shoptimax
Ihr E-Commerce-Experte

SINK OR SWIM

~~LOCAL~~ SOFTWARE-DEVELOPMENT ~~WITH DOCKER~~ IN HETEROGENEOUS TEAMS!?

About me

- Software- /DevOps-Engineer (PHP, JavaScript, Java, Docker, Ansible, Gitlab,…)
- IT / E-Commerce since 1999
- > 10 years OXID experience
- @upsettweety
- www.shoptimax.de
- moises@shoptimax.de
- Using Docker since 2015

shoptimax
Ihr E-Commerce-Experte

Preface

- I love Docker :)
- but … there are some points we need to discuss
- we are using Docker for local development since > 2 years
- multiple „experiments“ with Docker inside VBox, Vagrant, Teracy, …
- is local development always the best option? **SPOILER** - probably not :)
- sometimes, we need a **fallback / alternative**!

A BRIEF HISTORY OF (OUR) DEVELOPMENT WORKFLOW

FROM XAMP AND FTP TO DOCKER AND CI

The beginnings

file:///Users/Me/projects/site/index2.html

(S)FTP

vbox.dev.local/site2/wp

MAMP

LAMP

XAMP

http://dev.chaos.org/shop37

Vagrant

Virtualization - Huge Progress

vmware

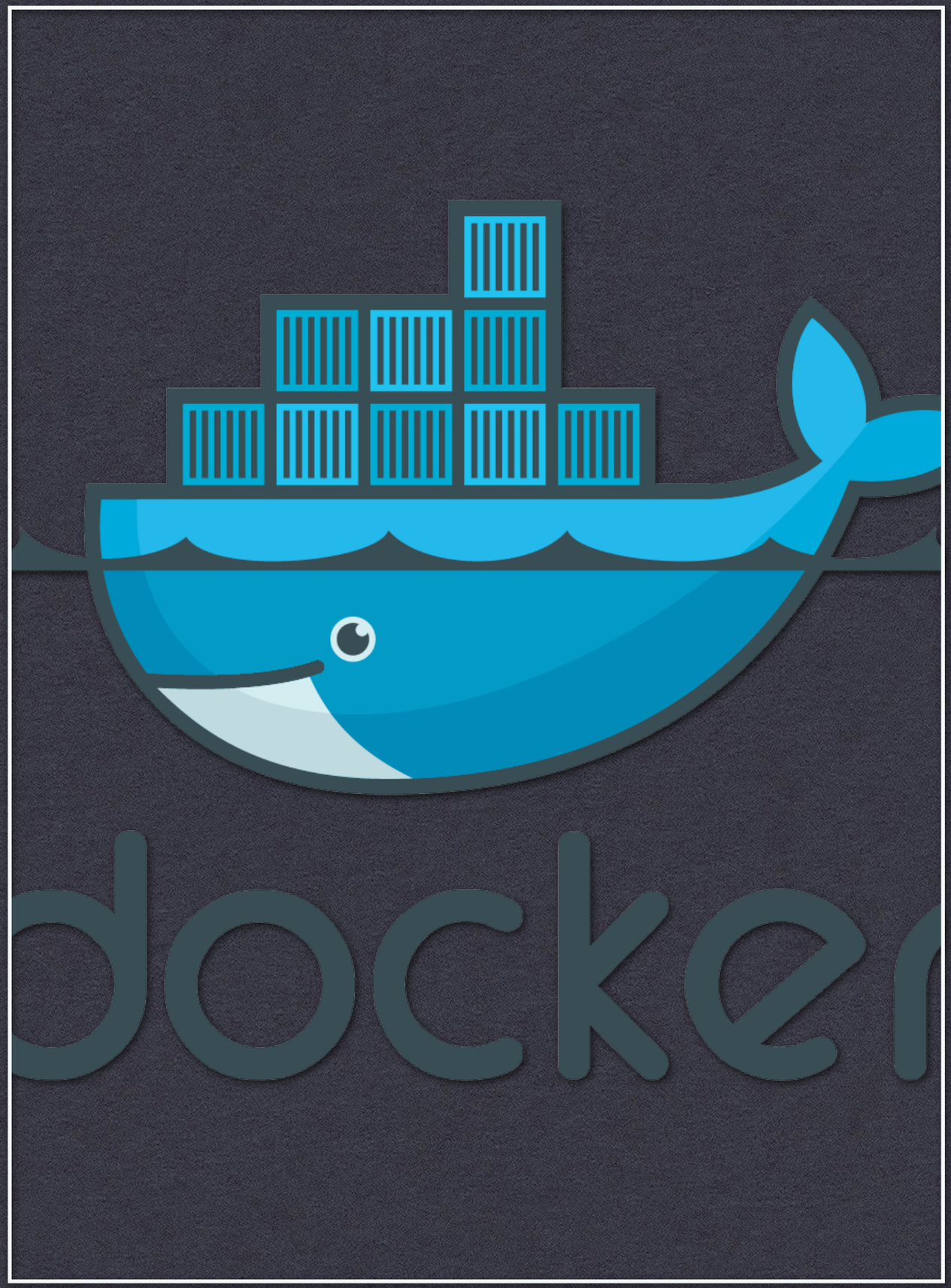
VAGRANT

But …

- lots of **monolithic VMs**
- sometimes **bloated and slow**
- sometimes bad **performance**, high **resource usage**
- eating up **disc space**
- **outdated** OS, cumbersome updates
- hard to run **multiple VMs** simultaneously

THE HOLY ~~GR~~WHALE?

