Rapid WordPress (Plugin) Development

Albert Šuntić

- Living in Podgorica, Montenegro 🇲🇪
- Father of two and husband
- Passionate **PHP** developer with over 20 years of experience in the field
- Working at **ServMask** on developing new features for **All In One WP Migration** plugin and extensions
- Sport (football) lover

How did it start?

How did it start?

Have you ever felt frustrated by the slow pace of development?

Let me share a story that might sound familiar.

Two years ago, at ServMask, I discovered the power of rapid development tools that changed everything for me....

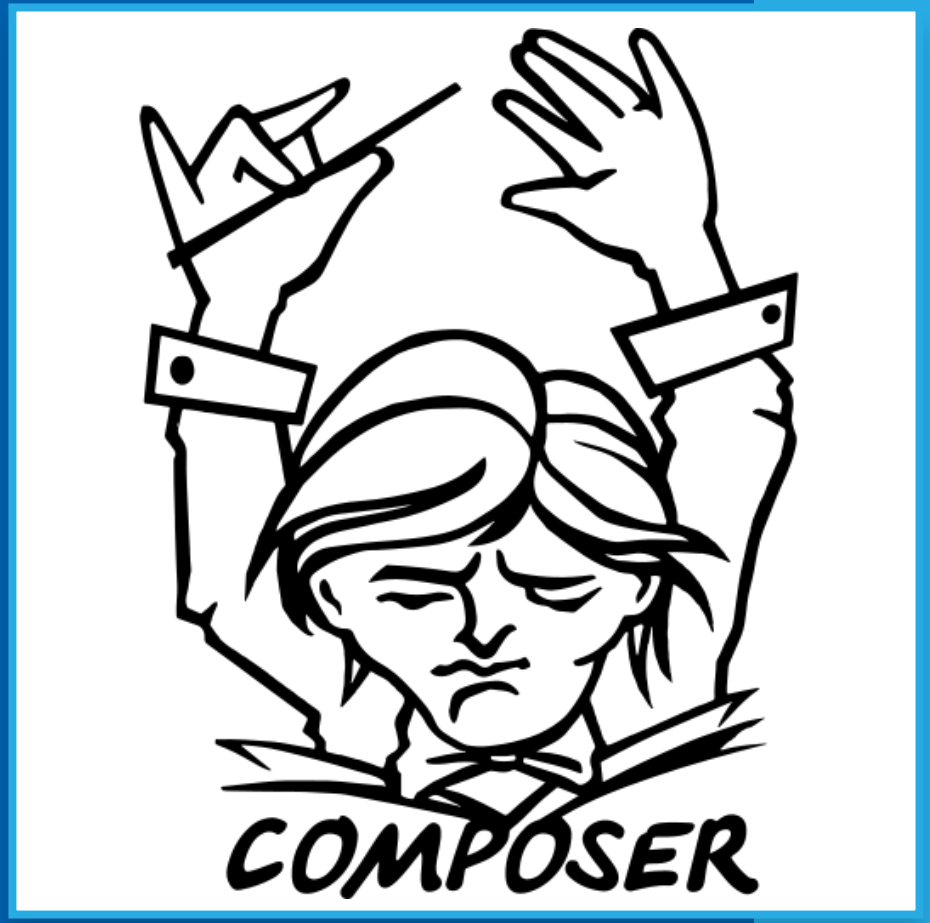

Engineering › ENG-1636
Add missi
All = 19
We have WP CLI
not available in the
rest of the cloud extensi
others, you will need to
check). This task is
s into all extensions
into all extensions

Composer

Composer

Composer offers several key benefits

- Efficiency
- Consistency
- Reusability
- Version Control
- Automation



Understanding Composer Packages

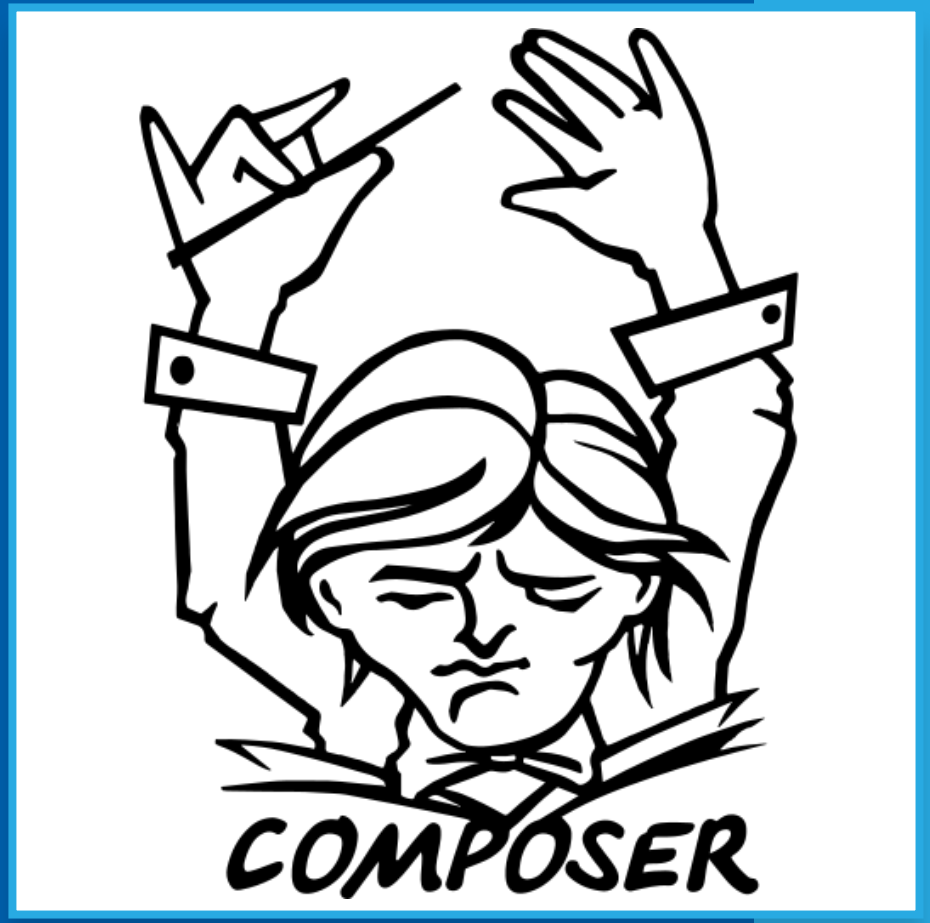
- Packages are essential building blocks in software development, providing reusable components and functionality.
- They encapsulate code, libraries, or resources that solve specific problems or fulfil certain requirements.

