Vercel Deployment Guide

Complete setup for Next.js TypeScript Boilerplate

Overview

This guide will walk you through the complete process of setting up automated deployments to Vercel using GitHub Actions. You'll learn how to configure tokens, secrets, and CI/CD pipelines for a seamless deployment experience.

What You'll Achieve:

- Automatic deployments on every push
- Secure authentication with Vercel
- Professional CI/CD pipeline
- Protected deployment environment

Prerequisites

- **GitHub Repository:** Your Next.js project pushed to GitHub
- **Vercel Account:** Free account at **vercel.com**
- **Vercel CLI:** Optional but recommended for local testing

□Step 1: Local Vercel Configuration

Install Vercel CLI

Option A: Global Installation

```
npm install -g vercel
```

Option B: Use NPX (Recommended)

```
npx vercel
```

Pro Tip: Using `npx vercel` ensures you always use the latest version without global installation.

Login to Vercel

```
npx vercel login
```

Choose your preferred authentication method:

- **GitHub:** Seamless integration with your repositories
- **GitLab:** For GitLab users
- **Email:** Direct email authentication

Link Your Project

Navigate to your project directory and run:

```
cd your-nextjs-project npx vercel link
```

Configuration Prompts:

1. **Scope Selection:** Choose your **personal account** for protected deployments
2. **Project Setup:**
   - Select "*Create a new project*" for first-time setup
   - Or "*Link to existing project*" if you've already created one
3. **Project Name:** Choose a memorable name (e.g., `nextjs-typescript-boilerplate`)

Verify Configuration

After linking, check the generated configuration:

```
cat .vercel/project.json
```

Expected Output:

```
{ "orgId": "your-org-id-here", "projectId": "your-project-id-here" }
```

⚠ **Important Security Note:** The `.vercel/project.json` file is **local only** and should **never be committed** to version control. It's automatically included in `.gitignore`.

Step 2: Vercel Token Generation

Create Personal Access Token

1. Go to **Vercel Tokens Page**
2. Click **"Create Token"**
3. Provide a meaningful name: `nextjs-boilerplate-ci`
4. Set expiration (recommended: 1 year for CI/CD)
5. Select scope: **Full Account** (for deployment permissions)
6. **Copy the token immediately** - it won't be shown again!

Security Best Practice: Store this token securely and never commit it to your repository. We'll add it to GitHub Secrets in the next step.

Token Usage Context

Where This Token Is Used:

- **GitHub Actions:** Authenticates CI/CD pipeline with Vercel
- **Automated Deployments:** Enables push-to-deploy workflow
- **Build Process:** Allows CI to build and deploy your app

Step 3: GitHub Secrets Configuration

Required Secrets

Secret Name	Value Source	Purpose
`VERCEL_TOKEN`	Personal access token from Vercel	Authentication with Vercel API
`VERCEL_ORG_ID`	`orgId` from `.vercel/project.json`	Organization identification
`VERCEL_PROJECT_ID`	`projectId` from `.vercel/project.json`	Project identification

⚙Adding Secrets to GitHub

1. Navigate to your GitHub repository
2. Go to **Settings** → **Secrets and variables** → **Actions**
3. Click **"New repository secret"**
4. Add each secret individually:
   - **Name:** `VERCEL_TOKEN`
     Value: Your generated token
   - **Name:** `VERCEL_ORG_ID`
     Value: From `.vercel/project.json`
   - **Name:** `VERCEL_PROJECT_ID`
     Value: From `.vercel/project.json`

Security Benefits:

- Secrets are encrypted and only accessible during workflow runs
- Contributors can't view secret values
- Automatic masking in logs prevents accidental exposure

Step 4: CI/CD Pipeline Configuration

GitHub Actions Workflow

Your project includes a pre-configured workflow at `.github/workflows/ci.yml`:

```
name: CI/CD Pipeline on: push: branches: [ main, develop ] pull_request: branches: [ main
] jobs: test: runs-on: ubuntu-latest steps: - uses: actions/checkout@v4 - uses:
actions/setup-node@v4 with: node-version: '18' cache: 'npm' - run: npm ci - run: npm run
lint - run: npm run format:ci - run: npm run build - run: npm run test:ci deploy: needs:
test runs-on: ubuntu-latest if: github.ref == 'refs/heads/main' steps: - uses:
actions/checkout@v4 - uses: actions/setup-node@v4 with: node-version: '18' cache: 'npm' -
run: npm ci - run: npm run build - uses: amondnet/vercel-action@v25 with: vercel-token:
${{ secrets.VERCEL_TOKEN }} vercel-org-id: ${{ secrets.VERCEL_ORG_ID }} vercel-project-id:
${{ secrets.VERCEL_PROJECT_ID }} vercel-args: '--prod'
```

