Test automation and pipelines one step at a time

Antonis Pavlakis
@pavlakis

Code

Unit Tests
(local environment)

Possible steps required

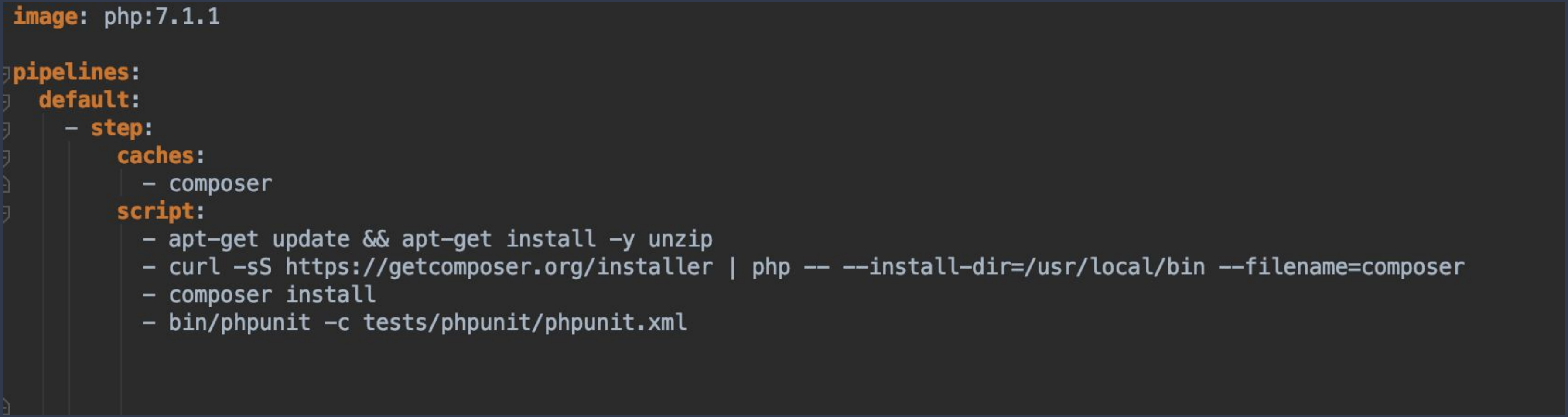
- composer install
- bin/phpunit -c tests/phpunit/phpunit.xml

Unit Tests
(CI environment)

Possible steps required

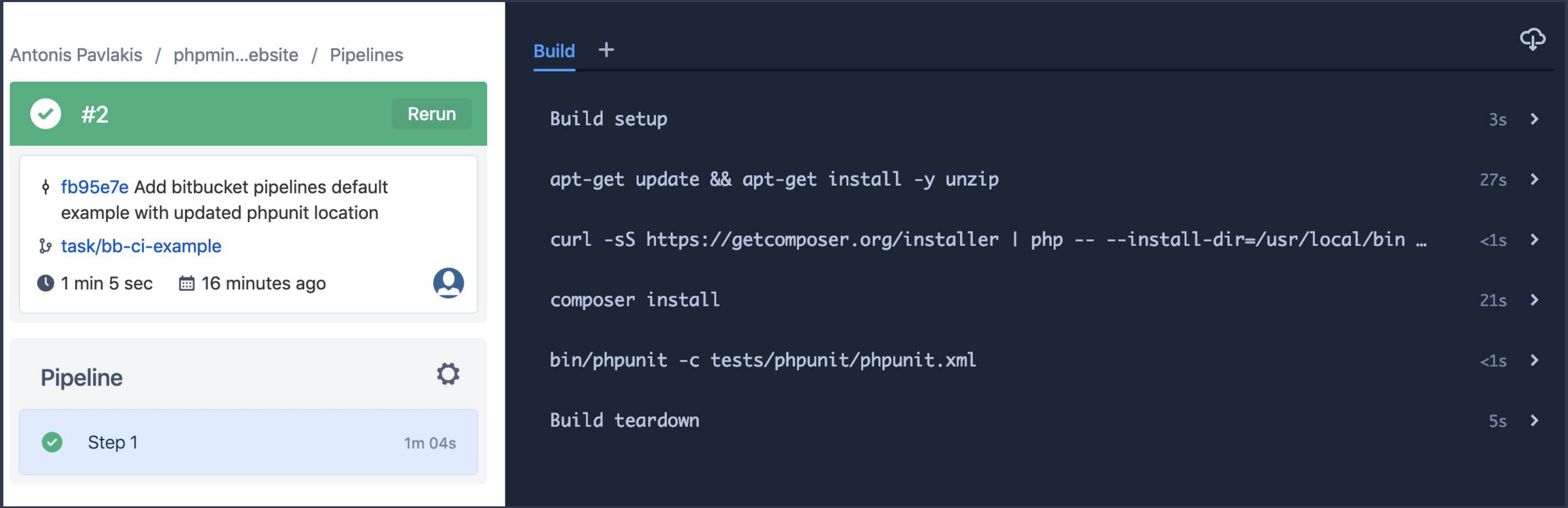
- Download composer
- composer install
- bin/phpunit -c tests/phpunit/phpunit.xml

Bitbucket Pipelines



```yaml
image: php:7.1.1

pipelines:
  default:
    - step:
        caches:
          - composer
        script:
          - apt-get update && apt-get install -y unzip
          - curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer
          - composer install
          - bin/phpunit -c tests/phpunit/phpunit.xml
```

Pipeline status



Antonis Pavlakis / phpmin...ebsite / Pipelines

✓ #2 Rerun

fb95e7e Add bitbucket pipelines default example with updated phpunit location

task/bb-ci-example

1 min 5 sec 16 minutes ago

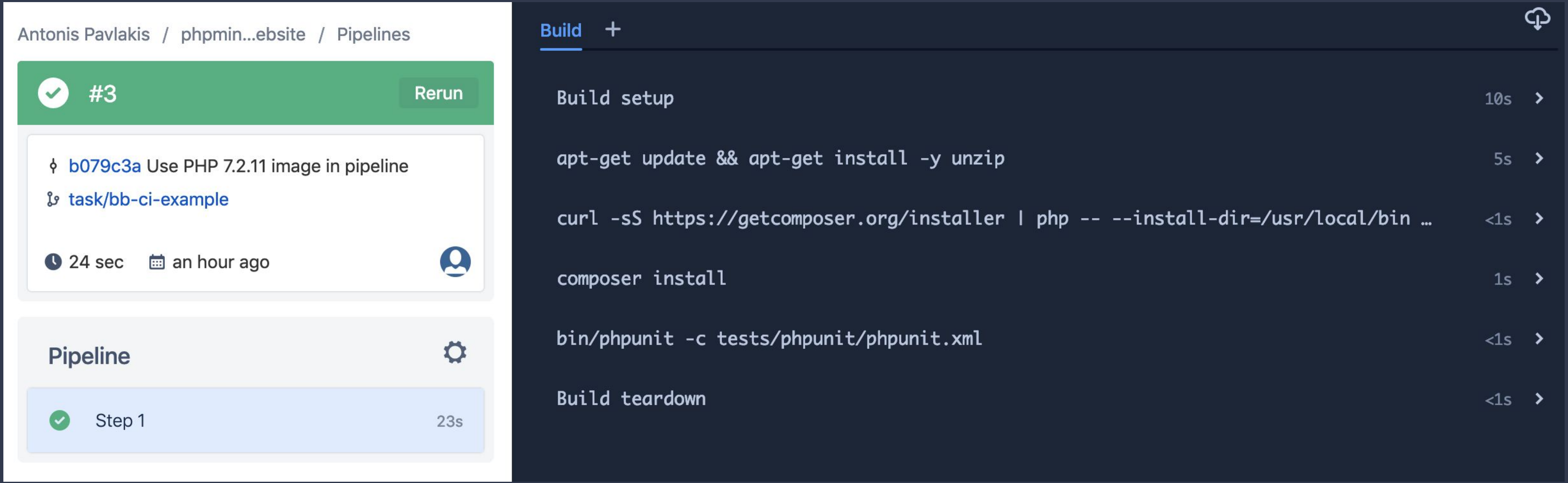
Pipeline

✓ Step 1 1m 04s

Build +

```
Build setup                                                                          3s
apt-get update && apt-get install -y unzip                                          27s
curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin …  <1s
composer install                                                                    21s
bin/phpunit -c tests/phpunit/phpunit.xml                                            <1s
Build teardown                                                                       5s
```

