

Sales Engineer Technical Exercise

Table of Contents

[1. Prerequisites and Installation 3](#_Toc10396210)

[1.1. My Datadog Dashboard: 3](#_Toc10396211)

[1.2. Docker 3](#_Toc10396212)

[1.3. Networking: 3](#_Toc10396213)

[1.4. Docker-Compose: 4](#_Toc10396214)

[1.5. Docker-compose and Azure Container cloud: 5](#_Toc10396215)

[1.5.1. Build custom Image and upload In the container registry 5](#_Toc10396216)

[1.5.2. Create a container assembly based on the Image: 7](#_Toc10396217)

[2. Collecting Metrics: 8](#_Toc10396218)

[2.1. Azure Integration: 8](#_Toc10396219)

[2.2. Agent TAGS: 8](#_Toc10396220)

[2.3. Custom Metrics: 9](#_Toc10396221)

[2.4. MySQL Metrics: 9](#_Toc10396222)

[2.5. Update from the 02nd of June 10](#_Toc10396223)

[I will Investigate this further In a second workload. 10](#_Toc10396224)

[3. Visualizing Data: 11](#_Toc10396225)

[4. Monitoring Data: 12](#_Toc10396226)

[5. Collecting Data: 13](#_Toc10396227)

[6. Final Questions: 14](#_Toc10396228)

[7. Load Testing and APP: 15](#_Toc10396229)

[8. Links and references 16](#_Toc10396230)

[9. First Installation/Configuration: 17](#_Toc10396231)

[9.1. Datadog Agent 17](#_Toc10396232)

[9.1. MySQL Server 18](#_Toc10396233)

[9.2. Datadog agent Integration with MySQL: 19](#_Toc10396234)

# Prerequisites and Installation

I decided to use Azure cloud to build this demo set so that I would not rely on any laptop or machine to show or run this environment.

Also I might use It for other training or demo purposes, I only pay when switched on.

## My Datadog Dashboard:

Check [Dashboard](https://app.datadoghq.com/dash/host/1038846355) to verify the host registration In Datadog:

(https://app.datadoghq.com/dash/host/1038846355)

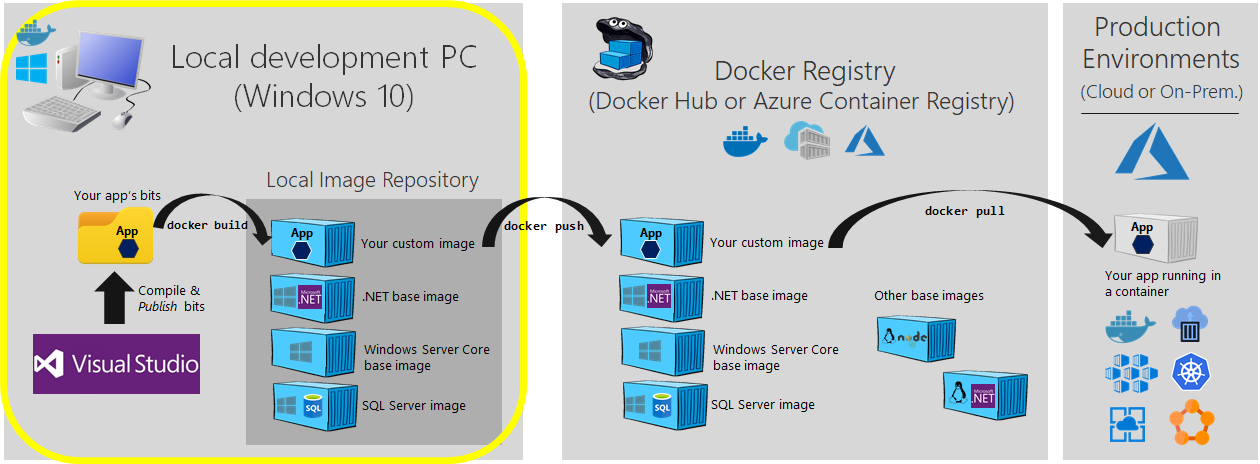
## Docker

Docker Is required to run the Datadog Agent to monitor

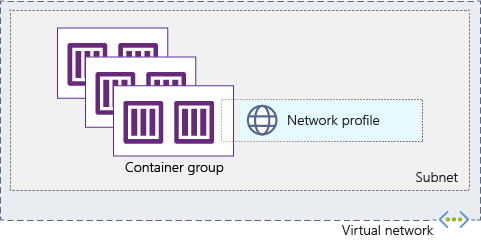
By creating an Azure Docker with a Public Repository Image from GitHub one can alleviate the need to Install an OS and docker and maintain It.

A very Important read:

<https://docs.datadoghq.com/integrations/faq/compose-and-the-datadog-agent/>



## Networking:



All as the communication goes from the agent to Datadog SaaS and not the other way around

## Docker-Compose:

After our first setup/attempt, I had to reverse and put everything Into one container using docker In order to have the datadog agent to communicate with the MySQL container.

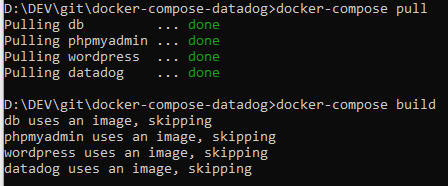
See the last chapter of this document for the details as to why.

