# TensorFlow Eager execution on Backend.AI

## : Using TensorFlow Nightly build

Jeongkyu Shin

Nov. 21, 2017

# Index

Backend.AI  Open-Source PaaS 01

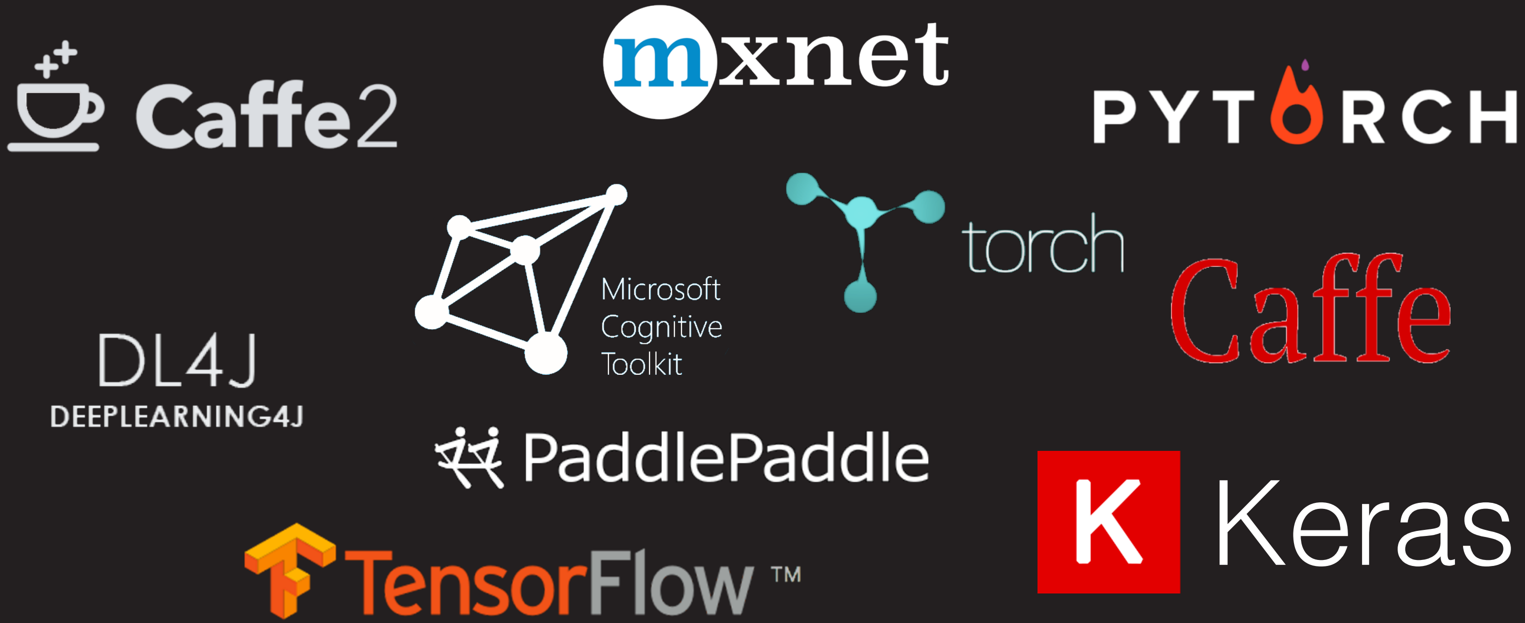
TensorFlow Eager Execution 02

Playing Eager Execution on Backend.AI 03

Closing 04

# Backend.AI OpenSource PaaS

# The era of Open-source ML frameworks



# Lablup.AI Platform

**Cloud** Pay-as-you-go

**PaaS** for **research, deep-learning model training** and ultra-convenient **coding education environment.**

**Ground** Bring Your Own Hardware

**Open-source edition** for deploying / developing your own Backend.AI Server Farm.

**Garden** Showcases

**Documents**, forum, **showcases** of Backend.AI platform.

**Backend.AI**

codeonweb.com

# Backend.AI Goal

## Easy

- Jupyter, VS Code, Atom, IntelliJ plugins
- Only need to set-up API key!

## Fast

- Combile container and GPU technology
- Make sure that the session takes less than a second

## Cheap

- Precise measurement of resource usage up to millisecond / KiB
- (Work In Progress!)

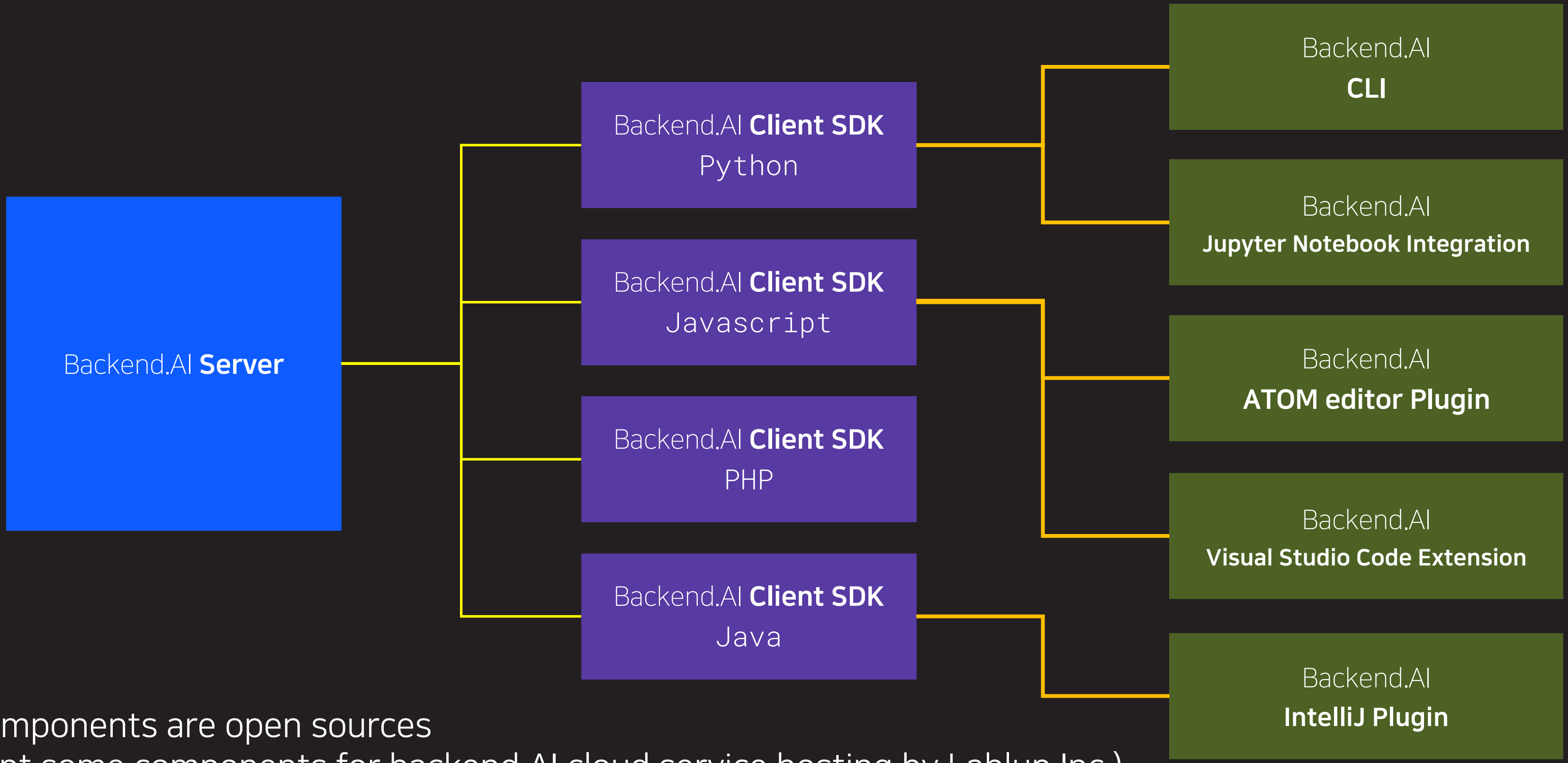
## Together

- Providing a language · version-specific virtual environment using containers
- Syscall sandboxing + enhanced Docker resource limit

