Środowiska deweloperskie, testowe i produkcyjne

Projektowanie i programowanie systemów internetowych I

mgr inż. Krzysztof Rewak

25 marca 2018

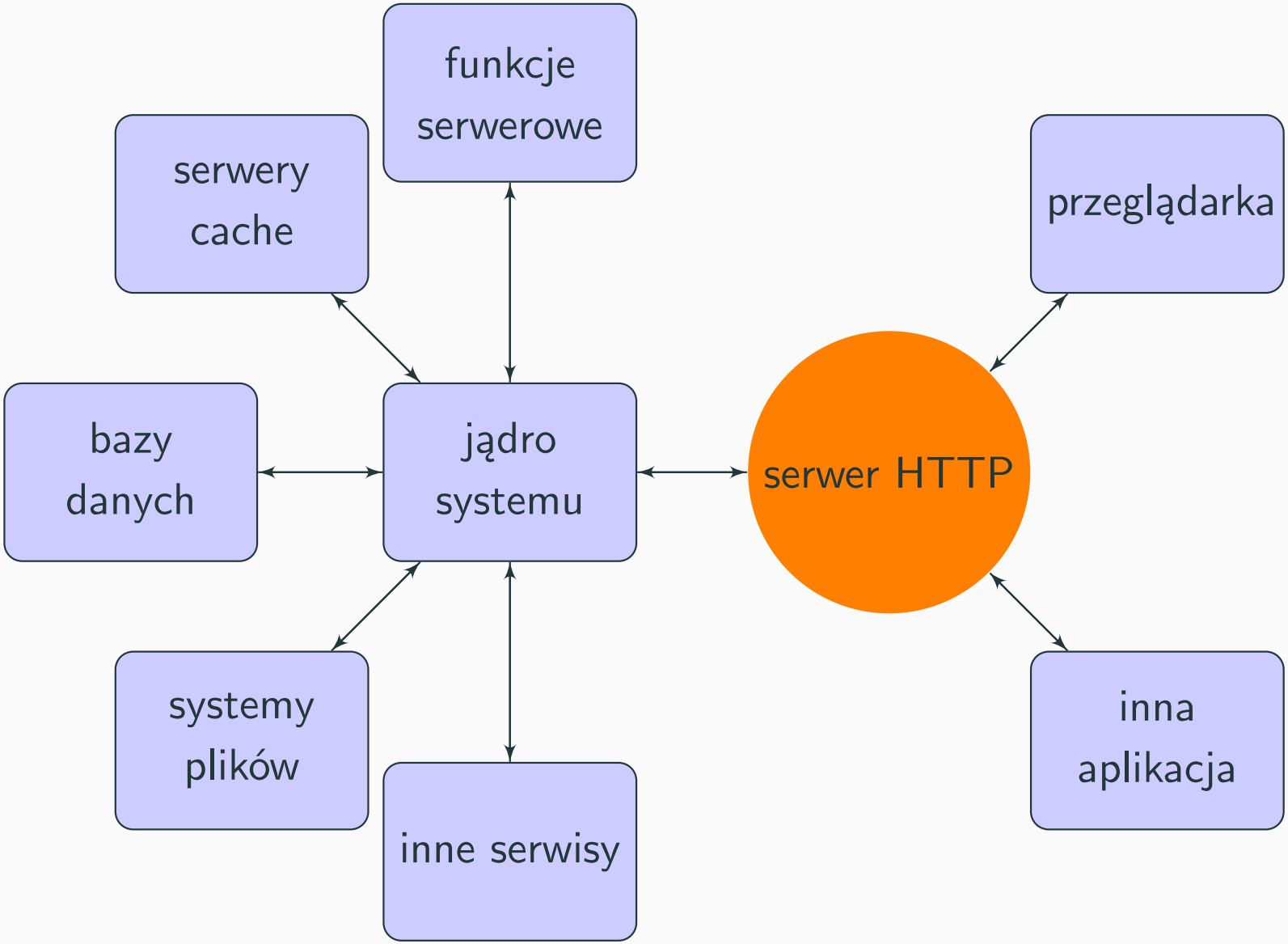
Wydział Nauk Technicznych i Ekonomicznych
Państwowa Wyższa Szkoła Zawodowa im. Witelona w Legnicy

Plan prezentacji

1. Środowisko programistyczne
2. Środowisko lokalne
3. Środowisko testowe
4. Środowisko stagingowe
5. Środowisko produkcyjne
6. Podsumowanie

Środowisko programistyczne

Budowa klasycznego systemu internetowego



Dramatis personae

Na co między innymi powinniśmy zwrócić uwagę?