I do test and validate everything first with Docker compose on my laptop Docker for Windows and everything works well:

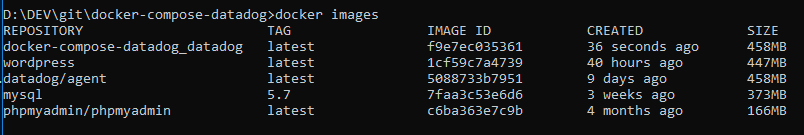
*docker-compose config first will validate you yaml file*

*docker-compose pull*

*docker-compose build*



docker images



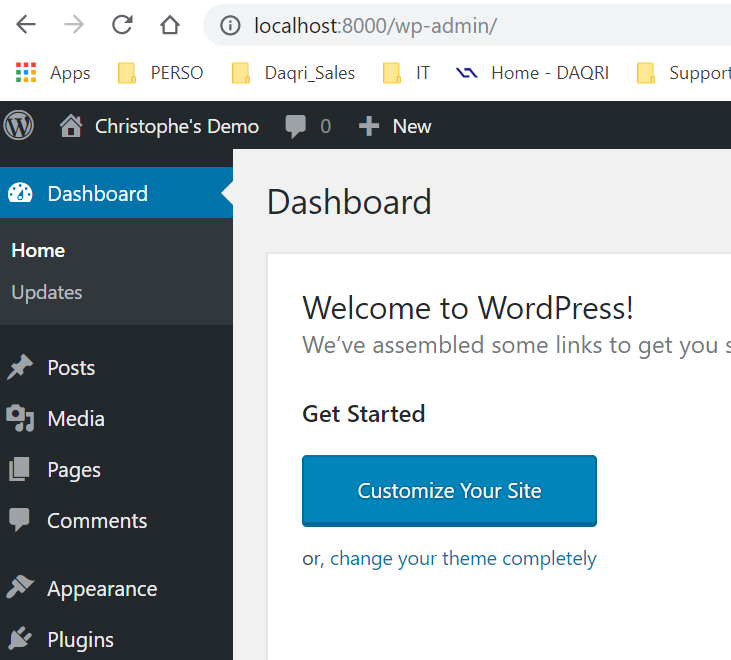
Environment variables are not working well if no attention.

this will provide a detailed json structure were one can inspect variables beforehand:

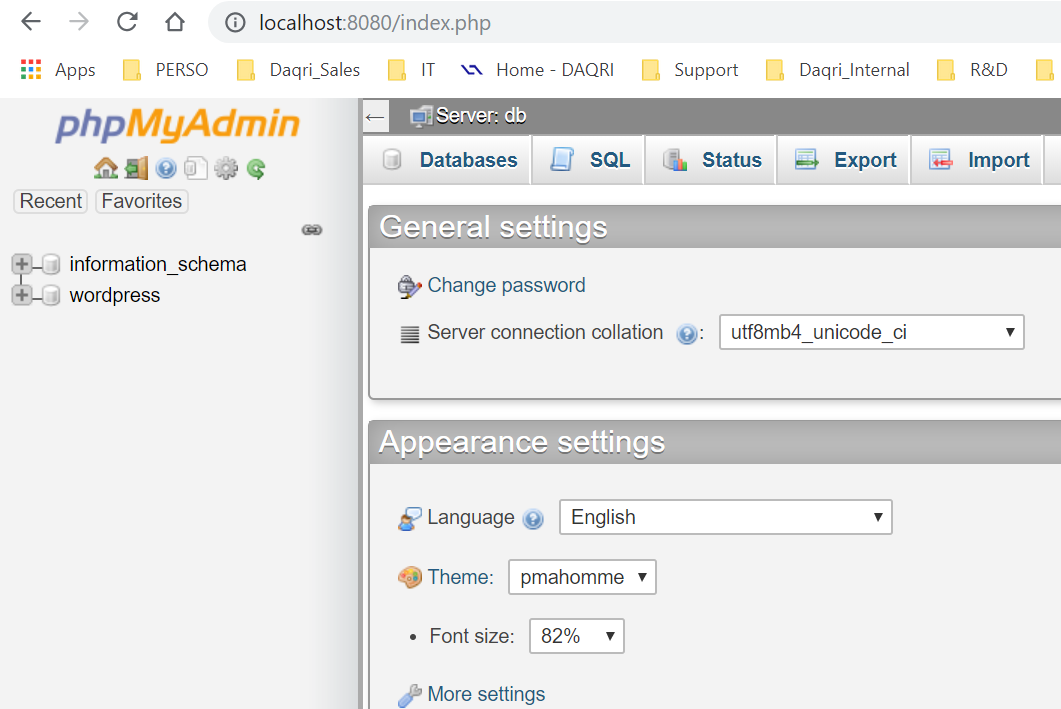
docker inspect 5088733b7951

then launch docker-compose up In the command line to start locally the container assembly but there Is not networking bridge with the outside world so we would need to either put everything In a cloud container solution or build a VM. I keep with my approach and chose a cloud based solution.

Verify one can login In wordpress:



And In php admin:



Stop the docker-compose as networking bridge does not allow to go to the Datadog Central SaaS.

We need create the container In our Cloud to avail

## Docker-compose and Azure Container cloud:

### Build custom Image and upload In the container registry

A. Create the Azure Container Registry: (**only once**)

az acr create --resource-group MyContainers --name RegistryCL --sku Basic --admin-enabled true

B. Login to registry

az acr login --name RegistryCL

C. get the name of the registry (only once)

az acr list --resource-group MyContainers --query "[].{acrLoginServer:loginServer}" --output table

which returns:

AcrLoginServer

---------------------

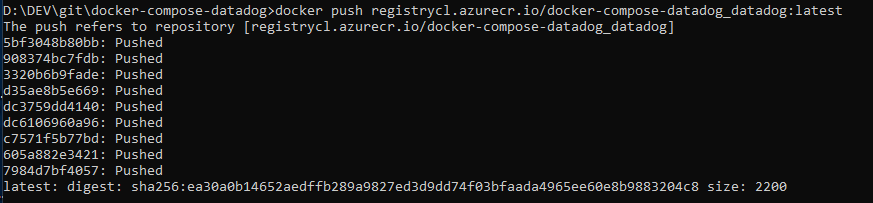
registrycl.azurecr.io

D. Tag the image:

docker tag docker-compose-datadog\_datadog registrycl.azurecr.io/docker-compose-datadog\_datadog:latest

