

# Reinforcement Learning with Ray and RLLib

Dean Wampler

January 28, 2021

[dean.wampler@dominodatalab.com](mailto:dean.wampler@dominodatalab.com)

[@deanwampler](https://twitter.com/deanwampler)

[ray.io](https://ray.io)



# Outline

- Why Ray?
- Why Reinforcement Learning?
- Ray RLlib
- Other Uses of Ray
- Next steps



# why Ray?



@deanwampler

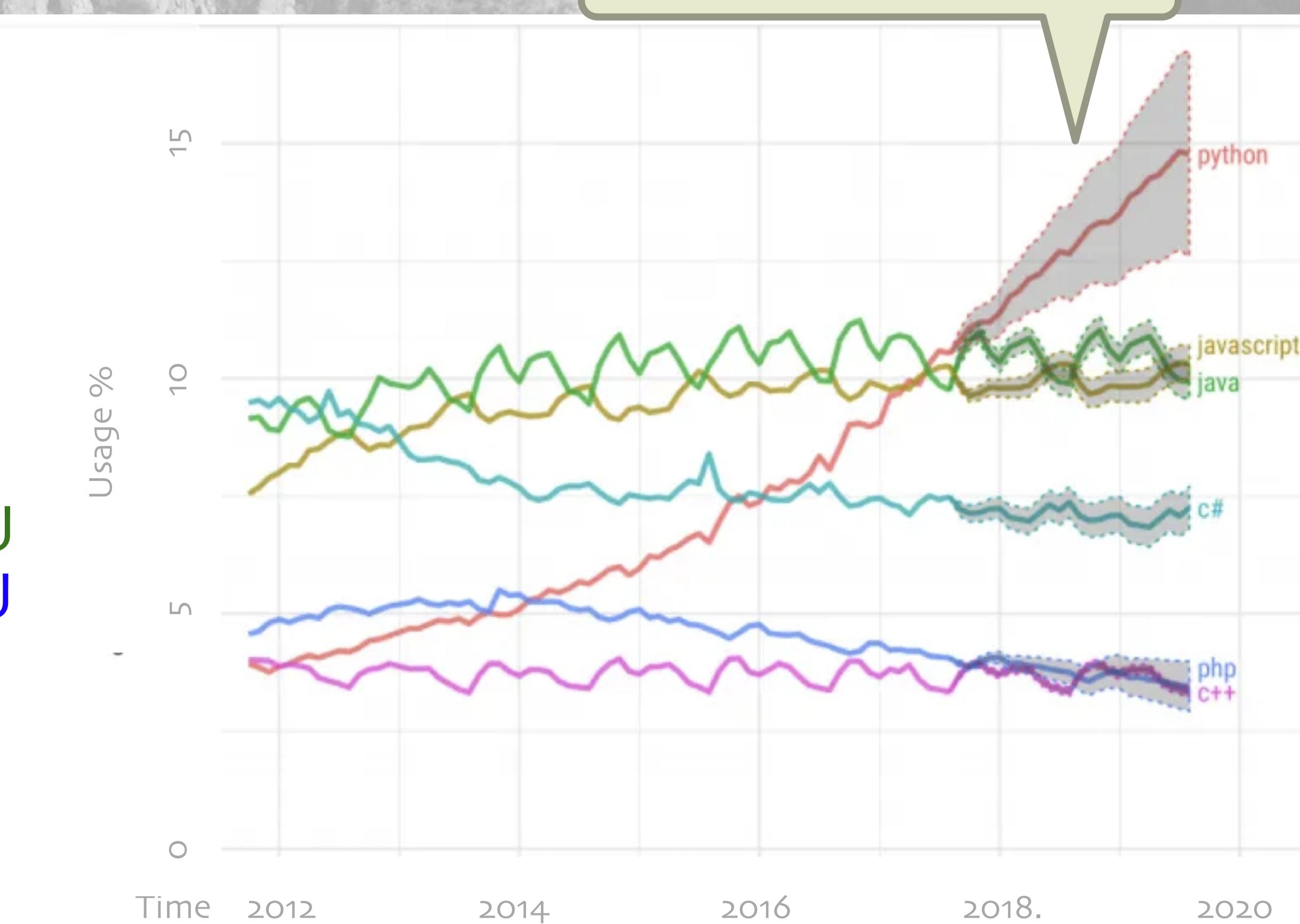
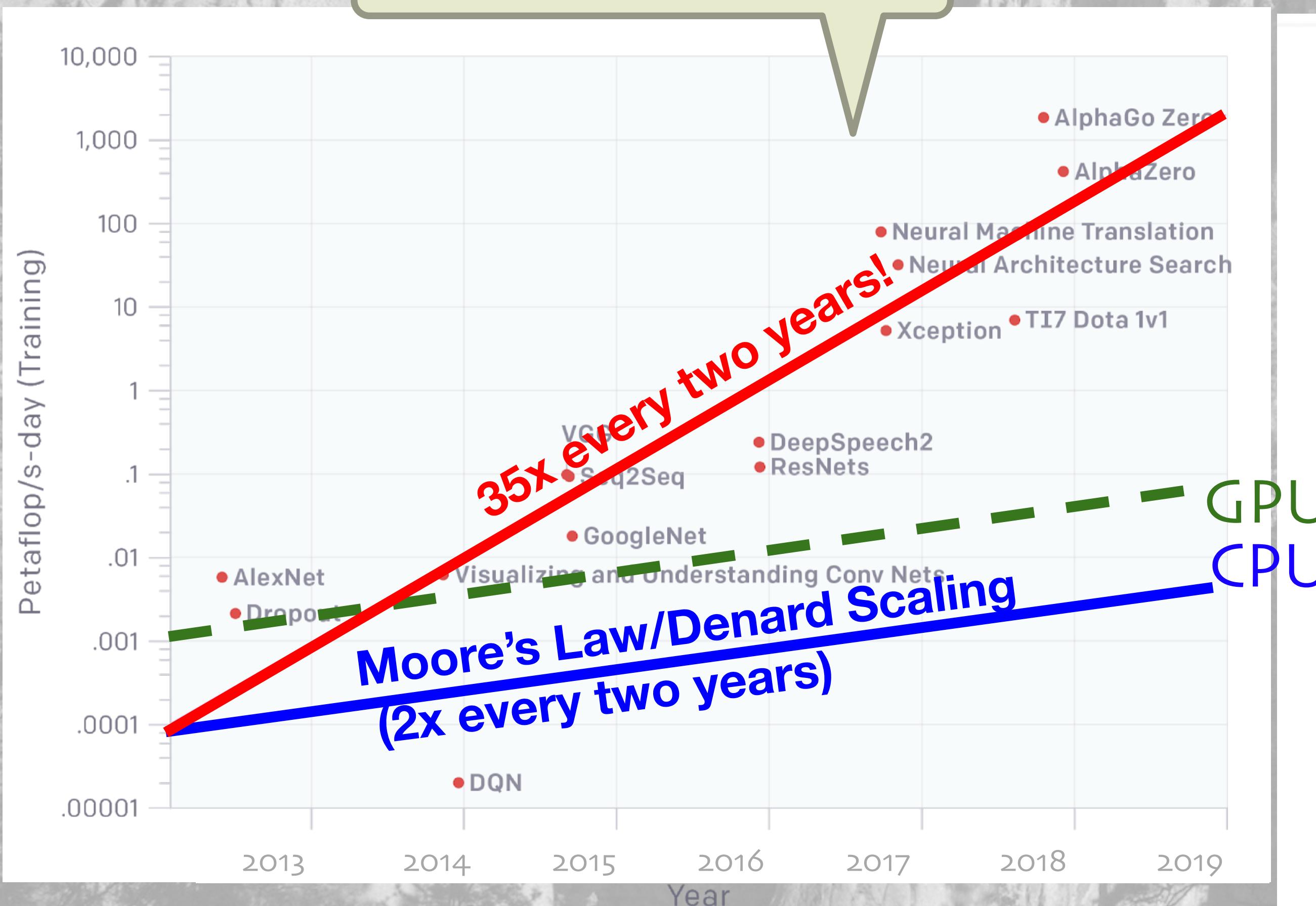


# Two Major Trends

Model sizes and therefore compute requirements outstripping Moore's Law

Hence, there is a pressing need for robust, easy to use solutions for distributed Python

Python growth driven by ML/AI and other data science workloads



# The ML Landscape Today

All require distributed implementations to scale

Featurization



Streaming



Hyperparam Tuning



Training



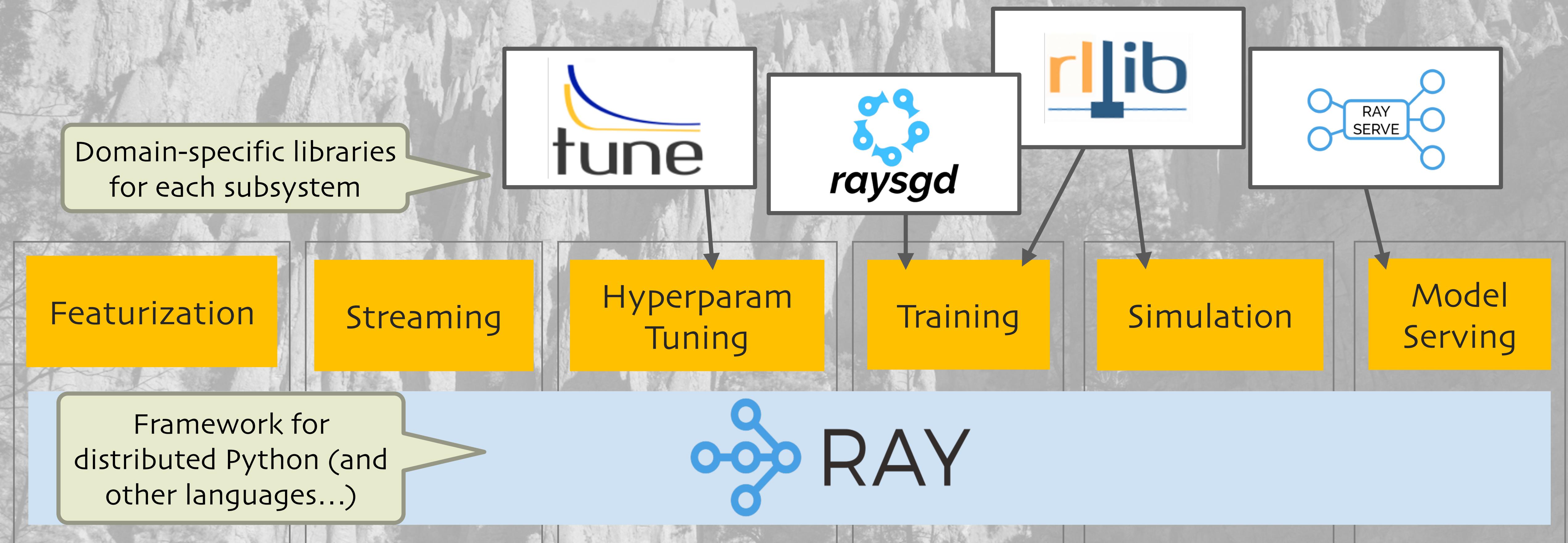
Simulation



Model Serving



# The Ray Vision: Sharing a Common Framework



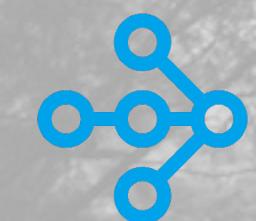
# API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a
```

```
def add_arrays(a, b):  
    return np.add(a, b)
```

The Python you  
already know...



# API - Designed to Be Intuitive and Concise

Functions -> Tasks

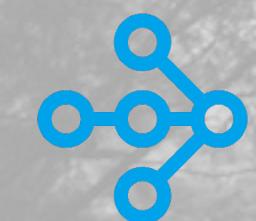
```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a
```

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

For completeness, add these first:

```
import ray  
import numpy as np  
ray.init()
```

Now these functions  
are remote "tasks"



# API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a
```

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

```
ref1 = make_array.remote(...)
```

make\_array

ref1



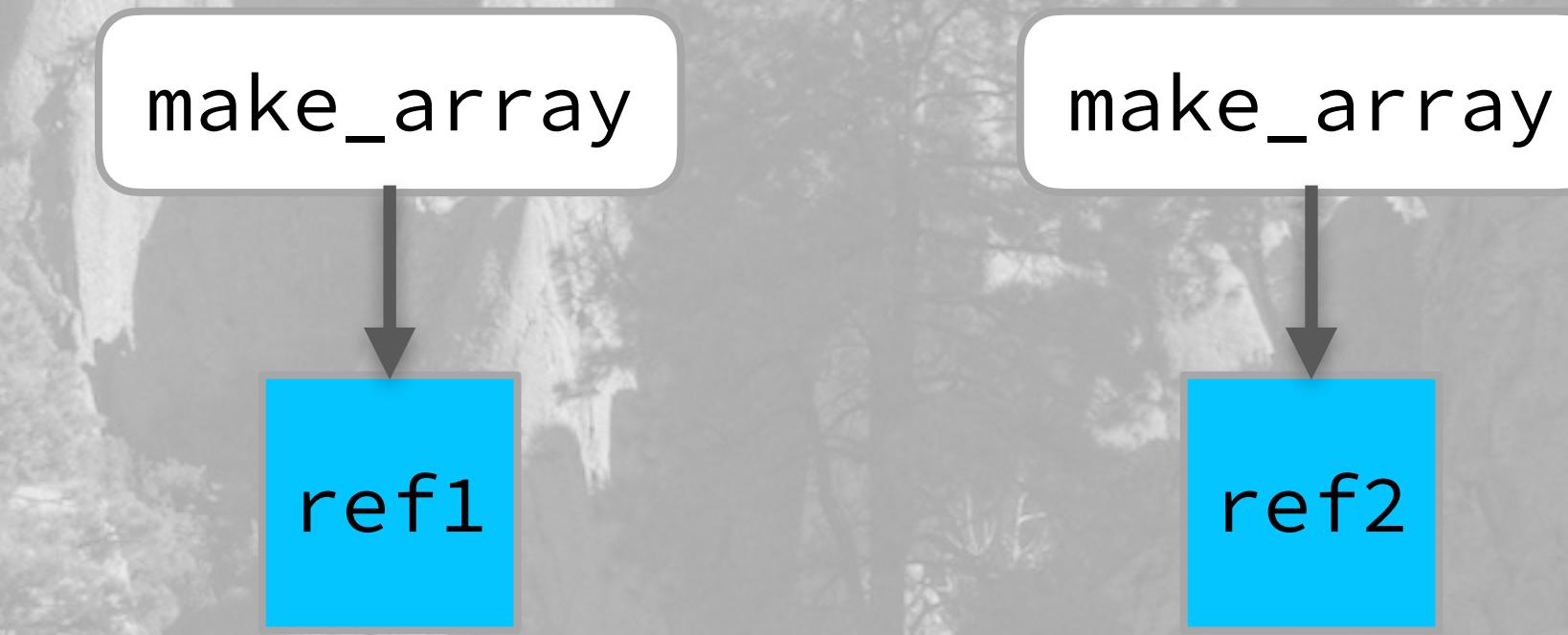
# API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a
```

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

```
ref1 = make_array.remote(...)  
ref2 = make_array.remote(...)
```



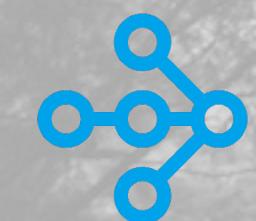
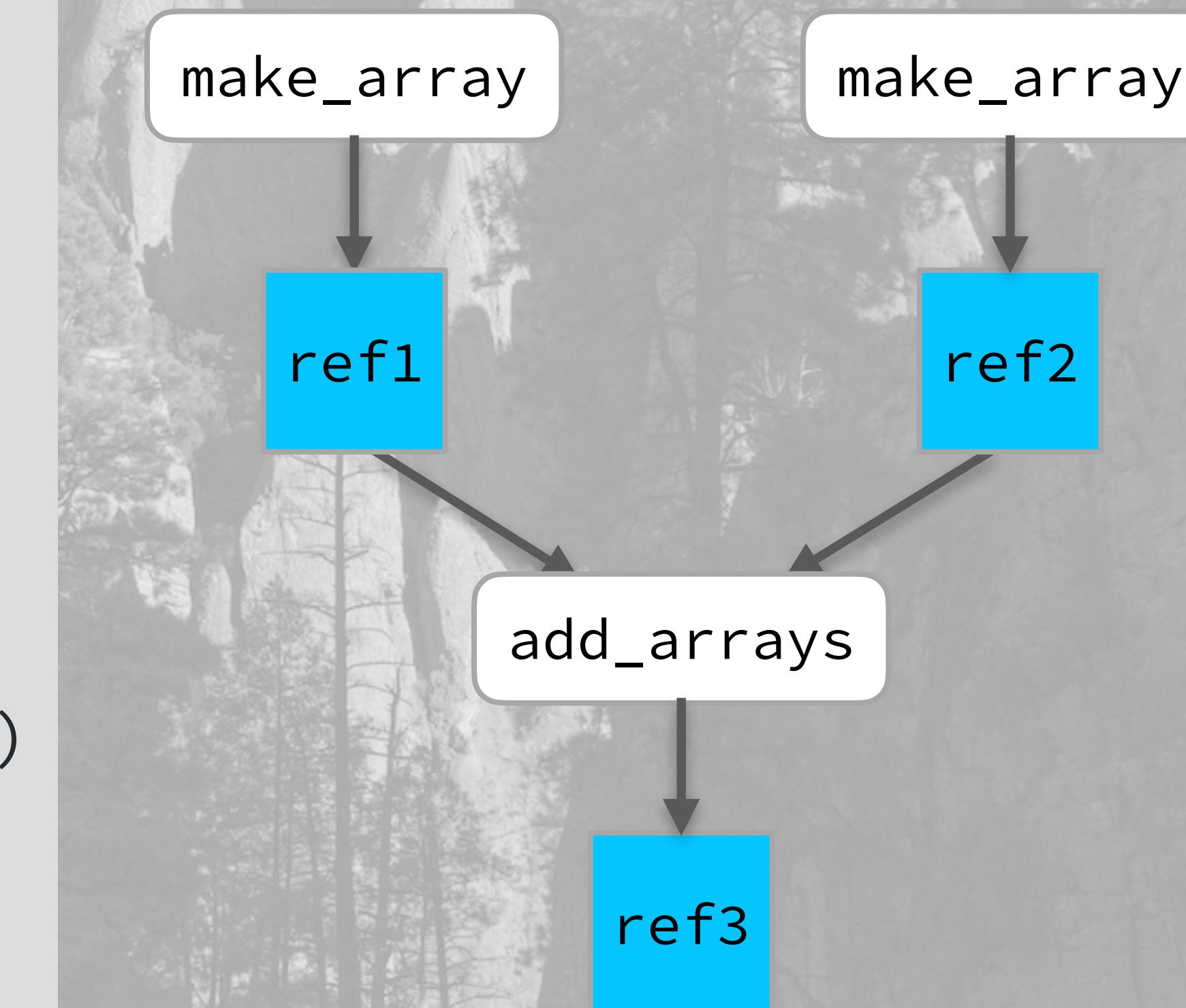
# API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote
def make_array(...):
    a = ... # Construct a NumPy array
    return a

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

ref1 = make_array.remote(...)
ref2 = make_array.remote(...)
ref3 = add_arrays.remote(ref1, ref2)
```



