HPC-Forge

A Software Development Infrastructure

Agenda

- Introduction
- The portal
- The services:
  - Subversion
  - Trac
  - Hudson
  - WebDAV

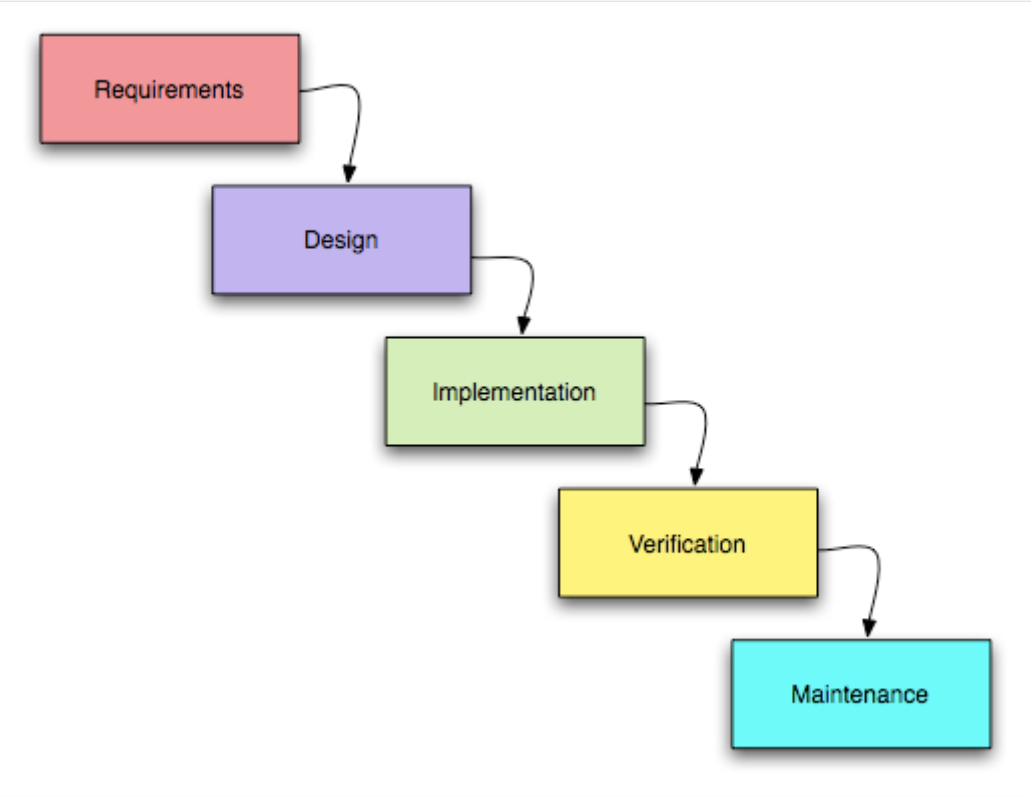
Software development process

- A **software development process** is a structure imposed on the development of a software product. Synonyms include software life cycle and *software process*. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process.

Software development process

- Software development activities
  - Planning
  - Implementation, testing and documenting
  - Deployment and maintenance
- Models
  - Iterative processes
  - XP: Extreme Programming
  - Waterfall processes
  - Other models

the waterfall model



What's HPC-Forge?

HPC-Forge is a software development infrastructure: a collection of services to support the software development process:

- **Source control management**: A system that provides a central place where the team members can store and access their entire source code base.
- **Requirements management**: A system used for recording and tracking product feature requests.

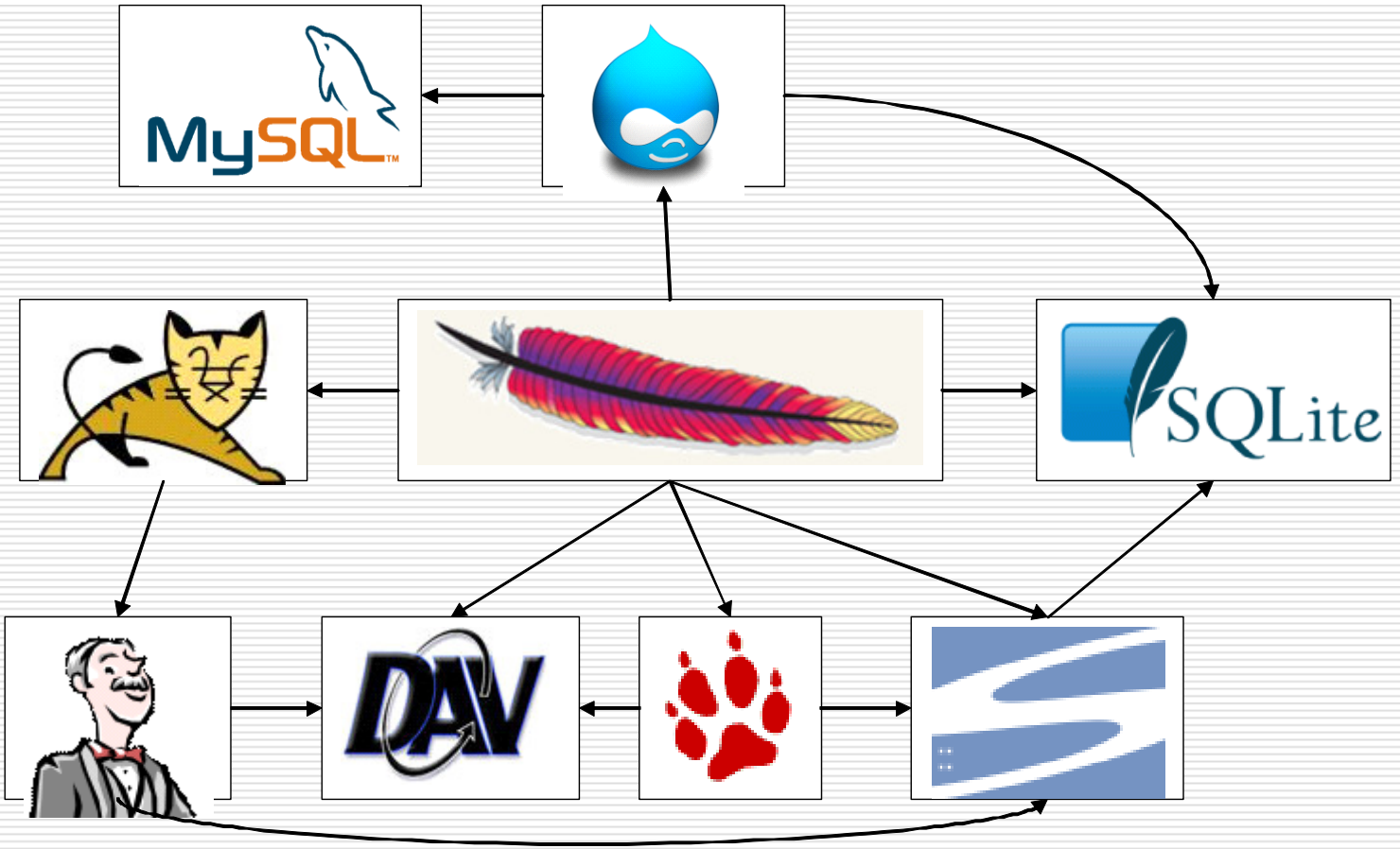
What's HPC-Forge?

- **Bug-tracking**: A system used to record and track errors and feature requests.
- **Automated build**: A system that builds the application every night by automatically executing the required build procedure steps at the scheduled time, without any human intervention.
- **Automated testing**: The tools that team developers and testers use to verify software and to detect and prevent software problems, such as functionality errors, reliability problems, performance problems, or security vulnerabilities..
- **Regression testing**: Any tool or combination of tools that can automatically run all of your existing tests on your entire code base on a regular basis (preferably nightly, as part of the automated build). Its purpose is to help you identify when code modifications cause previously working functionality to regress, or fail. For example, the regression system may be a script that runs one or more automated testing tools in batch mode.
- **Data repository**: A storage area (upload/download) to provide access to 'publish' releases, documentation, test data …

Which services?

- **Source control management:Subversion (http://subversion.tigris.org)**
- **Requirements management, Bug-tracking:Trac (http://trac.edgewall.org)**
- **Automated build an testing: Hudson (http://hudson-ci.org/)**
- **Data repository: WebDAV**
- **A web-based interface (portal): to access and create service instances, to manage users and their permissions.**

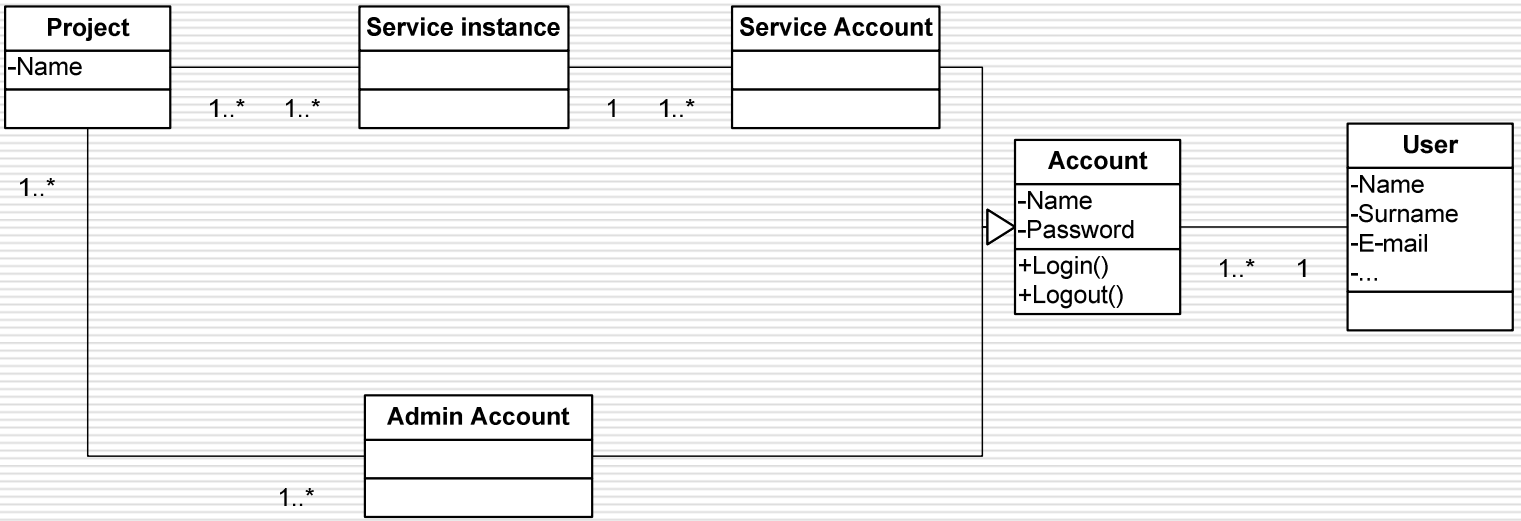
HPC-Forge Architecture



HPC-Forge's organization

1. HPC-Forge is a software development infrastructure oriented to projects.
2. The infrastrucure has an administrator (the "system administrator") to manage everything: projects, service instances, users, permissions, …
3. A project can get one or more service instances.
4. Every project has one or more administrator (the "project administrators") to setup the project itself and it's service instances.
5. Every service istance has one or more administrator (the "instance administrators") to setup the instance itself, it's users and their permissions.
6. The users can access the service instances following the rules established by the instance administrator and typical of the service.
7. The access can be anonymous for guests or authenticated for registered users. Authentication is in **single-sign-on** mode.
8. Every registered user has his personal information and one or more accounts.
9. Resources are in **auto-provisiong**. Every authenticated user can create projects and service instances by itself.