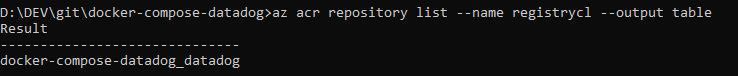
E. Push the docker Image to Azure:

docker push registrycl.azurecr.io/docker-compose-datadog\_datadog:latest

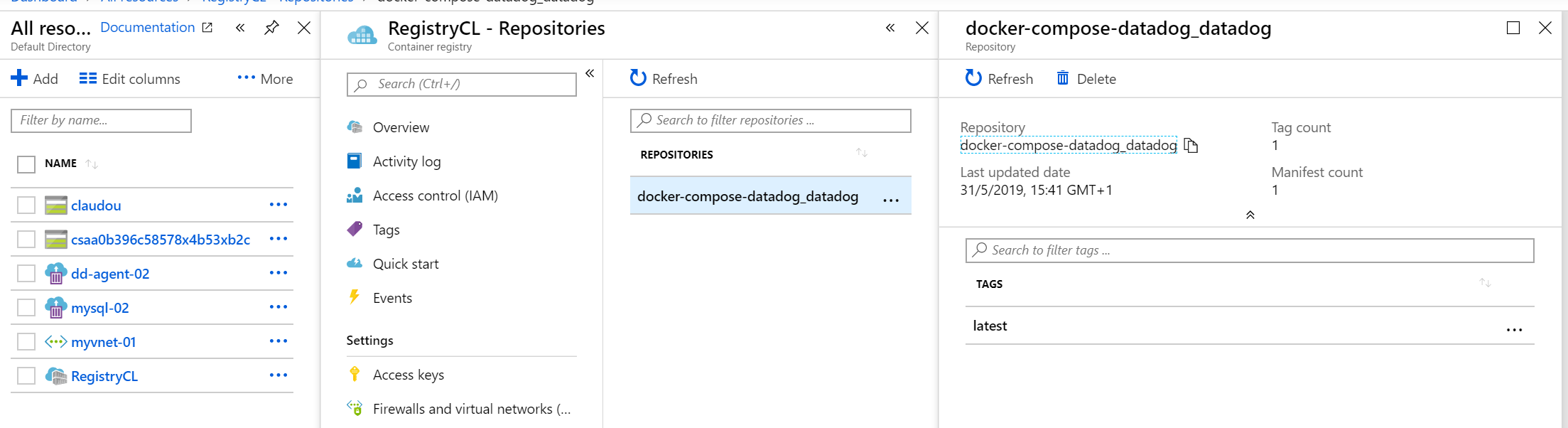


see the list:

az acr repository list --name registrycl --output table



Or In the portal:



F. Create a service principal: (only once)

see: https://docs.microsoft.com/en-us/azure/container-registry/container-registry-auth-service-principal

and script in azure folder

### Create a container assembly based on the Image:

create a new instance based on the image in Azure: (run from Cloud shell if on windows!!)

az container create --resource-group MyContainers --name datadog-demo --image registrycl.azurecr.io/docker-compose-datadog\_datadog:latest --cpu 1 --memory 1.5 --ports 8080 8000 80 \

--subscription Pay-As-You-Go --restart-policy OnFailure --location westeurope \

--dns-name-label cl-datadog-demo --ip-address public \

--registry-login-server registrycl.azurecr.io \

--registry-username 334cdc95-9890-4e70-a0ce-8a21f2d954e2 \

--registry-password 4450644b-f9a6-407f-989e-b0f495e32f22 \

--azure-file-volume-share-name containershare --azure-file-volume-account-name claudou --azure-file-volume-account-key SJH2STMwL7jrHGkQcWzVPdOVRp7Nk+qzodrICSXVSec5flUzzcTOy0hbIvasiQDESJwh+8GV9AHVgj01jErogA== --azure-file-volume-mount-path /conf.d

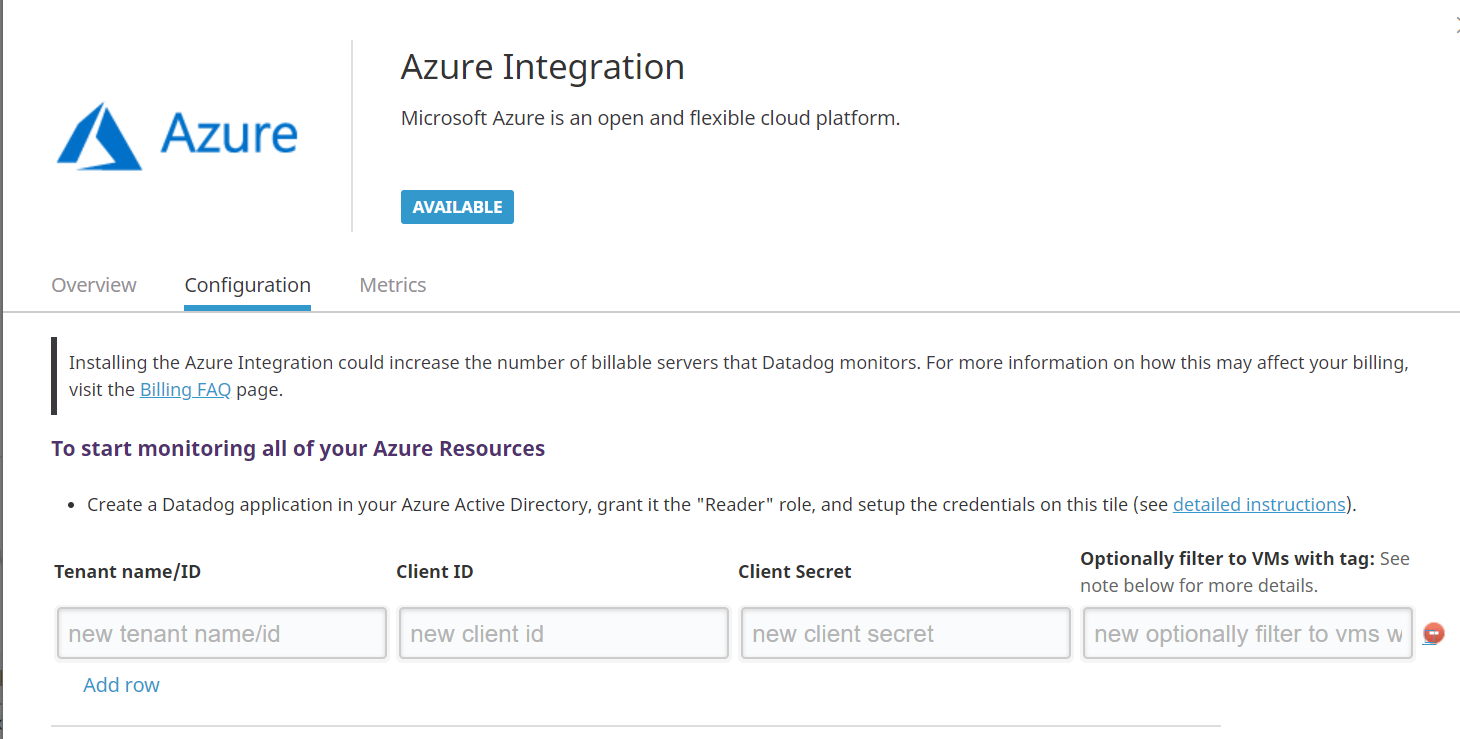
Verify datadog host In the datadog dashboard for Hosts:

