8. Microservice Architecture
Containerization

Agenda

- Containers

Containers and Microservices

- Containers and containerization are not, strictly speaking, necessary when building microservices
- Docker and Kubernetes (respectively).
- We’ll build a Docker image, share it on Docker Hub, and then learn how to deploy it to AKS (Azure Kubernetes Service) with Kubernetes

Container

- lightweight virtual machine
- virtualized host operating system
- Containers can be used beyond microservices

It Works on My Machine!

- Containers ship your machine to the customer (With a Docker image)
- Containers allow developers to run a software package locally with everything needed built into the container.
- This reduces the need to make sure the correct versions of all the dependencies are synchronized

Docker Compose

- a composition file (for Docker Compose this is a file in YAML format),
- the systems, servers, and software that a new developer needs to get started on a project can be declared in a small YAML file
- YAML files can be checked into source control

YAML

```yaml
version: '3.4'

services:
  mywebapp:
    image: ${DOCKER_REGISTRY-}mywebapp
    build:
      context: .
      dockerfile: MyWebApp/Dockerfile
  mssql:
    image: chriseaton/adventureworks:oltp
    environment:
    - ACCEPT_EULA=Y
    - SA_PASSWORD=myPassword123$
    ports:
    - "1433:1433"
  cb:
    image: couchbase:enterprise-7.0.0
    ports:
    - "8091:8091"
```

```yaml
services:
  mywebapp:
    image: <imagename>
  mssql:
    image: <imagename>
  cb:
    image: <imagename>
```

Docker-Compose

- Docker-Compose will take this YAML file and make it happen
- the project consists of three components:
- a web application,
- a Microsoft SQL Server instance, and
- a Couchbase Server instance

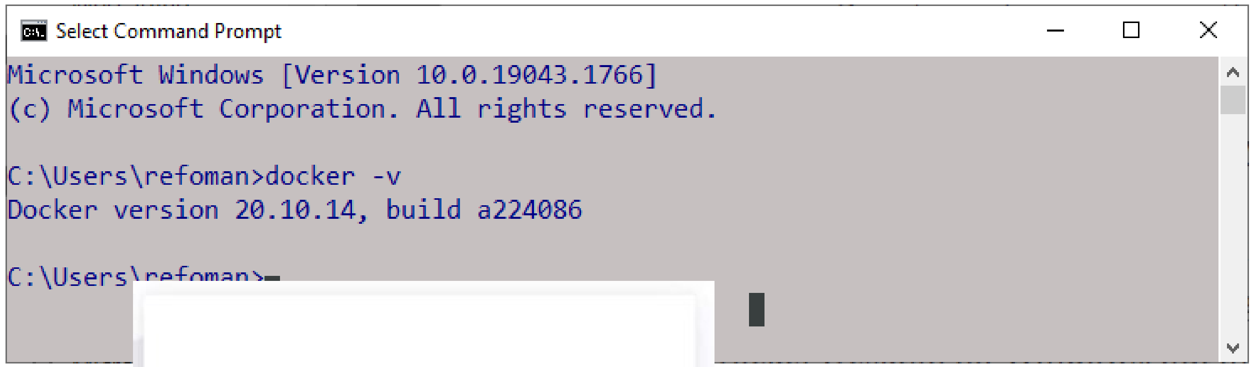
- As long as Docker is installed, a new developer on your team can pull down the latest from your source code repository and get started much faster than a developer who has to download, install, and manage each database, queueing system, web server, etc., themselves instead. Not to mention potential issues that could arise with environment factors, operating system versions, software versions, and more

Docker Terminology

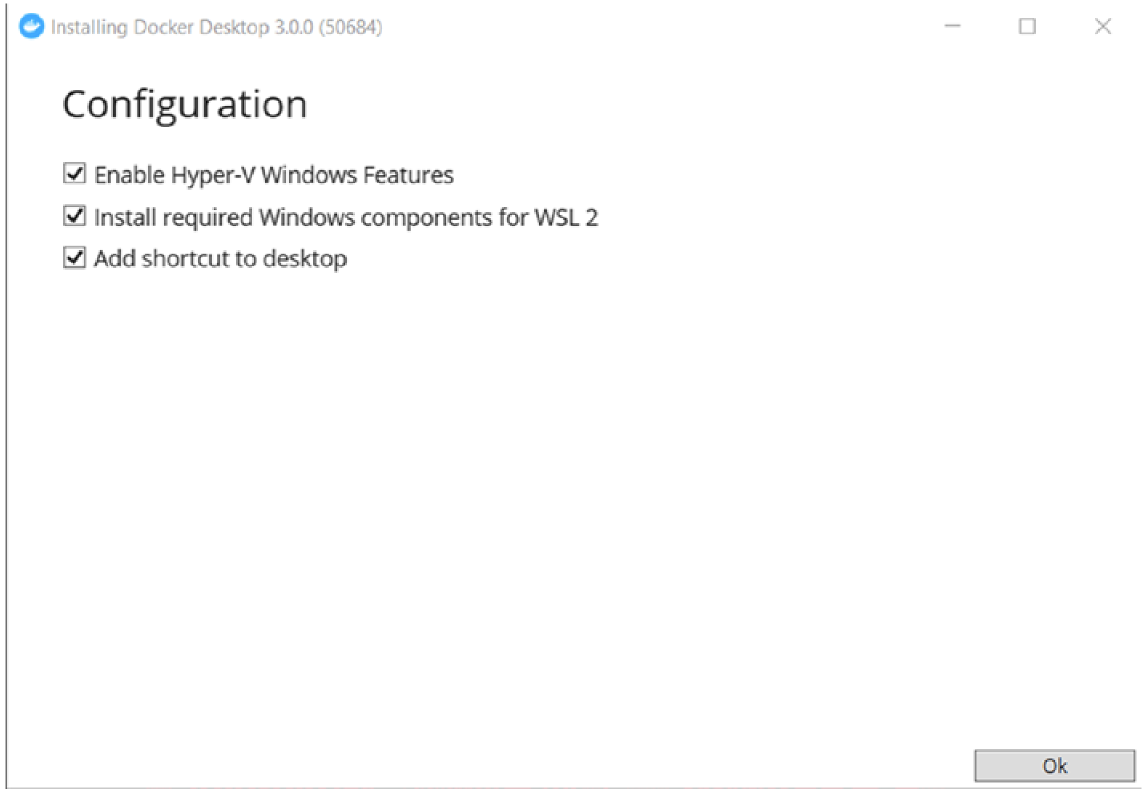
- Docker Enginer
- Docker container is created with a Docker image (blueprint)
  - An image is like a class and a container is like an object
- Image creation is defined by a Dockerfile
- Docker hub (like github)
- Azure Container Registry (ACR) can store images for private use
  - Use Docker to create an image
  - Push the image to Docker Hub (optional)
  - Once you have an image run a container based on that image

Install Docker

- Install from Docker.com
- Check with
  - docker -v



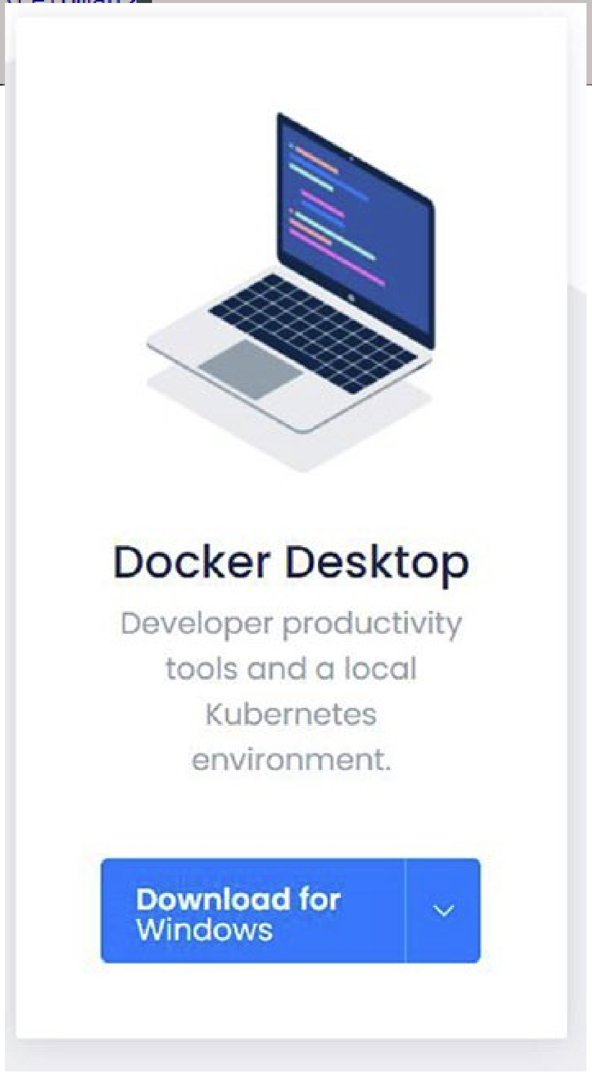
```
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\refoman>docker -v
Docker version 20.10.14, build a224086

C:\Users\refoman>
```



Installing Docker Desktop 3.0.0 (50684)

Configuration

☑ Enable Hyper-V Windows Features
☑ Install required Windows components for WSL 2
☑ Add shortcut to desktop



Docker Desktop

Developer productivity tools and a local Kubernetes environment.

Docker Basics

- Docker is a powerful, complex tool with many capabilities.
- focus on a few: creating, starting, and stopping containers, listing containers and images, removing containers and images, and inspecting a container.
- Later when we build an image, we'll let Visual Studio do some of the work for us.

Docker Run

docker run -d –-name myWebSite -p 8080:80 nginxdemos/hello

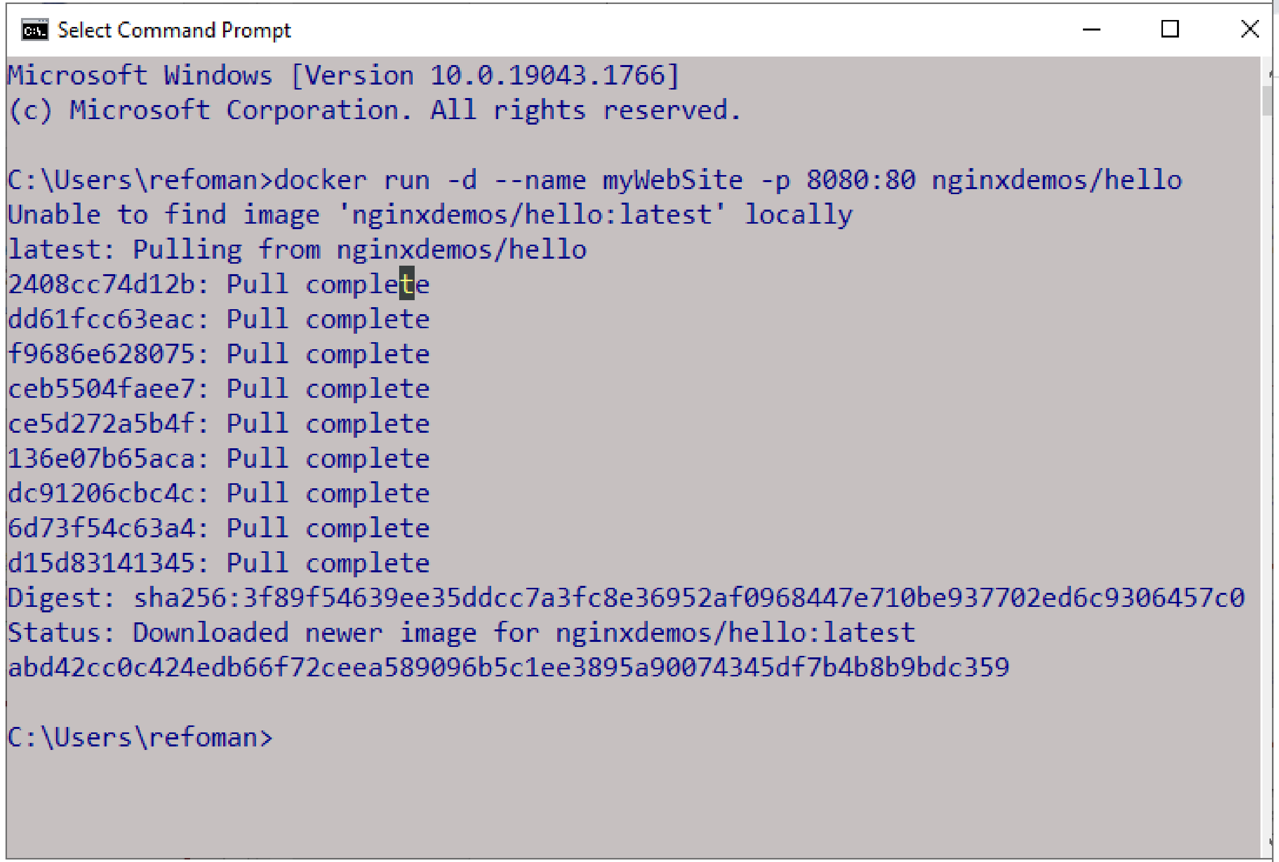
run : create a container and then start it

-d : run in detached mode. Simply means that the command prompt is released instead of turning over control of the input/output to Docker

