PASS MARATHON

Azure SQL Containers

Running SQL Server containers in Azure

Andrew Pruski
Moderated By: Charith Suriyakula

Immerse yourself

in the data community

Access deep-dive technical sessions, learn best practices, and discover new tips and tricks

Gain the technical skills and connections to advance your data career

Attend

PASS Summit

PASS
SUMMIT 2018
NOV 6-9 | SEATTLE WA

PASS Summit is the largest conference for technical professionals who leverage the Microsoft Data Platform.

See everything PASS Summit has to offer at
PASSsummit.com

More info:

Andrew Pruski

SQL Server DBA & Data Platform MVP

dbafromthecold.com

@dbafromthecold

dbafromthecold@gmail.com

PASS MARATHON

Azure SQL Containers

Running SQL Server containers in Azure

Andrew Pruski
Moderated By: Charith Suriyakula

Agenda

Azure Container Registry

Azure Container Instances

Azure Container Services (ACS & AKS)

Tools – Docker for Windows

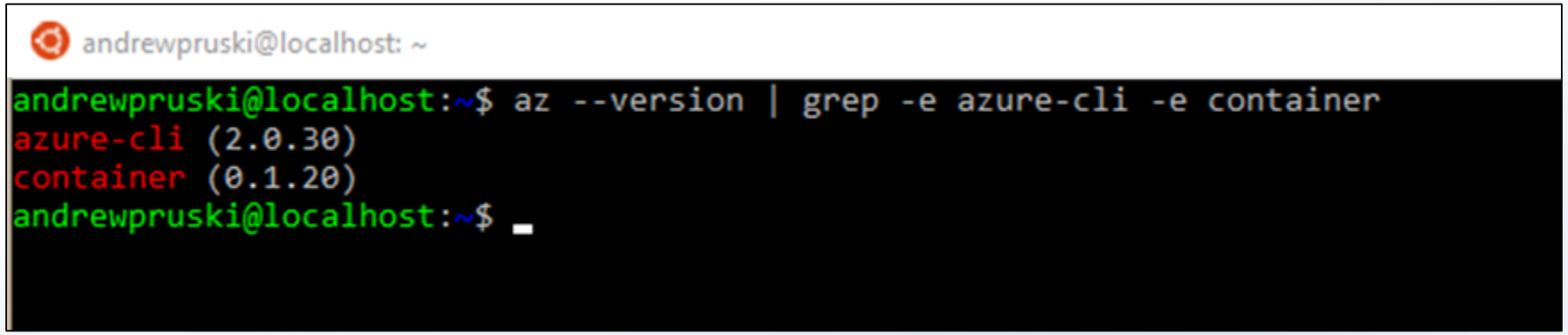
https://store.docker.com/editions/community/docker-ce-desktop-windows

Tools – Azure CLI

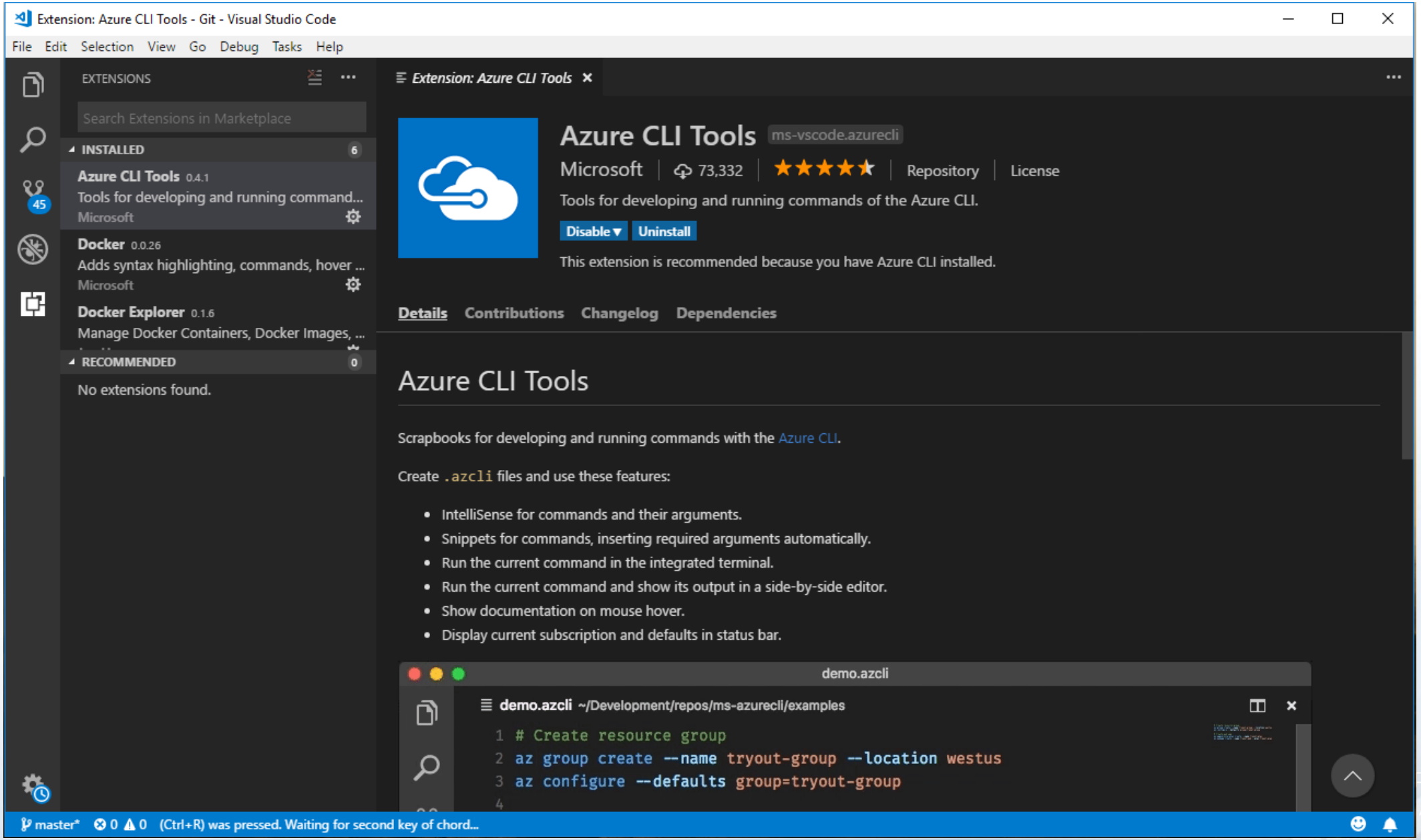
Command line tool for managing Azure resources

Available on Windows, Mac, or Linux (incl. WSL)

