Deploying Scalable ML for Data Science

Anthony Mipawa

Introduction

- ML Models building process
  - Deploying codes to servers.
  - Ensuring continuous availability.
  - Programmatic access(APIs)
  - Access controls
  - Scalability
- Requirements for ensuring that Applications are Scalable.
- Three Layers of Scalable ML stack
- Orchestration tools for managing clusters of servers
- Containers monitoring tools

All is to ensure best practice of deployment of ML models.

My assumptions to audience

- Familiarity with Machine Learning.
- Comfortable reading programming code.
- Understand operating system principles such as processes , CPU utilization , disk utilization and I/O etc

The Need to Scale ML Models

1.0 Building & Running ML for Data Scientist

Uses of Machine Learning

- Pricing
- Fraud detection
- Document Classification

Data Scientist we have to ask:

- What problem we’re trying to solve?
- What are business requirements need to consider?

The Need to Scale ML Models

1.1 Building & Running ML for Data Scientist

Defining the Problem

Be Precise	Too Vague
Describe the problem that can be modeled	General description of what we want
State quantifiable problem such as Reducing number of customers leaving for a competitor	A broad statement such as Improve customer satisfaction

The Need to Scale ML Models

1.2 Building & Running ML for Data Scientist

Building Models

Regression	Classification
Make numeric value predictions	Make choices among categorical variables
• Estimated values of a stock • Best price to charge for a hotel room • Number of new customers new sales pricing can attract	• Legitimate credit card charge vs Fraudulent charge • Identifying an object in an image • Customer likely to respond to an advertisement

The Need to Scale ML Models

1.3 Building & Running ML for Data Scientist

Data collection

- Time consuming
- What is the data describing?
- Are there quality control problems?
- Is there missing data?
- Inconsistent conversions

The Need to Scale ML Models

1.4 Building & Running ML for Data Scientist

Best practice

- Data distribution.
  - Explore data - understand kind of data you're working with.
- Iterative Model building

The Need to Scale ML Models

2.0 Building & Deploying ML Models for Production use:

What to consider

- Deploying codes to servers.
- Ensuring continuous availability.
- Programmatic access(APIs)
- Access controls
- Scalability

Persisting Models:

- Language specific
- Predictive Model Markup languages like html , xml

The Need to Scale ML Models

2.1 Building & Deploying ML Models for Production use:

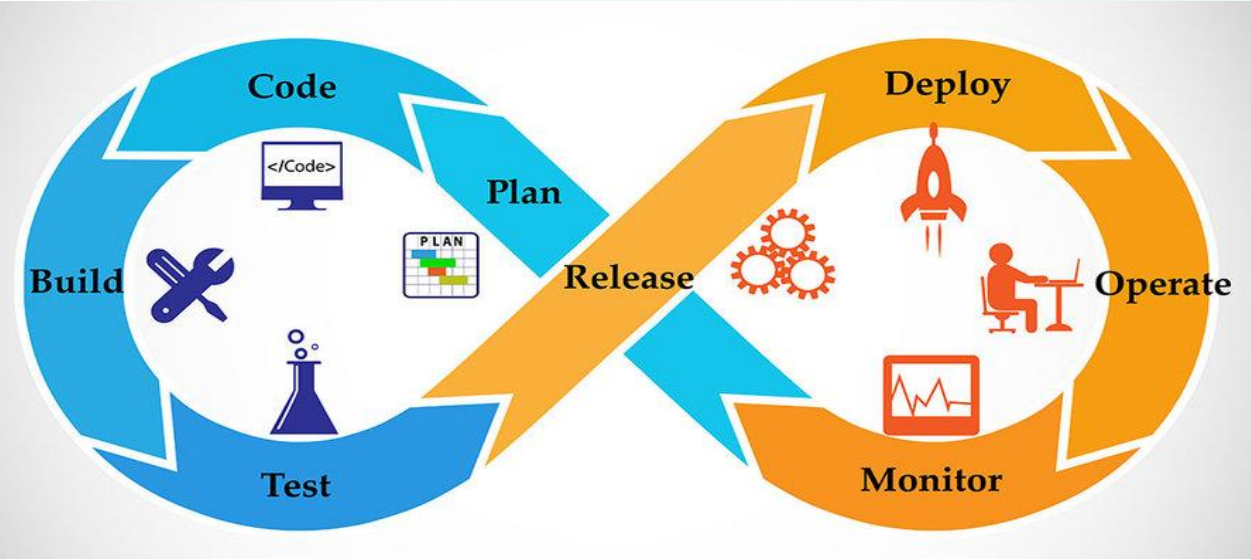
Models are Software

- Version control.
- Testing.
- Security after deployment

The Need to Scale ML Models

2.2 Building & Deploying ML Models for Production use:

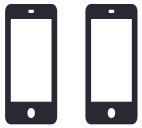
Update Models



The Need to Scale ML Models

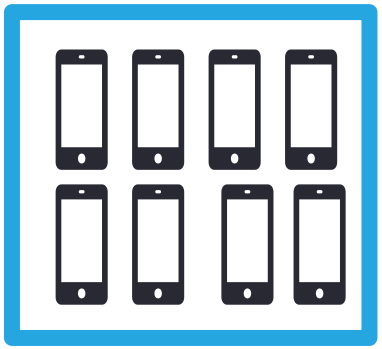
2.3 Building & Deploying ML Models for Production use:

Canary Deploys



canary deploy

Development & Test



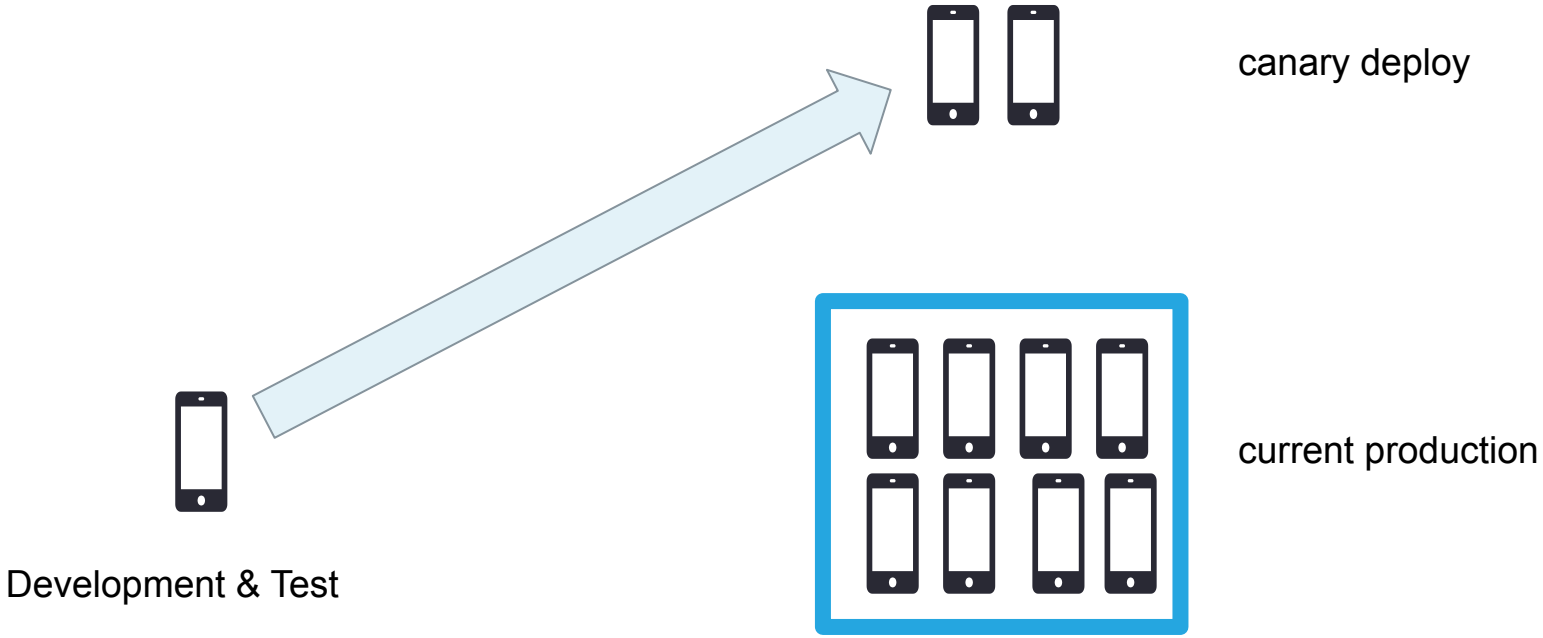
current production

pycon tanzania

The Need to Scale ML Models

2.4 Building & Deploying ML Models for Production use:

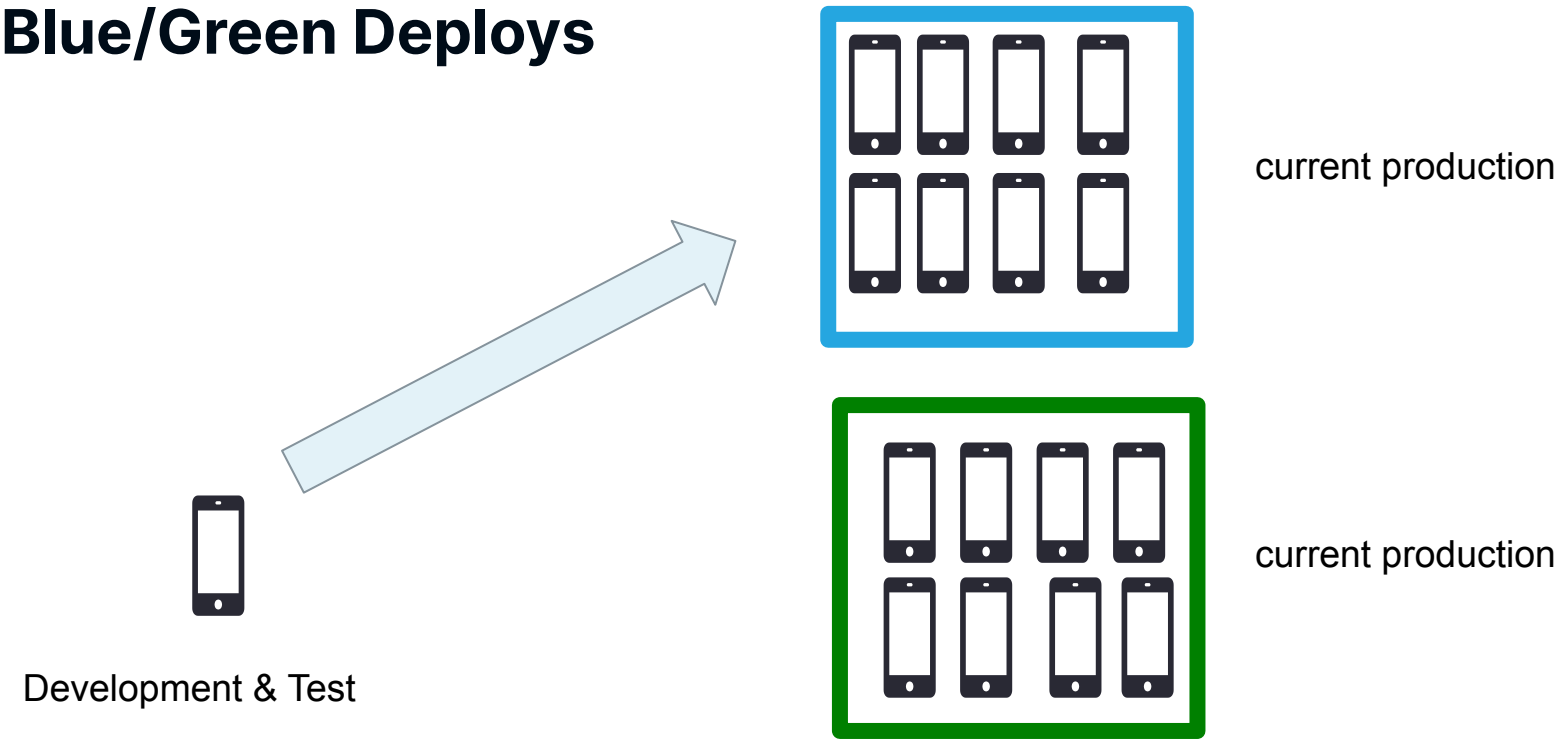
Canary Deploys



The Need to Scale ML Models

2.5 Building & Deploying ML Models for Production use:

Blue/Green Deploys



pycon tanzania

The Need to Scale ML Models

3.0 Definition of Scaling ML for Production

Meaning of Scalable ML:

- Keep up depending on the demand variations.
- Produce responses in a timely manner

How to know if your system is scalable?

- Scalable System can use additional resources when load increases
- Scalable System can release resources when are not in use.
- Load balancer
- Autoscaling Constraints

Building a Scalable systems is a technical challenge

The Need to Scale ML Models

3.1 Definition of Scaling ML for Production

Redundancy Compensates for Failure:

- Such as Servers crush, disk failure,Rack loses power etc. non-preventable failure

Common way of dealing with these frequent failures:-

- To use Redundant components
  - Cluster of servers
  - If server from one cluster fails by auto-scaling another server will be added to ensure availability of service.

The Need to Scale ML Models

