

github-laravel- actions

- [Sadržaj](#)

Sadržaj

Da, možete automatizirati sigurnosne kopije svog Laravel projekta na GitHub koristeći GitHub Actions. GitHub Actions omogućuje automatiziranje tijeka rada (workflow) direktno na GitHubu, uključujući i izradu sigurnosnih kopija.

Evo koraka kako postaviti GitHub Actions za sigurnosnu kopiju vašeg Laravel projekta:

Stvorite GitHub Actions tijekom rada (workflow)

U korijenu vašeg Laravel projekta, stvorite direktorij `.github/workflows` ako već ne postoji.

Unutar tog direktorija, stvorite novu YAML datoteku, npr. `backup.yml`:

yml Kopiraj kod name: Backup

on: schedule: # Pokreće se svakodnevno u ponoć - cron: '0 0 * * *' workflow_dispatch: # Omogućuje ručno pokretanje

jobs: backup: runs-on: ubuntu-latest

steps:

- name: Checkout repository

uses: actions/checkout@v2

- name: Set up PHP

uses: shivammathur/setup-php@v2

with:

php-version: '8.0' # Postavite vašu verziju PHP-a

- name: Install dependencies

run: composer install --no-progress --prefer-dist --optimize-autoloader

- name: Run database migrations

run: php artisan migrate --force

- name: Run Spatie Backup

env:

GOOGLE_DRIVE_CLIENT_ID: \${ secrets.GOOGLE_DRIVE_CLIENT_ID }

GOOGLE_DRIVE_CLIENT_SECRET: \${ secrets.GOOGLE_DRIVE_CLIENT_SECRET }

```
GOOGLE_DRIVE_REFRESH_TOKEN: ${ secrets.GOOGLE_DRIVE_REFRESH_TOKEN }  
GOOGLE_DRIVE_FOLDER_ID: ${ secrets.GOOGLE_DRIVE_FOLDER_ID }  
run: php artisan backup:run
```

Postavite tajne (secrets) u GitHub repozitorij

Idite na vaš GitHub repozitorij, zatim idite na Settings > Secrets > Actions.

Dodajte sljedeće tajne (secrets):

GOOGLE_DRIVE_CLIENT_ID GOOGLE_DRIVE_CLIENT_SECRET GOOGLE_DRIVE_REFRESH_TOKEN
GOOGLE_DRIVE_FOLDER_ID Postavite vrijednosti tih tajni prema podacima iz vaše .env datoteke.

Pokrenite tijek rada (workflow)

GitHub Actions će sada automatski pokretati izradu sigurnosnih kopija svakodnevno u ponoć prema postavljenom cron rasporedu. Također možete ručno pokrenuti tijek rada iz sekcije Actions vašeg GitHub repozitorija.

Objašnjenje YAML konfiguracije on: schedule definira raspored za pokretanje tijeka rada pomoću cron sintakse. U ovom primjeru, tijek rada će se pokrenuti svakodnevno u ponoć. jobs definira zadatak koji se treba izvršiti. steps definira korake koje tijek rada treba izvesti: actions/checkout@v2 klonira repozitorij. shivammathur/setup-php@v2 postavlja PHP okruženje. composer install instalira PHP ovisnosti. php artisan migrate --force pokreće migracije baze podataka. php artisan backup:run pokreće izradu sigurnosne kopije koristeći Spatie Backup paket. Ovim pristupom možete automatizirati sigurnosne kopije svog Laravel projekta na GitHub, koristeći GitHub Actions. Ova metoda osigurava redovite sigurnosne kopije i omogućava ručno pokretanje po potrebi.

Rekli ste: A github workflow i actions za backupe i npr schedule naredbe za čišćenje, optimizaciju i composer i npm dependencies ChatGPT je rekao: ChatGPT Da biste postavili GitHub Actions workflow za sigurnosne kopije, čišćenje, optimizaciju, te ažuriranje Composer i npm ovisnosti, slijedite ove korake:

Stvorite GitHub Actions tijek rada (workflow)

U korijenu vašeg Laravel projekta, stvorite direktorij .github/workflows ako već ne postoji.

