Agile Database Development

David E. Wheeler
iovation

PGCon 2013
Ottawa

Database Development

David E. Wheeler
iovation

PGCon 2013
Ottawa

David E. Wheeler

PGCon 2013
Ottawa

David E. Wheeler

PGCon 2013
Ottawa

Build My VC-Funded App for Me for Free

David E. Wheeler
CEO, Data Architect
Agile Assholes, Inc.

PGCon 2013
Ottawa

This is Genius

This is Genius

- I had this idea

This is Genius

- I had this idea
- Social networking is HAWT

This is Genius

- I had this idea
- Social networking is HAWT
- Has been for waaaaay too long

This is Genius

- I had this idea
- Social networking is HAWT
- Has been for waaaaay too long
- The backlash is long overdue

This is Genius

- I had this idea
- Social networking is HAWT
- Has been for waaaaay too long
- The backlash is long overdue
- Getting ahead of the curve

This is Genius

- I had this idea
- Social networking is HAWT
- Has been for waaaaay too long
- The backlash is long overdue
- Getting ahead of the curve
- Introducing...

How it Works

How it Works

- Microblogging platform

How it Works

- Microblogging platform
- Everyone follows you

How it Works

- Microblogging platform
- Everyone follows you
- New users follow everyone

How it Works

- Microblogging platform
- Everyone follows you
- New users follow everyone
- Goal: Alienate your followers

How it Works

- Microblogging platform
- Everyone follows you
- New users follow everyone
- Goal: Alienate your followers
- Get them to unfollow you

How it Works

- Microblogging platform
- Everyone follows you
- New users follow everyone
- Goal: Alienate your followers
- Get them to unfollow you
- Leaderboard: Those with fewest followers

Your Task

Your Task

- Create the database

Your Task

- Create the database
- Use agile development to make it right

Your Task

- Create the database
- Use agile development to make it right
- Contribute to VC-funded Corp

Your Task

- Create the database
- Use agile development to make it right
- Contribute to VC-funded Corp
- Profit!

Your Task

- Create the database
- Use agile development to make it right
- Contribute to VC-funded Corp
- Profit!
  - For VC-funded Corp


Flipr
antisocial network

Agile

WTF?

We'll get there

But first...

NDA

Okay, now that that's out of the way...

Please organize into pairs

Yes, that's right

You're gonna do pair database programming

App developers partner with DBAs

I'm waiting...

Job Requirements

Job Requirements

- PostgreSQL
  http://postgresql.org/download

Job Requirements

- PostgreSQL
  http://postgresql.org/download
- Git
  http://git-scm.com/

Job Requirements

- PostgreSQL
  http://postgresql.org/download
- Git
  http://git-scm.com/
- Sqitch
  http://sqitch.org/

Job Requirements

- PostgreSQL
  http://postgresql.org/download
- Git
  http://git-scm.com/
- Sqitch
  http://sqitch.org/
- pgTAP
  http://pgtap.org/

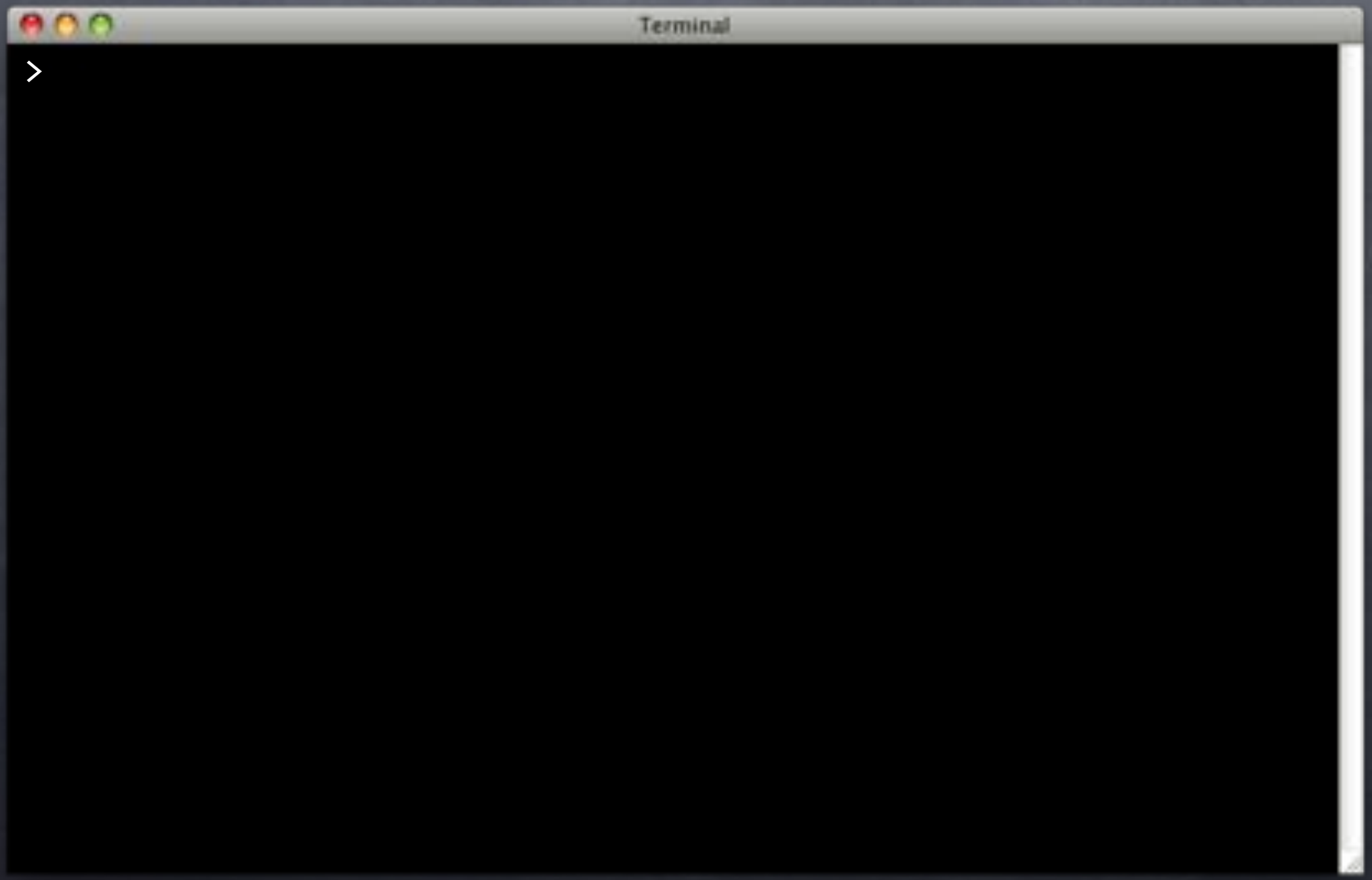
Job Requirements

- PostgreSQL
  http://postgresql.org/download
- Git
  http://git-scm.com/
- Sqitch
  http://sqitch.org/
- pgTAP
  http://pgtap.org/
- pg_prove
  http://metacpan.org/module/pg_prove

We good?

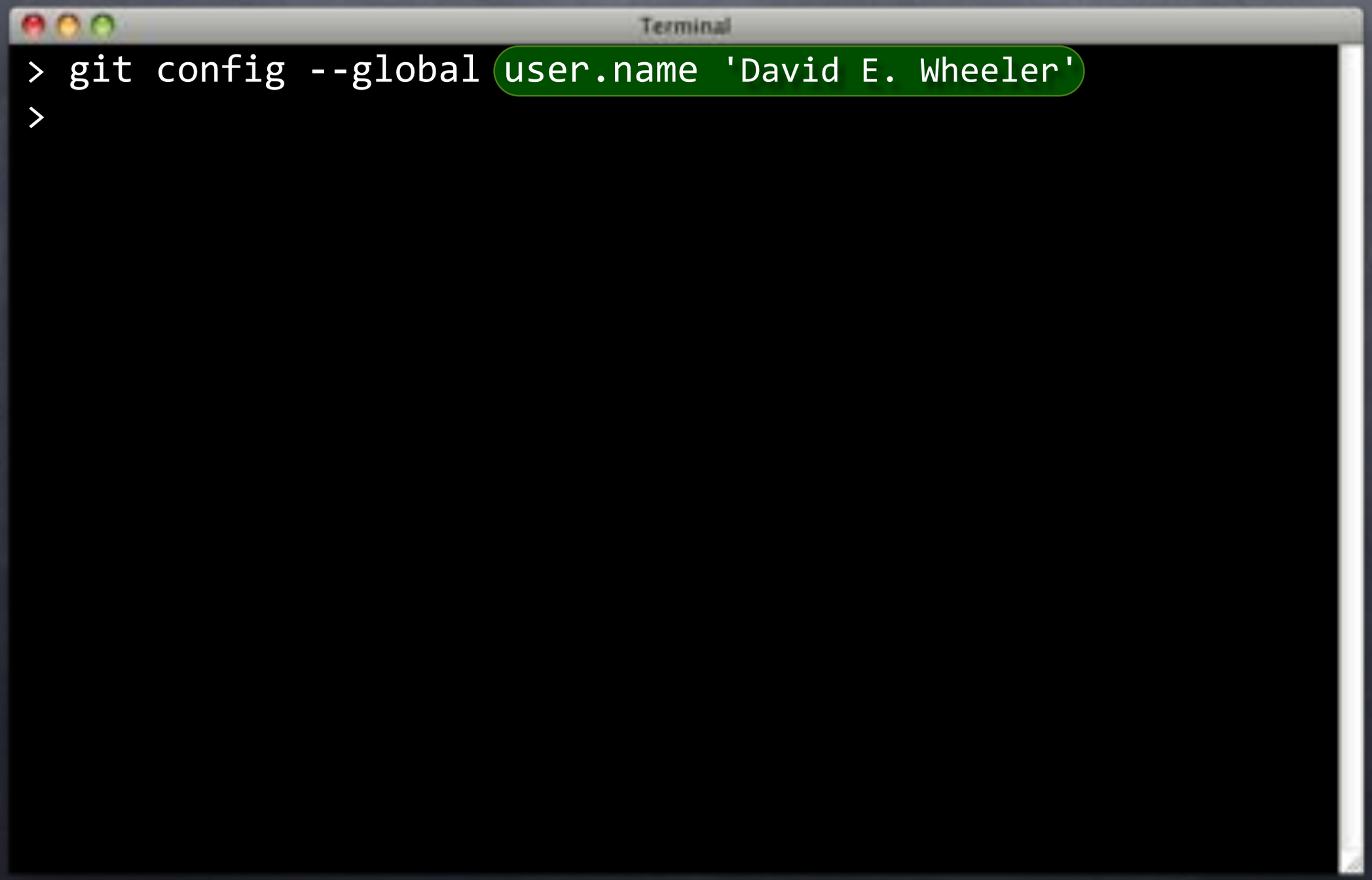
Let's do this thang.

Who Am I?

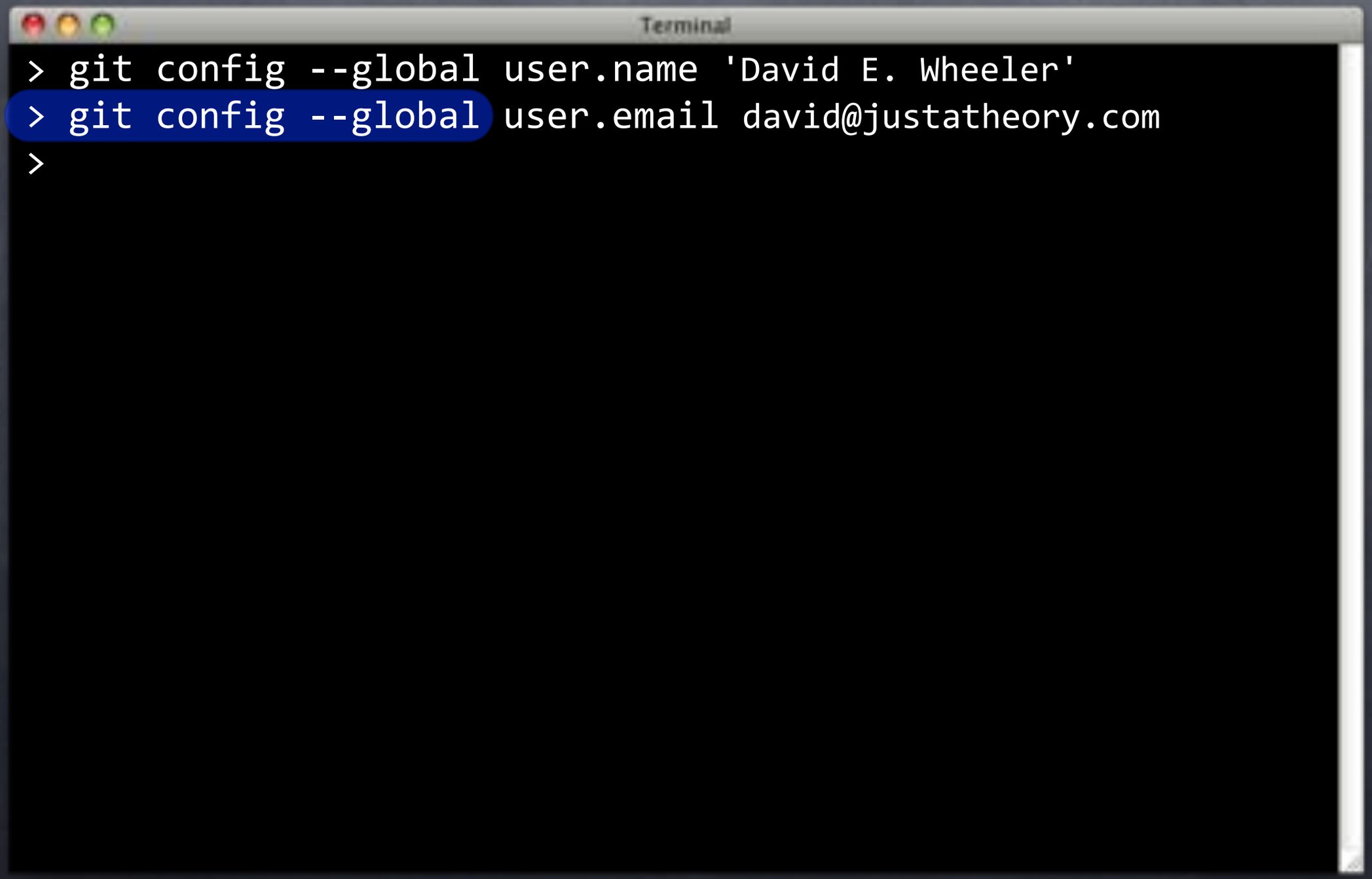


Who Am I?

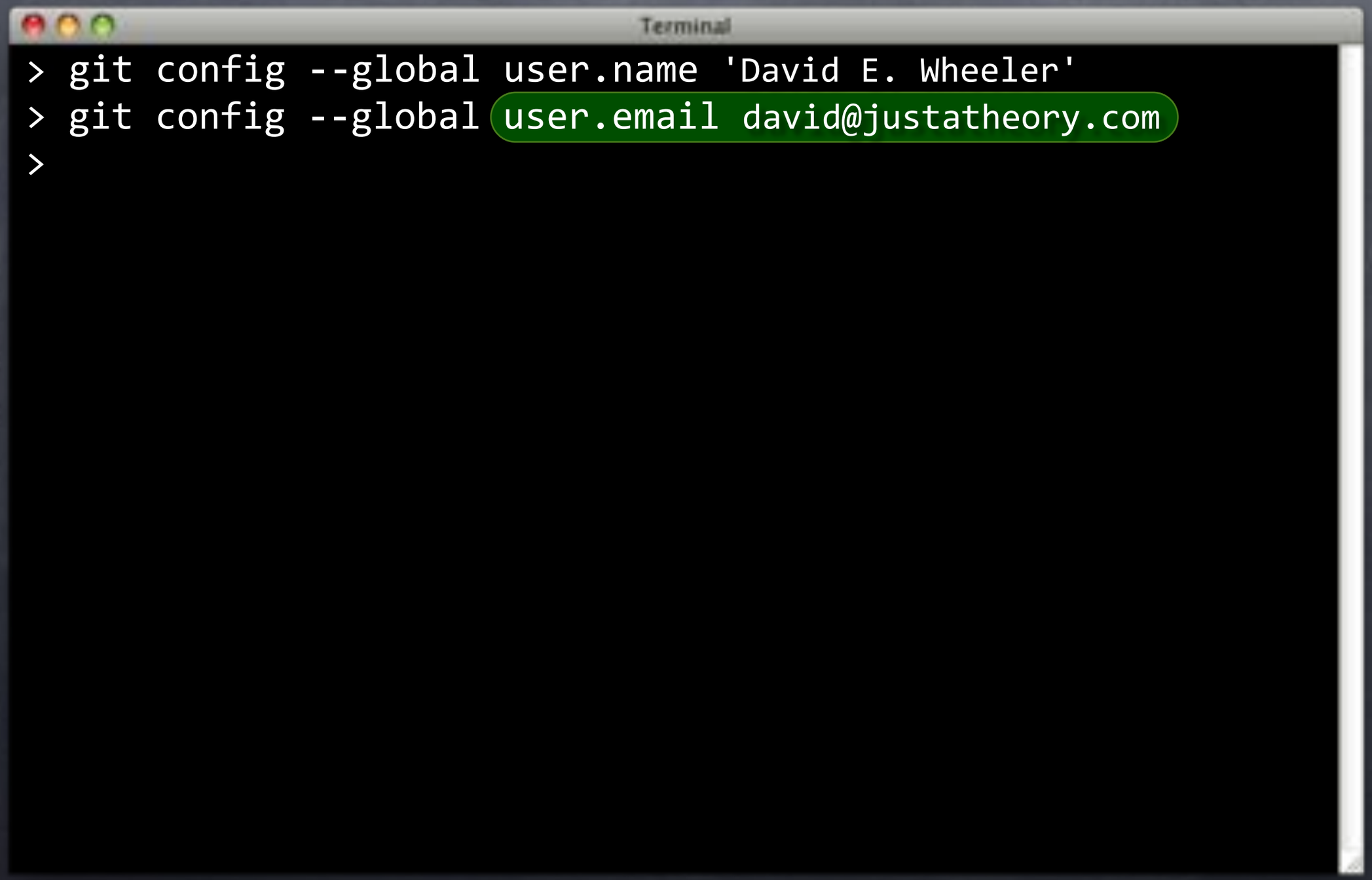
Who Am I?



Who Am I?


Terminal
> git config --global user.name 'David E. Wheeler'
> git config --global user.email david@justatheory.com
>

Who Am I?


Terminal
> git config --global user.name 'David E. Wheeler'
> git config --global user.email david@justatheory.com
>

Who Am I?


Terminal
> git config --global user.name 'David E. Wheeler'
> git config --global user.email david@justatheory.com
>
Please use your own name and email.

Who Am I?

Terminal

```
> git config --global user.name 'David E. Wheeler'
> git config --global user.email david@justatheory.com
> emacs ~/.gitconfig
>
```

~/.gitconfig

Emacs

```
[user]
    name = David E. Wheeler
    email = david@justatheory.com
```

-:** ~/.gitconfig All (SQL[ansi])----------

~/.gitconfig

Emacs

```
[user]
    name = David E. Wheeler
    email = david@justatheory.com
```

Good for all projects

-:**- ~/.gitconfig All (SQL[ansi])----------

Create a Remote

Create a Remote

- Create Git project on GitHub

Create a Remote

- Create Git project on GitHub
- Or BitBucket

Create a Remote

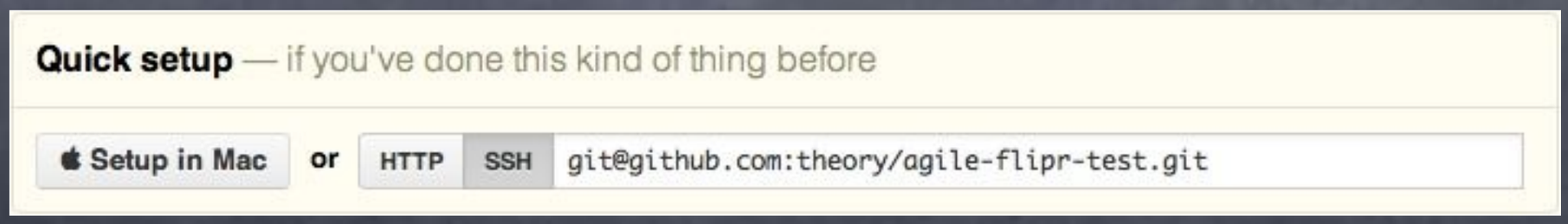
- Create Git project on GitHub
- Or BitBucket
- Wherever you like

Create a Remote

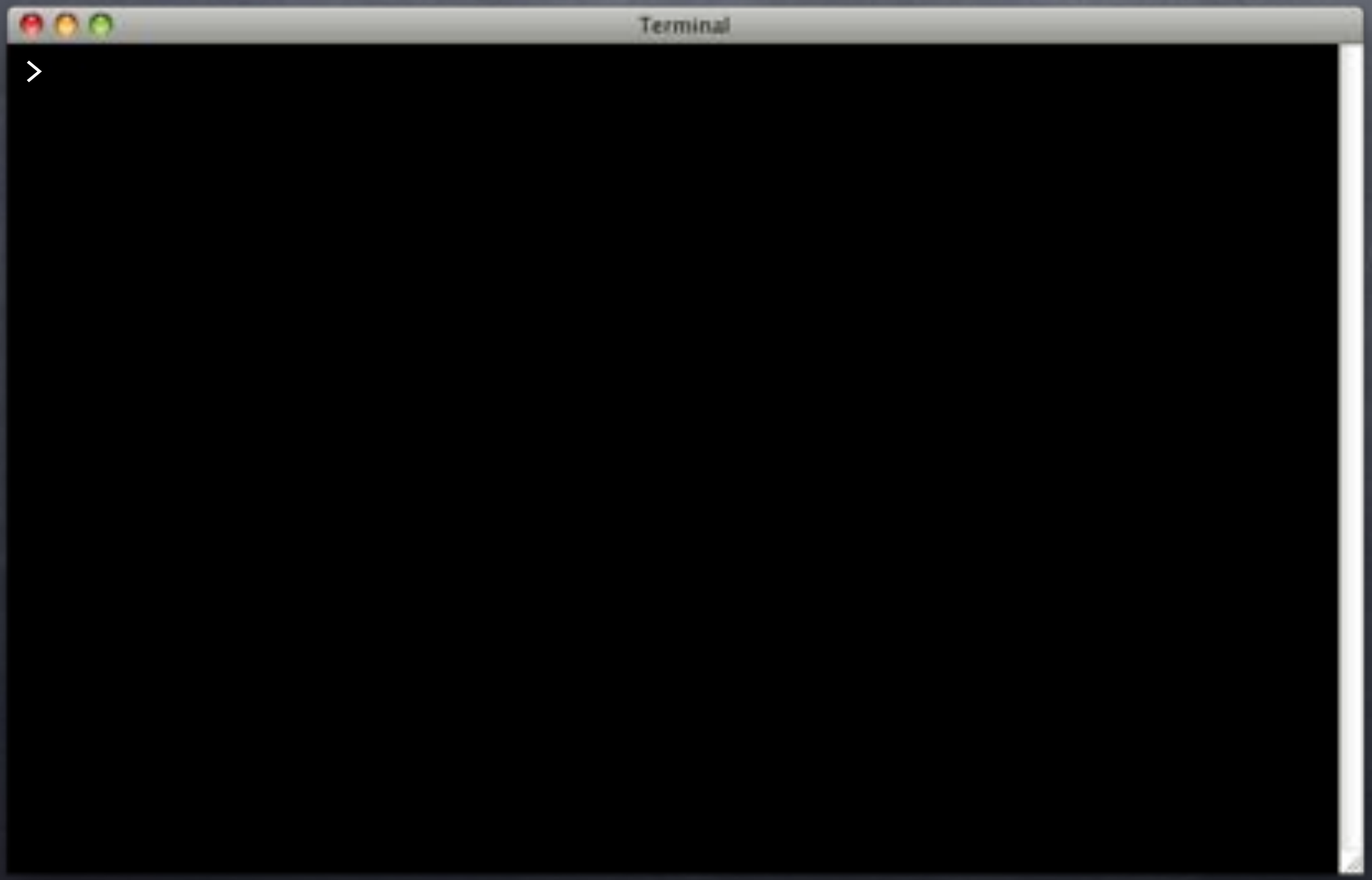
- Create Git project on GitHub
- Or BitBucket
- Wherever you like
- Record remote URL

Create a Remote

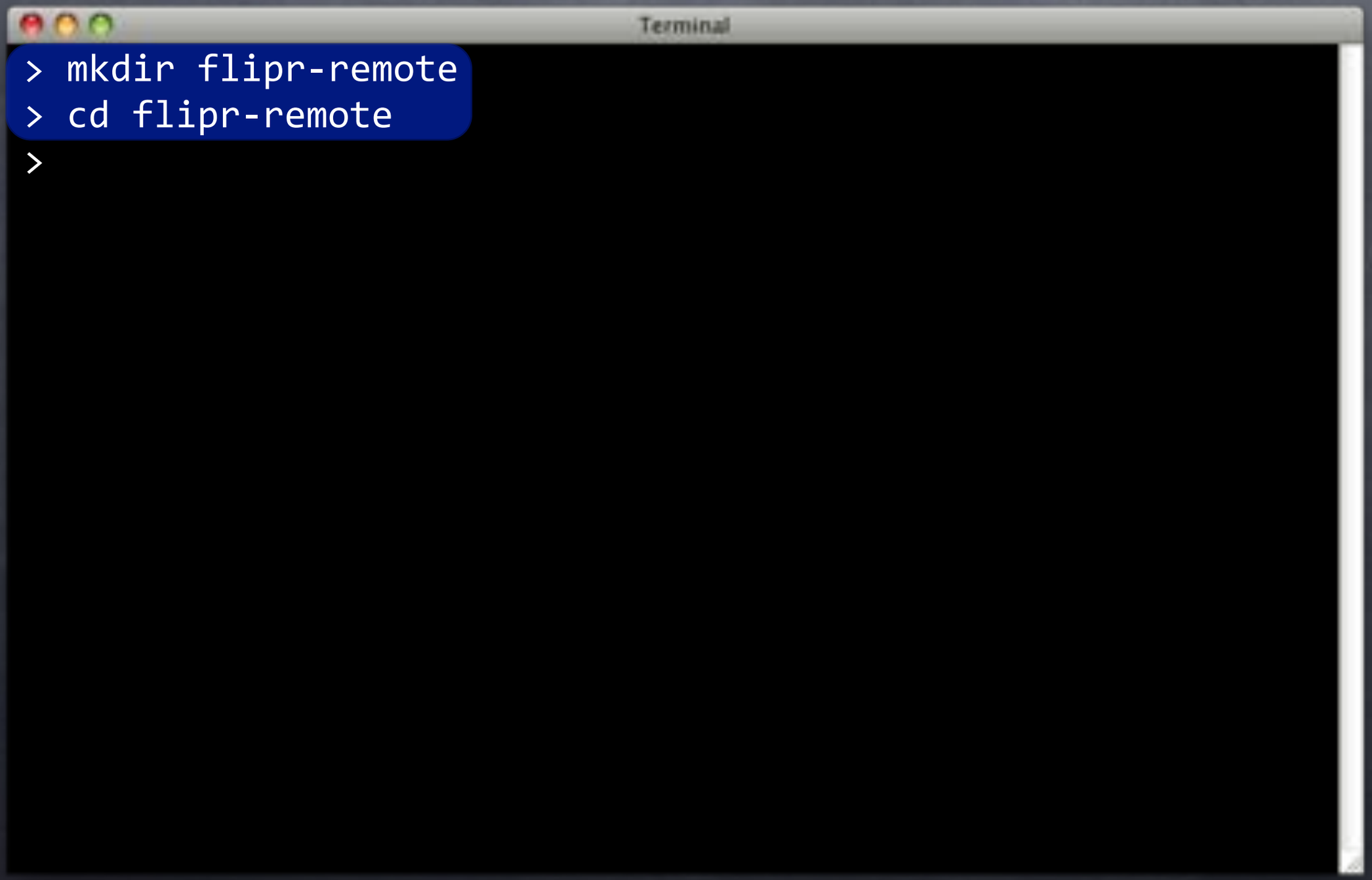
- Create Git project on GitHub
- Or BitBucket
- Wherever you like
- Record remote URL



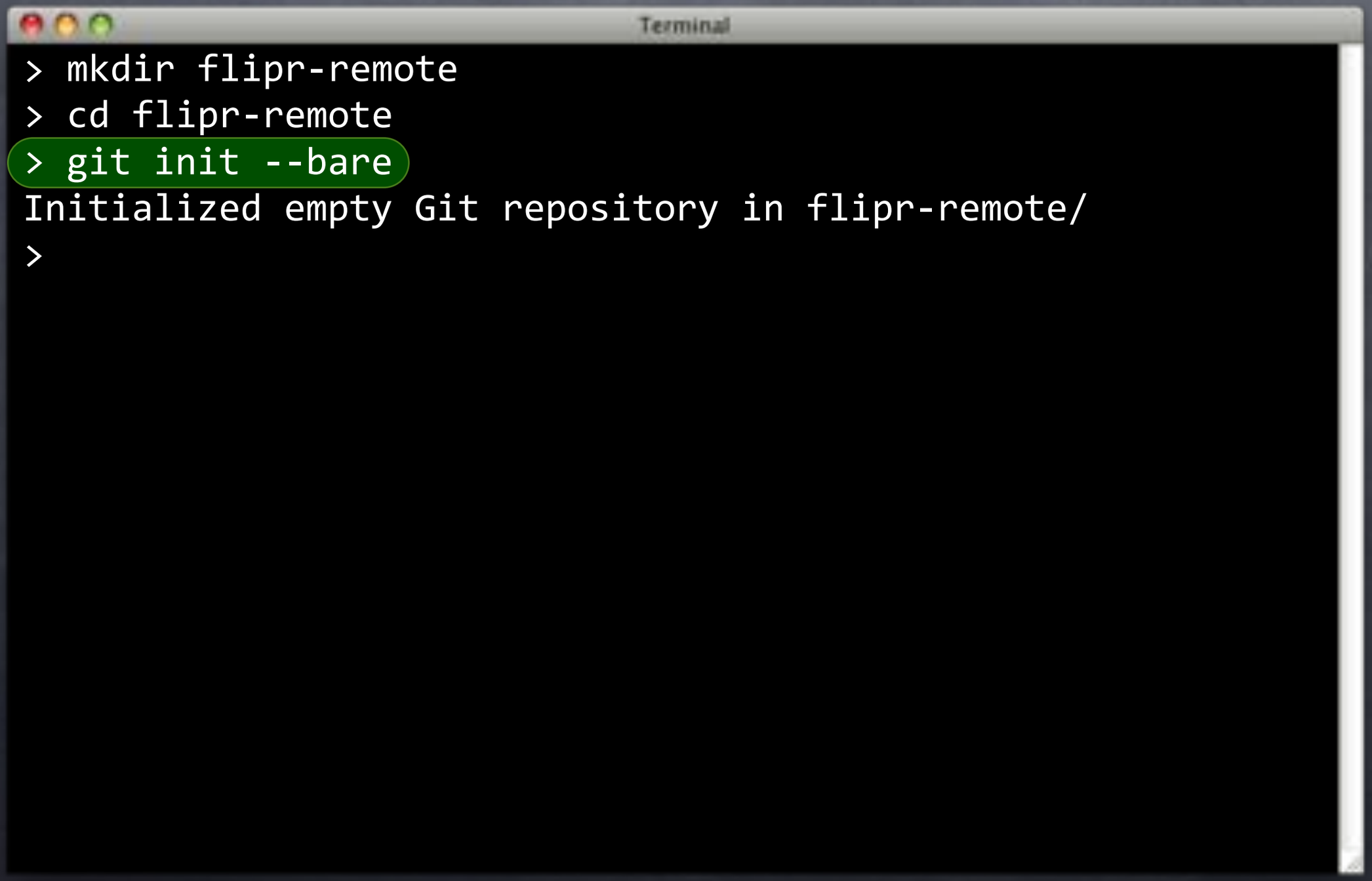
Or a Local Remote



Or a Local Remote



Or a Local Remote



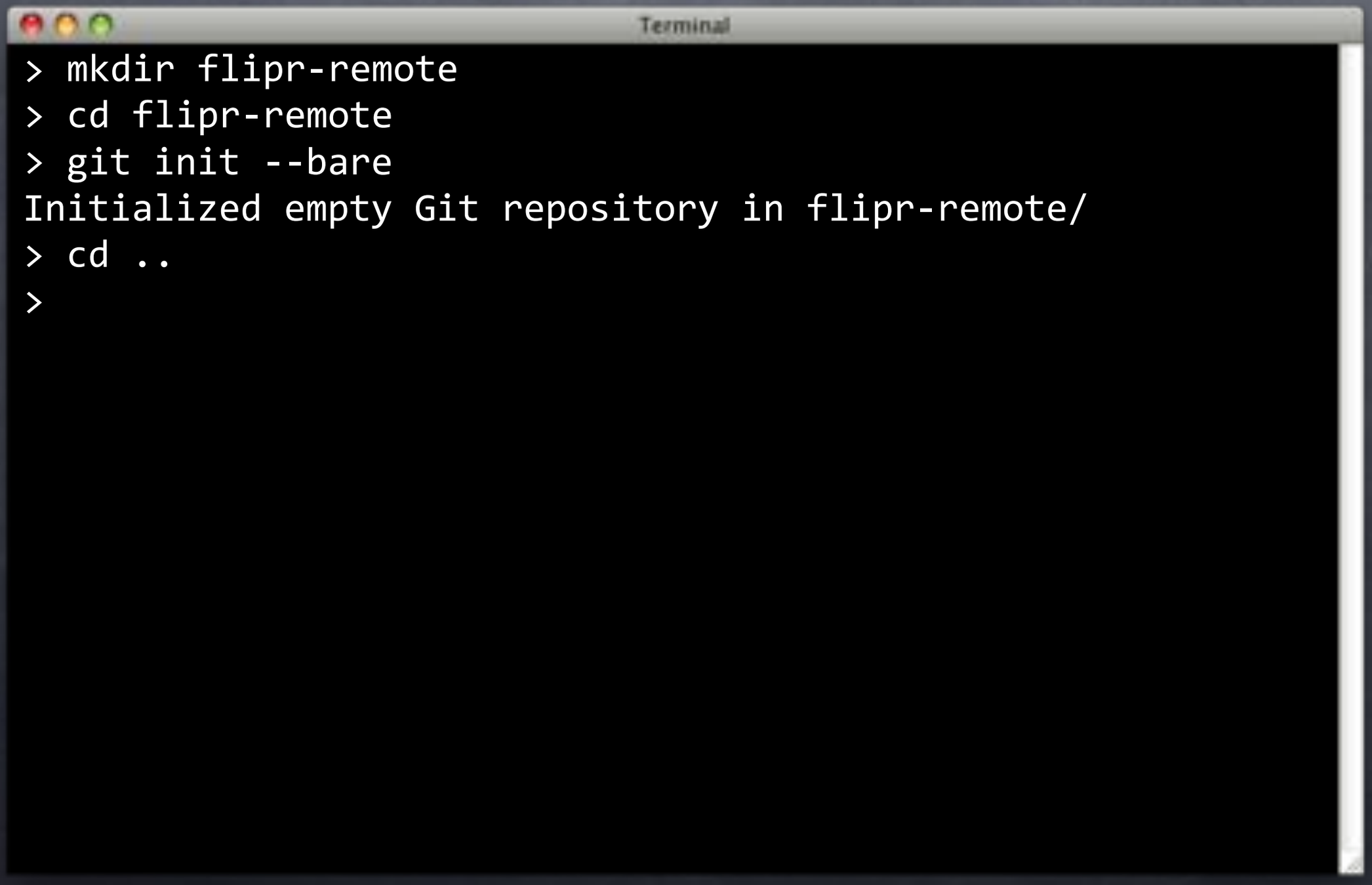
Or a Local Remote

Terminal

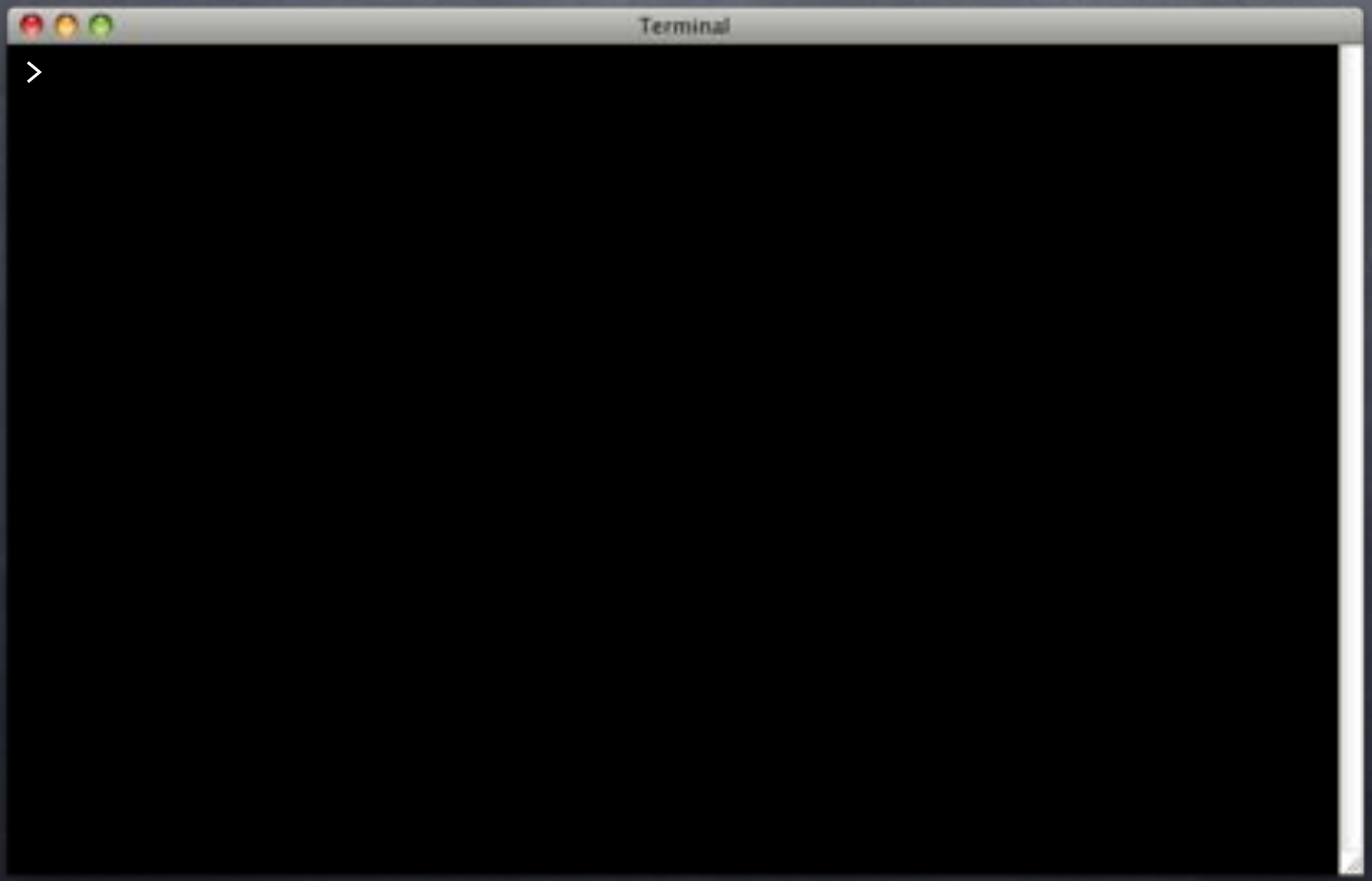
```
> mkdir flipr-remote
> cd flipr-remote
> git init --bare
Initialized empty Git repository in flipr-remote/
>
```

Remote URL is simply the path to this directory.

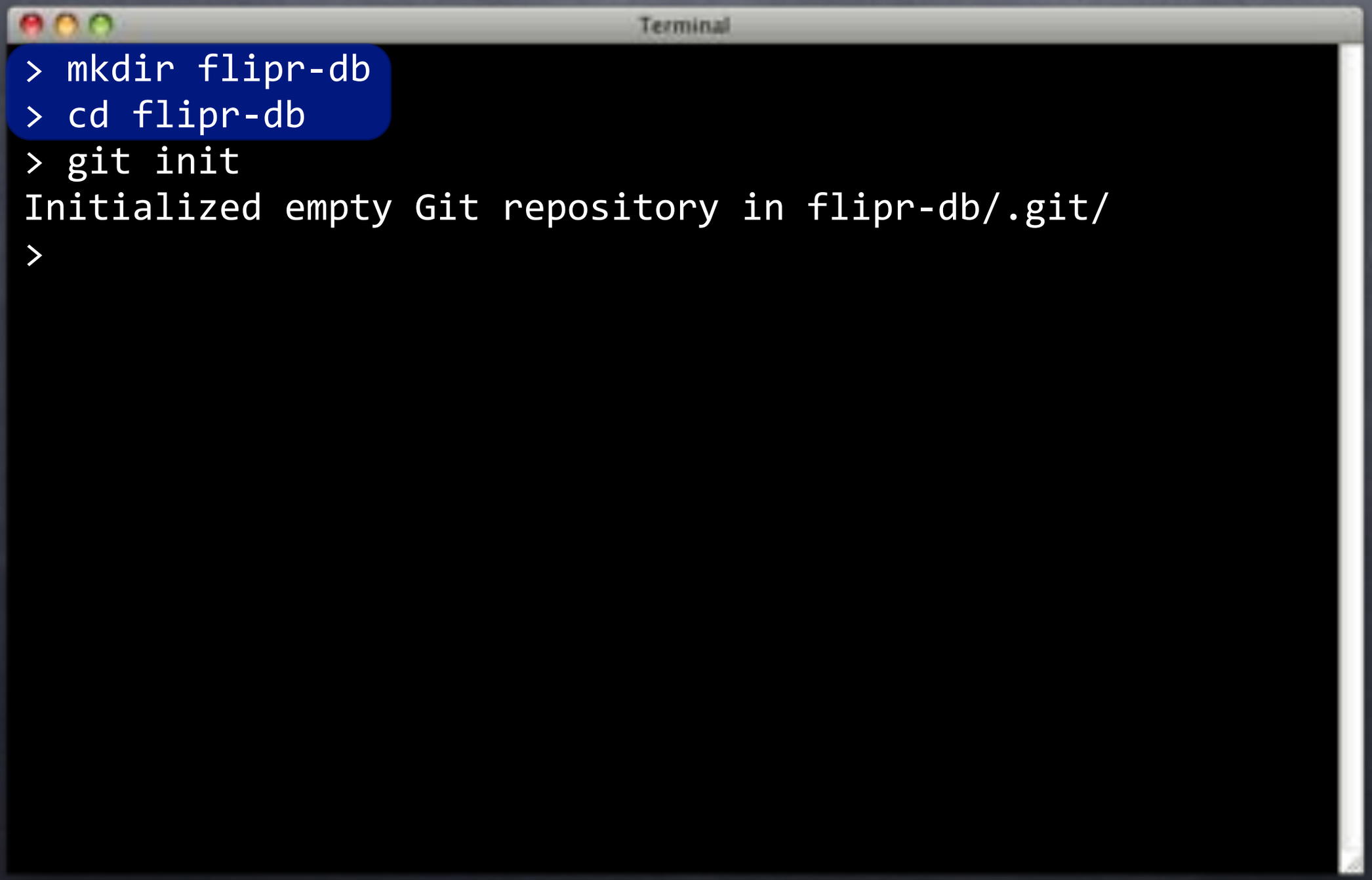
Or a Local Remote



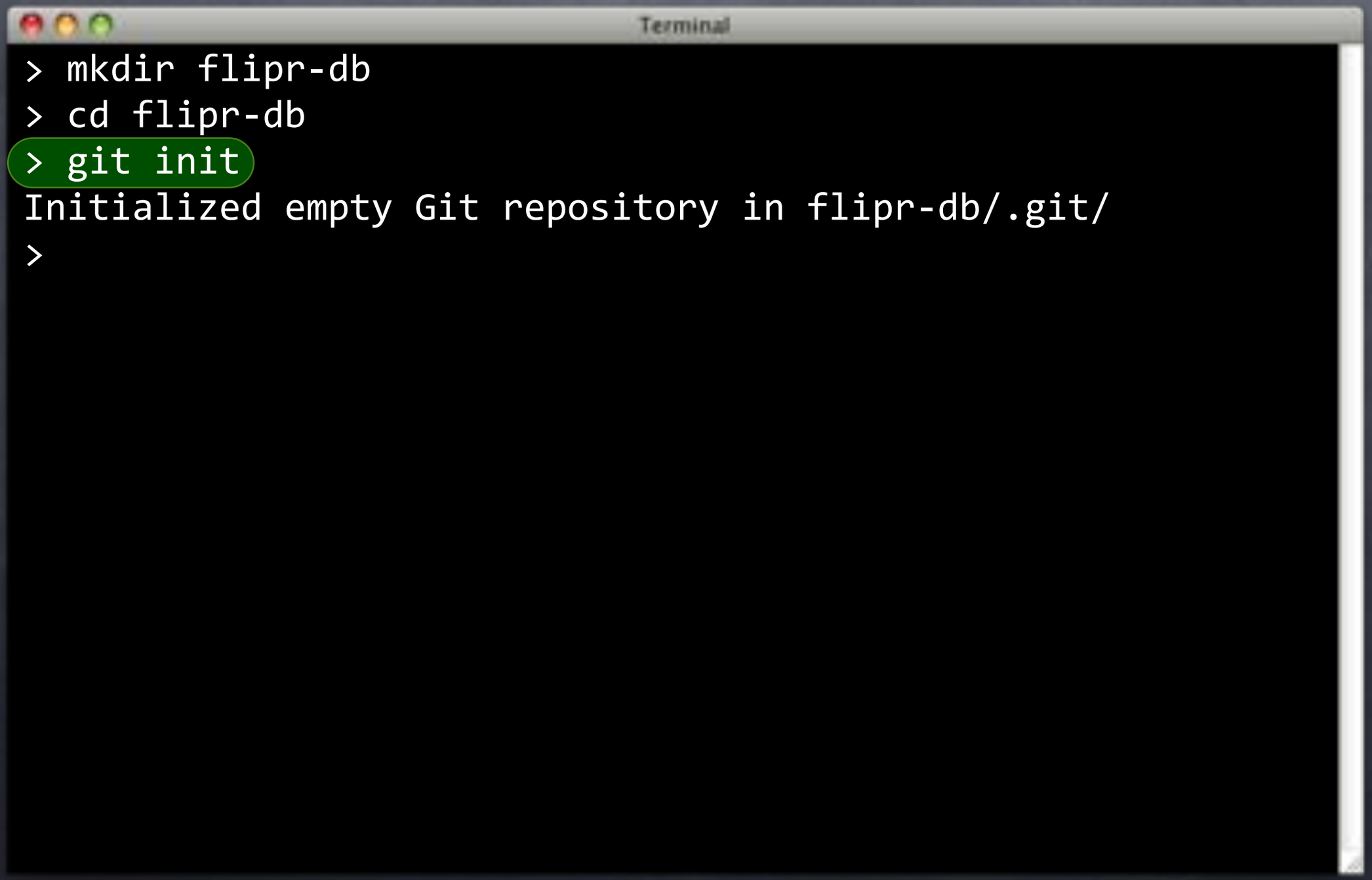
Gitiup



Gitiup


Terminal
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
>

Gitiup


Terminal
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
>

Gitiup

Terminal

```
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
> echo Flipr Database Project > README.md
>
```

Gitiup

Terminal

```
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
> echo Flipr Database Project > README.md
> git add .
>
>
```

Gitiup

Terminal

```
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
> echo Flipr Database Project > README.md
> git add .
> git commit -m 'Initialize repo, add README.'
[master (root-commit) 2c4b510] Initialize repo, add README.
1 file changed, 1 insertion(+)
 create mode 100644 README.md
>
```

Gitiup

Terminal

```
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
> echo Flipr Database Project > README.md
> git add .
> git commit -m 'Initialize repo, add README.'
[master (root-commit) 2c4b510] Initialize repo, add README.
1 file changed, 1 insertion(+)
 create mode 100644 README.md
>
```

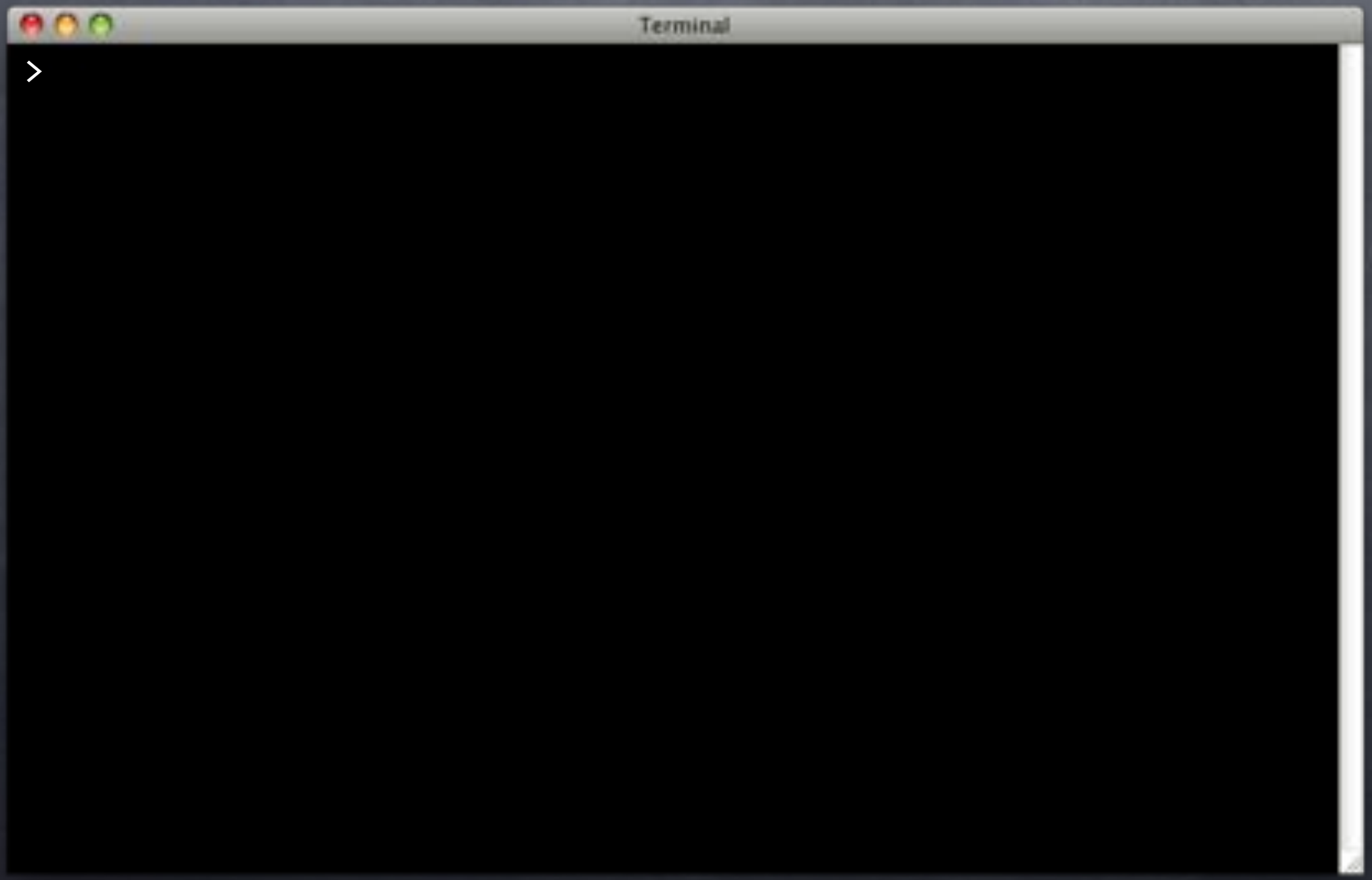
Gitiup

Terminal

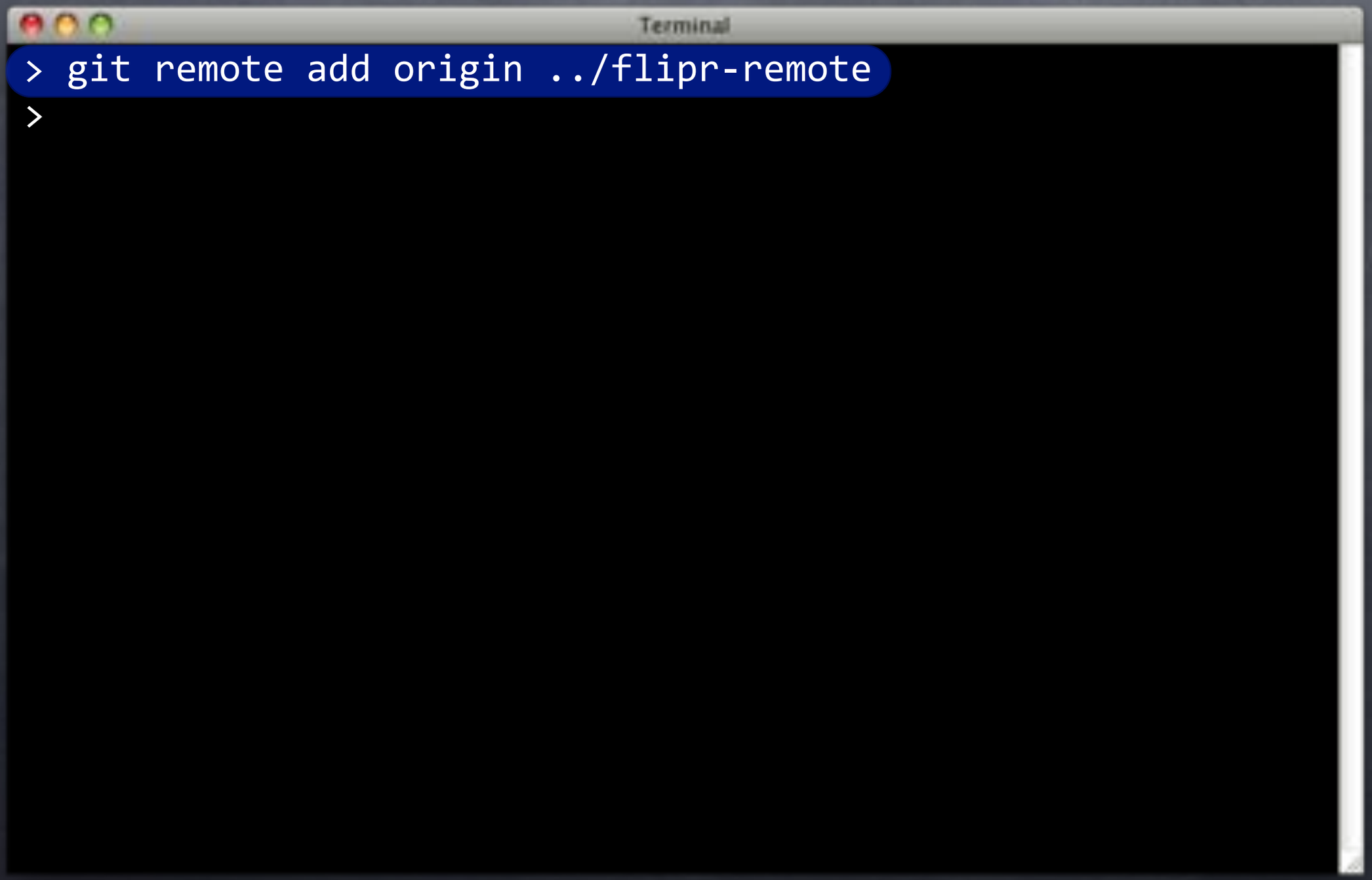
```
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
> echo Flipr Database Project > README.md
> git add .
> git commit -m 'Initialize repo, add README.'
[master (root-commit) 2c4b510] Initialize repo, add README.
1 file changed, 1 insertion(+)
 create mode 100644 README.md
>
```

We have a Git repo.