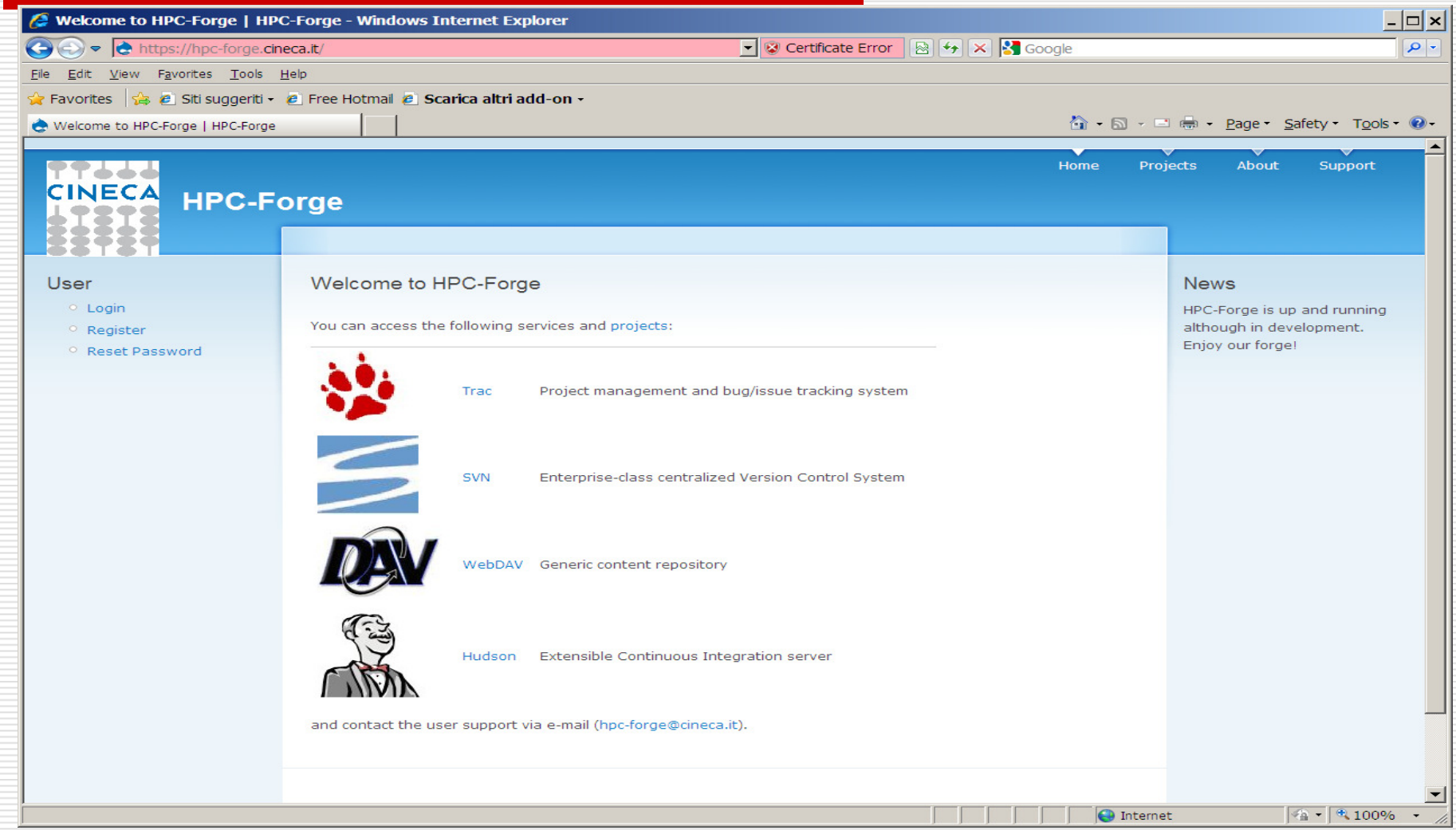
HPC-Forge's organization



HPC-Forge's organization

Function/role	guest	users	Project administrators	Instance administrators
General information	x	x	x	x
registration	x			
Personam information and password change		x	x	x
login/logout		x	x	x
Services access	x (anonymous)	x (authenticated)	x (authenticated)	x (authenticated)
Project creation		x	x	x
Project setup and instances creation			x	
Instance setup and user management				x

The HPC-Forge's portal

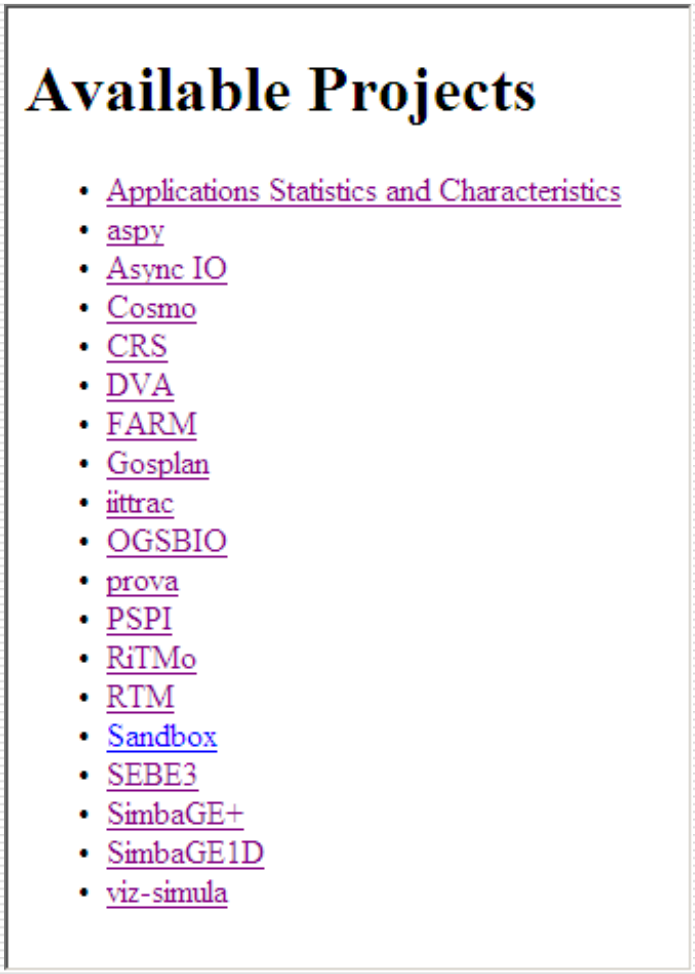


Surfing at HPC-Forge

- Service instances can be accessed throughout two paths:
  - By service collections:
    clicking on the link of a service at the home page. You will get a list of instances of that service.
  - By project collections:
    clickingt on the “Projects” link. You will get a list of instances of that project grouped by service type.

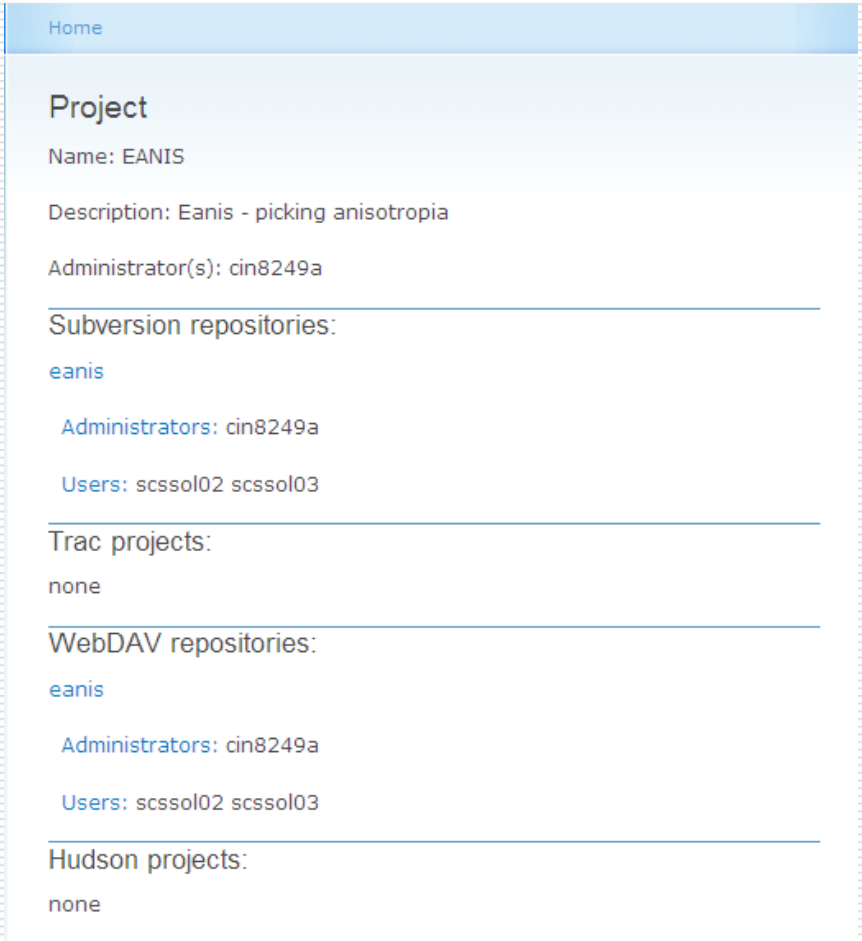
Service instances by service type

The user can access a certain instance (in this case a Trac project) by clicking on the corresponding link



