Joget on Azure Kubernetes Service

This article provides a tutorial on deploying, running and scaling Joget on Azure Kubernetes Service (AKS). AKS is a managed Kubernetes service offered by Azure.

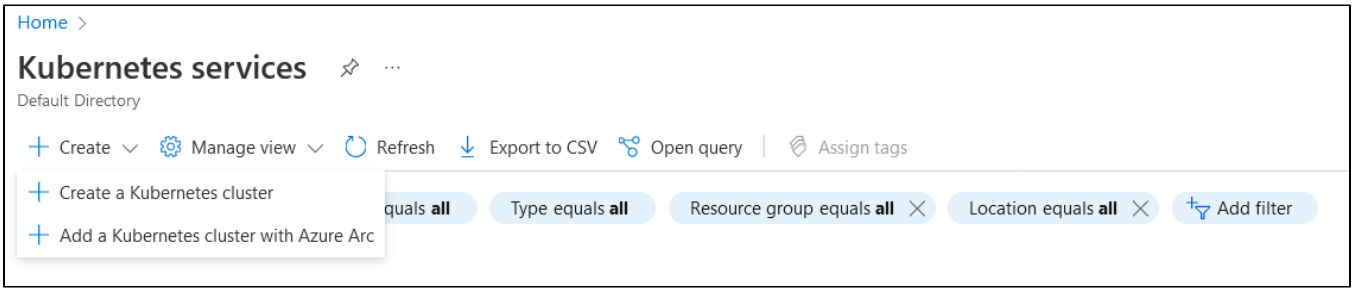
> If you are not familiar with Kubernetes, refer to Joget on Kubernetes for a quick introduction.

Deploy Joget on Azure Kubernetes Service

1. Create Kubernetes cluster in AKS

This guide will go through creation process with the Azure portal, if you want to create a cluster through Azure CLI please refer to the article Azure CLI.

From the Azure portal, go to the Kubernetes services then Create a Kubernetes cluster.



In the Basics page, choose the Subscription, Resource Group and input the Kubernetes cluster name. Adjust the other configuration settings as desired, or leave as default.

Home > Kubernetes services >

Create Kubernetes cluster

Basics | Node pools | Access | Networking | Integrations | Advanced | 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 *

Resource group * — (New) Resource group
Create new

Cluster details

Cluster preset configuration — Standard ($$)
To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time.
Learn more and compare presets

Kubernetes cluster name *

Region * — (US) West US 2

Availability zones — Zones 1,2,3
High availability is recommended for standard configuration.

Kubernetes version * — 1.23.12 (default)

API server availability
- (●) 99.95% — Optimize for availability.
- ( ) 99.5% — Optimize for cost.

99.95% API server availability is recommended for standard configuration.

Primary node pool

The number and size of nodes in the primary node pool in your cluster. For production workloads, at least 3 nodes are recommended for resiliency. For development or test workloads, only one node is required. If you would like to add additional node pools or to see additional configuration options for this node pool, go to the 'Node pools' tab above. You will be able to add additional node pools after creating your cluster. Learn more about node pools in Azure Kubernetes Service

Node size * — **Standard DS2 v2**
Standard DS2_v2 is recommended for standard configuration.
Change size

Scale method *
- ( ) Manual
- (●) Autoscale

Autoscaling is recommended for standard configuration.

Node count range * — 1 — 5

Review + create | < Previous | Next : Node pools >

In the Node pools tab, you can configure to add node pools into the cluster. Read on multiple node pools in AKS. For this guide, we will use a single node configuration.

Home > Kubernetes services >

Create Kubernetes cluster ...

Basics | **Node pools** | Access | Networking | Integrations | Advanced | Tags | Review + create

Node pools

In addition to the required primary node pool configured on the Basics tab, you can also add optional node pools to handle a variety of workloads Learn more about node pools

+ Add node pool Delete

Name	Mode	OS type	Node count	Node size
agentpool	System	Linux	1-5	Standard_DS2_v2

Enable virtual nodes

Virtual nodes allow burstable scaling backed by serverless Azure Container Instances. Learn more about virtual nodes

Enable virtual nodes

Node pool OS disk encryption

By default, all disks in AKS are encrypted at rest with Microsoft-managed keys. For additional control over encryption, you can supply your own keys using a disk encryption set backed by an Azure Key Vault. The disk encryption set will be used to encrypt the OS disks for all node pools in the cluster. Learn more

Encryption type: (Default) Encryption at-rest with a platform-managed key

For other tab options - Access, Networking, Integrations, Advanced and Tags, you can leave the default options or make adjustments/changes as necessary. After that, you can click on the Review + create and deploy the Kubernetes cluster.

Home > Kubernetes services >

Create Kubernetes cluster ...

Validation passed

Basics | Node pools | Access | Networking | Integrations | Advanced | Tags | **Review + create**

Basics

Subscription	
Resource group	(new) azureaks-resourcegroup
Region	West US 2
Kubernetes cluster name	jogetakscluster
Kubernetes version	1.23.12
Enable automatic upgrades	False

Node pools

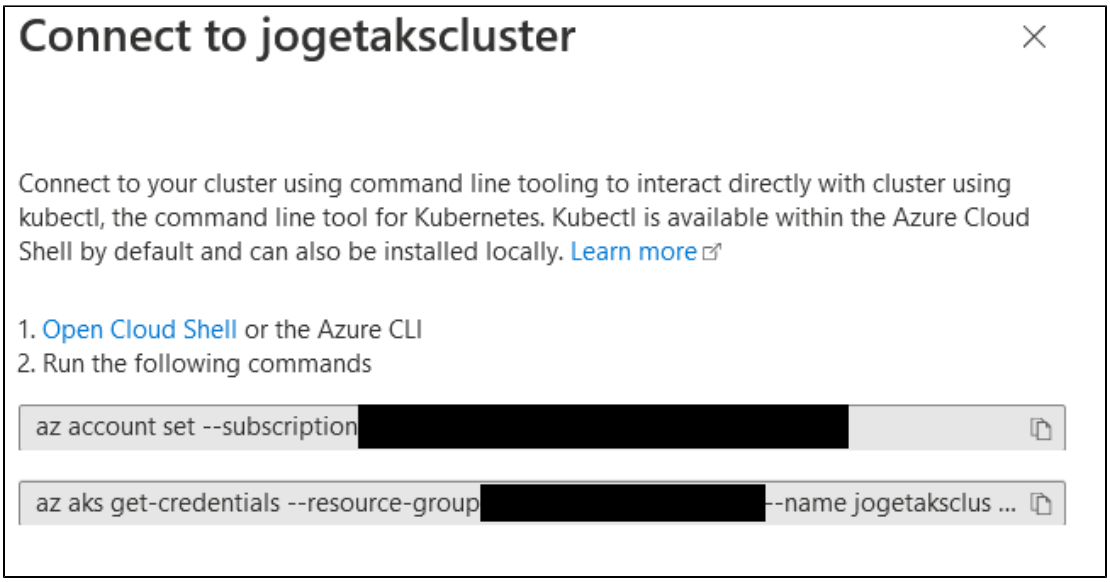
Node pools	1
Enable virtual nodes	Disabled

Access

Resource identity	System-assigned managed identity
Local accounts	Enabled
Authentication and Authorization	Local accounts with Kubernetes RBAC
Encryption type	(Default) Encryption at-rest with a platform-managed key

Create | < Previous | Next > | Download a template for automation

When the resource has completed their deployment, you can then connect to the cluster (read here) using Azure CLI/Azure Cloud Shell.