## Everywhere

- Open-source version (github.com/lablup/backend.ai)
- Cloud service (cloud.backend.ai)

# Backend.AI Structure



Backend.AI **Server**

Backend.AI **Client SDK** Python

Backend.AI **Client SDK** Javascript

Backend.AI **Client SDK** PHP

Backend.AI **Client SDK** Java

Backend.AI **CLI**

Backend.AI **Jupyter Notebook Integration**

Backend.AI **ATOM editor Plugin**

Backend.AI **Visual Studio Code Extension**

Backend.AI **IntelliJ Plugin**

All components are open sources
(except some components for backend.AI cloud service hosting by Lablup Inc.)

# Backend.AI Core features

- Instant access
  - Available right after issuing API key
  - Creates a virtual environment immediately upon user request
- Handle various requirements
  - Supports all major programming languages and runtimes
    Python, R, Julia, Octave, PHP, Go, C/C++, Java, NodeJS, Lua, Haskell, Rust
  - Supports multiple versions of the same machine learning library
    TensorFlow, Caffe, PyTorch, Keras
- Developer-friendly framework
  - Integrated with familiar user experiences (Editor, IDE, Web notebooks)
    Jupyter, VS Code, Atom Editor, IntelliJ beta
  - `Provides $ backend.ai run` CLI, Cloud Interpreter·Compiler
  - HTTP-based public API (REST/GraphQL) and language-specific SDK
    Python, Javascript (Node.js), PHP beta

# Backend.AI Core Technology

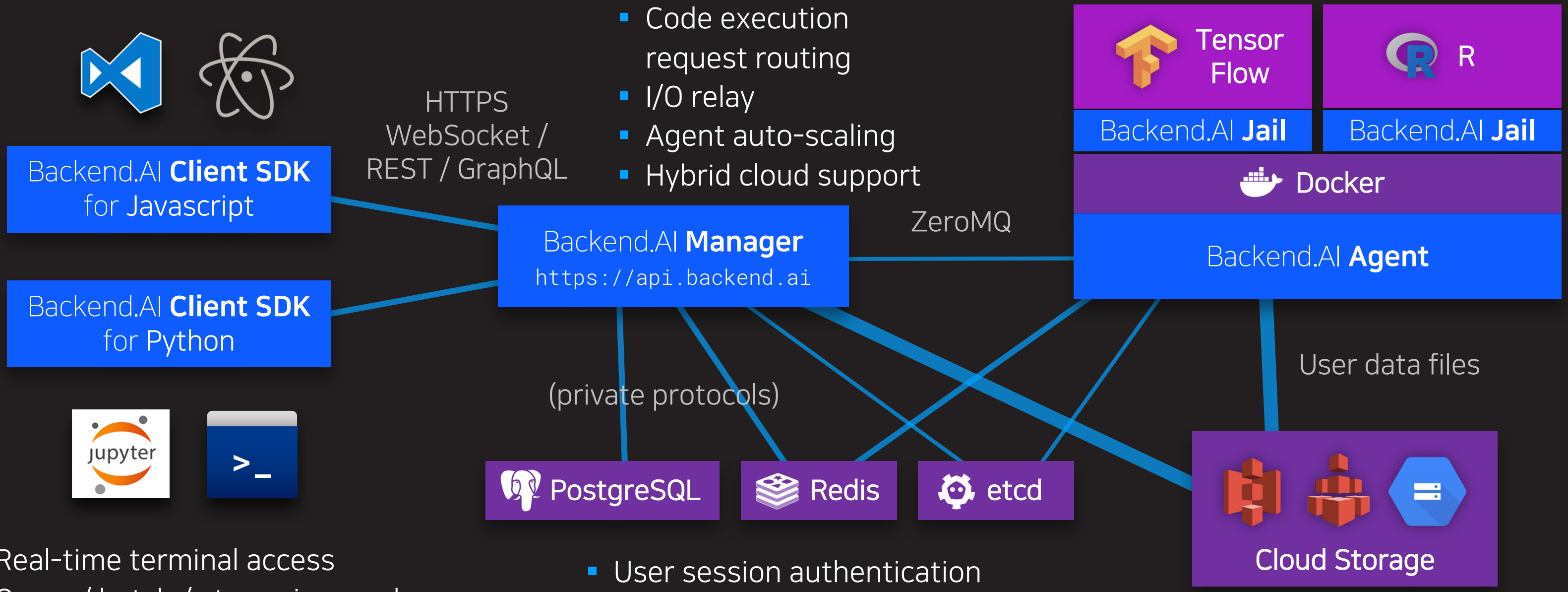
- High performance
  - Asynchronous-based Low latency & high density container pooling (Python asyncio + Docker)
  - GPU-acceleration support for multi-tenant environments (nvidia-docker)
- Security for multi-user / enterprise environments
  - Dynamic sandboxing: programmable syscall filter & rewriting
  - Enforcing resource constraints for legacy apps in Docker
- Hyper-scaling
  - On-premise private cluster
  - Hybrid cloud (on-premise + public cloud)
  - Public cloud (AWS + MS Azure + Google Cloud combination)
  - Route request to specific cloud, based on calculation type, load, or settings

# Language & Frameworks

| System | Languages | Frameworks | Developed Framework |
|---|---|---|---|
| Backend.AI **Ground** | Python 3.6 | aiodocker<br>aiozmq<br>alembic<br>pyzmq<br>SQLAlchemy<br>nvidia-docker | Backend.AI-manager<br>Backend.AI-agent |
| | Go | | Backend.AI-Jail |
| | ETC. | Docker 17.9 | |
| Backend.AI **Cloud**<br>CodeOnWeb.com | Python 3.6 | Django 1.11 | Customized Django<br>Site-independent overloading |
| | Javascript ES6 / HTML5 | Polymer 2.2 | Lablup-webcomponents |

# Backend.AI Structure (detail)

- Programmable system call sandbox
- Limit the number of cores, memory, and other container resources



- Code execution request routing
- I/O relay
- Agent auto-scaling
- Hybrid cloud support

Backend.AI **Client SDK** for Javascript

Backend.AI **Client SDK** for Python

HTTPS
WebSocket /
REST / GraphQL

Backend.AI **Manager**
`https://api.backend.ai`

ZeroMQ

Tensor Flow

R

Backend.AI **Jail**

Backend.AI **Jail**

Docker

Backend.AI **Agent**

User data files

(private protocols)

