# Custom docker images for AML Pipelines

This is a basic guide with steps on how to create your custom docker image with your preferred libraries and use it for running your AML pipeline workloads.

## Prerequisites

- Docker Cli
- Azure Cli

### Create a docker file

- You can create your own docker file using one of the base AzureML images listed in Microsoft Container Registry(MCR) as reference, depending on your ML workload (CPU/GPU etc.). For this guide we will be using Azureml Base.
- We would need to install all the necessary drivers in our custom image that are not shipped with the mcr base image. In this specific example we want to write stuff to Azure SQL database from a python script, possibly using the pandas library so we need two packages and their associated drivers pyodbc for writing to DB and sqlalchemy to enable pandas database operations and simpler DB syntax instead of cursors. You dockerfile should should therefore look like this:

```
# pull base image from Microsoft Container Registry
FROM mcr.microsoft.com/azureml/base:latest
# apt-get and system utilities
RUN apt-get update && apt-get install -y \
    curl apt-transport-https debconf-utils \
    && rm -rf /var/lib/apt/lists/*
# adding custom MS repository
RUN curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -
RUN curl https://packages.microsoft.com/config/ubuntu/16.04/prod.list >
/etc/apt/sources.list.d/mssql-release.list
# install SQL Server drivers and tools
RUN apt-get update && ACCEPT EULA=Y apt-get install -y msodbcsql17 mssql-
tools
RUN echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
RUN /bin/bash -c "source ~/.bashrc"
RUN apt-get -y install locales
RUN locale-gen en US.UTF-8
RUN update-locale LANG=en_US.UTF-8
RUN apt-get install -y --reinstall build-essential
RUN apt-get install -y unixodbc-dev
```

```
# python stuff not shipped with base mcr image
RUN pip install pyodbc
RUN pip install sqlalchemy
CMD /bin/bash
```

- Place the dockerfile into its own folder. We will use this folder as the context path for the next section.
- Start your command shell in the folder with our dockerfile

#### Build container image

• We will use docker build to create a new image named amlodbc with tag 1.0 using the supplied dockerfile as input and using current directory as build context. Note the '.' at the end of the command, specifying current directory as the docker context path.

docker build -f dockerfile -t amlodbc:1.0 .

• To see a list of containers, use command docker images. You will notice the base mcr image that we pulled as well as our new amlodbc image:

C:\Users\ λ docker images	iscail -y ur	\aml_azsql_image		
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
amlodbc mcr.microsoft.com/azureml/base	1.0 latest	30387e0+4d2c 15b2e3b5af7a	2 minutes ago 4 months ago	1.07GB 1.01GB

## Push image to Azure Container Registry

Go to azure portal and navigate to the resource group that has your workspace. Along with your AML workspace and default storage accounts, you'll see an item of type "Container Registry". If there are more than one registry resources you can select any one to register your image to. Lets select acrodbcdemo here:

	Ŕ			×
Resource group	+ Add == Edit columns		Ĉ) Refresh → Move ↓ I	
Cverview	Subscription (change)		Deployments 7 Succeeded	
Activity log	Subscription ID			
<sup>8</sup> ∕ <sub>♀</sub> Access control (IAM)				
🔷 Tags	Tags (change)			
🗲 Events				
C-Mara			*	
Settings	Filter by name	Type == all 🔕	on == all 🔊 (+ 🕁 Add filter	
🍊 Quickstart				
	Showing 1 to 8 of 8 records.	Show hidden types 🛈		No grouping $\checkmark$
Policies	Name ↑↓		Type $\uparrow_{\downarrow}$	Location $\uparrow \downarrow$
😤 Properties	acrodbcdemo		Container registry	
🔒 Locks			Storage account	
🖳 Export template			Machine Learning	
Cost Management			Container registry	
🎄 Cost analysis			Application Insights	
Cost alerts (preview)			Key vault	
③ Budgets			Storage account	
Advisor recommendations			Storage account	
Monitoring				

• Note the login server name from the registry details. It is usually of the form

[registry\_name].azurecr.io. In our case it will be acrodbcdemo.azurecr.io as can be seen in the below screenshot:

← acrodbcdemo ☆ Container registry			×
∕ာ Şearch (Ctrl+/) ≪	ightarrow Move 🛍 Delete 🍪 Update		1
Overview	Resource group (change)	Copy to clipboard	
Activity log		acrodbcdemo.azurecr.io	
<sup>8</sup> ∕ <sub>A</sub> Access control (IAM)	Location	Creation date	
🔷 Tags	Subscription (change)	sku	
🍊 Quick start		Standard	
🗲 Events	Subscription ID	Provisioning state Succeeded	
	^	:	

• Now we use docker tag to add ACR tag to our image:

docker tag amlodbc:1.0 acrodbcdemo.azurecr.io/amlodbc:1.0

• If you run docker images again you will see a new entry. Note how it has the same image id as amlodbc:

C:\Users\		\aml_azso	l_image		Nami_azs(
λ docker images					
REPOSITORY	TAG		IMAGE ID	CREATED	SIZE
acrodbcdemo.azurecr.io/amlodbc	1.0		30387e0f4d2c	42 minutes ago	1.07GB
amlodbc	1.0		30387e0f4d2c	42 minutes ago	1.07GB
mcr.microsoft.com/azureml/base	latest		15b2e3b5af7a	4 months ago	1.01GB

• Next, we log into our container registry using az acr login

```
az acr login --name acrodbcdemo
```

• Now we push to ACR using docker push

docker push acrodbcdemo.azurecr.io/amlodbc:1.0

• If everything went well, you should now see amlodbc appear as a new repository in azure portal:

acrodbcc     Container registry	d <b>emo</b>   Repo	ositories	×
✓ Search (Ctrl+/)		🕐 Refresh	
left Overview	<b>A</b>	- β Search to filter repositories	
Activity log	_	Repositories ↑↓	
Access control (IAN	Л)	amlodbe	
🔷 Tags	_		
🍊 Quick start	_		
🗲 Events	_		
Settings			
📍 Access keys			
Encryption	_		
🐍 Identity	_		
👸 Networking	_		
🔒 Locks	_		
🖳 Export template	_		
Services			
Repositories			

Use the custom docker image in your AML pipeline

• Make sure your private ACR has 'Admin user' enabled under its Access keys section for this method to work

<b>acrodbcdemo</b>   Acce Container registry	ss keys	> -	
P Search (Ctrl+/) 《			
Sverview	Registry name	Copy to clipboard	
Activity log	acrodbcdemo		
දි Access control (IAM)	Login server		
🔷 Tags	acrodbcdemo.azurecr.io		
🗳 Quick start			
🗲 Events	Admin user ①		
Settings	Enable Disable		
📍 Access keys	Username		
Encryption	acrodbcdemo		
🐍 Identity			
😵 Networking	Name Password		
🔒 Locks	password	C) 🚺	
Export template	password2	() ( <u>)</u>	

• Your custom docker image can be specified as a ContainerRegistry and passed to your pipeline step.

```
# Use a RunConfiguration to specify some additional requirements for this
step.
from azureml.core.runconfig import RunConfiguration
# from azureml.core.conda_dependencies import CondaDependencies
from azureml.core.container_registry import ContainerRegistry
# create a new runconfig object
run_config = RunConfiguration()
# enable Docker
run_config.environment.docker.enabled = True
# you can also point to an image in a private ACR
image_registry_details = ContainerRegistry()
image_registry_details.address = "acrodbcdemo.azurecr.io"
image_registry_details.username = "acrodbcdemo" # username is same as
registry name
image registry details.password = "<password>" # use any one of the two
passwords shown in portal
run_config.environment.docker.base_image_registry = image_registry_details
# this is an image in the image_registry
image_name = 'amlodbc:2.0'
run_config.environment.docker.base_image = image_name
# don't let the system build a new conda environment
run_config.environment.python.user_managed_dependencies = True
# specify CondaDependencies obj
# run_config.environment.python.conda_dependencies =
CondaDependencies.create(conda packages=['pyodbc','sqlalchemy'])
```

## **Further Reading**

- Dockerfile reference
- Microsoft ODBC driver installation guide for Linux
- Microsoft Container Registry github
- AML containers repository
- Estimator demo with custom docker images
- Distributed CNTK using custom docker images