DOCKER
DOCKER TOOLBOX
DOCKER FOR MAC
DOCKER FOR WINDOWS



First steps: „VM-Replacement“ as Antipattern

„Getting better“ w. Compose

phpbrew
PHP 5.6
PHP 7.0

ZendGuard
IonCube
Blackfire
XHProf
XHGui

Super-visor

MySQL

Docker Compose

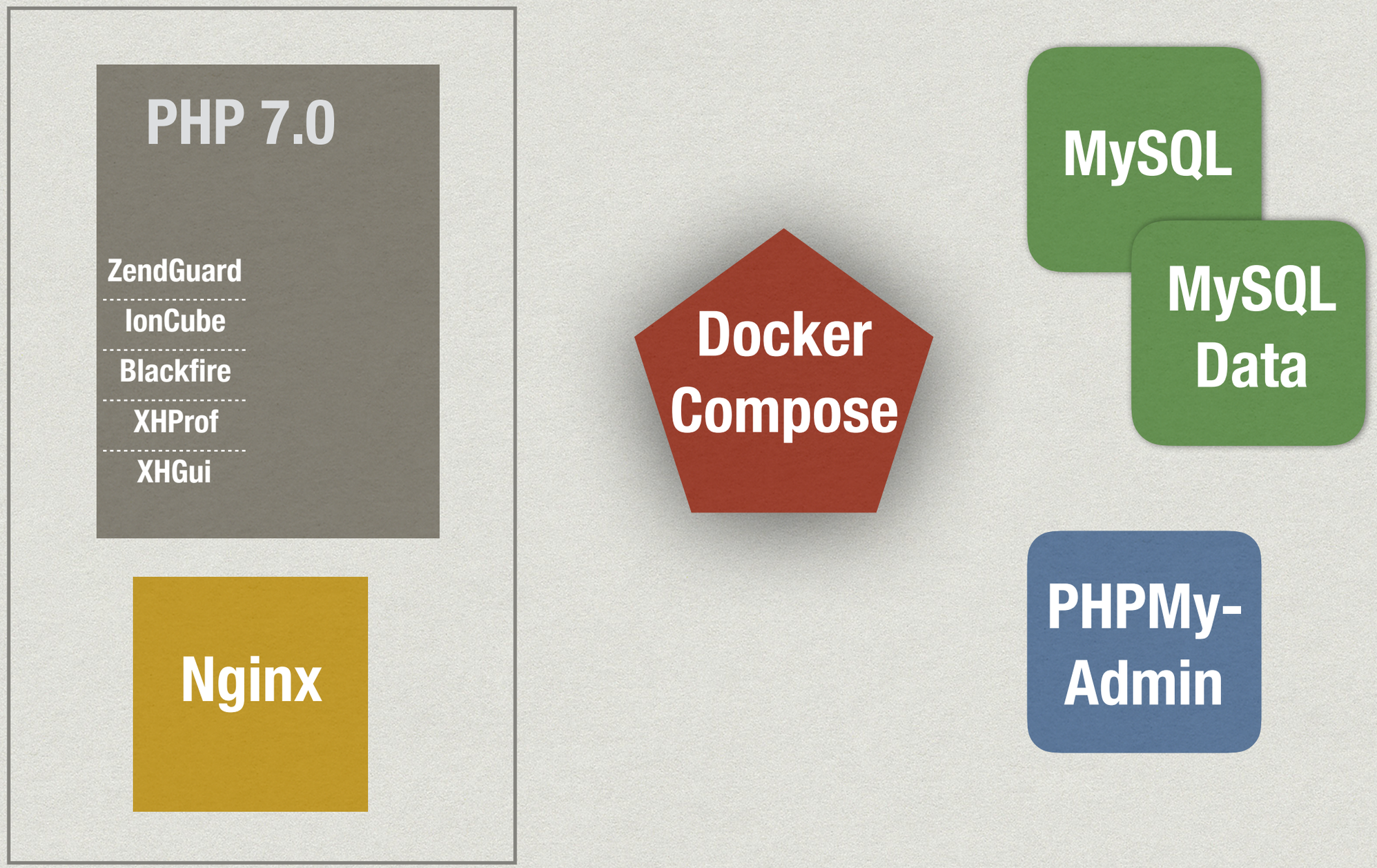
Apache

startup.sh

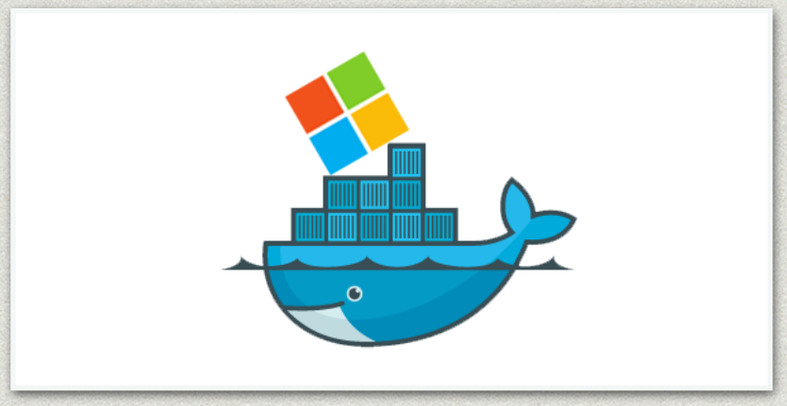
- git clone …
- mkdir …
- compose update
- …

