

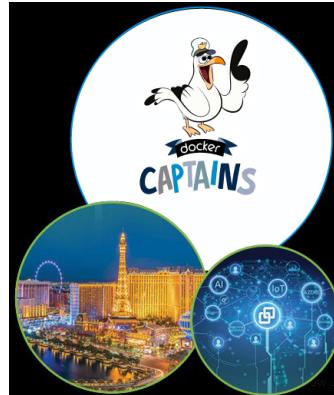
# Depend on Docker

## Get IT done with Docker on Azure



Microsoft Azure  
+ AI Conference

CO-PRODUCED BY  
Microsoft & DEVintersection





## Alex Iankoulski

Principal Software Architect

Baker Hughes, a GE Company  
<https://www.linkedin.com/in/alex-iankoulski/>

**BAKER HUGHES**  
a GE company



## Mahadevan Balasubramaniam

Principal Data Scientist

Baker Hughes, a GE Company  
<https://www.linkedin.com/in/bmdevan/>



## Arun Subramaniyan

VP Data Science & Analytics

Baker Hughes, a GE Company  
<https://www.linkedin.com/in/arunsubramaniyan/>

**Microsoft Azure**  
+AI Conference  
CO-PRODUCED BY  
Microsoft & DEVintersection

© BHGE. All rights reserved

# Some problems we care about



**Every 2  
Seconds**

A GE Jet engine takes  
off in the world



**~30%**

Of the world's power  
is produced by a GE  
turbine



**We are the first**

Fullstream O&G Company  
Exploration, Production,  
Transportation, Operations

## We need to model mission critical systems accurately

# Scale and complexity of industrial data

## O&G

**Drilling data**  
0.3GB/well/day

**Wireline data**  
5GB/well/day

**Fiber optic data**  
0.1GB/well/day

**ESP monitoring**  
0.4GB/well/day

**Seismic data**  
500GB/survey

## Aviation

**A flight from NY to London**  
~1 TB

**Atmospheric data**  
0.1GB/day

## Inspections

**Ultrasound: tubes**  
1.2TB/8 hours

**Corrosion monitoring**  
1GB/site/year  
**Pipeline inspection**  
1.5TB/600 km

## ERP Systems

**Transactional data**  
**Logistics reports**  
**Maintenance logs**  
**FSE reports**  
**Inventory reports**

**10 - 100x More volume**

**100 - 1000x More velocity**

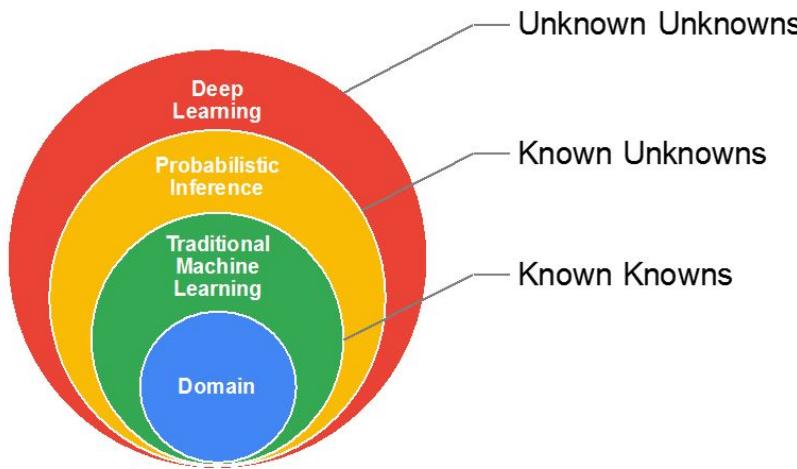
Hundreds of assets, multiple data silos

ERP  
Systems

Industrial

# Digital Twins

A live up-to-date digital representation of an asset, system, or process.



## Hybrid models

Physics-based



Probabilistics

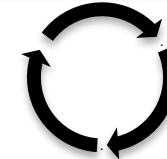


Deep learning  
(AI)

Known Knowns

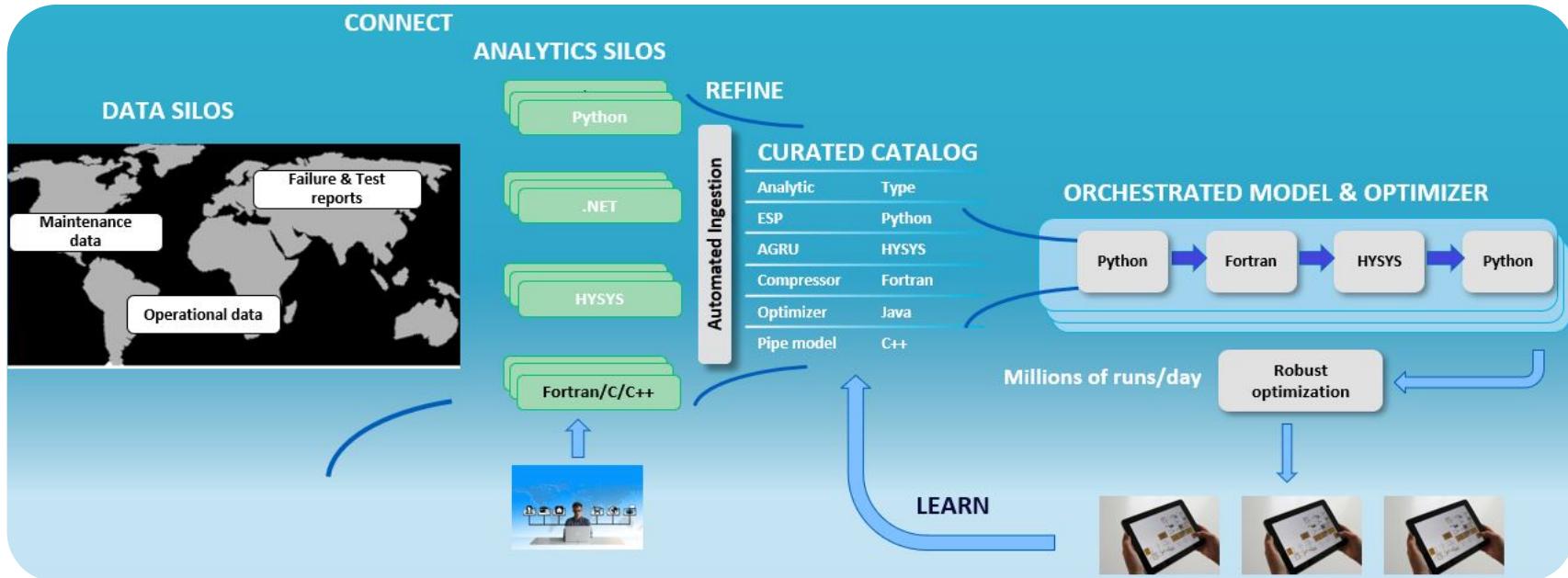
Known Unknowns

Unknown Unknowns



- Continuously tuned
- Scalable
- Adaptable

# Connecting Data & Analytics



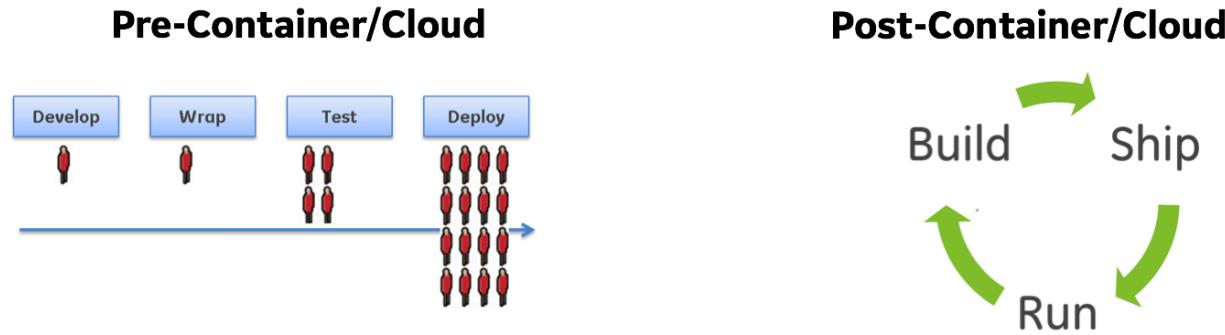
**SILOS OF DATA/ANALYTICS + FEDERATION + SCALE + CONSTANT LEARNING = SCALABLE COMPUTING**

Microsoft Azure  
+AI Conference

CO-PRODUCED BY  
Microsoft & DEVintersection

© BHGE. All rights reserved

# Democratize analytics through containers and cloud



- Production deployment: 6-12 months
  - New developer setup: 1 week
  - Quarterly releases
- Production: ~ 1 hour
  - New developer setup: 1 hour
  - **Continuous releases**

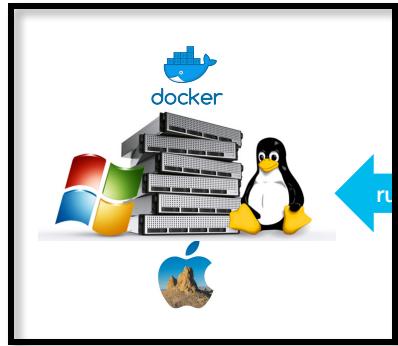
**Kubernetes allows polyglot auto-scaling and “self”-healing at scale**

