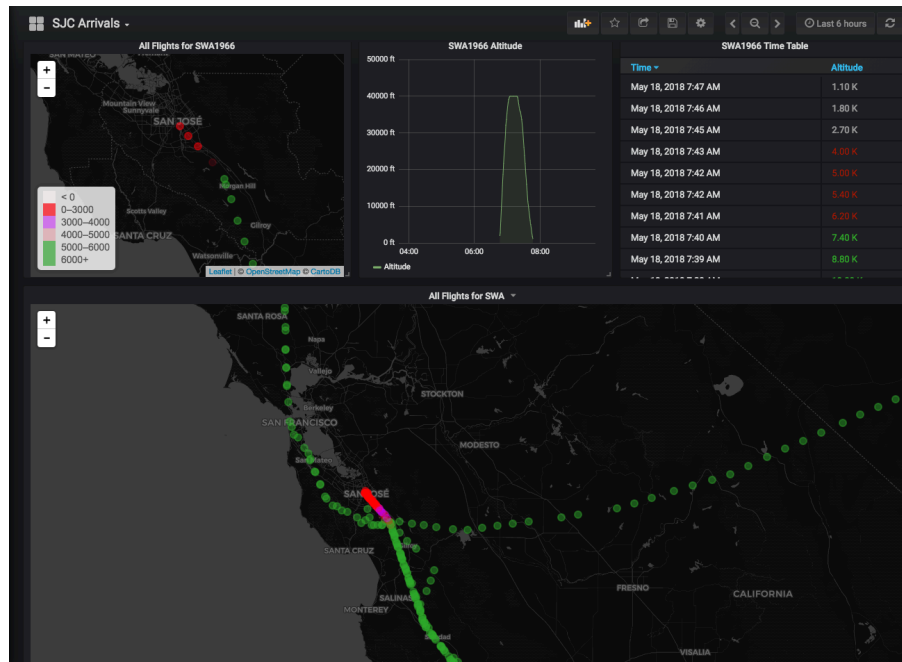
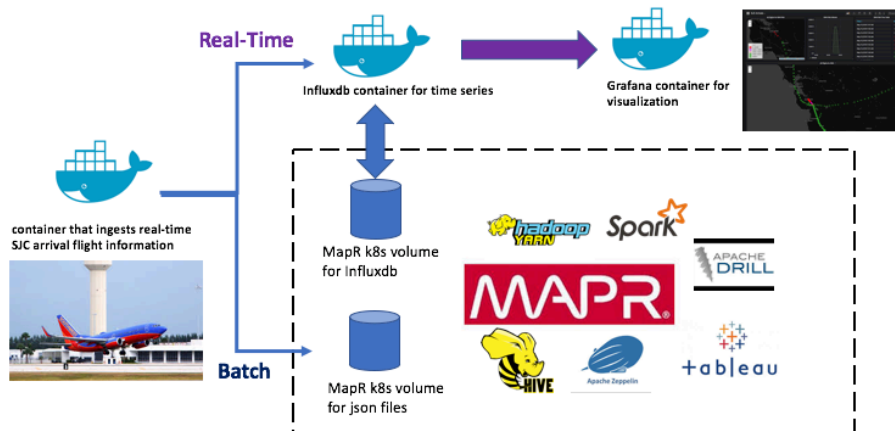


# Instructions on setting up SJC arrival flights monitoring with Azure Kubernetes Service (AKS) and MapR Data Fabric



This demo provides the steps to setup a real-time monitoring system for San Jose Airport (SJC) arrival flights on AKS using MapR Data Fabric for Kubernetes as the storage. It shows the flight tracks, flight numbers and altitudes of flights. Below is the architecture and data flow of the system.

## Demo (SJC Arrival Flights Tracking)



## Step 1: Create a MapR Sandbox on Azure

Go to <https://tinyurl.com/y77rxlar>, fill out the form and record your resource group name and login name and password. You will have a VM named 'mapr0' in the resource group in about 20 minutes. Click on 'mapr0' and record the IP address of it. You will need it to login later.

## Step 2: Create a AKS K8 cluster

Go to the Azure portal: <https://portal.azure.com>, select "Kubernetes Service", then create a K8s cluster. The K8s cluster has to be in the same resource group as the MapR Sandbox in Step 1 above.

Create Kubernetes cluster

Basics Networking Monitoring Tags Review + create

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. [Learn more about Azure Kubernetes Service](#)

**PROJECT DETAILS**

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

\* Subscription Demo\_and\_marketplace\_offering\_dev

\* Resource group ☐ Create new ☒ Use existing Check 'Use existing' and select the resource group for MapR Sandbox

jsunaks0508

**CLUSTER DETAILS**

\* Kubernetes cluster name jsunaks0510 ✓

\* Region East US

\* Kubernetes version 1.9.6

\* DNS name prefix jsunaks0510 ✓

**AUTHENTICATION**

\* Service principal (new) default service principal

[Config my service principal](#)

**SCALE**

The number and size of nodes in your cluster. For production workloads, at least 3 nodes are recommended for resiliency. For development or test workloads, only one node is required. You will not be able to change the node size after cluster creation, but you will be able to change the number of nodes in your cluster after creation. [Learn more about scaling in Azure Kubernetes Service](#)

\* Node size Standard D4s v3 (4 vcpus, 16 GB memory) ✓

[Change size](#)

\* Node count 3

[Review + create](#) [Next: Networking >](#) [Download a template for automation](#)


Select advanced networking configuration and use the same VNET "MapRSandboxVnet" as MapR Sandbox. Fill out the address range and DNS IP as described in screenshot below.


