openSUSE.org Build Service

Maintain One Source for all Linux Platforms

Adrian Schröter
adrian@suse.de

Klaas Freitag
freitag@suse.de

Challenges

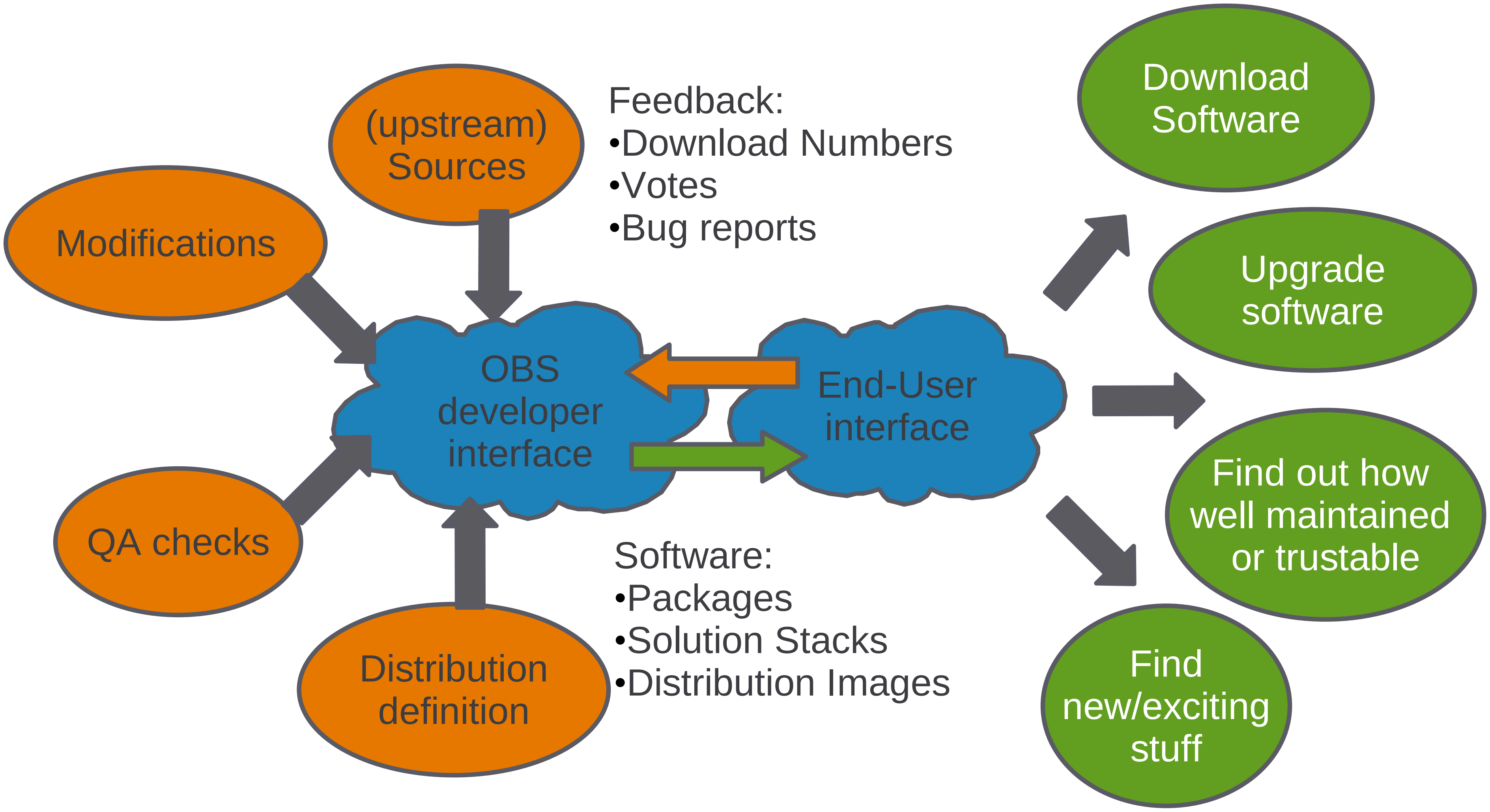
- Open source communities provide lots of source code, but building & installation is often hard for unexperienced users.
- User challenges:
  - Find additional software
  - Judge about the quality and trust of provided software
  - Find developer to give feedback
- Developer challenges:
  - Maintain sources for different target platforms
  - Maintain patches during upstream updates
  - Integrate contributions
  - No version updates for released distributions
  - Deal with differences on various distributions

Goals of the Build Service

- Make it simple to provide binary packages of software
- Support the “Release early, Release often” approach
- Involve and connect the open source communities
- Make it easy and secure to install new software

- **Open the openSUSE distribution development**

What Does it Do ?



Feedback:
- Download Numbers
- Votes
- Bug reports

Software:
- Packages
- Solution Stacks
- Distribution Images

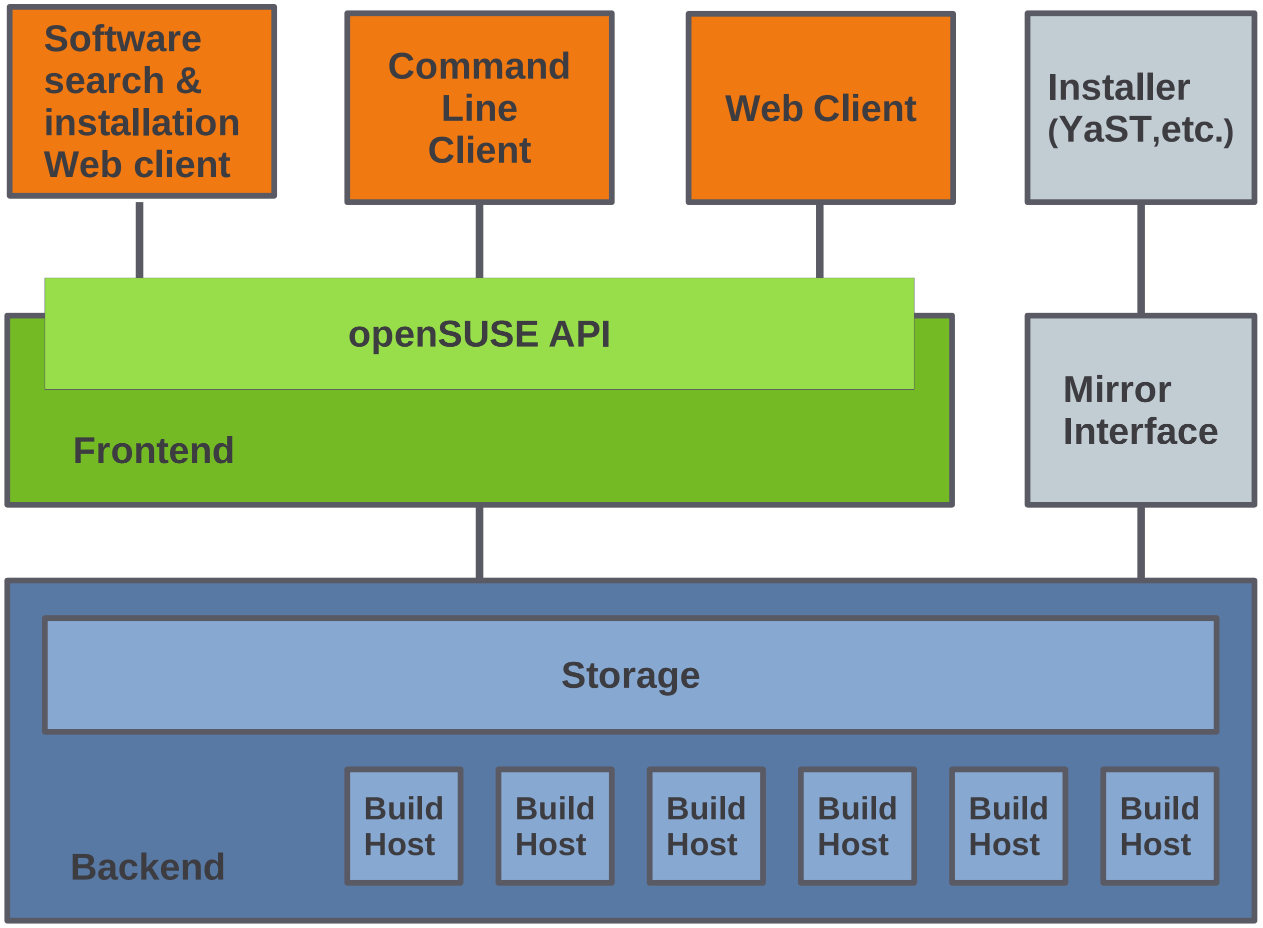
What is the Build Service ?