# API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a
```

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

```
ref1 = make_array.remote(...)  
ref2 = make_array.remote(...)  
ref3 = add_arrays.remote(ref1, ref2)  
ray.get(ref3)
```

Ray handles sequencing  
of async dependencies

Ray handles extracting the  
arrays from the object refs

make\_array

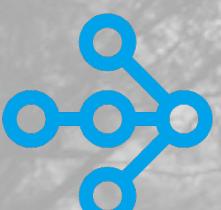
ref1

make\_array

ref2

add\_arrays

ref3



What about  
distributed  
state?

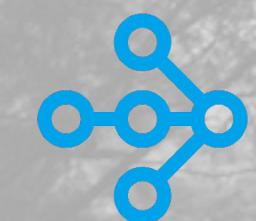
# API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote
def make_array(...):
    a = ... # Construct a NumPy array
    return a

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

ref1 = make_array.remote(...)
ref2 = make_array.remote(...)
ref3 = add_arrays.remote(ref1, ref2)
ray.get(ref3)
```



# API - Designed to Be Intuitive and Concise

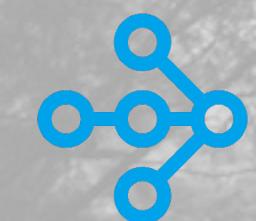
Functions -> Tasks

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a  
  
@ray.remote  
def add_arrays(a, b):  
    return np.add(a, b)  
  
ref1 = make_array.remote(...)  
ref2 = make_array.remote(...)  
ref3 = add_arrays.remote(ref1, ref2)  
ray.get(ref3)
```

Classes -> Actors

```
class Counter(object):  
    def __init__(self):  
        self.value = 0  
    def increment(self):  
        self.value += 1  
    return self.value
```

The Python  
classes you  
love...



# API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote  
def make_array(...):  
    a = ... # Construct a NumPy array  
    return a  
  
@ray.remote  
def add_arrays(a, b):  
    return np.add(a, b)  
  
ref1 = make_array.remote(...)  
ref2 = make_array.remote(...)  
ref3 = add_arrays.remote(ref1, ref2)  
ray.get(ref3)
```

Classes -> Actors

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

... now a remote  
“actor”

You need a  
“getter” method  
to read the state.



# API - Designed to Be Intuitive and Concise

Functions -> Tasks

```
@ray.remote
def make_array(...):
    a = ... # Construct a NumPy array
    return a

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

ref1 = make_array.remote(...)
ref2 = make_array.remote(...)
ref3 = add_arrays.remote(ref1, ref2)
ray.get(ref3)
```

