Scalable ML Deploys for Data Science

Jelai Wang

Dec 16, 2020

Problem Context

Machine Learning (ML) model building often happens interactively in Jupyter Notebook or R Studio.

Think about “0 to 1” idea.

This is the work of research & development.

Note non-ML research method development can fit here, too.

See “Building and running ML models for data scientists” [1] for further background.

Problem Statement

Now that you have completed ***development*** and have a model (or method) to share with a broader target audience, how do you deploy to ***production*** in a scalable way?

Think “1 to 100” and “100 to millions”.

This is the work of reduction to practice.

Focus shifts to software engineering and devops concerns.

A Common Design Pattern

A Common Design Pattern

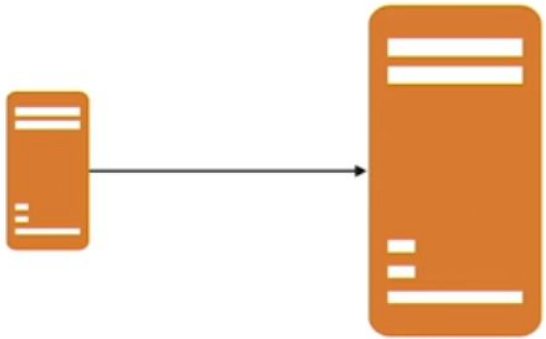
Microservices

+ Service (REST API)
+ Containers (Docker)
+ Orchestration (Kubernetes)

This pattern takes advantage of *horizontal* scaling for the cloud.

