Manage Environment Secrets

PREREQUISITE

You must have a running app from the Deploy Your First App tutorial before continuing.

In this guide, you'll learn how to store sensitive values in an encrypted store and inject them into your application as environment variables. Instead of hardcoding secrets in your deployment configuration, you'll use an **ExternalSecret** that automatically syncs values from your secret store into your Kubernetes pod.

We'll add a new secret alongside the `GREETING_MESSAGE` you set up in the previous tutorial.

1. Create a secret

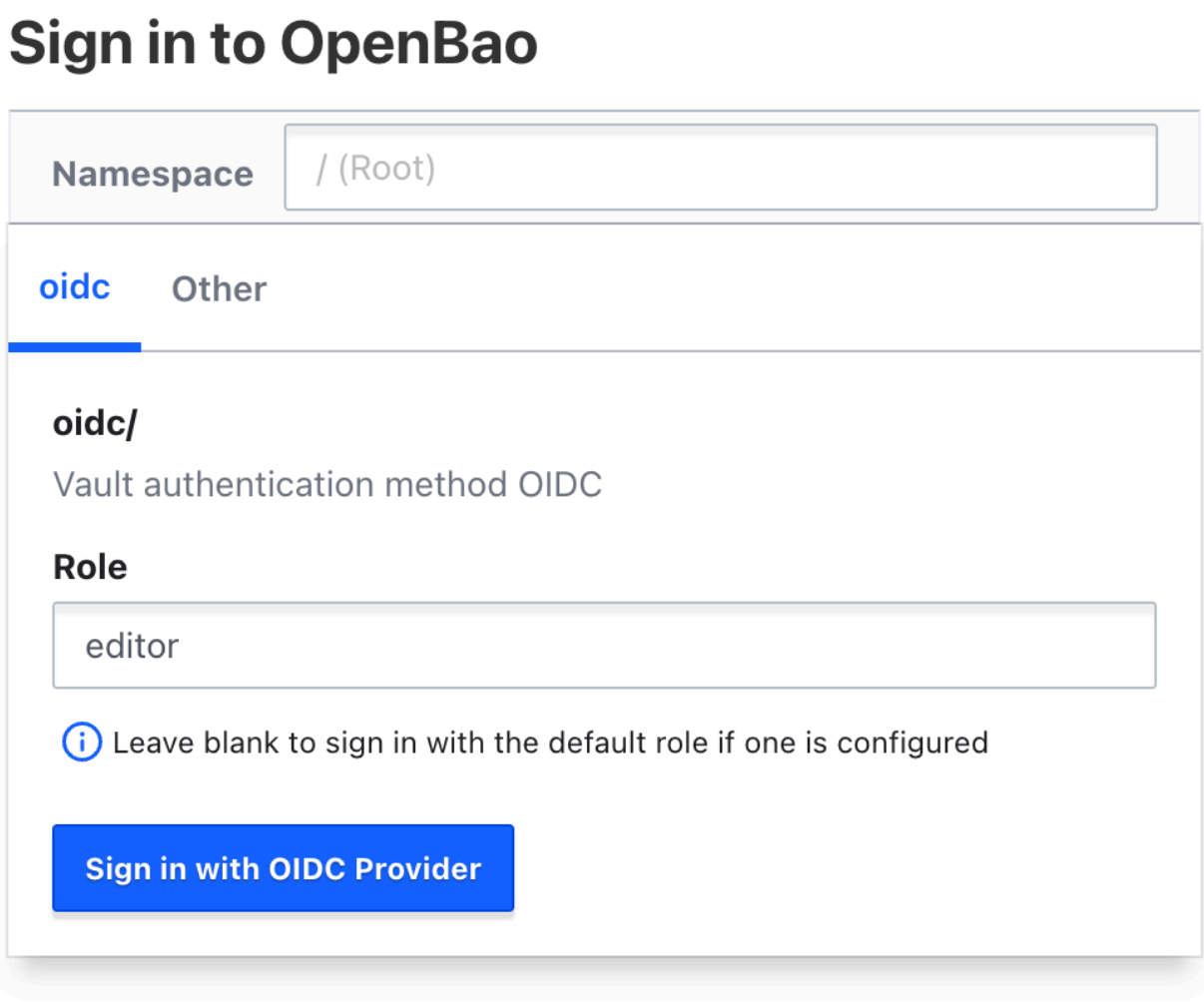
INFO

Your cluster information page has a link to the Encrypted Secret Store:

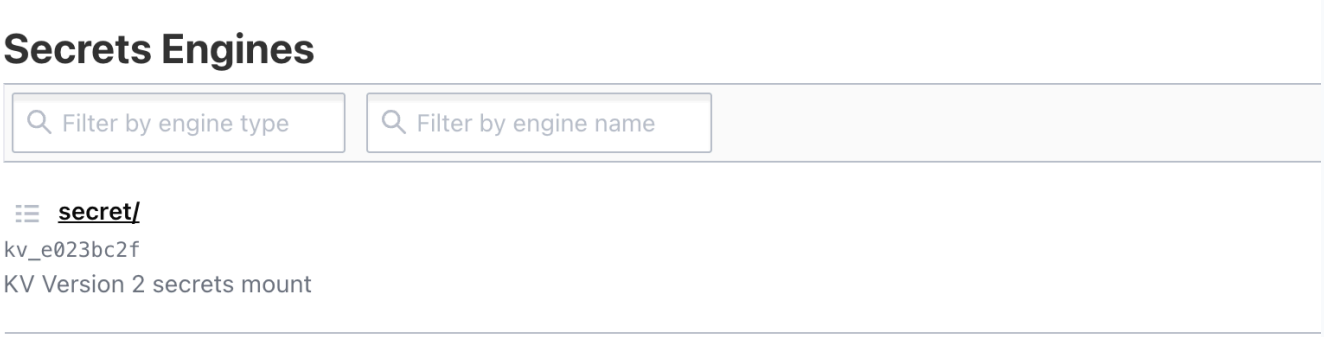
`https://cluster-info.nonprod.antoniostacos.ongluEops.com`

*If the domain above doesn't look right, update your **Captain Domain** in the top navigation bar.*

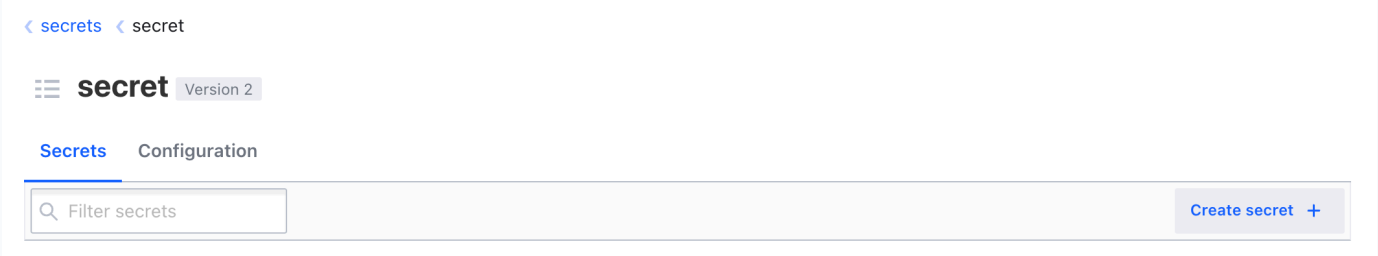
From the cluster info home page, click the `Link` for **Secrets Management** to open the secret store. For the role, enter **editor** (you need read/write access), then click **Sign in with OIDC Provider**.



Once logged in, select the **secret/** mount point.

