Delivery Excellence, DevOps:

Cloud-native Deployments of Data Science Models

Düsseldorf, Nov 14, 2018, Daniel Schulz

Capgemini

Two Model Servers are Available to Deploy ML Models to Production – in Traditional & Cloud Environments

TensorFlow Serving (TFS)

- TFS’ Docker Images from Docker Hub
  hub.docker.com/r/tensorflow/serving &
- Code on GitHub
  github.com/tensorflow/serving

MXNet Model Server from AWS’ DeepLearningTeam

- AWS’ Docker Images from Docker Hub
  hub.docker.com/r/awsdeeplearningteam/mms_cpu &
- Code on GitHub
  github.com/deep-learning-mms-bot/mxnet-model-server

An Almost Clean Slate

There Will be no Prior Deployments

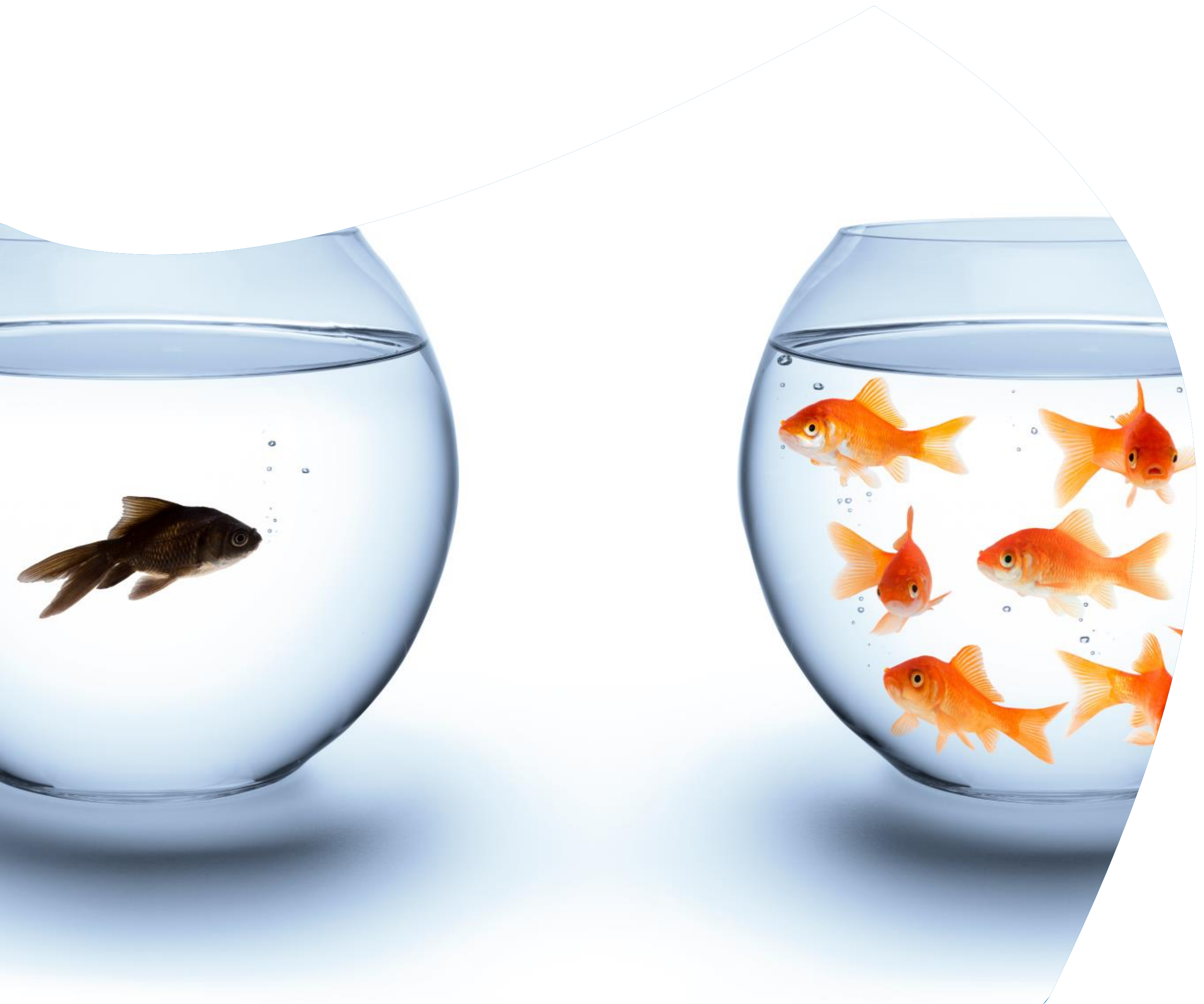
Building a Docker-based Microservice Architecture

- Software Isolation:
  due to Docker Containers, no conflicting libraries keep us out of Dependency Hell

- Infrastructure Isolation:
  conflicting ports, paths, environment variables, etc. cause no issues to the former

- Ease of Use:
  any solution is a bespoke setup and no monolith – this is a mixed bag on API consistency and architectural beauty, but support Agile projects' fail fast attitude

Isolation Simplifies Things



In traditional deployment models, any worker node runs several OS processes, which may interfere with one another. This is the crowded bowl on the right-hand side.

In isolated Docker Containers, any process is (almost) isolated from most other things. Its just got what it needs to interact with and shares only customized runtime dependencies provided:

- the OS Kernel
- mapped folders using (Persistent) Volumes
- network interfaces and ports

Docker Containers in its purest form are depicted on the left-hand side. This lonely fish cannot see its peers (processes) running on the same Worker Node in different Containers/Namespaces.

Docker in a Nutshell – 3 Kinds of Description Makes a Deployment

Data – Mapped Volumes

```
docker run […] \
    -v /host/path:/container/path:ro […] \
    tensorflow/serving
Pattern: -v
${HOSTPATH}:${CONTAINERPATH}:${PRIVILEGES}
```

At runtime, a Docker Container may override an Image path by using the Overlay File System. This path from the host may have privileges [`ro`, `rw`, `z`]

ReST API – Port Mapping

```
docker run […] -p 2008:8501 […] \
    tensorflow/serving
Pattern: -p ${HOSTPORT}:${CONTAINERPORT}
```

Each Container-internal port must be mapped to an outside port on the host's network interface.
Here the port 8501 stems from the TensorFlow Docker Image and cannot be altered at runtime – but mapped to 2008.

Residual Configuration

Additionally, Docker Containers get configured at runtime using e.g.:

- Environment variables for the Container (`-e KEY="VALUE"`),
- Resource limitations for `CPU`, `RAM` usage and Swapping, etc. and
- A custom process or `Entrypoint` script to run right away

Deployment Descriptor is a Modern Low Code Approach

This plain-Docker-setup will work analogous in more complex Docker-based Cloud environments like e.g.

- Docker w/ Docker Swarm,
- OpenShift,
- Mesos / DC/OS and
- Microsoft Azure's AKS

We’ll Use…

Mapped (Persistent) Volumes

To map our bespoke Machine Learning models into “our” Docker Containers

Port Mapping

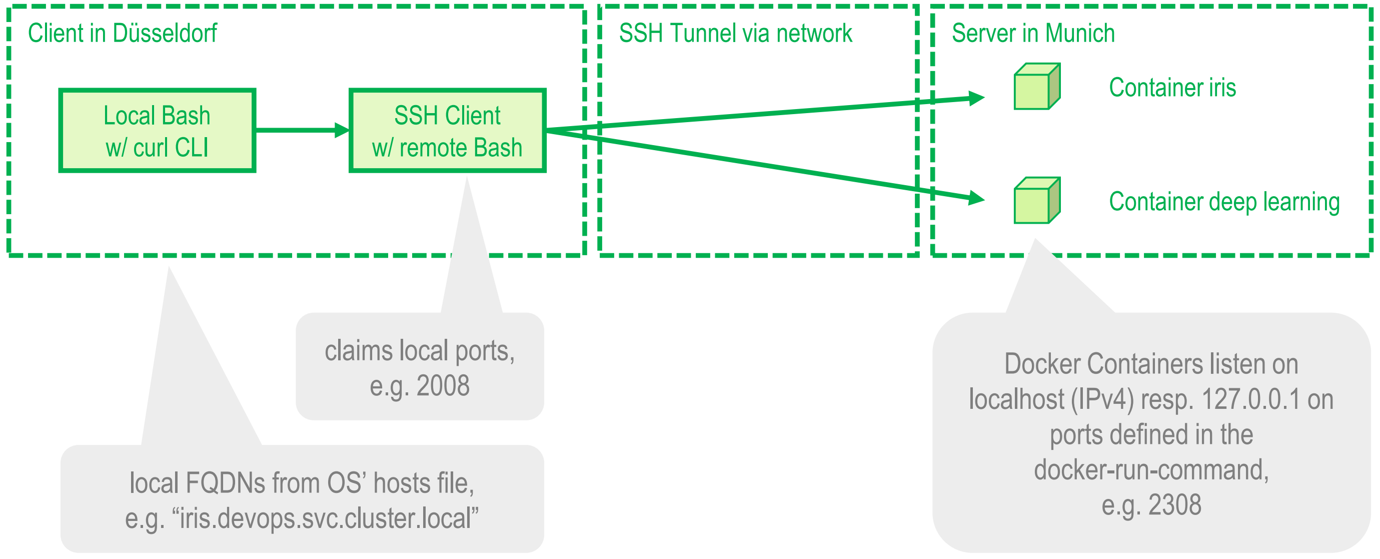
To map the predefined ports from the Docker Image to our respective instances of “our” Docker Containers

Building Blocks

Intended Content of this Proposed Technology Practice

Architecture & Deep Infrastructure

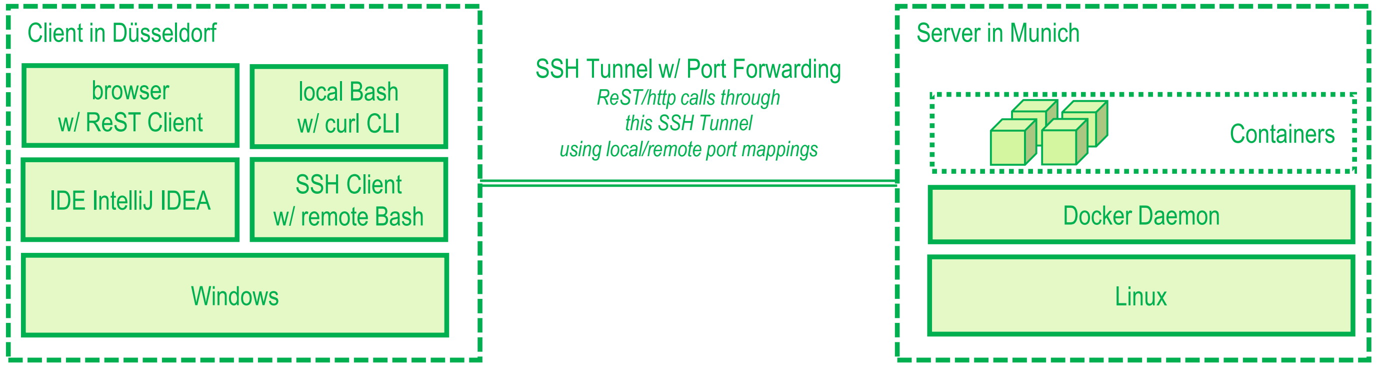
Client-Server-Architecture at its Best: My Client in Düsseldorf, our Data Center in Munich


Client in Düsseldorf
Local Bash w/ curl CLI
SSH Client w/ remote Bash
SSH Tunnel via network
Server in Munich
Container iris
Container deep learning
claims local ports, e.g. 2008
local FQDNs from OS' hosts file, e.g. "iris.devops.svc.cluster.local"
Docker Containers listen on localhost (IPv4) resp. 127.0.0.1 on ports defined in the docker-run-command, e.g. 2308

SSH stands in as a "poor man's Kubernetes." The upside is, this setup works in modern Cloud solutions like OpenShift right away.

Architecture & Deep Infrastructure

Local Development on Windows, Remote Execution on a Linux-based Docker Server

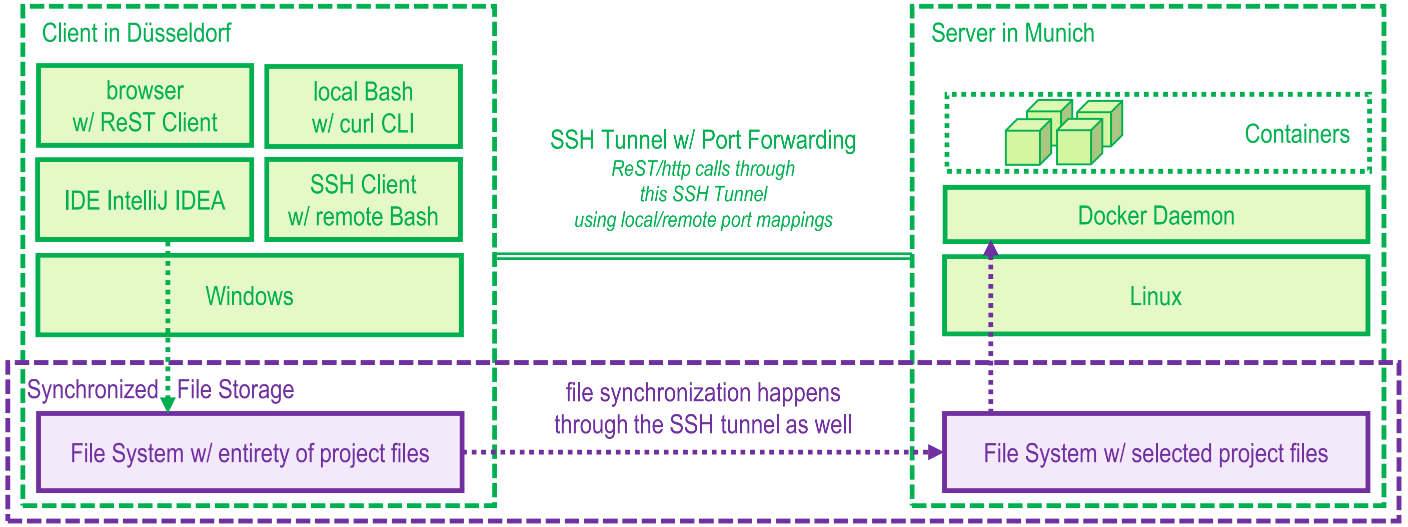


The client-side development in Windows gets executed remotely through an SSH tunnel. The technical reason is, Windows is not fully compatible with modern Docker solutions. The latter is fully based on modern Linux Kernels to such an extend, not even Mac OS *(due to the early fork of the Darwin Kernel from BSD)* is capable of running Docker *(wo/ a Hypervisor or VM in between)*.

