Quickstart container in the Azure cloud

To be able to run this quickstart, you need to have the following components:

- On your laptop:
  - Docker (for Windows or Mac) https://www.docker.com/products/docker-desktop
  - Azure CLI https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest
- In the cloud: access to an Azure cloud subscription. You can create a free account at https://azure.microsoft.com/en-us/free/

Using public Docker images

We start with using public available images on Docker Hub and see the difference between running it locally and the same in the cloud.

We will use 2 images: MySQL and PHPMyAdmin. The latter image is a web application which is an administration tool for MySQL (or MariaDB) which is used by many web hosting providers.

Local

Open a command prompt / terminal window and type:

```
docker run -d --name db1 -p 3306:3306 -e MYSQL_ROOT_PASSWORD=r00tr00t -e MYSQL_DATABASE=db -e MYSQL_USER=db_user -e MYSQL_PASSWORD=dbpwd123 mysql
```

This will pull the MySQL (latest) image from Docker Hub and create a MySQL Docker container with a database called `db1` and set a root password and create an additional user `db_user`.

Next create the container with the administration tool by typing:

```
docker run -d --name mydbadmin --link db1:db -p 8080:80 phpmyadmin/phpmyadmin
```

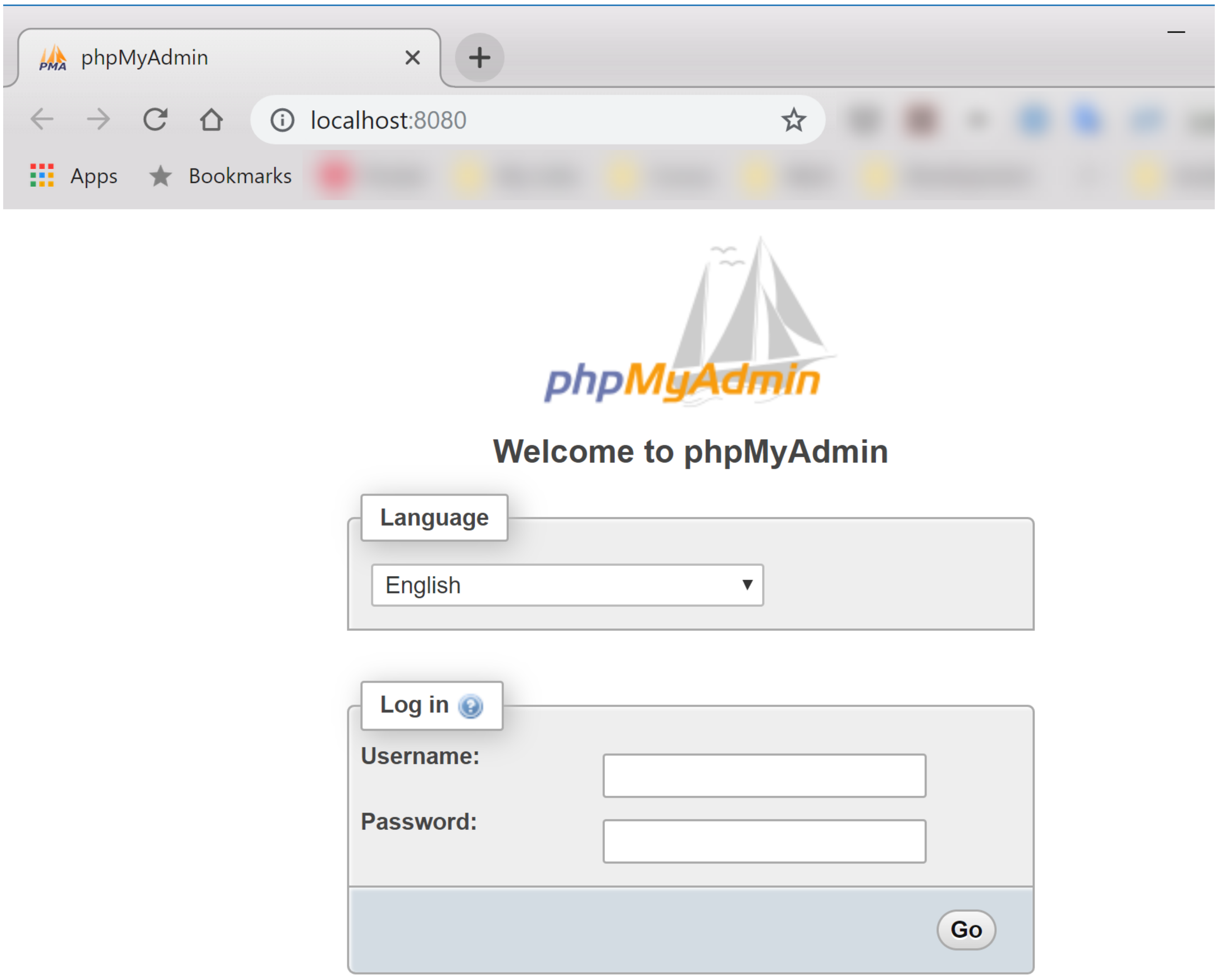
The `--link` command will give the container access to the other container.

For more information about both Docker images, see their Docker Hub pages:

- https://hub.docker.com/_/mysql
- https://hub.docker.com/r/phpmyadmin/phpmyadmin

Check if both containers are running: `docker ps`

Now open a web browser and go to http://localhost:8080



Try to login as db_user.