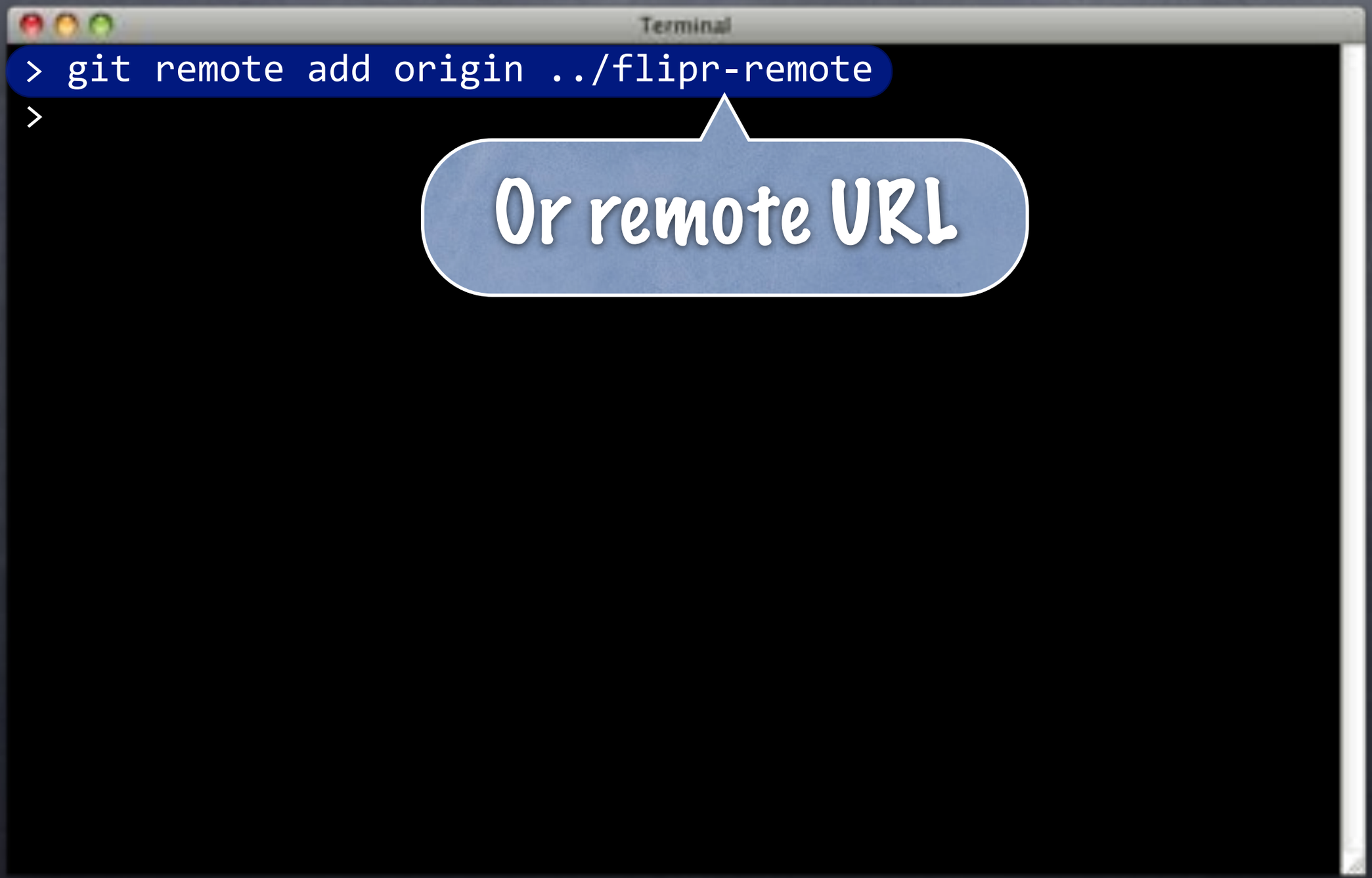
Origin



Origin



Origin



Origin

Terminal

```
> git remote add origin ../flipr-remote
> git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 260 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      master -> master
Branch master set up to track remote branch master from
origin.
>
```

Origin

Terminal

```
> git remote add origin ../flipr-remote
> git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 260 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      master -> master
Branch master set up to track remote branch master from
origin.
>
```

Origin

Terminal

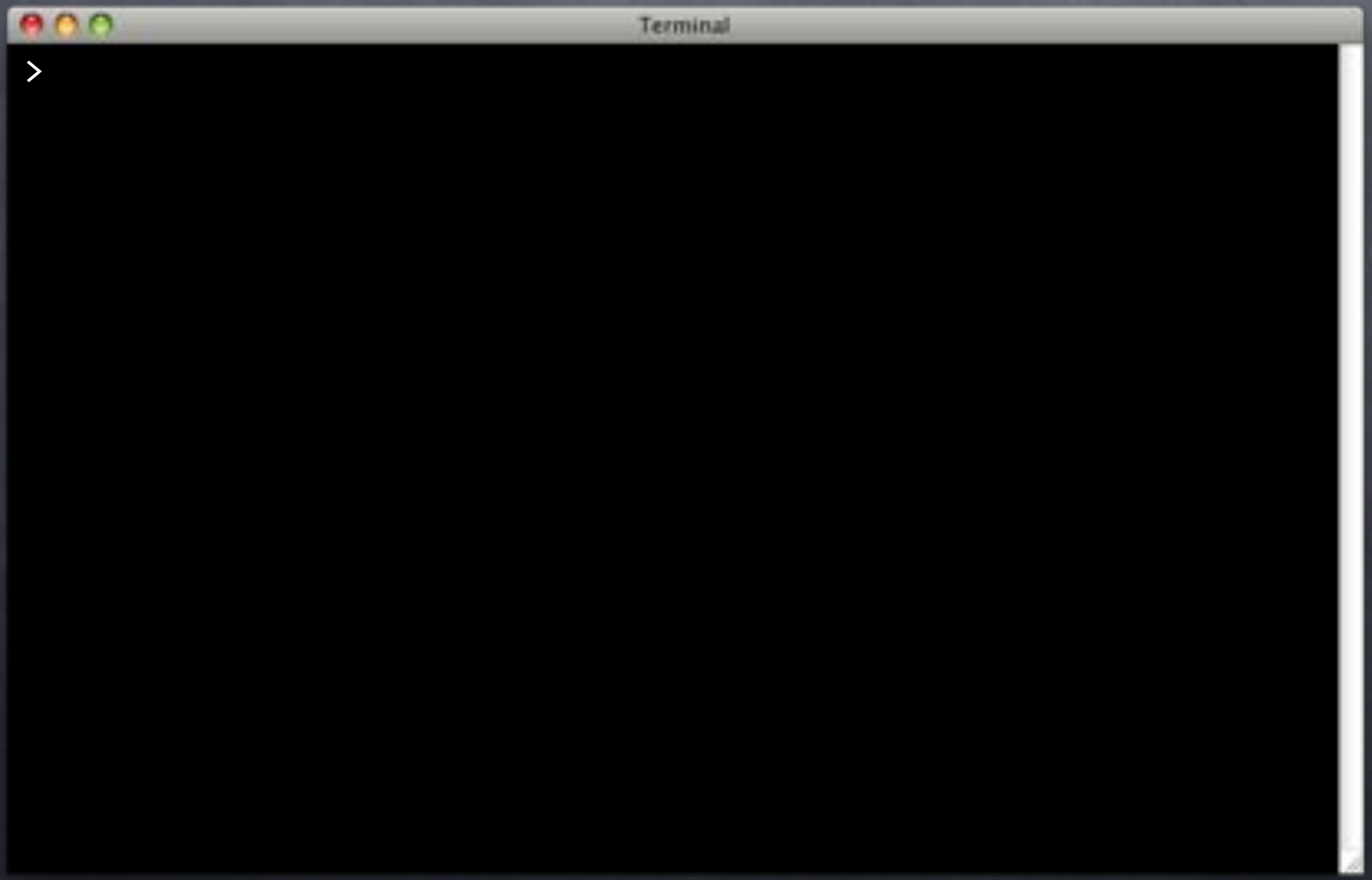
```
> git remote add origin ../flipr-remote
> git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 260 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      master -> master
Branch master set up to track remote branch master from
origin.
>
```

Origin

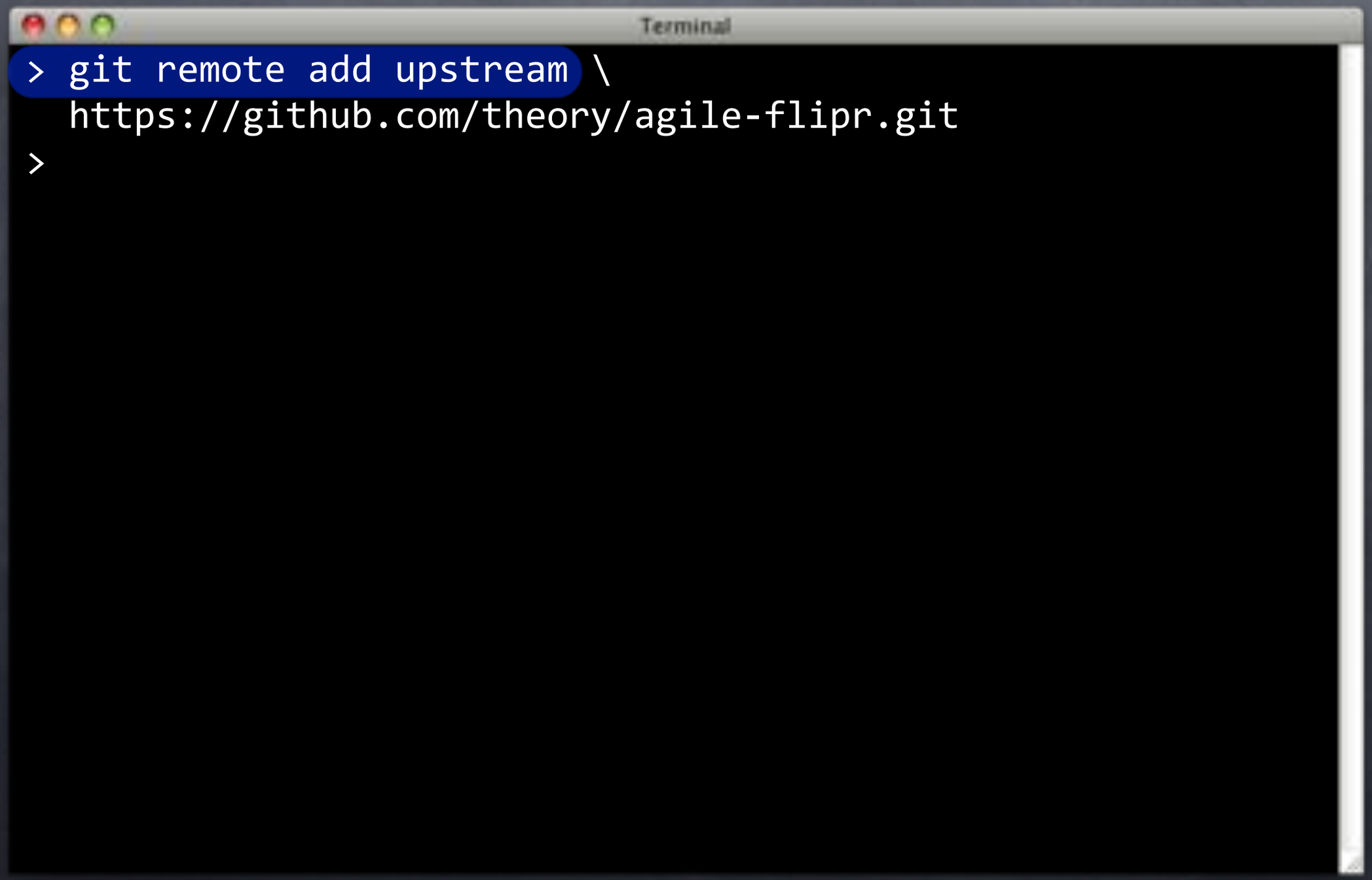
Terminal

```
> git remote add origin ../flipr-remote
> git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 260 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      master -> master
Branch master set up to track remote branch master from
origin.
>
```

Swimming Upstream



Swimming Upstream



Swimming Upstream

Terminal

```
> git remote add upstream \
  https://github.com/theory/agile-flipr.git
> git fetch upstream
From https://github.com/theory/agile-flipr
 * [new branch]      flips      -> upstream/flips
 * [new branch]      master     -> upstream/master
 * [new branch]      userfuncs  -> upstream/userfuncs
 * [new branch]      users      -> upstream/users
 * [new tag]         insert_user -> insert_user
 * [new tag]         appschema -> appschema
 * [new tag]         v1.0.0-r1  -> v1.0.0-r1
 * [new tag]         reltag     -> reltag
 * [new tag]         insert_user2 -> insert_user2
>
```

Swimming Upstream

Terminal

```
> git remote add upstream \
  https://github.com/theory/agile-flipr.git
> git fetch upstream
From https://github.com/theory/agile-flipr
 * [new branch]      flips      -> upstream/flips
 * [new branch]      master     -> upstream/master
 * [new branch]      userfuncs  -> upstream/userfuncs
 * [new branch]      users      -> upstream/users
 * [new tag]         insert_user -> insert_user
 * [new tag]         appschema -> appschema
 * [new tag]         v1.0.0-r1  -> v1.0.0-r1
 * [new tag]         reltag     -> reltag
 * [new tag]         insert_user2 -> insert_user2
>
```

For resetting later

Git?

Git?

- Manage tree of files over time

Git?

- Manage tree of files over time
- Distributed development

Git?

- Manage tree of files over time
- Distributed development
- Commit changes locally

Git?

- Manage tree of files over time
- Distributed development
- Commit changes locally
- Merge to remote (origin, upstream)

Git?

- Manage tree of files over time
- Distributed development
- Commit changes locally
- Merge to remote (origin, upstream)
- Speedy, responsive

Git?

- Manage tree of files over time
- Distributed development
- Commit changes locally
- Merge to remote (origin, upstream)
- Speedy, responsive
- Flexible, robust

Why Git?

Why Git?

- Anyone can clone

Why Git?

- Anyone can clone
- Complete repository copy

Why Git?

- Anyone can clone
- Complete repository copy
- Cheap branching

Why Git?

- Anyone can clone
- Complete repository copy
- Cheap branching
- Make and test local changes

Why Git?

- Anyone can clone
- Complete repository copy
- Cheap branching
- Make and test local changes
- Submit patches, pull requests

Why Git?

- Anyone can clone
- Complete repository copy
- Cheap branching
- Make and test local changes
- Submit patches, pull requests
- Pull from upstream

PiSHAi

PiSHA1

- SHA1 ID for every object

PiSHA1

- SHA1 ID for every object
  - commit, tag, tree, blob

PiSHA1

- SHA1 ID for every object
  - commit, tag, tree, blob
- Hashed commit text includes:

PiSHA1

- SHA1 ID for every object
  - commit, tag, tree, blob
- Hashed commit text includes:
  - tree ID

PiSHA1

- SHA1 ID for every object
  - commit, tag, tree, blob
- Hashed commit text includes:
  - tree ID
  - parent ID

PiSHA1

- SHA1 ID for every object
  - commit, tag, tree, blob
- Hashed commit text includes:
  - tree ID
  - parent ID
  - author

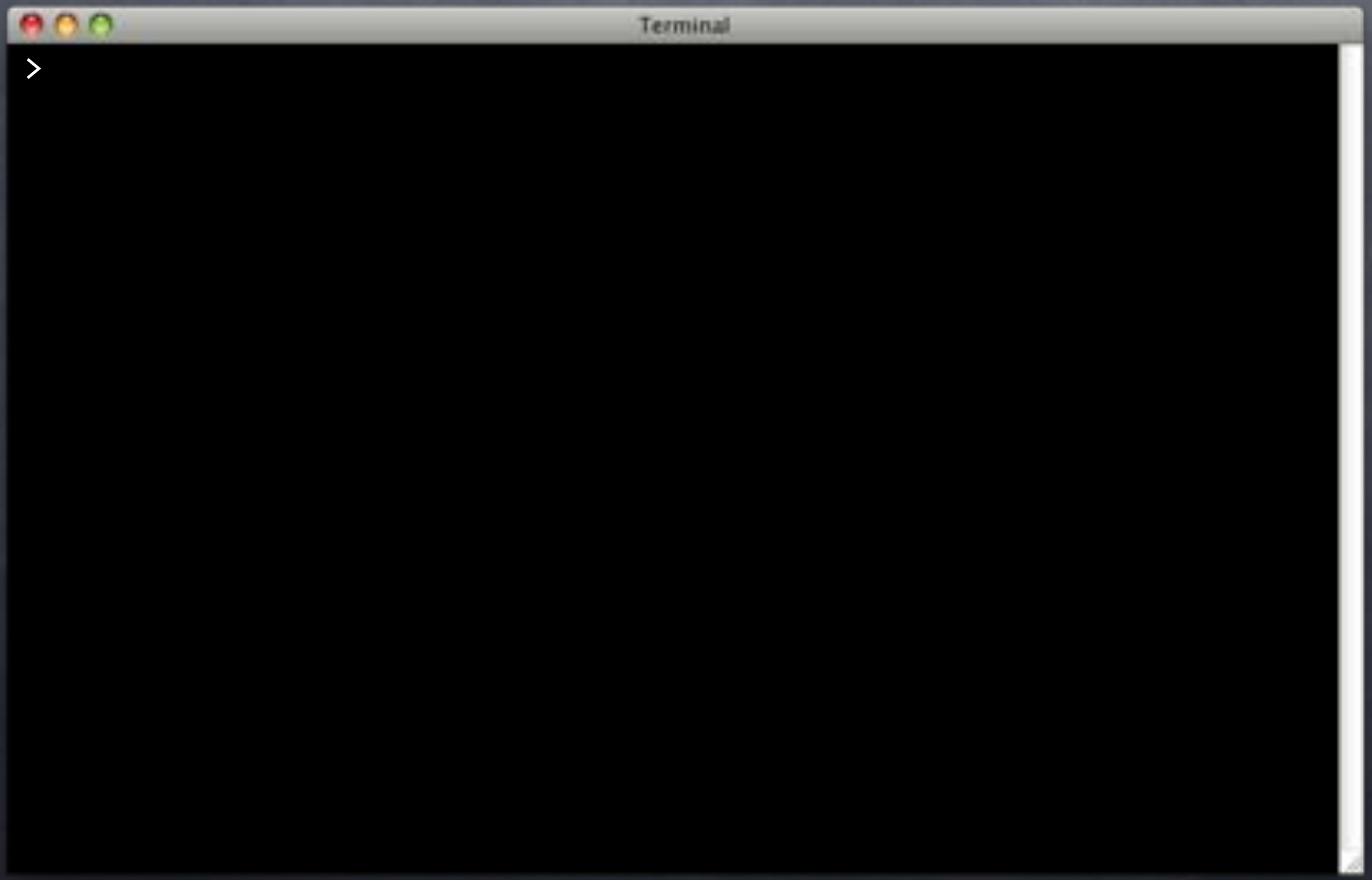
PiSHA1

- SHA1 ID for every object
  - commit, tag, tree, blob
- Hashed commit text includes:
  - tree ID
  - parent ID
  - author
  - committer

PiSHA1

- SHA1 ID for every object
  - commit, tag, tree, blob
- Hashed commit text includes:
  - tree ID
  - parent ID
  - author
  - committer
  - message

Making a hash of it

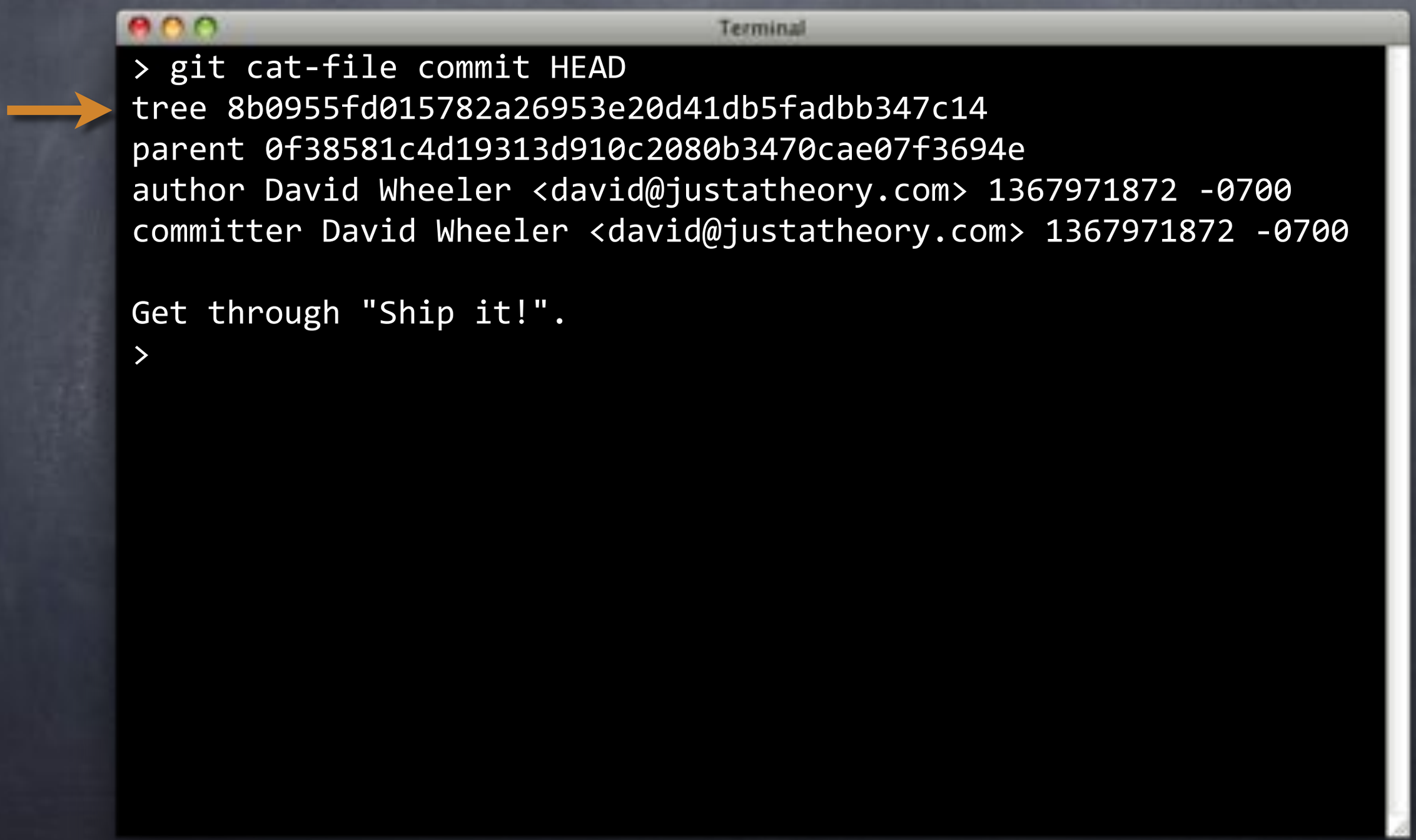


Making a hash of it

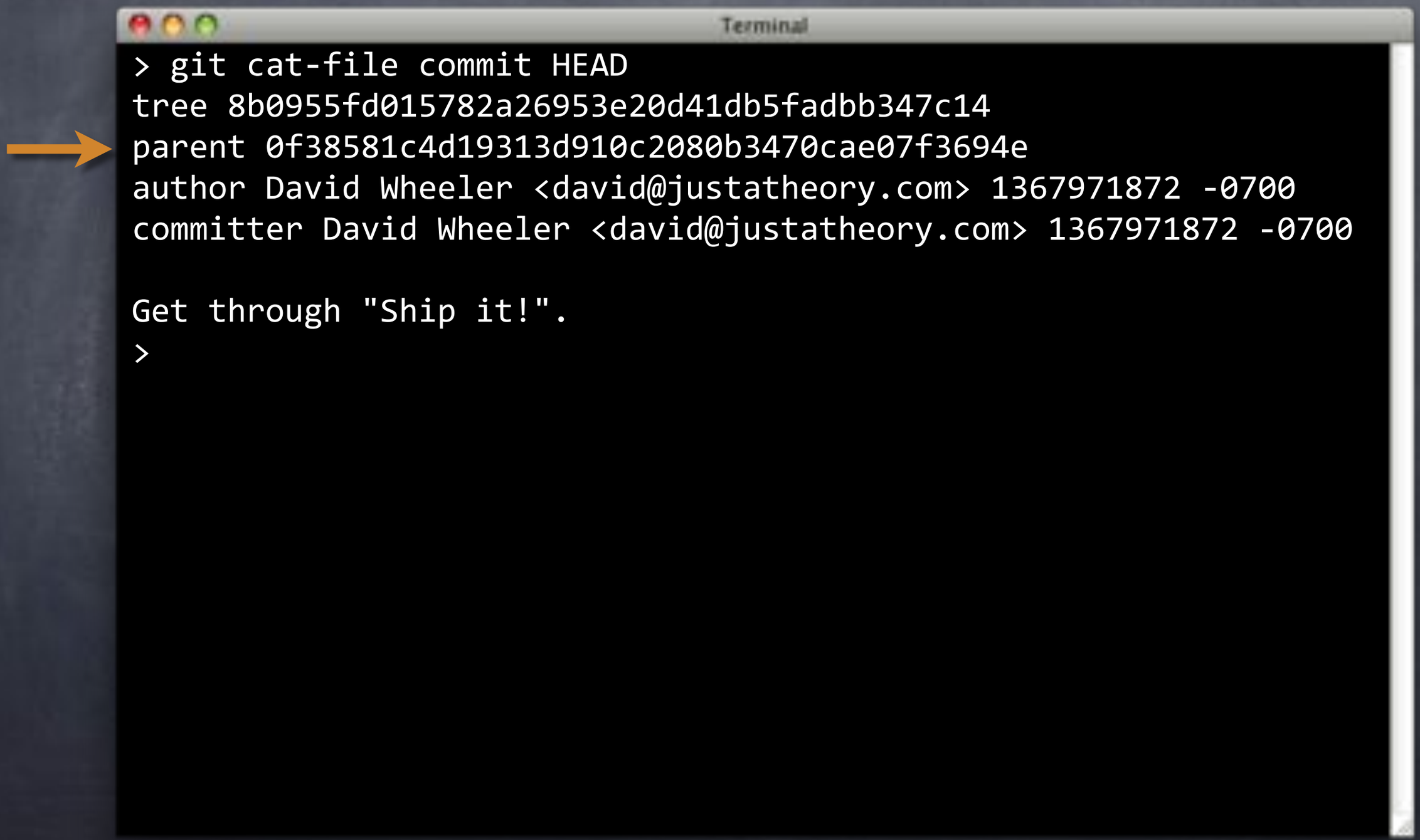
Terminal

```
> git cat-file commit HEAD
tree 8b0955fd015782a26953e20d41db5fadbb347c14
parent 0f38581c4d19313d910c2080b3470cae07f3694e
author David Wheeler <david@justatheory.com> 1367971872 -0700
committer David Wheeler <david@justatheory.com> 1367971872 -0700

Get through "Ship it!".
>
```

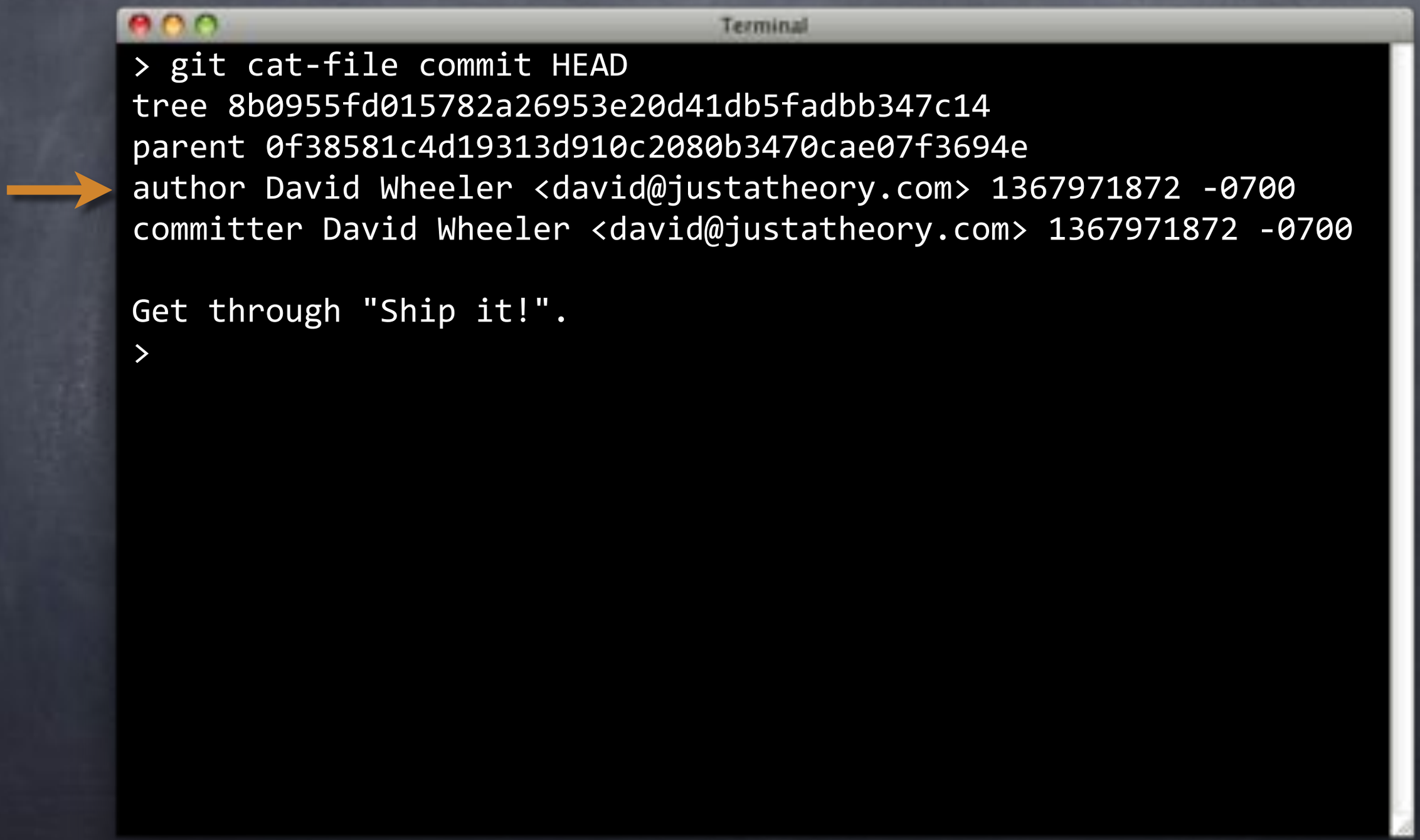
Making a hash of it



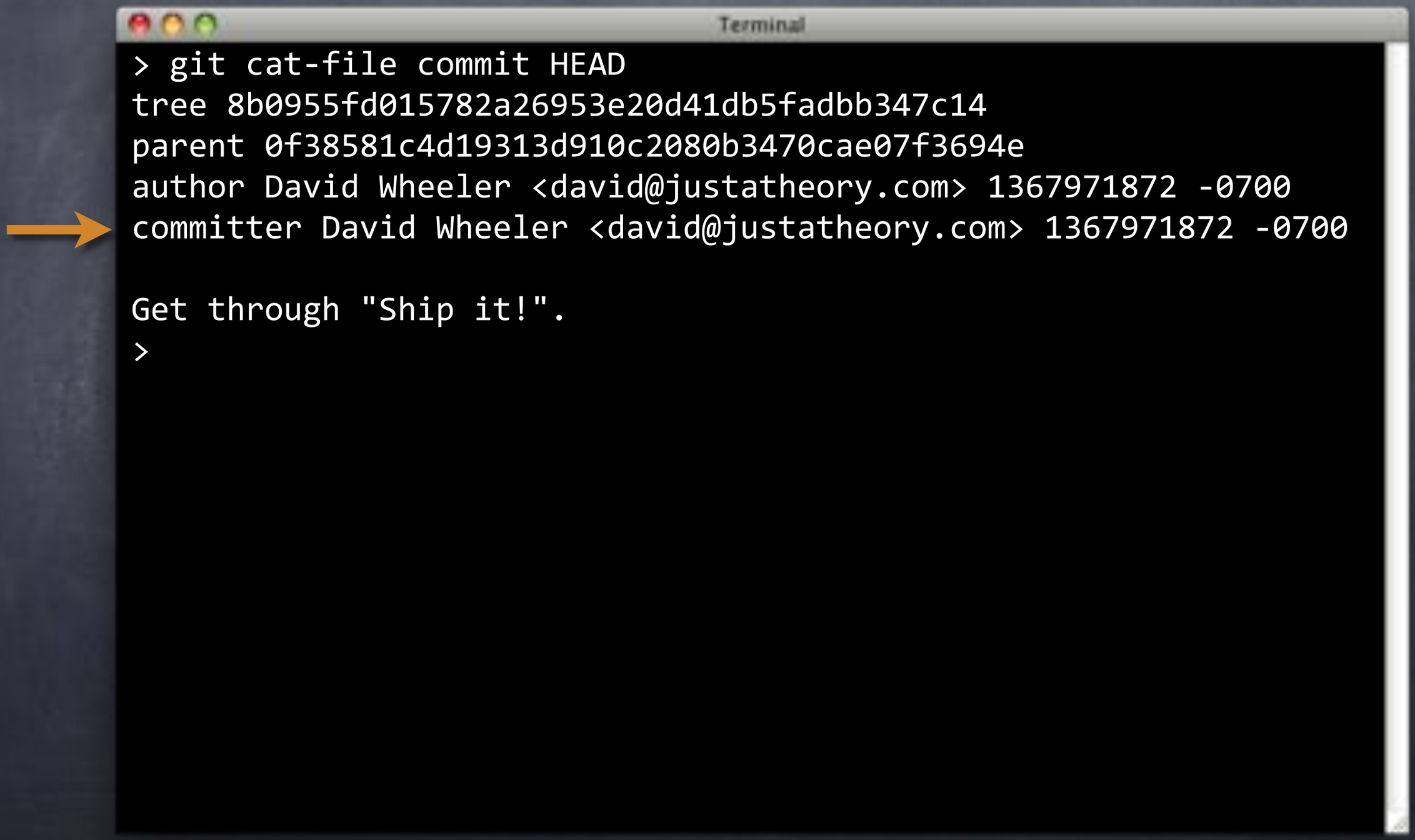
Making a hash of it



Making a hash of it



Making a hash of it



Making a hash of it

Terminal

```
> git cat-file commit HEAD
tree 8b0955fd015782a26953e20d41db5fadbb347c14
parent 0f38581c4d19313d910c2080b3470cae07f3694e
author David Wheeler <david@justatheory.com> 1367971872 -0700
committer David Wheeler <david@justatheory.com> 1367971872 -0700

Get through "Ship it!".
>
```

SHAzam!

SHAzam!

- Each commit (except first) includes parent

SHAzam!

- Each commit (except first) includes parent
- Can trace from any commit to the beginning

SHAzam!

- Each commit (except first) includes parent
- Can trace from any commit to the beginning
- Tampering (corruption) detectable

SHAzam!

- Each commit (except first) includes parent
- Can trace from any commit to the beginning
- Tampering (corruption) detectable
  - Because the hash will be wrong

SHAzam!

- Each commit (except first) includes parent
- Can trace from any commit to the beginning
- Tampering (corruption) detectable
  - Because the hash will be wrong
- Linus Torvalds's "greatest invention"
  http://perl.plover.com/yak/git/

Your Turn

Your Turn

- Configure Git

Your Turn

- Configure Git
- Initialize repository

Your Turn

- Configure Git
- Initialize repository
- Add origin remote

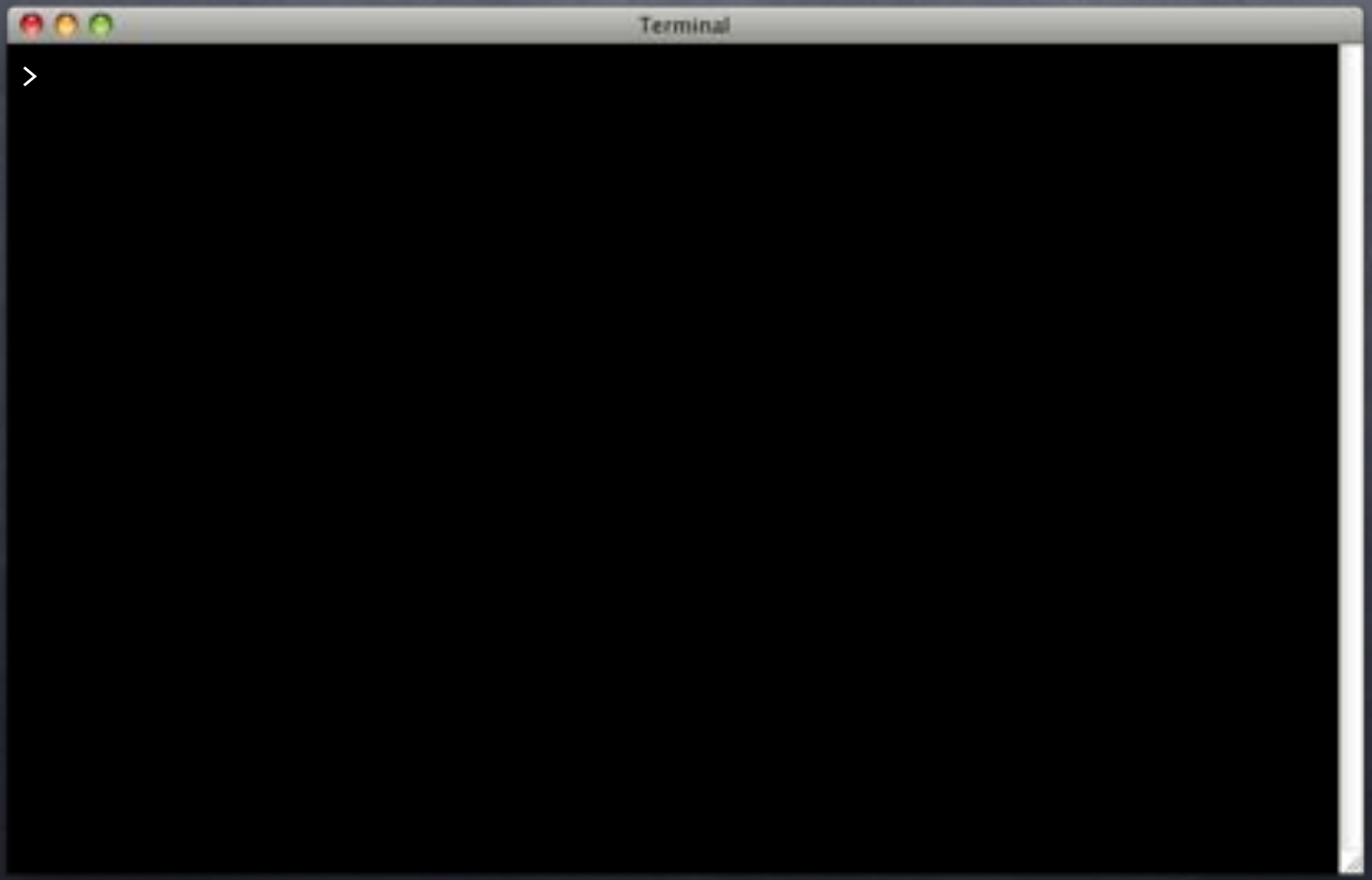
Your Turn

- Configure Git
- Initialize repository
- Add origin remote
- Add upstream remote

Your Turn

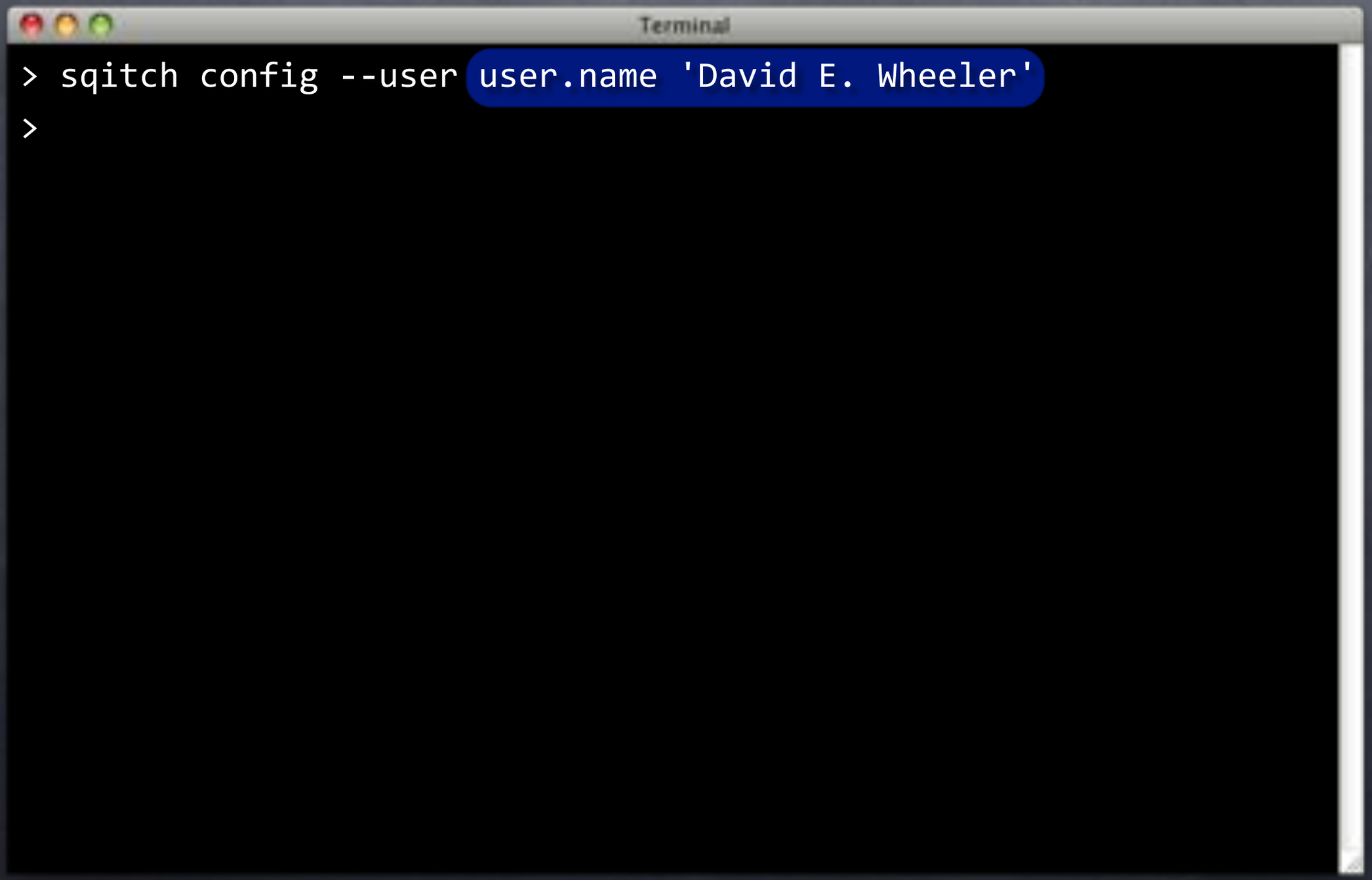
- Configure Git
- Initialize repository
- Add origin remote
- Add upstream remote
- https://github.com/theory/agile-flipr.git

Who am I again?

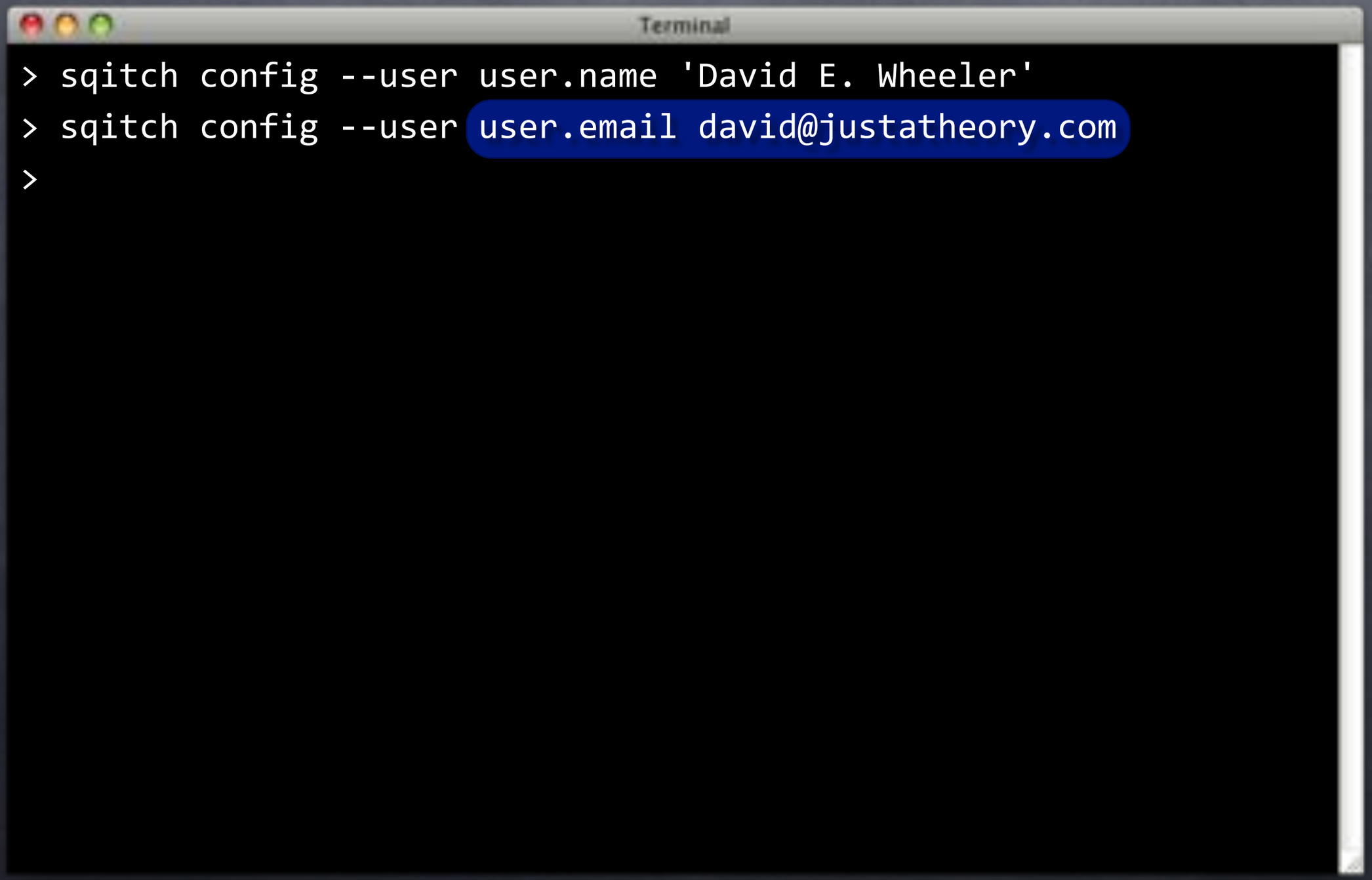


Who am I again?

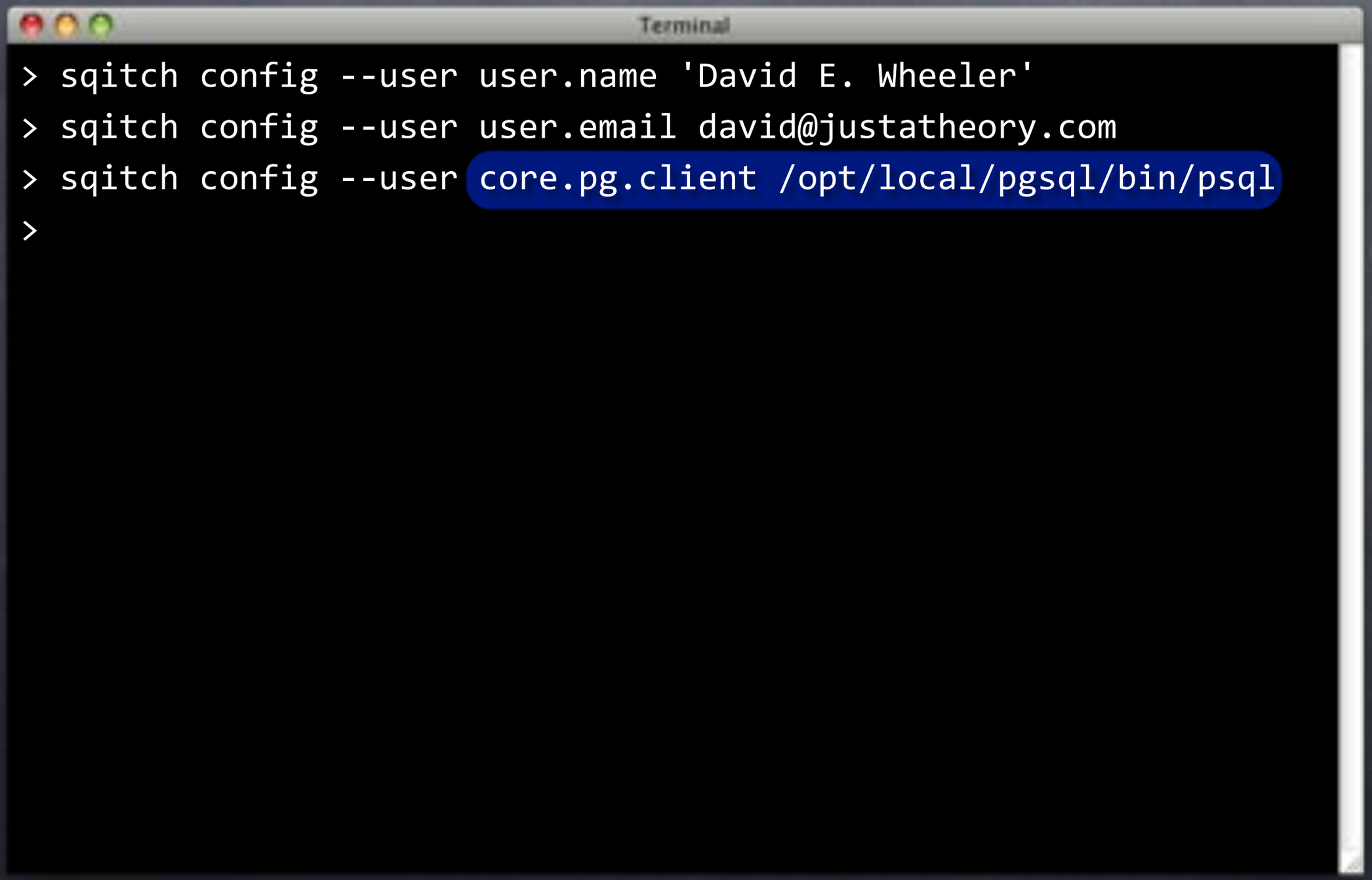
Who am I again?



Who am I again?



Who am I again?



Who am I again?

Terminal

```
> sqitch config --user user.name 'David E. Wheeler'
> sqitch config --user user.email david@justatheory.com
> sqitch config --user core.pg.client /opt/local/pgsql/bin/psql
> emacs ~/.sqitch/sqitch.conf
>
```

~/.sqitch/sqitch.conf

Emacs

```
[core "pg"]
    client = /opt/local/pgsql/bin/psql
[user]
    name = David E. Wheeler
    email = david@justatheory.com
```

-:** ~/.sqitch/sqitc All (SQL[ansi])----------

~/.sqitch/sqitch.conf

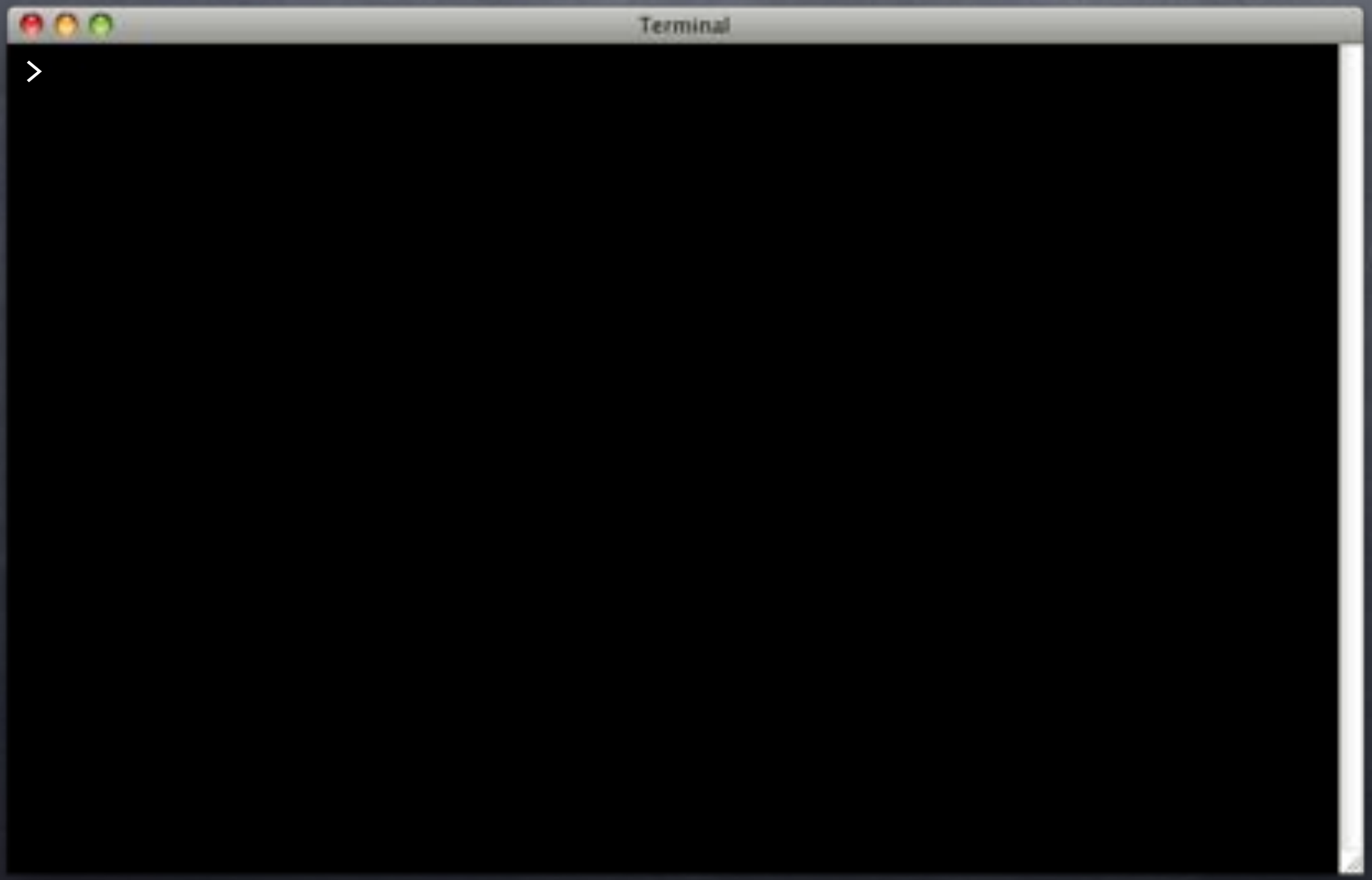
Emacs

```
[core "pg"]
    client = /opt/local/pgsql/bin/psql
[user]
    name = David E. Wheeler
    email = david@justatheory.com
```

Good for all projects

```
-:**-  ~/.sqitch/sqitc   All   (SQL[ansi])----------------------------------------------------------------------------
```

Scratch that Sqitch



Scratch that Sqitch

Scratch that Sqitch

Terminal

```
> sqitch --engine pg init flipr \
  --uri file://../flipr-remote
Created sqitch.conf
Created sqitch.plan
Created deploy/
Created revert/
Created verify/
>
```

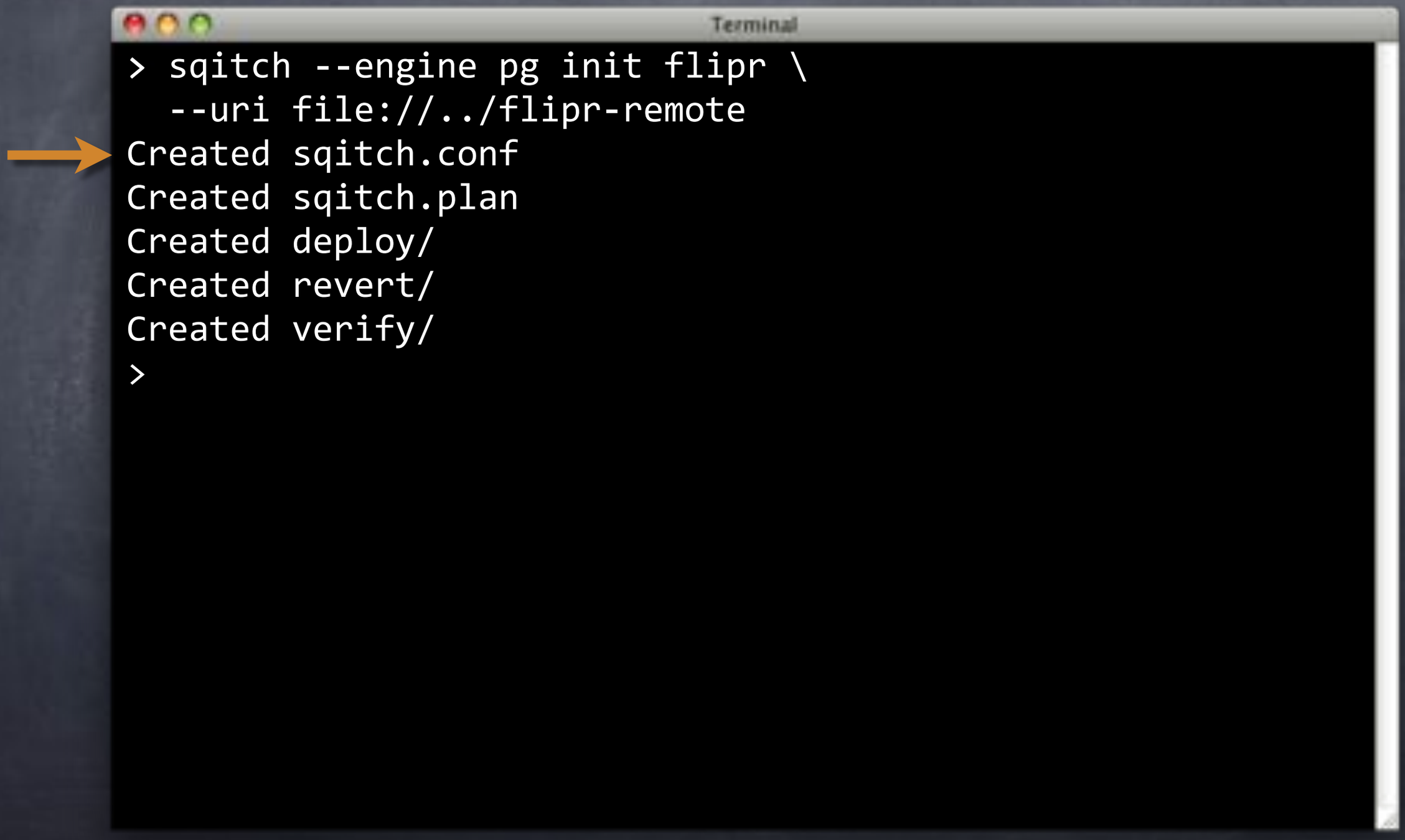
Scratch that Sqitch

Terminal

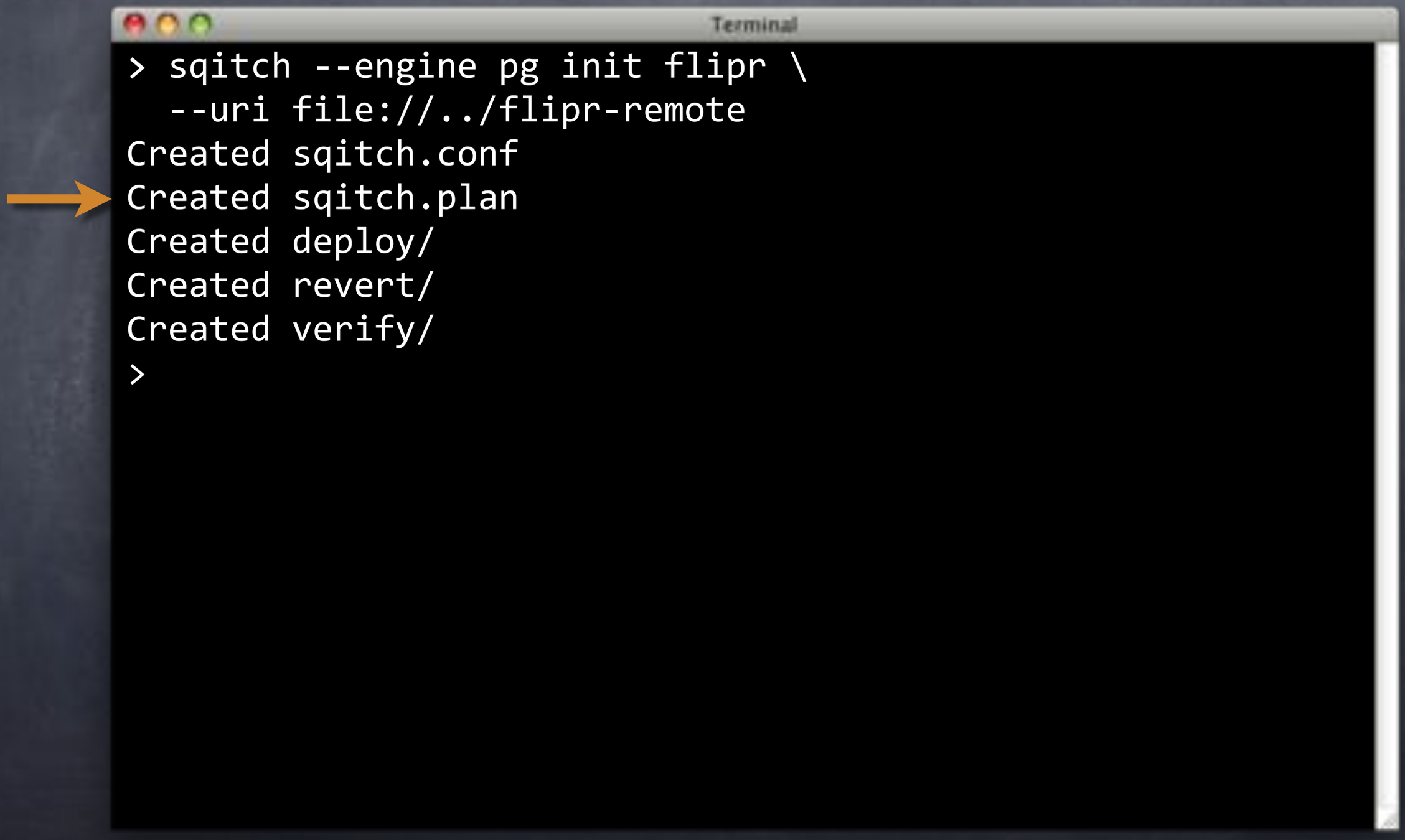
```
> sqitch --engine pg init flipr \
  --uri file://../flipr-remote
Created sqitch.conf
Created sqitch.plan
Created deploy/
Created revert/
Created verify/
>
```

