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.

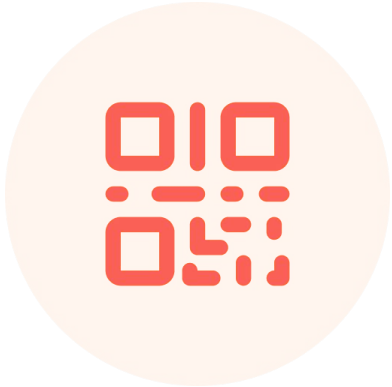
 Slido.com

Participate in polls, quizzes, and Q&A.

 Survey

One feedback survey per day.

slido



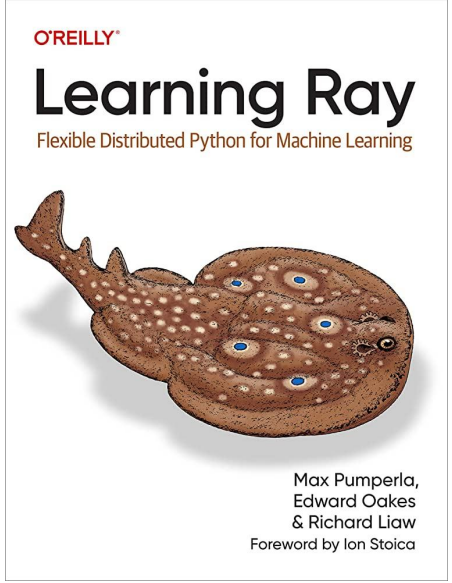
Join at slido.com
Enter code #ray

ⓘ Start presenting to display the joining instructions on this slide.

Prizes.

“Learning Ray” books

For the top finishers in our daily quiz.



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!

slido

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

ⓘ 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.

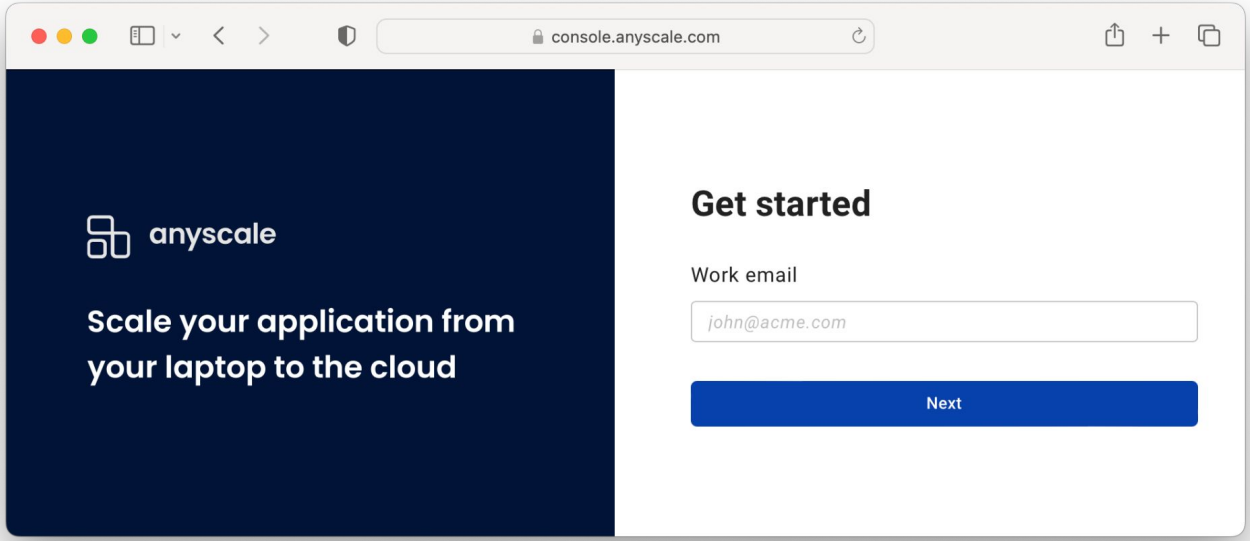
✅ Tech check.

 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.

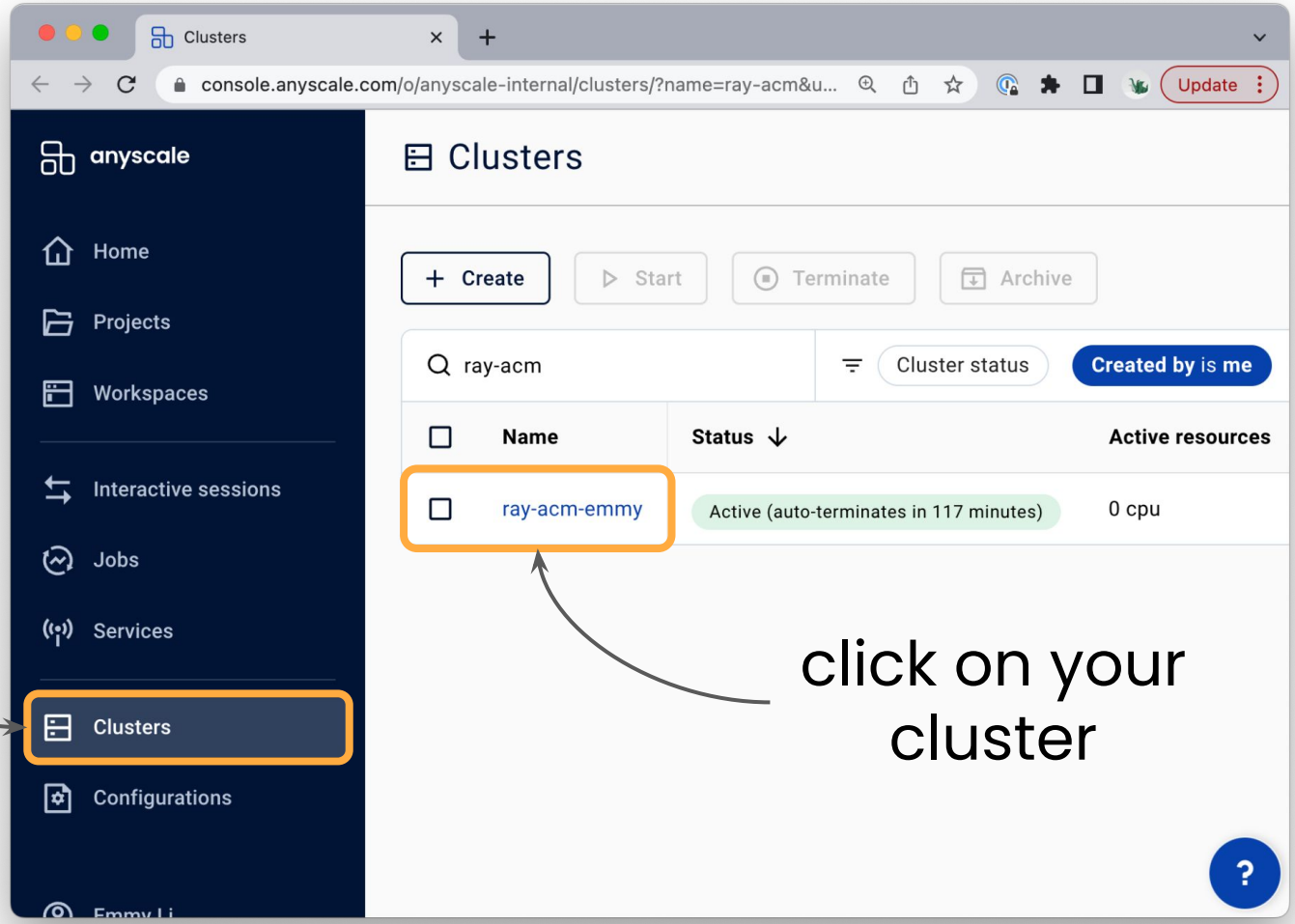
Anyscale Login

Link to Anyscale cluster: console.anyscale.com



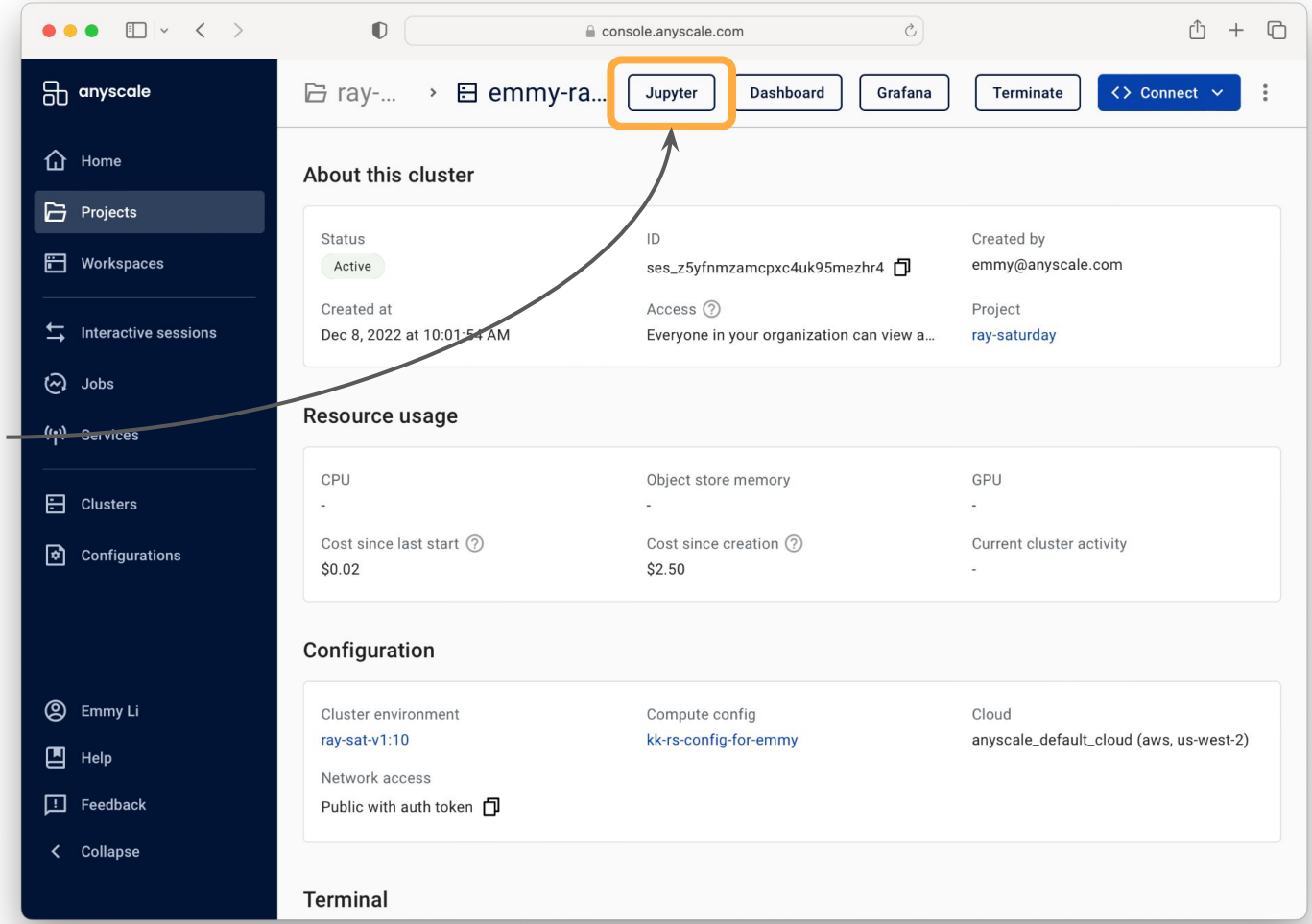
Check your **email** for your unique username and password!

select “Clusters”

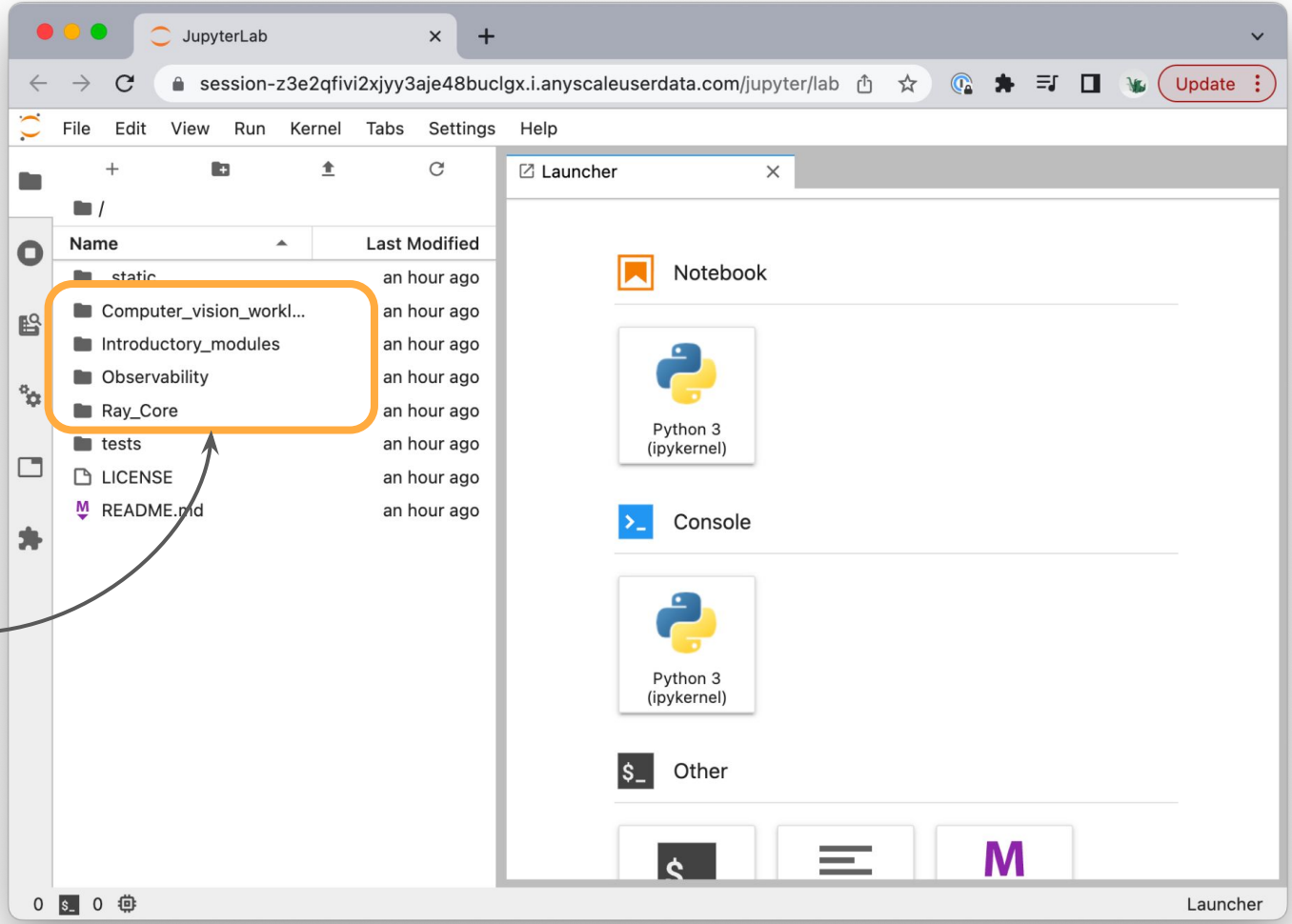


click on your cluster

“Start” the cluster, then select “Jupyter”



View modules here



slido

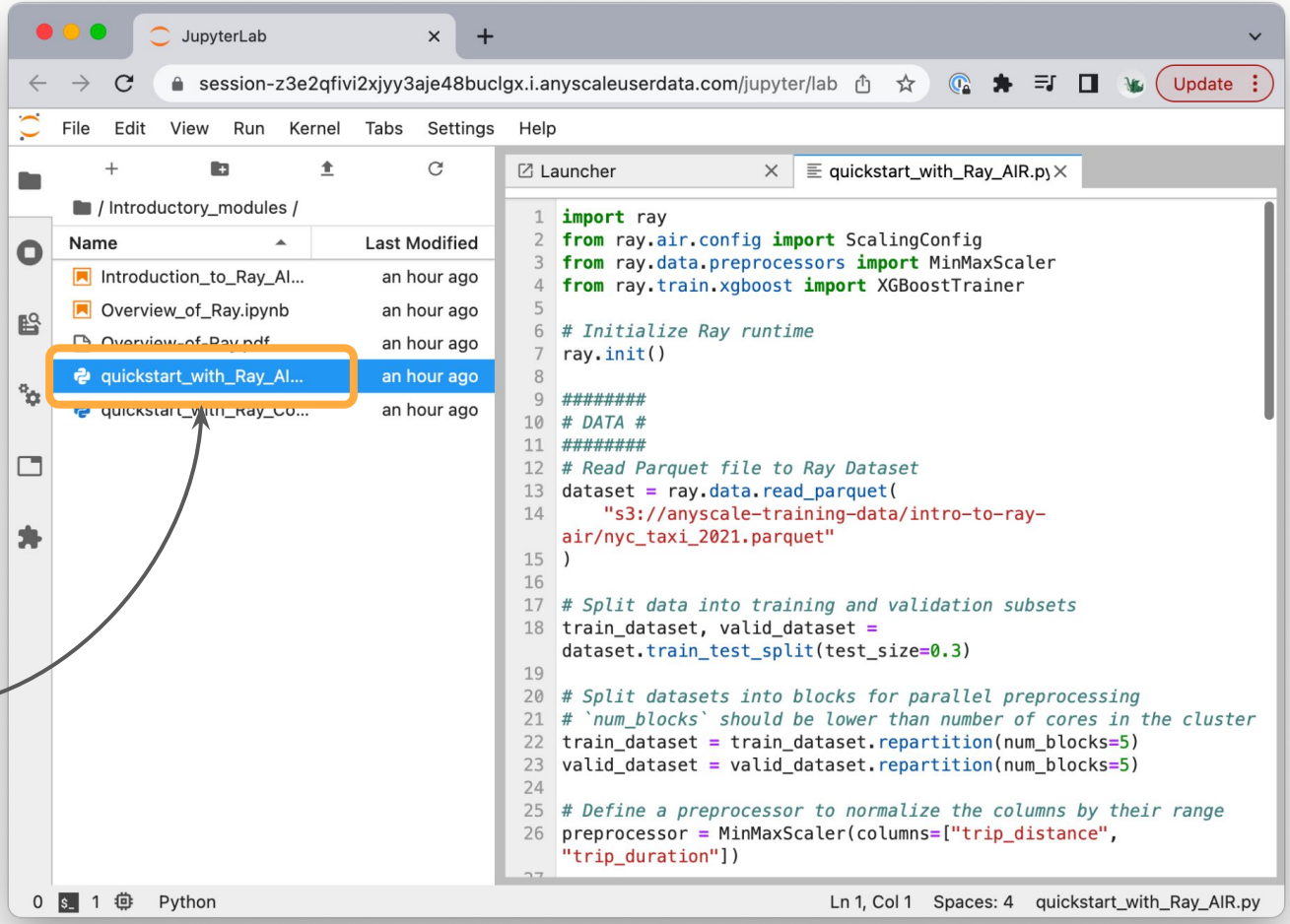
Audience Q&A Session

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

Overview of Ray

Hands-on coding lab.