2.Deploy MySQL Database

Once we have a running cluster, you will need to deploy a database to be used by the Joget platform. You can pretty much follow the same method of deploying MySQL DB as in the Joget Kubernetes page.

- Create persistent storage using PersistentVolume and PersistentVolumeClaim

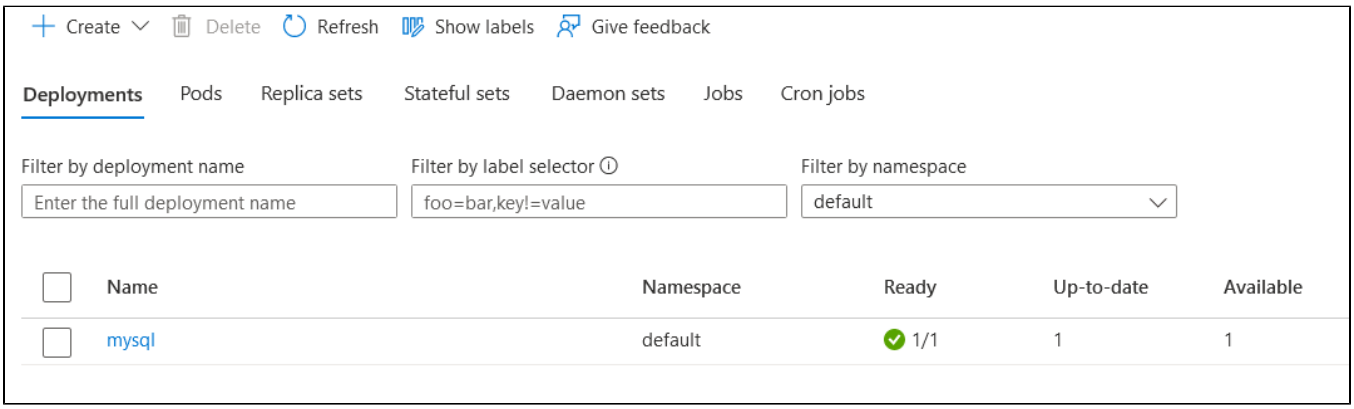
```
kubectl apply -f https://k8s.io/examples/application/mysql/mysql-pv.yaml
```

- Deploy the MySQL image

```
kubectl apply -f https://k8s.io/examples/application/mysql/mysql-deployment.yaml
```

- Inspect the deployment

```
kubectl describe deployment mysql
kubectl get pods -l app=mysql
kubectl describe pvc mysql-pv-claim
```



You need to modify the original yaml files for production usage (eg. using different version of MySQL image and setting up secret instead of plain password in the yaml).

3.Deploy shared storage in AKS

If you are running a multiple node Kubernetes cluster, you will need to allocate shared persistent storage with read write access by multiple nodes. In Azure, you can set up Azure NFS volume to be used in the Azure Kubernetes cluster. Refer to the official documentation here for detailed info and steps. You can also read more on other options for storage in Azure Kubernetes here.

- Create an Azure Ubuntu VM at the same Virtual Network as the AKS cluster.
- Setup NFS server into the VM.

From the link, you can use this script to set up the NFS server (edit the variables as necessary especially the AKS_SUBNET).

```
#!/bin/bash
# This script should be executed on Linux Ubuntu Virtual Machine

EXPORT_DIRECTORY=${1:-/export/data}
DATA_DIRECTORY=${2:-/data}
AKS_SUBNET=${3:-*}

echo "Updating packages"
apt-get -y update

echo "Installing NFS kernel server"

apt-get -y install nfs-kernel-server

echo "Making data directory ${DATA_DIRECTORY}"
mkdir -p ${DATA_DIRECTORY}

echo "Making new directory to be exported and linked to data directory: ${EXPORT_DIRECTORY}"
mkdir -p ${EXPORT_DIRECTORY}

echo "Mount binding ${DATA_DIRECTORY} to ${EXPORT_DIRECTORY}"
mount --bind ${DATA_DIRECTORY} ${EXPORT_DIRECTORY}

echo "Giving 777 permissions to ${EXPORT_DIRECTORY} directory"
chmod 777 ${EXPORT_DIRECTORY}

parentdir="$(dirname "$EXPORT_DIRECTORY")"
echo "Giving 777 permissions to parent: ${parentdir} directory"
chmod 777 $parentdir

echo "Appending bound directories into fstab"
echo "${DATA_DIRECTORY}    ${EXPORT_DIRECTORY}   none    bind  0  0" >> /etc/fstab

echo "Appending localhost and Kubernetes subnet address ${AKS_SUBNET} to exports configuration file"
echo "/export        ${AKS_SUBNET}(rw,async,insecure,fsid=1000,crossmnt,no_subtree_check)" >> /etc/exports
echo "/export        localhost(rw,async,insecure,fsid=1000,crossmnt,no_subtree_check)" >> /etc/exports

nohup service nfs-kernel-server restart
```

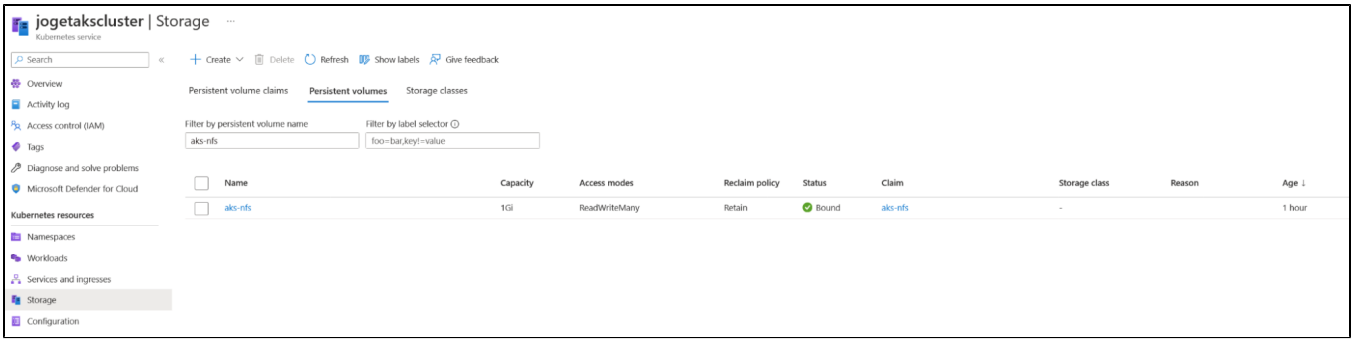
After the NFS server has been set up, you can then create the PersistentVolume and PersistentVolumeClaim.