PostgreSQL

Redis

etcd

Cloud Storage

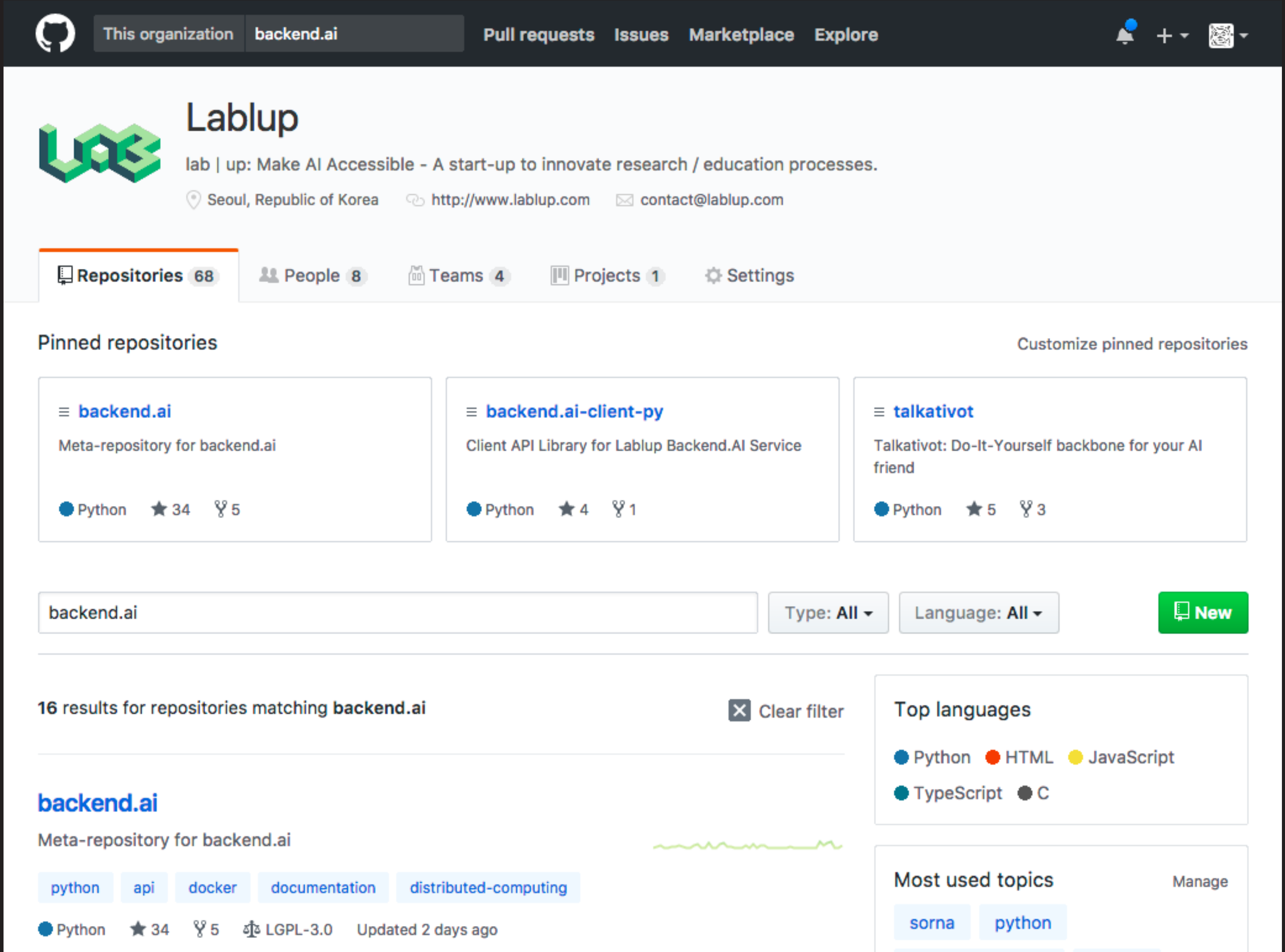
- Real-time terminal access
- Query / batch / streaming mode
- View usage and session status
- Multimedia output rendering

- User session authentication
- Resource usage statistics by session in real-time
- Automatic rolling upgrade

- Virtual folders per user
- Share·Example dataset

# Backend.AI Open-Source Project

- Communication
  - Slack → Synology Chat (+Synology CloudStation) → Microsoft Teams (+OneNote+OneDrive)
- Issue tracker
  - GitHub + Microsoft Teams
- Autodeploy
  - Lablup.AI : GitHub + post-commit signal to Azure → Autobuild on Azure webapp
  - CodeOnWeb: manual via script execution (to keep users safe)
- CI
  - Travis CI + Datadog
- Error reporting
  - Sentry + Datadog



Lablup

lab | up: Make AI Accessible - A start-up to innovate research / education processes.

Seoul, Republic of Korea http://www.lablup.com contact@lablup.com

Pinned repositories

| backend.ai | backend.ai-client-py | talkativot |
|---|---|---|
| Meta-repository for backend.ai | Client API Library for Lablup Backend.AI Service | Talkativot: Do-It-Yourself backbone for your AI friend |
| ● Python ★ 34 5 | ● Python ★ 4 1 | ● Python ★ 5 3 |

