



# Automate all the things with CI/CD in GitHub Actions

Rob Allen, May 2024



# How do we test and release software?

# Workflow to accept a code change

1. Checkout the source code
2. Install dependencies
3. Compile (or create container)
4. Run code style checks
5. Run tests
6. Send artifacts (logs, test output, etc.) to dev for debugging
7. Tell dev that it worked (or failed)

# Workflow to release a new version

1. Checkout the source code
2. Compile (or create container)
3. Upload container to registry (exe to Release)
4. Deploy to container orchestration platform
5. Publish release
6. Notify Slack

We never get this right  
every time!

Humans are bad at repetitive tasks

Humans are bad at repetitive tasks  
That's why we invented computers

Tests ensure our software works

CI ensures that we run them

CD releases it reliably



Our repository is the centre  
of our development world

GitHub Actions runs scripts  
when an event happens

# YAML all the way down!

sorry!



# .github/workflows/ci.yml

```
name: CI
on: [push, pull_request]
jobs:
  qa:
    name: QA checks
    runs-on: ubuntu-latest
    steps:
      - name: "Say Hello"
        run: echo "Hello World"
      - name: "Say Goodbye"
        run: echo "All done"
```

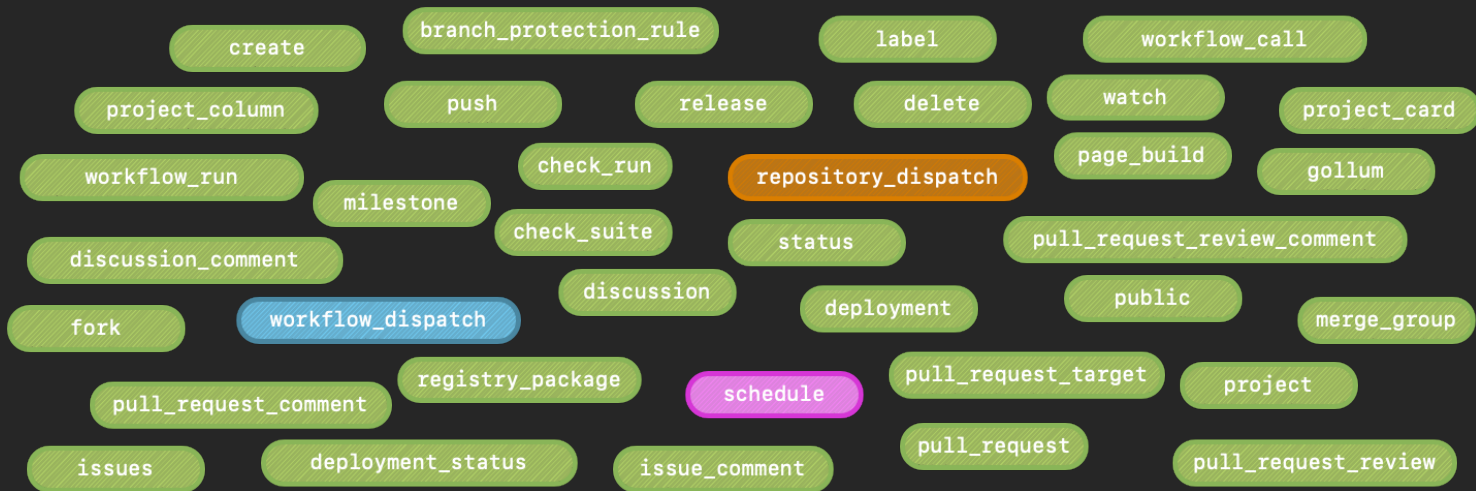
# .github/workflows/ci.yml

```
name: CI
on: [push, pull_request]
jobs:
  qa:
    name: QA checks
    runs-on: ubuntu-latest
    steps:
      - name: "Say Hello"
        run: echo "Hello World"
      - name: "Say Goodbye"
        run: echo "All done"
```

# .github/workflows/ci.yml

```
name: CI
on: [push, pull_request]
jobs:
  qa:
    name: QA checks
    runs-on: ubuntu-latest
    steps:
      - name: "Say Hello"
        run: echo "Hello World"
      - name: "Say Goodbye"
        run: echo "All done"
```

