

How Microservices and Serverless Computing Enable the Next Gen of Machine Intelligence



Jon Peck

Full-Spectrum Developer & Advocate

ALGORITHMIA

 jpeck@algorithmia.com

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

 **Algorithmia.com**

Making state-of-the-art algorithms
discoverable and **accessible** to everyone

The Problem: ML is in a huge growth phase, difficult/expensive for DevOps to keep up

Initially:

- A few models, a couple frameworks, 1-2 languages
- Dedicated hardware or VM Hosting
- IT Team for DevOps
- High time-to-deploy, manual discoverability
- Few end-users, heterogenous APIs (if any)

Pretty soon...

- > 5,000 algorithms (50k versions) on many runtimes / frameworks
- > 60k algorithm developers: heterogenous, largely unpredictable
- Each algorithm: 1 to 1,000 calls/second, a lot of variance
- Need auto-deploy, discoverability, low (15ms) latency
- Common API, composability, fine-grained security

The Need: an “Operating System for AI”

AI/ML scalable infrastructure on demand + marketplace

- Function-as-a-service for Machine & Deep Learning
- Discoverable, live inventory of AI via APIs
- Anyone can contribute & use
- Composable, Monetizable
- **Every developer on earth can make their app intelligent**



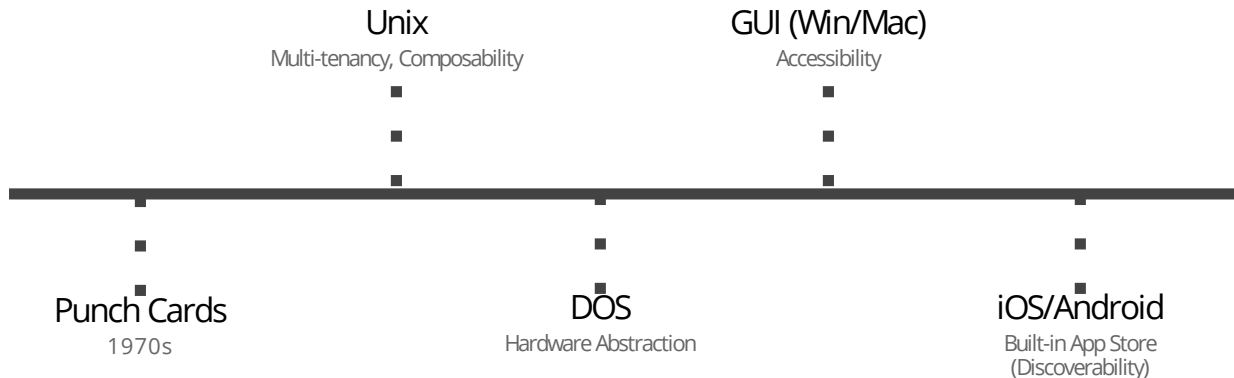
An Operating System for AI

What did the evolution of OS look like?

General-purpose computing had a long evolution, as we learned what the common problems were / what abstractions to build. AI is in the earlier stages of that evolution.

An Operating System:

- Provides common functionality needed by many programs
- Standardizes conventions to make systems easier to work with
- Presents a higher level abstraction of the underlying hardware



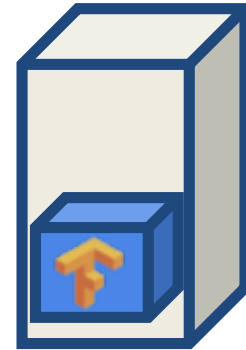
Use Case

Jian Yang made an app to recognize food "SeeFood"



Use Case

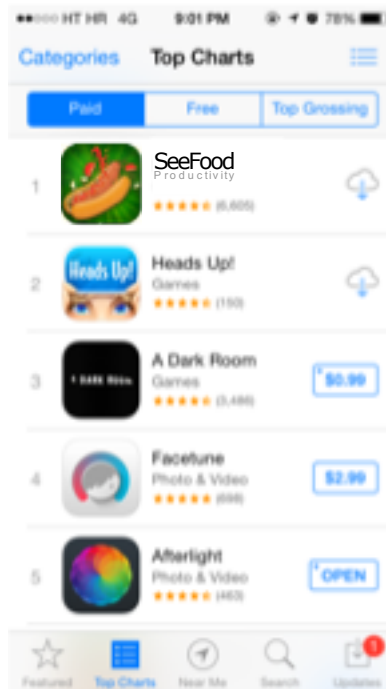
He deployed his trained model to a GPU-enabled server



GPU-enabled
Server

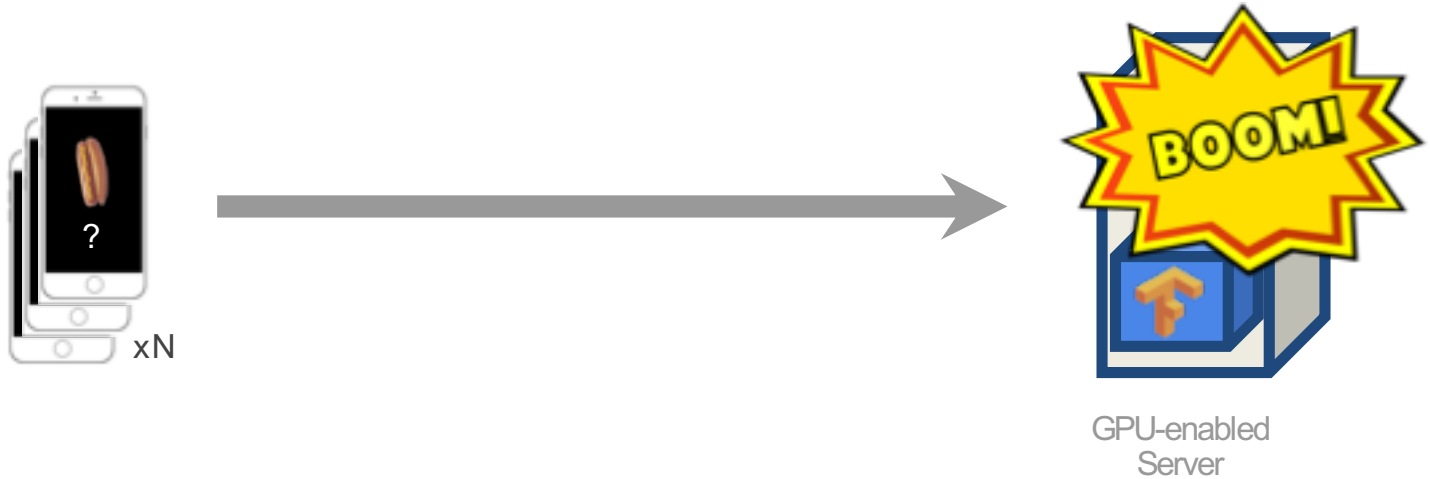
Use Case

The app is a hit!



