

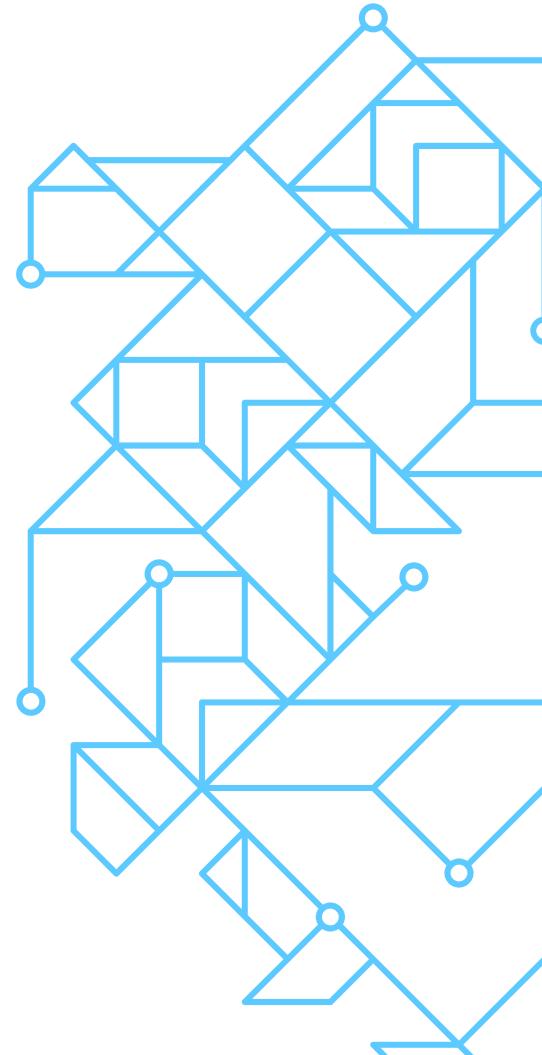
Ray Workshop

Scaling Computer Vision Workloads



Welcome!

We're happy to have you here.





Meet the team!



Kamil



Emmy



Jules



Ricky



Our goals.



Introduce Ray.

Overview of Ray for scaling computer vision workloads



Learn by doing.

Hands-on, relevant coding examples.



Reinforce through discussion.

Polls, quizzes, live Q&A, conversation at tables.

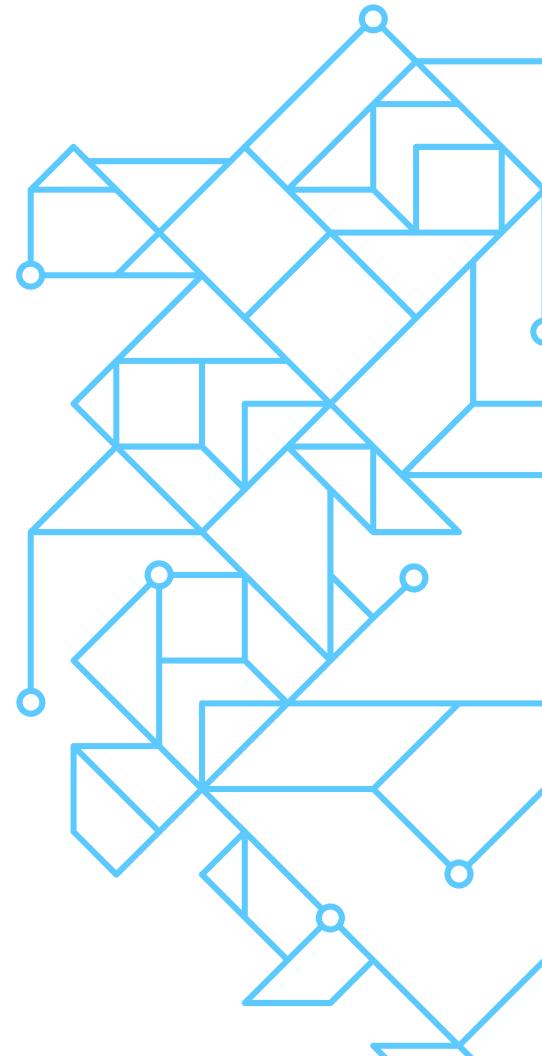


Cultivate community.

Find friends, network, knowledge share.

The Plan

Here's what to expect today.





Today's agenda.

Time	Description
10:00am	Welcome and introductions
10:30am	Break
11:00am	Course set-up
11:20am	Overview of Ray
12:00pm	Lunch
1:00pm	Overview of Ray

Time	Description
1:30pm	Ray Core
2:00pm	Break
2:30pm	Ray Core
3:30pm	Break
4:00pm	Introduction to Ray AIR
5:30pm	Wrap up and survey



Preview of tomorrow's agenda.

Time	Description
10:00am	Welcome back
10:20am	Distributed model training
12:00pm	Lunch
1:00pm	Observability on Ray

Time	Description
2:00pm	Break
2:30pm	Scaling batch inference
4:00pm	Break
4:30pm	Wrap up and Survey
4:40pm	What's next?



Important links.



[GitHub repository](#)

Access all the notebooks, slides, and scripts.



[Ray documentation](#)

API references and user guides.



[Slido.com](#)

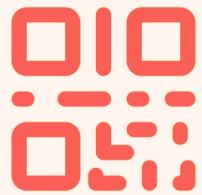
Participate in polls, quizzes, and Q&A.



[Survey](#)

One feedback survey per day.

slido



Join at [slido.com](https://www.slido.com)
Enter code #ray

- ① Start presenting to display the joining instructions on this slide.



Prizes.



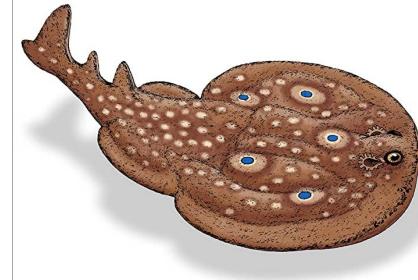
“Learning Ray” books

For the top finishers in our daily quiz.

O'REILLY®

Learning Ray

Flexible Distributed Python for Machine Learning



Max Pumperla,
Edward Oakes
& Richard Liaw
Foreword by Ion Stoica



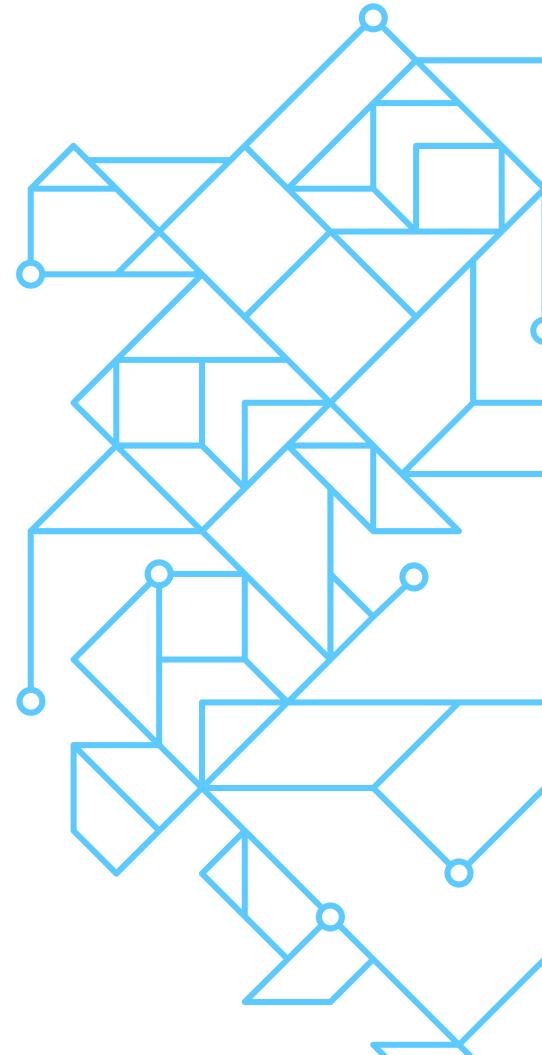
Ray Summit 2023 tickets

A random draw from survey submissions.



Introductions

Meet your neighbors!





Icebreaker.

- ⛏ Pick a neighbor or two.
- ⏰ Spend 5 min getting to know each other. Respond to the poll.
- 📣 We'll all come together, so you can introduce your neighbor to everyone!



What is a fun fact that you learned about your neighbor?

- ① Start presenting to display the poll results on this slide.



Icebreaker.

“Hi my name is [*your name*], and I would like to introduce you to [*their name*].

Some interesting things about my neighbor are...”

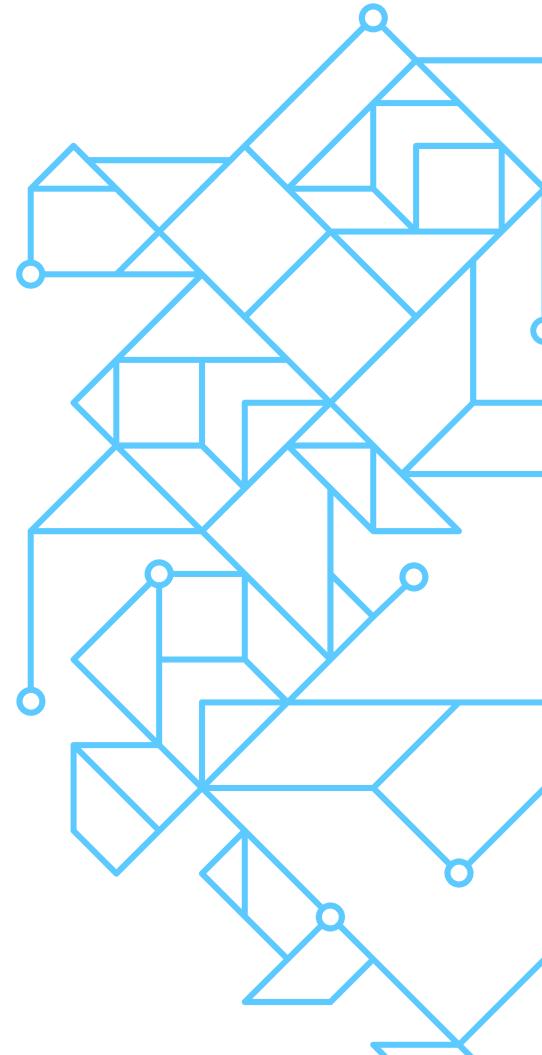


**Time for a
Break!**

30 minutes.

Course Logistics

How to access your cluster and other resources.





Tech check.



Participating via slido.com

- Join with code #ray
- Answer polls and quizzes.
 - Enter your name to compete for prizes.
- Ask questions.
 - Pose your own and upvote others.
 - TAs will be answering questions on a rolling basis.



Tech check.



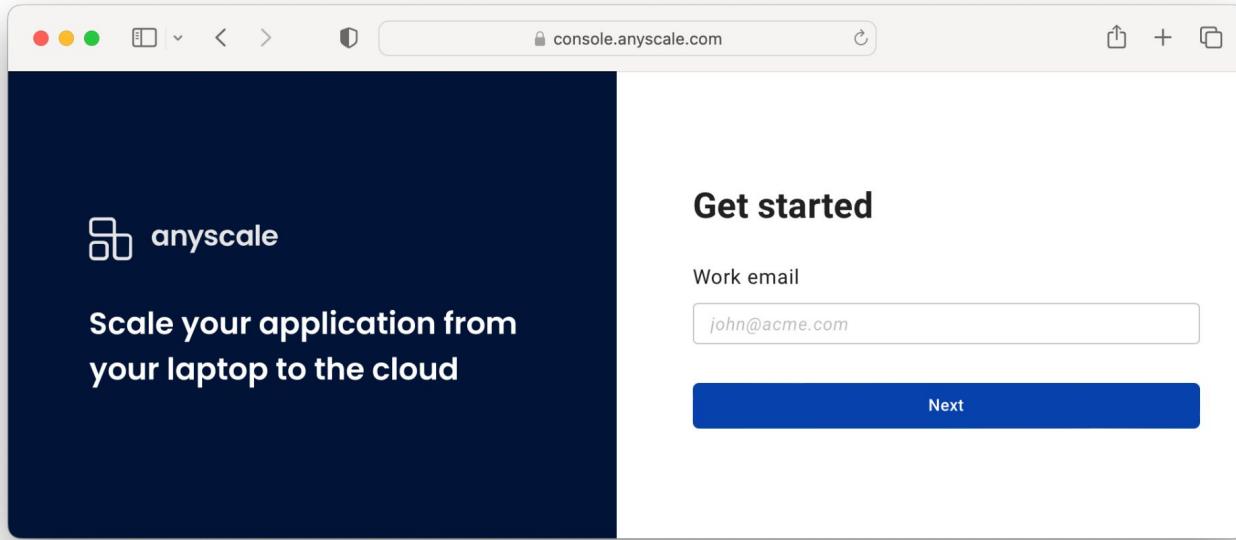
Accessing Anyscale clusters.

- All work will be in Anyscale provisioned clusters.
- Our GitHub repo will be mounted automatically.
- Access begins now.
 - Check your email for login information.
 - Step-by-step instructions to follow.



Anyscale Login

Link to Anyscale cluster: console.anyscale.com



Check your
email for your
unique
username and
password!

select
“Clusters”

The screenshot shows the Anyscale web console interface. On the left, a sidebar menu for the "anyscale" organization includes options like Home, Projects, Workspaces, Interactive sessions, Jobs, Services, Clusters (which is highlighted with an orange box), and Configurations. On the right, the main "Clusters" page displays a table of existing clusters. The table has columns for Name, Status, and Active resources. One cluster, "ray-acm-emmy", is highlighted with an orange box. A curved arrow points from the "Clusters" text on the left to this highlighted cluster. Another curved arrow points from the "click on your cluster" text below to the same cluster row.

<input type="checkbox"/>	Name	Status ↓	Active resources
<input type="checkbox"/>	ray-acm-emmy	Active (auto-terminates in 117 minutes)	0 cpu

click on your
cluster

“Start” the cluster,
then
select
“Jupyter”

The screenshot shows the anyscale console interface. On the left is a sidebar with options like Home, Projects (selected), Workspaces, Interactive sessions, Jobs, Services (selected), Clusters, Configurations, and user information (Emmy Li). The main content area shows a cluster named 'emmy-ray-saturday'. At the top, there are buttons for Jupyter, Dashboard, Grafana, Terminate, Connect, and more. A red box highlights the 'Jupyter' button. Below it, the 'About this cluster' section provides details: Status (Active), ID (ses_z5yfnmzamcpxc4uk95mezhr4), Created by (emmy@anyscale.com), Created at (Dec 8, 2022 at 10:01:54 AM), Access (Everyone in your organization can view a...), Project (ray-saturday). The 'Resource usage' section shows CPU, Object store memory, and GPU all at 0%. The 'Cost since last start' is \$0.02 and 'Cost since creation' is \$2.50. The 'Configuration' section lists Cluster environment (ray-sat-v1:10), Compute config (kk-rs-config-for-emmy), and Cloud (anyscale_default_cloud (aws, us-west-2)). The 'Network access' section indicates Public with auth token.

Jupyter

About this cluster

Status: Active

ID: ses_z5yfnmzamcpxc4uk95mezhr4

Created by: emmy@anyscale.com

Created at: Dec 8, 2022 at 10:01:54 AM

Access: Everyone in your organization can view a...

Project: ray-saturday

Resource usage

CPU: -

Object store memory: -

GPU: -

Cost since last start: \$0.02

Cost since creation: \$2.50

Current cluster activity: -

Configuration

Cluster environment: ray-sat-v1:10

Compute config: kk-rs-config-for-emmy

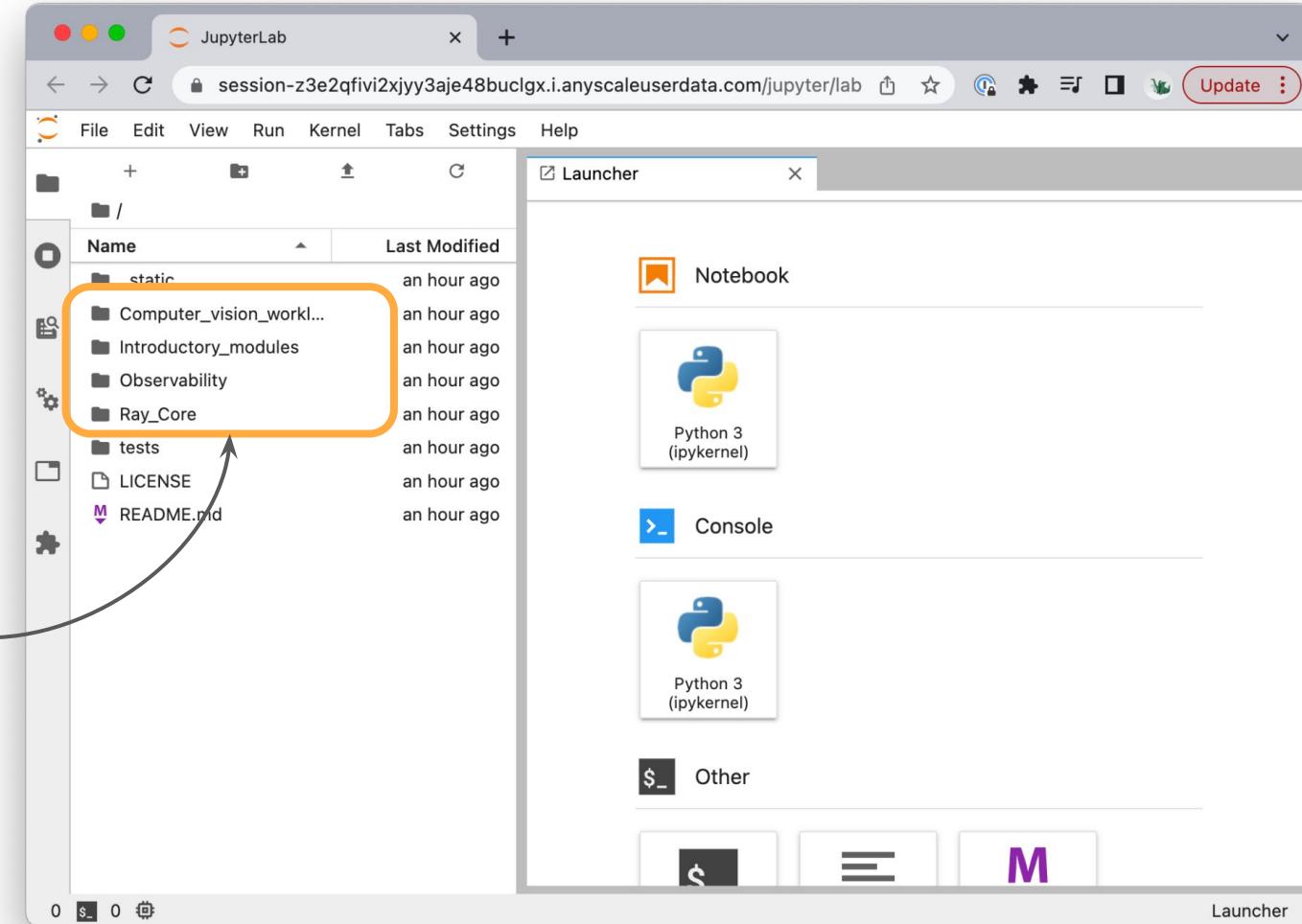
Cloud: anyscale_default_cloud (aws, us-west-2)

Network access

Public with auth token

Terminal

View
modules
here



Launcher

slido



Audience Q&A Session

- ⓘ Start presenting to display the audience questions on this slide.