Search field: backend.ai

**16** results for repositories matching **backend.ai**

Top languages

● Python ● HTML ● JavaScript ● TypeScript ● C

backend.ai

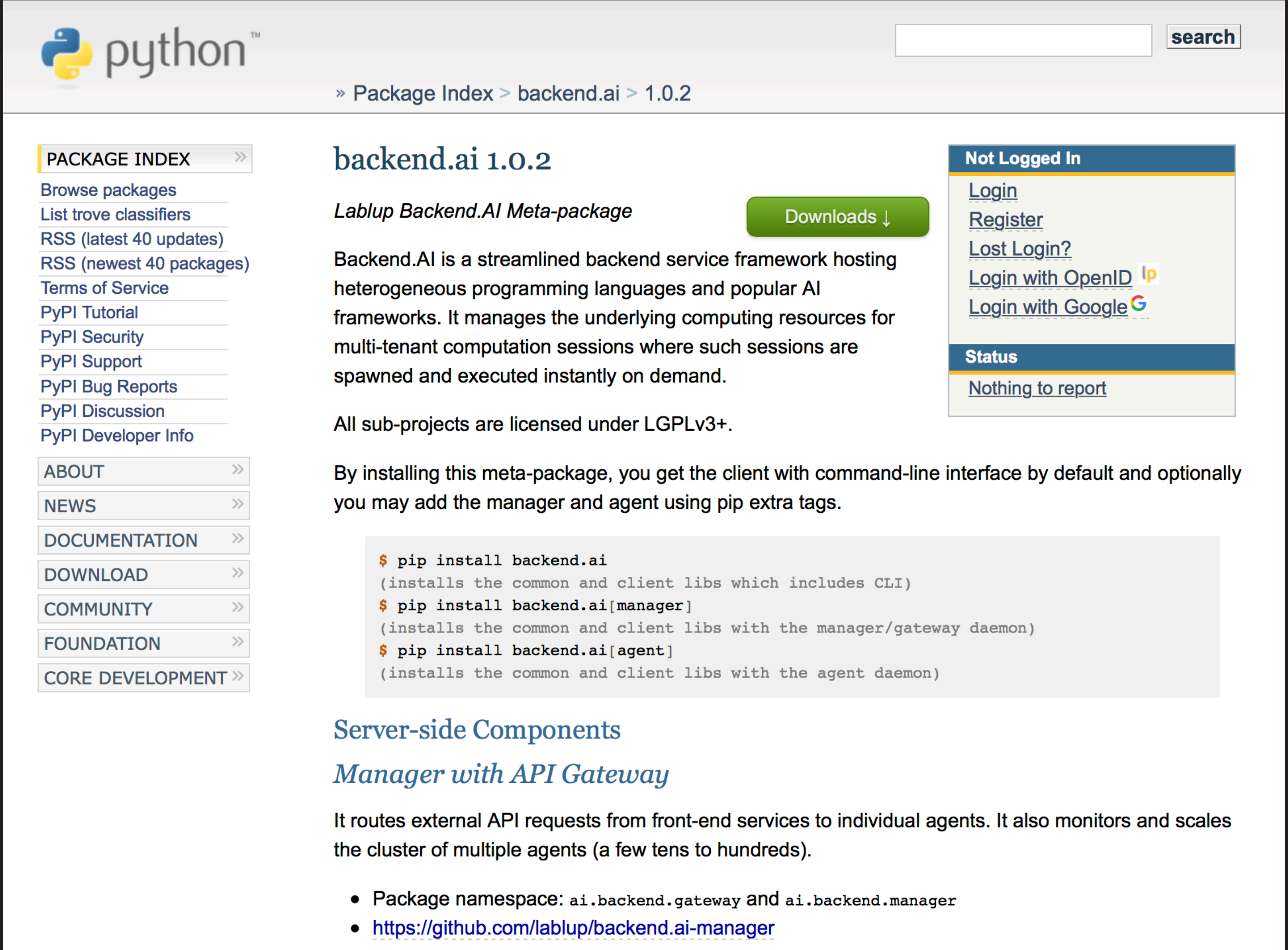
Meta-repository for backend.ai

python api docker documentation distributed-computing

● Python ★ 34 5 LGPL-3.0 Updated 2 days ago

Most used topics

sorna python



# backend.ai 1.0.2

*Lablup Backend.AI Meta-package*

Backend.AI is a streamlined backend service framework hosting heterogeneous programming languages and popular AI frameworks. It manages the underlying computing resources for multi-tenant computation sessions where such sessions are spawned and executed instantly on demand.

All sub-projects are licensed under LGPLv3+.

By installing this meta-package, you get the client with command-line interface by default and optionally you may add the manager and agent using pip extra tags.

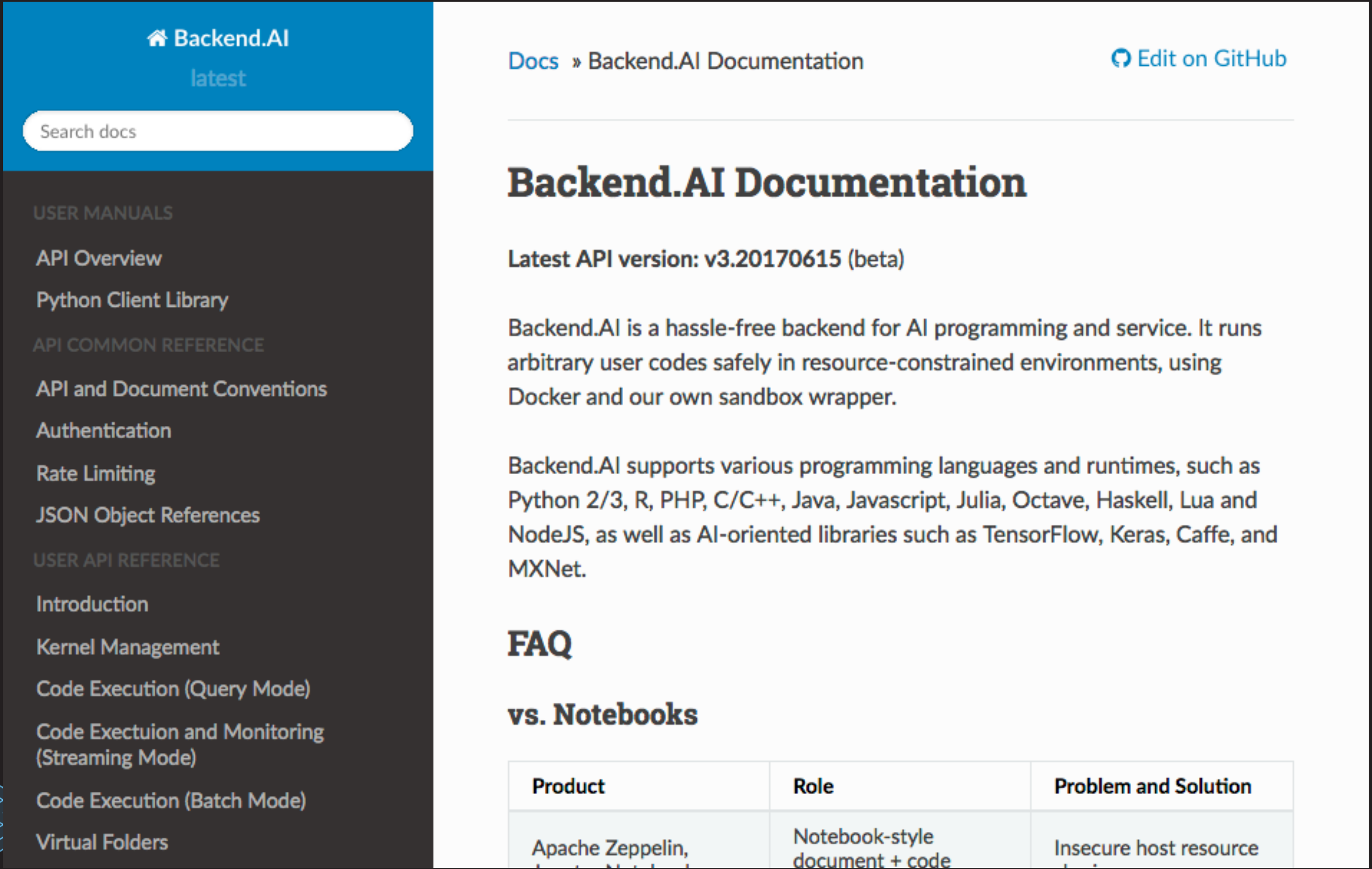
```
$ pip install backend.ai
(installs the common and client libs which includes CLI)
$ pip install backend.ai[manager]
(installs the common and client libs with the manager/gateway daemon)
$ pip install backend.ai[agent]
(installs the common and client libs with the agent daemon)
```

## Server-side Components

### *Manager with API Gateway*

It routes external API requests from front-end services to individual agents. It also monitors and scales the cluster of multiple agents (a few tens to hundreds).

- Package namespace: `ai.backend.gateway` and `ai.backend.manager`
- https://github.com/lablup/backend.ai-manager



# Backend.AI Documentation

**Latest API version: v3.20170615** (beta)

Backend.AI is a hassle-free backend for AI programming and service. It runs arbitrary user codes safely in resource-constrained environments, using Docker and our own sandbox wrapper.

Backend.AI supports various programming languages and runtimes, such as Python 2/3, R, PHP, C/C++, Java, Javascript, Julia, Octave, Haskell, Lua and NodeJS, as well as AI-oriented libraries such as TensorFlow, Keras, Caffe, and MXNet.

