Modern Software for Data Science

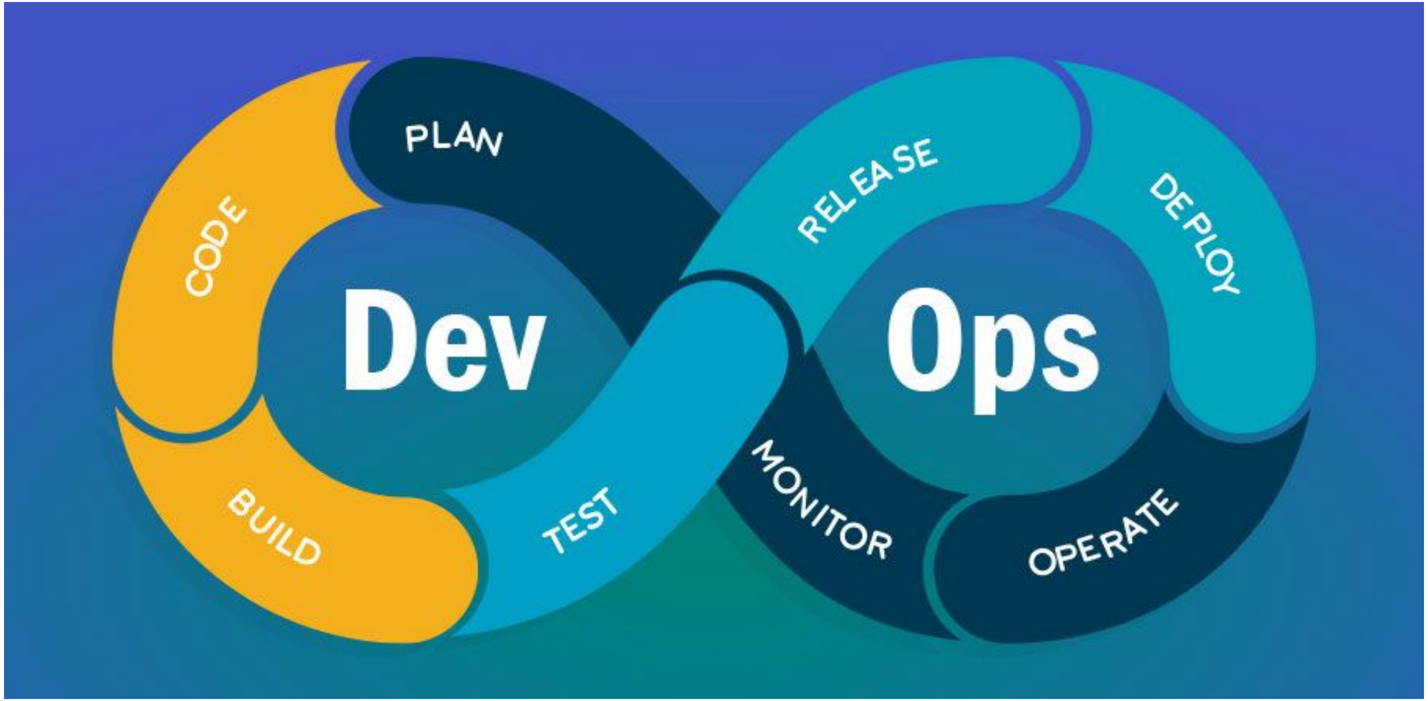
Michael Reid

About Me - Mike Reid

- B.S. Mathematics from UMBC
- 10 years' software development experience
- Currently software development manager at AWS (RDS)
- Formerly software engineer for machine learning team at Splunk
- Software engineering experience at Hootsuite, Global Relay, USDA, FAA

