Utilizing Container Technology to Streamline Data Science

2018 University of California and Lawrence Livermore National Laboratory Inaugural Data Science Workshop

Dr. Amanda J. Minnich
minnich2@llnl.gov

August 7-8, 2018

LLNL-PRES-751448
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Who am I?

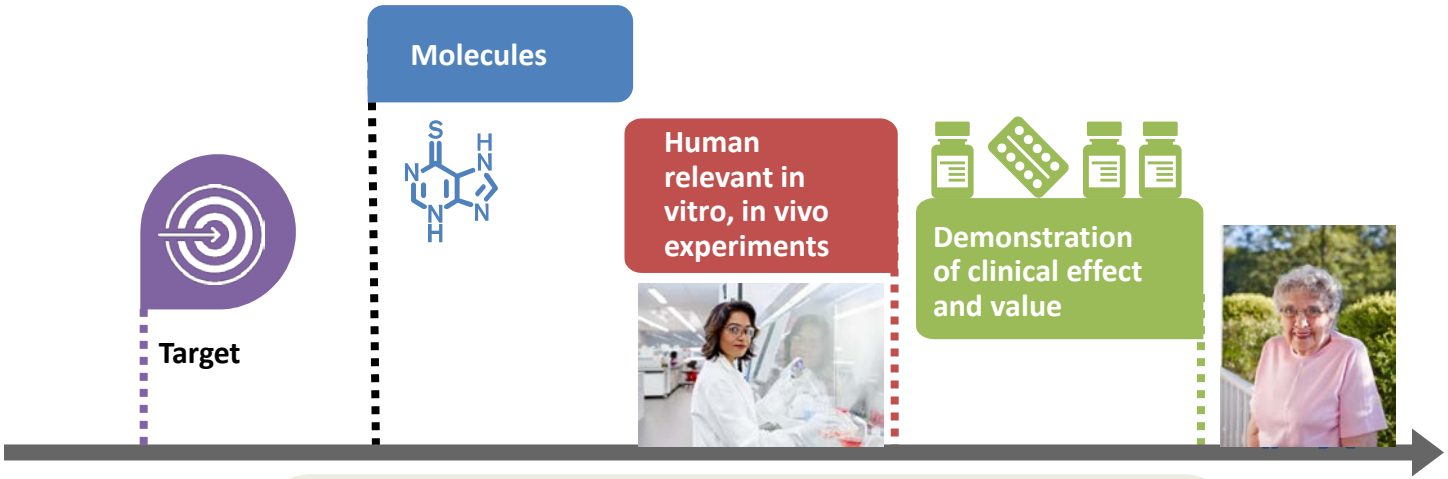
- Staff Research Scientist at LLNL
- Background in Biology and Computer Science
- Co-Lead of the ATOM Consortium's Data-Driven Modeling Team

What is the ATOM Consortium?

- ATOM: Accelerating Therapeutics for Opportunities in Medicine
- Public-private consortium whose goal is to innovate the drug discovery pipeline
- Founding Members:
  - LLNL
  - GlaxoSmithKline
  - Frederick National Lab (National Cancer Institute)
  - UCSF

Current drug discovery: long, costly, & high failure

Is there a better way to get medicines to patients?



