Using Docker, AWS Fargate and ECR.

Deploy and Scale ML Models on AWS

Delivered by: Noureddin Sadawi, PhD

About the Speaker

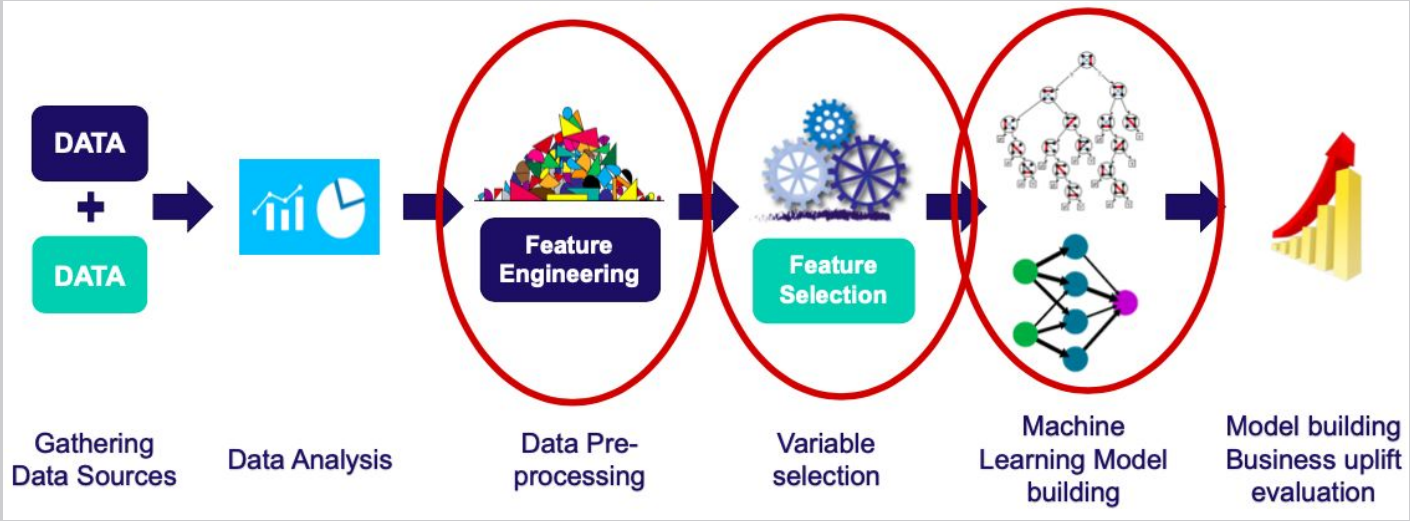
Name: Dr Noureddin Sadawi

Academic Qualification: PhD Computer Science

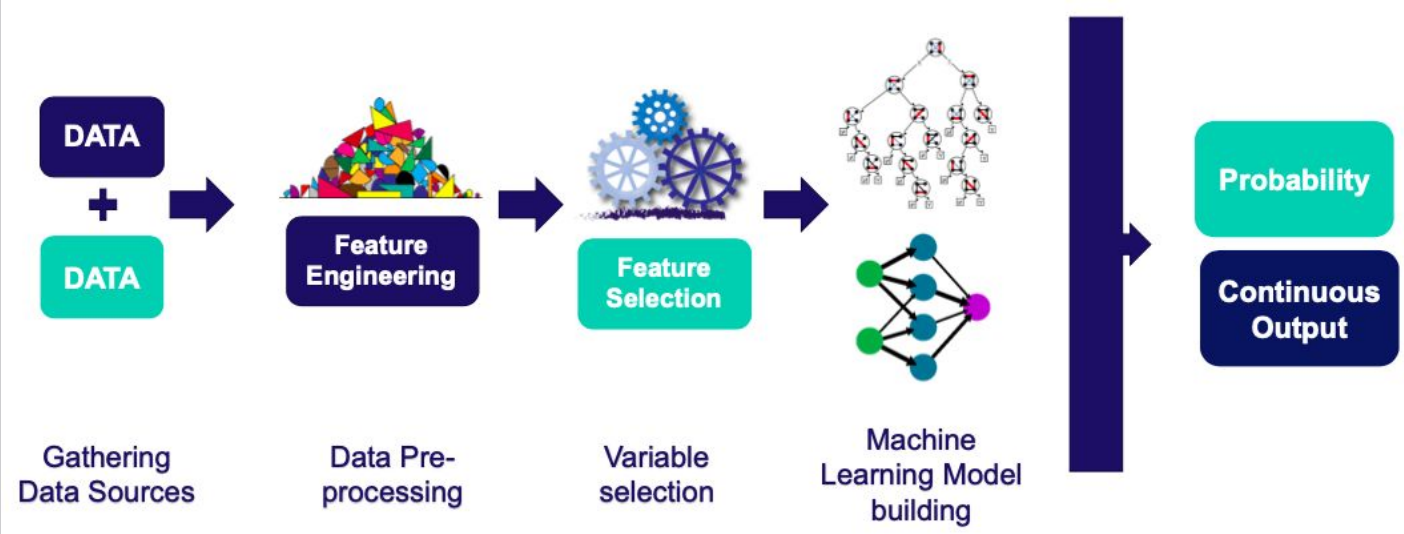
Experience: Several areas including but not limited to Docker, Machine Learning and Data Science, Python and more.