4.0 Overview of the tools and Techniques for Scalable ML

Microservices:

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

Deploying to Multiple servers:-

- From source codes
- Need libraries and Package
- Watch for differences in operating systems

The Need to Scale ML Models

4.1 Overview of the tools and Techniques for Scalable ML

Containerization:

- Solve Operating systems potential installations problems.
- Example of most used containerization service is **Docker**

Orchestrations:

- Platform for managing containers
- Example of most used containerization service is **Kubernetes**
- Management tasks by kubernetes:

  Horizontal Scaling,Optimized placement of containers, Automated rollouts of new versions of code, Storage management , self healing.

The Need to Scale ML Models

4.2 Overview of the tools and Techniques for Scalable ML

Three Levels to Scalability :

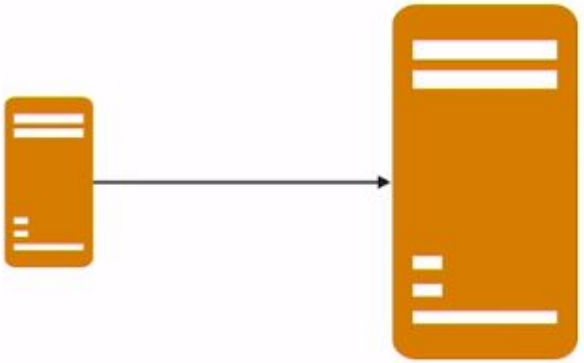
- Exposing the ML model through an APIs.
- Putting the model and APIs in a container.
- Managing the container with an orchestration platforms.

Design Patterns for Scalable ML Applications

1.0 Horizontal vs Vertical Scaling

Vertical Scaling:

- Replace a single server with larger servers(replace servers).
- Increase number of CPUs
- More Memory
- More storage