Overview of Ray

Hands-on coding lab.



Meet me
here!

JupyterLab session-z3e2qfivi2xjyy3aje48buculgx.i.anyscaleuserdata.com/jupyter/lab

File Introductory_modules /

Name Last Modified

- Introduction_to_Ray_AI... an hour ago
- Overview_of_Ray.ipynb an hour ago
- Overview_of_Ray.pdf an hour ago
- quickstart_with_Ray_AIR.py an hour ago
- quickstart_with_Ray_CO...

Launcher quickstart_with_Ray_AIR.py

```
1 import ray
2 from ray.air.config import ScalingConfig
3 from ray.data.preprocessors import MinMaxScaler
4 from ray.train.xgboost import XGBoostTrainer
5
6 # Initialize Ray runtime
7 ray.init()
8
9 #####
10 # DATA #
11 #####
12 # Read Parquet file to Ray Dataset
13 dataset = ray.data.read_parquet(
14     "s3://anyscale-training-data/intro-to-ray-air/nyc_taxi_2021.parquet"
15 )
16
17 # Split data into training and validation subsets
18 train_dataset, valid_dataset =
19 dataset.train_test_split(test_size=0.3)
20
21 # Split datasets into blocks for parallel preprocessing
22 # 'num_blocks' should be lower than number of cores in the cluster
23 train_dataset = train_dataset.repartition(num_blocks=5)
24 valid_dataset = valid_dataset.repartition(num_blocks=5)
25
26 # Define a preprocessor to normalize the columns by their range
27 preprocessor = MinMaxScaler(columns=["trip_distance",
    "trip_duration"])
28
```

0 1 Python Ln 1, Col 1 Spaces: 4 quickstart_with_Ray_AIR.py

slido



Audience Q&A Session

- ⓘ Start presenting to display the audience questions on this slide.



Time for Lunch!

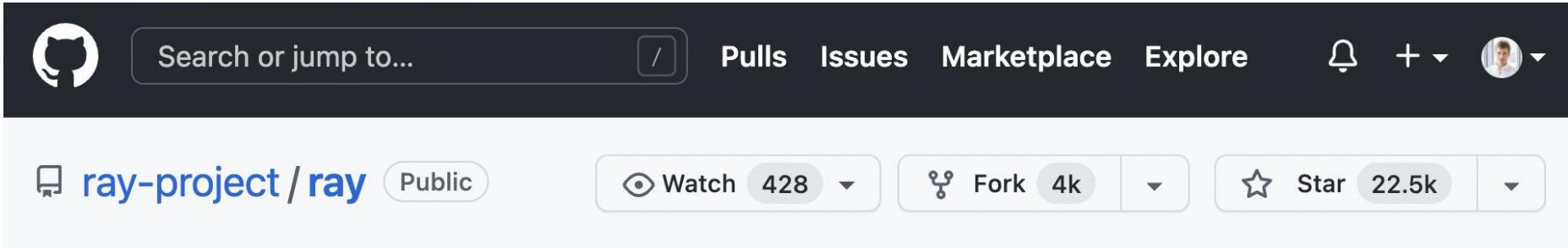
60 minutes.

Overview of Ray

Take a tour through what Ray can do.



Ray project



A screenshot of the GitHub repository page for 'ray-project/ray'. The page shows basic repository statistics: Watch (428), Fork (4k), and Star (22.5k). The GitHub interface includes a search bar, navigation links for Pulls, Issues, Marketplace, and Explore, and a user profile icon.

Ray is:

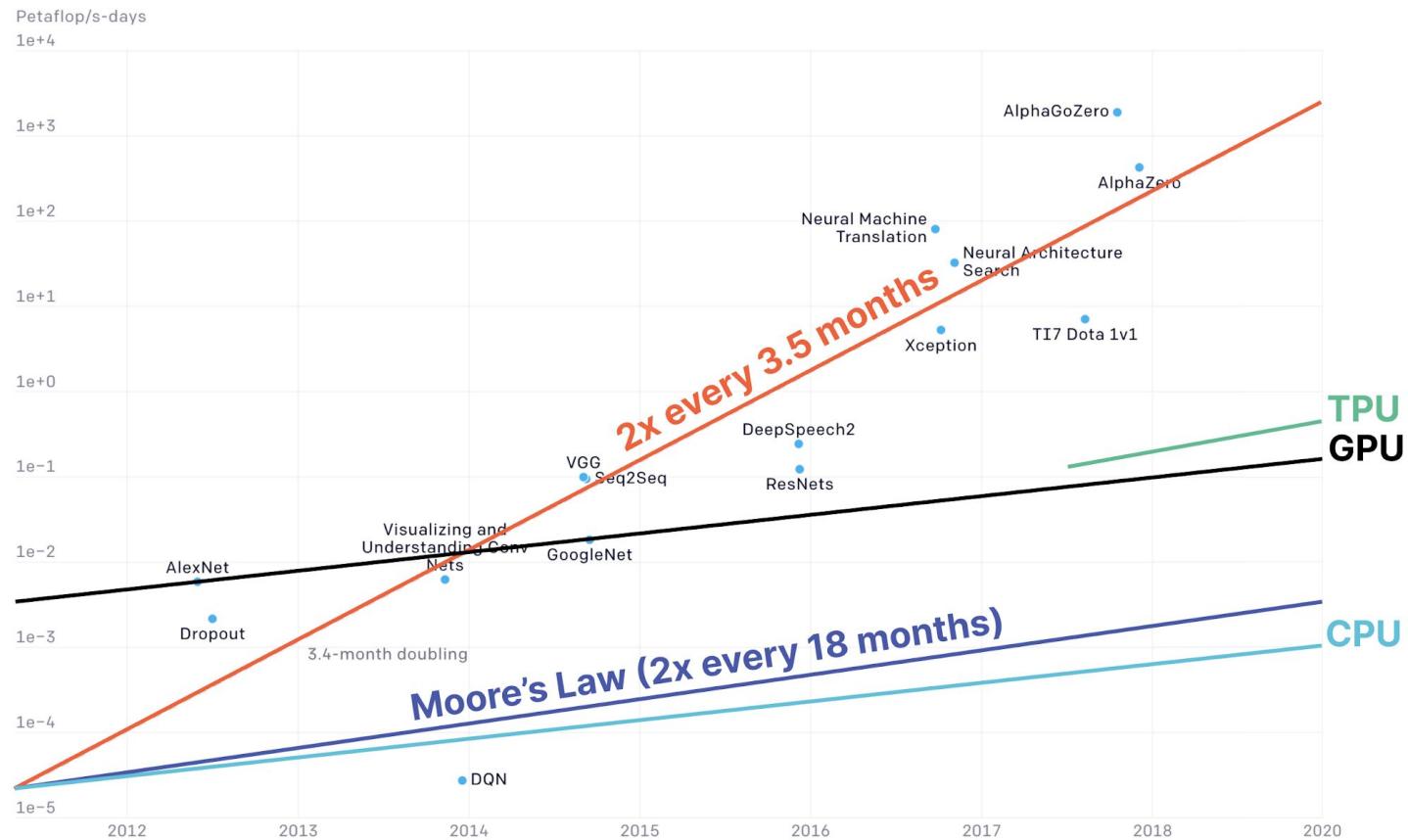
an open-source unified compute framework
that makes it easy to scale AI and Python workloads.

Ray handles

orchestration, scheduling, fault tolerance, auto-scaling and more
so that you can scale your apps without becoming a distributed
systems expert.