Classes -> Actors

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

c = Counter.remote()
ref4 = c.increment.remote()
ref5 = c.increment.remote()
ray.get([ref4, ref5]) # [1, 2]
```





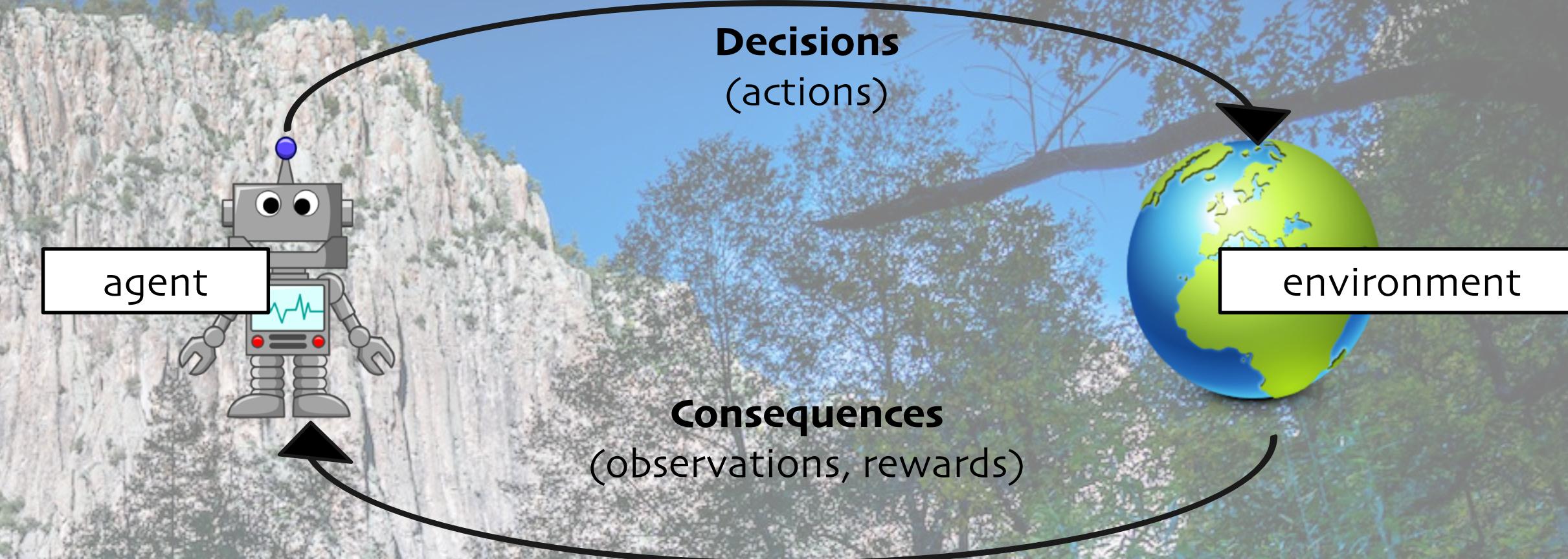
# Why Reinforcement Learning?



@deanwampler



# Reinforcement Learning



Games

Robotics,  
Autonomous  
Vehicles

Industrial  
Processes

System  
Optimization

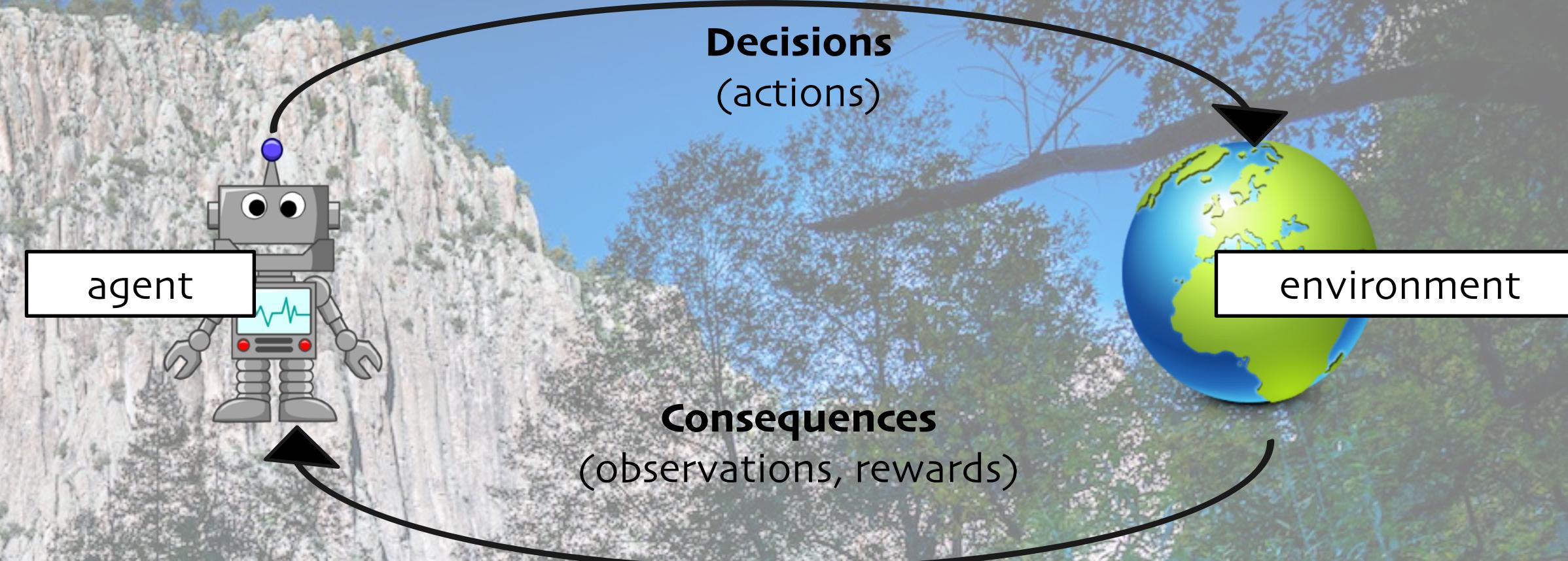
Advertising,  
Recommendations

Finance

RL applications



# Reinforcement Learning



Games

Robotics,  
Autonomous  
Vehicles

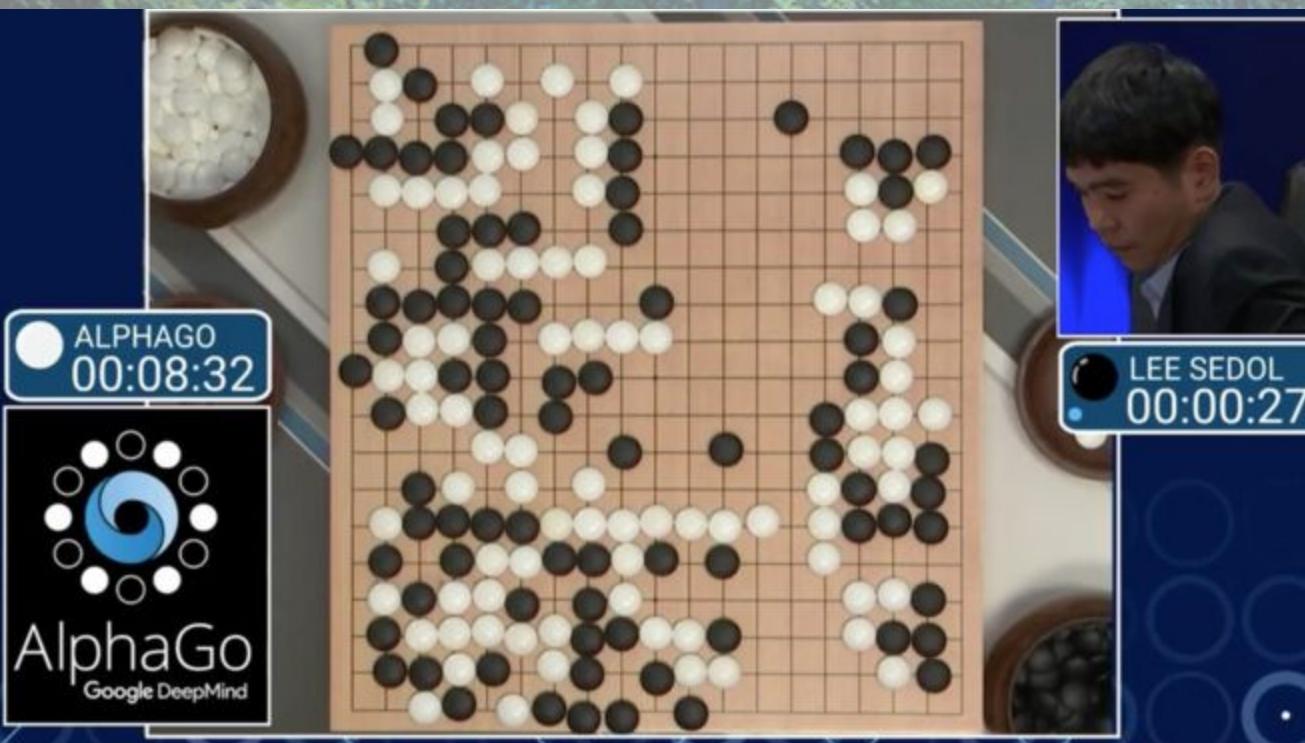
Industrial  
Processes

System  
Optimization

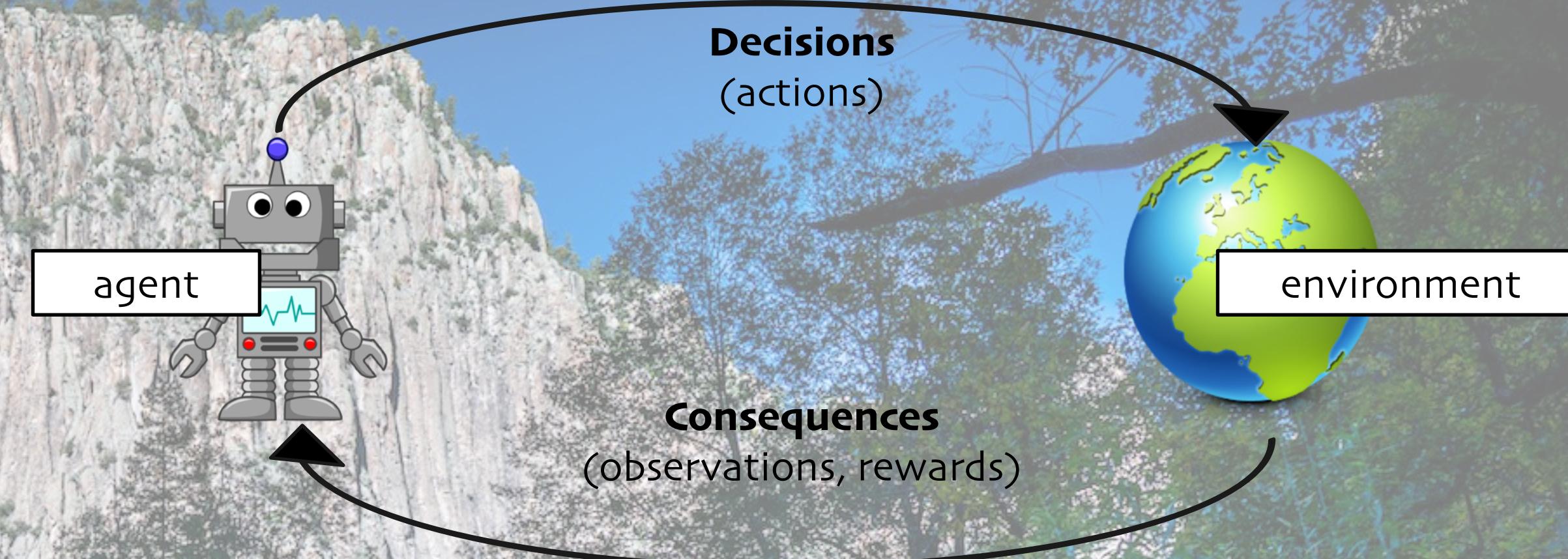
Advertising,  
Recommendations

Finance

RL applications



# Reinforcement Learning



Games

Robotics,  
Autonomous  
Vehicles

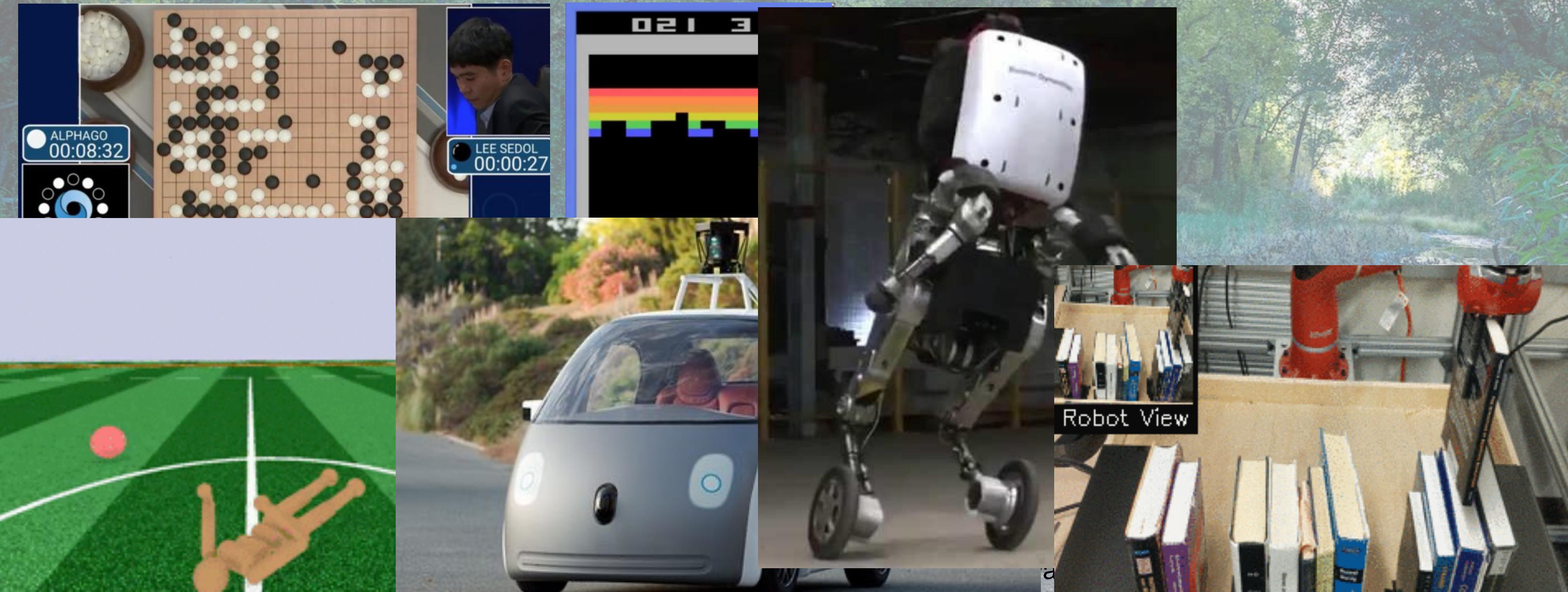
Industrial  
Processes

System  
Optimization

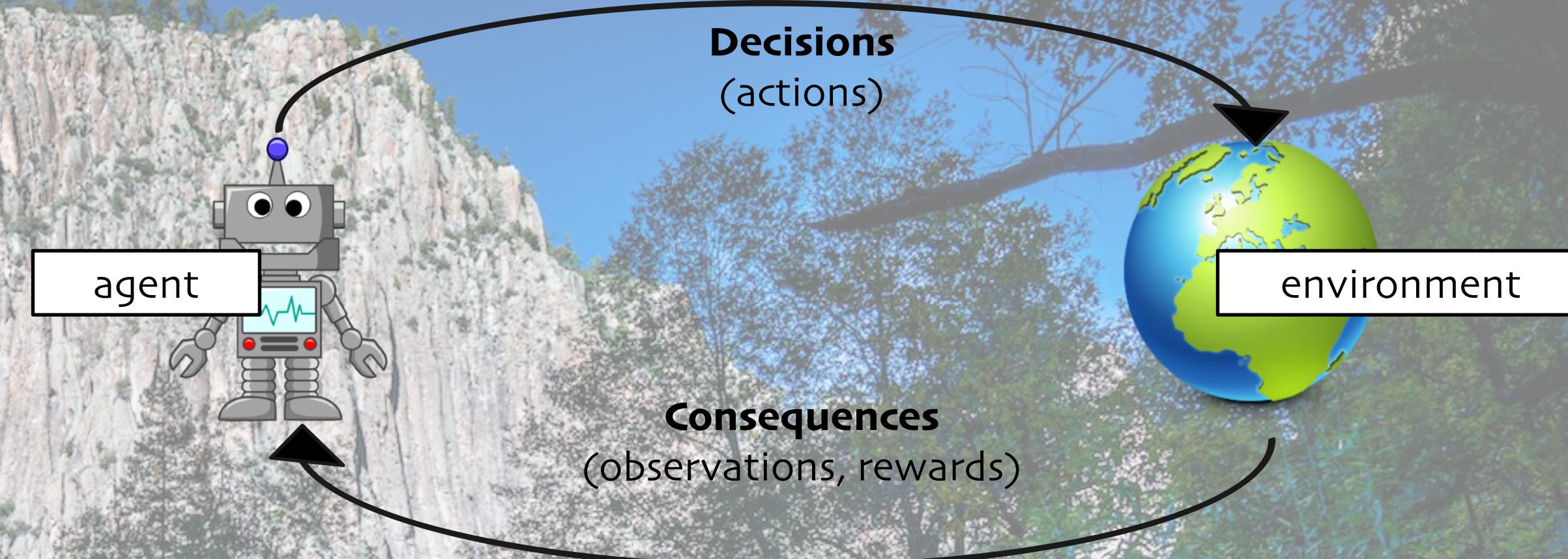
Advertising,  
Recommendations

Finance

RL applications



# Reinforcement Learning



Games

Robotics,  
Autonomous  
Vehicles

Industrial  
Processes

System  
Optimization

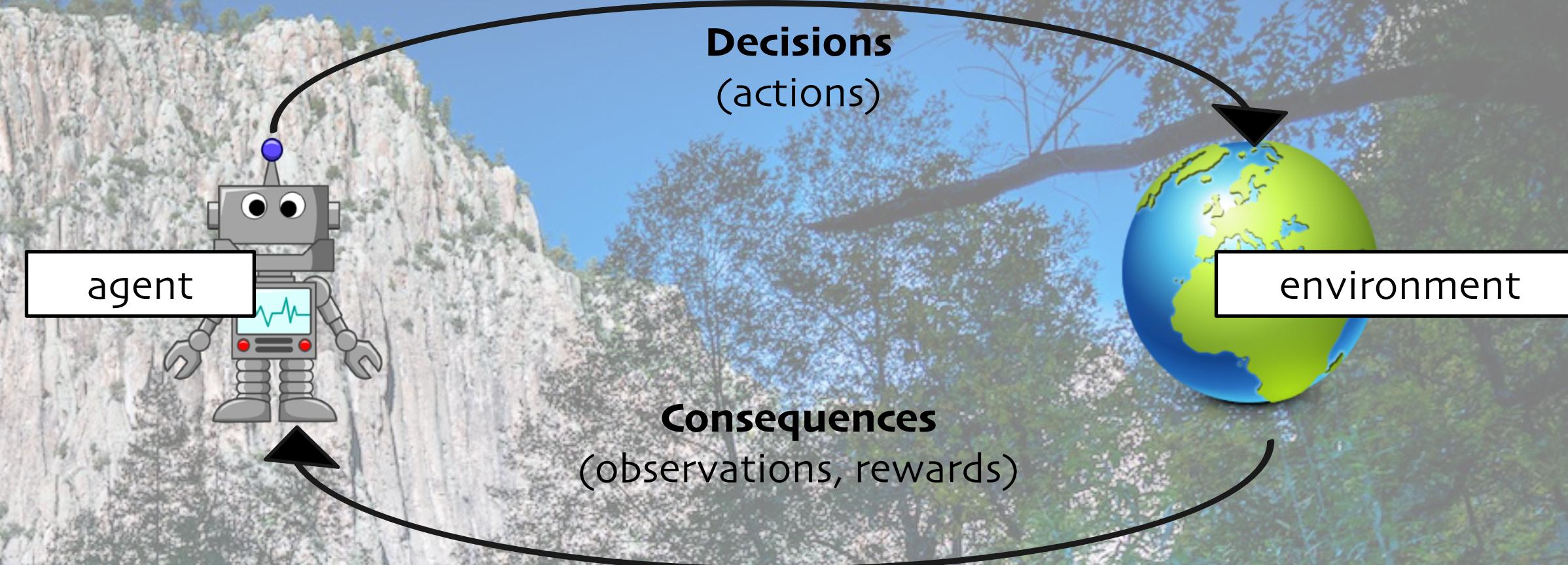
Advertising,  
Recommendations

Finance

RL applications



# Reinforcement Learning



Games

Robotics,  
Autonomous  
Vehicles

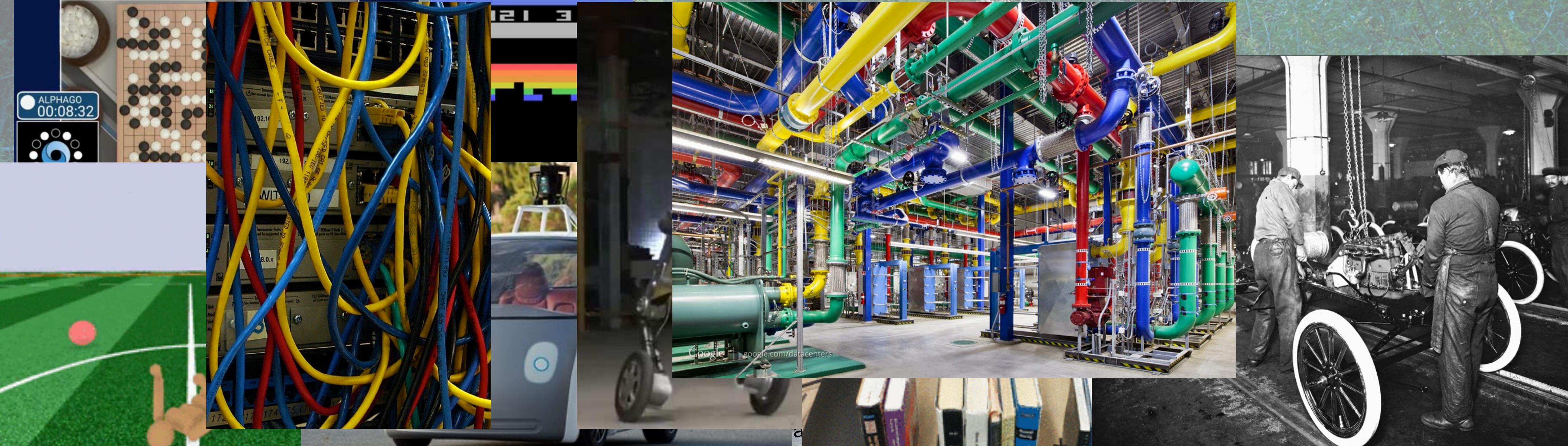
Industrial  
Processes

System  
Optimization

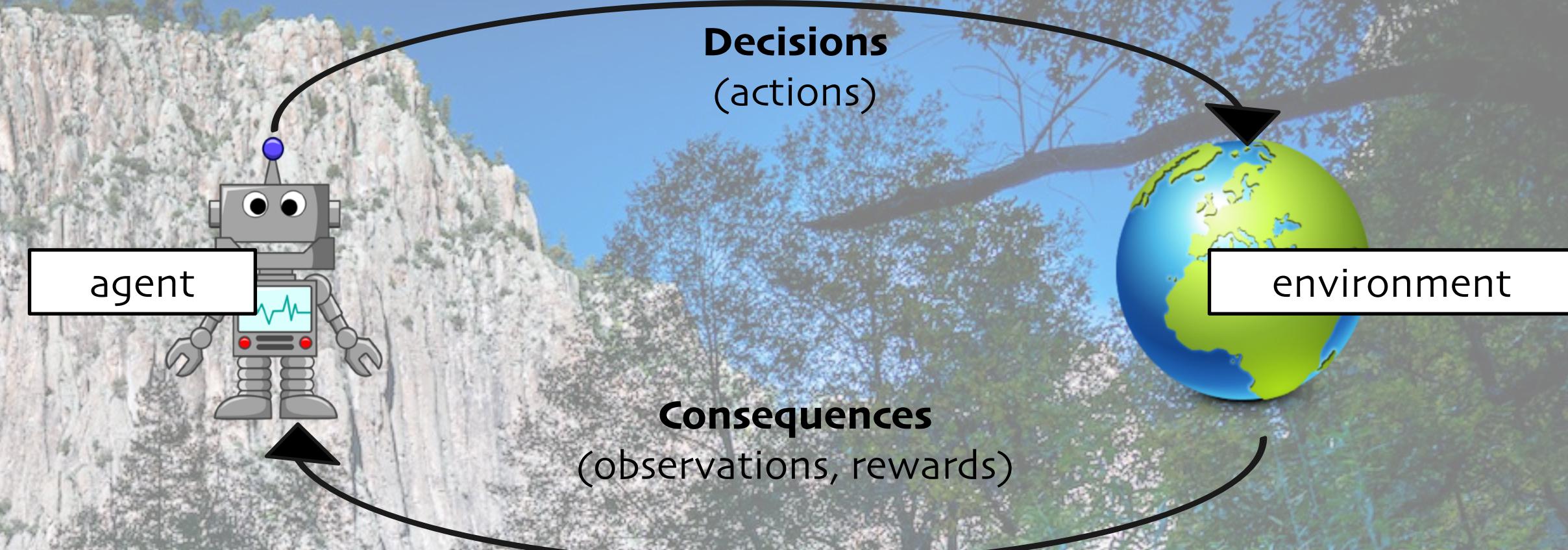
Advertising,  
Recommendations

Finance

RL applications



# Reinforcement Learning



Games

Robotics,  
Autonomous  
Vehicles

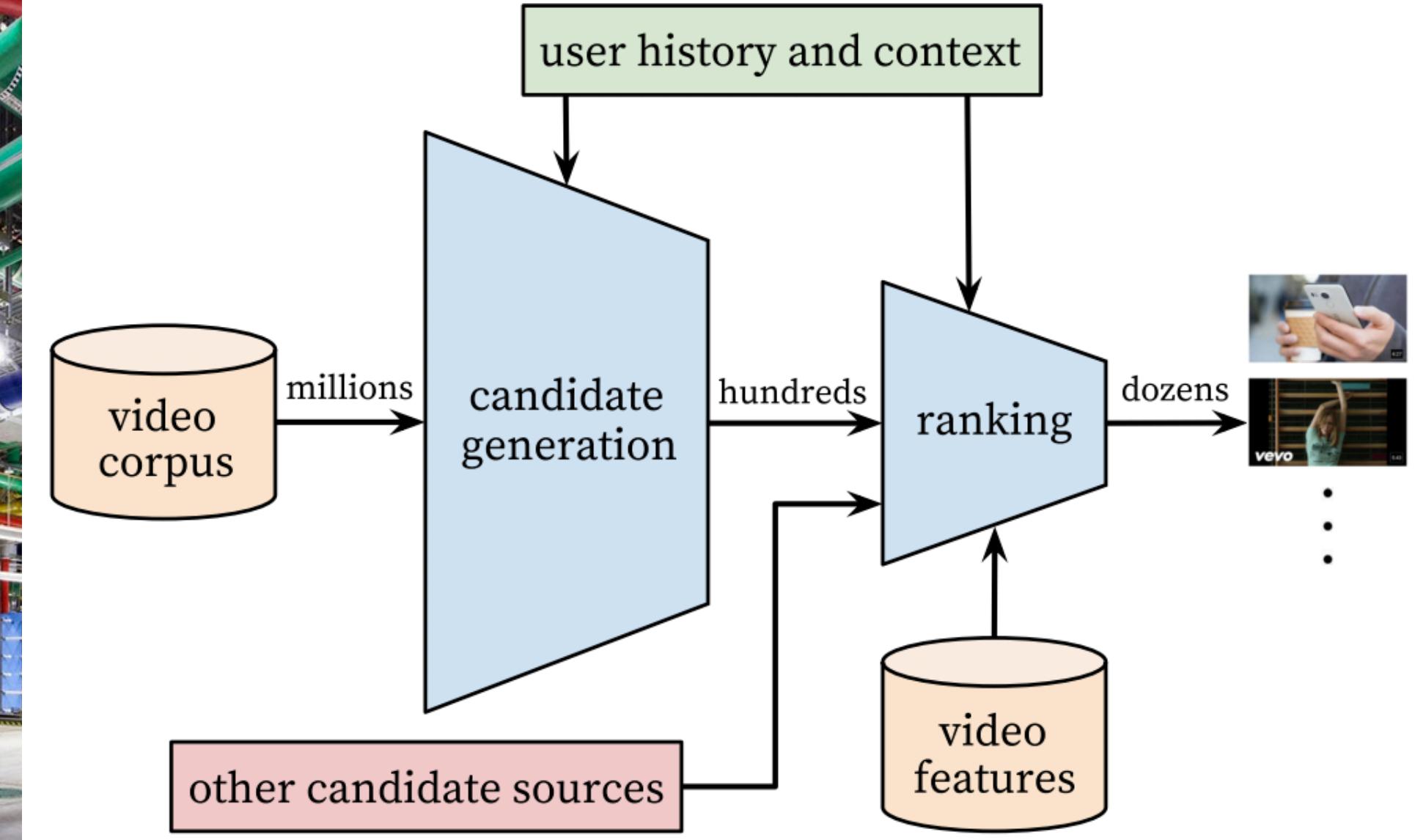
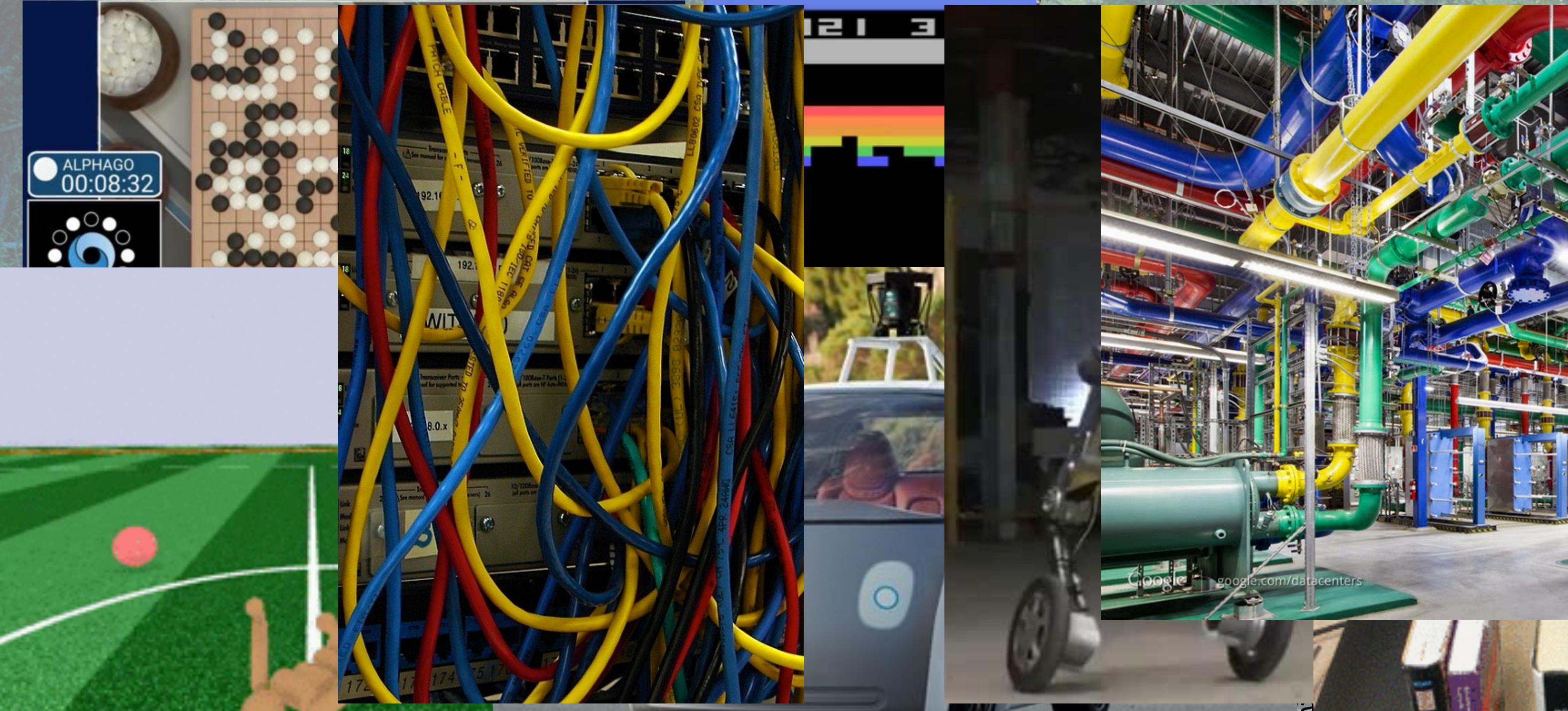
Industrial  
Processes

System  
Optimization

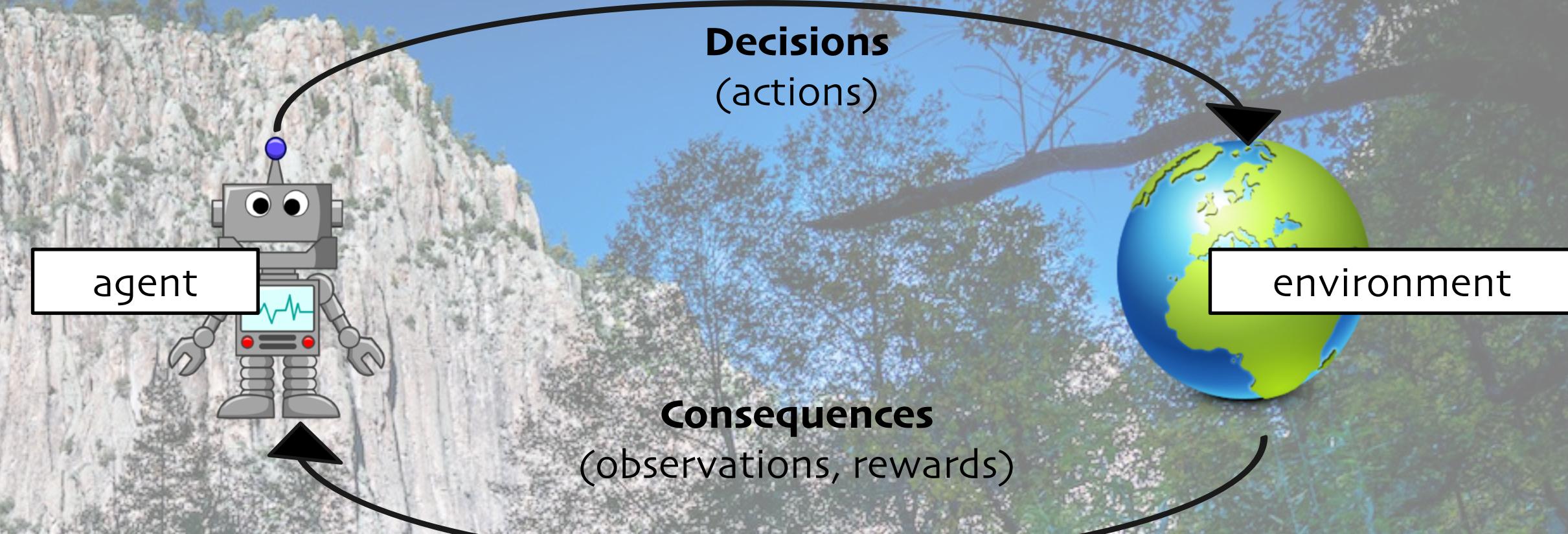
Advertising,  
Recommendations

Finance

RL applications



# Reinforcement Learning



Games

Robotics,  
Autonomous  
Vehicles

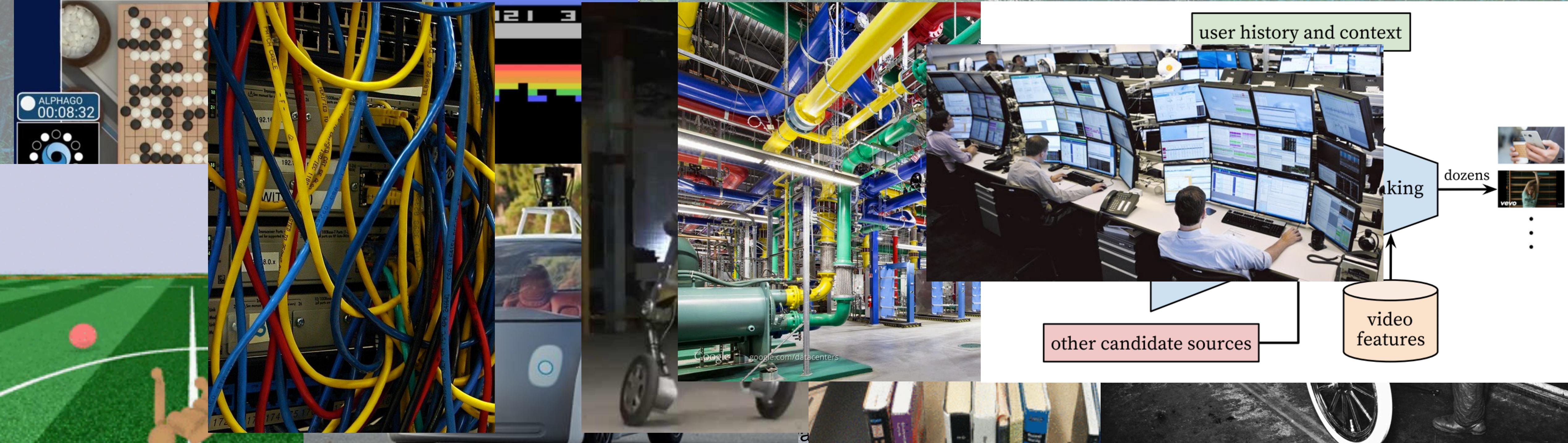
Industrial  
Processes

System  
Optimization

Advertising,  
Recommendations

Finance

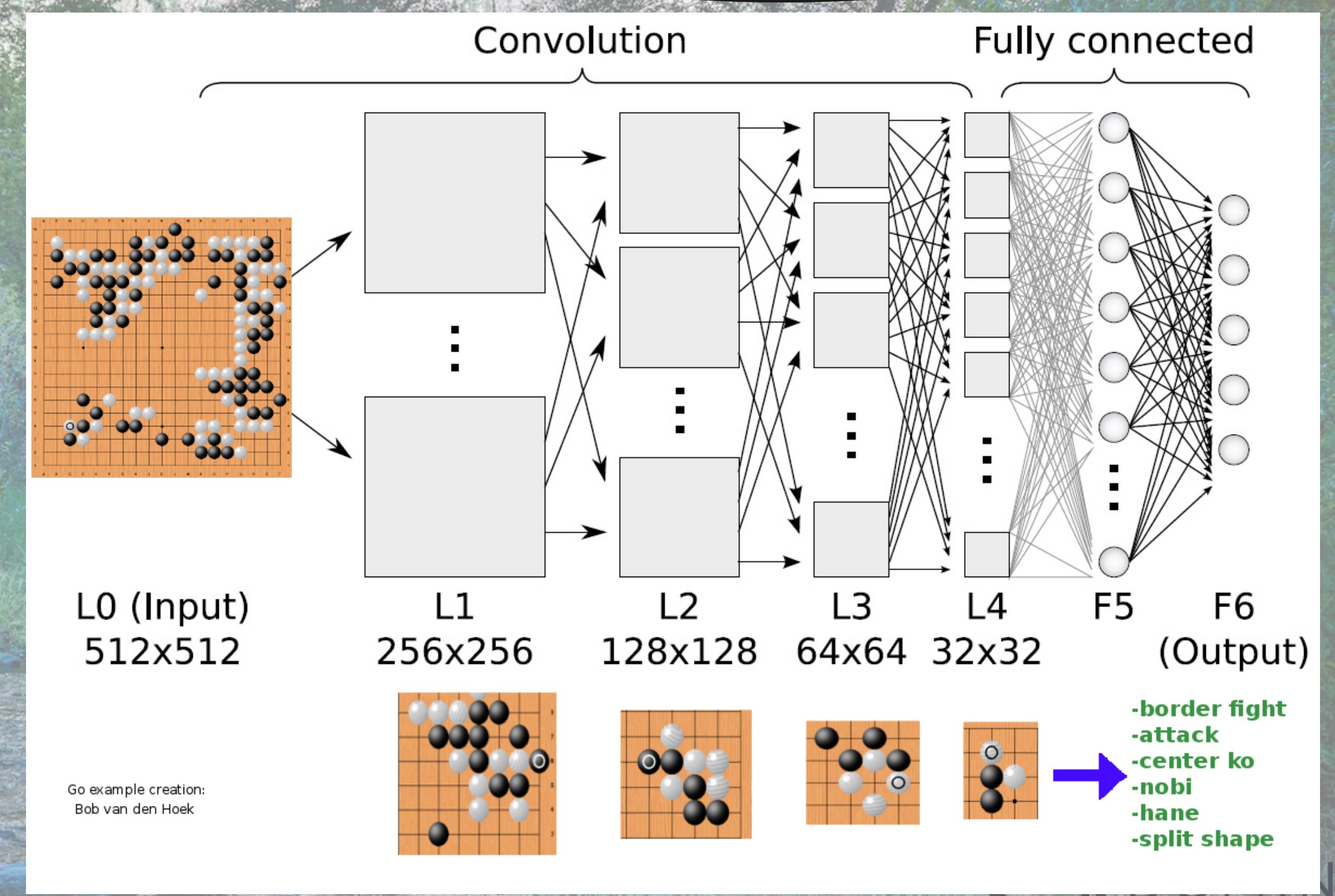
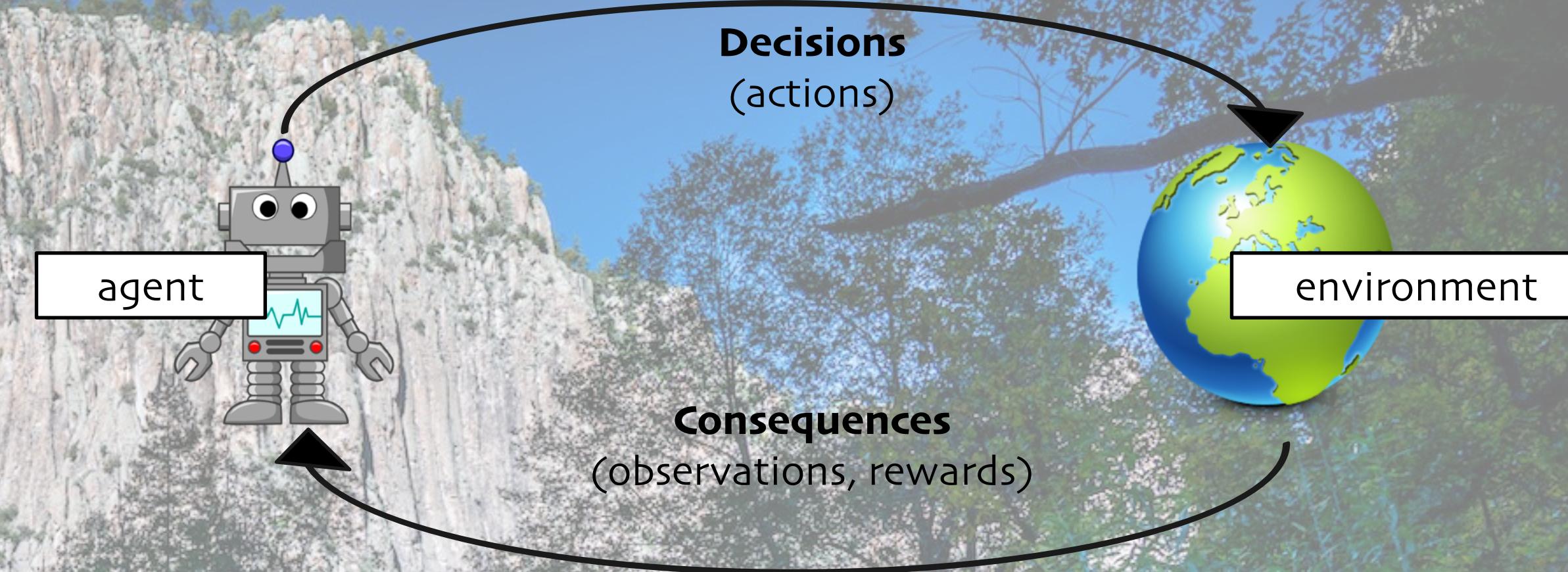
RL applications

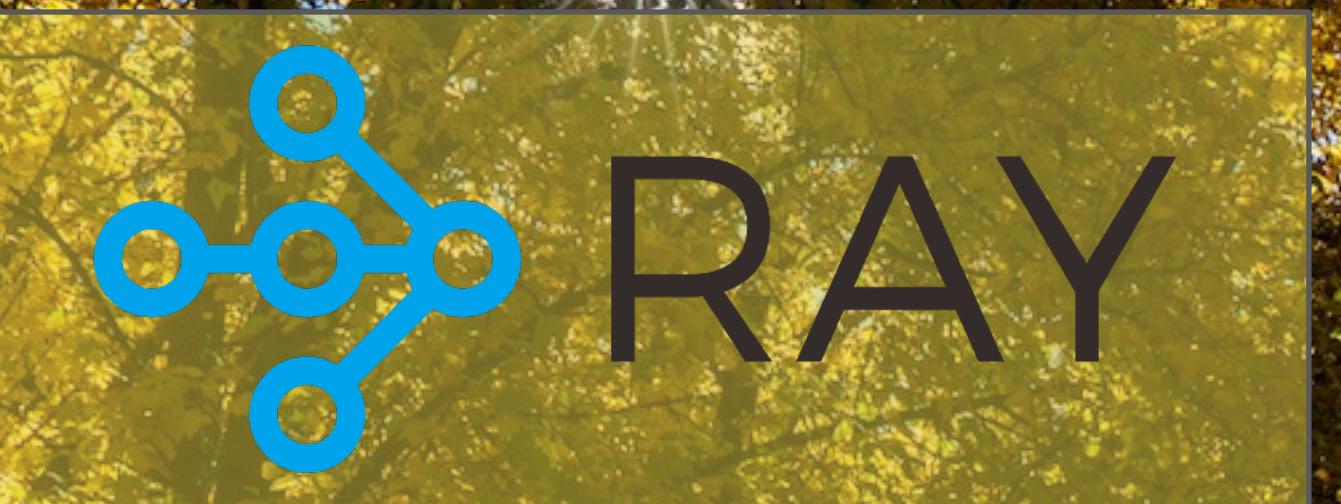


# Go as a Reinforcement Learning Problem

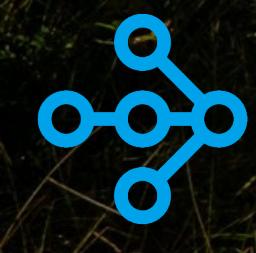
AlphaGo (Silver et al. 2016)

- **Observations:**
  - board state
- **Actions:**
  - where to place the stones
- **Rewards:**
  - 1 if win
  - 0 otherwise





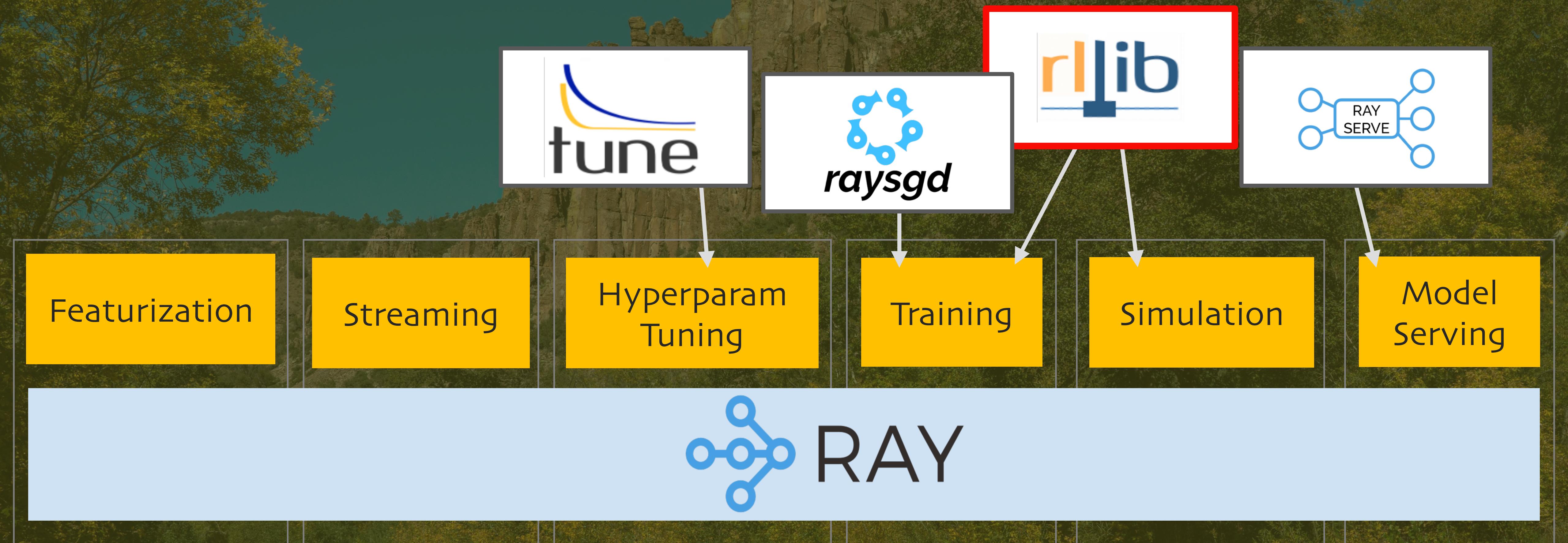
# Ray RLlib



@deanwampler

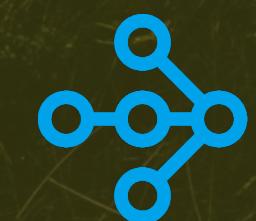


# Reinforcement Learning - Ray RLlib

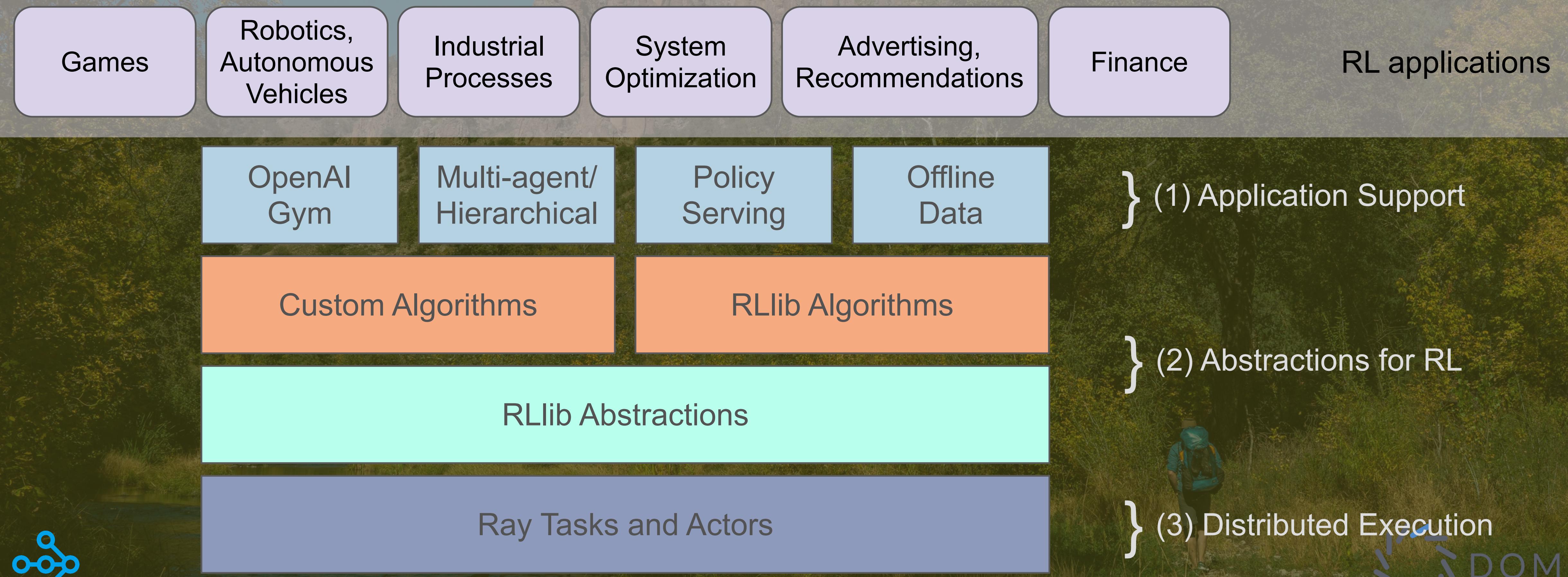


[rllib.io](http://rllib.io)

@deanwampler

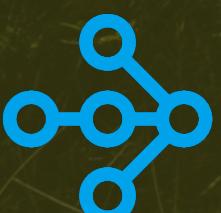


# RLLib: A Scalable, Unified Library for RL



# A Broad Range of Popular Algorithms

- High-throughput architectures