https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest



Tools – Visual Studio Code

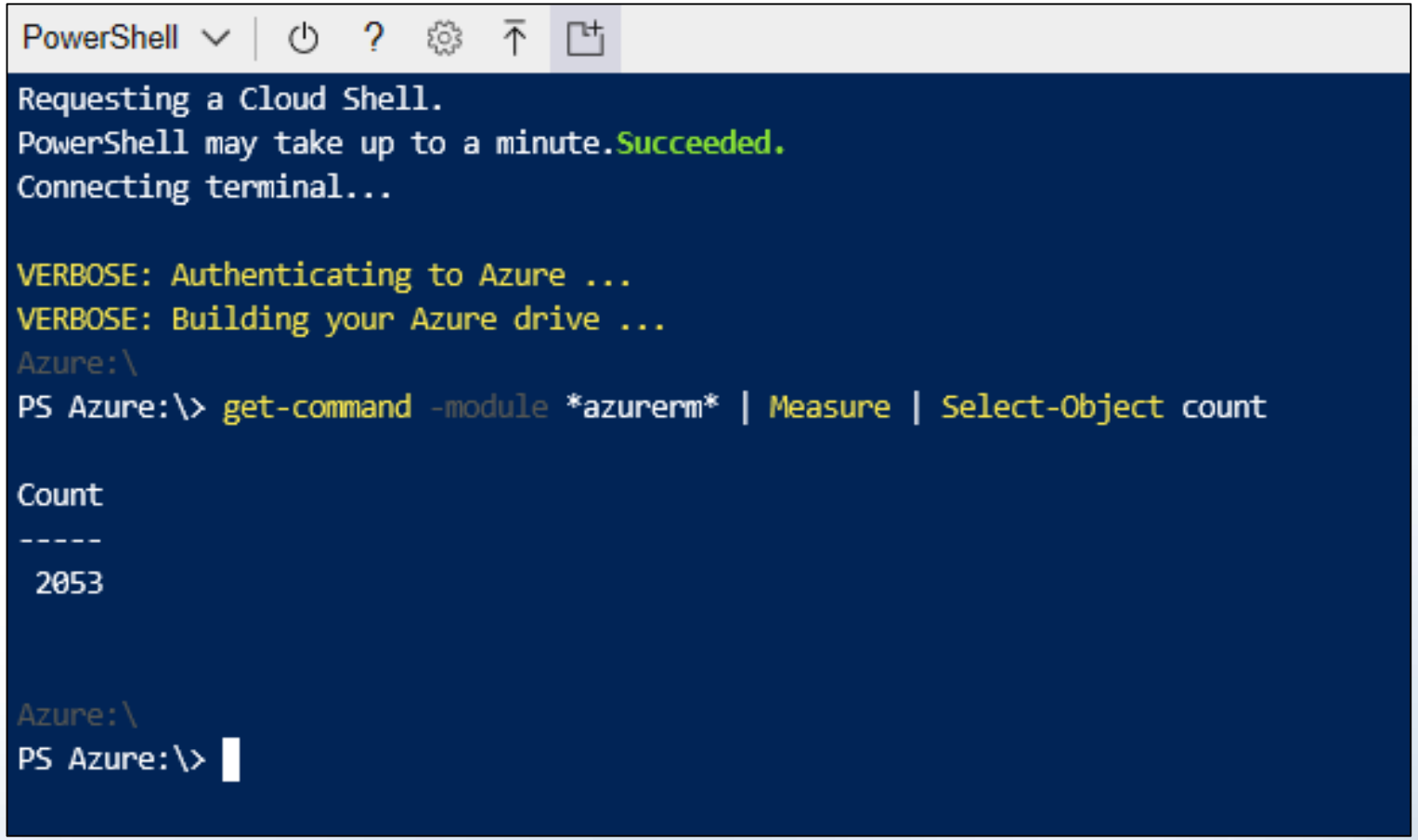


Tools – Powershell

AzureRM module

Install locally or in CloudShell

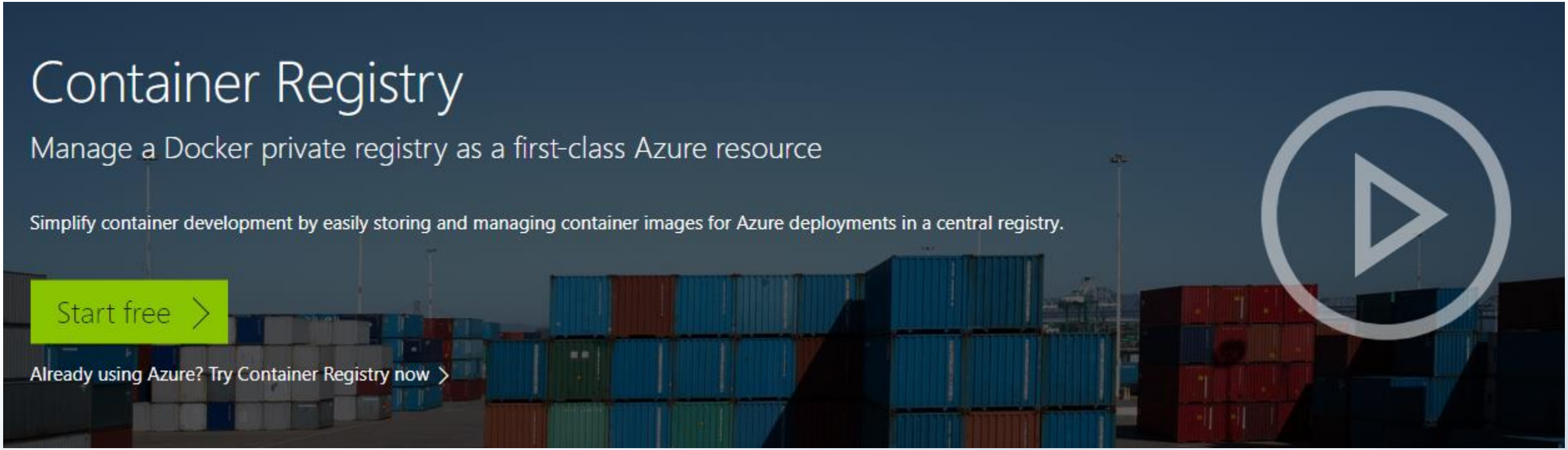
Huge amount of commands available



Azure Container Registry

Azure Container Registry

https://azure.microsoft.com/en-us/services/container-registry/



Azure Container Registry - Terminology

Registry

Service that stores container images

Repository

Groups of container images – Same name, identified by tags

Azure Container Registry - Features

Encryption

All images encrypted at rest

Geo-redundant storage

Replication of images

Geo-replication

Guards against total regional failure

ACR Build

Container image build service

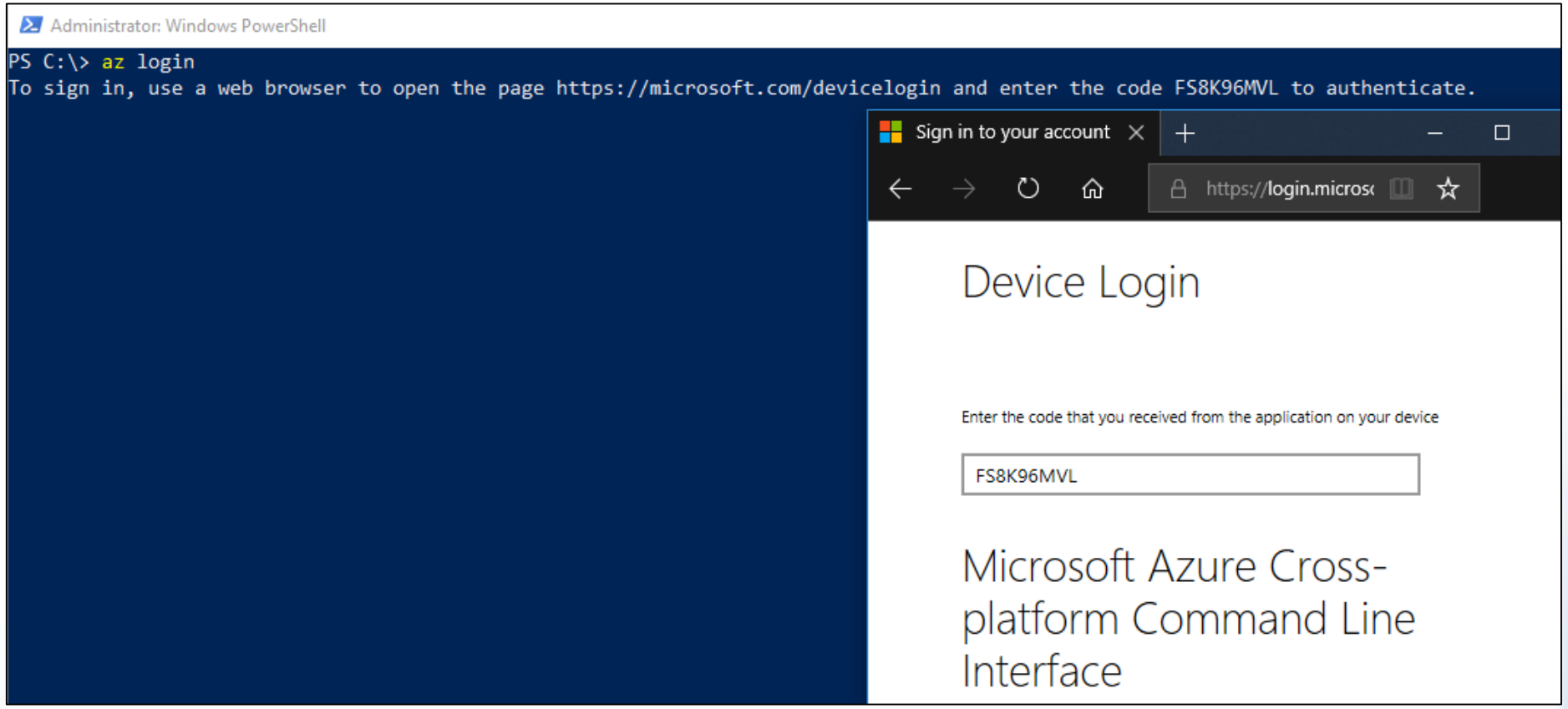
Azure Container Registry - SKUs

Type	Basic	Standard	Premium
Storage (GB)	10	100	500
ReadOps/min	1000	3000	10000
WriteOps/min	100	500	2000
Download MBps	30	60	100
Upload MBps	10	20	50
Webhooks	2	10	100
Geo-replication	No	No	Yes

https://docs.microsoft.com/en-us/azure/container-registry/container-registry-skus