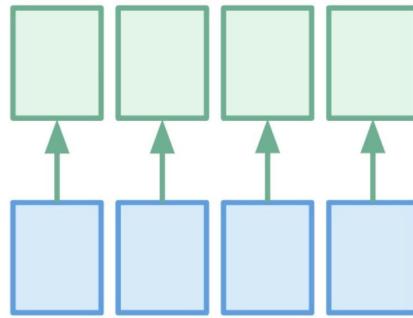
Distributed computing: a bit of context



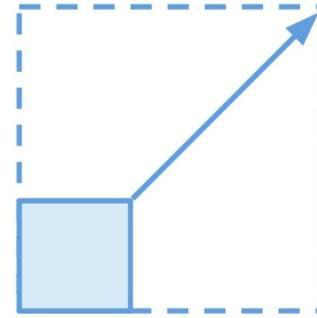
Key Ray characteristics



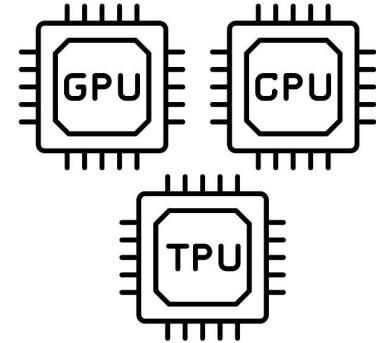
Python first approach



Simple and flexible API

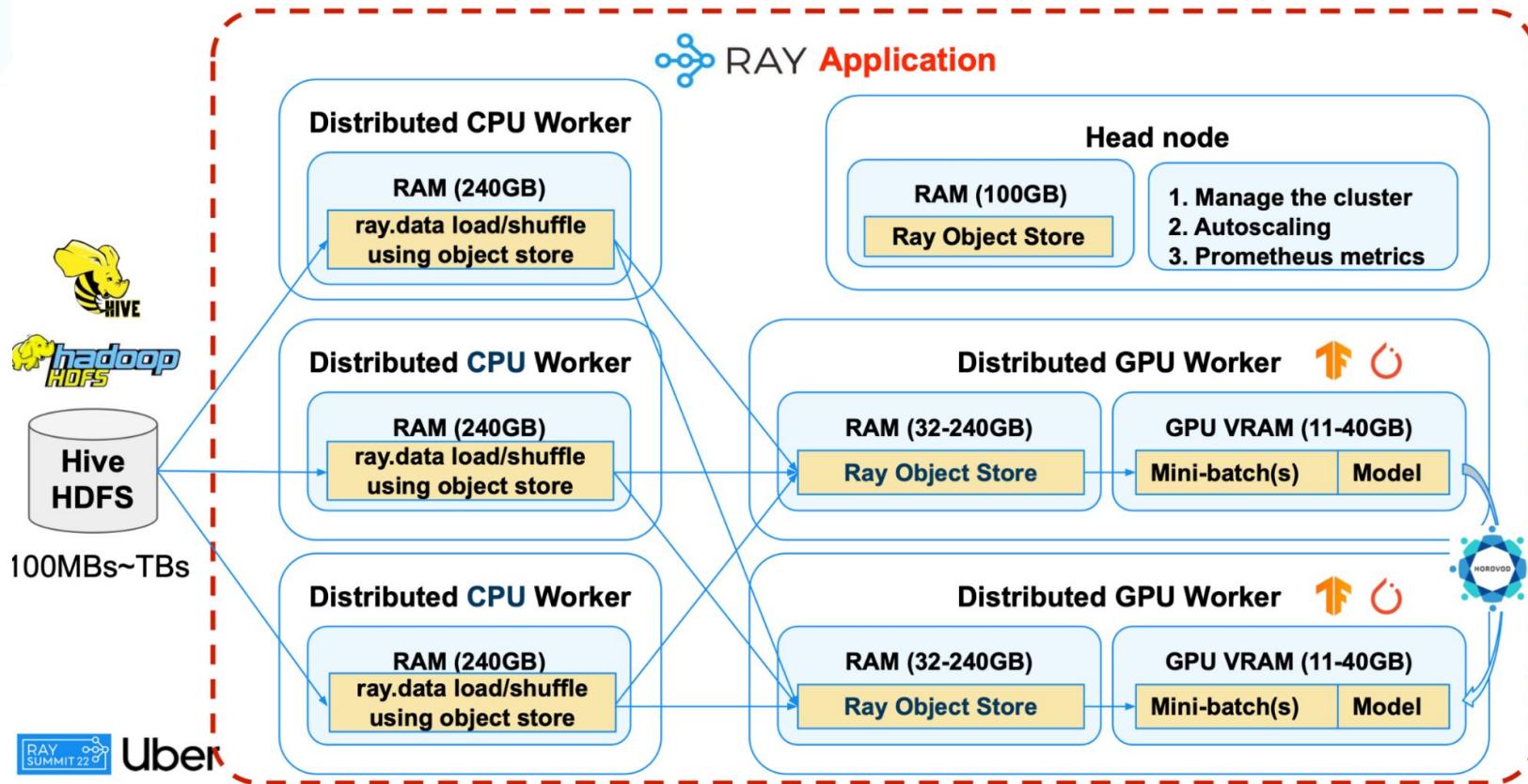


Scalability



Support for heterogeneous
hardware

Deep learning pipeline at Uber



Ray libraries



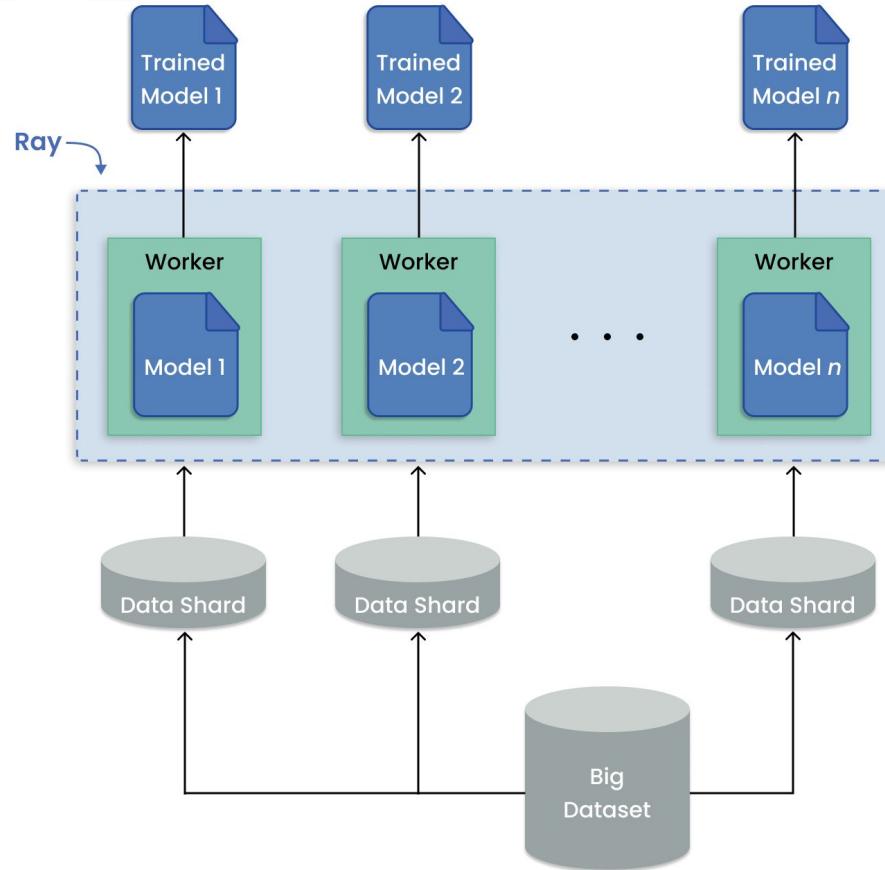
Ray AIR and integrations



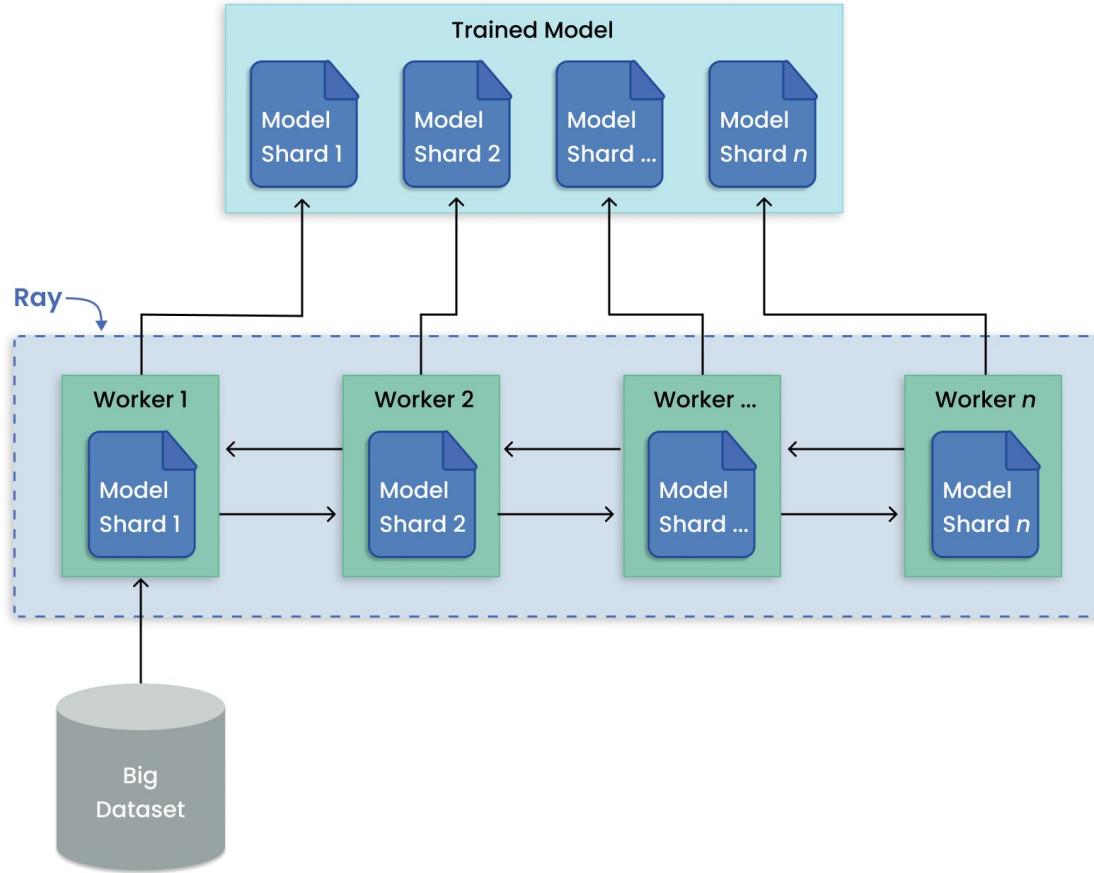
Ray use cases

Scaling ML workloads

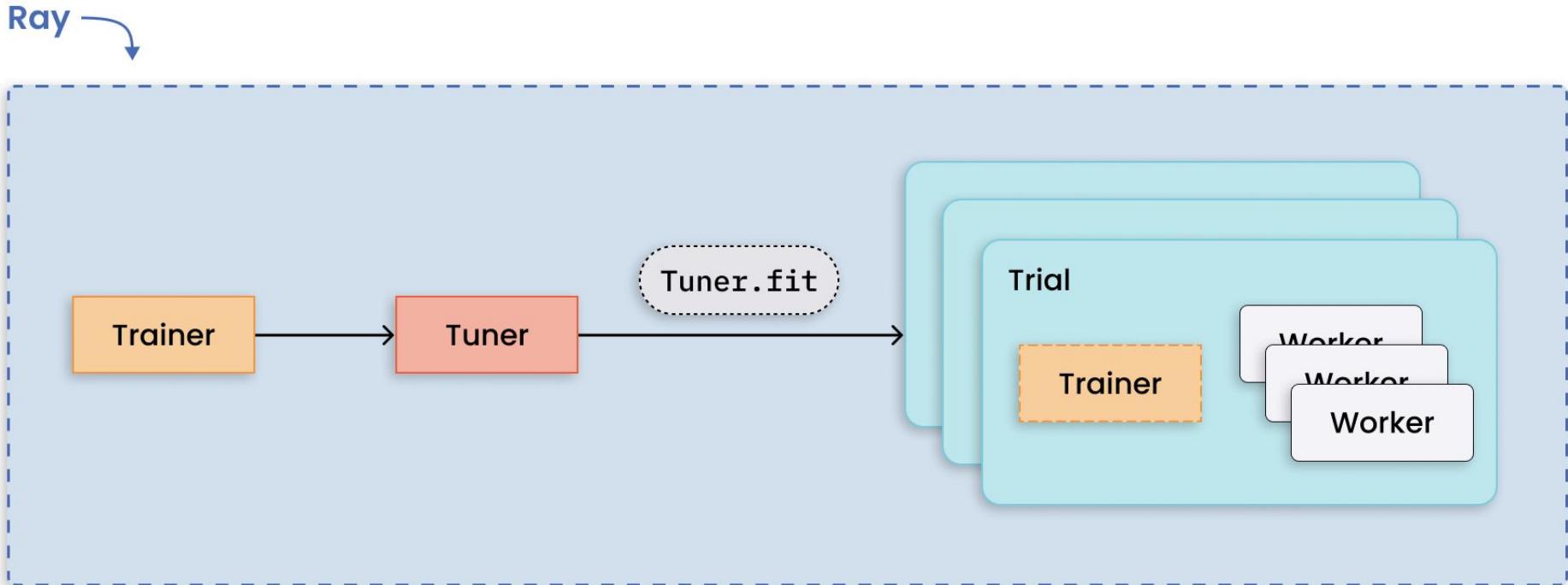
Parallel training of many models



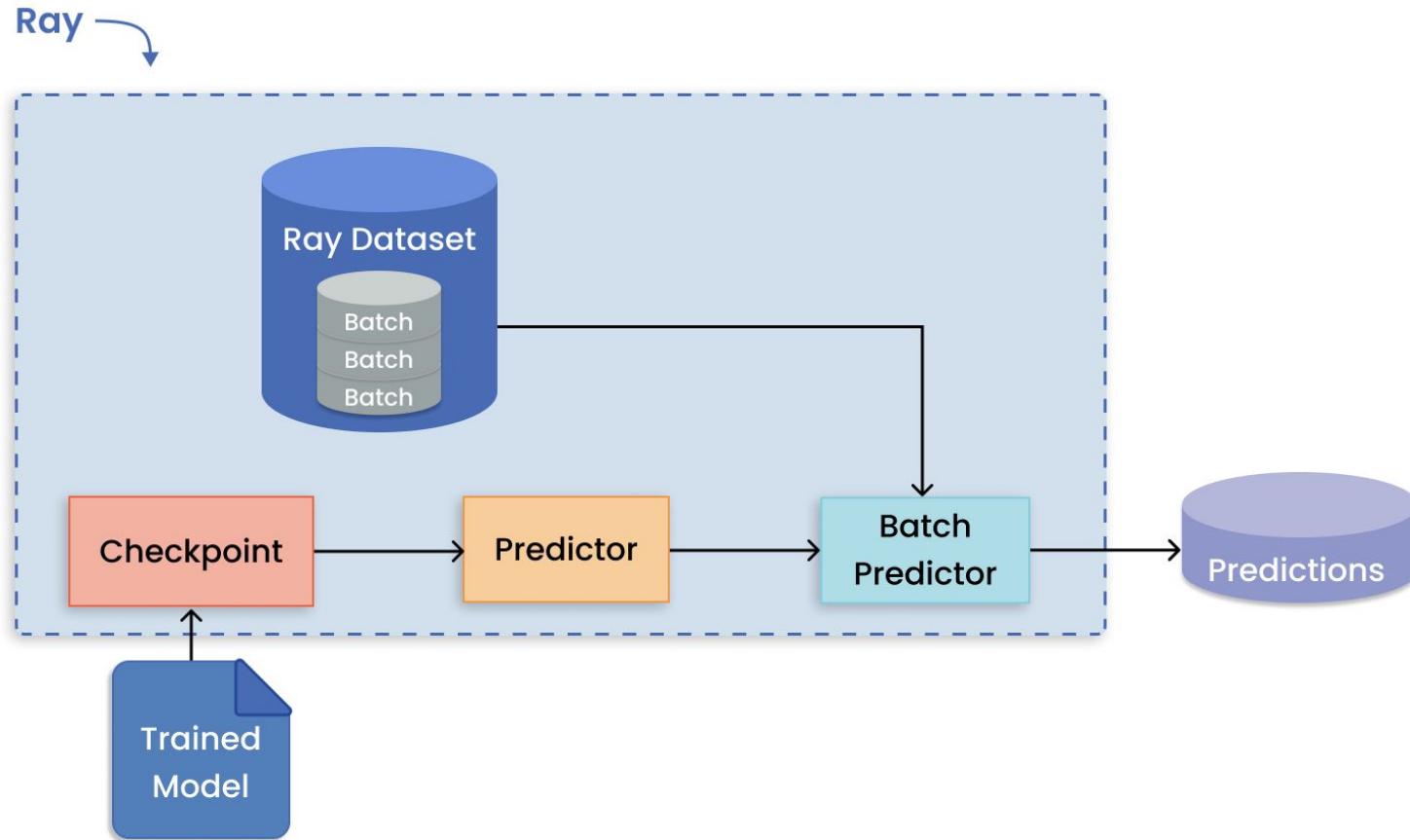
Distributed training of large models



Managing parallel hyperparameter tuning experiments



Batch inference on CPUs and GPUs

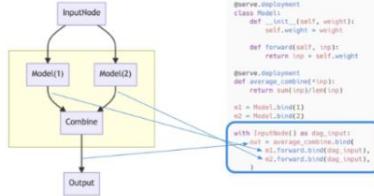


Multi-model composition for model serving

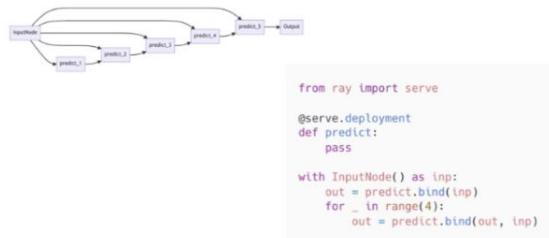
Chaining



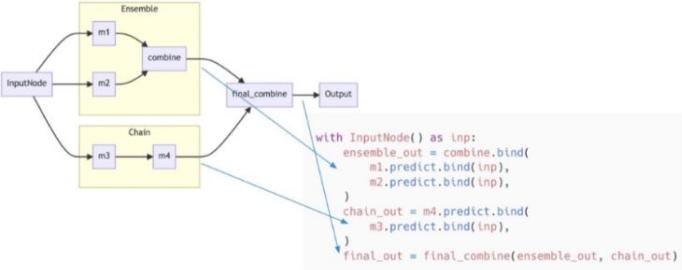
Ensemble



Tree

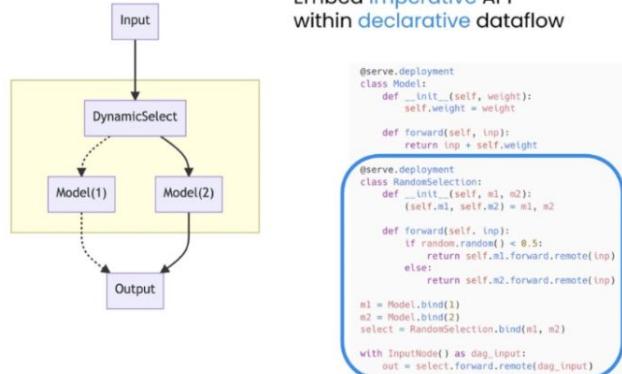


Ensemble + Chaining

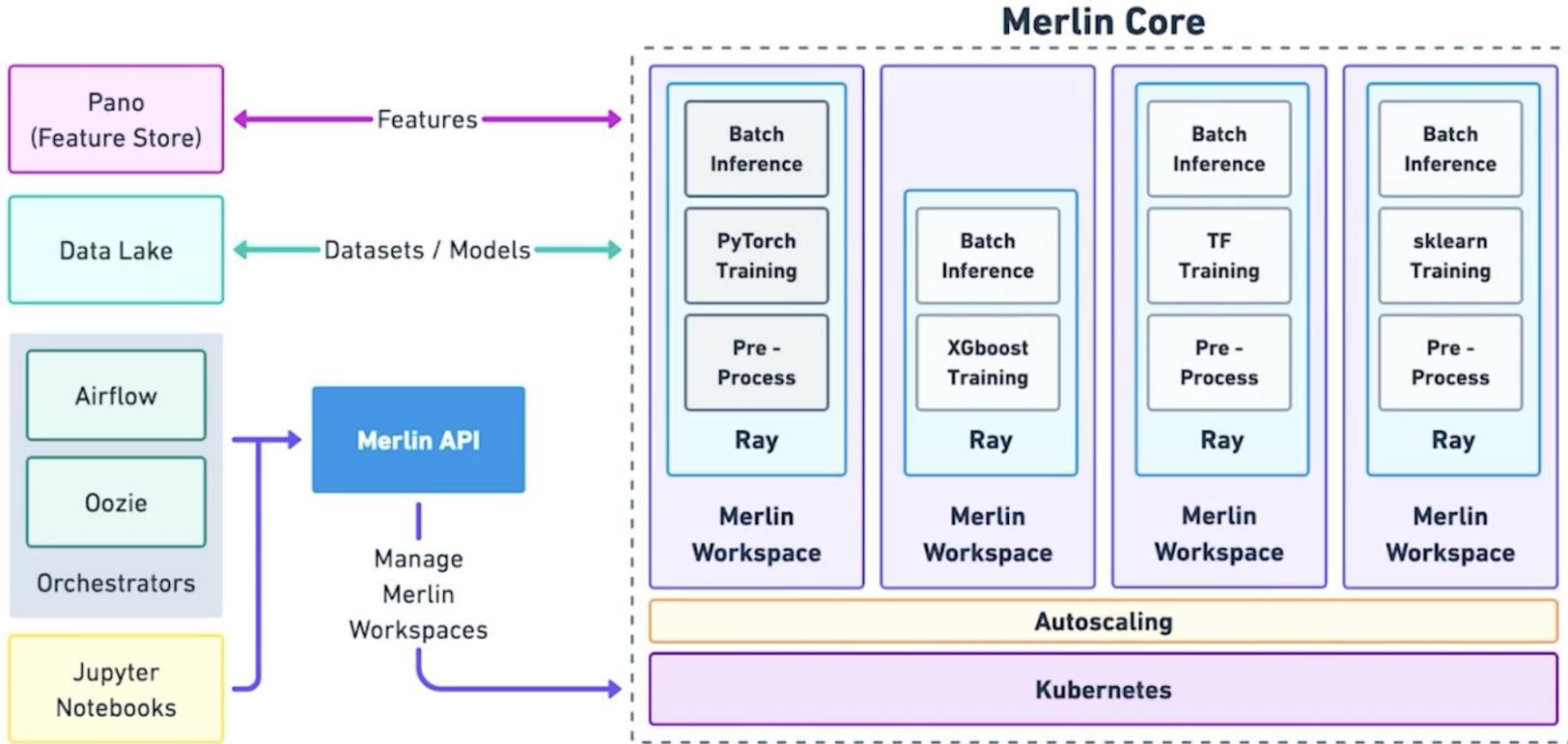


Dynamic Selection

Embed **imperative API** within **declarative dataflow**



ML platform





Which of the following is true about Ray?

- ⓘ Start presenting to display the poll results on this slide.



**What is one primary benefit
of using Ray?**

- ① Start presenting to display the poll results on this slide.

slido



Audience Q&A Session

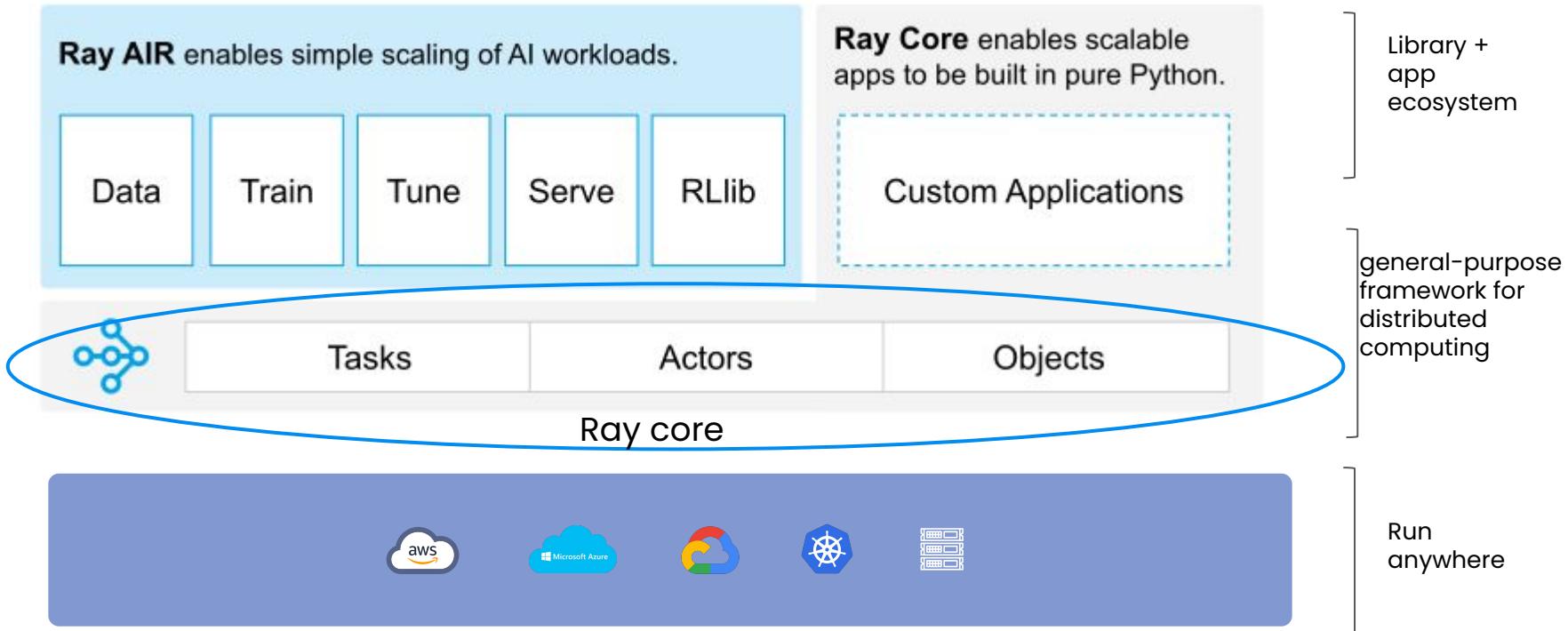
- ⓘ Start presenting to display the audience questions on this slide.

