

Sino Control Hub on Azure Container Apps

For companies on Microsoft 365: the hub as one container app, its database on Azure Database for PostgreSQL, secrets in Key Vault, HTTPS on your domain
Sino Privacy Vault (<https://sinosecure.app>)

These steps have not yet been checked against a live Azure subscription. Tell us anything that differs.

1. Image and Resources

```
RG=sino-hub; LOC=westeurope; ACR=sinohub$RANDOM
az group create -n $RG -l $LOC
az acr create -g $RG -n $ACR --sku Basic
az acr login -n $ACR
docker tag sino-hub:<version> $ACR.azurecr.io/sino-hub:<version>
docker push $ACR.azurecr.io/sino-hub:<version>
```

The image (`sino-hub:<version>`) comes from Sino with early access.

2. Postgres

```
az postgres flexible-server create -g $RG -n sino-hub-db -l $LOC --tier
Burstable --sku-name Standard_B1ms \
  --version 16 --admin-user sinoadmin --admin-password '<strong
password>' --public-access None
az postgres flexible-server db create -g $RG -s sino-hub-db -d sino_hub
```

Put the server in the same virtual network as the Container Apps environment (or use a private endpoint). Then make a `sino_hub` role that owns the `sino_hub` database. The

address is `jdbc:postgresql://sino-hub-db.postgres.database.azure.com:5432/sino_hub?sslmode=require`.

Prefer MySQL? Use Azure Database for MySQL flexible server (`az mysql flexible-server create`) instead, with a `jdbc:mysql://<host>:3306/sino_hub?sslMode=verify-full` address (see the *Deployment Guide*, section 5). The rest is the same.

3. Secrets in Key Vault

```
az keyvault create -g $RG -n sino-hub-kv-$RANDOM --enable-rbac-authorization
az keyvault secret set --vault-name <kv> -n master-key --value "$(openssl rand -base64 32)"
az keyvault secret set --vault-name <kv> -n db-password --value '<sino_hub role password>'
```

Back up the master key outside Azure as well. Without it the database cannot be opened.

4. The Container App

```
az containerapp env create -g $RG -n sino-hub-env -l $LOC --infrastructure-subnet-resource-id <subnet id>
az containerapp create -g $RG -n sino-hub --environment sino-hub-env \
  --image $ACR.azurecr.io/sino-hub:<version> --registry-server $ACR.azurecr.io --registry-identity system \
  --system-assigned --ingress external --target-port 8080 --transport http \
  --min-replicas 1 --max-replicas 1 --cpu 0.5 --memory 1Gi
```

Give the app's identity the **Key Vault Secrets User** role on the vault, then wire the secrets and settings:

```

az containerapp secret set -g $RG -n sino-hub \
  --secrets master-
key=keyvaultref:https://<kv>.vault.azure.net/secrets/master-
key,identityref:system \
  db-
password=keyvaultref:https://<kv>.vault.azure.net/secrets/db-
password,identityref:system
az containerapp update -g $RG -n sino-hub --set-env-vars \
  SINO_HUB_MASTER_KEY=secretref:master-key
SINO_HUB_DATABASE_PASSWORD=secretref:db-password \
  SINO_HUB_DATABASE_URL='jdbc:postgresql://sino-hub-
db.postgres.database.azure.com:5432/sino_hub?sslmode=require' \
  SINO_HUB_DATABASE_USER=sino_hub SINO_HUB_TRUST_PROXY=true

```

- **Keep exactly one replica** (min and max 1). The hub's timers and its setup code live in its process.
- **Use single revision mode** (the default), so an update replaces the hub rather than running two side by side.
- Add a liveness and a readiness probe on `GET /healthz`, port 8080, in the app's YAML (`az containerapp update --yaml`).
- The ingress appends the client's address to `X-Forwarded-For`, so `SINO_HUB_TRUST_PROXY=true` is right here.
- Container Apps allows a request 240 seconds, longer than the 3 minutes a PC's stream lasts.

5. Your Domain

```

az containerapp hostname add -g $RG -n sino-hub --hostname
hub.example.com
az containerapp hostname bind -g $RG -n sino-hub --hostname
hub.example.com --environment sino-hub-env --validation-method CNAME

```

This address is permanent: every device keeps it. Then read the setup code and claim the hub:

```
az containerapp logs show -g $RG -n sino-hub --tail 100 | grep "setup code"
```

Open the address, enter the code and register your passkey, then sign the hub's certificate in the Sino app (*Deployment Guide*, section 3, step 5).

6. SQLite on Azure Files Instead of Postgres

This is possible for a very small hub. Mount an Azure Files share at `/data` and leave out the database settings. SQLite's file locking over SMB is fragile, though: keep one replica, use the `nobrl` mount option, and back up the share. The hub keeps WAL's index in its own memory rather than in a shared file, so WAL works there. Postgres is the safer choice.

7. Storage and Sign-In

Add the company's Azure Blob storage in the console, under **Storage**, with either:

- **The account key.** The hub signs service SAS tokens.
- **An Entra app registration with a client secret** (preferred). The hub signs user delegation SAS tokens, so no account key is stored. Give the app **Storage Blob Data Contributor** on the storage account. That role covers the delegation key and making each vault's container (`sinov-...`) and each shared folder's (`sinos-...`).

Entra ID sign-in for the console and SCIM are set up in the console too (*Deployment Guide*, sections 11 and 12). Neither needs anything from the container app beyond outbound HTTPS.