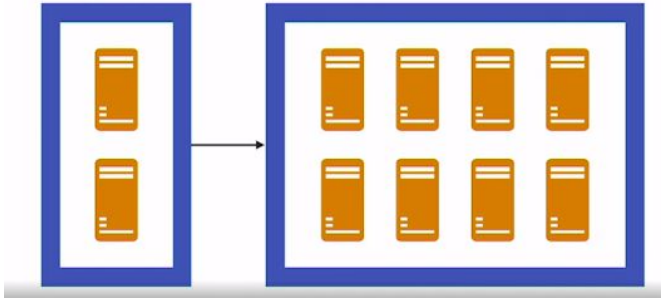


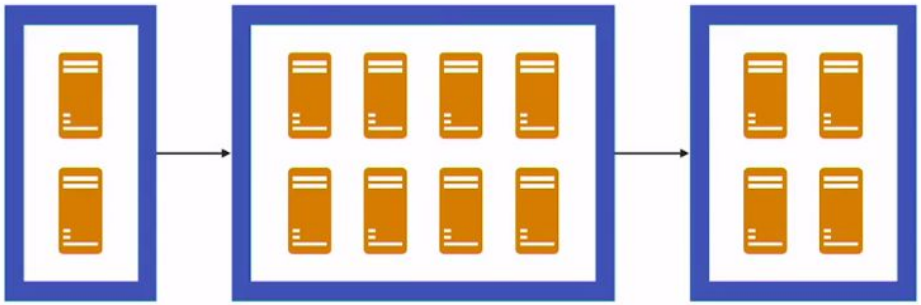
Design Patterns for Scalable ML Applications

1.1 Horizontal vs Vertical Scaling

Horizontal Scaling:

- Adding servers to a cluster or distributed servers.
- Removing servers to a cluster 0r distributed servers.
- Running same software.





Design Patterns for Scalable ML Applications

1.2 Horizontal vs Vertical Scaling

Peak Demand:

- With vertical scaling,need to plan ahead for peak demand.
- With horizontal scaling, adapt to peak demand
- Cloud services are designed to accommodate varying demand.
  - AWS
  - Azure
  - Google Compute Cloud

Design Patterns for Scalable ML Applications

2.0 Running Models as Services

What is services mean?

- Software abstraction for describing executing programs that share common characteristics.
- Accessible over network
- Invoked using standard protocols
- Perform a single function
- Can be deployed in parallel

Design Patterns for Scalable ML Applications

2.1 Running Models as Services

How to distinguish services?

- Way of data exchange
  - Example restful services - often pass data in JSON structure
  - High-volume services sometimes use other protocols such as thrifts or protocol buffers

Design Patterns for Scalable ML Applications

3.0 APIs for ML Model services

APIs best practice:

- Including term api in url

https://scalablemodels.com/api/classify

- Including version due to variation of updates

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

- Call APIs with parameters

https://scalablemodels.com/api/v01/classify?sl=5.1&sw=3.5&pl=1.5&pw=0.3

Once you invoke: {“species”:”Iris setosa”}

Design Patterns for Scalable ML Applications

3.0 APIs for ML Model services

APIs best practice:

- API key: unique string that identifies a user

https://scalablemodels.com/api/v01/classify?sl=5.1&sw=3.5&pl=1.5&pw=0.3&key=A1765342aH48ed

  - Key may be useful in access control and services payment

Design Patterns for Scalable ML Applications

4.0 Load balancing & Clusters of servers

Load Balancer Features:

- Distribute workload.
- Detect unhealthy servers and stop sending requests.
- Are configured for high availability
- Able to meet demand

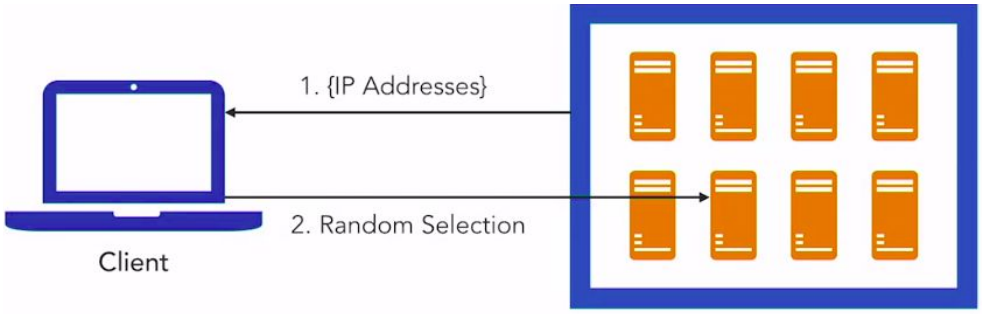
Load Balancing Methods:

- Client-side balancing
- Random choice
- Round-robin
- Least connections
- Least Load

