**Taking Sping-Boot apps to Azure**

1. Plugin to be used

<plugin>

<groupId>com.microsoft.azure</groupId>

<artifactId>azure-webapp-maven-plugin</artifactId>

<version>1.7.0</version>

</plugin>

2.Command to be used

mvn azure-webapp:config // To configure app

mvn azure-webapp:deploy // To deploy the app

az webapp log tail --name hello-world-rest-api- --resource-group hello-world-rest-api-rg // To view the log

docker run --detach --env MYSQL\_ROOT\_PASSWORD=dummypassword --env MYSQL\_USER=todos-user --env MYSQL\_PASSWORD=dummytodos --env MYSQL\_DATABASE=todos --name mysql --publish 3306:3306 mysql:5.7 //Running Mysql server Image

az webapp restart --name todo-web-application-mysql --resource-group todo-web-application-mysql-rg // To rollout the updates

az webapp log tail --name todo-web-application-mysql --resource-group todo-web-application-mysql-rg // To view updated logs

#Cloud Shell

mysql --host todo-web-application-mysql.database.azure.com --user todosuser@todo-web-application-in28minutes -p

#MySQLSh

mysqlsh todo-web-application-mysql.database.azure.com --user todosuser@todo-web-application

\sql

use todos;

select \* from todo;

mysqlsh todo-web-application-command-line.mysql.azure.com --user todosuser@todo-web-application-command-line -pdatabase.

az extension add --name db-up

az mysql up -g atabase -s todo-web-application-command-line -d todos -u todosuser -p YOUR@PASSWORD1

# Environment Variables

RDS\_HOSTNAME - todo-web-application-mysql.database.azure.com

todo-web-application-command-line.mysql.database.azure.com

RDS\_PORT - 3306

RDS\_DB\_NAME - todos

RDS\_USERNAME - todosuser@todo-web-application

todosuser@todo-web-application-command-line

RDS\_PASSWORD -

az webapp up --name frontend-full-stack -plan ServicePlan763a680f-840a-4de0 --resource-group rest-api-full-stack-rg --location westeurope

az webapp up --name frontend-full-stack -plan ServicePlan763a680f-840a-4de0 --resource-group rest-api-full-stack-front-end-rg --location westeurope

az group create --name container-resource-group --location westeurope

az appservice plan create --name container-service-plan --resource-group container-resource-group --sku P1v2 --is-linux

az webapp create --resource-group container-resource-group --plan container-service-plan --name todo-rest-api-h2-container --deployment-container-image-name vayalankatil/rest:1.0.0.RELEASE

docker run --detach --env MYSQL\_ROOT\_PASSWORD=dummypassword --env MYSQL\_USER=todos-user --env MYSQL\_PASSWORD=dummytodos --env MYSQL\_DATABASE=todos --name mysql --publish 3306:3306 mysql:5.7

docker stop 4ff5238b9464ed7050b89ef137e1b86bbf27c21aa74d5b62090f8fc3e86b4d17

docker rm 4ff5238b9464ed7050b89ef137e1b86bbf27c21aa74d5b62090f8fc3e86b4d17

docker run --detach --env MYSQL\_ROOT\_PASSWORD=dummypassword --env MYSQL\_USER=todos-user --env MYSQL\_PASSWORD=dummytodos --env MYSQL\_DATABASE=todos --name mysql --publish 3306:3306 mysql:5.7

docker run --name todos-api --publish 5000:5000 --link mysql:mysql Vayalankatil/mysql:1.0.0.RELEASE

docker-compose --version

docker container ls

docker container stop f25f59ebec84

docker container rm f25f59ebec84

docker-compose up

az group delete --name hello-world-rest-api-rg

az webapp restart -name hello-world-rest-api --resource-group hello-world-rest-api-rg

az provider list

az group delete --name container-resource-group --yes

az group delete --name hello-world-rest-api-rg

**Taking conatinerized app to cloud environment**

**Taking conatinerized app to cloud environment**

Step1: Az cli commands to be used in sequence

az group create --name container-resource-group --location westeurope

az appservice plan create --name container-service-plan --resource-group container-resource-group --sku P1v2 --is-linux

az webapp create --resource-group container-resource-group --plan container-service-plan --name rest-api-container-vayalankatil --deployment-container-image-name vayalankatil/rest-api-h3:latest

az webapp config appsettings set --resource-group container-resource-group --name rest-api-container-vayalankatil --settings WEBSITES\_PORT=80

