CLOUD COMPUTING SYSTEMS

Lab 10

João Resende, Nuno Preguiça

(jresende_at_fct.un.pt, nuno.preguica_at_fct.unl.pt)

GOAL

In the end of this lab you should be able to:

- Understand how to create a docker image
- Know how to launch a container in Azure Containers services

- Understand how to create a (simple) Kubernetes config file
- Understand how to deploy a Kubernetes-based system

GOAL

In the end of this lab you should be able to:

- **Understand how to create a docker image**
- Know how to launch a container in Azure Containers services

- Understand how to create a (simple) Kubernetes config file
- Understand how to deploy a Kubernetes-based system

DOCUMENTATION

Dockerfile

https://docs.docker.com/engine/reference/builder/

Azure Container Instances

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

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

CREATING A DOCKER IMAGE

Create a new directory.

Create a Dockerfile specifying the new image (name must be Dockerfile – Dockerfile.txt, etc. is not good).

DOCKERFILE (EXAMPLE)

```
FROM tomcat:9.0-jdk11-openjdk

WORKDIR /usr/local/tomcat

ADD scc2122-backend-1.0.war webapps

EXPOSE 8080
```

For more info on the commands, check the lecture or the link presented before.

CREATING A DOCKER IMAGE

Create a new directory.

Create a Dockerfile specifying the new image.

Copy all resources to be copied to the image to the new directory.

Run the command:

```
docker build -t tagname directory
```

Example:

```
docker build -t nunopreguica/scc2122-app dir
```

MAKING THE IMAGE AVAILABLE

Alternatives:

1. Use Docker Hub
2. Create Microsoft Repository

MAKING THE IMAGE AVAILABLE

Alternatives:

1. **Use Docker Hub**
2. Create Microsoft Repository

Deploy new image to Docker Hub

Create new account

https://hub.docker.com/signup

Run the command to push the image to Docker Hub registry

```
docker push tag
```

Example:

```
docker push nunopreguica/scc2122-app
```

RUNNING THE IMAGE LOCALLY

To run the image locally you can run:

```
docker run --rm -p 8080:8080 nunopreguica/scc2122-app
```

Your image will be available at URL:

http://localhost:8080/name_of_war_file

GOAL

In the end of this lab you should be able to:

- Understand how to create a docker image
- **Know how to launch a container in Azure Containers services**

- Understand how to create a (simple) Kubernetes config file
- Understand how to deploy a Kubernetes-based system

START A (STANDALONE) CONTAINER IN AZURE

Create a resource group, if needed.

```
az group create --name name --location  loc
```

Example:

```
az group create --name scc2122-cluster-4204 --location westeurope
```

START A (STANDALONE) CONTAINER IN AZURE (2)

Start a container.

Example:

```
az container create --resource-group scc2122-cluster-4204 --name scc-app --image
nunopreguica/scc2122-app --ports 8080 --dns-name-label scc-discord-4204
```

`--dns-name-label name` : dns name label for container with public IP

`--port port1 … portn` : list of ports to open

`--environment-variables prop=val …` : list of environment variable (these values can be read in your application using System.getenv(…))

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

START A (STANDALONE) CONTAINER IN AZURE (2)

Start a container.

Example:

```
az container create --resource-group scc2122-cluster-4204 --name scc-app --image
nunopreguica/scc2122-app --ports 8080 --dns-name-label scc-discord-4204
```

Check the output for the ipAddress of the container. The URL for accessing the application should be something like (depending on the war file and DNS label specified):

```
http://scc-discord-4204.westeurope.azurecontainer.io:8080/scc2122-
backend-1.0/ctrl/version
```

Delete a started (standalone) container in Azure (3)

Delete a container given the resource group and name.

Example:

```
az container delete --resource-group scc2122-cluster-4204 --name scc-app
```

TODO

1. Create a Docker image with the code of your application service.
2. Push the image to a registry (suggestion: use Docker Hub)
3. Create a Docker image with artillery and your tests.
   - Use a node.js image as base
   - Install artillery and the other dependencies
   - Copy your tests to the image.
4. Push the image to a registry (suggestion: use Docker Hub)
5. You can now try to start your server in one region and have artillery clients in different regions making calls to your server.

GOAL

In the end of this lab you should be able to:

- Understand how to create a docker image
- Know how to launch a container in Azure Containers services

- **Understand how to create a (simple) Kubernetes config file**
- Understand how to deploy a Kubernetes-based system

DOCUMENTATION

Kubernetes

https://kubernetes.io/docs/home/

Kubernetes service at Azure

https://docs.microsoft.com/en-us/azure/aks/

https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough

KUBERNETES KEY CONCEPTS

Pod: encapsulates an application's container (or multiple containers), storage resources, a unique network IP, and options that govern how the container(s) should run.

Service: a Service is an abstraction which defines a logical set of Pods and a policy by which to access them.

Volume: a volume is a directory which is accessible to the Containers in a Pod. A Kubernetes volume has the same lifetime of the Pod that encloses it.

Namespace: Namespaces provide a scope for names. Names of resources need to be unique within a namespace, but not across namespaces.