Use Case

... and now his server is overloaded.



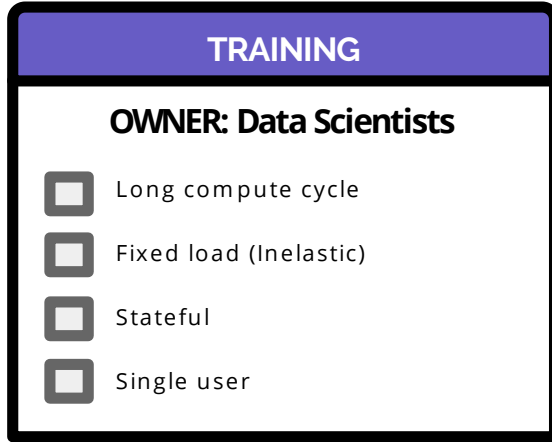
Characteristics of AI

- Two distinct phases: training and inference
- Lots of processing power
- Heterogenous hardware (CPU, GPU, FPGA, TPU, etc.)
- Limited by compute rather than bandwidth
- *“Tensorflow is open source, scaling it is not.”*

TRAINING

OWNER: Data Scientists

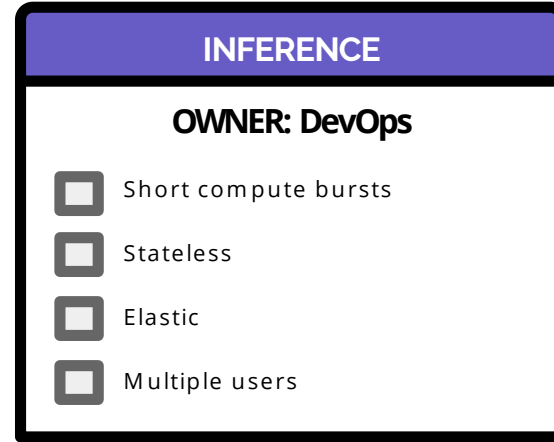
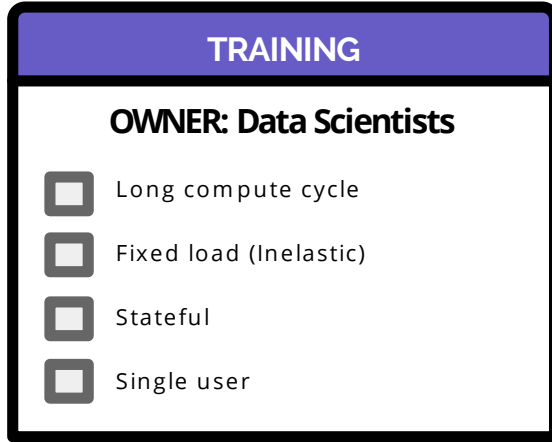
- ☐ Long compute cycle
- ☐ Fixed load (Inelastic)
- ☐ Stateful
- ☐ Single user



Analogous to dev tool chain.
Building and iterating over a
model is similar to building an
app.



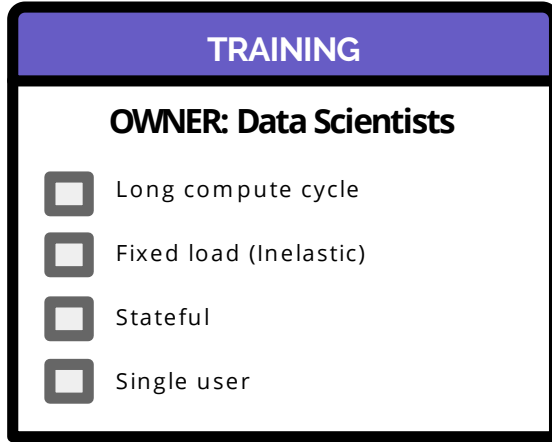
Metal or VM



Analogous to dev tool chain.
Building and iterating over a
model is similar to building an
app.



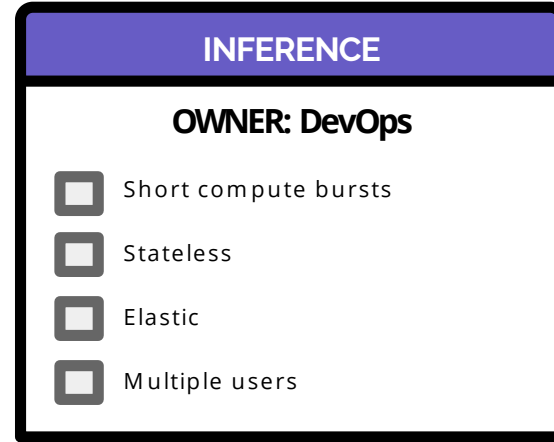
Metal or VM



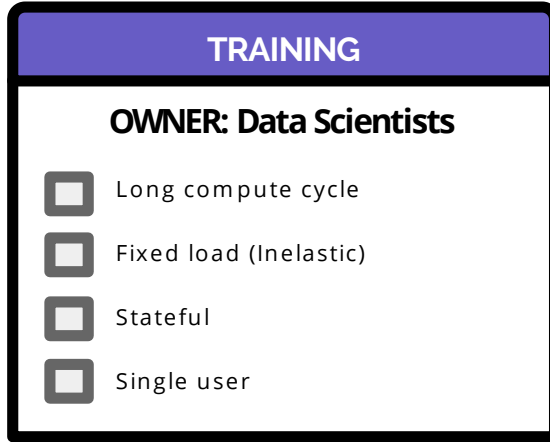
Analogous to dev tool chain.
Building and iterating over a
model is similar to building an
app.



