

MLRS 2019

**From development to production:
End-to-End Machine Learning and
Deep Learning on GCP, AWS, Azure,
Serverless and edge devices**

Setup your EC2 instance: shorturl.at/elJU3
Download Slides at shorturl.at/GKWy8



**By Warodom Khamphanchai, Ph.D.
Bangkok AI Ambassador**

MLRS 2019
Machine Learning Research School

4th -11th August 2019,
Digital Economy
Promotion Agency, Bangkok

SPEAKERS

Benjamin Van Roy Professor at Stanford University Research Lead at DeepMind, Mountain View	Kenji Fukumizu Professor at The Institute of Statistical Mathematics	Mijung Park "Privacy Preserving Machine Learning" Group Leader at Max Planck Institute for Intelligent Systems	André Martins Research Scientist at Unbabel
Hung Bui Director at VinAI Research	Elias Bareinboim Assistant Professor at Purdue University	Warodom Khamphanchai Bangkok AI Ambassador at CITY.AI	Prachya Boonwan Researcher at NECTEC, Thailand

Apply Now www.mlrs.ai

- Until 7th June 2019 at www.mlrs.ai
- Limited number of travel scholarships available

Time Table

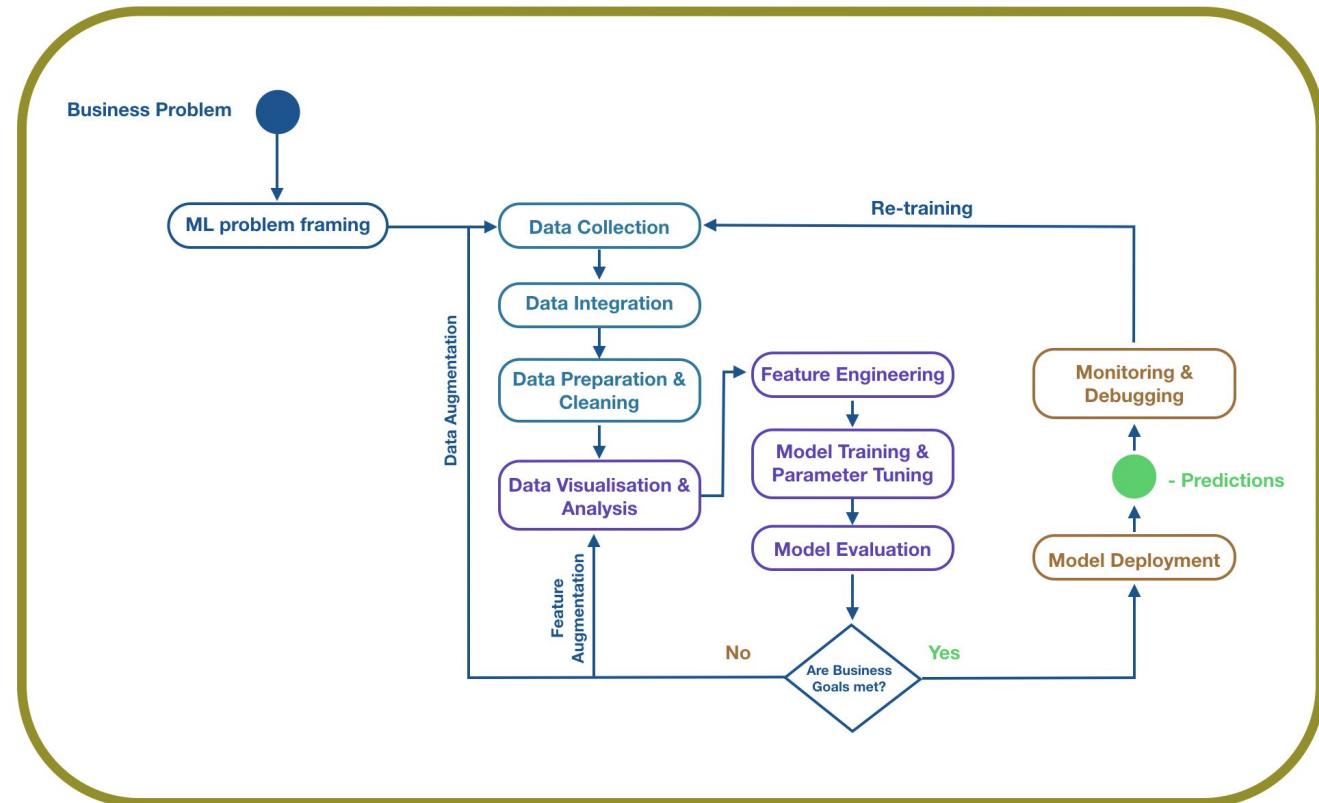
	4-Aug-2019	5-Aug-2019	6-Aug-2019	7-Aug-2019	8-Aug-2019	9-Aug-2019	10-Aug-2019	11-Aug-2019
	SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
6:00-6:30								
6:30-8:30								
8:30-9:00	Registration	Deep Learning I	Deep Learning III	Probabilistic Modeling II	Bus boarding Bus journal from Bangkok to VISTEC	Causal Inference I	Natural Language Processing III	Causal Inference III
9:00-9:30					Intro to VISTEC			
9:30-10:00								
10:00-10:30	Coffee Break				Coffee Break			
10:30-11:00	Welcome and Opening	Reinforcement Learning I	Probabilistic Modeling I	(TBA) II	Panel Discussion	Kernel Methods I	Causal Inference II	Privacy Preserving Machine Learning
11:00-11:30	ML Basics I							
11:30-12:00								
12:00-12:30	Lunch				Lunch			
12:30-13:00								
13:00-13:30	Get-to-Know Event	Poster Session I	Poster Session II	Poster Session III	Campus Tour	Poster Session IV	Poster Session V	Thai NLP: Challenges and Future Directions
13:30-14:00								
14:00-14:30								
14:30-15:00	ML Basics II	Reinforcement Learning II	(TBA) I	Probabilistic Modeling III	Workshops (8)	Natural Language Processing II	Kernel Methods II	Machine learning and AI Research in Thailand
15:00-15:30								
15:30-16:00	Coffee Break				Coffee Break			
16:00-16:30								
16:30-17:00	ML Basics III	Deep Learning II	Coding Session I	Natural Language Processing I	Workshop Presentations (incl. Dinner)	Coding Session II	Session by DEPA and AVALANT	Closing and Award Ceremony
17:00-17:30								
17:30-18:00								
18:00-18:30	Welcome Party	(An event organized by a diamond sponsor.)		Free Time for Open Discussion, Snacks Provided.	(An event organized by a diamond sponsor.)	Departure to Bangkok	Banquet	(Unscheduled. Free slot for an event by a diamond sponsor.)
18:30-19:00								
19:00-19:30								
19:30-20:00								
20:00-20:30								

Agenda

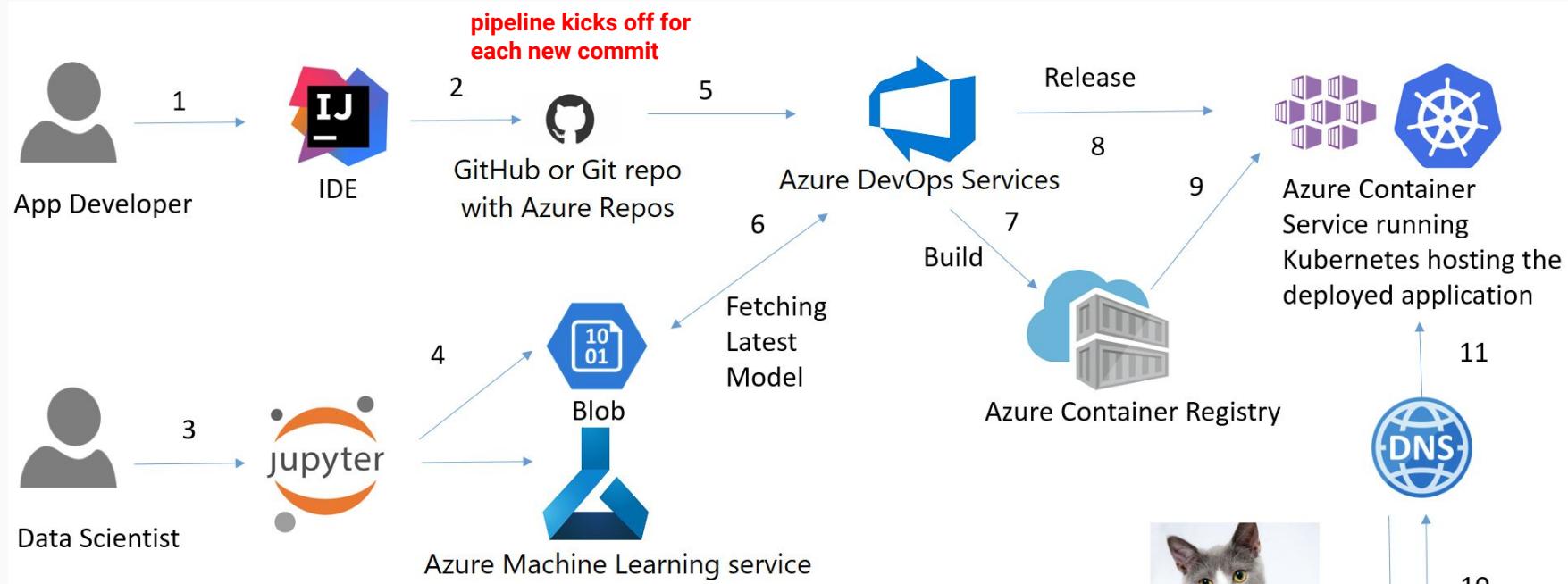
1. Basics DevOps
2. Local Model Deployment
3. End-to-end ML, DL model with Tensorflow Extended and Tensorflow Serving
4. End-to-end ML, DL with Serverless Framework (AWS Lambda)
5. End-to-end ML, DL model on MS Azure
6. End-to-end ML, DL model on AWS SageMaker
7. End-to-end Tensorflow Lite on Edge Device (Raspberry Pi)
8. Continuous Learning

Typical Machine Learning Process

The Machine Learning Process



Comparing CI/CD Pipeline - App Developer vs Data Scientist



This **decouples the app developers and data scientists**, to make sure that their production app is always running the latest code with latest ML model.

```
[ { "cat": 0.99218,  
  "feline": 0.81242,  
  "puma": 0.45456: } ]
```

Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips

{dsculley, gholt, dg, edavydov, toddphillips}@google.com
Google, Inc.

Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, Dan Dennison

{ebner, vchaudhary, mwyong, jfcrespo, dennison}@google.com
Google, Inc.

Abstract

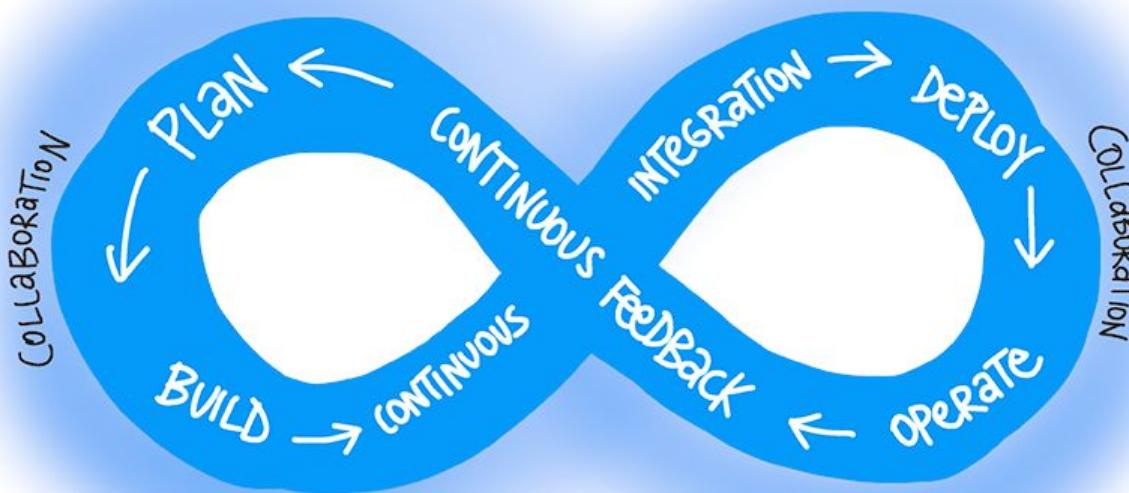
Machine learning offers a fantastically powerful toolkit for building useful complex prediction systems quickly. This paper argues it is dangerous to think of these quick wins as coming for free. Using the software engineering framework of *technical debt*, we find it is common to incur massive ongoing maintenance costs in real-world ML systems. We explore several ML-specific risk factors to account for in system design. These include boundary erosion, entanglement, hidden feedback loops, undeclared consumers, data dependencies, configuration issues, changes in the external world, and a variety of system-level anti-patterns.

1. Basics

DevOps Basics for ML, DL Developers

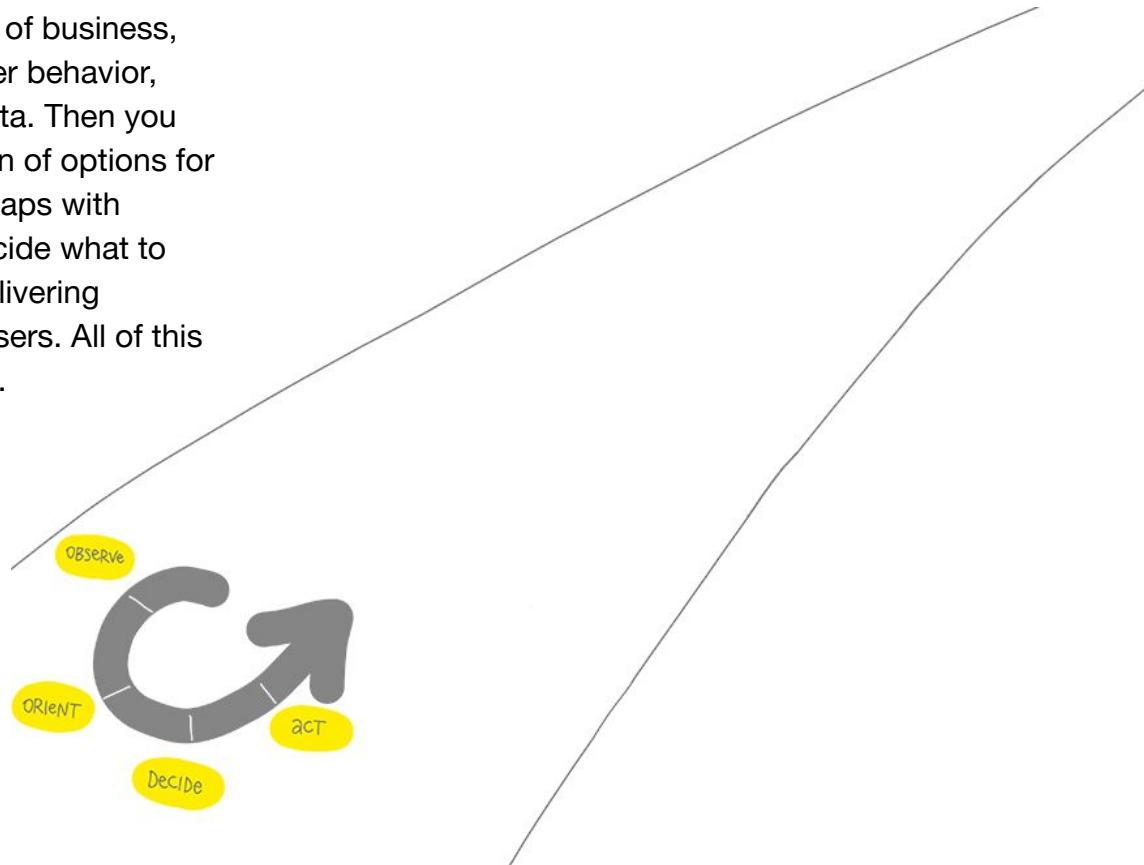
DevOps

DevOps is the union of people, process, and products to enable continuous delivery of value to our end users. The contraction of “Dev” and “Ops” refers to replacing siloed Development and Operations to create multidisciplinary teams that now work together with shared and efficient practices and tools. Essential DevOps practices include agile planning, continuous integration, continuous delivery, and monitoring of applications.



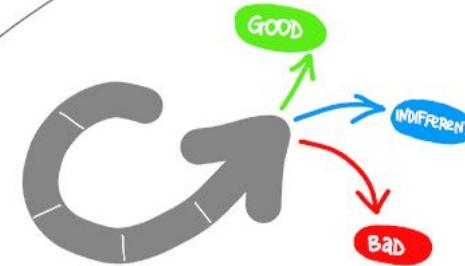
DevOps - Understand your Cycle Time

You start with observation of business, market, needs, current user behavior, and available telemetry data. Then you orient with the enumeration of options for what you can deliver, perhaps with experiments. Next you decide what to pursue, and you act by delivering working software to real users. All of this occurs in some cycle time.



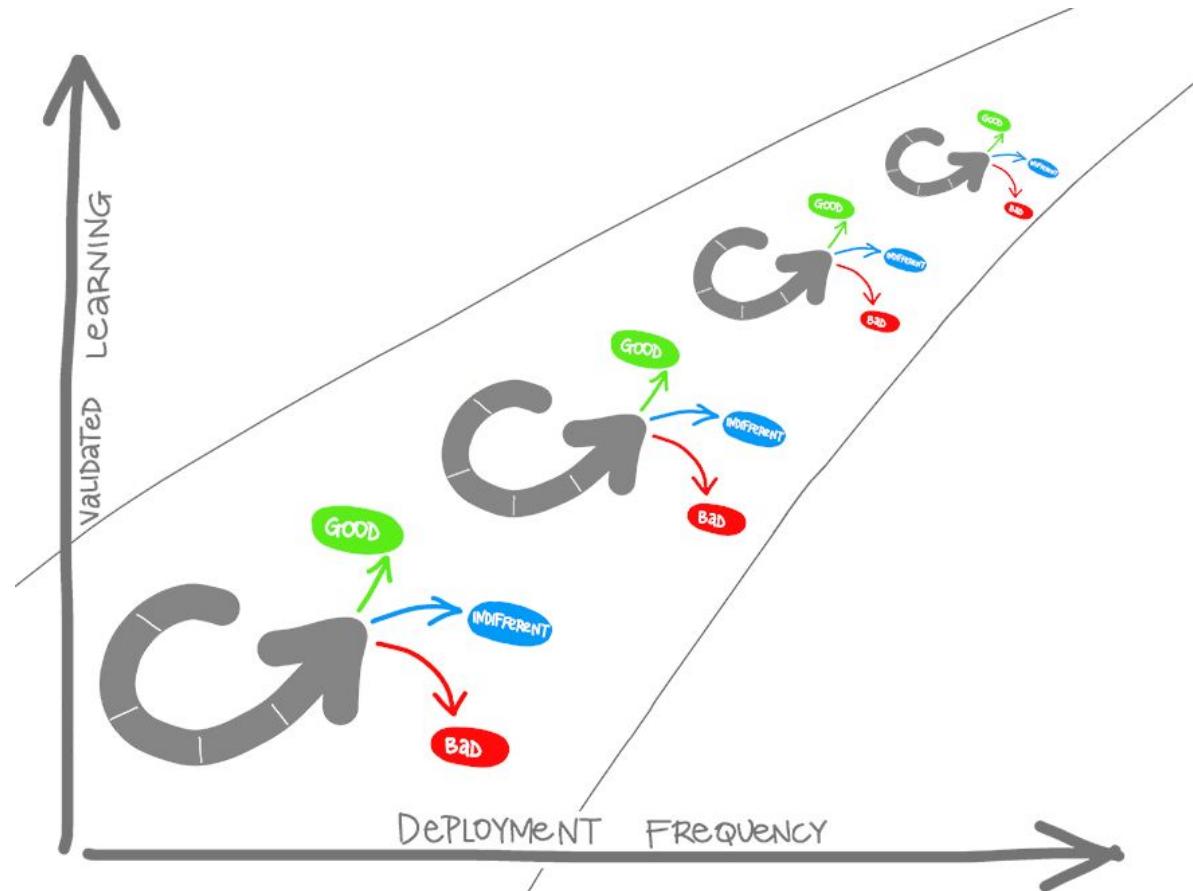
DevOps - Strive for Validated Learning

How quickly you can fail fast or double down is determined by how long that loop takes, or in lean terms, by your cycle time. Your cycle time determines how quickly you can gather feedback to determine what happens in the next loop. The feedback that you gather with each cycle should be real, actionable data. This is called validated learning.



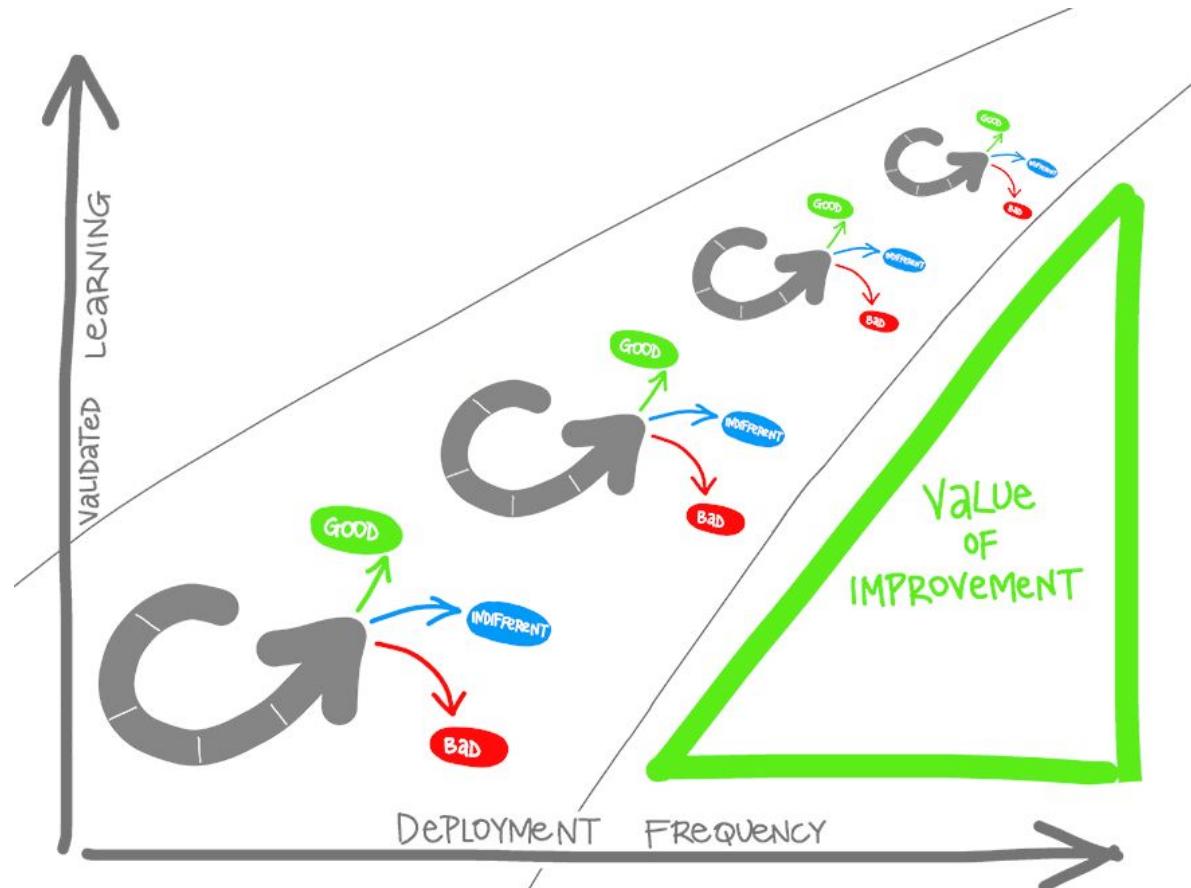
DevOps - Shorten Your Cycle Time

When you adopt DevOps practices, you shorten your cycle time by working in smaller batches, using more automation, hardening your release pipeline, improving your telemetry, and deploying more frequently.



DevOps - Optimize Validated Learning

The more frequently you deploy, the more you can experiment, the more opportunity you have to pivot or persevere, and to gain validated learning each cycle. This acceleration in validated learning is the value of improvement. Think of it as the sum of improvements that you achieve and the failures that you avoid.



DevOps - How to Achieve DevOps > Continuous Integration (CI)

Goal: shorten cycle time.

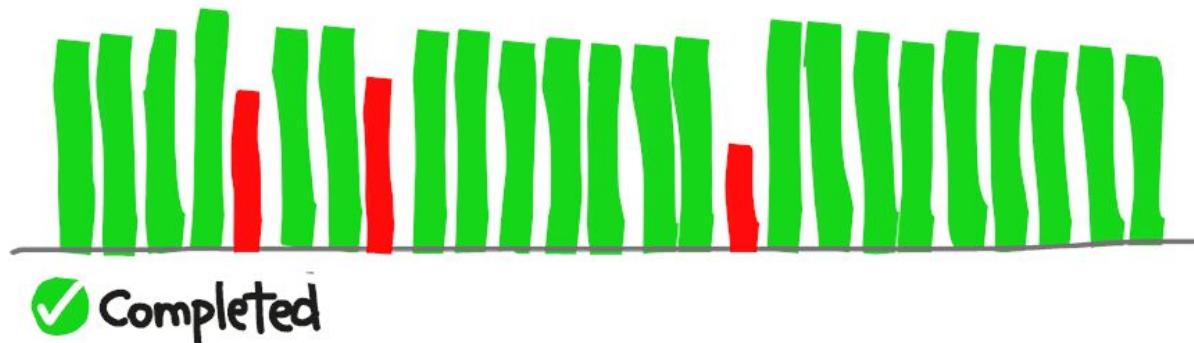
Question: How long does it take you to deploy a change of one line of code or configuration? Ultimately, that's the brake on your velocity.

Continuous Integration drives the

ongoing merging and testing of code, which leads to finding defects early.

Other benefits include less time wasted on fighting merge issues and rapid feedback for development teams.

BUILD Succeeded

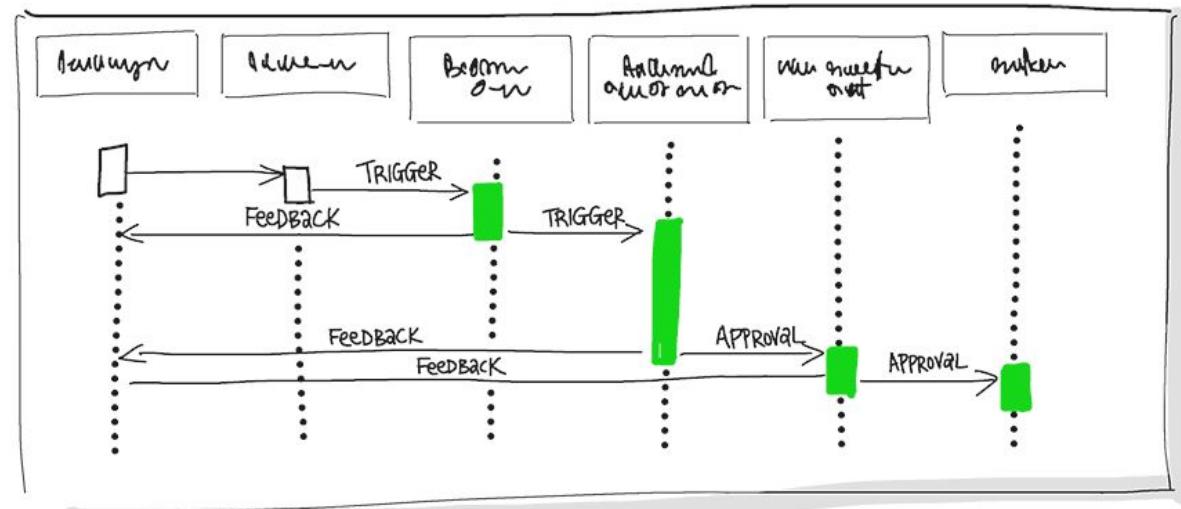


DevOps - How to Achieve DevOps > Continuous Delivery (CD)

Goal: shorten cycle time.

Question: How long does it take you to deploy a change of one line of code or configuration? Ultimately, that's the brake on your velocity.

Continuous Delivery of software solutions to production and testing environments helps organizations quickly fix bugs and respond to ever-changing business requirements



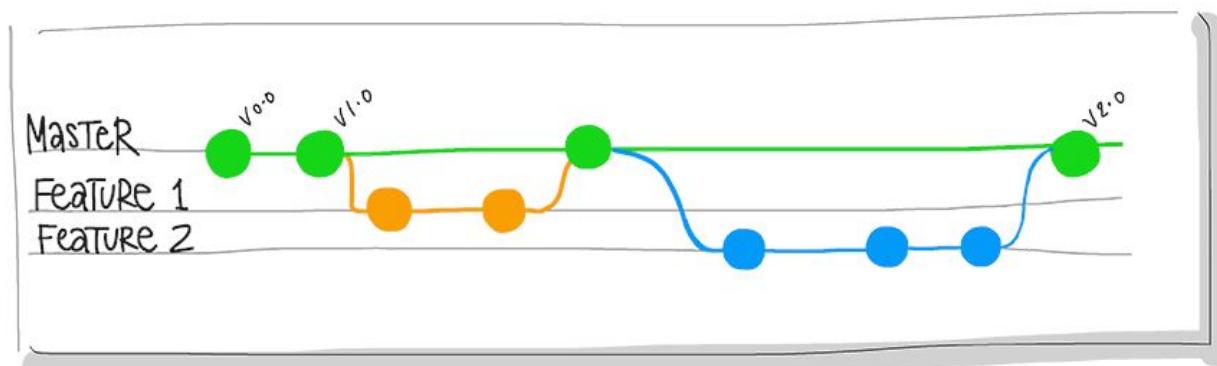
DevOps - How to Achieve DevOps > Version Control

Goal: shorten cycle time.

Question: How long does it take you to deploy a change of one line of code or configuration? Ultimately, that's the brake on your velocity.

Version Control, Usually With Git,

enables teams located anywhere in the world to communicate effectively during daily development activities as well as to integrate with software development tools for monitoring activities such as deployments



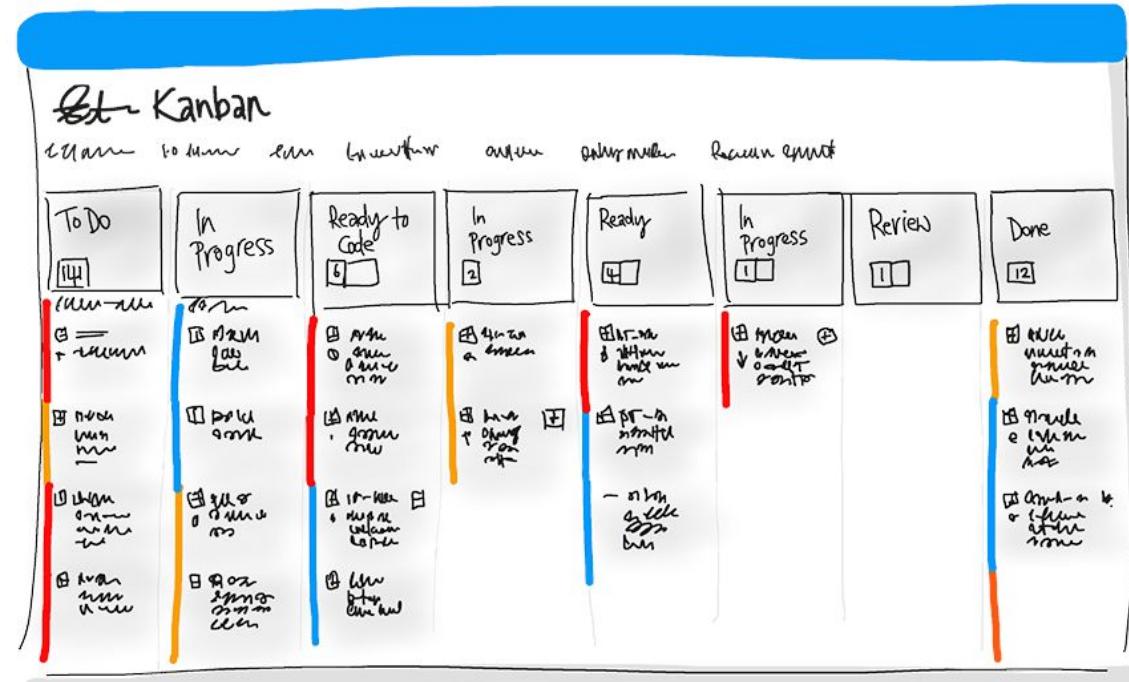
DevOps - How to Achieve DevOps > Kanban

Goal: shorten cycle time.

Question: How long does it take you to deploy a change of one line of code or configuration? Ultimately, that's the brake on your velocity.

Agile planning and lean project management

techniques are used to plan and isolate work into sprints, manage team capacity, and help teams quickly adapt to changing business needs. A DevOps Definition of Done is working software collecting telemetry against the intended business objectives.

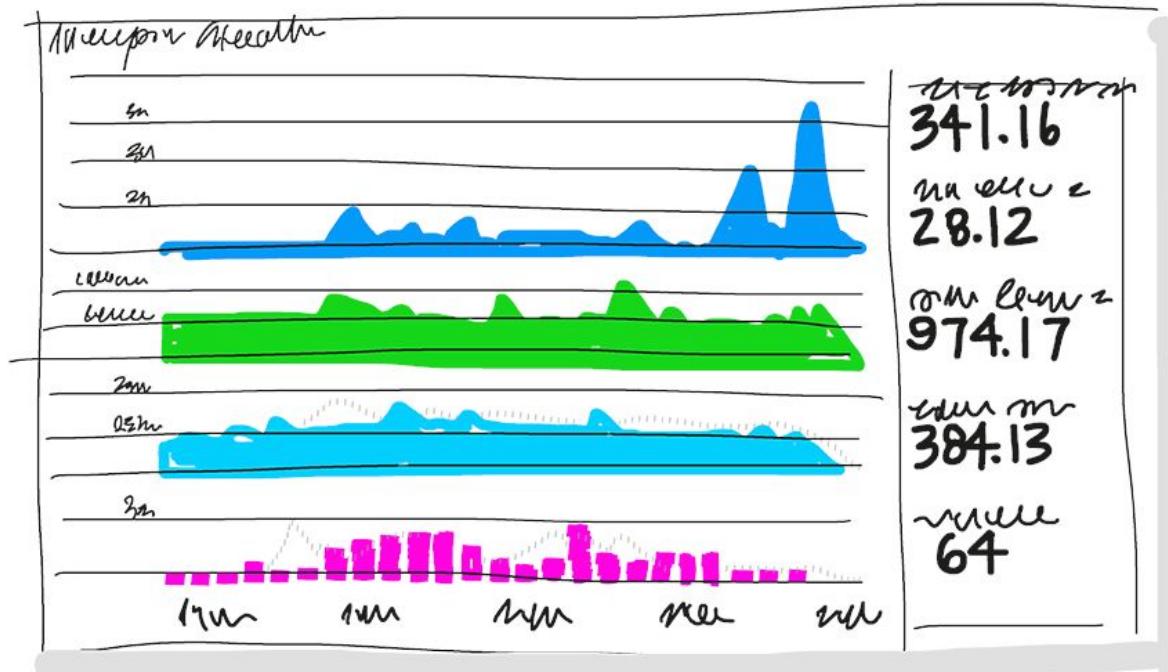


DevOps - How to Achieve DevOps > Monitoring and Logging

Goal: shorten cycle time.

Question: How long does it take you to deploy a change of one line of code or configuration? Ultimately, that's the brake on your velocity.

Monitoring and Logging of running applications including production environments for application health as well as customer usage, helps organizations form a hypothesis and quickly validate or disprove strategies. Rich data is captured and stored in various logging formats.



DevOps - How to Achieve DevOps > Public and Hybrid Clouds

Goal: shorten cycle time.

Question: How long does it take you to deploy a change of one line of code or configuration? Ultimately, that's the brake on your velocity.

Public and Hybrid Clouds have made the impossible easy. The cloud has removed traditional bottlenecks and helped commoditize infrastructure. Whether you use Infrastructure as a Service (IaaS) to lift and shift your existing apps, or Platform as a Service (PaaS) to gain unprecedented productivity, the cloud gives you a datacenter without limits.

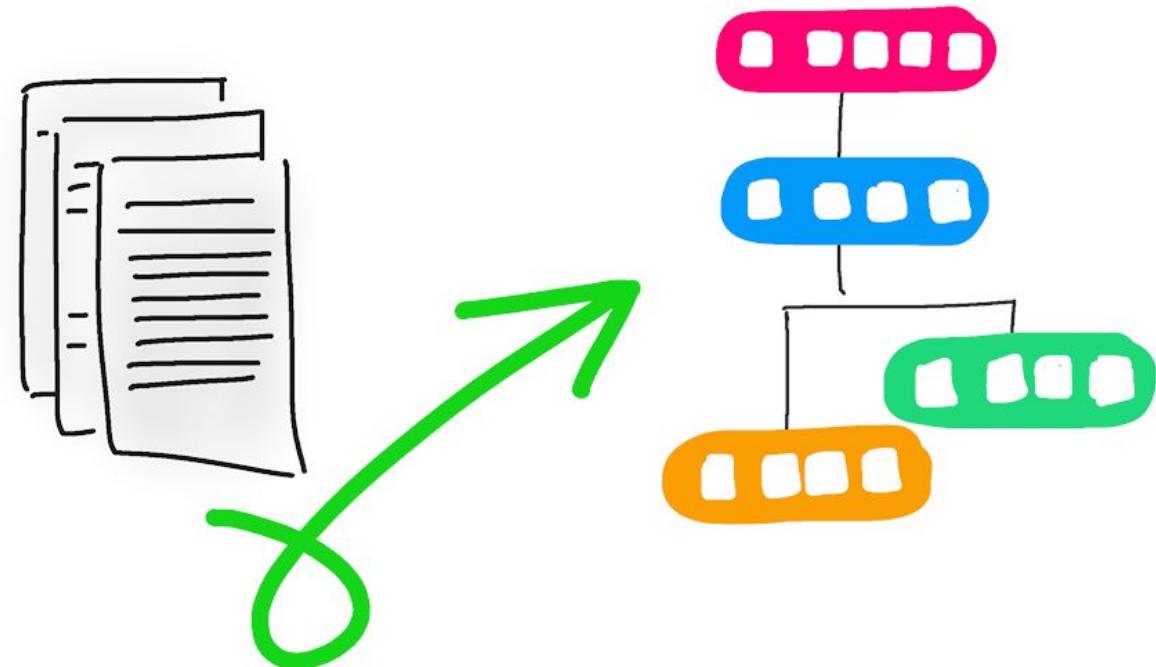


DevOps - How to Achieve DevOps > Infrastructure as Code

Goal: shorten cycle time.

Question: How long does it take you to deploy a change of one line of code or configuration? Ultimately, that's the brake on your velocity.

Infrastructure as Code (IaC) is a practice which enables the automation and validation of creation and teardown of environments to help with delivering secure and stable application hosting platforms.

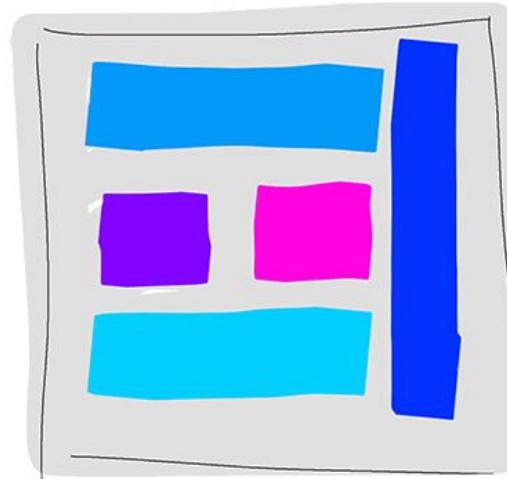


DevOps - How to Achieve DevOps > Microservices

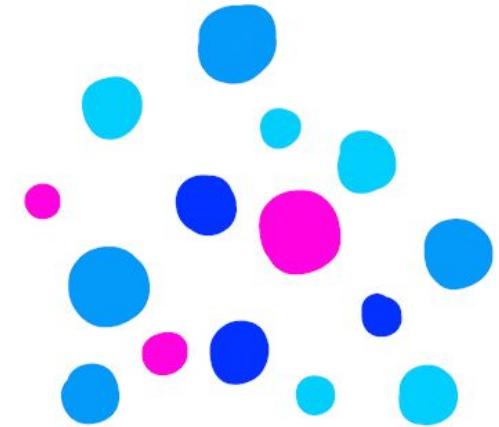
Goal: shorten cycle time.

Question: How long does it take you to deploy a change of one line of code or configuration? Ultimately, that's the brake on your velocity.

Microservices architecture is leveraged to isolate business use cases into small reusable services that communicate via interface contracts. This architecture enables scalability and efficiency.



MONOLITHIC/LAYERED



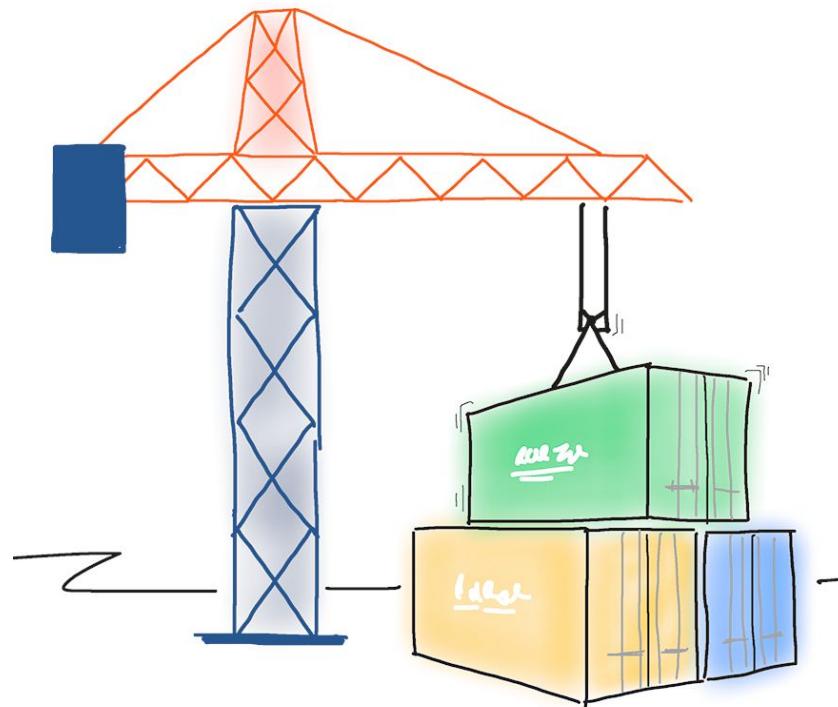
MICROSERVICES

DevOps - How to Achieve DevOps > Containers

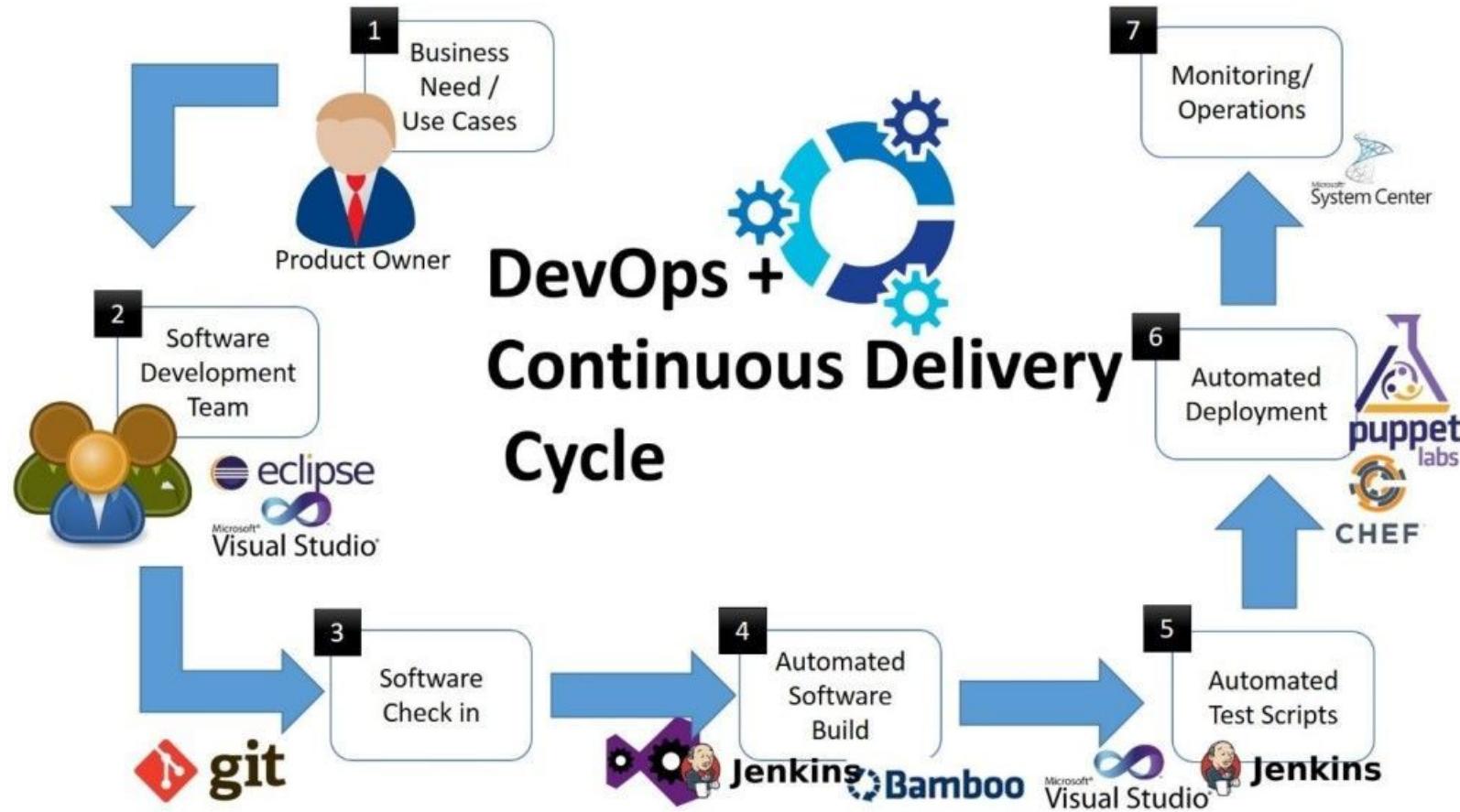
Goal: shorten cycle time.

Question: How long does it take you to deploy a change of one line of code or configuration? Ultimately, that's the brake on your velocity.

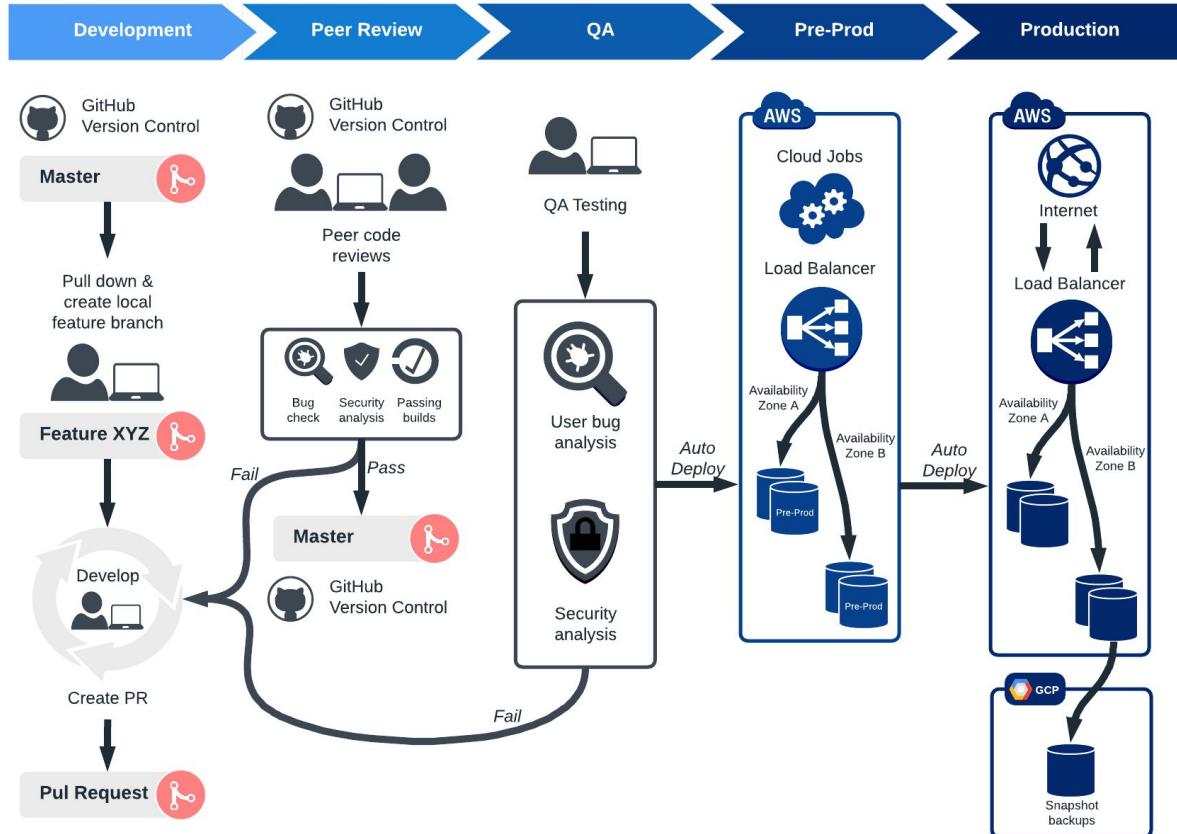
Containers are the next evolution in virtualization. They are much more lightweight than virtual machines, allow much faster hydration, and can be easily configured from files.



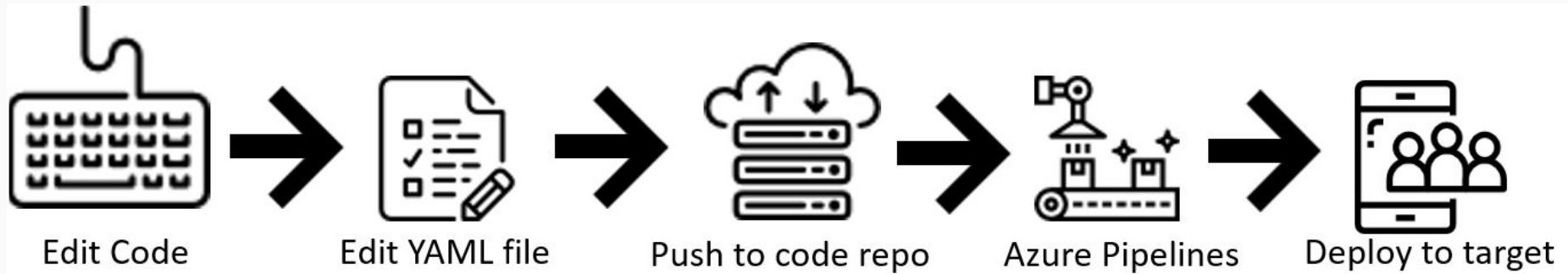
DevOps - Summary



DevOps - Summary



1. Basic DevOps



Why Continuous Delivery?

Time to market goes down

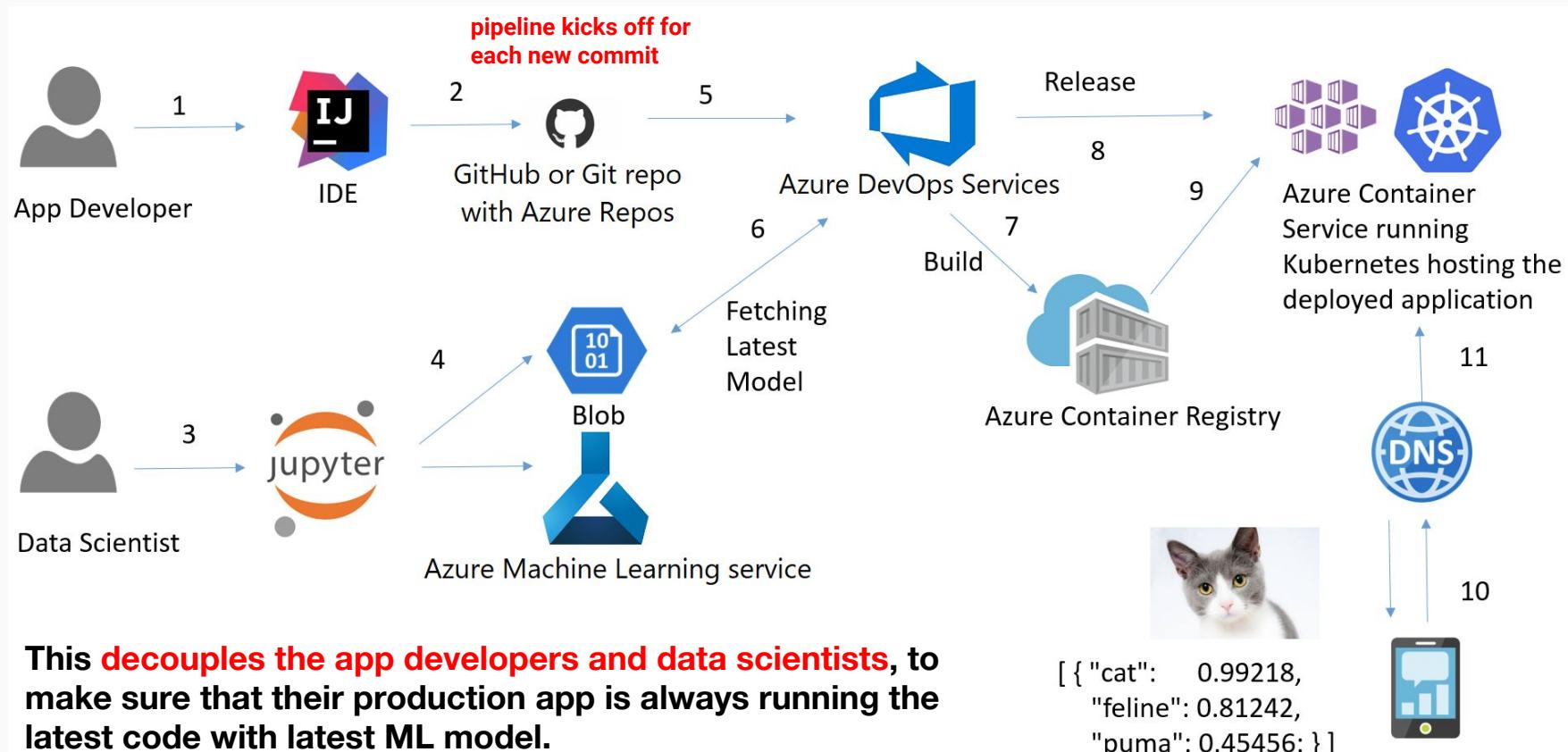
Quality increases, not decreases

Limit your work in progress

Shortens lead times for changes

Improves mean time to recover

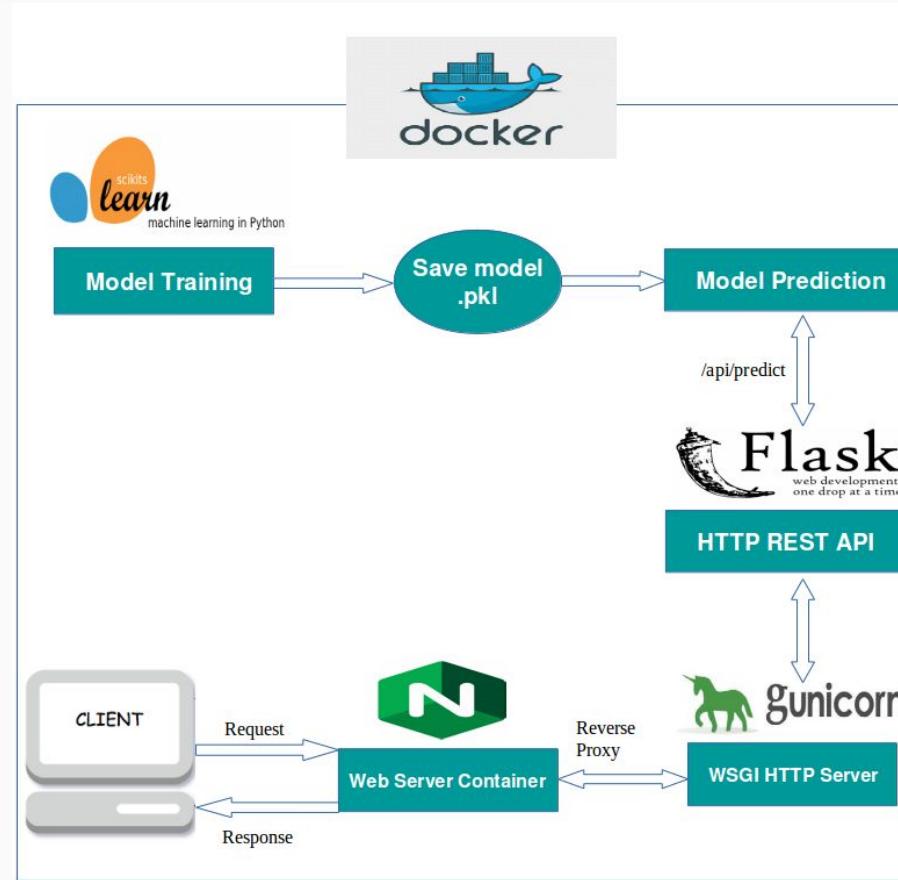
Creating CI/CD Pipeline on Azure using Docker, Kubernetes, and Python Flask



2. Local

Local ML Model Deployment

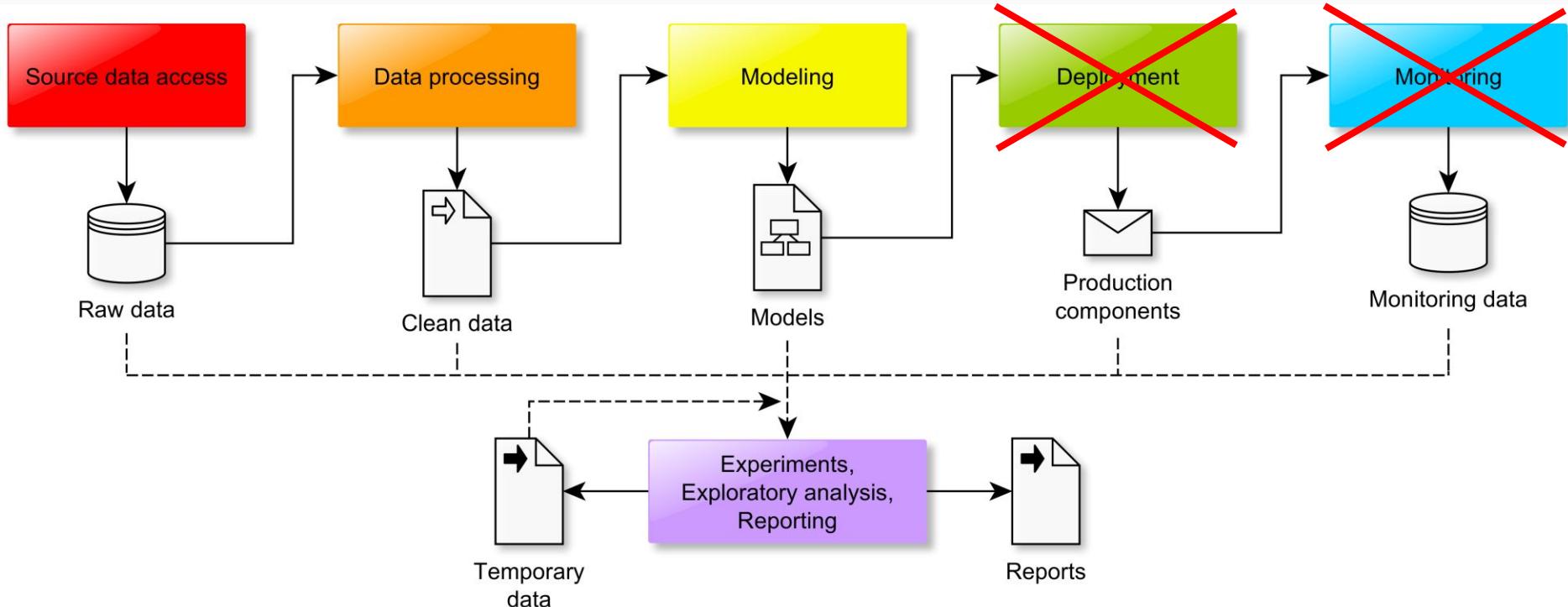
2. Local Model Deployment



3. TFX and TF Serving

Tensorflow
serving to
serve on GCP

General machine learning research workflow



TFX (Tensorflow extended)

TensorFlow Extended (TFX) is an end-to-end platform for deploying production ML pipelines

When you're ready to move your models from research to production, use TFX to create and manage a production pipeline.

[See tutorials](#)

[See the guide](#)

Tutorials show you how to use TFX with complete, end-to-end examples.

Guides explain the concepts and components of TFX.

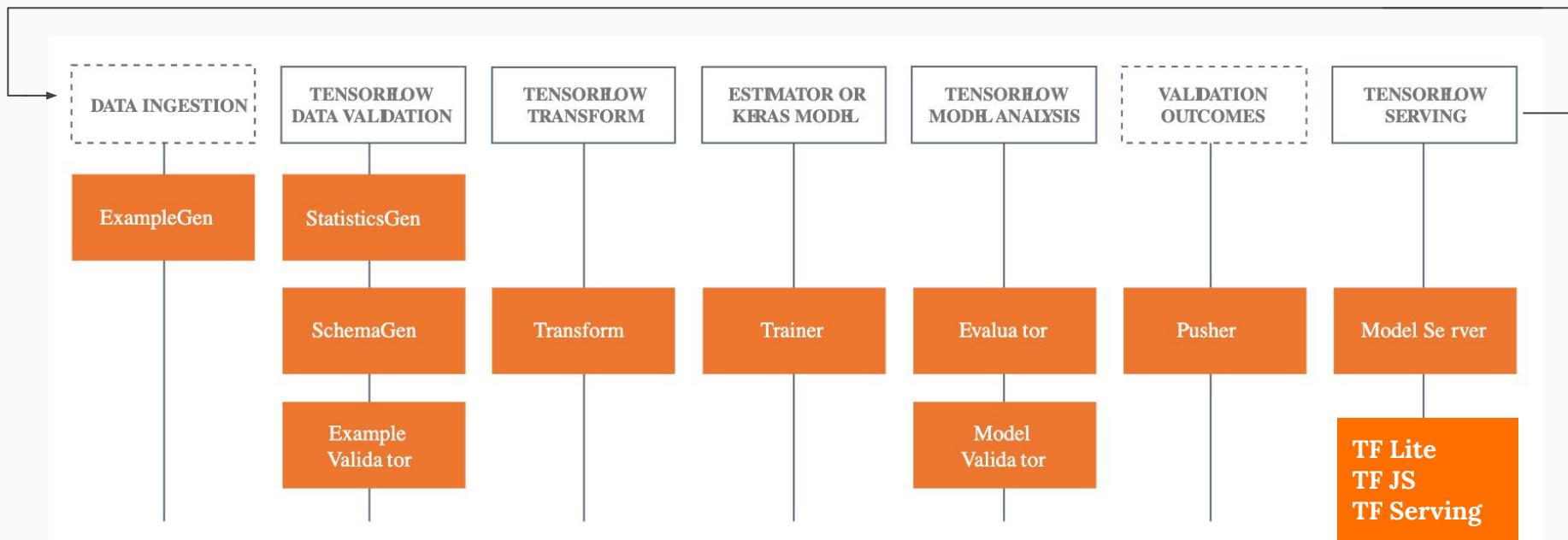
More info <https://www.tensorflow.org/tfx>



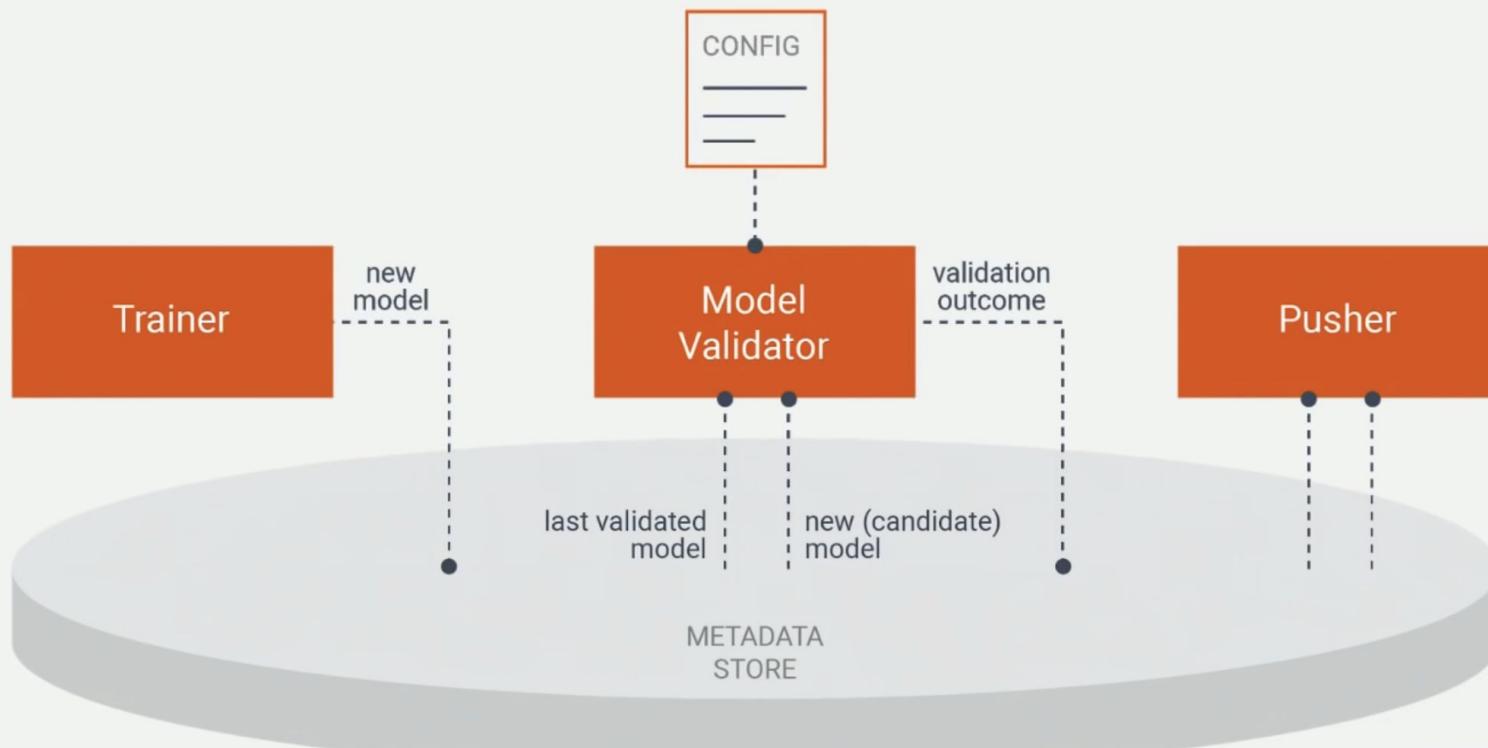
TFX ML workflow

TFX allows you to create production ML pipeline that includes many of the requirements for production software deployments and best practices.

