WorkShop 1

Step by Step Dotnetcore API

The goal is to get familiar with docker, and deployment environment

workshop 1
- We will create a sample app via VisualStudio WebApp template
- Create a docker image for the app
- Run the app in a container on Local

workshop 2
- Deploy the docker image to cloud (Azure kubernetes Services - AKS)
- Access the app from the browser

Prequisitos
- install Visual Studio 2017 Professional - 网上免费下载，公司email login
- install .NET Core SDK 2.2 in Visual Studio
- Hyper-V (from start, search "Turn Windows features on or off" to turn "Hyper-V" ON and [OK])
  win7 的 "Hyper-V Tools" 在 Remote Server Admin Tools - Role Admin Tools 里.
- Docker Desktop for Win10 Docker tool box for Win7 → Need to create user in DockerH

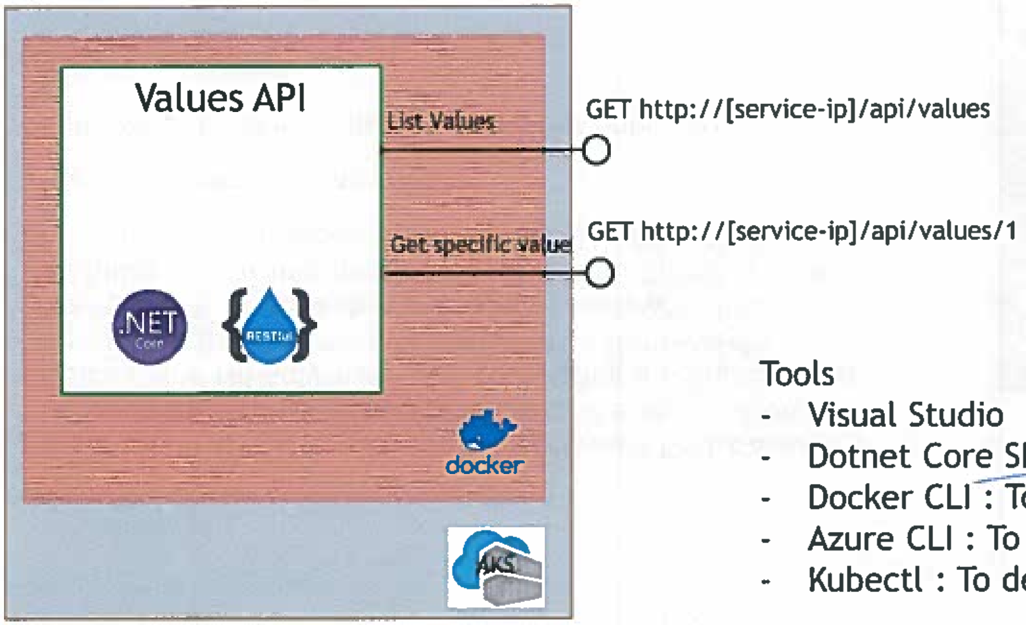
11061987
ALLhome512

0

Table of Contents

- Target Architecture
- Docker / Containers
  - What is containers
  - Install Docker on Local) Docker engin 可以 run 多个 container
    Win10 Dockerworkshop ← or DockerToolbox (Win7)
  - Hello World — In dockhub (hub. docker. com) search "hello-world" 这是 docker official image
- Create Values API and Run Locally
  - .NET Core / Visual Studio 现成的 webapp template .
- REST API with DotnetCore
  - Dotnetcore vs .NET Framework 多OS vs Windows OS
  - Setting Up PC
  - Creating API
  - Running in a container
  - Deploying to Kubernetes
  - Adding Swagger capability
  - Adding End-points