Example azurenfsstorage.yaml;

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: aks-nfs
  labels:
    type: nfs
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: NFS_INTERNAL_IP
    path: NFS_EXPORT_FILE_PATH
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: aks-nfs
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 1Gi
  selector:
    matchLabels:
      type: nfs
```

Replace the values for NFS_INTERNAL_IP, NFS_NAME and NFS_EXPORT_FILE_PATH with the actual settings from your NFS Server.

```
kubectl apply -f azurenfsstorage.yaml
```



4.Deploy Joget DX

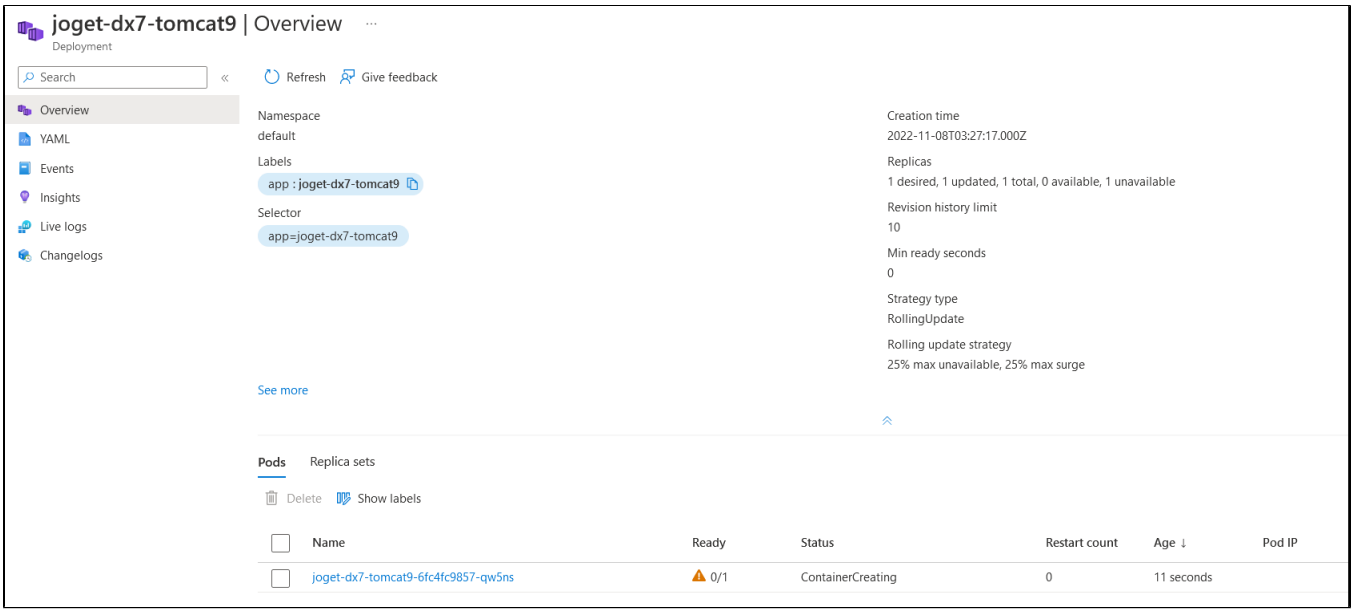
With the prerequisite database and persistent storage available, you can now deploy Joget. You can apply the example joget-dx7-tomcat9-aks.yaml file to deploy.

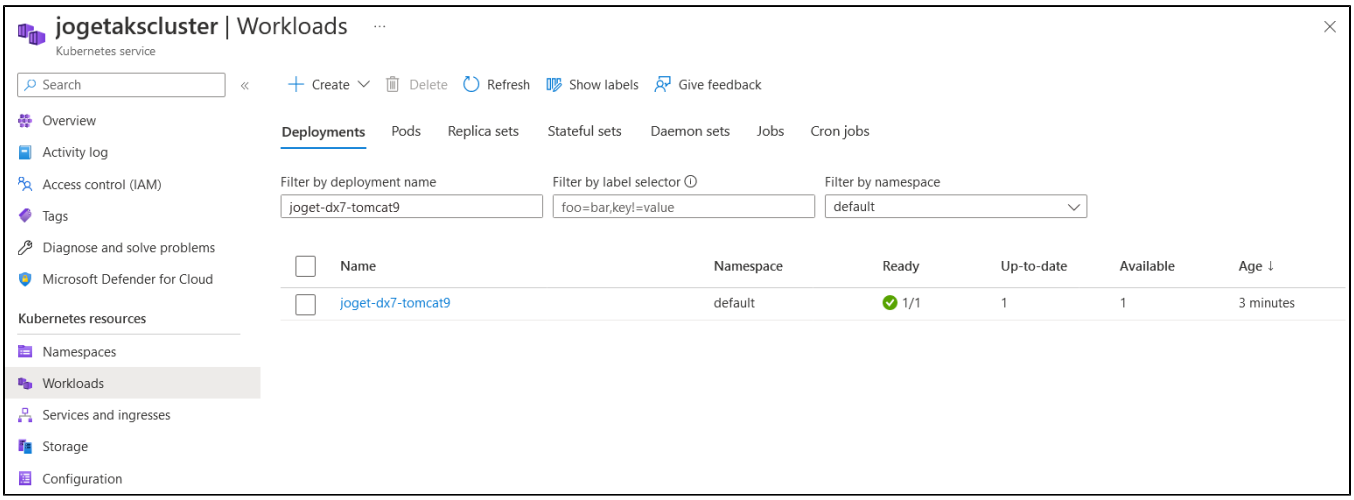
Example joget-dx7-tomcat9-aks.yaml;

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: joget-dx7-tomcat9
  labels:
    app: joget-dx7-tomcat9
spec:
  replicas: 1
  selector:
    matchLabels:
      app: joget-dx7-tomcat9
  template:
    metadata:
      labels:
        app: joget-dx7-tomcat9
    spec:
      initContainers:
       - name: init-volume
         image: busybox:1.28
         command: ['sh', '-c', 'chmod -f -R g+w /opt/joget/wflow; exit 0']
         volumeMounts:
           - name: joget-dx7-tomcat9-volume
             mountPath: "/opt/joget/wflow"
      volumes:
        - name: joget-dx7-tomcat9-volume
          persistentVolumeClaim:
            claimName: aks-nfs
      securityContext:
        runAsUser: 1000
        fsGroup: 0
      containers:
        - name: joget-dx7-tomcat9
          image: jogetworkflow/joget-dx7-tomcat9:latest
          ports:
            - containerPort: 8080
            - containerPort: 9080
          volumeMounts:
            - name: joget-dx7-tomcat9-volume
              mountPath: /opt/joget/wflow
          env:
            - name: KUBERNETES_NAMESPACE
              valueFrom:
                fieldRef:
                    fieldPath: metadata.namespace
---
apiVersion: v1
kind: Service
metadata:
  name: joget-dx7-tomcat9
  labels:
    app: joget-dx7-tomcat9
spec:
  ports:
  - name: http
    port: 8080
    targetPort: 8080
  - name: https
    port: 9080
    targetPort: 9080
  selector:
    app: joget-dx7-tomcat9
  type: ClusterIP
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: joget-dx7-tomcat9-clusterrolebinding
roleRef:
```

```
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: view
subjects:
  - kind: ServiceAccount
    name: default
    namespace: default
```

You can then check the deployment progress from the Azure portal. (Or use kubectl commands eg. kubectl get deployment joget-dx7-tomcat9)





5.Deploy Ingress for external connections

You can then expose the application for external access through Ingress. You can read more regarding Ingress in Kubernetes here. In this guide, we will use Nginx Ingress Controller as an example to access Joget.