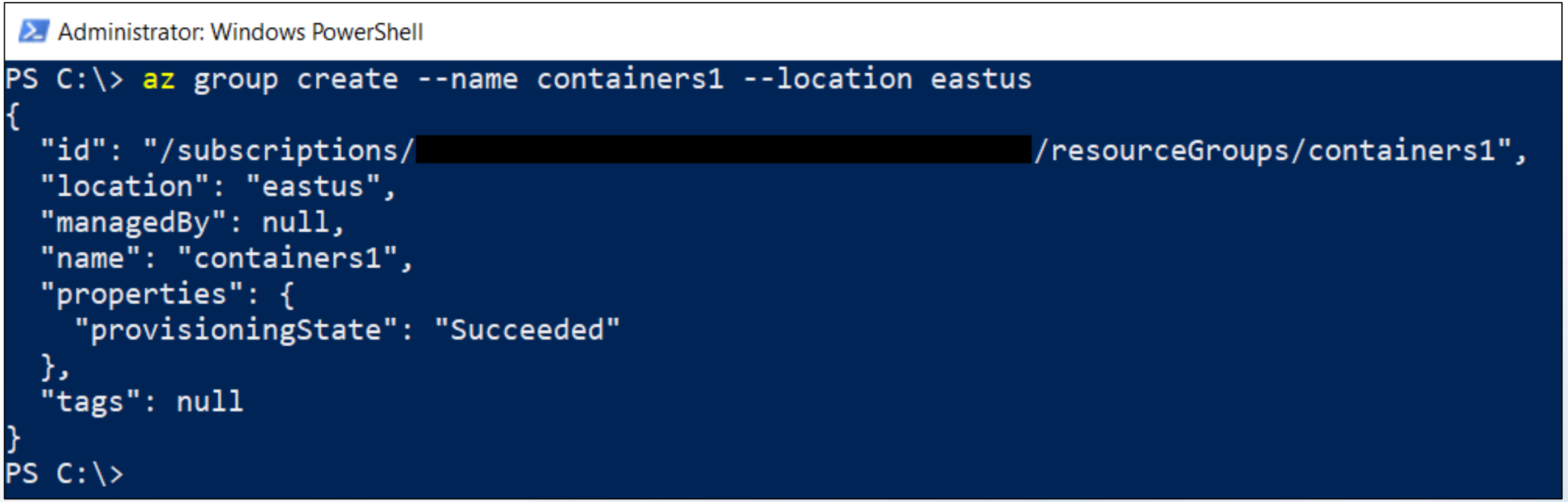
ACR - Login

```
az login
```



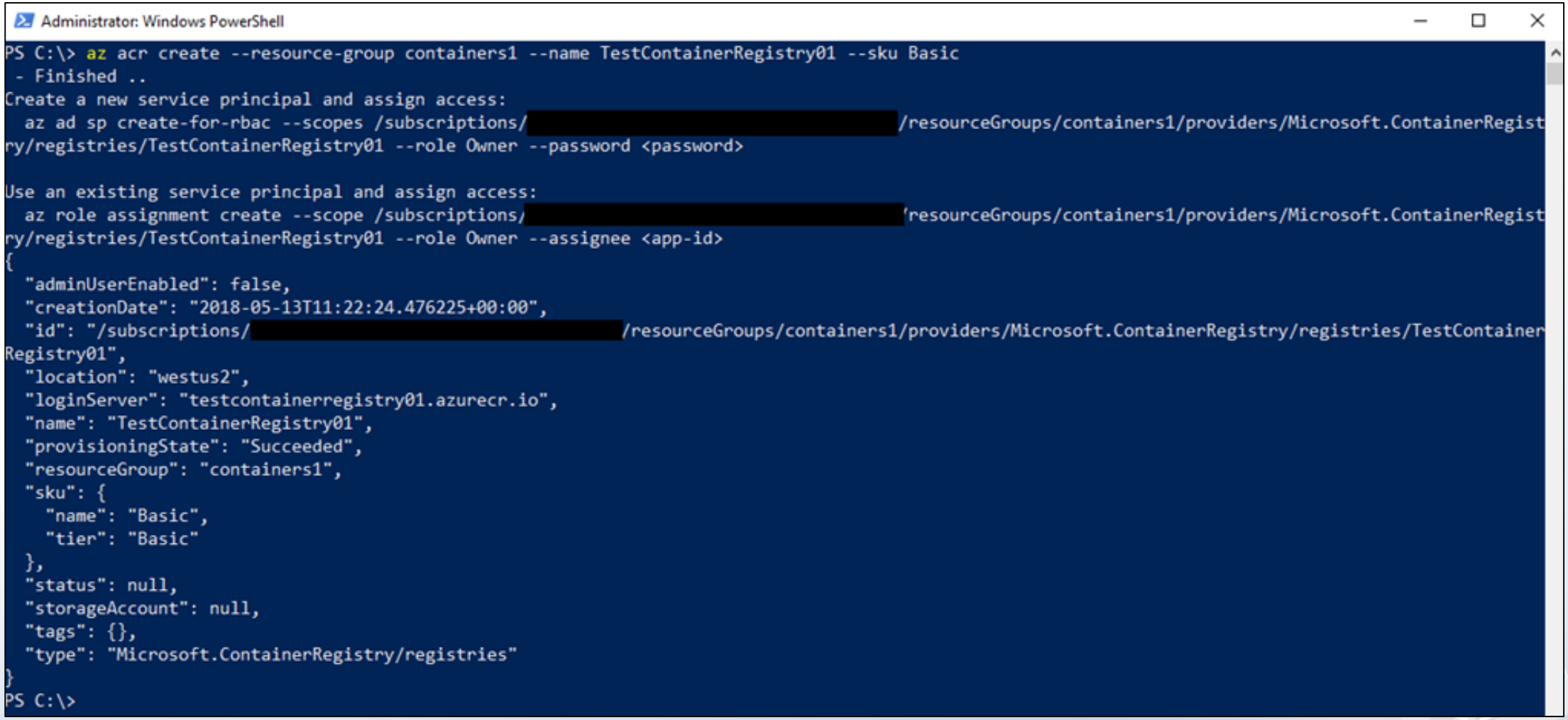
ACR – Create Resource Group

```
az group create --name containers1 --location eastus
```



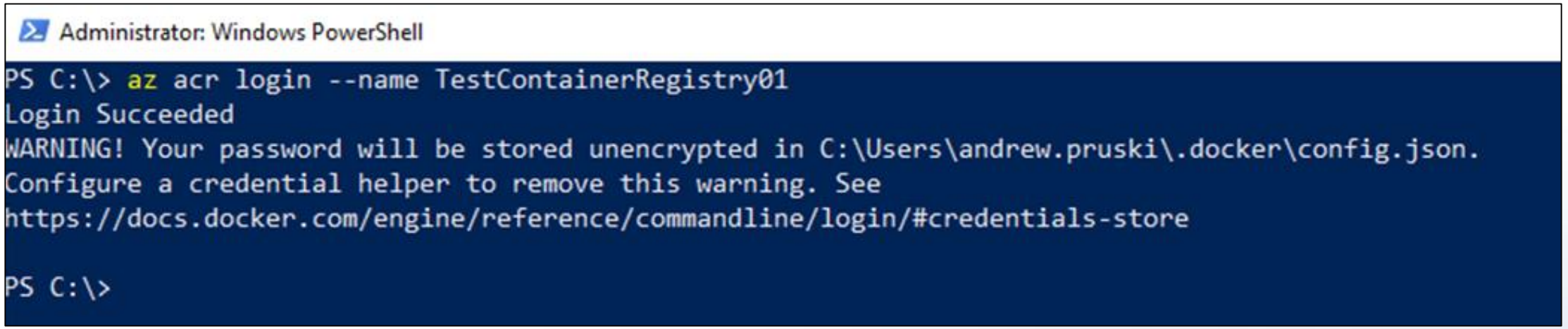
ACR – Create Container Registry

```
az acr create --resource-group containers1 `
    --name TestContainerRegistry01 --sku Basic
```



ACR – Log into Registry

```
az acr login --name TestContainerRegistry01
```



ACR – Get Registry Details

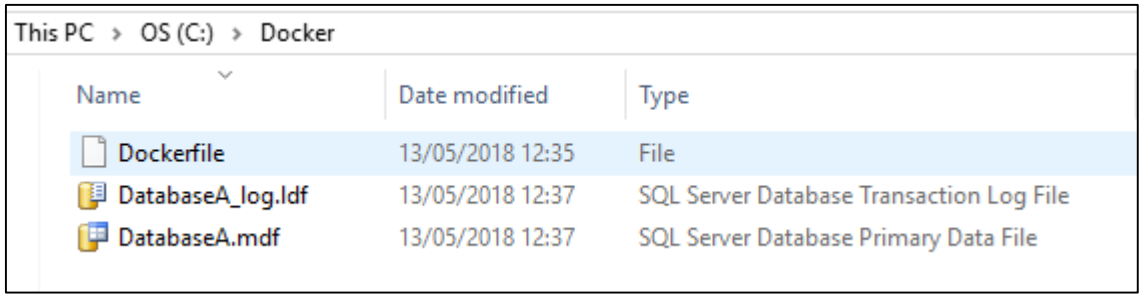
```
az acr list --resource-group containers1
```

Administrator: Windows PowerShell

```
PS C:\> az acr list --resource-group containers1
[
  {
    "adminUserEnabled": false,
    "creationDate": "2018-05-13T11:22:24.476225+00:00",
    "id": "/subscriptions/[redacted]/resourceGroups/containers1/providers/Microsoft.ContainerRegistry/registries/TestContainerRegistry01",
    "location": "westus2",
    "loginServer": "testcontainerregistry01.azurecr.io",
    "name": "TestContainerRegistry01",
    "provisioningState": "Succeeded",
    "resourceGroup": "containers1",
    "sku": {
      "name": "Basic",
      "tier": "Basic"
    },
    "status": null,
    "storageAccount": null,
    "tags": {},
    "type": "Microsoft.ContainerRegistry/registries"
  }
]
PS C:\>
```

ACR – Dockerfile

```
FROM microsoft/mssql-server-linux:latest

CMD  mkdir /var/opt/sqlserver

COPY DatabaseA.mdf /var/opt/sqlserver
COPY DatabaseA_log.ldf /var/opt/sqlserver

ENV MSSQL_BACKUP_DIR="/var/opt/sqlserver"
ENV MSSQL_DATA_DIR="/var/opt/sqlserver"
ENV MSSQL_LOG_DIR="/var/opt/sqlserver"

HEALTHCHECK --interval=10s  \
CMD  /opt/mssql/bin/sqlservr & \
        /opt/mssql-tools/bin/sqlcmd -S . -U sa -P $SA_PASSWORD \
                -Q "CREATE  DATABASE  [DatabaseA] ON (FILENAME = '/var/opt/sqlserver/DatabaseA.mdf'),
                        (FILENAME = '/var/opt/sqlserver/DatabaseA_log.ldf') FOR ATTACH"
```



