Scalable Magento Development with Containers
About me

Andreas Koch

• Software developer at arvato SCM
• Located in Germany
• Passionate about
  – Deployment automation
  – Scalable software and processes
  – PHP, C# and Go

andreaskoch
andykdocs.de
arvato – Bertelsmann

PARENT COMPANY

ARVATO KEY FIGURES IN 2015

- Revenue: €4,847 mn.
- Operating EBITDA: €394 mn.
- Employees: 72,000
## arvato SCM Solutions

### Service portfolio

<table>
<thead>
<tr>
<th>Service</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shop Development</td>
<td></td>
</tr>
<tr>
<td>Digital Imaging</td>
<td></td>
</tr>
<tr>
<td>Customer Service</td>
<td></td>
</tr>
<tr>
<td>Transport Management</td>
<td></td>
</tr>
<tr>
<td>Financial Services</td>
<td></td>
</tr>
<tr>
<td>Omnichannel</td>
<td></td>
</tr>
<tr>
<td>Global Online Marketing</td>
<td></td>
</tr>
<tr>
<td>E-Commerce Consulting</td>
<td></td>
</tr>
<tr>
<td>Warehousing &amp; Distribution</td>
<td></td>
</tr>
<tr>
<td>Returns Management</td>
<td></td>
</tr>
<tr>
<td>Business Intelligence</td>
<td></td>
</tr>
<tr>
<td>Value added services</td>
<td></td>
</tr>
</tbody>
</table>

### Reference customers

- Levi's
- C&A
- Nivea
- Marc O’Polo
- Tom Tailor
- Coty
- Tommy Hilfiger
- Douglas
- Esprit
- Triumph
- Fressnapf
- Imagine 2016
Development, Hosting & Operations of Magento Shops
Hosting Magento is simple. Right?

Nginx, PHP, MySQL

DEV
Hosting Magento is simple. Right? Nope!

- Nginx
- PHP
- MySQL
- Redis
- ElasticSearch
- RabbitMQ
- Varnish

DEV
Challenge #1: Knowledge Transfer

«I dunno how to configure Nginx. Andy always does that. But he is currently on vacation in Vegas.» → Knowledge Monopolies
Challenge #2: Parallelization

«Don’t touch the integration environment the next days – we are going to test feature xyz!»

→ Parallelization
Challenge #3: Operations

«How could that have ever worked on the integration environment?!»

Operations

Nginx, PHP, MySQL, Redis, ElasticSearch, RabbitMQ and Varnish

Development Environment  DEV  ——Deployment——  ?  Production Environment

OPS
Challenge #3: Operations

Keeping environments in sync is hard.

Nginx, PHP 7.0.4, MySQL, Redis, ElasticSearch, RabbitMQ and Varnish

Development Environment

DEV

Patching

OPS

Production Environment
Challenge #4: Scaling

«Just a little something for the current sprint, Andy: We need a new integration and production for the new project. asap!»

Scaling
Introduction to Docker

Container virtualization made easy
What is Docker?

Docker automates the deployment of applications inside containers

- Based on Linux technologies like
  - LXC
  - cgroups
  - aufs
- Avoids the overhead of classical virtual machines
  - Small and portable containers
  - Ideal for application packaging
Solving the Challenges

1. Knowledge Monopolies
2. Parallelization
3. Operations
4. Scaling
Solving Knowledge Monopolies

Docker allows you to

• split your infrastructure components into immutable units of execution by using **standard Docker images** such as Nginx, PHP and MySQL

• make your software configuration transparent and traceable with **Dockerfiles**

• combine all your software dependencies into a single **docker-compose.yml** file
Leveraging Docker Standard Images

Docker provides a long list of standard images that are maintained by Docker or the software authors.

Operating System Images
- Alpine
- Ubuntu
- Debian
- CentOS
- ...

Software components
- PHP
- MySQL
- Nginx
- Redis
- ...

