30 seconds to Code

openSUSE Conference 2018

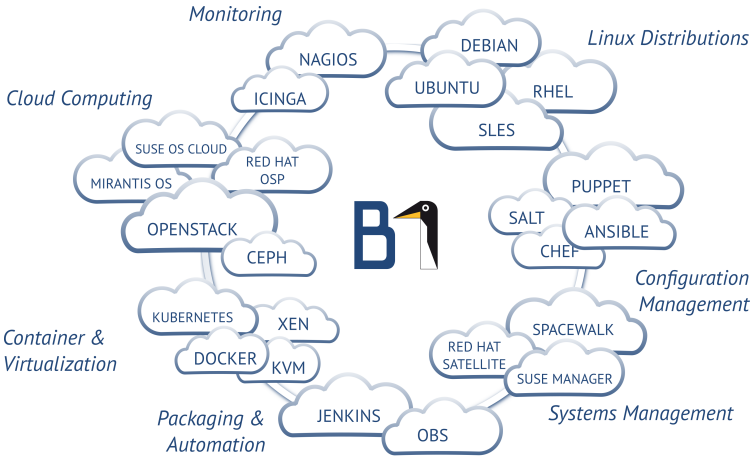
May 27, 2018

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

Introducing B1 Systems

- founded in 2004
- operating both nationally and internationally
- more than 100 employees
- vendor-independent (hardware and software)
- focus:
  - consulting
  - support
  - development
  - training
  - operations
  - solutions
- branch offices in Rockolding, Berlin, Cologne & Dresden

Areas of Expertise



30 seconds to Code

The problem

- you want to on-board a new member to your team or start from a fresh desktop
- the team member needs a developer setup for fiddling, including setup of all related tools
- you want a defined environment for running unit and integration tests
- you want to spend little time on updates & maintenance

How it used to be

- set up one or more virtual machines with a LAMP stack
- deploy any helper libraries and tools via `zypper`/`rpm`
- deploy the custom software (`git`, `rake`, `pear`, `composer`, `npm`, ...)
- configure the software, possibly run some migration/init for the DB

Tried Configuration Management

- yet another language, more infrastructure
- worked for Demo Setups
- sometimes got in the way of development setups

Sharing VM Images or building Images with Vagrant (or OBS)

- still relatively long cycles from commit to test
- quite some download time, resource usage
- need a deployment mechanism for new code

Enter Docker

- start over with a fresh container within seconds, if you need to
- little runtime overhead, very small download size
- edit code in your UI, save, test, without any transport step
- But what about updates?

Enter OBS

- OBS can rebuild your container image whenever any relevant rpm package changes
- OBS can deal with updates in git or svn repositories via source services
- supports both the docker native format and the kiwi XML format for defining container image content

Docker Images from KIWI/OBS

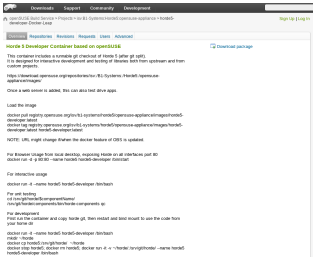


Figure: `https://build.opensuse.org/package/show/isv:B1-Systems:Horde5:opensuse-appliance/horde5-developer-Docker-Leap`

- this container includes a runnable git checkout of Horde 5
- designed for interactive development & testing of libraries both from upstream and custom projects

How to define an image 1/4

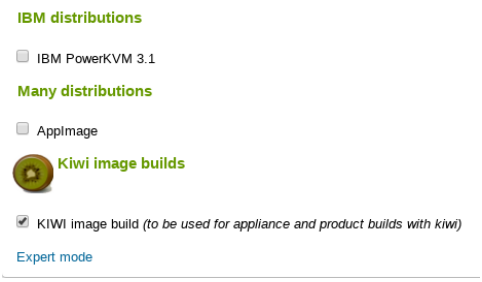


Figure: `https://build.opensuse.org/project/add_repository_from_default_list/isv:B1-Systems:Horde5:opensuse-appliance`

Add the Kiwi Images Target repository to your project.

How to define an image 2/4

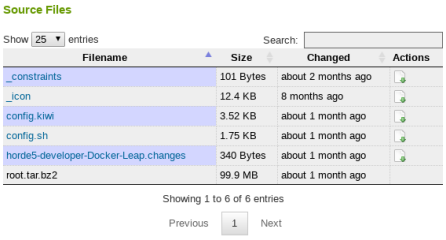


Figure: `https://build.opensuse.org/package/show/isv:B1-Systems:Horde5:opensuse-appliance/horde5-developer-Docker-Leap`

- start a new package via `osc` commandline or via GUI
- add a `config.kiwi` file to your project

How to define an image 3/4

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

- add container definitions like volumes, exposed ports, metadata
- add rpm packages to your container
- contents of `root.tar.gz` are automatically added to the container file system

How to define an image 4/4

- OBS will automatically build the image and put it into a docker registry for download.
- you can do more automation like adding a source service to automatically rebuild your container after every commit to the SCM
- also take note of Adrian's workshop on OBS to get a deeper explanation of all the options

How to use the image 1/4

Load the image

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

You might find the registry name a bit long, let's retag the image

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

How to use the image 2/4

Run the container, expose port 80 to public interfaces and localhost, use it from browser

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

Enter a bash session inside the running container to run horde cli tools

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

Example: Run unit tests for a component

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

How to use the image 3/4

Want to get the code into your IDE and test while you code?

```
# 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
```

- this runs the container first to copy the code to your home
- then restarts the container mounting your home copy inside

How to use the image 4/4

Run any development tools in your predefined environment.

For version 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
```

For Changelog entries

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

For snapshots

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

Where to go from here?

- add a `docker-compose` definition to start your app together with a database container
- split your image into a run-only base image and a derived image with all the developer tools
- deliver a default ready to run scenario and a "from scratch" scenario which still needs to be configured
- spend less time on setups, go write code

Thank You!

For more information, refer to info@b1-systems.de
or +49 (0)8457 - 931096