CICD pipeline

Enable webhook

**Multicontainer application**

<!-- Docker -->

<plugin>

<groupId>com.spotify</groupId>

<artifactId>dockerfile-maven-plugin</artifactId>

<version>1.4.10</version>

<executions>

<execution>

<id>default</id>

<goals>

<goal>build</goal>

<!-- <goal>push</goal> -->

</goals>

</execution>

</executions>

<configuration>

<repository>vayalankatil/${project.artifactId}</repository>

<tag>latest</tag>

<skipDockerInfo>true</skipDockerInfo>

</configuration>

</plugin>

**Db Configuration**

spring.jpa.hibernate.ddl-auto=create

spring.datasource.url=jdbc:mysql://${RDS\_HOSTNAME:localhost}:${RDS\_PORT:3306}/${RDS\_DB\_NAME:erecord}?serverTimezone=UTC

spring.datasource.username=${RDS\_USERNAME:sudheesh}

spring.datasource.password=${RDS\_PASSWORD:dummy}

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

docker run -d -e MYSQL\_ROOT\_PASSWORD=dummy -e MYSQL\_DATABASE=erecord -e MYSQL\_USER=sudheesh -e MYSQL\_PASSWORD=dummy -p 3306:3306 mysql

create network (docker network create my-network ) && docker network ls

docker run -d -e MYSQL\_ROOT\_PASSWORD=dummy -e MYSQL\_DATABASE=erecord -e MYSQL\_USER=sudheesh -e MYSQL\_PASSWORD=dummy --name mysqldb --net my-network -p 3306:3306 mysql:latest

docker run --name rest-api-multicontainer --publish 8080:8080 --net my-network --link mysqldb:mysqldb vayalankatil/dbdemo-multicontainer:latest

**Taking multicontainer to azure**

az group create --name docker-compose-resource-group --location westeurope

az appservice plan create --name docker-compose-service-plan --sku P1v2 --resource-group docker-compose-resource-group --is-linux

az webapp create --resource-group docker-compose-resource-group --plan docker-compose-service-plan --name rest-api-mysql-docker-compose-vayalankatil --multicontainer-config-type compose --multicontainer-config-file docker-compose.yaml

[https://<app-name>.scm.azurewebsites.net/api/logs/docker](https://%3Capp-name%3E.scm.azurewebsites.net/api/logs/docker) //To view Container logs replace app-name with the application name

**Manage Azure Monitor using Java**

[**https://azure.microsoft.com/en-us/blog/manage-azure-monitor-using-java/**](https://azure.microsoft.com/en-us/blog/manage-azure-monitor-using-java/)

[**https://urldefense.com/v3/\_\_https://docs.microsoft.com/en-us/azure/developer/java/spring-framework/configure-spring-boot-java-applicationinsights\_\_;!!GqivPVa7Brio!MUxljjmYbfykNfFXwjPB\_VTWgU1SoF-dgvJH7fss7RDL-2qPiW89XP428iya2bL9QdMYAQfUEcS5Qg$**](https://urldefense.com/v3/__https://docs.microsoft.com/en-us/azure/developer/java/spring-framework/configure-spring-boot-java-applicationinsights__;!!GqivPVa7Brio!MUxljjmYbfykNfFXwjPB_VTWgU1SoF-dgvJH7fss7RDL-2qPiW89XP428iya2bL9QdMYAQfUEcS5Qg$)