# Scale up & down at will

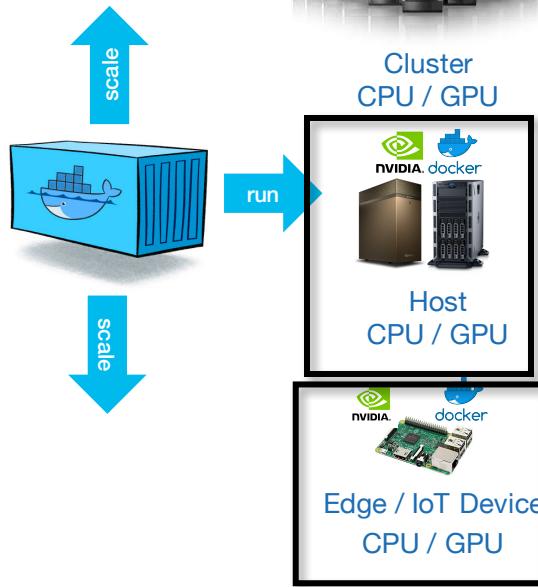
- Ship and Run anywhere

## Demo 1

### Analytics in Different OS



Operating System



Platform



MESOS



kubernetes

## Demo 4

### Scale Up / Down Containers



## Demo 2

### Leverage GPU's



## Demo 3

### Encrypted Transfer



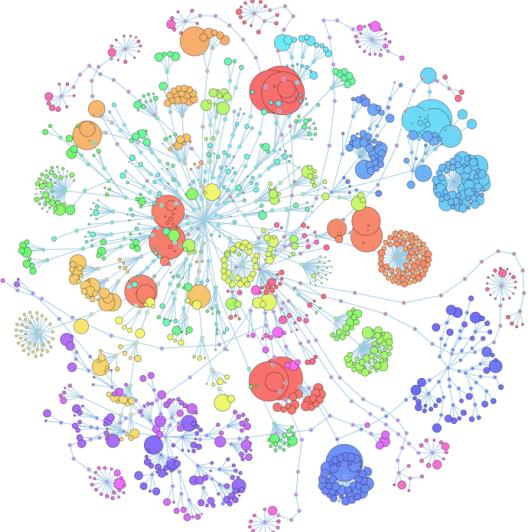
Orchestrator

Infrastructure

# Depend on Docker (DoD)

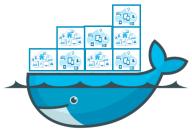
- A philosophy – don't install anything, build a container instead
- A type of project – needs only access to source and Docker
- Open source – <https://github.com/bhgedigital/depend-on-docker>
- Easy to use – create a new DoD project with a single command
- Flexible – defaults provided, but customizable
- Universal – works with Linux and Windows containers

# DOD Example - SEMTK



Example: semtk-opensource  
Code Flower

CONTAINERIZED



Containerized application  
{dependencies encapsulated,  
complexity reduced}

<https://github.com/ge-semtk/semtk>

```
> iankouls@1403111-C02T82RGGTFM:~/Projects/2018/github.com/iankoulski/semtk
> ./compose.sh up -d
Creating network "host_semtknet" with driver "bridge"
Creating host_semtk-sparqlgraph-results_1 ... done
Creating host_semtk-nodegroup-store_1 ... done
Creating host_semtk-ingestion_1 ... done
Creating host_semtk-hive_1 ... done
Creating host_semtk-sparqldb_1 ... done
Creating host_semtk-nodegroup_1 ... done
Creating host_semtk-dispatch_1 ... done
Creating host_semtk-nodegroup-execution_1 ... done
Creating host_semtk-sparql-query_1 ... done
Creating host_semtk-sparqlgraph-status_1 ... done
Creating host_semtk-sparqlgraph-web_1 ... done
Creating host_semtk-ontology-info_1 ... done
iankouls@1403111-C02T82RGGTFM:~/Projects/2018/github.com/iankoulski/semtk
> ./compose.sh ps
          Name           Command     State            Ports
-----  -----
host_semtk-dispatch_1   /bin/sh -c cd /service; ls ...  Up      0.0.0.0:12053->8080/tcp
host_semtk-hive_1       /bin/sh -c cd /service; ls ...  Up      0.0.0.0:12055->8080/tcp
host_semtk-ingestion_1  /bin/sh -c cd /service; ls ...  Up      0.0.0.0:12091->8080/tcp
host_semtk-nodegroup-execution_1 /bin/sh -c cd /service; ls ...  Up      0.0.0.0:12058->8080/tcp
host_semtk-nodegroup-store_1 /bin/sh -c cd /service; ls ...  Up      0.0.0.0:12056->8080/tcp
host_semtk-nodegroup_1   /bin/sh -c cd /service; ls ...  Up      0.0.0.0:12059->8080/tcp
host_semtk-ontology-info_1 /bin/sh -c cd /service; ls ...  Up      0.0.0.0:12057->8080/tcp
host_semtk-sparql_query_1 /bin/sh -c cd /service; ls ...  Up      0.0.0.0:12050->8080/tcp
host_semtk-sparqldb_1    /bin/sh -c /startup.sh        Up      0.0.0.0:1111->1111/tcp, 0.0.0.0:2420->8890/tcp
host_semtk-sparqlgraph-results_1 /bin/sh -c cd /service; ls ...  Up      0.0.0.0:12052->8080/tcp
host_semtk-sparqlgraph-status_1 /bin/sh -c cd /service; ls ...  Up      0.0.0.0:12051->8080/tcp
host_semtk-sparqlgraph-web_1 /bin/sh -c /usr/local/tomc ... Up      0.0.0.0:8860->8080/tcp
iankouls@1403111-C02T82RGGTFM:~/Projects/2018/github.com/iankoulski/semtk
> |
```

# Demo 1 – Depend on Docker Project

GitHub



<https://github.com/bhgedigital/depend-on-docker>

bitly

<http://bit.ly/dodockerw>

Docker Hub



`docker run bhgedigital/win-svn`

Microsoft Azure  
+ AI Conference

CO-PRODUCED BY  
Microsoft & DEVintersection

© BHGE. All rights reserved

# Demo 1: Step 1 – View project on GitHub (copy command)

GitHub, Inc. [US] https://github.com/bhgdigital/depend-on-docker

No results < > Options

windows	Update env.cmd	6 months ago
.gitignore	Add slides from DockerCon presentation	4 months ago
DoLogo.png	Add License and Logo	6 months ago
LICENSE	Add License and Logo	6 months ago
README.md	Added blog link to README	a month ago
create.bat	bhgdigital.org	6 months ago
create.sh	Add handling of relative paths to create.sh	6 months ago

README.md

**depend-on-docker**

Depend on Docker (DoD) is an open source project that makes any software easy to build, ship, and run! It removes complexity by containerizing your executables and all of their dependencies.

**Your only dependency**

If you haven't done so already, please install [Docker](#)

**Create your depend-on-docker project**

This project works on both Linux and Windows. It strives to drastically simplify your development process and make it easy to build Docker containers by providing an intuitive project template. To create a project, you can execute the corresponding script, or just run the command line relevant to your operating system below.



# Demo 1: Step 2 – Execute command to get create.bat

```
Administrator: Command Prompt - cmd --help

C:\demo\depend-on-docker>curl -L http://bit.ly/dodockerw > create.bat
% Total    % Received % Xferd  Average Speed   Time     Time      Current
                                         Dload  Upload   Total   Spent   Left  Speed
100    167  100    167     0      0    668      0 --:--:-- --:--:-- --:--:--  668
100    276  100    276     0      0    630      0 --:--:-- --:--:-- --:--:--  630

C:\demo\depend-on-docker>dir
Volume in drive C is Windows
Volume Serial Number is 5010-2DA1

Directory of C:\demo\depend-on-docker

12/05/2018  10:10 PM    <DIR>        .
12/05/2018  10:10 PM    <DIR>        ..
12/05/2018  10:10 PM                276 create.bat
                           1 File(s)       276 bytes
                           2 Dir(s)  100,761,288,704 bytes free

C:\demo\depend-on-docker>
```

# Demo 1: Step 3 – Execute create.bat [new\_project\_path]

```
Administrator: Command Prompt - cmd --help

C:\demo\depend-on-docker>create.bat c:\demo\depend-on-docker\azureai
55ab9d1b4fee4c54fd84166c7cc1bbfafbc4daf666457d1ef9305eaa22848b6d
C:\demo\depend-on-docker>dir
Volume in drive C is Windows
Volume Serial Number is 5010-2DA1

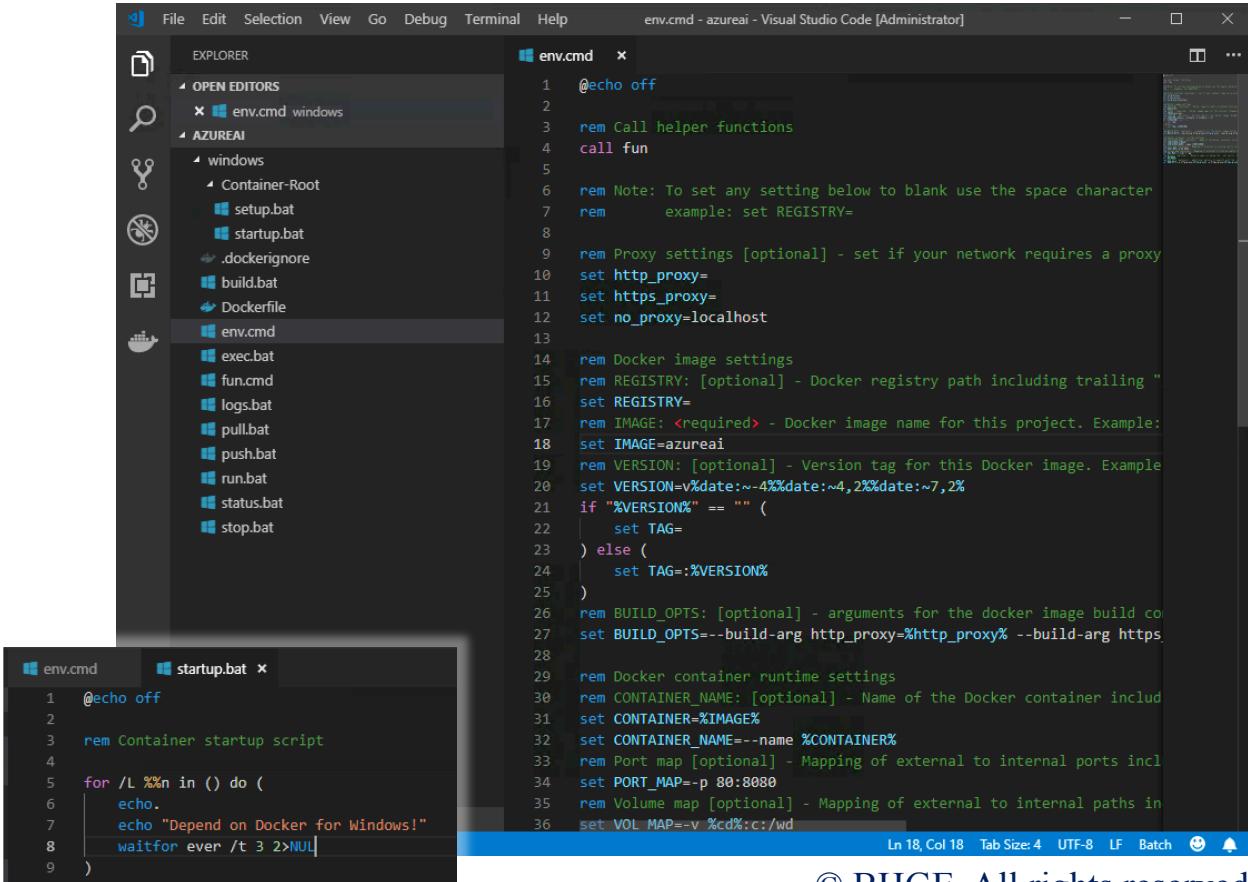
Directory of C:\demo\depend-on-docker

12/05/2018  10:17 PM    <DIR>          .
12/05/2018  10:17 PM    <DIR>          ..
12/05/2018  10:17 PM    <DIR>          azureai
12/05/2018  10:10 PM                276  create.bat
                           1 File(s)        276 bytes
                           3 Dir(s)  100,760,715,264 bytes free

C:\demo\depend-on-docker>
```

