June 2018 Products Tech Consulting Lab

Participant Instructions

Contents

Part 1 – Gathering Course Material and Running Sample Application 2
Prerequisites 2
Connecting to Lab Remote Desktop 2
Pull Course Materials from Innersource 2
Run the Application UI Locally 3
Part 2 – Creating and Running Local Docker Containers 4
Create a local MongoDB Instance using Docker 4
Create a Docker Container for Backing Services 4
Populate the Database with a Sample Ballot 5
Part 3 – Modifying the Application 6
Modify the Application UI and Run Locally to Test 6
Create a Docker Container for the UI 6
Part 4 – Deploying the Application on AWS Hosted Kubernetes 8
Push Containers to AWS Elastic Container Registry (ECR) 8
Deploy to Kubernetes 9
Extra Credit 1 – Populate Data in K8 10

Part 1 – Gathering Course Material and Running Sample Application

Prerequisites

Participation in this exercise requires access to the following software:

Software	***Purpose***
Remote Desktop Client	Connect to the lab virtual machine
Docker	Creating container images and running container instances
Git	Pulling course materials from innersource
Visual Studio Code	IDE for viewing and modifying application code. Integrated terminal will also be used for executing commands
Node.js & NPM	Supports both the UI and the backing services
Postman	Executing RESTful APIs and interacting with backing services
MongoDB Compass Community Edition	Exploring the Mongo database
AWS Command Line Tools	Interacting with the AWS ECR (Elastic Container Registry)
Kubectl tools	Interacting with Kubernetes cluster and deploying our application

This software, except for the remote desktop client, will be made available pre-installed in a AWS hosted virtual machine. Please ask your facilitator for connection details to your environment.

Connecting to Lab Remote Desktop

Open 'Remote Desktop Connection' from your PC

Enter the address provided by your lab facilitator

Username: `ubuntu`

Password: <provided by lab facilitator>

Pull Course Materials from Innersource

Open LXTerminal from the icon on the desktop

Ensure you are in the home directory by entering '`cd`' (without quotes) and hitting enter

Enter the following commands to pull resources from Innersource:

```
git clone ssh://git@innersource.accenture.com/prodtclab/lab-overview.git
```

```
git clone ssh://git@innersource.accenture.com/prodtclab/reference-app-voting-db.git
```

```
git clone ssh://git@innersource.accenture.com/prodtclab/reference-app-voting-services.git
```

```
git clone ssh://git@innersource.accenture.com/prodtclab/reference-app-voting-ui.git

git clone ssh://git@innersource.accenture.com/prodtclab/reference-app-voting-k8-deploy.git
```

Run the Application UI Locally

Open Visual Studio Code from the icon on the desktop

Choose 'File' then 'Open Folder'. Navigate to 'ubuntu' and then 'reference-app-voting-ui'. Select 'ok'

Open the integrated terminal by selecting 'View' then 'Integrated Terminal'

Install prerequisite node modules by entering '`npm install`' (this may take a few minutes)

Start the application by typing '`npm start dev`'

This should automatically open localhost:3000 and show the application UI. However, it isn't very interesting yet. We have no ballots in our voting application so we cannot do anything. Also, there is no back-end APIs supporting this UI or database for data persistence.

Note: Pause here to debrief with the rest of the class.

Part 2 – Creating and Running Local Docker Containers

Create a local MongoDB Instance using Docker

In Visual Studio Code choose 'File' then 'Open Folder'. Navigate to 'ubuntu' and then 'reference-app-voting-db'. Select 'ok'

Open the integrated terminal by selecting 'View' then 'Integrated Terminal'

This folder contains an extremely simple Dockerfile that points to the latest MongoDB container image.

Run the following command to build a container image based on MongoDB

```
docker build -t reference-app-voting-db .
```

Check that the image was successfully built by executing 'docker images'. You should see 'mongo' and 'reference-app-voting-db' listed in the 'REPOSITORY' column

Run a container instance of the reference-app-voting-db image by executing the following command

```
docker run -d -p 27017:27017 reference-app-voting-db
```

You should receive a Container ID such as '`22ddd7e…`' as a response

Confirm that the container is running by executing '`docker ps`'. The 'STATUS' column should show 'Up XX seconds'

Open 'MongoDB Compass Community' from the icon on the desktop. Connect with 'localhost' and '27017' as the port. You should see at least two databases (admin, local). This confirms that the container is up and running successfully and you (and eventually our application) can connect to it.

