redisconf19

Serving Deep Learning Models at Scale with RedisAI

Luca Antiga
[tensor]werk, *CEO*

presented by
redislabs
HOME OF REDIS

Agenda:

1. Don’t say AI until you productionize

   Deep learning and challenges for production

2. Introducing RedisAI + Roadmap

   Architecture, API, operation, what’s next

3. Demo

   Live notebook by Chris Fragly, Pipeline.ai

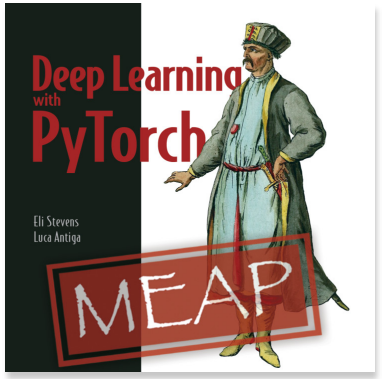
Who am I (@lantiga)

- Co-founder and CEO of **[tensor]werk**
  Infrastructure for data-defined software
- Co-founder of **Orobix**
  AI in healthcare/manufacturing/gaming/+
- **PyTorch** contributor in 2017-2018
- Co-author of **Deep Learning with PyTorch**, Manning (with **Eli Stevens**)

[tensor]werk

OROBIX
TOOLS WITH A MIND

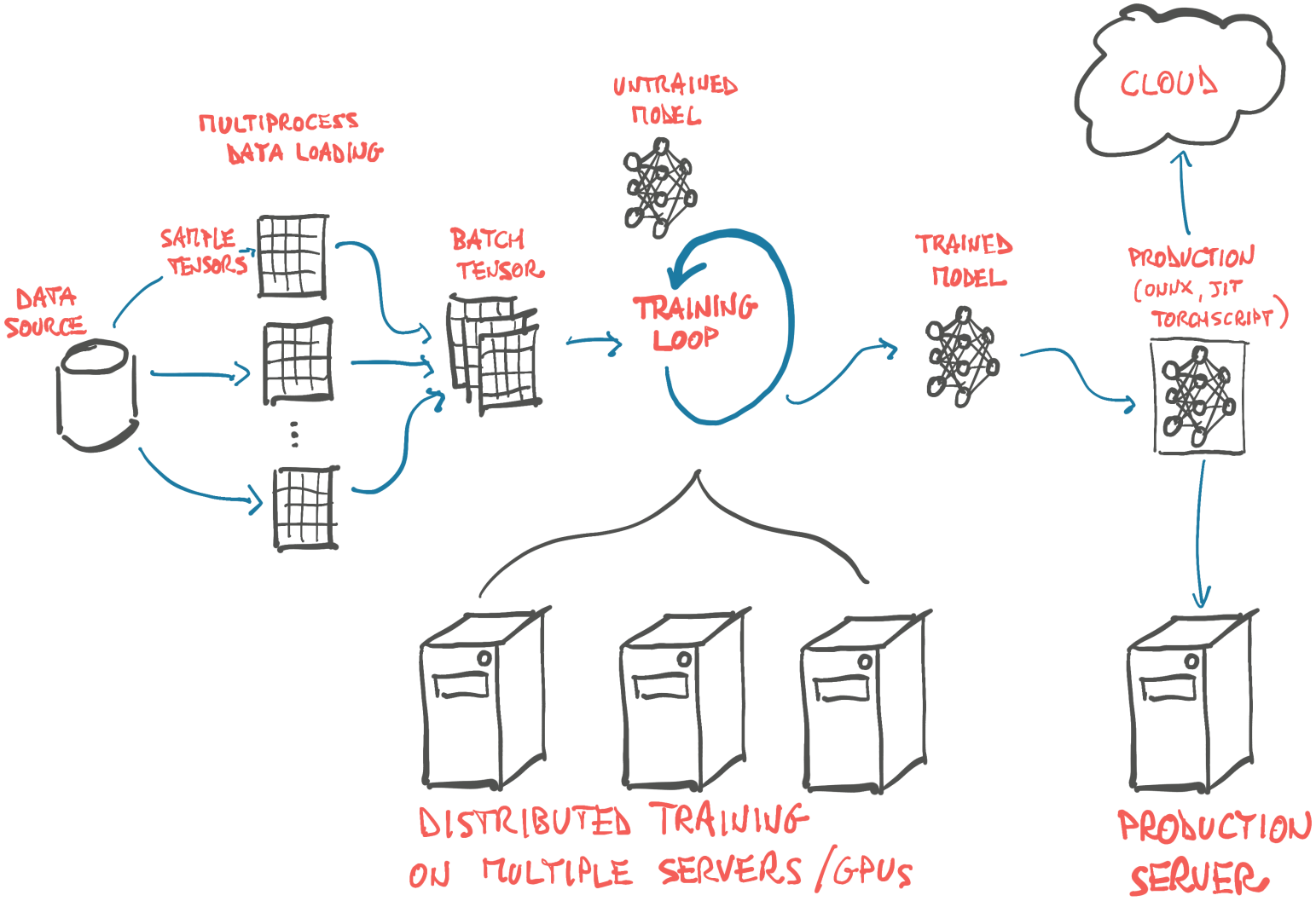
PyTorch



Don’t say AI until you productionize