# Demo 1: Step 4 – Customize your project

- **Dockerfile** – change base image and author info as needed
  - **env.cmd** – image and container settings
  - **setup.bat** – add commands to run while image is being built
  - **startup.bat** – add commands to run when the container starts up,
- Sample →



The screenshot shows the Visual Studio Code interface with two tabs open: "env.cmd" and "startup.bat".

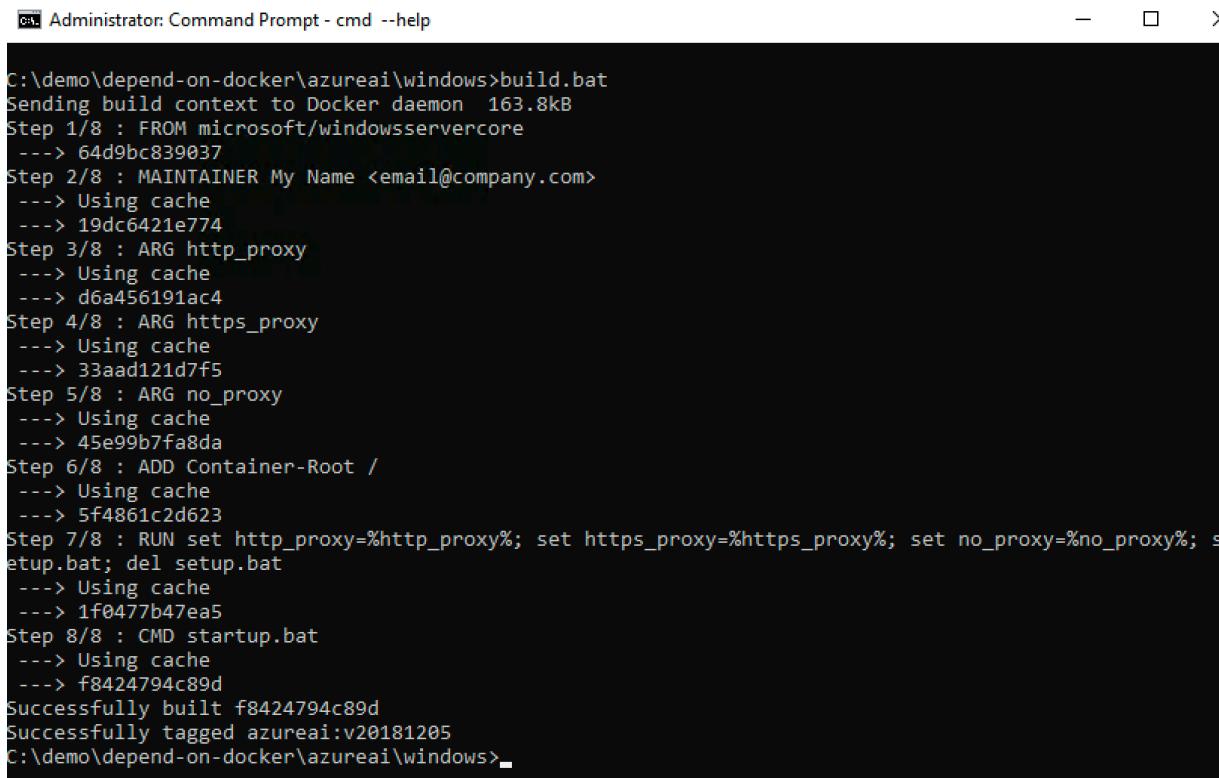
The "env.cmd" tab contains the following content:

```
1 @echo off
2
3 rem Call helper functions
4 call fun
5
6 rem Note: To set any setting below to blank use the space character
7 rem example: set REGISTRY=
8
9 rem Proxy settings [optional] - set if your network requires a proxy
10 set http_proxy=
11 set https_proxy=
12 set no_proxy=localhost
13
14 rem Docker image settings
15 rem REGISTRY: [optional] - Docker registry path including trailing "
16 set REGISTRY=
17 rem IMAGE: <required> - Docker image name for this project. Example:
18 set IMAGE=azureai
19 rem VERSION: [optional] - Version tag for this Docker image. Example
20 set VERSION=%date:~0,4%-%date:~4,2%-%date:~7,2%
21 if "%VERSION%" == "" (
22     set TAG=
23 ) else (
24     set TAG=%VERSION%
25 )
26 rem BUILD_OPTS: [optional] - arguments for the docker image build command
27 set BUILD_OPTS--build-arg http_proxy=%http_proxy% --build-arg https_
28
29 rem Docker container runtime settings
30 rem CONTAINER_NAME: [optional] - Name of the Docker container includ
31 set CONTAINER_NAME=%IMAGE%
32 set CONTAINER_NAME==name %CONTAINER%
33 rem Port map [optional] - Mapping of external to internal ports incl
34 set PORT_MAP=p 80:8080
35 rem Volume map [optional] - Mapping of external to internal paths in
36 set VOL_MAP=v %cd%:::/wd
```

The "startup.bat" tab contains the following content:

```
1 @echo off
2
3 rem Container startup script
4
5 for /L %%n in () do (
6     echo.
7     echo "Depend on Docker for Windows!"
8     waitfor ever /t 3 2>NUL
9 )
```

# Demo 1: Step 5 – Build your project



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt - cmd --help". The command run is "C:\demo\depend-on-docker\azureai\windows>build.bat". The output details the Docker build process:

```
C:\demo\depend-on-docker\azureai\windows>build.bat
Sending build context to Docker daemon 163.8kB
Step 1/8 : FROM microsoft/windowsservercore
--> 64d9bc839037
Step 2/8 : MAINTAINER My Name <email@company.com>
--> Using cache
--> 19dc6421e774
Step 3/8 : ARG http_proxy
--> Using cache
--> d6a456191ac4
Step 4/8 : ARG https_proxy
--> Using cache
--> 33aad121d7f5
Step 5/8 : ARG no_proxy
--> Using cache
--> 45e99b7fa8da
Step 6/8 : ADD Container-Root /
--> Using cache
--> 5f4861c2d623
Step 7/8 : RUN set http_proxy=%http_proxy%; set https_proxy=%https_proxy%; set no_proxy=%no_proxy%; setup.bat; del setup.bat
--> Using cache
--> 1f0477b47ea5
Step 8/8 : CMD startup.bat
--> Using cache
--> f8424794c89d
Successfully built f8424794c89d
Successfully tagged azureai:v20181205
C:\demo\depend-on-docker\azureai\windows>
```

# Demo 1: Step 6 – run.bat, status.bat, logs.bat, stop.bat

Administrator: Command Prompt - cmd --help

```
C:\demo\depend-on-docker\azureai\windows>run.bat
a7ad2ab7c853eccce8cfee14f31f331d57ddd8d8539ba19eb6f4ddfc6fc1fe65
C:\demo\depend-on-docker\azureai\windows>status.bat
a7ad2ab7c853      azureai:v20181205  "cmd /S /C startup.bat"  29 seconds ago  Up 4 seconds
      0.0.0.0:80->8080/tcp  azureai
C:\demo\depend-on-docker\azureai\windows>logs.bat

"Depend on Docker for Windows!"

"Depend on Docker for Windows!"

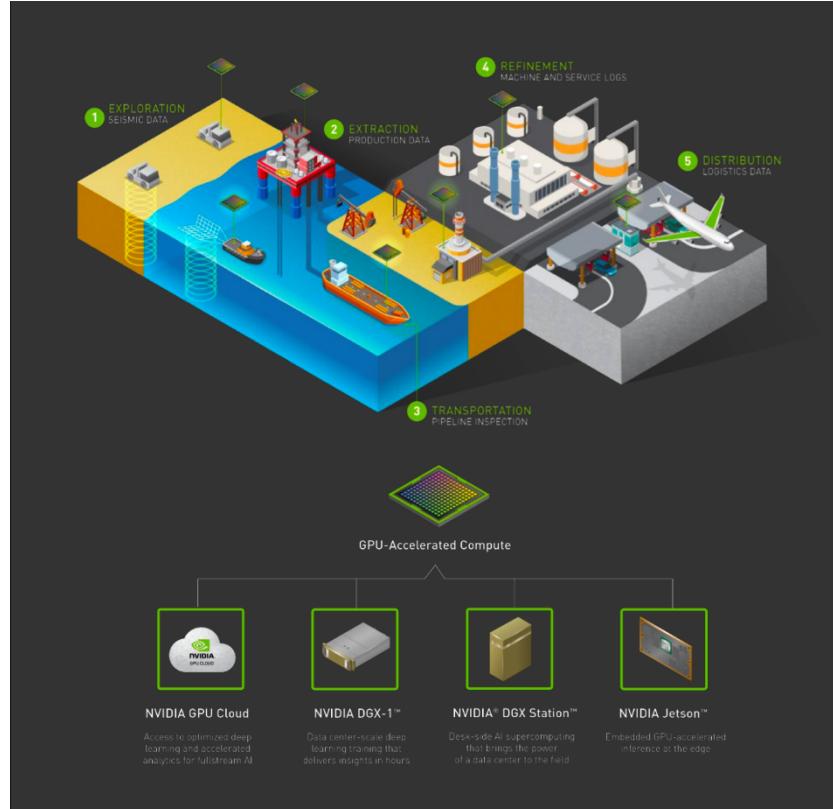
"Depend on Docker for Windows!"
Terminate batch job (Y/N)? Y

C:\demo\depend-on-docker\azureai\windows>stop.bat
azureai
C:\demo\depend-on-docker\azureai\windows>
```

