System Imaging with KIWI

Jan-Christoph Bornschlegel <jcborn@suse.de>

SUSE Linux Products GmbH – Build Service Team

20th May 2008
System Imaging with kiwi – Overview

1. Theory and History
   - Introduction
   - How does kiwi work?
   - The Configuration Directory
   - Invoking kiwi

2. A Real Life Example
   - Scenario
   - Solution

3. What’s next?
   - Autobuild System
   - Product Creation now and then

4. Questions and Answers
where are we

1. Theory and History
   - Introduction
   - How does kiwi work?
   - The Configuration Directory
   - Invoking kiwi

2. A Real Life Example
   - Scenario
   - Solution

3. What’s next?
   - Autobuild System
   - Product Creation now and then

4. Questions and Answers

Jan-Christoph Bornschlegel <jcborn@suse.de>

SUSE Linux Products GmbH – Build Service Team

System Imaging with KIWI
What \textit{kiwi} is and what it’s not

\textbf{KIWI is:}

- A command line based toolkit
- Usable as part of a process chain
- Usable as base tool for a high level application

\textbf{KIWI is not:}

- A product
kiwi history

- originated by **Marcus Schäfer**
- original purpose was creating “system on a stick”
- **James Willcox** (snorp) joins active development for Thin Client (SLETC)
- **Jigish Gohil** (CyberOrg) joins active development for LTSP project
- I join active development for Autobuild extension
Current project status

Used for the following products:

- SLEPOS – SuSE Linux Point of Sale
- SLETC – SuSE Linux Thin Client
- Hardware vendors for preload images
- JeOS

Community projects:

- Developers who deliver Live DVDs (KDE, openSUSE, …)
- Users who want to make images containing their application
where are we

1. Theory and History
   - Introduction
   - How does kiwi work?
   - The Configuration Directory
   - Invoking kiwi

2. A Real Life Example
   - Scenario
   - Solution

3. What's next?
   - Autobuild System
   - Product Creation now and then

4. Questions and Answers
Help Wanted!

Documentation is available throughout the web in various places

- [http://www.suse.de/~jcborn/kiwi-links.html](http://www.suse.de/~jcborn/kiwi-links.html)
- official documentation delivered with kiwi package: manpages and pdf
Setting up the buildhost

- Install kiwi, kiwi-tools and kiwi-desc-* packages
- create an image description file config.xml, or
- get and modify an existing image description

*caveat:* You must subscribe to the tools repository first!
How does kiwi work?

Buildsystem overview

Build host

Repo 1

Repo 2

Repo n

config files

image

Jan-Christoph Bornschlegel <jcborn@suse.de>

SUSE Linux Products GmbH – Build Service Team
How does kiwi work?

Buildsystem overview II

- Package repositories (local, network)
- Decent build host (esp. hdd)
- good network connection if using remote repositories
- Configuration file(s)

*caveat:* pick the correct `config.xml`
where are we

1. Theory and History
   - Introduction
   - How does kiwi work?
   - The Configuration Directory
   - Invoking kiwi

2. A Real Life Example
   - Scenario
   - Solution

3. What's next?
   - Autobuild System
   - Product Creation now and then

4. Questions and Answers
The Configuration Directory

Contents of the Configuration Directory

- **config.xml** contains every necessary image information (packages, repositories, settings, ...)
- **config.sh** customise the image after the packages are installed (end of `--prepare` stage)
- **image.sh** customise image at the beginning of the `--create` stage
- **root/** contains overlay files which are included in the image or needed in scripts
- **other** special YaST files and others
where are we

1. Theory and History
   - Introduction
   - How does kiwi work?
   - The Configuration Directory
   - Invoking kiwi

2. A Real Life Example
   - Scenario
   - Solution

3. What's next?
   - Autobuild System
   - Product Creation now and then

4. Questions and Answers
Invoking kiwi is basically divided in two steps

**prepare**  Creating a changeroot tree and install system into that

**create**  Create an image from that prepared tree

In case of USB image the deployment to the stick is third stage.
Invoking kiwi

Commands

The basic kiwi invocation looks like this

```
prepare kiwi -p <path-to-config.xml> -r <basedir>
create kiwi -c <basedir> -t <type> -d <imagedir>
deploy kiwi --bootstick <initrd>
    --bootstick-system <systemimage>
```
The base tree can be modified in some ways to shorten test time

- remove/install packages using chroot
- add/remove files
- modify configuration files
- add users, groups, ...

