Microsoft

Cloud Native ML with Kubeflow & TensorRT

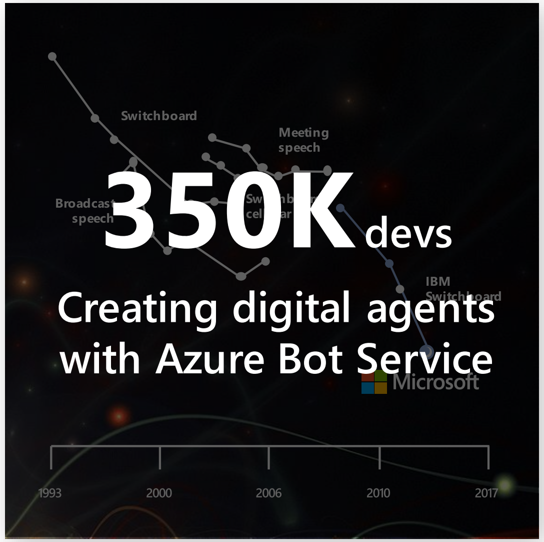
David Aronchick
Head of OSS ML Strategy, Azure

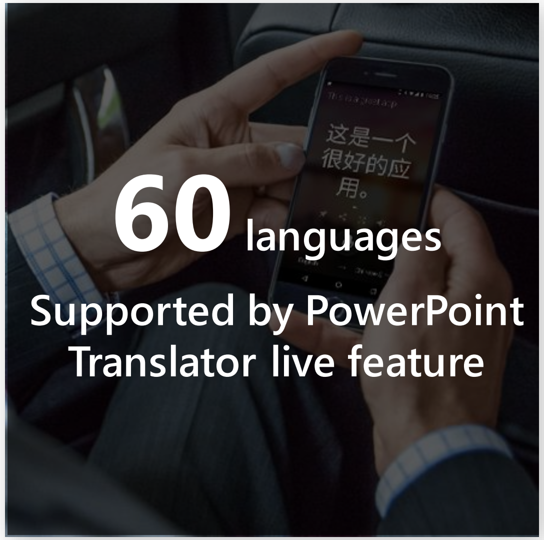
Seth Juarez
Senior Cloud Developer Advocate, Azure

One Year Ago…

What is Machine Learning?

Machine Learning is a way of solving problems without explicitly knowing how to create the solution.





2016

Object recognition
Human parity

2017

Speech recognition
Human parity

January 2018

Machine reading comprehension
Human parity

March 2018

Machine translation
Human parity

But ML is hard!

Four Years Ago…

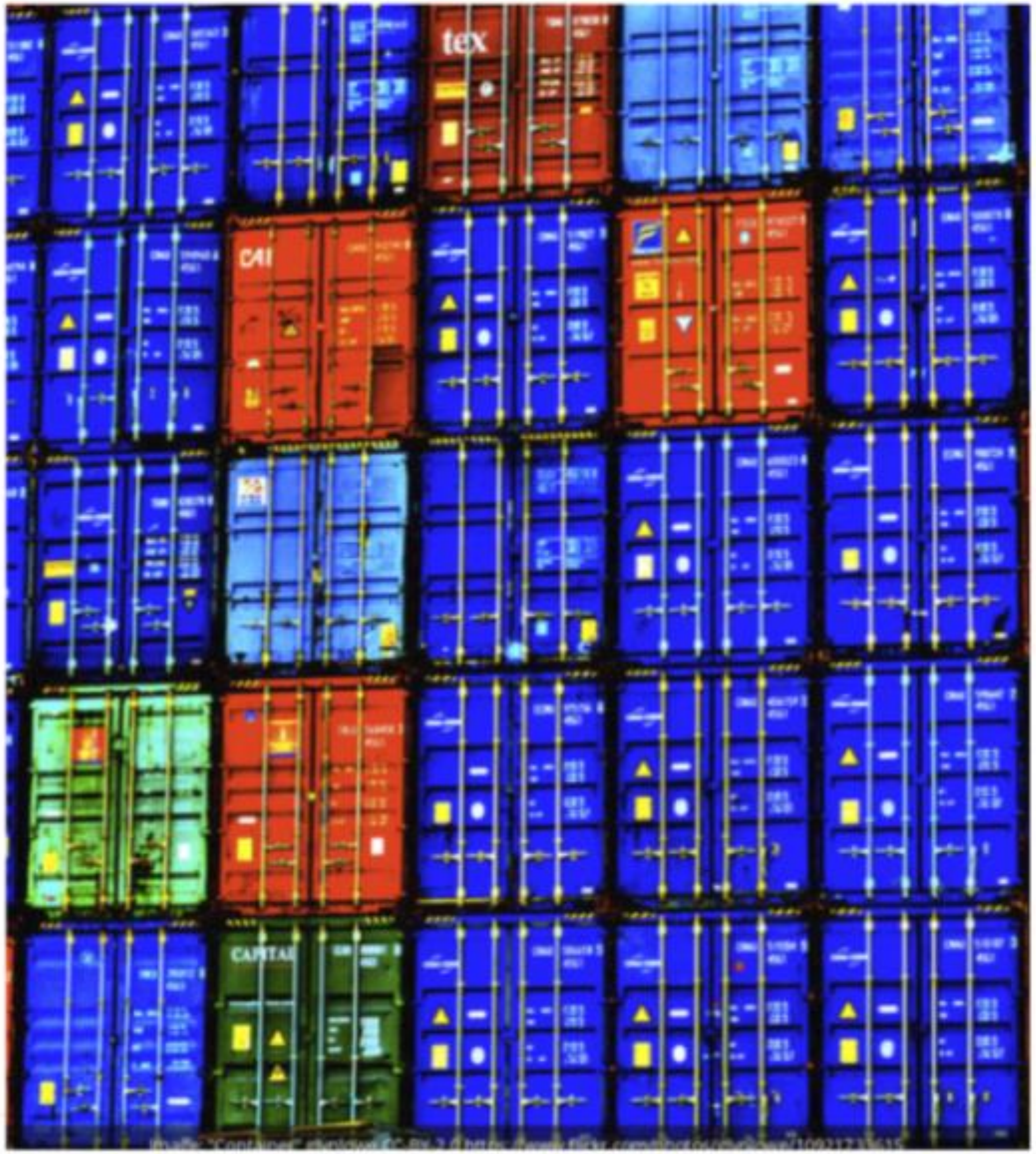
Google and Containers

Everything at Google runs in a container.

Internal usage:

- Resource isolation and predictability
- Quality of Services
    - batch vs. latency sensitive serving
- Overcommitment (not for GCE)
- Resource Accounting

We start over 2 billion containers per week.



Kubernetes

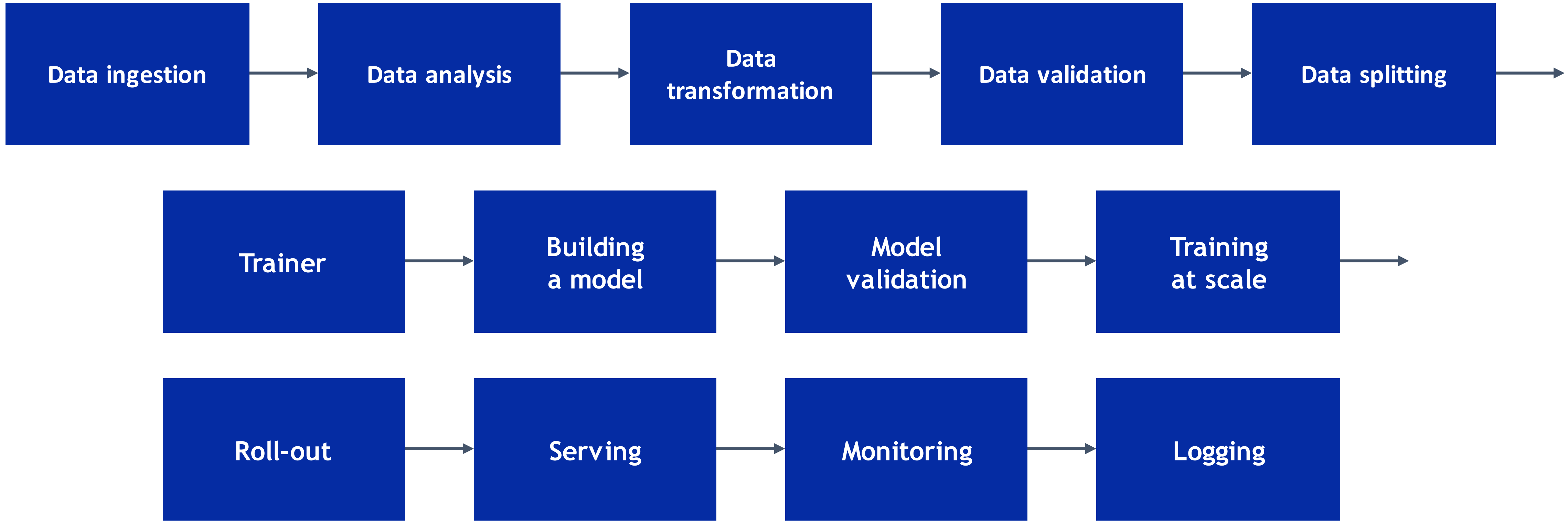
Cloud Native Apps

Cloud Native ML?

Platform

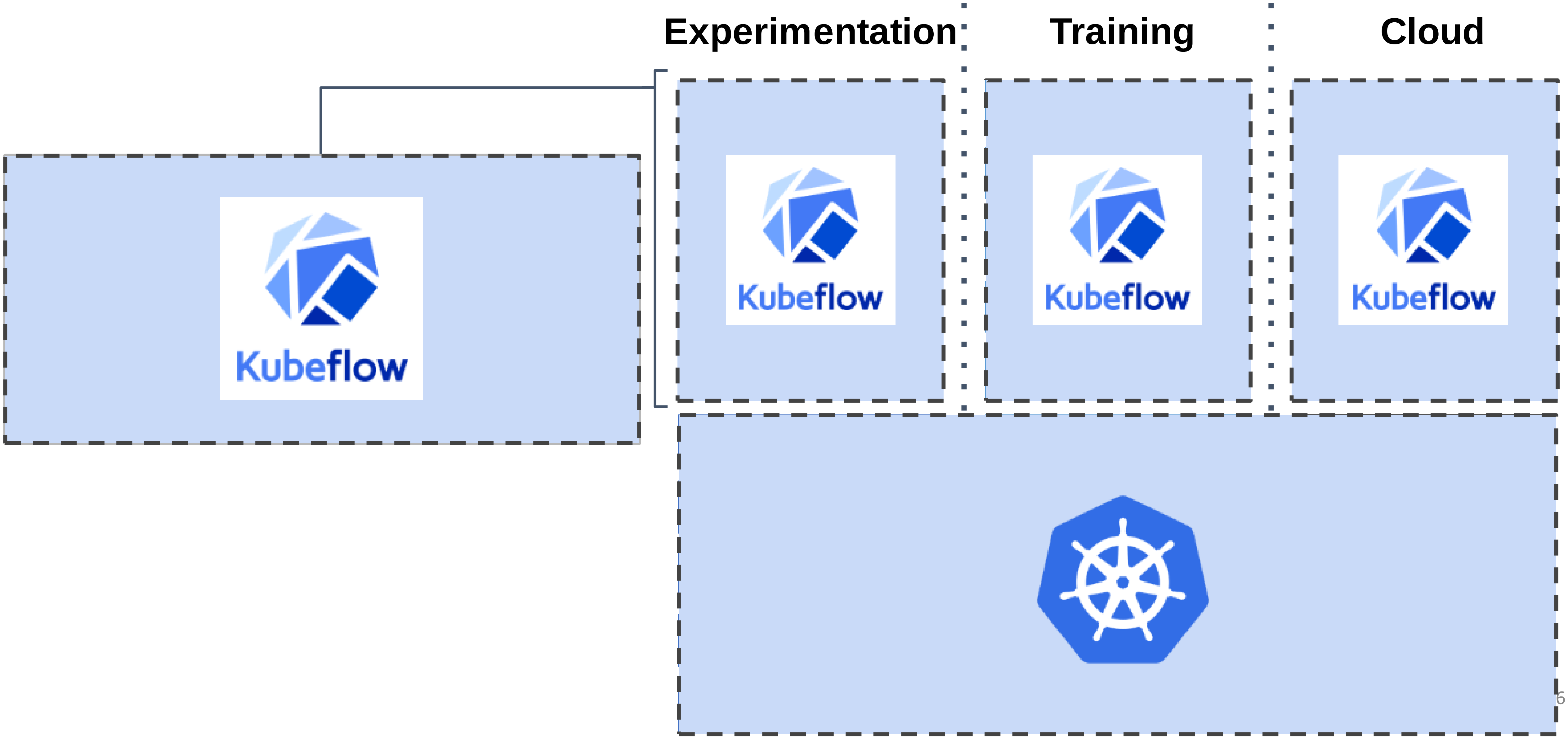
Building a model

Platform


Data ingestion
Data analysis
Data transformation
Data validation
Data splitting
Trainer
Building a model
Model validation
Training at scale
Roll-out
Serving
Monitoring
Logging