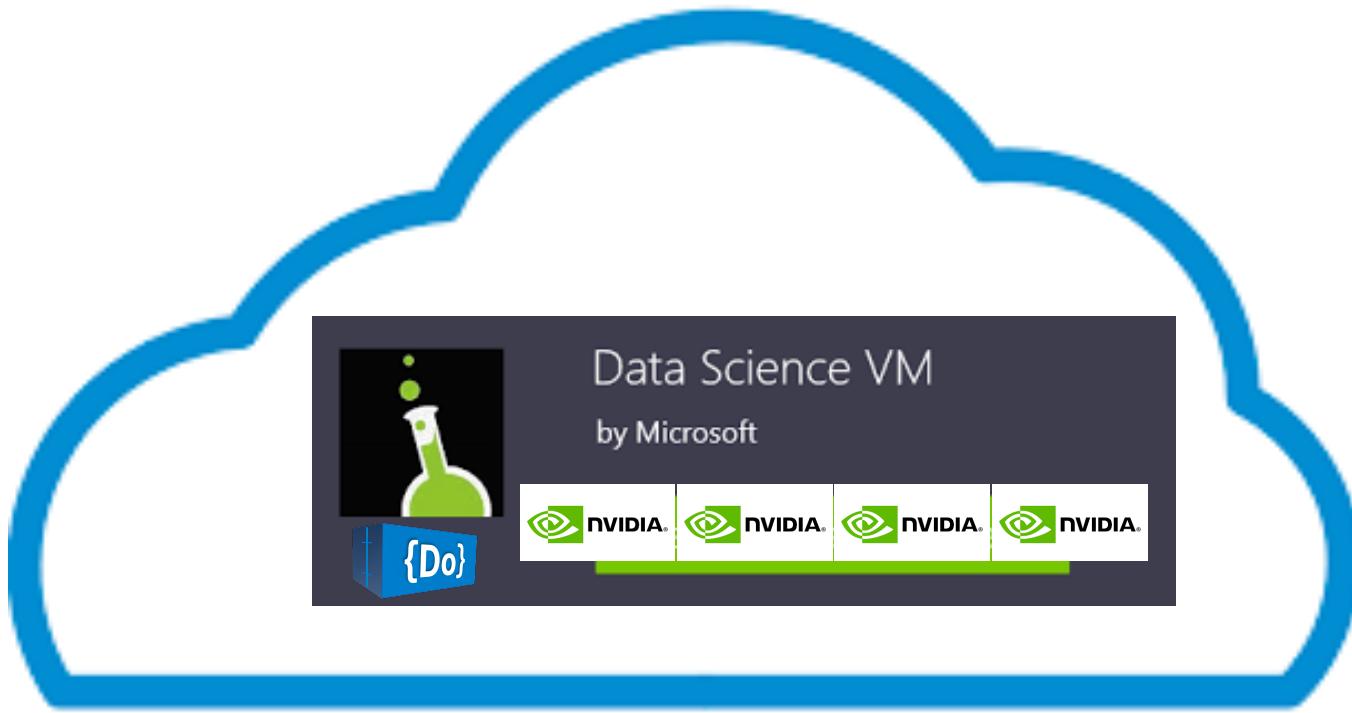
# AI Impacts Oil & Gas End to End

Robust Fault Prediction  
Automated Inspection  
Reservoir Characterization  
Seismic Interpretation  
Automated Drilling  
Production Optimization

**GPU acceleration essential to realize business outcomes @ Scale**



# Demo 2 – Depend on Docker Time Series Model on NVIDIA Tesla V100 GPU



# Demo 2: Step 1 – Show Input Datafile and GPUs



```
alex@data-science-ubuntu:~/demos/dsa-gp$ ls -al
total 80
drwxrwxr-x 4 alex  docker 4096 Dec  5 00:05 .
drwxr-xr-x 5 root  root  4096 Dec  3 18:28 ..
-rw-rw-r-- 1 alex  docker  201 May 21 2018 bashrc.txt
-rwxrwxr-x 1 alex  docker  111 Oct  9 23:07 bash.sh
-rwxrwxr-x 1 alex  docker  183 Oct  9 23:07 build.sh
-rw-rw-r-- 1 alex  docker  868 Oct  9 23:07 Dockerfile
-rwxrwxr-x 1 alex  docker  809 Dec  1 23:40 env.sh
-rw-rw-r-- 1 alex  docker  9682 Oct  9 23:07 gpregression_model_overview
-rwxrwxr-x 1 alex  docker  73 Oct  9 23:07 pull.sh
-rwxrwxr-x 1 alex  docker  73 Oct  9 23:07 push.sh
drwxrwxr-x 2 alex  docker  4096 Dec  1 18:26 scripts
drwxrwxr-x 3 alex  docker  4096 Dec  5 19:49 test
-rwxrwxr-x 1 bmddev  bmddev 1228 Dec  5 00:05 test01Build_GP.sh
-rwxrwxr-x 1 alex  docker  1224 Dec  2 00:19 test01Build.sh
-rwxrwxr-x 1 alex  docker  1243 Dec  2 00:18 test02Predict.sh
-rwxrwxr-x 1 alex  docker  1479 Dec  2 08:21 test03Build.sh
-rwxrwxr-x 1 alex  docker  983 Dec  2 00:16 test04Predict.sh
-rwxrwxr-x 1 alex  docker  1282 Oct  9 23:07 test.sh
alex@data-science-ubuntu:~/demos/dsa-gp$ head test/GP_example1.csv
var1,var2,var3,var4,var5,output
0.956354,0.800940,0.050776,0.315920,0.085017,0.882705
0.736891,0.791851,0.617783,0.493216,0.535094,0.651614
0.336492,0.269555,0.660863,0.406039,0.596710,0.257115
0.424432,0.307552,0.743217,0.884384,0.250924,0.033002
0.763954,0.232128,0.592406,0.824970,0.661324,0.934830
0.118156,0.715689,0.822638,0.520667,0.041701,0.345582
0.377589,0.053209,0.850301,0.658378,0.851081,0.793559
0.340355,0.793597,0.581418,0.284880,0.314973,0.848956
0.369887,0.831202,0.444267,0.584507,0.659407,0.526204
alex@data-science-ubuntu:~/demos/dsa-gp$
```

```
Every 0.1s: nvidia-smi
Wed Dec 5 23:09:23 2018
+-----+
| NVIDIA-SMI 410.72      Driver Version: 410.72      CUDA Version: 10.0 |
| Persistence-MI Bus-Id  Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |
|=====+=====+=====+=====+=====+=====+=====+=====|
|   0  Tesla V100-PCIE... Off | 00006F49:00:0.0 Off |          Off |
| N/A  31C   P0  36W / 250W |    0MiB / 16130MiB |   0%  Default |
+-----+
|   1  Tesla V100-PCIE... Off | 00007ED5:00:0.0 Off |          Off |
| N/A  30C   P0  37W / 250W |    0MiB / 16130MiB |   0%  Default |
+-----+
|   2  Tesla V100-PCIE... Off | 00009265:00:0.0 Off |          Off |
| N/A  32C   P0  36W / 250W |    0MiB / 16130MiB |   0%  Default |
+-----+
|   3  Tesla V100-PCIE... Off | 0000B3B2:00:0.0 Off |          Off |
| N/A  31C   P0  35W / 250W |    0MiB / 16130MiB |   1%  Default |
+-----+
+-----+
| Processes:                               GPU Memory |
| GPU     PID  Type  Process name        Usage        |
|=====+=====+=====+=====+=====+=====|
| No running processes found
+-----+
```

# Demo 2: Step 2 – Build model and see GPU0 load

The screenshot shows two terminal windows side-by-side. The left window displays the command-line output of a Gaussian Process Regression model build, while the right window shows the nvidia-smi command output.

**Left Terminal Output:**

```
alex@data-science-ubuntu:~/demos/dsa-gp$ ./test01Build_GP.sh
Testing GP Regression Build ...
Running on GPU # 0
nvidia-docker run -it --name dsa-gp --rm=true -e CUDA_VISIBLE_DEVICES=0 --privileged -v /demos/dsa-gp/test:/hostfiles aidocker.azurecr.io/dod/dsa-gp:CURRENT-SNAPSHOT python3 /localadm/scripts/GPregression.py /hostfiles/testInputGP.json
(10000, 5)
(10000, 1)
Starting
Centering output
Initializing model
2018-12-05 23:19:26.175848: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2018-12-05 23:19:37.301660: I tensorflow/core/common_runtime/gpu/gpu_device.cc:140] Found device 0 with properties:
name: Tesla V100-PCIE-16GB major: 7 minor: 0 memoryClockRate(GHz): 1.38
pciBusID: 6f49:00:00.0
totalMemory: 15.75GiB freeMemory: 15.34GiB
2018-12-05 23:19:37.301719: I tensorflow/core/common_runtime/gpu/gpu_device.cc:148] Adding visible gpu devices: 0
2018-12-05 23:19:46.810942: I tensorflow/core/common_runtime/gpu/gpu_device.cc:965] Device interconnect StreamExecutor with strength 1 edge matrix:
2018-12-05 23:19:46.810997: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971] 0
2018-12-05 23:19:46.811015: I tensorflow/core/common_runtime/gpu/gpu_device.cc:984] 0: N
2018-12-05 23:19:46.811315: I tensorflow/core/common_runtime/gpu/gpu_device.cc:109] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 14844 MB memory) -> physical GPU (device: 0, name: Tesla V100-PCIE-16GB, pci bus id: 6f49:00:00.0, compute capability: 7.0)
/usr/local/GPFlow/gpflow/densities.py:89: UserWarning: Shape of x must be 2D at computation.
warnings.warn('Shape of x must be 2D at computation.')
Training
2018-12-05 23:20:00.610499: I tensorflow/core/kernels/cuda_solvers.cc:159] Creating CudaSolver handles for stream 0x57e1410
```