AWS Certified: Machine Learning Specialty

Machine Learning Pipeline: Production



Machine Learning Pipeline: Production



Feature Engineering

- Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used in supervised learning.
- In order to make machine learning work well on new tasks, it might be necessary to design and train better features.
- A machine learning technique that leverages data to create new variables that aren’t in the training set.

Feature Engineering

- It can produce new features for both supervised and unsupervised learning, with the goal of **simplifying and speeding up data transformations** while also **enhancing model accuracy**.
- Feature engineering is required when working with machine learning models.
- Regardless of the data or architecture, a terrible feature will have a direct impact on your model.

Feature Selection

- The process of reducing the number of input variables when developing a predictive model.
- It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.
- Select a subset of original features, not replace them with new ones (compare with dimensionality reduction).

Why Feature Selection?

- Simple models are easier to interpret.
- Shorter training times.
- Enhanced generalisation by reducing overfitting.
- Easier to implement by SW developers -> Model production (i.e. less code to process and handle features).
- Reduced risk of data errors during model use.
- Data redundancy (many features provide similar info).

Feature Preprocessing

- The technique of making raw data into more meaningful data or the data which can be understood by the Machine Learning Model.
- Real-world **data** is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors.
- To tackle this, data preprocessing technique is introduced.
- Example techniques :
  - Vectorization (e.g. one-hot encoding).
  - Normalization (i.e. Scaling) and Standardization.
  - Handling Missing Values.

Reproducibility 1/3

- The ability to replicate a model precisely, so that given the exact same raw data as an input, the reproduced model will return the same output.
- Lack of reproducibility can have numerous negative effects.
- From a financial standpoint, if one were to invest significant resources into creating a model in a research environment, but they were unable to reproduce it in a production environment, little benefit would come of that model and its predictions.

Reproducibility 2/3

- Similarly, this would result in wasted time and effort, in addition to the financial loss.
- The machine learning model serves little use outside of the production environment.
- Most importantly, however, is that reproducibility is crucial to replicating prior results. Without this, one could never accurately determine if new models exceeded previous ones.

Reproducibility 3/3

- Slides by Rachael Tatman: https://www.rctatman.com/files/Tatman_2018_ReproducibleML.pdf
- Nice article by Sole Galli: https://trainindata.medium.com/how-to-build-and-deploy-a-reproducible-machine-learning-pipeline-20119c0ab941
- The Machine Learning Reproducibility Crisis by Pete Warden: https://petewarden.com/2018/03/19/the-machine-learning-reproducibility-crisis/
- Building a Reproducible Machine Learning Pipeline: https://arxiv.org/ftp/arxiv/papers/1810/1810.04570.pdf

Reproducibility with Keras and NNs

- How to Get Reproducible Results with Keras: https://machinelearningmastery.com/reproducible-results-neural-networks-keras/
- Reproducibility in ML: why it matters and how to achieve it: https://www.determined.ai/blog/reproducibility-in-ml
- StackOverflow: https://stackoverflow.com/questions/32419510/how-to-get-reproducible-results-in-keras

Research vs Production Environments

	Research	Production
Separate from customer facing software	✓	x
Reproducibility matters	Sometimes	Almost always
Scaling challenges	x	✓
Can be taken offline	✓	x
Infrastructure planning required	Sometimes	Almost always
Difficult to run experiments	x	✓

Sequential vs Pipeline Code Example

Python Code on Jupyter Notebooks

Titanic Survival

Production Code

- Production code is designed to be deployed to end users.
- Considerations:
  - Testability & Maintainability.
  - Scalability & Performance.
  - Reproducibility.

Python Packages

A **module** is a file which contains various Python functions and global variables. It is just a file with a .py extension which has python executable code.

A **package** is a collection of modules.

