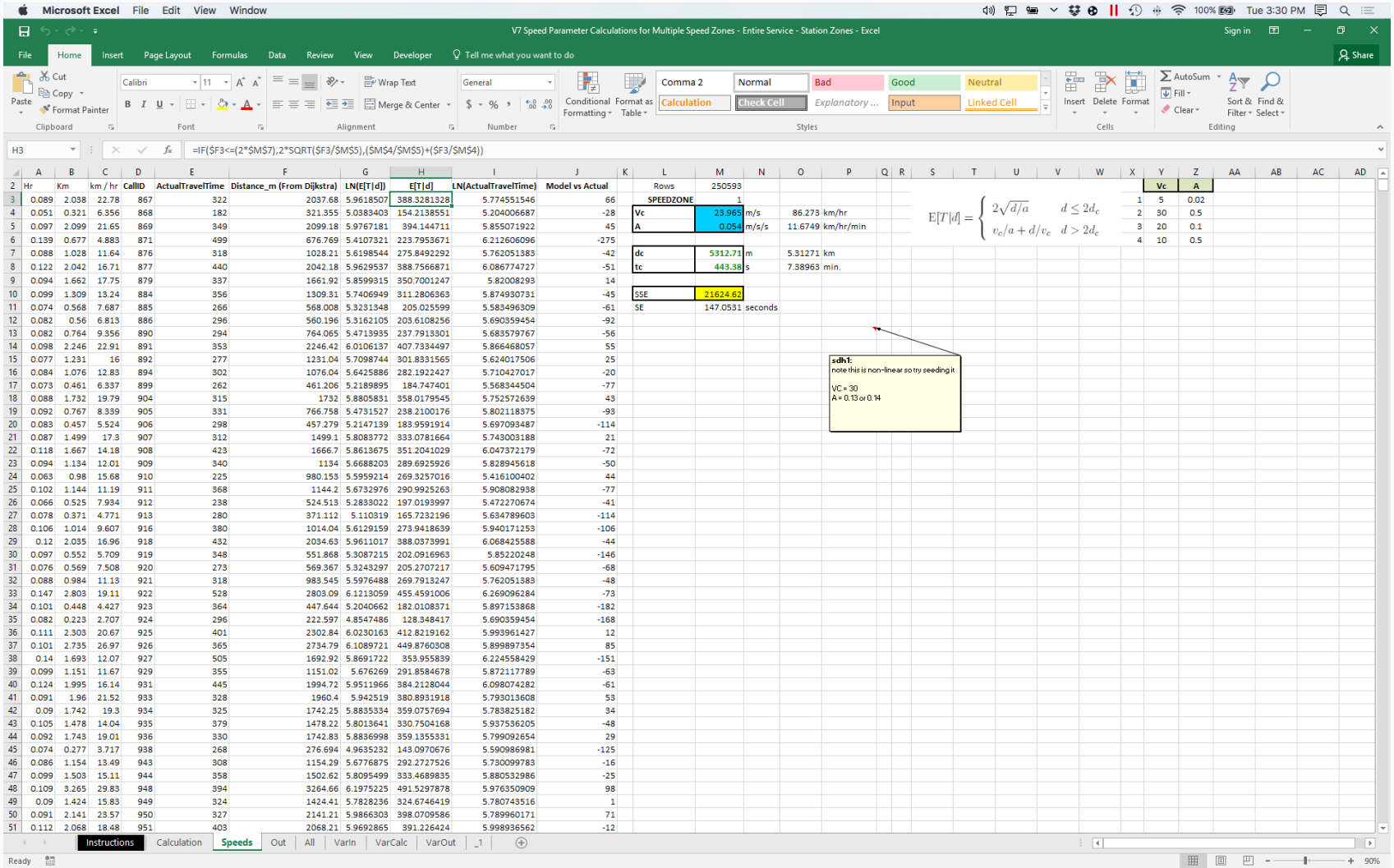
A talker on Docker:

How containers can make your work more reproducible, accessible, and ready for production.

Finbarr Timbers, Analyst, Darkhorse Analytics

Three stories.

One: Moving a nonlinear regression from Excel to Python.



One: Moving a nonlinear regression from Excel to Python.

At Darkhorse, we had a nonlinear regression that we had been running for several years. We used Excel to calculate the parameters of the model, which we would then input into our emergency services application.