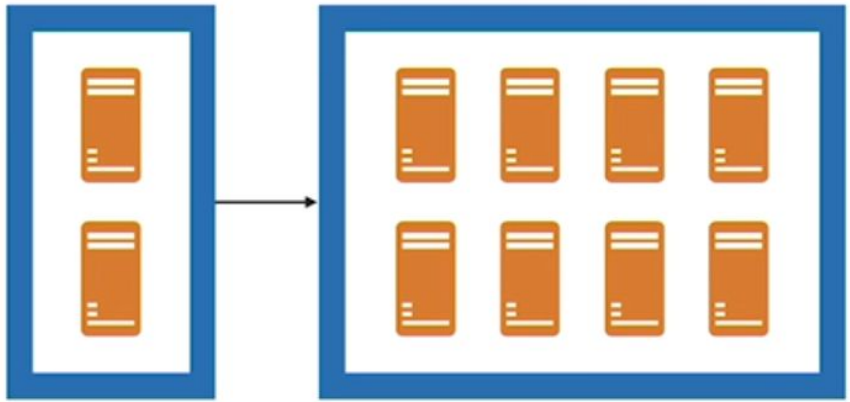
Vertical Scaling

- Replace servers
- Increase number of CPUs
- More memory
- More storage

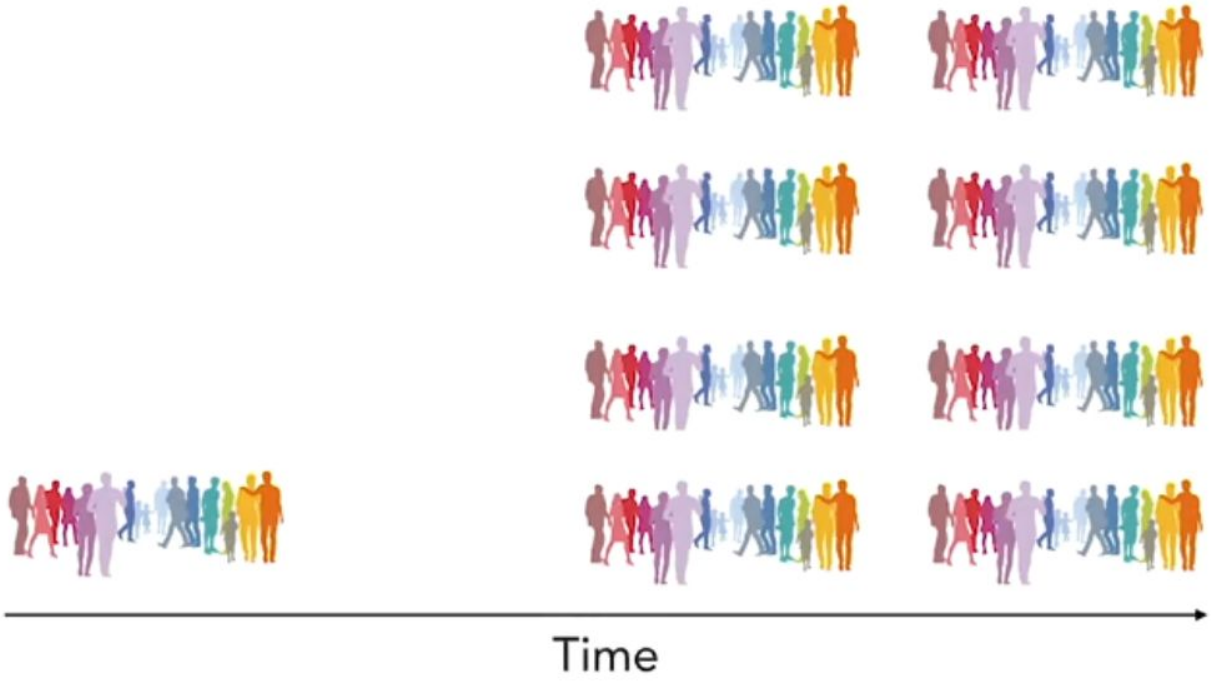


Horizontal Scaling

- Add servers
- Running same software



Increasing Demand



Backlog of Work

Key Concepts

Microservices

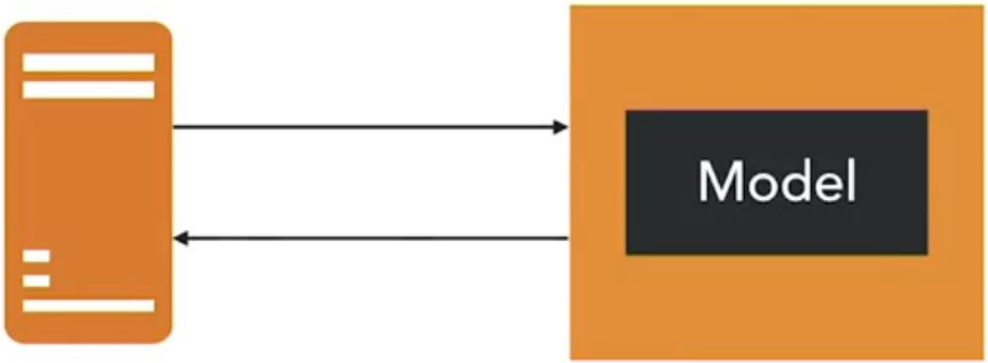
- Deploy the model with an API
- Run the model and API in a container
- Manage the containers using an orchestration service

Key Concept, *Service*

Services

- Accessible over network
- Invoked using standard protocols
- Perform a single function
- Deployed in parallel

Models as Services



Iris Classifier

https://scalablemodels.com/**api**/**v01**/classify

Features

- Petal length
- Petal width
- Sepal length
- Sepal width

API Call with Parameter

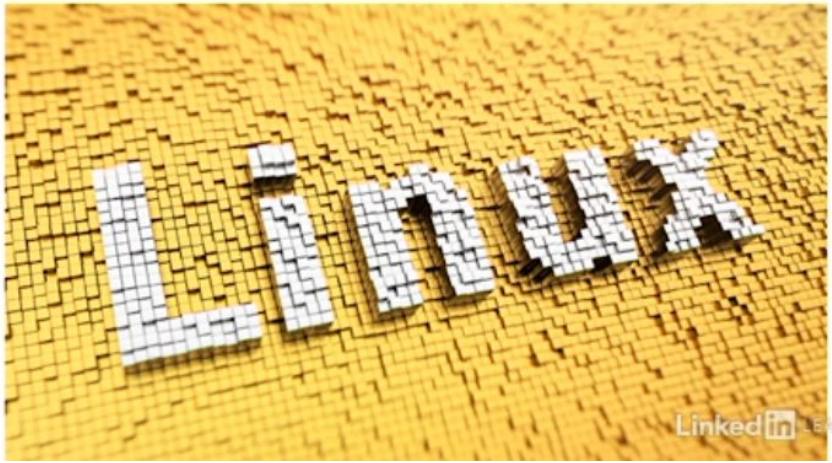
https://scalablemodels.com/api/v01/classify?sl=5.1&sw=3.5&pl=1.4&pw=0.3

```
{ "species" : "Iris setosa" }
```