For this setup, it is technically equivalent whether to use a remote Linux through a Hypervisor, Virtual Machine or another Linux server. The former two may reside on the physical client machine or anywhere else.

Architecture & Deep Infrastructure

Development Happens in my Local IDE, which gets Synced to the Remote Docker Server Automatically


Client in Düsseldorf
browser w/ ReST Client
local Bash w/ curl CLI
IDE IntelliJ IDEA
SSH Client w/ remote Bash
Windows
SSH Tunnel w/ Port Forwarding
ReST/http calls through this SSH Tunnel using local/remote port mappings
Server in Munich
Containers
Docker Daemon
Linux
Synchronized File Storage
File System w/ entirety of project files
file synchronization happens through the SSH tunnel as well
File System w/ selected project files

A manual synchronization between client and server is “only” less convenient. The fully automated way and manual transferal are technically equivalent and both will work.

Windows and Linux-based Operating Systems store files differently. When working across OS borders, I always make sure to use the exact same Line Endings on both sides. For me, `LF` (Linux’ `Line Feeds`) work best for both Linux systems and git repositories. I tend to use `LF` Line Endings all over the place when the target system is Linux-based – e.g. Docker.

Unforeseeable and hardly to solve problems arise when `LF` (Linux) and `CRLF` (Windows) get mixed up. This problem is known as Newline or `EOL` (End of Line), rather dated and is here to stay. :-/

Working Across Borders – Docker Development on Windows for Linux Systems

IDE Integration

For the Windows-based Docker development with remote Docker execution in a Hypervisor, Virtual Machine or a remote Linux server, several IDE addons may be used.

Docker plugins by IDE:

Eclipse & Neon

- Docker Tooling Support
- Eclipse Docker Tooling for Neon

IntelliJ IDEA

- Docker integration
- Docker

Oracle's NetBeans

- Docker
- Docker Hub in NetBeans IDE

The available plugin for my favorite IDE(A) was working with flaws only. So I got back to the Manual Bash Command approach.

Manual Bash Commands

For year now, I tend work the manual way by issuing Docker/Bash commands. It turned out to be helpful to script the most often used chains and Copy/Paste almost everything then. Therefore the following references shall be fixed:

- Docker Image & Tag names –
  the younger build will replace the existing Image/Tag
- Docker Container names –
  might clash in case of duplicates at same time

This way is more elaborate but highly portable because it uses industry's *de facto* standards in & out.

Port Management –
A ReSTful API for the Iris Model on TensorFlow

From the outside available network ports on the host map to the Docker Container's internal ports:

Docker Image:

- name: `tensorflow/serving`
- tag: `latest`

Host – defined by requirements/developer for usage:

- FQDN: `iris.devops.svc.cluster.local`
- Port: `2008`

Docker Container – pre-defined by provider of Docker Image:

- Port: `8501`
- URI: `/v1/models/iris`

Port Management –
A ReSTful API for the Iris Model on MXNet

From the outside available network ports on the host map to the Docker Container’s internal ports:

Docker Image:

- name: `awsdeeplearningteam/mms_cpu mxnet-model-server`
- tag: `latest`

Host – defined by requirements/developer for usage:

- FQDN: `deeplearning.devops.svc.cluster.local`
- Port: `2308`

Docker Container – pre-defined by provider of Docker Image:

- Port: `8080`
- URI: `/squeezenet/predict`

Finding Sources File in the GitHub Repository
github.com/danielschulz/DevOps_AiToProduction_DockerizedLeanApproach

TensorFlow / Iris, JSON-based Structured Learning

Source:

`code/1-uc-structured-data-json`

MXNet / Deep Learning, Image Classification

Source:

`code/2-uc-deeplearning-image`

Two TensorFlow Serving (TFS) APIs – on Solution Architecture

C++ API

Foremost applicable for

- Existing software projects in `C++` or
- Architectural monoliths capable for integrating `C++` modules, like `C++`, `Java`, `Scala`, etc.

The transport overhead might be minimalized due to native integration – foremost in `C++` projects. In other solutions, the integration way is the decisive factor.

ReST API

Foremost applicable for

- Microservice Architectures and
- Web projects

The transport overhead is slightly higher due to `http(s)` transmission and the serialization before. On the other hand, TensorFlow Serving is a highly optimized web service with low latency and no need to restart the server on model changes.

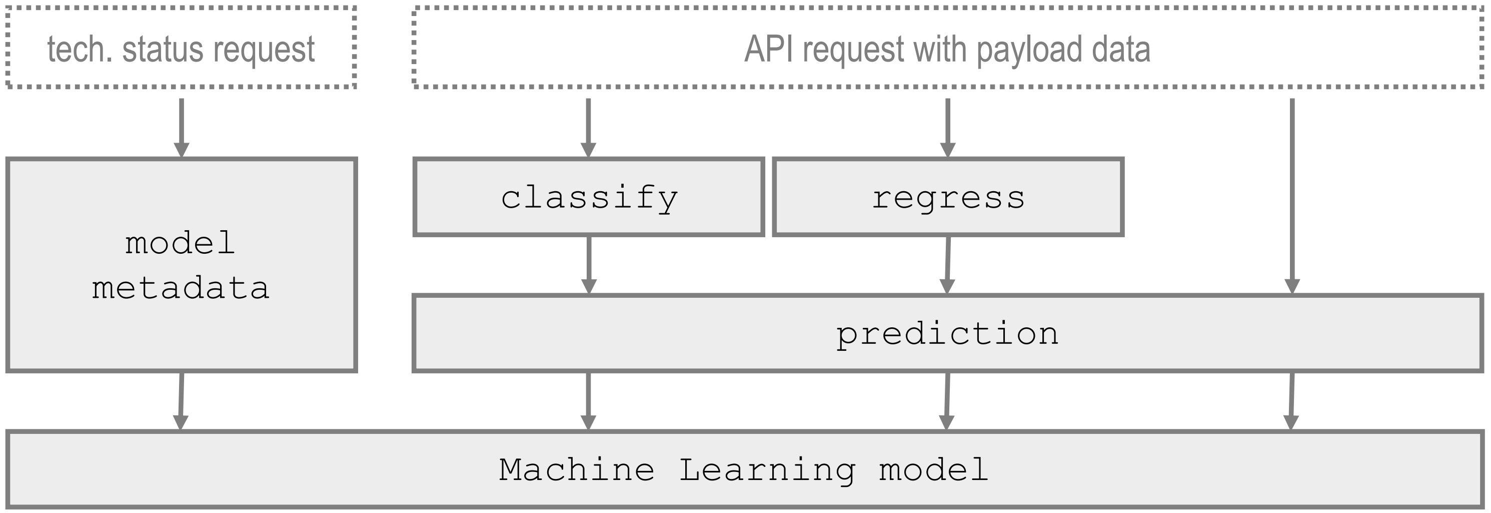
Two TensorFlow Serving (TFS) APIs – on TFS Services resp. Resource Endpoints

Classify or Regress URI

High-level API building up on the prediction API

Prediction URI

The foundational API to perform classifications and regressions based on the Machine Learning model provided



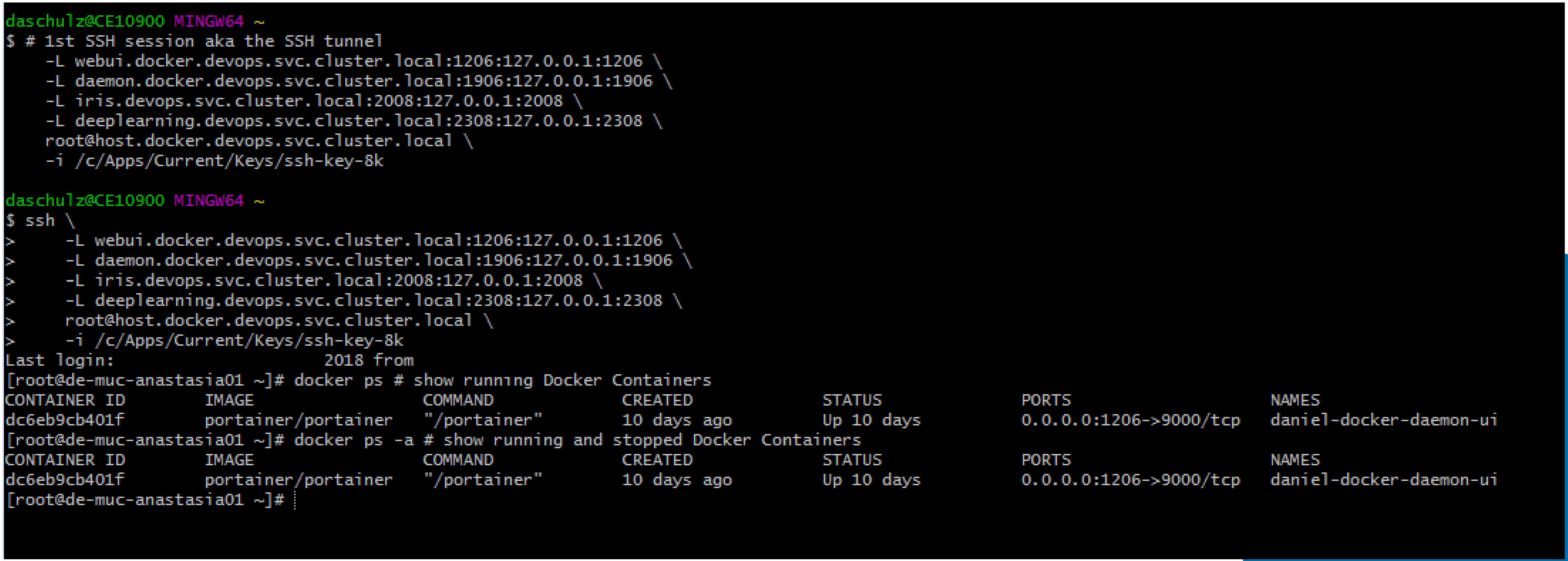
Let’s Get Ready to ~~Play~~ Deploy Our Containers

“We’re gonna set up and start to play” (SaySaySay, Beatsteaks)

Establish the SSH Tunnel to the Remote Linux/Docker Server

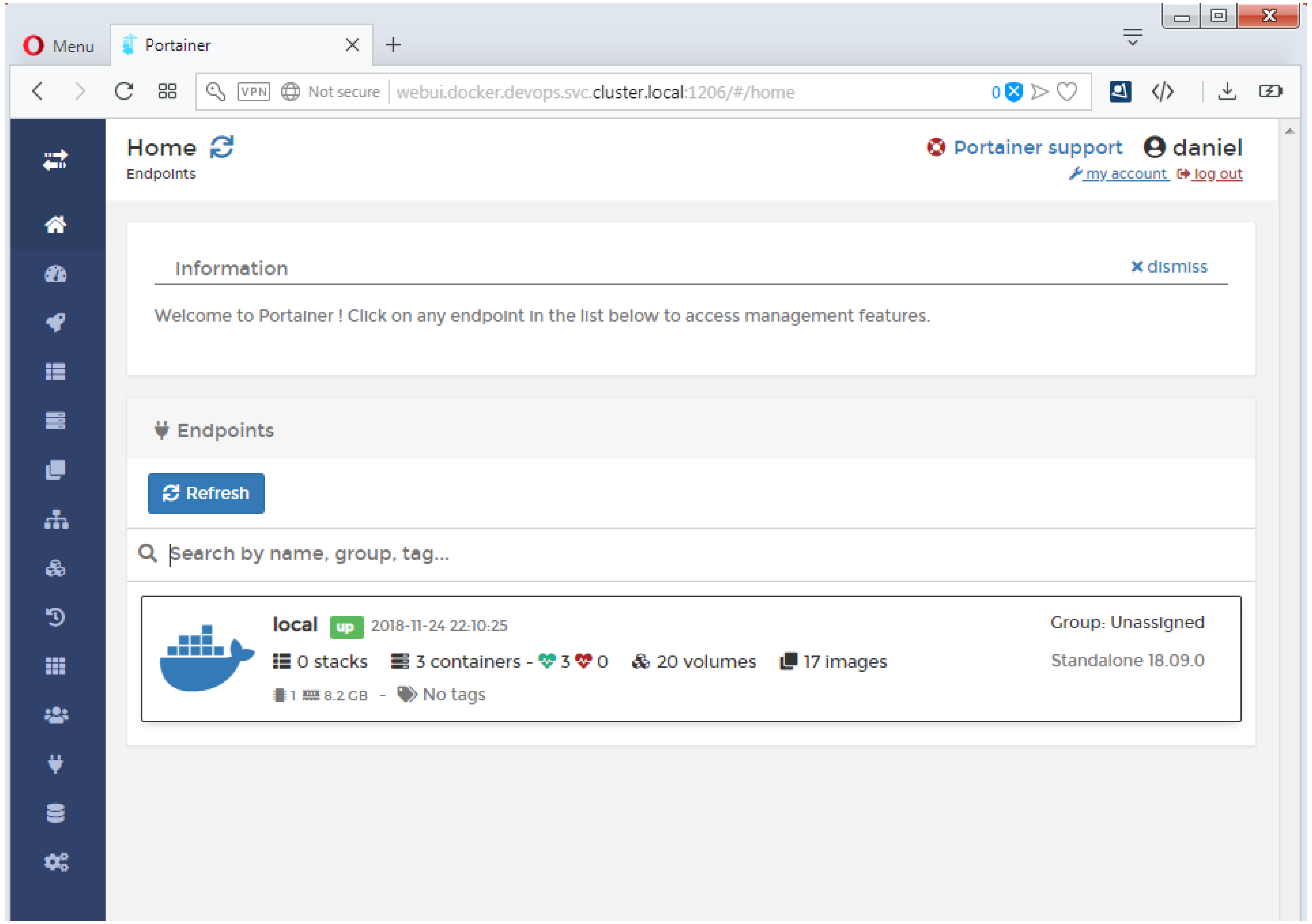
Start your first SSH session as an SSH tunnel like so

