

Table of Contents

SFTPGo for AKS.....	2
Initial installation and configuration.....	2
Additional environment variables.....	4
Environment variables from Secrets.....	5
Environment variables from ConfigMaps.....	5
Services.....	7
Volumes.....	8
Troubleshooting and support.....	9

SFTPGo for AKS

This guide explains how to customize the Azure Marketplace SFTPGo offer for AKS. This document explains configuration options specific to SFTPGo and it assumes you are familiar with Kubernetes and Azure concepts.

Initial installation and configuration

To deploy from the marketplace page:

- Click **Get it Now**.
- Click **Continue**. You will be redirected to the Azure portal and may need to login to your account.
- **Click Create**.

The initial setup screen allows you to create a new cluster or use an existing one.

The only required settings is the “Cluster extension resource name”, set it to something like “sftpgo”.

You can configure the other main configuration settings now or, recommended, accept the defaults and reconfigure the cluster later once it is up and running properly.

Basics	Cluster Details	SFTPGo Configuration	Review + create
Cluster extension resource name *	<i>i</i>	<input type="text"/>	
Replica count	<i>i</i>	<input type="range"/>	<input type="text" value="1"/>
Database			
Driver *	<i>i</i>	<input type="text" value="SQLite"/>	<i>v</i>
Name *	<i>i</i>	<input type="text" value="sftpgo.db"/>	
Host	<i>i</i>	<input type="text"/>	
Port	<i>i</i>	<input type="text"/>	
Username	<i>i</i>	<input type="text"/>	
Password	<i>i</i>	<input type="text"/>	
Confirm password	<i>i</i>	<input type="text"/>	
TLS	<i>i</i>	<input type="text" value="Disabled"/>	<i>v</i>

The settings are self explanatory, you can enable autoscaling or set a fixed number of replicas and configure the database connection parameters.

Embedded databases like SQLite and Bolt DB require no configuration but cannot be shared across multiple pods. So, if you plan to use multiple pods, you need to configure the connection parameters to a PostgreSQL or MySQL compatible database. Azure Database for PostgreSQL/MySQL will work fine.

If you want to use an embedded database and a single pod, you need to mount a volume at `/var/lib/sftpgo` to persist the database across pod restarts.

To use the local filesystem for SFTPGo users you need to mount a volume at `/srv/sftpgo`, cloud storage backends like Azure Blob don't require any volume.

Once the cluster is up and running you can use **az** CLI to get the access credentials and then manage the cluster using **kubectl**. More detail at the following link.

<https://learn.microsoft.com/en-us/cli/azure/aks?view=azure-cli-latest#az-aks-get-credentials>

Please note that the SFTPGo cluster will run in a dedicated namespace so remember to add the namespace to your kubectl commands, for example:

```
kubectl get pods -o wide --namespace=sftpgo
```

Using kubectl you can, for example, gain shell access to pods or consult their logs.





From the Azure portal, select **Kubernetes service**, you can configure SFTPGo from the **Extensions + applications**, expand the **Configuration settings** section, you will see the previously configured parameters like in the following screenshot.

Configuration settings

Configuration settings are key value pairs that can be used to customize an extension. To prevent problems when changing key value pairs refer to SFTPGo Authors documentation to understand the settings you're allowed to update. [Learn more](#)

Note: Protected configurations settings can only be updated through Azure CLI.

Changing these key value pairs may cause problems.

database.driver	sqlite	
database.name	sftpgo.db	
database.sslMode	0	
replicaCount	1	

You can also use **az** CLI to manage the above settings. Learn [more](#).

Main configuration parameters:

- **autoscaling.enabled**, default *false*
- **autoscaling.minReplicas**, default *1*

- **autoscaling.maxReplicas**, default *100*
- **autoscaling.targetCPUUtilizationPercentage**, default *80*
- **replicaCount**, number of replicas (pods) to launch if auto scaling is disabled. Default *1*
- **database.driver**, supported values: `sqlite`, `postgresql`, `mysql`, `cockroachdb`, `bolt`
- **database.name**, for `postgresql`, `mysql` and `cockroachdb` driver, this database must exist. SFTPGo will create/update the required tables on startup
- **database.host**
- **database.port**
- **database.username**
- **database.password**
- **database.sslMode**, 0 disabled, 1 enabled
- **sftpd.enabled**, true/false. Default *true*
- **ftpd.enabled**, true/false. Default *false*. FTP requires persistent sessions.
- **httpd.enabled**, true/false. Default *true*
- **webdavd.enabled**, true/false. Default *false*. WebDAV requires persistent sessions.
- **serviceAccount.create**, true/false. Default *true*

WARNING: please note that SFTPGo pods will refuse to start if they cannot connect to the configured database and may get stuck in **CrashLoopBackOff** mode. Therefore it is critical to test configuration changes in a test environment before applying them to the production environment.