ACR – Build image

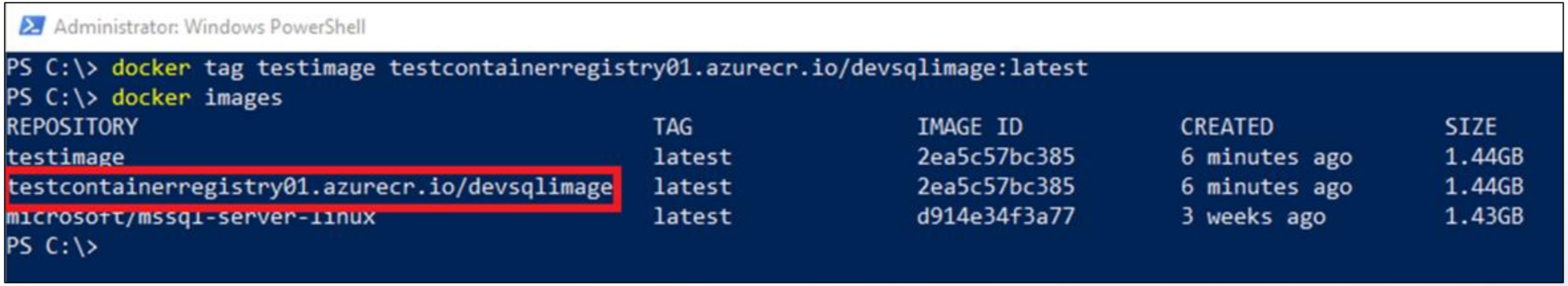
```
docker build -t testimage C:\docker
```

Administrator: Windows PowerShell

```
PS C:\> docker build -t testimage C:\docker
Sending build context to Docker daemon  16.78MB
Step 1/8 : FROM microsoft/mssql-server-linux:latest
 ---> d914e34f3a77
Step 2/8 : RUN mkdir /var/opt/sqlserver
 ---> Running in 426429c75657
Removing intermediate container 426429c75657
 ---> e38f4516df30
Step 3/8 : COPY DatabaseA.mdf /var/opt/sqlserver
 ---> 4a3bce6da149
Step 4/8 : COPY DatabaseA_log.ldf /var/opt/sqlserver
 ---> 9d75d3d2efce
Step 5/8 : ENV MSSQL_BACKUP_DIR="/var/opt/sqlserver"
 ---> Running in f2abb95e91f7
Removing intermediate container f2abb95e91f7
 ---> 13a5c79f0808
Step 6/8 : ENV MSSQL_DATA_DIR="/var/opt/sqlserver"
 ---> Running in 1cec1ed80a1f
Removing intermediate container 1cec1ed80a1f
 ---> 31ad98e47754
Step 7/8 : ENV MSSQL_LOG_DIR="/var/opt/sqlserver"
 ---> Running in 3f40faa1a9d9
Removing intermediate container 3f40faa1a9d9
 ---> 528d7d166840
Step 8/8 : HEALTHCHECK --interval=10s   CMD /opt/mssql/bin/sqlservr &   /opt/mssql-tools/bin/sqlcmd -S . -U sa -P $SA_PASSWORD          -Q "CREATE DAT
ABASE [DatabaseA] ON (FILENAME = '/var/opt/sqlserver/DatabaseA.mdf'),(FILENAME = '/var/opt/sqlserver/DatabaseA_log.ldf') FOR ATTACH"
 ---> Running in 57a1da7b0ed7
Removing intermediate container 57a1da7b0ed7
 ---> 22b5248f1988
Successfully built 22b5248f1988
Successfully tagged testimage:latest
```

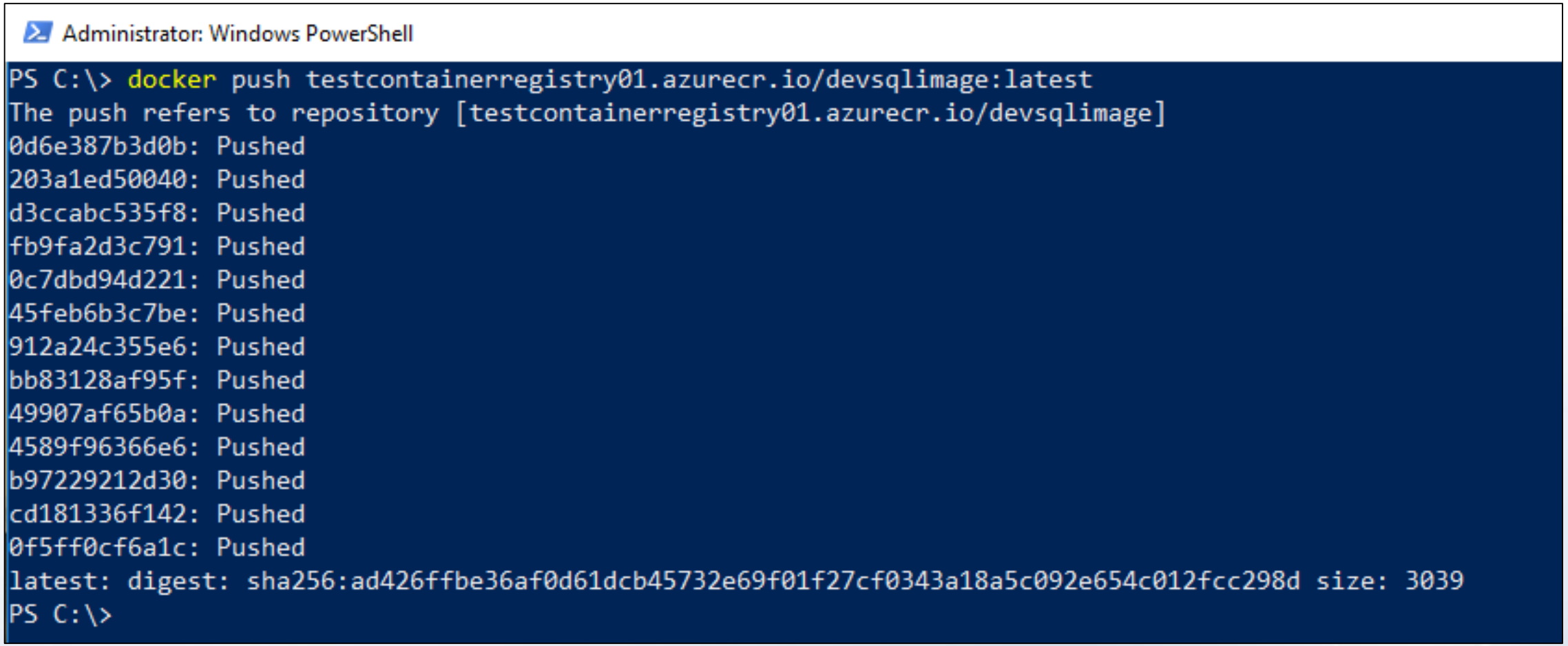
ACR – Tag Custom Image

```
docker tag testimage `
      testcontainerregistry01.azurecr.io/devsqlimage:latest
```



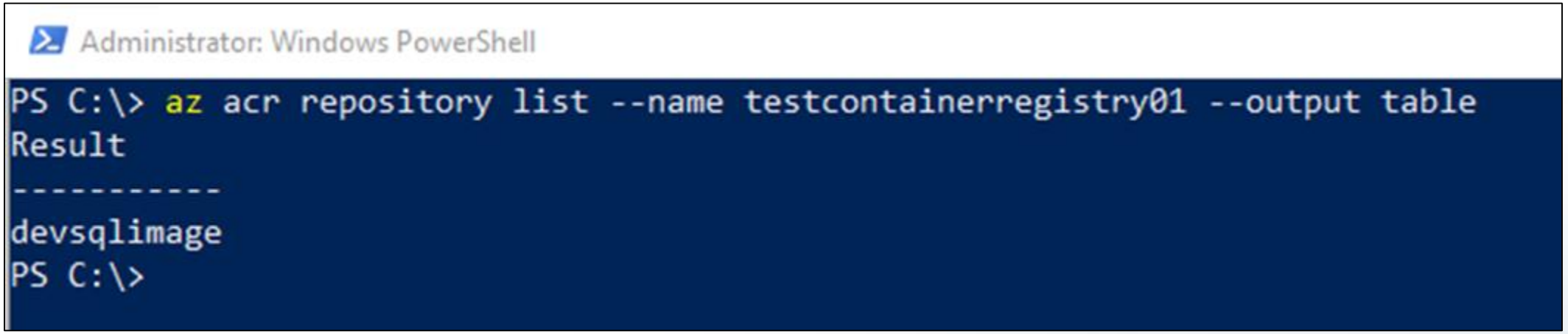
ACR – Push Image

```
docker push testcontainerregistry01.azurecr.io/devsqlimage:latest
```