Target Architecture – What are we building

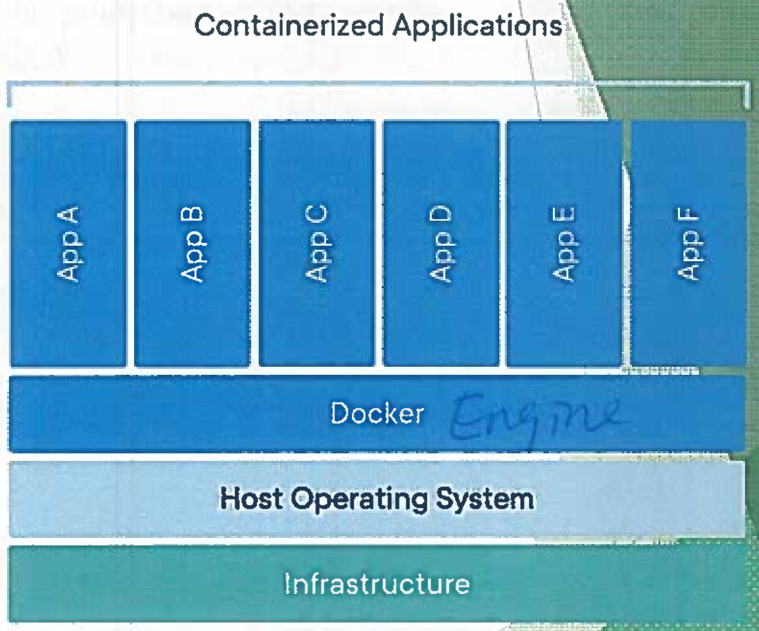

Values API
List Values
Get specific value
GET http://[service-ip]/api/values
GET http://[service-ip]/api/values/1
.NET Core
RESTful
docker
AKS

Tools

- Visual Studio
- Dotnet Core SDK — comandline
- Docker CLI : To build/push docker image
- Azure CLI : To connect to Azure
- Kubectl : To deploy Calculator API

Docker - Containers

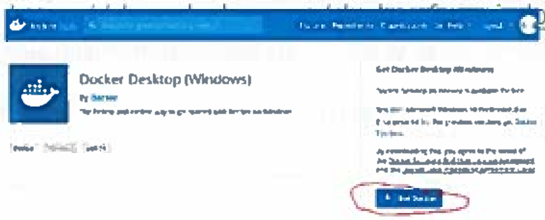
- A container is a standard **unit of software** that **packages** up **code and all its dependencies** so the application runs quickly and reliably from one computing environment to another. A Docker container image is a **lightweight,** 6MB **standalone, executable** package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.
- Standard - portable anywhere
- Lightweight - Only required things for your app
- Secure - Isolated



More info: https://www.docker.com/resources/what-container

Docker - Installing Docker for Windows

- Download Docker Desktop for Windows
  - Hyper-V must be installed on your PC (from Add Remove Programs) from start search "turn on features" check Hyper-V from list [OK]
    / under Remote Server Admin Tools
    在win7 — Role Admin Tools
  - Register to Dockerhub 1106981是用户名. 密码是ALL..512
  - Download and install (win7用Docker toolbox
    - ws/install/



- Login to Docker
  - From Docker for Windows
  - Or run the following command in command line/powershell
    - docker login -u [your-username] -p [yourpassword]

~~这不work~~
界面比cmd要好用.
从start搜powersell

Hands-on Lab

Docker – Docker Hub / Hello World

- Browse Docker-hub
  - Search hello-world docker image → 这是 hub. docker. com 里的 docker official images
- Run hello-world from command-line
  - docker run hello-world 相当于从 docker hub check out hello-world image 到 local
- Some docker-cli commands
  - docker ps : shows running docker containers Containers are running (or stopped) instances of image
  - docker images : shows images exist in local repo Image 是 container 冻结不可变的 snapshot.
  - docker search hello- : lists available images in the remote repo
  - docker build : builds a new docker image from dockerfile
  - docker push : pushes locally built image to remote repo

femon
恶魔
daemon
守护进程

用 dock ps -a 看到 ContainerID 和 Image Name 后用 docker rm ContainerID 再用 docker rmi ImageName

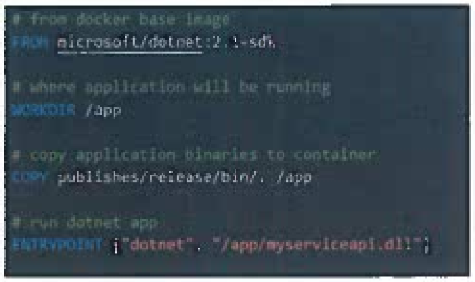
all containers

* 有时需用 exit 跳出回到 C:\Users\j.qi> 下.

Docker - Dockerfile

- Docker can **build images automatically** by reading the instructions from a **Dockerfile.**
- A Dockerfile is a **text document** that contains all the commands a user could call on the command line to assemble an image.