Create a Docker Container for Backing Services

In Visual Studio Code choose 'File' then 'Open Folder'. Navigate to 'ubuntu' and then 'reference-app-voting-services'. Select 'ok'

Open the integrated terminal by selecting 'View' then 'Integrated Terminal'

Run the following command to build a container image for the backing services

```
docker build -t reference-app-voting-services .
```

Check that the image was successfully built by executing 'docker images'. You should see 'node' and 'reference-app-voting-services' listed in the 'REPOSITORY' column

Identify your IP address. The easiest way is to look at the prompt in your terminal. If it is '`ubuntu@ip-172-31-8-141`' then your IP address is 172.31.8.141

Run a container instance of the reference-app-voting-services image by executing the following command

```
docker run -d -p 8080:3000 -e MONGO_DB_HOST=<YOUR IP ADDRESS> -e MONGO_DB_PORT=27017
reference-app-voting-services
```

You should receive a Container ID such as '`22ddd7e…`' as a response

Confirm that the container is running by executing '`docker ps`'. The 'STATUS' column should show 'Up XX seconds'

Open Postman from the icon on the desktop

Choose 'File', 'import'

Navigate to 'ubuntu' under 'Places' and then 'lab-overview' and then 'resources'

Open the 'TC-Lab-Collection.postman_collection.json' file

Expand the collection in the left pane and click click on the 'GET localhost:8080/ballots/' item

Click 'Send'. You should receive a response code of '200 OK' and the 'Body' tab should show a JSON entry with an empty 'Ballots' array. This indicates that the container is running and can receive incoming requests

Populate the Database with a Sample Ballot

Continue in Postman. Click on the 'POST localhost:8080/' in the left pane

Click on the 'Body' tab. A JSON representation of a voting ballot is included. Feel free to modify the ballot to your liking. Replace the ballot name, questions, and options as you prefer. You can use https://jsonlint.com to validate the structure of your input prior to submitting.

When you're comfortable with your ballot, click 'Send'. You should receive a response code of '201 Created' and the response body should include your ballot JSON with a 'ballotId' added (e.g. `ebe81827-5d50…`)

Click on the 'GET localhost:8080/ballots/' item in the left pane and click 'Send'. The body of the response should now include the '`ballotName`' and '`ballotId`' of your newly created ballot.

Note: Pause here to debrief with the rest of the class.

Part 3 – Modifying the Application

Modify the Application UI and Run Locally to Test

In Visual Studio Code choose 'File' then 'Open Folder'. Navigate to 'ubuntu' and then 'reference-app-voting-ui'. Select 'ok'

Open the integrated terminal by selecting 'View' then 'Integrated Terminal' (if it isn't already open)

Start the application by typing '`npm start dev`'

This should automatically open localhost:3000 and show the application UI. The drop down should now be populated with the ballot you submitted in Part 2.

Select your recently created ballot from the drop down.

Fill out the ballot and click 'Vote'. You should receive a message that says 'Vote Submitted'

Return to MongoDB Compass Community and click the refresh button on the upper left side

Click the 'reference-voting-db' and then 'reference-voting-votes'. You should see a document stored that contains the submitted ballot with populated 'selectedOption' fields

Return to Visual Studio Code. Expand the 'src' folder and double click on 'App.js'. Scroll to approximately line 40 and look for '`<h1>Reference Voting Application</h1>`'. Change the text in between the HTML tags to something unique (e.g. Jane's Voting App). Save your changes.

Return to Chrome. Your changes should be visible. Refresh or revisit localhost:3000 if necessary.

Create a Docker Container for the UI

Return to Visual Studio Code

In the terminal, stop the application by entering 'Ctrl+c'

Run the following command to build a container image for the UI

```
docker build -t reference-app-voting-ui .
```

This may take a few minutes.

Check that the image was successfully built by executing 'docker images'. You should see 'nginx' and 'reference-app-voting-ui' listed in the 'REPOSITORY' column

Run a container instance of the reference-app-voting-services image by executing the following command

```
docker run -d -p 80:80 reference-app-voting-ui
```

You should receive a Container ID such as '`22ddd7e…`' as a response

Confirm that the container is running by executing ‘docker ps’. The ‘STATUS’ column should show ‘Up XX seconds’

Confirm that the application is working as expecting by visiting localhost (NOTE: without specifying port 3000) in Chrome

Note: Pause here to debrief with the rest of the class.