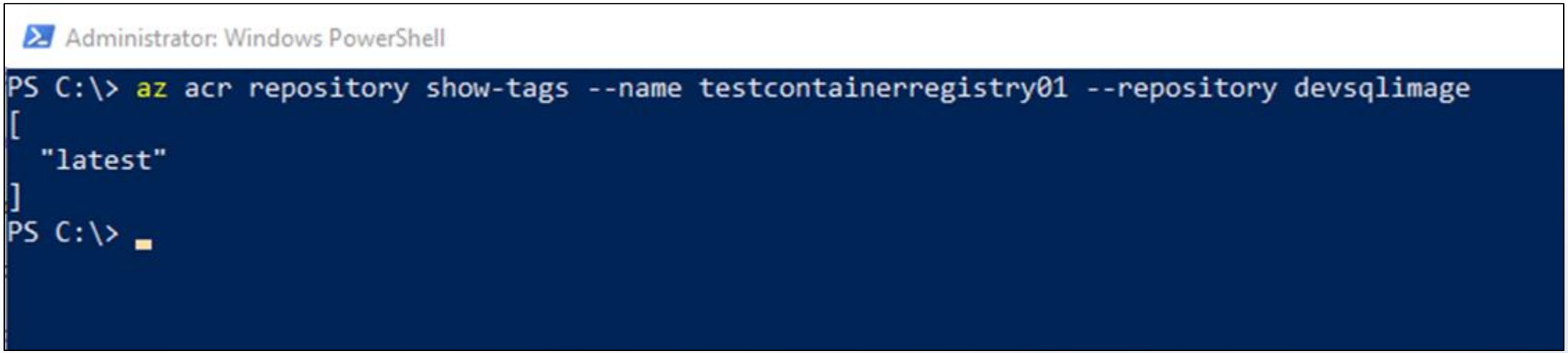
ACR – List Images

```
az acr repository list --name testcontainerregistry01 --output table
```



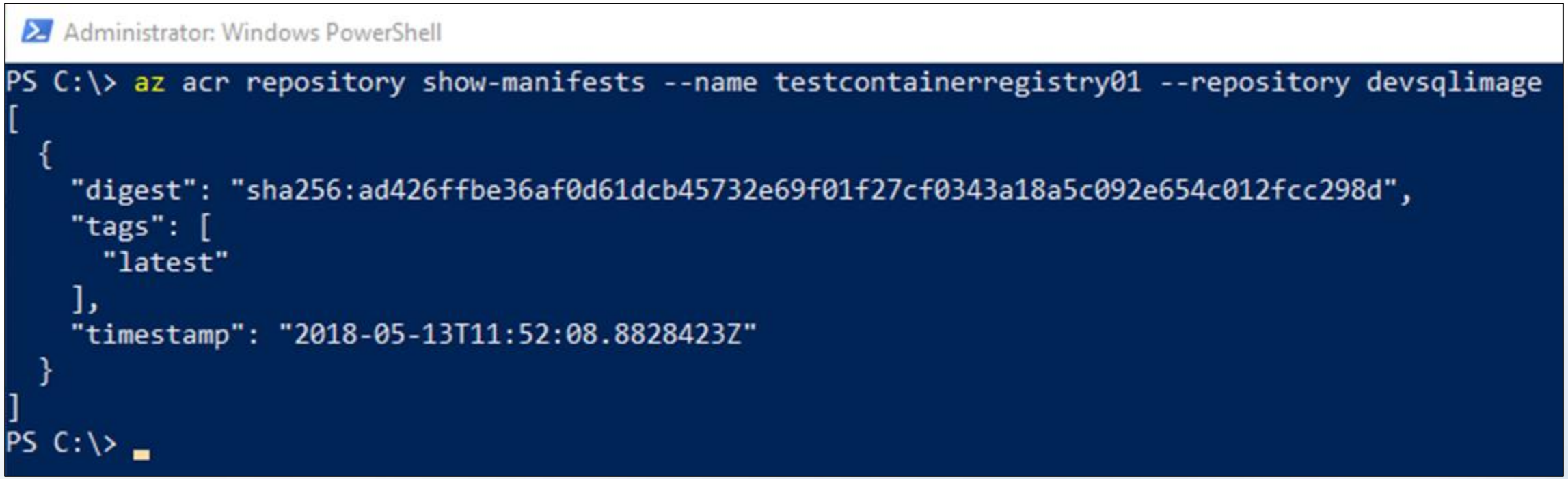
ACR – Show Tags

```
az acr repository show-tags `
    --name testcontainerregistry01 `
        --repository devsqlimage
```



ACR – Show Manifest

```
az acr repository show-manifests `
    --name testcontainerregistry01 `
        --repository devsqlimage
```

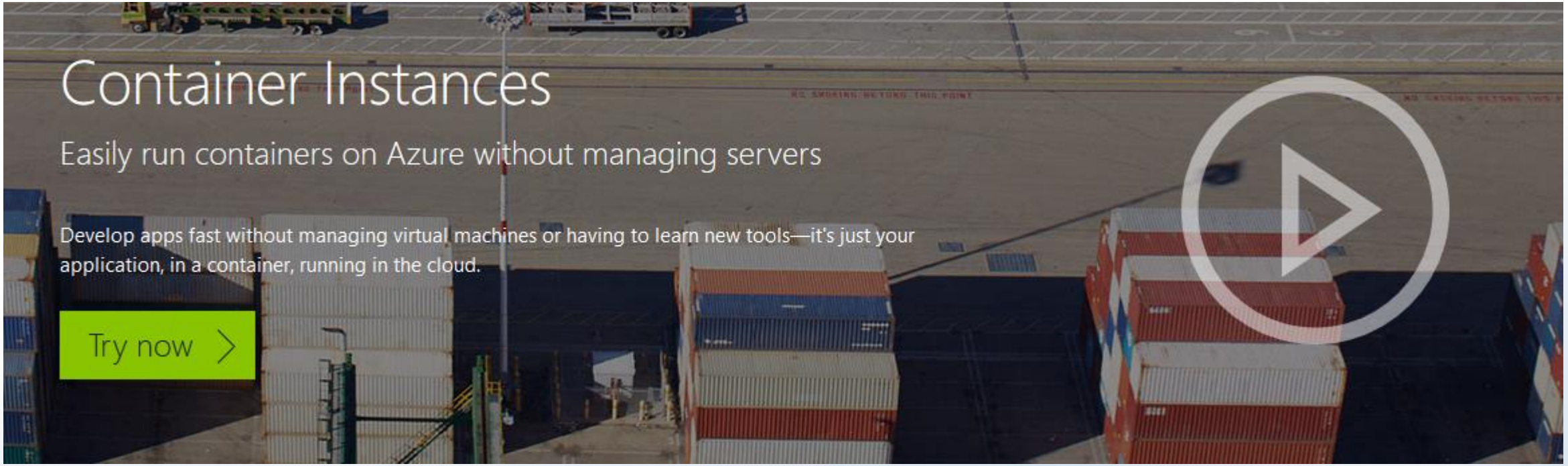


Demo

Azure Container Instances

Azure Container Instances

https://azure.microsoft.com/en-us/services/container-instances/



Azure Container Instances

Running containers without servers

No need to manage VMs

Quick deployment

Deployed via the CLI, powershell, or Azure Portal

Billed by the second

ACI - Options

- Linux & Windows containers
- Containers exposed directly to the internet
  - IP Address and FQDN
- Hypervisor level isolation
- Persistent storage
  - Azure files shares

ACI – Container Groups

Similar in concept to K8s pods

Multiple containers running on the same host

Share IP address, containers exposed on ports

Supports mounting external volumes

ACI – Get ACR Credentials

```
# enable admin
az acr update -n TestContainerRegistry01 --admin-enabled true


# get credentials
az acr credential show -n TestContainerRegistry01
```

ACI – Create Keyvault

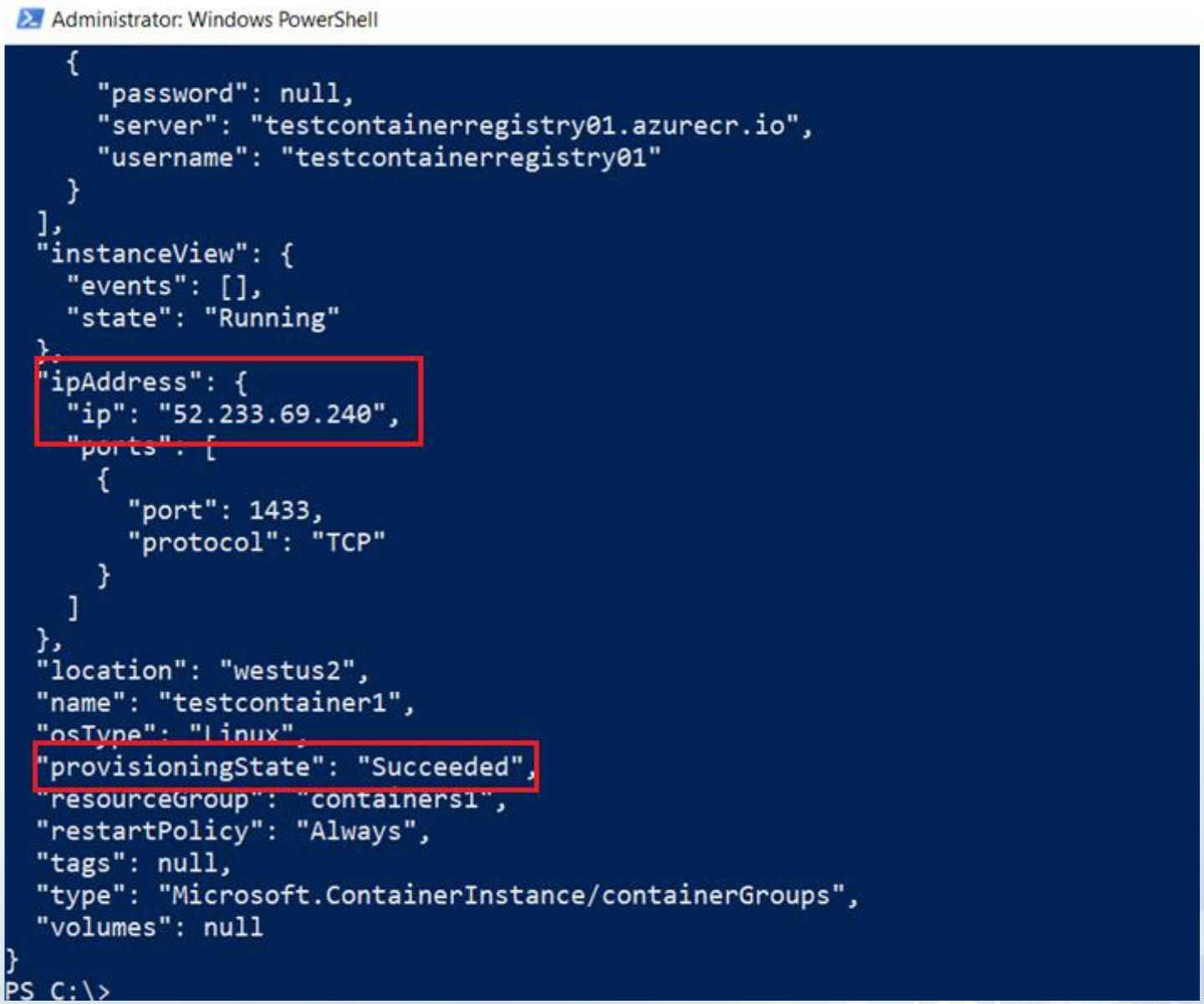
```
az keyvault create --resource-group containers1 --name aptestkeyvault01

az keyvault secret set `
  --vault-name aptestkeyvault01 `
  --name testcontainerregistry01-pull-pwd `
  --value $(az ad sp create-for-rbac `
      --name testcontainerregistry01-pull `
      --scopes $(az acr show --name testcontainerregistry01 --query id –output tsv)`
                  --role reader --query password --output tsv)

az keyvault secret set `
  --vault-name aptestkeyvault01 `
  --name testcontainerregistry01-pull-usr `
  --value $(az ad sp show --id http://testcontainerregistry01-pull `
              --query appId --output tsv)
```