Deploy Nginx Ingress Controller to AKS cluster

You can refer to the AKS documentation regarding creating ingress-nginx and also the nginx-ingress document.

There are 2 known methods of deploying the Nginx Ingress Controller to the AKS cluster;

1. Deploy through Helm
2. Use yaml file from the Nginx Ingress Controller Github

Install using Helm

Using Azure CLI/Cloud shell, set up the Helm for Nginx Ingress

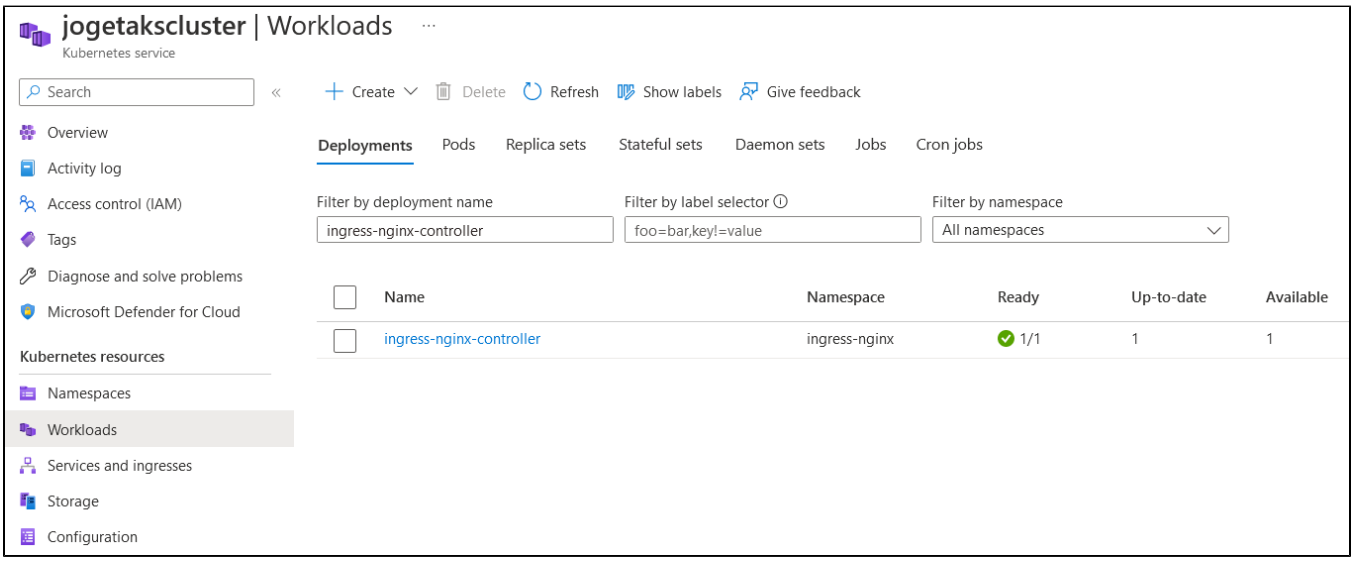
```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update

helm install ingress-nginx ingress-nginx/ingress-nginx --create-namespace --namespace nginx-ingress
```

Install using yaml file

You can use kubectl apply command.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.4.0/deploy/static
/provider/cloud/deploy.yaml
```



After the Ingress Controller has been deployed, we can then apply the Ingress yaml so that we can access the Joget application externally.

Example joget-ingress.yaml;

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: joget-dx7-tomcat9-ingress
  annotations:
    nginx.ingress.kubernetes.io/affinity: cookie
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
spec:
  ingressClassName: nginx
  rules:
    - http:
        paths:
          - path: /jw
            pathType: Prefix
            backend:
                service:
                  name: joget-dx7-tomcat9
                  port:
                    number: 8080
```

After the Ingress deployment is completed, you can get the public IP from the Kubernetes resources > Services and Ingresses pane in the Azure portal (eg. http://<external-ip>/jw).

jogetakscluster | Services and ingresses
Kubernetes service

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Microsoft Defender for Cloud

Kubernetes resources

- Namespaces
- Workloads
- Services and ingresses
- Storage
- Configuration

Create | Delete | Refresh | Show labels | Give feedback

Services | Ingresses

Filter by service name: Enter the full service name

Filter by namespace: ingress-nginx

Name	Namespace	Status	Type	Cluster IP	External IP	Ports	Age ↓
ingress-nginx-controller	ingress-nginx	Ok	LoadBalancer	10.0.137.101	20[redacted]	80:31032/TCP,4...	5 minutes
ingress-nginx-controller...	ingress-nginx	Ok	ClusterIP	10.0.188.71		443/TCP	5 minutes

Setup Database

To complete the Joget deployment, you need to perform a one-time Database Setup. Key in the MySQL service name and the Database Username and Password. Click on Save.