Types of Packages

1. Library
2. Framework
3. Utility
4. Application

Benefits of Composer Packages

1. Modularity
2. Reusability
3. Scalability
4. Community



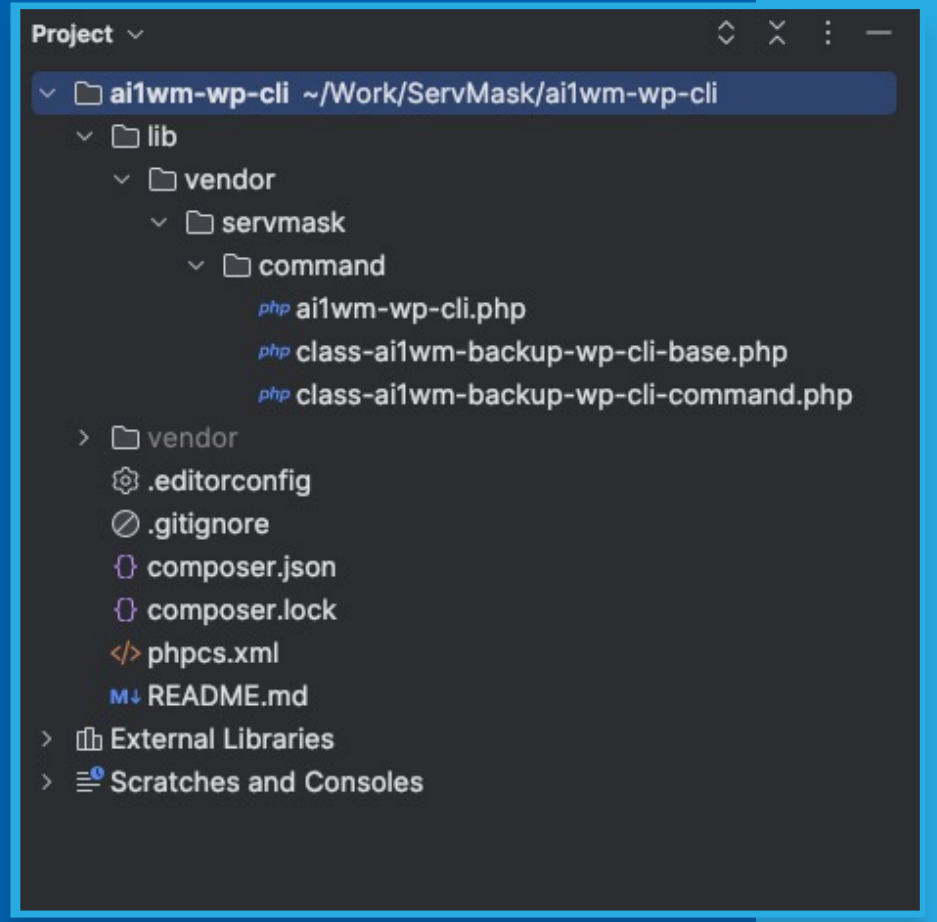
Composer Private Packages

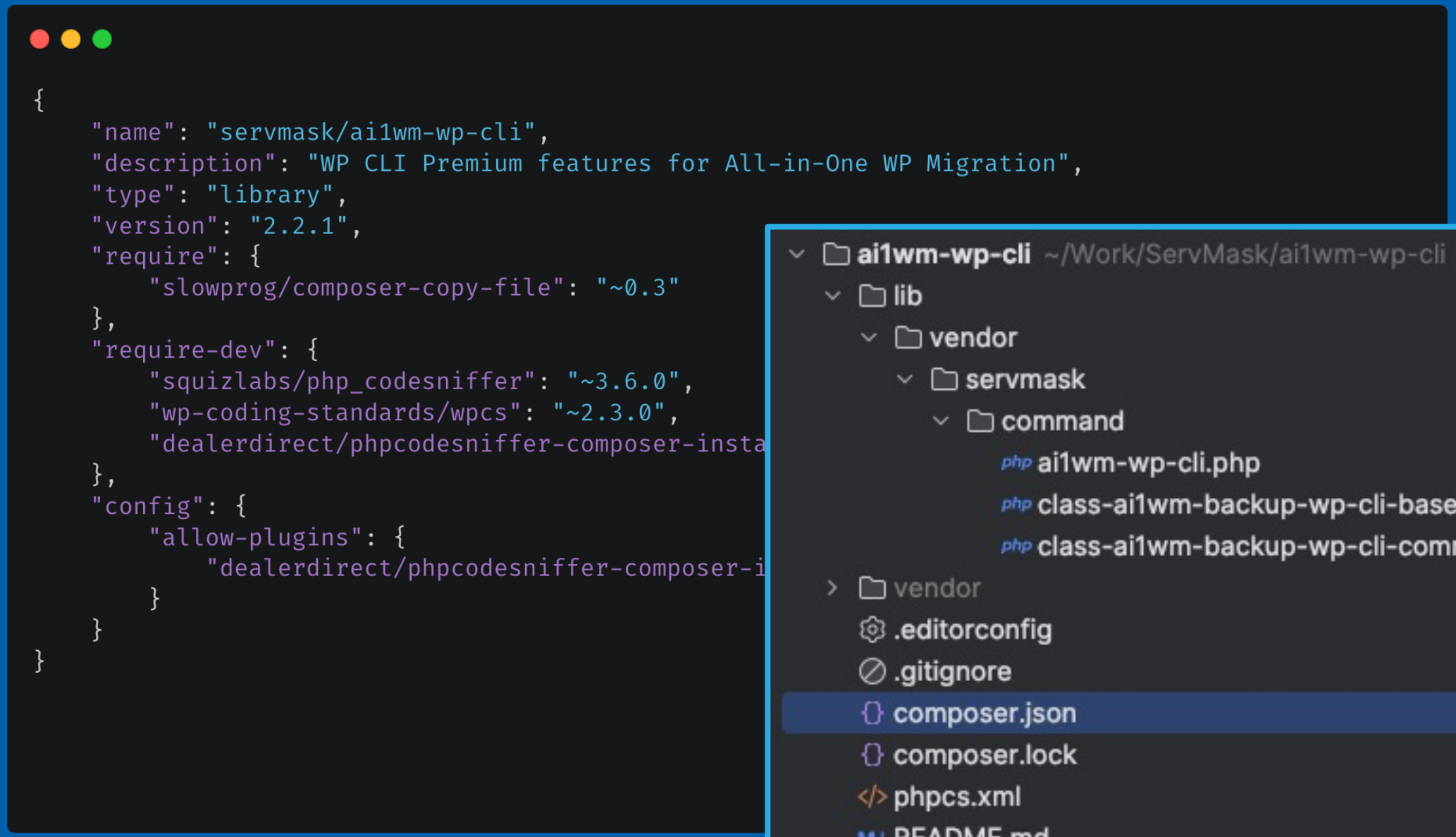
What are Private Packages?

