Deploy Application to Azure Kubernetes Service

In this lab we will create an Azure Kubernetes Service (AKS) cluster using the Azure CLI. We will deploy an application to the cluster, it will be a a multi-container application that includes a web front end and a Redis Cache instance. We will then see monitor the health of the cluster and pods that run the sample application.

Lab Tasks:

* Task 1: Setup the environment and create an Azure resource group
* Task 2: Create AKS cluster
* Task 3: Connect to the cluster
* Task 4: Create sample application and deploy it to the AKS cluster
* Task 5: Test the running application
* Task 6: Monitor application and cluster health

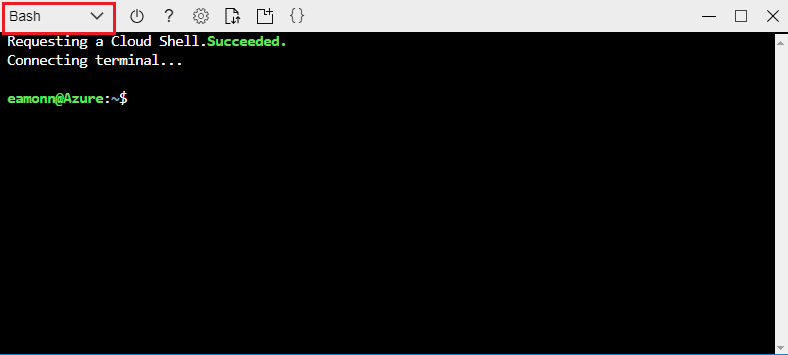
Estimated Lab Time:

* approx. 45 minutes

Task 1: Setup the environment and create an Azure resource group

We will use Azure Cloud Shell for ease of use and because many of the elements we need such as Azure CLI and the Kubernetes command line client are pre-installed, ready for use.

1. Open Azure Cloud Shell by going to https://shell.azure.com, or using the Azure Portal and selecting **Bash** as the environment option.



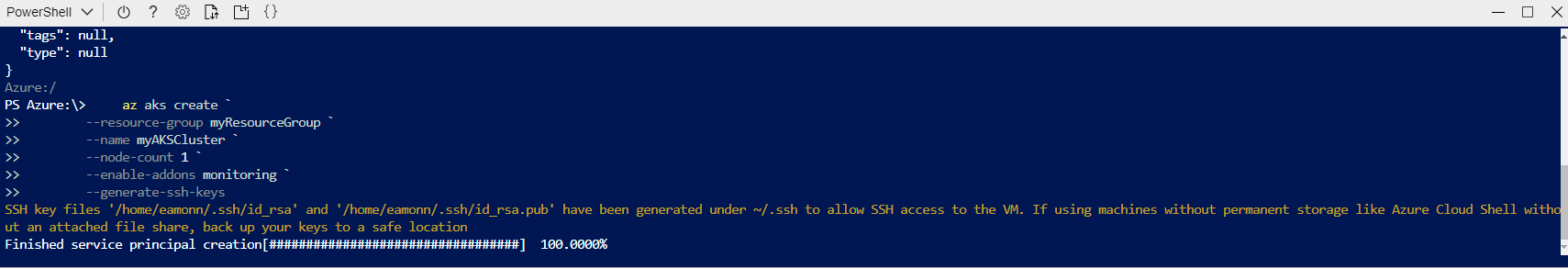
1. Create an Azure resource group by running the following command:

az group create --name myResourceGroup --location < datacenter nearest you >

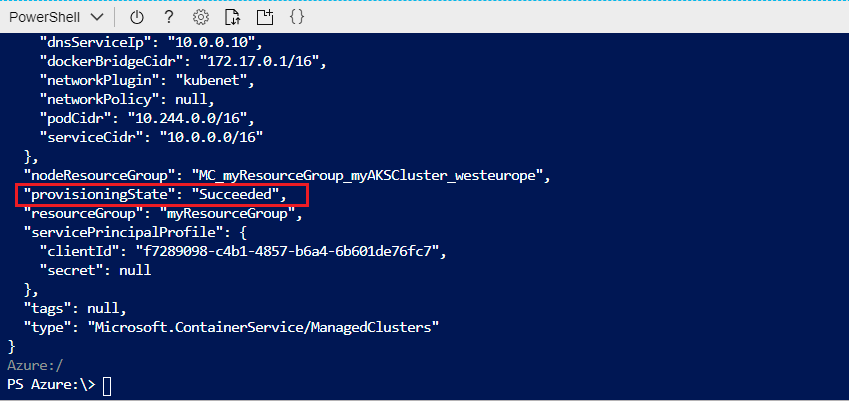
Task 2: Create AKS cluster

1. Create an AKS cluster by running the following command:

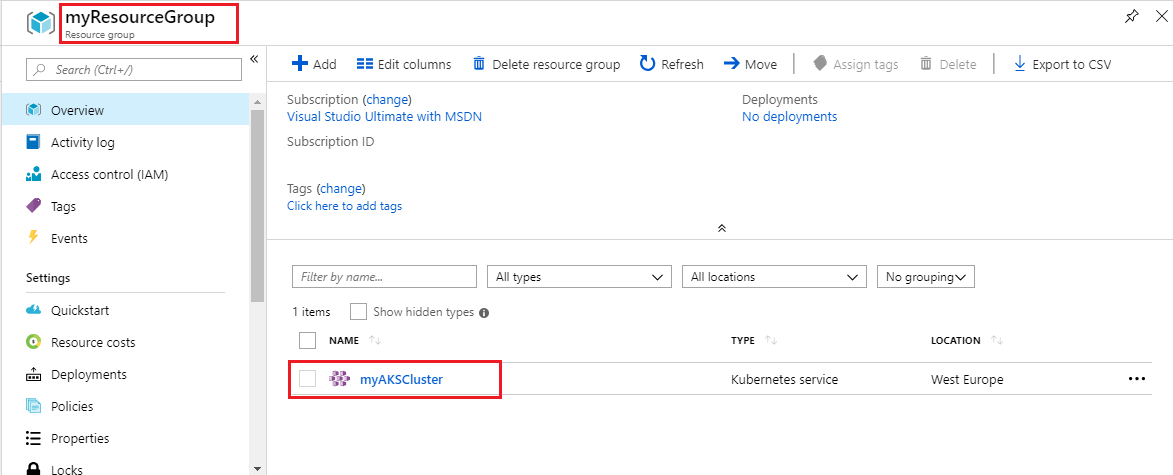
az aks create --resource-group myResourceGroup --name myAKSCluster --node-count 1 --enable-addons monitoring --generate-ssh-keys



1. It can take up to 5 minutes to deploy the AKS cluster. After a few minutes, the command completes and returns JSON-formatted information about the cluster.



1. You can also go into the Azure Portal and vew the created Azure Kubernetes cluster



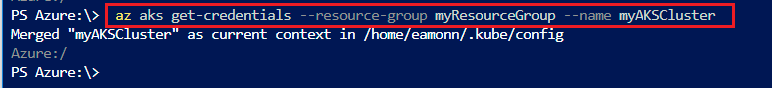
Task 3: Connect to the cluster

1. To manage a Kubernetes cluster, you use kubectl, the Kubernetes command-line client. If you use Azure Cloud Shell, kubectl is already installed. To install kubectl locally, use the following command:

az aks install-cli

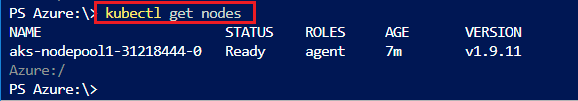
1. To configure kubectl to connect to your Kubernetes cluster, use the az aks get-credentials command. This command downloads credentials and configures the Kubernetes CLI to use them:

az aks get-credentials **--resource-group** myResourceGroup **--name** myAKSCluster



1. Verify the connection to your cluster by running the following command. Make sure that the status of the node is Ready:

