DTA What The Hack – Proctor Guide

Kubernetes as Infrastructure

Challenge Set 1: Introduction

Lecture:

- Introduction to Kubernetes
- Introduction to the Azure Kubernetes Service

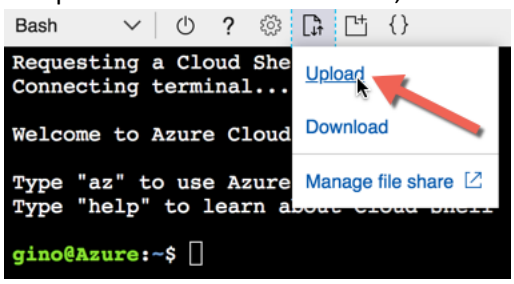
Challenges:

- Make sure that you have joined the Teams group for this track. Join via this code:
    - **vrgbd4d**
- Install the recommended tool-set:
    - Windows Subsystem for Linux
    - Azure CLI
        - Update to the latest
        - Must be at least version 2.0.42
    - Visual Studio Code
- 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 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.

Proctor Notes & Guidelines

- This first hour is the most critical to make sure all teams make it through and aren't left behind.
    - Make sure that ALL team members have joined the Teams group for this track. Join via this code:
        - **vrgbd4d**
    - Make sure that ALL team members are set up with the tools.
    - Encourage teams to use the CLI and WSL to make things easier.
    - **NOTE**: If someone has to install the tools, make sure they first create the cluster in the portal and then install the tools in parallel.

        - To upload files to the Cloud Shell, use the upload button:



        - Show them BOTH of these if they are unfamiliar with how to navigate to them.
- Remind teams that kubectl can be installed through the CLI, but don't give away the answer:
    - **az aks install-cli**
- All teams should have an AKS cluster stood up fairly quickly
    - Version shouldn't matter, but good to make sure it is the latest
    - Keep it simple: Basic networking, RBAC disabled, let it create a new service principal.
    - **NOTE**: Sometimes during the validation step when creating a new cluster, it will fail because it cannot find the new Service Principal. This is a timing issue. Click the Previous button to go back one page and then Forward to redo the validation.
    - They can use the CLI for this with a simple command to make a 3-node cluster:
        - **az aks create --resource-group myAKSCluster --name myAKSCluster --node-count 3 --enable-addons monitoring --generate-ssh-keys**
    - Docs to install AKS:
        - Portal: https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough-portal
        - CLI: https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough
- Have the teams show you the running cluster with:
    - **kubectl get nodes**
    - It should have more than 1 node.
    - **NOTE**: They will need to learn how to connect kubectl to their cluster using "az aks get-credentials".
- The Kubernetes dashboard can be brought up with the CLI easily:
    - **az aks browse --name myAKSCluster --resource-group myAKSCluster**

Challenge Set 2: Your First Deployment

Lecture:

- Review the basic components of Kubernetes
  - The physical architecture
  - Pods
  - Deployments
  - Services
    - Talk about externally facing services using a load balancer in AKS this is an Azure LB
- Introduce the FabMedical app and its components
  - It's an app that manages medical conferences and speakers at the events
  - Written in node.js

Challenges:

- 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** through the dashboard using these settings
  - 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.

Proctor Notes & Guidelines

- The apps are staged on Docker Hub so they should not require any logging into a docker registry and thus docker need not be installed on their machines.
- Students need to figure out on their own where in the dashboard you create an app.
  - Click "+ CREATE" button on the top right
  - Use the "CREATE AN APP" tab
  - Advanced Settings will be needed for CPU and Memory
- To verify that the API app is correct deployed the students need to:
  - Figure out the name of the pod the API app was deployed to, eg: content-api-23kkdj
  - Then use a kubectl command like this to get a bash shell:
    - **kubectl exec -it content-api-23kkdj -- /bin/bash**
  - To verify the API app is working curl the /speakers endpoint:
    - **curl http://localhost:3001/speakers**
  - They should see a huge JSON document printed to the screen.

Challenge Set 3: Your Second Deployment

Lecture:

- Review the structure of YAML files for Deployments and Services
- Introduce the web app for FabMedical and its dependencies on the API app.