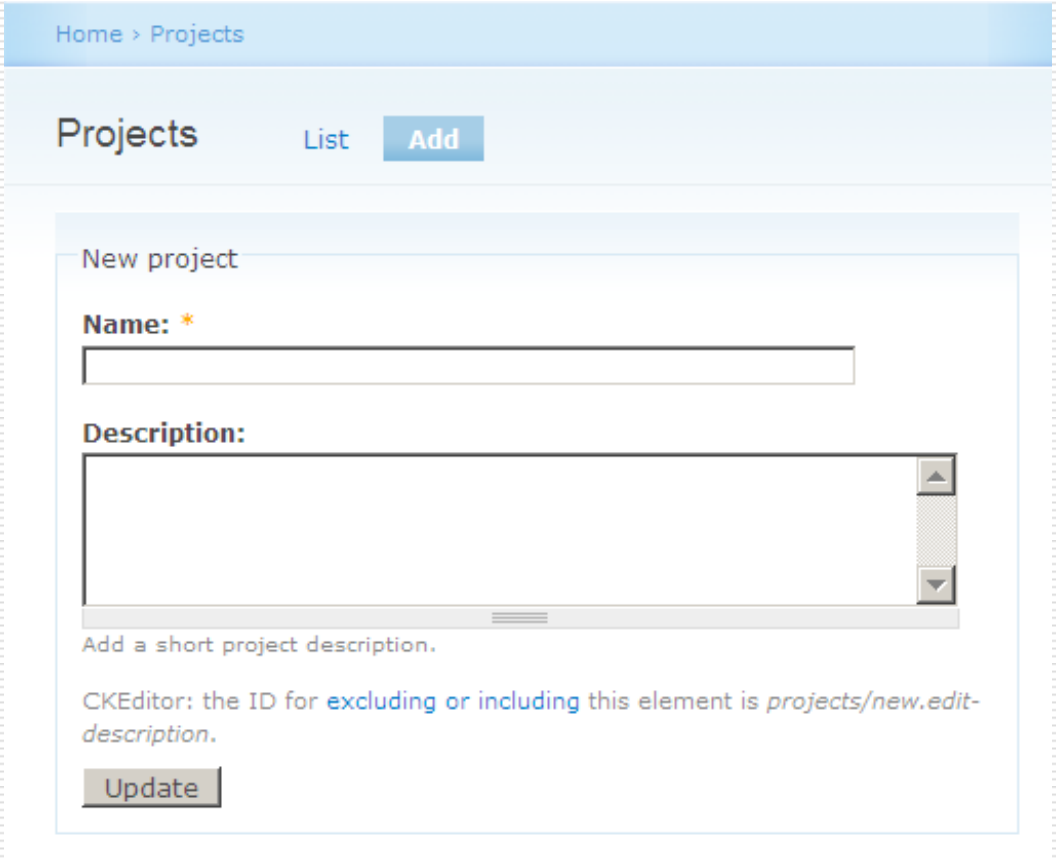
Service instances of a project

The user can access a certain instance by clicking on the corresponding link



Creating projects

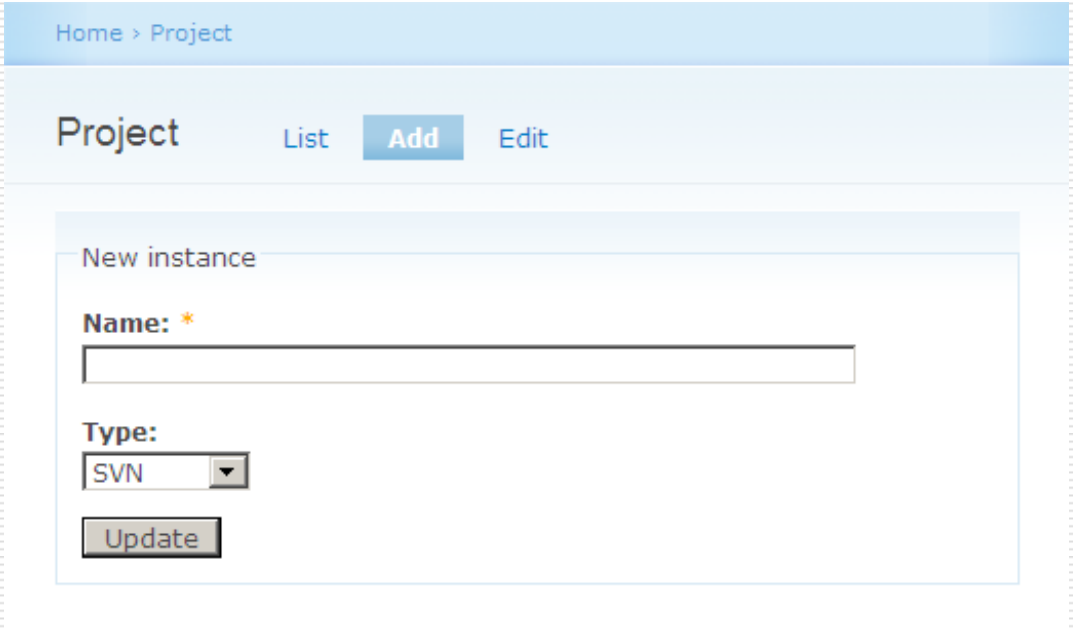
Only authenticatd users can create project through the "Add" menù of the "Projects" page



Creating service instances

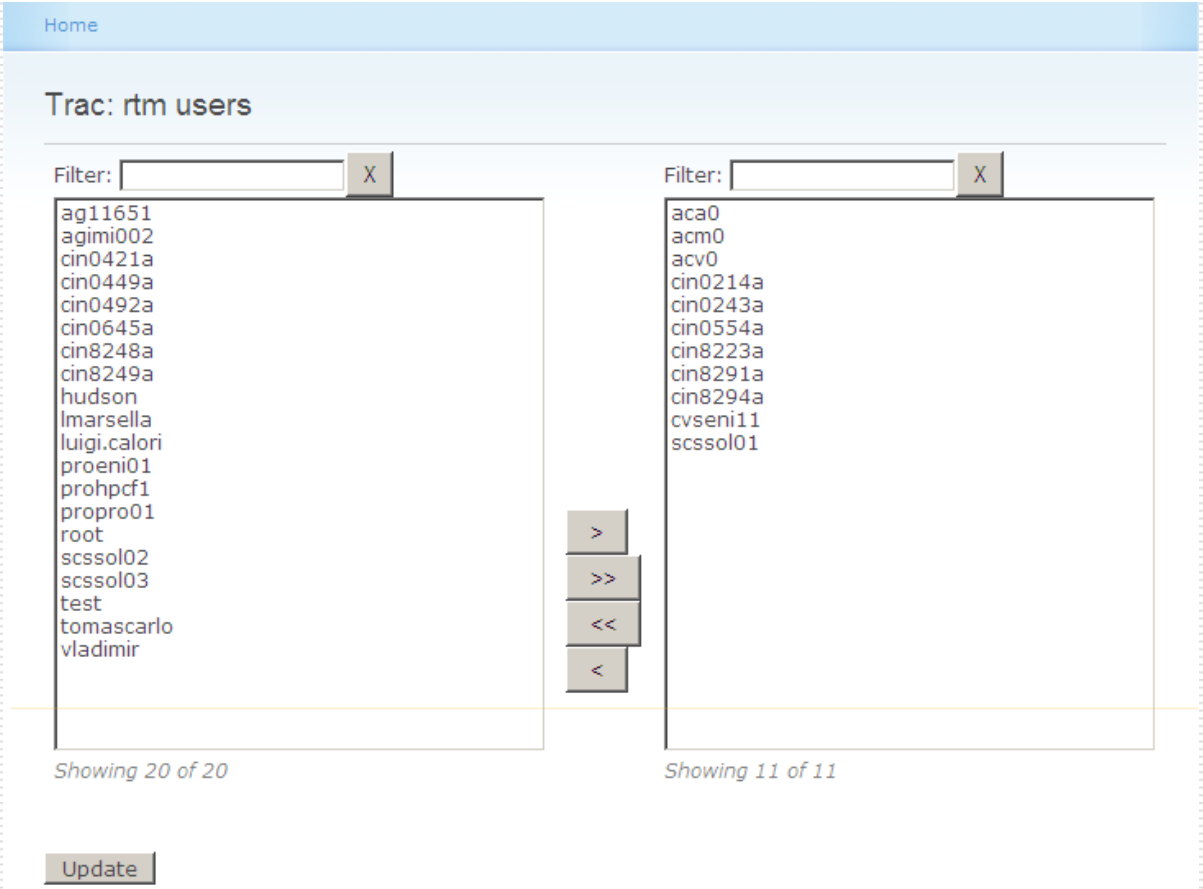
Only project administrators can create service instances through the “Add” menù of a project page.

They become also administrators of the new instances.



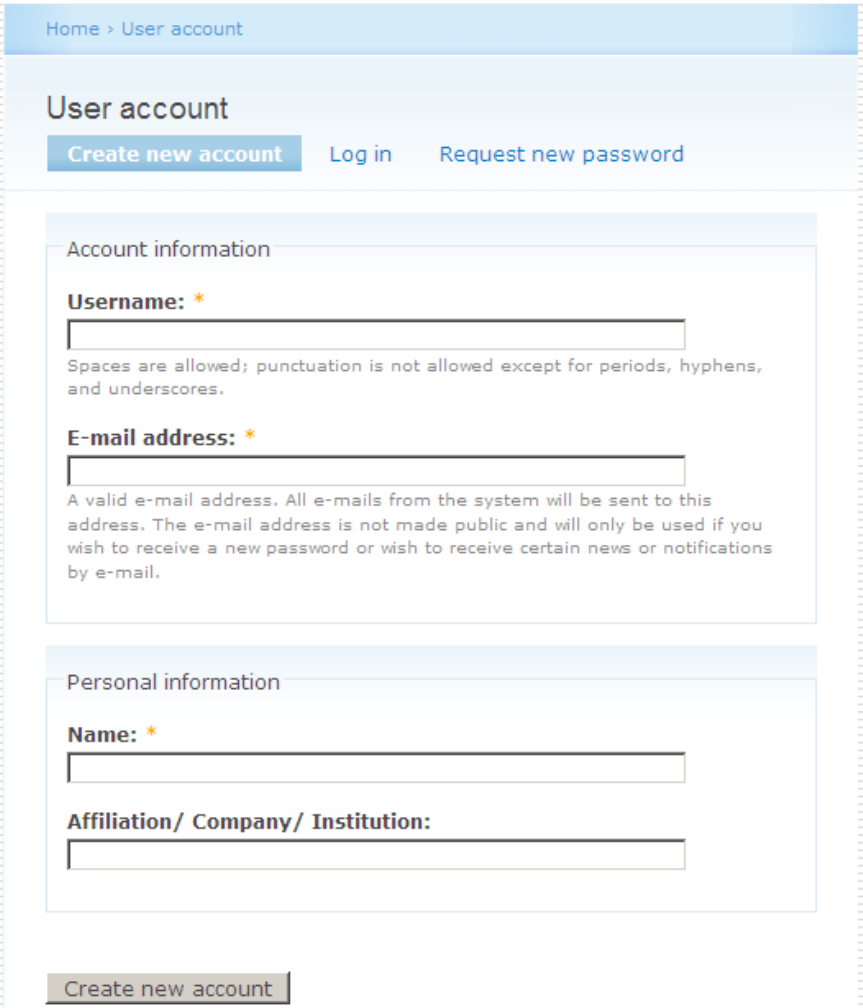
Administer users

Only service instance administrators can administer users.



Register yourself

- Click on the register link
- Fill the form
- You will receive an e-mail with a temporary password to login and change it.



Why Revision Control?

- Provides a place to store your code
- Reduce clutter
- Independent of individual accounts
- Historical record of what was done over time
- Safety net
- Synchronization between developers

Why Use Subversion?

- Functional superset of CVS
- Directory versioning (rename and moves)
- Atomic Commits
- File meta-data
- True client-server model
- Cross-platform, open-source

Subversion Architecture

- Each working copy has a .svn directory
  - Similar to the CVS's CVS directory
  - Stores metadata about each file
- Local or Network Repository
- Network access over HTTP or SSH
- Encrypted authentication
  - Cleartext password stored in *~/.subversion*
- Fine-grained authorization
- Command line client is `svn`

CVS vs SVN

- Most CVS commands exist in SVN
  - Checkout, commit, update, status…
- Arguments position matters in CVS

```
$ cvs -d /cvsroot update -d
```

- Not so in SVN

```
$ svn log -r 123 foo.c
$ svn log foo.c -r 123
```

Revisions (1)

- Revision numbers are applied to an object to identify a unique version of that object.
- CVS
  - Revision numbers are per file.
  - No connection between two files with the same revision number.
  - A commit only modifies the version number of the files that were modified.
    - foo.c rev 1.2 and bar.c rev 1.10.
  - After commit of bar.c:
    - foo.c rev 1.2 and bar.c rev 1.11.

Revisions (2)

- Revision numbers are applied to an object to identify a unique version of that object.
- SVN
  - Revision numbers are global across the whole repository.
  - Snapshot in time of the whole repository.
  - A commit modifies the version number of all the files.
    - foo.c rev 25 and bar.c rev 25.
  - After commit of bar.c:
    - foo.c rev 26 and bar.c rev 26.
  - foo.c rev 25 and 26 are identical.

Basic Work Cycle (1)

- Checkout a working copy
- Update a working copy
- Make changes
- Examine your changes
- Commit your changes

Basic Work Cycle (2)

- Checkout a working copy

```
$ svn checkout \
> https://hpc-forge.cineca.it/svn/TEST/foo
$ cd foo
```

- Update a working copy
  - Update all files

```
$ svn update
```

  - Update to an older revision

```
$ svn update -r 45
```

  - Update an older revision of *bar.c*

```
$ svn update -r 23 bar.c
$ svn update -r 1
A  changepwd.form
D  trunk
D  branches
Updated to revision 1.
```

Basic Work Cycle (3)

- Update output
  - *U* foo
    - File *foo* was (U)pdated (pulled from repository)
  - *A* foo
    - File *foo* was (A)dded to your working copy
  - *D* foo
    - File *foo* was (D)eleted from your working copy
  - *R* foo
    - File *foo* was (R)eplaced, that is it was deleted and a new file with the same name was added.
  - *G* foo
    - File *foo* received new changes and was also changed in your working copy. The changes did not collide and so were mer(G)ed.
  - *C* foo
    - File *foo* received (C)onflicting changes from the server. The overlap needs to be resolved by you.