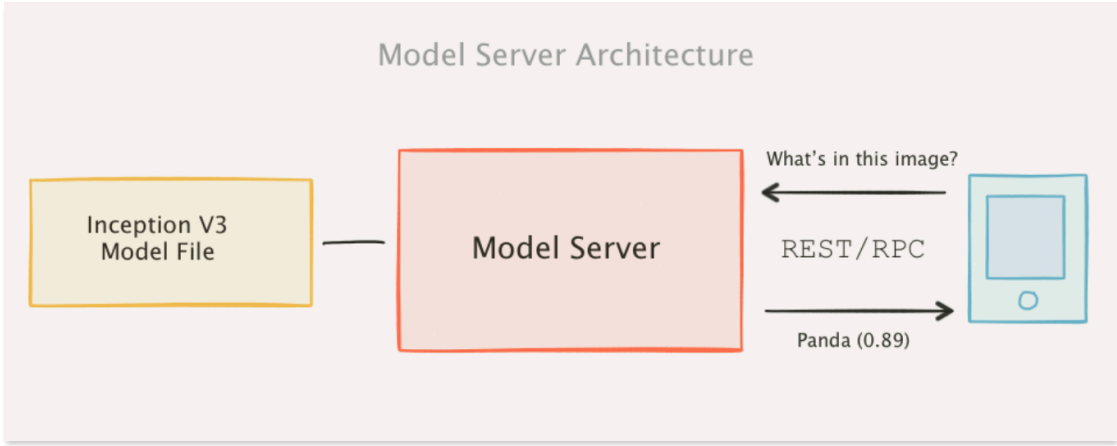
Deep learning frameworks

- TensorFlow
- PyTorch
- mxnet
- CNTK, Chainer,
- DyNet, DL4J,
- Flux, …



Production strategies

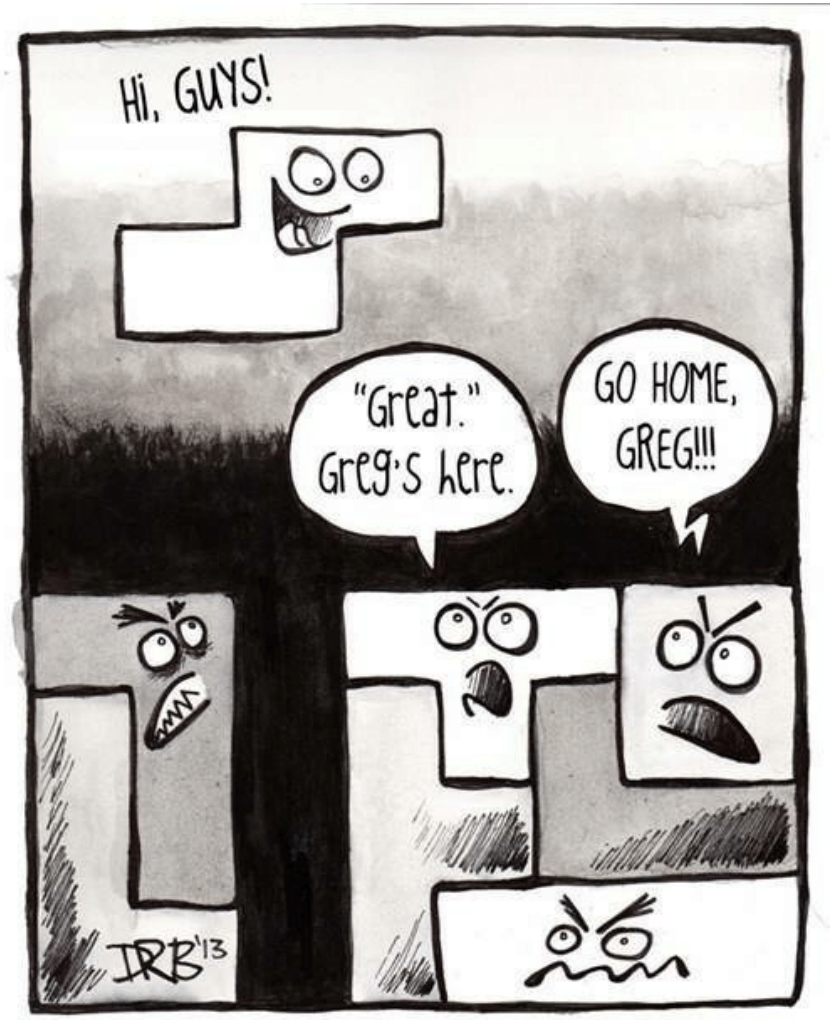
- Python code behind e.g. Flask
- Execution service from cloud provider
- Runtime
  - TensorFlow serving
  - Clipper
  - NVIDIA TensorRT inference server
  - MXNet Model Server
  - …
- Bespoke solutions (C++, …)



https://medium.com/@vikati/the-rise-of-the-model-servers-9395522b6c58

Production requirements

- Must **fit** the technology **stack**
- Not just about languages, but about **semantics**, **scalability**, **guarantees**
- Run **anywhere**, **any size**
- **Composable** building blocks
- Must try to **limit** the amount of **moving parts**
- And the amount of **moving data**
- Must make best **use of resources**