Professional Packaging

```
parent/
├── regression_model/
│   └── config/
│   └── datasets/
│   └── processing/
│   └── trained_models/
│   ├── config.yml
│   ├── pipeline.py
│   ├── predict.py
│   ├── train_pipeline.py
│   └── VERSION
├── requirements/
│   ├── requirements.txt
│   └── test_requirements.txt
├── setup.py
├── MANIFEST.in
├── pyproject.toml
├── tox.ini
└── tests/
```

- Makes your package easy to maintain, extend, test, install, deploy etc.
- This is a result of experience.
- Opinionated!

tox

tox is a generic virtualenv management and test command line tool you can use for:

- Checking that your package installs correctly with different Python versions and interpreters.
- Running your tests in each of the environments, configuring your test tool of choice.
- Acting as a frontend to Continuous Integration servers, greatly reducing boilerplate and merging CI and shell-based testing.

More here: https://tox.wiki/en/latest/

tox

tox lets you develop in multiple versions of Python.

- virtualenv per interpreter
- installs your package
- pip / easy_install requirements
- runs tests

```
$ cd myproject
$ tox -e py27,py38
```

tox Config (tox.ini file)

```
[tox]
envlist=py27,py38

[testenv]
deps=nose
commands=
  nosetests []
```

Test runner args

```
[tox]
envlist=py26,py32

[testenv]
deps=nose
commands=
  # tox -- --with-nicedots --stop
  nosetests []
```

Any test command

```
[tox]
envlist=py26,py32

[testenv]
deps=pytest
commands=
  py.test []
```

Demo

Build Package

Run Tests

Commands to Upload to pypi

Make sure you have a pypi account.

```
$> python3 -m build
$> twine upload -r pypi dist/titanic*
```

More here:
https://packaging.python.org/en/latest/tutorials/packaging-projects/

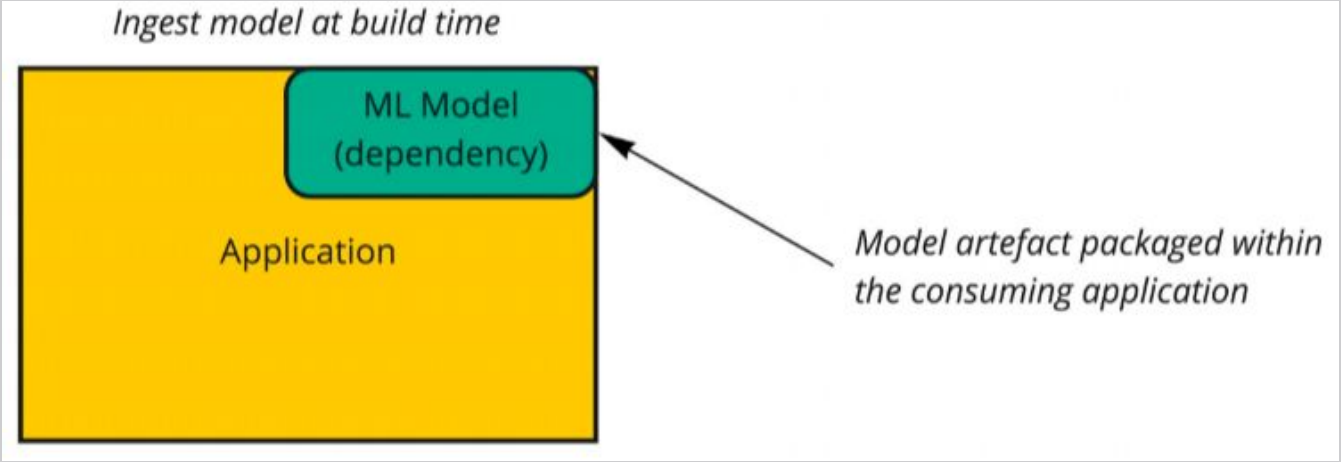
End of Part 1

ML System Architectures

1. Model embedded in application.
2. Served via a dedicated service.
3. Model published as data (streaming).
4. Batch prediction (offline process).