Use remote URL

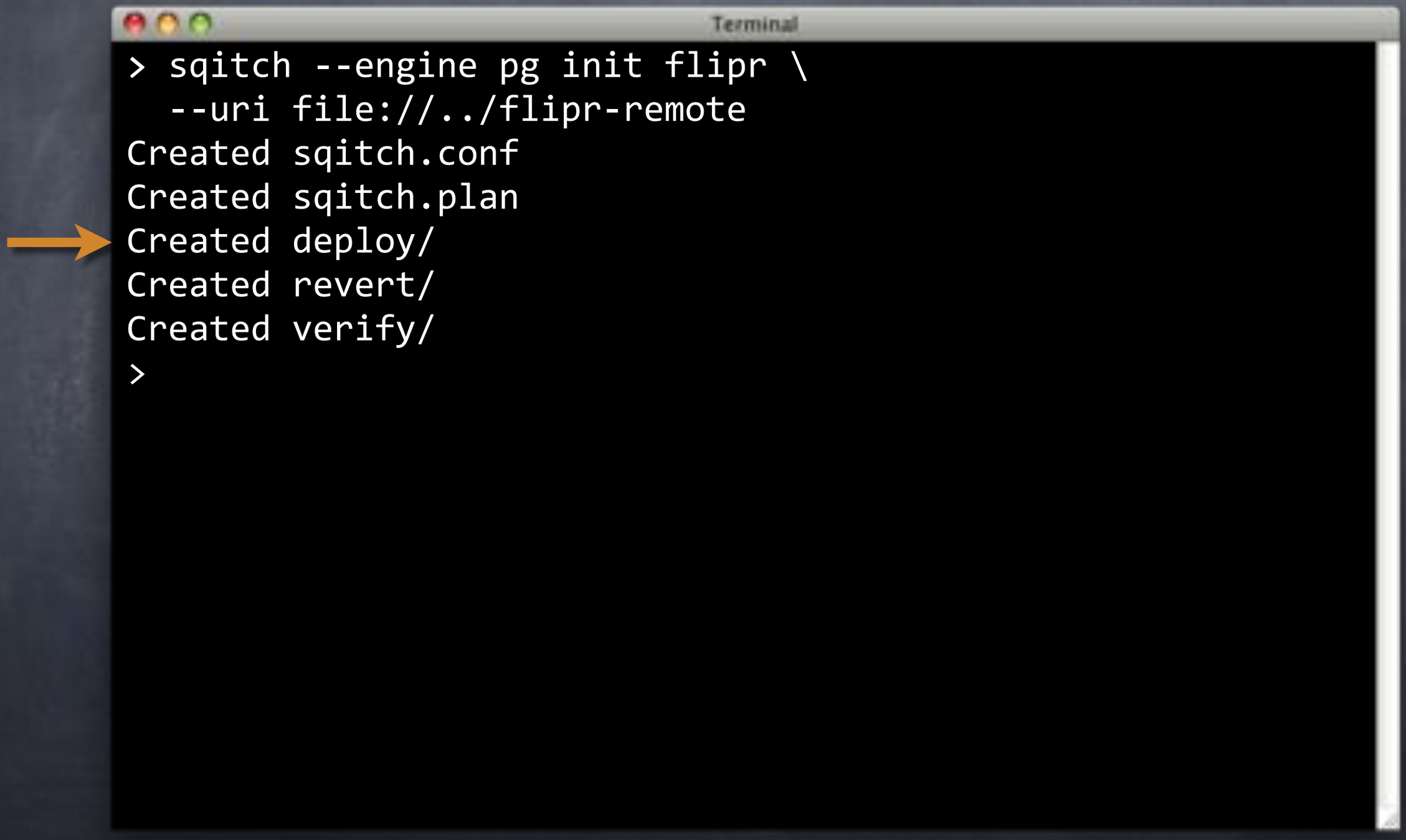
Scratch that Sqitch



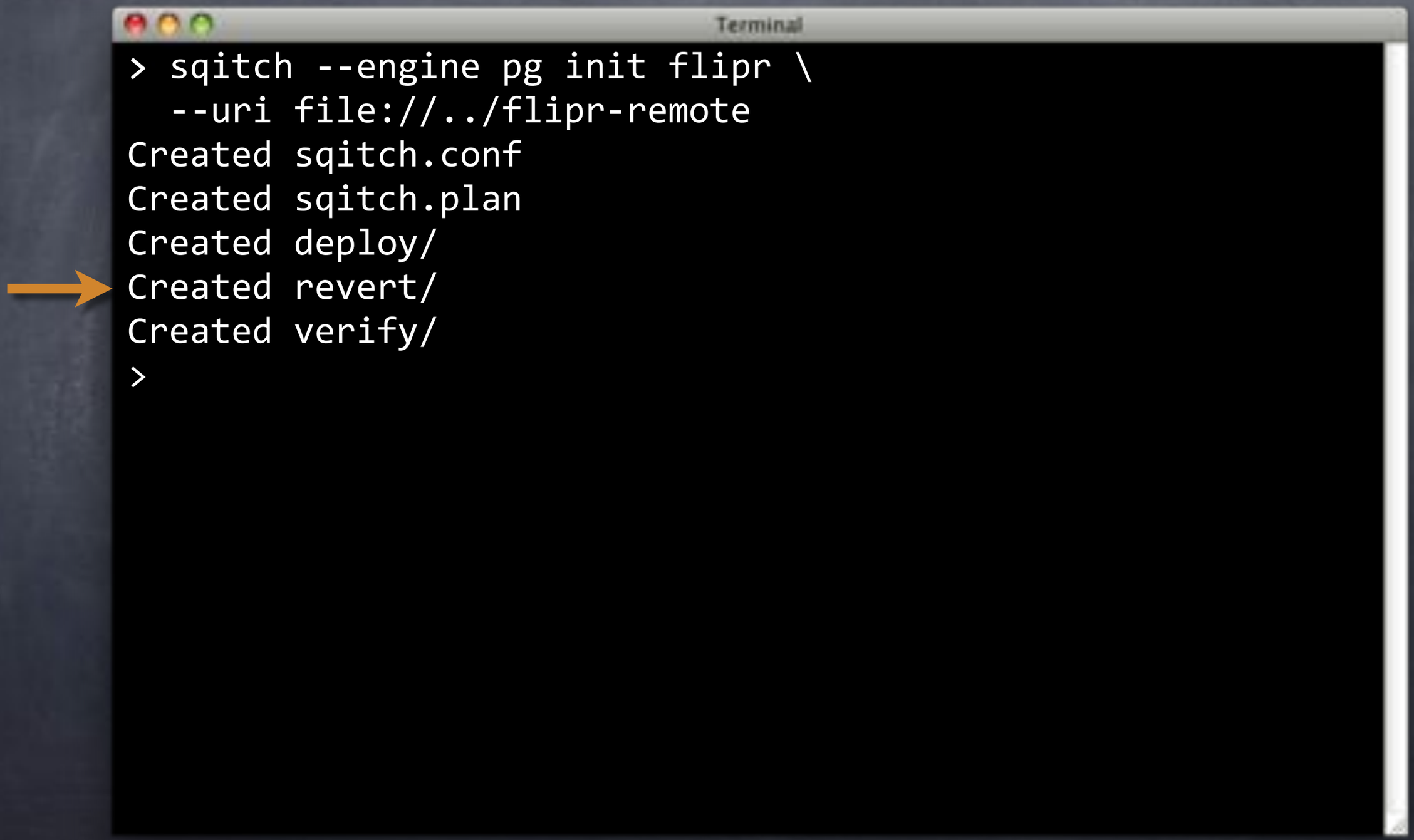
Scratch that Sqitch



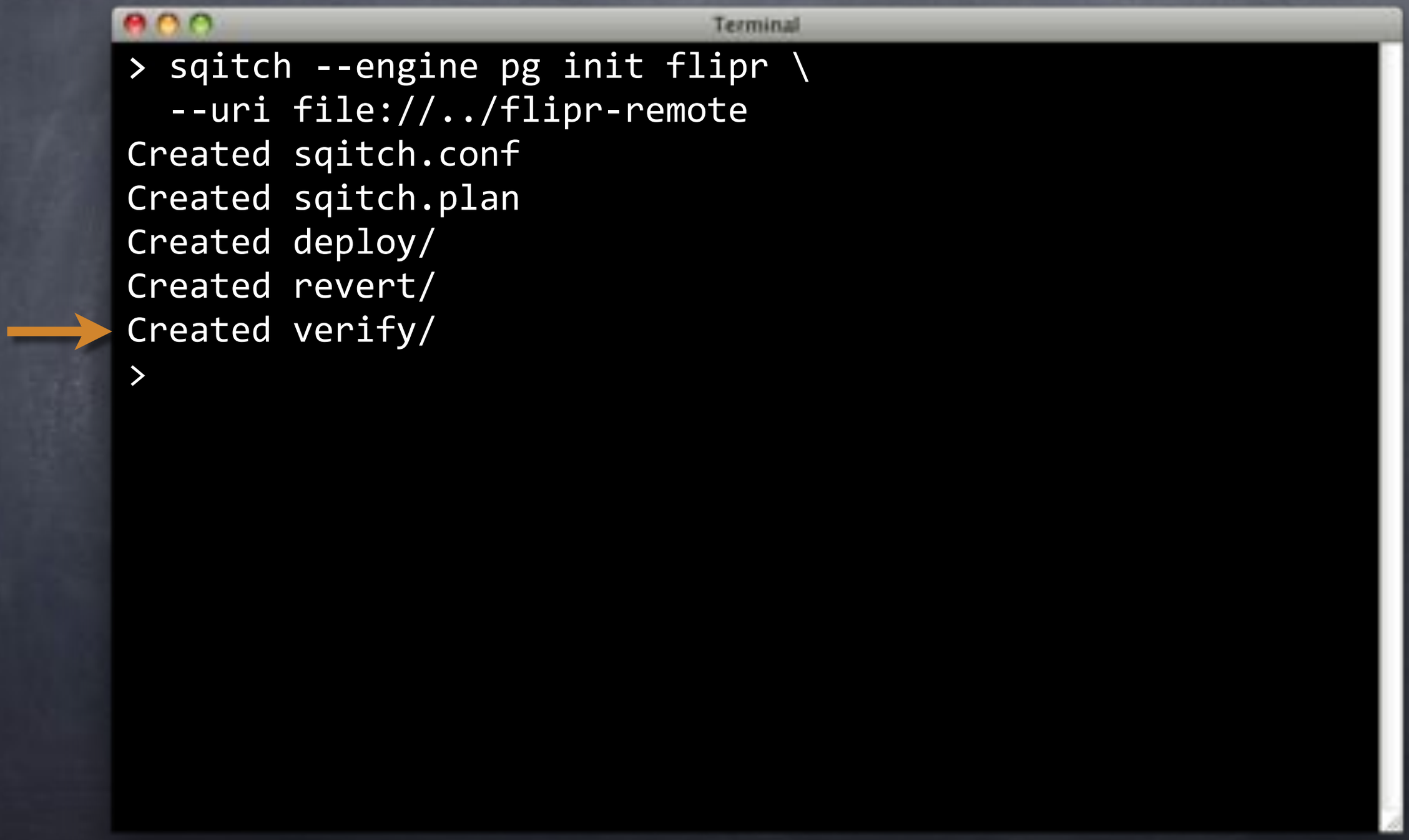
Scratch that Sqitch



Scratch that Sqitch



Scratch that Sqitch



Scratch that Sqitch

Terminal

```
> sqitch --engine pg init flipr \
  --uri file://../flipr-remote
Created sqitch.conf
Created sqitch.plan
Created deploy/
Created revert/
Created verify/
> emacs sqitch.conf
>
```

sqitch.conf

Emacs

```
[core]
    engine = pg
    # plan_file = sqitch.plan
    # top_dir = .
    # deploy_dir = deploy
    # revert_dir = revert
    # verify_dir = verify
    # extension = sql
# [core "pg"]
    # client = /usr/local/pgsql/bin/psql
    # username =
    # password =
    # db_name =
    # host =
    # port =
    # sqitch_schema = sqitch
```

-:** sqitch.conf All (SQL[ansi])

sqitch.conf

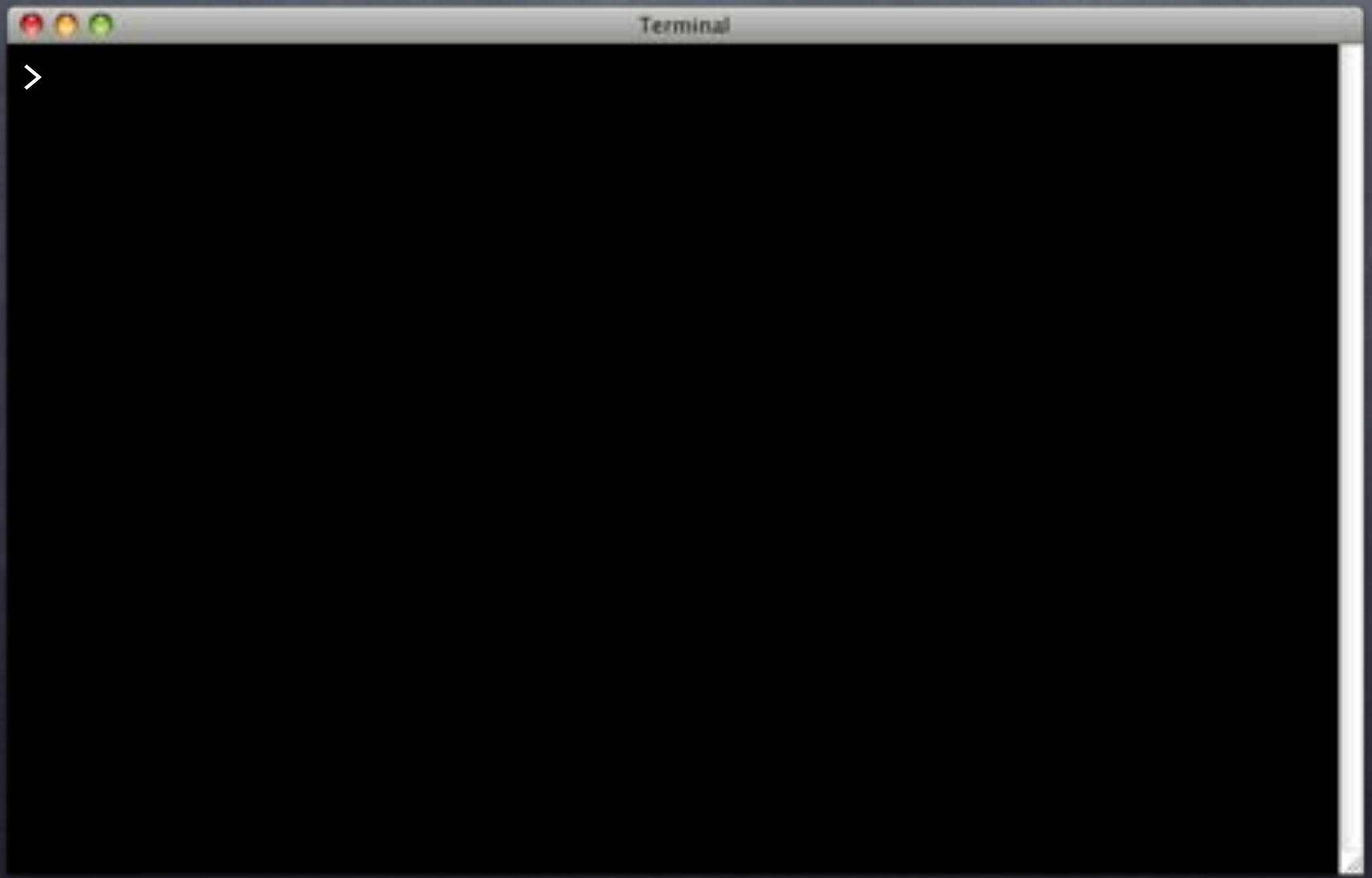
Emacs

```
[core]
    engine = pg
    # plan_file =
    # top_dir = .
    # deploy_dir = deploy
    # revert_dir = revert
    # verify_dir = verify
    # extension = sql
# [core "pg"]
    # client = /usr/local/pgsql/bin/psql
    # username =
    # password =
    # db_name =
    # host =
    # port =
    # sqitch_schema = sqitch
```

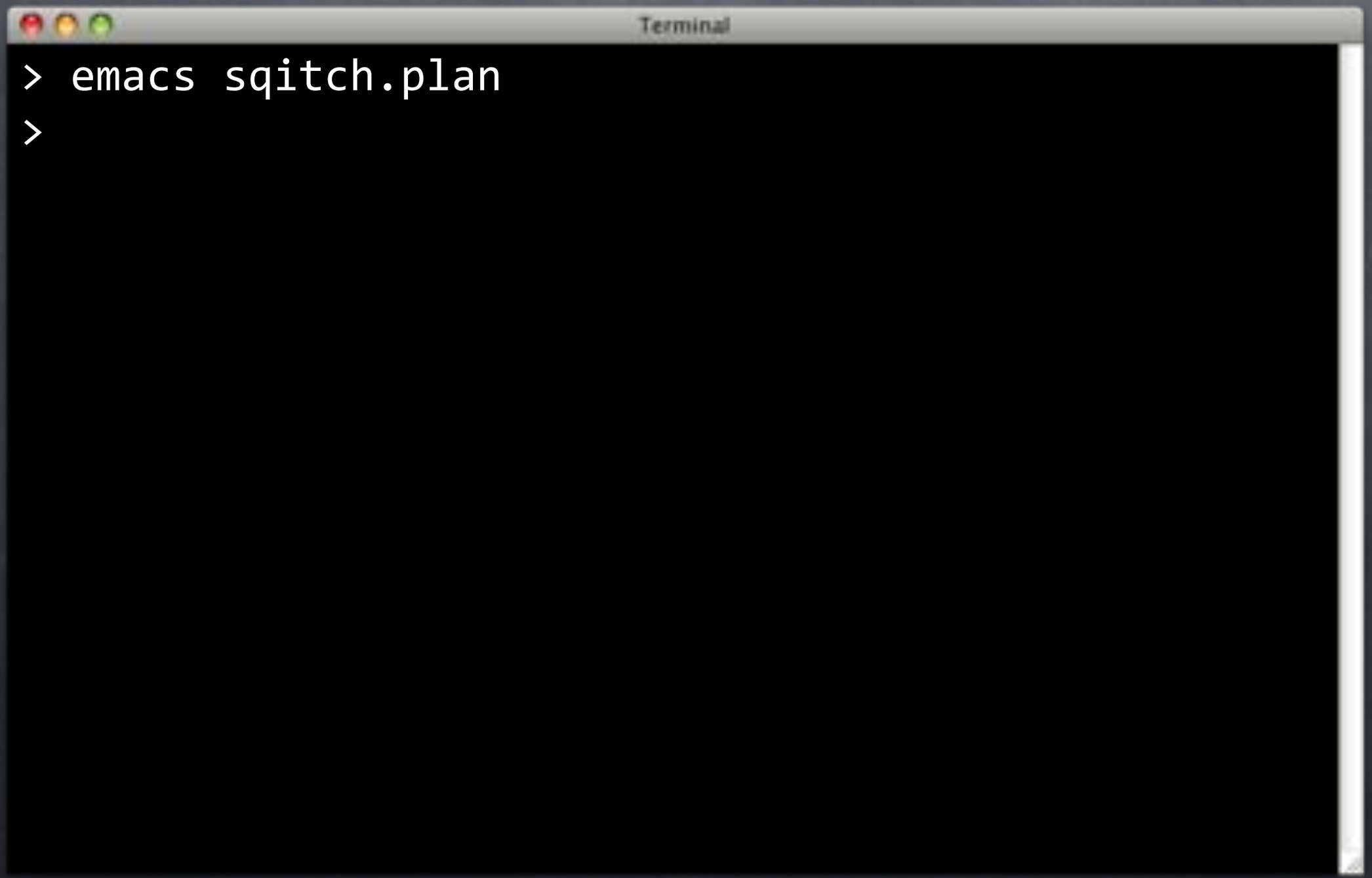
--engine pg

-:**- sqitch.conf All (SQL[ansi])

What's the Plan Man?



What's the Plan Man?



sqitch.plan

Emacs

```
%syntax-version=1.0.0-b1
%project=flipr
%uri=file://../flipr-remote
```

-:** sqitch.plan All (SQL[ansi])----------

sqitch.plan

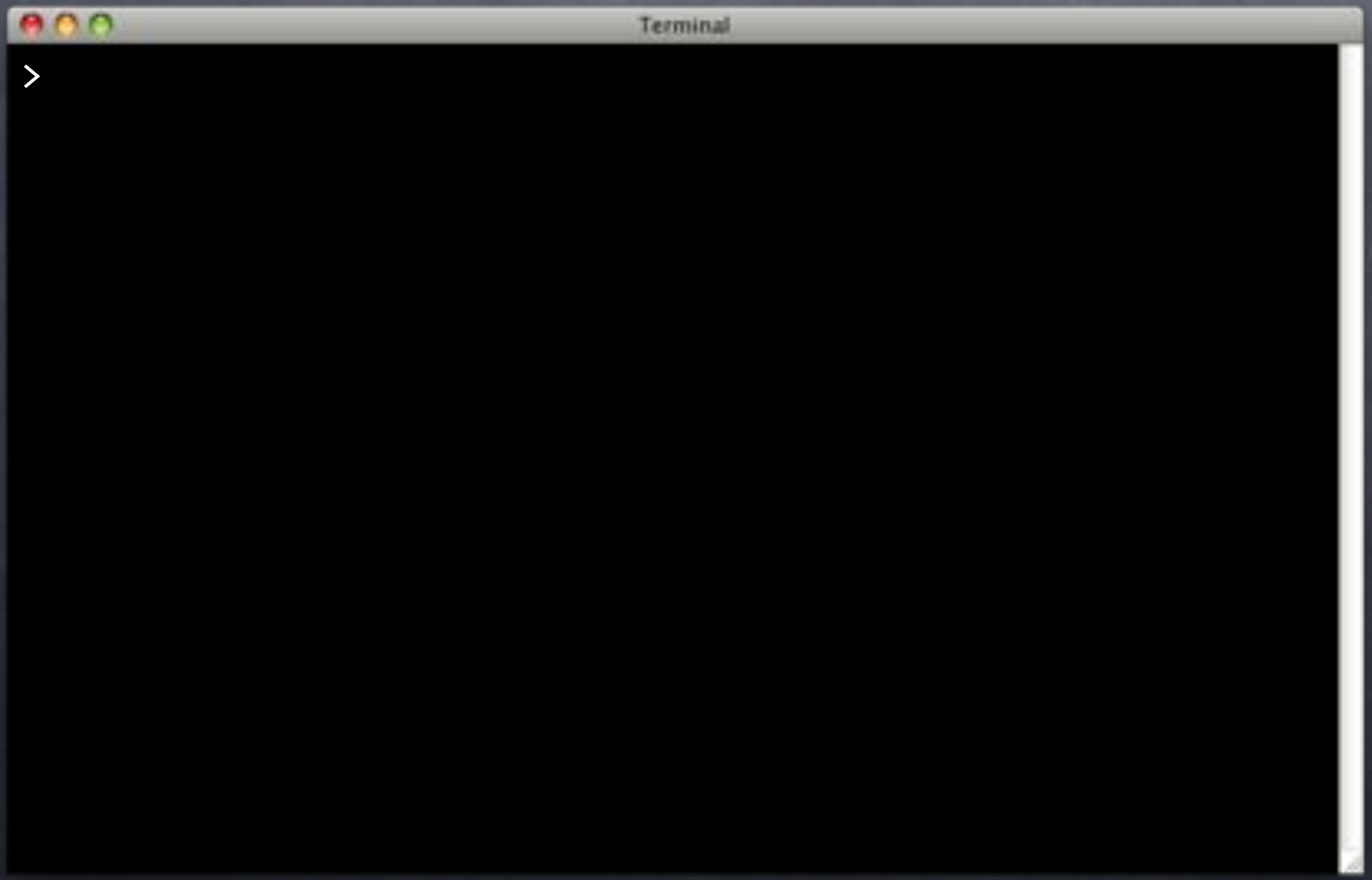
Emacs

```
%syntax-version=1.0.0-b1
%project=flipr
%uri=file://../flipr-remote
```

Identified

-:**- sqitch.plan   All (SQL[ansi])----------

Make It So



Make It So

Terminal

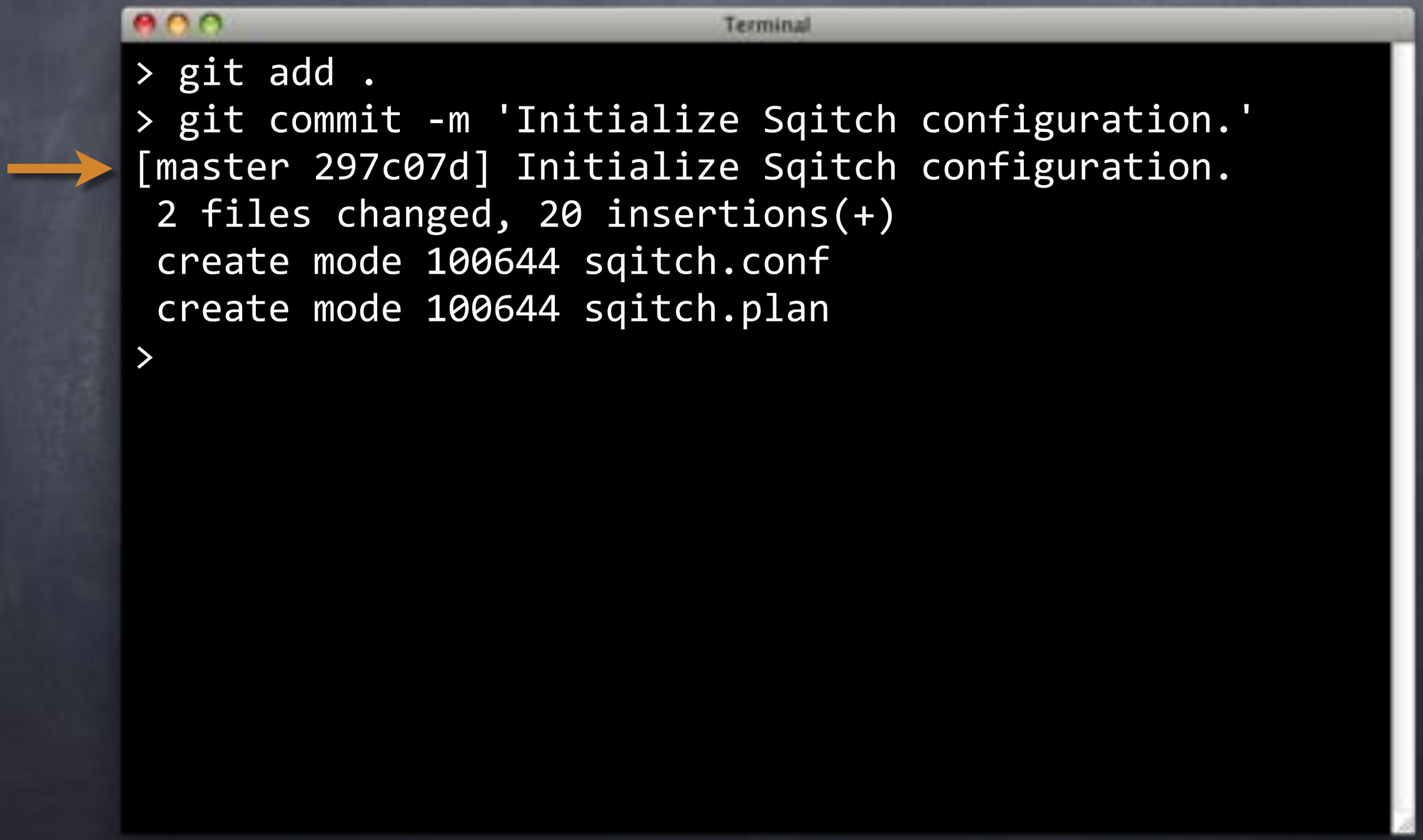
```
> git add .
> git commit -m 'Initialize Sqitch configuration.'
[master 297c07d] Initialize Sqitch configuration.
 2 files changed, 20 insertions(+)
 create mode 100644 sqitch.conf
 create mode 100644 sqitch.plan
>
```

Make It So

Terminal

```
> git add .
> git commit -m 'Initialize Sqitch configuration.'
[master 297c07d] Initialize Sqitch configuration.
 2 files changed, 20 insertions(+)
 create mode 100644 sqitch.conf
 create mode 100644 sqitch.plan
>
```

Make It So


Terminal
> git add .
> git commit -m 'Initialize Sqitch configuration.'
[master 297c07d] Initialize Sqitch configuration.
 2 files changed, 20 insertions(+)
 create mode 100644 sqitch.conf
 create mode 100644 sqitch.plan
>

Make It So

Terminal

```
> git add .
> git commit -m 'Initialize Sqitch configuration.'
[master 297c07d] Initialize Sqitch configuration.
 2 files changed, 20 insertions(+)
 create mode 100644 sqitch.conf
 create mode 100644 sqitch.plan
> git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 564 bytes, done.
Total 4 (delta 0), reused 0 (delta 0)
To ../flipr-remote
   7d28be0..297c07d  master -> master
>
```

Make It So

Terminal

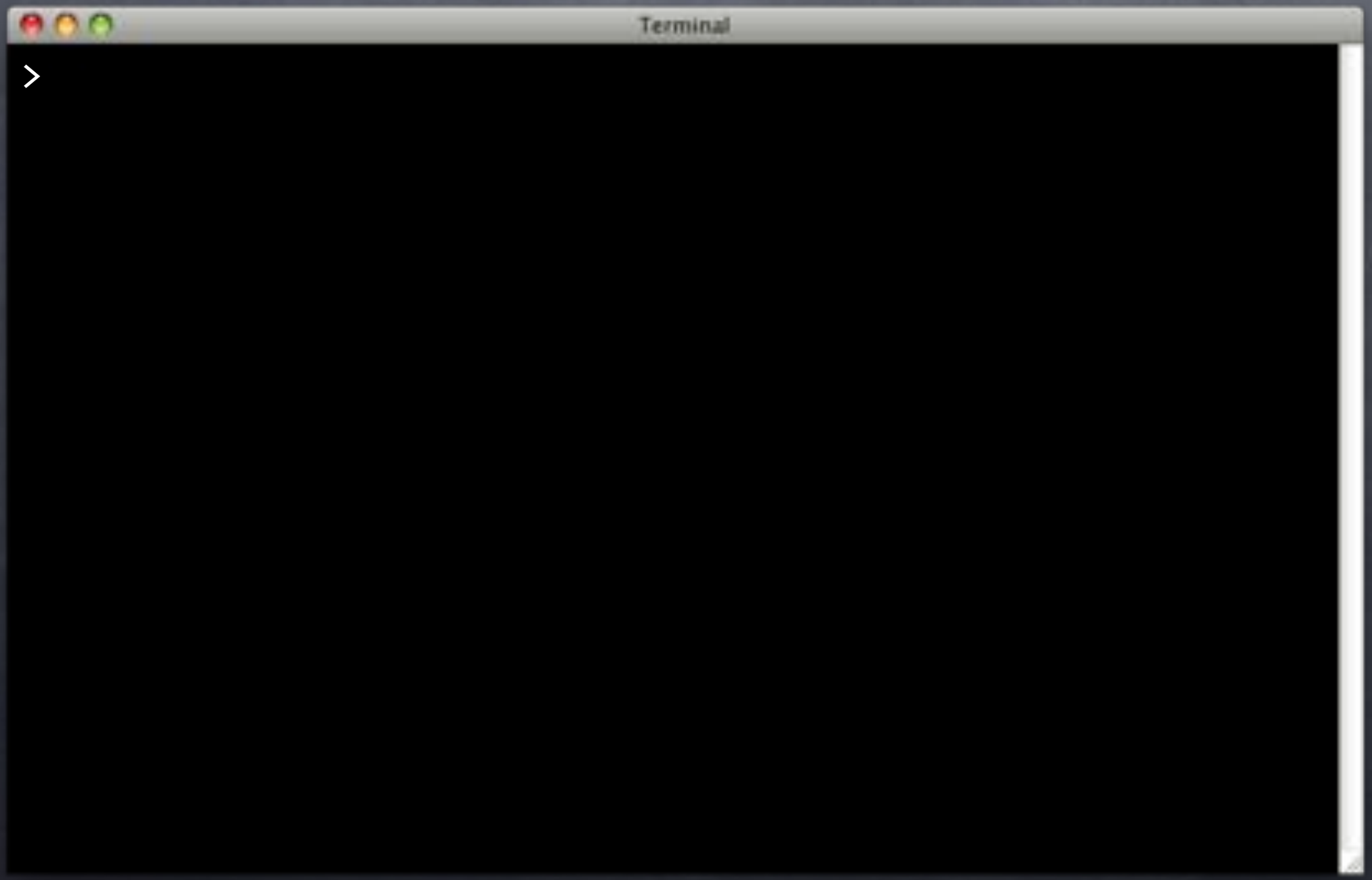
```
> git add .
> git commit -m 'Initialize Sqitch configuration.'
[master 297c07d] Initialize Sqitch configuration.
 2 files changed, 20 insertions(+)
 create mode 100644 sqitch.conf
 create mode 100644 sqitch.plan
> git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 564 bytes, done.
Total 4 (delta 0), reused 0 (delta 0)
To ../flipr-remote
   7d28be0..297c07d  master -> master
>
```

Make It So

Terminal

```
> git add .
> git commit -m 'Initialize Sqitch configuration.'
[master 297c07d] Initialize Sqitch configuration.
 2 files changed, 20 insertions(+)
 create mode 100644 sqitch.conf
 create mode 100644 sqitch.plan
> git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 564 bytes, done.
Total 4 (delta 0), reused 0 (delta 0)
To ../flipr-remote
   7d28be0..297c07d  master -> master
>
```

Where've We Been?



Where've We Been?

Terminal

```
> git log
commit a4f765f88c1875ffe9427e73f72a6b66ce363ed4
Author: David E. Wheeler <david@justatheory.com>
Date:   Tue May 21 15:00:33 2013 -0400

    Initialize Sqitch configuration.

commit 2c4b51017929634a30f4a74e20268c05effdfcb3
Author: David E. Wheeler <david@justatheory.com>
Date:   Tue May 21 14:59:16 2013 -0400

    Initialize repo, add README.
>
```

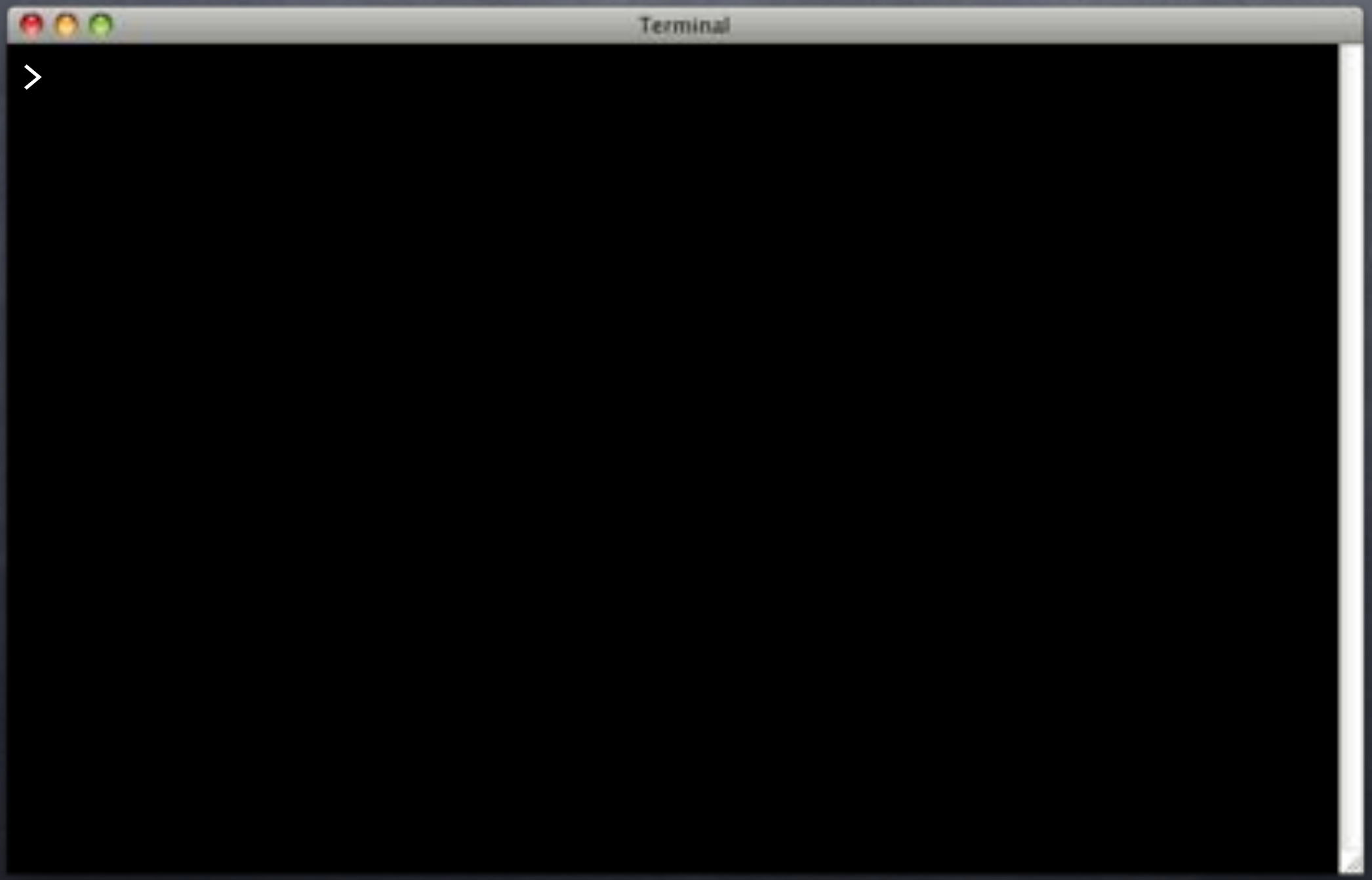
Where've We Been?

Where've We Been?

Terminal

```
> git log
commit a4f765f88c1875ffe9427e73f72a6b66ce363ed4
Author: David E. Wheeler <david@justatheory.com>
Date:   Tue May 21 15:00:33 2013 -0400

    Initialize Sqitch configuration.

commit 2c4b51017929634a30f4a74e20268c05effdfcb3
Author: David E. Wheeler <david@justatheory.com>
Date:   Tue May 21 14:59:16 2013 -0400

    Initialize repo, add README.
>
```

First Change



First Change

Terminal

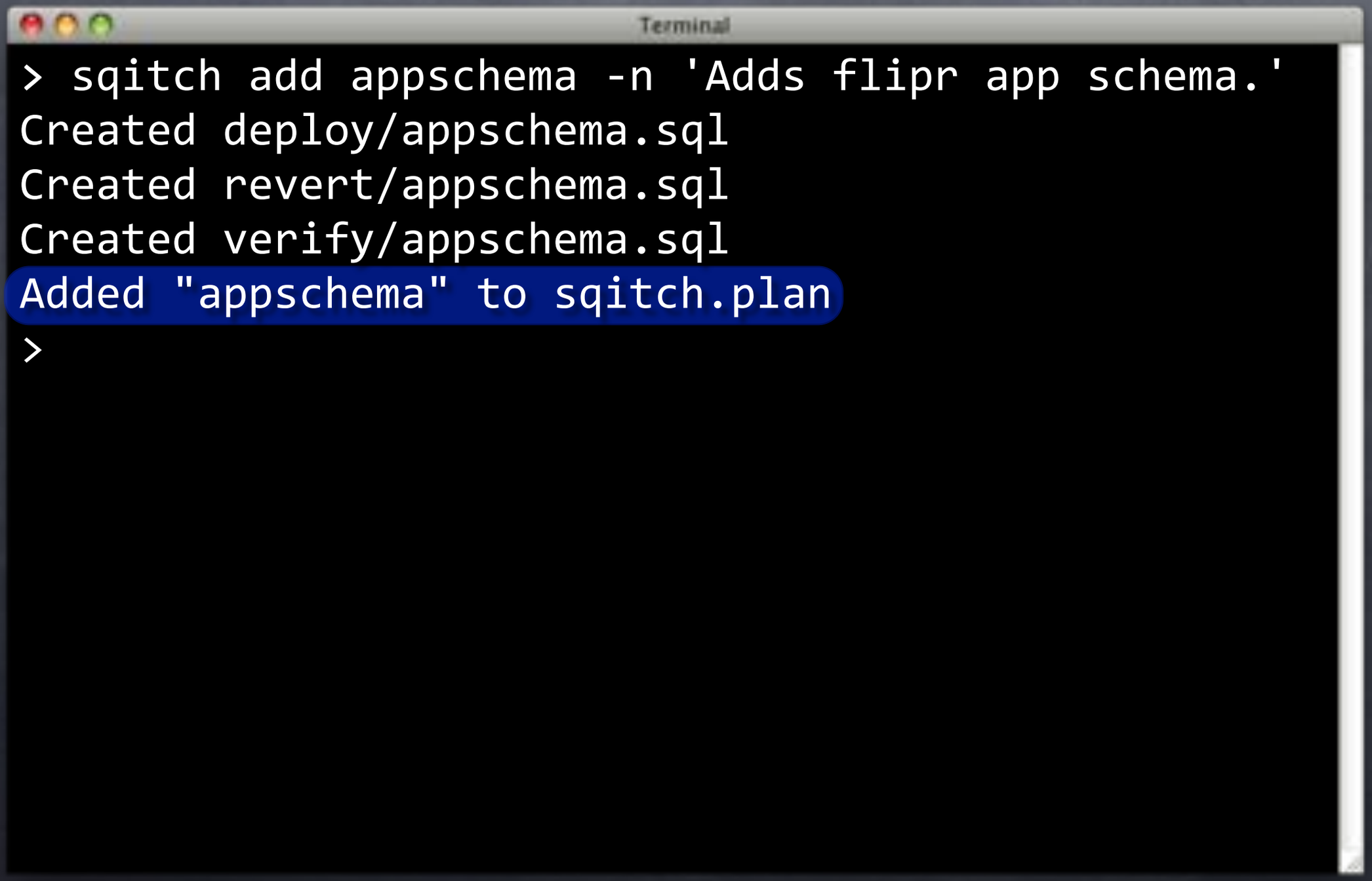
```
> sqitch add appschema -n 'Adds flipr app schema.'
Created deploy/appschema.sql
Created revert/appschema.sql
Created verify/appschema.sql
Added "appschema" to sqitch.plan
>
```

First Change

Terminal

```
> sqitch add appschema -n 'Adds flipr app schema.'
Created deploy/appschema.sql
Created revert/appschema.sql
Created verify/appschema.sql
Added "appschema" to sqitch.plan
>
```

First Change



First Change

Terminal

```
> sqitch add appschema -n 'Adds flipr app schema.'
Created deploy/appschema.sql
Created revert/appschema.sql
Created verify/appschema.sql
Added "appschema" to sqitch.plan
> emacs deploy/appschema.sql
>
```

deploy/appschema.sql

Emacs

```
-- Deploy appschema

BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

-:** deploy/appschem All (SQL[ansi])

deploy/appschema.sql

Emacs

```
-- Deploy appschema

BEGIN;

CREATE SCHEMA flipr;

COMMIT;
```

-:** deploy/appschem All (SQL[ansi])----------

First Change

Terminal

```
> sqitch add appschema -n 'Adds flipr app schema.'
Created deploy/appschema.sql
Created revert/appschema.sql
Created verify/appschema.sql
Added "appschema" to sqitch.plan
> emacs deploy/appschema.sql
>
```

First Change

Terminal

```
> sqitch add appschema -n 'Adds flipr app schema.'
Created deploy/appschema.sql
Created revert/appschema.sql
Created verify/appschema.sql
Added "appschema" to sqitch.plan
> emacs deploy/appschema.sql
> emacs revert/appschema.sql
>
```

revert/appschema.sql

Emacs

```
-- Revert appschema

BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

-:** revert/appschem All (SQL[ansi])----------

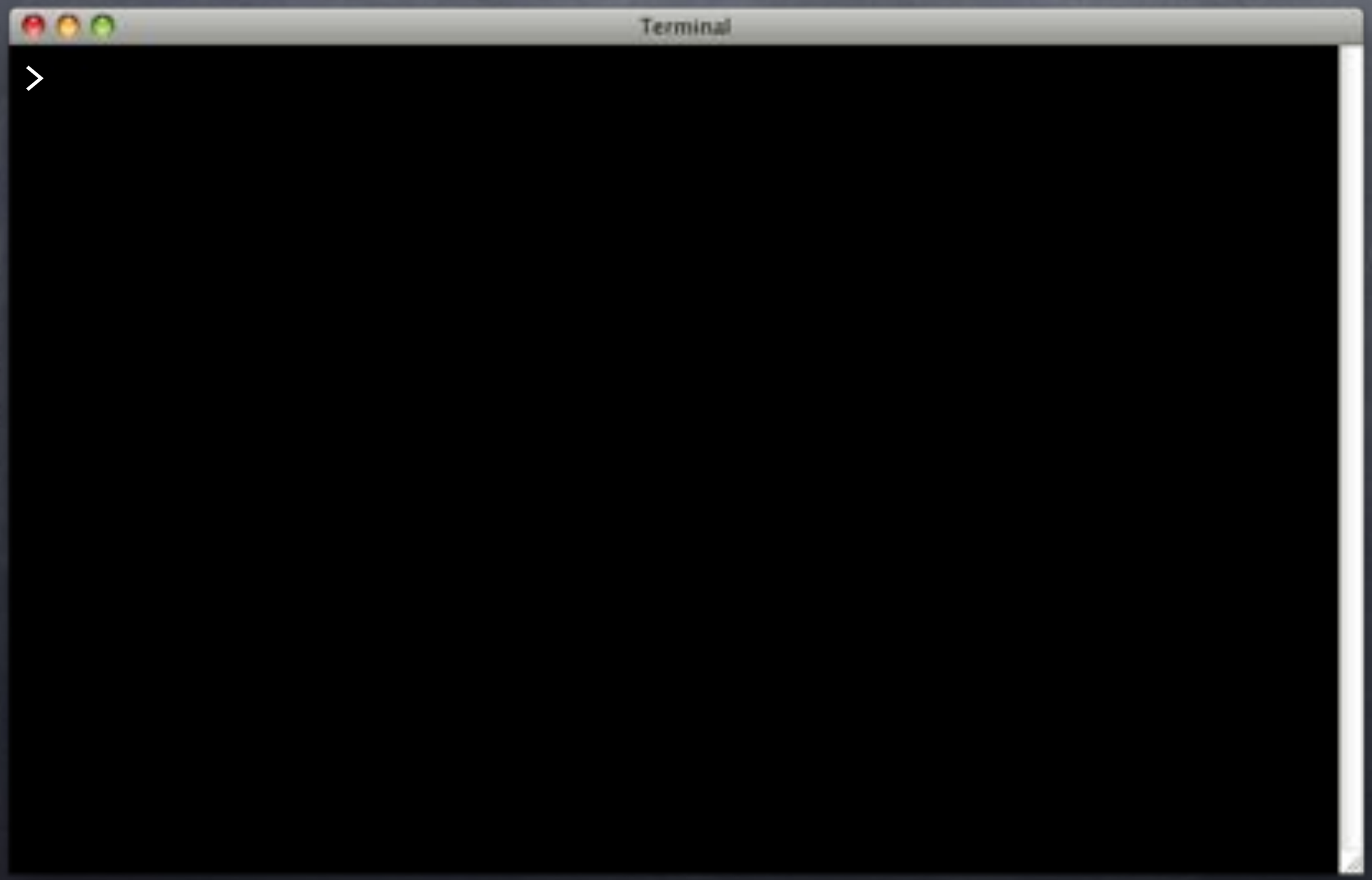
revert/appschema.sql

Emacs

```
-- Revert appschema

BEGIN;

DROP SCHEMA flipr;

COMMIT;
```

-:** revert/appschem All (SQL[ansi])

Make it So!

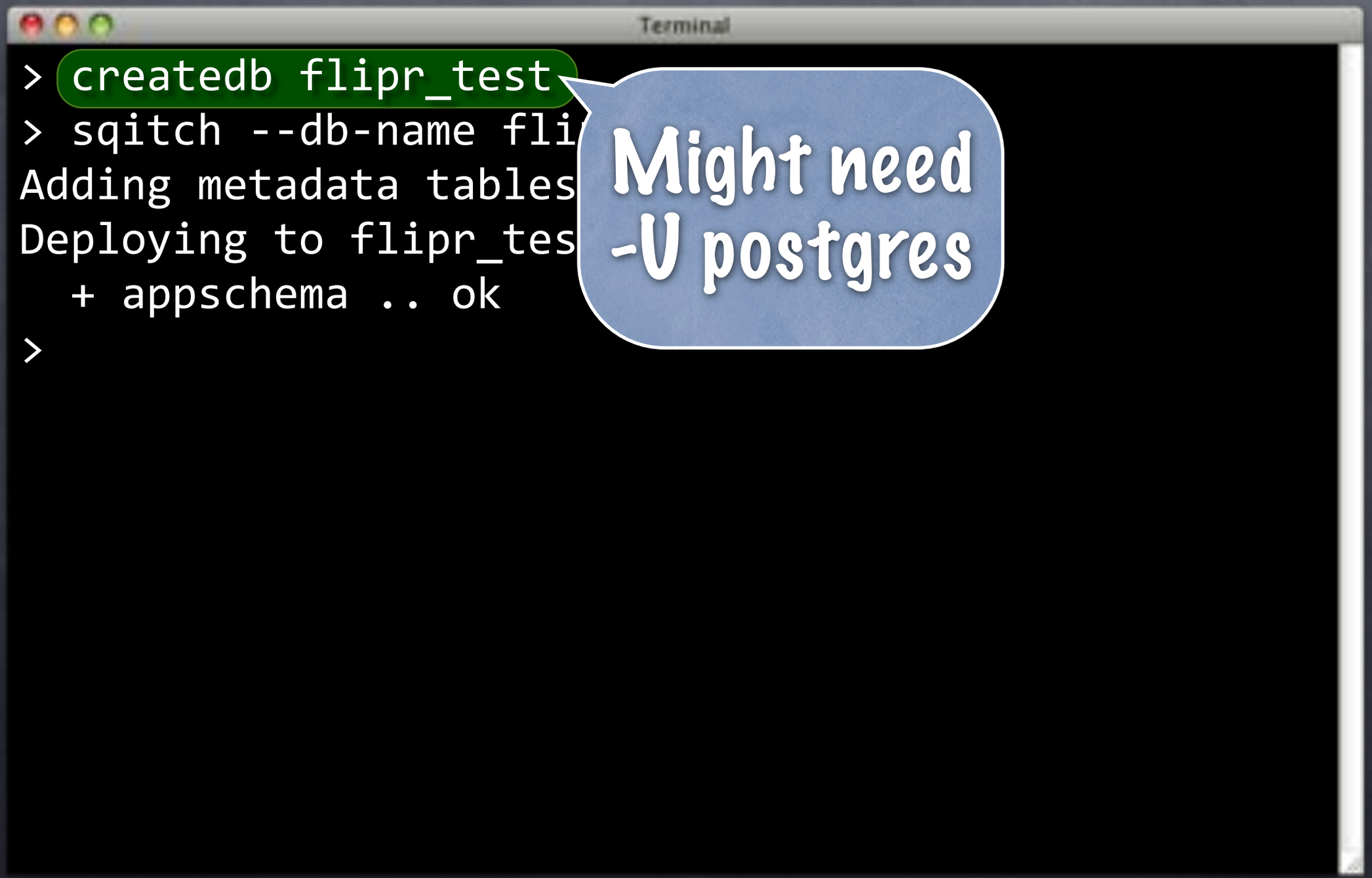


Make it So!

Terminal

```
> createdb flipr_test
> sqitch --db-name flipr_test deploy
Adding metadata tables to flipr_test
Deploying to flipr_test
  + appschema .. ok
>
```

Make it So!

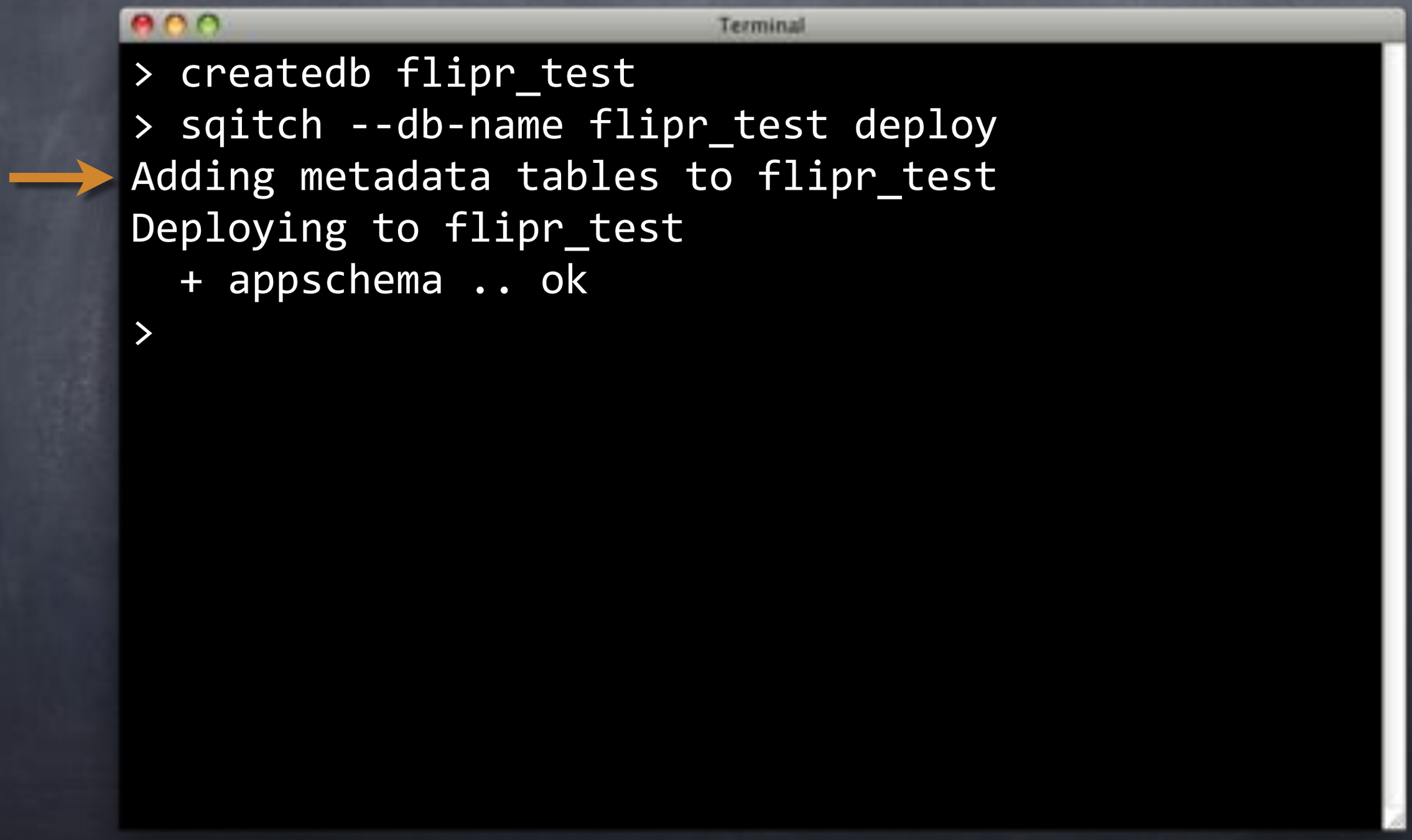

Terminal
> createdb flipr_test
> sqitch --db-name fli
Adding metadata tables
Deploying to flipr_tes
+ appschema .. ok
>
Might need
-U postgres

Make it So!

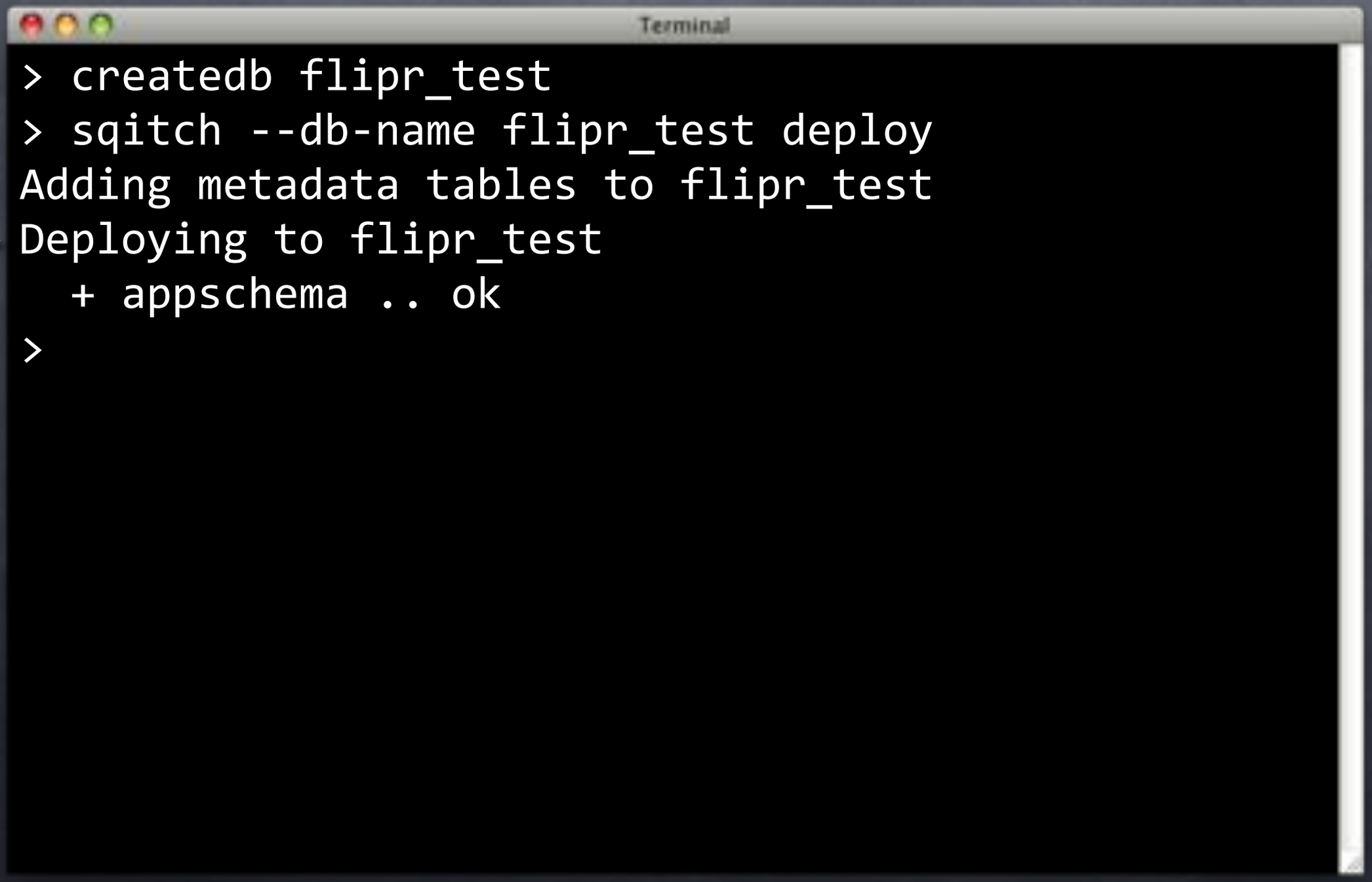
Terminal

```
> createdb flipr_test
> sqitch --db-name flipr_test deploy
Adding metadata tables to flipr_test
Deploying to flipr_test
  + appschema .. ok
>
```

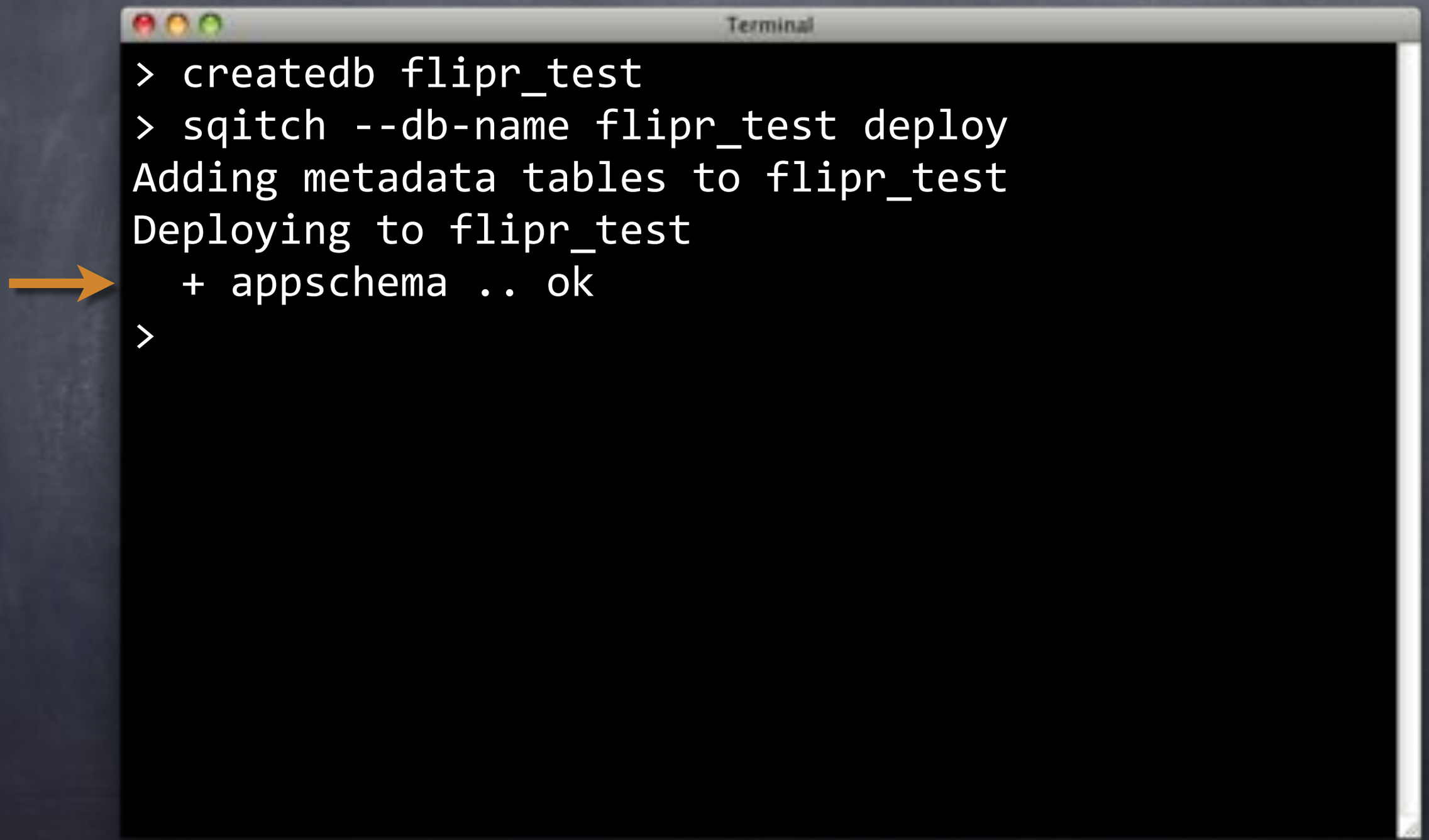
Make it So!


Terminal
> createdb flipr_test
> sqitch --db-name flipr_test deploy
Adding metadata tables to flipr_test
Deploying to flipr_test
  + appschema .. ok
>

Make it So!


Terminal
> createdb flipr_test
> sqitch --db-name flipr_test deploy
Adding metadata tables to flipr_test
Deploying to flipr_test
  + appschema .. ok
>

Make it So!


Terminal
> createdb flipr_test
> sqitch --db-name flipr_test deploy
Adding metadata tables to flipr_test
Deploying to flipr_test
  + appschema .. ok
>

Make it So!

Terminal

```
> createdb flipr_test
> sqitch --db-name flipr_test deploy
Adding metadata tables to flipr_test
Deploying to flipr_test
  + appschema .. ok
> psql -d flipr_test -c '\dn flipr'
List of schemas
 Name  | Owner
-------+-------
 flipr | david
>
```

Make it So!

Terminal

```
> createdb flipr_test
> sqitch --db-name flipr_test deploy
Adding metadata tables to flipr_test
Deploying to flipr_test
  + appschema .. ok
> psql -d flipr_test -c '\dn flipr'
List of schemas
 Name  | Owner
-------+-------
 flipr | david
>
```

Trust, but verify

Make it So!