All good.

# Collecting Metrics:

## Azure Integration:

<https://docs.datadoghq.com/integrations/azure/?tab=azurecliv20>

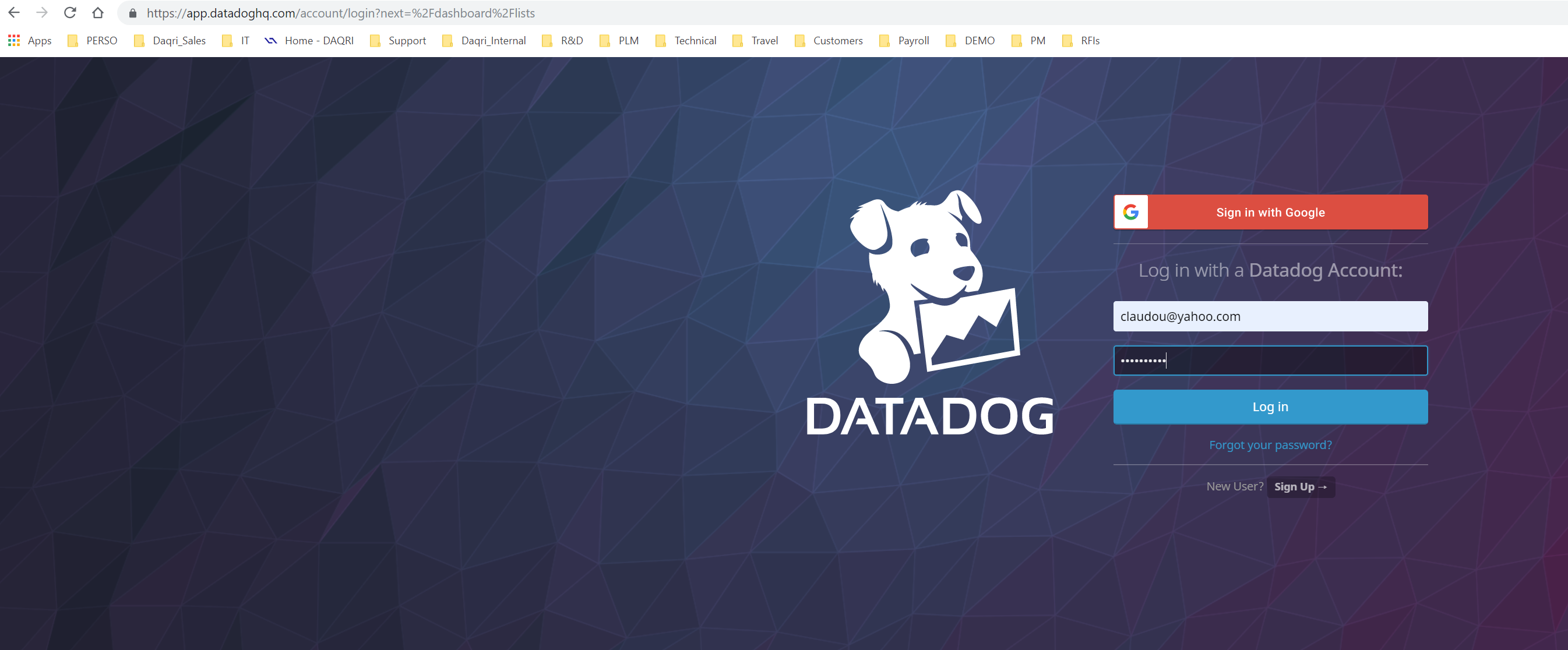


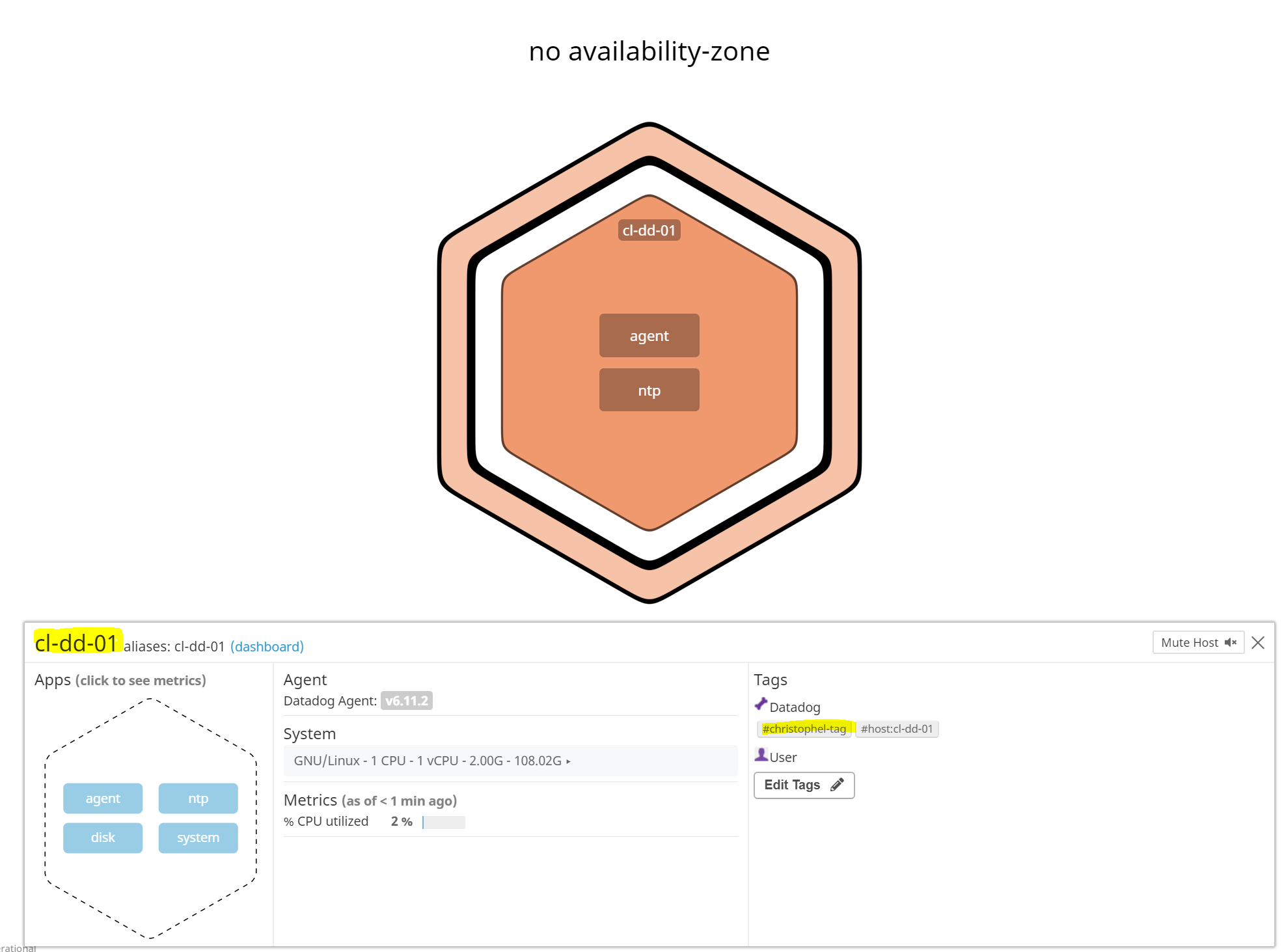
Directory ID Is : 1ea8375f-c7c7-4e95-8cdf-693520d2538a

Application IDL e41b528b-1add-4a29-a11e-52b8a8fb810d

we also generate a client secret and copy It accordingly: HK8N1WlBnPy\_ACN1fPOxIIB+l\_1NL.=g

## Agent TAGS:





## Custom Metrics:

## MySQL Metrics:

Log Collection

Available for Agent >6.0

By default MySQL logs everything in /var/log/syslog which requires root access to read.

To make the logs more accessible, follow these steps:

Edit /etc/mysql/conf.d/mysqld\_safe\_syslog.cnf and remove or comment the lines.

Edit /etc/mysql/my.cnf and add following lines to enable general, error, and slow query logs:

[mysqld\_safe]

log\_error=/var/log/mysql/mysql\_error.log

[mysqld]

general\_log = on

general\_log\_file = /var/log/mysql/mysql.log

log\_error=/var/log/mysql/mysql\_error.log

slow\_query\_log = on

slow\_query\_log\_file = /var/log/mysql/mysql-slow.log

long\_query\_time = 2

Save the file and restart MySQL using following commands: service mysql restart

Make sure the Agent has read access on the /var/log/mysql directory and all of the files within.

Double-check your logrotate configuration to make sure those files are taken into account and that the permissions are correctly set there as well.

In /etc/logrotate.d/mysql-server there should be something similar to:

/var/log/mysql.log /var/log/mysql/mysql.log /var/log/mysql/mysql-slow.log {

daily

rotate 7

missingok

create 644 mysql adm

Compress

}

I got all of this done In my first Installation/configuration. But then In the second I realised that I could not actually access the MySQL container. Purely because It was not created the right way