无后缀

- Using docker build users can create an automated build that executes several command-line instructions in succession.
- Sample: dockerfile
  - 

```
# from docker base image
FROM microsoft/dotnet:2.1-sdk

# where application will be running
WORKDIR /app

# copy application binaries to container
COPY publishes/release/bin/. /app

# run dotnet app
ENTRYPOINT ["dotnet", "/app/myserviceapi.dll"]
```

Hands-on Lab

Docker – Azure Container Registry

- ACR – Similar to Github
  - Our private repo
  - Hosted in Azure – Fully managed
- Login to Azure Container Registry
  - docker login osrammarkhamdev.azurecr.io -u [will-be-provided] -p [will-be-provided]

从Azure Web Browser Protol里copy出来

osrammarkhamdev TJ+tX/97·…hhpN

Hands-on Lab

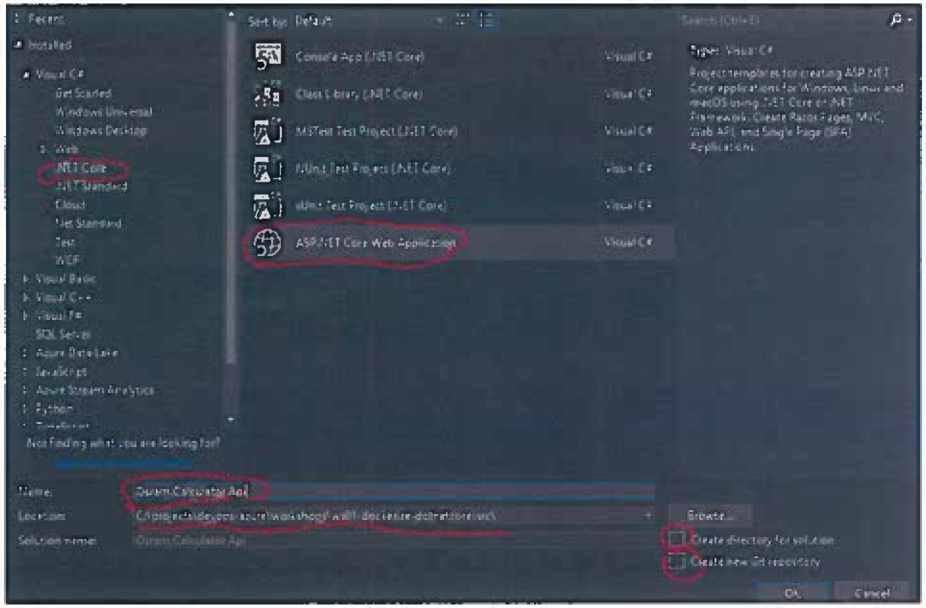
Values API – Setup local

- If not installed
  - Install Visual Studio
  - Install .NET Core 2.2 SDK
    https://dotnet.microsoft.com/download
- Clone git repo
  https://OYesil-ext@dev.azure.com/OYesil-ext/workshops/_git/ws01-dockerize-dotnetcore
- Create a branch under your name, and push that branch to remote
  - git checkout -b branch [firstname].[lastname]   #example: git checkout -b branch omer.Yesil
  - git push -u origin [firstname].[lastname]

Hands-on Lab

Values API - Create Values API

- ▶ From Visual Studio, select New Project from File menu, and create an "ASP.NET Core Web Application" from the templates



Recent
Installed
Visual C#
Get Started
Windows Universal
Windows Desktop
Web
.NET Core
.NET Standard
Cloud
.Net Standard
Test
WCF
Visual Basic
Visual C++
Visual F#
SQL Server
Azure Data Lake
JavaScript
Azure Stream Analytics
Python
Not finding what you are looking for?

Sort by: Default

Search (Ctrl+E)

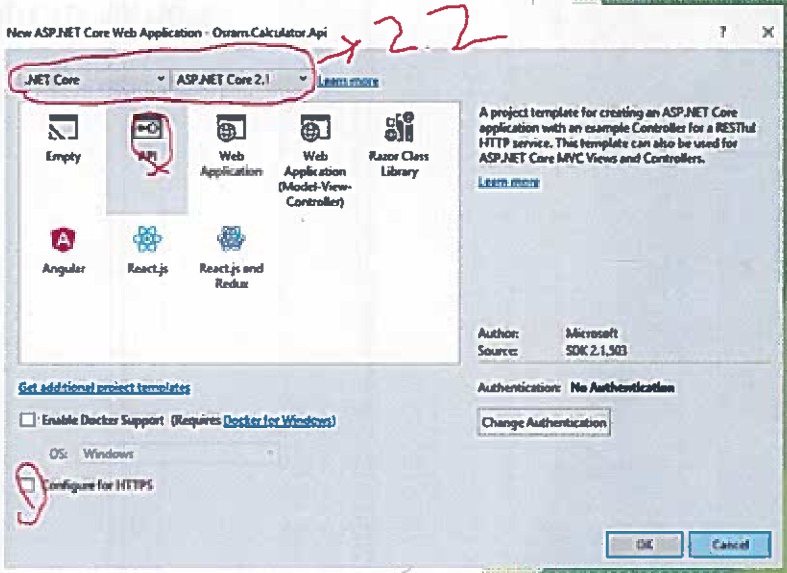
Console App (.NET Core)	Visual C#
Class Library (.NET Core)	Visual C#
MSTest Test Project (.NET Core)	Visual C#
NUnit Test Project (.NET Core)	Visual C#
xUnit Test Project (.NET Core)	Visual C#
ASP.NET Core Web Application	Visual C#

