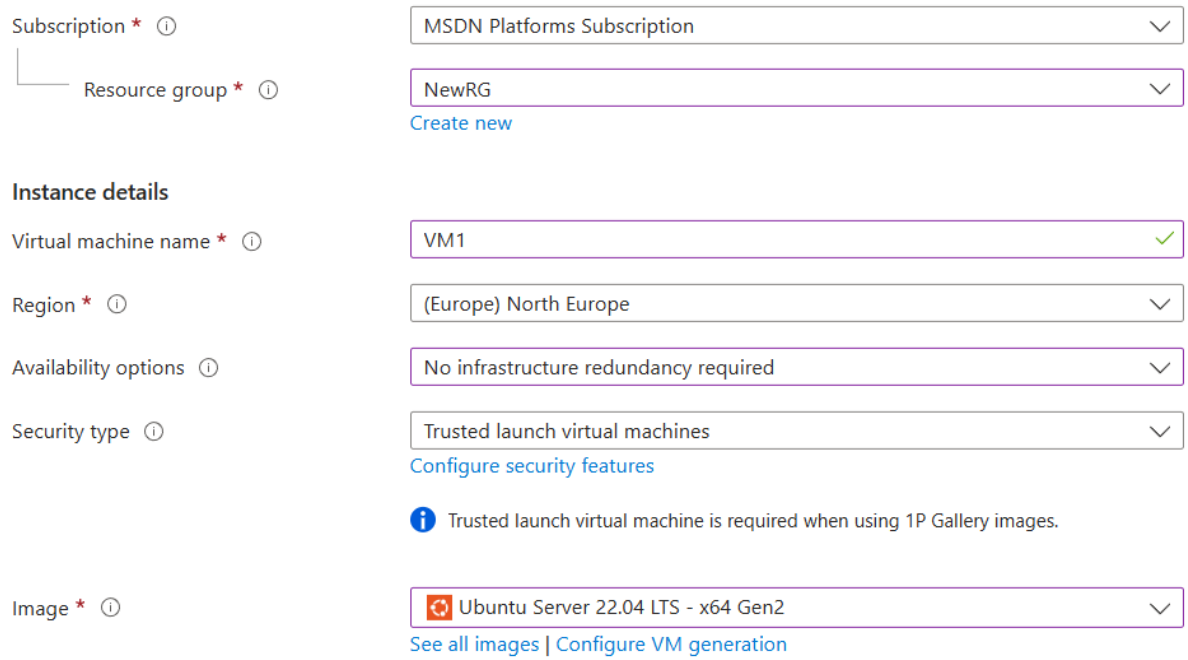
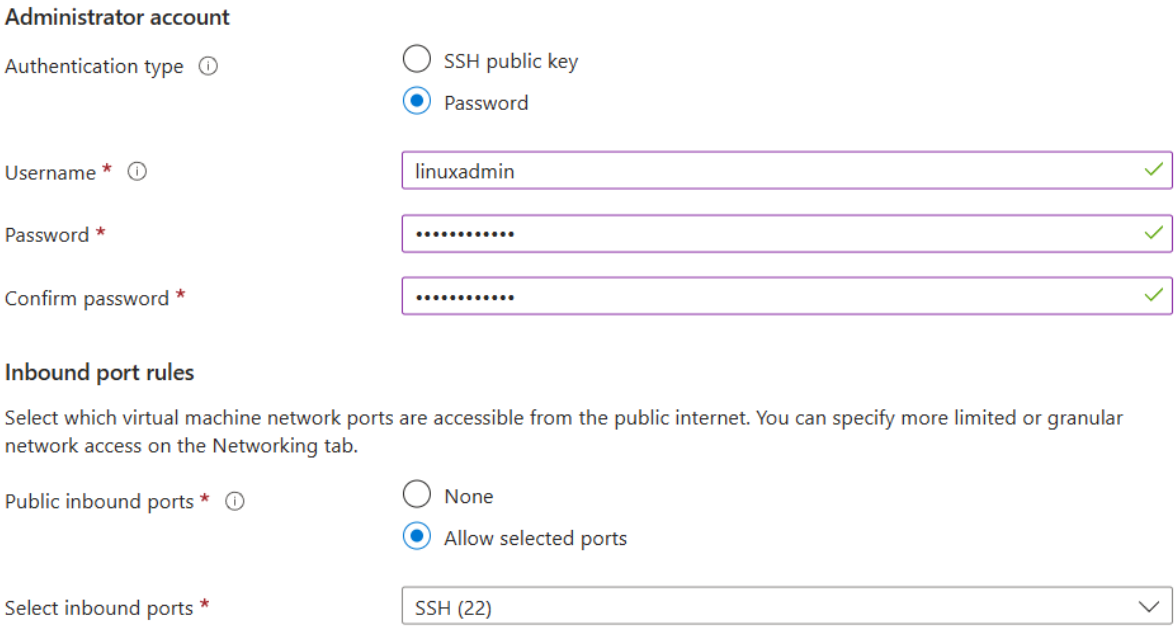
😊 Deploying Docker Host

The process involves deploying an Ubuntu-based Virtual Machine (VM) in Azure, installing Docker, and transferring necessary files via WinSCP. A Docker image is built for MySQL and pushed to Azure Container Registry. An Azure Container Instance is then created to run the MySQL database. The container's public IP is configured in an application to store and retrieve data. The end goal is to deploy a MySQL container on Azure, connect it with an application, and display data on a browser, ensuring seamless database hosting and accessibility using Docker and Azure Container Services.

1. In your Azure Portal go and create a Virtual Machine based on Ubuntu Image.
2. Choose your resource group then give a name to your VM. Then choose your image as Ubuntu server.