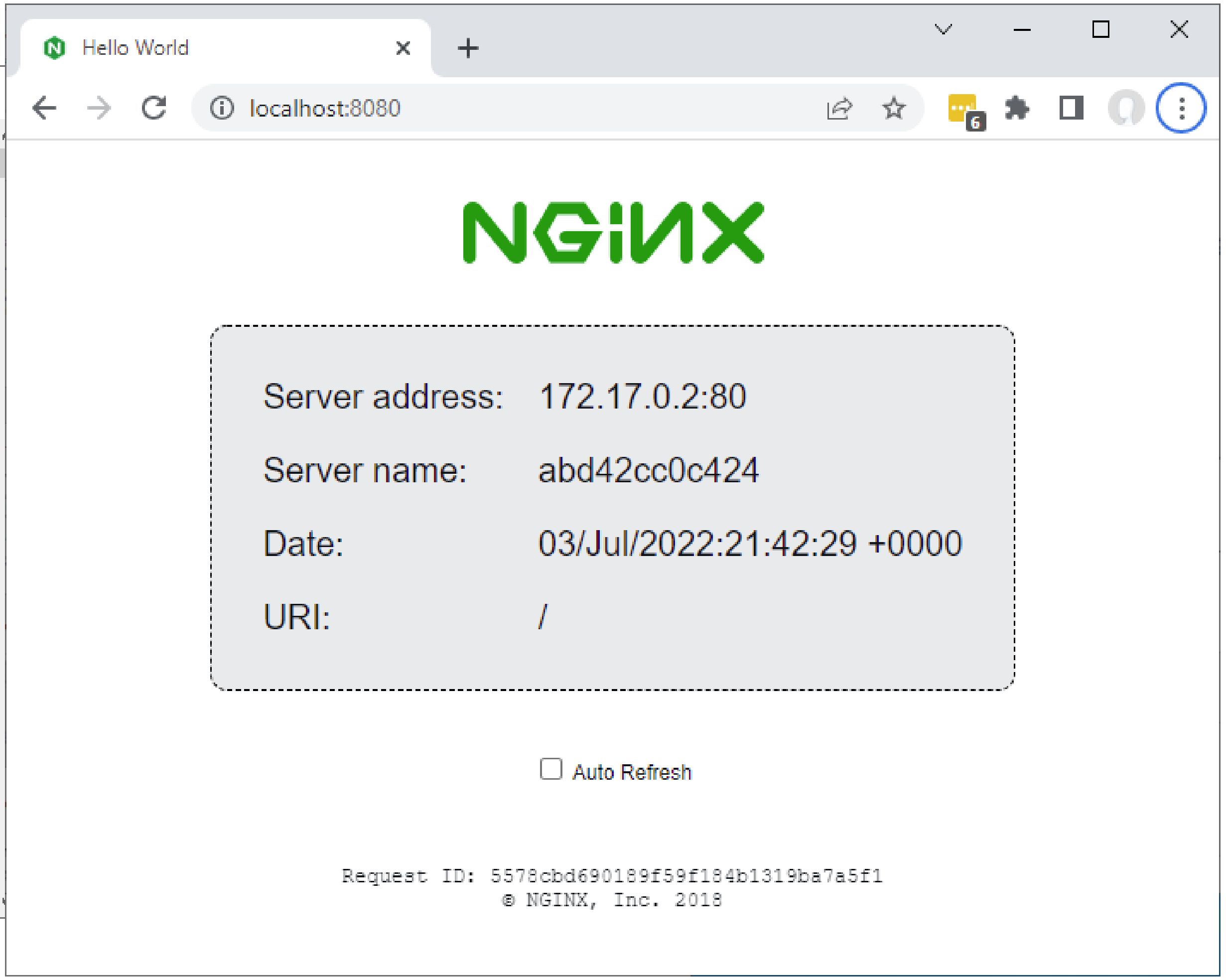
--name myWebSite   : give useful name to container

-p : specifies port mapping in 8080:80 docker will expose to 8080 externally, the 80 port of the app

ngixdemos/hello : an image to build the container from, checks local registry and then goes to Docker Hub



```
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\refoman>docker run -d --name myWebSite -p 8080:80 nginxdemos/hello
Unable to find image 'nginxdemos/hello:latest' locally
latest: Pulling from nginxdemos/hello
2408cc74d12b: Pull complete
dd61fcc63eac: Pull complete
f9686e628075: Pull complete
ceb5504faee7: Pull complete
ce5d272a5b4f: Pull complete
136e07b65aca: Pull complete
dc91206cbc4c: Pull complete
6d73f54c63a4: Pull complete
d15d83141345: Pull complete
Digest: sha256:3f89f54639ee35ddcc7a3fc8e36952af0968447e710be937702ed6c9306457c0
Status: Downloaded newer image for nginxdemos/hello:latest
abd42cc0c424edb66f72ceea589096b5c1ee3895a90074345df7b4b8b9bdc359

C:\Users\refoman>
```



Server address: 172.17.0.2:80

Server name: abd42cc0c424

Date: 03/Jul/2022:21:42:29 +0000

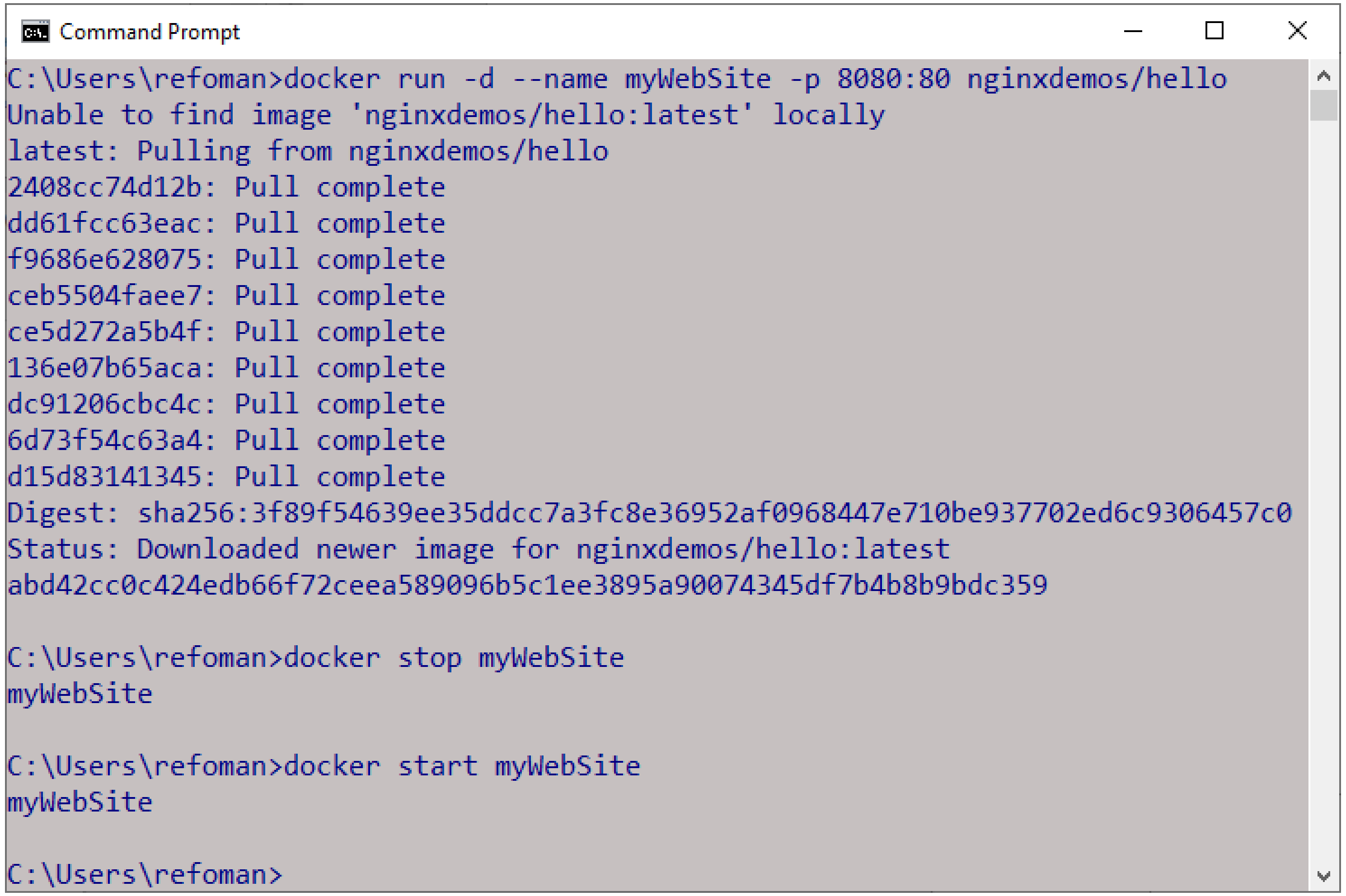
URI: /

Auto Refresh

```
Request ID: 5578cbd690189f59f184b1319ba7a5f1
© NGINX, Inc. 2018
```

