Machine Learning Metamorphosis: Breaking Models Out from your Local Machine and Releasing them into the Cloud

Duke Statistical Science Proseminar
February 1st, 2023

Dr. Zack Abzug
Dan Salo

Speakers

Dan Salo

- Data Science Manager
- Engineering degrees from NC State and Duke
- Professionally: NLP and CV applications in cloud and social media
- Free time: Basketball, Piano, Cooking

Zack Abzug

- Data Science Manager
- BME @ Duke (BS/MS/PhD)
- Previously: ML for phish/malware detection + malware forensic analysis
- Currently: AI-based network detection and response
- Free time: Soccer, Cooking

Outline

- Job Landscape
- Cloud Motivation
- Core Cloud Concepts
- Demo

Job Landscape

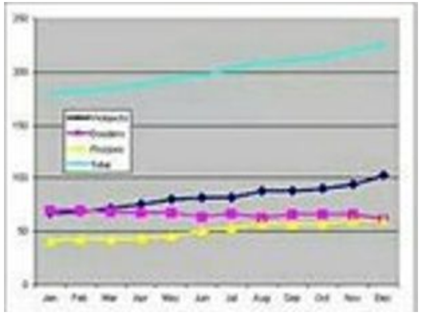
- Roles and Responsibilities
- Skills Overlap
- Future Trends

Data Analyst / Business Analyst / Data Scientist

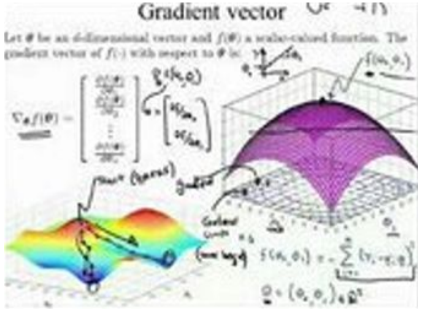
What my friends think I do

What my mom thinks I do

What society thinks I do



What my boss thinks I do



What I think I do

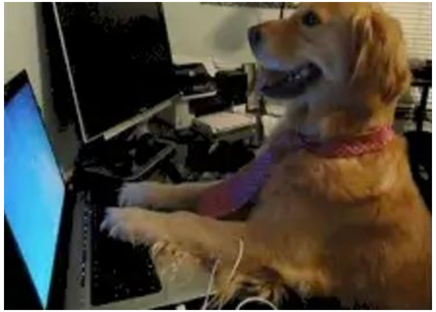
What I actually do

Machine Learning Engineer / Data Scientist

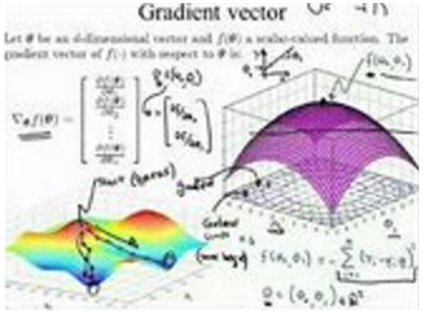
What society thinks I do

What my mom thinks I do

What other engineers think I do



What research scientists think I do



What I think I do

```
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("distilgpt2")

model = AutoModelForCausalLM.from_pretrained("distilgpt2")
```

What I actually do

Roles and Responsibilities

Data Analyst / Business Analyst / Data Scientist

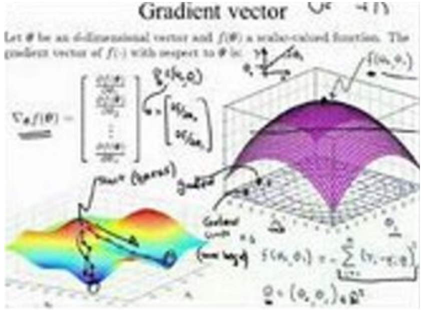
- **Responsibility**: Produce actionable insights from data. Insights are unknown beforehand.
- **Tools**: BI dashboards, databases, statistics, modeling, R/Python, SQL
- **Tasks**: data exploration, data querying, feature engineering, model selection, presenting insights, creating dashboard and database views, interacting with data pipelines

Machine Learning Engineer / Data Scientist

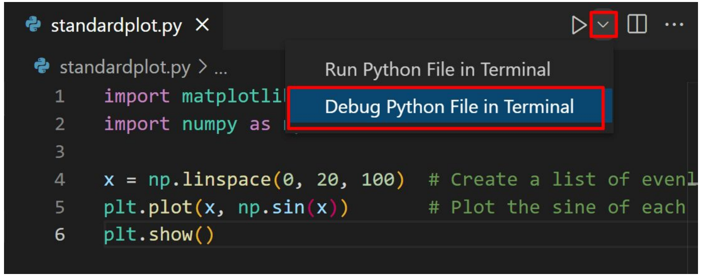
- **Responsibility**: Produce data-driven software features. Desired feature is known beforehand.
- **Tools**: cloud services, databases, machine learning, modeling, Python, SQL
- **Tasks**: write production code, write a training script, feature engineering, model selection, run ML experiments, participate in code reviews, interacting with data pipelines

Skills Overlap

Databases and Pipelines
(Data)



Math and Modeling
(Model)