3. Then choose the password in the authentication type and move to the review page to create your VM.



4. Once your VM is created SSH into it using the command line or the Putty tool. We will be using the command line to connect with our VM. Using the below command we can connect with the VM. We are using the public IP address.
   ssh linuxadmin@40.87.152.100

```
$ ssh linuxadmin@40.87.152.100
The authenticity of host '40.87.152.100 (40.87.152.100)' can't be established.
ED25519 key fingerprint is SHA256:IunWWXFD869h9oTJfgNjlivAPHV5eWnMAWJ3lP84FGU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '40.87.152.100' (ED25519) to the list of known hosts.
linuxadmin@40.87.152.100's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1021-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro
```

5. Once we are connected with our VM then we will install Docker into it. You can visit the official page to install docker or you can use the link below.

   https://docs.docker.com/engine/install/ubuntu/

1. Set up Docker's `apt` repository.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.do
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

2. Install the Docker packages.

Latest | Specific version

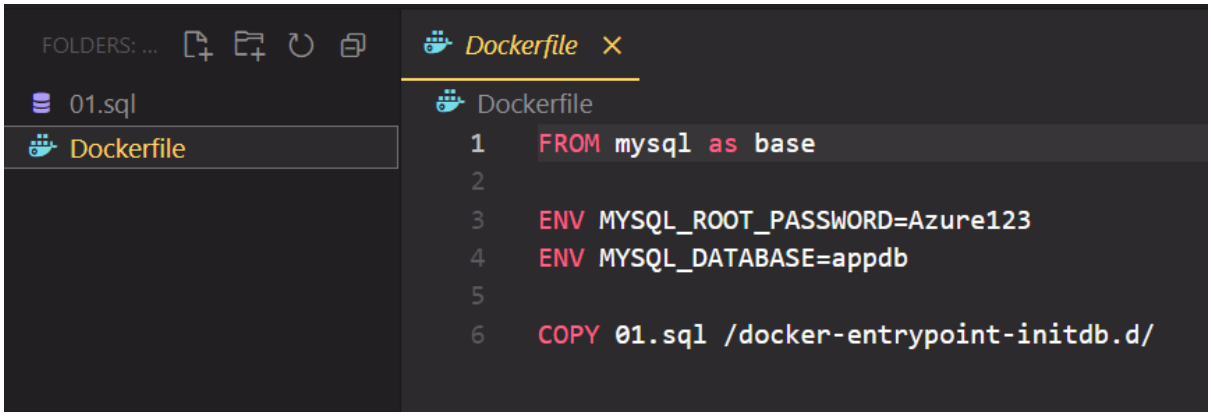
To install the latest version, run:

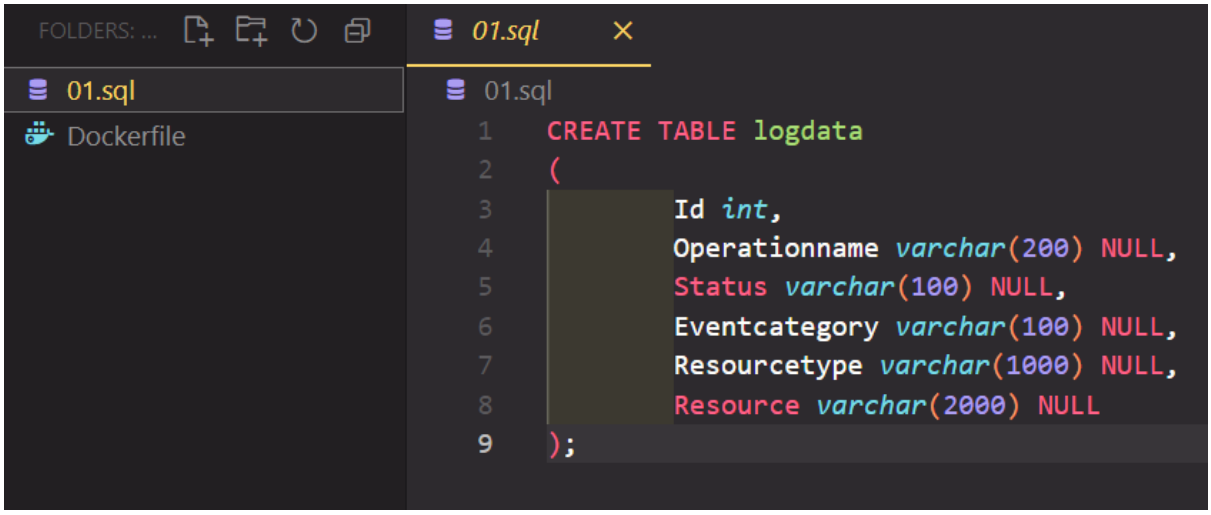
```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plug
```