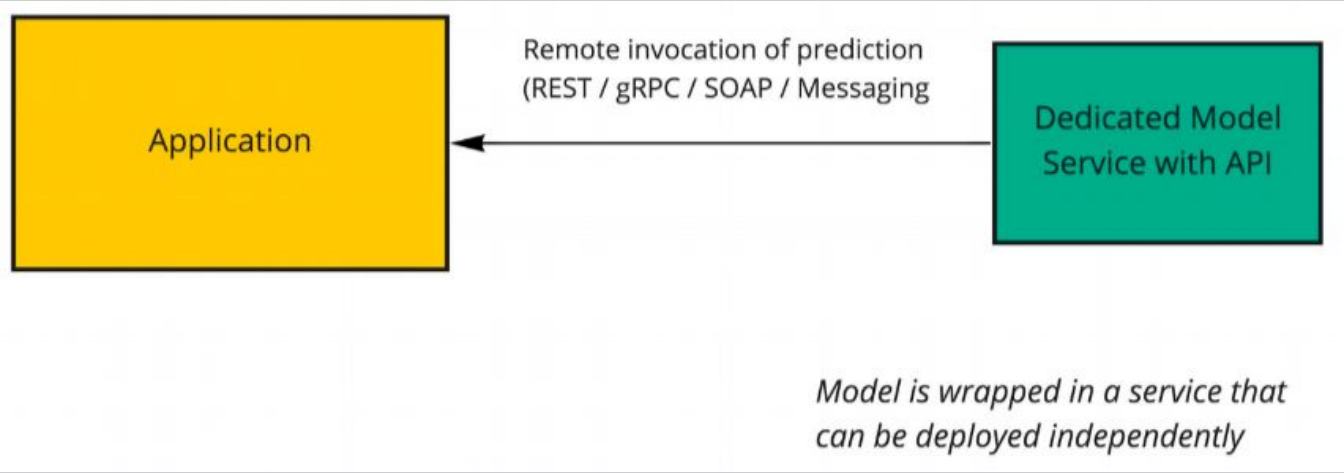
Architecture 1: Embedded

- Pre-Trained.
- Predict on the Fly.



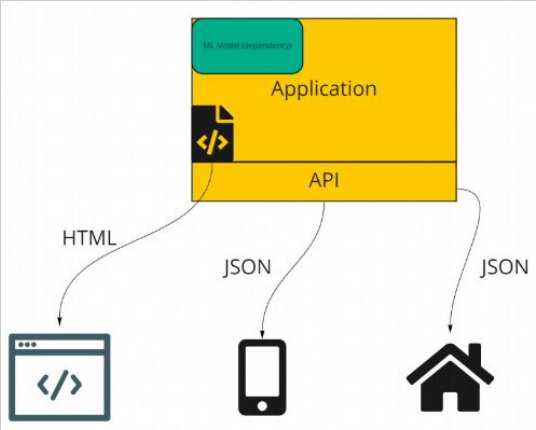
Architecture 2: Dedicated Model API

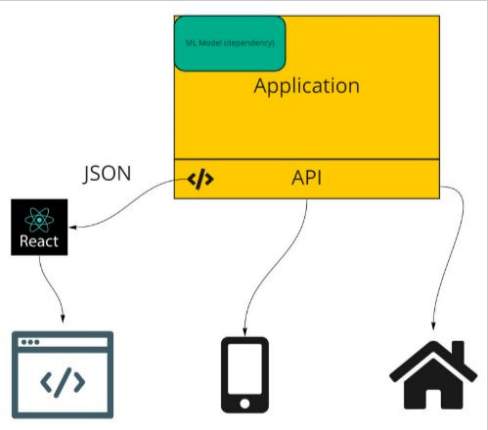
- Pre-Trained.
- Predict on the Fly.



How can our API be Consumed?

- Web browsers (HTML)
- Mobile Devices
- IOT
- Other applications





Web Frameworks

Provide tools to handle & automate standard tasks when building web applications.

- Session management.
- Templating.
- Database Access.
- Much more!.

Python Web Frameworks

	Github Stars	Github Forks	Launch Year
Flask	55.3k	14.3k	2011
Django	57.5k	24.5k	2006
FastAPI	31.1k	2.2k	2019

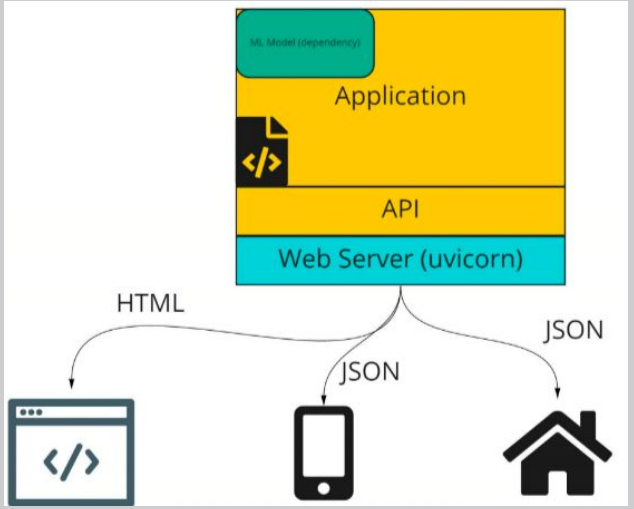
FastAPI

- It's Fast – leverages Python async capabilities.
- Validation with type hints and Pydantic.
- Automatic Documentation.
- Dependency Injection.
- Much more!

https://fastapi.tiangolo.com/

Uvicorn

- For a production deployment, we require a web server.
- Although we tend to use the terms interchangeably, technically the server deals with handling income requests and outgoing responses.
- A server implements a "server gateway interface".