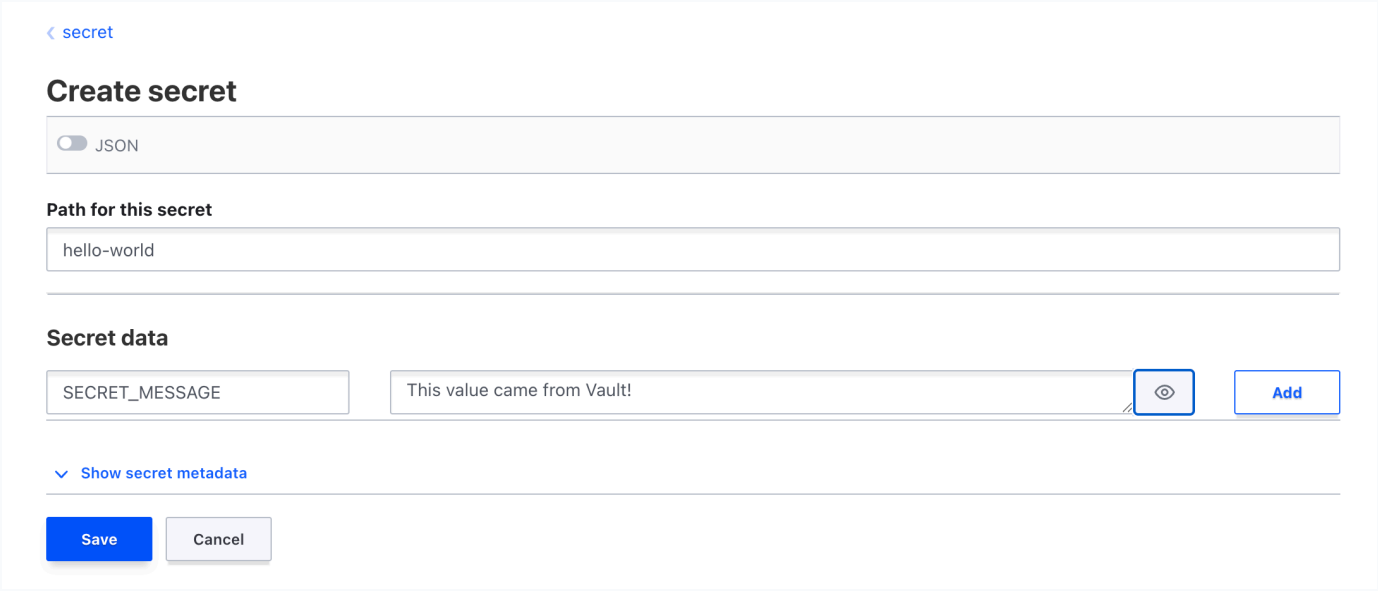


Click **Create secret +**

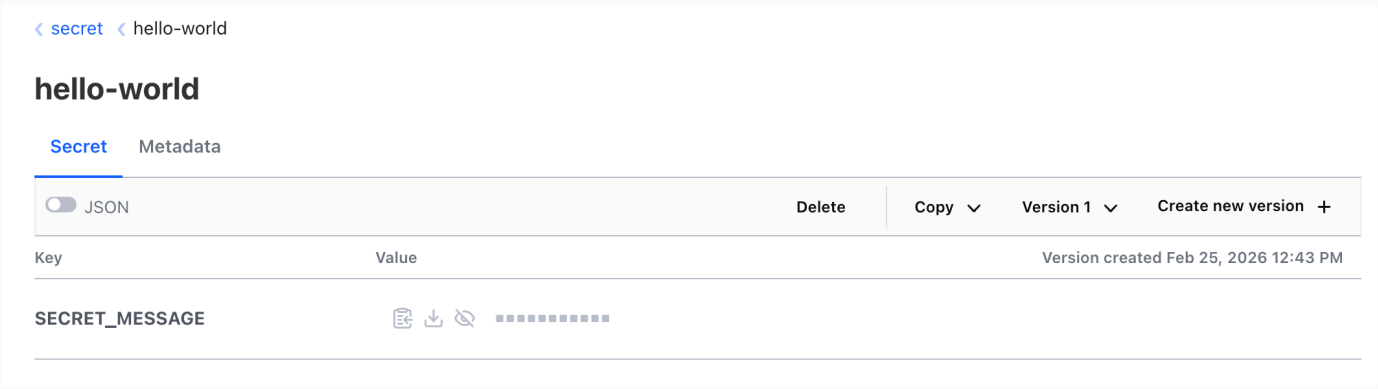


Enter the details for your secret:

- **Path for this secret:** Use your app name (e.g., `hello-world`)
- **Key:** `SECRET_MESSAGE` (case sensitive)
- **Value:** `This value came from your Secret Store!`



Click **Save**. You should see a confirmation page:



2. Add the ExternalSecret to your deployment

Open `envs/prod/values.yaml` in your deployment-configurations repository — the same file from the previous tutorial:

```
deployment-configurations
└── apps
    └── hello-world
        ├── base
        │   └── base-values.yaml
        └── envs
            └── prod
                └── values.yaml   ← edit this file
```

Add the `externalSecret` block to the bottom of the file. Here's the complete updated file:

apps/hello-world/envs/prod/values.yaml

```yaml
deployment:
  enabled: true
  replicas: 1
  envVariables:
    - name: GREETING_MESSAGE
      value: "Hello, World!"

service:
  enabled: true

ingress:
  enabled: true
  ingressClassName: public-traefik
  entries:
    - name: public
      hosts:
        - hostname: '{{ include "app.name" . }}.apps.{{
.Values.captain_domain }}'

externalSecret:
  enabled: true
  secrets:
    hello-world:
      dataFrom:
        key: secret/hello-world
```

HOW IT WORKS

- `externalSecret.enabled: true` turns on the ExternalSecret integration.
- `secrets.hello-world` creates an ExternalSecret named after your app.
- `dataFrom.key: secret/hello-world` tells it to pull **all** key-value pairs from the secret path `secret/hello-world` and inject them as environment variables.

The `GREETING_MESSAGE` from `envVariables` remains unchanged — it's a static value defined in your config. The secret (`SECRET_MESSAGE`) is injected alongside it automatically.

3. Deploy

Commit the changes and push. ArgoCD will sync the update in ~3 minutes.

4. Verify

Visit `https://hello-world-prod.apps.nonprod.antoniostacos.onglueops.com/?env=true` and look for both:

- `GREETING_MESSAGE=Hello, World!` — your static environment variable
- `SECRET_MESSAGE=This value came from your Secret Store!` — injected via ExternalSecret

TIP

If the secret doesn't appear immediately, give it a moment — the ExternalSecret controller refreshes every few seconds. You can also check the ExternalSecret resource status in the ArgoCD dashboard.

Key concepts

Concept	Description
ExternalSecret	A Kubernetes resource that syncs secrets from an external provider (e.g. OpenBao) into a Kubernetes Secret.

Concept	Description
`dataFrom`	Pulls all key-value pairs from a secret path — no need to list each key individually.
`envVariables`	Static env vars defined directly in your values file. Good for non-sensitive config.
OpenBao	The secrets management system. Secrets are stored at paths like `secret/hello-world`.

Next steps

- Traefik Ingress & Routing — Explore advanced routing patterns with Traefik.