Type: Visual C#

Project templates for creating ASP.NET Core applications for Windows, Linux and macOS using .NET Core or .NET Framework. Create Razor Pages, MVC, Web API, and Single Page (SPA) Applications.

Name: Ozram.Calculator.Api
Location: C:\projects\devops-azure\workshop\will-dockerize-dotnetcore\src\
Solution name: Ozram.Calculator.Api

□ Create directory for solution
□ Create new Git repository



New ASP.NET Core Web Application - Ozram.Calculator.Api

.NET Core | ASP.NET Core 2.1

2.2

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Author: Microsoft
Source: SDK 2.1.503

Authentication: No Authentication

□ Enable Docker Support (Requires Docker for Windows)

OS: Windows

□ Configure for HTTPS

- ▶ Run the application to test on local

、按 ▶ IIS Express 开始

按 ● 停止 .

dockerfile 内容

docker parent image — FROM microsoft/dotnet:2.2-aspnetcore-runtime AS runtime

create的image的home directory — WORKDIR /app

把这些Visual Studio的directory里的东西copy到image里 — COPY publish/. /app/

ENTRYPOINT ["dotnet", "Osram.Values.Api.dll"]

-创建的dotnet project　　要create的image的名字

放一个dockerfile到这个directory下面

可能叫Solution name更适当

Hands-on Lab

Values API - Publish the API Runtimes

hoover visualsonics.cs files所在的directory. C:\Users\j.qi\source\repos\projectname\projectname

↓ Visual Studio projectname

E:\dooOSRAMCalculatorApi

cmd 或 powershell 里先转到上面的 directory 再做下面

- Publish the Application. Run the dotnet cli command under ../src/Osram.Calculator.Api folder (you can also publish from Visual Studio)
  - `dotnet publish -c release -o publish`

    也可以从Visual Studio里build - Publish 里Publish. Publish之后看Publish folder

- To test the published code run the following under the publish folder　在哪里dll的名字是什么
  - `dotnet Osram.Values.Api.dll` → 先Navigate到Publish directory里 把名字改成Jia.dll

    ↳server开始run了 local

    ↓ 我的Visual Studio的project名字叫Jia

    dll和Publish folder里面所有东西即是一个protoble的microservices
  - Verify : http://localhost:5000/api/values　与在Visual Studio里compile弹出的browser内容一样都是["value1", "value2"]

    ↳默认是5000端
  - Now you can kill the running dotnet process (CTRL+C)

在我的电脑上相当于一个local service

(server里面一堆services)

↓ 我的电脑

↓ Jia.dll是一个service

Hands-on Lab

Values API – Dockerize

- Create a new text file named dockerfile under ../src/Osram.Calculator.Api folder, and set its' content as below:

```
FROM microsoft/dotnet:2.2-aspnetcore-runtime AS runtime

WORKDIR /app
COPY publish/. /app/
ENV ASPNETCORE_URLS=http://+:5000

ENTRYPOINT ["dotnet", "Osram.Values.Api.dll"]
```

→ remote ~~folder~~ server的current folder里app

→把local 的publish folder里所有东西都copy ~~移~~到remote server的app里

↳根据这个位置：要把dockerfile放到publish folder的上层folder里

—改成与publish folder里Jia.dll一个名字

~~删除~~ Jia.dll (在remote server里跑的话，run的第一个file就是entrypoint Jia.dll()

copy过去的时候Jia.dll名字不会变

- Run the following command to build docker image

```
docker build -t osram.values.api .
```

—如果用docker hub repository的话 docker build -t 11061981/jia

-tag dockerhub用户名

current directory of docker file.

这个名字必须lower case. 可以和Jia.dll不同名. 这个会是docker image的名字

~~doc~~ jia1st docker image 我取了[illegible]名字

- Verify : `docker images`

→在我的电脑的docker里生成一个docker image

→要navigate到有dockerfile的directory下再输入指令.