Meet me here!



Introductory_modules/quickstart_with_Ray_AIR.py

slido

Audience Q&A Session

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

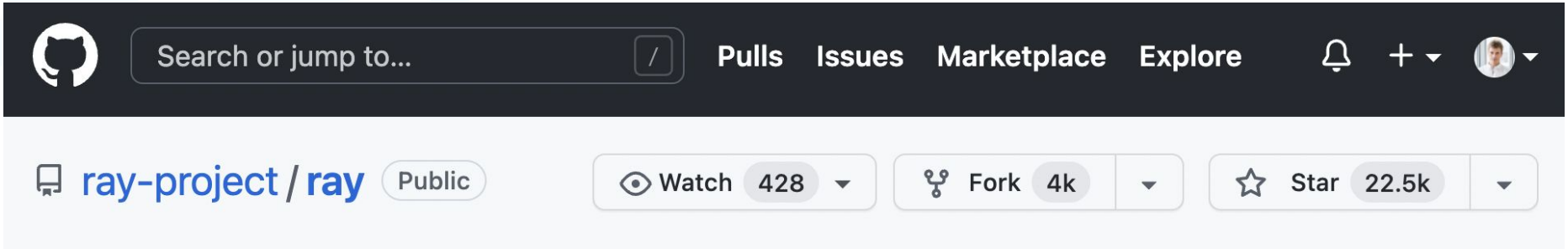
🍴 Time for Lunch!

60 minutes.

Overview of Ray

Take a tour through what Ray can do.

Ray project

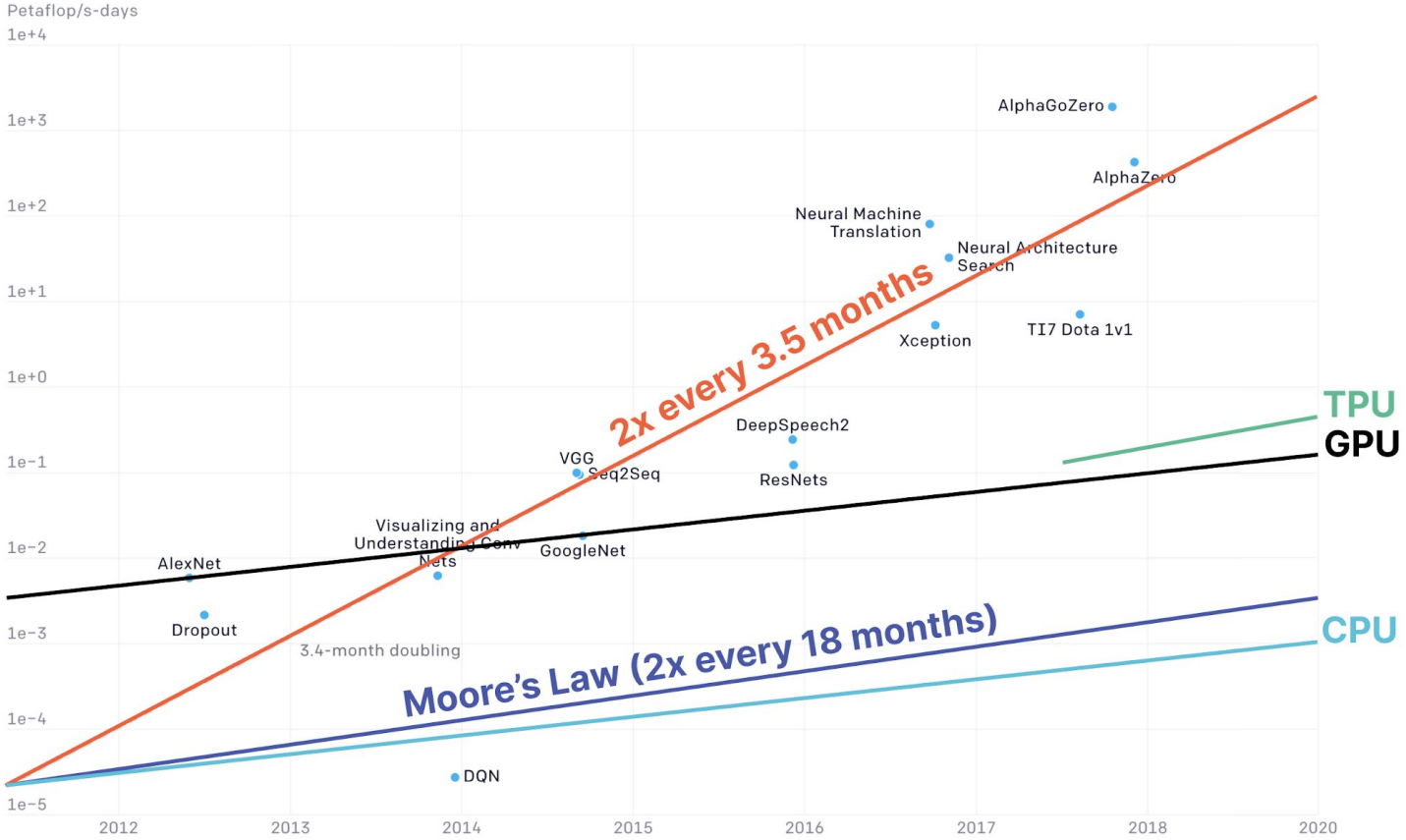


Ray is:
an open-source unified compute framework
that makes it easy to scale AI and Python workloads.