WSGI & ASGI

- Historically, Python web frameworks have used the Web Server Gateway Interface (WSGI) introduced in PEP 333 (and revised in PEP 3333).
- With the introduction of Python's asyncio library it was possible to go beyond the constraints of WSGI (which could not be upgraded without being backwards incompatible). Hence the Asynchonous Server Gateway Interface (ASGI) was born.
- Uvicorn is an implementation of this interface.

Running the API Locally

- After explaining the structure of the API, we can now run it locally.
- We are using `tox` again so here is what you need to do:
- cd into the api folder (titanic-api)
- Then $> `tox -e run`
- Now you should be able to invoke the API.
- Check it on your browser!

What is Docker?

- Put simply, Docker is a tool to make creating, deploying and running containers easy.
- Docker is open source.
- Released in 2013.
- A Docker container is a standardized unit of software development, containing everything that your software application needs to run: code, runtime, system tools, system libraries, etc.
- Containers are created from a read-only template called an image.

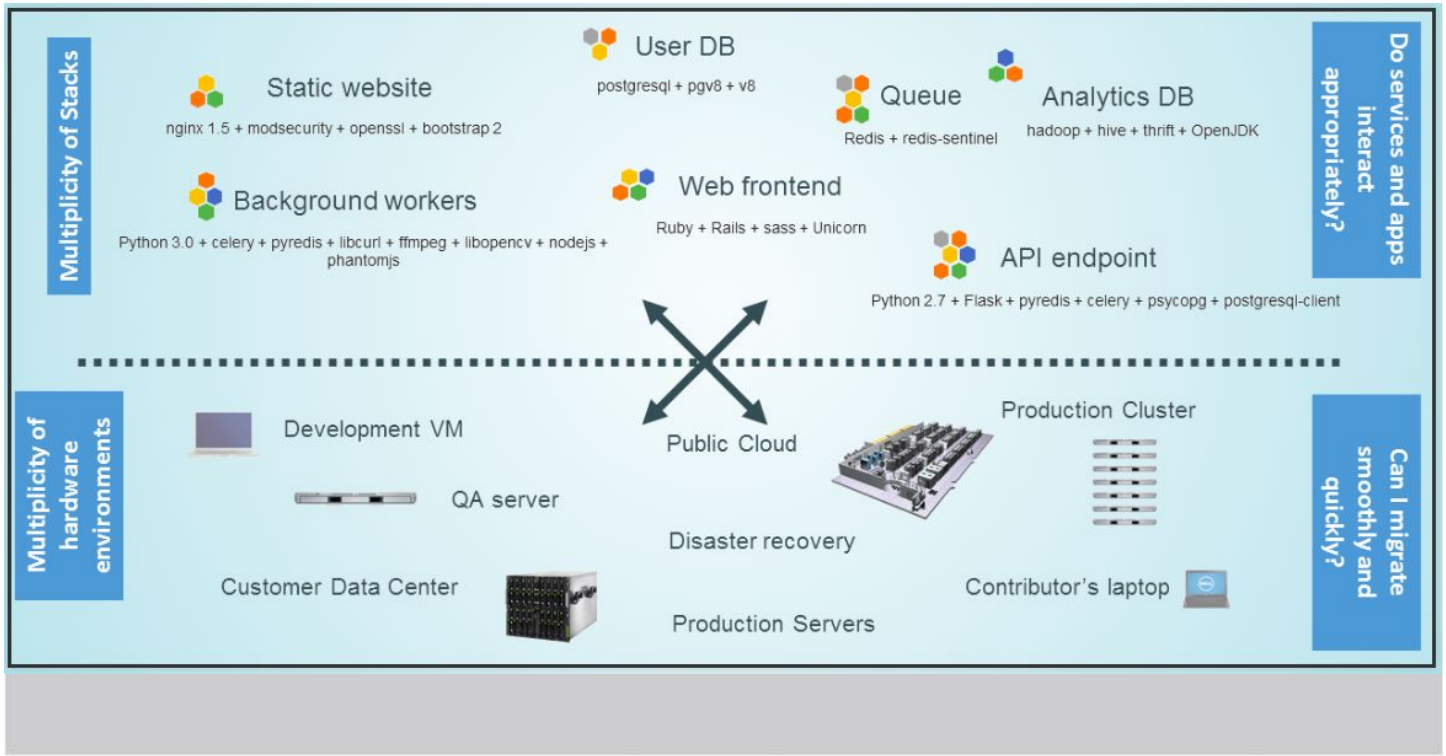
What is a Container?

"A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another."