- rodzaj, wersja i konfiguracja serwera
- wersja języka lub interpretera
- zainstalowane programy
- wersja i typ bazy danych
- wersja frameworka
- zakres funkcjonalności

(uogólniona) Definicja

Środowiskiem programistycznym możemy uogólniając nazwać zestaw konkretnych programów, ich wersji oraz ich konfiguracji.

Pierwsze problemy

Pierwszy problem pojawi się w momencie zaistnienia różnic między środowiskiem jakie mamy na komputerze przy którym pracujemy a środowiskiem na maszynie, na której system będzie działał docelowo.

Więcej problemów?

U mnie działa już dawno temu przestało być wiarygodną wymówką.

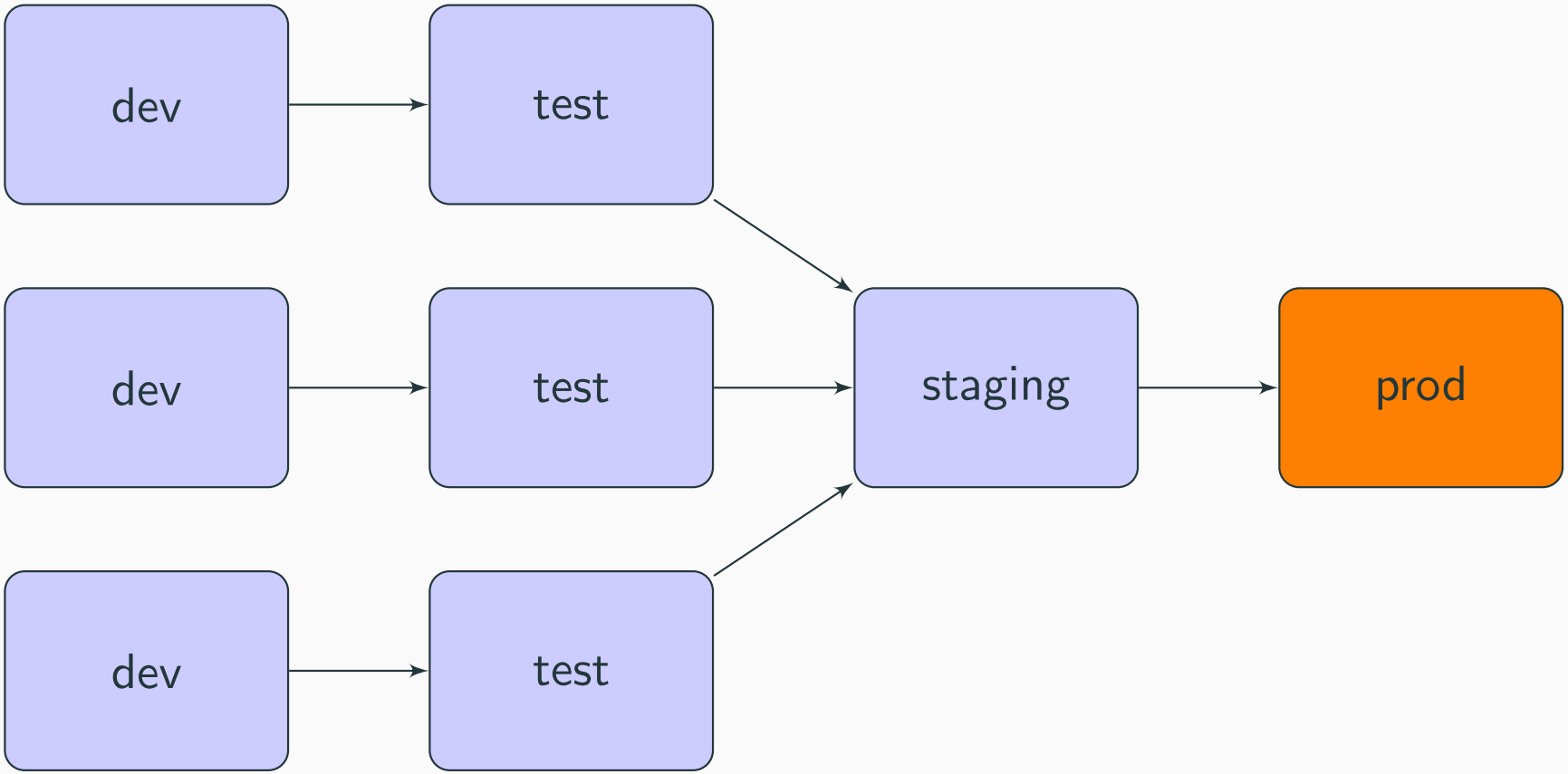
Profesjonalista powinien przedstawić klientowi swoją aplikację. Ale jak to sensownie zrobić, jeżeli klient mieszka hen daleko w Honolulu, więc nie możemy odwiedzić go z naszym laptopem?