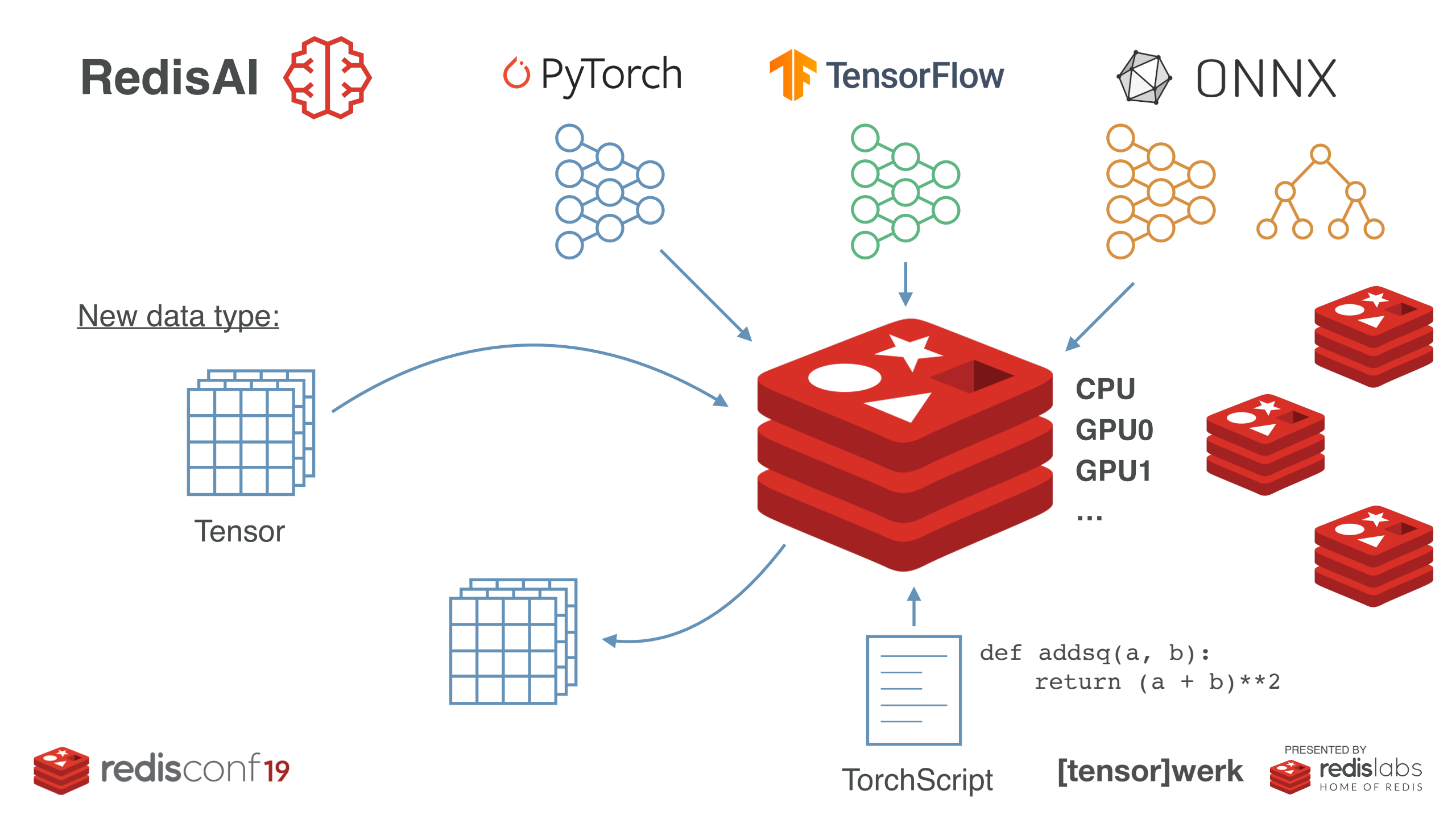


RedisAI

What it is

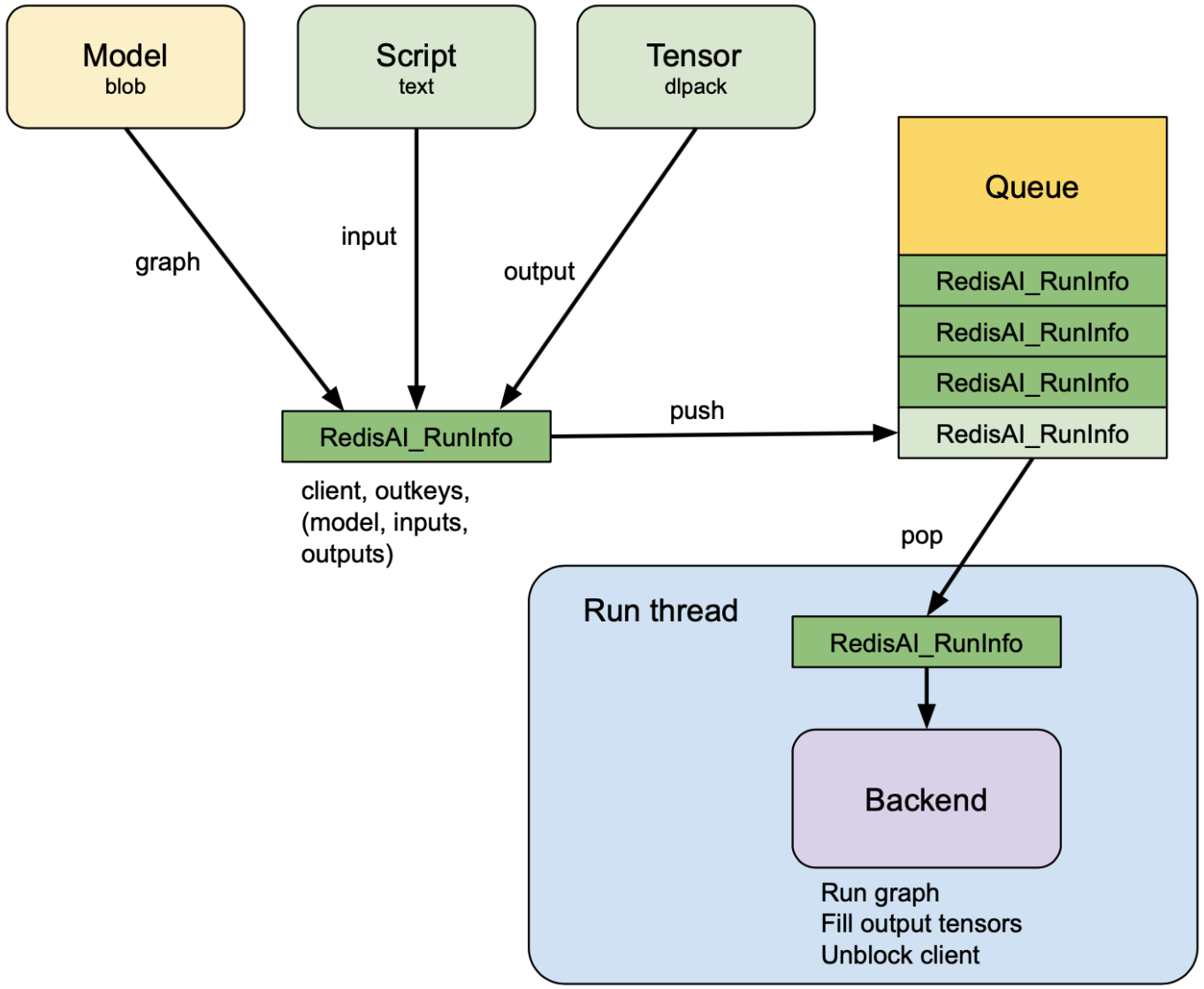
- A Redis module providing
- Tensors as a data type
- and Deep Learning model execution
- on CPU and GPU

- It turns Redis into a **full-fledged deep learning runtime**
- While still being **Redis**


RedisAI
PyTorch
TensorFlow
ONNX
New data type:
Tensor
CPU
GPU0
GPU1
...
def addsq(a, b):
return (a + b)**2
TorchScript

Architecture

- Tensors: framework-agnostic
- Queue + processing thread
  - Backpressure
  - Redis stays responsive
- Models are kept hot in memory
- Client blocks



Where to get it

- redisai.io
- github.com/RedisAI/RedisAI

```
docker run -p 6379:6379 -it --rm redisai/redisai
```

API: Tensor

- AI.TENSORSET
- AI.TENSORGET

```
AI.TENSORSET foo FLOAT 2 2 VALUES 1 2 3 4
AI.TENSORSET foo FLOAT 2 2 BLOB < buffer.raw

AI.TENSORGET foo BLOB
AI.TENSORGET foo VALUES
AI.TENSORGET foo META
```

