# DineWise Deployment Guide

## Table of Contents

1. [Local Development Setup](#1fob9te)
   * [Prerequisites](#3znysh7)
   * [Frontend Setup (React)](#2et92p0)
   * [Backend Setup (Flask)](#tyjcwt)
2. [Google Cloud Platform (GCP) Deployment](#3dy6vkm)
   * [Frontend Deployment (Cloud Run)](#1t3h5sf)
   * [Backend Deployment (Cloud Run)](#4d34og8)
3. [DigitalOcean Deployment](#2s8eyo1)
   * [Prerequisites](#17dp8vu)
   * [Frontend Deployment (App Platform)](#3rdcrjn)
   * [Backend Deployment (Droplet)](#26in1rg)
4. [Important Security Notes](#lnxbz9)
5. [Troubleshooting](#35nkun2)
   * [Common Issues](#1ksv4uv)

## Local Development Setup

### Prerequisites

* Node.js 16+ and npm
* Python 3.8+
* Git
* Firebase account

### Frontend Setup (React)

1. Clone the repository:

git clone [your-repo-url]  
cd dinewise

1. Install dependencies:

npm install

1. Create a .env file with your Firebase configuration:

VITE\_FIREBASE\_API\_KEY=your\_api\_key  
VITE\_FIREBASE\_AUTH\_DOMAIN=your\_domain  
VITE\_FIREBASE\_PROJECT\_ID=your\_project\_id  
VITE\_FIREBASE\_STORAGE\_BUCKET=your\_bucket  
VITE\_FIREBASE\_MESSAGING\_SENDER\_ID=your\_sender\_id  
VITE\_FIREBASE\_APP\_ID=your\_app\_id

1. Start the development server:

npm run dev

### Backend Setup (Flask)

1. Create and activate a virtual environment:

python -m venv venv  
source venv/bin/activate *# On Windows: .\venv\Scripts\activate*

1. Install dependencies:

pip install -r requirements.txt

1. Add your Firebase credentials file:

* Download your Firebase service account key
* Save it as firebase\_credentials.json in the root directory

1. Start the Flask server:

python app.py

## Google Cloud Platform (GCP) Deployment

### Frontend Deployment (Cloud Run)

1. Install Google Cloud CLI:

* Download from https://cloud.google.com/sdk/docs/install

1. Initialize GCP:

gcloud init  
gcloud auth configure-docker

1. Build and push Docker image:

docker build -t gcr.io/[PROJECT\_ID]/dinewise-frontend .  
docker push gcr.io/[PROJECT\_ID]/dinewise-frontend

1. Deploy to Cloud Run:

gcloud run deploy dinewise-frontend \  
 --image gcr.io/[PROJECT\_ID]/dinewise-frontend \  
 --platform managed \  
 --region [REGION] \  
 --allow-unauthenticated

### Backend Deployment (Cloud Run)

1. Build backend Docker image:

docker build -t gcr.io/[PROJECT\_ID]/dinewise-backend -f Dockerfile.backend .  
docker push gcr.io/[PROJECT\_ID]/dinewise-backend

1. Deploy backend:

gcloud run deploy dinewise-backend \  
 --image gcr.io/[PROJECT\_ID]/dinewise-backend \  
 --platform managed \  
 --region [REGION] \  
 --allow-unauthenticated

## DigitalOcean Deployment

### Prerequisites

* DigitalOcean account
* doctl CLI installed

### Frontend Deployment (App Platform)

1. Install doctl and authenticate:

doctl auth init

1. Create app specification (app.yaml):

name**:** dinewise-frontend  
region**:** nyc  
services**:**  
**-** name**:** web  
 github**:**  
 repo**:** your-repo  
 branch**:** main  
 source\_dir**:** /  
 envs**:**  
 **-** key**:** NODE\_ENV  
 value**:** production

1. Deploy using App Platform:

doctl apps create --spec app.yaml

### Backend Deployment (Droplet)

1. Create a Droplet:

doctl compute droplet create dinewise-backend \  
 --image ubuntu-20-04-x64 \  
 --size s-1vcpu-1gb \  
 --region nyc1

1. SSH into the Droplet:

doctl compute ssh dinewise-backend

1. Install dependencies and setup:

sudo apt update **&&** sudo apt upgrade -y  
sudo apt install python3-pip nginx -y  
git clone [your-repo-url]  
cd dinewise  
pip3 install -r requirements.txt

1. Setup Nginx and Gunicorn:

sudo nano /etc/nginx/sites-available/dinewise

Add configuration:

server {  
 listen 80;  
 server\_name your\_domain.com;  
  
 location / {  
 proxy\_pass http://localhost:8000;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 }  
}

1. Enable and start services:

sudo ln -s /etc/nginx/sites-available/dinewise /etc/nginx/sites-enabled  
sudo systemctl restart nginx  
gunicorn -w 4 app:app

## Important Security Notes

1. Always use HTTPS in production
2. Secure your environment variables
3. Keep Firebase credentials private
4. Regularly update dependencies
5. Set up monitoring and logging
6. Configure proper CORS settings

## Troubleshooting

### Common Issues

1. **Firebase Connection Issues**
   * Verify credentials file location
   * Check Firebase console permissions
2. **Deployment Failures**
   * Verify environment variables
   * Check build logs
   * Ensure sufficient resources
3. **Performance Issues**
   * Monitor resource usage
   * Check database indexes
   * Optimize API calls

For more detailed troubleshooting, consult the error logs in your respective platform’s monitoring dashboard.