Unutar tog direktorija, stvorite novu YAML datoteku, npr. maintenance.yml:

yml Kopiraj kod name: Maintenance

on: schedule: # Pokreće se svakodnevno u ponoć - cron: '0 0 * * *' workflow_dispatch: #
Omogućuje ručno pokretanje

jobs: maintenance: runs-on: ubuntu-latest

steps:

- name: Checkout repository

uses: actions/checkout@v2

- name: Set up PHP

uses: shivammathur/setup-php@v2

with:

php-version: '8.0' # Postavite vašu verziju PHP-a

- name: Cache Composer dependencies

uses: actions/cache@v2

with:

path: vendor

key: \${{ runner.os }}-composer-\${{ hashFiles('**/composer.lock') }}

restore-keys: \${{ runner.os }}-composer-

- name: Install Composer dependencies

run: composer install --no-progress --prefer-dist --optimize-autoloader

- name: Cache npm dependencies

uses: actions/cache@v2

with:

path: node_modules

key: \${{ runner.os }}-node-\${{ hashFiles('**/package-lock.json') }}

restore-keys: \${{ runner.os }}-node-

- name: Install npm dependencies

run: npm install

- name: Run database migrations

run: php artisan migrate --force

- name: Run optimizations

run: |

php artisan optimize:clear

php artisan cache:clear

php artisan config:cache

php artisan route:cache

php artisan view:cache

```
- name: Run Composer updates
  run: composer update

- name: Run npm updates
  run: npm update

- name: Run Spatie Backup
  env:
    GOOGLE_DRIVE_CLIENT_ID: ${ secrets.GOOGLE_DRIVE_CLIENT_ID }
    GOOGLE_DRIVE_CLIENT_SECRET: ${ secrets.GOOGLE_DRIVE_CLIENT_SECRET }
    GOOGLE_DRIVE_REFRESH_TOKEN: ${ secrets.GOOGLE_DRIVE_REFRESH_TOKEN }
    GOOGLE_DRIVE_FOLDER_ID: ${ secrets.GOOGLE_DRIVE_FOLDER_ID }
  run: php artisan backup:run

- name: Run cleanup commands
  run: |
    php artisan schedule:run
```

Postavite tajne (secrets) u GitHub repozitorij

Idite na vaš GitHub repozitorij, zatim idite na Settings > Secrets > Actions.

Dodajte sljedeće tajne (secrets):

GOOGLE_DRIVE_CLIENT_ID GOOGLE_DRIVE_CLIENT_SECRET GOOGLE_DRIVE_REFRESH_TOKEN
GOOGLE_DRIVE_FOLDER_ID Postavite vrijednosti tih tajni prema podacima iz vaše .env datoteke.

Pokrenite tijek rada (workflow)

GitHub Actions će sada automatski pokretati zadatke svakodnevno u ponoć prema postavljenom cron rasporedu. Također možete ručno pokrenuti tijek rada iz sekcije Actions vašeg GitHub repozitorija.

Objašnjenje YAML konfiguracije on: schedule definira raspored za pokretanje tijeka rada pomoću cron sintakse. U ovom primjeru, tijek rada će se pokrenuti svakodnevno u ponoć. jobs definira zadatak koji se treba izvršiti. steps definira korake koje tijek rada treba izvesti: actions/checkout@v2 klonira repozitorij. shivammathur/setup-php@v2 postavlja PHP okruženje. actions/cache@v2 kešira Composer i npm ovisnosti radi bržeg pokretanja. composer install instalira PHP ovisnosti. npm install instalira npm ovisnosti. php artisan migrate --force pokreće migracije baze podataka. php artisan optimize:clear i slične naredbe čiste i keširaju različite dijelove aplikacije. composer update i npm update ažuriraju PHP i npm ovisnosti. php artisan backup:run pokreće izradu sigurnosne kopije koristeći Spatie Backup paket. php artisan schedule:run pokreće Laravelove zakazane zadatke. Ovim pristupom možete automatizirati različite zadatke održavanja vašeg Laravel projekta koristeći GitHub Actions.