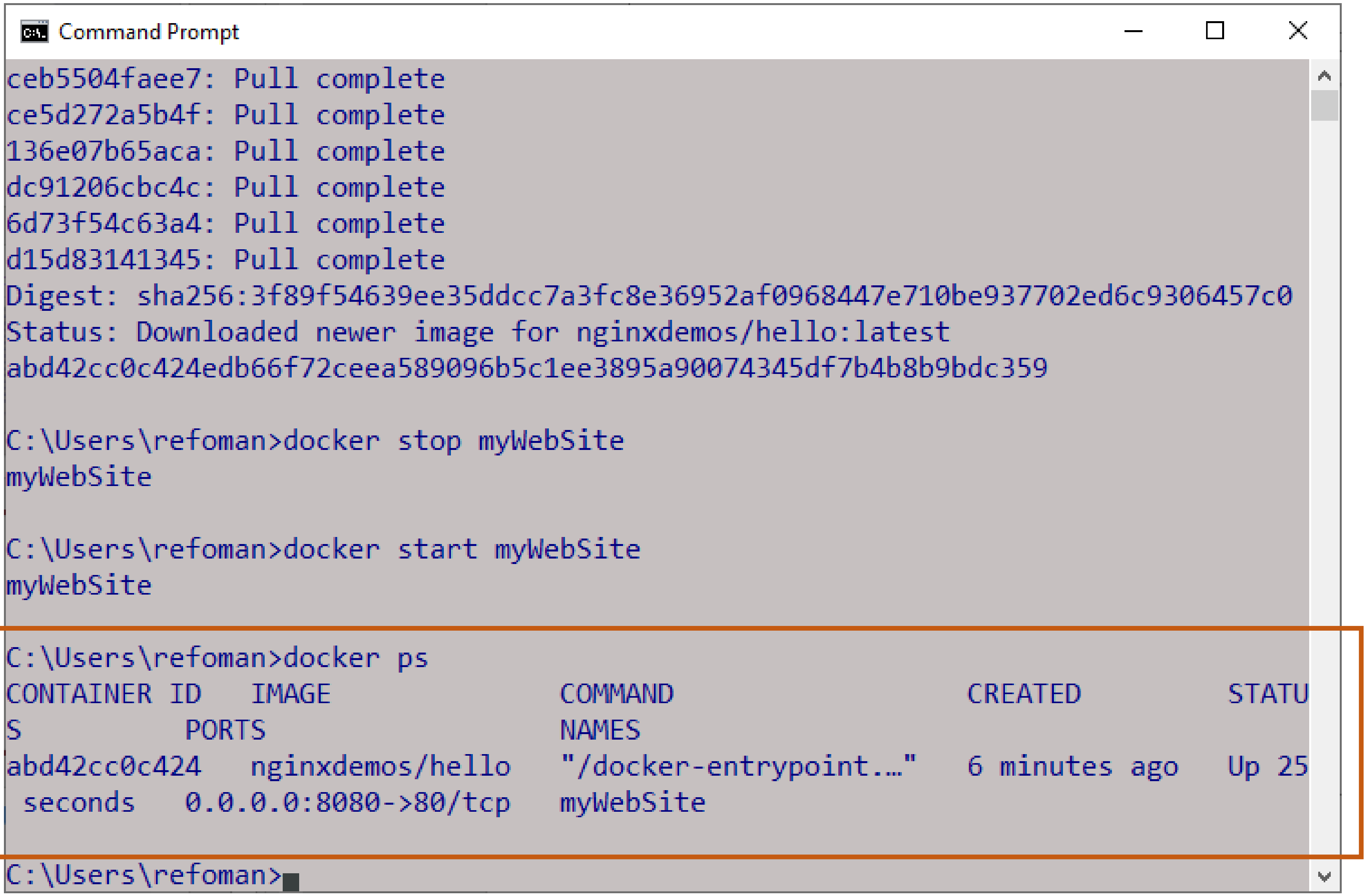
Docker stop / Docker start

- docker stop myWebSite

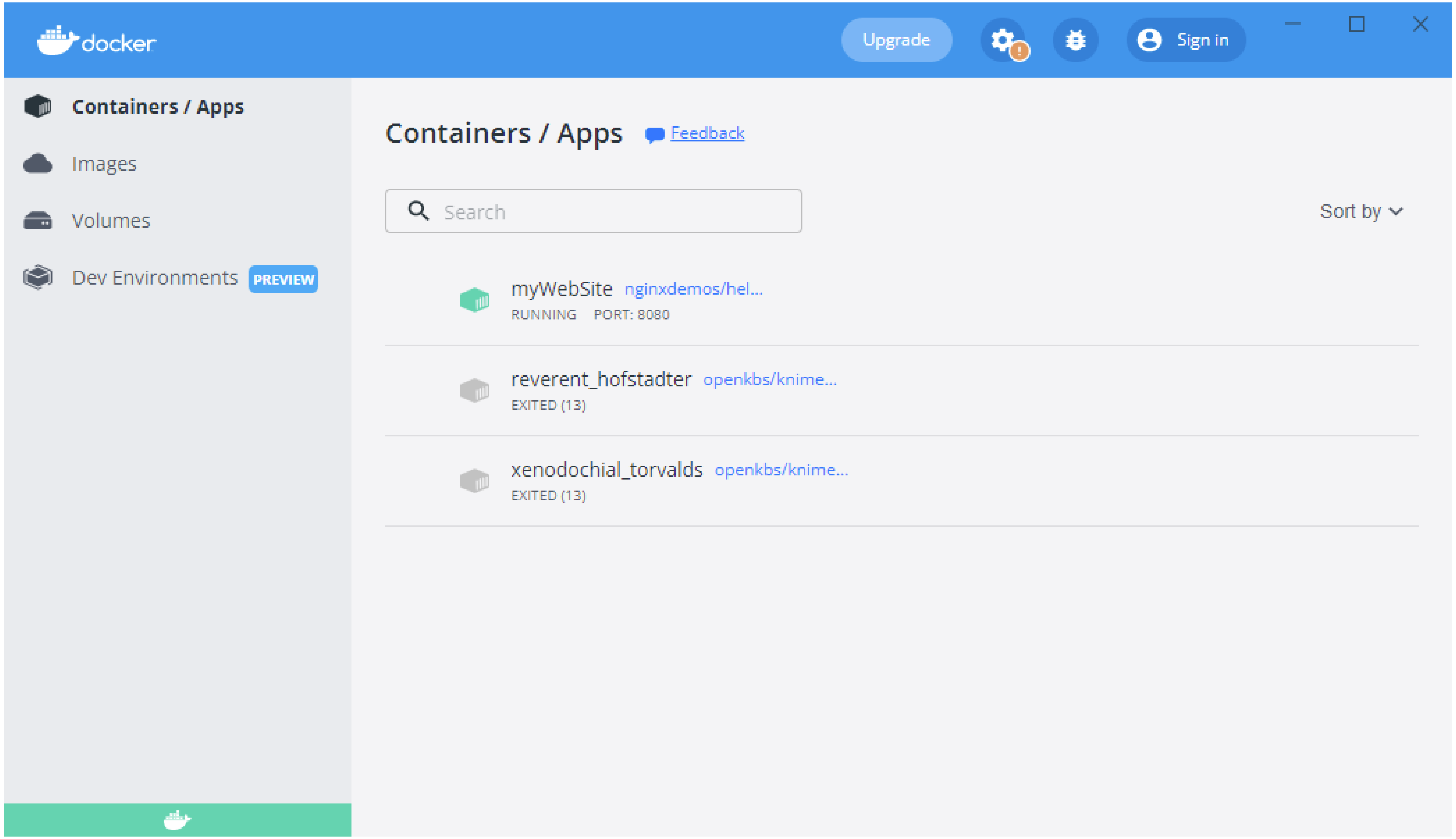


Command Prompt

```
C:\Users\refoman>docker run -d --name myWebSite -p 8080:80 nginxdemos/hello
Unable to find image 'nginxdemos/hello:latest' locally
latest: Pulling from nginxdemos/hello
2408cc74d12b: Pull complete
dd61fcc63eac: Pull complete
f9686e628075: Pull complete
ceb5504faee7: Pull complete
ce5d272a5b4f: Pull complete
136e07b65aca: Pull complete
dc91206cbc4c: Pull complete
6d73f54c63a4: Pull complete
d15d83141345: Pull complete
Digest: sha256:3f89f54639ee35ddcc7a3fc8e36952af0968447e710be937702ed6c9306457c0
Status: Downloaded newer image for nginxdemos/hello:latest
abd42cc0c424edb66f72ceea589096b5c1ee3895a90074345df7b4b8b9bdc359

C:\Users\refoman>docker stop myWebSite
myWebSite

C:\Users\refoman>docker start myWebSite
myWebSite

C:\Users\refoman>
```



```
ceb5504faee7: Pull complete
ce5d272a5b4f: Pull complete
136e07b65aca: Pull complete
dc91206cbc4c: Pull complete
6d73f54c63a4: Pull complete
d15d83141345: Pull complete
Digest: sha256:3f89f54639ee35ddcc7a3fc8e36952af0968447e710be937702ed6c9306457c0
Status: Downloaded newer image for nginxdemos/hello:latest
abd42cc0c424edb66f72ceea589096b5c1ee3895a90074345df7b4b8b9bdc359

C:\Users\refoman>docker stop myWebSite
myWebSite

C:\Users\refoman>docker start myWebSite
myWebSite

C:\Users\refoman>docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATU
S            PORTS                  NAMES
abd42cc0c424   nginxdemos/hello     "/docker-entrypoint.…"   6 minutes ago    Up 25
 seconds    0.0.0.0:8080->80/tcp   myWebSite

C:\Users\refoman>
```



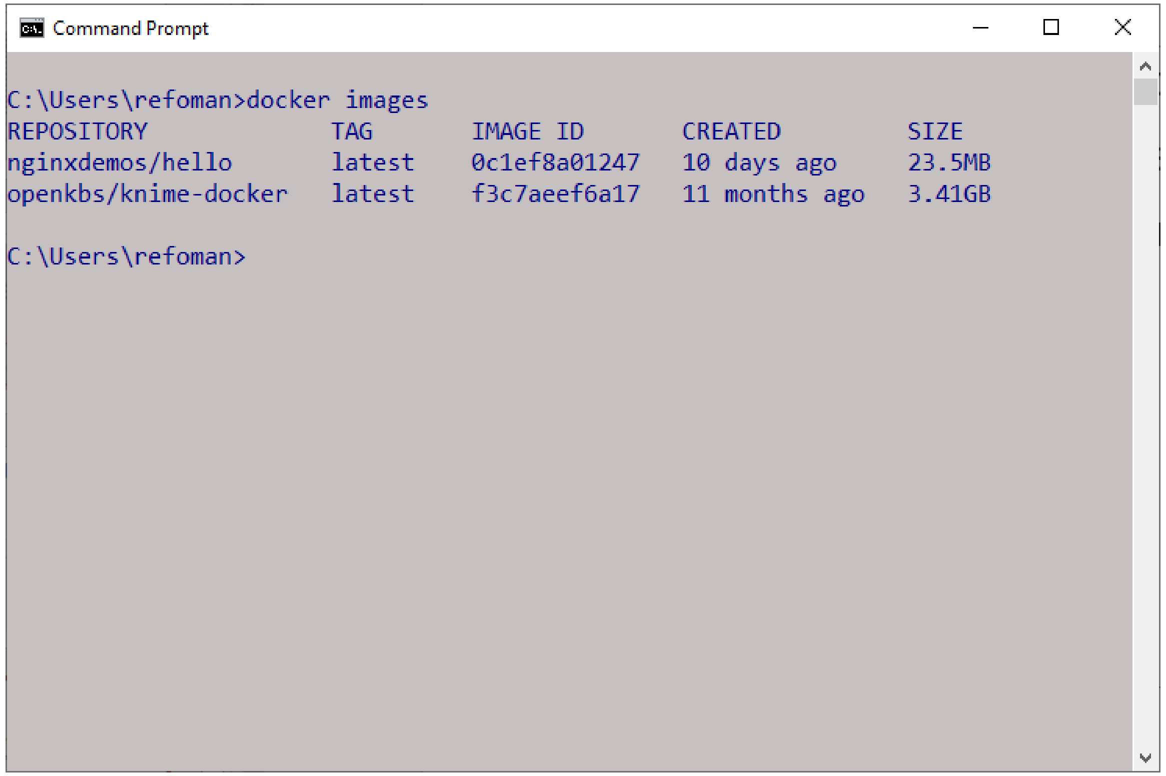
myWebSite nginxdemos/hel...
RUNNING PORT: 8080

reverent_hofstadter openkbs/knime...
EXITED (13)

xenodochial_torvalds openkbs/knime...
EXITED (13)

Docker Images

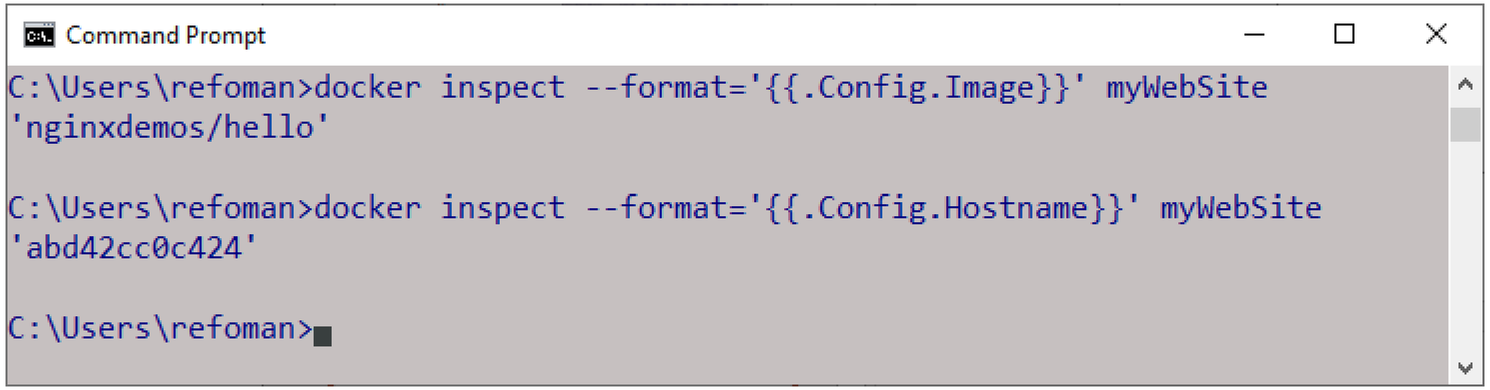
- Images are the blueprints to create containers
- Even when a container is removed, the image remains cached.
- You can get a list of these images:
- docker images



```
C:\Users\refoman>docker images
REPOSITORY              TAG       IMAGE ID       CREATED         SIZE
nginxdemos/hello        latest    0c1ef8a01247   10 days ago     23.5MB
openkbs/knime-docker    latest    f3c7aeef6a17   11 months ago   3.41GB

C:\Users\refoman>
```

Docker inspect

- docker inspect myWebSite
- Full information in JSON format
- docker inspect --format='{{.Config.Image}}' myWebSite

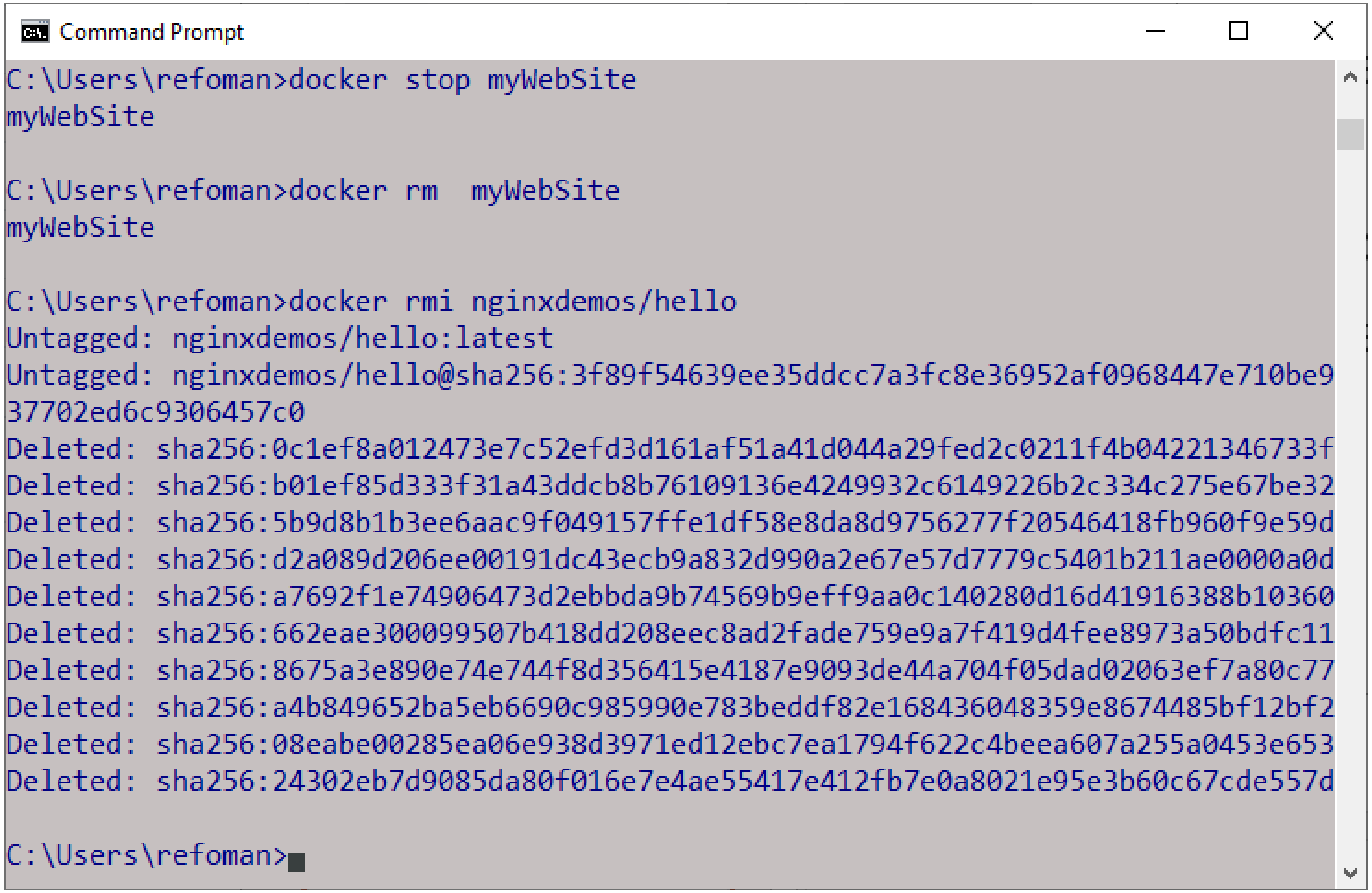


```
C:\Users\refoman>docker inspect --format='{{.Config.Image}}' myWebSite
'nginxdemos/hello'

C:\Users\refoman>docker inspect --format='{{.Config.Hostname}}' myWebSite
'abd42cc0c424'

C:\Users\refoman>
```

Docker rm & rmi

- Remove container (first needs to be stopped)
- docker rm myWebSite

- Remove image (there must be no container referencing it)
- docker rmi nginxdemos/hello