API: Tensor

- based on dlpack https://github.com/dmlc/dlpack
- framework-independent

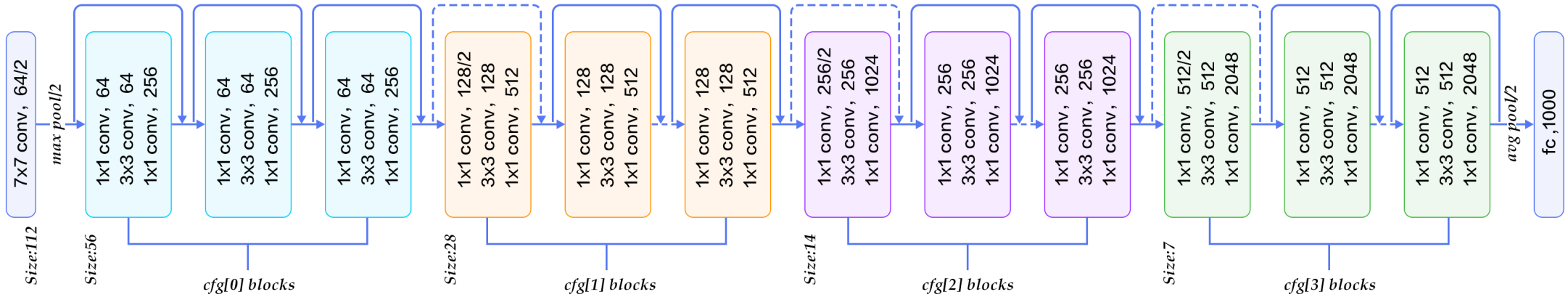
```
typedef struct {
  void* data;
  DLContext ctx;
  int ndim;
  DLDataType dtype;
  int64_t* shape;
  int64_t* strides;
  uint64_t byte_offset;
} DLTensor;
```

API: Model

- AI.MODELSET

```
AI.MODELSET resnet18 TORCH GPU < foo.pt
```

```
AI.MODELSET resnet18 TF CPU INPUTS in1 OUTPUTS linear4 < foo.pt
```



https://www.codeproject.com/Articles/1248963/Deep-Learning-using-Python-plus-Keras-Chapter-Re

API: Model

- AI.MODELRUN

```
AI.MODELRUN resnet18 INPUTS foo OUTPUTS bar
```

https://www.codeproject.com/Articles/1248963/Deep-Learning-using-Python-plus-Keras-Chapter-Re

Exporting models

- TensorFlow (+ Keras): freeze graph

```
import tensorflow as tf

var_converter = tf.compat.v1.graph_util.convert_variables_to_constants

with tf.Session() as sess:
    sess.run([tf.global_variables_initializer()])
    frozen_graph = var_converter(sess, sess.graph_def, ['output'])

tf.train.write_graph(frozen_graph, '.', 'resnet50.pb', as_text=False)
```

https://github.com/RedisAI/redisai-examples/blob/master/models/imagenet/tensorflow/model_saver.py

Exporting models

- PyTorch: JIT model

```
import torch

batch = torch.randn((1, 3, 224, 224))

traced_model = torch.jit.trace(model, batch)

torch.jit.save(traced_model, 'resnet50.pt')
```

https://github.com/RedisAI/redisai-examples/blob/master/models/imagenet/pytorch/model_saver.py

API: Script

- AI.SCRIPTSET
- AI.SCRIPTRUN

addtwo.txt

```
def addtwo(a, b):
    return a + b
```

```
AI.SCRIPTSET myadd2 GPU < addtwo.txt
```

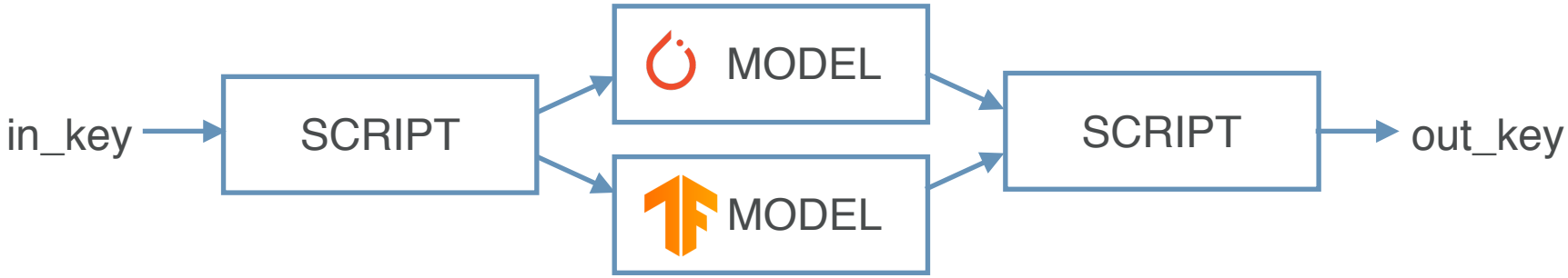
```
AI.SCRIPTRUN myadd2 addtwo INPUTS foo OUTPUTS bar
```

Scripts?

- SCRIPT is a TorchScript interpreter
- Python-like syntax for **tensor ops**
- on **CPU** and **GPU**
- **Vast library** of tensor operations
- Allows to prescribe computations directly (without exporting from a Python env, etc)
- Pre-proc, post-proc (but not only)

TORCHSCRIPT

Ref: https://pytorch.org/docs/stable/jit.html



Persistence

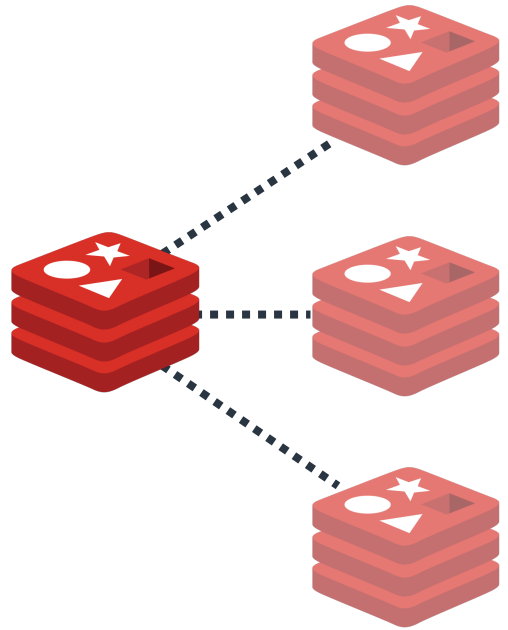
- RDB supported
- AOF almost :-)