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

RAY

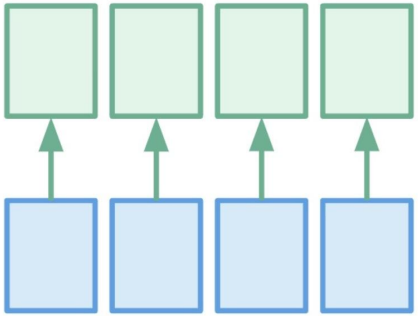
Distributed computing: a bit of context



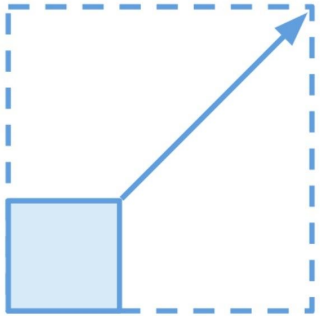
RAY

Key Ray characteristics

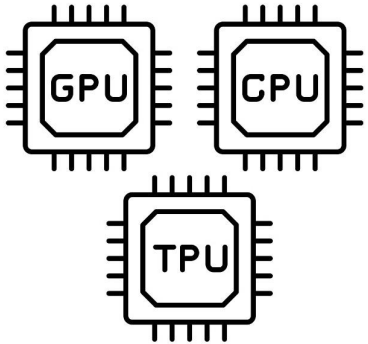
Python first approach



Simple and flexible API

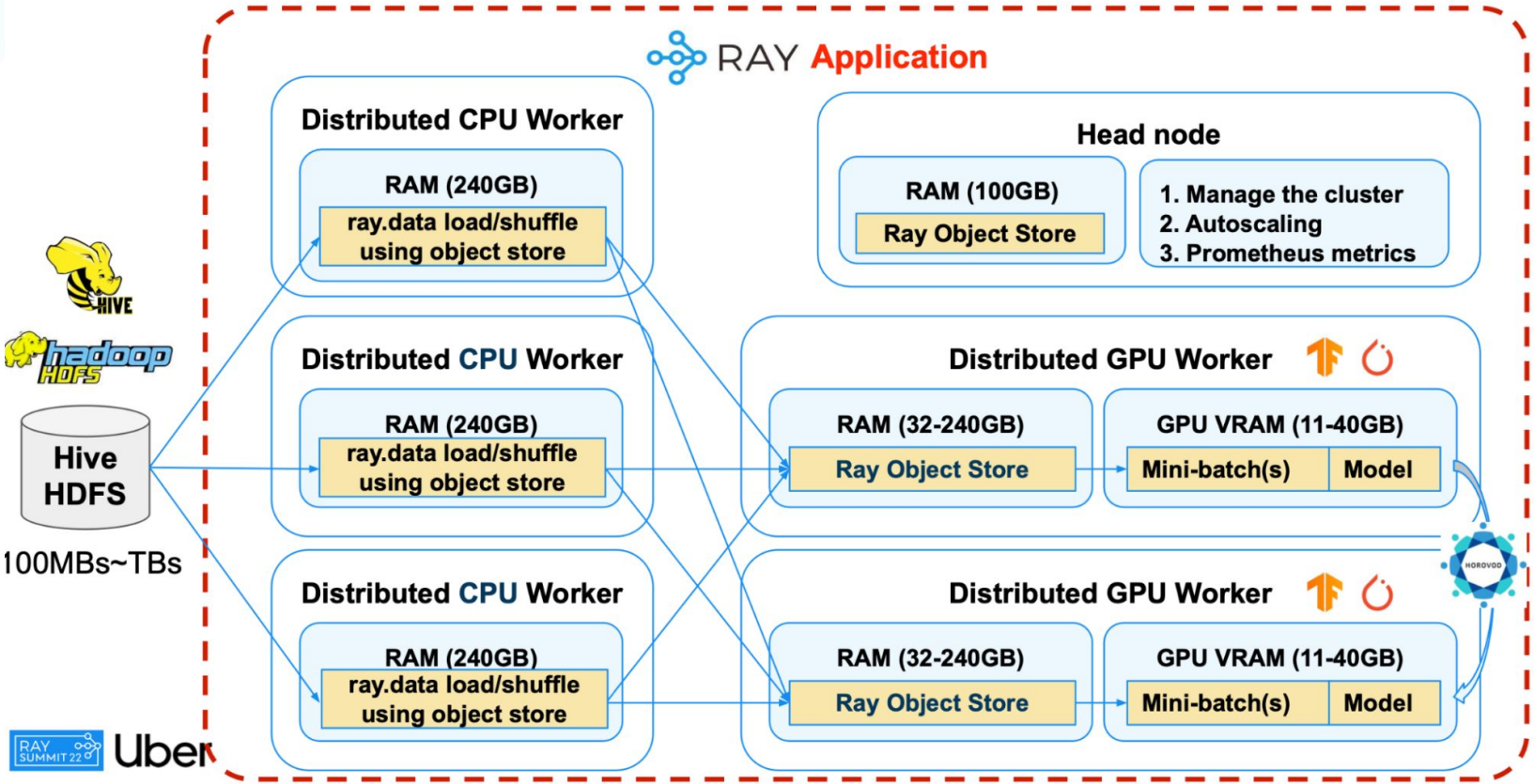


Scalability

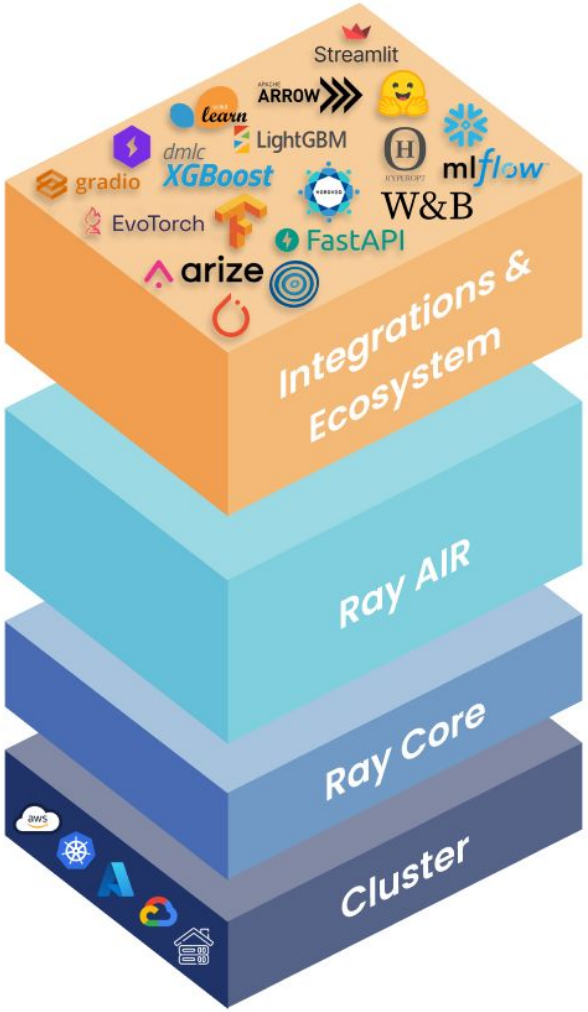


Support for heterogeneous hardware

Deep learning pipeline at Uber



Ray libraries



RAY

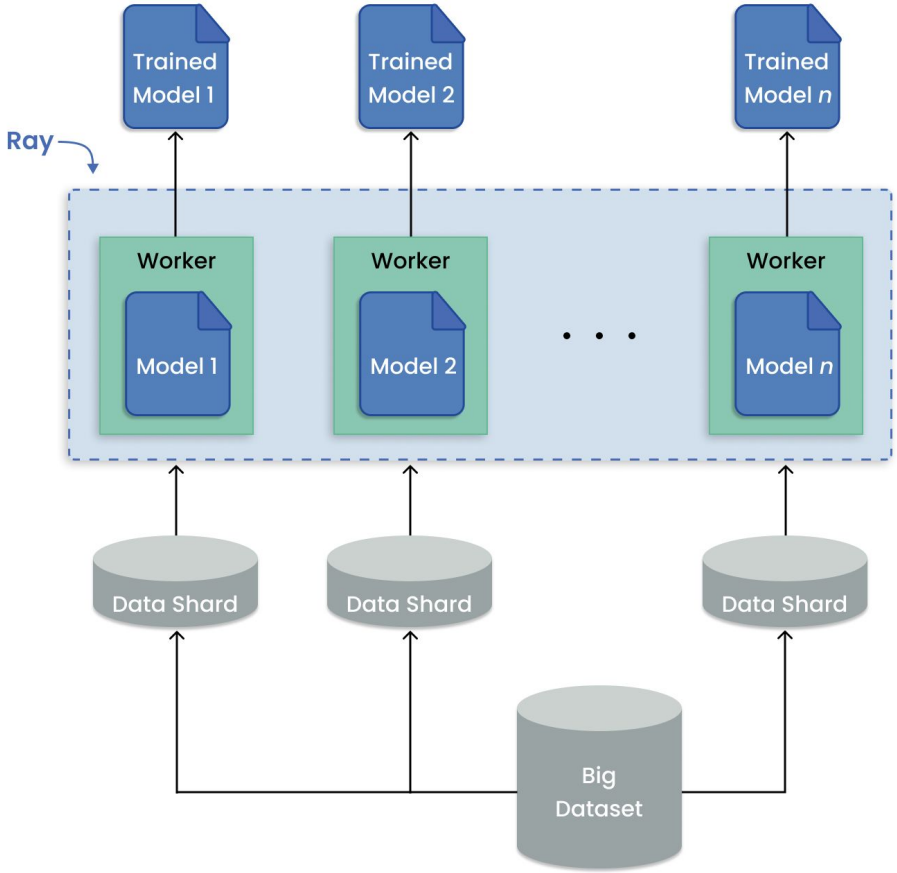
Ray AIR and integrations

RAY

Ray use cases

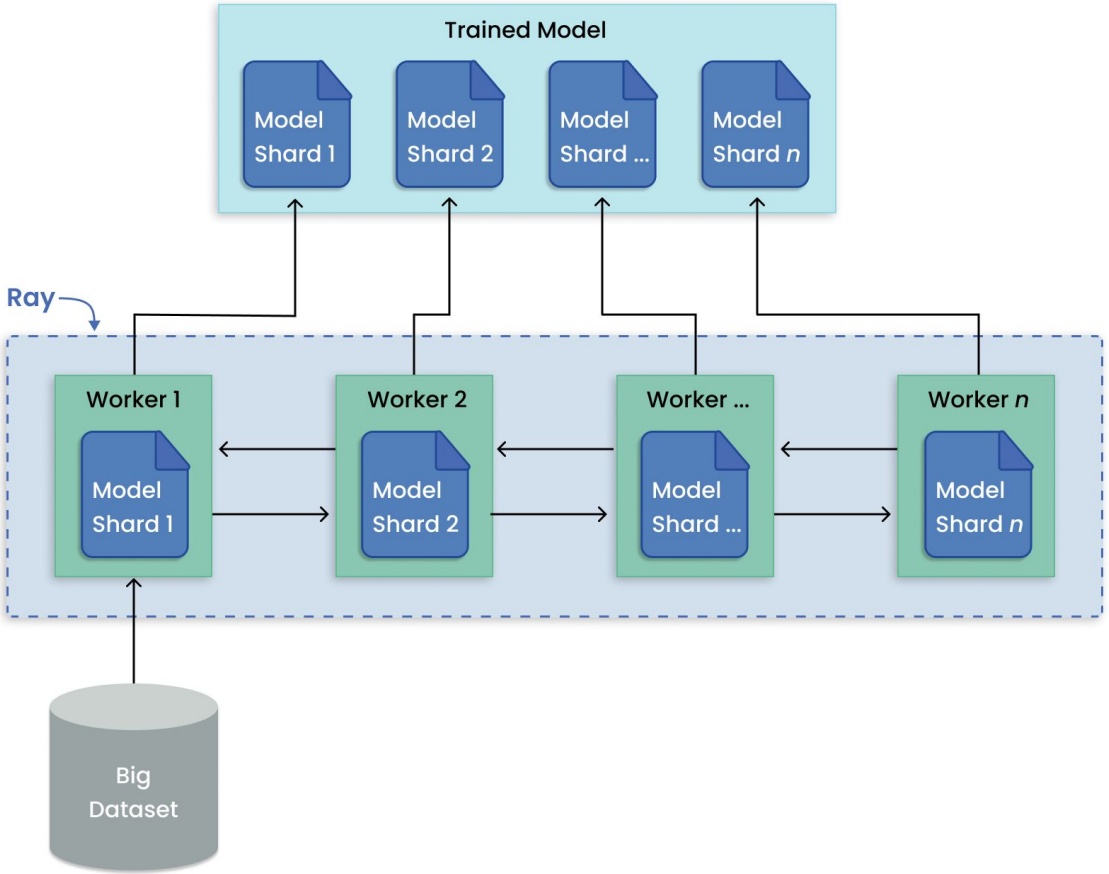
Scaling ML workloads

Parallel training of many models



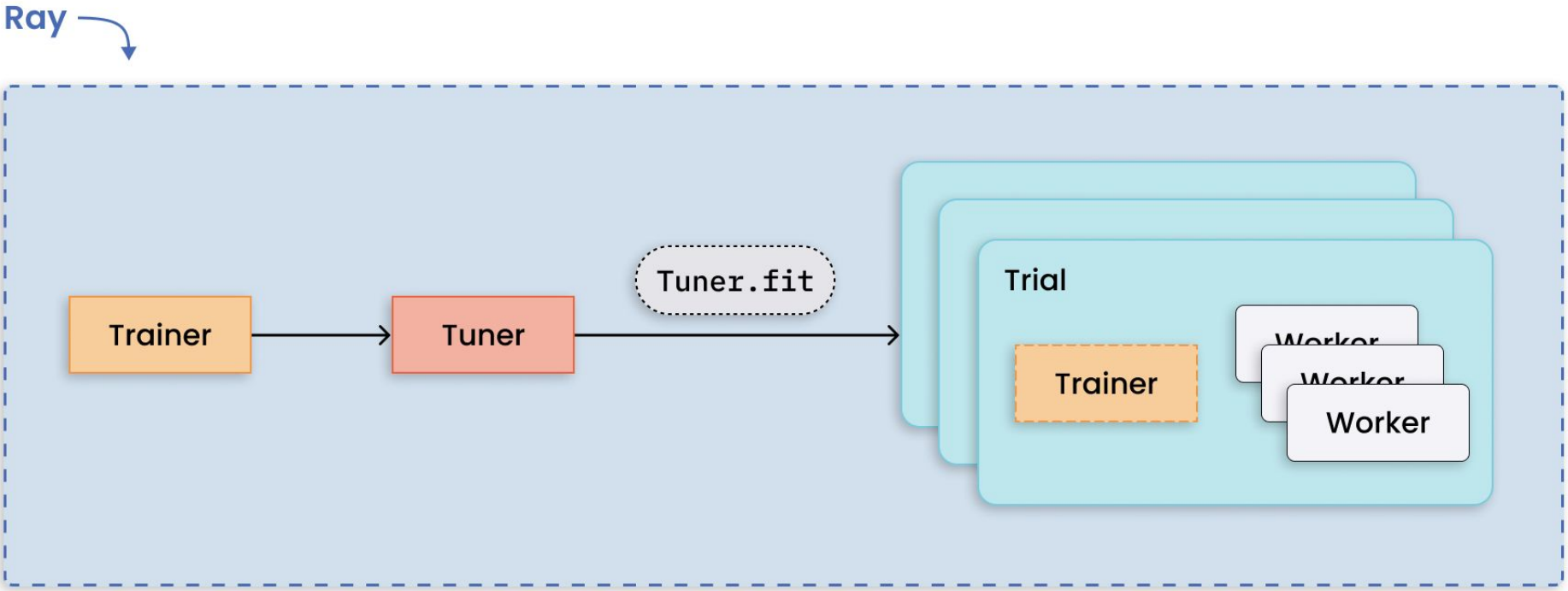
RAY

Distributed training of large models

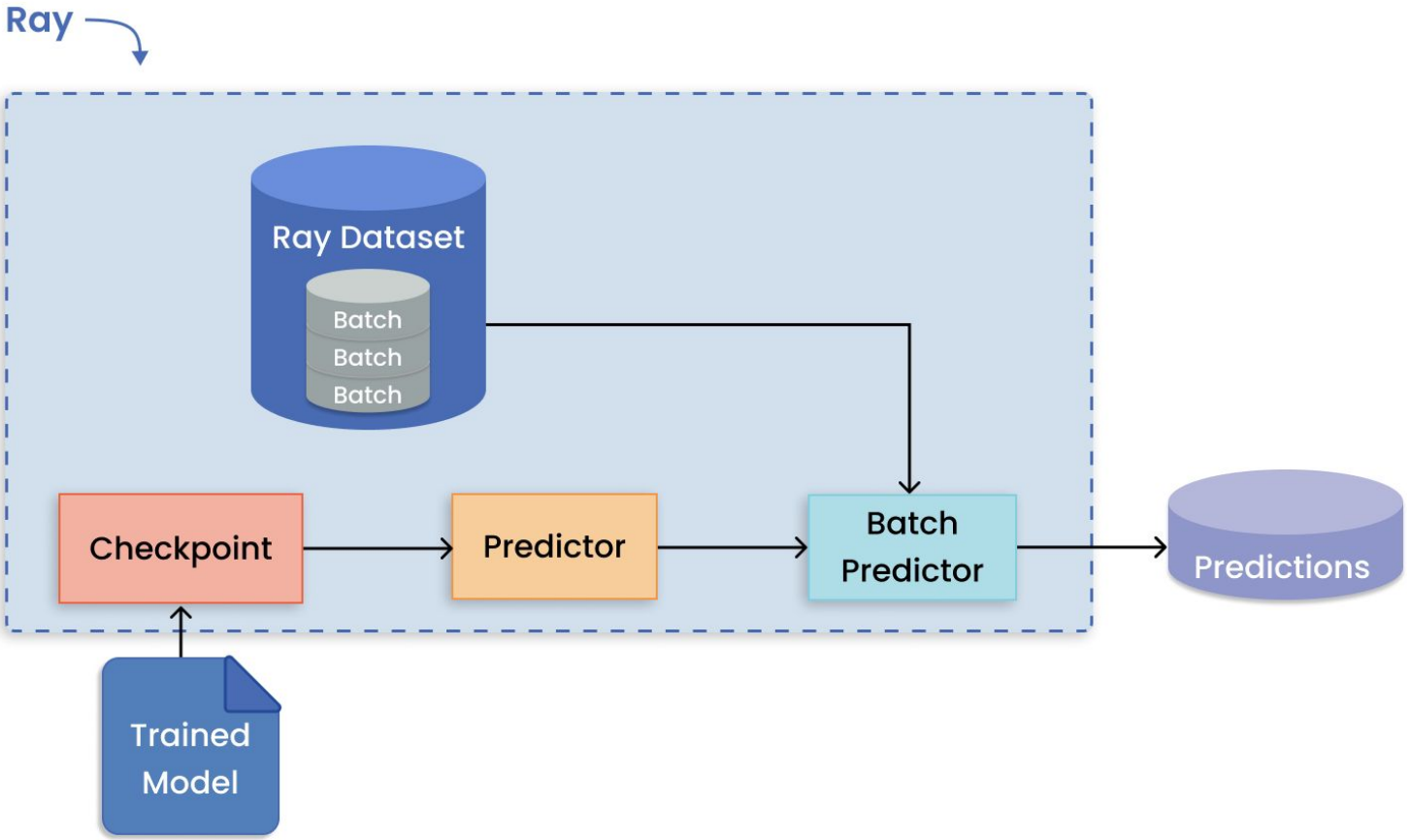


RAY

Managing parallel hyperparameter tuning experiments



Batch inference on CPUs and GPUs



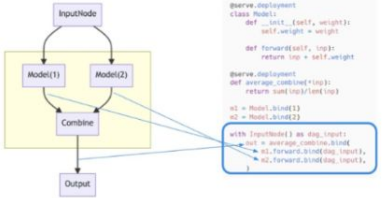
RAY

Multi-model composition for model serving

Chaining

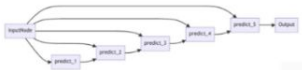

```
from ray import serve
from ray.serve.deployment_graph import InputNode

@serve.deployment
class Model:
    def __init__(self, weight):
        self.weight = weight

    def forward(self, inp):
        return inp + self.weight

m1 = Model.bind(1)
m2 = Model.bind(2)

with InputNode() as dag_input:
    out = m1.forward.bind(dag_input)
    out = m2.forward.bind(out)
```

Ensemble



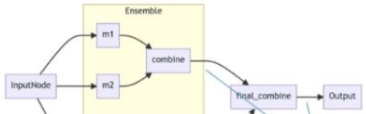

```
@serve.deployment
class Model:
    def __init__(self, weight):
        self.weight = weight

    def forward(self, inp):
        return inp + self.weight

@serve.deployment
def average_combine(*inp):
    return sum(inp)/len(inp)

m1 = Model.bind(1)
m2 = Model.bind(2)

with InputNode() as dag_input:
    out = average_combine.bind(
        m1.forward.bind(dag_input),
        m2.forward.bind(dag_input),
    )
```

Tree



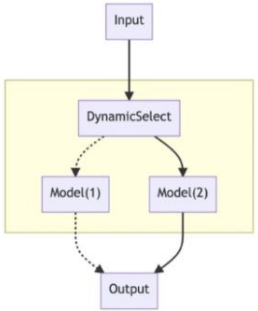
```
from ray import serve

@serve.deployment
def predict:
    pass

with InputNode() as inp:
    out = predict.bind(inp)
    for _ in range(4):
        out = predict.bind(out, inp)
```

Ensemble + Chaining




```
with InputNode() as inp:
    ensemble_out = combine.bind(
        m1.predict.bind(inp),
        m2.predict.bind(inp),
    )
    chain_out = m4.predict.bind(
        m3.predict.bind(inp),
    )
    final_out = final_combine(ensemble_out, chain_out)
```

Dynamic Selection

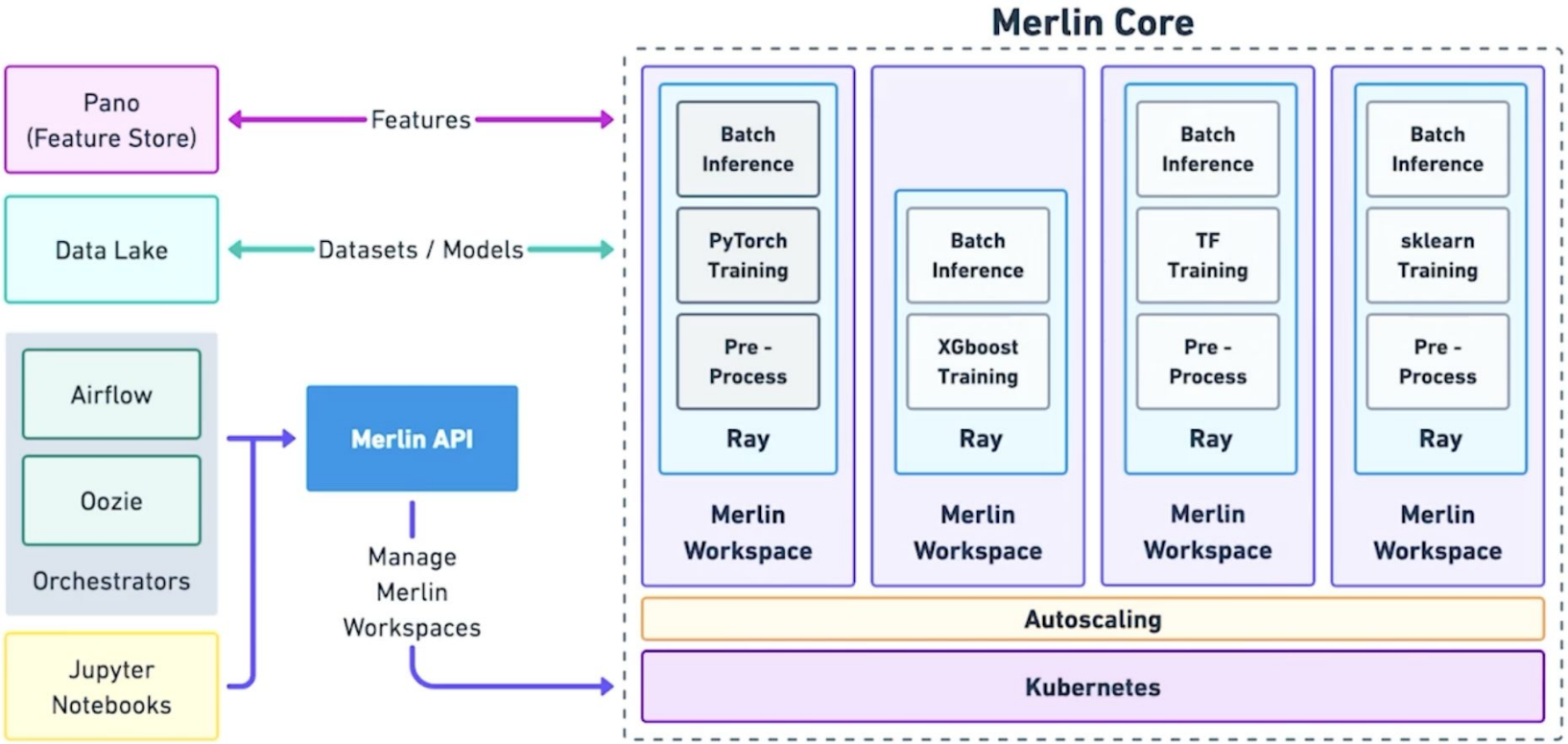


Embed imperative API
within declarative dataflow

```
@serve.deployment
class Model:
    def __init__(self, weight):
        self.weight = weight

    def forward(self, inp):
        return inp + self.weight

@serve.deployment
class RandomSelection:
    def __init__(self, m1, m2):
        (self.m1, self.m2) = m1, m2

    def forward(self, inp):
        if random.random() < 0.5:
            return self.m1.forward.remote(inp)
        else:
            return self.m2.forward.remote(inp)

m1 = Model.bind(1)
m2 = Model.bind(2)
select = RandomSelection.bind(m1, m2)

with InputNode() as dag_input:
    out = select.forward.remote(dag_input)
```

RAY

ML platform



slido

Which of the following is true about Ray?

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

slido

What is one primary benefit of using Ray?