Rozwiązanie jest proste

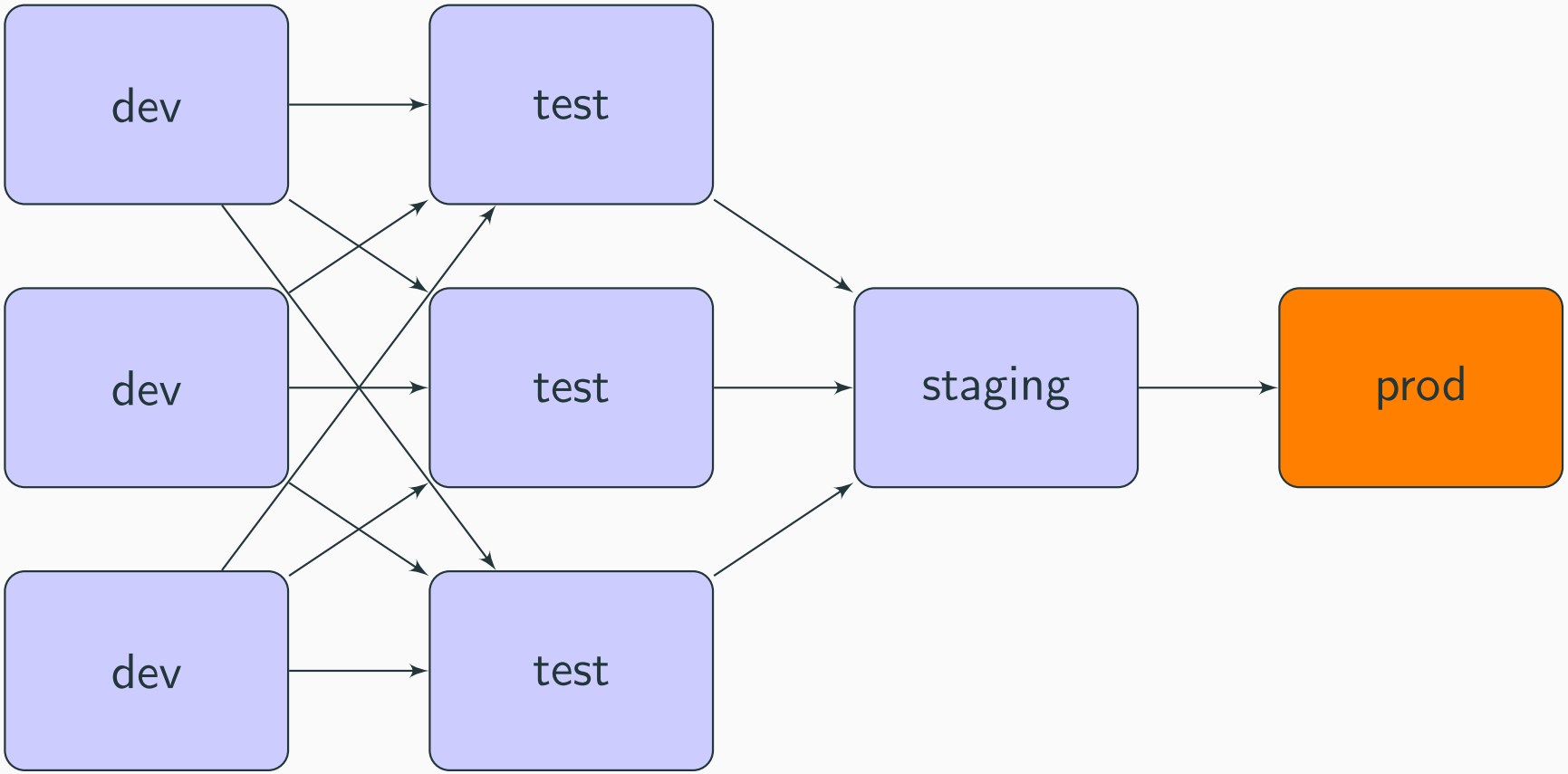
Aby usprawić proces wytwarzania oprogramowania, przyjęto schemat pracy z wieloma środowiskami programistycznymi, które są ze sobą powiązane w konkretny sposób. Należy wyróżnić środowiska:

- lokalne
- deweloperskie
- testowe
- *stagingowe*
- produkcyjne

Co, gdzie, kiedy, jak?



Co, gdzie, kiedy, jak?



Kanon środowisk?

Powiązania między środowiskami oczywiście są dowolne, a przede wszystkim powinny zależeć od wymagań klienta.

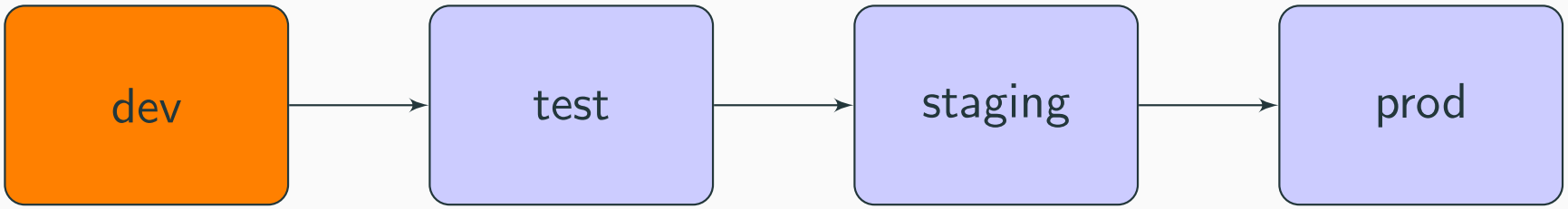
Łatwo sobie wyobrazić dodanie dodatkowych węzłów w poprzednio przedstawionych slajdach. Wszystko powinno być jednakże opracowywane z umiarem.

Środowisko lokalne

Środowisko lokalne

Środowisko lokalne to środowisko stacji roboczej przy której pracuje programista.

ENV=dev



Środowisko lokalne

Korzystając z wirtualizacji lub konteneryzacji można zbliżyć do tożsamości środowiska deweloperskie i produkcyjne. Jest to sytuacja idealna, ostatnio coraz częściej promowana, jednak niestety wciąż nieczęsto spotykana na rynku pracy.

Prawda jest taka, że dbając o *higienę* programistyczną można spokojnie programować w ulubionych lub dostępnym systemie operacyjnym i środowisku.

Czym powinna być taka *higiena*? Przede wszystkim znajomością różnic między systemami i umiejętnością radzenia sobie z problemami, które mogą z tego wyniknąć.

Środowisko lokalne

Czym się **może** różnić środowisko lokalne od pozostałych?

- zazwyczaj jest tzw. sandboksem,
- posiada bardziej rozbudowane system logów,
- ma uruchomione narzędzia deweloperskie: debugbar, profiler, itd.
- ma wyłączone lub przekierowane funkcje analizy ruchu,
- na nim uruchamiane jest IDE.