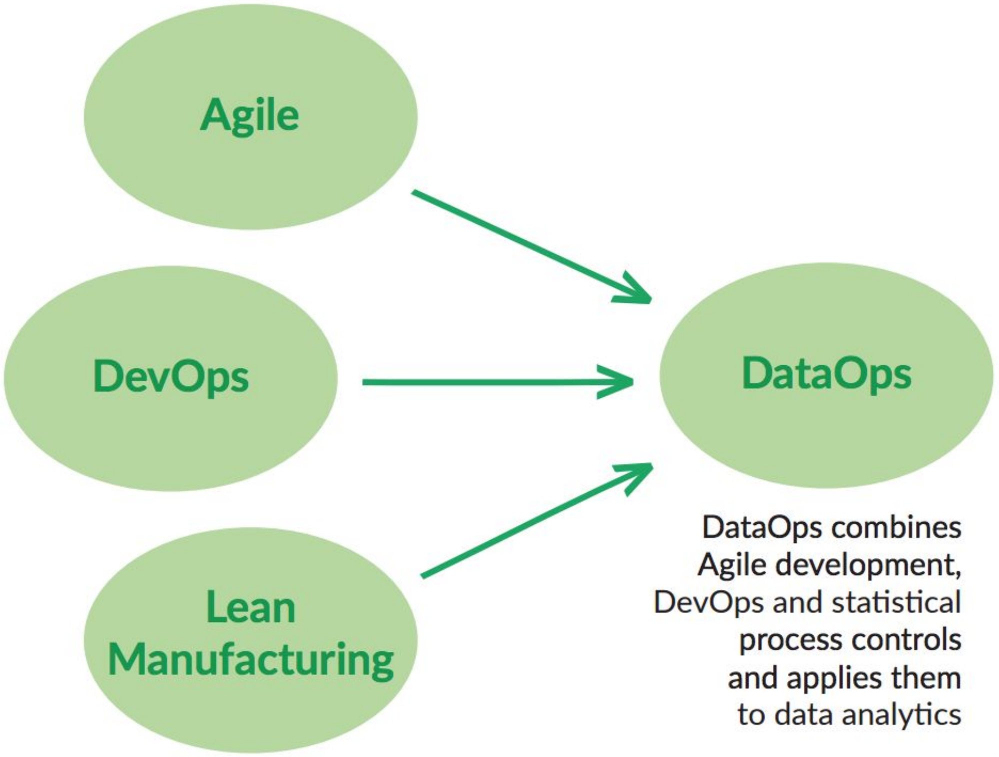
Development and Ops, the Old Days

DevOps


PLAN
CODE
Dev
BUILD
TEST
RELEASE
DEPLOY
Ops
OPERATE
MONITOR

Data Scientists and Ops, Present Day

DataOps?



DataOps combines
Agile development,
DevOps and statistical
process controls
and applies them
to data analytics

Continuous Integration

- What testing do you do today before you push models and algorithms to production?
- Automate it

Continuous Integration - Testing Ideas

- Input data handling
    - Missing/incorrectly typed fields are handled correctly
    - Clearly invalid values are handled correctly
    - Abnormally large data is handled correctly
    - Out-of-order events are handled correctly
- Model performance changes
    - Test set accuracy
    - Edge case or adversarial data handling

Continuous Integration Tools

- Jenkins - de facto standard
- Bamboo - Atlassian stack
- TeamCity - JetBrains offering
- TravisCI/CircleCI - hosted solutions, Github integration

Version Controlling Code

- git

Version Controlling Data?

- No great options
  - git (LFS)
  - Dropbox
  - Amazon S3
- Recommendation - store data externally and immutably, version control lists of data used in training/validation/test sets
  - This is what git LFS does.