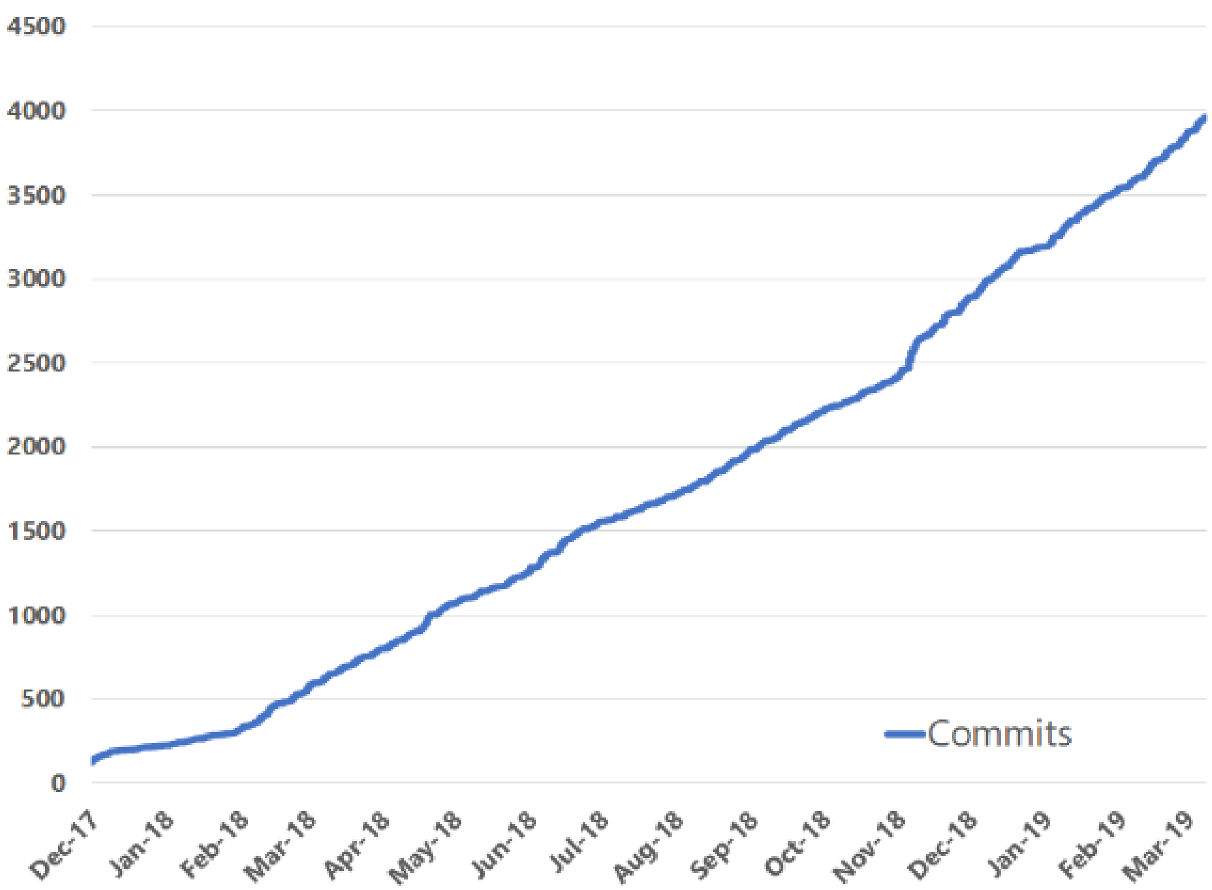
Kubecon 2017

Make it Easy for Everyone
to Develop, Deploy and Manage
Portable, Distributed ML
on Kubernetes


Experimentation
Training
Cloud
Kubeflow
Kubeflow
Kubeflow
Kubeflow

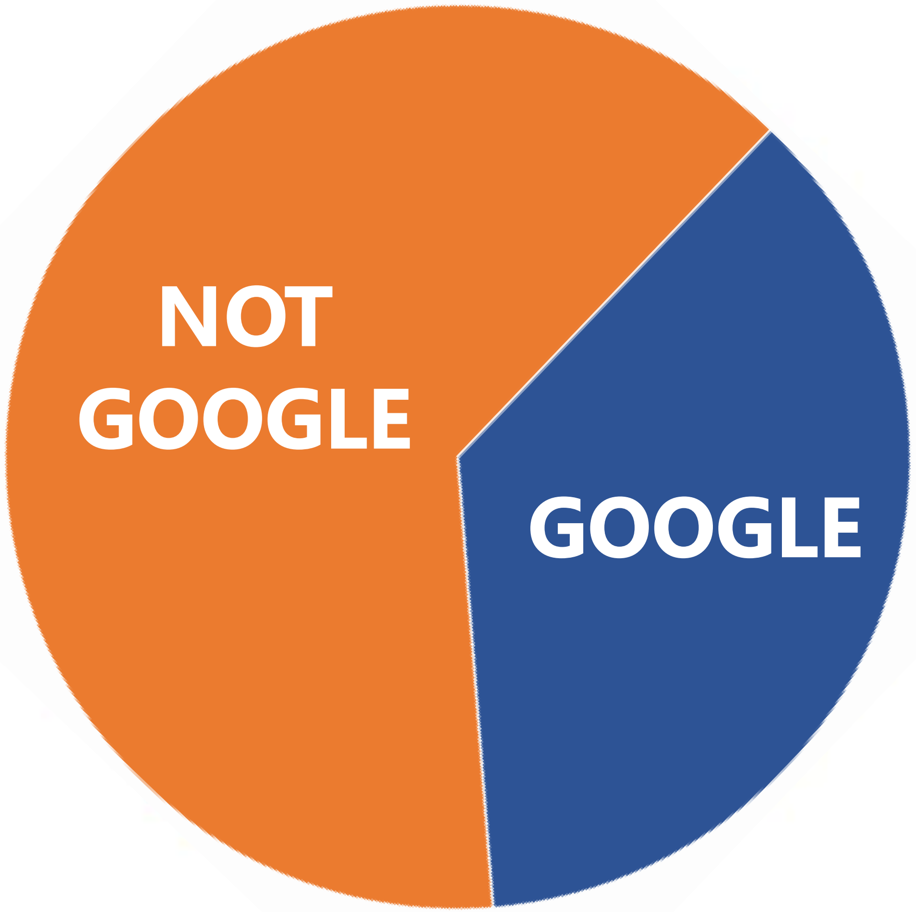
Cloud Native ML!

Momentum!

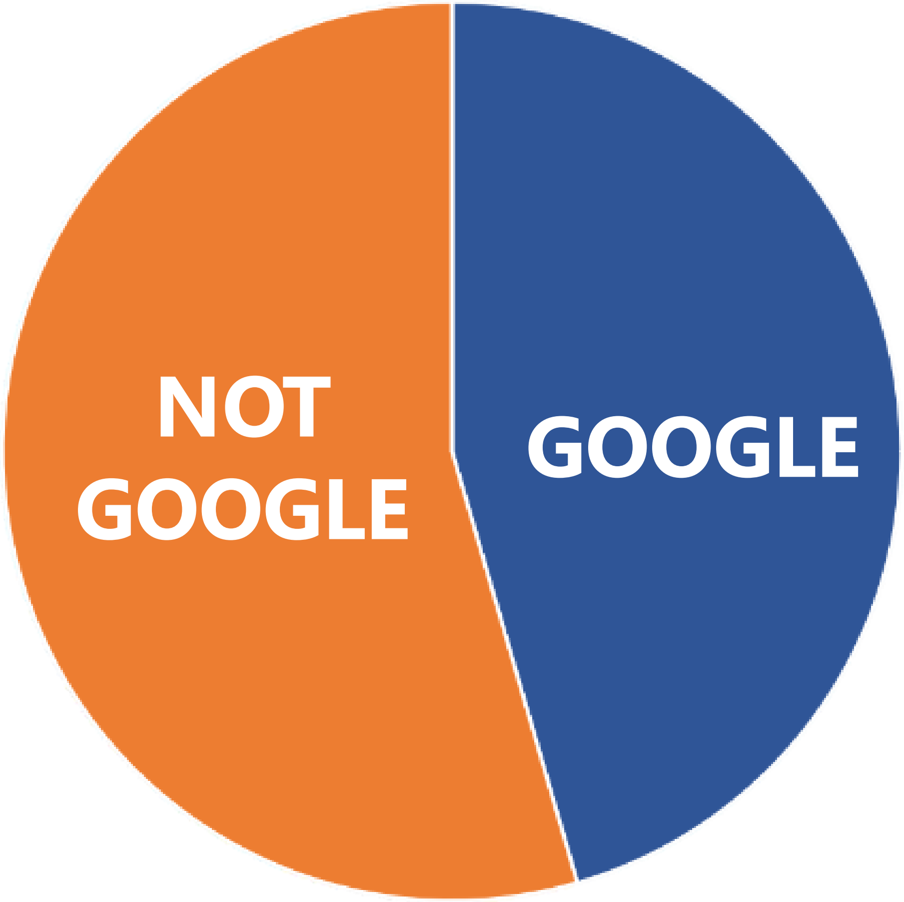


- ~4000 commits
- ~200 community contributors
- ~50 companies contributing, including:

Community Contributions



Kubernetes



Kubeflow

Critical User Journey Comparison

2017

- Experiment with Jupyter
- Distribute your training with TFJob
- Serve your model with TF Serving

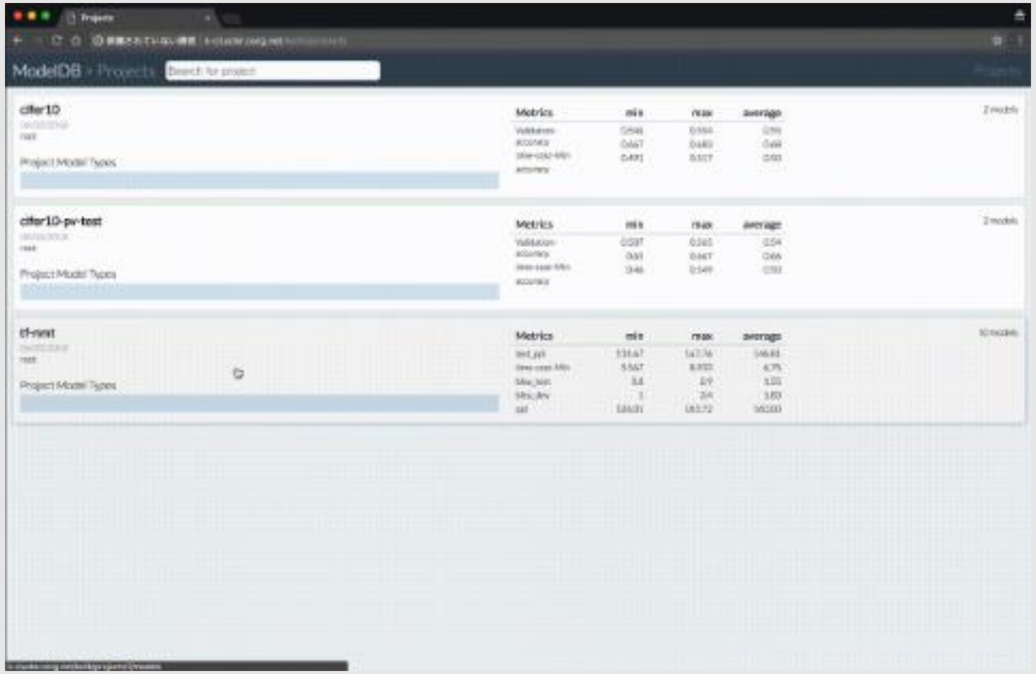
2019

- Setup locally with miniKF
- Access your cluster with Istio/Ingress
- Ingest your data with Pachyderm
- Transform your data with TF.T
- Analyze the data with TF.DV
- Experiment with Jupyter
- Hyperparam sweep with Katib
- Distribute your training with TFJob
- Analyze your model with TF.MA
- Serve your model with Seldon
- Orchestrate everything with KF.Pipelines