Part 4 – Deploying the Application on AWS Hosted Kubernetes

Push Containers to AWS Elastic Container Registry (ECR)

This section can be executed from the terminal within Visual Studio Code or from an LXTerminal window

Get a docker login from ECR using the following command

```
aws ecr get-login --no-include-email --region us-east-2
```

Enter the and execute the output returned by the previous command to authenticate Docker with the AWS based container registry. (Note: Copy and paste will be helpful here)

```
Docker login -u AWS – p <key e.g. eyJwYx……>
```

Tag your database image to prepare it to be pushed to the AWS based container registry by executing the following command. NOTE: notice that you should replace the '<your name>' with your name so your container image is unique in the repository

```
docker tag reference-app-voting-db:latest 886700761579.dkr.ecr.us-east-
2.amazonaws.com/reference-app-voting-db:<your name>
```

Confirm that your image was tagged successfully by running 'docker images'. Notice that it will share the same image ID with your original image 'reference-app-voting-db'

Push your image to ECR by issuing the following command. NOTE: notice that you should replace the '<name>' with your name in the same was as the previous step.

```
docker push 886700761579.dkr.ecr.us-east-2.amazonaws.com/reference-app-voting-db:<your
name>
```

Execute the following command to confirm that the image tagged with your name has successfully been pushed to ECR

```
aws ecr list-images --repository-name reference-app-voting-db --output json
```

You should see an entry with your name as well as entries for any class members completing this step prior to you.

Repeat the steps for the 'reference-app-voting-services' and 'reference-app-voting-ui' images using the following commands.

```
docker tag reference-app-voting-services:latest 886700761579.dkr.ecr.us-east-
2.amazonaws.com/reference-app-voting-services:<your name>


docker push 886700761579.dkr.ecr.us-east-2.amazonaws.com/reference-app-voting-
services:<your name>


docker tag reference-app-voting-ui:latest 886700761579.dkr.ecr.us-east-
2.amazonaws.com/reference-app-voting-ui:<your name>
```

```
docker push 886700761579.dkr.ecr.us-east-2.amazonaws.com/reference-app-voting-ui:<your name>
```

Deploy to Kubernetes

Our docker images should now be available to fuse beyond just our individual remote desktops. This next step will prepare a YAML file with the details necessary for Kubernetes to pull the image from ECR and run the applications.

In Visual Studio Code choose 'File' then 'Open Folder'. Navigate to 'ubuntu' and then 'reference-app-voting-k8-deploy'. Select 'ok'

Open the integrated terminal by selecting 'View' then 'Integrated Terminal' (if it isn't open already)

Enter the following command to set an environment variable for kubectl to find its configuration. kubectl is a command line interface for running commands against Kubernetes clusters.

```
export KUBECONFIG=/home/ubuntu/kubeconfig
```

Run '`kubectl get nodes`' to ensure the configuration is working and you can connect to the Kubernetes cluster. This should return 7 nodes.

Double click on 'reference-app-voting-deploy-and-service.yaml' to open it the editor. There are four instances of '<INSERT NAMESPACE HERE>' that need to be replaced with a unique namespace. Namespaces are a way to divide cluster resources between multiple users or purposes. For example, this could be used for logical environments (e.g. 'DEV', 'QA', 'STAGING', 'PROD') or users (e.g., 'JANE', 'JILL', 'SAM').

Replace '<INSERT NAMESPACE HERE>' with something similar to '`reference-app-voting-jane-smith-ns`' replacing '`jane-smith`' with your name. (Note: Lines 4, 6, 12, 48)

We need to indicate which container images to use in our deployment. Replace the '<UPDATE THIS TAG>' on the 'image:' attributes with the tag used in the previous section. There are three instances in total. (Note: Lines 25, 29, 40).

Save your work.

Run the following command to deploy your application onto the Kubernetes cluster

```
kubectl create -f reference-app-voting-deploy-and-service.yaml
```

Three resources should be successfully created (namespace, deployment.apps, service).

Explore the resources that were created by using the following command.

```
kubectl --namespace <YOUR NAMESPACE> get all
```

Within a couple minutes you should be able to find your application on a public URL. To find this URL use the following command.

```
kubectl --namespace <YOUR NAMESPACE> describe service reference-app-voting-service
```

The ‘LoadBalancer Ingress’ field will indicate the public URL where your application is available. (e.g. aa54a49b764lclle89c3402ffb6c0882-80366700.us-east-2.elb.amazonaws.com). Attempt to open this URL in a web browser. (Note: it can take a few minutes to come available)

Extra Credit 1 – Populate Data in K8

Use postman to publish a ballot to your AWS/Kubernetes hosted application. Does it work?