ACI – Create Container

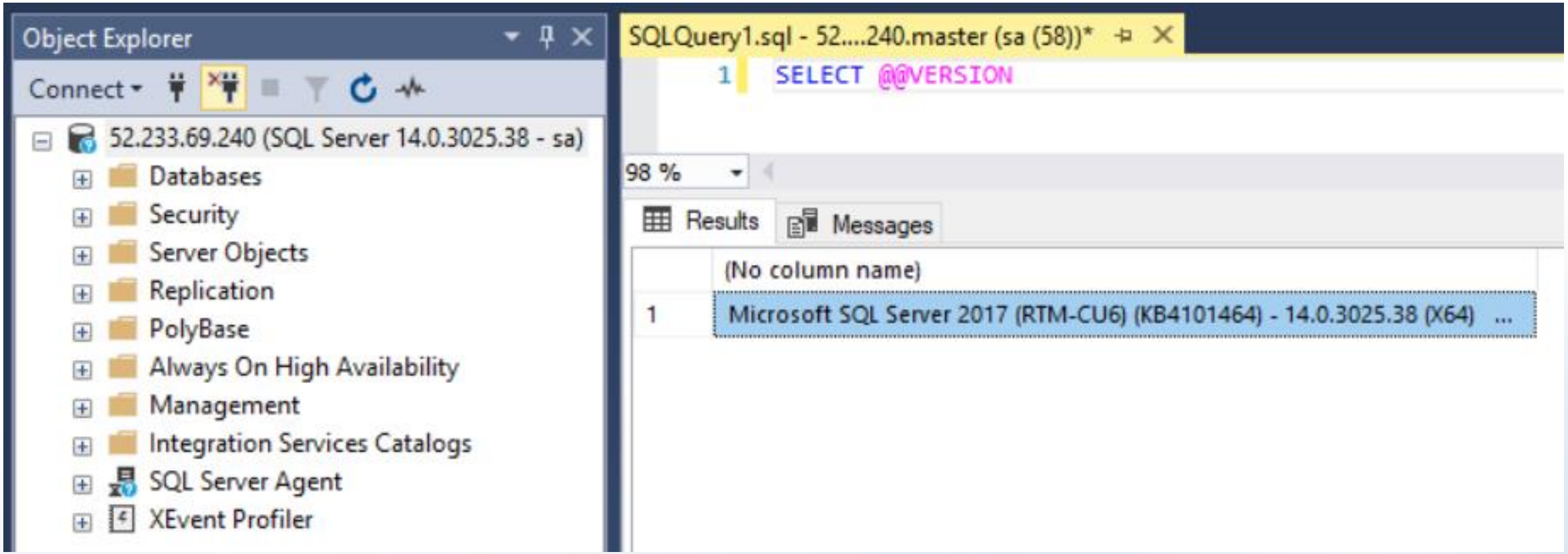
```
az container create `
  --resource-group containers1 `
  --image testcontainerregistry01.azurecr.io/devsqlimage:latest `
  --registry-username $(az keyvault secret show `
                        --vault-name aptestkeyvault01 `
                        --name testcontainerregistry01-pull-usr `
                        --query value --output tsv) `
  --registry-password $(az keyvault secret show `
                        --vault-name aptestkeyvault01 `
                        --name testcontainerregistry01-pull-pwd `
                        --query value --output tsv) `
  --name testcontainer1 `
  --cpu 2 --memory 4 `
  --environment-variables ACCEPT_EULA=Y SA_PASSWORD=Testing1122 `
  --ip-address public `
  --ports 1433
```

ACI – Get Container Status

```
az container show `
  --name testcontainer1
       --resource-group containers1
```



ACI – Connect to SQL Server

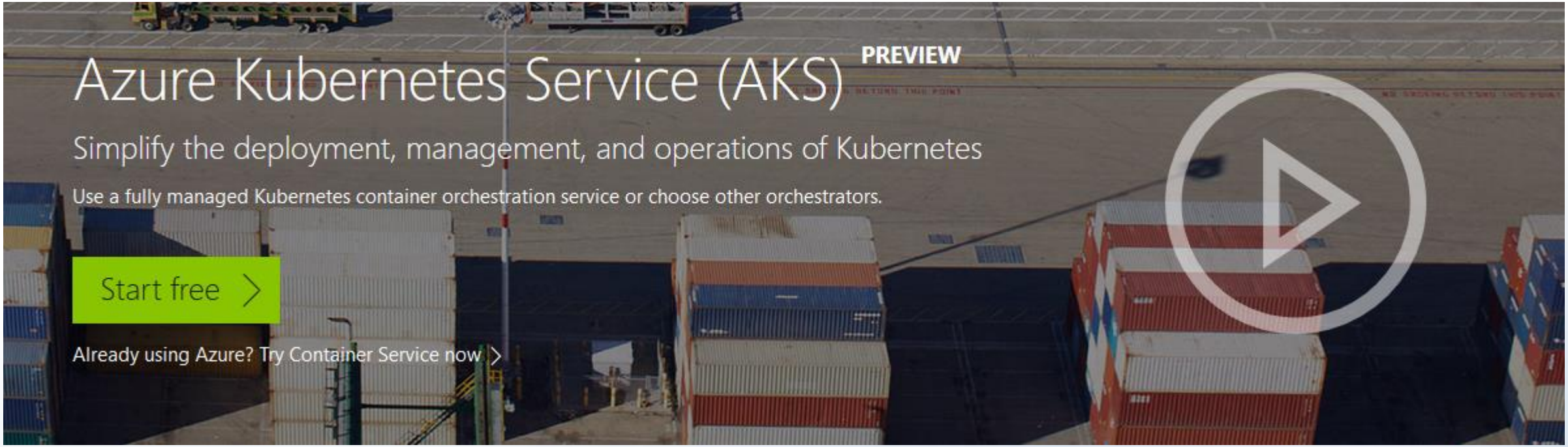


Demo

Azure Container Services

Azure Container Services

https://azure.microsoft.com/en-us/services/container-service/



Azure Container Services

Two flavours

- Azure Container Services (ACS)
- Azure Container Services (AKS)

ACS provides container hosting using DC/OS, Swarm, or K8s

AKS is specifically built to implement Kubernetes

Kubernetes

Open Source system for managing containers

Deployed as a cluster containing a master and multiple nodes

Pods hold containers running on the nodes

Services define/allow access to sets of pods

Deployments created and managed via Kubectl

Azure Container Services (AKS)

Simplifies deployment of Kubernetes clusters in Azure

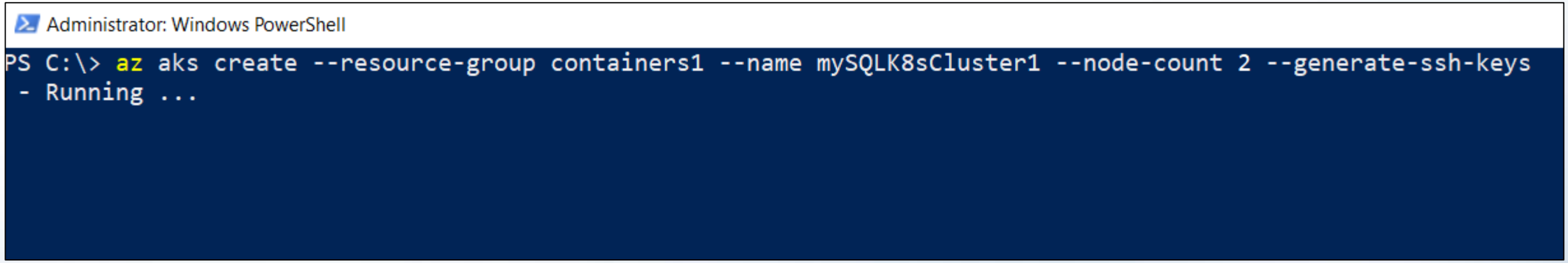
Cluster can be spun up with one line of code