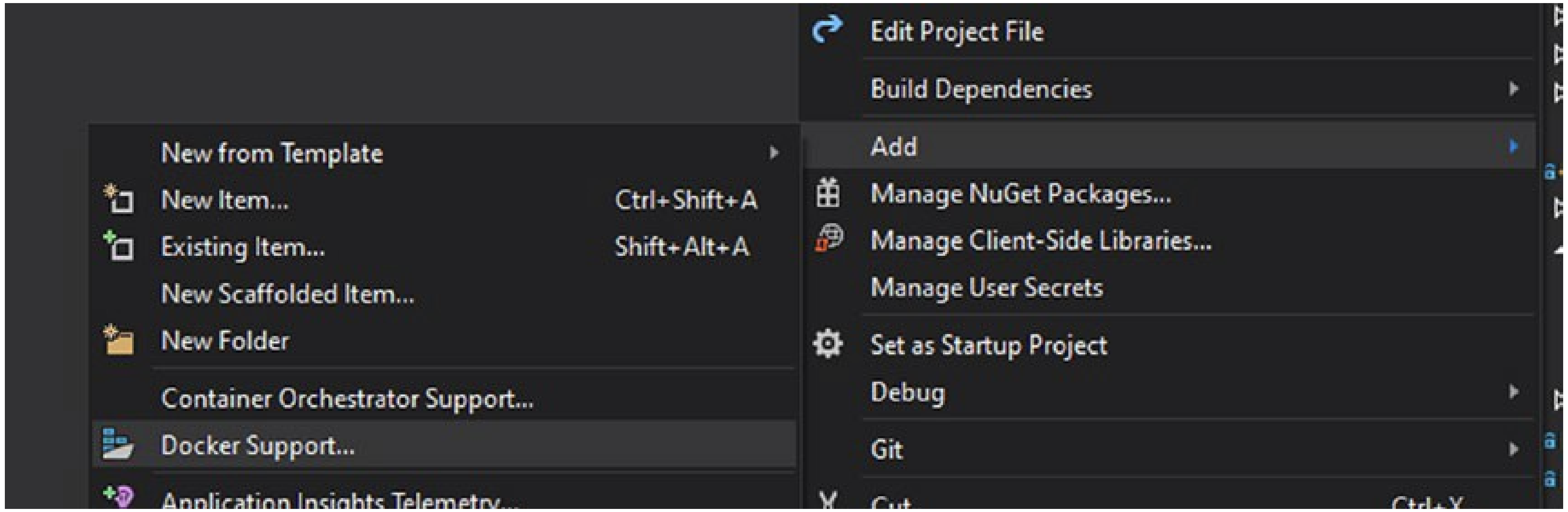
```
C:\Users\refoman>docker stop myWebSite
myWebSite

C:\Users\refoman>docker rm  myWebSite
myWebSite

C:\Users\refoman>docker rmi nginxdemos/hello
Untagged: nginxdemos/hello:latest
Untagged: nginxdemos/hello@sha256:3f89f54639ee35ddcc7a3fc8e36952af0968447e710be9
37702ed6c9306457c0
Deleted: sha256:0c1ef8a012473e7c52efd3d161af51a41d044a29fed2c0211f4b04221346733f
Deleted: sha256:b01ef85d333f31a43ddcb8b76109136e4249932c6149226b2c334c275e67be32
Deleted: sha256:5b9d8b1b3ee6aac9f049157ffe1df58e8da8d9756277f20546418fb960f9e59d
Deleted: sha256:d2a089d206ee00191dc43ecb9a832d990a2e67e57d7779c5401b211ae0000a0d
Deleted: sha256:a7692f1e74906473d2ebbda9b74569b9eff9aa0c140280d16d41916388b10360
Deleted: sha256:662eae300099507b418dd208eec8ad2fade759e9a7f419d4fee8973a50bdfc11
Deleted: sha256:8675a3e890e74e744f8d356415e4187e9093de44a704f05dad02063ef7a80c77
Deleted: sha256:a4b849652ba5eb6690c985990e783beddf82e168436048359e8674485bf12bf2
Deleted: sha256:08eabe00285ea06e938d3971ed12ebc7ea1794f622c4beea607a255a0453e653
Deleted: sha256:24302eb7d9085da80f016e7e4ae55417e412fb7e0a8021e95e3b60c67cde557d

C:\Users\refoman>
```

Microservices and Docker

- ASP.NET web application
- + SQL Server for persistence
- Containerize the web app
- Add docker support
- For ease of use we ll include the SQL Server within the Docker Host
- Super useful for development purposes
- Deployment may not be via a container for the DB

Create ASP.NET image

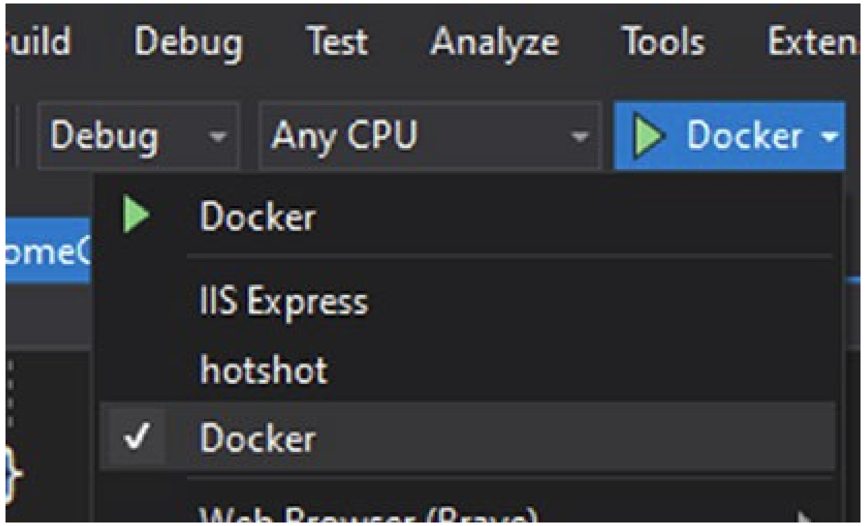
- Right click project and click add – docker support



ASP.NET dockerfile

With this Dockerfile, Visual Studio now knows how to create an image, create a container, deploy your latest changes to the container, and run your service in Docker (even when debugging).

Docker as one of the execution options



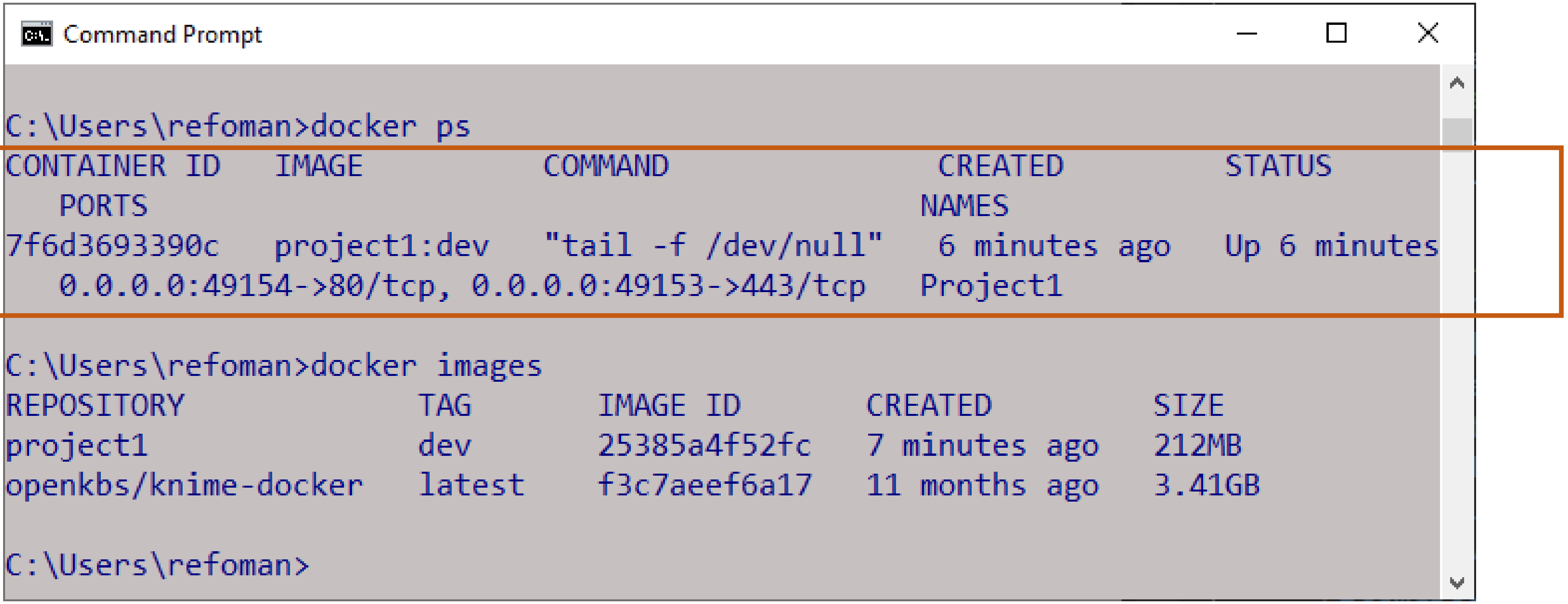
```
#See https://aka.ms/containerfastmode to understand how Visual
Studio uses this Dockerfile to build your images for faster
debugging.

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
# Install NodeJs
RUN apt-get update && \
apt-get install -y wget && \
apt-get install -y gnupg2 && \
wget -qO- https://deb.nodesource.com/setup_12.x | bash - && \
apt-get install -y build-essential nodejs
# End Install
WORKDIR /src
COPY ["Project1/Project1.csproj", "Project1/"]
RUN dotnet restore "Project1/Project1.csproj"
COPY . .
WORKDIR "/src/Project1"
RUN dotnet build "Project1.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "Project1.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "Project1.dll"]
```

https://github.com/Apress/pro-microservices-in-.net-6/blob/main/Chapter8/Dockerfile

Run Docker

- Visual Studio will instruct Docker to create a new image (if necessary)
- create a new container (if necessary), and
- deploy the latest compilation of your service to that container.

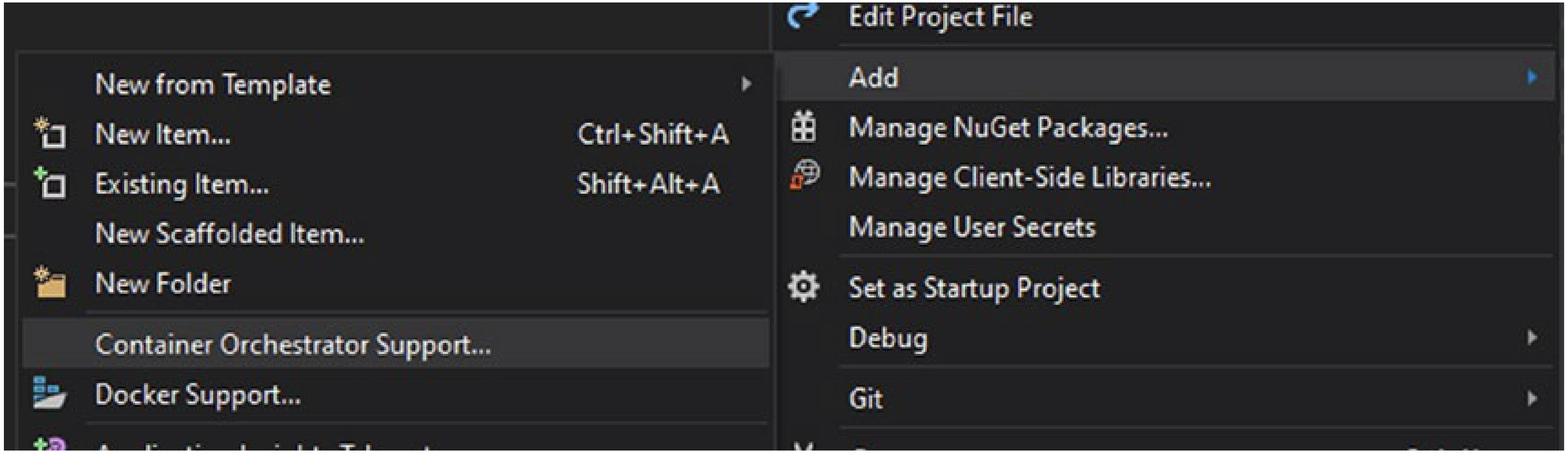
Microservices and Docker


Command Prompt

```
C:\Users\refoman>docker ps
CONTAINER ID   IMAGE           COMMAND                CREATED          STATUS
   PORTS                                          NAMES
7f6d3693390c   project1:dev    "tail -f /dev/null"    6 minutes ago    Up 6 minutes
   0.0.0.0:49154->80/tcp, 0.0.0.0:49153->443/tcp   Project1

C:\Users\refoman>docker images
REPOSITORY             TAG       IMAGE ID       CREATED          SIZE
project1               dev       25385a4f52fc   7 minutes ago    212MB
openkbs/knime-docker   latest    f3c7aeef6a17   11 months ago    3.41GB

C:\Users\refoman>
```

Docker Compose

- image for ASP.NET is created
- we still need a database for it to work with
- add “orchestration” support (the ability to manage more than one container in concert)
- Again, right-click the project in Visual Studio,
- click Add, and
- this time click “Container Orchestrator Support.”
- Select “Docker Compose” for now


Edit Project File
Add
Manage NuGet Packages...
Manage Client-Side Libraries...
Manage User Secrets
Set as Startup Project
Debug
Git
New from Template
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Scaffolded Item...
New Folder
Container Orchestrator Support...
Docker Support...

New Files in the Solution

- docker-compose.yml

Only one service is defined here: the ASP.NET web service.