Terminal

```
> createdb flipr_test
> sqitch --db-name flipr_test deploy
Adding metadata tables to flipr_test
Deploying to flipr_test
  + appschema .. ok
> psql -d flipr_test -c '\dn flipr'
List of schemas
 Name  | Owner
-------+-------
 flipr | david
> emacs verify/appschema.sql
```

verify/appschema.sql

Emacs

```
-- Verify appschema

BEGIN;

-- XXX Add verifications here.

ROLLBACK;
```

-:** verify/appschem All (SQL[ansi])

verify/appschema.sql

Emacs

```
-- Verify appschema

BEGIN;

SELECT pg_catalog.has_schema_privilege('nada', 'usage');

ROLLBACK;
```

-:** verify/appschem All (SQL[ansi])

verify/appschema.sql

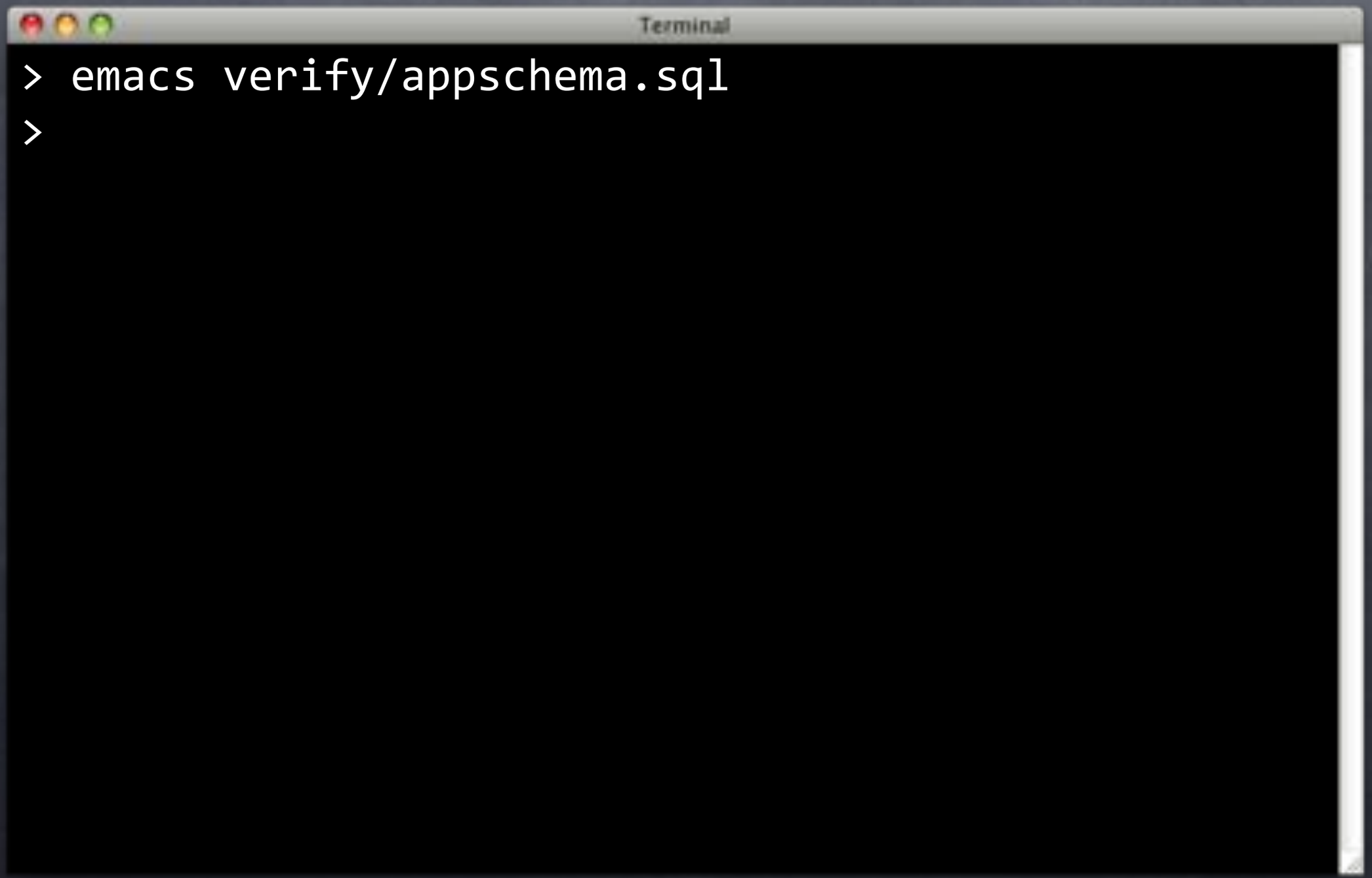
Emacs

```
-- Verify appschema

BEGIN;

SELECT pg_catalog.has_schema_privilege('nada', 'usage');

ROLLBACK;
```

Let's try it, first

-:**- verify/appschem All (SQL[ansi])----------

Trust, But Verify



Trust, But Verify

Terminal

```
> emacs verify/appschema.sql
> sqitch --db-name flipr_test verify
Verifying flipr_test
  * appschema .. psql:verify/appschema.sql:5:
ERROR:  schema "nada" does not exist
# Verify script "verify/appschema.sql" failed.
not ok

Verify Summary Report
---------------------
Changes: 1
Errors:  1
Verify failed
>
```

Trust, But Verify

Terminal

```
> emacs verify/appschema.sql
> sqitch --db-name flipr_test verify
Verifying flipr_test
  * appschema .. psql:verify/appschema.sql:5:
ERROR:  schema "nada" does not exist
# Verify script "verify/appschema.sql" failed.
not ok

Verify Summary Report
---------------------
Changes: 1
Errors:  1
Verify failed
>
```

Trust, But Verify

Terminal

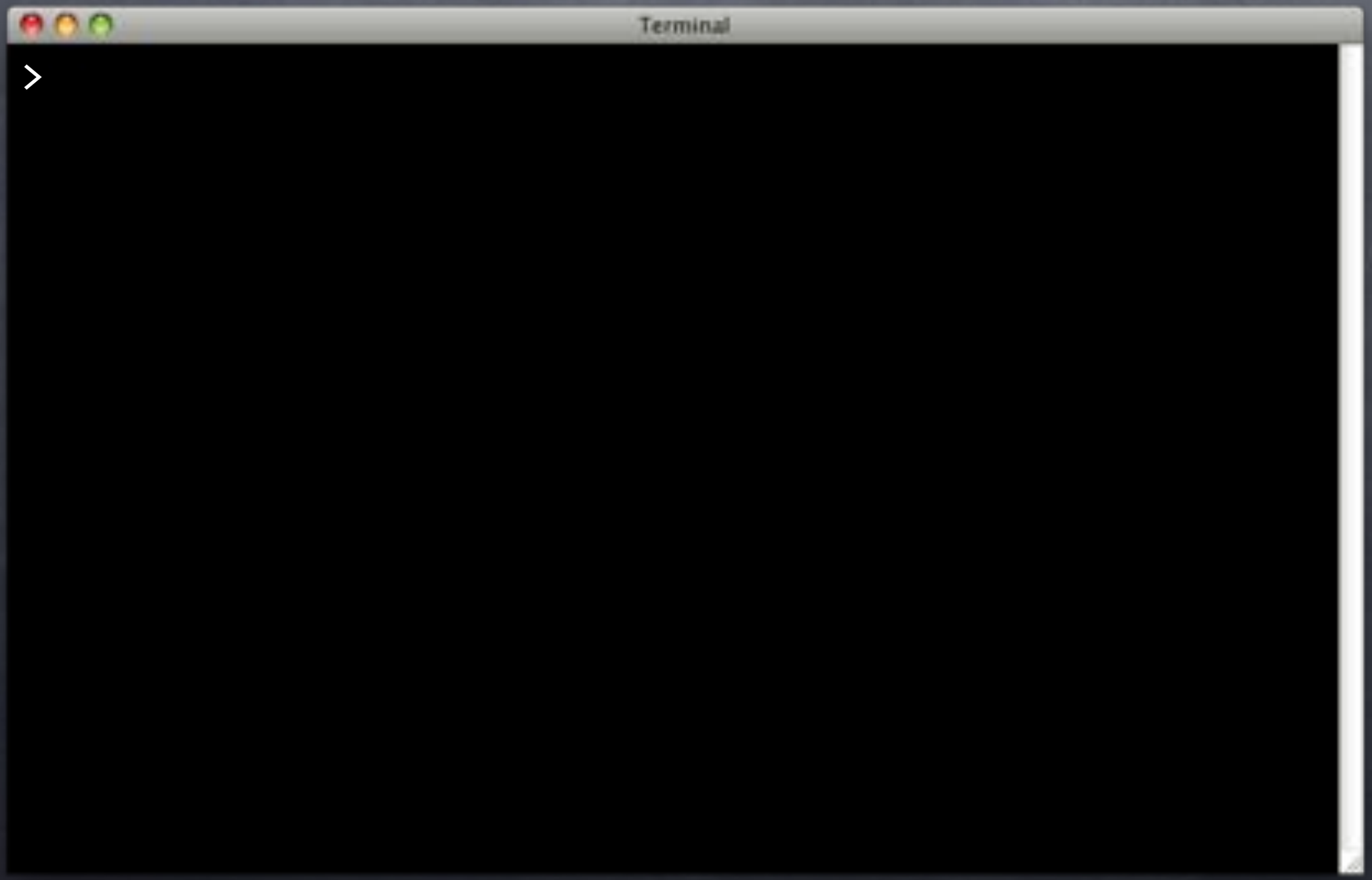
```
> emacs verify/appschema.sql
> sqitch --db-name flipr_test verify
Verifying flipr_test
  * appschema .. psql:verify/appschema.sql:5:
ERROR:  schema "nada" does not exist
# Verify script "verify/appschema.sql" failed.
not ok

Verify Summary Report
---------------------
Changes: 1
Errors:  1
Verify failed
>
```

Trust, But Verify

Terminal

```
> emacs verify/appschema.sql
> sqitch --db-name flipr_test verify
Verifying flipr_test
  * appschema .. psql:verify/appschema.sql:5:
ERROR:  schema "nada" does not exist
# Verify script "verify/appschema.sql" failed.
not ok

Verify Summary Report
---------------------
Changes: 1
Errors:  1
Verify failed
>
```

Trust, But Verify



Trust, But Verify

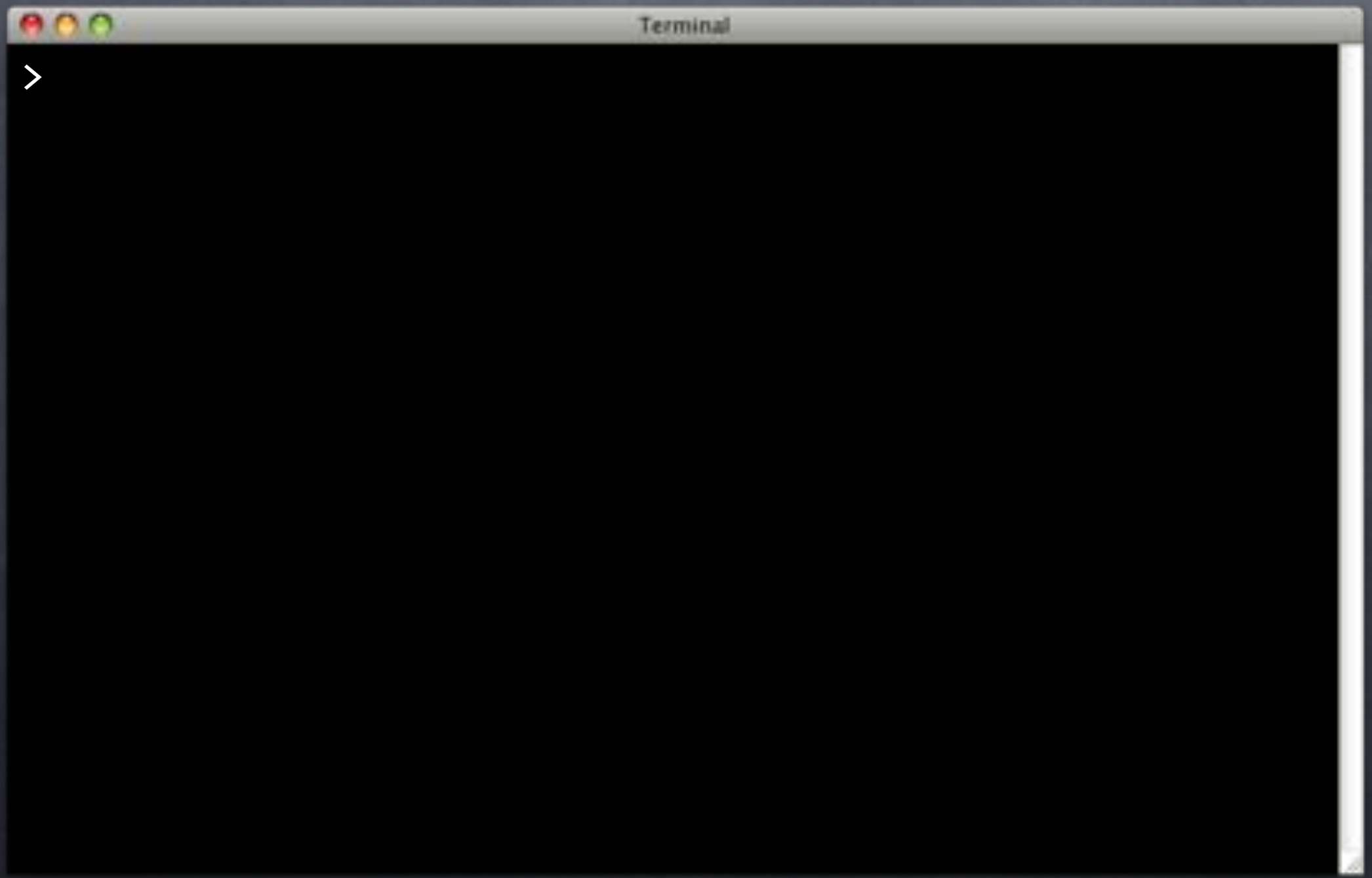
Trust, But Verify

Terminal

```
> perl -i -pe 's/nada/flipr/' verify/appschema.sql
> sqitch --db-name flipr_test verify
Verifying flipr_test
  * appschema .. ok
Verify successful
>
```

Mo betta.

How's it Look?



How's it Look?

Terminal

```
> sqitch -d flipr_test status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8b7088fd75ad8d7aecda3
# Name:     appschema
# Deployed: 2013-05-21 15:04:08 -0400
# By:       David E. Wheeler <david@justatheory.com>
#
Nothing to deploy (up-to-date)
>
```

How's it Look?

Terminal

```
> sqitch -d flipr_test status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8b7088fd75ad8d7aecda3
# Name:     appschema
# Deployed: 2013-05-21 15:04:08 -0400
# By:       David E. Wheeler <david@justatheory.com>
#
Nothing to deploy (up-to-date)
>
```

How's it Look?

Terminal

```
> sqitch -d flipr_test status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8b7088fd75ad8d7aecda3
# Name:     appschema
# Deployed: 2013-05-21 15:04:08 -0400
# By:       David E. Wheeler <david@justatheory.com>
#
Nothing to deploy (up-to-date)
>
```

How's it Look?

Terminal

```
> sqitch -d flipr_test status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8b7088fd75ad8d7aecda3
# Name:     appschema
# Deployed: 2013-05-21 15:04:08 -0400
# By:       David E. Wheeler <david@justatheory.com>
#
Nothing to deploy (up-to-date)
>
```

How's it Look?

Terminal

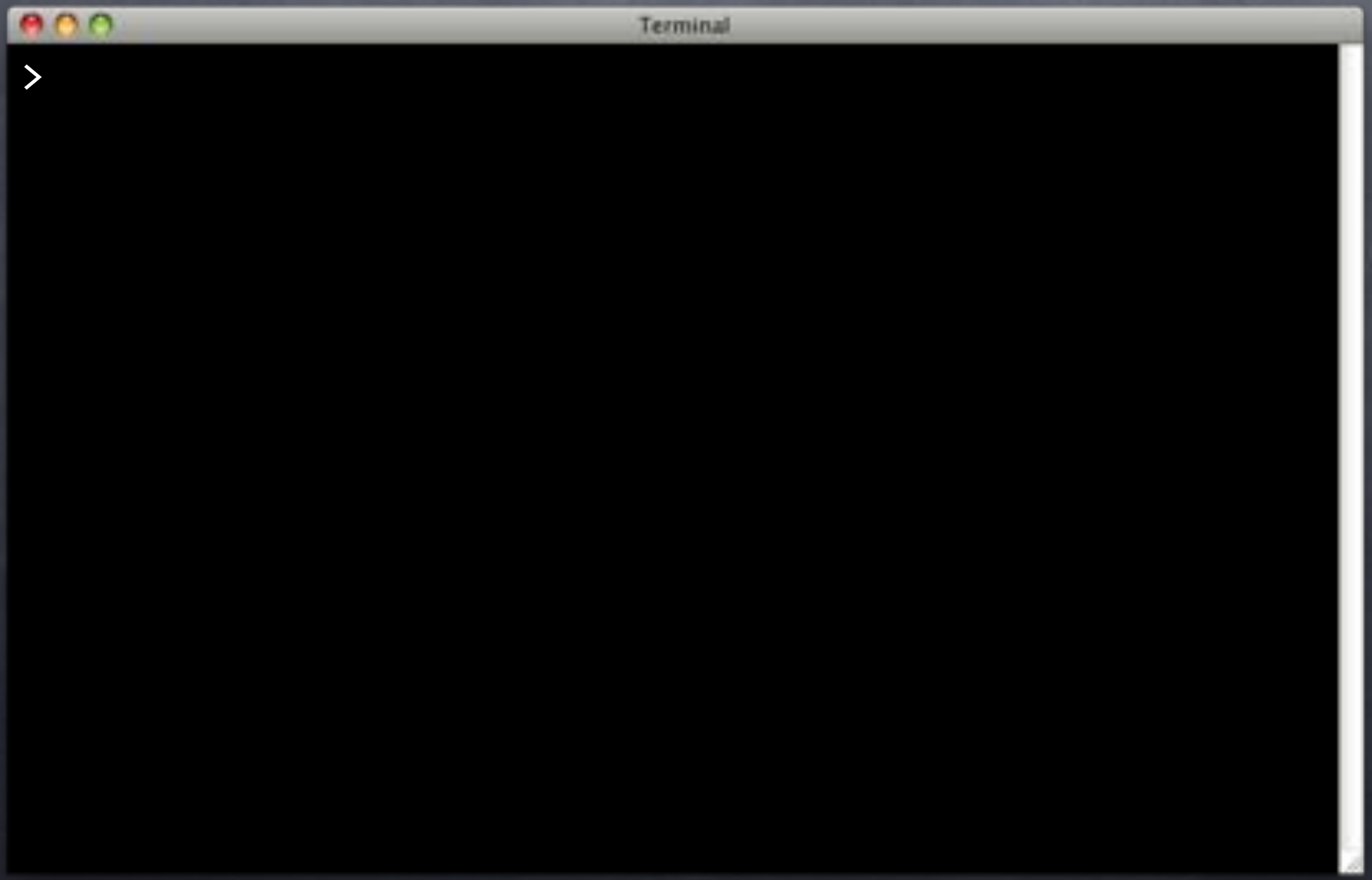
```
> sqitch -d flipr_test status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8b7088fd75ad8d7aecda3
# Name:     appschema
# Deployed: 2013-05-21 15:04:08 -0400
# By:       David E. Wheeler <david@justatheory.com>
#
Nothing to deploy (up-to-date)
>
```

How's it Look?

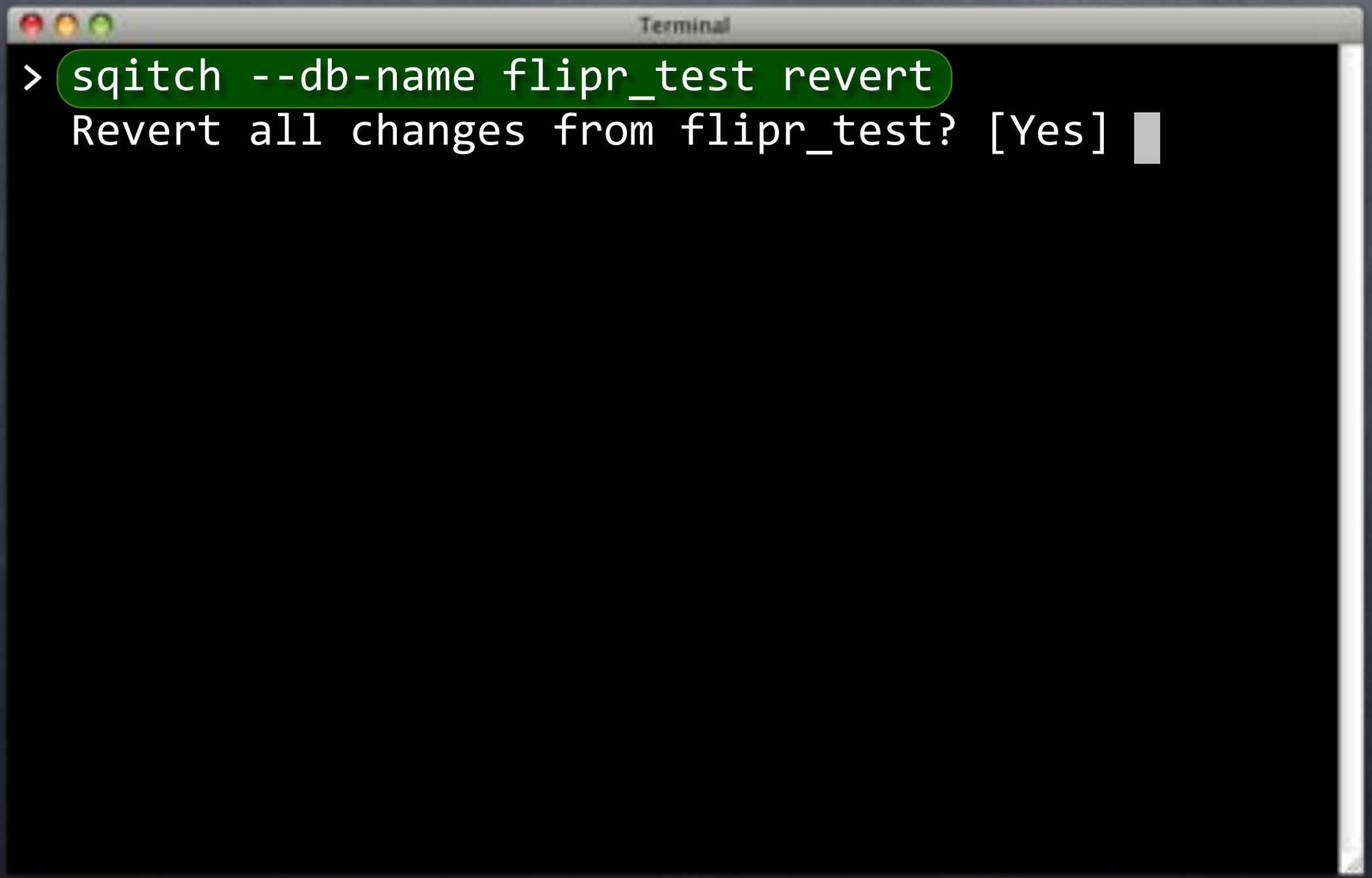
Terminal

```
> sqitch -d flipr_test status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8b7088fd75ad8d7aecda3
# Name:     appschema
# Deployed: 2013-05-21 15:04:08 -0400
# By:       David E. Wheeler <david@justatheory.com>
#
Nothing to deploy (up-to-date)
>
```

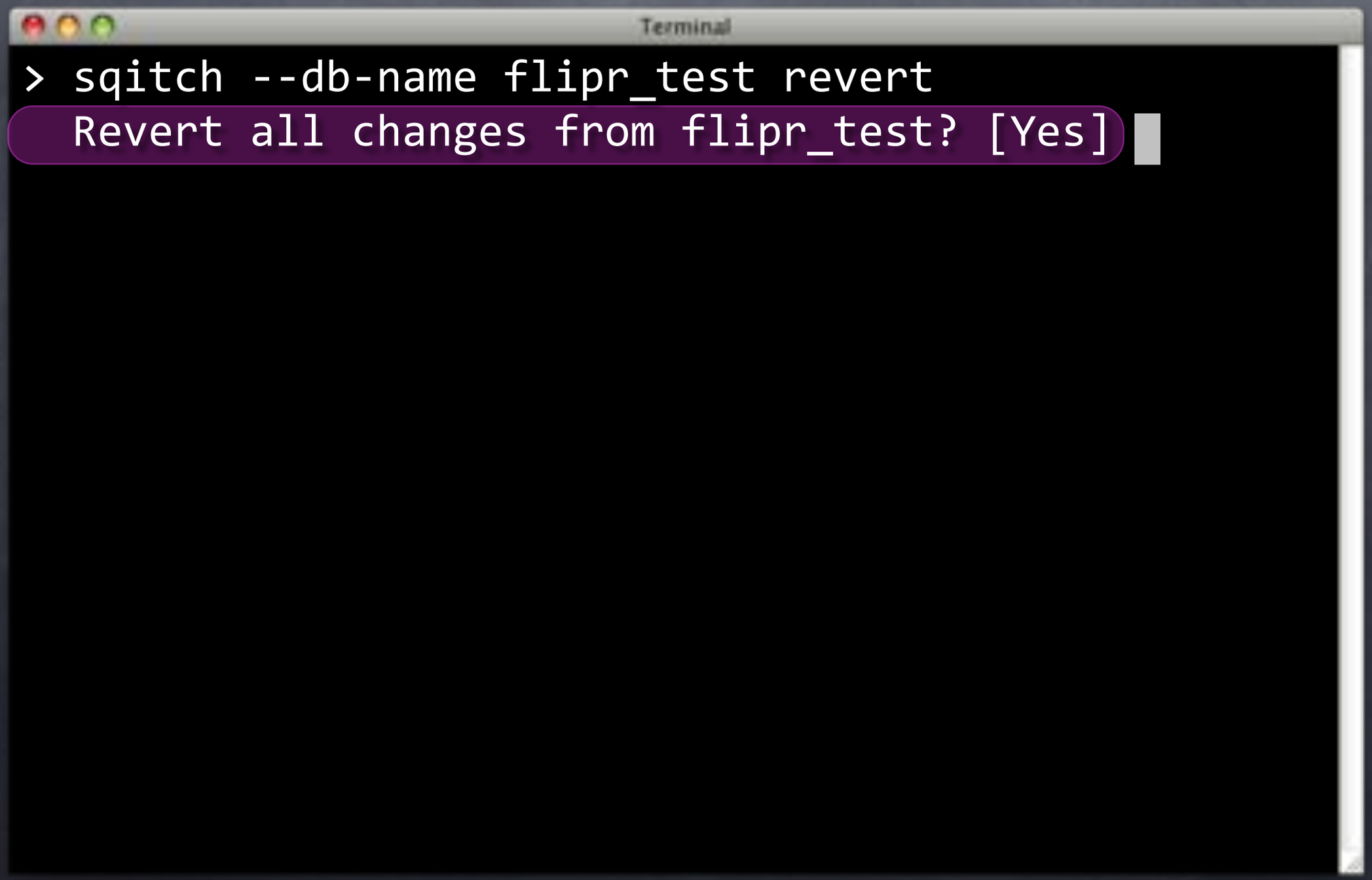
Go Back



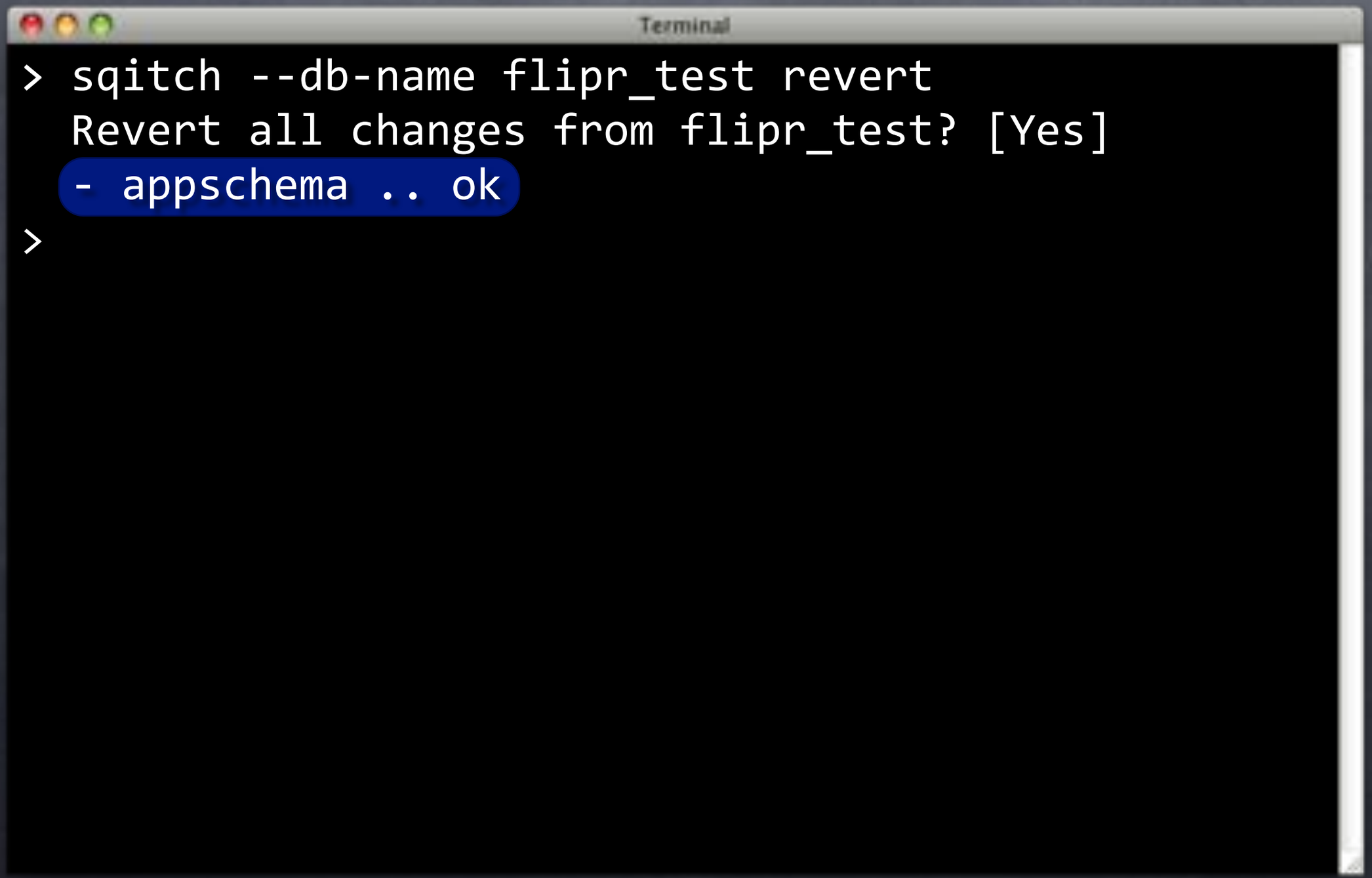
Go Back


Terminal
> sqitch --db-name flipr_test revert
Revert all changes from flipr_test? [Yes]

Go Back


Terminal
> sqitch --db-name flipr_test revert
Revert all changes from flipr_test? [Yes]

Go Back


Terminal
> sqitch --db-name flipr_test revert
Revert all changes from flipr_test? [Yes]
- appschema .. ok
>

Go Back

Terminal

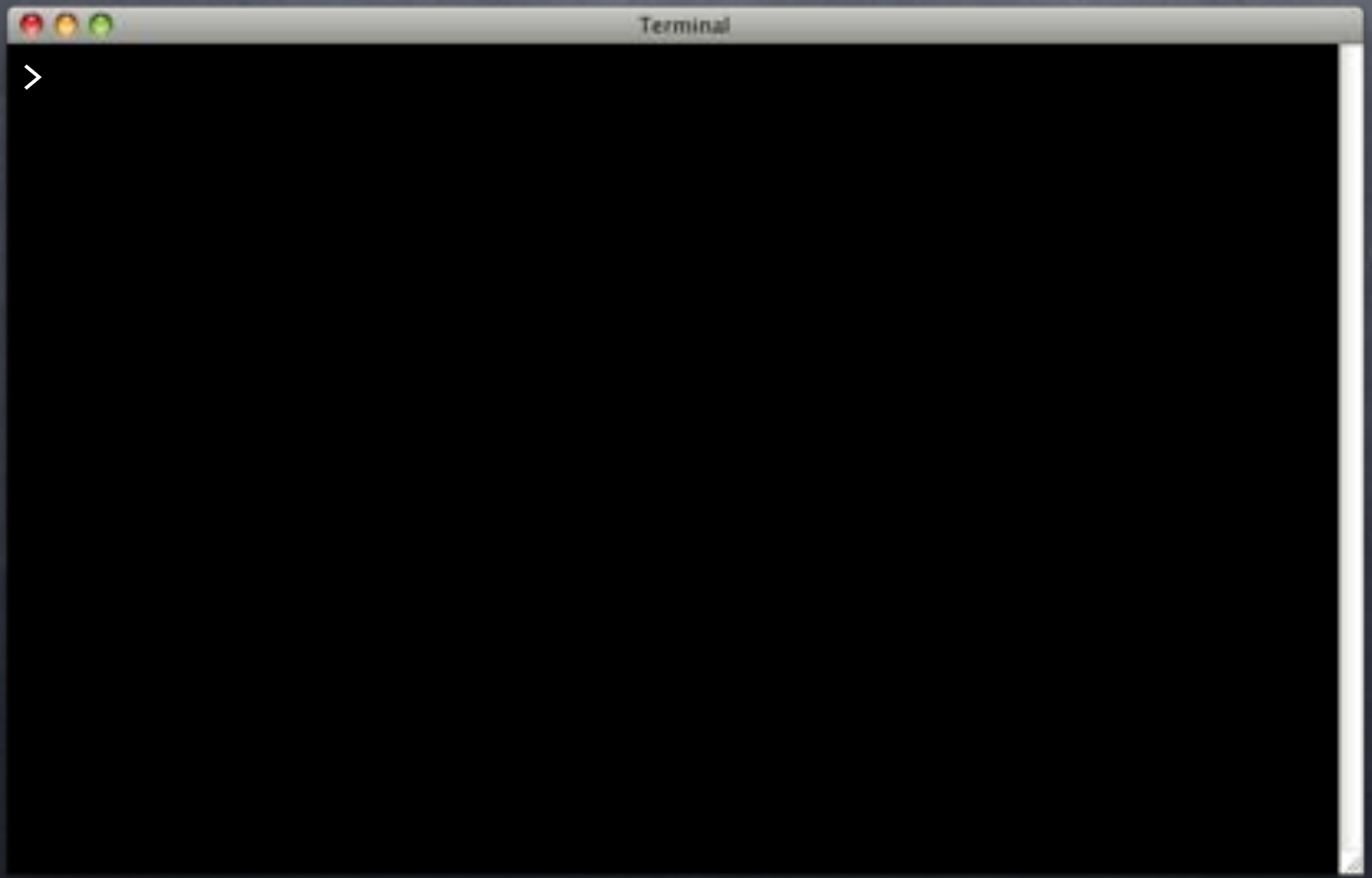
```
> sqitch --db-name flipr_test revert
  Revert all changes from flipr_test? [Yes]
  - appschema .. ok
> psql -d flipr_test -c '\dn flipr'
           List of roles
List of schemas
 Name  | Owner
------+-------
>
```

Go Back

Terminal

```
> sqitch --db-name flipr_test revert
  Revert all changes from flipr_test? [Yes]
  - appschema .. ok
> psql -d flipr_test -c '\dn flipr'
          List of roles
List of schemas
 Name | Owner
------+-------
> sqitch -d flipr_test status
# On database flipr_test
No changes deployed
>
```

History



History

Terminal

```
> sqitch -d flipr_test log
On database flipr_test
Revert 748346dfe73cf2af32a8b7088fd75ad8d7aecda3
Name:      appschema
Committer: David E. Wheeler <david@justatheory.com>
Date:      2013-05-21 15:07:07 -0400

    Adds flipr app schema.

Deploy 748346dfe73cf2af32a8b7088fd75ad8d7aecda3
Name:      appschema
Committer: David E. Wheeler <david@justatheory.com>
Date:      2013-05-21 15:04:08 -0400

    Adds flipr app schema.
```

History

Terminal

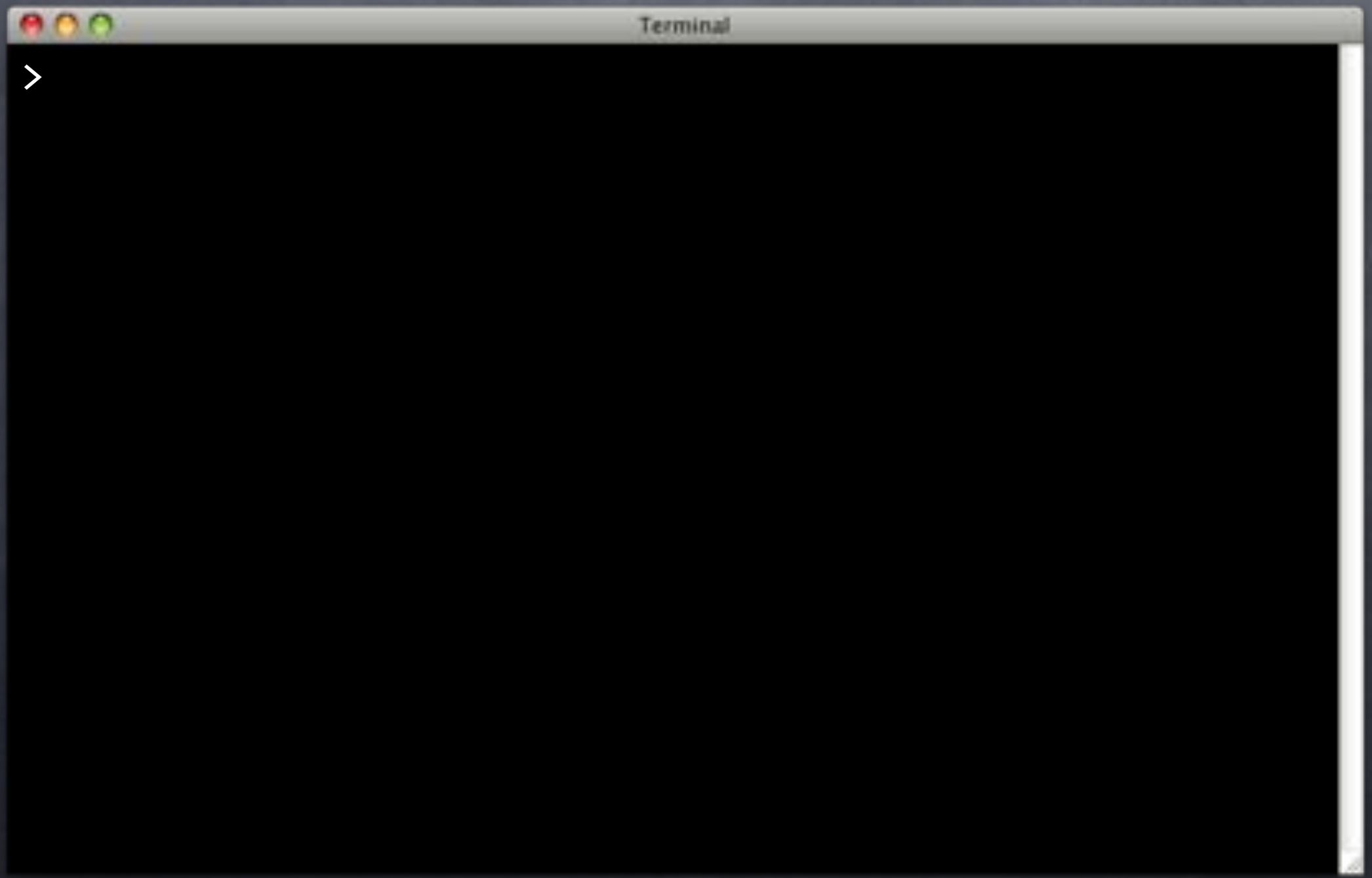
```
> sqitch -d flipr_test log
On database flipr_test
Revert 748346dfe73cf2af32a8b7088fd75ad8d7aecda3
Name:      appschema
Committer: David E. Wheeler <david@justatheory.com>
Date:      2013-05-21 15:07:07 -0400

    Adds flipr app schema.

Deploy 748346dfe73cf2af32a8b7088fd75ad8d7aecda3
Name:      appschema
Committer: David E. Wheeler <david@justatheory.com>
Date:      2013-05-21 15:04:08 -0400

    Adds flipr app schema.
```

History

Terminal

```
> sqitch -d flipr_test log
On database flipr_test
Revert 748346dfe73cf2af32a8b7088fd75ad8d7aecda3
Name:      appschema
Committer: David E. Wheeler <david@justatheory.com>
Date:      2013-05-21 15:07:07 -0400

    Adds flipr app schema.

Deploy 748346dfe73cf2af32a8b7088fd75ad8d7aecda3
Name:      appschema
Committer: David E. Wheeler <david@justatheory.com>
Date:      2013-05-21 15:04:08 -0400

    Adds flipr app schema.
```

Commit It!



Commit It!

Terminal

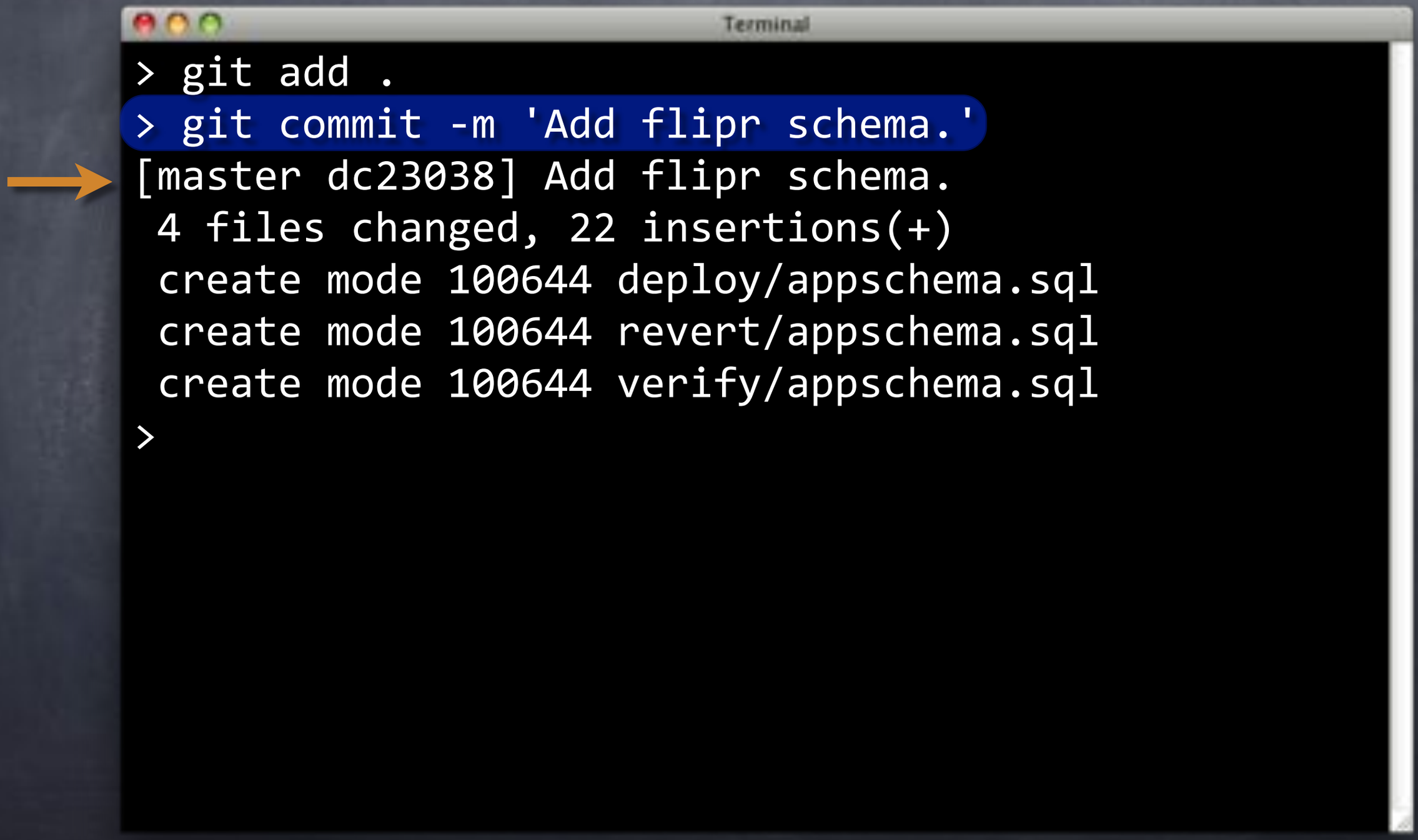
```
> git add .
> git commit -m 'Add flipr schema.'
[master dc23038] Add flipr schema.
 4 files changed, 22 insertions(+)
 create mode 100644 deploy/appschema.sql
 create mode 100644 revert/appschema.sql
 create mode 100644 verify/appschema.sql
>
```

Commit It!

Terminal

```
> git add .
> git commit -m 'Add flipr schema.'
[master dc23038] Add flipr schema.
 4 files changed, 22 insertions(+)
 create mode 100644 deploy/appschema.sql
 create mode 100644 revert/appschema.sql
 create mode 100644 verify/appschema.sql
>
```

Commit It!


Terminal
> git add .
> git commit -m 'Add flipr schema.'
[master dc23038] Add flipr schema.
 4 files changed, 22 insertions(+)
 create mode 100644 deploy/appschema.sql
 create mode 100644 revert/appschema.sql
 create mode 100644 verify/appschema.sql
>

Commit It!

Terminal

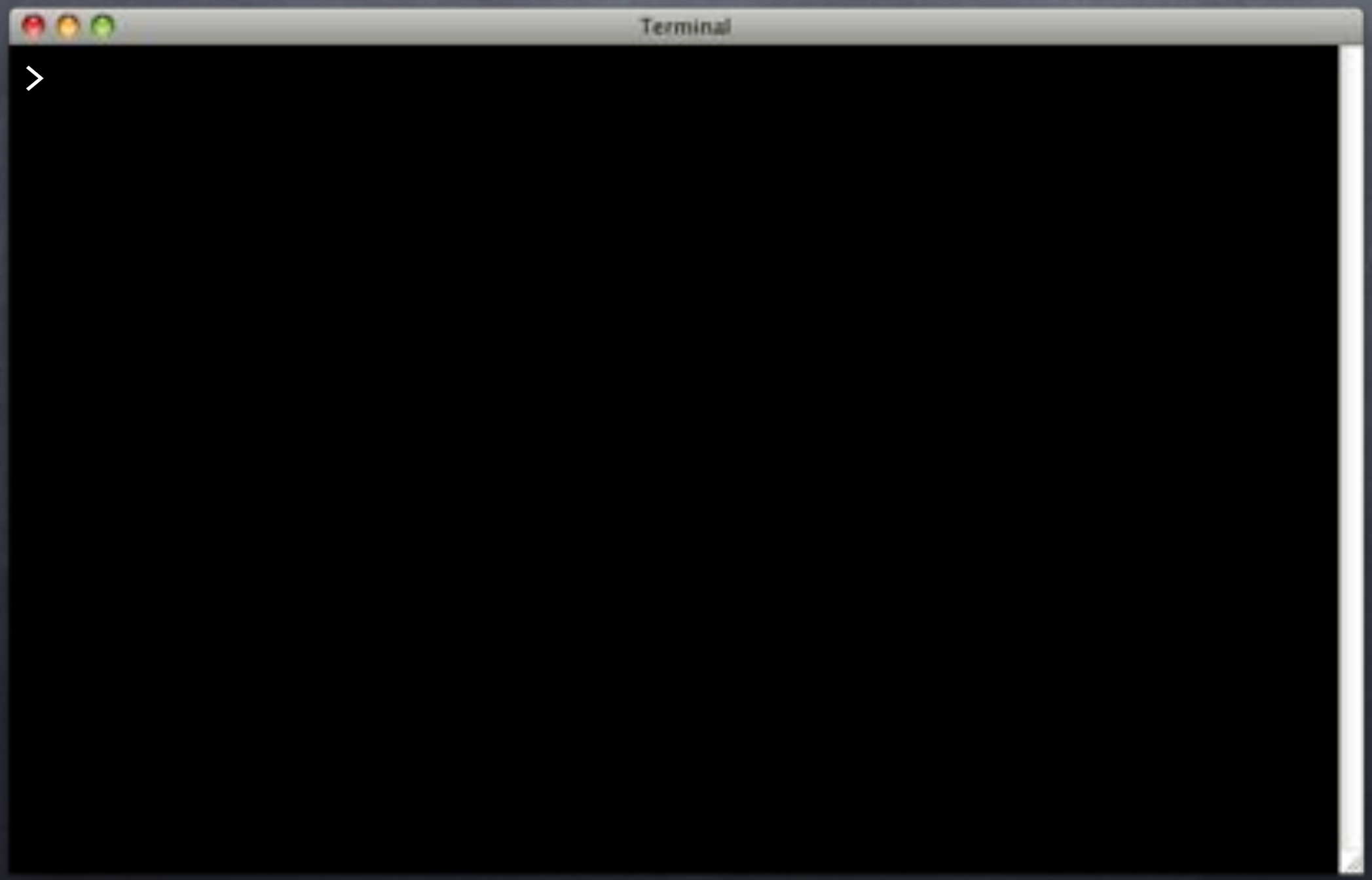
```
> git add .
> git commit -m 'Add flipr schema.'
[master dc23038] Add flipr schema.
 4 files changed, 22 insertions(+)
 create mode 100644 deploy/appschema.sql
 create mode 100644 revert/appschema.sql
 create mode 100644 verify/appschema.sql
> git push
Counting objects: 11, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 943 bytes, done.
Total 9 (delta 0), reused 0 (delta 0)
To ../flipr-remote
   a4f765f..dc23038  master -> master
```

Commit It!

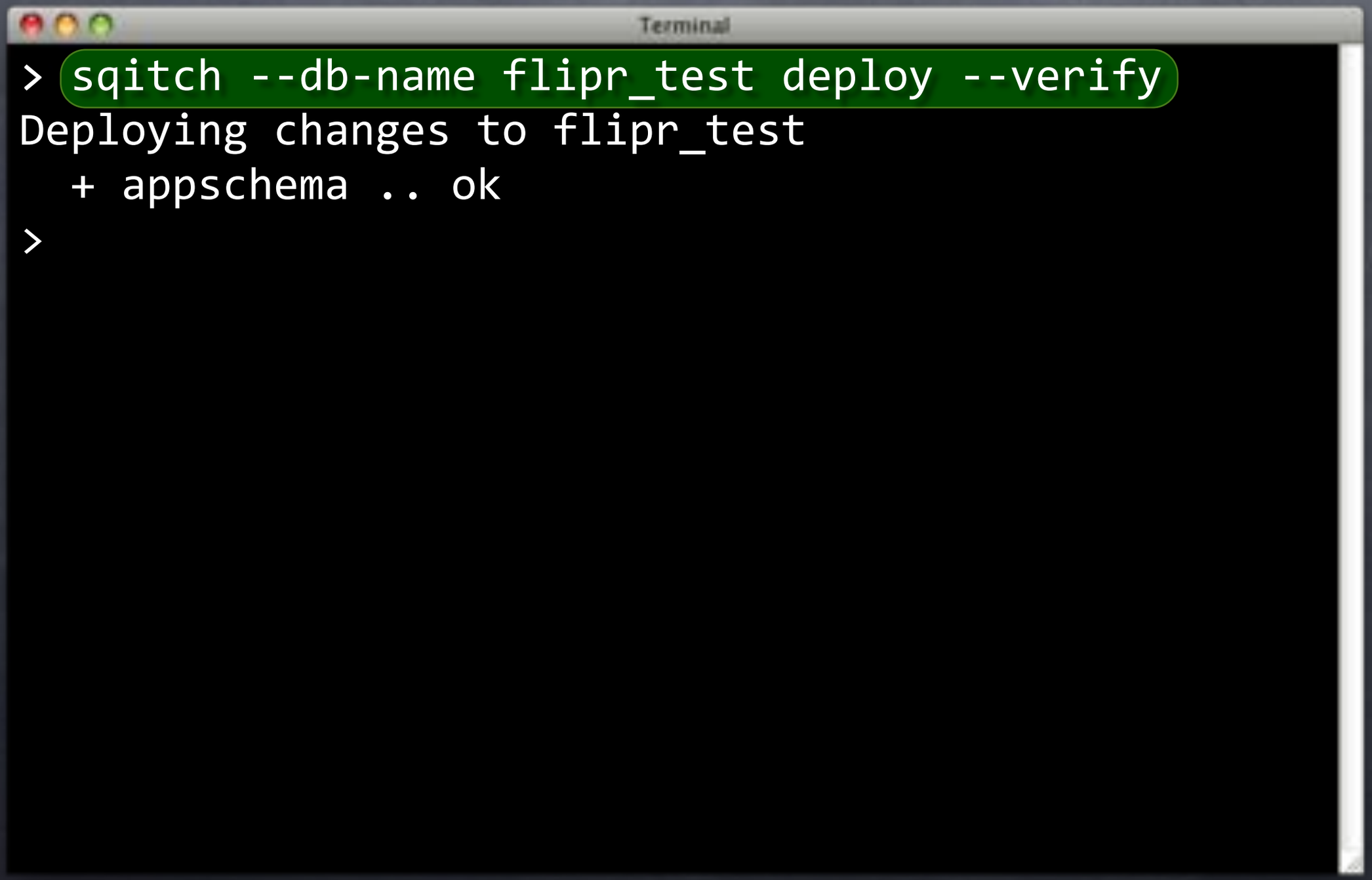
Terminal

```
> git add .
> git commit -m 'Add flipr schema.'
[master dc23038] Add flipr schema.
 4 files changed, 22 insertions(+)
 create mode 100644 deploy/appschema.sql
 create mode 100644 revert/appschema.sql
 create mode 100644 verify/appschema.sql
> git push
Counting objects: 11, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 943 bytes, done.
Total 9 (delta 0), reused 0 (delta 0)
To ../flipr-remote
   a4f765f..dc23038  master -> master
```

Redeploy



Redeploy


Terminal
> sqitch --db-name flipr_test deploy --verify
Deploying changes to flipr_test
  + appschema .. ok
>

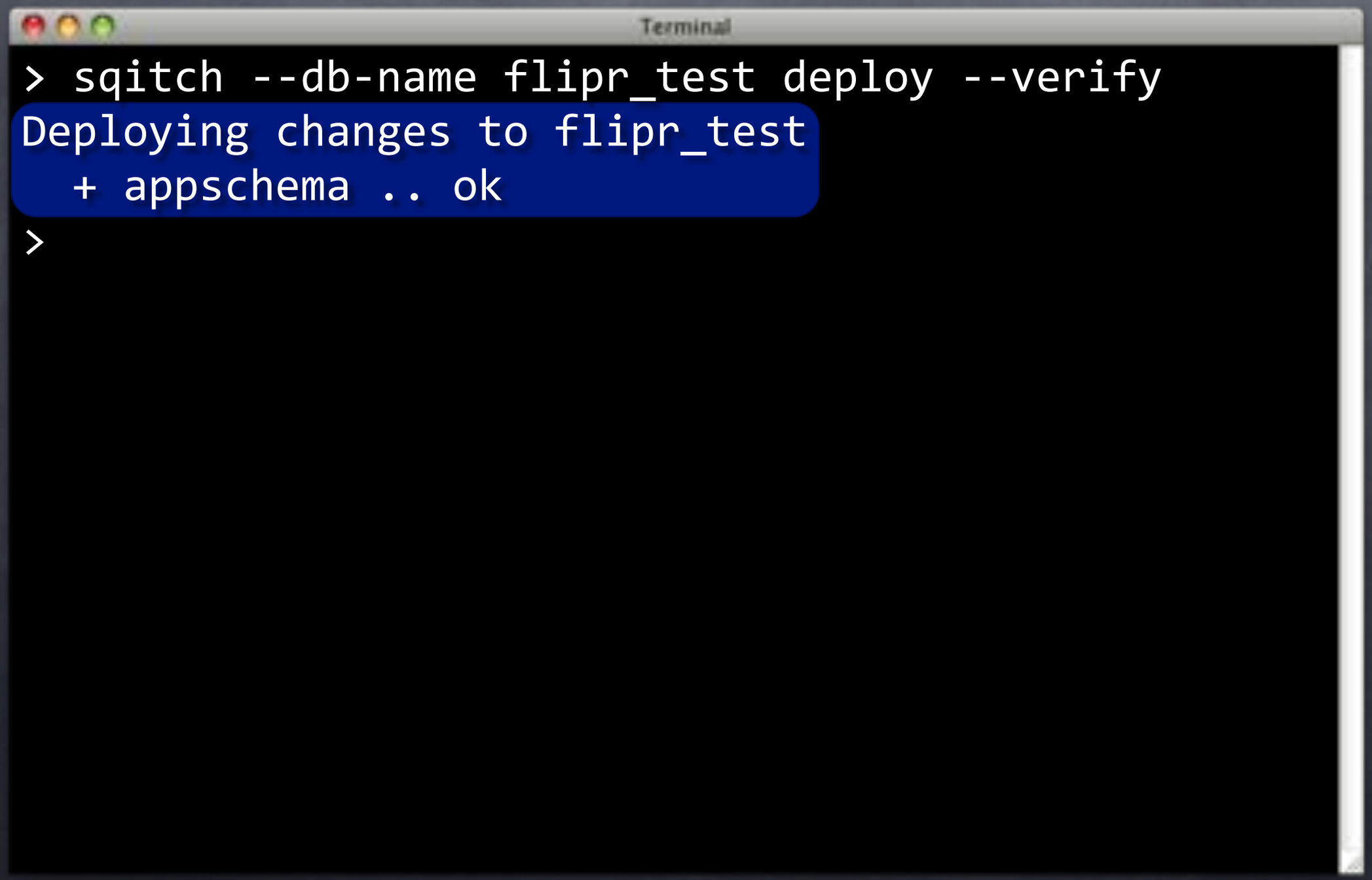
Redeploy

Integrated!