I am basically facing a docker compose issue with azure containers. I get the datadog agent container up and running but not the mysql or phpadmin one while it works on my laptop with docker-compose. I think it is an images in repository issu

## Update from the 02nd of June

## I will Investigate this further In a second workload.

# Visualizing Data:

# Monitoring Data:

# Collecting Data:

# Final Questions:

Graphical dashboard to build linked between the various agents (cluster) and servers.

# Load Testing and APP:

# Links and references

Datadog:

<https://docs.datadoghq.com/security/agent/>

<https://github.com/DataDog/docker-compose-example>

Microsoft:

<https://docs.microsoft.com/en-us/azure/container-instances/container-instances-quickstart>

<https://dev.mysql.com/doc/refman/8.0/en/linux-installation-docker.html>

<https://docs.docker.com/toolbox/toolbox_install_windows/>

<https://docs.microsoft.com/en-us/azure/app-service/containers/app-service-linux-ssh-support>

<https://docs.microsoft.com/en-us/azure/container-instances/container-instances-vnet>

<https://docs.microsoft.com/en-us/azure/container-instances/container-instances-multi-container-group>

https://github.com/MicrosoftDocs/azure-docs/tree/master/articles/container-instances

Docker:

<https://docs.docker.com/engine/reference/builder/#environment-replacement>

General:

<https://koukia.ca/push-docker-images-to-azure-container-registry-ed21facefd0c>

<https://vsupalov.com/docker-env-vars/>

# First Installation/Configuration:

## Datadog Agent

The Datadog documentation states to run the Installation with the following command line:

DOCKER\_CONTENT\_TRUST=1 docker run -d --name dd-agent -v /var/run/docker.sock:/var/run/docker.sock:ro -v /proc/:/host/proc/:ro -v /sys/fs/cgroup/:/host/sys/fs/cgroup:ro -e DD\_API\_KEY=57b752e8a4a68a1eafbd340a03bdbbb3 datadog/agent:latest

This Is translated In Azure GUI In the followings (don’t forget the environment variables like the DD\_API-KEY) If you want to have the agent registered with your Datadog dashboard.

