Back to Overview

Running a custom SQL Server Docker Image in Amazon AWS

November 26, 2018 ▪ Klaus Aschenbrenner ▪ Docker, Linux, SQLServer ▪ One Comment

In one of my last blog postings I have shown you how I run SQL Server natively on my Mac. In today's blog posting I want to continue with that approach, and I want to show you how you can create a custom SQL Server Docker Image, and how you can run that one in Amazon ECS – Amazon Container Services.

Creating and publishing a custom SQL Server Docker Image

If you have followed my blog posting about how to run SQL Server on the Mac, you should have now a fully working SQL Server 2019 CTP 2.1 environment running locally. One of the greatest things about Docker Images is also that you can create your own custom images that you can finally publish for other peers for reuse.

If you create your own custom image, your image will be based on some other image. Docker uses internally a layered file system, and therefore your custom image will be also quite small, because it only contains the changes on which your custom image is based on. In our case, the custom image that we will build today, is based on the SQL Server 2019 CTP 2.1 image.

One of the first things that I'm always doing when I install SQL Server is the restore of some databases that I will always use. In my case that's the **AdventureWorks** database, and in some cases I also restore the **ContosoRetailDW** database. You have already seen in my last blog posting how you can copy a database backup file into your SQL Server container, and how you can perform the restore itself.

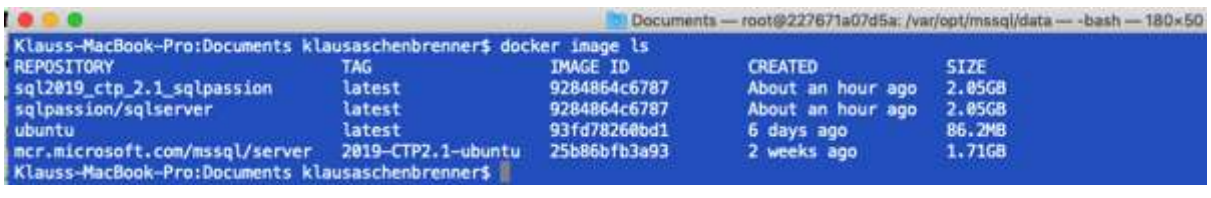
To make things easy today (and the custom Docker image small), I have only restored the **AdventureWorks** database. After the restore operation I have finally stopped the Docker Container with the following command line:

```
docker stop sql2019_ctp2.1
```

As soon as your Docker Container is in a stopped state, you can create your custom Dokcer Image that is based on the current state of the chosen Docker Container. To create a custom Docker Image you will use the following command line:

```
docker commit sql2019_ctp2.1 sql2019_ctp2.1_sqlpassion
```

As you can see from the command line, you have to pass in the name of the Docker Container, and the name of your new custom Docker Image. Easy, isn't it? If you now check your local Docker Image Repository, you can see that you have an additional Docker Image available – the custom one that you have just built!



Publishing a custom Docker Image

But how can you now publish your local created Docker image for yourself and also for other people? The answer is quite easy: you create your own repository on docker.com! If you check-out the following URL, you can see my own repository that I have created:

```
https://hub.docker.com/r/sqlpassion/
```

To be able to push custom Docker Images into your repository, you have to tag them accordingly. In my case I have tagged the image with the sqlpassion/sqlserver:latest tag. If you don't tag your image, you can't push it into your repository!

```
docker image tag sql2019_ctp_2.1_sqlpassion:latest sqlpassion/sqlserver:latest
```

And after the tagging is done, let's push the image into my Docker repository:

```
docker push sqlpassion/sqlserver
```

```
Klauss-MacBook-Pro:Documents klausaschenbrenner$ docker push sqlpassion/sqlserver
The push refers to repository [docker.io/sqlpassion/sqlserver]
7a36dd618e51: Pushed
d4fd7df04168: Pushed
6af8f3ce2b62: Pushed
911ad08015cf: Pushed
d33a8ef9dea5: Pushed
2de391e51d73: Pushed
d73dd9e65295: Pushed
686245e78935: Pushed
d7ff1dc646ba: Pushed
644879075e24: Pushed
latest: digest: sha256:8a981b3817796774e46b06b53fc13dac1b7280a0fd0dbf92931925b00f9ec1c8 size: 2413
```