You probably get some errors about the authentication method used. This has to do with a change in MySQL v8 which we are using. We can solve this by updating the password in the MySQL database.

On the command prompt, type:

```
docker exec -it db1 bash
```

Now you are logged in the MySQL container.

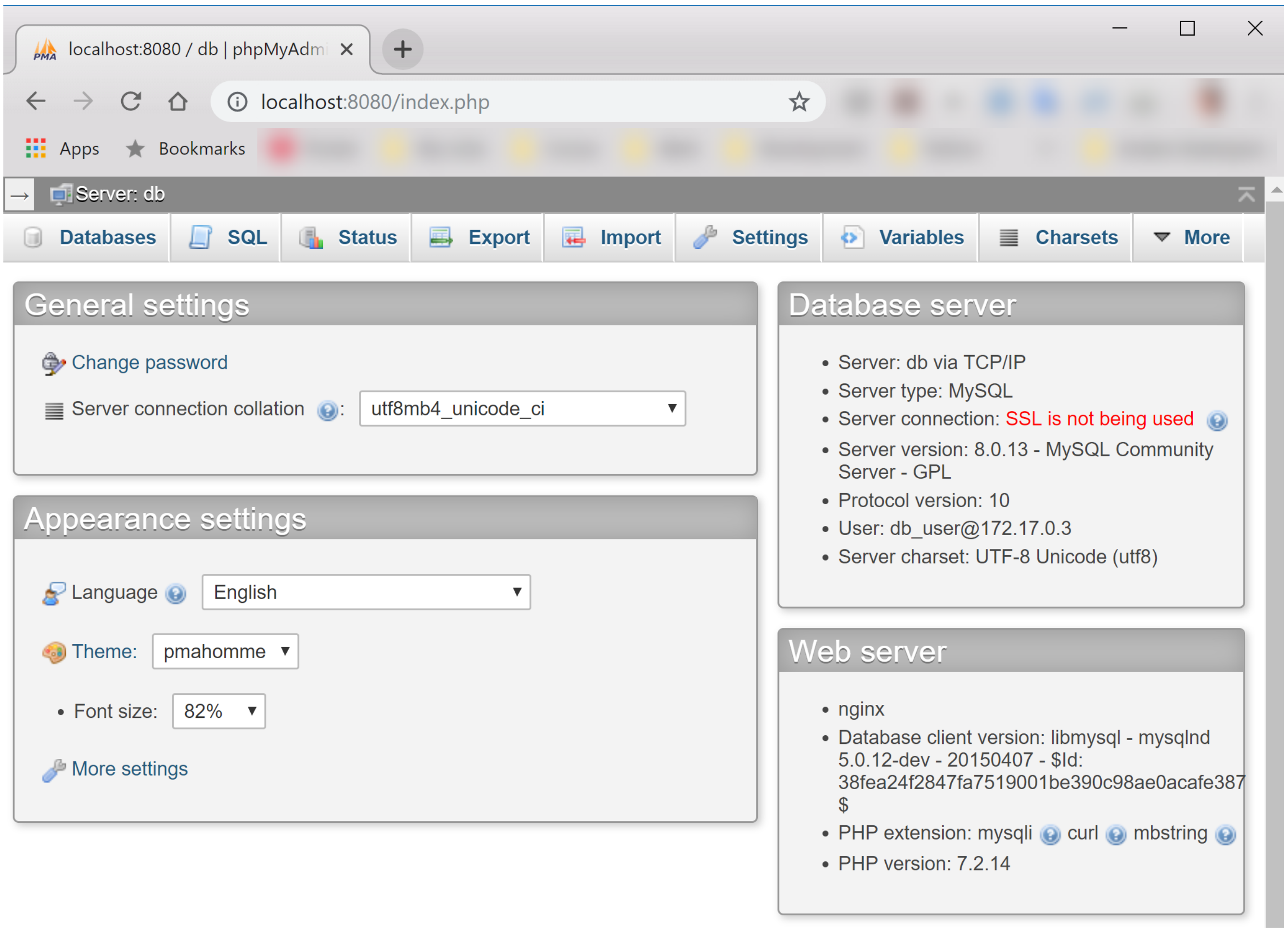
Type: `mysql -u root -pr00tr00t` (use the same password for root as used when you created the container).

Now update the password for the `db_user` user:

```
alter user db_user identified with mysql_native_password by 'secretuserpassword';
```

Type: `exit` and `exit` again to return to your local command prompt.

Now try again to login.



localhost:8080 / db | phpMyAdm

localhost:8080/index.php

Server: db

General settings

Change password

Server connection collation: utf8mb4_unicode_ci

Appearance settings

Language: English

Theme: pmahomme

- Font size: 82%

More settings

Database server

- Server: db via TCP/IP
- Server type: MySQL
- Server connection: SSL is not being used
- Server version: 8.0.13 - MySQL Community Server - GPL
- Protocol version: 10
- User: db_user@172.17.0.3
- Server charset: UTF-8 Unicode (utf8)

Web server

- nginx
- Database client version: libmysql - mysqlnd 5.0.12-dev - 20150407 - $Id: 38fea24f2847fa7519001be390c98ae0acafe387 $
- PHP extension: mysqli curl mbstring
- PHP version: 7.2.14

If you go to the Databases tab you will see the `db` database without any tables.

Ok. That is working.

Now we try the same in the Azure cloud.

Azure cloud

There are multiple ways to create containers in Azure. We will start with using the Azure Portal.

