# Introduction to the Pakfire Build Service

The new IPFire build system

Michael Tremer

**IPFire Project** 

April 28, 2012

What is the Pakfire Build Service? First things first...Terminology Conclusion

The Pakfire Build Service is the central build system of the IPFire firewall distribution.

What is the Pakfire Build Service? First things first...Terminology Conclusion

It automates the whole process from source code to a binary package that can be easily installed on any IPFire system.

In general, a package is a collection of files and their metadata.

In general, a package is a collection of files and their metadata.

It is a single file and hence can easily be sent around for further development or distribution. File Extension: pfm.

In general, a package is a collection of files and their metadata.

It is a single file and hence can easily be sent around for further development or distribution. File Extension: pfm.

There are two different types of packages...

In general, a package is a collection of files and their metadata.

It is a single file and hence can easily be sent around for further development or distribution. File Extension: pfm.

There are two different types of packages...

Source packages

In general, a package is a collection of files and their metadata.

It is a single file and hence can easily be sent around for further development or distribution. File Extension: pfm.

There are two different types of packages...

- Source packages
- Binary packages

What is the Pakfire Build Service? First things first...Terminology Conclusion

# Source packages

# Source packages

 A source package contains source code and build instructions to compile the sources.

#### Source packages

- A source package contains source code and build instructions to compile the sources.
- Source code is considered to be the release tarball of an upstream project and a number of patches.

#### Source packages

- A source package contains source code and build instructions to compile the sources.
- Source code is considered to be the release tarball of an upstream project and a number of patches.
- The result after putting a source package into a build is one or more binary packages.

# Binary packages

Binary packages are ready for installation on target systems.

It's easy:

pakfire install htop-1.0.1-1.ip3.x86\_64.pfm

# Binary packages: Example

audit - User space tools for 2.6 kernel auditing

audit-2.2-2.ip3.src.pfm

### Binary packages: Example

#### audit - User space tools for 2.6 kernel auditing

audit-2.2-2.ip3.src.pfm

#### The source package produces a whole set of binary packages:

- audit-2.2-2.ip3.x86 64.pfm
- audit-debuginfo-2.2-2.ip3.x86\_64.pfm
- audit-devel-2.2-2.ip3.x86\_64.pfm
- audit-libs-2.2-2.ip3.ip3.x86\_64.pfm

Packages have different kinds of *dependencies* which define relationships to each other.

Packages have different kinds of *dependencies* which define relationships to each other.

Packages have different kinds of *dependencies* which define relationships to each other.

A dependency can be defined as:

• An other package by name (openssl, beep or zlib-devel).

Packages have different kinds of *dependencies* which define relationships to each other.

- An other package by name (openssl, beep or zlib-devel).
- An other package by name and version (glibc >= 2.15 or udev < 182).</li>

Packages have different kinds of *dependencies* which define relationships to each other.

- An other package by name (openssl, beep or zlib-devel).
- An other package by name and version (glibc >= 2.15 or udev < 182).</li>
- A filename like /usr/bin/bash.

Packages have different kinds of *dependencies* which define relationships to each other.

- An other package by name (openssl, beep or zlib-devel).
- An other package by name and version (glibc >= 2.15 or udev < 182).</li>
- A filename like /usr/bin/bash.
- A string that is provided by any package (e.g. bootloader).
   You should not use this.

A given package X may have the following dependencies:

 Requires: The installation of package X requires the availability of everything in this list.

- Requires: The installation of package X requires the availability of everything in this list.
- Provides: The package in hand comes with the features in this list.

- Requires: The installation of package X requires the availability of everything in this list.
- Provides: The package in hand comes with the features in this list.
- Conflicts: Package X cannot be installed if any of these dependencies is provided by an other package.

- Requires: The installation of package X requires the availability of everything in this list.
- Provides: The package in hand comes with the features in this list.
- Conflicts: Package X cannot be installed if any of these dependencies is provided by an other package.
- Obsoletes: If package X is installed, there is no need for anything in this list.

- Install only what you need.
  - Keeps the system small and fast.

- Install only what you need.
  - Keeps the system small and fast.
- Better updates:
  - Update only what you want (e.g. only security updates).
  - Faster distribution of hotfixes.
  - Rollback bad updates.

- Install only what you need.
  - Keeps the system small and fast.
- Better updates:
  - Update only what you want (e.g. only security updates).
  - Faster distribution of hotfixes.
  - Rollback bad updates.
- Use other package sources.
  - Third party repositories for closed source stuff.
  - Packages we don't want to have in IPFire.
  - Whatever you might think of...

