BOOTS ON THE GROUND AI

AI Solutions for Small Business Owners

FLY.IO PLAYBOOK

Deploy Anywhere. Scale Everywhere.

Your complete guide to deploying on Fly.io.
From CLI commands to multi-region scaling,
this playbook ensures every deployment is
fast, reliable, and production-ready.

Built for teams that move fast.

BOOTS APPROVED

FLY.IO

Version 1.0 | February 12, 2026
Boots On The Ground AI | bootsagentai.com

TABLE OF CONTENTS

Master Fly.io deployment

WHY FLY.IO?

The new way to deploy

What is Fly.io?

Fly.io is a global application platform that runs your apps close to your users. Instead of managing servers in one region, Fly.io lets you deploy once and run everywhere—automatically routing users to the nearest server. It's like having a CDN for your entire app.

Traditional Hosting vs Fly.io

Aspect	Traditional Hosting	Fly.io
Deployment	Manual uploads, SSH access	Single command (fly deploy)
Scaling	Buy bigger servers or replicas manually	Auto-scaling built-in
Regions	Choose one location	Multi-region in seconds
Configuration	Complex config files, environment setup	Simple fly.toml + Dockerfile
Speed	Minutes to hours	Deploy in seconds
Databases	Manage your own PostgreSQL	Managed Fly Postgres
Monitoring	DIY logging and monitoring	fly logs and fly status included

BOOTS APPROVED

BOOTS SAYS: Small businesses spend money on infrastructure they don't need. Fly.io changes that. Pay for what you use, deploy anywhere, and never manage servers again. That's the Boots Standard.

10 GOLDEN RULES OF FLY.IO

Principles for success

1. **Always use flyctl CLI for deployments**
   Consistency and reproducibility come from the command line.

2. **Use fly.toml as your single source of truth**
   All config lives here. No secrets, no surprises.

3. **Deploy to regions closest to your users**
   Latency kills user experience. Region selection matters.

4. **Use Fly Postgres for databases**
   Managed, automated, simple. Don't reinvent the wheel.

5. **Store secrets with fly secrets set**
   Never hardcode credentials. Ever. Use fly secrets set instead.

6. **Use health checks to keep your app alive**
   Fly can't fix what it doesn't know is broken. Configure health checks.

7. **Scale with fly scale count**
   Don't hope for scaling. Command it. fly scale count controls your destiny.

8. **Use volumes for persistent storage**
   Machines are ephemeral. Volumes are permanent. Know the difference.

9. **Set up CI/CD with GitHub Actions**
   Automate deployments. Make fly deploy part of your workflow.

10 Monitor with fly logs and fly status always

You can't manage what you don't measure. Check logs, check status.

CLI INSTALLATION & SETUP

Get flyctl running

Install flyctl

macOS / Linux:

```
curl -L https://fly.io/install.sh | sh
```

Windows (PowerShell):

```
powershell -Command "iwr https://fly.io/install.ps1 -useb | iex"
```

Homebrew (macOS):

```
brew install flyctl
```

TIP

BOOTS EXPLAINS: flyctl is your CLI tool. It controls everything—deployments, scaling, secrets, databases. Get comfortable with it. It's your new best friend.

Essential Commands

- **fly auth login** – Authenticate with your Fly.io account
- **fly auth signup** – Create a new Fly.io account
- **fly launch** – Initialize a new app (generates fly.toml + Dockerfile)
- **fly deploy** – Build and deploy your app
- **fly status** – Check app status and running machines
- **fly logs** – View live application logs
- **fly ssh console** – SSH into your running machine for debugging
- **fly secrets set KEY=value** – Set environment variables securely
- **fly scale count 2** – Scale to 2 machines
- **fly regions add ord** – Add Chicago region to your app
- **fly postgres create** – Create a managed PostgreSQL database
- **fly destroy** – Delete your app (careful!)

START HERE

BOOTS EXPLAINS: Start with fly auth login. Then fly launch to initialize your app. fly deploy is your daily command. fly logs is your friend when things break. Learn these first.

THE FLY.TOML FILE

Your app's configuration home

Annotated Template (Next.js App)

```
app = "my-nextjs-app" primary_region = "ord" [build] [build.args] NEXT_PUBLIC_API_URL =
"https://api.example.com" [http_service] internal_port = 3000 force_https = true
auto_stop_machines = true auto_start_machines = true min_machines_running = 1 [[vm]]
cpu_kind = "shared" cpus = 1 memory_mb = 512
```

Key Configuration Sections:

app: Your app name on Fly.io (unique across the platform)

primary_region: Where your app deploys first (ord=Chicago, iad=Virginia, lax=LA, etc.)

[build.args]: Non-secret build variables passed to Docker

internal_port: What port your app listens on inside the machine

force_https: Always use HTTPS (Fly handles the certificate)

auto_stop_machines: Stop machines when they're idle (save money)

min_machines_running: Minimum machines always on (prevents cold starts)

[[vm]]: Machine type and size (shared-cpu-1x = cheap, performance = fast)

DANGER

IMPORTANT: fly.toml is checked into git. It contains NO secrets. Secrets go in fly secrets set, not fly.toml.

DOCKERFILE FOR FLY.IO

Multi-stage build for Next.js

```
# Builder stage FROM node:18-alpine AS builder WORKDIR /app COPY package*.json ./ RUN npm ci COPY . . RUN npm run build # Runner stage FROM node:18-alpine WORKDIR /app ENV NODE_ENV=production COPY --from=builder /app/node_modules ./node_modules COPY --from=builder /app/.next ./.next COPY --from=builder /app/public ./public COPY --from=builder /app/package*.json ./ EXPOSE 3000 CMD ["npm", "start"]
```

Why Multi-Stage?