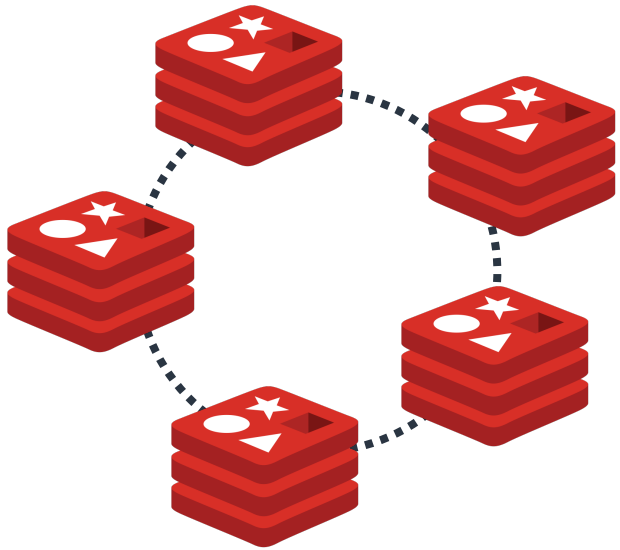
- Tensors are serialized (meta + blob)
- Models are serialized back into protobuf
- Scripts are serialized as strings

Replication

- **Master-replica** supported for all data types
- Right now, run cmds replicated too (post-conf: replication of results of computations where appropriate)

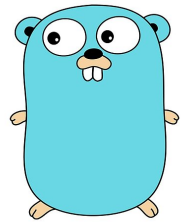
- **Cluster** supported, caveat: sharding models and scripts
- For the moment, use *hash tags*
  `{foo}resnet18 {foo}input1234`





RedisAI client libraries

- **NOTE**: any Redis client works **right now**
- JRedisAI https://github.com/RedisAI/JRedisAI
- redisai-py https://github.com/RedisAI/redisai-py
- Coming up: NodeJS, Go, … (community?)



RedisAI from NodeJS

```
async function load_model() {

  let redis = new Redis({ parser: 'javascript' });

  const model = fs.readFileSync(model_path, {'flag': 'r'})
  const script = fs.readFileSync(script_path, {'flag': 'r'})

  redis.call('AI.MODELSET', 'imagenet_model', 'TF', 'CPU',
             'INPUTS', 'images', 'OUTPUTS', 'output', model)
  redis.call('AI.SCRIPTSET', 'imagenet_script', 'CPU', script)
}
```

RedisAI from NodeJS

```
async function run(filename) {

  let redis = new Redis({ parser: 'javascript' });

  let image = await Jimp.read(filename);
  let input_image = image.cover(image_width, image_height);
  let buffer = Buffer.from(input_image.bitmap.data);

  redis.call('AI.TENSORSET', 'image1',
                'UINT8', image_height, image_width, 4,
                'BLOB', buffer)

  redis.call('AI.SCRIPTRUN', 'imagenet_script', 'pre_process_4ch',
                'INPUTS', 'image1', 'OUTPUTS', 'temp1')

  redis.call('AI.MODELRUN', 'imagenet_model',
                'INPUTS', 'temp1', 'OUTPUTS', 'temp2')

  redis.call('AI.SCRIPTRUN', 'imagenet_script', 'post_process',
                'INPUTS', 'temp2', 'OUTPUTS', 'out')

  let out = await redis.call('AI.TENSORGET', 'out', 'VALUES')

  let idx = out[2][0]

  console.log(idx, labels[idx.toString()])
}
```

Advantages of RedisAI today

- Keep the **data local**
- Keep stack **short**
- Run **everywhere** Redis runs
- Run **multi-backend**
- Stay **language-independent**
- **Optimize** use of **resources**
- Keep models **hot**
- HA with **sentinel**, clustering

https://github.com/RedisAI/redisai-examples

Roadmap

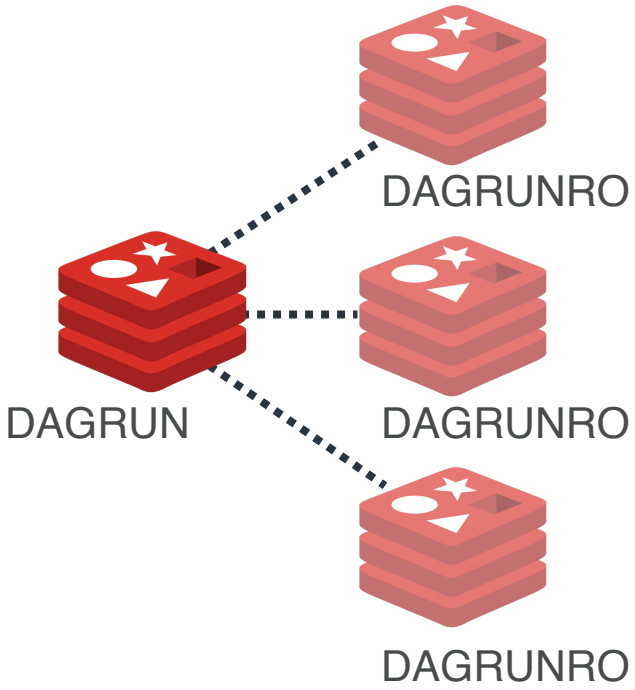
Roadmap: DAG

- DAG = Direct Acyclic Graph
- Atomic operation
- **Volatile** keys (~x~): command-local, don't touch keyspace

```
AI.DAGRUN
    SCRIPTRUN preproc normalize img ~in~
    MODELRUN resnet18 INPUTS ~in~ OUTPUTS ~out~
    SCRIPTRUN postproc probstolabel INPUTS ~out~ OUTPUTS label
```

Roadmap: DAG

- AI.DAGRUNRO:
  - if no key is written, **replicas can execute**
  - errors if commands try to write to non-volatile keys