Metal or VM



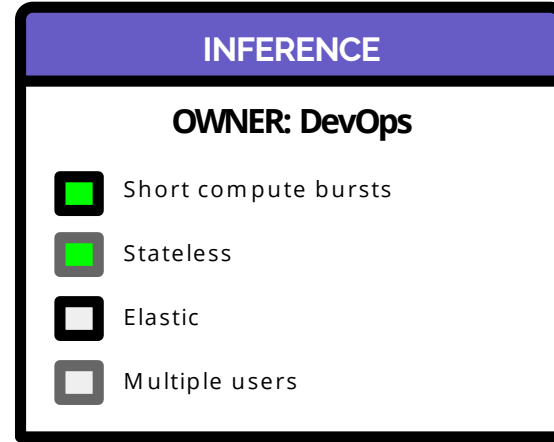
Analogous to an OS.
Running concurrent models
requires task scheduling.



Analogous to dev tool chain.
Building and iterating over a
model is similar to building an
app.



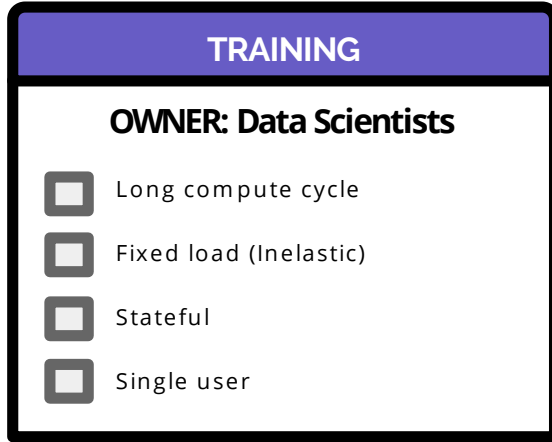
Metal or VM



Analogous to an OS.
Running concurrent models
requires task scheduling.



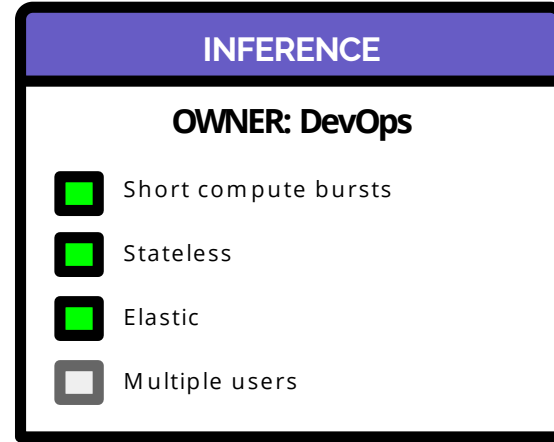
Containers



Analogous to dev tool chain.
Building and iterating over a
model is similar to building an
app.



Metal or VM



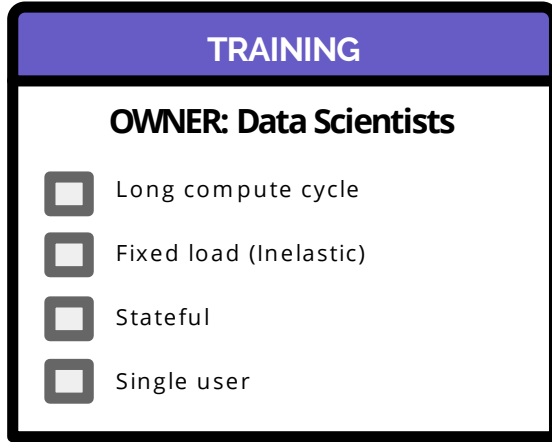
Analogous to an OS.
Running concurrent models
requires task scheduling.



Containers



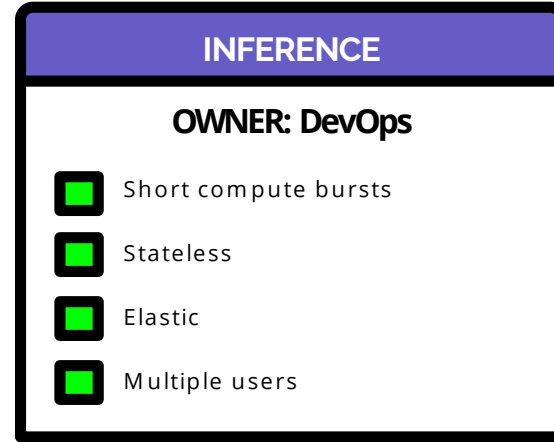
Kubernetes



Analogous to dev tool chain.
Building and iterating over a model is similar to building an app.



Metal or VM



Analogous to an OS.
Running concurrent models requires task scheduling.



Containers



Kubernetes



Microservices & Serverless Computing => ML Hosting

MICROSERVICES: the design of a system as independently deployable, loosely coupled services.

ADVANTAGES

- Maintainable, Scalable
- Software & Hardware Agnostic
- Rolling deployments

SERVERLESS: the encapsulation, starting, and stopping of singular functions per request, with a just-in-time-compute model.