Cache performance



Antonis Pavlakis / phpmin...ebsite / Pipelines

b079c3a Use PHP 7.2.11 image in pipeline

task/bb-ci-example

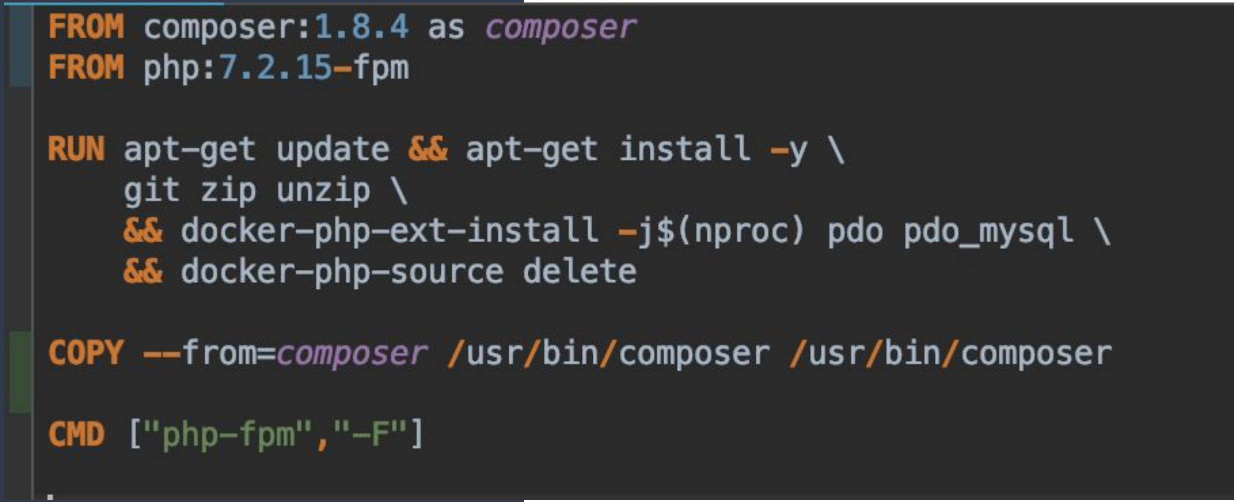
24 sec   an hour ago

Step 1   23s

```
Build setup                                                                    10s
apt-get update && apt-get install -y unzip                                     5s
curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin …   <1s
composer install                                                               1s
bin/phpunit -c tests/phpunit/phpunit.xml                                       <1s
Build teardown                                                                 <1s
```

Using a custom Docker image

Dockerfile



```
FROM composer:1.8.4 as composer
FROM php:7.2.15-fpm

RUN apt-get update && apt-get install -y \
    git zip unzip \
    && docker-php-ext-install -j$(nproc) pdo pdo_mysql \
    && docker-php-source delete

COPY --from=composer /usr/bin/composer /usr/bin/composer

CMD ["php-fpm","-F"]
```

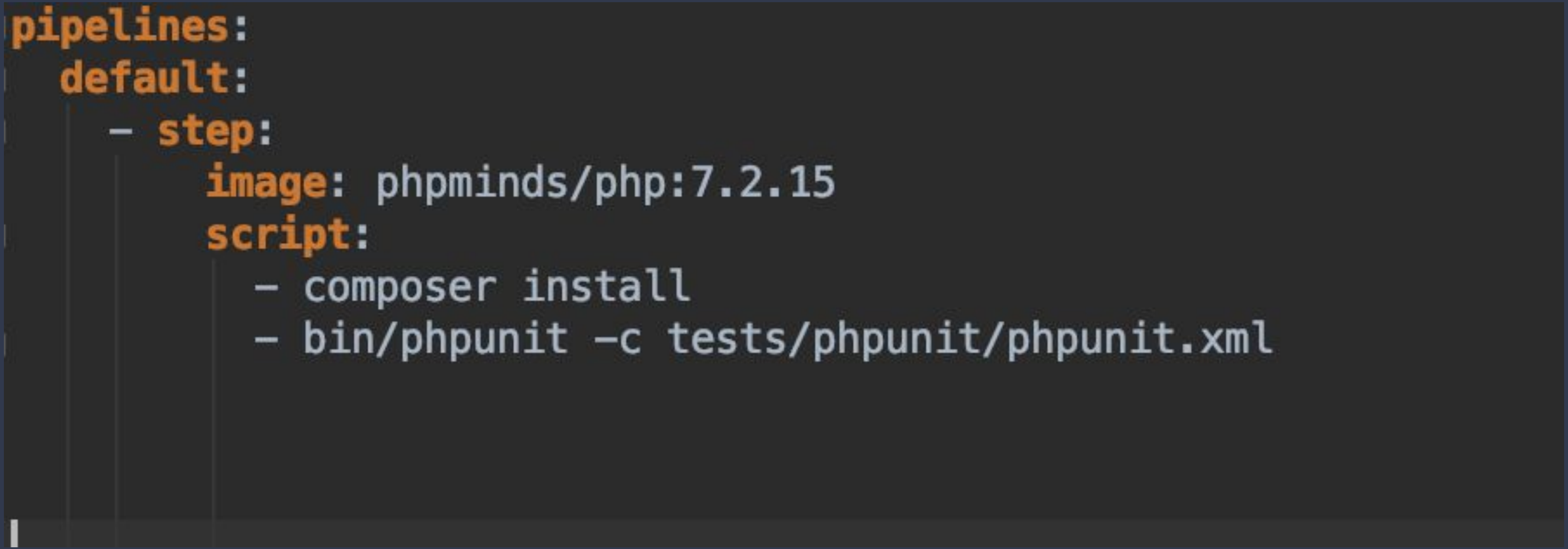
Build Docker image

```
docker build -t
phpminds/php:7.2.15 .
```