Community Contribution

- Pluggable microservice architecture for HP tuning
  - Different optimization algorithms
  - Different frameworks
- StudyJob (K8s CRD)
  - Hides complexity from user
  - No code needed to do HP tuning

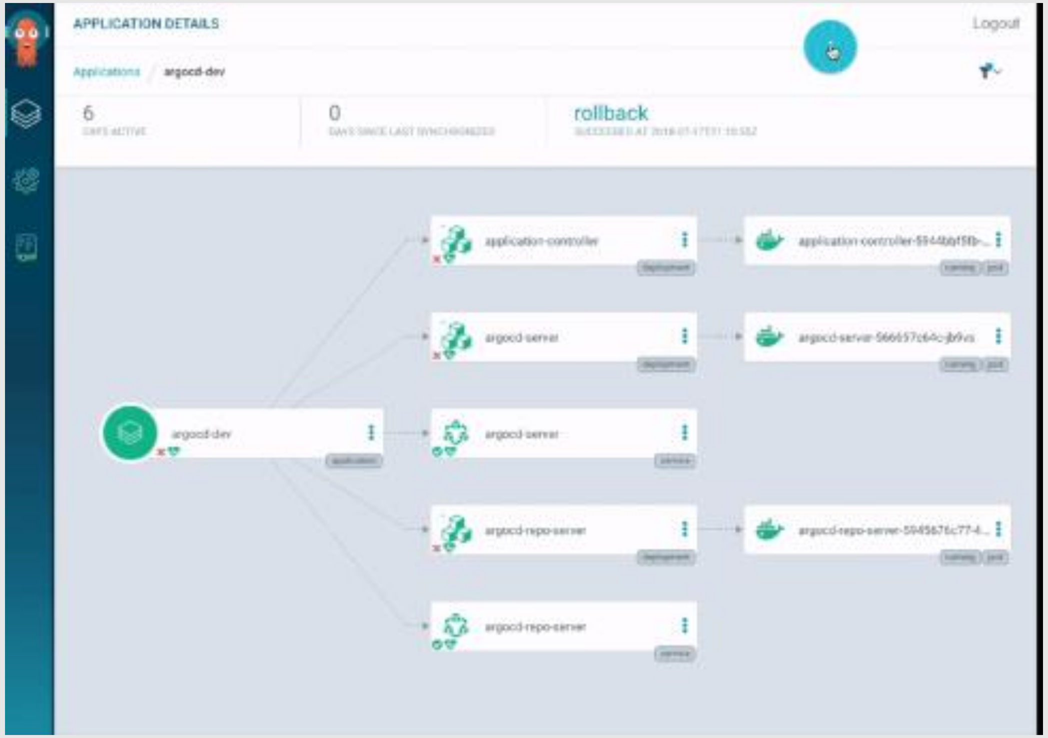
Katib from NTT



Community Contribution

- Argo CRD for workflows
- Argo CRD is engine for Pipelines
- Argo CD for GitOps

Argo from Intuit



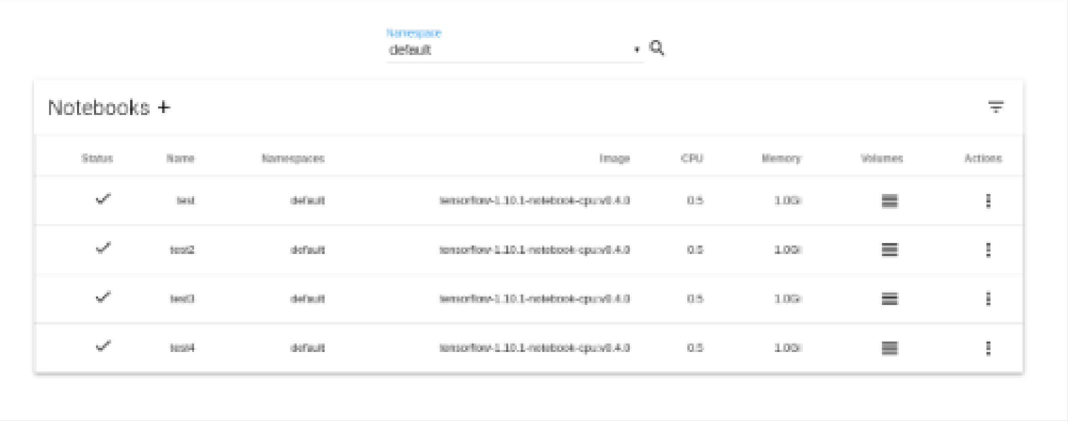
Community Contribution

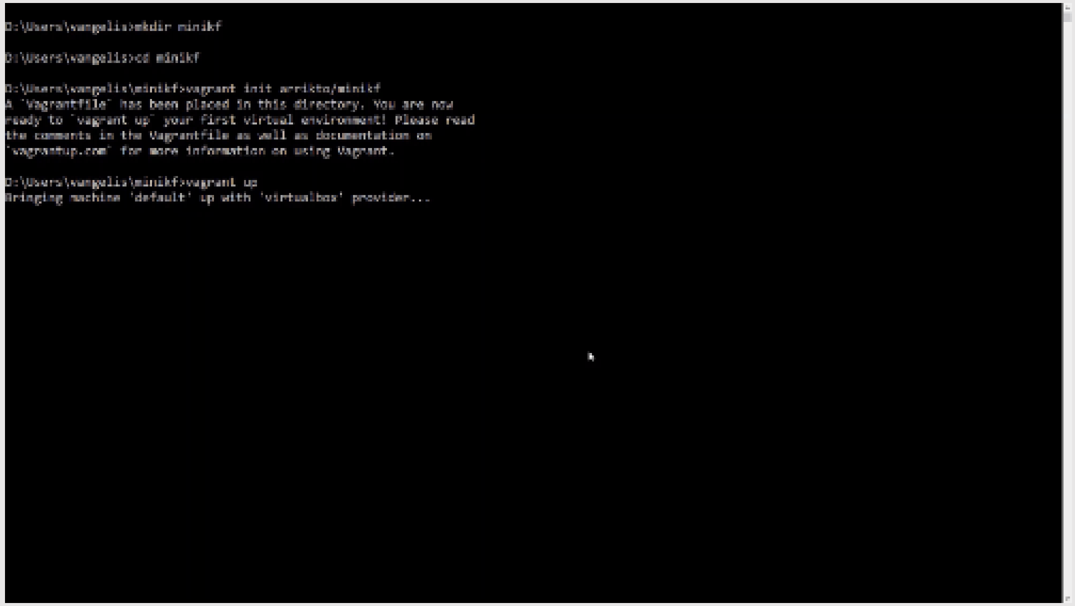
- Core Notebook Experience
  - **0.4:** New JupyterHub-based UI
  - **0.5:** K8s-Native Notebooks UI

- Pipelines: Support for local storage
- Multiple Persistent Volumes

- MiniKF: All-in-one packaging for seamless local deployments

NB & Storage from Arrikto

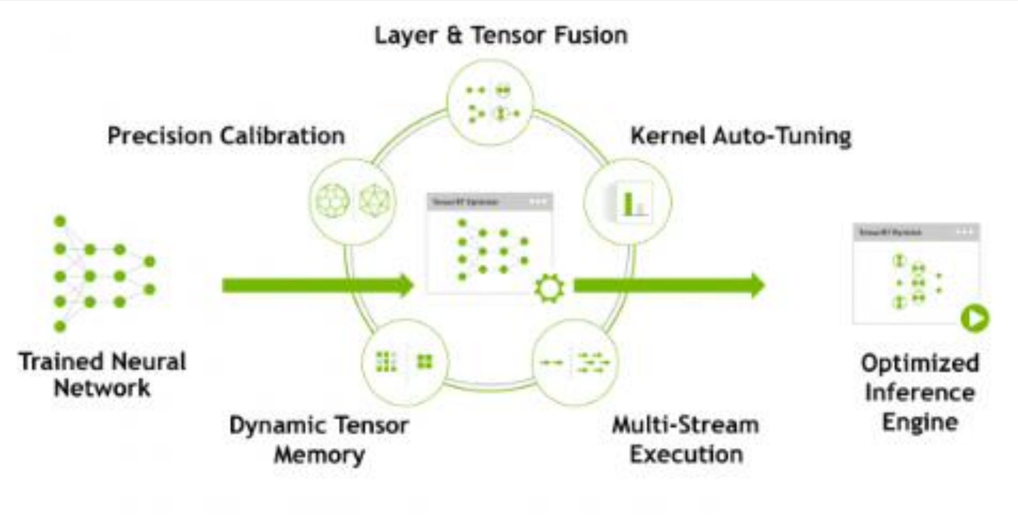




Community Contribution

- Production datacenter inferencing server
- Maximize real-time inference performance of GPUs
- Multiple models per GPU per node
- Supports heterogeneous GPUs & multi GPU nodes
- Integrates with orchestration systems and auto scalers via latency and health metrics

TensorRT from NVidia



Introducing Kubeflow 0.5

What's in the box?

UX investments - First class notebooks & central dashboard

- Build/Train/Deploy From notebook
- Better multi-user support
- A new web-based spawer

Enterprise readiness

- Better namespace support
- API stability
- Upgradability with preservation of historical metadata

Advanced composability & tooling

- Advanced support for calling out to web services
- Ability to specify GPU/TPUs for pipeline steps
- New metadata backend

Better/Faster/Production Notebooks!

User Goal = Just give me a notebook!

Problem

- Setting up A notebook is O(easy)
- Setting up a rich, production-ready notebook is O(hard)
- Setting up a rich, production-ready notebook that works anywhere, on any cloud, with a minimum of changes is O(very very hard)

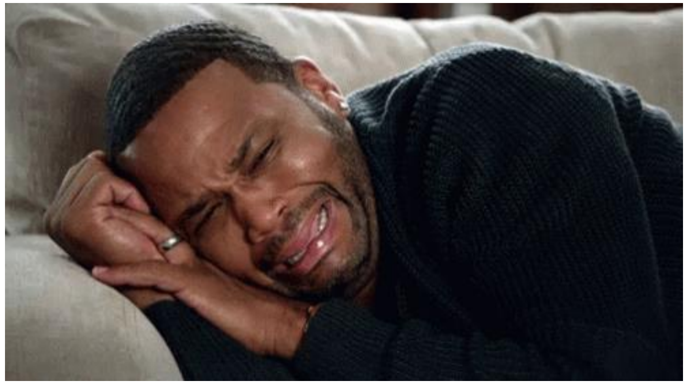
Better/Faster/Production Notebooks!