Ray Core

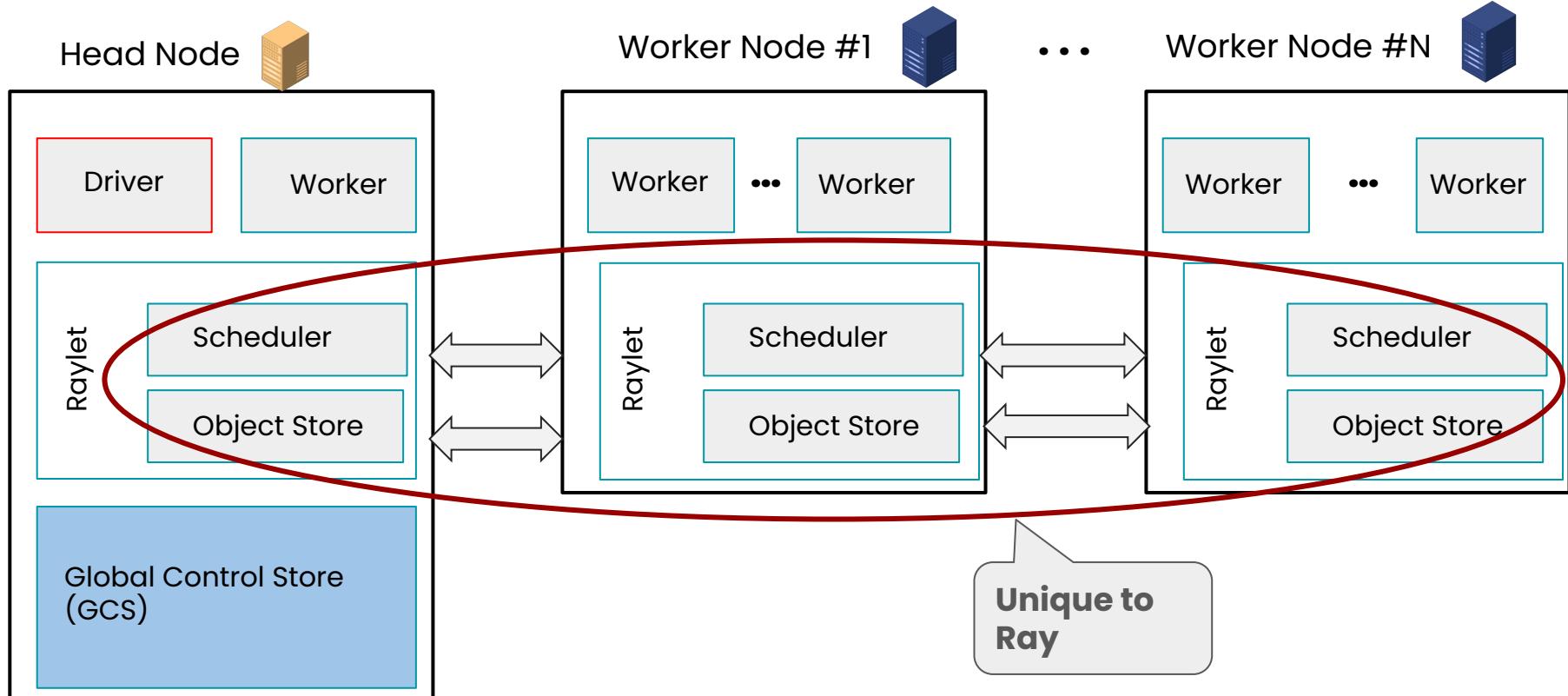
Learn the fundamentals with
remote functions.



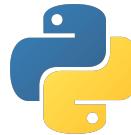
A Layered Cake and Ecosystem



An anatomy of a Ray cluster



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
f.remote()
```

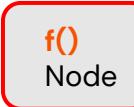


```
@ray.remote
class Cls():
    def __init__(self, x):
        def f(self, a):
            ...
        def g(self, a):
            ...
cls = Cls.remote()
cls.f.remote(a)
```

Distributed



...



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

Actor