3. Verify that the installation is successful by running the `hello-world` image:

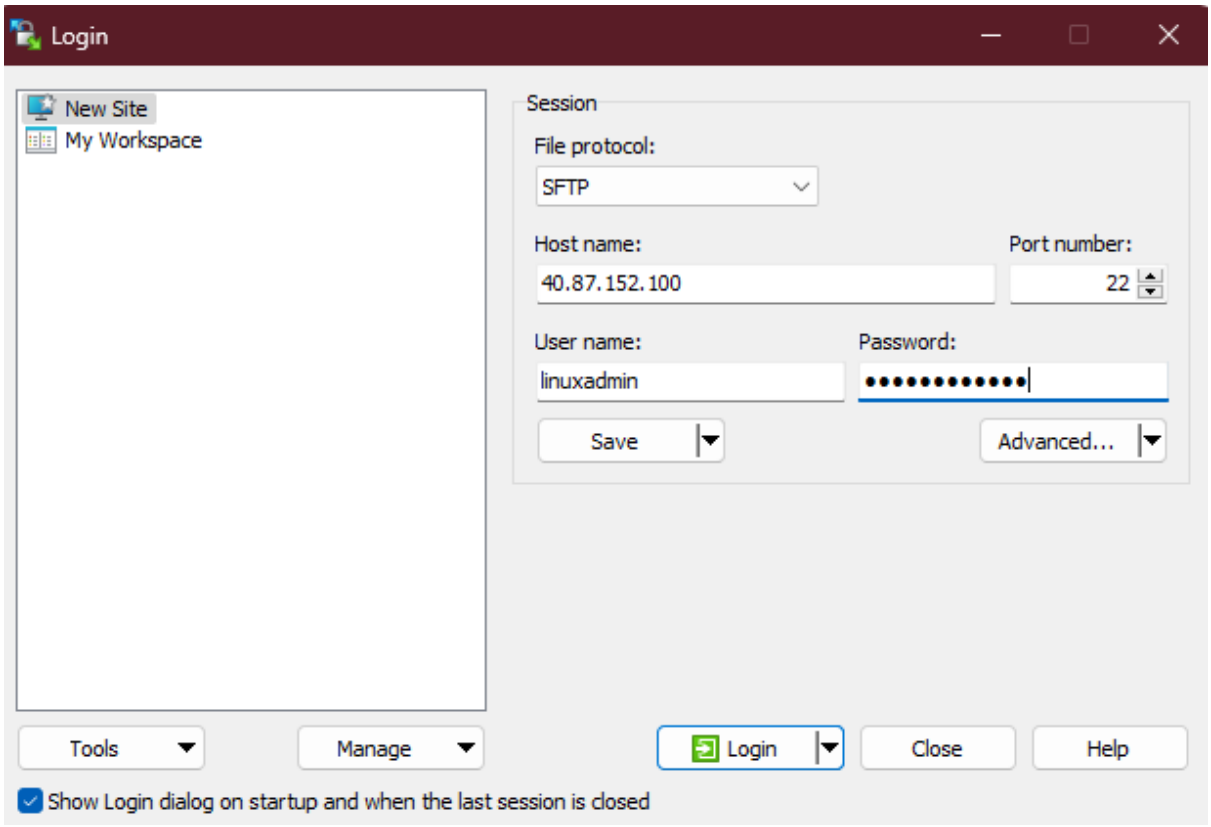
```
$ sudo docker run hello-world
```

6. Now we have two files that we need to transfer from our laptop to the VM. The first file is the docker file in this we are specifying MySQL as the base which will be downloaded onto the VM from the docker hub. Also, we are saying that copy the 01.sql as the entry point. In the 01.sql file, we have the log data table.
7. To copy all these two files onto the Linux VM we are going to use the **Win SCP tool**.

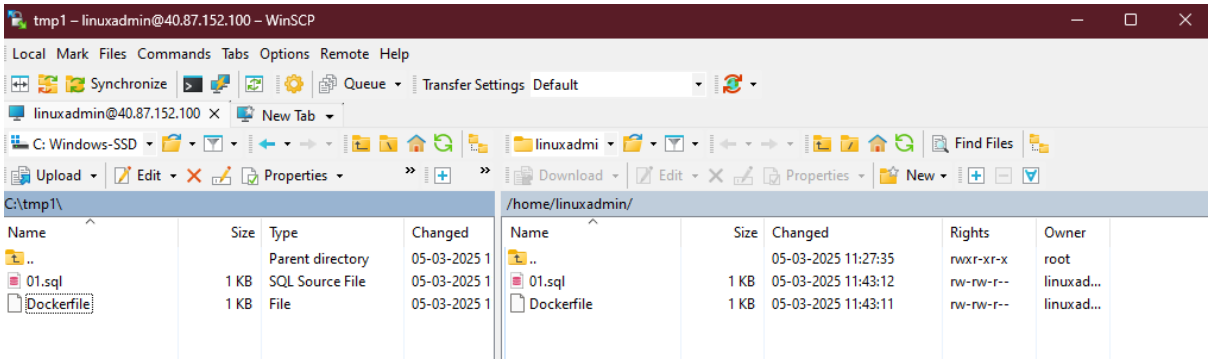




8. Open the Win SCP tool, in the hostname use the public IP address of the VM then give the username and password of your VM. Click on Login.



9. Then we need to go to the folder where we have put the files then just drag those files to the right-hand side.



10. Once the files are copied then just come back to the terminal and run the below command to build our image. If you do a listing of objects inside the terminal you can see them.

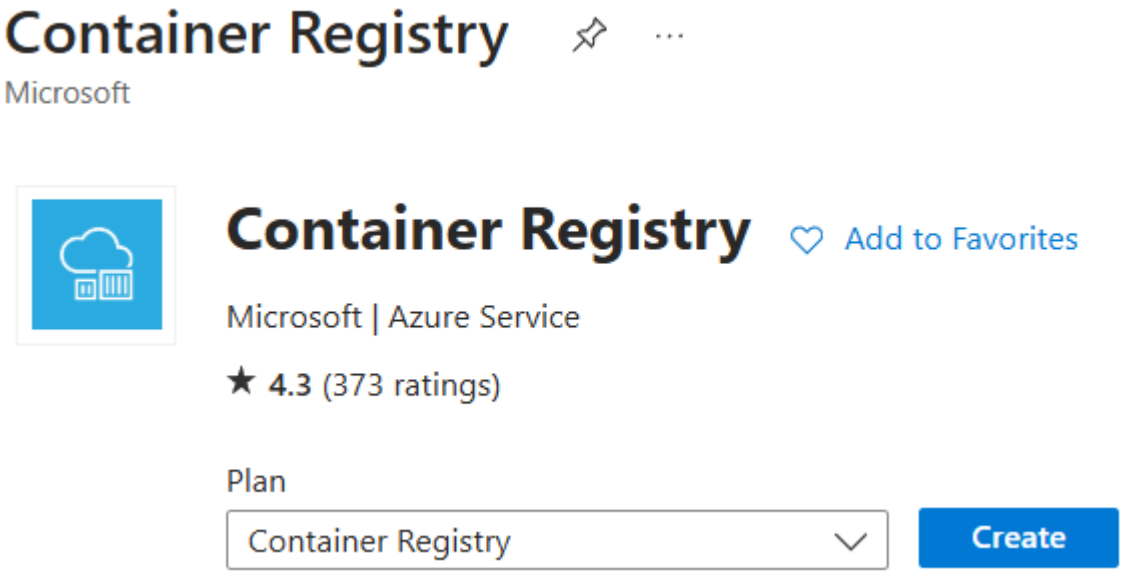
```
linuxadmin@VM1:~$ ls
01.sql  Dockerfile
linuxadmin@VM1:~$
```