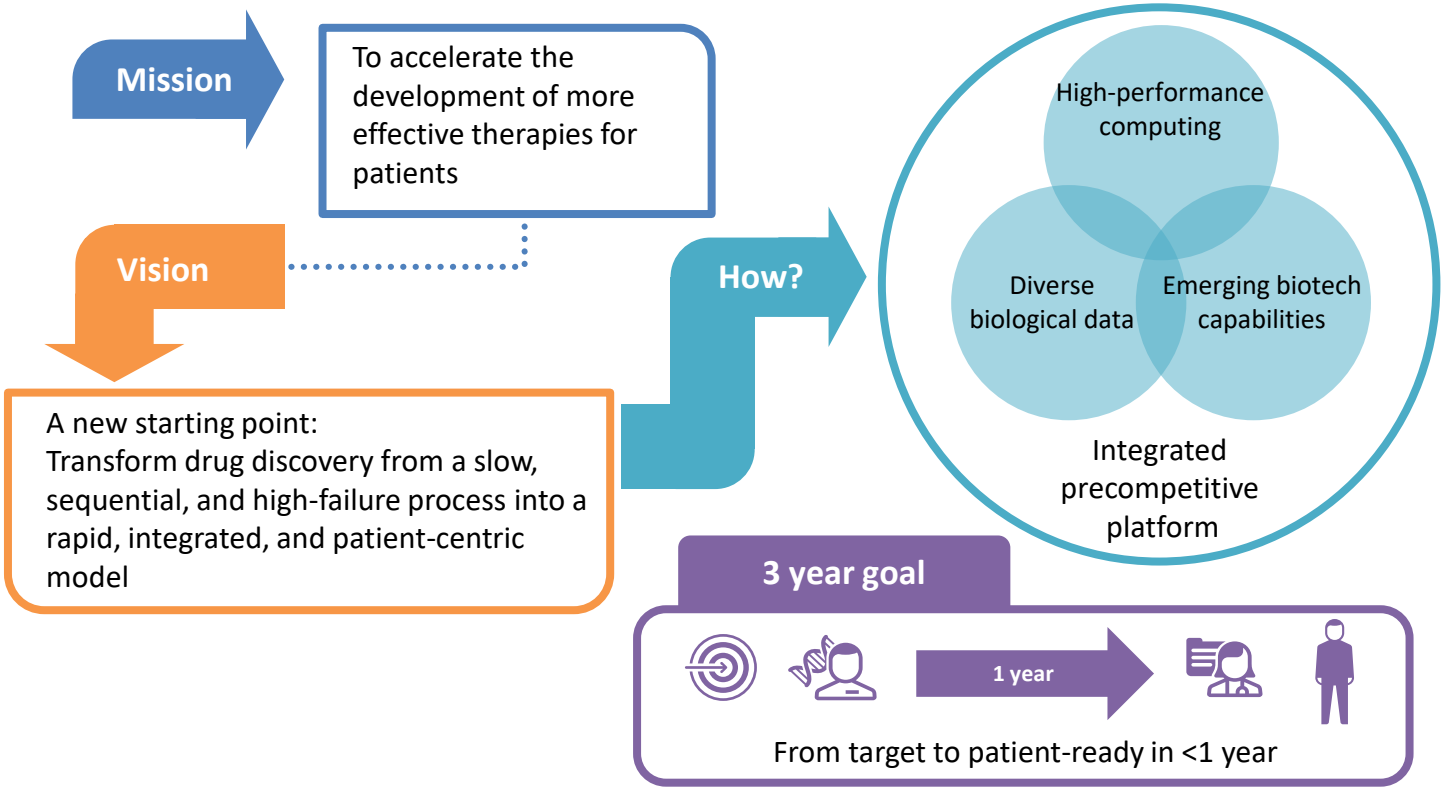
Preclinical issues:

- Average time: 5.5 years
- 33% of total cost of medicine development
- Millions of molecules tested, 1000s made, and most fail
- Clinical success rates still only 12%, indicating poor translation in patients

Source: http://www.nature.com/nrd/journal/v9/n3/pdf/nrd3078.pdf

ATOM Consortium

Accelerating Therapeutics for Opportunities in Medicine



ATOM has unique requirements

- Reproducibility and traceability of data processing and model building
- Ability to keep up with rapidly-changing data science libraries
- Easy-to-access development server for pharmacology domain experts
- Data services to handle fairly large, disparate, messy datasets

It was clear the existing setup would not support these requirements, so we had to look elsewhere for solutions.