Basic Work Cycle (4)

- Make changes
  - Add new files and directories

```
$ touch README.txt
$ svn mkdir Presentations
$ touch Presentations/simple.txt
$ svn add Presentations/simple.txt README.txt
```

  - Delete files

```
$ svn rm foo
```

  - Rename files

```
$ svn mv README.txt README_OLD.txt
```

  - Copy files and directories

```
$ svn cp Presentations Presentation_new
```

Basic Work Cycle (5)

- Examine your changes
  - svn status: list of changed files

```
?       arcanum.pdf File is not managed by svn
M       howto.tex   File has local content modifications
A       howto.toc  File is scheduled for addition
D       Makefile    File scheduled for deletion
```

- Even more details with -v
  - Revision numbers
  - Who made last modification
- Status of repository with -u
  - Shows changes in repository as well

Basic Work Cycle (6)

- Examine your changes
  - svn diff: shows your modifications
  - In your local working copy

    ```
    $ svn diff
    ```

  - Between a repository revision and your local copy

    ```
    $ svn diff –r 34 foo.c
    ```

  - Between two repository revisions

    ```
    $ svn diff –r 2:5 foo.c
    ```

- Revert your changes

  ```
  $ svn revert foo.c
  ```

Basic Work Cycle (7)

- Commit your changes

```
$ svn commit
```

- Will open an editor to type in change log
- Alternatively, short logs can be input inline

```
$ svn commit –m "my short log"
```

- *Logs can be retrieved for a file or a tree*

```
$ svn log foo.c
```

Conflict Resolution

- Look for "C" when you update
- Better than CVS:
  - Conflict markers are placed in the file to display the overlap (just like CVS).
  - Three *tmp* files are created. these are the original three files that could not be merged.
  - SVN will not allow you to commit files with conflicts.
- Solutions to resolve
  - Hand-merge the files
  - copy one of the *tmp* files on top of your working copy
  - `svn revert` to toss out your changes
- Once resolved, you need to tell svn about it

```
$ svn resolve foo.c
```

File & Directory Properties (1)

- Each file and directory has a list of properties associated with it
- Arbitrary properties & values
- Subversion defines some properties:

```
svn:ignore
svn:eol-style
svn:mime-type
svn:executable
svn:keywords
```

- Listing properties

```
$ svn proplist README.txt
Properties on 'README.txt':
  svn:mime-type
  svn:eol-style
```

File & Directory Properties (2)

- Getting a property value

```
$ svn propget svn:mime-type README.txt
```

- Setting a property

```
$ svn propset svn:eol-style native README.txt
```

Dealing with binary files

- Subversion is optimized for dealing with text files (source code, LaTeX documents, etc)
- But, it can deal with binary files
  - Will not diff nor merge
  - Will not change EOL nor apply keywords
- SVN has a binary detection algorithm, but it sometimes fails (PDF have a text header)
  - Need to set *svn:mime-type* property manually to application/octet-stream

Repository Organization

- Per-project directories
- Three subdirectories per project:
  - trunk, tags, branches
- Trunk is for main development
- Tags is for read-only snapshots
- Branches is a work area

Working with Branches

- Create a new branch (NOTE. Replace TEST by the module that you want to work with)

```
$ svn cp https://hpc-forge.cineca.it/svn/TEST/trunk \
https://hpc-forge.cineca.it/svn/TEST/branches/my-branch
Commited revision 6
```

- Move to branch

```
$ svn switch https://hpc-
forge.cineca.it/svn/TEST/branches/my-branch
```

- Make Changes...
- Back to the main trunk

```
$ svn switch https://hpc-forge.cineca.it/svn/TEST/trunk .
```

- Merge branch into trunk

```
$ svn merge \
https://hpc-forge.cineca.it/svn/TEST/branches/my-branch .
```

Best Practices

- Commit early, commit often
- Commit logical changesets
- Track merges manually
  - When committing the result of a merge, write a descriptive log

```
Merged revisions 3490:4120 of /branches/foobranch to /trunk
```

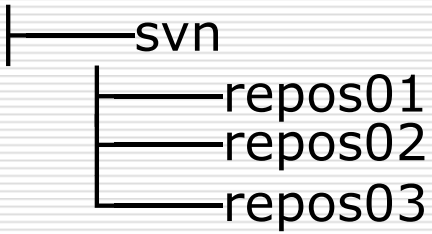
- Be patient with large files and repositories
- Know when to create branches
- Trunk should be *stable*

popular Subversion clients

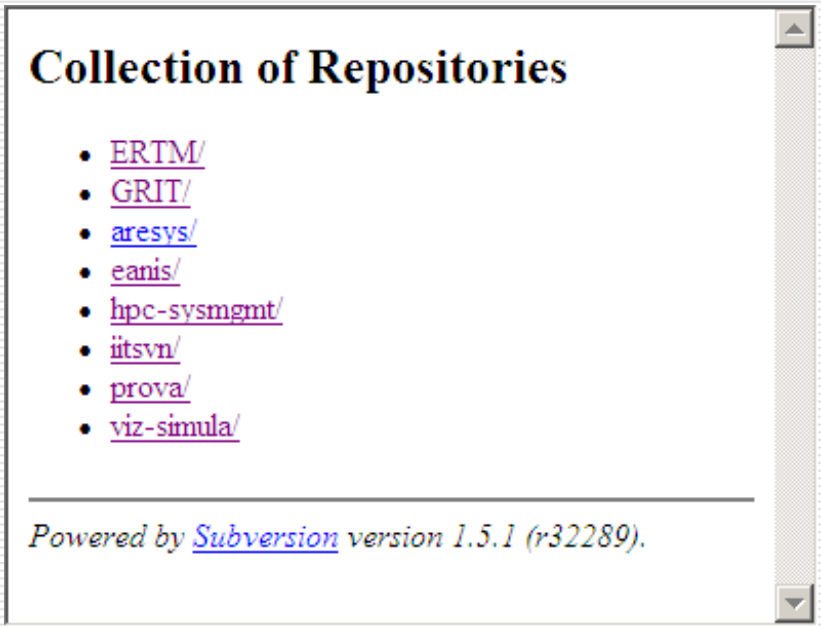
- svn
  - This is the standard command-line client for Subversion. It is free, runs on any platform, and comes packaged with the standard Subversion download.
- TortoiseSVN
  - TortoiseSVN is a free client for Windows users who prefer graphical interfaces. It works as an extension of the standard Windows Explorer interface.
- IDE Plug-ins
  - Several IDEs includes clients. Subclipse adds all the features of Subversion into Eclipse so that you do not have to download Subversion separately.
- RapidSVN
  - is a graphical SVN client available for many platforms

Subversion at HPC-Forge

https://hpc-forge.cineca.it/svn



Anonymous access to a repository is denied by default but can be enabled throughout the "path based" permissions.



Repositories can be accessed via https:

```
svn checkout https://hpc-forge.cineca.it/svn/repos1/src
```

path-based access control

```
### This file is an example authorization file for svnserve.
### Its format is identical to that of mod_authz_svn
      authorization
### files.
### As shown below each section defines authorizations
      for the path and
### (optional) repository specified by the section name.
### The authorizations follow. An authorization line can
      refer to:
###  - a single user,
###  - a group of users defined in a special [groups]
      section,
###  - an alias defined in a special [aliases] section,
###  - all authenticated users, using the '$authenticated'
      token,
###  - only anonymous users, using the '$anonymous'
      token,
###  - anyone, using the '*' wildcard.
###
### A match can be inverted by prefixing the rule with
      '~'. Rules can
### grant read ('r') access, read-write ('rw') access, or no
      access
### ('').
```

```
[aliases]
# joe = /C=XZ/ST=Dessert/L=Snake City/O=Snake
      Oil, Ltd./OU=Research Institute/CN=Joe
      Average

[groups]
# harry_and_sally = harry,sally
# harry_sally_and_joe = harry,sally,&joe

# [/foo/bar]
# harry = rw
# &joe = r
# * =

# [repository:/baz/fuz]
# @harry_and_sally = rw
# * = r
```

For More Information

- Subversion project home
  - http://subversion.tigris.org
- Subversion online book
  - http://svnbook.red-bean.com
- Subversion QuickRef
  - http://subversion.tigris.org/files/documents/15/177/svn-ref.ps

What is Trac?

- Lightweight web based project management framework
- Open Source - Modified BSD License
- Developed at http://trac.edgewall.com
- Widely used by a variety of Open Source projects

MythTV

Search

Login | Help/Guide | About Trac | Preferences

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search

Start Page | Index | History | Last Change

Welcome to the MythTV Trac server

Stable: If you'd like to get the 0.21 stable branch, do this:

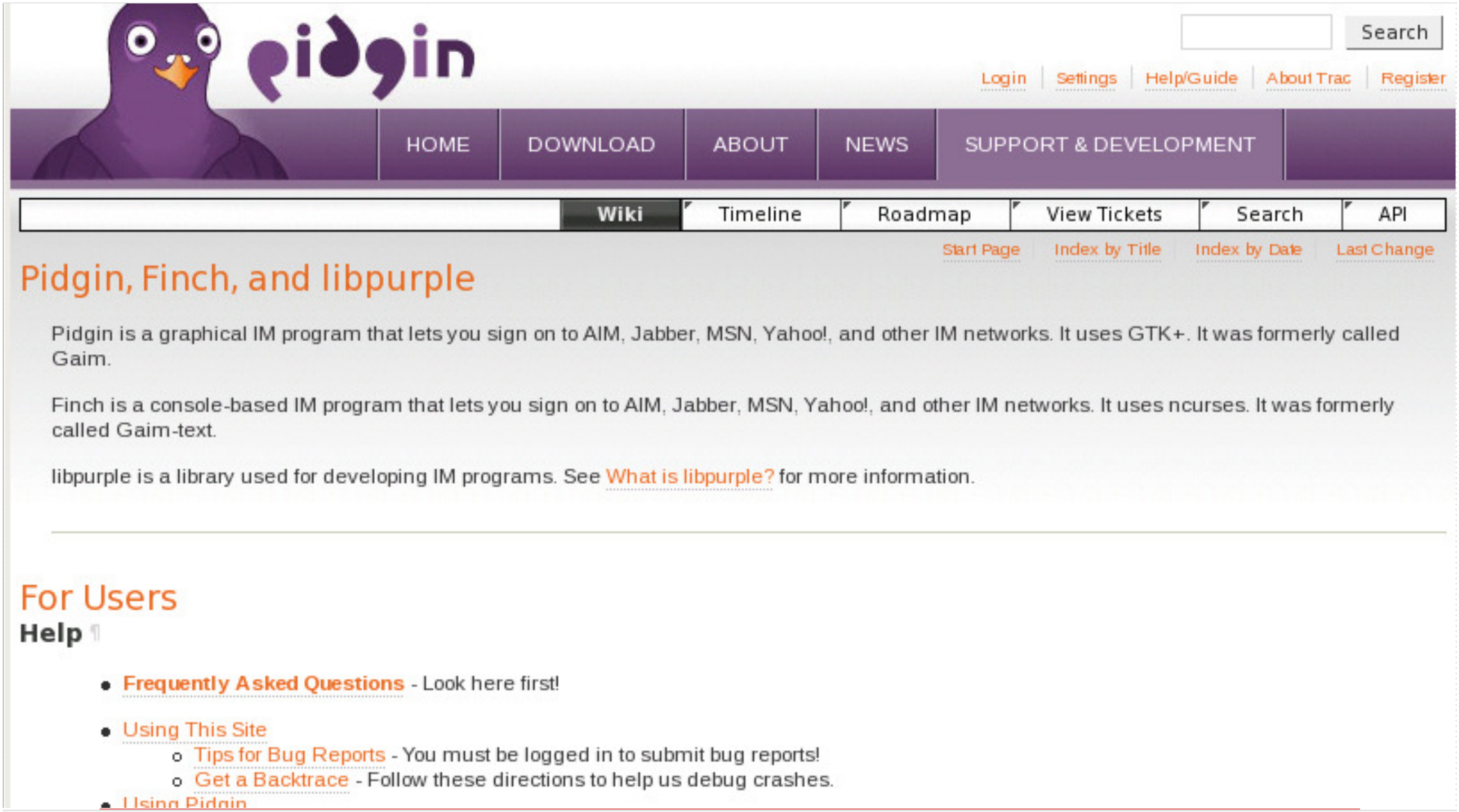
```
svn co http://svn.mythtv.org/svn/branches/release-0-21-fixes/
```