PHPMy-Admin

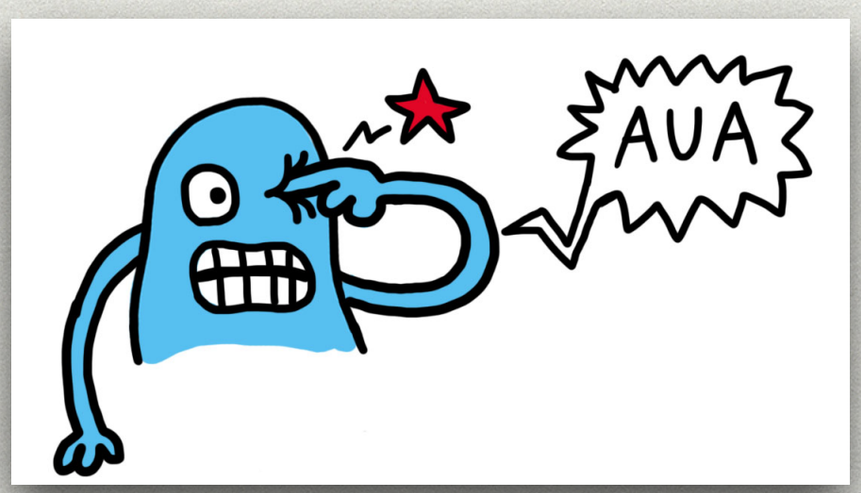
„And better …“


PHP 7.0
ZendGuard
IonCube
Blackfire
XHProf
XHGui
Nginx
Docker Compose
MySQL
MySQL Data
PHPMy-Admin

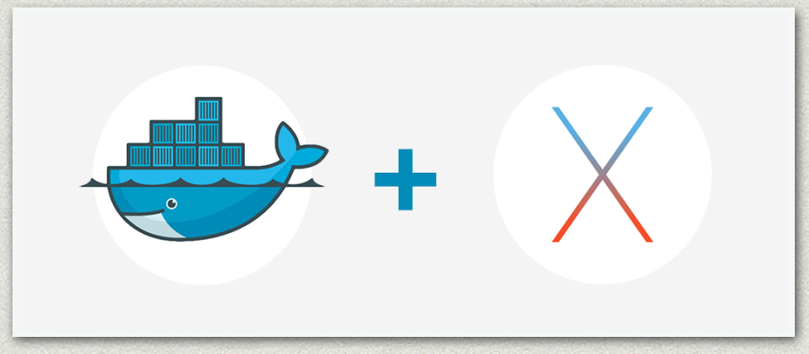
„Build once, run anywhere“?!



- „Hyper-V is **not supported** on my machine :(“
- „My machine is **slow and lagging** when I start the shop **and** *PHPStorm*.“
- „since the last XY **update** Docker isn’t working anymore“
- „Symfony / Shopware / OXID takes **30 seconds for one page** to load on Windows“



- „**Docker crashed**, I have to rebuild everything again“
- „**Bash-Script error** '…^M: not found’“
- „My **C:\ drive** isn’t accessible in Docker, only a restart helps“
- „Importing the **database dump** takes 45 minutes…“
- „I want to **shutdown** windows every evening and I’m having problems restarting the containers in the morning“
- „I only want to **change 2 lines of CSS/LESS**, why do I need to wait 45 minutes to insert the db, etc. until I can use the shop!?“