  - [Distributed Prioritized Experience Replay \(Ape-X\)](#)
  - [Importance Weighted Actor-Learner Architecture \(IMPALA\)](#)
  - [Asynchronous Proximal Policy Optimization \(APPO\)](#)
- Gradient-based
  - [Soft Actor-Critic \(SAC\)](#)
  - [Advantage Actor-Critic \(A<sub>2</sub>C, A<sub>3</sub>C\)](#)
  - [Deep Deterministic Policy Gradients \(DDPG, TD3\)](#)
  - [Deep Q Networks \(DQN, Rainbow, Parametric DQN\)](#)
  - [Policy Gradients](#)
  - [Proximal Policy Optimization \(PPO\)](#)
- gradient-free
  - [Augmented Random Search \(ARS\)](#)
  - [Evolution Strategies](#)
- Multi-agent specific
  - [QMIX Monotonic Value Factorisation \(QMIX, VDN, IQN\)](#)
- Offline
  - [Advantage Re-Weighted Imitation Learning \(MARWIL\)](#)



# Amazon SageMaker RL

Reinforcement learning for every developer and data scientist



## Amazon SageMaker RL

End-to-end examples for classic RL and real-world RL applications

Robotics

Industrial Control

HVAC

Autonomous Vehicles

Operations

Finance

Games

NLP

RL Environments to model real-world problems

AWS Simulation Environments

Amazon Sumerian

AWS RoboMaker

Open Source Environments

EnergyPlus

RoboSchool

PyBullet

...

Custom Environments

Bring Your Own

Commercial simulators

MATLAB & Simulink

Open AI Gym

RL Toolkits that provide RL agent algorithm implementations

RL-Coach

DQN

PPO

HER

Rainbow

...

RL-Ray RLLib

APEX

ES

IMPALA

A3C

...

Open AI Baselines

TRPO

GAIL

...

...

TensorFlow

MxNet

PyTorch

Chainer

Training Options

Single Machine / Distributed

Local / Remote simulation

CPU / GPU Hardware

SageMaker supported

Customer BYO

# Now in Azure

The screenshot shows a Microsoft Docs page titled "Reinforcement learning (preview) with Azure Machine Learning". The page is part of the "Azure Machine Learning Documentation" under the "Overview" section. It includes a note about the feature being a preview and supported only by Ray and RLLib frameworks. The page also lists the Python library Ray\_RLLib as a dependency for managing distributed RL jobs. The URL for the page is <https://docs.microsoft.com/en-us/azure/machine-learning/tutorials/reinforcement-learning-configure-training-environment>.

Microsoft | Docs Documentation Learn Q&A Code Samples

Azure Product documentation ▾ Architecture ▾ Learn Azure ▾ Develop ▾ Resources ▾

Azure / Machine Learning Bookmark

Azure Machine Learning Documentation

✓ Overview

- What is Azure Machine Learning?
- Azure Machine Learning vs Studio (classic)
- Architecture & terms

✓ Tutorials

- > Studio
- > Python SDK
- > R SDK
- > Machine Learning CLI
- > Visual Studio Code

> Samples

> Concepts

## Reinforcement learning (preview) with Azure Machine Learning

05/05/2020 • 11 minutes to read • 