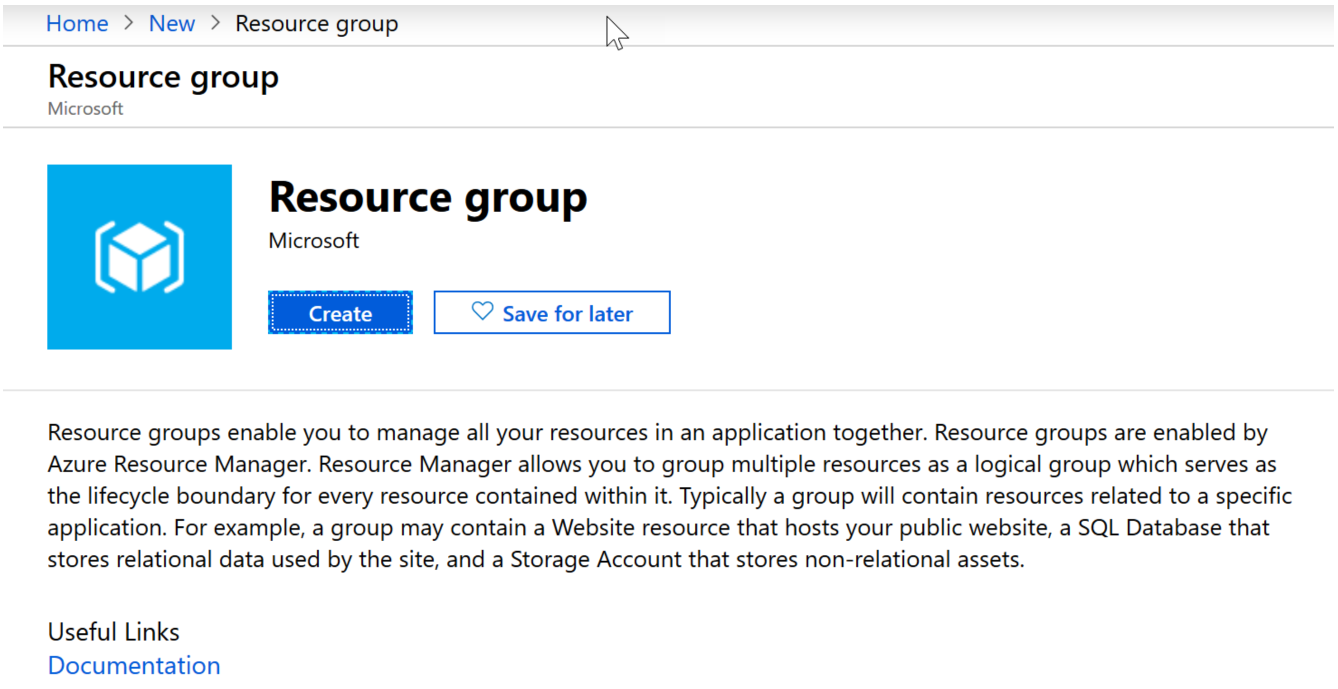
Azure Portal

In your browser go to https://portal.azure.com and login into your Azure subscription.

First we will create a new resource group in which we will add our container. Creating a resource group makes it easier to find your components but also to clean it up.

Press the `Create a resource` link in the left sidebar.

Search for `Resource Group` .


Home > New > Resource group

Resource group
Microsoft

Resource group
Microsoft

Resource groups enable you to manage all your resources in an application together. Resource groups are enabled by Azure Resource Manager. Resource Manager allows you to group multiple resources as a logical group which serves as the lifecycle boundary for every resource contained within it. Typically a group will contain resources related to a specific application. For example, a group may contain a Website resource that hosts your public website, a SQL Database that stores relational data used by the site, and a Storage Account that stores non-relational assets.

Useful Links
Documentation

Press the `Create` button.

Select a subscription and enter a name for the new resource group, e.g. `containertest`

Select a region, e.g.: `West Europe`.

Press `Review + Create` and then `Create`.

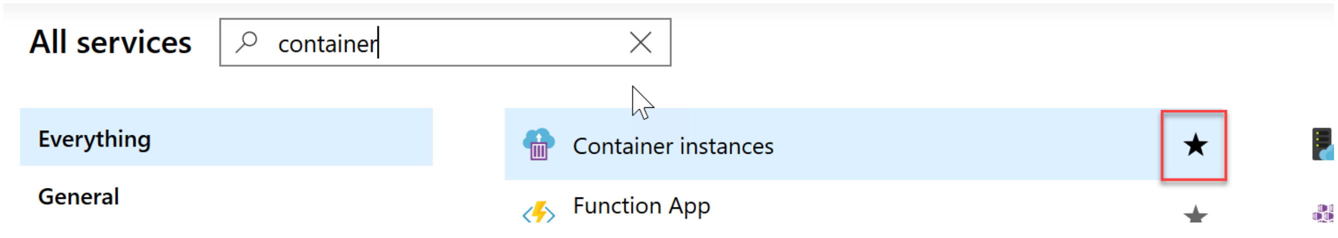
Resource groups are just a grouping of Azure components, there are no costs involved, so you can create as many resource groups as you like (if you are allowed to do so).

Now we will add the MySQL container.

In Azure single Docker containers are created in a component called Azure Container Instance (which is a wrapper for one or more containers). For more information, see the Azure documentation.

Lets create the container.