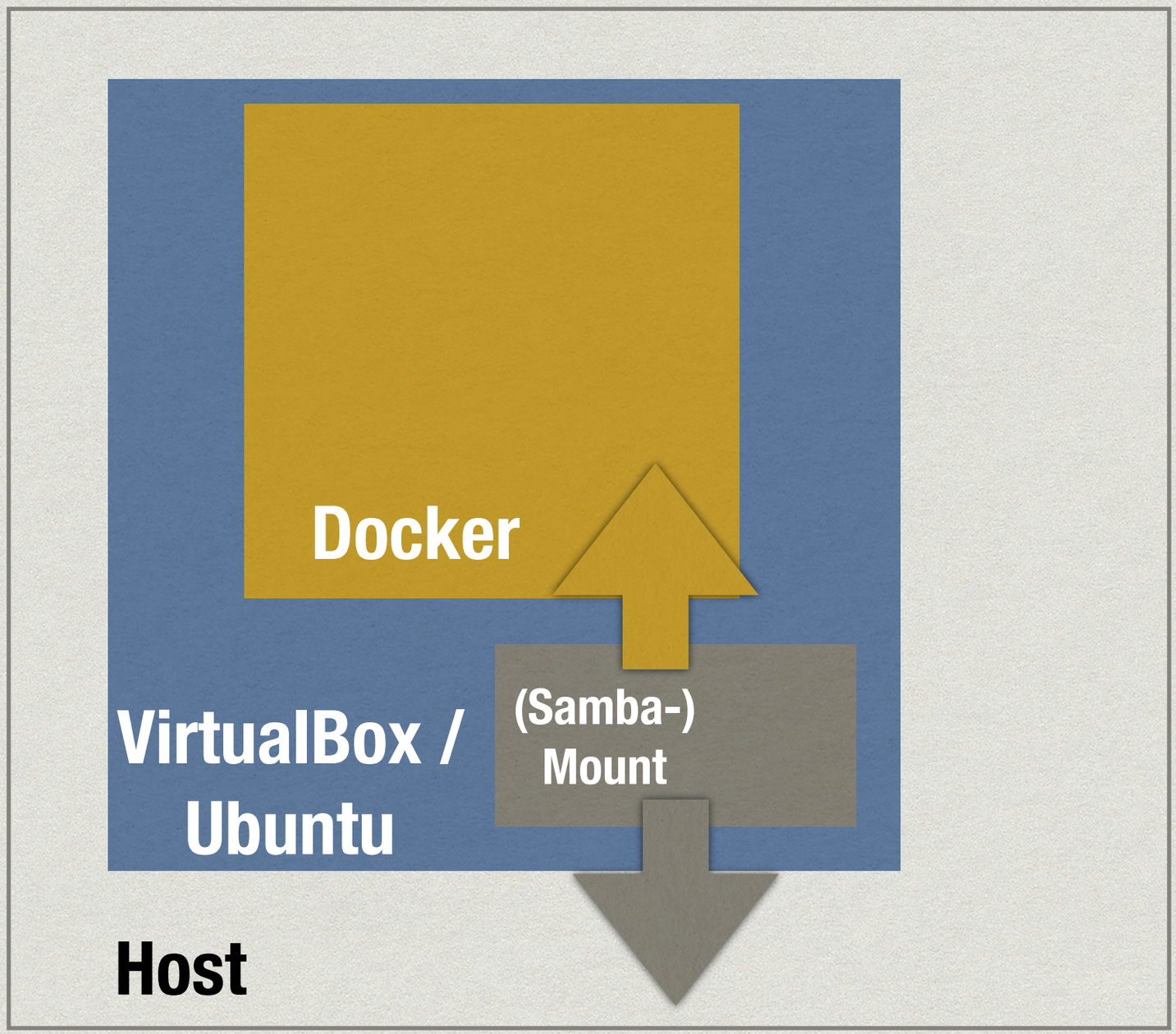
3 problem areas for local development

1. different behavior and problems between Linux / Windows (Versions) / MacOS
2. general performance on MacOS and especially on Windows
3. handle big/complex applications / shops (performance, time to setup, disc space, etc.) on user machines

EXPERIMENT 1: DOCKER IN A BOX!?

DOCKER WITH VIRTUALBOX / VMWARE - UNIFY DOCKER BETWEEN LINUX, MAC AND WINDOWS?

Host - VirtualBox - Docker



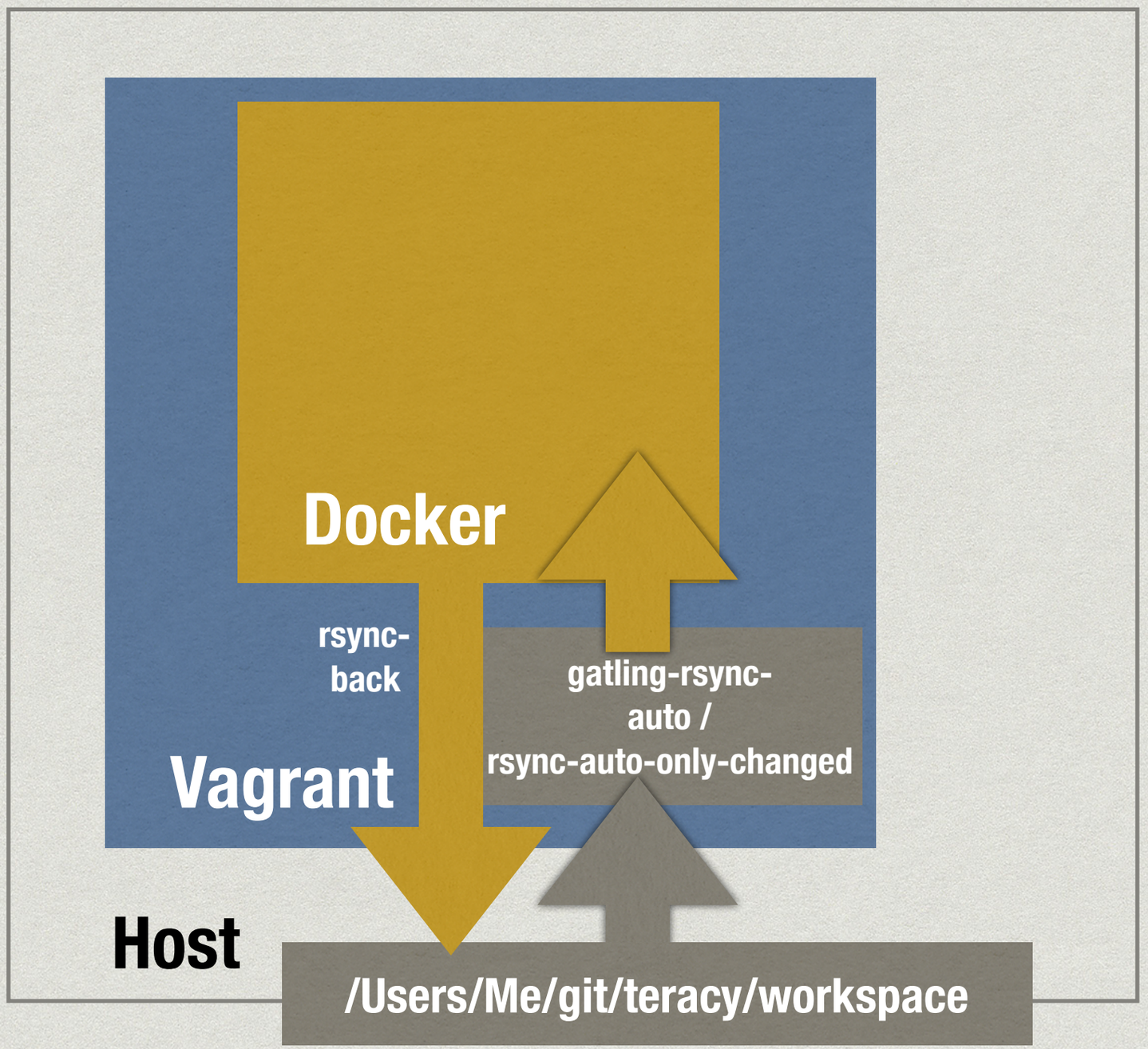
Nice plan, but …

- PHPStorm, Netbeans, Git-Clients, … don’t like network mounts … slooooow!
- workaround: duplicate local directory (one for the VM, one for the IDE), copy „on save“ into „VM folder“ on Samba-Mount … kinda ugly
- „My VM is broken!!“ (Day 1)
- …

Experiment 2 - Teracy-dev - Vagrant + Docker + Plugins

„Teracy-dev – the Only Truly Universal Productive Development Platform With Docker on macOS, Linux and Windows“