# Events



# .github/workflows/ci.yml


```
name: CI
on: [push, pull_request]
jobs:
  qa:
    name: QA checks
    runs-on: ubuntu-latest
    steps:
      - name: "Say Hello"
        run: echo "Hello World"
      - name: "Say Goodbye"
        run: echo "All done"
```



# .github/workflows/ci.yml

```
name: CI
on: [push, pull_request]
jobs:
  qa:
    name: QA checks
    runs-on: ubuntu-latest
    steps:
      - name: "Say Hello"
        run: echo "Hello World"
      - name: "Say Goodbye"
        run: echo "All done"
```

# Success



**QA checks** Beta Give feedback  





Started 3m 35s ago

- >  Set up job 0s
- ▼  Say Hello 0s
  - 1 ▼ Run echo "Hello World"
  - 2 `echo "Hello World"`
  - 3 `shell: /usr/bin/bash -e {0}`
  - 4 Hello World
- ▼  Say Goodbye 0s
  - 1 ▼ Run echo "All done"
  - 2 `echo "All done"`
  - 3 `shell: /usr/bin/bash -e {0}`
  - 4 All done
- >  Complete job 1s

# Failure

**QA checks**  
failed now in 0s

Beta Give feedback   

- >  Set up job 0s
- ▼  Say Hello 0s
  - 1 ▶ Run echo "Hello World" && exit 1
  - 4 Hello World
  - 5 **Error:** Process completed with exit code 1.
-  Say Goodbye 0s
- >  Complete job 0s



# PHP quality checks

# Set up the pipeline

```
name: PHP Checks
on: [pull_request]
jobs:
  php-checks:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3

      - name: Create .env file
        run: cp .env.ci .env
```

# Set up the pipeline

```
name: PHP Checks
on: [pull_request]
jobs:
  php-checks:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3

      - name: Create .env file
        run: cp .env.ci .env
```

# Set up the pipeline

```
name: PHP Checks
on: [pull_request]
jobs:
  php-checks:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3

      - name: Create .env file
        run: cp .env.ci .env
```

# Grab PHP

```
- name: Install PHP
  uses: "shivammathur/setup-php@v2"
  with:
    coverage: "pcov"
    php-version: "8.3.4"
    tools: composer:v2, cs2pr
```



# Dependencies

- `name`: Run composer  
`run`: `composer install --prefer-dist --no-progress --no-ansi --no-interaction`
- `name`: Install npm  
`run`: `npm install`



# Code quality

- `name`: Check code style  
`run`: `vendor/bin/phpcs -q --report=checkstyle | cs2pr`



# Code quality

- name: Check code style  
run: `vendor/bin/phpcs -q --report=checkstyle | cs2pr`
- name: Run static analysis checks  
run: `vendor/bin/phpstan analyse`

# Code quality

- name: Check code style  
run: `vendor/bin/phpcs -q --report=checkstyle | cs2pr`
- name: Run static analysis checks  
run: `vendor/bin/phpstan analyse`
- name: Run unit tests  
run: `vendor/bin/phpunit -c phpunit-ci.xml --testsuite=unit`

# Other checks

- `name`: Check licenses of PHP dependencies  
# (see [akrobat.com/check-licenses-of-composer-dependencies](https://social.akrobat.com/check-licenses-of-composer-dependencies))  
`run`: `php bin/check-licenses.php`



# Other checks

- name: Check licenses of PHP dependencies  
# (see [akrabat.com/check-licenses-of-composer-dependencies](https://akrabat.com/check-licenses-of-composer-dependencies/))  
run: `php bin/check-licenses.php`
  
- name: Check we can cache routes  
run: `php bash -c "php artisan route:cache  
&& php artisan route:clear"`



# Other checks

- name: Check licenses of PHP dependencies  
# (see [akrobat.com/check-licenses-of-composer-dependencies](https://akrobat.com/check-licenses-of-composer-dependencies))  
run: `php bin/check-licenses.php`
- name: Check we can cache routes  
run: `php bash -c "php artisan route:cache  
&& php artisan route:clear"`
- name: Check tailwind-build has been run.  
run: `npm run tailwind-build  
&& [ -z "$(git status --porcelain)" ]`



