The Drizzle Contributor Tutorial

Jay Pipes
jaypipes@gmail.com
http://joinfu.com

what we'll cover today

- Being a Drizzle contributor
- The Drizzle source code
- Walk-through of a SELECT statement
- Plugin development tutorial

Being a Drizzle Contributor

being a Drizzle contributor

- About the Drizzle community
- IRC and the mailing lists
- Using Launchpad
- Using Bazaar

some things to remember...

- No blame
- No shame
- Be open and transparent
- Learn something from someone? Pass it on!
  - By adding to the wiki (http://drizzle.org/wiki/)
  - By sharing it with another contributor
  - By blogging about it
  - By posting what you learn to the mailing list
- *There is no such thing as a silly question*

IRC and the mailing list

- IRC is often best way to get in touch
  - irc.freenode.net #drizzle
- Mailing list for longer discussions
  - Signup here: http://launchpad.net/~drizzle-discuss
  - Archive: https://lists.launchpad.net/drizzle-discuss/

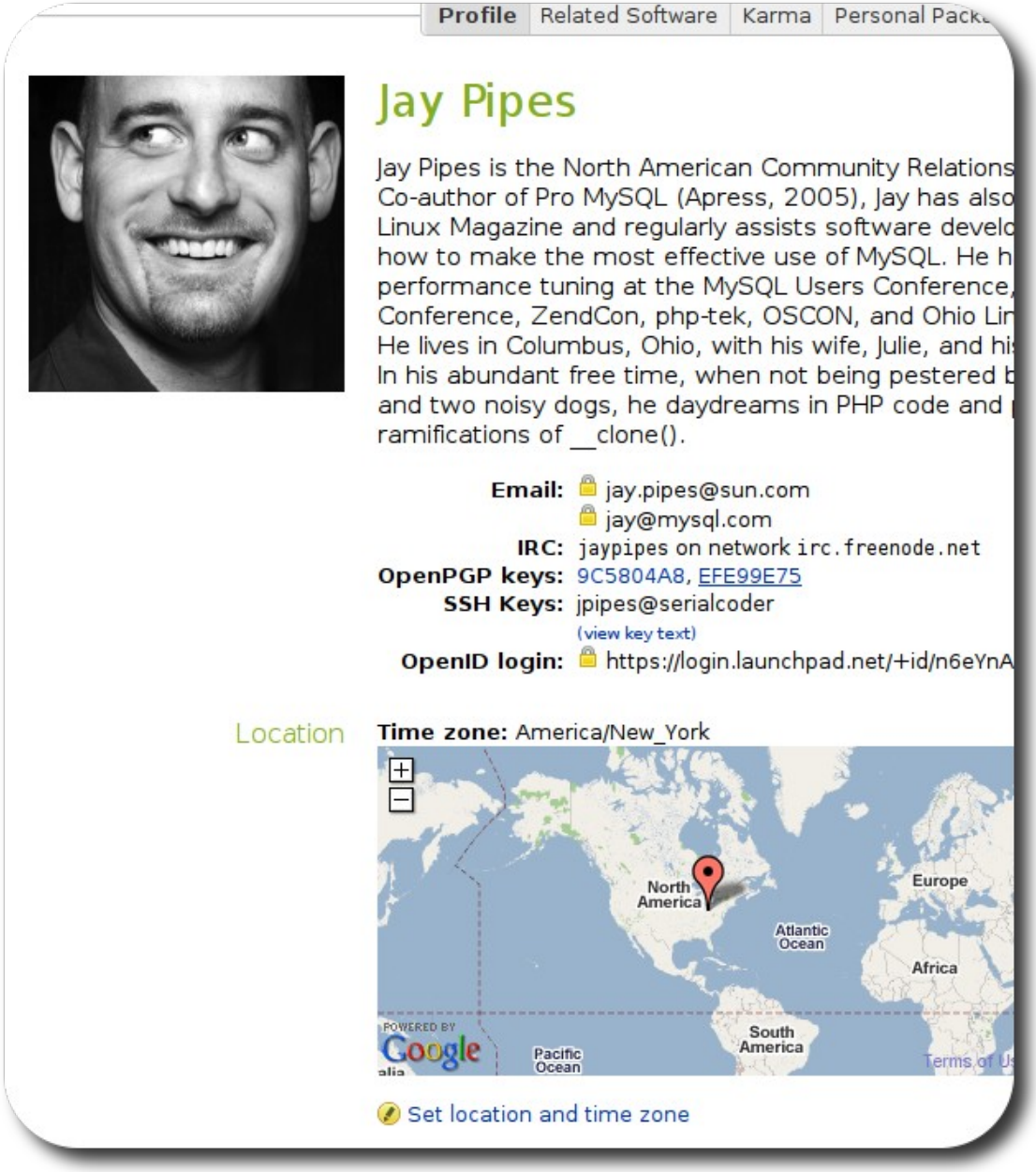
NO TROLLS.

about launchpad

- Written by the folks at Canonical/Ubuntu
- Online platform for project and code management
- Features
  - Source control services
  - Translation services
  - Bug tracking
  - Task management
  - Answers/FAQ functionality

your launchpad home page

- Profile
- Location
- SSH public keys
- Team memberships
- Most active projects
- Karma



https://launchpad.net/~jaypipes

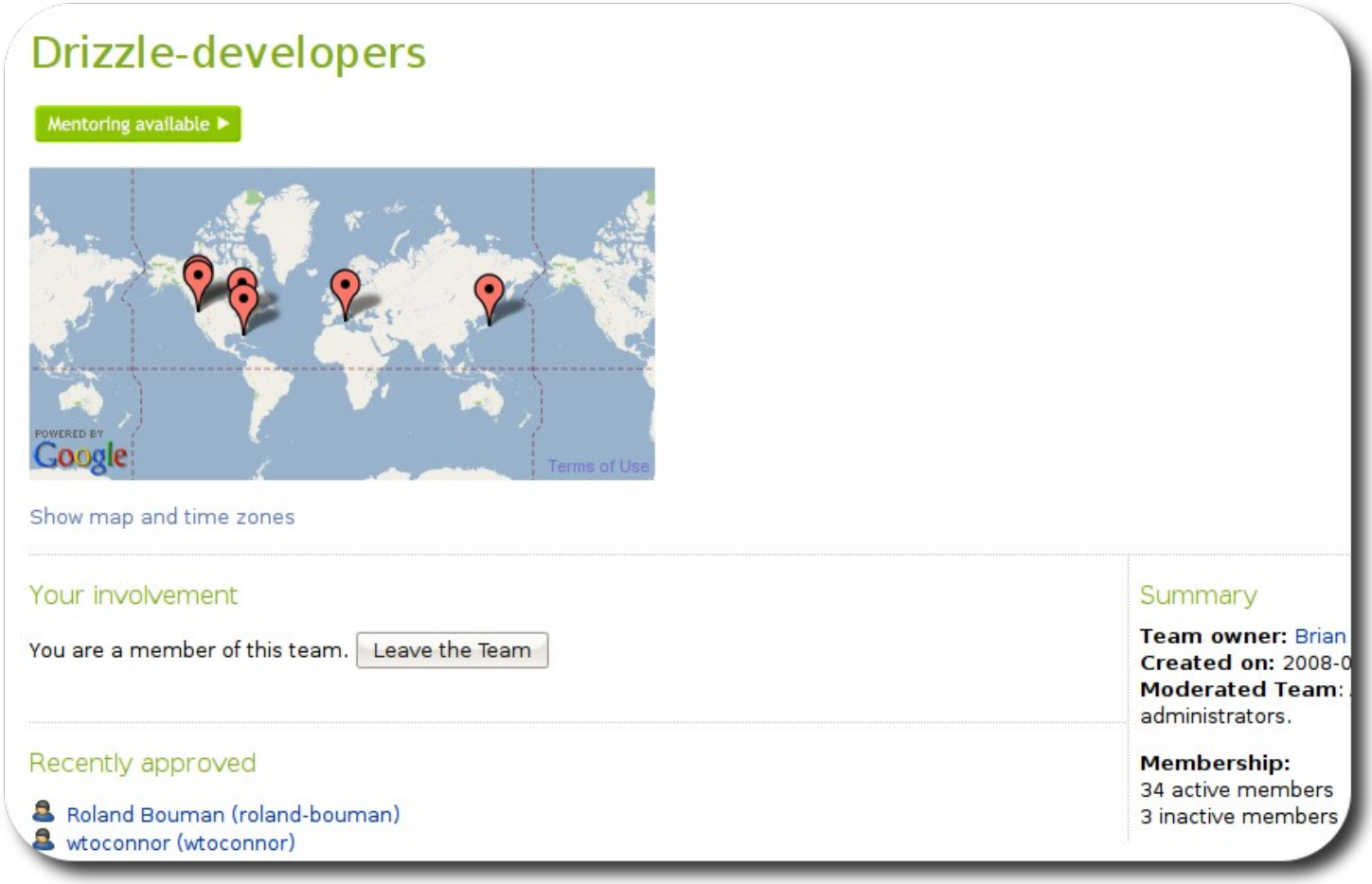
projects on launchpad

- Each project has a home page, shown to the right
- Main project team is listed as the “driver”