Requirements and Tools

Requirements	Tools
Environment consistency	Containers
Easy software updates	Containers
Orchestration of resources (including GPUs)	Kubernetes
Role-based access to mounted volumes	Kubernetes
Intuitive GUI	JupyterLab
Data services (SQL, noSQL, metadata annotation)	Containers + RESTful API

Requirements and Tools

Requirements	Tools
Environment consistency	**Containers**
Easy software updates	**Containers**
Orchestration of resources (including GPUs)	Kubernetes
Role-based access to mounted volumes	Kubernetes
Intuitive GUI	JupyterLab
Data services (SQL, noSQL, metadata annotation)	Containers + RESTful API

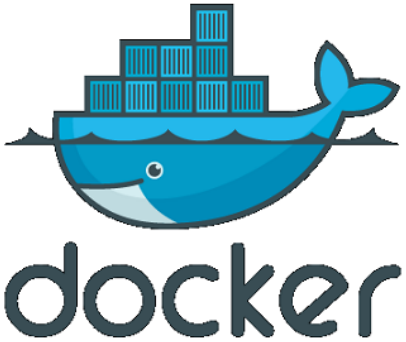
Features of containers

- Containers allow you to:
  - Package up code and dependencies together
  - Create new environments with different software versions without needing server admin privileges
  - Version a file system via tags, ensuring traceability of work and reproducibility of results
- Much more lightweight than VMs: can put lots of them on a single host operating system

Underlying philosophy: if it works on my machine it will work on yours.

What is docker?

- docker is a an open-source project based on Linux containers.
- Basically acts as helpful tool for packing, shipping, and running applications within these containers.
- Adheres to the Open Container Initiative (OCI), which means it is supported by standard container orchestration tools like Kubernetes
- Caches image layers, saving huge amounts of time for developers



Containers are ideal for two of ATOM's requirements

1. Environment consistency:
   - Image tags allow tracking of different software version environments
   - For example: atom-app:deepchem1.2, atom-app:deepchem1.3, atom-app:deepchem2.0

2. Easy software updates:
   - To install a new version of a library, simply update dockerfile, rebuild, and push image
   - Allows installation of new software without having root on the remote server

Requirements and Tools

Requirements	Tools
✓ Environment consistency	Containers
✓ Easy software updates	Containers
Orchestration of resources (including GPUs)	**Kubernetes**
Role-based access to mounted volumes	**Kubernetes**
Intuitive GUI	JupyterLab
Data services (SQL, noSQL, metadata annotation)	Containers + RESTful API

Orchestrating containers

- Once you have your containers, you need to handle:
  - User access
  - Directory access
  - Resource access
- Docker compose works for local development, but to coordinate many users and many resources, need a more heavy duty orchestrator
- We chose to use Kubernetes for this task

What is Kubernetes?

- Kubernetes is an open source container orchestrator, used to manage containerized workloads and services
- Combines simplicity of Platform as a Service (PaaS) with flexibility of Infrastructure as a Service (IaaS)
- Developed by Google, who has been running containers in production for over 10 years

kubernetes
by Google

We use Kubernetes to:

- Orchestrate the allocation of compute resources: CPU, GPU, and memory
- Manage access to container-based data services
- Handle Role-Based Access Control (RBAC) to various mounted volumes

```
➜  data_science git:(master) ✗ kubectl get pods -n minnich2
NAME                        READY     STATUS    RESTARTS   AGE
atom-app-dc2tf15            1/1       Running   0          9d
atom-app-nogpu-dc2bokeh     1/1       Running   0          7d
atom-app-nogpu-r            1/1       Running   1          8d
```