Applications deployed to cluster via yaml files

Managed by Azure-CLI/powershell and kubectl

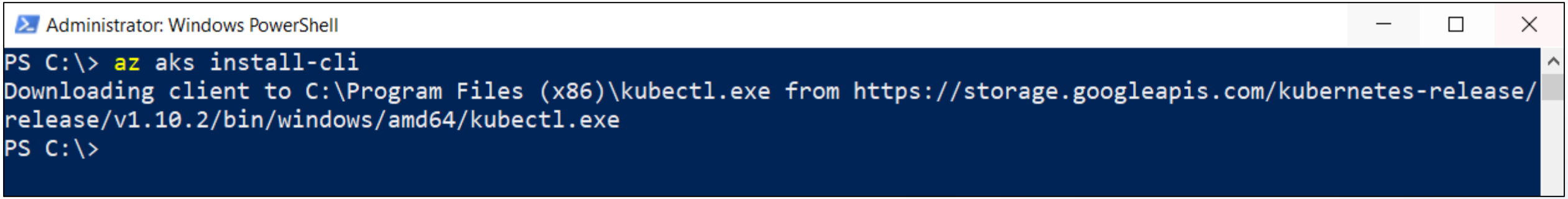
AKS – Create Cluster

```
az aks create `
    --resource-group containers1 `
        --name mySQLK8sCluster1 `
            --node-count 2 `
                --generate-ssh-keys
```



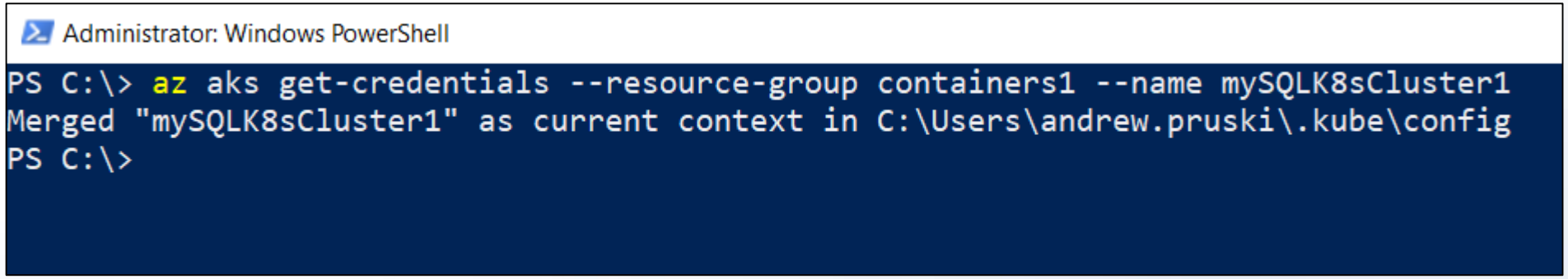
AKS – Install kubectl

```
az aks install-cli
```



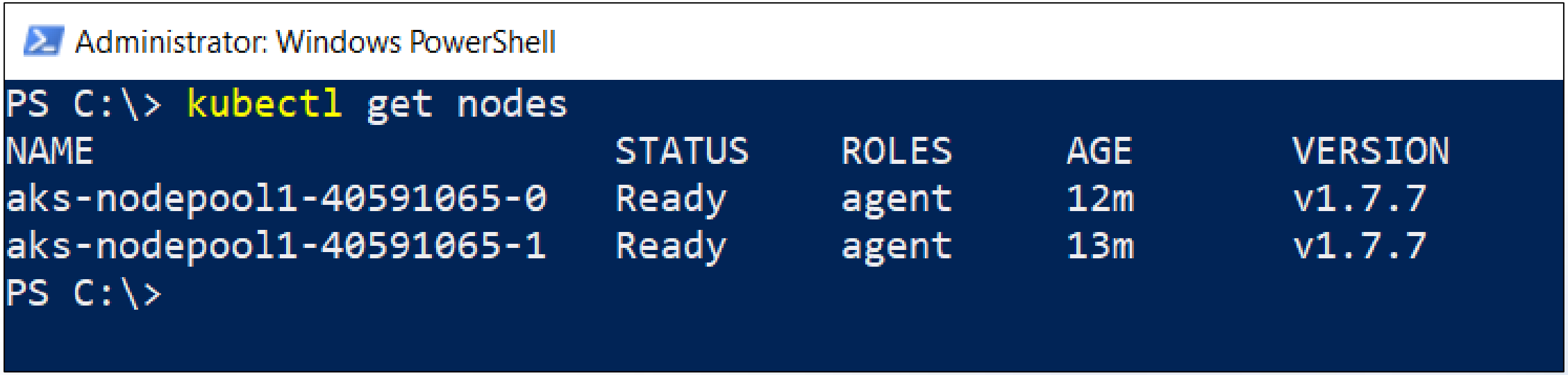
AKS – Get Cluster Credentials

```
az aks get-credentials `
    --resource-group containers1 `
        --name mySQLK8sCluster1
```



AKS – View Cluster Nodes

```
kubectl get nodes
```



AKS – Get Cluster & ACR Details

Get AKS Service Principal ID

```
az aks show `
      --resource-group containers1 `
            --name mySQLK8sCluster1 `
                  --query "servicePrincipalProfile.clientId" --output tsv
```

Get ACR Resource ID

```
az acr show `
      --name TestContainerRegistry02 `
            --resource-group containers1 `
                  --query "id" `
                        --output tsv
```

AKS – Create Role to Deploy

Create role to grant access

```
az role assignment create `
    --assignee <<CLIENTID>> `
        --role Reader `
            --scope <<ACRID>>
```

AKS – yaml file

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: sqlserver
  labels:
    app: sqlserver
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: sqlserver
```

AKS – yaml file

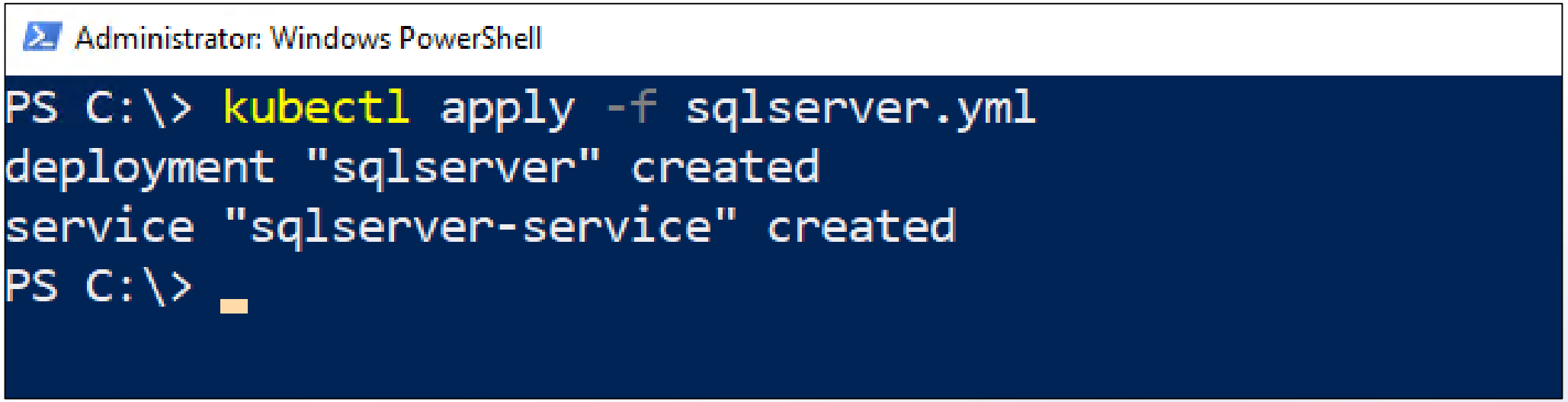
```
spec:
    containers:
    - name: sqlserver1
      image: testcontainerregistry01.azurecr.io/devsqlimage:latest
      ports:
      - containerPort: 1433
      env:
      - name: SA_PASSWORD
        value: "Testing1122"
      - name: ACCEPT_EULA
        value: "Y"
```

AKS – yaml file