Press the `Create a resource` link again and search for `container instance` or let's do it another way. Because you can navigate to the home tab of all your container instances by pressing the `All Services` link in the side bar and search for `container instances`. If you click on the star next to the Container instances services, the link to the home tab will be added to the side bar.



Now press the `Container instance` link and the home tab for all container instances will be shown.

Now press the `Add` button.

On the first page, select the previous created resource group. And enter a name for the container.

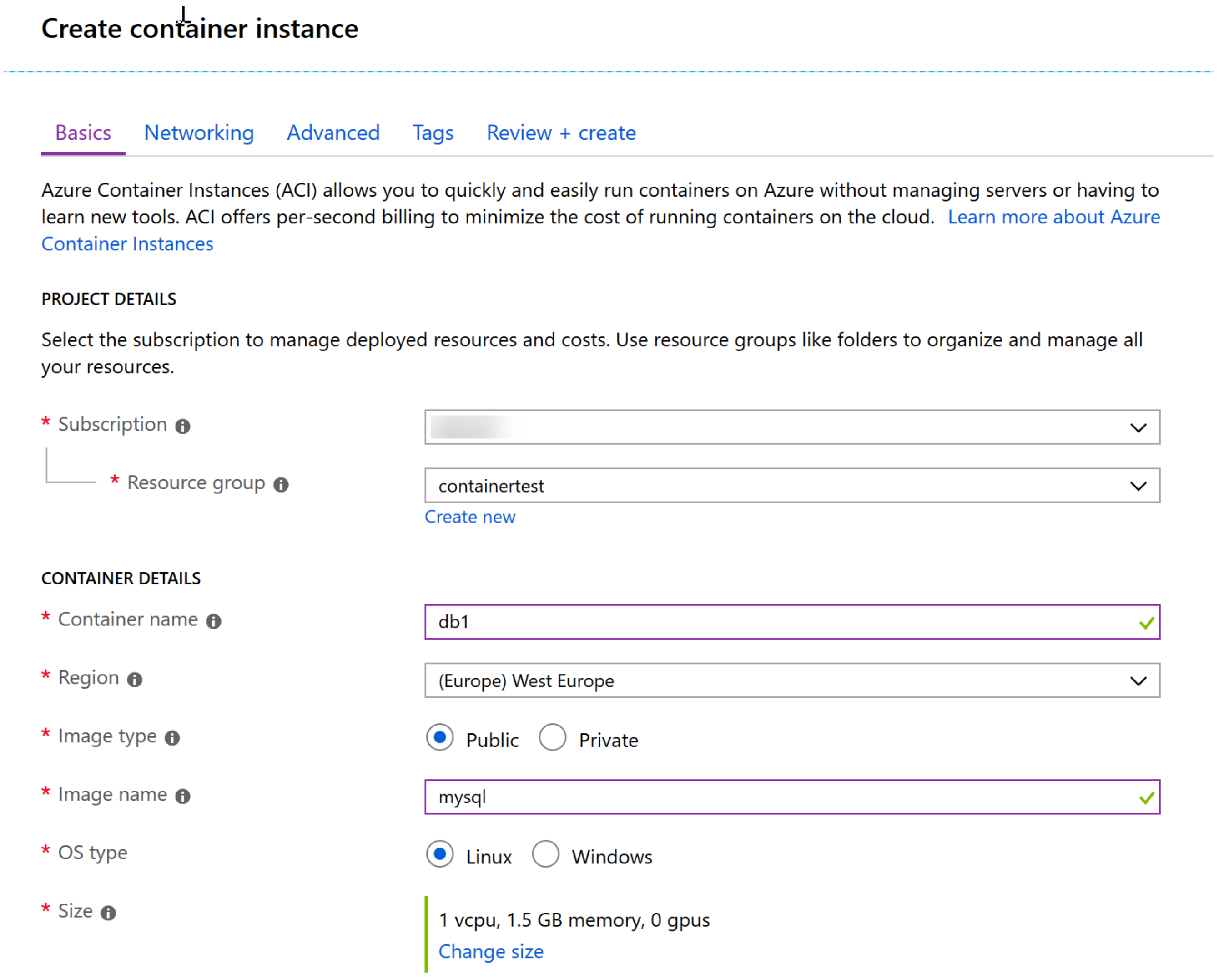
Select the region (which is probably already set correctly).

We will use a public image, so the Image Type is Public.

Enter the image name: `mysql`.

Set OS Type to Linux.

And leave the size to 1 cpu and 1.5 GB.



Create container instance

Azure Container Instances (ACI) allows you to quickly and easily run containers on Azure without managing servers or having to learn new tools. ACI offers per-second billing to minimize the cost of running containers on the cloud. Learn more about Azure Container Instances

PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription	
* Resource group	containertest
	Create new

CONTAINER DETAILS

* Container name	db1
* Region	(Europe) West Europe
* Image type	[✓ Public] Private
* Image name	mysql
* OS type	[✓ Linux] Windows
* Size	1 vcpu, 1.5 GB memory, 0 gpus Change size

Press Next.