Terminal

```
> sqitch --db-name flipr_test deploy --verify
Deploying changes to flipr_test
  + appschema .. ok
>
```

Redeploy

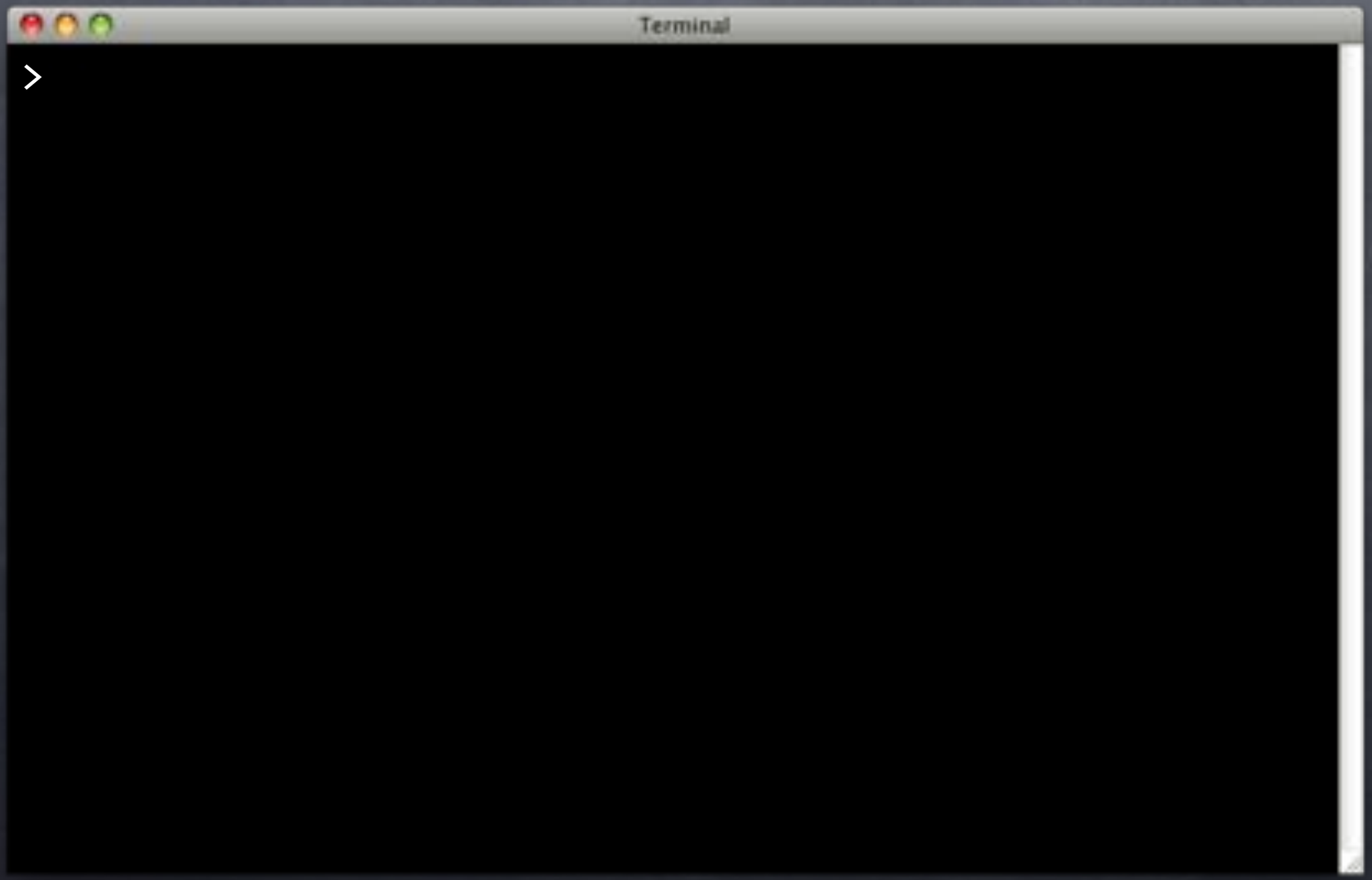

Terminal
> sqitch --db-name flipr_test deploy --verify
Deploying changes to flipr_test
  + appschema .. ok
>

Redeploy

Terminal

```
> sqitch --db-name flipr_test deploy --verify
Deploying changes to flipr_test
  + appschema .. ok
> psql -d flipr_test -c '\dn flipr'
List of schemas
 Name  | Owner
-------+-------
 flipr | david
>
```

Status Update

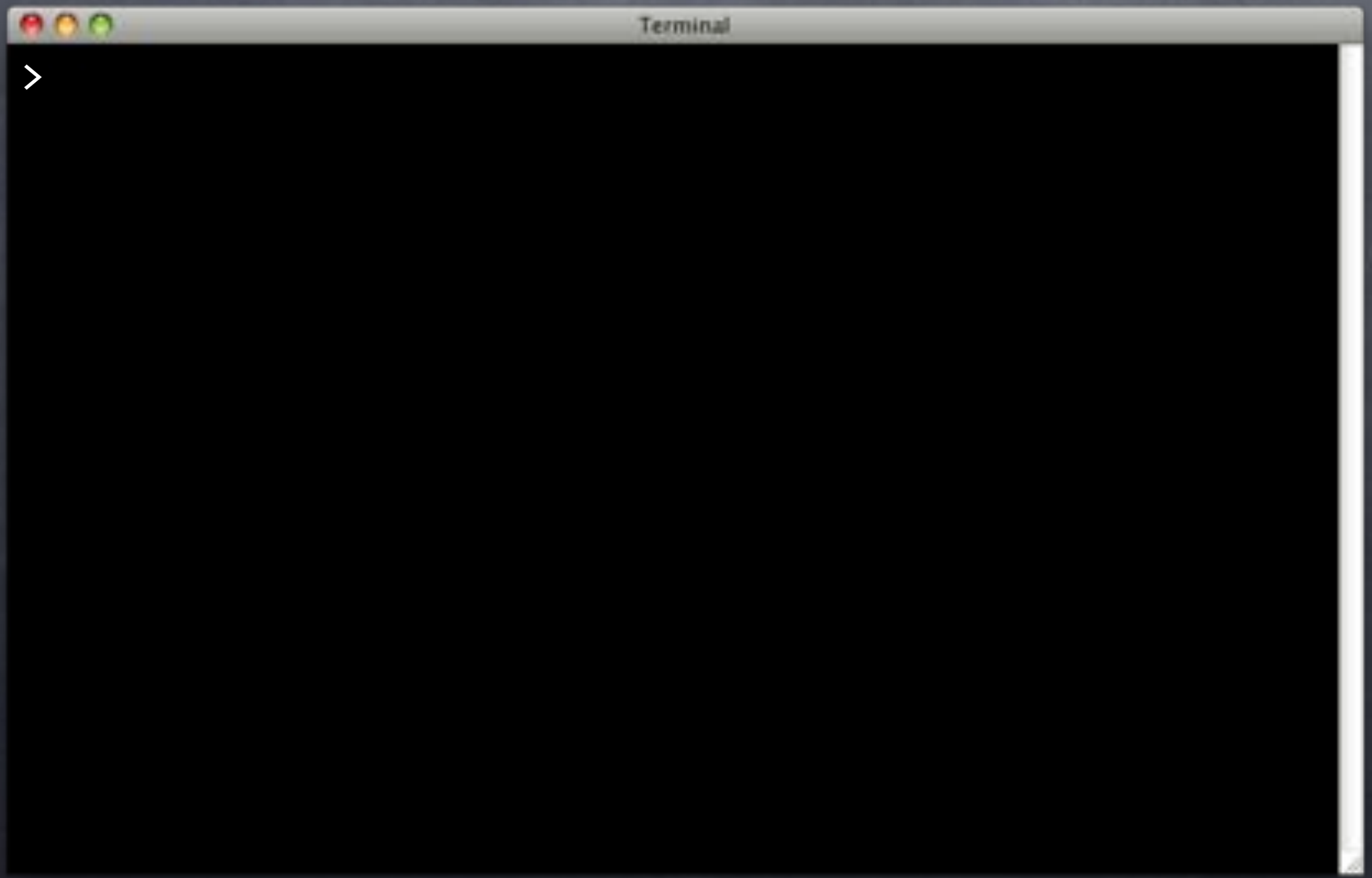


Status Update

Terminal

```
> sqitch -d flipr_test status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8b7088fd75ad8d7aecda3
# Name:     appschema
# Deployed: 2013-05-21 15:10:02 -0400
# By:       David E. Wheeler <david@justatheory.com>
#
Nothing to deploy (up-to-date)
>
```

Save My Fingers



Save My Fingers

Save My Fingers

Terminal

```
> sqitch config core.pg.db_name flipr_test
> sqitch status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8                          ecda3
# Name:     appschema
# Deployed: 2013-05-21 15:10:02 -0400
# By:       David E. Wheeler <david@justatheory.com>
#
Nothing to deploy (up-to-date)
>
```

No --db-name

Save My Fingers

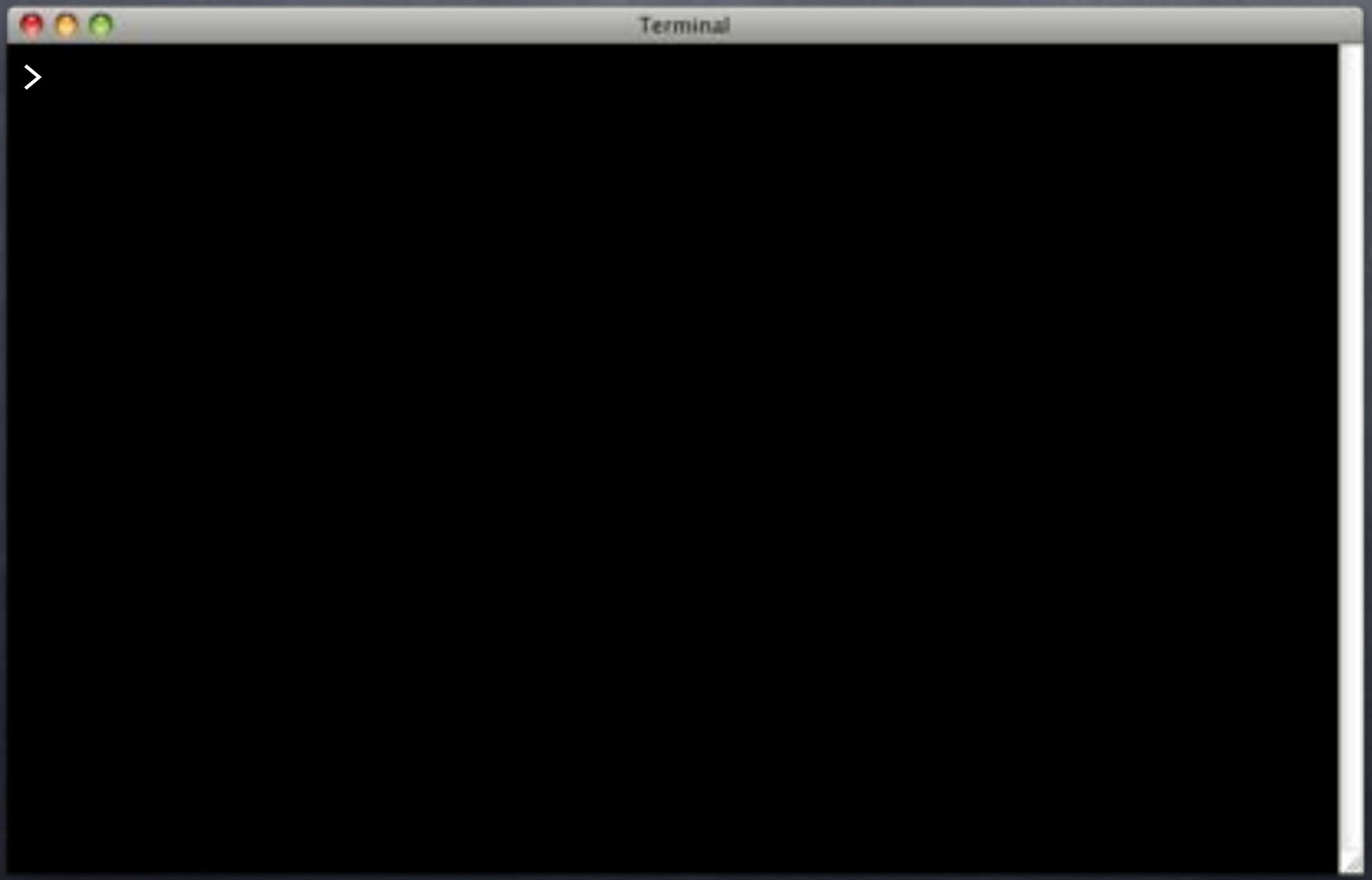
Terminal

```
> sqitch config core.pg.db_name flipr_test
> sqitch status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8                          ecda3
# Name:     appschema
# Deployed: 2013-05-21 15:10:02 -0400
# By:       David E. Wheeler <david@justatheory.com>
#
Nothing to deploy (up-to-date)
> sqitch config --bool deploy.verify true
> sqitch config --bool rebase.verify true
>
```

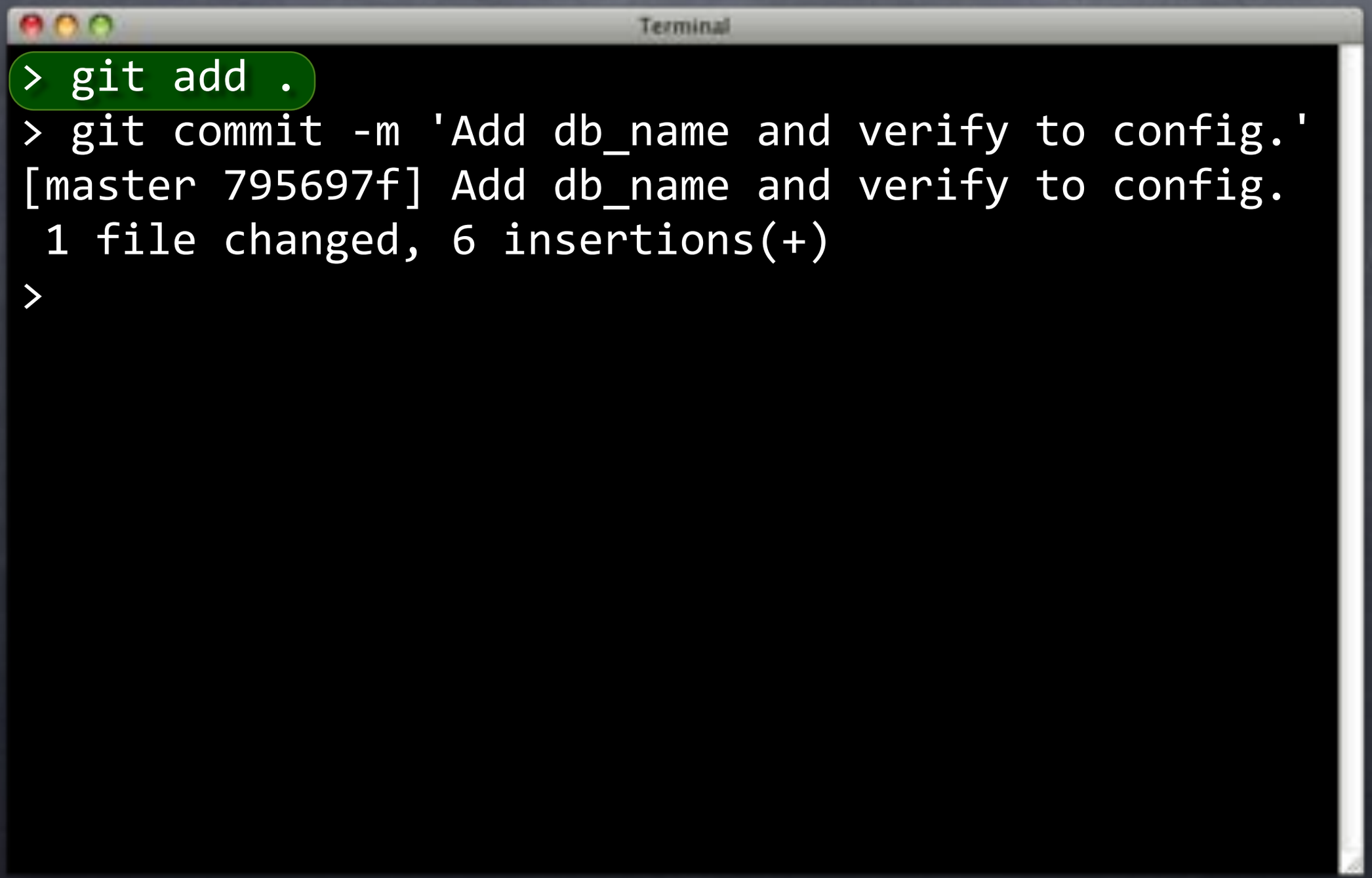
No --db-name

Always verify.

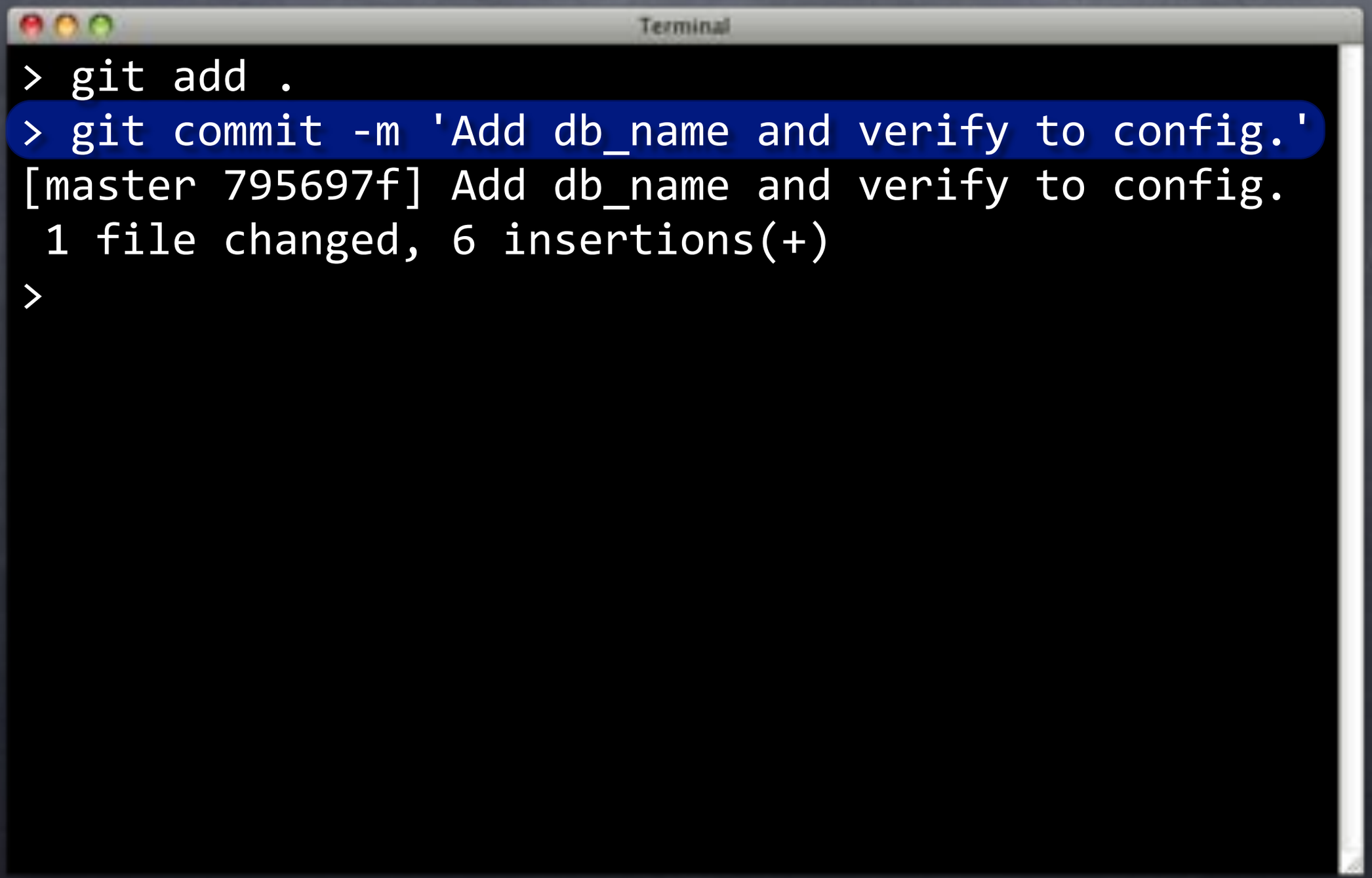
Commit Config



Commit Config



Commit Config



Commit Config

Terminal

```
> git add .
> git commit -m 'Add db_name and verify to config.'
[master 795697f] Add db_name and verify to config.
 1 file changed, 6 insertions(+)
> git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 376 bytes, done.
Total 3 (delta 2), reused 0 (delta 0)
To ../flipr-remote
   dc23038..795697f  master -> master
>
```

Not Migrations?

Not Migrations?

- Incomplete mini-language

Not Migrations?

- Incomplete mini-language
- No logical replication integration

Not Migrations?

- Incomplete mini-language
- No logical replication integration
- Numbered scripts difficult to track

Not Migrations?

- Incomplete mini-language
- No logical replication integration
- Numbered scripts difficult to track
- No VCS awareness

SQL Migrations?

- Incomplete mini-language
- No logical replication integration
- Numbered scripts hard to track
- No VCS awareness

SQL Migrations?

- ~~Incomplete mini-language~~
- No logical replication integration
- Numbered scripts hard to track
- No VCS awareness

SQL Migrations?

- ~~Incomplete mini-language~~
- ~~No logical replication integration~~
- Numbered scripts hard to track
- No VCS awareness

SQL Migrations?

- ~~Incomplete mini-language~~
- ~~No logical replication integration~~
- Numbered scripts hard to track
- No VCS awareness
- Managing procedures is a PITA

Sq—what?

sql changes

Sq—what?

sq ch

Sq—what?

sqitch

Sq—what?

sqitch

There is no “u”

Sqitch Philosophy

Sqitch Philosophy

- No opinions

Sqitch Philosophy

- No opinions
- Native scripting (psql, sqlite3, SQL*Plus)

Sqitch Philosophy

- No opinions
- Native scripting (psql, sqlite3, SQL*Plus)
- Cross-project dependency resolution

Sqitch Philosophy

- No opinions
- Native scripting (psql, sqlite3, SQL*Plus)
- Cross-project dependency resolution
- Distribution bundling

Sqitch Philosophy

- No opinions
- Native scripting (psql, sqlite3, SQL*Plus)
- Cross-project dependency resolution
- Distribution bundling
- Integrated verification testing

Sqitch Philosophy

- No opinions
- Native scripting (psql, sqlite3, SQL*Plus)
- Cross-project dependency resolution
- Distribution bundling
- Integrated verification testing
- No numbering

Sqitch Philosophy

- No opinions
- Native scripting (psql, sqlite3, SQL*Plus)
- Cross-project dependency resolution
- Distribution bundling
- Integrated verification testing
- No numbering
- Reliable sequential deployment ordering

SHAzbat

SHAzbat

- SHA1 ID for every object

SHAzbat

- SHA1 ID for every object
  - Stolen from Git

SHAzbat

- SHA1 ID for every object
  - Stolen from Git
  - change, tag, deploy, revert, verify

SHAzbat

- SHA1 ID for every object
  - Stolen from Git
  - change, tag, deploy, revert, verify
- Hashed change text includes:

SHAzbat

- SHA1 ID for every object
  - Stolen from Git
  - change, tag, deploy, revert, verify
- Hashed change text includes:
  - Project

SHAzbat

- SHA1 ID for every object
  - Stolen from Git
  - change, tag, deploy, revert, verify
- Hashed change text includes:
  - Project
  - Name

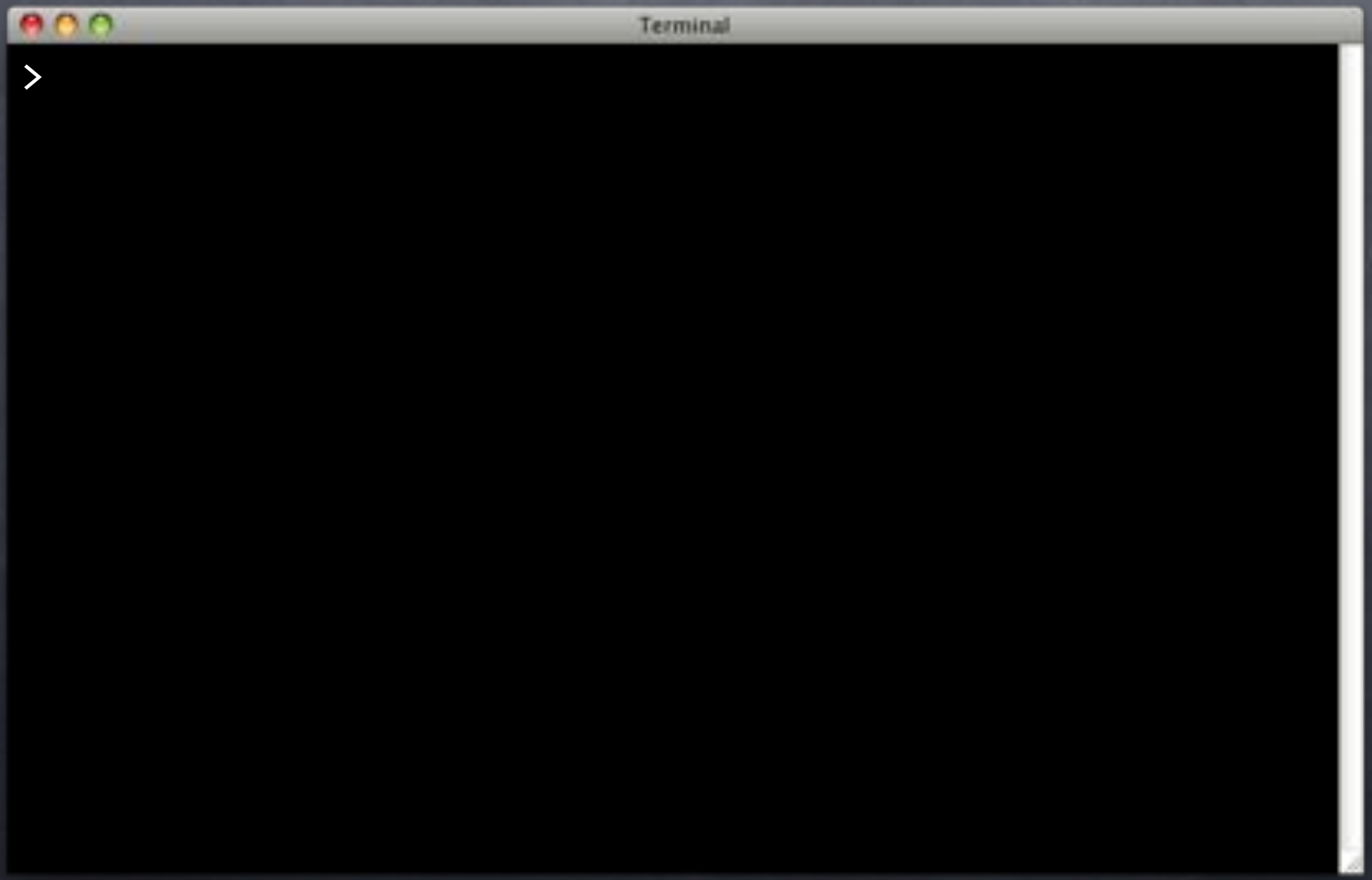
SHAzbat

- SHA1 ID for every object
  - Stolen from Git
  - change, tag, deploy, revert, verify
- Hashed change text includes:
  - Project
  - Name
  - Parent ID

SHAzbat

- SHA1 ID for every object
  - Stolen from Git
  - change, tag, deploy, revert, verify
- Hashed change text includes:
  - Project
  - Name
  - Parent ID
  - Planner...

SHAsome



SHAsome

Terminal

```
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/pgxn-manager.git
change distview
parent 39d7e105cb00e1798cfabb45f4b74cbc14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
  + roles
  + pgxn_core:types
  + distributions
  + extensions

Adds the filters table.
>
```

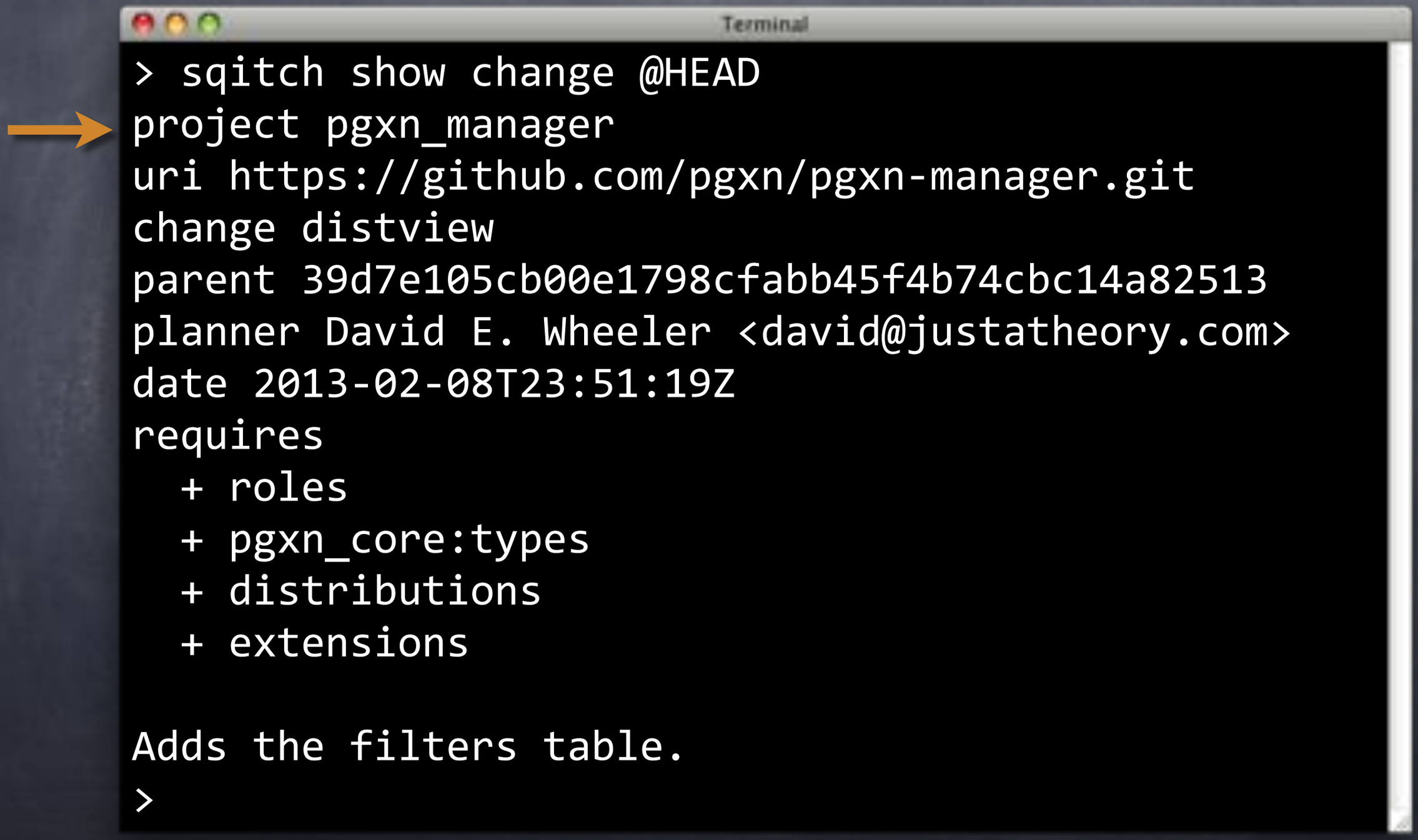
SHAsome

Terminal

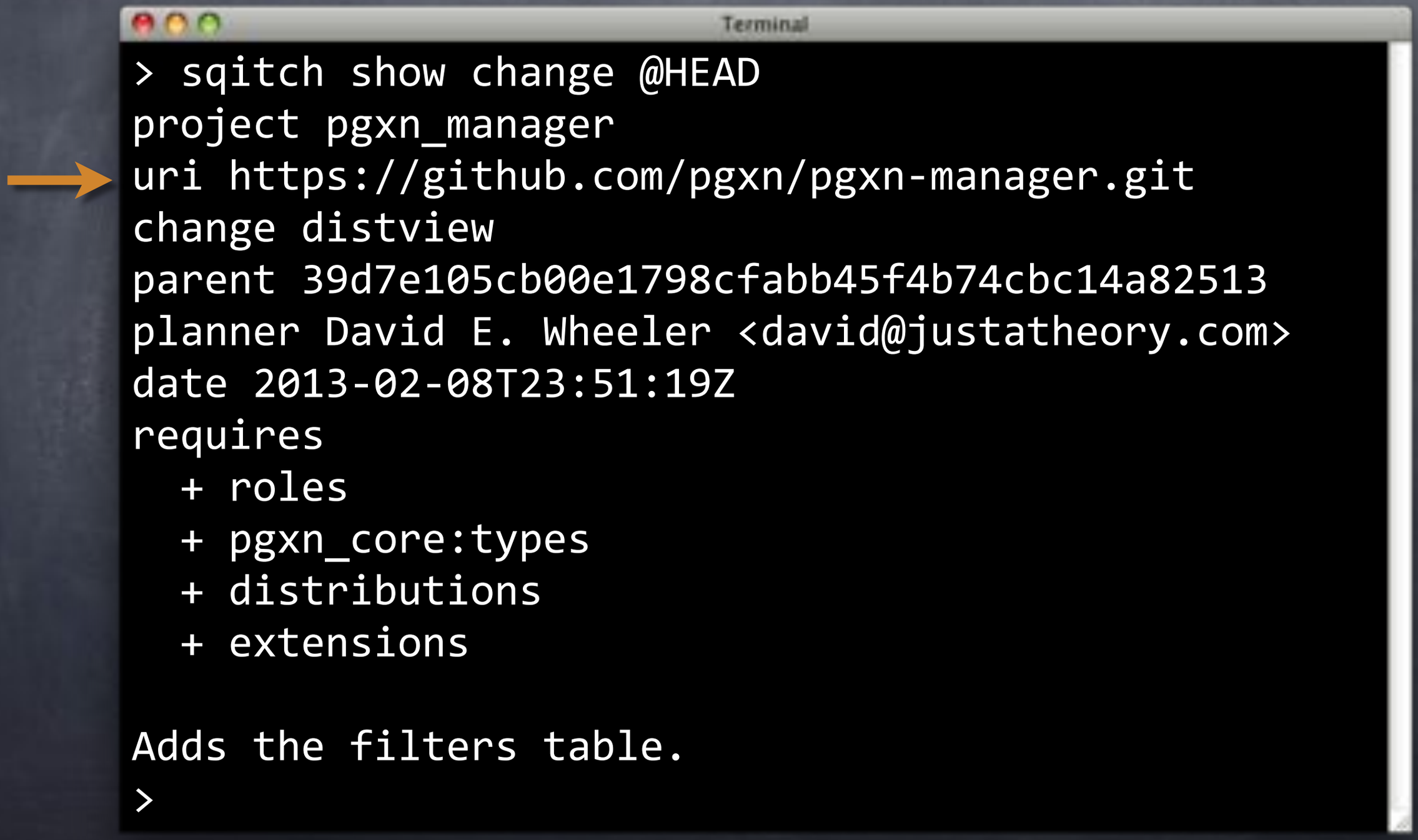
```
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/
change distview
parent 39d7e105cb00e1798cfabb451407c0c14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
  + roles
  + pgxn_core:types
  + distributions
  + extensions

Adds the filters table.
>
```

Symbolic Sqitch tag

SHAsome


Terminal
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/pgxn-manager.git
change distview
parent 39d7e105cb00e1798cfabb45f4b74cbc14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
+ roles
+ pgxn_core:types
+ distributions
+ extensions
Adds the filters table.
>

SHAsome


Terminal
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/pgxn-manager.git
change distview
parent 39d7e105cb00e1798cfabb45f4b74cbc14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
+ roles
+ pgxn_core:types
+ distributions
+ extensions
Adds the filters table.
>

SHAsome

Terminal

```
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/pgxn-manager.git
change distview
parent 39d7e105cb00e1798cfabb45f4b74cbc14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
  + roles
  + pgxn_core:types
  + distributions
  + extensions

Adds the filters table.
>
```

SHAsome

Terminal

```
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/pgxn-manager.git
change distview
parent 39d7e105cb00e1798cfabb45f4b74cbc14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
  + roles
  + pgxn_core:types
  + distributions
  + extensions

Adds the filters table.
>
```

SHAsome

Terminal

```
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/pgxn-manager.git
change distview
parent 39d7e105cb00e1798cfabb45f4b74cbc14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
  + roles
  + pgxn_core:types
  + distributions
  + extensions

Adds the filters table.
>
```

SHAsome

Terminal

```
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/pgxn-manager.git
change distview
parent 39d7e105cb00e1798cfabb45f4b74cbc14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
  + roles
  + pgxn_core:types
  + distributions
  + extensions

Adds the filters table.
>
```

SHAsome

Terminal

```
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/pgxn-manager.git
change distview
parent 39d7e105cb00e1798cfabb45f4b74cbc14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
  + roles
  + pgxn_core:types
  + distributions
  + extensions

Adds the filters table.
>
```

SHAsome

Terminal

```
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/pgxn-manager.git
change distview
parent 39d7e105cb00e1798cfabb45f4b74cbc14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
  + roles
  + pgxn_core:types
  + distributions
  + extensions

Adds the filters table.
>
```

SHAsome

Terminal

```
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/pgxn-manager.git
change distview
parent 39d7e105cb00e1798cfabb45f4b74cbc14a82513
planner David E. Wheeler <david@justatheory.com>
date 2013-02-08T23:51:19Z
requires
  + roles
  + pgxn_core:types
  + distributions
  + extensions

Adds the filters table.
>
```

SHApay!

SHApay!

- Each change (except first) includes parent

SHApay!

- Each change (except first) includes parent
- Can trace from any change to the beginning

SHApay!

- Each change (except first) includes parent
- Can trace from any change to the beginning
- Change tampering (corruption) detectable

SHApay!

- Each change (except first) includes parent
- Can trace from any change to the beginning
- Change tampering (corruption) detectable
  - Because the hash will be wrong

SHApay!

- Each change (except first) includes parent
- Can trace from any change to the beginning
- Change tampering (corruption) detectable
  - Because the hash will be wrong
- Stole Linus Torvalds's "greatest invention"

Sqitch Features

Sqitch Features

- Reduced duplication

Sqitch Features

- Reduced duplication
- Built-in configuration

Sqitch Features

- Reduced duplication
- Built-in configuration
- Iterative development

Sqitch Features

- Reduced duplication
- Built-in configuration
- Iterative development
- Deployment planning

Sqitch Features

- Reduced duplication
- Built-in configuration
- Iterative development
- Deployment planning
- Git-style interface

Sqitch Features

- Reduced duplication
- Built-in configuration
- Iterative development
- Deployment planning
- Git-style interface
- Deployment tagging

Your Turn

Your Turn

- Configure Sqitch

Your Turn

- Configure Sqitch
- Initialize project

Your Turn

- Configure Sqitch
- Initialize project
- Add appschema change

Your Turn

- Configure Sqitch
- Initialize project
- Add appschema change
- Deploy/Revert

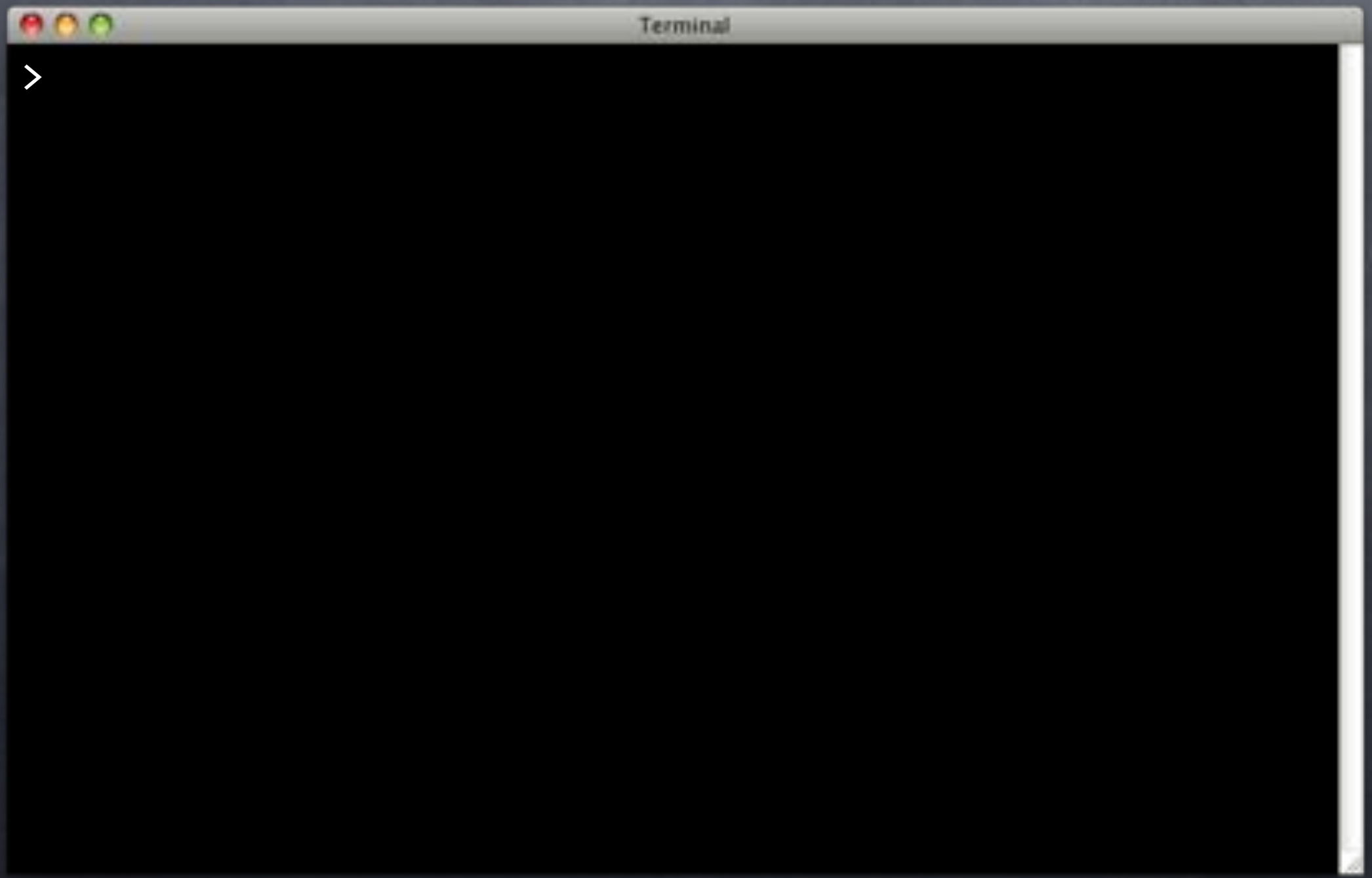
Your Turn

- Configure Sqitch
- Initialize project
- Add appschema change
- Deploy/Revert
- Commit/push

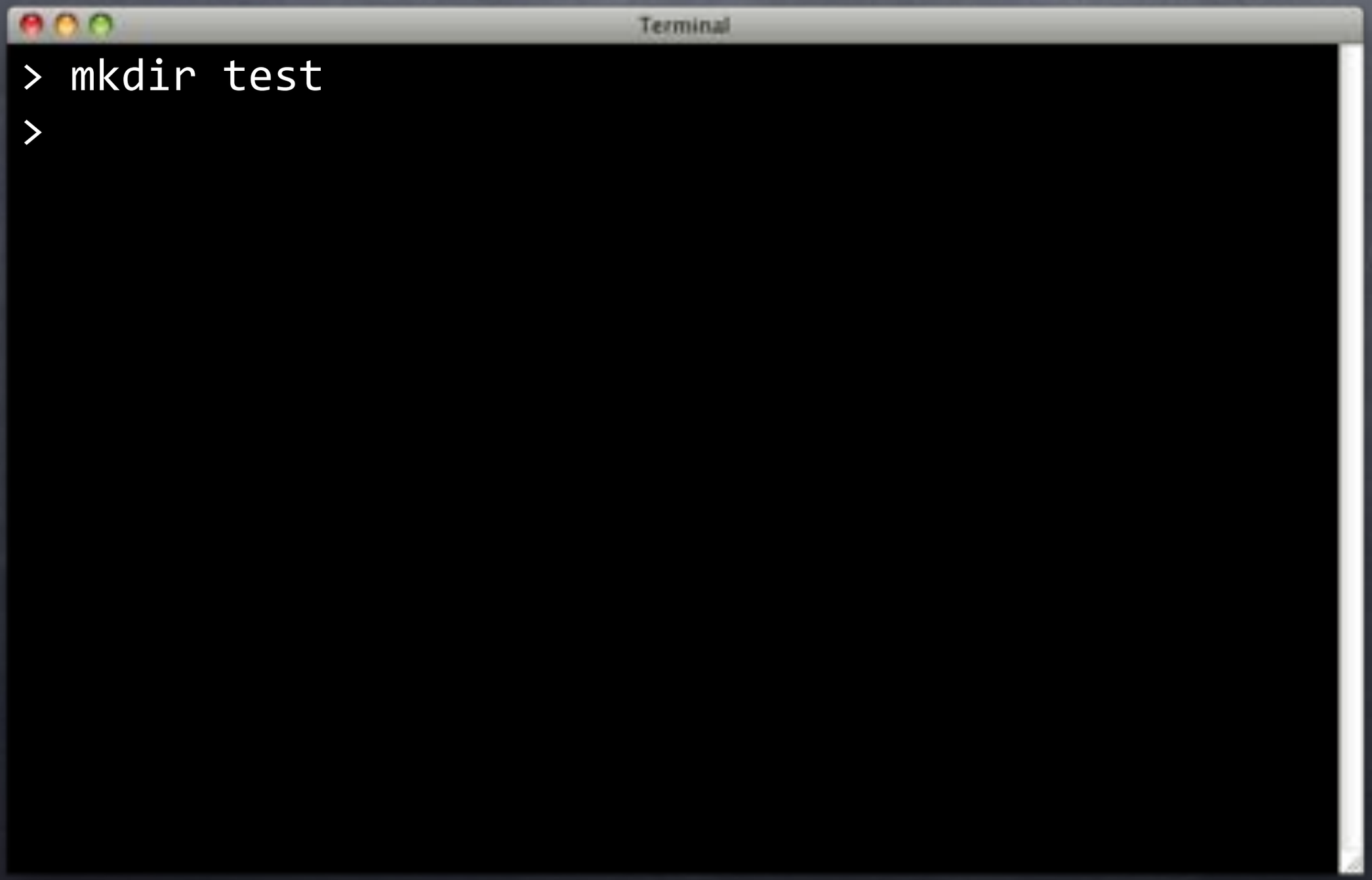
Your Turn

- Configure Sqitch
- Initialize project
- Add appschema change
- Deploy/Revert
- Commit/push
- https://github.com/theory/agile-flipr.git

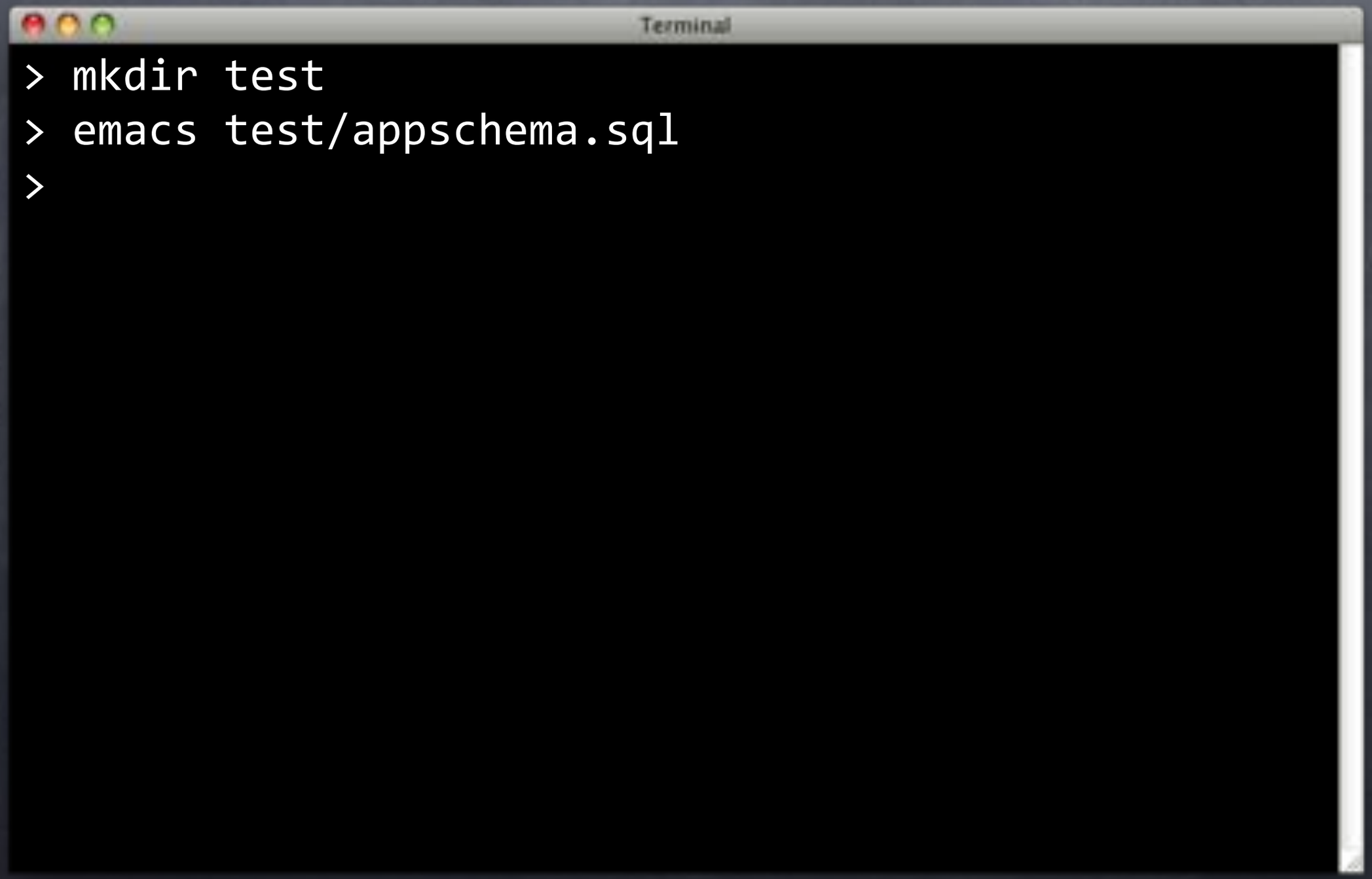
Add Tests



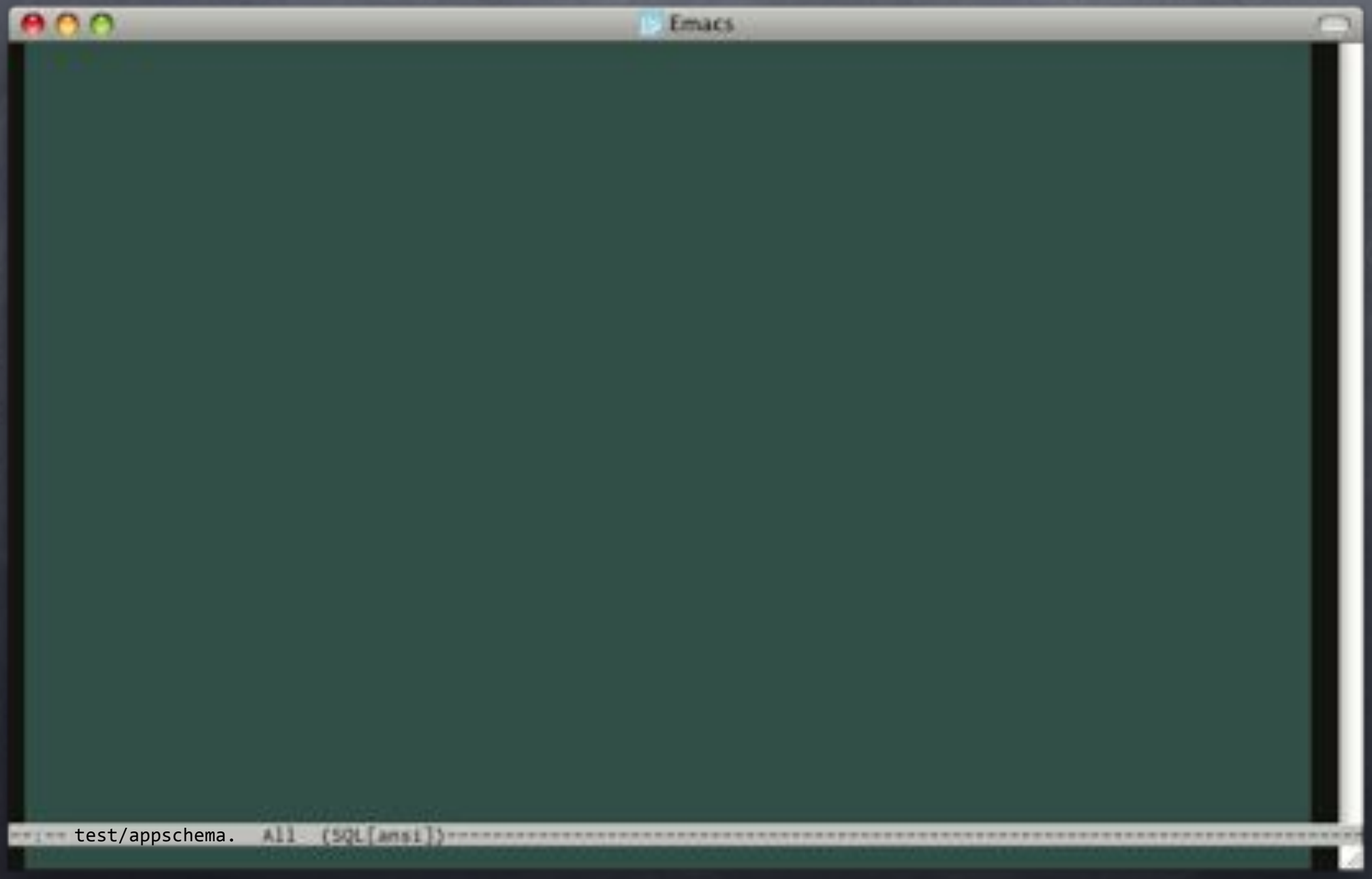
Add Tests



Add Tests



pgTAP Basics



pgTAP Basics

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;

-- Plan the tests.
SELECT plan(1);

SELECT has_schema('nada');

-- Clean up.
SELECT finish();
ROLLBACK;
```

-:** test/appschema. All (SQL[ansi])----------

pgTAP Basics

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;

-- Plan the tests.
SELECT plan(1);

SELECT has_schema('nada');

-- Clean up.
SELECT finish();
ROLLBACK;
```

-:** test/appschema. All (SQL[ansi])----------

pgTAP Basics

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;

-- Plan the tests.
SELECT plan(1);

SELECT has_schema('nada');

-- Clean up.
SELECT finish();
ROLLBACK;
```

-:** test/appschema. All (SQL[ansi])----------

pgTAP Basics

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;

-- Plan the tests.
SELECT plan(1);

SELECT has_schema('nada');

-- Clean up.
SELECT finish();
ROLLBACK;
```

-:** test/appschema. All (SQL[ansi])----------

pgTAP Basics

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;

-- Plan the tests.
SELECT plan(1);

SELECT has_schema('nada');

-- Clean up.
SELECT finish();
ROLLBACK;
```

-:** test/appschema. All (SQL[ansi])----------

pgTAP Basics

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;

-- Plan the tests.
SELECT plan(1);

SELECT has_schema('nada');

-- Clean up.
SELECT finish();
ROLLBACK;
```

-:** test/appschema. All (SQL[ansi])----------

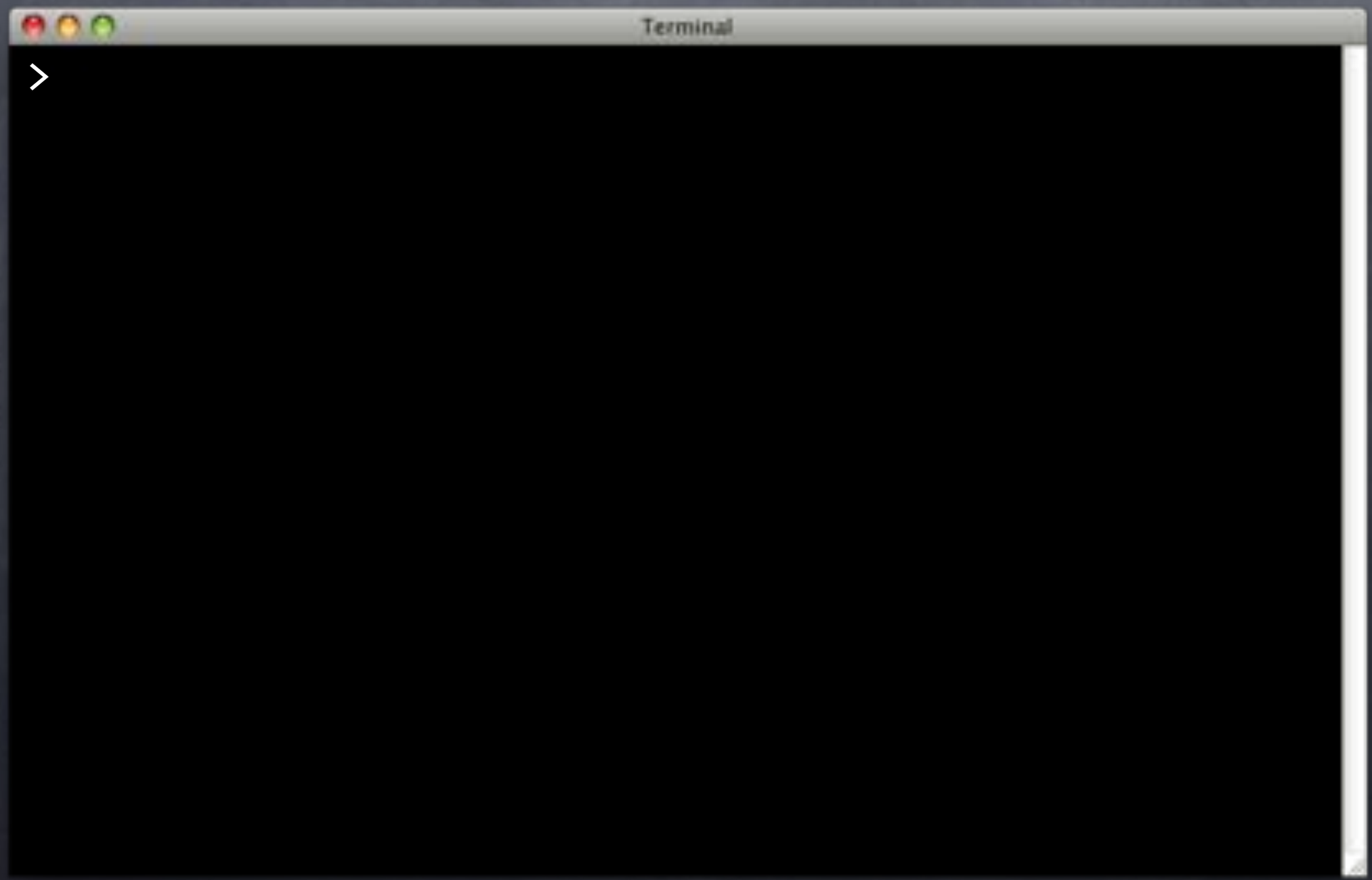
pgTAP Basics

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;

-- Plan the tests.
SELECT plan(1);

SELECT has_schema('nada');

-- Clean up.
SELECT finish();
ROLLBACK;
```

-:** test/appschema. All (SQL[ansi])----------

Run the Test



Run the Test

Terminal

```
> pg_prove -v -d flipr_test test/appschema.sql
test/appschema.sql ..
1..1
not ok 1 - Schema nada should exist
# Failed test 1: "Schema nada should exist"
# Looks like you failed 1 test of 1
Failed 1/1 subtests

Test Summary Report
-------------------
test/appschema.sql (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs
Result: FAIL
```

Run the Test

Terminal

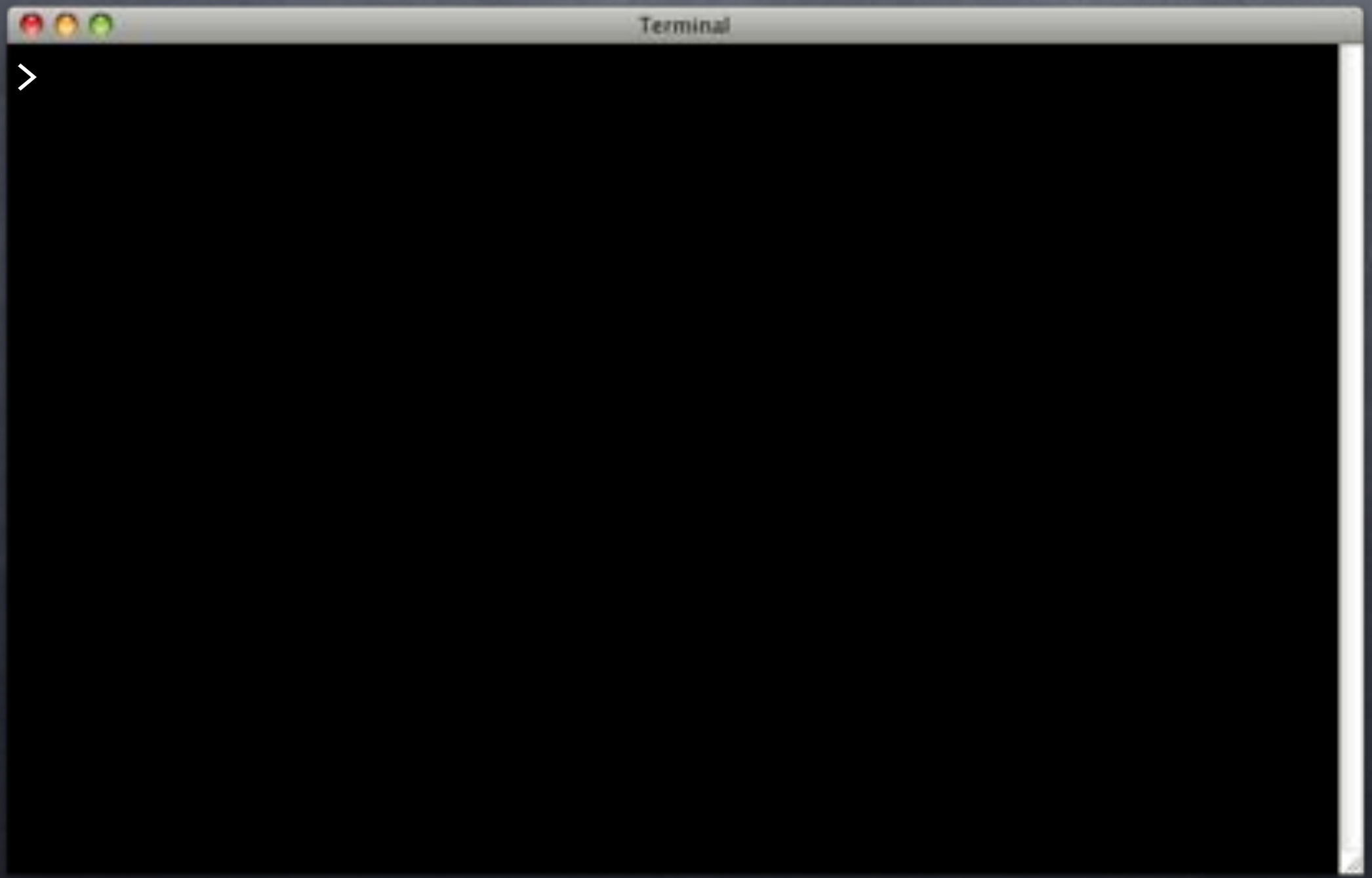
```
> pg_prove -v -d flipr_test test/appschema.sql
test/appschema.sql ..
1..1
not ok 1 - Schema nada should exist
# Failed test 1: "Schema nada should exist"
# Looks like you failed 1 test of 1
Failed 1/1 subtests