https://code.launchpad.net/drizzle

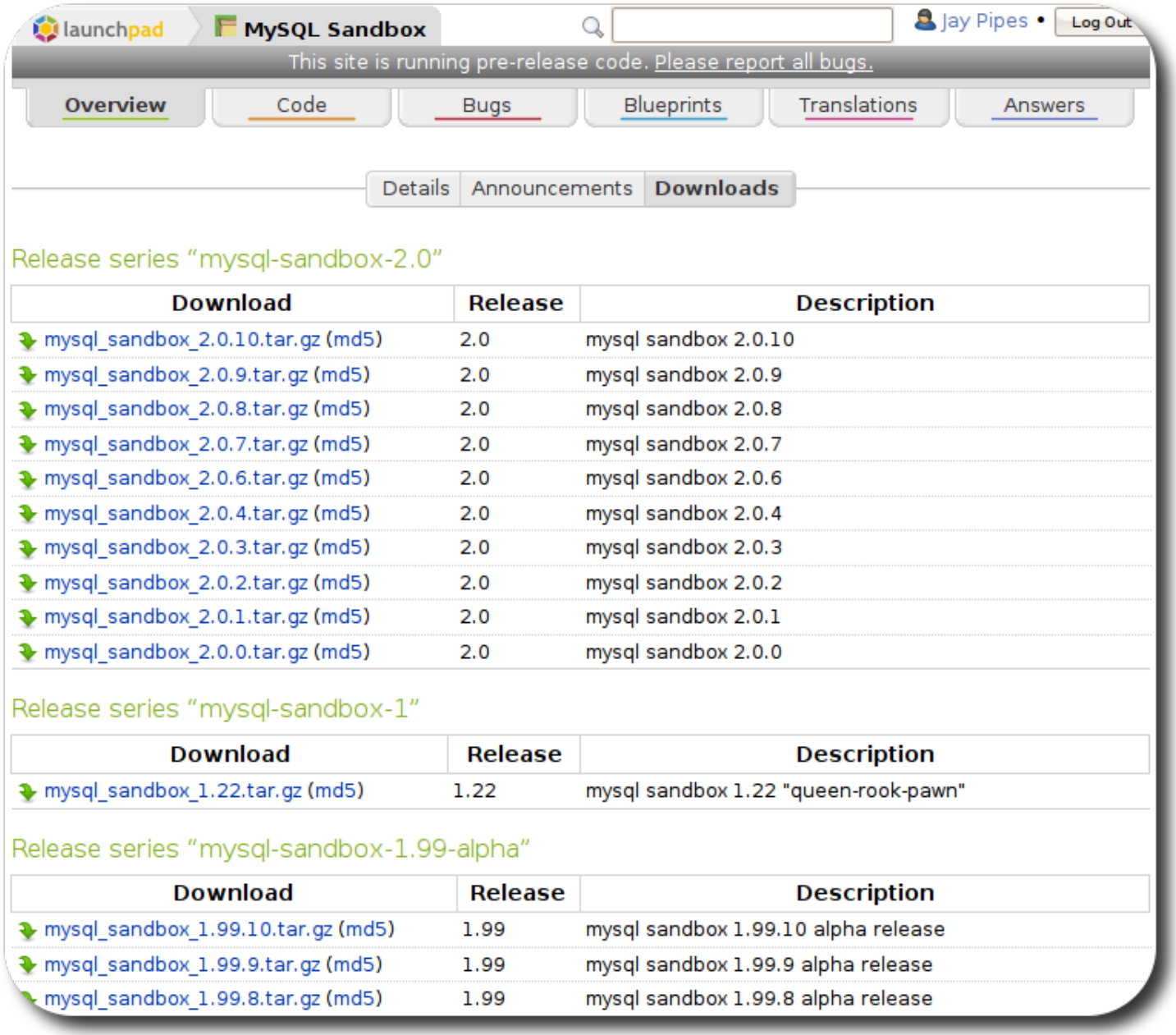
project teams on launchpad

- To join the team, click the team link, then “Join this team”



https://launchpad.net/~drizzle-developers

project downloads



https://edge.launchpad.net/mysql-sandbox/+download

code management on launchpad

Bazaar branches of Drizzle

Register a branch

97 branches owned by 26 people and 1 team, 198 commits by 14 people in the last month

2 active reviews, 0 approved merges

You can browse the source code for the development focus branch or get a copy of the branch using the command:

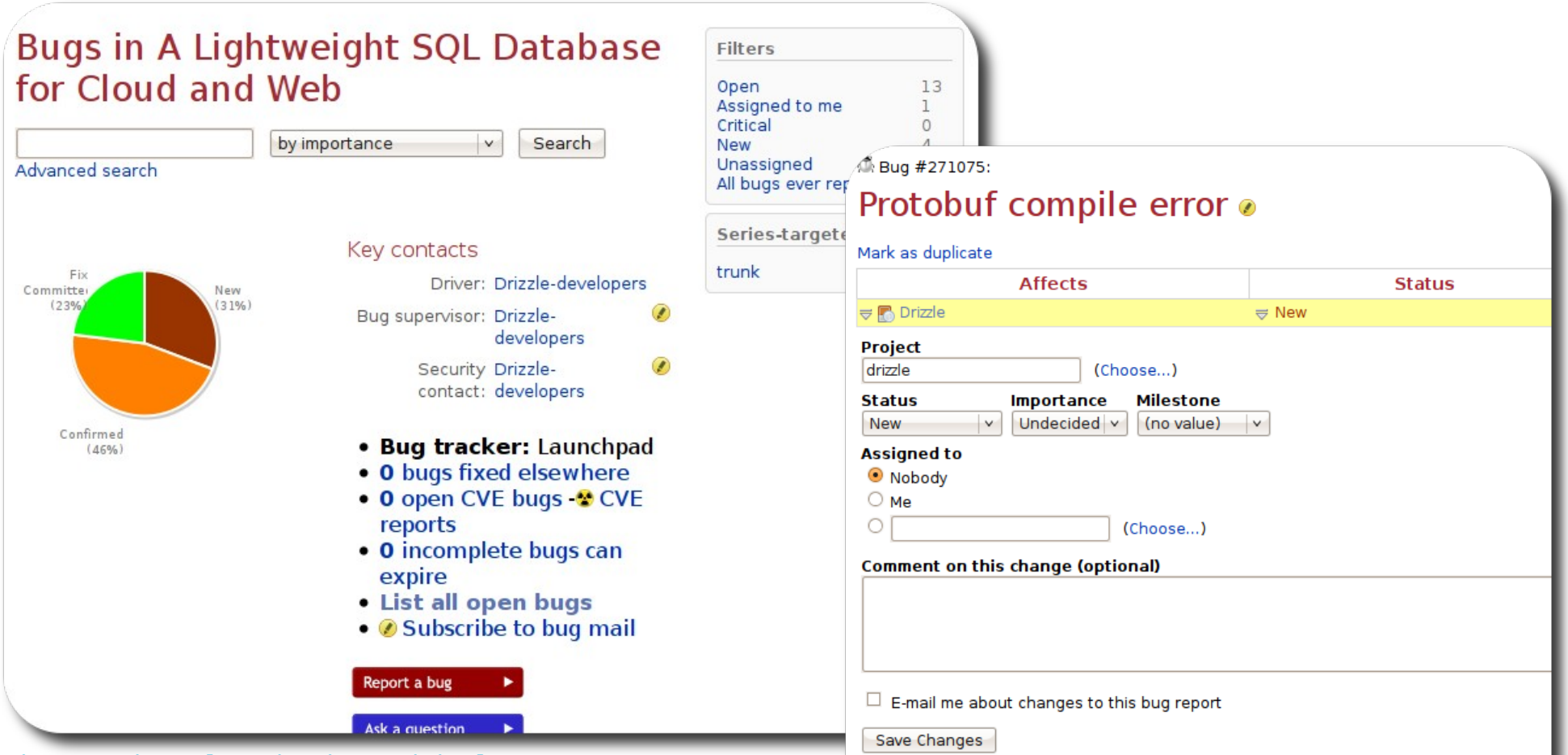
```
bzr branch lp:drizzle
```

Branches with status: Any active status | by most interesting

Name	Status	Last Modified	Last Commit
lp:drizzle **Series:** trunk	New	4 days ago	395. Fixed uint/ushort issue in libdrizzle
lp:~jaypipes/drizzle/proto-definitions	New	2 hours 30 minutes ago	373. Added FieldDefinition class, which is...
lp:~mordred/drizzle/codestyle	New	11 hours ago	399. Renamed max/min to cmax/cmin in more ...
lp:~andrey-zhakov/drizzle/drizzle-virtual-columns	Development	12 hours ago	410. Disabling test vcol_blackhole
lp:~mordred/drizzle/split-and-rename-client	New	4 days ago	395. Fixed uint/ushort issue in libdrizzle
lp:~grant-glsoftware/drizzle/datetime-changes	Experimental	4 days ago	381. merge from trunk
lp:~ro4tub/drizzle/main	New	4 days ago	*This branch has not been pushed to yet.*
lp:~fallenpegasus/drizzle/pluglog	New	7 days ago	384. add pluggable logging
lp:~mkindahl/drizzle/stdize-code	Experimental	7 days ago	330. Merging stdize-code with main trunk.
lp:~drizzle-developers/drizzle/enable-tests	New	8 days ago	413. Added func_length and func_str to tes...
lp:~mordred/drizzle/drizzle-innodb-plugin	New	10 days ago	345. Fixed a few build errors.
lp:~petdance/drizzle/testing-overhaul	New	11 days ago	292. merging
lp:~patg-patg/drizzle/tests-addback	New	2 weeks ago	374. Added back tests, work in progress
lp:~grant-glsoftware/drizzle/drizzle-innodb-plugin	Development	3 weeks ago	344. OK I'm stuck