Only one Docker Container is currently running – the Portainer



```
daschulz@CE10900 MINGW64 ~
$ # 1st SSH session aka the SSH tunnel
    -L webui.docker.devops.svc.cluster.local:1206:127.0.0.1:1206 \
    -L daemon.docker.devops.svc.cluster.local:1906:127.0.0.1:1906 \
    -L iris.devops.svc.cluster.local:2008:127.0.0.1:2008 \
    -L deeplearning.devops.svc.cluster.local:2308:127.0.0.1:2308 \
    root@host.docker.devops.svc.cluster.local \
    -i /c/Apps/Current/Keys/ssh-key-8k

daschulz@CE10900 MINGW64 ~
$ ssh \
>     -L webui.docker.devops.svc.cluster.local:1206:127.0.0.1:1206 \
>     -L daemon.docker.devops.svc.cluster.local:1906:127.0.0.1:1906 \
>     -L iris.devops.svc.cluster.local:2008:127.0.0.1:2008 \
>     -L deeplearning.devops.svc.cluster.local:2308:127.0.0.1:2308 \
>     root@host.docker.devops.svc.cluster.local \
>     -i /c/Apps/Current/Keys/ssh-key-8k
Last login:                     2018 from
[root@de-muc-anastasia01 ~]# docker ps # show running Docker Containers
CONTAINER ID        IMAGE                 COMMAND             CREATED             STATUS              PORTS                      NAMES
dc6eb9cb401f        portainer/portainer   "/portainer"        10 days ago         Up 10 days          0.0.0.0:1206->9000/tcp     daniel-docker-daemon-ui
[root@de-muc-anastasia01 ~]# docker ps -a # show running and stopped Docker Containers
CONTAINER ID        IMAGE                 COMMAND             CREATED             STATUS              PORTS                      NAMES
dc6eb9cb401f        portainer/portainer   "/portainer"        10 days ago         Up 10 days          0.0.0.0:1206->9000/tcp     daniel-docker-daemon-ui
[root@de-muc-anastasia01 ~]#
```

Portainer Provides a Mgmt. Web UI



Information

× dismiss

Welcome to Portainer ! Click on any endpoint in the list below to access management features.

Endpoints

Refresh

Search by name, group, tag...

local up 2018-11-24 22:10:25

0 stacks 3 containers - 3 0 20 volumes 17 images

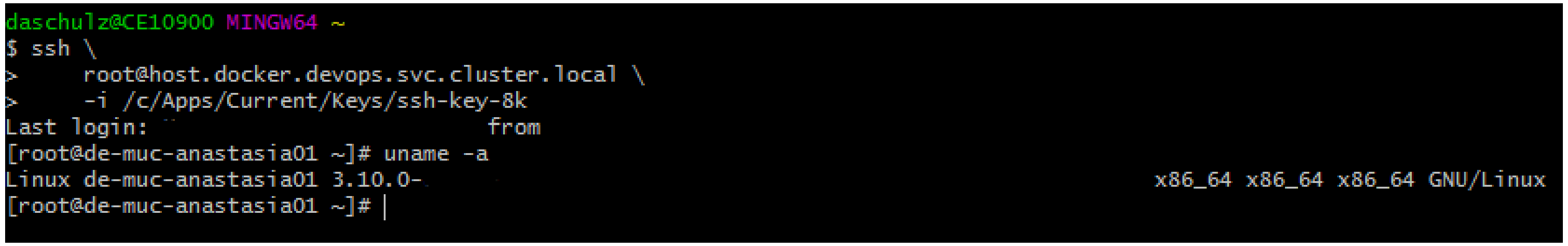
1 8.2 GB - No tags

Group: Unassigned

Standalone 18.09.0

Optional: Establish the 2nd, 3rd, 4th, etc. Remote SSH Bash Sessions for Your Convenience

This command does not claim any ports for them to be already claimed by your 1st SSH tunnel session



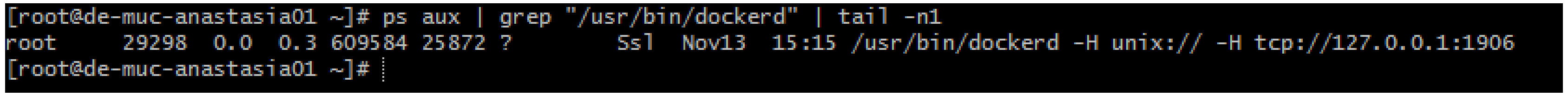
```
daschulz@CE10900 MINGW64 ~
$ ssh \
>     root@host.docker.devops.svc.cluster.local \
>     -i /c/Apps/Current/Keys/ssh-key-8k
Last login: [redacted]                          from [redacted]
[root@de-muc-anastasia01 ~]# uname -a
Linux de-muc-anastasia01 3.10.0-[redacted]                              x86_64 x86_64 x86_64 GNU/Linux
[root@de-muc-anastasia01 ~]# |
```

Optional: for the IDE Integration to Work, the Docker Daemon Needs to Listen to an Accessible Port

Port `1906` on `127.0.0.1` is only accessible

- from the server by `` `-H unix://` `` and
- through `127.0.0.1:1906` the SSH tunnel by `` `-H tcp://127.0.0.1:1906` ``

in the `dockerd` start command



```
[root@de-muc-anastasia01 ~]# ps aux | grep "/usr/bin/dockerd" | tail -n1
root     29298  0.0  0.3 609584 25872 ?        Ssl  Nov13  15:15 /usr/bin/dockerd -H unix:// -H tcp://127.0.0.1:1906
[root@de-muc-anastasia01 ~]#
```

Port Management –
A ReSTful API for the Iris Model on TensorFlow

From the outside available network ports on the host map to the Docker Container's internal ports:

Docker Image:

- name: `tensorflow/serving`
- tag: `latest`

Host – defined by requirements/developer for usage:

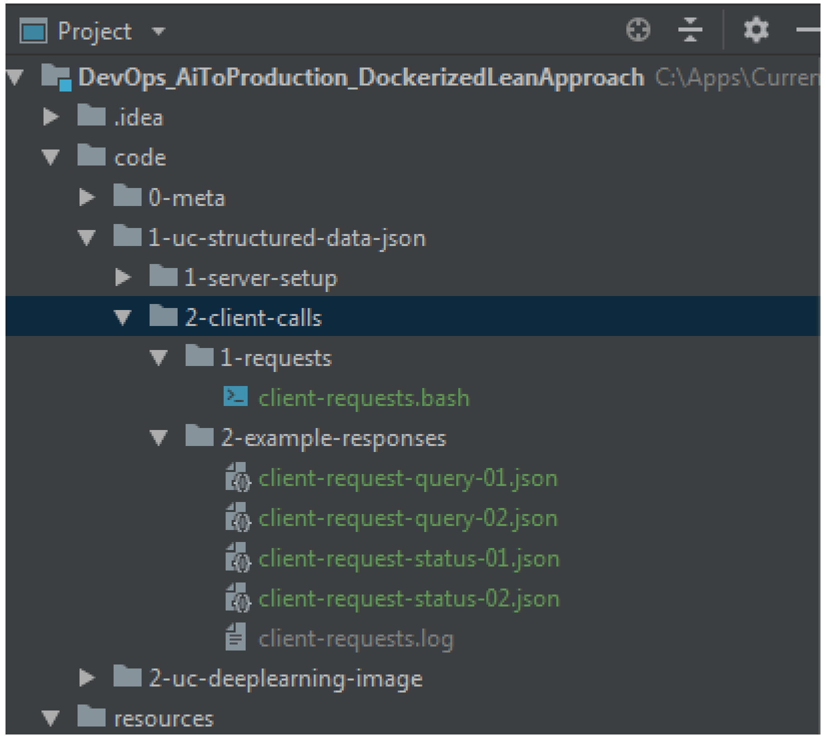
- FQDN: `iris.devops.svc.cluster.local`
- Port: `2008`

Docker Container – pre-defined by provider of Docker Image:

- Port: `8501`
- URI: `/v1/models/iris`

The TensorFlow / Iris Queries for the Structured, JSON-based Classification can be Found Here in the Git Repository

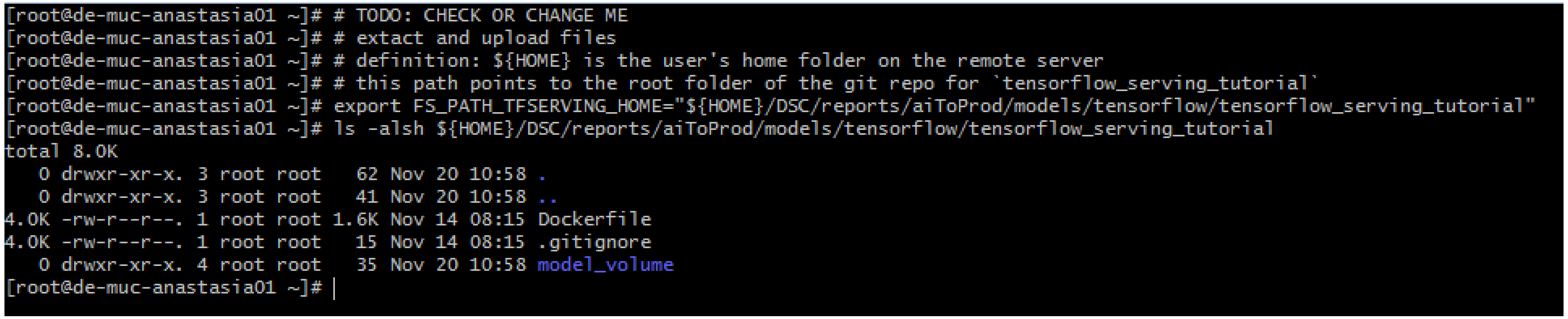
`code/1-uc-structured-data-json/2-client-calls`



```
Project
▼ DevOps_AiToProduction_DockerizedLeanApproach C:\Apps\Curren
  ▶ .idea
  ▼ code
    ▶ 0-meta
    ▼ 1-uc-structured-data-json
      ▶ 1-server-setup
      ▼ 2-client-calls
        ▼ 1-requests
            client-requests.bash
        ▼ 2-example-responses
            client-request-query-01.json
            client-request-query-02.json
            client-request-status-01.json
            client-request-status-02.json
            client-requests.log
    ▶ 2-uc-deeplearning-image
  ▼ resources
```

TensorFlow Serving Relies on the Host Machine to Have the Model’s Root Path Defined in/to `${FS_PATH_TFSERVING_HOME}`

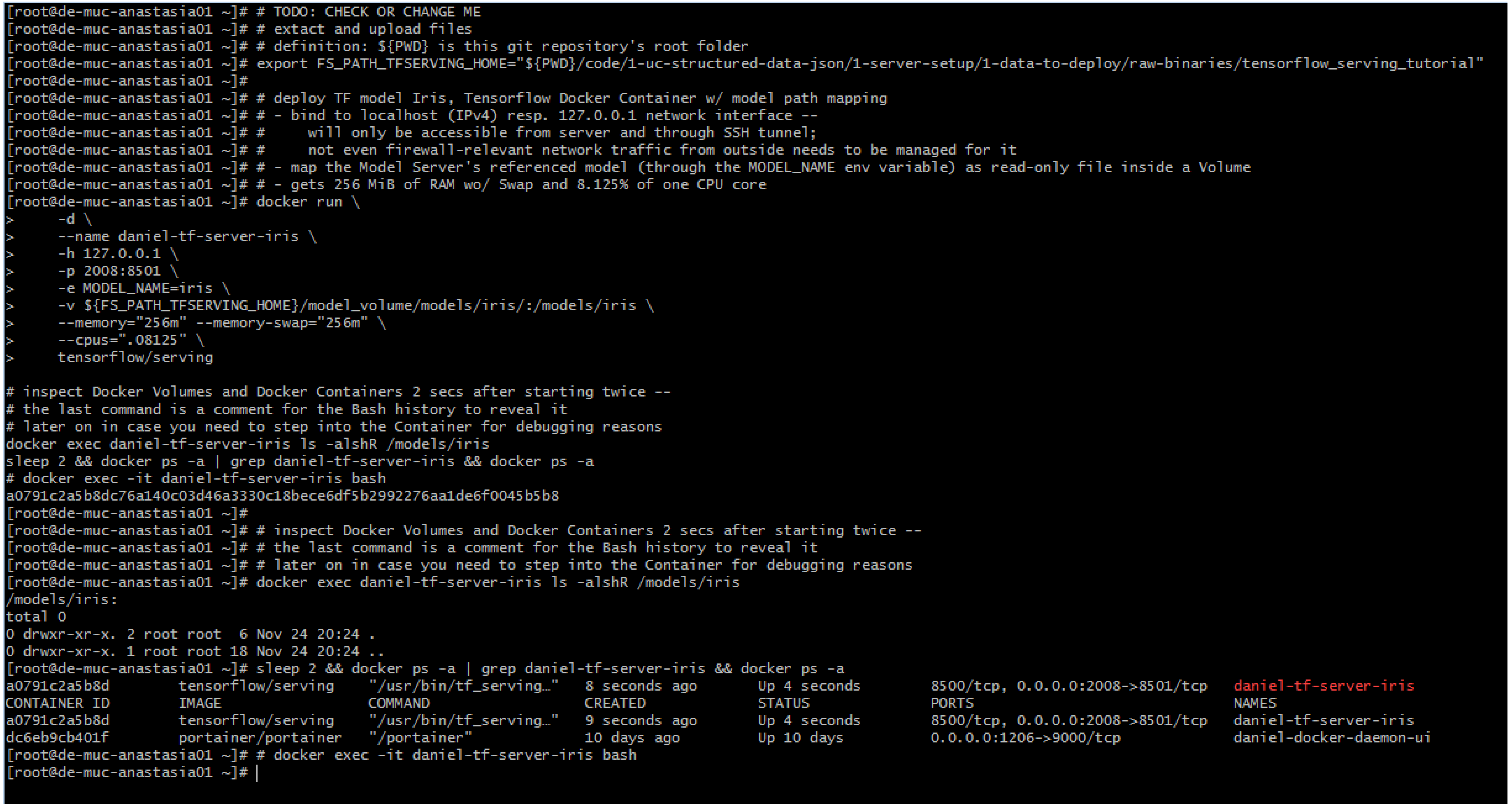
The Data is Found in it’s Expected Place



```
[root@de-muc-anastasia01 ~]# # TODO: CHECK OR CHANGE ME
[root@de-muc-anastasia01 ~]# # extact and upload files
[root@de-muc-anastasia01 ~]# # definition: ${HOME} is the user's home folder on the remote server
[root@de-muc-anastasia01 ~]# # this path points to the root folder of the git repo for `tensorflow_serving_tutorial`
[root@de-muc-anastasia01 ~]# export FS_PATH_TFSERVING_HOME="${HOME}/DSC/reports/aiToProd/models/tensorflow/tensorflow_serving_tutorial"
[root@de-muc-anastasia01 ~]# ls -alsh ${HOME}/DSC/reports/aiToProd/models/tensorflow/tensorflow_serving_tutorial
total 8.0K
   0 drwxr-xr-x. 3 root root   62 Nov 20 10:58 .
   0 drwxr-xr-x. 3 root root   41 Nov 20 10:58 ..
4.0K -rw-r--r--. 1 root root 1.6K Nov 14 08:15 Dockerfile
4.0K -rw-r--r--. 1 root root   15 Nov 14 08:15 .gitignore
   0 drwxr-xr-x. 4 root root   35 Nov 20 10:58 model_volume
[root@de-muc-anastasia01 ~]# |
```