*Caveat*: risk of inconsistent system
where are we

1. Theory and History
   - Introduction
   - How does kiwi work?
   - The Configuration Directory
   - Invoking kiwi

2. A Real Life Example
   - Scenario
   - Solution

3. What’s next?
   - Autobuild System
   - Product Creation now and then

4. Questions and Answers
Requirements and Regressions

- You want your own openSUSE based distribution
- You want own packages from your own BuildService repo on it
- You may want to include “evil” packages
where are we

1. Theory and History
   - Introduction
   - How does kiwi work?
   - The Configuration Directory
   - Invoking kiwi

2. A Real Life Example
   - Scenario
   - Solution

3. What's next?
   - Autobuild System
   - Product Creation now and then

4. Questions and Answers
Solution

Get the “evil” packages built in your BS instance

- set your BS’ repository as source
- add the packages’ names to the `<packages>` section
- Build your image
Alternative Solution (wip)

- Get the “evil” packages built in your BS instance
- Build your own installation source and release your own installable media

This will soon be possible with the `kiwi-instsource` package (see later)
Repository Configuration Example

mounted DVD (base repo) → Build host → your own image

your home project
where are we

1. Theory and History
   - Introduction
   - How does kiwi work?
   - The Configuration Directory
   - Invoking kiwi
2. A Real Life Example
   - Scenario
   - Solution
3. What’s next?
   - Autobuild System
   - Product Creation now and then
4. Questions and Answers
Autobuild System

Purpose

- Autobuild is current internal package and media factory
- openSUSE BuildService will be the next generation package factory
- kiwi will be able to create installation sources (WIP)
How it works

Autobuild is a distributed system

- Build clients build single RPMs based on a central scheduler and source base
- every employee’s machine can (and should) be a build host
- scheduler collects built rpm files to internal “full trees” for each codebase and architecture
- metadata is created
where are we

1. Theory and History
   - Introduction
   - How does kiwi work?
   - The Configuration Directory
   - Invoking kiwi

2. A Real Life Example
   - Scenario
   - Solution

3. What’s next?
   - Autobuild System
   - Product Creation now and then

4. Questions and Answers

Jan-Christoph Bornschlegel <jcborn@suse.de>
SUSE Linux Products GmbH – Build Service Team
System Imaging with KIWI
Current Product Creation

- full trees for target architectures are sync’d to dedicated machines
- rpm files are selected and collected to one repository
- metadata for this particular repository is created
- finally all sorts of media are made:
  - ftp repositories
  - CD, DVD, torrent, ...
Collecting the target repository must be integrated into kiwi.

- Expansion of the config.xml syntax
- add module for repository creation
- allow priority value for repositories
- allow exceptions
- implement script hooks

Autobuild knowledge is necessary to create package lists and scripts
Generation of YaST metadata
- package description based on PDB
- patterns (through metapackage)
- contents, media, product files
- checksums
- root tree (through metapackage)

Creating the media itself uses \textit{m\_cd} atm.
State of project

- project is public as `kiwi-instsource`
- uses `instsourceutils`
- extended syntax for kiwi config file (still wip)
- first CD is in progress: `JeOS` – SLES based minimal system
Product Creation with *kiwi* cont.

Next steps planned:

- *LimeJeOS* installation CDs
- Code cleanup and performance enhancement
- Tests with released products
- Integration into BuildService for automatic instsource creation
Questions?

QUESTIONS...?
Yet another talk is over

Thank you for your attention!

See you on

- irc.freenode.net #opensuse-kiwi
- for kiwi issues: <kiwi-users@lists.berlios.de>
- for packaging issues: <opensuse-packaging@opensuse.org>
- Bugzilla for kiwi: product “opensuse.org”, component “system imaging”