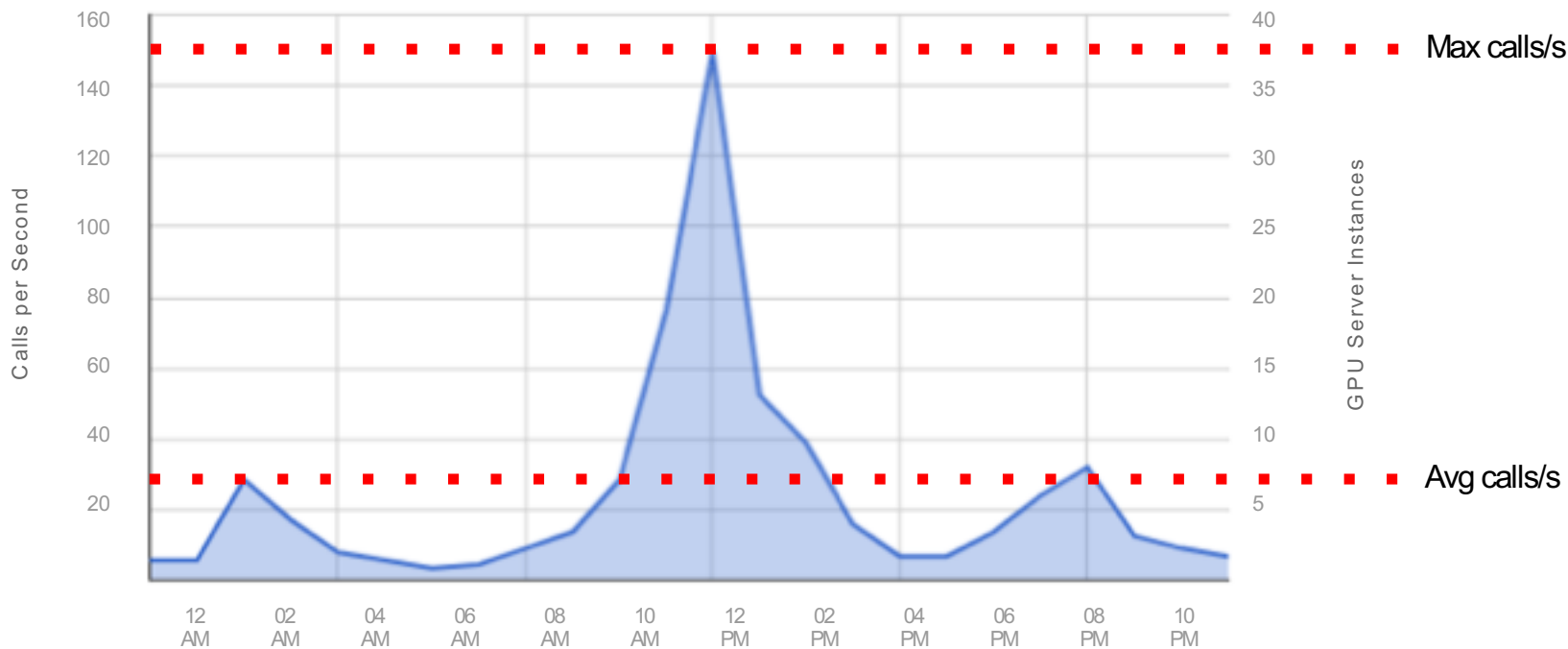
ADVANTAGES

- Elasticity, Cost Efficiency
- Concurrency
- Improved Latency



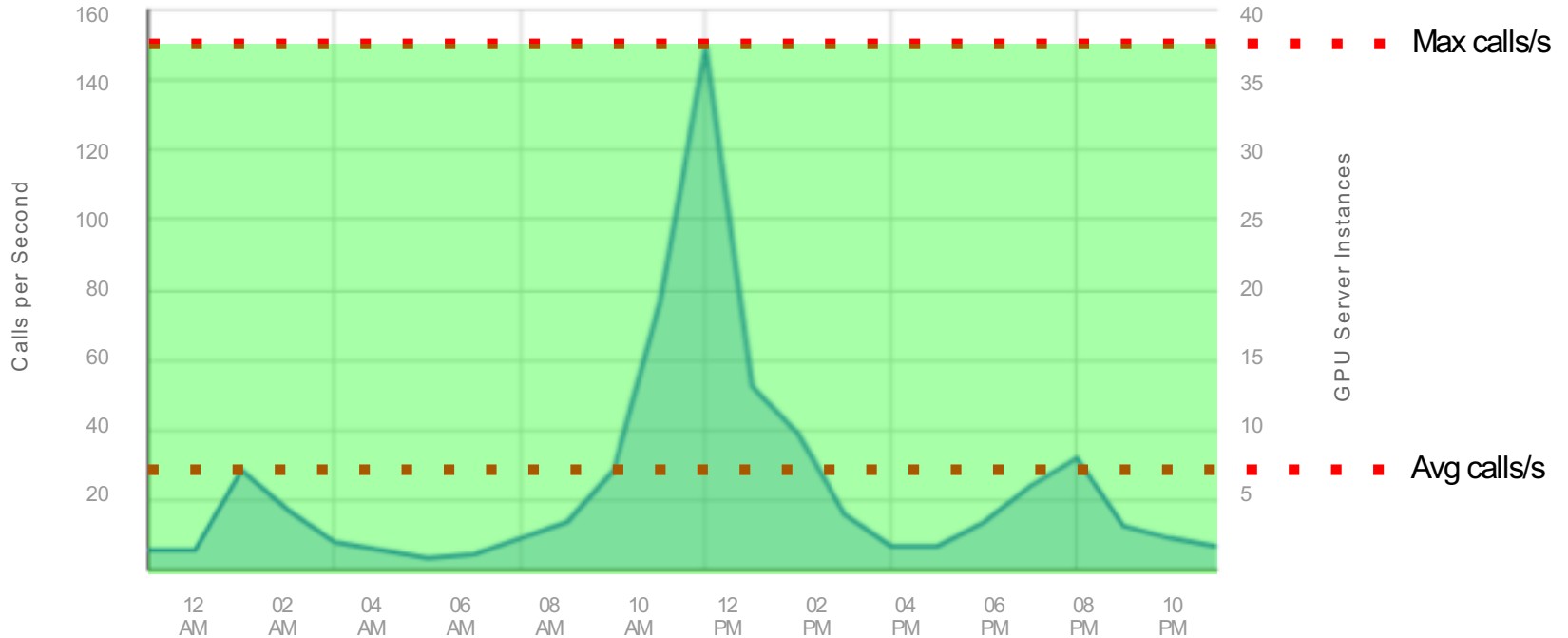
Why Serverless - Cost Efficiency

Jian Yang's "SeeFood" is most active during lunchtime.