TensorFlow Serving’s Container Startup Log w/ Following Inspection Commands

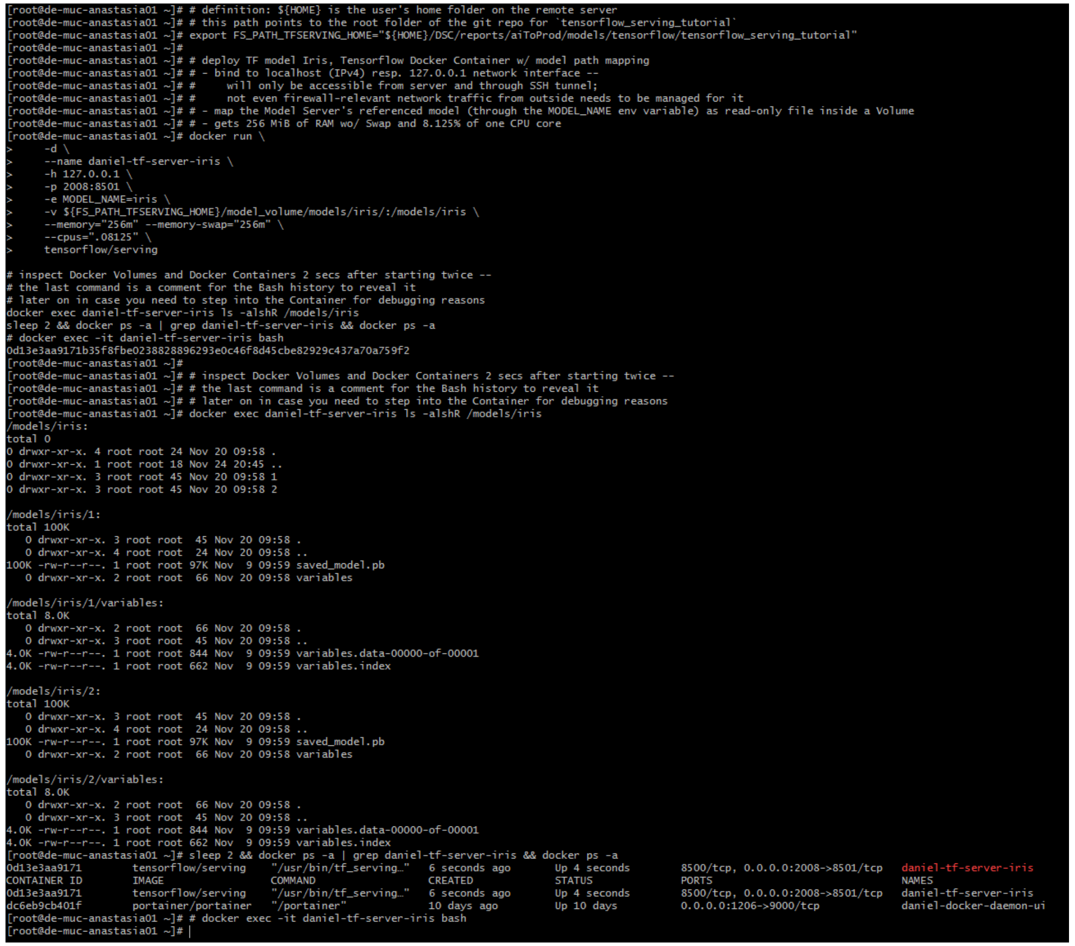
A Successful Deployment Finds the Container Up & Running



```
[root@de-muc-anastasia01 ~]# # TODO: CHECK OR CHANGE ME
[root@de-muc-anastasia01 ~]# # extact and upload files
[root@de-muc-anastasia01 ~]# # definition: ${PWD} is this git repository's root folder
[root@de-muc-anastasia01 ~]# export FS_PATH_TFSERVING_HOME="${PWD}/code/1-uc-structured-data-json/1-server-setup/1-data-to-deploy/raw-binaries/tensorflow_serving_tutorial"
[root@de-muc-anastasia01 ~]#
[root@de-muc-anastasia01 ~]# # deploy TF model Iris, Tensorflow Docker Container w/ model path mapping
[root@de-muc-anastasia01 ~]# # - bind to localhost (IPv4) resp. 127.0.0.1 network interface --
[root@de-muc-anastasia01 ~]# #     will only be accessible from server and through SSH tunnel;
[root@de-muc-anastasia01 ~]# #     not even firewall-relevant network traffic from outside needs to be managed for it
[root@de-muc-anastasia01 ~]# # - map the Model Server's referenced model (through the MODEL_NAME env variable) as read-only file inside a Volume
[root@de-muc-anastasia01 ~]# # - gets 256 MiB of RAM wo/ Swap and 8.125% of one CPU core
[root@de-muc-anastasia01 ~]# docker run \
>     -d \
>     --name daniel-tf-server-iris \
>     -h 127.0.0.1 \
>     -p 2008:8501 \
>     -e MODEL_NAME=iris \
>     -v ${FS_PATH_TFSERVING_HOME}/model_volume/models/iris/:/models/iris \
>     --memory="256m" --memory-swap="256m" \
>     --cpus=".08125" \
>     tensorflow/serving

# inspect Docker Volumes and Docker Containers 2 secs after starting twice --
# the last command is a comment for the Bash history to reveal it
# later on in case you need to step into the Container for debugging reasons
docker exec daniel-tf-server-iris ls -alshR /models/iris
sleep 2 && docker ps -a | grep daniel-tf-server-iris && docker ps -a
# docker exec -it daniel-tf-server-iris bash
a0791c2a5b8dc76a140c03d46a3330c18bece6df5b2992276aa1de6f0045b5b8
[root@de-muc-anastasia01 ~]#
[root@de-muc-anastasia01 ~]# # inspect Docker Volumes and Docker Containers 2 secs after starting twice --
[root@de-muc-anastasia01 ~]# # the last command is a comment for the Bash history to reveal it
[root@de-muc-anastasia01 ~]# # later on in case you need to step into the Container for debugging reasons
[root@de-muc-anastasia01 ~]# docker exec daniel-tf-server-iris ls -alshR /models/iris
/models/iris:
total 0
0 drwxr-xr-x. 2 root root  6 Nov 24 20:24 .
0 drwxr-xr-x. 1 root root 18 Nov 24 20:24 ..
[root@de-muc-anastasia01 ~]# sleep 2 && docker ps -a | grep daniel-tf-server-iris && docker ps -a
a0791c2a5b8d        tensorflow/serving    "/usr/bin/tf_serving…"   8 seconds ago       Up 4 seconds        8500/tcp, 0.0.0.0:2008->8501/tcp   daniel-tf-server-iris
CONTAINER ID        IMAGE                 COMMAND                  CREATED             STATUS              PORTS                              NAMES
a0791c2a5b8d        tensorflow/serving    "/usr/bin/tf_serving…"   9 seconds ago       Up 4 seconds        8500/tcp, 0.0.0.0:2008->8501/tcp   daniel-tf-server-iris
dc6eb9cb401f        portainer/portainer   "/portainer"             10 days ago         Up 10 days          0.0.0.0:1206->9000/tcp             daniel-docker-daemon-ui
[root@de-muc-anastasia01 ~]# # docker exec -it daniel-tf-server-iris bash
[root@de-muc-anastasia01 ~]# |
```

TensorFlow Serving’s Container Startup Log w/ Following Inspection Commands

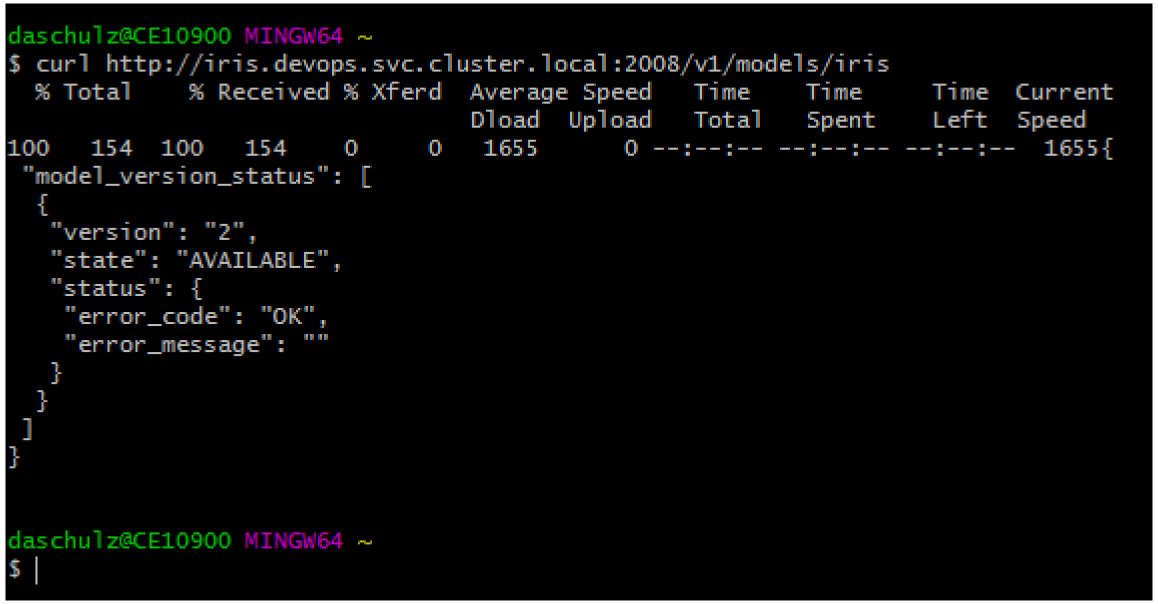
A Successful Deployment Finds also the Model Files Mapped Inside the Docker Container



```
[root@de-muc-anastasia01 ~]# # definition: ${HOME} is the user's home folder on the remote server
[root@de-muc-anastasia01 ~]# # this path points to the root folder of the git repo for `tensorflow_serving_tutorial`
[root@de-muc-anastasia01 ~]# export FS_PATH_TFSERVING_HOME="${HOME}/DSC/reports/aiToProd/models/tensorflow/tensorflow_serving_tutorial"
[root@de-muc-anastasia01 ~]#
[root@de-muc-anastasia01 ~]# # deploy TF model Iris, Tensorflow Docker Container w/ model path mapping
[root@de-muc-anastasia01 ~]# # - bind to localhost (IPv4) resp. 127.0.0.1 network interface --
[root@de-muc-anastasia01 ~]# #     will only be accessible from server and through SSH tunnel;
[root@de-muc-anastasia01 ~]# #     not even firewall-relevant network traffic from outside needs to be managed for it
[root@de-muc-anastasia01 ~]# # - map the Model Server's referenced model (through the MODEL_NAME env variable) as read-only file inside a Volume
[root@de-muc-anastasia01 ~]# # - gets 256 MiB of RAM wo/ Swap and 8.125% of one CPU core
[root@de-muc-anastasia01 ~]# docker run \
>     -d \
>     --name daniel-tf-server-iris \
>     -h 127.0.0.1 \
>     -p 2008:8501 \
>     -e MODEL_NAME=iris \
>     -v ${FS_PATH_TFSERVING_HOME}/model_volume/models/iris/:/models/iris \
>     --memory="256m" --memory-swap="256m" \
>     --cpus=".08125" \
>     tensorflow/serving

# inspect Docker Volumes and Docker Containers 2 secs after starting twice --
# the last command is a comment for the Bash history to reveal it
# later on in case you need to step into the Container for debugging reasons
docker exec daniel-tf-server-iris ls -alshR /models/iris
sleep 2 && docker ps -a | grep daniel-tf-server-iris && docker ps -a
# docker exec -it daniel-tf-server-iris bash
0d13e3aa9171b35f8fbe0238828896293e0c46f8d45cbe82929c437a70a759f2
[root@de-muc-anastasia01 ~]#
[root@de-muc-anastasia01 ~]# # inspect Docker Volumes and Docker Containers 2 secs after starting twice --
[root@de-muc-anastasia01 ~]# # the last command is a comment for the Bash history to reveal it
[root@de-muc-anastasia01 ~]# # later on in case you need to step into the Container for debugging reasons
[root@de-muc-anastasia01 ~]# docker exec daniel-tf-server-iris ls -alshR /models/iris
/models/iris:
total 0
0 drwxr-xr-x. 4 root root 24 Nov 20 09:58 .
0 drwxr-xr-x. 1 root root 18 Nov 24 20:45 ..
0 drwxr-xr-x. 3 root root 45 Nov 20 09:58 1
0 drwxr-xr-x. 3 root root 45 Nov 20 09:58 2

/models/iris/1:
total 100K
   0 drwxr-xr-x. 3 root root  45 Nov 20 09:58 .
   0 drwxr-xr-x. 4 root root  24 Nov 20 09:58 ..
100K -rw-r--r--. 1 root root 97K Nov  9 09:59 saved_model.pb
   0 drwxr-xr-x. 2 root root  66 Nov 20 09:58 variables

/models/iris/1/variables:
total 8.0K
   0 drwxr-xr-x. 2 root root  66 Nov 20 09:58 .
   0 drwxr-xr-x. 3 root root  45 Nov 20 09:58 ..
4.0K -rw-r--r--. 1 root root 844 Nov  9 09:59 variables.data-00000-of-00001
4.0K -rw-r--r--. 1 root root 662 Nov  9 09:59 variables.index

/models/iris/2:
total 100K
   0 drwxr-xr-x. 3 root root  45 Nov 20 09:58 .
   0 drwxr-xr-x. 4 root root  24 Nov 20 09:58 ..
100K -rw-r--r--. 1 root root 97K Nov  9 09:59 saved_model.pb
   0 drwxr-xr-x. 2 root root  66 Nov 20 09:58 variables

/models/iris/2/variables:
total 8.0K
   0 drwxr-xr-x. 2 root root  66 Nov 20 09:58 .
   0 drwxr-xr-x. 3 root root  45 Nov 20 09:58 ..
4.0K -rw-r--r--. 1 root root 844 Nov  9 09:59 variables.data-00000-of-00001
4.0K -rw-r--r--. 1 root root 662 Nov  9 09:59 variables.index
[root@de-muc-anastasia01 ~]# sleep 2 && docker ps -a | grep daniel-tf-server-iris && docker ps -a
0d13e3aa9171        tensorflow/serving    "/usr/bin/tf_serving…"   6 seconds ago       Up 4 seconds        8500/tcp, 0.0.0.0:2008->8501/tcp   daniel-tf-server-iris
CONTAINER ID        IMAGE                 COMMAND                  CREATED             STATUS              PORTS                              NAMES
0d13e3aa9171        tensorflow/serving    "/usr/bin/tf_serving…"   6 seconds ago       Up 4 seconds        8500/tcp, 0.0.0.0:2008->8501/tcp   daniel-tf-server-iris
dc6eb9cb401f        portainer/portainer   "/portainer"             10 days ago         Up 10 days          0.0.0.0:1206->9000/tcp             daniel-docker-daemon-ui
[root@de-muc-anastasia01 ~]# # docker exec -it daniel-tf-server-iris bash
[root@de-muc-anastasia01 ~]# |
```