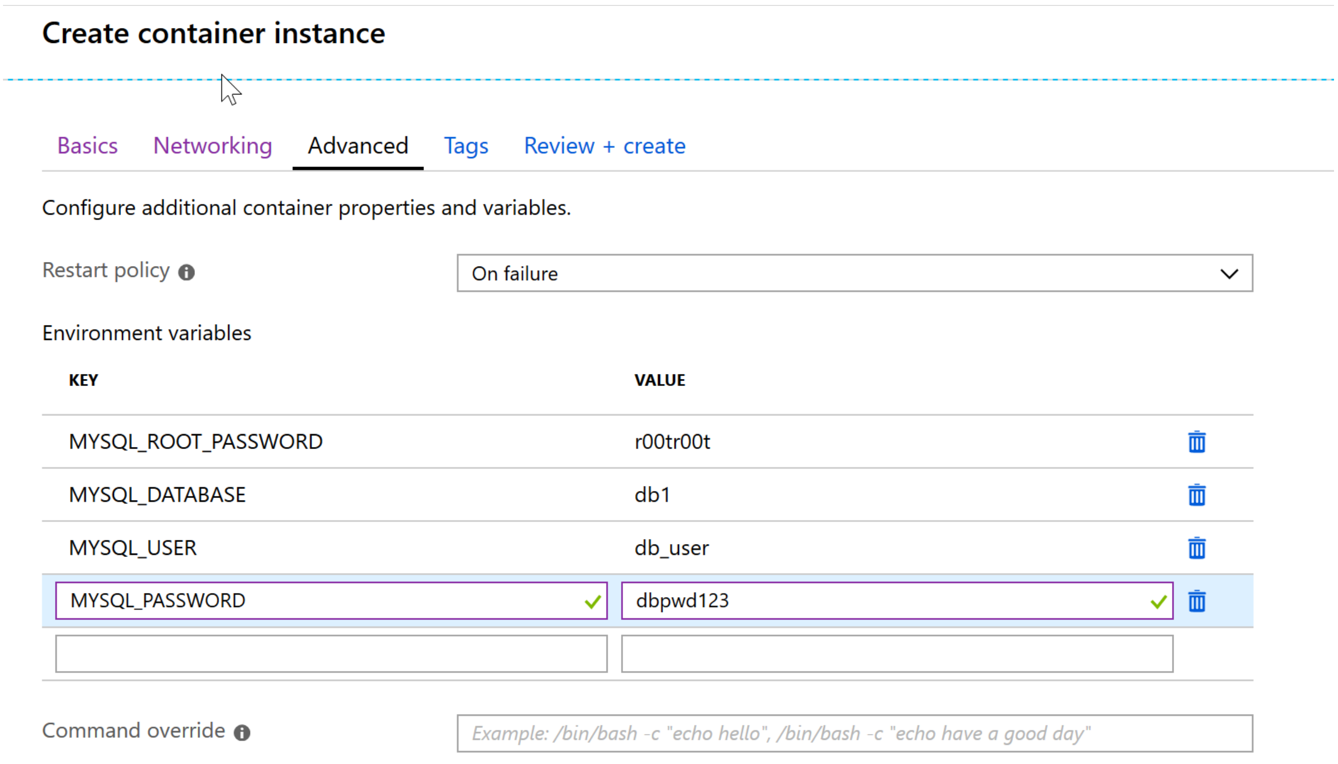
Set Include public IP adress to Yes.

Change the default port 80 to 3306.

Set the DNS name label to something (global) unique:

Press Next.

On the next page enter the same environment variables you used when creating the container locally.



Create container instance

Basics | Networking | Advanced | Tags | Review + create

Configure additional container properties and variables.

Restart policy: On failure

Environment variables

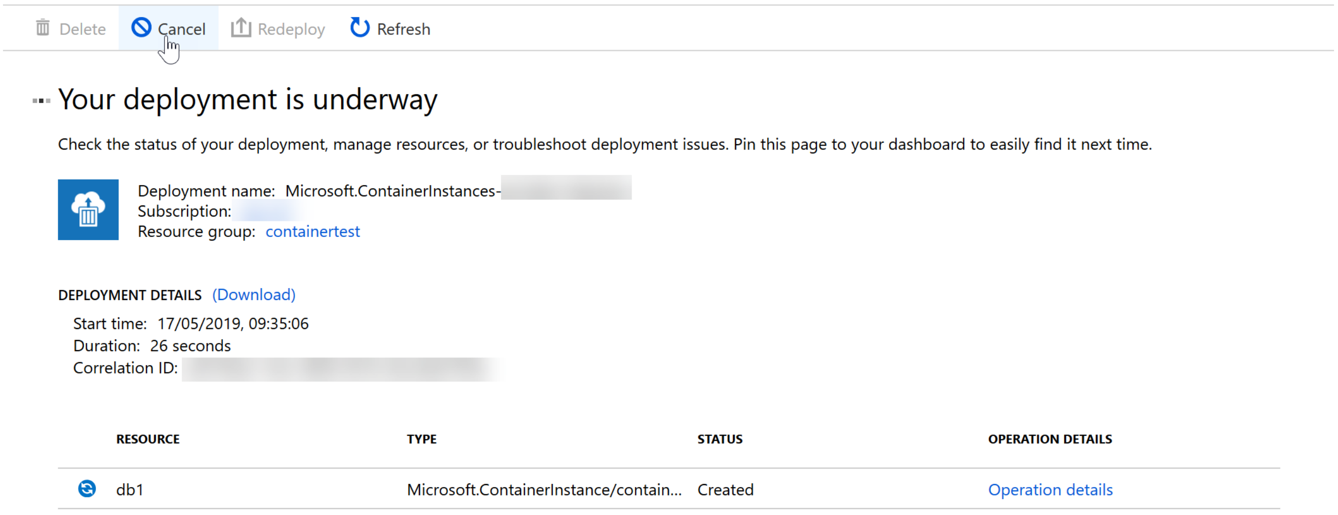
KEY	VALUE
MYSQL_ROOT_PASSWORD	r00tr00t
MYSQL_DATABASE	db1
MYSQL_USER	db_user
MYSQL_PASSWORD	dbpwd123

Command override: *Example: /bin/bash -c "echo hello", /bin/bash -c "echo have a good day"*

Press Review+Create.

