Deploy Your First App

In this guide you'll deploy a pre-built web application onto the GlueOps platform. No custom code, Dockerfile, or CI pipeline is needed — you'll configure two small YAML files and push them to your deployment-configurations repository. ArgoCD handles the rest.

By the end, you'll have a running app with a public URL and a custom environment variable.

Prerequisites

- Access to your team's **deployment-configurations** repository. Your Platform Administrator will provide the repo name and location.
- Your cluster information page: `https://cluster-info.nonprod.antoniostacos.onglueops.com`

TIP

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

1. Create the app directory structure

Inside your deployment-configurations repository, create the following directory structure:

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

- `base/base-values.yaml` — Configuration shared across all environments (image, port).

- `envs/prod/values.yaml` — Environment-specific settings (replicas, ingress, env vars).

2. Add base values

This file defines the container image. We're using `traefik/whoami`, a lightweight web server that displays request and system information — perfect for testing deployments.

apps/hello-world/base/base-values.yaml

```yaml
image:
  registry: docker.io
  repository: traefik/whoami
  tag: latest
  port: 80
```

3. Add environment values

This file enables the Deployment, Service, and Ingress, and sets a `GREETING_MESSAGE` environment variable:

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 }}'
```

HOW THE HOSTNAME WORKS

The `hostname` uses Helm template expressions that are resolved automatically at deploy time:

- `{{ include "app.name" . }}` → the app's directory name plus environment suffix (e.g., `hello-world-prod`)
- `{{ .Values.captain_domain }}` → your cluster's domain, injected by the platform

You don't need to hardcode any values — the platform fills them in for you.

4. Deploy

Commit both files and push to your deployment-configurations repository. ArgoCD will detect the changes and deploy your app automatically.

- **ArgoCD sync:** ~3 minutes
- **DNS propagation:** ~2 additional minutes

You can monitor the deployment status from the ArgoCD dashboard, accessible via your `cluster information page`.

5. Verify

Once deployed, open your browser and visit:

`https://hello-world-prod.apps.nonprod.antoniostacos.onglueops.com`

You should see the `whoami` response page showing hostname, IP addresses, and HTTP request details.

To confirm your `GREETING_MESSAGE` environment variable was injected, visit:

`https://hello-world-prod.apps.nonprod.antoniostacos.onglueops.com/?env=true`

Look for `GREETING_MESSAGE=Hello, World!` in the output.

Key concepts

Concept	Description
Base values	Shared config (image, port) that applies to every environment.
Environment values	Per-environment overrides (replicas, ingress, env vars) in `envs/<env>/values.yaml`.
captain_domain	Your cluster's domain — injected automatically by the platform. Never hardcode it.
ArgoCD	Watches your deployment-configurations repo and syncs changes to the cluster automatically.

Next steps

- Add Secrets — Pull sensitive configuration from your secret store instead of hardcoding values.
- Traefik Ingress & Routing — Explore advanced routing patterns: path-based routing, middleware, rate limiting, and more.