If you search on your local Docker installation for **sqlpassion**, you will be also able to find my custom Docker Image:

```
Documents — root@227671a07d5a: /var/
Klauss-MacBook-Pro:Documents klausaschenbrenner$ docker search sqlpassion
NAME                    DESCRIPTION         STARS               OFFICIAL            AUTOMATED
sqlpassion/sqlserver                        0
Klauss-MacBook-Pro:Documents klausaschenbrenner$
```

If you pull that one, you can deploy it locally on your hardware. Docker is just cool! 🙂

Running a Docker Container in Amazon AWS

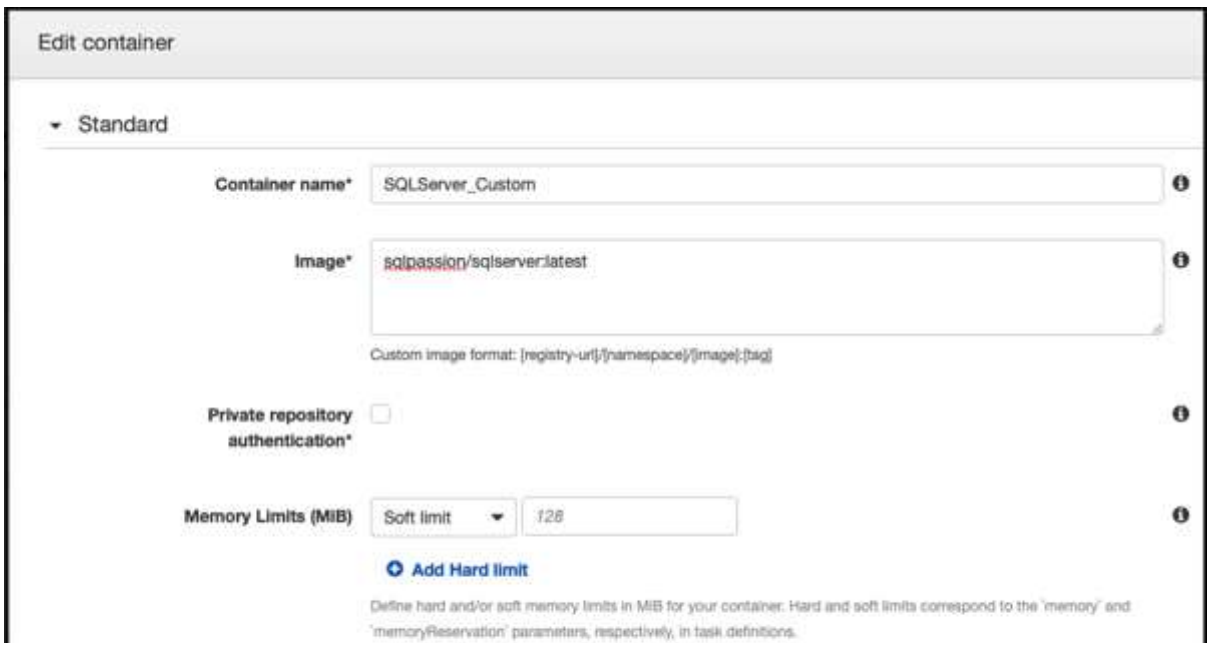
In my case I wanted to try how I can run a custom Docker Image in the cloud. Because I'm using the Amazon AWS, and especially Amazon EC2 already for some years, I wanted to try if it is possible to spin up my custom Docker Image in Amazon AWS.

One of the cool things that I have found out about Amazon AWS is the fact that they offer Amazon ECS – Amazon Container Services. You can run natively a Docker Container in Amazon AWS without creating EC2 instances. The creation of the necessary Virtual Machines (that actually run your Docker Containers) is just done transparently in the background of Amazon ECS.

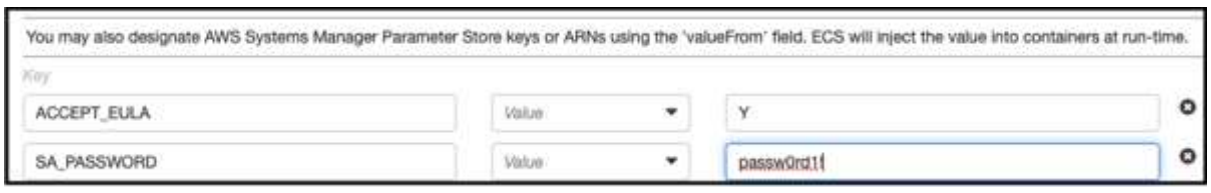
So let's have now a look on the necessary steps to spin up a few instances of my custom Docker Image. To be able to run a Docker Container, you have to configure the following Amazon ECS objects:

- Container Definition
- Task Definition
- Service Definition
- Cluster Definition

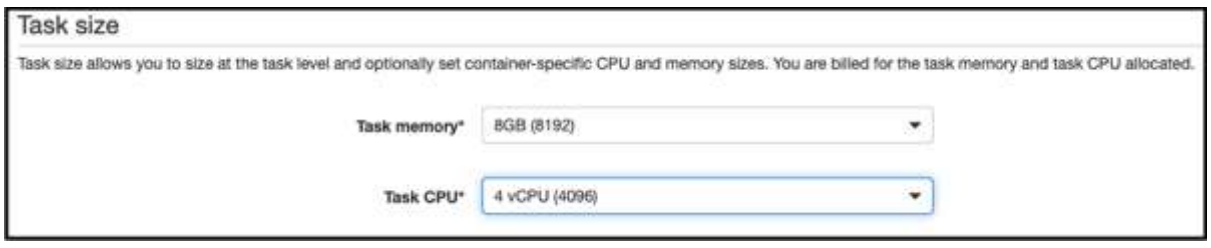
So let's have a look on these objects, and how to configure them accordingly. In the first step you have to create a **Container Definition**. You can choose from existing Container Images (like nginx, tomcat, ...), or you choose a third-party Docker Image – like the one we have created in the previous step.



As you can see, you give the Container a name, and you have to specify your Docker Image source. When you don't specify a Registry URL, they are defaulting to the Docker Registry. In my case I have provided the Image URL sqlpassion/sqlserver:latest. And you also have to create a Port Mapping for the port 1433, so that we can access SQL Server from the outside. And under the Advanced Container Settings, I have specified the necessary environment variables for SQL Server:

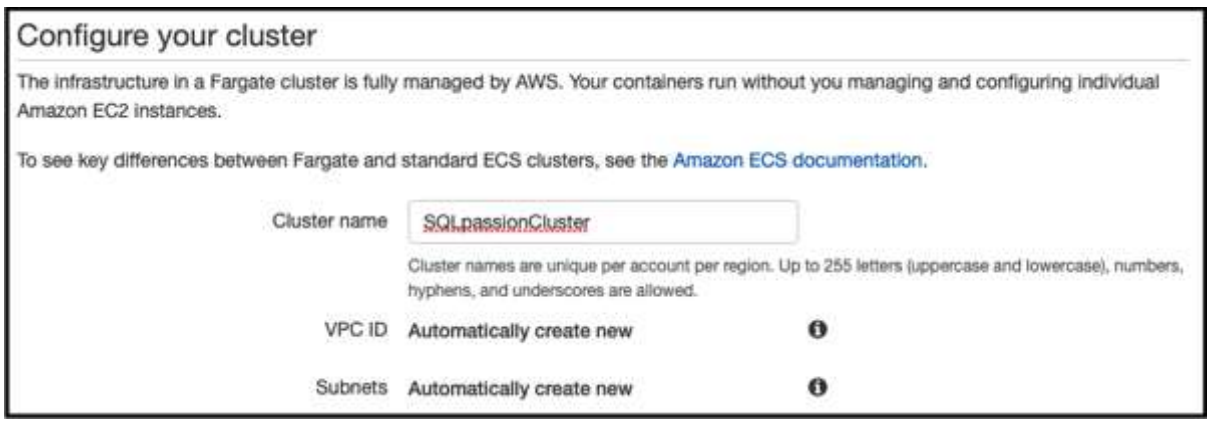


In the next step you have to configure the Task Definition. With a **Task Definition** you specify how many CPUs and how much memory you want to assign to your Docker Container. In my case I have given the Docker Container 4 CPUs and 8 GB RAM.