```
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
```

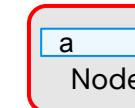
Distributed



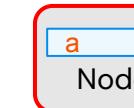
...



Distributed



...



Meet me
here!

JupyterLab session-z3e2qfivi2xjy3aje48buculgx.i.anyscaleuserdata.com/jupyter/lab

File Edit View Run Kernel Tabs Settings Help

Name / Ray_Core / Last Modified

- data an hour ago
- solutions an hour ago
- model_helper_utils.py an hour ago
- Ray_Core_1_Remote_Func... an hour ago
- Ray_Core_2_Remote_O... an hour ago
- Ray_Core_3_Remote_Cl... an hour ago
- Ray_Core_4_Remote_Cl... an hour ago
- tasks_helper_utils.py an hour ago

Ray_Core_1_Remote_Func...

Markdown Python 3 (ipykernel)

A Guided Tour of Ray Core: Remote Tasks

© 2019-2023, Anyscale. All Rights Reserved

Introduction

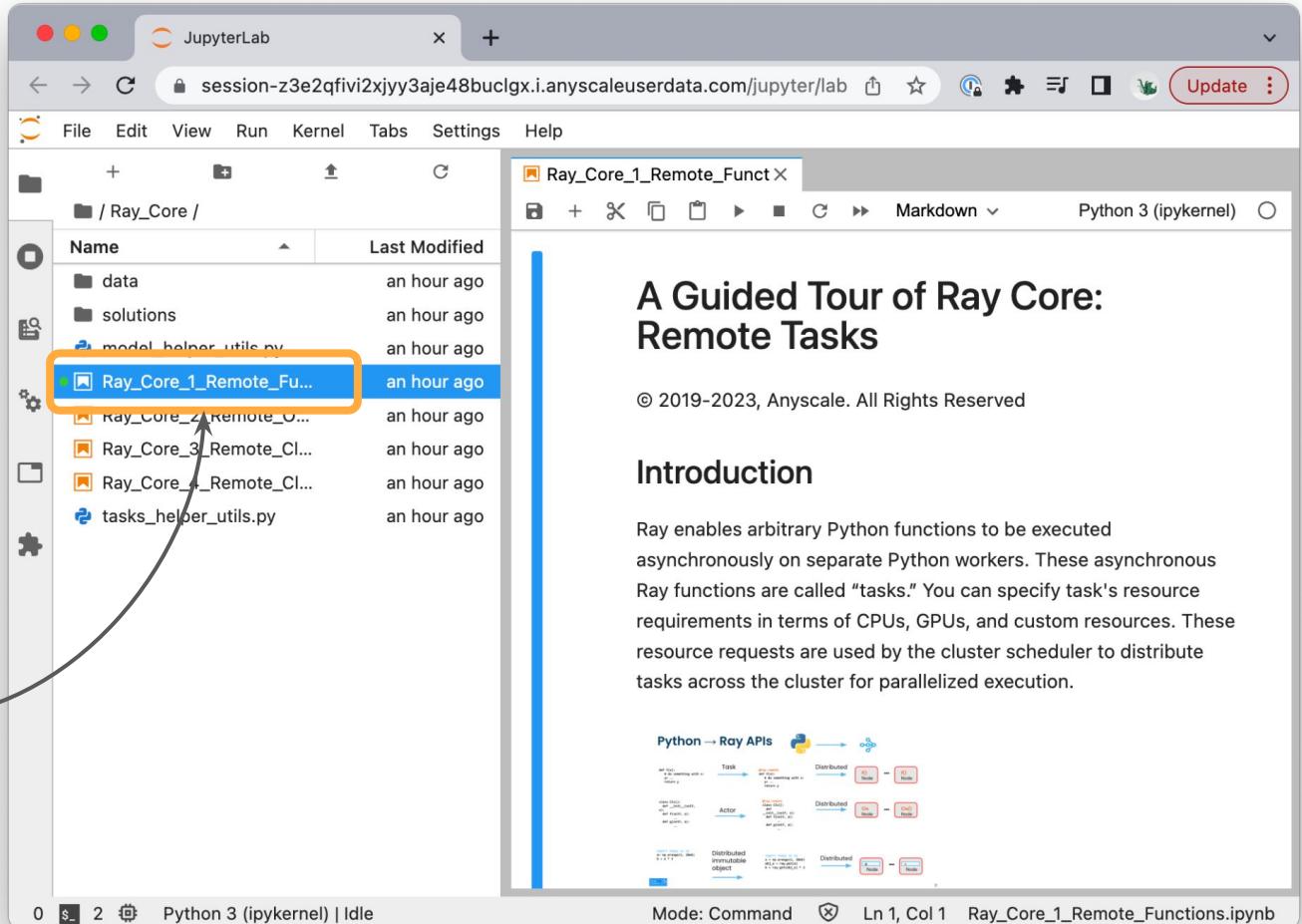
Ray enables arbitrary Python functions to be executed asynchronously on separate Python workers. These asynchronous Ray functions are called “tasks.” You can specify task’s resource requirements in terms of CPUs, GPUs, and custom resources. These resource requests are used by the cluster scheduler to distribute tasks across the cluster for parallelized execution.

Python → Ray APIs → Task → Distributed → Node → Worker

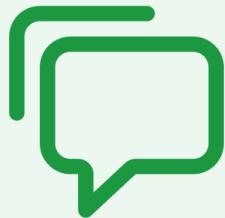
Actor → Actor → Distributed → Node → Worker

Distributed immutable object → Distributed → Node → Worker

Mode: Command Ln 1, Col 1 Ray_Core_1_Remote_Functions.ipynb



slido



Audience Q&A Session

- ⓘ Start presenting to display the audience questions on this slide.

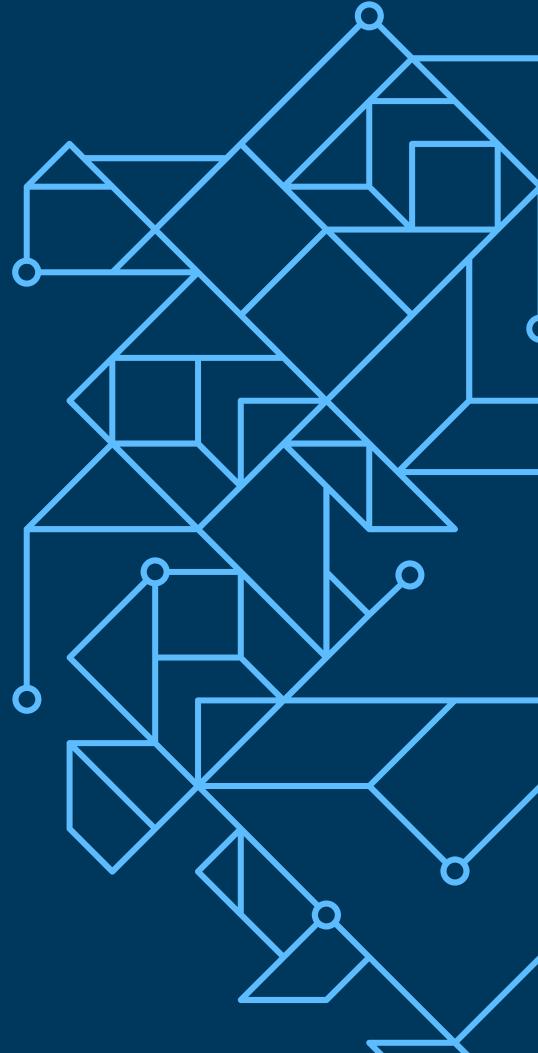


**Time for a
Break!**

30 minutes.

Ray Core

Remote objects and classes.



Meet me
here!

The screenshot shows a JupyterLab interface. On the left, a file tree displays a directory named 'Ray_Core' containing several files: 'data', 'solutions', 'model_helper_utils.py', 'Ray_Core_1_Remote_Obj...', 'Ray_Core_2_Remote_Obj...', 'Ray_Core_3_Remote_Cl...', 'Ray_Core_4_Remote_Cl...', and 'tasks_helper_utils.py'. The file 'Ray_Core_2_Remote_Obj...' is highlighted with an orange border and has a blue arrow pointing from it towards the right panel. The right panel contains a notebook cell titled 'Ray_Core_2_Remote_Objects.ipynb' in Python 3 (ipykernel) mode. The cell content is a guided tour of Ray Core Remote Objects, starting with an overview and explaining how tasks and actors create and compute on objects, which are stored in a distributed shared-memory object store across the cluster.

A Guided Tour of Ray Core:
Remote Objects

© 2019-2023, Anyscale. All Rights Reserved

Overview

In Ray, tasks and actors create and compute on objects. We refer to these objects as remote objects because they can be stored anywhere in a Ray cluster, and we use object refs to refer to them. Remote objects are cached in Ray's distributed shared-memory object store, and there is one object store per node in the cluster. In the cluster setting, a remote object can live on one or many nodes, independent of who holds the object ref(s). Collectively, these individual object store makes a shared object store across the the Ray Cluster, as shown in the diagram below.

Remote Objects reside in a distributed [shared-memory object store](#).

Mode: Command 0 3 Python 3 (ipykernel) | Idle Ray_Core_2_Remote_Objects.ipynb

Meet me
here!

The screenshot shows a JupyterLab interface. On the left, a file tree displays a directory named 'Ray_Core' containing several files: 'data', 'solutions', 'model_helper_utils.py', 'Ray_Core_1_Remote_Cl...', 'Ray_Core_2_Remote_O...', 'Ray_Core_3_Remote_Cl...', 'Ray_Core_4_Remote_Cl...', and 'tasks_helper_utils.py'. The file 'Ray_Core_3_Remote_Cl...' is highlighted with a blue selection bar and has a yellow arrow pointing to it from the text 'Meet me here!'. The right panel shows a notebook titled 'Ray_Core_3_Remote_Class.ipynb' in Python 3 (ipykernel) mode. The content of the notebook is:

A Guided Tour of Ray Core: Remote Stateful Classes

© 2019-2023, Anyscale. All Rights Reserved

Overview

Actors extend the [Ray API](#) from functions (tasks) to classes. An actor is essentially a stateful worker (or a service). When a new actor is instantiated, a new worker is created or an existing worker is used. The methods of the actor are scheduled on that specific worker and can access and mutate the state of that worker. Like tasks, actors support CPU, GPU, and custom resource requirements.

Learning objectives

In this tutorial, we'll discuss Ray Actors and learn about:

Mode: Command No Kernel | Idle Ln 1, Col 1 Ray_Core_3_Remote_Cl..._part_1.ipynb



Which of these are Ray Core primitives? Select all that apply.

- ⓘ Start presenting to display the poll results on this slide.



Which of the following is true about Ray Core?

- ⓘ Start presenting to display the poll results on this slide.



**Check-In: How well are you
understanding the material
so far?**

- ⓘ Start presenting to display the poll results on this slide.

slido



Audience Q&A Session

- ⓘ Start presenting to display the audience questions on this slide.

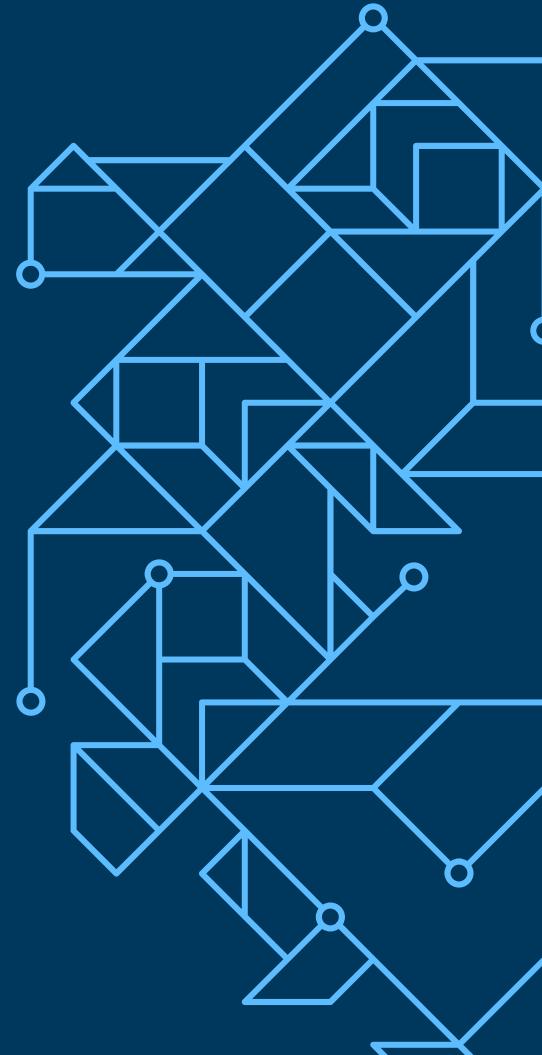


**Time for a
Break!**

30 minutes.

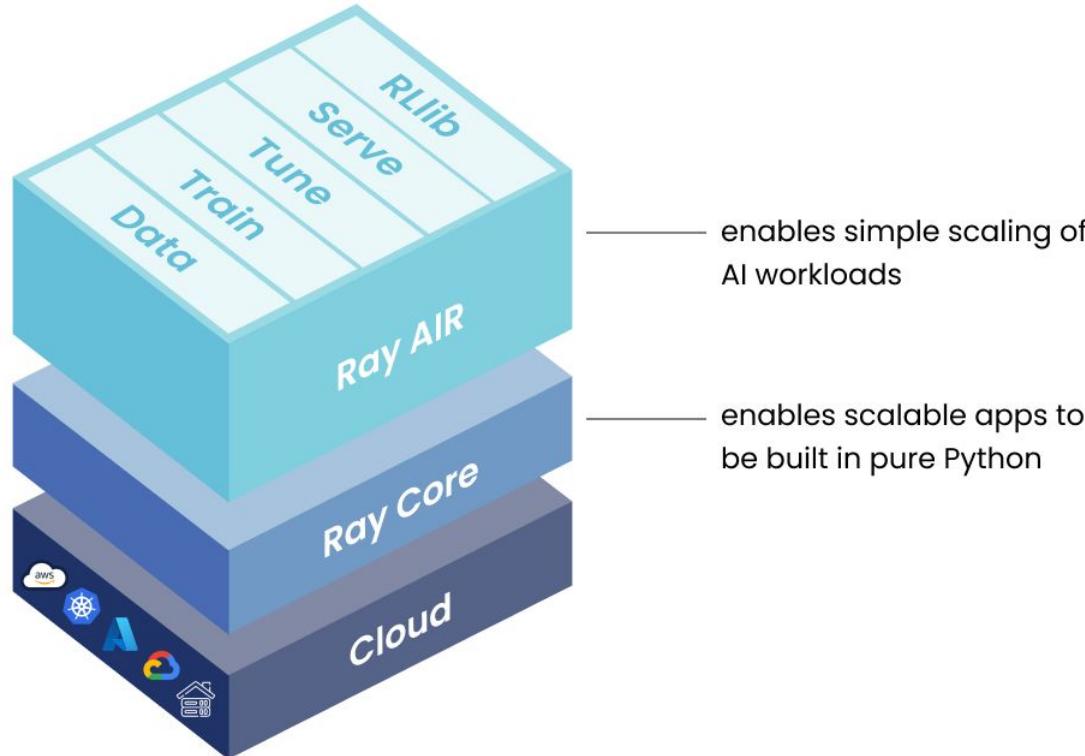
Introduction to Ray AIR

Scalable, end-to-end machine
learning.

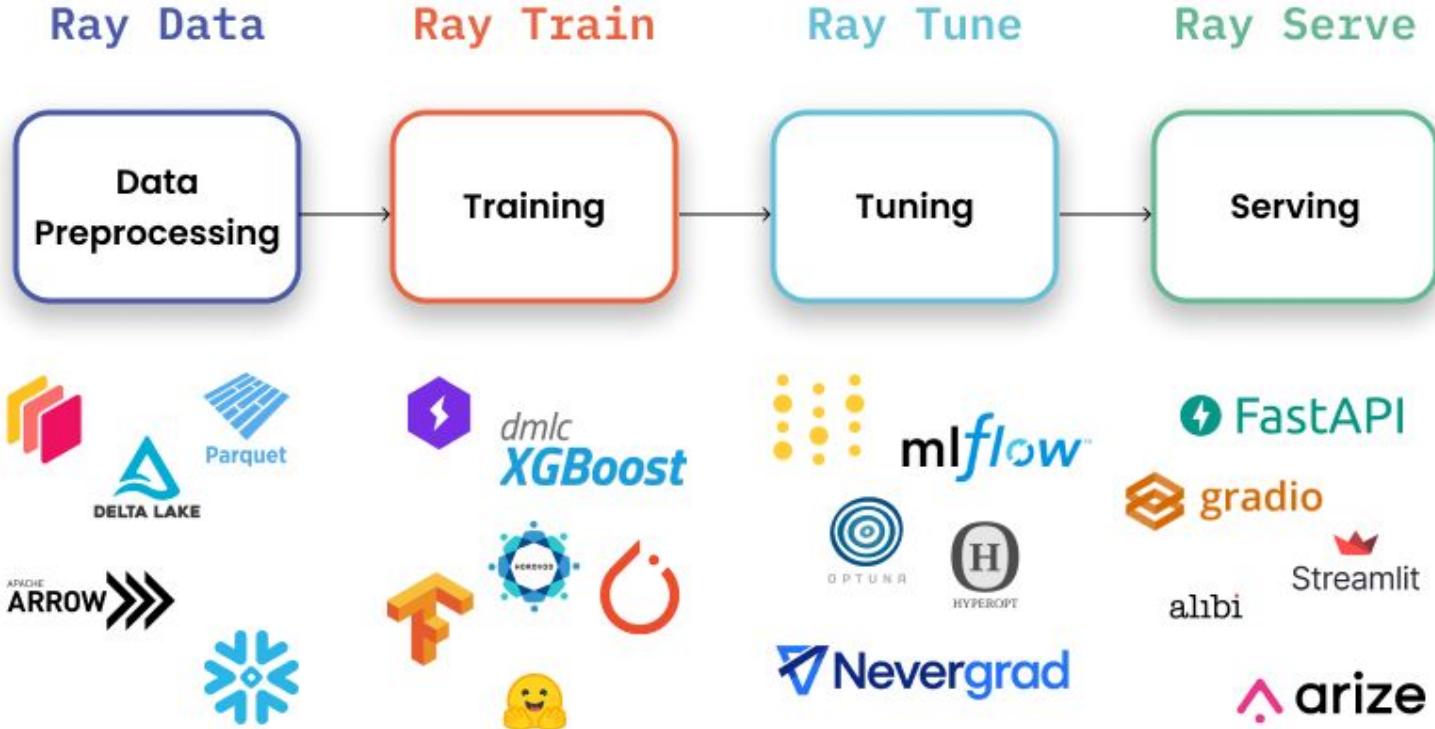




Where Ray AIR sits.



↔ End-to-end ML scaling.



Meet me
here!

The screenshot shows a JupyterLab interface. On the left, a file tree displays a folder named 'Introductory_modules' containing several files: 'Introduction_to_Ray_AIR.ipynb' (selected and highlighted with an orange box), 'Overview_of_Ray.ipynb', 'Overview-of-Ray.pdf', 'quickstart_with_Ray_AIR...', and 'quickstart_with_Ray_Cor...'. The main panel shows the content of the selected notebook, titled 'Introduction to Ray AI Runtime (AIR)'. It features the RAY logo and sections for 'About this notebook' and 'Is this module right for you?'. The text in the 'Is this module right for you?' section describes the module's purpose: learning how to use Ray AIR for end-to-end machine learning applications. The bottom status bar indicates 'Mode: Command' and the current file path 'Ln 1, Col 1 Introduction_to_Ray_AIR.ipynb'.

File Edit View Run Kernel Tabs Settings Help

Name Last Modified

- Introduction_to_Ray_AIR.ipynb
- Overview_of_Ray.ipynb
- Overview-of-Ray.pdf
- quickstart_with_Ray_AIR...
- quickstart_with_Ray_Cor...

Introduction to Ray AI Runtime (AIR)

 RAY

About this notebook

Is this module right for you?

In this module, you will learn how to use Ray AIR to build an end-to-end machine learning application. It covers the entire process from data loading to training and hyperparameter tuning to prediction and serving. Along the way, each section will introduce you to key components of Ray AIR and provide hands-on coding exercises to demonstrate usage.

The ideal learner will have the following attributes:

- A basic understanding of the Ray project.

Mode: Command  Ln 1, Col 1 Introduction_to_Ray_AIR.ipynb



Which of the following is
NOT a Ray library?

- ⓘ Start presenting to display the poll results on this slide.



Which Ray AIR component stores the state of a model, making it accessible during training, tuning, inference, and serving?

- ① Start presenting to display the poll results on this slide.



The `Trainer` component provides a way to scale training with popular ML frameworks. Which other Ray AIR component can use `Trainers` to distribute its workload?

- ⓘ Start presenting to display the poll results on this slide.

slido

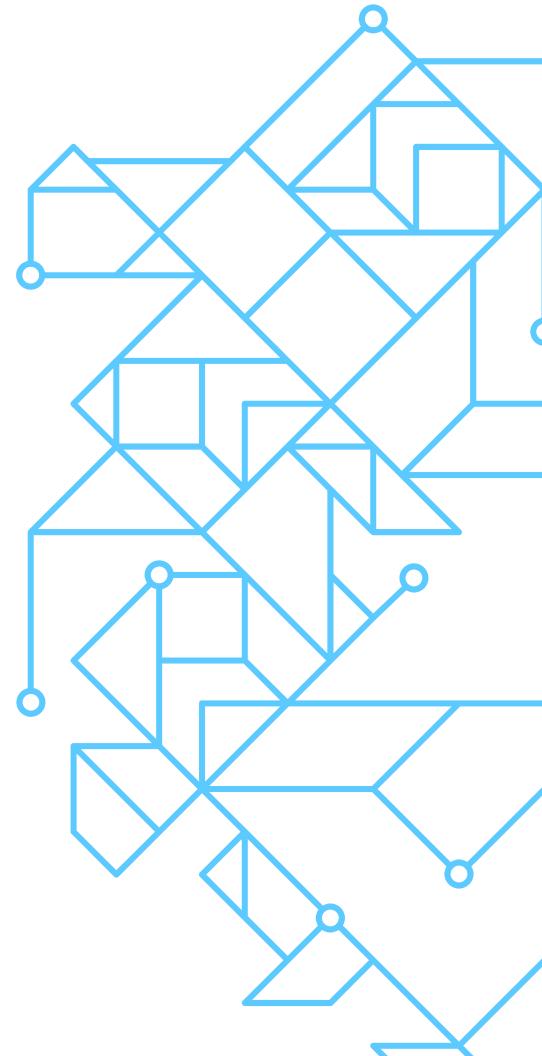


Audience Q&A Session

- ⓘ Start presenting to display the audience questions on this slide.

Wrap-up Day 1

You did it!





Today we learned...



Overview of Ray.

A unified compute layer for distributed Python and ML.



Ray Core.

Deep dive into tasks, actors, and objects.



Introduction to Ray AIR.

Scaling end-to-end machine learning pipelines.



Preview of tomorrow's agenda.

Time	Description
10:00am	Welcome back
10:20am	Distributed model training
12:00pm	Lunch
1:00pm	Observability on Ray

Time	Description
2:00pm	Break
2:30pm	Scaling batch inference
4:00pm	Break
4:30pm	Wrap up and Survey
4:40pm	What's next?



Fill out the survey.

🔗 Go to bit.ly/acm-ray-feedback-1

Remember that we will draw winners for free Ray Summit tickets from completed submissions of both today and tomorrow's surveys.



Let's check the leaderboard.

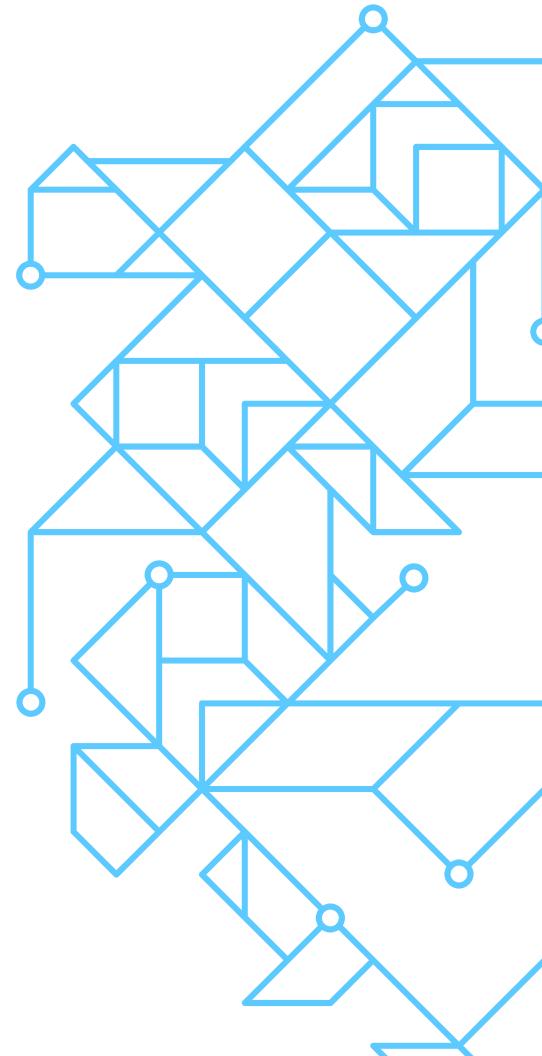


Today's copy of "Learning Ray" goes to...

You'll get another opportunity to compete in tomorrow's quiz challenge.

Welcome to Day 2

We're glad to see you again!





Important links.



[GitHub repository](#)

Access all the notebooks, slides, and scripts.



[Ray documentation](#)

API references and user guides.



[Slido.com](#)

Participate in polls, quizzes, and Q&A.

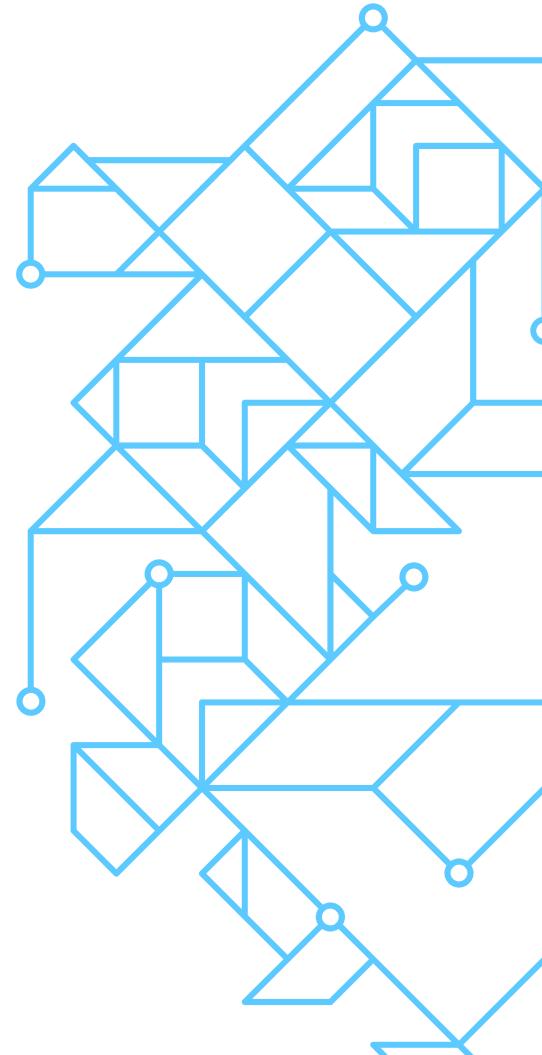


[Survey](#)

One feedback survey per day.

Course Logistics

How to access your cluster and other resources.