Host - Vagrant - Docker



Teracy - Pros

- Symfony / Shopware / OXID are relatively **fast** thanks to rsync-Plugin

- vagrant **suspend / resume** => Docker containers may keep running, even during Windows reboots :)

- **„unified“ (well, kind of …)** Docker environments, i.e. between WIN/MAC

Teracy - Cons

- adds lots of **additional complexity**, complex setup
- rsync plugin **not very reliable**, stops working every now and then
- all project files have to reside in **„workspace“ folder**
- **Performance:** rsync has problems if you have lots of projects in the workspace folder, takes up to 15s after a change!
- for the DevOps/Admin-Team you now have **Teracy- and Vagrant-Support** instead of / in addition to Docker-Support! :P

„And now?“

–A valid question …

Current state - simple Vagrant VM with Docker inside

- simple **Ubuntu VM** with **Docker** inside, provisioned with **Vagrant**
- multiple containers per project (apache, mysql, mysql data, solr, …) with J. Wilders' **Nginx Proxy (every project can use port 80 then for the webserver)**
- using **synced_folder** in Vagrant
- **kind of** unifies Docker on MAC and Windows
- Users can **pause / resume** VM to keep Containers running
- usually, only parts of the shop are mounted, e.g. modules and template folders (**performance**!)
- project **MAKEFILEs**, launched inside the VM for common project tasks (docker-compose up/down/pull, OXID Composer install, DB backup, insert dump, run grunt, index Solr, etc.)
- e.g. *„make install"* in VM gives you a ready-to-use OXID 6 shops with activated modules (via oxrun), project data, etc.
- but **still problems** with Windows, Performance, Setup, **limited PC resources** etc.

MAKEFILE

```
root:
    docker exec -ti $(APP) bash

fixdos:
    find . -name '*.sh' -exec dos2unix {} +

builddb: up
    # delete old archives
    docker exec $(DB) sh -c "rm -f /var/tmp/*.tar.gz"
    # download tar.gz defined in docker-compose.yml
    docker exec $(APP) /bin/bash /usr/bin/ddb
    docker cp ./db/buildscripts/builddb.sh $(DB):/var/tmp/builddb.sh
    docker cp ./db/buildscripts/sqldump/. $(DB):/var/tmp/
    # delete *sql files
    docker exec $(DB) sh -c "rm -f /var/tmp/*.sql"
    # run script
    docker exec $(DB) /bin/bash /var/tmp/builddb.sh

dbbackup:
    docker exec $(APP) /bin/bash /usr/bin/backup-databases -d app -b /var/www/html/$(PROJECTNAME)/source/_sql

copymedia:
    docker cp ./media $(APP):/var/www/html/$(PROJECTNAME)/source/

cleantmp:
    docker exec $(APP) sh -c "cd /var/www/html/$(PROJECTNAME)/source/ && rm -Rf tmp/*"

initgrunt:
    docker exec $(APP) sh -c "cd /var/www/html/$(PROJECTNAME)/source/Application/views/flow && yarn global add grunt-cli
```

Still problems …

- Docker / Docker Compose / Vagrant is **not intuitive for everyone**
- **gets worse if you combine Docker - Vagrant - VirtualBox (+ Teracy)!**
- you need lots of documentation, trainings, FAQs, **constant support**(-Chat), … and quick help for developers
- **„ unified Docker in Vagrant“ concept** still has problems as soon as different systems and users come into play (line endings, Vagrant versions, Windows versions, RAM, …)!
- **performance, speed and space problems** on local machines with Vagrant/Docker (RAM usage, HDD usage, mounted directories sync speed, etc.)

=> **local dev environments with Vagrant / Docker aren't always the solution or the preferred way to work for everyone or for every project!**

„Ich installiere mir lokal eine OXID EE mit 12 Subshops, 500 MB Datenbank-Dump, 2 GB Bilddaten, wenn ich eine LESS-Datei ändern und testen will?!?“