- Infrastructure
  - Software search interface
  - Build systems to create packages
  - Download and mirror infrastructure for packages
  - Collaboration framework
- Tools
  - Tools are used for local operations on the workstation or for remote operations on the Build Service server.

The Open Design of the Build Service

- Everyone is able to use the Build Service.
- The Build Service is 100% free software (GPL).
- The Build Service provides a public API.
- Multiple Build Service instances can get connected
- The Build Service can be integrated into existing tools.
- The Build Service is not limited to openSUSE.
- Integration with existing web pages is possible.

Components Overview

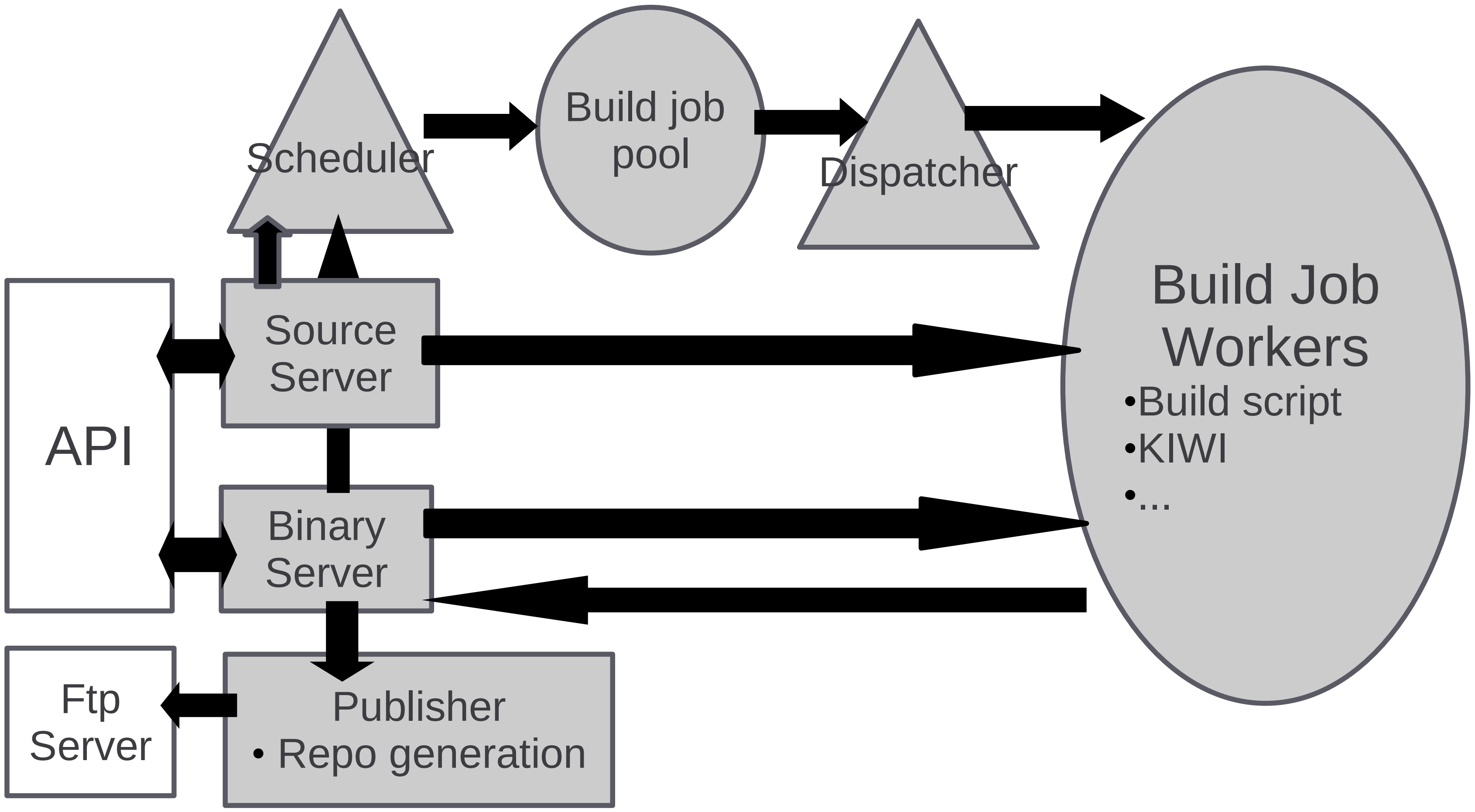


Backend

- Building Packages

- Storage for sources (version controlled)
- Farm of build hosts for building packages
- Run build in specified environment
- Build for multiple hardware architectures (currently i586, x86_64)
- Storage for built packages
- Provide build status and logs

Build Process Implementation



Software Search and Installation

- Software can be
  - Package (deb or rpm)
  - Solution Stack (aka patterns)
  - Images (for example Lime JeOS, Live CD, Installation DVD)
  - More can be supported in future
- http://software.opensuse.org/search/
- “1-Click” Installation on openSUSE