From the Client Machine, TensorFlow Serving Responds to Status Requests

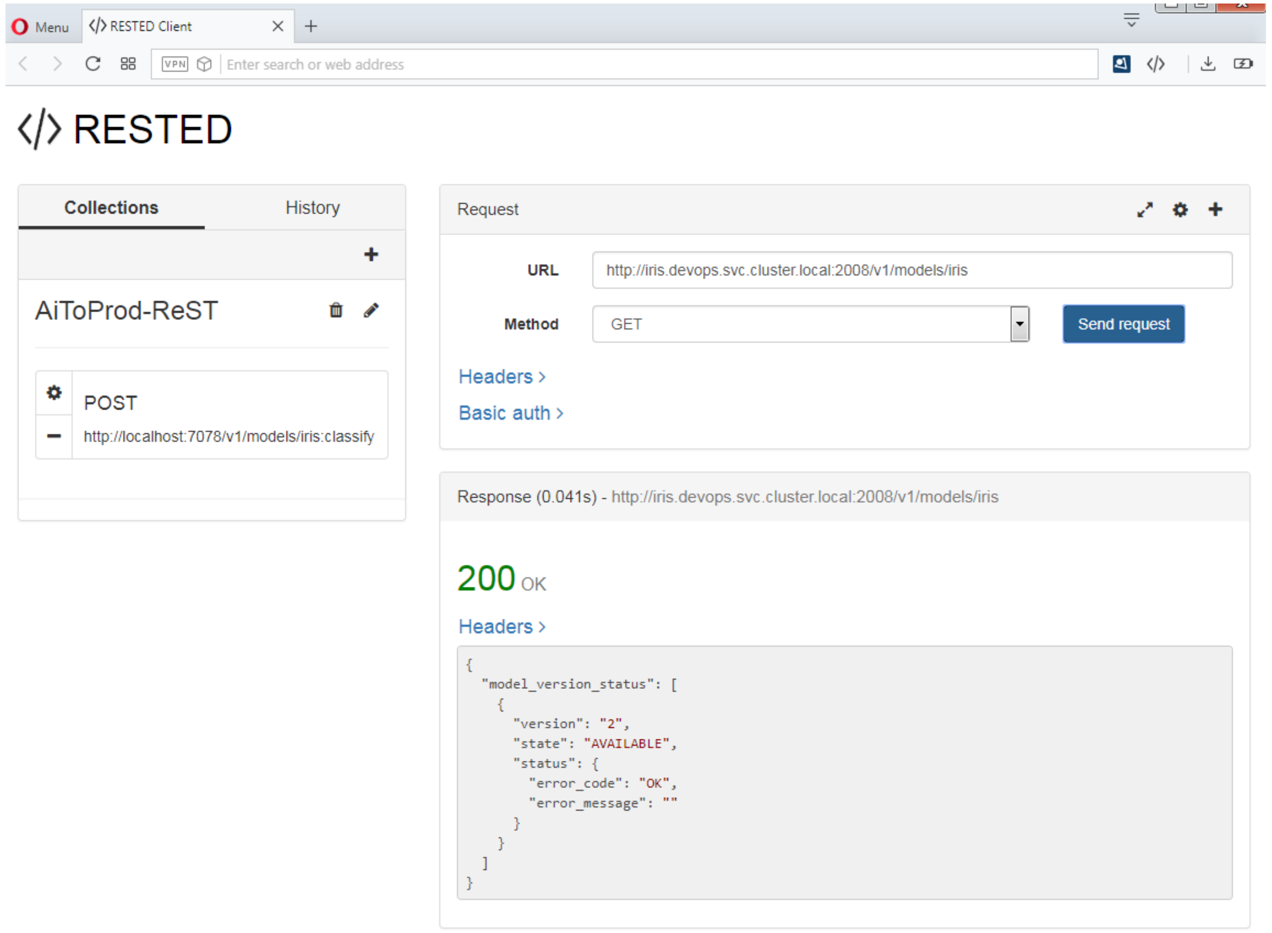
Once the Model is Successfully Deployed, the Status API Speaks with us



```
daschulz@CE10900 MINGW64 ~
$ curl http://iris.devops.svc.cluster.local:2008/v1/models/iris
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   154  100   154    0     0   1655      0 --:--:-- --:--:-- --:--:--  1655{
 "model_version_status": [
  {
   "version": "2",
   "state": "AVAILABLE",
   "status": {
    "error_code": "OK",
    "error_message": ""
   }
  }
 ]
}


daschulz@CE10900 MINGW64 ~
$ |
```

From the Client Machine, TensorFlow Serving Responds to Status Requests

Once the Model is Successfully Deployed, the Status API Speaks with us



URL: http://iris.devops.svc.cluster.local:2008/v1/models/iris

Method: GET

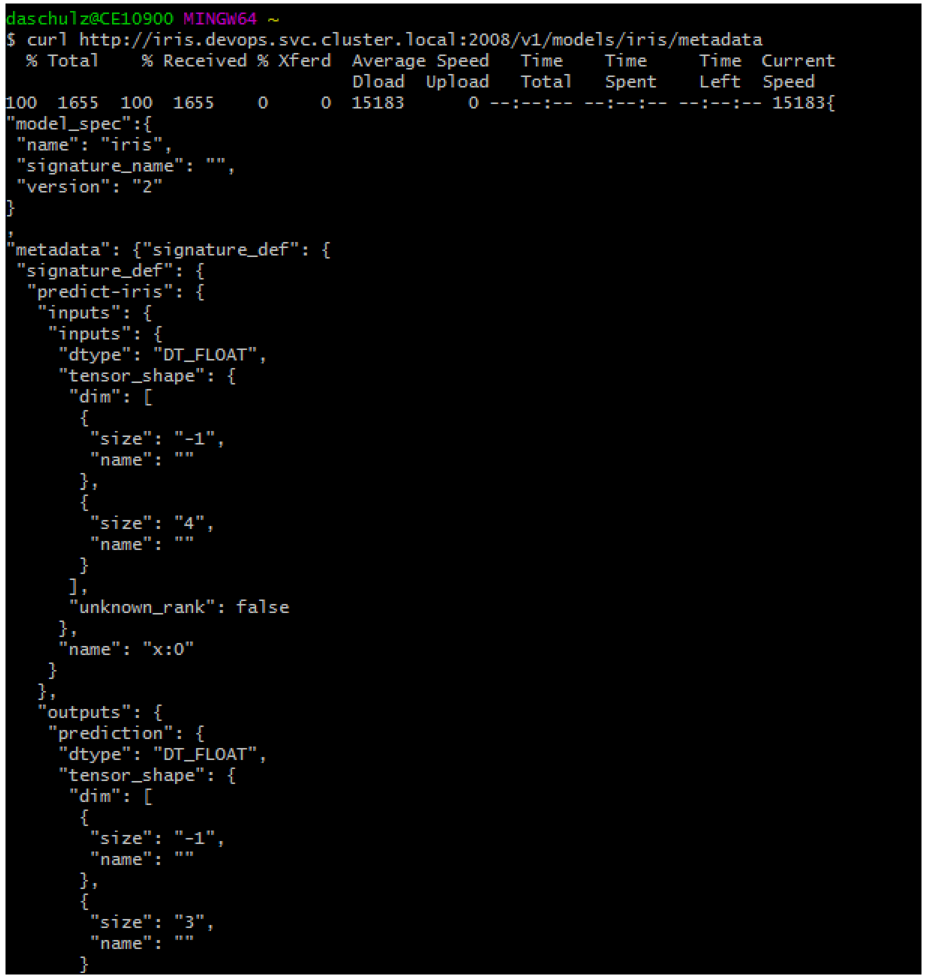
Response (0.041s) - http://iris.devops.svc.cluster.local:2008/v1/models/iris

200 OK

```
{
  "model_version_status": [
    {
      "version": "2",
      "state": "AVAILABLE",
      "status": {
        "error_code": "OK",
        "error_message": ""
      }
    }
  ]
}
```

From the Client Machine, TensorFlow Serving Responds to Status Requests

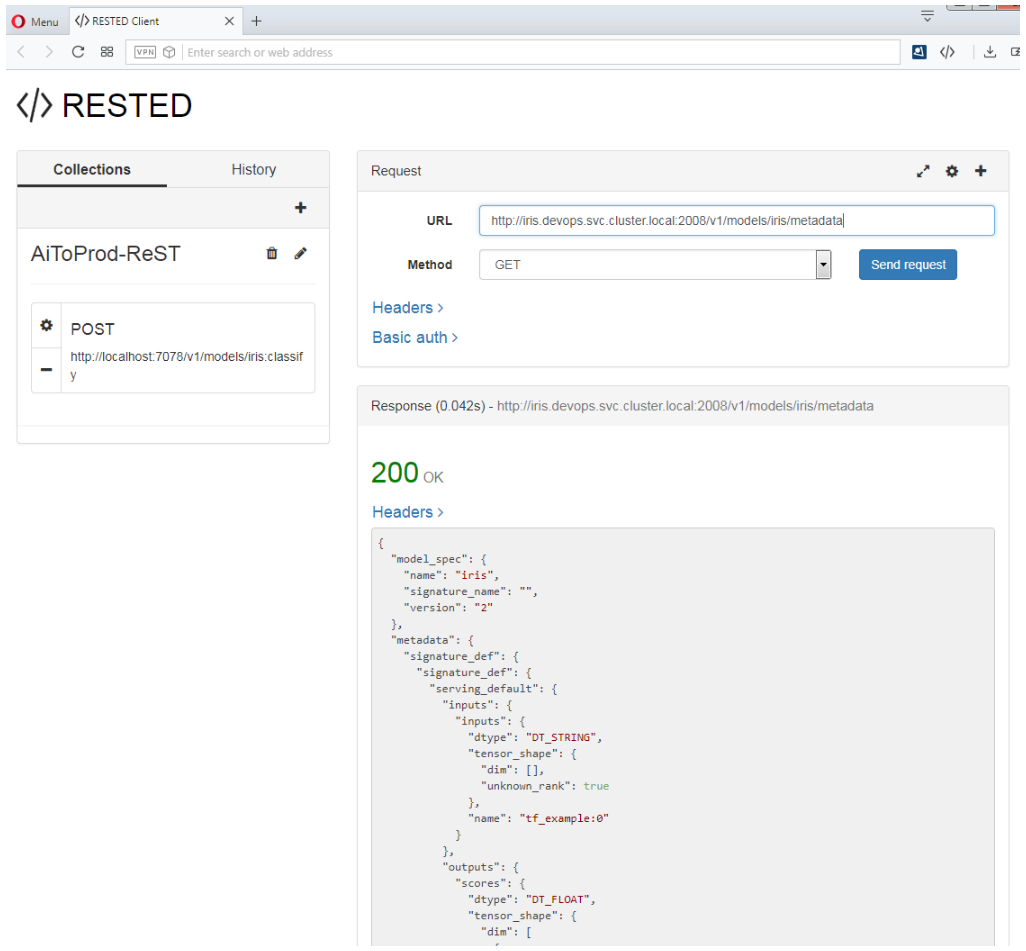
Once the Model is Successfully Deployed, the Status API Speaks with us



```
daschulz@CE10900 MINGW64 ~
$ curl http://iris.devops.svc.cluster.local:2008/v1/models/iris/metadata
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1655  100  1655    0     0  15183      0 --:--:-- --:--:-- --:--:-- 15183{
"model_spec":{
 "name": "iris",
 "signature_name": "",
 "version": "2"
}
,
"metadata": {"signature_def": {
 "signature_def": {
  "predict-iris": {
   "inputs": {
    "inputs": {
     "dtype": "DT_FLOAT",
     "tensor_shape": {
      "dim": [
       {
        "size": "-1",
        "name": ""
       },
       {
        "size": "4",
        "name": ""
       }
      ],
      "unknown_rank": false
     },
     "name": "x:0"
    }
   },
   "outputs": {
    "prediction": {
     "dtype": "DT_FLOAT",
     "tensor_shape": {
      "dim": [
       {
        "size": "-1",
        "name": ""
       },
       {
        "size": "3",
        "name": ""
       }
```