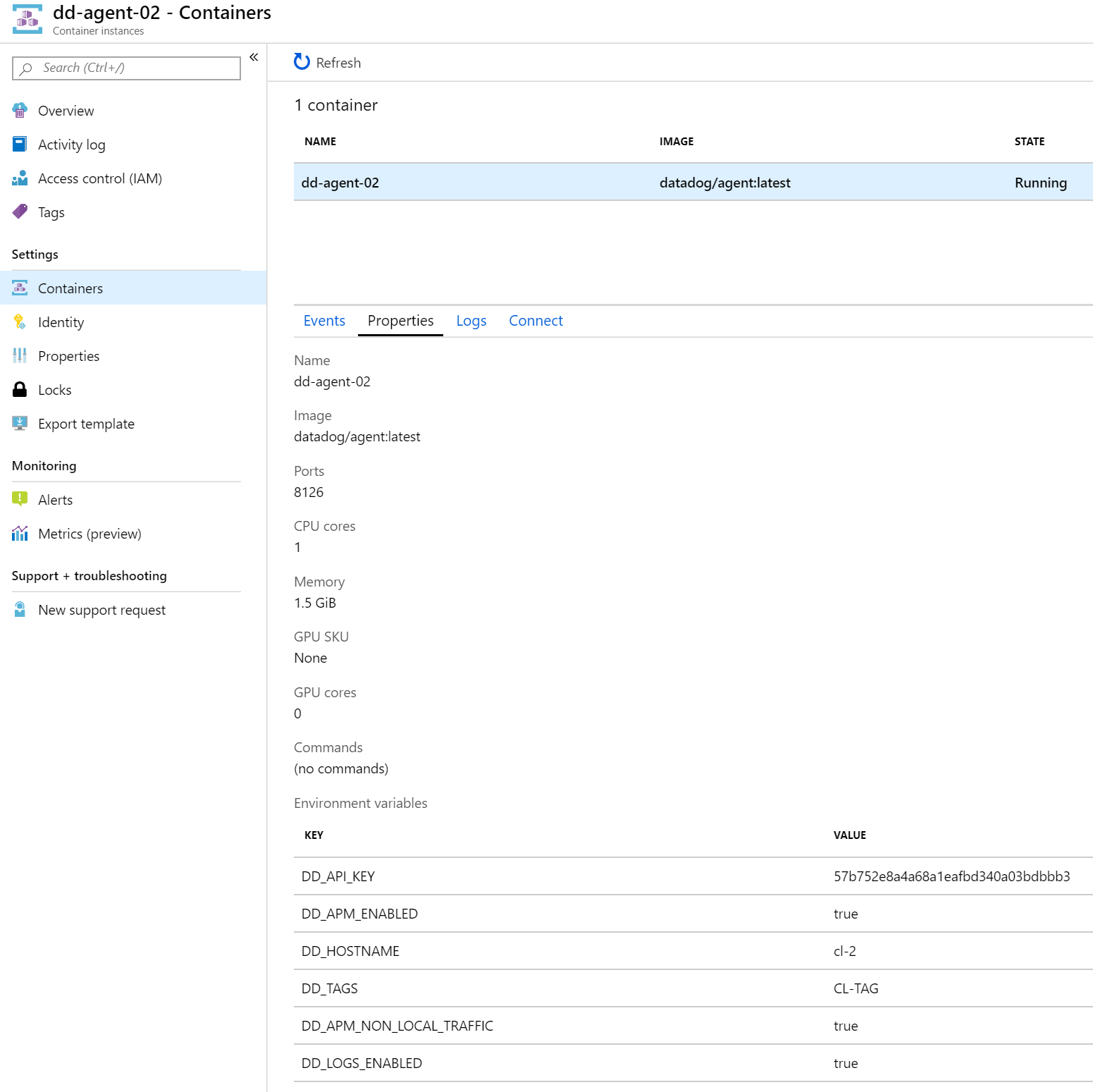
<https://github.com/DataDog/datadog-agent/tree/master/Dockerfiles/agent>

Best Is to use create\_container\_01.sh script to Batch the right setup.