# Use Docker? Run in Docker!

- `name: Docker Compose Pull`  
`run: docker compose pull`





# Use Docker? Run in Docker!

```
- name: Docker Compose Pull  
  run: docker compose pull
```

```
# Cache Docker layers
```

```
- uses: jpribyl/action-docker-layer-caching@v0.1.1  
  continue-on-error: true
```



# Use Docker? Run in Docker!

- name: Docker Compose Pull  
run: docker compose pull

- # Cache Docker layers

- uses: jpribyl/action-docker-layer-caching@v0.1.1  
continue-on-error: true

- name: Start the containers  
run: docker compose up --build -d

# Tests that need the database

- `name`: Ensure MySQL is available  
# (uses `raphaelahrens/wait-for-it`)  
`run`: `docker-compose exec -T php ./wait-for-it -t 10 db:3306`



# Tests that need the database

- name: Ensure MySQL is available  
# (uses raphaelahrens/wait-for-it)  
run: docker-compose exec -T php ./wait-for-it -t 10 db:3306
- name: Run migrations  
run: docker-compose exec -T php bash -c  
"php artisan migrate:fresh --seed"

# Tests that need the database

- name: Ensure MySQL is available  
# (uses raphaelahrens/wait-for-it)  
run: docker-compose exec -T php ./wait-for-it -t 10 db:3306
- name: Run migrations  
run: docker-compose exec -T php bash -c  
"php artisan migrate:fresh --seed"
- name: Execute tests  
run: docker-compose exec -T vendor/bin/phpunit  
-c phpunit-ci.xml --testsuite=integration

# Upload assets

```
- name: Upload test output
  uses: actions/upload-artifact@v2
  if: failure()
  with:
    name: failed-tests
    path: tests/output
    retention-days: 8
```



# Upload assets

```
- name: Upload test output
  uses: actions/upload-artifact@v4
  if: failure()
  with:
    name: failed-tests
    path: tests/output
    retention-days: 8
```



# Upload assets

```
- name: Upload test output
  uses: actions/upload-artifact@v4
  if: failure()
  with:
    name: failed-tests
    path: tests/output
    retention-days: 8
```





Everything we run in CI  
we also run locally

# Tag and Release



# When a milestone is closed...

```
on:  
  milestone:  
    types: [closed]
```

# do a full checkout...

steps:

- name: Checkout code
- uses: actions/checkout@v3
- with:
  - ref: main
  - fetch-depth: 0

# so we can create & push a tag...

```
- name: Create Tag
  uses: rickstaa/action-create-tag@v1
  id: create-tag
  with:
    tag: "${{ github.event.milestone.title }}"
    message: "Tag ${{ github.event.milestone.title }}"
```



# and create a GitHub Release

```
- name: Create GitHub Release
  uses: actions/github-script@v6
  with:
    script: |
      await github.rest.repos.createRelease({
        generate_release_notes: true,
        name: "${{github.event.milestone.title}}",
        tag_name: "${{github.event.milestone.title}}"
      });
```



# and create a GitHub Release

```
- name: Create GitHub Release
  uses: actions/github-script@v6
  with:
    script: |
      await github.rest.repos.createRelease({
        generate_release_notes: true,
        name: "${{github.event.milestone.title}}",
        tag_name: "${{github.event.milestone.title}}"
      });
```



# along with a new Milestone

```
- name: 'Get next minor version'  
  id: semvers  
  uses: "WyriHaximus/github-action-next-semvers@v1"  
  with:  
    version: ${{github.event.milestone.title}}  
- name: 'Create new milestone'  
  uses: "WyriHaximus/github-action-create-milestone@v1"  
  with:  
    title: ${{ steps.semvers.outputs.patch }}  
  env:  
    GITHUB_TOKEN: "${{ secrets.GITHUB_TOKEN }}"
```



# along with a new Milestone

```
- name: 'Get next minor version'  
  id: semvers  
  uses: "WyrriHaximus/github-action-next-semvers@v1"  
  with:  
    version: "${{github.event.milestone.title}}"
```

```
- name: 'Create new milestone'  
  uses: "WyrriHaximus/github-action-create-milestone@v1"  
  with:  
    title: "${{ steps.semvers.outputs.patch }}"  
  env:  
    GITHUB_TOKEN: "${{ secrets.GITHUB_TOKEN }}"
```