Pipeline Stages Explained

Test Stage

- **Code Quality:** ESLint checks for code standards

- **Formatting:** Prettier ensures consistent code style
- **Build Verification:** Ensures the app compiles successfully
- **Unit Tests:** Jest + React Testing Library validation

Deploy Stage

- **Conditional:** Only runs on `main` branch
- **Production Build:** Optimized build for deployment
- **Vercel Deploy:** Automated deployment using official action

⚡Triggering Deployments

Automatic Triggers:

- **Push to main:** Full production deployment
- **Push to develop:** Preview deployment
- **Pull Requests:** Preview deployments for review

Step 5: Testing Your Setup

Local Testing

Before pushing to production, test locally:

```
# Build and test locally npm run build npm run test # Test Vercel deployment locally npx vercel --prod
```

First Deployment

1. Make a small change to your project
2. Commit and push to the `main` branch:

   ```
   git add . git commit -m "feat: initial deployment setup" git push origin main
   ```

3. Monitor the deployment:
   - **GitHub Actions:** Check the "Actions" tab in your repo
   - **Vercel Dashboard:** Monitor deployment progress

Step 6: Monitoring & Management

Vercel Dashboard

Access your deployment metrics at **Vercel Dashboard**

Key Features:

- **Deployment History:** Track all deployments and rollbacks
- **Performance Metrics:** Core Web Vitals and loading times
- **Preview Deployments:** Test branches and PRs
- **Custom Domains:** Configure your own domain

GitHub Actions Monitoring

Workflow Status:

- **Success:** Green checkmark indicates successful deployment
- **Failure:** Red X shows build or deployment issues
- **In Progress:** Yellow circle indicates running workflow

Troubleshooting: Check workflow logs for detailed error messages if deployments fail.

⚡Advanced Configuration

Environment Variables

Configure environment-specific variables in Vercel:

1. Go to your project in Vercel Dashboard
2. Navigate to **Settings** → **Environment Variables**
3. Add variables for different environments:
    - **Production:** Live environment variables
    - **Preview:** Staging/development variables
    - **Development:** Local development overrides

Custom Domains

Custom Domains

Adding Your Domain:

1. Purchase and configure your domain
2. In Vercel Dashboard: **Settings → Domains**
3. Add your domain and configure DNS
4. Enable automatic HTTPS (included free)

Security Best Practices

Security Checklist:

- **Never commit** `.vercel/` folder to version control
- **Rotate tokens** regularly (every 6-12 months)
- **Use scoped tokens** with minimal required permissions
- **Monitor deployment logs** for suspicious activity
- **Enable branch protection** on main branch

Success Indicators

You'll know everything is working when:

- ✅ Green checkmarks appear on GitHub commits
- ✅ Vercel dashboard shows successful deployments
- ✅ Your app is accessible at the Vercel URL
- ✅ Preview deployments work for feature branches
- ✅ Environment variables are properly configured

Troubleshooting Guide

Common Issues

Build Failures

- **TypeScript Errors:** Fix type issues before deployment

- **Dependency Issues:** Ensure `package-lock.json` is committed
- **Environment Variables:** Check all required vars are set

Authentication Problems

- **Invalid Token:** Regenerate and update GitHub secrets
- **Wrong IDs:** Verify org/project IDs from `.vercel/project.json`
- **Permissions:** Ensure token has deployment permissions

Deployment Issues

- **Timeout:** Optimize build process or increase timeout
- **Size Limits:** Check for large files or dependencies
- **Runtime Errors:** Test thoroughly before deployment

Debug Commands

```
# Check Vercel CLI version npx vercel --version # List your projects npx vercel ls # Check
deployment status npx vercel inspect # View deployment logs npx vercel logs [deployment-url]
```

Need Help?

Visit Vercel Documentation or Community Discussions

Created with ❤ for Next.js TypeScript Boilerplate