Rekli ste: Wow odlično hvala! Kako u slučaju greške i/ili problema napraviti workflow za error reporting te error handling? ChatGPT je rekao: ChatGPT Da biste postavili error reporting i error handling u svom GitHub Actions workflow-u, možete koristiti nekoliko tehnika. Jedan od načina je dodavanje koraka koji će slati obavijesti ili prijaviti greške kada dođe do neuspjeha u bilo kojem dijelu workflow-a. U tu svrhu možete koristiti alate kao što su Slack, email obavijesti, ili neki od servisa za praćenje grešaka kao što je Sentry.

Evo primjera kako postaviti error reporting koristeći GitHub Actions:

Dodavanje koraka za obavijesti o greškama Koristit ćemo Slack za slanje obavijesti u slučaju greške. Možete prilagoditi ovaj primjer prema svojim potrebama.

Stvorite Slack webhook

Prvo, trebate stvoriti Slack webhook za kanal u koji želite slati obavijesti. Slijedite upute na Slack API za kreiranje webhook URL-a.

Dodajte Slack webhook URL u GitHub Secrets

Idite na vaš GitHub repozitorij, zatim idite na Settings > Secrets > Actions, i dodajte novi secret SLACK_WEBHOOK_URL s vrijednošću vašeg Slack webhook URL-a.

Prilagodite GitHub Actions workflow

Dodajte korake za slanje obavijesti u slučaju greške koristeći curl za slanje POST zahtjeva na Slack webhook URL.

yaml Kopiraj kod name: Maintenance

on: schedule: # Pokreće se svakodnevno u ponoć - cron: '0 0 * * *' workflow_dispatch: # Omogućuje ručno pokretanje

jobs: maintenance: runs-on: ubuntu-latest

```
steps:
  - name: Checkout repository
    uses: actions/checkout@v2

  - name: Set up PHP
    uses: shivammathur/setup-php@v2
    with:
      php-version: '8.0' # Postavite vašu verziju PHP-a

  - name: Cache Composer dependencies
    uses: actions/cache@v2
```

with:

path: vendor

key: \${{ runner.os }}-composer-\${{ hashFiles('**/composer.lock') }}

restore-keys: \${{ runner.os }}-composer-

- name: Install Composer dependencies

run: composer install --no-progress --prefer-dist --optimize-autoloader

- name: Cache npm dependencies

uses: actions/cache@v2

with:

path: node_modules

key: \${{ runner.os }}-node-\${{ hashFiles('**/package-lock.json') }}

restore-keys: \${{ runner.os }}-node-

- name: Install npm dependencies

run: npm install

- name: Run database migrations

run: php artisan migrate --force

- name: Run optimizations

run: |

php artisan optimize:clear

php artisan cache:clear

php artisan config:cache

php artisan route:cache

php artisan view:cache

- name: Run Composer updates

run: composer update

- name: Run npm updates

run: npm update

- name: Run Spatie Backup

env:

GOOGLE_DRIVE_CLIENT_ID: \${{ secrets.GOOGLE_DRIVE_CLIENT_ID }}

GOOGLE_DRIVE_CLIENT_SECRET: \${{ secrets.GOOGLE_DRIVE_CLIENT_SECRET }}

GOOGLE_DRIVE_REFRESH_TOKEN: \${{ secrets.GOOGLE_DRIVE_REFRESH_TOKEN }}

```
GOOGLE_DRIVE_FOLDER_ID: ${ secrets.GOOGLE_DRIVE_FOLDER_ID }
```

```
run: php artisan backup:run
```

```
- name: Run cleanup commands
```

```
run: |
```

```
php artisan schedule:run
```

```
error-reporting: needs: maintenance runs-on: ubuntu-latest if: failure() steps: - name: Send Slack notification env: SLACK_WEBHOOK_URL: ${ secrets.SLACK_WEBHOOK_URL } run: | curl -X POST -H 'Content-type: application/json' --data '{"text":"GitHub Actions Workflow Failed: ${GITHUB_WORKFLOW}"}' $SLACK_WEBHOOK_URL
```