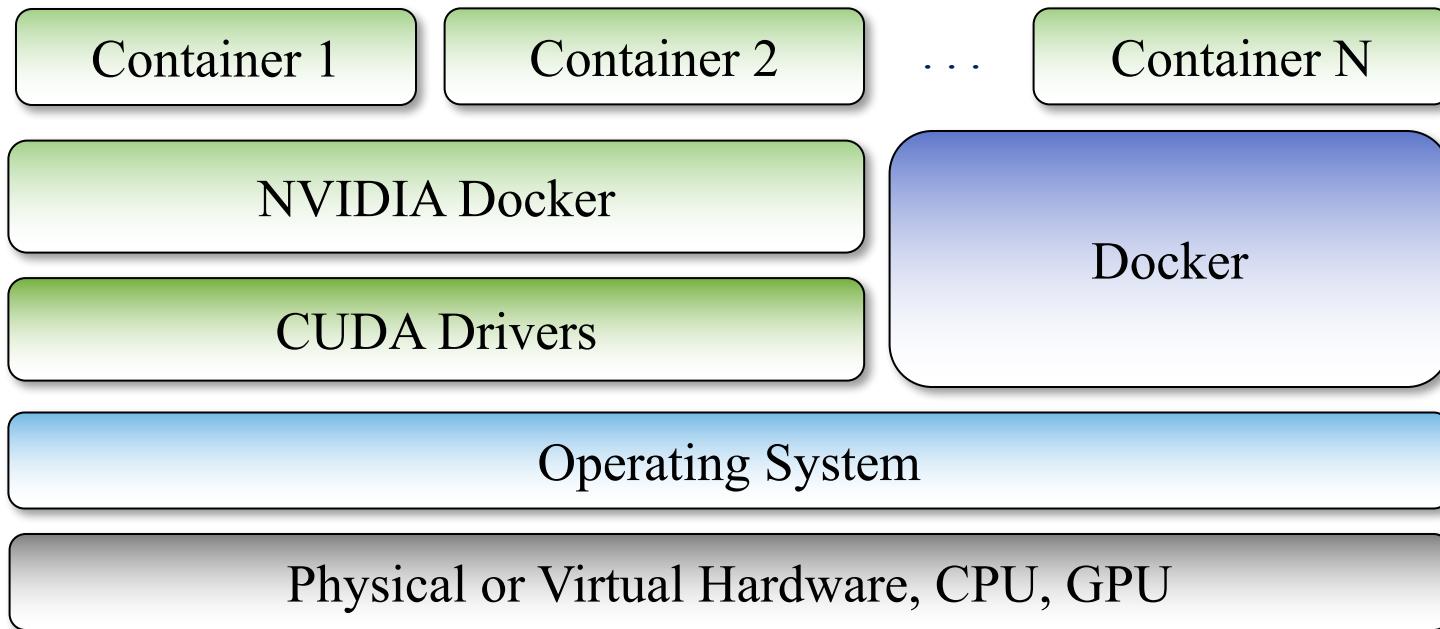
**Right Terminal Output:**

```
alex@data-science-ubuntu:~(ssh) Every 0.1s: nvidia-smi
Wed Dec 5 23:20:20 2018
+-----+
| NVIDIA-SMI 410.72      Driver Version: 410.72      CUDA Version: 10.0 |
| GPU Name Persistence-MI Bus-Id Disp.A | Volatile Uncorr. ECC |
| Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |
+-----+
| 0 Tesla V100-PCIE... Off | 00006f49:00:00.0 Off | Off |
| N/A 44C P0 231W / 250W | 15448MiB / 16130MiB | 100% Default |
+-----+
| 1 Tesla V100-PCIE... Off | 00007ed5:00:00.0 Off | Off |
| N/A 30C P0 37W / 250W | 0MiB / 16130MiB | 0% Default |
+-----+
| 2 Tesla V100-PCIE... Off | 00009265:00:00.0 Off | Off |
| N/A 32C P0 36W / 250W | 0MiB / 16130MiB | 0% Default |
+-----+
| 3 Tesla V100-PCIE... Off | 0000b382:00:00.0 Off | Off |
| N/A 32C P0 35W / 250W | 0MiB / 16130MiB | 5% Default |
+-----+
+-----+
| Processes:                               GPU Memory |
| GPU PID Type Process name               Usage     |
+-----+
| 0 72960 C python3                         15437MiB |
+-----+
```

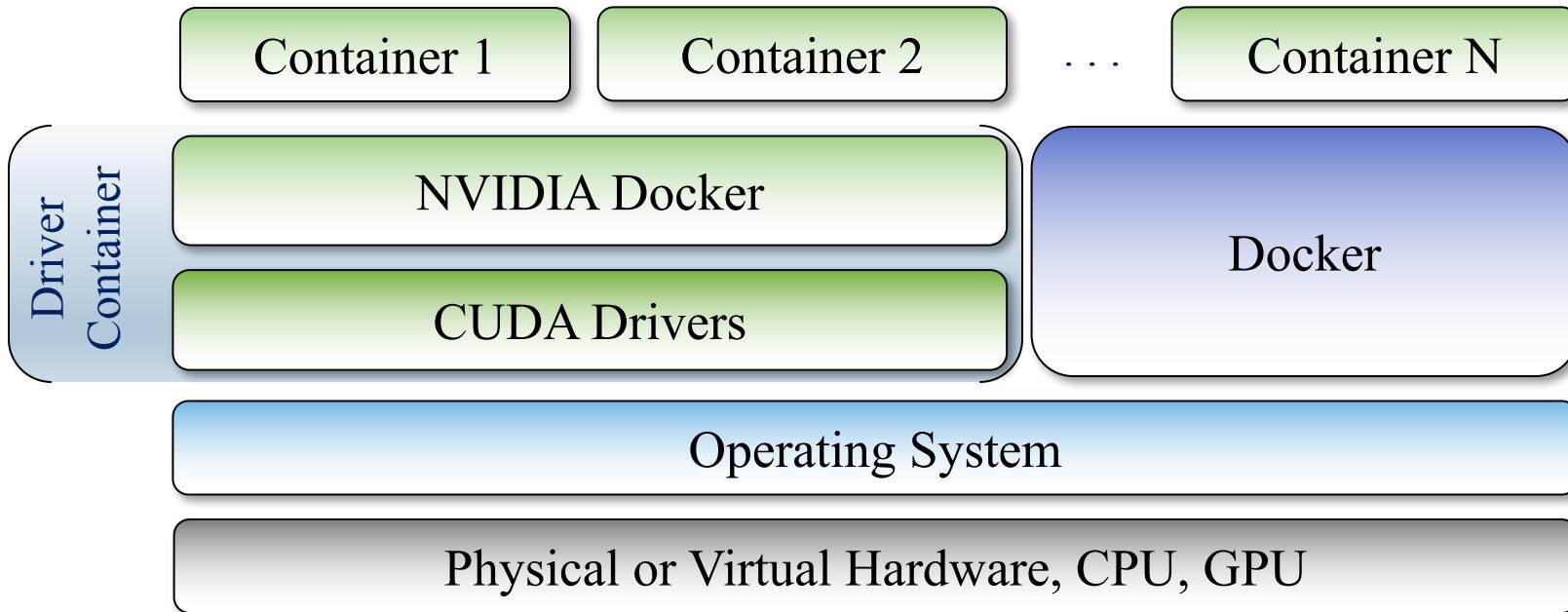
## Demo 2: Step 3 – Show predictions

```
alex@data-science-ubuntu:/demos/dsa-gp$ head test/inputDataPredictions.csv
var1,var2,var3,var4,var5,output,predicted_mean,std,mean - 2*std,mean+2*std
0.956354,0.800940,0.050776,0.315920,0.085017,0.882705,0.892235,0.078100,0.736035,1.048434
0.736891,0.791851,0.617783,0.493216,0.535094,0.651614,0.656932,0.015587,0.625758,0.688107
0.336492,0.269555,0.660863,0.406039,0.596710,0.257115,0.265201,0.026873,0.211455,0.318946
0.424432,0.307552,0.743217,0.884384,0.250924,0.033002,0.041741,0.038647,-0.035552,0.119035
0.763954,0.232128,0.592406,0.824970,0.661324,0.934830,0.936326,0.055599,0.825128,1.047523
0.118156,0.715689,0.822638,0.520667,0.041701,0.345582,0.349199,0.018316,0.312566,0.385831
0.377589,0.053209,0.850301,0.658378,0.851081,0.793559,0.801760,0.092743,0.616274,0.987246
0.340355,0.793597,0.581418,0.284880,0.314973,0.848956,0.854729,0.031081,0.792566,0.916891
0.369887,0.831202,0.444267,0.584507,0.659407,0.526204,0.532105,0.084864,0.362377,0.701833
```