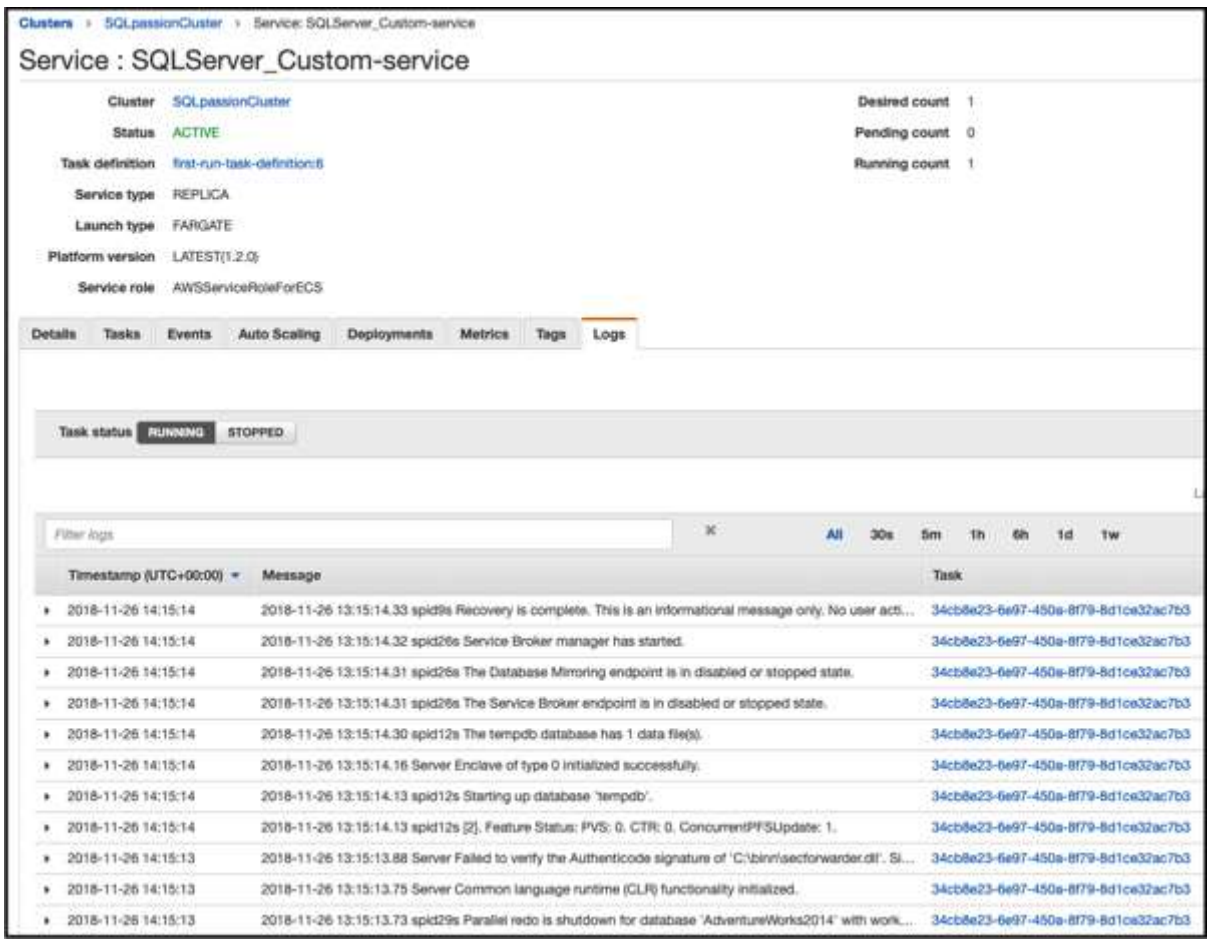


With the **Service Definition** you can specify how many instances of your Docker Container you want to start. In my case I have configured just 1 Task.