Setting up a notebook is easy!

```
$ curl -O
https://repo.continuum.io/archive/Anaconda3-5.0.1-
Linux-x86_64.sh

$ bash -c Anaconda3-5.0.1-Linux-x86_64.sh

$ conda create -y -n mlenv python=2 pip scipy
gevent sympy

$ source activate mlenv

$ pip install tensorflow==1.13.0 | tensorflow-
gpu==1.7.0

$ open http://127.0.0.1:8080
```

Except…

- Custom libraries
- HW provisioning (especially GPUs) & **drivers**
- Portability (between laptop and clouds)
- Security profiles
- Service accounts
- Credentials
- Lots more…



Better/Faster/Production Notebooks!

Solution –

Declarative Data Science Environments with Kubeflow!

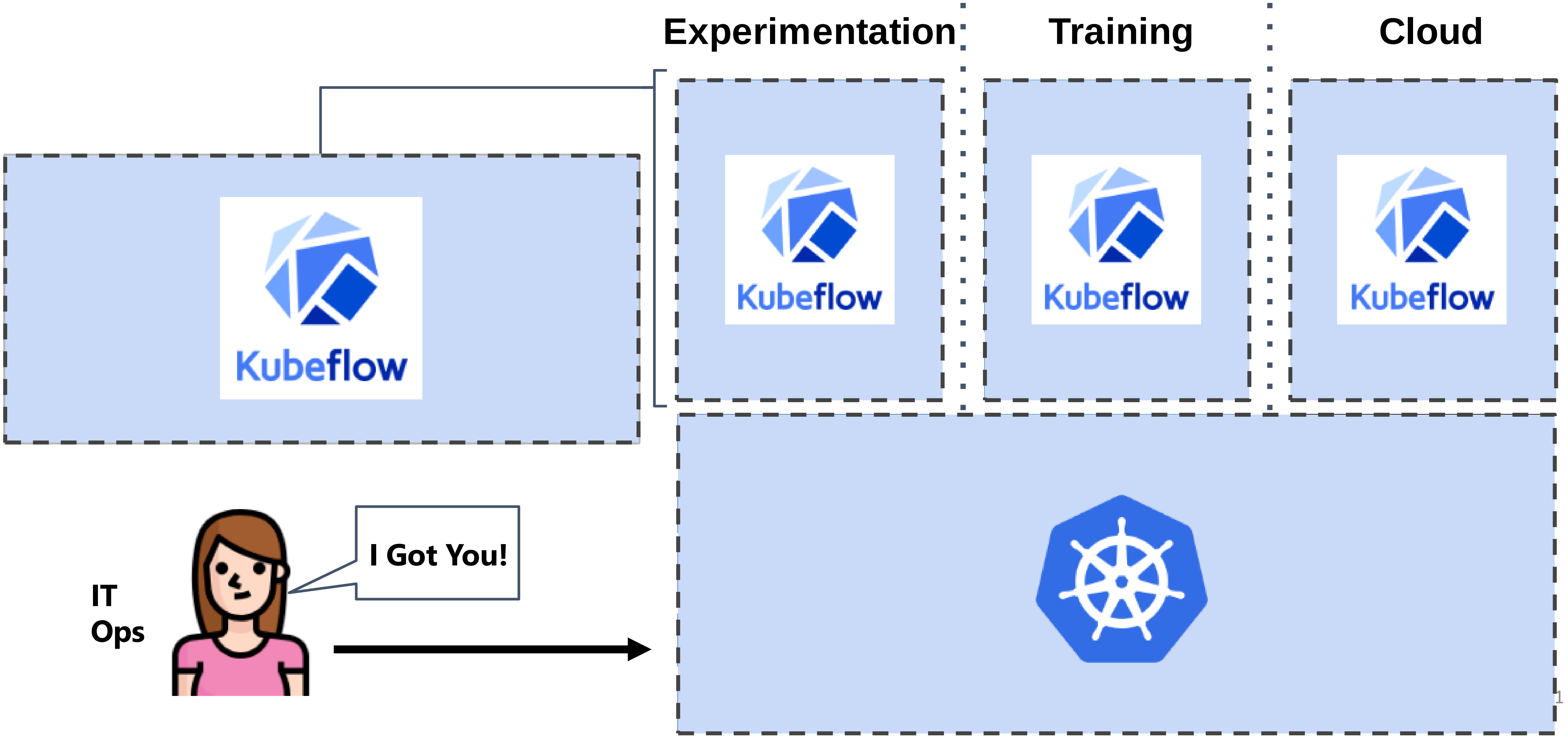
Better/Faster/Production Notebooks!

Setting up a declarative environment is easy!

```
$ kfctl.sh init
$ kfctl.sh --platform aks \
           --project my-project
$ kfctl.sh generate platform
$ kfctl.sh apply platform
$ kfctl.sh generate k8s
$ kfctl.sh apply k8s
```

Add your custom components!

```
# Add Seldon Server
$ ks pkg install kubeflow/seldon

# Add XGBoost
$ ks pkg install kubeflow/xgboost

# Add hyperparameter tuning
$ ks pkg install kubeflow/katib

# Add Seldon Server
$ ks pkg install kubeflow/seldon
```


Experimentation
Training
Cloud
Kubeflow
Kubeflow
Kubeflow
Kubeflow
IT
Ops
I Got You!

DEMO

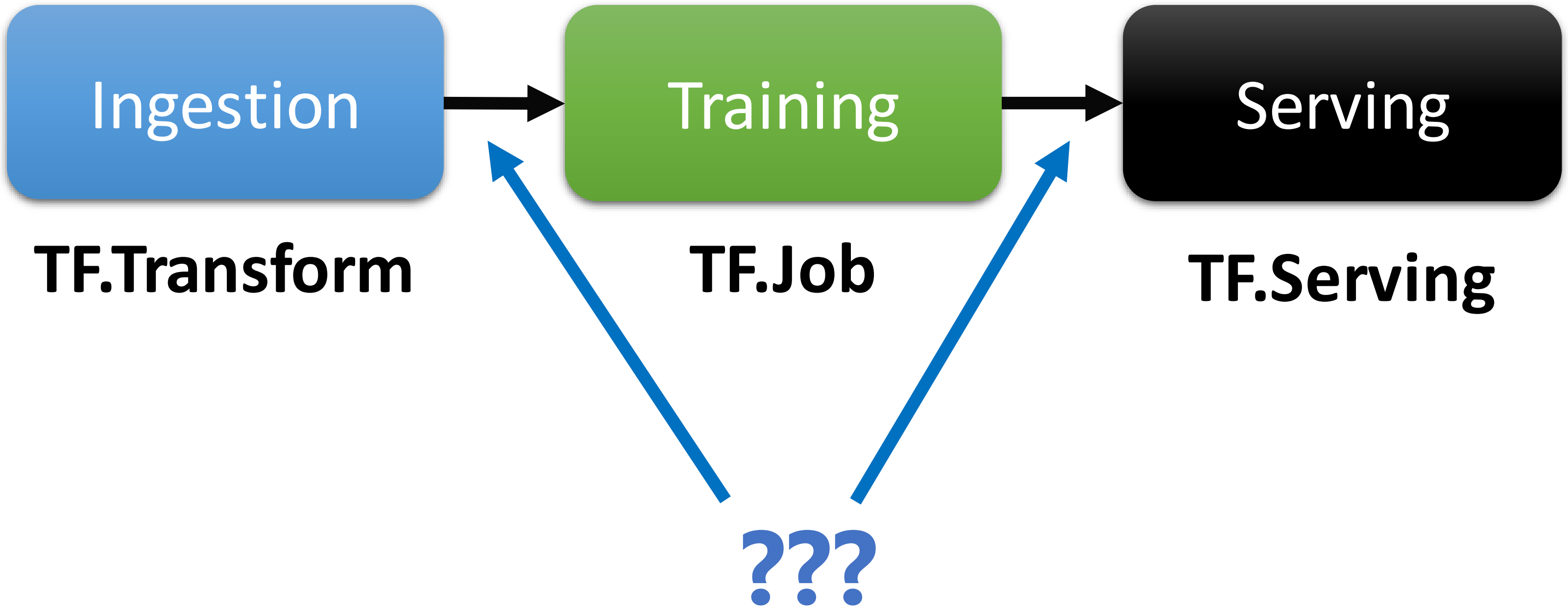
Rich Container Based Pipelines

User Goal = Repeatable, multi-stage ML training

Problem

- Tools not built to be containerized/orchestrated
- Coordinating between steps often requires writing custom code
- Different tools have different infra requirements

Rich Container Based Pipelines



Pipelines should:

- Be cloud native (microservice oriented, loosely coupled) and ML aware
- Support both data and task driven workflows
- Understand non-Kubeflow-based services (e.g. external to the cluster)

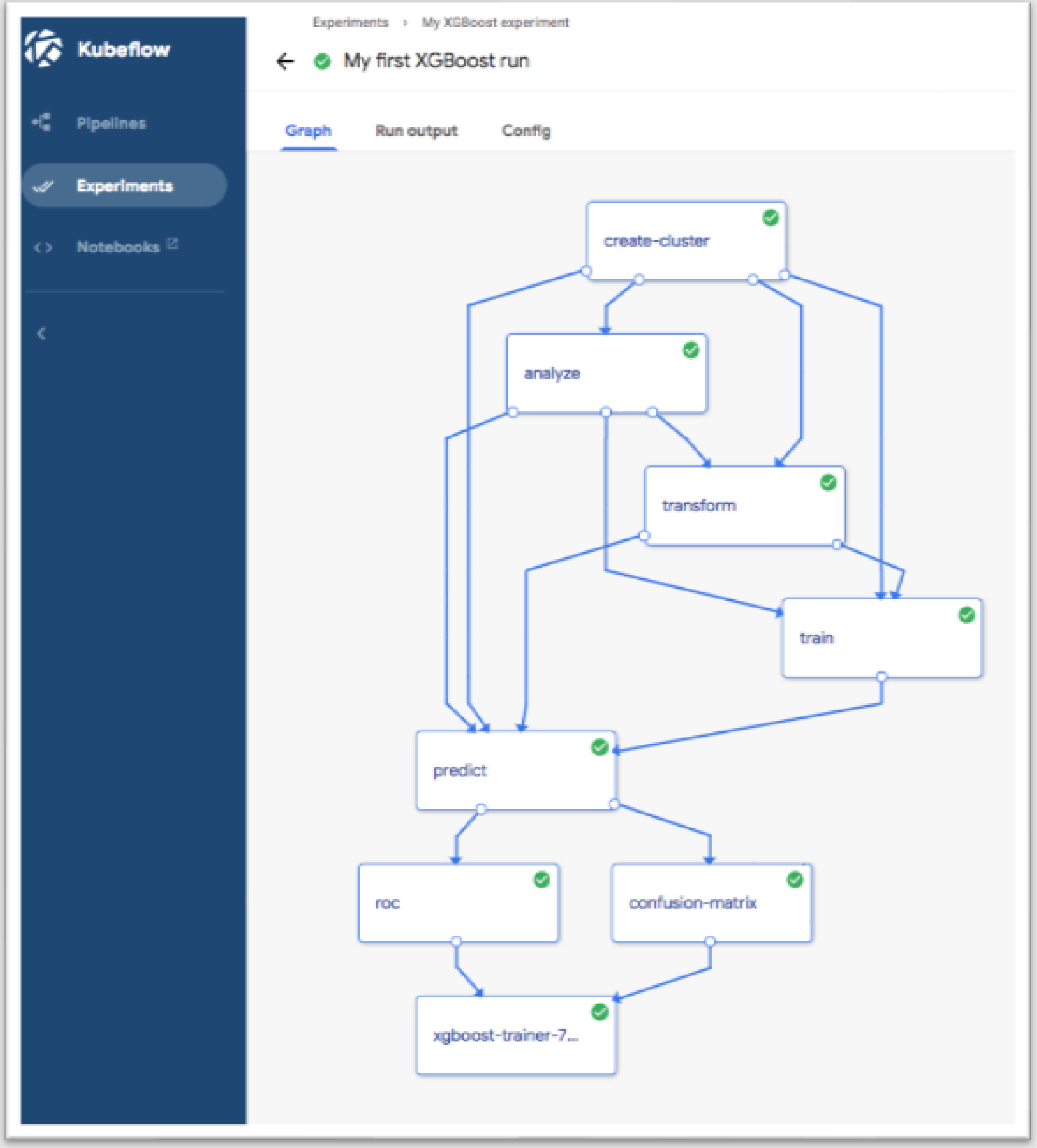
Rich Container Based Pipelines

Solution –

Kubeflow Pipelines!