Additional environment variables

SFTPGo allows to override all the available configuration options using environment variables.

Learn [more](#).

To set an environment variable simply use **env.<environment variable name>** as key. Here is an example.



Environment variables from Secrets

You can create environment variables from Secrets by setting:

- **envFromSecrets[idx].name**, this is the environment variable name
- **envFromSecrets[idx].secretName**, this is the name of the Kubernetes Secret
- **envFromSecrets[idx].secretKey**, this is the key to set from the Kubernetes Secret

idx starts from 0.

Suppose you have a Secret named **encryption-key** like this:

apiVersion: v1




data:

masterkey: YW1haDRvb3c5eGFleGFleXUwZWVYNG90OWFldGVlcHUK

kind: Secret

...

Here is an example configuration to set an environment variable from it.

envFromSecrets[0].name	SFTPGO_KMS__SECRETS__MASTER_KEY	
envFromSecrets[0].secretName	encryption-key	
envFromSecrets[0].secretKey	masterkey	

Environment variables from ConfigMaps

You can create environment variables from Secrets by setting:

- **envFromConfigMaps[idx].name**, this is the environment variable name
- **envFromConfigMaps[idx].configMapName**, this is the name of the Kubernetes ConfigMap
- **envFromConfigMaps[idx].configMapKey**, this is the key to set from the Kubernetes ConfigMap

idx starts from 0.

Suppose you have a configMap named **defender** like this:

apiVersion: v1

data:







ban_time: "60"

enabled: "1"

kind: ConfigMap

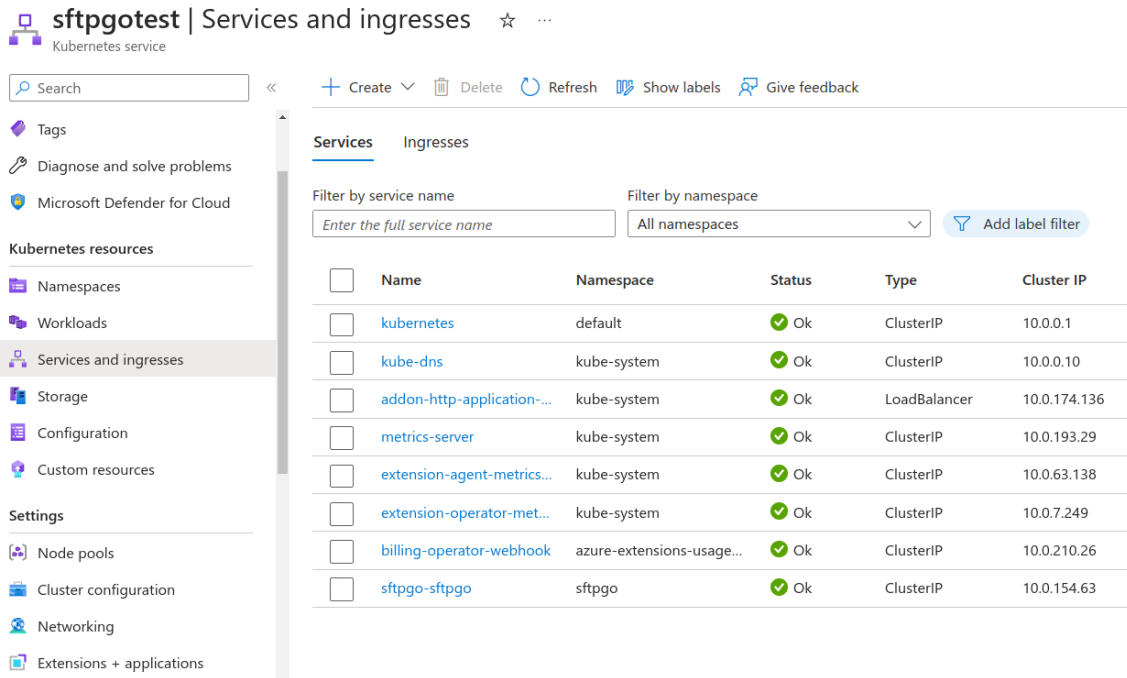
...

