Setting Up CI/CD with GitHub Actions for Laravel Deployment

Here's a comprehensive guide to set up automated deployment for your Laravel application using GitHub Actions.

1. Basic GitHub Actions Workflow

Create a `.github/workflows/laravel.yml` file in your repository:

```
name: Laravel CI/CD

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  tests:
    runs-on: ubuntu-latest

    services:
      mysql:
        image: mysql:5.7
        env:
          MYSQL_ROOT_PASSWORD: secret
          MYSQL_DATABASE: laravel
        ports:
          - 3306:3306
        options: --health-cmd="mysqladmin ping" --health-interval=10s --
health-timeout=5s --health-retries=3

    steps:
    - uses: actions/checkout@v4

    - name: Setup PHP
      uses: shivammathur/setup-php@v2
      with:
        php-version: '8.2'
        extensions: mbstring, ctype, fileinfo, openssl, PDO, mysql,
pgsql, sqlite, gd, exif, pcntl, bcmath, intl
        coverage: none

    - name: Install dependencies
      run: |
        composer install --prefer-dist --no-interaction --no-progress
```

```
    - name: Copy .env
      run: |
        cp .env.example .env
        php artisan key:generate

    - name: Directory Permissions
      run: |
        mkdir -p storage/framework/{sessions,views,cache}
        chmod -R 777 storage bootstrap/cache

    - name: Execute tests
      env:
        DB_CONNECTION: mysql
        DB_DATABASE: laravel
        DB_USERNAME: root
        DB_PASSWORD: secret
      run: |
        php artisan migrate:fresh --env=testing --force
        php artisan test
```

2. Deployment Workflow (to a Server)

For deployment to a server (like a VPS), add this to your workflow:

```
  deploy:
    needs: tests
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4

    - name: Install dependencies
      run: composer install --prefer-dist --no-dev --no-interaction --
no-progress

    - name: Upload files via SSH
      uses: appleboy/scp-action@v0.1.3
      with:
        host: ${{ secrets.SSH_HOST }}
        username: ${{ secrets.SSH_USERNAME }}
        key: ${{ secrets.SSH_PRIVATE_KEY }}
        port: ${{ secrets.SSH_PORT }}
        source: "./"
        target: "/var/www/your-project"

    - name: Run deployment commands
      uses: appleboy/ssh-action@v0.1.10
      with:
        host: ${{ secrets.SSH_HOST }}
        username: ${{ secrets.SSH_USERNAME }}
```

```
          key: ${{ secrets.SSH_PRIVATE_KEY }}
          port: ${{ secrets.SSH_PORT }}
          script: |
            cd /var/www/your-project
            composer install --no-dev
            php artisan migrate --force
            php artisan optimize:clear
            php artisan optimize
```

3. Required GitHub Secrets

You'll need to set up these secrets in your GitHub repository (Settings > Secrets):

- `SSH_HOST`: Your server IP or domain
- `SSH_USERNAME`: SSH username (usually "root" or your sudo user)
- `SSH_PRIVATE_KEY`: Your private SSH key
- `SSH_PORT`: SSH port (usually 22)

4. Alternative: Deployment to Shared Hosting

For shared hosting without SSH access:

```
  deploy:
    needs: tests
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4

    - name: Install dependencies
      run: composer install --prefer-dist --no-dev --no-interaction --
no-progress

    - name: Zip artifacts
      run: zip -r deploy.zip . -x '*.git*'

    - name: Upload artifact
      uses: actions/upload-artifact@v3
      with:
        name: deployment-package
        path: deploy.zip
```

Then manually download the artifact and upload to your hosting.

5. Additional Optimizations

Consider adding these steps to your workflow:

PROF

```
- name: Cache Composer packages
  uses: actions/cache@v3
  with:
    path: vendor
    key: ${{ runner.os }}-php-${{ hashFiles('**/composer.lock') }}
    restore-keys: |
      ${{ runner.os }}-php-

- name: Cache NPM packages
  if: steps.yarn-cache.outputs.cache-hit != 'true'
  uses: actions/cache@v3
  with:
    path: node_modules
    key: ${{ runner.os }}-node-${{ hashFiles('**/package-lock.json') }}
    restore-keys: |
      ${{ runner.os }}-node-

- name: Install NPM dependencies
  run: npm ci && npm run prod
```

6. Environment-Specific Workflows

For different environments (staging/production):

```
on:
  push:
    branches:
      - main
    paths-ignore:
      - 'docs/**'
      - 'README.md'

  workflow_dispatch:
    inputs:
      environment:
        description: 'Environment to deploy to'
        required: true
        default: 'staging'
        type: choice
        options:
        - staging
        - production
```

PROF

This setup provides:

1. Automated testing on every push/pull request
2. Secure deployment to your server
3. Proper environment handling

4. Caching for faster builds
5. Flexibility for different deployment targets

Remember to adjust paths, server details, and commands according to your specific project requirements.