Kubeflow Pipeline Details

- Containerized Implementations of ML Tasks
  - Escapsulates all the dependencies of a step with no conflicts
  - Step can be singular or distributed
  - Can also involve external services
- Specified via Python SDK
- Inputs/outputs/parameters can be chained together



Rich Container Based Pipelines

TF.Transform **TF.Job** **TF.Serving**

```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
             file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
            arguments=[ingestStep.outputs['bucket’]])

servingStep = dsl.ContainerOp(image=tfs_image, <params>,
              arguments=[convertStep.outputs['bucket']])
```

Can I Change a Step?

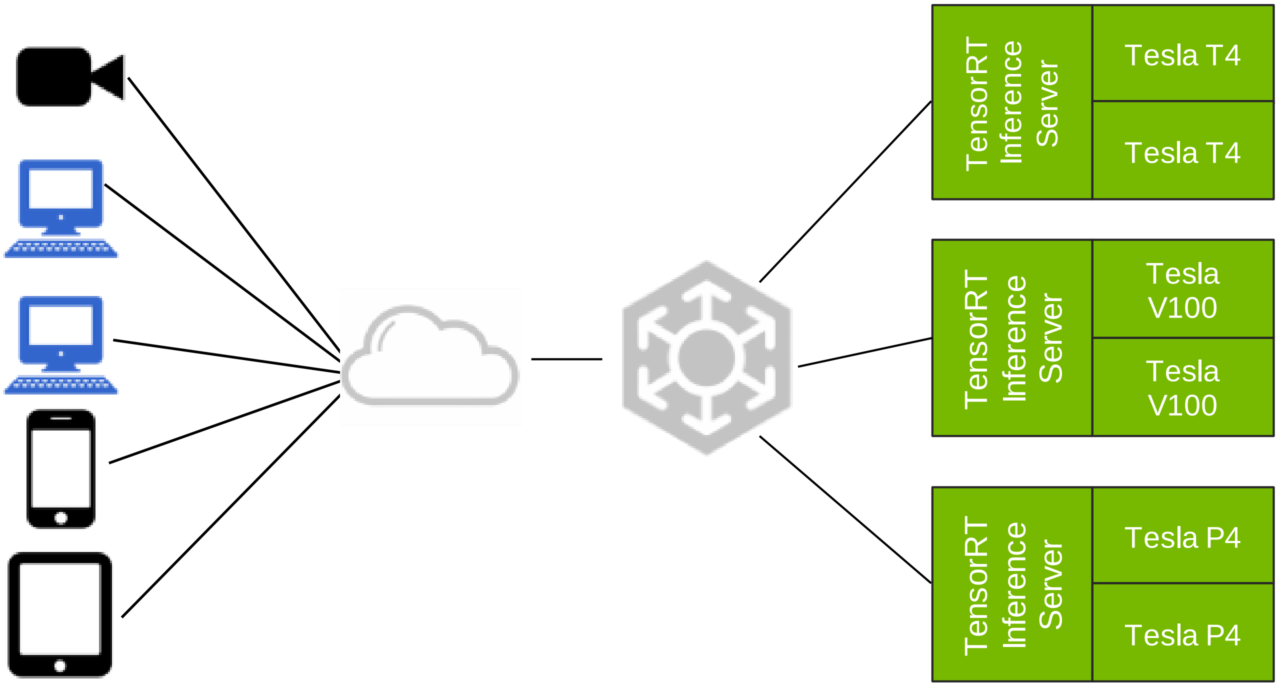
Rich Container Based Pipelines

TF.Transform **TF.Job** **TF.Serving**

```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
             file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
            arguments=[ingestStep.outputs['bucket’]])

servingStep = dsl.ContainerOp(image=tfs_image, <params>,
              arguments=[convertStep.outputs['bucket']])
```

NVIDIA TENSORRT INFERENCE SERVER

Production Data Center Inference Server



Maximize inference throughput & GPU utilization

Quickly deploy and manage multiple models per GPU per node

Easily scale to heterogeneous GPUs and multi GPU nodes

Integrates with orchestration systems and auto scalers via latency and health metrics

Now open source for thorough customization and integration

FEATURES

Concurrent Model Execution
Multiple models (or multiple instances of same model) may execute on GPU simultaneously

Eager Model Loading
Any mix of models specified at server start. All models loaded into memory.

CPU Model Inference Execution
Framework native models can execute inference requests on the CPU

Metrics
Utilization, count, and latency

Custom Backend
Custom backend allows the user more flexibility by providing their own implementation of an execution engine through the use of a shared library

Dynamic Batching
Inference requests can be batched up by the inference server to 1) the model-allowed maximum or 2) the user-defined latency SLA

Multiple Model Format Support
TensorFlow GraphDef/SavedModel
TensorFlow and TensorRT GraphDef
TensorRT Plans
Caffe2 NetDef (ONNX import path)

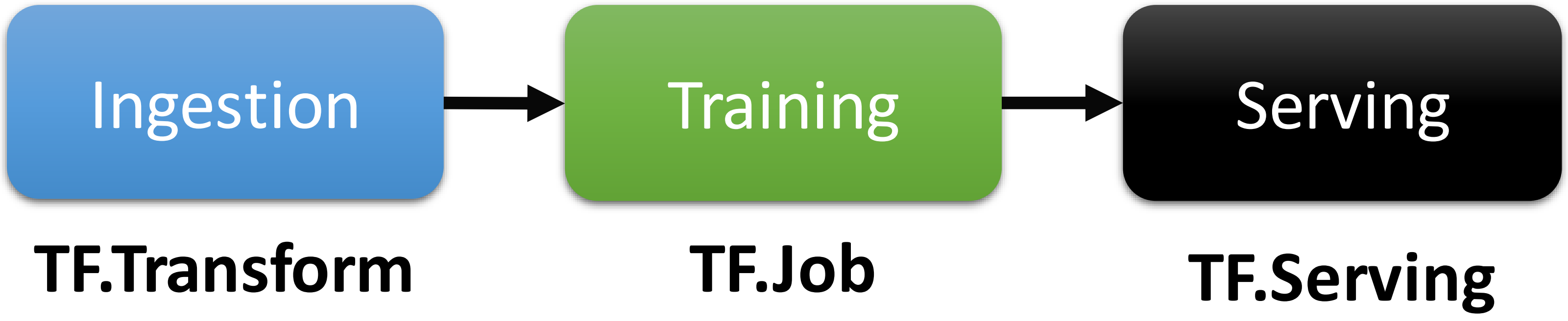
Mounted Model Repository
Models must be stored on a locally accessible mount point

Rich Container Based Pipelines

TF.Transform **TF.Job** **TF.Serving**

```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
             file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
            arguments=[ingestStep.outputs['bucket’]])

servingStep = dsl.ContainerOp(image=tfs_image, <params>,
              arguments=[convertStep.outputs['bucket']])
```

Rich Container Based Pipelines




```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
             file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
            arguments=[ingestStep.outputs['bucket’]])

servingStep = dsl.ContainerOp(image=trt_image, <params>,
              arguments=[convertStep.outputs['bucket']])
```

Rich Container Based Pipelines

TF.Transform | TF.Job | TensorFlow RT

```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
             file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
            arguments=[ingestStep.outputs['bucket’]])

servingStep = dsl.ContainerOp(image=trt_image, <params>,
              arguments=[convertStep.outputs['bucket']])
```

Now, Add a Step

Rich Container Based Pipelines

TF.Transform TF.Job TensorFlow RT

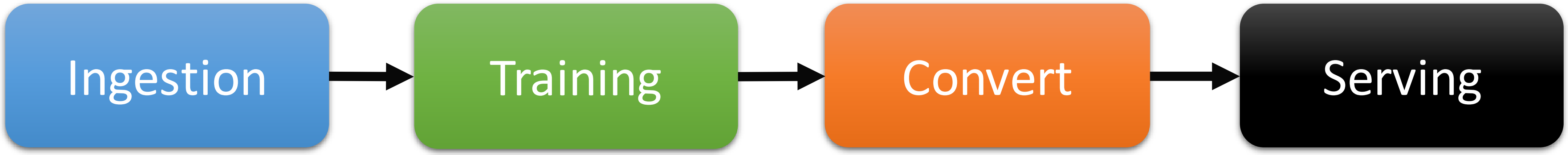
```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
             file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
            arguments=[ingestStep.outputs['bucket’]])

servingStep = dsl.ContainerOp(image=trt_image, <params>,
              arguments=[convertStep.outputs['bucket']])
```

Rich Container Based Pipelines

TF.Transform **TF.Job** **TensorFlow RT**

```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
             file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
            arguments=[ingestStep.outputs['bucket’]])

convertStep = dsl.ContainerOp(image=convert_image, <params>,
              arguments=[trainStep.outputs['bucket’]])

servingStep = dsl.ContainerOp(image=trt_image, <params>,
              arguments=[convertStep.outputs['bucket']])
```

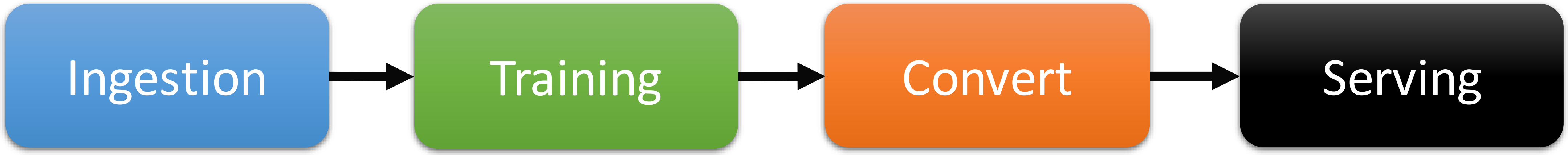
Rich Container Based Pipelines



TF.Transform | TF.Job | ONNX Convert | TensorFlow RT

```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
             file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
            arguments=[ingestStep.outputs['bucket’]])

convertStep = dsl.ContainerOp(image=convert_image, <params>,
              arguments=[trainStep.outputs['bucket’]])

servingStep = dsl.ContainerOp(image=trt_image, <params>,
              arguments=[convertStep.outputs['bucket']])
```

