**BOOTS ON THE GROUND AI**

AI Solutions for Small Business Owners

# TERMINAL COMMANDS PLAYBOOK

## Every CLI Command You Need. One Book.

The complete terminal command reference for your entire deployment stack:

DOCKER | GIT / GITHUB | FLY.IO | SENTRY

Organized by frequency: Most Used first, then Superpower commands, then Danger Zone.

$_

TERMINAL

BOOTS APPROVED


Version 1.0 | February 12, 2026

# TABLE OF CONTENTS

Your complete terminal reference

# HOW TO USE THIS PLAYBOOK

Read this first

This playbook is your single reference for every terminal command across your deployment stack. Each tool section is organized into three tiers:

**>>> MOST USED - Commands you'll run daily**

These are your bread and butter. You'll use these commands every single day. Master these first and you can handle 90% of your workflow.

***** SUPERPOWER - Advanced commands for pros**

Once you're comfortable with the basics, these commands will 10x your productivity. They handle complex scenarios, debugging, and optimization.

**!!! DANGER ZONE - Destructive commands (be careful!)**

These commands can delete data, destroy resources, or cause irreversible changes. Always double-check before running them. There is no undo.

**BOOTS APPROVED**

BOOTS SAYS: Print this playbook and keep it next to your keyboard. When you're stuck in the terminal, flip to the right section and find your command. No Googling required.

# D DOCKER COMMANDS

## >>> MOST USED - Your daily Docker commands

**`docker build -t myapp .`**
Build an image from Dockerfile in current directory, tag it 'myapp'

**`docker run -p 3000:3000 myapp`**
Run container, map port 3000 on host to 3000 in container

**`docker ps`**
List all running containers (your first check when debugging)

**`docker logs <container_id>`**
View container logs (what's happening inside)

**`docker stop <container_id>`**
Gracefully stop a running container

**`docker compose up -d`**
Start all services defined in docker-compose.yml (detached)

**`docker compose down`**
Stop and remove all containers from docker-compose

**`docker images`**
List all local images (check what you've built)

**`docker exec -it <container_id> sh`**
Open a shell inside a running container (debug live)

**`docker compose logs -f`**
Follow logs from all compose services in real-time

## *** SUPERPOWER - Level up your Docker game

**`docker build --target builder -t myapp:builder .`**
Build only a specific stage in a multi-stage Dockerfile

**`docker compose up --build --force-recreate`**
Force rebuild and recreate all containers (fix stale images)

**`docker inspect <container_id>`**
Get ALL details about a container (networking, mounts, env vars)

**`docker stats`**
Live CPU/memory/network usage for all running containers

**`docker network ls`**
List Docker networks (debug container connectivity)

**`docker volume ls`**
List all volumes (persistent data storage)

**`docker system df`**
Show Docker disk usage (images, containers, volumes, build cache)

**`docker compose exec app npm run migrate`**
Run a one-off command inside a running compose service

**`docker build --no-cache -t myapp .`**
Build fresh without cache (fixes weird build issues)

**`docker cp <container_id>:/app/file.txt ./`**
Copy a file from container to your local machine

## !!! DANGER ZONE - These delete things permanently

**`docker system prune -a`**
Remove ALL unused images, containers, networks, and build cache
**WARNING: Deletes everything not currently running. You'll re-download all base images.**

**`docker rm -f $(docker ps -aq)`**
Force remove ALL containers (running and stopped)
**WARNING: Kills and removes every single container on your system.**

**`docker rmi $(docker images -q)`**
Remove ALL local images
**WARNING: You'll need to rebuild or re-pull every image.**

**`docker volume prune`**
Remove all unused volumes
**WARNING: Database data stored in volumes will be permanently deleted.**

```
docker compose down -v --rmi all
```