Logging and collecting new
data for retrain

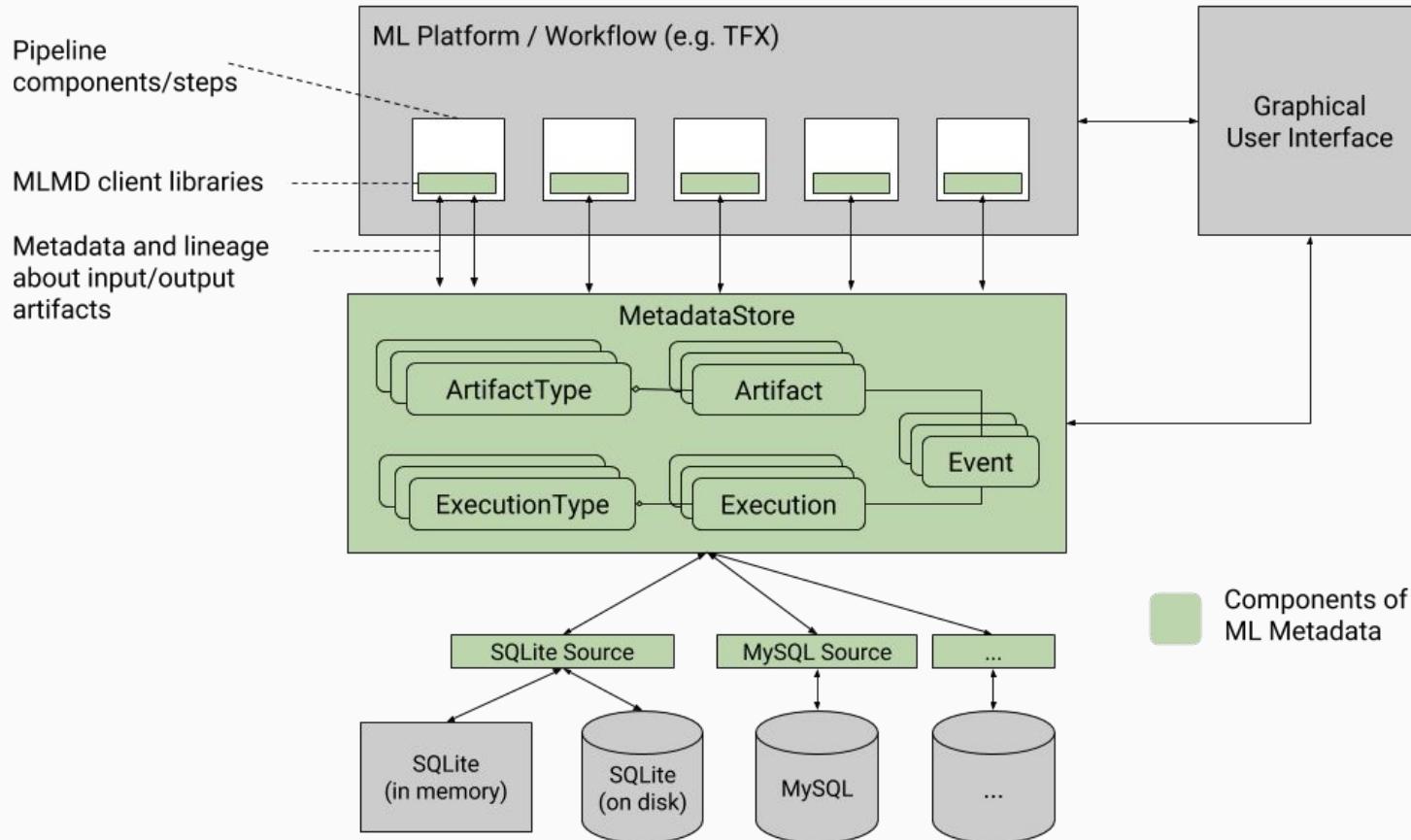


TFX MetadataStore



Modularity

TFX MetadataStore



TFX Compatible Versions

Compatible versions

The following table describes how the `tfx` package versions are compatible with its major dependency PyPI packages. This is determined by our testing framework, but other *untested* combinations may also work.

tfx	tensorflow	tensorflow-data-validation	tensorflow-model-analysis	tensorflow-metadata	tensorflow-transform	ml-metadata	apache-beam[gcp]	pyarrow
GitHub master	nightly (1.x)	0.13.1	0.13.2	0.13.0	0.13.0	0.13.2	2.12.0	0.11.1
0.13.0	1.13.1	0.13.1	0.13.2	0.13.0	0.13.0	0.13.2	2.12.0	n/a
0.12.0	1.12	0.12.0	0.12.1	0.12.1	0.12.0	0.13.2	2.10.0	n/a

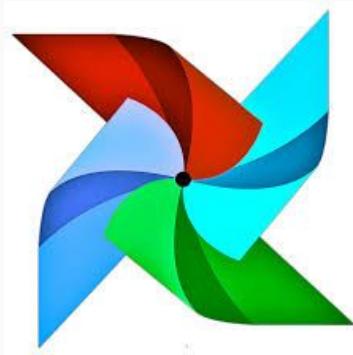
Source: <https://github.com/tensorflow/tfx>

Production ML properties

- Scalability
- Extensibility
- Modularity
- Testability
- Reproducibility
- Configuration
- Monitoring
- Safety
- Best practices

Orchestration

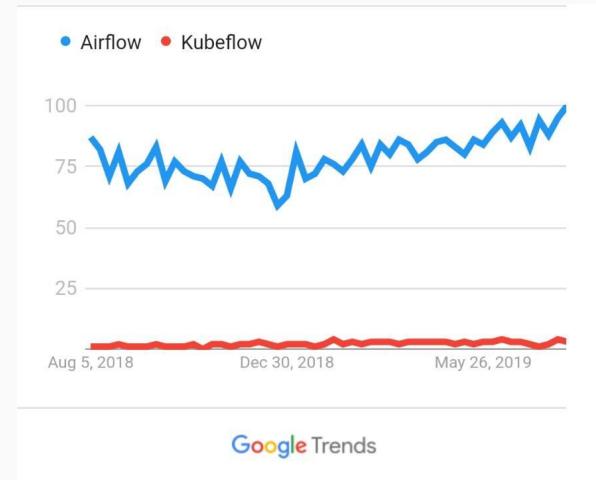
TFX uses an orchestration platform to manage its workflows
In this session we use airflow as an orchestration



Airflow



Kubeflow



What is airflow

The screenshot shows two parts of the Apache Airflow documentation and interface. On the left is a sidebar with links to Project, License, Quick Start, Installation, Tutorial, How-to Guides, UI / Screenshots, Concepts, Data Profiling, Command Line Interface, Scheduling & Triggers, Plugins, Security, Time zones, Experimental Rest API, Integration, Metrics, Kubernetes, Lineage, Changelog, FAQ, and Macros reference. The main area shows the Apache Airflow Documentation homepage with a logo, navigation links (Docs, View page source), and a section about Airflow being a platform to programmatically author, schedule, and monitor workflows. It also includes a snippet of the Airflow web interface showing an 'Ad Hoc Query' section with a table of task data.

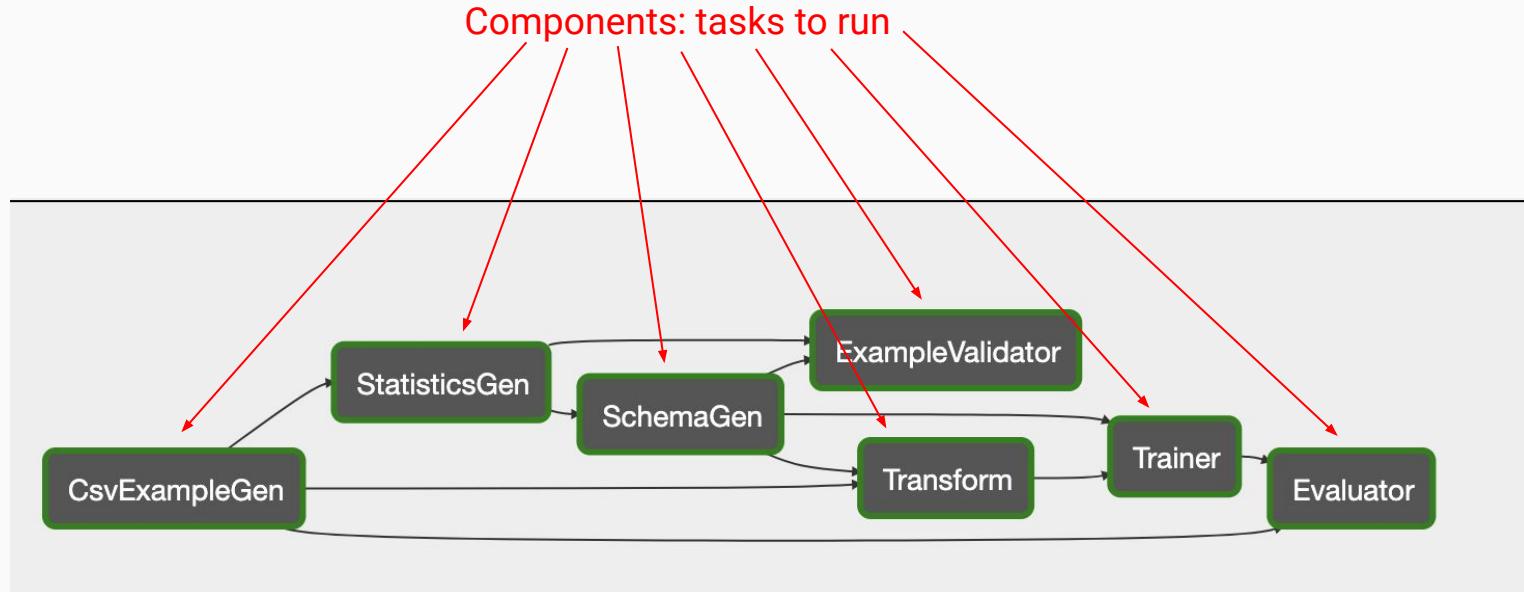
Airflow is a framework for scheduler management

Airflow has many convenience functions for doing interval jobs

Airflow has web service feature to visually manage a workflow

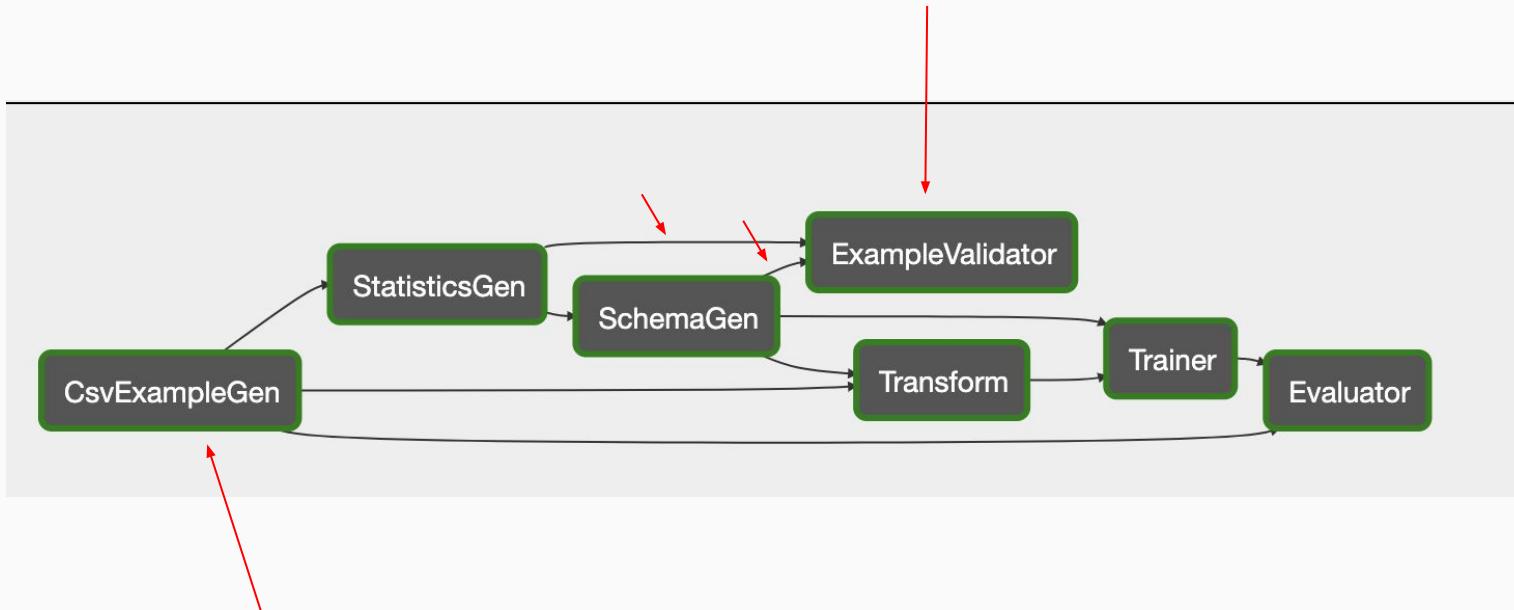
Airflow form tasks structure as directed acyclic graph

Directed acyclic graph in airflow



Directed acyclic graph in airflow

Ex: Before running this component StatisticGen and SchemaGen must be run completely



Airflow webserver features

The screenshot shows the Airflow webserver interface for managing Directed Acyclic Graphs (DAGs). The top navigation bar includes links for Airflow, DAGs, Data Profiling, Browse, Admin, Docs, and About, along with a timestamp of 2019-08-04 14:40:39 UTC.

The main content area displays a table of DAGs. The table has columns for DAG (with a detailed view icon), Schedule (with a dropdown menu showing 'None'), Owner (Airflow), Recent Tasks (with a count of 9), Last Run (2019-07-06 18:12), DAG Runs (with a count of 8 and one highlighted run circled in red with a value of 5), and Links (with various monitoring and control icons).

Annotations in red text and arrows highlight specific elements:

- An arrow labeled "Interval time" points to the "Schedule" column.
- An arrow labeled "Success run" points to the "Last Run" timestamp.
- An arrow labeled "Failure run" points to the circled value "5" in the "DAG Runs" column.
- An arrow labeled "Directed acyclic graph name (DAG name)" points to the "taxi" entry in the "DAG" column.
- An arrow labeled "DAGs" points to the "DAGs" link in the top navigation bar.

At the bottom left, there is a pagination control with buttons for «, <, 1, >, and ».

At the bottom right, a message states "Showing 1 to 2 of 2 entries".

DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
taxi	None	Airflow	9	2019-07-06 18:12	8	8
taxi_solution	None	Airflow	0		5	5

Airflow webserver features

Screenshot of the Airflow webserver showing the DAG: taxi page.

The top navigation bar includes: Airflow logo, DAGs, Data Profiling, Browse, Admin, Docs, About, and the timestamp 2019-08-04 15:00:32 UTC.

The main header shows "DAG: taxi" with a status of "Off" and "schedule: None".

Below the header are several navigation links: Graph View (highlighted with a red arrow), Tree View, Task Duration, Task Tries, Landing Times, Gantt, Details, Code, Trigger DAG (highlighted with a red arrow), Refresh, and Delete.

Filtering options include: Base date: 2019-07-06 18:12:47, Number of runs: 25, Run: manual_2019-07-06T18:12:46.493040+00:00, Layout: Left->Right, and Go.

Search bar: Search for... and a component search bar: Component.

Status filters: success, running, failed, skipped, up_for_reschedule, up_for_retry, queued, no_status.

The main content area displays the DAG graph:

```
graph LR; CsvExampleGen --> StatisticsGen; CsvExampleGen --> SchemaGen; StatisticsGen --> ExampleValidator; SchemaGen --> Transform; ExampleValidator --> Trainer; Transform --> Trainer; Trainer --> Evaluator;
```

Annotations:

- A red arrow points from the "Graph View" link to the DAG graph area.
- A red arrow points from the "Trigger DAG" link to the DAG graph area.
- A red arrow points from the "Component" search bar to the DAG graph area.
- A red box highlights the "failed" status filter, with the text "Manually trigger DAG" pointing to it.

Airflow webserver features

The screenshot shows the Airflow webserver interface for the DAG: taxi. The top navigation bar includes links for Airflow, DAGs, Data Profiling, Browse, Admin, Docs, About, and the current timestamp (2019-08-04 15:04:15 UTC). The main content area displays the DAG tree, run history, and code, with various buttons and filters.

DAG: taxi

Buttons and Filters:

- Graph View (disabled)
- Tree View** (selected)
- Task Duration
- Task Tries
- Landing Times
- Gantt
- Details
- Code
- Trigger DAG
- Refresh
- Delete

Base date: 2019-07-06 18:12:46 Number of runs: 25 Go

schedule: None

Annotations:

- View DAG as tree**: Points to the Tree View button.
- View DAG code**: Points to the Code button.
- Refresh view**: Points to the Refresh button.

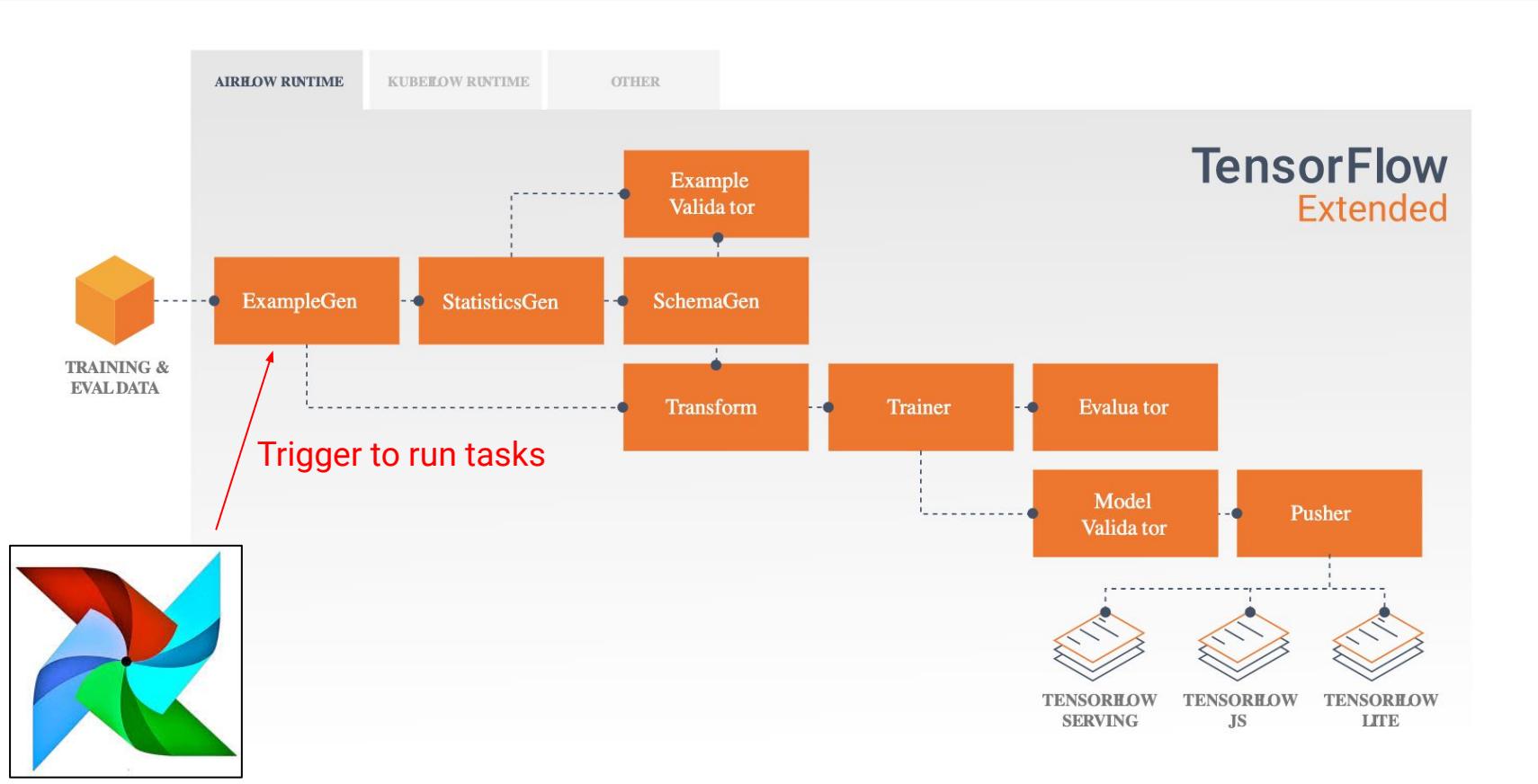
DAG tree: Shows a hierarchical tree of tasks. Nodes are colored by status: blue for running, red for failed, black for succeeded. Edges represent dependencies between tasks.

```
graph TD; DAG[DAG] --> ExampleValidator[ExampleValidator]; ExampleValidator --> StatisticsGen1[StatisticsGen]; ExampleValidator --> CsvExampleGen1[CsvExampleGen]; ExampleValidator --> SchemaGen1[SchemaGen]; ExampleValidator --> Evaluator[Evaluator]; Evaluator --> CsvExampleGen2[CsvExampleGen]; Evaluator --> Trainer[Trainer]; Evaluator --> SchemaGen2[SchemaGen]; Evaluator --> Transform[Transform]; Transform --> CsvExampleGen3[CsvExampleGen]; Transform --> SchemaGen3[SchemaGen]
```

Run history: A Gantt chart showing task execution status over time. The x-axis spans from Fri 05 to Jul 07. The y-axis lists individual tasks. Colored squares indicate task status: green for success, red for failure, yellow for up_for_retry, and grey for queued or no_status.

Date	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14	Task 15	Task 16	Task 17	Task 18	Task 19	Task 20	Task 21	Task 22	Task 23	Task 24	Task 25
Fri 05	success																								
Sat 06	running																								
Sun 07	success																								

TFX Pipeline



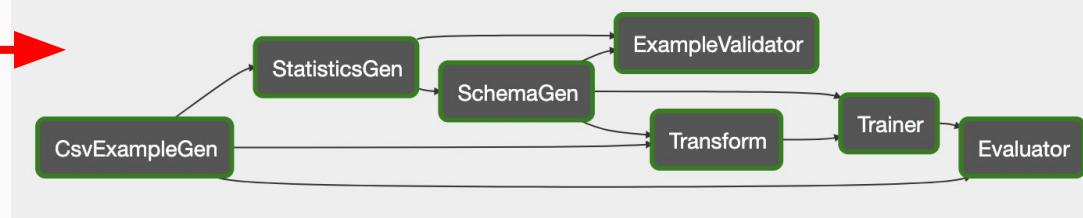
3. TFX

Components
Settings

TFX airflow/dags/taxi_pipeline.py (dag file) “Settings”

Airflow read files in path “airflow/dags” and construct DAG from code

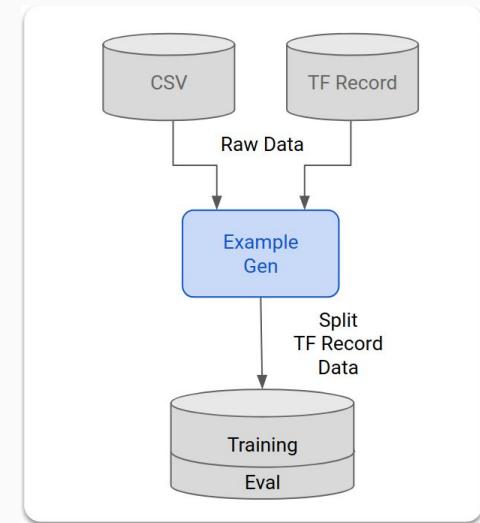
```
74 # Logging overrides
75 logger_overrides = {
76     'log_root': _log_root,
77     'log_level': logging.INFO
78 }
79
80
81 def _create_pipeline():
82     """Implements the chicago taxi pipeline with TFX."""
83     examples = csv_input(_data_root)
84
85     # Brings data into the pipeline or otherwise joins/converts training data.
86     example_gen = CsvExampleGen(input_base=examples)
87
88
89     # pylint: disable=line-too-long
90     statistics_gen = StatisticsGen(input_data=example_gen.outputs.examples) # Step 3
91     # pylint: enable=line-too-long
92
93     # Generates schema based on statistics files.
94     infer_schema = SchemaGen(stats=statistics_gen.outputs.output) # Step 3
95
96     # Performs anomaly detection based on statistics and data schema.
97     validate_stats = ExampleValidator() # Step 3
98     stats=statistics_gen.outputs.output, # Step 3
99     schema=infer_schema.outputs.output # Step 3
100
101     # Performs transformations and feature engineering in training and serving.
102     transform = Transform() # Step 4
103     input=example_gen.outputs.examples, # Step 4
104     schema=infer_schema.outputs.output, # Step 4
105     module_file=_taxi_module_file() # Step 4
106
107     # Uses user-provided Python function that implements a model using TF-Learn.
108     trainer = Trainer() # Step 5
109     module_file=_taxi_module_file(), # Step 5
110     transformed_examples=transform.outputs.transformed_examples, # Step 5
111     schema=infer_schema.outputs.output, # Step 5
112     transform_output=transform.outputs.transform_output, # Step 5
113     train_args=trainer_pb2.TrainArgs(num_steps=10000), # Step 5
114     eval_args=trainer_pb2.EvalArgs(num_steps=5000) # Step 5
115
116     # Uses TFM to compute a evaluation statistics over features of a model.
117     model_analyzer = Evaluator() # Step 6
118     examples=example_gen.outputs.examples, # Step 6
119     model_exports=trainer.outputs.output, # Step 6
120     feature_slicing_spec=evaluator_pb2.FeatureSlicingSpec(specs=[ # Step 6
121         evaluator_pb2.SingleSlicingSpec( # Step 6
122             column_for_slicing=['trip_start_hour']) # Step 6
123     ]) # Step 6
124
125     # Performs quality validation of a candidate model (compared to a baseline).
126     model_validator = ModelValidator() # Step 7
127     examples=example_gen.outputs.examples, # Step 7
```



ExampleGen



- ExampleGen generate train and evaluation data from datasource (CSV, TF records)
- First component in a flow



ExampleGen “Settings”

- The data directory path is passed as argument of csv_input
- Create CsvExampleGen component to retrieve data from that directory

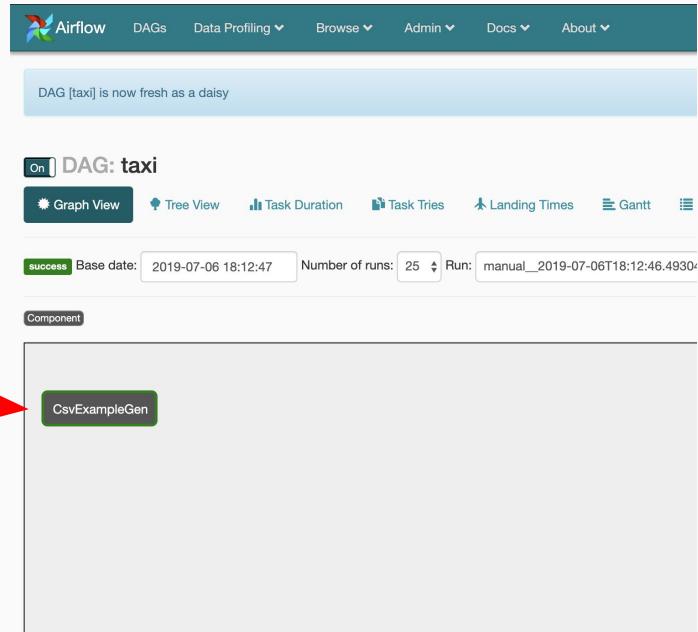
```
def _create_pipeline():
    """Implements the chicago taxi pipeline with TFX."""
    examples = csv_input(_data_root) # directory which collect the data files

    # Brings data into the pipeline or otherwise joins/converts training data.
    example_gen = CsvExampleGen(input_base=examples)

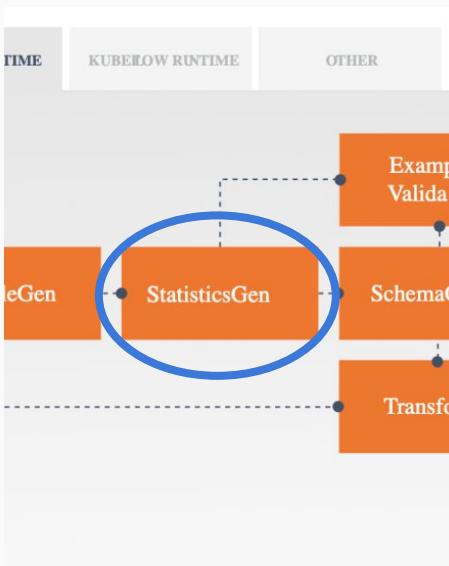
    return pipeline.Pipeline(
        pipeline_name='taxi',
        pipeline_root=_pipeline_root,
        components=[
            example_gen,
        ],
        enable_cache=True,
        metadata_db_root=_metadata_db_root,
        additional_pipeline_args={'logger_args': logger_overrides},
    )

airflow_pipeline = AirflowDAGRunner(
    _airflow_config).run(_create_pipeline())
```

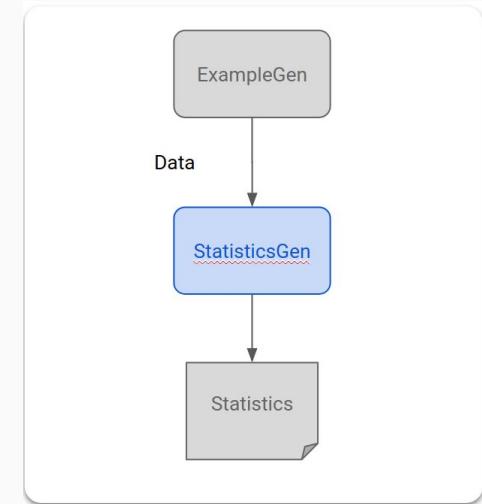
Create component



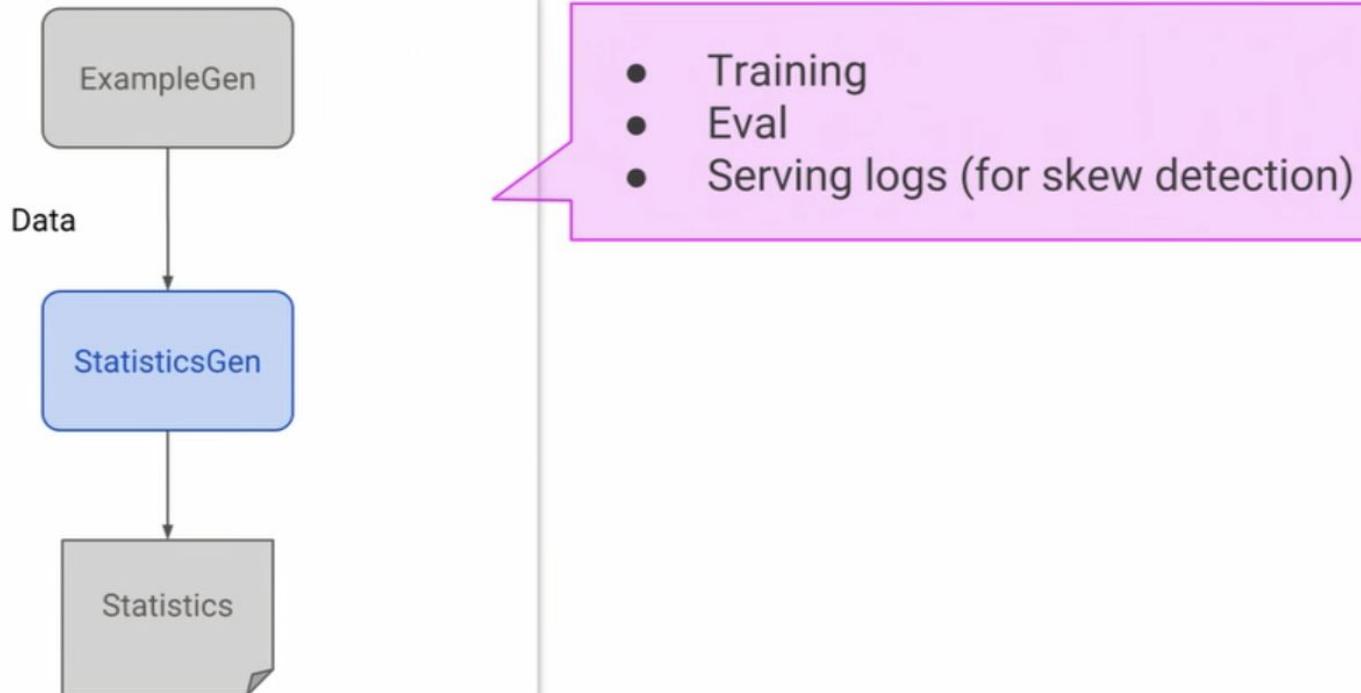
Statistics Gen



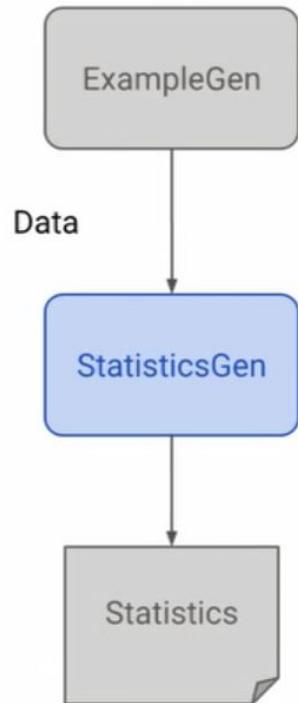
- Create statistics of specific data from example gen
- StatisticsGen will apply basic statistics tools in data
- Statistics can be used to create a schema and validate data



Statistics Gen “Inputs and Outputs”



Statistics Gen “Inputs and Outputs”



- Captures shape of data
- Visualization highlights unusual stats
- Overlay helps with comparison

Statistics Gen “Settings”

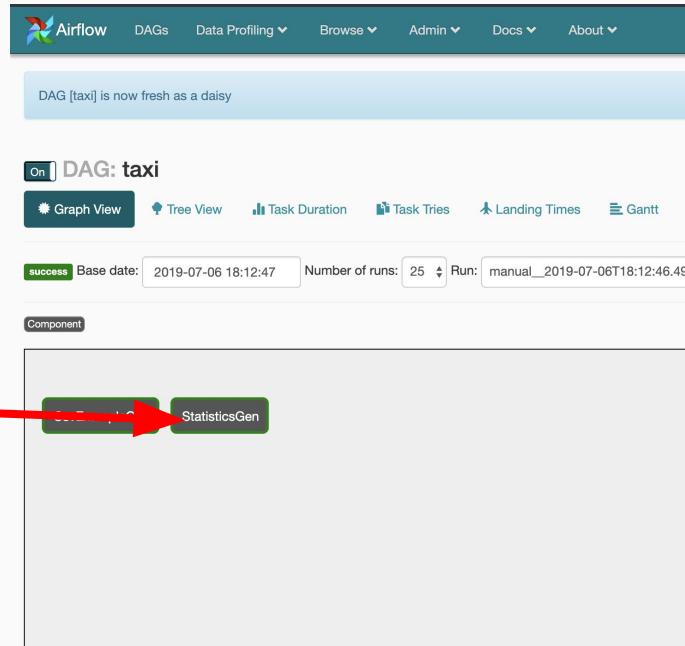
- Loaded examples is passed to StatisticsGen to extract some basic statistic

```
def _create_pipeline():
    """Implements the chicago taxi pipeline with TFX."""
    examples = csv_input(_data_root) # directory which collect the data files

    # Brings data into the pipeline or otherwise joins/converts training data.
    example_gen = CsvExampleGen(input_base=examples)

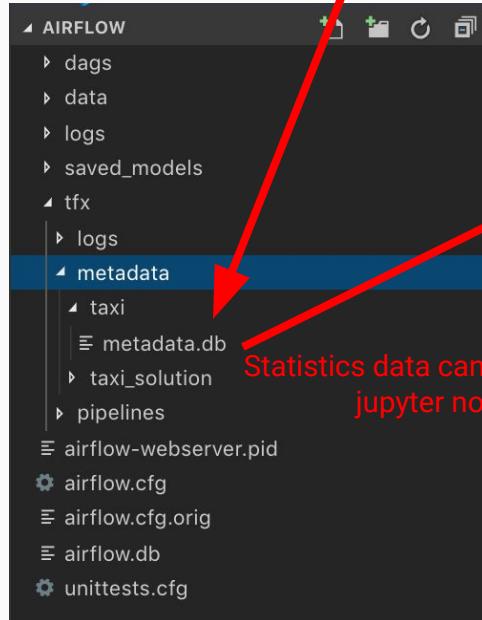
    statistics_gen = StatisticsGen(input_data=example_gen.outputs.examples) StatisticsGen component

    return pipeline.Pipeline(
        pipeline_name='taxi',
        pipeline_root=_pipeline_root,
        components=[
            example_gen,
            statistics_gen,
        ],
        enable_cache=True,
        metadata_db_root=_metadata_db_root,
        additional_pipeline_args={'logger_args': logger_overrides},
    )
```



Statistics Gen “Metadata”

Statistic data is generated in database after
Trigger a StatisticsGen component



In [2]:

```
from __future__ import print_function
import os
import tfx_utils
import matplotlib.notebook
def _make_default_sqlite_uri(pipeline_name):
    return os.path.join(os.environ['HOME'], 'airflow/tfx/metadata', pipeline_name, 'metadata.db')
def get_metadata_store(pipeline_name):
    return tfx_utils.TFXReadonlyMetadataStore.from_sqlite_db(_make_default_sqlite_uri(pipeline_name))
pipeline_name = 'taxi'
pipeline_db_path = _make_default_sqlite_uri(pipeline_name)
store = get_metadata_store(pipeline_name)
store.display_stats_for_examples(2)
```

Sort by Feature order Reverse order Feature search (regex enabled)

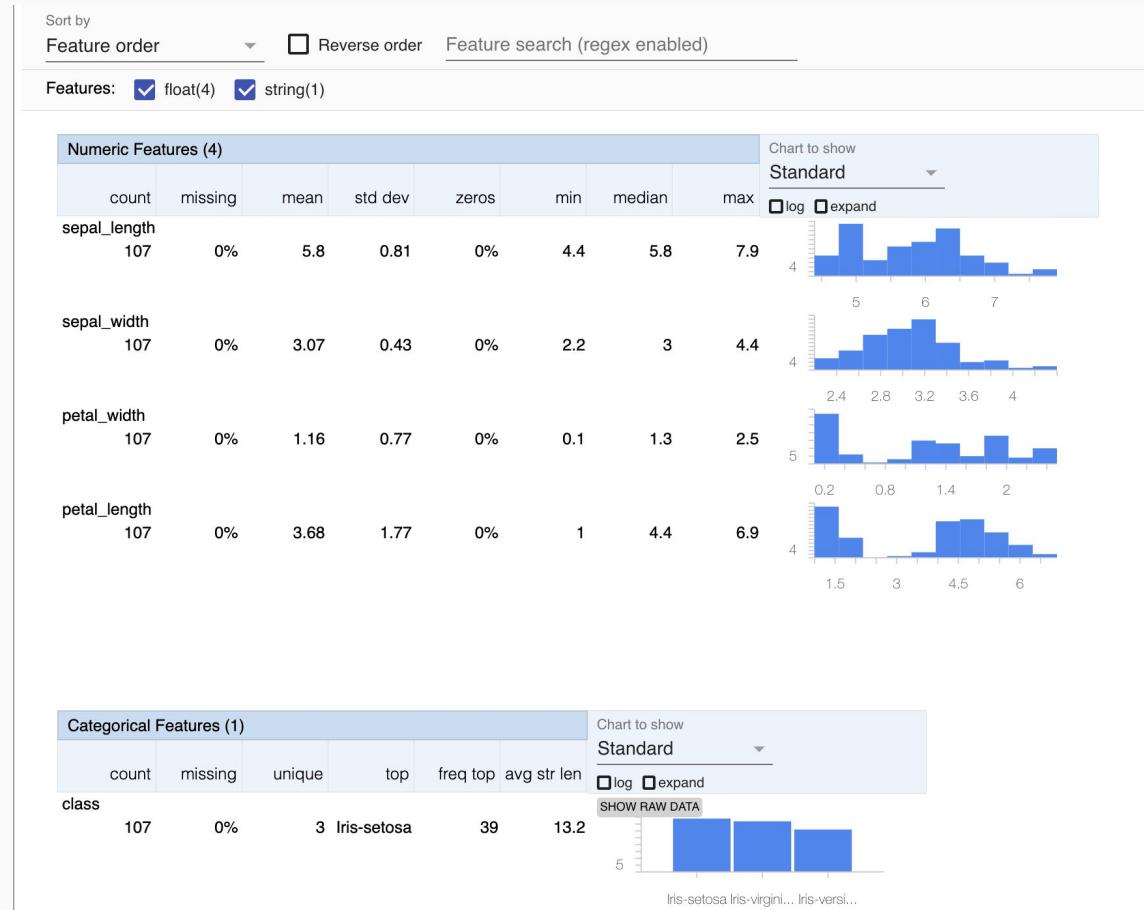
Features: int(5) float(10) string(2) unknown(1)

	count	missing	mean	std dev	zeros	min	median	max
dropoff_latitude	9,734	3.01%	41.9	0.04	0%	41.66	41.89	42.01
trip_seconds	10.0k	0.03%	767.94	816.68	3.2%	0	540	35.0k
trip_start_month	10.0k	0%	6.6	3.37	0%	1	7	12
trip_start_day	10.0k	0%	4.2	2.01	0%	1	4	7
trip_miles	10.0k	0%	2.74	6.01	26.98%	0	1	178
pickup_community_area	10.0k	0%	22.08	19.26	0%	1	8	77
dropoff_census_tract	7,198	28.28%	17.0B	0	0%	17.0B	17.0B	17.0B

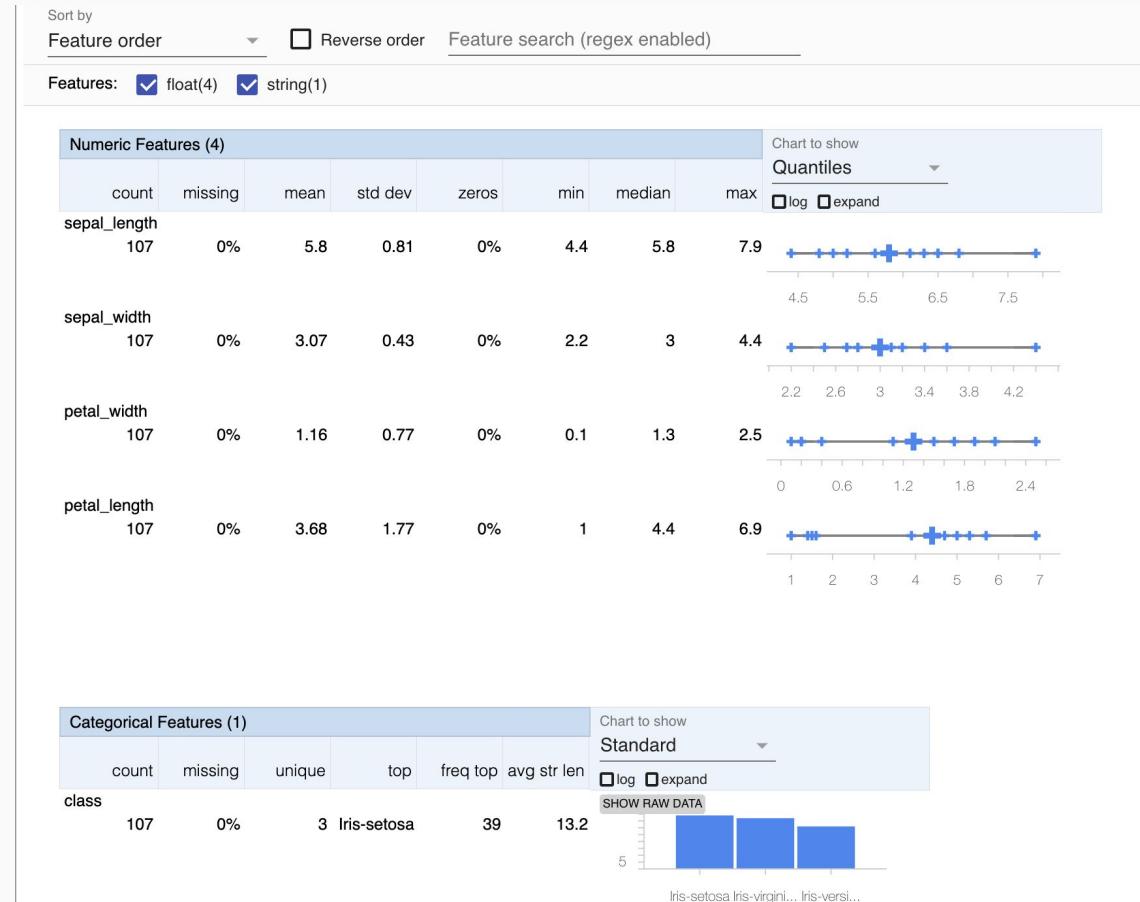
Chart to show Standard Log expand

Statistics data can be displayed in jupyter notebook

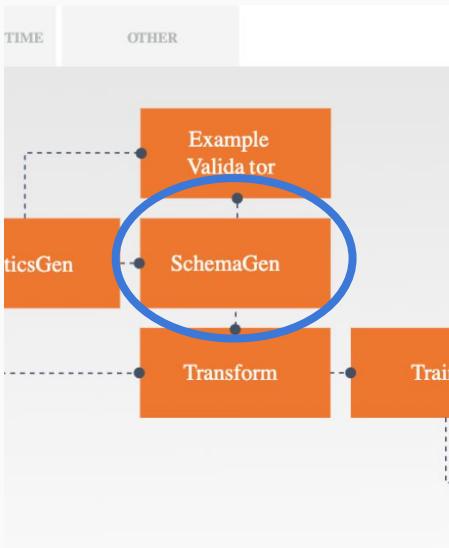
Statistics Gen “Visualization”



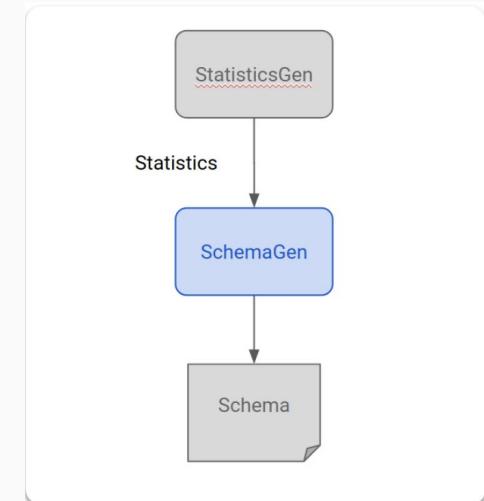
Statistics Gen “Visualization”



Schema Gen

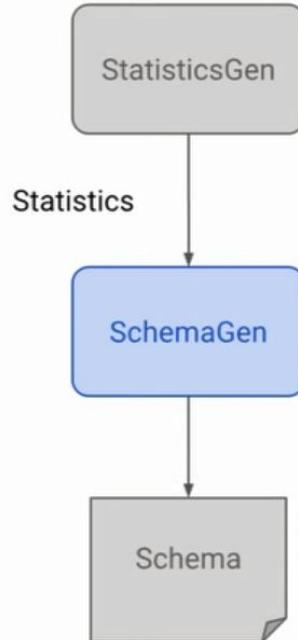


- Generate schema for dataset in each feature
- SchemaGen consume statistics from StatisticsGen



Schema Gen “Inputs and Outputs”

Inputs and Outputs



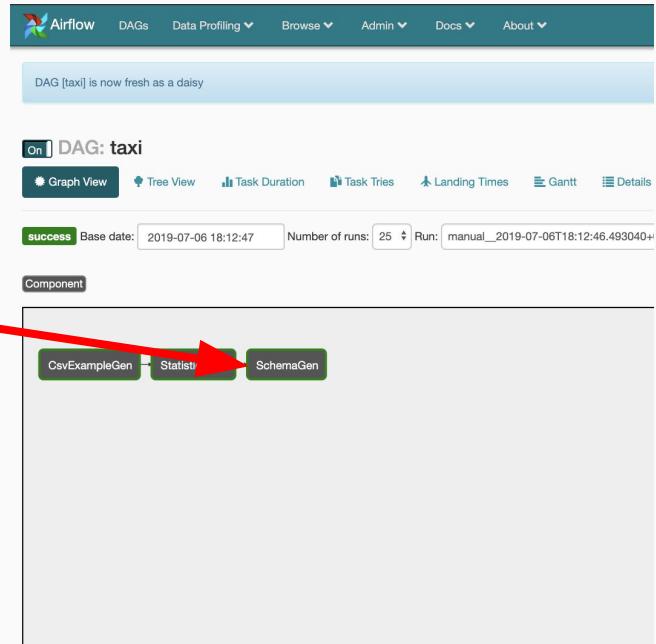
- High-level description of the data
 - Expected features
 - Expected value domains
 - Expected constraints
 - and much more!
- Codifies expectations of “good” data
- Initially inferred, then user-curated

Schema Gen “Settings”

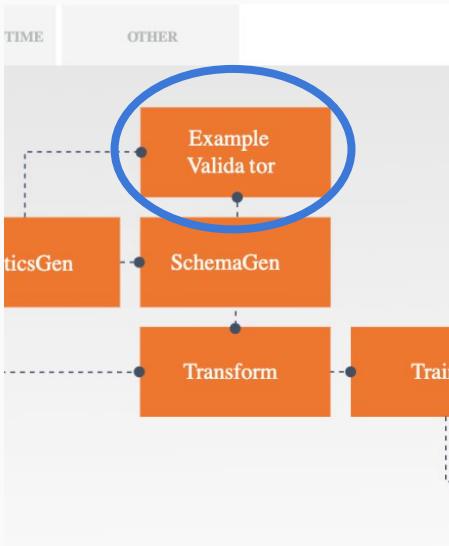
- Generate schema from data statistics
- User can manually modify the schema later

```
# Generates schema based on statistics files.
infer_schema = SchemaGen(stats=statistics_gen.outputs.output)

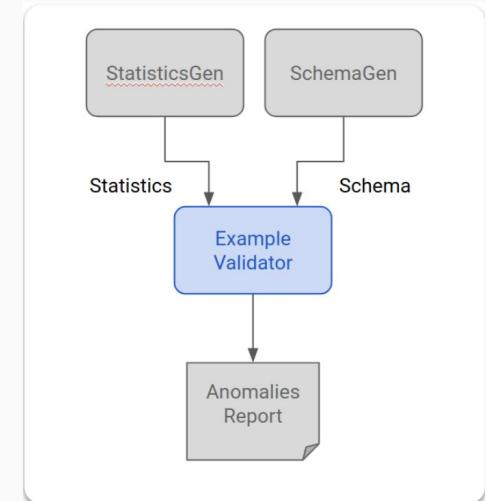
return pipeline.Pipeline(
    pipeline_name='taxi',
    pipeline_root=_pipeline_root,
    components=[
        example_gen,
        statistics_gen,
        infer_schema,
    ],
    enable_cache=True,
    metadata_db_root=_metadata_db_root,
    additional_pipeline_args={'logger_args': logger_overrides},
)
```



Example Validator

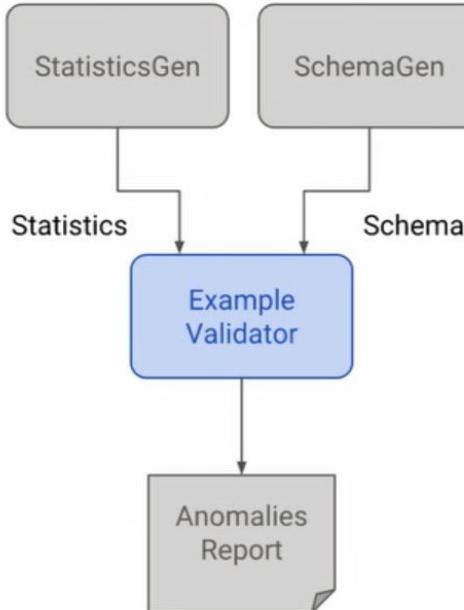


- Generate schema for dataset in each feature
- SchemaGen consume statistics from StatisticsGen



Example Validator “Inputs and Outputs”

Inputs and Outputs



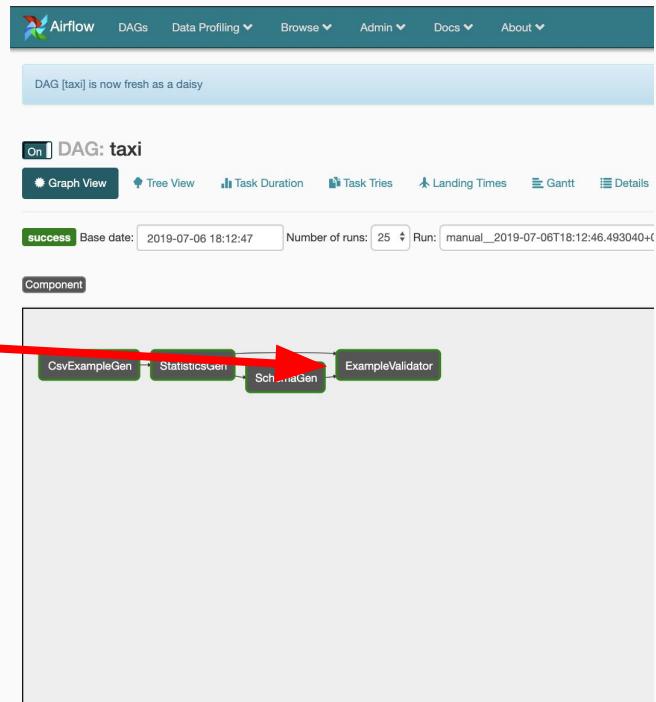
- Missing features
- Wrong feature valency
- Training/serving skew
- Data distribution drift
- ...

Example Validator “Settings”

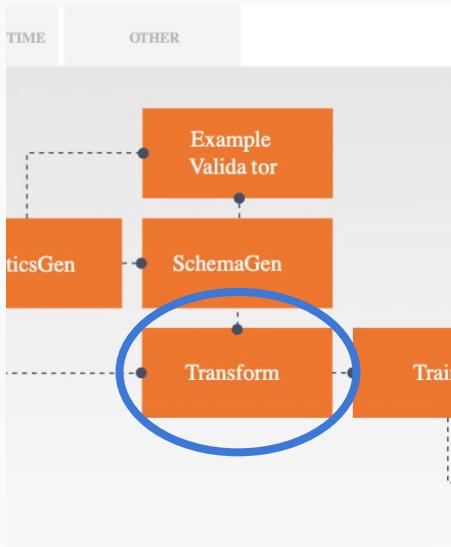
- Compare data statistics (at that time) and data schema to check anomalies

```
# Performs anomaly detection based on statistics and data schema.
validate_stats = ExampleValidator(
    stats=statistics_gen.outputs.output,
    schema=infer_schema.outputs.output)

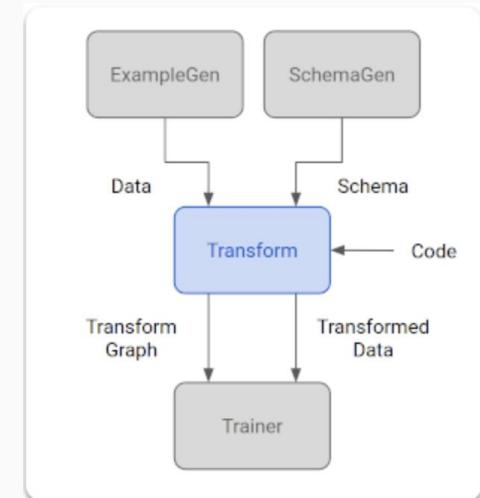
return pipeline.Pipeline(
    pipeline_name='taxi',
    pipeline_root=_pipeline_root,
    components=[
        example_gen,
        statistics_gen,
        infer_schema,
        validate_stats,
    ],
    enable_cache=True,
    metadata_db_root=_metadata_db_root,
    additional_pipeline_args={'logger_args': logger_overrides},
)
```



Transform

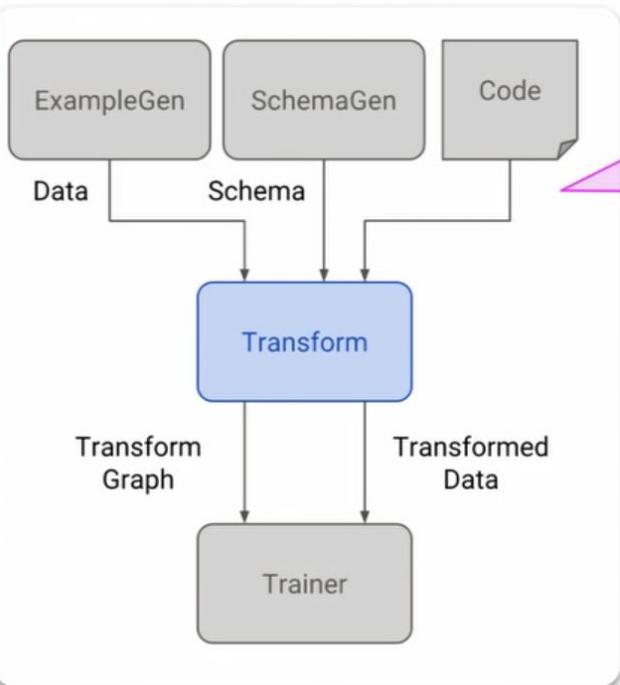


- Generate schema for dataset in each feature
- SchemaGen consume statistics from StatisticsGen



Transform “Inputs and Outputs”

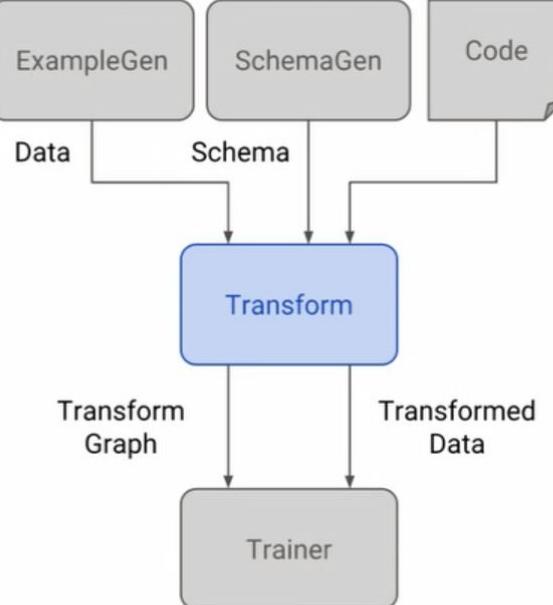
Inputs and Outputs



- User-provided transform code (TF Transform)
- Schema for parsing

Transform “Inputs and Outputs”

Inputs and Outputs



Transform Graph

- Applied at training time
- Embedded in serving graph

(Optional) Transformed Data

- For performance optimization

Transform “Concept”

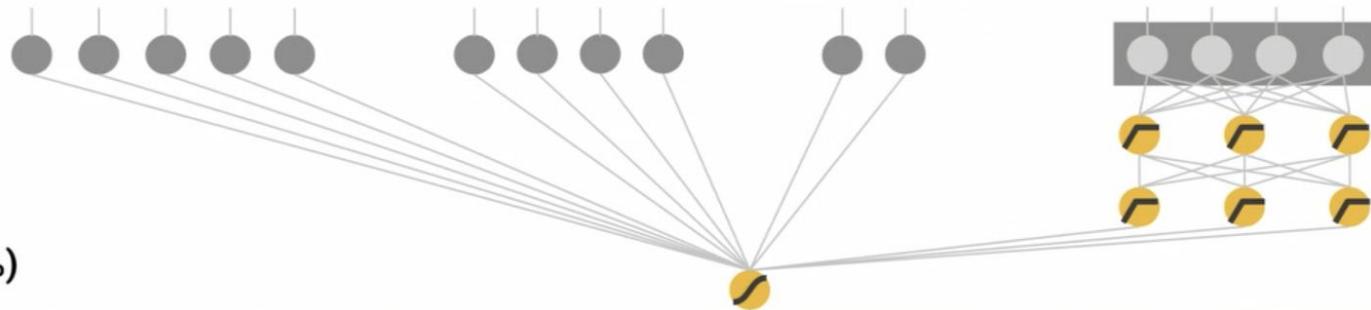
Features

Categorical Features	Bucket Features	Vocab Features	Dense Float Features
trip_start_hour	pickup_latitude	payment_type	trip_miles
trip_start_day	pickup_longitude	company	fare
trip_start_month	dropoff_latitude		trip_seconds
pickup/dropoff_census_tract	dropoff_longitude		
pickup/dropoff_community_area			

Transforms

Model (Wide+Deep)

Label = tips > (fare * 20%)



Transform “Settings”

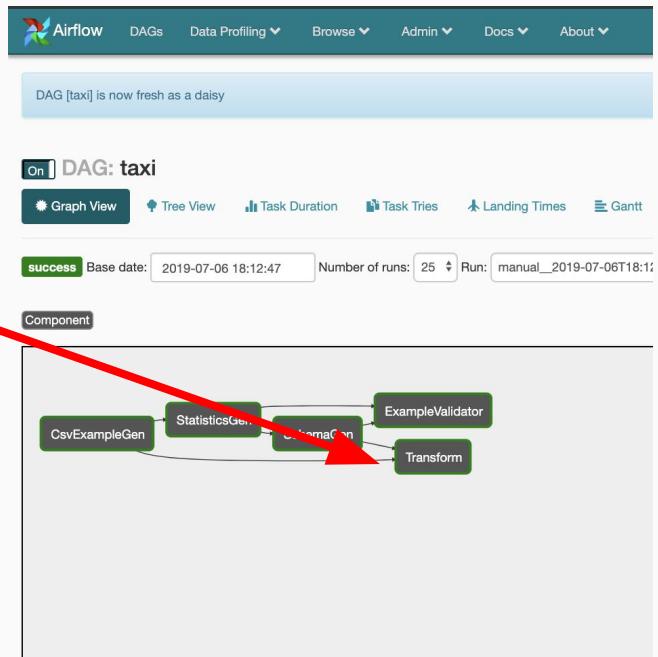
- Transform data (preprocess) to a valid form
- Predefined function from user

```
taxi_pipeline.py  taxi_utils.py  trainer_fn.py

105
106 def preprocessing_fn(inputs):
107     """tf.transform's callback function for preprocessing inputs.
108
109     Args:
110         inputs: map from feature keys to raw not-yet-transformed features.
111
112     Returns:
113         Map from string feature key to transformed feature operations.
114     """
115     outputs = {}
116     for key in _DENSE_FLOAT_FEATURE_KEYS:
117         # Preserve this feature as a dense float, setting nan's to the mean.
118         # tf_transform_fn automatically handles sparse (e.g., vocabulary)
119         # and dense features.
120         outputs[_transformed_name(key)] = tft.scale_to_z_score(
121             _fill_in_missing(inputs[key]),
122             _VOCAB_FEATURE_KEYS)
123
124     for key in _VOCAB_FEATURE_KEYS:
125         # Build a vocabulary for this feature.
126         outputs[_transformed_name(key)] = tft.compute_and_apply_vocabulary(
127             _fill_in_missing(inputs[key]),
128             top_k=_VOCAB_SIZE,
129             num_pov_buckets=_POV_SIZE)
130
131     for key in _BUCKET_FEATURE_KEYS:
132         outputs[_transformed_name(key)] = tft.bucketize(
133             _fill_in_missing(inputs[key]), _FEATURE_BUCKET_COUNT)
134
135     for key in _CATEGORICAL_FEATURE_KEYS:
136         outputs[_transformed_name(key)] = _fill_in_missing(inputs[key])
137
138     # Was this passenger a bit tipsy?
139     taxi_fare = _fill_in_missing(inputs[_FARE_KEY])
140     tips = _fill_in_missing(inputs[_LABEL_KEY])
141     outputs[_transformed_name(_LABEL_KEY)] = tf.where(
142         tf.is_nan(taxi_fare),
143         tf.zeros_like(taxi_fare), tf.int64(0),
144         # Test if the tip was > 20% of the fare.
145         tf.cast(
146             tf.greater(tips, tf.multiply(taxi_fare, tf.constant(0.2))), tf.int64(1))
147
148     return outputs
```

```
# Performs transformations and feature engineering in training and serving.
transform = Transform( # Step 4
    input_data=example_gen.outputs.examples, # Step 4
    schema=infer_schema.outputs.output, # Step 4
    module_file=_taxi_module_file) # Step 4

return pipeline.Pipeline(
    pipeline_name='taxi',
    pipeline_root=_pipeline_root,
    components=[
        example_gen,
        statistics_gen,
        infer_schema,
        validate_stats,
        transform,
    ],
    enable_cache=True,
    metadata_db_root=_metadata_db_root,
    additional_pipeline_args={'logger_args': logger_overrides},
)
```