11. Just run the below command to build an image. Once the command is executed you will see the image.

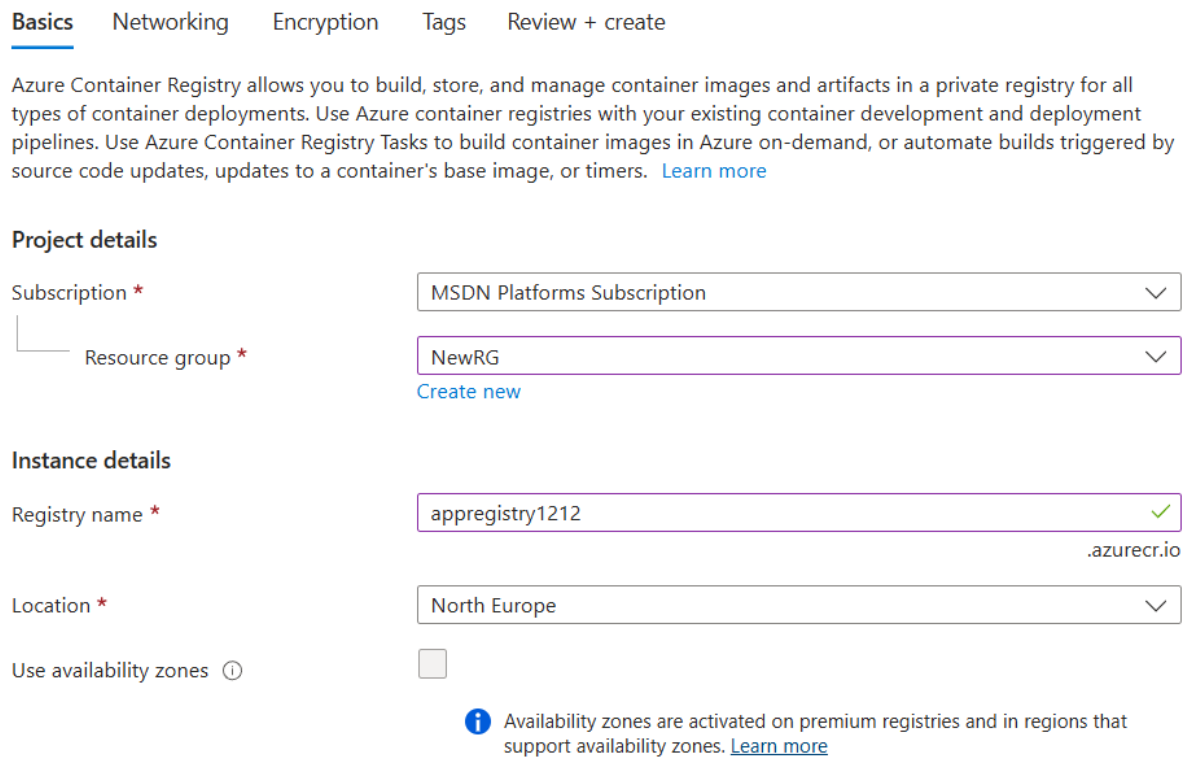
    sudo docker build -t appsqlimage .

```
linuxadmin@VM1:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
appsqlimage   latest    3757f909f7bc   9 minutes ago    797MB
linuxadmin@VM1:~$
```

12. Now we will create a new resource Azure Container registry. It is a private docker registry available on the Azure platform. So, in the marketplace search for it. Click on create.



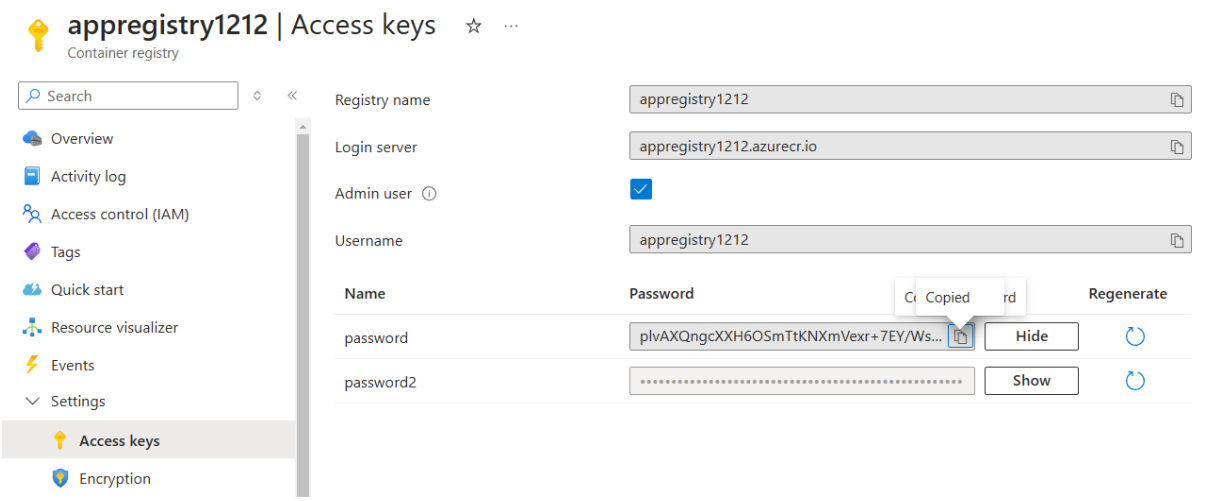
13. Choose your Resource group, give it a unique name and move to review page to create your registry.