Client Tools

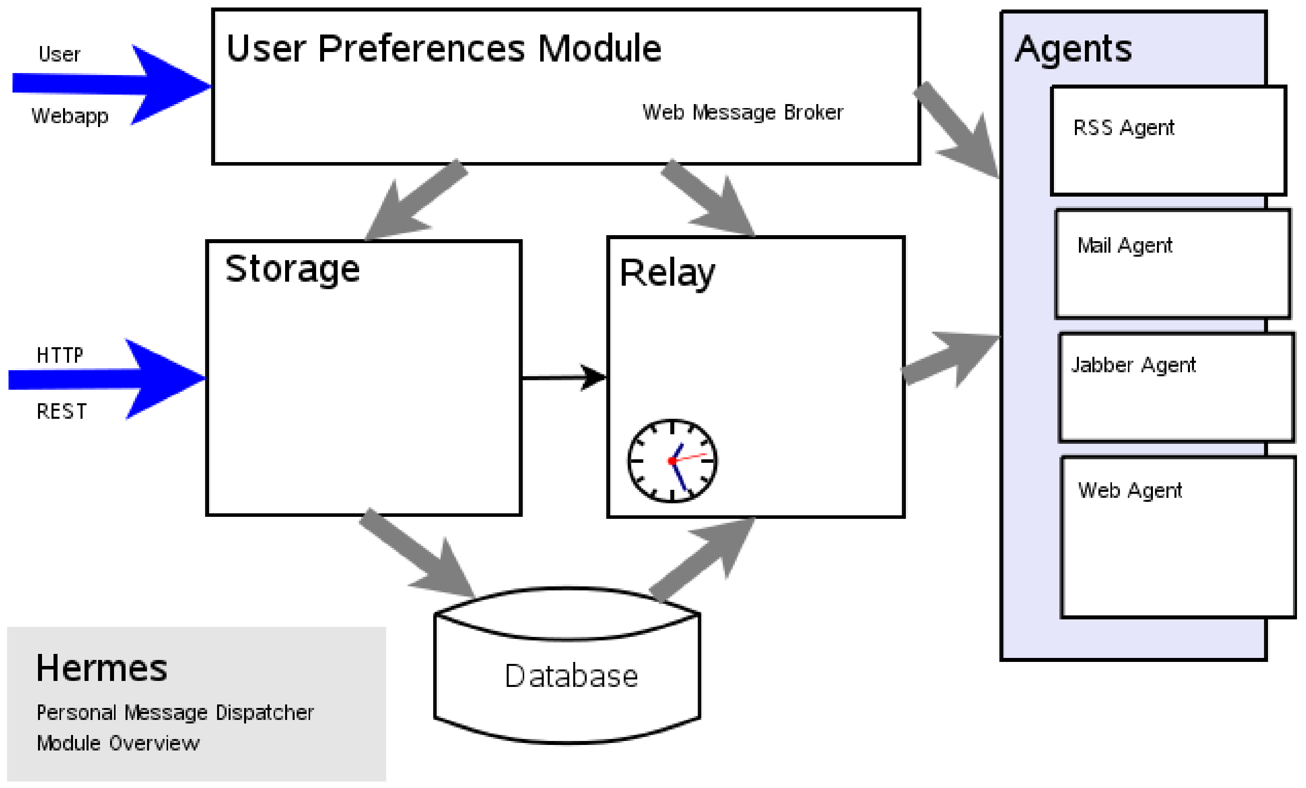
- User Interface for Developers and Packagers

- Web Client
  - Easy browsing and project administration
  - Editing and uploading of sources
  - Downloading of built packages

- Command Line Client
  - Editing and uploading of sources
  - Start local build for debugging

Notifications (WIP)

- You decide which messages when and how!



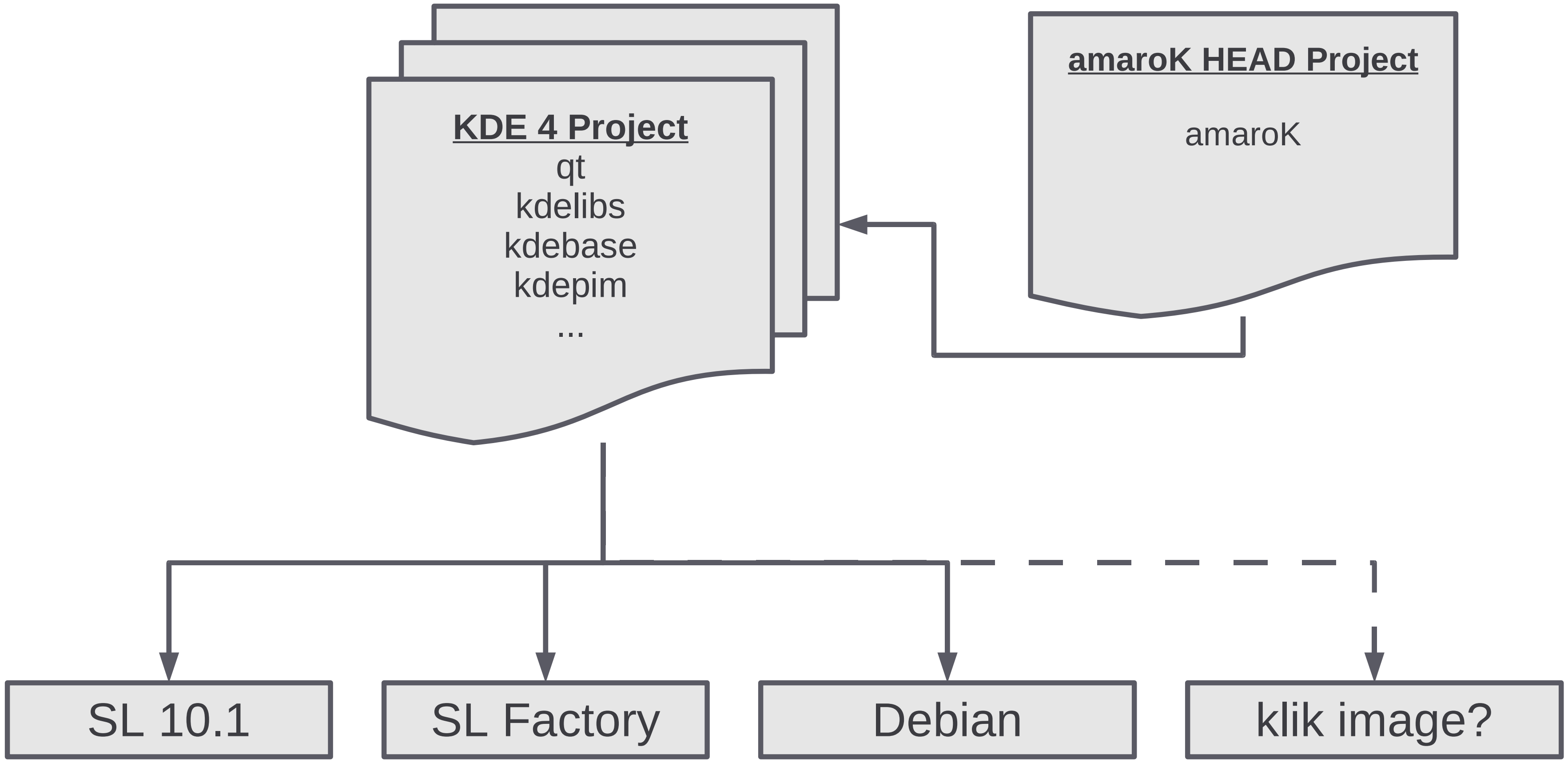
Project Model

Project Model 1/2

- A project is a workspace which can be created by any user. It may contain:
  - A list of users with write access to it
  - Sources or a description how to download them
  - Link to existing sources to be built in a different enviroment
  - Changes for existing packages
  - A list of build targets to build binary packages for
- The result will be one or more package repositories.

Project Model 2/2

•



Trust Model

- The Build Service does guarantee that the binary package got build from the sources, but it can't judge about the sources itself.
- Everyone can submit source, this causes a potential security problem.
- The decision to trust a package or not is up to the end-user.
- The trust level of a project depends on the trust level of its contributors.