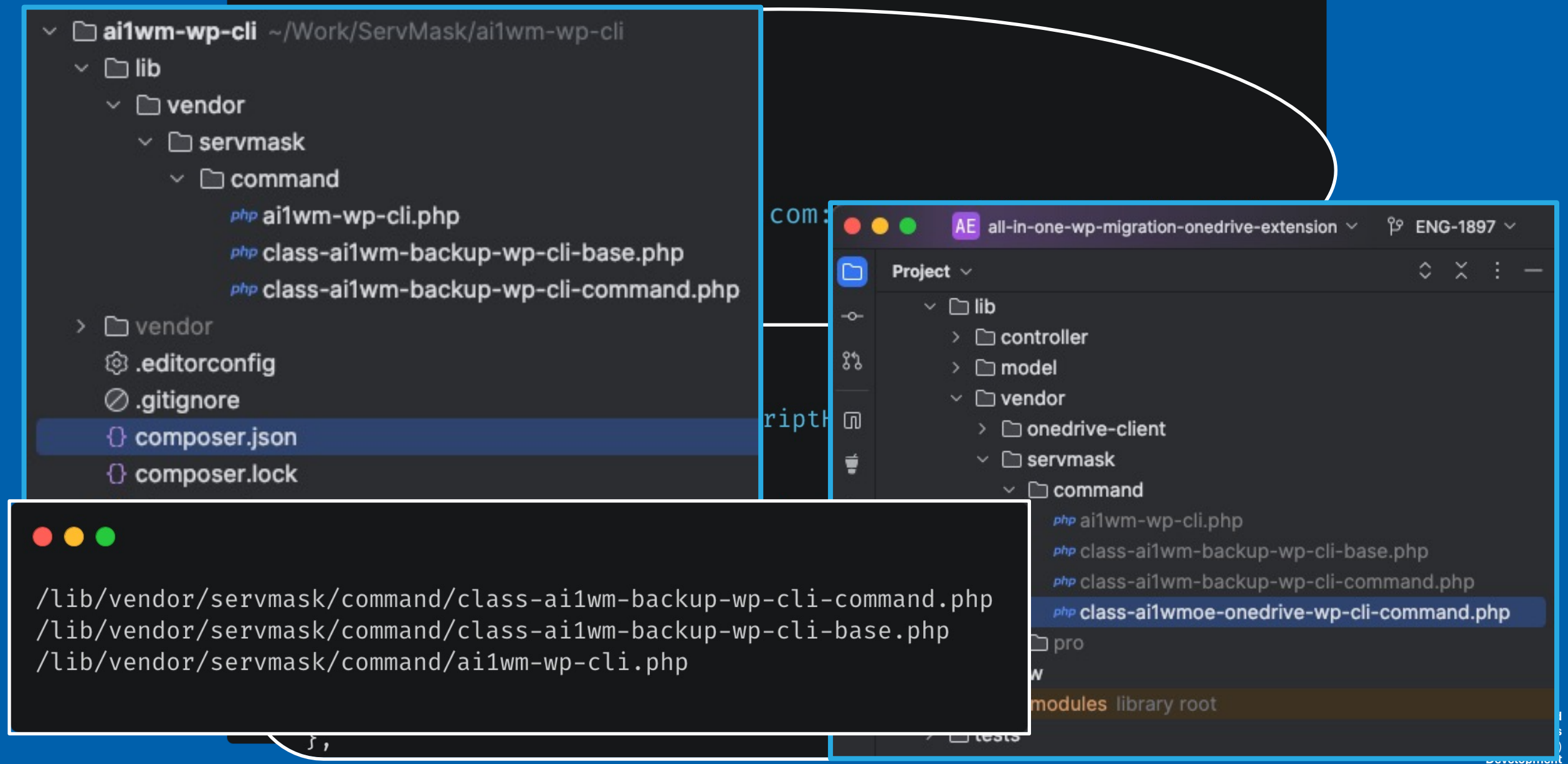
- Private packages are Composer packages that are not publicly available on Packagist or other repositories.
- They are typically internal libraries developed by organizations for their own use.

Setting Up Private Packages

- Private packages are hosted in private repositories, such as GitHub, GitLab, or Bitbucket.
- Composer supports authentication with these repositories using tokens or SSH keys.
- Once authenticated, private packages can be installed and managed just like public packages.




{
"name": "servmask/ai1wm-wp-cli",
"description": "WP CLI Premium features for All-in-One WP Migration",
"type": "library",
"version": "2.2.1",
"require": {
"slowprog/composer-copy-file": "~0.3"
},
"require-dev": {
"squizlabs/php_codesniffer": "~3.6.0",
"wp-coding-standards/wpcs": "~2.3.0",
"dealerdirect/phpcodesniffer-composer-insta
},
"config": {
"allow-plugins": {
"dealerdirect/phpcodesniffer-composer-i
}
}
}
ai1wm-wp-cli ~/Work/ServMask/ai1wm-wp-cli
lib
vendor
servmask
command
ai1wm-wp-cli.php
class-ai1wm-backup-wp-cli-base.php
class-ai1wm-backup-wp-cli-command.php
vendor
.editorconfig
.gitignore
composer.json
composer.lock
phpcs.xml
README.md


ai1wm-wp-cli ~/Work/ServMask/ai1wm-wp-cli
lib
vendor
servmask
command
ai1wm-wp-cli.php
class-ai1wm-backup-wp-cli-base.php
class-ai1wm-backup-wp-cli-command.php
vendor
.editorconfig
.gitignore
composer.json
composer.lock
/lib/vendor/servmask/command/class-ai1wm-backup-wp-cli-command.php
/lib/vendor/servmask/command/class-ai1wm-backup-wp-cli-base.php
/lib/vendor/servmask/command/ai1wm-wp-cli.php
all-in-one-wp-migration-onedrive-extension
ENG-1897
Project
lib
controller
model
vendor
onedrive-client
servmask
command
ai1wm-wp-cli.php
class-ai1wm-backup-wp-cli-base.php
class-ai1wm-backup-wp-cli-command.php
class-ai1wmoe-onedrive-wp-cli-command.php
pro
modules library root

ENG-1636

Links

#57 ENG-1636 Add missing WP CLI commands — Merged 1 year ago

#129 ENG-1636 Add missing WP CLI commands

#45 ENG-1636 Add

#76 ENG-1636 Add

#139 ENG-1636 Ad

#52 ENG-1636 Add

#52 ENG-1636 Add missing WP CLI commands

#113 ENG-1636 Add missing WP CLI commands

#229 ENG-1636 Add missing WP CLI commands

#71 ENG-1636 Add missing WP CLI commands

#53 ENG-1636 Add missing WP CLI commands — Merged 1 year ago

erged 1 year ago

erged 1 year ago

erged 1 year ago

erged 1 year ago

erged 1 year ago

#96 ENG-1636 Add missing WP CLI commands — Merged 1 year ago

#69 ENG-1636 Add missing WP CLI commands — Merged 1 year ago

#48 ENG-1636 Add missing WP CLI commands — Merged 1 year ago

#231 ENG-1636 Add missing WP CLI commands — Merged 1 year ago

developeritsme commented on May 31, 2022

Implement base WP CLI commands as a composer package
Change local, and GH Action build scripts

Development

GitHub Actions

GitHub Actions

- It is a game-changer in the world of software development and deployment, and it's no different for WordPress developers.
- It is a powerful and flexible automation tool offered by GitHub, which is tightly integrated with your code repositories.
- It allows you to define workflows that automate various tasks related to your GitHub-hosted projects. Whether it's **building, testing, deploying**, or any other task that can be scripted, GitHub Actions can handle it.