Stop everything, remove volumes AND images

**WARNING: Nuclear option. Removes containers, volumes, images, and networks.**

# G GIT / GITHUB COMMANDS

## >>> MOST USED - Your daily Git workflow

**`git status`**
See what files changed, what's staged, what branch you're on

**`git add .`**
Stage all changes for commit (use git add <file> for specific files)

**`git commit -m "your message"`**
Save staged changes with a descriptive message

**`git push origin main`**
Push commits to remote main branch

**`git pull origin main`**
Pull latest changes from remote (always pull before you push)

**`git checkout -b feature/my-feature`**
Create and switch to a new branch

**`git checkout main`**
Switch back to the main branch

**`git log --oneline -10`**
See last 10 commits in one-line format (quick history)

**`git diff`**
See unstaged changes (what you modified but haven't added yet)

**`git merge feature/my-feature`**
Merge a feature branch into your current branch

## *** SUPERPOWER - Advanced Git mastery

**`git stash`**
Temporarily save uncommitted changes (switch branches without committing)

**`git stash pop`**
Restore your stashed changes back to the working directory

**`git rebase main`**
Replay your branch commits on top of main (cleaner than merge)

**`git cherry-pick <commit_hash>`**
Apply a specific commit from another branch to your current branch

**`git log --graph --oneline --all`**
Visualize the full branch tree in your terminal

**`git blame <file>`**
See who changed each line and when (find who broke it)

**`git bisect start / bad / good`**
Binary search through commits to find which one introduced a bug

**`git remote -v`**
List all remote repositories and their URLs

**`git tag v1.0.0`**
Tag current commit as a release version

**`gh pr create --title "My PR" --body "Description"`**
Create a GitHub Pull Request from the CLI (requires gh CLI)

## !!! DANGER ZONE - These rewrite history

**`git reset --hard HEAD`**
Discard ALL uncommitted changes (staged and unstaged)
**WARNING: Everything not committed is gone forever. No recovery.**

**`git push --force origin main`**
Force push to remote (overwrites remote history)
**WARNING: Overwrites the remote branch. Other people's work can be lost.**

**`git reset --hard HEAD~3`**
Undo the last 3 commits AND delete all their changes
**WARNING: Permanently removes commits and their code changes.**

**`git clean -fd`**
Delete all untracked files and directories
**WARNING: Removes files Git doesn't know about. New files you forgot to add will vanish.**

```
git branch -D feature/old-branch
```

Force delete a branch even if not merged

**WARNING: Unmerged work on that branch is gone permanently.**

# F FLY.IO COMMANDS

## >>> MOST USED - Your daily Fly.io workflow

**`fly deploy`**
Build and deploy your app (your most-used Fly command)

**`fly status`**
Check if your app is running and healthy

**`fly logs`**
View application logs (first place to look when something breaks)

**`fly logs -f`**
Follow logs in real-time (like watching a live feed of your app)

**`fly launch`**
Initialize a new Fly app (creates fly.toml and Dockerfile)

**`fly open`**
Open your deployed app in the browser

**`fly secrets set KEY=value`**
Set an environment variable securely

**`fly secrets list`**
See which secrets are set (values are hidden)

**`fly ssh console`**
SSH into your running machine (debug live issues)

**`fly auth login`**
Authenticate with your Fly.io account

## *** SUPERPOWER - Scale and optimize

**`fly scale count 3`**
Scale to 3 machines (handle more traffic)

**`fly scale show`**
See current machine count and VM sizes

**`fly scale vm shared-cpu-2x --vm-memory 1024`**
Upgrade to 2 CPUs and 1GB RAM

**`fly regions add ord lax iad`**
Deploy to Chicago, LA, and Virginia simultaneously

**`fly regions list`**
See all available Fly.io regions worldwide

**`fly postgres create`**
Create a managed PostgreSQL database on Fly

**`fly postgres connect -a mydb`**
Connect to your Fly Postgres with psql

**`fly volumes list`**
List all persistent storage volumes

**`fly volumes create data --size 10 --region ord`**
Create a 10GB volume in Chicago

**`fly wireguard create`**
Set up private networking to your Fly apps

## !!! DANGER ZONE - Destroy and delete resources

**`fly apps destroy myapp`**
Permanently delete your entire Fly app
**WARNING: Deletes the app, all machines, and all associated resources. No undo.**

**`fly secrets unset KEY`**
Remove a secret from your app
**WARNING: If your app depends on this secret, it will crash on next deploy.**

**`fly volumes destroy vol_xxxxx`**
Permanently delete a storage volume
**WARNING: All data on that volume is gone forever. Databases, uploads, everything.**

**`fly postgres destroy mydb`**
Permanently delete a managed Postgres database
**WARNING: All database data is permanently deleted. Back up first.**

```
fly scale count 0
```

Scale to zero machines (app goes completely offline)

**WARNING: Your app becomes unreachable. No machines = no app.**

# S SENTRY CLI COMMANDS

## >>> MOST USED - Daily Sentry workflow

**`npx @sentry/wizard@latest -i nextjs`**
Set up Sentry in your Next.js project (one-command install)

**`sentry-cli login`**
Authenticate with your Sentry account

**`sentry-cli info`**
Check your current authentication and org/project settings

**`sentry-cli releases list`**
List all releases for your project

**`sentry-cli releases new v1.0.0`**
Create a new release version

**`sentry-cli releases finalize v1.0.0`**
Mark a release as finalized (deployed)

**`sentry-cli sourcemaps upload --release=v1.0.0 ./dist`**
Upload source maps for better error traces

**`npm install --save @sentry/nextjs`**
Install Sentry SDK for Next.js

**`npm install -g @sentry/cli`**
Install Sentry CLI globally

**`sentry-cli projects list`**
List all projects in your Sentry organization

## *** SUPERPOWER - Advanced error tracking

**`sentry-cli releases set-commits v1.0.0 --auto`**
Auto-associate Git commits with a release (see which code caused errors)

