



Productionizing ML & DL Models At Scale

STAC Summit June 6, 2019

Rajesh Vadlamudi
Solutions Manager

About Paperspace

Our Team

- Founded by Dillon Thompson Erb and Daniel Kobran in 2015
- Based in Brooklyn, NY, with 30+ team members worldwide

Our Goal

- Provide a high-performance cloud infrastructure and integrated development/deployment environment for deep learning



Agenda

1. The State of ML/DL Development
2. Barriers to Entry
3. CI/CD for Traditional Apps vs. ML/DL
4. Emerging Best Practices

“

By 2023, AI and deep-learning techniques will be the most common approaches for new applications of data science”

Alexander Linden

VP, Artificial Intelligence & Machine Learning at Gartner



The State of ML/DL Today

Raw components are here.

- Algorithms
- Massive data streams
- Parallel compute

Currently limited to select few.

Only accessible to Uber/Facebook that have built large pipelines internally.

Primed to enter mainstream.

Every company stands to gain. Most are struggling with putting all the pieces together to start.

Challenges

No tooling, ML-Ops is nascent.

Nothing for developers to plug into. Loss of productivity.

AI hardware is rapidly evolving.

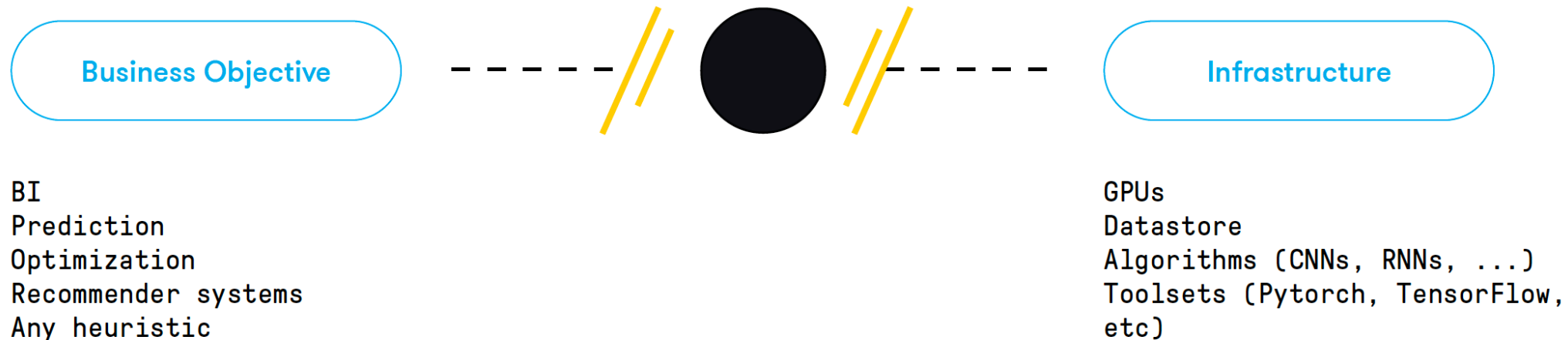
Infrastructure is quickly outdated and difficult to maintain.

Hurdles to productionize.

Significant upfront investment, time to market is 6 to 18 months!

The biggest barrier to AI adoption is an infrastructure and tooling problem, not an algorithm problem

The field of ML / DL / AI has quickly outpaced the tools available to it's practitioners



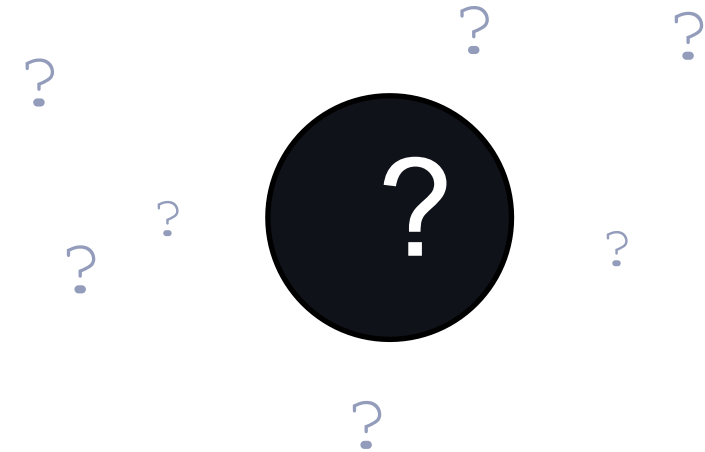
The Software Development ecosystem is rich

Traditional Software



Storage, CDN, deploy, monitor, VPC,
Load balance, IPsec, CI/CD, DNS...