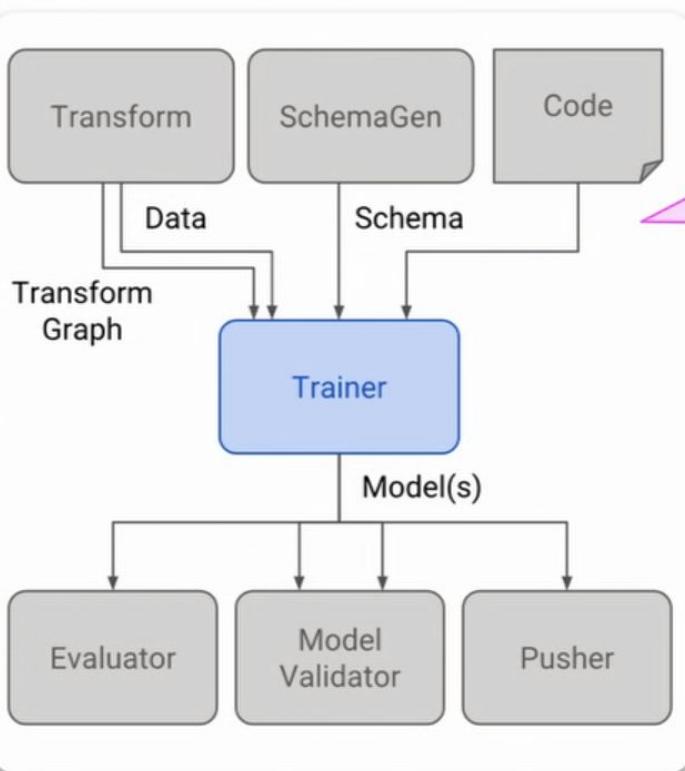


- Train the models from preprocessed data
- Using estimators to train the model
- Preallocated input spec



Trainer “Inputs and Outputs”

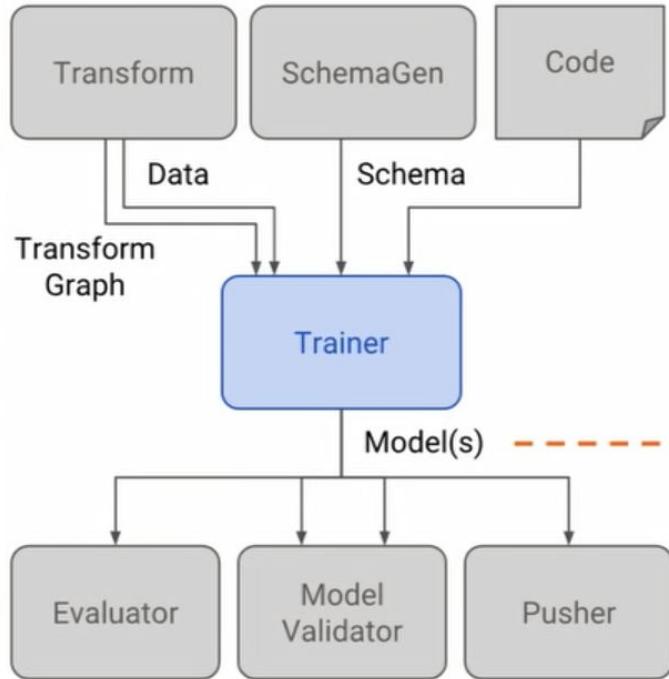
Inputs and Outputs



- User-provided training code (TensorFlow)
- Optionally, transformed data

Trainer “Inputs and Outputs”

Inputs and Outputs



Highlight: SavedModel Format

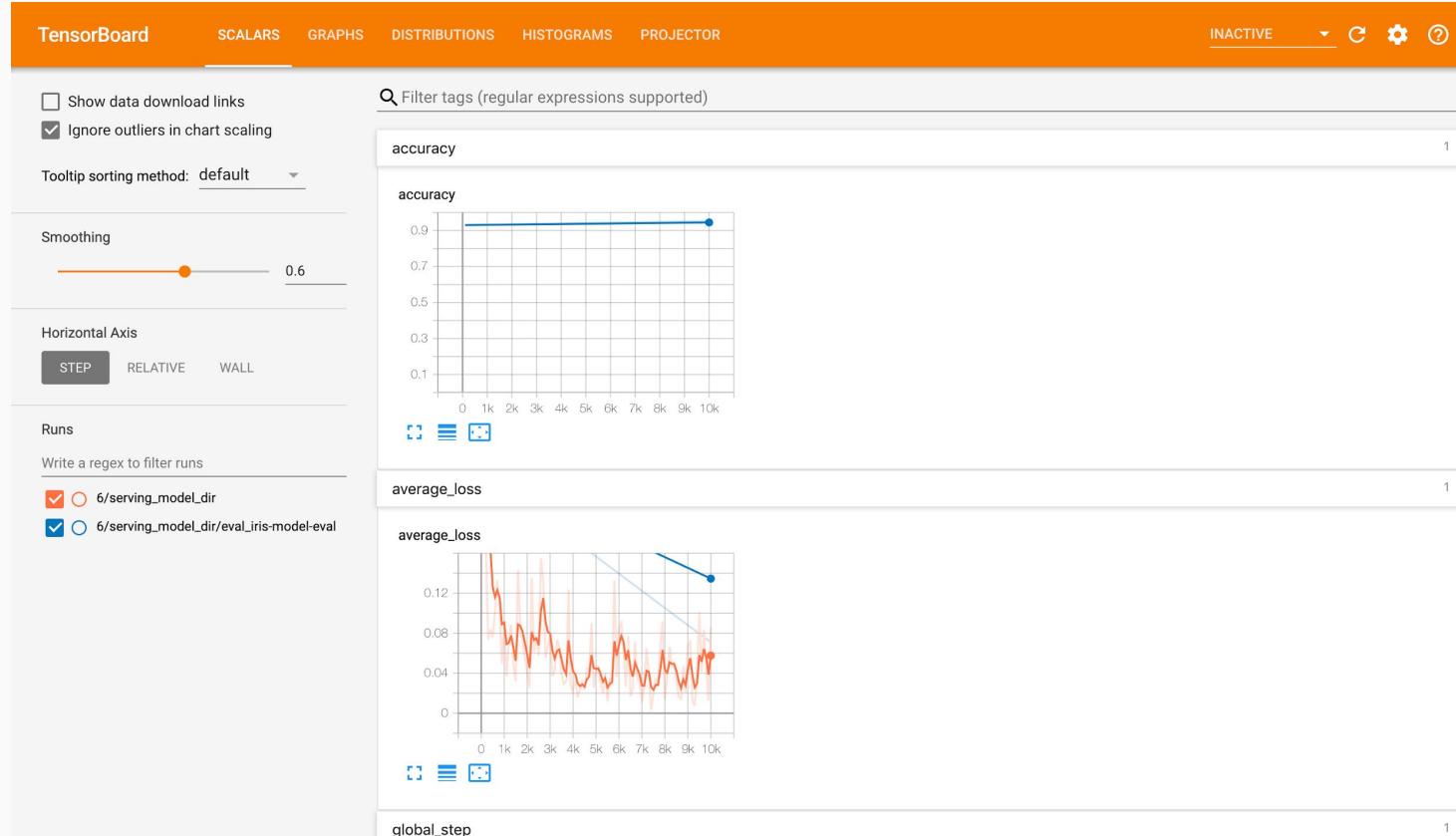
Train, Eval, and Inference Graphs



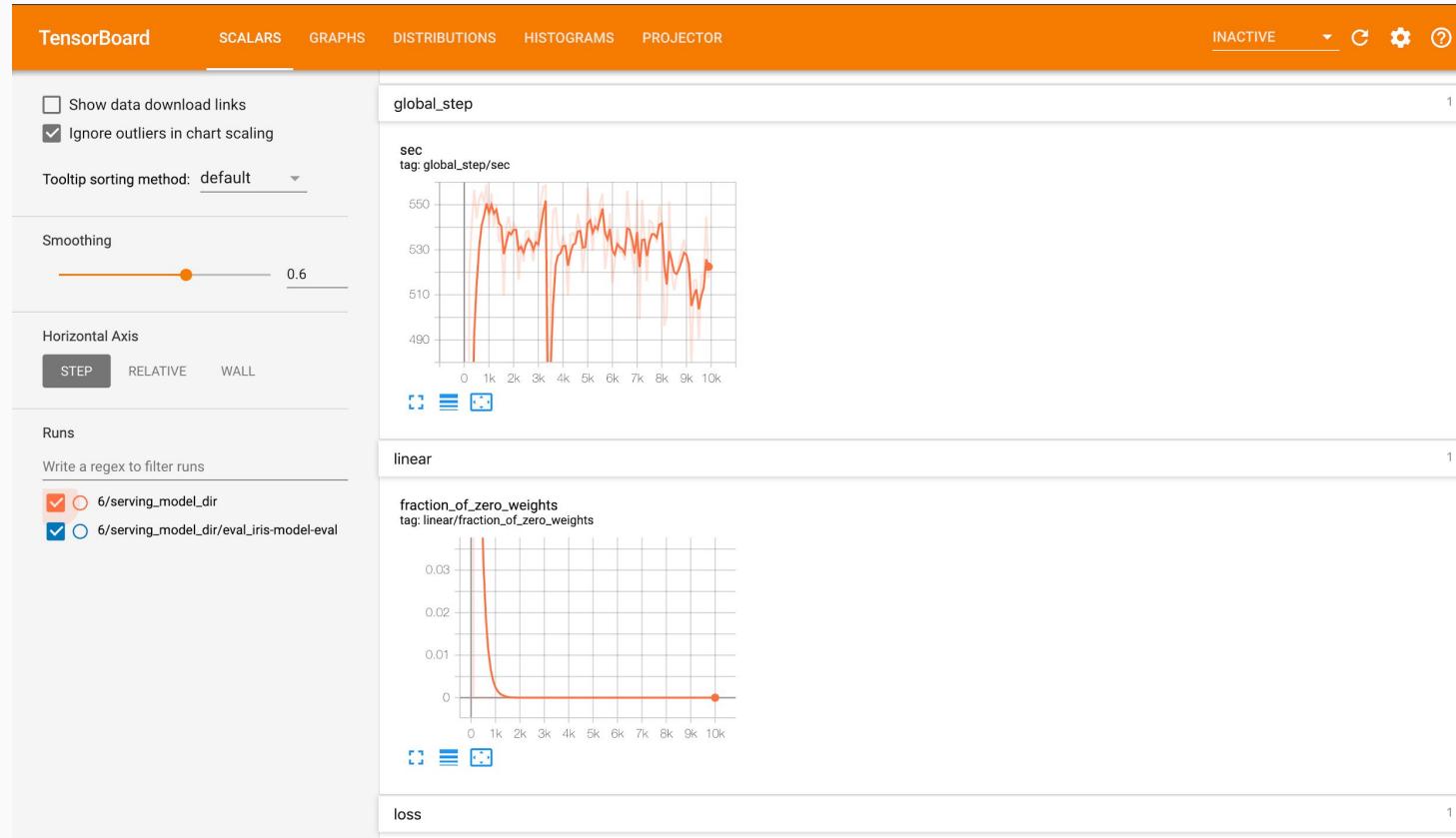
TensorFlow
Model Analysis

TensorFlow
Serving

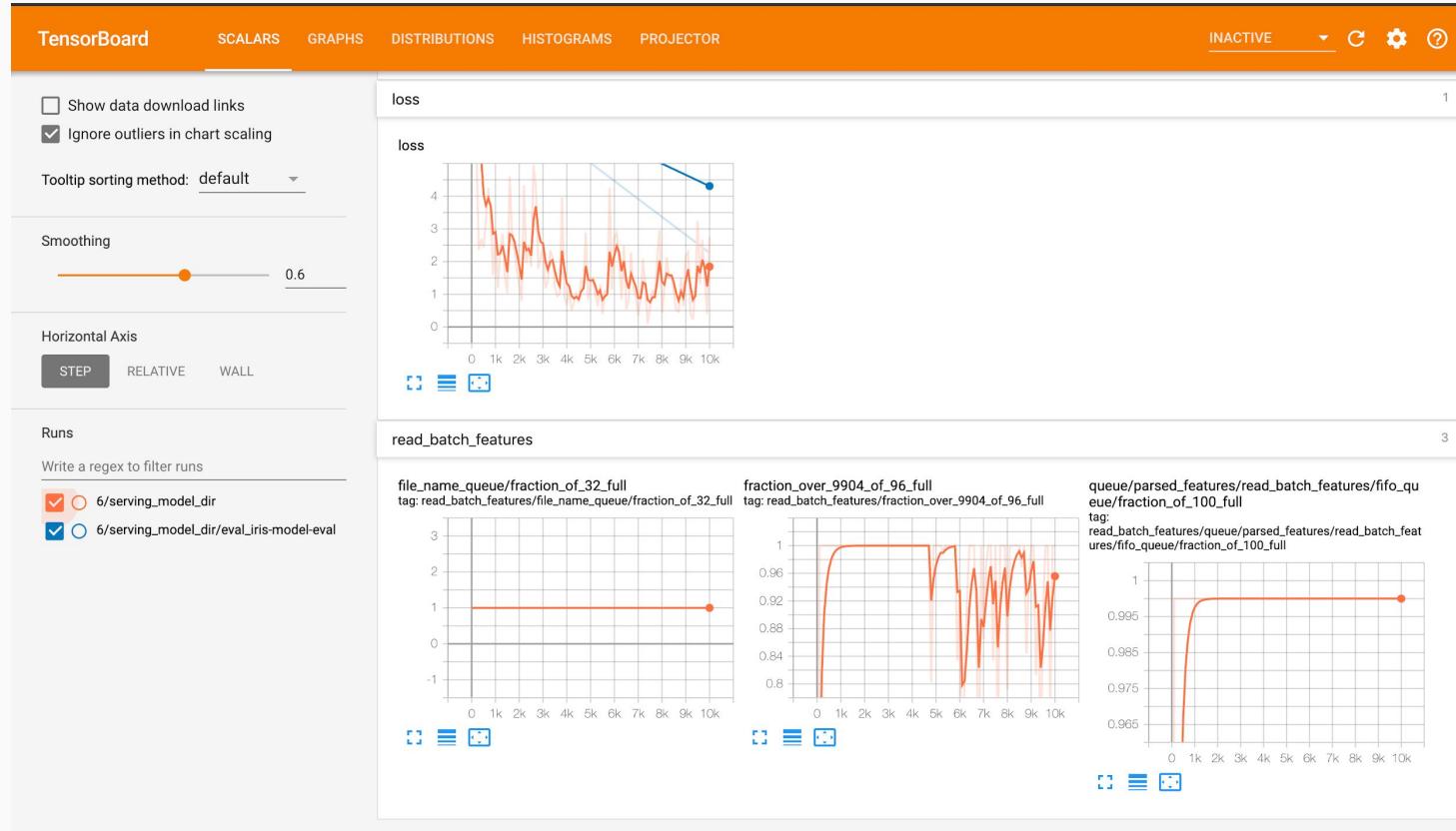
Trainer “Visualization”



Trainer “Visualization”



Trainer “Visualization”

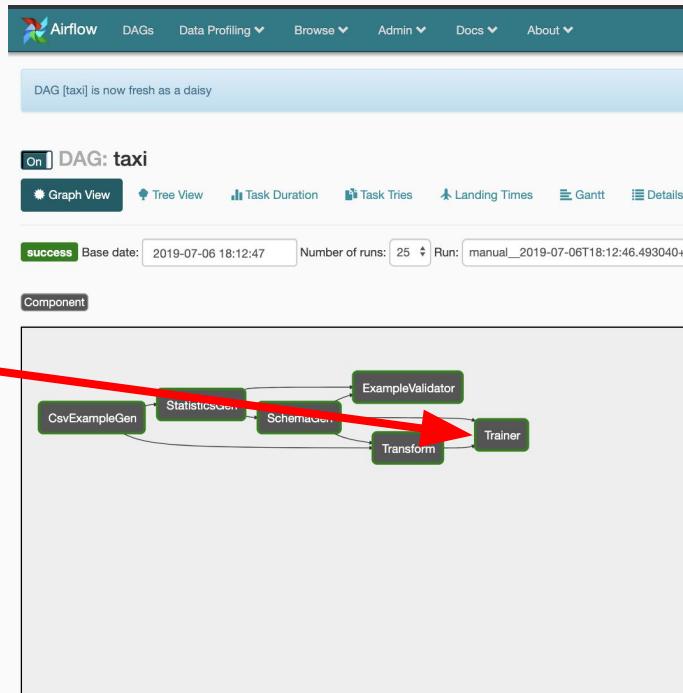


Trainer “Settings”

- Consume preprocessed data (Transform output) to fit new model
- Using predefined estimator and trainer_fn

```
taxi_pipeline.py  taxi_utils.py ●  
dags > taxi_utils.py > trainer_fn  
283  
284 def _build_estimator(config, hidden_units=None, warm_start_from=None):  
285     .  
286     .  
287     return tf.estimator.Estimator(...)  
288  
289 # TFX will call this function  
290 def trainer_fn(hparams, schema):  
291     """Build the estimator using the high level API.  
292  
293     Args:  
294         hparams: Holds hyperparameters used to train the model as name/value pairs.  
295         schema: Holds the schema of the training examples.  
296  
297     Returns:  
298         A dict of the following:  
299             - estimator: The estimator that will be used for training and eval.  
300             - train_spec: Spec for training.  
301             - eval_spec: Spec for eval.  
302             - eval_input_receiver_fn: Input function for eval.  
303     """  
304  
305     .  
306     .  
307     .  
308     return {  
309         'estimator': estimator,  
310         'train_spec': train_spec,  
311         'eval_spec': eval_spec,  
312         'eval_input_receiver_fn': receiver_fn  
313     }  
314
```

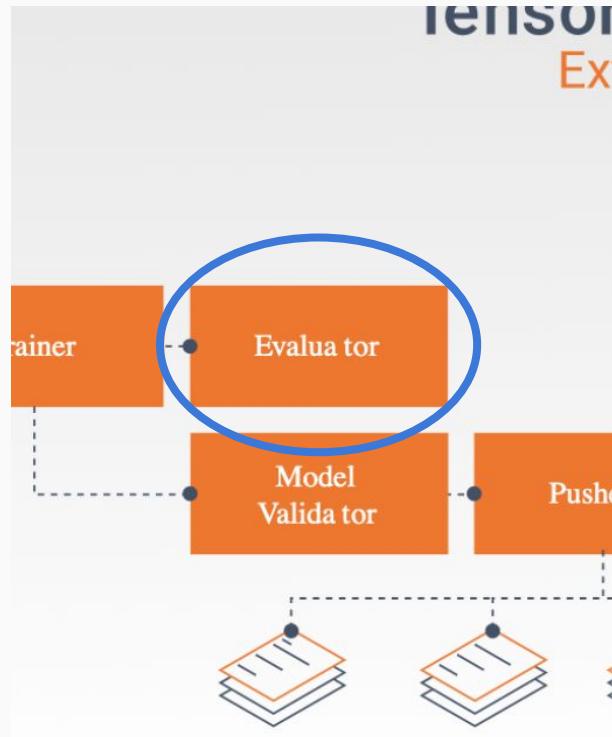
```
# Uses user-provided Python function that implements a model using TF-Learn.  
trainer = Trainer( # Step 5  
    module_file=_taxi_module_file, # Step 5  
    + untransformed_examples=transform.outputs.transformed_examples, # Step 5  
    schema=infer_schema.outputs.output, # Step 5  
    transform_output=transform.outputs.transform_output, # Step 5  
    train_args=trainer_pb2.TrainArgs(num_steps=10000), # Step 5  
    eval_args=trainer_pb2.EvalArgs(num_steps=5000)) # Step 5  
  
return pipeline.Pipeline(  
    pipeline_name='taxi',  
    pipeline_root=_pipeline_root,  
    components=[  
        example_gen,  
        statistics_gen,  
        infer_schema,  
        validate_stats,  
        transform,  
        trainer,  
    ],  
    enable_cache=True,  
    metadata_db_root=_metadata_db_root,  
    additional_pipeline_args={'logger_args': logger_overrides},  
)
```



Sadly, this time still not support for keras

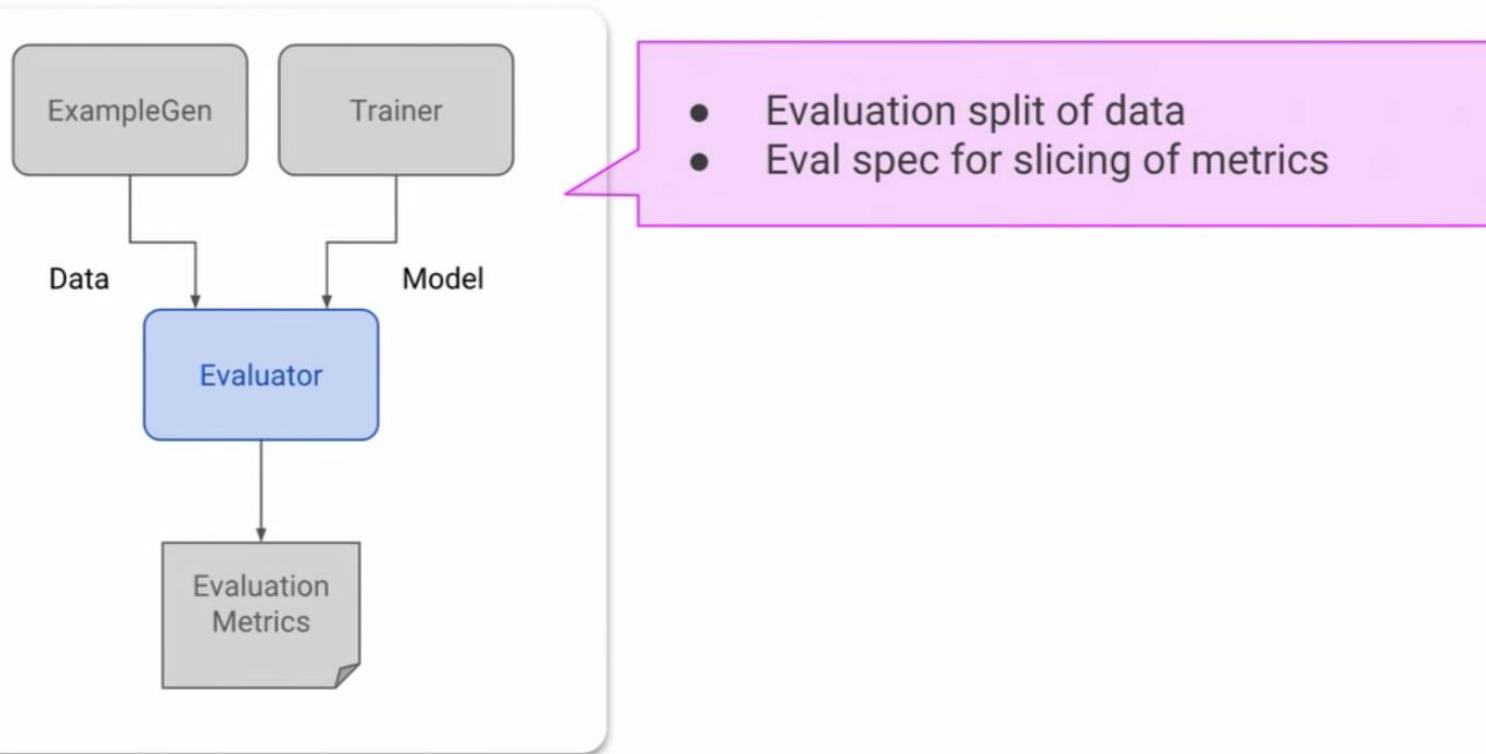
Evaluator

- Evaluate model performances
- Using to select which model to deploy
- Estimator in mode “EVAL”

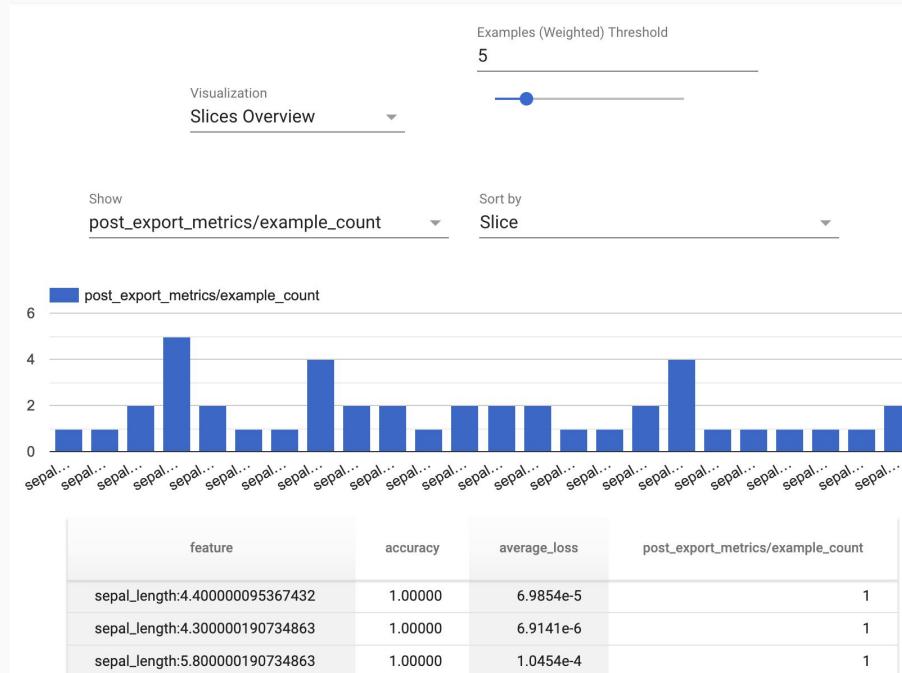


Evaluator “Inputs and Outputs”

Inputs and Outputs

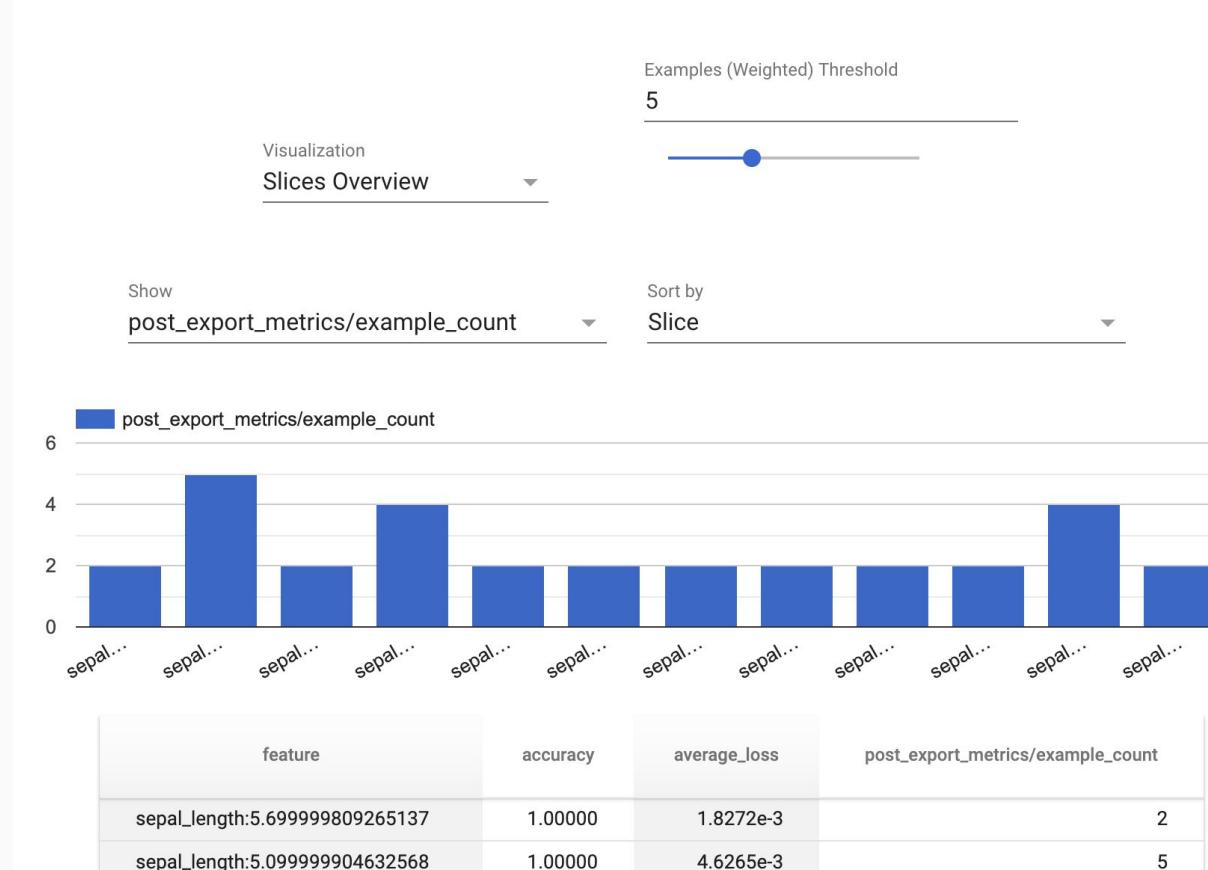


Evaluator “Visualization Example_count”

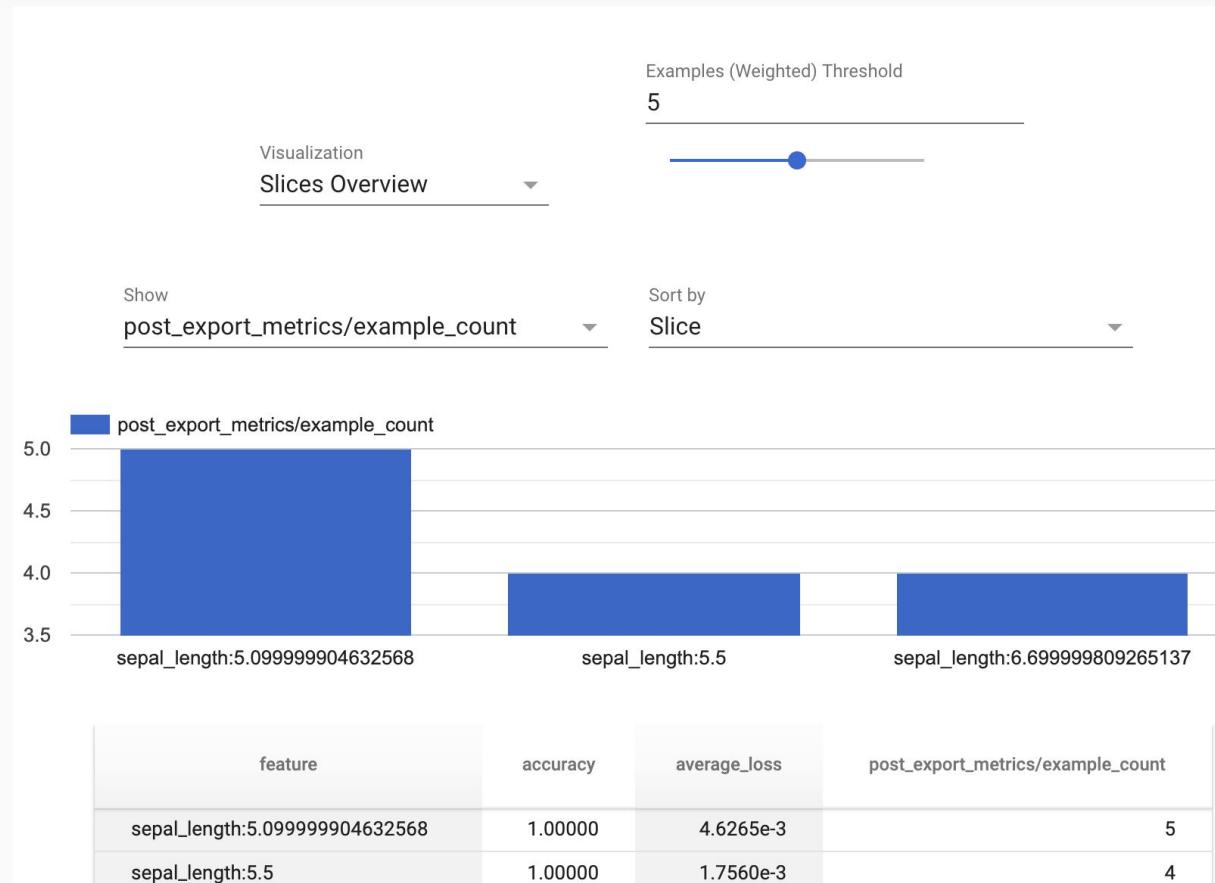


feature	accuracy	average_loss	post_export_metrics/example_count
sepal_length:4.40000095367432	1.00000	6.9854e-5	1
sepal_length:4.30000190734863	1.00000	6.9141e-6	1
sepal_length:5.800000190734863	1.00000	1.0454e-4	1
sepal_length:5.699999809265137	1.00000	1.8272e-3	2
sepal_length:5.09999904632568	1.00000	4.6265e-3	5
sepal_length:5.199999809265137	1.00000	6.0885e-3	2
sepal_length:5.5	1.00000	1.7560e-3	4
sepal_length:5.300000190734863	1.00000	6.0318e-5	1
sepal_length:7.0	1.00000	5.0867e-3	1
sepal_length:6.5	1.00000	0.02372	2
sepal_length:5.0	1.00000	3.5434e-3	2
sepal_length:5.90000095367432	0.50000	0.83960	2
sepal_length:6.0	1.00000	0.08510	2
sepal_length:6.09999904632568	1.00000	0.03060	2
sepal_length:5.59999904632568	1.00000	4.7584e-3	2
sepal_length:6.699999809265137	0.75000	0.22371	4
sepal_length:5.40000095367432	1.00000	0.12356	1
sepal_length:7.09999904632568	1.00000	2.2170e-4	1
sepal_length:7.59999904632568	1.00000	1.8239e-5	1

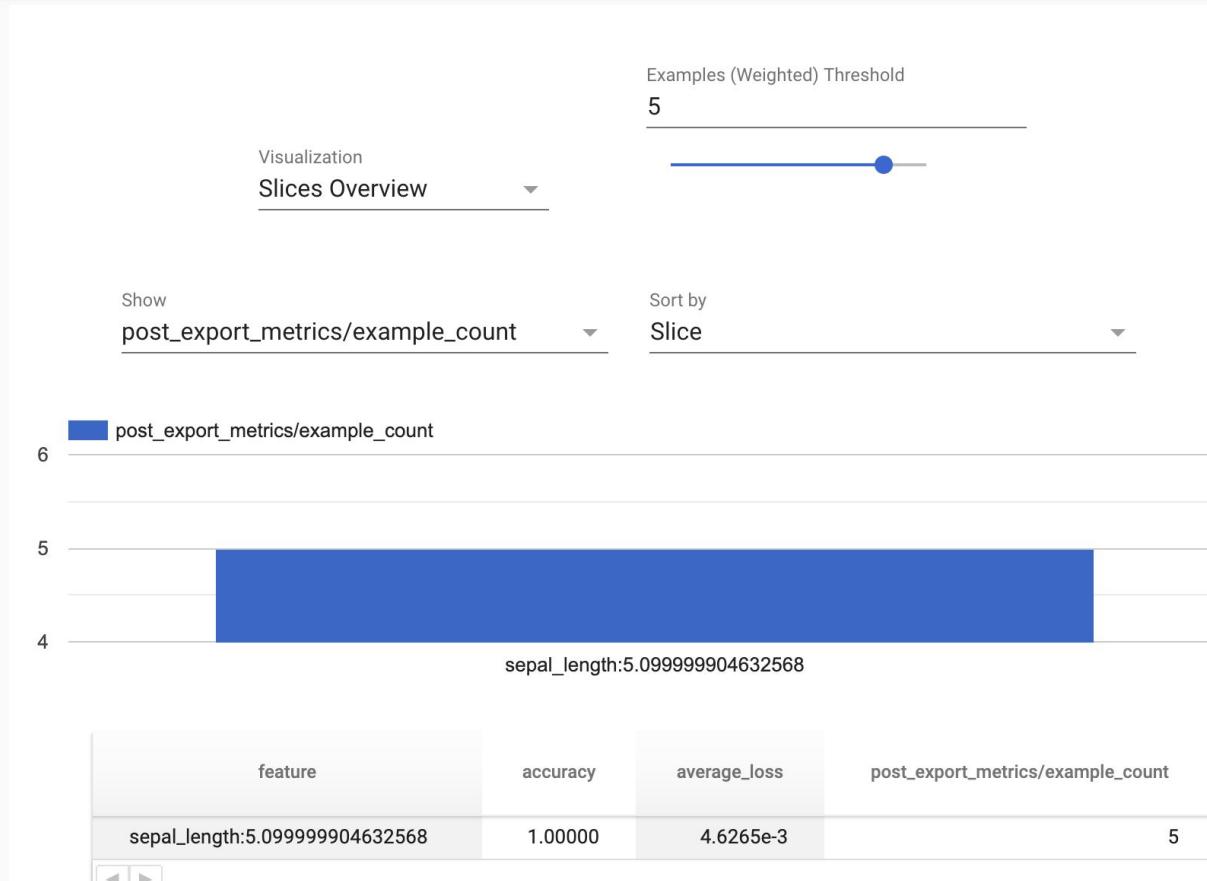
Evaluator “Visualization Example_count”



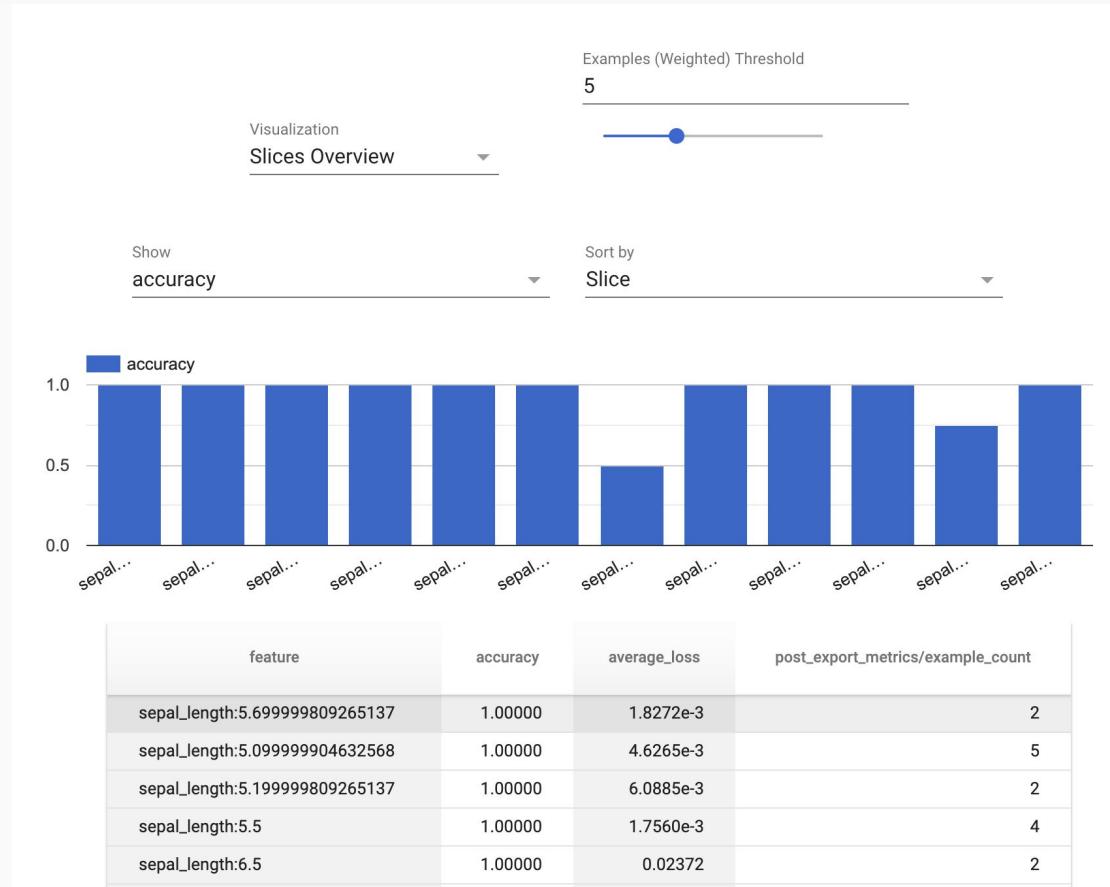
Evaluator “Visualization Example_count”



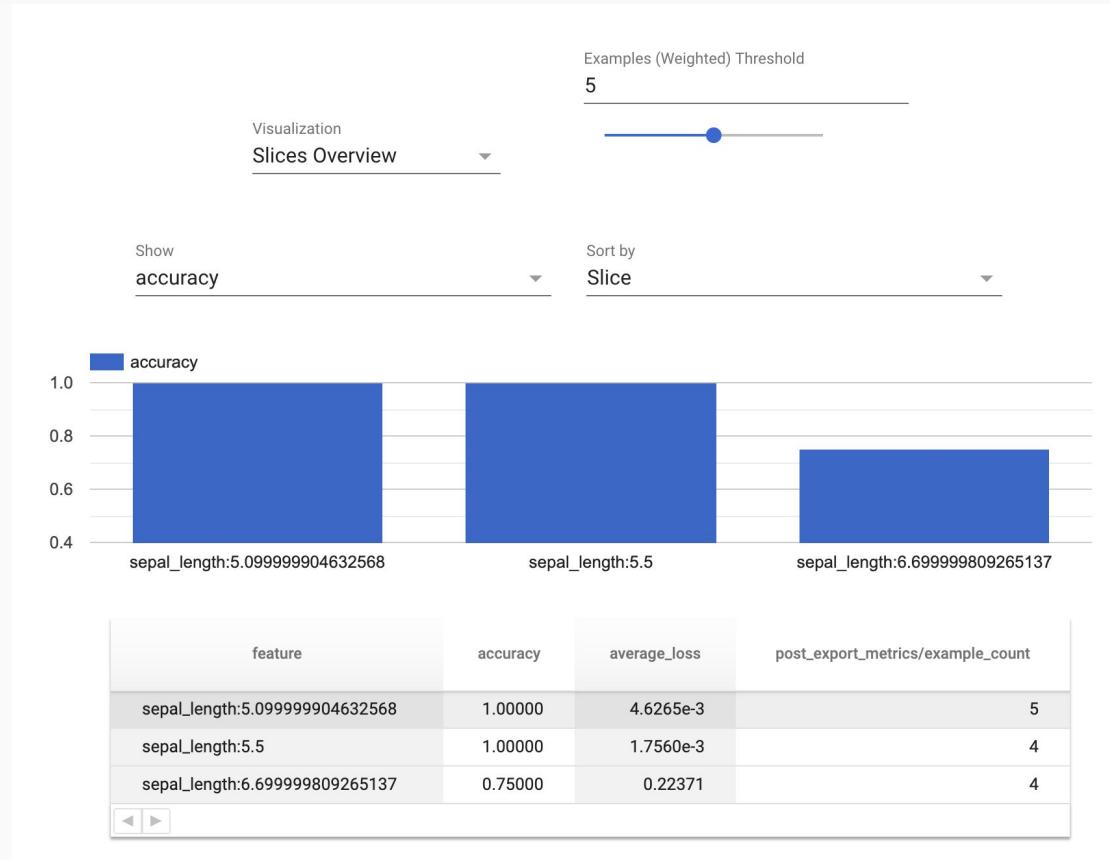
Evaluator “Visualization Example_count”



Evaluator “Visualization Accuracy”



Evaluator “Visualization Accuracy”

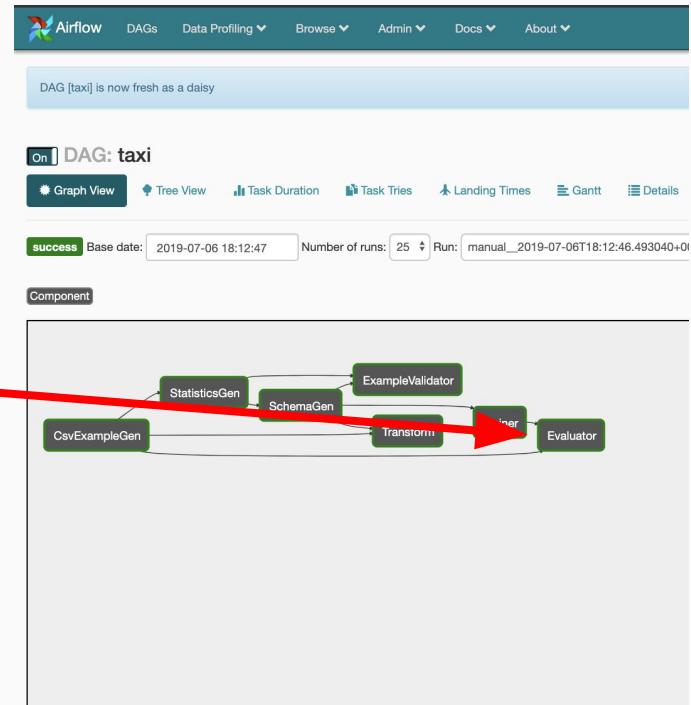


Evaluator “Settings”

- Evaluate fitted model performances with validation data
- Compare to model performances baseline

```
# Uses TFMA to compute evaluation statistics over features of a model.
model_analyzer = Evaluator( # Step 6
    examples=example_gen.outputs.examples, # Step 6
    model_exports=trainer.outputs.output, # Step 6
    feature_slicing_spec=evaluator_pb2.FeatureSlicingSpec(specs=[ # Step 6
        evaluator_pb2.SingleSlicingSpec( # Step 6
            column_for_slicing=['trip_start_hour']) # Step 6
    ]) # Step 6

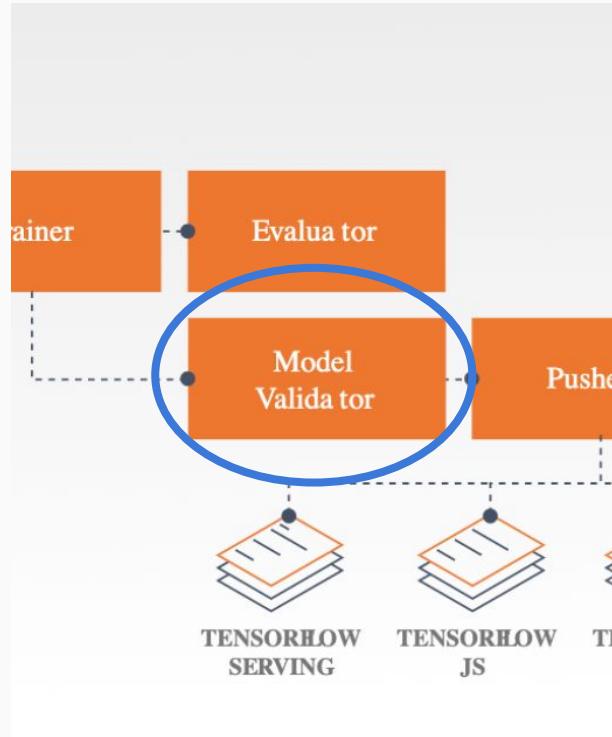
return pipeline.Pipeline(
    pipeline_name='taxi',
    pipeline_root=_pipeline_root,
    components=[
        example_gen,
        statistics_gen,
        infer_schema,
        validate_stats,
        transform,
        trainer,
        model_analyzer,
    ],
    enable_cache=True,
    metadata_db_root=_metadata_db_root,
    additional_pipeline_args={'logger_args': logger_overrides},
)
```



Sadly, this time still not support for keras

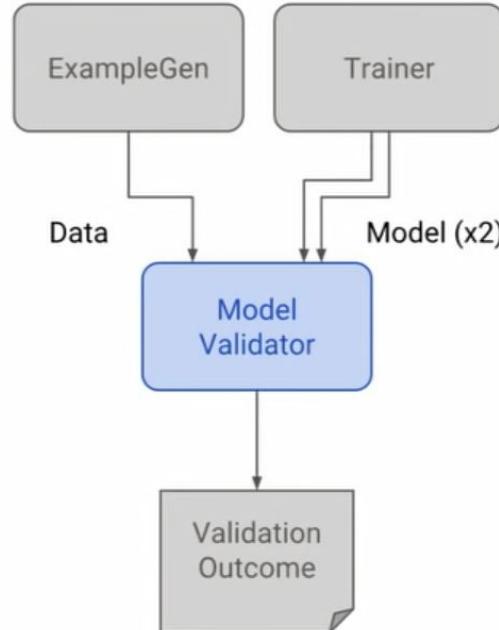
ModelValidator

- Compare new model performances with baseline
- Select good enough model to be deployed



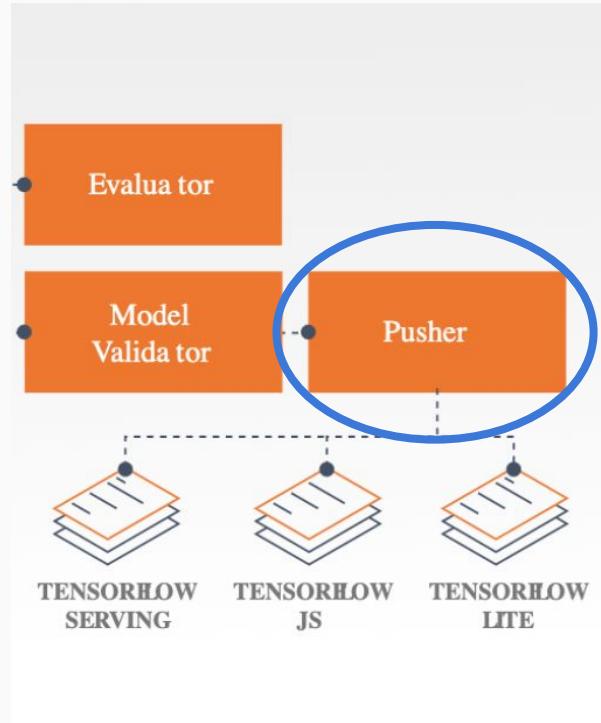
ModelValidator “Inputs and Outputs”

Inputs and Outputs

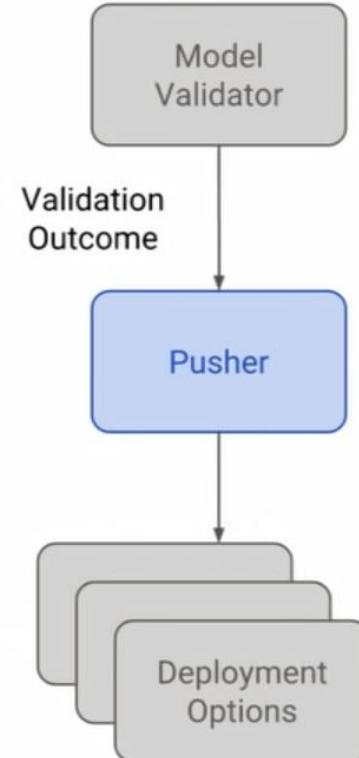


- Evaluation split of data
- Last validated model
- New candidate model

- Push selected model to the destination
- Model can be served after this operation



Pusher “Inputs and Outputs”



3. TFX

Let's do it together

** We will run our codes on EC2 instance **

** Optionally you can run codes on Google Colab on your own risk ***

https://docs.google.com/presentation/d/1vPBaVFdb9qmmKnc_xNVsRCXRA_Pv6A7fbOLvY0bs1gl/edit#slide=id.g5e3bc437cc_0_280

Go to this URL

https://docs.google.com/presentation/d/1wi6DxLrJmW06mJti8jeafKPIHO4DQeoJ9WR2_fFg39Y/edit?usp=sharing

Clone the repository

Run

```
git clone https://gitlab.com/galiblenight/mlrs_ai
```

Wait until finish

```
● ○ ● ○ Downloads — ubuntu@ip-172-31-16-10: /home/ubuntu — ssh -i mlrs_ec2.pem ubuntu@ec2...
ubuntu@ip-172-31-16-10:~/home/ubuntu$ git clone https://gitlab.com/galiblenight/mlrs_ai
```

```
● ○ ● ○ Downloads — ubuntu@ip-172-31-16-10: /home/ubuntu — ssh -i mlrs_ec2.pem ubuntu@ec2...
[ubuntu@ip-172-31-16-10:~/home/ubuntu$ git clone https://gitlab.com/galiblenight/mlrs_ai
Cloning into 'mlrs_ai'...
warning: redirecting to https://gitlab.com/galiblenight/mlrs_ai.git/
remote: Enumerating objects: 26191, done.
remote: Counting objects: 100% (26191/26191), done.
remote: Compressing objects: 100% (21002/21002), done.
remote: Total 26191 (delta 4034), reused 26157 (delta 4001)
Receiving objects: 100% (26191/26191), 260.96 MiB | 2.76 MiB/s, done.
Resolving deltas: 100% (4034/4034), done.
Checking out files: 100% (26075/26075), done.
ubuntu@ip-172-31-16-10:~/home/ubuntu$
```

Setup an environment

1

cd mlrs_ai # get in the directory
source setup.sh # run setup environment

2

sudo echo "deb
http://storage.googleapis.com/tensorflow-servi
ng-apt stable tensorflow-model-server
tensorflow-model-server-universal" | sudo tee
/etc/apt/sources.list.d/tensorflow-serving.list
&& curl
[https://storage.googleapis.com/tensorflow-servi
ng-apt/tensorflow-serving.release.pub.gpg](https://storage.googleapis.com/tensorflow-servi
ng-apt/tensorflow-serving.release.pub.gpg) |
sudo apt-key add - # add tfserver in your
repository

3

sudo apt update # update apt to get tf-serving
sudo apt-get install tensorflow-model-server #
Install TF server

Command

Output

● ● ● Downloads — ubuntu@ip-172-31-16-10: /home/ubuntu/mlrs_ai — ssh -i mlrs_ec2.pem ubun...
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
Requirement already satisfied: virtualenv in /usr/local/lib/python3.6/dist-packages
airflow: command not found
mkdir: cannot create directory 'airflow/data': No such file or directory
mkdir: cannot create directory 'airflow/data/iris': No such file or directory
mkdir: cannot create directory 'airflow/dags': No such file or directory
mkdir: cannot create directory 'airflow/tfx': No such file or directory
cp: cannot create regular file 'airflow/data/iris/': No such file or directory
deb http://storage.googleapis.com/tensorflow-serving-apt stable tensorflow-model-server ten
sorflow-model-server-universal
% Total % Received % Xferd Average Speed Time Time Current
Dload Upload Total Spent Left Speed
100 2943 100 2943 0 0 124k 0 --:--:-- --:--:-- 124k
OK
(Hit:1 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease
(Get:2 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
)
(Hit:3 http://storage.googleapis.com/tensorflow-serving-apt stable InRelease
(Get:4 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 252 kB in 1s (202 kB/s)
Reading package lists... Done
Building dependency tree... 50%

[(venv) root@ip-172-31-16-10:~# exit
exit
[ubuntu@ip-172-31-16-10: /home/ubuntu/mlrs_ai\$ ls
TFX airflow data.csv setup.sh venv
ubuntu@ip-172-31-16-10: /home/ubuntu/mlrs_ai\$]

Bug Fixed (for Transform component) ***

Copy this and run in your instance

```
cd /home/ubuntu/mlrs_ai/ # Get in folder  
  
curl -L "https://docs.google.com/uc?export=download&id=1X9fbkx6F752XX02pJIJP3g4Di1v7adef" > requirements.txt #  
Download requirements file  
  
source /home/ubuntu/mlrs_ai/venv/bin/activate  
  
pip install -r requirements.txt # install new packages  
  
pip uninstall tensorflow_estimator  
pip install -Iv tensorflow_estimator==1.13.0
```

Then you have to restart all screen applications

```
killall screen
```

Then continue to run other services in next page

Run jupyter notebook

Command

1 screen # Run the screen

```
[ubuntu@ip-172-31-16-10:/home/ubuntu/mlrs_ai$ ls  
TFX airflow data.csv setup.sh venv  
ubuntu@ip-172-31-16-10:/home/ubuntu/mlrs_ai$ screen
```

```
cd /home/ubuntu/mlrs_ai/  
source venv/bin/activate
```

```
jupyter notebook --ip 0.0.0.0 --port 8888 #  
Run jupyter notebook
```

2

```
● ○ ● Downloads — ubuntu@ip-172-31-16-10: /home/ubuntu/mlrs_ai — ssh -i mlrs_ec2.pem ubun...  
See "man sudo_root" for details.  
  
ubuntu@ip-172-31-16-10:/home/ubuntu/mlrs_ai$ source venv/bin/activate  
(venv) ubuntu@ip-172-31-16-10:/home/ubuntu/mlrs_ai$ jupyter notebook --ip 0.0.0.0 --port 88  
88  
[I 19:22:03.851 NotebookApp] Writing notebook server cookie secret to /home/ubuntu/mlrs_ec2/.local/share/jupyter/runtime/notebook_cookie_secret  
[I 19:22:04.061 NotebookApp] Serving notebooks from local directory: /home/ubuntu/mlrs_ai  
[I 19:22:04.062 NotebookApp] The Jupyter Notebook is running at:  
[I 19:22:04.062 NotebookApp] http://ip-172-31-16-10:8888/?token=33f56ca6308bc5e7b3491b6ee04bf  
bf336e8628baf1863d0d5  
[I 19:22:04.062 NotebookApp] or http://127.0.0.1:8888/?token=33f56ca6308bc5e7b3491b6ee04bf  
336e8628baf1863d0d5  
[I 19:22:04.062 NotebookApp] Use Control-C to stop this server and shut down all kernels (t  
wice to skip confirmation).  
[W 19:22:04.066 NotebookApp] No web browser found: could not locate runnable browser.  
[C 19:22:04.066 NotebookApp]  
  
To access the notebook, open this file in a browser:  
file:///home/ubuntu/mlrs_ec2/.local/share/jupyter/runtime/nserver-13750-open.html  
Or copy and paste one of these URLs:  
http://ip-172-31-16-10:8888/?token=33f56ca6308bc5e7b3491b6ee04bf336e8628baf1863d0d5  
or http://127.0.0.1:8888/?token=33f56ca6308bc5e7b3491b6ee04bf336e8628baf1863d0d5
```

3 After that press “Ctrl+a” and “d” to exit screen

Copy this

Open jupyter notebook

Actions	Connect	Launch Instance

search : mlrs Add filter

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
i-08cf60a11d5ca6319	t2.medium	ap-southeast-1b	running	2/2 checks ...	None	ec2-13-250-95-92.ap-s...	13.250.95.92	

```
Downloads — ubuntu@ip-172-31-16-10:~/home/ubuntu/mlrs_ai — ssh -i mlrs_ec2.pem ubun...
See "man sudo_root" for details.

ubuntu@ip-172-31-16-10:~/home/ubuntu/mlrs_ai$ source venv/bin/activate
(venv) ubuntu@ip-172-31-16-10:~/home/ubuntu/mlrs_ai$ jupyter notebook --ip 0.0.0.0 --port 8888
[I 19:22:03.851 NotebookApp] Writing notebook server cookie secret to /home/ubuntu/mlrs_ec2/.local/share/jupyter/runtime/notebook_cookie_secret
[I 19:22:04.061 NotebookApp] Serving notebooks from local directory: /home/ubuntu/mlrs_ai
[I 19:22:04.062 NotebookApp] The Jupyter Notebook is running at:
[I 19:22:04.062 NotebookApp] http://ip-172-31-16-10:8888/?token=33f56ca6308bc5e7b3491b6ee04bf336e8628baf1863d0d5
[I 19:22:04.062 NotebookApp] or http://127.0.0.1:8888/?token=33f56ca6308bc5e7b3491b6ee04bf336e8628baf1863d0d5
[I 19:22:04.062 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 19:22:04.066 NotebookApp] No web browser found: could not locate runnable browser.
[C 19:22:04.066 NotebookApp]

To access the notebook, open this file in a browser:
  file:///home/ubuntu/mlrs_ec2/.local/share/jupyter/runtime/nbserver-13750-open.html
Or copy and paste one of these URLs:
  http://ip-172-31-16-10:8888/?token=33f56ca6308bc5e7b3491b6ee04bf336e8628baf1863d0d5
  or http://127.0.0.1:8888/?token=33f56ca6308bc5e7b3491b6ee04bf336e8628baf1863d0d5
```

Open jupyter notebook at url..

http://{your EC2 ip addr}:8888/?token={your token}

Open jupyter notebook

The screenshot shows a Jupyter Notebook interface with a blue header bar. The main content area displays a file browser. At the top left is the Jupyter logo. On the right are 'Quit' and 'Logout' buttons. Below the header is a navigation bar with 'Files' (selected), 'Running', and 'Clusters' tabs. A message 'Select items to perform actions on them.' is displayed above the file list. On the right side of the file list are 'Upload', 'New', and a refresh icon. The file list itself has a header row with 'Name', 'Last Modified', and 'File size' columns. The data is presented in a table:

	Name	Last Modified	File size
<input type="checkbox"/>	/		
<input type="checkbox"/>	airflow	25 minutes ago	
<input type="checkbox"/>	TFX	an hour ago	
<input type="checkbox"/>	venv	26 minutes ago	
<input type="checkbox"/>	data.csv	an hour ago	4.61 kB
<input type="checkbox"/>	setup.sh	27 minutes ago	981 B

Run airflow webserver

1

screen # Run the another screen

```
[detached from 15734 pts-0 ip-172-31-16-10]
[venv] ubuntu@ip-172-31-16-10:~/home/ubuntu/mlrs_ai$ ls
TFX airflow data.csv setup.sh venv
[venv] ubuntu@ip-172-31-16-10:~/home/ubuntu/mlrs_ai$ screen
```

2

```
cd /home/ubuntu/mlrs_ai/  
source venv/bin/activate  
airflow webserver -p 8080 # Run airflow  
webserver
```

```
[detached from 27504 pts-0. ip-172-31-16-10]
[venv] ubuntu@ip-172-31-16-10:~$ airflow webserver -p 8080
[2019-08-05 05:47:32,751] {__init__.py:51} INFO - Using executor SequentialExecutor
_____
_____|_(_)_|_____| / |_____| / |
_____| / | / | / | / | / | / | / |
_____| / | / | / | / | / | / | / |
_____| / | / | / | / | / | / | / |
[2019-08-05 05:47:33,103] {__init__.py:305} INFO - Filling up the DagBag from /home/ubuntu/mlrs_ai/www/dags
/home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/airflow/www/app.py:193: DeprecationWarning: 'werkzeug.wsgi.DispatcherMiddleware' has moved to 'werkzeug.middleware.dispatcher.DispatcherMiddleware'. This import is deprecated as of version 0.15 and will be removed in version 1.0.
    app = DispatcherMiddleware(root_app, {base_url: app})
Running the Gunicorn Server with:
Workers: 4 sync
Host: 0.0.0.0:8080
Timeout: 120
Logfiles: -
=====
[2019-08-05 05:47:34 +0000] [27561] [INFO] Starting gunicorn 19.9.0
[2019-08-05 05:47:34 +0000] [27561] [INFO] Listening at: http://0.0.0.0:8080 (27)
```

2

press “Ctrl+a” and “d” to exit screen

Run airflow scheduler

1 screen # Run the another screen

```
[detached from 15754 pts/0] ip 172-31-10-10]
(venv) ubuntu@ip-172-31-16-10:~/home/ubuntu/mlrs_ais$ ls
TFX airflow data.csv setup.sh venv
(venv) ubuntu@ip-172-31-16-10:~/home/ubuntu/mlrs_ais$ screen
```

```
2 source venv/bin/activate
airflow scheduler # Run airflow
scheduler for using trigger DAGs
```

3 press “Ctrl+a” and “d” to exit screen

Open airflow webserver

Launch Instance		Connect	Actions						
<input type="text"/> search : mtrs (?) Add filter (K < 1 to 1 of 1 >)									
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	
i-08cf60a11d5ca6319	i2.medium	ap-southeast-1b	running	2/2 checks ...	None	ec2-13-250-95-92.ap-s...	13.250.95.92		

Go to this URL: `http://<your ec2 ip addr>:8080` # port 8080 as in settings

Write iris DAG file

Back to jupyter notebook
and go to

/TFX/TFX_Files.ipynb

The screenshot shows the Jupyter Notebook interface with the 'Files' tab selected. The file tree on the left shows a directory structure: airflow, TFX (selected), venv, data.csv, and setup.sh. A red arrow labeled '1' points to the 'TFX' folder. The list of files on the right shows:

Name	Last Modified	File size
seconds ago		
11 hours ago		
10 hours ago		
11 hours ago	4.61 kB	
21 minutes ago	1.25 kB	

The screenshot shows the Jupyter Notebook interface with the 'Files' tab selected. The file tree on the left shows a directory structure: .., model_validation.ipynb, statistics_and_schema.ipynb, TFX_Files.ipynb (selected), training.ipynb, transform.ipynb, airflow_url.txt, tfx_utils.py, and utils.py. A red arrow labeled '2' points to the 'TFX_Files.ipynb' file. The list of files on the right shows:

Name	Last Modified	File size
seconds ago		
11 hours ago	11.5 kB	
11 hours ago	11.6 kB	
11 hours ago	20.3 kB	
11 hours ago	2.91 kB	
11 hours ago	16.5 kB	
11 hours ago	0 B	
11 hours ago	7.89 kB	
11 hours ago	19.3 kB	

Write iris DAG file

iris_pipeline.py

Pipeline structure definition of iris dag

```
In [2]: 1 %%writefile ../airflow/dags/iris_pipeline.py
2
3 from __future__ import absolute_import
4 from __future__ import division
5 from __future__ import print_function
6
7 import datetime
8 import logging
9 import os
10 import json
11 import requests
12 import numpy as np
13
14 from tfx.components.example_gen.csv_example_gen.component import CsvExampleGen
15
16 from tfx.components.statistics_gen.component import StatisticsGen
17 from tfx.components.schema_gen.component import SchemaGen
18 from tfx.components.example_validator.component import ExampleValidator
19
20 from tfx.components.transform.component import Transform
21
22 from tfx.orchestration import pipeline
23
24 from tfx.proto import trainer_pb2
25 from tfx.components.trainer.component import Trainer
26
27 from tfx.proto import evaluator_pb2
28 from tfx.components.evaluator.component import Evaluator
29
30 from tfx.proto import pusher_pb2
31 from tfx.components.model_validator.component import ModelValidator
32 from tfx.components.pusher.component import Pusher
33
34
```

Cell for write iris_pipeline.py

To create Airflow DAG

iris_utils.py

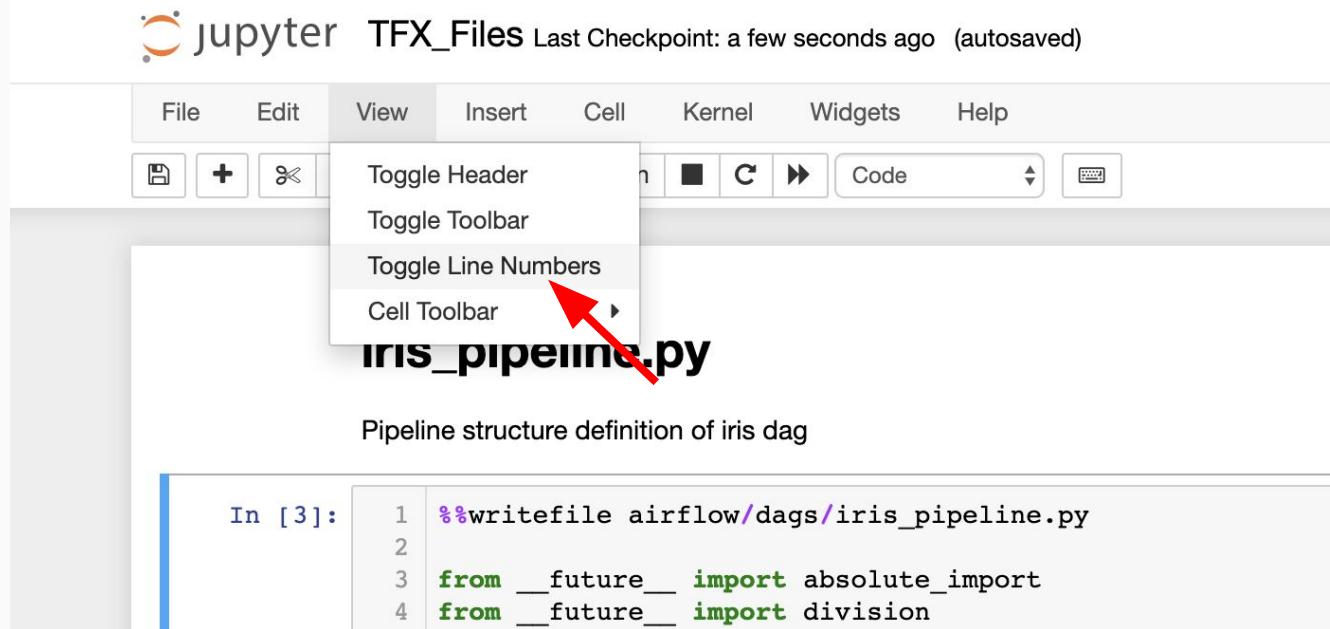
Utility functions for transform data and training models

```
In [3]: 1 %%writefile ../airflow/dags/iris_utils.py
2
3 import os
4
5 import tensorflow as tf
6
7 import tensorflow_transform as tft # Step 2
8 from tensorflow_transform.beam.tft_beam_io import transform_fn_io # Step 2
9 from tensorflow_transform.saved import saved_transform_io # Step 2
10 from tensorflow_transform.tf_metadata import metadata_io # Step 2
11 from tensorflow_transform.tf_metadata import schema_utils # Step 2
12
13 import tensorflow_model_analysis as tma # Step 3
14 from tensorflow.layers import dense # Step 3
15 ModeKeys = tf.estimator.ModeKeys # Step 3
16 TRAIN, EVAL, PREDICT = ModeKeys.TRAIN, ModeKeys.EVAL, ModeKeys.PREDICT # Step 3
17
18 _DENSE_FLOAT_FEATURE_KEYS = ["sepal_length", "sepal_width", "petal_length", "petal_width"]
19
20 _LABEL_KEY = 'class'
21
22 # Step 2 START -----
23
24 # Function for change name of preprocessed feature
25 def _transformed_name(key):
26     return key + '_xf'
27
28 # Function for change names of preprocessed features
29 def _transformed_names(keys):
30     return [_transformed_name(key) for key in keys]
31
32 # Function for getting feature specs of schema for using in input_fn functions
33 def _get_raw_feature_spec(schema):
34     return schema_utils.schema_as_feature_spec(schema).feature_spec
```

Cell for write iris_utils.py

Util functions of the pipeline

Show line numbers



If line numbers not shown, you can toggle it via menu “View/Toggle Line Numbers”

Create iris pipeline

To write new (example) DAGs files

1

Run write_iris_pipeline.py
Run write_iris_utils.py

Shift+Enter

After create new DAG file, airflow may not process it immediately.