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

slido

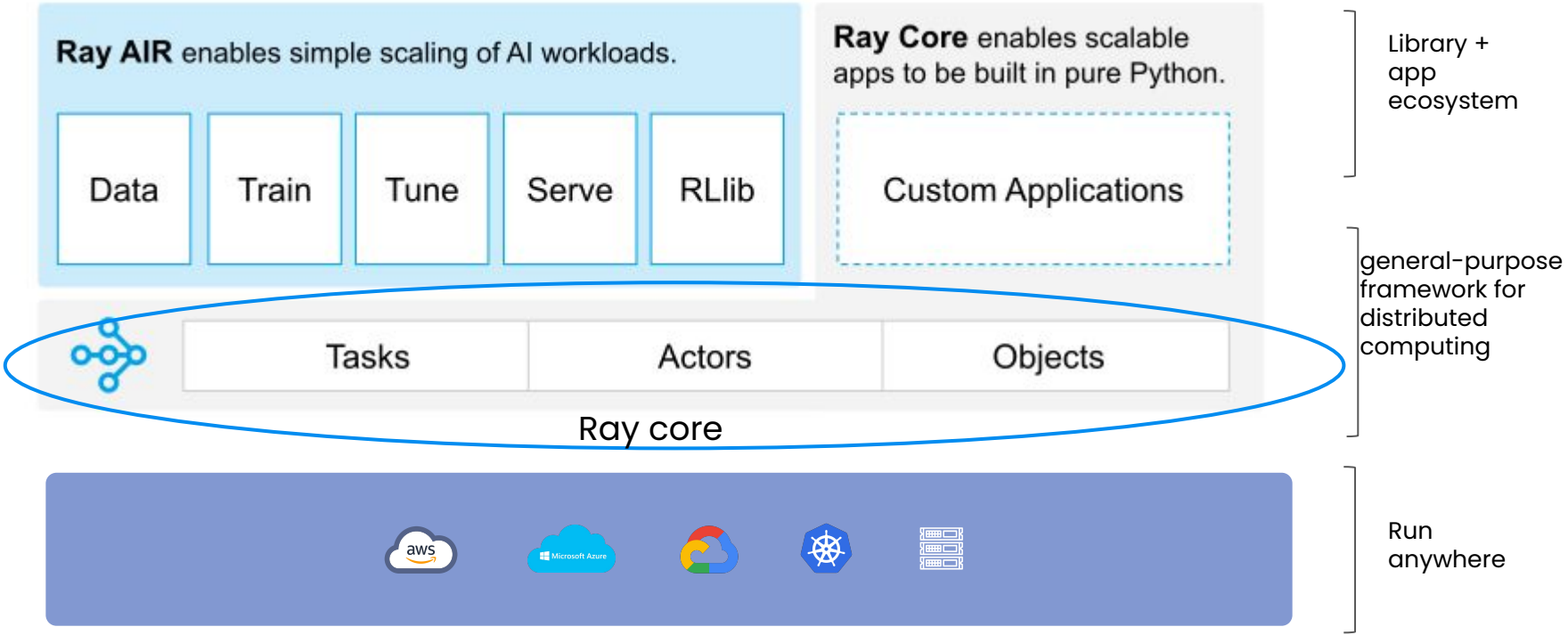
Audience Q&A Session

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

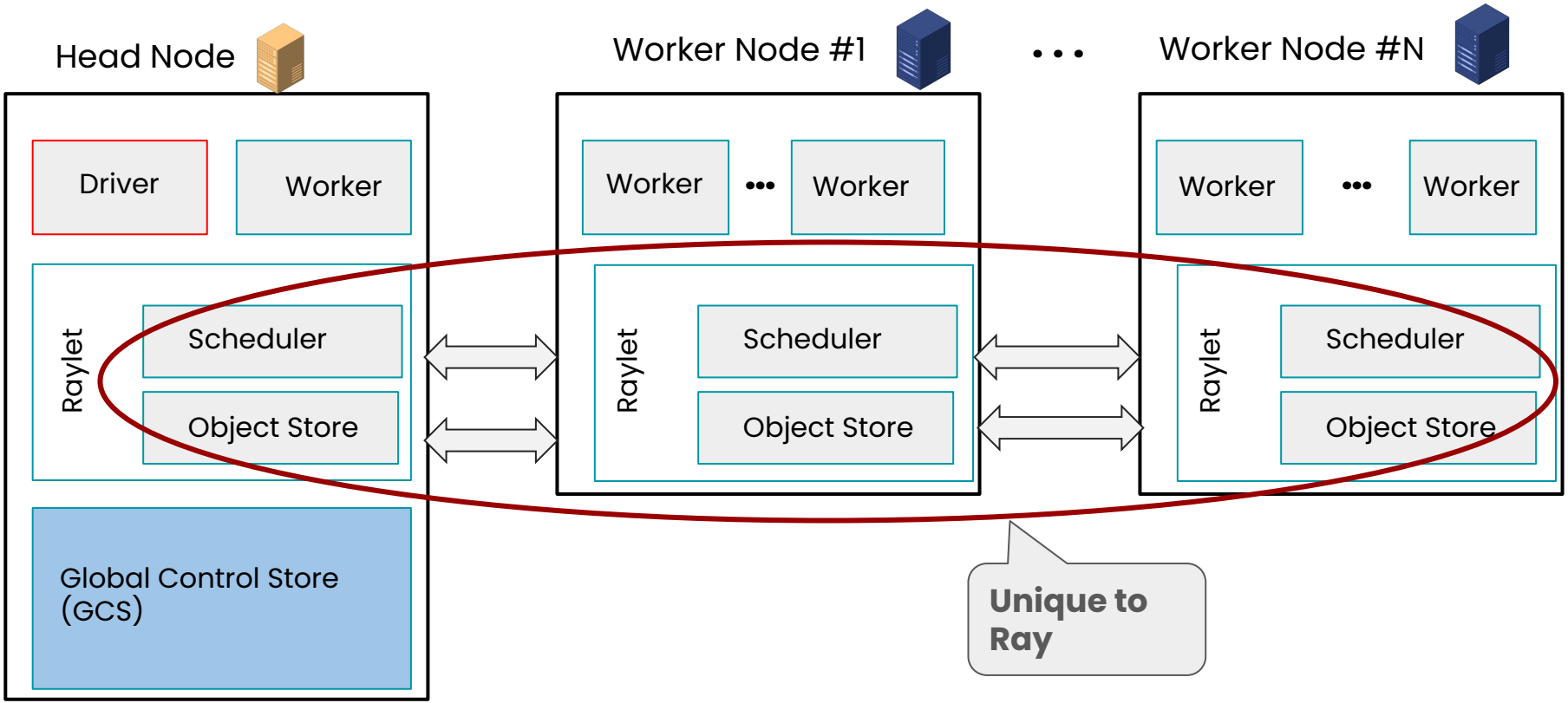
Ray Core

Learn the fundamentals with remote functions.

A Layered Cake and Ecosystem



An anatomy of a Ray cluster



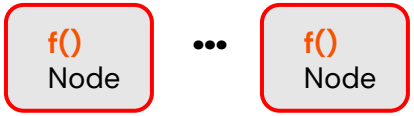
Python → Ray APIs

```
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()
```

Distributed




```
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):
        ...
cls = Cls.remote()
cls.f.remote(a)
```

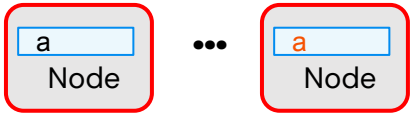
Distributed


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

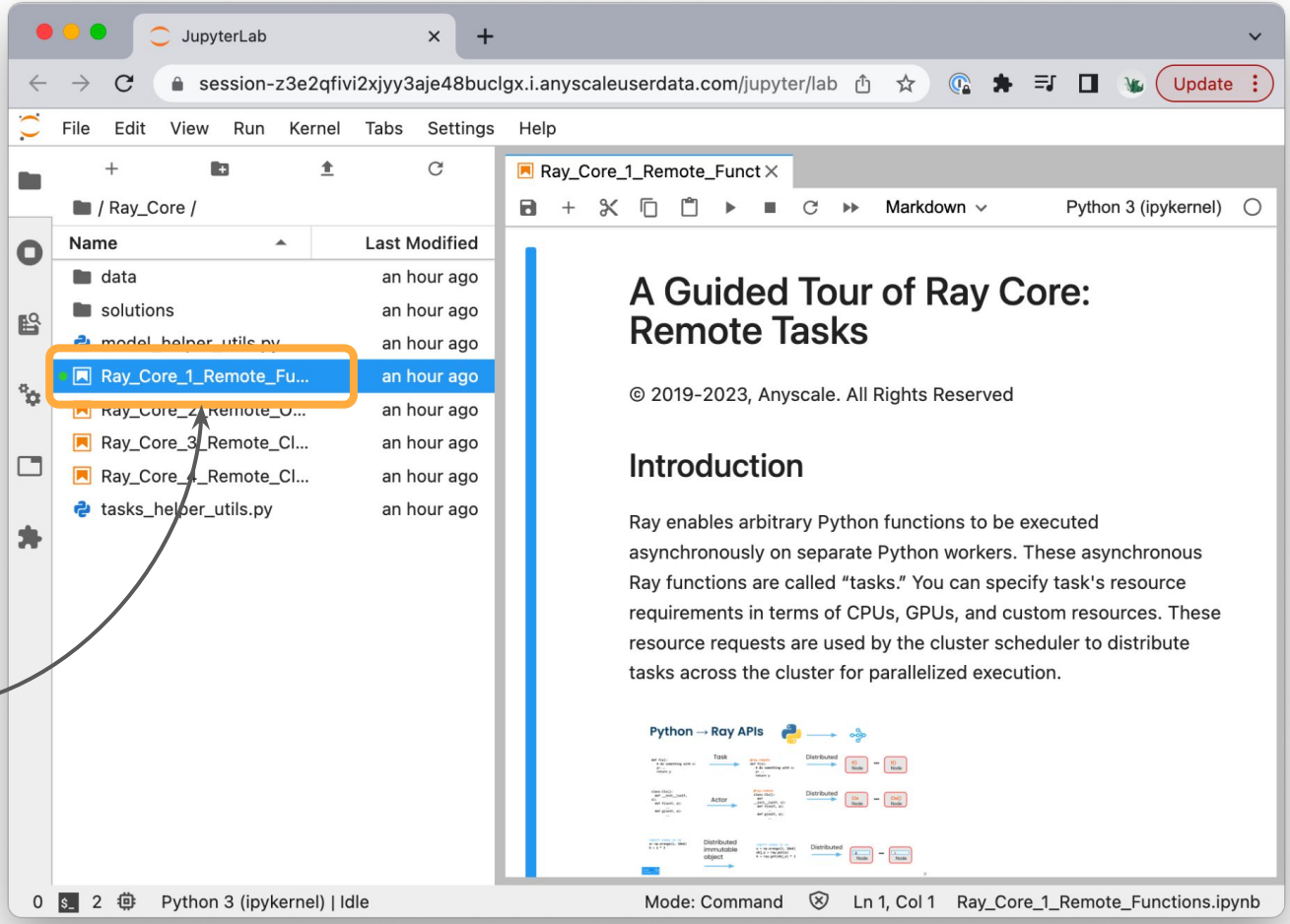
Distributed immutable object

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

Distributed



Meet me here!



Ray_Core/Ray_Core_1_Remote_Functions.ipynb

slido

Audience Q&A Session

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

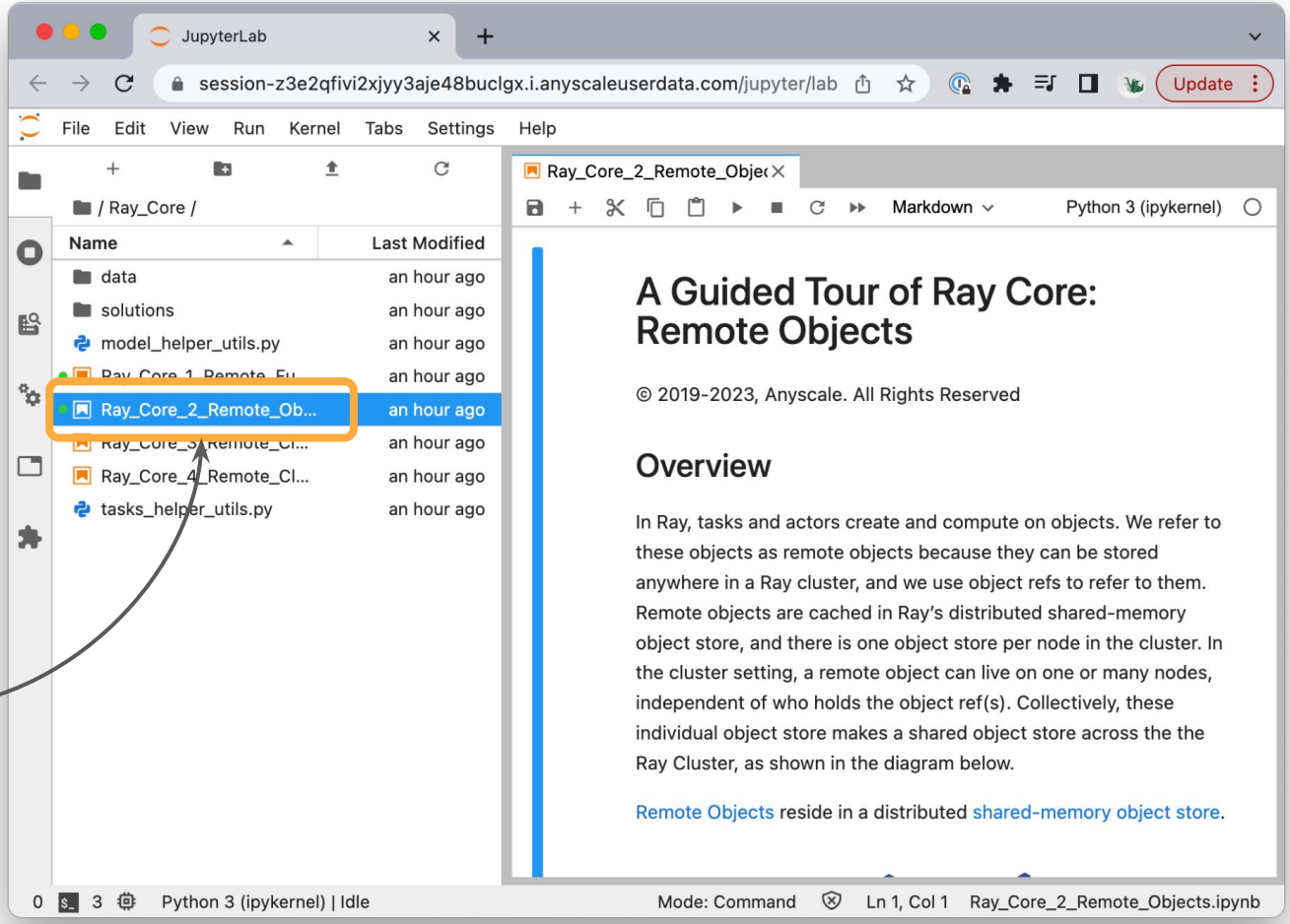
Time for a Break!

30 minutes.

Ray Core

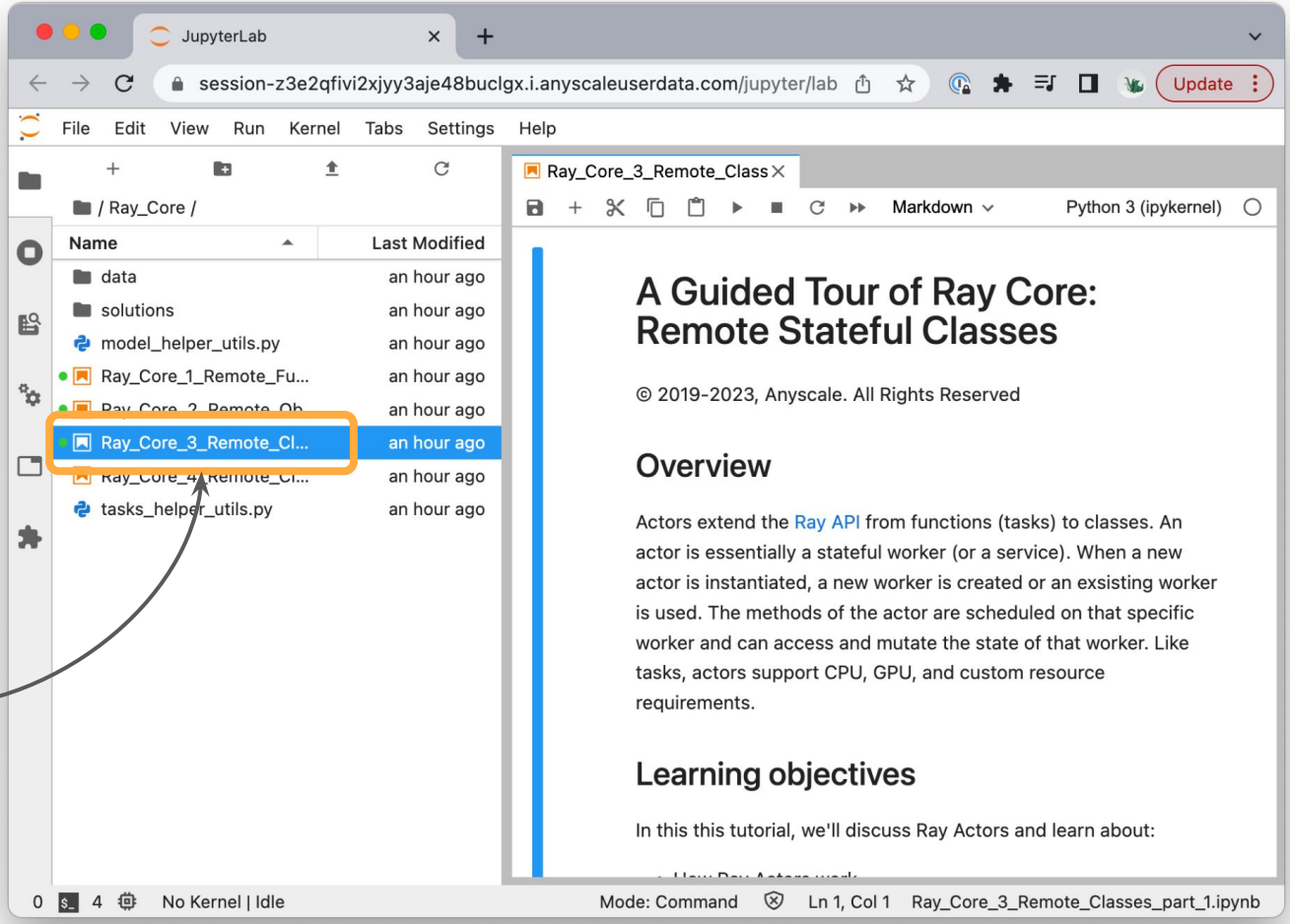
Remote objects and classes.