Problems:

1. It was slow.

One: Moving a nonlinear regression from Excel to Python.

At Darkhorse, we had a nonlinear regression that we had been running for several years. We used Excel to calculate the parameters of the model, which we would then input into our emergency services application.

Problems:

1. It was slow.
2. Hard to automate.

One: Moving a nonlinear regression from Excel to Python.

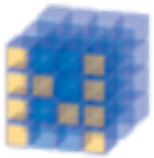
At Darkhorse, we had a nonlinear regression that we had been running for several years. We used Excel to calculate the parameters of the model, which we would then input into our emergency services application.

Problems:

1. It was slow.
2. Hard to automate.
3. Hard to test.

One: Moving a nonlinear regression from Excel to Python.

The solution:



NumPy

Base N-dimensional array package

SciPy library

Fundamental library for scientific computing

One: Moving a nonlinear regression from Excel to Python.

120 lines of code later, I was done. 10000x speedup.

But…

“Hey Finbarr, can you help? The code doesn’t seem to run.”

Two.

2. Data scientist needs to run Tensorlfow for 1 000 000 epochs.

Three

3. We need deploy a Python application to a client's system if they don't have python installed?

Is there common thread?

Solution?

Fiddle with your computer until it works!

If only there was something better...

Ideal solution would be:

1. Portable. It works, quickly & easily, on every computer.

Ideal solution would be:

1. Portable. It works, quickly & easily, on every computer.
2. Easy to configure.

Ideal solution would be:

1. Portable. It works, quickly & easily, on every computer.
2. Easy to configure.
3. Compatible with production environment.

Ideal solution would be:

1. Portable. It works, quickly & easily, on every computer.
2. Easy to configure.
3. Compatible with production environment.
4. Works on Windows, OS X, and Linux.

Ideal solution would be:

1. Portable. It works, quickly & easily, on every computer.
2. Easy to configure.
3. Compatible with production environment.
4. Works on Windows, OS X, and Linux.
5. Independent— can run multiple independently.

Ideal solution would be:

1. Portable. It works, quickly & easily, on every computer.
2. Easy to configure.
3. Compatible with production environment.
4. Works on Windows, OS X, and Linux.
5. Independent— can run multiple independently.
6. Historical— works forever.

Production

- DOn't need to worry about depenecnies (don't need to instlall all of the *extremely heavy* dependencies- numpy, scipy, pandas, sklearn, etc.

Production

- DOn’t need to worry about depenecnies (don’t need to instlall all of the *extremely heavy* dependencies- numpy, scipy, pandas, sklearn, etc.
- Can easily move your applcation around

Production

- DOn’t need to worry about depenecnies (don’t need to instlall all of the *extremely heavy* dependencies- numpy, scipy, pandas, sklearn, etc.
- Can easily move your applcation around
- Less fragile

Production

- DOn’t need to worry about depenecnies (don’t need to instlall all of the *extremely heavy* dependencies- numpy, scipy, pandas, sklearn, etc.
- Can easily move your applcation around
- Less fragile
- Drop: “daemons we don’t need, virtualization layers, duplicated functions, large iamges”

Example

1. We need python 3.2. Damn, distro doesn't provide it.

Example

1. We need python 3.2. Damn, distro doesn't provide it.
2. Need a library that clashes with the system's version. Damn.

Example

1. We need python 3.2. Damn, distro doesn't provide it.
2. Need a library that clashes with the system's version. Damn.
3. We package the whole mess, using virtualenv and some glue.

Example

1. We need python 3.2. Damn, distro doesn't provide it.
2. Need a library that clashes with the system's version. Damn.
3. We package the whole mess, using virtualenv and some glue.
4. Oh, and we need redis 2.6 for Lua support. Distro has 2.4. Triple damn.

Example

1. We need python 3.2. Damn, distro doesn't provide it.
2. Need a library that clashes with the system's version. Damn.
3. We package the whole mess, using virtualenv and some glue.
4. Oh, and we need redis 2.6 for Lua support. Distro has 2.4. Triple damn.
5. Oh, and the QA team has standardized on Centos.

Example

1. We need python 3.2. Damn, distro doesn't provide it.
2. Need a library that clashes with the system's version. Damn.
3. We package the whole mess, using virtualenv and some glue.
4. Oh, and we need redis 2.6 for Lua support. Distro has 2.4. Triple damn.
5. Oh, and the QA team has standardized on Centos.
6. Oh, BTW, the new search service is a node.js app, that's cool, right?