**APPLIES TO:**  Basic edition  Enterprise edition [\(Upgrade to Enterprise edition\)](#)

 **Note**

Azure Machine Learning Reinforcement Learning is currently a preview feature. Only Ray and RLLib frameworks are supported at this time.

In this article, you learn how to train a reinforcement learning (RL) agent to play the video game Pong. You will use the open-source Python library [Ray\\_RLLib](#) with Azure Machine Learning to manage the complexity of distributed RL jobs.

In this article you will learn how to:

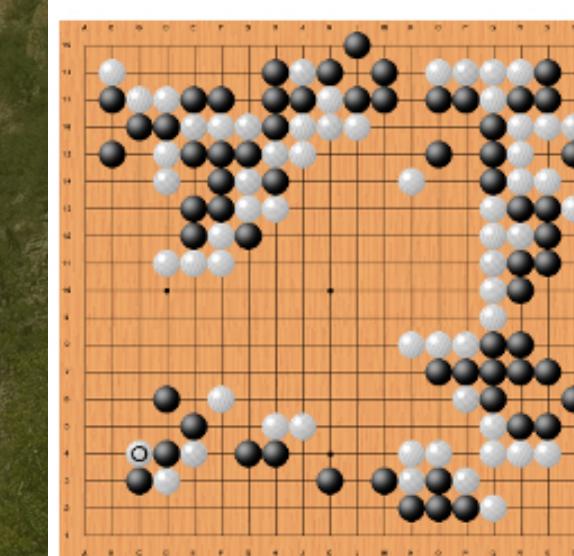
# Diverse Compute Requirements Motivated Creation of Ray!

And repeated play,  
over and over again,  
to train for achieving  
the best reward

Simulator (game  
engine, robot sim,  
factory floor sim...)

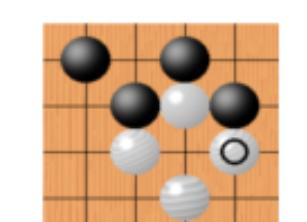
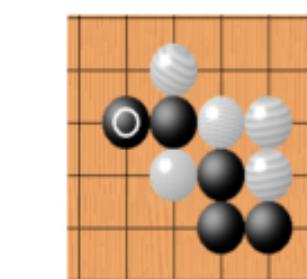
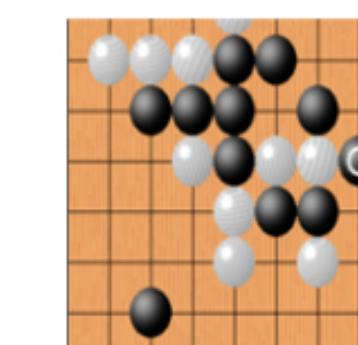
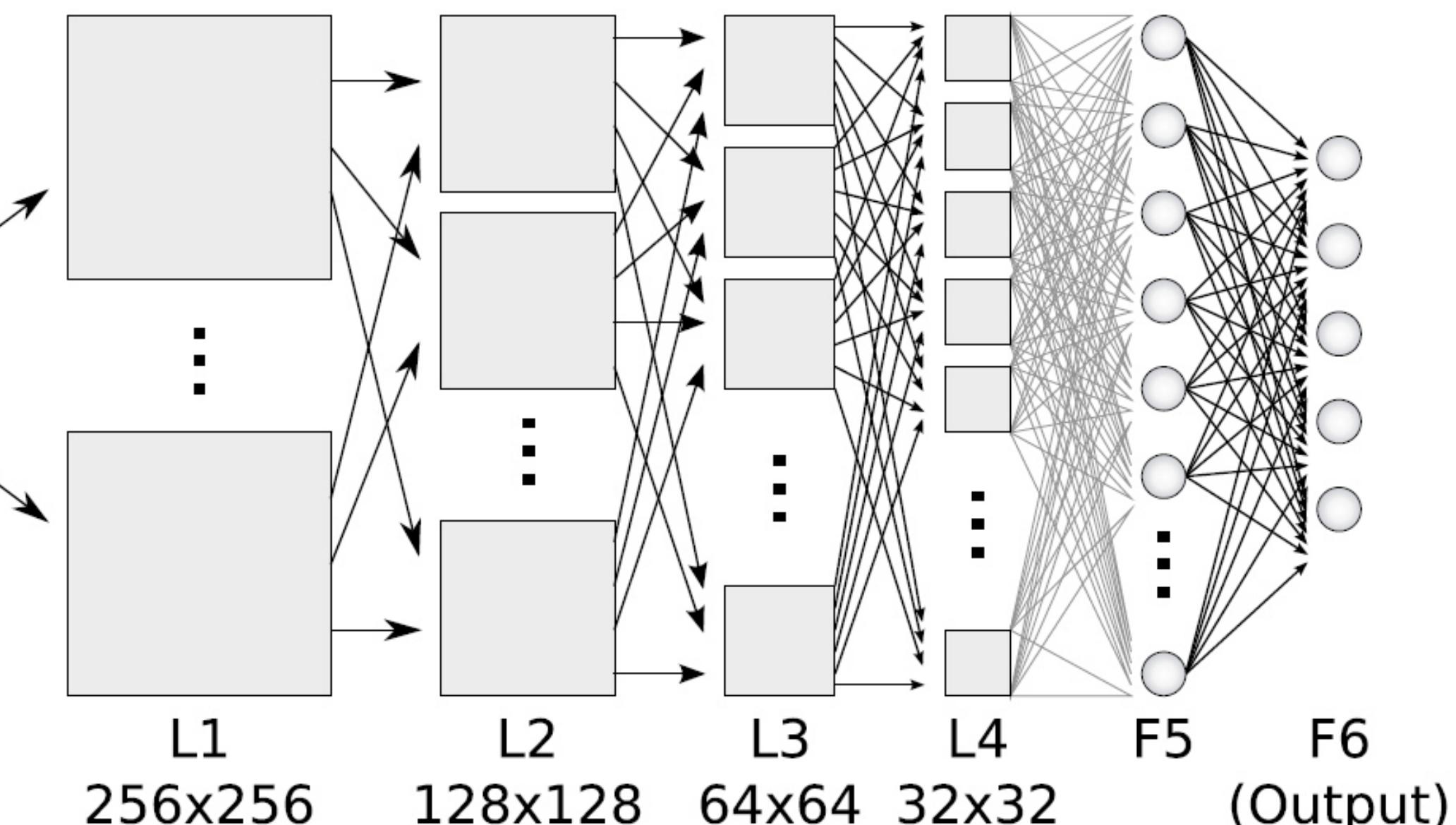
Complex agent?

Neural network  
“stuff”



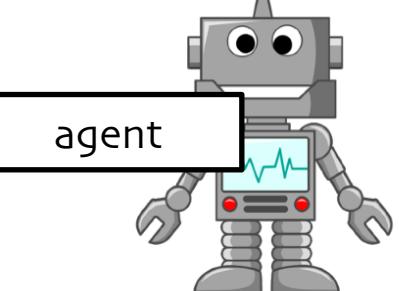
L0 (Input)  
512x512

Convolution



-border fight  
-attack  
-center ko  
-nobi  
-hane  
-split shape

Decisions (**actions**)

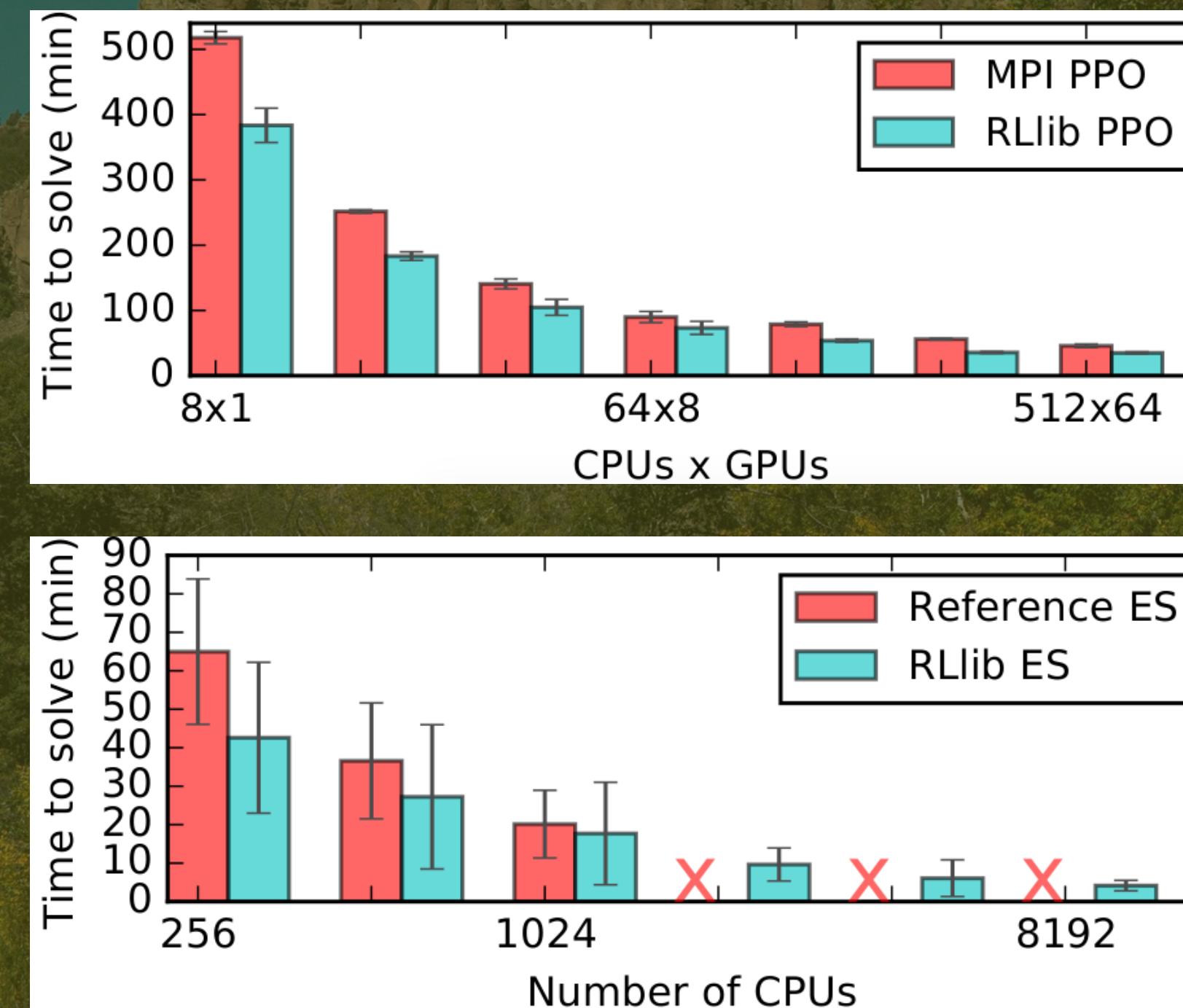


Consequences  
(**observations, rewards**)

Bob van den Hoek  
example creation:

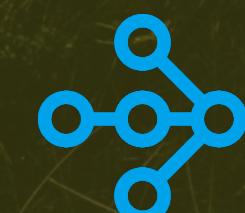
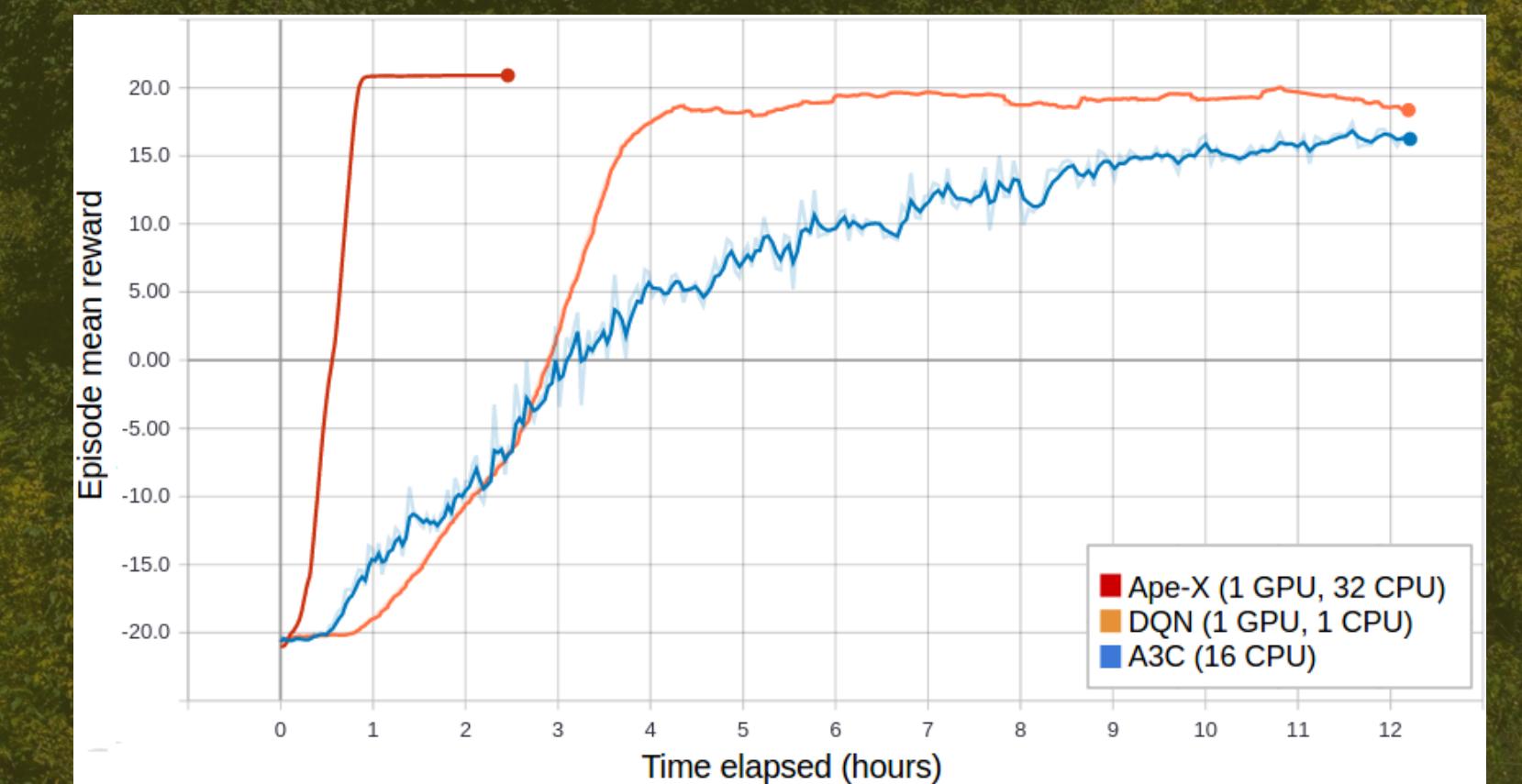
# RLLib Provides a Unified Framework for Scalable RL that Doesn't Compromise on Performance

Distributed PPO



Evolution  
Strategies

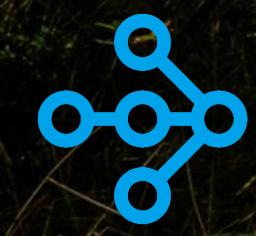
Ape-X Distributed  
DQN, DDPG



A scenic landscape featuring a large, rugged, reddish-brown rock formation rising from a lush green hillside. In the foreground, a person wearing a blue backpack walks along a path. A small stream flows through the bottom left corner. The sky is clear and blue.

# demo

<https://github.com/anyscale/academy>

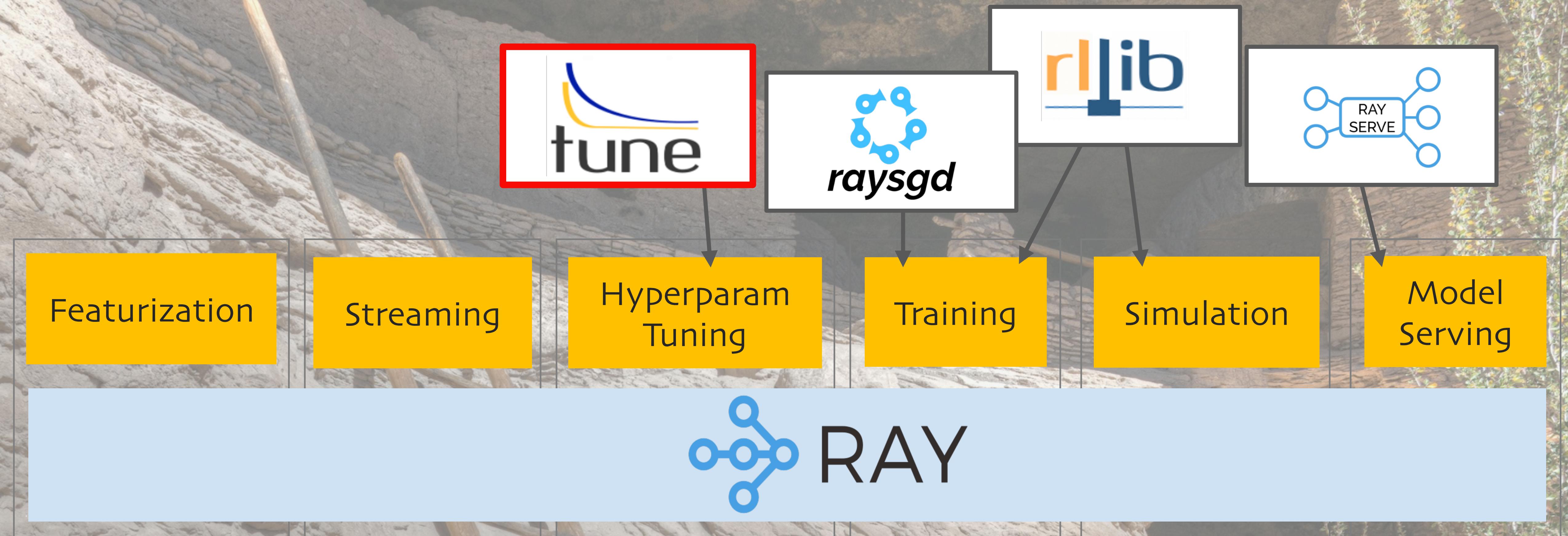




@deanwampler

DOMINO

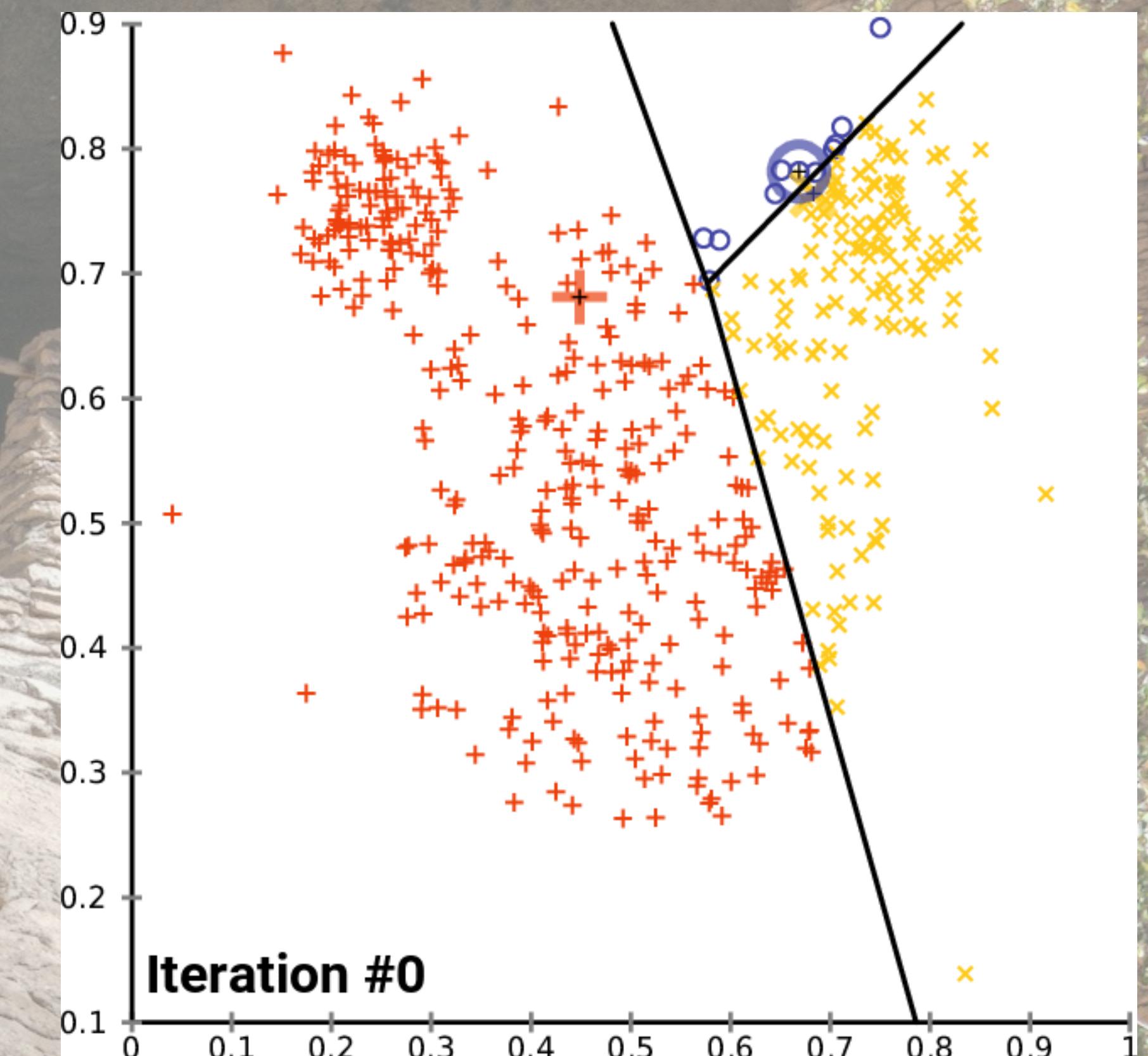
# Hyperparameter Tuning - Ray Tune



# What Is Hyperparameter Tuning?

Trivial example:

- What's the best value for "k" in k-means??
- k is a "hyperparameter"
- The resulting clusters are defined by "parameters"

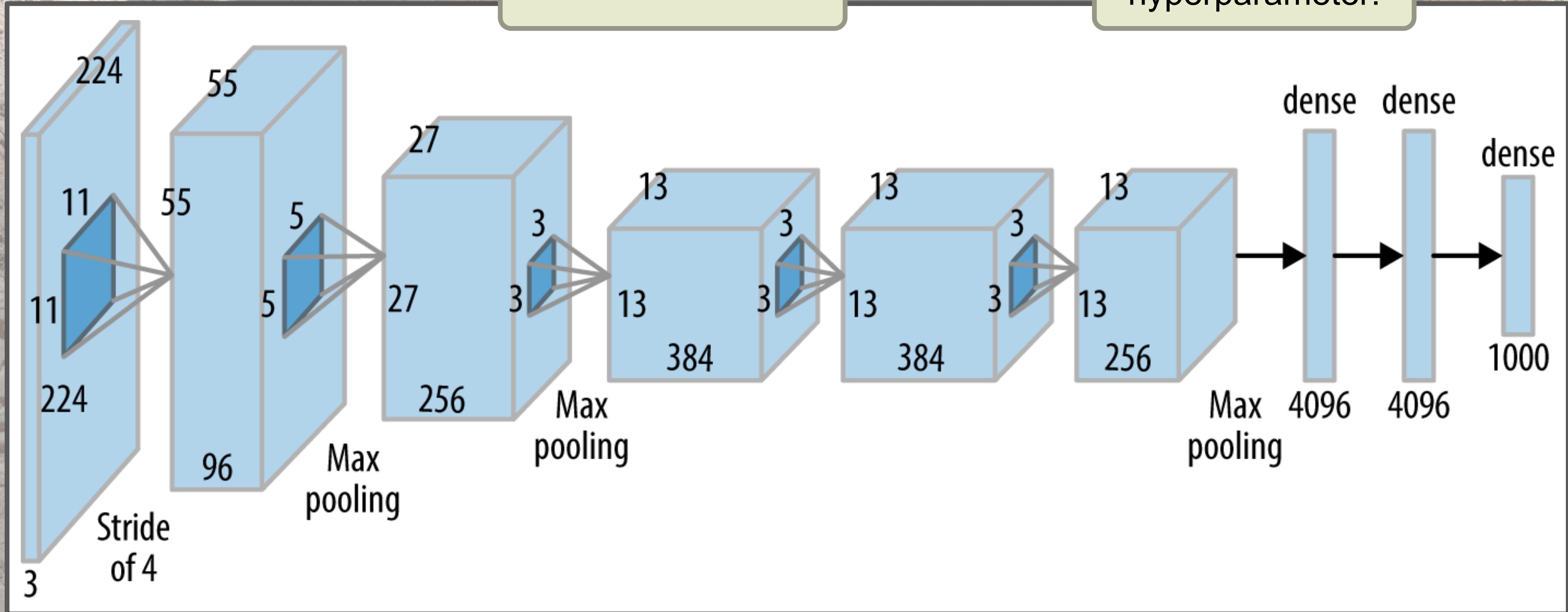


credit: [https://commons.wikimedia.org/wiki/File:K-means\\_convergence.gif](https://commons.wikimedia.org/wiki/File:K-means_convergence.gif)

# Nontrivial Example - Neural Networks

How many layers?  
What kinds of layers?

Every number  
shown is a  
hyperparameter!

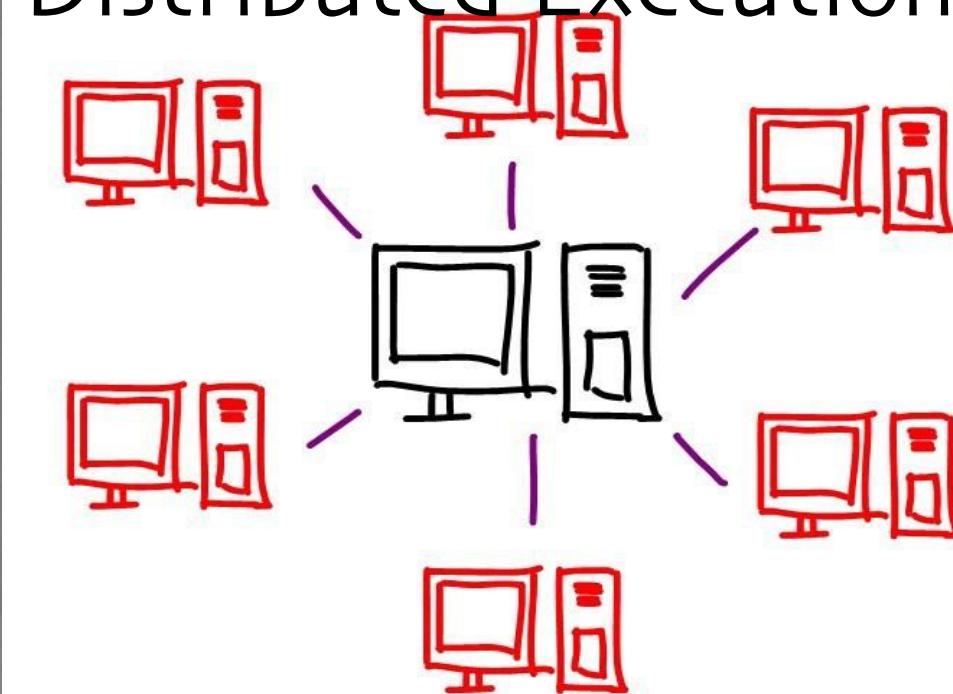


# Tune is Built with Deep Learning as a Priority

Resource Aware  
Scheduling



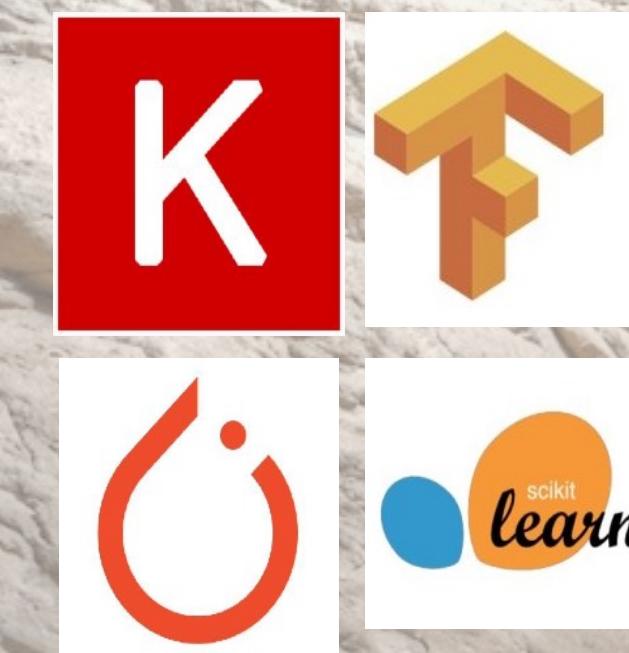
Seamless  
Distributed Execution



Simple API for  
new algorithms