Can mount external directories into container

atom-app-pod.yaml

```yaml
# What is a Pod?  http://kubernetes.io/docs/user-guide/pods/
# PodSpec: http://kubernetes.io/docs/api-reference/v1/definitions/#_

apiVersion: v1
kind: Pod
metadata:
  name: atom-app-dc2tf15
  labels:
    app: atom-app-dc2tf15-sel
spec:
    volumes:
        - name: datastore
          hostPath:
              path: /data
        - name: projdata
          hostPath:
              path: /projdata
        - name: data
          hostPath:
              path: /home/minnich2/data
        - name: code
          hostPath:
              path: /home/minnich2/src
        - name: home
          hostPath:
              path: /home/minnich2
        - name: dotlocal
          hostPath:
              path: /home/minnich2/.local-atom-app-dc2tf15

    containers:
        - name: atom-app-dc2tf15
          image: atom-registry.llnl.gov/atom/atom-app:dc2.0_tf1.5
          imagePullPolicy: Always
          resources:
              limits:
                  nvidia.com/gpu: 1  # request 1 GPU
          volumeMounts:
              - name: datastore
                mountPath: /ds/data
              - name: projdata
                mountPath: /ds/projdata
              - name: data
                mountPath: /usr/local/data
              - name: code
                mountPath: /usr/local/code
              - name: home
                mountPath: /usr/local/home
              - name: dotlocal
                mountPath: /home/minnich2/.local
          securityContext:
              privileged: true

          command: [/usr/local/bin/entrypoint.sh, /usr/local/bi

          env:
              - name: LOCAL_USERNAME
                value: minnich2
              - name: LOCAL_USER_ID
                value: "26575"
              - name: KERAS_BACKEND
                value: tensorflow
              - name: BAASIC_BROKER_URL
                value: amqp://localhost:5672
              - name: BAASIC_APP_CMD_QUEUE
                value: baasic-apps
              - name: USER_NOTEBOOK_DIR
                value: /usr/local/code

          ports:
              - name: p-jupyter
                containerPort: 8888
              - name: p-jupyter2
                containerPort: 8889
              - name: p-jupyterhub
                containerPort: 8000
```

Requirements and Tools

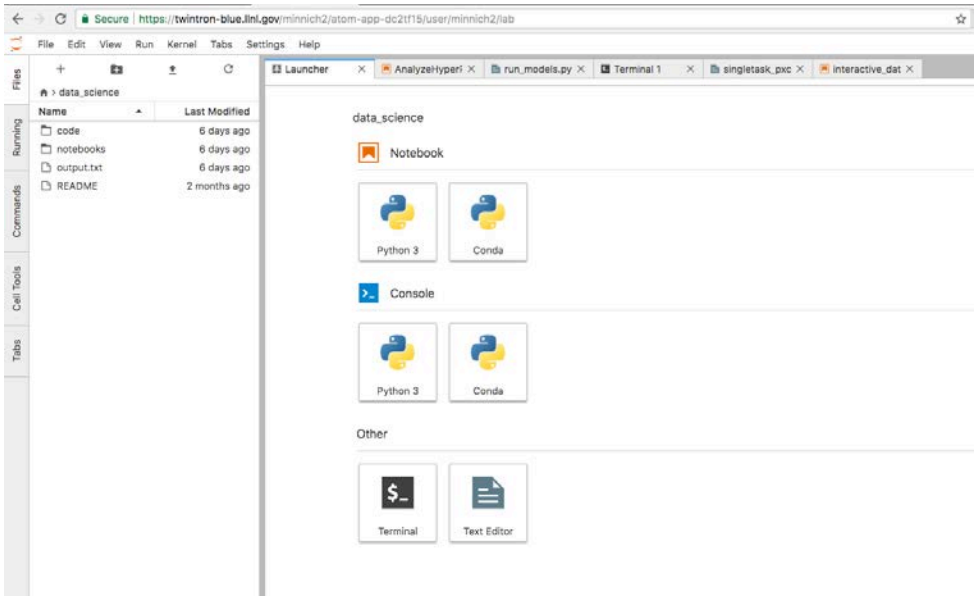
Requirements	Tools
✓ Environment consistency	Containers
✓ Easy software updates	Containers
✓ Orchestration of resources (including GPUs)	Kubernetes
✓ Role-based access to mounted volumes	Kubernetes
Intuitive GUI	**JupyterLab**
Data services (SQL, noSQL, metadata annotation)	Containers + RESTful API