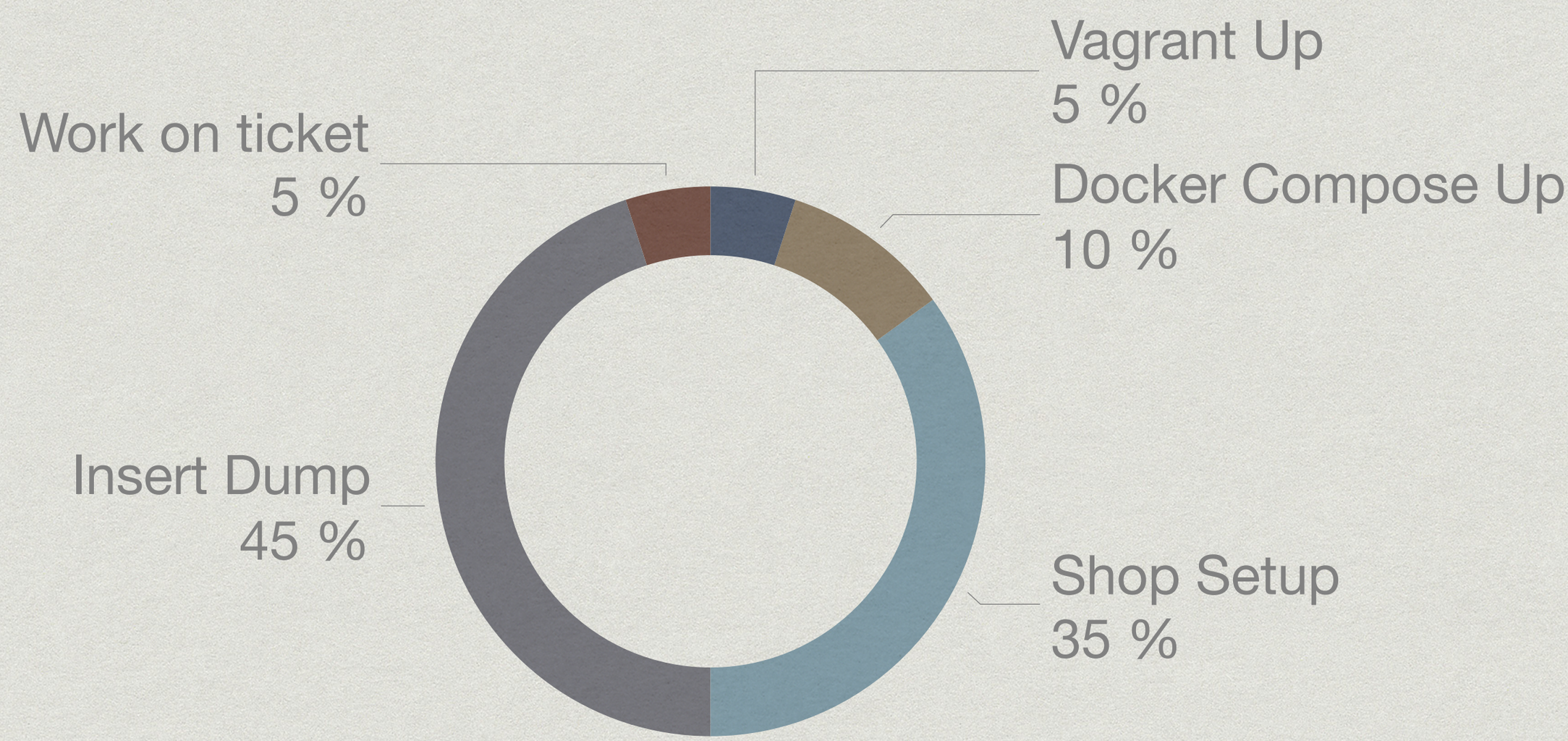
„I’m supposed to install OXID EE with 12 subshops, a 500 MB database dump and 2 GB of media data, if all I want is to change one single LESS file?!?“

–unknown Frontend Dev

Sometimes I feel like …

Productivity!?

„Why did that ticket take that long!?“



„So … what now!?“

–Helpless DevOps Engineer

Any ideas / alternatives to local development?

- „Cloud-VM“ for every Developer (Docker, Kubernetes,….)?
- Back to the roots - WAMP / XAMP / LAMP?

„Ah, Cloud, Kubernetes, AWS … - I love that stuff, let’s use it! :)“

–Anonymous DevOps Engineer

KISS

- Keep it simple, stupid!

KISS

- hm… we already use **(Gitlab) CI and Ansible** for deploying stage and production servers automatically … how about deploying **„custom dev environments“** for every developer?

Server 1 (DEV)

Server 2 (TEST)

Server 3 (STAGE)

Server 4 (LIVE)

ioly cookbook

- gn2processor.json
- ocb_cleartmp.json
- vtdevutils.json
- etc.

Ansible

Docker (Apache, PHP, MySQL, Codeception, PHP Codesniffer, ...)

GITLAB CI

GITLAB

kunden_source

.gitlab-ci.yml
composer.json
docker-compose.yml

Shop DB

backup-databases.sh

smx_cookbook

- shoptifind.json
- smxresponsive.json
- smxaktionen.json
- smxlogger.json
- etc.

oxid_source

- EE_5.3.x
- EE_5.2.9
- EE_5.2.8
- EE_5.2.7
- etc.

ioly Core / ioly Installer

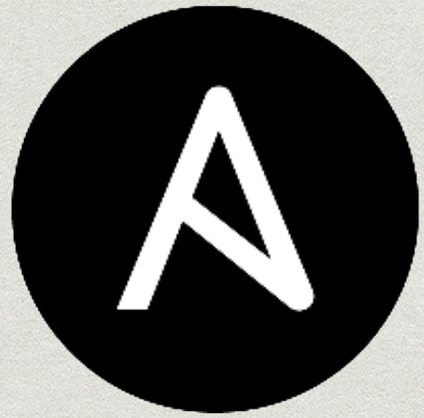
Docker Volume
(lokalesVerzeichnis)

Docker-Container (Apache, PHP, MySQL, PhpMyAdmin, ...)