## FAQ

### vs. Notebooks

| Product | Role | Problem and Solution |
|---|---|---|
| Apache Zeppelin, | Notebook-style document + code | Insecure host resource |

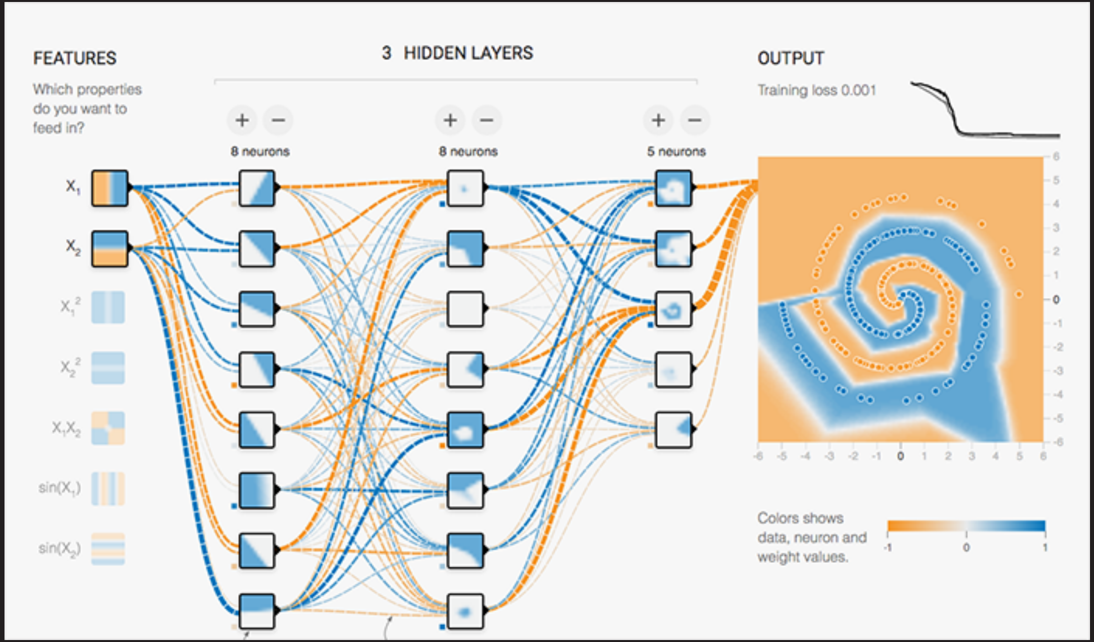
# TensorFlow Eager Execution

# TensorFlow: Summary

- Statistics
  - More than 24000 commits since Dec. 2015
  - More than 1140 committers
  - More than 6400 TensorFlow-related repository created on GitHub
- Current
  - Complete ML model prototyping
  - Distributed training
  - CPU / GPU / TPU / Mobile support
- TensorFlow Serving
  - Enables easier inference / model serving
- XLA compiler (1.0~)
  - Support various environments / speedups
- Keras API Support (1.2~)
  - High-level programming API
  - Keras-compatible API
- Eager Execution (1.4~)
  - Interactive mode of TensorFlow
  - Treat TensorFlow python code as real python code

# How TensorFlow works

- CPU
  - Multiprocessor
    - ✓ AVX-based acceleration
    - ✓ GPU part in chip
  - OpenMP
- GPU
  - CUDA (NVidia) ➜ cuDNN
  - OpenAL (AMD)
- TPU (1st, 2nd gen.)
  - ASIC for accelerating matrix calculation
  - In-house development by Google



FEATURES

Which properties do you want to feed in?

3 HIDDEN LAYERS

OUTPUT

Training loss 0.001

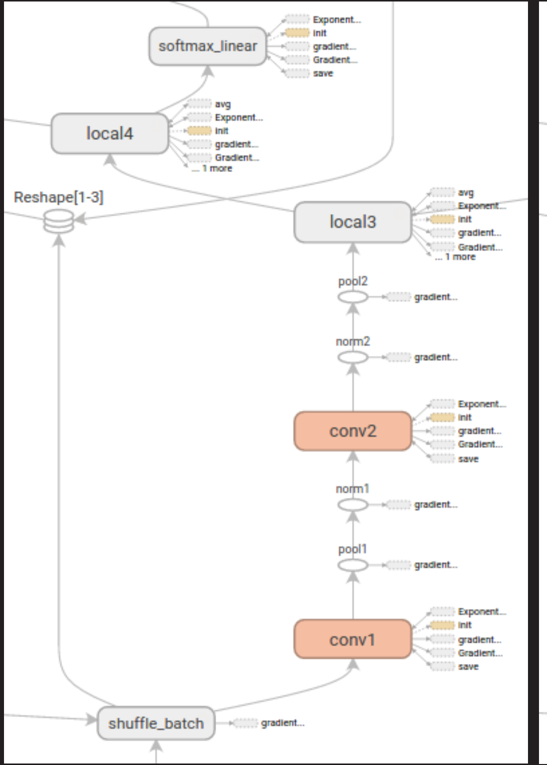
Colors shows data, neuron and weight values.

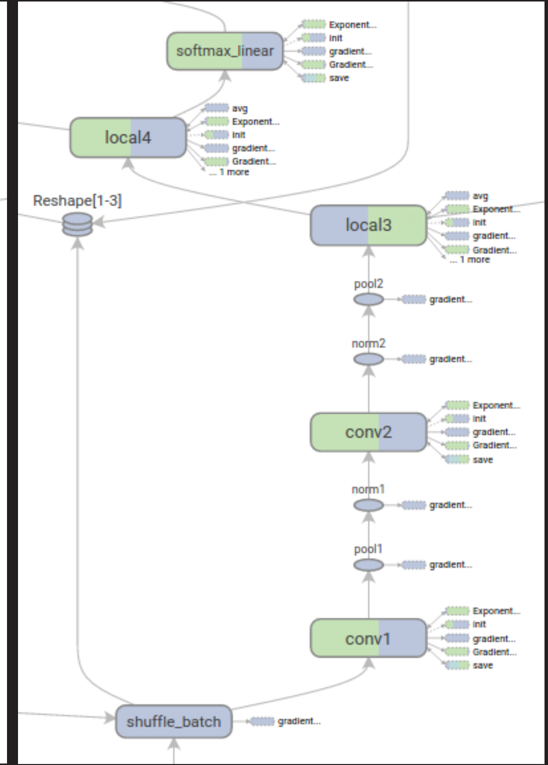
https://www.tensorflow.org/get_started/graph_viz