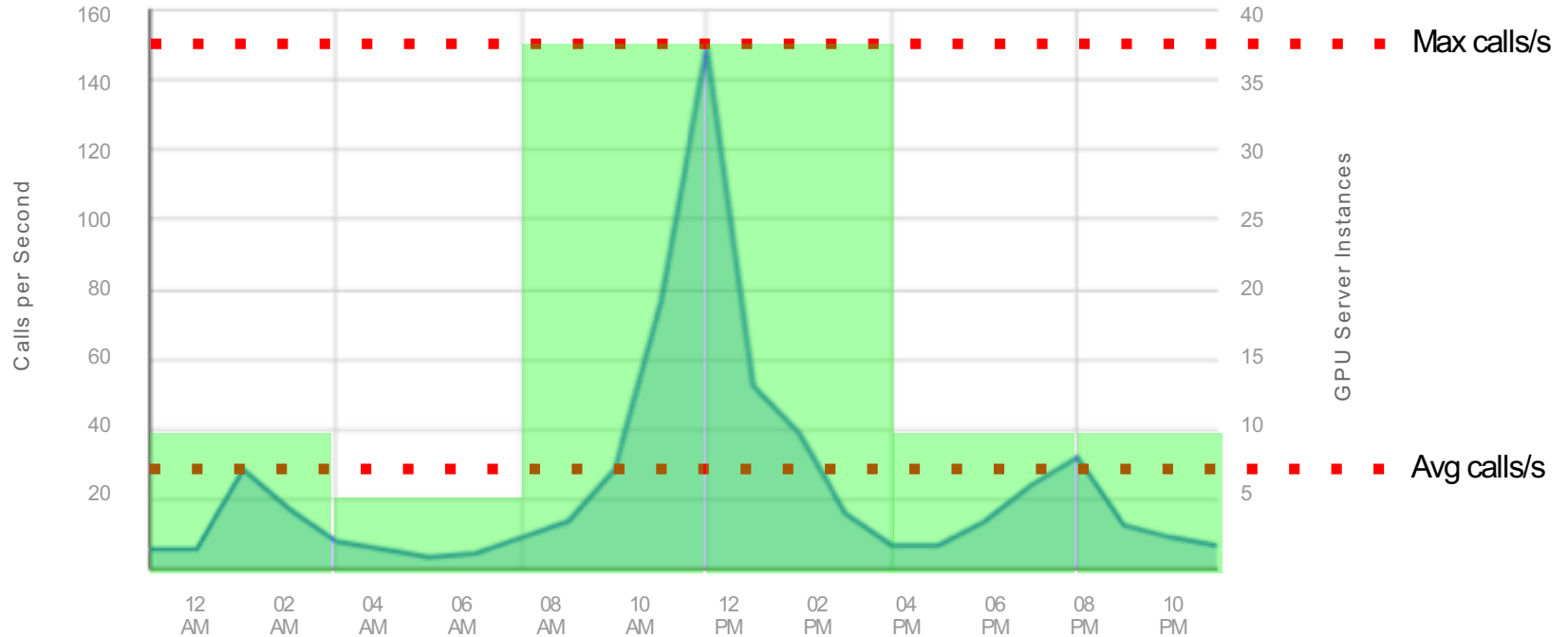
Traditional Architecture - Design for Maximum

40 machines 24 hours. $\$648 * 40 = \text{\$25,920}$ per month



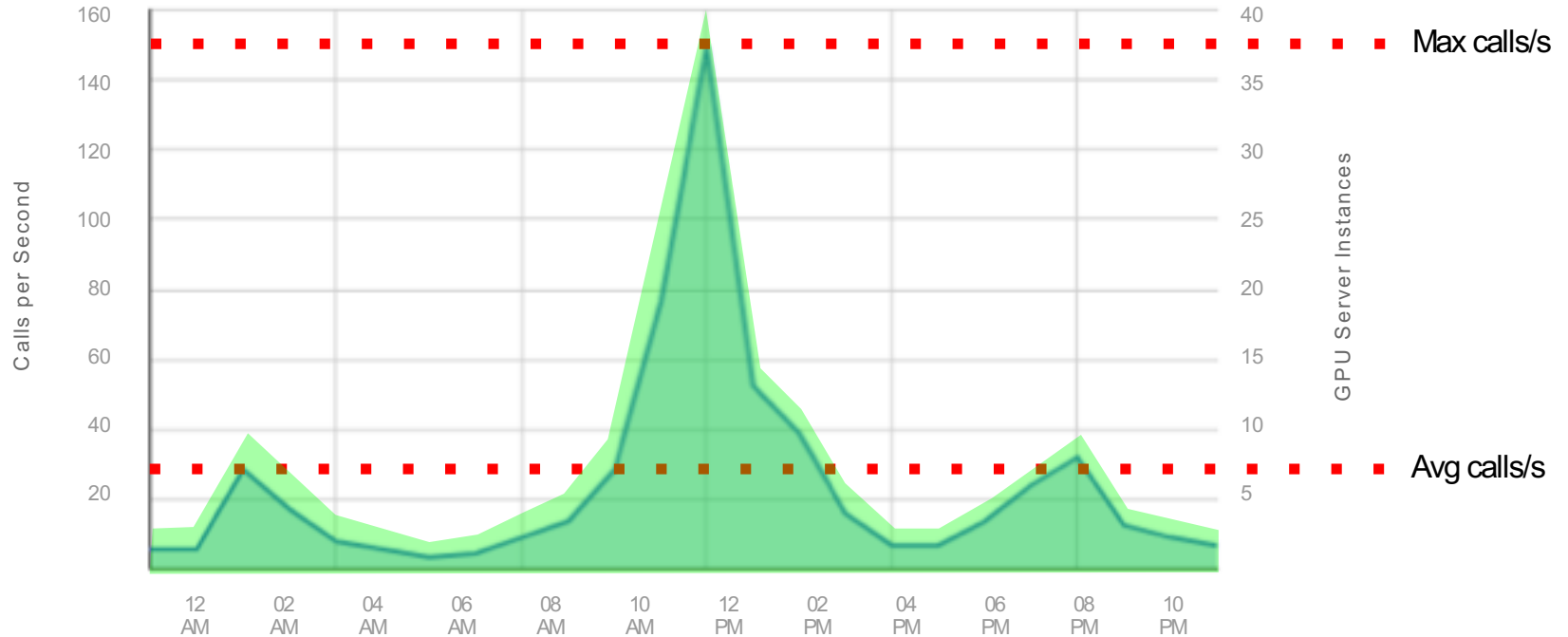
Autoscale Architecture - Design for Local Maximum