This is currently researched by Marko Jung <mjung@suse.de>

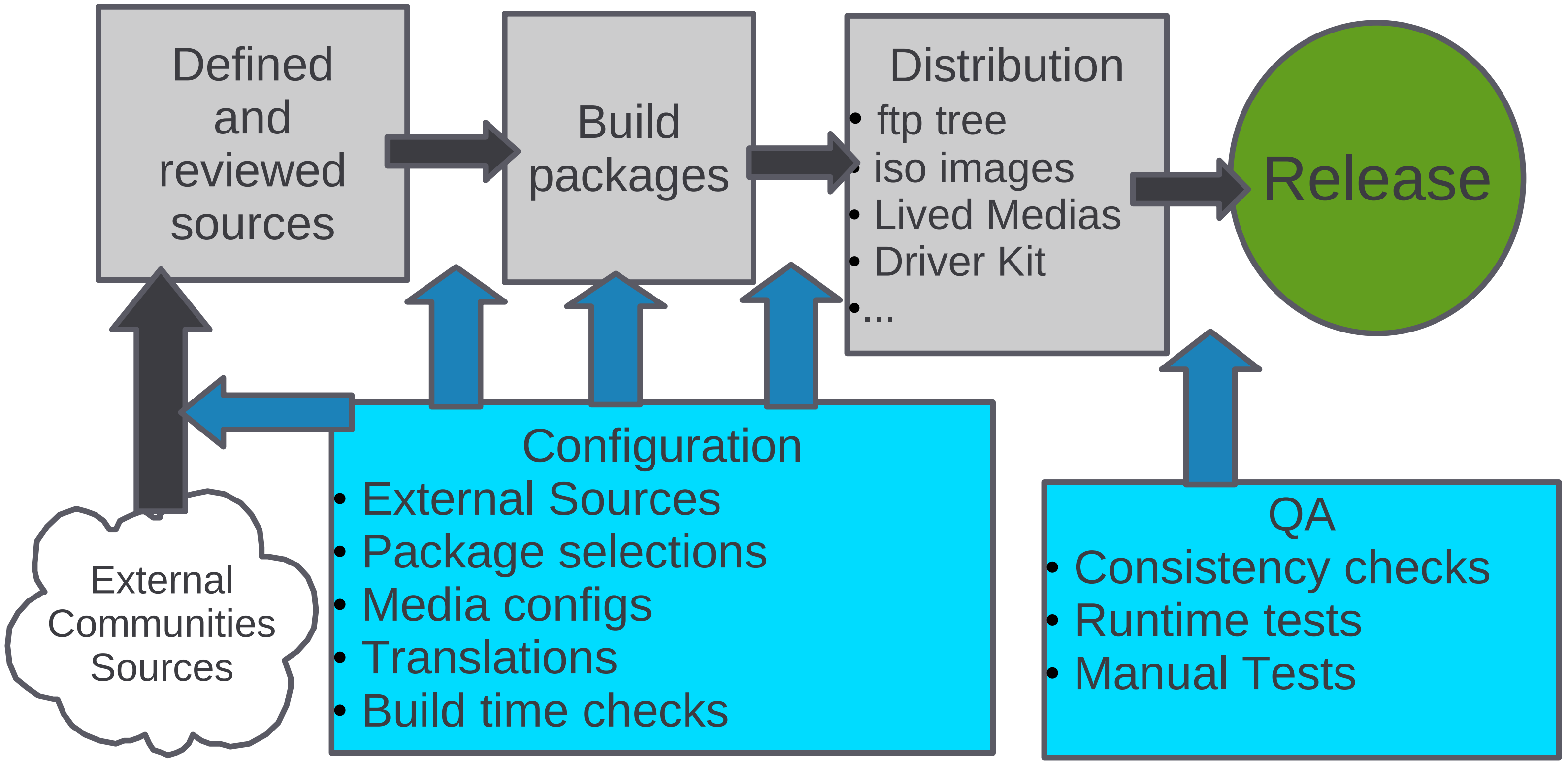
Collaboration

Collaboration Features

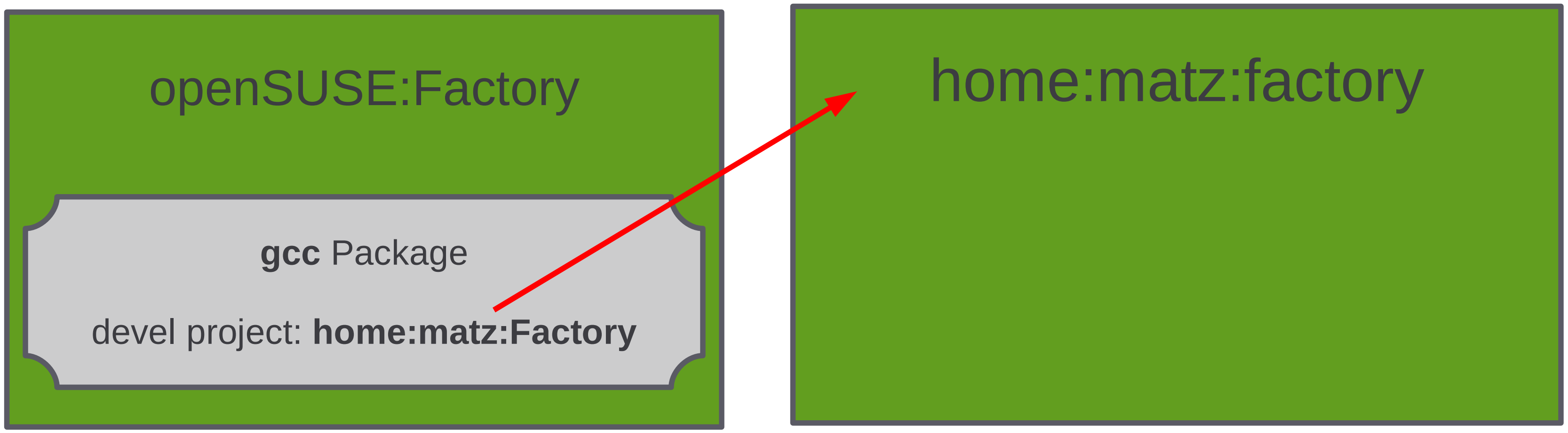
Project or package owners can grant write access to others. This is the fastest way to collaborate for a group working close together. However this works not in all cases:

- Unknown contributors have no write access.
- Trust is decreased to the person of lowest trust.
- Even people with write access might want a review of their changes before checkin.
- All people with write access can trigger or block package build of others.