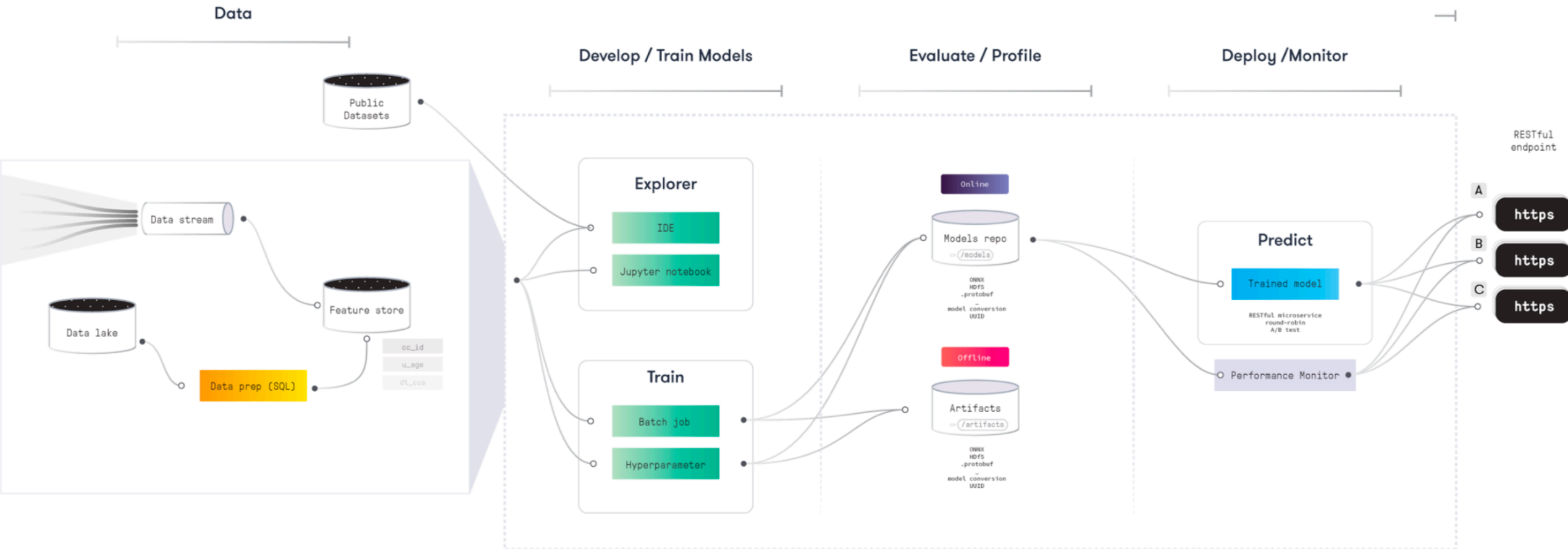
Deep Learning



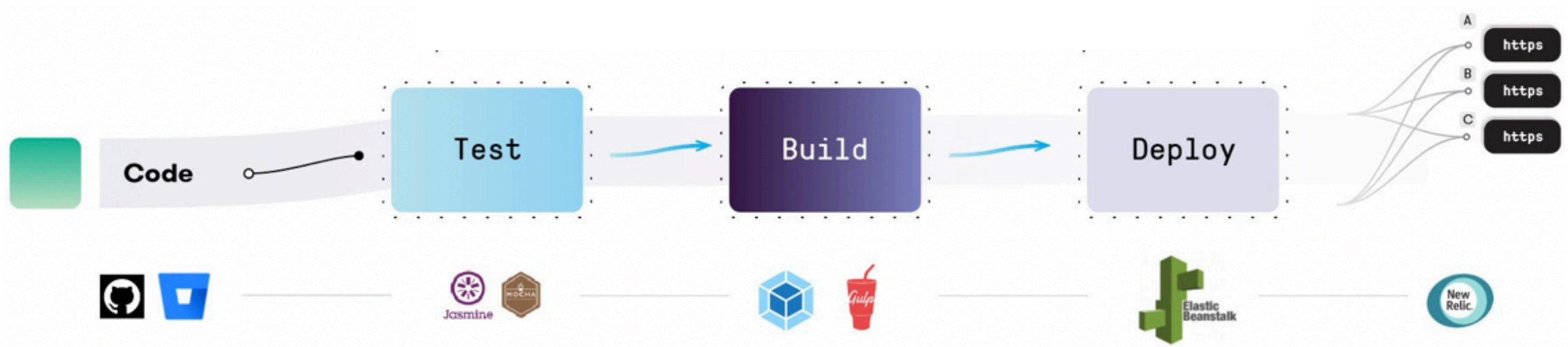
data, notebooks, train, visualize,
collaborate, version, hyperparameters ...