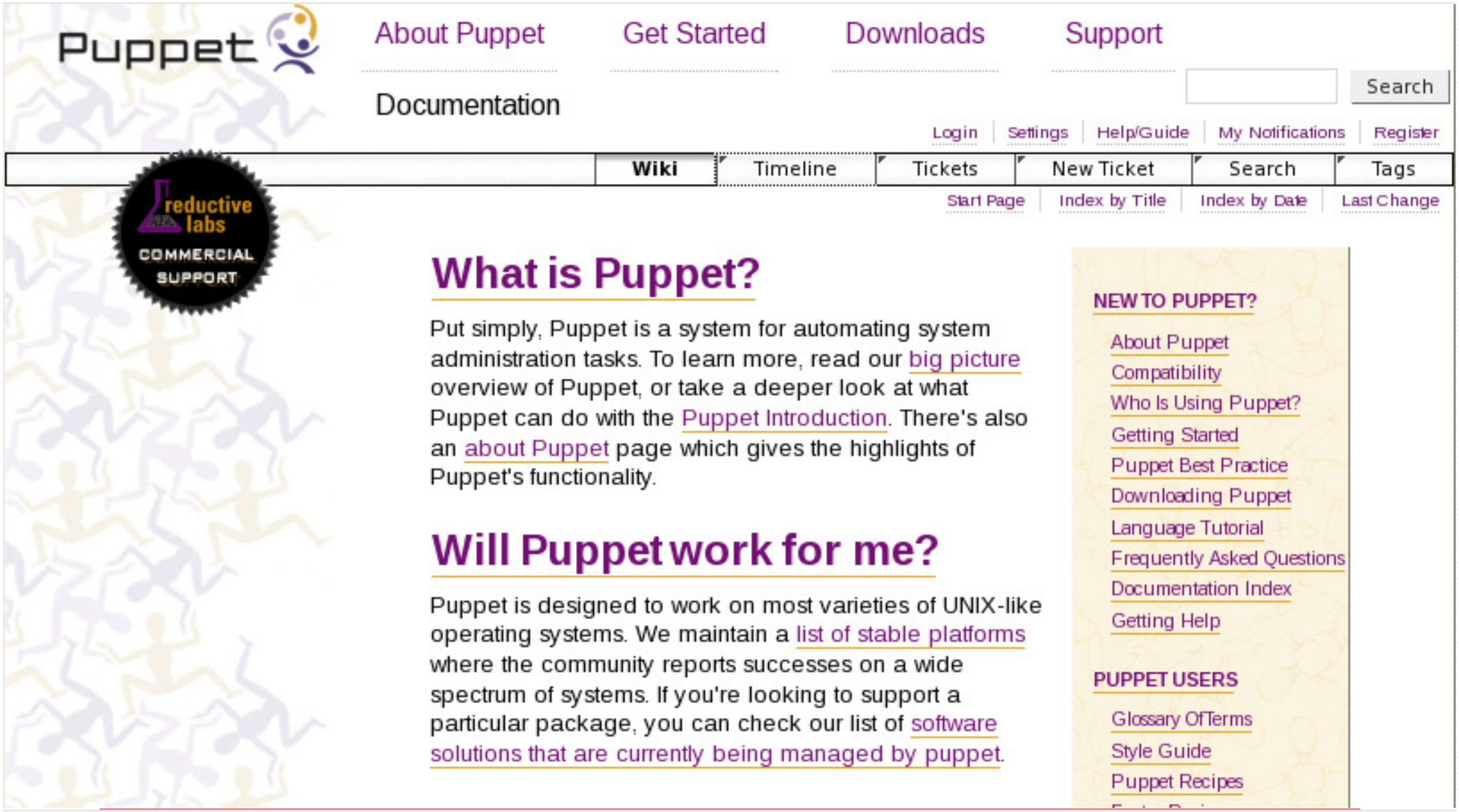
Unstable: If you're looking for anonymous subversion access for the development version, use the following command to grab the latest code:

```
svn co http://svn.mythtv.org/svn/trunk/
```

Once you have a checkout, you can just do a **svn update** at any time to update it to the most recent revision. If you're making modifications, a simple **svn diff** will generate a unified diff. It's quite similar to CVS, but the subversion book has all the information you'd ever want on how to use it. The Basic Work Cycle chapter is especially handy.

Please subscribe to the mythtv-dev mailing list if you plan to run trunk! Instead of sending patches to the mythtv-dev mailing list for inclusion to MythTV, please create a ticket instead (TicketHowTo). This should allow us to track patches better, and hopefully get them handled in a much more timely manner (and avoid losing patches).


pidgin
Search
Login
Settings
Help/Guide
About Trac
Register
HOME
DOWNLOAD
ABOUT
NEWS
SUPPORT & DEVELOPMENT
Wiki
Timeline
Roadmap
View Tickets
Search
API
Start Page
Index by Title
Index by Date
Last Change
Pidgin, Finch, and libpurple
Pidgin is a graphical IM program that lets you sign on to AIM, Jabber, MSN, Yahoo!, and other IM networks. It uses GTK+. It was formerly called Gaim.
Finch is a console-based IM program that lets you sign on to AIM, Jabber, MSN, Yahoo!, and other IM networks. It uses ncurses. It was formerly called Gaim-text.
libpurple is a library used for developing IM programs. See What is libpurple? for more information.
For Users
Help
Frequently Asked Questions - Look here first!
Using This Site
Tips for Bug Reports - You must be logged in to submit bug reports!
Get a Backtrace - Follow these directions to help us debug crashes.
Using Pidgin


Puppet
About Puppet
Get Started
Downloads
Support
Documentation
Search
Login
Settings
Help/Guide
My Notifications
Register
Wiki
Timeline
Tickets
New Ticket
Search
Tags
Start Page
Index by Title
Index by Date
Last Change
reductive labs
COMMERCIAL SUPPORT
What is Puppet?
Put simply, Puppet is a system for automating system administration tasks. To learn more, read our big picture overview of Puppet, or take a deeper look at what Puppet can do with the Puppet Introduction. There's also an about Puppet page which gives the highlights of Puppet's functionality.
Will Puppet work for me?
Puppet is designed to work on most varieties of UNIX-like operating systems. We maintain a list of stable platforms where the community reports successes on a wide spectrum of systems. If you're looking to support a particular package, you can check our list of software solutions that are currently being managed by puppet.
NEW TO PUPPET?
About Puppet
Compatibility
Who Is Using Puppet?
Getting Started
Puppet Best Practice
Downloading Puppet
Language Tutorial
Frequently Asked Questions
Documentation Index
Getting Help
PUPPET USERS
Glossary Of Terms
Style Guide
Puppet Recipes

Why Trac?

- Provides an integrated approach to managing a software development project or team
- Key features -
  - Ticketing for tasks and bug tracking
  - Documentation via searchable simple to use Wiki
  - Version control with strong support for Subversion
  - All sections can reference each other
- Simple to install, configure, manage and use

Trac Front Page

trac
Integrated SCM & Project Management

Search

Login | Preferences | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | Search

Start Page | Index | History | Last Change

Welcome to Trac 0.11

Trac is a **minimalistic** approach to **web-based** management of **software projects**. Its goal is to simplify effective tracking and handling of software issues, enhancements and overall progress.

All aspects of Trac have been designed with the single goal to **help developers write great software** while **staying out of the way** and imposing as little as possible on a team's established process and culture.

As all Wiki pages, this page is editable, this means that you can modify the contents of this page simply by using your web-browser. Simply click on the "Edit this page" link at the bottom of the page. WikiFormatting will give you a detailed description of available Wiki formatting commands.

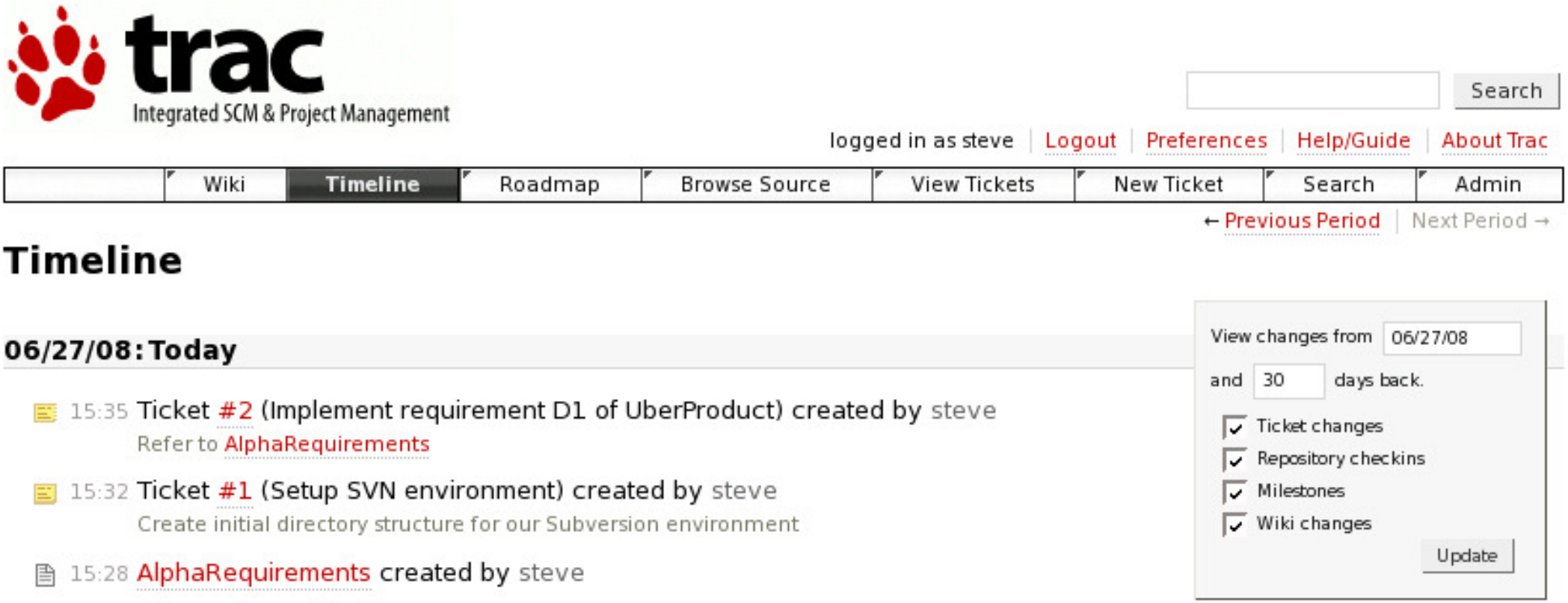
"trac-admin *yourenvdir* initenv" created a new Trac environment, containing a default set of wiki pages and some sample data. This newly created environment also contains documentation to help you get started with your project.

You can use trac-admin to configure ⇨ Trac to better fit your project, especially in regard to *components*, *versions* and *milestones*.

TracGuide is a good place to start.

Enjoy!