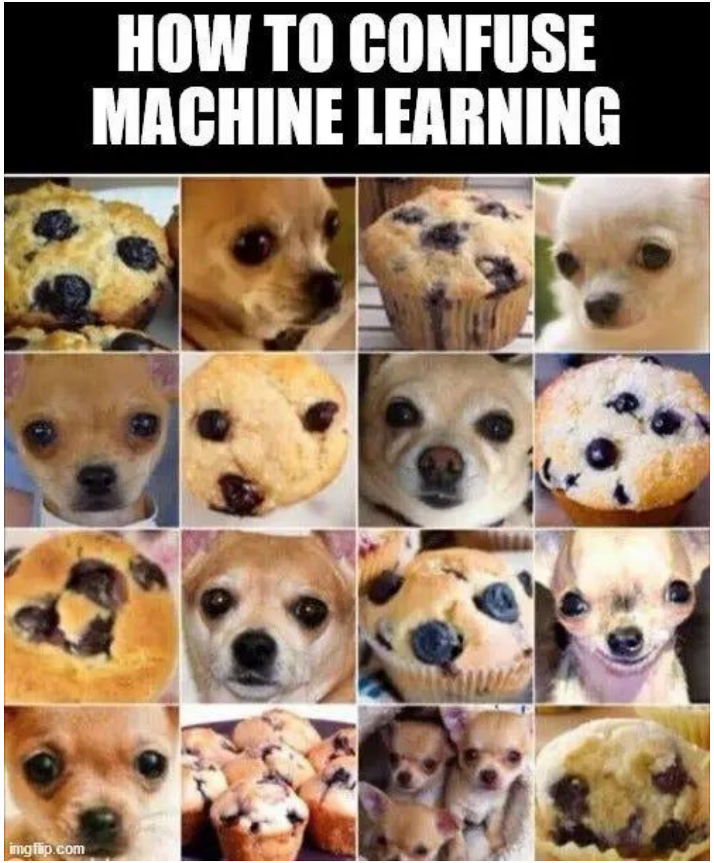


Software Skills
(Code)

Future Trends

- ⬆️ Commoditized frameworks and models
- ⬆️ Debugging and tuning for edge cases
- ⬆️ Other professionals learning data skills
- ⬆️ Business need for dataset curation, model selection, general problem-solving

Data science professionals with advanced degrees will debug and tune standard frameworks to solve problems.



Cloud Motivation

- Compliance & Auditability
- Data Security & Privacy
- Compute & Data Infrastructure

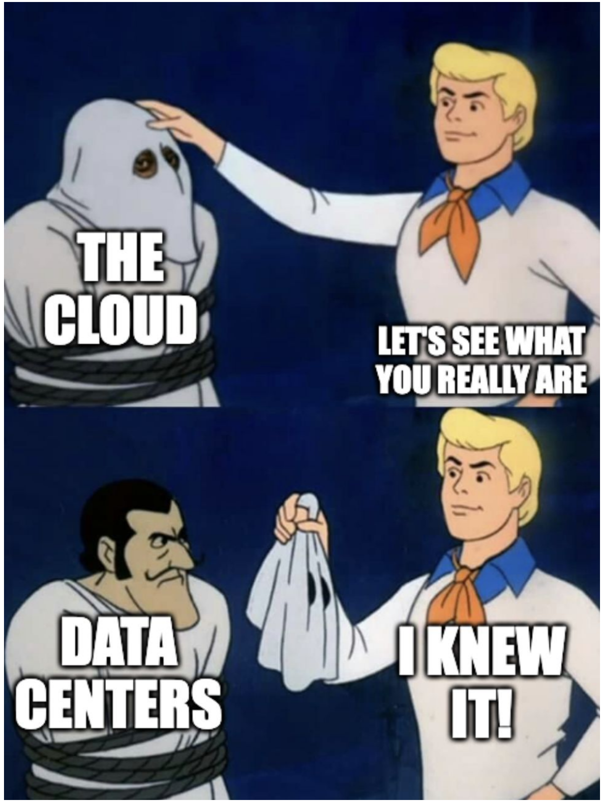
But first, a point of clarification…

But first, a point of clarification…

- Much of what we’re talking about today isn’t strictly “cloud-specific”
- We’re mostly talking about “remote development” more generally
  - Cloud provider, data center, shared cluster, etc.

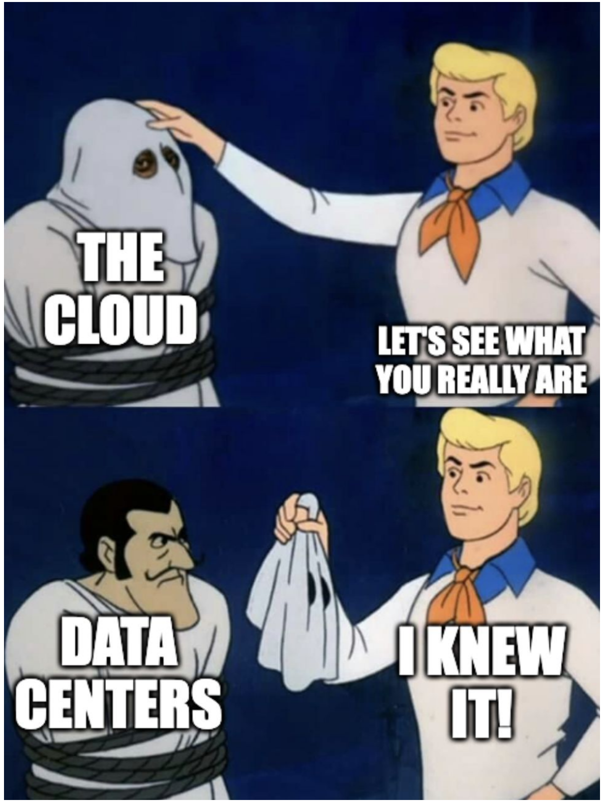
But first, a point of clarification…

- Much of what we're talking about today isn't strictly "cloud-specific"
- We're mostly talking about "remote development" more generally
  - Cloud provider, data center, shared cluster, etc.
  - **The "cloud" is really just an abstraction layer over a collection physical data centers!**



But first, a point of clarification…

- Much of what we're talking about today isn't strictly "cloud-specific"
- We're mostly talking about "remote development" more generally
  - Cloud provider, data center, shared cluster, etc.
  - **The "cloud" is really just an abstraction layer over a collection physical data centers!**
- We'll point out what pieces are truly cloud-specific!