→

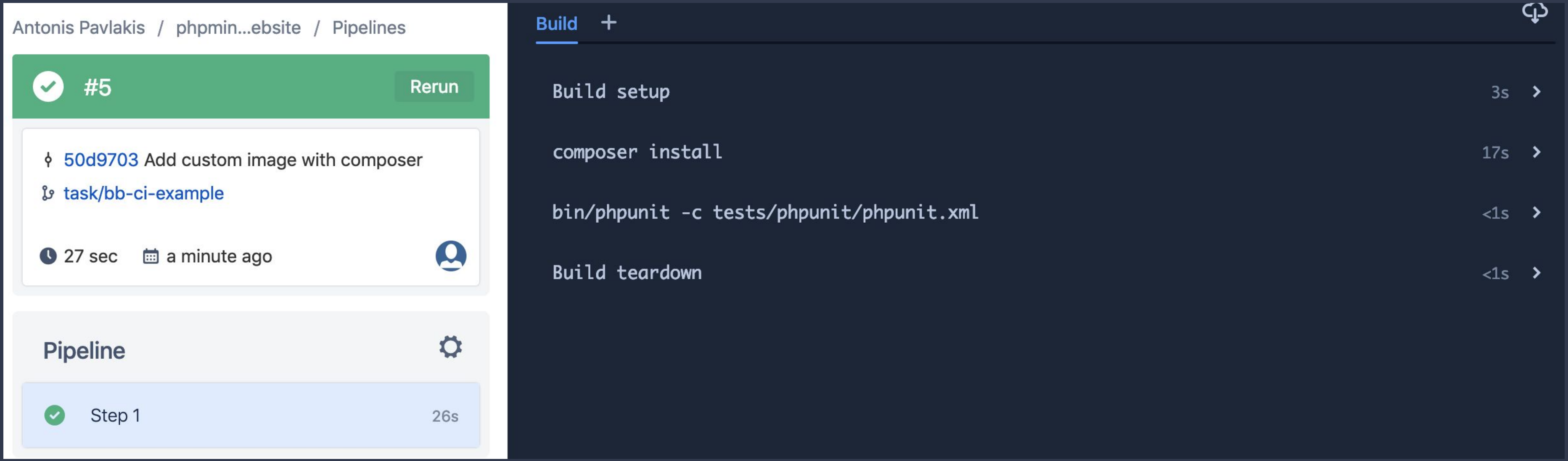
```
docker push
phpminds/php:7.2.15
```

Pipeline with custom image



```
pipelines:
  default:
    - step:
        image: phpminds/php:7.2.15
        script:
          - composer install
          - bin/phpunit -c tests/phpunit/phpunit.xml
```

Pipeline performance



Antonis Pavlakis / phpmin...ebsite / Pipelines

✓ #5 Rerun

50d9703 Add custom image with composer

task/bb-ci-example

27 sec a minute ago

Pipeline

✓ Step 1 26s

Build +

```
Build setup                                  3s
composer install                            17s
bin/phpunit -c tests/phpunit/phpunit.xml    <1s
Build teardown                              <1s
```

Integration Tests (local environment)

Possible steps required

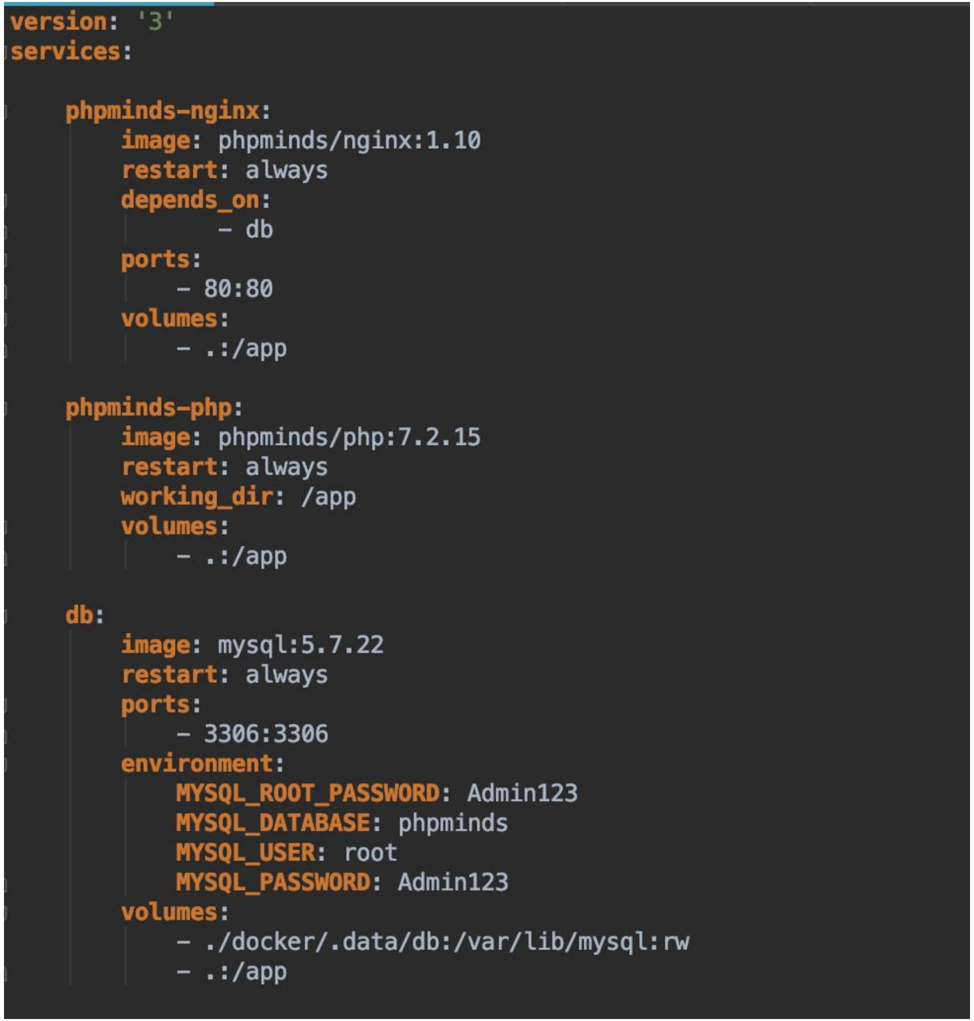
- composer install
- composer reset-db
- composer populate-db
- bin/phpunit -c tests/phpunit/phpunit.xml --group integration

Set DB to a known state

Using composer scripts

```
"scripts": {
 "reset-db": [
        "bin/console doctrine:database:drop --force",
        "bin/console doctrine:database:create",
        "bin/console doctrine:schema:update --force"
 ],
 "populate-db": [
        "bin/console doctrine:fixtures:load"
 ]
}
```