Trac Timeline



Trac Roadmap

Search

logged in as steve | Logout | Preferences | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search | Admin

Roadmap

Show already completed milestones

Update

Milestone: Alpha Release of Uber Project

Due in 5 weeks (08/04/08)

Needs to meet all of the specifications for UberProduct defined under AlphaRequirements

Milestone: Beta 1 of UberProduct

Due in 3 months (10/01/08)

Must hit the first beta by 1st October if UberProduct is to make it into the shops for Christmas

Add new milestone

Trac Tickets

- Capture all of your work items.
- Reasonably standard set of fields
  - Type - e.g. defect, enhancement or task
  - Component - The project module or subsystem
  - Priority - The importance of this bug, task etc.
  - Milestone – Based on Roadmap entries
  - Assigned to - Principal person responsible for ticket
  - Summary – Single line brief description of the ticket
  - Description – Make use of TracWiki syntax

Trac New Ticket

Search

logged in as steve | Logout | Preferences | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search | Admin

Create New Ticket

Properties

Summary: Implement requirement D1 of UberProduct

Reporter: steve

Description:

Refer to AlphaRequirements

Trac New Ticket

Assign to:		Type:	task
Priority:	critical	Milestone:	Alpha Release of Uber Project
Component:	UberProduct	Version:	
Keywords:		Cc:	

☐ I have files to attach to this ticket

Preview Create ticket

Note: See TracTickets for help on using tickets.

Trac Completed Ticket

trac
Integrated SCM & Project Management

Search

logged in as steve | Logout | Preferences | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search | Admin

← Previous Ticket | Next Ticket →

Ticket #2 (new task)

Implement requirement D1 of UberProduct

Opened 2 seconds ago

Reported by:	steve	Owned by:	steve
Priority:	critical	Milestone:	Alpha Release of Uber Project
Component:	UberProduct	Version:	
Keywords:		Cc:	

Description

Refer to AlphaRequirements

Reply

Attachments

Attach file

Version Control

- Doesn't have an integrated version control tool.
- Leverage's Subversion
- Support for other Version Control tools in development

  http://trac.edgewall.org/wiki/VersioningSystemBackend
- Excellent web based browser and diff tool for Subversion

Trac – User Management

- Work out how you want to organise your team(s)
- Try to start with a clean set of permissions
- Assign permissions to groups, and then assign your team to the groups
- Covered in detail at http://trac.edgewall.org/wiki/TracPermissions

User permissions

Manage Permissions

Subject	Action	
administrator	☐ PERMISSION_ADMIN	☐ REPORT_ADMIN
	☐ ROADMAP_ADMIN	☐ TICKET_ADMIN
	☐ WIKI_ADMIN	
agr0	☐ administrator	
anonymous	☐ BROWSER_VIEW	☐ CHANGESET_VIEW
	☐ CONFIG_VIEW	☐ EMAIL_VIEW
	☐ FILE_VIEW	☐ LOG_VIEW
	☐ MILESTONE_VIEW	☐ REPORT_SQL_VIEW
	☐ REPORT_VIEW	☐ ROADMAP_VIEW
	☐ SEARCH_VIEW	☐ TICKET_VIEW
	☐ TIMELINE_VIEW	☐ WIKI_VIEW
authenticated	☐ REPORT_SQL_VIEW	☐ REPORT_VIEW
	☐ TICKET_CREATE	☐ TICKET_MODIFY
	☐ WIKI_CREATE	☐ WIKI_MODIFY
prohpcf1	☐ TRAC_ADMIN	

Ticket Types and Components

- Default Types are development focused
- Defect, enhancement and task
- Default Components are simple examples and should be replaced
- Components allow auto assignment of new tickets to team members
- Effective use means simple reports can be easily generated

Using Trac Effectively

- TracLinks allows seamless linking between tickets, the wiki and subversion
- Wiki pages should use CamelCase where possible or [wiki:Page] where this isn't appropriate
- Tickets can be referenced via #number or [ticket:number] e.g. #27 or [ticket:27]
- Subversion change sets can be referenced by revision number e.g. r21 or [changeset:21]
- You can link to a specific location with your Subversion repository via source:/path e.g.

```
[source:/trunk/project/documentation/Readme]
```

TracLinks Examples

- A new ticket 91 with the following description:

```
Build additional Apache virtual server for the
WebApplication team based off environment developed for
ProjectPurple in #57
```

  - Documentation is auto referenced into the Wiki
    - Keep track of the details on ProjectPurple
    - References to contacts for the WebApplication team.
- The Apache configuration should be kept under version control with an appropriate commit message.

```
Apache configuration for additional ProjectPurple virtual
instance - see ticket #91 and #57
```

Subversion post-commit hook

- Highly recommend development teams to utilise the trac-post-commit-hook add-on
- Installation details covered in the Trac FAQ

  http://trac.edgewall.org/wiki/TracFaq
- Auto-updates Trac tickets by using a simple syntax in Subversion commit messages
  - closes #ticket – Marks ticket closed with comment
  - refs #ticket – Just adds comment to ticket

Sub Tickets

- Break out larger tasks into logical items of work
  - No inbuilt method of generating sub-tickets
  - MasterTicketsPlugin from TracHacks website -
    - Adds a custom field to all tickets that can point at a parent
    - Parent and child can see connection
    - Parent cannot be closed until all children are also closed
  - Easy to define a manual process for handling sub-tickets, but plug-in simplifies process

Tag milestones on release

- Tag each release in subversion as part of closing a Roadmap milestone
- Tags are cheap in Subversion – use them
  - Make sure all changes are committed to Subversion
  - Create a Subversion tag based on your release

```
svn copy http://trac.ourcompany.org/svn/MyProject/trunk \
http://trac.ourcompany.org/svn/MyProject/tags/release-1 \
-m "First Release milestone"
```

  - Link milestone description to revision

```
Release tagged in r81
```

  - Close the milestone and re-assign any open tickets

WikiMacros

- Covered in detail in integrated documentation
- [[PageOutline]]
  - Table of contents of a wiki page based on headings.
  - A must have once a wiki entry exceeds a page
- [[Image]]
  - Provides control when embedding images
- InterTrac and InterWiki
  - Rapid links to other sources such as MythTV Trac or Wikipedia, e.g. [mythtv:ticket]

Don't go plug-in crazy

- Lots of cool stuff on TracHacks
  - Do you really need the plug-in?
  - How well supported is it?
  - Will it work in the next release of Trac?
- Make sure you test in a sandbox environment
- Same rules apply to adding additional Macros

Links and References

- Edgewall
  - http://www.edgewall.org/
  - http://trac.edgewall.org/
- TracHacks
  - http://trac-hacks.org/wiki/TracHacks

Trac at HPC-Forge

https://hpc-forge.cineca.it/trac

```
trac
    project01
    project02
    project03
```

Anonymous access to a project is denied by default but can be enabled through the admin panel of Trac.
Fine grained permissions can be specified on any kind of Trac resources, even at the level of specific versions of such resources.
Every project can handles more than one repository.

Available Projects

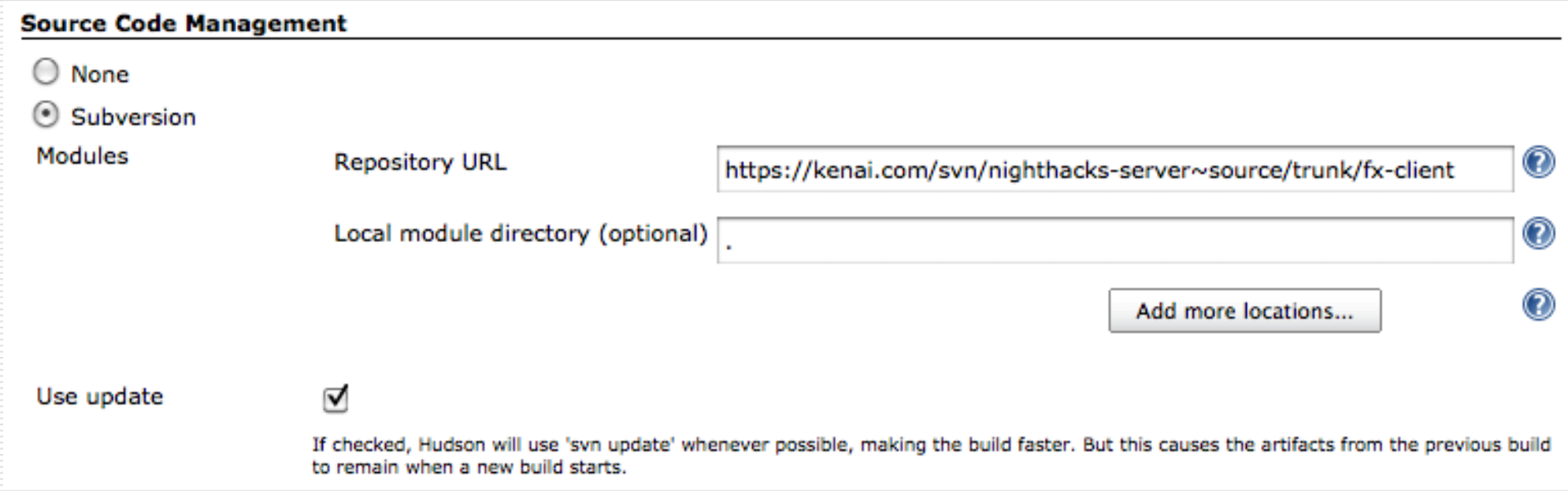
- Applications Statistics and Characteristics
- aspy
- Async IO
- Cosmo
- CRS
- DVA
- FARM
- Gosplan
- iittrac
- OGSBIO
- prova
- PSPI
- RiTMo
- RTM
- Sandbox
- SEBE3
- SimbaGE+
- SimbaGE1D
- viz-simula

What is Hudson?

- Hudson is an open source “continuous
- integration” (CI) server. A CI server can do
- various tasks like
- ●check-out source code
- ●build and test the project
- ●publish the results
- ●communicate the results to team members
- and much more ..

Configuring the Job

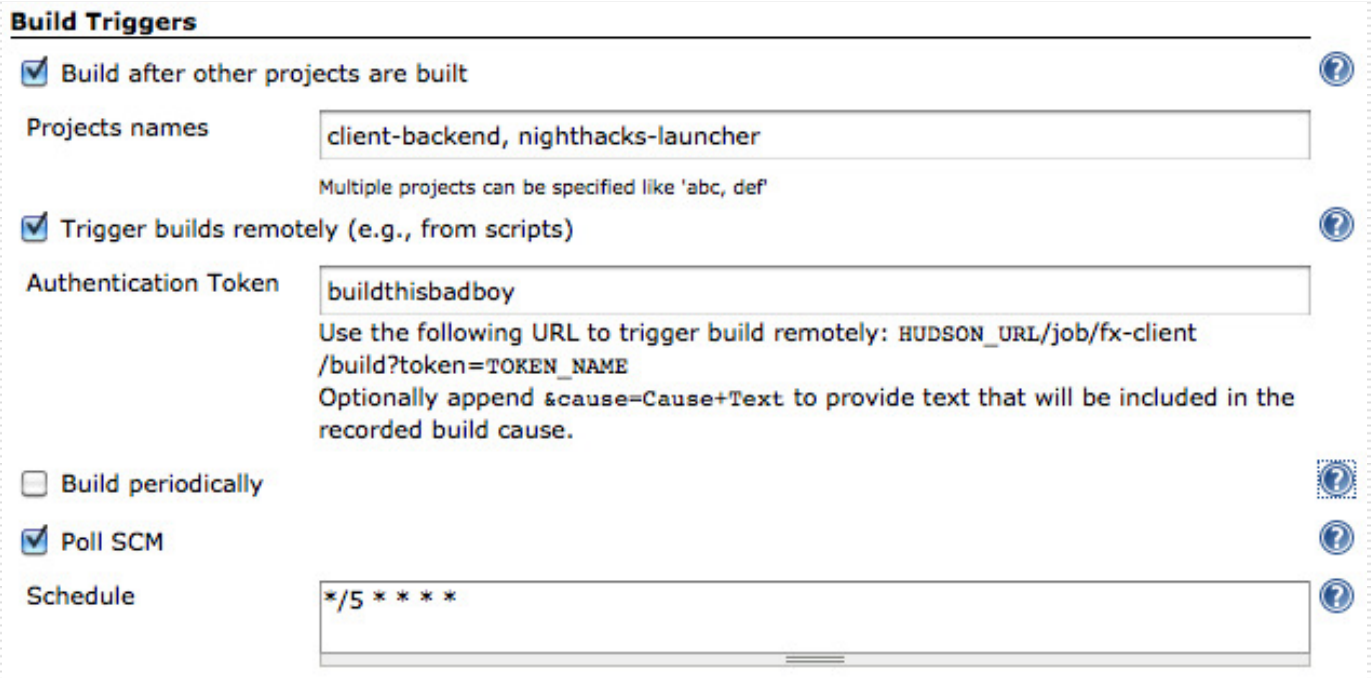
If the job is for building a project sources, we must provide a Source Code Management information from where the sources can be downloads. By default only CVS and Subversion are supported. But plugins are available for other SCM such as Clearcase, Git, perforce, mercurial, VSS, accirev, tfs etc.