14. Now we are going to use the below commands to push the image onto the registry. First, you need to change the name of the registry in the commands and on line 6 you will see that you have added the password to get it you need to open the access keys in your container registry and enable the admin user it will create a username and password for you. Copy it and paste it.

commands.txt

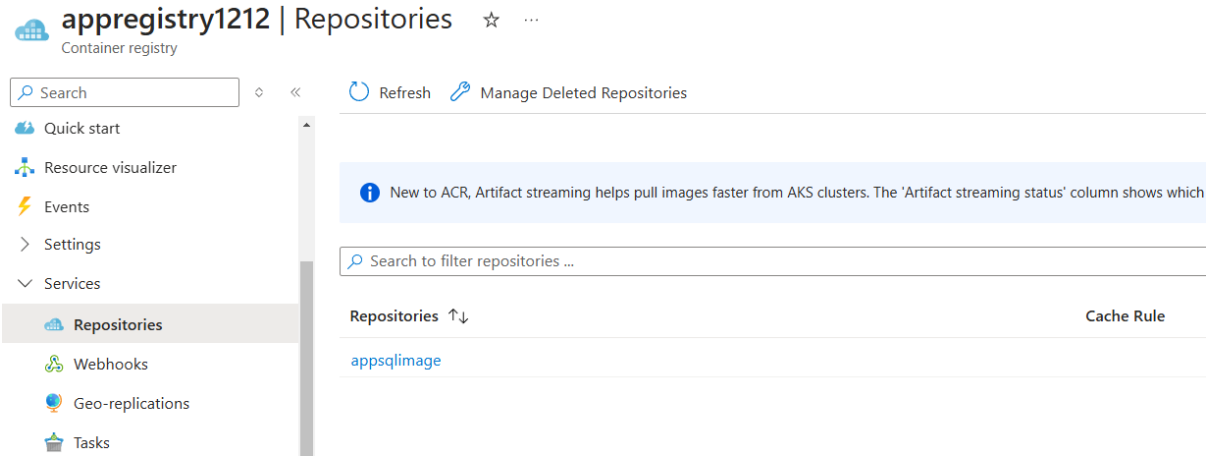
```
1  sudo docker build -t appsqlimage .
2
3
4  Push this to Azure Container Registry
5
6  sudo docker login appregistry1212.azurecr.io -u appregistry1212 -p plvAXQngcXXH6OSmTtKNXmVexr+7EY/WsJNaox4QkK+ACRBDCzZ3
7
8  sudo docker tag appsqlimage appregistry1212.azurecr.io/appsqlimage
9
10 sudo docker push appregistry1212.azurecr.io/appsqlimage
```



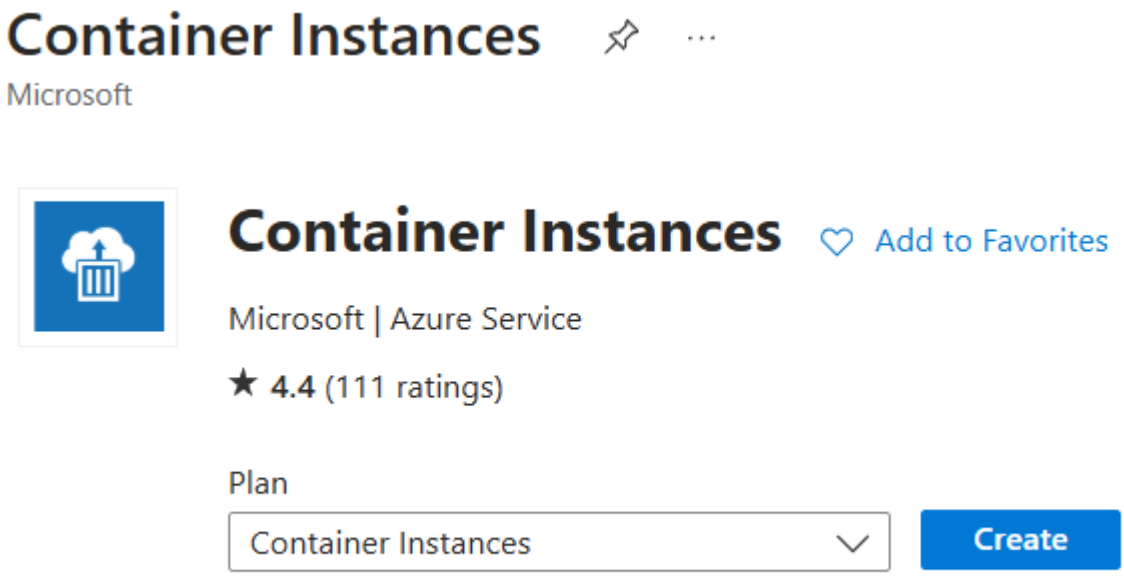
15. Once you are done setting up the commands, run them in the terminal. Below you can see that our image has been pushed to the registry.

```
linuxadmin@VM1:~$ sudo docker login appregistry1212.azurecr.io -u appregistry1212 -p plvAXQngcXXH6OSmTtKNXmVexr+7EY/WsJNaox4QkK+ACRBDCzZ3
WARNING! Using --password via the CLI is insecure. Use --password-stdin.

WARNING! Your credentials are stored unencrypted in '/root/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
linuxadmin@VM1:~$ sudo docker tag appsqlimage appregistry1212.azurecr.io/appsqlimage
linuxadmin@VM1:~$ sudo docker push appregistry1212.azurecr.io/appsqlimage
Using default tag: latest
The push refers to repository [appregistry1212.azurecr.io/appsqlimage]
540719094230: Pushed
bccedbf428d2: Pushed
cfb04b43e338: Pushed
21cd7e76ec3c: Pushed
501c17081758: Pushed
bfba4799a5e3: Pushed
5494a032973c: Pushed
433b6ea8deb0: Pushed
8c039733b996: Pushed
9986054dfacc: Pushed
d7b2257a2277: Pushed
latest: digest: sha256:88f48b1ef790ac34852e242b36a7b6a1b17658d57045d2f14f181a47a75b90e0 size: 2619
linuxadmin@VM1:~$
```

