

Images mit OBS und KIWI

GUUG-Frühjahrsfachgespräch 2019

11. April 2019

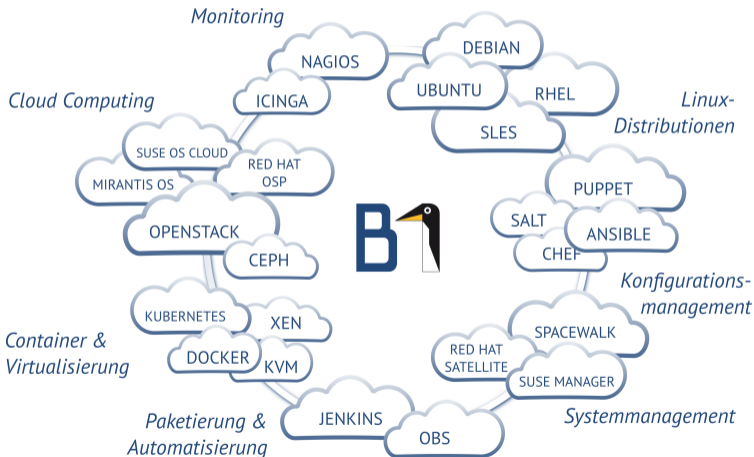


Ralf Lang
Linux Consultant & Developer
B1 Systems GmbH
lang@b1-systems.de

Vorstellung B1 Systems

- gegründet 2004
- primär Linux/Open Source-Themen
- national & international tätig
- über 100 Mitarbeiter
- unabhängig von Soft- und Hardware-Herstellern
- Leistungsangebot:
 - Beratung & Consulting
 - Support
 - Entwicklung
 - Training
 - Betrieb
 - Lösungen
- Standorte in Rockolding, Köln, Berlin & Dresden

Schwerpunkte



Images mit OBS und KIWI

Die Anforderungen . . .

- frischer Desktop für neuen Mitarbeiter oder selber von Null aus starten
- der Desktop soll ein Entwicklersetup zum Rumschrauben/-testen inklusive aller benötigten Werkzeuge enthalten
- definierte Umgebung für Unit und Integration Tests benötigt
- minimaler Zeitaufwand für Updates und Maintenance

So war es früher ...

- eine oder mehrere VMs mit einem LAMP Stack aufsetzen
- Hilfsbibliotheken/-werkzeuge ausrollen via zypper und rpm
- benötigte Spezialsoftware installieren (git, rake, pear, composer, npm, ...)
- Software konfigurieren; ggf. Datenbank migrieren/initialisieren

Configuration Management?

- noch eine weitere Sprache zu lernen, weitere Infrastruktur aufzubauen
- funktioniert für Demo Setups
- stört manchmal bei Entwicklungssetups

VM Images teilen oder bauen mit Vagrant (und OBS)

- relativ lange Zyklen zwischen Commit und Test
- relativ lange Downloadzeit und intensive Ressourcennutzung
- Deploymentmechanismus für neuen Code benötigt

Docker

- Mit einem frischen Container binnen Sekunden loslegen, wenn man:
 - kleinen Runtime Overhead und kleine Downloads braucht
 - im eigenen UI Code verändern, speichern und testen will ohne Zwischenschritt für den Transport
- Was ist denn dann mit Updates?

OBS

- OBS baut das Container Image neu, sobald ein relevantes RPM Paket sich ändert
- OBS verarbeitet Updates in Git oder SVN Repositories über Source Services
- OBS unterstützt sowohl natives Docker Format als auch KIWI XML für die Definition des Containerinhalts

Ein Image definieren 1/4

IBM distributions

- IBM PowerKVM 3.1

Many distributions

- Appliance



Kiwi image builds

- KIWI image build (to be used for appliance and product builds with kiwi)

[Expert mode](#)






Abbildung: `https://build.opensuse.org/project/add_repository_from_default_list/isv:B1-Systems:Horde5:opensuse-appliance`

Dem Projekt das KIWI Images Target Repository hinzufügen

Ein Image definieren 2/4

Source Files

Show entries Search:

Filename	Size	Changed	Actions
_constraints	101 Bytes	about 2 months ago	
_icon	12.4 KB	8 months ago	
config.kiwi	3.52 KB	about 1 month ago	
config.sh	1.75 KB	about 1 month ago	
horde5-developer-Docker-Leap.changes	340 Bytes	about 1 month ago	
root.tar.bz2	99.9 MB	about 1 month ago	

Showing 1 to 6 of 6 entries

Previous Next

Abbildung: <https://build.opensuse.org/package/show/isv:B1-Systems:Horde5:opensuse-appliance/horde5-developer-Docker-Leap>

- Neues Paket via osc Kommandozeile oder GUI anlegen
- config.kiwi zum Projekt hinzufügen