Let’s add another service to this YML (YAML) file to add SQL Server as a database. (If SQL Server is already installed and running outside of Docker, turn it off to avoid any port conflicts from this point forward.)

```yaml
version: '3.4'

services:
  project1:
    image: ${DOCKER_REGISTRY-}project1
    build:
      context: .
      dockerfile: Project1/Dockerfile
  mssql:
    image: 
mcr.microsoft.com/mssql/server:2019-latest
    environment:
    - ACCEPT_EULA=Y
    - SA_PASSWORD=myStrongPassword1!
    ports:
    - "31433:1433"
```

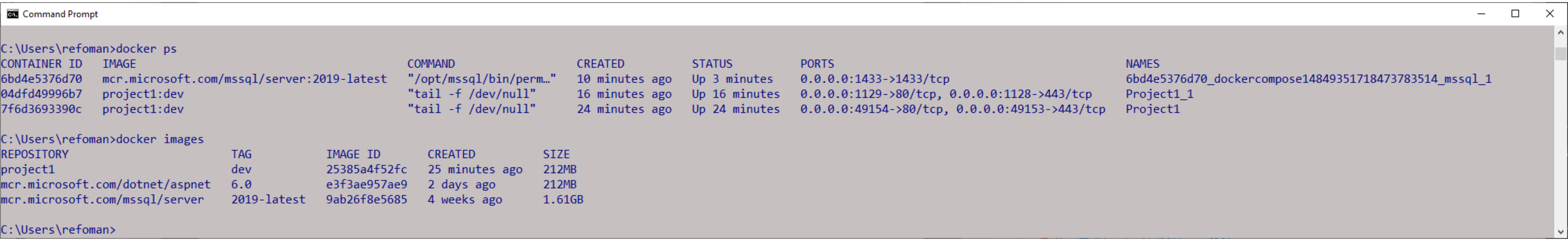
YAML mssql service section

- **Image**: the name of the image that the docker will try to retrieve to build the container

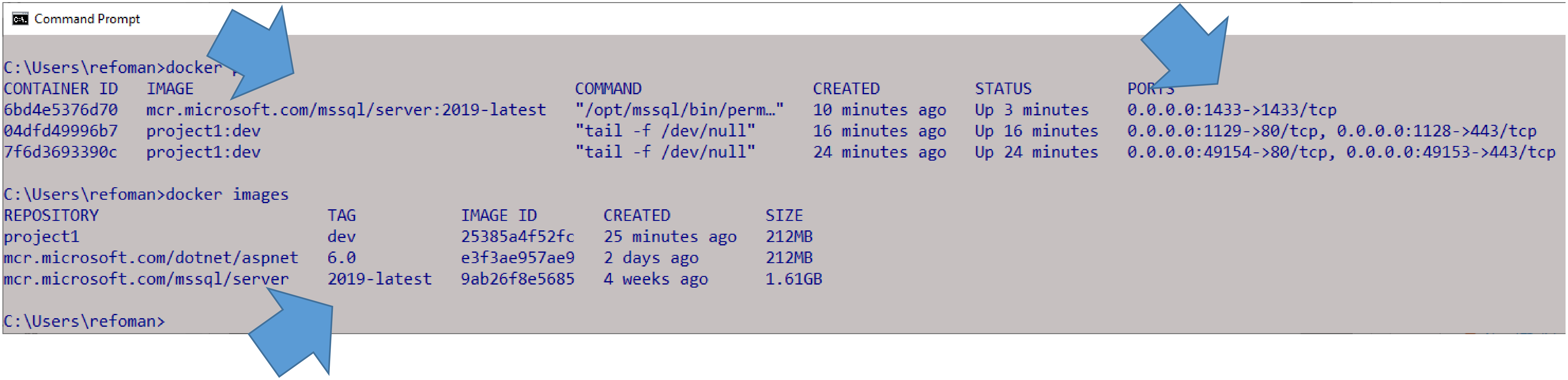
Microsoft publishes SQL Server images to docker hub

https://hub.docker.com/_/microsoft-mssql-server

- **Environment**: Environment is an array of values that will be set as environment variables within the container. (Accept EULA, and create a login for SA
- **Ports** : port mappings (like in docker run)
- Changes in docker-compose are executed immediately and automatically upon save



```
C:\Users\refoman>docker ps
CONTAINER ID   IMAGE                                        COMMAND                  CREATED          STATUS           PORTS                                            NAMES
6bd4e5376d70   mcr.microsoft.com/mssql/server:2019-latest   "/opt/mssql/bin/perm…"   10 minutes ago   Up 3 minutes     0.0.0.0:1433->1433/tcp                           6bd4e5376d70_dockercompose14849351718473783514_mssql_1
04dfd49996b7   project1:dev                                 "tail -f /dev/null"      16 minutes ago   Up 16 minutes    0.0.0.0:1129->80/tcp, 0.0.0.0:1128->443/tcp      Project1_1
7f6d3693390c   project1:dev                                 "tail -f /dev/null"      24 minutes ago   Up 24 minutes    0.0.0.0:49154->80/tcp, 0.0.0.0:49153->443/tcp    Project1

C:\Users\refoman>docker images
REPOSITORY                         TAG           IMAGE ID       CREATED          SIZE
project1                           dev           25385a4f52fc   25 minutes ago   212MB
mcr.microsoft.com/dotnet/aspnet    6.0           e3f3ae957ae9   2 days ago       212MB
mcr.microsoft.com/mssql/server     2019-latest   9ab26f8e5685   4 weeks ago      1.61GB

C:\Users\refoman>
```



```
C:\Users\refoman>docker
CONTAINER ID   IMAGE                                        COMMAND                  CREATED          STATUS           PORTS
6bd4e5376d70   mcr.microsoft.com/mssql/server:2019-latest   "/opt/mssql/bin/perm…"   10 minutes ago   Up 3 minutes     0.0.0.0:1433->1433/tcp
04dfd49996b7   project1:dev                                 "tail -f /dev/null"      16 minutes ago   Up 16 minutes    0.0.0.0:1129->80/tcp, 0.0.0.0:1128->443/tcp
7f6d3693390c   project1:dev                                 "tail -f /dev/null"      24 minutes ago   Up 24 minutes    0.0.0.0:49154->80/tcp, 0.0.0.0:49153->443/tcp

C:\Users\refoman>docker images
REPOSITORY                         TAG           IMAGE ID       CREATED          SIZE
project1                           dev           25385a4f52fc   25 minutes ago   212MB
mcr.microsoft.com/dotnet/aspnet    6.0           e3f3ae957ae9   2 days ago       212MB
mcr.microsoft.com/mssql/server     2019-latest   9ab26f8e5685   4 weeks ago      1.61GB

C:\Users\refoman>
```

Docker has its own DNS service

- The “mssql” name that you specified is now the hostname for SQL Server.
- So, a connection string in your ASP.NET service should look like

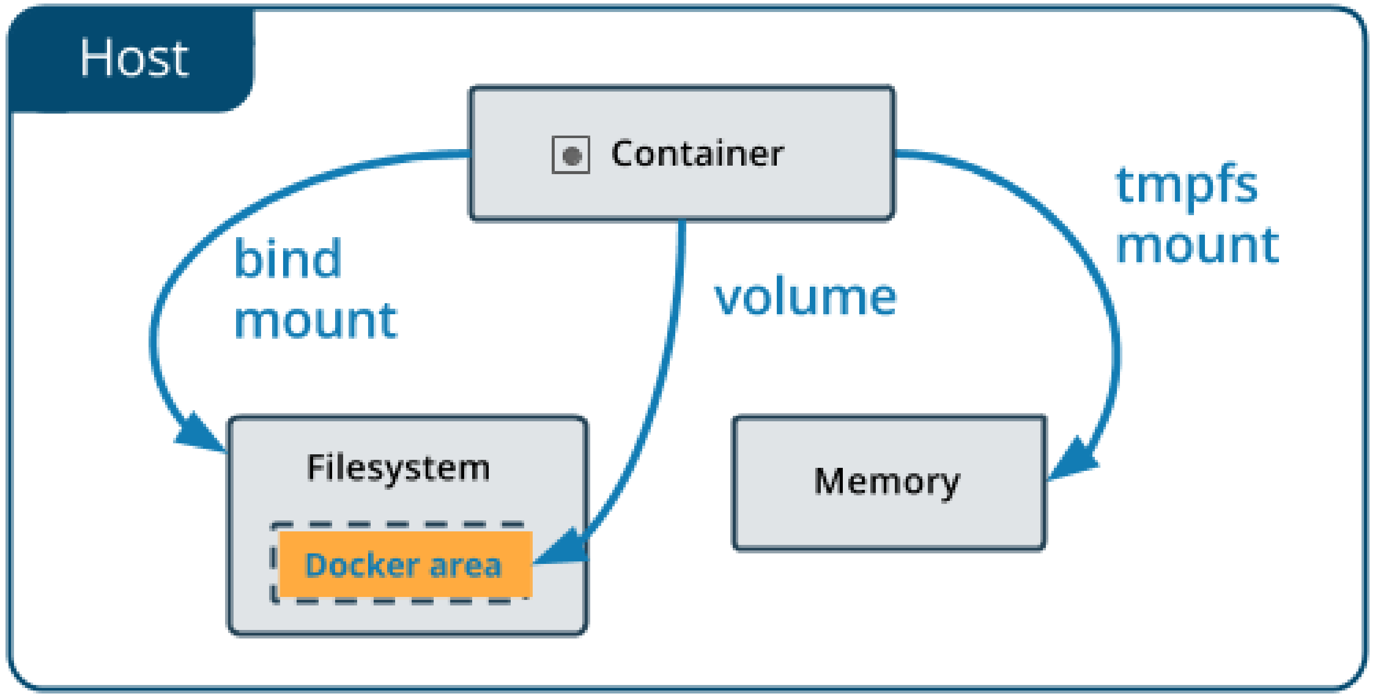
```
Server=mssql;Database=myDatabase;User=sa;Password=myStrongPassword1!
```

Closing VS.NET

- when you close your Solution in Visual Studio, these images will also be shut down and removed.
- **Any state saved to the database will be swept away**
- (unless you use Docker volumes).
- One more important note: the SQL Server image used earlier **does not come with any predefined databases, tables, or data in it**. You can create these with other tooling, thanks to port 1433 being open.
- Alternatively, you can create your own custom SQL Server Docker image that will create a database for you.

Docker Volumes

- https://docs.docker.com/storage/volumes/

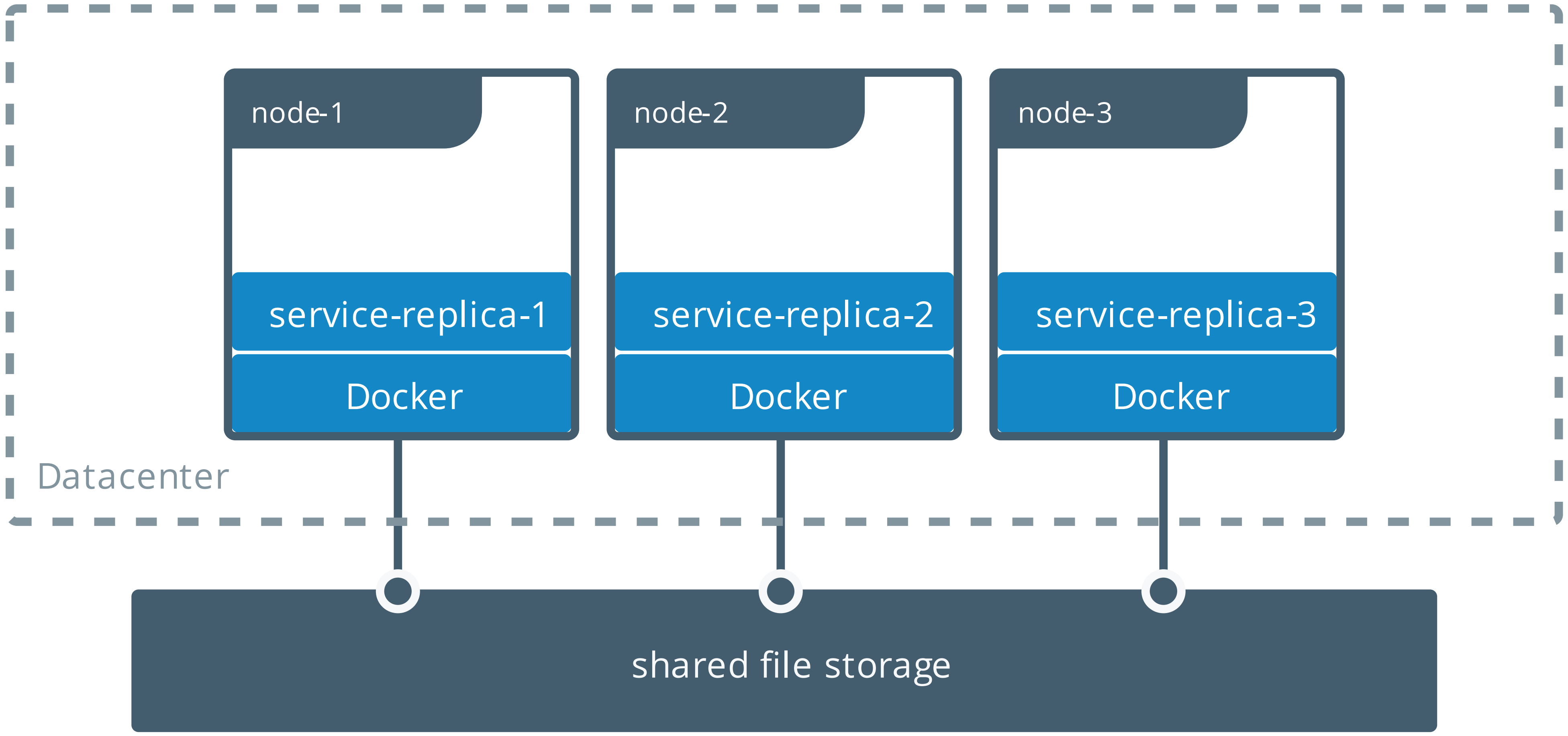


Docker Volumes

- docker volume create my-vol
- docker volume inspect my-vol
- docker volume rm my-vol

```
docker run -d \
  --name devtest \
  --mount source=myvol2,target=/app \
  nginx:latest
docker inspect devtest
```

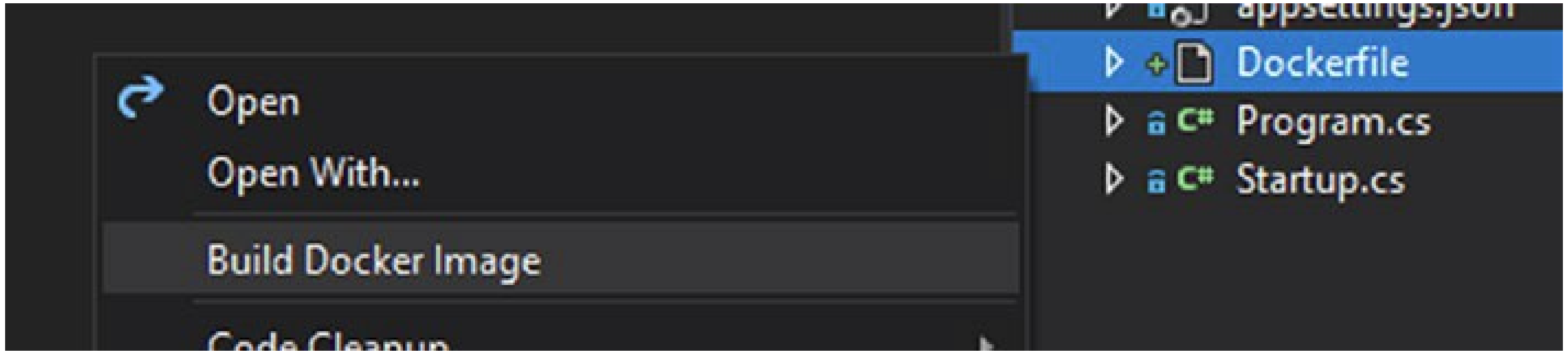
Share data among machines



Build an Image (to push to Docker Hub)

- Docker Compose has been building images for development (notice “hotshot:dev”)
- build an image that’s suitable for deployment.
- From Visual Studio, you can do this by right-clicking

“Dockerfile” and selecting “Build Docker Image.”