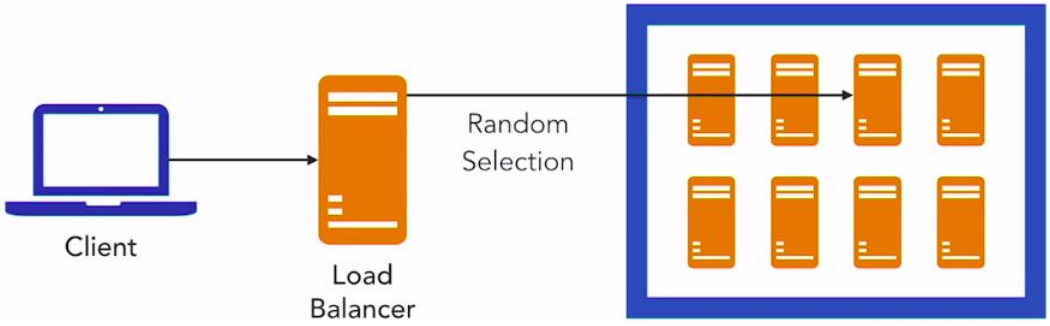
Design Patterns for Scalable ML Applications

4.1 Load balancing & Clusters of servers

Client-side balancing:



Random choice balancing:

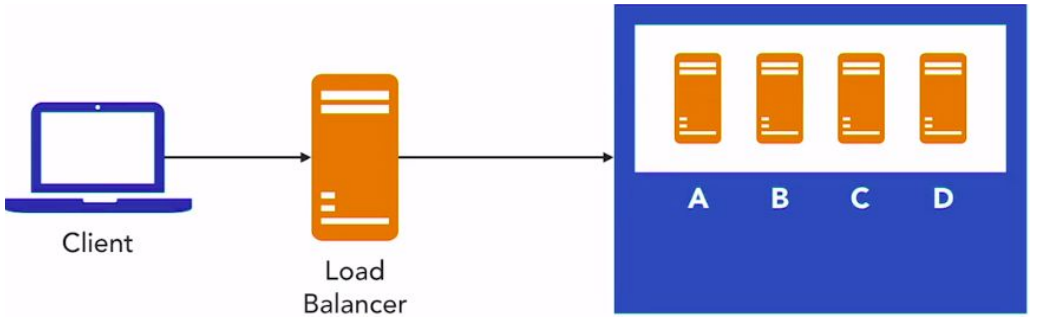


pycon tanzania

Design Patterns for Scalable ML Applications

4.2 Load balancing & Clusters of servers

Round-Robin Balancing:



Load Balancer Additional Features:

- Application health check
- HTTP compression
- Firewall
- Intrusion detection and prevention

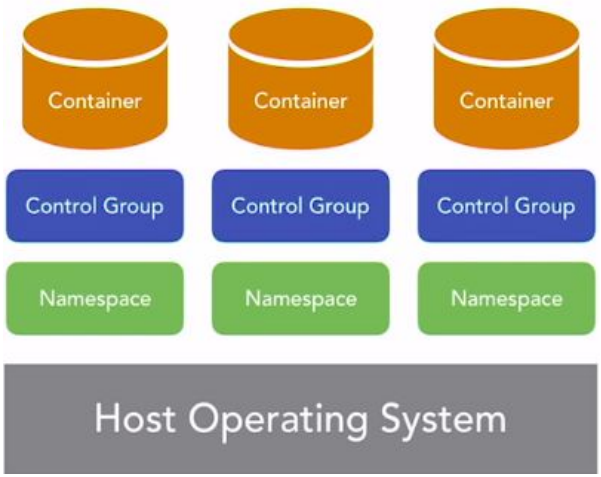
pycon tanzania

Design Patterns for Scalable ML Applications

4.3 Load balancing & Clusters of servers

Container images:

- Base Operating System
- Programming Languages
- Machine Learning code
- Packages and Libraries



Design Patterns for Scalable ML Applications

4.4 Load balancing & Clusters of servers

How to Create Container images?