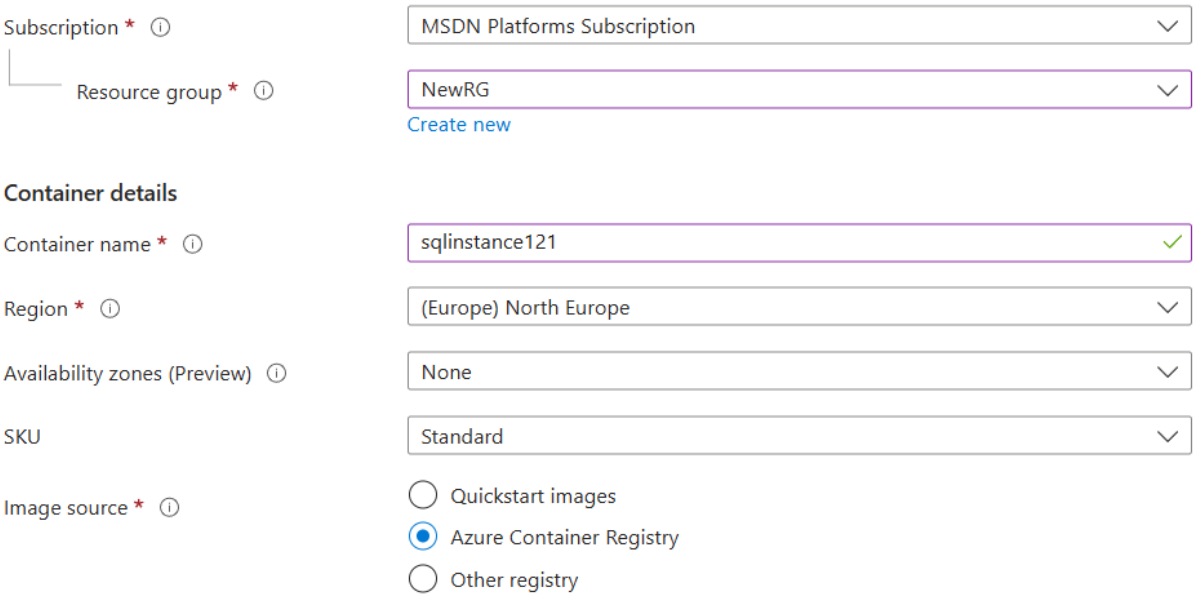
16. Now on the Portal if you go to repositories tab you will see a repository there.



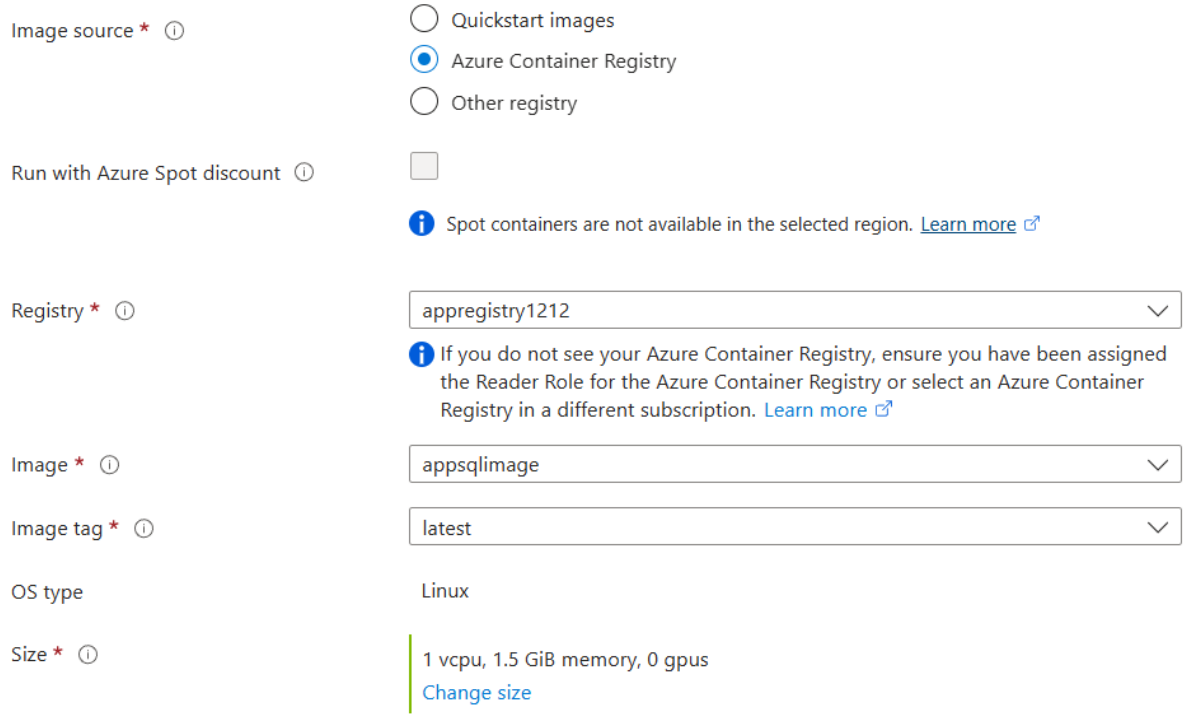
17. We can now create a running container out of that image, and then an application can actually connect to the MySQL database running in that container. We could run that container on a machine that has the Docker engine installed. But in this case, we are going to be using the Azure Container Instance to go ahead and run our container.
18. In the marketplace search for container instances and click on Create.



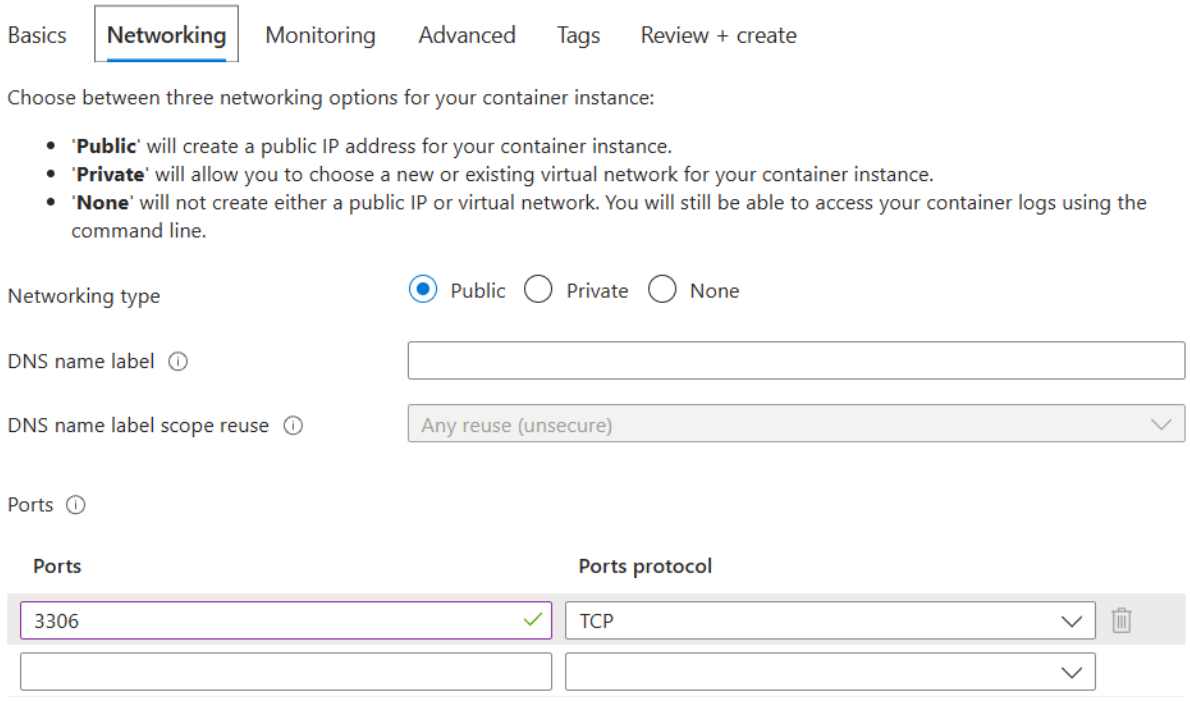
19. Then choose your resource group, and give a name to your container. Then for the image source choose Azure Container Registry.