Compliance & Auditability

- There are *many* reasons why it’s important to have an automated audit log of how a model was generated and got pushed to production

Compliance & Auditability

- There are *many* reasons why it’s important to have an automated audit log of how a model was generated and got pushed to production
- In practice, this is much easier to do when there is a centralized platform for model training and deployment!

Compliance & Auditability

- There are *many* reasons why it's important to have an automated audit log of how a model was generated and got pushed to production
- In practice, this is much easier to do when there is a centralized platform for model training and deployment!
- Cloud providers have "access control" services that…control access
  - E.g., ensure that models aren't manually tampered with

Data Security & Privacy

- Respecting sensitive data should be sacred
  - Legal reasons
  - Customer agreement reasons
  - Ethical reasons

Data Security & Privacy

- Respecting sensitive data should be sacred
  - Legal reasons
  - Customer agreement reasons
  - Ethical reasons
- Minimize risk → don’t have sensitive data on your laptop!

Data Security & Privacy

- Respecting sensitive data should be sacred
  - Legal reasons
  - Customer agreement reasons
  - Ethical reasons
- Minimize risk → don’t have sensitive data on your laptop!
- Also ties into auditability: it’s much easier to implement access controls in the cloud

Compute & Data Infrastructure

- Freedom to choose the best tools for the task!

Compute & Data Infrastructure

- Freedom to choose the best tools for the task!
- Compute questions:
    - How many CPUs?
    - How much memory?
    - GPU? TPU?
    - Spark? Hadoop? MapReduce?

Compute & Data Infrastructure

- Freedom to choose the best tools for the task!
- Compute questions:
  - How many CPUs?
  - How much memory?
  - GPU? TPU?
  - Spark? Hadoop? MapReduce?
- Data questions:
  - What is your data volume?
  - SQL? NoSQL?
  - Flat storage?
  - ElasticSearch?

Core Cloud Concepts

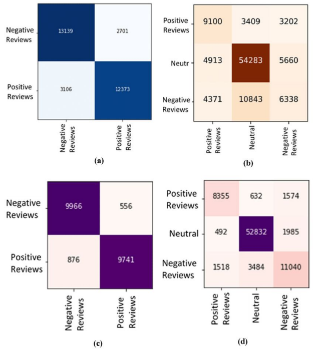
- Production ML Overview
- Training & Versioning
- Containerization
- Deployment & Monitoring

Production ML Overview

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa

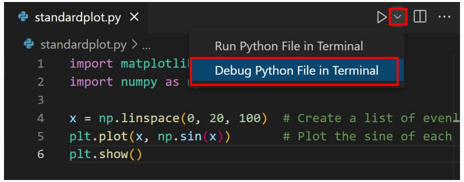
Iris Dataset

Trained Model



Model Evaluation

Local Deploy



Training Code

Question: How is production ML in the cloud different than this workflow? How is it the same?

https://ibm.github.io/data-science-best-practices/versioning.html

Production ML Overview — The Cloud

Automate and Scale Your Local Workflow

- Feature Engineering
- Training Experiments on batch data
- Hyperparameter Tuning
- Model Evaluation
- Model Selection

Address New Challenges

- Real-time training
- Changing data schema and datasets
- Tracking multiple re-trainings
- Monitoring models for re-training
- Sampling from production data for labeling
- Tracking labels from multiple labelers

Training & Versioning

3 basic ingredients for ML training:

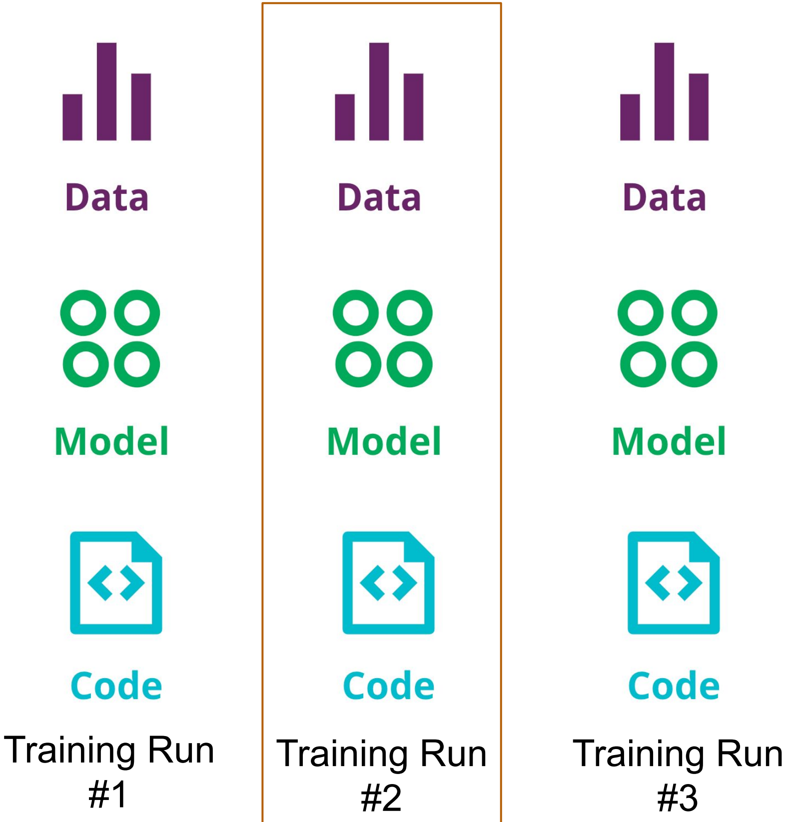
Training Dataset
Validation Dataset
Test Dataset
Out-of-time Test Dataset