Press Create.

The container will now be created.



Your deployment is underway

Check the status of your deployment, manage resources, or troubleshoot deployment issues. Pin this page to your dashboard to easily find it next time.

Deployment name: Microsoft.ContainerInstances-
Subscription:
Resource group: containertest

DEPLOYMENT DETAILS (Download)

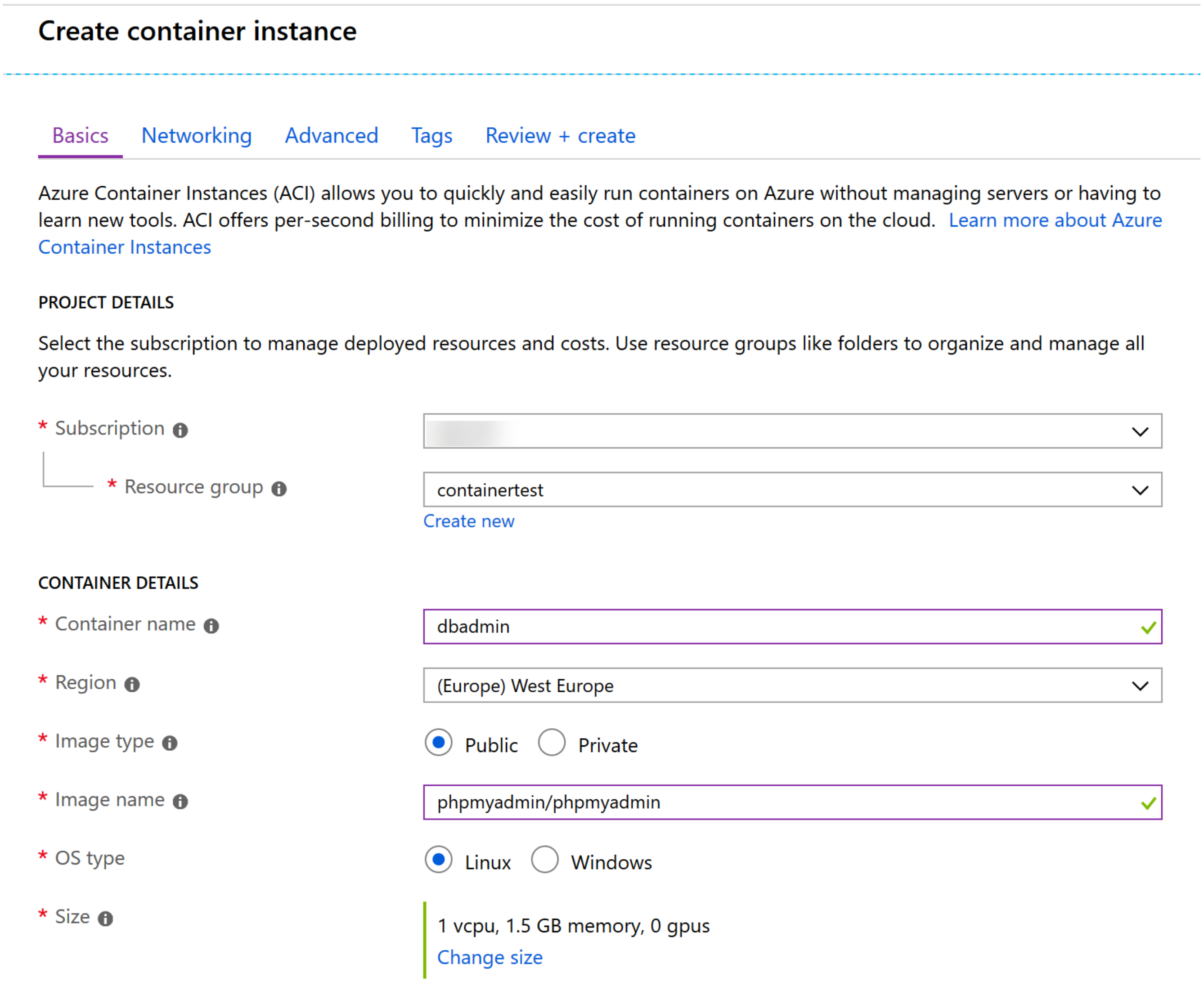
Start time: 17/05/2019, 09:35:06
Duration: 26 seconds
Correlation ID:

RESOURCE	TYPE	STATUS	OPERATION DETAILS
db1	Microsoft.ContainerInstance/contain...	Created	Operation details

When ready click on `Go to resource`.

Make a note of the FQDN (the DNS name the container can be accessed on). We need that for our second container.

Now add another container for PHPMyAdmin.


Create container instance

Basics | Networking | Advanced | Tags | Review + create

Azure Container Instances (ACI) allows you to quickly and easily run containers on Azure without managing servers or having to learn new tools. ACI offers per-second billing to minimize the cost of running containers on the cloud. Learn more about Azure Container Instances

PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription	
* Resource group	containertest Create new

CONTAINER DETAILS

* Container name	dbadmin ✓
* Region	(Europe) West Europe
* Image type	[✓ Public] Private
* Image name	phpmyadmin/phpmyadmin ✓
* OS type	[✓ Linux] Windows
* Size	1 vcpu, 1.5 GB memory, 0 gpus Change size

On the networking tab leave the port tot 80 (currently it is not possible to map a container port to another port as we do locally).