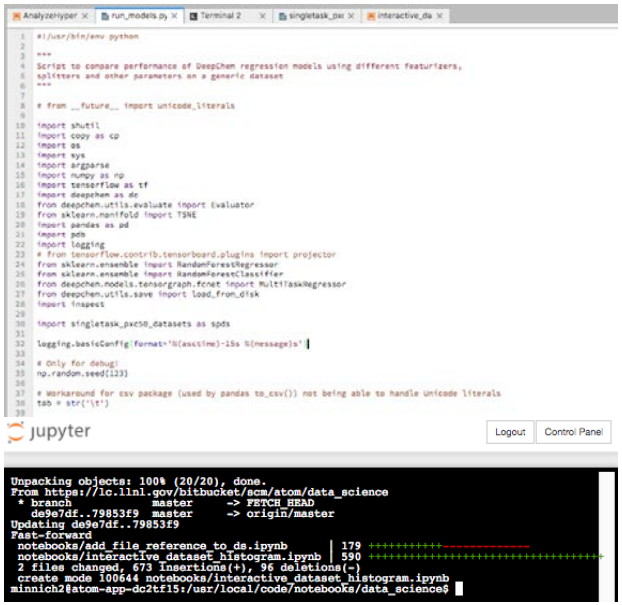
Intuitive GUI

- Domain experts need to be able to view raw data files, generate visualizations, and compute basic statistics
- Need access to a variety of Python/R libraries
- Would like to be able to share code/pages with each other
- Want it to be accessible through a web page, rather than an ssh session

JupyterLab

- JupyterLab running out of a container:
  - Provides an interactive GUI via a URL
  - Has access to the notebook environment containing the desired packages
  - Has access to a terminal window to push/pull from Git
  - Has text editor window for editing code or viewing files





Requirements and Tools

Requirements	Tools
✓ Environment consistency	Containers
✓ Easy software updates	Containers
✓ Orchestration of resources (including GPUs)	Kubernetes
✓ Role-based access to mounted volumes	Kubernetes
✓ Intuitive GUI	JupyterLab
Data services (SQL, noSQL, metadata annotation)	**Containers + RESTful API**