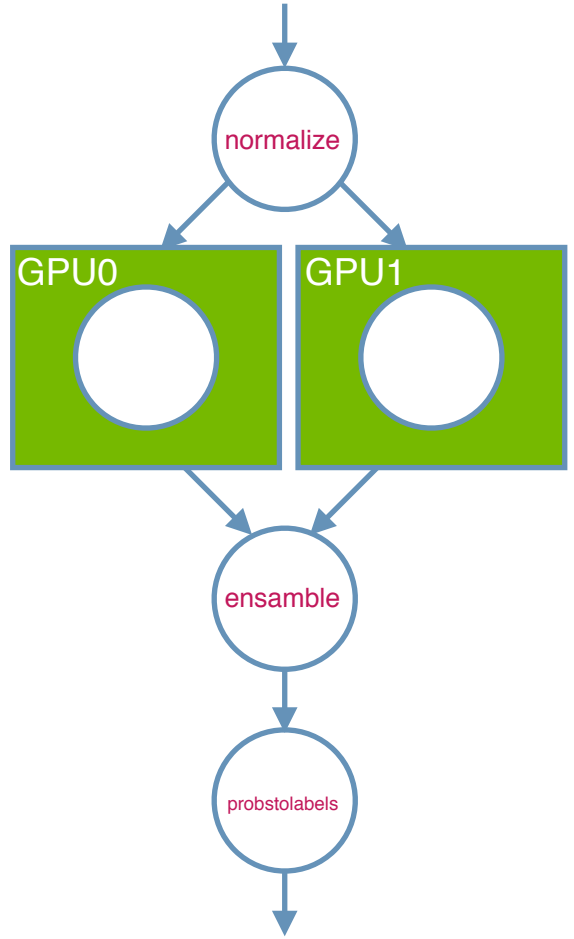



```
AI.DAGRUNRO
    TENSORSET ~img~ FLOAT 1 3 224 224 BLOB ...
    SCRIPTRUN preproc normalize ~img~ ~in~
    MODELRUN resnet18 INPUTS ~in~ OUTPUTS ~out~
    SCRIPTRUN postproc probstolabel INPUTS ~out~ OUTPUTS ~label~
    TENSORGET ~label~ VALUES
```

Roadmap: DAG

- Parallel multi-device execution
- One queue per device

```
AI.MODELSET resnet18a TF GPU0 ...
AI.MODELSET resnet18b TORCH GPU1 ...
AI.TENSORSET img ...
AI.DAGRUN
    SCRIPTRUN preproc normalize img ~in~
    MODELRUN resnet18a INPUTS ~in~ OUTPUTS ~out1~
    MODELRUN resnet18b INPUTS ~in~ OUTPUTS ~out2~
    SCRIPTRUN postproc ensamble INPUTS ~out1~ ~out2~ OUTPUTS ~out~
    SCRIPTRUN postproc probstolabel INPUTS ~out~ OUTPUTS label
```



Roadmap: DAG

- AI.DAGRUNASYNC:
  - non-blocking, returns an ID
  - ID can be used to later retrieve status + keyspace notification

```
AI.DAGRUNASYNC
    SCRIPTRUN preproc normalize img ~in~
    MODELRUN resnet18 INPUTS ~in~ OUTPUTS ~out~
    SCRIPTRUN postproc probstolabel INPUTS ~out~ OUTPUTS label
=> 12634
                AI.DAGRUNINFO 1264
                => RUNNING [... more details …]
                AI.DAGRUNINFO 1264
                => DONE
```

Roadmap: streams

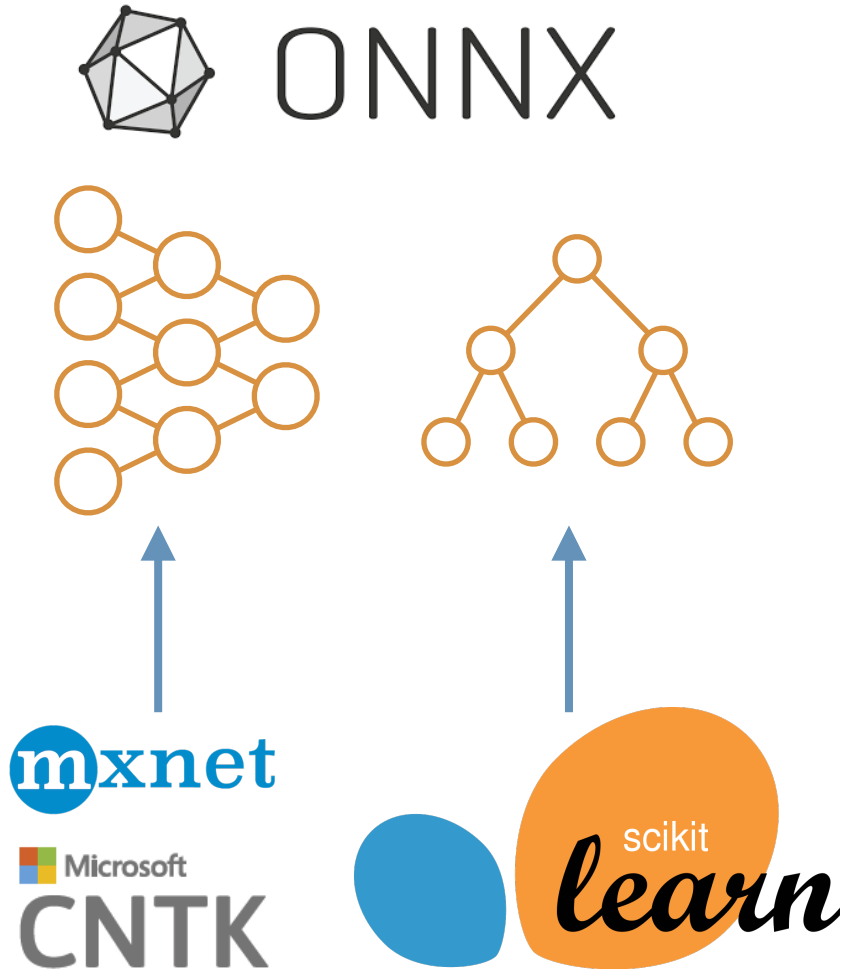
- Pervasive use of streams

```
AI.DAGRUN
   SCRIPTRUN preproc normalize instream ~in~
   MODELRUN resnet18 INPUTS ~in~ OUTPUTS ~out~
   SCRIPTRUN postproc probstolabel INPUTS ~out~ OUTPUTS outstream

AI.DAGRUNASYNC
   SCRIPTRUN preproc normalize instream ~in~
   MODELRUN resnet18 INPUTS ~in~ OUTPUTS ~out~
   SCRIPTRUN postproc probstolabel INPUTS ~out~ OUTPUTS outstream
```