Key Concept, *Containers*

Deploying to Multiple Servers

- From source code
- Need libraries and packages
- Watch for differences in operating systems

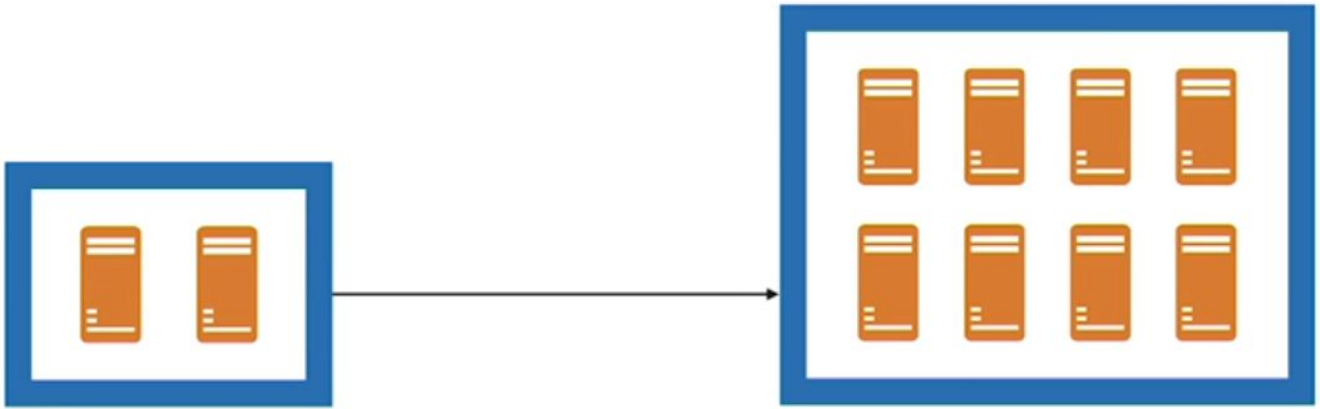


Key Concept, *Orchestration*

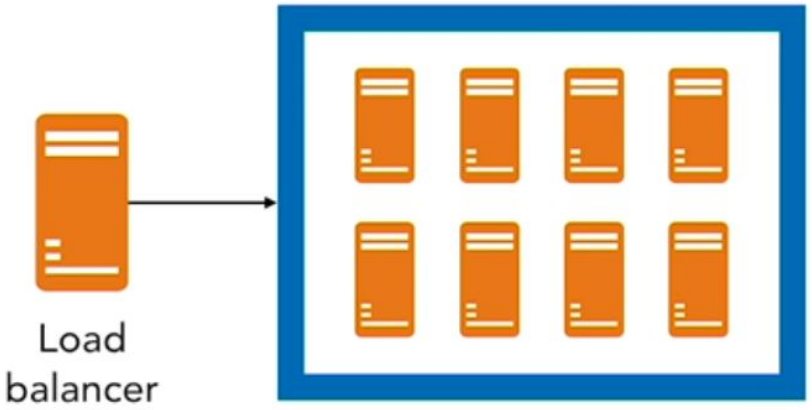
Orchestration with Kubernetes

- Horizontal scaling
- Optimized placement of containers
- Automated rollouts of new versions of code
- Storage management
- Self-healing