19 machines 24 hours. $\$648 * 40 = \$12,312$ per month

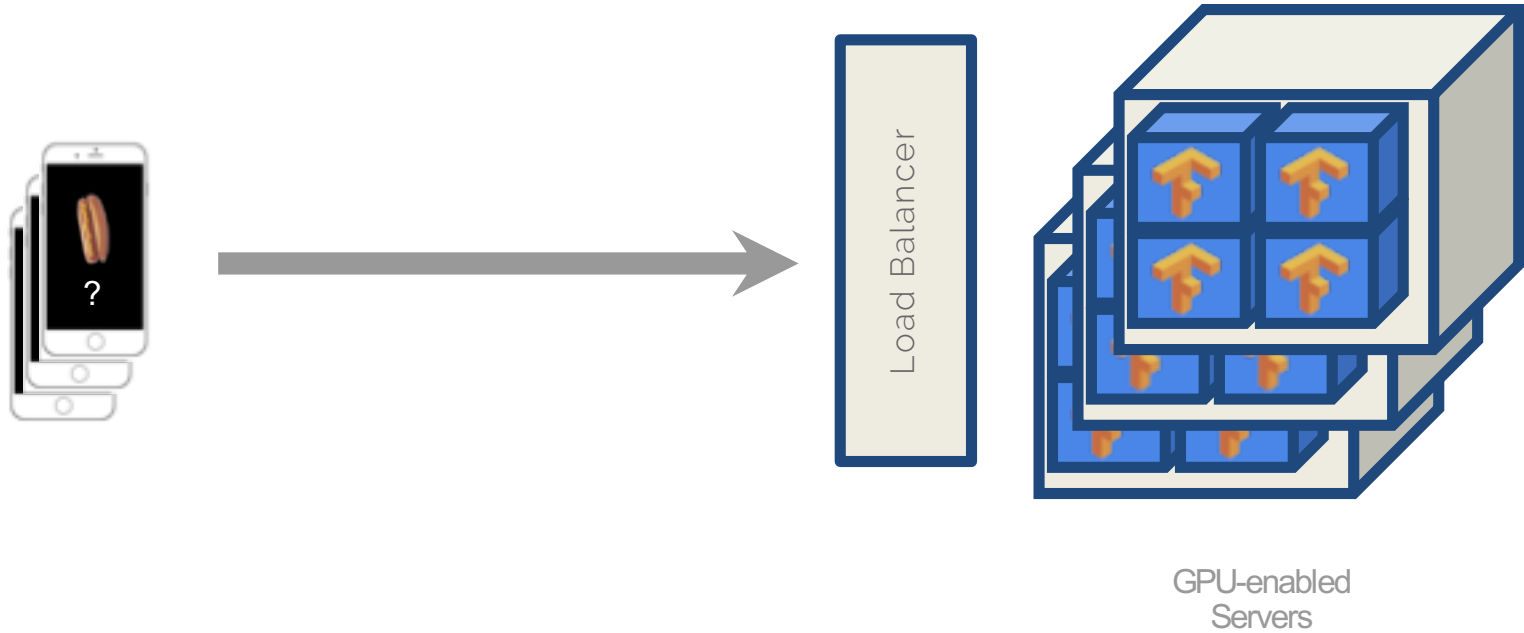


Serverless Architecture - Design for Minimum

Avg. of 21 calls / sec, or equivalent of 6 machines. $\$648 * 6 = \$3,888$ per month



Why Serverless - Concurrency



Why Serverless - Improved Latency

Portability = Low Latency

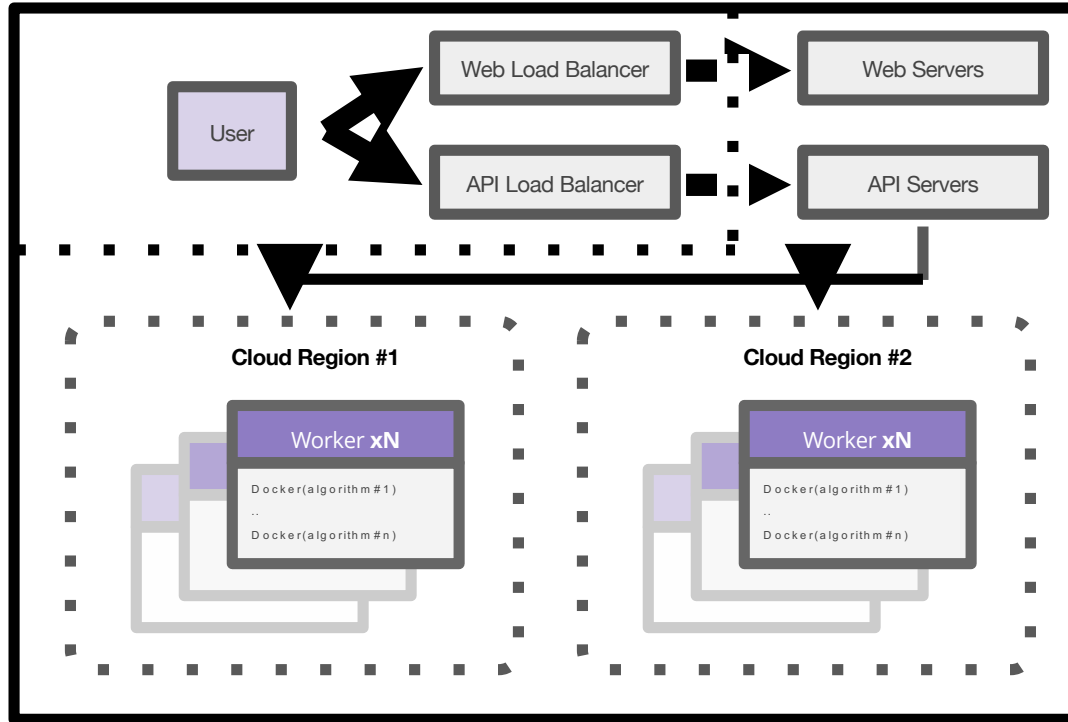




Almost there! We also need:

GPU Memory Management, Job Scheduling, Cloud Abstraction,
Discoverability, Authentication, Logging, etc.

Elastic Scale



Elastic Scaling with Intelligent Orchestration

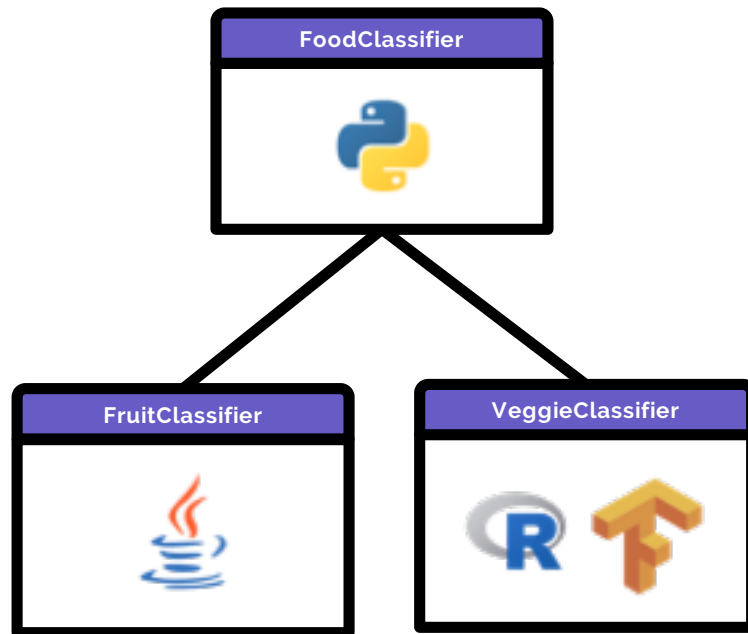
Knowing that:

- **Algorithm A** always calls **Algorithm B**
- Algorithm A consumes X CPU, X Memory, etc
- Algorithm B consumes X CPU, X Memory, etc

Therefore we can slot them in a way that:

- Reduce network latency
- Increase cluster utilization
- Build dependency graphs

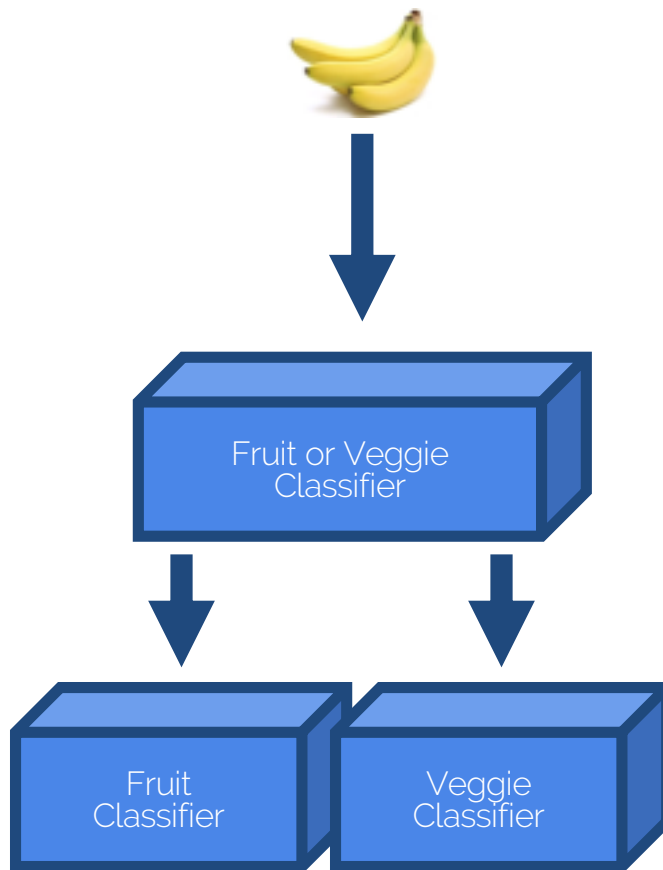
Runtime Abstraction



Composability

Composability is critical for AI workflows because of data processing pipelines and ensembles.

```
cat file.csv | grep foo | wc -l
```




Cloud Abstraction - Storage

No storage abstraction

```
s3      = boto3.client("s3")  
obj     = s3.get_object(Bucket="bucket-name", Key="records.csv")  
data    = obj["Body"].read()
```

With storage abstraction





```
data    = client.file("blob://records.csv").get()
```



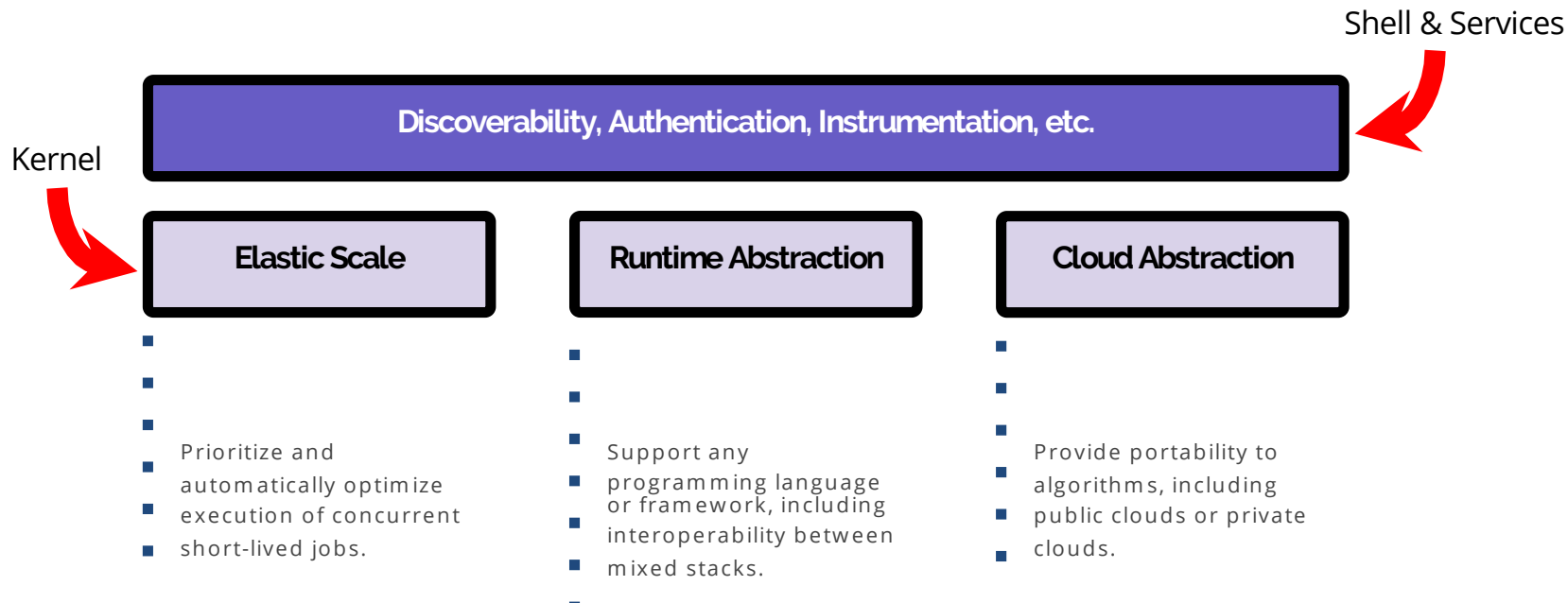
A diagram consisting of a horizontal purple line with a vertical purple line extending downwards from its center, pointing towards the abstraction box below.

```
s3://foo/bar  
blob://foo/bar  
hdfs://foo/bar  
dropbox://foo/bar  
etc.
```