```
apiVersion: v1
  kind: Service
metadata:
  name: sqlserver-service
spec:
  ports:
  - name: sqlserver
     port: 1433
     targetPort: 1433
  selector:
    name: sqlserver
    type: LoadBalancer
```

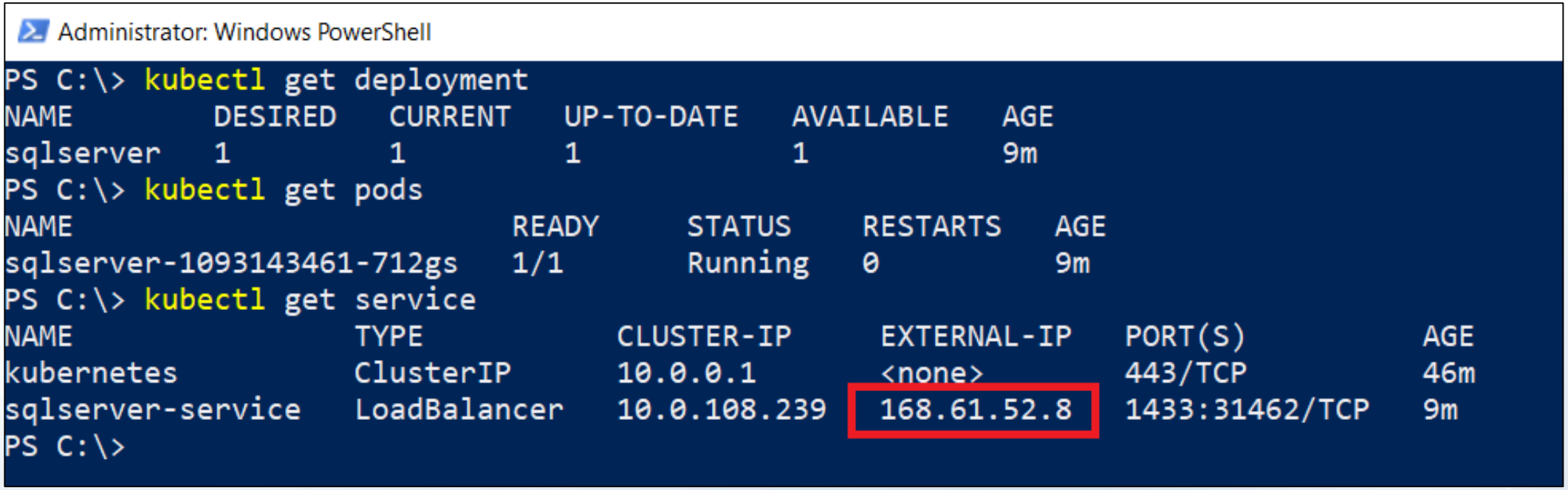
AKS – Deploy to Cluster

```
kubectl create -f sqlserver.yml
```

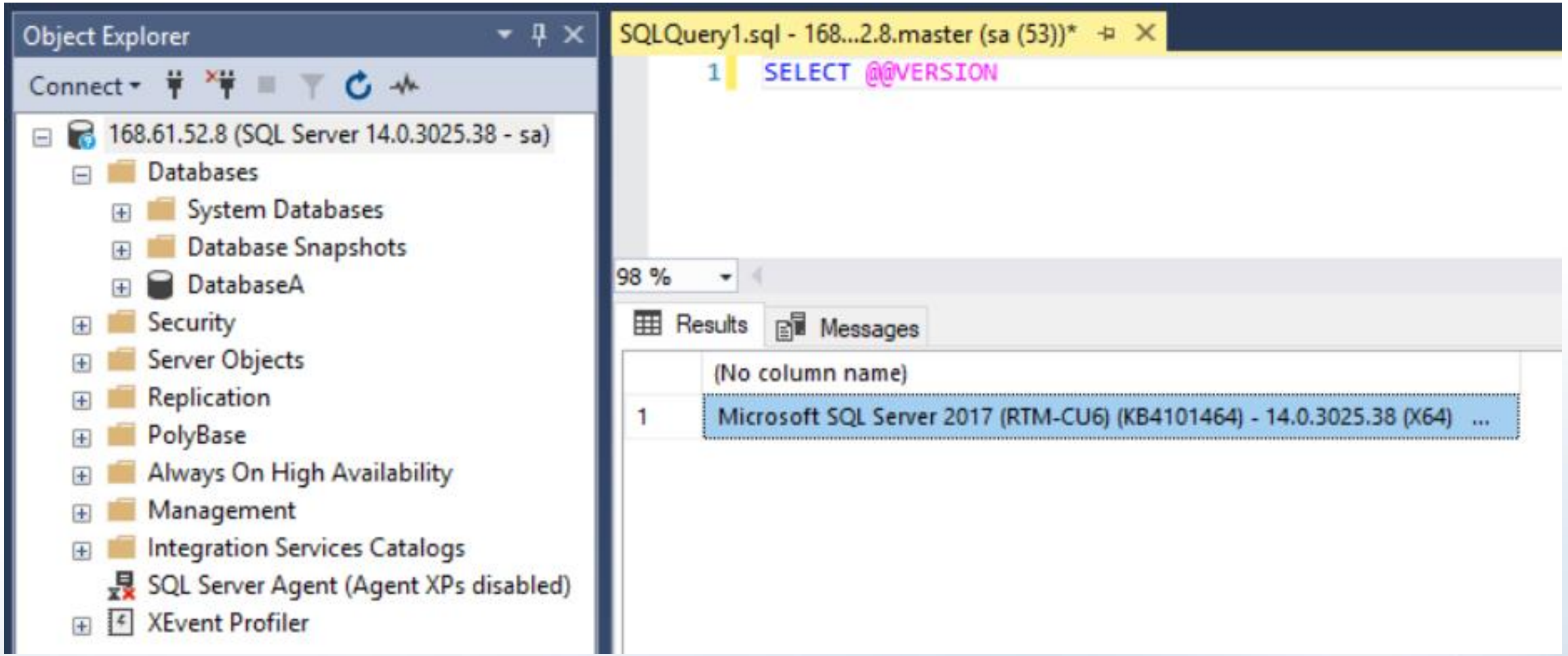


AKS – View Deployment Information

```
kubectl get deployments
kubectl get pods
Kubectl get service
```

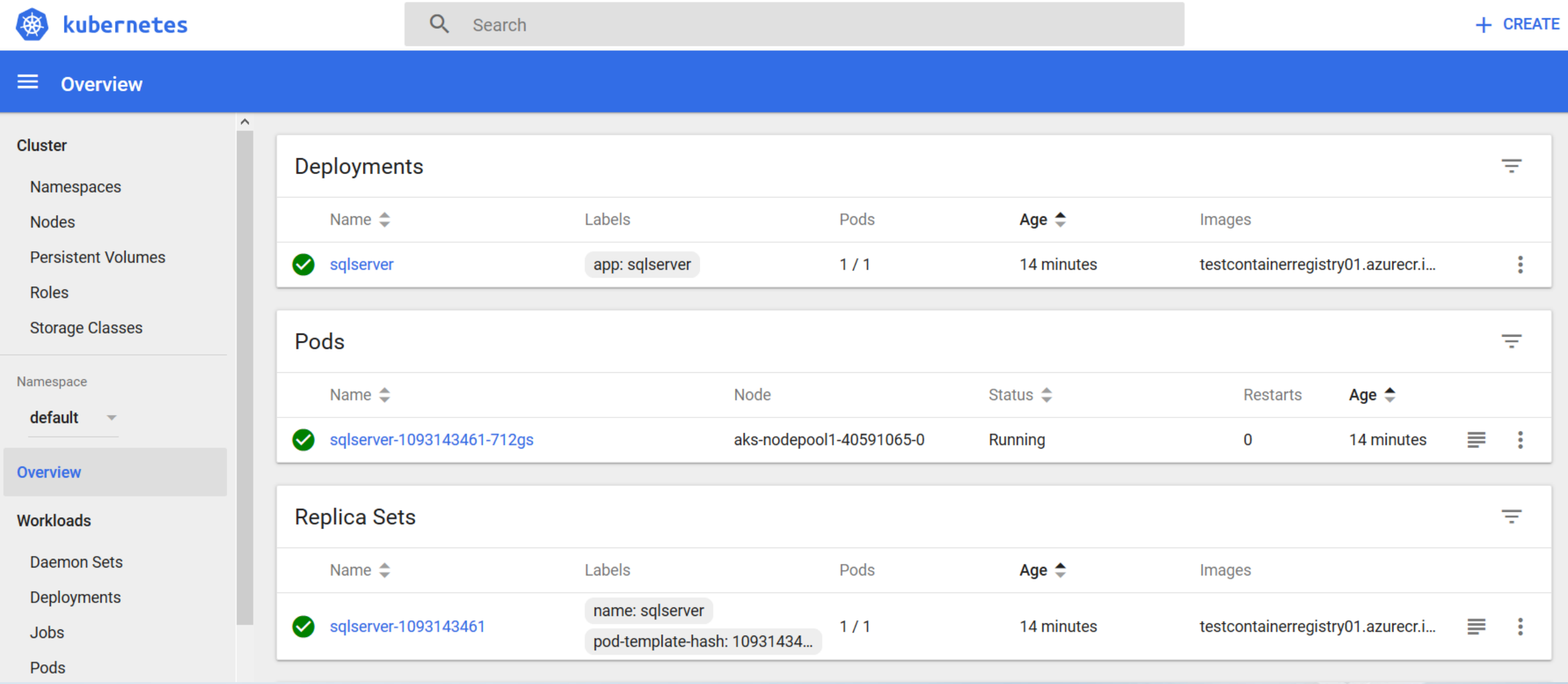


AKS – Connect to SQL Server



AKS – Connect to K8s Dashboard

```
az aks browse --resource-group containers1 --name mySQLK8sCluster1
```



Demo

Resources

https://azure.microsoft.com/en-us/services/container-registry/

https://azure.microsoft.com/en-us/services/container-instances/

https://azure.microsoft.com/en-us/services/container-service/

https://dbafromthecold.com/2017/03/15/summary-of-my-container-series/

https://github.com/dbafromthecold/ContainersInTheCloud

Thank you for attending

Learn more from Andrew Pruski

@dbafromthecold

dbafromthecold@gmail.com


@sqlpass #sqlpass


@PASScommunity


PASS
MARATHON