From the Client Machine, TensorFlow Serving Responds to Status Requests

Once the Model is Successfully Deployed, the Status API Speaks with us



URL: http://iris.devops.svc.cluster.local:2008/v1/models/iris/metadata
Method: GET

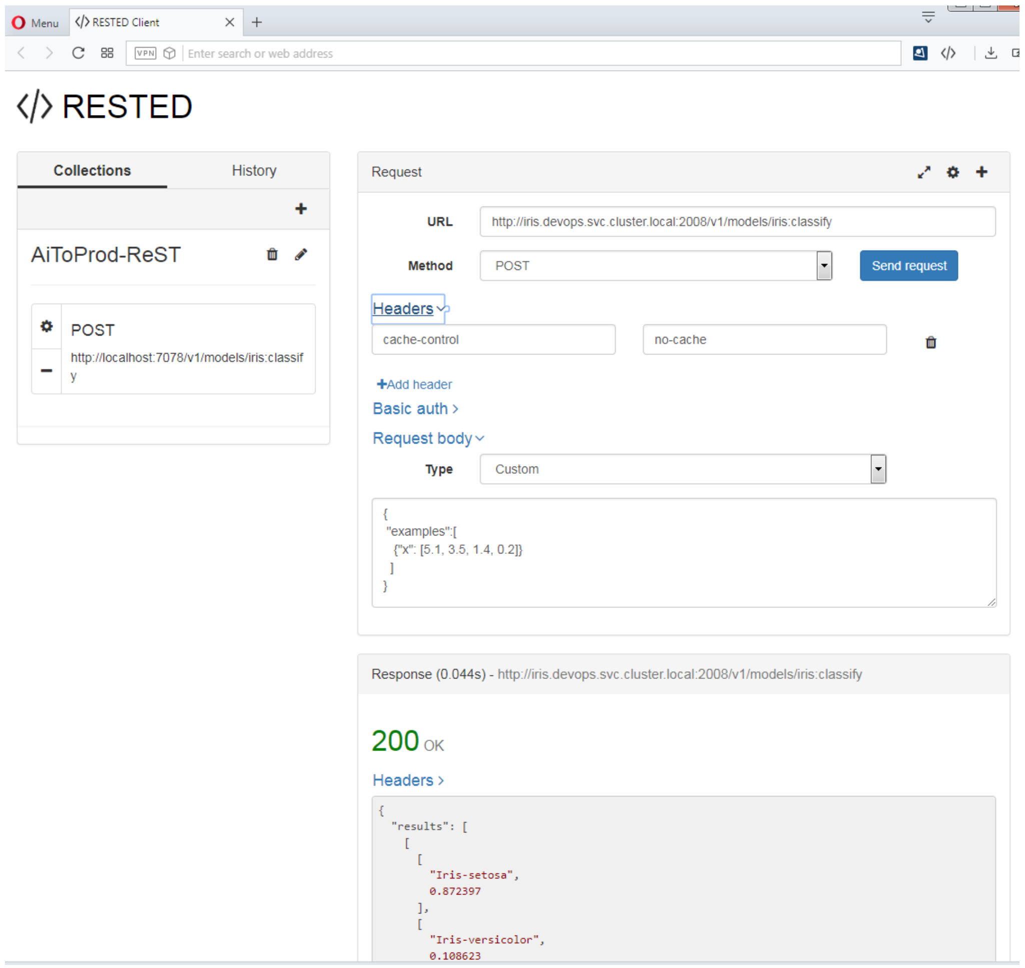
Response (0.042s) - http://iris.devops.svc.cluster.local:2008/v1/models/iris/metadata

200 OK

```
{
  "model_spec": {
    "name": "iris",
    "signature_name": "",
    "version": "2"
  },
  "metadata": {
    "signature_def": {
      "signature_def": {
        "serving_default": {
          "inputs": {
            "inputs": {
              "dtype": "DT_STRING",
              "tensor_shape": {
                "dim": [],
                "unknown_rank": true
              },
              "name": "tf_example:0"
            }
          },
          "outputs": {
            "scores": {
              "dtype": "DT_FLOAT",
              "tensor_shape": {
                "dim": [
```

From the Client Machine, TensorFlow Serving Responds to the Classification Request #1

Once the Model is Successfully Deployed, the Status API Speaks with us



POST
http://localhost:7078/v1/models/iris:classify

URL: http://iris.devops.svc.cluster.local:2008/v1/models/iris:classify
Method: POST

Headers: cache-control: no-cache

Type: Custom

```
{
 "examples":[
  {"x": [5.1, 3.5, 1.4, 0.2]}
 ]
}
```

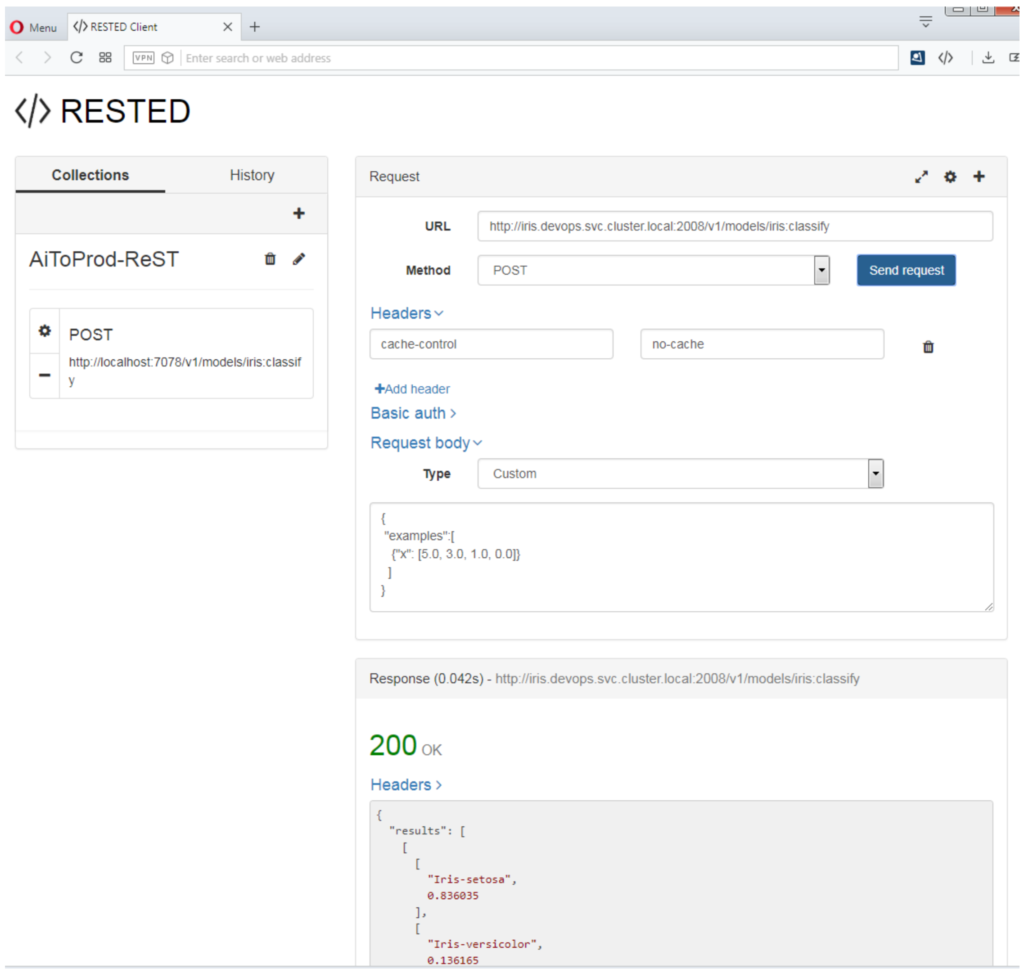
Response (0.044s) - http://iris.devops.svc.cluster.local:2008/v1/models/iris:classify

200 OK

```
{
  "results": [
    [
      [
        "Iris-setosa",
        0.872397
      ],
      [
        "Iris-versicolor",
        0.108623
```

From the Client Machine, TensorFlow Serving Responds to the Classification Request #2

Once the Model is Successfully Deployed, the Status API Speaks with us



POST http://localhost:7078/v1/models/iris:classify

URL: http://iris.devops.svc.cluster.local:2008/v1/models/iris:classify

Method: POST

cache-control: no-cache

Type: Custom

```
{
 "examples":[
  {"x": [5.0, 3.0, 1.0, 0.0]}
 ]
}
```

Response (0.042s) - http://iris.devops.svc.cluster.local:2008/v1/models/iris:classify

200 OK

```
{
  "results": [
    [
      [
        "Iris-setosa",
        0.836035
      ],
      [
        "Iris-versicolor",
        0.136165
```

Port Management –
A ReSTful API for the Iris Model on MXNet

From the outside available network ports on the host map to the Docker Container's internal ports:

Docker Image:

- name: `awsdeeplearningteam/mms_cpu mxnet-model-server`
- tag: `latest`

Host – defined by requirements/developer for usage:

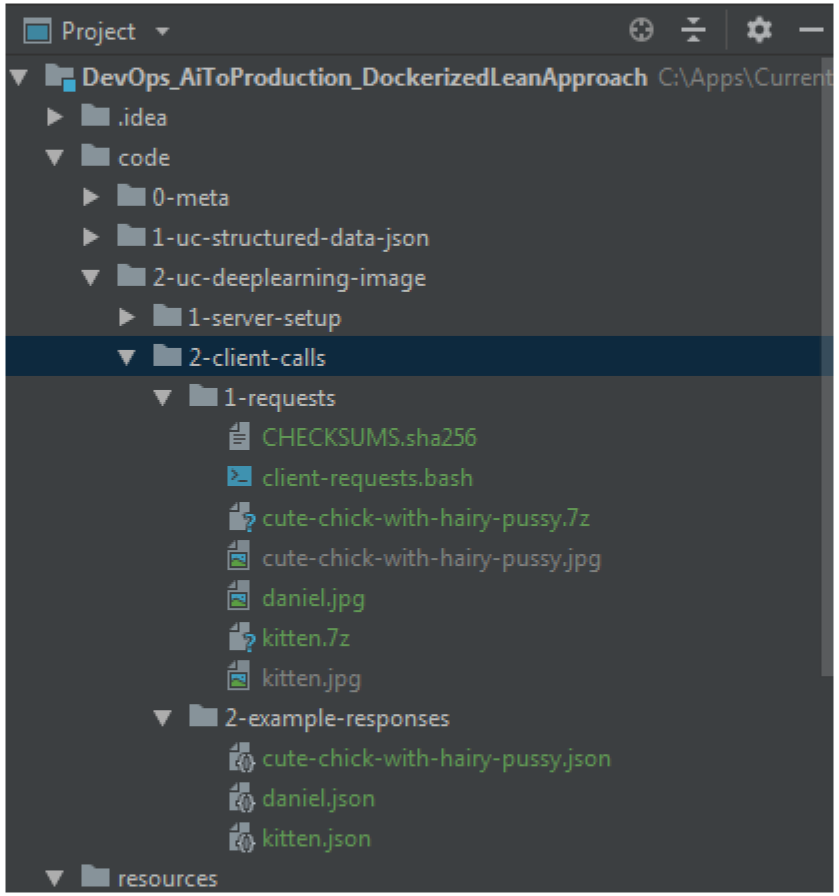
- FQDN: `deeplearning.devops.svc.cluster.local`
- Port: `2308`

Docker Container – pre-defined by provider of Docker Image:

- Port: `8080`
- URI: `/squeezenet/predict`

The MXNet / Deep Learning Queries for the Image Classification can be Found Here in the Git Repository

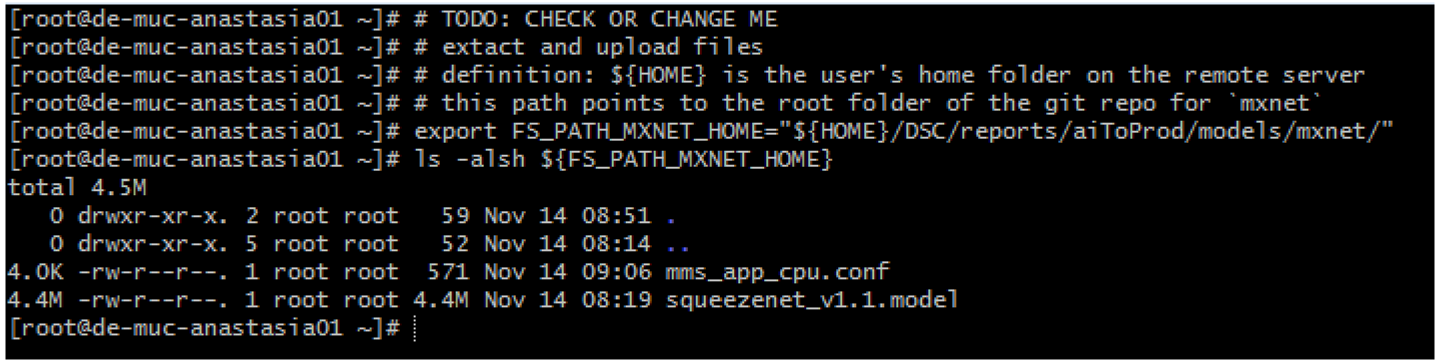
`code/2-uc-deeplearning-image/2-client-calls`



```
Project
▼ DevOps_AiToProduction_DockerizedLeanApproach C:\Apps\Current
  ▶ .idea
  ▼ code
    ▶ 0-meta
    ▶ 1-uc-structured-data-json
    ▼ 2-uc-deeplearning-image
      ▶ 1-server-setup
      ▼ 2-client-calls
        ▼ 1-requests
            CHECKSUMS.sha256
            client-requests.bash
            cute-chick-with-hairy-pussy.7z
            cute-chick-with-hairy-pussy.jpg
            daniel.jpg
            kitten.7z
            kitten.jpg
        ▼ 2-example-responses
            cute-chick-with-hairy-pussy.json
            daniel.json
            kitten.json
  ▼ resources
```