Here is an example configuration to set an environment variable from it.

envFromConfigMaps[0].name	SFTPGO_COMMON_DEFENDER_BAN_TIME	
envFromConfigMaps[0].configMapName	defender	
envFromConfigMaps[0].configMapKey	ban_time	
envFromConfigMaps[1].name	SFTPGO_COMMON_DEFENDER_ENABLED	
envFromConfigMaps[1].configMapName	defender	
envFromConfigMaps[1].configMapKey	enabled	

Services

From the **Services and ingresses** section you can configure how to reach the configured SFTPGo services.



The screenshot shows the Kubernetes dashboard for a cluster named 'sftpgotest'. The 'Services and ingresses' section is active, displaying a list of services. The table below represents the data shown in the screenshot:

Name	Namespace	Status	Type	Cluster IP
kubernetes	default	Ok	ClusterIP	10.0.0.1
kube-dns	kube-system	Ok	ClusterIP	10.0.0.10
addon-http-application...	kube-system	Ok	LoadBalancer	10.0.174.136
metrics-server	kube-system	Ok	ClusterIP	10.0.193.29
extension-agent-metrics...	kube-system	Ok	ClusterIP	10.0.63.138
extension-operator-met...	kube-system	Ok	ClusterIP	10.0.7.249
billing-operator-webhook	azure-extensions-usage...	Ok	ClusterIP	10.0.210.26
sftpgo-sftpgo	sftpgo	Ok	ClusterIP	10.0.154.63

The SFTPGo extension creates a default service. Here are the settings to customize it:

- **service.type**, default *ClusterIP*. Learn [more](#).
- **service.LoadBalancerIP**, default unset
- **service.ports.sftp.port**, default 22
- **service.ports.ftp.port**, default 21
- **service.ports.ftpPassiveRange.start**, default 31100
- **service.ports.ftpPassiveRange.end**, default 31120
- **service.ports.webdav.port**, default 81
- **service.ports.http.port**, default 80
- **externalTrafficPolicy**, default unset. Learn [more](#).
- **sessionAffinity**, default unset. Learn [more](#).

You can also configure additional services exposing servers (SFTP, FTP, WebDAV, HTTP) individually. Additional services need at least one port. Let's see an example of the config keys for a service named "public" which exposes SFTP and FTP, you can replace "public" with any string:

- **services.public.type**
- **services.public.LoadBalancerIP**
- **services.public.externalTrafficPolicy**
- **services.public.sessionAffinity**
- **services.public.ports.sftp.port**
- **services.public.ports.ftp.port**
- **services.public.ports.ftp.enablePassiveRange**, boolean

Volumes

Volumes are defined by a name and can have different configuration keys based on their type. Volumes can be configured by setting:

- **volumes[idx].name**
- **volumes[idx].<config key>**

Volumes mounts can be configured by setting:

- **volumeMounts[idx].name**
- **volumeMounts[idx].mountPath**
- **volumeMounts[idx].readOnly**, true/false, optional

idx starts from 0.

Suppose you have a Secret named **encryption-key** like this

apiVersion: v1

data:

masterkey: YW1haDRvb3c5eGFleGFleXUwZWVYNG90OWFldGVlcHUK

kind: Secret

...

and you want to mount it as volume by adding these sections to your Kubernetes deployment

volumes:

- *name: master-key*

secret:

secretName: encryption-key






volumeMounts:

- *name: master-key*

mountPath: "/etc/sftpggo/masterkey"

readOnly: true

Here is the resulting configuration.

volumes[0].name	master-key	
volumes[0].secret.secretName	encryption-key	
volumeMounts[0].name	master-key	
volumeMounts[0].mountPath	/etc/sftpggo/masterkey	
volumeMounts[0].readOnly	true	

If you login to the pod you can see that the volume is mounted as expected.

```
sftpgo@sftpgo-65787577c5-gdv4z:~$ cat /etc/sftpggo/masterkey/masterkey  
amah4oow9xaexayu0eeX4ot9aeteepu
```

Troubleshooting and support

If you need support, please send an email explaining your issue to support@sftpgo.com and always include relevant logs and your Marketplace subscription ID.

Please note that the resource based billing includes basic email support for configuration customization and basic troubleshooting, we'll answer questions about SFTPGo, not Azure services and/or Kubernetes configurations.

If you require more advanced support check our pricing

<https://sftpgo.com/#pricing>