#### For developers:

Each developer gets a small chunk to maintain.

- · Each developer gets a small chunk to maintain.
- One broken package does not break the whole distribution.

- Each developer gets a small chunk to maintain.
- One broken package does not break the whole distribution.
- Faster builds:
  - Just rebuild one package.
  - Never the whole distribution.

- Each developer gets a small chunk to maintain.
- One broken package does not break the whole distribution.
- Faster builds:
  - Just rebuild one package.
  - Never the whole distribution.
- Send unstable updates to each other. Test them.

- Each developer gets a small chunk to maintain.
- One broken package does not break the whole distribution.
- Faster builds:
  - Just rebuild one package.
  - Never the whole distribution.
- Send unstable updates to each other. Test them.
- Push out updates earlier:
  - No release planning for core updates.
  - Don't bother with issues that have already been fixed.

#### Conclusion

Packages have a lot of advantages.

#### Conclusion

Packages have a lot of advantages.

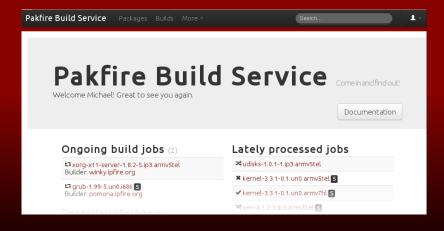
But the management of more than 1250 packages per architecture is a tough job.

#### Conclusion

Packages have a lot of advantages.

But the management of more than 1250 packages per architecture is a tough job.

We need a management tool for that!



 It has been developed by developers of the IPFire project to achieve a very high level of quality in the build process in conjunction with the community.

- It has been developed by developers of the IPFire project to achieve a very high level of quality in the build process in conjunction with the community.
- Written in pure Python.

- It has been developed by developers of the IPFire project to achieve a very high level of quality in the build process in conjunction with the community.
- Written in pure Python.
- Comes with a web user interface and an XMLRPC management interface.

- It has been developed by developers of the IPFire project to achieve a very high level of quality in the build process in conjunction with the community.
- Written in pure Python.
- Comes with a web user interface and an XMLRPC management interface.
- Heavily based on the Pakfire package management system.

Introduction
The Pakfire Build Service
Learning by example
The end

Excursion: pakfire More definitions A package A build

# Excursion: pakfire

The Pakfire package management system manages installations, updates, removes and more of packages on an IPFire system.

Written in Python with performance critical code in C.

- Written in Python with performance critical code in C.
- Fast and most robust dependency solving algorithm in the world.

- Written in Python with performance critical code in C.
- Fast and most robust dependency solving algorithm in the world.
- Very tiny. Runs on embedded systems.

- Written in Python with performance critical code in C.
- Fast and most robust dependency solving algorithm in the world.
- Very tiny. Runs on embedded systems.
- Built-in build system.

# Package

Primarily, the term *package* describes a source package. It also can rarely mean the actual package file.

# Package

Primarily, the term *package* describes a source package. It also can rarely mean the actual package file.

#### Build

A build is an ordering unit which contains a source package and multiple *build jobs*. There is also some meta information about the reason of the build (commit message, associated bugs). A build is what we are managing here. We don't bother about a

A build is what we are managing here. We don't bother about a single package file.

### Package

Primarily, the term *package* describes a source package. It also can rarely mean the actual package file.

#### Build

A build is an ordering unit which contains a source package and multiple *build jobs*. There is also some meta information about the reason of the build (commit message, associated bugs). A build is what we are managing here. We don't bother about a single package file.

# Build job or just: job

A build contains several jobs. One for each architecture the build is built for.

### Package

Primarily, the term *package* describes a source package. It also can rarely mean the actual package file.

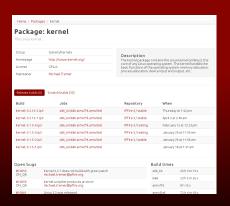
#### Build

A build is an ordering unit which contains a source package and multiple *build jobs*. There is also some meta information about the reason of the build (commit message, associated bugs). A build is what we are managing here. We don't bother about a single package file.

# Build job or just: job

A build contains several jobs. One for each architecture the build is built for.

# PBS by example: A package



This is the kernel package.

The page contains some general information about the package and in the row below a list of all known builds.

At the bottom: Open bugs, some statistics.

# PBS by example: A build

A build can be created in two ways and thus they distinguish themselves in two categories.

# PBS by example: A build

A build can be created in two ways and thus they distinguish themselves in two categories.

Release builds.