ML engineers spend 80% of their time managing infrastructure and tooling.

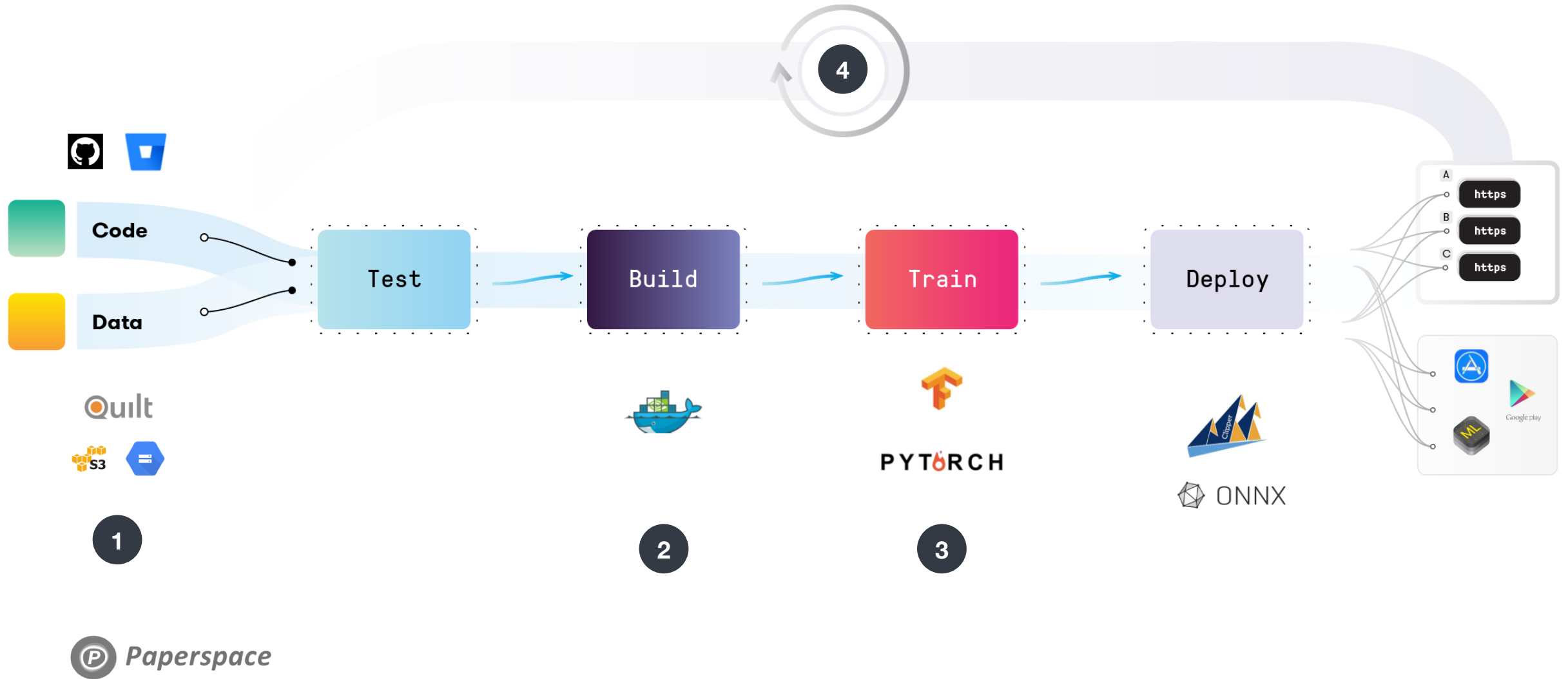
The ML/DL Development Lifecycle



CI/CD Workflow for Traditional Apps



CI/CD Workflow for ML/DL



Semantics

Input

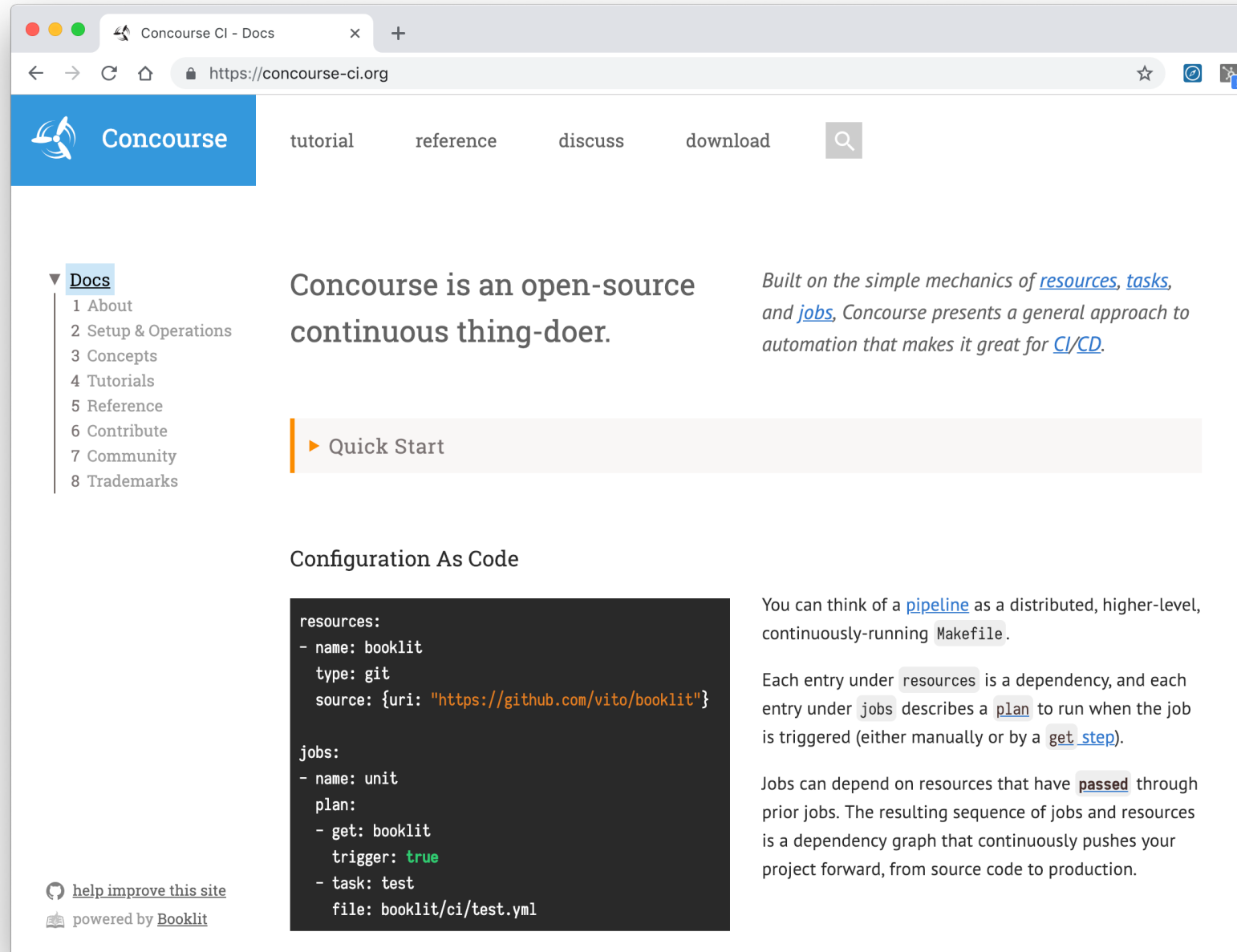
1. Data governance
2. Data versioning

Triggers

1. Data drift
2. Model drift

Output

1. Artifact management
2. Multi-armed bandit
3. Edge vs datacenter



The screenshot shows the Concourse CI website. The browser address bar displays 'https://concourse-ci.org'. The page has a blue header with the Concourse logo and navigation links: 'tutorial', 'reference', 'discuss', and 'download'. A search icon is also present. On the left, a sidebar menu lists the following items: 'Docs' (selected), '1 About', '2 Setup & Operations', '3 Concepts', '4 Tutorials', '5 Reference', '6 Contribute', '7 Community', and '8 Trademarks'. The main content area features the heading 'Concourse is an open-source continuous thing-doer.' followed by a 'Quick Start' button. Below this, the 'Configuration As Code' section displays a YAML configuration snippet for a pipeline named 'unit'. To the right of the code, explanatory text describes the 'resources' and 'jobs' sections of the configuration. At the bottom left, there is a link to 'help improve this site' and a note 'powered by Booklit'.