https://code.launchpad.net/drizzle

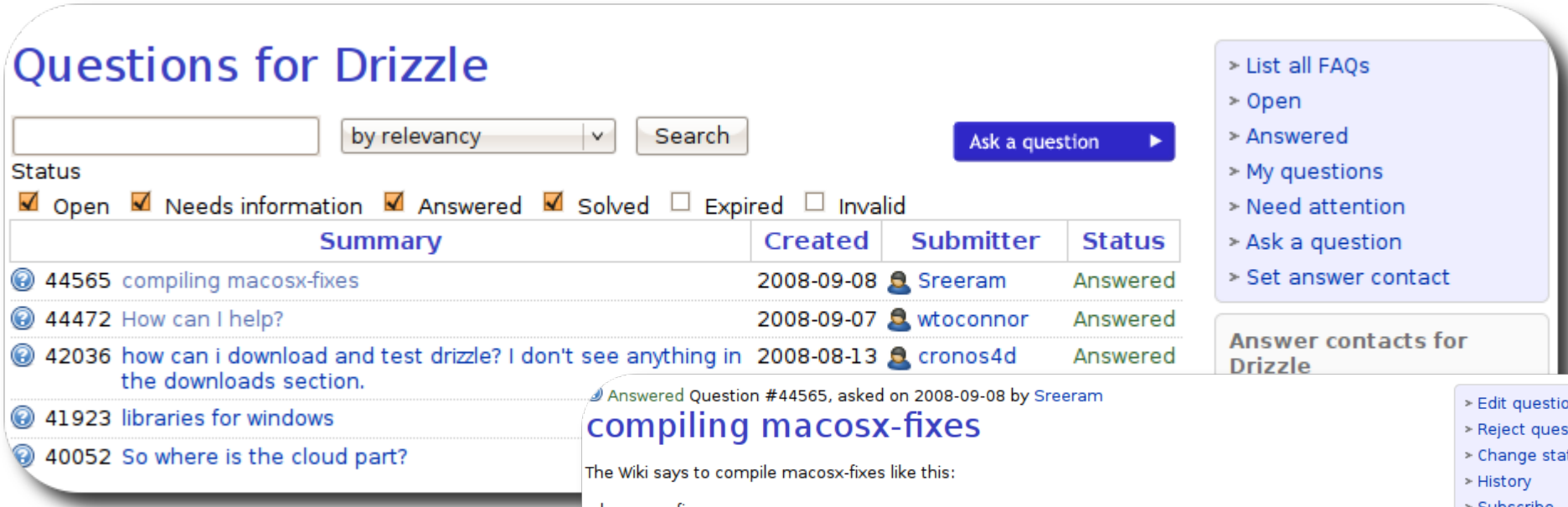
bug tracking on launchpad



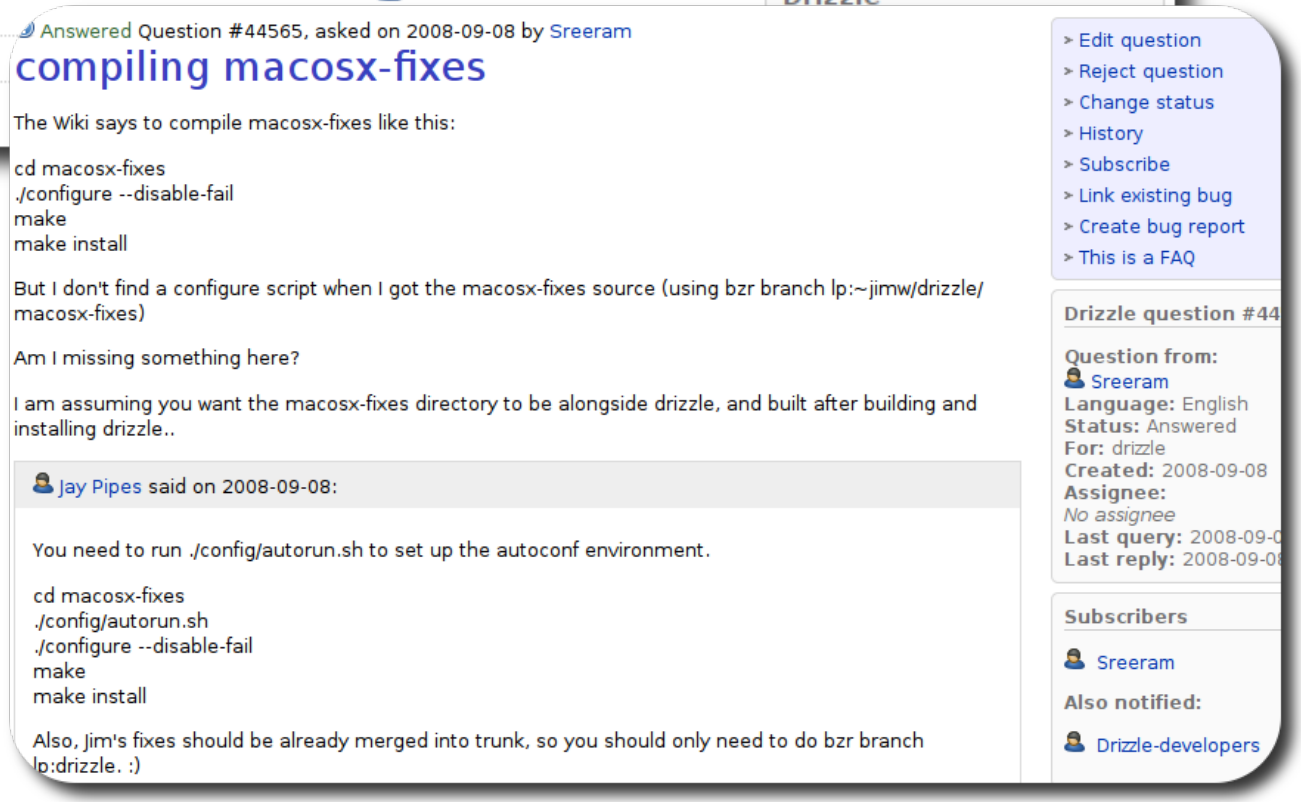
https://bugs.launchpad.net/drizzle

https://bugs.launchpad.net/drizzle/+bug/271075

answers on launchpad



https://answers.launchpad.net/drizzle



https://answers.edge.launchpad.net/drizzle/+question/44565

translations on launchpad

Translation status for drizzle

This template has no description, you can create one now.

Language	Status	Untranslated	Need review	Changed	Last Edited	By
Arabic		1383	—	—	2008-09-02	Fox Mind
Brazilian Portuguese		1139	—	1	2008-09-04	Henrique M. Decaria
Catalan		1362	—	—	2008-08-16	el_libre XDDDDDDDDDDD...
Chinese (Hong Kong)		1332	—	—	2008-08-16	Monty Taylor
Czech		1377	—	—	2008-09-12	Petr Pulc
Dutch		600	—	6	2008-09-11	Mark Daems
English (Australia)		1386	—	—	2008-08-16	Morgan Tocker
English (United ingdom)		1026	—	—	2008-09-14	Matthew Gadd

- Show translati
- Upload a file
- Download transla
- Change details

Template details

1387 messages
28 Languages
Owner: Monty Taylor
Created: on 2008
Domain: drizzle
Priority: 0
Translation Grou
None assigned
Permissions: Ope
Description:
None given

https://translations.launchpad.net/drizzle/trunk/+pots/drizzle

task management on launchpad

Blueprints for Drizzle

Show only blueprints containing: Search blueprints

Register a blueprint

- List all blueprints
- List documentation
- Assignments
- Register a blueprint

1 → 47 of 47 results

First • Previous • **Next** • Last

Priority	Blueprint	Design	Delivery	Assignee	Series
High	enable-passing-tests	Approved	Slow progress	Jay Pipes	
High	fix-and-enable-tests	Approved	Started	Jay Pipes	
High	fix-tests-syntax-failures	Approved	Started	Jay Pipes	
High	testing-overhaul	Approved	Started	Andy Lester	
High	split-and-rename-client	New	Started	Monty Taylor	
Medium	buildbot	Approved	Good progress	Ronald Bradford	
Medium	enable-disabled-tests	Approved	Slow progress	Jay Pipes	
Medium	remove-dead-code	Approved	Slow progress		
Medium	tests-refactor-runner	Approved	Started	Andy Lester	
Medium	new-discovery	Review	Started	Jay Pipes	
Low	doxygenation	Approved	Good progress	Jay Pipes	
Low	item-class-file-reorg	Approved	Blocked	Jay Pipes	
Low	use-replace-funcs	Approved	Unknown	Monty Taylor	
Low	rename-test-runner-mysql-to-drizzle	Pending Approval	Unknown		
Low	libicu-charsets	Discussion	Deferred		
Undefined	deb-package	Approved	Unknown	Monty Taylor	
Undefined	macport-package	Approved	Unknown		
Undefined	rename-mysql-to-drizzle	Approved	Good progress		
Undefined	style-cleanup	Approved	Good progress		

Latest blueprint

- Plug-in Service Registry
- Port Batch Key patches from 6
- Find occurrenc MySQL in test r and replace wi names
- Fix tests cases syntax failures changed data SQL mode
- Enable disable the test suite should be enab and/or fixes

https://blueprints.launchpad.net/drizzle

launchpad

about bazaar

- Written in C and Python
  - C modules for networking and I/O layers
  - Python for everything else

- Maintained by a small group of developers with community-driven roadmap
- Bazaar is released *every ~4 weeks*

installing bazaar on linux

- Ubuntu/Debian

```
$> sudo apt-get install bzr bzrtools meld
```

- Red Hat

```
$> su -c 'rpm -Uvh
  http://download.fedora.redhat.com/pub/epel/5/i3
  86/epel-release-5-2.noarch.rpm'
$> su -c 'yum install bzr'
```

- Bazaar Tools
  - GUI tools and useful plugins
- Meld is a graphical source conflict resolver

installing bazaar on windows and mac osx

- Windows:
  - Grab the installer from:

    http://bazaar-vcs.org/Download
  - Run it. :)
- Mac OSX
  - Either grab an image from:

    http://bazaar-vcs.org/Download
  - Or use MacPorts

    $> sudo port install bzr bzrtools meld

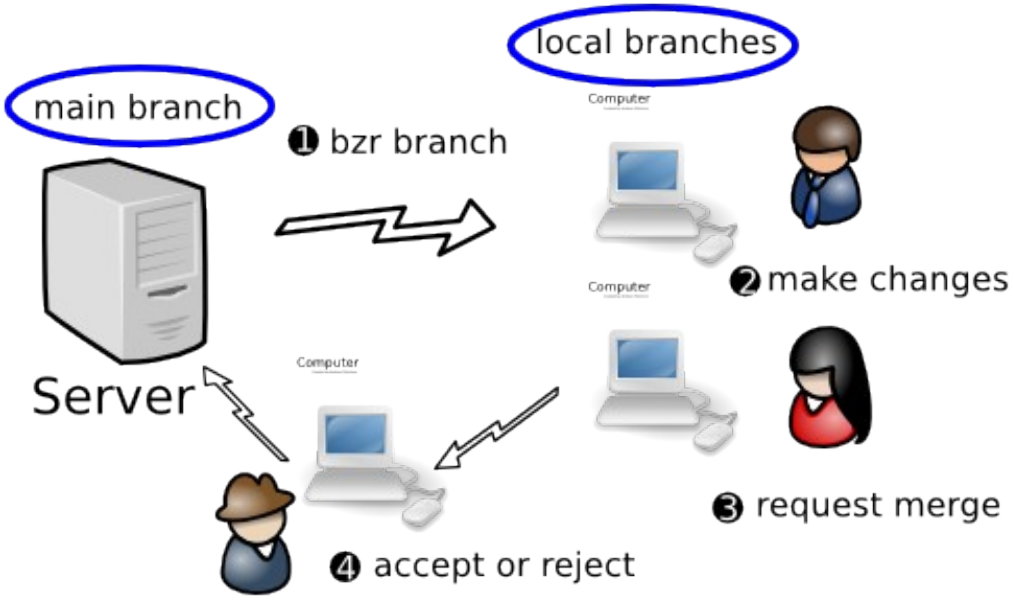
distributed source control concepts

- A *repository* is a collection of *branches*
  - But...there is no central server with single controlling repository
- Instead, developers work on their own branches of a project's code
- Code is maintained for a project by merging branches together
- The project's source tree is merely the collection of branches that contain source code

decentralized model with gatekeeper

1. Developer creates a local branch from some branch of the project
2. Developer makes changes to the local branch
3. Developer *commits* changes to the local branch
4. Developer *pushes* a changeset to another branch
5. Gatekeeper (or *merge captain*) reviews the pushed changeset
6. Gatekeeper merges changes into a branch which serves as a “trunk” or active development branch
7. Developer *pulls* from trunk branch to update locally

decentralized model with gatekeeper




```
$> bzr branch lp:drizzle trunk
$> vi somefile.cc
$> bzr commit somefile.cc -m “code change comment”
$> bzr push lp:~username/drizzle/reviews
$> bzr branch lp:~username/drizzle/reviews username-reviews
$> cd local-trunk && bzr merge ../username-reviews && bzr push
$> cd trunk; bzr pull
```

setting up a local development repository

```
$> mkdir ~/repos
$> cd repos; bzr init-repo drizzle
$> cd drizzle
$> bzr branch lp:drizzle trunk
```