Roadmap: ONNXRuntime backend

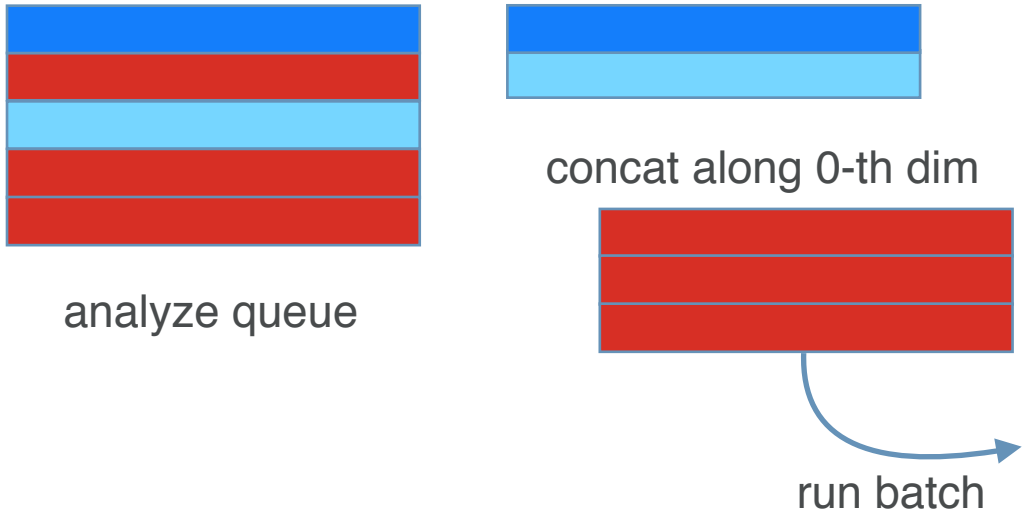
- Runtime from Microsoft
- **ONNX**
  - exchange format for NN
  - export from many frameworks (MXNet, CNTK, …)
- **ONNX-ML**
  - ONNX for **machine learning models** (RandomForest, SVN, K-means, etc)
  - export from scikit-learn

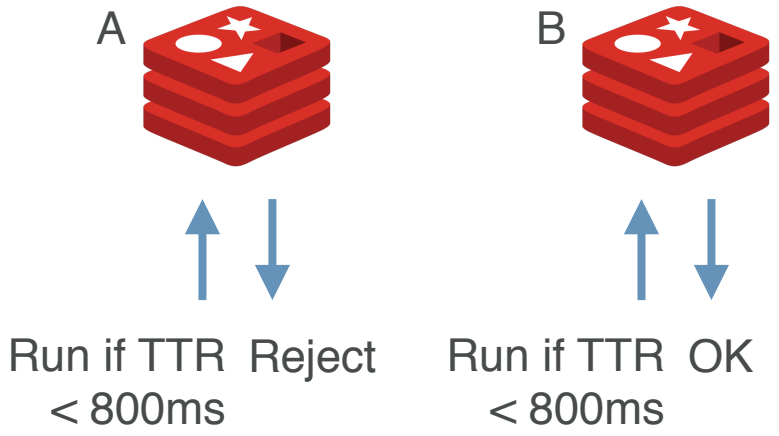


Roadmap: Auto-batching

- Auto-batching
  - Transparent batching of requests
  - Queue gets rearranged according to other analogous requests in the queue, time to response
  - Only if different clients or async

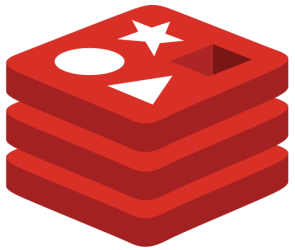
- Time-to-response
  - ~Predictable in DL graphs
  - Can reject request if TTR > estimated time of queue + run





Roadmap: Misc

- Dynamic loading of backends:
  - don't load what you don't need (especially on GPU)



```
AI.CONFIG LOADBACKEND TF CPU
AI.CONFIG LOADBACKEND TF GPU
AI.CONFIG LOADBACKEND TORCH CPU
AI.CONFIG LOADBACKEND TORCH GPU
...
```

- Monitoring with AI.INFO:
  - more granular information on running operations, commands, memory

```
AI.INFO
AI.INFO MODEL key
AI.INFO SCRIPT key
...
```