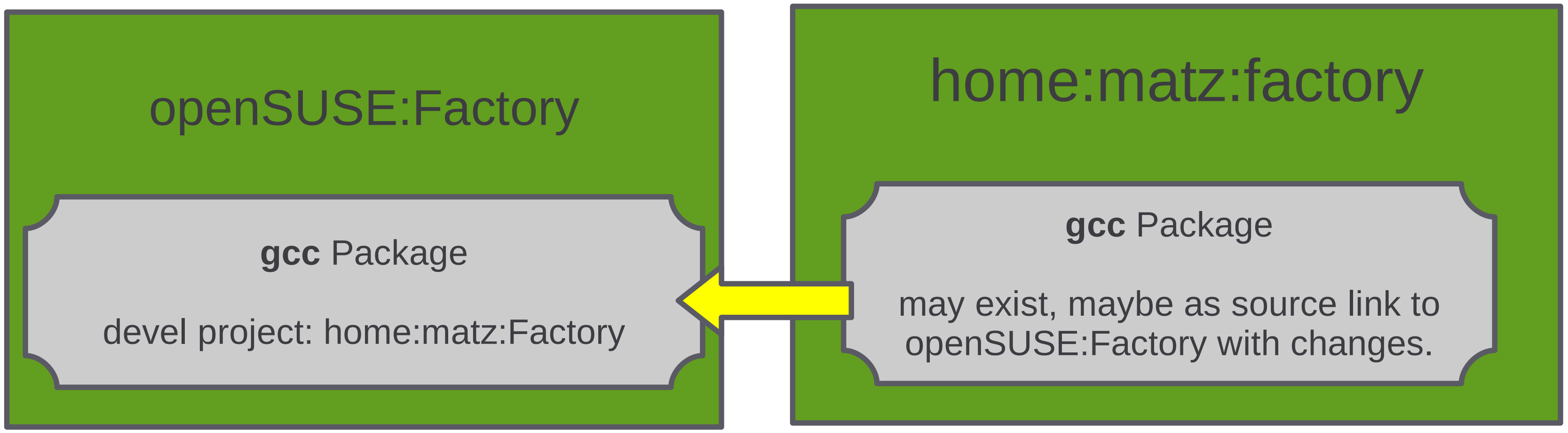
High Level Distribution Build Process



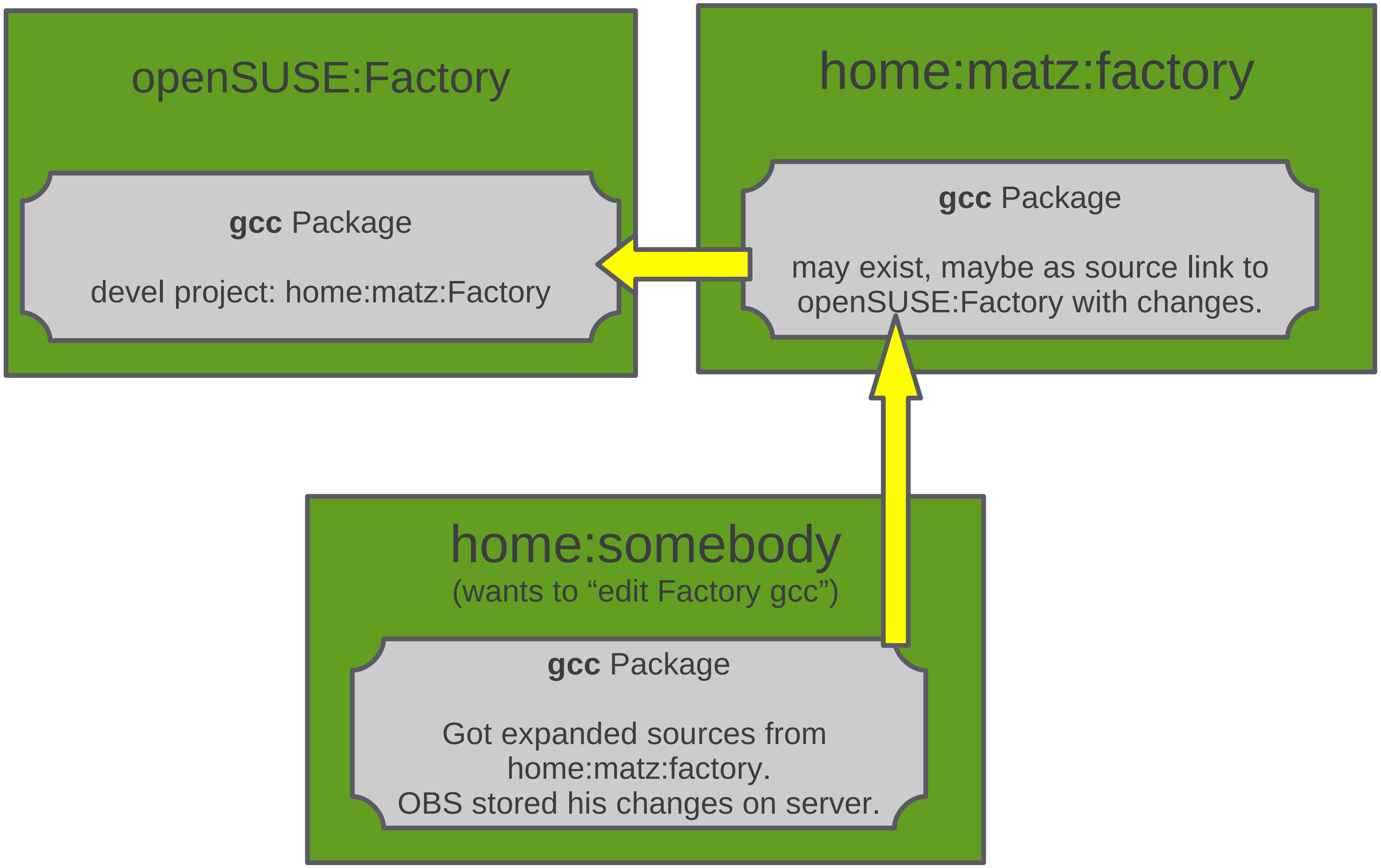
Factory Source Submissions



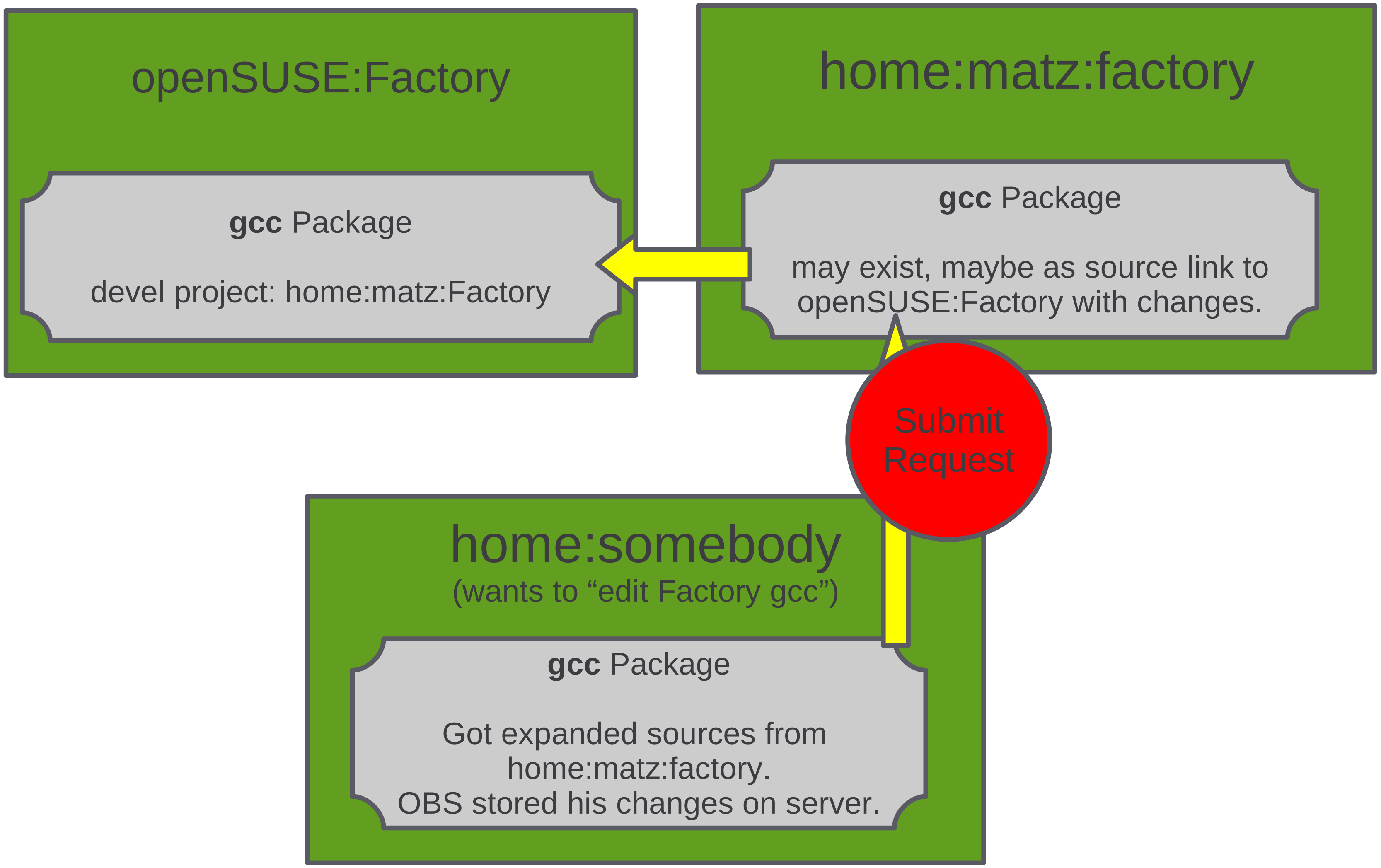
Factory Source Submissions


openSUSE:Factory
gcc Package
devel project: home:matz:Factory
home:matz:factory
gcc Package
may exist, maybe as source link to openSUSE:Factory with changes.