# along with a new Milestone

```
- name: 'Get next minor version'  
  id: semvers  
  uses: "WyriHaximus/github-action-next-semvers@v1"  
  with:  
    version: "${{github.event.milestone.title}}"
```

```
- name: 'Create new milestone'  
  uses: "WyriHaximus/github-action-create-milestone@v1"  
  with:  
    title: "${{ steps.semvers.outputs.patch }}"  
  env:  
    GITHUB_TOKEN: "${{ secrets.GITHUB_TOKEN }}"
```



# along with a new Milestone

```
- name: 'Get next minor version'  
  id: semvers  
  uses: "WyriHaximus/github-action-next-semvers@v1"  
  with:  
    version: ${{github.event.milestone.title}}  
- name: 'Create new milestone'  
  uses: "WyriHaximus/github-action-create-milestone@v1"  
  with:  
    title: ${{ steps.semvers.outputs.patch }}  
env:  
  GITHUB_TOKEN: "${{ secrets.GITHUB_TOKEN }}"
```



# Compile & upload binaries



# When a release is published...

```
on:  
  release:  
    types:  
      - published
```

# build the binaries...

steps:

```
# checkout, setup Go etc...
```

```
- name: Build the Rodeo executables
```

```
# (akrabat.com/building-go-binaries-for-different-platforms)
```

```
run: ./build-exes.sh ${{ github.ref_name }}
```



# and upload them

```
- name: Upload the Rodeo binaries
  uses: actions/svenstaro/upload-release-action@v2
  with:
    repo_token: ${{ secrets.GITHUB_TOKEN }}
    tag: ${{ github.ref }}
    file: ./release/rodeo-*
    file_glob: true
```



# Build and push to ECR





# Build container...

```
on:  
  release:  
    types:  
      - published
```

# Build container...

```
on:  
  release:  
    types:  
      - published
```

## steps:

```
# checkout, etc...
```

```
- name: Build Docker Image
```

```
  run: docker build --tag
```

```
    img-name:${{ github.ref_name }} .
```

# and push to ECR

- name: Push to ECR

uses: jwalton/gh-ecr-push@v1

with:

access-key-id: \${{ secrets.AWS\_ACCESS\_KEY\_ID }}

secret-access-key: \${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

region: us-east-2

local-image: img-name:\${{ github.ref\_name }}

image: img-name:\${{ github.ref\_name }}, img-name:latest

# and push to ECR

- name: Push to ECR

uses: jwalton/gh-ecr-push@v1

with:

access-key-id: \${{ secrets.AWS\_ACCESS\_KEY\_ID }}

secret-access-key: \${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

region: us-east-2

local-image: img-name:\${{ github.ref\_name }}

image: img-name:\${{ github.ref\_name }}, img-name:latest

# and push to ECR

- name: Push to ECR

uses: jwalton/gh-ecr-push@v1

with:

access-key-id: \${{ secrets.AWS\_ACCESS\_KEY\_ID }}

secret-access-key: \${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

region: us-east-2

local-image: img-name:\${{ github.ref\_name }}

image: img-name:\${{ github.ref\_name }}, img-name:latest

More!

# More!

- Secrets live in GitHub, not git!
- Use conditionals to save time & resources
- Don't like bash? Use Python with `shell: python`
- The GitHub cli (`gh`) is preinstalled
- Building a library? Use matrices to test on multiple PHPs
- Pre-built: <https://github.com/marketplace?type=actions>



# Laminas Automatic Releases

Assuming your project has Github Actions enabled, each time you close a milestone, this action will perform all following steps (or stop with an error):

1. determine if all issues and pull requests associated with this milestone are closed
2. determine if the milestone is named with the SemVer `x.y.z` format
3. create a changelog by looking at the milestone description and associated issues and pull requests
4. select branch `x.y.z` for the release (e.g. `1.1.x` for a `1.1.0` release)
5. create a tag named `x.y.z` on the selected branch, with the generated changelog
6. publish a release named `x.y.z`, with the generated tag and changelog
7. create (if applicable), a pull request from the selected branch to the next release branch
8. create (if necessary) a "next minor" release branch `x.y+1.z`
9. switch default repository branch to newest release branch





To sum up



*“a deployment pipeline is an automated manifestation of your process for getting software from version control into the hands of your users.”*

David Farley





Thank you!