IDE

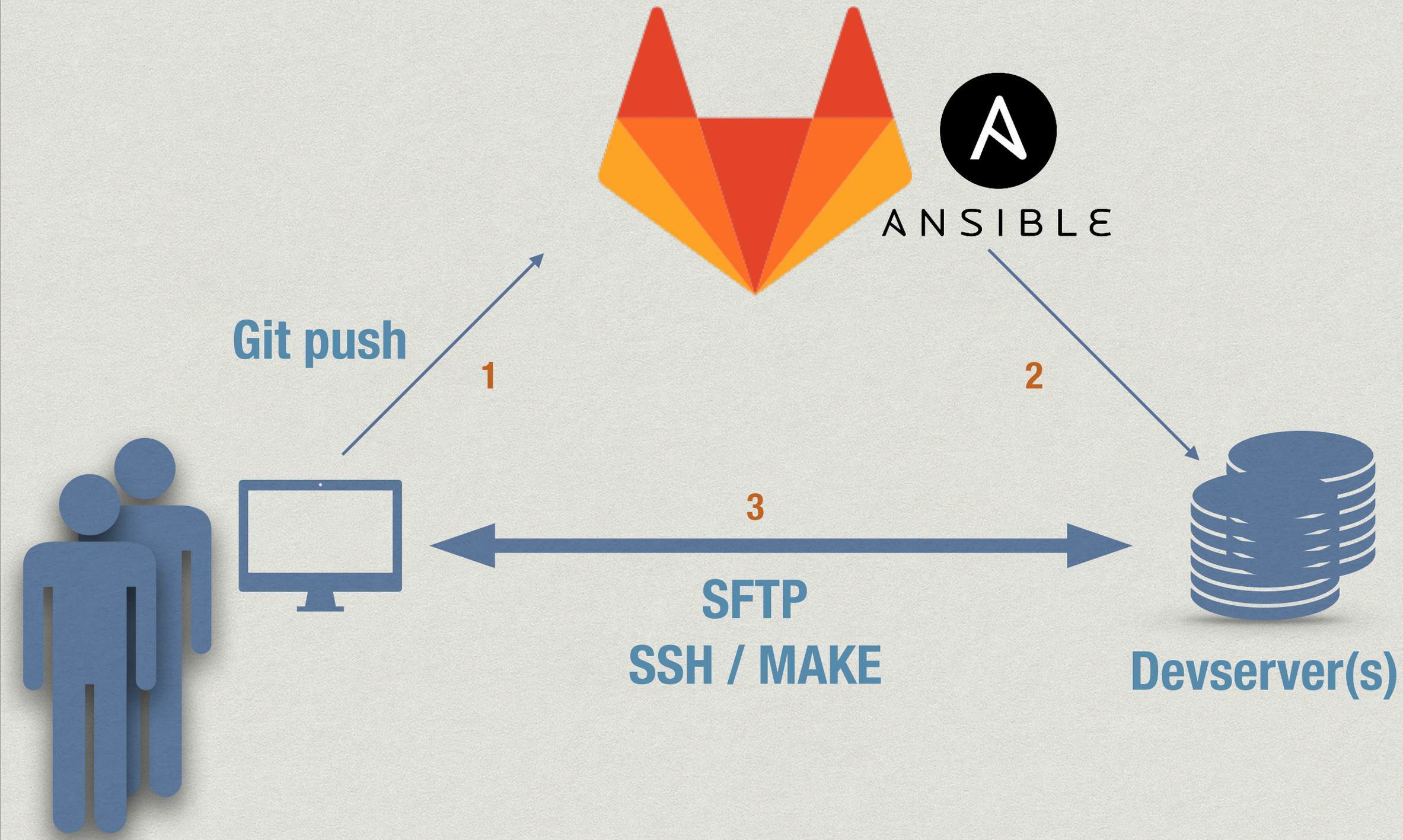
Enter „Auto-provisioned personal dev environment“ (APE)

- Idea: a simple „Git push“ into a certain branch (or with a specific flag set) automatically creates a **user-specific** dev environment on a shared (remote) server, using:
- Apache wildcard ServerAliases („project.user.devserver.de")
- Gitlab CI Deployment
- Ansible Playbook



ANSIBLE

APE Workflow



APE - Devserver Preparation

- Ansible playbook for the devserver setup …
- installs Apache
- configures wildcard ServerAliases
- installs phpbrew for multiple PHP versions
- installs MySQL-Server
- installs PHPMyAdmin / Adminer

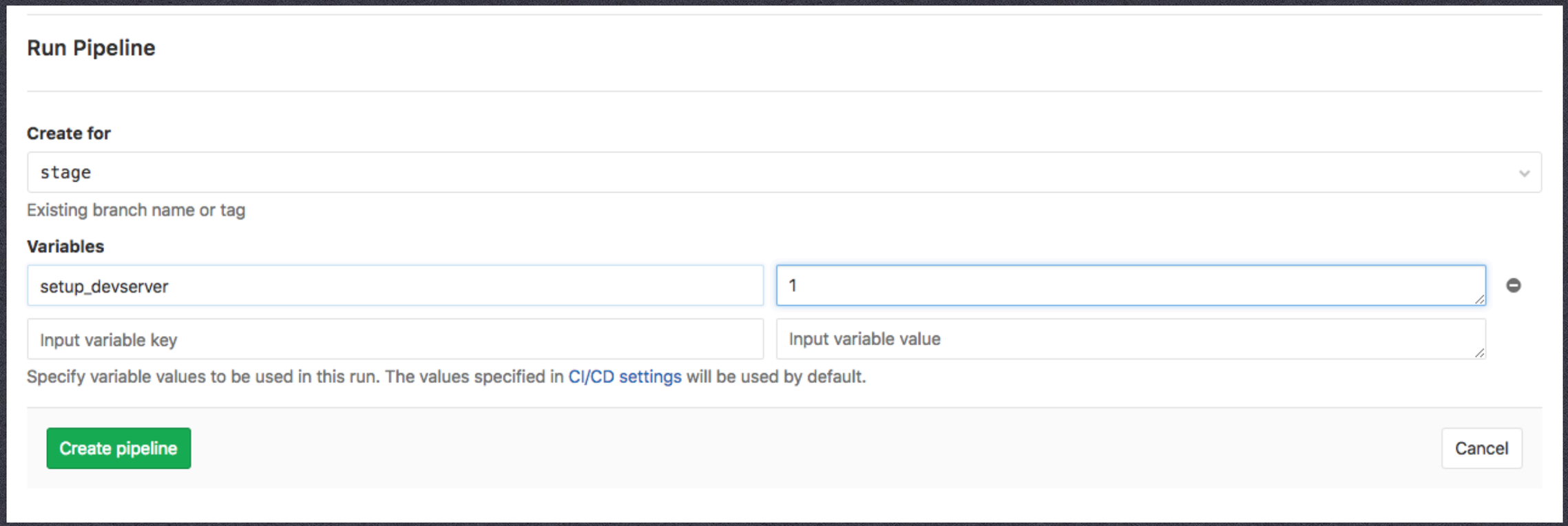
```yaml
---
# PROVISIONS A PHP DEVSERVER WITH WILDCARD APACHE SUBDOMAINS
- hosts: all
  vars_files:
    - vars/main.yml

  roles:
    - { role: ysz.phpbrew, become: true }
    - { role: geerlingguy.apache, become: true }
    - { role: geerlingguy.mysql, become: true }
    - { role: geerlingguy.phpmyadmin, become: true }
    - { role: JHeimbach.apache_envvars, become: true }
    - { role: idealista.tomcat-role, become: true }
    - { role: morbidick.semaphore, become: true }
    - { role: geerlingguy.nodejs, become: true}

  pre_tasks:
    - name: Update apt cache if needed.
      apt: update_cache=yes cache_valid_time=3600
  tasks:
    # INSTALL PACKAGES
    - name: Install the package "jq" to parse JSON responses
      apt:
        name: jq
```