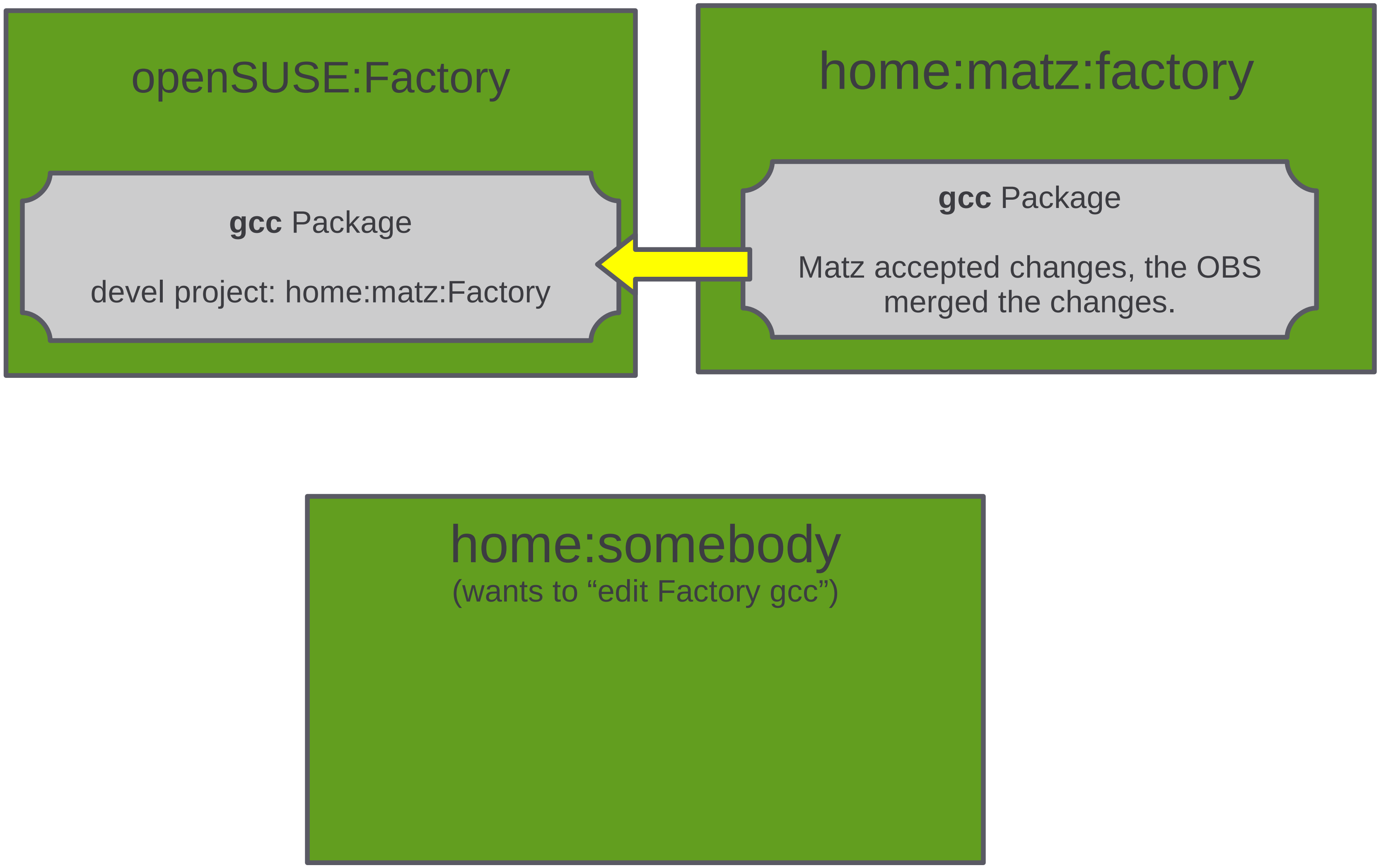
Factory Source Submissions


openSUSE:Factory
gcc Package
devel project: home:matz:Factory
home:matz:factory
gcc Package
may exist, maybe as source link to openSUSE:Factory with changes.
home:somebody
(wants to "edit Factory gcc")
gcc Package
Got expanded sources from home:matz:factory.
OBS stored his changes on server.