Cloud Abstraction

				
Compute	EC2	CE	VM	Nova
Autoscaling	Autoscaling Group	Autoscaler	Scale Set	Heat Scaling Policy
Load Balancing	Elastic Load Balancer	Load Balancer	Load Balancer	LBaaS
Remote Storage	Elastic Block Store	Persistent Disk	File Storage	Block Storage

An Operating System for AI: the “AI Layer”




Discoverability: an App Store for AI

Categories **Top Charts**

Paid Free Top Grossing


1



Minecraft: Pocket Edition
Games
★★★★★ (3,538)

\$6.99
In-App Purchases


2



Heads Up!
Games
★★★★★ (86)

\$0.99
In-App Purchases


3



Facetune
Photo & Video
★★★★★ (1,810)

\$3.99


4



Toca Lab
Entertainment
★★★★☆ (117)

\$2.99

5



Enlight
Photo & Video
★★★★☆ (17)

\$3.99

 **Text Analysis**
Make sense of unstructured text

 **Machine Learning**
Teach your app to teach itself

 **Computer Vision**
Identify objects in images

Popular Tags:

Utilities Microservices Web Tools Time Series Sentiment Analysis

★ **Top Rated**

⌚ Most Called

⌚ Recently Added



Colorful Image Colorization

deeplearning/ColorfulImageColorization

Colorizes given black & white images.



Summarizer

nlp/Summarizer

Summarize english text



Sentiment Analysis

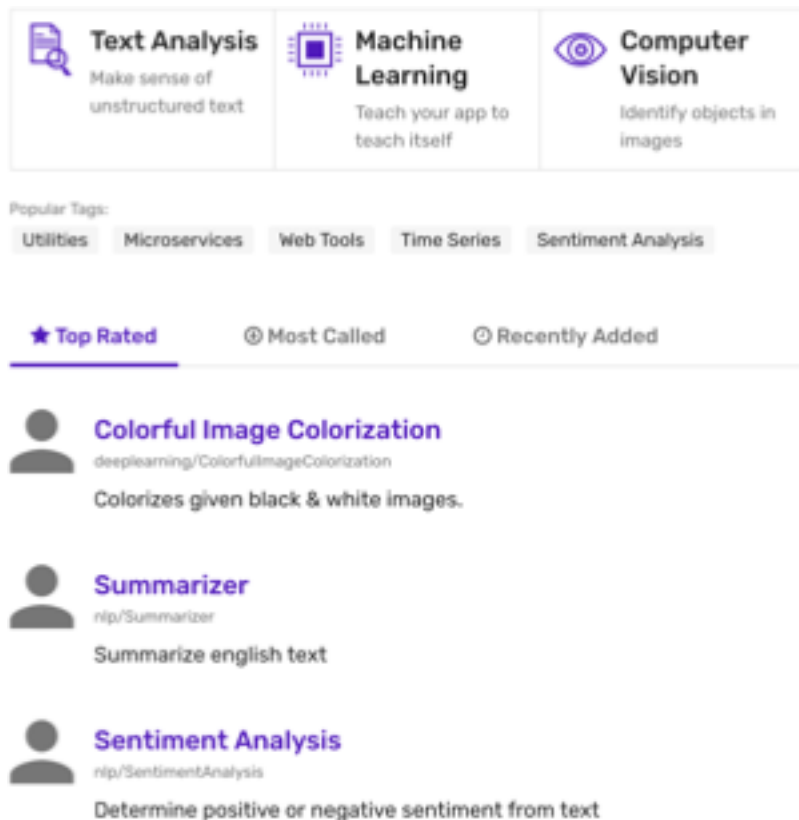
nlp/SentimentAnalysis

Determine positive or negative sentiment from text

Algorithmia's OS for AI: *discover* a model

1. Discover a model

- AppStore-like interface
- Categorized, tagged, rated
- Well-described
(purpose, source, API)



Algorithmia's OS for AI: *execute* a model

2. Execute from any language

- Raw JSON, or lang stubs
- Common syntax
- Autoscaled elastic cloud-exec
- Secure, isolated
- Concurrent, orchestrated
- 15ms overhead
- Hardware agnostic

Sentiment Analysis

Determine positive or negative sentiment from text

Royalty-free API calls 13,275,191

Tags

microservices nlp sentiment analysis stanford corenlp
text analysis

Permissions

Algorithmia Platform License • Internet Access • Calls Other Algorithms

Install & Use



```
import Algorithmia

input = {
  "document": "I really like Algorithmia!"
}
client = Algorithmia.client('API_KEY')
algo = client.algo('nlp/SentimentAnalysis/1.0.4')
print(algo.pipe(input))
```

OUTPUT

```
{
  "sentiment": 0.474,
  "document": "I really like
Algorithmia!"
}
```

Algorithmia's OS for AI: *add* a model

3. Add new models

- Many languages, frameworks
- Instant JSON API
- Call other models seamlessly (regardless of lang)
- Granular permissions
- GPU environments
- Namespaces & versioning

Create a new algorithm

Account or Organization

jpeck

Algorithm Name

StyleTransfer

Algorithm ID

algo://jpeck/StyleTransfer

Language

Java

Source Code

☒ Open Source ☐ Closed Source

Algorithmia Platform License

Special Permissions

☒ Requires full access to the internet

☐ No internet access required

☒ Can call other algorithms

☐ Not allowed to call other algorithms

☒ Advanced GPU

☐ Standard execution environment

Publishing jpeck/StyleTransfer version 0.1.0

Changes

Sample I/O

Versioning

Visibility

☒ Public (Anyone can call this version)

☐ Private (Only you can call this version)

Pricing [Help me understand pricing](#)

1 cr/call

You will receive 70% of the royalty cost: 0.70cr per call.

Semantic Versioning

What version should this be published as?

☐ 1.0.0 (Major: for API breaking changes)

☒ 0.1.0 (Minor: for backward-compatible features)

☐ 0.0.1 (Revision: n/a when changing price)

Thank you!

Jon Peck Developer Advocate

✉ jpeck@algorithmia.com

🐦 @peckjon

📄 **Algorithmia.com**

ALGORITHMIA

FREE STUFF

\$50 free at Algorithmia.com
signup code: **WOSC18**

WE ARE HIRING

algorithmia.com/jobs

- Seattle or Remote
- Bright, collaborative env
- Unlimited PTO
- Dog-friendly