KUBERNETES KEY CONCEPTS (2)

Deployments: A Deployment provides declarative updates for Pods and ReplicaSets.

ReplicaSet: A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time.

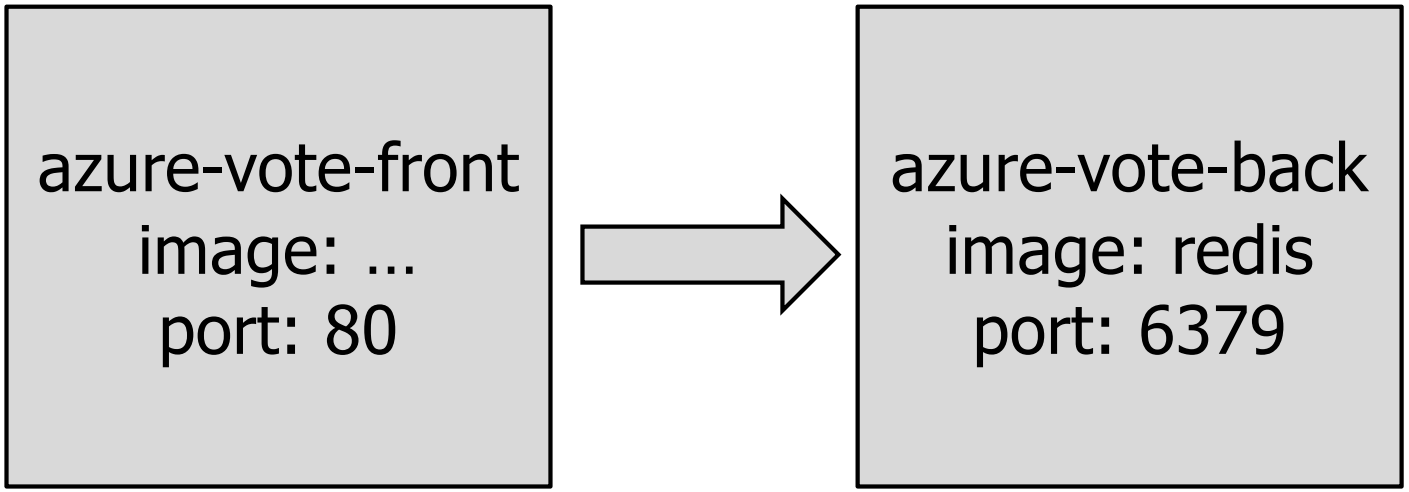
DaemonSet: A *DaemonSet* ensures that all Nodes run a copy of a Pod. As nodes are added/removed to the cluster, Pods are added to/deleted from them.

StatefulSet: StatefulSet is the workload API object used to manage stateful applications.

Job: A Job creates one or more Pods and ensures that a specified number of them successfully terminate

SIMPLE KUBERNETES EXAMPLE

From: https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-back           # name of deployment
spec:
  replicas: 1                     # number of replica of...
  selector:
    matchLabels:
      app: azure-vote-back        # ... app azure-vote-back

...
```

```
template:
  metadata:                         # pod definition for
    labels:
      app: azure-vote-back          # azure-vote-back
  spec:
    nodeSelector:                   # nodes to run the pod
      "kubernetes.io/os": linux
    containers:                     # containers to start
    - name: azure-vote-back         # name of the container
      image: redis                  # image (from docker hub)
      env:                          # environment variables for redis
      - name: ALLOW_EMPTY_PASSWORD
        value: "yes"
      resources:                    # resources to assign
        requests:
          cpu: 100m                 # 100 mili cpu (10% of cpu time)
          memory: 128Mi             # 128 MB
        limits:
          cpu: 250m
          memory: 256Mi
      ports:
      - containerPort: 6379         # port for accessing the container
        name: redis
```

```
---
apiVersion: v1
kind: Service                       # define a service
metadata:
  name: azure-vote-back             # name to access the service
spec:
  ports:
  - port: 6379                      # port for the outside
  selector:
    app: azure-vote-back            # app associated with the service
---
```

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-front
spec:

...                                 # spec omitted

---
apiVersion: v1
kind: Service
metadata:
  name: azure-vote-front
spec:
  type: LoadBalancer                # exposes the service using the
  ports:                            # cloud load balancer
  - port: 80                        # port for the outside
    targetPort: 80                  # port of the app/pods
  selector:
    app: azure-vote-front
```

GOAL

In the end of this lab you should be able to:

- Understand how to create a docker image
- Know how to launch a container in Azure Containers services

- Understand how to create a (simple) Kubernetes config file
- **Understand how to deploy a Kubernetes-based system**

Using Azure Kubernetes Service (1)

Create a resource group:

```
$ az group create --name scc2223-cluster-4204 --location westeurope
{
  "id": "/subscriptions/83abecdf-8b40-49a0-bcae-
b5fba4011353/resourceGroups/scc2223-cluster-4204",
  "location": "westeurope",
  "managedBy": null,
  "name": "scc2223-cluster-4204",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

USING AZURE KUBERNETES SERVICE (2)

Create a service principal:

```
$ az ad sp create-for-rbac --name http://scc2223-admin --role
Contributor
{
  "appId": "31c09123-e077-4f72-9fbe-f5b99cafbf12",
  "displayName": "http://scc2223-admin",
  "name": "31c09123-e077-4f72-9fbe-f5b99cafbf12",
  "password": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "tenant": "ae7e50a2-ed26-41f7-bd75-f49683f2433a"
}
```

Using Azure Kubernetes Service (3)

Create a cluster (appId and password should be the values returned from the service principal; for VM sizes and pricing check: https://docs.microsoft.com/en-us/azure/virtual-machines/sizes):

```
$ az aks create --resource-group scc2223-cluster-4204 --name my-scc2223-cluster-4204 --node-vm-size Standard_B2s --generate-ssh-keys --node-count 2 --service-principal appId_REPLACE --client-secret password_REPLACE
{
...
}
```

Using Azure Kubernetes Service (4)

Get credentials to access the Kubernetes cluster:

```
$ az aks get-credentials --resource-group scc2223-cluster-4204 --
name my-scc2223-cluster-4204

Merged "my-scc2223-cluster-4204" as current context in
/Users/nmp/.kube/config
```

After creating the Kubernetes cluster @ Azure, we will use standard Kubernetes commands.

Using Azure Kubernetes Service (5)

Deploy an application:

```
$ kubectl apply -f azure-vote.yaml
deployment.apps/azure-vote-back created
service/azure-vote-back created
deployment.apps/azure-vote-front created
service/azure-vote-front created
```

NOTE: If you don't have kubectl in your computer, you can install it running: az aks install-cli

Using Azure Kubernetes Service (6)

Check the application services:

```
$ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
azure-vote-back	ClusterIP	10.0.224.135	<none>	6379/TCP	18s
azure-vote-front	LoadBalancer	10.0.75.174	51.105.126.35	80:30891/TCP	18s
kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP	4m47s

The application running in the Kubernetes service is accessible using the External IP.
Example: http://51.105.126.35:80

Using Azure Kubernetes Service (7)

Check the application Pods:

```
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
azure-vote-back-798985f86b-xzrg8    1/1     Running   0          9m36s
azure-vote-front-84c8bf64fc-29254   1/1     Running   0          9m36s
```

Using Azure Kubernetes Service (8)

Stream the logs from one Pod:

```
$ kubectl logs -f azure-vote-front-84c8bf64fc-29254
... Messages
```

Using Azure Kubernetes Service (9)

Delete all objects (but persistent volumes) on Kubernetes.

```
$ kubectl delete deployments,services,pods --all
deployment.apps "azure-vote-back" deleted
deployment.apps "azure-vote-front" deleted
service "azure-vote-back" deleted
service "azure-vote-front" deleted
service "kubernetes" deleted
pod "azure-vote-back-798985f86b-xzrg8" deleted
pod "azure-vote-front-84c8bf64fc-29254" deleted
```

Using Azure Kubernetes Service (10)

Delete persistent volumes on Kubernetes (if you have created any).

```
$ kubectl delete pv --all
```

Using Azure Kubernetes Service (11)

Delete cluster

```
$ az group delete --resource-group scc2223-cluster-4204
```

If commands fail, do not forget to delete resources on Azure portal!

GUIDELINE: REDIS

When launching a container based on the default “redis” image, you have:

- Default port: 6379
- No password
- Simple HTTP - no TLS

GUIDELINE: MEDIA STORAGE AT PERSISTENT STORAGE

At Kubernetes, volumes have the lifetime of the Pod where they are mounted.

Kubernetes also supports persistent volumes that persist beyond the life of a Pod.

For information about persistent volumes at Azure, please check:

https://docs.microsoft.com/en-us/azure/aks/concepts-storage#persistent-volumes

GUIDELINE: MEDIA STORAGE AT PERSISTENT STORAGE (2)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: azure-managed-disk
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: azurefile
  resources:
    requests:
      storage: 1Gi
```

Allow to claim storage from the Kubernetes environment.

Can be mounted by as read-write by a single node.

Standard storage for azure files.

GUIDELINE: MEDIA STORAGE AT PERSISTENT STORAGE (3)

```
spec:
     containers:
     - name: ...
       image: ...
       volumeMounts:
       - mountPath: "/mnt/vol"
         name: mediavolume
       ...
     volumes:
     - name: mediavolume
       persistentVolumeClaim:
         claimName: azure-managed-disk
```

Define mount point and specify the volume name

Define volume and associate it to a persistentVolumeClaim

GUIDELINE: DATABASE

Suggestion:

- Create a service + pod with a **single instance** of the database you want to use.
- Check docker hub for info on how to launch a container with a single instance.
- Suggested database: mongodb, postgres.
- https://hub.docker.com/_/mongo
- https://hub.docker.com/_/postgres

GUIDELINE: AZURE FUNCTIONS

Suggestion:

- Replace Timer-triggered functions by Kubernetes cronjobs.
- https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

- Replace HTTP-triggered functions by a REST resource.