It is recommended to change the VM size to Standard D4s v3 for better performance, click 'Next Networking' when done.


You can enable HTTP application routing and choose between two networking options: "Basic" or "Advanced".


- **"Basic"** networking creates a new VNet for your cluster using default values.
- **"Advanced"** networking allows clusters to use a new or existing VNet with customizable addresses. Application pods are connected directly to the VNet, which allows for native integration with VNet features.

[Learn more about networking in Azure Kubernetes Service](#)


HTTP application routing  ☒ No ☐ Yes


Network configuration  ☐ Basic ☒ Advanced

\* Virtual network 


MapRSandboxVnet 


  
[Create new](#)


\* Subnet 


MapRSandboxSubnet 


  
[Create new](#)


\* Kubernetes service address range 

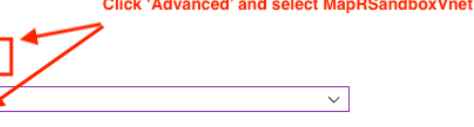
10.1.0.0/24  ✓

\* Kubernetes DNS service IP address 

10.1.0.10  ✓

\* Docker Bridge address 

172.17.0.1/16  ✓



[Review + create](#)

[« Previous: Basics](#)

[Next: Monitoring »](#)

[Download a template for automation](#)

Basics Networking Monitoring Tags Review + create

#### BASICS

Subscription	[REDACTED]
Resource group	jsunaks0508
Kubernetes cluster name	jsunaks0510
Region	East US
Kubernetes version	1.9.6
DNS name prefix	jsunaks0510
Node count	3
Node size	Standard_D4s_v3

#### NETWORKING

HTTP application routing	No
Network configuration	Advanced
Virtual network	MapRSandboxVnet
Subnet	MapRSandboxSubnet
Kubernetes service address range	10.1.0.0/24
Kubernetes DNS service IP address	10.1.0.10
Docker Bridge address	172.17.0.1/16

#### MONITORING

Enable container monitoring	Yes
Log Analytics workspace	DefaultWorkspace-[REDACTED]