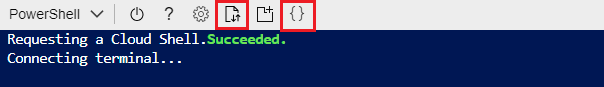
kubectl get nodes



Task 4: Create sample application and deploy it to the AKS cluster

1. We define our application in a yaml file and we will apply thsi application definition to the AKS cluster using an already created file at <https://raw.githubusercontent.com/Microsoft/PartsUnlimited/master/Labfiles/AZ-400T05_Implementing_Application_Infrastructure/M03/azure-vote.yaml>. Open the file and have a look at its contents to get a feeling for what it is doing.
   * **Note** the following items in the *.yaml* definition:
     + The presence of different **Kind** values to designate *deployments* and *services*
     + The delimeter --- needs to be present to separate the *deployments* and *services* When copying the code ensure it is present in your yaml file. This demarcates the *services* and *deployments*.
     + The definition of container resources for ports, cpu and memory limits, container images to use and other configuration values

**Note**: In general if you are using the **Azure Cloud Shell**, you could also create this or any other configuration file needed by using the **vi** or **nano** editors, in either the bash or PowerShell terminals. Another option is to use the built in Editor in the **Azure Cloud Shell** by clicking the editor icon in the **Azure Cloud Shell** taskbar. A final option is to create the file locally, and the use the Upload/Download file option within the Azure Cloud Shell, to upload the file.



1. Deploy the application by running the following command in **Azure Cloud Shell**:

kubectl apply **-f** https://raw.githubusercontent.com/Microsoft/PartsUnlimited/master/Labfiles/AZ-400T05\_Implementing\_Application\_Infrastructure/M03/azure-vote.yaml

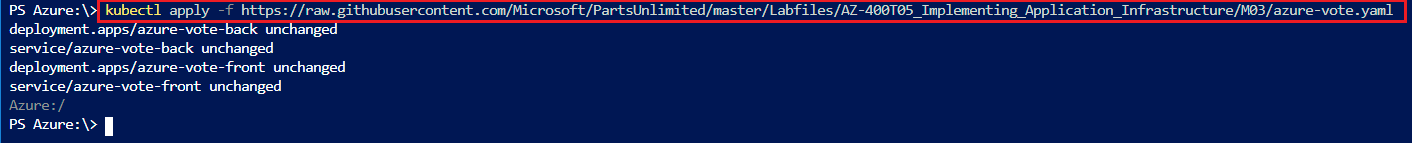
You should receive output showing the Deployments and Services were created successfully after it runs as per the below.

deployment "azure-vote-back" created

service "azure-vote-back" created

deployment "azure-vote-front" created

service "azure-vote-front" created



Task 5: Test the running application

1. When the application runs, a Kubernetes service exposes the application front end to the internet. This process can take a few minutes to complete. To monitor progress run the command

kubectl get service azure-vote-front **--watch**

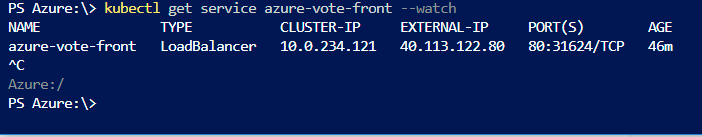
1. Initially the **EXTERNAL-IP** for the azure-vote-front service is shown as pending.

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

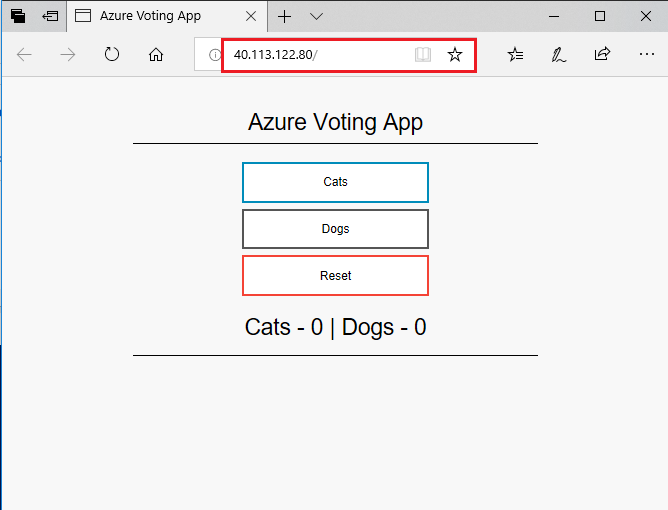
azure-vote-front LoadBalancer 10.0.37.27 < pending > 80:30572/TCP 6s

1. When the **EXTERNAL-IP** address changes from pending to an actual public IP address, use **CTRL-C** to stop the kubectl watch process. The following example output shows a valid public IP address assigned to the service:

azure-vote-front LoadBalancer 10.0.37.27 52.179.23.131 80:30572/TCP 2m



1. To see the Azure Vote app in action, open a web browser to the external IP address of your service. You can click the Cats and Dogs and Reset buttons to verify functionality and generate some workload.