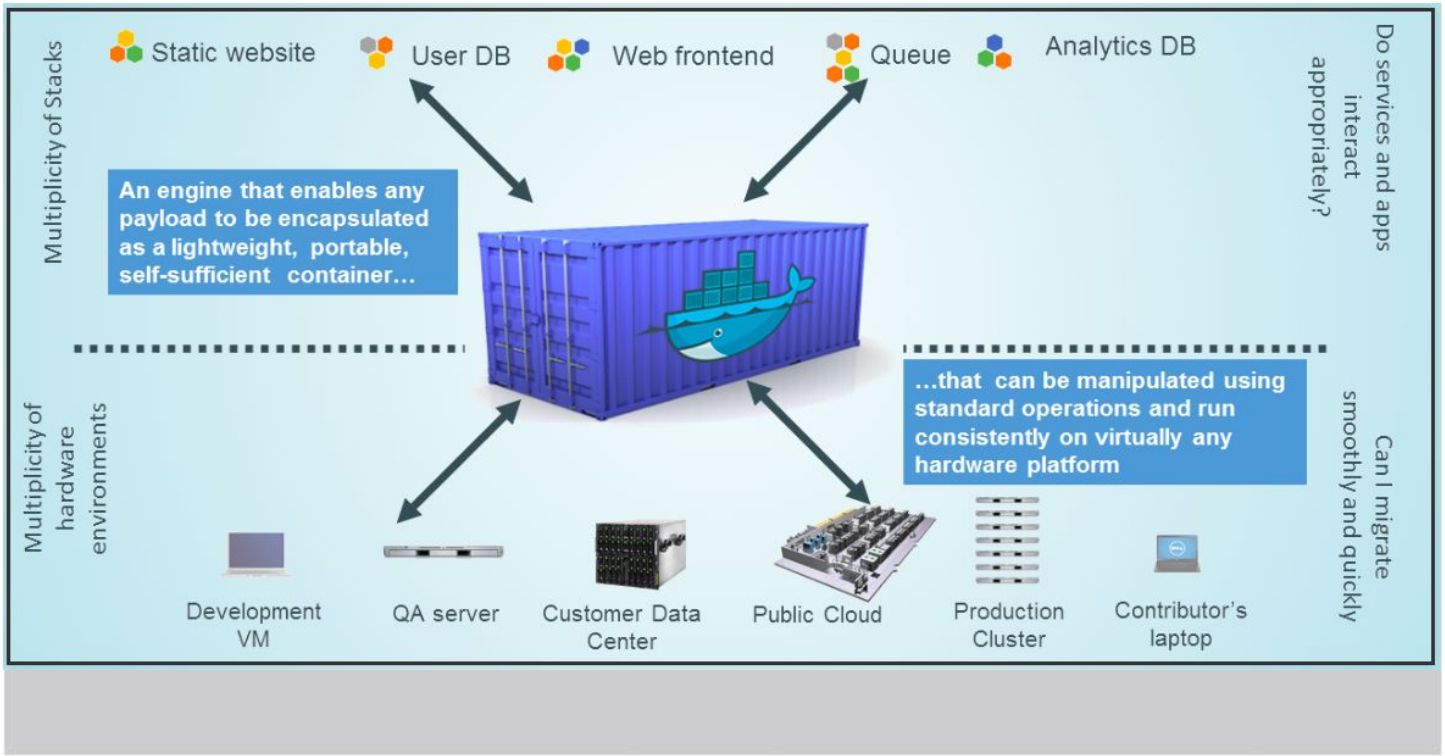
Why use Containers?

- Reproducibility.
- Isolation.
- Simplicity of environment management (Great for making staging/UAT match production).
- Ease of continuous integration.
- Much faster and more lightweight than a VM.
- Container orchestration options (e.g. Kubernetes).
- Docker is the most popular tool for creating and running containers.

Docker


Multiplicity of Stacks
Static website
nginx 1.5 + modsecurity + openssl + bootstrap 2
User DB
postgresql + pgv8 + v8
Queue
Redis + redis-sentinel
Analytics DB
hadoop + hive + thrift + OpenJDK
Do services and apps interact appropriately?
Background workers
Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs
Web frontend
Ruby + Rails + sass + Unicorn
API endpoint
Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client
Multiplicity of hardware environments
Development VM
QA server
Public Cloud
Production Cluster
Disaster recovery
Customer Data Center
Production Servers
Contributor's laptop
Can I migrate smoothly and quickly?