# PBS by example: A build

A build can be created in two ways and thus they distinguish themselves in two categories.

- Release builds.
- Scratch builds.

A release build is created out of a commit in a git repository.

A release build is created out of a commit in a git repository.

It is automatically created from source.

A release build is created out of a commit in a git repository.

- It is automatically created from source.
- Always linked to a distribution.

A release build is created out of a commit in a git repository.

- It is automatically created from source.
- Always linked to a distribution.
- Goes its way through all repositories of the distribution (i.e. from testing to stable).

In contrast to the release builds, a scratch build is created by an individual.

Command: pakfire-client build beep/beep.nm

In contrast to the release builds, a scratch build is created by an individual.

Command: pakfire-client build beep/beep.nm

Used to test if a packages does build on all architectures.

In contrast to the release builds, a scratch build is created by an individual.

Command: pakfire-client build beep/beep.nm

- Used to test if a packages does build on all architectures.
- Used to distribute experimental packages.

In contrast to the release builds, a scratch build is created by an individual.

Command: pakfire-client build beep/beep.nm

- Used to test if a packages does build on all architectures.
- Used to distribute experimental packages.

We will learn what exciting things we can do with them in a minute...

Introduction
The Pakfire Build Service
Learning by example
The end

Step by step A package A build A job

# Step by step

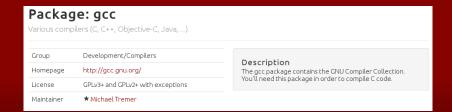
Going step by step through the web user interface of the Pakfire Build Service, you will get an impression about what it can do...

# A package

Home / Packages / gcc Package: gcc Development/Compilers Group Description Homepage http://gcc.gnu.org/ The gcc package contains the GNU Compiler Collection. You'll need this package in order to compile C code. License GPLv3+ and GPLv2+ with exceptions ★ Michael Tremer Maintainer Release builds (2) Build Jobs Repository When acc-4.6.3-6.ip3 x86\_64 i686 armv7hl armv5tel IPFire 3 / testing 14 hours ago x86\_64 i686 armv7hl armv5tel IPFire 3 / stable gcc-4.6.3-5.ip3 Saturday at 6:57 pm

Step by step A package A build A job A package file

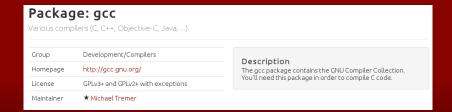
# A package



At the top of the page, you will find the most important metadata of the package

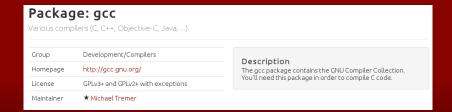
Step by step A package A build A job A package file

# A package



At the top of the page, you will find the most important metadata of the package which is: name and a short sentence what the package does (headline)

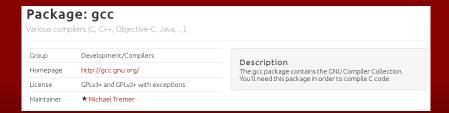
# A package



At the top of the page, you will find the most important metadata of the package which is: name and a short sentence what the package does (headline), a longer description

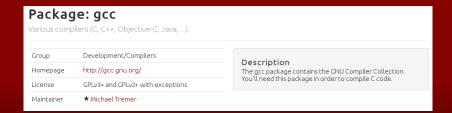
Step by step A package A build A job A package file

# A package



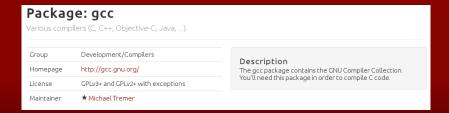
At the top of the page, you will find the most important metadata of the package which is: name and a short sentence what the package does (headline), a longer description, the home of the package (we call this upstream)

# A package



At the top of the page, you will find the most important metadata of the package which is: name and a short sentence what the package does (headline), a longer description, the home of the package (we call this upstream), license

## A package



At the top of the page, you will find the most important metadata of the package which is: name and a short sentence what the package does (headline), a longer description, the home of the package (we call this upstream), license, and the maintainer within the IPFire project.

## A package

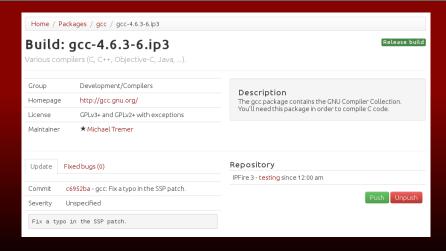


At the bottom you will find a list of all builds for this package.