Docker Standard Images: PHP

Software components

- PHP
- MySQL
- Nginx
- Redis
- ...
Docker Standard Images: MySQL

Software components
- PHP
- MySQL
- Nginx
- Redis
- …
Docker Standard Images: Nginx

Software components
- PHP
- MySQL
- Nginx
- Redis
- …
Docker Standard Images: Redis

Software components
- PHP
- MySQL
- Nginx
- Redis
- ...
FROM ubuntu:latest

RUN apt-get update && apt-get install nginx

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]

1. Base image

2. Setup commands

3. Expose ports

4. Entry point
FROM debian:jessie

MAINTAINER NGINX Docker Maintainers "docker-maint@nginx.com"

ENV NGINX_VERSION 1.9.12-1-jessie

RUN apt-key adv --keyserver hkp://pgp.mit.edu:80 --recv-keys 573BFDB6B3DBFBC641079A6ABE586D3B7BD9BF62 \ 
    && echo "deb http://nginx.org/packages/mainline/debian/ jessie nginx" >> /etc/apt/sources.list \ 
    && apt-get update \ 
    && apt-get install -y \ 
    ca-certificates \ 
    nginx=${NGINX_VERSION} \ 
    nginx-module-xslt \ 
    nginx-module-geoip \ 
    nginx-module-image-filter \ 
    gettext-base \ 
    \ 
    && rm -rf /var/lib/apt/lists/*

# forward request and error logs to docker log collector
RUN ln -sf /dev/stdout /var/log/nginx/access.log \ 
    && ln -sf /dev/stderr /var/log/nginx/error.log

EXPOSE 80 443

CMD ["nginx", ",-g", ",daemon off;" ]
Running a Default Nginx Container

```
$ docker run -p 8080:80 nginx:1.9
```
Customization via Docker Volumes

Customizing dockers container with volumes:

```bash
$ docker run -p 8080:80
   -v `pwd`/custom-content:/usr/share/nginx/html
   nginx:1.9
```
Customization via Docker Volumes
Customization via Docker Volumes
Orchestration with Docker Compose

- Multi-container orchestration
- Define infrastructure setups
- YAML-based config format
- Define
  - Images
  - Ports
  - Volumes

```yaml
version: '2'

services:
  web:
    image: nginx:stable-alpine
    ports:
      - 8080:80
    volumes_from:
      - php:ro

  php:
    image: php:7.0-fpm
    volumes:
      - ./web:/var/www/html
```

docker-compose.yml
Orchestration with Docker Compose

- Multi-container orchestration
- Define infrastructure setups
- YAML-based config format
- Define
  - Images
  - Ports
  - Volumes

```
version: '2'

services:
  web:
    image: nginx:stable-alpine
    ports:
      - 8080:80
    volumes_from:
      - php:ro

  php:
    image: php:7.0-fpm
    volumes:
      - ./web:/var/www/html
```

`docker-compose.yml`
Orchestration with Docker Compose

- Multi-container orchestration
- Define infrastructure setups
- YAML-based config format
- Define
  - Images
  - Ports
  - Volumes

```yaml
version: '2'

services:
  web:
    image: nginx:stable-alpine
    ports:
      - 8080:80
    volumes_from:
      - php:ro

  php:
    image: php:7.0-fpm
    volumes:
      - ./web:/var/www/html
```

*docker-compose.yml*
Orchestration with Docker Compose

- Multi-container orchestration
- Define infrastructure setups
- YAML-based config format
- Define
  - Images
  - Ports
  - Volumes

```yaml
version: '2'
services:
  web:
    image: nginx:stable-alpine
    ports:
      - 8080:80
    volumes_from:
      - php:ro
  php:
    image: php:7.0-fpm
    volumes:
      - ./web:/var/www/html

docker-compose.yml
```
Orchestration with Docker Compose
Solving Knowledge Monopolies