Meet me here!



Ray_Core/Ray_Core_2_Remote_Objects.ipynb

Meet me here!



Ray_Core/Ray_Core_3_Remote_Classes_part_1.ipynb

slido

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

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

slido

Which of the following is true about Ray Core?

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

slido

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.

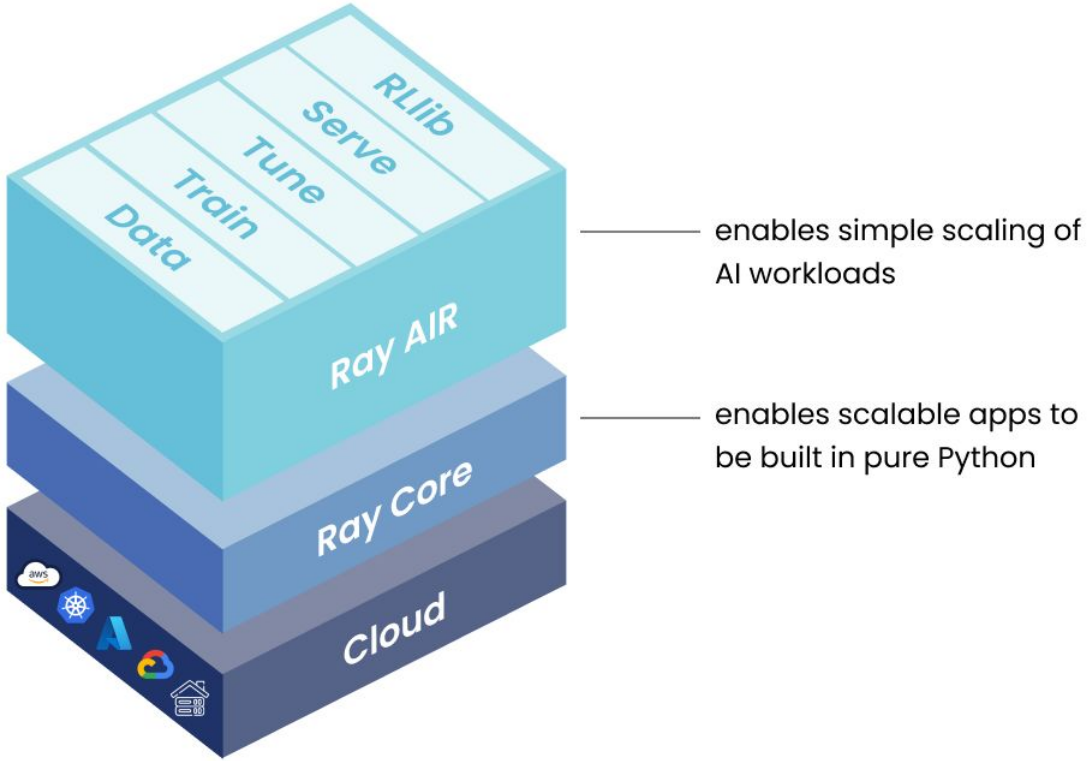
Time for a Break!

30 minutes.

Introduction to Ray AIR

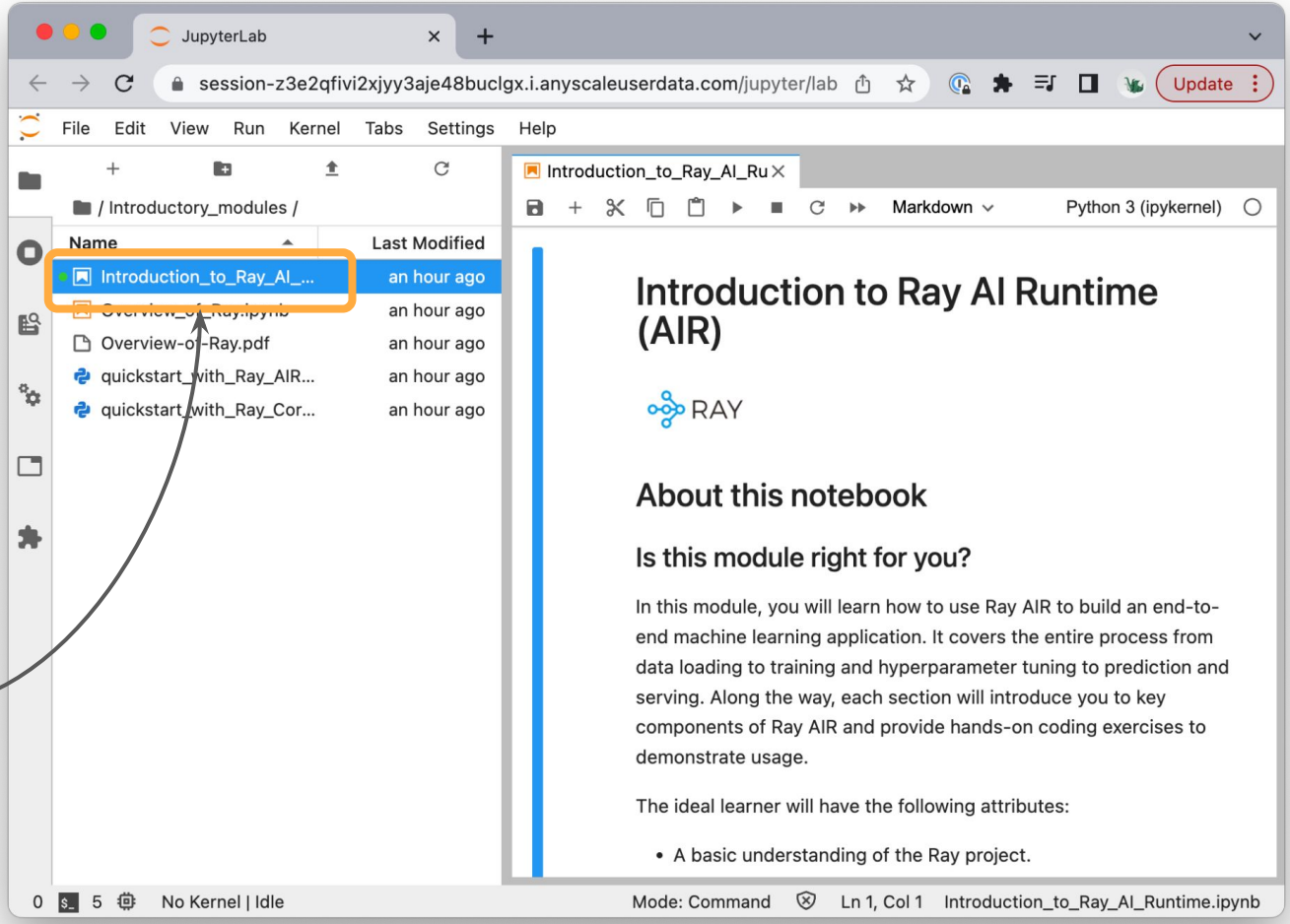
Scalable, end-to-end machine learning.

Where Ray AIR sits.



↔ End-to-end ML scaling.

Meet me here!



Introductory_modules/Introduction_to_Ray_AIR.ipynb

slido

Which of the following is NOT a Ray library?

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

slido

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

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

slido

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

- 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.

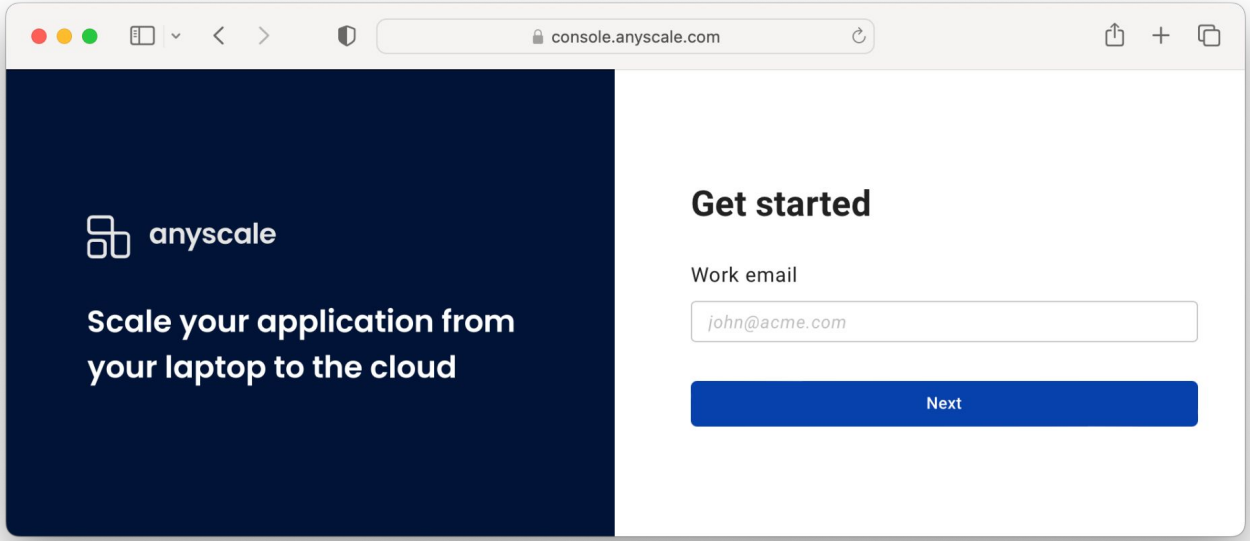
✅ Tech check.

 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.

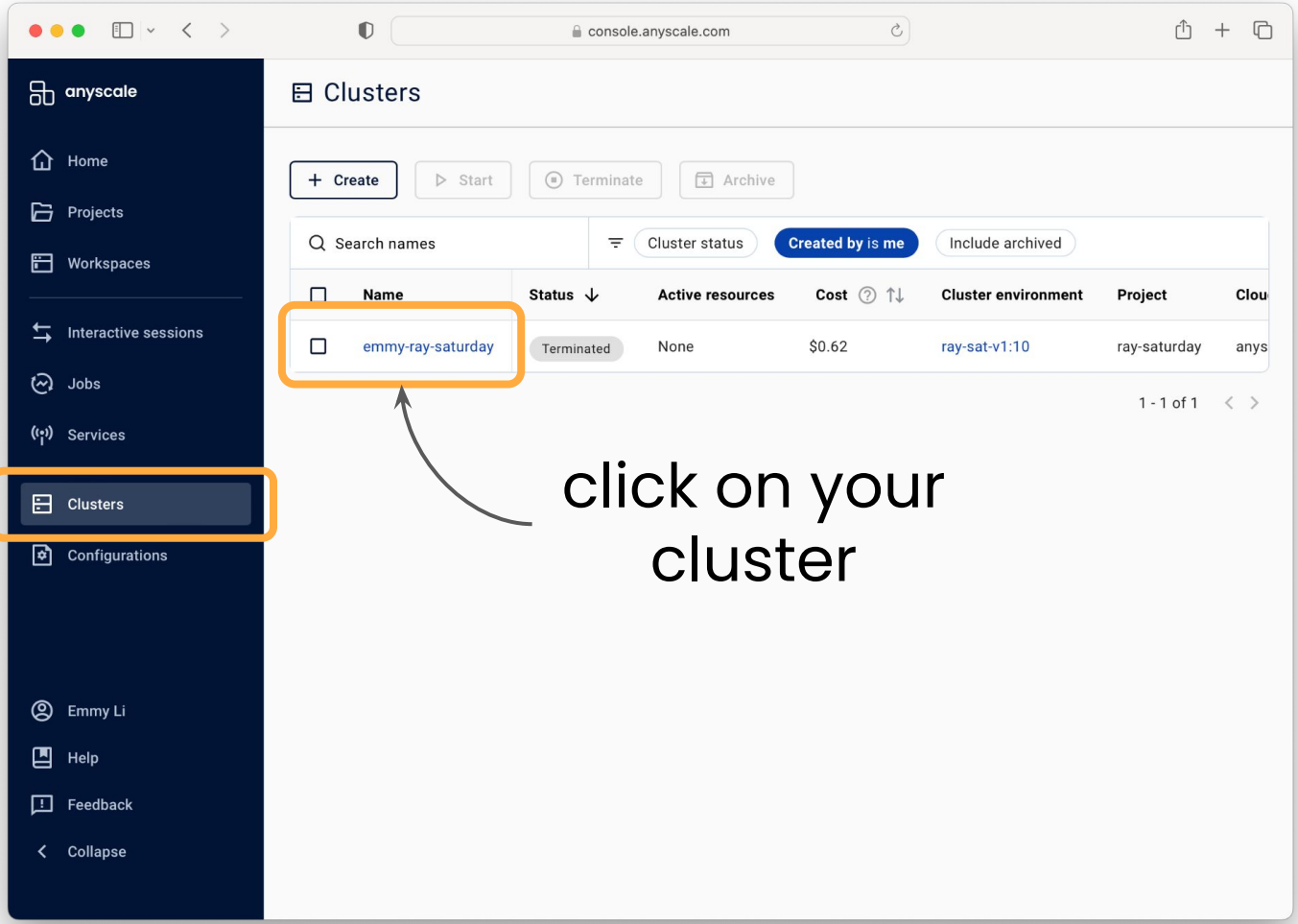
Anyscale Login

Link to Anyscale cluster: console.anyscale.com



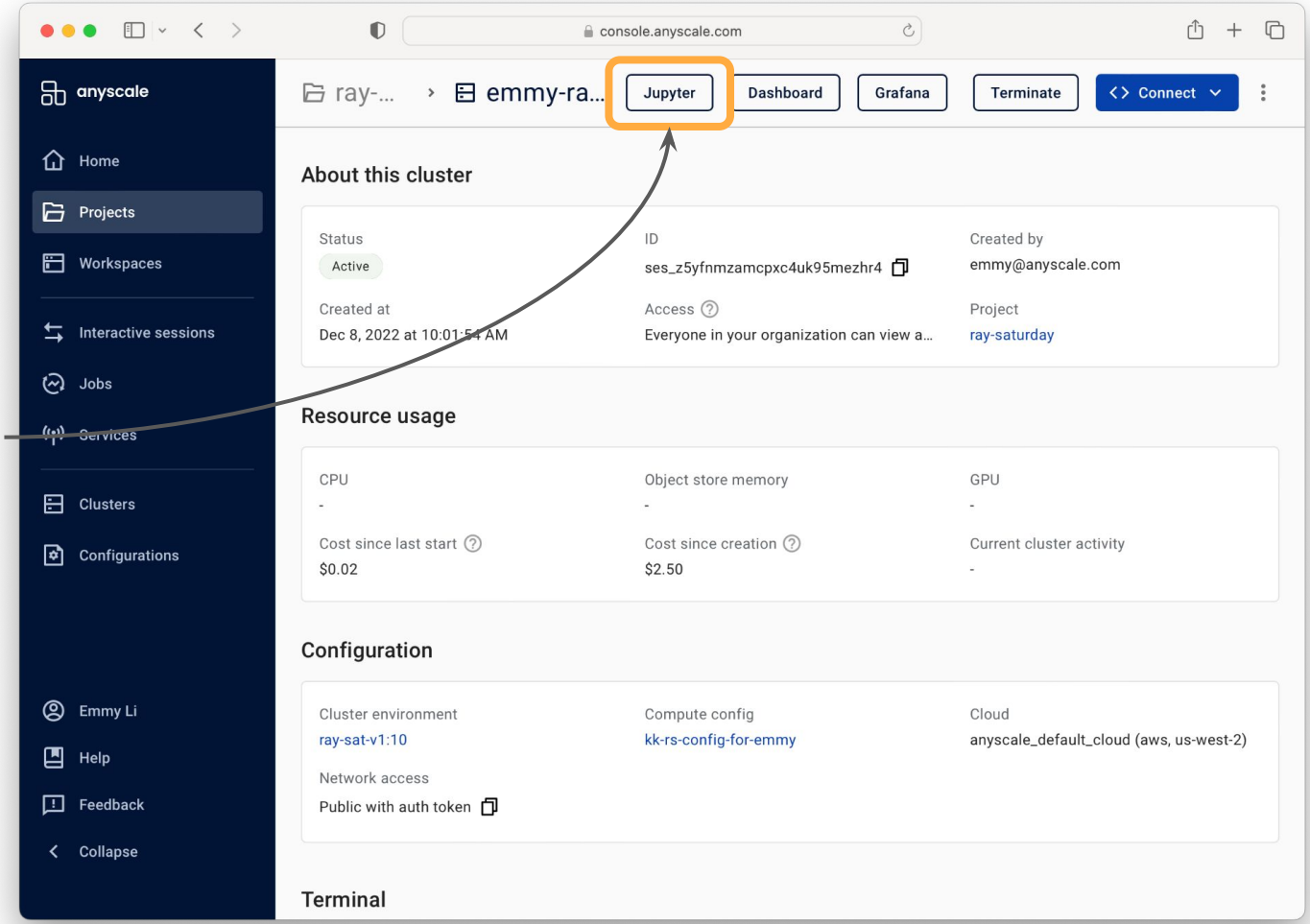
Check your **email** for your unique username and password!