JOGET DX SETUP

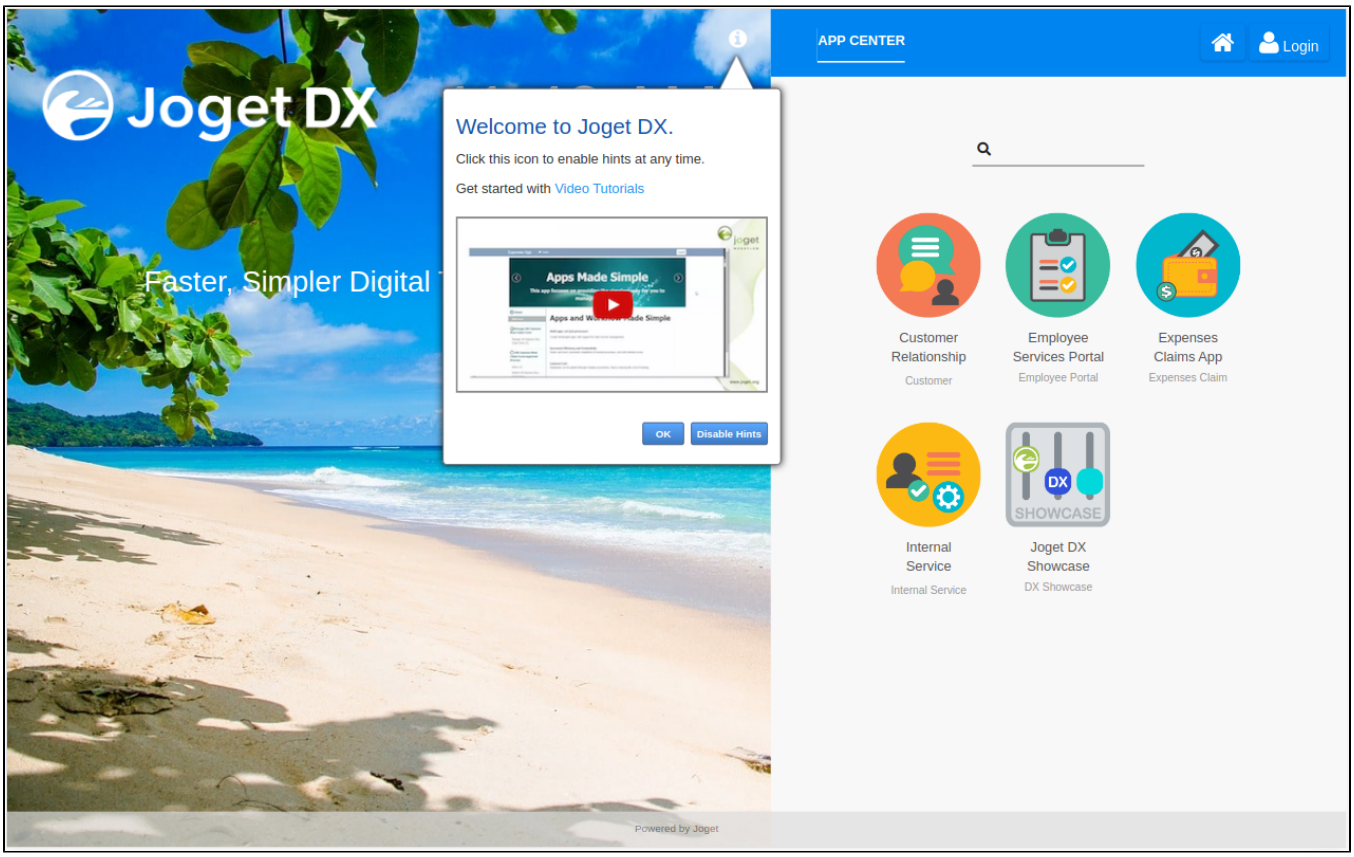
DATABASE SETUP

No database configuration was detected, so please configure your database settings below.
Please ensure that the database server is installed and running first. More Information

Database Type	MySQL
Database Host	mysql
Database Port	3306
Database Name	jwdb
Database User	root
Database Password	••••••••
Include Sample Apps	☑
Include Sample Users	☑

Save

Once the setup is complete, click on Done and you will be brought to the Joget App Center.



6.Setup cert-manager for TLS termination

Before starting the TLS setup, you need to set 'enable-underscores-in-headers' as true for Ingress by using configmap.

Example ingress-configmap.yaml;

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ingress-nginx-controller
  namespace: ingress-nginx
data:
  enable-underscores-in-headers: "true"
  allow-snippet-annotations: "true"
```

Update the Ingress configuration with **kubectl apply -f ingress-configmap.yaml**

Install cert-manager into the cluster

Similar to installing the ingress controller, you can install cert-manager either through Helm or through yaml file. Refer to the cert-manager official documentation here for detail. For this guide we will be using the yaml file method.

**Before going further with these steps, make sure that you have set up DNS to the public IP of the ingress that has been generated by AKS earlier.

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.10.0/cert-manager.yaml
```

Configure Let's Encrypt issuer

Example stagingissuer.yaml file;

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-staging
spec:
  acme:
    # The ACME server URL
    server: https://acme-staging-v02.api.letsencrypt.org/directory
    # Email address used for ACME registration
    email: [update email here]
    # Name of a secret used to store the ACME account private key
    privateKeySecretRef:
      name: letsencrypt-staging
    # Enable the HTTP-01 challenge provider
    solvers:
      - http01:
          ingress:
            class: nginx
```

```
kubectl apply -f stagingissuer.yaml
```

You can check on the status of the issuer resource after you have deployed it

```
kubectl describe issuer letsencrypt-staging
```

Deploy/Update the Ingress with TLS configuration

As we have previously deploy the Ingress without TLS configuration, we can update the Ingress yaml file to include the TLS configuration.

Example Ingress yaml with TLS;

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: joget-dx7-tomcat9-ingress
  annotations:
    nginx.ingress.kubernetes.io/affinity: cookie
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
    cert-manager.io/cluster-issuer: "letsencrypt-staging"
spec:
  ingressClassName: nginx
  tls:
  - hosts:
    - exampledomain.com
    secretName : aks-jogetworkflow
  rules:
    - host: exampledomain.com
      http:
        paths:
          - path: /jw
            pathType: Prefix
            backend:
              service:
                name: joget-dx7-tomcat9
                port:
                  number: 9080
```

This staging procedure is to ensure that the certificate is generated correctly before we setup the Issuer with Let's Encrypt production.

```
kubectl get certificate
```

```
[ ~/jogetaks ]$ kubectl get certificate
NAME                READY   SECRET              AGE
aks-jogetworkflow   True    aks-jogetworkflow   30s
```

```
kubectl describe certificate aks-jogetworkflow
```

If the certificate is generated correctly then we can set up the production Issuer.

Example productionissuer.yaml file;

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    # The ACME server URL
    server: https://acme-v02.api.letsencrypt.org/directory
    # Email address used for ACME registration
    email: [update email here]
    # Name of a secret used to store the ACME account private key
    privateKeySecretRef:
      name: letsencrypt-prod
    # Enable the HTTP-01 challenge provider
    solvers:
    - http01:
        ingress:
          class:  nginx
```

Update the ingress yaml file with the production annotation.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: joget-dx7-tomcat9-ingress
  annotations:
    nginx.ingress.kubernetes.io/affinity: cookie
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
spec:
  ingressClassName: nginx
  tls:
  - hosts:
    - exampledomain.com
    secretName : aks-jogetworkflow
  rules:
    - host: exampledomain.com
      http:
        paths:
          - path: /jw
            pathType: Prefix
            backend:
              service:
                name: joget-dx7-tomcat9
                port:
                  number: 9080
```

After applying the updated ingress yaml, you need to delete the previous secret so that the new certificate can be generated for the production.

```
kubectl delete secret aks-jogetworkflow
```

Then run back the describe command to check on the cert status

```
kubectl describe certificate aks-jogetworkflow
```