Automate the environment setup with Docker Compose

Nginx, PHP, MySQL, Redis, ElasticSearch, RabbitMQ and Varnish

Manual Setup

DEV

DEV
Solving Knowledge Monopolies

Automate the environment setup with Docker Compose
Solving the Challenges

1. Knowledge Monopolies ✓
2. Parallelization
3. Operations
4. Scaling
Enable Parallelization

Combination of tools and processes:

- Docker Compose
- Git
- Composer

Combine infrastructure and software

- git-flow
  Branch on feature level

- Cloud hosting
  Create feature instances for testing on-the-fly

→ Develop and test features in parallel
Enable Parallelization

Get rid of bottlenecks
Enable Parallelization

Get rid of bottlenecks … by creating feature instances
Enable Parallelization

Create **feature-instance** with the arvato CloudAPI on-the-fly:

![Cloud API Interface](image)

**Create a cloud instance**

- **Project**: Demo Shop
- **Branch**: feature/140097-wysiwyg
- **Environment**: Default
- **Server Name**: demo-feature-140097-wysiwyg
- **Server-Flavor**: 1 GB General Purpose v1

Recommended "1GB General Purpose v1" ($0.0035/hr). For the prices of other flavors click here: Prices.
Solving the Challenges

1. Knowledge Monopolies ✓
2. Parallelization ✓
3. Operations
4. Scaling
Ease Operations

Docker and Docker Compose will reduce operation efforts:

- **Better quality.** All environments look the same
Ease Operations

Docker and Docker Compose will reduce operation efforts:

- **Better quality.** All environments look the same

- **Easier patching.** Download the latest Docker images and restart
Ease Operations

Docker and Docker Compose will reduce operation efforts:

• **Better quality.** All environments look the same

• **Easier patching.** Download the latest Docker images and restart

• **Easier handling.** All projects can be controlled by the same commands
Ease Operations

Docker and Docker Compose will reduce operation efforts:

• **Better quality.** All environments look the same

• **Easier patching.** Download the latest Docker images and restart

• **Easier handling.** All projects can be controlled by the same commands

• **Easier handover.** Between development and operations team(s)
Ease Operations: Deployment & Patching

1. **Copy** the new project version to production
2. **Pull** the latest Docker images
3. **Run** the new project version
Solving the Challenges

1. Knowledge Monopolies ✓
2. Parallelization ✓
3. Operations ✓
4. Scaling
Scaling

Scaling can mean different things:

• More Load
• More Features
• More Businesses
Scaling

What can you do to scale?

• Add more developers and testers to your team
• Optimize the infrastructure and code
• Add more RAM, CPU and storage to your existing servers
• Add new servers to your production environment
Complexity with Automation

- Load
- Features
- Businesses

Complexity

More
Complexity without Automation

![Graph showing the increase in complexity with more load, features, and businesses.](image_url)
Scaling with Docker

- **More Load**: Scale horizontally or vertically
- **More Features**: We can now parallelize our development
- **More Businesses**: Copy an existing project and customize it

Still not easy. But easier than without it!
Solving the Challenges

1. Knowledge Monopolies ✓
2. Parallelization ✓
3. Operations ✓
4. Scaling ✓
Combining Infrastructure & Code

Project structure of a dockerized Magento shop
Combining Code + Infrastructure

- app
- bin
- vendor
- pub
- composer.json
- index.php

- config
  - mysql
  - nginx
  - php
  - docker-compose.yml
Handling different environments

- app
- ...
- config
  - mysql
  - nginx
  - php-dev
  - php-prod
- docker-compose.yml
- docker-compose.prod.yml

Configs for other environments should also be tracked in the same project, e.g.