Docker


Multiplicity of Stacks
Static website
User DB
Web frontend
Queue
Analytics DB
Do services and apps interact appropriately?
An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…
…that can be manipulated using standard operations and run consistently on virtually any hardware platform
Multiplicity of hardware environments
Can I migrate smoothly and quickly
Development VM
QA server
Customer Data Center
Public Cloud
Production Cluster
Contributor's laptop

Docker

- A clean, safe, hygienic, portable runtime environment for your app.
- No worries about missing dependencies, packages and other pain points during subsequent deployments.
- Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying.
- Automate testing, integration, packaging...anything you can script.
- Reduce/eliminate concerns about compatibility on different platforms, either your own or your customers.
- Cheap, zero-penalty containers to deploy services. A VM without the overhead of a VM. Instant replay and reset of image snapshots.

Dockerizing our App

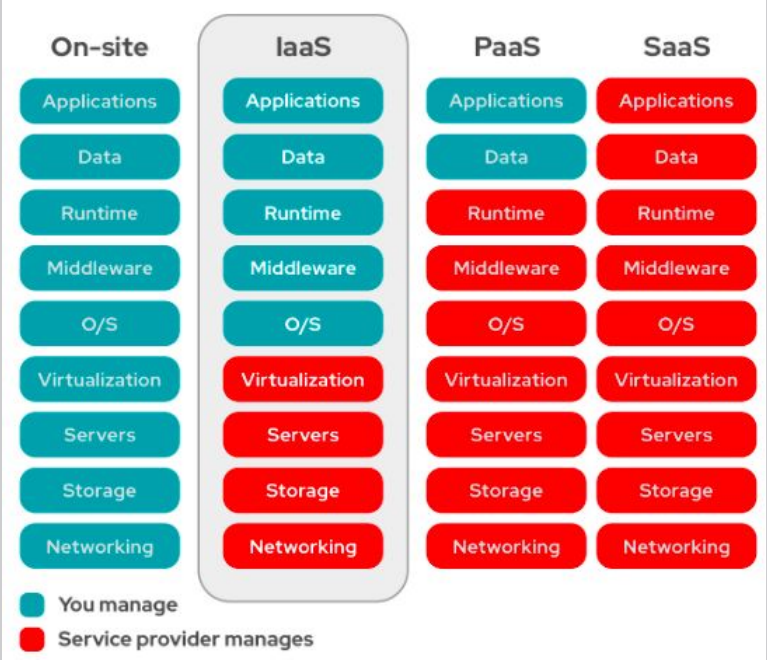
- Dockerfile has been provided.
- Create image: `docker build -t titanic-api .`
- Run a container: `docker run -p 8001:8001 -e PORT=8001 titanic-api`

End of Part 2

IaaS

Infrastructure-as-a-service (IaaS), also known as cloud infrastructure services, is a form of cloud computing in which IT infrastructure is provided to end users through the internet. IaaS is commonly associated with serverless computing.

https://www.redhat.com/en/topics/cloud-computing/what-is-iaas



Amazon Web Services

- Amazon’s cloud computing services.
- Largest provider globally.
- Vast range of services.

Costs

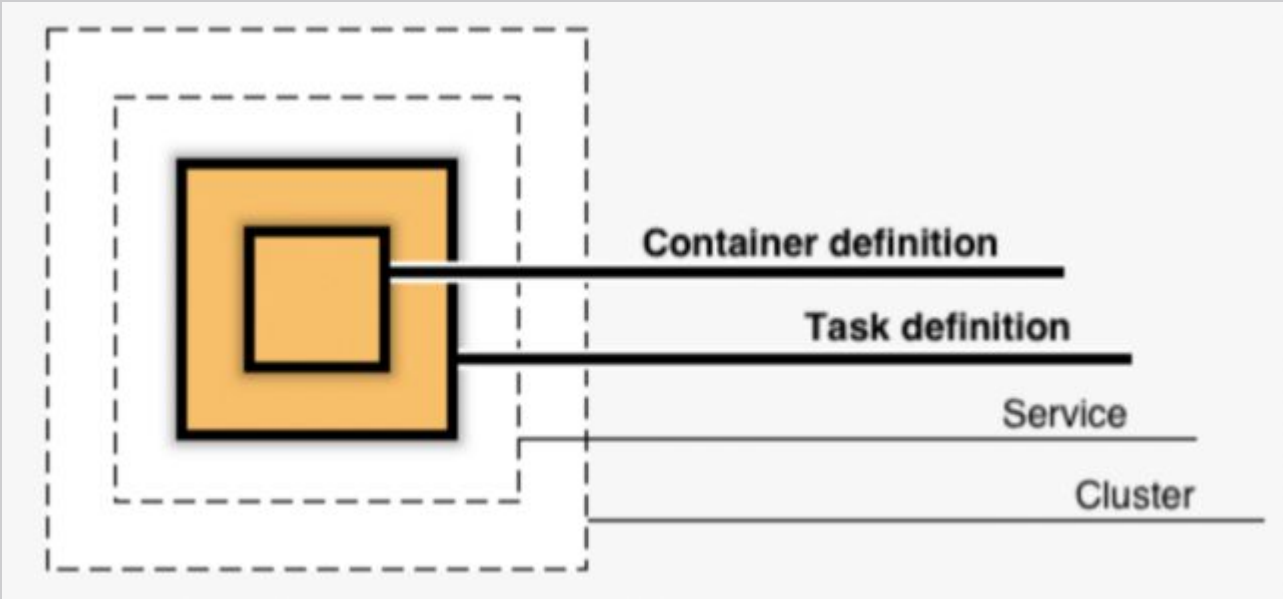
- This section of the course will incur some modest costs (less than $5) *if* you adhere to the following rules:
  - Delete your clusters after use (there is a lecture demonstrating this).
  - Do not create large EC2 instances (stick to t2.micro).
  - Do not upload many images to the registry (although this will be pretty negligible).
- Be careful with security!