MXNet Model Server Relies on the Host Machine to Have the Model’s Root Path Defined in/to `${FS_PATH_MXNET_HOME}`

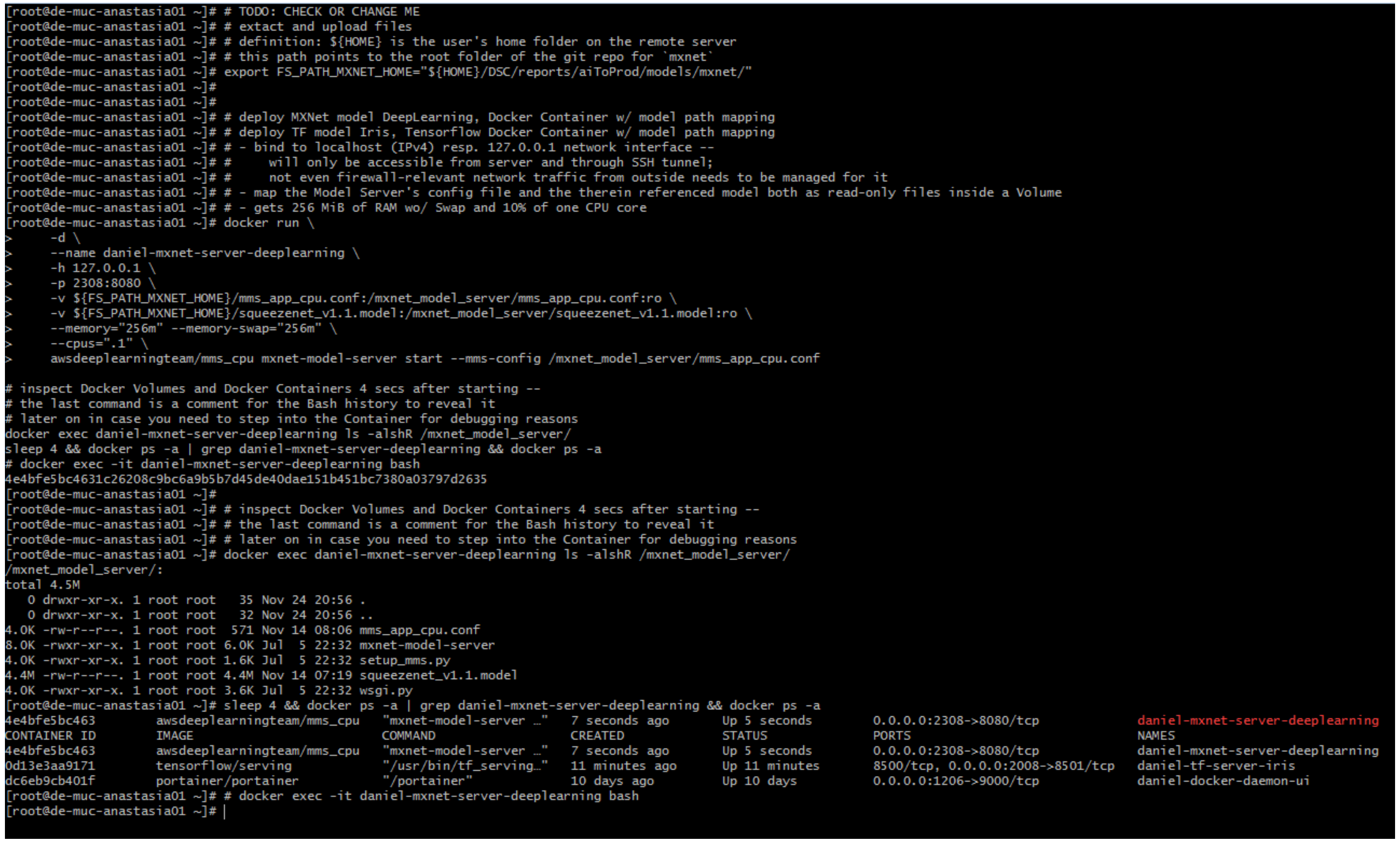
The Data is Found in it’s Expected Place



```
[root@de-muc-anastasia01 ~]# # TODO: CHECK OR CHANGE ME
[root@de-muc-anastasia01 ~]# # extact and upload files
[root@de-muc-anastasia01 ~]# # definition: ${HOME} is the user's home folder on the remote server
[root@de-muc-anastasia01 ~]# # this path points to the root folder of the git repo for `mxnet`
[root@de-muc-anastasia01 ~]# export FS_PATH_MXNET_HOME="${HOME}/DSC/reports/aiToProd/models/mxnet/"
[root@de-muc-anastasia01 ~]# ls -alsh ${FS_PATH_MXNET_HOME}
total 4.5M
   0 drwxr-xr-x. 2 root root   59 Nov 14 08:51 .
   0 drwxr-xr-x. 5 root root   52 Nov 14 08:14 ..
4.0K -rw-r--r--. 1 root root  571 Nov 14 09:06 mms_app_cpu.conf
4.4M -rw-r--r--. 1 root root 4.4M Nov 14 08:19 squeezenet_v1.1.model
[root@de-muc-anastasia01 ~]#
```

MXNet Model Server’s Container Startup Log w/ Following Inspection Commands

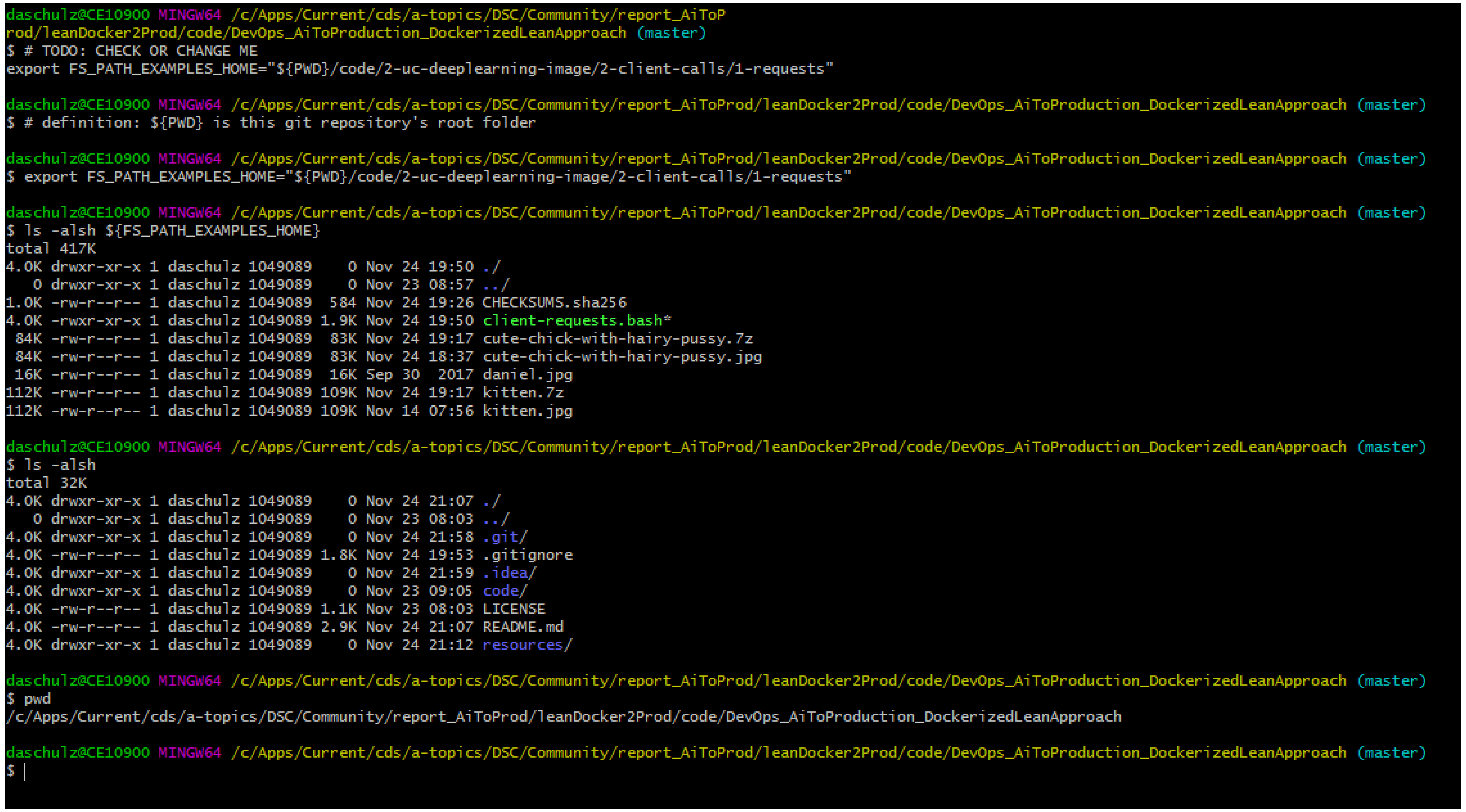
A Successful Deployment Finds the Container Up & Running



```
[root@de-muc-anastasia01 ~]# # TODO: CHECK OR CHANGE ME
[root@de-muc-anastasia01 ~]# # extact and upload files
[root@de-muc-anastasia01 ~]# # definition: ${HOME} is the user's home folder on the remote server
[root@de-muc-anastasia01 ~]# # this path points to the root folder of the git repo for `mxnet`
[root@de-muc-anastasia01 ~]# export FS_PATH_MXNET_HOME="${HOME}/DSC/reports/aiToProd/models/mxnet/"
[root@de-muc-anastasia01 ~]#
[root@de-muc-anastasia01 ~]#
[root@de-muc-anastasia01 ~]# # deploy MXNet model DeepLearning, Docker Container w/ model path mapping
[root@de-muc-anastasia01 ~]# # deploy TF model Iris, Tensorflow Docker Container w/ model path mapping
[root@de-muc-anastasia01 ~]# # - bind to localhost (IPv4) resp. 127.0.0.1 network interface --
[root@de-muc-anastasia01 ~]# #     will only be accessible from server and through SSH tunnel;
[root@de-muc-anastasia01 ~]# #     not even firewall-relevant network traffic from outside needs to be managed for it
[root@de-muc-anastasia01 ~]# # - map the Model Server's config file and the therein referenced model both as read-only files inside a Volume
[root@de-muc-anastasia01 ~]# # - gets 256 MiB of RAM wo/ Swap and 10% of one CPU core
[root@de-muc-anastasia01 ~]# docker run \
>     -d \
>     --name daniel-mxnet-server-deeplearning \
>     -h 127.0.0.1 \
>     -p 2308:8080 \
>     -v ${FS_PATH_MXNET_HOME}/mms_app_cpu.conf:/mxnet_model_server/mms_app_cpu.conf:ro \
>     -v ${FS_PATH_MXNET_HOME}/squeezenet_v1.1.model:/mxnet_model_server/squeezenet_v1.1.model:ro \
>     --memory="256m" --memory-swap="256m" \
>     --cpus=".1" \
>     awsdeeplearningteam/mms_cpu mxnet-model-server start --mms-config /mxnet_model_server/mms_app_cpu.conf

# inspect Docker Volumes and Docker Containers 4 secs after starting --
# the last command is a comment for the Bash history to reveal it
# later on in case you need to step into the Container for debugging reasons
docker exec daniel-mxnet-server-deeplearning ls -alshR /mxnet_model_server/
sleep 4 && docker ps -a | grep daniel-mxnet-server-deeplearning && docker ps -a
# docker exec -it daniel-mxnet-server-deeplearning bash
4e4bfe5bc4631c26208c9bc6a9b5b7d45de40dae151b451bc7380a03797d2635
[root@de-muc-anastasia01 ~]#
[root@de-muc-anastasia01 ~]# # inspect Docker Volumes and Docker Containers 4 secs after starting --
[root@de-muc-anastasia01 ~]# # the last command is a comment for the Bash history to reveal it
[root@de-muc-anastasia01 ~]# # later on in case you need to step into the Container for debugging reasons
[root@de-muc-anastasia01 ~]# docker exec daniel-mxnet-server-deeplearning ls -alshR /mxnet_model_server/
/mxnet_model_server/:
total 4.5M
   0 drwxr-xr-x. 1 root root   35 Nov 24 20:56 .
   0 drwxr-xr-x. 1 root root   32 Nov 24 20:56 ..
4.0K -rw-r--r--. 1 root root  571 Nov 14 08:06 mms_app_cpu.conf
8.0K -rwxr-xr-x. 1 root root 6.0K Jul  5 22:32 mxnet-model-server
4.0K -rwxr-xr-x. 1 root root 1.6K Jul  5 22:32 setup_mms.py
4.4M -rw-r--r--. 1 root root 4.4M Nov 14 07:19 squeezenet_v1.1.model
4.0K -rwxr-xr-x. 1 root root 3.6K Jul  5 22:32 wsgi.py
[root@de-muc-anastasia01 ~]# sleep 4 && docker ps -a | grep daniel-mxnet-server-deeplearning && docker ps -a
4e4bfe5bc463        awsdeeplearningteam/mms_cpu   "mxnet-model-server …"   7 seconds ago       Up 5 seconds        0.0.0.0:2308->8080/tcp              daniel-mxnet-server-deeplearning
CONTAINER ID        IMAGE                         COMMAND                  CREATED             STATUS              PORTS                               NAMES
4e4bfe5bc463        awsdeeplearningteam/mms_cpu   "mxnet-model-server …"   7 seconds ago       Up 5 seconds        0.0.0.0:2308->8080/tcp              daniel-mxnet-server-deeplearning
0d13e3aa9171        tensorflow/serving            "/usr/bin/tf_serving…"   11 minutes ago      Up 11 minutes       8500/tcp, 0.0.0.0:2008->8501/tcp    daniel-tf-server-iris
dc6eb9cb401f        portainer/portainer           "/portainer"             10 days ago         Up 10 days          0.0.0.0:1206->9000/tcp              daniel-docker-daemon-ui
[root@de-muc-anastasia01 ~]# # docker exec -it daniel-mxnet-server-deeplearning bash
[root@de-muc-anastasia01 ~]# |
```