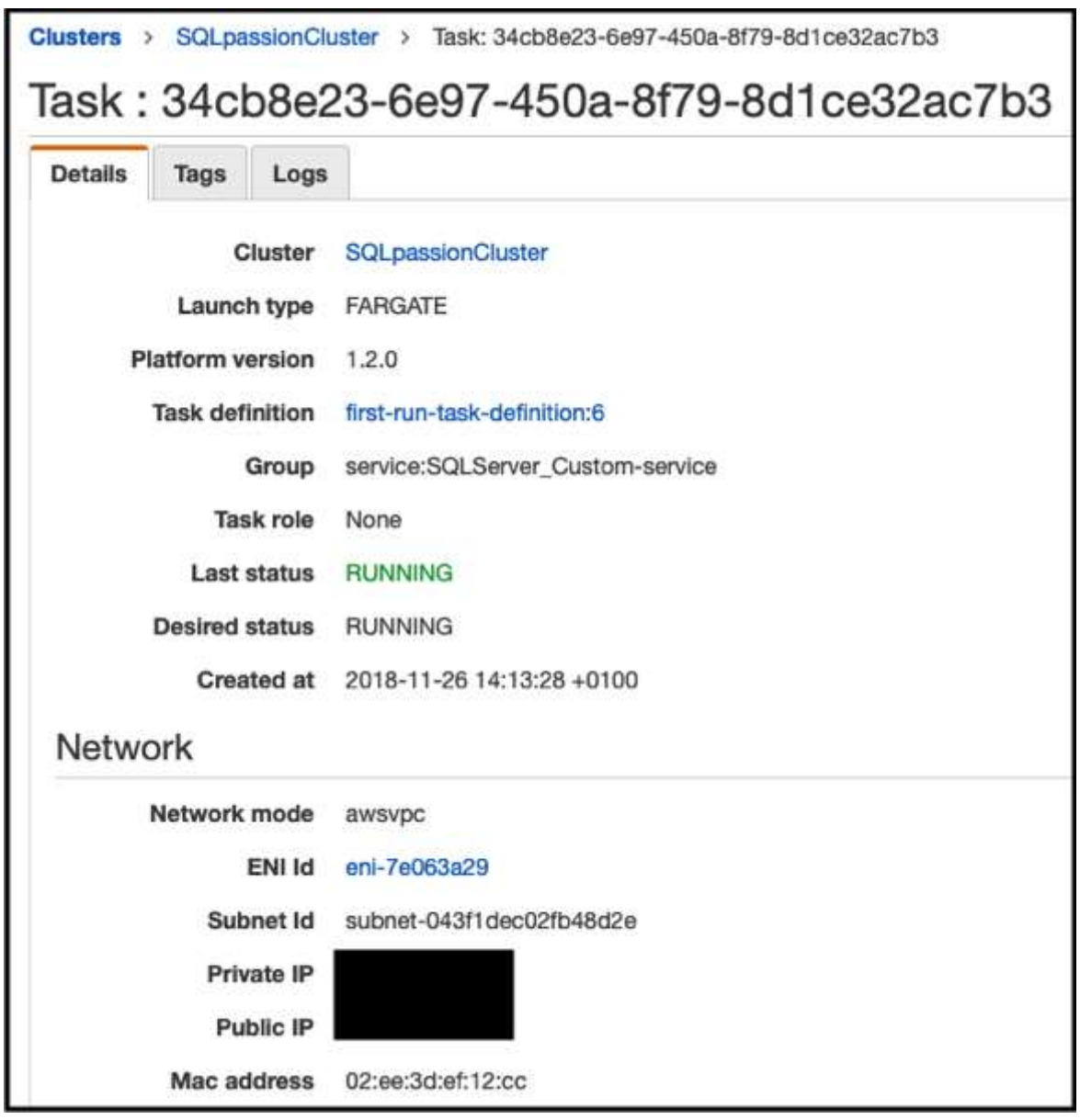
And finally you have your **Cluster** a name:



After you have confirmed all your settings, it takes a few minutes and your Service is up and ready. When you look into the tab Logs, you can see that SQL Server is up and running within the Docker Container:



If you go to the Task itself, you can see in the Networking settings the public IP of your Docker Container. You can use that IP address within SQL Server Management Studio or Azure Data Studio to connect to your SQL Server Instance:



If you have connected to SQL Server, you can finally see that you have deployed a custom Docker Image, because the **AdventureWorks** database is already there!

Summary

I have done over the last few weeks a lot of research and work with Docker and Kubernetes. But Docker changes everything! Imagine what you can do with custom published Docker Images. You could have for each conference talk a custom Docker Image, and the attendees are just pulling it from your Docker Repository to follow along your talk.

Or imagine you want to offer Lab-based training materials. You could create a custom Docker Image for some Hands-On Labs that you can run on your local hardware, or in the Cloud. The options are just endless – at least for me…

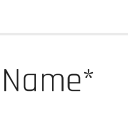
Thanks for your time,

-Klaus

PREVIOUS
SQLpassion Black Friday Deal 2018

NEXT
How many Cores per Socket do I need?

1 thought on "Running a custom SQL Server Docker Image in Amazon AWS"



SHANJAMES

11/08/2019 AT 11:10 AM

Thanks for sharing this article as its a great example of how easy it is to suddenly find yourself in trouble in these environments.

Reply

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment »

SQLpassion Performance Tuning Training Plan – Signup

Be sure to checkout the FREE SQLpassion Performance Tuning Training Plan – you get a weekly email packed with all the essential knowledge you need to know about performance tuning on SQL Server.

Your E-Mail

YES, I WANT IT