Ein Image definieren 3/4

```
openSUSE Build Service > Projects > in B1-Systems:Horde5:opensuse-appliance > horde5-developer-Docker-Leap > config.kiwi
Overview  Reinstates  Reverts  Requests  Users  Advanced
File config.kiwi of Package horde5-developer-Docker-Leap
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE kiwi [
3 <!-- @author=mailto:leap@opensuse.org -->
4 <!-- @description=type:system -->
5 <!-- @contact=mailto:leap@opensuse.org -->
6 <!-- @contact=mailto:leap@opensuse.org -->
7 <!-- @description=type:system / Quality check container for horde components from git -->
8 </!-->
9 </!-->
10 <preferences>
11 <type image="docker" derived_from="sbs-repositories/opensuse42.3"
12 <containerconfig name="horde-developer" tag="latest">
13 <!-->
14 <!-->
15 <!-->
16 <!-->
17 <!-->
18 <!-->
19 <!-->
20 <!-->
21 <!-->
22 <!-->
23 <!-->
24 </type>
25 </preferences>
26 <!-->
27 <!-->
28 <!-->
29 <!-->
30 <!-->
31 <!-->
32 </kiwi>
```

Abbildung: https://build.opensuse.org/package/view_file/isv:B1-Systems:Horde5:opensuse-appliance/horde5-developer-Docker-Leap/config.kiwi?expand=1

- Container Definitionen wie Volumes, offene Ports, Metadaten hinzufügen
- RPM Pakete zum Container hinzufügen
- Der Inhalt von root.tar.gz wird automatisch zum Container Filesystem hinzugefügt.

Ein Image definieren 4/4

- OBS baut das Image automatisch und stellt es in einem Docker Registry zum Download bereit.
- mehr Automatisierung ist möglich, z.B. Hinzufügen eines Source Service, der automatisch nach jedem Commit ins SCM den Container neu baut

Das Image einsetzen 1/4

Laden des Image

```
# docker pull registry.opensuse.org/isv/b1-systems/horde5/  
opensuse-appliance/images/horde5-developer:latest
```

Re-taggen des Image für kürzeren Registry-Namen

```
# docker tag registry.opensuse.org/isv/b1-systems/horde5/  
opensuse-appliance/images/horde5-developer:latest \  
horde5-developer:latest
```


Das Image einsetzen 2/4

Container starten, Port 80 für localhost und öffentliche Schnittstellen freigeben; aus dem Browser verwenden:

```
# docker run -d -p 80:80 --name horde5 horde5-developer /bin/start
```

Bash Session im laufenden Container öffnen, um Horde CLI Werkzeuge zu verwenden:

```
# docker run -it --name horde5 horde5-developer /bin/bash
```

Beispiel: Unit Tests für eine Komponente

```
# cd /srv/git/horde/$componentName/  
/srv/git/horde/components/bin/horde-components qc
```

Das Image einsetzen 3/4

Code in den Container übernehmen & Testen während des Codens?

```
# docker run -it --name horde5 horde5-developer /bin/bash
# mkdir ~/horde
# docker cp horde5:/srv/git/horde/ ~/horde
# docker stop horde5; docker rm horde5; docker run \
  -it -v ~/horde:/srv/git/horde/ --name horde5 \
  horde5-developer /bin/bash
```

Diese Befehlsfolge:

- 1 startet den Container zuerst, um den Code ins Home zu kopieren
- 2 startet den Container erneut und mountet dabei die Kopie des Home-Verzeichnisses

Das Image einsetzen 4/4

Beliebige Entwicklerwerkzeuge in definierter Umgebung laufen lassen:

Für Versions-Upgrades:

```
# /srv/git/horde/components/bin/horde-components update \  
--new-api="2.0.0" --new-state=stable --new-version="2.0.0" \  
--new-apistate=stable
```

Für Changelog Entries:

```
# /srv/git/horde/components/bin/horde-components \  
changed "[xyz] Added Foo."
```

Für Snapshots:

```
# /srv/git/horde/components/bin/horde-components \  
snapshot --keep-version
```

Wie geht's weiter?

- zusätzliche `docker-compose` Definition, um die App zusammen mit dem Datenbank-Container zu starten
- Image splitten: „run-only“ Basis-Image und ein abgeleitetes Image mit allen Entwickler-Werkzeugen
- standardmäßiges Ready-to-run Szenario ausliefern, alternativ zum From-Scratch Szenario, das noch konfiguriert werden muss
- weniger Zeit mit Setups zubringen – mehr Zeit für's Coden!

Vielen Dank für Ihre Aufmerksamkeit!

Bei weiteren Fragen wenden Sie sich bitte an info@b1-systems.de oder +49 (0)8457 -
931096