```
class TrialScheduler:  
    def on_result(self, trial, result): ...  
    def choose_trial_to_run(self): ...
```

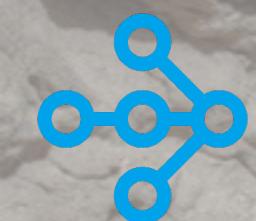
Framework Agnostic



[tune.io](https://tune.io)

@deanwampler

 DOMINO

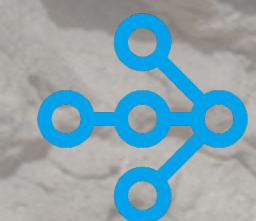
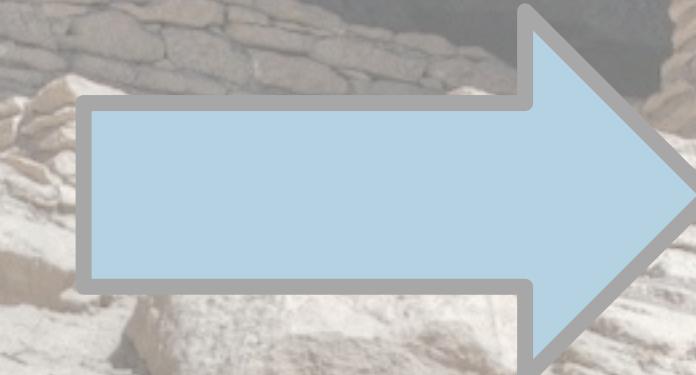


# Why We Need a Framework for Tuning Hyperparameters

We want the best model

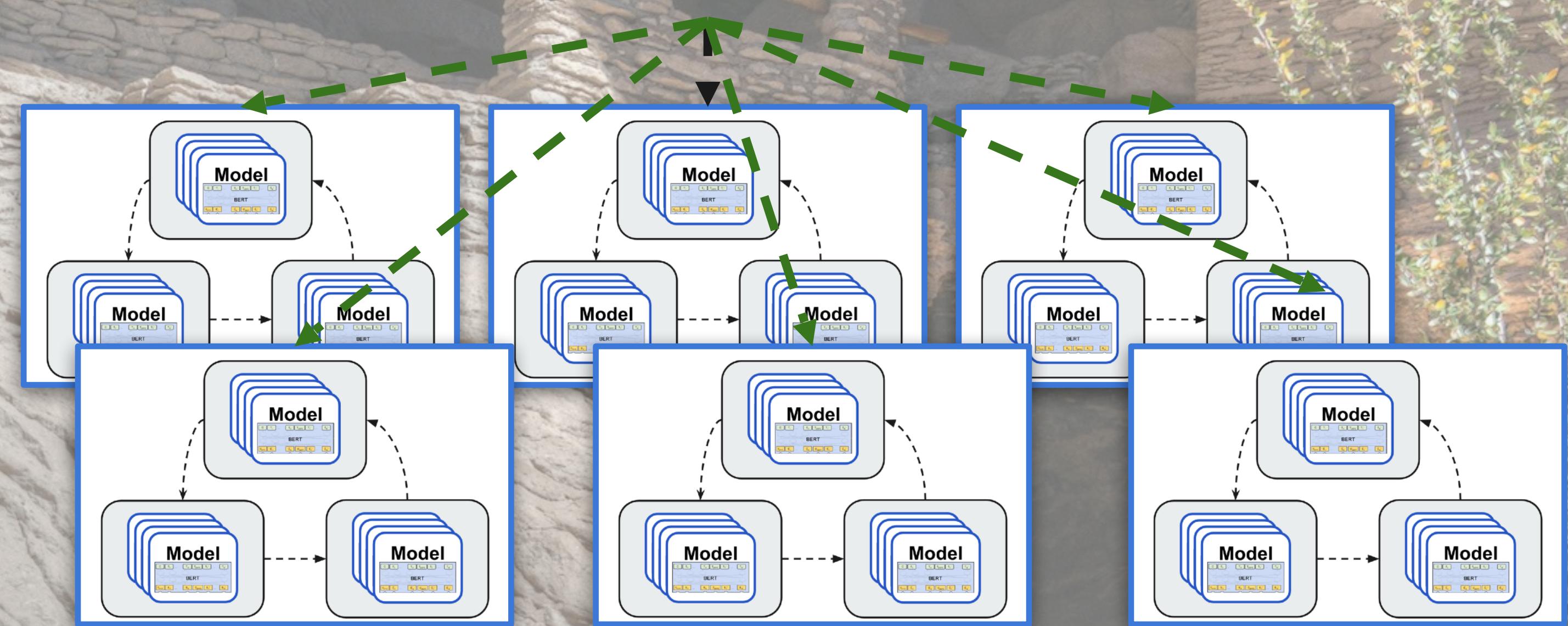
Resources are expensive

Model training is time-consuming

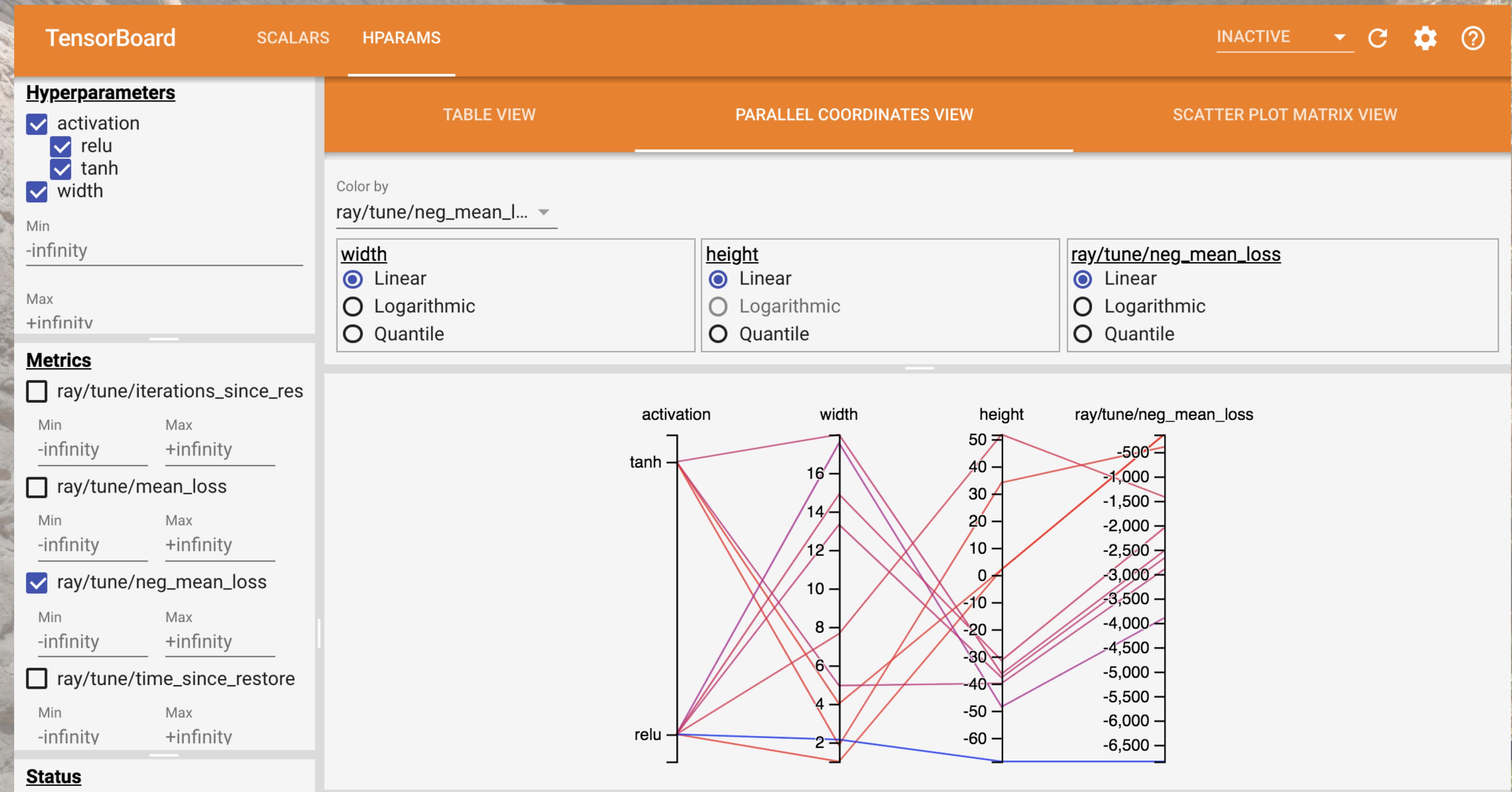


# Tuning + Distributed Training