# How TensorFlow works

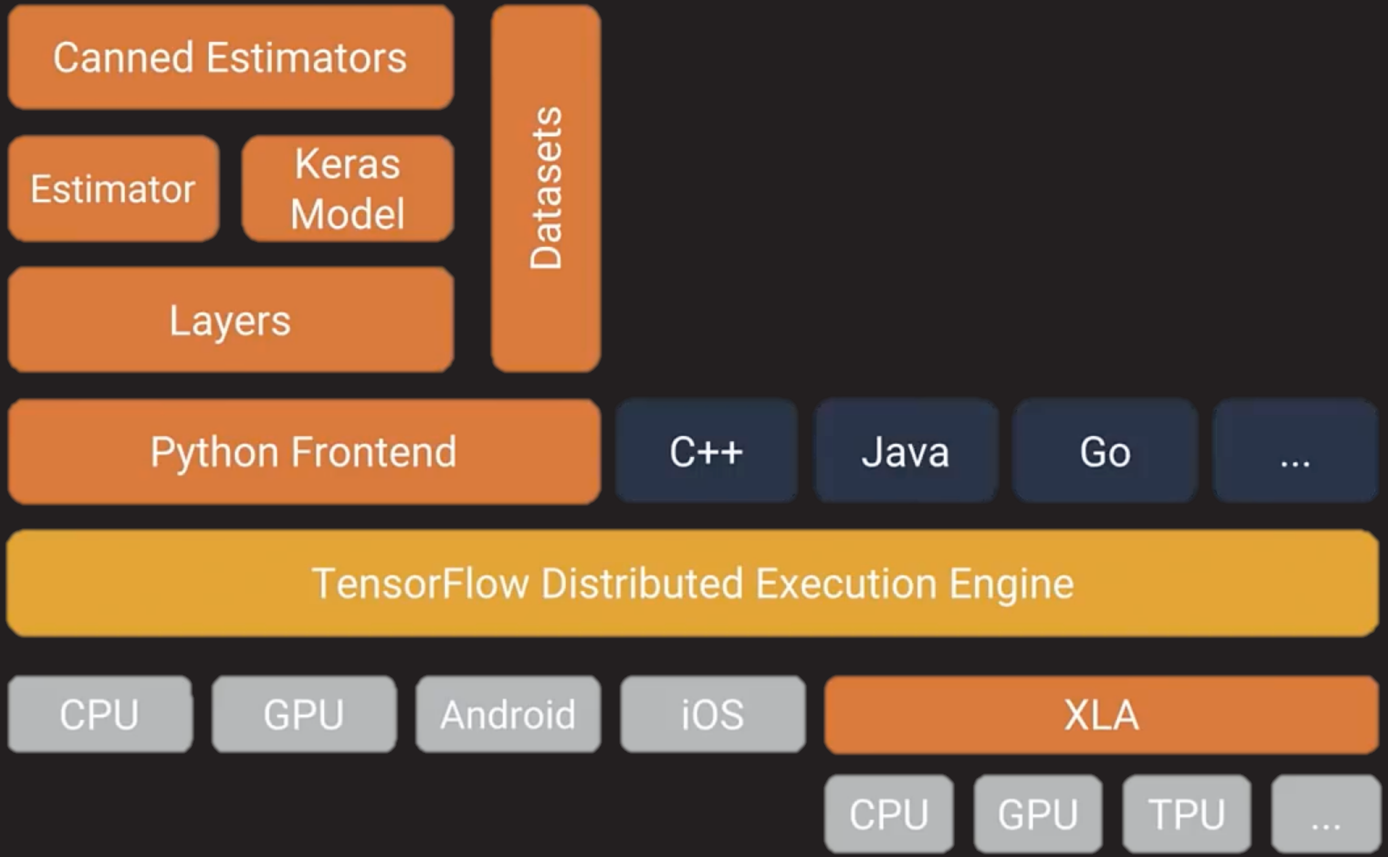
- Python but not Python
  - Python API is default API for TensorFlow
  - However, TF core is written in C++, with CuDNN library (for GPU acceleration)

- Computation Graph
  - **User TF code is not a code**
    - ✓ it is a configuration to generate computation graph
  - Session
    - ✓ Creates a computation graph and run the training using C++ core
  - Tedious debug process





# How TensorFlow works



Google I/O 2017 / TensorFlow Frontiers

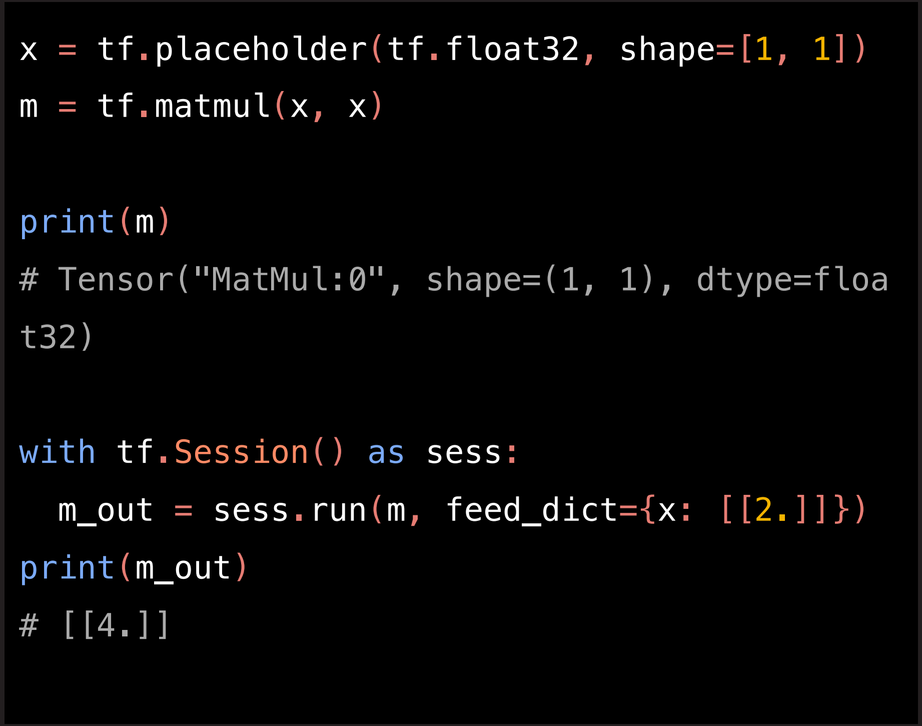
# Eager execution

- Announced at Oct. 30, 2017
- Makes TensorFlow execute operations immediately
  - Returns concrete values
- Provides
  - A NumPy-like library for numerical computation
  - Support for GPU acceleration and automatic differentiation
  - A flexible platform for machine learning research and experiments
- Advantages
  - Python debugger tools
  - Immediate error reporting
  - Easy control flow
  - Python data structures