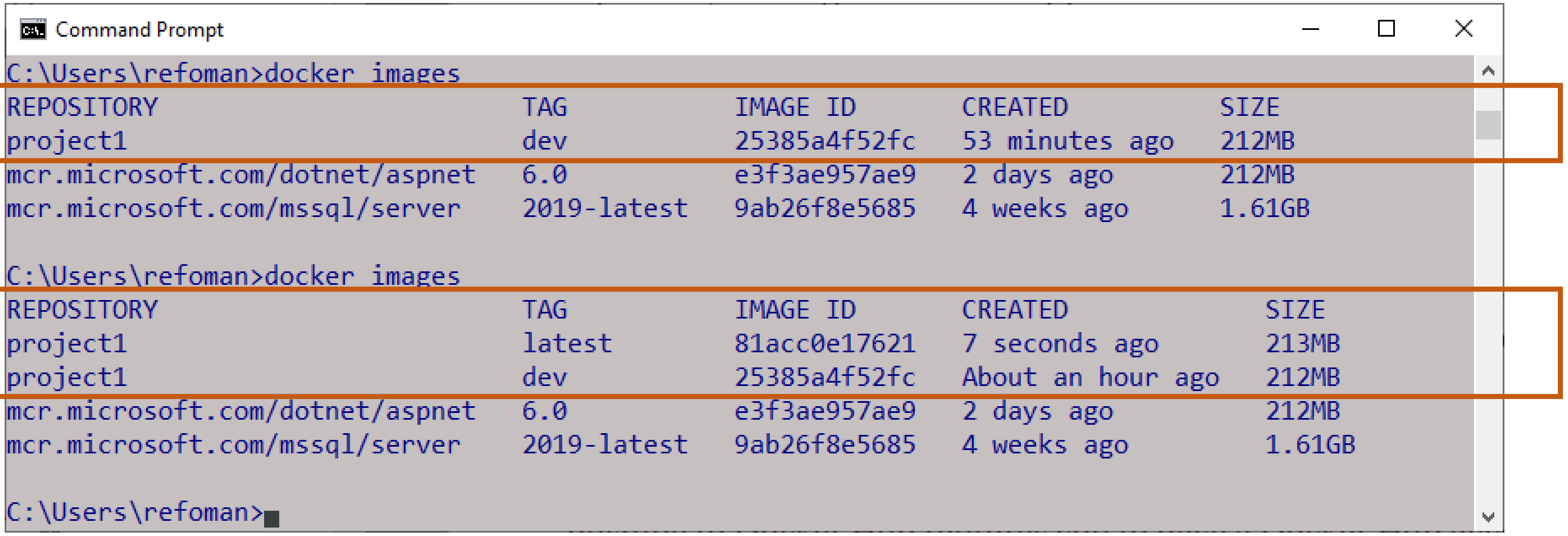
docker build -f "c:\path\to\hotshot\hotshot\Dockerfile" -t hotshot "c:\path\to\hotshot"

Fix

- Add nodejs in the docker

```
#See https://aka.ms/containerfastmode to understand how Visual
Studio uses this Dockerfile to build your images for faster
debugging.

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
# Install NodeJs
RUN apt-get update && \
apt-get install -y wget && \
apt-get install -y gnupg2 && \
wget -qO- https://deb.nodesource.com/setup_12.x | bash - && \
apt-get install -y build-essential nodejs
# End Install
WORKDIR /src
COPY ["Project1/Project1.csproj", "Project1/"]
RUN dotnet restore "Project1/Project1.csproj"
COPY . .
WORKDIR "/src/Project1"
RUN dotnet build "Project1.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "Project1.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "Project1.dll"]
```

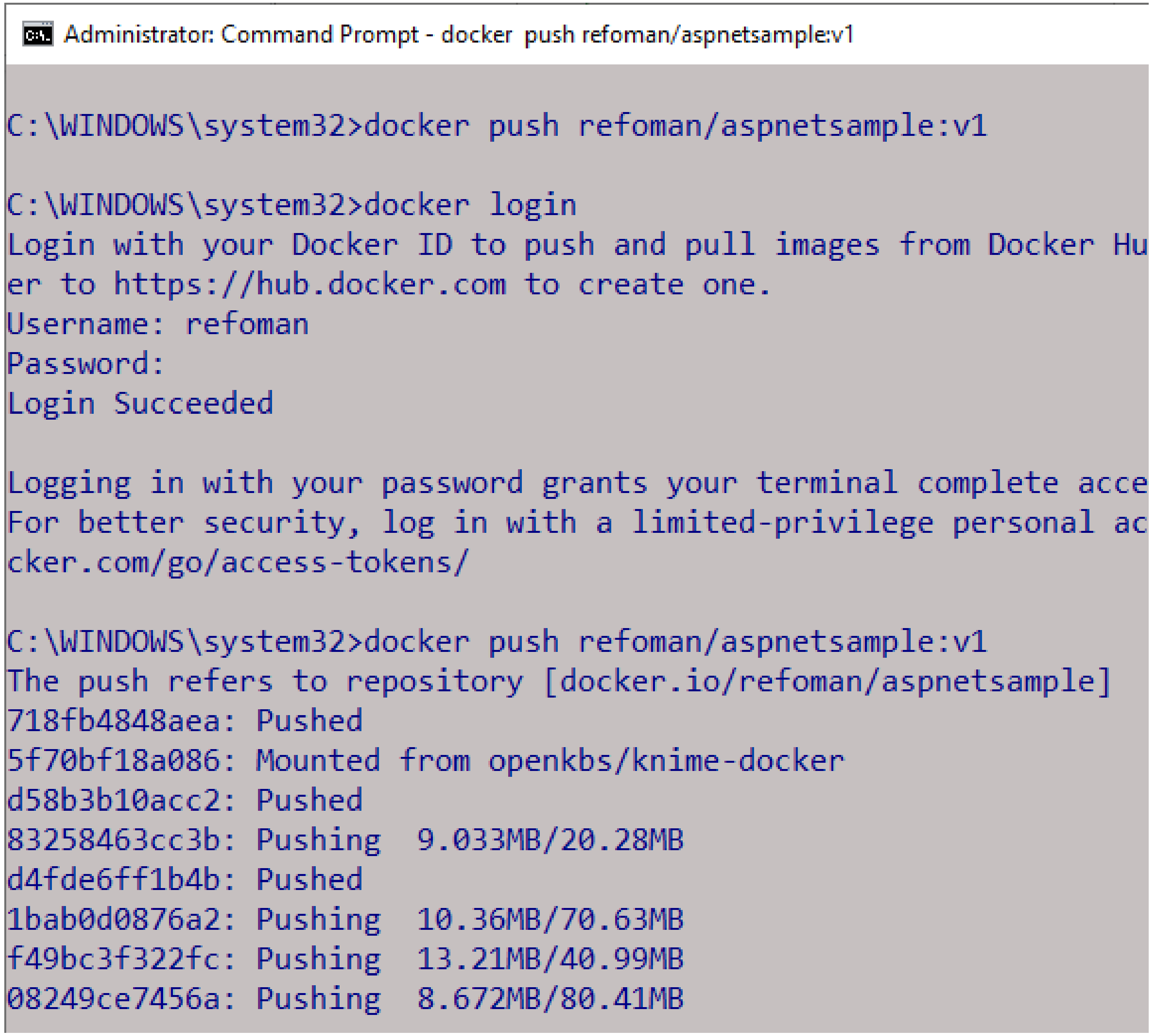
Build docker file



```
C:\Users\refoman>docker images
REPOSITORY                        TAG           IMAGE ID       CREATED            SIZE
project1                          dev           25385a4f52fc   53 minutes ago     212MB
mcr.microsoft.com/dotnet/aspnet   6.0           e3f3ae957ae9   2 days ago         212MB
mcr.microsoft.com/mssql/server    2019-latest   9ab26f8e5685   4 weeks ago        1.61GB

C:\Users\refoman>docker images
REPOSITORY                        TAG           IMAGE ID       CREATED               SIZE
project1                          latest        81acc0e17621   7 seconds ago         213MB
project1                          dev           25385a4f52fc   About an hour ago     212MB
mcr.microsoft.com/dotnet/aspnet   6.0           e3f3ae957ae9   2 days ago            212MB
mcr.microsoft.com/mssql/server    2019-latest   9ab26f8e5685   4 weeks ago           1.61GB

C:\Users\refoman>
```

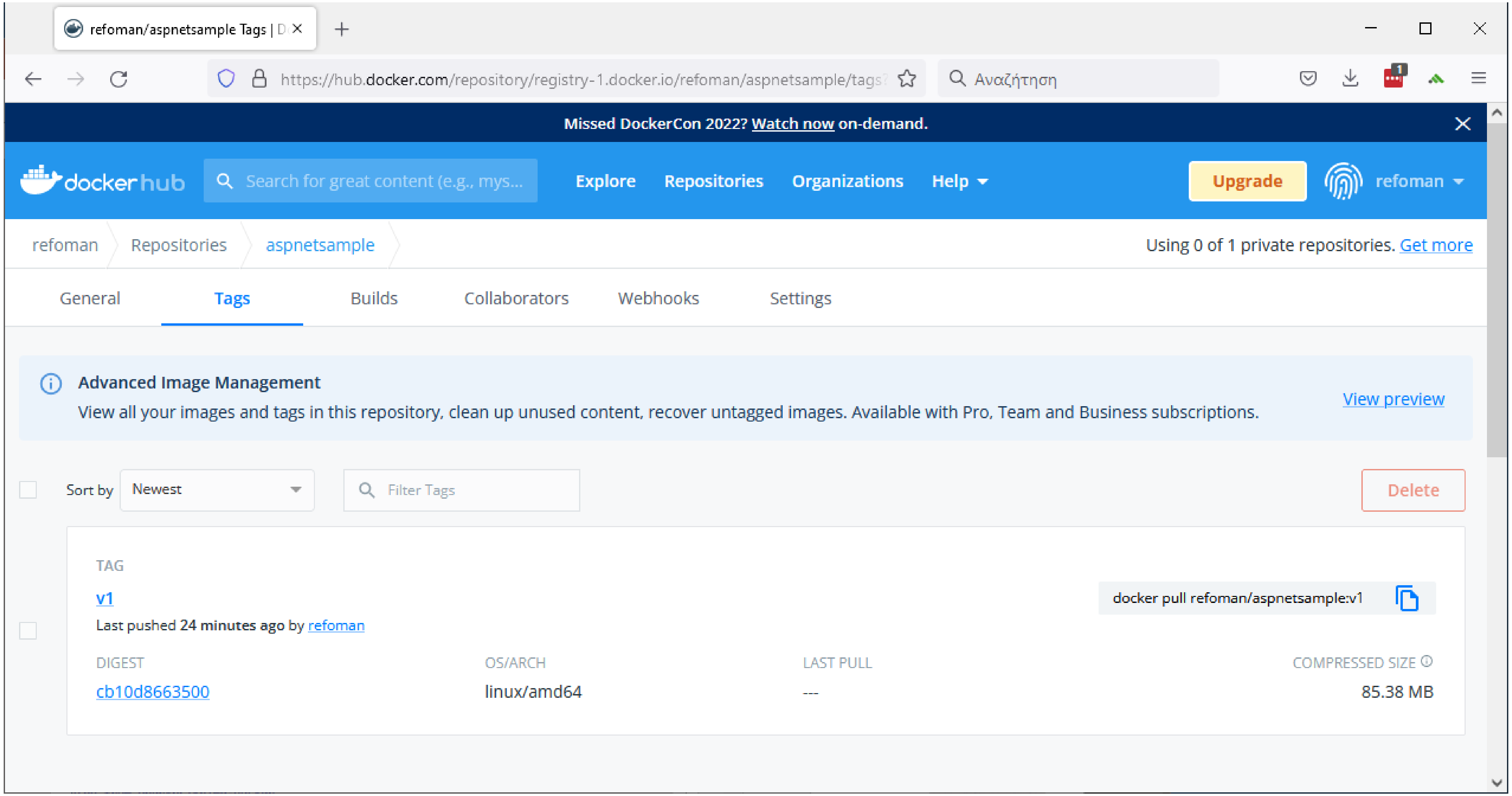
Docker Push

- docker image tag hotshot:latest microservicemogul/hotshot:v1
- Now your image is ready to push. Execute a push:
- docker push microservicemogul/hotshot:v1



Administrator: Command Prompt - docker push refoman/aspnetsample:v1

```
C:\WINDOWS\system32>docker push refoman/aspnetsample:v1

C:\WINDOWS\system32>docker login
Login with your Docker ID to push and pull images from Docker Hu
er to https://hub.docker.com to create one.
Username: refoman
Password:
Login Succeeded

Logging in with your password grants your terminal complete acce
For better security, log in with a limited-privilege personal ac
cker.com/go/access-tokens/

C:\WINDOWS\system32>docker push refoman/aspnetsample:v1
The push refers to repository [docker.io/refoman/aspnetsample]
718fb4848aea: Pushed
5f70bf18a086: Mounted from openkbs/knime-docker
d58b3b10acc2: Pushed
83258463cc3b: Pushing  9.033MB/20.28MB
d4fde6ff1b4b: Pushed
1bab0d0876a2: Pushing  10.36MB/70.63MB
f49bc3f322fc: Pushing  13.21MB/40.99MB
08249ce7456a: Pushing  8.672MB/80.41MB
```


refoman/aspnetsample Tags | D

https://hub.docker.com/repository/registry-1.docker.io/refoman/aspnetsample/tags

Missed DockerCon 2022? Watch now on-demand.

refoman Repositories aspnetsample

Using 0 of 1 private repositories. Get more

General **Tags** Builds Collaborators Webhooks Settings

Advanced Image Management
View all your images and tags in this repository, clean up unused content, recover untagged images. Available with Pro, Team and Business subscriptions.