RedisAI Enterprise

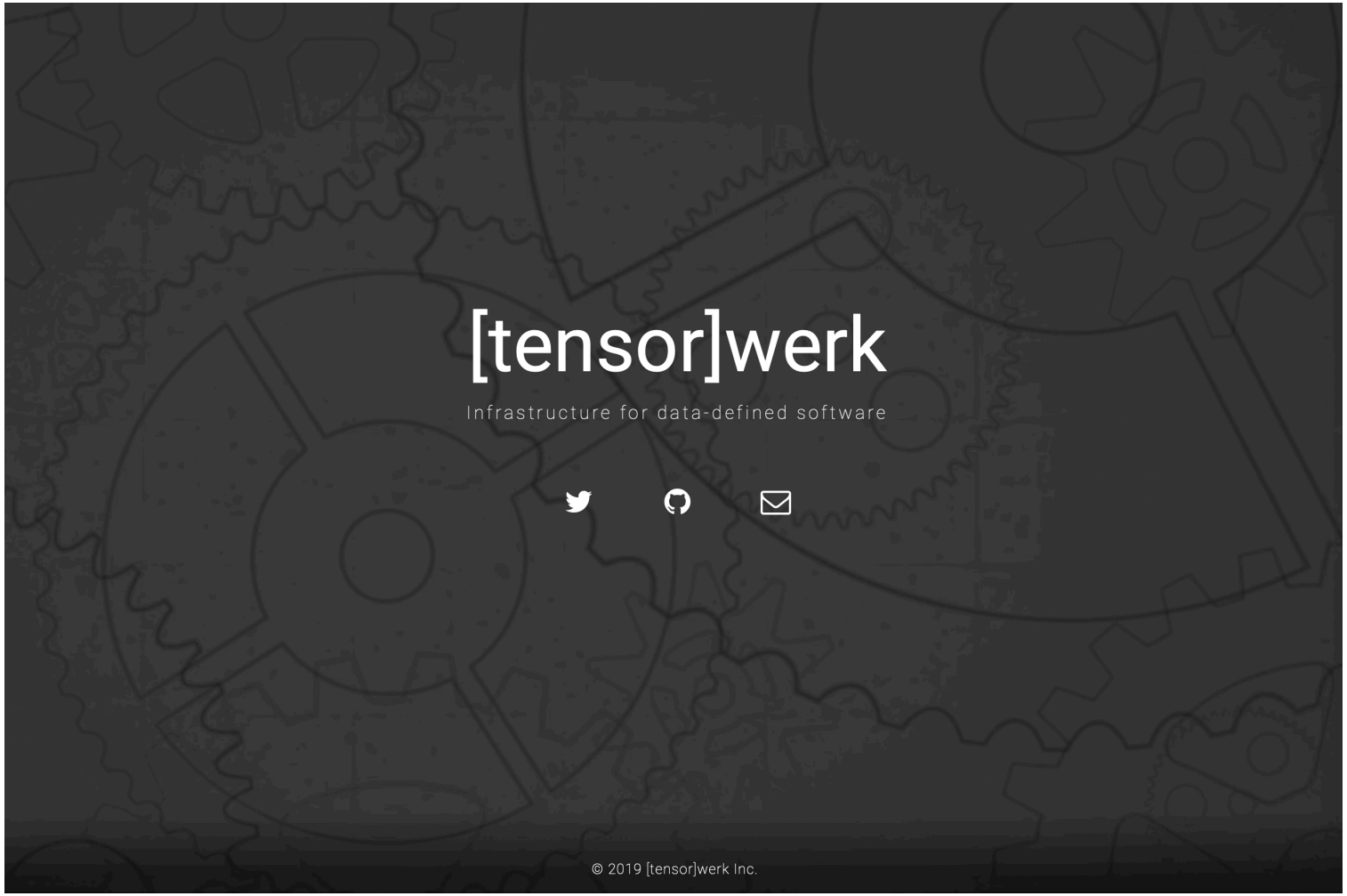
- Advanced monitoring
  - Health
  - Performance
  - Model metrics (reliability)
- A/B testing
- Module integration, e.g.
  - RediSearch (FAISS)
  - Anomaly detection with RedisTS

- Training/fine-tuning in-Redis

Acknowledgements

- **[tensor]werk**
  - Sherin Thomas, Rick Izzo, Pietro Rota
- **RedisLabs**
  - Guy Korland, Itamar Haber, Pieter Cailliau, Meir Shpilraien, Mark Nunberg, Ariel Madar
- **Orobix**
  - Everyone!
  - Manuela Bazzana, Daniele Ciriello, Lisa Lozza, Simone Manini, Alessandro Re

Hit me up!



tensorwerk.com

Development tools for the data-defined software era

1. Launching April 2019
2. Looking for **investors, users, contributors**

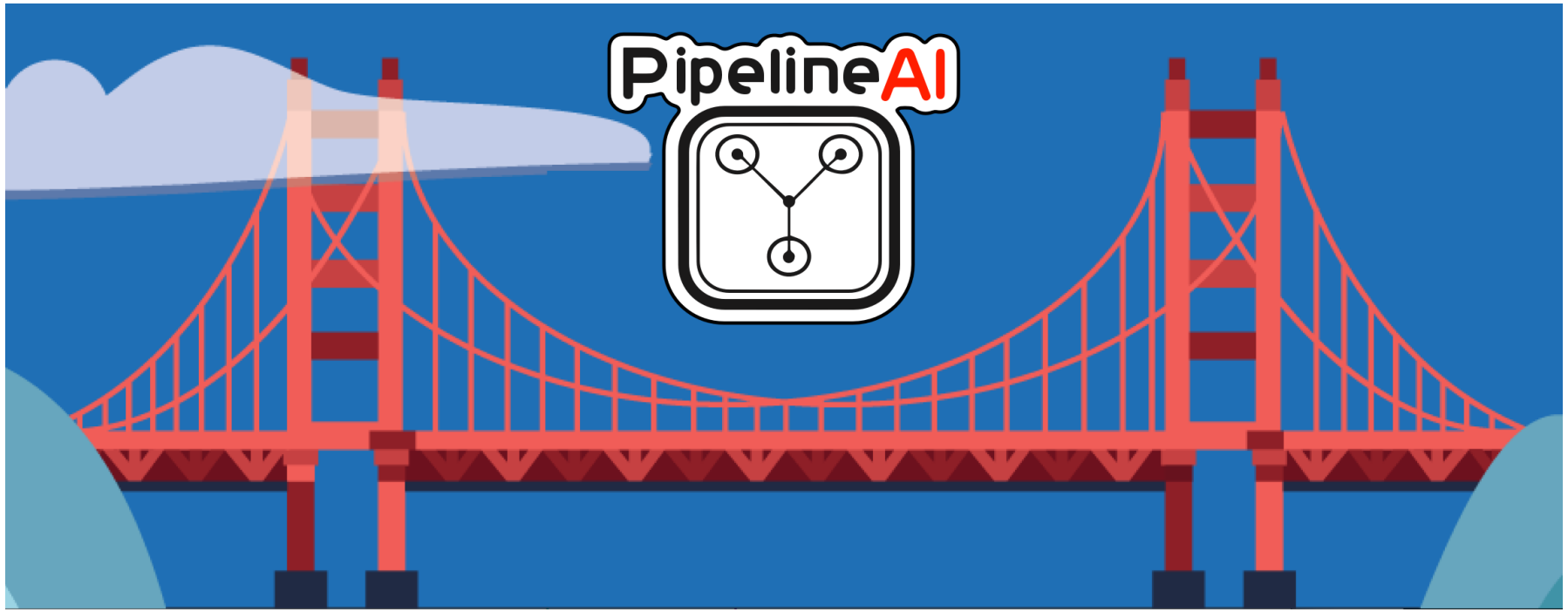
Projects:

- **RedisAI**: enterprise-grade runtime (with Redis Labs)
- **Hangar**: version control for tensor data (mid April, BSD)

luca@tensorwerk.com

Chris Fregly, Pipeline.ai

- End-to-End Deep Learning from Research to Production
- Any Cloud, Any Framework, Any Hardware
- Free Community Edition: https://community.pipeline.ai



redisconf19

Thank you!

TS.RANGE key FROM_TIMESTAMP
TO_TIMESTAMP [aggregationType]
[bucketSizeSeconds]

presented by

redislabs
HOME OF REDIS

redis