GitHub Actions

Key Features

- **Workflow Automation**
- **Extensive Ecosystem**
- **Integration with GitHub**
- **Cross-Platform Support**

GitHub Actions

Components

- **Workflow Files**
- **Actions**
- **Events**
- **Jobs and Steps**

GitHub Actions

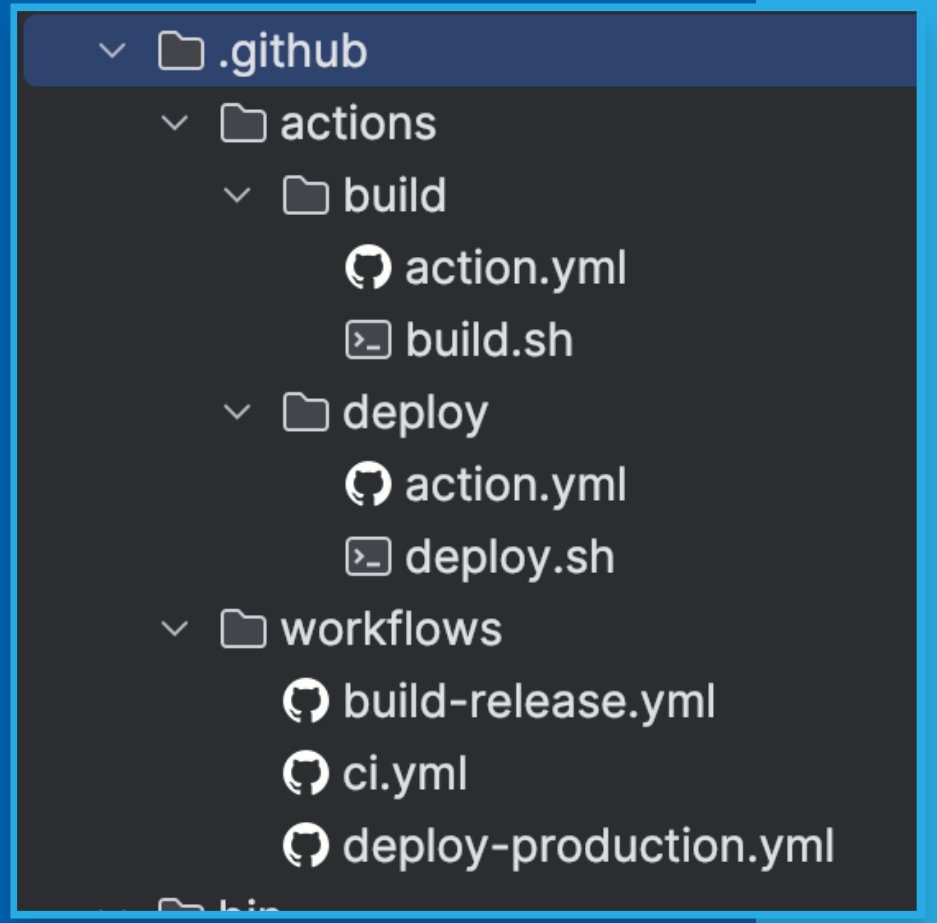
Practical Use Cases

- **Continuous Integration (CI)**
- **Continuous Deployment (CD)**
- **Automated Code Reviews**
- **Scheduled Tasks**

GitHub Actions

Getting Started

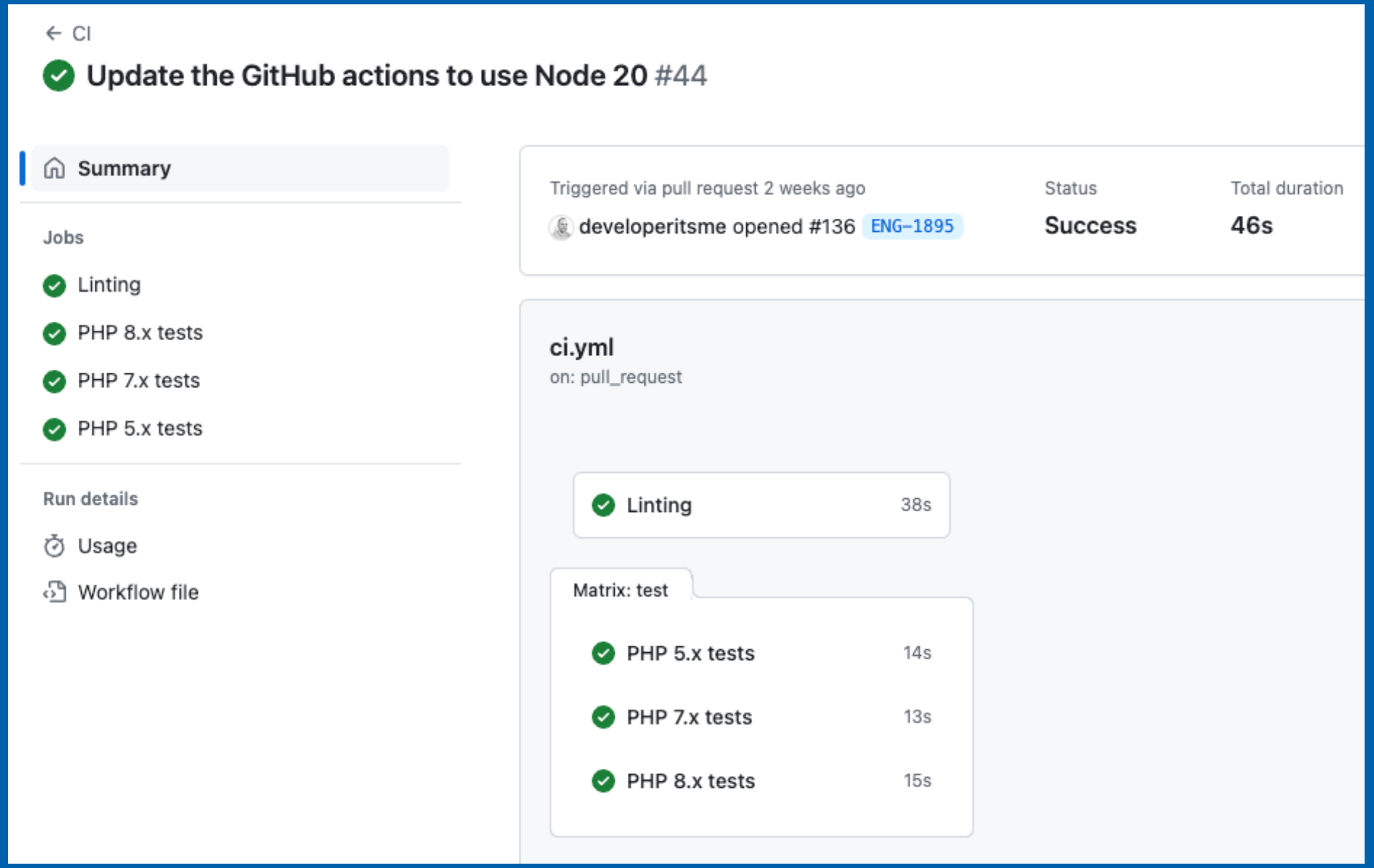
- **Creating Workflows**
- **Using Actions**
- **Testing Workflows**
- **Deploying Workflows**



```yaml
name: CI

on: [ pull_request ]

jobs:
    lint:
        runs-on: ubuntu-latest
        name: Linting
        steps:
            -   name: Checkout
                uses: actions/checkout@v4
            -   name: Install node
                uses: actions/setup-node@v4
                with:
                    node-version: '16'
            -   name: Cache npm dependencies
                uses: actions/cache@v4
                with:
                    path: node_modules
                    key: node-modules-${{ hashFiles('package-lock.json') }}
            -   name: Install PHP
                uses: shivammathur/setup-php@v2
                with:
                    php-version: '7.4'
                run: |
                    composer config --global github-oauth.github.com ${{ secrets.GH_CI_ACCESS_TOKEN }}
                    composer install --no-interaction --prefer-dist --optimize-autoloader
                    npm install
                    ./vendor/bin/phpcs -p --report=full
                    ./node_modules/.bin/eslint --ext .js --ext .vue ./lib/view/assets-development/javascript -f codeframe
```