Immutability

- Being able to consistently reproduce results has been a tenet of science for years, and now software development is catching up
- If the software artifacts you had running in production disappeared, could you create an *exact* copy?
  - Requires mature version control + build process + configuration management

Configuration Management

- Fundamental problem: The development data store is different than the production data store, but you’re running the same artifact
- Typical solution: store configuration that changes per environment in a config file
- Follow-up problem: How do you manage the values in that configuration file?
- *Configuration Management* - use a tool which stores *configuration as code*, in version control, to control your per-environment configuration

Configuration Management Tools

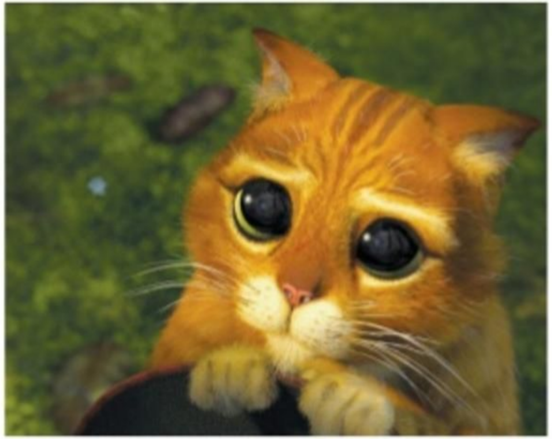
- Ansible - most popular despite being newest
- Puppet, Chef - require DSL knowledge and setup
- Salt (aka Saltstack) - simpler, but less powerful

Infrastructure Management

- Thought experiment
    - *If you went into the office one day and found out your servers had been wiped clean, how difficult would it be to get everything back to the way it was?*

Cattle, Not Pets

- Pets have names, require special care, and need individual attention
- Cattle are numbered and replaceable (and delicious)



To The Cloud

- On the cloud, every server **must** be a cattle
  - Your servers can and will be replaced
- AWS, GCP and Azure all have APIs and automation tools for server management

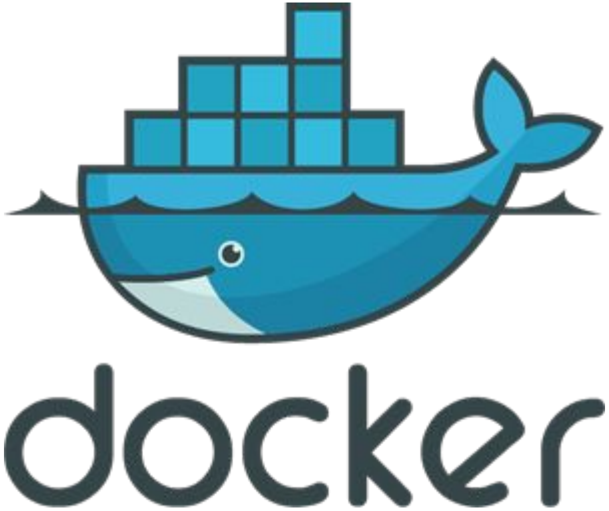
amazon web services

Google Cloud Platform

Microsoft Azure

Docker

- Infrastructure immutability
- Every package, library, etc is built into a single compact image that can be distributed to any system
- Configuration is managed through environment variables or external sources