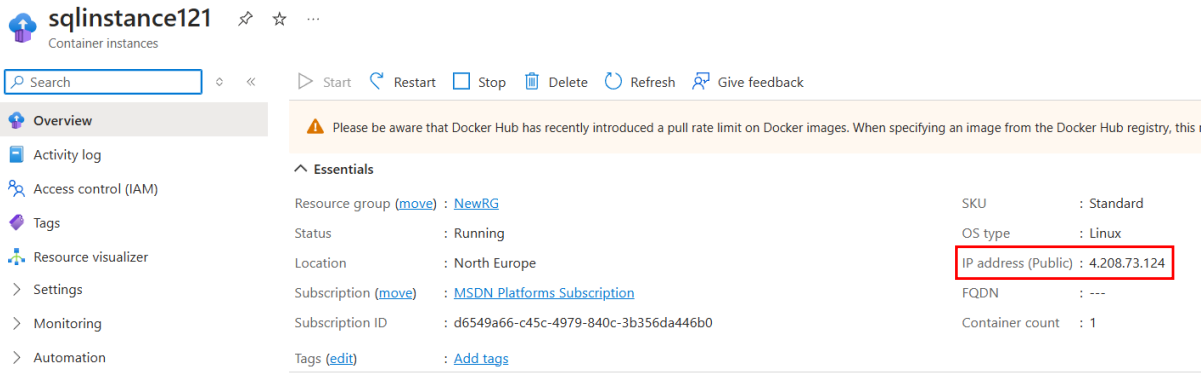
20. Now you can see that it has picked up our image itself which we just uploaded.



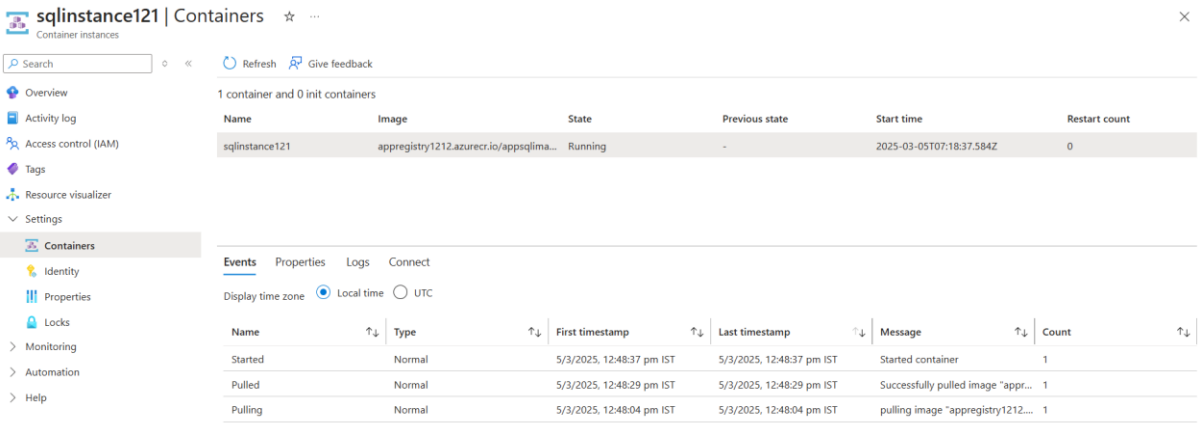
21. Then in the networking, for the ports use 3306 as shown below.