Rich Container Based Pipelines

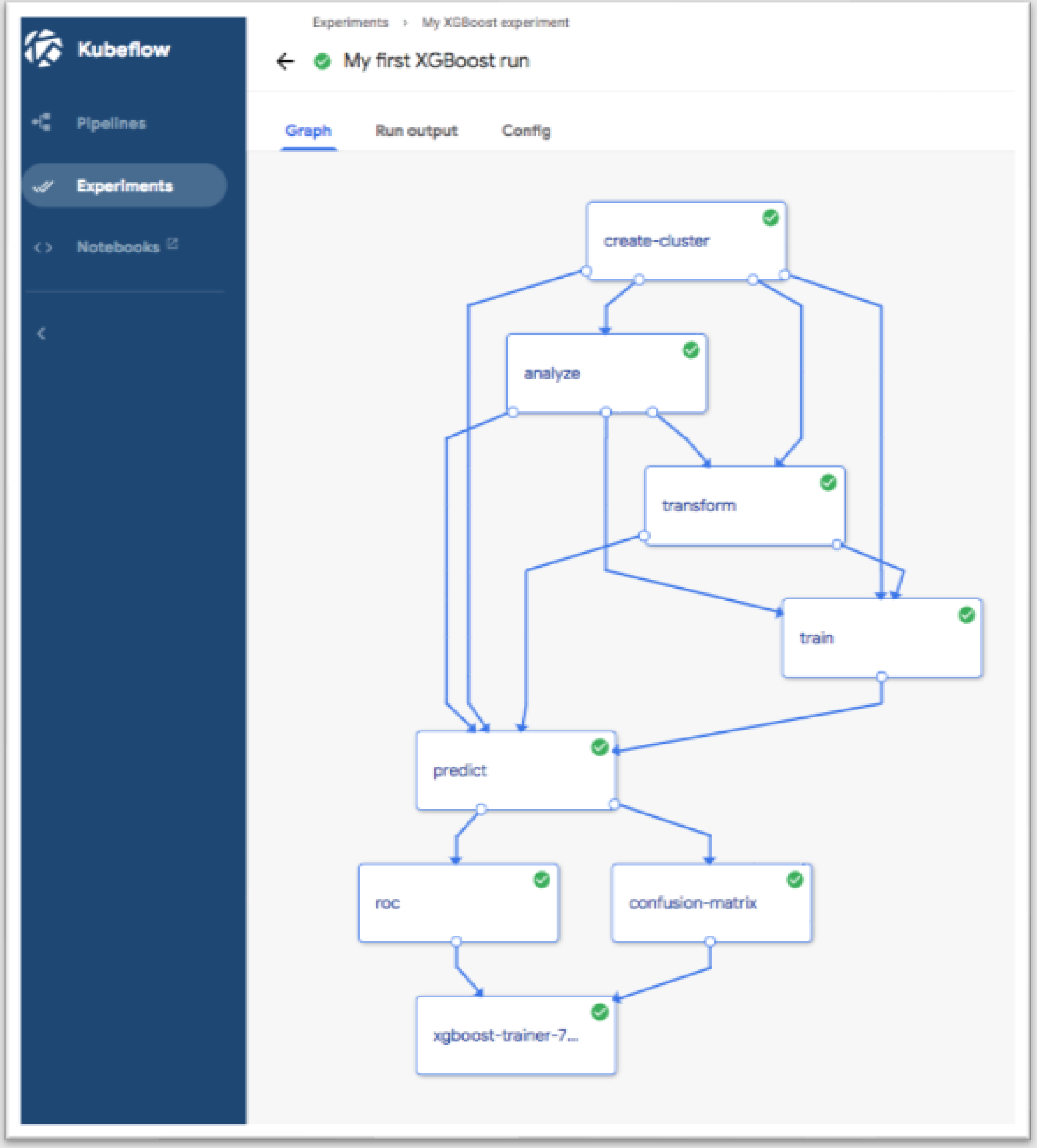


TF.Transform | **TF.Job** | **ONNX Convert** | **TensorFlow RT**

```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
                 file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
                arguments=[ingestStep.outputs['bucket’]])

convertStep = dsl.ContainerOp(image=convert_image, <params>,
                  arguments=[trainStep.outputs['bucket’]])

servingStep = dsl.ContainerOp(image=trt_image, <params>,
                  arguments=[convertStep.outputs['bucket']])
```

Kubeflow Pipeline 0.5

- **UI/UX/SDK improvements**
  - Ability to specify GPU/TPU for pipeline steps
  - Improved job search
  - Metadata Backend to store and query metadata about artifacts produced by pipeline steps
- **Production readiness**
  - Ability to upgrade a cluster without losing information about past runs
  - Lots of stability improvements
- **Improved composability**
  - Define and easily re-use a pipeline component.
  - Compose a larger pipeline using smaller pipelines as building blocks



DEMO

Integrate External Services into Pipeline

User Goal = Just deploy manage it for me

Problem

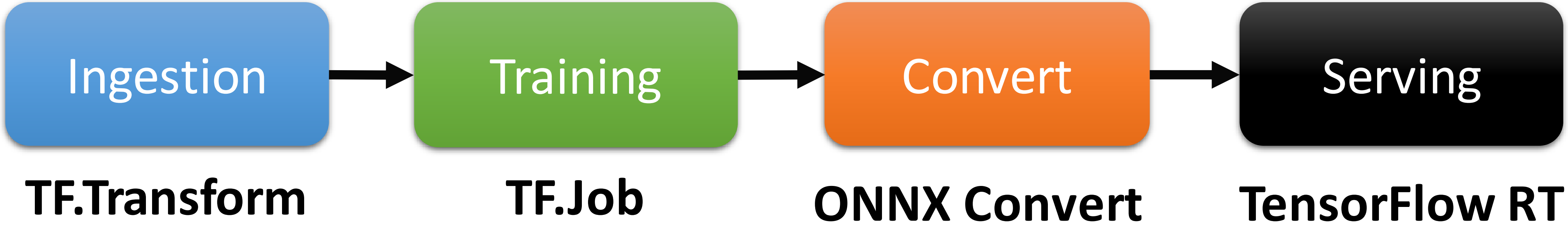
- Self-hosting is customizable but requires (too much) management
- Production requirements for model hosting
- Ability to scale dynamically based on demand, uptime, etc.

Integrate External Services into Pipeline

Solution –

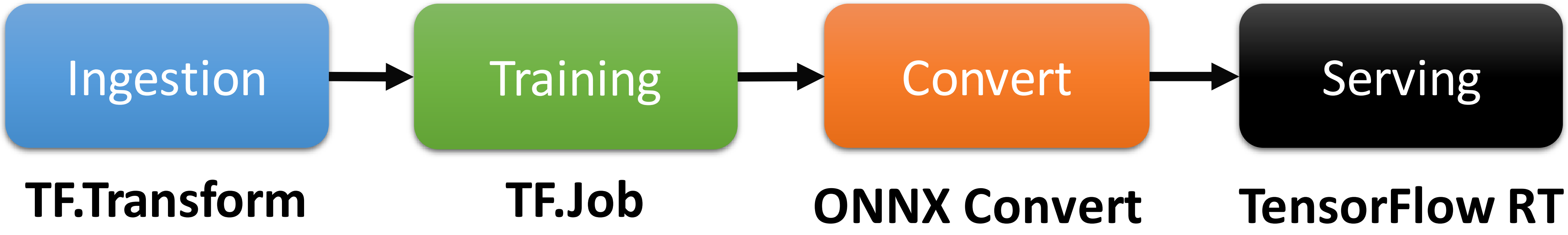
Kubeflow Pipelines!
(Again!)

Rich Container Based Pipelines



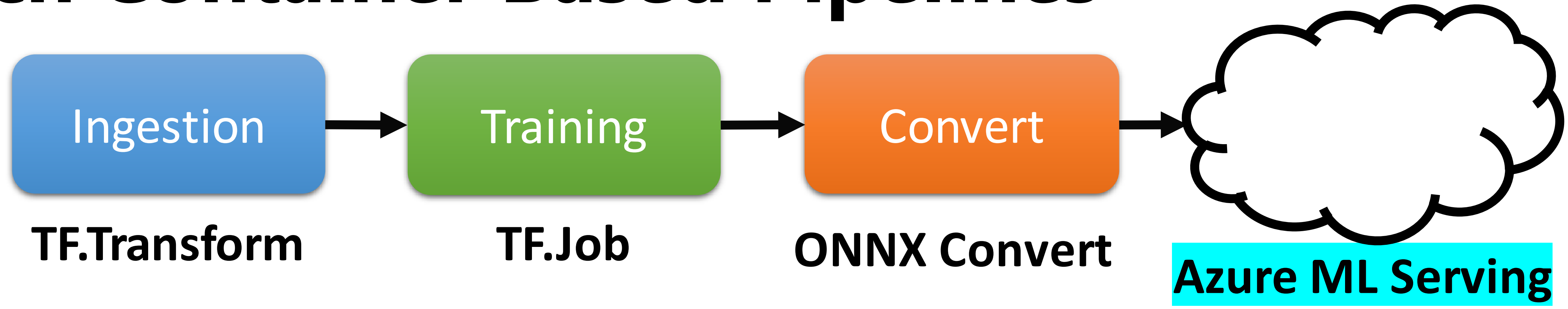

```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
                   file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
                  arguments=[ingestStep.outputs['bucket’]])

convertStep = dsl.ContainerOp(image=convert_image, <params>,
                    arguments=[trainStep.outputs['bucket’]])

servingStep = dsl.ContainerOp(image=trt_image, <params>,
                    arguments=[convertStep.outputs['bucket']])
```

Rich Container Based Pipelines




```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
             file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
            arguments=[ingestStep.outputs['bucket’]])

convertStep = dsl.ContainerOp(image=convert_image, <params>,
              arguments=[trainStep.outputs['bucket’]])

deployStep = dsl.ContainerOp(image=aml_deploy_image, <params>,
              arguments=[convertStep.outputs['bucket']])
```

Rich Container Based Pipelines




```
ingestStep = dsl.ContainerOp(image=tft_image, <params>,
             file_outputs={'bucket': '/output.txt’})

trainStep = dsl.ContainerOp(image=tfjob_image, <params>,
            arguments=[ingestStep.outputs['bucket’]])

convertStep = dsl.ContainerOp(image=convert_image, <params>,
              arguments=[trainStep.outputs['bucket’]])

deployStep = dsl.ContainerOp(image=aml_deploy_image, <params>,
              arguments=[convertStep.outputs['bucket']])
```

DEMO

We’re just getting started!

Our roadmap:

- Transition off of ksonnet
- Infrastructure request/provisioning via Fairing
- Improvements in the notebook manager
- **You tell us!** (Or better yet, help!)

It’s a whole new world

- Data science will touch **EVERY** industry.
- We can’t ask people to become a PhD in statistics though.
- How do **WE** help <u>everyone</u> take advantage of this transformation?

Kubeflow is open!

Open comm-unity

Open design

Open to ideas

Open source

Come Help!

- website: https://kubeflow.org
- github: https://github.com/kubeflow/kubeflow
- slack: kubeflow (http://kubeflow.slack.com)
- twitter: @kubeflow

David Aronchick @aronchick (david.aronchick@microsoft.com)

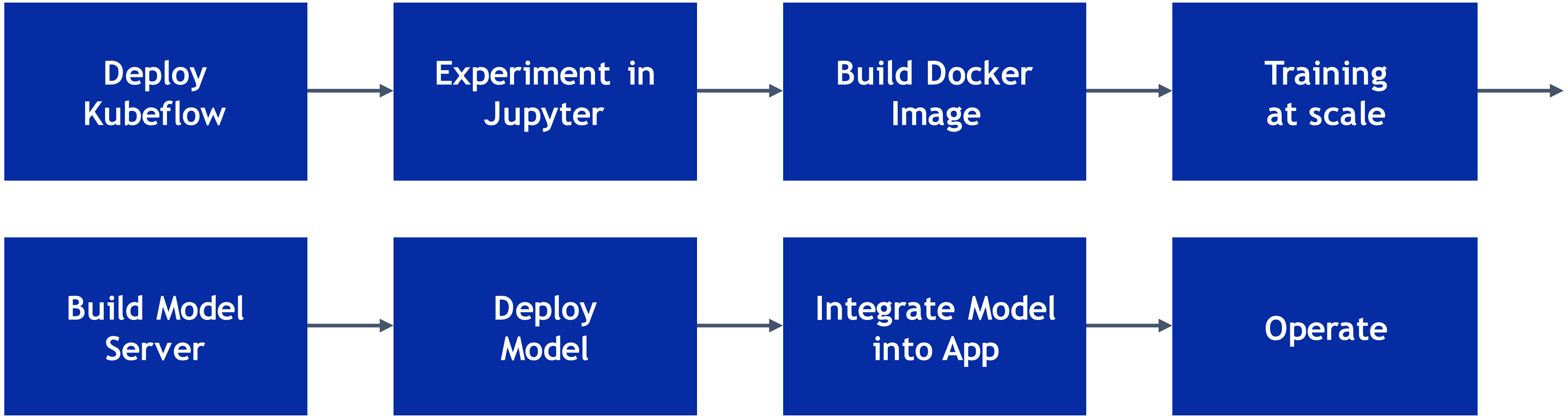
Seth Juarez (sejuare@microsoft.com)