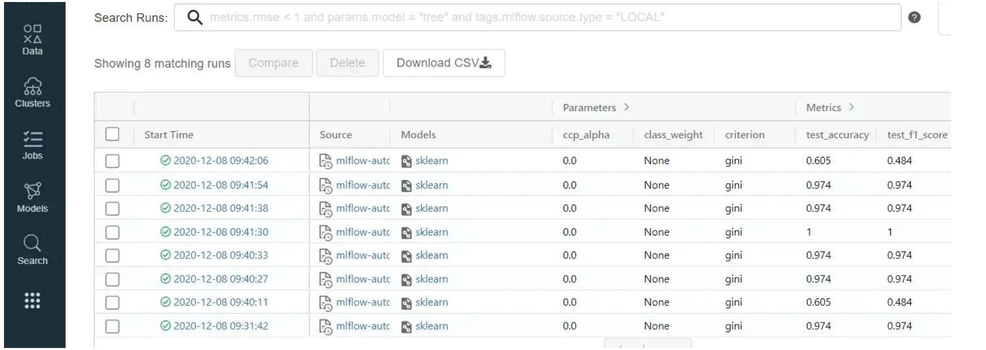
Model Architecture
Pre-trained weights

Training Script
Evaluation Script

https://ibm.github.io/data-science-best-practices/versioning.html

Training & Versioning

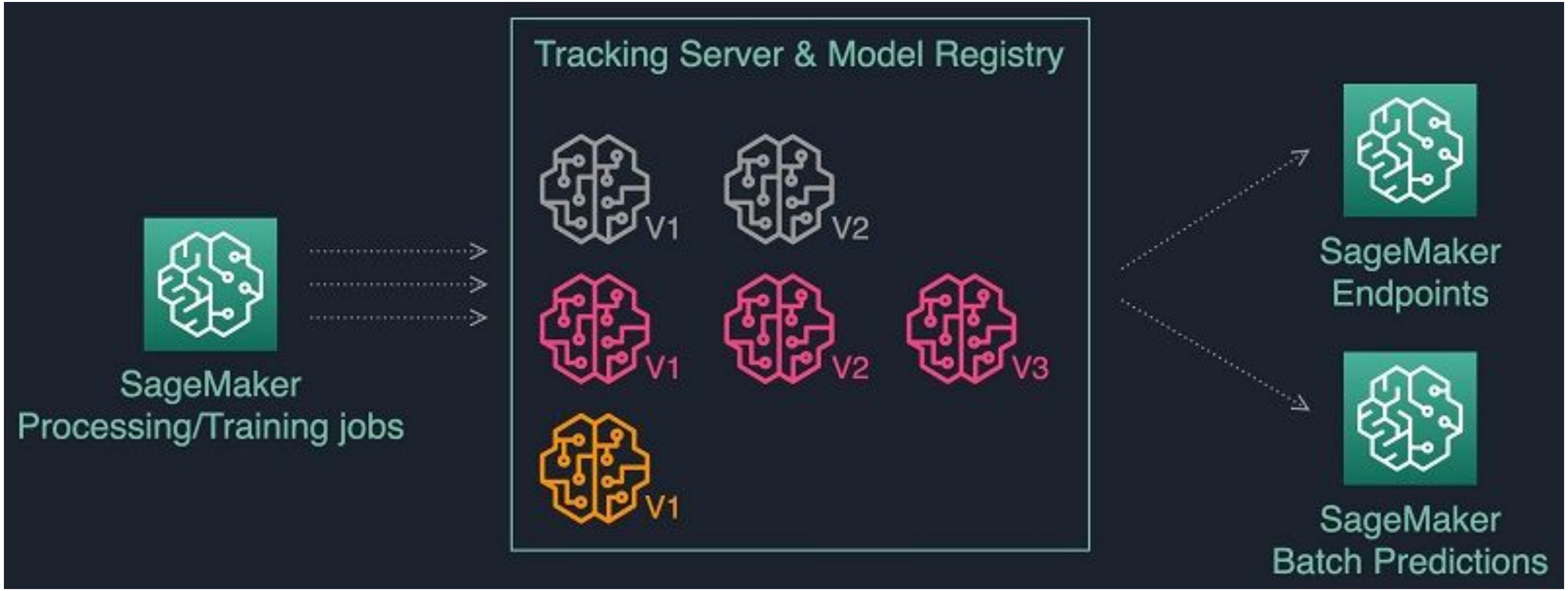




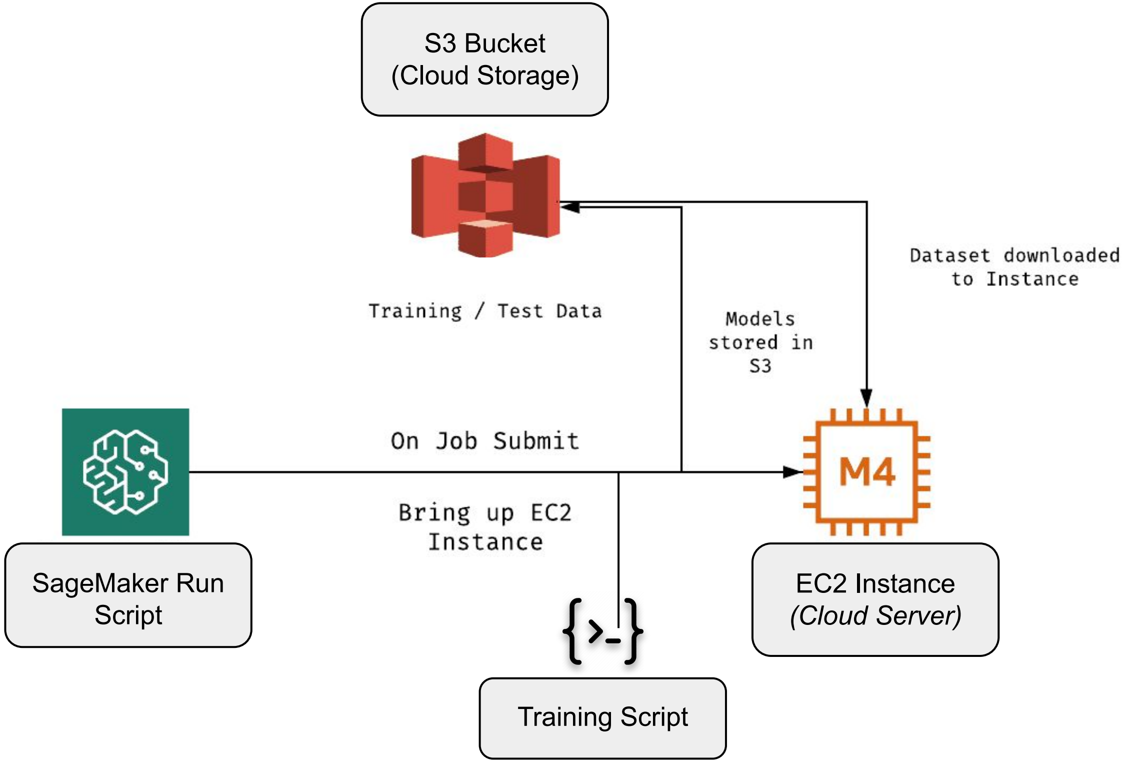
			Parameters			Metrics	
Start Time	Source	Models	ccp_alpha	class_weight	criterion	test_accuracy	test_f1_score
2020-12-08 09:42:06	mlflow-autc	sklearn	0.0	None	gini	0.605	0.484
2020-12-08 09:41:54	mlflow-autc	sklearn	0.0	None	gini	0.974	0.974
2020-12-08 09:41:38	mlflow-autc	sklearn	0.0	None	gini	0.974	0.974
2020-12-08 09:41:30	mlflow-autc	sklearn	0.0	None	gini	1	1
2020-12-08 09:40:33	mlflow-autc	sklearn	0.0	None	gini	0.974	0.974
2020-12-08 09:40:27	mlflow-autc	sklearn	0.0	None	gini	0.974	0.974
2020-12-08 09:40:11	mlflow-autc	sklearn	0.0	None	gini	0.605	0.484
2020-12-08 09:31:42	mlflow-autc	sklearn	0.0	None	gini	0.974	0.974

- Data-Model-Code snapshots enable auditable training through reproducibility
- MLFlow (or others) can track experiments and metrics
- Versioning allows incoming predictions to be tagged and tied to a model

Training & Versioning



Training & Versioning



After executing the SageMaker Run Script:

1. EC2 Instance starts up
2. Training script loaded onto instance
3. Datasets downloaded to instance from s3
4. Training script runs
5. Model saved to instance
6. Model uploaded

https://awsmachinelearning.dev/machine-learning-workflow-aws-sagemaker/

Training & Versioning

Example SageMaker Run Script, pointing to *your_training_script.py*:

```
smd_mp_estimator = Estimator(
    entry_point="your_training_script.py",
    role=sagemaker.get_execution_role(),
    instance_type='ml.p3.16xlarge',
    sagemaker_session=sagemaker_session,
    image_uri='your_aws_account_id.dkr.ecr.region.amazonaws.com/name:tag'
    instance_count=1,
    distribution={
        "smdistributed": smp_options,
        "mpi": mpi_options
    },
    base_job_name="SMD-MP-demo",
)

smd_mp_estimator.fit('s3://my_bucket/my_training_data/')
```

Containerization

- Define/control the environment code executes in independent of the underlying hardware

Containerization

- Define/control the environment code executes in independent of the underlying hardware
- Containerization != Docker (at least in theory)