- To pull in any changesets merged into the trunk branch, do this on a periodic basis:

```
$> cd ~/repos/drizzle/trunk; bzr pull
```

- Consider scripting the pull in cron

working on a bug locally

```
$> bzr branch trunk bug32124-crash-ps-var
$> cd bug32124-crash-ps-var
```

- Make code changes

```
$> bzr commit filename.cc # Repeat as needed
```

- Enter descriptive comments about the change in your editor and save

```
$> bzr push lp:~username/drizzle/reviews
```

proposing a branch for review

- In Launchpad.net, click “Propose for merging into another branch” as shown in the screenshot below:

https://code.launchpad.net/~drizzle-developers/drizzle/enable-tests

best practices for using launchpad and bazaar

- Use small, manageable “work units”
  - GOOD: “split and enable func_md5 plugin tests”
  - BAD: “refactor the server” :)
- Use the Blueprints system to relate common and dependent tasks
  - Very helpful in organizing larger blueprints with smaller, more specific subtasks
- Have a personal “reviews” branch
  - Good for small patch reviews for your own code
- Have specific branches for larger tasks

small work units

- A blueprint should be small and as descriptive as possible
- Blueprints can be created to group smaller tasks into a common “super-task”
- Use the “Add dependency” action to link a subtask to a super-task
- The super-task will show a graphical representation of the dependencies

a task with subtasks

Enable as many tests in the existing test suite as possible and fix failing tests

This is an over-arching task to encapsulate enabling passing, failing, and disabled tests in the Drizzle test suite

Set the URL for this specification

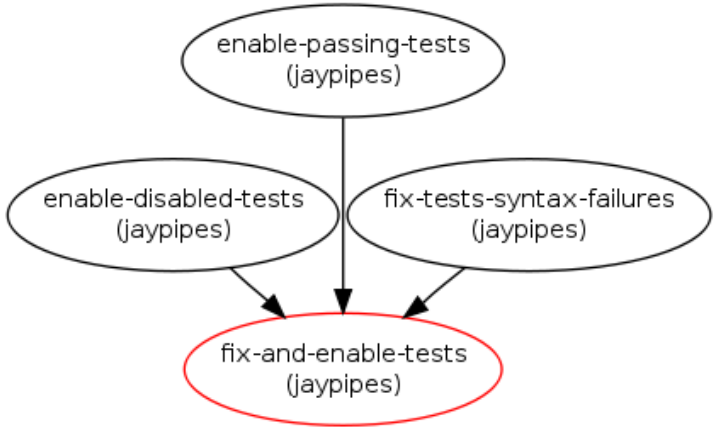
Milestone target: cirrus.

» Offer to mentor someone implementing this blueprint

Priority: High
Definition: Approved
Implementation: Started
Assignee: Jay Pipes
Drafter: Jay Pipes
Approver: Jay Pipes

- Edit title and summary
- Change people
- Change status
- Change priority
- Edit whiteboard
- Propose as goal
- Target milestone
- Request feedback
- Subscribe yourself
- Subscribe someone else
- Link to bug report
- Link to branch
- Offer mentorship
- Add dependency
- Remove dependency
- Show dependencies
- Propose for meeting agenda
- Mark superseded
- Retarget

Dependency tree



* Blueprints in grey have been implemented.

Lifecycle

Registered by: Jay Pipes
When: 2008-09-09
Started: 2008-09-09
by: Jay Pipes

Branches

enable-tests: (no title)

https://blueprints.launchpad.net/drizzle/+spec/fix-and-enable-tests

using milestones to provide direction

- A milestone is simply a container which acts as a “target” for developers
- You can link bugs, blueprints, and branches to a milestone
- Multiple milestones can exist simultaneously for a project

a milestone

Milestone cirrus for Drizzle

First stable release of the Drizzle Server.

Targeted features/functionality:

* Clean up of the code style
* Removal of custom libraries for regex, mysys, dbug
* Use of Google Proto Buffers in place of current discovery system and .frm files
* Overhaul of testing framework
* Standardization on stdint
* Removal of all compiler warnings

9 blueprints targeted

Specification	Priority	Assignee	Delivery
Enable as many tests in the existing test suite as possib...	High	Jay Pipes	Started
Fix tests cases with syntax failures due to changed data ...	High	Jay Pipes	Started
Overhaul the testing infrastructure	High	Andy Lester	Started
Re-enable passing tests for the Drizzle test suite	High	Jay Pipes	Slow progress
Enable disabled tests in the test suite which should be e...	Medium	Jay Pipes	Slow progress
Remove dead code	Medium		Slow progress
A new table discovery mechanism	Medium	Jay Pipes	Started
Use autotools AC_REPLACE_FUNCS for duplicate functions	Low	Monty Taylor	Unknown
Clean up coding style	Undefined		Good progress

2 bugs targeted

Report	Importance	Assignee	Status
268263 Failed Tests - ZEROFILL Syntax Errors	High		Confirmed
267978 gettext dependency not properly picked up in build process	Low	Monty Taylor	Confirmed

https://blueprints.launchpad.net/drizzle/+milestone/cirrus

closing a launchpad bug through bazaar

- Extremely useful option to the bzr commit command
- Allows you to automatically relate a bug to a *specific* commit in the source tree
- Automatically changes the bug status to *fix committed*

```
$> bzr commit --fixes=lp:XXXXXX
```

- Where XXXXXX is the bug ID from Launchpad
- Also prefixes for other bug trackers (see bzr manual)

merging and resolving conflicts

- Use bzr merge to merge one branch's changes into another
- A *merge conflict* occurs when source files cannot be cleanly merged together
  - Typically when two developers have edited the same lines of the same source file
- Conflicting files will have file.OTHER, file.THIS, and file.BASE in the source directory
- Resolve the conflicts using Meld

merging one branch into another branch

- To merge, go to the target branch and specify the source directory:

```
$> cd ~/repos/drizzle/enable-tests
```

```
$> bzr merge ../trunk
```

- This would merge trunk's changes into the enable-tests branch
- Conflicts listed at end of merge output:

```
Text conflict in tests/t/func_gconcat.test
Text conflict in tests/t/func_str.test
2 conflicts encountered.
```

using bzr gconflicts

- Conflicts can always be listed with bzr conflicts:

```
[510][jpipes@serialcoder: /home/jpipes/repos/drizzle/enable-tests]$ bzr conflicts
Text conflict in tests/t/func_gconcat.test
Text conflict in tests/t/func_str.test
```

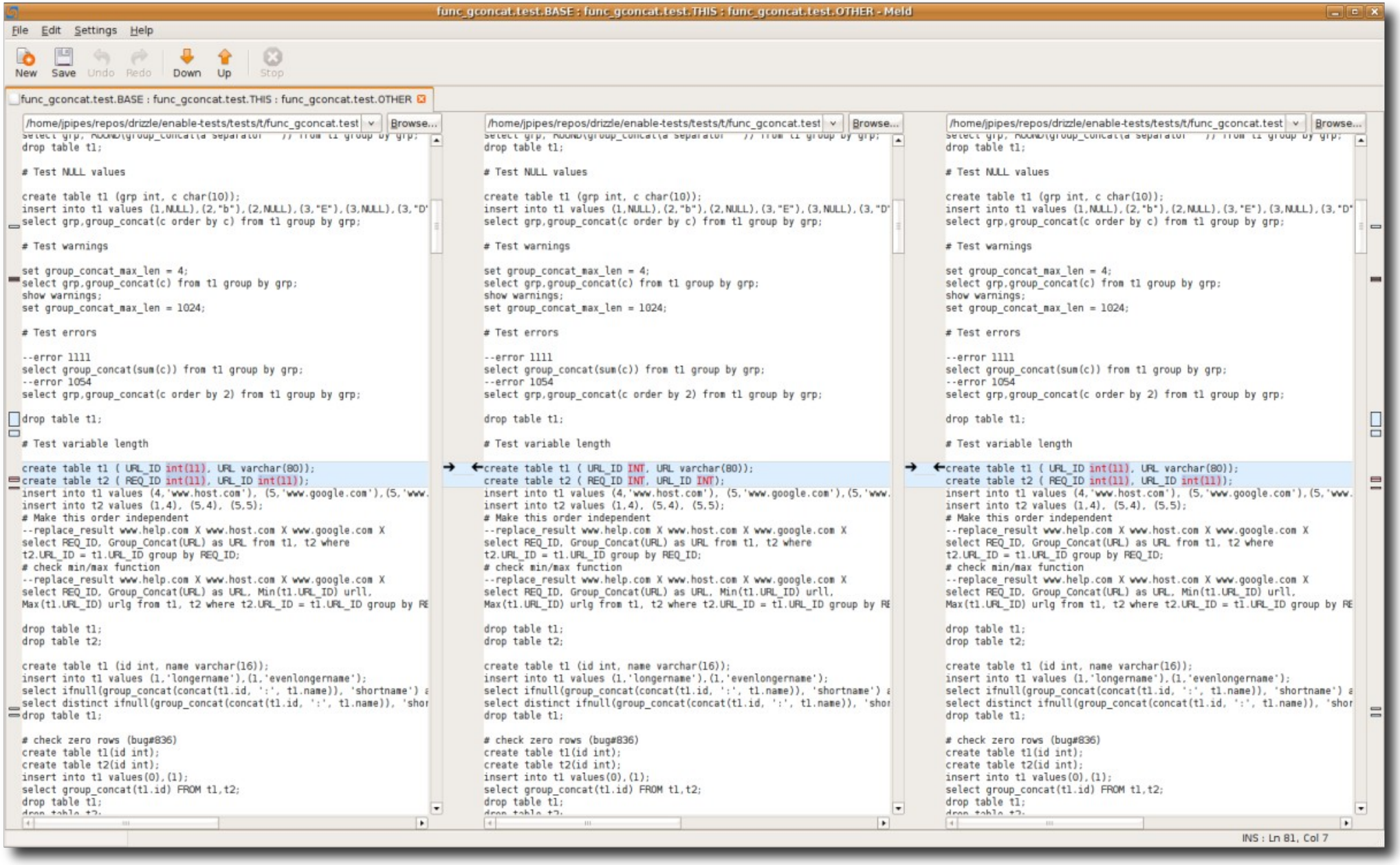
- Use Meld to resolve a conflict

$> bzr gconflicts

- This will list all files with conflicts
- Choose a file to resolve and select “meld” from the dropdown of utilities