**Check Pods**

kubectl get pods

**Replicate vote-Front**

kubectl scale --replicas=3 deployment/azure-vote-front

**Check with below command:**

kubectl get pods

**AutoScale Rule**

kubectl autoscale deployment azure-vote-front --cpu-percent=50 --min=3 --max=10

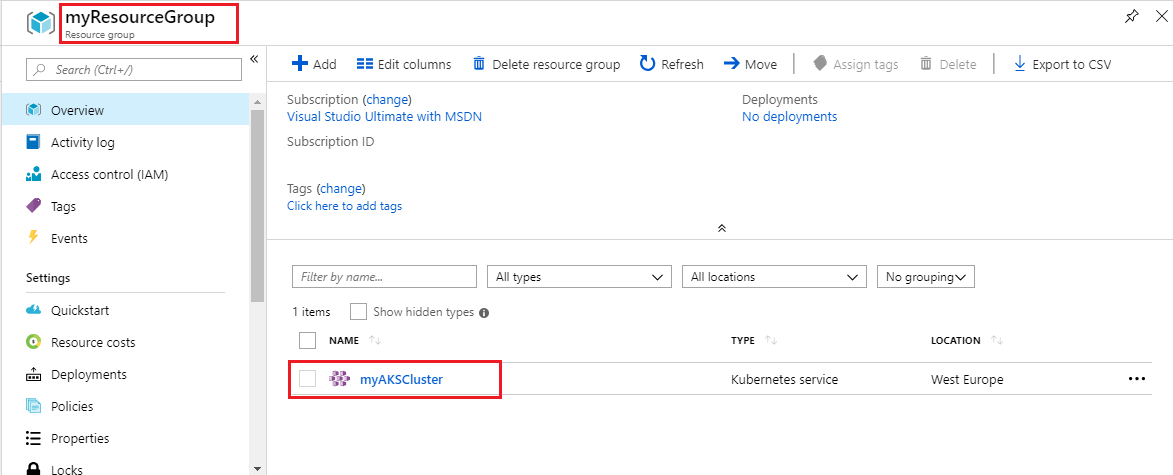
**Manually Scale of nodes**

az aks scale --resource-group myResourceGroup --name myAKSCluster --node-count 2

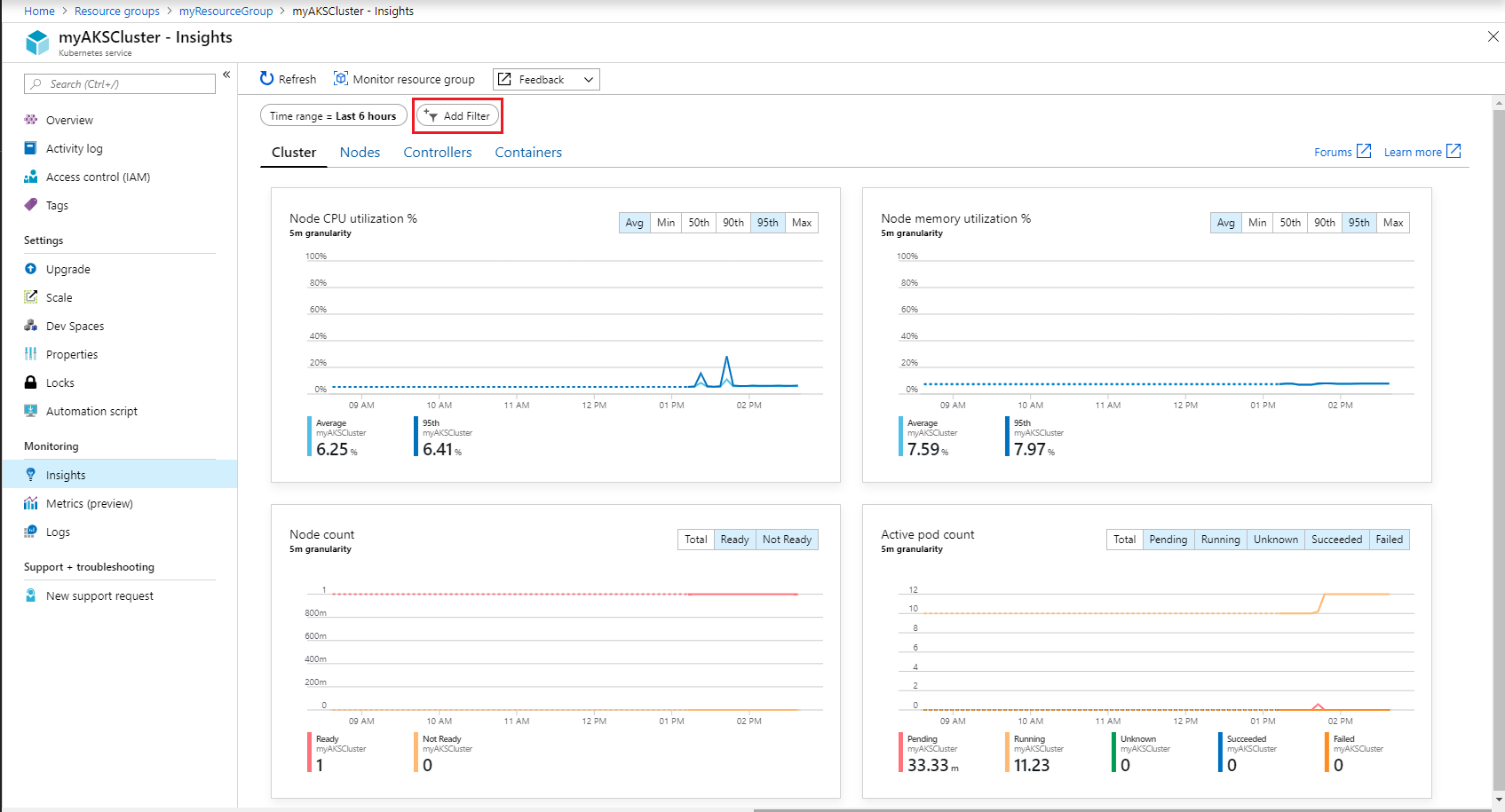
Task 6: Monitor application and cluster health

When the AKS cluster was created, Azure Monitor for containers was enabled to capture health metrics for both the cluster nodes and pods. These health metrics are available in the Azure portal. To see current status, uptime, and resource usage for the Azure Vote pods, complete the following steps in the Azure portal:

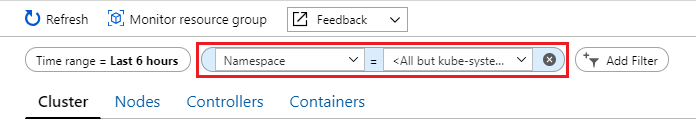
1. Open the azure portal at [https://portal.azure.com](https://portal.azure.com/)
2. Select your resource group, such as **myResourceGroup**, then select your AKS cluster, such as **myAKSCluster**.