- Specify container contents
- Execute build operation to create the image
- Store the image in a repository
- Download image to servers

Deploying ML Models as Services

1.0 Service encapsulate ML models

Documentation:

- The HTTP URL
- List of parameters and types
- Request body if JSON structure is used

Swagger is the popular tool for automating and generating API documentation. For more info visit https://swagger.io

Deploying ML Models as Services

2.0 Using Flask to Create APIs

Example of python Classifier:

```
from flask import Flask

app = Flask(__name__)

@app.route("/classify",methods=['GET', 'POST')
def classify():
    pl = request.args.get("pl")
    pw = request.args.get("pw")
    sl = request.args.get("sl")
    sw = request.args.get("sw")
    results = xgboost_classify_iris(pl, pw, sl, sw)
    return results
```

Deploying ML Models as Services

2.1 Using Flask to Create APIs

Running python program with Flask:

```
$ export FLASK_APP=classify.py
$ python -m flask run --host=0.0.0.0 –port=8080
```

Calling python service with Flask from client:

```
$ curl --data “sl=5.1&sw=3.5&pl=1.4&pw=0.3” \
    “http://scalablemodel:8080/classify”
```

Results:

```
{ “species” : “Iris setosa” }
```

pycon tanzania

Deploying ML Models as Services

3.0 Best Practices for API Design for ML Models

RESTful Best Practices:

- Use a RESTful interface
- Use the GET or POST commands.
- Document your APIs

General APIs Best Practices:

- Name the endpoints with informative names ,like classify
- Use API tools, like Plumber and Flask
- Include the term API in the path of you URLs
- Include the version number in the API
- Use API keys to control access of your services

Running ML Services in Containers

1.0 Why Containers?

- Install and test once
- Not each time we start a new server
- Each deploy runs the same set of components
- Minimal delay to servicing requests

Docker is the most widely used containerization platform.

Running ML Services in Containers

Docker Components

- Docker engine
- Docker client
- Docker Registry

Getting a Docker Image:

- Download from Registry
- Build from Dockerfile

Running ML Services in Containers

Dockerfile

```
FROM ubuntu

#Install libraries & packages
RUN apt-get update && apt-get install -y/
python-pip \
python-dev \
python-lumpy \
Python-scipy \
pip

RUN pip install scikit-learn flask-restful

# Copy model code from current directory
COPY ./model

#expose the port for the API
EXPOSE 8100

#run the API
CMD [“python’]
```

Scaling ML Models with Kubernetes

1.0 Management Tasks

- Starting and Stopping Containers
- Determining where to run containers
- Checking the health of containers
- Restarting and Replacing unhealthy containers

Scaling ML Models with Kubernetes

2.0 Orchestration Platforms

- Kubernetes
- Docker Swarm
- Mesos and Marathon

Scaling ML Models with Kubernetes

3.0 Introduction to Kubernetes

- Open source platform
- Automating deployment
- Scaling
- Management of containerized applications

Scaling ML Models with Kubernetes

4.0 Creating a Kubernetes cluster

- Usually a kubernetes cluster is run by an infrastructure team.
- Kubernetes components
  - Minikube
  - A virtual Machine
  - Kubectl: a command line tool

Scaling ML Models with Kubernetes

5.0 Creating a Kubernetes cluster

- Run created kubernetes cluster
- Deploy containers in a Kubernetes cluster
- Scaling up a Kubernetes cluster
- Autoscaling a Kubernetes cluster

ML Services in Production

1.0 Monitoring service performance

Monitoring targets:

- Baseline performance
- Unusual events

Three Monitoring Domains

- Services
- Containers
- Clusters

Conclusion

Best Practices for scaling ML

- Horizontal Scaling
- Deploy model with APIs
- Use containers
- Use orchestration tools
- Monitor all levels of the stack

pycon tanzania

'Kind Sponsors & Supporters.'

Without whom the Python Conference for Software Developers in Tanzania would not be possible!

python SOFTWARE FOUNDATION

TISPA
TANZANIA INTERNET SERVICE PROVIDER ASSOCIATION

flashnet

maishabroadband
unlimited. high speed. reliable

HABARINODE
Better Solutions • Lasting Connections

beem

Organizers & Supporters

tanzania irabu IO INSTITUTE INSTITUTE OF ACCOUNTANCY ARUSHA

@pycontanzania