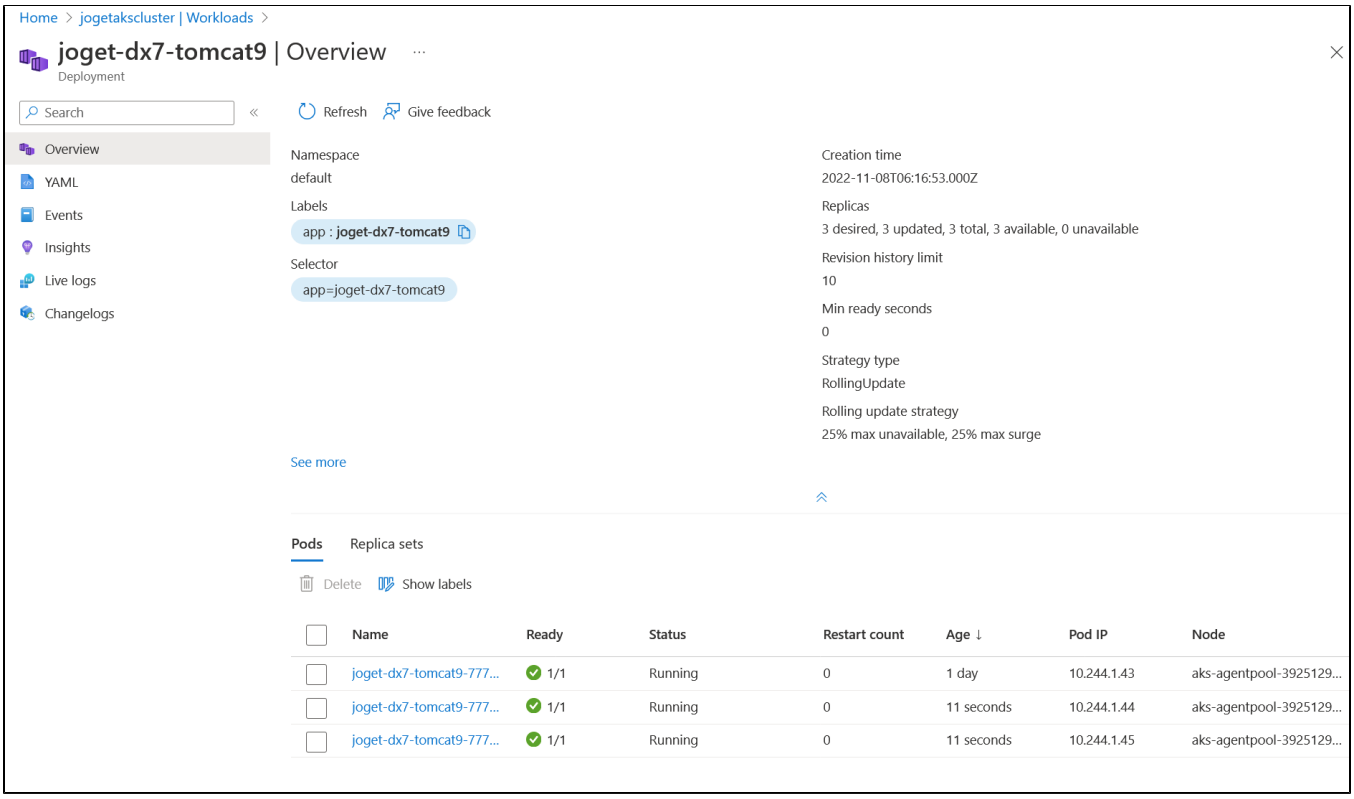
After the new certificate has been issued, you can then access the Joget domain with https to ensure that everything is working properly.

7.Scale Deployment

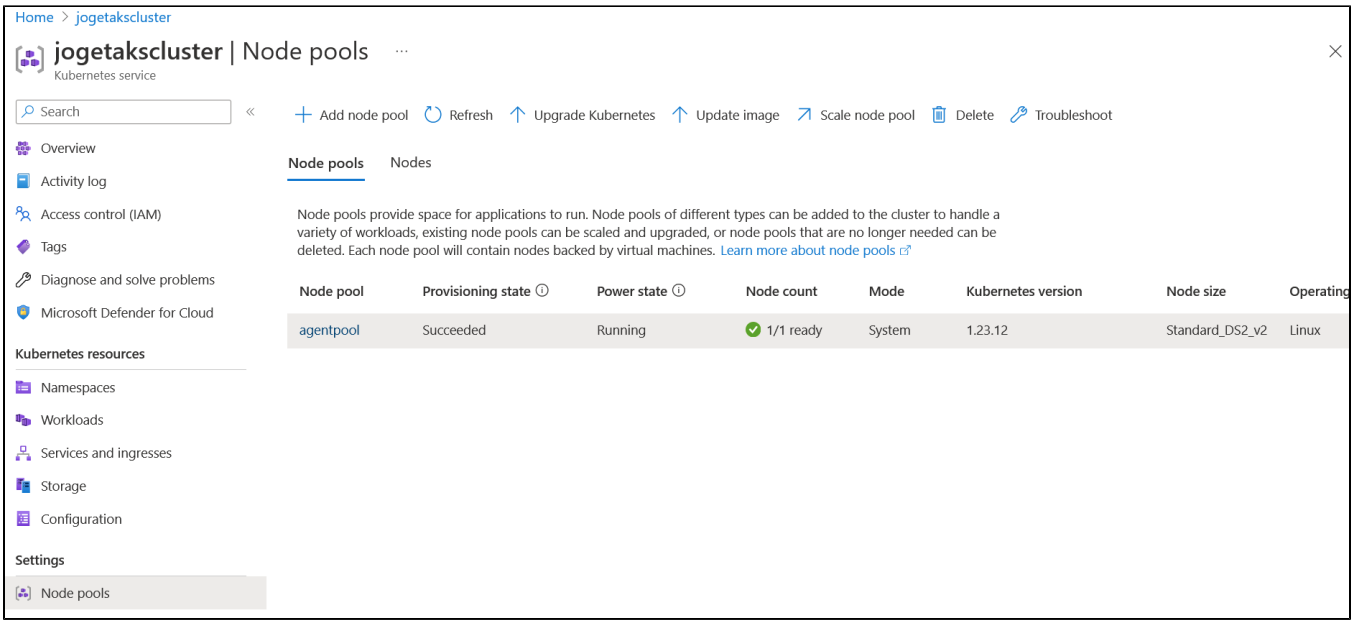
While you can set the nodes or pods to autoscale in AKS (read here), you can also scale the number of nodes or pods manually. To scale the number of pods running Joget, you can use the kubectl command.