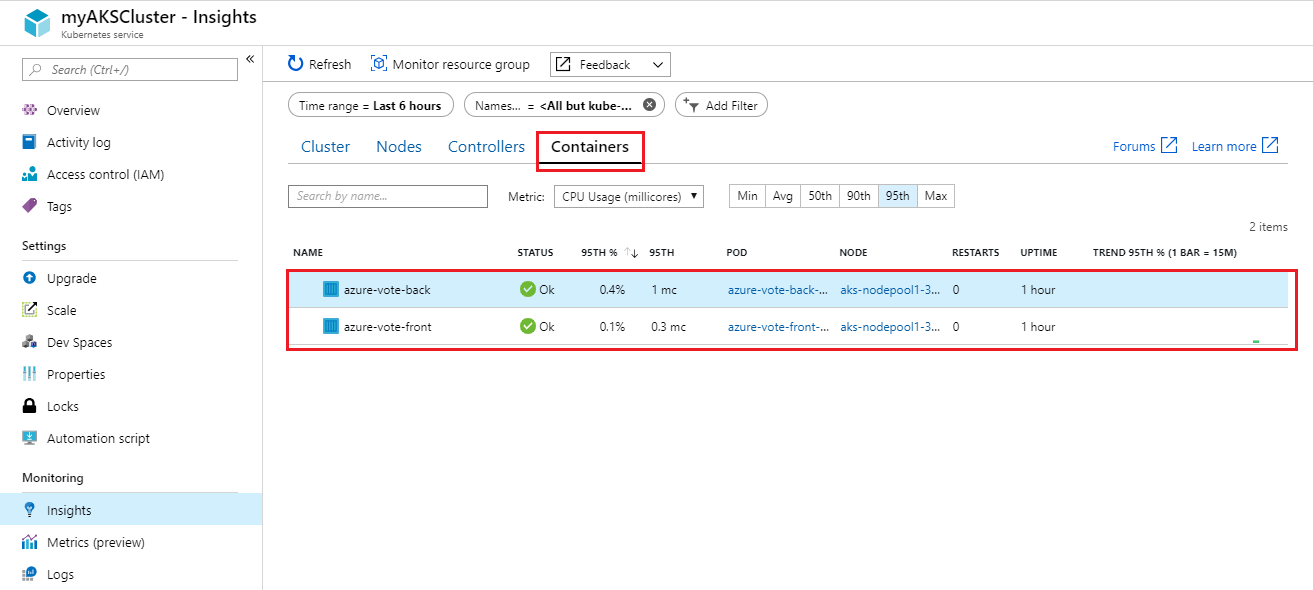
1. Under **Monitoring** on the left-hand side, choose **Insights** and then across the top, choose to **+ Add Filter**



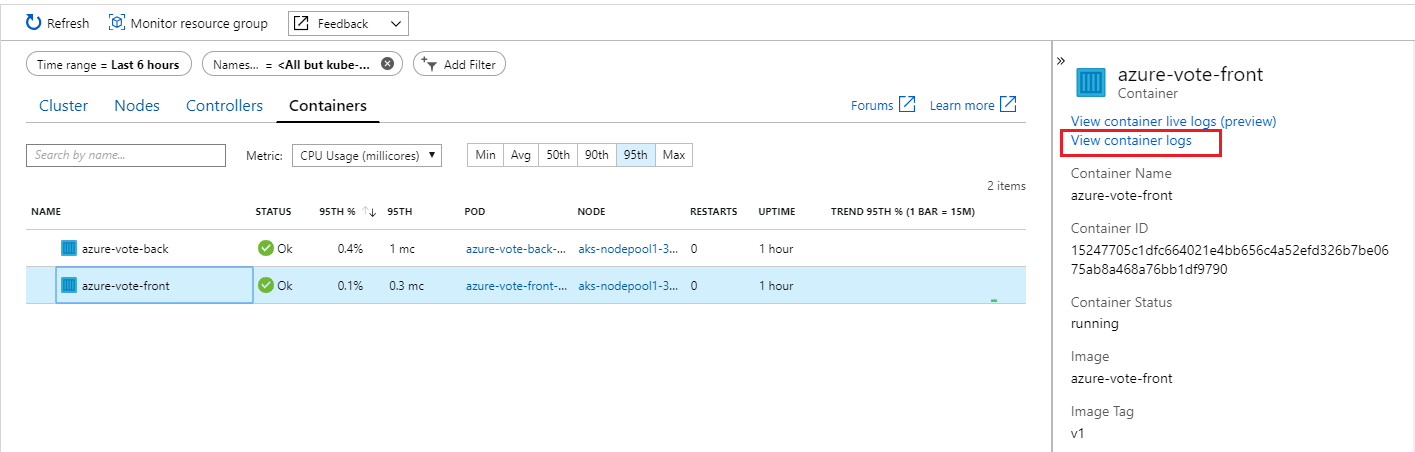
1. Select **Namespace** as the property, then choose **< All but kube-system >**