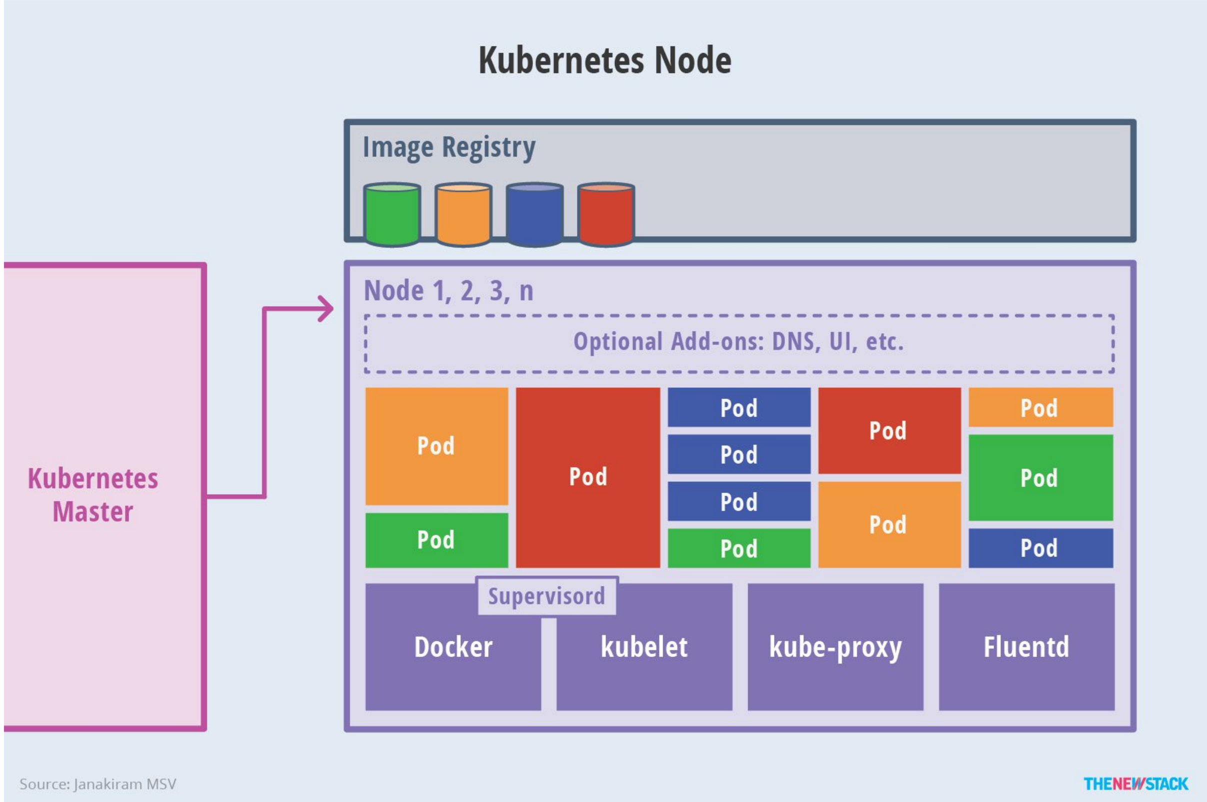
Docker - Why?

- No more version differences of system libraries in dev/staging/prod
- Simple packaging - one artifact that can be deployed and run anywhere
- Minimal time to boot; extremely low overhead compared to virtual machines

Kubernetes

- An orchestration layer that sits on top of containers (i.e. Docker)
- Provides mechanisms outside of your immutable infrastructure for maintaining your runtime: service discovery, service mesh, complex routing rules, etc.
- In a sense, a cloud without a cloud