From the Client Machine, Our Pictures Reside in the Clone's Path of this Git Repository – Extracted & Cloned Images are in their Default Location

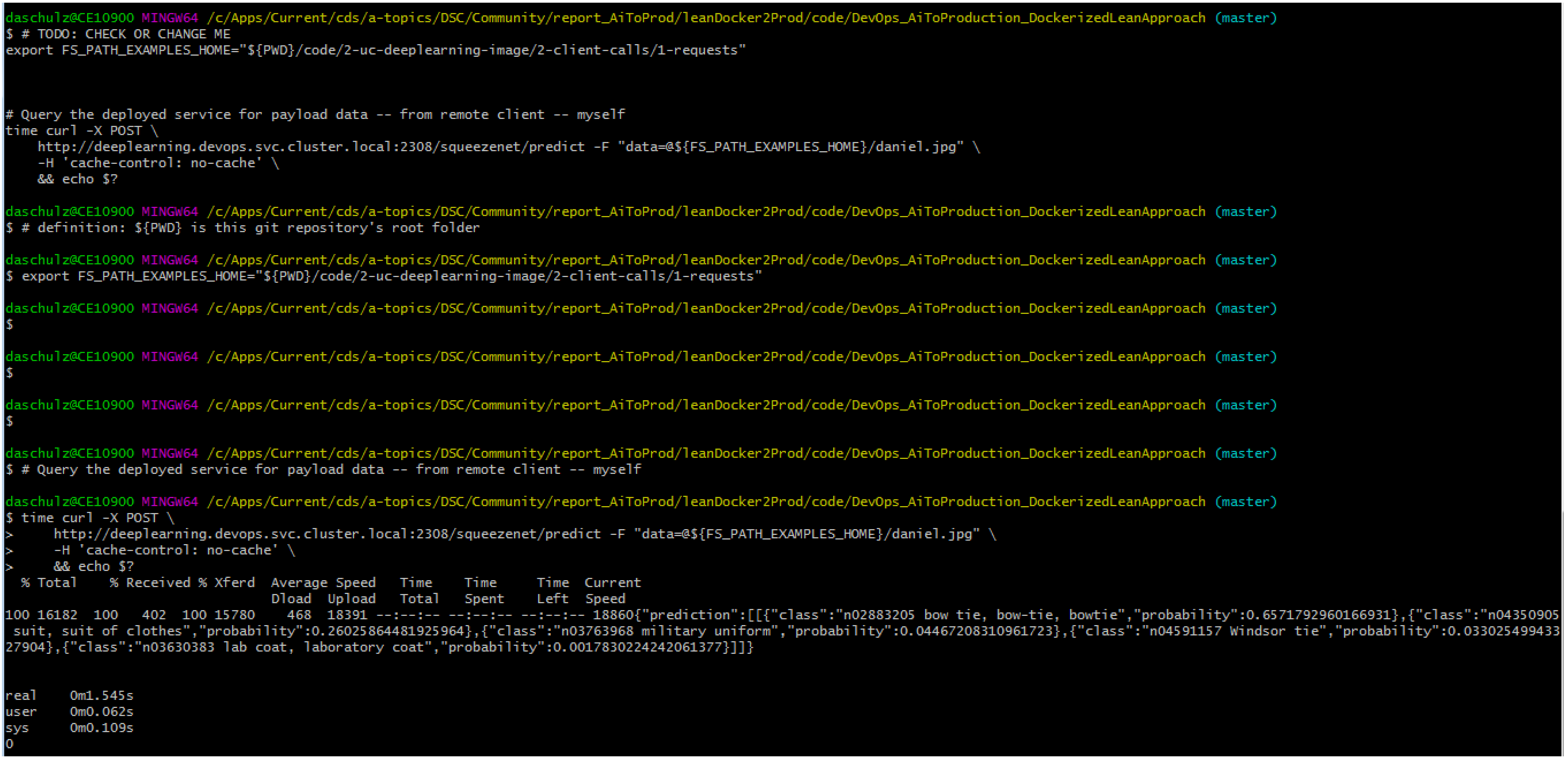
All Following Commands are Issued from the Git Repo's Root Folder



```
daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToP
rod/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ # TODO: CHECK OR CHANGE ME
export FS_PATH_EXAMPLES_HOME="${PWD}/code/2-uc-deeplearning-image/2-client-calls/1-requests"

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ # definition: ${PWD} is this git repository's root folder

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ export FS_PATH_EXAMPLES_HOME="${PWD}/code/2-uc-deeplearning-image/2-client-calls/1-requests"

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ ls -alsh ${FS_PATH_EXAMPLES_HOME}
total 417K
4.0K drwxr-xr-x 1 daschulz 1049089    0 Nov 24 19:50 ./
   0 drwxr-xr-x 1 daschulz 1049089    0 Nov 23 08:57 ../
1.0K -rw-r--r-- 1 daschulz 1049089  584 Nov 24 19:26 CHECKSUMS.sha256
4.0K -rwxr-xr-x 1 daschulz 1049089 1.9K Nov 24 19:50 client-requests.bash*
 84K -rw-r--r-- 1 daschulz 1049089  83K Nov 24 19:17 cute-chick-with-hairy-pussy.7z
 84K -rw-r--r-- 1 daschulz 1049089  83K Nov 24 18:37 cute-chick-with-hairy-pussy.jpg
 16K -rw-r--r-- 1 daschulz 1049089  16K Sep 30  2017 daniel.jpg
112K -rw-r--r-- 1 daschulz 1049089 109K Nov 24 19:17 kitten.7z
112K -rw-r--r-- 1 daschulz 1049089 109K Nov 14 07:56 kitten.jpg

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ ls -alsh
total 32K
4.0K drwxr-xr-x 1 daschulz 1049089    0 Nov 24 21:07 ./
   0 drwxr-xr-x 1 daschulz 1049089    0 Nov 23 08:03 ../
4.0K drwxr-xr-x 1 daschulz 1049089    0 Nov 24 21:58 .git/
4.0K -rw-r--r-- 1 daschulz 1049089 1.8K Nov 24 19:53 .gitignore
4.0K drwxr-xr-x 1 daschulz 1049089    0 Nov 24 21:59 .idea/
4.0K drwxr-xr-x 1 daschulz 1049089    0 Nov 23 09:05 code/
4.0K -rw-r--r-- 1 daschulz 1049089 1.1K Nov 23 08:03 LICENSE
4.0K -rw-r--r-- 1 daschulz 1049089 2.9K Nov 24 21:07 README.md
4.0K drwxr-xr-x 1 daschulz 1049089    0 Nov 24 21:12 resources/

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ pwd
/c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ |
```

From the Client Machine, the ReST Calls w/ the Daniel Picture as Payload Responds

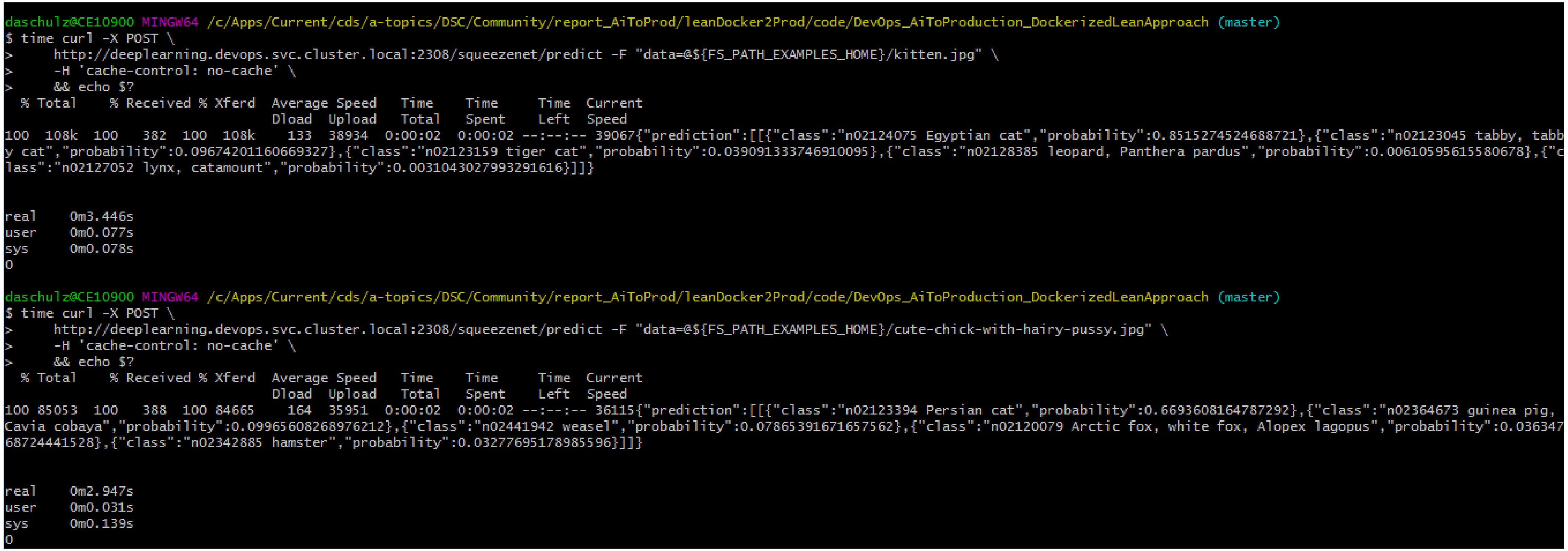
The Model Found Me Wearing a Tie & Suit, which is Correct



```
daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ # TODO: CHECK OR CHANGE ME
export FS_PATH_EXAMPLES_HOME="${PWD}/code/2-uc-deeplearning-image/2-client-calls/1-requests"



# Query the deployed service for payload data -- from remote client -- myself
time curl -X POST \
    http://deeplearning.devops.svc.cluster.local:2308/squeezenet/predict -F "data=@${FS_PATH_EXAMPLES_HOME}/daniel.jpg" \
    -H 'cache-control: no-cache' \
    && echo $?

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ # definition: ${PWD} is this git repository's root folder

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ export FS_PATH_EXAMPLES_HOME="${PWD}/code/2-uc-deeplearning-image/2-client-calls/1-requests"

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ # Query the deployed service for payload data -- from remote client -- myself

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ time curl -X POST \
>     http://deeplearning.devops.svc.cluster.local:2308/squeezenet/predict -F "data=@${FS_PATH_EXAMPLES_HOME}/daniel.jpg" \
>     -H 'cache-control: no-cache' \
>     && echo $?
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 16182  100   402  100 15780    468  18391 --:--:-- --:--:-- --:--:-- 18860{"prediction":[[{"class":"n02883205 bow tie, bow-tie, bowtie","probability":0.6571792960166931},{"class":"n04350905
 suit, suit of clothes","probability":0.26025864481925964},{"class":"n03763968 military uniform","probability":0.04467208310961723},{"class":"n04591157 Windsor tie","probability":0.033025499433
27904},{"class":"n03630383 lab coat, laboratory coat","probability":0.0017830224242061377}]]}


real    0m1.545s
user    0m0.062s
sys     0m0.109s
0
```

From the Client Machine, the ReST Calls w/ the Kitten Pictures as Payload Responds

The Model Found Cats in those Pictures, which is Correct



```
daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ time curl -X POST \
>     http://deeplearning.devops.svc.cluster.local:2308/squeezenet/predict -F "data=@${FS_PATH_EXAMPLES_HOME}/kitten.jpg" \
>     -H 'cache-control: no-cache' \
>     && echo $?
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  108k  100   382  100  108k    133  38934  0:00:02  0:00:02 --:--:-- 39067{"prediction":[[{"class":"n02124075 Egyptian cat","probability":0.8515274524688721},{"class":"n02123045 tabby, tabb
y cat","probability":0.09674201160669327},{"class":"n02123159 tiger cat","probability":0.039091333746910095},{"class":"n02128385 leopard, Panthera pardus","probability":0.006105956155580678},{"c
lass":"n02127052 lynx, catamount","probability":0.003104302799329161б}]]}
```

```


real    0m3.446s
user    0m0.077s
sys     0m0.078s
0

daschulz@CE10900 MINGW64 /c/Apps/Current/cds/a-topics/DSC/Community/report_AiToProd/leanDocker2Prod/code/DevOps_AiToProduction_DockerizedLeanApproach (master)
$ time curl -X POST \
>     http://deeplearning.devops.svc.cluster.local:2308/squeezenet/predict -F "data=@${FS_PATH_EXAMPLES_HOME}/cute-chick-with-hairy-pussy.jpg" \
>     -H 'cache-control: no-cache' \
>     && echo $?
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 85053  100   388  100 84665    164  35951  0:00:02  0:00:02 --:--:-- 36115{"prediction":[[{"class":"n02123394 Persian cat","probability":0.6693608164787292},{"class":"n02364673 guinea pig,
Cavia cobaya","probability":0.09965608268976212},{"class":"n02441942 weasel","probability":0.07865391671657562},{"class":"n02120079 Arctic fox, white fox, Alopex lagopus","probability":0.036347
68724441528},{"class":"n02342885 hamster","probability":0.03277695178985596}]]}


real    0m2.947s
user    0m0.031s
sys     0m0.139s
0
```

Topics Left Out

Next Up: Migrating to the Cloud,
Creating our own Docker Images,
CI/CD Pipelines,
etc.

One More Thing

“Simple TensorFlow Serving” for the Win

A liberal fork of TensorFlow Serving
is capable of deploying various kinds of model architectures:

- TensorFlow models (even from Keras)
- XGBoost
- MXNet
- SparkML
- CNTK
- PyTorch
- Caffe2
- H2O

etc.

github.com/tobegit3hub/simple_tensorflow_serving

People matter, results count.

About Capgemini

A global leader in consulting, technology services and digital transformation, Capgemini is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. It is a multicultural company of 200,000 team members in over 40 countries. The Group reported 2016 global revenues of EUR 12.5 billion.

Learn more about us at

www.capgemini.com