```
tune.run(PytorchTrainable,  
 config={  
     "model_creator": PretrainBERT,  
     "data_creator": create_data_loader,  
     "use_gpu": True,  
     "num_replicas": 8,  
     "lr": tune.uniform(0.001, 0.1)  
 },  
 num_samples=100,  
 search_alg=BayesianOptimization()
```



# Native Integration with TensorBoard HParams





# Other Uses of Ray: Microservices?

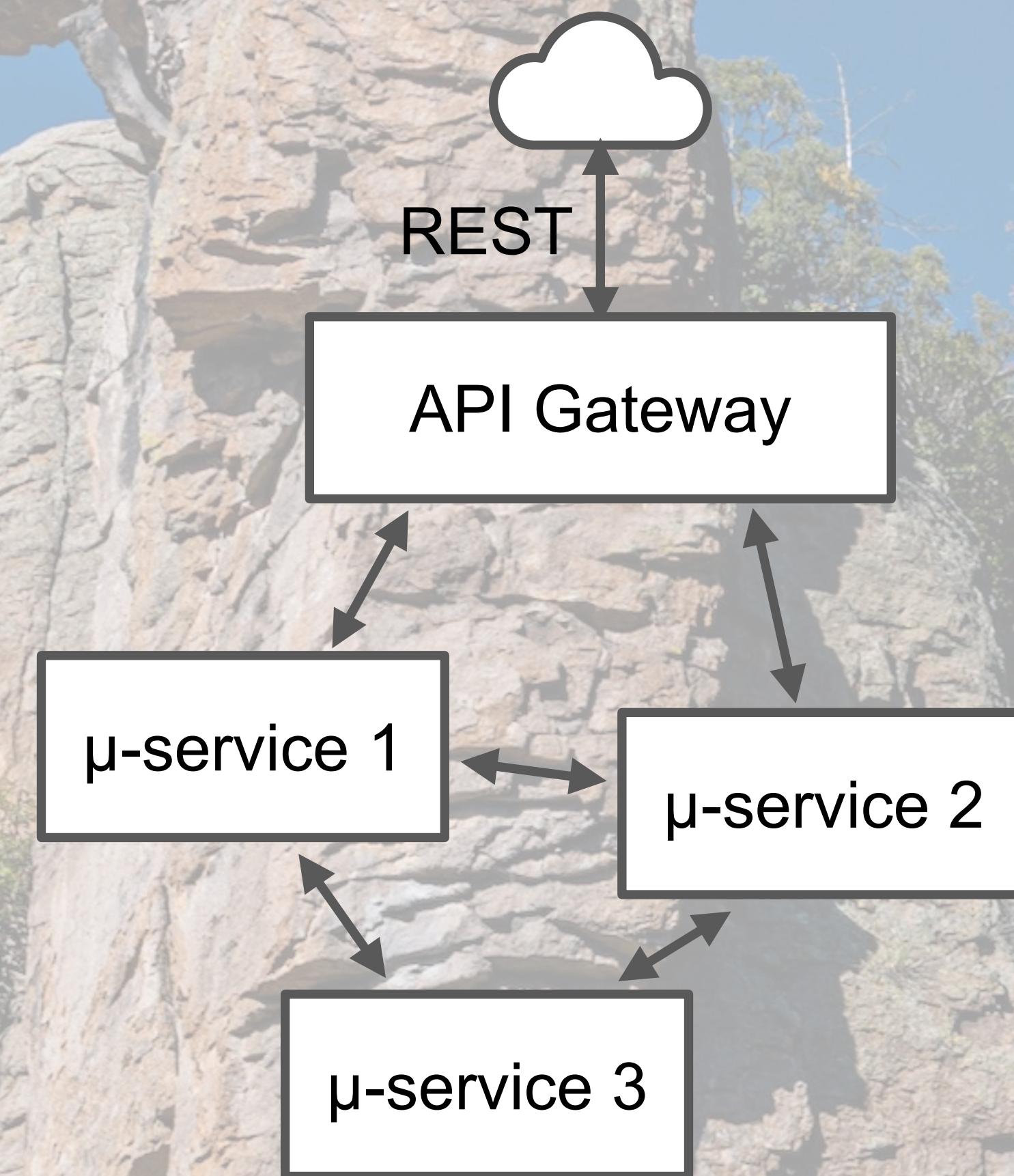


@deanwampler

DOMINO

# What Are Microservices?

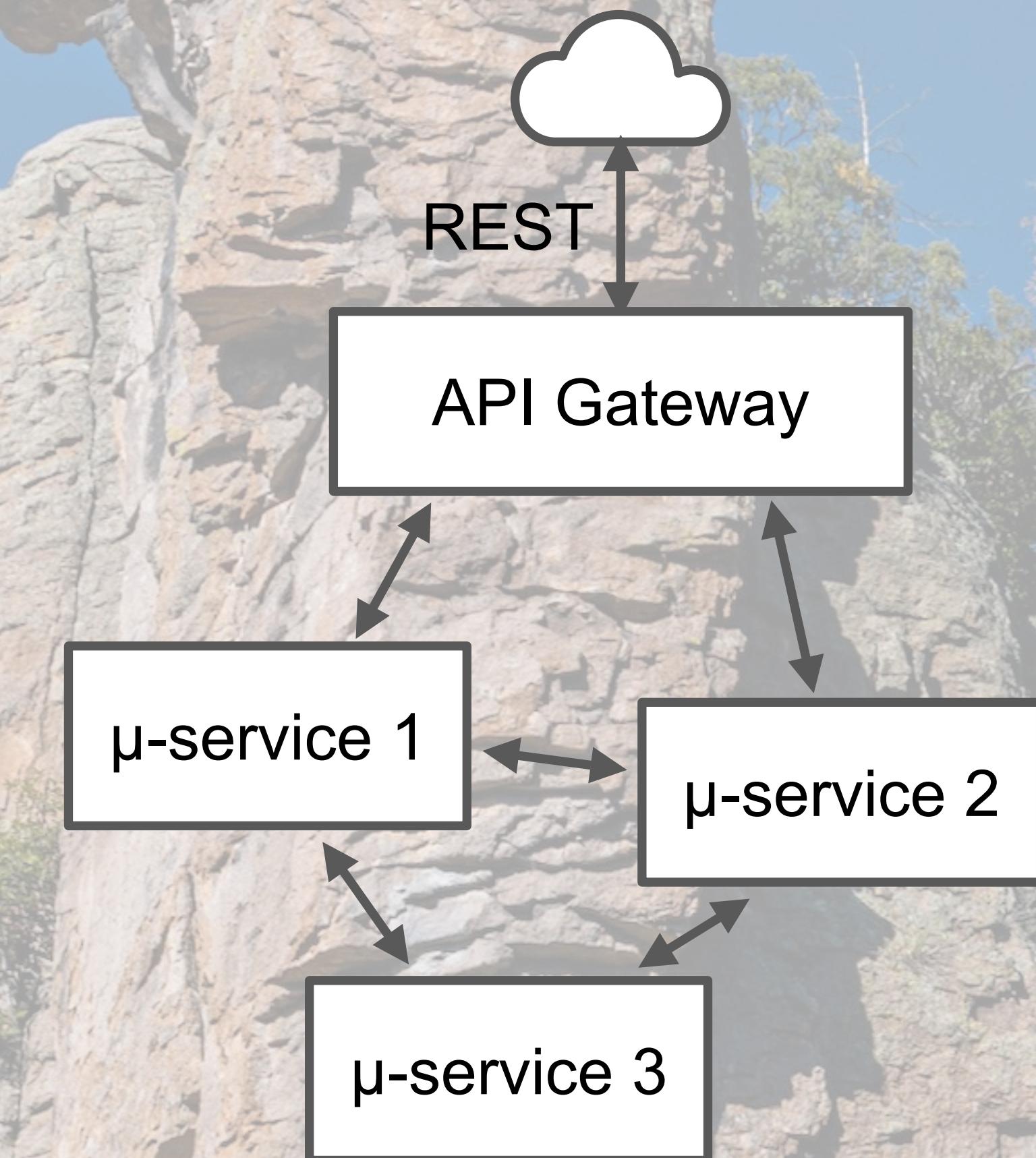
- They partition the domain
  - Conway's Law - Embraced
  - Separate responsibilities
- Separate management



# What Are Microservices?

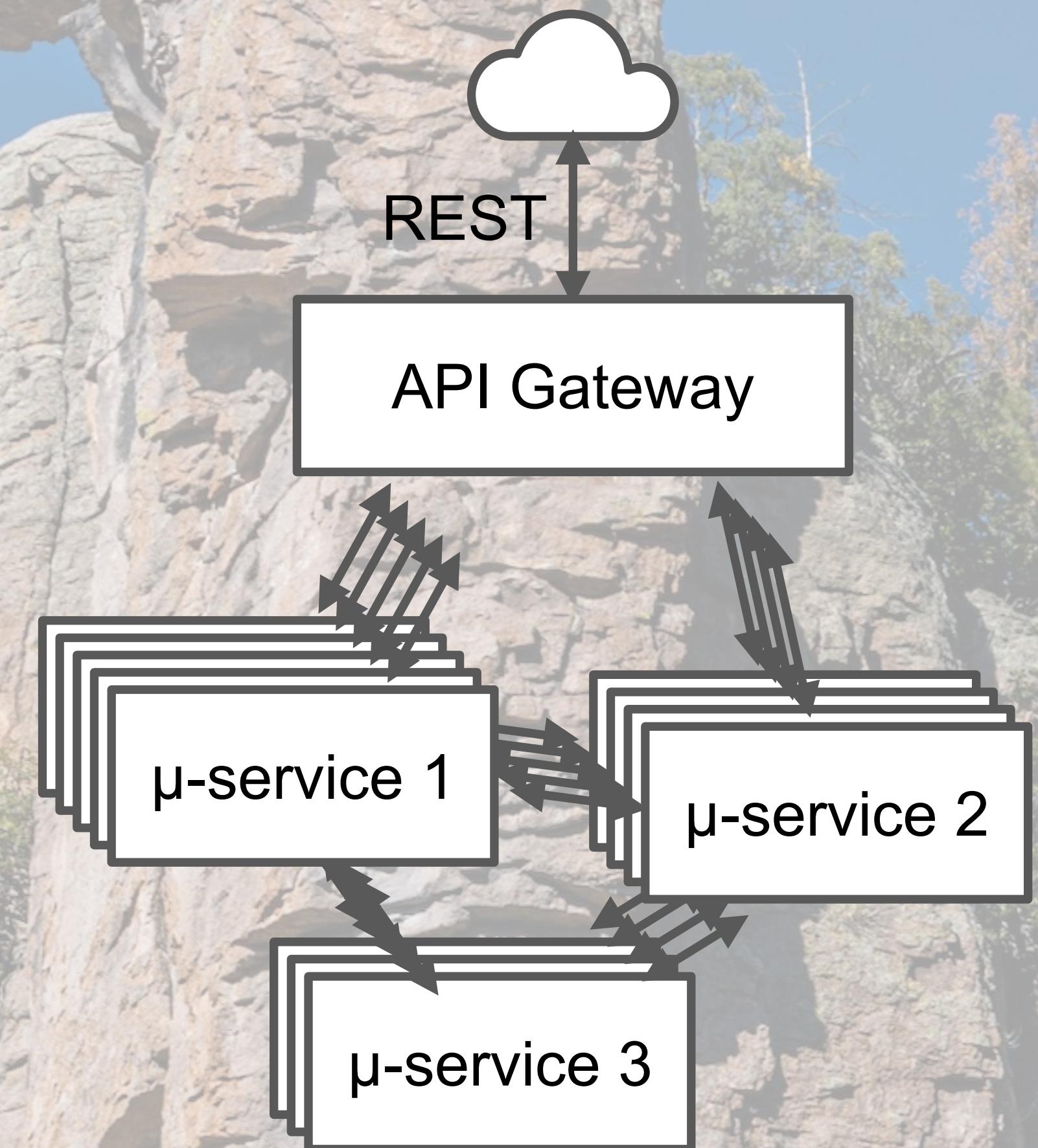
- They partition the domain
  - Conway's Law - Embraced
  - Separate responsibilities
- Separate management

What we mostly care  
about for today's talk, the  
“Ops in DevOps”



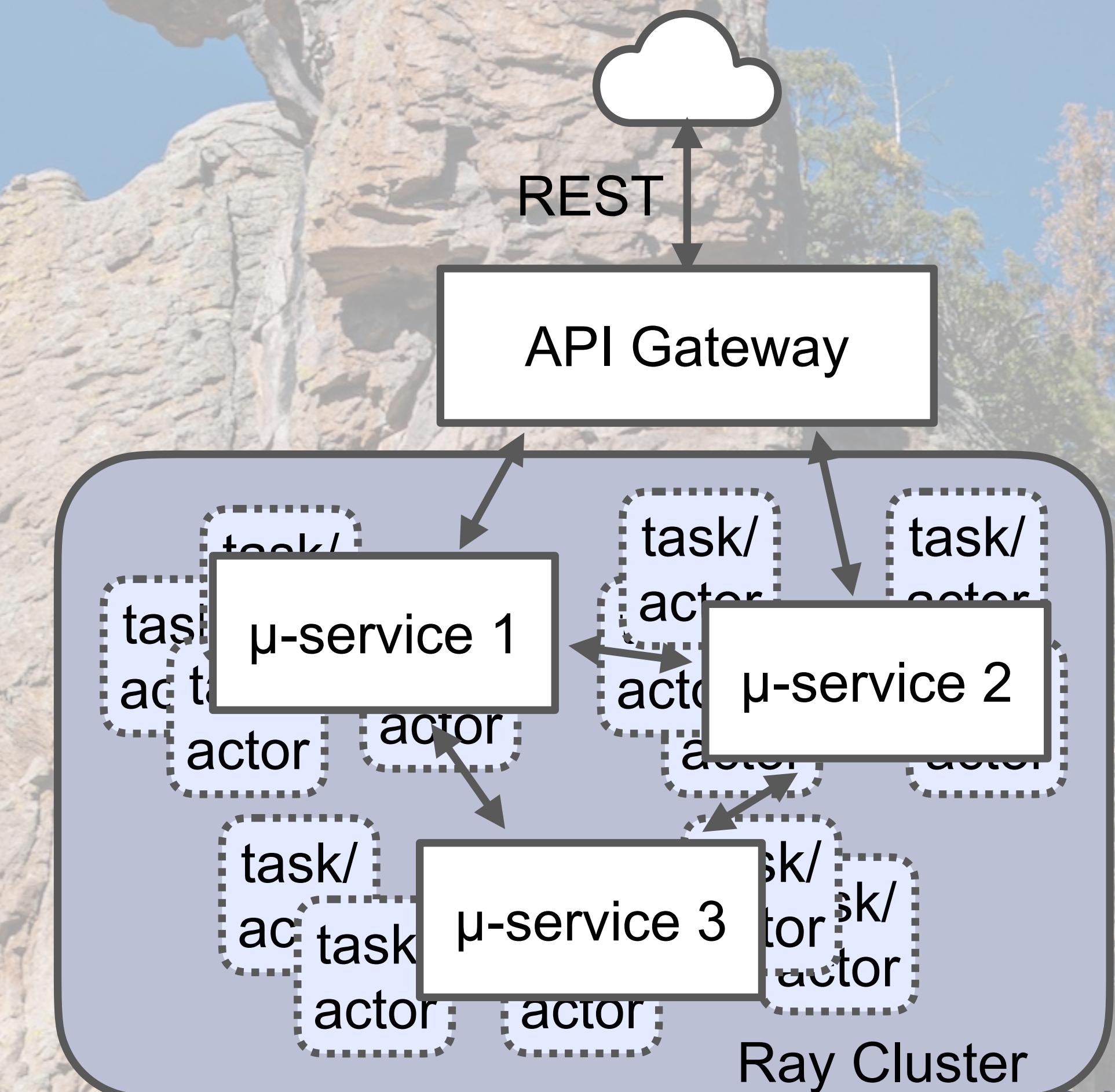
# Separate Management

- Each team manages its own instances
- Each microservice has a different number of instances for scalability and resiliency
- But they have to be managed **explicitly**



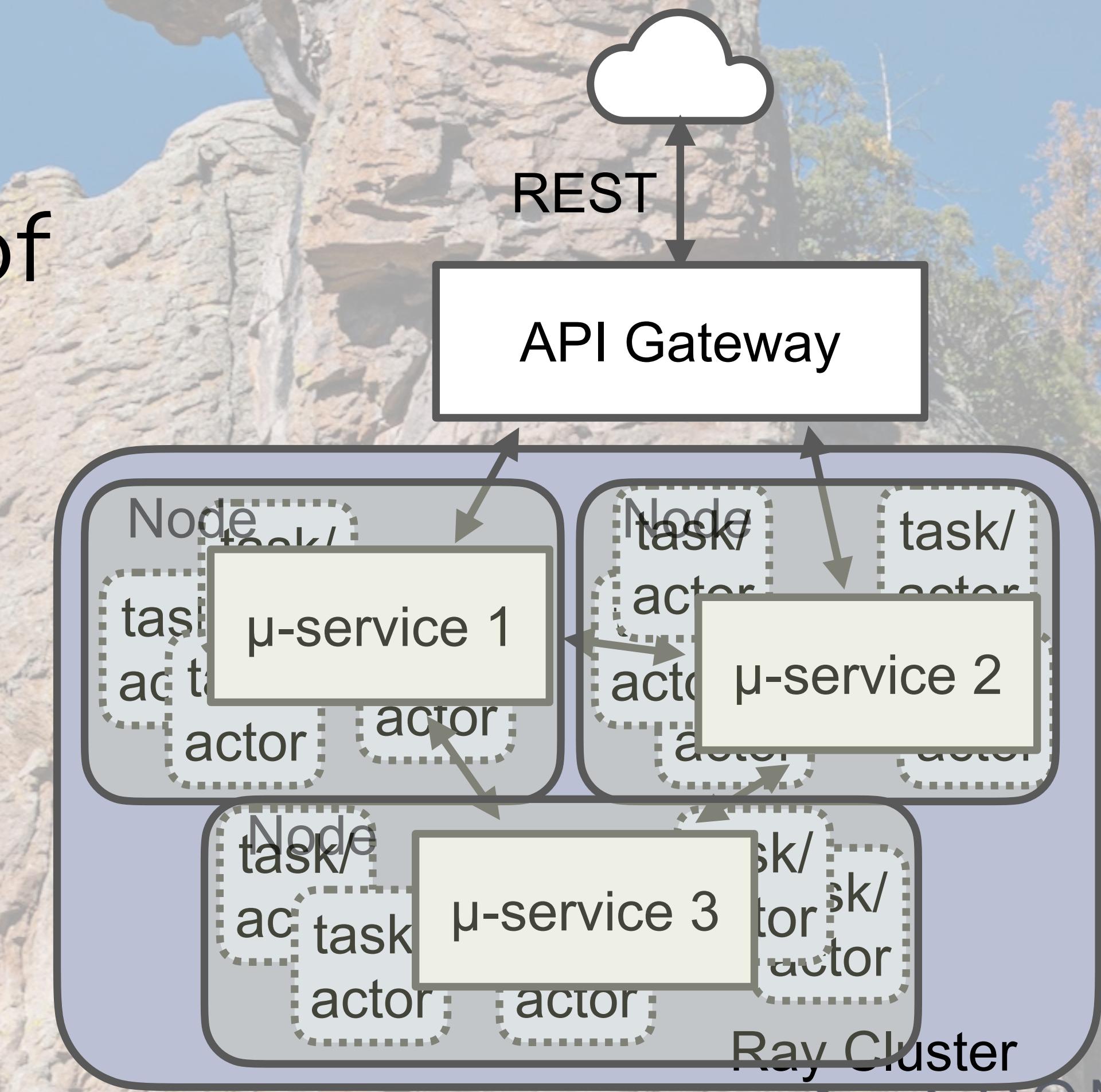
# Management - Simplified

- With Ray, you have one “logical” instance to manage and Ray does the cluster-wide scaling for you.



# What about Kubernetes (and others...)?

- Ray scaling is very fine grained.
- It operates within the “nodes” of coarse-grained managers
- Containers, pods, VMs, or physical machines





# Next Steps: Adopting Ray



@deanwampler



# If you're already using...

- joblib
- multiprocessing.Pool

For example, from this:

```
from multiprocessing.pool import Pool
```

To this:

```
from ray.util.multiprocessing.pool import Pool
```

- Use Ray's implementations
  - Drop-in replacements
  - Change import statements
  - Break the one-node limitation!

- ... And Ray is integrated with asyncio

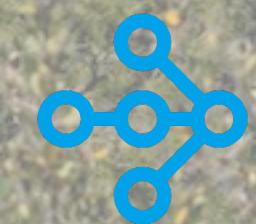
See these blog posts:

<https://medium.com/distributed-computing-with-ray/how-to-scale-python-multiprocessing-to-a-cluster-with-one-line-of-code-d19f242f60ff>

<https://medium.com/distributed-computing-with-ray/easy-distributed-scikit-learn-training-with-ray-54ff8b643b33>

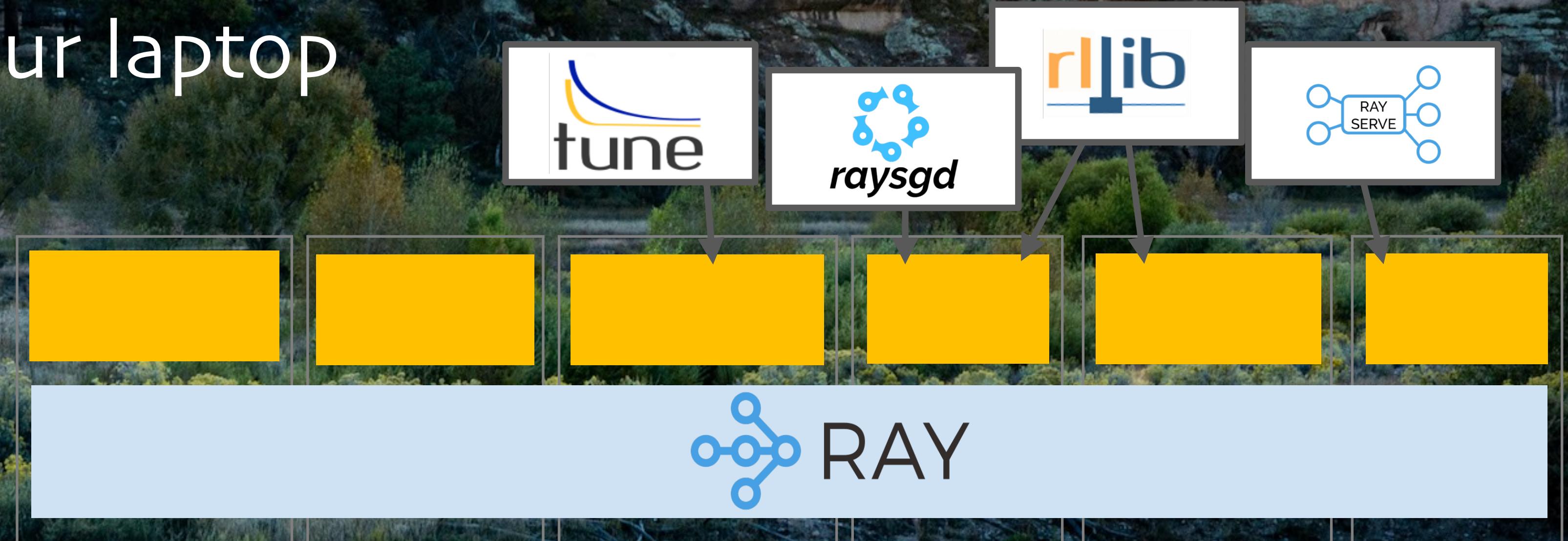
# Ray Community and Resources

- [ray.io](https://ray.io)
- Need help?
  - Ray Slack: [ray-distributed.slack.com](https://ray-distributed.slack.com)
  - [ray-dev](https://groups.google.com/g/ray-dev) Google group



# Recap

- Ray is the new state-of-the-art for distributed computing
  - The shortest path from your laptop to the cloud
  - Run complex distributed tasks on large clusters from simple code on your laptop



# Thank You

ray.io

dean.wampler@dominodatalab.com

@deanwampler