View preview

Sort by: Newest

TAG
v1
Last pushed 24 minutes ago by refoman

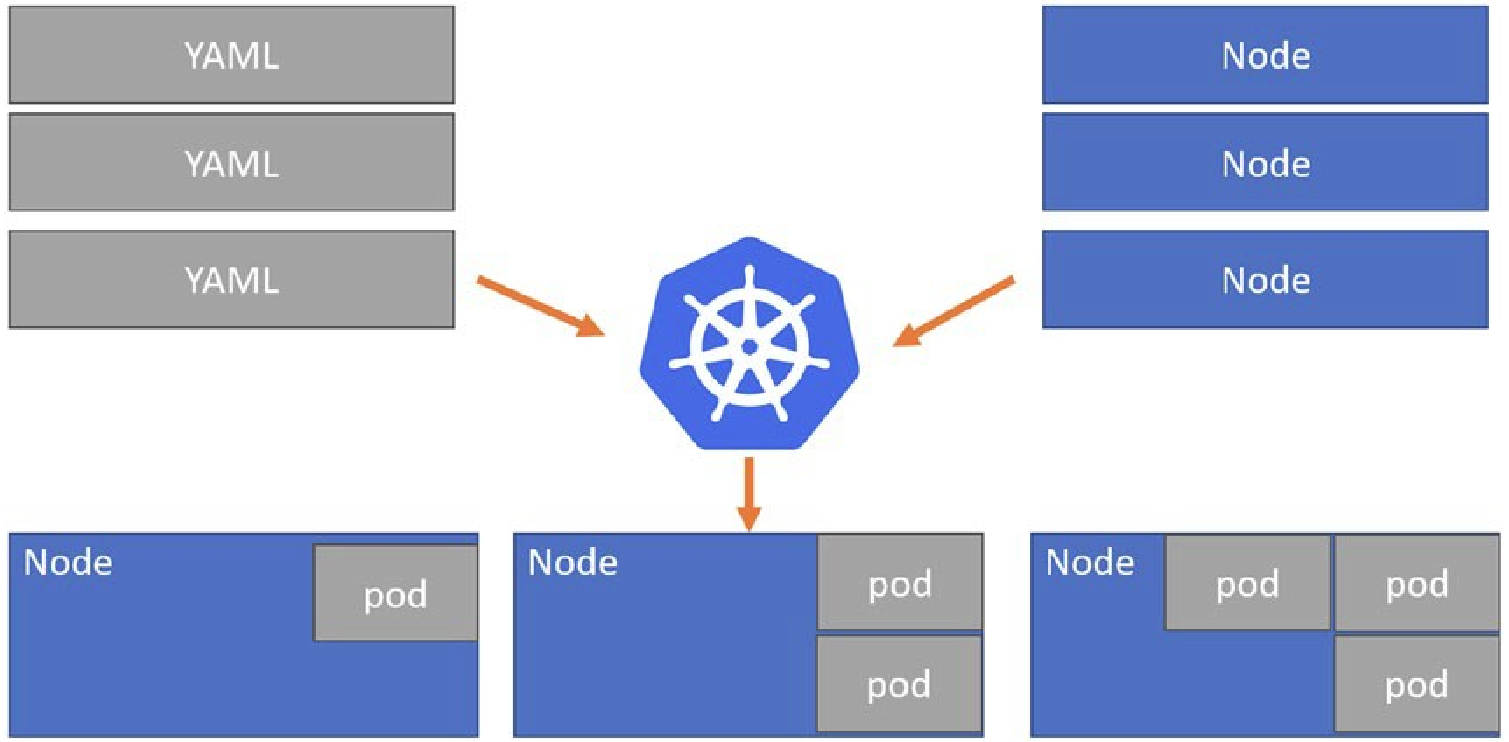
```
docker pull refoman/aspnetsample:v1
```

DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
cb10d8663500	linux/amd64	---	85.38 MB

Kubernetes

- container orchestration tool – open source
- Docker Compose & K8S allows you to declaratively define multiple services.
- Unlike Docker Compose, it can manage an entire cluster of machines (as opposed to Docker Compose running everything on a single machine).
- Kubernetes can handle your need to deploy multiple web servers (behind a load balancer) to handle an increase in traffic

K8S Model



- Kubernetes Objects are defined by YAML files
- various “kinds,” such as Deployment, Service, etc
- A “Node” is a machine where one or more containers can be deployed
- Each machine must be running a container runtime (most commonly Docker) and a suite of other Kubernetes components
- Given a cluster and some YAML, Kubernetes will follow the instructions in the YAML and create “pods” on the nodes. Pods are typically running a single container

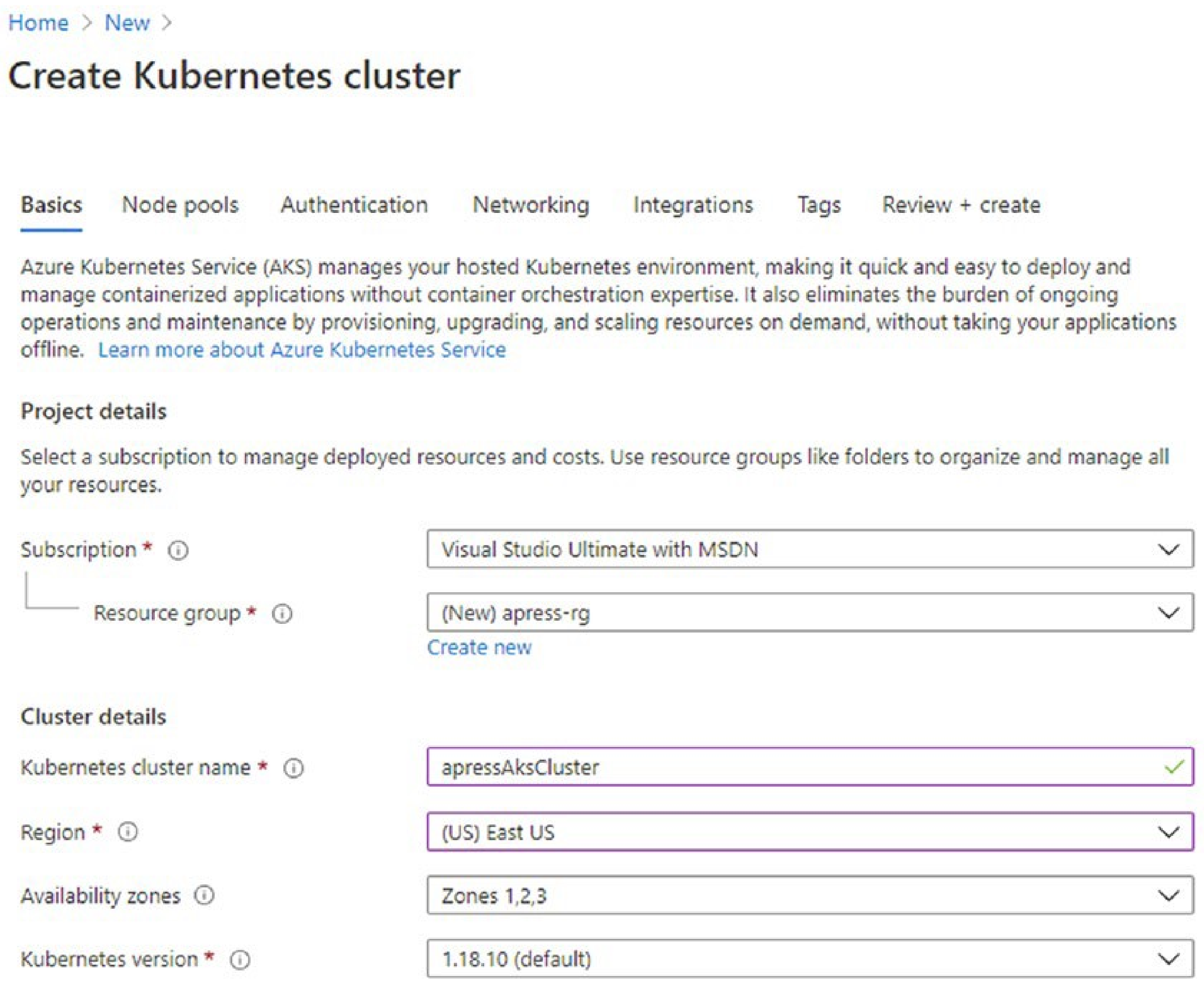
Orchestration

- Declarative (we tell Kubernetes what to do instead of doing it ourselves)
- Resilient (Kubernetes will recover crashed pods on its own)
- Scalable (we specify the number of instances we want to deploy and
- Kubernetes makes sure that the number of instances continues to exist)
- Standardized “infrastructure as code” (YAML files all follow the Kubernetes API, and deployments are all managed in containers)

Kubernetes on Azure: AKS

- create an AKS cluster through
- the Azure Portal website UI,
- the Azure Cloud Shell command line (in-browser on Azure Portal), or
- from your local command line by installing Azure CLI

AKS



Home > New >

Create Kubernetes cluster

Basics | Node pools | Authentication | Networking | Integrations | Tags | Review + create

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. Learn more about Azure Kubernetes Service

Project details

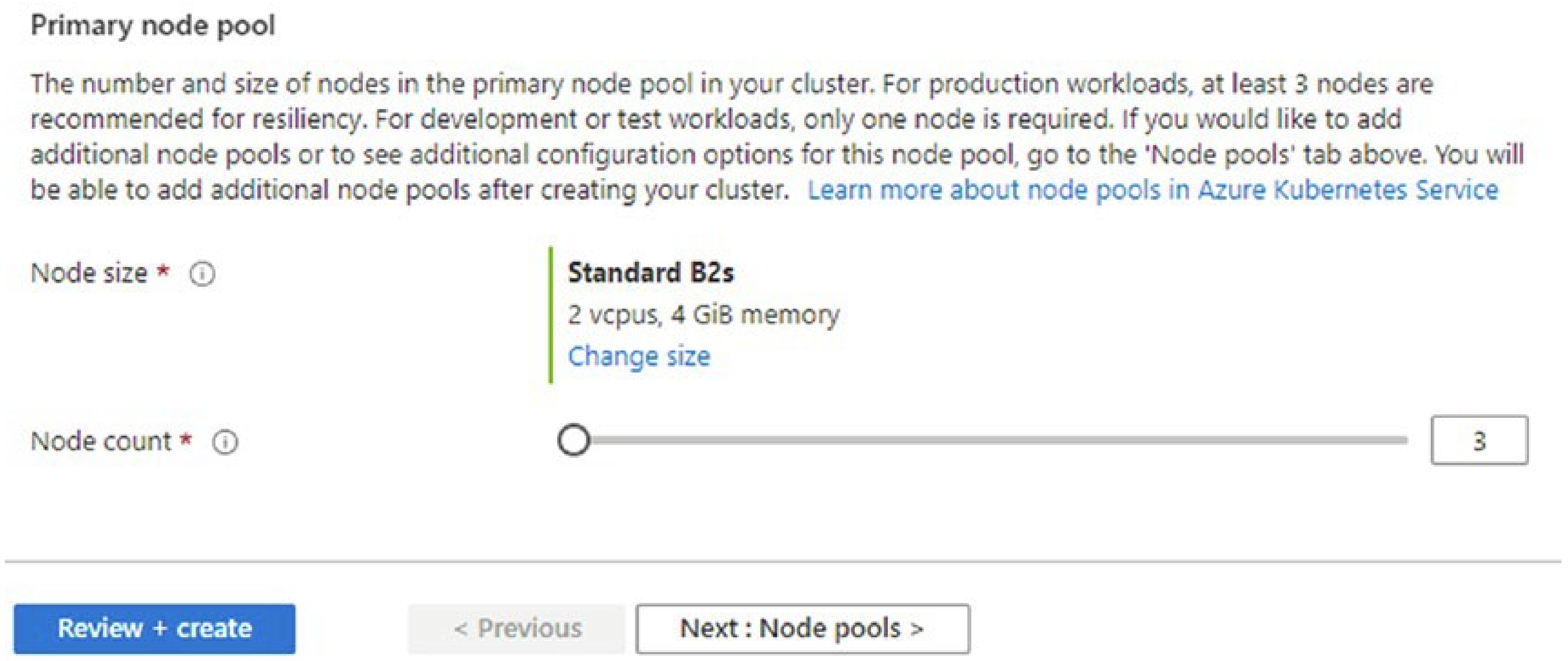
Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Visual Studio Ultimate with MSDN
Resource group *	(New) apress-rg Create new

Cluster details

Kubernetes cluster name *	apressAksCluster
Region *	(US) East US
Availability zones	Zones 1,2,3
Kubernetes version *	1.18.10 (default)