Docker Compose

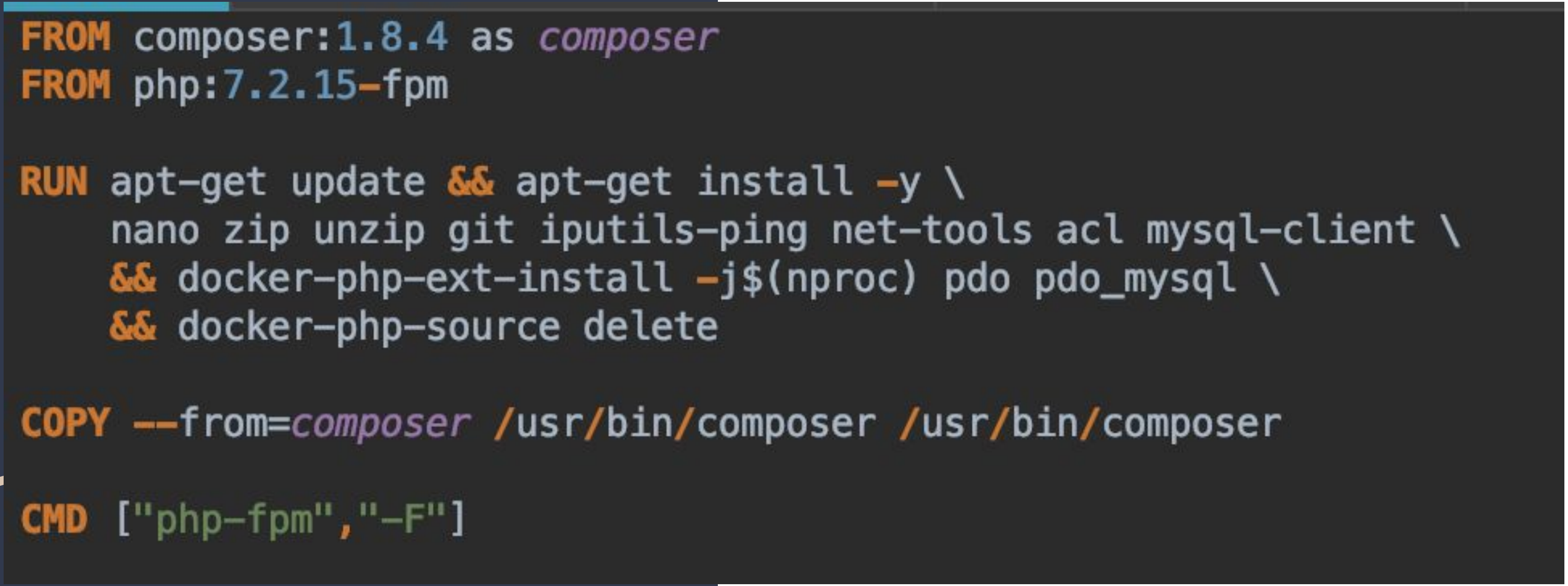


```yaml
version: '3'
services:

    phpminds-nginx:
        image: phpminds/nginx:1.10
        restart: always
        depends_on:
               - db
        ports:
            - 80:80
        volumes:
            - .:/app

    phpminds-php:
        image: phpminds/php:7.2.15
        restart: always
        working_dir: /app
        volumes:
            - .:/app

    db:
        image: mysql:5.7.22
        restart: always
        ports:
            - 3306:3306
        environment:
            MYSQL_ROOT_PASSWORD: Admin123
            MYSQL_DATABASE: phpminds
            MYSQL_USER: root
            MYSQL_PASSWORD: Admin123
        volumes:
            - ./docker/.data/db:/var/lib/mysql:rw
            - .:/app
```

Update Docker image

Update Docker image

- Add mysql client



```
FROM composer:1.8.4 as composer
FROM php:7.2.15-fpm

RUN apt-get update && apt-get install -y \
    nano zip unzip git iputils-ping net-tools acl mysql-client \
    && docker-php-ext-install -j$(nproc) pdo pdo_mysql \
    && docker-php-source delete

COPY --from=composer /usr/bin/composer /usr/bin/composer

CMD ["php-fpm","-F"]
```

Run integration tests using Docker Compose

If docker compose is not already running:

```
docker-compose up --build -d
```

Run tests from inside the running container:

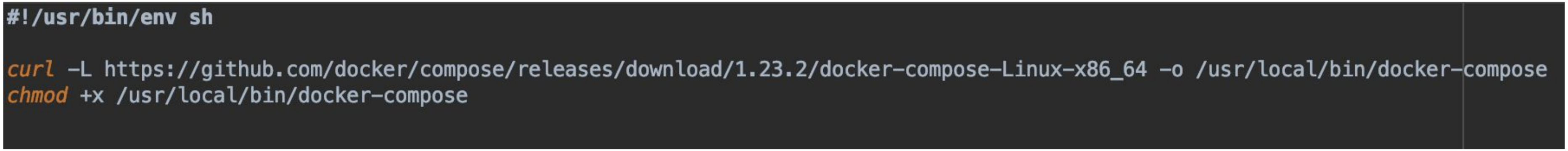
```
docker-compose exec -T phpminds-php sh -c "bin/phpunit -c tests/phpunit/phpunit.xml --group integration"
```

Docker Compose in Bitbucket Pipelines

Steps required

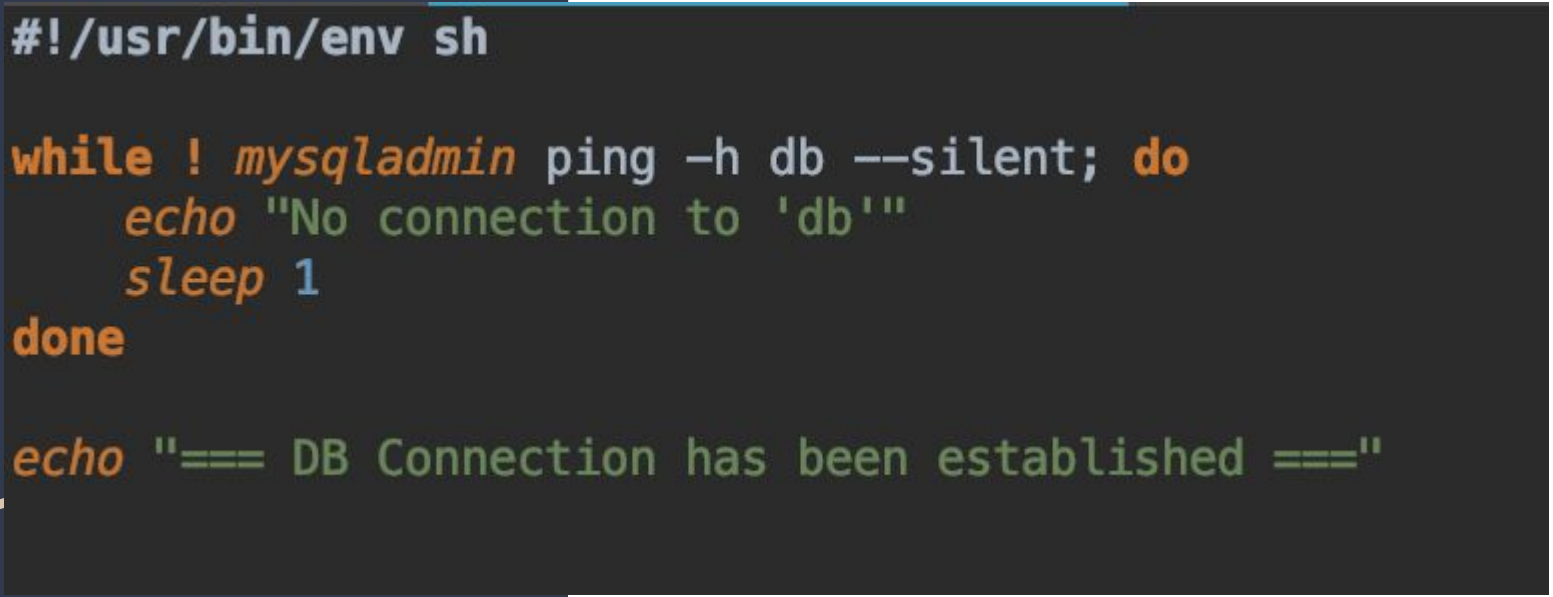
- Install Docker Compose
- docker-compose up --build -d
- composer install
- (wait for DB container to start)
- composer reset-db
- composer populate-db
- bin/phpunit -c tests/phpunit/phpunit.xml --group integration