# GPU setup – a challenge at any scale



# GPU setup – a challenge at any scale



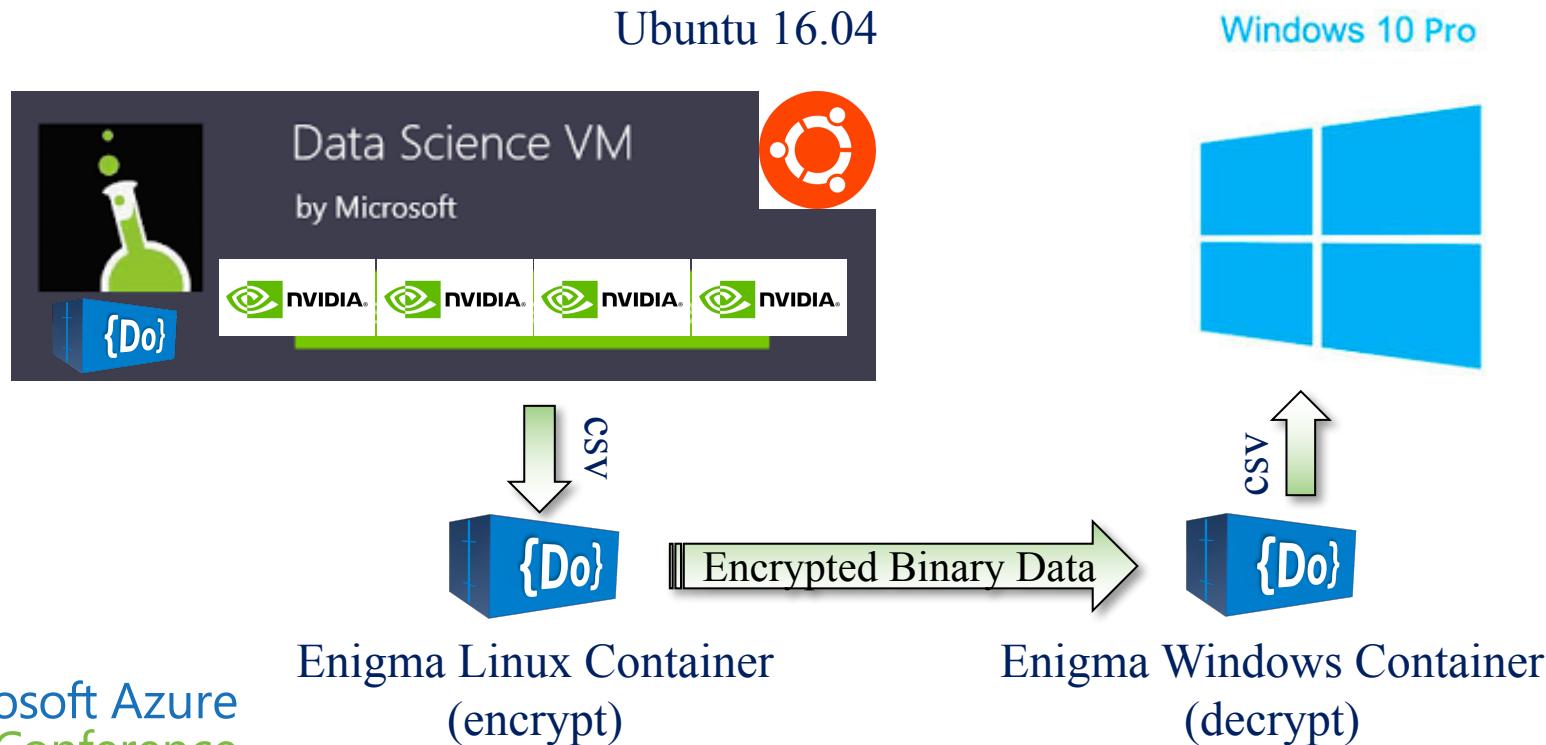
# Depend-on-Docker Example Enigma Project

<https://github.com/bhgedigital/enigma>

```
1. bash
> ls
Container-Root README.md      decrypt.sh    keys          push.sh
Dockerfile      build.sh      encrypt.sh    pull.sh
iankouls@GC02T82RGTFME:~/Projects/2018/github.com/iankoulski/enigma
> ./encrypt.sh Depend-on-Docker
tt90QUp1uU80/4/TRaK+VpPx8vgSEKNHI12IGqKgHnJZ0HandwcFKlaoT02uRv/yAQWhVtaQwRnpv0DWwigG7V1
qL6zmZql6b4cHPpTDRCr0Lb6cTR5oDPeLtAmgZxfW3fUX/JcP+e+40vebYXw1MiwFtLojzV4uthT5SAgz7tZJoq
t0CN4TsD1dfkFMXrJr0VepPVpVP31I0DVwJP0YfQpUSRw6zUVa2t3YTEQGE6BNevAH060ZsUcgJcht7JPjMuRfq
Jt2JLukuBvSrp4A4W/mxj9y5/hHRgBR+r5IGby0kaVpAwnFKCS+BKDZCM3z1CQGKuYn03LE/bAtJeNcaQ==
iankouls@GC02T82RGTFME:~/Proj
>
```

```
1. bash
> ls
Container-Root README.md      decrypt.sh    keys          push.sh
Dockerfile      build.sh      encrypt.sh    pull.sh
iankouls@GC02T82RGTFME:~/Projects/2018/github.com/iankoulski/enigma
> ./decrypt.sh tt90QUp1uU80/4/TRaK+VpPx8vgSEKNHI12IGqKgHnJZ0HandwcFKlaoT02uRv/yAQWhVtaQw
Rnpv0DWwigG7V1qL6zmZql6b4cHPpTDRCr0Lb6cTR5oDPeLtAmgZxfW3fUX/JcP+e+40vebYXw1MiwFtLojzV4ut
hT5SAgz7tZJoq t0CN4TsD1dfkFMXrJr0VepPVpVP31I0DVwJP0YfQpUSRw6zUVa2t3YTEQGE6BNevAH060ZsUcgJ
cht7JPjMuRfqJt2JLukuBvSrp4A4W/mxj9y5/hHRgBR+r5IGby0kaVpAwnFKCS+BKDZCM3z1CQGKuYn03LE/bAtJeNcaQ==
Depend-on-Docker
iankouls@GC02T82RGTFME:~/Projects/2018/github.com/iankoulski/enigma
>
```

# Demo 3 – Data Security with Depend on Docker Enigma Container



# Demo 3: Step 1 – Encrypt predictions on Linux

Linux enigma image →

encrypt predictions csv file →

First few lines of encrypted content →

A terminal window titled "Alex I" showing the command-line steps to encrypt a CSV file. The commands include:

```
alex@data-science-ubuntu:/demos/dsa-gp/test$ echo ${REGISTRY}${IMAGE}${TAG}
bgdedigital/enigma:latest
alex@data-science-ubuntu:/demos/dsa-gp/test$ cat encrypt.sh
#!/bin/bash

source .env

if [ -z "$1" ]; then
    encryptUsage
    exit 0
fi

docker container run ${RUN_OPTS} ${MODE} ${VOL_MAP} --rm ${REGISTRY}${IMAGE}${TAG} encrypt -i /wd/$1 -o /wd/$1.enc
alex@data-science-ubuntu:/demos/dsa-gp/test$ ./encrypt.sh inputDataPredictions.csv
alex@data-science-ubuntu:/demos/dsa-gp/test$ head inputDataPredictions.csv.enc
```

The terminal then displays the first few lines of the encrypted binary data, which appear as a series of random characters.

copy encrypted file to shared folder →

```
alex@data-science-ubuntu:/demos/dsa-gp/test$ cp -vf inputDataPredictions.csv.enc /z/encrypted/
'inputDataPredictions.csv.enc' -> '/z/encrypted/inputDataPredictions.csv.enc'
alex@data-science-ubuntu:/demos/dsa-gp/test$
```