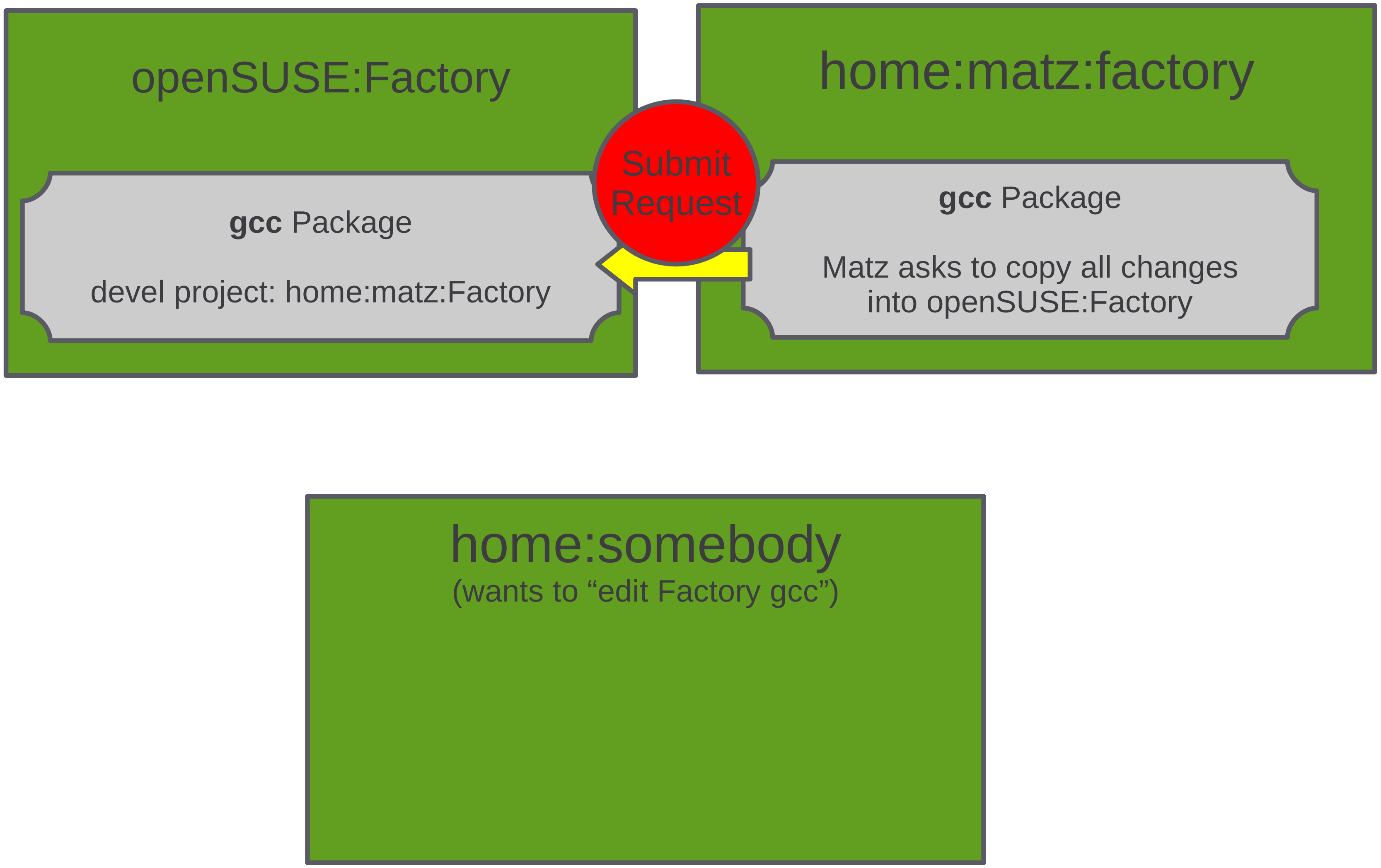
Factory Source Submissions


openSUSE:Factory
gcc Package
devel project: home:matz:Factory
home:matz:factory
gcc Package
may exist, maybe as source link to openSUSE:Factory with changes.
Submit Request
home:somebody
(wants to “edit Factory gcc”)
gcc Package
Got expanded sources from home:matz:factory.
OBS stored his changes on server.

Factory Source Submissions


openSUSE:Factory
gcc Package
devel project: home:matz:Factory
home:matz:factory
gcc Package
Matz accepted changes, the OBS merged the changes.
home:somebody
(wants to “edit Factory gcc”)

Factory Source Submissions


openSUSE:Factory
gcc Package
devel project: home:matz:Factory
Submit Request
home:matz:factory
gcc Package
Matz asks to copy all changes into openSUSE:Factory
home:somebody
(wants to "edit Factory gcc")

Collaboration Improvements

- Submissions can come via the OBS instead via bugzilla only
- Maintainers can see build success or failure before merged into project
- Extended responsibility for the contributor until acceptance of changes
- Transparent responsibilities and contributor log

Next steps

- We prepare “best practice” examples and Documentation during “1.0” development.

http://en.opensuse.org/Build_Service/Collaboration

- Participation and feedback is explicitly welcome!

- Create your home account and join us.

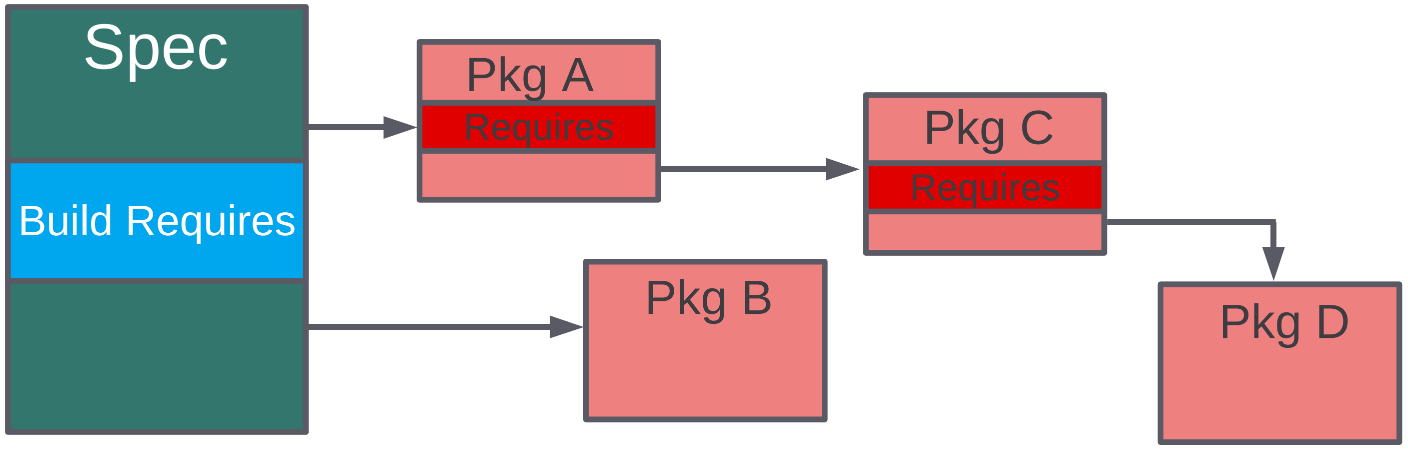
What is missing ?

- Internal STABLE checkin tool based on submissions from OBS
- Branch command in api.o.o and osc
- Factory ftp tree generation with OBS
- Lots of small improvements to make it nice usable. This depends on feedback :)

Advanced Packaging

Setting Up the Build Environment

- The build service parses BuildRequires / Build-Depends from spec file / dsc file.
  - These packages get added to a “base system”
  - Packages get automatically added so that all of the run-time dependencies are met



Breaking Dependencies

- To get rid of excess packages one can break the unwanted dependencies.
- Dependencies can be broken on the project level (affects every package of the project) or on the package level:
  - project level: by adding “Ignore:” lines to the project configuration
    ```
    Ignore: tetex:xorg-x11-libs
    ```
  - package level: by adding “#!BuildIgnore” lines to the specfile
    ```
    #!BuildIgnore: xorg-x11-libs
    ```

Dealing with Ambiguities

- Ambiguities can happen if two packages provide the same functionality.
- The system treats ambiguities as errors:

- Specfile:

```
        BuildRequires: apache2
```

-

```
expansion errors:
 have choice for apache2-MPM needed by
 apache2: apache2-prefork apache2-worker
```

Dealing with Ambiguities

- To solve ambiguities, either select one of the choices or deselect all unwanted ones:
  - project level: “Prefer” lines

```
    Prefer: apache2-prefork
    Prefer: -apache2-worker
```

  - package level: “BuildRequires” / “#!BuildIgnores”

```
    BuildRequires: apache2-prefork
    #!BuildIgnore: apache2-worker
```

Automatic Dependency Rewriting

- Problem: packages get renamed or are named different for different distributions.
  - Example: package containing shared libraries for canna

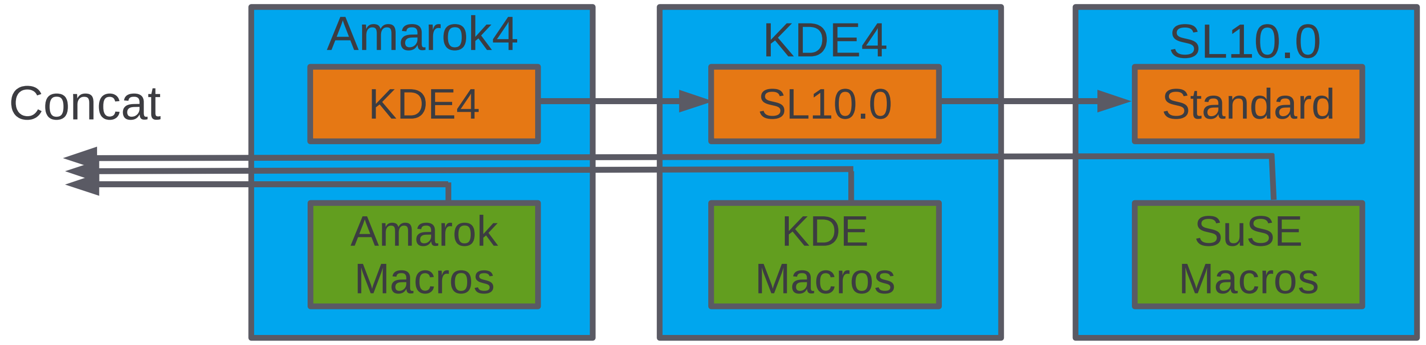
SUSE:	canna-libs
Fedora:	Canna-libs
Mandriva:	libcanna1
Debian:	libcanna1g

- Project can specify per repository dependency rewrite rules:

  `Substitute:` *<package> <replacement packages>*

Project Specific Build Data

- A project consists of:
  - A number of packages and repositories
  - Macros for the project
  - Information for setting up the build environment
- The build process concatenates the configuration of every involved project.
  - The repository search path defines which projects to use



Adding a Service

- openSUSE:

  postinstall:

  ```
  %{fillup_and_insserv -f <srv>}
  ```

  preuninstall:

  ```
  %stop_on_removal <srv>
  ```

  postuninstall:

  ```
  %restart_on_update <srv>
  %insserv_cleanup
  ```

- Mandriva:

  postinstall:

  ```
  %_post_service <srv>
  ```

  preuninstall:

  ```
  %_preun_service <srv>
  ```

Adding a service (cont.)

