Table of Contents

2
2
4
rets5
figMaps5

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

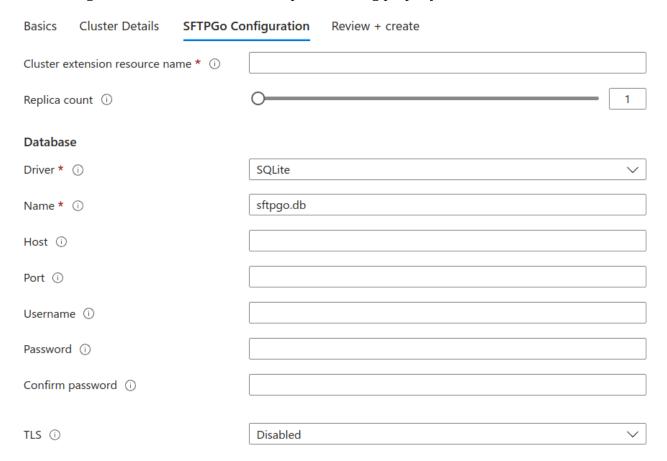
To deploy from the <u>markeplace page</u>:

- 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.



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.

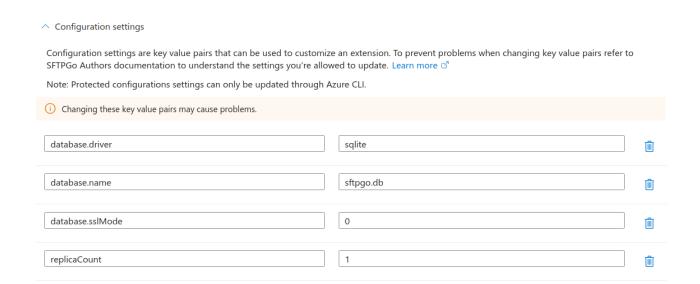
https://learn.microsoft.com/en-us/cli/azure/aks?view=azure-cli-latest#az-aks-get-credentials

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.



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*
- **serviceAccount.annotations**, array of annotations. Each array must contain a struct with a key and a values. Example: serviceAccount.annotations[0].key="....", serviceAccount.annotations[0].value="...."
- **podAnnotations**, array of annotations. Each array must contain a struct with a key and a values. Example: podAnnotations[0].key="....", podAnnotations[0].value="...."

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 <u>more</u>.

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

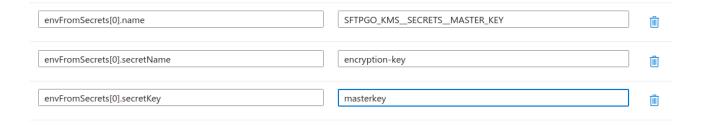
data:

masterkey: YW1haDRvb3c5eGFleGFleXUwZWVYNG90OWFldGVlcHUK

kind: Secret

. . .

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



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:

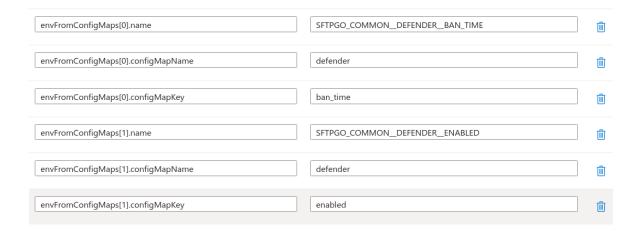
ban_time: "60"

enabled: "1"

kind: ConfigMap

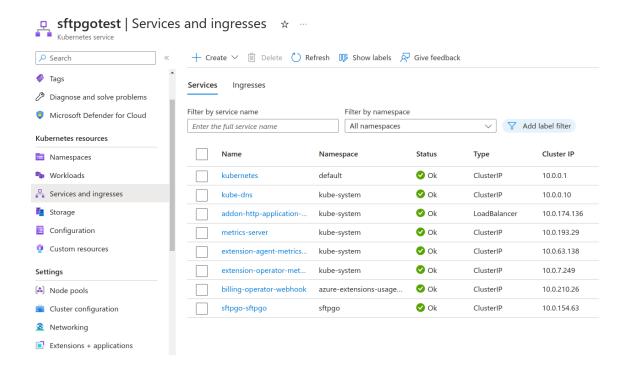
•••

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



Services

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



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

- service.type, default ClusterIP. Learn more.
- **service.annotations**, array of annotations. Each array must contain a struct with a key and a values. Example: service.annotations[0].key="....", service.annotations[0].value="...."
- 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
- **service.externalTrafficPolicy**, default unset. Learn more.
- **service.sessionAffinity**, default unset. Learn <u>more</u>.

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.annotations
- 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/sftpgo/masterkey" readOnly: true Here is the resulting configuration. volumes[0].name master-key Ŵ volumes[0].secret.secretName encryption-key Ŵ

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

sftpgo@sftpgo-65787577c5-gdv4z:~\$ cat /etc/sftpgo/masterkey/masterkey amah4oow9xaexaeyu0eeX4ot9aeteepu

true

master-key

/etc/sftpgo/masterkey

圃

圃

ŵ

Security Context

volumeMounts[0].name

volumeMounts[0].mountPath

volumeMounts[0].readOnly

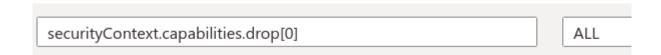
SFTPGo runs using 1000 as UID/GID. You can configure <u>pod</u> and <u>container</u> security contexts.

Pod security context can be configured using the *podSecurityContext* configuration key. For example:

• **podSecurityContext.fsGroup**, by default the fsGroup is set to 1000

Container security context can be configured using the *securityContext* configuration key. For example:

securityContext.capabilities.drop[0]



By default no container security context is set.

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