



OpenRun

# OpenRun: Internal Tools Deployment Platform

Platform for securely deploying internal tools across your team. Deploy web apps declaratively, on a single machine or on Kubernetes.

```
● ● ●  
→ ~ curl -sSL https://openrun.dev/install.sh | sh  
# Installing OpenRun...  
→ ~ openrun server start &
```

## AUDIENCE

Developers & DevOps Engineers

[openrun.dev](https://openrun.dev)

[github.com/openrundev](https://github.com/openrundev)

## INTRODUCTION

# What is OpenRun?

A self-hosted platform that makes declarative web-app deployment simple. Built for teams to deploy internal tools.



Runs on **Docker/Podman** or **Kubernetes**.  
Managed via **Git**.



## Open Source

Self-hosted alternative to solutions like **Google Cloud Run** and **Heroku**. Run your internal tools on your own infrastructure.



## Declarative GitOps

Zero "ClickOps". Define apps and configurations in Git. OpenRun reconciles state automatically.



## Simplicity & Reliability

Single binary architecture with minimal dependencies. No third-party web servers or complex mesh required.



## Focused Scope

Not a full PaaS. Focus is deploying your web apps while maintaining operational simplicity.

# Key Features



## Declarative GitOps

Define your desired state in Git: OpenRun reconciles automatically. Add or update apps purely through config changes—**no manual "ClickOps"** or complex deployment scripts required.



## Operational Simplicity

Distributed as a **single binary** with flexible runtime support. Container orchestration using **Docker/Podman** or with **Kubernetes**.



## Security & Governance

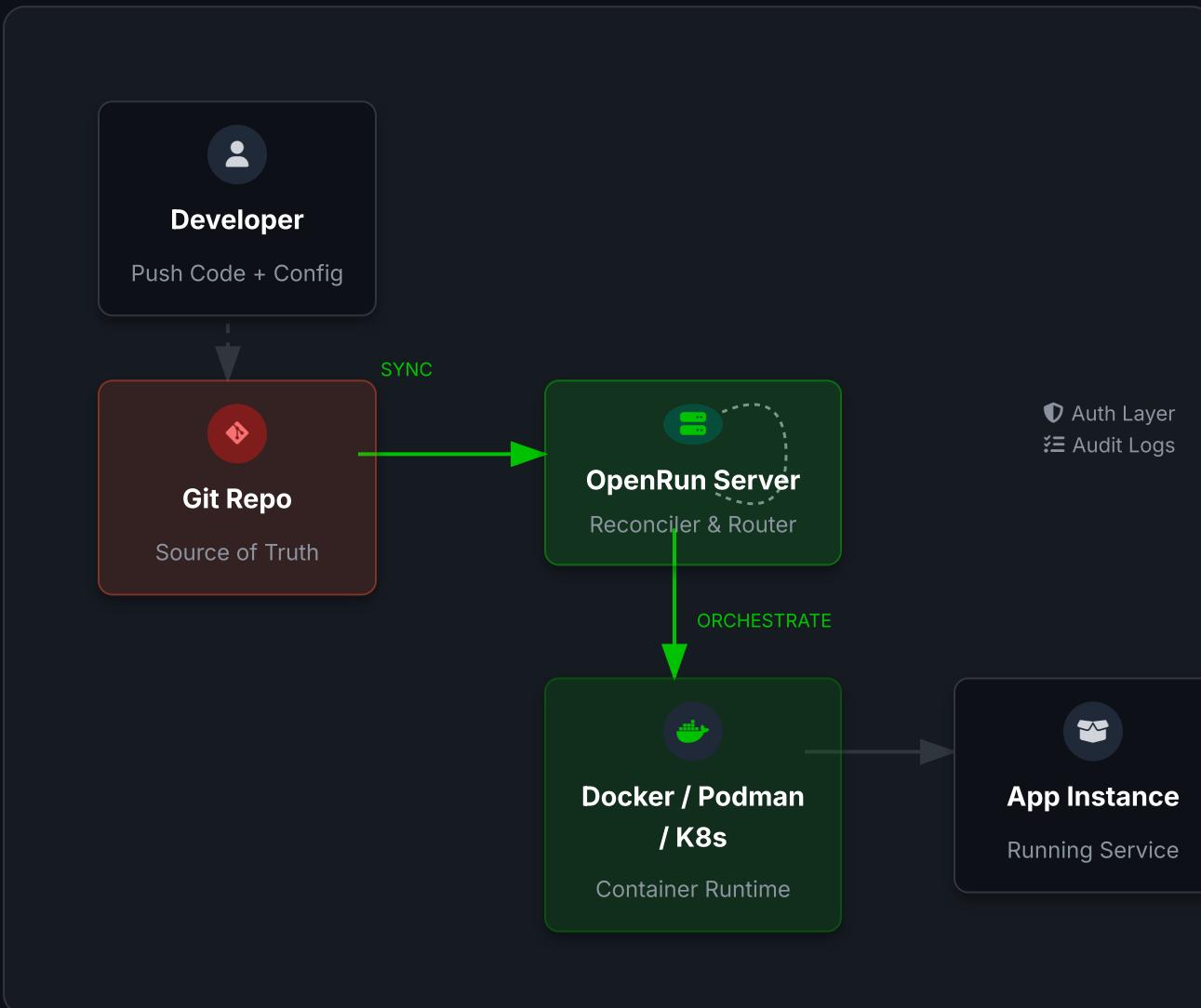
OAuth/OIDC/SAML authentication with **RBAC** without changing app source code. Includes comprehensive API audit logging for compliance, ensuring full traceability of actions.