Kubernetes - In Detail

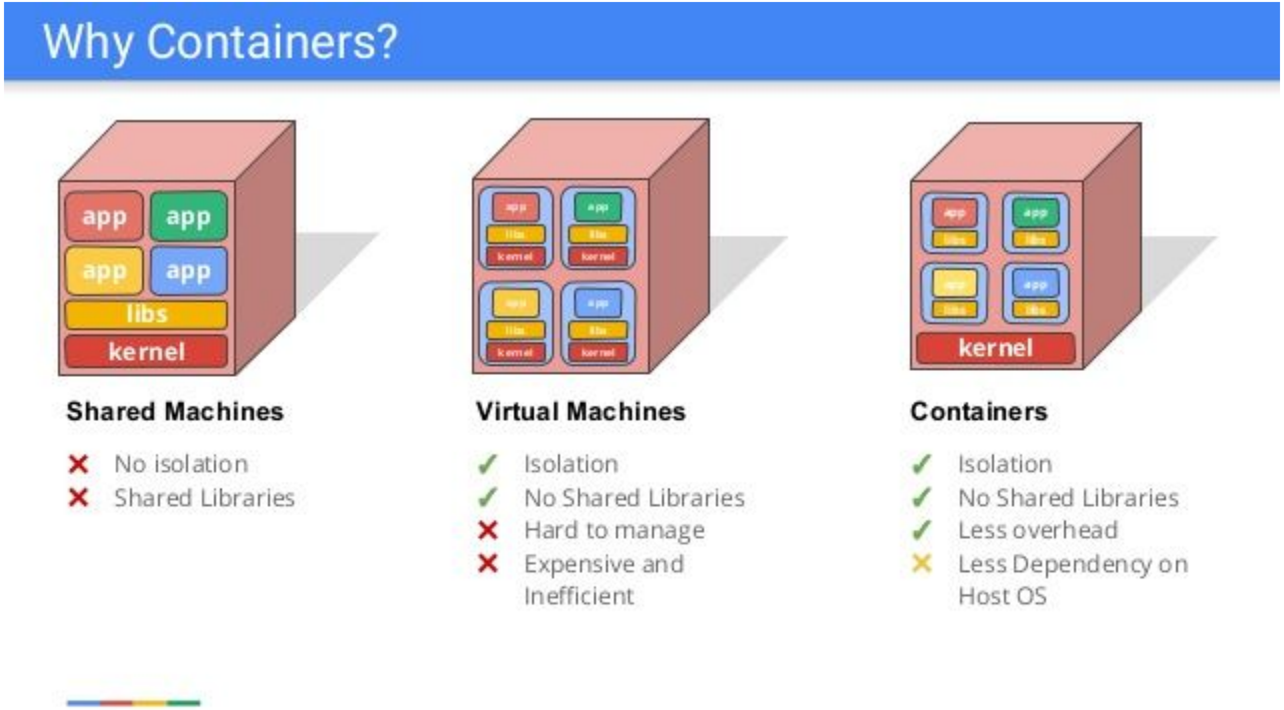


Source: Janakiram MSV

What’s the point of it all?

- Provides a clear separation of ownership between data scientists and operations (IT)
    - Operations owns and operates the underlying Kubernetes infrastructure and physical hardware
    - Data Science organization owns the containers and software running therein
- Fewer touch points allows for higher agility for both data science teams and operations

To Summarize...



Why Containers?

Shared Machines

- ✗ No isolation
- ✗ Shared Libraries

Virtual Machines

- ✓ Isolation
- ✓ No Shared Libraries
- ✗ Hard to manage
- ✗ Expensive and Inefficient

Containers

- ✓ Isolation
- ✓ No Shared Libraries
- ✓ Less overhead
- ✗ Less Dependency on Host OS

Monitoring and Operations

- If a server falls in the woods, who fixes it?
- With great power comes great responsibility
  - Developers and data scientists take ownership of the operation of their software

envoy

sensu

Nagios®

pagerduty

(x) matters®

Warning - Data Platforms Can Lock You In

- Splunk - tough to spin up clusters in an automated fashion
- Spark - job submission and management is tricky to automate
- Amazon ML, Azure ML, Google ML, IBM Watson - all have tight vendor lock-in by default, need to devise workflows to be cloud-agnostic
- If your platform isn’t built with DataOps in mind, it’s hard to adjust

The Future - Stream Processing

- Spark Streaming - not really streaming
- Apache Flink and Apache Apex - next-generation native streaming platforms
- Streaming ML algorithms are not as common or frequently used

The Future - Secure Machine Learning

- Problem: You want to train a model, but you don’t want to buy expensive GPUs
- Solution: Train on the cloud!
- Problem: Your data is sensitive. You can’t put it on the cloud
- Solution: ???

Homomorphic Encryption

- *Fully Homomorphic Encryption* are cryptosystems that allow arbitrary computations on ciphertext
  - These cryptosystems exist, but are slow
- *Partially Homomorphic Encryption* supports certain operations on ciphertexts
  - Addition or multiplication, for example
- If you can perform encrypted multiplication and encrypted addition, you can do encrypted linear algebra

Thank You!

- Michael Reid
- reidmichaeljames@gmail.com
- https://github.com/mjreid