Conda, Docker and Kubernetes: The Cloud Native Future of Data Science

Mathew Lodge
SVP Product, Anaconda
Who Am I?

• 25+ year career in tech
• Wrote code that flew (flies?) on ISS and Boeing 777. Connected 6 countries to the Internet in the early 1990s.
• Schlumberger, Cisco, Symantec, VMware and a number of start-ups in between
• Governing Board Member, Cloud-Native Computing Foundation (CNCF) 2015-16

Mathew Lodge
SVP Products and Marketing
Anaconda
Fundamental Data Science Problem: How To Go Faster
New Data Science Challenge Is Deployment

Data Engineering
- Acquisition
- Processing
- Governance

Data Science
- Data Wrangling
- Visualization and Analysis
- Model development
- Reports and Dashboards

Deployment
- Model training
- Serve & Manage

© 2018 Anaconda - Confidential & Proprietary
**What Is Cloud Native?**

Not a place, but a way to do computing: How Google, Netflix, Amazon and others work today

<table>
<thead>
<tr>
<th>1. Container-based</th>
<th>[Docker] Container as the unit of isolation and scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. API-oriented</td>
<td>Loosely-coupled components talk via APIs in a distributed system</td>
</tr>
<tr>
<td>3. Dynamically orchestrated</td>
<td>Applications are dynamic and organic: they grow, shrink and adapt</td>
</tr>
</tbody>
</table>

Run in your data centers or public cloud
Cloud Native Impact On Software Development

Cloud Native and DevOps leaders vs laggards*

• 46x more frequent deployment
• 96x faster MTTR
• 66% lower failure rate

* Puppet Labs 2017 State of DevOps report
Cloud Native: Container-based

- Repeatable, standardized
- Predictable behavior
- Starts in seconds
- Scales out (not up)

*NB: Not a YARN container!*
Dockerfile is the Container “Recipe”

FROM continuumio/miniconda3

RUN apt-get update && apt-get install -y \ 
libpq-dev build-essential && rm -rf /var/lib/apt/lists/*

ENTRYPOINT [ "/bin/bash", "-c" ]

# Use the environment.yml to create the conda environment.
ADD environment.yml /tmp/environment.yml
WORKDIR /tmp
RUN [ "conda", "env", "create" ]

ADD . /code
WORKDIR /code/shared
RUN [ "/bin/bash", "-c", "source activate your-environment && python setup.py develop" ]
Before API orientation: 3-Tier Architecture

Old school web application architecture

- Apache
- Relational Database
- Store UI
- Accounting
- Credit card
- Inventory
- Shipping
Cloud Native: API-Oriented

API-oriented (microservices) architecture web application:

- Mobile API
- Web UI
- Recommend
- Cart
- Order
- Accounting
- Credit card
- Inventory
- Shipping

Languages:
- Java
- Python
- Go

Storage

© 2018 Anaconda, Inc. All Rights Reserved.
Cloud Native: Dynamically Orchestrated

**Objective**

- Edit a file in Jupyter?
- Run a Spark DB query?
- Train a model?
- Run a job?
- Deploy a model?
- Upgrade a model?
- Downgrade a model?
- Scale up a model?
- Scale down a model?

**Orchestrator Actions**

- Start containers
- and/or
- Stop containers
Example: Upgrade a Model

- **Container v1**
  - Application:
    - REST API
    - Model code v1
  - Dependencies:
    - Packages: conda, .CSV files
    - GPU driver

- **Container v2**
  - Application:
    - REST API
    - Model code v2
  - Dependencies:
    - Packages: conda, .CSV files
    - GPU driver

- Operating System
- Hardware
Old School: Incremental Patching
Cloud Native: No Patching
Example: Scale Up

Load Balancer

Container v1
- Application
  - Model code v1
  - REST API
- Dependencies
  - Packages: conda
  - GPU driver: .CSV files

Container v1
- Application
  - Model code v1
  - REST API
- Dependencies
  - Packages: conda
  - GPU driver: .CSV files