- Fedora:

```
postinstall:
        /sbin/chkconfig --add <srv>
preuninstall:
        if [ “$1” = 0 ] ; then
                service <srv> stop >/dev/null 2>&1
                /sbin/chkconfig --del <srv>
        fi
postuninstall:
        if [ “$1” -ge 1 ] ; then
                service <srv> condrestart >/dev/null
2>&1
        fi
```

Adding a Service (cont.)

- Proposed macros:

  postinstall:

  ```
  %service_add <srv>
  ```

  preuninstall:

  ```
  %service_del_preun <srv>
  ```

  postuninstall:

  ```
  %service_del_postun <srv>
  ```

Adding Specials to Spec Files

- Used statements:

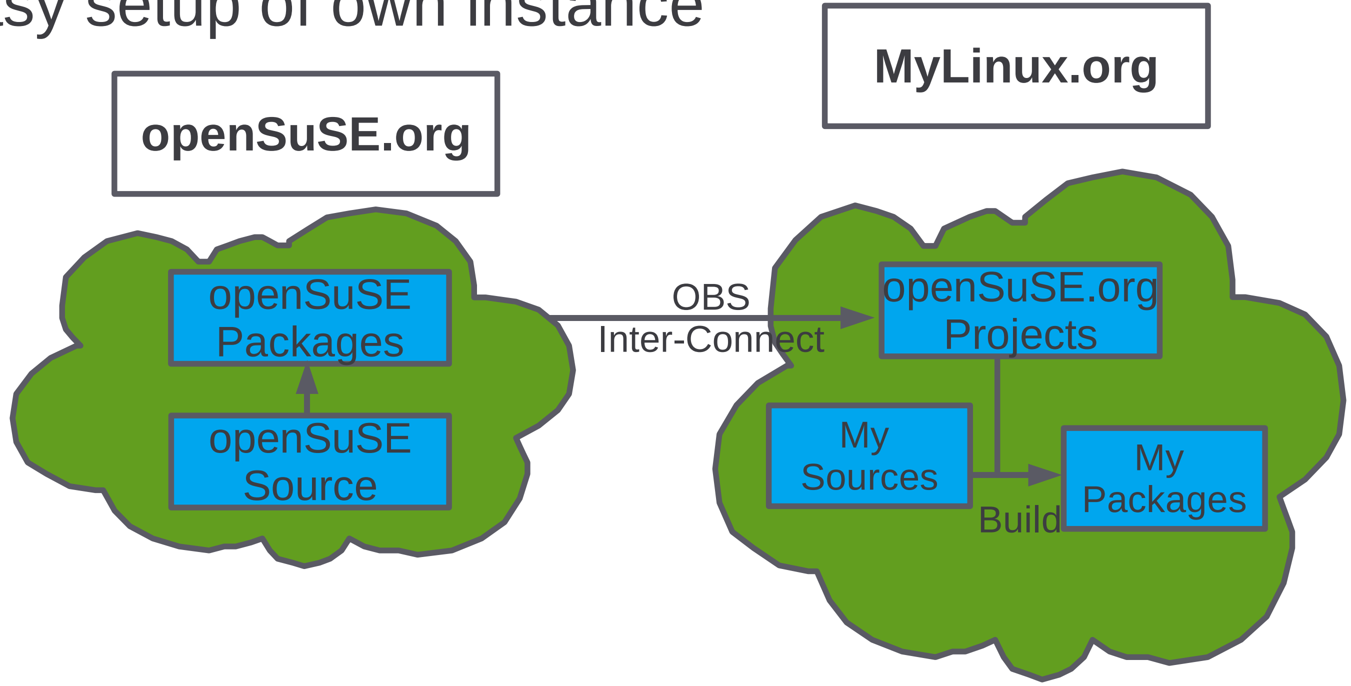
```
%if 0%{?suse_version} < 1010
        do something
%endif

%if 0%{?mandriva_version}
        Name: libopensync
%else
        Name: opensync
%endif
```

Build Service Inter-Connect

Inter-Connect

- Projects on remote build service instance gets accessable.
- Automatic event handling on changes
- Easy setup of own instance



Possible Use Cases

- Reuse Base Projects from openSUSE.org
- Recompile with different compile flags
- Compile for new hardware architecture
- Replace base components (other compiler for example)
- Compile more and more often with own build power

Distribution Generation

Goals of the Image Building in OBS

- Have all related input data in one place
- Same interface for all kind of data
- Focused on clean and reproducable builds from scratch
- Allow easy creatable derivates of our distribution medias
- No space for a high number of forks on our server and mirrors, but we can host a nearly unlimited number of configurations.
- Allow client side building and debugging

Batch Job Processing

- The job scheduling via dependencies leads to batch job processing.

- Advantages
  - Consistency
  - Earliest possible job start
  - Scalability
- Disadvantages
  - Job start is not defined by user
  - No user interactivity with batch job

Dependency Calculation

- Binary Builds
  - RPM: Requires: / Provides: / BuildRequires:
  - DEB: Requires / Provides / BuildRequires
  - Images: Package & Pattern definition from KIWI config
  - QA: TBD
  - Windows Binaries: TBD
  - MacOS-X Binaries: TBD

Skipped Topics

- So much to tell and
- Multi-Distribution/Architecture Add-On Builds
- Interfaces of OBS
- Collaboration Concepts
  - Derivate setup
  - Developer Interaction
- Trust handling
  - Trust in sources

Current Status

- Implementation supporting KIWI is WIP
- Additional support of other imaging tools should be easy afterwards

- Problems / Limitations:
  - KIWI lacks currently support for installation medias (WIP)
  - Conflicting definitions of source repos in project definition and kiwi config
  - Only repos from OBS are allowed to keep reproducable results and trust concept

Current State / Next Steps

Current State

- Everybody can register at build.opensuse.org
- Everybody can setup an own instance
- Package building for openSUSE, Mandriva, Fedora, Debian, Ubuntu, SLE and RHEL works
- Software is accessable via http://software.opensuse.org/search

Current WIP

- New software portal gets developed
- Developer collaboration (enables Factory contribution)
  - Merge request and execution
- Notifications
- Version 1.0 is coming up.

Outlook

- Complete the distribution build support
  - Maintanance handling
  - Hidden builds (needed for security fixes)
- Trust system to rate developers
- LSB conform builds
- QA and automated test case framework

Future Topics

- Future ideas are collected at
- http://www.opensuse.org/Build_Service/Future_Ideas

- Template based package creating
- Translation framework

Resources

- http://build.opensuse.org
- A running instance of the Build Service.
- Contains links to documentation and source.

- opensuse-buildservice@opensuse.de
- The mailing list for discussing the Build Service.

- #opensuse-buildservice on freenode
- Our IRC channel

Join the Lizard Blizzard!

... Questions?

Novell®