## Efficient Resource Utilization

Save resources with **scale-to-zero** capabilities for idle apps. Run workloads on your own hardware, VMs, or **Kubernetes clusters** within your own network control.

# Technical Architecture



## Architecture Flow

### Single Binary Orchestration

The core server is a single Go binary that manages the entire lifecycle. It acts as both the controller for reconciliation and the router for incoming requests.

### Flexible Runtime Support

Deploys directly to Docker, Podman, or Kubernetes. Single node deployment uses an embedded SQLite database, Kubernetes deployment uses Postgres for metadata.

### Git-Based State Reconciliation

Periodically pulls configuration from the specified Git repository, diffs the desired state against the running containers/pods and applies changes.

### Built-in Governance Layer

Authentication (OAuth/OIDC/SAML) and auditing are handled at the platform layer, shielding applications from security complexity.

# Installation & Setup

## Prerequisites

### OPTION A: SINGLE BINARY

- ✓ Linux/Mac/Windows host with Docker/Podman

### OPTION B: KUBERNETES

- ✓ K8s cluster + Helm

### OPTION C: TERRAFORM

- ✓ Terraform install

Network access to GitHub

## PRO TIP

For production setups, consider externalizing the Postgres database and container registry for better persistence, regardless of deployment method.

### Option A: Single Binary

RECOMMENDED START

```
curl -sSL https://openrun.dev/install.sh | sh  
openrun server start &
```

### Option B: Kubernetes (Helm)

SCALABLE

```
helm repo add openrun https://openrundev.github.io/charts/  
helm install openrun openrun/openrun -n openrun --create-namespace
```

### Option C: Terraform (IaC)

INFRA AS CODE

```
terraform init && terraform apply  
Spins up OpenRun with an EKS cluster, RDS database and ECR registry
```

# GitOps in Action

## 1. Schedule the Sync

```
● ● ● bash  
→ openrun sync schedule --approve --promote \  
github.com/openrundev/openrun/examples/utils.star
```



## 2. Automated Reconciliation Pipeline

01



### Watch Repo

OpenRun monitors the specified Git URL for any commits or configuration changes.

02



### Read Config

Fetches the latest app configuration and application source code from the repository.

03



### Provision

Builds images, creates new app instances, or updates existing ones to match desired state.

04



### Secure

Automatically applies configured OAuth/OIDC policies and enables API audit logging.

05



### Reconcile

Continuous loop ensures the live environment always matches the Git configuration.

