Deploying Apps — Reference

This page is the detailed reference for custom apps. For the push-to-deploy overview, see **Deploying Your App** in Getting Started.

Deploy from the portal

1. Go to portal → **Apps**.
2. Fill in the deploy form: **App Name** (lowercase, alphanumeric + hyphens, e.g. `my-api`), **Image** (full registry path, e.g. `registry.compacthive.dev/<project>/my-api:latest`), **Port** (the port your app listens on inside the container).
3. Click **Deploy**.

CompactHive creates a Kubernetes Deployment, Service, and Ingress for your app, reachable at `https://<your-slug>-<app-name>.compacthive.dev`.

Resource	Purpose
Deployment	Runs your container (1 replica by default)
Service	ClusterIP service on your specified port
Ingress	Public URL via the `*.compacthive.dev` wildcard

Resource defaults

Custom apps get a default **request** of ~25m CPU / 64Mi memory if you don't set one. If you don't set an explicit memory/CPU **limit**, the namespace's default limit applies (200m CPU / 256Mi memory) — raise it on the Apps page if your app needs more headroom, within your tier's overall quota.

App limits per tier

Tier	Max apps
Basic	2

Tier	Max apps
Team	5
Pro	10
Business	25
Enterprise	50

Update or delete a running app

On the Apps page, click your app to expand its details. You can update the **image** (triggers a rolling update), **replicas**, and **environment variables**. Deleting an app removes its Deployment, Service, and Ingress — registry images are not deleted (manage those separately from the Registry page).

End-to-end example

This walkthrough takes an empty repo to a running app. The same process works for any language with a Dockerfile — this example uses Node.js.

`server.js`

```
const http = require('http');
const port = process.env.PORT || 3000;

const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/html' });
  res.end('<h1>Hello from CompactHive!</h1>');
});

server.listen(port, () => console.log(`Listening on :${port}`));
```

`Dockerfile`

```
FROM node:20-alpine
WORKDIR /app
COPY server.js .
EXPOSE 3000
CMD ["node", "server.js"]
```

1. Push `server.js`, `Dockerfile`, and a `.woodpecker.yaml` (see **CI/CD Pipelines**) to your Forgejo repo — or, on GitHub (beta), connect it and run Settings → "Set Up" to add CI.
2. Push. Woodpecker clones the repo, builds the image with Kaniko, and pushes it to `registry.compacthive.dev/<project>/<repo>:latest`.
3. Go to portal → **Apps**, enter the app name, the image above, and port `3000`, then click **Deploy**.
4. Your app is live at `https://<your-slug>-<app-name>.compacthive.dev`. Every subsequent push rebuilds and (with auto-deploy on) redeploys it automatically.

Revision #1
Created 2026-11-03 14:21:32 UTC by Docs Sync
Updated 2026-11-03 14:21:32 UTC by Docs Sync