Adding Resources

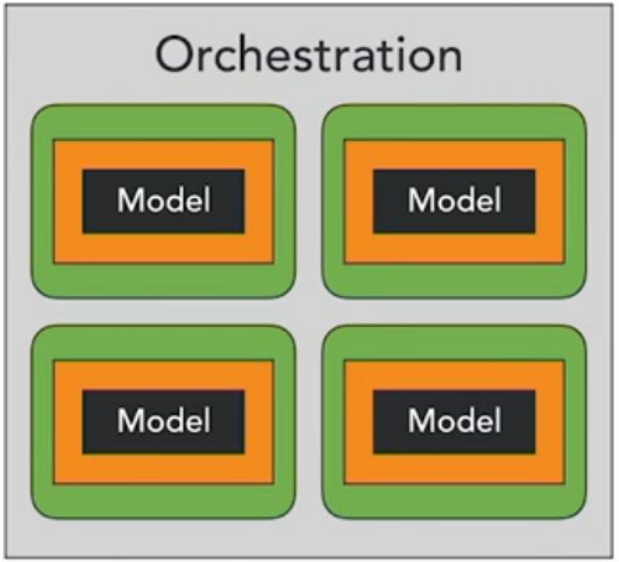


Scalable Cluster



Three Levels to Scalability

- Exposing the model through an API
- Putting the model and API in a container
- Managing containers with an orchestration platform



Why do we care?

Can we help you get from “1 to 100”?

Proof-of-concept

Proof-of-concept, *The Method*

Need an algorithm that is easy to implement, but representative of ML models and methods developed by our target audience.

Calculate pi to a million digits after the decimal point. TY to Jake for this suggestion.

Implement the picrunch algorithm by Don Cross. See https://medium.com/@cosinekitty/how-to-calculate-a-million-digits-of-pi-d62ce3db8f58 for further detail.

Proof-of-concept, *The Service*

Adapt *picrunch* to REST API.

```
GET /api/v0/pi?ndigits=99
```


```
{"ndigits":99,"pi":"3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706 7"}
```


See REST API documentation in README at https://github.com/jelaiw/scalable-ml-deploy#picrunch-algorithm-as-a-service-rest-api.

Proof-of-concept, *The Container*

Create Dockerfile (recipe for building a Docker image) for *picrunch* service.

Build Docker image and deploy to a Docker registry.

`docker://jelaiw/picrunch:0.2`

See Docker Hub at https://hub.docker.com/repository/docker/jelaiw/picrunch.

Proof-of-concept, *The Orchestration*

Provision OpenStack VM, see infrastructure code at https://gitlab.rc.uab.edu/gbm-pdx/gbm-openstack-infrastructure.

Install minikube (1-node cluster).

Write a simple kubernetes deployment for *picrunch*.

See https://github.com/jelaiw/scalable-ml-deploy/blob/2b9bdfc5daf2189677a8f51228dce398e1220dfe/picrunch.yaml.

Proof-of-concept, *The Control Experiment*

Need a sort of control experiment so we can compare results and make a better decision regarding whether to move forward from PoC to pilot.

Deploy *picrunch* service to http://dev.ubrite.org/api/v0/pi.

In other words: service, but no container or orchestration.

Next Step, *Share Findings From PoC*

Proof-of-concept, *Findings from End User POV*

Can research teams using U-BRITE handle deploying their methods as Docker services?

- Probably yes, but may be a non-trivial lift if there is no real dev.
- See our RA2 DREAM Challenge experience as an example of success, especially the amount of handholding that Robert had to perform.

How can U-BRITE help these teams?

- Expand training offerings, add examples/templates.
- Embed a developer with teams.
    - Short term, just to help get from method to Docker service.

Proof-of-concept, *Findings from Infrastructure POV*

Do we want to set up and maintain a Kubernetes cluster for U-BRITE users?

- This assumes U-BRITE users will handle the Service and Docker parts of this design pattern.
- Maintaining a Kubernetes control plane will require additional, dedicated resources.
  - This assumes the OpenStack infrastructure will continue to be maintained by Research Computing.
  - However, the OpenStack and Kubernetes configuration will need to be maintained by someone with an ops role in the U-BRITE team.
  - This is non-trivial and typically a full-time position in industry settings.
- Consider the same kind of pivot we made with Binder during the pilot.