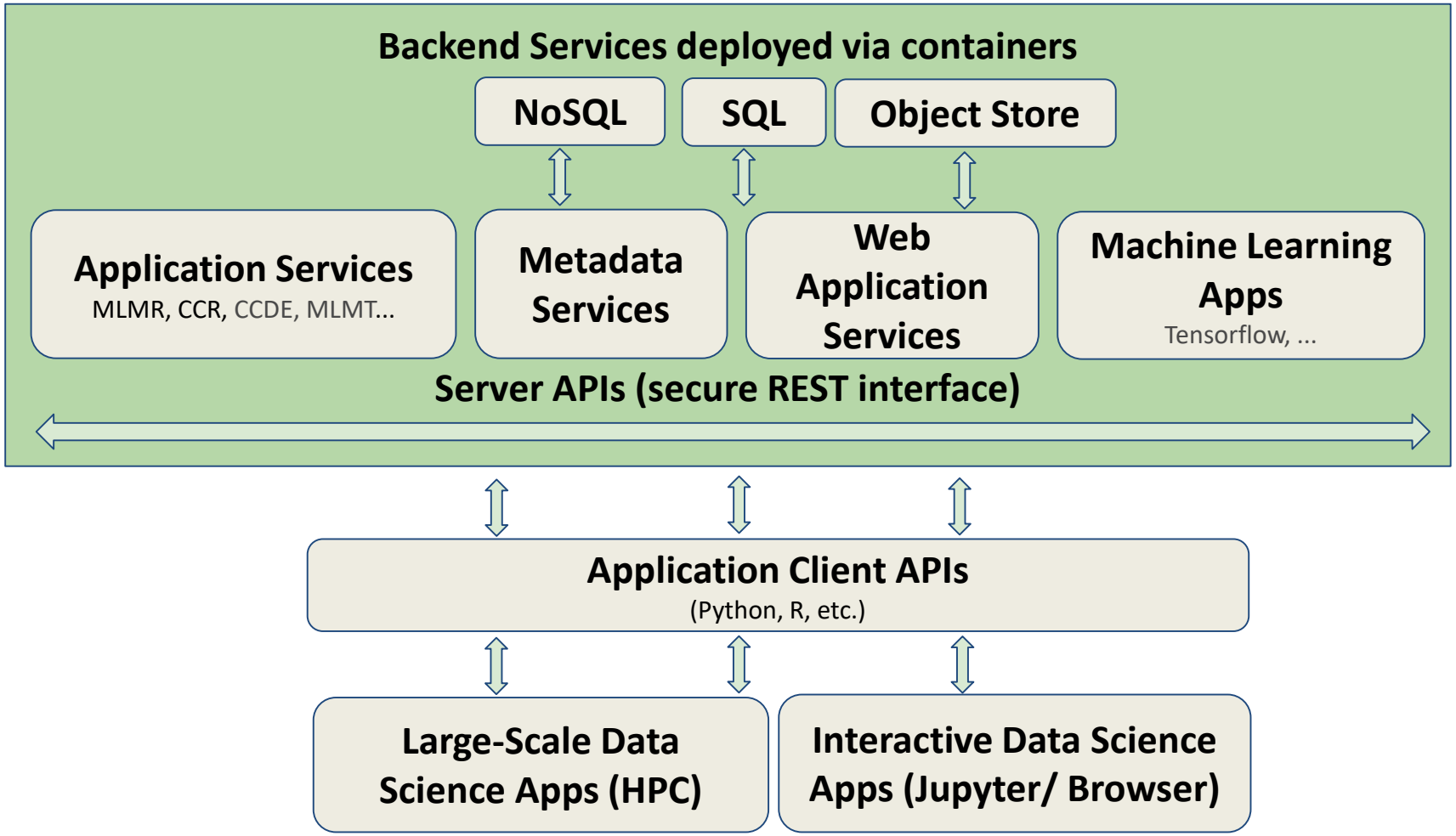
Data Services

- Data services are required to organize both raw data and metadata annotations
- Need ability to save model-ready train/test datasets, serialized models, and simulation output
- Have data services that use SQL and noSQL databases as back-ends
- Accessible via a client that uses a RESTful interface
- Allow the end-user to write database-agnostic code

ATOM's data services

- In-house datastore service
  - Allows for association of complex metadata with any type of file
- mongoDB
  - Used as backend for datastore
  - Initially helpful to have more unstructured database option
- MySQL
  - Much of the GSK data is in CSV files, which fits well with relational framework
  - Probably will expand to other relational dbs

Overall structure of data services



Requirements and Tools

Requirements	Tools
✓ Environment consistency	Containers
✓ Easy software updates	Containers
✓ Orchestration of resources (including GPUs)	Kubernetes
✓ Role-based access to mounted volumes	Kubernetes
✓ Intuitive GUI	JupyterLab
✓ Data services (SQL, noSQL, metadata annotation)	Containers + RESTful API

Requirements for success

- DevOps support
  - Team members can train up, but there will be problems that require deep knowledge and more advanced skills
- Sys admin support
  - Getting this setup to play nice with a server comes with its own set of challenges
- Team members willing to spend time acting as tech support
  - Need people to help troubleshoot issues or no one will be willing to adopt this workflow

Challenges

- Had to learn Docker and Kubernetes
- We were facing a learning curve ourselves while trying to assist others
- Did not have dedicated DevOps/sys admin help for the first 6 months
- Supporting users with different levels of IT skills
- Other projects share (and occasionally break) the server
- Have to handle LLNL security concerns (certs are a pain)

Successes

- Have the entire ATOM Data-Driven Modeling Team working out of containers
- Can get new team members up and running in a day
- Have generated ~1k deep learning and random forest models
- Have ~20 notebooks being shared with data intake, processing, curation, and visualization capabilities
- Domain experts have begun interacting with data through these notebooks
- Computational biologist built a “Data Explorer” webpage
- Have ~10 domain experts putting data in datastore with curated metadata tags

Future plans

- Install kubespawner for JupyterHub so team members can launch containers without needing to run kubectl
- Model zoo: formalized pipeline for reading in model-ready datasets and outputting serialized models with hyperparameters used and performance metrics calculated
- Active learning loop that incorporates the datastore
- Eventually would like to have data science cluster that is separate from the HPC systems, but can sit inside same network

Thanks for your time!
Questions?

Lawrence Livermore
National Laboratory