APE - Step 1: Gitlab CI

- User pushes into a **special GIT branch** „devserver“ (or uses a specific CI variable)
- Gitlab CI runs an *internal Docker container*, **builds a complete shop** (create Composer project, run composer update, install modules, …), clones Ansible repo and runs **Ansible Playbook** to configure the project for the user on the devserver



RUN A GITLAB PIPELINE

DEVSERVER SETUP

APE - Step 2: Ansible

- **creates a new shell user** on the deserver (with Gitlab Username)
- gets the users' **public key** via Gitlab API and copies it to the devserver in *„authorized_keys“*
- creates a custom **Mysql user, a Mysql DB** „username_projectname“and downloads and inserts a **DB dump** (e.g. from stage server)
- **deploys the Gitlab CI artifact** (the „shop") to the users' project folder on the devserver
- creates a custom config.inc.php **include snippet** from a „Jinja“ Template with the dynamic subdomain (e.g. „project-subshopname.username.devserver.de") and the DB access data
- for OXID EE, creates **symlinks** for all the subshops, pointing to the main shop dir
- does some further shop configuration etc. via JSON/YAML/JINJA files

APE - Step 3: Developer

- can **auto-magically login via SSH** on the devserver
- set up **SFTP sync** in the IDE to sync the local files with his project dir on the devserver
- can now change locally to any GIT branch and sync the files „on-save“ with the „project skeleton“ on the devserver :)
- can now work any time on the devserver as an alternative to the local Docker environment
- can re-push into the „devserver“ Branch anytime to build everything from scratch again
- has **additional MAKEFILE targets** to interact (via ssh) with the devserver, e.g. to clear tmp (using oxrun e.g.), create views, run grunt, activate modules, re-insert a new DB dump etc.

APE - Config

```
ServerAlias *.*.mydevmachine.com
<Directory "/var/www/vhosts/projects">
  AllowOverride All
  Options -Indexes +FollowSymLinks
  Require all granted
</Directory>
VirtualDocumentRoot "/var/www/vhosts/projects/%2/%1/source"
```

```
apache_remove_default_vhost: true
apache_mods_enabled:
  - rewrite.load
  - actions.load
  - cgi.load
  - vhost_alias.load
apache_vhosts:
```

```
<?php
// created from config-inc.j2
$this->dbName = "{{ ci_user_name }}_{{ ci_project_name }}";
$this->dbUser = "{{ ci_user_name }}";
$this->dbPwd = "{{ ci_user_name }}";
$this->sShopURL = "http://{{ ci_project_name }}.{{
ci_user_name }}.{{ devserver_basedomain }}/";
```

```
# special task to setup an internal devserver -
# installs projects in user-specific directories, inserts
# db-dumps, creates users, adds public keys etc.
---
- hosts: '{{ target }}'
  remote_user: "
{{ remote_deployment_user | default('root') }}"
  # we need sudo here
  become: yes
  tasks:
    - include_vars: tasks/vars.yml
    - include: tasks/local_prepare.yml
      tags:
        - local_prepare
    - include: tasks/07-devserver/add_user.yml
      tags:
        - devserver
    - include: tasks/07-devserver/create_database.yml
      tags:
        - devserver
    - include: tasks/07-devserver/configure_shop.yml
      tags:
        - devserver
```

Gitlab CI

SMX-534 another devserver make target

4 jobs from `devserver` in 11 minutes 29 seconds (queued for 1 second)

`628c440a` ...

Pipeline Jobs 4

Build	Deploy	Prepare	Stage
prepare_shop	deploy_devserv...	prepare_devse...	activate_devse...

APE - Pros

- **truly identical environment** for every developer
- **fully automated project setup** with a simple „git push“
- PMs, other devs etc. can **test/view different developers feature branches** etc. on the devserver
- **devserver itself is setup automatically via Ansible**, easy updates, reproducible setup, multiple devserver (VMs) with identical setups easily possible
- no **performance** problems on local machine
- **no personal Docker support** needed for devs

APE - Todos, Cons?

- **xdebug** doesn't like symlinks, paths may have to be adjusted from time to time
- **scaling** - devserver may have problems with lots of developers and projects running simultaneously (=> but we can orchestrate more devservers via Ansible)

„Thoughts / Questions?“

–Discussion

Links

- http://www.shoptimax.de/blog/allgemein/zierfisch-oder-wal-lokale-software-entwicklung-mit-docker
- https://www.shoptimax.de/wie-wir-arbeiten/entwicklungsworkflow
- https://github.com/teracyhq/dev
- https://www.borncity.com/blog/2017/04/20/native-linux-container-fr-windows-server/
- https://www.heise.de/newsticker/meldung/Linux-Container-bald-nativ-unter-Windows-3689608.html
- https://blog.docker.com/2017/05/user-guided-caching-in-docker-for-mac/
- https://github.com/nuncanada/vagrant-rsync-only-changed
- https://github.com/jwilder/nginx-proxy
- https://www.ansible.com
- http://jinja.pocoo.org/
- https://about.gitlab.com/