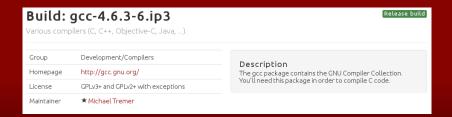
That are two release builds for this example, one in testing, the other already in stable state.

Let's have a closer look at them...

#### A build



#### A build

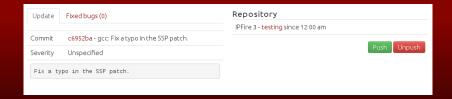


The same metadata as we already know from the package page.

Additionally tagged as a release build or scratch build in the top right corner.

#### A build

Further down, there is a block that shows us more information about the reason for this build and the current state.



#### A build

Further down, there is a block that shows us more information about the reason for this build and the current state.



This build for example was created by commit *c6952ba* and is in the testing repository.

#### A build

At the bottom of a package is a short summary of the build jobs. You get an overview about how the build of the package goes.

Build jobs Properties					
Arch	State	Host	Duration		
x86_64	Finished	alecto.ipfire.org	1 h 29 m 42 s		
i686	Finished	pomona.ipfire.org	2 h 48 m 41 s		
armv7hl	Finished	winky.ipfire.org	6 h 36 m 37 s		
armv5tel	Dispatching	dobby.ipfire.org	15 s		

#### A build

By clicking on the *Properties* tab, you can see a couple of more meta information about this build.



Step by step A package A build A job A package file

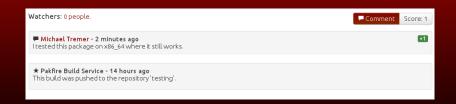
#### A build: Comments & score

One major design goal of the PBS was the involvement of the community of testers into the development process.

#### A build: Comments & score

One major design goal of the PBS was the involvement of the community of testers into the development process.

We achieved that by giving our users the opportunity to comment on builds and rate them like in this example:



Step by step
A package
A build
A job
A package file

A package
A build
A job
A package f

### We gain the following advantages:

 Users get go and grab a newer package if they are experiencing an issue with the current version.

- Users get go and grab a newer package if they are experiencing an issue with the current version.
- If the package fixes their bug, they only need to type a few lines into the comment box and rate the build.

- Users get go and grab a newer package if they are experiencing an issue with the current version.
- If the package fixes their bug, they only need to type a few lines into the comment box and rate the build.
- The developer gets immediate feedback about what he has created. This has been difficult in the past.

- Users get go and grab a newer package if they are experiencing an issue with the current version.
- If the package fixes their bug, they only need to type a few lines into the comment box and rate the build.
- The developer gets immediate feedback about what he has created. This has been difficult in the past.
- If things don't go that well, it's the same game.

### We gain the following advantages:

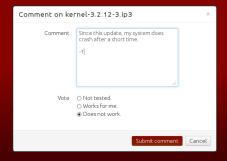
- Users get go and grab a newer package if they are experiencing an issue with the current version.
- If the package fixes their bug, they only need to type a few lines into the comment box and rate the build.
- The developer gets immediate feedback about what he has created. This has been difficult in the past.
- If things don't go that well, it's the same game.

At the end of the day, it is much easier for the developer to decide if a package is broken or if the update is working well and worth being *pushed*.

Who is allowed to comment and vote?

Every registered user is allowed to comment builds.

At the current point in time, it is necessary to be enabled by an admin to vote to assure a certain quality.



## A job



On a job page, you will find a lot of information about the state and progress of a build job for a certain architecture.

NOTE: The interface may still change a bit.

You have access to the build log files which contain the whole build process and are very useful for debugging.



They are easy to download or can be directly viewed in the web browser.

Step by step
A package
A build
A job
A package file

Another very important feature is the so called buildroot.

Step by step A package A build A job A package file

Another very important feature is the so called buildroot.

If there is a package with a serious bug which affected other packages that have been built with that package, we need to know which packages are infected.

Step by step A package A build A job A package file

Another very important feature is the so called buildroot.

If there is a package with a serious bug which affected other packages that have been built with that package, we need to know which packages are infected.

The buildroot gives us a list of all packages that were extracted to the buildroot at this build job.

Another very important feature is the so called buildroot.

If there is a package with a serious bug which affected other packages that have been built with that package, we need to know which packages are infected.

The buildroot gives us a list of all packages that were extracted to the buildroot at this build job.

Another application: If a build fails, we are able to create the exactly same buildroot and reproduce the complete build process.

A job

## Job buildroot: gcc-4.6.3-6.ip3.x86 64

This is the buildroot of build job gcc-4.6.3-6.ip3.x86 64.