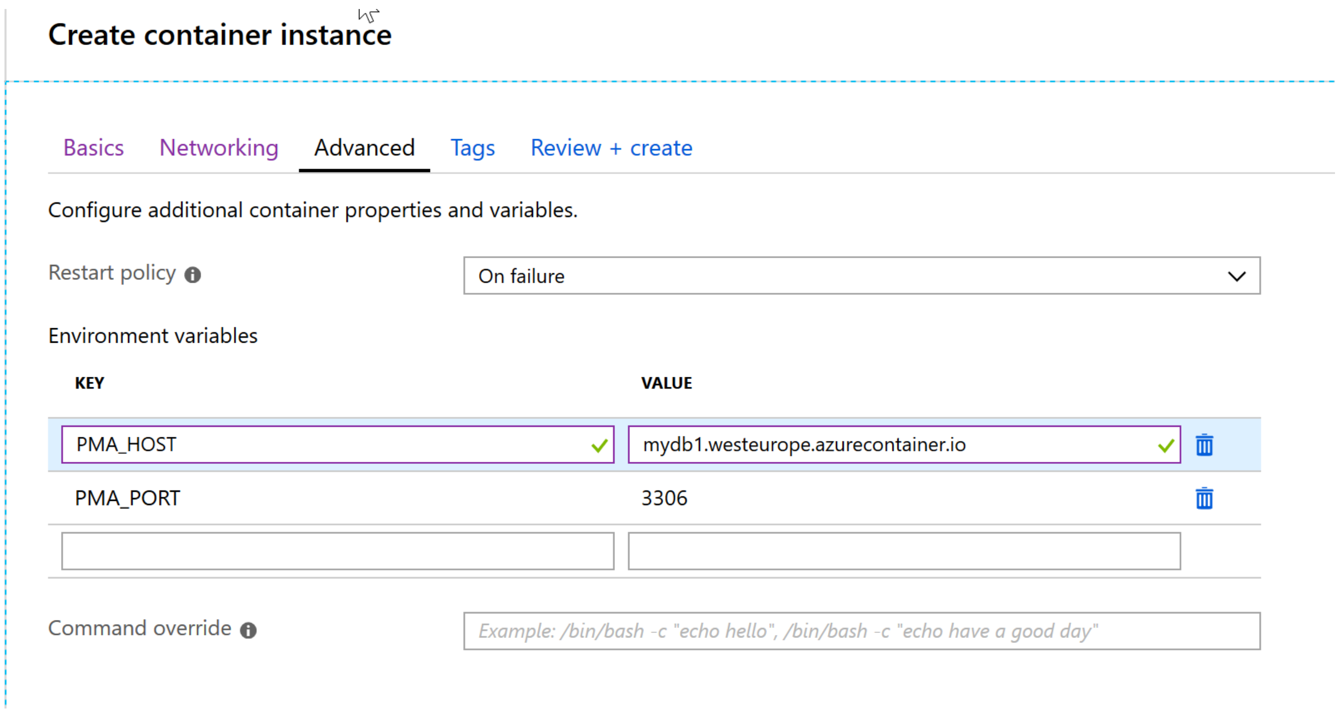
Set a DNS name label.

Press Next.

On the advanced tab, add two environment settings:

PMA_HOST =

PMA_PORT = 3306


Create container instance

Configure additional container properties and variables.