What is ECS?

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage Docker containers on a cluster.

https://aws.amazon.com/ecs/

ECS Components



ECS Task Definition

- A task definition is like a blueprint for your application (a bit like a Dockerfile). In this step, you will specify a task definition so Amazon ECS knows which Docker image to use for containers, how many containers to use in the task, and the resource allocation for each container.
- Your entire application stack does not need to exist on a single task definition, and in most cases it should not.

ECS Services

- Amazon ECS allows you to run and maintain a specified number of instances of a task definition simultaneously in an Amazon ECS cluster. This is called a service.
- If any of your tasks should fail or stop for any reason, the Amazon ECS service scheduler launches another instance of your task definition to replace it and maintain the desired count of tasks in the service depending on the scheduling strategy used.

ECS Cluster

- An Amazon ECS cluster is a logical grouping of tasks or services.
- Two main launch types: Fargate and EC2.
- https://docs.aws.amazon.com/AmazonECS/latest/developerguide/clusters.html

Available Options

- Containerization has transformed how we deploy software.
- Key options: Kubernetes, ECS, Docker Swarm.
- Kubernetes is the container orchestration engine of choice...
- ...But has the most complex setup and management .
- Self - Managed vs. Fully managed service.
- Kubernetes is a good choice in many scenarios (but not all).
- New Player: Amazon Elastic Container Service for Kubernetes (EKS).

AWS Account Setup

- Install AWS CLI on your machine
  https://aws.amazon.com/cli/
- Create a user on IAM and assign permissions.
- Configure CLI:

```
aws configure
```

- Provide required access and secret keys
- Verify Access - List S3 Buckets in the account

```
aws s3 ls
```

Steps on ECR

- On ECR Create a private repo (copy the URI, especially the starting ID) - Example: 479320215787.dkr.ecr.us-east-1.amazonaws.com/titanic-ml-api
- Tag it: `docker tag titanic-ml-api:latest 479320215787.dkr.ecr.us-east-1.amazonaws.com/titanic-ml-api:latest`
- Login to ECR: `aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 479320215787.dkr.ecr.us-east-1.amazonaws.com`
- Push image: `docker push 479320215787.dkr.ecr.us-east-1.amazonaws.com/titanic-ml-api:latest`
- Now let's use Fargate (Launch Method because it requires less config).

Steps on ECS

- Copy the latest image URI from ECR
- Go to ECS -> clusters -> Get started (this will show we're using FarGate) -> Custom container definition -> Fill out details (remember PORT and env variable)
- When ready, go to View Service -> Tasks
- When Status is RUNNING click on Task -> Check Logs -> Copy Public IP and paste it in the browser with port 8001

рахмат
danke
謝謝
ngiyabonga
teşekkür ederim
Баярлалаа
spas
mersi
kia ora
barka
welalin
tack
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
спасибі
blagodaram
dank je
misaotra
matondo
paldies
grazzi
gracias
хвала
kiitos
dankie
nanni
nandri
dhanyavad
mauruuru
thank you
asante
manana
obrigada
tenki
bedankt
bayarlalaa
gracie
hvala
köszönöm
akun
dankon
aċiū
djiere dieuf
tau
mochchakkeram
murakoze
enkosi
dziękuję
дякую
mamnun
chokrane
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
sobodi
dēkuji
sagolun
sukriya
kop khun krap
taiku
obrigado
mèsi
didi madloba
ありがとう
tanemirt
grazie
arigatô
takk
dakujem
trugarez
najis tuke
terima kasih
rahmet
dhanyavadagalu
shukriya
mercê
мерси
kam sah hamnida
rahmat
diolch
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ
merci