2

You might run `airflow list_dags` to force airflow read all files

Shift+Enter

List DAGs (for refresh)

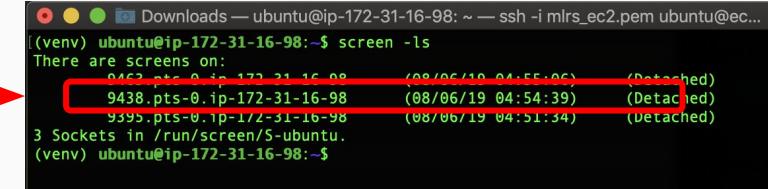
```
In [12]: 1 !airflow list_dags 2> /dev/null
```

101

Restart airflow webserver (if iris pipeline is not shown)

1

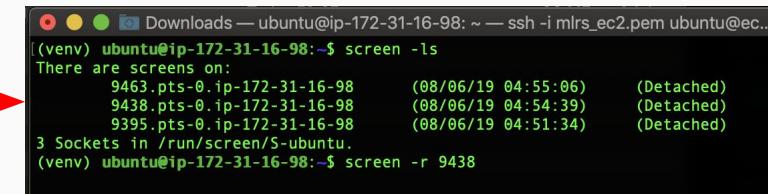
```
screen -ls # List all screen processes  
the process number will sorted in  
descending order "So airflow webserver  
is in second line"
```



```
(venv) ubuntu@ip-172-31-16-98:~$ screen -ls  
There are screens on:  
 9463.pts-0.ip-172-31-16-98          (08/06/19 04:55:06)  (Detached)  
 9438.pts-0.ip-172-31-16-98          (08/06/19 04:54:39)  (Detached)  
 9395.pts-0.ip-172-31-16-98          (08/06/19 04:51:34)  (Detached)  
3 Sockets in /run/screen/S-ubuntu.  
(venv) ubuntu@ip-172-31-16-98:~$
```

2

```
screen -r < process-id > # Connect to  
the screen
```



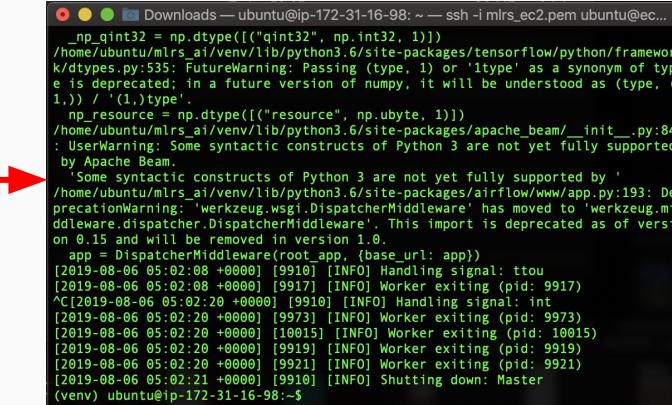
```
(venv) ubuntu@ip-172-31-16-98:~$ screen -ls  
There are screens on:  
 9463.pts-0.ip-172-31-16-98          (08/06/19 04:55:06)  (Detached)  
 9438.pts-0.ip-172-31-16-98          (08/06/19 04:54:39)  (Detached)  
 9395.pts-0.ip-172-31-16-98          (08/06/19 04:51:34)  (Detached)  
3 Sockets in /run/screen/S-ubuntu.  
(venv) ubuntu@ip-172-31-16-98:~$ screen -r 9438
```

3

Stop airflow webserver by Ctrl+C and
then start it again

4

```
airflow webserver -p 8080 # Connect to  
the screen
```



```
_np_qint32 = np.dtype([('qint32', np.int32, 1)])  
/home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:535: FutureWarning: Passing (type, 1) or 'Itype' as a synonym for type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,Itype'.  
  np_resource = np.dtype([('resource', np.ubyte, 1)])  
/home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/apache_beam/_init_.py:84 : UserWarning: Some syntactic constructs of Python 3 are not yet fully supported by Apache Beam.  
  'Some syntactic constructs of Python 3 are not yet fully supported by '  
/home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/airflow/www/app.py:193: DeprecationWarning: 'werkzeug.wsgi.DispatcherMiddleware' has moved to 'werkzeug.middleware.dispatcher.DispatcherMiddleware'. This import is deprecated as of version 0.15 and will be removed in version 1.0.  
    app = DispatcherMiddleware(root_app, {base_url: app})  
[2019-08-06 05:02:08 +0000] [9918] [INFO] Handling signal: ttou  
[2019-08-06 05:02:08 +0000] [9917] [INFO] Worker exiting (pid: 9917)  
^C[2019-08-06 05:02:20 +0000] [9910] [INFO] Handling signal: int  
[2019-08-06 05:02:20 +0000] [9973] [INFO] Worker exiting (pid: 9973)  
[2019-08-06 05:02:20 +0000] [10015] [INFO] Worker exiting (pid: 10015)  
[2019-08-06 05:02:20 +0000] [9919] [INFO] Worker exiting (pid: 9919)  
[2019-08-06 05:02:20 +0000] [9921] [INFO] Worker exiting (pid: 9921)  
[2019-08-06 05:02:21 +0000] [9918] [INFO] Shutting down: Master  
(venv) ubuntu@ip-172-31-16-98:~$
```

5

Then press "Ctrl+a" and "d" to exit
screen

Create iris pipeline

From the settings in **iris_pipeline.py**
Set the pipeline (DAG) name as here

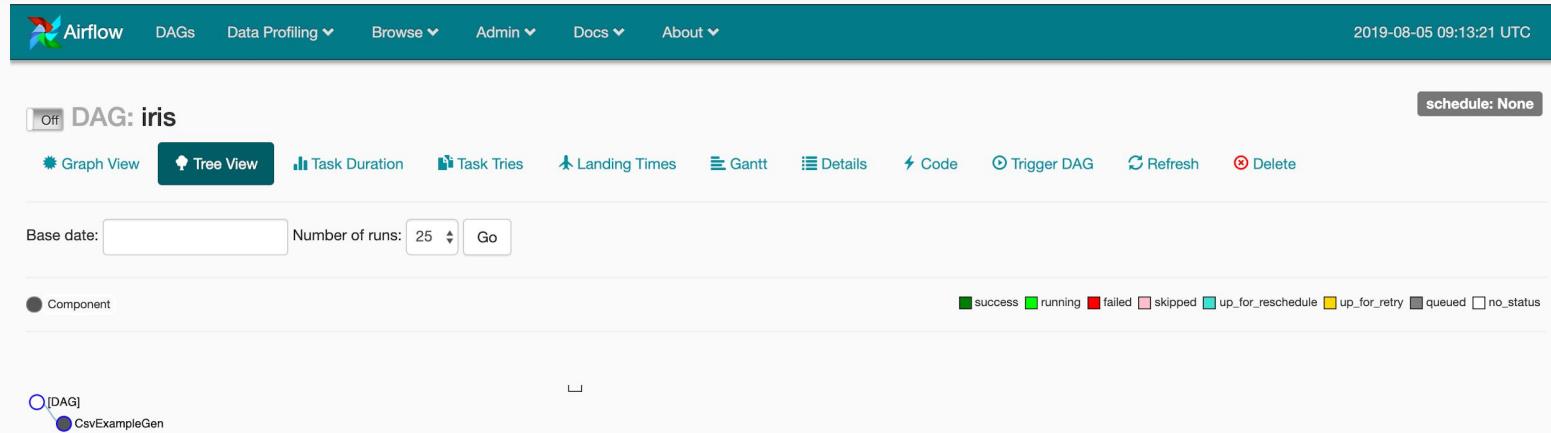
```
111
112     return pipeline.Pipeline(
113         pipeline_name='iris',
114         pipeline_root=_pipeline_root,
115         components=[
116             example_gen,
117             # statistics_gen, infer_schema, validate_stats, # Step 1
118             # transform, # Step 2
119             # trainer, # Step 3
120             # model_analyzer, # Step 4
121             # model_validator, pusher # Step 5
122         ],
123         enable_cache=True,
124         metadata_db_root=_metadata_db_root,
125         additional_pipeline_args={'logger_args': logger_overrides},
126     )
```

When refresh the page, you must see iris pipeline



<input checked="" type="checkbox"/>	<input type="button" value="Off"/>	example_subdag_operator	@once
<input checked="" type="checkbox"/>	<input type="button" value="Off"/>	example_trigger_controller_dag	@once
<input checked="" type="checkbox"/>	<input type="button" value="Off"/>	example_trigger_target_dag	None
<input checked="" type="checkbox"/>	<input type="button" value="Off"/>	example_xcom	@once
<input checked="" type="checkbox"/>	<input type="button" value="Off"/>	iris	None
<input checked="" type="checkbox"/>	<input type="button" value="Off"/>	latest_only	4:00:00
<input checked="" type="checkbox"/>	<input type="button" value="Off"/>	latest_only_with_trigger	4:00:00

Iris pipeline first see



Pipeline of TFX

Review the iris_pipeline.py

```
1 %%writefile ../airflow/dags/iris_pipeline.py
2
3 from __future__ import absolute_import
4 from __future__ import division
5 from __future__ import print_function
6
7 import datetime
8 import logging
9 import os
10 import json
11 import requests
12 import numpy as np
13
14 from tfx.components.example_gen.csv_example_gen.component import CsvExampleGen
15
16 from tfx.components.statistics_gen.component import StatisticsGen
17 from tfx.components.schema_gen.component import SchemaGen
18 from tfx.components.example_validator.component import ExampleValidator
19
20 from tfx.components.transform.component import Transform
21
22 from tfx.orchestration import pipeline
23
24 from tfx.proto import trainer_pb2
25 from tfx.components.trainer.component import Trainer
26
27 from tfx.proto import evaluator_pb2
28 from tfx.components.evaluator.component import Evaluator
29
30 from tfx.proto import pusher_pb2
31 from tfx.components.model_validator.component import ModelValidator
32 from tfx.components.pusher.component import Pusher
33
34 from tfx.orchestration.airflow.airflow_runner import AirflowDAGRunner
35 from tfx.utils.dsl_utils import csv_input
```

Components

Libraries Import section

Review the iris_pipeline.py

Parameters settings

```
36
37 _airflow_root = os.path.join(os.environ['AIRFLOW_HOME']) # Set airflow home
38 _data_root = os.path.join(_airflow_root, 'data/iris') # set data path
39
40 _iris_module_file = os.path.join(_airflow_root, 'dags/iris_utils.py') # set module file (utils)
41 _serving_model_dir = os.path.join(_airflow_root, 'saved_models/iris') # set where to serve model
42
43 _tfx_root = os.path.join(_airflow_root, 'tfx') # set where is tfx home
44 _pipeline_root = os.path.join(_tfx_root, 'pipelines') # set where pipelines metadata will be saved
45 _metadata_db_root = os.path.join(_tfx_root, 'metadata') # set where tfx metadata will be saved
46 _log_root = os.path.join(_tfx_root, 'logs') # set where is logs path
47
48 _airflow_config = {
49     'schedule_interval': None, # Set how often DAG should be triggered
50     'start_date': datetime.datetime(2019, 1, 1), # Set start date of this DAG
51 }
52
53 logger_overrides = {
54     'log_root': _log_root, # Set where to save log
55     'log_level': logging.INFO # Set log levels
56 }
```

Review the iris_pipeline.py

```
57 def _create_iris_pipeline():
58     examples = csv_input(_data_root)
59     example_gen = CsvExampleGen(input_base=examples)
60
61     # Computes statistics over data for visualization and example validation.
62     #   statistics_gen = StatisticsGen(input_data=example_gen.outputs.examples) # Step 1
63
64     # Generates schema based on statistics files.
65     #   infer_schema = SchemaGen(stats=statistics_gen.outputs.output) # Step 1
66
67     # Performs anomaly detection based on statistics and data schema.
68     #   validate_stats = ExampleValidator( # Step 1
69     #       stats=statistics_gen.outputs.output, # Step 1
70     #       schema=infer_schema.outputs.output) # Step 1
71
72     # Performs transformations and feature engineering in training and serving.
73     #   transform = Transform( # Step 2
74     #       input_data=example_gen.outputs.examples, # Step 2
75     #       schema=infer_schema.outputs.output, # Step 2
76     #       module_file=_iris_module_file) # Step 2
77
78     # Uses user-provided Python function that implements a model using TF-Learn.
79     #   trainer = Trainer( # Step 3
80     #       module_file=_iris_module_file, # Step 3
81     #       transformed_examples=transform.outputs.transformed_examples, # Step 3
82     #       schema=infer_schema.outputs.output, # Step 3
83     #       transform_output=transform.outputs.transform_output, # Step 3
84     #       train_args=trainer_pb2.TrainArgs(num_steps=10000), # Step 3
85     #       eval_args=trainer_pb2.EvalArgs(num_steps=5000)) # Step 3
86
87     # Uses TFMA to compute a evaluation statistics over features of a model.
88     #   model_analyzer = Evaluator( # Step 4
89     #       examples=example_gen.outputs.examples, # Step 4
90     #       model_exports=trainer.outputs.output, # Step 4
91     #       feature_slicing_spec=evaluator_pb2.FeatureSlicingSpec(specs=[ # Step 4
92     #           evaluator_pb2.SingleSlicingSpec( # Step 4
93     #               ...)
```

} Uncommented steps

Pipeline definition

} Commented steps

Review the iris_pipeline.py

Pipeline settings (return pipeline)

```
111
112     return pipeline.Pipeline(
113         pipeline_name='iris',
114         pipeline_root=_pipeline_root,
115         components=[
116             example_gen, # Step 1
117             statistics_gen, infer_schema, validate_stats, # Step 1
118             transform, # Step 2
119             trainer, # Step 3
120             model_analyzer, # Step 4
121             model_validator, pusher # Step 5
122         ],
123         enable_cache=True,
124         metadata_db_root=metadata_db_root,
125         additional_pipeline_args={'logger_args': logger_overrides},
126     )
127
128
129     airflow_pipeline = AirflowDAGRunner(
130         _airflow_config).run(_create_iris_pipeline())
```

Uncommented:
Added components

Commented:
Future steps

Review the iris_utils.py

```
2 import os
3
4 import tensorflow as tf
5
6
7 import tensorflow_transform as tft # Step 2
8 from tensorflow_transform.beam.tft_beam_io import transform_fn_io # Step 2
9 from tensorflow_transform.saved import saved_transform_io # Step 2
10 from tensorflow_transform.tf_metadata import metadata_io # Step 2
11 from tensorflow_transform.tf_metadata import schema_utils # Step 2
12
13 import tensorflow_model_analysis as tfma # Step 3
14 from tensorflow.layers import dense # Step 3
```

Libraries Import section

Review the iris_utils.py

```
15 ModeKeys = tf.estimator.ModeKeys # Step 3
16 TRAIN, EVAL, PREDICT = ModeKeys.TRAIN, ModeKeys.EVAL, ModeKeys.PREDICT # Step 3
17
18 _DENSE_FLOAT_FEATURE_KEYS = ["sepal_length", "sepal_width", "petal_length", "petal_width"]
19
20 _LABEL_KEY = 'class'
21
22
```

Parameter settings

Review the iris_utils.py

```
25 # Function for change name of preprocessed feature
26 def _transformed_name(key):
27     return key + '_xf'
28
29 # Function for change names of preprocessed features
30 def _transformed_names(keys):
31     return [_transformed_name(key) for key in keys]
32
33 # Function for getting feature specs of schema for using in input_fn functions
34 def _get_raw_feature_spec(schema):
35     return schema_utils.schema_as_feature_spec(schema).feature_spec
36
37 # Function to read gzip files
38 def _gzip_reader_fn():
39     """Small utility returning a record reader that can read gzip'ed files."""
40     return tf.TFRecordReader(
41         options=tf.python_io.TFRecordOptions(
42             compression_type=tf.python_io.TFRecordCompressionType.GZIP))
43
44 # Function for change sparse tensor to dense (Tensorflow layers required dense tensor)
45 def _fill_in_missing(x):
46     """Replace missing values in a SparseTensor.
47
48     Fills in missing values of `x` with '' or 0, and converts to a dense tensor.
49
50     Args:
51         x: A `SparseTensor` of rank 2. Its dense shape should have size at most 1
52             in the second dimension.
53
54     Returns:
55         A rank 1 tensor where missing values of `x` have been filled in.
56     """
57     default_value = '' if x.dtype == tf.string else 0
58     return tf.squeeze(
59         tf.sparse.to_dense(
60             tf.SparseTensor(x.indices, x.values, [x.dense_shape[0], 1]),
61             default_value),
62         axis=1)
63
64 # Function for Preprocess in transformation step (Step 2).
65 # Specific function name.
```

Preprocessing methods

Review the iris_utils.py

```
67 def preprocessing_fn(inputs):
68     outputs = {}
69
70     # Change input to z-score
71     for key in _DENSE_FLOAT_FEATURE_KEYS:
72         outputs[_transformed_name(key)] = tft.scale_to_z_score(
73             _fill_in_missing(inputs[key]))
74
75     outputs[_transformed_name(_LABEL_KEY)] = tft.compute_and_apply_vocabulary(
76         _fill_in_missing(inputs[_LABEL_KEY]), # Sparse to dense
77         top_k=3,
78         num_oov_buckets=0)
79
80     return outputs
81
82
83 }
```

Preprocessing function
(TFX will call this to transform
data)

Review the iris_utils.py

Methods to create model

Review the iris_utils.py

```
231 # Function for create training model of TFX --> Train, Eval, Test
232 # Specific function name.
233 # TFX will call this function ****
234
235
236 def trainer_fn(hparams, schema):
237     """Build the estimator using the high level API.
238
239     Args:
240         hparams: Holds hyperparameters used to train the model as name/value pairs
241         schema: Holds the schema of the training examples.
242
243     Returns:
244         A dict of the following:
245         - estimator: The estimator that will be used for training and eval.
246         - train_spec: Spec for training.
247         - eval_spec: Spec for eval.
248         - eval_input_receiver_fn: Input function for eval.
249     """
250
251     train_batch_size = 32
252     eval_batch_size = 32
253
254     train_input_fn = lambda: _input_fn(
255         hparams.train_files,
256         hparams.transform_output,
257         batch_size=train_batch_size)
258
259     eval_input_fn = lambda: _input_fn(
260         hparams.eval_files,
261         hparams.transform_output,
262         batch_size=eval_batch_size)
263
264     train_spec = tf.estimator.TrainSpec(
265         train_input_fn,
266         max_steps=hparams.train_steps)
267
268     serving_receiver_fn = lambda: _example_serving_receiver_fn(
269         hparams.transform_output, schema)
```

Trainer function (TFX will call this to create and train model)

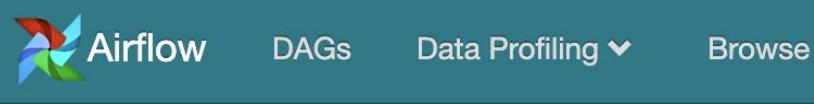
How components are added

```
57
58 def _create_iris_pipeline():
59
60     examples = csv_input(_data_root)
61     example_gen = CsvExampleGen(input_base=examples)
62
63
64     return pipeline.Pipeline(
65         pipeline_name='iris',
66         pipeline_root=_pipeline_root,
67         components=[
68             example_gen,
69             # statistics_gen, infer_schema,
70             # transform, # Step 2
71     ])
```

Component

CsvExampleGen

Before run the DAG



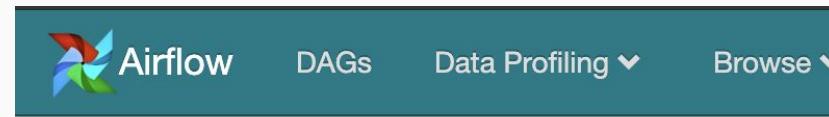
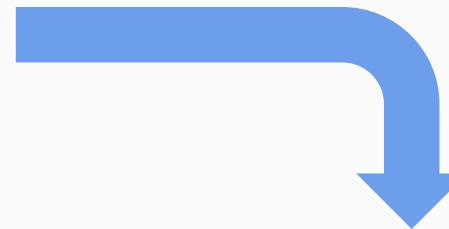
Off DAG: iris

Graph View Tree View Task Duration

Click this

A red arrow points to the 'Graph View' button, which is highlighted with a teal background and white text. The 'DAG: iris' status is currently set to 'Off'.

You must unpause the DAG !



On DAG: iris

Graph View Tree View Task Duration

Trigger the pipeline

The screenshot shows a user interface for managing a pipeline. At the top, there is a navigation bar with several tabs: Graph View (selected), Tree View, Task Duration, Task Tries, Landing Times, Gantt, Details, Code, Trigger DAG (highlighted with a red arrow), Refresh, and Delete. Below the navigation bar are search and filter controls: 'Base date:' set to '2019-08-05 10:07:35', 'Number of runs:' set to '25', 'Run:' dropdown, 'Layout:' dropdown set to 'Left->Right', and a 'Go' button. To the right is a search input field 'Search for...'. Underneath these controls is a legend for component status: success (green), running (yellow), failed (red), skipped (pink), up_for_reschedule (light green), up_for_retry (light yellow), queued (orange), and no_status (grey). A large central area contains a component named 'CsvExampleGen'. To the right of this component is a refresh icon. A red arrow points from the text 'Click to run this pipeline' down towards the 'Trigger DAG' button in the navigation bar.

Graph View

Tree View

Task Duration

Task Tries

Landing Times

Gantt

Details

Code

Trigger DAG

Refresh

Delete

None Base date: 2019-08-05 10:07:35 Number of runs: 25 Run: Layout: Left->Right Go

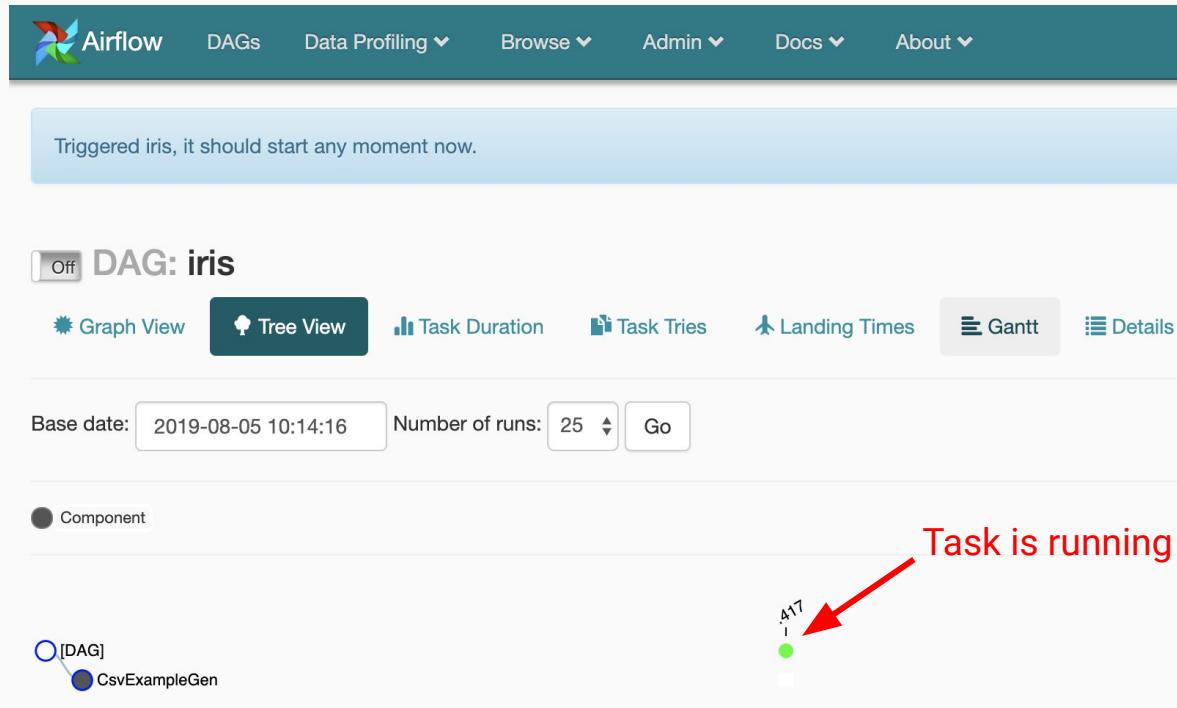
Component

success running failed skipped up_for_reschedule up_for_retry queued no_status

CsvExampleGen

Click to run this pipeline

Trigger the pipeline



Try to refresh page until the tasks are success

After trigger the pipeline, webpage will be redirected to Tree View

Trigger the pipeline

The screenshot shows the Airflow web interface for the DAG: iris. At the top, there are navigation links: Airflow, DAGs, Data Profiling, Browse, Admin, Docs, and About. The date is listed as 2019-08-05 10:23:59 UTC. Below the header, the DAG name 'iris' is displayed with a status of 'On'. A 'schedule: None' button is visible. The main area shows the DAG structure with components: [DAG] and CsvExampleGen. The CsvExampleGen component has three tasks: one green dot (success) and two black squares (running). A red arrow points to the 'Trigger DAG' button for the CsvExampleGen task. Below the DAG structure, there are filters for Base date (2019-08-05 10:14:16), Number of runs (25), and a Go button. A legend at the bottom indicates: success (green square), running (yellow square), failed (red square), skipped (pink square), up_for_reschedule (cyan square), up_for_retry (orange square), queued (grey square), and no_status (white square).

After a minute (if you refresh a web) the task should be run completely

Average time:
About 30s - 2m for each component

Statistical Schema and Data Validation

Uncomment step 1

```
111
112     return pipeline.Pipeline(
113         pipeline_name='iris',
114         pipeline_root=_pipeline_root,
115         components=[
116             example_gen,
117             #     statistics_gen, infer_schema, validate_stats, # Step 1
118             #     transform, # Step 2
119             #     trainer, # Step 3
120             #     model_analyzer, # Step 4
121             #     model_validator, pusher # Step 5
122         ],
123         enable_cache=True,
124         metadata_db_root=_metadata_db_root,
125         additional_pipeline_args={'logger_args': logger_overrides},
126     )
127 
```

```
111
112     return pipeline.Pipeline(
113         pipeline_name='iris',
114         pipeline_root=_pipeline_root,
115         components=[
116             example_gen,
117             statistics_gen, infer_schema, validate_stats, # Step 1
118             #     transform, # Step 2
119             #     trainer, # Step 3
120             #     model_analyzer, # Step 4
121             #     model_validator, pusher # Step 5
122         ],
123         enable_cache=True,
124         metadata_db_root=_metadata_db_root,
125         additional_pipeline_args={'logger_args': logger_overrides},
126     )
127 
```

```
0 1     example_gen = CsvExampleGen(input_base=examples)
0 2
0 3     # Computes statistics over data for visualization and example validation.
0 4     statistics_gen = StatisticsGen(input_data=example_gen.outputs.examples) # Step 1
0 5
0 6     # Generates schema based on statistics files.
0 7     infer_schema = SchemaGen(stats=statistics_gen.outputs.output) # Step 1
0 8
0 9     # Performs anomaly detection based on statistics and data schema.
010    # validate_stats = ExampleValidator( # Step 1
011    #     stats=statistics_gen.outputs.output, # Step 1
012    #     schema=infer_schema.outputs.output) # Step 1
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073 
```

```
62
63
64
65
66
67
68
69
70
71
72
73 
```

```
# Computes statistics over data for visualization and example validation.
statistics_gen = StatisticsGen(input_data=example_gen.outputs.examples) # Step 1

# Generates schema based on statistics files.
infer_schema = SchemaGen(stats=statistics_gen.outputs.output) # Step 1

# Performs anomaly detection based on statistics and data schema.
validate_stats = ExampleValidator( # Step 1
    stats=statistics_gen.outputs.output, # Step 1
    schema=infer_schema.outputs.output) # Step 1 
```

Statistical Schema and Data Validation

In jupyter notebook

Rewrite DAGs files

Run write iris_pipeline.py

Shift+Enter

iris_pipeline.py

Pipeline structure definition of iris dag

```
In [2]: 1 #!/usr/bin/env python
2
3 from __future__ import absolute_import
4 from __future__ import division
5 from __future__ import print_function
6
7 import datetime
8 import logging
9 import json
10 import requests
11 import numpy as np
12
13 from tfx.components.example_gen.csv_example_gen.component import CsvExampleGen
14
15 from tfx.components.statistics_gen.component import StatisticsGen
16 from tfx.components.schema_gen.component import SchemaGen
17 from tfx.components.example_validator.component import ExampleValidator
18
19 from tfx.components.transform.component import Transform
20
21 from tfx.orchestration.pipeline import pipeline
22
23 from tfx.proto import transform_pb2
24 from tfx.components.trainer.component import Trainer
25
26 from tfx.proto import evaluator_pb2
27 from tfx.components.evaluator.component import Evaluator
28
29 from tfx.proto import pusher_pb2
30 from tfx.components.model_validator.component import ModelValidator
31 from tfx.components.pusher.component import Pusher
32
33
```

In airflow webserver

DAG: iris

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Trigger DAG Refresh Delete

Base date: 2019-08-05 10:14:16 Number of runs: 25 Go

Component

DAG: CsvExampleGen



DAG [iris] is now fresh as a daisy

On DAG: iris

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Trigger DAG Refresh Delete

Base date: 2019-08-05 10:14:16 Number of runs: 25 Go

Component

DAG: CsvExampleGen

DAG: CsvExampleGen

Task: CsvExampleGen

Task: StatisticsGen

Task: SchemaGen

Task: ExampleValidator

Task: Transform

Task: Trainer

Task: Evaluator

Task: ModelValidator

Task: Pusher

Back to airflow and refresh page

It should be like this

Statistical Schema and Data Validation

On DAG: iris schedule: None

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Trigger DAG Refresh Delete

success Base date: 2019-08-05 10:14:17 Number of runs: 25 Run: manual_2019-08-05T10:14:16.417423+00:00 Layout: Left->Right Go Search for... Components success running failed skipped up_for_reschedule up_for_retry queued no_status

Component

```
graph LR; CsvExampleGen[CsvExampleGen] --> StatisticsGen[StatisticsGen]; StatisticsGen --> SchemaGen[SchemaGen]; SchemaGen --> ExampleValidator[ExampleValidator]
```

You just added new 3 components (StatisticsGen, SchemaGen and ExampleValidator)

Statistical Schema and Data Validation

Screenshot of the Airflow web interface showing the DAG: iris page.

The top navigation bar includes links for Airflow, DAGs, Data Profiling, Browse, Admin, Docs, and About, along with the timestamp 2019-08-05 15:33:20 UTC.

The main header shows "On [] DAG: iris" and "schedule: None".

Below the header are several navigation buttons: Graph View (highlighted in green), Tree View, Task Duration, Task Tries, Landing Times, Gantt, Details, Code, Trigger DAG (with a red arrow pointing to it), Refresh, and Delete.

Filtering options include: success (green), Base date: 2019-08-05 10:14:17, Number of runs: 25, Run: manual_2019-08-05T10:14:16.417423+00:00, Layout: Left->Right, Go, and a search bar.

Status indicators at the bottom include: success, running, failed (red), skipped, up_for_reschedule, up_for_retry, queued, and no_status.

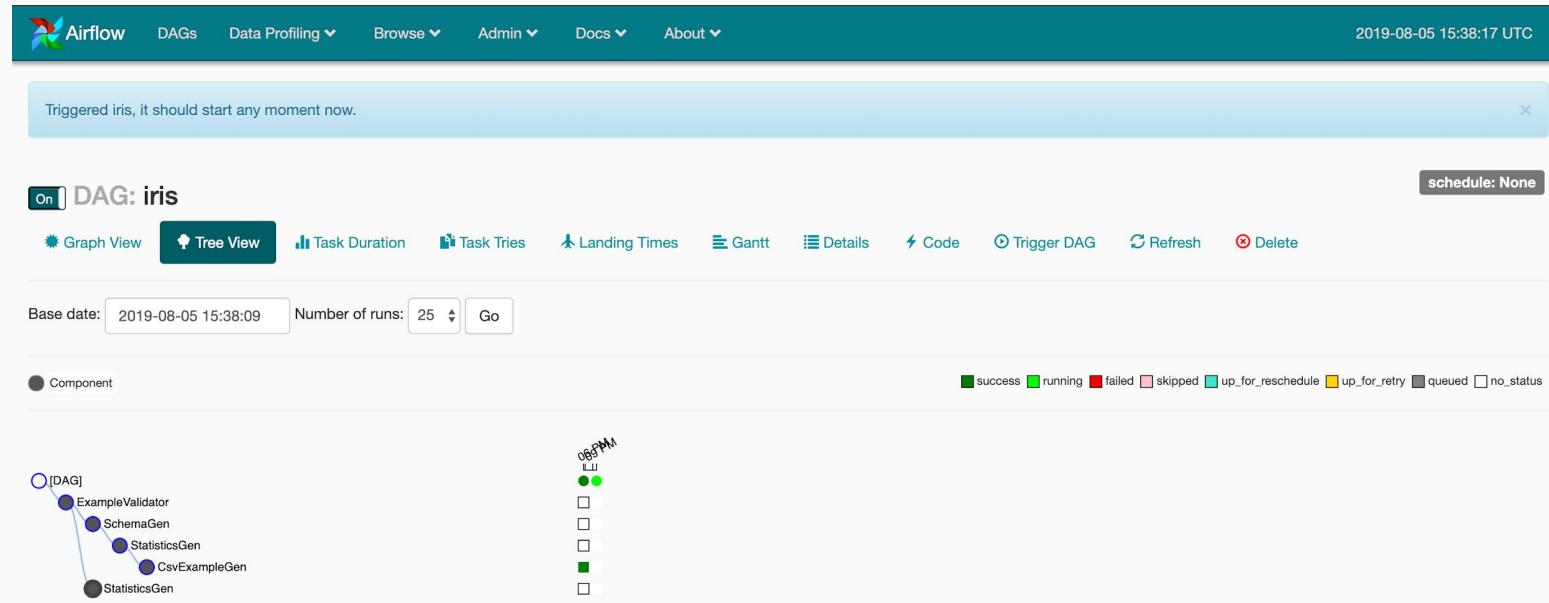
The DAG visualization shows the following tasks and their dependencies:

```
graph LR; CsvExampleGen --> StatisticsGen; StatisticsGen --> SchemaGen; SchemaGen --> ExampleValidator;
```

A red box highlights the "Trigger DAG" button, with the text "Trigger DAG to run tasks" overlaid in red.

The bottom right corner of the interface has a page number 123.

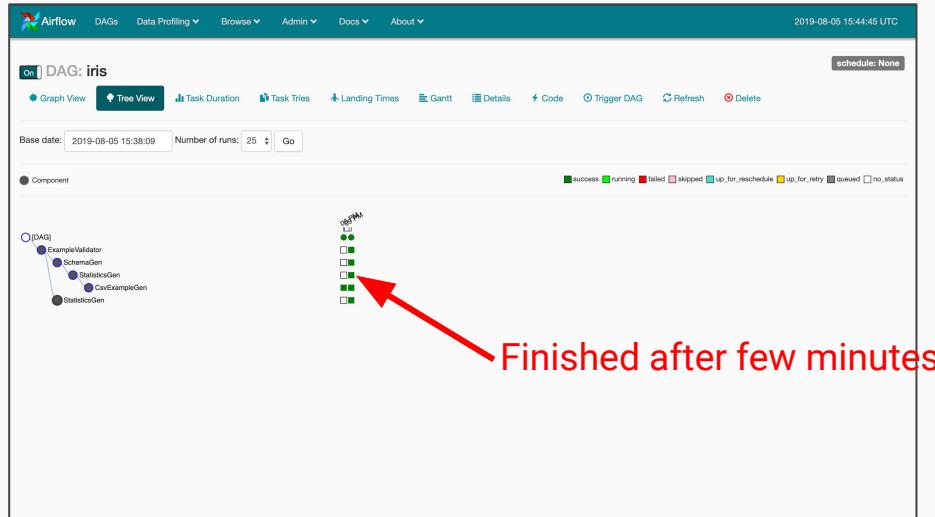
Statistical Schema and Data Validation



Statistical Schema and Data Validation

You just run 3 components

- StatisticsGen: to create statistics summary of iris dataset
- SchemaGen: to construct a schema of iris dataset
- ExampleValidation: to check statistics and schema is valid



View the statistical graph

Files Running Clusters

Select items to perform actions on them.

Upload New

	Name	Last Modified	File size
<input type="checkbox"/> 0	..	seconds ago	
<input type="checkbox"/> model_validation.ipynb		21 hours ago	11.5 kB
<input type="checkbox"/> statistics_and_schema.ipynb		21 hours ago	11.6 kB
<input type="checkbox"/> TFX_Files.ipynb		Running 14 minutes ago	21.3 kB
<input type="checkbox"/> training.ipynb		21 hours ago	2.91 kB
<input type="checkbox"/> transform.ipynb		21 hours ago	16.5 kB
<input type="checkbox"/> airflow_url.txt		21 hours ago	0 B
<input type="checkbox"/> tfx_utils.py		21 hours ago	7.89 kB
<input type="checkbox"/> utils.py		21 hours ago	19.3 kB

 Swap to jupyter. Open statistics_and_schema.ipynb

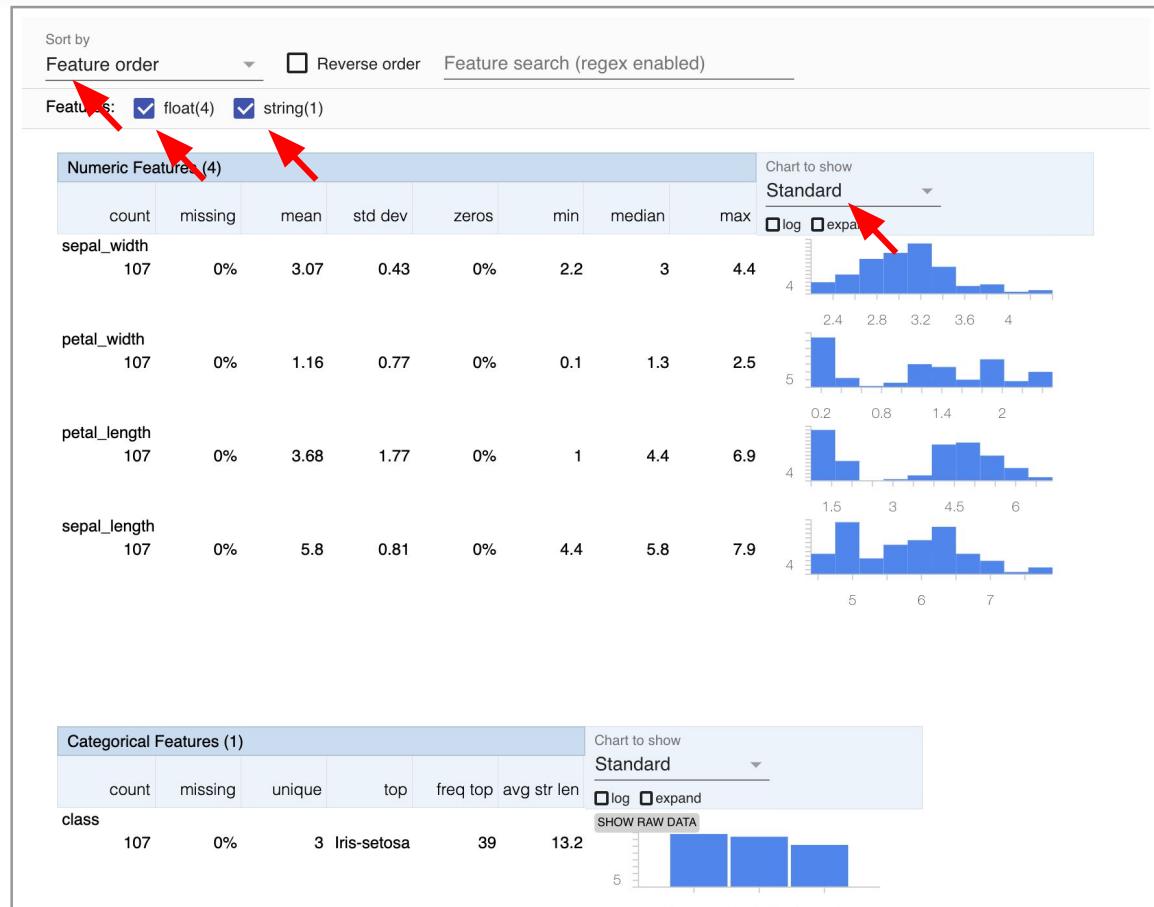
View the statistical graph

```
In [1]: 1 from __future__ import print_function  
2  
3 import os  
4 import tfx_utils  
5 %matplotlib notebook  
6  
7 def _make_default_sqlite_uri():  
8     return os.path.join(os.environ['AIRFLOW_HOME'], 'tfx/metadata/iris/metadata.db')  
9  
10 def get_metadata_store():  
11     return tfx_utils.TFX ReadonlyMetadataStore.from_sqlite_db(_make_default_sqlite_uri())  
12  
13 pipeline_db_path = _make_default_sqlite_uri()  
14  
15 store = get_metadata_store()  
16  
17 store.display_stats_for_examples(2)
```

Run the chunk
It will show you statistical
graph of iris dataset



View the statistical graph



Transform the data

```
73
74     # Performs transformations and feature engineering in training and serving.
75     # transform = Transform( # Step 2
76     #     input_data=example_gen.outputs.examples, # Step 2
77     #     schema=infer_schema.outputs.output, # Step 2
78     #     module_file=_iris_module_file) # Step 2
79
```

uncomment

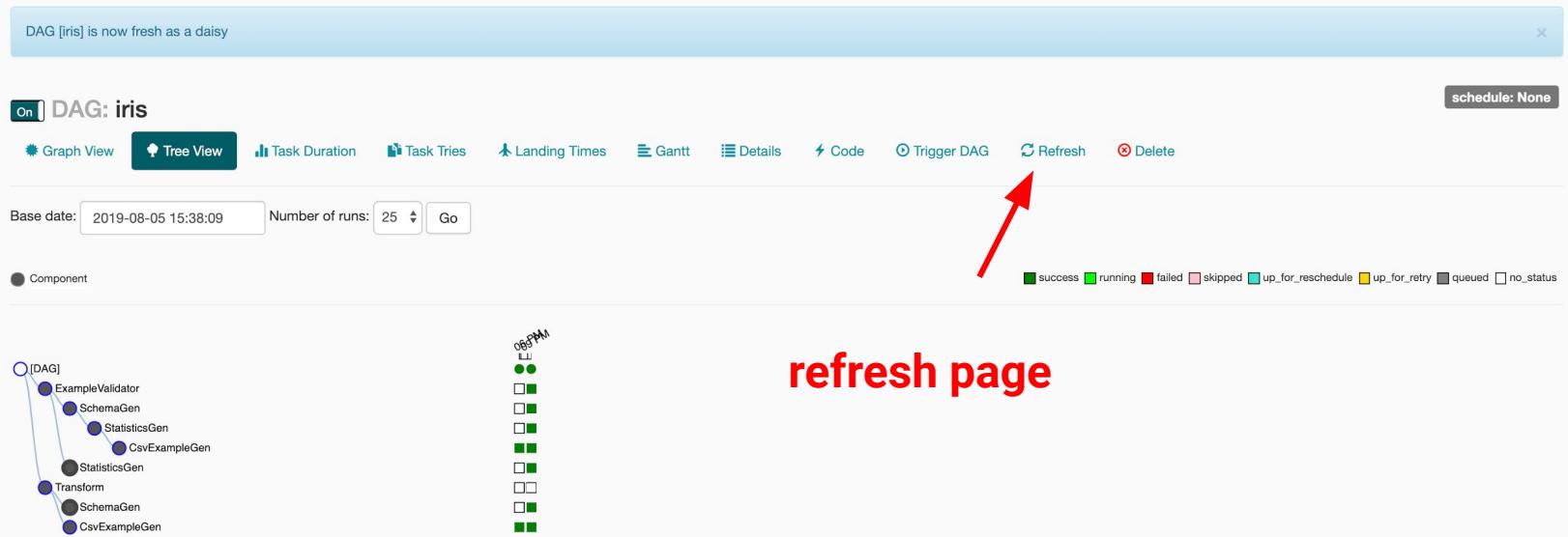
```
73
74     # Performs transformations and feature engineering in training and serving.
75     transform = Transform( # Step 2
76         input_data=example_gen.outputs.examples, # Step 2
77         schema=infer_schema.outputs.output, # Step 2
78         module_file=_iris_module_file) # Step 2
79
```

```
111
112     return pipeline.Pipeline(
113         pipeline_name='iris',
114         pipeline_root=_pipeline_root,
115         components=[
116             example_gen,
117             statistics_gen, infer_schema, validate_stats, # Step 1
118             # transform, # Step 2
119             # trainer, # Step 3
120             # model_analyzer, model_validator, # Step 4
121             # pusher # Step 5
122         ],
123         enable_cache=True,
124         metadata_db_root=_metadata_db_root,
125         additional_pipeline_args={'logger_args': logger_overrides},
126     )
127
```

uncomment

```
111
112     return pipeline.Pipeline(
113         pipeline_name='iris',
114         pipeline_root=_pipeline_root,
115         components=[
116             example_gen,
117             statistics_gen, infer_schema, validate_stats, # Step 1
118             # transform, # Step 2
119             # trainer, # Step 3
120             # model_analyzer, model_validator, # Step 4
121             # pusher # Step 5
122         ],
123         enable_cache=True,
124         metadata_db_root=_metadata_db_root,
125         additional_pipeline_args={'logger_args': logger_overrides},
126     )
127
```

Transform the data



Transform the data

Triggered iris, it should start any moment now.

DAG: iris

Graph View Tree View Task Duration Task Tries Landing Times Gantt

Base date: 2019-08-05 15:50:20 Number of runs: 25 Go

Component

```
graph TD; DAG([DAG]) --> ExampleValidator; ExampleValidator --> SchemaGen1[SchemaGen]; SchemaGen1 --> StatisticsGen1[StatisticsGen]; StatisticsGen1 --> CsvExampleGen1[CsvExampleGen]; StatisticsGen1 --> Transform1[Transform]; Transform1 --> SchemaGen2[SchemaGen]; SchemaGen2 --> CsvExampleGen2[CsvExampleGen];
```

DAG [iris] is now fresh as a daisy

DAG: iris

Graph View Tree View Task Duration Task Tries Landing Times Gantt

Base date: 2019-08-05 15:50:20 Number of runs: 25 Go

Component

```
graph TD; DAG([DAG]) --> ExampleValidator; ExampleValidator --> SchemaGen1[SchemaGen]; SchemaGen1 --> StatisticsGen1[StatisticsGen]; StatisticsGen1 --> CsvExampleGen1[CsvExampleGen]; StatisticsGen1 --> Transform1[Transform]; Transform1 --> SchemaGen2[SchemaGen]; SchemaGen2 --> CsvExampleGen2[CsvExampleGen];
```

Run until finish

Call

```
67
68 def preprocessing_fn(inputs):
69
70     outputs = {}
71
72     # Change input to z-score
73     for key in _DENSE_FLOAT_FEATURE_KEYS:
74         outputs[_transformed_name(key)] = tft.scale_to_z_score(
75             _fill_in_missing(inputs[key]))
76
77     outputs[_transformed_name(_LABEL_KEY)] = tft.compute_and_apply_vocabulary(
78         _fill_in_missing(inputs[_LABEL_KEY]), # Sparse to dense
79         top_k=3,
80         num_oov_buckets=0)
81
82     return outputs
83
```

Test a transform function

Files Running Clusters

Select items to perform actions on them.

Upload New

	Name	Last Modified	File size	
<input type="checkbox"/> 0	..	seconds ago		
<input type="checkbox"/>	model_validation.ipynb	21 hours ago	11.5 kB	
<input type="checkbox"/>	statistics_and_schema.ipynb	Running	2 minutes ago	12.1 kB
<input type="checkbox"/>	TFX_Files.ipynb	Running	a minute ago	21.3 kB
<input type="checkbox"/>	training.ipynb		21 hours ago	2.91 kB
<input type="checkbox"/>	transform.ipynb		21 hours ago	16.5 kB
<input type="checkbox"/>	airflow_url.txt		21 hours ago	0 B
<input type="checkbox"/>	tfx_utils.py		21 hours ago	7.89 kB
<input type="checkbox"/>	utils.py		21 hours ago	19.3 kB



Test a transform function

Run the chunk

```
In [1]: 1 from __future__ import print_function
2
3 import os
4 import tempfile
5 import pandas as pd
6
7 import tensorflow as tf
8 import tensorflow_transform as tft
9 from tensorflow_transform import beam as tft_beam
10 import tfx_utils
11 from tfx.utils import io_utils
12 from tensorflow_metadata.proto.v0 import schema_pb2
13
14 # For DatasetMetadata boilerplate
15 from tensorflow_transform.tf_metadata import dataset_metadata
16 from tensorflow_transform.tf_metadata import dataset_schema
17 from tensorflow_transform.tf_metadata import schema_utils
18
19 # Function for call sqlite database (tfx database)
20 def _make_default_sqlite_uri(pipeline_name):
21     return os.path.join(os.environ['HOME'], 'airflow/tfx/metadata', pipeline_name, 'metadata.db')
22
23 # Function for get metadata
24 def get_metadata_store(pipeline_name):
25     return tfx_utils.TFX ReadonlyMetadataStore.from_sqlite_db(_make_default_sqlite_uri(pipeline_name))
26
27 pipeline_name = 'iris'
28
29 # Get tfx (iris) database
30 pipeline_db_path = _make_default_sqlite_uri(pipeline_name)
31 print('Pipeline DB:\n{}'.format(pipeline_db_path))
32
33 store = get_metadata_store(pipeline_name)
34
35 _np_quint8 = np.dtype([('quint8', np.uint8, 1)])
/home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:528: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
36 _np_qint16 = np.dtype([('qint16', np.int16, 1)])
/home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:529: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
37 _np_quint16 = np.dtype([('quint16', np.uint16, 1)])
/home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:530: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
```

Test a transform function

Run the chunk

```
In [4]: 1 # Get the schema URI from the metadata store
2 schemas = store.get_artifacts_of_type_df(tfx_utils.TFXArtifactTypes.SCHEMA)
3 assert len(schemas.URI) == 1
4 schema_uri = schemas.URI.iloc[0] + 'schema.pbtxt'
5 print ('Schema URI:\n{}'.format(schema_uri))
```

```
Schema URI:
/home/ubuntu/mlrs_ai/airflow/tfx/pipelines/iris/SchemaGen/output/3/schema.pbtxt
```

```
In [5]: 1 # Parse schema and metadata
2 schema_proto = io_utils.parse_pbtxt_file(file_name=schema_uri, message=schema_pb2.Schema())
3 feature_spec, domains = schema_utils.schema_as_feature_spec(schema_proto)
4 legacy_metadata = dataset_metadata.DatasetMetadata(dataset_schema.from_feature_spec(feature_spec, domains))
```

```
In [6]: 1 _DENSE_FLOAT_FEATURE_KEYS = ["sepal_length", "sepal_width", "petal_length", "petal_width"]
2
3 _LABEL_KEY = 'class'
4
5 # Step 2 START -----
6
7 def _transformed_name(key):
8     return key + '_xf'
9
10 def preprocessing_fn(inputs):
11
12     outputs = {}
13     for key in _DENSE_FLOAT_FEATURE_KEYS:
14         outputs[_transformed_name(key)] = tft.scale_to_z_score(
15             inputs[key])
16
17     #     print(inputs[_LABEL_KEY])
18     #     tf.ones_like(inputs[_LABEL_KEY])
19     # #
20     # #         outputs[_transformed_name(_LABEL_KEY)] = tf.ones_like(inputs[_LABEL_KEY]) + \
21     # #                                         tf.cast(tf.equal(inputs[_LABEL_KEY], "Iris-virginica"),
22     # #                                         2 * tf.cast(tf.equal(inputs[_LABEL_KEY], "Iris-versico.
23
24     outputs[_transformed_name(_LABEL_KEY)] = tft.compute_and_apply_vocabulary(
25         inputs[_LABEL_KEY],
26         top_k=3,
27         num_oov_buckets=0)
28
29     #     print(outputs[_transformed_name(_LABEL_KEY)])
30     return outputs
```

Test a transform function

Run the chunk

```
In [7]: 1 # Test transform function
2
3 from IPython.display import display
4 with tft_beam.Context(temp_dir=tempfile.mkdtemp()):
5     raw_examples = [
6         {
7             "sepal_length": [1.],
8             "sepal_width": [1.],
9             "petal_length": [.5],
10            "petal_width": [0.],
11            "class": ["Iris-setosa"],
12        },
13        {
14            "sepal_length": [5.],
15            "sepal_width": [1.],
16            "petal_length": [0.1],
17            "petal_width": [.5],
18            "class": ["Iris-virginica"],
19        },
20        {
21            "sepal_length": [2.],
22            "sepal_width": [.2],
23            "petal_length": [.7],
24            "petal_width": [.3],
25            "class": ["Iris-versicolor"],
26        },
27    ]
28 (transformed_examples, transformed_metadata), transform_fn = (
29     (raw_examples, legacy_metadata)
30     | 'AnalyzeAndTransform' >> tft_beam.AnalyzeAndTransformDataset(
31         preprocessing_fn))
32 display(pd.DataFrame(transformed_examples))
```

```
WARNING:tensorflow:Expected binary or unicode string, got type_url: "type.googleapis.com/tensorflow.AssetFileDef"
value: "\n\\013\\n\\tConst_8:0\\022-vocab_compute_and_apply_vocabulary_vocabulary"
```

```
W0805 15:59:45.275808 140419297957696 ops.py:6153] Expected binary or unicode string, got type_url: "type.googleapis.com/tensorflow.AssetFileDef"
value: "\n\\013\\n\\tConst_8:0\\022-vocab_compute_and_apply_vocabulary_vocabulary"
```

```
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
```

```
I0805 15:59:45.277959 140419297957696 saver.py:1483] Saver not created because there are no variables in the graph to
restore
```

Output of
transform
function

	class_xf	petal_length_xf	petal_width_xf	sepal_length_xf	sepal_width_xf
0	[2]	[0.33968312]	[-1.297715]	[-0.9805807]	[0.7071068]
1	[0]	[-1.3587326]	[1.1355498]	[1.3728129]	[0.7071068]
2	[1]	[1.0190493]	[0.1622214]	[-0.3922323]	[-1.4142138]

Train the model

```
111
112     return pipeline.Pipeline(
113         pipeline_name='iris',
114         pipeline_root=_pipeline_root,
115         components=[
116             example_gen,
117             statistics_gen, infer_schema, validate_stats, # Step 1
118             transform, # Step 2
119             # trainer, # Step 3
120             # model_analyzer, model_validator, # Step 4
121             # pusher # Step 5
122         ],
123         enable_cache=True,
124         metadata_db_root=_metadata_db_root,
125         additional_pipeline_args={'logger_args': logger_overrides},
126     )
127
128 }
```

uncomment

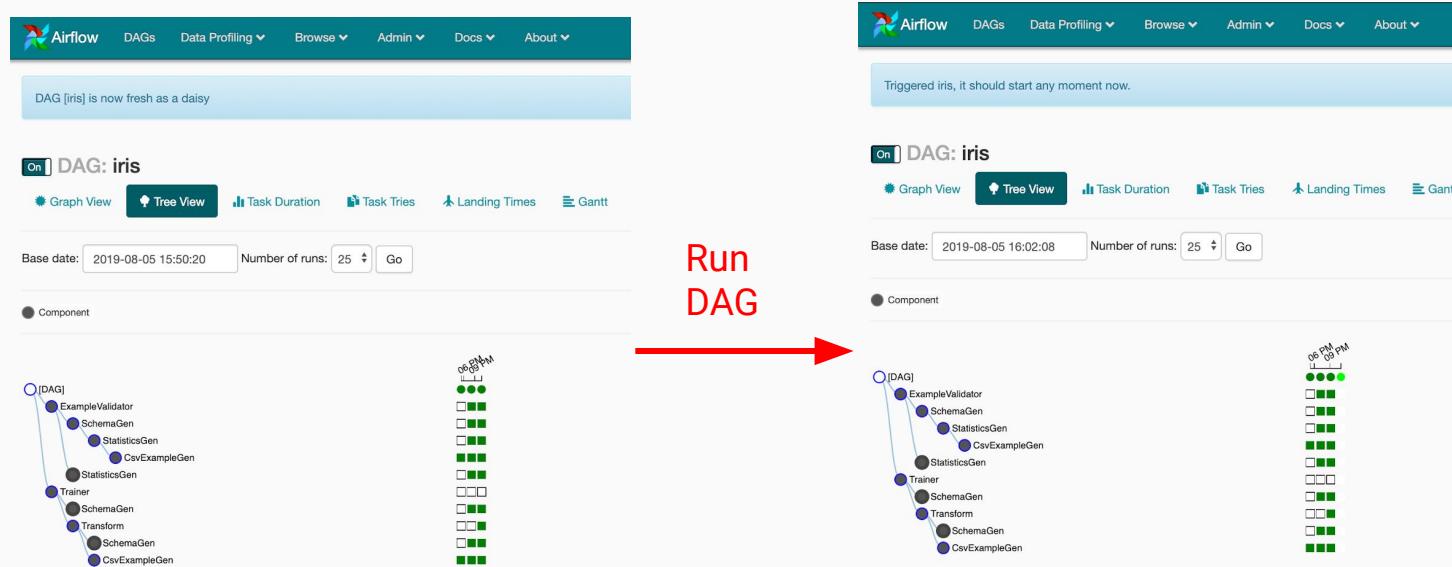
```
79
80     # Uses user-provided Python function that implements a model using TF-Learn.
81     # trainer = Trainer( # Step 3
82     #     module_file=_iris_module_file, # Step 3
83     #     transformed_examples=transform.outputs.transformed_examples, # Step 3
84     #     schema=infer_schema.outputs.output, # Step 3
85     #     transform_output=transform.outputs.transform_output, # Step 3
86     #     train_args=trainer_pb2.TrainArgs(num_steps=10000), # Step 3
87     #     eval_args=trainer_pb2.EvalArgs(num_steps=5000)) # Step 3
88
```

uncomment

```
111
112     return pipeline.Pipeline(
113         pipeline_name='iris',
114         pipeline_root=_pipeline_root,
115         components=[
116             example_gen,
117             statistics_gen, infer_schema, validate_stats, # Step 1
118             transform, # Step 2
119             # trainer, # Step 3
120             # model_analyzer, model_validator, # Step 4
121             # pusher # Step 5
122         ],
123         enable_cache=True,
124         metadata_db_root=_metadata_db_root,
125         additional_pipeline_args={'logger_args': logger_overrides},
126     )
127
```

```
79
80     # Uses user-provided Python function that implements a model using TF-Learn.
81     # trainer = Trainer( # Step 3
82     #     module_file=_iris_module_file, # Step 3
83     #     transformed_examples=transform.outputs.transformed_examples, # Step 3
84     #     schema=infer_schema.outputs.output, # Step 3
85     #     transform_output=transform.outputs.transform_output, # Step 3
86     #     train_args=trainer_pb2.TrainArgs(num_steps=10000), # Step 3
87     #     eval_args=trainer_pb2.EvalArgs(num_steps=5000)) # Step 3
88
```

Train the model



Tracking model performances

Files Running Clusters

Select items to perform actions on them.

Upload New

	Name	Last Modified	File size	
<input type="checkbox"/> 0	..	seconds ago		
<input type="checkbox"/>	model_validation.ipynb	21 hours ago	11.5 kB	
<input type="checkbox"/>	statistics_and_schema.ipynb	Running	2 minutes ago	12.1 kB
<input type="checkbox"/>	TFX_Files.ipynb	Running	a minute ago	21.3 kB
<input type="checkbox"/>	training.ipynb	21 hours ago	2.91 kB	
<input type="checkbox"/>	transform.ipynb	21 hours ago	16.5 kB	
<input type="checkbox"/>	airflow_url.txt	21 hours ago	0 B	
<input type="checkbox"/>	tfx_utils.py	21 hours ago	7.89 kB	
<input type="checkbox"/>	utils.py	21 hours ago	19.3 kB	



Tracking model performances “Tensorboard”

Run the
chunks to
run
tensorboard

It will run
on port
6006

The screenshot shows a Jupyter Notebook interface with the title "jupyter training" and status "Last Checkpoint: 21 hours ago (unsaved changes)". The top menu includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Not Trusted, Python 3, and Logout. Below the menu is a toolbar with icons for file operations and cell execution.

