is an info card

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua Ut enim ad minim veniam, quis nostrud exercitation

# How about a slide with 3?

## Here is an info card

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit, sed do  
eiusmod tempor incididunt ut  
labore et dolore magna aliqua Ut  
enim ad minim veniam, quis  
nostrud exercitation

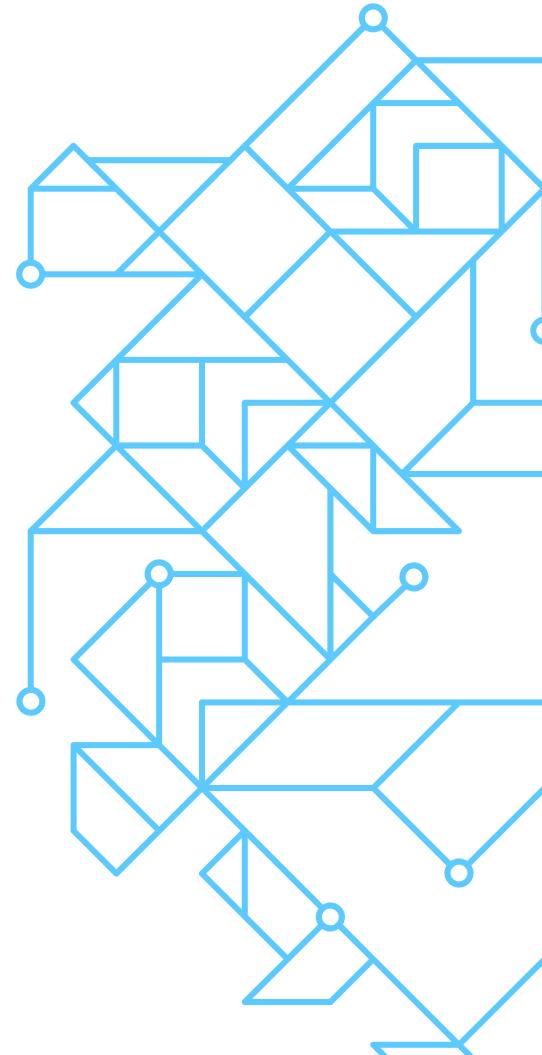
## Here is an info card

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit, sed do  
eiusmod tempor incididunt ut  
labore et dolore magna aliqua Ut  
enim ad minim veniam, quis  
nostrud exercitation

## Here is an info card

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit, sed do  
eiusmod tempor incididunt ut  
labore et dolore magna aliqua Ut  
enim ad minim veniam, quis  
nostrud exercitation

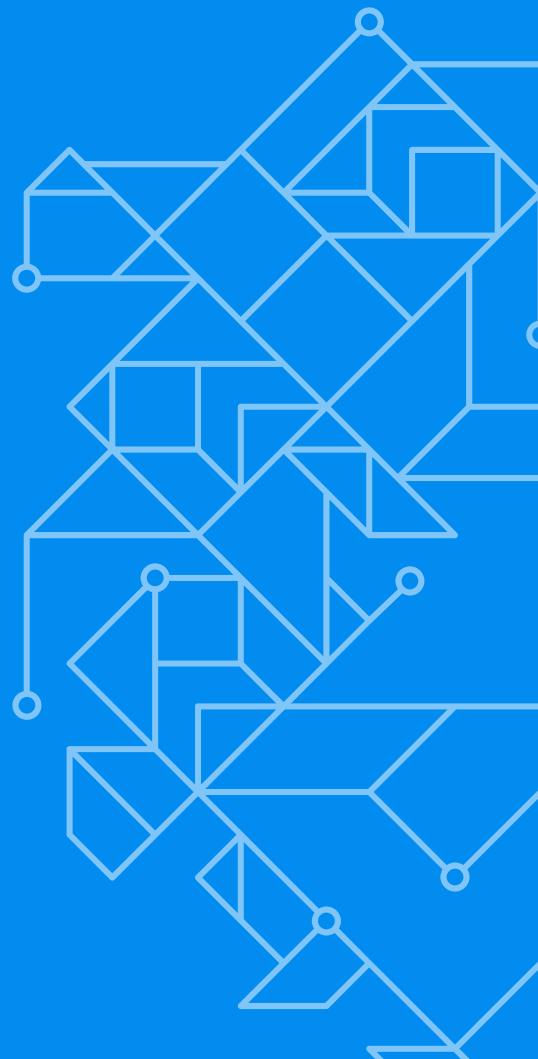
# Here is a Section Header



# Here is a Section Header



# Here is a Section Header



# Thank you.

Follow up information can go here.