Install Docker Compose



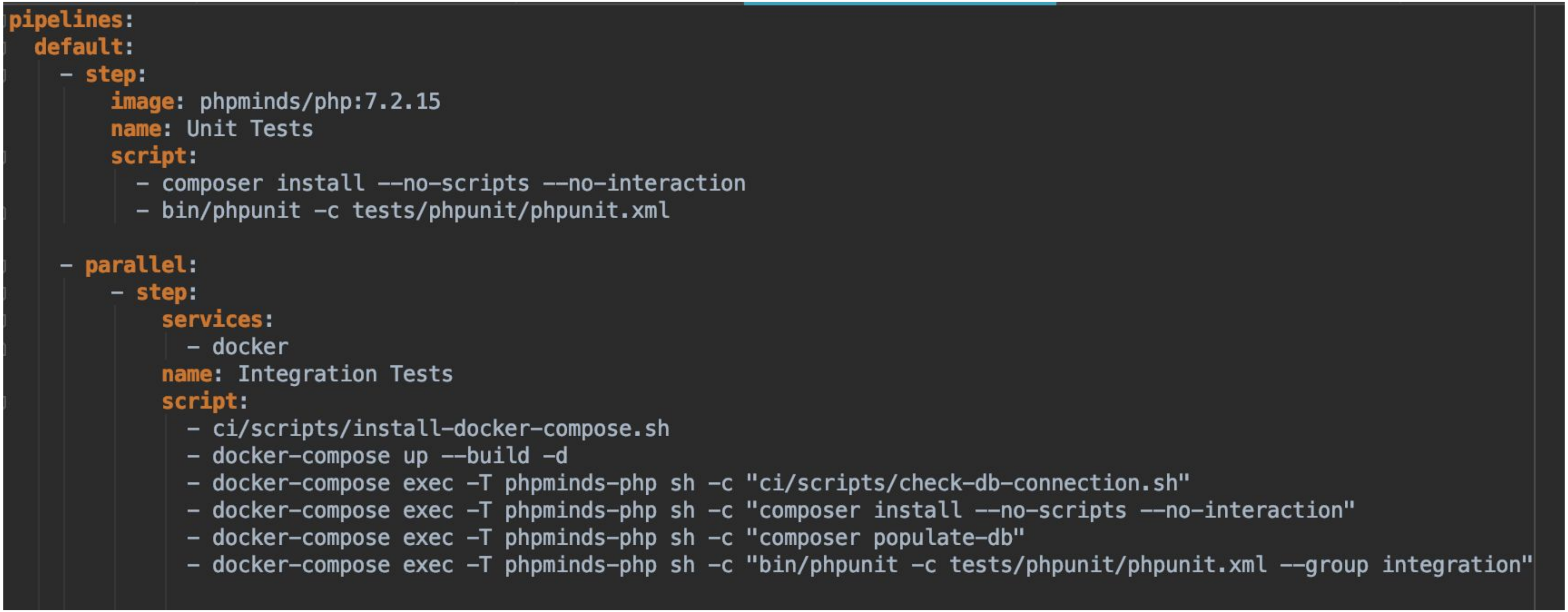
```
#!/usr/bin/env sh

curl -L https://github.com/docker/compose/releases/download/1.23.2/docker-compose-Linux-x86_64 -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

Check DB connection



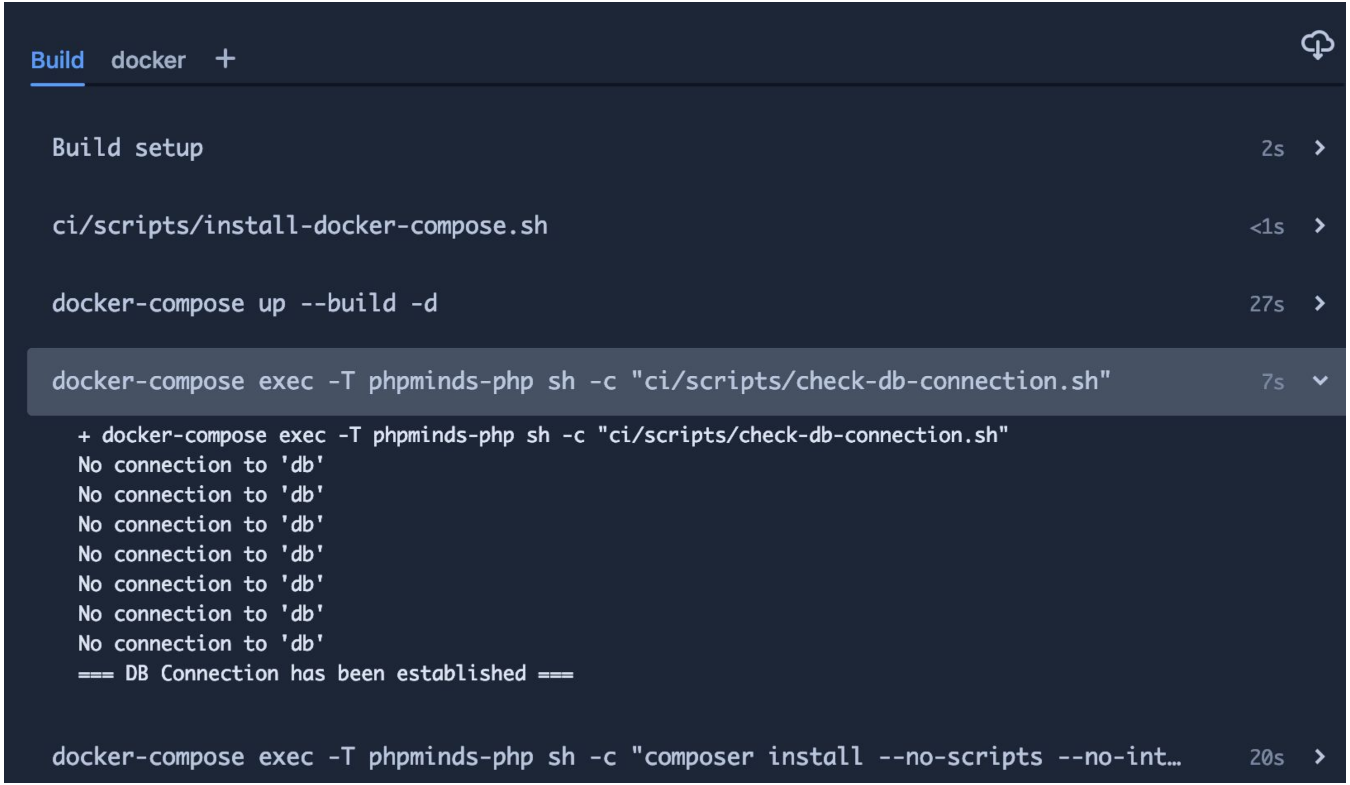
```
#!/usr/bin/env sh

while ! mysqladmin ping -h db --silent; do
    echo "No connection to 'db'"
    sleep 1
done

echo "=== DB Connection has been established ==="
```

Putting it all together



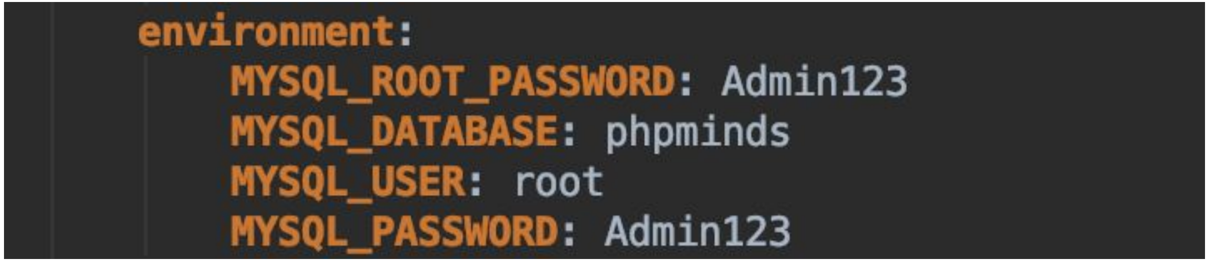
```yaml
pipelines:
  default:
    - step:
        image: phpminds/php:7.2.15
        name: Unit Tests
        script:
          - composer install --no-scripts --no-interaction
          - bin/phpunit -c tests/phpunit/phpunit.xml