22. So, this container instance will also get a public IP address.



23. Now if you go to the containers tab you will see your container in the running state.

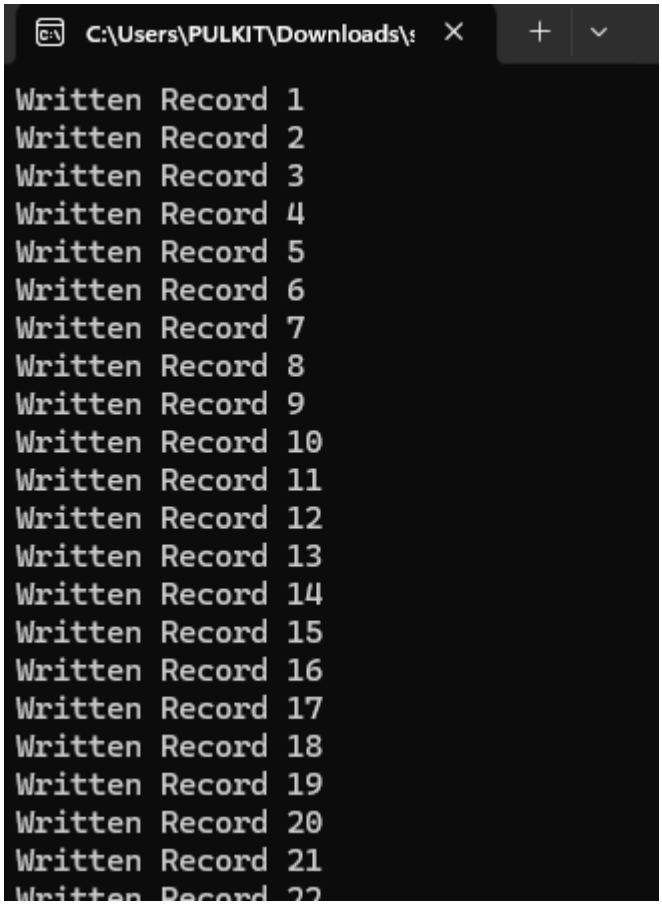


24. Now we will connect our container with the application we have used in the previous lab.

25. So, first copy the public IP address of your container instance and paste it on line number 10 of the first program. Basically, this IP address is working as a server name for our container instance.
26. Then in the User ID we have to write root and the password is Azure123. So, you can get the username and password from the docker file as well as the database name.
27. Once you are done with the settings just run the application.

```
using MySql.Data.MySqlClient;
using System.Data;
using System.Data.SqlClient;
using System.Reflection.Metadata;

string strFilePath = "C:\\tmp1\\Log.csv";
StreamReader? logReader =null;
int Id = 1;
MySqlConnection appdbConnection =
    new MySqlConnection("Server=4.208.73.124;UserID = root;Password=Azure123;Database=appdb");

MySqlParameter paramId = new MySqlParameter();
paramId.ParameterName = "@Id";

MySqlParameter paramOperationname = new MySqlParameter();
paramOperationname.ParameterName = "@Operationname";


MySqlParameter paramStatus = new MySqlParameter();
paramStatus.ParameterName = "@Status";


MySqlParameter paramEventcategory = new MySqlParameter();
paramEventcategory.ParameterName = "@Eventcategory";
```

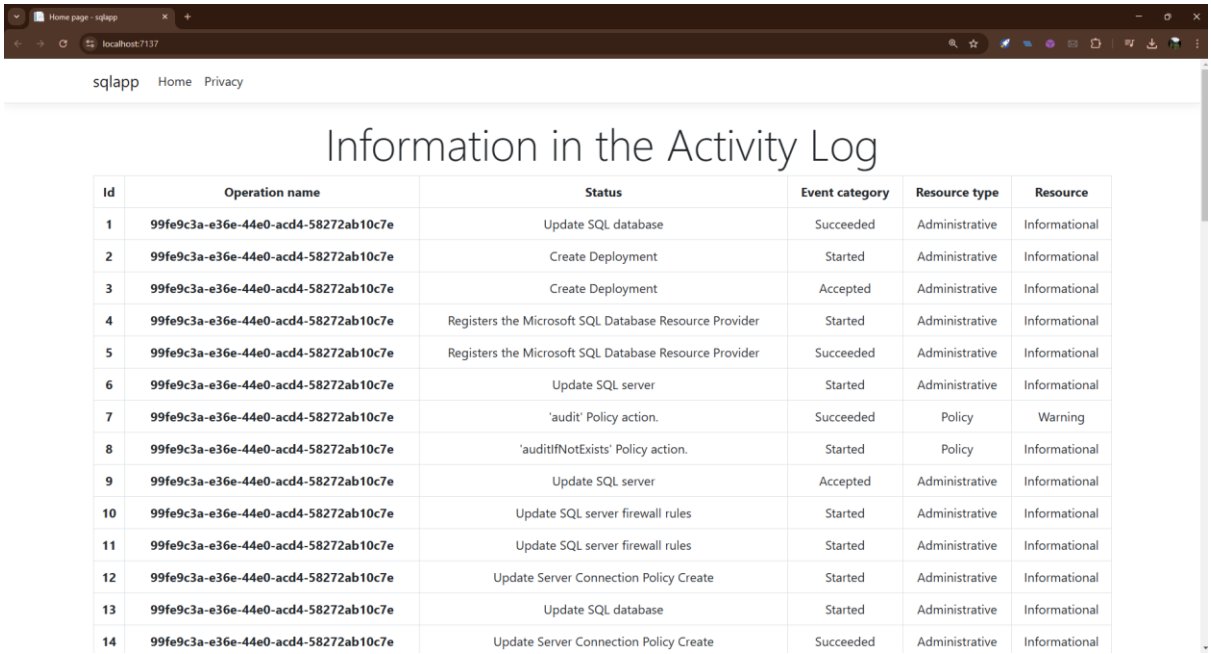
28. Now you can see that the records are being written in the table.



29. This is the second program that we have used to display the data onto the browser and after changing the server's name user ID and password you can see that it is also working properly.

```
using MySql.Data.MySqlClient;
using sqlapp.Models;


namespace sqlapp.Services
{

    2 references
    public class ActivityService
    {

        1 reference
        private MySqlConnection GetConnection()
        {

            return new MySqlConnection("Server=4.208.73.124;UserID = root;Password
        }
        1 reference
        public List<Activity> GetActivities()
        {
            List<Activity> _activity_lst = new List<Activity>();
            string _statement = "SELECT Id,Operationname,Status,Eventcategory,Reso
            MySqlConnection _connection = GetConnection();
```

No issues found Ln: 14 Ch: 105 SPC CRLF



Id	Operation name	Status	Event category	Resource type	Resource
1	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL database	Succeeded	Administrative	Informational
2	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Create Deployment	Started	Administrative	Informational
3	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Create Deployment	Accepted	Administrative	Informational
4	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Registers the Microsoft SQL Database Resource Provider	Started	Administrative	Informational
5	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Registers the Microsoft SQL Database Resource Provider	Succeeded	Administrative	Informational
6	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server	Started	Administrative	Informational
7	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	'audit' Policy action.	Succeeded	Policy	Warning
8	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	'auditIfNotExists' Policy action.	Started	Policy	Informational
9	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server	Accepted	Administrative	Informational
10	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server firewall rules	Started	Administrative	Informational
11	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server firewall rules	Started	Administrative	Informational
12	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update Server Connection Policy Create	Started	Administrative	Informational
13	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL database	Started	Administrative	Informational
14	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update Server Connection Policy Create	Succeeded	Administrative	Informational