# DTA What The Hack – Challenges

Kubernetes as Infrastructure

## Challenge Set 1: Introduction

### 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.

# Challenge Set 2: Your First Deployment

## 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.

# Challenge Set 3: Your Second Deployment

## 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.

# Challenge Set 4: 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 5: Updates and Rollbacks

## 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.