# Demo 3: Step 2 – Decrypt predictions on Windows

copy encrypted file from  
shared folder →

decrypt using enigma  
windows container →

encrypted content →

decrypted content →

```
Administrator: Command Prompt

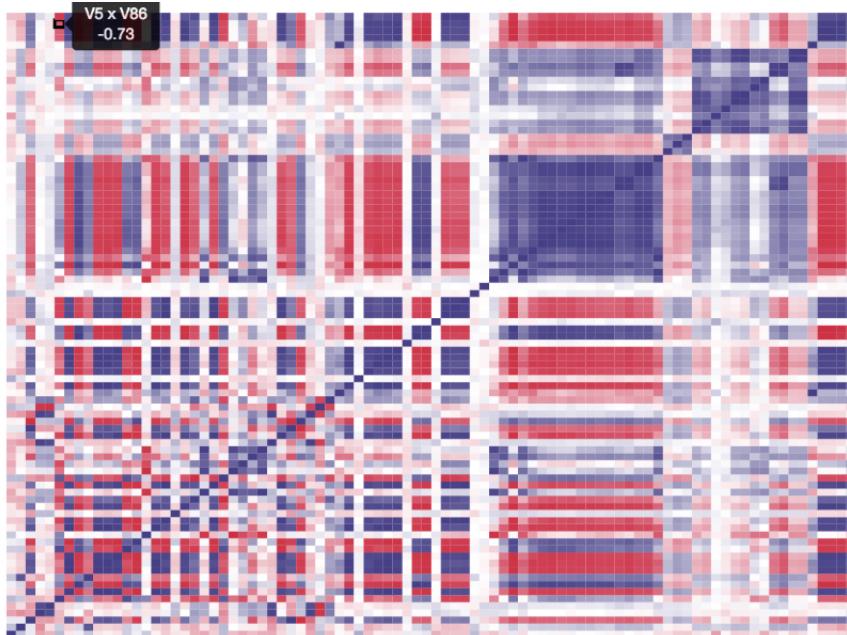
C:\demo\enigma\Windows>copy Z:\encrypted\inputDataPredictions.csv.enc .
      1 file(s) copied.

C:\demo\enigma\Windows>decrypt-predictions.bat inputDataPredictions.csv.enc
docker container run -e http_proxy= -e https_proxy= -e no_proxy=localhost --name enigma -it -p 80:8080 -v c:/keys:c:/keys -v C:\demo\enigma\Windows:c:/wd --rm bhgedigital/enigma:windows powershell -Command "decrypt-file c:\wd\inputDataPredictions.csv.enc c:\wd\inputDataPredictions.csv.decrypted"

C:\demo\enigma\Windows>head -n 5 inputDataPredictions.csv.enc
ok+TCd*8Y@ očo oZomo8s0ooAs0ooop>B!oGooT>ooooo0oo)3oWko:noch-o1o7ko@ooo7@polYr:oYo~oKoohoJoo5oo&ovoT@[yIooo7ooconoKooo={3xooNdhAo
(8ooooooooooooW%
oxo(o0oo9op,;o0oL>oooIoRI>doo}oonrofsoPoo+ooo9oo^oB1?oato0at
oooPoo0@ooooo,ko{oojzovoooA o ~ooio0ovogpoo+    @roor*o solooo26HofTi=o@o          oNZ6ooooxoa@NoXo$o          oo`>or;ho=Ao*&Taa2      oo]S@o
o!oo
&ooooXo>oo^oo-@oo8Co@ooYooak>o|oo=oo/oooo=ooooooMoooo@ooooG@oo)ooooooKt%ooooUoQ)oIVic0o@byyyo$5Lcoo!-o@o@oooooZ 'S@o@J9oooox>Booo$@Fjoo
ooLo@Z@oo

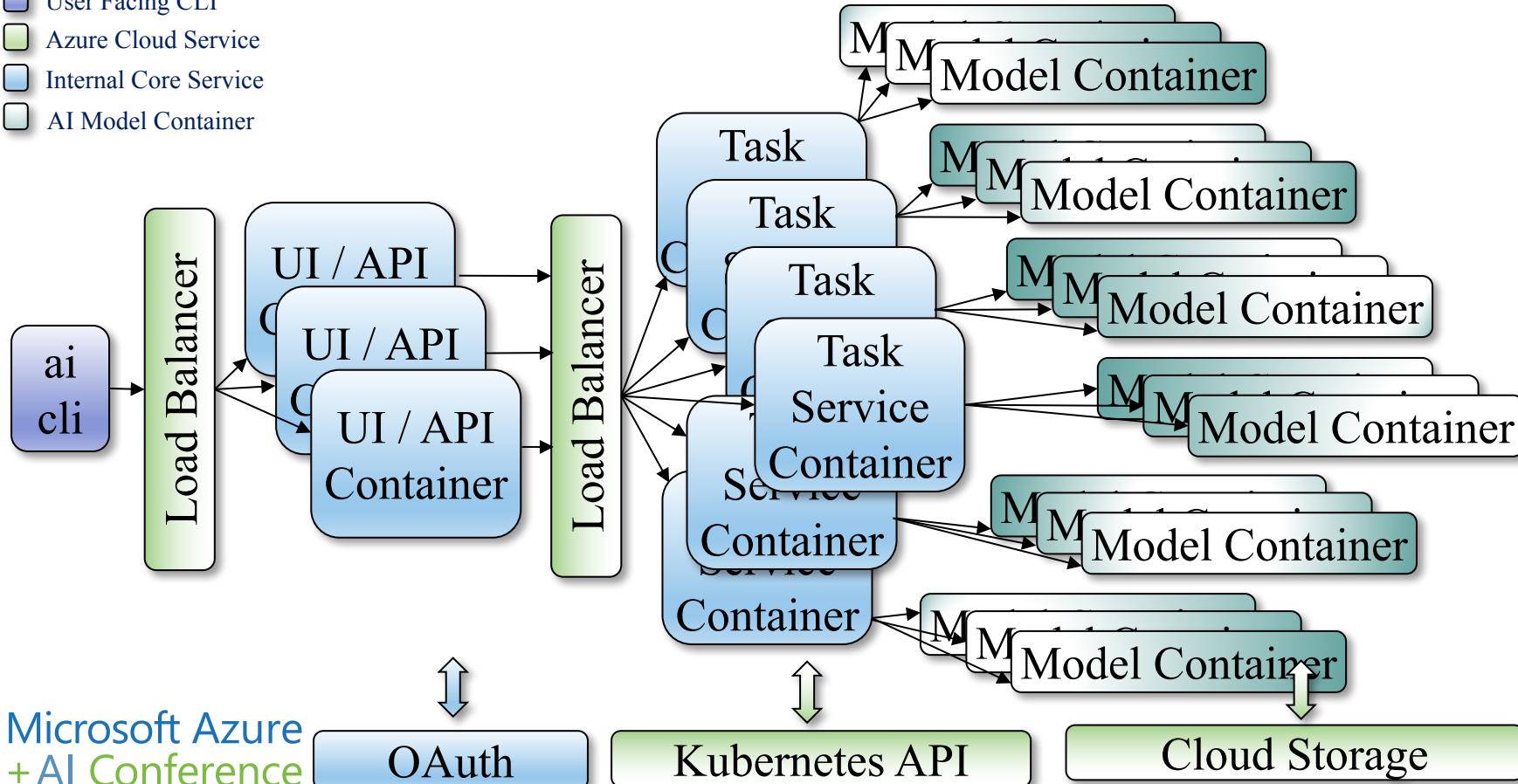
C:\demo\enigma\Windows>head inputDataPredictions.csv.decrypted
var1,var2,var3,var4,var5,output,predicted_mean,mean - 2*std,mean+2*std
0.956354,0.800940,0.050776,0.315920,0.085017,0.882705,0.892235,0.078100,0.736035,1.048434
0.736891,0.791851,0.617783,0.493216,0.535094,0.651614,0.656932,0.015587,0.625758,0.688107
0.336492,0.269555,0.660863,0.406039,0.596710,0.257115,0.265201,0.026873,0.211455,0.318946
0.424432,0.307552,0.743217,0.884384,0.250924,0.033002,0.041741,0.038647,-0.035552,0.119035
0.763954,0.232128,0.592406,0.824970,0.661324,0.934830,0.936326,0.055599,0.825128,1.047523
0.118156,0.715689,0.822638,0.520667,0.041701,0.345582,0.349199,0.018316,0.312566,0.385831
0.377589,0.053209,0.850301,0.658378,0.851081,0.793559,0.801760,0.092743,0.616274,0.987246
0.340355,0.793597,0.581418,0.284880,0.314973,0.848956,0.854729,0.031081,0.792566,0.916891
0.369887,0.831202,0.444267,0.584507,0.659407,0.526204,0.532105,0.084864,0.362377,0.701833
```

# Guided AI: Automatically build models



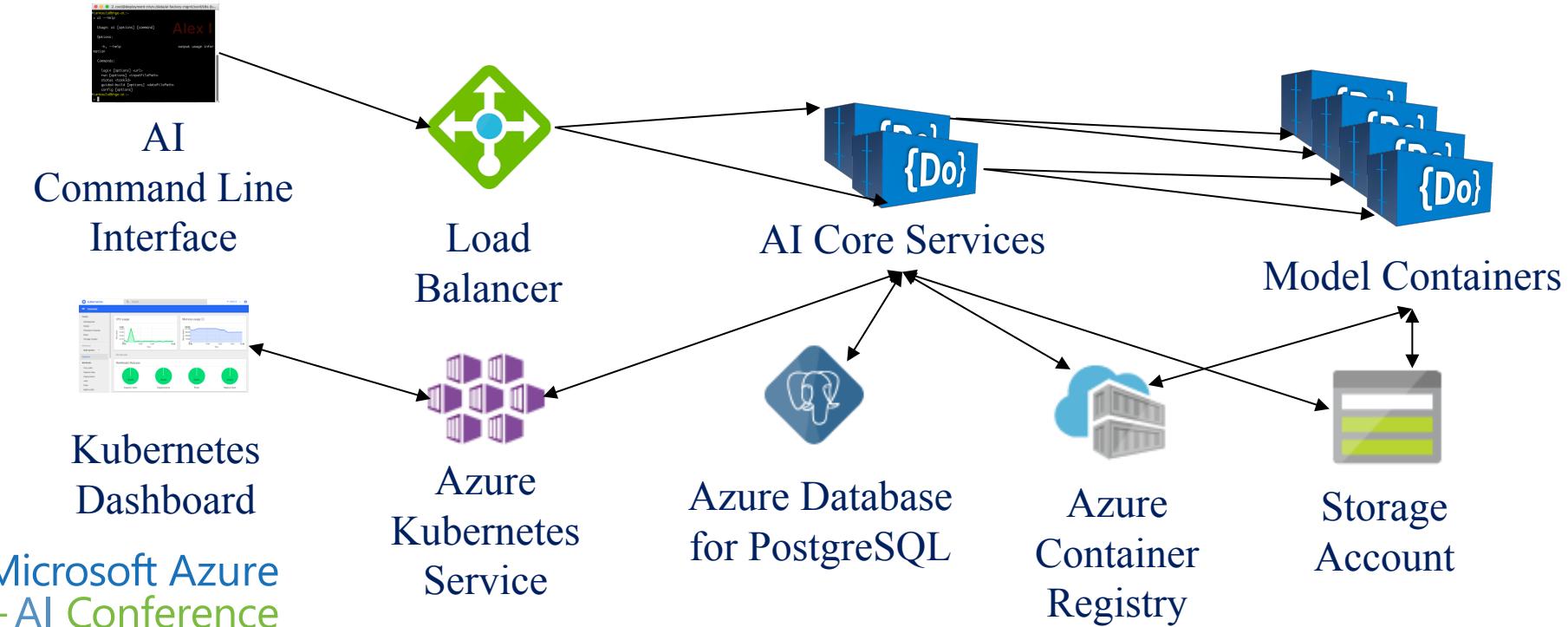
# Guided AI - Architecture

- User Facing CLI
- Azure Cloud Service
- Internal Core Service
- AI Model Container



# Demo 4 – Guided AI

## Scale up on Azure



# Demo 4: Step 1 – Logon to Kubernetes Dashboard

This customized version of the Kubernetes Dashboard has a Graph tab which allows us to visualize the workloads. The Task service is in the upper right part of the graph and is scaled to 20 pods.

The screenshot shows the Kubernetes Dashboard interface. The left sidebar has a tree view: Cluster, Namespaces, Nodes, Persistent Volumes, Roles, Storage Classes, Namespace (with 'default' selected), Overview, Workloads (Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Graph, Replica Sets, Replication Controllers, Stateful Sets), Discovery and Load Balancing (Ingresses, Services), Config and Storage (Config Maps, Persistent Volume Claims, Secrets). The 'Graph' tab is currently active. The main content area has a 'Pods Status' table:

Running	Pending	Total
58	0	58

Below the table is a large network graph with many blue circular nodes and connecting lines. A cluster of nodes in the upper right is highlighted and labeled 'Task'. The entire dashboard has a light blue background.