Test Summary Report
-------------------
test/appschema.sql (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs
Result: FAIL
```

Run the Test

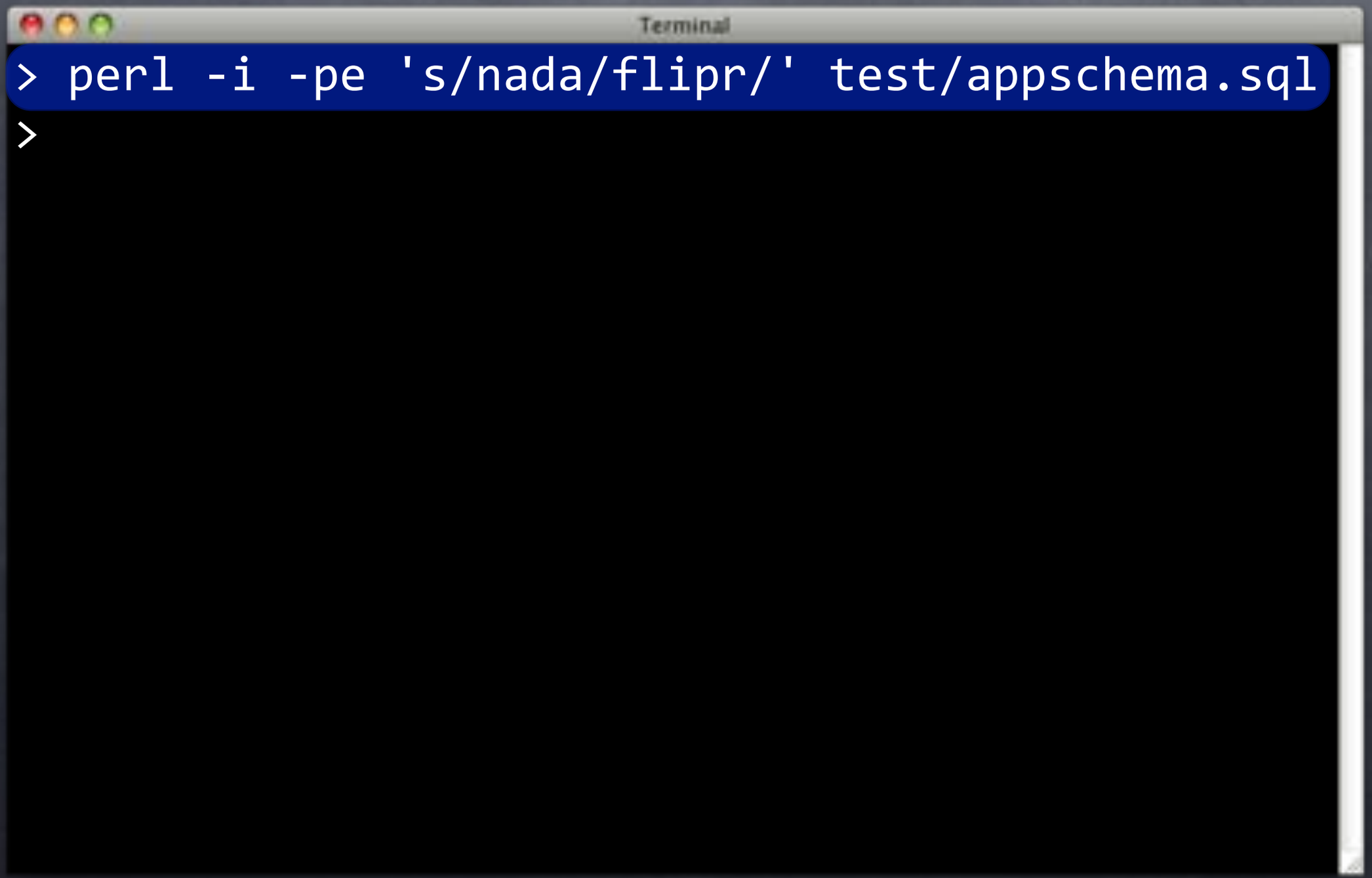
Terminal

```
> pg_prove -v -d flipr_test test/appschema.sql
test/appschema.sql ..
1..1
not ok 1 - Schema nada should exist
# Failed test 1: "Schema nada should exist"
# Looks like you failed 1 test of 1
Failed 1/1 subtests

Test Summary Report
-------------------
test/appschema.sql (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs
Result: FAIL
```

First Pass



First Pass



First Pass

Terminal

```
> perl -i -pe 's/nada/flipr/' test/appschema.sql
> pg_prove -v -d flipr_test test/appschema.sql
test/appschema.sql ..
1..1
ok 1 - Schema flipr should exist
ok
All tests successful.
Files=1, Tests=1,  0 wallclock secs
Result: PASS
```

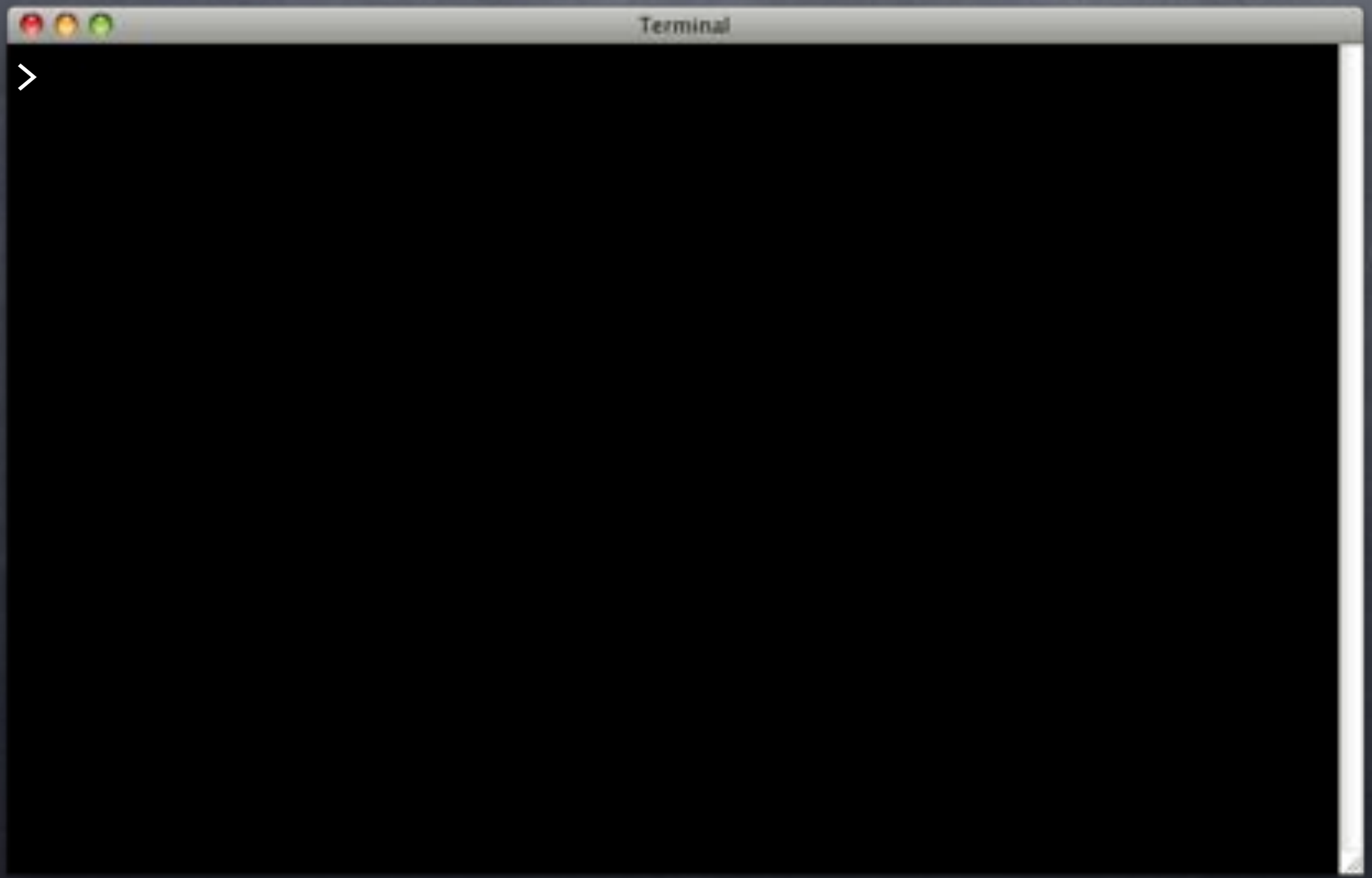
First Pass

Terminal

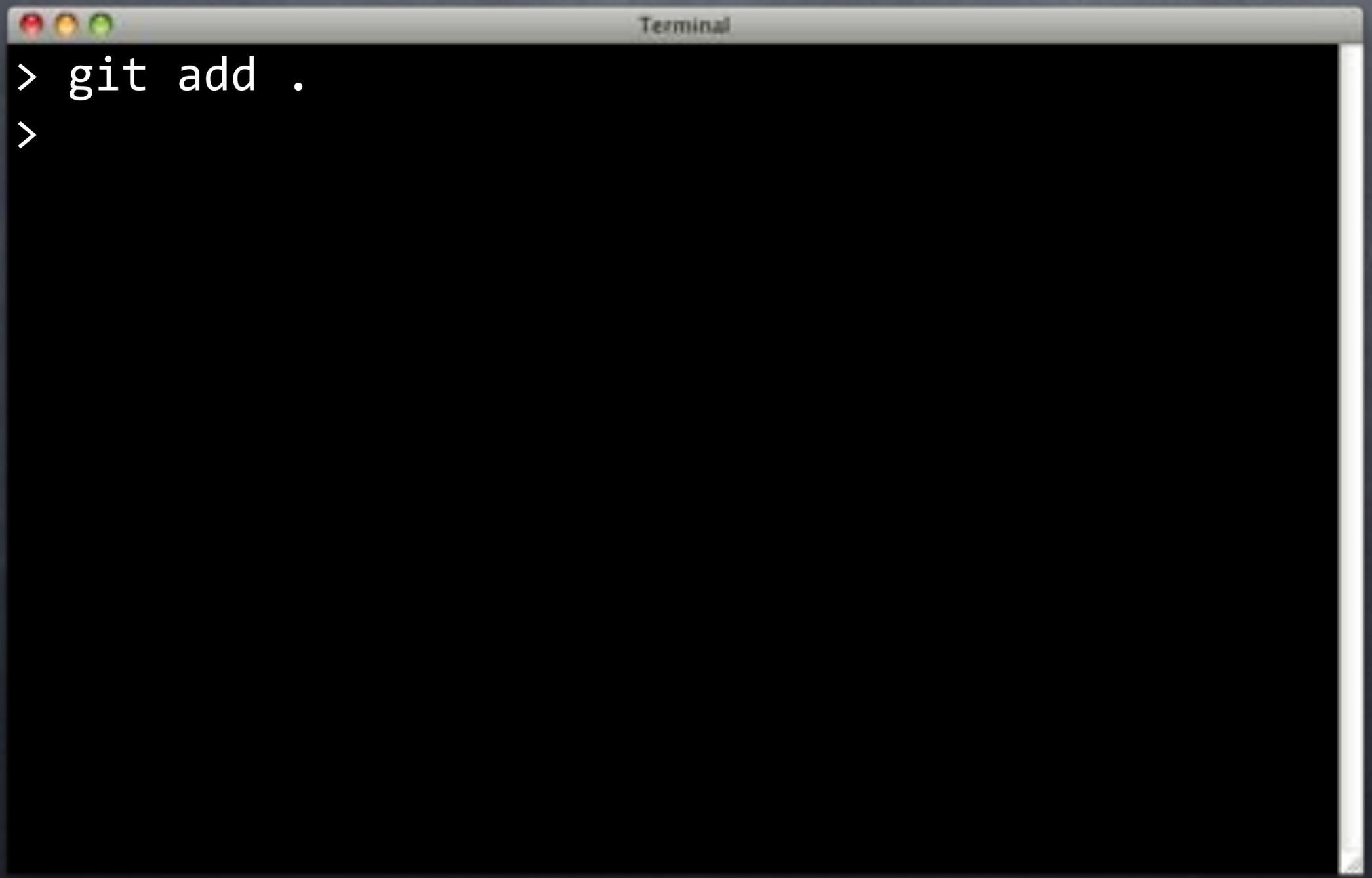
```
> perl -i -pe 's/nada/flipr/' test/appschema.sql
> pg_prove -v -d flipr_test test/appschema.sql
test/appschema.sql ..
1..1
ok 1 - Schema flipr should exist
ok
All tests successful.
Files=1, Tests=1,  0 wallclock secs
Result: PASS
```

WOOt!

Pass it On



Pass it On



Pass it On

Terminal

```
> git add .
> git commit -m 'Add appschema test.'
[master e46bdf9] Add appschema test.
 1 file changed, 14 insertions(+)
 create mode 100644 test/appschema.sql
>
```

Pass it On

Terminal

```
> git add .
> git commit -m 'Add appschema test.'
[master e46bdf9] Add appschema test.
 1 file changed, 14 insertions(+)
 create mode 100644 test/appschema.sql
> git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 487 bytes, done.
Total 4 (delta 1), reused 0 (delta 0)
To ../flipr-remote
   795697f..e46bdf9  master -> master
>
```

OMG TAP WTF?

The Test Anything Protocol (TAP) is a protocol to allow communication between unit tests and a test harness. It allows individual tests (TAP producers) to communicate test results to the testing harness in a language-agnostic way.
—Wikipedia

What is TAP?

What is TAP?

- What does that mean in practice?

What is TAP?

- What does that mean in practice?
- Test output easy to interpret

What is TAP?

- What does that mean in practice?
- Test output easy to interpret
- By humans

What is TAP?

- What does that mean in practice?
- Test output easy to interpret
  - By humans
  - By computers

What is TAP?

- What does that mean in practice?
- Test output easy to interpret
- By humans
- By computers
  - pg_prove (the harness)

What is TAP?

- What does that mean in practice?
- Test output easy to interpret
- By humans
- By computers
  - pg_prove (the harness)
- By aliens

What is TAP?

- What does that mean in practice?
- Test output easy to interpret
- By humans
- By computers
  - pg_prove (the harness)
- By aliens
- By gum!

What's the plan, man?

What's the plan, man?

- Includes Test controls:

What's the plan, man?

- Includes Test controls:
  - plan() — How many tests?

What's the plan, man?

- Includes Test controls:
  - plan() — How many tests?
  - no_plan() — Unknown number of tests

What's the plan, man?

- Includes Test controls:
  - plan() — How many tests?
  - no_plan() — Unknown number of tests
  - diag() — Diagnostic output

What's the plan, man?

- Includes Test controls:
  - plan() — How many tests?
  - no_plan() — Unknown number of tests
  - diag() — Diagnostic output
  - finish() — Test finished, report!

Scalarly

Scalarly

- Includes simple scalar test functions:

Scalarly

- Includes simple scalar test functions:
  - ok() — Boolean

Scalarly

- Includes simple scalar test functions:
  - ok() — Boolean
  - is() — Value comparison

Scalarly

- Includes simple scalar test functions:
  - ok() — Boolean
  - is() — Value comparison
  - isnt() — NOT is()

Scalarly

- Includes simple scalar test functions:
  - ok() — Boolean
  - is() — Value comparison
  - isnt() — NOT is()
  - cmp_ok() — Compare with specific operator

Scalarly

- Includes simple scalar test functions:
  - ok() — Boolean
  - is() — Value comparison
  - isnt() — NOT is()
  - cmp_ok() — Compare with specific operator
  - matches() — Regex comparison

Scalarly

- Includes simple scalar test functions:
  - ok() — Boolean
  - is() — Value comparison
  - isnt() — NOT is()
  - cmp_ok() — Compare with specific operator
  - matches() — Regex comparison
  - imatches() — Case-insensitive regex comparison

It's All Relative

It's All Relative

- Includes functions for testing relations:

It's All Relative

- Includes functions for testing relations:
  - results_eq() — Ordered results

It's All Relative

- Includes functions for testing relations:
  - results_eq() — Ordered results
  - set_eq() — Set of values

It's All Relative

- Includes functions for testing relations:
  - results_eq() — Ordered results
  - set_eq() — Set of values
  - bag_eq() — Bag of values

It's All Relative

- Includes functions for testing relations:
  - results_eq() — Ordered results
  - set_eq() — Set of values
  - bag_eq() — Bag of values
  - row_eq() — Single row

It's All Relative

- Includes functions for testing relations:
  - results_eq() — Ordered results
  - set_eq() — Set of values
  - bag_eq() — Bag of values
  - row_eq() — Single row
  - is_empty() — No results

I'm Okay, You're Okay

I'm Okay, You're Okay

- throws_ok() — Throws an exception

I'm Okay, You're Okay

- throws_ok() — Throws an exception
- throws_like() — Exception matches regex

I'm Okay, You're Okay

- throws_ok() — Throws an exception
- throws_like() — Exception matches regex
- skip() — Skip a subset of tests

I'm Okay, You're Okay

- throws_ok() — Throws an exception
- throws_like() — Exception matches regex
- skip() — Skip a subset of tests
- todo() — Expect subset of tests to fail

Schematics

Schematics

- has_table(), has_view(), has_function(), etc.

Schematics

- has_table(), has_view(), has_function(), etc.
- columns_are(), has_pk(), fk_ok(), etc.

Schematics

- has_table(), has_view(), has_function(), etc.
- columns_are(), has_pk(), fk_ok(), etc.
- col_type_is(), col_not_null(), col_default_is()

Schematics

- has_table(), has_view(), has_function(), etc.
- columns_are(), has_pk(), fk_ok(), etc.
- col_type_is(), col_not_null(), col_default_is()
- So, so much more...

Other Features and Topics

Other Features and Topics

- xUnit-Style testing

Other Features and Topics

- xUnit-Style testing
- Test-Driven development

Other Features and Topics

- xUnit-Style testing
- Test-Driven development
- Integration with Perl unit tests

Other Features and Topics

- xUnit-Style testing
- Test-Driven development
- Integration with Perl unit tests
- Integration with pg_regress

Other Features and Topics

- xUnit-Style testing
- Test-Driven development
- Integration with Perl unit tests
- Integration with pg_regress
- Negative assertions

Other Features and Topics

- xUnit-Style testing
- Test-Driven development
- Integration with Perl unit tests
- Integration with pg_regress
- Negative assertions
- Roles and privileges assertions

Other Features and Topics

- xUnit-Style testing
- Test-Driven development
- Integration with Perl unit tests
- Integration with pg_regress
- Negative assertions
- Roles and privileges assertions
- http://pgtap.org/

Other Features and Topics

- xUnit-Style testing
- Test-Driven development
- Integration with Perl unit tests
- Integration with pg_regress
- Negative assertions
- Roles and privileges assertions
- http://pgtap.org/
- http://pgxn.org/extension/pgtap/

Other Features and Topics

- xUnit-Style testing
- Test-Driven development
- Integration with Perl unit tests
- Integration with pg_regress
- Negative assertions
- Roles and privileges assertions
- http://pgtap.org/
- http://pgxn.org/extension/pgtap/

Let's do it!

Let's do it!

- Create appschema test

Let's do it!

- Create appschema test
- Use pgTAP

Let's do it!

- Create appschema test
- Use pgTAP
- Run test with pg_prove

Let's do it!

- Create appschema test
- Use pgTAP
- Run test with pg_prove
- Make it fail

Let's do it!

- Create appschema test
- Use pgTAP
- Run test with pg_prove
- Make it fail
- Make it pass!

Let's do it!

- Create appschema test
- Use pgTAP
- Run test with pg_prove
- Make it fail
- Make it pass!
- Commit/Push

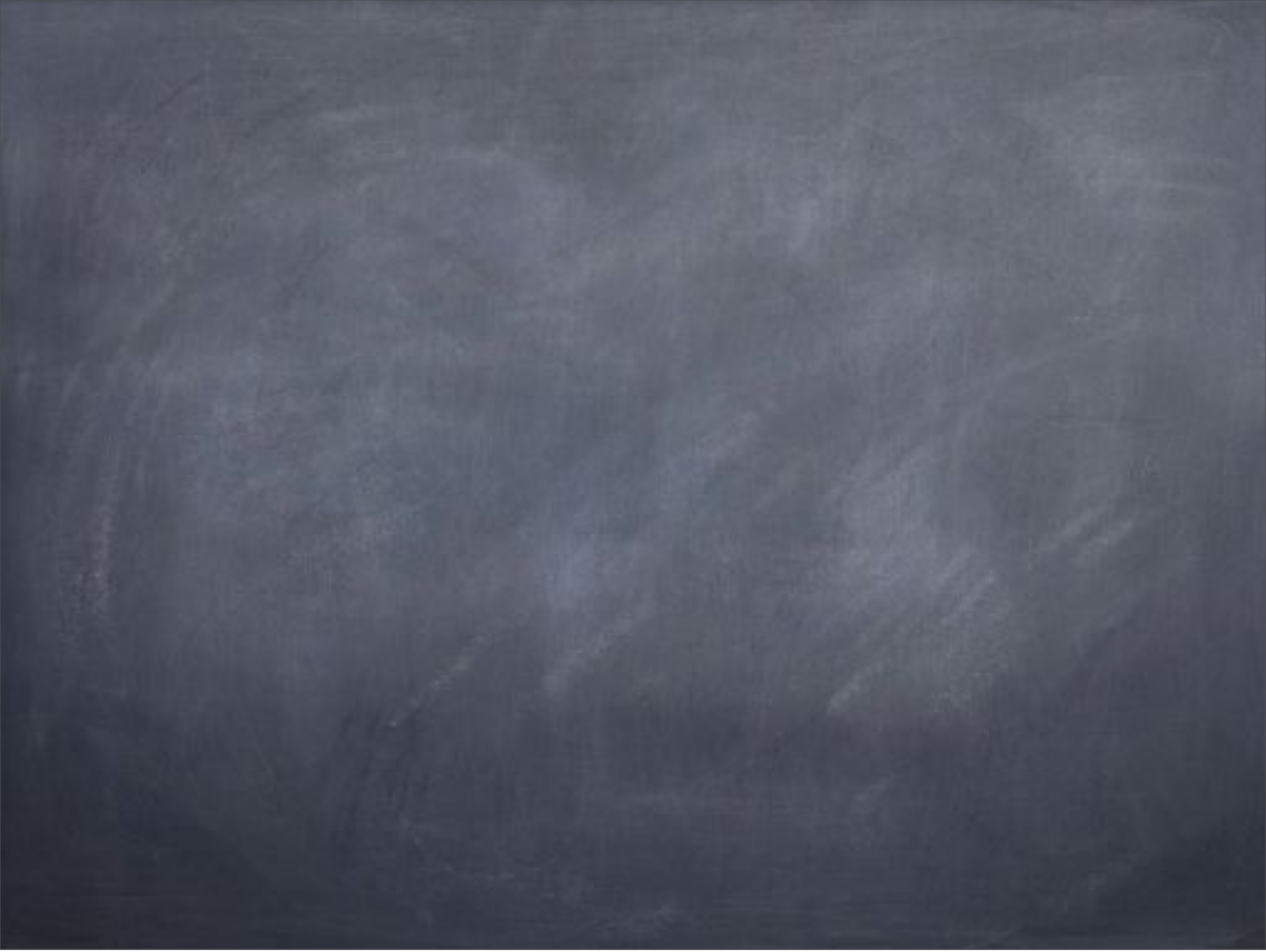
Let's do it!

- Create appschema test
- Use pgTAP
- Run test with pg_prove
- Make it fail
- Make it pass!
- Commit/Push
- https://github.com/theory/agile-flipr.git

Let's talk about...


Flipr
antisocial network

TDD



"The act of writing a unit test is more an act of design than of verification. It is also more an act of documentation than of verification. The act of writing a unit test closes a remarkable number of feedback loops, the least of which is the one pertaining to the verification of function."

—Robert C. Martin


Flipr
antisocial network

TDD is an act of design.


Flipr
antisocial network

TDD is an act of documentation.


Flipr
antisocial network

Okay.


Flipr
antisocial network

But...

Database Design

Database Design

- Specify requirements

Database Design

- Specify requirements
- Implement schema design

Database Design

- Specify requirements
- Implement schema design
- Program applications

Database Design

- Specify requirements
- Implement schema design
- Program applications
- QA

Database Design

- Specify requirements
- Implement schema design
- Program applications
- QA

Database Design

- Specify requirements
- Implement schema design
- Program applications
- QA

Pricy to fix

Database Design

- Specify requirements
- Implement schema design
- Program applications
- QA
- Never mind bureaucracy of DBA department

Why TDDD

Why TDDD

- Ensure data quality

Why TDDD

- Ensure data quality
  - Data is a core asset

Why TDDD

- Ensure data quality
  - Data is a core asset
- Validate business rules

Why TDDD

- Ensure data quality
  - Data is a core asset
- Validate business rules
  - Stored procedures

Why TDDD

- Ensure data quality
  - Data is a core asset
- Validate business rules
  - Stored procedures
  - Triggers

Why TDDD

- Ensure data quality
  - Data is a core asset
- Validate business rules
  - Stored procedures
  - Triggers
  - Views

Why TDDD

Why TDDD

- Identify defects early

Why TDDD

- Identify defects early
  - ...and often!

Why TDDD

- Identify defects early
  - ...and often!
- Create design iteratively

Why TDDD

- Identify defects early
  - ...and often!
- Create design iteratively
  - Evolutionarily

Why TDDD

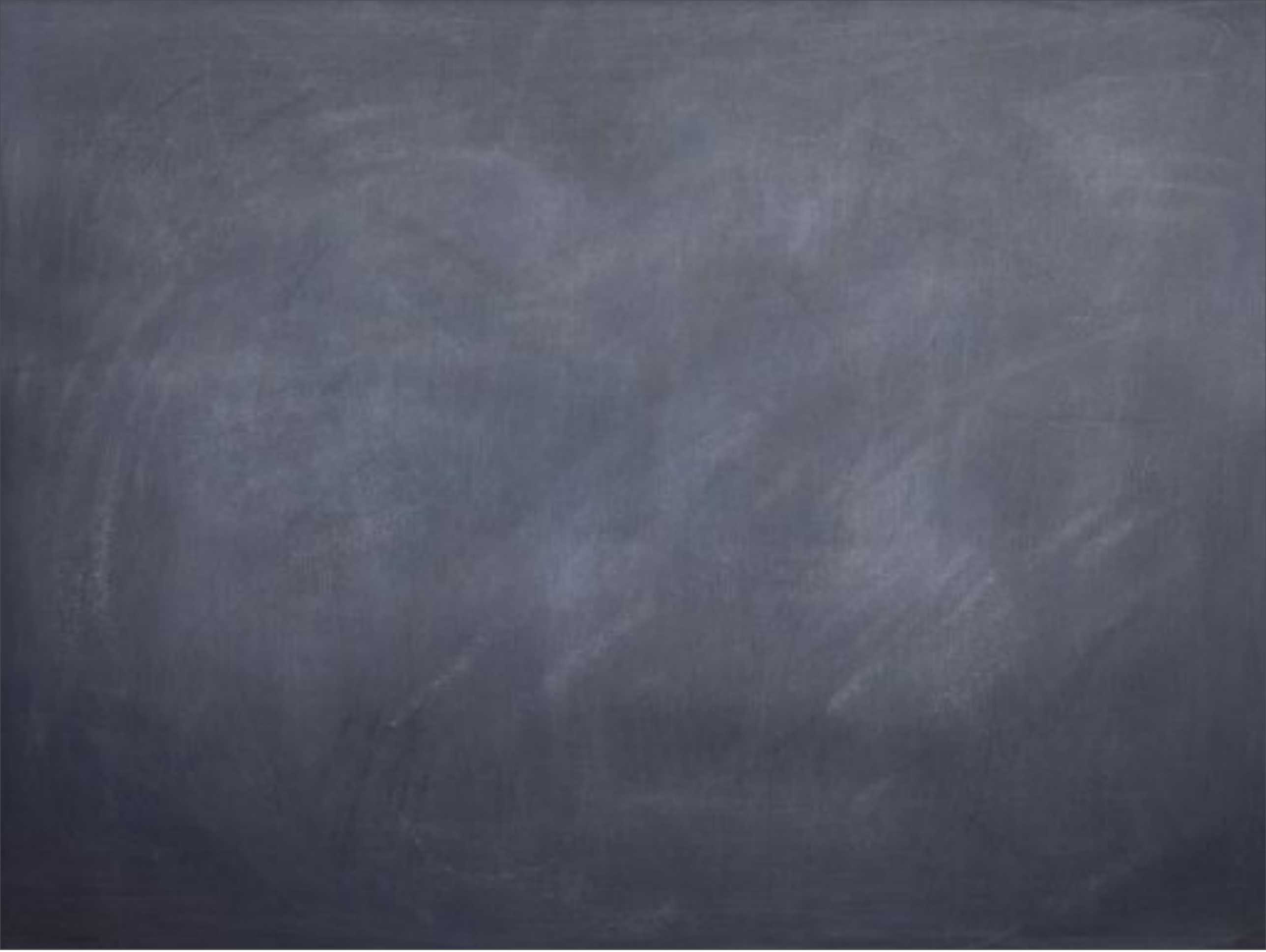
- Identify defects early
  - ...and often!
- Create design iteratively
  - Evolutionarily
- Validate refactorings

Why TDDD

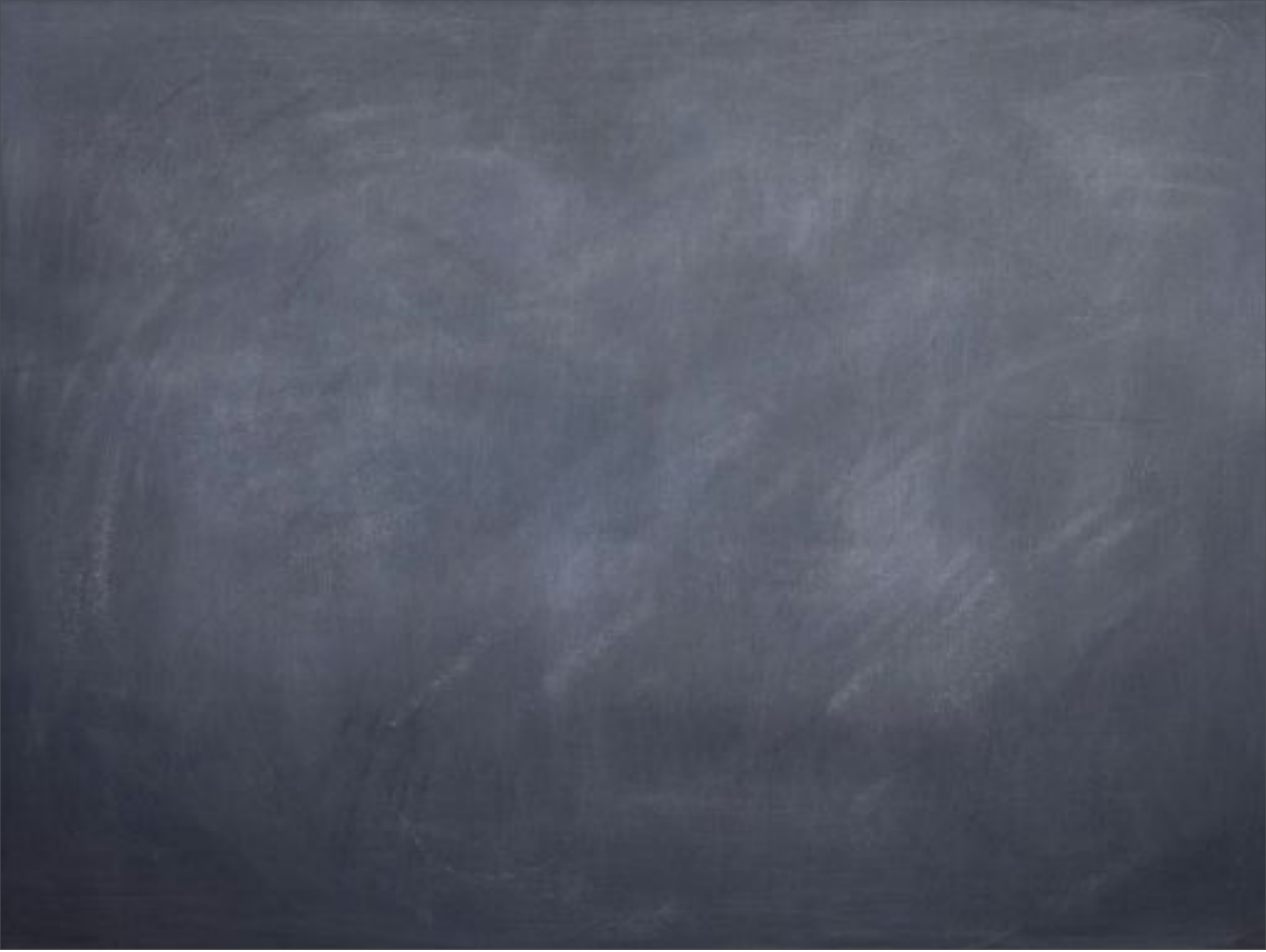
- Identify defects early
  - ...and often!
- Create design iteratively
  - Evolutionarily
- Validate refactorings
  - Make sure nothing breaks

Three Questions for Database Professionals

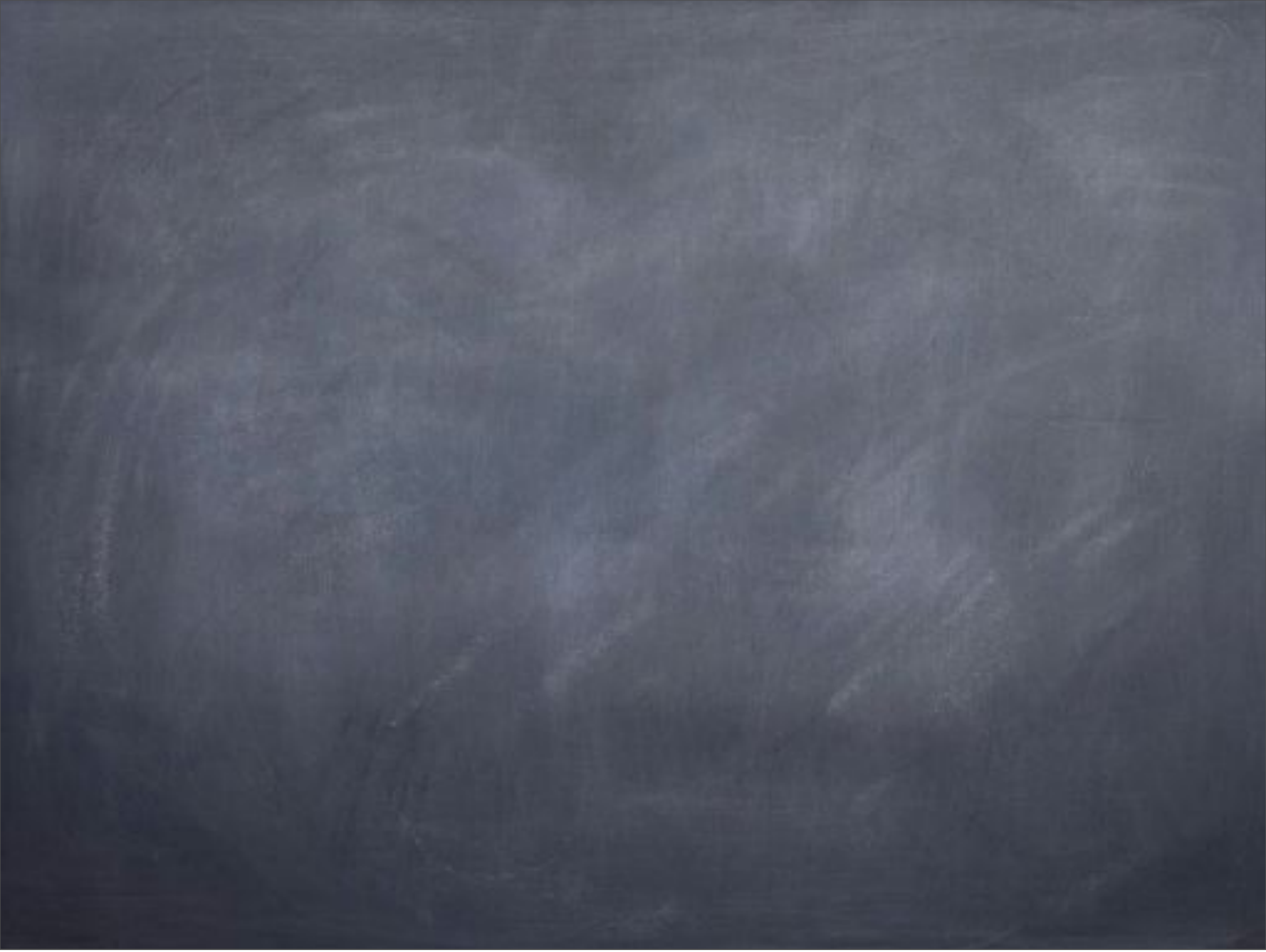
from
Scott Ambler



"If you're implementing code in the DB in the form of stored procedures, triggers... shouldn't you test that code to the same level that you test your app code?"



"Think of all the data quality problems you've run into over the years. Wouldn't it have been nice if someone had originally tested and discovered those problems before you did?"



"Wouldn't it be nice to have a test suite to run so that you could determine how (and if) the DB actually works?"


Flipr
antisocial network

Okay.


Flipr
antisocial network

How?

TDD How

TDD How

- Ideally separate from app tests

TDD How

- Ideally separate from app tests
  - May be many apps

TDD How

- Ideally separate from app tests
  - May be many apps
  - DB should present interface to all

TDD How

- Ideally separate from app tests
  - May be many apps
  - DB should present interface to all
  - Apps may use different permissions

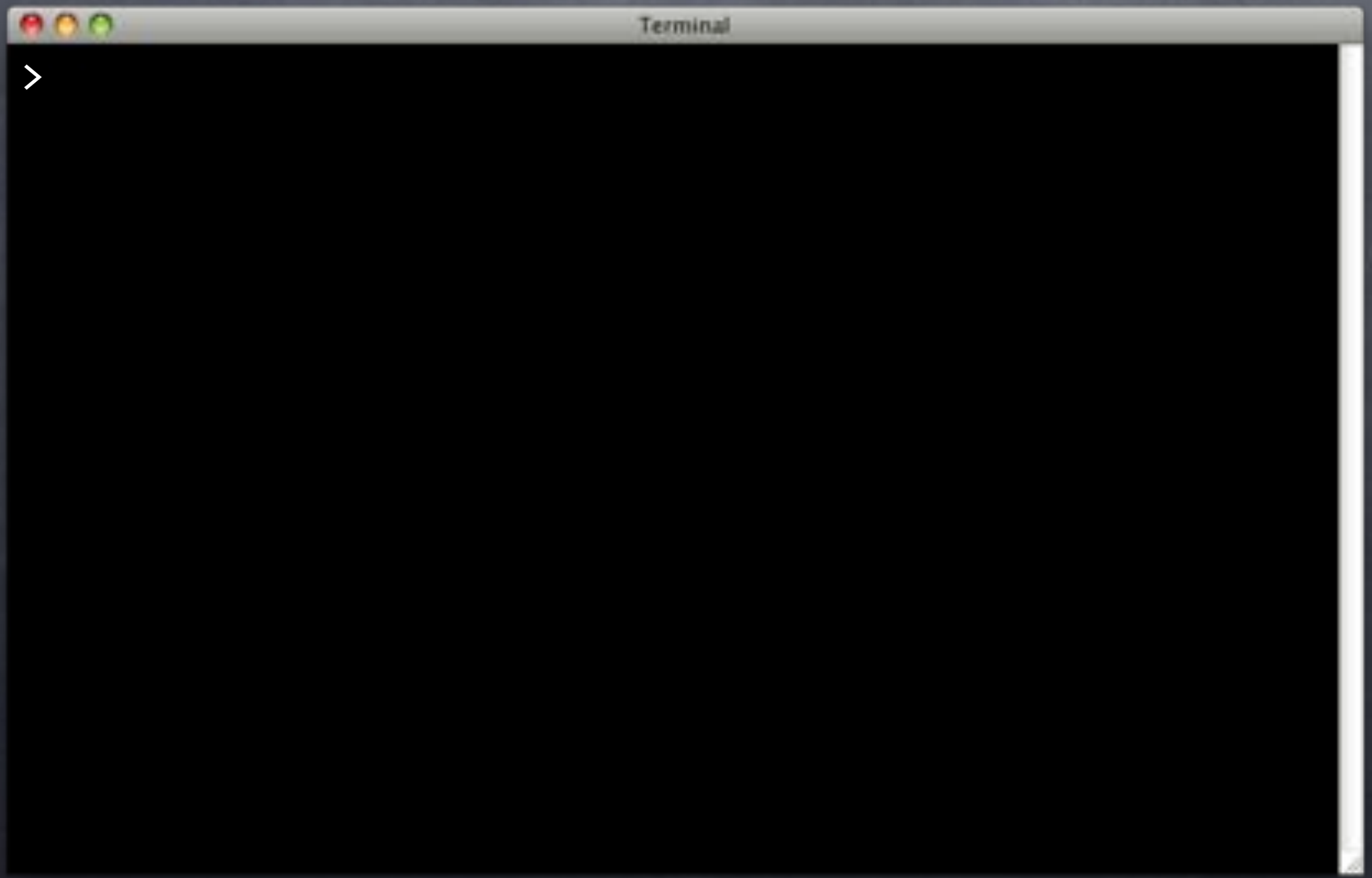
TDD How

- Ideally separate from app tests
  - May be many apps
  - DB should present interface to all
  - Apps may use different permissions
- Ideally use DB test Framework

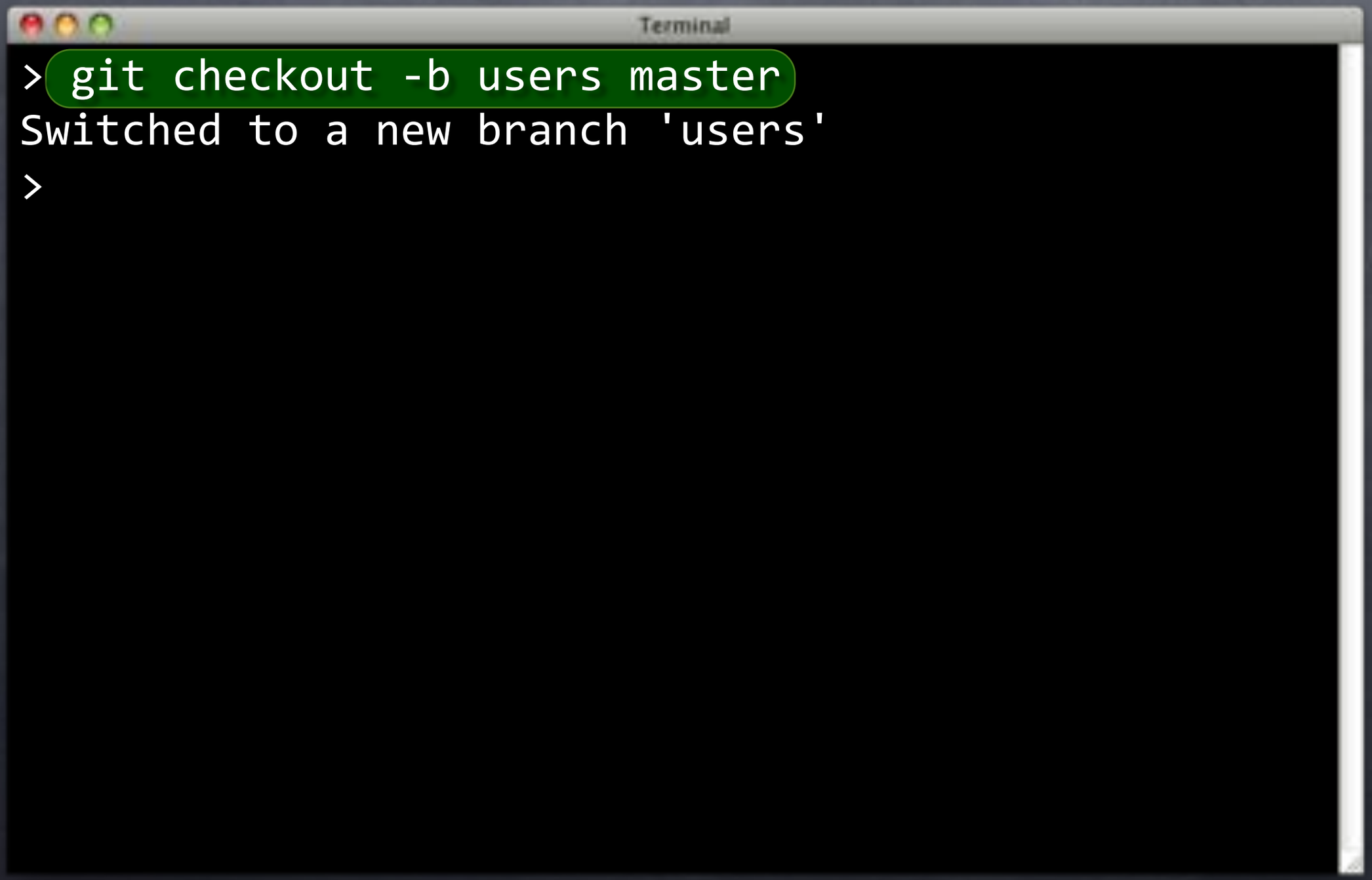
TDD How

- Ideally separate from app tests
  - May be many apps
  - DB should present interface to all
  - Apps may use different permissions
- Ideally use DB test Framework
- Like...pgTAP!

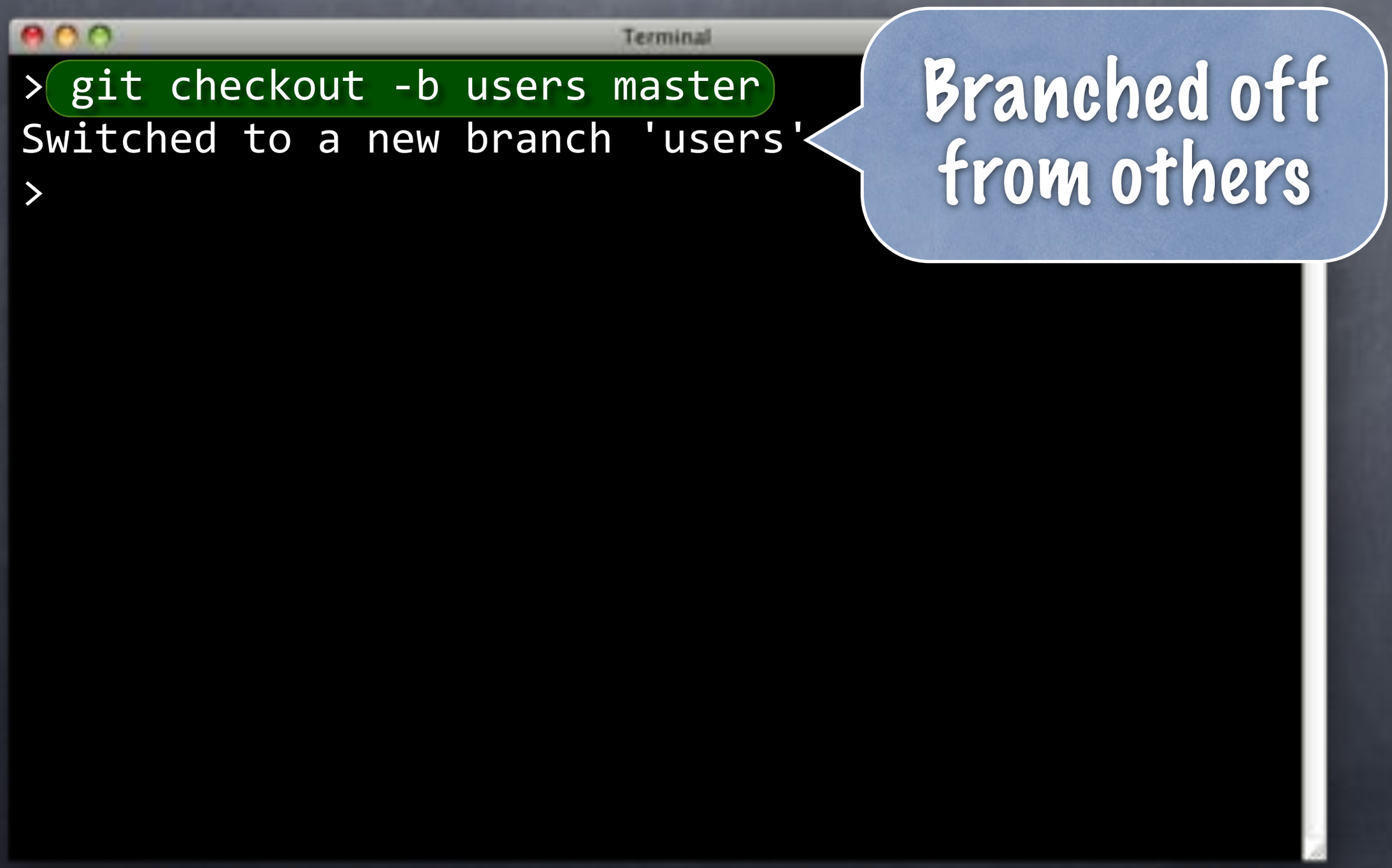
Branching Out



Branching Out



Branching Out



Branching Out

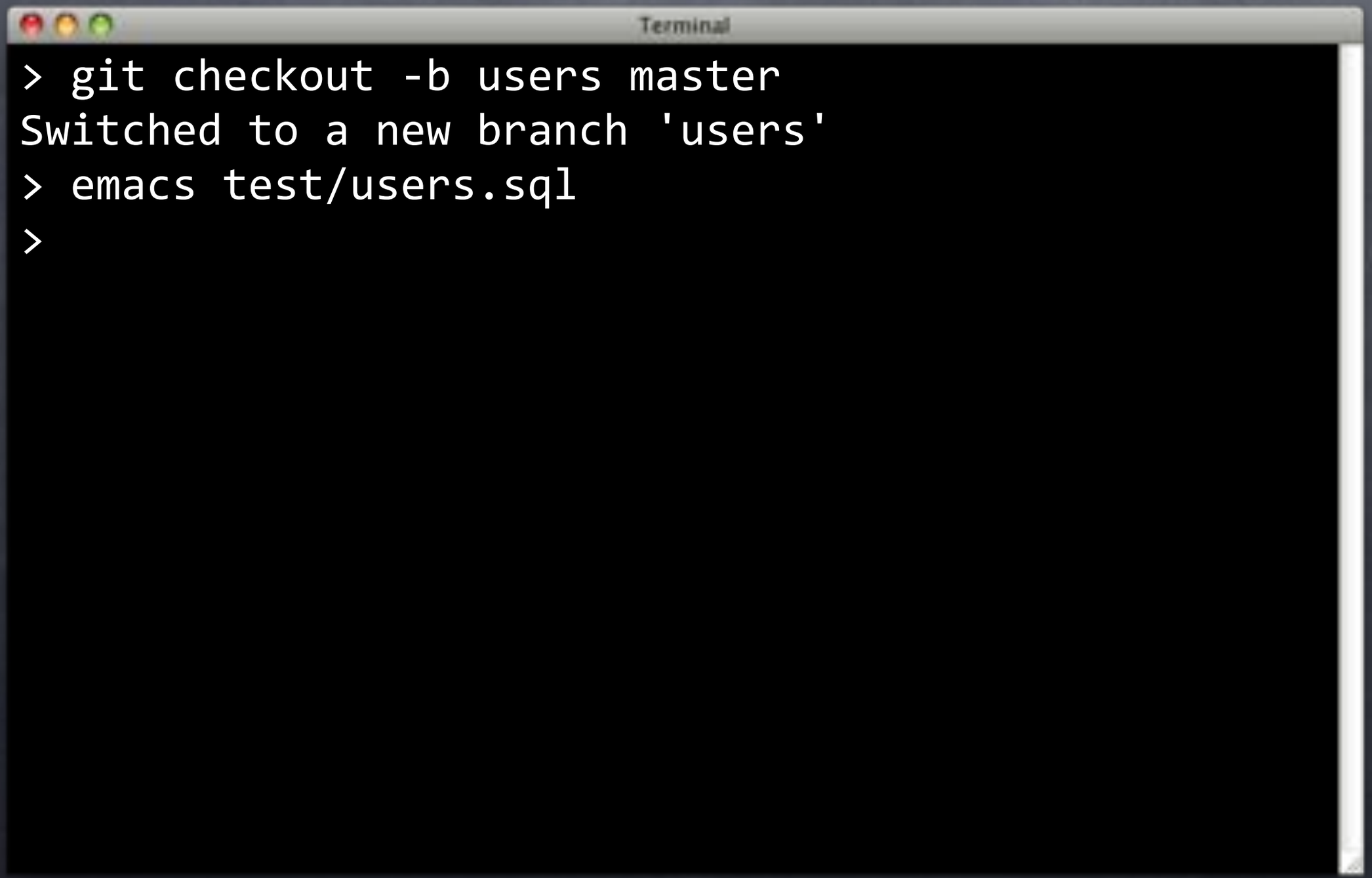


Table For One

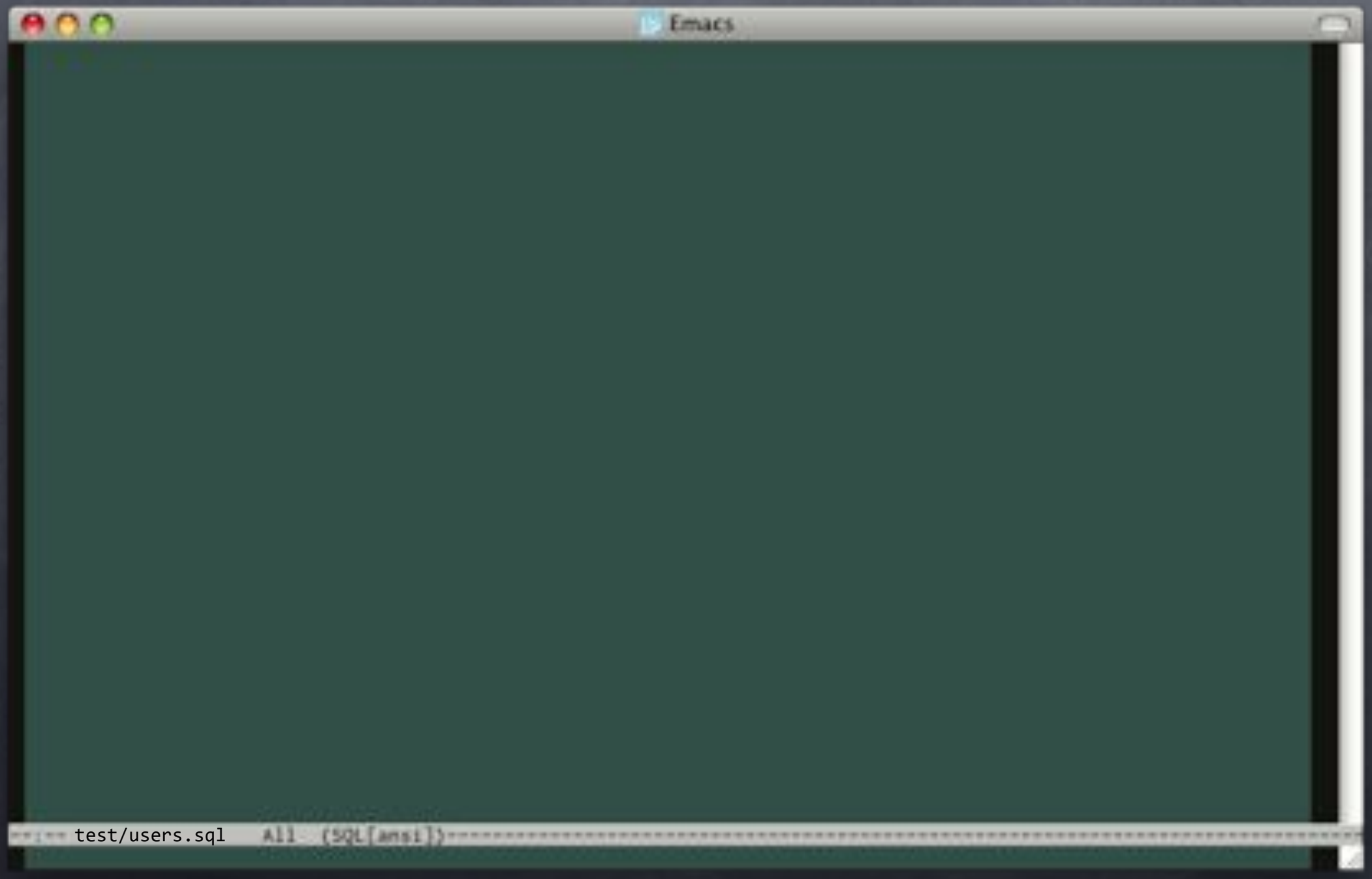


Table For One

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;
SELECT no_plan();
SET search_path TO flipr,public;

SELECT has_table( 'users' );

SELECT finish();
ROLLBACK;
```

-:**- test/users.sql All (SQL[ansi])

Table For One

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;
SELECT no_plan();
SET search_path TO flipr,public;

SELECT has_table( 'users' );

SELECT finish();
ROLLBACK;
```

-:**- test/users.sql   All (SQL[ansi])

Table For One

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;
SELECT no_plan();
SET search_path TO flipr,public;

SELECT has_table( 'users' );

SELECT finish();
ROLLBACK;
```

-:**- test/users.sql All (SQL[ansi])

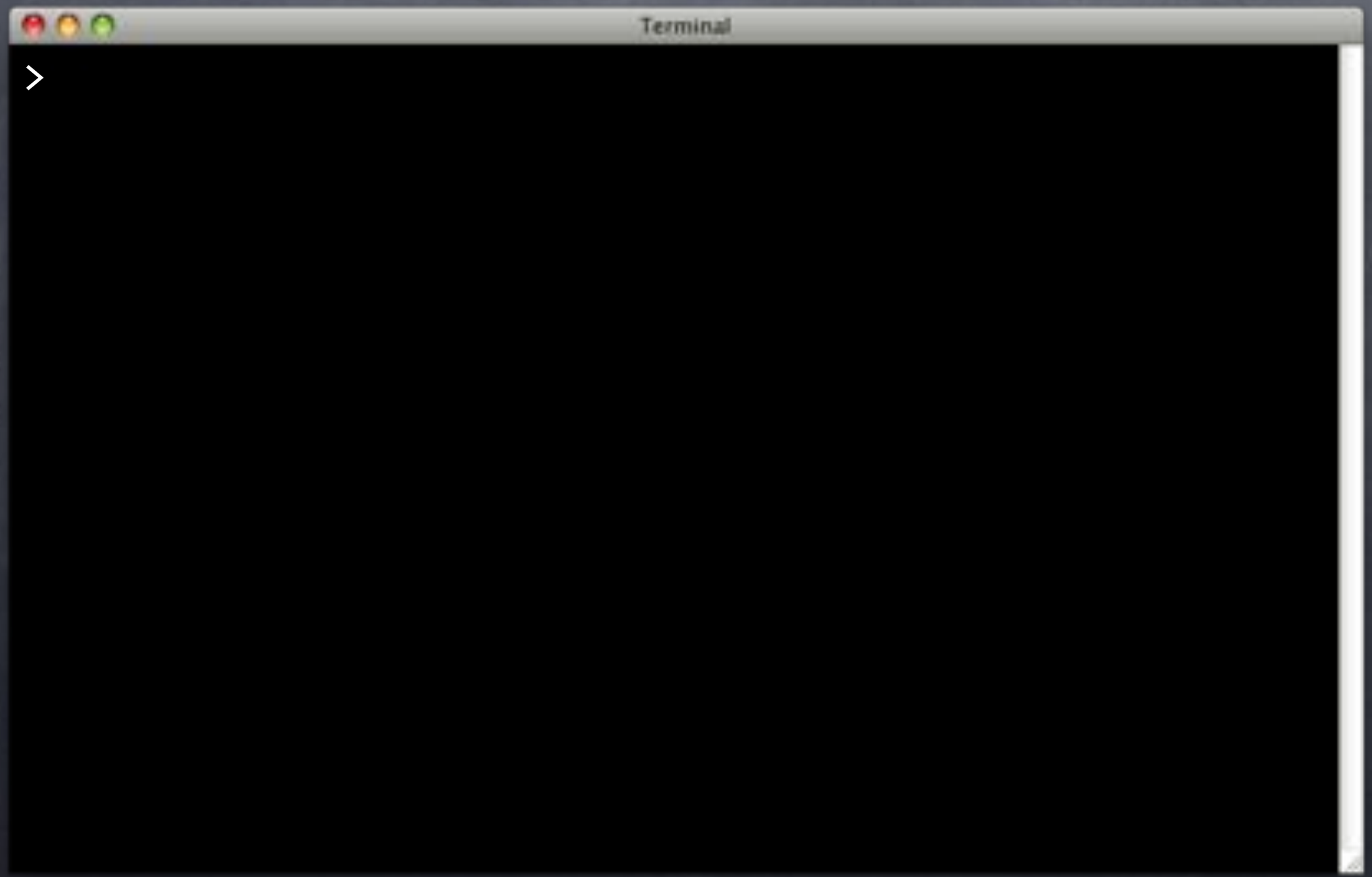
Table For One

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;
SELECT no_plan();
SET search_path TO flipr,public;

SELECT has_table( 'users' );

SELECT finish();
ROLLBACK;
```

-:--- test/users.sql All (SQL[ansi])

Run 'Em



Run 'Em

Terminal

```
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
not ok 1 - Table users should exist
# Failed test 1: "Table users should exist"
1..1
# Looks like you failed 1 test of 1
Failed 1/1 subtests

Test Summary Report
-------------------
test/users.sql (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs
Result: FAIL
>
```

Run 'Em

Terminal

```
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
not ok 1 - Table users should exist
# Failed test 1: "Table users should exist"
1..1
# Looks like you failed 1 test of 1
Failed 1/1 subtests

Test Summary Report
-------------------
test/users.sql (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1,  0 wallclock secs
Result: FAIL
>
```

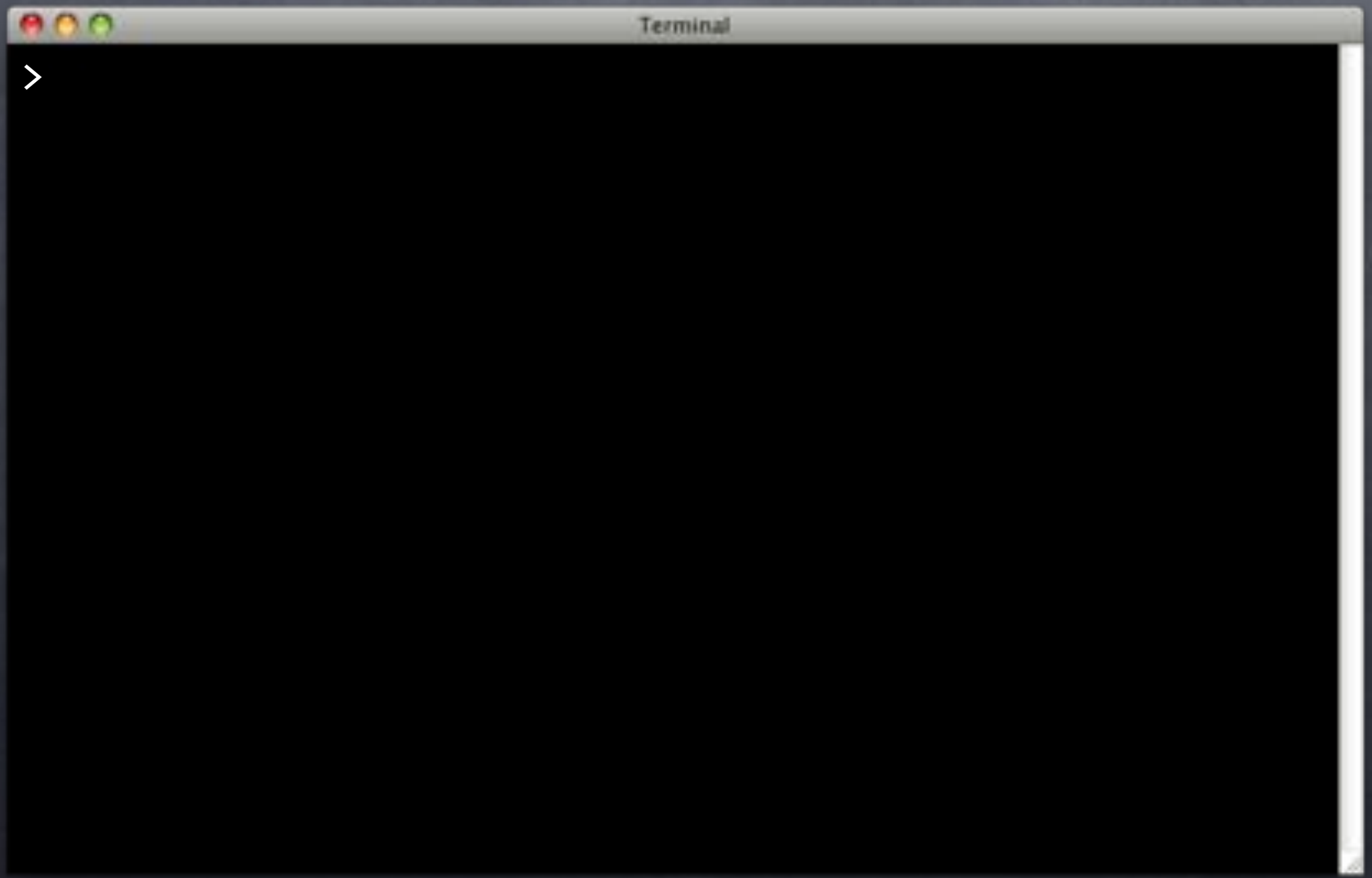
Run 'Em

Terminal

```
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
not ok 1 - Table users should exist
# Failed test 1: "Table users should exist"
1..1
# Looks like you failed 1 test of 1
Failed 1/1 subtests

Test Summary Report
-------------------
test/users.sql (Wstat: 0 Tests: 1 Failed: 1)
  Failed test:  1
Files=1, Tests=1
Result: FAIL
>
```

As expected.

Sqitch Dependencies!



Sqitch Dependencies!

Terminal

```
> sqitch add users --requires appschema \
  -n 'Creates table to track our users.'
Created deploy/users.sql
Created revert/users.sql
Created verify/users.sql
Added "users [appschema]" to sqitch.plan
>
```

Sqitch Dependencies!

Terminal

```
> sqitch add users --requires appschema \
  -n 'Creates table to track our users.'
Created deploy/users.sql
Created revert/users.sql
Created verify/users.sql
Added "users [appschema]" to sqitch.plan
>
```

Sqitch Dependencies!

Terminal

```
> sqitch add users --requires appschema \
  -n 'Creates table to track our users.'
Created deploy/users.sql
Created revert/users.sql
Created verify/users.sql
Added "users [appschema]" to sqitch.plan
> emacs deploy/users.sql
>
```

deploy/users.sql

Emacs

```
-- Deploy users
-- requires: appschema

BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

-:** deploy/users.sq All (SQL[ansi])

deploy/users.sql

Emacs

```
-- Deploy users
-- requires: appschema

BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

-:** deploy/users.sq All (SQL[ansi])

deploy/users.sql

Emacs

```
-- Deploy users
-- requires: appschema

BEGIN;

SET client_min_messages = 'warning';
CREATE TABLE flipr.users (
    nickname  TEXT
);

COMMIT;
```

-:** deploy/users.sq All (SQL[ansi])

deploy/users.sql

Emacs

```
-- Deploy users
-- requires: appschema

BEGIN;

SET client_min_messages = 'warning';
CREATE TABLE flipr.users (
    nickname  TEXT
);

COMMIT;
```

Bare Minimum

-:**- deploy/users.sq All (SQL[ansi])

Verily, Users

Terminal

```
> sqitch add users --requires appschema \
  -n 'Creates table to track our users.'
Created deploy/users.sql
Created revert/users.sql
Created verify/users.sql
Added "users [appschema]" to sqitch.plan
> emacs deploy/users.sql
>
```

Verily, Users

Terminal

```
> sqitch add users --requires appschema \
  -n 'Creates table to track our users.'
Created deploy/users.sql
Created revert/users.sql
Created verify/users.sql
Added "users [appschema]" to sqitch.plan
> emacs deploy/users.sql
> emacs verify/users.sql
```

verify/users.sql

Emacs

```
-- Verify users

BEGIN;

-- XXX Add verifications here.