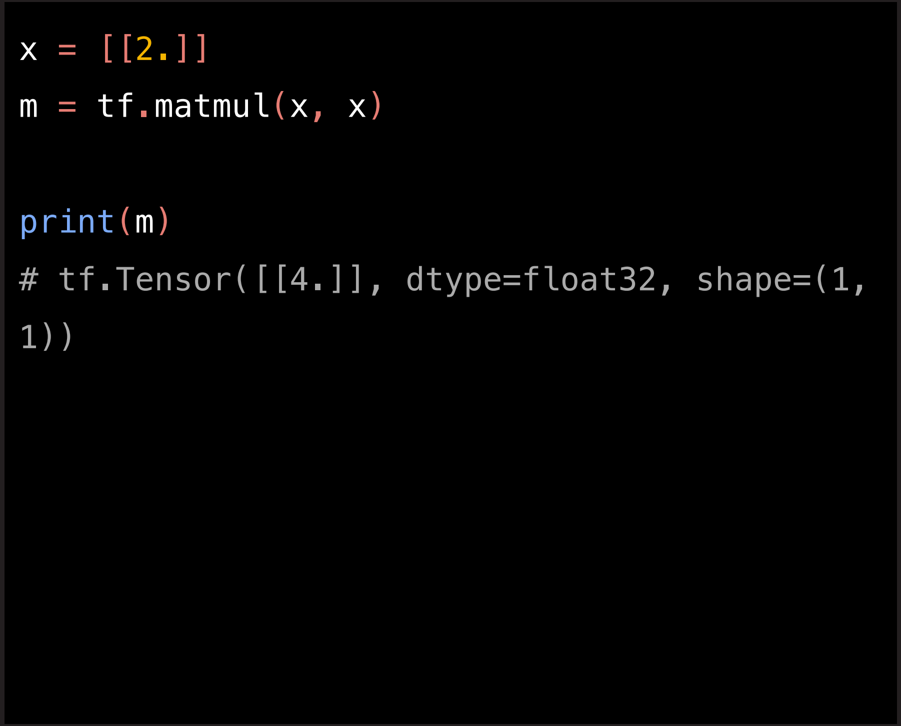
# Playing Eager Execution On Backend.AI

# Setting up Backend.AI

# Eager execution: Session

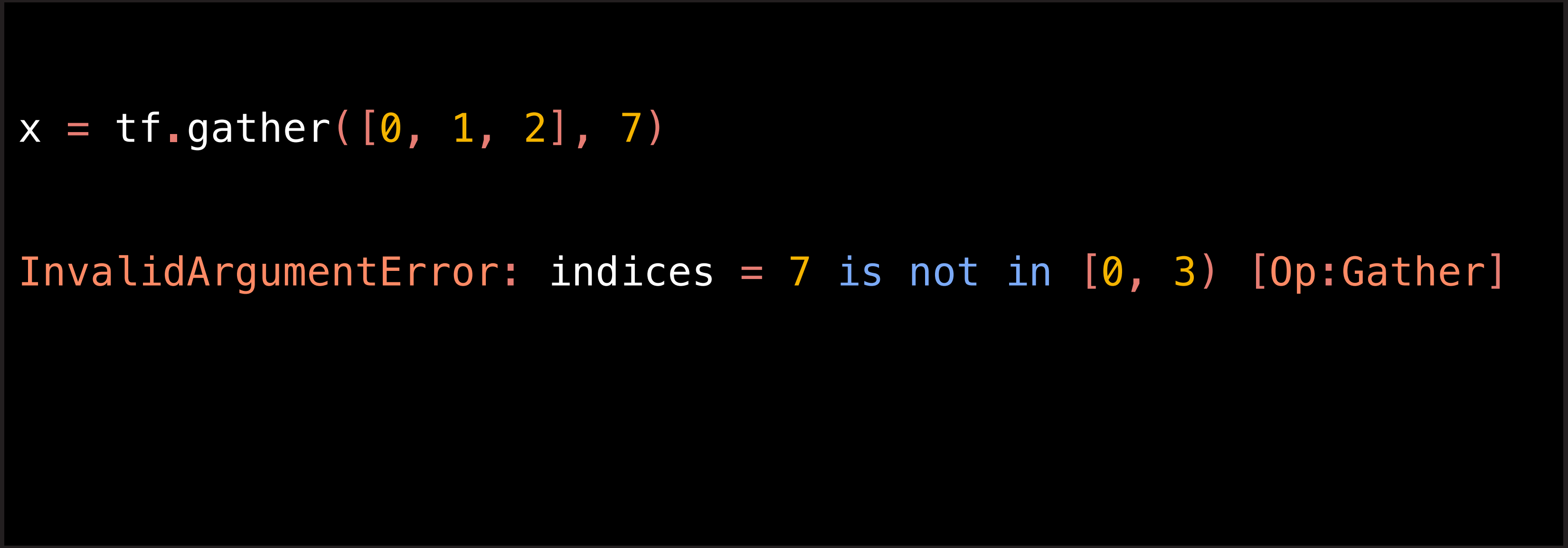


```python
x = tf.placeholder(tf.float32, shape=[1, 1])
m = tf.matmul(x, x)

print(m)
# Tensor("MatMul:0", shape=(1, 1), dtype=floa
t32)

with tf.Session() as sess:
  m_out = sess.run(m, feed_dict={x: [[2.]]})
print(m_out)
# [[4.]]
```



```python
x = [[2.]]
m = tf.matmul(x, x)

print(m)
# tf.Tensor([[4.]], dtype=float32, shape=(1,
1))
```

# Eager execution: Instant error



```
x = tf.gather([0, 1, 2], 7)

InvalidArgumentError: indices = 7 is not in [0, 3) [Op:Gather]
```

# Eager execution: removing metaprogramming

```python
x = tf.random_uniform([2, 2])

with tf.Session() as sess:
  for i in range(x.shape[0]):
    for j in range(x.shape[1]):
      print(sess.run(x[i, j]))
```

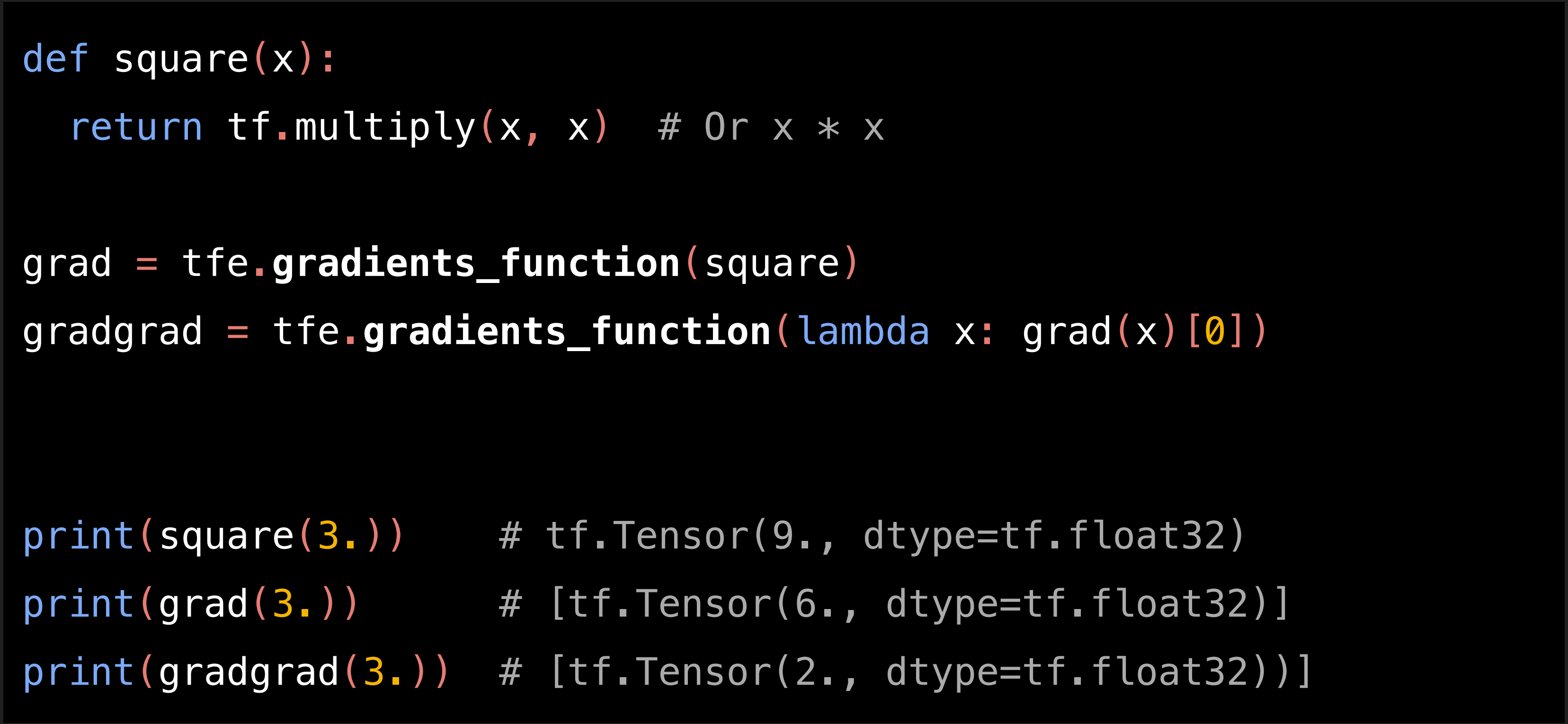
```python
x = tf.random_uniform([2, 2])

for i in range(x.shape[0]):
  for j in range(x.shape[1]):
    print(x[i, j])
```