- The production config of your PHP container should not contain Xdebug
- Usage of a central MySQL cluster on production
Handling different environments

- app
- ...
- config
  - mysql
  - nginx
  - php-dev
  - php-prod
- docker-compose.yml
- docker-compose.prod.yml

Using a different Docker Compose file:

```
> docker-compose -f docker-compose.prod.yml up -d
```
Development Environment

*Using Docker on Linux, OS X and Windows*
Development Environment

The dockerized development process works on Linux, OS X & Windows. All you need is:

- Docker Machine (only for Mac and Windows)
- Docker & Docker Compose
- Git + PHP + Composer

+ An editor or IDE of your choice
Linux

Install Docker:

> curl -fsSL https://get.docker.com/ | sh

and Docker Compose:

> curl -L https://github.com/docker/compose/releases/download/1.7.0-rc1/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
<?php
    /**
     * Public alias for the application entry point ...
     */

    use ...

    try {
        require realpath( _DIR_ ). '/../app/bootstrap.php';
    }
    catch (Exception $e) {
        echo <<<HTML
            <div style="font:12px/1.35em arial, helvetica, sans-serif;">
                <h3 style="...">AutoLoad error</h3>
            </div>
        HTML;
        echo $e->getMessage();
    }
?>
Local path: `~/magento2`

Path in container: `/var/www/html`
Docker for Windows and Mac

Install the Docker Toolbox to use Docker on Mac or Windows
Docker Machine

Usage: docker-machine [OPTIONS] COMMAND [arg...]

Create and manage machines running Docker.

Version: 0.6.0, build e27fb87

Author:
Docker Machine Contributors

Options:
--debug, -D
-s, --storage-path "/[Users/STORAGE_PATH]]
--tls-ca-cert
HINE_TLS_CA_CERT]
--tls-ca-key
es [MACHINE_TLS_CA_KEY]
--tls-client-cert
NE_TLS_CLIENT_CERT]
--tls-client-key
docker-machine ls
docker-machine create
Docker Machine
Using Docker on Linux, Mac and Windows

- Linux works best
- Mac OS works good
  - Use `docker-machine-nfs` for better performance
- Windows is (imho) a bit awkward to use
- Always get your file system permissions right!
Demo

Dockerizing Magento 2
Dockerizing Magento 2

1. Create a Magento 2 project
2. Install `arvatoscm/dockerize-magento2`
3. Start the Docker infrastructure and install Magento
Tool check
docker-machine version
docker-machine ls
docker version
docker-compose version
git version
git flow version
composer --version
composer create-project
composer create-project

Authentication component for Identity plugin
zendframework/zend-mvc suggests installing zendframework/zend-session (Zend\Session component for FlashMessenger, PRG, and FPRG plugins)
zendframework/zend-mvc suggests installing zendframework/zend-version (Zend\Version component)
zendframework/zend-crypt suggests installing ext-mcrypt (Required for most features of Zend\Crypt)
phpunit/php-code-coverage suggests installing ext-xdebug (>=2.2.1)
phpunit/phpunit suggests installing phpunit/php-invoker (~1.1)
symfony/dependency-injection suggests installing symfony/proxy-manager-bridge (Generate service proxies to lazy load them)
sjparkinson/static-review suggests installing sensiolabs/security-checker (Required for ComposerSecurityReview.)
lusitanian/oauth suggests installing symfony/http-foundation (Allows using the Symfony Session storage backend.)
lusitanian/oauth suggests installing predis/predis (Allows using the Redis storage backend.)
Writing lock file
Generating autoload files
git flow init

ony Session storage backend.)
lusitanian/oauth suggests installing predis/predis (Allows using the Redis storage backend.)
Writing lock file
Generating autoload files

$ ~ cd magento2
$ magento2 ls
CHANGELOG.md
CONTRIBUTING.md
CONTRIBUTOR_LICENSE_AGREEMENT.html
COPYING.txt
Gruntfile.js
LICENSE.txt
LICENSE_AFL.txt
README.md
app
bin
composer.json
composer.lock
$ magento2
$ magento2 git flow init
Initialized empty Git repository in /Users/andyk/magento2/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
$ magento2 git:(develop) ✗
git flow feature start