Tech check.



Participating via [slido.com](https://www.slido.com)

- Join with code #ray
- Answer polls and quizzes.
 - Enter your name to compete for prizes.
- Ask questions.
 - Pose your own and upvote others.
 - TAs will be answering questions on a rolling basis.



Tech check.

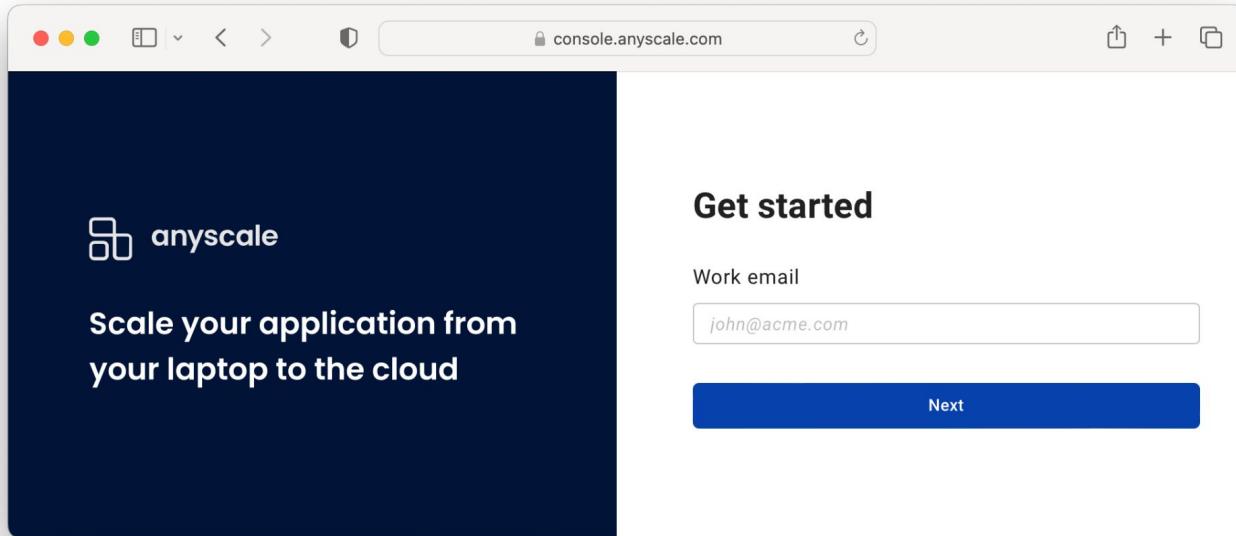


Accessing Anyscale clusters.

- All work will be in Anyscale provisioned clusters.
- Our GitHub repo will be mounted automatically.
- Access begins now.
 - Check your email for login information.
 - Step-by-step instructions to follow.



Link to Anyscale cluster: console.anyscale.com



Check your **email** for your unique username and password!

select
“Clusters”

The screenshot shows the Anyscale console interface. On the left, a dark sidebar menu lists "Home", "Projects", "Workspaces", "Interactive sessions", "Jobs", "Services", "Clusters" (which is highlighted with an orange box and has a curved arrow pointing to it from the left), and "Configurations". Below these are user profile information ("Emmy Li"), "Help", "Feedback", and a "Collapse" button. The main content area is titled "Clusters" and contains a table with the following data:

Name	Status	Active resources	Cost	Cluster environment	Project	Cloud provider
emmy-ray-saturday	Terminated	None	\$0.62	ray-sat-v1:10	ray-saturday	Anyscale

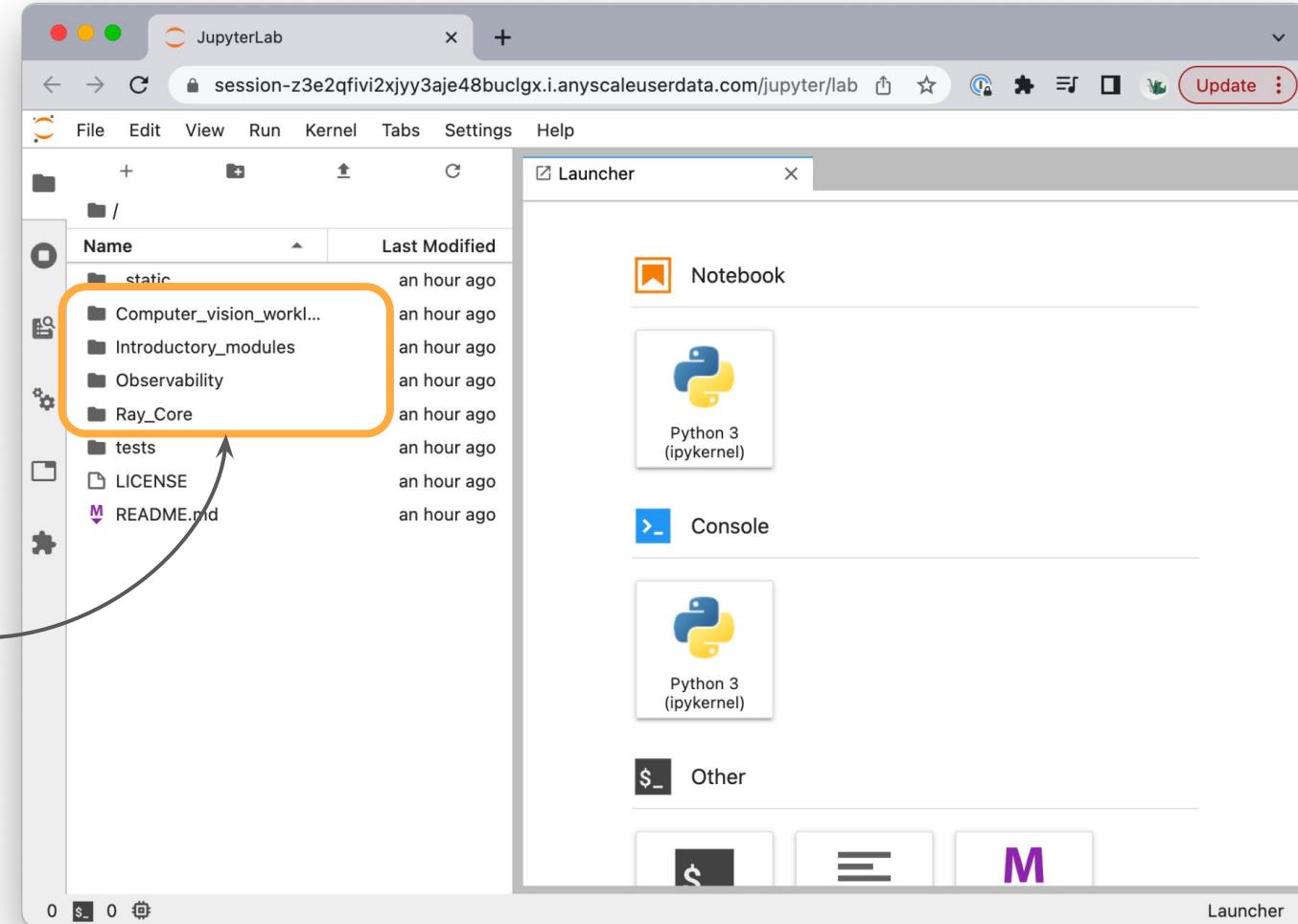
At the bottom right of the table, there are navigation links: "1 - 1 of 1" and arrows for "Previous" and "Next".

click on your cluster

“Start” the cluster,
then
select
“Jupyter”

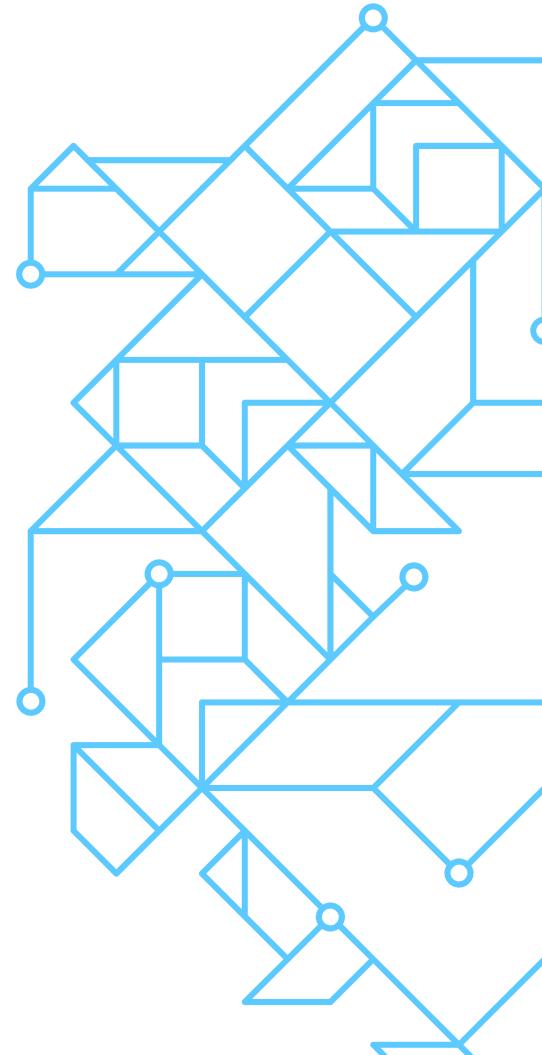
The screenshot shows the anyscale console interface. On the left is a sidebar with navigation links: Home, Projects (highlighted), Workspaces, Interactive sessions, Jobs, Services (highlighted), Clusters, Configurations, Emmy Li, Help, Feedback, and Collapse. The main content area has a breadcrumb path: ray... > emmy-ra... > Jupyter. The Jupyter button is highlighted with an orange box and an arrow pointing to it from the 'Services' link in the sidebar. Below the breadcrumb is a section titled 'About this cluster' with details: Status (Active), ID (ses_z5yfnmzamcpxc4uk95mezhr4), Created by (emmy@anyscale.com), Created at (Dec 8, 2022 at 10:01:54 AM), Access (Everyone in your organization can view a...), and Project (ray-saturday). A section titled 'Resource usage' shows CPU, Object store memory, and GPU all at 0%. It also displays Cost since last start (\$0.02) and Cost since creation (\$2.50). A 'Configuration' section lists Cluster environment (ray-sat-v1:10), Compute config (kk-rs-config-for-emmy), and Cloud (anyscale_default_cloud (aws, us-west-2)). At the bottom is a 'Terminal' section.

View
modules
here



The Plan

Here's what we'll do today.





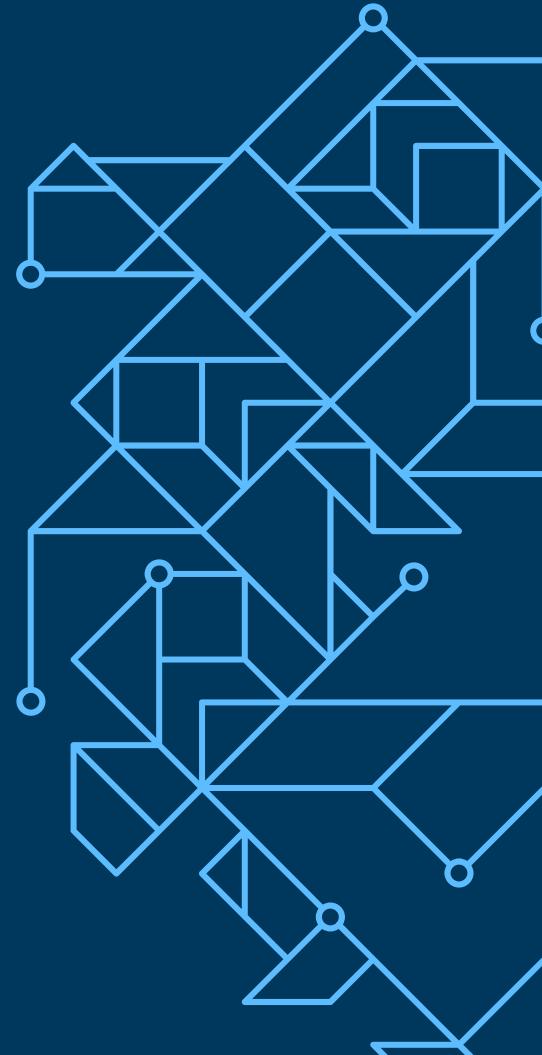
Today's agenda.

Time	Description
10:00am	Welcome back
10:20am	Distributed model training
12:00pm	Lunch
1:00pm	Observability on Ray

Time	Description
2:00pm	Break
2:30pm	Scaling batch inference
4:00pm	Break
4:30pm	Wrap up and Survey
4:40pm	What's next?

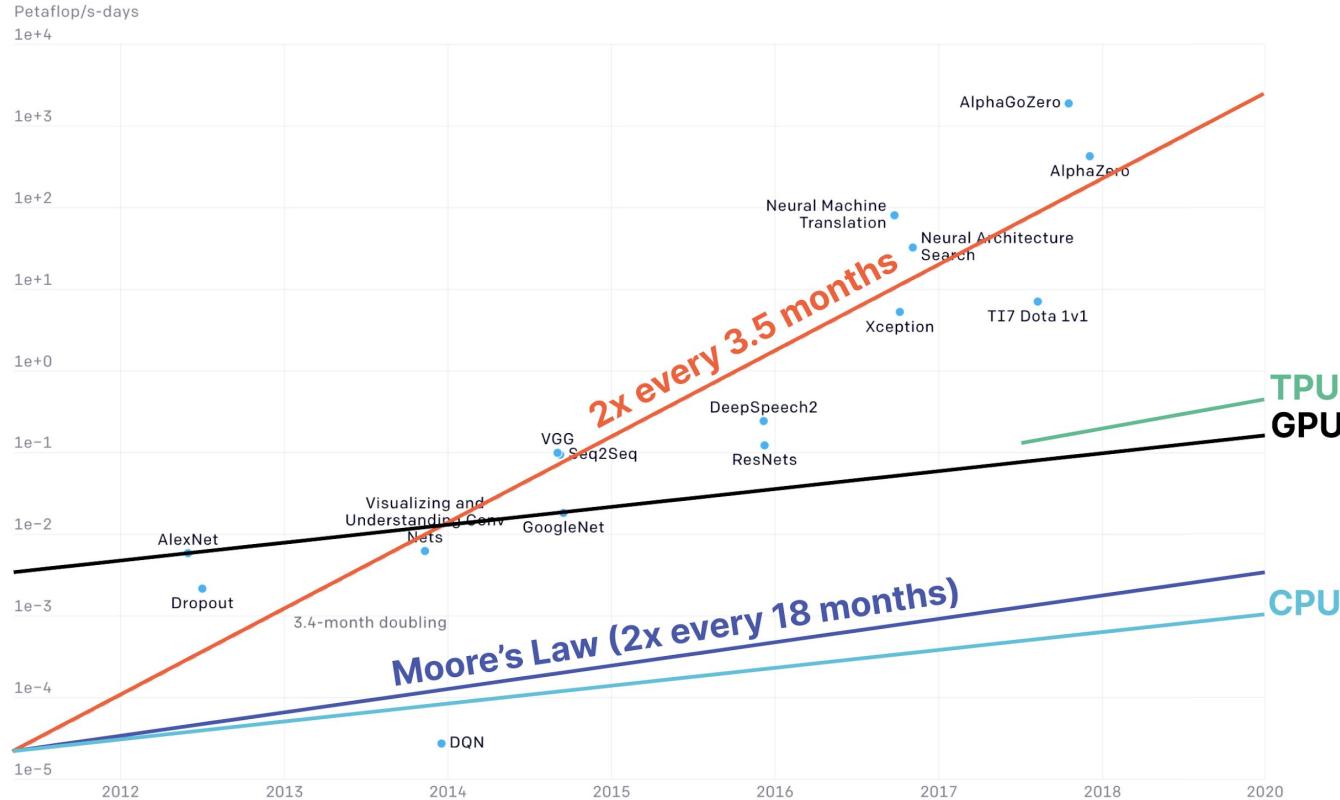
Distributed Model Training

Data parallelism across
multiple machines.



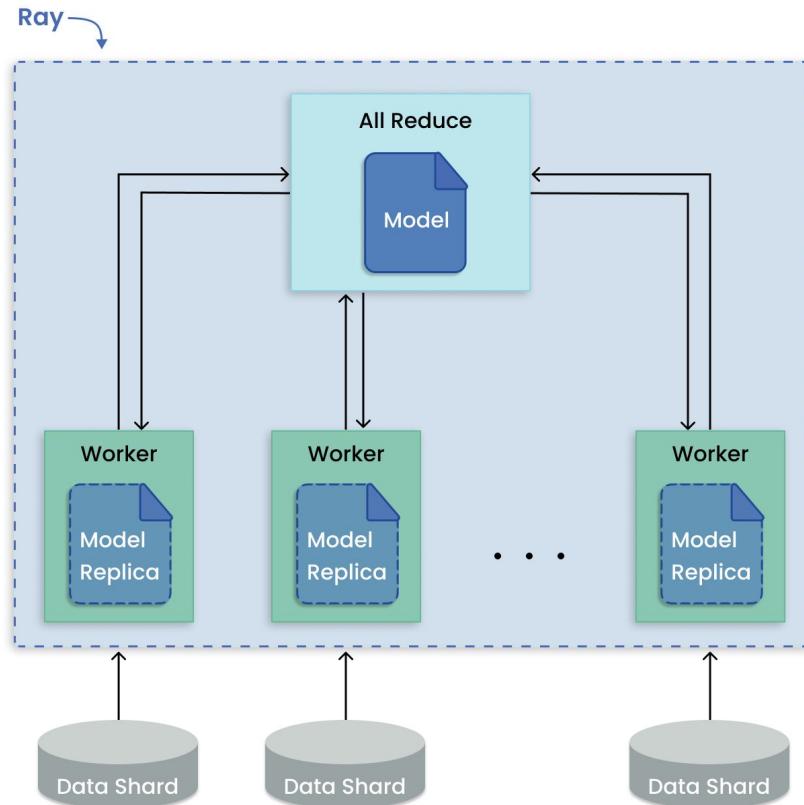


Why distribute training?





Data parallelism





The task.



Semantic segmentation, pixel-by-pixel.



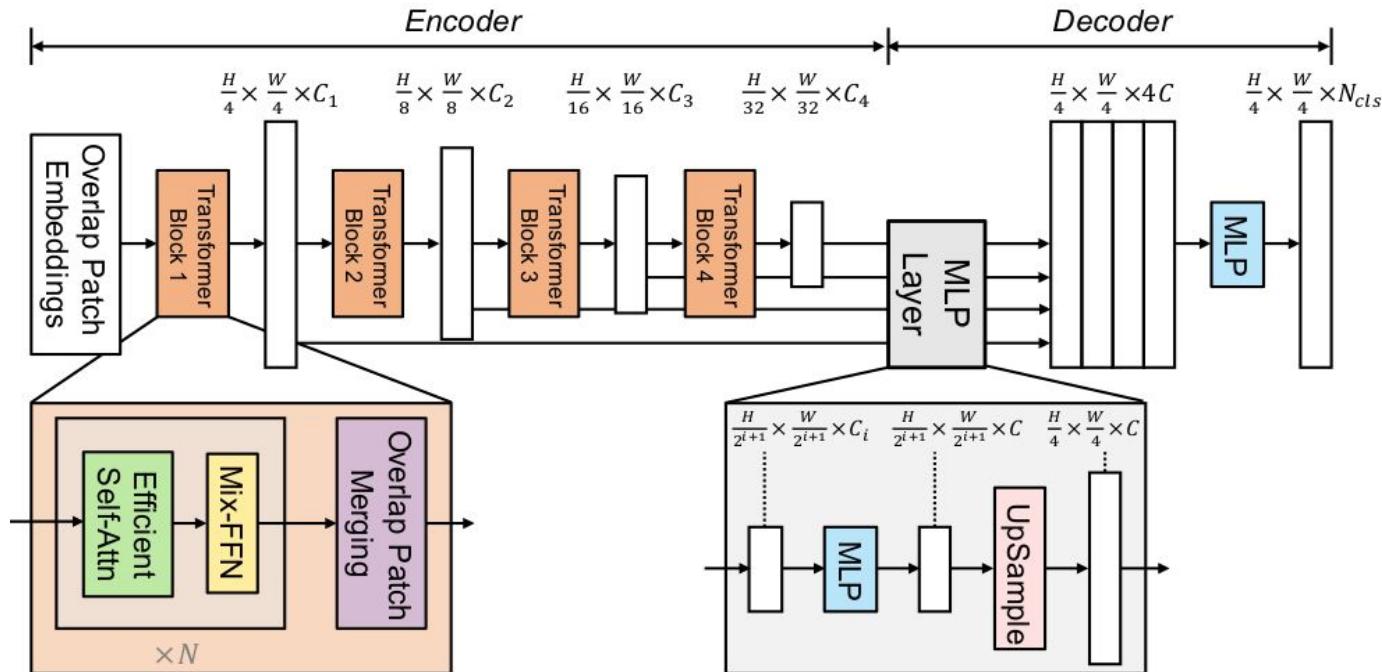
The data.

MIT ADE20K Scene Parsing Benchmark

- 20k annotated, scene-centric training images.
- 3.3k unlabeled test images.
- 150 semantic categories (person, car, bed, sky, etc.)



The model.



Segformer, fine-tuned on MIT ADE20K

Meet me
here!

The screenshot shows a JupyterLab interface. On the left, a file tree displays a directory structure under '/Computer_vision_workloads / Semantic_segmentation'. The 'Scaling_model_training.ipynb' file is highlighted with an orange box and has an arrow pointing from the text 'Meet me here!' towards it. The main panel shows the content of the selected notebook, titled 'Scaling model training' with a RAY logo. The notebook includes sections for 'About this notebook', 'Is this module right for you?', 'Prerequisites', and 'Requirements', along with a bulleted list of prerequisites.

JupyterLab

session-z3e2qfivi2xjyy3aje48buculgx.i.anyscaleuserdata.com/jupyter/lab

File Edit View Run Kernel Tabs Settings Help

/ Computer_vision_workloads / Semantic_segmentation

Name Last Modified

- Batch_inference_with_R... an hour ago
- Scaling_batch_inference an hour ago
- Scaling_model_training.ipynb seconds ago
- utils.py an hour ago

Scaling_model_training.ipynb X

Markdown Python 3 (ipykernel)

Scaling model training

RAY

About this notebook

Is this module right for you?

This module guides you through distributed model training with Ray. Through fine-tuning a transformer for a computer vision task, ML practitioners will learn how to scale training workloads using deep learning models on large datasets.

Prerequisites

For this notebook, you should satisfy the following minimum requirements:

- Practical Python knowledge.
- Familiarity with training deep learning models

Mode: Command Ln 1, Col 1 Scaling_model_training.ipynb



What is data parallelism?

- ⓘ Start presenting to display the poll results on this slide.



Which component was responsible for sharding the data for the distributed data parallelism pattern?

- ① Start presenting to display the poll results on this slide.



**When you create a HuggingFace Trainer,
what does the `CheckpointConfig`
parameter in the `RunConfig` control?**

- ⓘ Start presenting to display the poll results on this slide.

slido



Audience Q&A Session

- ⓘ Start presenting to display the audience questions on this slide.



Time for Lunch!

60 minutes.

Observability

Understanding the internals of
your Ray application.



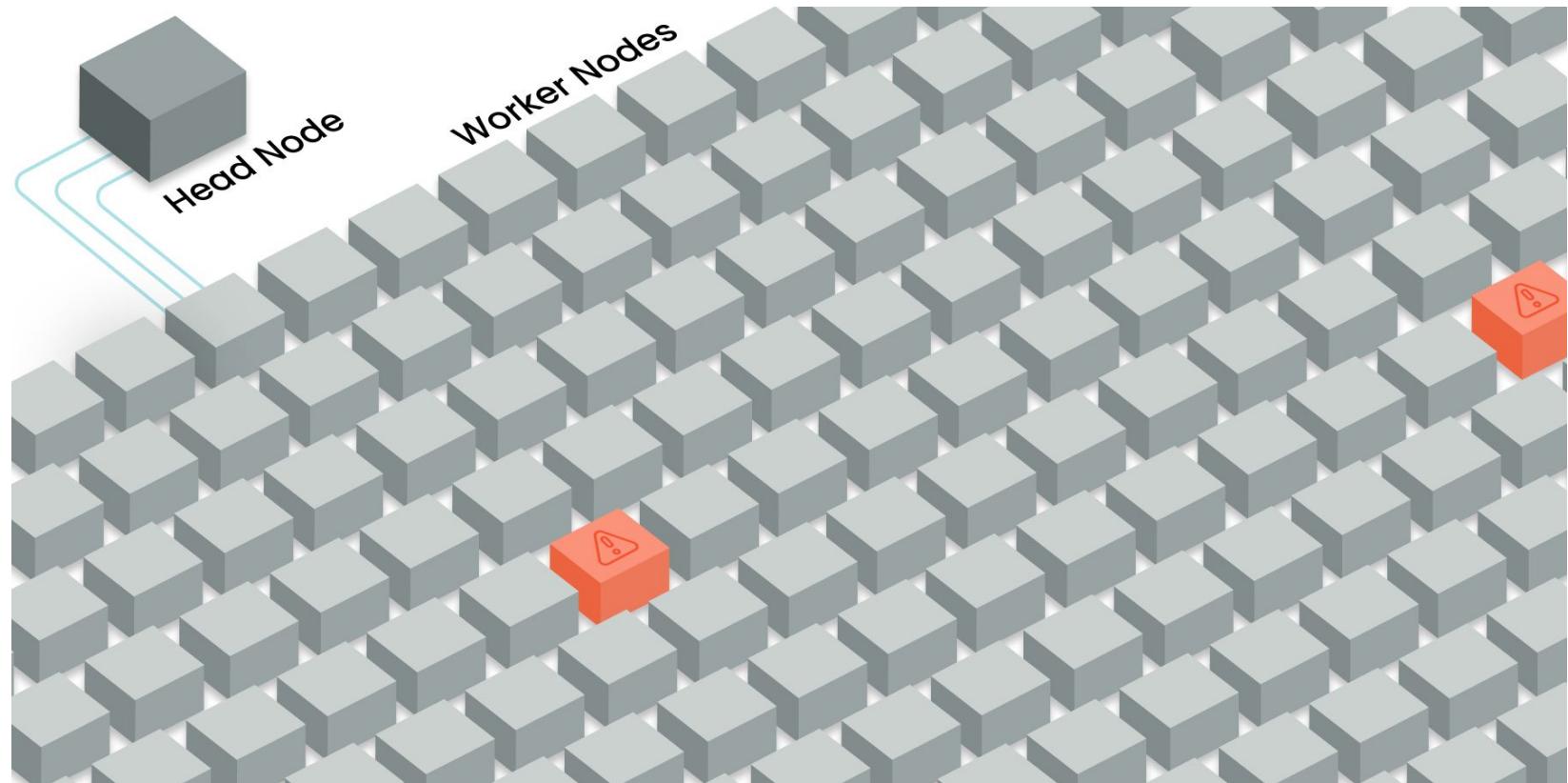


What is observability?

The ability to understand the behavior of the internal state of a system inferred from its external outputs.



Why is it important?



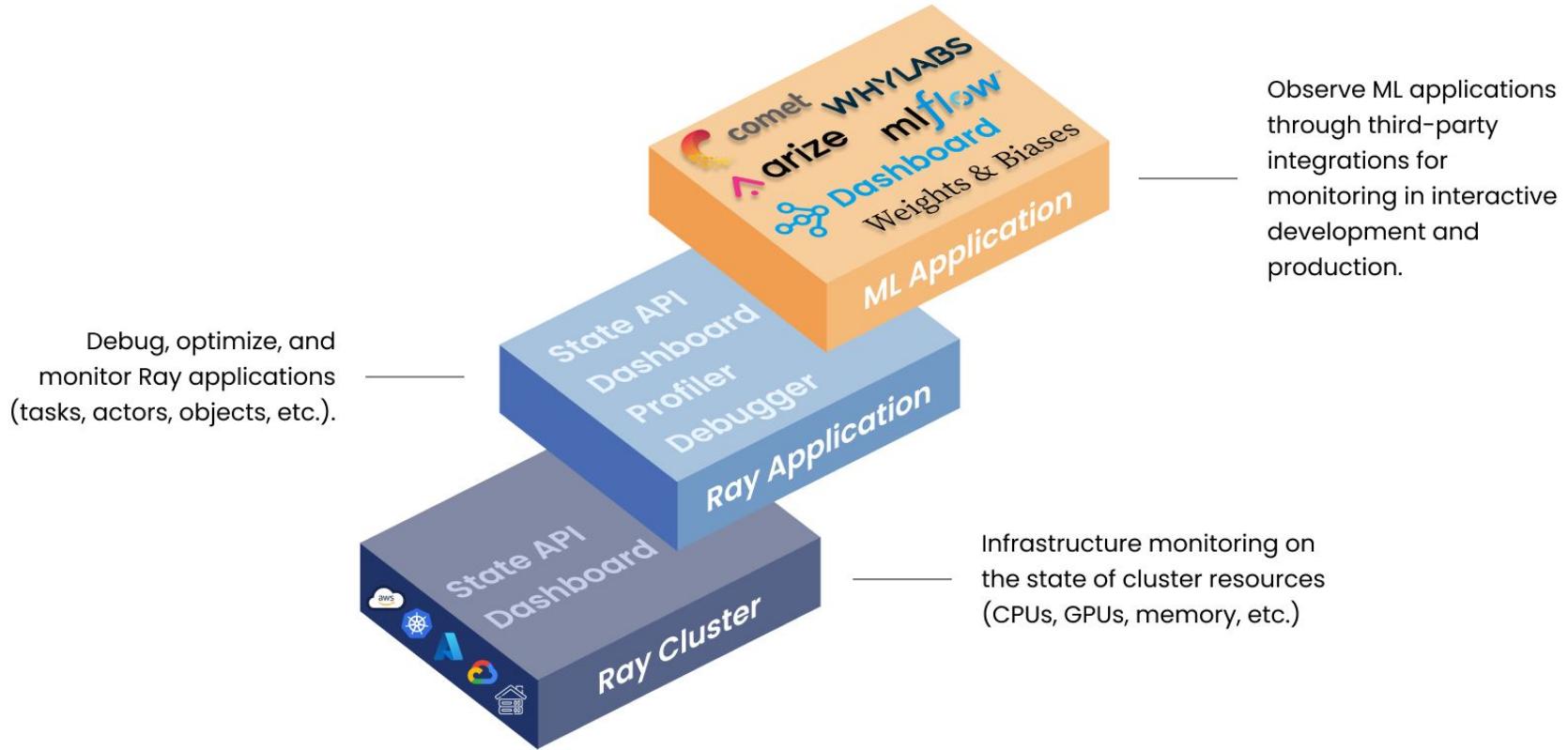


The thought process.

- How do I know if an actor has failed?
- When I become aware actor failures, how do I know which one(s) caused the issue?
- Once I know which actor(s) to inspect, how can I find the log and fix the bug?



Observability tooling by layer.



Meet me
here!

The screenshot shows a JupyterLab interface. On the left, there's a file browser window titled 'Observability /' with a list of files. An arrow points from the text 'Meet me here!' to the file 'Ray_observability_part_1.ipynb', which is highlighted with an orange border. The main content area is titled 'Ray Observability Part 1' and features a RAY logo. It includes sections for 'About this notebook', 'Is this module right for you?', 'Prerequisites', and minimum requirements. The status bar at the bottom shows 'No Kernel | Idle'.

JupyterLab

session-z3e2qfivi2xjyy3aje48buculgx.i.anyscaleuserdata.com/jupyter/lab

File Edit View Run Kernel Tabs Settings Help

/ Observability /

Name Last Modified

Ray_observability_part_1.ipynb an hour ago

Ray Observability Part 1

RAY

About this notebook

Is this module right for you?

This module provides a general purpose introduction to the most common observability tools to effectively debug, optimize, and monitor Ray applications. It is for data scientists, ML practitioners, ML engineers, and Python developers looking for ways to understand the behavior of their Ray systems.

Prerequisites

For this notebook, you should satisfy the following minimum requirements:

- Practical Python experience

Mode: Command ↵ Ln 1, Col 1 Ray_observability_part_1.ipynb



What Ray observability tool acts as a central hub for accessing metrics, logs, cluster resources, and more?

- ① Start presenting to display the poll results on this slide.



Which of the following is NOT a good way to observe an OOM in a Ray application?

- ① Start presenting to display the poll results on this slide.



Which of the following is not accessible through the Ray Dashboard?

- ① Start presenting to display the poll results on this slide.



**Check-in: How well are you
understanding the material
so far?**

- ⓘ Start presenting to display the poll results on this slide.

slido



Audience Q&A Session

- ⓘ Start presenting to display the audience questions on this slide.

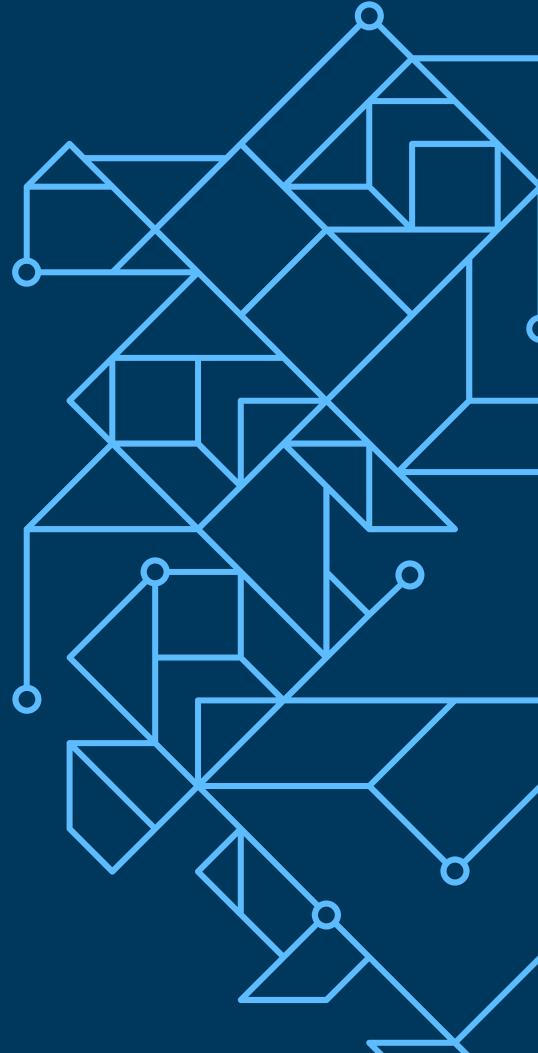


Time for a Break!

30 minutes.

Batch Inference

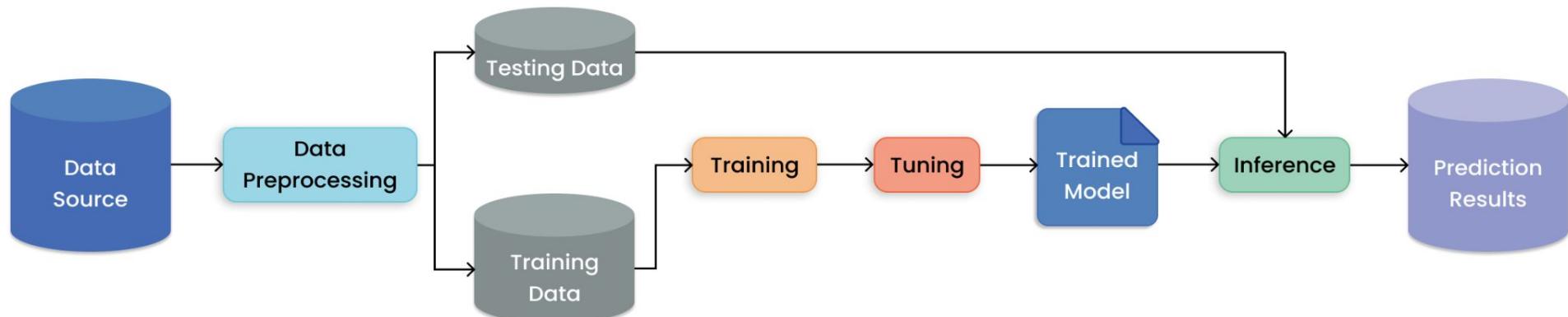
Generating predictions on
batches of data.





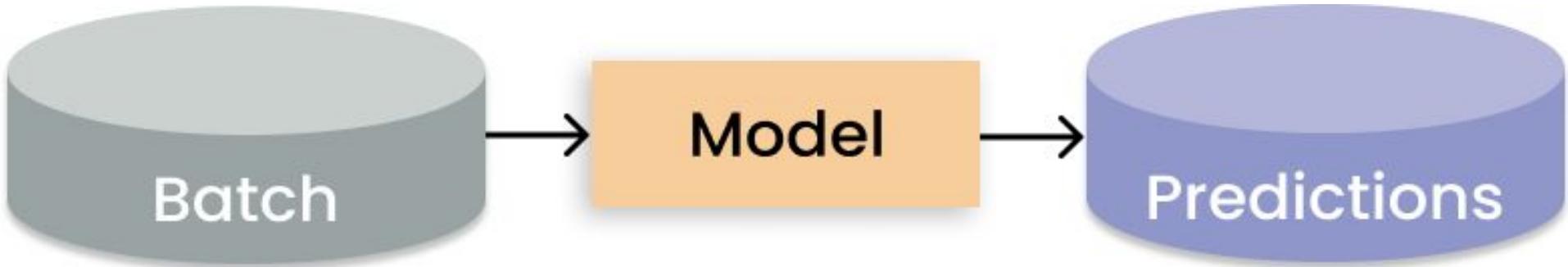
What is batch inference?

The process of generating predictions on a large set, or “batch,” of data.





What is batch inference?





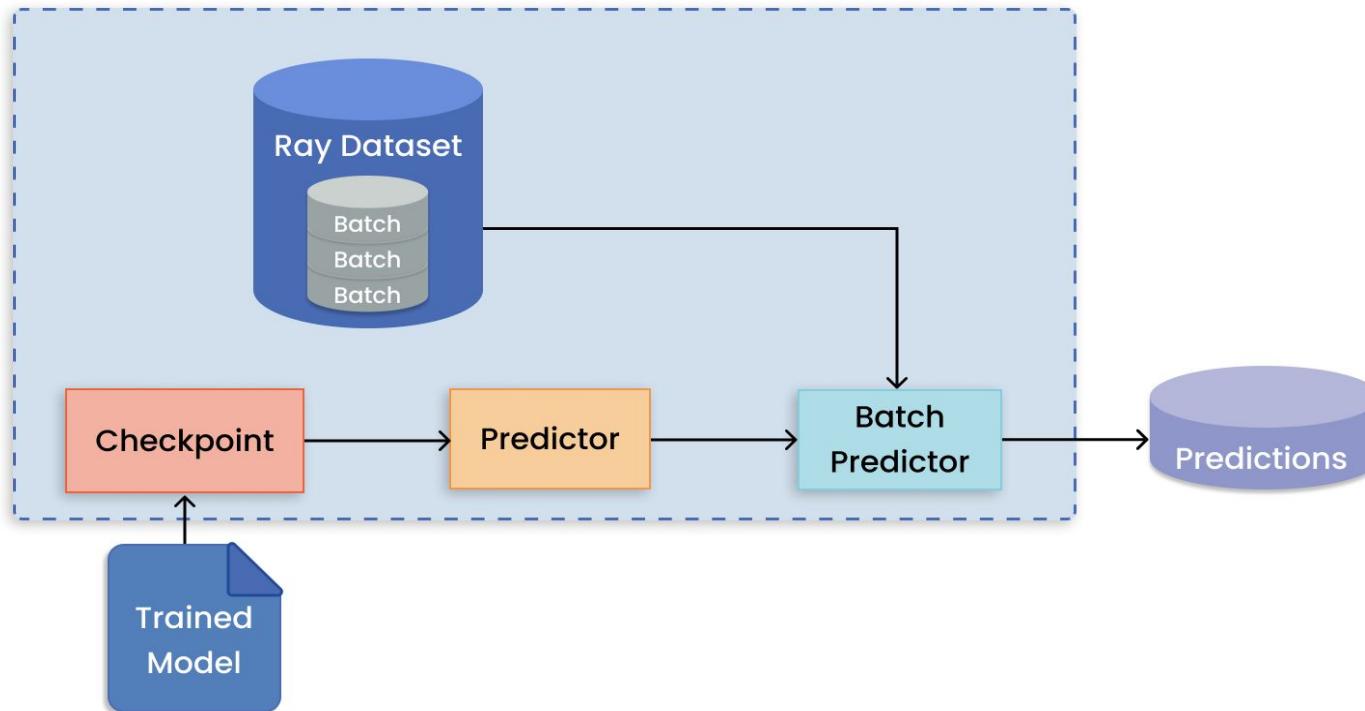
Design patterns for batch inference with Ray

- Ray AIR BatchPredictor
- Ray Core Actors
 - Stateful inference
- Ray Core Tasks
 - Stateless inference



Method: Ray AIR

Batch Ray ↘



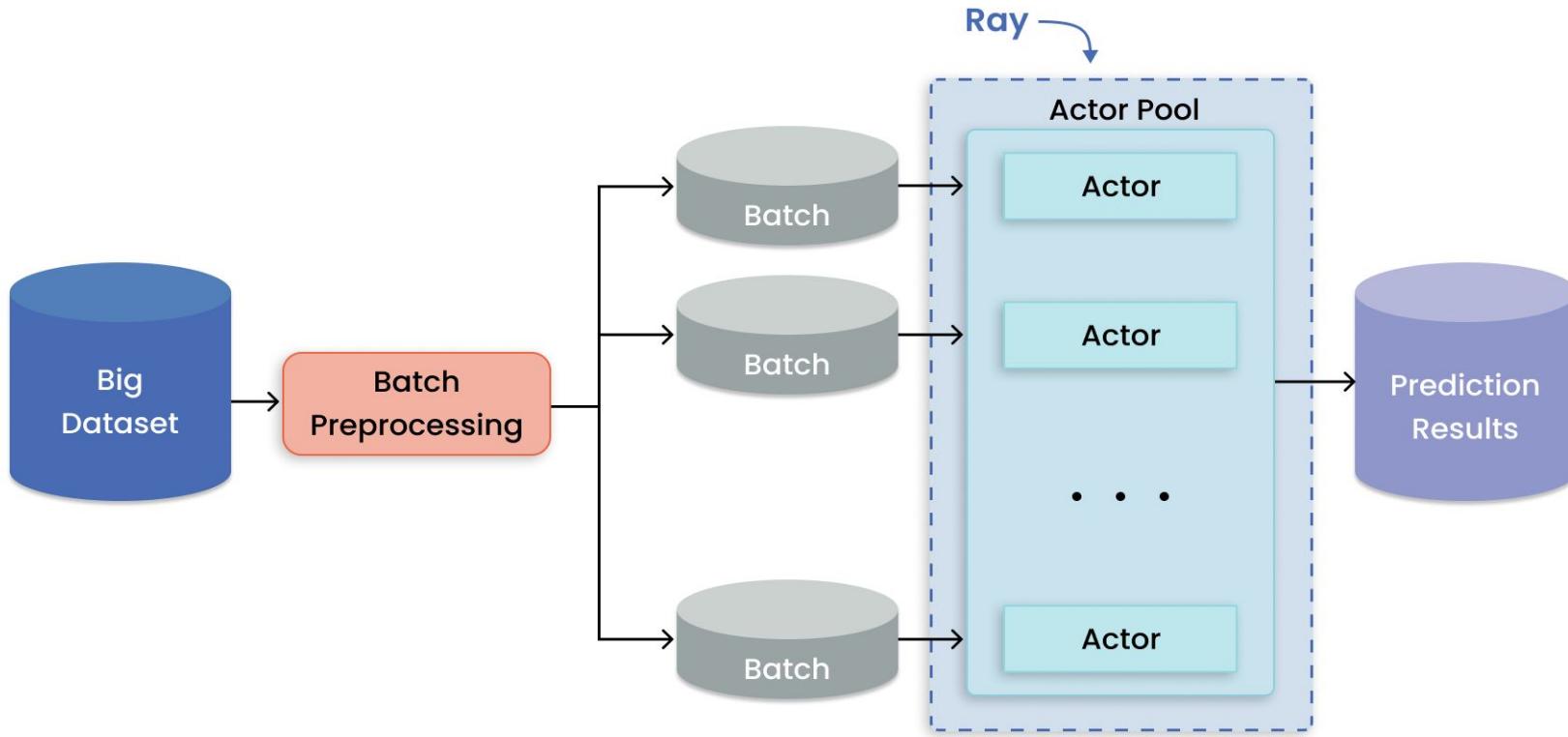


Method: Ray AIR BatchPredictor