# Eager execution: Python Control Flow

```python
a = tf.constant(6)
while not tf.equal(a, 1):
  if tf.equal(a % 2, 0):
    a = a / 2
  else:
    a = 3 * a + 1
  print(a)
```

```
# Outputs
tf.Tensor(3, dtype=int32)
tf.Tensor(10, dtype=int32)
tf.Tensor(5, dtype=int32)
tf.Tensor(16, dtype=int32)
tf.Tensor(8, dtype=int32)
tf.Tensor(4, dtype=int32)
tf.Tensor(2, dtype=int32)
tf.Tensor(1, dtype=int32)
```

# Eager execution: Gradients



```python
def square(x):
  return tf.multiply(x, x)  # Or x * x

grad = tfe.gradients_function(square)
gradgrad = tfe.gradients_function(lambda x: grad(x)[0])



print(square(3.))    # tf.Tensor(9., dtype=tf.float32)
print(grad(3.))      # [tf.Tensor(6., dtype=tf.float32)]
print(gradgrad(3.))  # [tf.Tensor(2., dtype=tf.float32))]
```

# Eager execution: Custom Gradients

```
def log1pexp(x):
  return tf.log(1 + tf.exp(x))
grad_log1pexp = tfe.gradients_function(log1pexp)

print(grad_log1pexp(0.))
```

*Works fine, prints [0.5]*

# Eager execution: Custom Gradients

```
def log1pexp(x):
  return tf.log(1 + tf.exp(x))
grad_log1pexp = tfe.gradients_function(log1pexp)

print(grad_log1pexp(100.))
```

*[nan] due to numeric instability*

# Eager execution: Custom Gradients

```
@tfe.custom_gradient
def log1pexp(x):
  e = tf.exp(x)
  def grad(dy):
    return dy * (1 - 1 / (1 + e))
  return tf.log(1 + e), grad
grad_log1pexp = tfe.gradients_function(log1pexp)

# Gradient at x = 0 works as before.
print(grad_log1pexp(0.))   # [0.5]
# And now gradient computation at x=100 works as well.
print(grad_log1pexp(100.))  # [1.0]
```

# Eager execution: Using GPUs

`tf.device()` for manual placement

```
with tf.device(“/gpu:0”):
  x = tf.random_uniform([10, 10])
  y = tf.matmul(x, x)
  # x and y reside in GPU memory
```

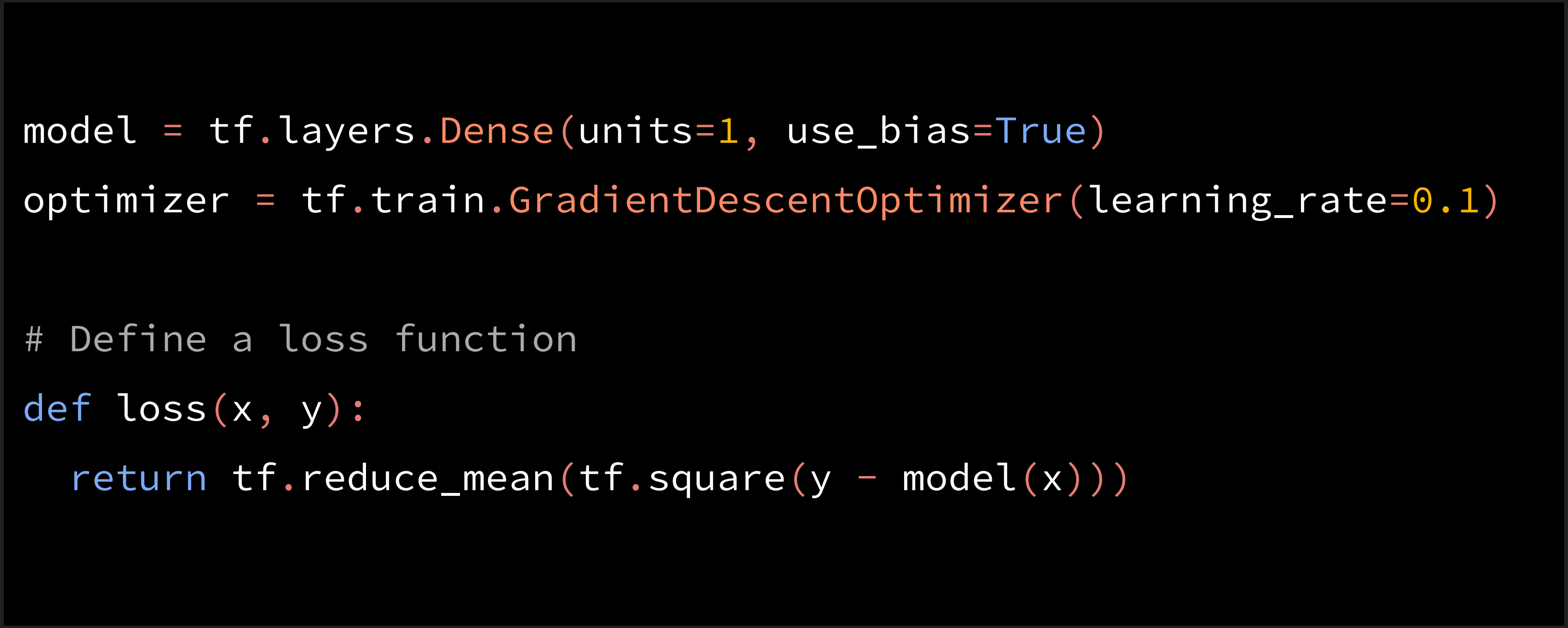
# Eager execution: Building Models

The same APIs as graph building

(`tf.layers, tf.train.Optimizer, tf.data` etc.)

```
model = tf.layers.Dense(units=1, use_bias=True)
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
```

# Eager execution: Building Models



```
model = tf.layers.Dense(units=1, use_bias=True)
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)

# Define a loss function
def loss(x, y):
  return tf.reduce_mean(tf.square(y - model(x)))
```

# Eager execution: Training Models

## Compute and apply gradients

```
grad_fn = tfe.implicit_gradients(loss)

for (x, y) in get_next_batch():
  optimizer.apply_gradients(grad_fn(x, y))
```

# Status

- Alpha/Preview version out now!
- Single GPU, ResNet benchmark performance comparable to graphs
- Overheads on smaller operations is high
- Watch the release notes for upcoming TensorFlow releases for updates

# Backend.AI: How to join?

- https://github.com/lablup/backend.ai
  - We look forward to participating in Backend.AI development!
    - ✓ Would it be better if you come up with a pick? 🤗
  - V1.0 release! (Oct. 2017)
    - ✓ Installation and development manual provided! (Finally!)

- Future roadmap
  - Scheduler enhancements
    - ✓ Hybrid cloud - on-premise binding
  - Enhanced auto-scaling
    - ✓ Scale-in protection for long-time execution sessions
    - ✓ Cold/hot instance group management, depending on available cpu/memory/gpu slot capacity

# THANK YOU

If you have question, please contact via contact@lablup.com !