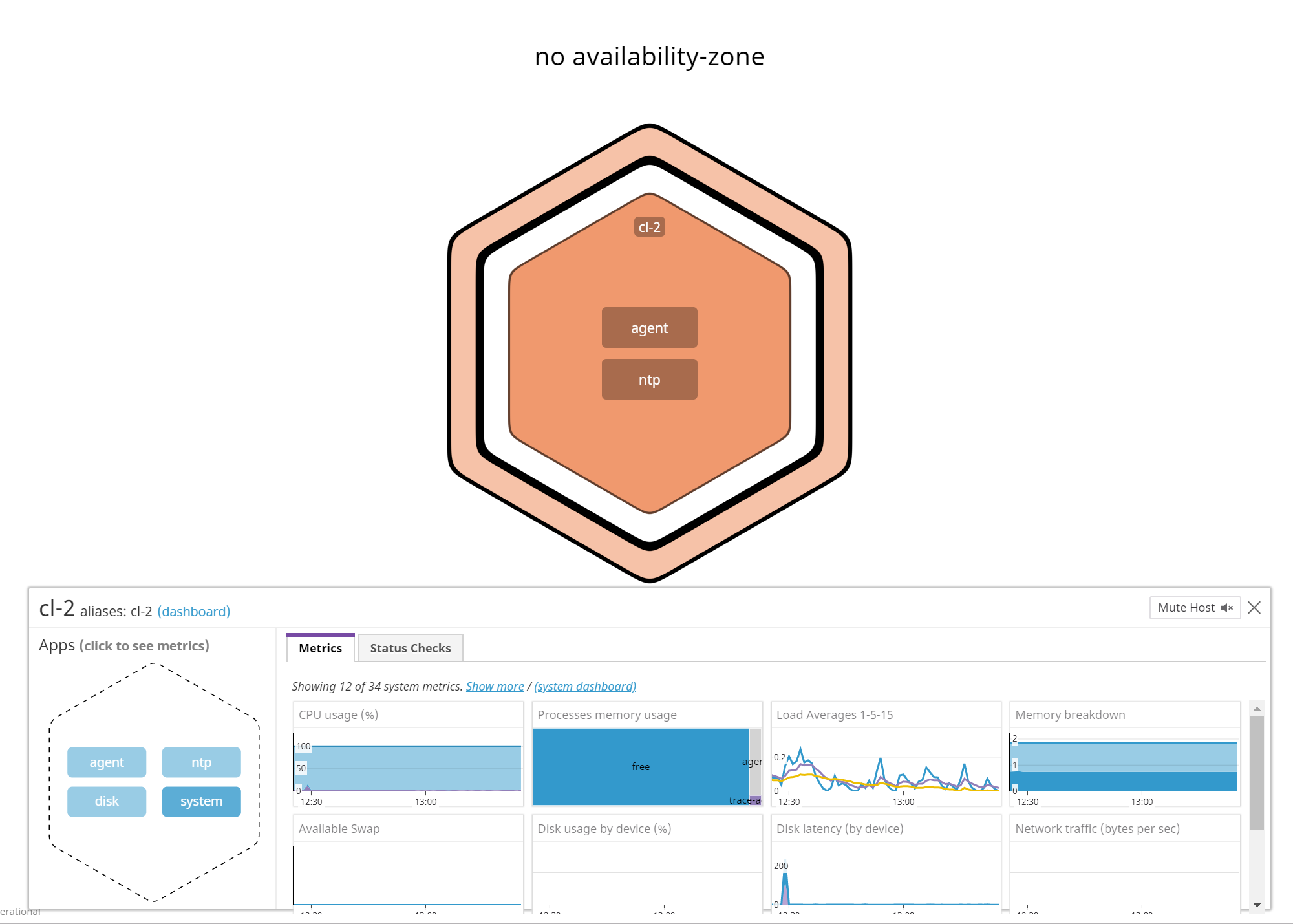
saves this as a script in the azure shell and then execute:

chmod +x ./create\_container\_01.sh

./create\_container\_01.sh



The Datadog Agent Azure container:



And It associated Datadog Host view

## MySQL Server

The official page Is: <https://github.com/mysql/mysql-docker>

And we are building using: mysql/mysql-server:latest

We are not being concerned for this exercise by the consistency of the storage. But In real world we should mount a consistent storage with this container In order to keep the RDBMS data consistent through time

we also use the environment variable: MYSQL\_ROOT\_PASSWORD=####

with ## being a password of our choice

SQL Connection:

Prepare the datadog user:

CREATE USER 'datadog'@'localhost' IDENTIFIED WITH mysql\_native\_password by '<UNIQUEPASSWORD>';

And

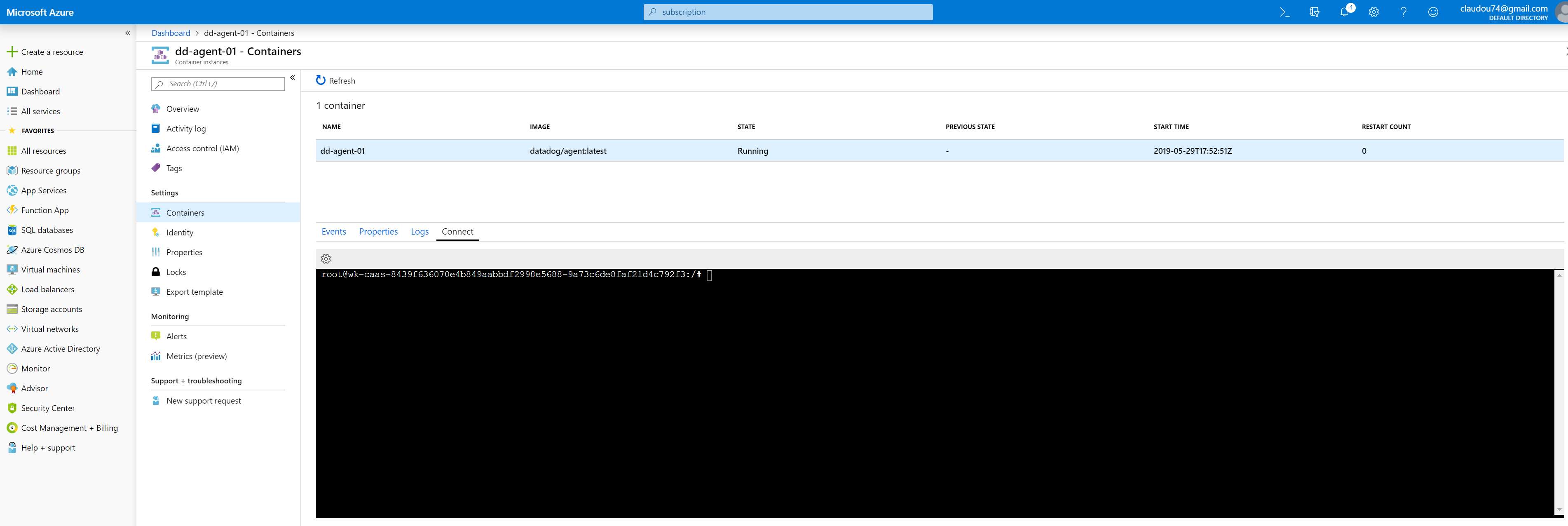
CREATE USER 'datadog'@'IP\_Datadogagent' IDENTIFIED WITH mysql\_native\_password by '<UNIQUEPASSWORD>';

Add the grants as In text file

## Datadog agent Integration with MySQL:

All details available In <https://app.datadoghq.com/account/settings#integrations/mysql>

we need to access our agent configuration:



we then update the configuration of the agent as follow:

Add this configuration block to your mysql.d/conf.yaml to collect your MySQL metrics:

init\_config:

instances:

- server: 10.1.0.5

user: datadog

pass: 'Dd1!'

port: 3306

options:

replication: false

galera\_cluster: true

extra\_status\_metrics: true

extra\_innodb\_metrics: true

extra\_performance\_metrics: true

schema\_size\_metrics: false

disable\_innodb\_metrics: false

restart agent !!