Easy to deploy


docker

What is Docker?

- Allows for the automatic deployment of “containers”

What is Docker?

- Allows for the automatic deployment of “containers”
- Containers are lightweight VMs that wrap up code with everything needed to run

What is Docker?

- Allows for the automatic deployment of “containers”
- Containers are lightweight VMs that wrap up code with everything needed to run
- “Write once run everywhere”

What is Docker?

- Allows for the automatic deployment of “containers”
- Containers are lightweight VMs that wrap up code with everything needed to run
- “Write once run everywhere”
- Easy to write and use

Demos!

Benefits

1. Super fast to make code changes.

Benefits

1. Super fast to make code changes.
2. Snapshot in time- doesn’t update without being told.

Benefits

1. Super fast to make code changes.
2. Snapshot in time- doesn't update without being told.
3. Easy to push far away.

Benefits

1. Super fast to make code changes.
2. Snapshot in time- doesn't update without being told.
3. Easy to push far away.
4. Can run on bare metal (Docker v. Heroku)

Drawbacks

1. Slows down iteration speed.

Drawbacks

1. Slows down iteration speed.
2. Logging/exporting results (currently...)

Drawbacks

1. Slows down iteration speed.
2. Logging/exporting results (currently...)
3. Secrets. Most people migrating to containers rely on configuration management to provision secrets on machines securely; however, continuing down the path of configuration management for secrets in containers is clunky. Another alternative is distributing them with the image, but that poses security risks and makes it difficult to securely recycle images between development, CI, and production. The most pure solution is to access secrets over the network, keeping the filesystem of containers stateless. Until recently nothing container-oriented existed in this space, but recently two compelling secret brokers, Vault and Keywhiz, were open-sourced. At Shopify we developed ejson a year and a half

Drawbacks

1. Slows down iteration speed.
2. Logging/exporting results (currently...)
3. Secrets. Most people migrating to containers rely on configuration management to provision secrets on machines securely; however, continuing down the path of configuration management for secrets in containers is clunky. Another alternative is distributing them with the image, but that poses security risks and makes it difficult to securely recycle images between development, CI, and production. The most pure solution is to access secrets over the network, keeping the filesystem of containers stateless. Until recently nothing container-oriented existed in this space, but recently two compelling secret brokers, Vault and Keywhiz, were open-sourced. At Shopify we developed ejson a year and a half

Docker CLI basics

Using it

- any containers running? `docker ps`

Using it

- any containers running? `docker ps`
- run the ubuntu base (change to be more DSey `docker run -i -t ubuntu:12.10 bash`

Using it

- any containers running? `docker ps`
- run the ubuntu base (change to be more DSey `docker run -i -t ubuntu:12.10 bash`
- tag it `docker commit -m "installed redis server"` `docker images`

expose to SQL Server

- show bitbucket repo

expose to SQL Server

- show bitbucket repo
- discuss problems we had

expose to SQL Server

- show bitbucket repo
- discuss problems we had
- show how Docker fixes it (should work on non-OS X systems)

Create example api based on speed paramter optimizations

- create locally, copies file, runs, spits out back `docker run --net host -d --name myiris pythoniris`

Create example api based on speed paramter optimizations

- create locally, copies file, runs, spits out back `docker run --net host -d --name myiris pythoniris`
- works on Windows

Create example api based on speed paramter optimizations

- create locally, copies file, runs, spits out back `docker run --net host -d --name myiris pythoniris`
- works on Windows
- API-ify it and deploy to GCE

Example app

- time it

```
root♫precise64:~# docker stop 8222
root♫precise64:~# time docker start 8222
    real    0m0.150s

root♫precise64:~# time service redis-server start
Starting redis-server: redis-server.
    real    0m0.165s

root♫precise64:~# time docker run -p 6379 -d -i -t jbarra
    real    0m0.147s
```

Install service manually

Now, let’s automate it

Sample Dockerfile:

- Now, edit the code and redploy

Now, let’s automate it

Sample Dockerfile:

- Now, edit the code and redploy
- Layering dockerfiles

Competitors

Pricing

- Check out Heroku

Security

- less stuff on the box, less vectors

Security

- less stuff on the box, less vectors
- layering dockerfiles

Links