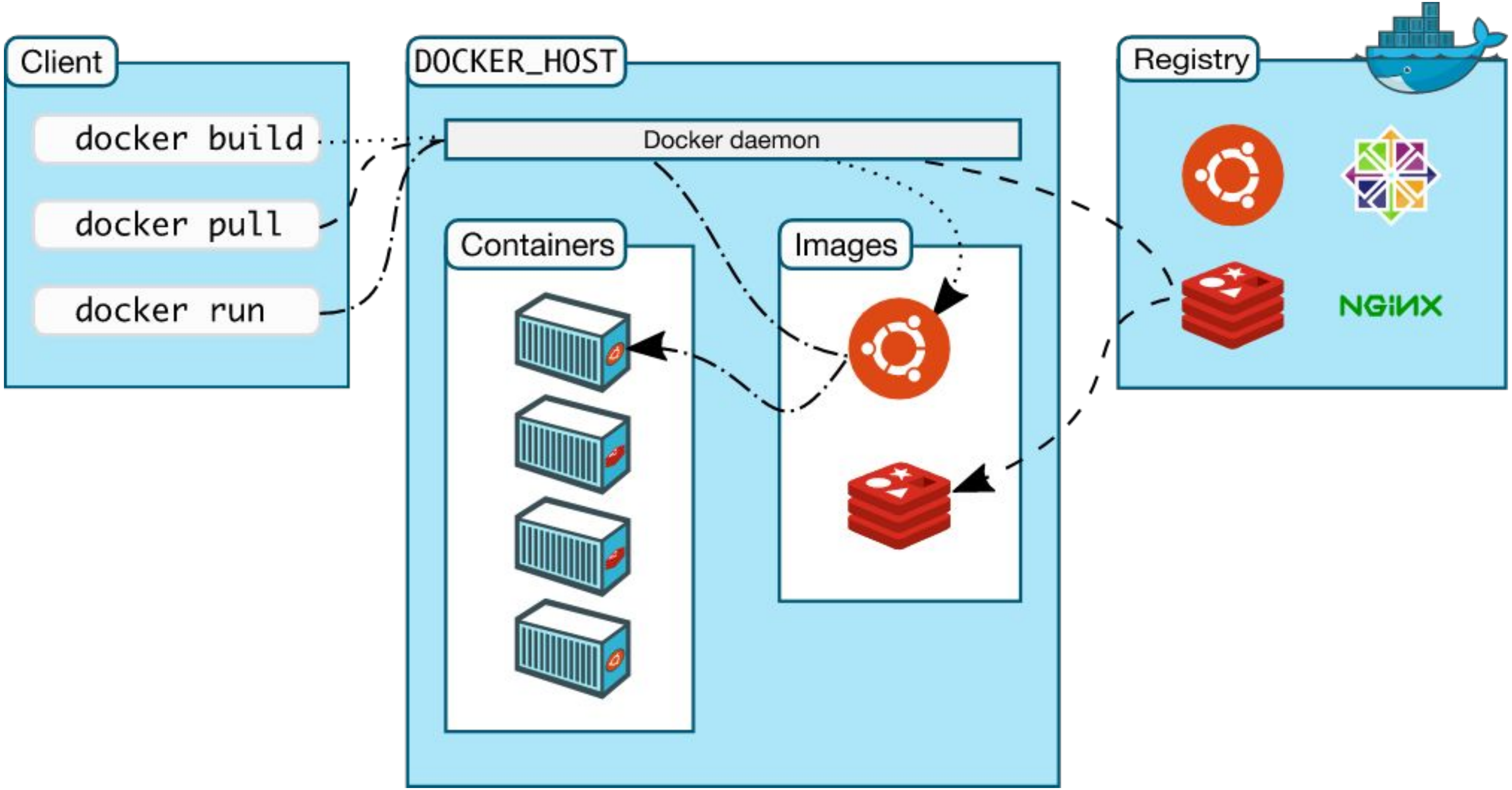
Containerization

- Define/control the environment code executes in independent of the underlying hardware
- Containerization != Docker (at least in theory)
- Docker *Image*: a static software artifact on disk
    - Operating system
    - Installed packages
    - Environment variables
    - Your code
    - An "entrypoint"

Containerization

- Define/control the environment code executes in independent of the underlying hardware
- Containerization != Docker (at least in theory)
- Docker *Image*: a static software artifact on disk
    - Operating system
    - Installed packages
    - Environment variables
    - Your code
    - An “entrypoint”
- Docker *Container*: a dynamic instance of a running image (sort of like a VM)
    - Executes entrypoint
    - Optionally: interactive shell

Containerization

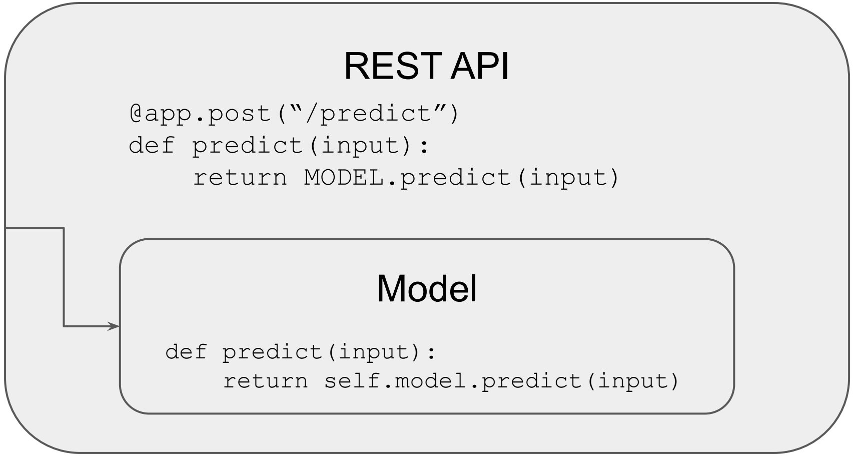


Deployment & Monitoring

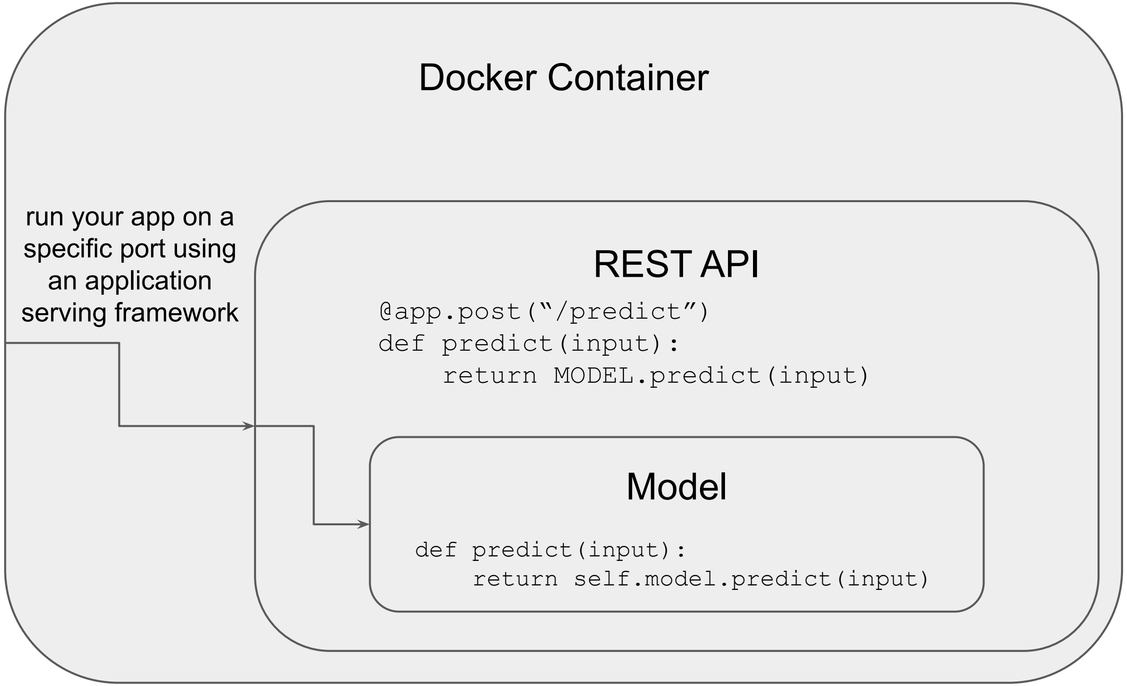
Model

```
def predict(input):
    return self.model.predict(input)
```