In [1]:

```
from __future__ import print_function
import os
import webbrowser
tensorboard_logdir = os.path.join(os.environ['HOME'], 'airflow/tfx/pipelines/iris/Trainer/output')
print('tensorboard_logdir: {}'.format(tensorboard_logdir))
os.environ['TENSORBOARD_LOGDIR'] = tensorboard_logdir
```

tensorboard_logdir: /home/ubuntu/mlrs_ai/airflow/tfx/pipelines/iris/Trainer/output

In [*]:

```
!tensorboard --logdir="${TENSORBOARD_LOGDIR}"
```

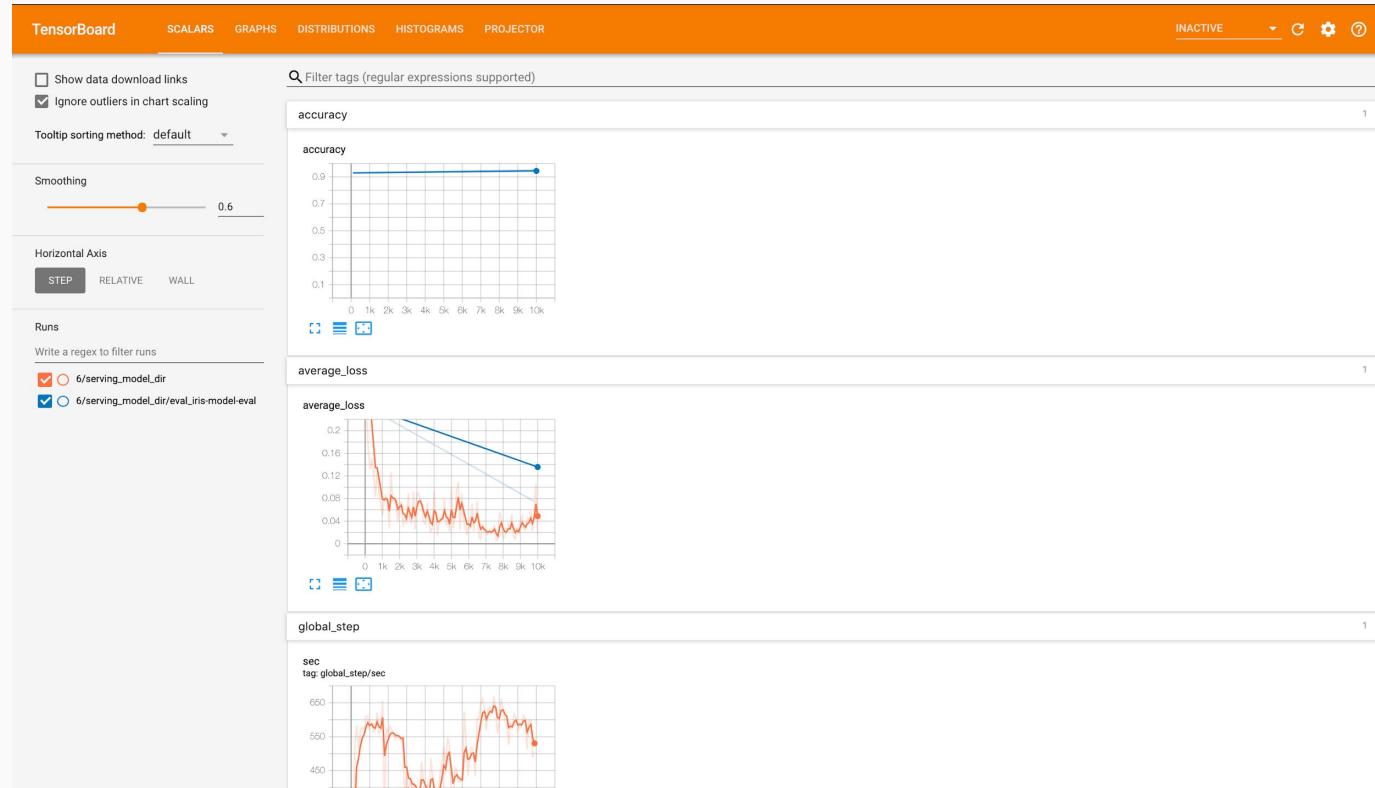
Output log:

```
I0805 16:11:21.169603 140304669726464 _internal.py:122] ::ffff:49.49.235.80 - - [05/Aug/2019 16:11:21] "GET /tf-interactive-inference-dashboard/editedexample.png HTTP/1.1" 200 -
I0805 16:11:21.172351 140304661333760 _internal.py:122] ::ffff:49.49.235.80 - - [05/Aug/2019 16:11:21] "GET /tf-interactive-inference-dashboard/distance.png HTTP/1.1" 200 -
I0805 16:11:21.233359 140304652941056 _internal.py:122] ::ffff:49.49.235.80 - - [05/Aug/2019 16:11:21] "GET /tf-interactive-inference-dashboard/explorecounterfactuals.png HTTP/1.1" 200 -
I0805 16:11:21.234569 140304434853632 _internal.py:122] ::ffff:49.49.235.80 - - [05/Aug/2019 16:11:21] "GET /tf-interactive-inference-dashboard/pdplots.png HTTP/1.1" 200 -
I0805 16:11:21.347387 140304669726464 _internal.py:122] ::ffff:49.49.235.80 - - [05/Aug/2019 16:11:21] "GET /data/plugins/listing HTTP/1.1" 200 -
WARNING:tensorflow:From /home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/tensorboard/plugins/projector/projector_plugin.py:410: checkpoint_exists (from tensorflow.python.training.checkpoint_management) is deprecated and will be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
W0805 16:11:21.350814 140304426460928 deprecation.py:323] From /home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/tensorboard/plugins/projector/projector_plugin.py:410: checkpoint_exists (from tensorflow.python.training.checkpoint_management) is deprecated and will be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
```

In []: 1

Tracking model performances “Tensorboard”

Go to URL: http://{{Your_EC2_IPAddr}}:6006



Tracking model performances “Tensorboard”



Validate the model

```
112     return pipeline.Pipeline(  
113         pipeline_name='iris',  
114         pipeline_root=_pipeline_root,  
115         components=[  
116             example_gen,  
117             statistics_gen, infer_schema, validate_stats, # Step 1  
118             transform, # Step 2  
119             trainer, # Step 3  
120             model_analyzer, model_validator, # Step 4  
121             pusher # Step 5  
122         ],  
123         enable_cache=True,  
124         metadata_db_root=_metadata_db_root,  
125         additional_pipeline_args={'logger_args': logger_overrides},  
126     )  
127 
```

uncomment

```
88  
89     # Uses TFMA to compute a evaluation statistics over features of a model.  
90     # model_analyzer = Evaluator( # Step 4  
91     #     examples=example_gen.outputs.examples, # Step 4  
92     #     model_exports=trainer.outputs.output, # Step 4  
93     #     feature_slicing_spec=evaluator_pb2.FeatureSlicingSpec(specs=[ # Step 4  
94     #         evaluator_pb2.SingleSlicingSpec( # Step 4  
95     #             column_for_slicing=['sepal_length']) # Step 4  
96     #     ]) # Step 4  
97  
98     # Performs quality validation of a candidate model (compared to a baseline).  
99     # model_validator = ModelValidator( # Step 4  
100    #     examples=example_gen.outputs.examples, # Step 4  
101    #         model=trainer.outputs.output) # Step 4  
102 
```

uncomment

```
112     return pipeline.Pipeline(  
113         pipeline_name='iris',  
114         pipeline_root=_pipeline_root,  
115         components=[  
116             example_gen,  
117             statistics_gen, infer_schema, validate_stats, # Step 1  
118             transform, # Step 2  
119             trainer, # Step 3  
120             model_analyzer, model_validator, # Step 4  
121             pusher # Step 5  
122         ],  
123         enable_cache=True,  
124         metadata_db_root=_metadata_db_root,  
125         additional_pipeline_args={'logger_args': logger_overrides},  
126     )  
127 
```

```
88  
89     # Uses TFMA to compute a evaluation statistics over features of a model.  
90     # model_analyzer = Evaluator( # Step 4  
91     #     examples=example_gen.outputs.examples, # Step 4  
92     #     model_exports=trainer.outputs.output, # Step 4  
93     #     feature_slicing_spec=evaluator_pb2.FeatureSlicingSpec(specs=[ # Step 4  
94     #         evaluator_pb2.SingleSlicingSpec( # Step 4  
95     #             column_for_slicing=['sepal_length']) # Step 4  
96     #     ]) # Step 4  
97  
98     # Performs quality validation of a candidate model (compared to a baseline).  
99     # model_validator = ModelValidator( # Step 4  
100    #     examples=example_gen.outputs.examples, # Step 4  
101    #         model=trainer.outputs.output) # Step 4  
102 
```

Validate the model

Triggered iris, it should start any moment now.

DAG: iris

On DAG View Tree View Task Duration Task Tries Landing Times Gantt Details Trigger DAG Refresh Delete schedule: None

Base date: 2019-08-05 16:40:42 Number of runs: 25 Go

Component

Refresh and Trigger airflow again to run pipeline

Legend: success, running, failed, skipped, up_for_reschedule, up_for_retry, queued, no_status

```
graph TD; DAG[DAG] --> ExampleValidator[ExampleValidator]; ExampleValidator --> SchemaGen1[SchemaGen]; SchemaGen1 --> StatisticsGen1[StatisticsGen]; StatisticsGen1 --> CsvExampleGen1[CsvExampleGen]; CsvExampleGen1 --> Evaluator[Evaluator]; Evaluator --> Trainer1[Trainer]; Trainer1 --> Transform[Transform]; Transform --> SchemaGen2[SchemaGen]; SchemaGen2 --> CsvExampleGen2[CsvExampleGen]; CsvExampleGen2 --> ModelValidator[ModelValidator]; ModelValidator --> Trainer2[Trainer]; Trainer2 --> CsvExampleGen3[CsvExampleGen]
```

Model performance validation “Slicing performance”

Files Running Clusters

Select items to perform actions on them.

Upload New

	Name	Last Modified	File size
0	..	seconds ago	
model_validation.ipynb	a day ago	11.5 kB	
statistics_and_schema.ipynb	Running 22 minutes ago	11.5 kB	
TFX_Files.ipynb	Running a minute ago	21.3 kB	
training.ipynb	Running seconds ago	27.9 kB	
transform.ipynb	Running 14 minutes ago	18.4 kB	
airflow_url.txt	4 minutes ago	0 B	
tfx_utils.py	a day ago	7.89 kB	
utils.py	a day ago	19.3 kB	



Model performance validation “Slicing performance”

```
In [1]: 1 from __future__ import print_function  
2  
3 import os  
4 import tfx_utils  
5 import tensorflow_model_analysis as tfma  
6  
7 def _make_default_sqlite_uri(pipeline_name):  
8     return os.path.join(os.environ['HOME'], 'airflow/tfx/metadata', pipeline_name, 'metadata.db')  
9  
10 def get_metadata_store(pipeline_name):  
11     return tfx_utils.TFXReadonlyMetadataStore.from_sqlite_db(_make_default_sqlite_uri(pipeline_name))  
12  
13 pipeline_name = 'iris' # or taxi_solution  
14 pipeline_db_path = _make_default_sqlite_uri(pipeline_name)  
15 print('Pipeline DB:\n{}'.format(pipeline_db_path))  
16  
17 store = get_metadata_store(pipeline_name)
```

```
/home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/apache_beam/__init__.py:84: UserWarning: Some syntactic constructs of Python 3 are not yet fully supported by Apache Beam.  
    'Some syntactic constructs of Python 3 are not yet fully supported by '  
/home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:526: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
```

Model performance validation “Slicing performance”

Run this

Pipeline DB:

/home/ubuntu/mlrs_ai/airflow/tfx/metadata/iris/metadata.db

In [2]: 1 store.get_artifacts_of_type_df(tfx_utils.TFXArtifactTypes.MODEL)

Out[2]:

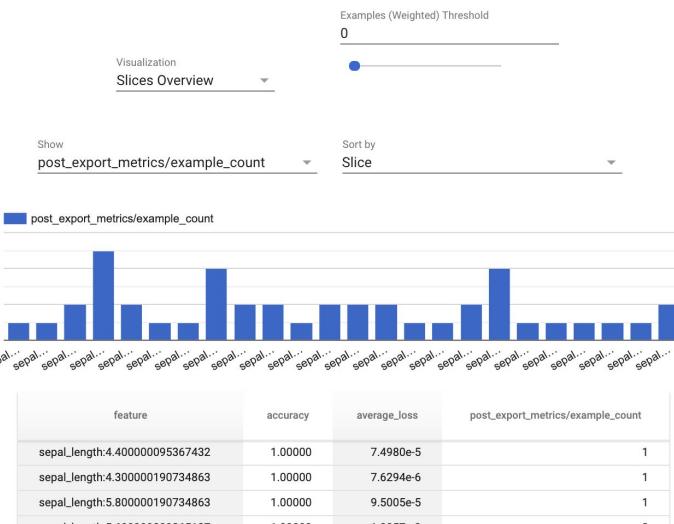
ID	URI	TYPE_NAME	SPLIT	STATE	SPAN
11	/home/ubuntu/mlrs_ai/airflow/tfx/pipelines/iri...	ModelExportPath		published	1

Model performance validation “Slicing performance”

Run this

```
In [4]: 1 store.display_tfma_analysis(11, slicing_column='sepal_length')
```

```
WARNING:tensorflow:From /home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/tensorflow_model_analysis/evaluators/metrics_and_plots_evaluator.py:83: tf_record_iterator (from tensorflow.python.lib.io.tf_record) is deprecated and will be removed in a future version.  
Instructions for updating:  
Use eager execution and:  
`tf.data.TFRecordDataset(path)`  
  
WARNING: Logging before flag parsing goes to stderr.  
W0806 01:23:16.615738 139874994034496 deprecation.py:323] From /home/ubuntu/mlrs_ai/venv/lib/python3.6/site-packages/tensorflow_model_analysis/evaluators/metrics_and_plots_evaluator.py:83: tf_record_iterator (from tensorflow.python.lib.io.tf_record) is deprecated and will be removed in a future version.  
Instructions for updating:  
Use eager execution and:  
`tf.data.TFRecordDataset(path)`
```



Pipeline DB:
/home/ubuntu/mlrs_ai/airflow/tfx/metadata/iris/metadata.db

```
In [2]: 1 store.get_artifacts_of_type_df(tfx_utils.TFXArtifactTypes.MODEL)
```

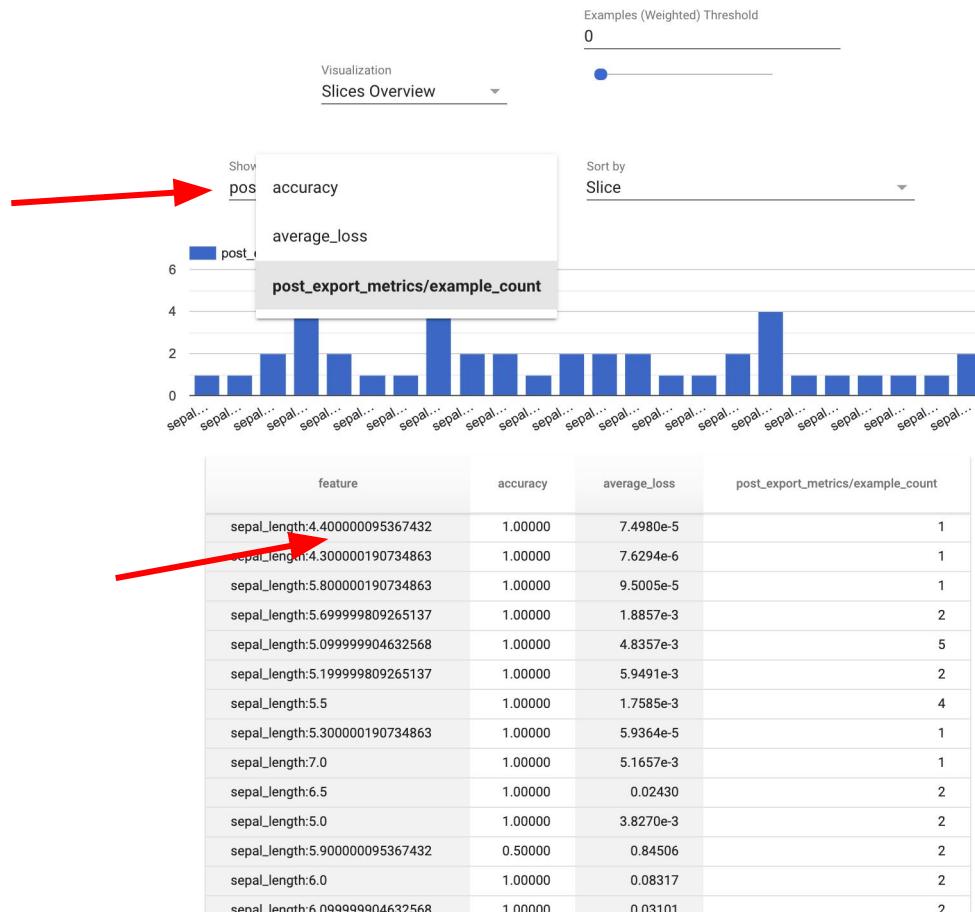
Out[2]:

ID	URI	TYPE_NAME	SPLIT	STATE	SPAN
11	/home/ubuntu/mlrs_ai/airflow/tfx/pipelines/ir...	ModelExportPath		published	1

Copy model ID and paste to
below function

Model performance validation “Slicing performance”

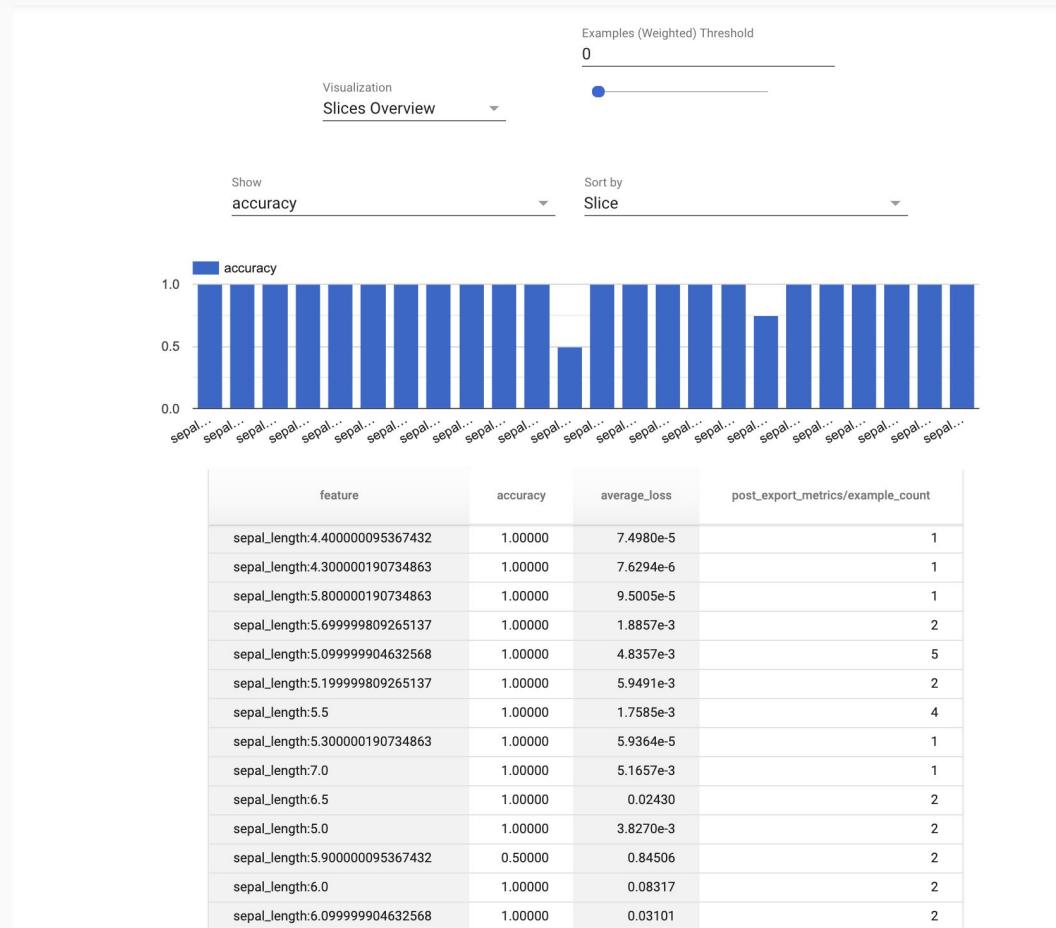
Change mode to display



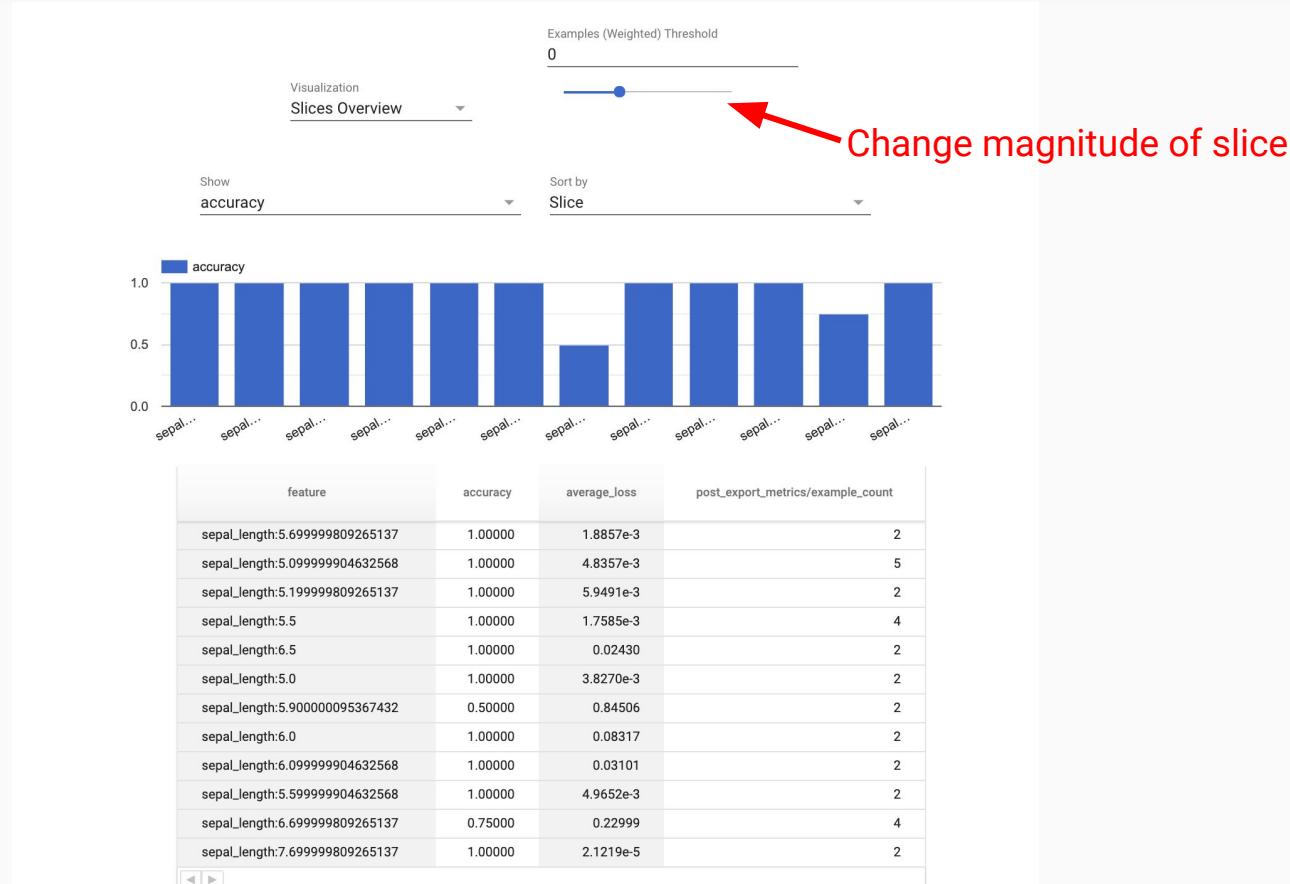
This will show values (accuracy, number of samples in specific range)

Range

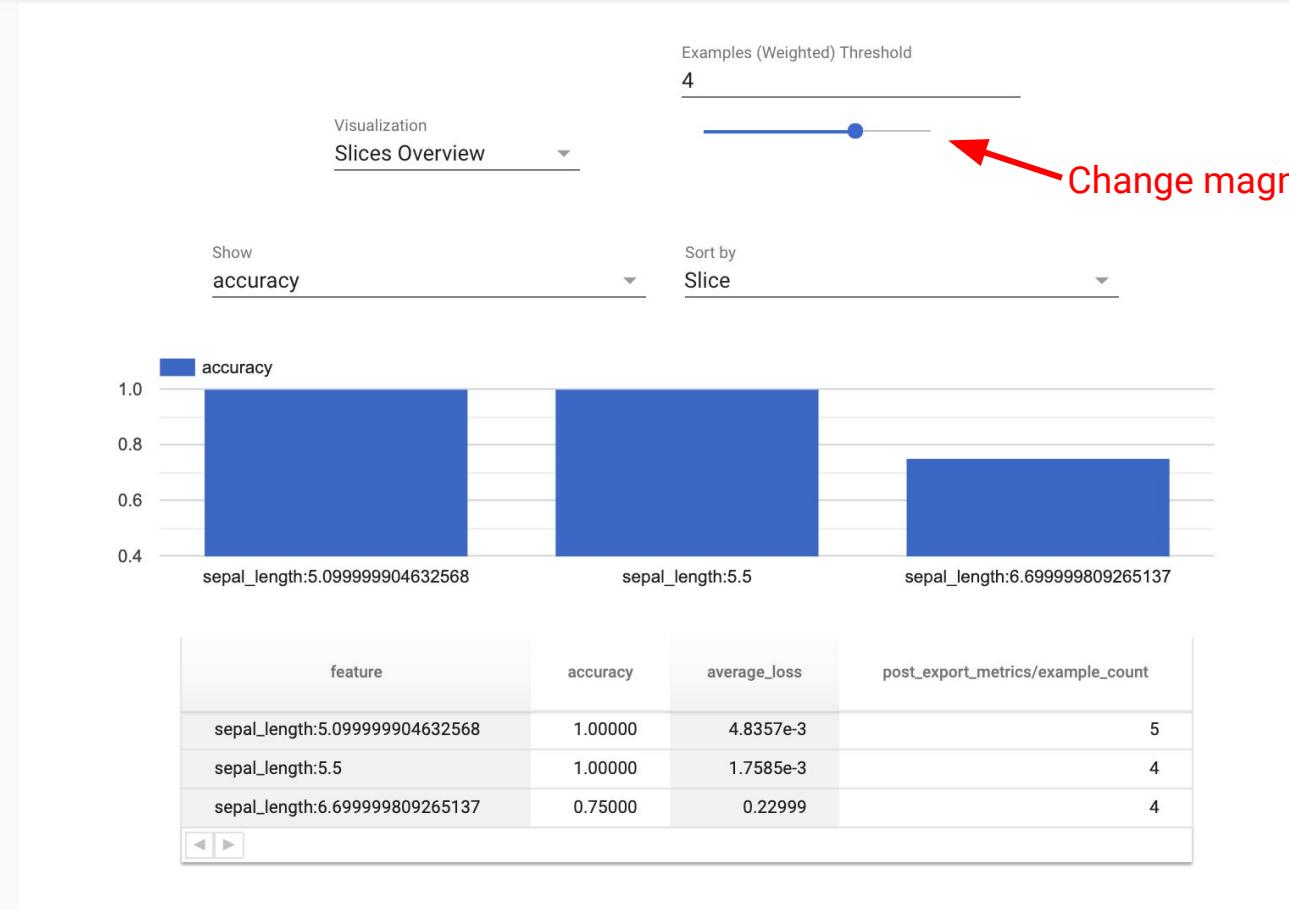
Model performance validation “Slicing performance”



Model performance validation “Slicing performance”



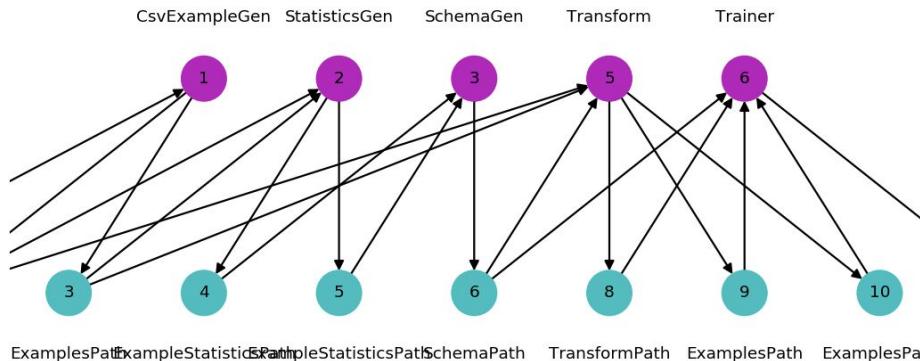
Model performance validation “Slicing performance”



Model performance validation “Pipeline structure”

```
In [5]: 1 # Try different IDs here. Click stop in the plot when changing IDs.  
2 %matplotlib notebook  
3 store.plot_artifact_lineage(11)
```

Figure 1



This will show the structure of your pipeline

Push your model to serve

```
111
112     return pipeline.Pipeline(
113         pipeline_name='iris',
114         pipeline_root=_pipeline_root,
115         components=[
116             example_gen,
117             statistics_gen, infer_schema, validate_stats, # Step 1
118             transform, # Step 2
119             trainer, # Step 3
120             model_analyzer, model_validator, # Step 4
121             pusher # Step 5
122         ],
123         enable_cache=True,
124         metadata_db_root=_metadata_db_root,
125         additional_pipeline_args={'logger_args': logger_overrides},
126     )
127
128
```

uncomment

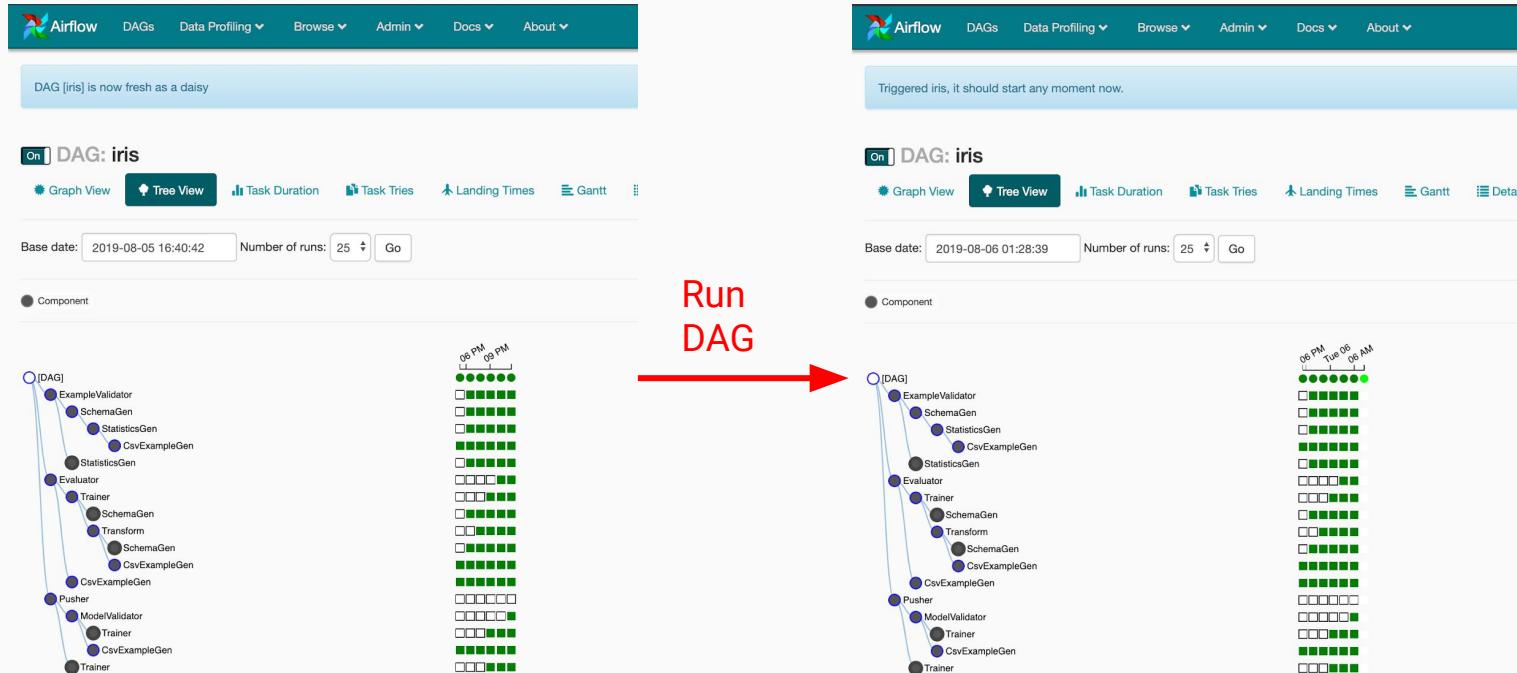
```
111
112     return pipeline.Pipeline(
113         pipeline_name='iris',
114         pipeline_root=_pipeline_root,
115         components=[
116             example_gen,
117             statistics_gen, infer_schema, validate_stats, # Step 1
118             transform, # Step 2
119             trainer, # Step 3
120             model_analyzer, model_validator, # Step 4
121             pusher # Step 5
122         ],
123         enable_cache=True,
124         metadata_db_root=_metadata_db_root,
125         additional_pipeline_args={'logger_args': logger_overrides},
126     )
127
```

```
103
104
105     # Checks whether the model passed the validation steps and pushes the model
106     # to a file destination if check passed.
107     pusher = Pusher( # Step 5
108         model_export=trainer.outputs.output, # Step 5
109         model_blessing=model_validator.outputs.blessing, # Step 5
110         push_destination=pusher_pb2.PushDestination( # Step 5
111             filesystem=pusher_pb2.PushDestination.Filesystem( # Step 5
112                 base_directory=_serving_model_dir)) # Step 5
```

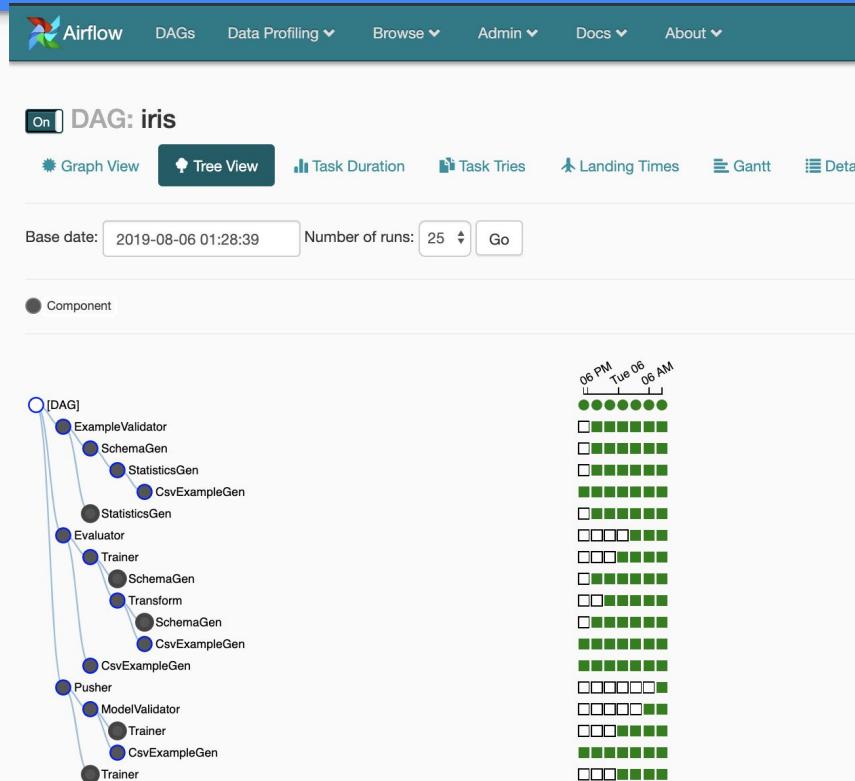
uncomment

```
102
103
104     # Checks whether the model passed the validation steps and pushes the model
105     # to a file destination if check passed.
106     #
107     # pusher = Pusher( # Step 5
108     #     model_export=trainer.outputs.output, # Step 5
109     #     model_blessing=model_validator.outputs.blessing, # Step 5
110     #     push_destination=pusher_pb2.PushDestination( # Step 5
111     #         filesystem=pusher_pb2.PushDestination.Filesystem( # Step 5
112     #             base_directory=_serving_model_dir)) # Step 5
```

Push your model to serve



Push your model to serve



Run until finish

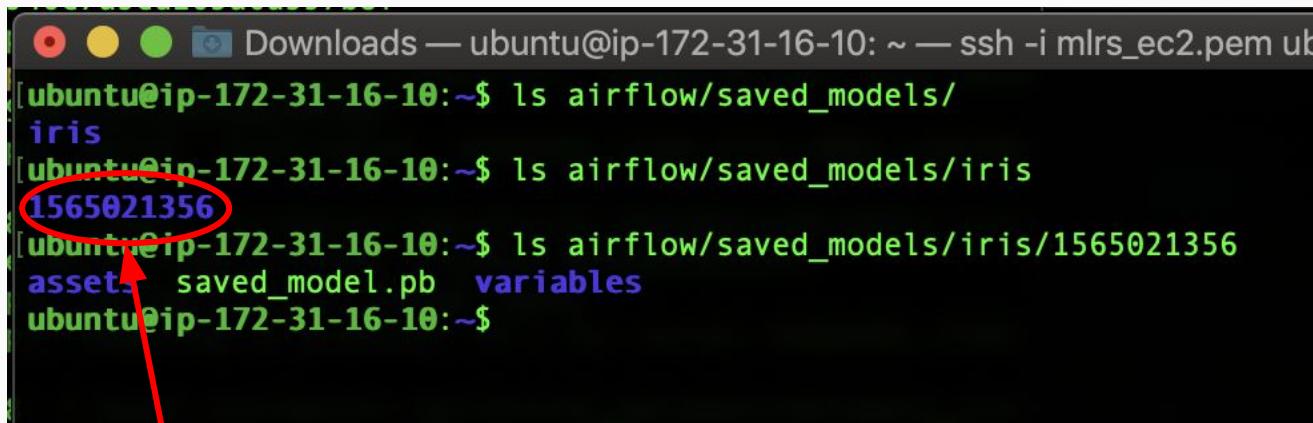
Serving TF model

Saved model path was defined in iris_pipeline.py

```
37 _airflow_root = os.path.join(os.environ['AIRFLOW_HOME']) # Set airflow home
38 _data_root = os.path.join(_airflow_root, 'data/iris') # set data path
39
40 _iris_module_file = os.path.join(_airflow_root, 'dags/iris_utils.py') # set module file (utils)
41 _serving_model_dir = os.path.join(_airflow_root, 'saved_models/iris') # set where to serve model
42
43 _tfx_root = os.path.join(_airflow_root, 'tfx') # set where is tfx home
44 _pipeline_root = os.path.join(_tfx_root, 'pipelines') # set where pipelines metadata will be saved
45 _metadata_db_root = os.path.join(_tfx_root, 'metadata') # set where tfx metadata will be saved
46 _log_root = os.path.join(_tfx_root, 'logs') # set where is logs path
47
```

After pusher is finished

You can list the saved model in terminal



A terminal window showing the output of the command `ls airflow/saved_models/iris`. The output lists a single directory named `1565021356`, which is circled in red. A red arrow points from the text "Model ID (version)" at the bottom left to this circled directory name.

```
Downloads — ubuntu@ip-172-31-16-10: ~ — ssh -i mlrs_ec2.pem ub
[ubuntu@ip-172-31-16-10:~$ ls airflow/saved_models/
iris
[ubuntu@ip-172-31-16-10:~$ ls airflow/saved_models/iris
1565021356
[ubuntu@ip-172-31-16-10:~$ ls airflow/saved_models/iris/1565021356
assets  saved_model.pb  variables
ubuntu@ip-172-31-16-10:~$
```

Model ID (version)

Serving TF model

Files Running Clusters

Select items to perform actions on them.

Upload New

	Name	Last Modified	File size
0	..	seconds ago	
<input type="checkbox"/>	model_serving.ipynb	seconds ago	4.22 kB
<input type="checkbox"/>	model_validation.ipynb	Running 22 minutes ago	180 kB
<input type="checkbox"/>	statistics_and_schema.ipynb	Running 41 minutes ago	8.07 kB
<input type="checkbox"/>	TFX_Files.ipynb	Running 20 minutes ago	21.3 kB
<input type="checkbox"/>	training.ipynb	Running 9 hours ago	95.6 kB
<input type="checkbox"/>	transform.ipynb	Running 10 hours ago	18.4 kB
<input type="checkbox"/>	airflow_url.txt	10 hours ago	0 B
<input type="checkbox"/>	tfx_utils.py	an hour ago	7.88 kB
<input type="checkbox"/>	utils.py	an hour ago	19.3 kB



```
In [5]: 1 # Your model is saved here  
2 !ls ..../airflow/saved_models/iris
```

You also can list the saved model in jupyter

1565021356

```
In [23]: 1 # If model error  
2 !pkill tensorflow*
```

If you want to run new tf server, kill it first

```
In [25]: 1 %%bash --bg  
2 nohup tensorflow_model_server \  
3 --rest_api_port=8501 \  
4 --model_name=iris \  
5 --model_base_path=/home/ubuntu/mlrs_ai/airflow/saved_models/iris > server.log 2>&1
```

Run TF server on port 8501

```
In [26]: 1 # View the tensorflow server log  
2 !tail server.log
```

List the logs of TF server

```
2019-08-06 02:11:01.988070: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:54] Reading meta graph with  
tags { serve }  
2019-08-06 02:11:01.990459: I external/org_tensorflow/tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU sup  
ports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA  
2019-08-06 02:11:02.011005: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:202] Restoring SavedModel b  
undle.  
2019-08-06 02:11:02.025321: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:151] Running initialization  
op on SavedModel bundle at path: /home/ubuntu/mlrs_ai/airflow/saved_models/iris/1565021356  
2019-08-06 02:11:02.032065: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:311] SavedModel load for ta  
gs { serve }; Status: success. Took 46202 microseconds.  
2019-08-06 02:11:02.032132: I tensorflow_serving/servables/tensorflow/saved_model_warmup.cc:103] No warmup data file  
found at /home/ubuntu/mlrs_ai/airflow/saved_models/iris/1565021356/assets.extra/tf_serving_warmup_requests  
2019-08-06 02:11:02.032269: I tensorflow_serving/core/loader_harness.cc:86] Successfully loaded servable version {nam  
e: iris version: 1565021356}  
2019-08-06 02:11:02.033767: I tensorflow_serving/model_servers/server.cc:324] Running gRPC ModelServer at 0.0.0.0:850  
0 ...  
2019-08-06 02:11:02.034344: I tensorflow_serving/model_servers/server.cc:344] Exporting HTTP/REST API at:localhost:85  
01 ...  
[evhttp_server.cc : 239] RAW: Entering the event loop ...
```

Serving TF model

Testing the API by python requests

```
In [33]: 1 import json, numpy as np, requests  
2 # Testing fetch features to API  
3  
4 # You can changes data of features  
5 feature = {"sepal_length": 3.0, "sepal_width": 4.0, "petal_length": 2.0, "petal_width": 5.0}  
6  
7 data = json.dumps({"examples": [feature]})  
8 headers = {"content-type": "application/json"}  
9 json_response = requests.post('http://localhost:8501/v1/models/iris/versions/{}:classify'.format(< YOUR MODEL ID >)  
10  
11 # Print output as a array  
12 predicted = json.loads(json_response.text)  
13 print(predicted)  
14  
15 # Find maximum probability class  
16 result = np.argmax(np.array(predicted["results"])[0]).astype(float)[:, 1]  
17 print("Class:", result)
```

Test data

Replace this with your model ID

```
{"results": [[[0', 1.67529663e-06], [1', 0.999998331], [2', 2.57071274e-12]]]}  
Class: 1
```

Predicted value

Test API online

Alternatively, you can call the service from your laptop

```
curl -d '{"examples": [{"sepal_length": 3.0, "sepal_width": 4.0, "petal_length": 2.0, "petal_width": 5.0}]}'  
-X POST http://<YOUR EC2 URL>:8501/v1/models/iris/versions/<YOUR MODEL ID>:classify
```

Request form

```
[→ ~ curl -d '{"examples": [{"sepal_length": 3.0, "sepal_width": 4.0, "petal_length": 2.0, "petal_width": 5.0}]}'  
-X POST http://13.250.95.92:8501/v1/models/iris/versions/1565021356:classify
```

```
{  
    "results": [[[["0", 1.67529663e-06], ["1", 0.999998331], ["2", 2.57071274e-12]]  
    ]  
}
```

Prediction response

Other TFX & TF-serving examples

TFX on colab :

https://docs.google.com/presentation/d/1vPBaVFdb9qmmKnc_xNVsRCXRAPv6A7bfOLvY0bs1gl/edit#slide=id.g5e3bc437cc_0_280

TF-serving for tf.estimator :

https://colab.research.google.com/drive/117em0BbGYhxT0pEYgMSq4MC9nrDB3oHN#scrollTo=jonq_oICp35Q

TF-serving for tf.keras :

<https://colab.research.google.com/drive/1jVRoewe-pFLJNoLliBrpSYjRifY0CYdt>

4. Serverless

Clone demo repo at

https://github.com/kwarodom/Serverless_Deep_Learning_MLRS.git

[<<< only lab 1](https://github.com/kwarodom/mlrs_serverless)

**Serverless Deep
Learning Model
Deployment on AWS
Lambda Function**

Agenda

1. Benefits of serverless?
2. What is Serverless?
3. Why Serverless Deep Learning?
4. Where Serverless Deep Learning Works and Where it doesn't work?
5. Intro to AWS Lambda Function
6. Deep Learning Deployment process on AWS Lambda Function
7. Deploying Tensorflow Model with AWS Lambda Function
8. Deploying Tensorflow Model using Serverless Framework
9. Creating Deep Learning API
10. Creating Deep Learning Pipeline
11. Creating Deep Learning Workflow

1.Benefits of Serverless Deep Learning

Main benefits of serverless for business owners



Shorter time to market



Quick deployment



Event-driven project scaling



Pay only for what you use



Reduced costs of running a server

Benefits of serverless FaaS approach for developers



Great scalability and flexibility of the backend



Less effort for maintaining project infrastructure



Easier to introduce new features

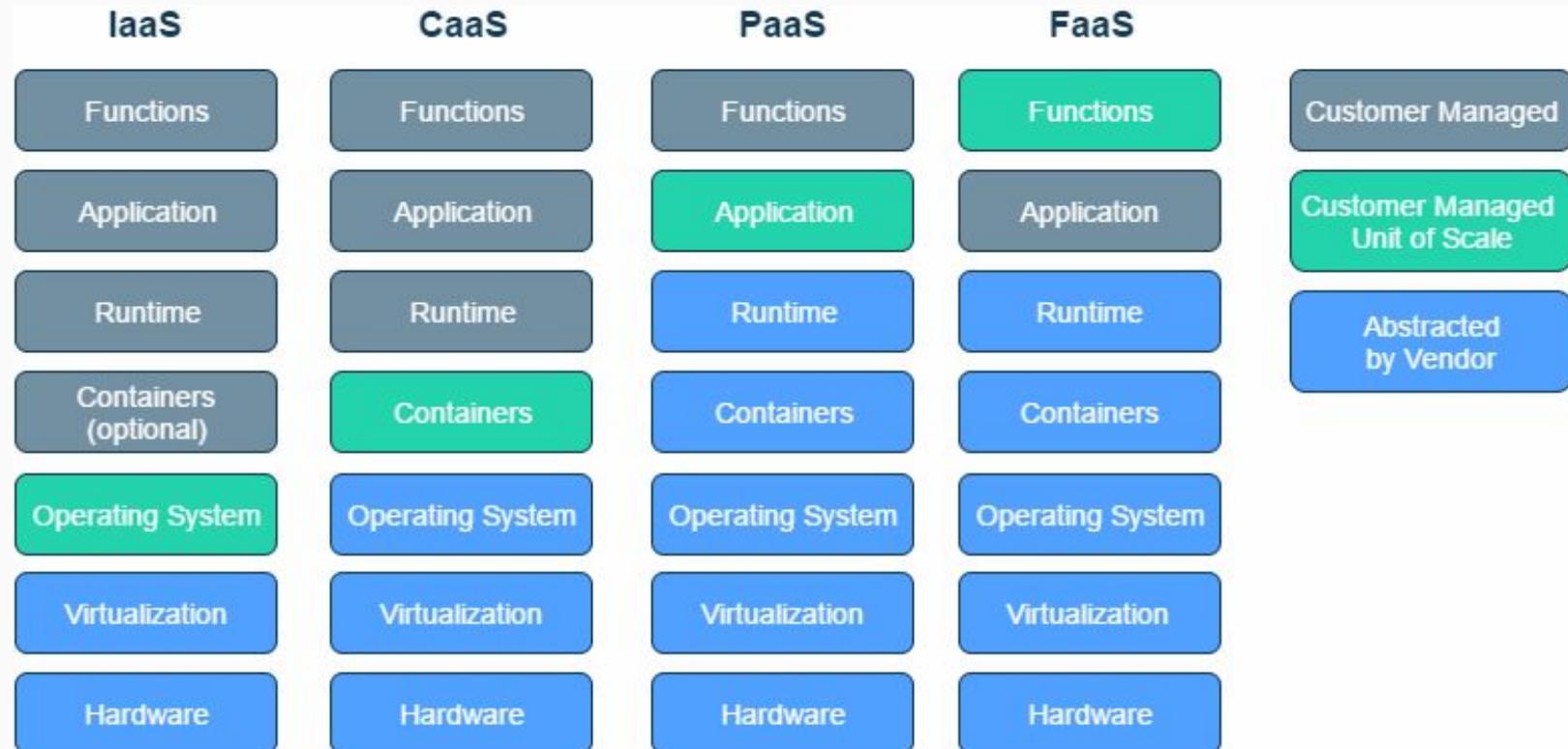


Greater testability of the application logic



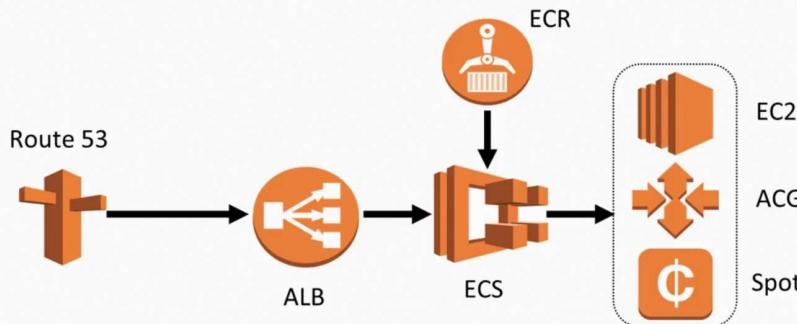
Most infrastructure problems are handled automatically

2. What is Serverless?

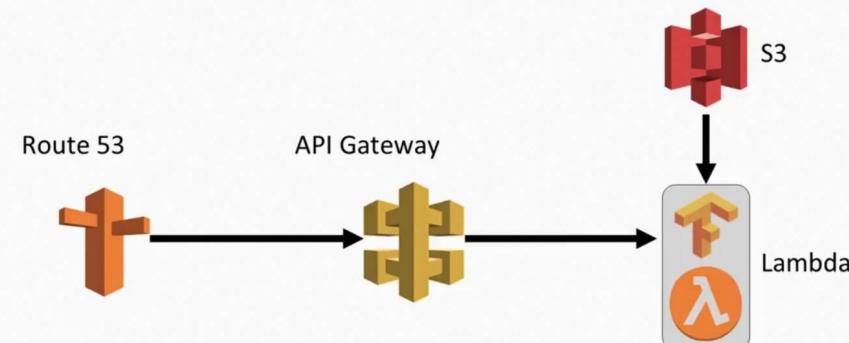


3. Why Serverless Deep Learning?

- only code + libraries
- 1,000 concurrent executions (upto 10,000)
- ~20,000 runs for \$1
- pay as you go pricing model
- perfect for early stage projects
- compare server and serverless architecture for DL



Usual AWS Architecture for DL



Architecture for DL Using Lambda

4. Where Serverless Deep Learning Works and Where it Doesn't Work?

Where it works?	Where it doesn't work?
Deploy your model for pet project	Have very complex model
Want to make simple MVP for your startup project	Your model consumes a lot of data
Simple model this architecture will reduce cost	Your model requires high CPU server to operate
Have peak loads and it is hard to manage clusters	You need to have real-time response

Need to consider:

- Time limitation, CPU limitation, Memory limitation

5. Intro to AWS Lambda Functions



A screenshot of a code editor window titled "lambda_function". The file "lambda_function.py" contains the following Python code:

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9
```

Pros	Cons	Limits
Easy to deploy (no docker)	No local debug	Max 3 GB RAM
Easy to connect to triggers (API, S3, SQS, DynamoDB)	Unpredictable warm containers	Max 500 MB disk
Easy to scale	Stateless	Max 5 min execution time
Relatively cheap		CPU is proportional to provisioned memory

5. Intro to AWS Lambda Functions

Step1: create AWS account to get “Access Key ID” and “Secret Access Key”

The screenshot shows the AWS Management Console with the URL `console.aws.amazon.com` in the address bar. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a user icon for 'Rustem', and 'Global' and 'Support' dropdowns. Below the navigation, there are four numbered steps: 1, 2, 3, and 4, with step 4 highlighted in blue. The main content area has a heading 'Add user' and a success message box. The message box contains a green checkmark icon and the word 'Success'. It states: 'You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.' It also provides a sign-in URL: <https://339543757547.sigin.aws.amazon.com/console>. At the bottom of the message box is a 'Download .csv' button with a CSV icon. Below the message box is a table with three columns: 'User', 'Access key ID', and 'Secret access key'. A single row is visible, showing a user named 'lambda' with an Access key ID of 'AKIAJ6FGC7EUS375BTLA' and a Secret access key starting with '***** Show'.

User	Access key ID	Secret access key
lambda	AKIAJ6FGC7EUS375BTLA	***** Show

5. Intro to AWS Lambda Functions

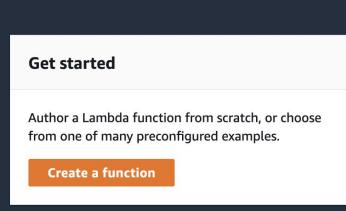
LAB 1

Step2: create “Hello World” AWS Lambda Function

COMPUTE

AWS Lambda
lets you run code without
thinking about servers.

You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.



Basic information

Function name

Enter a name that describes the purpose of your function.

helloLambda

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime

[Info](#)

Choose the language to use to write your function.

Python 3.7

Permissions

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ Choose or create an execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Create a new role from AWS policy templates

Role creation might take a few minutes. The new role will be scoped to the current function. To use it with other functions, you can modify it.

Role name

Enter a name for your new role.

helloLamb

Use only letters, numbers, hyphens, or underscores with no spaces.

Policy templates

[Info](#)

Choose one or more policy templates.

Simple microservice permissions X

DynamoDB

How it works

```
1- exports.handler = (event, context, callback) => {
2   // Succeed with the string "Hello world!"
3   callback(null, 'Hello world!');
4 };
```

Run

Next: Lambda responds to events

2.1 visit <https://ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#/begin>

Just write the code

Above is a simple Node.js Lambda function. Try changing callback values and then run the function before going to the next step.

2.2 create Lambda function

5. Intro to AWS Lambda Functions

LAB 1

Step2: create “Hello World” AWS Lambda Function

2.3 Check your Lambda function at Designer view

The screenshot shows the AWS Lambda function configuration page for a function named "helloLambda". The ARN of the function is listed as `arn:aws:lambda:ap-southeast-1:013044983696:function:helloLambda`. The top navigation bar includes links for Lambda, Functions, and helloLambda, along with buttons for Throttle, Qualifiers, Actions, Select a test event, Test, and Save.

The main interface is divided into two tabs: Configuration (selected) and Monitoring. Below these tabs, there is a section titled "Designer" which is currently collapsed (indicated by a downward arrow).

The "Designer" section displays the function's architecture. It shows a central box labeled "helloLambda" with an orange Lambda icon. Below it is a "Layers" section indicating "(0)" layers. To the left of the main function box is a key icon with a plus sign, and below it is a button labeled "+ Add trigger".

On the right side of the Designer section, there are two boxes representing triggers: "Amazon CloudWatch Logs" and "Amazon DynamoDB". Below these boxes is a dashed-line box containing the text "Resources that the function's role has access to appear here".

5. Intro to AWS Lambda Functions

LAB 1

Step2: create “Hello World” AWS Lambda Function

2.4 You can modify Function Code here

Function code [Info](#)

Code entry type [Edit code inline](#) Runtime [Python 3.7](#)

The screenshot shows the AWS Lambda function configuration interface. At the top, there are dropdown menus for 'Code entry type' (set to 'Edit code inline') and 'Runtime' (set to 'Python 3.7'). Below this is a code editor window titled 'lambda_function'. The left sidebar shows an 'Environment' section with a 'helloLambda' folder containing a file named 'lambda_function.py'. The main code editor area contains the following Python code:

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9
```

Step3: Installation of Serverless Framework

3.1 Make sure you install all of these requirements

- Node, NPM:
<https://nodejs.org/download>
- Python, Pip:
<https://www.python.org/downloads>
<https://pip.pypa.io/en/stable/installing>
- AWS CLI:
<https://docs.aws.amazon.com/cli/latest/userguide/installing.html>

3.2 Installing Serverless

<https://serverless.com/framework/docs/provider/aws/guide/installation>

```
$ npm install -g serverless  
$ serverless --version (e.g., 1.30.2)
```

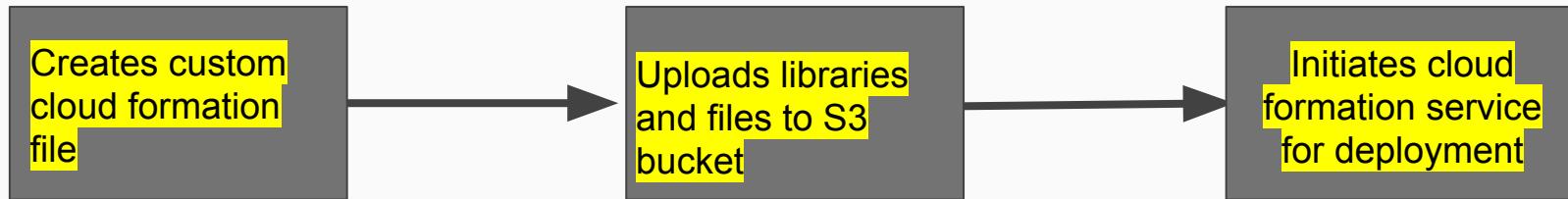
```
+-----+  
|  
| Serverless Framework successfully installed!  
| To start your first project, run "serverless".  
|  
+-----+
```

5. Intro to AWS Lambda Functions

LAB 1

Step4: Deploying AWS Lambda Functions using serverless framework

Deployment process



```
index.py x serverless.yml x
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

```
service: helloworld
frameworkVersion: ">=1.2.0 <2.0.0"
provider:
  name: aws
  region: ap-southeast-1
  runtime: python3.6
  memorySize: 128
  timeout: 10
functions:
  main:
    handler: index.handler
```

```
lab4.2
├── .serverless
│   ├── bleach
│   │   └── bleach-1.5.0.dist-info (bleach 1.5.0)
│   ├── enum
│   │   └── enum34-1.1.6.dist-info (enum34 1.1.6)
│   ├── external
│   │   └── google
│   │       ├── html5lib
│   │       │   └── html5lib-0.9999999.dist-info
│   │       ├── markdown
│   │       │   └── Markdown-3.0.1.dist-info (Markdown 3.0.1)
│   │       ├── numpy
│   │       │   └── numpy-1.15.4.dist-info
│   │       ├── pkg_resources
│   │       │   └── protobuf-3.6.1.dist-info
│   │       ├── setuptools
│   │       │   └── setuptools-40.6.2.dist-info
│   │       └── six-1.11.0.dist-info (six 1.11.0)
│   ├── tensorflow
│   │   ├── tensorflow-1.4.0.dist-info (tensorflow 1.4.0)
│   │   └── tensorflow_tensorboard-0.4.0.dist-info (tensorflow-tensorboard 0.4.0)
│   └── werkzeug
        └── Werkzeug-0.14.1.dist-info (Werkzeug 0.14.1)
index.py
protobuf-3.6.1-py3.6-nspkg.pth
serverless.yml
six.py
```

```
(enerkey_backend) warodoms-mbp:lab3.2 kwarodoms serverless deploy
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
.....
```

3. config file

1. libraries

2. lambda code

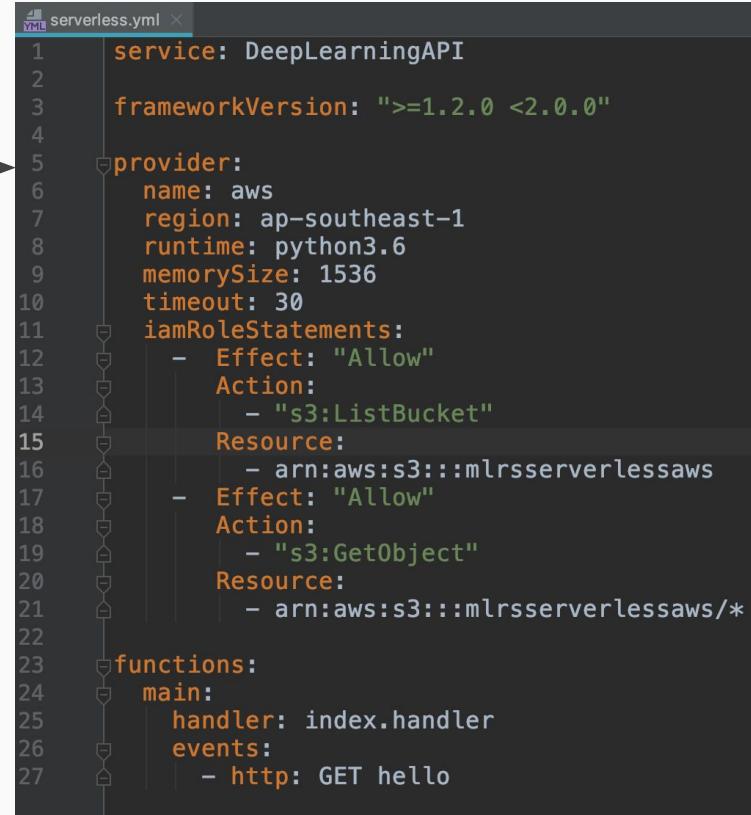
5. Intro to AWS Lambda Functions

LAB 1

Step4: Deploying AWS Lambda Functions using serverless framework

4.1 Creates custom cloud formation file

Creates custom
cloud formation file



```
serverless.yml ×
1  service: DeepLearningAPI
2
3  frameworkVersion: ">=1.2.0 <2.0.0"
4
5  provider:
6    name: aws
7    region: ap-southeast-1
8    runtime: python3.6
9    memorySize: 1536
10   timeout: 30
11
12   iamRoleStatements:
13     - Effect: "Allow"
14       Action:
15         - "s3>ListBucket"
16       Resource:
17         - arn:aws:s3:::mlrsserverlessaws
18     - Effect: "Allow"
19       Action:
20         - "s3GetObject"
21       Resource:
22         - arn:aws:s3:::mlrsserverlessaws/*
23
24   functions:
25     main:
26       handler: index.handler
27       events:
28         - http: GET hello
```

5. Intro to AWS Lambda Functions

LAB 1

4.2 Upload libraries and files to S3 bucket and deploy AWS Lambda Function

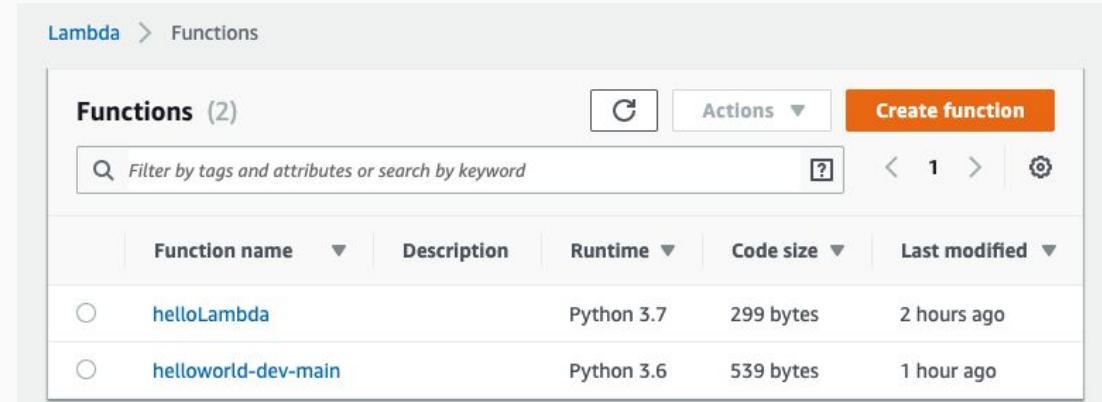
4.2.1 Setup with serverless config credentials command

```
$ serverless config credentials --provider aws --key AKIAIOSFODNN7EXAMPLE --secret  
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

4.2.2 Upload libraries and files to S3 bucket

```
$ serverless deploy
```

```
Warodoms-MBP:Chapter02 kwarodom$ serverless deploy
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
.....
Serverless: Stack create finished...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading artifacts...
Serverless: Uploading service helloworld.zip file to S3 (539 B)...
Serverless: Validating template...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.....
Serverless: Stack update finished...
Service Information
service: helloworld
stage: dev
region: ap-southeast-1
stack: helloworld-dev
resources: 5
api keys:
  None
endpoints:
  None
functions:
  main: helloworld-dev-main
layers:
  None
Serverless: Run the "serverless" command to setup monitoring, troubleshooting and testing.
Warodoms-MBP:Chapter02 kwarodom$
```



The screenshot shows the AWS Lambda Functions console interface. At the top, there's a navigation bar with 'Lambda' and 'Functions'. Below it is a search bar with placeholder text 'Filter by tags and attributes or search by keyword'. To the right of the search bar are buttons for 'Actions' and 'Create function'. The main area is titled 'Functions (2)' and lists two entries:

Function name	Description	Runtime	Code size	Last modified
helloLambda		Python 3.7	299 bytes	2 hours ago
helloworld-dev-main		Python 3.6	539 bytes	1 hour ago

4.3 Invoke lambda function

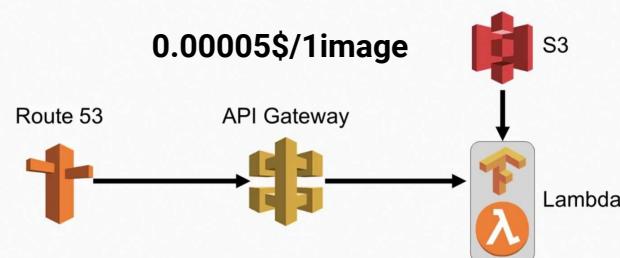
```
$ serverless invoke --function main # invoke lambda function on cloud
```

```
$ serverless invoke local --function main # invoke lambda function locally
```

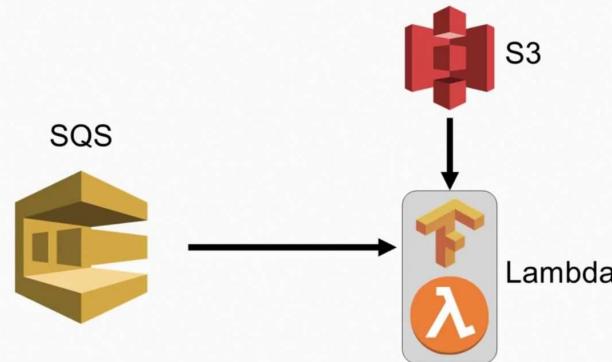
```
Warodoms-MBP:Chapter02 kwarodom$ serverless invoke --function main  
"Hello world"
```

6. Deep Learning Deployment Process on AWS Lambda Functions

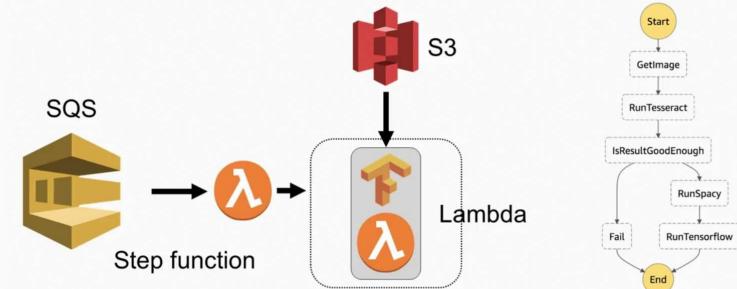
What will we make



1. Deep Learning API



**2. Deep Learning
Batch Processing**



3. Deep Learning Workflow

<https://github.com/tensorflow/models>
<https://tfhub.dev>

6. Deep Learning Deployment Process on AWS Lambda Functions

Simple Tensorflow Example - MNIST Example



```
2019-08-01 22:22:31.726315: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU  
not compiled to use: AVX2 FMA  
60000/60000 [=====] - 9s 145us/step - loss: 0.2022 - acc: 0.9414  
Epoch 2/2  
60000/60000 [=====] - 9s 142us/step - loss: 0.0803 - acc: 0.9758  
Evaluation:  
10000/10000 [=====] - 0s 32us/step  
[0.07298920549713075, 0.9765]  
Evaluation by imported model:  
10000/10000 [=====] - 0s 35us/step  
[0.07298920549713075, 0.9765]
```

```
import tensorflow as tf  
  
mnist = tf.keras.datasets.mnist  
  
(x_train, y_train), (x_test, y_test) = mnist.load_data()  
x_train, x_test = x_train / 255.0, x_test / 255.0  
  
model = tf.keras.models.Sequential([  
    tf.keras.layers.Flatten(input_shape=(28, 28)),  
    tf.keras.layers.Dense(512, activation=tf.nn.relu),  
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)  
)  
model.compile(optimizer='adam',  
              loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])  
  
model.fit(x_train, y_train, epochs=2)  
print('Evaluation: ')  
print(model.evaluate(x_test, y_test))  
model.save('MNISTmodel.h5')  
modelImported = tf.keras.models.load_model('MNISTmodel.h5')  
print('Evaluation by imported model: ')  
print(modelImported.evaluate(x_test, y_test))
```

6. Deep Learning Deployment Process on AWS Lambda Functions

```
import numpy as np
import tensorflow as tf
import re

class NodeLookup(object):
    """Converts integer node ID's to human readable labels."""

    def __init__(self,
                 label_lookup_path=None,
                 uid_lookup_path=None):
        if not label_lookup_path:
            label_lookup_path = os.path.join(
                '/tmp/imagenet/', 'imagenet_2012_challenge_label_map_proto.pbtxt')
        if not uid_lookup_path:
            uid_lookup_path = os.path.join(
                '/tmp/imagenet/', 'imagenet_synset_to_human_label_map.txt')
        self.node_lookup = self.load(label_lookup_path, uid_lookup_path)

    def load(self, label_lookup_path, uid_lookup_path):...

    def id_to_string(self, node_id):...

image = 'inputimage.png'
image_data = tf.gfile.FastGFile(image, 'rb').read()

with tf.gfile.FastGFile('classify_image_graph_def.pb', 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    tf.import_graph_def(graph_def, name='')

SESSION = tf.InteractiveSession()
softmax_tensor = tf.get_default_graph().get_tensor_by_name('softmax:0')

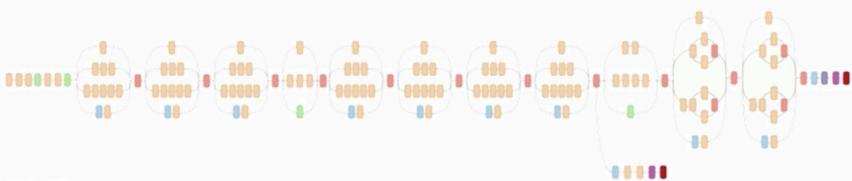
predictions = SESSION.run(softmax_tensor, {'DecodeJpeg/contents:0': image_data})

predictions = np.squeeze(predictions)
node_lookup = NodeLookup(label_lookup_path='imagenet_2012_challenge_label_map_proto.pbtxt',
                         uid_lookup_path='imagenet_synset_to_human_label_map.txt')

top_k = predictions.argsort()[-5:][::-1]
strResult = '%s (score = %.5f)' % (node_lookup.id_to_string(top_k[0]), predictions[top_k[0]])
print()
for node_id in top_k:
    human_string = node_lookup.id_to_string(node_id)
    score = predictions[node_id]
    print('%s - %s (score = %.5f)' % (node_id, human_string, score))
```

```
169 - giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score = 0.89342)
103 - ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus (score = 0.00277)
7 - lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens (score = 0.00215)
163 - American black bear, black bear, Ursus americanus, Euarctos americanus (score = 0.00143)
61 - brown bear, bruin, Ursus arctos (score = 0.00122)
```

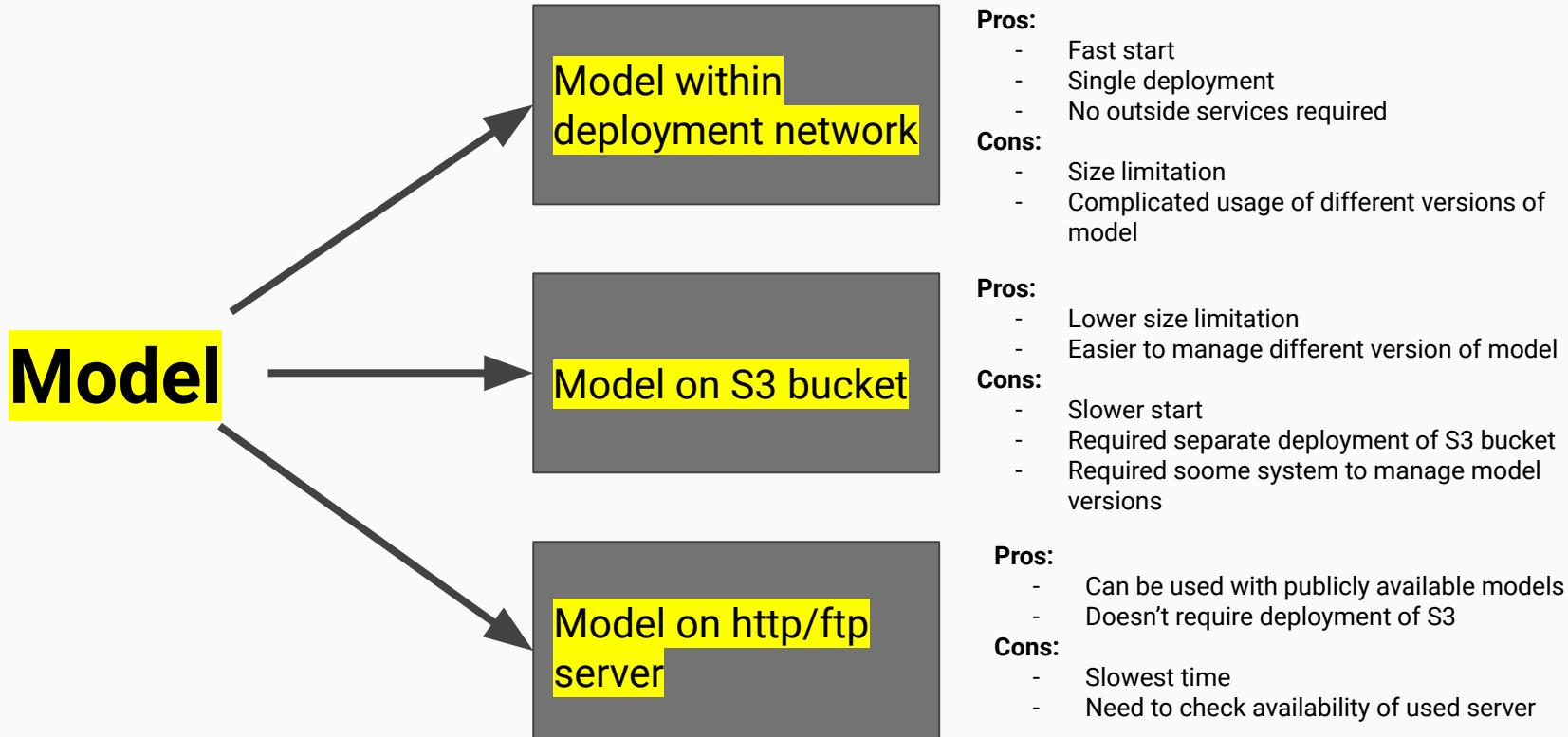
Inception v3



- Files:
 - classify_image_graph_def.pb
 - imagenet_2012_challenge_label_map_proto.pbtxt
 - imagenet_synset_to_human_label_map.txt

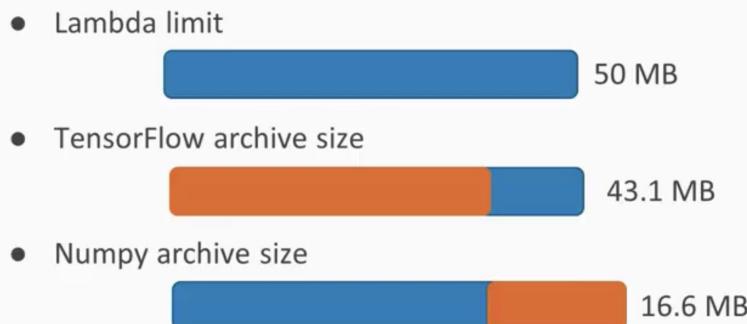
6. Deep Learning Deployment Process on AWS Lambda Functions

Where can we store our model?