Rapid WordPress (Plugin) Development

```yaml
    runs-on: ubuntu-latest
    strategy:
        matrix:
            php-versions: [ '8.x', '7.x' ]
            phpunit-versions: [ '8.5' ]
            include:
                -   php-versions: '5.x'
                    phpunit-versions: '5.7'
    name: PHP ${{ matrix.php-versions }} te
name
    steps:
        -   name: Checkout
            uses: actions/checkout@v4
        -   name: Install PHP
            uses: shivammathur/setup-php@v2
            with:
                php-version: ${{ matrix.php-versions }}
                tools: composer:v2, phpunit:${{ matrix.phpunit-versions }}
        -   name: Run Tests
            run: |
                composer config --global github-oauth.github.com ${{ secrets.GH_CI_ACCESS_TOKEN }}
                composer install --no-interaction --prefer-dist --optimize-autoloader
                phpunit -c phpunit.xml
```



```yaml
name: Build Release

on:
    release:
        types:
            - prereleased
            - released

jobs:
    build-release:
        runs-on: ubuntu-latest
        name: Build Release
        env:
            PLUGIN_NAME: 'all-in-one-wp-migration-onedrive-extension'
            PRODUCT_SLUG: 'onedrive-extension'
        steps:
            -   name: Checkout
                uses: actions/checkout@v4
                with:
                    fetch-depth: 0

            -   name: Set environment variables
                run: |
                    echo "HOME_DIR=${{ github.workspace }}/dist" >> $GITHUB_ENV
                    echo "VERSION=${{ github.ref_name }}" >> $GITHUB_ENV
                    echo "BRANCH=$(git rev-parse --abbrev-ref HEAD)" >> $GITHUB_ENV

            -   name: Install PHP
                uses: shivammathur/setup-php@v2
                with:
                    php-version: '7.4'
                    tools: composer:v2
```