BONEYARD

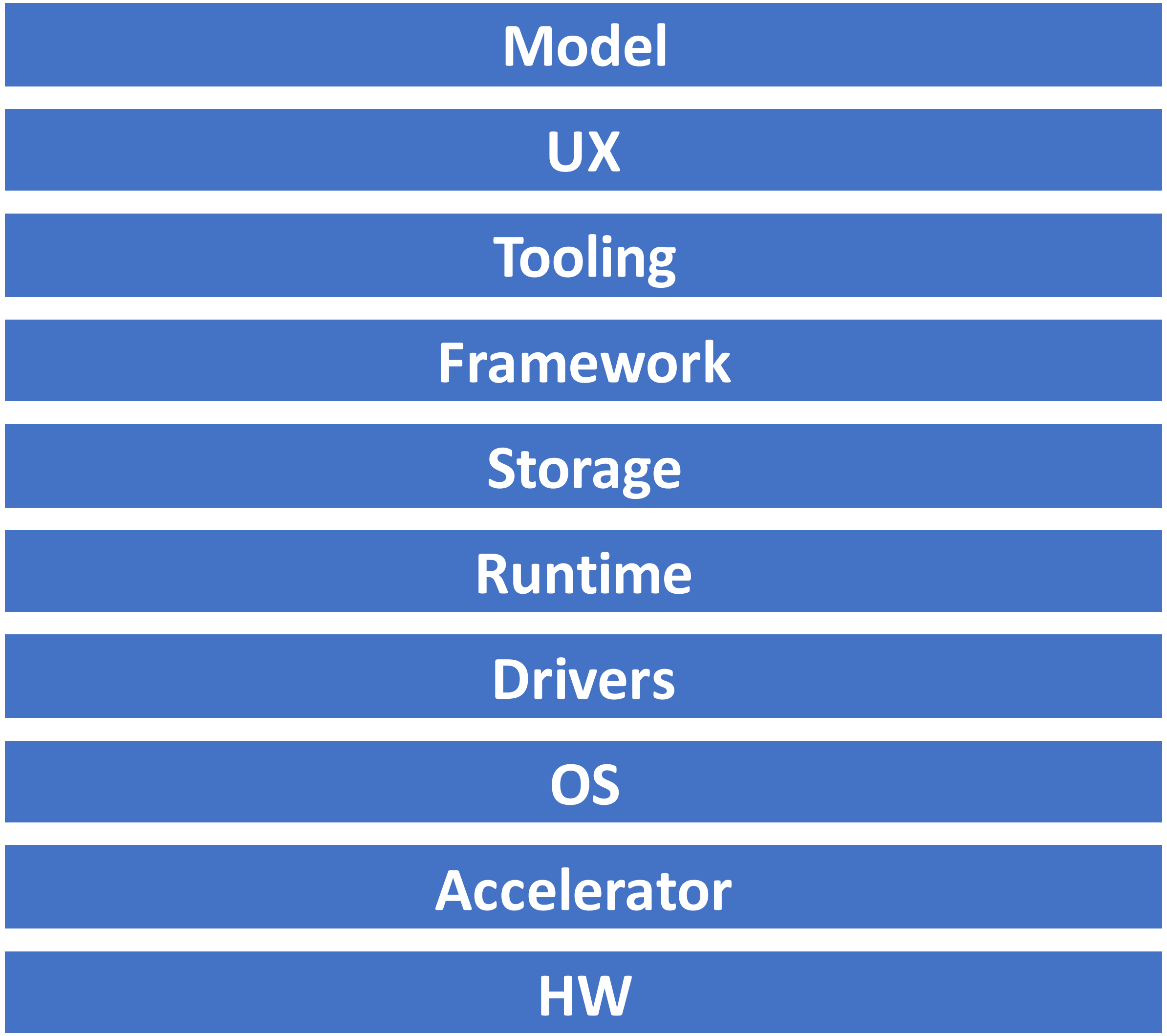
Click to Deploy

- **Problem**: It's too hard to install Kubeflow!
- **Solution**: A one-click installation tool, available via a clean web interface
- **How**:
  - Click to deploy uses a bootstrap container and kfctl.sh with all the necessary dependencies included
  - Also enables use of declarative infrastructure deployment (e.g. Deployment Manager on GCP)
  - **NO TEMPLATING TOOL NEEDED**