→打错可Ctrl+C退出

→必须先run docker Quickstart Terminal (由docker Toolbox install的) — 这是一个docker的VM.
只有这个跑起来才能run docker指令，即docker指令让这个docker VM跑一堆东西才能生成docker image

Hands-on Lab

Values API – Run Docker Container on Local

这个比dotnet Jia.dll 要进一步. dll是用dotnet run local dll service, 这个是用docker 基于image模板 create a container 在local 并run service

模拟的server — 1对1的关系 的microservices

- Run the following command to create a container from the image we created in the previous slide:
  - `docker run ~~osram.values.api~~ jia1stdockerimage -p 5000:~~5000~~ 80` (docker默认是80)

    windows默认的是5000 (visual studio用的)
  - To verify, browse: http://~~localhost~~ 192.168.99.100:5000/api/values

    在Docker Quickstart Terminal里看到docker VM with IP 192.168.99.100
- Kill the process (CTRL-C)

也可以

docker run -p 80:80 jia

http://192.168.99.100/api/values

默认的80不用写. 写80也work

也可以

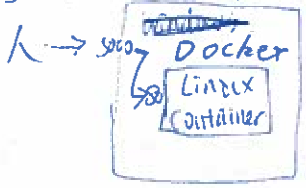
docker run -p 1234:80 jia

http://192.168.99.100:1234/api/values

5000 is the port for human to enter windows

80 is the port to enter Linux container

5000:80 is port forwarding



我电脑的IP是192.168.99.1 ∴ localhost是192.168.99.1 但 docker is configured to use the default machine with IP 192.168.99.100 ∴ 为什么localhost不work ∵ 要run docker container on local docker.

Hands-on Lab

Values API – Connecting to a running Container

这个想展示的是 -d 是running at backgrond 不出现 Press Ctrl+C to shut down 的提示，但用

docker run -d --name #{container_name} #{image}

- Run the previous command with additional parameter (-d): 更好 detach, run at background
  - `docker run osram.values.api -p 5000:5000 -d`

docker run --name jiacontainer jia

docker run -d --name jiacontainer2 jia

- To see the running containers
  - `docker ps`

interactively

- Connect to the running container
  - `docker exec -it [containerId] /bin/bash` — 打开linux系统下的command line

用前三个字母即可 f32

root@f32e71473edb:/app# ← 光标 (已进入了container里并打开了cmd)

用exit退出

↳ dockerfile 里定义的 WORKDIR /app (或叫home directory)

装docker Toolbox时装了Oracle VM VirtualBox Manager. 这个是个Linux VM

当launch Docker Quickstart Terminal之后 这个Oracle VM就会running 在 background

如果pause 或 close - power off 这个oracle 则docker 指令就不work了

docker run 什么的

- Now we can run any Linux bash command
- To exit from the container, type ‘exit’ and Enter

Hands-on Lab

Values API – Pushing Local Docker Image to Remote Docker Repo

- Login to docker repo (ACR) / 从Azure browser portal里得到的
  USERNAME and PASSWORD will be provided
  - `docker login osrammarkhamdev.azurecr.io -u USERNAME -p PASSWORD`

    osrammarkhamdev TJ+tX/g7PLl GaZRIW7vyvA 3fs PQ3/hhpN

- Tag the local docker image
  - `docker tag [your-docker-image-id] osrammarkhamdev.azurecr.io/workshop-values-api:yourfirstname.~~yourlastname~~`

    可改成jia

比如 docker tag 1e497596 69e7 osrammarkhamdev.azurecr.io/~~osram.values-api~~ . 1.0

tag local image with ID "1e4.." into "osrammarkhamdev/jia" repository with ~~version~~ 1.0

- Push the image
  - `docker push osrammarkhamdev.azurecr.io/workshop-values-api:yourfirstname.yourlastname`

比如 docker push osrammarkhamdev.azurecr.io/jia:1.0

可到Azure browser portal里看repository里多了一个image

- Now we can run any Linux bash command
- To exit from the container, type ‘exit’ and Enter

pull it — docker run osrammarkhamdev.azurecu.io/jia:1.0 -p 1010:5000

或用docker hub的repository 用户名 11061981 密码 ALL… 512

在Docker Quick Start Terminal里面

① docker login ↙ 直接

② Username(11061981):↙

③ Password: 不会显示的↙

④ docker images 看image ID前三位 2be

⑤ docker tag 2be 11061981/jia — repository的名字 其不存在dockerhub会自动建

docker push 11061981/jia

enable SonarQube in Jenkins

CI: Developer Check code in bitbucket → Jenkins run unit test, Code Quality test & Automated tests $\xrightarrow[\text{pass}]{\text{if}}$ Jenkins creat Docker Image → push to ACR (Azure Container Registry)

Jenkinsfile stages: Build → Unit Tests → SonarQube → Docker Build (含 docker build & docker push)

Deploy. Jenkinsfile stages:

CD: Jenkins tell kubernetes to grab the Docker image from ACR with ACR's ~~certain~~ secreate (password) to create container (kubernetes deployment objects) in Pod

然后 Deploy 到 AKS 里 (Azure kubernetes/Container Services)

Deploy a load balance services 到 AKS，得到 External IP (一个services 一个 external IP) 一指向一堆同名的pods

每个pod有不同的 internal IP.