launchpad

using meld to resolve a conflict



using meld for visual diffs

- Ctrl-D to jump to the next conflict (Ctrl-E goes back one conflict)
- The left-most file is the BASE file, the middle file is the local file and the right-most file is the one from the merged branch
- Decide how to resolve the conflict and edit the middle file (filename.THIS)
- Changes can be applied by editing the files directly or by clicking the small arrows in the gray area in between the files
- Ctrl-S to save the changes

resolving the conflict in bzr

- Tell bzr that you've resolved things
- Then commit

```
[518][jpipes@serialcoder: enable-tests]$ bzr resolve --all
[525][jpipes@serialcoder: enable-tests]$ bzr commit -m \
"Merge from trunk and resolved conflicts func_gconcat and func_str - simple removal of ZEROFILL args"
```

resources

- Bazaar online manual and user guide

http://doc.bazaar-vcs.org/bzr.dev/en/user-reference/bzr_man.html

http://doc.bazaar-vcs.org/bzr.dev/en/user-guide/index.html

- Launchpad user's wiki

https://help.launchpad.net/

- IRC on Freenode #launchpad and #bzr

the one slide on licensing

- I hate talking about licensing
  - Everybody has a different opinion...
- But here's what matters to you:
  - If you contribute code inside the `/drizzled` directory, that code is contributed under the GPL since the kernel is derived from MySQL
  - If you contribute a plugin implementation or module, you can contribute that code under a license of your choice
- Feel free to get into lots of debate and arguments about licensing
  - Just not with me

A Word About Style

Consistent Rules for Coding

Code Style Rules

- Yes, these are enforced in code review... :)
- Consistency is the key
- Nobody agrees with all of the style, but everyone should follow it
- Otherwise the code is very difficult to navigate
- No TABs
- TABs should be expanded as spaces
- 2 space indentation
- Use Vim? Check out my vimrc on joinfu.com

Class Names

- Pascal casing, no underscores
- Inconsistent in code...cleanup underway
- CORRECT:

```
class MyClassName;
```

- INCORRECT:

```
class My_Class_Name;
```

- INCORRECT:

```
class MY_CLASS_NAME;
```

Class *Method* Names

- Camel casing, no underscores
- Inconsistent in code...cleanup underway
- CORRECT:

```
int getSomeValue();
```

- INCORRECT:

```
int get_some_value();
```

- INCORRECT:

```
int GetSomeValue();
```

Class Design

- Keep class member variable *private* unless there is a good reason not to
  - And there really isn't any good reason
- Write *public* accessors and setters for these member variables
- General rules of class design:
  - Only expose the classes' API
  - Only expose what is necessary to expose
  - RAII
    - Use constructors and destructors properly
    - Objects should own their own memory

Assignment

- Zero spaces before assignment operator
- One and only one space afterwards
- CORRECT:

```
uint32_t my_counter= 0;
```

- INCORRECT:

```
uint32_t my_counter = 0;
```

- INCORRECT:

```
uint32_t my_counter=      0;
```

Comparison

- One and only one space before and after comparison operator
- CORRECT:

```
if (my_counter == 1)
```

- INCORRECT:

```
if (my_counter==1)
```

- INCORRECT:

```
if ( my_counter== 1 )
```

Braces

- Braces should be on their own line
- *else* should be on its own line
- CORRECT:

```
if (my_counter == 1)
{
  // do something
}
```

- INCORRECT:

```
if (my_counter == 1) {
  // do something
}
```

Braces (cont'd)

- Classes and namespaces follow same standard
- Same with switch!
- CORRECT:

```
class MyClass :public SomeOtherClass
{
private:
  int my_counter;
};
```

- INCORRECT:

```
class MyClass :public SomeOtherClass {
  private:
  int my_counter;
};
```

If in doubt...

Check the Wiki:

http://drizzle.org/wiki/Coding_Standards

Inside the Code

Overview of the Drizzle Code Base

directory organization

- /client
  - Client programs (drizzle.cc, drizzledump.cc etc)
- /config
  - Scripts such as autorun.sh for the build process
- /extra
  - Contains my_print_defaults.cc
  - Will be going away this summer (yeah! \o/)
- /gnulib
  - Portability headers

directory organization (cont'd)

- /support-files
  - Various utility scripts
- /tests
  - Unit and functional test cases and suites
  - As a contributor, you will want to familiarize yourself with this directory! :)
- /drizzled
  - ALL kernel code
  - Optimizer, parser, runtime, plugin *APIs*

/drizzled directory

- /drizzled/memory
  - Legacy memory allocation
  - Will be a day of days when it is removed
- /drizzled/internal
  - MySQL portability/system library
  - Many things removed from original MySQL mysys library
  - You should take care when using any function in here
    - Check for a standard library prototype first!

/drizzled (kernel code)

- /drizzled/atomic
  - Portable C++ atomic<> implementation
- /drizzled/message
  - Google Protobuffer proto definitions
- /drizzled/utf8
  - C++ UTF8 thin library
- /drizzled/util
  - Bits and pieces of utility code

/drizzled (cont'd)

- /drizzled/plugin
  - Plugin base interface class definitions
- /drizzled/item
  - Item derived classes
- /drizzled/field
  - Field storage classes
- /drizzled/function
  - Built-in SQL functions

/drizzled (cont'd)

- /drizzled/optimizer
  - Most optimizer code, range operations, aggregation
- /drizzled/statement
  - SQL Statement classes
  - e.g. statement::Insert
- /drizzled/algorithm
  - crc32, sha1, etc..

/plugin (module code)

- Lots of plugin examples and default implementations
  - Authentication
  - Data Dictionaries (TableFunction)
  - Replicators
  - Transaction log
  - Logging
  - Session scheduling
  - Pluggable functions
  - Storage engines

libdrizzle

- BSD licensed, written in pure C by Eric Day
- Client/server communication protocol
- Clean, stack-based approach
  - http://launchpad.net/libdrizzle
- Requirement for developing Drizzle:

```
bzr branch lp:libdrizzle libdrizzle
cd libdrizzle; ./config/autorun.sh; ./configure
make && make check
sudo make install
```

Easy First Steps

Where to start?

don't dig too deep!

- It's best to start with small, attainable goals
- Very easy to go down “ratholes” in the code
- Have clear, well-defined tasks
- Stay out of the optimizer until you've coded on Drizzle for >3 months ;)
- Lots of little tasks that make it easy to get your feet wet and feel like you've gotten stuff accomplished
  - See low-hanging-fruit milestone

get your feet wet

- Refactoring and code cleanup
  - Replacing custom code with STL or libc
  - Cleaning up style and indentation problems
- Documenting the large parts of the source code which are undocumented
  - Great way to learn the source code without altering
- Creating test cases
  - Look at where the source code is weak on test coverage: http://drizzle.org/lcov/
  - Work on creating tests to cover missing spots or remove dead code

lcov

LCOV - code coverage report

Current view: directory
Test: drizzle_lcov.info
Date: 2010-04-27

	Found	Hit	Coverage
Lines:	124808	84820	68.0 %
Functions:	11185	7959	71.2 %

Directory	Line Coverage		Functions	
client	58.9 %	3769 / 6403	68.8 %	234 / 340
drizzled	80.1 %	24733 / 30865	80.1 %	1674 / 2089
drizzled/algorithm	98.6 %	71 / 72	100.0 %	5 / 5
drizzled/field	81.3 %	1072 / 1319	84.3 %	183 / 217
drizzled/function	83.0 %	787 / 948	89.4 %	126 / 141
drizzled/function/math	86.3 %	572 / 663	94.9 %	131 / 138
drizzled/function/str	81.4 %	839 / 1031	89.2 %	99 / 111
drizzled/function/time	78.1 %	958 / 1227	90.0 %	117 / 130
drizzled/internal	68.1 %	2005 / 2943	85.7 %	108 / 126
drizzled/item	83.7 %	5922 / 7073	77.7 %	932 / 1199
drizzled/memory	92.1 %	163 / 177	95.2 %	20 / 21
drizzled/message	54.9 %	5262 / 9580	49.4 %	886 / 1794
drizzled/optimizer	83.7 %	3454 / 4125	79.9 %	183 / 229
drizzled/plugin	67.8 %	1381 / 2038	74.2 %	236 / 318
drizzled/statement	85.6 %	971 / 1135	100.0 %	123 / 123
drizzled/util	16.3 %	7 / 43	42.9 %	3 / 7
extra	47.3 %	26 / 55	75.0 %	3 / 4
plugin/archive	75.8 %	762 / 1005	87.1 %	81 / 93
plugin/ascii	90.9 %	20 / 22	81.8 %	9 / 11
plugin/auth_file	80.0 %	56 / 70	66.7 %	8 / 12
plugin/benchmark	81.2 %	39 / 48	75.0 %	9 / 12
plugin/blackhole	73.3 %	137 / 187	64.0 %	32 / 50
plugin/charlength	91.7 %	22 / 24	72.7 %	8 / 11
plugin/collation_dictionary	100.0 %	109 / 109	79.3 %	23 / 29
plugin/compression	87.3 %	62 / 71	75.0 %	12 / 16
plugin/connection_id	87.0 %	20 / 23	75.0 %	9 / 12
plugin/console	11.3 %	17 / 150	16.2 %	6 / 37
plugin/crc32	92.0 %	23 / 25	81.8 %	9 / 11
plugin/csv	76.1 %	401 / 527	76.4 %	55 / 72
plugin/database_function	100.0 %	20 / 20	80.0 %	8 / 10
plugin/default_replicator	100.0 %	9 / 9	80.0 %	4 / 5
plugin/drizzle_protocol	1.8 %	15 / 850	9.9 %	9 / 91
plugin/errmsg_stderr	100.0 %	13 / 13	80.0 %	4 / 5
plugin/filtered_replicator	63.9 %	131 / 205	68.4 %	13 / 19
plugin/function_engine	90.2 %	101 / 112	81.5 %	22 / 27
plugin/gearman_udf	19.0 %	29 / 153	19.6 %	11 / 56
plugin/heap	81.3 %	1400 / 1722	86.9 %	119 / 137
plugin/hello_world	92.9 %	13 / 14	70.0 %	7 / 10