6. Deep Learning Deployment Process on AWS Lambda Functions

Consideration when you want to deploy model with AWS Lambda Function



Solution

1. Compress shared libraries (.so files)
2. Delete .pyc files
3. Remove test folders, visualization folders
4. Remove libs which already exist on AWS Lambda



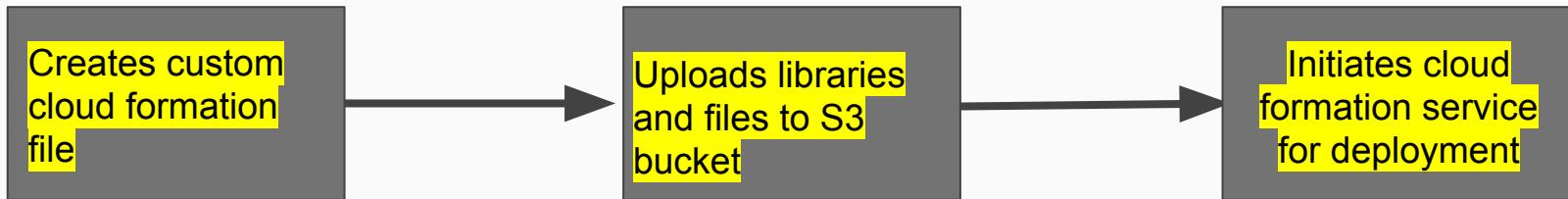
```
$ find name "*.so" | xargs strip  
$ find name "*.so.*" | xargs strip  
$ find . -name \*.pyc -delete  
  
$ rm -r pip*  
$ rm -r wheel*  
$ rm easy_install.py
```

```
$ find -type d --name "tests" -exec rm -rf {} + 182
```

7. Deploying Tensorflow Model with AWS Lambda Function

LAB 3.1

Deploying Tensorflow on AWS Lambda using Pre-trained Model



```
index.py x serverless.yml x
1  service: helloworld
2
3  frameworkVersion: ">=1.2.0 <2.0.0"
4
5  provider:
6    name: aws
7    region: ap-southeast-1
8    runtime: python3.6
9    memorySize: 128
10   timeout: 10
11
12  functions:
13    main:
14      handler: index.handler
```

```
lab4.2
├── serverless
├── bleach
├── enum
├── enum34
├── external
├── google
├── html5lib
├── markdown
├── numpy
├── numpy-1.15.4.dist-info
├── pkg_resources
├── protobuf
├── setuptools
├── six
├── tensorflow
├── tensorflow-1.4.0.dist-info
├── tensorflow-1.4.0.tensorflow-0.4.0.dist-info
└── werkzeug
    └── Werkzeug-0.14.1.dist-info
```

```
(enerkey_backend) warodoms-mbp:lab3.2 kwarodom$ serverless deploy
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
.....
```

3. config file

1. libraries

2. lambda code

7. Deploying Tensorflow Model with AWS Lambda Function

LAB 3.1

Deploying Tensorflow on AWS Lambda using Pre-trained Model

1. Create bucket

The screenshot shows the AWS S3 Management Console interface. On the left, there's a sidebar with 'Amazon S3' and a 'Buckets' section. In the main area, there's a search bar, a dropdown for 'All access types', and a button for 'Create bucket'. Below these are buttons for 'Edit public access settings', 'Empty', and 'Delete'. To the right, it says 'Regions' and '61 Buckets'. A table lists five buckets:

Bucket Name	Region	Last Modified
stepf-dev-serverlessdeploymentbucket-17...	US West (Oregon)	Sep 5, 2018 9:06:22 AM GMT-0700
stepfcluster-dev-serverlessdeploymentbuc...	US East (N. Virginia)	Oct 3, 2018 10:51:48 PM GMT-0700
tensorflow-dev-serverlessdeploymentbuck...	US East (N. Virginia)	Jul 6, 2018 11:12:05 AM GMT-0700
testservice-dev-serverlessdeploymentbuc...	US East (N. Virginia)	Aug 3, 2018 5:44:25 PM GMT-0700

At the bottom, there are links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

2. Upload .zip file to S3

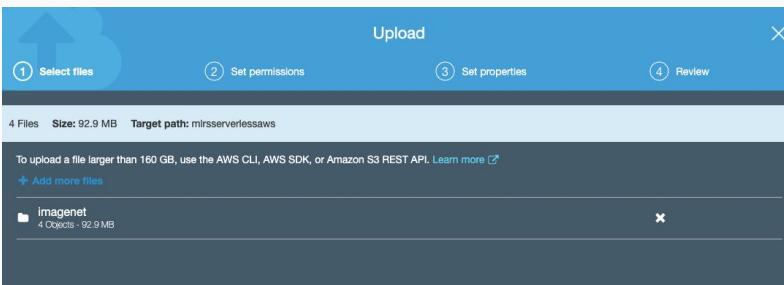
The screenshot shows the 'Upload' wizard for AWS S3. It has four steps: 1. Select files, 2. Set permissions, 3. Set properties, and 4. Review. Step 1 is active, showing '1 Files' with a size of '44.0 MB' and a target path of 'mirserverlessaws'. There's a note about uploading large files via the AWS CLI or SDK. Step 2 shows a file named 'serverlessDeepLearning.zip' with a size of '44.0 MB'. Step 3 and 4 are partially visible at the bottom.

7. Deploying Tensorflow Model with AWS Lambda Function

LAB 3.1

Deploying Tensorflow on AWS Lambda using Pre-trained Model

3. Upload model files



4. Create IAM role for AWS Lambda function

The screenshot shows the 'Create role' interface in AWS IAM. Step 1 is selected. It asks 'Select type of trusted entity' with four options: 'AWS service' (selected), 'Another AWS account', 'Web identity', and 'SAML 2.0 federation'. Below this, it says 'Allows AWS services to perform actions on your behalf. Learn more'. Step 2 is highlighted in yellow and says 'Choose the service that will use this role'. It lists services under 'EC2' and 'Lambda'. Under 'Lambda', it says 'Allows Lambda functions to call AWS services on your behalf.' A yellow box highlights the 'Lambda' section with the text 'Choose Lambda'.

EC2	Lambda
Allows EC2 instances to call AWS services on your behalf.	Allows Lambda functions to call AWS services on your behalf.
API Gateway	Comprehend
AWS Backup	Config
AWS Support	Connect
Amplify	DMS
AppSync	Data Lifecycle Manager
	ElastiCache
	Elastic Beanstalk
	Elastic Container Service
	Elastic Transcoder
	Elastic Load Balancing
	Lex
	License Manager
	Macie
	MediaConvert
	SNS
	SWF
	SageMaker
	SMS
	Security Hub

Deploying Tensorflow on AWS Lambda using Pre-trained Model

4. Create role for our lambda function

Create role

1 2 3 4

▼ Attach permissions policies

Choose one or more policies to attach to your new role.

[Create policy](#) [Cancel](#)

For simplicity choose AdministratorAccess

Showing 558 results

	Policy name	Used as	Description
<input checked="" type="checkbox"/>	AdministratorAccess	Permissions policy (3)	Provides full access to AWS services ...
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	None	Provide device setup access to Alexa...
<input type="checkbox"/>	AlexaForBusinessFullAccess	None	Grants full access to AlexaForBusiness...
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	None	Provide gateway execution access to ...

7. Deploying Tensorflow Model with AWS Lambda Function

LAB 3.1

Deploying Tensorflow on AWS Lambda using Pre-trained Model

4. Create role for our lambda function

Create role

Review

Provide the required information below and review this role before you create it.

1 2 3 4

Name your Role for Lambda Func.

Role name* LambdaAdminRole

Use alphanumeric and '+=, @-' characters. Maximum 64 characters.

Role description Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=, @-' characters.

Trusted entities AWS service: lambda.amazonaws.com

Policies AdministratorAccess

Permissions boundary Permissions boundary is not set

No tags were added.

Click done

The role LambdaAdminRole has been created.

Deploying Tensorflow on AWS Lambda using Pre-trained Model

5. Create Lambda Function

Basic information

Function name

Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info

Choose the language to use to write your function.

Permissions Info

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ Choose or create an execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

[View the LambdaAdminRole role](#) on the IAM console.

Deploying Tensorflow on AWS Lambda using Pre-trained Model

6. Change Handler

Change this

Function code [Info](#)

Code entry type [Edit code inline](#)

Runtime [Python 3.7](#)

Handler [Info](#) index_handler

File Edit Find View Go Tools Window

Environment testtensorflowlambda lambda_function.py

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9
```

Deploying Tensorflow on AWS Lambda using Pre-trained Model

7. Make sure you have enough resources to run your Lambda Function

Basic settings

Description

Change this

Memory (MB) Info

Your function is allocated CPU proportional to the memory configured.



Timeout Info

 min sec

Change this

Deploying Tensorflow on AWS Lambda using Pre-trained Model

8. Parse the link of your code to Lambda Function

Function code [Info](#)

 The deployment package of your Lambda function "testtensorflowlambda" is too large to enable inline code editing. However, you can still invoke your function.

Code entry type

Upload a file from Amazon S3

Runtime

Python 3.7

Handler [Info](#)

index_handler

Amazon S3 link URL

Paste an S3 link URL to your function code .zip.

s3://mlrsserverlessaws/serverlessDeepLearning.zip

Change this



Deploying Tensorflow on AWS Lambda using Pre-trained Model

9. Let's test the function

The screenshot shows the AWS Lambda function configuration page for 'testtensorflowlambda'. The top navigation bar includes 'Lambda > Functions > testtensorflowlambda'. On the right, there are buttons for 'Throttle', 'Qualifiers ▾', 'Actions ▾', 'Select a test event ▾', 'Test' (which has a yellow box around it and an arrow pointing to it), and 'Save'. Below the navigation, the function name 'testtensorflowlambda' is displayed with a 'Saved' badge. The 'Configuration' tab is selected, showing the 'Designer' section. The designer interface displays the function code and its layers. A button '+ Add trigger' is visible on the left. At the bottom, a note says 'Resources that the function's role has access to appear here'. A yellow box highlights the 'Test' button, and an arrow points to it from the right.

7. Deploying Tensorflow Model with AWS Lambda Function

LAB 3.1

Deploying Tensorflow on AWS Lambda using Pre-trained Model

10. Test Lambda Function

The screenshot shows the AWS Lambda function test interface for a function named "testlambda". The ARN is listed as `arn:aws:lambda:ap-southeast-1:013044983696:function:testlambda`. The execution result is shown as succeeded, with a log entry:

```
[{"giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score = 0.89342)", "ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus (score = 0.00277)", "lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens (score = 0.00215)", "American black bear, black bear, Ursus americanus, Euarctos americanus (score = 0.00143)", "brown bear, bruin, Ursus arctos (score = 0.00122)"}]
```

The summary section includes:

- Code SHA-256: `t+Nbk7cKNTEvFAuTKBBFUzqplEkBxPLkkJGpvRm5E=`
- Duration: 4599.54 ms
- Resources configured: 1728 MB
- Request ID: `7c708ac7-f084-453b-afb8-67e55b404154`
- Billed duration: 4600 ms
- Max memory used: 511 MB

The Log output section shows:

```
START RequestId: 7c708ac7-f084-453b-afb8-67e55b404154 Version: $LATEST
END RequestId: 7c708ac7-f084-453b-afb8-67e55b404154
REPORT RequestId: 7c708ac7-f084-453b-afb8-67e55b404154 Duration: 4599.54 ms Billed Duration: 4600 ms Memory Size: 1728 MB Max Memory Used: 511 MB
```

7. Deploying Tensorflow Model with AWS Lambda Function

LAB 3.1

Deploying Tensorflow on AWS Lambda using Pre-trained Model

12. Let's see the code what was happened

The screenshot shows a code editor with a dark theme. The left sidebar displays a project structure for 'Serverless_Deep_Learning_MLR'. The 'index.py' file is open in the editor. The code is a Lambda function handler that downloads an image from S3, runs inference on it using a pre-trained Tensorflow model, and returns the result.

```
Project ▾ index.py ▾
Serverless_Deep_Learning_MLR
├── Inceptions
├── lab1
├── lab2
└── lab3
    ├── lab3.1
    │   └── imagenet
    └── Lambdapack
        └── Archive4.zip
    └── lab3.2
└── lab4
    └── LICENSE
    └── README.md
└── External Libraries
└── Scratches and Consoles

1  import boto3
2  import numpy as np
3  import tensorflow as tf
4  import os.path
5  import re
6  from urllib.request import urlretrieve
7  import json
8
9  SESSION = None
10 strBucket = 'mlrsserverlessaws'
11
12 def handler(event, context):
13     global strBucket
14     if not os.path.exists('/tmp/imagenet/'):
15         os.makedirs('/tmp/imagenet/')
16
17     strFile = '/tmp/imagenet/inputimage.png'
18
19     downloadFromS3(strBucket, 'imagenet/inputimage.png', strFile)
20
21     global SESSION
22     if SESSION is None:
23         downloadFromS3(strBucket, 'imagenet/imagenet_2012_challenge_label_map_prototxt')
24         downloadFromS3(strBucket, 'imagenet/imagenet_synset_to_human_label_map.txt')
25         image = os.path.join('/tmp/imagenet/', 'inputimage.png')
26         strResult = run_inference_on_image(image)
27
28     return strResult
29
30 def run_inference_on_image(image):
31     image_data = tf.gfile.FastGFile(image, 'rb').read()
32     global SESSION
33     if SESSION is None:
34         SESSION = tf.InteractiveSession()
35         create_graph()
```

Deploying Tensorflow on AWS Lambda using Serverless Framework

Here is the deployment process

1. Create a custom cloud formation file
2. Upload libraries and files to S3 bucket
3. Initiates cloud formation service for deployment

8. Deploying Tensorflow Model with Serverless Framework

LAB 3.2

2. Check serverless.yml, index.py files

The screenshot shows a code editor with two tabs: 'serverless.yml' and 'index.py'. The 'serverless.yml' tab displays the following YAML configuration:

```
service: deeplearninglambda
frameworkVersion: ">=1.2.0 <2.0.0"
provider:
  name: aws
  region: ap-southeast-1
  runtime: python3.6
  memorySize: 1536
  timeout: 60
  iamRoleStatements:
    - Effect: "Allow"
      Action:
        - "s3>ListBucket"
      Resource:
        - arn:aws:s3:::serverlessdeeplearningarm
    - Effect: "Allow"
      Action:
        - "s3GetObject"
      Resource:
        - arn:aws:s3:::serverlessdeeplearningarm/*
functions:
  main:
    handler: index.handler
```

The project directory structure on the left is as follows:

- Serverless_Deep_Learning_MLRS (sources root)
 - Inceptions
 - lab1
 - lab2
 - lab3
 - lab3.1
 - lab3.2
 - .serverless
 - bleach
 - bleach-1.5.0.dist-info (bleach 1.5.0)
 - enum
 - enum34-1.1.6.dist-info (enum34 1.1.6)
 - external
 - google
 - html5lib
 - html5lib-0.9999999.dist-info
 - markdown
 - Markdown-3.0.1.dist-info (Markdown 3.0.1)
 - numpy
 - numpy-1.15.4.dist-info
 - pkg_resources
 - protobuf-3.6.1.dist-info
 - setupools
 - setupools-40.6.2.dist-info
 - six-1.11.0.dist-info (six 1.11.0)
 - tensorboard
 - tensorflow
 - tensorflow-1.4.0.dist-info (tensorflow 1.4.0)
 - tensorflow_tensorboard-0.4.0.dist-info (tenso
 - werkzeug
 - Werkzeug-0.14.1.dist-info (Werkzeug 0.14.1)
 - index.py
 - protobuf-3.6.1-py3.6-nspkg.pth
 - serverless.yml
 - six.py

3. Deploy serverless function

```
(enerkey_backend) warodoms-mbp:lab3.2 kwarodom$ serverless deploy
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
.....
Serverless: Stack create finished...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading artifacts...
Serverless: Uploading service deeplearninglambda.zip file to S3 (55.97 MB)...
Serverless: Validating template...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.....
Serverless: Stack update finished...
Service Information
service: deeplearninglambda
stage: dev
region: ap-southeast-1
stack: deeplearninglambda-dev
resources: 5
api keys:
  None
endpoints:
  None
functions:
  main: deeplearninglambda-dev-main
layers:
  None
Serverless: Run the "serverless" command to setup monitoring, troubleshooting and testing.
```

8. Deploying Tensorflow Model with Serverless Framework

LAB 3.2

4. Check if code is uploaded to S3 bucket

The screenshot shows the AWS S3 buckets console. At the top, there is a search bar labeled "Search for buckets". Below it are four buttons: "+ Create bucket", "Edit public access settings", "Empty", and "Delete". A dropdown menu is open under the "+ Create bucket" button. The main list contains the following buckets:

- Bucket name ▾
- altotech
- altotechpublic
- aws-athena-query-results-013044983696-us-west-2
- deeplearninglambda-dev-serverlessdeploymentbucket-1wkos7xdrss3
- deeplearninglambda-dev-serverlessdeploymentbucket-3lbl5cdfswxk

folder will be zipped and
uploaded to S3 bucket

8. Deploying Tensorflow Model with Serverless Framework

LAB 3.2

5. Check if the Lambda function get created

deeplearninglambda-dev-main

Throttle Qualifiers Actions test Test Save

This function belongs to the AWS CloudFormation stack **deeplearninglambda-dev**. [Manage this stack](#) on the CloudFormation console. X

Configuration Monitoring

▼ Designer

Go back to application [deeplearninglambda-dev](#)

 **deeplearninglambda-dev-main**

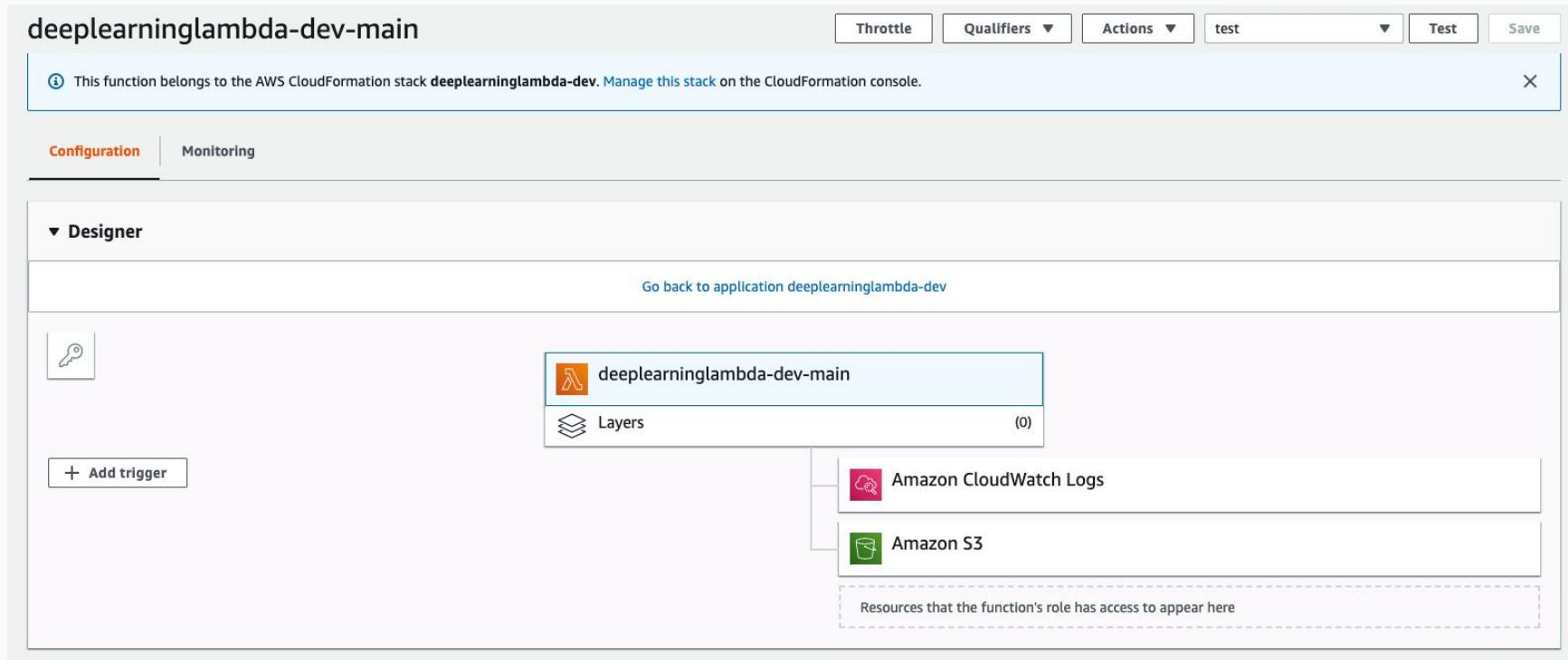
 Layers (0)

 + Add trigger

 Amazon CloudWatch Logs

 Amazon S3

Resources that the function's role has access to appear here



8. Deploying Tensorflow Model with Serverless Framework

LAB 3.2

6. Now let's test the deployed function

deeplearninglambda-dev-main

Throttle Qualifiers Actions test Test Save

This function belongs to the AWS CloudFormation stack **deeplearninglambda-dev**. [Manage this stack](#) on the CloudFormation console.

Execution result: succeeded (logs)

Details

The section below shows the result returned by your function execution.

```
[{"name": "giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score = 0.89342)", "score": 0.89342}, {"name": "ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus (score = 0.00277)", "score": 0.00277}, {"name": "lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens (score = 0.00215)", "score": 0.00215}, {"name": "American black bear, black bear, Ursus americanus, Euarctos americanus (score = 0.00143)", "score": 0.00143}, {"name": "brown bear, bruin, Ursus arctos (score = 0.00122)", "score": 0.00122}]
```

Summary

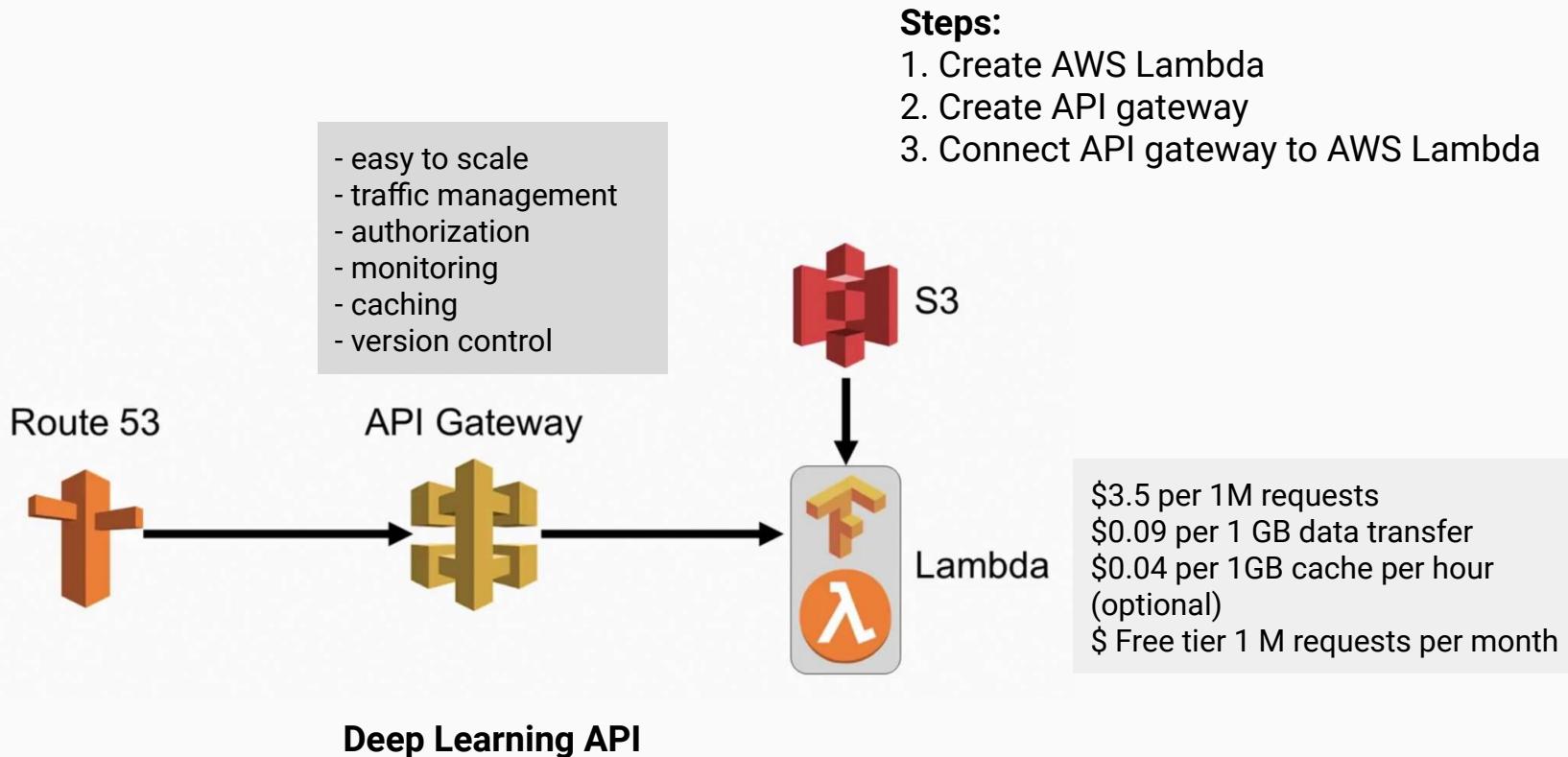
Code SHA-256	Request ID
Yh+dyqn3Rej7qbSljpeGIHdMLXb9fEdQgJUVbaquhKg=	58cc1334-97af-4724-82b1-b42543188dcc
Duration	Billed duration
5121.98 ms	5200 ms
Resources configured	Max memory used
1536 MB	581 MB

Log output

The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: 58cc1334-97af-4724-82b1-b42543188dcc Version: $LATEST
END RequestId: 58cc1334-97af-4724-82b1-b42543188dcc
REPORT RequestId: 58cc1334-97af-4724-82b1-b42543188dcc Duration: 5121.98 ms Billed Duration: 5200 ms Memory Size: 1536 MB Max Memory Used: 581 MB
```

9. Creating Deep Learning API



9. Creating Deep Learning API

LAB 4.1

1. Create Lambda Function

The screenshot shows the AWS Lambda function configuration interface for a function named 'testapigateway'. The top navigation bar includes 'Lambda > Functions > testapigateway' and an ARN link. The main header has tabs for 'Throttle', 'Qualifiers', 'Actions', and 'test'. Below the header, a message box displays a green checkmark and the text 'Execution result: succeeded (logs)' with a 'Details' link. The 'Configuration' tab is selected, showing the 'Designer' section. In the Designer, there is a placeholder icon for triggers ('+ Add trigger') and a main panel for the function itself. The function panel contains the name 'testapigateway', a 'Layers' section with '(0)', and an 'All' button. A dashed box at the bottom indicates where resources can be added. The 'Monitoring' tab is also visible.

9. Creating Deep Learning API

LAB 4.1

2. Add trigger

Add API Gateway
as a trigger

API Gateway

api application-services aws serverless

We'll set up an API Gateway endpoint with a [proxy integration type](#) (learn more about the [input](#) and [output](#) format). Any method (GET, POST, etc.) will trigger your integration. To set up more advanced method mappings or subpath routes, visit the [Amazon API Gateway console](#).

API
Pick an existing API, or create a new one.

Create a new API

Security
Configure the security mechanism for your API endpoint.

Open

Warning: Your API endpoint will be publicly available and can be invoked by all users.

Additional settings

API name
Enter a name to uniquely identify your API.

testapigateway-API

Deployment stage
The name of your API's deployment stage.

default

Lambda > Add trigger

Add trigger

Trigger configuration

Select a trigger

API Gateway

AWS IoT

Application Load Balancer

CloudWatch Events

CloudWatch Logs

CodeCommit

Cognito Sync Trigger

DynamoDB

Kinesis

S3

SNS

SQS

9. Creating Deep Learning API

LAB 4.1

3. Now we can test API in a browser

The screenshot shows the AWS Lambda Test API interface. At the top, there's a header with "testapigateway" and buttons for "Throttle" and "Qualifiers". Below the header, there's a sidebar with icons for Lambda, Layers, and API Gateway. The "API Gateway" section is selected and highlighted in blue. It contains a "Layers" section with "(0)" and a "All" section with a note: "Resources that the function's role has access to". A "Add trigger" button is also present. To the right of the sidebar, a URL bar shows the endpoint: <https://cayxwyj9kd.execute-api.ap-southeast-1.amazonaws.com/default/testapigateway>. The main content area displays the response "Hello from Lambda! ". On the left, there's a detailed view of the API Gateway settings, including the API name "testapigateway-API", endpoint "https://cayxwyj9kd.execute-api.ap-southeast-1.amazonaws.com/default/testapigateway", and various configuration details like "Authorization: NONE", "Enable metrics and error logging: No", and "Method: ANY".

testapigateway

Throttle Qualifiers ▾

Layers (0)

API Gateway All

Resources that the function's role has access to

+ Add trigger

https://cayxwyj9kd.execute-api.ap-southeast-1.amazonaws.com/default/testapigateway

"Hello from Lambda! "

API Gateway

testapigateway-API
arn:aws:execute-api:ap-southeast-1:013044983696:cayxwyj9kd/*/*/testapigateway

▼ Details

API: api-gateway/cayxwyj9kd/*/*/testapigateway

API endpoint: <https://cayxwyj9kd.execute-api.ap-southeast-1.amazonaws.com/default/testapigateway>

API name: testapigateway-API

Authorization: NONE

Enable metrics and error logging: No

Method: ANY

Resource path: /testapigateway

Security: NONE

Stage: default

9. Creating Deep Learning API

LAB 4.1

4. You can also create Lambda Function with API gateway from Serverless

```
serverless.yml x index.py x
YAML JSON
1 service: HelloWorldAPI
2
3 frameworkVersion: ">=1.2.0 <2.0.0"
4
5 provider:
6   name: aws
7   region: ap-southeast-1
8   runtime: python3.6
9   memorySize: 128
10  timeout: 10
11
12  functions:
13    main:
14      handler: index.handler
15      events:
16        - http: GET hello|
```

```
(enerkey_backend) warodoms-mbp:lab4.1 kwarodom$ serverless deploy
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
.....
Serverless: Stack create finished...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading artifacts...
Serverless: Uploading service HelloWorldAPI.zip file to S3 (260 B)...
Serverless: Validating template...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.....
Serverless: Stack update finished...
Service Information
service: HelloWorldAPI
stage: dev
region: ap-southeast-1
stack: HelloWorldAPI-dev
resources: 10
api keys:
  None
endpoints:
  GET - https://eg0anzlj5k.execute-api.ap-southeast-1.amazonaws.com/dev/hello
functions:
  main: HelloWorldAPI-dev-main
layers:
  None
Serverless: Run the "serverless" command to setup monitoring, troubleshooting and testing.
(enerkey_backend) warodoms-mbp:lab4.1 kwarodom$ curl https://eg0anzlj5k.execute-api.ap-southeast-1.amazonaws.com/dev/hello
(enerkey_backend) warodoms-mbp:lab4.1 kwarodom$ |
```

```
serverless.yml x index.py x
YAML JSON
1 import json
2
3 def handler(event, context):
4     print('Log event', event)
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello world API!')
8     }
```

9. Creating Deep Learning API

LAB 4.2

1 Now we will create Deep Learning API

The screenshot shows the PyCharm IDE interface. The top bar displays the project path: Serverless_Deep_Learning_MLRS / lab4 / lab4.2 / index.py. The left sidebar shows the project structure for 'lab4.2' and 'enerkey_backend'. The main code editor window contains the Python script 'index.py' with syntax highlighting for code and comments. The code uses the AWS Lambda Python runtime to download an image from S3 and run it through a pre-trained TensorFlow model for image classification.

```
from urllib.request import urlretrieve
import json

SESSION = None
strBucket = 'mlrsserverlessaws'

def handler(event, context):
    global strBucket
    global SESSION

    if not os.path.exists('/tmp/imagenet'):
        os.makedirs('/tmp/imagenet')

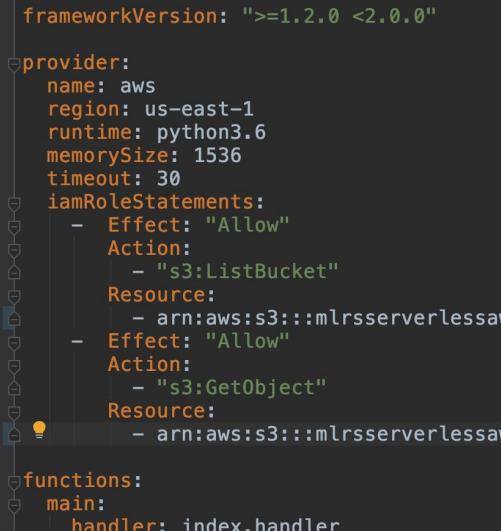
    if SESSION is None:
        downloadFromS3(strBucket, 'imagenet/imagenet_2012_challenge_label_map_proto.pb')
        downloadFromS3(strBucket, 'imagenet/imagenet_synset_to_human_label_map.txt')

    strFile = '/tmp/imagenet/inputimage.png'

    if ('queryStringParameters' in event):
        if (event['queryStringParameters'] is not None):
            if ('url' in event['queryStringParameters']):
                urlretrieve(event['queryStringParameters']['url'], strFile)
            else:
                downloadFromS3(strBucket, 'imagenet/inputimage.png', strFile)
        else:
            downloadFromS3(strBucket, 'imagenet/inputimage.png', strFile)
    else:
        downloadFromS3(strBucket, 'imagenet/inputimage.png', strFile)

    strResult = run_inference_on_image(strFile)

    return {
        'statusCode': 200,
        'body': json.dumps(strResult)
```



The screenshot shows a code editor with two tabs open: `index.py` and `serverless.yml`. The `serverless.yml` file is the primary focus, displaying configuration for a DeepLearningAPI service. The configuration includes provider details (aws, us-east-1, python3.6, 1536 memory, 30 timeout), IAM role statements for S3 actions, and a function named `main` with an HTTP event handler.

```
1  service: DeepLearningAPI
2
3  frameworkVersion: ">=1.2.0 <2.0.0"
4
5  provider:
6    name: aws
7    region: us-east-1
8    runtime: python3.6
9    memorySize: 1536
10   timeout: 30
11
12   iamRoleStatements:
13     - Effect: "Allow"
14       Action:
15         - "s3>ListBucket"
16       Resource:
17         - arn:aws:s3:::mlrsserverlessaws
18     - Effect: "Allow"
19       Action:
20         - "s3GetObject"
21       Resource:
22         - arn:aws:s3:::mlrsserverlessaws/*
23
24   functions:
25     main:
26       handler: index.handler
27       events:
28         - http: GET hello
```

9. Creating Deep Learning API

LAB 4.2

2. Let's deploy to AWS Lambda

```
(enerkey_backend) warodoms-mbp:lab4.2 kwarodom$ serverless deploy
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
.....
Serverless: Stack create finished...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading artifacts...
Serverless: Uploading service DeepLearningAPI.zip file to S3 (55.97 MB)...
Serverless: Validating template...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.....
Serverless: Stack update finished...
Service Information
service: DeepLearningAPI
stage: dev
region: ap-southeast-1
stack: DeepLearningAPI-dev
resources: 10
api keys:
  None
endpoints:
  GET - https://yzwhmr81n9.execute-api.ap-southeast-1.amazonaws.com/dev/hello
functions:
  main: DeepLearningAPI-dev-main
layers:
  None
Serverless: Run the "serverless" command to setup monitoring, troubleshooting and testing.
```

The screenshot shows the AWS Lambda console interface. On the left, a sidebar menu includes 'Dashboard', 'Applications' (which is currently selected), 'Functions', and 'Layers'. The main content area displays the 'Applications' section with four entries: 'DeepLearningAPI-dev', 'HelloWorldAPI-dev', 'deeplearninglambda-dev', and 'DeepLearningAPI-dev'. Below this, the 'DeepLearningAPI-dev-main' function configuration is shown. The 'Configuration' tab is active, displaying the function's ARN and a note that it belongs to the 'DeepLearningAPI-dev' stack. The 'Designer' tab is also visible. At the bottom, a network diagram shows the function's role interacting with 'API Gateway', 'Amazon CloudWatch Logs', and 'Amazon S3'.

3. Let's test our API



https://yzwhmr81n9.execute-api.ap-southeast-1.amazonaws.com/dev/hello

```
[  
    "giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score = 0.89342)",  
    "ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus (score = 0.00277)",  
    "lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens (score = 0.00215)",  
    "American black bear, black bear, Ursus americanus, Euarctos americanus (score = 0.00143)",  
    "brown bear, bruin, Ursus arctos (score = 0.00122)"  
]
```

9. Creating Deep Learning API

LAB 4.2

4. Now try with any image you like

Google dog

All Images Videos Maps News More Settings Tools

puppy cartoon german shepherd baby husky pug cat pitbull labrador golden retriever

Dog - Wikipedia
en.wikipedia.org

How dogs contribute to your health and ...
medicalnewstoday.com

Sudden Diarrhea in Dogs | PetMD | petMD
petmd.com

a day ago Dog saliva can cause serious bacterial ...
insider.com

List of Dog Breeds | Petfinder
petfinder.com

Dog E royalc

2 days ago Dogs - latest news, breaking stories ...
independent.co.uk

dog foods and heart disease
cnbc.com

Dog - Wikipedia
en.wikipedia.org

Buy Guide Dogs Victoria P ...
guidedogsvictoria.com.au

Dog disease that can jump to humans w...
bgr.com

Our partners - Royal Canin
royalcanin.com

9. Creating Deep Learning API

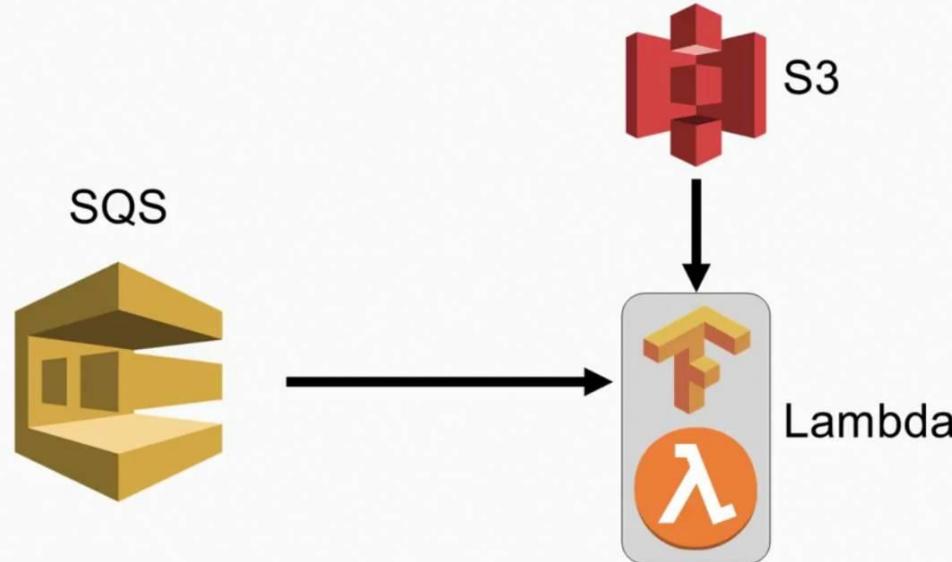
LAB 4.2

5. Now try with any image you like



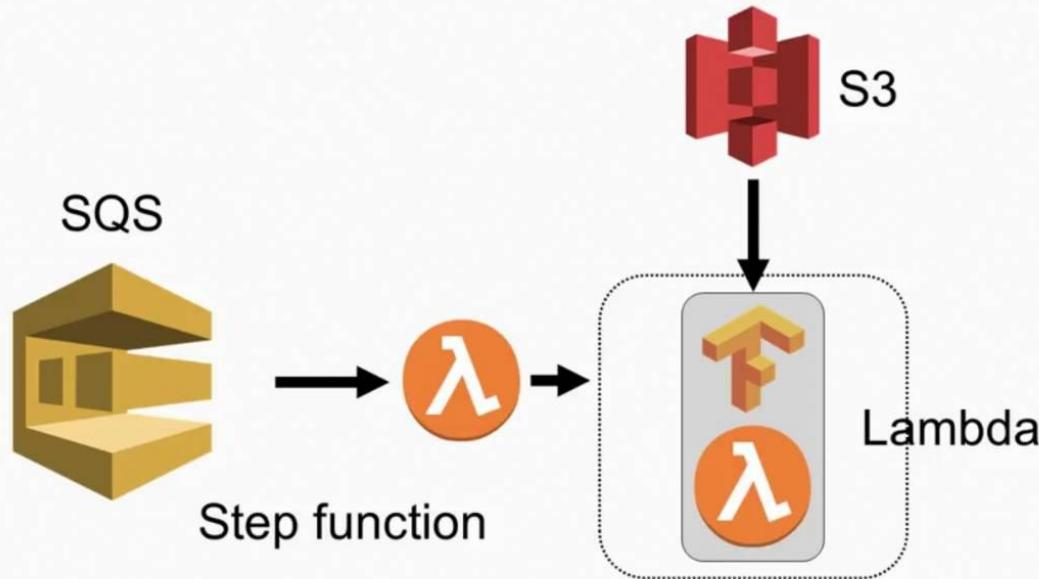
```
← → C ⌂ https://yzwhmr81n9.execute-api.ap-southeast-1.amazonaws.com/dev/hello?url=https://www.petmd.com/sites/default/files/Acute-Dog-Diarrhea-47066074.jpg
[
  "pug, pug-dog (score = 0.91775)",
  "bulldog (score = 0.00680)",
  "Brabancon griffon (score = 0.00128)",
  "French bulldog (score = 0.00121)",
  "cheeseburger (score = 0.00105)"
]
```

10. Creating Deep Learning Pipeline

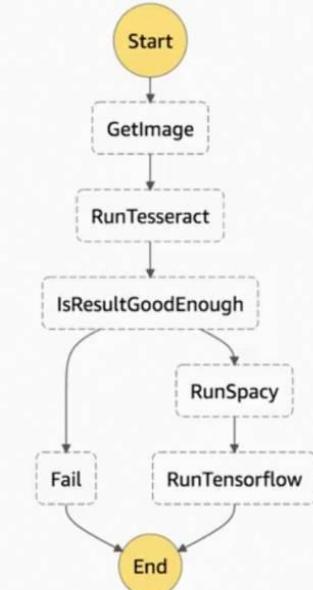


**2. Deep Learning
Batch Processing**

11. Creating Deep Learning Workflow



Deep Learning Workflow



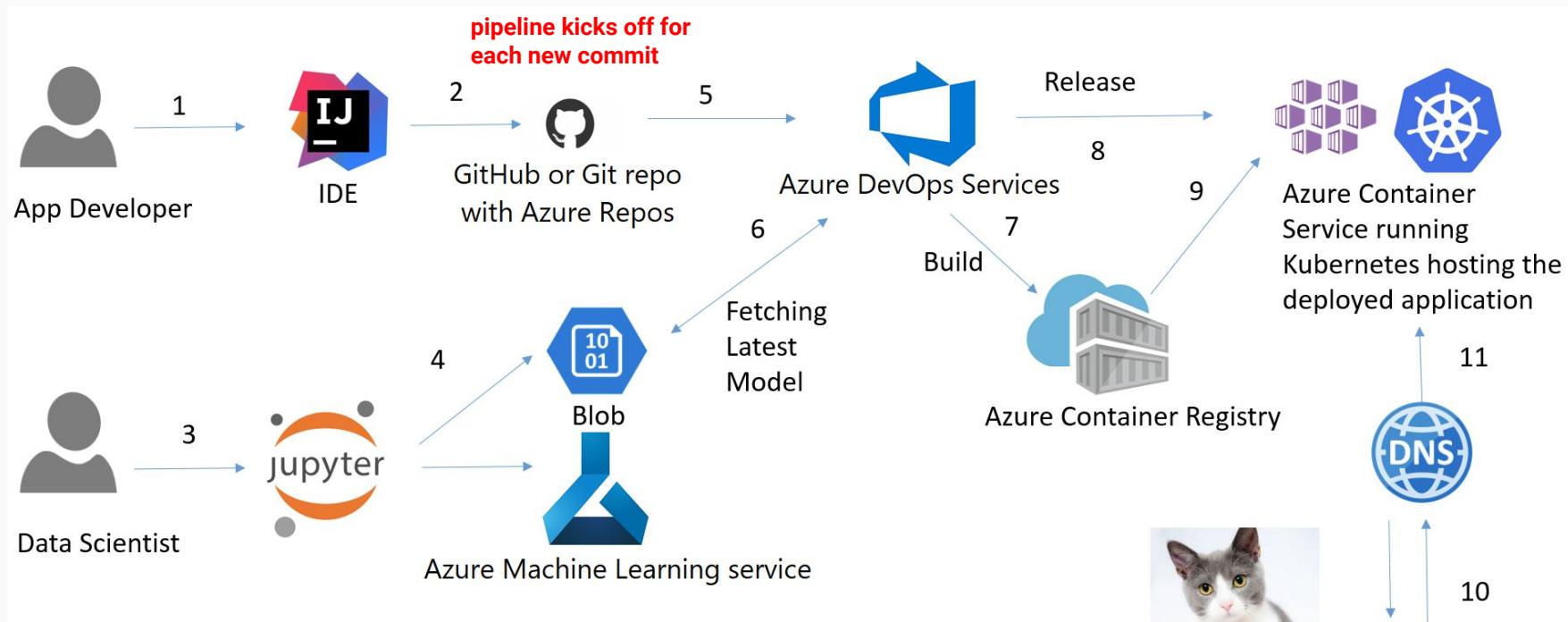
5. Azure

Clone demo repo at

<https://github.com/kwarodom/mlsruzurekubernetes>

Deploying Deep
Learning on
Kubernetes Cluster
with GPUs

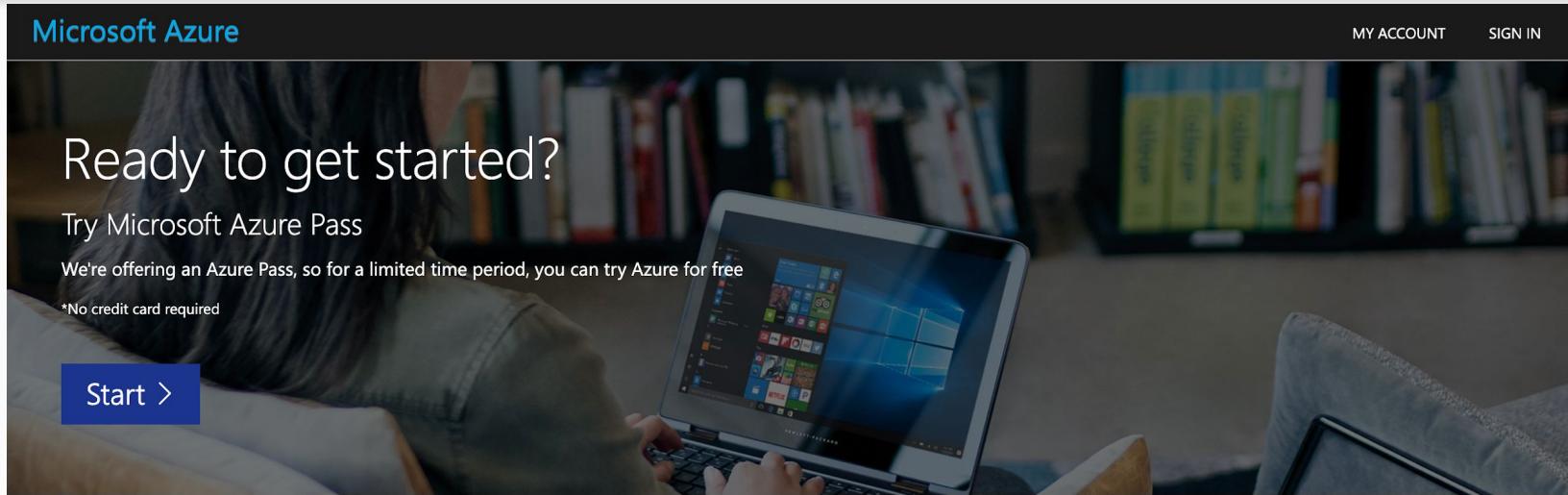
Creating CI/CD Pipeline on Azure using Docker, Kubernetes, and Python Flask



This **decouples the app developers and data scientists**, to make sure that their production app is always running the latest code with latest ML model.

[{ "cat": 0.99218,
"feline": 0.81242,
"puma": 0.45456: }]

Redeem Your Azure Credits



Microsoft Azure

MY ACCOUNT SIGN IN

Ready to get started?

Try Microsoft Azure Pass

We're offering an Azure Pass, so for a limited time period, you can try Azure for free

*No credit card required

[Start >](#)

Use the links below to learn more

[Redemption Process Guide](#)

[Azure Documentation](#)

[Explore Azure](#)

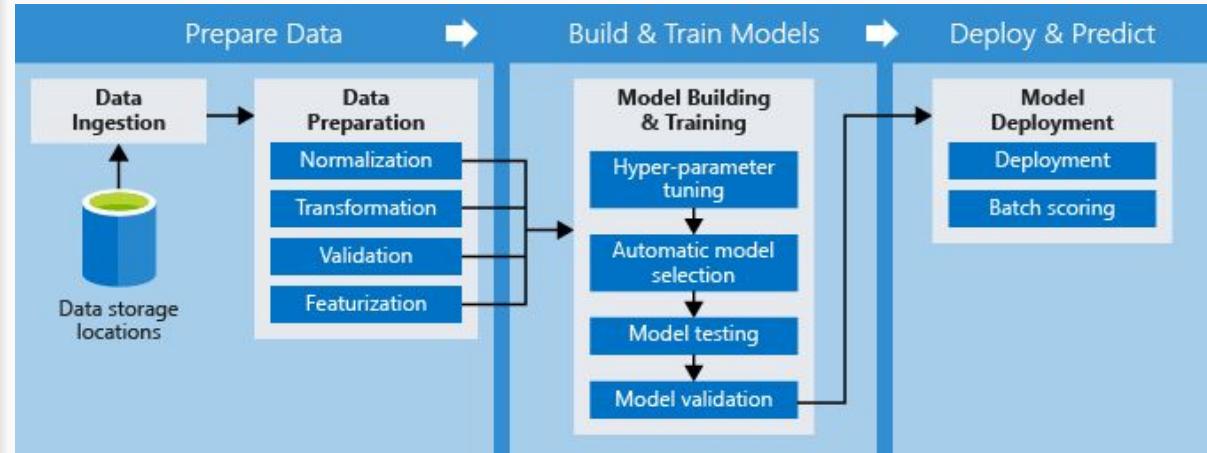
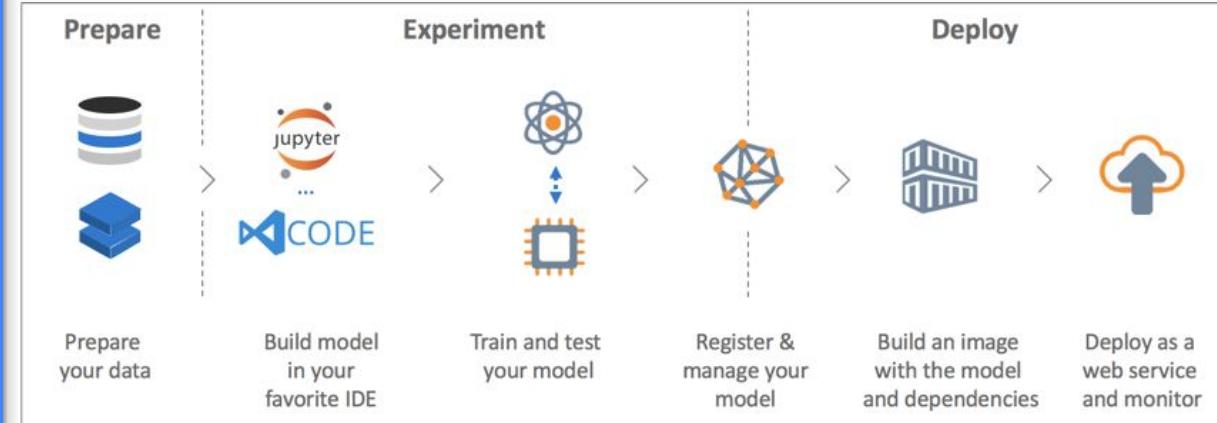
<https://www.microsoftazurepass.com>

Get your code here: **shorturl.at/evyFV**

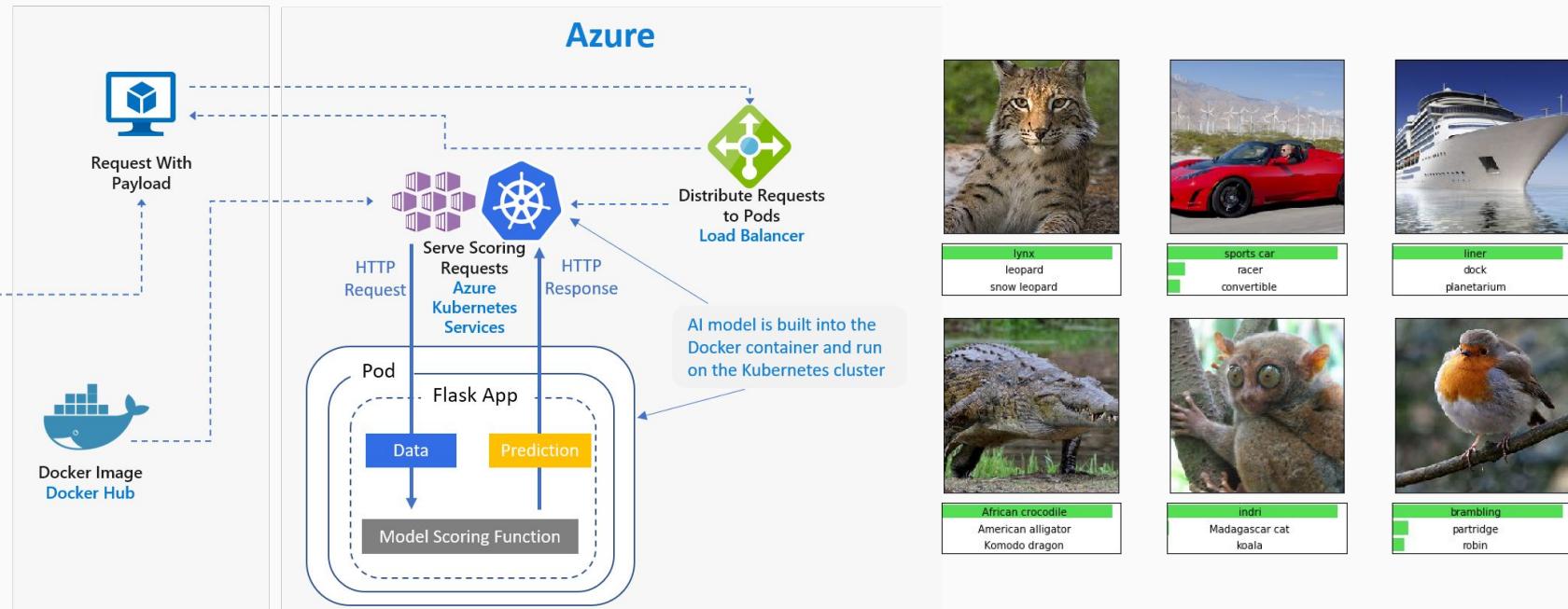
© 2019 Microsoft

Support | Privacy & Cookies | Terms of Use

Azure Machine Learning Pipeline



Deploying Deep Learning CNN on Kubernetes Cluster with GPUs



Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

For each framework, we go through the following steps:

- **Step1:** Model development where we load the pretrained model and test it by using it to score images (h5)
- **Step2:** Developing the interface our Flask app will use to load and call the model (h5>function>Flask app with dependencies)
- **Step3:** Building the Docker Image with our Flask REST API and model
- **Step4:** Testing our Docker image before deployment
- **Step5:** Creating our Kubernetes cluster and deploying our application to it
- **Step6:** Testing the deployed model
- **Step7:** Testing the throughput of our model
- **Step8:** Cleaning up resources

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Here is the application flow:

- **Step1:** The client sends a HTTP POST request with the encoded image data
- **Step2:** The Flask app extracts the image from the request.
- **Step3:** The image is then appropriately preprocessed and sent to the model for scoring.
- **Step4:** The scoring result is then piped into a JSON object and returned to the client.

NOTE: The tutorial goes through step by step how to deploy a deep learning model on Azure; it does not include enterprise best practices such as securing the endpoints and setting up remote logging etc.