之后可从browser 访问了 (可访问到内部所有pods)

指这个external IP

services的

Cloud Good Practises: Define Domain → Define Multiple microservices under domain → Define Data Contract (Swagger endpoints) → Define Use Case (test cases) → Automate deployment with tests

DEPLOYMENT

workshop 2 :

- kubernetes concept
- Creating kubernetes deployment objects for a sample API (with Docker Image) and deploying them to Cloud (Azure kubernetes / Container Services - AKS)
- Accessing the API through browser

Kubernetes Cluster

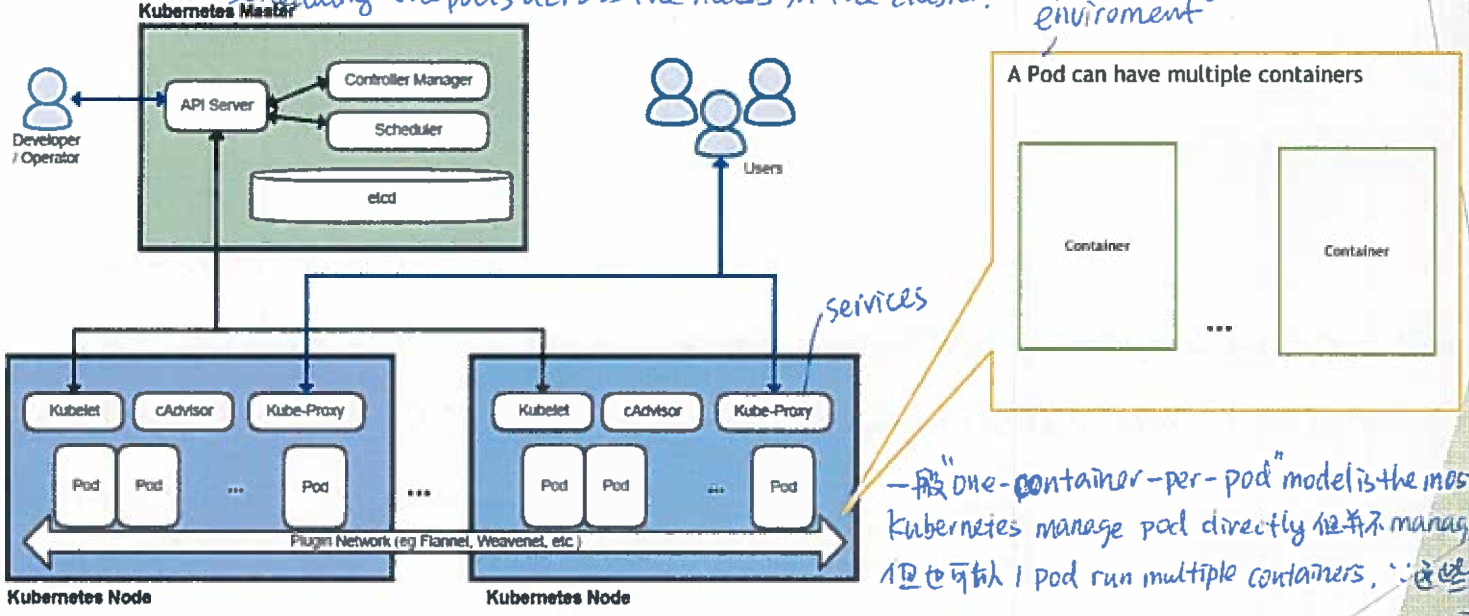
1 container <-> 1 image

1 kubernete pod <-> multiple containers
" deployment environment

1 kubernete cluster <-> a collection of pods

master automatically handles scheduling the pods across the nodes in the cluster.

Dev/QA/Staging/Production enviroment



一般"one-container-per-pod" model is the most common Kubernetes use
Kubernetes manage pod directly 但并不 manage container directly
但也可以 1 pod run multiple containers, ∵这些 container 要 work together 组成app

A pod always runs on a Node. A Node is a worker machine in kubernetes and maybe either a virtual or a physical machine. depending on the cluster.