Configuring the Job Continued..

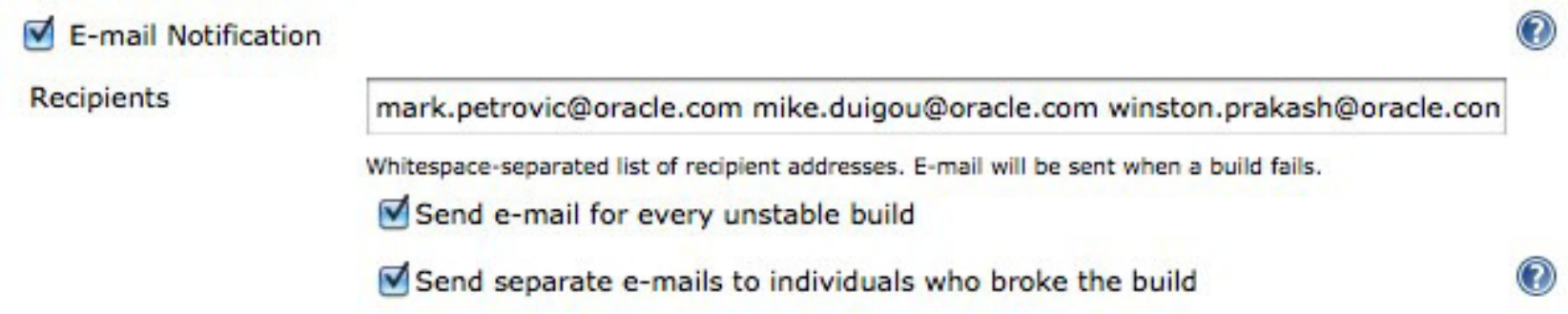
Next we must specify when the build should get triggered. An obvious choice for software project can be when somebody checked into SCM.

In the following example, SCM is polled for every 5 minutes to see if any new checkin has happened. Optionally it is possible to make the current project to build after other projects are built.



Configuring the Job Continued...

- Another important option is to tell Hudson whom to send e-mail when the builds become unstable.



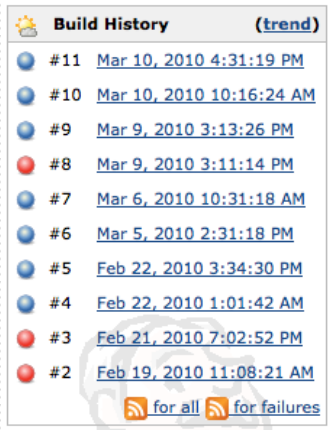
There are several other options
- Deploy War after build
- Invoke post batch jobs after build completes
- Archive the artifacts associated with build
- Update relevant JIRA Issue
- Plot build data

to name a few.

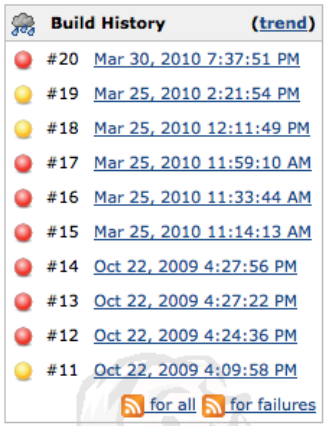
Hudson is ready to Build

- That's all. Hudson is ready for Continuous integration. Automatically builds will get triggered when ever someone checked in to the SCM.

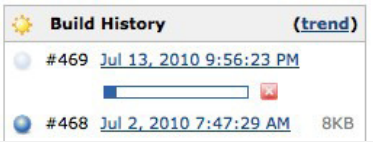
When a build is successful it is indicated by a blue ball. A red ball denotes a failed build.



Unstable (Ex. Failed test) builds are indicated by Yellow ball.

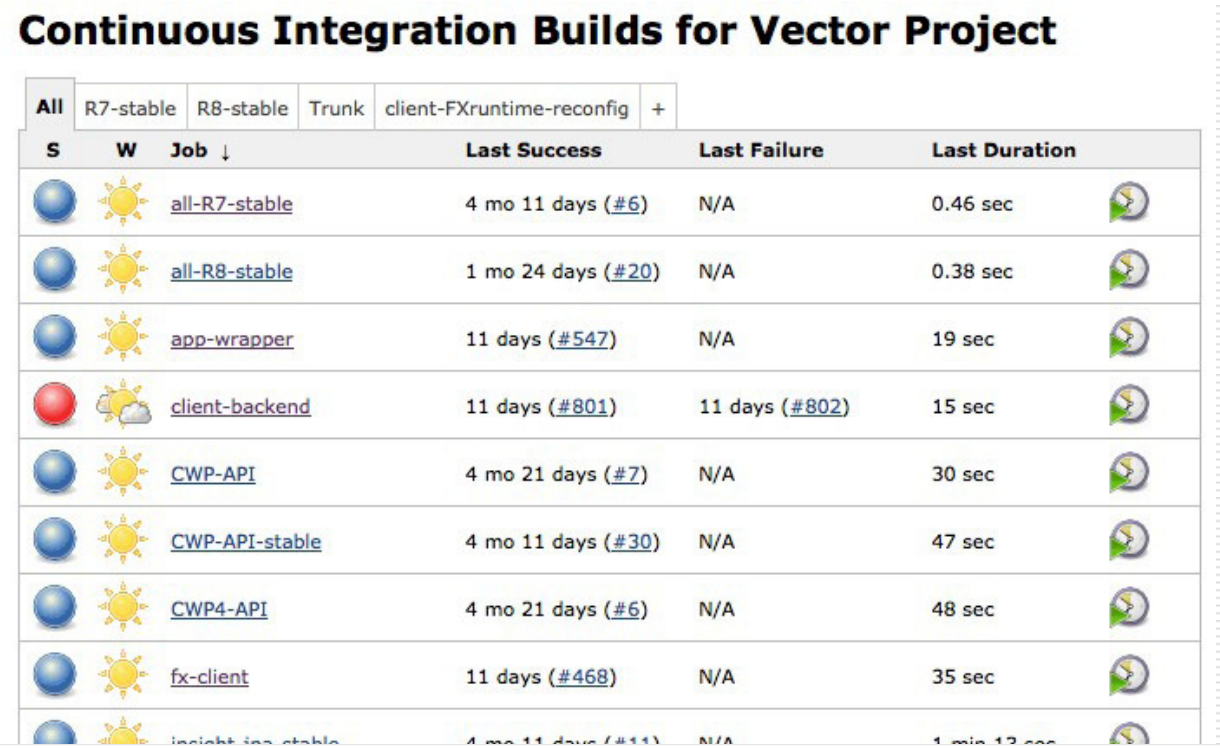


If the ball is blinking, then it represents an ongoing build.



Hudson Main Dashboard

Hudson Main Dashboard provides a summary view of all the projects (jobs). Hudson also provide a way to tag the jobs to different views, so that it makes it easier to list the view by milestone or by other criteria.



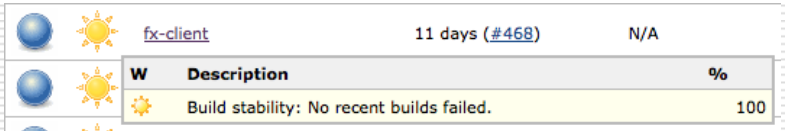
Continuous Integration Builds for Vector Project

All | R7-stable | R8-stable | Trunk | client-FXruntime-reconfig | +

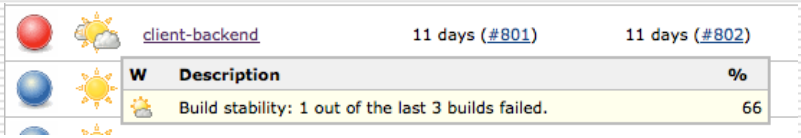
S	W	Job ↓	Last Success	Last Failure	Last Duration
		all-R7-stable	4 mo 11 days (#6)	N/A	0.46 sec
		all-R8-stable	1 mo 24 days (#20)	N/A	0.38 sec
		app-wrapper	11 days (#547)	N/A	19 sec
		client-backend	11 days (#801)	11 days (#802)	15 sec
		CWP-API	4 mo 21 days (#7)	N/A	30 sec
		CWP-API-stable	4 mo 11 days (#30)	N/A	47 sec
		CWP4-API	4 mo 21 days (#6)	N/A	48 sec
		fx-client	11 days (#468)	N/A	35 sec

Build Stability

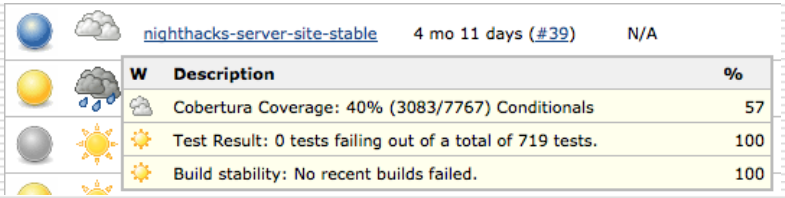
Highly Stable



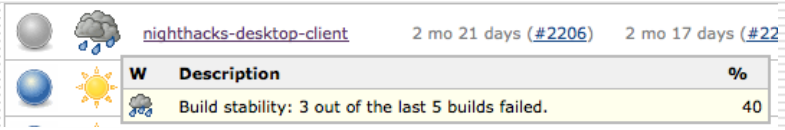
Slightly Unstable



Unstable



Highly Stable



Project Relationship

When you have projects that depend on each other,Hudson can track which build of the
upstream project is used by which build of the downstream project

The project relationship is accomplished by the conditions

- The upstream project records the fingerprints of its build artifacts
- The downstream project notes the fingerprints of the upstream jar files it uses

Fingerprints

The fingerprint of a file is simply a MD5 checksum. Hudson maintains a database of md5sum. For each md5sum, hudson maps it to a project and corresponding build. These files are stored at $HUDSON_HOME/fingerprints.

Project Relationship is maintained by
- jar files that your Upstream project produces.
- jar files that your dependent (downstream) project rely on.