    - parallel:
        - step:
            services:
              - docker
            name: Integration Tests
            script:
              - ci/scripts/install-docker-compose.sh
              - docker-compose up --build -d
              - docker-compose exec -T phpminds-php sh -c "ci/scripts/check-db-connection.sh"
              - docker-compose exec -T phpminds-php sh -c "composer install --no-scripts --no-interaction"
              - docker-compose exec -T phpminds-php sh -c "composer populate-db"
              - docker-compose exec -T phpminds-php sh -c "bin/phpunit -c tests/phpunit/phpunit.xml --group integration"
```

A build with Docker Compose

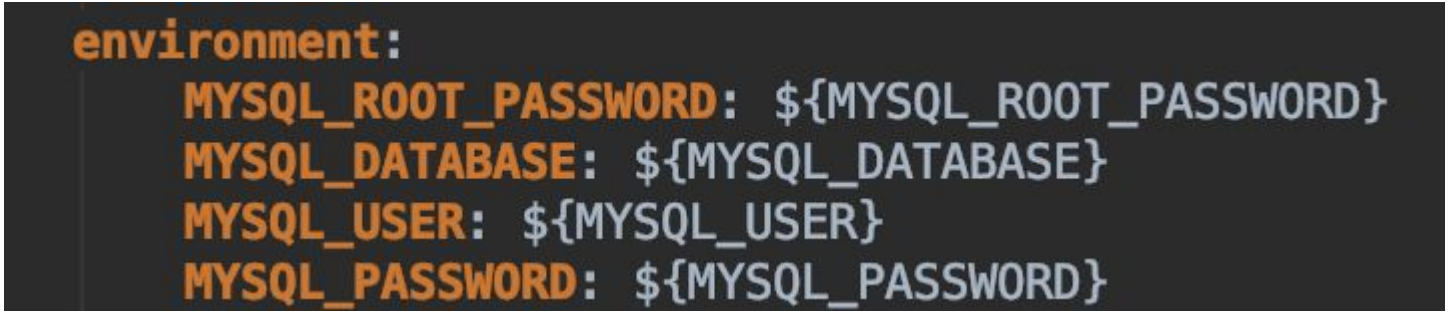


```
Build setup                                                              2s  >
ci/scripts/install-docker-compose.sh                                    <1s  >
docker-compose up --build -d                                            27s  >
docker-compose exec -T phpminds-php sh -c "ci/scripts/check-db-connection.sh"   7s  v
  + docker-compose exec -T phpminds-php sh -c "ci/scripts/check-db-connection.sh"
  No connection to 'db'
  No connection to 'db'
  No connection to 'db'
  No connection to 'db'
  No connection to 'db'
  No connection to 'db'
  No connection to 'db'
  === DB Connection has been established ===
docker-compose exec -T phpminds-php sh -c "composer install --no-scripts --no-int…   20s  >
```

Environment Variables

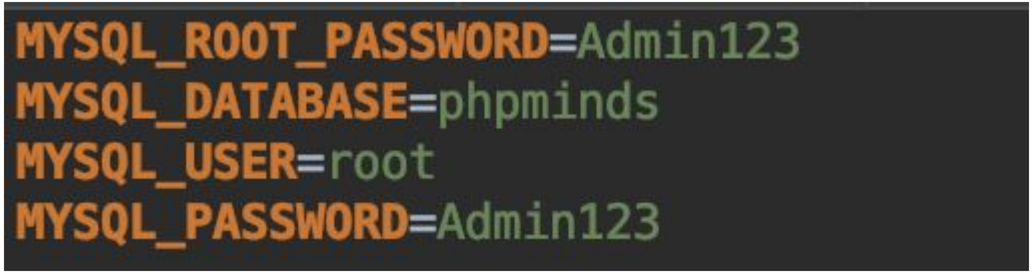


```
environment:
    MYSQL_ROOT_PASSWORD: Admin123
    MYSQL_DATABASE: phpminds
    MYSQL_USER: root
    MYSQL_PASSWORD: Admin123
```



```
environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
```

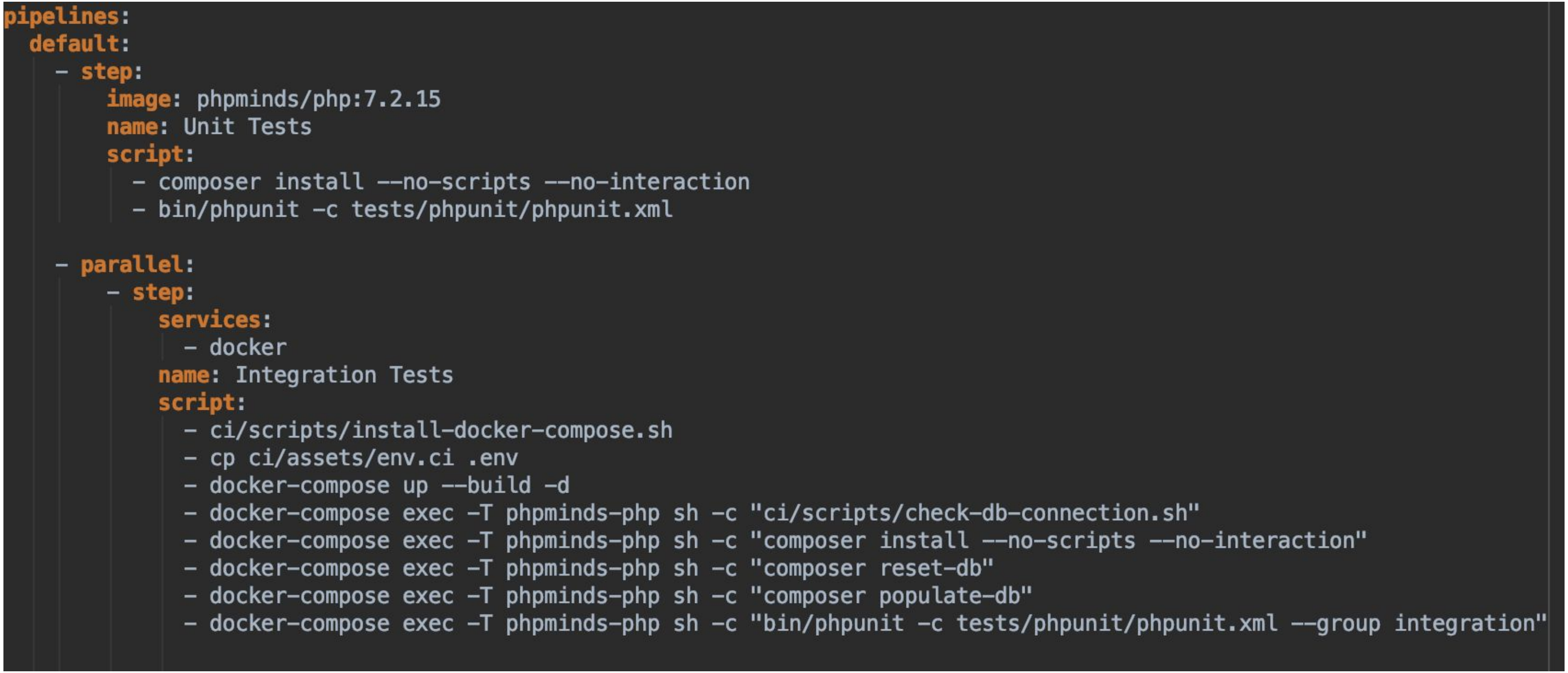
Using .env



