GitHUb Actions Deploy to Production

GitHub Actions provides a powerful way to automate deployments for your projects. Below is a step-by-step guide to set up deployment workflows using GitHub Actions.

1. Create a Workflow File

- Navigate to your repository and create a .github/workflows/deploy.yml file.
- Define the workflow triggers, such as push or workflow_dispatch (manual trigger).

Example:

name: Deploy Application

on:

push:

branches:

- main

workflow_dispatch:

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout Code

uses: actions/checkout@v4

- name: Set Up Node.js

uses: actions/setup-node@v4

with:

node-version: '16'

- name: Install Dependencies

```
run: npm install

- name: Build Application
run: npm run build

- name: Deploy to Production
run: |
npx vercel --token ${{ secrets.VERCEL_TOKEN }} --prod --yes
```

2. Use Environments for Deployment

- Define environments like staging or production in your repository settings.
- Add environment-specific secrets (e.g., API keys, deployment tokens).
- Reference the environment in your workflow:

```
environment: production
```

3. Add Deployment Protection Rules

- Configure rules such as required approvals or automated checks before deployment.
- Example use case: Require a team lead's approval before deploying to production.

4. Monitor and Debug Deployments

- View real-time logs and deployment history in the "Actions" tab of your repository.
- Use status badges in your README to display workflow status:

```
![Deployment Status](https://github.com/<user>/<repo>/actions/workflows/deploy.yml/badge.svg)
```

Best Practices

- Use secrets for sensitive data like API keys or tokens.
- Test workflows in staging environments before deploying to production.
- Leverage GitHub Marketplace actions for specific deployment platforms (e.g., AWS, Azure, Vercel).

This setup ensures secure, automated, and efficient deployments for your applications