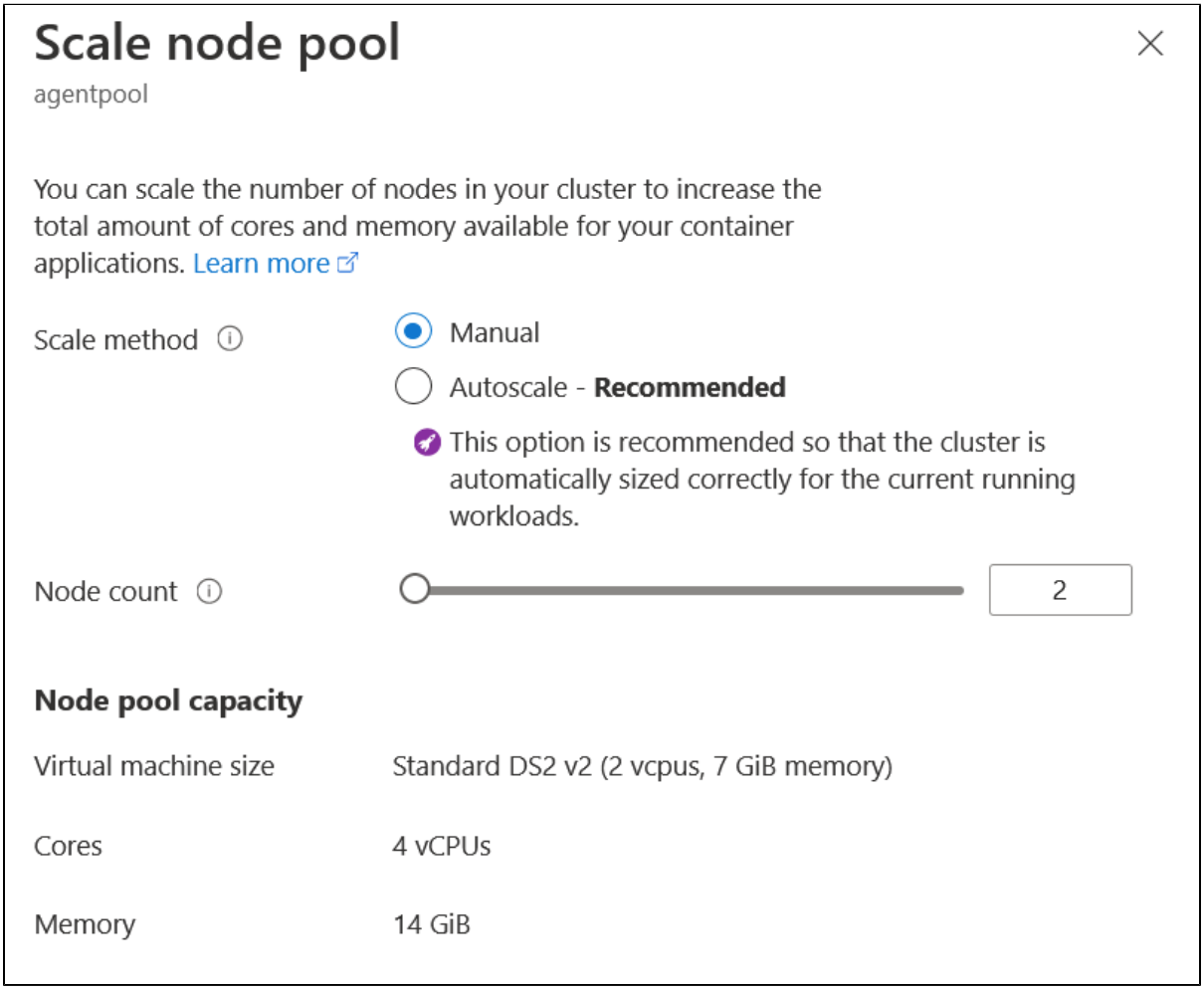
```
kubectl scale --replicas=3 deployment/joget-dx7-tomcat9
```

Adjust the replica number as you desired and the desired number of pods will initialize and startup.



As for the node, you can scale the node count of the node pool from the Azure portal. Go to the Cluster in the Kubernetes service (in this guide example jogetakscluster) > Settings > Node pools. Select the node pool and then click on the Scale node pool. Choose Manual as the Scale method and input the desired node count (maximum available resource is based on the VM size that you have chosen).





8.Additional Note :- Configure Joget in AKS with Azure Database for MySQL

This additional note is to guide on how to configure Joget in AKS connecting to the Azure Database for MySQL. It is based on the Azure guide here which has been modified to use the Joget application. This guide uses the Azure portal as to assist in visual guidance.

Assuming that a resource group has been created, then from the Azure portal, go to the Virtual network services then Create virtual network and also the subnets for the MySQL and AKS resources.

Based on the Microsoft recommendations of using Azure CNI to setup the configuration of AKS and Azure DB. You can read more on the AKS networking best practises here.

Home > Virtual networks >

Create virtual network

Basics | Security | IP addresses | Tags | Review + create

Azure Virtual Network (VNet) is the fundamental building block for your private network in Azure. VNet enables many types of Azure resources, such as Azure Virtual Machines (VM), to securely communicate with each other, the internet, and on-premises networks. VNet is similar to a traditional network that you'd operate in your own data center, but brings with it additional benefits of Azure's infrastructure such as scale, availability, and isolation. Learn more.

Project details

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

Subscription * : Joget Inc

Resource group * : faiztest-rg

Create new

Instance details

Virtual network name * : mysqlaksdemo

Region * : (US) East US

Deploy to an edge zone

Then add the IP address space for the Virtual network.

Add an IP address space

The address space for a virtual network has one or more non-overlapping address ranges. It is recommended to use private (RFC 1918), shared (RFC 6598), or local (RFC 4193) address ranges. Learn more.

Address space type : IPv4 / IPv6

Starting address * : 155.55.0.0

Address space size * : /16 (65536 addresses)

IP address space : 155.55.0.0 - 155.55.255.255 (65536 addresses)

After that create 2 subnets for the MySQL resource and also the AKS cluster.

Add a subnet

Select an address space and configure your subnet. You can customize a default subnet or select from subnet templates if you plan to add select services later. Learn more

IP address space: 155.55.0.0/16

155.55.0.0 - 155.55.255.255 (65536 addresses)

Subnet details

Subnet template: Default

Name *: mysql

Starting address *: 155.55.1.0

Subnet size: /24 (256 addresses)

IP address space: 155.55.1.0 - 155.55.1.255 (256 addresses)

Add a subnet

Select an address space and configure your subnet. You can customize a default subnet or select from subnet templates if you plan to add select services later. Learn more

IP address space: 155.55.0.0/16

155.55.0.0 - 155.55.255.255 (65536 addresses)

Subnet details

Subnet template: Default

Name *: aks

Starting address *: 155.55.2.0

Subnet size: /24 (256 addresses)

IP address space: 155.55.2.0 - 155.55.2.255 (256 addresses)

We can then create the Virtual network resource.

8.1 Deploy Azure MySQL Flexible Server

Search for resource Azure Database for MySQL flexible servers. Then click on Create Flexible server.

In the Basics tab, we configured as below since we are testing the resource. Modify as needed (also note in the screenshot the MySQL version is 5.7, we have tested with version 8 also).

Server names, networking connectivity method, zone redundant HA and backup redundancy cannot be changed after server

Project details

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

Subscription * : Joget Inc

Resource group * : faiztest-rg

Create new

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name * : mysqlaksdemo

Region * : East US

MySQL version * : 5.7

Workload type :

- ○ **For small or medium size databases**
- ○ **Tier 1 Business Critical Workloads**
- ◉ **For development or hobby projects**

Compute + storage :

Burstable, B1ms
1 vCores, 2 GiB RAM, 20 GiB storage, 360 IOPS
Geo-redundancy : Disabled
Configure server

Availability zone : No preference

High availability

Same zone and zone redundant high availability provide additional server resilience in the event of a failure. You can also specify high availability options in 'Compute + storage'.

Enable high availability : ☐

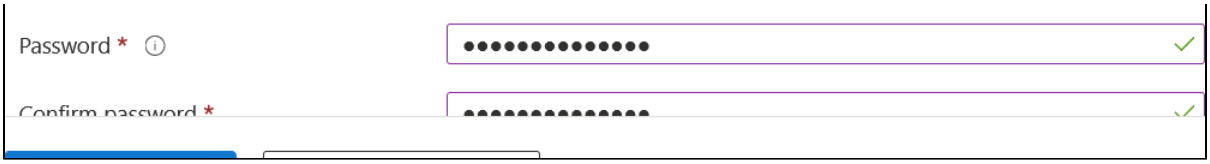
Authentication

Select the authentication methods you would like to support for accessing this MySQL server. MySQL password authentication allows you to create and use a ROLEs (usernames) and use a password to authenticate.
Enabling Azure Active Directory authentication allows you to create ROLEs based on your Azure Active Directory accounts and generate an authentication token with which to authenticate. Learn more

Authentication method :

- ◉ MySQL authentication only
- ○ Azure Active Directory authentication only
- ○ MySQL and Azure Active Directory authentication

Admin username * : joget



Then in the Networking tab, we need configure the resource to use the Virtual network that we created earlier.