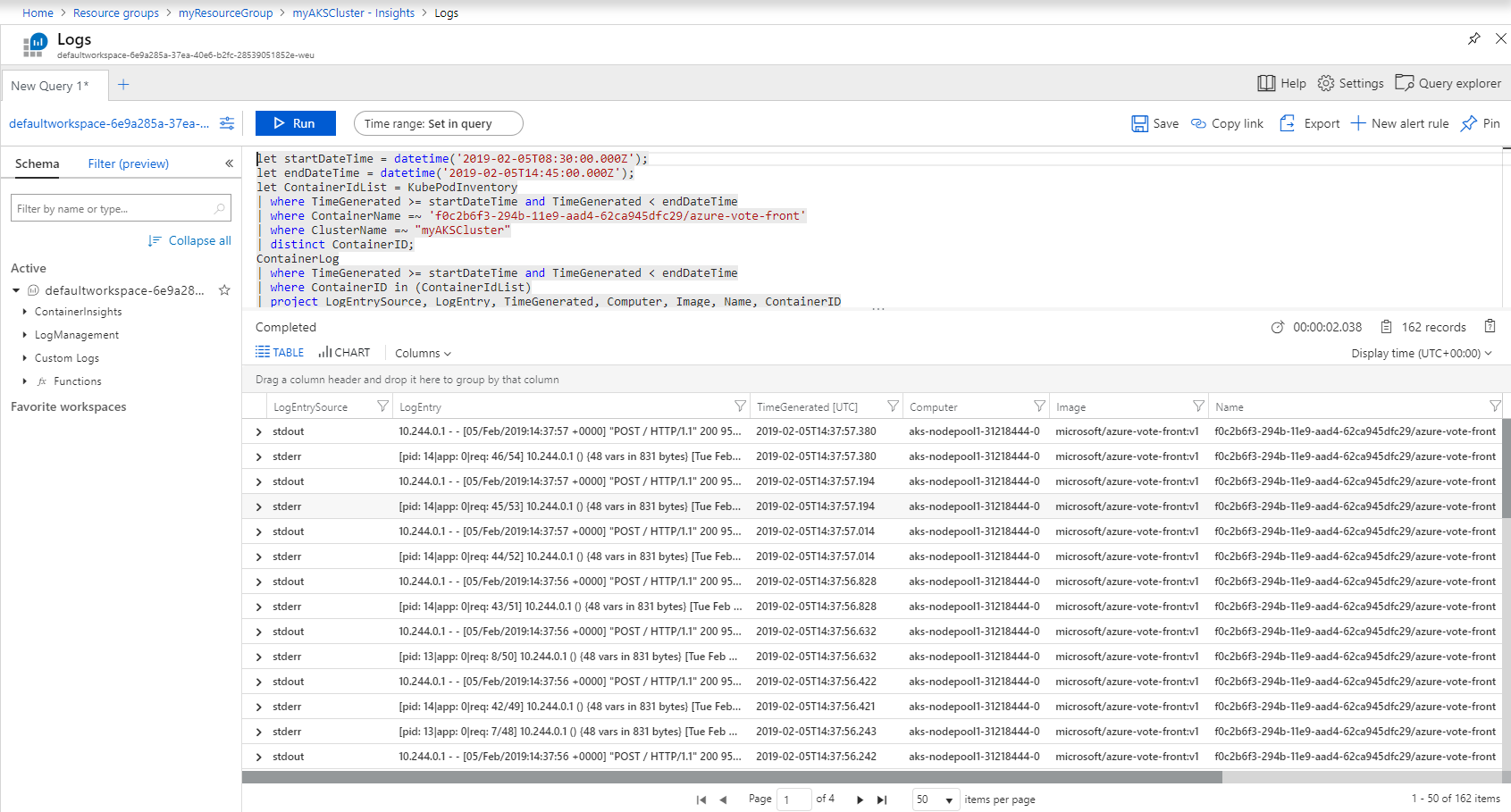
1. Select the **Containers** tab. There are two containers present, **azure-vote-back** and **azure-vote-front**



1. View logs for the **azure-vote-front** pod by selecting the **View container logs** link on the right-hand side of the containers list.



1. These logs include the stdout and stderr streams from the container.



If you have time you can click around some more within the settings to get a feel for structure and configuration, looking at the **Cluster**, **Node**, **Controller** and **Container** settings. Also go to the **Settings** > **scale** section to view how to configure scale. You can scale the number of nodes in your cluster to increase the total amount of cores and memory available for your container applications. Having at least 3 nodes is recommended for a more resilient cluster.

Congratulations! You have created and deployed a sample Application to an Azure Kubernetes cluster