**`sentry-cli releases deploys v1.0.0 new -e production`**
Record a deployment event (track when releases go live)

**`sentry-cli sourcemaps explain --release=v1.0.0`**
Debug source map issues (why aren't error traces readable?)

**`sentry-cli releases files v1.0.0 list`**
List all uploaded files for a release

**`sentry-cli monitors list`**
List configured Cron Monitors (uptime checks)

**`sentry-cli send-event -m "Test event"`**
Send a test event to verify Sentry is working

**`sentry-cli organizations list`**
List all organizations you have access to

**`sentry-cli releases files v1.0.0 upload-sourcemaps ./out --rewrite`**
Upload and rewrite source maps for static exports

**`SENTRY_AUTH_TOKEN=xxx sentry-cli info`**
Test a specific auth token without changing your config

**`sentry-cli uninstall`**
Remove Sentry CLI from your system

## !!! DANGER ZONE - Delete tracking data

**`sentry-cli releases delete v1.0.0`**
Permanently delete a release and all its data
**WARNING: Error tracking data for that release is gone. Source maps removed.**

**`sentry-cli releases files v1.0.0 delete --all`**
Delete all uploaded files for a release
**WARNING: Source maps gone. Error traces become unreadable for that release.**

**`npm uninstall @sentry/nextjs`**
Remove Sentry SDK from your project
**WARNING: All error tracking stops immediately. You're flying blind.**

**`Revoking auth token in Sentry dashboard`**
Invalidate your CLI authentication token
**WARNING: All CI/CD pipelines using this token will break immediately.**

# QUICK REFERENCE CHEAT SHEET

Copy, paste, deploy

## New Project Setup (from zero to deployed)

**1.** `git init`

*Initialize Git in your project*

**2.** `git add . && git commit -m "initial commit"`

*Stage and commit everything*

**3.** `gh repo create myapp --public --push`

*Create GitHub repo and push (requires gh CLI)*

**4.** `docker build -t myapp .`

*Build your Docker image*

**5.** `docker run -p 3000:3000 myapp`

*Test it locally in Docker*

**6.** `fly launch`

*Initialize Fly.io app*

**7.** `fly secrets set DATABASE_URL=...`

*Set your secrets*

**8.** `fly deploy`

*Deploy to production*

**9.** `npx @sentry/wizard@latest -i nextjs`

*Add Sentry error monitoring*

**10.** `fly open`

*Open your live app and celebrate*

## Daily Deploy Workflow

**`git pull origin main`**
Get latest code

**`git checkout -b feature/new-thing`**
Create feature branch

```
# ... write code ...
```
Do your work

```
docker compose up -d
```
Test locally

```
git add . && git commit -m "feat: new thing"
```
Commit your work

```
git push origin feature/new-thing
```
Push to GitHub

```
gh pr create
```
Create Pull Request

```
# ... after merge ...
```

```
git checkout main && git pull
```
Update local main

```
fly deploy
```
Deploy to production

```
fly logs -f
```
Watch logs for errors

```
sentry-cli releases new v1.x.x
```
Tag release in Sentry

# PRE-DEPLOY TERMINAL CHECKLIST

Run these before every deploy

## Git Status Check

- [ ] git status shows clean working tree
- [ ] git pull origin main completed (no conflicts)
- [ ] Feature branch merged and PR approved
- [ ] git log confirms your commits are there

## Docker Verification

- [ ] docker build completes without errors
- [ ] docker run works locally (test the app)
- [ ] docker compose up -d starts all services
- [ ] .dockerignore excludes node_modules, .git, .env

## Fly.io Pre-Deploy

- [ ] fly.toml configured correctly (check internal_port)
- [ ] fly secrets set for all environment variables
- [ ] No secrets in fly.toml or Dockerfile
- [ ] fly status shows healthy before deploying new version

## Sentry Monitoring

- [ ] Sentry SDK installed and configured
- [ ] sentry-cli releases new created for this version
- [ ] Source maps uploaded with sentry-cli sourcemaps upload
- [ ] Test error sent with sentry-cli send-event

## Final Check

- [ ] fly deploy succeeds without errors

- [ ] fly logs shows clean startup (no crashes)
- [ ] Live URL is accessible and functional
- [ ] Sentry dashboard shows the new release

**FINAL CHECKPOINT**

THE BOOTS STANDARD: Every checkbox checked means your deployment is solid, monitored, and production-ready. No shortcuts. This checklist is your shield against downtime.