```
MYSQL_ROOT_PASSWORD=Admin123
MYSQL_DATABASE=phpminds
MYSQL_USER=root
MYSQL_PASSWORD=Admin123
```

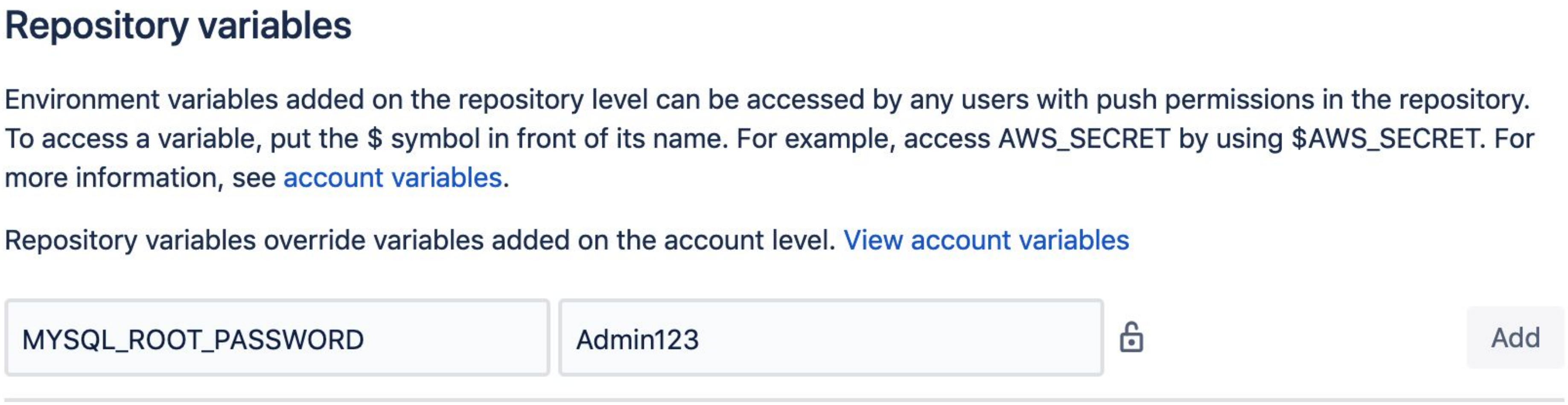
ci/assets/env.ci

Using .env



```yaml
pipelines:
  default:
    - step:
        image: phpminds/php:7.2.15
        name: Unit Tests
        script:
          - composer install --no-scripts --no-interaction
          - bin/phpunit -c tests/phpunit/phpunit.xml

    - parallel:
      - step:
          services:
            - docker
          name: Integration Tests
          script:
            - ci/scripts/install-docker-compose.sh
            - cp ci/assets/env.ci .env
            - docker-compose up --build -d
            - docker-compose exec -T phpminds-php sh -c "ci/scripts/check-db-connection.sh"
            - docker-compose exec -T phpminds-php sh -c "composer install --no-scripts --no-interaction"
            - docker-compose exec -T phpminds-php sh -c "composer reset-db"
            - docker-compose exec -T phpminds-php sh -c "composer populate-db"
            - docker-compose exec -T phpminds-php sh -c "bin/phpunit -c tests/phpunit/phpunit.xml --group integration"
```

Managing secrets



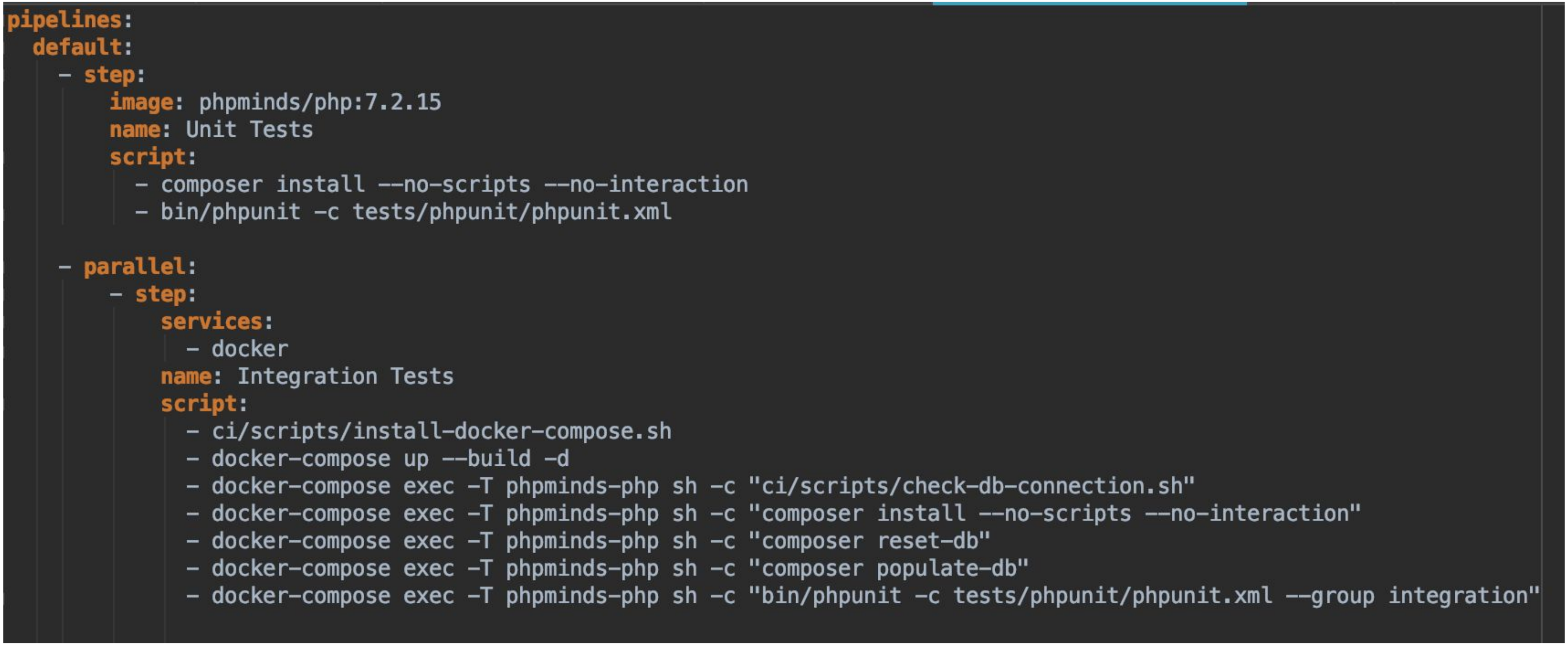
Repository variables

Environment variables added on the repository level can be accessed by any users with push permissions in the repository. To access a variable, put the $ symbol in front of its name. For example, access AWS_SECRET by using $AWS_SECRET. For more information, see account variables.

Repository variables override variables added on the account level. View account variables

MYSQL_ROOT_PASSWORD	Admin123	🔓

Using repository variables



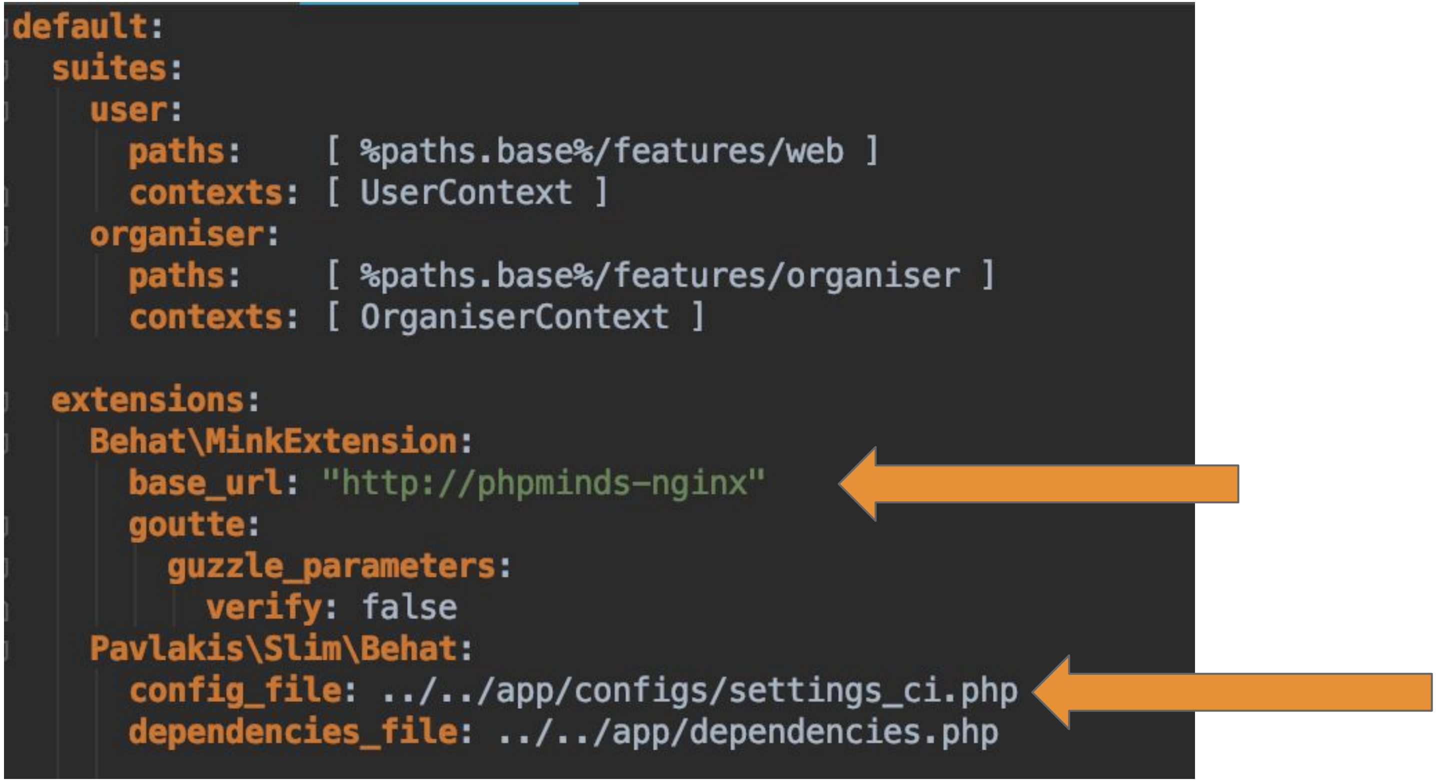
```yaml
pipelines:
  default:
    - step:
        image: phpminds/php:7.2.15
        name: Unit Tests
        script:
          - composer install --no-scripts --no-interaction
          - bin/phpunit -c tests/phpunit/phpunit.xml