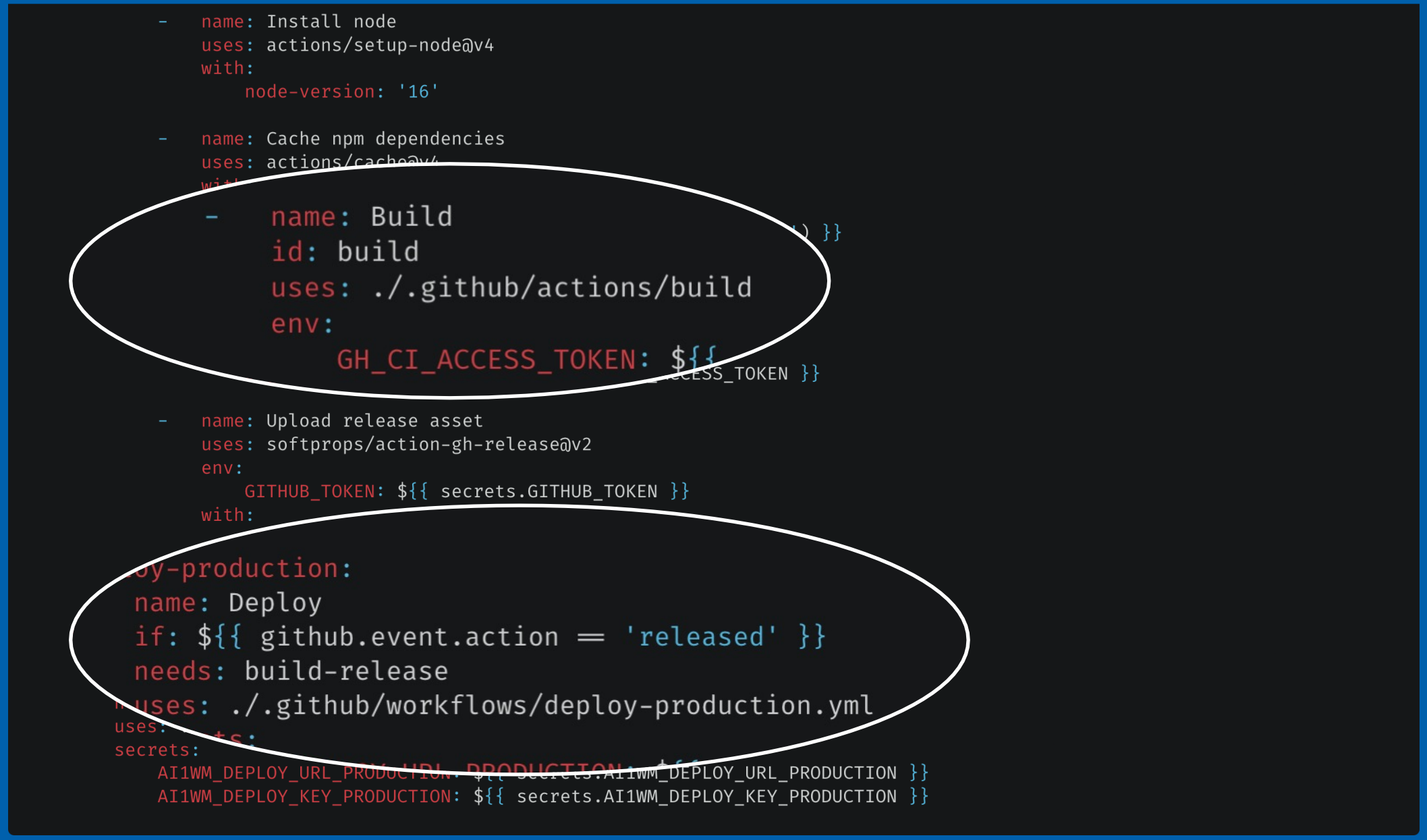
Rapid WordPress (Plugin) **Development**

```yaml
      - name: Install node
        uses: actions/setup-node@v4
        with:
          node-version: '16'

      - name: Cache npm dependencies
        uses: actions/cache@v4
        with:

      - name: Build
        id: build
        uses: ./.github/actions/build
        env:
          GH_CI_ACCESS_TOKEN: ${{ secrets.GH_CI_ACCESS_TOKEN }}

      - name: Upload release asset
        uses: softprops/action-gh-release@v2
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        with:
          files: ${{ steps.build.outputs.ASSET_PATH }}


  deploy-production:
    name: Deploy
    if: ${{ github.event.action == 'released' }}
    needs: build-release
    uses: ./.github/workflows/deploy-production.yml
    secrets:
      AI1WM_DEPLOY_URL_PRODUCTION: ${{ secrets.AI1WM_DEPLOY_URL_PRODUCTION }}
      AI1WM_DEPLOY_KEY_PRODUCTION: ${{ secrets.AI1WM_DEPLOY_KEY_PRODUCTION }}
```

```yaml
name: Build

description: Build files for deployment

outputs:
    DEPLOY:
        description: Deploy strategy
        value: ${{ steps.build-sh.outputs.DEPLOY }}
    ZIP_CHECKSUM:
        description: Checksum of zip file
        value: ${{ steps.build-sh.outputs.ZIP_CHECKSUM }}
    ASSET_PATH:
        description: Asset path to upload
        value: ${{ steps.build-sh.outputs.ASSET_PATH }}
    ASSET_NAME:
        description: Asset name to upload
        value: ${{ steps.build-sh.outputs.ASSET_NAME }}

runs:
    using: composite
    steps:
        -   id: build-sh
            run: ${{ github.action_path }}/build.sh
            shell: bash
```

```bash
#!/bin/bash

sudo apt-get install -y zip
echo "Building $VERSION on $BRANCH"
composer config --global github-oauth.github.com $GH_CI_ACCESS_TOKEN
composer install
npm install
npm run build
echo "Preparing distribution ... "
rm -rf dist
mkdir -p dist/$PLUGIN_NAME
cp -R {End-User-License-Agreement.txt,lib,
$PLUGIN_NAME.php,constants.php,exceptions.php,functions.php,loader.php,readme.txt,uninstall.php,User-
Guide.txt} dist/$PLUGIN_NAME
rm -rf dist/$PLUGIN_NAME/lib/view/assets-development
sed -i s/Version:\ develop/Version:\ $VERSION/ dist/$PLUGIN_NAME/$PLUGIN_NAME.php
sed -i s/define\(\ \'AI1WMOE_VERSION\',\ \'develop\'\ \)/define\(\ \'AI1WMOE_VERSION\',\ \'$VERSION\'\ \)/
dist/$PLUGIN_NAME/constants.php
echo -n $VERSION > dist/$PLUGIN_NAME/VERSION
cd dist
zip -vrm9 $PLUGIN_NAME.zip ./$PLUGIN_NAME
ZIP_CHECKSUM=$(sha256sum $PLUGIN_NAME.zip | awk '{ print $1 }')
if [[ $BRANCH =~ master$|HEAD$ && $VERSION =~ ^[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
    DEPLOY="production"
elif [[ $BRANCH =~ master$|hotfix.*|HEAD$ ]]; then
    DEPLOY="development"
else
    DEPLOY="none"
fi
echo "DEPLOY=$DEPLOY" >> $GITHUB_OUTPUT
echo "ZIP_CHECKSUM=$ZIP_CHECKSUM" >> $GITHUB_OUTPUT
echo "ASSET_PATH=dist/$PLUGIN_NAME.zip" >> $GITHUB_OUTPUT
echo "ASSET_NAME=$PLUGIN_NAME-$VERSION.zip" >> $GITHUB_OUTPUT
```