1 cluster -> multiple nodes & their master
1 node -> multiple pods
1 pod -> multiple containers
1 container - 1 image
(can be multiple images)

Accessing to Kubernetes

commandline

Kubernetes CLI : kubectl

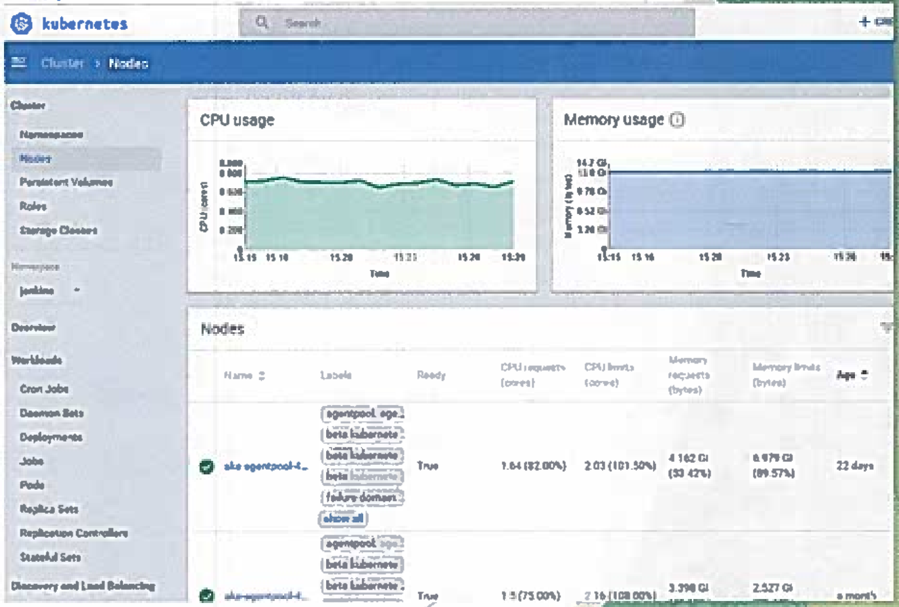
https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-with-powershell-from-psgallery

SAMPLE CALLS:
kubectl create -f mydeploymentfile.yml -n testnamespace
kubectl get nodes
kubectl get pods -n testname

Kubernetes Dashboard

从Azure Dashboard里launch Kubernetes Dashboard，而Proxy：. Kubernetes Dashboard 的URL看起来像本机 IP.

127.1.0.0//...

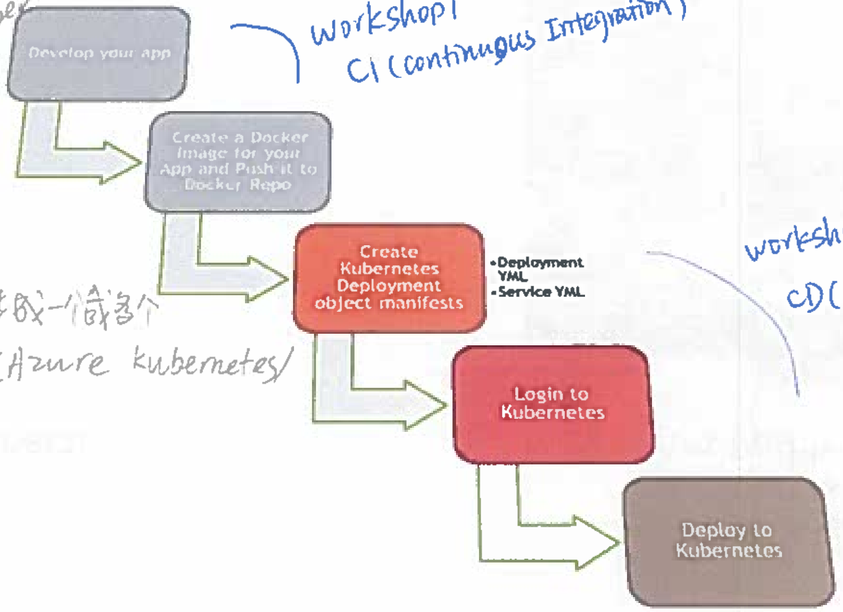


Nodes

	Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
✓	aks-agentpool-4...	agentpool: age... beta.kubernete... beta.kubernete... beta.kubernete... failure-domain... show all	True	1.64 (82.00%)	2.03 (101.50%)	4.162 Gi (53.42%)	6.979 Gi (89.57%)	22 days
✓	aks-agentpool-4...	agentpool: age... beta.kubernete... beta.kubernete...	True	1.5 (75.00%)	2.16 (108.00%)	3.398 Gi	2.527 Gi	a month