But as the docker agent Is not persistent we have to ensure to have our mysql.d/conf.yaml file updated Into a persistent storage that we mounted when created the container Instance for the datadog agent

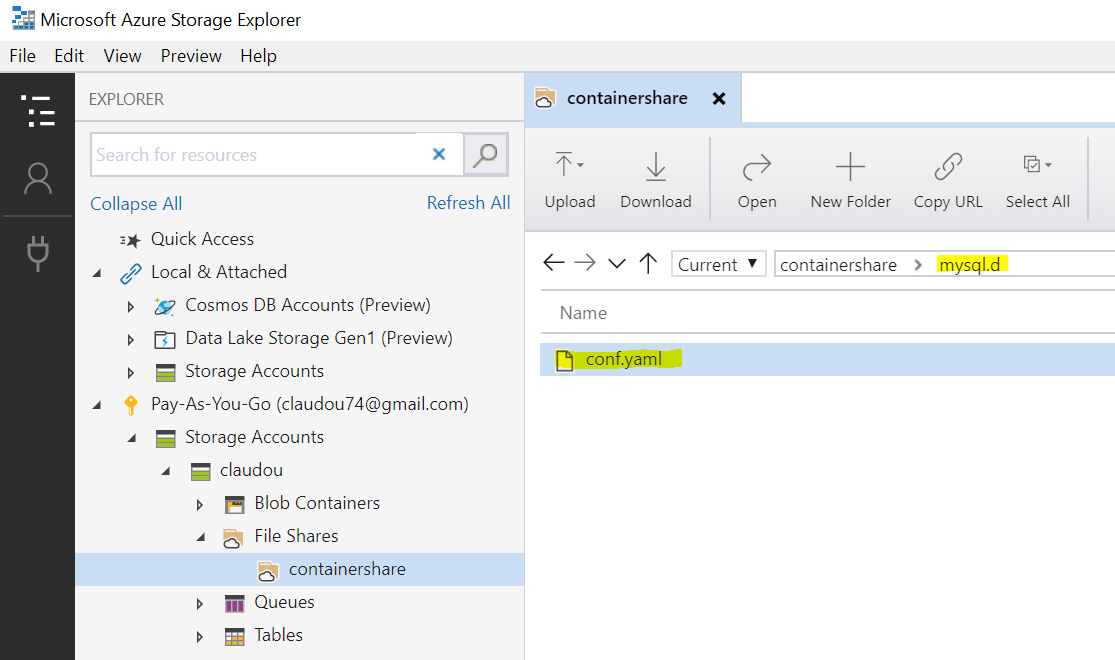
This Is due to the details provided In the document:

If you mount YAML configuration files in the /conf.d folder, they are automatically copied to /etc/datadog-agent/conf.d/ when the container starts.

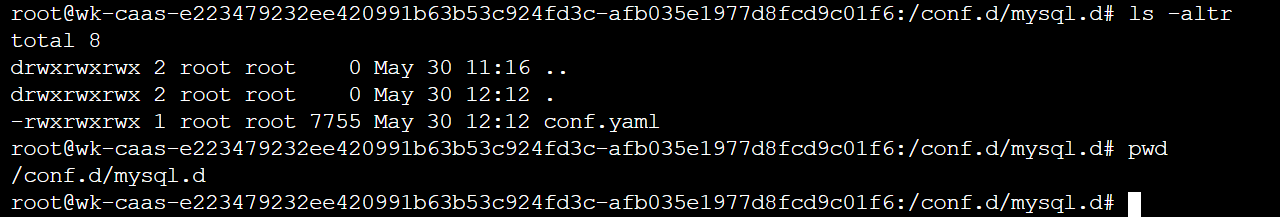
The same can be done for the /checks.d folder. Any Python files in the /checks.d folder are automatically copied to /etc/datadog-agent/checks.d/ when the container starts.

Insert the conf.yaml file here

we upload this file from our local disk to the share:



We then verify the availability of It at the container level:



And then only restart the container and verify that the custom file is actually taken into account:

I then configured MySQL for the agent to collect Information:

./create\_container\_02.sh

Verify that container instance 1 can ping instance 2

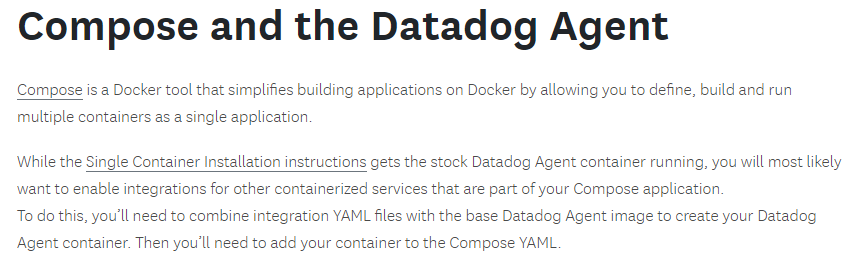
bash-4.2# curl https://10.1.0.4:8126

curl: (35) SSL received a record that exceeded the maximum permissible length.

So In our first setup we have two containers and everything seems ok and working fine.

We also think that the datadog agent will be able to communicate with the MySQL Server container.

This Is where I lost my day's work after finally searching while my agent could not communicate with the MySQL:



In the page that Is not easily found while docker agent Is detailed plenty:

<https://docs.datadoghq.com/integrations/faq/compose-and-the-datadog-agent/>