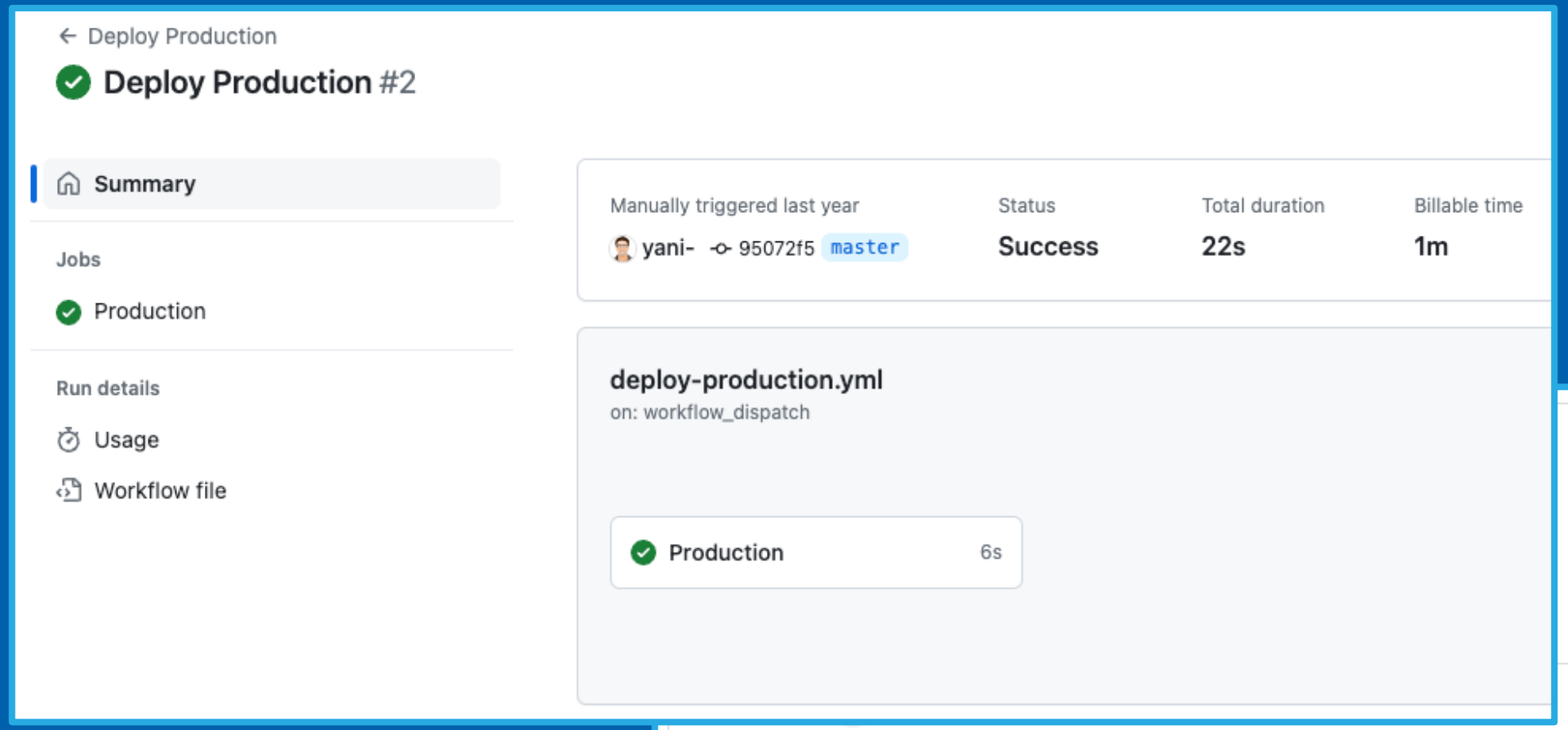
Rapid WordPress (Plugin) Development

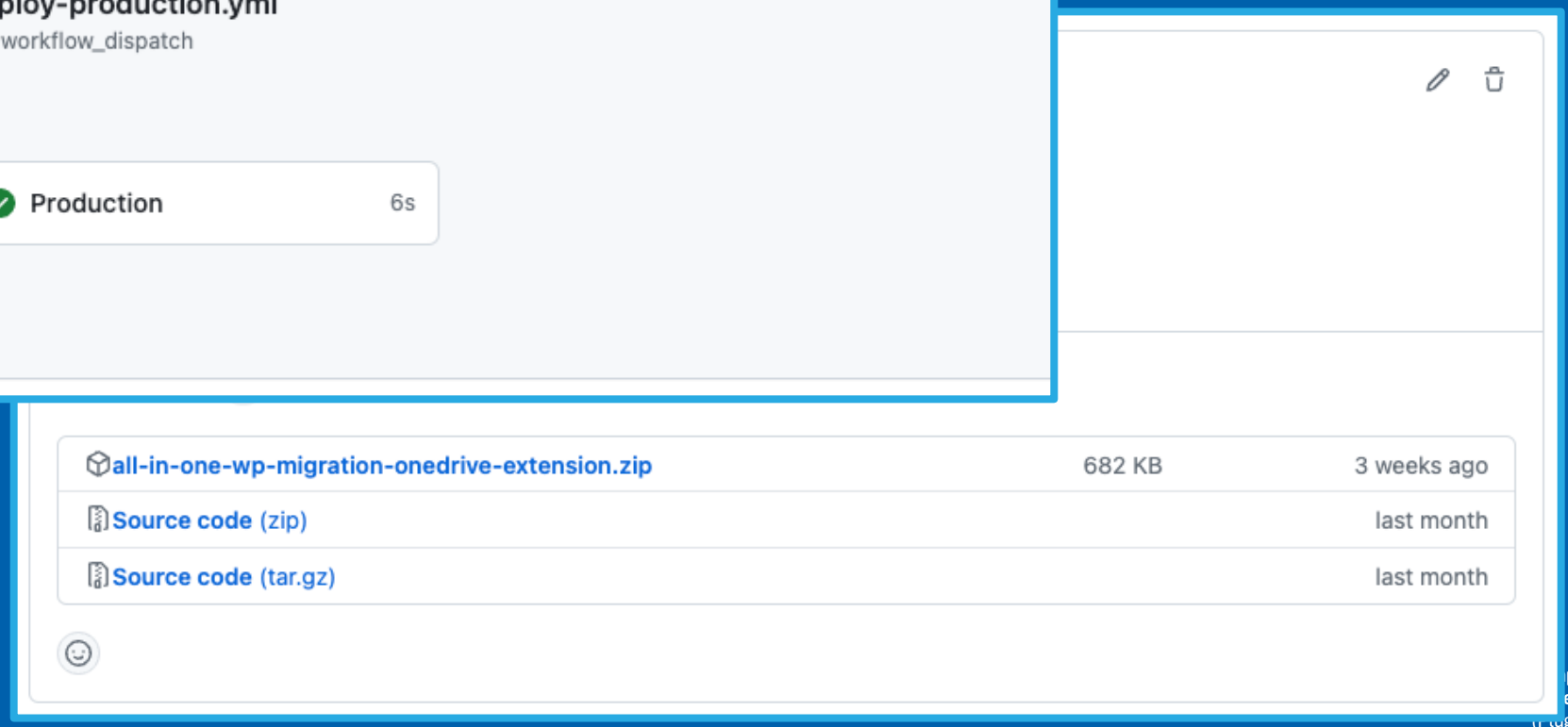


Rapid
WordPress
(Plugin)
Development

```yaml
name: Deploy Production

on:
    workflow_dispatch:
        inputs:
            reason:
                description: What are you releasing?
                required: true

    workflow_call:
        secrets:
            AI1WM_DEPLOY_URL_PRODUCTION:
                required: true
            AI1WM_DEPLOY_KEY_PRODUCTION:
                required: true

jobs:
    deploy:
        runs-on: ubuntu-latest
        name: Production
        env:
            PLUGIN_NAME: 'all-in-one-wp-migration-onedrive-extension'
            PRODUCT_SLUG: 'onedrive-extension'
        steps:
            -   name: Checkout
                uses: actions/checkout@v4
            -   name: Set environment variables
                run: |
                    echo "HOME_DIR=${{ github.workspace }}/dist" >> $GITHUB_ENV
                    echo "VERSION=${{ github.ref_name }}" >> $GITHUB_ENV
                    echo "BRANCH=$(git rev-parse --abbrev-ref HEAD)" >> $GITHUB_ENV
            -   name: Deploy
                uses: ./.github/actions/deploy
                env:
                    TOKEN: ${{ github.token }}
                    REPO: ${{ github.repository }}
                    DEPLOY_URL: ${{ secrets.AI1WM_DEPLOY_URL_PRODUCTION }}
                    DEPLOY_KEY: ${{ secrets.AI1WM_DEPLOY_KEY_PRODUCTION }}
```