The packages listed below were used for the build.

- audit-libs-2.2-2.ip3.x86 64
- autogen-5.14-1.ip3.x86 64
- autogen-libs-5.14-1.ip3.x86 64
- bash-4.2-9.ip3.x86 64
- beecrypt-4.2.1-2.ip3.x86\_64
- binutils-2.22-3.ip3.x86 64
- bzip2-1.0.6-6.ip3.x86 64
- ca-certificates-2012.81-1.ip3.noarch
- ccache-3.1.7-1.ip3.x86 64
- cloog-ppl-0.15.11-1.ip3.x86 64
- cloog-ppl-devel-0.15.11-1.ip3.x86 64
- coreutils-8.16-3.ip3.x86 64
- cpio-2.11-2.ip3.x86 64
- cpp-4.6.3-5.ip3.x86 64
- cracklib-2.8.18-2.ip3.x86\_64
- cracklib-dicts-2.8.18-2.ip3.x86\_64
- cvrus-sasl-libs-2.1.25-2.ip3.x86 64
- db4-1:4.8.30-5.ip3.x86 64

A package
A build
A job
A package file

# A build job also owns the package files that came out of the build process.

Package files (13)						
Name	Version	Arch	Size			
срр	4.6.3-6.ip3	×86_64	4M	<b>●</b> Download		
gcc	4.6.3-6.ip3	×86_64	7M	<b>●</b> Download		
gcc-c++	4.6.3-6.ip3	×86_64	5M	<b>⊕</b> Download		
gcc-debuginfo	4.6.3-6.ip3	×86_64	27M	<b>●</b> Download		
gcc-plugin-devel	4.6.3-6.ip3	×86_64	560k	<b>⊕</b> Download		
libgcc	4.6.3-6.ip3	×86_64	120k	<b>⊕</b> Download		
libgomp	4.6.3-6.ip3	×86_64	40k	<b>⊕</b> Download		
libmudflap	4.6.3-6.ip3	×86_64	80k	<b>⊕</b> Download		
libmudflap-devel	4.6.3-6.ip3	×86_64	20k	<b>●</b> Download		
libquadmath	4.6.3-6.ip3	×86_64	130k	① Download		

Step by step A package A build A job A package file

Let's click on one of these...

## A package file

Again, we have a lot of meta information which explains itself.



A package file

You may view the dependencies of a package online...

## Dependencies

#### Provides

#### Requires

- libgcc s.so.1()(64bit)
- libacc s.so.1(GCC 3.0)(64bit)
- libgcc s.so.1(GCC 3.3)(64bit)
- libgcc s.so.1(GCC 3.3.1)(64bit)
- libacc s.so.1(GCC 3.4)(64bit) libgcc s.so.1(GCC 3.4.2)(64bit)
- libgcc\_s.so.1(GCC\_3.4.4)(64bit)
- libgcc s.so.1(GCC 4.0.0)(64bit)
- libacc s.so.1(GCC 4.2.0)(64bit)
- libgcc s.so.1(GCC 4.3.0)(64bit)
- libgcc s.so.1(libgcc s.so.1)(64bit)

...as well as the list of files in the package (with permissions and user) which is both very useful for debugging.



In conclusion, the Pakfire Build Service has increased our productivity enormously, because it is so much faster in bringing out package updates.

In conclusion, the Pakfire Build Service has increased our productivity enormously, because it is so much faster in bringing out package updates.

Bugs get fixed very fast and do not stop you working on something.

In conclusion, the Pakfire Build Service has increased our productivity enormously, because it is so much faster in bringing out package updates.

Bugs get fixed very fast and do not stop you working on something.

Developers are able to test each others changes very easily and provide feedback at a central place. So are users.

In conclusion, the Pakfire Build Service has increased our productivity enormously, because it is so much faster in bringing out package updates.

Bugs get fixed very fast and do not stop you working on something.

Developers are able to test each others changes very easily and provide feedback at a central place. So are users.

The integration with the bugtracker (Bugzilla) closes bugs and makes sure that the reporter of a bug is informed about a fix and encouraged to test it.

Altogether, we are very proud of what we have done.

Altogether, we are very proud of what we have done.

The Pakfire Build Service makes development fun. It helps to achieve higher goals.

Altogether, we are very proud of what we have done.

The Pakfire Build Service makes development fun. It helps to achieve higher goals.

We don't consider the service fully done, because we have so many ideas on our minds...

# THE END

Now go out and show what you have learned today at https://pakfire.ipfire.org