[**https://urldefense.com/v3/\_\_https://docs.microsoft.com/en-us/azure/azure-monitor/app/java-agent\_\_;!!GqivPVa7Brio!MUxljjmYbfykNfFXwjPB\_VTWgU1SoF-dgvJH7fss7RDL-2qPiW89XP428iya2bL9QdMYAQdqJsj2Dw$**](https://urldefense.com/v3/__https://docs.microsoft.com/en-us/azure/azure-monitor/app/java-agent__;!!GqivPVa7Brio!MUxljjmYbfykNfFXwjPB_VTWgU1SoF-dgvJH7fss7RDL-2qPiW89XP428iya2bL9QdMYAQdqJsj2Dw$)

**Taking Spring-Boot Application to Kubernates Clusters**

Step 1. Create a resource Group

**az group create --name kubernetes-resource-group --location westeurope**

note down the resource group name which we need in step 3

Step 2: Create service principal

**az ad sp create-for-rbac --skip-assignment --name kubernetes-cluster-service-principal**

the above command will output something like below

{

"appId": "9dd3323b-b903-4fcb-8b3f-9faba4978849",

"displayName": "kubernetes-cluster-service-principal",

"name": "[http://kubernetes-cluster-service-principal](http://kubernetes-cluster-service-principal/)",

"password": "R\_ikjsFGp-tpqLp-iEg4hWrdwcZrE2BCPt",

"tenant": "59da348b-032d-4148-97ef-d5f7d92b53a3"

}

Note down the appid & password which you need in the next command where we actually create a cluster

Step3: create a cluster

**az aks create --name cluster-name --node-count 4 --enable-addons monitoring --resource-group kubernetes-resource-group --vm-set-type VirtualMachineScaleSets --load-balancer-sku standard --enable-cluster-autoscaler --min-count 1 --max-count 7 --generate-ssh-keys --service-principal <appid> --client-secret <password>**

give a your own valid cluster-name , replace the resourcegroup with the one you noted in step1

replace appid & password with the one you noted in step 2

min-count and max-count keep it 1 & 2 respectively as free tier won’t support more than that

Step4: Connect to the cluster created in step 3

**az aks get-credentials --resource-group kubernetes-resource-group --name cluster-name**

Once you successfully created and obtained the cluster you can verify the same by hitting the below command

**Kubectl version**

It shows both client & server information which means you have successfully connected to the cluster

Step5: deploy the containers to the pods using .yaml files

The git hub repository <https://github.com/SudheeshKrishnan/DayTrader> has all the deployment.yaml and service.yaml for all five microservices.

**Note** : If you want to make use of your own image for deployment please change the image tag to your own image tag name in deployment.yaml of each .yaml files in this repository after cloning the same

Go to the folder of each microservices and deploy the images of the same using below commands

**kubectl apply –f deployment.yaml**

**kubectl apply –f service.yaml**

Repeat the same for all five microservices & once done you can check the deployment

by hitting the command

**kubectl get pods –o wide**

You must see all the five pods in ready and up status, copy the the external Ip of the webapp pod hit it in the browser you must be abale to see the home page of DatraderwebApp

**Helm Charts and Service Mesh(Istios)**

helm install ./currency-exchange/ --name=currency-services

helm install ./currency-conversion/ --name=currency-services-1

helm install ./currency-conversion/ --name=currency-services-3 --debug --dry-run

helm history currency-services-1

helm upgrade currency-services-1 ./currency-conversion/

helm rollback currency-services-1 1

helm upgrade currency-services-1 ./currency-conversion/ --debug --dry-run

helm upgrade currency-services-1 ./currency-conversion/

helm history currency-services-1

Spring boot Dependency for Distribute Tracing for Spring Boot Microsevices with Istio and Jaegar

</dependency>

<groupId>io.opentracing.contrib</groupId>

<artifactId>opentracing-spring-cloud-starter</artifactId>

<version>0.1.7</version>

</dependency>

<dependency>

<groupId>io.jaegertracing</groupId>

<artifactId>jaeger-tracerresolver</artifactId>

<version>0.29.0</version>

</dependency>