Przykład

```
use Phalcon\Mvc\Application;
use Whoops\Provider\Phalcon\WhoopsServiceProvider as Whoops;

error_reporting(E_ALL);

define("APP_PATH", realpath(".."));

require_once APP_PATH . "/vendor/autoload.php";

$config = include APP_PATH . "/app/config.php";
$di = include APP_PATH . "/app/services.php";

$application = new Application($di);

if($config->application->environment == "dev") {
    new Whoops($di);
}

echo $application->handle()->getContent();
```

Przykład

```
{
  "name": "pwsz-teacher-homepage",
  "version": "0.0.2",
  "description": "Phalcon/Vue.js app",
  "author": "Krzysztof Rewak",
  "private": true,
  "scripts": {
    "dev": "node frontend/build/dev-server.js",
    "start": "node frontend/build/dev-server.js",
    "build": "node frontend/build/build.js"
  },
  "dependencies": {
    "semantic-ui-css": "^2.2.12",
    "sizzle": "^2.3.3",
    "vue": "^2.3.3",
    "vue-analytics": "^5.1.1",
    "vue-cookie": "^1.1.4",
    "vue-resource": "^1.3.4",
    "vue-router": "^2.6.0",
    "vue2-editor": "^2.3.34"
  },
  "devDependencies": {
    "autoprefixer": "^7.1.2",
    "babel-core": "^6.22.1",
    "babel-loader": "^7.1.1",
    "babel-plugin-transform-runtime": "^6.22.0",
    "babel-preset-env": "^1.3.2",
    "babel-preset-stage-2": "^6.22.0",
    "babel-register": "^6.22.0",
    "cash-dom": "^1.3.5",
    "chalk": "^2.0.1",
    "connect-history-api-fallback": "^1.3.0",
```

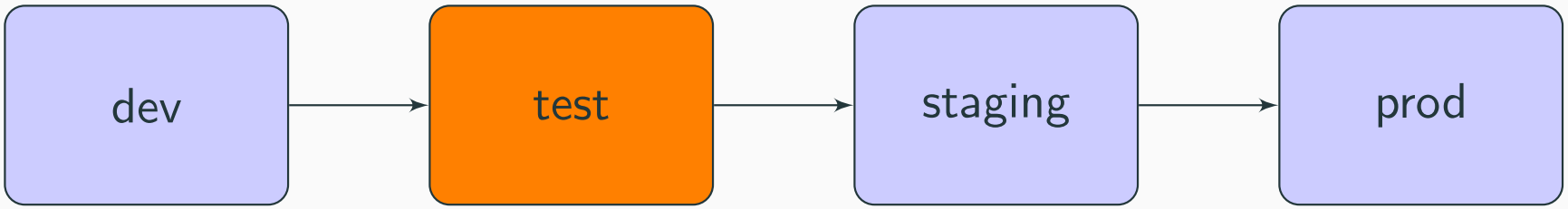
Rysunek 1: Podział na zależności wymagane dla środowiska deweloperskiego oraz innych

Środowisko testowe

Środowisko testowe

Środowisko testowe może funkcjonować zarówno jako drugie środowisko na maszynie deweloperskiej lub jako osobny byt. Jego celem jest uruchomienie wszelkiego rodzaju testów, które sprawdzają czy aplikacja działa w poprawny sposób.

ENV=test



Jak to działa od strony repozytorium?

Bez względu na wybrany sposób dzielenia gałęzi repozytorium projektu, dobrą praktyką jest utworzenie gałęzi głównej, ale nie będącej gałęzią `master`, która *zazwyczaj* jest zarezerwowana na ostateczny kod trafiający na serwer produkcjny.

Jak to działa od strony repozytorium?

Przykładowo dobrym pomysłem może być utworzenie gałęzi *dev*, do której progamiści będą włączać wszystkie swoje zmiany. Taką gałąź warto przetestować zanim aplikacja zostanie wypuszczona w świat.

Ale jakie testy?

Obecnie najpopularniejszym i na pewno wartym do poznania sposobem testowania aplikacji są **testy jednostkowe**.

Polegają - zgodnie z nazwą - na sprawdzaniu czy pojedyncze elementy aplikacji działają poprawnie. Elementami takimi mogą być metody, obiekty lub nawet grupy obiektow.