    - parallel:
        - step:
            services:
              - docker
            name: Integration Tests
            script:
              - ci/scripts/install-docker-compose.sh
              - docker-compose up --build -d
              - docker-compose exec -T phpminds-php sh -c "ci/scripts/check-db-connection.sh"
              - docker-compose exec -T phpminds-php sh -c "composer install --no-scripts --no-interaction"
              - docker-compose exec -T phpminds-php sh -c "composer reset-db"
              - docker-compose exec -T phpminds-php sh -c "composer populate-db"
              - docker-compose exec -T phpminds-php sh -c "bin/phpunit -c tests/phpunit/phpunit.xml --group integration"
```

Acceptance Tests

Possible steps required

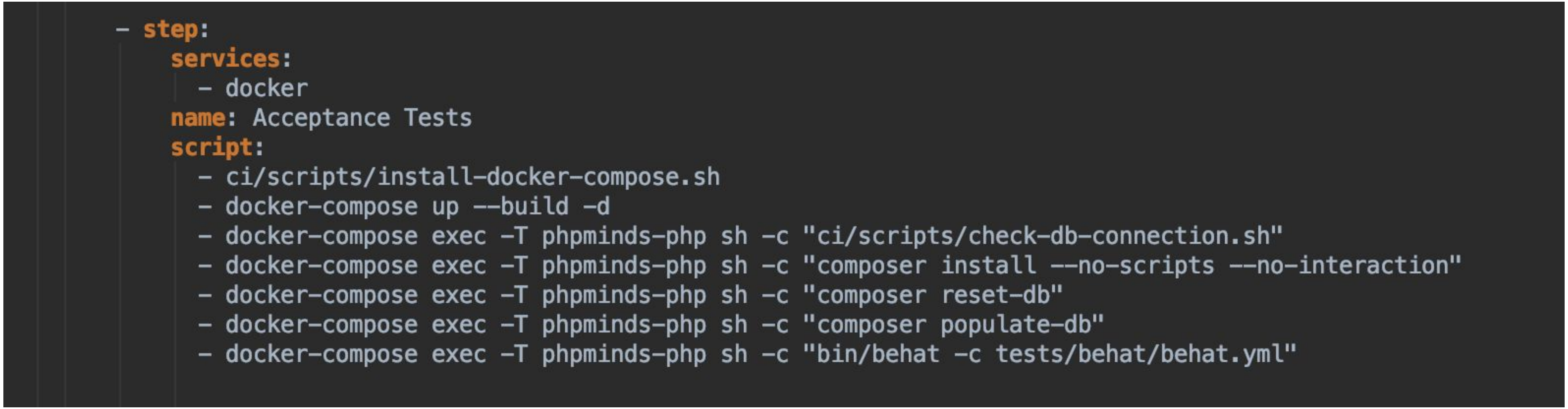
- composer install
- composer reset-db
- composer populate-db
- bin/behat -c tests/behat/behat.yml

Example behat.yml



```
default:
  suites:
    user:
      paths:    [ %paths.base%/features/web ]
      contexts: [ UserContext ]
    organiser:
      paths:    [ %paths.base%/features/organiser ]
      contexts: [ OrganiserContext ]

  extensions:
    Behat\MinkExtension:
      base_url: "http://phpminds-nginx"
      goutte:
        guzzle_parameters:
          verify: false
    Pavlakis\Slim\Behat:
      config_file: ../../app/configs/settings_ci.php
      dependencies_file: ../../app/dependencies.php
```

Add pipeline step



```
- step:
    services:
      - docker
    name: Acceptance Tests
    script:
      - ci/scripts/install-docker-compose.sh
      - docker-compose up --build -d
      - docker-compose exec -T phpminds-php sh -c "ci/scripts/check-db-connection.sh"
      - docker-compose exec -T phpminds-php sh -c "composer install --no-scripts --no-interaction"
      - docker-compose exec -T phpminds-php sh -c "composer reset-db"
      - docker-compose exec -T phpminds-php sh -c "composer populate-db"
      - docker-compose exec -T phpminds-php sh -c "bin/behat -c tests/behat/behat.yml"
```

Thank You

Antonis Pavlakis

@pavlakis

php[MiNDS]