# Declarative Config Comparison

## OpenRun Config

STARLARK

```
# Set resource limits (optional)
limits = {"cpus": "1", "memory": "512m"}

# Streamlit App Declaration
app("/misc/streamlit_example",
    "github.com/streamlit/streamlit-example",
    git_branch="master",
    spec="python-streamlit",
    container_opts=limits)

# FastHTML App Declaration
app("fasthtml:",
    "github.com/AnswerDotAI/fasthtml/examples",
    spec="python-fasthtml",
    params={"APP_MODULE": "basic_ws:app"},
    container_opts=limits)

# Complete config for two apps. Defines where to get the source
code, how to build the image, how to route requests and what
resource limits to set
```

VS

## Kubernetes Manifests

YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: streamlit-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: streamlit
  template:
    metadata:
      labels:
        app: streamlit
    spec:
      containers:
      - resources:
          limits:
            memory: "512Mi"
            cpu: "1"
# Partial k8s config for one app. Jenkins (CI), ArgoCD,
CertManager, IDP etc require additional config.
```

# Comparison: Self-hosted Platforms

COMPARED WITH



Kamal



Coolify



Dokku

## OpenRun Advantages



### Declarative GitOps

No "ClickOps" or "CommandOps". New apps and updates are done with GitOps, ensuring version control and reproducibility.



### Built-in Security & Governance

Centralized OAuth/OIDC/SAML integration and API audit logging, without needing external identity proxies.



### Single Binary Architecture

Single binary. No dependency on third-party web servers like Traefik or Nginx to manage routing.



### Scale to Zero

Automatically scales inactive services down to zero, significantly reducing resource consumption.

## Trade-offs & Scope



### Not a Full PaaS

OpenRun focuses purely on stateless app deployment. It does not provision or manage stateful services like databases (Postgres, MySQL, Redis etc).



### No Docker Compose

Designed for single-container services. Multi-container orchestration via Docker Compose is not supported; use Kubernetes for complex scenarios.



### Focused Use Cases

Optimized specifically for web applications and internal tools, rather than general-purpose infrastructure management.

# Comparison: Cloud Services

COMPARED WITH

G Google Cloud Run

aws AWS App Runner

## OpenRun Advantages



### Network Control & Data Locality

Deploy on your own hardware in your private network. Full control over network security and data residency.



### Cost Efficiency

Run high-performance or custom hardware workloads without the markup of managed cloud compute.



### Seamless Access Control

Add OAuth/OIDC/SAML authentication instantly via configuration, without modifying the application code itself.



### GitOps Native

Declarative configuration ensures your deployment pipeline is transparent and version-controlled, avoiding opaque cloud console settings.

## Considerations



### Infrastructure Management

You are responsible for the underlying hardware, OS updates, and uptime. There is no cloud provider SLA for the physical infrastructure.



### No Global Edge Network

Unlike global cloud services, you don't automatically get a worldwide CDN or multi-region failover unless you build it yourself.



### Integration Effort

Cloud-native conveniences (like managed DBs, SQS, Pub/Sub) are not built-in; you must integrate and host these auxiliary services separately.

# OpenRun vs. Complex DIY K8S Stacks

CONTEXT

✓ OpenRun On Kubernetes

vs. Custom ArgoCD Stacks

## Why Choose OpenRun?



### Deployment Flexibility

Run standalone on Docker/Podman for simplicity, or deploy on Kubernetes for scale.



### All-in-One Platform Layer

Replaces the need to glue together ArgoCD/Flux, IDP, Cert-Manager and OIDC providers. Get a PaaS experience.



### Better Developer Experience

Developers use a clean declarative config, not complex Kubernetes YAML or Helm charts, even when the underlying runtime is K8s.



### Instant Capabilities

Auth, Scale-to-Zero, and GitOps out of the box, rather than building a custom internal developer platform (IDP).

## Trade-offs & Scope



### Complex Orchestration

If you need advanced Service Mesh features (Istio/Linkerd) or complex multi-cluster networking, a custom K8s stack might be required.



### Custom Operators

OpenRun abstracts the runtime. If you heavily rely on custom Kubernetes Operators and CRDs for app logic, raw K8s is better.



### Granular Scheduling

Advanced node affinity, taints/tolerations, and custom pod anti-affinity rules are abstracted away in OpenRun's simplified config.

# Common Use Cases



## Rapid Web App Deployment

Empower data science and internal tool teams to deploy web apps with zero or low configuration. Removes the friction of infrastructure setup for framework-based applications.

Streamlit

Gradio

FastHTML

NiceGUI



## Operations Automation

Modernize your operational tooling. Replace heavy Jenkins or Rundeck jobs with lightweight OpenRun Actions to automate scripts, maintenance tasks, and runbook jobs.

Runbook Automation

Maintenance Scripts

Provisioning



## Secure Internal APIs

Expose REST APIs for internal consumption securely. Automatically layer OAuth/OIDC authentication and enforce audit logging without changing application code.

OAuth 2.0

OIDC / SAML

Audit Logs



## GitOps Standardization

Eliminate manual deployment toil and "ClickOps". Standardize all team deployments through a single Git repository, ensuring version history and peer review for all changes.

Git Flow

Version Control

Peer Review

# Common Use Cases — Individuals



## Zero-Config Local Dev

Replicate production environments locally instantly. Spin up dev environments that mirror your deployment target without wrestling with complex Docker Compose files or manual networking.

localhost

Dev/Prod Parity

Rapid Iteration



## Secure Personal Sharing

Host web apps for friends and family safely. Leverage built-in OAuth/OIDC integration to add secure login to your hobby projects without writing a single line of authentication code.

OAuth

Hobby/Vibe-Coded Apps

Secure Access



## Efficient Homelab

Perfect for running lightweight services on Raspberry Pi or old hardware. The "scale-to-zero" feature puts inactive apps to sleep, saving power and freeing up system resources.

Scale-to-Zero

Raspberry Pi

Green IT



## Master GitOps

The best way to learn modern DevOps patterns hands-on. Experience true GitOps workflows and declarative infrastructure management without the steep learning curve of Kubernetes.

Learning

GitOps

DevOps Skills

# Quick Start

## Demo Checkpoints

- ✓ Install platform
- ✓ Start server process
- ✓ Connect to Git repo
- ✓ Verify deployment

## GitOps Workflow

By scheduling a sync, OpenRun automatically pulls config from the repo. No manual "build" or "deploy" commands are needed for individual apps—the state is reconciled from Git.

## 1 Install & Start