Rapid WordPress (Plugin) Development





pid
ess
(Plugin)
Development

Command-Line Tools

Command-Line Tools

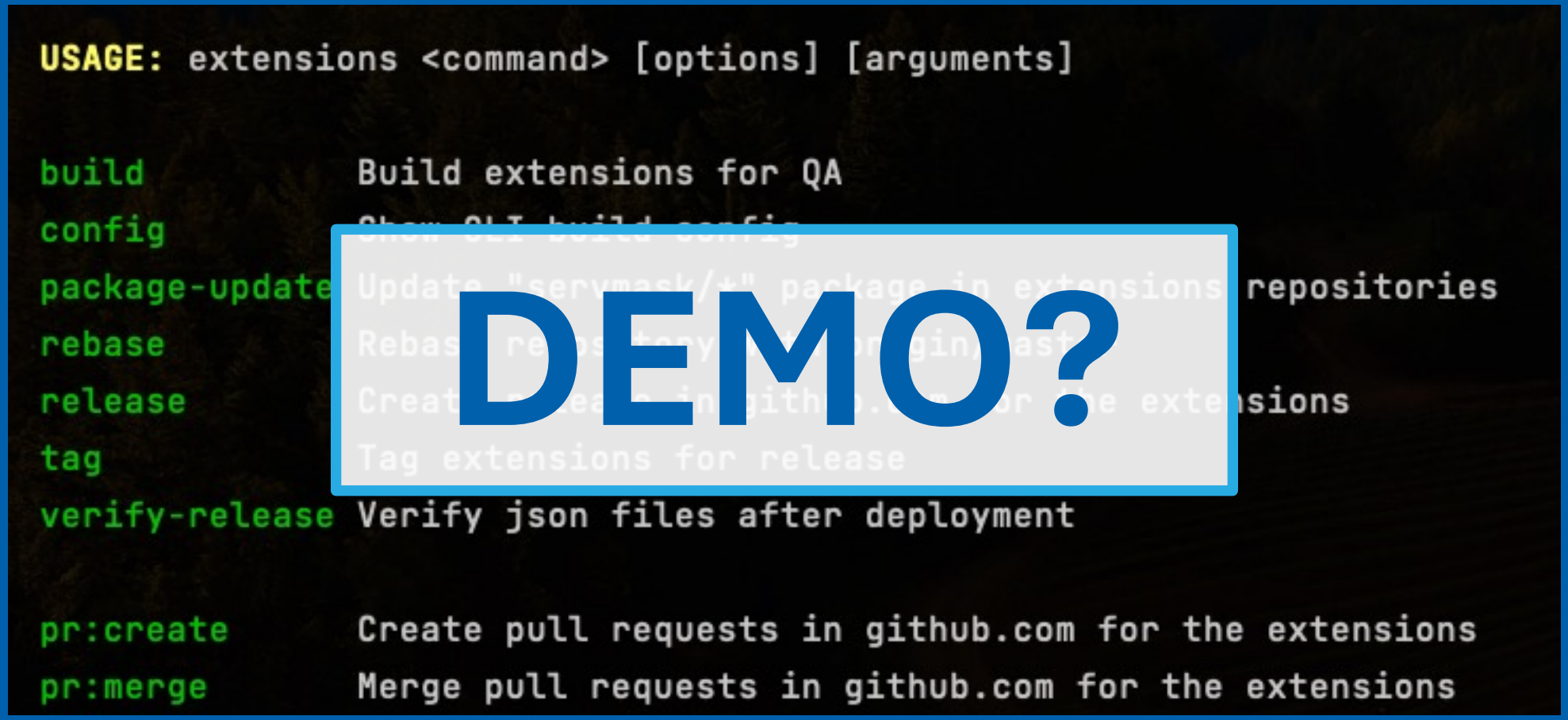
Key Command-Line Tools for WordPress Development:

- **WP-CLI (WP Command-Line Interface)**
- **Git (Version Control System)**
- **Composer (Dependency Manager)**
- **Custom**

Command-Line Tools

Practical Use Cases

- **Plugin Installation and Management**
- **Version Control and Collaboration**
- **Dependency Management**
- **Automation and Scripting**



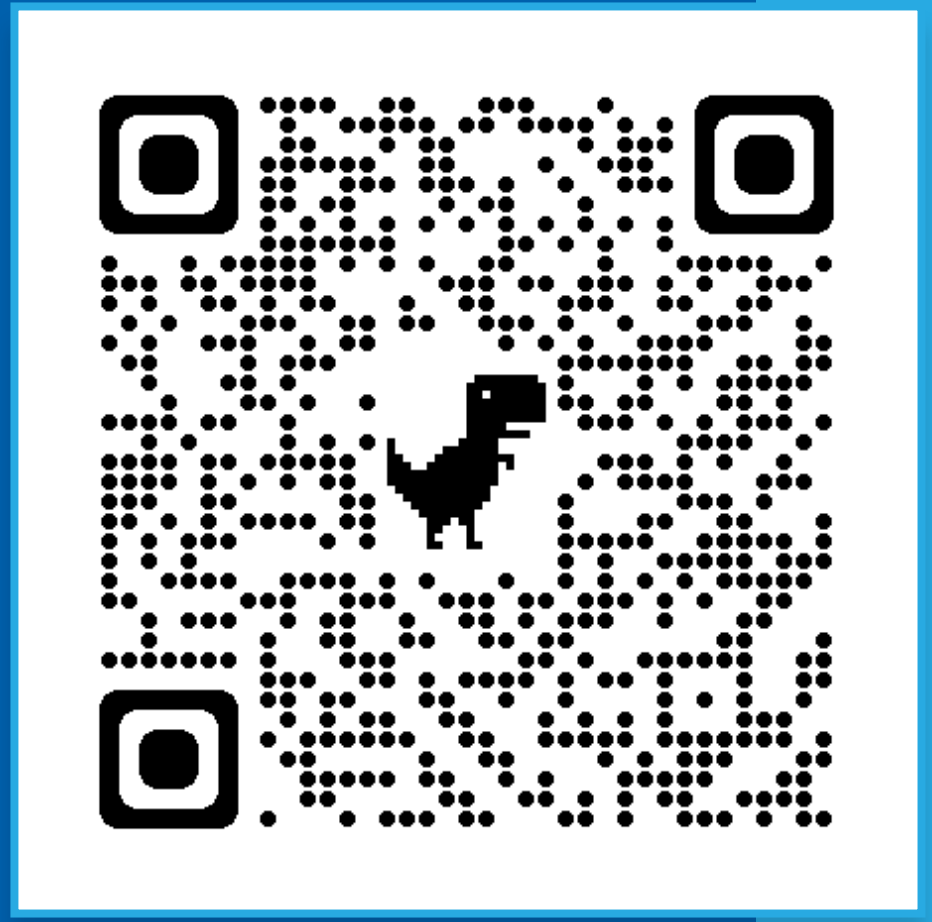
Rapid
WordPress
(Plugin)
Development

Rapid WordPress (plugin) Development

- Achievable through the integration of **Composer**, **GitHub Actions**, and (custom) **command-line tools**.
- These tools streamline workflows, automate processes, and enhance collaboration, leading to **faster development** cycles and **higher-quality** code.
- Using the **Composer** for dependency management, **GitHub Actions** for automation, and **command-line tools** for efficiency, you can accelerate your WordPress development journey.
- Embrace these tools, unleash their power, and embark on a journey to elevate your WordPress development experience to new heights of **speed**, **efficiency**, and **innovation**.

QA

Thank You!



@AlbertSuntic

Rapid
WordPress
(Plugin)
Development