```
batch_predictor = BatchPredictor(  
    Checkpoint,  
    Predictor  
)
```



Method: Ray Core Actors



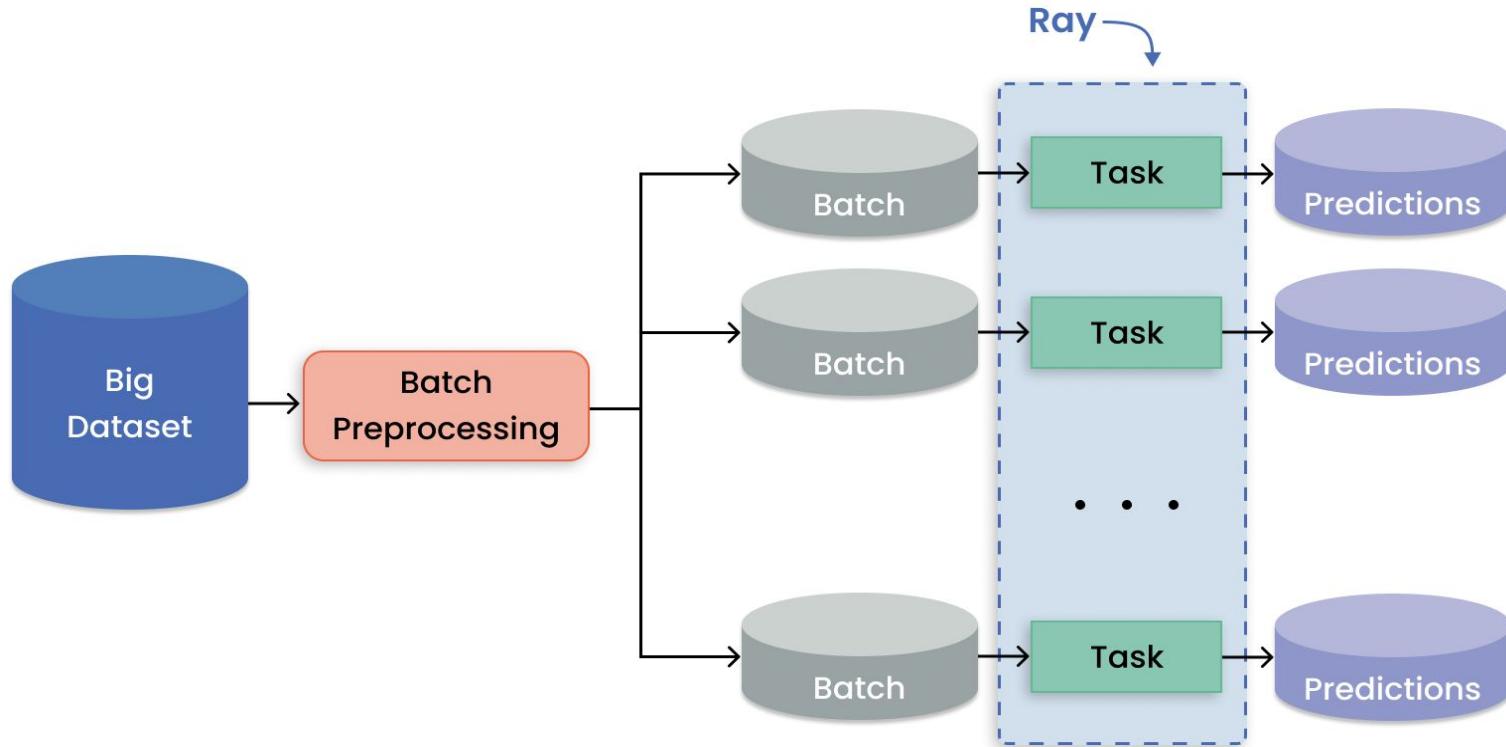


Method: Ray Core Actors