Part 2: Provision Ubuntu Data Science VM

Microsoft Azure Data Science Virtual Machine for Linux (Ubuntu)

The Data Science Virtual Machine for Linux is an Ubuntu-based virtual machine image that makes it easy to get started with machine learning, including deep learning, on Azure. Deep learning tools include:

- [Caffe](#): A deep learning framework built for speed, expressivity, and modularity
- [Caffe2](#): A cross-platform version of Caffe
- [Microsoft Cognitive Toolkit](#): A deep learning software toolkit from Microsoft Research
- [H2O](#): An open-source big data platform and graphical user interface
- [Keras](#): A high-level neural network API in Python for Theano and TensorFlow
- [MXNet](#): A flexible, efficient deep learning library with many language bindings
- [NVIDIA DIGITS](#): A graphical system that simplifies common deep learning tasks
- [PyTorch](#): A high-level Python library with support for dynamic networks
- [TensorFlow](#): An open-source library for machine intelligence from Google
- [Theano](#): A Python library for defining, optimizing, and efficiently evaluating mathematical expressions involving multi-dimensional arrays
- [Torch](#): A scientific computing framework with wide support for machine learning algorithms
- CUDA, cuDNN, and the NVIDIA driver
- Many sample Jupyter notebooks

Microsoft Azure Data Science Virtual Machine for Linux (Ubuntu)

The Data Science Virtual Machine for Linux also contains popular tools for data science and development activities, including:

- Microsoft R Server Developer Edition with Microsoft R Open
- Anaconda Python distribution (versions 2.7 and 3.5), including popular data analysis libraries
- JuliaPro - a curated distribution of Julia language with popular scientific and data analytics libraries
- Standalone Spark instance and single node Hadoop (HDFS, Yarn)
- JupyterHub - a multiuser Jupyter notebook server supporting R, Python, PySpark, Julia kernels
- Azure Storage Explorer
- Azure command-line interface (CLI) for managing Azure resources
- Machine learning tools
 - [Vowpal Wabbit](#): A fast machine learning system supporting techniques such as online, hashing, allreduce, reductions, learning2search, active, and interactive learning
 - [XGBoost](#): A tool providing fast and accurate boosted tree implementation
 - [Rattle](#): A graphical tool that makes getting started with data analytics and machine learning in R easy
 - [LightGBM](#): A fast, distributed, high-performance gradient boosting framework
- Azure SDK in Java, Python, node.js, Ruby, PHP
- Libraries in R and Python for use in Azure Machine Learning and other Azure services
- Development tools and editors (RStudio, PyCharm, IntelliJ, Emacs, vim)

Create Your Data Science Virtual Machine for Linux

Setup

To set up your environment to run these notebooks, please follow these steps. They setup the notebooks to use Docker and Azure seamlessly.

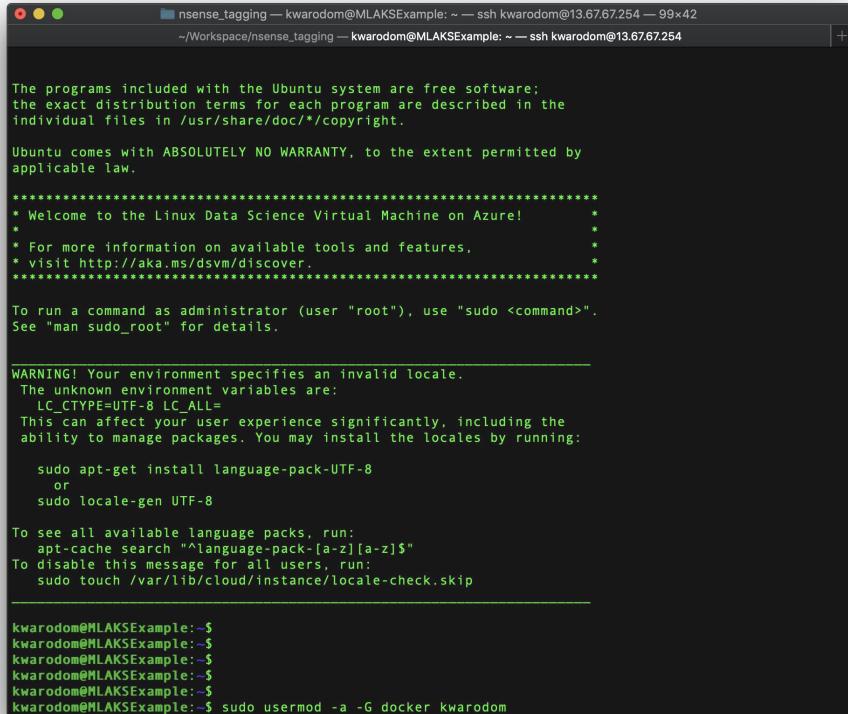
1. Create a *Linux DSVM*.

<https://docs.microsoft.com/en-us/azure/machine-learning/data-science-virtual-machine/dsvm-ubuntu-intro>

- you can go to your vm using <public ip address>:8000

2. In a bash shell on the DSVM, add your login to the docker group:

```
$ sudo usermod -a -G docker <login>
```



The screenshot shows a terminal window titled "nsense_tagging — kwarodom@MLAKSExample: ~ — ssh kwarodom@13.67.67.254 — 99x42". The window displays the output of a usermod command. It includes the standard Ubuntu license notice, a welcome message for the Data Science Virtual Machine, and a warning about invalid locales. At the bottom, it shows the command used to add the user to the docker group.

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

*****
* Welcome to the Linux Data Science Virtual Machine on Azure!
*
* For more information on available tools and features,
* visit http://aka.ms/dsvm/discover.
*****

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

WARNING! Your environment specifies an invalid locale.
The unknown environment variables are:
  LC_CTYPE=UTF-8 LC_ALL=
This can affect your user experience significantly, including the
ability to manage packages. You may install the locales by running:
  sudo apt-get install language-pack-UTF-8
  or
  sudo locale-gen UTF-8

To see all available language packs, run:
  apt-cache search "^language-pack-[a-z][a-z]$"
To disable this message for all users, run:
  sudo touch /var/lib/cloud/instance/locale-check.skip

kwarodom@MLAKSExample:~$ kwarodom@MLAKSExample:~$ kwarodom@MLAKSExample:~$ kwarodom@MLAKSExample:~$ kwarodom@MLAKSExample:~$ kwarodom@MLAKSExample:~$ kwarodom@MLAKSExample:~$ sudo usermod -a -G docker kwarodom
```

Create Your Data Science Virtual Machine for Linux

1. Go to “All resources”

The screenshot shows the Microsoft Azure 'All resources' page. On the left is a navigation sidebar with various service icons and links. The main area has a search bar at the top. Below it, the title 'All resources' is displayed with a 'Default Directory' dropdown. A red arrow points to the 'Add' button, which is located just below the title. The 'Add' button is blue with a white plus sign. To its right are 'Edit columns', 'Refresh', 'Assign tags', 'Delete', and 'Try preview' buttons. The main content area lists 8 items, each with a checkbox, name, type, resource group, location, and subscription status. The names listed are codetogetherdiag, codetogether-vnet, trial, trial_lun_0_2_f2e6e0832d7e4fb90990e2af251e23b, trial_OsDisk_1_a7768404ce2543109bf2bcc1bce4f385, trial896, trial-ip, and trial-nsg.

	NAME	TYPE	RESOURCE GROUP	LOCATION	SUBSCRIPTION
<input type="checkbox"/>	codetogetherdiag	Storage account	codetogether	Southeast Asia	(Disabled) Azure Pass - Spo...
<input type="checkbox"/>	codetogether-vnet	Virtual network	codetogether	Southeast Asia	(Disabled) Azure Pass - Spo...
<input type="checkbox"/>	trial	Virtual machine	codetogether	Southeast Asia	(Disabled) Azure Pass - Spo...
<input type="checkbox"/>	trial_lun_0_2_f2e6e0832d7e4fb90990e2af251e23b	Disk	CODETOGETHER	Southeast Asia	(Disabled) Azure Pass - Spo...
<input type="checkbox"/>	trial_OsDisk_1_a7768404ce2543109bf2bcc1bce4f385	Disk	CODETOGETHER	Southeast Asia	(Disabled) Azure Pass - Spo...
<input type="checkbox"/>	trial896	Network interface	codetogether	Southeast Asia	(Disabled) Azure Pass - Spo...
<input type="checkbox"/>	trial-ip	Public IP address	codetogether	Southeast Asia	(Disabled) Azure Pass - Spo...
<input type="checkbox"/>	trial-nsg	Network security group	codetogether	Southeast Asia	(Disabled) Azure Pass - Spo...

2. Click Add

Create Your Data Science Virtual Machine for Linux

The screenshot shows the Microsoft Azure portal interface. On the left is the navigation sidebar with various service icons. The main area shows the 'All resources' dashboard with a search bar at the top. A red arrow points from the text '3. Search for Data Science Virtual Machine for Linux (Ubuntu)' to the search input field, which contains the text 'data science'. Below the search bar is a list of resources, with the 'Data Science Virtual Machine for Linux (Ubuntu)' option highlighted.

Microsoft Azure

Dashboard > All resources > New

All resources Default Directory

Add Edit columns More

Filter by name...

NAME

- codetogetherdiag
- codetogether-vnet
- trial
- trial_lun_0_2_f2e6e0832d7e4fb909...
- trial_OsDisk_1_a7768404ce2543109b...
- trial896
- trial-ip
- trial-nsg

New

data science

Data Science Virtual Machine - Windows 2016

Geo AI Data Science VM with ArcGIS

Data Science Virtual Machine for Linux (Ubuntu)

Data Science Virtual Machine for Linux (CentOS)

Networking

Storage Web App Quickstart tutorial

Web SQL Database Quickstart tutorial

Mobile

Containers

Databases Serverless Function App Quickstart tutorial

Analytics

AI + Machine Learning Cosmos DB Quickstart tutorial

Internet of Things

Integration

Security Kubernetes Service Quickstart tutorial

Identity

Developer Tools DevOps Project Quickstart tutorial

Management Tools

Software as a Service (SaaS)

Blockchain Storage Account Quickstart tutorial

3. Search for Data Science Virtual Machine for Linux (Ubuntu)

Create Your Data Science Virtual Machine for Linux

The screenshot shows the Microsoft Azure portal's 'Create a resource' interface. The left sidebar lists services like Home, Dashboard, All services, Favorites, All resources, Resource groups, App Services, Function Apps, SQL databases, Azure Cosmos DB, Virtual machines, Load balancers, Storage accounts, Virtual networks, Azure Active Directory, Monitor, Advisor, Security Center, Cost Management + Billing, and Help + support. The main content area is titled 'Data Science Virtual Machine for Linux (Ubuntu)' and contains a detailed description of the VM, a list of installed tools, and a note about connecting via X2Go. At the bottom, there is a prominent blue 'Create' button.

4. Click Create

Create Your Data Science Virtual Machine for Linux

The screenshot shows the 'Create a virtual machine' wizard in the Microsoft Azure portal. The 'Basics' tab is selected. The 'PROJECT DETAILS' section includes fields for 'Subscription' (set to 'Azure Pass - Sponsorship'), 'Resource group' (with a dropdown menu showing 'Select existing...' and 'Create new'), 'Virtual machine name' (empty), 'Region' (set to 'Southeast Asia'), 'Availability options' (set to 'No infrastructure redundancy required'), 'Image' (set to 'Data Science Virtual Machine for Linux (Ubuntu)'), and 'Size' (set to 'Standard DS3 v2'). The 'ADMINISTRATOR ACCOUNT' section shows 'Authentication type' with 'Password' selected (indicated by a red arrow). Below it are fields for 'Username' and 'SSH public key'. At the bottom, there are buttons for 'Review + create', 'Previous', and 'Next : Disks >'.

5. Select your AZ pass subscription

6. Create new resource group

7. Name your VM

8. Choose “Password” authentication type

9. Next

Create Your Data Science Virtual Machine for Linux

Select a VM size

Browse available virtual machine sizes and their features

Search by VM size... X

Clear all filters

Size : Small (0-4) X

Generation : Current X

Family : General purpose X

Premium disk : Supported X

+ ↴ Add filter

Showing 12 of 207 VM sizes.

| Subscription: Azure Pass - Sponsorship

| Region: Southeast Asia

| Current size: Standard_DS3_v2

VM SIZE	OFFERING	FAMILY	VCPUS	RAM (GB)	DATA DISKS	MAX IOPS	TEMPORARY STORA...	PREMIUM DISK SUP...	COST/MONTH (ESTI...
B1ms	Standard	General purpose	1	2	2	800	4 GB	Yes	\$19.64
B1s	Standard	General purpose	1	1	2	400	4 GB	Yes	\$9.82
B2ms	Standard	General purpose	2	8	4	2400	16 GB	Yes	\$78.86
B2s	Standard	General purpose	2	4	4	1600	8 GB	Yes	\$39.28
B4ms	Standard	General purpose	4	16	8	3600	32 GB	Yes	\$156.98
D2s_v3	Standard	General purpose	2	8	4	3200	16 GB	Yes	\$93.00
D4s_v3	Standard	General purpose	4	16	8	6400	32 GB	Yes	\$186.00
DS1_v2	Standard	General purpose	1	3.5	4	3200	7 GB	Yes	\$58.78
DS2_v2	Standard	General purpose	2	7	8	6400	14 GB	Yes	\$117.55
DS3_v2	Standard	General purpose	4	14	16	12800	28 GB	Yes	\$235.10
DS2_v2 ⓘ	Promo	General purpose	2	7	8	6400	14 GB	Yes	\$117.55
DS3_v2 ⓘ	Promo	General purpose	4	14	16	12800	28 GB	Yes	\$235.10

Note: you can also select VM size based on your workload

Create Your Data Science Virtual Machine for Linux

The screenshot shows the Microsoft Azure portal interface for creating a virtual machine. The left sidebar contains a navigation menu with various service icons. The main area is titled 'Create a virtual machine' and is currently on the 'Basics' tab. The 'Disk Options' section shows an OS disk type set to 'Premium SSD'. The 'Data Disks' section lists one disk pre-defined by the selected image, with a 'Read-only' caching option. Below these sections are buttons for 'Create and attach a new disk' and 'Attach an existing disk'. A large red arrow points from the bottom right towards the 'Next : Networking >' button at the bottom of the page. The bottom navigation bar includes 'Review + create', 'Previous', and 'Next : Networking >'. The top navigation bar shows the current path: Dashboard > All resources > New > Data Science Virtual Machine for Linux (Ubuntu) > Create a virtual machine.

10. Next

Create Your Data Science Virtual Machine for Linux

Microsoft Azure

Search resources, services, and docs

Dashboard > All resources > New > Data Science Virtual Machine for Linux (Ubuntu) > Create a virtual machine > Create network security group

Create a virtual machine

Basics Disks Networking Management Guest config Tags Review + create

Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution. [Learn more](#)

NETWORK INTERFACE

When creating a virtual machine, a network interface will be created for you.

CONFIGURE VIRTUAL NETWORKS

* Virtual network [Create new](#)
Subnet [Create new](#)
Public IP [Create new](#)
NIC network security group [None](#) [Basic](#) [Advanced](#)
This VM image has preconfigured NSG rules

* Configure network security group [Create new](#)

Accelerated networking [On](#) [Off](#)
The selected image does not support accelerated networking.

LOAD BALANCING

You can place this virtual machine in the backend pool of an existing Azure load balancing solution. [Learn more](#)

Place this virtual machine behind an existing [Yes](#) [No](#) load balancing solution?

Review + create Previous Next : Management >

11. Configure network security group

Jupyter hub : TCP 8000
Jupyter notebook: TCP 8888
RStudioserver: TCP 8787
SSH: TCP 22
Webservice: 80

12. Click “OK”

Name: dldevops-nsgr

Inbound rules:

- 1010: JupyterHub Any Custom (TCP/8000) ✓ ...
- 1020: RStudioServer Any Custom (TCP/8787) ✓ ...
- 1030: default-allow-ssh Any SSH (TCP/22) ✓ ...

+ Add an inbound rule

Outbound rules:

- No results

+ Add an outbound rule

OK

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Dashboard > All resources > New > Data Science Virtual Machine for Linux (Ubuntu) > Create a virtual machine

Create a virtual machine

Basics Disks Networking Management **Guest config** Tags Review + create

Configure monitoring and management options for your VM.

MONITORING

Boot diagnostics On Off

OS guest diagnostics On Off

* Diagnostics storage account (new) devopsdiag733 [Create new](#)

IDENTITY

System assigned managed identity On Off

AUTO-SHUTDOWN

Enable auto-shutdown On Off

13. Next

Review + create Previous **Next : Guest config >**

Dashboard > All resources > New > Data Science Virtual Machine for Linux (Ubuntu) > Create a virtual machine

Create a virtual machine

Basics Disks Networking Management **Guest config** Review + create

Add additional configuration, agents, scripts or applications via virtual machine extensions or cloud-init.

EXTENSIONS

Extensions [Select an extension to install](#)

CLOUD INIT

Cloud init is a widely used approach to customize a Linux VM as it boots for the first time. You can use cloud-init to install packages and write files or to configure users and security. [Learn more](#)

The selected image does not support cloud init.

14. Next

Review + create Previous **Next : Tags >**

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Dashboard > All resources > New > Data Science Virtual Machine for Linux (Ubuntu) > Create a virtual machine

Create a virtual machine

Basics Disks Networking Management Guest config Tags Review + create

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. [Learn more](#)

Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

KEY	VALUE	RESOURCE TYPE
		All resources to be created

13. Next

Review + create Previous Next : Review + create >

Home > All resources > New > Data Science Virtual Machine for Linux (Ubuntu) > Create a virtual machine

Create a virtual machine

✓ Validation passed

Basics Disks Networking Management Guest config Tags Review + create

PRODUCT DETAILS

Data Science Virtual Machine for Linux (Ubuntu) Not covered by credits ⓘ 0.0000 THB/hr
by Microsoft Terms of use | Privacy policy

Standard DS3 v2 0.3160 USD/hr
by Microsoft Terms of use | Privacy policy Pricing for other VM sizes

TERMS

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace Terms](#) for additional details.

BASICS

Subscription	Azure Pass - Sponsorship
Resource group	(new) devops
Virtual machine name	dldevops
Region	Southeast Asia
Availability options	No infrastructure redundancy required
Authentication type	Password
Username	nsense

DISKS

OS disk type	Premium SSD
Use managed disks	Yes
Data disks	1

Create Previous Next Download a template for automation

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Home > CreateVm-microsoft-ads.linux-data-science-vm-ubun-20181222110349 - Overview

CreateVm-microsoft-ads.linux-data-science-vm-ubun-20181222110349 - Overview

Deployment

 Delete  Cancel  Redeploy  Refresh

Overview

 Inputs

 Outputs

 Template

... Your deployment is underway

Check the status of your deployment, manage resources, or troubleshoot deployment issues. Pin this page to your dashboard to easily find it next time.



Deployment name: CreateVm-microsoft-ads.linux-data-science-vm-ubun-20181222110349
Subscription: [Azure Pass - Sponsorship](#)
Resource group: [devops](#)

DEPLOYMENT DETAILS [\(Download\)](#)

Start time: 12/22/2018, 11:08:21 AM

Duration: 12 seconds

Correlation ID: b9f56f81-537d-4fdb-998c-19fea64e0121

RESOURCE	TYPE	STATUS	OPERATION DETAILS
No results.			

Please wait

Additional Resources



Windows Server 2016 VM
[Quickstart tutorial](#)



Cosmos DB
[Quickstart tutorial](#)



Web App
[Quickstart tutorial](#)



SQL Database
[Quickstart tutorial](#)



Storage Account
[Quickstart tutorial](#)

Helpful Links

[Get started with Azure](#) 
[Azure architecture center](#) 

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Home > CreateVm-microsoft-ads.linux-data-science-vm-ubun-20181222110349 - Overview

CreateVm-microsoft-ads.linux-data-science-vm-ubun-20181222110349 - Overview

Deployment

Overview

Inputs

Outputs

Template

Delete Cancel Redeploy Refresh

Your deployment is complete

Go to resource



Deployment name: CreateVm-microsoft-ads.linux-data-science-vm-ubun-20181222110349
Subscription: Azure Pass - Sponsorship
Resource group: devops

DEPLOYMENT DETAILS [\(Download\)](#)

Start time: 12/22/2018, 11:08:21 AM

Duration: 2 minutes 56 seconds

Correlation ID: b9f5f6f81-537d-4fdb-998c-19fea64e0121

RESOURCE	TYPE	STATUS	OPERATION DETAILS
dldevops	Microsoft.Compute/virtualMac...	OK	Operation details
dldevops589	Microsoft.Network/networkInte...	Created	Operation details
devopsdiag625	Microsoft.Storage/storageAcco...	OK	Operation details
devops-vnet	Microsoft.Network/virtualNetw...	OK	Operation details
dldevops-ip	Microsoft.Network/publicIpAdd...	OK	Operation details
dldevops-nsg	Microsoft.Network/networkSec...	OK	Operation details

15. Go to resource

Additional Resources



Windows Server 2016 VM
[Quickstart tutorial](#)



Cosmos DB
[Quickstart tutorial](#)



Web App
[Quickstart tutorial](#)



SQL Database
[Quickstart tutorial](#)



Storage Account
[Quickstart tutorial](#)

Helpful Links

[Get started with Azure](#) [Azure architecture center](#)

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Microsoft Azure

Search resources, services, and docs

nsenseai@gmail.com
DEFAULT DIRECTORY

Home > CreateVm-microsoft-ads.linux-data-science-vm-ubun-20181222110349 - Overview > dldevops

dldevops Virtual machine

Connect Start Restart Stop Capture Delete Refresh

Resource group (change)
dldevops

Status
Running

Location
Southeast Asia

Subscription (change)
Azure Pass - Sponsorship

Subscription ID
882c22a7-74db-4a55-93c0-ca32f1bc3949

Computer name
dldevops

Operating system
Linux

Size
Standard DS3 v2 (4 vcpus, 14 GB memory)

Public IP address
104.215.153.103

Virtual network/subnet
devops-vnet/default

DNS name
Configure

Tags (change)
Click here to add tags

Show data for last: 1 hour 6 hours 12 hours 1 day 7 days 30 days

CPU (average)

Network (total)

Disk bytes (total)

Disk operations/sec (average)

Percentage CPU (Avg) dldevops 0 %

Network In (Sum) dldevops 392.34 kB

Network Out (Sum) dldevops 3.98 kB

Disk Read Bytes (Sum) dldevops 0 B

Disk Write Bytes (Sum) dldevops 0 B

100/s

10:30 AM 10:45 AM 11 AM 11:15 AM

100%

80%

60%

40%

20%

0%

10:30 AM 10:45 AM 11 AM 11:15 AM

400kB

300kB

200kB

100kB

0B

10:30 AM 10:45 AM 11 AM 11:15 AM

100B

80B

60B

40B

20B

0B

10:30 AM 10:45 AM 11 AM 11:15 AM

235

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

To check your balance visit : <https://www.microsoftazuresponsorships.com/Balance>

Microsoft | Azure Sponsorships

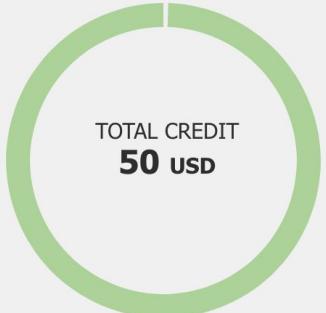
Balance

Manage

Usage

SPONSORED

✓ Active - Offer expiring in 30 days



TOTAL CREDIT
50 USD

USED
0 USD

REMAINING
50 USD

SUBSCRIPTIONS
1 ACTIVE

⌚ 22 Dec 2018 - 21 Jan 2019 *Based on usage through 12/22/2018

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

These resources will be created on the created resource group

Home > Resource groups > bkkaidedevops

bkkaidedevops Resource group

Search (Ctrl+I)

Add Edit columns Delete resource group Refresh Move Assign tags Delete

Subscription (change) Microsoft Azure Subscription ID f973e87c-e4b6-44f1-95ea-2f146d41c844 Deployments 1 Succeeded

Tags (change) Click here to add tags

Overview Activity log Access control (IAM) Tags Events

Filter by name... All types All locations No grouping

8 items Show hidden types

<input type="checkbox"/>	NAME	TYPE	LOCATION	...
<input type="checkbox"/>	bkkaidedevopsdiag	Storage account	Southeast Asia	...
<input type="checkbox"/>	bkkaidedevops-vnet	Virtual network	Southeast Asia	...
<input type="checkbox"/>	devopsvm	Virtual machine	Southeast Asia	...
<input type="checkbox"/>	devopsvm_lun_0_2_f481ade9b5574f359d1b9d212b3df331	Disk	Southeast Asia	...
<input type="checkbox"/>	devopsvm_OsDisk_1_52bfa5e49dad429db90dc28c75c7f7ce	Disk	Southeast Asia	...
<input type="checkbox"/>	devopsvm224	Network interface	Southeast Asia	...
<input type="checkbox"/>	devopsvm-ip	Public IP address	Southeast Asia	...
<input type="checkbox"/>	devopsvm-nsg	Network security group	Southeast Asia	...

Quickstart Resource costs Deployments Policies Properties Locks Automation script Monitoring Insights (preview)

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

devopsvm - Networking

Virtual machine

Search (Ctrl+ /)

Attach network interface Detach network interface

Network Interface: devopsvm224 Effective security rules Topology

Virtual network/subnet: bkkaidevops-vnet/default Public IP: 104.215.188.155 Private IP: 10.0.1.4 Accelerated networking: Disabled

APPLICATION SECURITY GROUPS

Configure the application security groups

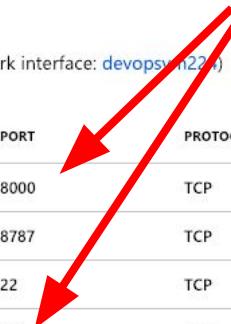
INBOUND PORT RULES

Network security group devopsvm-nsg (attached to network interface: devopsvm224)
Impacts 0 subnets, 1 network interfaces

Add inbound port rule

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION	...
1010	JupyterHub	8000	TCP	Any	Any	Allow	...
1020	RStudioServer	8787	TCP	Any	Any	Allow	...
1030	⚠ default-allow-ssh	22	TCP	Any	Any	Allow	...
1040	Port_80	80	Any	Any	Any	Allow	...
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow	...
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow	...
65500	DenyAllInBound	Any	Any	Any	Any	Deny	...

Check!!



238

Part 3: Access Your DSVM

3 Options:

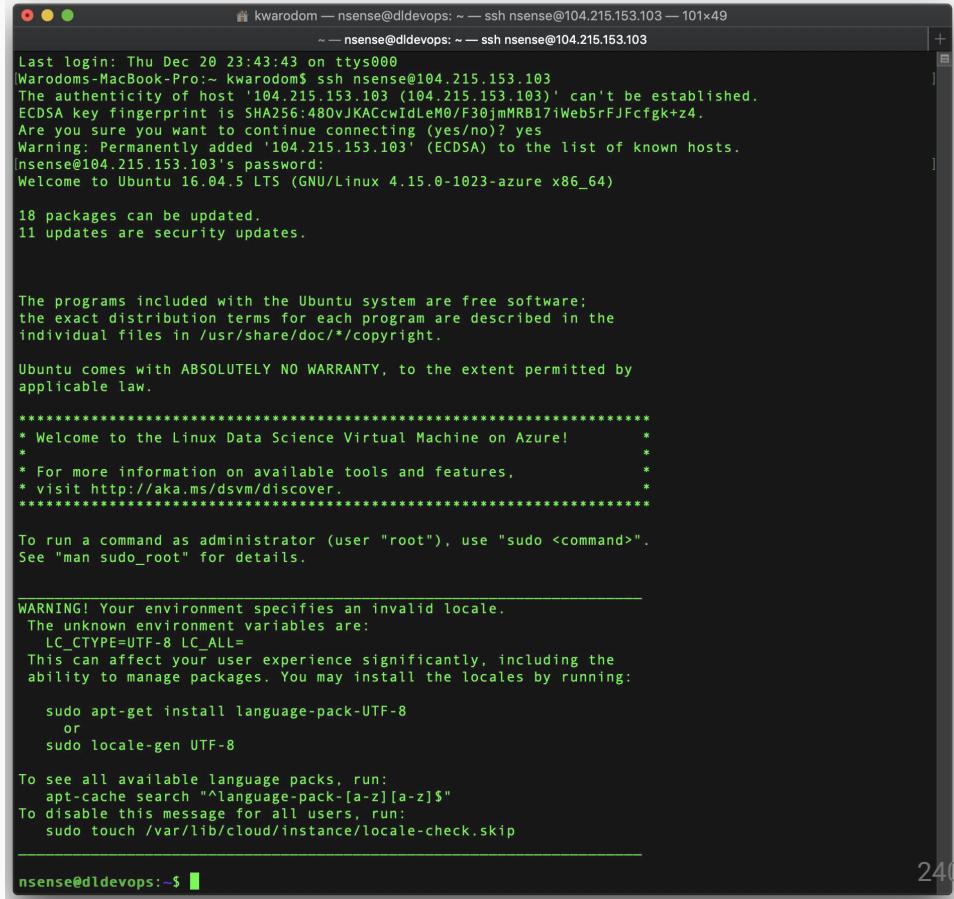
1. SSH
2. X2Go
3. JupyterHub and JupyterLab for Jupyter Notebooks

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Method 1: SSH

Step1: Open your terminal (Mac OS, Linux), Putty/Bash shell (Windows), then go to your vm using

```
$ ssh username@<public ip address>
```



A screenshot of a macOS terminal window titled "kwarodom — nsense@dldevops: ~ — ssh nsense@104.215.153.103 — 101x49". The window shows an SSH session from a MacBook Pro to a Ubuntu 16.04.5 LTS VM. The session starts with a warning about host key fingerprinting, followed by a password prompt, and a welcome message. It then displays package update information, the standard Ubuntu license notice, and a welcome message for the Azure Data Science Virtual Machine. Finally, it shows a warning about locale settings and provides instructions for resolving them.

```
Last login: Thu Dec 20 23:43:43 on ttys000
[kwarodom-MacBook-Pro:~ kwarodom]$ ssh nsense@104.215.153.103
The authenticity of host '104.215.153.103' can't be established.
ECDSA key fingerprint is SHA256:480vJKACcwIdLeM0/F30jimMRB17iWeb5rFJFcgk+z4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '104.215.153.103' (ECDSA) to the list of known hosts.
[nsense@104.215.153.103's password:
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-1023-azure x86_64)

18 packages can be updated,
11 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/**/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

*****
* Welcome to the Linux Data Science Virtual Machine on Azure!
*
* For more information on available tools and features,
* visit http://aka.ms/dsvm/discover.
*****


To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

WARNING! Your environment specifies an invalid locale.
The unknown environment variables are:
  LC_CTYPE=UTF-8  LC_ALL=
This can affect your user experience significantly, including the
ability to manage packages. You may install the locales by running:

  sudo apt-get install language-pack-UTF-8
  or
  sudo locale-gen UTF-8

To see all available language packs, run:
  apt-cache search "^language-pack-[a-z][a-z]$"
To disable this message for all users, run:
  sudo touch /var/lib/cloud/instance/locale-check.skip

nsense@dldevops:~$
```

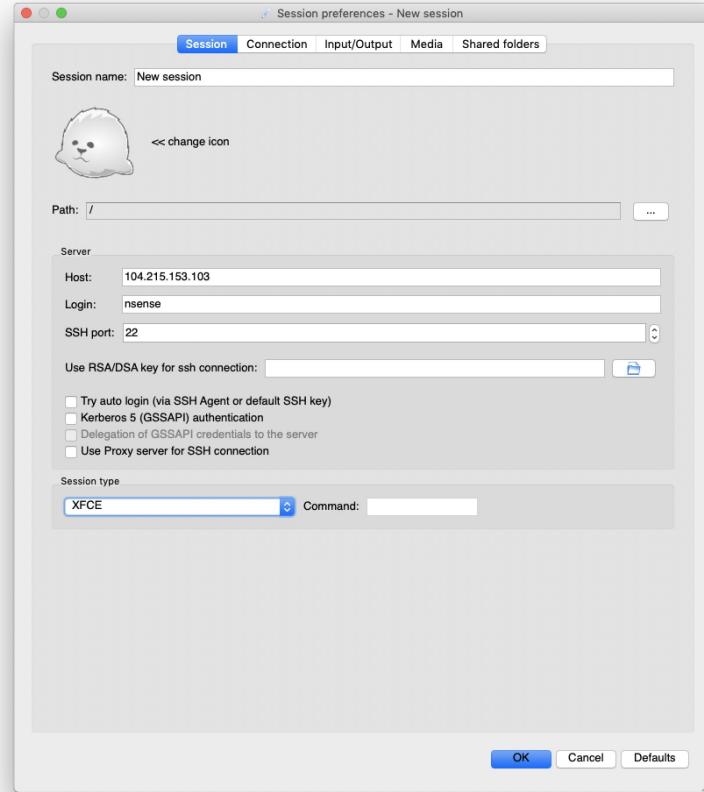
Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Method 2: X2Go

Step1: Download and install the X2Go client for your client platform from [X2Go](#).

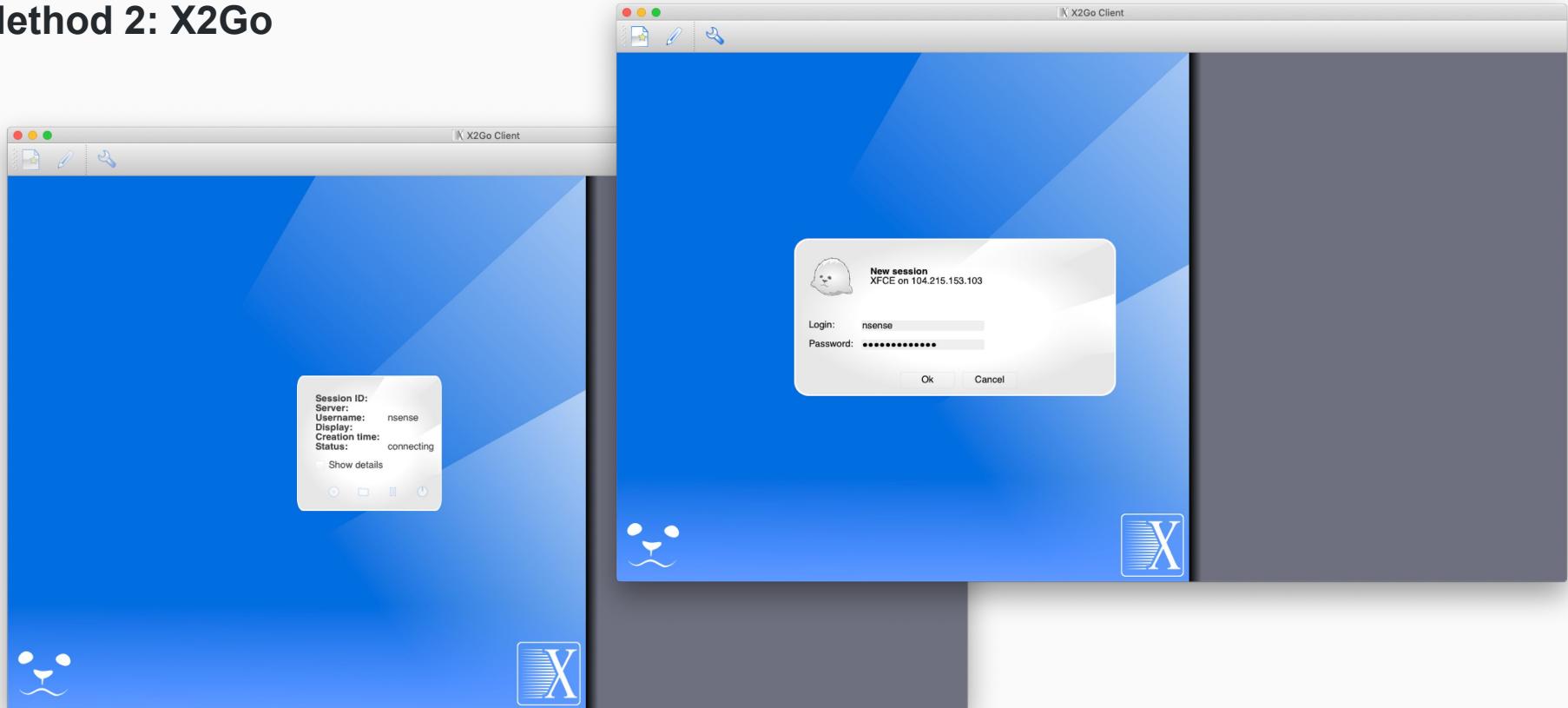
Step2: Run the X2Go client, and select New Session. It opens a configuration window with multiple tabs. Enter the following configuration parameters:

- Session tab:
 - Host: The host name or IP address of your Linux Data Science VM.
 - Login: User name on the Linux VM.
 - SSH Port: Leave it at 22, the default value.
 - Session Type: Change the value to XFCE. Currently the Linux VM only supports XFCE desktop.
- Media tab: You can turn off sound support and client printing if you don't need to use them.
- Shared folders: If you want directories from your client machines mounted on the Linux VM, add the client machine directories that you want to share with the VM on this tab



Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Method 2: X2Go

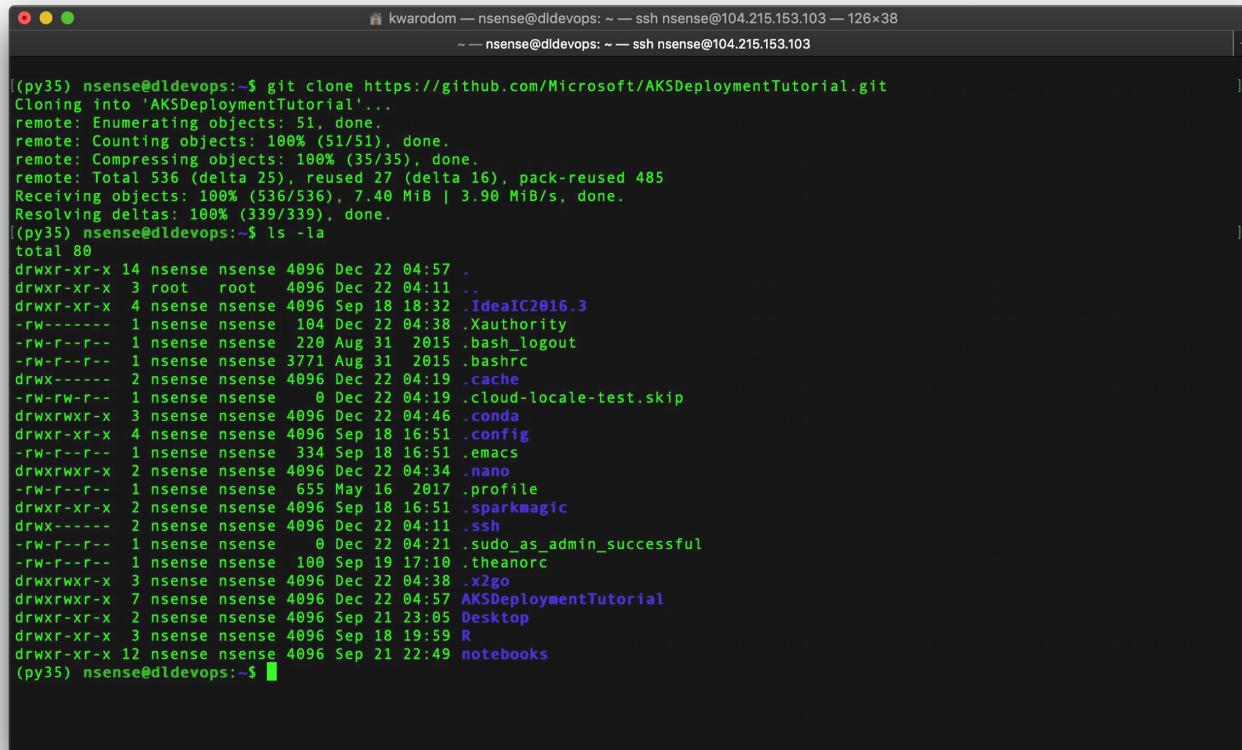


Part 4: Get started with DL on Kubernetes Cluster with GPUs - Run JupyterHub

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Step1: Clone the repo to DSVM

```
$ git clone https://github.com/kwarodom/mlrsazurekubernetes.git
```



```
kwarodom — nsense@dldevops: ~ — ssh nsense@104.215.153.103 — 126x38
~ — nsense@dldevops: ~ — ssh nsense@104.215.153.103

(py35) nsense@dldevops:~$ git clone https://github.com/Microsoft/AKSDeploymentTutorial.git
Cloning into 'AKSDeploymentTutorial'...
remote: Enumerating objects: 51, done.
remote: Counting objects: 100% (51/51), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 536 (delta 25), reused 27 (delta 16), pack-reused 485
Receiving objects: 100% (536/536), 7.40 MiB | 3.90 MiB/s, done.
Resolving deltas: 100% (339/339), done.
(py35) nsense@dldevops:~$ ls -la
total 80
drwxr-xr-x 14 nsense nsense 4096 Dec 22 04:57 .
drwxr-xr-x  3 root   root   4096 Dec 22 04:11 ..
drwxr-xr-x  4 nsense nsense 4096 Sep 18 18:32 .IdeaIC2016.3
-rw-----  1 nsense nsense 104 Dec 22 04:38 .Xauthority
-rw-r--r--  1 nsense nsense 220 Aug 31 2015 .bash_logout
-rw-r--r--  1 nsense nsense 3771 Aug 31 2015 .bashrc
drwxr----  2 nsense nsense 4096 Dec 22 04:19 .cache
-rw-rw-r--  1 nsense nsense 0 Dec 22 04:19 .cloud-locale-test.skip
drwxrwxr-x  3 nsense nsense 4096 Dec 22 04:46 .conda
drwxr-xr-x  4 nsense nsense 4096 Sep 18 16:51 .config
-rw-r--r--  1 nsense nsense 334 Sep 18 16:51 .emacs
drwxrwxr-x  2 nsense nsense 4096 Dec 22 04:34 .nano
-rw-r--r--  1 nsense nsense 655 May 16 2017 .profile
drwxr-xr-x  2 nsense nsense 4096 Sep 18 16:51 .sparkmagic
drwxr----- 2 nsense nsense 4096 Dec 22 04:11 .ssh
-rw-r--r--  1 nsense nsense 0 Dec 22 04:21 .sudo_as_admin_successful
-rw-r--r--  1 nsense nsense 100 Sep 19 17:10 .theanorc
drwxrwxr-x  3 nsense nsense 4096 Dec 22 04:38 .x2go
drwxrwxr-x  7 nsense nsense 4096 Dec 22 04:57 AKSDeploymentTutorial
drwxr-xr-x  2 nsense nsense 4096 Sep 21 23:05 Desktop
drwxr-xr-x  3 nsense nsense 4096 Sep 18 19:59 R
drwxr-xr-x 12 nsense nsense 4096 Sep 21 22:49 notebooks
(py35) nsense@dldevops:~$
```

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Step2: Login to Docker with your username and password

```
$ sudo groupadd docker  
$ sudo usermod -aG docker $USER
```

```
$ docker login
```

Step3: Go to the framework folder

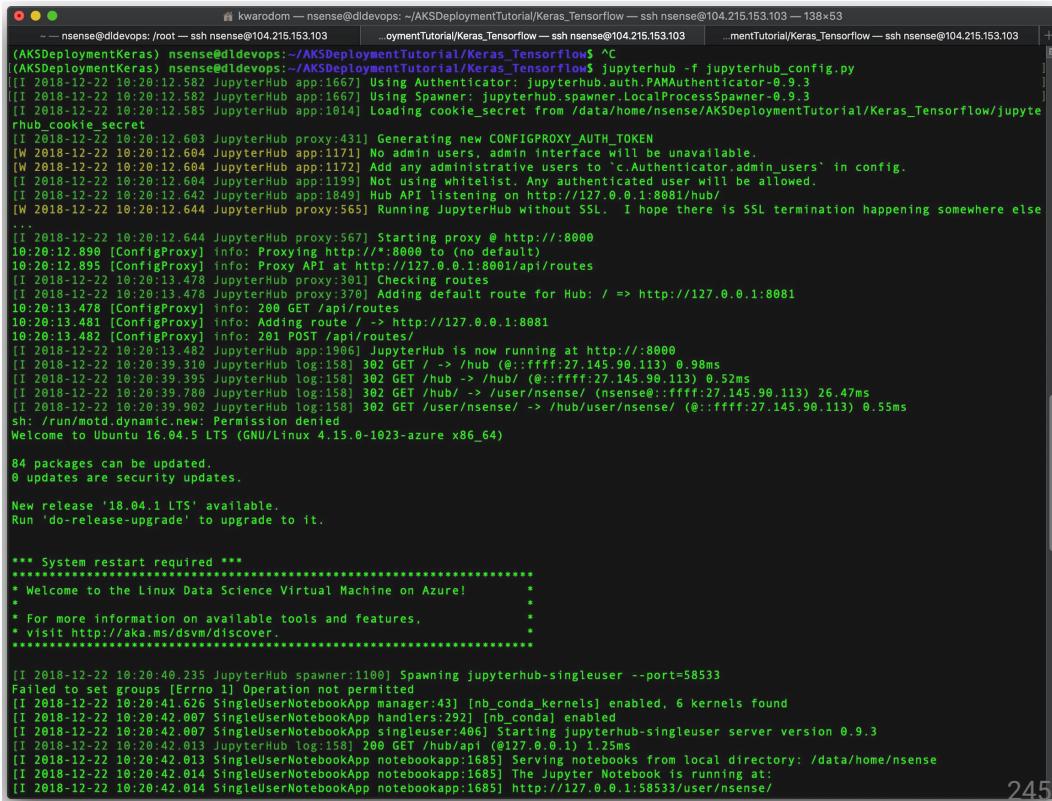
```
$ cd mlrsazurekubernetes  
$ cd Keras_Tensorflow
```

Step4: Create Conda environment

```
$ conda env create -f tutorial_env.yml  
$ conda info --envs
```

Step5: Activate environment

```
$ source activate tutorial_env
```



Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Step6: Login to Azure

```
$ az login
```

Note: if you have msi error try

```
$ sudo -i az extension remove --name azure-ml-admin-cli
```

```
$ sudo az login
```

Step7: Run

```
$ conda install -c conda-forge jupyterhub==0.9.3
```

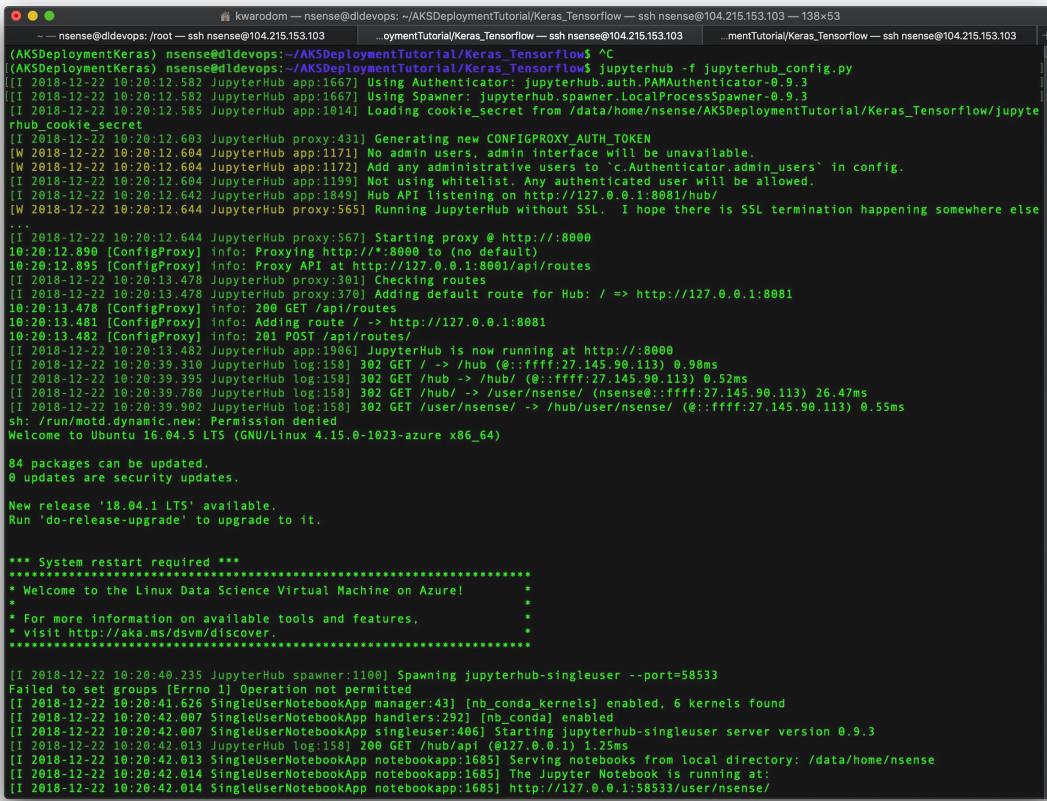
```
$ jupyterhub --generate-config
```

(Note if you have error on port 8081

```
sudo kill `sudo lsof -t -i:8081`)
```

Step8: Run

```
$ jupyterhub -f jupyterhub_config.py
```



```
kwarodom ~nsense@dldevops:~/AKSDeploymentTutorial/Keras_Tensorflow — ssh nsense@104.215.153.103 — 138x53
...nmentTutorial/Keras_Tensorflow — ssh nsense@104.215.153.103 ...mentTutorial/Keras_Tensorflow — ssh nsense@104.215.153.103 +[

(AKSDeploymentKeras) nsense@dldevops:~/AKSDeploymentTutorial/Keras_Tensorflow$ ^C
[AWSDeploymentKeras] nsense@dldevops:~/AKSDeploymentTutorial/Keras_Tensorflow$ jupyterhub_config.py
[I] 2018-12-22 10:20:12.582 JupyterHub app:1667) Using Authenticator: jupyterhub.auth.PAMAuthenticator-0.9.3
[I] 2018-12-22 10:20:12.582 JupyterHub app:1667) Using Spawner: jupyterhub.spawner.LocalProcessSpawner-0.9.3
[I] 2018-12-22 10:20:12.585 JupyterHub app:1014] Loading cookie_secret from /data/home/nsense/AKSDeploymentTutorial/Keras_Tensorflow/jupyterhub_cookie_secret
[I] 2018-12-22 10:20:12.603 JupyterHub proxy:431] Generating new CONFIGPROXY_AUTH_TOKEN
[W] 2018-12-22 10:20:12.604 JupyterHub app:1171] No admin users, admin interface will be unavailable.
[W] 2018-12-22 10:20:12.604 JupyterHub app:1172] Add any administrative users to 'c.Authenticator.admin_users' in config.
[I] 2018-12-22 10:20:12.604 JupyterHub app:1199] Not using whitelist. Any authenticated user will be allowed.
[I] 2018-12-22 10:20:12.642 JupyterHub app:1849] Hub API listening on http://127.0.0.1:8081/hub/
[W] 2018-12-22 10:20:12.644 JupyterHub proxy:565] Running JupyterHub without SSL. I hope there is SSL termination happening somewhere else
...
[I] 2018-12-22 10:20:12.644 JupyterHub proxy:567] Starting proxy @ http://:8000
[0] 2018-12-22 10:20:12.890 [ConfigProxy] Info: Proxying http://*:8000 to (no default)
[0] 2018-12-22 10:20:13.478 JupyterHub proxy:301] Checking routes
[0] 2018-12-22 10:20:13.478 JupyterHub proxy:370] Adding default route for Hub: / => http://127.0.0.1:8081
[0] 2018-12-22 10:20:13.479 [ConfigProxy] Info: 200 GET /api/routes
[0] 2018-12-22 10:20:13.481 [ConfigProxy] Info: Adding route / -> http://127.0.0.1:8081
[0] 2018-12-22 10:20:13.482 [ConfigProxy] Info: 201 POST /api/routes/
[0] 2018-12-22 10:20:13.482 JupyterHub app:1986] JupyterHub is now running at http://:8000
[0] 2018-12-22 10:20:39.310 JupyterHub log:[158] 302 GET / -> /hub (@::ffff:27.145.90.113) 0.98ms
[0] 2018-12-22 10:20:39.395 JupyterHub log:[158] 302 GET /hub -> /hub/ (@::ffff:27.145.90.113) 0.52ms
[0] 2018-12-22 10:20:39.788 JupyterHub log:[158] 302 GET /hub/ -> /user/nsense/ (/nsense@::ffff:27.145.90.113) 26.47ms
[0] 2018-12-22 10:20:39.982 JupyterHub log:[158] 302 GET /user/nsense/ -> /hub/user/nsense/ (@::ffff:27.145.90.113) 0.55ms
sh: /run/motd.dynamic: new: Permission denied
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-1023-azure x86_64)

84 packages can be updated.
0 updates are security updates.

New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
***** Welcome to the Linux Data Science Virtual Machine on Azure! *****
* For more information on available tools and features,
* visit http://aka.ms/dsvm/discover.
*****[

[I] 2018-12-22 10:20:40.235 JupyterHub spawner:1100] Spawning jupyterhub-singleuser --port=58533
Failed to set groups [Errno 1] Operation not permitted
[I] 2018-12-22 10:20:41.626 SingleUserNotebookApp manager:43] [nb_conda_kernels] enabled, 6 kernels found
[I] 2018-12-22 10:20:42.007 SingleUserNotebookApp handlers:292] [nb_conda] enabled
[I] 2018-12-22 10:20:42.007 SingleUserNotebookApp singluser:406] Starting jupyterhub-singleuser server version 0.9.3
[I] 2018-12-22 10:20:42.013 JupyterHub log:[158] 200 GET /hub/api (@127.0.0.1) 1.25ms
[I] 2018-12-22 10:20:42.013 SingleUserNotebookApp notebookapp:1685] Serving notebooks from local directory: /data/home/nsense
[I] 2018-12-22 10:20:42.014 SingleUserNotebookApp notebookapp:1685] The Jupyter Notebook is running at:
[I] 2018-12-22 10:20:42.014 SingleUserNotebookApp notebookapp:1685] http://127.0.0.1:58533/user/nsense/
```

Deploying Python Models on a Kubernetes Clusters for Real-Time Scoring

Step1: Open Browser (e.g., Chrome or Safari)

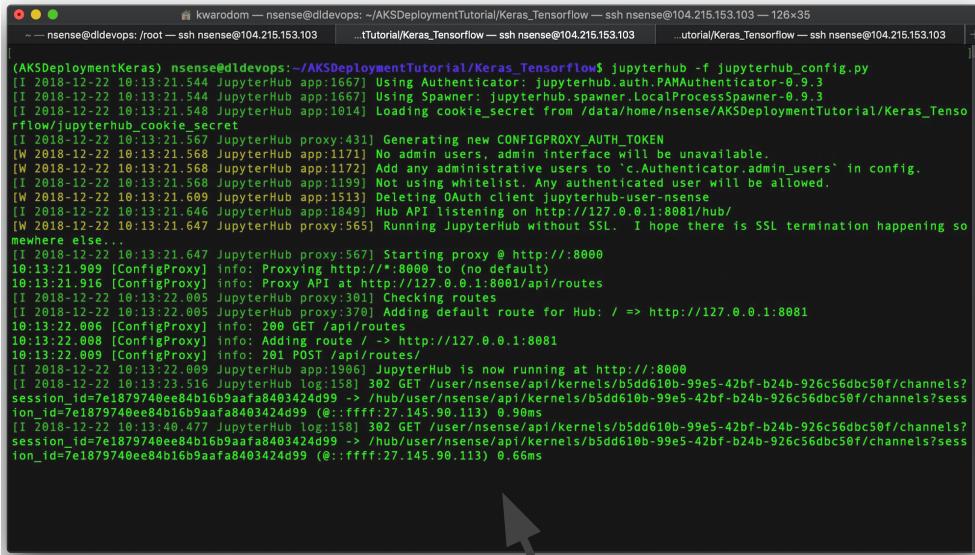
Step2: Go to “http://<your_dsvm_ip>:8000”

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** jupyter, Logout, Control Panel.
- Menu Bar:** Files, Running, Clusters, Conda.
- Toolbar:** Select items to perform actions on them, Upload, New, Refresh.
- File List:** A table showing files in the AKSDeploymentTutorial workspace.

	Name	Last Modified	File size
0	..	seconds ago	
1	Keras_Tensorflow	a minute ago	
2	Pytorch	6 hours ago	
3	static	6 hours ago	
4	Tensorflow	6 hours ago	
5	jupyterhub.sqlite	5 hours ago	102 kB
6	jupyterhub_config.py	5 hours ago	33.7 kB
7	jupyterhub_cookie_secret	5 hours ago	65 B
8	LICENSE	6 hours ago	1.18 kB
9	README.md	6 hours ago	4.82 kB

Deploying Python Models on a Kubernetes Clusters for Real-Time Scoring



```
(AKSDeploymentKeras) nsense@dldevops:~/AKSDeploymentTutorial/Keras_Tensorflow$ jupyterhub -f jupyterhub_config.py
[2018-12-22 10:13:21.544 JupyterHub app:1667] Using Authenticator: jupyterhub.auth.PAMAuthenticator-0.9.3
[2018-12-22 10:13:21.544 JupyterHub app:1667] Using Spawner: jupyterhub.spawner.LocalProcessSpawner-0.9.3
[2018-12-22 10:13:21.548 JupyterHub app:1014] Loading cookie_secret from /data/home/nsense/AKSDeploymentTutorial/Keras_Tensorflow/jupyterhub_cookie_secret
[2018-12-22 10:13:21.567 JupyterHub proxy:4311 Generating new CONFIGPROXY_AUTH_TOKEN
[W 2018-12-22 10:13:21.568 JupyterHub app:1171] No admin users, admin interface will be unavailable.
[W 2018-12-22 10:13:21.568 JupyterHub app:1172] Add any administrative users to 'c.Authenticator.admin_users' in config.
[2018-12-22 10:13:21.568 JupyterHub app:1199] Not using whitelist. Any authenticated user will be allowed.
[W 2018-12-22 10:13:21.609 JupyterHub app:1513] Deleting OAuth client jupyterhub-client-nsense
[2018-12-22 10:13:21.646 JupyterHub app:1849] Hub API listening on http://127.0.0.1:8881/hub/
[W 2018-12-22 10:13:21.647 JupyterHub proxy:565] Running JupyterHub without SSL. I hope there is SSL termination happening somewhere else.
[2018-12-22 10:13:21.647 JupyterHub proxy:567] Starting proxy @ http://:8000
10:13:21.809 [ConfigProxy] info: Proxying http://*:8000 to (no default)
10:13:21.916 [ConfigProxy] info: Proxy API at http://127.0.0.1:8001/api/routes
[2018-12-22 10:13:22.085 JupyterHub proxy:301] Checking routes
[2018-12-22 10:13:22.085 JupyterHub proxy:370] Adding default route for Hub: / => http://127.0.0.1:8081
10:13:22.086 [ConfigProxy] info: 200 GET /api/routes
10:13:22.088 [ConfigProxy] info: Adding route / -> http://127.0.0.1:8081
10:13:22.089 [ConfigProxy] info: 201 POST /api/routes/
[2018-12-22 10:13:22.089 JupyterHub app:1986] JupyterHub is now running at http://:8000
[2018-12-22 10:13:23.516 JupyterHub log:158] 302 GET /user/nsense/api/kernels/b5dd610b-99e5-42bf-b24b-926c56dbc50f/channels?session_id=7e1879740ee84b16b9aaaf8403424d99 -> /hub/user/nsense/api/kernels/b5dd610b-99e5-42bf-b24b-926c56dbc50f/channels?session_id=7e1879740ee84b16b9aaaf8403424d99 (@::ffff:27.145.99.113) 0.90ms
[2018-12-22 10:13:40.477 JupyterHub log:158] 302 GET /user/nsense/api/kernels/b5dd610b-99e5-42bf-b24b-926c56dbc50f/channels?session_id=7e1879740ee84b16b9aaaf8403424d99 -> /hub/user/nsense/api/kernels/b5dd610b-99e5-42bf-b24b-926c56dbc50f/channels?session_id=7e1879740ee84b16b9aaaf8403424d99 (@::ffff:27.145.99.113) 0.66ms
```

If you cannot login type
1. **\$ passwd**
to change UNIX password and then use it
for JupyterHub

If import tensorflow error

2. **\$ sudo apt-get install cuda-9-1**

Check if cuda driver is installed correctly

3. **\$ nvcc -V**

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

jupyter

Logout Control Panel

Files Running Clusters Conda

Select items to perform actions on them.

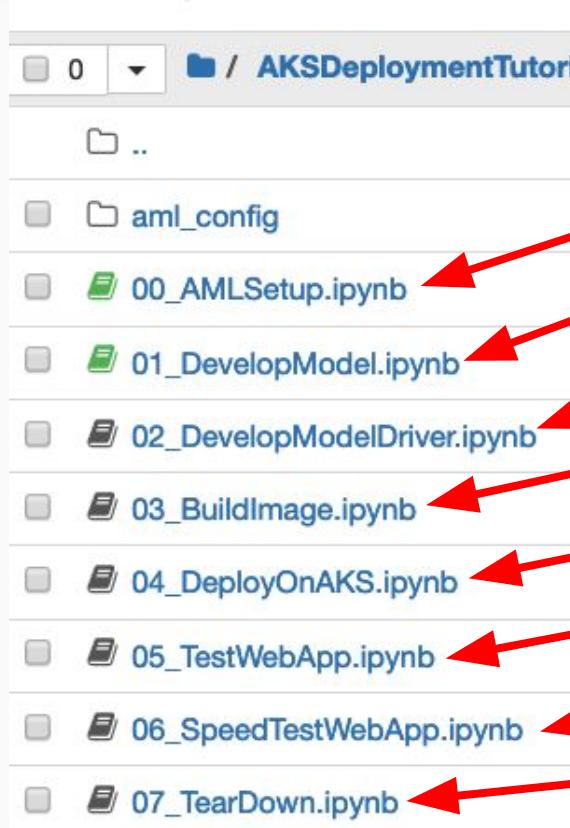
Upload New

	Name	Last Modified	File size
0	/ AKSDeploymentTutorial		
	..	seconds ago	
	Keras_Tensorflow	a minute ago	
	Pytorch	6 hours ago	
	static	6 hours ago	
	Tensorflow	6 hours ago	
	jupyterhub.sqlite	5 hours ago	102 kB
	jupyterhub_config.py	5 hours ago	33.7 kB
	jupyterhub_cookie_secret	5 hours ago	65 B
	LICENSE	6 hours ago	1.18 kB
	README.md	6 hours ago	4.82 kB

Choose “Keras_Tensorflow”

<https://resources.wolframcloud.com/NeuralNetRepository/resources/ResNet-152-Trained-on-ImageNet-Competition-Data>

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs



We will deploy ResNet 152 Model on GPU enabled Kubernetes cluster using Keras with Tensorflow

Step0: Azure Machine Learning Setup

Step1: Model Development

Step2: Developing the interface

Step3: Building the docker image

Step4: Testing our docker image

Step5: Creating our Kubenetes cluster

Step6: Testing the deployed model

Step7: Testing the throughput

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Documentation for individual models

Ref: <https://keras.io/applications/>

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	99 MB	0.749	0.921	25,636,712	168
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

IMAGENET 14,197,122 images, 21841 synsets indexed

Explore Download Challenges Publications CoolStuff About
Not logged in. Login | Signup

ImageNet is an image database organized according to the **WordNet** hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.

[Click here](#) to learn more about ImageNet, [Click here](#) to join the ImageNet mailing list.



What do these images have in common? *Find out!*

[Check out the ImageNet Challenge on Kaggle!](#)

Dataset: <http://www.image-net.org/>

© 2016 Stanford Vision Lab, Stanford University, Princeton University support@image-net.org Copyright infringement

The screenshot shows a Jupyter Notebook interface with the title "jupyter 00_AMLSetup (autosaved)". The notebook has a "Not Trusted" status and is set to "Python 3". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and various execution and presentation buttons.

Section 1: Installation and configuration

This notebook configures the notebooks in this tutorial to connect to an Azure Machine Learning (AML) Workspace. You can use an existing workspace or create a new one.

In [1]:

```
import azureml.core
from azureml.core import Workspace
from dotenv import set_key, get_key, find_dotenv
from pathlib import Path
```

Section 2: Prerequisites

If you have already completed the prerequisites, you can execute following command to ensure you are using correct conda environment. The output of this command should contain "tutorial_env" in the path, e.g. /anaconda/envs/tutorial_env/bin/python

In [2]:

```
!which python
```

/anaconda/envs/tutorial_env/bin/python

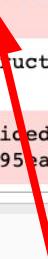
mlaks

mlaksworkspace

```
In [8]: parameter x TODO x
resource_group = '<YOUR_RESOURCE_GROUP>' e.g. myamlrg
workspace_name = '<YOUR_WORKSPACE_NAME>' # e.g. myamlworkspace
workspace_region = '<YOUR_WORKSPACE_REGION>' # e.g. eastus2
```

southeastasia

```
In [*]:  
  
# import the Workspace class and check the azureml SDK version  
from azureml.core import Workspace  
  
ws = Workspace.create(name = workspace_name,  
                      subscription_id = subscription_id,  
                      resource_group = resource_group,  
                      location = workspace_region,  
                      create_resource_group=True,  
                      exist_ok=True)  
# persist the subscription id, resource group name, and workspace name in aml_config/config.json.  
ws.write_config()  
  
Falling back to use azure cli credentials. This fall back to use azure cli credentials will be removed in the next release.  
Make sure your code doesn't require 'az login' to have happened before using azureml-sdk, except the case when you are specifying AzureCliAuthentication in azureml-sdk.  
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code H87ZJW5WS to authenticate.  
  
Performing interactive authentication. Please follow the instructions on the terminal.  
Interactive authentication successfully completed.  
  