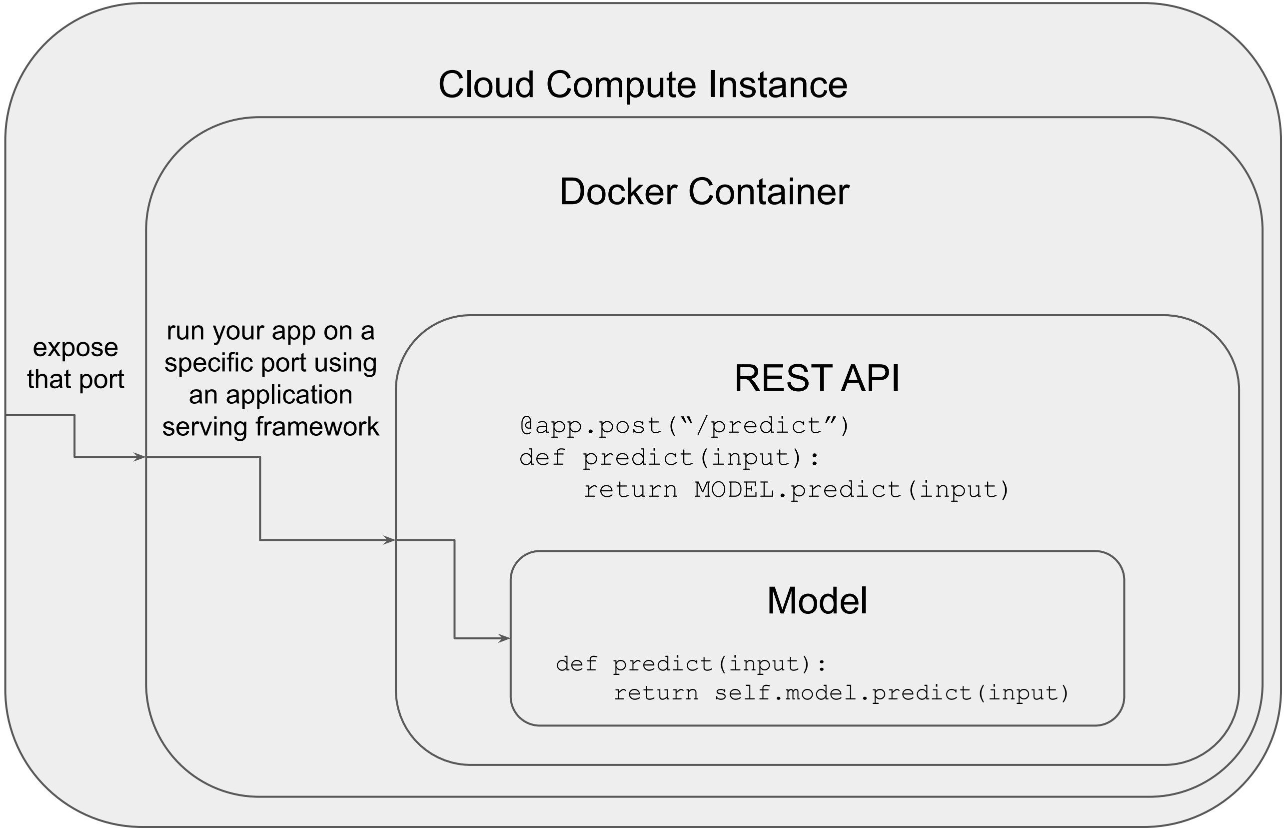
Deployment & Monitoring



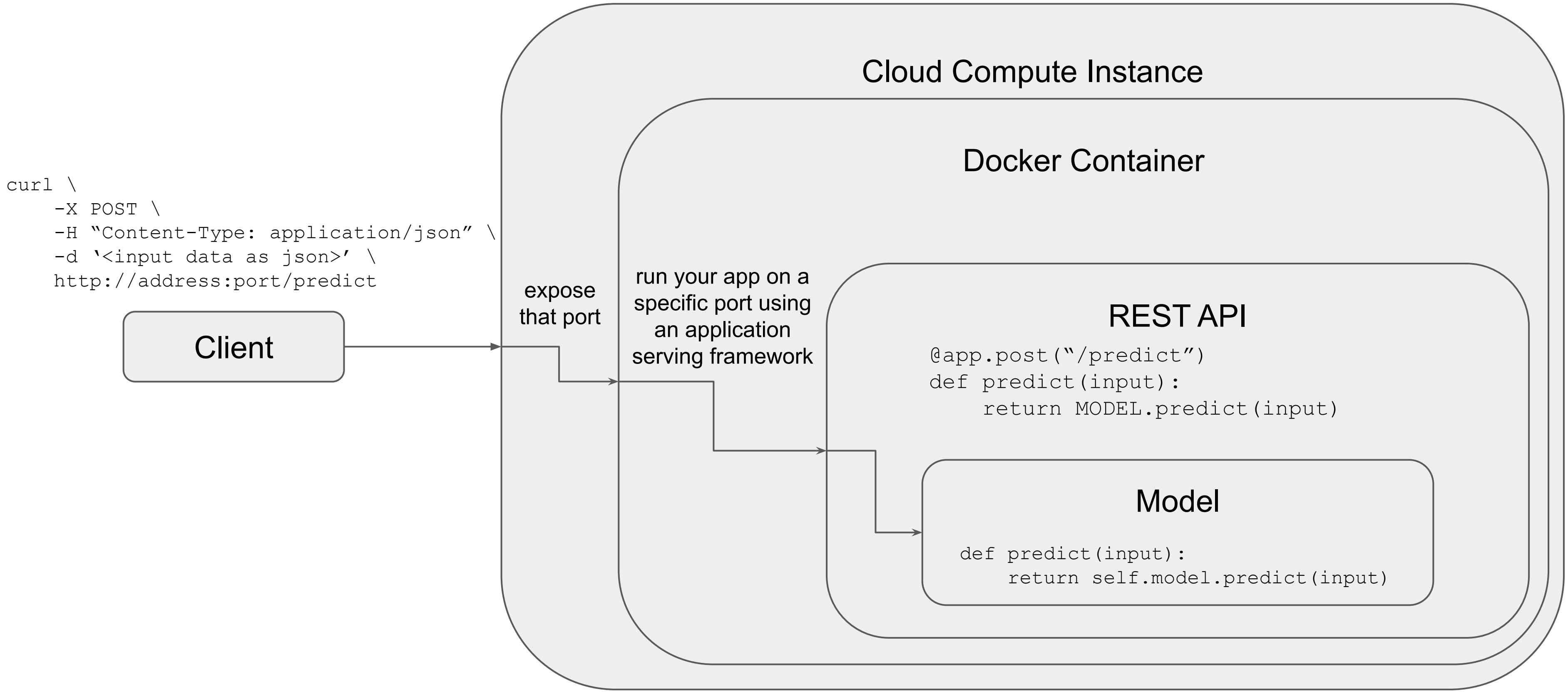
Deployment & Monitoring



Deployment & Monitoring



Deployment & Monitoring



Deployment & Monitoring

- How do we make sure our model is performing as expected?
- How do we know when it's time to retrain our model?
- Things you may want to monitor or log:
    - Prediction latency / application latency
    - Distribution of predictions
    - Distribution of input features
    - Random sample of raw input data
    - Application failures

Deployment & Monitoring

- How do we make sure our model is performing as expected?
- How do we know when it's time to retrain our model?
- Things you may want to monitor or log:
  - Prediction latency / application latency
  - Distribution of predictions
  - Distribution of input features
  - Random sample of raw input data
  - Application failures

Best practice: don't log sensitive data! Any data that may be sensitive should be written to a more secure (cloud) data source.

Deployment & Monitoring

- Cloud providers have specialized “model hosting” services that aim to reduce boilerplate and build-in monitoring capabilities

Deployment & Monitoring

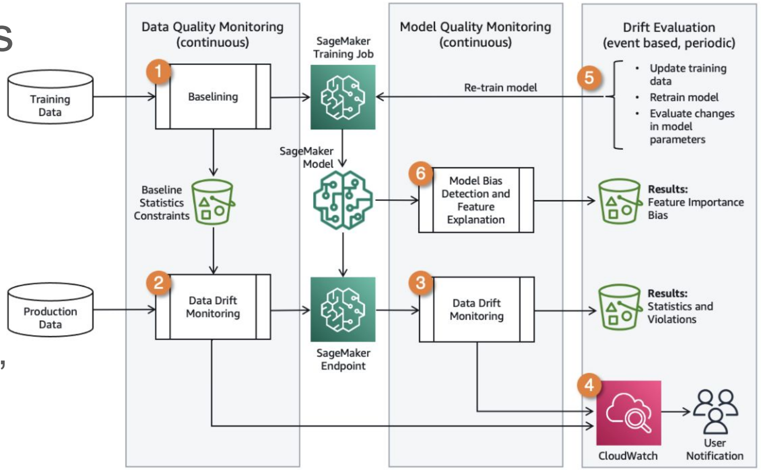
- Cloud providers have specialized “model hosting” services that aim to reduce boilerplate and build-in monitoring capabilities
  - E.g., AWS SageMaker Endpoints require an `inference.py` file with two functions:
    - `model_fn`: load your model
    - `transform_fn`: receive an input, make a prediction, return output

Deployment & Monitoring

- Cloud providers have specialized “model hosting” services that aim to reduce boilerplate and build-in monitoring capabilities
  - E.g., AWS SageMaker Endpoints require an `inference.py` file with two functions:
    - `model_fn`: load your model
    - `transform_fn`: receive an input, make a prediction, return output
    - All other layers (REST API, serving framework, Docker, Compute) are abstracted away
    - Plays especially nicely with SageMaker Training

Deployment & Monitoring

- Cloud providers have specialized “model hosting” services that aim to reduce boilerplate and build-in monitoring capabilities
  - E.g., AWS SageMaker Endpoints require an `inference.py` file with two functions:
    - `model_fn`: load your model
    - `transform_fn`: receive an input, make a prediction, return output
    - All other layers (REST API, serving framework, Docker, Compute) are abstracted away
    - Plays especially nicely with SageMaker Training



Demos

- Remote Jupyter
- Docker 101

Thank you!

Any questions?

proofpoint®

VECTRA®

SECURITY THAT THINKS.®