Proof-of-concept, *Possible Path from PoC to Pilot*

Do we have a real use case?

- Which team wants to scale in this way right now, but is prevented from doing so due to lack of infrastructure/expertise?
- Who has the need?

What infrastructure need to be built?

- Production OpenStack is not yet available (3Q or 4Q 2021? Check with JPR).
- Consider pivot to a managed cloud Kubernetes for pilot?

References

1. “Deploying Scalable Machine Learning for Data Science”, Dan Sullivan, LinkedIn Learning, https://www.linkedin.com/learning/deploying-scalable-machine-learning-for-data-science/scaling-ml-models, Aug 2018.

2. “Machine Learning: How to Build Scalable Machine Learning Models”, Satwik Kansal, https://www.codementor.io/blog/scalable-ml-models-6rvtbf8dsd#deploying-and-real-world-machine-learning, Feb 2020.

3. “Scaling Machine Learning from 0 to millions of users”, Julien Simon, https://medium.com/faun/scaling-machine-learning-from-0-to-millions-of-users-part-2-80b0d1d7fc61, Feb 2019.

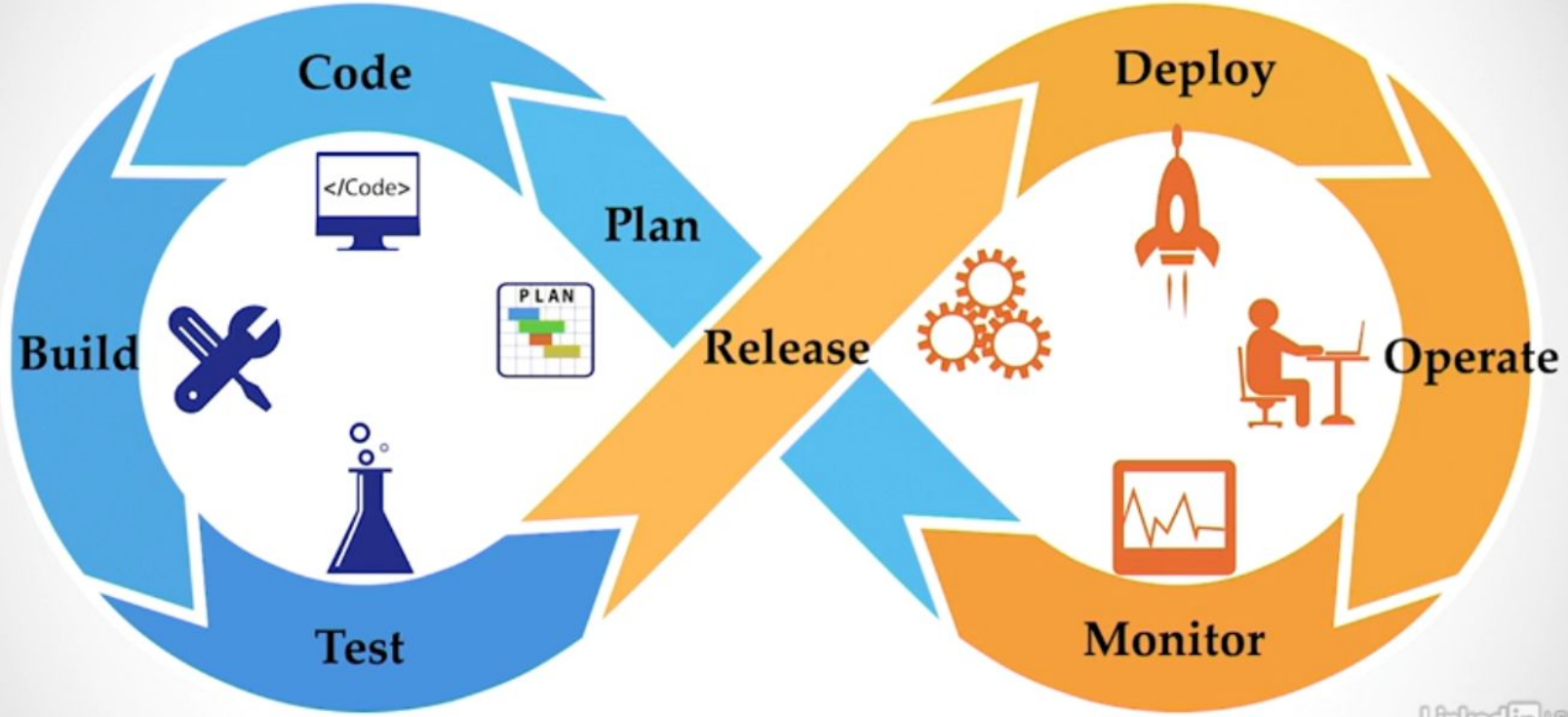
4. “Scaling Machine Learning at Uber with Michelangelo”, Jeremy Hermann, Mike Del Balso, https://eng.uber.com/scaling-michelangelo/, Nov 2018.