UserWarning: The resource group doesn't exist or was not provided. AzureML SDK is creating a resource group=mlaks in location=southeastasia using subscription=f973e87c-e4b6-44f1-95ea-2f146d41c844.
```



open web browser to authen

Deploying DL on Kubernetes Cluster with GPUs: 00_AMLSetup

4

The screenshot shows the Azure portal interface for the 'mlaks' resource group. On the left, there's a navigation pane with 'Resource groups' and a search bar. The main area displays the 'mlaks' resource group details, including its subscription information (Microsoft Azure, Subscription ID: f973e87c-e4b6-44f1-95ea-2f146d41c844), deployment status (1 Succeeded), and tags (creationTime : 1549700704.718854, creationSource : azureml-sdk). Below this, a table lists the resources within the group:

NAME	TYPE	LOCATION	...
mlaksworacrdoubkpty	Container registry	Southeast Asia	...
mlaksworinsightsdwmmvzly	Application Insights	Southeast Asia	...
mlaksworkeyvaultgbfmzgk	Key vault	Southeast Asia	...
mlaksworkspace	Machine Learning service wor...	Southeast Asia	...
mlaksworstoragejukhphon	Storage account	Southeast Asia	...

Check if all of these resources are created

Deploying DL on Kubernetes Cluster with GPUs: 01_Develop Model

1

In [1]:

```
import PIL
import numpy as np
from PIL import Image, ImageOps
import wget
from resnet152 import ResNet152
from keras.preprocessing import image
from keras.applications.imagenet_utils import preprocess_input, decode_predictions
from azureml.core import Workspace
from azureml.core.compute import AksCompute, ComputeTarget
from azureml.core.webservice import Webservice, AksWebservice
from azureml.core.model import Model

from dotenv import set_key, get_key, find_dotenv
    file "/anaconda/envs/tutorial_env/lib/python3.6/site-packages/tensorflow/python/pywrap_tensorflow_internal.py", 1
ine 28, in <module>
    _pywrap_tensorflow_internal = swig_import_helper()
    File "/anaconda/envs/tutorial_env/lib/python3.6/site-packages/tensorflow/python/pywrap_tensorflow_internal.py", 1
ine 24, in swig_import_helper
    _mod = imp.load_module('_pywrap_tensorflow_internal', fp, pathname, description)
    File "/anaconda/envs/tutorial_env/lib/python3.6/imp.py", line 243, in load_module
        return load_dynamic(name, filename, file)
    File "/anaconda/envs/tutorial_env/lib/python3.6/imp.py", line 343, in load_dynamic
        return _load(spec)
ImportError: libcublas.so.9.0: cannot open shared object file: No such file or directory
```

Failed to load the native TensorFlow runtime.

See https://www.tensorflow.org/install/install_sources#common_installation_problems

for some common reasons and solutions. Include the entire stack trace
above this error message when asking for help.

If you find this error, run >
\$ sudo apt-get install cuda-9-1

Deploying DL on Kubernetes Cluster with GPUs: 01_Develop Model

2

```
~ -- ssh kwarodom@13.67.67.254
cuda-misc-headers-9-0 cuda-npp-9-0 cuda-npp-dev-9-0 cuda-nvgraph-9-0 cuda-nvgraph-dev-9-0 cuda-nvml-dev-9-0 cuda-nvrtc-9-0
cuda-nvrtc-dev-9-0 cuda-runtime-9-0 cuda-samples-9-0 cuda-toolkit-9-0 cuda-visual-tools-9-0
The following NEW packages will be installed:
cuda-9-0 cuda-command-line-tools-9-0 cuda-core-9-0 cuda-cUBLAS-dev-9-0 cuda-cuDART-9-0 cuda-cuDART-dev-9-0 cuda-cUFFT-9-0
cuda-cUFFT-dev-9-0 cuda-curand-9-0 cuda-curand-dev-9-0 cuda-cuSOLVER-9-0 cuda-cuSOLVER-dev-9-0 cuda-cuSPARSE-9-0
cuda-cuSPARSE-dev-9-0 cuda-demo-suite-9-0 cuda-documentation-9-0 cuda-driver-dev-9-0 cuda-libraries-9-0 cuda-libraries-dev-9-0
cuda-misc-headers-9-0 cuda-npp-9-0 cuda-npp-dev-9-0 cuda-nvgraph-9-0 cuda-nvgraph-dev-9-0 cuda-nvml-dev-9-0 cuda-nvrtc-9-0
cuda-nvrtc-dev-9-0 cuda-runtime-9-0 cuda-samples-9-0 cuda-toolkit-9-0 cuda-visual-tools-9-0
0 upgraded, 31 newly installed, 0 to remove and 86 not upgraded.
Need to get 1074 MB of archives.
After this operation, 2271 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-misc-headers-9-0 9.0.176-1 [684 kB]
Get:2 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-core-9-0 9.0.176.3-1 [16.9 MB]
Get:3 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-cuDART-9-0 9.0.176-1 [106 kB]
Get:4 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-driver-dev-9-0 9.0.176-1 [10.9 kB]
Get:5 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-cuDART-dev-9-0 9.0.176-1 [767 kB]
Get:6 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-command-line-tools-9-0 9.0.176-1 [25.4 MB]
Get:7 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-nvRTC-9-0 9.0.176-1 [6348 kB]
Get:8 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-nvRTC-dev-9-0 9.0.176-1 [9334 B]
Get:9 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-cuSOLVER-9-0 9.0.176-1 [26.2 MB]
Get:10 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-cuSOLVER-dev-9-0 9.0.176-1 [5317 kB]
Get:11 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-cUBLAS-dev-9-0 9.0.176.4-1 [51.3 MB]
Get:12 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-cUFFT-9-0 9.0.176-1 [84.1 MB]
Get:13 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-cUFFT-dev-9-0 9.0.176-1 [73.7 MB]
Get:14 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-curand-9-0 9.0.176-1 [38.8 MB]
Get:15 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-curand-dev-9-0 9.0.176-1 [57.9 MB]
Get:16 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-cuSPARSE-9-0 9.0.176-1 [25.2 MB]
Get:17 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-cuSPARSE-dev-9-0 9.0.176-1 [25.3 MB]
Get:18 http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64 cuda-npp-9-0 9.0.176-1 [46.6 MB]
44% [18 cuda-npp-9-0 22.8 kB/46.6 MB 0%]
53.4 MB/s 11s 8
```

Create the model

In [5]:

```
# If you see error msg "InternalError: Dst tensor is not initialized.", it indicates there are not enough memory.  
model = ResNet152(weights='imagenet')  
print("model loaded")
```

```
Downloading data from https://github.com/adamcasson/resnet152/releases/download/v0.1/resnet152_weights_tf.h5  
243187712/243179624 [=====] - 20s 0us/step  
model loaded
```

In [4]:

```
parameters ✖  
  
_MODEL_NAME = '<YOUR_MODEL_NAME>' # e.g. _MODEL_NAME = 'resnet_model'
```



resnet_model

Register the model

Register an existing trained model, add description and tags.

In [13]:

```
# Get workspace
# Load existing workspace from the config file info.
from azureml.core.workspace import Workspace

ws = Workspace.from_config()
print(ws.name, ws.resource_group, ws.location, ws.subscription_id, sep = '\n')

Found the config file in: /data/home/kwarodom/AKSDeploymentTutorialAML/Keras_Tensorflow/aml_config/config.json
mlaksworkspace
mlaks
southeastasia
f973e87c-e4b6-44f1-95ea-2f146d41c844
```

In [*]:

```
model.save_weights("model_resnet_weights.h5")
```



Prepare to register this model to AZ ML service workspace

Home > Resource groups > mlaks > mlaksworkspace > mlaksworkspace

mlaksworkspace
Machine Learning service workspace

Experiments Pipelines Compute **Models** Images Deployments Activities

Models

Refresh Create Image Add Model Delete

<input type="checkbox"/> NAME	VERSION	DESCRIPTION	CREATED ON
<input type="checkbox"/> resnet_model	1	resnet 152 model	02/09/2019, 9:28:18 AM UTC

[Back](#) [Next](#)

Go back to AZ web portal to check the registered model



resnet_model

Back to Models Refresh Create Image Delete Get Link

Details Deployments

ATTRIBUTES	
Version	1
ID	resnet_model:1
Date Registered	02/09/2019, 9:28:18 AM UTC
Location	aml://asset/83ccac5bdb
Description	resnet 152 model
Tags	model : dl, framework : resnet

Develop Model Driver

In this notebook, we will develop the API that will call our model. This module initializes the model, transforms the input so that it is in the appropriate format and defines the scoring method that will produce the predictions. The API will expect the input to be in JSON format. Once a request is received, the API will convert the json encoded request body into the image format. There are two main functions in the API: init() and run(). The init() function loads the model and returns a scoring function. The run() function process the images and uses the first function to score them.

Note: Always make sure you don't have any lingering notebooks running (Shutdown previous notebooks). Otherwise it may cause GPU memory issue.

```
In [1]: from azureml.core import Workspace
from azureml.core.compute import AksCompute, ComputeTargetException
from azureml.core.webservice import Webservice, AksWebservice
from azureml.core.image import Image
from azureml.core.model import Model
from dotenv import set_key, get_key, find_dotenv
import logging
from testing_utilities import img_url_to_json
```

```
In [2]: import keras
import tensorflow
print("Keras: ", keras.__version__)
print("Tensorflow: ", tensorflow.__version__)
```

Using TensorFlow backend.

```
Keras: 2.2.0
Tensorflow: 1.10.0
```

```
In [3]: env_path = find_dotenv(raise_error_if_not_found=True)
```

```
In [16]: _MODEL_NAME = get_key(env_path, 'model_name')
_MODEL_NAME
```

```
Out[16]: 'resnet_model'
```



This will load the model you saved in AZ ML workspace 262

jupyter 03_BuildImage (unsaved changes)

Logout Control Panel

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Build Image

In this notebook, we show the following steps for deploying a web service using AML:

- Create an image
- Test image locally

In [1]:

```
import os
import json
import docker
import requests
import numpy as np
import matplotlib.pyplot as plt
from azureml.core import Workspace
from azureml.core.compute import AksCompute, ComputeTarget
from azureml.core.webservice import Webservice, AksWebservice
from azureml.core.image import Image
from azureml.core.model import Model
from azureml.core.conda_dependencies import CondaDependencies
from azureml._model_management._util import (get_docker_client, pull_docker_image, get_docker_port,
                                             container_scoring_call, cleanup_container)
from azureml._model_management._constants import MMS_WORKSPACE_API_VERSION
from testing_utilities import to_img, img_url_to_json, plot_predictions
from dotenv import set_key, get_key, find_dotenv
```

Deploying DL on Kubernetes Cluster with GPUs: 03_Build Image

2

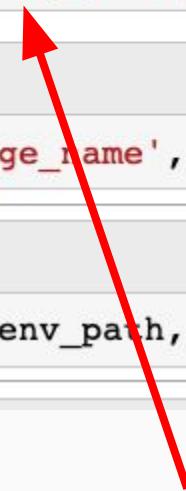
In [2]:

```
env_path = find_dotenv(raise_error_if_not_found=True)
```

In [3]:

parameters ✘ TODO ✘

```
image_name = '<YOUR_IMAGE_NAME>' # image_name = "image1"
```



In []:

```
set_key(env_path, 'image_name', image_name)
```

In [6]:

```
_MODEL_NAME = get_key(env_path, 'model_name')
```

mlaksimage

mlaksimage1

[Back to Images](#) [Refresh](#) [Create Deployment](#) [Delete](#) [Get Link](#)

Details [Models](#) [Deployments](#)

ATTRIBUTES

Description

Image for AKS Deployment Tutorial

ID

mlaksimage1:1

Date Registered

02/09/2019, 10:01:13 AM UTC

Version

1

Location

mlaksworacrdoubkpyt.a 

Environment

Docker

Type

Running

Status

name : AKS, project : AML

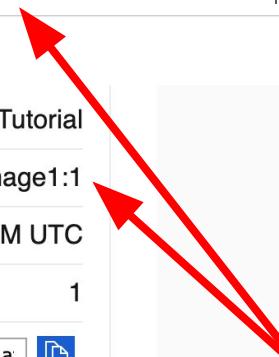
mlaksimage1

[Back to Images](#) [Refresh](#) [Create Deployment](#) [Delete](#) [Get Link](#)

[Details](#) [Models](#) [Deployments](#)

NAME	VERSION	DESCRIPTION
resnet_model	1	resnet 152 model

[Back](#) [Next](#)



**Your image with loaded
model will get created**

Deploying DL on Kubernetes Cluster with GPUs: 03_Build Image

4

In [17]:

```
# create image. It may take upto 15-20 minutes.
image = ContainerImage.create(name = image_name,
                               # this is the model object
                               models = [ws.models[_MODEL_NAME]],
                               image_config = image_config,
                               workspace = ws)

image.wait_for_creation(show_output = True)
```

```
Creating image
Runnin
g.....
```

```
SucceededImage creation operation finished for image mlaksimage1:1, operation "Succeeded"
```

mlaksworkspace
Machine Learning service workspace

Experiments Pipelines Compute Models Images Deployments Activities

Images

Refresh Create Deployment Delete

NAME	VERSION	DESCRIPTION	IMAGE TYPE	STATE
mlaksimage1	1	Image for AKS Deployment Tutorial	Docker	Succeeded

Back Next

Wait until you get succeeded

In [12]:

```
# make sure port 80 is not occupied
container_labels = {'containerName': 'kerasgpu'}
container = dc.containers.run(image.image_location,
                             detach=True,
                             ports={'5001/tcp': 80},
                             labels=container_labels,
                             runtime='nvidia')

-----
HTTPError                                                 Traceback (most recent call last)
/anaconda/envs/tutorial_env/lib/python3.6/site-packages/docker/api/client.py in _raise_for_status(self, response)
    255     try:
--> 256         response.raise_for_status()
    257     except requests.exceptions.HTTPError as e:

/anaconda/envs/tutorial_env/lib/python3.6/site-packages/requests/models.py in raise_for_status(self)
    939     if http_error_msg:
--> 940         raise HTTPError(http_error_msg, response=self)
    941

HTTPError: 500 Server Error: Internal Server Error for url: http+docker://localhost/v1.38/containers/9d952e2b99efef14d3272f6a36e65cfea4f322d9822251aa3fbf119ac566baa36/start

During handling of the above exception, another exception occurred:

APIError                                                 Traceback (most recent call last)
<ipython-input-12-8e9f808c29f2> in <module>
      5
      6
----> 7                                         ports={'5001/tcp': 80},
      8                                         labels=container_labels,
      9                                         runtime='nvidia' )
```

Error!!
You need to update
nvidia driver



In [3]: parameters x TODO x

```
aks_service_name = '<YOUR_AKS_SERVICE_NAME>' # aks_service_name = "my-aks-service-1"
aks_name = '<YOUR_AKS_NAME>' # aks_name = "my-aks-gpu1"
aks_location = '<YOUR_AKS_LOCATION>' # aks_location = "eastus"
```

my-aks-service-1

my-aks-gpu1

southeastasia

Add “sudo”,

In [10]:

```
results = subprocess.run(["sudo",
    "az", "vm", "list-usage",
    "--location", get_key(env_path, "aks_location"),
    "--query", "[?contains(localName, '%s')].{max:limit, current:currentValue}" % (vm_family)
], stdout=subprocess.PIPE)
quota = json.loads(''.join(results.stdout.decode('utf-8'))))
diff = int(quota[0]['max']) - int(quota[0]['current'])
```

In [30]:

```
vm_dict = {  
    "NV": {  
        "size": "Standard_NV6",  
        "cores": 6  
    }  
}
```

Change to “Standard_NV6”

In [31]:

```
vm_family = "NV"  
node_count = 1  
requested_cores = node_count * vm_dict[vm_family]["cores"]
```

In [32]:

```
results = subprocess.run(["sudo",  
    "az", "vm", "list-usage",  
    "--location", get_key(env_path, "aks_location"),  
    "--query", "[?contains(localName, '%s')].{max:limit, current:currentValue}" % (vm_family)  
, stdout=subprocess.PIPE)  
quota = json.loads(''.join(results.stdout.decode('utf-8')))  
diff = int(quota[0]['max']) - int(quota[0]['current'])
```

In [33]:

```
if diff <= requested_cores:  
    print("Not enough cores of NC6 in region, asking for {} but have {}".format(requested_cores, diff))  
    raise Exception("Core Limit", "Note enough cores to satisfy request")  
print("There are enough cores, you may continue...")
```

There are enough cores, you may continue...

Deploying DL on Kubernetes Cluster with GPUs: 04_DeployOnAKS

3

Subscription ([change](#))
Microsoft Azure

Subscription ID
f973e87c-e4b6-44f1-95ea-2f146d41c844

Tags ([change](#))

creationTime : **1549700704.7188854** creationSource : **azureml-sdk**

Filter by name... All types All locations No grouping

6 items Show hidden types [i](#)

<input type="checkbox"/> NAME ↑↓	TYPE ↑↓	LOCATION ↑↓	...
<input type="checkbox"/> mlaksworacrdoubkpyt	Container registry	Southeast Asia	...
<input type="checkbox"/> mlaksworinsightsdwmmvzly	Application Insights	Southeast Asia	...
<input type="checkbox"/> mlaksworkeyvaultgbfmzglk	Key vault	Southeast Asia	...
<input type="checkbox"/> mlakworkspace	Machine Learning service wor...	Southeast Asia	...
<input type="checkbox"/> mlaksworstoragejukhphon	Storage account	Southeast Asia	...
<input type="checkbox"/> my-aks-gpu2f3673002ac	Kubernetes service	Southeast Asia	...

You will see the Kubernetes service has been created

In [35]:

```
%%time
aks_target.wait_for_completion(show_output = True)
print(aks_target.provisioning_state)
print(aks_target.provisioning_errors)
```

```
Creatin
```

```
g.....
```

```
SucceededProvisioning operation finished, operation "Succeeded"
```

```
Succeeded
```

```
None
```

```
CPU times: user 2.02 s, sys: 115 ms, total: 2.14 s
```

```
Wall time: 11min 39s
```



This takes around 11-12 mins

Deploy web service to AKS¶

In [37]:

```
#Deploy web service to AKS
#Set the web service configuration (using customized configuration)
aks_config = AksWebservice.deploy_configuration(autoscale_enabled=False, num_replicas=1)
```

In [39]:

```
# get the image built in previous notebook
image_name = get_key(env_path, 'image_name')
image = ws.images[image_name]
```

In [*]:

```
%%time

aks_service = Webservice.deploy_from_image(workspace = ws,
                                             name = aks_service_name,
                                             image = image,
                                             deployment_config = aks_config,
                                             deployment_target = aks_target)
aks_service.wait_for_deployment(show_output = True)
print(aks_service.state)
```

Creating service



Check AZ portal for the created web service

In [40]:

```
%time

aks_service = Webservice.deploy_from_image(workspace = ws,
                                             name = aks_service_name,
                                             image = image,
                                             deployment_config = aks_config,
                                             deployment_target = aks_target)
aks_service.wait_for_deployment(show_output = True)
print(aks_service.state)
```

```
Creating service
Running.....  
SucceededAKS service creation operation finished, operation "Succeeded"
Healthy
CPU times: user 1.24 s, sys: 63.1 ms, total: 1.3 s
Wall time: 5min 57s ← 6 mins to complete deploy web service
```

Deploying DL on Kubernetes Cluster with GPUs: 04_DeployOnAKS

7

Home > Resource groups > mlaks > mlaksworkspace > mlaksworkspace

mlaksworkspace

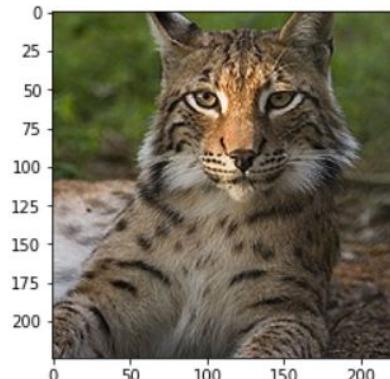
Machine Learning service workspace

Experiments Pipelines Compute Models Images Deployments Activities

ATTRIBUTES

Description	
State	Healthy
Compute Type	AKS
Service ID	my-aks-service-2
Tags	
Creation date	02/09/2019, 11:45:57 AM UTC
Last updated	02/09/2019, 11:45:57 AM UTC
Compute name	my-aks-gpu2
Image ID	mlaksimage1:1
Scoring URI	http://137.116.138.1/api  
Authentication enabled	true
Primary key	<input type="text" value="6EjuEhMixezS0GrDRcRq"/>  
Secondary key	<input type="text" value="ZklREPuSwZ5w3ce8xf5E"/>  
Scoring timeout	
CPU	0.1
Memory	0.5 GB
Autoscale enabled	false

```
Out[42]: <matplotlib.image.AxesImage at 0x7f6c123f4be0>
```



You should be able to make prediction by making a request to the deployed web service

```
In [43]:
```

```
jsonimg = img_url_to_json(IMAGEURL)
jsonimg[:100]
```

```
Out[43]: '{"input": {"image": "\\\\"iVBORw0KGgoAAAANSUhEUgAAAOAAAAADgCAIAAACVT/22AAABJGLDQ1BJQ0MgUHJvZmlsZQAAeJxjY"}}
```

```
In [44]:
```

```
resp = aks_service.run(input_data = jsonimg)
print(resp)
```

```
[{"image": ["n02127052", "lynx", "0.9816483"], ["n02128385", "leopard", "0.0077441484"], ["n02123159", "tiger_cat", "0.0036861342"]}], 'Computed in 5511.99 ms'
```

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter 05_TestWebApp Last Checkpoint: 2 hours ago (autosaved)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Not Trusted, Python 3
- Buttons:** File, New, Open, Save, Run, Stop, Next, Previous, Markdown, Edit Presentation, Show Presentation
- Section Header:**

Test deployed web application
- Description:** This notebook pulls some images and tests them against the deployed web application on AKS.
- In [12]:** Python code to import various libraries and modules from testing_utilities, azureml.core.workspace, azureml.core.image, azureml.core.webservice, and dotenv.
- In [2]:** Python code to find_dotenv with raise_error_if_not_found=True.
- Description:** Get the external url for the web application running on AKS cluster.
- In []:** Python code to print the workspace configuration: ws.name, ws.resource_group, ws.location, ws.subscription_id, separated by newlines.
- Description:** Let's retrieve web service.

Deploying DL on Kubernetes Cluster with GPUs: 05_TestWebApp

2

```
In [14]: results = [requests.post(scoring_url, data=img_url_to_json(img), headers=headers) for img in im]
```

```
In [15]: plot_predictions(images, results)
```



lynx
leopard
tiger_cat

sports_car
convertible
racer

liner
dock
wreck



African_crocodile
American_alligator
Komodo_dragon

indri
Madagascar_cat
koala

brambling
partridge
water_ouzel

Deploying DL on Kubernetes Cluster with GPUs: 06_SpeedTestWebApp

1

The screenshot shows a Jupyter Notebook interface with the title "jupyter 06_SpeedTestWebApp (unsaved changes)". The notebook has a Python logo icon in the top right, and buttons for "Logout" and "Control Panel". The toolbar includes "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". Below the toolbar are buttons for "Run", "Code", "Edit Presentation", and "Show Presentation". The main content area displays the following code:

```
In [1]: import asyncio
import json
import urllib.request
from timeit import default_timer

import aiohttp
import matplotlib.pyplot as plt
from testing_utilities import to_img, gen_variations_of_one_image
from tqdm import tqdm
import requests
from dotenv import set_key, get_key, find_dotenv
from azureml.core.workspace import Workspace
from azureml.core.webservice import AksWebservice

%matplotlib inline
```

```
In [2]: print(aiohttp.__version__)
3.3.2
```

```
In [3]: env_path = find_dotenv(raise_error_if_not_found=True)
```

```
In [ ]: ws = Workspace.from_config()
print(ws.name, ws.resource_group, ws.location, ws.subscription_id, sep="\n")
```

```
In [18]: loop = asyncio.get_event_loop()
start_time = default_timer()
complete_responses = loop.run_until_complete(asyncio.ensure_future(run(url_list, num_concurrent)))
# elapsed = default_timer() - start_time
# print('Total Elapsed {}'.format(elapsed))
# print('Avg time taken {0:4.2f} ms'.format(1000*elapsed/len(url_list)))
```

\$ pip install tornado==4.5.3

```
-----  
RuntimeError                                     Traceback (most recent call last)  
<ipython-input-18-ee69af8fc2f0> in <module>  
      1 loop = asyncio.get_event_loop()  
      2 start_time = default_timer()  
----> 3 complete_responses = loop.run_until_complete(asyncio.ensure_future(run(url_list, num  
concurrent=CONCURRENT_REQUESTS)))  
      4 # elapsed = default_timer() - start_time  
      5 # print('Total Elapsed {}'.format(elapsed))
```

```
/anaconda/envs/tutorial_env/lib/python3.6/asyncio/base_events.py in run_until_complete(self,  
future)
```

```
    469         future.add_done_callback(_run_until_complete_cb)
    470     try:
--> 471         self.run_forever()
    472     except:
    473         if new_task and future.done() and not future.cancelled():
```

```
/anaconda/envs/tutorial_env/lib/python3.6/asyncio/base_events.py in run_forever(self)
```

```
    423         self._check_closed()
    424         if self.is_running():
--> 425             raise RuntimeError('This event loop is already running')
    426         if events._get_running_loop() is not None:
    427             raise RuntimeError(
```

```
RuntimeError: This event loop is already running
```

Then close and reopen this notebook

```
In [ ]: ! pip install tornado==4.5.3
```

```
In [15]: loop = asyncio.get_event_loop()
start_time = default_timer()
complete_responses = loop.run_until_complete(asyncio.ensure_future(run(url_list, num_concurrent))
elapsed = default_timer() - start_time
print('Total Elapsed {}'.format(elapsed))
print('Avg time taken {:.4f} ms'.format(1000*elapsed/len(url_list)))
```

```
100%|██████████| 100/100 [00:05<00:00, 17.19it/s]
```

```
Total Elapsed 5.82007851400067
```

```
Avg time taken 58.20 ms
```



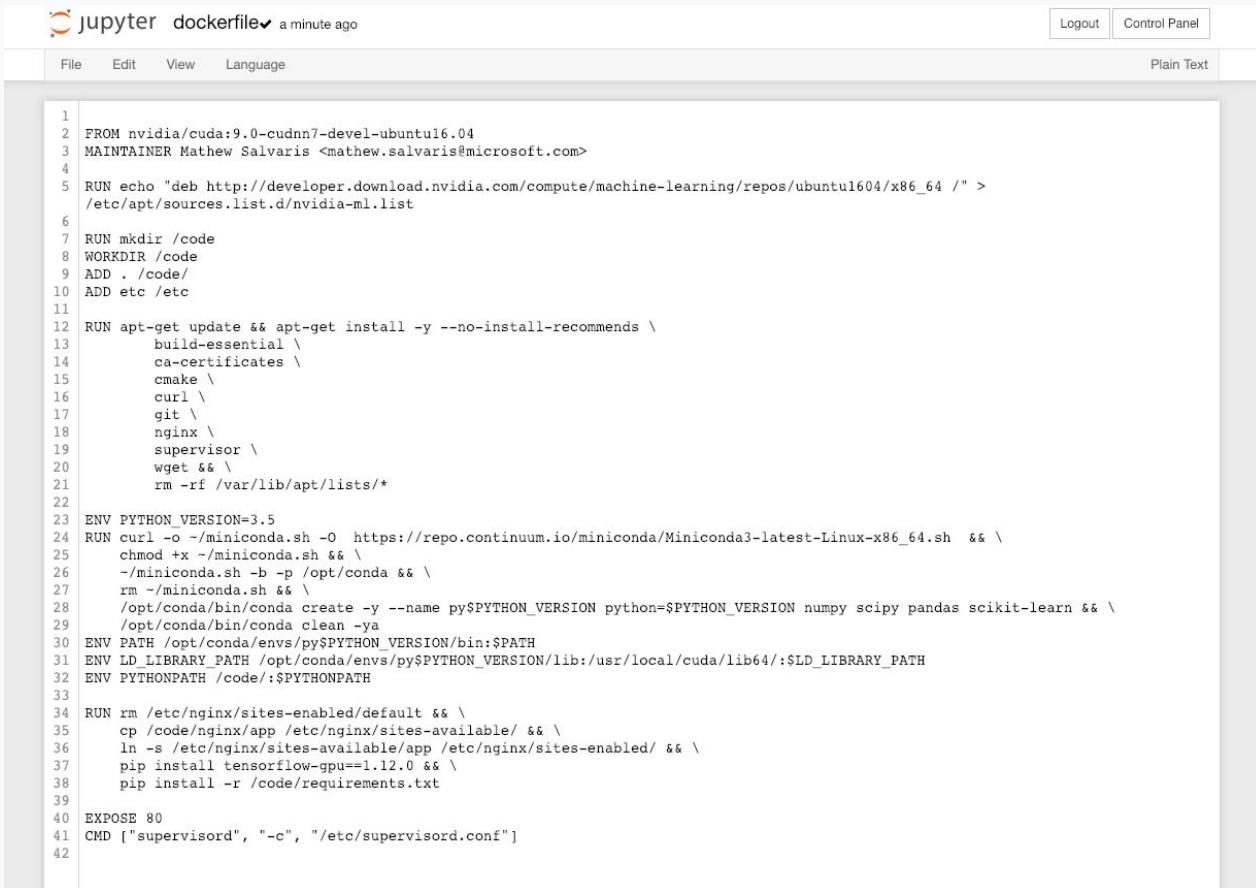
Now should be working

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

jupyter requirements.txt ✓ a minute ago

```
1 pillow
2 click==6.7
3 configparser==3.5.0
4 Flask==0.11.1
5 gunicorn==19.6.0
6 json-logging-py==0.2
7 MarkupSafe==1.0
8 olefile==0.44
9 requests==2.12.3
10 tensorflow==1.12.0
11 keras=2.2.0
12
```

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

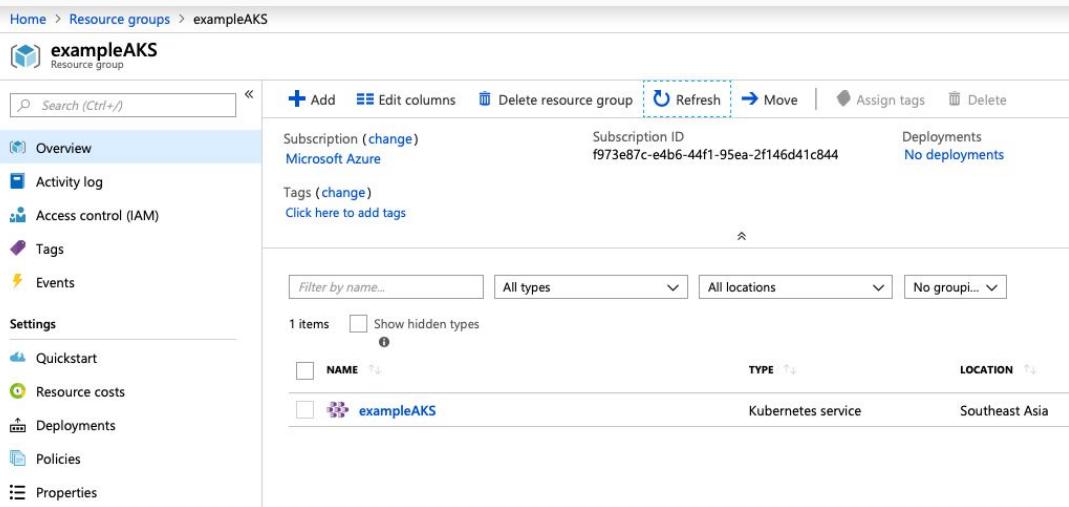


The screenshot shows a Jupyter notebook interface with a Dockerfile titled "dockerfile". The notebook has a header with "Logout" and "Control Panel" buttons, and a toolbar with "File", "Edit", "View", "Language", and "Plain Text" tabs.

```
1 FROM nvidia/cuda:9.0-cudnn7-devel-ubuntu16.04
2 MAINTAINER Mathew Salvaris <mathew.salvaris@microsoft.com>
3
4 RUN echo "deb http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1604/x86_64 /" >
5     /etc/apt/sources.list.d/nvidia-ml.list
6
7 RUN mkdir /code
8 WORKDIR /code
9 ADD . /code/
10 ADD etc /etc
11
12 RUN apt-get update && apt-get install -y --no-install-recommends \
13     build-essential \
14     ca-certificates \
15     cmake \
16     curl \
17     git \
18     nginx \
19     supervisor \
20     wget && \
21     rm -rf /var/lib/apt/lists/*
22
23 ENV PYTHON_VERSION=3.5
24 RUN curl -o ~/miniconda.sh -O https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh && \
25     chmod +x ~/miniconda.sh && \
26     ~/miniconda.sh -b -p /opt/conda && \
27     rm ~/miniconda.sh && \
28     /opt/conda/bin/conda create -y --name py$PYTHON_VERSION python=$PYTHON_VERSION numpy scipy pandas scikit-learn && \
29     /opt/conda/bin/conda clean -ya
30 ENV PATH /opt/conda/envs/py$PYTHON_VERSION/bin:$PATH
31 ENV LD_LIBRARY_PATH /opt/conda/envs/py$PYTHON_VERSION/lib:/usr/local/cuda/lib64/:$LD_LIBRARY_PATH
32 ENV PYTHONPATH /code/:$PYTHONPATH
33
34 RUN rm /etc/nginx/sites-enabled/default && \
35     cp /code/nginx/app /etc/nginx/sites-available/ && \
36     ln -s /etc/nginx/sites-available/app /etc/nginx/sites-enabled/ && \
37     pip install tensorflow-gpu==1.12.0 && \
38     pip install -r /code/requirements.txt
39
40 EXPOSE 80
41 CMD ["supervisord", "-c", "/etc/supervisord.conf"]
42
```

Deploying Python Models on a Kubernetes Clusters for Real-Time Scoring

Deployment on AKS



The screenshot shows the Azure portal interface for the 'exampleAKS' resource group. The left sidebar includes links for Home, Resource groups, exampleAKS, Overview, Activity log, Access control (IAM), Tags, Events, Quickstart, Resource costs, Deployments, Policies, and Properties. The main content area displays the following information:

- Subscription (change): Microsoft Azure** (Subscription ID: f973e87c-e4b6-44f1-95ea-2f146d41c844)
- Tags (change): Click here to add tags**
- Deployments: No deployments**
- Resource List:** A table showing one item: 'exampleAKS' (Kubernetes service, Southeast Asia).

Below, we create the AKS cluster with 5 nodes in the resource group we created earlier. This step can take ten or more minutes.

```
In [*]: %%time
!sudo az aks create --resource-group $resource_group --name $aks_name --node-count 5 --genera
```

- Running ...principal creation[#####] 100.0000%

Deploying Python Models on a Kubernetes Clusters for Real-Time Scoring

Deployment on AKS

Home > Resource groups > exampleAKS > exampleAKS

exampleAKS

Kubernetes service

Search (Ctrl+/
)

Move Delete Refresh

Creating

Resource group (change)
[exampleAKS](#)

Status
Creating

Location
Southeast Asia

Kubernetes version
1.9.11

Subscription (change)
[Microsoft Azure](#)

Subscription ID
f973e87c-e4b6-44f1-95ea-2f146d41c844

API server address
[exampleaks-exampleaks-f973e8-2e96591d.hcp.southeastasia.azmk8s.io](#)

Total cores
40

Total memory
140

HTTP application routing domain
N/A

Tags (change)
Click here to add tags

Monitor containers
Get health and performance insights
[Go to Azure Monitor insights](#)

View logs
Search and analyze logs using ad-hoc queries
[Go to Azure Monitor logs](#)

View Kubernetes dashboard
Learn how to connect to the Kubernetes dashboard
[View connection steps](#)

Overview

Activity log

Access control (IAM)

Tags

Settings

Upgrade

Scale

Properties

Locks

Automation script

Monitoring

Insights (preview)

Metrics (preview)

Logs

Support + troubleshooting

New support request

Deploying Python Models on a Kubernetes Clusters for Real-Time Scoring

Deployment on AKS

Kubernetes dashboard

To open your Kubernetes dashboard, complete the following steps:

- 1 Open Azure CLI version 2.0.27 or later. This will not work in cloud shell and must be running on your local machine. [How to install the Azure CLI](#)
- 2 If you do not already have kubectl installed in your CLI, run the following command:
`az aks install-cli`
- 3 Get the credentials for your cluster by running the following command:
`az aks get-credentials --resource-group exampleAKS --name exampleAKS`
- 4 Open the Kubernetes dashboard by running the following command:
`az aks browse --resource-group exampleAKS --name exampleAKS`

Useful links

[What is Kubernetes dashboard?](#)

[Full instructions for opening the Kubernetes dashboard](#)

Let's verify connection by listing the nodes.

```
In [10]: !sudo kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
aks-nodepool1-36721700-0	Ready	agent	6m45s	v1.9.11
aks-nodepool1-36721700-1	Ready	agent	7m1s	v1.9.11
aks-nodepool1-36721700-2	Ready	agent	6m45s	v1.9.11
aks-nodepool1-36721700-3	Ready	agent	6m43s	v1.9.11
aks-nodepool1-36721700-4	Ready	agent	6m31s	v1.9.11

Let's check the pods on our cluster.

```
In [12]: !sudo kubectl get pods --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	heapster-5884fdbc48-fc9d8	2/2	Running	0	20m
kube-system	kube-dns-v20-b8ff799f7-5b6xq	3/3	Running	0	20m
kube-system	kube-dns-v20-b8ff799f7-8hh8m	3/3	Running	0	20m
kube-system	kube-proxy-9gbzx	1/1	Running	0	12m
kube-system	kube-proxy-cgjdh	1/1	Running	0	12m
kube-system	kube-proxy-dhpfb	1/1	Running	0	12m
kube-system	kube-proxy-ghqgs	1/1	Running	0	12m
kube-system	kube-proxy-md6gg	1/1	Running	0	12m
kube-system	kube-svc-redirect-5xn9g	2/2	Running	0	12m
kube-system	kube-svc-redirect-jdp49	2/2	Running	0	12m
kube-system	kube-svc-redirect-jf5sb	2/2	Running	0	12m
kube-system	kube-svc-redirect-nr5wd	2/2	Running	0	12m
kube-system	kube-svc-redirect-smdd2	2/2	Running	0	12m
kube-system	kubernetes-dashboard-cb6558ddb-qpff6	1/1	Running	3	20m
kube-system	tunnelfront-fb9d76575-gt7d2	1/1	Running	0	20m

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

jupyter 04_DeployOnAKS (unsaved changes) Logout Control Panel

File Edit View Insert Cell Kernel Help Not Trusted Python [conda env:AKSDeployment] O

File Edit View Insert Cell Kernel Help Not Trusted Python [conda env:AKSDeployment] O

Let's check if the pod is deployed.

In [23]: `!kubectl get pods --all-namespaces`

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
default	azuredl-556584cdf7-hwkkk	0/1	ContainerCreating	0	1s
kube-system	heapster-779db6bd48-mmwsj	2/2	Running	0	10m
kube-system	kube-dns-v20-6df5d5c657-5tt9k	4/4	Running	0	10m
kube-system	kube-dns-v20-6df5d5c657-zmg94	4/4	Running	0	10m
kube-system	kube-proxy-vf152	1/1	Running	0	6m38s
kube-system	kube-svc-redirect-v2fgl	2/2	Running	0	6m38s
kube-system	kubernetes-dashboard-7fbf669f58-bgcpf	1/1	Running	1	10m
kube-system	tunnelfront-6dd66887b45-dlpc	1/1	Running	0	10m

If anything goes wrong you can use the commands below to observe the events on the node as well as review the logs.

In [24]: `!kubectl get events`

LAST SEEN	TYPE	REASON	KIND	MESSAGE
6m42s	Normal	Starting	Node	Starting kubelet.
6m42s	Normal	NodeHasSufficientDisk	Node	Node aks-nodepool1-12279994-0 status is now: NodeHasSuffi
6m42s	Normal	NodeHasSufficientMemory	Node	Node aks-nodepool1-12279994-0 status is now: NodeHasSuffi
6m42s	Normal	NodeHasNoDiskPressure	Node	Node aks-nodepool1-12279994-0 status is now: NodeHasNoDis
6m42s	Normal	NodeAllocatableEnforced	Node	Updated Node Allocatable limit across pods
6m38s	Normal	RegisteredNode	Node	Node aks-nodepool1-12279994-0 event: Registered Node aks-
nodepool1-12279994-0		in Controller		
6m22s	Normal	NodeReady	Node	Node aks-nodepool1-12279994-0 status is now: NodeReady
4m59s	Normal	Starting	Node	Starting kube-proxy.
5s	Normal	Scheduled	Pod	Successfully assigned azuredl-556584cdf7-hwkkk to aks-nod
epool1-12279994-0				
5s	Normal	SuccessfulMountVolume	Pod	MountVolume.SetUp succeeded for volume "nvidia"
5s	Normal	SuccessfulMountVolume	Pod	MountVolume.SetUp succeeded for volume "default-token-zdg
mk"				
4s	Normal	Pulling	Pod	pulling image "kwarodom/tfresnet-gpu"
5s	Normal	SuccessfulCreate	ReplicaSet	Created pod: azuredl-556584cdf7-hwkkk
5s	Normal	ScalingReplicaSet	Deployment	Scaled up replica set azuredl-556584cdf7 to 1
5s	Normal	EnsuringLoadBalancer	Service	Ensuring load balancer

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

MC_bkkaidevops_devopsaks_southeastasia

Add Edit columns Delete resource group Refresh Move Assign tags Delete

Subscription (change) Microsoft Azure Subscription ID f973e87c-e4b6-44f1-95ea-2f146d41c844 Deployments 1 Succeeded

Tags (change) Click here to add tags

Filter by name... All types All locations No grouping

9 items Show hidden types

NAME	TYPE	LOCATION	...
aks-agentpool-12279994-nsg	Network security group	Southeast Asia	...
aks-agentpool-12279994-routetable	Route table	Southeast Asia	...
aks-nodepool1-12279994-0	Virtual machine	Southeast Asia	...
aks-nodepool1-12279994-0_OsDisk_1_129db5d6c2d14cd1ae21fb144e20...	Disk	Southeast Asia	...
aks-nodepool1-12279994-nic-0	Network interface	Southeast Asia	...
aks-vnet-12279994	Virtual network	Southeast Asia	...
kubernetes	Load balancer	Southeast Asia	...
kubernetes-a50b2d62a066611e9afa3a652af07bc3	Public IP address	Southeast Asia	...
nodepool1-availabilitySet-12279994	Availability set	Southeast Asia	...

Check if the container is created!!

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Below we run the 100 requests against our deployed service

```
In [15]: loop = asyncio.get_event_loop()
start_time = default_timer()
complete_responses = loop.run_until_complete(asyncio.ensure_future(run(url_
elapsed = default_timer() - start_time
print('Total Elapsed {}'.format(elapsed))
print('Avg time taken {:.2f} ms'.format(1000*elapsed/len(url_list)))

100% ██████████ | 100/100 [00:19<00:00,  5.08it/s]

Total Elapsed 19.709361603000616
Avg time taken 197.09 ms
```

Below we can see the output of some of our calls

**Check tornado version
to be 4.5.3**

Azure / MLAKSDeployment

Code Issues 4 Pull requests 0 Projects 0 Wiki Insights

Branch: master ▾ **MLAKSDeployment** / environment.yml

Mario Bourgoin Workaround azure-cli-interactive bug.

1 contributor

20 lines (19 sloc) | 401 Bytes

```
1 name: MLAKSDeployment
2 channels:
3   - conda-forge
4 dependencies:
5   - python=3.5
6   - nb_conda==2.2.0
7   - ipywidgets==7.3.0
8   - tornado==4.5.3
9   - pandas==0.23.3
10  - scikit-learn==0.19.1
11  - pip:
12    - aiohttp==3.3.2
13    - toolz==0.9.0
14    - tqdm==4.23.4
15    - azure-cli==2.0.41
16    - prompt_toolkit==2.0.6
17    - lightgbm==2.1.2
18    - papermill==0.14.1
19    - git+https://github.com/theskumar/python-dotenv
```

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

Below we run the 100 requests against our deployed service

```
In [15]: loop = asyncio.get_event_loop()
start_time = default_timer()
complete_responses = loop.run_until_complete(asyncio.ensure_future(run(url_list, num_concurrent=CONCURRENT_REQUESTS)))
elapsed = default_timer() - start_time
print('Total Elapsed {}'.format(elapsed))
print('Avg time taken {:.2f} ms'.format(1000*elapsed/len(url_list)))

100%|██████████| 100/100 [00:19<00:00,  5.08it/s]

Total Elapsed 19.709361603000616
Avg time taken 197.09 ms
```

Below we can see the output of some of our calls

Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

```
In [15]: loop = asyncio.get_event_loop()
start_time = default_timer()
complete_responses = loop.run_until_complete(asyncio.ensure_future(run(url_list, num_concurrent=CONCURRENT_REQUESTS)))
elapsed = default_timer() - start_time
print('Total Elapsed {}'.format(elapsed))
print('Avg time taken {0:4.2f} ms'.format(1000*elapsed/len(url_list)))

100%|██████████| 100/100 [00:19<00:00,  5.08it/s]

Total Elapsed 19.709361603000616
Avg time taken 197.09 ms
```

Below we can see the output of some of our calls

```
In [16]: complete_responses[:3]

Out[16]: [{('result': [{'image': [['n02127052 lynx, catamount', 0.9974349141120911],
['n02128385 leopard, Panthera pardus', 0.0015204271767288446],
['n02128757 snow leopard, ounce, Panthera uncia',
0.0005193596007302403]}],
'Computed in 192.92 ms']},
0.20412451300035173),
{('result': [{'image': [['n02127052 lynx, catamount', 0.9974443912506104],
['n02128385 leopard, Panthera pardus', 0.0015130586689338088],
['n02128757 snow leopard, ounce, Panthera uncia',
0.0005179222207516432]}],
'Computed in 194.12 ms'}},
0.3988716359999671),
{('result': [{'image': [['n02127052 lynx, catamount', 0.9974856376647949],
['n02128385 leopard, Panthera pardus', 0.0014885306591168046],
['n02128757 snow leopard, ounce, Panthera uncia',
0.0005080725532025099]}],
'Computed in 200.07 ms'}},
0.6001250520002941)]
```

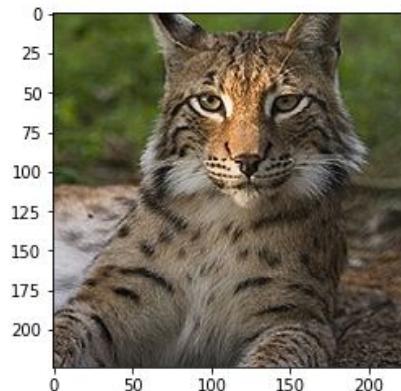
Deploying Deep Learning CNN on Kubernetes Cluster with GPUs

```
In [17]: num_successful=[i[0]['result'][0]['image'][0][0] for i in complete_responses].count('n02127052 lynx, catamount')
print('Successful {} out of {}'.format(num_successful, len(url_list)))
```

Succesful 100 out of 100

```
In [18]: # Example response
plt.imshow(to_img(IMAGEURL))
complete_responses[0]
```

```
Out[18]: {'result': [{'image': [['n02127052 lynx, catamount', 0.9974349141120911],
                                ['n02128385 leopard, Panthera pardus', 0.0015204271767288446],
                                ['n02128757 snow leopard, ounce, Panthera uncia',
                                 0.0005193596007302403]]},
                         'Computed in 192.92 ms'],
            0.20412451300035173}
```



Update the web service

When you create a new image, you must manually update each service that you want to use the new image. To update the web service, use the `update` method. The following code demonstrates how to update the web service to use a new image:

Python

 Copy

```
from azureml.core.webservice import Webservice
from azureml.core.image import Image

service_name = 'aci-mnist-3'
# Retrieve existing service
service = Webservice(name = service_name, workspace = ws)

# point to a different image
new_image = Image(workspace = ws, id="myimage2:1")

# Update the image used by the service
service.update(image = new_image)
print(service.state)
```

For more information, see the reference documentation for the [Webservice](#) class.

Tear it all down

Once you are done with your cluster you can use the following two commands to destroy it all.

```
In [1]: from azureml.core import Workspace  
from azureml.core.compute import AksCompute, ComputeTarget  
from dotenv import set_key, get_key, find_dotenv
```

```
In [ ]: ws = Workspace.from_config()  
print(ws.name, ws.resource_group, ws.location, ws.subscription_id, sep="\n")
```

```
In [2]: env_path = find_dotenv(raise_error_if_not_found=True)  
resource_group = get_key(env_path, 'resource_group')  
aks_name = get_key(env_path, 'aks_name')
```

Once you are done with your cluster you can use the following command to delete the AKS cluster. This step may take a few minutes.

```
In [ ]: aks_target = AksCompute(name=aks_name, workspace=ws)  
aks_aml_name = aks_target.cluster_resource_id.rsplit("/")[-1]
```

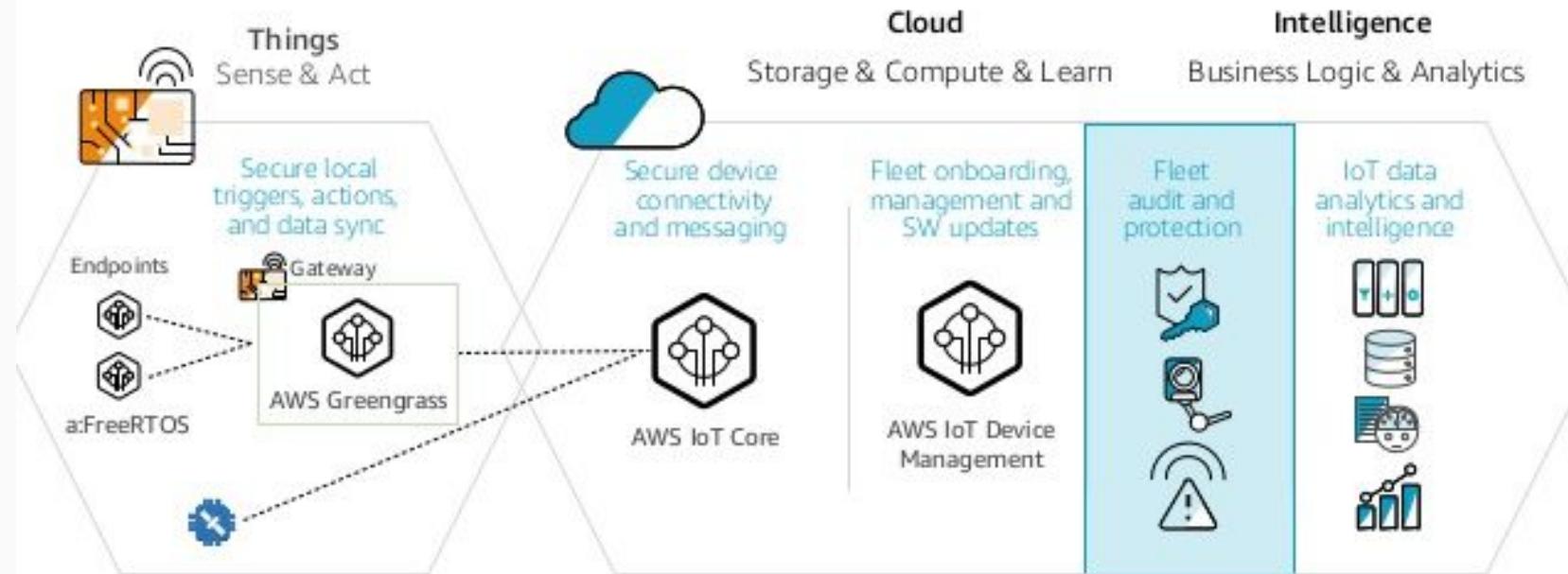
```
In [ ]: !az aks delete -n $aks_aml_name -g $resource_group -y
```

Finally, you should delete the resource group. This also deletes the AKS cluster and can be used instead of the above command if the resource group is only used for this purpose.

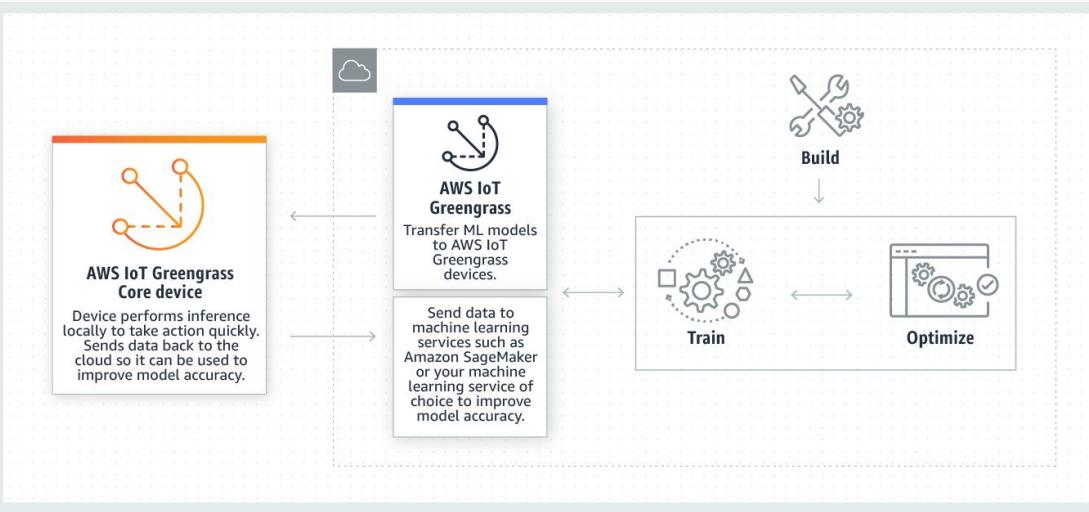
6. Edge

Develop and
deploy model on
edge devices

Solving IoT Business Challenges with the Extended AWS IoT Services Suite

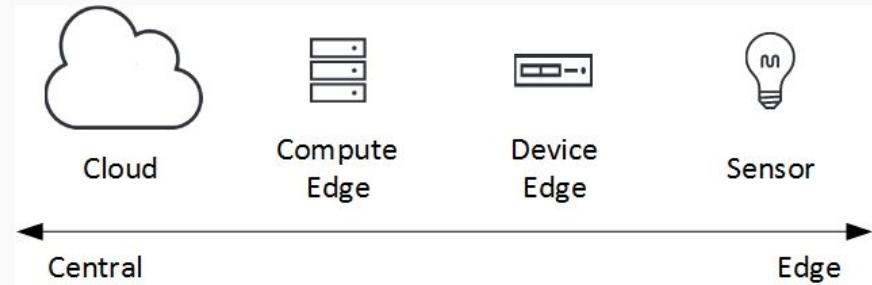


Deploy machine learning on iot devices

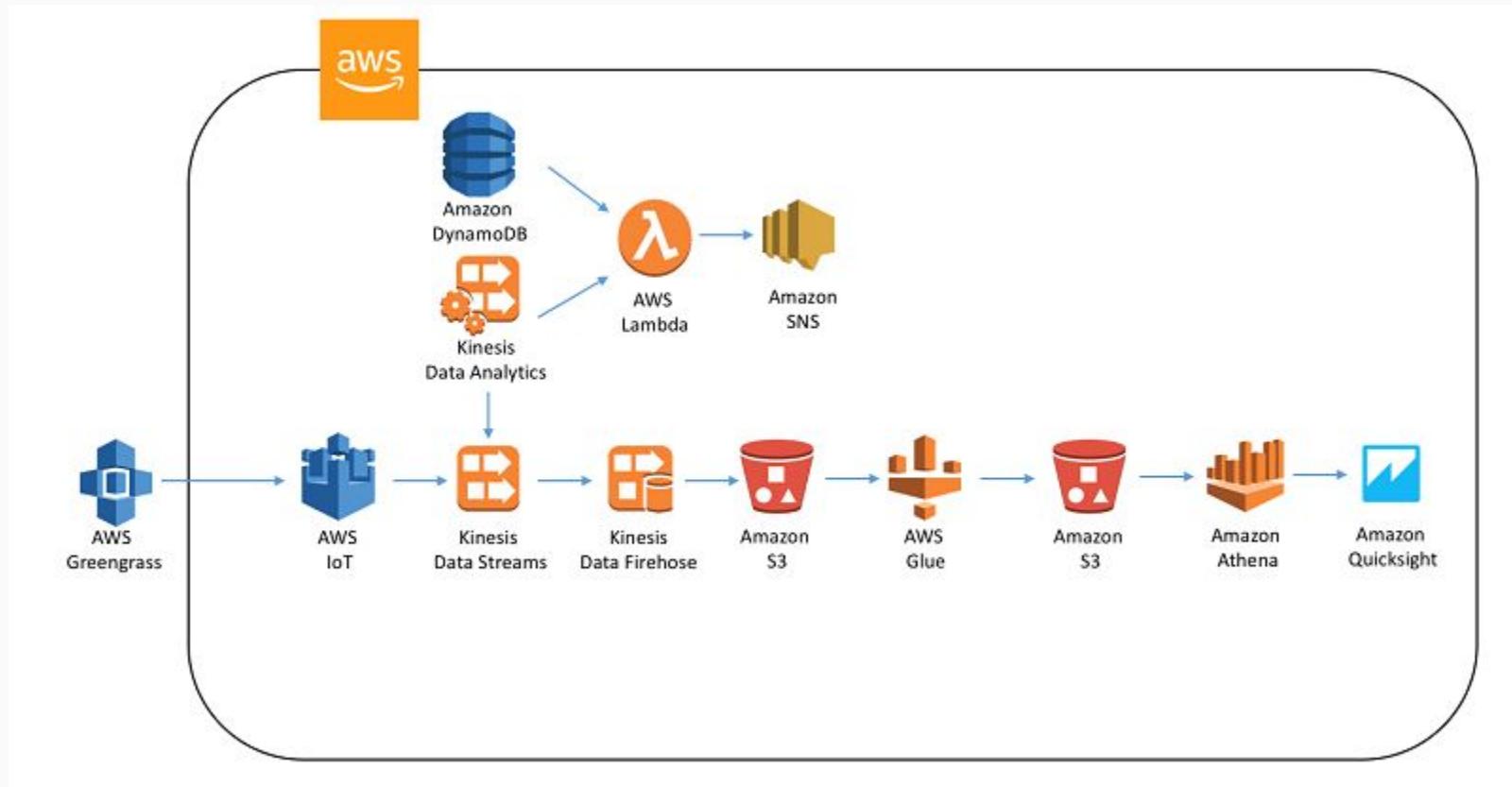


Training model in cloud platforms and then deploy selected model in the edge devices

Edge computing is more scalable than centralized (data have been preprocessed before be uploaded to the cloud) .



Example AWS greengrass usage



Next Steps

<https://onnx.ai>

The screenshot shows the ONNX website's "Next Steps" section. It features two main sections: "Frameworks" and "Converters".

Frameworks:

- Caffe2
- Chainer
- Cognitive Toolkit
- mxnet
- PyTorch
- PaddlePaddle
- MATLAB
- sas Neural Network Libraries

Converters:

- TensorFlow
- Keras
- MLlib
- Scikit-learn
- XGBoost
- LibSVM
- Caffe



ONNX is an open format to represent deep learning models. With ONNX, AI developers can more easily move models between state-of-the-art tools and choose the combination that is best for them. ONNX is developed and supported by a community of partners.

Example: <https://github.com/onnx/tutorials/blob/master/tutorials/TensorflowToOnnx-1.ipynb>

ONNX support many platforms



Facebook
Open Source



HUAWEI



ONNX potentials

Exporting model from PyTorch to ONNX

In this tutorial, we describe how to use ONNX to convert a model defined in PyTorch into the ONNX format.

ONNX exporter is part of the [PyTorch repository](#).

For working with this tutorial, you will need to install [onnx](#). You can get binary builds of onnx with `conda install -c conda-forge onnx`.

NOTE: ONNX is under active development so for the best support consider building PyTorch master branch which can be installed by following [the instructions here](#)

Invoking exporter

Pretty much it's a matter of replacing `my_model(input)` with `torch.onnx.export` in your script.

Convert Tensorflow model to ONNX

Tensorflow and ONNX both define their own graph format to represent a model to ONNX.

We divide the guide into 2 parts: part 1 covers basic conversion and part 2 advanced order:

1. Procedures to convert tensorflow model

- get tensorflow model
- convert to ONNX
- validate

2. Key conceptions

- opset
- data format

Step 1 - Get Tensorflow model

Tensorflow uses several file formats to represent a model, such as checkpoint files, graph with weight (called frozen graph next)

Artifacts which wrote from different framework can be convert to the same type

Serving the Model Archive with MXNet Model Server

Now that we have the **Model Archive**, we can start the server and ask it to setup HTTP endpoints to serve the model, emit real-time operational metrics and more.

We'll also test the server's endpoints.

```
In [ ]: # Spawning a new process to run the server
import subprocess as sp
server = sp.Popen("mxnet-model-server --models squeezenet=squeezenet.model", shell=True)
```

```
In [ ]: # Check out the health endpoint
```

Ref: <https://github.com/onnx/tutorials>

ONNX potentials

Hosting ONNX models with Amazon Elastic Inference

Amazon Elastic Inference (EI) is a resource you can attach to your Amazon EC2 instances to accelerate your deep learning (DL) inference workloads. EI allows you to add inference acceleration to an Amazon SageMaker hosted endpoint or Jupyter notebook for a fraction of the cost of using a full GPU instance. It reduces the cost of running deep learning inference by up to 75%.

Amazon EI provides support for Apache MXNet, TensorFlow and ONNX models. The [Open Neural Network Exchange](#) (ONNX) is an open standard format for deep learning models that enables interoperability between deep learning frameworks such as Apache MXNet, Caffe2, Microsoft Cognitive Toolkit(CNTK), PyTorch and more. This means that we can use any of these frameworks to train the model, export these pretrained models in ONNX format and then import them in MXNet for inference. For more information please visit: <https://docs.aws.amazon.com/sagemaker/latest/dg/ei.html>

In this example, we will use the ResNet-152v1 model from [Deep residual learning for image recognition](#). This model, alongside many others, can be found at the [ONNX Model Zoo](#).

We will use the SageMaker Python SDK to host this ONNX model in SageMaker, and perform inference requests.

Setup

First, we'll get the IAM execution role from our notebook environment, so that SageMaker can access resources in account later in the example.

```
In [ ]: from sagemaker import get_execution_role  
role = get_execution_role()
```

The hosting script

We'll need to provide a hosting script that can run on the SageMaker platform. This script will be invoked by SageMaker to perform inference.

The script we're using here implements two functions:

- `model_fn()` - the SageMaker model server uses this function to load the model
- `transform_fn()` - this function is for using the model to take the input and produce the output

```
In [ ]: !pygmentize resnet152.py
```

ONNX also have feature to serve models to API services

Serving ONNX model with MXNet Model Server

This tutorial demonstrates how to serve an ONNX model with MXNet Model Server. We'll use a pre-trained SqueezeNet model from ONNX model zoo. The same example can be easily applied to other ONNX models.

Frameworks used in this tutorial:

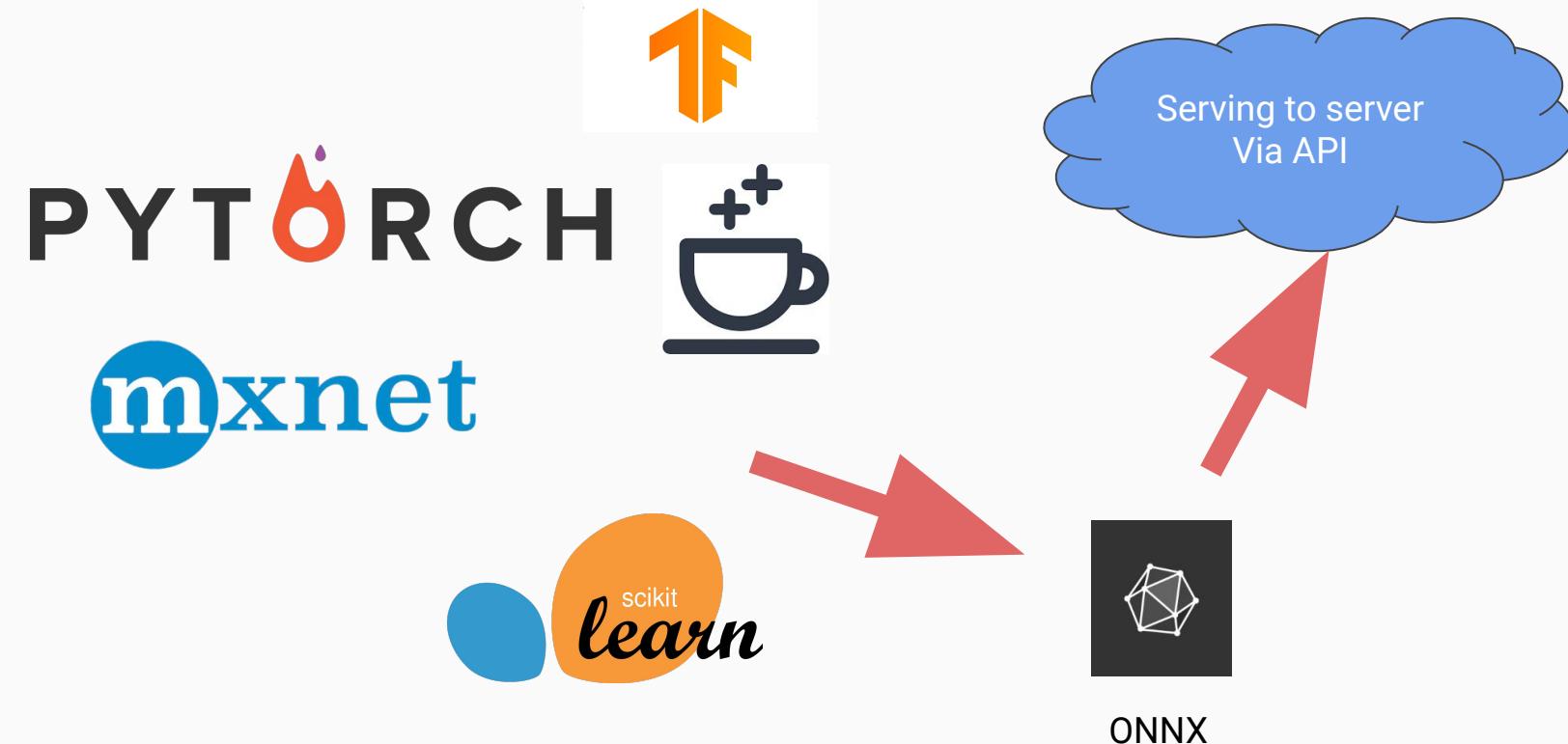
- [MXNet Model Server](#) that uses [MXNet](#)
- [ONNX](#)

Installing pre-requisites:

Next we'll install the pre-requisites:

Ref: <https://github.com/onnx/tutorials>

ONNX pipeline



Thank you