coverage of a source code file

LCOV - code coverage report

		Found	Hit	Coverage
Current view:	directory - drizzled/statement - start_transaction.cc (source / functions)			
Test:	drizzle_lcov.info	**Lines:**	9	6	66.7 %
Date:	2010-04-27	**Functions:**	3	3	100.0 %

```
                :  /* -*- mode: c++; c-basic-offset: 2; indent-tabs-mode: nil; -*-
                :   *  vim:expandtab:shiftwidth=2:tabstop=2:smarttab:
                :   *
                :   *  Copyright (C) 2009 Sun Microsystems
                :   *
                :   *  This program is free software; you can redistribute it and/or modify
                :   *  it under the terms of the GNU General Public License as published by
                :   *  the Free Software Foundation; either version 2 of the License, or
                :   *  (at your option) any later version.
                :   *
                :   *  This program is distributed in the hope that it will be useful,
                :   *  but WITHOUT ANY WARRANTY; without even the implied warranty of
                :   *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
                :   *  GNU General Public License for more details.
                :   *
                :   *  You should have received a copy of the GNU General Public License
                :   *  along with this program; if not, write to the Free Software
                :   *  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
                :   */
                :
                :  #include "config.h"
                :  #include <drizzled/show.h>
                :  #include <drizzled/session.h>
                :  #include <drizzled/statement/start_transaction.h>
                :
                :  namespace drizzled
                :  {
                :
             89 :  bool statement::StartTransaction::execute()
                :  {
             89 :    if (session->transaction.xid_state.xa_state != XA_NOTR)
                :    {
                :      my_error(ER_XAER_RMFAIL, MYF(0),
              0 :               xa_state_names[session->transaction.xid_state.xa_state]);
              0 :      return false;
                :    }
                :    /*
                :       Breakpoints for backup testing.
                :     */
             89 :    if (! session->startTransaction(start_transaction_opt))
                :    {
              0 :      return true;
                :    }
             89 :    session->my_ok();
             89 :    return false;
                :  }
                :
            129 :  }
```

Generated by: LCOV version 1.7

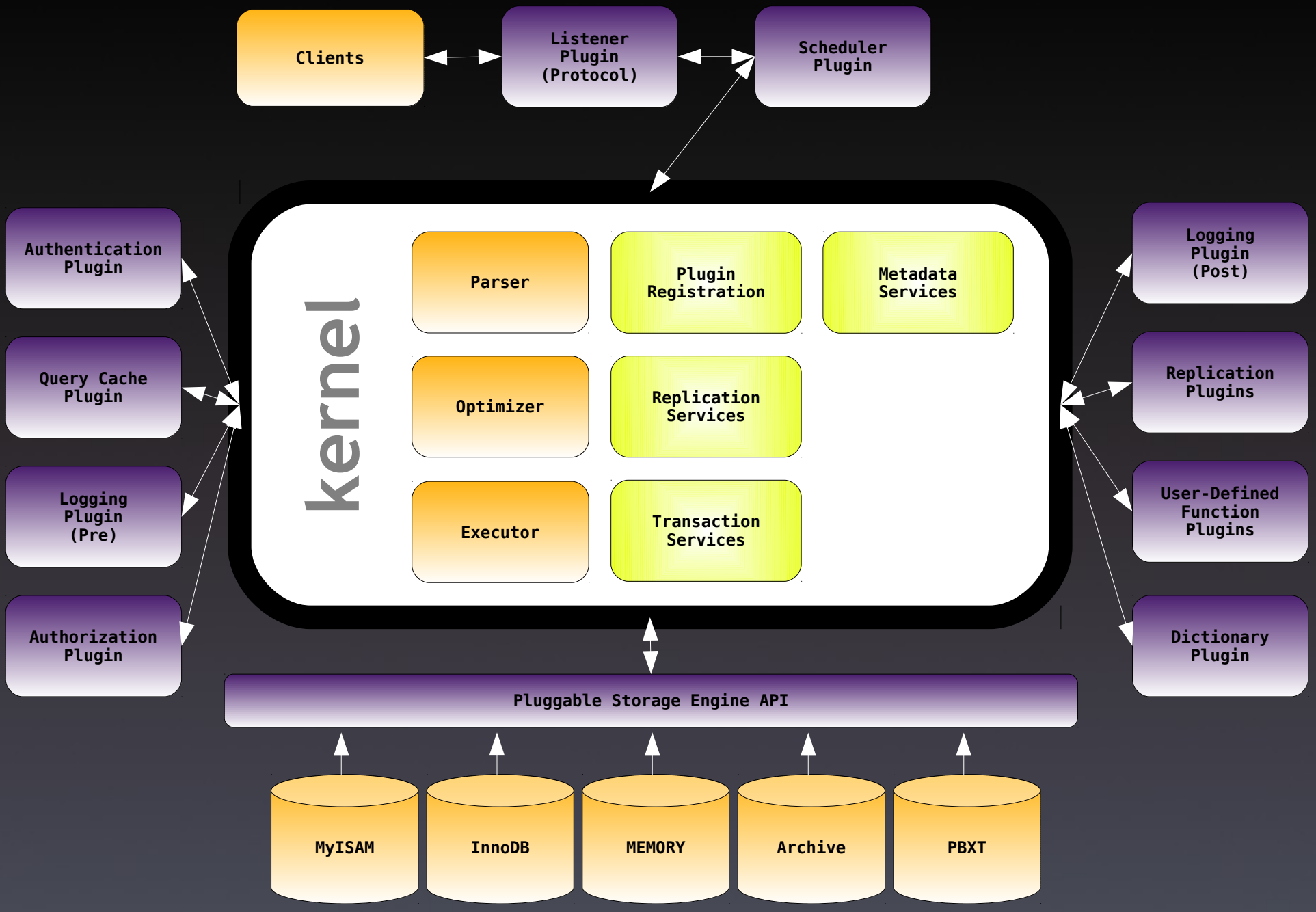
Overview of Drizzle's Architecture

drizzle's system architecture

- “Microkernel” design means most features are built as plugins
  - Authentication, replication, logging, information schema, storage engine, etc
  - The kernel is really just the parser, optimizer, and runtime
- We are C++, not C+
- We use open source libraries as much as possible
  - STL, gettext, Boost, pcre, GPB, etc
  - Don't reinvent the wheel

drizzle's system architecture

- No single “right way” of implementing something
  - Your solution may be great for your environment, but not good for others
  - And that's fine - it's what the plugin system is all about
- We focus on the APIs so you can focus on the implementation
- Drizzle is just one part of a large ecosystem
  - Web servers, caching layers, authentication systems


Clients
Listener Plugin (Protocol)
Scheduler Plugin
Authentication Plugin
Query Cache Plugin
Logging Plugin (Pre)
Authorization Plugin
kernel
Parser
Plugin Registration
Metadata Services
Optimizer
Replication Services
Executor
Transaction Services
Logging Plugin (Post)
Replication Plugins
User-Defined Function Plugins
Dictionary Plugin
Pluggable Storage Engine API
MyISAM
InnoDB
MEMORY
Archive
PBXT

ignore the kernel

- You should be able to ignore the kernel as a “black box”
- Plugin developers should focus on their plugin or module and not change anything in the kernel
- If you need to meddle with or change something in the kernel, it is a sign of a bad interface
  - And you should file a bug! :)

Under the Hood

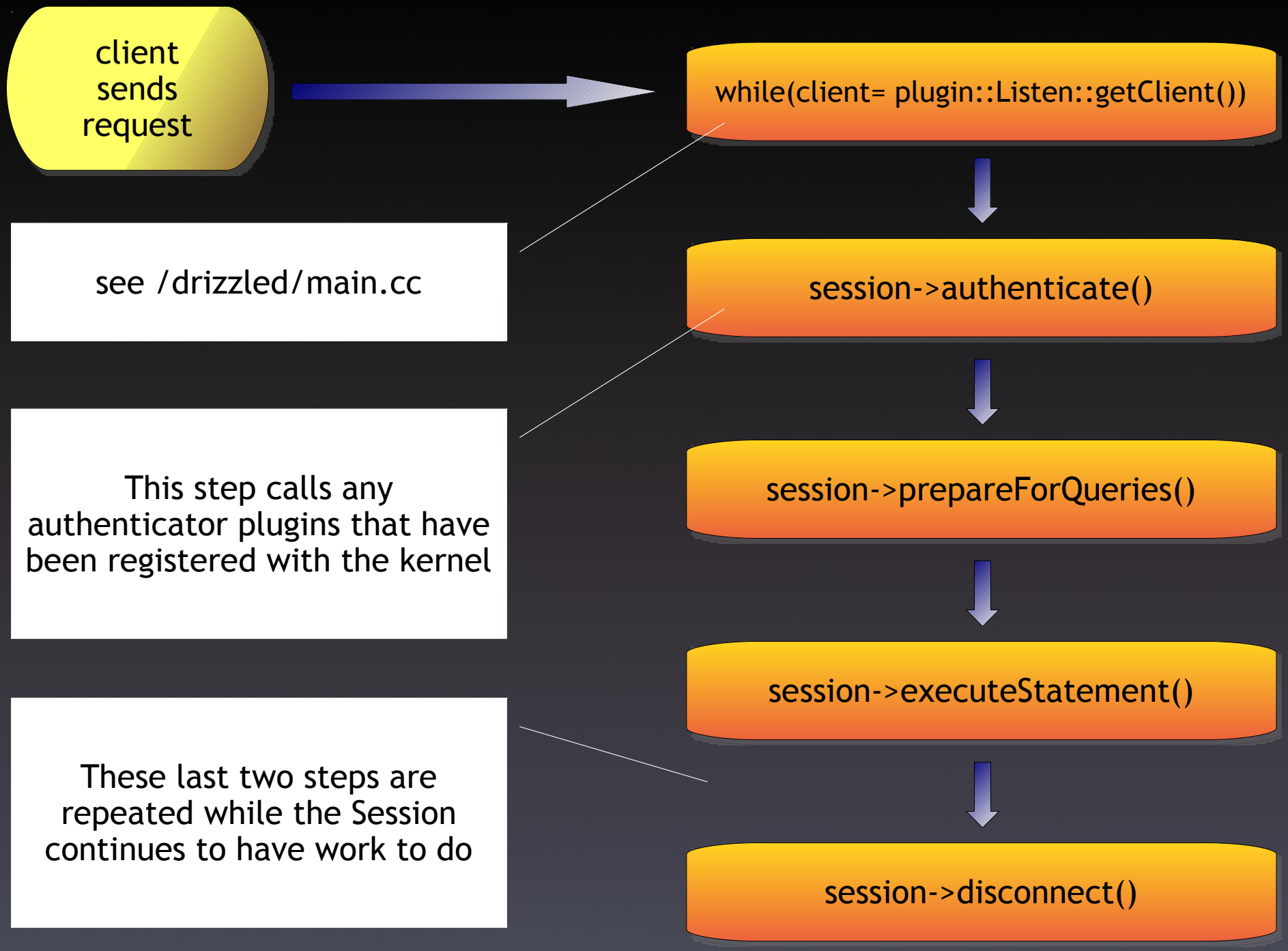
Kernel Code Walk-through

Drizzle kernel

- Written in C++
  - Not C, Not C+
- Responsible for the “runtime” and coordinating communication between various plugins, clients, and itself
- Big parts:
  - Session handling
  - SQL statement parsing and optimization
  - Execution of parsed statements
  - Registering and communicating with plugins