Restart policy: On failure

Environment variables

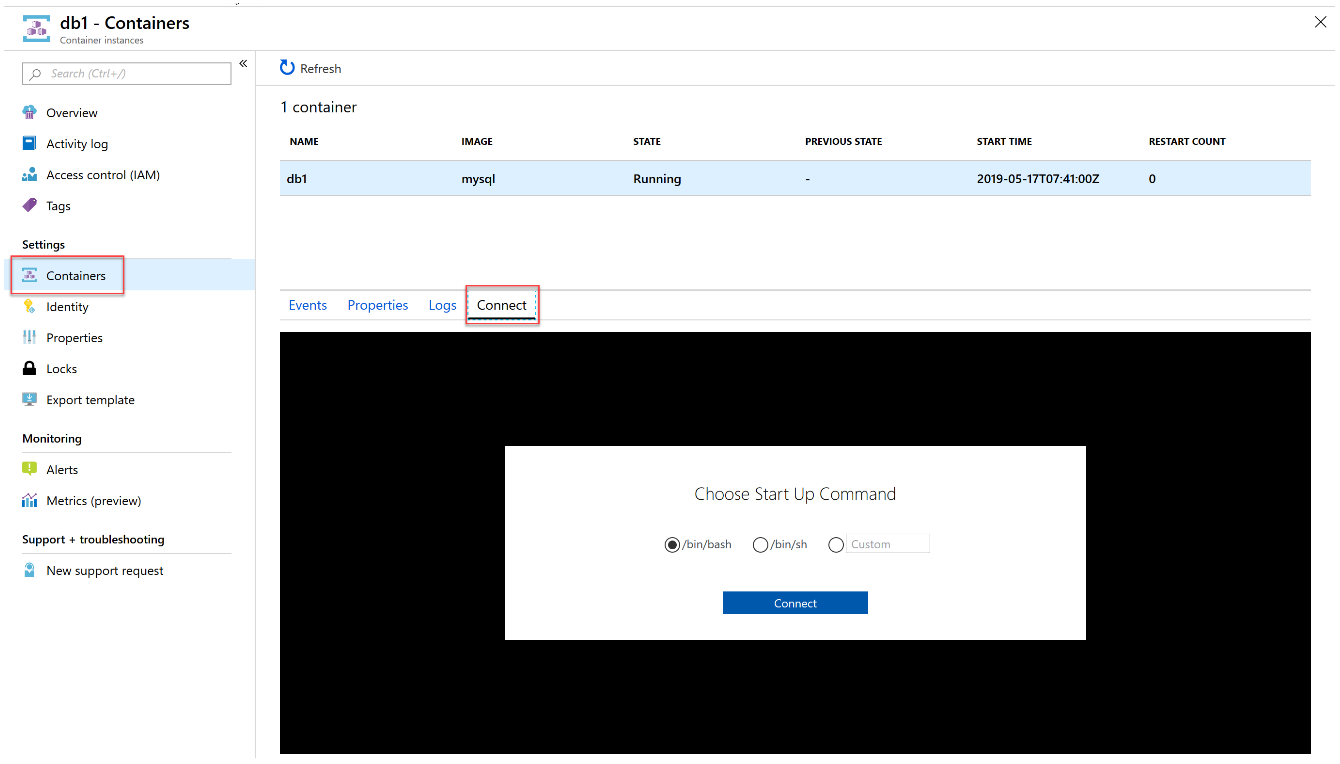
KEY	VALUE
PMA_HOST	mydb1.westeurope.azurecontainer.io
PMA_PORT	3306

Command override: Example: /bin/bash -c "echo hello", /bin/bash -c "echo have a good day"

And create the container.

Also here we will have the same issue with authentication.

Go to the db1 container and open the `Container` subpage and then the `Connect` tab.


db1 - Containers
Container instances

1 container

NAME	IMAGE	STATE	PREVIOUS STATE	START TIME	RESTART COUNT
db1	mysql	Running	-	2019-05-17T07:41:00Z	0

Choose Start Up Command

[✓ /bin/bash] /bin/sh Custom

Select /bin/bash and press Connect.

And perform the same steps as you have locally to update the `db_user` password.

And log into PHPMyAdmin. You can find the url on the overview page of the dbadmin container.

And check if you can log in.

Now if you really want to use the database, you will need some persistent storage. Otherwise any change you make to the current running container instance will be lost at restart time of the container. Note that this is different from running it locally. On your local machine the data will remain as long as the container has not been deleted.

Unfortunately you can not add volumes in the Azure portal like you do locally. But it can be done using the Azure CLI.

If not done already install Azure CLI on your machine (see the prerequisites).

Azure CLI

Azure CLI is a set of commands you can use on the command prompt / terminal. On Windows you can use either command prompt, powershell or WSL (Windows Subsystem for Linux) for this.

The first step is to connect your machine to your Azure account.

Open the command prompt / terminal and type:

`az login`

A browser window will appear and you will need to log into your Azure account. If you have done this, you are ready to go.

We already created the resource group via the portal, but if you wish to create a resource group via Azure CLI, use the following command:

`az group create --name containertest --location westeurope`

Next we create a storage account to store our files for the mysql container:

`az storage account create --name sadb001files --resource-group containertest --sku Standard_LRS` (use any name you like, but only lowercase alphanumeric characters are allowed).

Next create a file share on this account which we will use a a volume mount for the container. You need the connection string to the created storage account for this. You can either find it in the azure portal or use the following azure cli command: `az storage account show-connection-string --name sadbfiles --resource-group containertest -o tsv`.

`az storage share create --name dbfileshare --connection-string "..."` (note that the connection string must be enclosed by quotes).

For the volume mount you need the account key for the storage. Use the following azure cli command to retrieve this: `az storage account keys list --resource-group containertest --account-name sadbfiles --query "[0].value" --output tsv`.

Next we will create the MySQL container:

```
az container create --resource-group containertest --location westeurope --name db2 --image
mysql --ip-address public --port 3306 --cpu 1 --memory 1.5 --dns-name-label db2 --
environment-variables MYSQL_ROOT_PASSWORD=secretpasspass MYSQL_DATABASE=db
MYSQL_USER=db_user MYSQL_PASSWORD=dbpwd123 --azure-file-volume-account-name sadbfiles --
azure-file-volume-account-key "..." --azure-file-volume-share-name dbfileshare --azure-
file-volume-mount-path /var/lib/mysql
```

And create the PHPMyAdmin container:

```
az container create --resource-group containertest --location westeurope --name dbadmin2 --
image phpmyadmin/phpmyadmin --ip-address public --port 80 --cpu 1 --memory 1.5 --dns-name-
label dbadmin2 --environment-variables PMA_HOST=db2.westeurope.azurecontainer.io
PMA_PORT=3306
```

Also here: update password for the MySQL user and then try to connect using the new PHPMyAdmin container.

Create a table and add some data.

Now stop and start the mysql container and check if the data is still available.

And do the same with the mysql container you created using the portal.

You should see that the second created container still has its data and the first one does not.

> **There is one major note to mention: the container are now running in the cloud and can be accessed over the internet, so if you want to run these containers for a longer period of time you MUST add security measurements!**