Concourse CI - Docs

https://concourse-ci.org

Concourse

tutorial reference discuss download

▼ Docs

- 1 About
- 2 Setup & Operations
- 3 Concepts
- 4 Tutorials
- 5 Reference
- 6 Contribute
- 7 Community
- 8 Trademarks

Concourse is an open-source continuous thing-doer.

Built on the simple mechanics of [resources](#), [tasks](#), and [jobs](#), Concourse presents a general approach to automation that makes it great for [CI/CD](#).

► Quick Start

Configuration As Code

```
resources:  
- name: booklit  
  type: git  
  source: {uri: "https://github.com/vito/booklit"}  
  
jobs:  
- name: unit  
  plan:  
  - get: booklit  
    trigger: true  
  - task: test  
    file: booklit/ci/test.yml
```

You can think of a [pipeline](#) as a distributed, higher-level, continuously-running Makefile.

Each entry under `resources` is a dependency, and each entry under `jobs` describes a [plan](#) to run when the job is triggered (either manually or by a [get step](#)).

Jobs can depend on resources that have [passed](#) through prior jobs. The resulting sequence of jobs and resources is a dependency graph that continuously pushes your project forward, from source code to production.

[help improve this site](#)

powered by [Booklit](#)

Observations

1. Tooling will change to accommodate more complicated semantics.
2. Binary (pass/fail) will give way to richer abstractions.
3. Tools will go wide and deep to close the ML-Ops feedback loop.
4. From a hardware perspective, the edge and the datacenter will merge.

Update config.yaml #2
dte wants to merge 6 commits into master from config-version-update

gradientci-dev bot commented 6 hours ago • edited

Paperspace Experiment Report

Project prpp97zs7
Experiment es1vquob8hxrdo

User Defined Checks

Metric	Aggregate	Value	Success	Summary
tensorflow:loss	max	0.025719	✓	Metric tensorflow:loss/max is in range 0.000..1.000.
tensorflow:accuracy	median	0.993400	✓	Metric tensorflow:accuracy/median is in range 0.990..1.000.

Experiment Details

Experiment Parameters for Single Node

Parameter	Value
Machine Type	
Server Count	1
Command	""
Container	

Job Environments

Job 0

Parameter	Value
Handle	jswzesw8l8973u
GPU Name	Tesla K80
GPU Serial	0320617087317
GPU Device	/dev/nvidia0
GPU Driver	410.48
Hostname	gradient-host-1553793082-82415ed3
CPU Count	2
CPU Model	Intel(R) Xeon(R) CPU @ 2.30GHz
Host Memory	12297216 kB

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe
You're receiving notifications because you authored the thread.

2 participants

Lock conversation

Emerging Trends

- Jupyter
 - Standard for learning. Seeing traction for training and some deployments.
- Containers and Kubernetes
 - Silo concerns of data scientists and dev ops
 - K8s for container orchestration
- Model drift monitoring
 - Monitoring several models and their performance over time.
- Rollback models and model repos
 - Input new data to existing models
 - Pre-trained models for emerging use cases specific for your business.

In the Field: Uber's Michelangelo Platform

End-to-End Workflow

Build, train, refine,
and deploy models

Model Developer Velocity

Innovation comes
from lots of iteration.

ML-as-Engineering

Apply CI/CD methods
from software to ML

Modular & Tiered

Assembled for
specialized use cases.



MANAGEMENT LAYER

Access Control Collaboration Activity Tracking Governance Alerting Resource Controls

USER LAYER

Interactive Workspace

Jupyter Notebooks,
RStudio, etc.

CLI & Python Library

Run experiments, track results,
define end-to-end pipelines

Deployment Engine

Deploy models into
production, scale & monitor.

Version Control Git Integration Graphing Hyperparameter Tuning Dataset Repo Pipeline Definitions

FRAMEWORK LAYER

Frameworks



Keras

PYTORCH

Pytorch



TensorFlow



CUDA

cuDNN

cuDNN



Docker

Drivers & Dependencies

ORCHESTRATION LAYER

Job scheduling Data Orchestration Container Registry Cluster Manager Unified Logging Artifact Management

COMPUTE LAYER



Cloud

All major providers

On-prem



Kubernetes



Virtual Machine



Bare-metal

Build, train, and deploy your ML/DL models at scale.

Learn more at www.gradient.cloud
Email me at raj@paperspace.com

Thanks for listening!