Container v1
- Application
  - Model code v1
  - REST API
- Dependencies
  - Packages: conda
  - GPU driver: .CSV files
Example: A/B Test

Smart Load Balancer

49% 49% 2%

Container v1
Application
Model code v1
REST API
Dependencies
Packages
conda
GPU driver
.CSV files

Container v1
Application
Model code v1
REST API
Dependencies
Packages
conda
GPU driver
.CSV files

Container v2
Application
Model code v2
REST API
Dependencies
Packages
conda
GPU driver
.CSV files
Example: Champion / Challenger

Smart Load Balancer

50% 50% Duplicate 50%

Container v1
Application
Model code v1
REST API

Dependencies
Packages conda
GPU driver .CSV files

Container v1
Application
Model code v1
REST API

Dependencies
Packages conda
GPU driver .CSV files

Container v2
Application
Model code v2
REST API

Dependencies
Packages conda
GPU driver .CSV files
2016: Container Orchestrator Wars
How That Played Out (Google Trends)
2018: Kubernetes Everywhere
Kubernetes Architecture

Things Kubernetes Provides

- Health checks and restarts on failure
- Cluster scaling
- Container networking
- L7 load balancing
- Versioned deployments
- Jobs
- Autoscaling
- Access control
- Scheduling constraints (e.g. affinity / anti-affinity)
Kubernetes Is Declarative

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
labels:
  app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

Makes it easy to return cluster to correct state in presence of
• Failed nodes
• Temporarily disconnected nodes
• Retired nodes
• New nodes
• All of the above at the same time

Also: Kubernauts learn to love YAML
Hadoop “Big Data” vs. Cloud Native

Hadoop: Yahoo’s 2005 interpretation of Google’s 2004 MapReduce paper

“Big Data”
1. Java-based
2. MapReduce-oriented
3. Batch orchestrated

Cloud Native
1. Container-based
2. Loosely coupled API-oriented
3. Dynamically orchestrated
Java-Centric Is a Problem in 2018

88% of execution time spent serializing/de-serializing the data
Map-Reduce: Hadoop’s Hammer

The overall MapReduce word count process

Input

Splitting

Mapping

Shuffling

Reducing

Final result

Deer Bear River
Car Car River
Deer Car Bear

Deer, 1
Bear, 1
River, 1

Car, 1
Car, 1
River, 1

Car, 1
Car, 1

Deer, 1
Deer, 1

River, 1
River, 1

Bear, 1
Bear, 1

Bear, 2

Car, 3

Car, 3

Deer, 2

Deer, 2

River, 2

Bear, 2

Car, 3

Deer, 2

River, 2
But… Today’s ML Doesn’t Fit MapReduce Well

- Google moved on from MapReduce
- Now uses data flow graphs
  - E.g. TensorFlow
Credit Risk Model Example (Using Anaconda Dask)

https://www.anaconda.com/blog/developer-blog/credit-modeling-with-dask/
## Anaconda Enterprise: Kubernetes And Containers

### Anaconda Enterprise

<table>
<thead>
<tr>
<th>User Interface</th>
<th>UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Auth</td>
</tr>
<tr>
<td>AE Services</td>
<td>Deploy</td>
</tr>
<tr>
<td></td>
<td>Sessions</td>
</tr>
<tr>
<td></td>
<td>Repository</td>
</tr>
<tr>
<td>Database / Storage</td>
<td>Postgres</td>
</tr>
</tbody>
</table>

© 2018 Anaconda, Inc. All Rights Reserved.
Example: Simple Model Deployed On AE5

API request

Proxy container

Docker Container

Your ML model
REST API wrapper
Your Model Code

Dependencies
Packages
conda
GPU driver
.CSV files

Docker container
AE5 Authentication API
Authentication Service

Dependencies
Packages
conda
Libraries
Config

© 2018 Anaconda, Inc. All Rights Reserved.
Accelerate Your Data Science Lifecycle With Cloud Native
Questions?

@mathewlodge