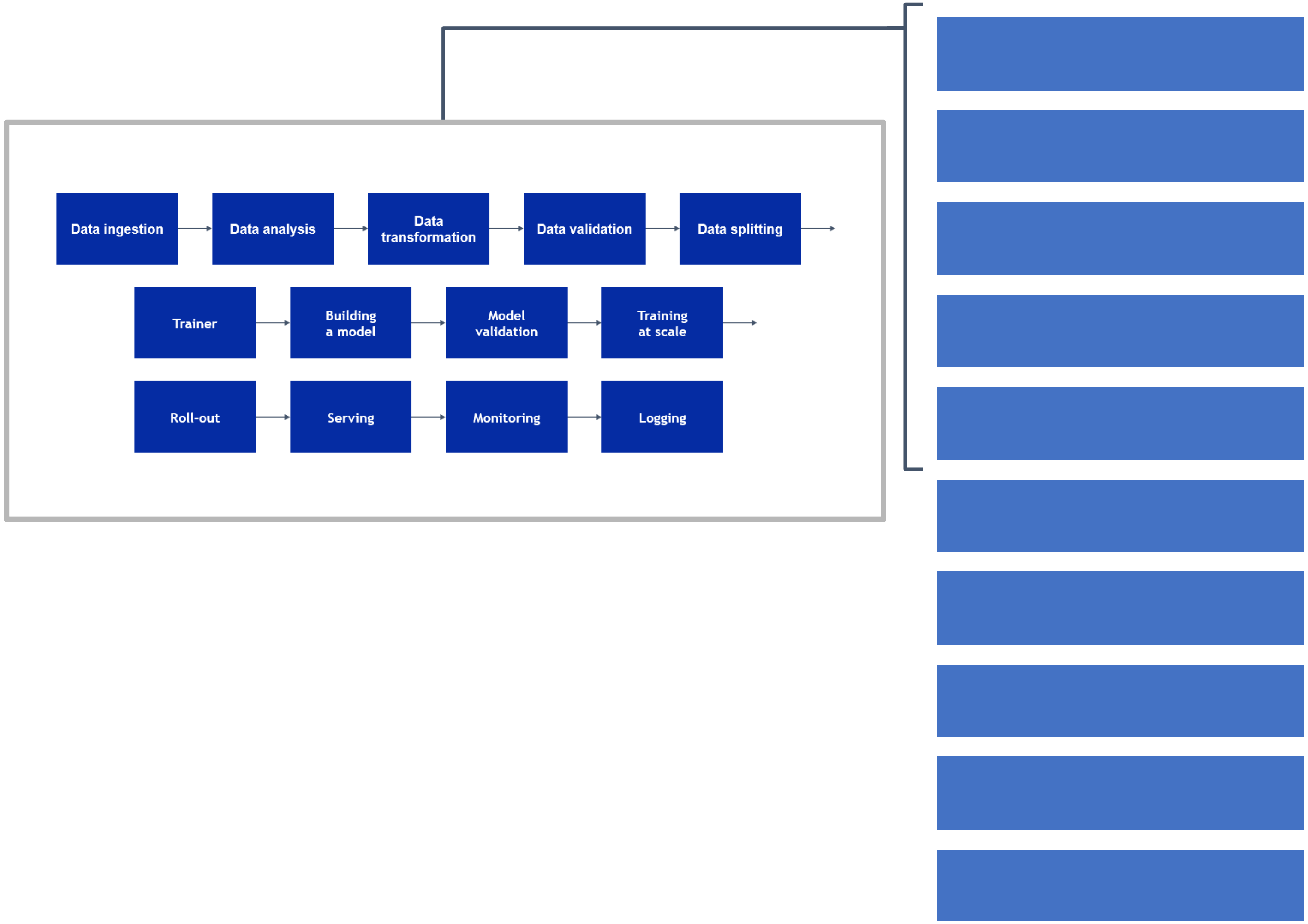
User Experience



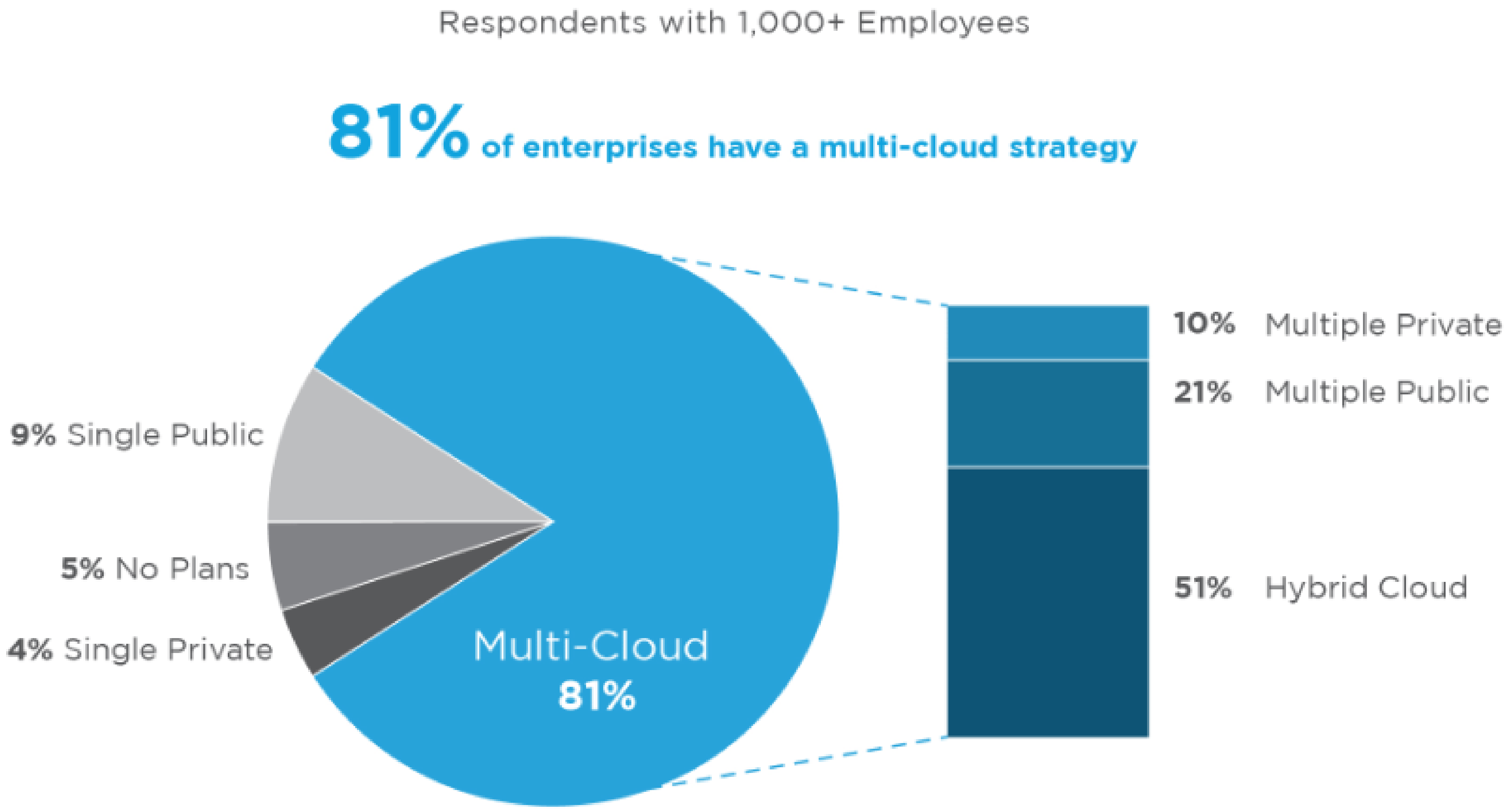
Experimentation



Experimentation



Multi-Cloud is the Reality



Source: RightScale 2018 State of the Cloud Report

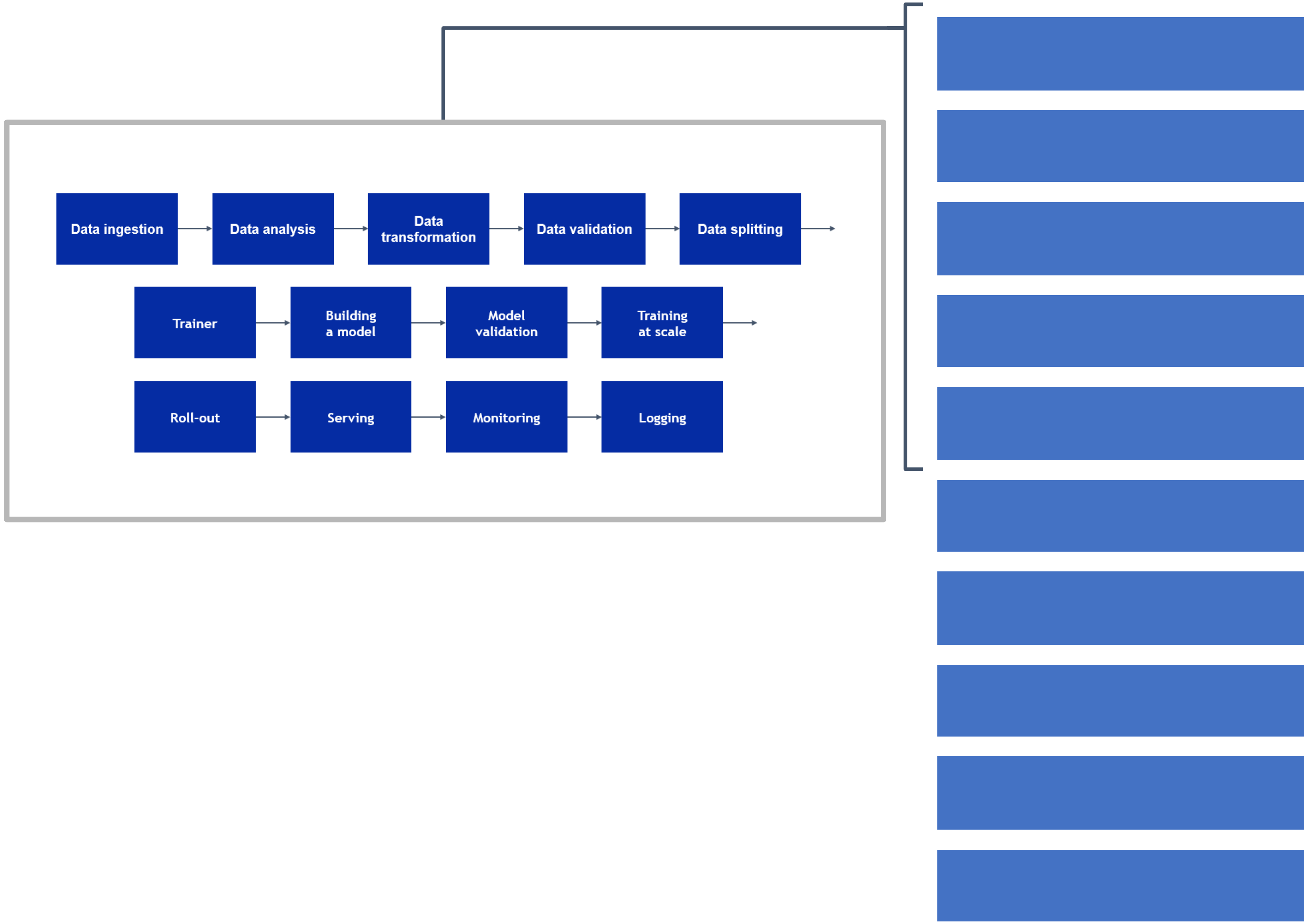
And Not Just One Cloud!

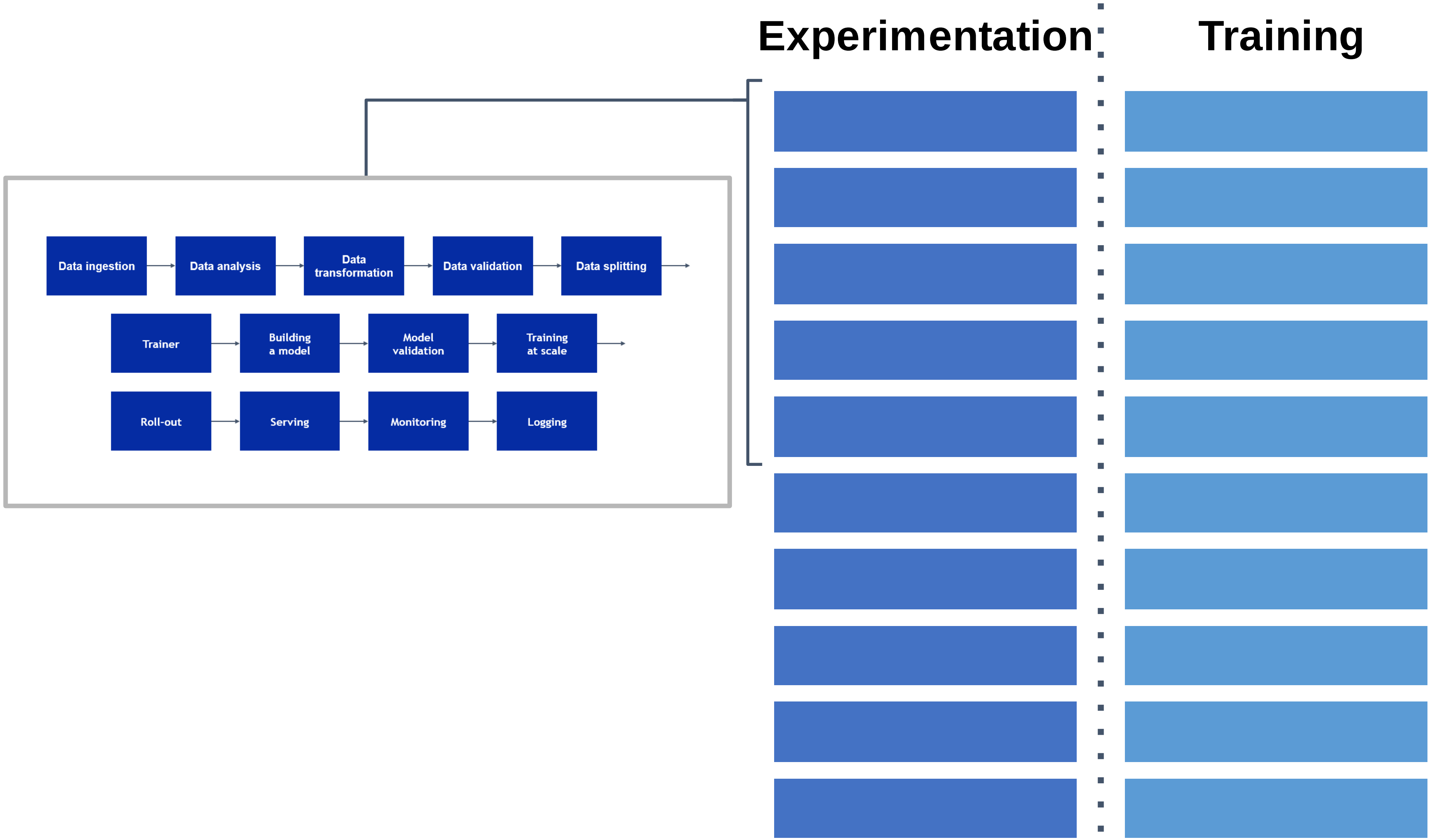
Companies using almost 5 public and private clouds on average

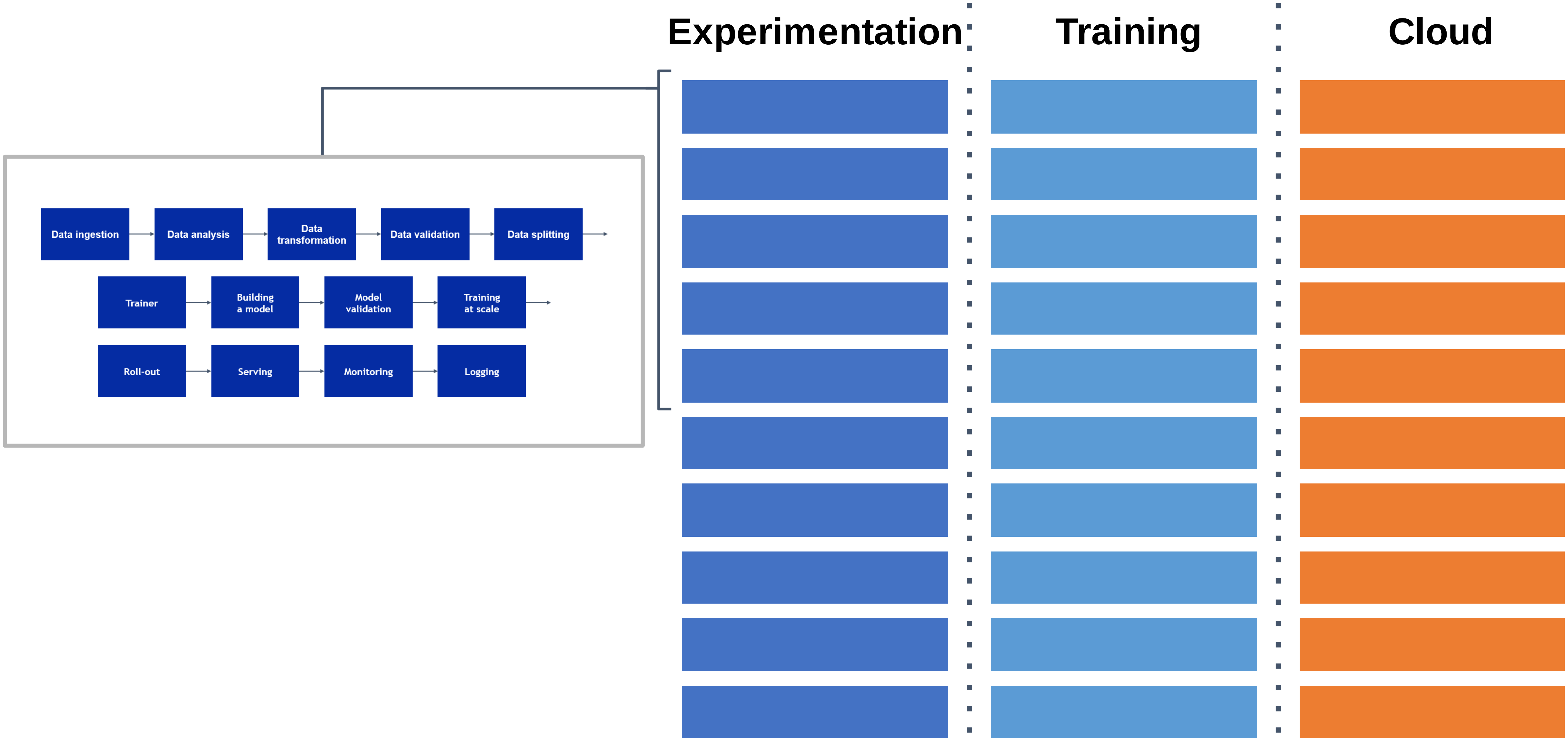
Public + Private Clouds Used	**Average** *All respondents*	**Median** *All respondents*
Running Applications	3.1	3.0
Experimenting	1.7	1.0
Total	**4.8**	**4.0**

Source: RightScale 2018 State of the Cloud Report

Experimentation




Experimentation
Training
Data ingestion
Data analysis
Data transformation
Data validation
Data splitting
Trainer
Building a model
Model validation
Training at scale
Roll-out
Serving
Monitoring
Logging



Experimentation
Training
Cloud
Data ingestion
Data analysis
Data transformation
Data validation
Data splitting
Trainer
Building a model
Model validation
Training at scale
Roll-out
Serving
Monitoring
Logging