# Demo 4: Step 2 – Submit dataset using AI CLI

```
alex@dod-ubuntu1804:/demo/ai-build$ ai guided-build -i 10 -e ./guided_build_1000.csv
{"uploaded": [{"filename": "guided_build_1000.csv", "id": "8f700b0e201542819c79c12c053bd9a2"}]}
```

Created: 0	Pending: 6000	Total: 6000
------------	---------------	-------------

Task f69d2c0e-a469-4020-bc18-ff72e6498cc0 created (20009 ms).

Created: 1	Pending: 5999	Total: 6000
------------	---------------	-------------

Task 06d7bf3a-1376-4844-ad11-ff95e999910f created (20043 ms).

Created: 2	Pending: 5998	Total: 6000
------------	---------------	-------------

Task ff560320-6888-4c52-805c-787be0e15824 created (20156 ms).

Created: 3	Pending: 5997	Total: 6000
------------	---------------	-------------

..  
Task 9bc79495-8e59-4eee-89b6-b89f5652aae5 created (34305 ms).

Created: 5996	Pending: 4	Total: 6000
---------------	------------	-------------

Task 597c1b32-665e-4d94-a7ef-7666b5f7b3f6 created (26186 ms).

Created: 5997	Pending: 3	Total: 6000
---------------	------------	-------------

Task 4494d20b-9088-4fba-9a16-712f681e6b5b created (39505 ms).

Created: 5998	Pending: 2	Total: 6000
---------------	------------	-------------

Task 61b7dc60-3378-43cf-ac82-4b51673202c9 created (22940 ms).

Created: 5999	Pending: 1	Total: 6000
---------------	------------	-------------

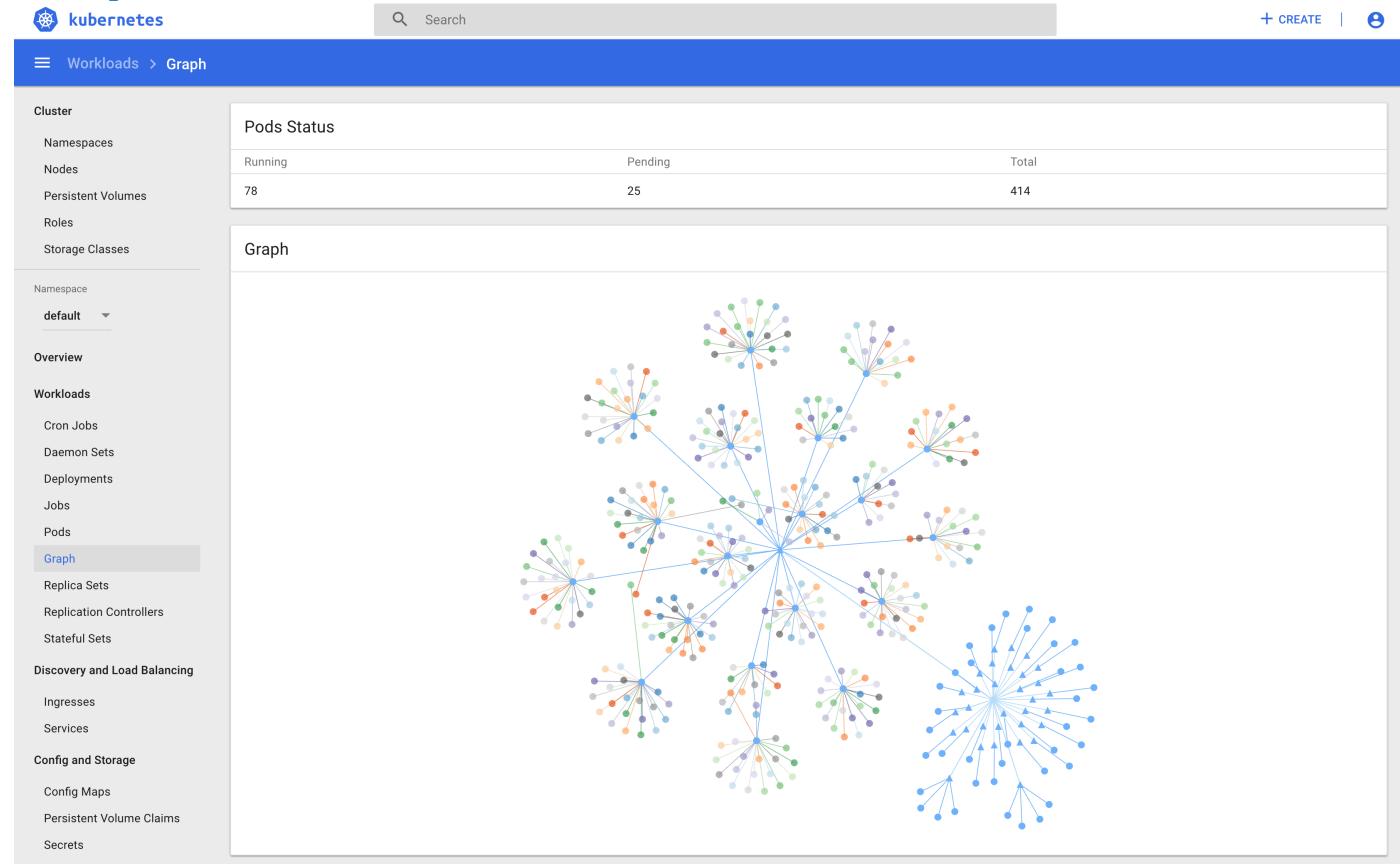
Task ad111432-fb88-40d1-87f0-50808986920c created (19931 ms).

Created: 6000	Pending: 0	Total: 6000
---------------	------------	-------------

```
alex@dod-ubuntu1804:/demo/ai-build$ 
```

# Demo 4: Step 3 – Observe new model containers

Each colorful dot is a model pod. Model pods are connected to their parent task service pod, which makes this workload visualization look like a fireworks explosion.

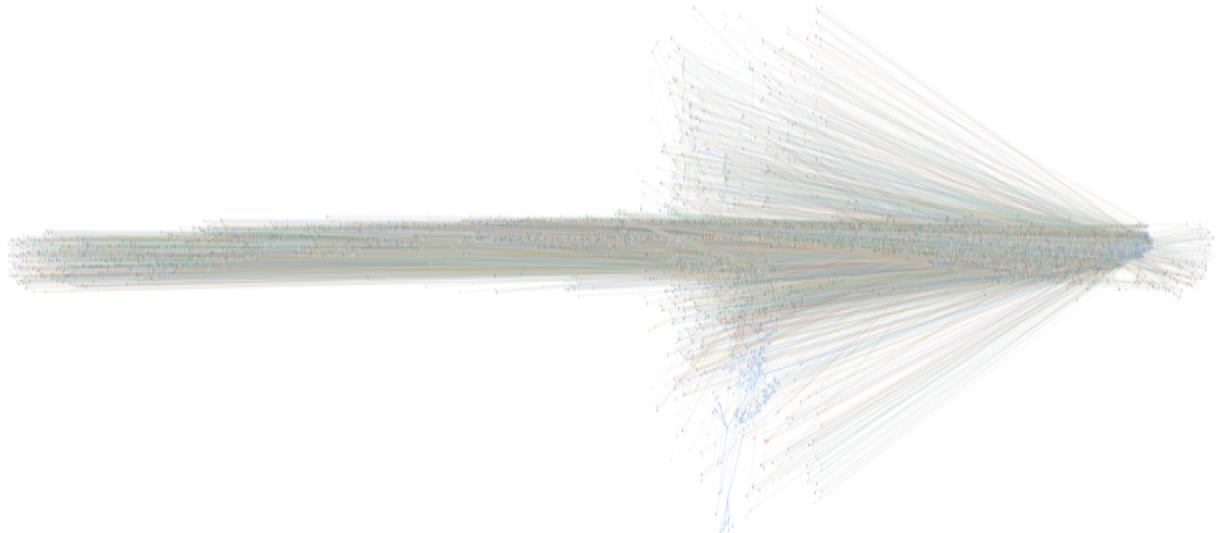


# 30,000 Models built concurrently

50 Nodes

4800 vCPUs

30 TB RAM



Microsoft Azure  
+AI Conference

CO-PRODUCED BY  
Microsoft & DEVintersection

© BHGE. All rights reserved

# Places where you will find containers

- CI / CD Systems – Use Docker in Docker to build and ship container images
- Software Test execution – Create environment, execute test, remove container
- Development – IDE in a container, Cloud IDE, super easy developer onboarding
- CLI – e.g. Azure cli in Docker, SQL client in Docker
- Diagnostics, Troubleshooting, Support – Run Docker to collect diagnostic info
- Patching - Run Docker command to fix and patch installed software
- Backups – Run your backup job in a container
- Firmware updates via containers!
- Initialization scripts – run privileged to install software or do admin work on host

# Depend on Docker Summary

- It is easy to build Windows and Linux containers
- Just run, instead of “install first, and then run”
- You choose whether to run your workload on CPU or GPU
- Docker enables us to scale workloads up and down at will

Docker makes IT possible



# References

- GE Aircraft Engines  
<https://www.ge.com/reports/every-two-seconds-aircraft-powered-ge-technology-takes-off-somewhere-world/>
- GE Power Generation  
<https://www.ge.com/reports/ultra-super-critical-badass-machines-help-make-30-percent-worlds-power/>
- Container Hacks and Fun Images, Jessie Frazelle <https://www.youtube.com/watch?v=cYsVvV1aVss>
- GE SemTK Project - Paul Cuddihy, Jenny Weisenberg, Justin McHugh, et.al. <https://github.com/ge-semtk/semtk>
- Enigma DoD Project – Yekta Yazdani <https://github.com/bhgedigital/enigma>
- Google Images <https://images.google.com/>
- Depend-on-Docker Project <https://github.com/bhgedigital/depend-on-docker>
- Depend on Docker DockerCon'18: <https://embed.vidyard.com/share/5hUnYnLyxkJGSuK89woNn>
- NVIDIA Driver Containers: [https://github.com/NVIDIA/nvidia-docker/wiki/Driver-containers-\(EXPERIMENTAL\)](https://github.com/NVIDIA/nvidia-docker/wiki/Driver-containers-(EXPERIMENTAL))
- Microsoft Azure: <https://azure.microsoft.com/en-us/>

# Questions?

*Please use EventsXD to fill out a session evaluation.*

Thank you!