ROLLBACK;
```

```
-:**-  verify/users.sq   All   (SQL[ansi])----------------------------------------------------------------------------
```

verify/users.sql

Emacs

```
-- Verify users

BEGIN;

SELECT nickname
  FROM flipr.users
 WHERE FALSE;

ROLLBACK;
```

-:** verify/users.sq All (SQL[ansi])

Unusered

Terminal

```
> sqitch add users --requires appschema \
  -n 'Creates table to track our users.'
Created deploy/users.sql
Created revert/users.sql
Created verify/users.sql
Added "users [appschema]" to sqitch.plan
> emacs deploy/users.sql
> emacs verify/users.sql
>
```

Unusered

Terminal

```
> sqitch add users --requires appschema \
  -n 'Creates table to track our users.'
Created deploy/users.sql
Created revert/users.sql
Created verify/users.sql
Added "users [appschema]" to sqitch.plan
> emacs deploy/users.sql
> emacs verify/users.sql
> emacs revert/users.sql
```

revert/users.sql

Emacs

```
-- Revert users

BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

-:** revert/users.sq All (SQL[ansi])----------

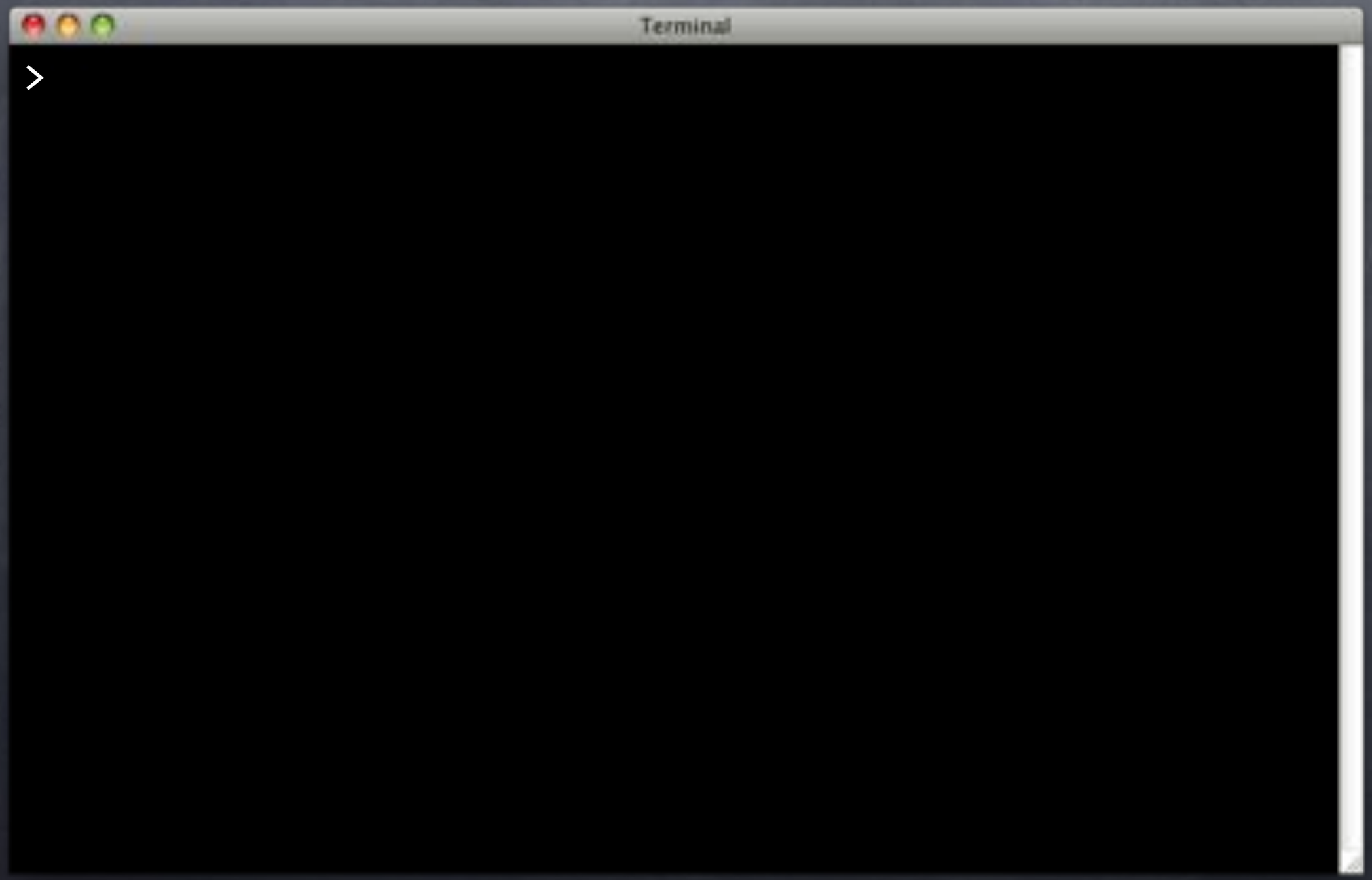
revert/users.sql

Emacs

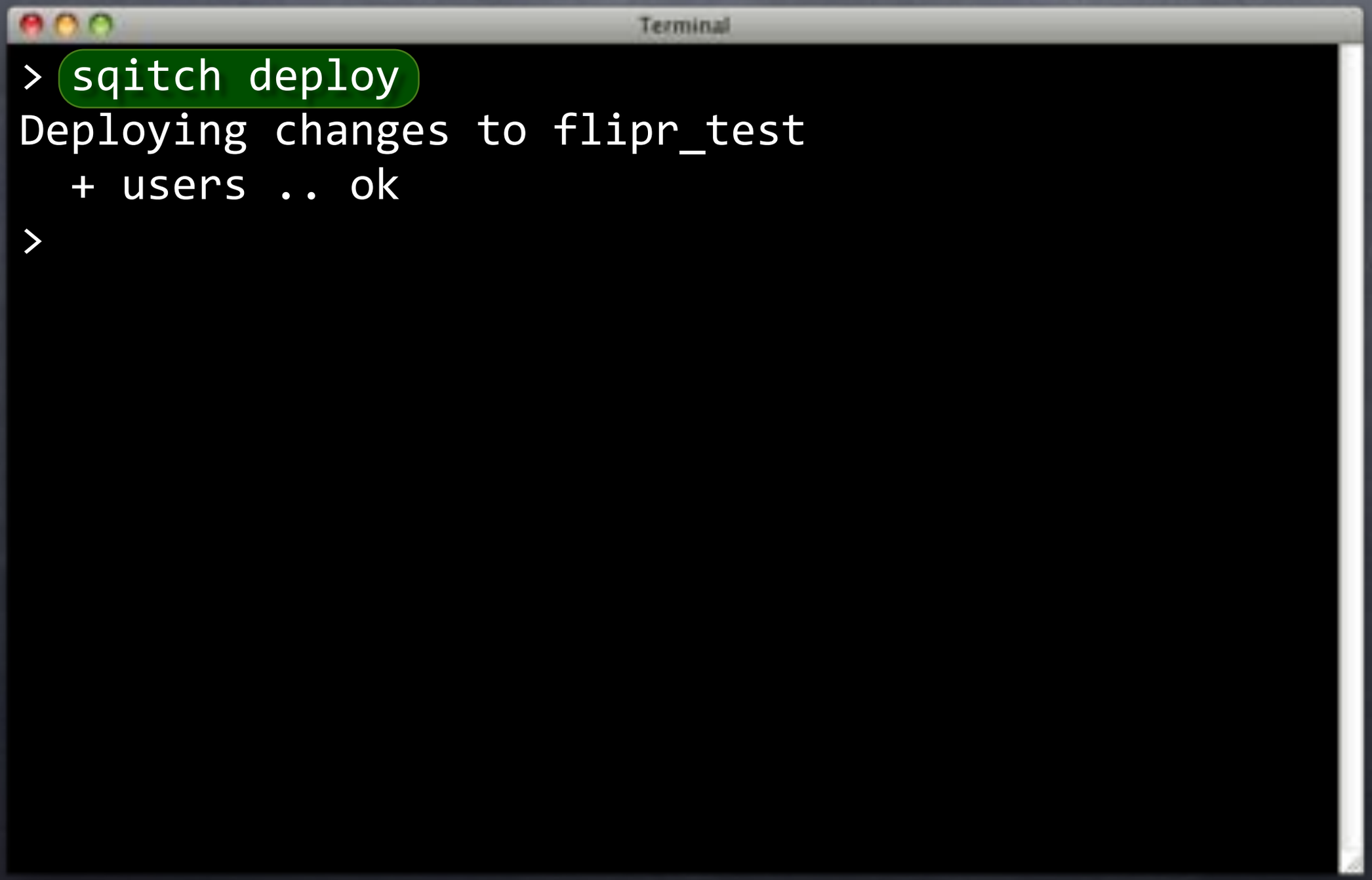
```
-- Revert users

BEGIN;

DROP TABLE flipr.users;

COMMIT;
```

-:**- revert/users.sq All (SQL[ansi])----------

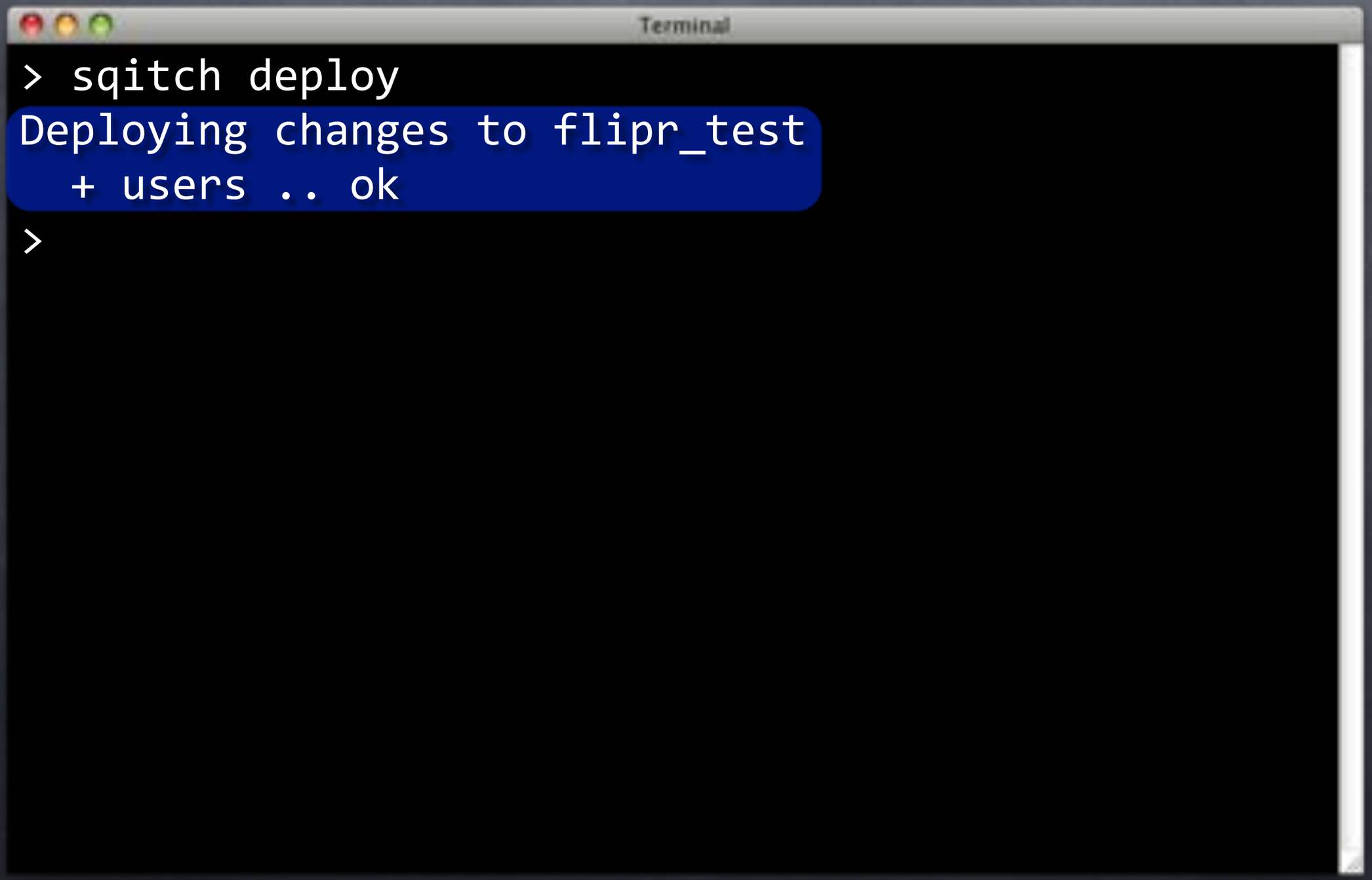
Make Users



Make Users


Terminal
> sqitch deploy
Deploying changes to flipr_test
  + users .. ok
>

Make Users


Terminal
> sqitch deploy
Deploying changes to flipr_test
  + users .. ok
>

Make Users

Terminal

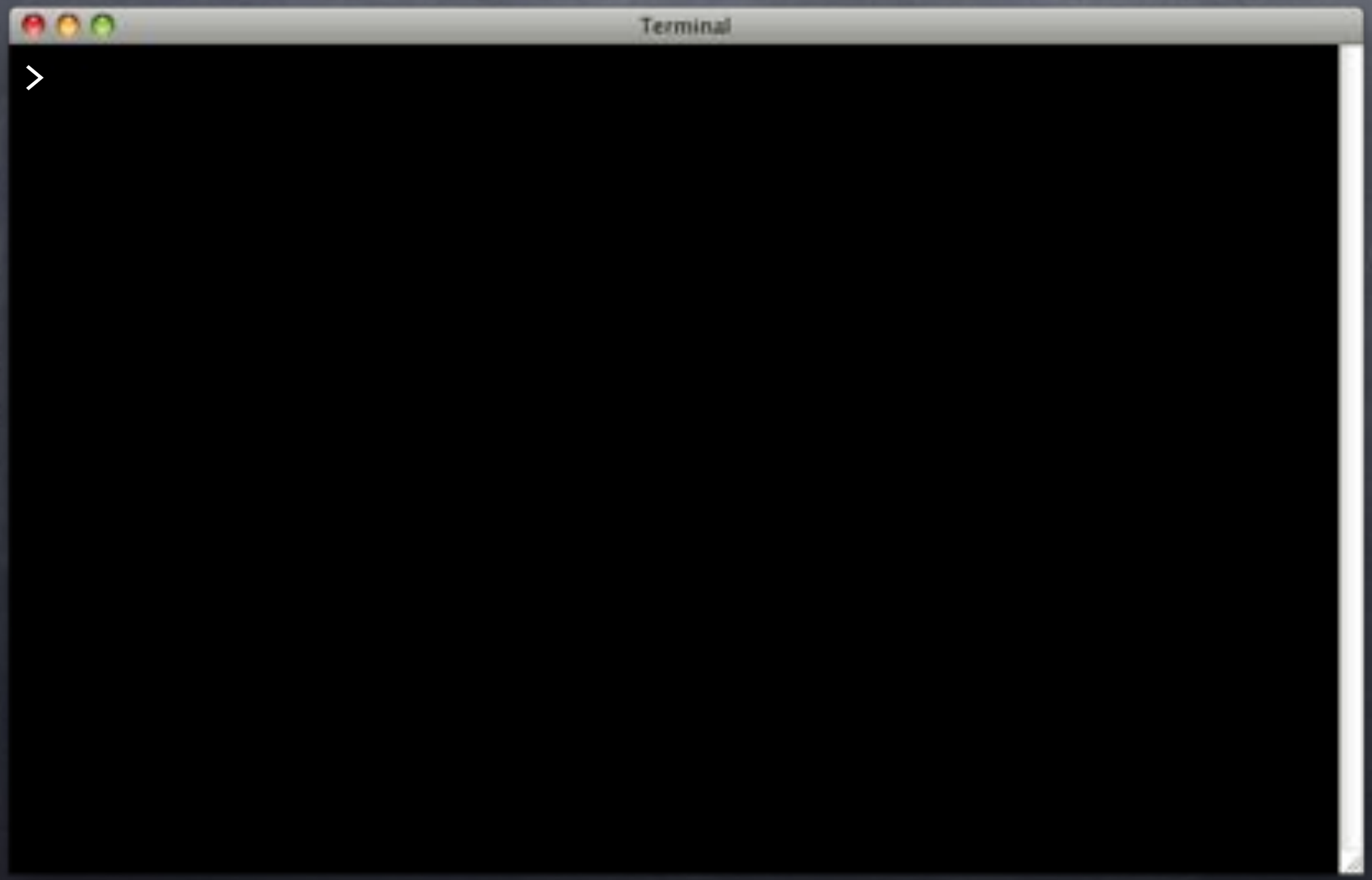
```
> sqitch deploy
Deploying changes to flipr_test
  + users .. ok
> psql -d flipr_test -c '\d flipr.users'
    Table "flipr.users"
  Column  | Type | Modifiers
----------+------+-----------
 nickname | text |
>
```

Make Users

Terminal

```
> sqitch deploy
Deploying changes to flipr_test
  + users .. ok
> psql -d flipr_test -c '\d flipr.users'
    Table "flipr.users"
  Column  | Type | Modifiers
----------+------+-----------
 nickname | text |
> sqitch verify
Verifying flipr_test
  * appschema .. ok
  * users ..... ok
Verify successful
>
```

Make Users



Make Users

Terminal

```
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
ok 1 - Table users should exist
1..1
ok
All tests successful.
Files=1, Tests=1,  1 wallclock secs
Result: PASS
>
```

Make Users

Terminal

```
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
ok 1 - Table users should exist
1..1
ok
All tests successful.
Files=1, Tests=1,  1 wallclock secs
Result: PASS
>
```

Make Users

Terminal

```
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
ok 1 - Table users should exist
1..1
ok
All tests successful.
Files=1, Tests=1,  1 wallclock secs
Result: PASS
>
```

Woohoo!

Make Users

Terminal

```
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
ok 1 - Table users should exist
1..1
ok
All tests successful.
Files=1, Tests=1,  1 wallclock secs
Result: PASS
> emacs test/users.sql
>
```

Columnist

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;
SELECT no_plan();
SET search_path TO flipr,public;

SELECT has_table( 'users' );

SELECT finish();
ROLLBACK;
```

-:**- test/users.sql   All (SQL[ansi])

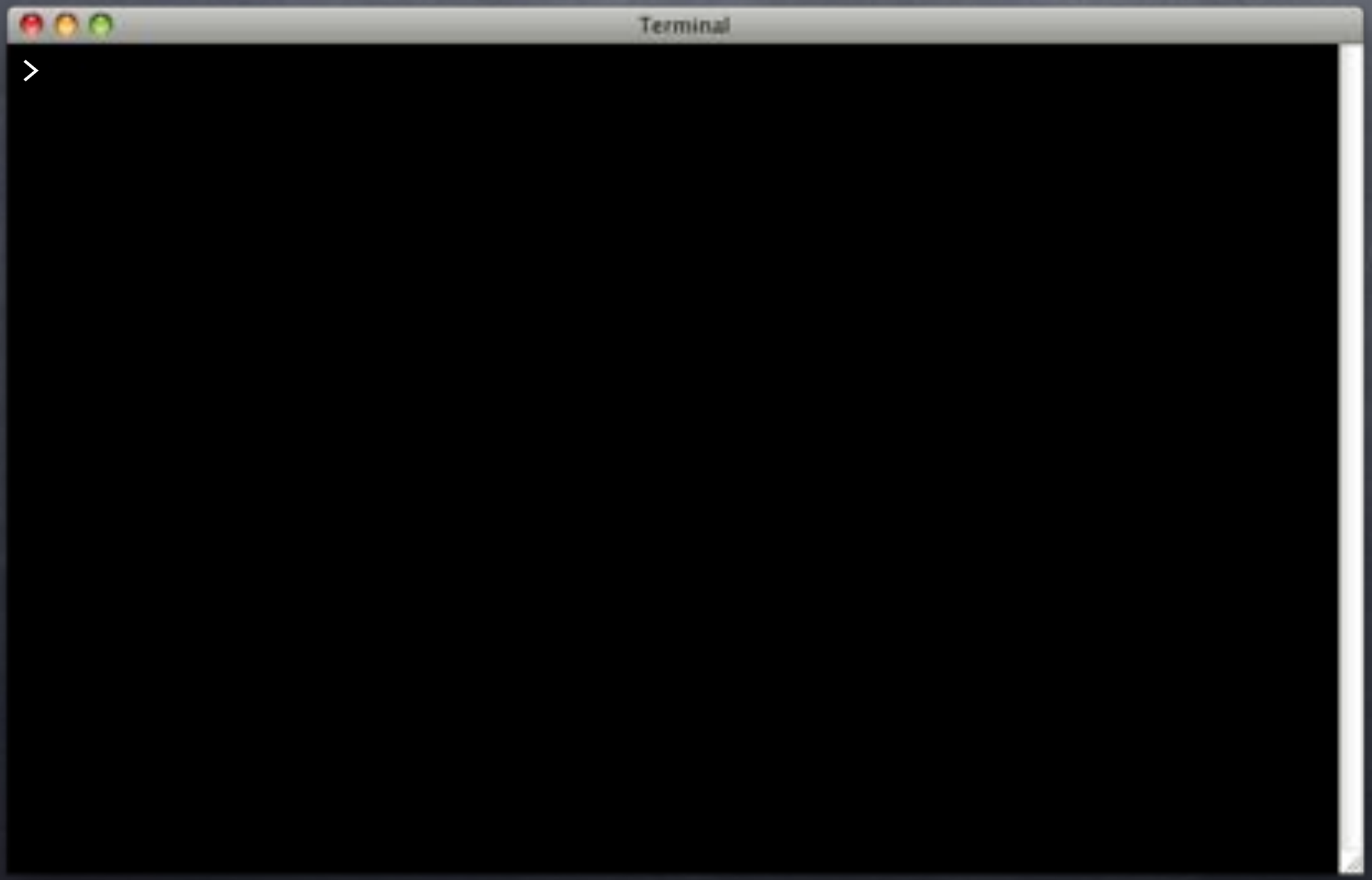
Columnist

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

BEGIN;
SELECT no_plan();
SET search_path TO flipr,public;

SELECT has_table( 'users' );
SELECT has_column( 'users', 'nickname'  );
SELECT has_column( 'users', 'password'  );
SELECT has_column( 'users', 'timestamp' );

SELECT finish();
ROLLBACK;
```

-:**- test/users.sql All (SQL[ansi])

Dead Again



Dead Again

Terminal

```
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
ok 1 - Table users should exist
ok 2 - Column users.nickname should exist
not ok 3 - Column users.password should exist
# Failed test 3: "Column users.password should exist"
not ok 4 - Column users."timestamp" should exist
# Failed test 4: "Column users."timestamp" should exist"
1..4
# Looks like you failed 2 tests of 4
Failed 2/4 subtests

Test Summary Report
-------------------
test/users.sql (Wstat: 0 Tests: 4 Failed: 2)
  Failed tests:  3-4
Files=1, Tests=4,  0 wallclock secs
Result: FAIL
```

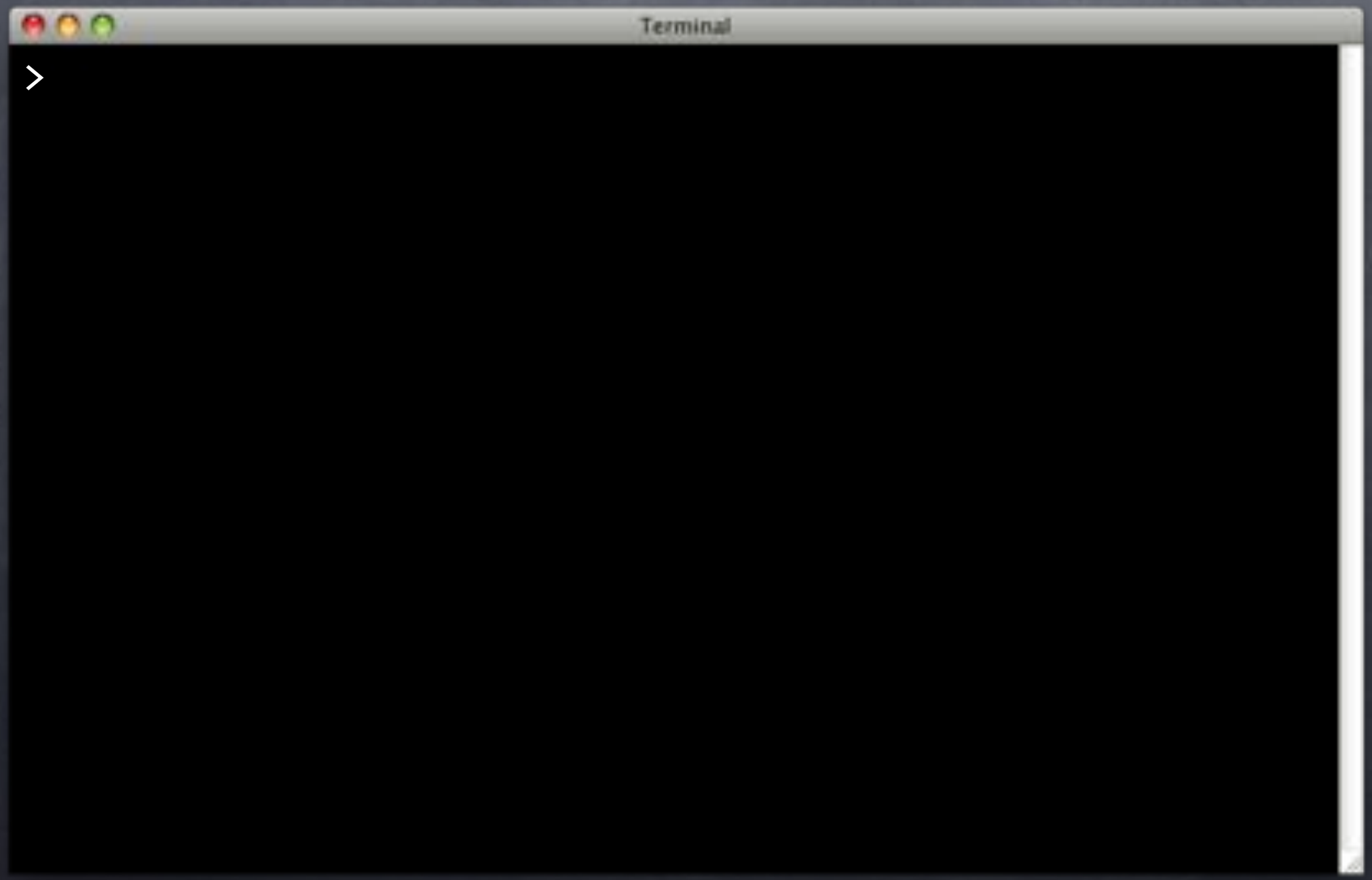
Dead Again

Terminal

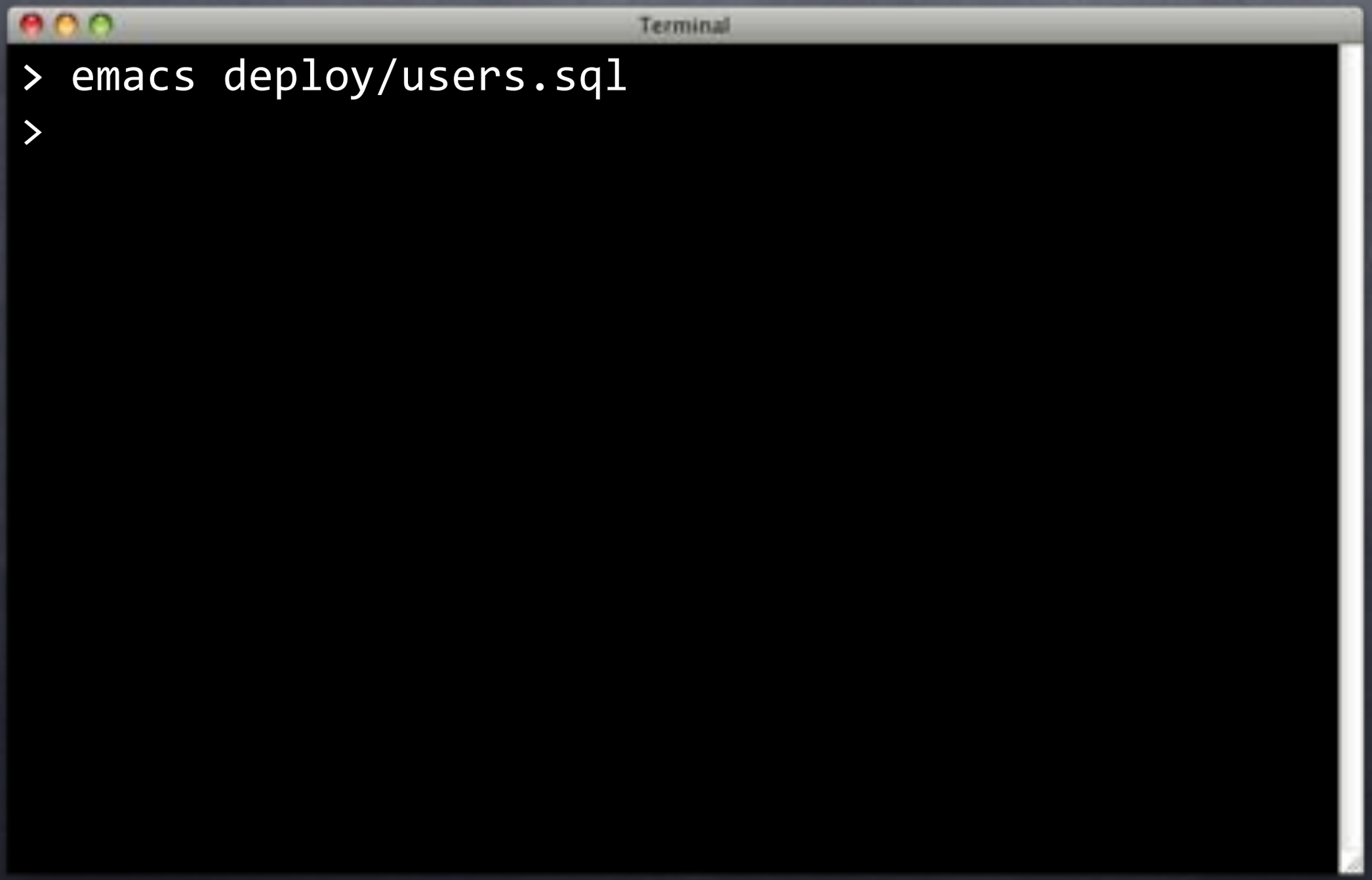
```
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
ok 1 - Table users should exist
ok 2 - Column users.nickname should exist
not ok 3 - Column users.password should exist
# Failed test 3: "Column users.password should exist"
not ok 4 - Column users."timestamp" should exist
# Failed test 4: "Column users."timestamp" should exist"
1..4
# Looks like you failed
Failed 2/4 subtests

Test Summary Report
-------------------
test/users.sql (Wstat: 0
  Failed tests:  3-4
Files=1, Tests=4,  0 wallclock secs
Result: FAIL
```

Guess we should add them.

MOAR Deploy



MOAR Deploy



deploy/users.sql

Emacs

```
-- Deploy users
-- requires: appschema

BEGIN;

SET client_min_messages = 'warning';
CREATE TABLE flipr.users (
    nickname  TEXT
);

COMMIT;
```

-:** deploy/users.sq All (SQL[ansi])

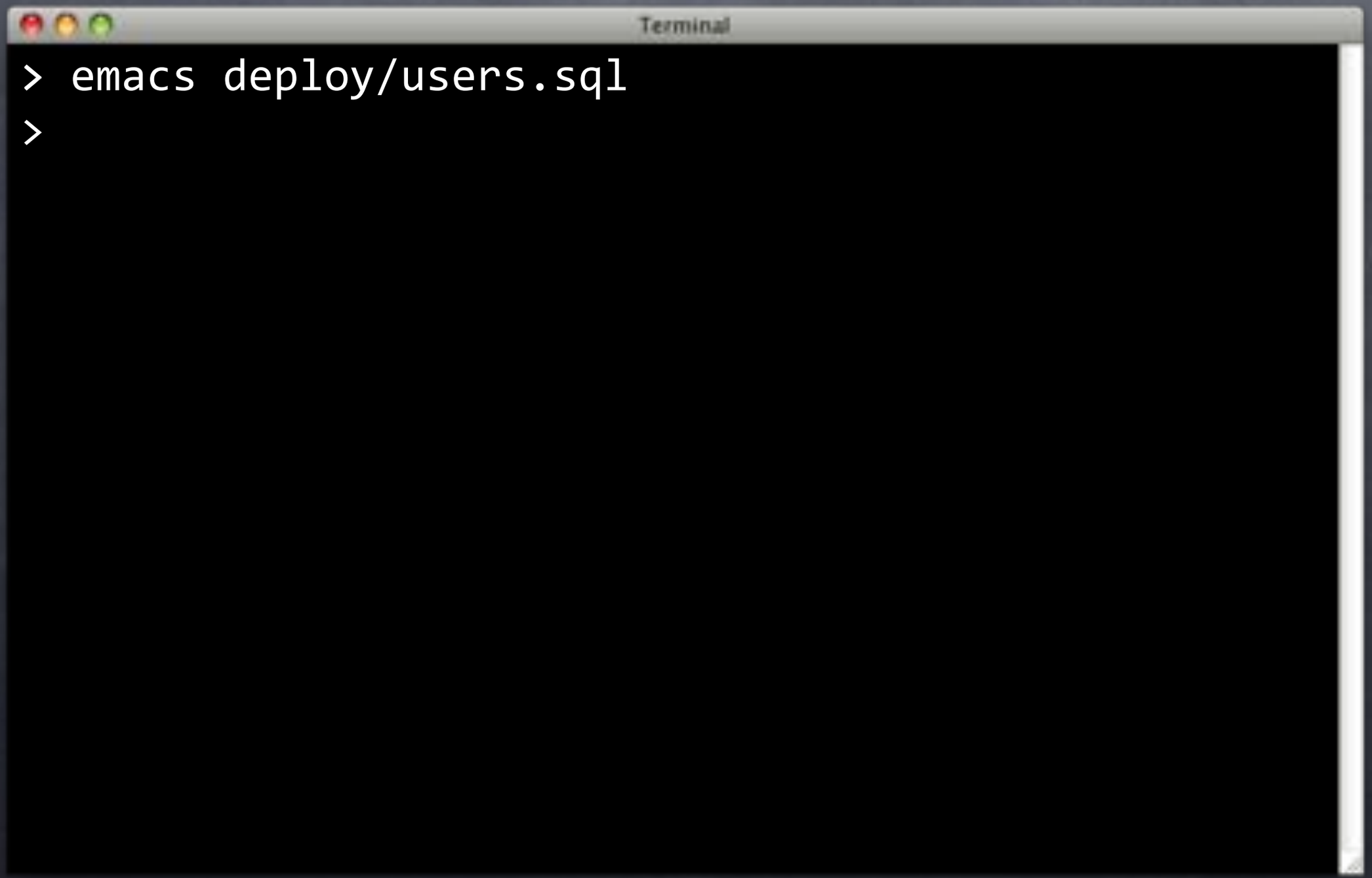
deploy/users.sql

Emacs

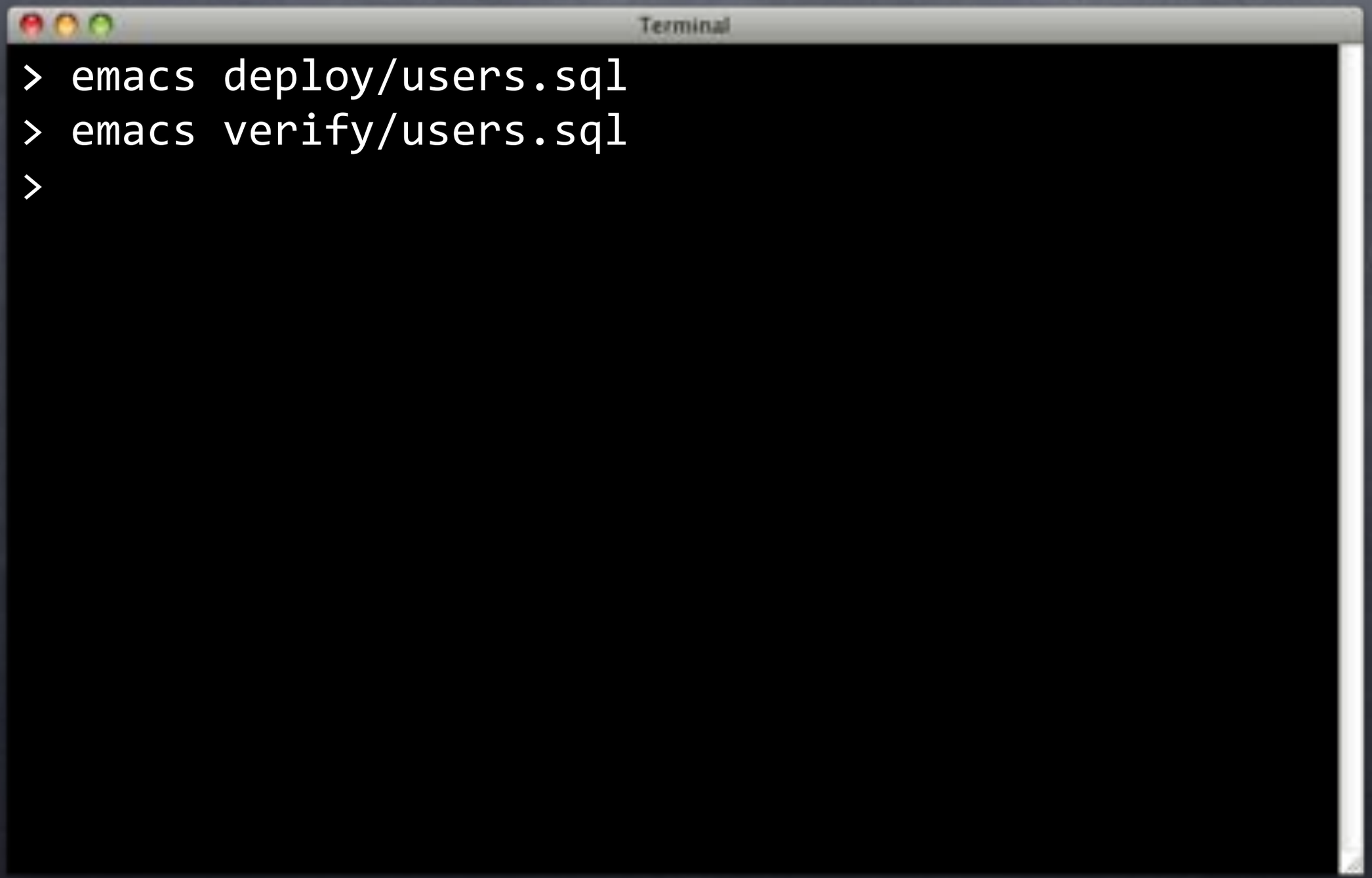
```
-- Deploy users
-- requires: appschema

BEGIN;

SET client_min_messages = 'warning';
CREATE TABLE flipr.users (
    nickname  TEXT,
    password  TEXT,
    timestamp TIMESTAMPTZ
);

COMMIT;
```

-:** deploy/users.sq All (SQL[ansi])

Update Verify



Update Verify



verify/users.sql

Emacs

```
-- Verify users

BEGIN;

SELECT nickname
  FROM flipr.users
 WHERE FALSE;

COMMIT;
```

-:** verify/users.sq All (SQL[ansi])----------

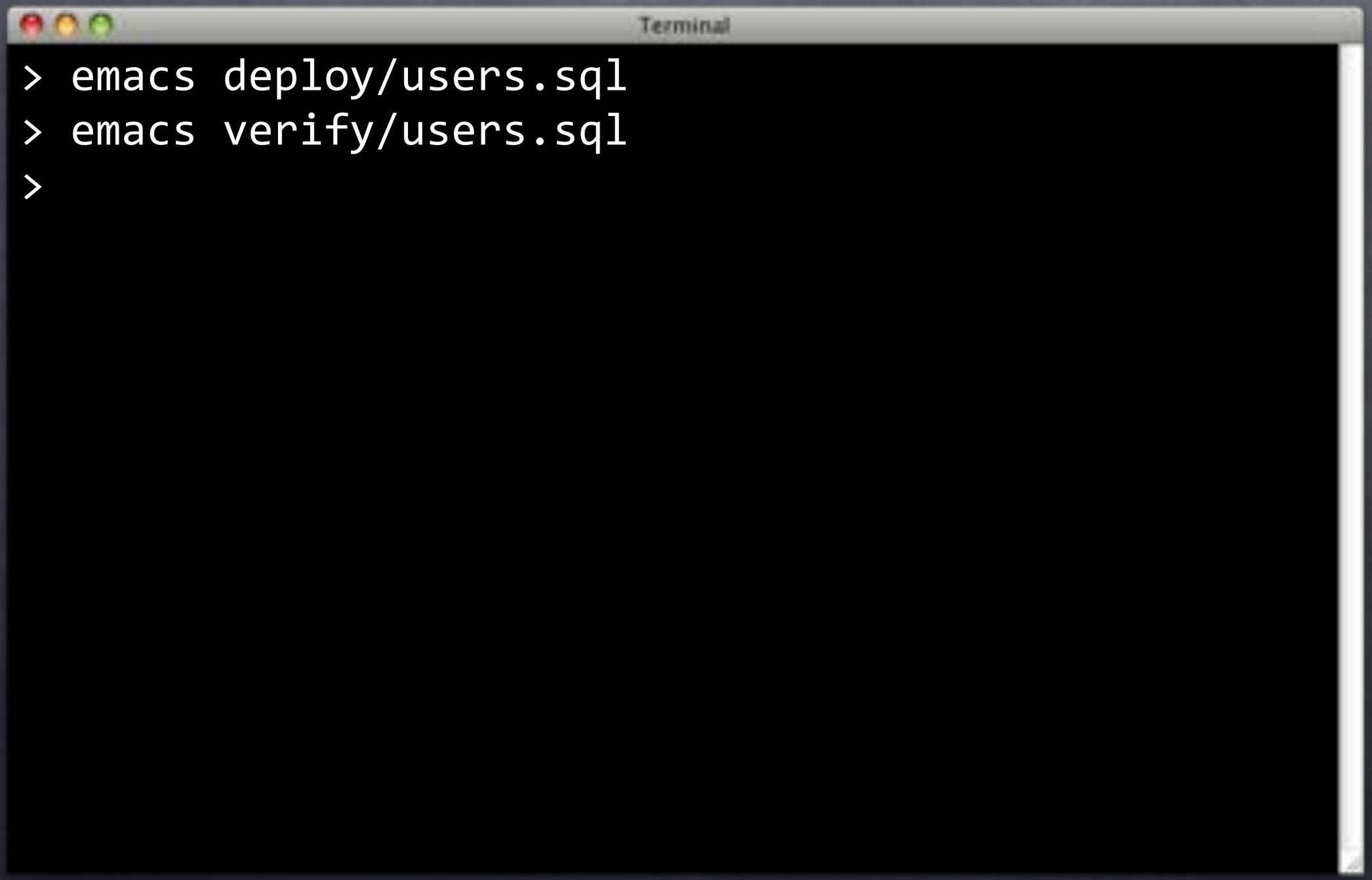
verify/users.sql

Emacs

```
-- Verify users

BEGIN;

SELECT nickname, password, timestamp
  FROM flipr.users
 WHERE FALSE;

COMMIT;
```

-:** verify/users.sq All (SQL[ansi])----------

Revert Overhead



Revert Overhead

Terminal

```
> emacs deploy/users.sql
> emacs verify/users.sql
> sqitch revert --to @HEAD^ -y
Reverting changes to appschema from flipr_test
  - users .. ok
>
```

Revert Overhead

Terminal

```
> emacs deploy/users.sql
> emacs verify/users.sql
> sqitch revert --to @HEAD^ -y
Reverting changes to appschema from flipr_test
  - users .. ok
>
```

Yes, really.

Revert Overhead

Terminal

```
> emacs deploy/users.sql
> emacs verify/users.sql
> sqitch revert --to @HEAD^ -y
Reverting changes to appschema from flipr_test
  - users .. ok
>
```

Revert Overhead

Terminal

```
> emacs deploy/users.sql
> emacs verify/users.sql
> sqitch revert --to @HEAD^ -y
Reverting changes to appschema from flipr_test
  - users .. ok
>
```

Remove

Revert Overhead

What's that?

```
> emacs deploy/users.sql
> emacs verify/users.sql
> sqitch revert --to @HEAD^ -y
Reverting changes to appschema from flipr_test
  - users .. ok
>
```

Sqitch Tags

Sqitch Tags

- Start with @

Sqitch Tags

- Start with @
- To distinguish from changes

Sqitch Tags

- Start with @
- To distinguish from changes
- Two symbolic tags:

Sqitch Tags

- Start with @
- To distinguish from changes
- Two symbolic tags:
  - @HEAD Last change

Sqitch Tags

- Start with @
- To distinguish from changes
- Two symbolic tags:
  - @HEAD Last change
  - @ROOT First change

Sqitch Tags

- Start with @
- To distinguish from changes
- Two symbolic tags:
  - @HEAD Last change
  - @ROOT First change
- Two modifiers:

Sqitch Tags

- Start with @
- To distinguish from changes
- Two symbolic tags:
  - @HEAD Last change
  - @ROOT First change
- Two modifiers:
  - ^ Previous change

Sqitch Tags

- Start with @
- To distinguish from changes
- Two symbolic tags:
  - @HEAD Last change
  - @ROOT First change
- Two modifiers:
  - ^ Previous change
  - ~ Following change

Specifying Changes

Specifying Changes

- users — Change named “users”

Specifying Changes

- users — Change named “users”
- @HEAD^ — Second to last change

Specifying Changes

- users — Change named “users”
- @HEAD^ — Second to last change
- users^ — Change before users

Specifying Changes

- users — Change named "users"
- @HEAD^ — Second to last change
- users^ — Change before users
- @ROOT~ — Second change

Specifying Changes

- users — Change named "users"
- @HEAD^ — Second to last change
- users^ — Change before users
- @ROOT~ — Second change
- appschema~ — Change after appschema

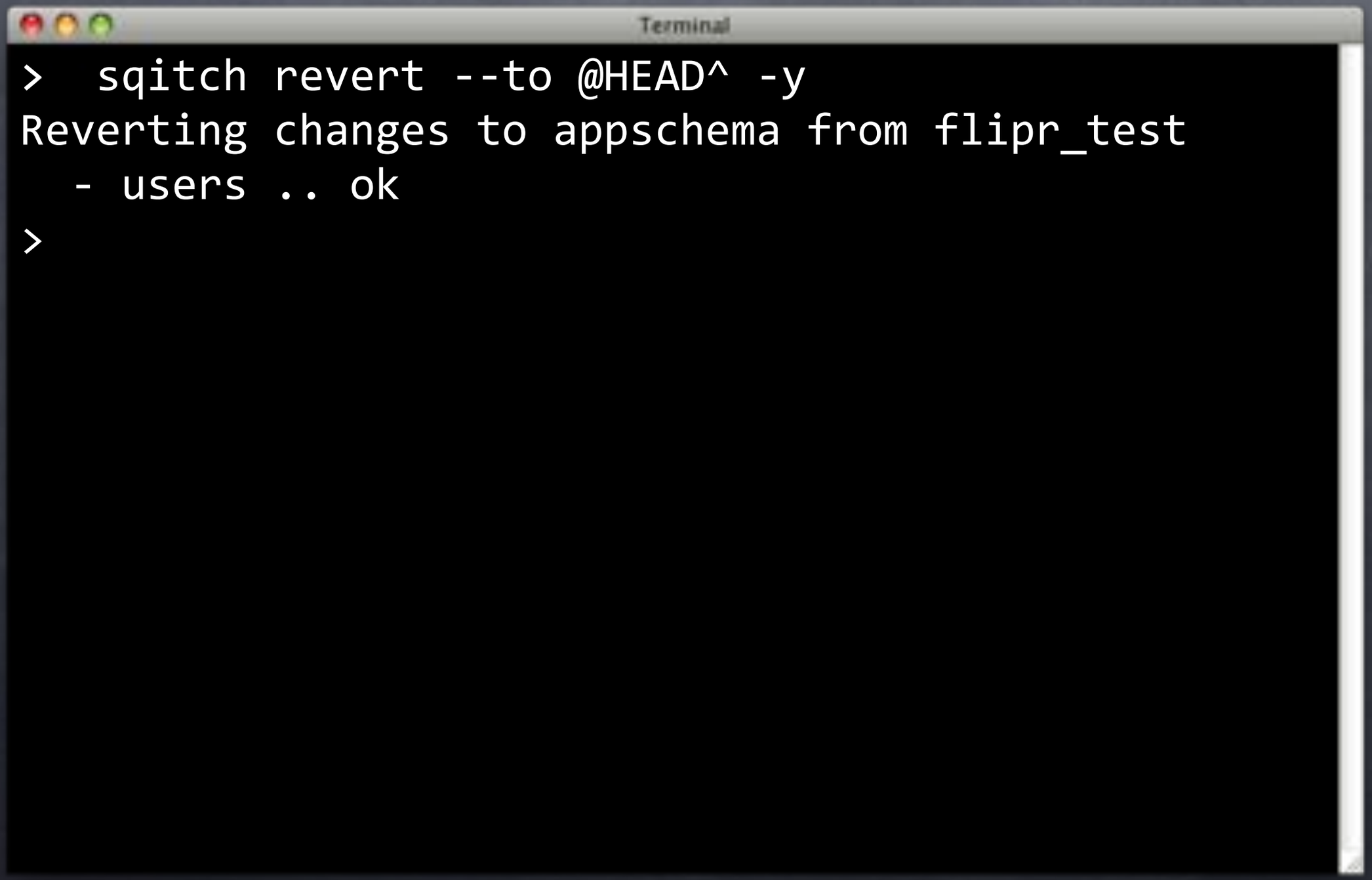
Specifying Changes

- users — Change named "users"
- @HEAD^ — Second to last change
- users^ — Change before users
- @ROOT~ — Second change
- appschema~ — Change after appschema
- @HEAD^^ — Third to last change

Specifying Changes

- users — Change named "users"
- @HEAD^ — Second to last change
- users^ — Change before users
- @ROOT~ — Second change
- appschema~ — Change after appschema
- @HEAD^^ — Third to last change
- users~4 — 4th change after users

Whither Users



Whither Users

Terminal

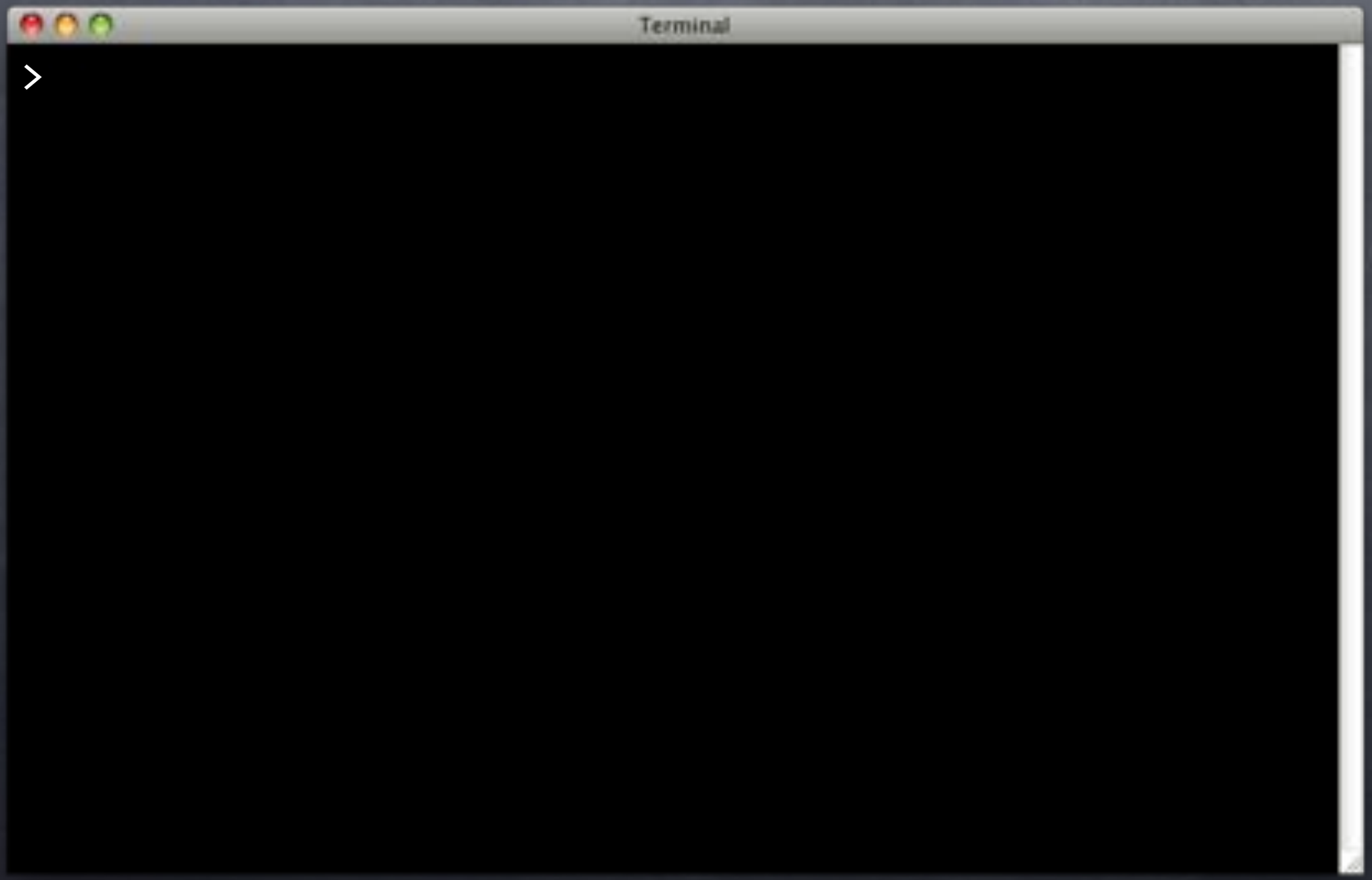
```
>  sqitch revert --to @HEAD^ -y
Reverting changes to appschema from flipr_test
  - users .. ok
> psql -d flipr_test -c '\d flipr.users'
Did not find any relation named "flipr.users".
>
```

Whither Users

Terminal

```
>  sqitch revert --to @HEAD^ -y
Reverting changes to appschema from flipr_test
  - users .. ok
> psql -d flipr_test -c '\d flipr.users'
Did not find any relation named "flipr.users".
> sqitch status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8b7088fd75ad8d7aecda3
# Name:     appschema
# Deployed: 2013-05-21 15:10:02 -0400
# By:       David E. Wheeler <david@justatheory.com>
#
Undeployed change:
  * users
```

Whither Users

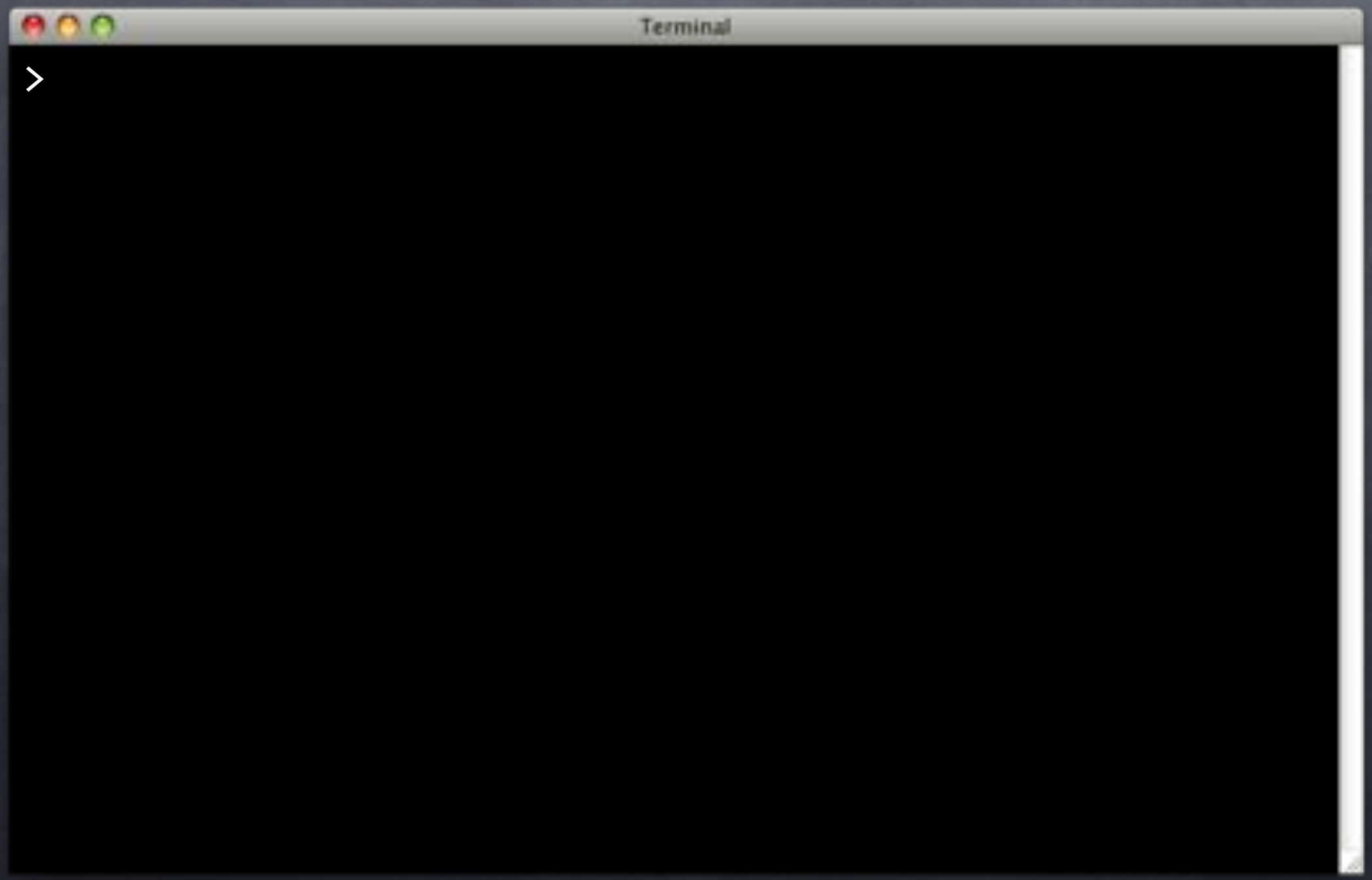


Whither Users

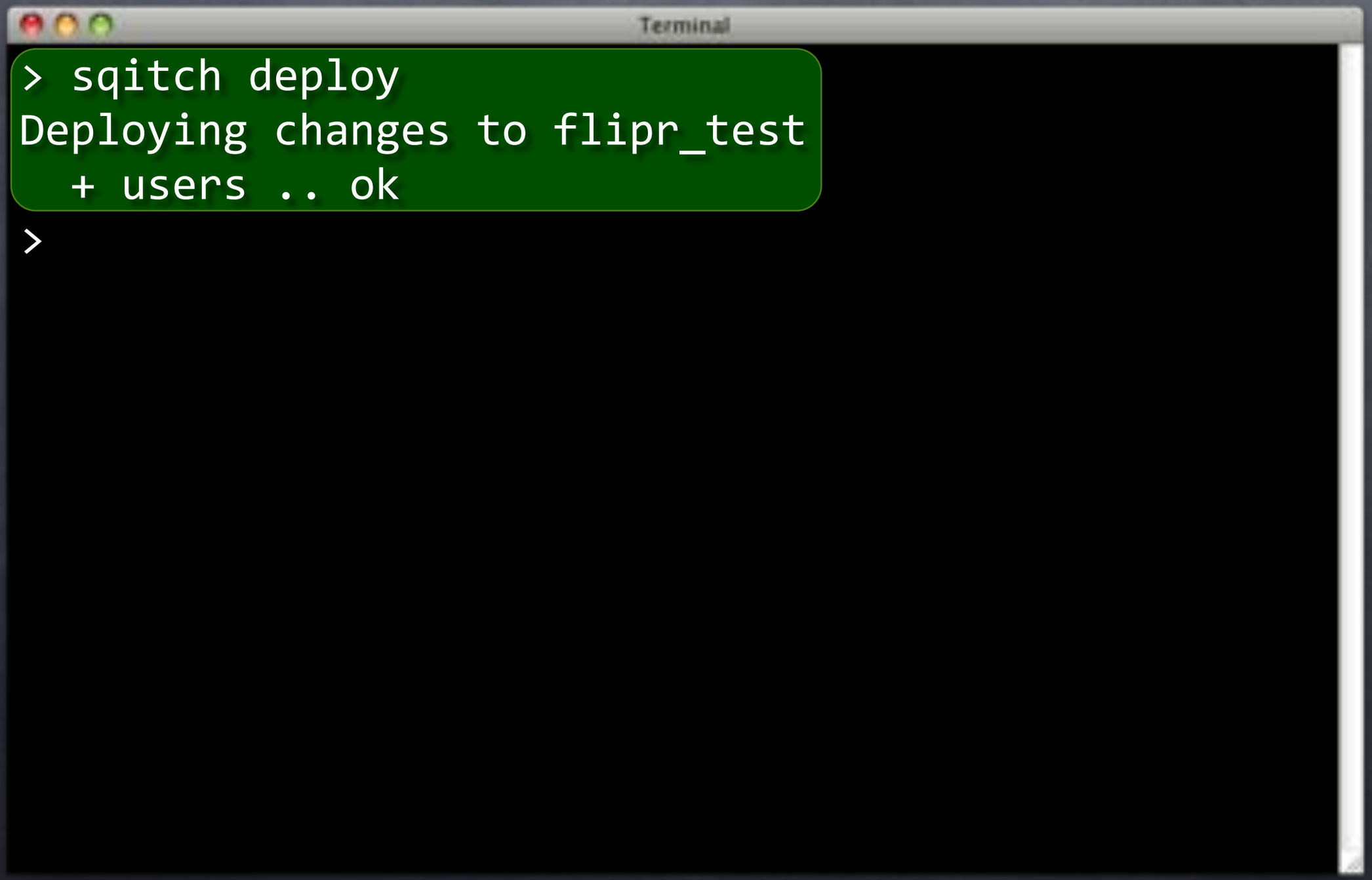
Terminal

```
> sqitch verify
Verifying flipr_test
  * appschema .. ok
Undeployed change:
  * users
Verify successful
>
```

Back At It



Back At It


Terminal
> sqitch deploy
Deploying changes to flipr_test
  + users .. ok
>

Back At It

Terminal

```
> sqitch deploy
Deploying changes to flipr_test
  + users .. ok
>
```

Add

Back At It

Terminal

```
> sqitch deploy
Deploying changes to flipr_test
  + users .. ok
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
ok 1 - Table users should exist
ok 2 - Column users.nickname should exist
ok 3 - Column users.password should exist
ok 4 - Column users."timestamp" should exist
1..4
ok
All tests successful.
Files=1, Tests=4,  0 wallclock secs
Result: PASS
>
```

Back At It

Terminal

```
> sqitch deploy
Deploying changes to flipr_test
  + users .. ok
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
ok 1 - Table users should exist
ok 2 - Column users.nickname should exist
ok 3 - Column users.password should exist
ok 4 - Column users."timestamp" should exist
1..4
ok
All tests successful.
Files=1, Tests=4,  0 wallclock secs
Result: PASS
>
```

Woot!