The Session

- Session != OS Thread
- Represents the series of SQL commands received from a client
- Defined in `/drizzled/session.h`
- Contains its own separate memory area, called a mem_root, for memory allocated that lives for the lifetime of the Session object


client sends request
while(client= plugin::Listen::getClient())
see /drizzled/main.cc
session->authenticate()
This step calls any authenticator plugins that have been registered with the kernel
session->prepareForQueries()
session->executeStatement()
These last two steps are repeated while the Session continues to have work to do
session->disconnect()

main client-listening loop

/drizzled/main.cc

```
292   /* Listen for new connections and start new session for each connection
293      accepted. The listen.getClient() method will return NULL when the server
294      should be shutdown. */
295   while ((client= plugin::Listen::getClient()) != NULL)
296   {
297     if (!(session= new Session(client)))
298     {
299       delete client;
300       continue;
301     }
302
303     /* If we error on creation we drop the connection and delete the session. */
304     if (session->schedule())
305       Session::unlink(session);
306   }
```

scheduling a session

/drizzled/session.cc

```
bool Session::schedule()
{
  scheduler= plugin::Scheduler::getScheduler();

  // <snip>

  pthread_mutex_lock(&LOCK_thread_count);
  getSessionList().push_back(this);
  pthread_mutex_unlock(&LOCK_thread_count);

  if (scheduler->addSession(this))
  {
    // <snip>
    return true;
  }

  return false;
}
```

/plugin/multi_thread/multi_thread.cc

```
 1 namespace
 2 {
 3   extern "C" pthread_handler_t session_thread(void *arg)
 4   {
 5     Session *session= static_cast<Session*>(arg);
 6     MultiThreadScheduler *sched=
 7       static_cast<MultiThreadScheduler*>(session->scheduler);
 8     sched->runSession(session);
 9     return NULL;
10   }
11 }
12
13 bool MultiThreadScheduler::addSession(Session *session)
14 {
15   if (thread_count >= max_threads)
16     return true;
17
18   thread_count.increment();
19
20   if (pthread_create(&session->real_id, &attr, session_thread,
21                      static_cast<void*>(session)))
22   {
23     thread_count.decrement();
24     return true;
25   }
26
27   return false;
28 }
```

running a session

/plugin/multi_thread/multi_thread.h

```
57   void runSession(drizzled::Session *session)
58   {
59     if (drizzled::internal::my_thread_init())
60     {
61       session->disconnect(drizzled::ER_OUT_OF_RESOURCES, true);
62       statistic_increment(drizzled::aborted_connects, &LOCK_status);
63       killSessionNow(session);
64     }
65
66     session->thread_stack= (char*) &session;
67     session->run();
68     killSessionNow(session);
69   }
```

/drizzled/session.cc

```
1  void Session::run()
2  {
3    if (initGlobals() || authenticate())
4    {
5      disconnect(0, true);
6      return;
7    }
8
9    prepareForQueries();
10
11   while (! client->haveError() && killed != KILL_CONNECTION)
12   {
13     if (! executeStatement())
14       break;
15   }
16
17   disconnect(0, true);
18 }
```

Session::executeStatement()

- Lots 'o stuff happening
- Depends on the command received from the client
- Eventually, the `dispatch_command()` function is reached, which dispatches the execution to the `drizzled::statement::Statement` subclass created in the parser
  - Command is an integer SQLCOM_XXX
  - See /drizzled/sql_parse.cc
- The actual `drizzled::statement::Statement` subclass has its `execute()` method called

Session::executeStatement()

/drizzled/session.cc

```
 1 bool Session::executeStatement()
 2 {
 3   char *l_packet= 0;
 4   uint32_t packet_length;
 5
 6   enum enum_server_command l_command;
 7
 8   // <snip>
 9
10   if (client->readCommand(&l_packet, &packet_length) == false)
11     return false;
12
13   if (packet_length == 0)
14     return true;
15
16   l_command= (enum enum_server_command) (unsigned char) l_packet[0];
17
18   if (command >= COM_END)
19     command= COM_END;                          // Wrong command
20
21   assert(packet_length);
22   return ! dispatch_command(l_command, this, l_packet+1, (uint32_t) (packet_length-1));
23 }
```

dispatch_command()

/drizzled/sql_parse.cc

```
 1 bool dispatch_command(enum enum_server_command command, Session *session,
 2                        char* packet, uint32_t packet_length)
 3 {
 4   plugin::Logging::preDo(session);
 5   switch (command) {
 6   case COM_QUERY:
 7   {
 8     if (! session->readAndStoreQuery(packet, packet_length))
 9       break;
10     plugin::QueryRewriter::rewriteQuery(session->db, session->query);
11     mysql_parse(session, session->query.c_str(), session->query.length());
12     break;
13   }
14
15   TransactionServices &transaction_services= TransactionServices::singleton();
16   transaction_services.autocommitOrRollback(session, session->is_error());
17
18   session->client->sendOK();
19
20   /* Free tables */
21   session->close_thread_tables();
22
23   plugin::Logging::postDo(session);
24
25   return error;
26 }
```

Parsing of a statement

- `COM_QUERY` commands have a corresponding string of SQL text passed to the `mysql_parse()` method
- This string must be parsed
- Grammar stored in a Yacc file
  - see /drizzled/sql_yacc.yy
- `DRIZZLEparse()` and `DRIZZLElex()` are the two functions which handle parsing
  - see /drizzled/sql_parse.cc
  - see /drizzled/sql_lex.cc

mysql_parse()

/drizzled/sql_parse.cc

```
 1 void mysql_parse(Session *session, const char *inBuf, uint32_t length)
 2 {
 3   LEX *lex= session->lex;
 4
 5   Lex_input_stream lip(session, inBuf, length);
 6
 7   bool err= parse_sql(session, &lip);
 8
 9   if (!err)
10   {
11     if (! session->is_error())
12     {
13       /* Actually execute the query */
14       mysql_execute_command(session);
15     }
16   }
17   session->end_statement();
18   return;
19 }
```

Parsing (cont'd)

- The parsing process actually does a lot more than just lex and parse the statement's SQL string
  - This is unfortunate, because it makes modifying and modularizing the parser difficult
  - Work is underway to address this
- The parsing process allocates a series of Item class objects, and constructs a **LEX** object which represents the parsed statement
- The **LEX** *is not* an abstract syntax tree, nor is it a compiled execution plan

Parsing (cont'd)

- After the `LEX` is constructed, it may go through some post-processing (particularly in the case of a SELECT statement)
- The `LEX` is eventually tacked onto the `Session` so that routines processing the statement can refer to its parsed structure
  - see /drizzled/sql_lex.h
  - see /drizzled/sql_lex.cc
- After this point, the type of command being executed determines what happens next...

mysql_execute_command()

/drizzled/sql_parse.cc

```
1 static int mysql_execute_command(Session *session)
2 {
3   /* now we are ready to execute the statement */
4   res= lex->statement->execute();
5   return (res || session->is_error());
6 }
```

- OK, so that's not the whole function, but it's the important part of it
- Want an easy starter task? Get rid of `mysql_execute_command()` and put all of the code in it into `mysql_parse()` and then rename `mysql_parse()` to `drizzled::parseStatement()`

statement::Select::execute()

/drizzled/statement/select.cc

```
28 bool statement::Select::execute()
29 {
30   TableList *all_tables= session->lex->query_tables;
31   session->status_var.last_query_cost= 0.0;
32   bool res= execute_sqlcom_select(session, all_tables);
33
34   return res;
35 }
```

- This is one of the statement subclasses which does not do much
- Other statement subclass' execute() methods do much more -- for instance
  statement::AlterTable::execute()

execute_sqlcom_select()

/drizzled/sql_parse.cc

```
 1 bool execute_sqlcom_select(Session *session, TableList *all_tables)
 2 {
 3   if (not (res= session->openTablesLock(all_tables)))
 4   {
 5     if (lex->describe)
 6     {
 7       if (!(result= new select_send()))
 8         return true;
 9       session->send_explain_fields(result);
10       optimizer::ExplainPlan planner;
11       res= planner.explainUnion(session, &session->lex->unit, result);
12       if (res)
13         result->abort();
14       else
15         result->send_eof();
16       delete result;
17     }
18     else
19     {
20       if (!result && !(result= new select_send()))
21         return true;
22       res= handle_select(session, lex, result, 0);
23       if (result != lex->result)
24         delete result;
25     }
26   }
27   return res;
28 }
```

Session::openTablesLock()

- Should really be called something more descriptive
- It doesn't "lock" the tables used in the query necessarily
- It "advises" the underlying engine(s) of the query's *intent*
  - Does the query *intend* to modify rows?
  - Does the query *intend* to write rows?
- A thin wrapper around the eventual call to `mysql_lock_tables()`

mysql_lock_tables()

/drizzled/lock.cc (heavily abbreviated)

```
 1 DRIZZLE_LOCK *mysql_lock_tables(Session *session, Table **tables, uint32_t count,
 2                                 uint32_t flags, bool *need_reopen)
 3 {
 4   for (;;)
 5   {
 6     if (! (sql_lock= get_lock_data(session, tables, count, true,
 7                                    &write_lock_used)))
 8       break;
 9
10     // <snip>
11     for_each(involved_engines.begin(),
12              involved_engines.end(),
13              bind2nd(mem_fun(&plugin::StorageEngine::startStatement), session));
14
15     if (sql_lock->table_count && lock_external(session, sql_lock->table,
16                                                sql_lock->table_count))
17     {
18       /* Clear the lock type of all lock data to avoid reusage. */
19       reset_lock_data_and_free(&sql_lock);
20       break;
21     }
22   }
23   return (sql_lock);
24 }
```

handle_select()