Configure networking access and security for your server.

Network connectivity

You can connect to your server by specifying a public IP address specified below or from within a selected virtual network.

Connectivity method

- ○ Public access (allowed IP addresses)
- ◉ Private access (VNet Integration)

Connections from within the virtual network configured below will have access to this server. Learn more

Virtual network

Virtual networks are logically isolated from each other in Azure. Virtual network gives you a highly secure environment to run your MySQL Flexible Server and other types of Azure resources

Subscription * : Joget Inc

Virtual network * : mysqlaksdemo

Manage selected virtual network
Create virtual network

Subnet * : mysqlaksdemo/mysql (155.55.1.0/24) (Delegation required to service...

This subnet will be delegated for use only with MySQL Flexible Server (Microsoft.DBforMySQL/flexibleServers).

Your current subnet selection has 251 addresses available.

Private DNS integration

Private DNS zone integration is required to connect to your Flexible Server in virtual network using server name (fully qualified domain name).
A new private DNS zone will be created or you can optionally choose an existing one linked to the selected virtual network. With private DNS zone integration, the DNS records for the server name will be updated automatically in case the IP address of your Flexible Server changes. Learn more

Subscription * : Joget Inc

Private DNS zone * : (New) mysqlaksdemo.private.mysql.database.azure.com

Encrypted connections

This server supports encrypted connections using Transport Layer Security (TLS 1.2). For information on downloading the certificate, refer to connecting with TLS/SSL. Learn more

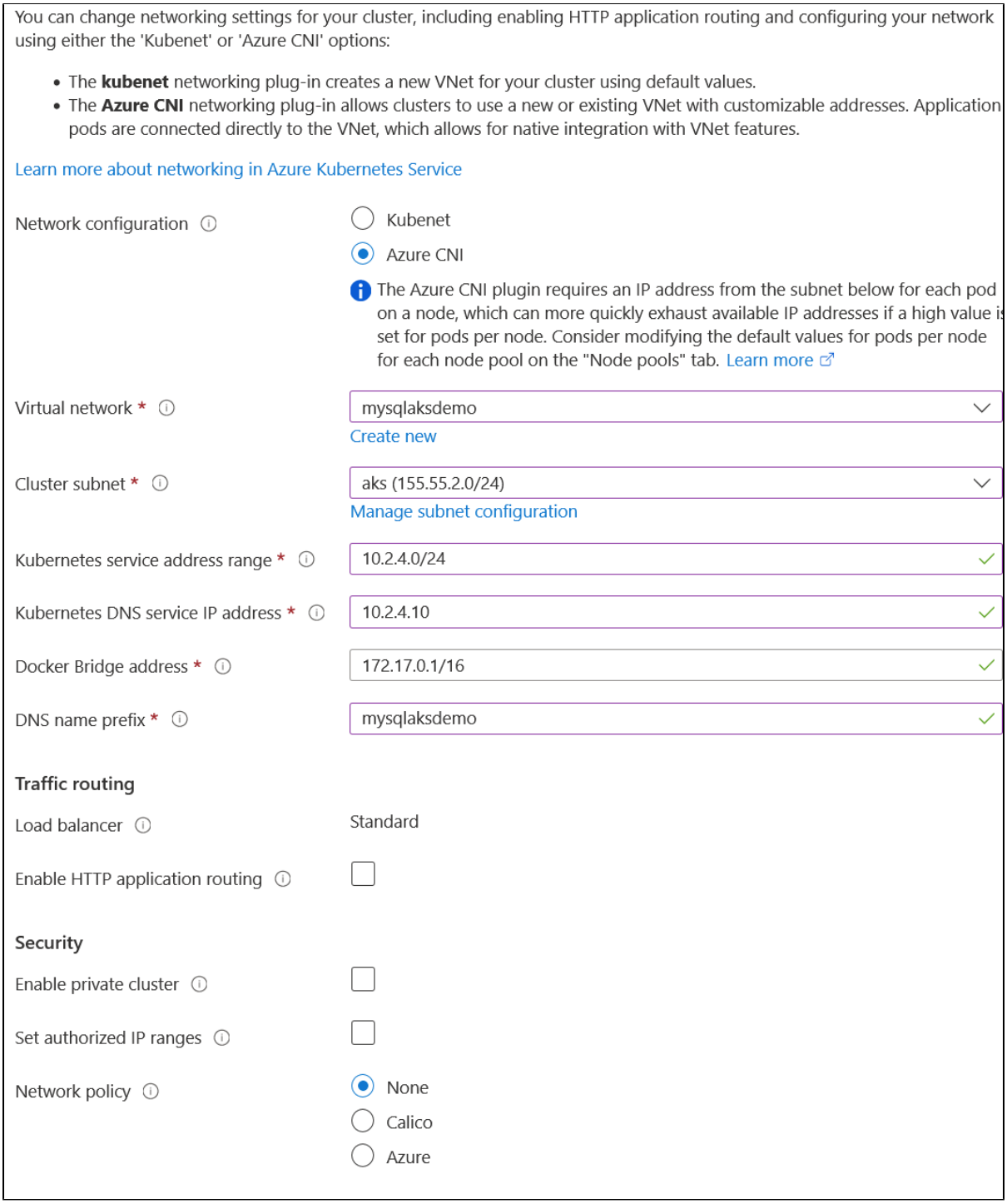
For Security and Tags tabs we can leave as default or make changes as necessary. After done we can create the resource.

For testing purpose, after the MySQL resource has been created, we turned off the *require_secure_transport* parameter. This is so that we will be able to intialise through the Joget setup page. Should you need this parameter to be enabled, you can then edit the *app_datasource-<profile>.properties* file. Example of the workflowUrl parameter with the *require_secure_transport* parameter turned on;

```
workflowUrl=jdbc\:mysql\://<azuredburlhere>\:3306/jwdb?characterEncoding\=UTF-8&useSSL\=true&allowPublicKeyRetrieval\=true
```

8.2 Deploy AKS Cluster and Joget

As to deploying the AKS cluster and Joget itself, the steps are similar as above in this KB page. The only different part is when setting up the AKS cluster, in the Networking tab, we need to specify to use Azure CNI and associate the virtual network and subnet that we have created earlier.



After AKS and Joget have been deployed, we will be able to do the DB setup on Joget.

DATABASE SETUP

No database configuration was detected, so please configure your database settings below.
Please ensure that the database server is installed and running first. More Information

Database Type	MySQL
Database Host	mysqlaksdemo.mysql.database.azure.com
Database Port	3306
Database Name	jwdb
Database User	joget
Database Password	●●●●●●●●●●●●●●
Include Sample Apps	☑
Include Sample Users	☑

Save