5. “Containers: Real Adoption and Use Cases In 2017”, Forrester Consulting, https://www.dell.com/learn/us/en/555/business~solutions~whitepapers~en/documents~containers_real_adoption_2017_dell_emc_forrester_paper.pdf, Mar 2017.

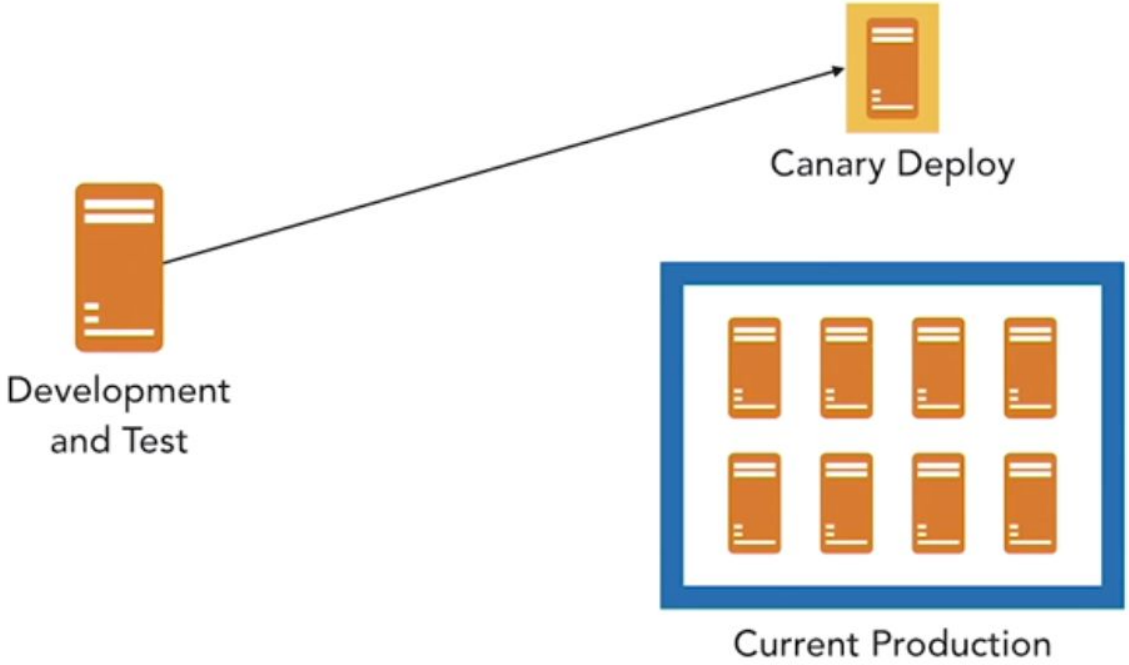
6. “Learning Kubernetes”, Karthik Gaekwad, https://www.linkedin.com/learning/learning-kubernetes/, Jun 2020.

Extras

Update Models



Canary Deploys



Blue/Green Deploys