/drizzled/sql_select.cc

```
 1 bool handle_select(Session *session, LEX *lex, select_result *result,
 2                    uint64_t setup_tables_done_option)
 3 {
 4   register Select_Lex *select_lex= &lex->select_lex;
 5
 6   if (select_lex->master_unit()->is_union() ||
 7       select_lex->master_unit()->fake_select_lex)
 8   {
 9     res= drizzle_union(session, lex, result, &lex->unit,
10            setup_tables_done_option);
11   }
12   else
13   {
14     Select_Lex_Unit *unit= &lex->unit;
15     unit->set_limit(unit->global_parameters);
16     session->session_marker= 0;
17     res= mysql_select(session, &select_lex->ref_pointer_array,
18           (TableList*) select_lex->table_list.first,
19           select_lex->with_wild, select_lex->item_list,
20           select_lex->where,
21           select_lex->order_list.elements +
22           select_lex->group_list.elements,
23           (order_st*) select_lex->order_list.first,
24           (order_st*) select_lex->group_list.first,
25           select_lex->having,
26           select_lex->options | session->options |
27                       setup_tables_done_option,
28           result, unit, select_lex);
29   }
30   return res;
31 }
```

mysql_select()

/drizzled/sql_select.cc (heavily edited)

```
 1 bool mysql_select(...)
 2 {
 3   join= new JOIN(session, fields, select_options, result);
 4
 5   join->prepare(rref_pointer_array, tables, wild_num,
 6                 conds, og_num, order, group, having,
 7                 select_lex, unit);
 8
 9   err= join->optimize();
10
11   join->exec();
12 }
```

Optimization of SELECT statements

- During execution of SELECT statements, the optimizer “module” is called
  - It's not really a module, more of a loose collection of classes and functions in /drizzled/optimizer/
  - See /drizzled/sql_select.cc
  - See /drizzled/join.cc
  - See /drizzled/optimizer/range.cc
- The `Join` class is the dominant class used in the optimizer's routines

Optimization (cont'd)

- It may not be obvious by looking at the code, but the `Join` class' responsibility is to query the storage engine (`plugin::StorageEngine` and `drizzled::Cursor`) and determine how best to perform the nested loops join algorithm
- In other words, determine the best access plan to the data in the storage engine
  - choose_plan():/drizzled/join.cc
  - best_access_path():/drizzled/join.cc
  - Join::prepare() (224 LOC)
  - Join::optimize() (685 LOC!)

Execution

- Nested loops join algorithm
- Implemented using the `READ_RECORD` struct and a set of routines in /drizzled/sql_select.cc
  - join_read_system()
  - join_read_const()
  - join_read_key(), etc...
- Think of `READ_RECORD` as a rudimentary cursor over the storage engine's raw records
- `READ_RECORD` has a variable `read_record` of type pointer to function, which controls reading
  - See /drizzled/records.cc

join_read_first() & join_read_next()

/drizzled/sql_select.cc (abbreviated)

```
 1 int join_read_first(JoinTable *tab)
 2 {
 3   if (!table->cursor->inited)
 4     table->cursor->startIndexScan(tab->index, tab->sorted);
 5   if ((error=tab->table->cursor->index_first(tab->table->record[0])))
 6   {
 7     return -1;
 8   }
 9   return 0;
10 }
11
12 int join_read_next(READ_RECORD *info)
13 {
14   int error;
15   if ((error=info->cursor->index_next(info->record)))
16     return info->table->report_error(error);
17   return 0;
18 }
```

Packaging results

- The `select_send` class is used to "package" up rows as they are sent to a client
- Each field in the resultset of data is bound to an `Item` instance
- `select_send::send_data()` sends a *single row of data* back to a client

select_send::send_data()

/drizzled/select_send.h

```
 1 bool send_data(List<Item> &items)
 2 {
 3   List_iterator_fast<Item> li(items);
 4   char buff[MAX_FIELD_WIDTH];
 5   String buffer(buff, sizeof(buff), &my_charset_bin);
 6
 7   Item *item;
 8   while ((item=li++))
 9   {
10     if (item->send(session->client, &buffer))
11     {
12       my_message(ER_OUT_OF_RESOURCES, ER(ER_OUT_OF_RESOURCES), MYF(0));
13       break;
14     }
15   }
16   session->sent_row_count++;
17   if (session->is_error())
18     return true;
19   return session->client->flush();
```

Walkthrough of Drizzle Plugin Basics

plugin/module development basics

- A working C++ development environment
  - http://www.joinfu.com/2008/08/getting-a-working-c-c-plusplus-development-environment-for-developing-drizzle/
- A module in Drizzle is a set of source files in `/plugin/` that implements some functionality
  - For instance `/plugin/transaction_log/*` contains all files for the Transaction Log module
- Each module must have a `plugin.ini` file
  - The fabulous work by Monty Taylor on the Pandora build system automates most work for you

plugin/module development basics

- A module contains one or more implementations of a plugin class
- A plugin class is any class interface declared in `/drizzled/plugin/`
  - For instance, the header file `/drizzled/plugin/transaction_applier.h` declares the interface for the `plugin::TransactionApplier` API
  - The header files contain documentation for the plugin interfaces
  - You can also see documentation on the drizzle.org website: `http://drizzle.org/doxygen/`

the plugin.ini

- A description file for the plugin
- Read during compilation and Pandora build system creates appropriate linkage for you
- Required fields:
  - headers= <list of all header files in module>
  - sources= <list of all source files in module>
  - title= <name of the module/plugin>
  - description= <decription for the module>

from plugin.ini to data dictionary

```
[plugin]
title=Filtered Replicator
author=Padraig O Sullivan
version=0.2
license=PLUGIN_LICENSE_GPL
description=
  A simple filtered replicator which allows a user to filter out events based on a schema or
table name
load_by_default=yes
sources=filtered_replicator.cc
headers=filtered_replicator.h
```

```
drizzle> SELECT * FROM DATA_DICTIONARY.MODULES
    -> WHERE MODULE_NAME LIKE 'FILTERED%'\G
*************************** 1. row ***************************
       MODULE_NAME: filtered_replicator
    MODULE_VERSION: 0.2
     MODULE_AUTHOR: Padraig O'Sullivan
        IS_BUILTIN: FALSE
    MODULE_LIBRARY: filtered_replicator
MODULE_DESCRIPTION: Filtered Replicator
    MODULE_LICENSE: GPL

drizzle> SELECT * FROM DATA_DICTIONARY.PLUGINS
    -> WHERE PLUGIN_NAME LIKE 'FILTERED%'\G
*************************** 1. row ***************************
PLUGIN_NAME: filtered_replicator
PLUGIN_TYPE: TransactionReplicator
  IS_ACTIVE: TRUE
MODULE_NAME: filtered_replicator
```

module initialization

- Recommend placing module-level variables and routines in `/plugin/$module/module.cc`
- Required: an initialization function taking a reference to the `plugin::Context` object for your module as its only parameter
  - Typically named `init()`
- Optional: module-level system variables
- Required: `DECLARE_PLUGIN($init, $vars)` macro inside above source file

module initialization example

```
static DefaultReplicator *default_replicator= NULL; /* The singleton replicator */

static int init(plugin::Context &context)
{
  default_replicator= new DefaultReplicator("default_replicator");
  context.add(default_replicator);
  return 0;
}

DRIZZLE_PLUGIN(init,  NULL);
```

what are plugin hooks?

- Places in the source code that notify plugins about certain events are called *plugin hooks*
- During the course of a query's execution, many plugin hooks can be called
- The subclass of plugin::Plugin determines on which events a plugin is notified and what gets passed as a state parameter to the plugin during notification
- These plugin hooks define the plugin's *API*

Example: plugin::Authentication

```
class Authentication : public Plugin
{
public:
  explicit Authentication(std::string name_arg)
    : Plugin(name_arg, "Authentication")
  {}
  virtual ~Authentication() {}

  virtual bool authenticate(const SecurityContext &sctx,
                            const std::string &passwd)= 0;

  static bool isAuthenticated(const SecurityContext &sctx,
                              const std::string &password);
};
```

- **authenticate()** is the pure virtual method that an implementing class should complete
- **isAuthenticated()** is the plugin hook that is called by the kernel to determine authorization

example plugin hook

```
class AuthenticateBy : public unary_function<plugin::Authentication *, bool>
{
...
  inline result_type operator()(argument_type auth)
  {
    return auth->authenticate(sctx, password);
  }
};

bool plugin::Authentication::isAuthenticated(const SecurityContext &sctx,
                                             const string &password)
{
...
  /* Use find_if instead of foreach so that we can collect return codes */
  vector<plugin::Authentication *>::iterator iter=
    find_if(all_authentication.begin(), all_authentication.end(),
            AuthenticateBy(sctx, password));
...
  if (iter == all_authentication.end())
  {
    my_error(ER_ACCESS_DENIED_ERROR, MYF(0),
             sctx.getUser().c_str(),
             sctx.getIp().c_str(),
             password.empty() ? ER(ER_NO) : ER(ER_YES));
    return false;
  }
  return true;
}
```

testing your plugin

- No plugin should be without corresponding test cases
- Luckily, again because of the work of Monty Taylor, your plugin can easily hook into the Drizzle testing system
- Create a `tests/` directory in your plugin's directory, containing a `t/` and an `r/` subdirectory (for “test” and “result”)

creating test cases

- Your plugin will most likely not be set to load by default
- To activate your plugin, you need to start the server during your tests with:

```
--plugin-add=$module
```

- To automatically have the server started with command-line options by the Drizzle test suite, create a file called `$testname-master.opt` and place it along with your test case in your `/plugin/$module/tests/t/` directory

running your test cases

- Simply run the test-run.pl script with your suite:

```
jpipes@serialcoder:~/repos/drizzle/trunk$ cd tests/
jpipes@serialcoder:~/repos/drizzle/trunk/tests$ ./test-run --suite=transaction_log
Drizzle Version 2010.04.1439
...
==============================================================================
DEFAULT STORAGE ENGINE: innodb
TEST                                                       RESULT    TIME (ms)
------------------------------------------------------------------------------

transaction_log.alter                                      [ pass ]    1025
transaction_log.auto_commit                                [ pass ]     650
transaction_log.blob                                       [ pass ]     661
transaction_log.create_select                              [ pass ]     688
transaction_log.create_table                               [ pass ]     413
transaction_log.delete                                     [ pass ]    1744
transaction_log.filtered_replicator                        [ pass ]    6132
...
transaction_log.schema                                     [ pass ]     137
transaction_log.select_for_update                          [ pass ]    6496
transaction_log.slap                                       [ pass ]   42522
transaction_log.sync_method_every_write                    [ pass ]      23
transaction_log.temp_tables                                [ pass ]     549
transaction_log.truncate                                   [ pass ]     441
transaction_log.truncate_log                               [ pass ]     390
transaction_log.udf_print_transaction_message              [ pass ]     408
transaction_log.update                                     [ pass ]    1916
------------------------------------------------------------------------------
Stopping All Servers
All 28 tests were successful.
```