➜ magento2  git:(develop)
Install arvatoscm/dockerize-magento2

$ magento2 git:(develop) git flow feature start dockerize
Switched to a new branch 'feature/dockerize'

Summary of actions:
- A new branch 'feature/dockerize' was created, based on 'develop'
- You are now on branch 'feature/dockerize'

Now, start committing on your feature. When done, use:

    git flow feature finish dockerize

$ magento2 git:(feature/dockerize)
Adapt .gitignore

$ magento2 git:(feature/dockerize) composer require --ignore-platform-reqs \ > arvatoscm/dockerize-magento2

Using version ^1.0 for arvatoscm/dockerize-magento2
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing arvatoscm/dockerize-magento2 (v1.0.0)
    Downloading: 100%

Writing lock file
Generating autoload files
$ magento2 git:(feature/dockerize)
Review installed files

- magento2 git:(feature/dockerize) × git status
- On branch feature/dockerize
- Changes not staged for commit:
  - (use "git add <file>..." to update what will be committed)
  - (use "git checkout -- <file>..." to discard changes in working directory)
    - modified: composer.json
    - modified: composer.lock
- Untracked files:
  - (use "git add <file>..." to include in what will be committed)
    - .gitignore
    - config/
    - dev/
    - docker-compose.yml
- no changes added to commit (use "git add" and/or "git commit -a")
- magento2 git:(feature/dockerize) ×
docker-compose.yml

```
version: '2'
services:
  web:
    image: nginx:latest
    ports:
      - "80:80"  
      - "443:443"  
    volumes:
      - ./config/nginx/sites-enabled/default.conf:/etc/nginx/sites-enabled/default.conf
      - ./config/nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./config/nginx/ssl:/etc/nginx/ssl
      - ./config/nginx/auth:/etc/nginx/auth
    volumes_from:
      - appdata:ro
    depends_on:
      - php

  php:
    build: config/php/image
    volumes:
      - ./config/php/php.ini:/usr/local/etc/php/php.ini
```

composer.json

```
{"repositories": ["type": "composer", "url": "https://repo.magento.com/"],
  "require": {
    "magento/product-community-edition": "2.0.4",
    "composer/composer": "@alpha",
    "arvatoscm/dockerize-magento2": "^1.0"
  },
  "require-dev": {
    "phpunit/phpunit": "4.1.0",
    "squizlabs/php_codesniffer": "1.5.3",
    "phpmd/phpmd": "2.3.*",
    "pdepend/pdepend": "2.2.2",
    "sниксон/static-review": "~4.1",
    "fabpot/php-cs-fixer": "~1.2",
    "lusitanian/oauth": "~0.3 <= 0.7.0"
  },
  "config": {
    ...
  }
}
```
config folder

```
++app/
++bin/
~config/
  |++appdata/
  |++mysql/
  |++nginx/
  |++php/
  |  |++image/
  |  |  |ext-xdebug.ini
  |  |  |php-fpm.conf
  |  |  |  |php.ini
  |~env.sh
+dev/
+lib/
+phpserver/
+pub/
+setup/
+update/
+var/
+vendor/
~CHANGELOG.md
~composer.json
/Users/andyk/magento2
"config/php/php.ini" 37L, 802C
```
bin/console script

# A proxy script that executes bin/magento inside the PHP docker container.

SCRIPTNAME="bin/$(basename $0)"

PROJECTPATH=$(pwd)

CONFIG_FOLDER contains the path to the config folder.

SSL_CREDENTIALS_FOLDER contains the path to the SSL certificate folder that is used by Nginx.