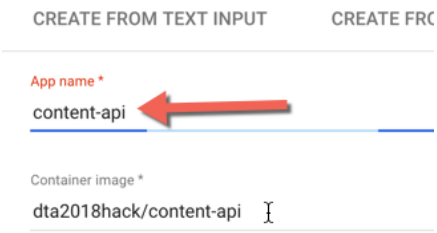
Challenges:

- We have staged the FabMedical apps on Docker Hub at these locations:
    - **API app**: dta2018hack/content-api
    - **Web app**: dta2018hack/content-web
- 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.

Proctor Notes & Guidelines

- The app is staged on Docker Hub so they should not require any logging into a docker registry and thus docker need not be installed on their machines.
- Students will need to find extra settings to add to their template YAML files. They should make use of the Kubernetes docs in addition to whatever else they can find on the web.
    - **NOTE**: Fully fleshed out YAML files are available to proctors in the Files area of the Proctor's Infra channel on Teams.
    - **NOTE**: If they go to "Edit Deployment" on the API app deployed from the dashboard, they will see the full YAML for that deployment. This hint should be saved until they get desperate.
- In the Deployment YAML, they'll need these settings in the spec section:

- containers.resources.requests.cpu: 0.5 (or 500m)
- containers.resources.requests.memory: 128Mi
- containers.ports.containerPort: 3001
- containers.env.name: CONTENT_API_URL
- containers.env.value: http://content-api:3001
  - The value “content-api” in the URL must be whatever was used as the name of the app during deployment on this screen:



- In the Service YAML, they need to figure out that the type should be changed to “LoadBalancer”.
- When the service is deployed it will take some time for an External IP to be assigned.
  - In the dashboard, go to the Services page and look at the “External Endpoints” column
  - Issue the following kubectl command and look in the “EXTERNAL-IP” column
    - **kubectl get services**
  - You will see “<pending>” if the IP hasn’t yet been assigned.

Challenge Set 4: Scale and High Availability

Lecture:

- Review scaling in Kubernetes, scaling pods vs. adding nodes to the cluster
- Show an example of scaling and how Kubernetes treats the replicas number as a desired state.
- Discuss scaling out the cluster and how this is a manual process (for now!)
- Review HA in Kubernetes, how it is pod-centric.

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.

Proctor Notes & Guidelines

- In the YAML file, they will have to update the "spec.replicas" value.
- The error they will encounter is that there aren't enough CPUs in the cluster to support the number of replicas they want to scale to.
- The two fixes are:
    - Use the Azure portal to add more nodes to the AKS cluster.
    - Change the deployment and reduce the needed CPU number from "0.5" to "0.125" (500m to 125m).
        - This is the preferred solution.
- **NOTE**: In case they do NOT get an error and are able to scale up, check how many nodes they have in their cluster and the size of the node VMs. Over provisioned clusters will not fail.
    - If a team doesn't get a failure, just have them double the number of Web and API app instances.

Challenge Set 5: Updates and Rollbacks

Lecture:

- Review deployments
- Explain Kubernetes rolling updates mechanism as well as deployment rollbacks.
- Mention Blue/Green deployments (similar to deployment slots) for apps that can't handle rolling updates.

Challenges:

- We have staged an updated version of the Web app on Docker Hub with id and version:
  - **dta2018hack/content-web:v2**
- Perform a rolling update on your cluster to this new version
  - 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.

Proctor Notes & Guidelines

- They will need to use the "kubectl set image" command to perform the rolling update:
  - **kubectl set image deployment/content-web content-web=dta2018hack/content-web:v2**
    - Where "deployment/content-web" is the name of the deployment used
  - "kubectl set image" takes the name of the deployment and the new image to update to.
- **kubectl get pods –** Running this will show all the pods getting updated and terminated.
- Rollbacks are performed with:
  - **kubectl rollout undo deployment/content-web**
  - This will roll-back the last update to the "content-web" deployment.
- Blue/Green deployments are described here:
  - https://www.ianlewis.org/en/bluegreen-deployments-kubernetes
  - Basically, they will create a separate deployment YAML with different tags and deploy it.
    - **NOTE:** Look in Teams for an updated example YAML showing the changes.
  - When the new pods are ready to go, they will update the service YAML to point to the new tags.