Objašnjenje YAML konfiguracije Definiranje poslova (jobs)

maintenance job sadrži sve korake za održavanje kao što su instalacija ovisnosti, optimizacija, ažuriranja i izrada sigurnosne kopije. error-reporting job je definiran za pokretanje u slučaju neuspjeha (if: failure()) u maintenance jobu. Korištenje Slack webhooka

U error-reporting jobu koristimo curl za slanje POST zahtjeva na Slack webhook URL s informacijama o grešci. Povezivanje poslova

error-reporting job koristi needs: maintenance što znači da će se pokrenuti tek nakon što maintenance job završi, a uvjet if: failure() osigurava da se error-reporting job pokrene samo u slučaju greške. Ovim pristupom osiguravate da budete obaviješteni u slučaju problema ili grešaka u vašem GitHub Actions workflow-u, što omogućuje brzu reakciju i rješavanje problema. Možete prilagoditi korake za obavijesti prema svojim potrebama koristeći druge alate ili servise po izboru.

Rekli ste: Odlično hvala! Kako u slučaju nedostupnosti mog cloud servera osigurati failover cijelog app i baze ChatGPT je rekao: ChatGPT Da biste osigurali failover cijelog vašeg aplikacijskog i baznog sloja u slučaju nedostupnosti vašeg cloud servera, trebate postaviti visoko dostupnu (HA) infrastrukturu. Ovo uključuje replikaciju baze podataka, automatsko prebacivanje (failover) i balansiranje opterećenja. Evo koraka kako to možete postići:

1. Replikacija baze podataka Replikacija baze podataka omogućuje vam da imate kopiju vaše baze podataka na drugom serveru koji može preuzeti ulogu glavne baze u slučaju kvara.

Koraci za postavljanje replikacije: Odaberite bazu podataka:

Na primjeru MySQL:

Postavite glavnu (master) bazu podataka. Postavite jednu ili više replikacijskih (slave) baza podataka. Konfiguracija MySQL Master-Slave replikacije:

Na glavnom serveru (master):

sql Kopiraj kod -- U MySQL konfiguracijskoj datoteci (my.cnf) [mysqld] log-bin=mysql-bin server-id=1

-- Na MySQL konzoli CREATE USER 'replica'@'%' IDENTIFIED BY 'password'; GRANT REPLICATION SLAVE ON . TO 'replica'@'%'; FLUSH PRIVILEGES; FLUSH TABLES WITH READ LOCK; SHOW MASTER STATUS; Zabilježite File i Position iz izlaza naredbe SHOW MASTER STATUS;.

Na replikacijskom serveru (slave):

sql Kopiraj kod -- U MySQL konfiguracijskoj datoteci (my.cnf) [mysqld] server-id=2

-- Na MySQL konzoli CHANGE MASTER TO MASTER_HOST='master_ip', MASTER_USER='replica', MASTER_PASSWORD='password', MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=position; START SLAVE; SHOW SLAVE STATUS; Ovim koracima postavljate osnovnu MySQL replikaciju.

2. Balansiranje opterećenja i failover Koristite balansere opterećenja za raspodjelu prometa i osiguranje dostupnosti vaše aplikacije.

Koraci za postavljanje balansiranja opterećenja: Postavite Load Balancer:

Koristite AWS Elastic Load Balancer (ELB), Google Cloud Load Balancer, ili Nginx kao softverski load balancer. Konfigurirajte health checks za vaše servere. Konfiguracija Nginx kao Load Balancer:

nginx Kopiraj kod upstream backend { server backend1.example.com weight=5; server backend2.example.com; }