select “Clusters”

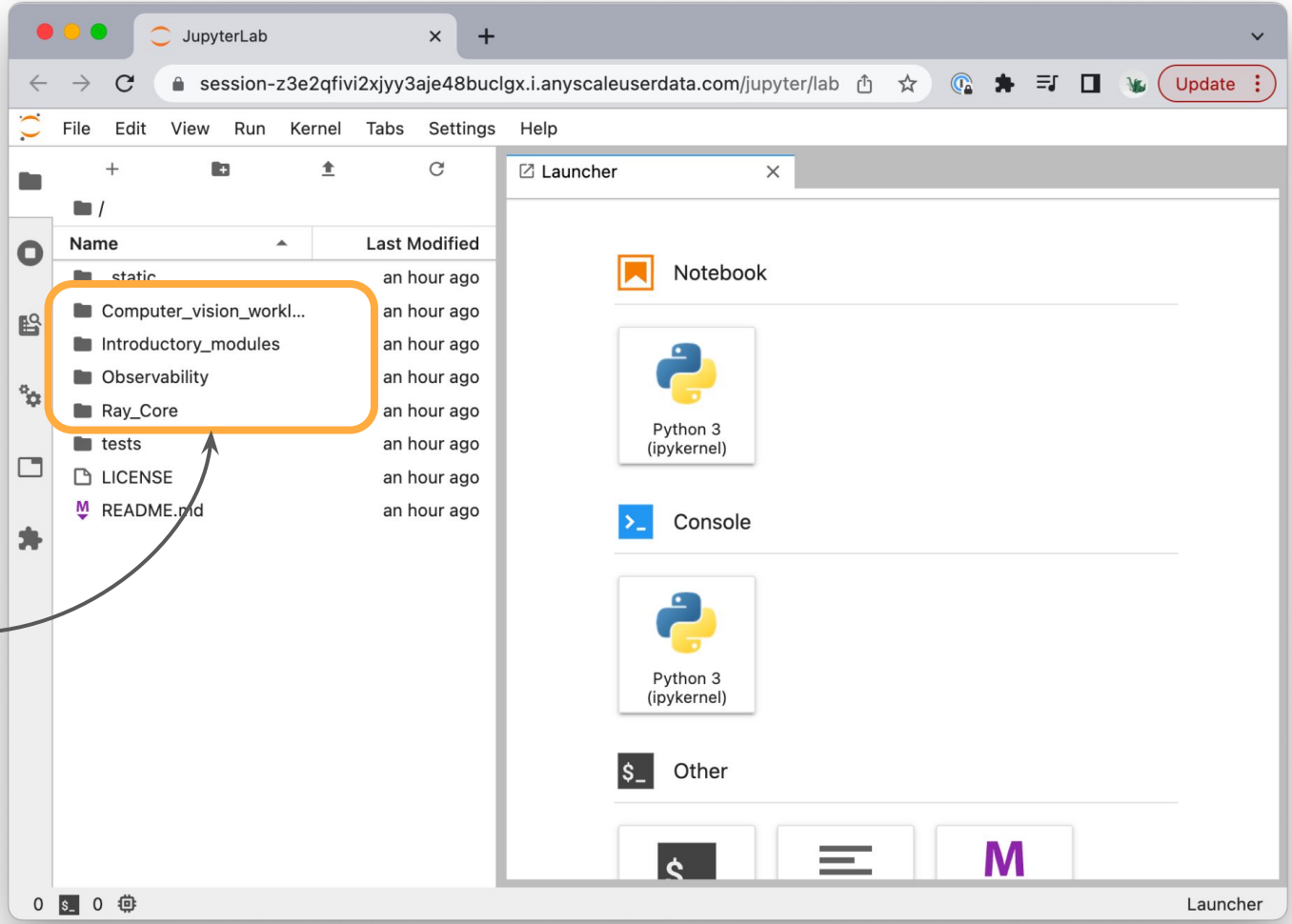


click on your cluster

“Start” the cluster, then select “Jupyter”



View modules here



The Plan

Here’s what we’ll do today.

Today's agenda.

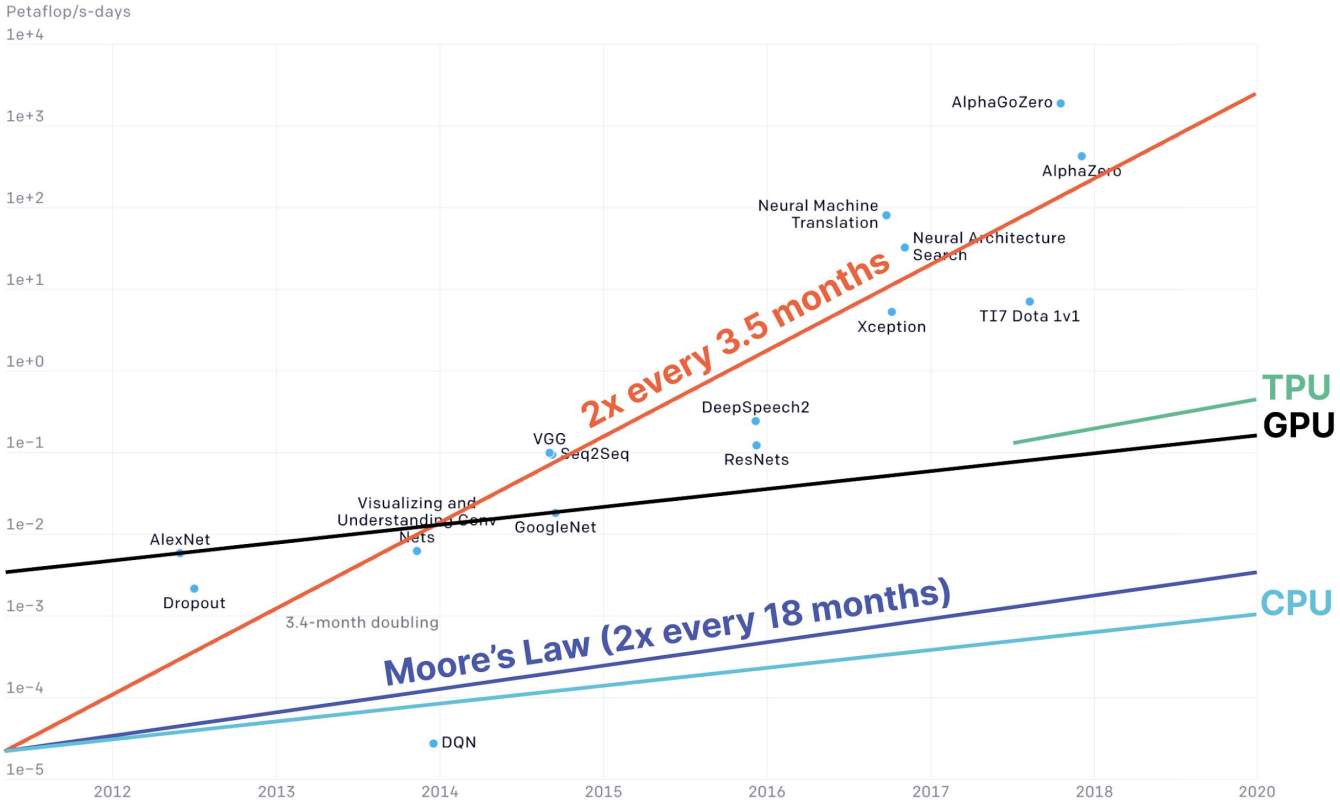
Time	Description
10:00am	Welcome back
10:20am	Distributed model training
12:00pm	Lunch
1:00pm	Observability on Ray

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

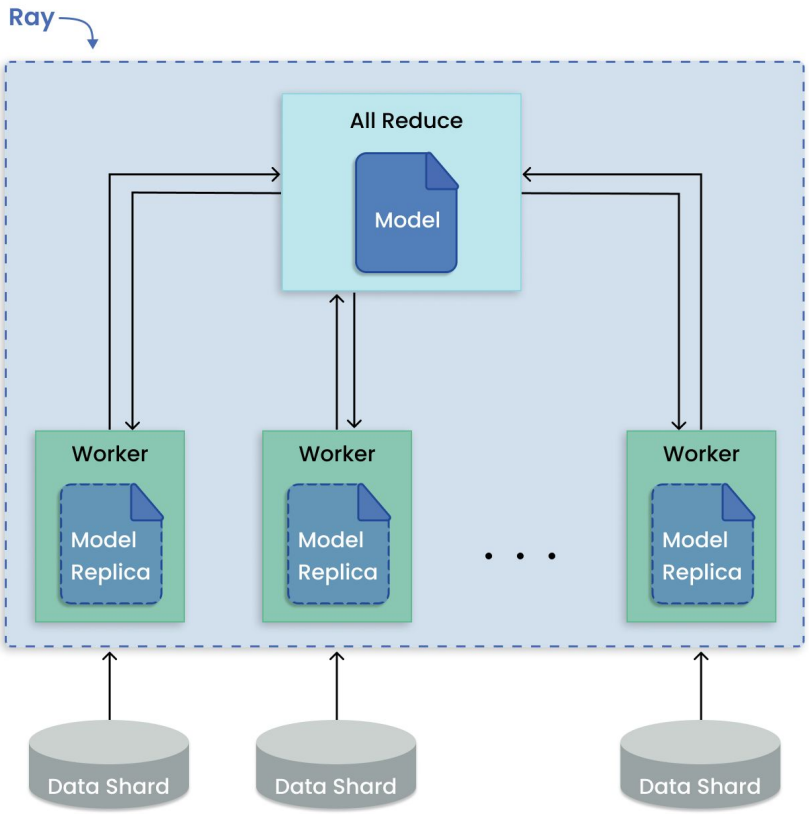
Distributed Model Training

Data parallelism across multiple machines.

Why distribute training?



Data parallelism



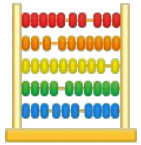
👓 The task.

Semantic segmentation, pixel-by-pixel.

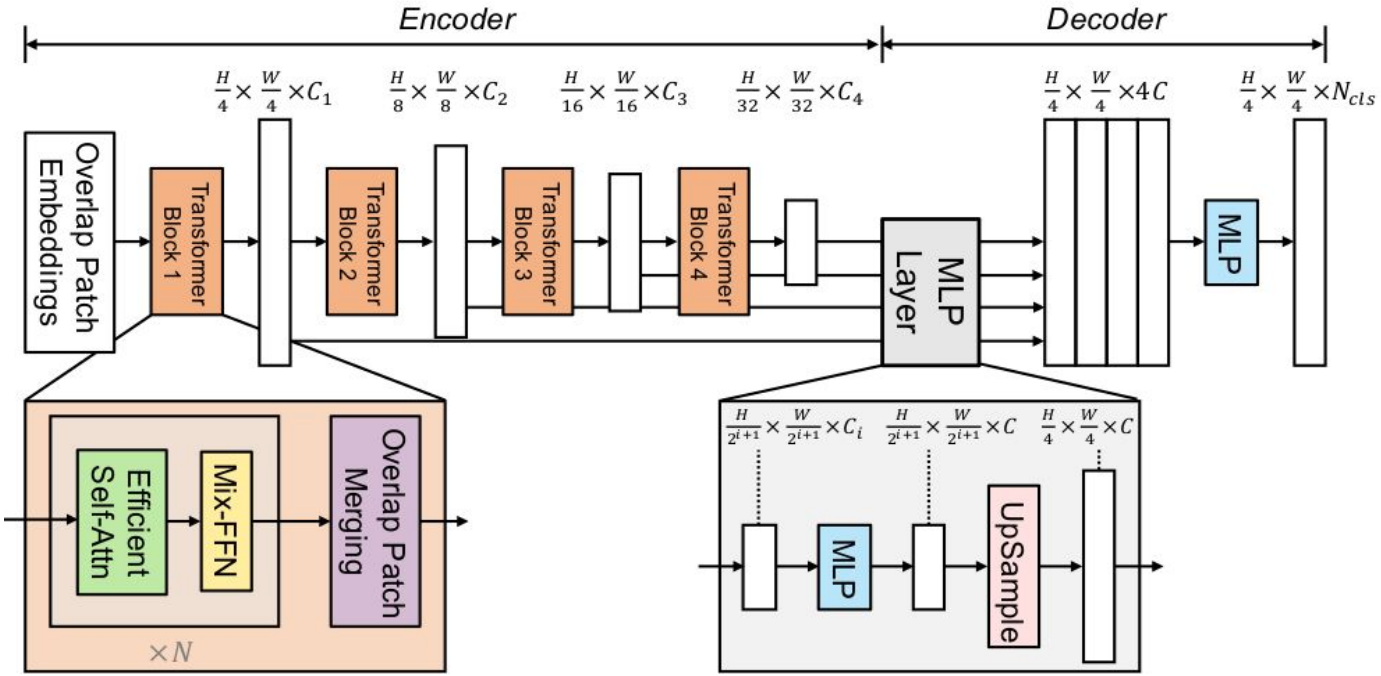
The data.

MIT ADE20K Scene Parsing Benchmark

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

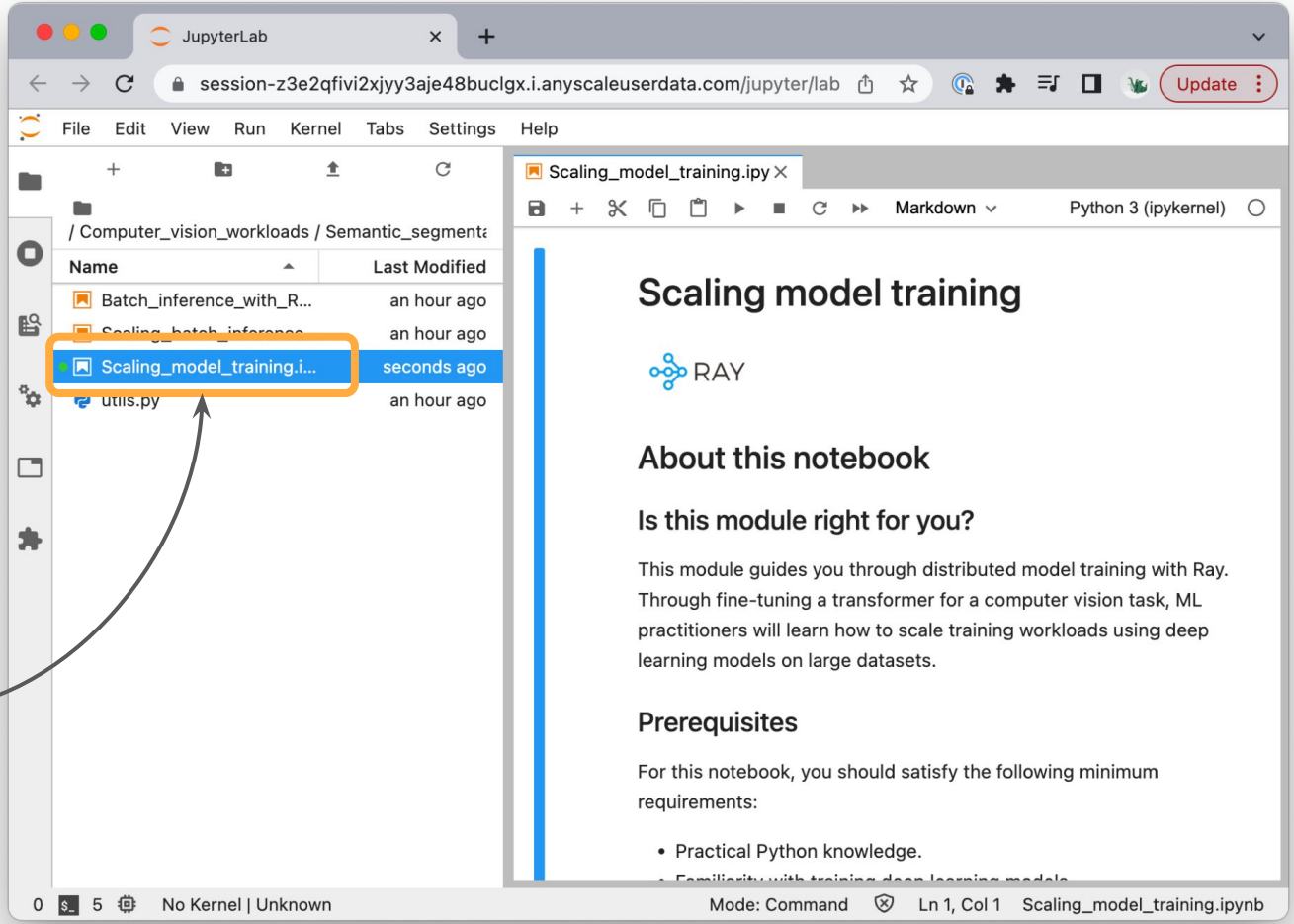


The model.