ENVIRONMENT_VARIABLES_FILE contains the path to the environment variables file.
Make bin/console executable
Install Magento 2

```bash
magento2 git:(feature/dockerize) X bin/console
Utility for controlling dockerized Magento projects

Usage:

    bin/console <action> <arguments...>

Actions:

    install     Install Magento
    exec        Execute bin/magento inside docker
    start       Start the server and all of its components
    restart     Restart the server
    stop        Stop the server
    status      Get the current server status
```

© 2023 Magento
Install Magento 2: Generate SSL Certificates

```bash
magento2 git:(feature/dockerize) ✗ bin/console install docker.local
Generated new SSL certificates for docker.local:
    Key: /Users/andyk/magento2/config/nginx/ssl/key.pem
    Cert: /Users/andyk/magento2/config/nginx/ssl/cert.pem
```
Install Magento 2: Start containers

```
magento2 git:(feature/dockerize) $ bin/console install docker.local
Generated new SSL certificates for docker.local:
  Key:  /Users/andyk/magento2/config/nginx/ssl/key.pem
  Cert: /Users/andyk/magento2/config/nginx/ssl/cert.pem
```
Install Magento 2: Download Docker Images

```
$ magento2 git:(feature/dockerize) X bin/console install docker.local
Generated new SSL certificates for docker.local:
  Key: /Users/andyk/magento2/config/nginx/ssl/key.pem
  Cert: /Users/andyk/magento2/config/nginx/ssl/cert.pem
Creating network "magento2_default" with the default driver
Creating magento2_appdata_1
Creating magento2_phpmyadmin_1
```
Install Magento 2: Trigger Magento Installer

```
1d007c18c656: Pull complete
ad75a8697e9c: Pull complete
30e2a5e0acbe: Pull complete
5db2b51e0e9: Pull complete
8f359895dbf8: Pull complete
Digest: sha256:3df6902f054108596c35b7e44a774d518b960e42761ea8075ae264ee5ed5f100
Status: Downloaded newer image for redis:latest
Creating magento2_cache_1
Creating magento2_mysql1_1
Creating magento2_php_1
Pulling web (nginx:latest)...
latest: Pulling from library/nginx
fdd5d7827f33: Already exists
a3ed95caeb02: Pull complete
716f7a5f3082: Pull complete
7b10f093a0309: Pull complete
Digest: sha256:f6a001272d5d324c4c9f3f183e1b69e9e0ff12debeb7a092730d638c33e0de3e
Status: Downloaded newer image for nginx:latest
Creating magento2_web_1
```
Install Magento 2: Display Shop URLs

Module 'Magento_GoogleAnalytics':
Module 'Magento_GoogleOptimizer':
Module 'Magento_GroupedImportExport':
Module 'Magento_GroupedProduct':
Module 'Magento_DownloadableImportExport':
Module 'Magento_Checkout':
Module 'Magento_Integration':
Module 'Magento_LayeredNavigation':
Module 'Magento_Marketplace':
Module 'Magento_MediaStorage':
Module 'Magento_CatalogRule':
Module 'Magento_Multishipping':
Module 'Magento_ConfigurableProduct':
Module 'Magento_Newsletter':
Module 'Magento_OfflinePayments':
Module 'Magento_SalesRule':
Module 'Magento_PageCache':
Module 'Magento_Captcha':
Module 'Magento_Paypal':
Module 'Magento_Persistent':
translate
config_webservice
Customer Grid index has been rebuilt successfully in 00:00:01
Category Products index has been rebuilt successfully in 00:00:00
Product Categories index has been rebuilt successfully in 00:00:00
Product Price index has been rebuilt successfully in 00:00:00
Product EAV index has been rebuilt successfully in 00:00:00
Stock index has been rebuilt successfully in 00:00:00
Catalog Rule Product index has been rebuilt successfully in 00:00:01
Catalog Product Rule index has been rebuilt successfully in 00:00:00
Catalog Search index has been rebuilt successfully in 00:00:00

Installation complete.

Frontend http://docker.local/
Backend https://docker.local/management
Username: admin
Password: enAVINa2

magento2 git:(feature/dockerize)
Try it yourself

1. Install arvatoscm/dockerize-magento2

2. Have fun with Magento & Docker
Feedback, questions and pull-requests are welcome.

github.com/arvatoSCM/dockerize-magento2

Thank you.