Deployment Progress

我感觉CI是根据developer code 生成 docker image 存在 ACR (Azure Container Registry) 里


Develop your app
Create a Docker Image for your App and Push it to Docker Repo
Create Kubernetes Deployment object manifests
• Deployment YML
• Service YML
Login to Kubernetes
Deploy to Kubernetes

workshop1
CI (continuous Integration)

workshop2
CD (continuous Deployment)

CD是根据 docker image 生成一个或多个 docker container 存在AKS (Azure kubernetes/ Container Services) 里

asp
mic
microsoft/dotnet-samples:aspnetapp

Creating Values API Kubernetes Manifests

tab doesn't work for yaml，要用space打空格.

sample-api-deployment.yaml ← 放在C:\Users\J.qi下

```
---
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: YOURFIRSTNAME-YOURLASTNAME-api  (jia-qi-api) — POD name
  namespace: workshop
spec:
  replicas: 3 → 在Pod里会出现3个~~container~~ PODS ~~container~~ (Pod 名字的后面有不同的random #) 写0的话会remove所有之前已在Pod里的)
  template:
    metadata:
      labels:
        app: YOURFIRSTNAME-YOURLASTNAME-api  (jia-qi-api) ← same
    spec:
      containers:
      - name: YOURFIRSTNAME-YOURLASTNAME-api (jia-qi-api) — POD里的container name
        image: osram.repo/fmui:betel.0
        ports:
        - containerPort: 80
        env:
          - name: MY_TEST_ENV
            value: "HELLO THERE"
      imagePullSecrets:
      - name: dockerhub — dockerhub的secret
```

docker 默认 port是80

也可是Dockerhub里现成的image，比如 microsoft/dotnet-samples: aspnetapp

dockerhub的secret 如果是从ACR里拿image，则改成ACR的secret(密码) ↳Azure dashboard里拿

a list of containers
You can repeat这一段
则可在Pod里加入多个container.

sample-api-service.yaml

```
---
apiVersion: v1
kind: Service
metadata:
  name: YOURFIRSTNAME-YOURLASTNAME-api-svc  (jia-qi-api-svc)
  namespace: workshop
spec:
  type: LoadBalancer ← 要求kubernete loadbalance
  selector:
    app: YOURFIRSTNAME-YOURLASTNAME-api  (jia-qi-api)
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 80    #POD's port number
```

3个pod一样的lable不同IP

Deployment时可能replicate3个pods
这三个pods会分得不同的IP，则需要
一个service 从 External IP接收human的访问request
按loadbalance的方法发给某个pod处理

见下页黑库

Accessing to Kubernetes Cluster

- Install and Login to Azure CLI
  - Install : https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest
  - Login :

    ```
    az login
    ```

    + you need to login Azure to use kubernetes.
    – this cmd will launch the browser & choose j.qi@osram.com to login, the browser will close
- Install Kubernetes CLI (用 kubernetes command-line client manage kubernetes cluster)
  - https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-with-powershell-from-psgallery
  - OR

    ```
    az aks install-cli
    ```

    – need to add change enviroment PATH, add "C:\Users\j.qi\.azure-kubectl".
- Configure kubectl for our cluster = connect to a kubernete cluster (credential-subscription-resource group-resource name)

  ```
  az aks get-credentials --resource-group kubernetes-dev --name osram-markham-k8s-dev
  ```

  Azure Kubernete Services

  刚才已经通过"az login"了 ∴ AKs可grab你的credentials

  -- subscription "PAY-AS-You-Go Dev/Test"

  az login return里"name" field
- Verify the kubectl configuration

  ```
  kubectl get nodes
  ```

Deploying to Kubernetes

- To deploy the application
  `kubectl create -f sample-api-deployment.yaml`
  To verify app deployment
  `kubectl get pods -n workshop` ← 在workshop的domain下
  `kubectl get deployments -n workshop`

- To deploy the service (load balancer service)
  `kubectl create -f sample-api-service.yaml`
  To verify the service deployment and get the IP address of it
  `kubectl get services -n workshop`

```
kubectl
```

- Copy/paste the External IP address on Chrome(or any browser,) and browse it

Connecting to a POD running in Kubernetes

- Similar to docker exec command, we can use kubectl exec command to connect (kind a SSH) to a running container
  - Get your container name

    ```
    kubectl get pods -n workshop
    ```

    Name listed in return

  - Connect to the Container

    ```
    kubectl exec -it [CONTAINERNAME] /bin/bash -n workshop
    ```