```
actors = [ActorCls.remote(input) for _ in range(10)]
```



Method: Ray Core Tasks





Method: Ray Core Tasks

```
[ object_refs = [task.remote(input) for _ in range(10)] ]
```



The task.



Semantic segmentation, pixel-by-pixel.



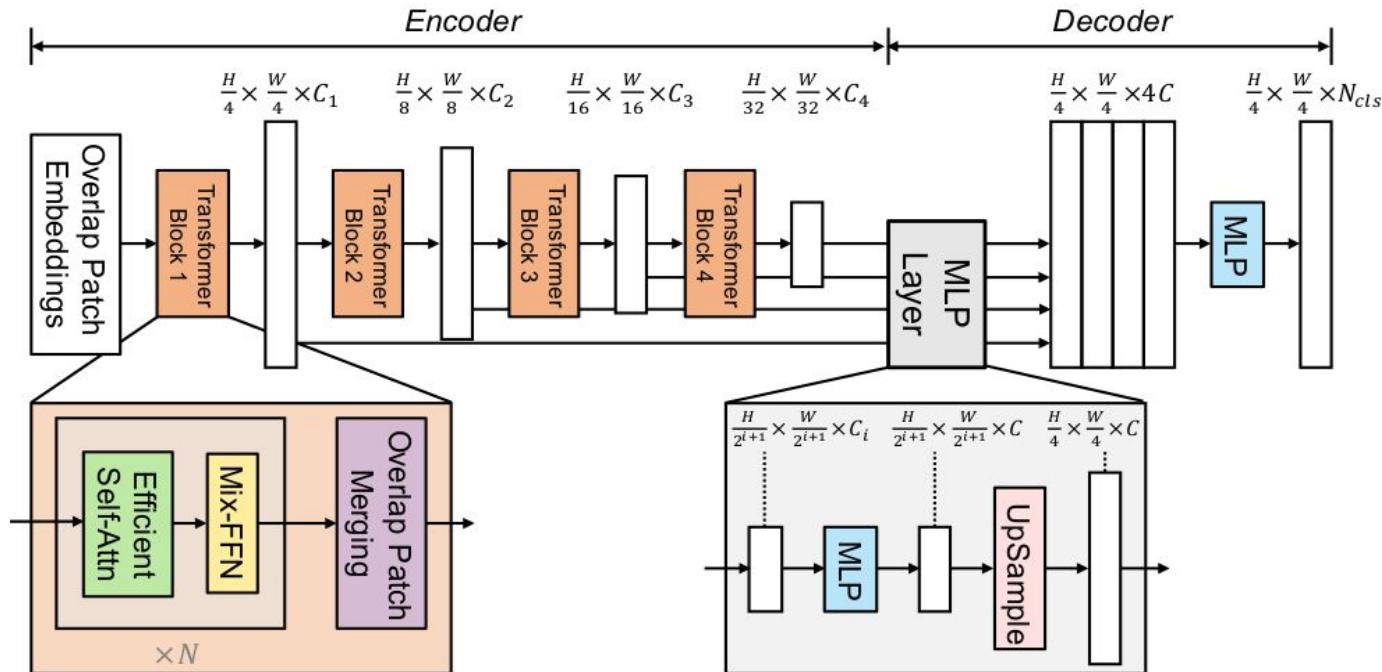
The data.

MIT ADE20K Scene Parsing Benchmark

- 20k annotated, scene-centric training images.
- 3.3k unlabeled test images.
- 150 semantic categories (person, car, bed, sky, etc.)



The model.



Segformer, fine-tuned on MIT ADE20K

Meet me
here!

The screenshot shows a JupyterLab interface. On the left, there's a file tree with a folder named 'Computer_vision_workloads / Semantic_segmentation'. Inside this folder are several files: 'Batch_inference_with_R.R' (modified 'an hour ago'), 'Scaling_batch_inference.ipynb' (highlighted with an orange box and selected, modified 'an hour ago'), 'Scaling_model_training.ipynb' (modified '2 minutes ago'), and 'utils.py' (modified 'an hour ago'). A large black arrow points from the text 'Meet me here!' on the left towards the 'Scaling_batch_inference.ipynb' file in the file tree. The main content area displays the first few lines of the 'Scaling_batch_inference.ipynb' notebook:

```
Scalable Batch Inference with Ray
```

About this notebook

Is this module right for you?

This module presents several approaches for scaling batch inference on Ray. Through hands-on practice with inference on a computer vision task, you will implement and compare different inference architectures to better understand Ray AIR and Ray Core.

To get the most out of this notebook, the following scenarios may apply to you:

- You observe performance bottlenecks when working on batch inference problems in computer vision projects.

Mode: Command No Kernel | Connecting Ln 1, Col 1 Scaling_batch_inference.ipynb



If I wanted to perform distributed batch inference with a high-level approach, I would choose to use...

- ① Start presenting to display the poll results on this slide.



**When we create a
BatchPredictor, we specified
a Checkpoint. Why?**

- ① Start presenting to display the poll results on this slide.



**What is the fundamental difference
between using Ray Tasks vs. Ray Actors
for batch inference?**

- ① Start presenting to display the poll results on this slide.

slido



Audience Q&A Session

- ⓘ Start presenting to display the audience questions on this slide.

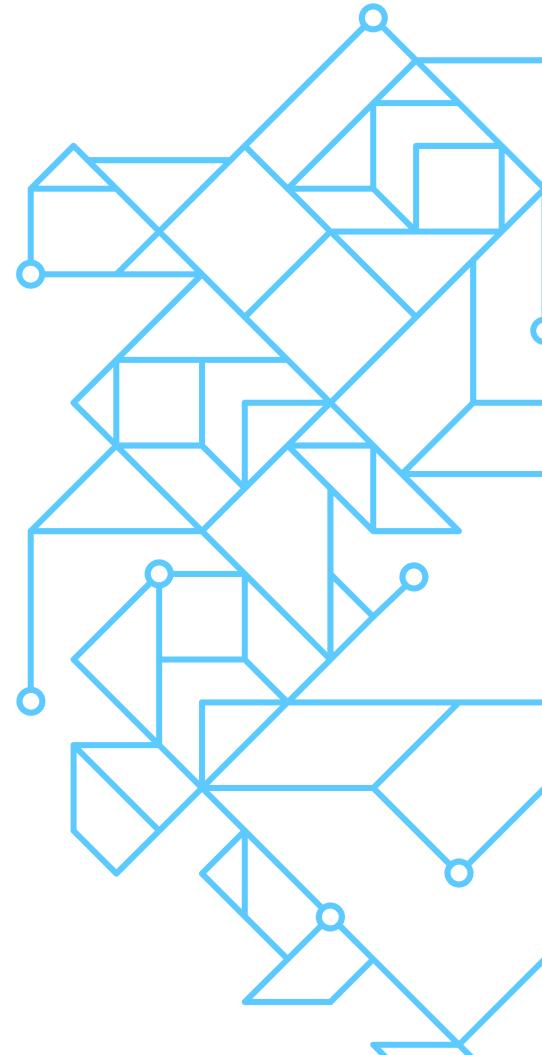


**Time for a
Break!**

30 minutes.

Wrap Up & Survey

It's been a blast.





Today we learned...



Distributed model training.

Framework-agnostic distributed data parallelism.



Observability.

Understanding the internals of your Ray cluster.



Scaling batch inference.

Generating predictions on new data.



Fill out the survey.

🔗 Go to bit.ly/acm-ray-feedback-2

Remember that we will draw winners for free Ray Summit tickets from completed submissions of both today and tomorrow's surveys.

Winners will be drawn and announced shortly after the event. You will be notified via email if you've won.



Let's check the leaderboard.

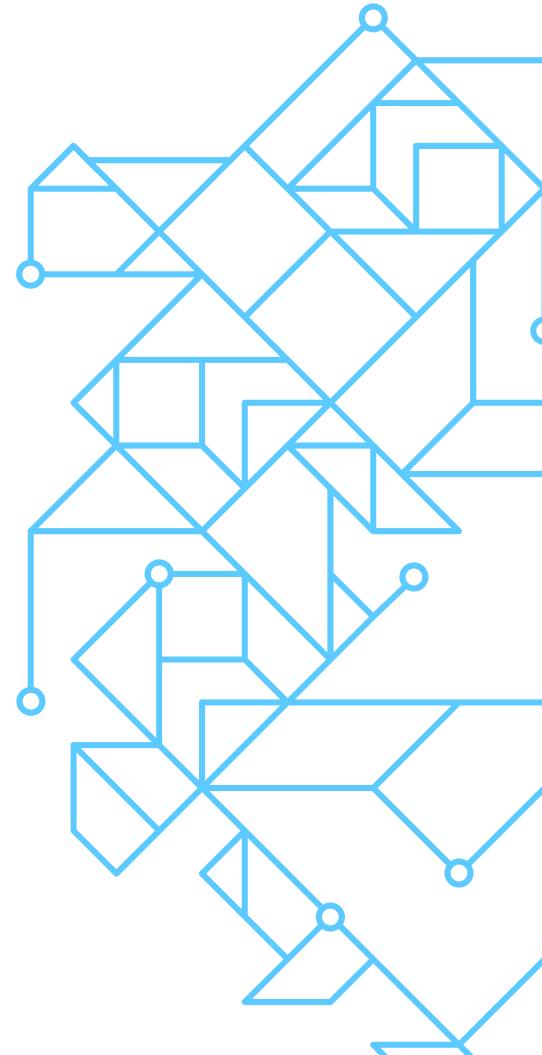


Today's copies of "Learning Ray"
go to...

*Congratulations to everyone who participated for
helping to create an engaging and fun
environment!*

What's Next?

For the Ray community.







What does an open source community mean to you?

What do “we” hope to accomplish with a community?

What makes a community successful?

Value

→ **Useful**
communities
create *value*



Engagement

→ **Engaged**
communities
foster
relationships
beyond *value*



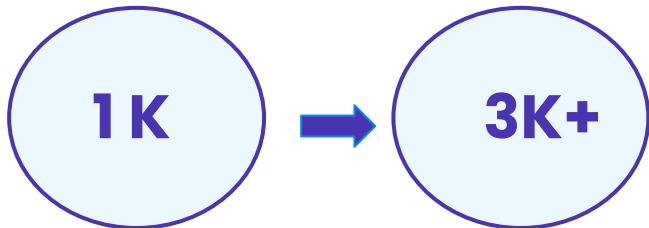
Shared vision

→ **Successful**
communities
foster & facilitate
a *shared vision*



Aspired Growth & Activities for Ray Community ... 2023

Meetups Membership



National/Global Ray Meetups

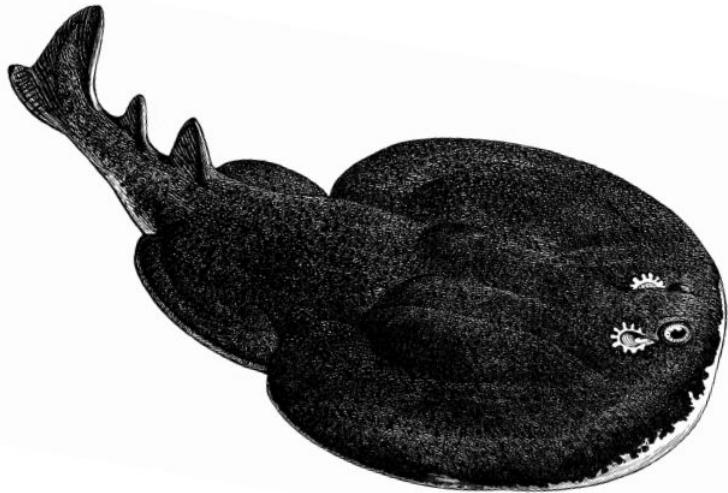


- Ray Summit 2023
- Ray Saturdays and more training tutorials....
- Ray AMAs + Office hours
- Ray talks/tutorials at major ML/AI/Python conferences

O'REILLY®

Learning Ray

Flexible Distributed Python for Machine Learning



Max Pumperla,
Edward Oakes
& Richard Liaw

O'REILLY®

Scaling Python with Ray

Adventures in Cloud and Serverless Patterns



Holden Karau &
Boris Lublinsky

Foreword by Robert Nishihara

Copyrighted Material

Copyrighted Material



Upcoming events



Ray NYC Community Meetup

Virtual meetup featuring Nixtla and Daft Dataframes



AWS and Anyscale

In-person in NYC and SF at AWS startup loft.



Ray and Anyscale

Livestream of infrastructure for foundation models.



Connect with the community.



Join the community

[Attend events](#), [subscribe to newsletter](#), [follow on Twitter](#).



Get support

[Join Ray Slack](#), [ask questions on forum](#), [open an issue](#).



Contribute to Ray

[Read contributor guide](#), [create a pull request](#).

Thank you!
We hope to meet again.