Segformer, fine-tuned on MIT ADE20K

Meet me here!



Computer_vision_workloads/Semantic_segmentation/Scaling_model...

slido

What is data parallelism?

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

slido

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

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

slido

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.

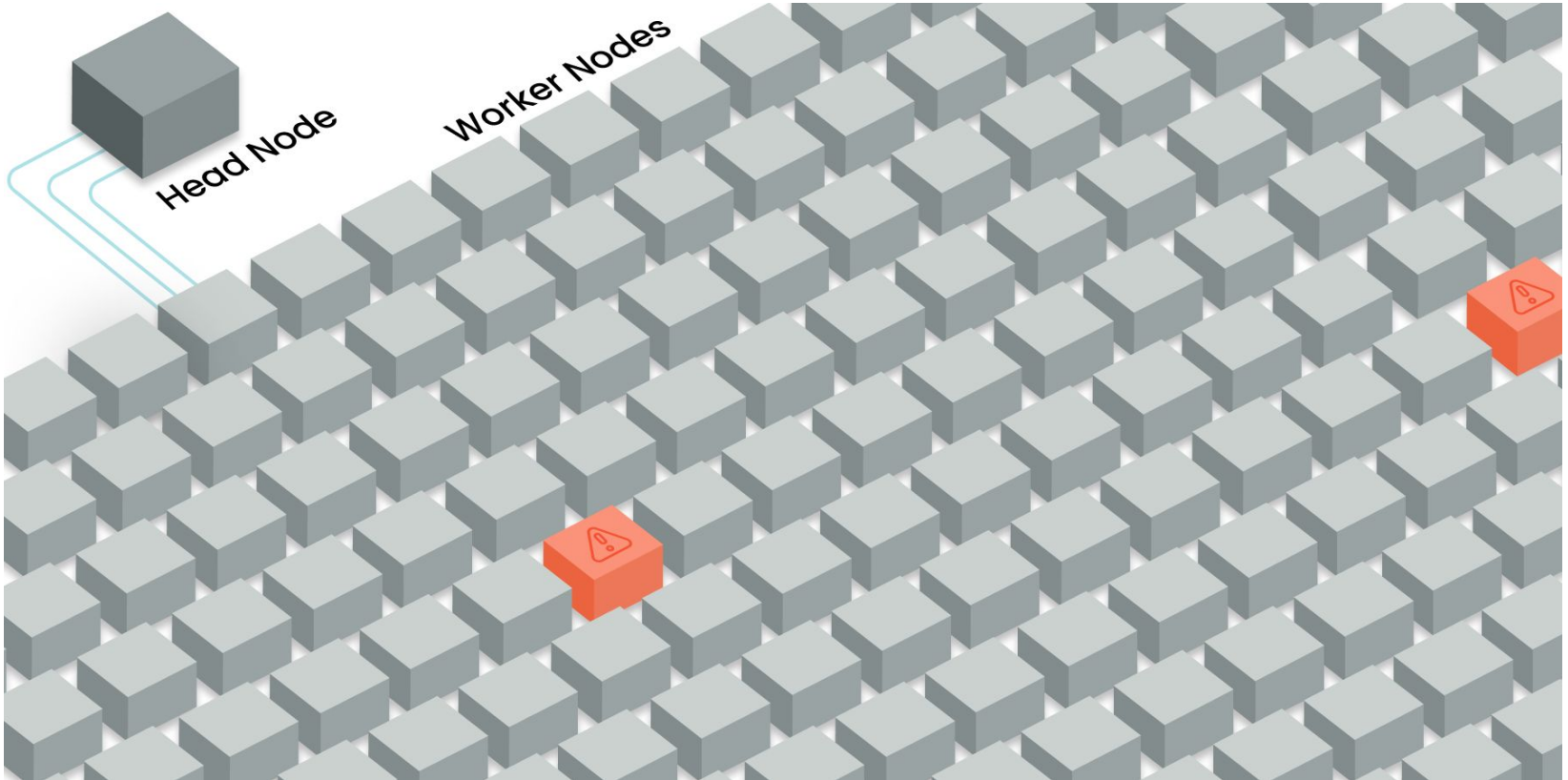
Observability

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.

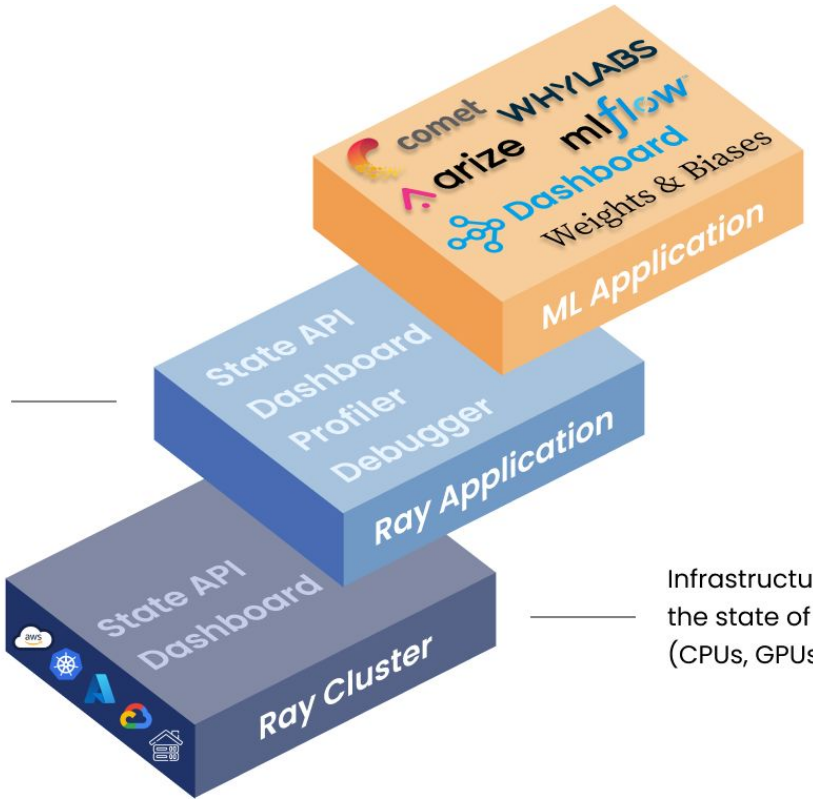
👀 Why is it important?



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.

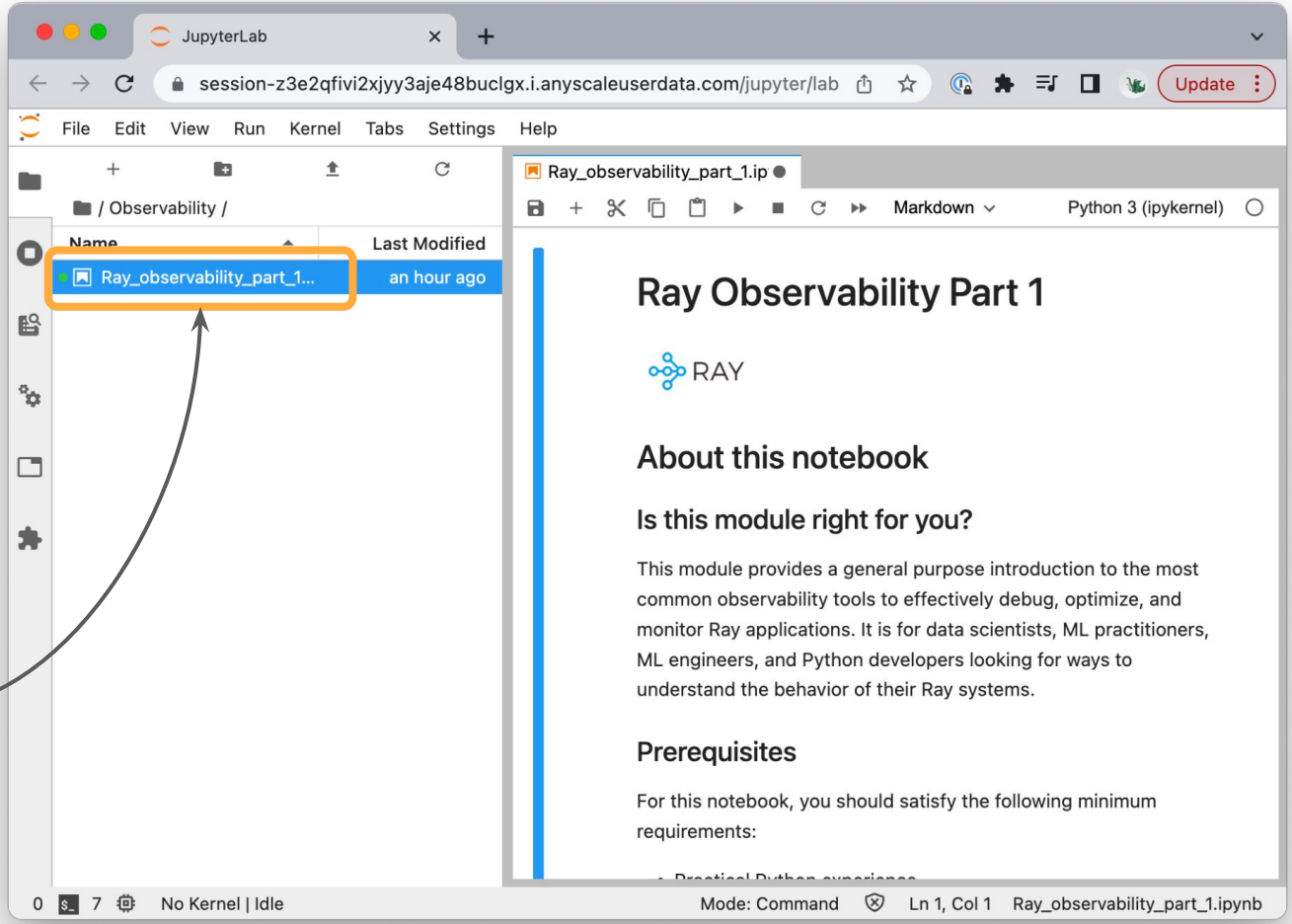


Observe ML applications through third-party integrations for monitoring in interactive development and production.

Debug, optimize, and monitor Ray applications (tasks, actors, objects, etc.).

Infrastructure monitoring on the state of cluster resources (CPUs, GPUs, memory, etc.)

Meet me here!



Observability/Ray_observability_part_1.ipynb

slido

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

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

slido

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

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

slido

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

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

slido

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!

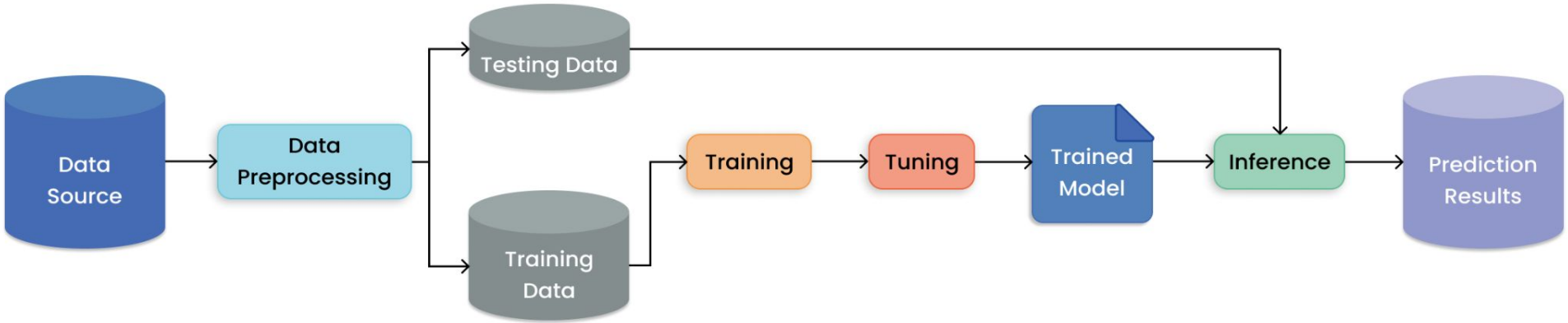
30 minutes.

Batch Inference

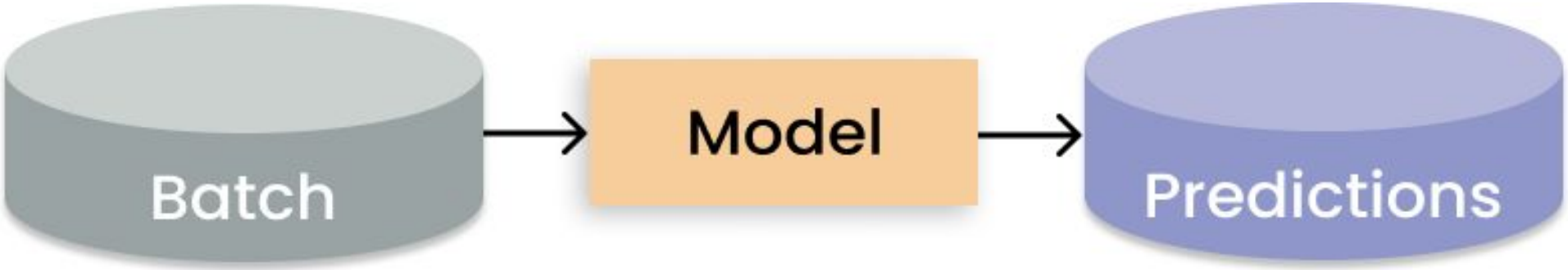
Generating predictions on batches of data.

What is batch inference?

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



🧐 What is batch inference?