Testy jednostkowe

```
[TestMethod]
public void Withdraw_ValidAmount_ChangesBalance()
{
    // arrange
    double currentBalance = 10.0;
    double withdrawal = 1.0;
    double expected = 9.0;
    var account = new CheckingAccount("JohnDoe", currentBalance);

    // act
    account.Withdraw(withdrawal);
    double actual = account.Balance;

    // assert
    Assert.AreEqual(expected, actual);
}
```

Pliki środowiskowe

Niektóre frameworki bazują na plikach `.env`, w których w wygodny sposób można przetrzymywać istotne dane. Zasada mówi, że każde środowisko powinno mieć swój własny plik środowiskowy.

Ważnym jest zatem, aby nie dodawać ich nigdy do repozytorium, ponieważ zawierają dane, które a) będą niepoprawne dla różnych maszyn i b) niekoniecznie powinny być publiczne ze względów bezpieczeństwa.

Pliki środowiskowe

```
ENV=dev

APP_KEY=7dbba255-1c7d-46b4-8c5a
APP_URL=http://pwsz.local/

DATABASE_HOST=localhost
DATABASE_USERNAME=root
DATABASE_PASSWORD=alohomora
DATABASE_NAME=pwsz

LOGGER_LEVEL=debug
LOGGER_DRIVER=file
```

```
ENV=test

APP_KEY=b0f246b4-9f0a-43b0-a922
APP_URL=http://test.pwsz.local/

DATABASE_HOST=localhost
DATABASE_USERNAME=root
DATABASE_PASSWORD=alohomora
DATABASE_NAME=pwsz-test

LOGGER_LEVEL=error
LOGGER_DRIVER=file
```

Co dalej?

Dopiero gdy przejdą wszystkie testy automatyczne, najlepiej na środowisku lokalnym i testowym, aplikacja powinna zostać *pchnięta* dalej.

Istnieje kilka sposobów, aby dostarczyć zmiany na serwer:

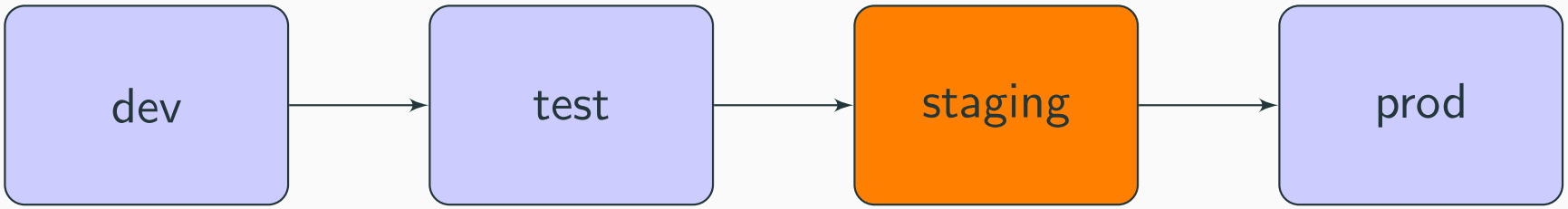
- najgorszy: FTP lub SCP
- najpopularniejszy: git pull
- najlepszy: CI

Środowisko stagingowe

Środowisko stagingowe

Środowisko stagingowe to pewnego rodzaju pokaz prapremierowy aplikacji. Na *scenie* uruchamiane są wszystkie procesy podłączone do własnych serwisów i zasobów, a klient ma możliwość sprawdzić czy wszystko działa, wygląda i zachowuje się tak jak powinno.

ENV=staging



Środowisko stagingowe

Czasami:

- programista wykonał wszystko zgodnie z dostarczoną specyfikacją projektu,
- wszystkim w zespole wydaje się, że wszystko jest okej,
- kod ma 100% pokrycia testami i wszystkie testy przechodzi śpiewająco...

... a i tak biznes odrzuci nową wersję aplikacji.

Środowisko stagingowe

Powodów może być mnóstwo:

- źle zrozumiany i zaimplementowany proces,
- nowe warunki walidacji,
- brak integracji z istniejącymi dotąd danymi,
- brzydki font, zły kolor przycisku, zbyt duży margines...