Primary node pool



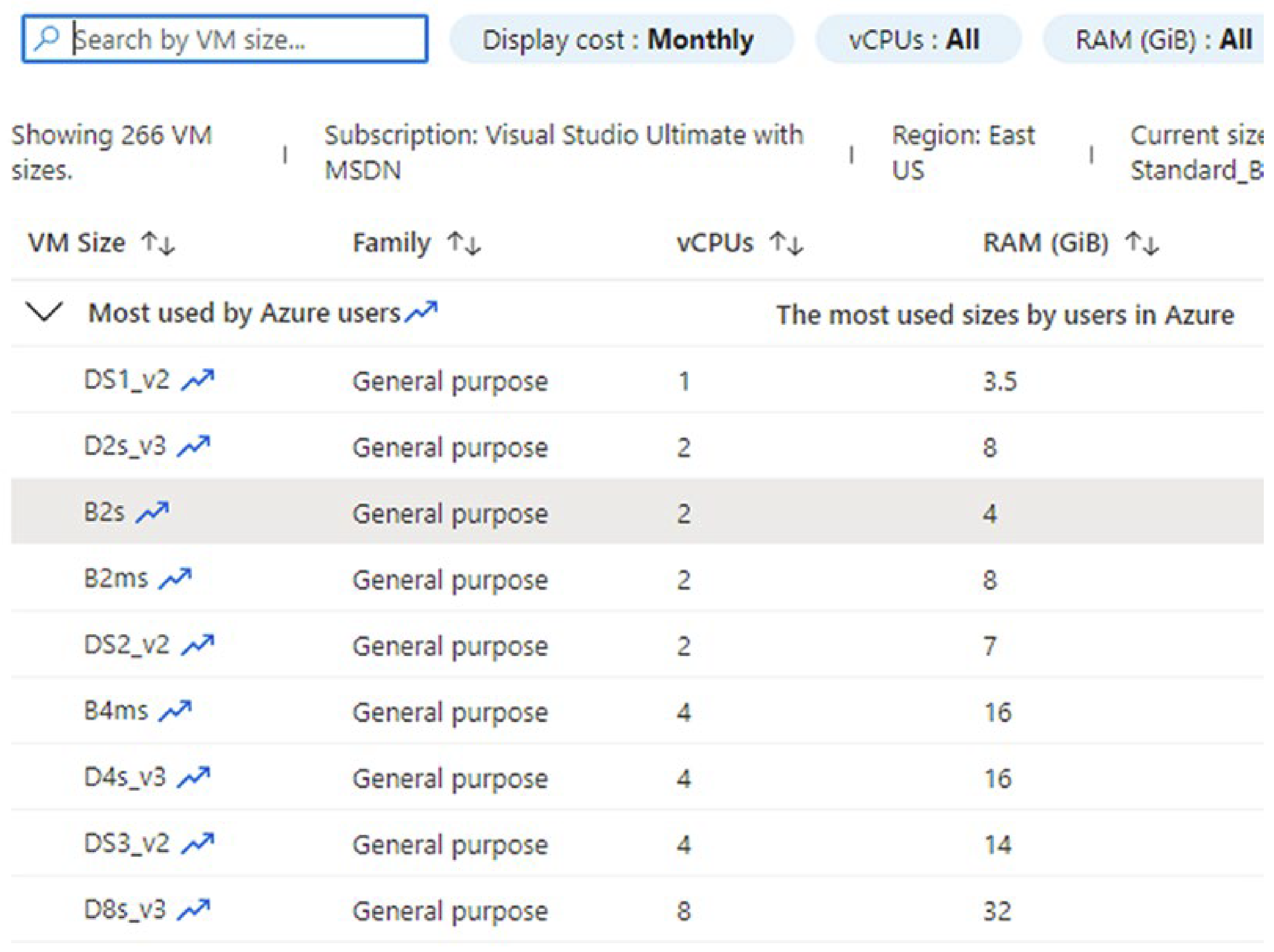
Primary node pool

The number and size of nodes in the primary node pool in your cluster. For production workloads, at least 3 nodes are recommended for resiliency. For development or test workloads, only one node is required. If you would like to add additional node pools or to see additional configuration options for this node pool, go to the 'Node pools' tab above. You will be able to add additional node pools after creating your cluster. Learn more about node pools in Azure Kubernetes Service

Node size * ⓘ — **Standard B2s**
2 vcpus, 4 GiB memory
Change size

Node count * ⓘ — 3

Select a VM size



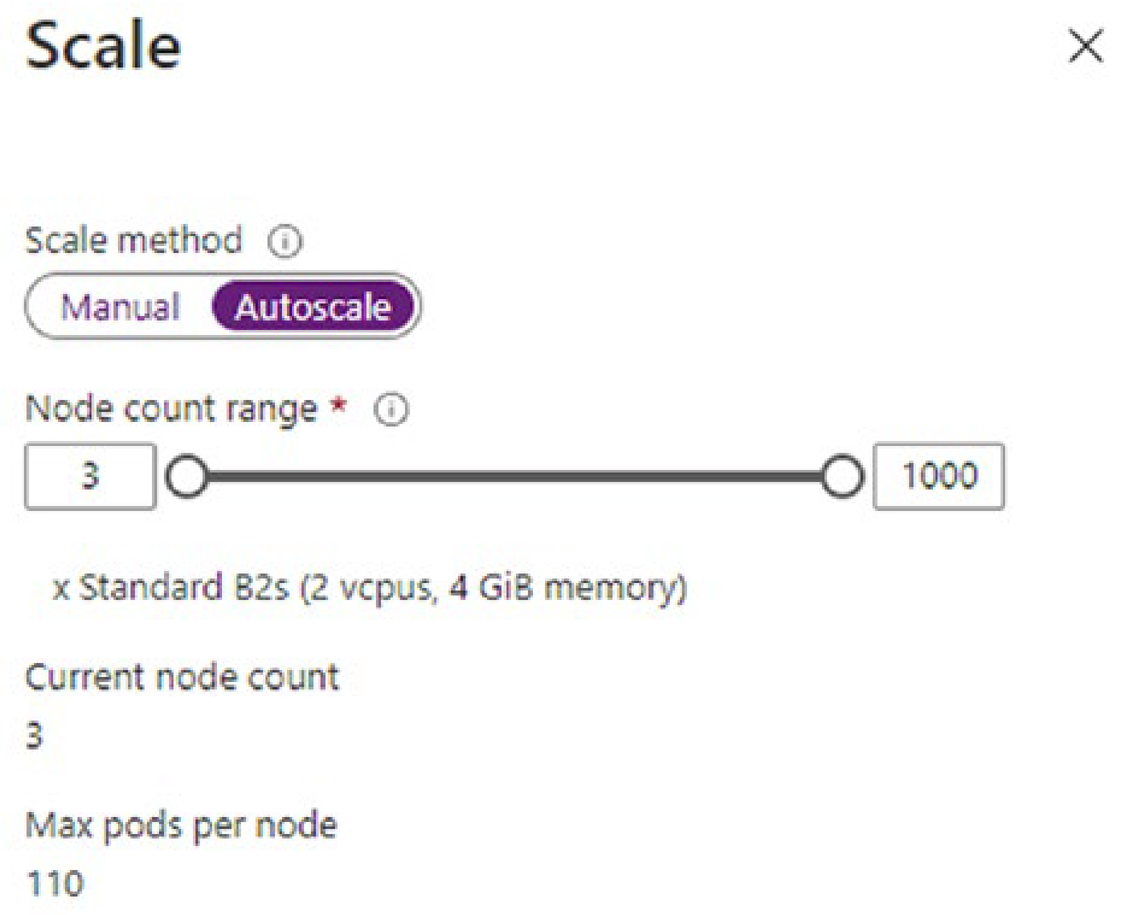
Search by VM size... Display cost : **Monthly** vCPUs : **All** RAM (GiB) : **All**

Showing 266 VM sizes. | Subscription: Visual Studio Ultimate with MSDN | Region: East US | Current size Standard_B

VM Size ↑↓	Family ↑↓	vCPUs ↑↓	RAM (GiB) ↑↓
Most used by Azure users			The most used sizes by users in Azure
DS1_v2	General purpose	1	3.5
D2s_v3	General purpose	2	8
B2s	General purpose	2	4
B2ms	General purpose	2	8
DS2_v2	General purpose	2	7
B4ms	General purpose	4	16
D4s_v3	General purpose	4	16
DS3_v2	General purpose	4	14
D8s_v3	General purpose	8	32

Autoscaling

- ATTENTION:
- Careful thought and monitoring,
- MAYBE surprised by a large bill
- (if you set your node count max too high) and/or a microservice system that fails under stress (if you set your node count max too low).



Scale

Scale method

Manual [✓ Autoscale]

Node count range *

3 — 1000

x Standard B2s (2 vcpus, 4 GiB memory)

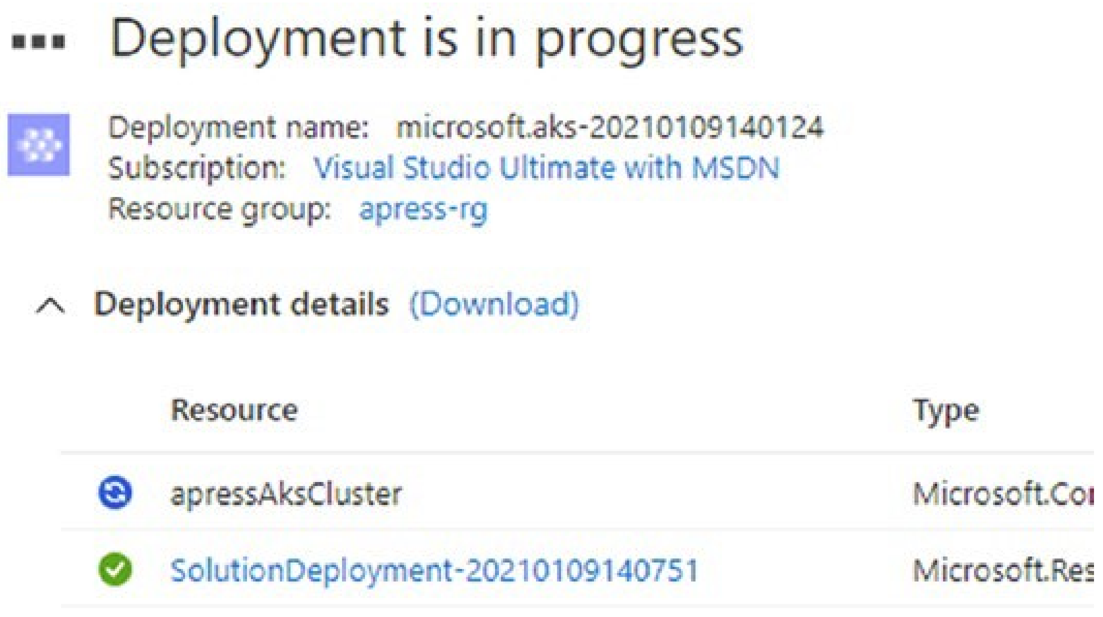
Current node count

3

Max pods per node

110

Deployment in progress



Deployment is in progress

Deployment name: microsoft.aks-20210109140124
Subscription: Visual Studio Ultimate with MSDN
Resource group: apress-rg

Deployment details (Download)

Resource	Type
apressAksCluster	Microsoft.Co
SolutionDeployment-20210109140751	Microsoft.Res

Connect to the Cluster (Azure CLI)

- kubectl version –client

az aks get-credentials --resource-group apress-rg --name apressAksCluster Merged "apressAksCluster" as current context in C:\Users\myusername\.kube\ config

Define Kubernetes Objects

- loadbalancer.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: myweb
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: mywebservice
```

```
kubectl apply -f app.yaml

kubectl apply -f loadbalancer.yaml
```

app.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mywebservicedeployment
spec:
  selector:
    matchLabels:
      app: mywebservice
  replicas: 3
  template:
    metadata:
      labels:
        app: mywebservice
    spec:
      containers:
      - image: "nginxdemos/hello"
        resources:
          limits:
            memory: 128Mi
            cpu: 500m
        imagePullPolicy: Always
        name: mywebservice
        ports:
          - containerPort: 80
```

Let's get a listing of all the pods:

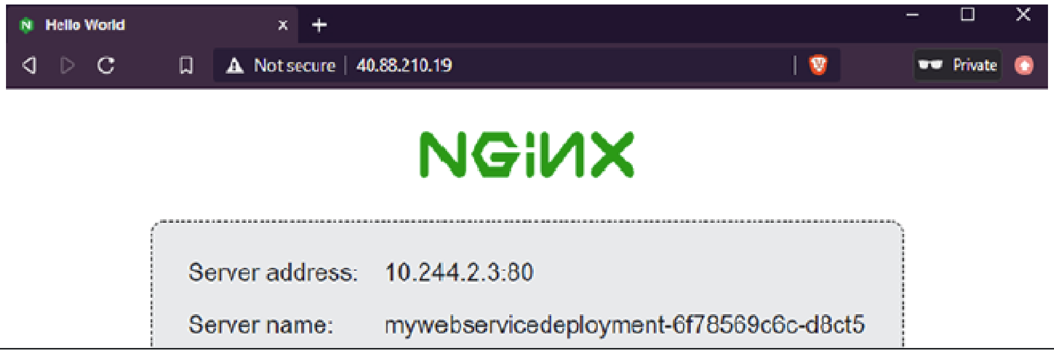
```
> kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
mywebservicedeployment-6f78569c6c-cbndf   1/1     Running   0          108s
mywebservicedeployment-6f78569c6c-d8ct5   1/1     Running   0          108s
mywebservicedeployment-6f78569c6c-x2bjx   1/1     Running   0          108s
```

Make a note of the pod names. They have a randomly generated element (e.g., "d8ct5") at the end of each of their names that we will see again in a minute.

And we can get a listing of all the services:

```
> kubectl get services
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP      10.0.0.1       <none>         443/TCP        9m13s
myweb        LoadBalancer   10.0.219.161   40.88.210.19   80:31679/TCP   2m5s
```

Notice that the LoadBalancer called "myweb" has an external IP address. Open this IP in a web browser (the IP you get will look different than mine). You should see a page very similar to Figure 8-21.



Server address: 10.244.2.3:80

Server name: mywebservicedeployment-6f78569c6c-d8ct5