```
→ ~ curl -sSL https://openrun.dev/install.sh | sh  
→ ~ openrun server start &
```

## 2 Schedule Git Sync



```
→ ~ openrun sync schedule --approve --promote github.com/openrundev/openrun/  
examples/utils.star  
[INFO] Sync scheduled successfully  
[INFO] Fetching config from github.com...  
[SUCCESS] 2 apps created, 1 updated.
```

## 3 Verify & Access

- Check running apps: `openrun app list`
- Apps are now live at configured domains (e.g., `app.localhost`)
- Updates to Git repo will automatically trigger new deploys

# Benefits Summary



## Developer Velocity

Accelerate delivery with GitOps-driven deployments. Push to Git and let OpenRun handle the rest, eliminating manual build steps and context switching.



## Operational Simplicity

A single binary architecture with minimal dependencies. No complex orchestration layers, no third-party web servers.



## Security & Compliance

Built-in OAuth/OIDC/SAML support and comprehensive API audit logging ensure your apps are secure and compliant by default, with zero code changes.



## Cost Efficiency

Run on your own commodity hardware and leverage "scale-to-zero" to minimize resource consumption when apps are idle. No per-request cloud billing.



## Portability

Standard Docker/Podman container runtime ensures your apps run anywhere. Migrate easily between on-prem servers and cloud VMs without lock-in.



## Focused Scope

Purpose-built for web apps and internal tools. Avoids the complexity of full PaaS solutions by doing one thing well.

# Conclusion & Resources

## WHEN TO CHOOSE OPENRUN



### Cloud-like Simplicity

You want the "Cloud Run" experience—stateless containers, auto-scaling—but on your own hardware or cheaper VPS.



### GitOps Preference

You prefer defining infrastructure as code in Git over manual "ClickOps" UI interactions or complex CLI scripting.



### Focused Scope

You don't need the platform to manage stateful databases (RDS/Postgres) or complex Docker Compose stacks.

## >\_ Install Now

Get started with a single command on any Linux host with Docker/Podman.

```
curl -sSL https://openrun.dev/install.sh | sh
```

 Website

 GitHub

## Next Steps

1

### Run the Quick Start

Follow the install guide and verify the server is running.

2

### Star the OpenRun Repo

Star [openrundev/openrun](https://github.com/openrundev/openrun)

3

### Deploy Your First App

Schedule a sync pointing to your repo and watch it go live!