Środowisko stagingowe

O ileż lepiej przeprowadzić tak zwane testy manualne i mieć stuprocentową pewność, że wszystko jest okej?

Z tego powodu środowiska stagingowe wystawia się na publiczny serwer HTTP, często pod osłoną `.htaccess` lub innego sposobu uwierzytelniania, często pod domeną `beta.nazwastrony.pl` lub `test.nazwastrony.pl`, ale praktycznie zawsze na maszynie identycznej do tej, na której stoi serwer produkcyjny.

Środowisko stagingowe

To w tym miejscu administracja aplikacji może poznać nowe funkcjonalności, a upoważniona osoba odebrać zamówione zmiany.

Środowisko stagingowe

Oczywiście środowisko stagingowe powinno być wciąż w pewien sposób odseparowane od *prawdziwego świata*. Z tego powodu chociażby:

- bazy danych seeduje się adresami email w domenie własnej firmy lub `@example.com`,
- wyłączone lub podpięte pod inny klucz są analityki i śledzenia klientów,
- wciąż czasami włączone są powiadomienia serwisowe o błędach,
- ustawiony jest wyższy poziom zapisywania logów.

Środowisko stagingowe

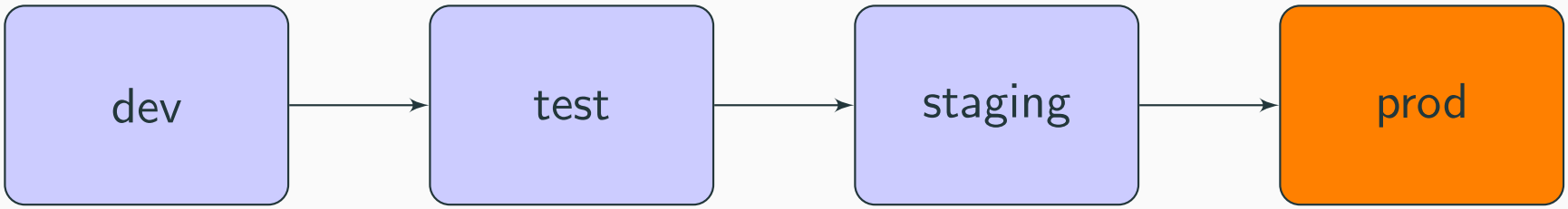
Jeżeli jednak wszystko gra, aplikacja powinna trafić na serwer produkcyjny już bez problemów.

Środowisko produkcyjne

Środowisko produkcyjne

Środowisko produkcyjne to ostatnia stacja na trasie wdrażania systemu internetowego. Jest to maszyna lub klaster maszyn, które są dostępne dla całego świata. Tutaj nie ma już miejsca na błędy i szybkie hotfiksy.

ENV=prod



Środowisko produkcyjne

Proces aktualizacji środowiska produkcyjnego powinien być zawsze przemyślany.

Istnieją sposoby, aby zrobić aktualizację oprogramowania (czasami wraz z testami!) w czasie coraz bardziej dążącym do zera. Jednym z takich rozwiązać jest *blue-green deployment*.

Środowisko produkcyjne

Jeżeli jednak z jakiegokolwiek powodu deploy robiony jest ręcznie, należy zrobić to w najodpowiedniejszej chwili:

- w godzinie najmniejszego obłożenia według logów serwera lub analityk wejść,
- w uprzednio zapowiedzianym czasie,
- w terminie, który pasuje klientowi.

Niestety nie ma jednego gotowego scenariusza, który pasowałby do każdego projektu.

Podsumowanie

Bibliografia i ciekawe źródła

- https://en.wikipedia.org/wiki/Deployment_environment
- https://msdn.microsoft.com/en-us/library/hh694602.aspx
- https://docs.docker.com/compose/env-file/
- https://martinfowler.com/bliki/BlueGreenDeployment.html

Pytania?

Kod prezentacji dostępny jest w repozytorium git pod adresem
`https://bitbucket.org/krewak/pwsz-ppsi`

Wszystkie informacje dot. kursu dostępne są pod adresem
`http://pwsz.rewak.pl/kursy/4`