What The Hack – Proctor's Guide

Kubernetes as Infrastructure

Challenges:

- Make sure that you have joined the Teams group for this track. The first person on your team at your table should create a new channel in this Team with your team name.
- Install the recommended tool-set:
  - Windows Subsystem for Linux
  - Azure CLI
    - Update to the latest
    - Must be at least version 2.0.42
    - **NOTE:** If you're running into issues running Azure CLI command on Windows, Disable Global Protect (VPN)
  - Visual Studio Code
- **Note:** You can start the next challenge even if this one is still running by using the the Azure Cloud Shell.
- **Tip:** You can complete almost all of the challenges with the Azure Cloud Shell! But be a good cloud architect and make sure you have experience installing the tools locally.

Challenge Set 1: Got Containers?

Challenges:

- Deploy build agent VM with Linux + Docker using provided ARM Template & parameters file in the “Files” tab of the Team’s General channel.
- Run the Fab Medical application locally on the VM & verify access
    - Each part of the app (api & web) runs independently.
    - Build the API app by navigating to the **content-api** folder and run “**npm install**”.
    - To start the app, run “**nodejs ./server.js &**”
    - Verify the API app runs by hitting its URL with one of the three function names. Eg: “**http://localhost:3000/speakers**”
    - Repeat for the steps above for the content-web app, but verify it’s available via a browser on the Internet!
    - **NOTE:** The content-web app expects an environment variable named “**CONTENT_API_URL**” that points to the API app’s URL.
- Create a Dockerfile for the content-api app that will:
    - Create a container based on the node:8 container image
    - Build the Node application like you did above (**Hint:** npm install)
    - Exposes the needed port
    - Starts the node application
- Create a Dockerfile for the content-web app that will:
    - Do the same as the Dockerfile for the content-api
    - Also sets the environment variable value as above
- Build Docker images for both content-api & content-web
- Run both containers you just built and verify that it is working.
    - **Hint:** Run the containers in ‘detached’ mode so that they run in the background.
    - **NOTE:** The containers need to run in the same network to talk to each other.
        - Create a Docker network named “fabmedical”
        - Run each container using the "fabmedical” network
        - **Hint:** Each container you run needs to have a “name” on the fabmedical network and this is how you access it from other containers on that network.
        - **Hint:** You can run your containers in “detached” mode so that the running container does NOT block your command prompt.

Challenge Set 2: Azure Container Registry

Challenges:

- Deploy an Azure Container Registry (ACR)
- Ensure your ACR has proper permissions and credentials set up
- Login to your ACR
- Push your Docker container(s) to the ACR
- List all images in your ACR

Challenge Set 3: Introduction to Kubernetes

Challenges:

- Install the Kubernetes command line tool (kubectl).
    - **Hint**: This can be done easily with the Azure CLI
- Create a new, multi-node AKS cluster with **RBAC disabled.**
    - Use a single core DS1v2 machine for your worker nodes.
    - Use the latest version of Kubernetes supported by AKS.
- Use kubectl to prove that the cluster is a multi-node cluster and is working
- Bring up the Kubernetes dashboard in your browser
    - **Hint**: Again, the Azure CLI makes this very easy.

Challenge Set 4: Your First Deployment

Challenges:

- **NOTE:** If you are not able to deploy your containers to the Azure Container Registry, we have staged the FabMedical apps on Docker Hub at these locations:
    - **API app**: dta2018hack/content-api
    - **Web app**: dta2018hack/content-web
- Deploy the **API app** from the command line using kubectl and YAML files
    - Number of pods: 1
    - Service: Internal
    - Port and Target Port: 3001
    - CPU: 0.5
    - Memory: 128MB
- We have not exposed the API app to the external world. Therefore, to test it you need to:
    - Figure out how to get a bash shell on the API app pod just deployed.
    - Curl the url of the "/speakers" end point.
- You should get a huge json document in response.
- Deploy the **Web app** from the command line using kubectl and YAML files
    - **NOTE**: Sample YAML files to get you started can be found in the Files section of the General channel in Teams.
    - **NOTE**: The Web app expects to have an environment variable pointing to the URL of the API app named:
        - **CONTENT_API_URL**
    - Create a deployment yaml file for the Web app using the specs from the API app, except for:
        - Port and Target Port: 3000
    - Create a service yaml file to go with the deployment
        - **Hint**: Not all "types" of Services are exposed to the outside world
    - **NOTE:** Applying your YAML files with kubectl can be done over and over as you update the YAML file. Only the delta will be changed.
    - **NOTE**: The Kubernetes documentation site is your friend. The full YAML specs can be found there: https://kubernetes.io/docs
- Find out the External IP that was assigned to your service. You can use kubectl or the dashboard for this.
- Test the application by browsing to the Web app's external IP and port and seeing the front page come up.
    - Ensure that you see a list of both speakers and sessions on their respective pages.
    - If you don't see the lists, then the web app is not able to communicate with the API app.

Challenge Set 5: Scale and High Availability

Challenges:

- Scale the Web app to 2 instances
    - This should be done by modifying the YAML file for the Web app and re-deploying it.
- Scale the API app to 4 instances
    - This should be done through the Kubernetes dashboard.
- Watch the ReplicaSets and Pods pages in the dashboard to see how they change.
    - You will find an error occurs because the cluster does not have enough resources to support that many instances.
    - There are two ways to fix this: increase the size of your cluster or decrease the resources needed by the deployments.
- To fully deploy the application, you will need 4 instances of the API app running and 2 instances of the Web app.
    - **Hint:** If you fixed the issue above correctly, you should be able to do this with the resources of your original cluster.
- When your cluster is fully deployed, browse to the "/stats.html" page of the web application.
    - Keep refreshing to see the API app's host name keep changing between the deployed instances.
- Scale the API app back down to 1, and immediately keep refreshing the "/stats.html" page.
    - You will notice that without any downtime it now directs traffic only to the single instance left.

Challenge Set 6: Deploy MongoDB to AKS

Challenges:

- Deploy a MongoDB container in a pod for v2 of the FabMedical app
- **Hint:** Check out the Docker Hub container registry and see what you can find.
- Confirm it is running with:
    - **kubectl exec -it <mongo pod name> -- mongo "--version"**

Challenge Set 7: Updates and Rollbacks

Challenges:

- We have staged an updated version of the app on Docker Hub with id and version:
  - **dta2018hack/content-web:v2**
  - **dta2018hack/content-api:v2**
- For version two, you will also need an initializer container available on Docker Hub at:
  - **dta2018hack/content-init**
  - Use the content-init "Job" yaml provided to run the initialization of MongoDB for our new version of the app.
- Perform a rolling update of the Web app on your cluster to the new version two of content-web
  - You'll be doing this from the command-line with a kubectl command (remember, Kubernetes docs are your friend!)
  - In the Kubernetes dashboard on the Pods page, you should be able to see new pods with the new version come online and the old pods terminate
    - You can also do this by listing the pods with kubectl.
  - At the same time, hit the front page to see when you're on the new version by refreshing constantly until you see the conference dates updated to 2019.
- Now roll back this update.
  - Again, this is done from the command-line using a (different) kubectl command.
  - Confirm that we are back to the original version of the app by checking that the conference dates are back to 2017.
- Perform the update again, this time using the blue/green deployment methodology.
  - You will need a separate deployment file using different tags.
  - Cut over is done by modifying the app's service to point to this new deployment.