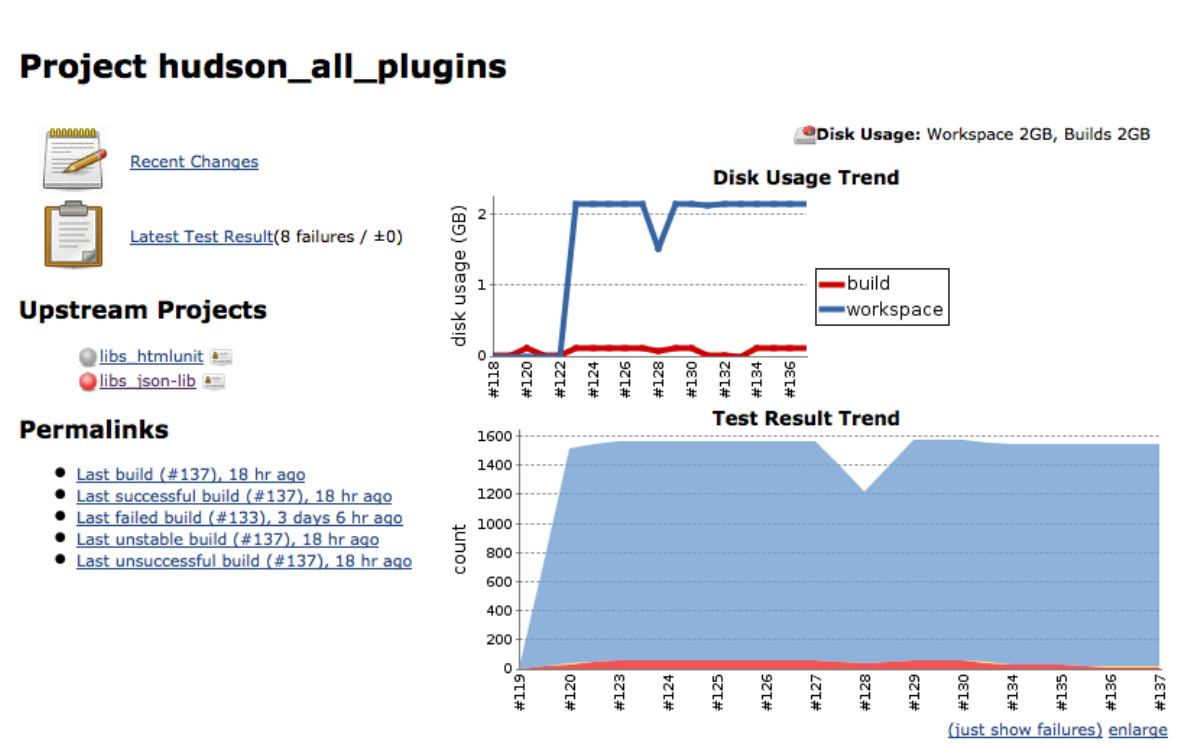
Suppose there are two projects TOP and BOTTOM project and assume TOP depends on BOTTOM. You are working on the BOTTOM project. The TOP team reported that bottom.jar that they are using causes an NPE, which you thought you fixed in BOTTOM #32. Hudson can tell you which TOP builds are using (or not using) your bottom.jar #32 via fingerprints.

Project Dashboard

The Dashboard for particular project provides view for:

- Last Successful Build Info
- Latest Test Result
- Monitoring Disk Usage
- Actions like configuring the job etc
- Test Result Trend
- Recent changes that caused the build

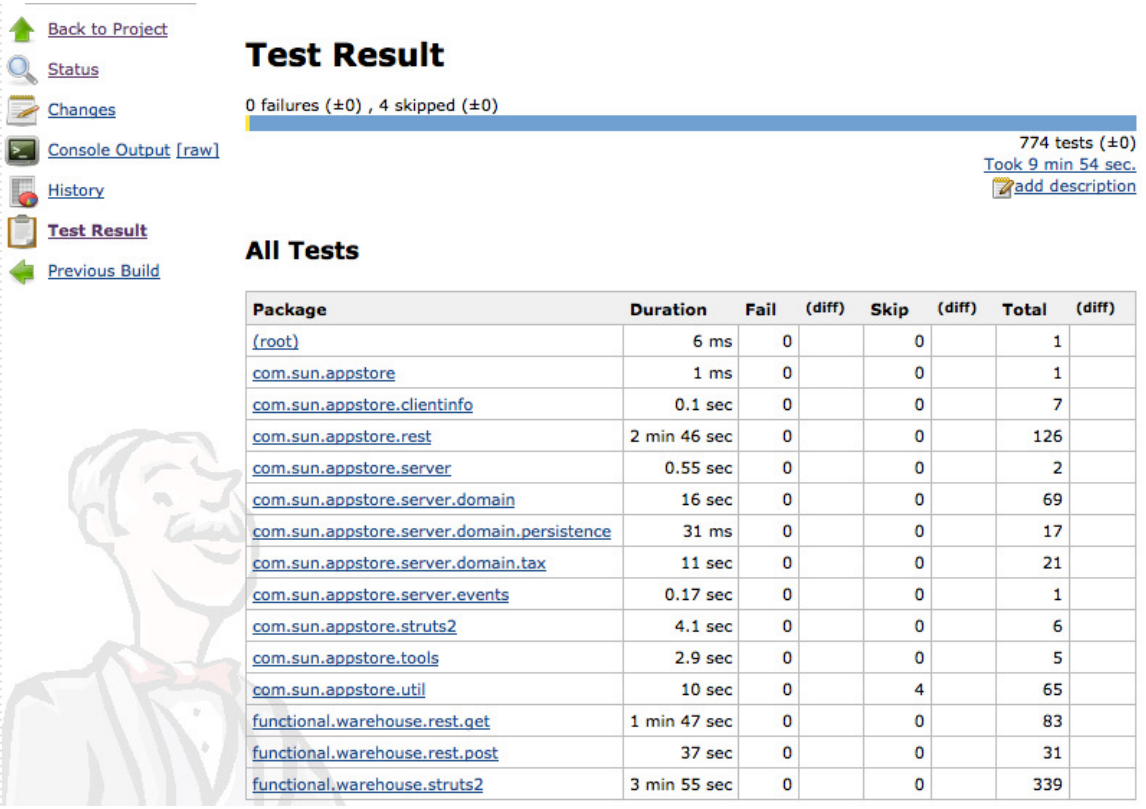
Various views in the project dashboard depends on various plugins installed.



Build Dashboard

The Dashboard for a particular Build provides view for

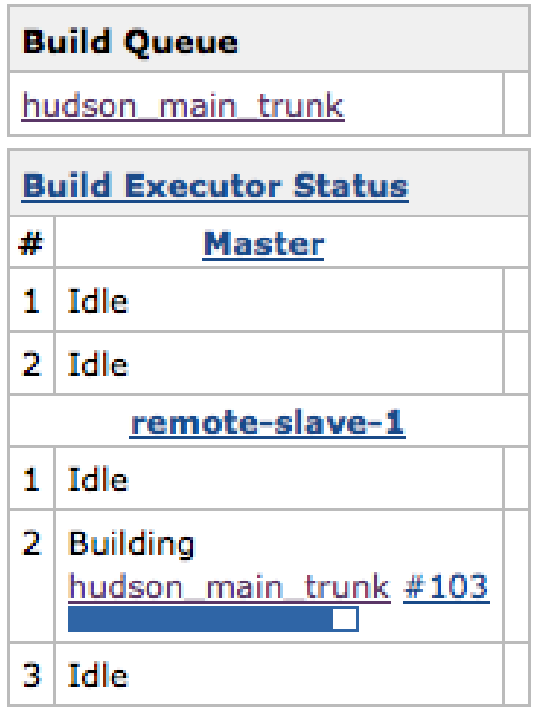
- Artifacts corresponding to this Build
- Changes that caused this Build
- Test Results
- Build console output



Package	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
(root)	6 ms	0		0		1	
com.sun.appstore	1 ms	0		0		1	
com.sun.appstore.clientinfo	0.1 sec	0		0		7	
com.sun.appstore.rest	2 min 46 sec	0		0		126	
com.sun.appstore.server	0.55 sec	0		0		2	
com.sun.appstore.server.domain	16 sec	0		0		69	
com.sun.appstore.server.domain.persistence	31 ms	0		0		17	
com.sun.appstore.server.domain.tax	11 sec	0		0		21	
com.sun.appstore.server.events	0.17 sec	0		0		1	
com.sun.appstore.struts2	4.1 sec	0		0		6	
com.sun.appstore.tools	2.9 sec	0		0		5	
com.sun.appstore.util	10 sec	0		4		65	
functional.warehouse.rest.get	1 min 47 sec	0		0		83	
functional.warehouse.rest.post	37 sec	0		0		31	
functional.warehouse.struts2	3 min 55 sec	0		0		339	

Distributed Building

Hudson supports the "master/slave" mode for distributed building.
Additional workload of building projects are delegated to multiple "slave" nodes
Provides different environments needed for builds/tests (Unix/Windows/Linux/Mac)
Master is an installation of Hudson. It serves all HTTP requests, and it also builds projects on its own.
Slaves are computers that are set up to build projects for a master. Hudson runs a separate program called **slave agent** on slaves. Master starts these slave agents on demand.



Popular Competitive Offerings

- **Apache Continuum** — continuous integration server supporting Apache Maven and Apache Ant (open source)
- **Bamboo** — commercial continuous integration server by Atlassian Software Systems
- **CruiseControl** — Java-based framework for a continuous build process (open source)
- **TeamCity** — commercial continuous-integration server by JetBrains.
- **Team Foundation Server** — commercial continuous integration server and source code repository by Microsoft
- **Tinderbox** — Mozilla-based product (open source)
- **Rational Team Concert** — commercial software development collaboration platform by IBM

Hudson at HPC-Forge

Anonymous access to a project is denied by default but can be enabled through the administration panel.
A project can be linked to a specific slave; a login before job running is the best practice.

What's WebDAV

1. **WebDAV** is an abbreviation of Web-based Distributed Authoring and Versioning, which refers to both an IETF working group and the set of extensions to the HTTP protocol that the group defined, which allows users to collaboratively edit and manage files on remote web servers.
2. Its aim is to provide the functionality to create and manage documents on a web server. The obvious use for this is for authoring and publishing the documents that a web server serves, but it can also be utilized for general web-based file storage that is accessible from anywhere. Support for WebDAV is provided by most modern operating systems, and with the right client and a fast network it can be almost as easy to use files on a WebDAV server as those stored in local directories.

WebDAV contents

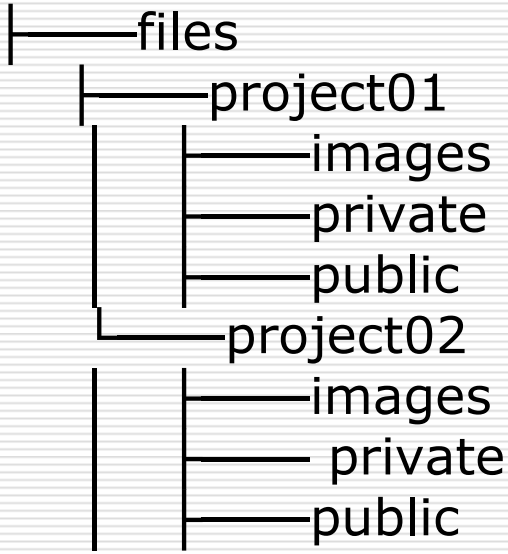
- Our use for WebDAV is to store:
  - 'publish' releases and files for download;
  - non-wiki documents;
  - images to be included in wiki pages without using attachments.

Common WebDAV clients

Software	Type	Windows	Mac	Linux	Description
cadaver	Standalone WebDAV application		X	X	Command-line WebDAV client supporting file transfer, tree, and locking operations
DAV Explorer	Standalone WebDAV application	X	X	X	Java GUI tool for exploring WebDAV shares
Microsoft Web Folders	File-explorer WebDAV extension	X			GUI file explorer program able to perform tree operations on a WebDAV share
davfs2	WebDAV filesystem implementation			X	Linux filesystem driver that allows you to mount a WebDAV share
AnyClient	Free FTP client software with support for FTP/S, SFTP and WebDAV protocols	X	X	X	available both as a web based service requiring no software installation, and as a downloadable application

WebDAV at HPC-Forge

https://hpc-forge.cineca.it/files



Images and public are accessible anonymously. Just private requires authentication. Directory structure can't be modified.

Suggested Reading

- Managing Software Development with Trac and Subversion: Simple project management for software development.
- Continuos Integration with Hudson
- Version control with Subversion