server { listen 80;

```
location / {
    proxy_pass http://backend;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

Rekli ste: Ok hvala više sam mislio možda i na plaćeni servis tipa laravel forge i/ili slično, te integracija s github actions ako je izvedivo. ChatGPT je rekao: ChatGPT Da, Laravel Forge može biti vrlo korisno rješenje za postavljanje i upravljanje vašom Laravel aplikacijom, uključujući failover i integraciju s GitHub Actions. Laravel Forge upravlja vašim serverima i može automatizirati mnoge zadatke, dok GitHub Actions može upravljati kontinuiranom integracijom i isporukom (CI/CD).

Korištenje Laravel Forge za visoko dostupnu infrastrukturu Postavljanje servera na Laravel Forge

Prijavite se na Laravel Forge. Dodajte novi server na željenom cloud provideru (AWS, DigitalOcean, Linode, itd.). Konfigurirajte server prema vašim potrebama. Postavljanje MySQL replikacije na Laravel Forge

Laravel Forge podržava postavljanje MySQL master-slave replikacije:

U Laravel Forge, odaberite vaš master server. Pod "Meta" odjeljak, pronađite opciju za postavljanje slave servera. Dodajte slave server i Forge će automatski konfigurirati replikaciju. Postavljanje balansiranja opterećenja

Laravel Forge također podržava balansiranje opterećenja:

U Laravel Forge, stvorite load balancer i povežite ga s vašim backend serverima. Forge će automatski konfigurirati Nginx ili HAProxy za balansiranje opterećenja. Integracija s GitHub Actions Integracija GitHub Actions s Laravel Forge omogućuje automatizirano postavljanje i upravljanje vašom aplikacijom.

Stvaranje deploy ključ na GitHub

Idite na vaš GitHub repozitorij, zatim idite na Settings > Deploy keys. Dodajte novi deploy key koji je generiran iz Laravel Forge. Postavljanje GitHub Actions za automatski deploy

Stvorite GitHub Actions workflow datoteku, npr. .github/workflows/deploy.yml:

yml Kopiraj kod name: Deploy

on: push: branches: - main

jobs: deploy: runs-on: ubuntu-latest

```
steps:
  - name: Checkout repository
    uses: actions/checkout@v2

  - name: Set up PHP
    uses: shivammathur/setup-php@v2
    with:
      php-version: '8.0' # Postavite vašu verziju PHP-a

  - name: Install Composer dependencies
    run: composer install --no-progress --prefer-dist --optimize-autoloader --no-dev

  - name: Install npm dependencies
    run: npm install --production

  - name: Run Laravel Forge deploy script
    env:
      FORGE_API_TOKEN: ${ secrets.FORGE_API_TOKEN }
```

```
FORGE_SERVER_ID: ${ secrets.FORGE_SERVER_ID }}
FORGE_SITE_ID: ${ secrets.FORGE_SITE_ID }}
run: |
  curl -X POST "https://forge.laravel.com/api/v1/servers/${FORGE_SERVER_ID}/sites/${FORGE_SITE_ID}/deploy" \
  -H "Authorization: Bearer ${FORGE_API_TOKEN}" \
  -H "Content-Type: application/json"
```

Dodavanje tajni (secrets) u GitHub

Idite na vaš GitHub repozitorij, zatim idite na Settings > Secrets > Actions.

Dodajte sljedeće tajne (secrets):

FORGE_API_TOKEN: Vaš Laravel Forge API token (možete ga generirati u vašem Laravel Forge korisničkom računu). FORGE_SERVER_ID: ID vašeg servera na Laravel Forge. FORGE_SITE_ID: ID vašeg site-a na Laravel Forge. Objašnjenje GitHub Actions Workflow:

on: push osigurava da se workflow pokreće svaki put kada se dogodi push na main granu. deploy job preuzima repozitorij, instalira Composer i npm ovisnosti, i pokreće Laravel Forge deploy script koristeći Forge API. Laravel Forge:

Automatski upravlja serverima, uključujući postavljanje Nginx-a, MySQL replikaciju, balansiranje opterećenja i druge zadatke održavanja. Integracija s GitHub omogućuje automatski deploy vašeg koda svaki put kada se dogodi push na specifičnu granu. Ovim pristupom osiguravate visoku dostupnost vaše aplikacije i automatski deploy koristeći Laravel Forge i GitHub Actions. To vam omogućuje da se fokusirate na razvoj aplikacije, dok se infrastruktura automatski održava i optimizira.