WeatherWear Deployment Manual

Table of Contents

Prerequisites

Required Services

- **Node.js (18.x or later)**: Ensure Node.js is installed and up-to-date.
- **Git**: A Git client must be installed for version control.
- **Firebase Account**: Required for authentication and related services.
- **RapidAPI Subscription**: Access to OpenWeather and Google Maps Geocoding APIs.

Required Tools

- **npm or yarn**: For dependency management.
- **Firebase CLI**: For deploying and managing Firebase services.
- **AWS CLI**: For AWS deployments (if applicable).
- **Vercel CLI**: For deployments to Vercel.

Required Environment Variables

Create a local environment file (e.g., .env.local) and set the following variables:

```
# API Keys
RAPIDAPI_KEY=your_rapidapi_key
```

Weatherwear

```
# Firebase Configuration
FIREBASE_API_KEY=your_firebase_api_key
FIREBASE_AUTH_DOMAIN=your-app.firebaseapp.com
FIREBASE_PROJECT_ID=your-project-id
FIREBASE_STORAGE_BUCKET=your-app.appspot.com
FIREBASE_MESSAGING_SENDER_ID=your_sender_id
FIREBASE_APP_ID=your_app_id

# Application Version
NEXT_PUBLIC_VERSION=1.0.0
```

Local Development

Initial Setup

```
# Clone the repository
git clone https://github.com/your-org/weatherwear.git
cd weatherwear

# Install dependencies
npm install

# Set up local environment variables
cp .env.example .env.local
# Edit .env.local with your API keys and configurations

# Start the development server
npm run dev
```

Build and Test

```
# Run test suite
npm run test

# Build the application
npm run build

# Start the production server locally
```

weatherwear

```
npm run start
```

Environment Configuration

It is recommended to maintain separate environment files for each stage:

- **Development** (.env.development)
- **Production** (.env.production)
- **Testing** (.env.test)

Each file should include environment-specific variables.

Example .env.development:

```
NODE_ENV=development
# Add any development-specific environment variables here
```

Deployment Options

Vercel Deployment

Setup

1. Install the Vercel CLI:
   ```
   npm i -g vercel
   ```
2. Log in to Vercel:
   ```
   vercel login
   ```

1. **Deploy**

Initial deployment:

```
vercel
```

Production deployment:

```
vercel --prod
```

2. Configuration (Example vercel.json)

Weathenwear

```
{
  "version": 2,
  "builds": [
    {
      "src": "package.json",
      "use": "@vercel/next"
    }
  ],
  "env": {
    "RAPIDAPI_KEY": "@rapidapi-key",
    "FIREBASE_API_KEY": "@firebase-api-key"
  }
}
```

AWS Deployment

Prerequisites

- An active AWS account
- AWS CLI installed and configured
- ECR repository (if using containerized deployment)
- ECS cluster (if using ECS)

Option 1: AWS Amplify

1.Install the Amplify CLI:

```
npm install -g @aws-amplify/cli
```

2. Configure Amplify:

```
amplify configure
```

3. Initialize the project:

```
amplify init
```

4.Deploy to Amplify:

```
amplify push
```

Option 2: Containerized Deployment (ECS)

Dockerfile example:

weatherwear

```
FROM node:18-alpine AS base

# Dependencies
FROM base AS deps
RUN apk add --no-cache libc6-compat
WORKDIR /app
COPY package*.json ./
RUN npm ci

# Builder
FROM base AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .
RUN npm run build

# Runner
FROM base AS runner
WORKDIR /app
ENV NODE_ENV production
COPY --from=builder /app/public ./public
COPY --from=builder /app/.next/standalone ./
COPY --from=builder /app/.next/static ./.next/static
CMD ["node", "server.js"]
```

Deploy Steps:

1. Authenticate and push the Docker image to ECR:

   ```
   aws ecr get-login-password --region <region> | docker login --username AWS
   --password-stdin <account_id>.dkr.ecr.<region>.amazonaws.com
docker build -t weatherwear .
docker tag weatherwear:latest
<account_id>.dkr.ecr.<region>.amazonaws.com/weatherwear:latest
docker push <account_id>.dkr.ecr.<region>.amazonaws.com/weatherwear:latest
```

2.Update your ECS service (either via the AWS Console or CLI) to use the new image.

Monitoring

weatherwear

Application Monitoring

Implement metrics and logging within your code to track performance and error rates. For example:

```
import { metrics } from './lib/metrics';

export const monitor = {
  logError(error: Error, context?: Record<string, any>) {
    metrics.increment('error', { ...context });
    console.error(error);
  },

  logAPICall(endpoint: string, duration: number) {
    metrics.timing('api.call', duration, { endpoint });
  }
};
```

Health Checks

Implement a simple health check endpoint to verify uptime and deployment status:

```
// pages/api/health.ts
export default function handler(req, res) {
  const health = {
    uptime: process.uptime(),
    timestamp: Date.now(),
    status: 'healthy',
    version: process.env.NEXT_PUBLIC_VERSION
  };

  res.status(200).json(health);
}
```

weatherwear

Maintenance

Regular Tasks

1.Update Dependencies:

```
npm audit
npm update
```

2.Review Logs:

```
# Vercel
vercel logs

# AWS CloudWatch
aws logs get-log-events --log-group-name /aws/weatherwear
```

Backup Procedures

1.Database Backup (Firebase):

```
firebase firestore:export backups/$(date +%Y%m%d)
```

2.Environment Variable Backup:

```
# Vercel
vercel env pull .env.backup

# AWS Parameter Store
aws ssm get-parameters-by-path --path /weatherwear/prod > env_backup.json
```

Troubleshooting

Common Issues

1. API Connection Errors:

```
try {
  const response = await fetch(API_URL);
  if (!response.ok) {
    throw new Error(`API Error: ${response.status}`);
  }
} catch (error) {
  monitor.logError(error, { service: 'weather-api' });
  // Consider implementing retry logic here
}
```

2. Build Failures:

```
rm -rf .next node_modules
npm cache clean --force
npm install
```

3. Authentication Issues:

```
firebase.auth().onAuthStateChanged((user) => {
```

weatherwear

```
  if (!user) {
    console.error('Authentication failed');
    // Implement recovery logic as needed
  }
});
```

Performance Optimization

1. **Enable Caching**:

```
// next.config.js
module.exports = {
  async headers() {
    return [
      {
        source: '/api/:path*',
        headers: [
          {
            key: 'Cache-Control',
            value: 'public, max-age=300, stale-while-revalidate=60'
          }
        ]
      }
    ];
  }
};
```

2. Image Optimization:

```
// next.config.js
module.exports = {
  images: {
    domains: ['your-cdn-domain'],
    deviceSizes: [640, 750, 828, 1080, 1200],
    imageSizes: [16, 32, 48, 64, 96]
  }
};
```

Emergency Procedures

1. **Rollbacks**:

```
# Vercel
vercel rollback

# AWS Amplify
```

weatherwear

```
amplify push --codegen

# ECS (revert to previous task definition)

aws ecs update-service --cluster <cluster-name> --service <service-name>
--task-definition <previous-task-definition>
```

weatherwear