The builder stage compiles your app and installs dependencies. The runner stage copies only what you need. This keeps your final image small and fast.

Key Points:

- **node:18-alpine:** Lightweight Node image
- **npm ci:** Clean install (better than npm install for CI)
- **EXPOSE 3000:** Matches internal_port in fly.toml
- **CMD:** The command that starts your app

DEPLOYMENT WORKFLOW

Your step-by-step deployment

1. **Write your code locally**
   Develop normally on your machine.
2. **Test locally with npm run dev**
   Make sure it works before you push.
3. **Run fly launch (first time only)**
   Generates fly.toml and Dockerfile.
4. **Run fly deploy (every deploy)**
   Builds, tests, and deploys your app.
5. **Check fly status**
   Confirm deployment succeeded.
6. **View fly logs**
   Watch logs for errors or warnings.
7. **Test your live URL**
   Click the link and verify everything works.
8. **Scale if needed**
   Use fly scale count when traffic demands it.

AUTOMATION

PRO TIP: Automate this workflow with GitHub Actions. Set up a CI/CD pipeline so fly deploy runs automatically when you push to main. Fire and forget deployment.

SECRETS & ENVIRONMENT VARIABLES

Keep secrets secret

Setting Secrets

fly secrets set DATABASE_URL=postgres://user:pass@host/db – Set a secret

fly secrets set API_KEY=sk_live_xxxxx – Set your API key

fly secrets list – See what secrets are set (values hidden)

fly secrets unset OLD_KEY – Remove a secret

DANGER

NEVER PUT SECRETS IN FLY.TOML OR DOCKERFILE. Use fly secrets set for everything sensitive: API keys, database passwords, tokens, credentials.

Secrets vs Config

Item	Where It Goes	Example
Database URL	fly secrets set	DATABASE_URL=postgres://...
API Keys	fly secrets set	STRIPE_KEY=sk_live_...
Auth Tokens	fly secrets set	JWT_SECRET=...
Build args (non-secret)	fly.toml [build.args]	NEXT_PUBLIC_API_URL=https://...
App version	fly.toml [build.args]	VERSION=1.0.0
Feature flags (public)	fly.toml [build.args]	FEATURE_BETA=true

SCALING & REGIONS

Grow without limits

Scaling Machines

fly scale show – See current number of machines

fly scale count 2 – Scale to 2 machines

fly scale count 5 – Scale to 5 machines

fly scale vm shared-cpu-1x – Change VM type (shared vs performance)

fly scale vm --vm-memory 512 – Change memory to 512MB

Adding Regions

fly regions list – See all available regions

fly regions add ord – Add Chicago

fly regions add lax ord iad – Add LA, Chicago, Virginia

fly regions remove lax – Remove LA

Common Regions

Code	City	Region	Best For
ord	Chicago	North America (Central)	US Midwest
iad	Ashburn, Virginia	North America (East)	US East
lax	Los Angeles	North America (West)	US West
sjc	San Jose	North America (West)	Tech Companies
yyz	Toronto	Canada	Canadian Users
lhr	London	Europe	UK/EU Users
fra	Frankfurt	Europe	Central EU
nrt	Tokyo	Asia	Asia/Pacific
syd	Sydney	Australia	Australia/NZ

BOOTS APPROVED

BOOTS SAYS: Start with one region closest to your users. Add regions when your traffic demands it. Multi-region deployment is your competitive advantage—use it.

TROUBLESHOOTING

When things go wrong

Problem: Deploy fails

Solution: Check Dockerfile syntax. Run docker build locally to test. Check fly logs for build errors.

Problem: App not responding

Solution: Check health checks in fly.toml. Verify internal_port matches your app. Check fly logs.

Problem: 500 errors in logs

Solution: Check DATABASE_URL is set with fly secrets set. Verify Postgres is running with fly status.

Problem: Out of memory

Solution: App consuming too much? Use fly scale vm with more memory. Or optimize your code.

Problem: SSL/HTTPS issues

Solution: Fly handles certificates automatically. Ensure force_https=true in fly.toml.

Problem: Slow cold starts

Solution: Set min_machines_running=1 in fly.toml to keep at least one machine warm.

Problem: Database connection timeouts

Solution: Check if Postgres is in same region. Use fly postgres attach to create one.

Problem: Machines crashing repeatedly

Solution: Check internal_port. Check health check endpoint is returning 200. Check logs.

DEBUG PRO TIP

Pro debugging workflow: fly logs (see what's happening) → fly ssh console (SSH in to inspect) → fly logs -f (follow logs in real-time).

PRE-FLIGHT CHECKLIST

Before you deploy

Project Setup

- [ ] fly.toml exists and is configured
- [ ] Dockerfile builds successfully (test with docker build)
- [ ] App tested locally with npm run dev
- [ ] .dockerignore excludes node_modules, .git, etc.

Secrets & Security

- [ ] No secrets in fly.toml or Dockerfile
- [ ] All sensitive values set with fly secrets set
- [ ] DATABASE_URL set if using Postgres
- [ ] .env.local in .gitignore

Deployment

- [ ] fly deploy succeeds without errors
- [ ] Health check endpoint returning 200
- [ ] Live URL is accessible
- [ ] App is functioning (test critical features)

Scaling & Performance

- [ ] Regions selected (primary_region set)
- [ ] VM size appropriate for workload
- [ ] min_machines_running configured
- [ ] fly status shows healthy machines

Monitoring

- [ ] fly logs accessible and readable
- [ ] Error tracking set up (Sentry, etc.)
- [ ] Uptime monitoring active
- [ ] Alarms configured for your critical services

FINAL CHECKPOINT

THE BOOTS STANDARD: You're ready to deploy when every item above is checked. No shortcuts. No guessing. This is how we ensure reliable, production-ready applications.