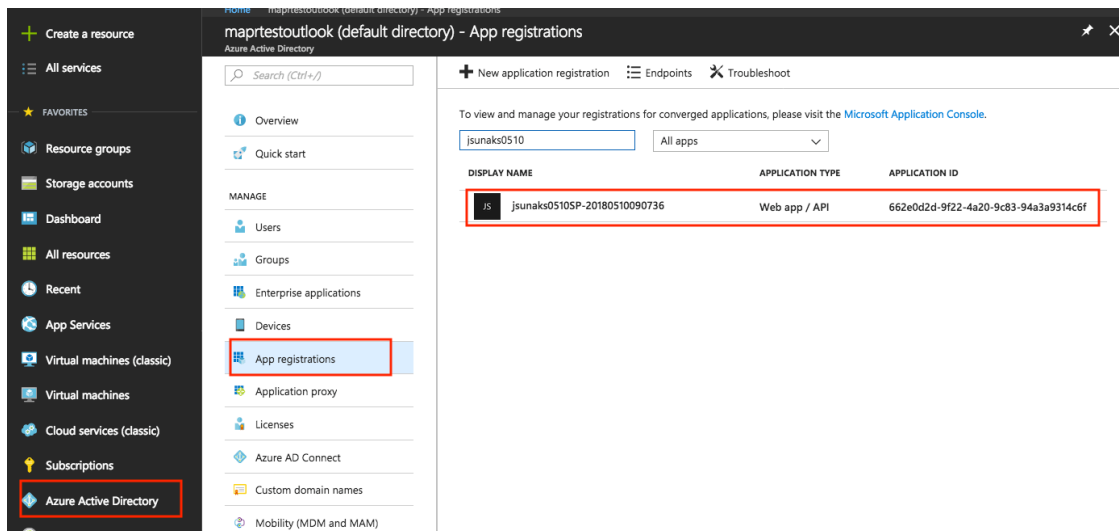
Create

« Previous: Tags

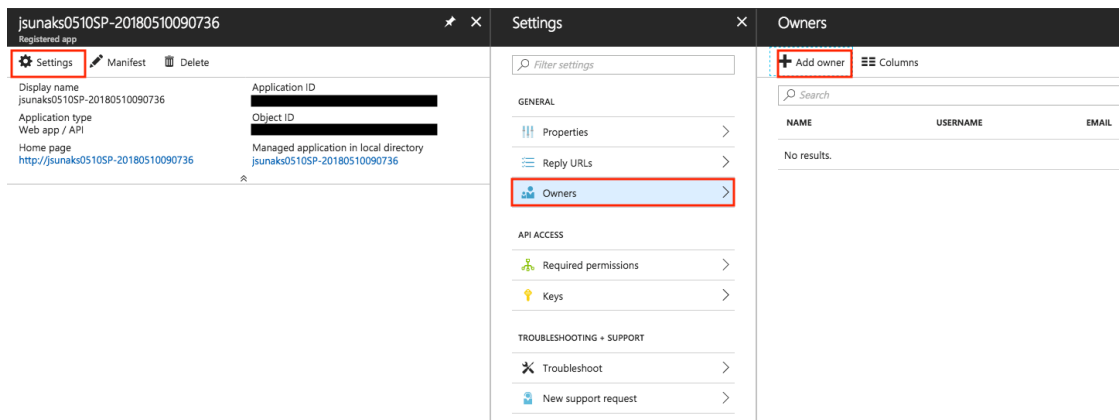
[Download a template for automation](#)

Step 3: Change AKS service principal role and owner so it can create load balancer in the VNET

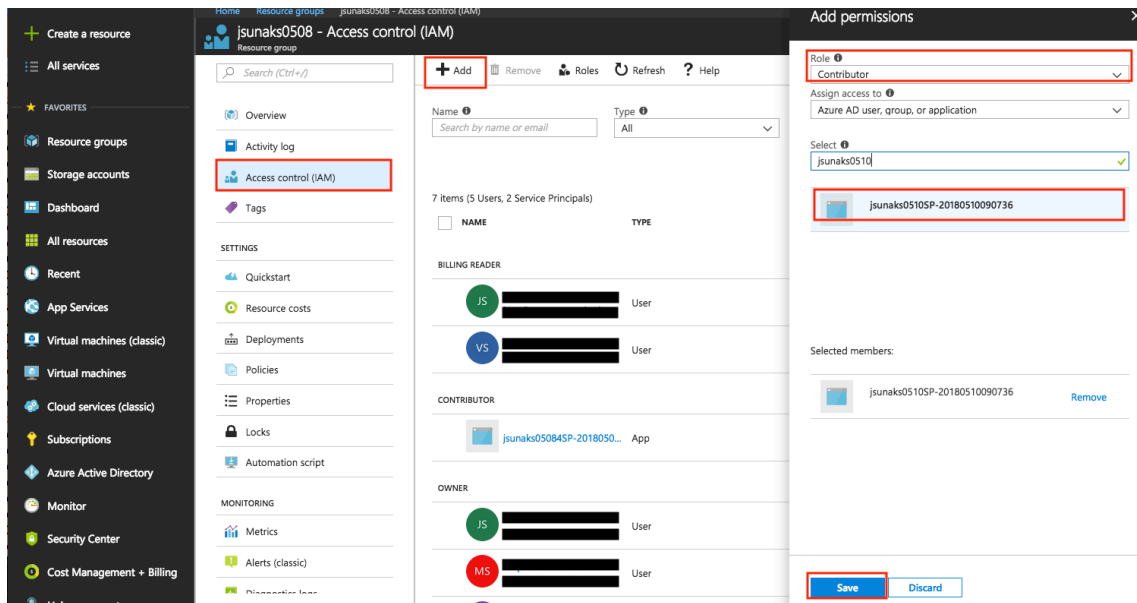
Find the service principal name created by AKS by going to 'Azure Active Directory' -> 'App registrations' and search for the SP name, typically it has a form like this: <resource group name>SP-<number string>, e.g. jsunaks0510SP-20180510090736.



Select the service principal, click on 'Settings' -> 'Owners' -> 'Add owner', then search for your login name and add



Now we need to assign a 'contributor' role to the service principal by going to the resource group created in Step 1, click on 'Access control (IAM)' -> 'Add' -> 'Role' -> 'Contributor' -> add the service principal name -> 'Save'



#### Step 4: Install Azure Cli Tool

login to Sandbox using the IP address, username and password as recorded in Step 1,  
sudo to become root

```
#curl -L https://raw.githubusercontent.com/maprpartners/SJC/master/config-azcli | bash
```

Follow the instructions to login to your Azure account

use "az account set" if you have multiple subscriptions,

use "az aks list" to find out K8s cluster name from Step 2 if you forgot

#### Step 5: Configure K8s client on MapR Sandbox

In the MapR Sandbox as root:

```
#git clone https://github.com/maprpartners/SJC.git
```

```
#cd SJC
```

Now we configure K8s

```
#bash config-k8s
```

"kubectl get node -o wide" to verify it is working

#### Step 6: Install MapR Data Fabric for K8s volume plugin

```
#bash inst_mapr_plugin
```

kubectl get pod --all-namespaces to verify, you should see mapr-kdfplugin-xxx  
daemon set running on each K8s slave

## Step 7: Deploy the SJC flight monitoring demo

```
#bash run
```

When completed, the script will provide a URL for you to look at the real world demo in action.