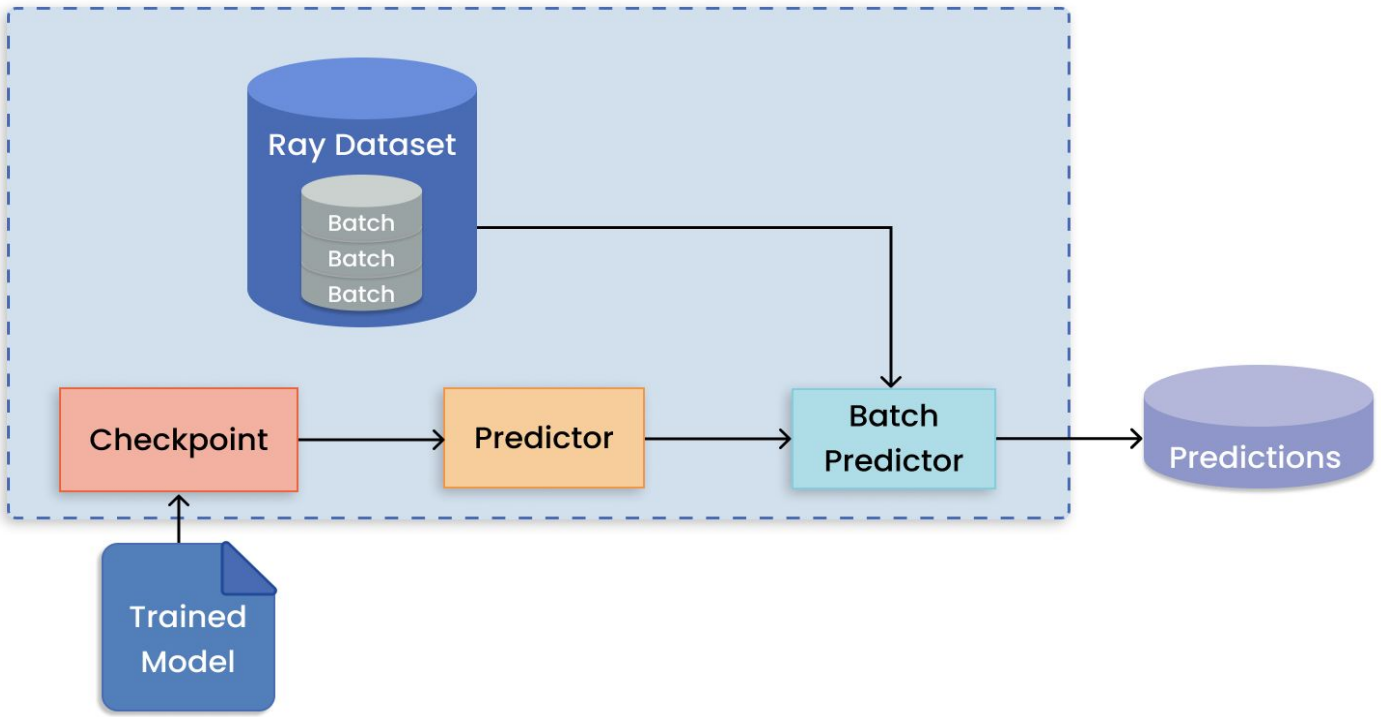


Design patterns for batch inference with Ray

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

Method: Ray AIR

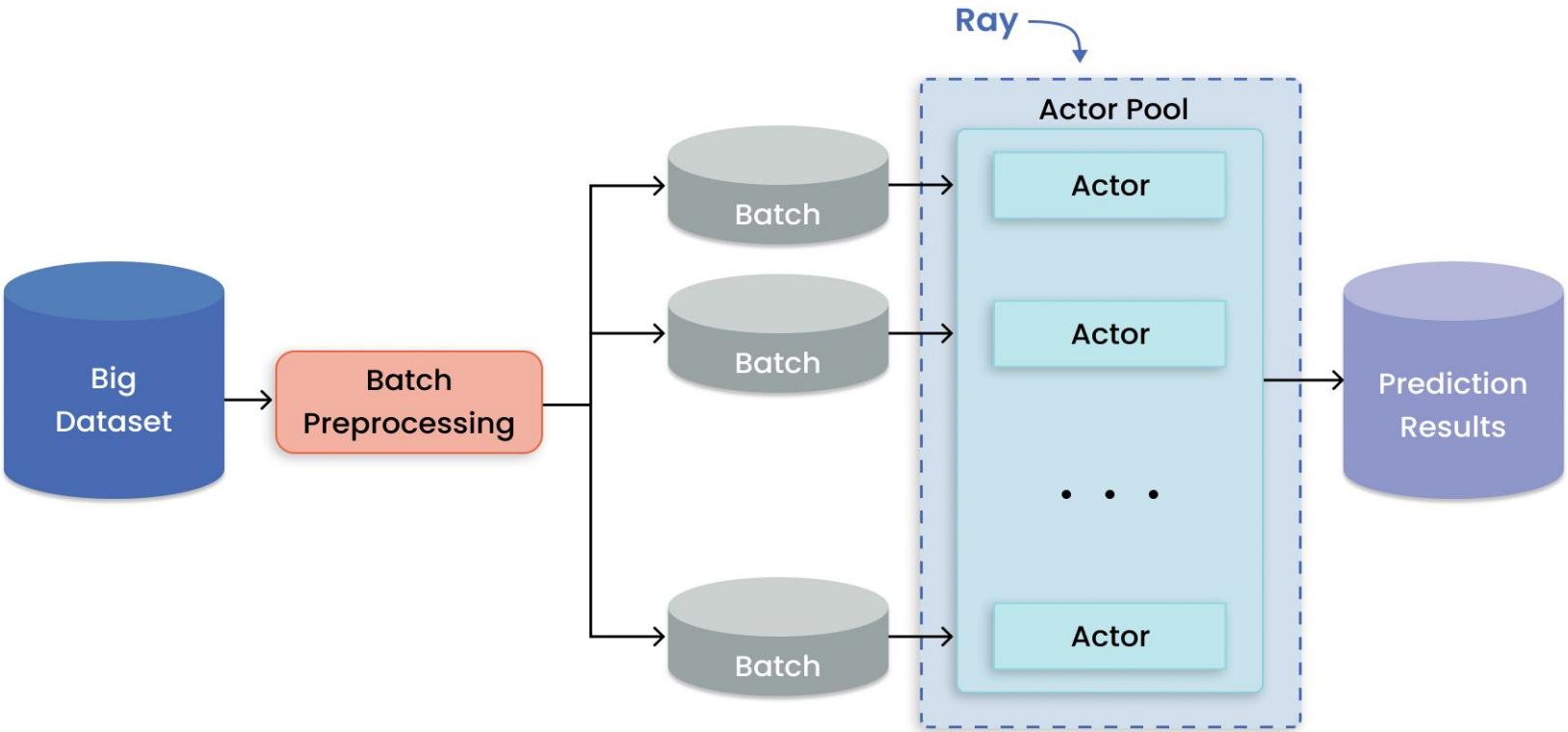
Bat



☁️ Method: Ray AIR BatchPredictor

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

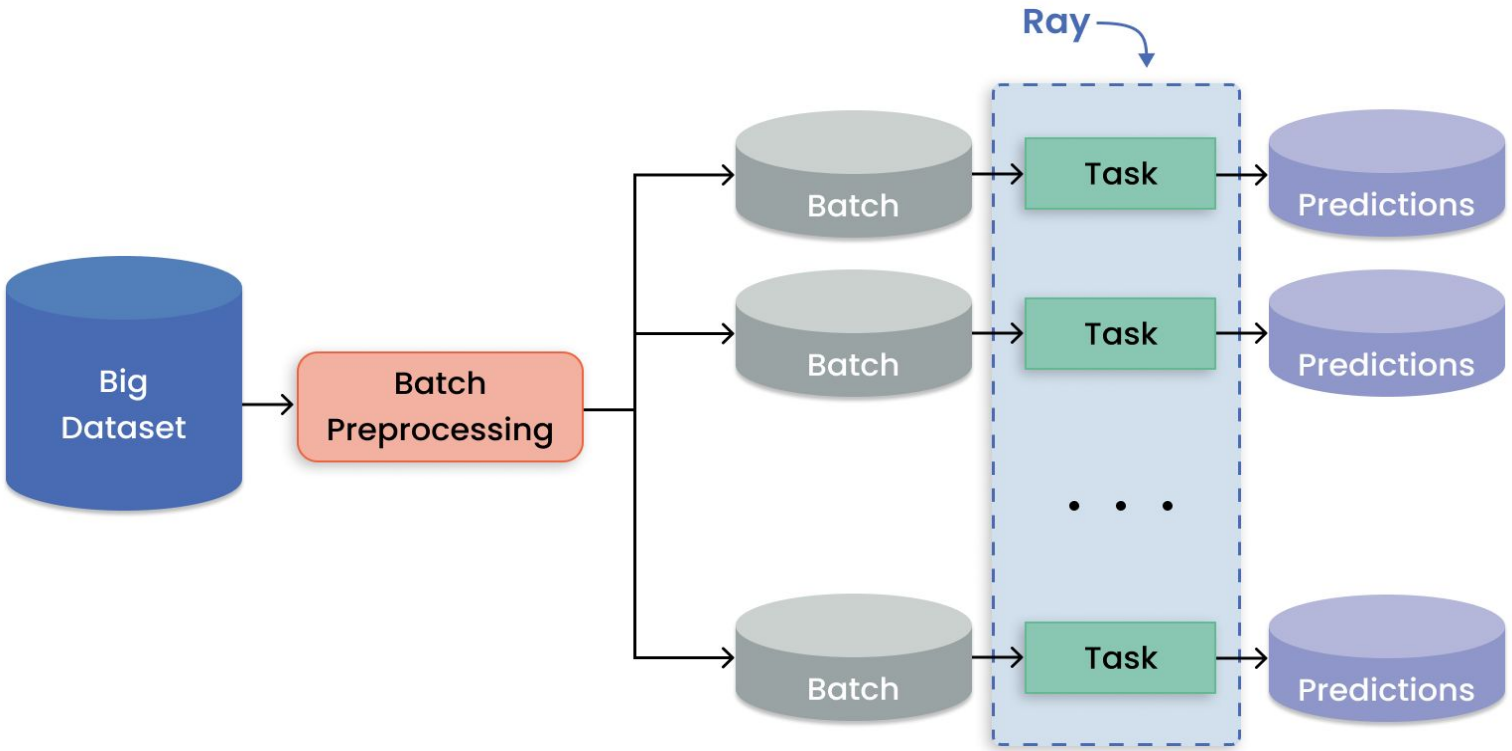
Method: Ray Core Actors



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)]
```

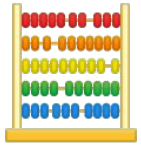
👓 The task.

Semantic segmentation, pixel-by-pixel.

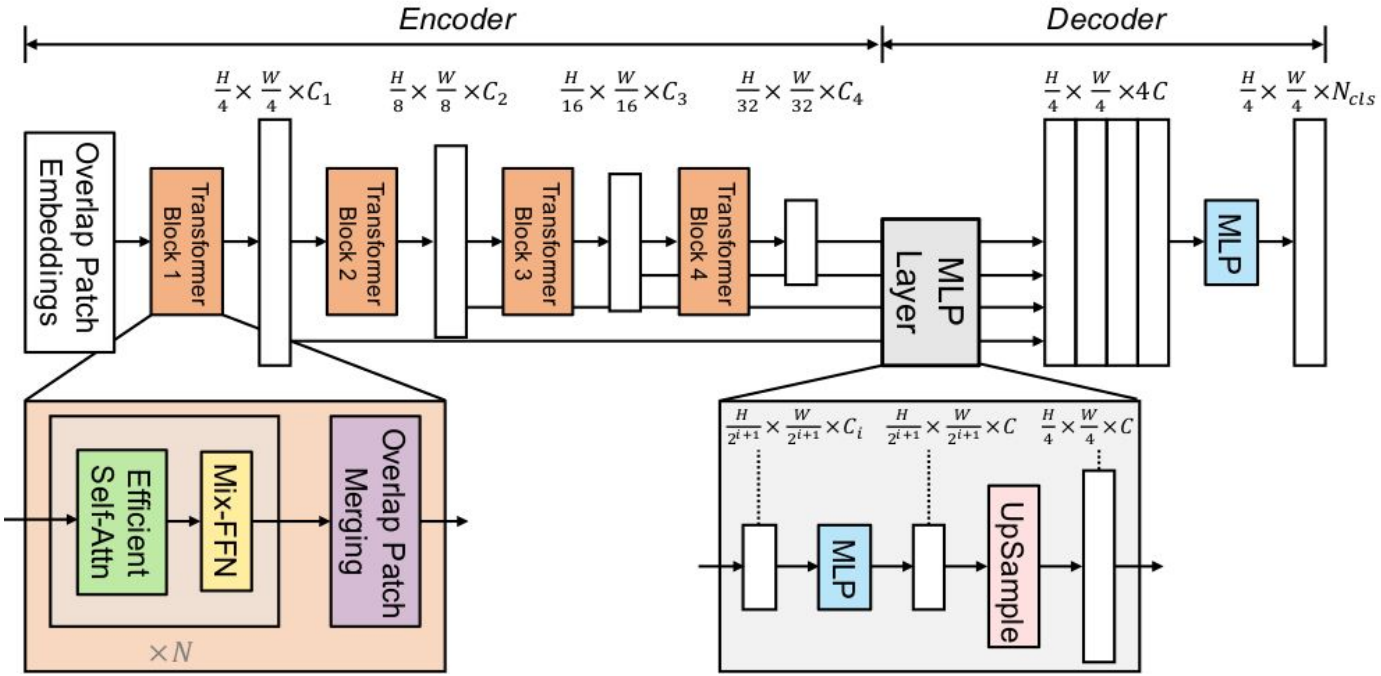
The data.

MIT ADE20K Scene Parsing Benchmark

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

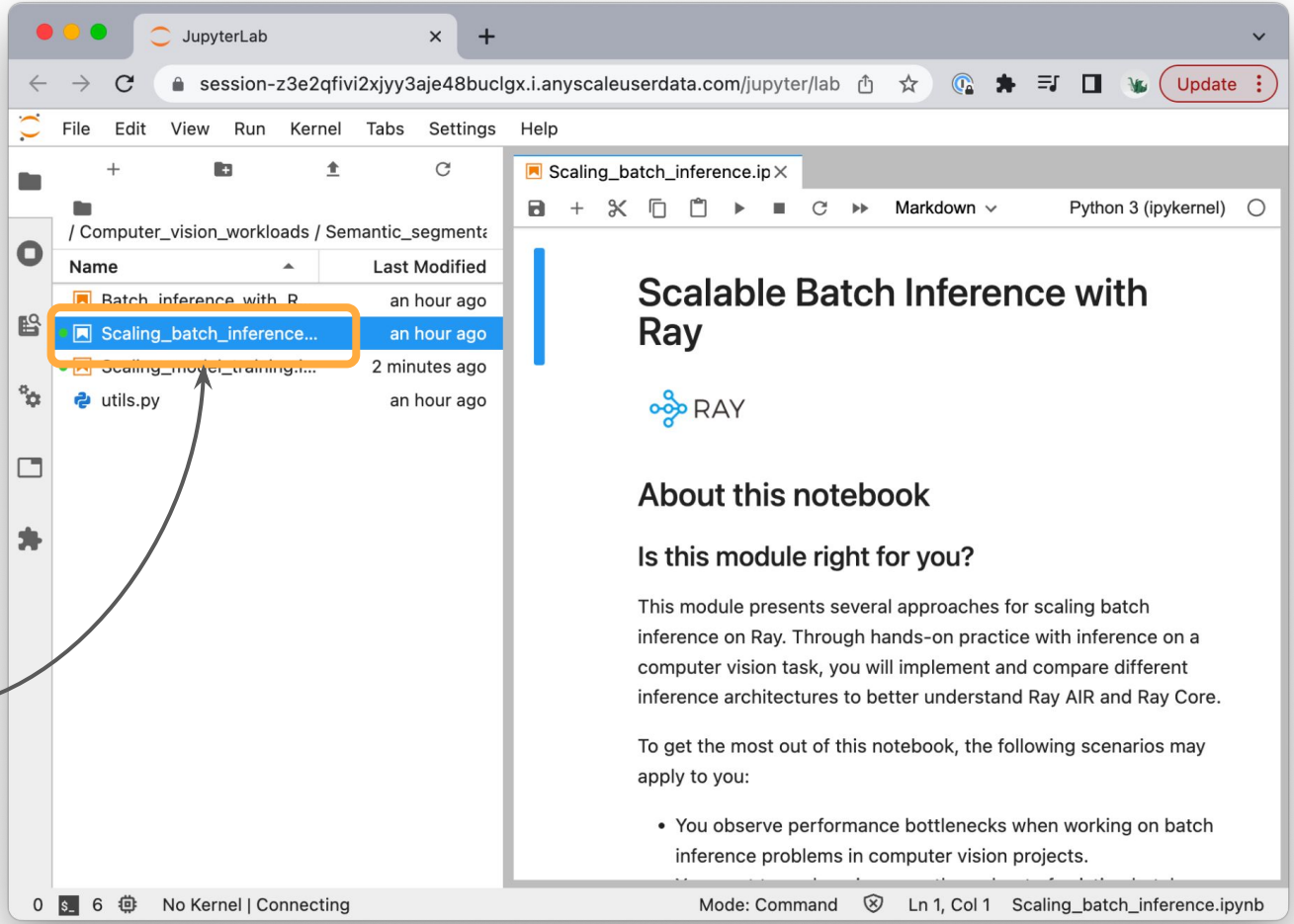


The model.



Segformer, fine-tuned on MIT ADE20K

Meet me here!



Computer_vision_workloads/Semantic_segmentation/Scaling_batch...

slido

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

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

slido

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

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

slido

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

ⓘ 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*

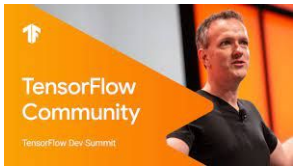
Engagement → **Engaged** communities foster relationships beyond *value*

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

LINUX FOUNDATION

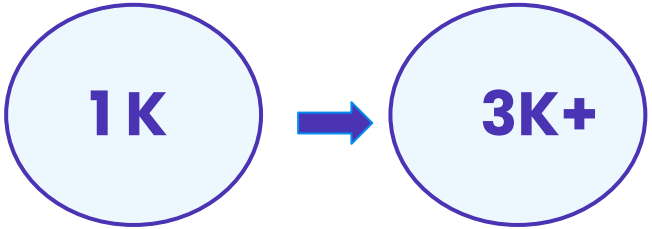
THE APACHE SOFTWARE FOUNDATION

Kubernetes Community Days



Aspired Growth & Activities for Ray Community … 2023

Meetups Membership



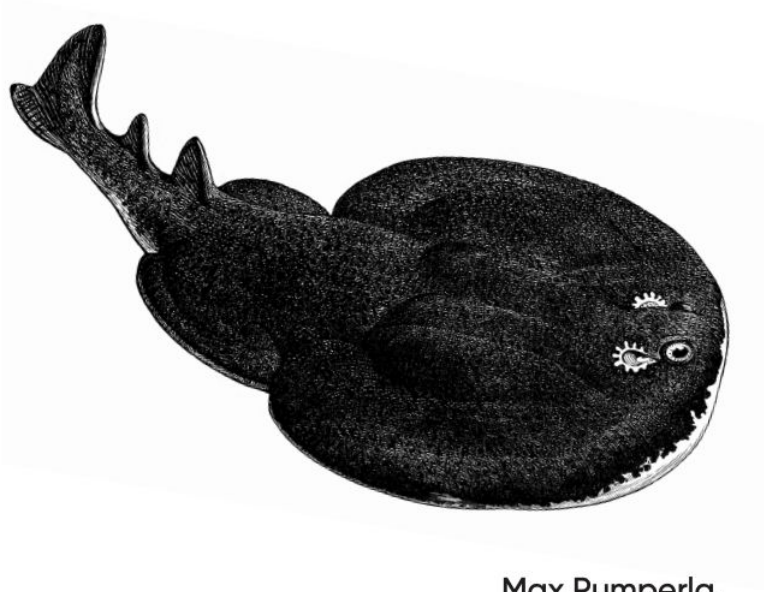
National/Global Ray Meetups

- 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

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.