Back At It

Terminal

```
> sqitch deploy
Deploying changes to flipr_test
  + users .. ok
> pg_prove -d flipr_test -v test/users.sql
test/users.sql ..
ok 1 - Table users should exist
ok 2 - Column users.nickname should exist
ok 3 - Column users.password should exist
ok 4 - Column users."timestamp" should exist
1..4
ok
All tests successful.
Files=1, Tests=4,  0 wallclock secs
Result: PASS
> emacs verify/users.sql
```

Emacs

```
SET search_path = public,tap;

BEGIN;
SELECT * FROM no_plan();

SELECT has_table(  'users' );
SELECT has_column(         'users', 'nickname' );
SELECT has_column(         'users', 'password' );
SELECT has_column(         'users', 'timestamp' );

SELECT finish();
ROLLBACK;
```

-:**- test/users.sql   All (SQL[ansi])----------

Emacs

```
SET search_path = public,tap;

BEGIN;
SELECT * FROM no_plan();

SELECT has_table(  'users' );
SELECT has_pk(     'users' );

SELECT has_column(        'users', 'nickname' );
SELECT has_column(        'users', 'password' );
SELECT has_column(        'users', 'timestamp' );

SELECT finish();
ROLLBACK;
```

-:**- test/users.sql   All (SQL[ansi])----------

Emacs

```
SET search_path = public,tap;

BEGIN;
SELECT * FROM no_plan();

SELECT has_table(  'users' );
SELECT has_pk(     'users' );

SELECT has_column(         'users', 'nickname' );
SELECT col_type_is(        'users', 'nickname', 'text' );
SELECT col_hasnt_default( 'users', 'nickname' );
SELECT col_is_pk(          'users', 'nickname' );

SELECT has_column(         'users', 'password' );
SELECT has_column(         'users', 'timestamp' );

SELECT finish();
ROLLBACK;
```

```
-:**-  test/users.sql    All  (SQL[ansi])----------------------------------------------------------------------------
```

Emacs

```
SET search_path = public,tap;

BEGIN;
SELECT * FROM no_plan();

SELECT has_table(  'users' );
SELECT has_pk(     'users' );

SELECT has_column(         'users', 'nickname' );
SELECT col_type_is(        'users', 'nickname', 'text' );
SELECT col_hasnt_default( 'users', 'nickname' );
SELECT col_is_pk(          'users', 'nickname' );

SELECT has_column(         'users', 'password' );
SELECT col_type_is(        'users', 'password', 'text' );
SELECT col_not_null(       'users', 'password' );
SELECT col_hasnt_default( 'users', 'password' );

SELECT has_column(         'users', 'timestamp' );

SELECT finish();
ROLLBACK;
```

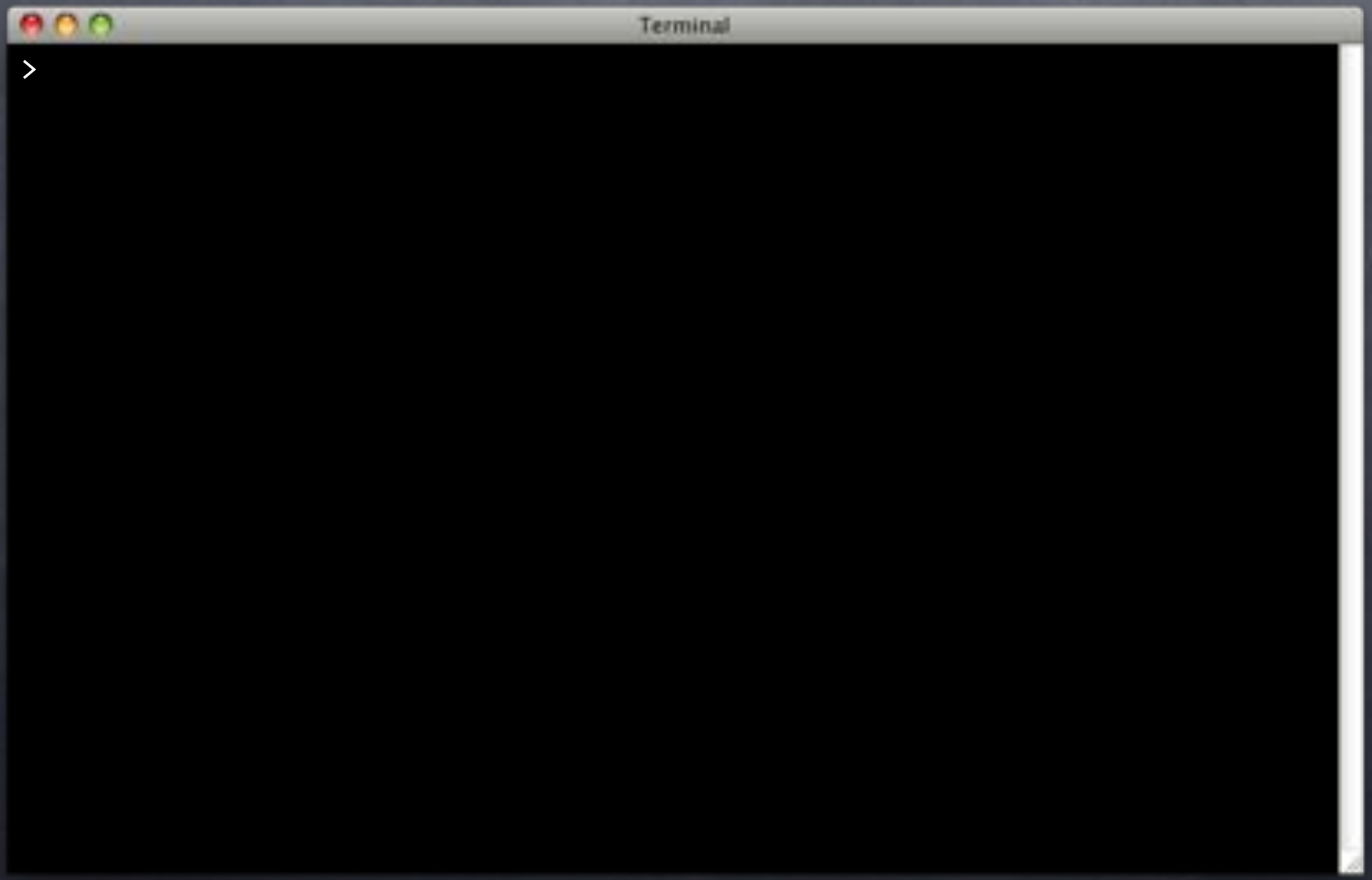
-:**- test/users.sql All (SQL[ansi])

Emacs

```
SET search_path = public,tap;

BEGIN;
SELECT * FROM no_plan();

SELECT has_table(  'users' );
SELECT has_pk(     'users' );

SELECT has_column(         'users', 'nickname' );
SELECT col_type_is(        'users', 'nickname', 'text' );
SELECT col_hasnt_default( 'users', 'nickname' );
SELECT col_is_pk(          'users', 'nickname' );

SELECT has_column(         'users', 'password' );
SELECT col_type_is(        'users', 'password', 'text' );
SELECT col_not_null(       'users', 'password' );
SELECT col_hasnt_default( 'users', 'password' );

SELECT has_column(         'users', 'timestamp' );
SELECT col_type_is('users', 'timestamp', 'timestamp with time zone');
SELECT col_not_null(       'users', 'timestamp' );
SELECT col_has_default(    'users', 'timestamp' );
SELECT col_default_is(     'users', 'timestamp', 'now()' );

SELECT finish();
ROLLBACK;
```

-:--- test/users.sql All (SQL[ansi])

Columny



Columny

Terminal

```
> pg_prove -d flipr_test test/users.sql
test/users.sql .. 1/?
# Failed test 2: "Table users should have a primary key"
# Failed test 6: "Column users(nickname) should be a primary key"
#         have: NULL
#         want: {nickname}
# Failed test 9: "Column users.password should be NOT NULL"
# Failed test 13: "Column users."timestamp" should be NOT NULL"
# Failed test 14: "Column users."timestamp" should have a default"
# Failed test 15: "Column users."timestamp" should default to 'now()'"
#     Column users."timestamp" has no default
# Looks like you failed 6 tests of 15
test/users.sql .. Failed 6/15 subtests

Test Summary Report
-------------------
test/users.sql (Wstat: 0 Tests: 15 Failed: 6)
  Failed tests:  2, 6, 9, 13-15
Files=1, Tests=15,  0 wallclock secs
Result: FAIL
```

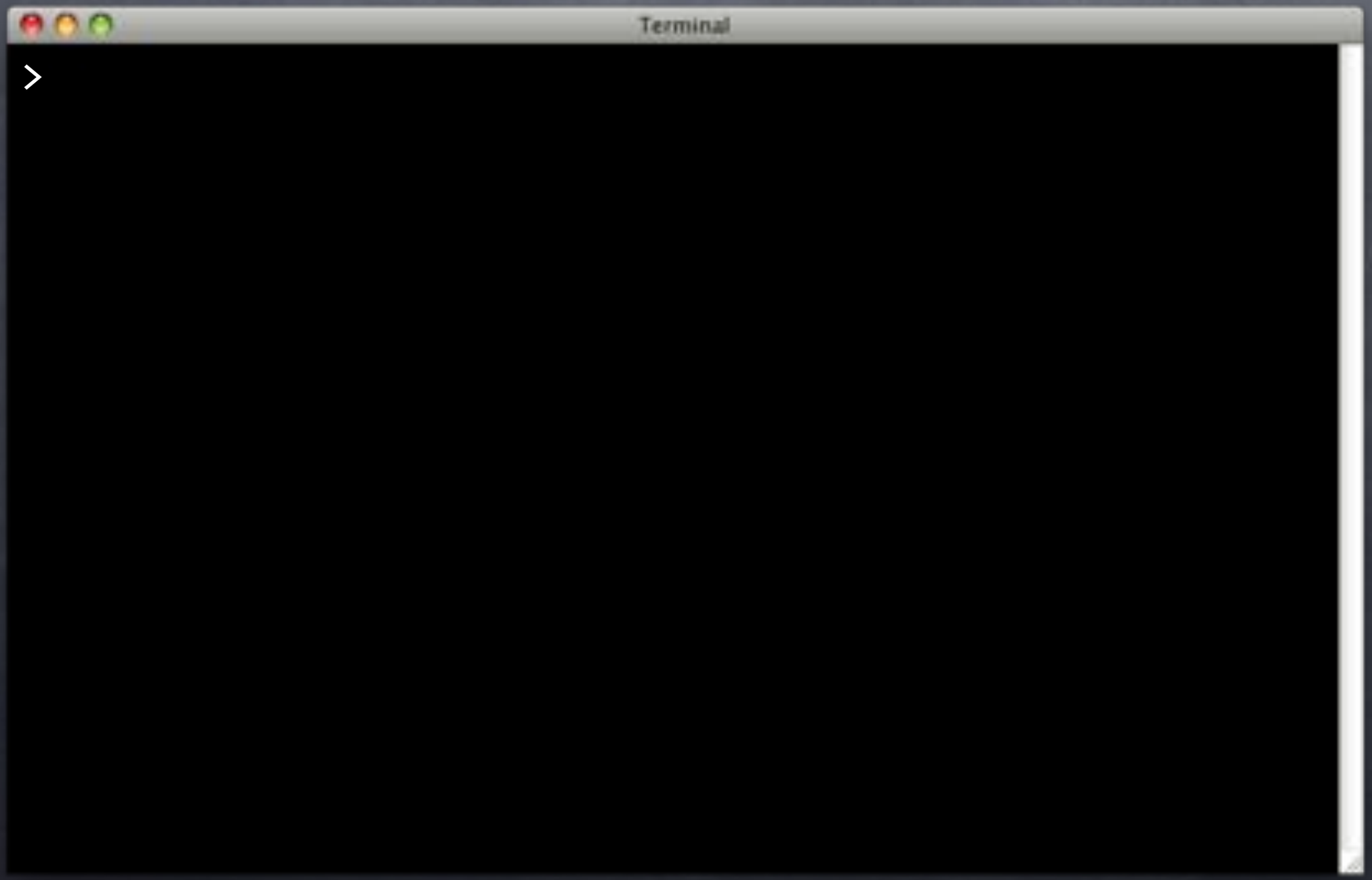
Columny

Terminal

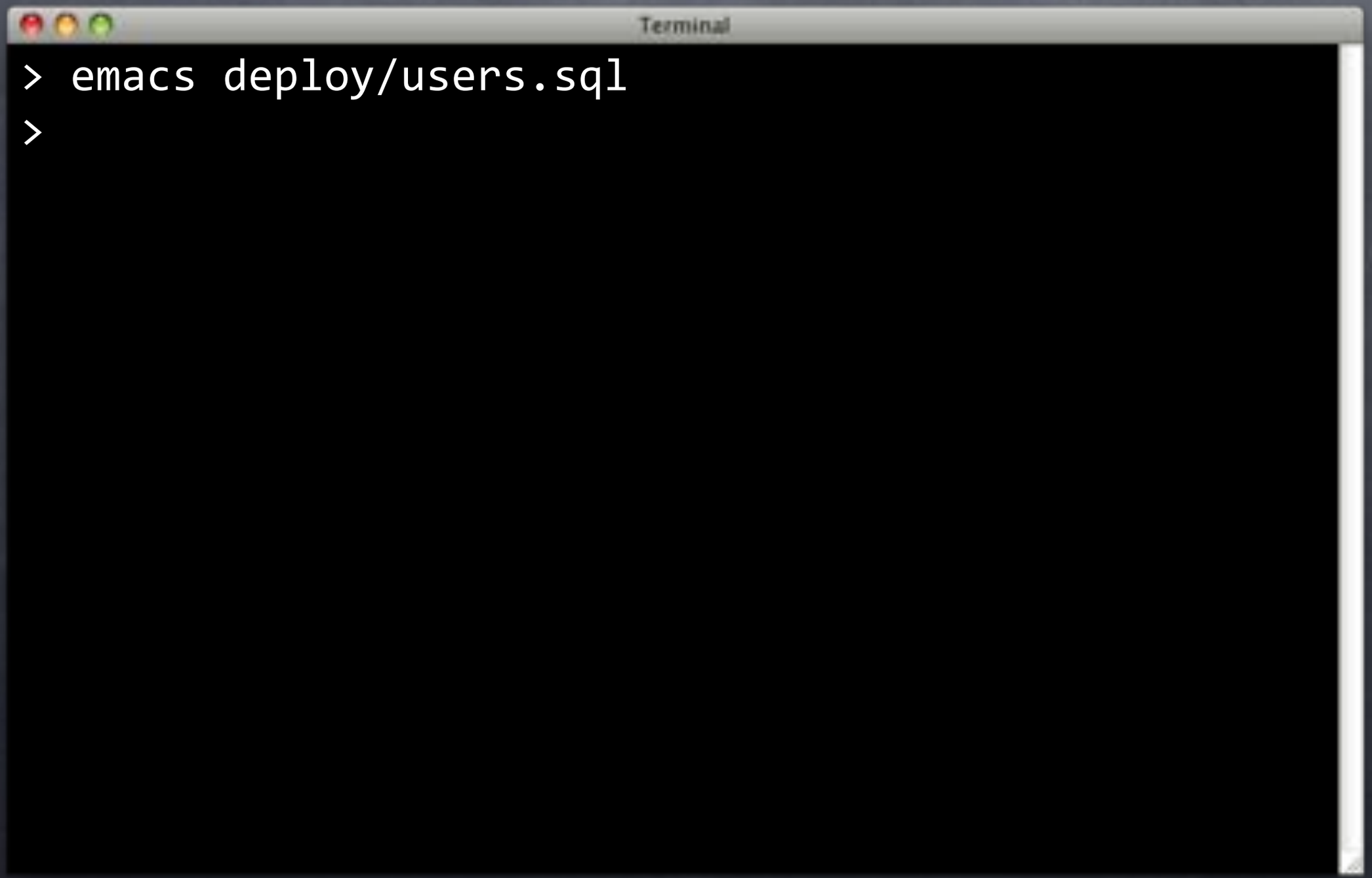
```
> pg_prove -d flipr_test test/users.sql
test/users.sql .. 1/?
# Failed test 2: "Table users should have a primary key"
# Failed test 6: "Column users(nickname) should be a primary key"
#         have: NULL
#         want: {nickname}
# Failed test 9: "Column users.password should be NOT NULL"
# Failed test 13: "Column users."timestamp" should be NOT NULL"
# Failed test 14: "Column users."timestamp" should have a default"
# Failed test 15: "Column users."timestamp" should default to 'now()'"
#     Column users."timestamp" has no default
# Looks like you failed 6 tests of 15
test/users.sql .. Failed 6/15 subtests

Test Summary Report
-------------------
test/users.sql (Wstat: 0 Tests: 15 Failed
  Failed tests:  2, 6, 9, 13-15
Files=1, Tests=15,  0 wallclock secs
Result: FAIL
```

Let's make it so.

User Typography



User Typography



deploy/users.sql

Emacs

```
-- Deploy users
-- requires: appschema

BEGIN;

SET client_min_messages = 'warning';
CREATE TABLE flipr.users (
    nickname  TEXT,
    password  TEXT,
    timestamp TIMESTAMPTZ
);

COMMIT;
```

-:**- deploy/users.sq All (SQL[ansi])

deploy/users.sql

Emacs

```
-- Deploy users
-- requires: appschema

BEGIN;

SET client_min_messages = 'warning';
CREATE TABLE flipr.users (
    nickname  TEXT        PRIMARY KEY,
    password  TEXT,
    timestamp TIMESTAMPTZ
);

COMMIT;
```

-:** deploy/users.sq All (SQL[ansi])

deploy/users.sql

Emacs

```
-- Deploy users
-- requires: appschema

BEGIN;

SET client_min_messages = 'warning';
CREATE TABLE flipr.users (
    nickname  TEXT        PRIMARY KEY,
    password  TEXT        NOT NULL,
    timestamp TIMESTAMPTZ
);

COMMIT;
```

-:** deploy/users.sq All (SQL[ansi])

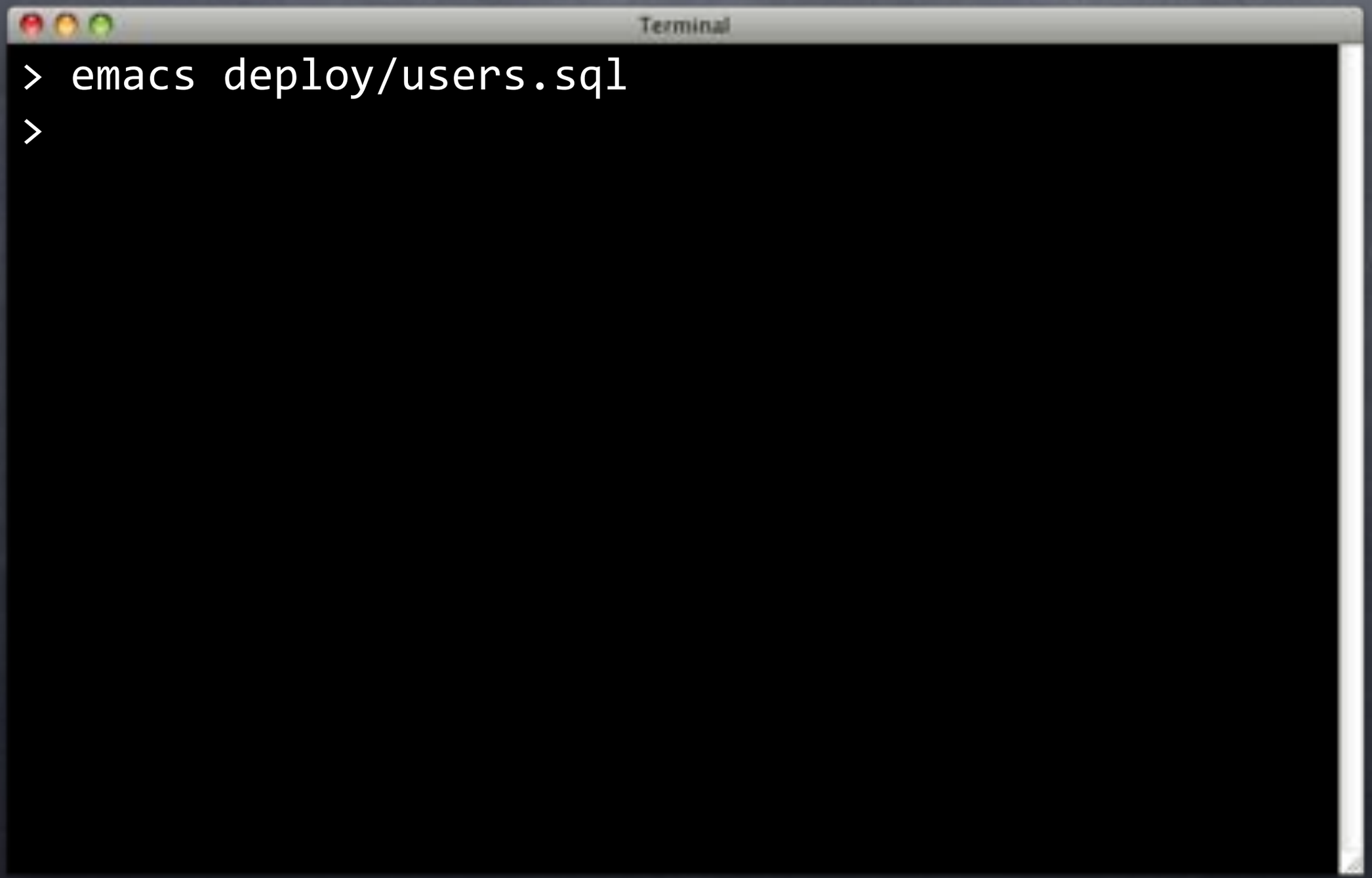
deploy/users.sql

Emacs

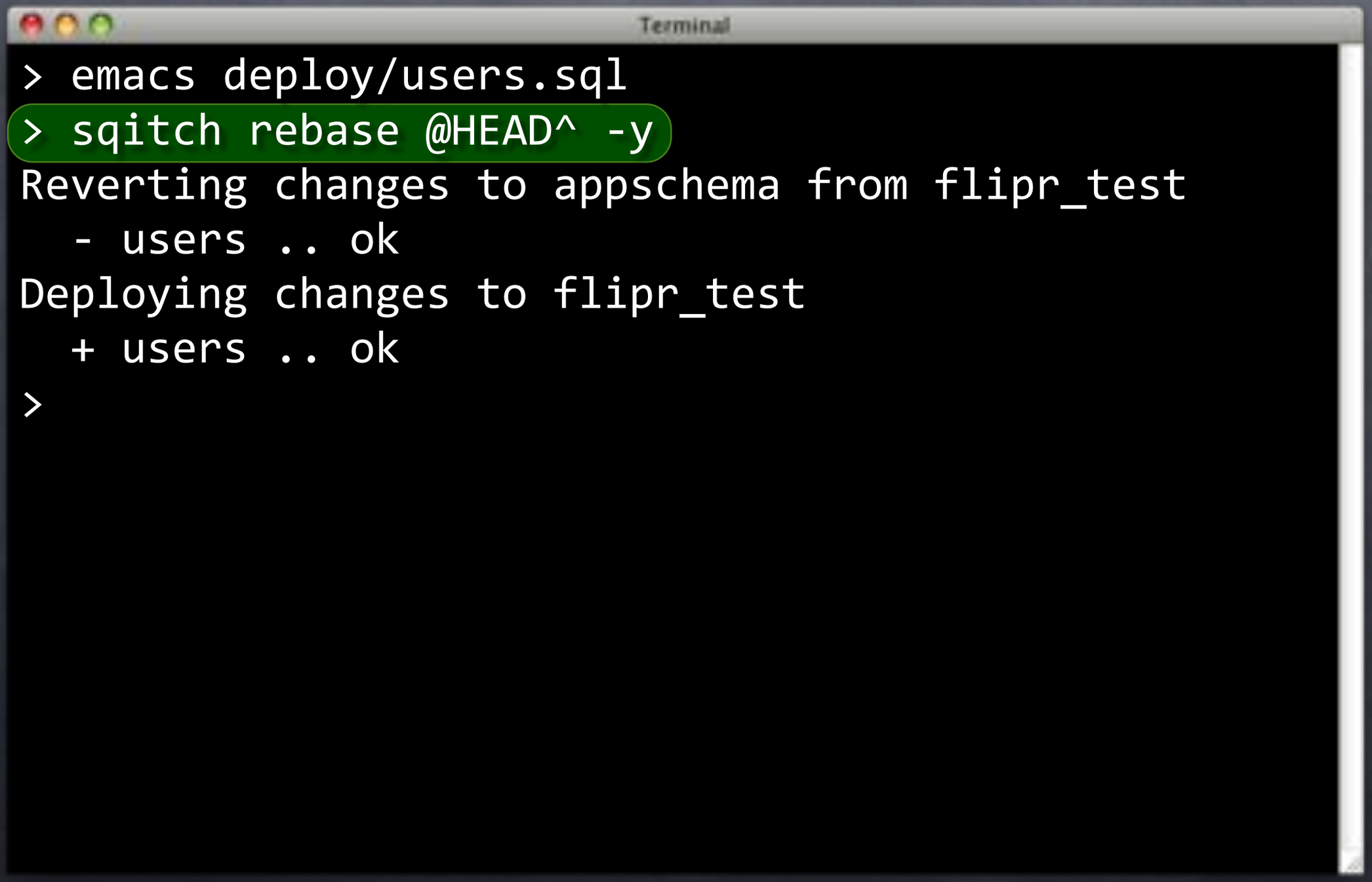
```
-- Deploy users
-- requires: appschema

BEGIN;

SET client_min_messages = 'warning';
CREATE TABLE flipr.users (
    nickname  TEXT        PRIMARY KEY,
    password  TEXT        NOT NULL,
    timestamp TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

COMMIT;
```

-:** deploy/users.sq All (SQL[ansi])

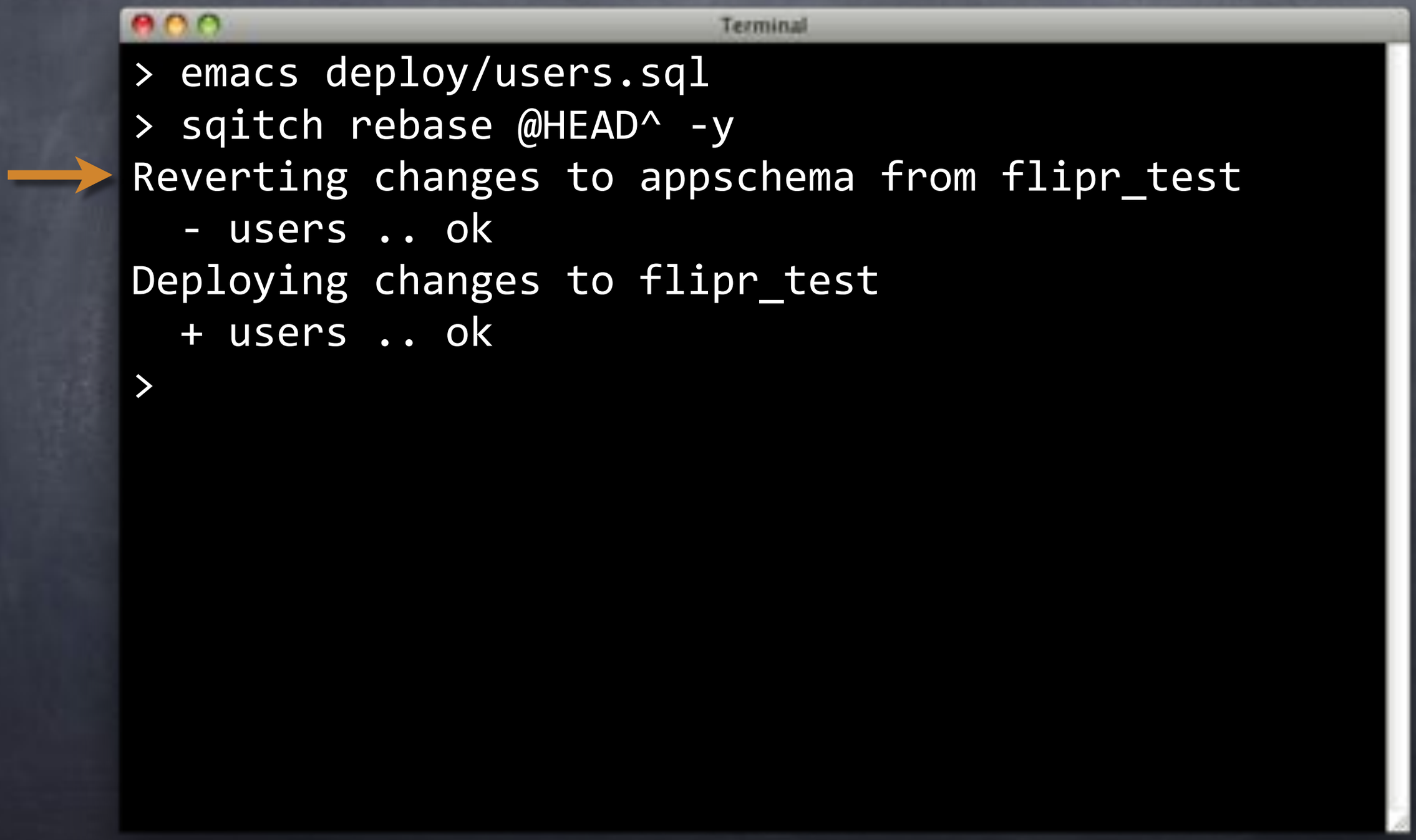
User Typography



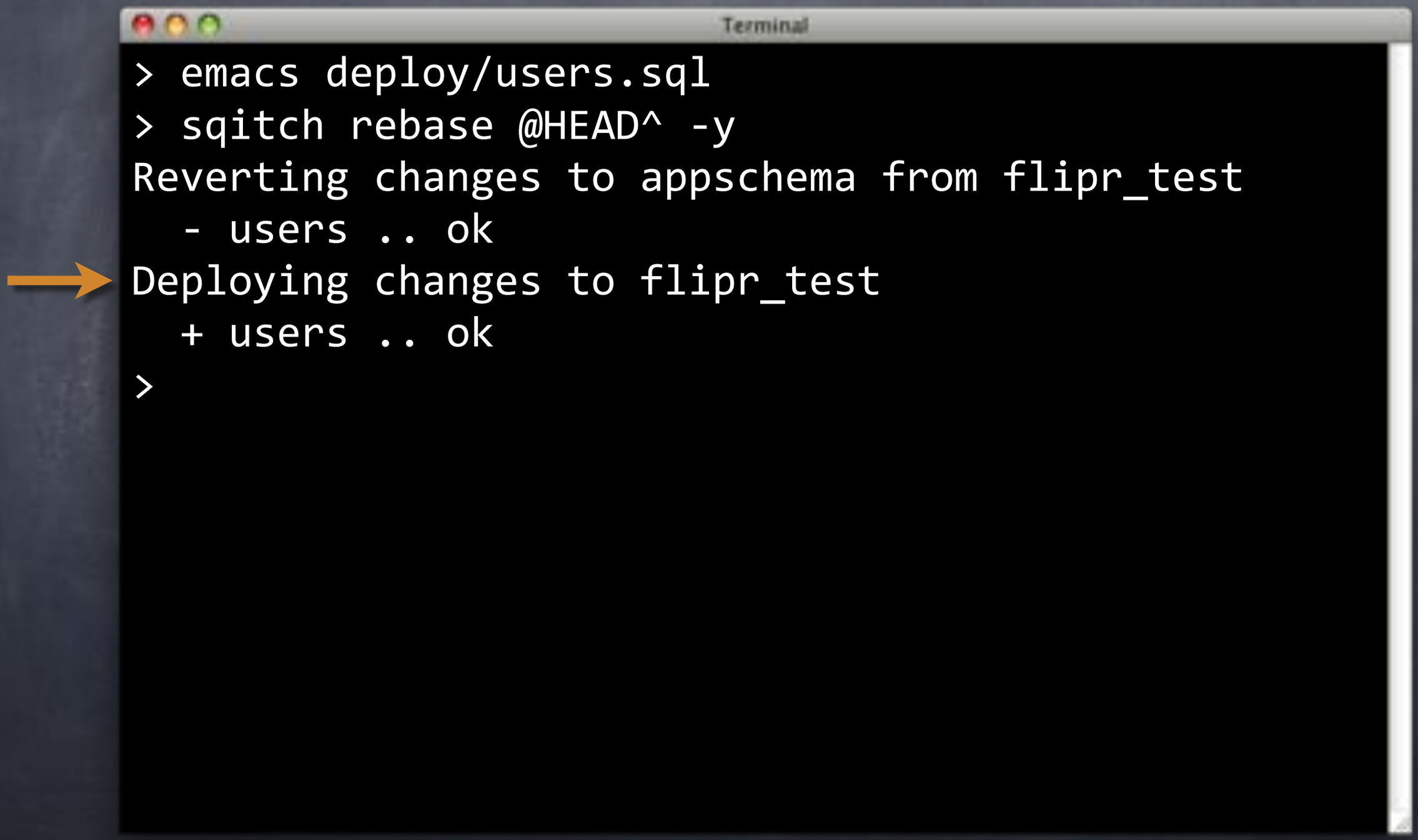
User Typography



User Typography



User Typography



User Typography

Terminal

```
> emacs deploy/users.sql
> sqitch rebase @HEAD^ -y
Reverting changes to appschema from flipr_test
  - users .. ok
Deploying changes to flipr_test
  + users .. ok
> pg_prove -d flipr_test test/users.sql
test/users.sql .. ok
All tests successful.
Files=1, Tests=15,  0 wallclock secs
Result: PASS
>
```

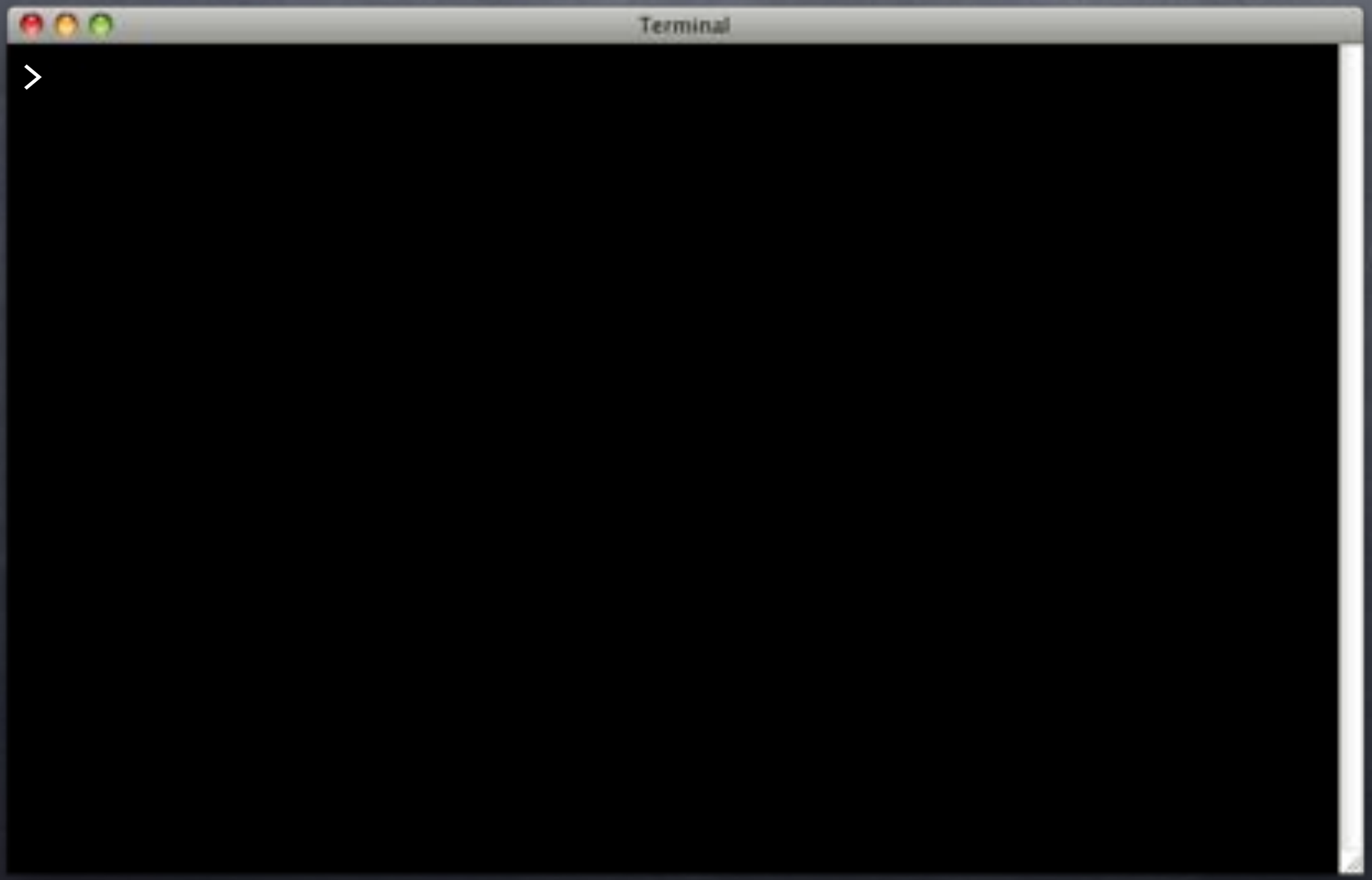
User Typography

Terminal

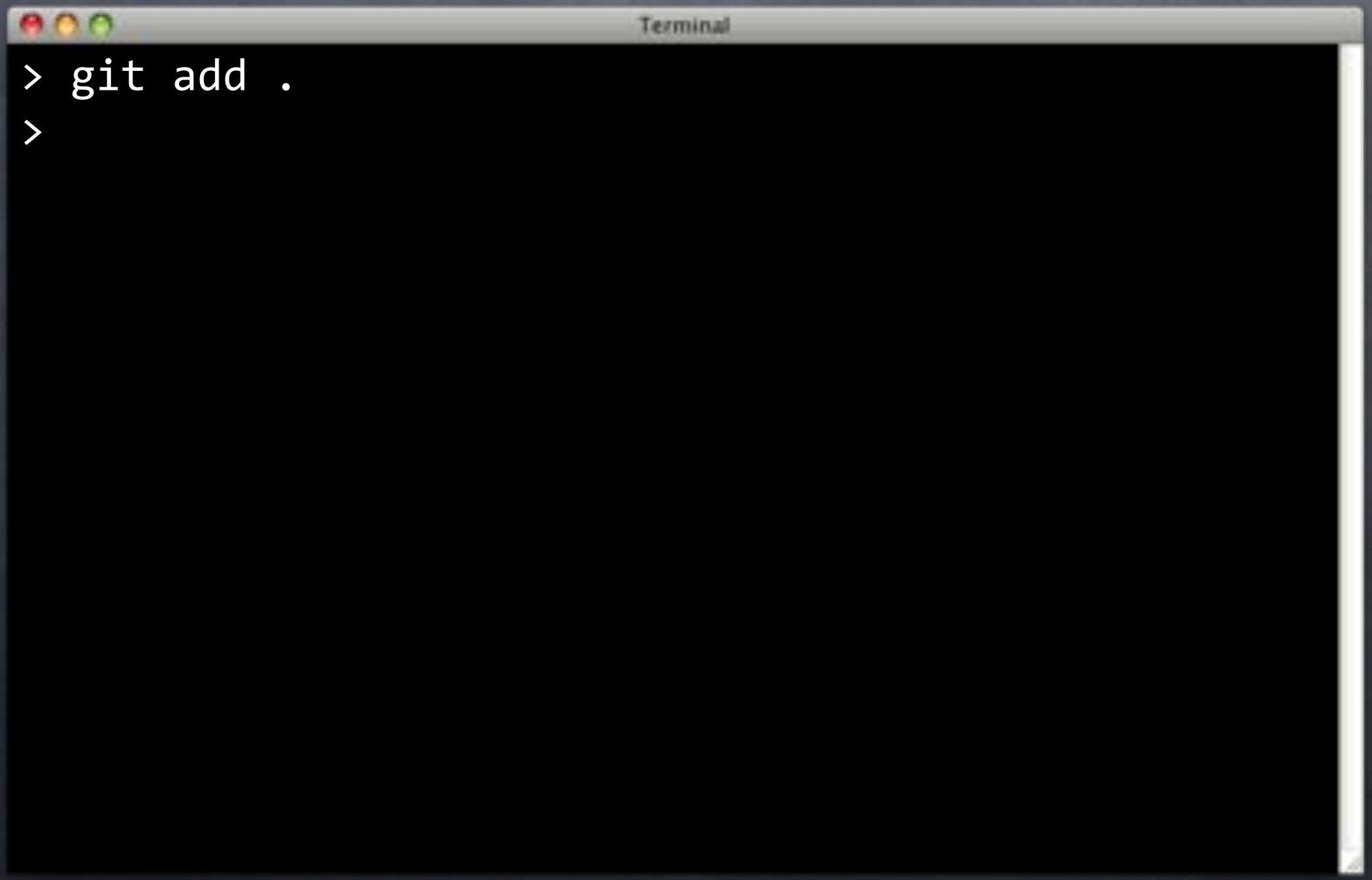
```
> emacs deploy/users.sql
> sqitch rebase @HEAD^ -y
Reverting changes to appschema from flipr_test
  - users .. ok
Deploying changes to flipr_test
  + users .. ok
> pg_prove -d flipr_test test/users.sql
test/users.sql .. ok
All tests successful.
Files=1, Tests=15,  0 wallclock secs
Result: PASS
>
```

Boom.

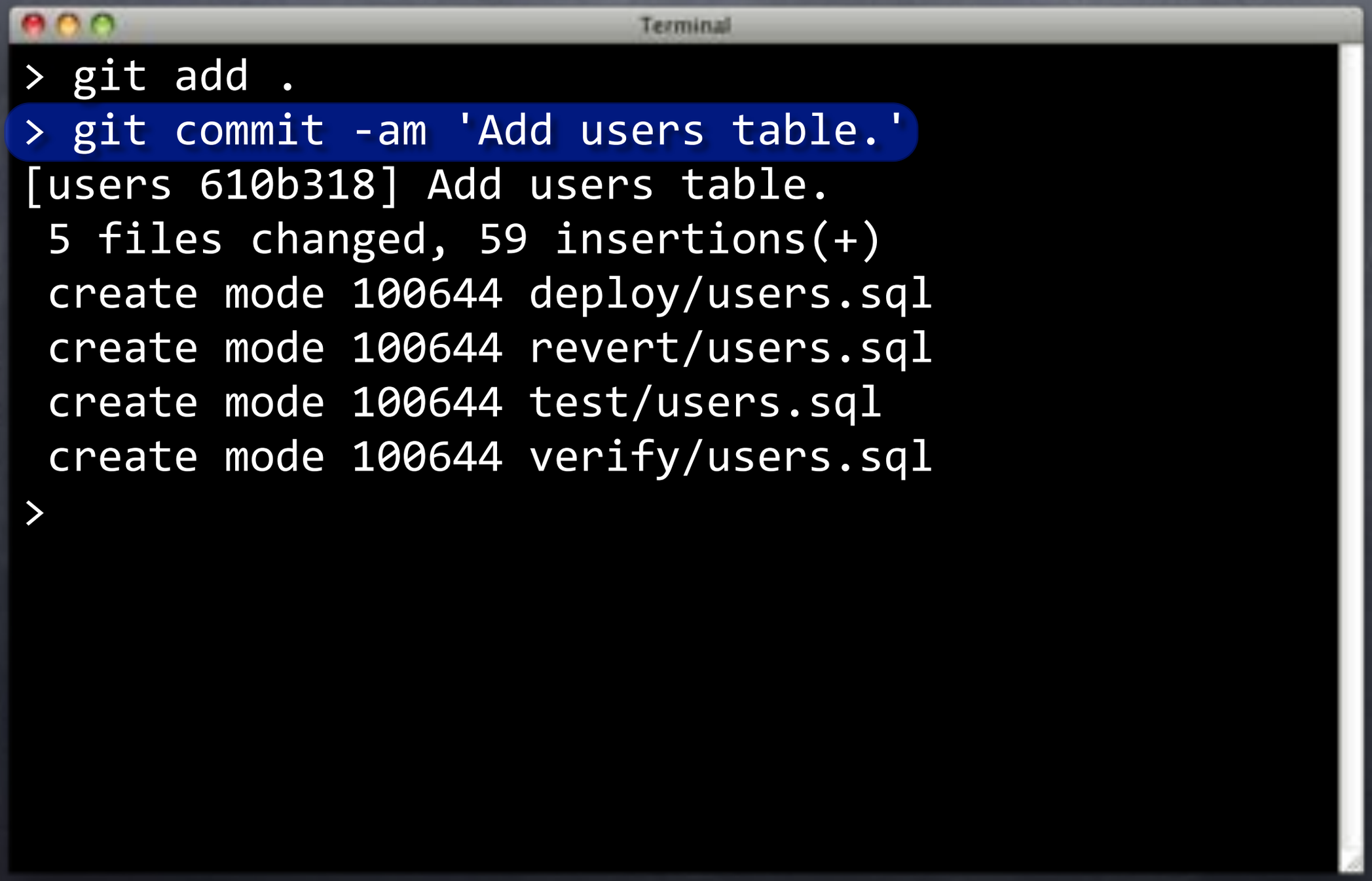
Additives



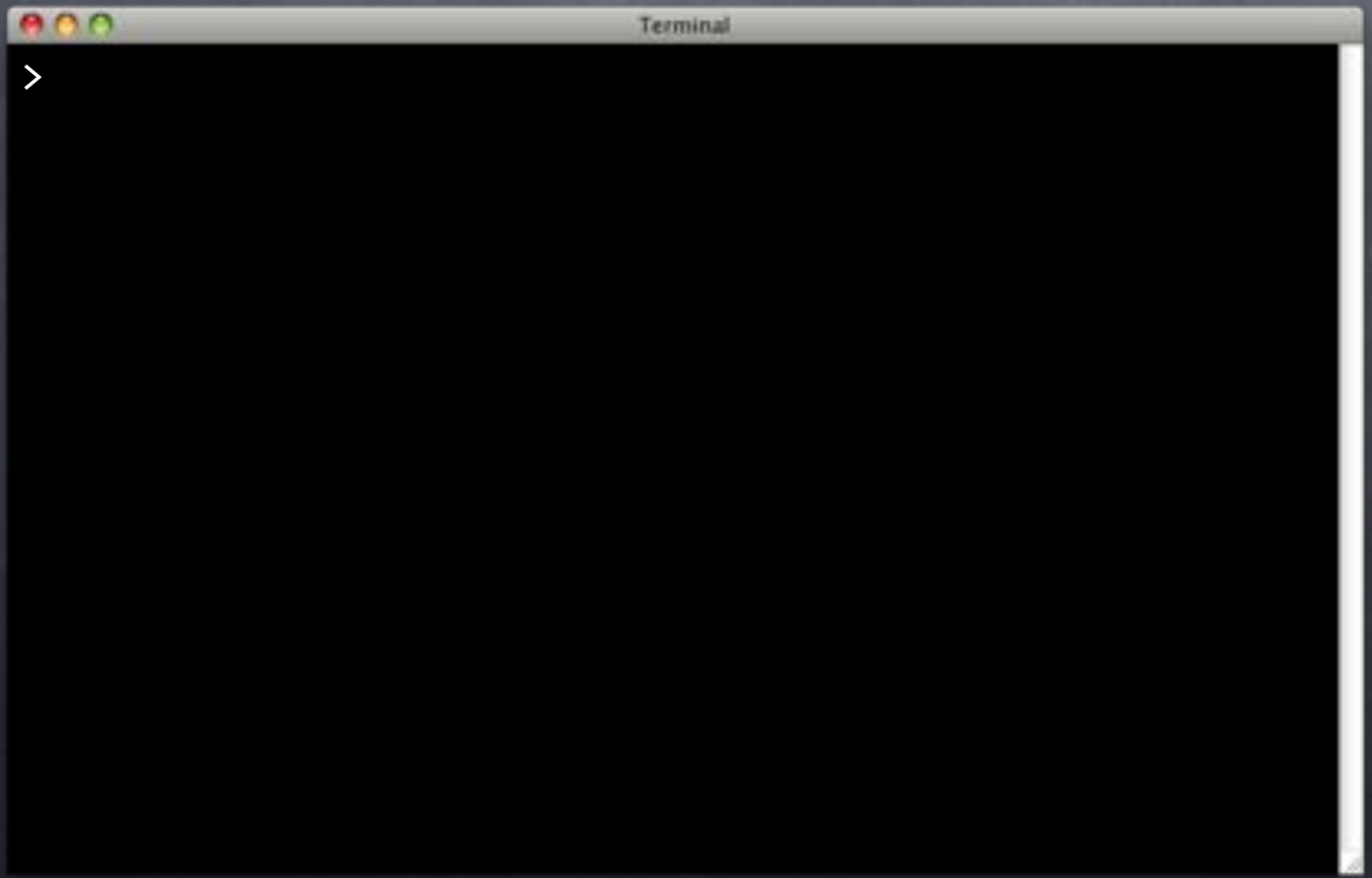
Additives



Additives


Terminal
> git add .
> git commit -am 'Add users table.'
[users 610b318] Add users table.
 5 files changed, 59 insertions(+)
 create mode 100644 deploy/users.sql
 create mode 100644 revert/users.sql
 create mode 100644 test/users.sql
 create mode 100644 verify/users.sql
>

Pushers



Pushers

Terminal

```
> git push --set-upstream origin users
Counting objects: 17, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (11/11), 1.53 KiB, done.
Total 11 (delta 1), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      users -> users
Branch users set up to track remote branch users
from origin.
>
```

Pushers

Terminal

```
> git push --set-upstream origin users
Counting objects: 17, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (11/11), 1.53 KiB, done.
Total 11 (delta 1), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      users -> users
Branch users set up to track remote branch users
from origin.
>
```

Pushers

Terminal

```
> git push --set-upstream origin users
Counting objects: 17, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (11/11), 1.53 KiB, done.
Total 11 (delta 1), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      users -> users
Branch users set up to track remote branch users
from origin.
>
```

Wash, Rinse, Repeat

Wash, Rinse, Repeat

- Add failing simple test

Wash, Rinse, Repeat

- Add failing simple test
- Add and deploy change

Wash, Rinse, Repeat

- Add failing simple test
- Add and deploy change
- Revise test

Wash, Rinse, Repeat

- Add failing simple test
- Add and deploy change
- Revise test
- Revise and rebase change

Wash, Rinse, Repeat

- Add failing simple test
- Add and deploy change
- Revise test
- Revise and rebase change
- Wash, Rinse, Repeat

Wash, Rinse, Repeat

- Add failing simple test
- Add and deploy change
- Revise test
- Revise and rebase change
- Wash, Rinse, Repeat
- Commit/Push when done

Wash, Rinse, Repeat

- Add failing simple test
- Add and deploy change
- Revise test
- Revise and rebase change
- Wash, Rinse, Repeat
- Commit/Push when done
- Breathe in, breathe out

Time to Work!

Time to Work!

- Prepare to hack!

Time to Work!

- Prepare to hack!
  - git checkout master

Time to Work!

- Prepare to hack!
  - git checkout master
  - git branch -D users

Time to Work!

- Prepare to hack!
  - git checkout master
  - git branch -D users
  - git checkout -b users

Time to Work!

- Prepare to hack!
  - git checkout master
  - git branch -D users
  - git checkout -b users
  - git reset --hard upstream/users

Time to Work!

- Prepare to hack!
  - git checkout master
  - git branch -D users
  - git checkout -b users
  - git reset --hard upstream/users
  - git log

Time to Work!

- Prepare to hack!
  - git checkout master
  - git branch -D users
  - git checkout -b users
  - git reset --hard upstream/users
  - git log
  - Should be at "Add users table."

Caution: Hard Reset Ahead

Caution: Hard Reset Ahead

- A rare destructive Git command

Caution: Hard Reset Ahead

- A rare destructive Git command
- Deletes HEAD snapshot

Caution: Hard Reset Ahead

- A rare destructive Git command
- Deletes HEAD snapshot
- Replaces it with new snapshot

Caution: Hard Reset Ahead

- A rare destructive Git command
- Deletes HEAD snapshot
- Replaces it with new snapshot
- Almost un-reversible

Caution: Hard Reset Ahead

- A rare destructive Git command
- Deletes HEAD snapshot
- Replaces it with new snapshot
- Almost un-reversible
- Useful for starting from known point

Caution: Hard Reset Ahead

- A rare destructive Git command
- Deletes HEAD snapshot
- Replaces it with new snapshot
- Almost un-reversible
- Useful for starting from known point
- USE WITH CAUTION!

Flip Out

Flip Out

- Create flips branch

Flip Out

- Create flips branch
- Create flips table

Flip Out

- Create flips branch
- Create flips table
  - flip_id SERIAL PK

Flip Out

- Create flips branch
- Create flips table
  - flip_id SERIAL PK
  - nickname FK

Flip Out

- Create flips branch
- Create flips table
  - flip_id SERIAL PK
  - nickname FK
  - body TEXT

Flip Out

- Create flips branch
- Create flips table
  - flip_id SERIAL PK
  - nickname FK
  - body TEXT
  - timestamptz

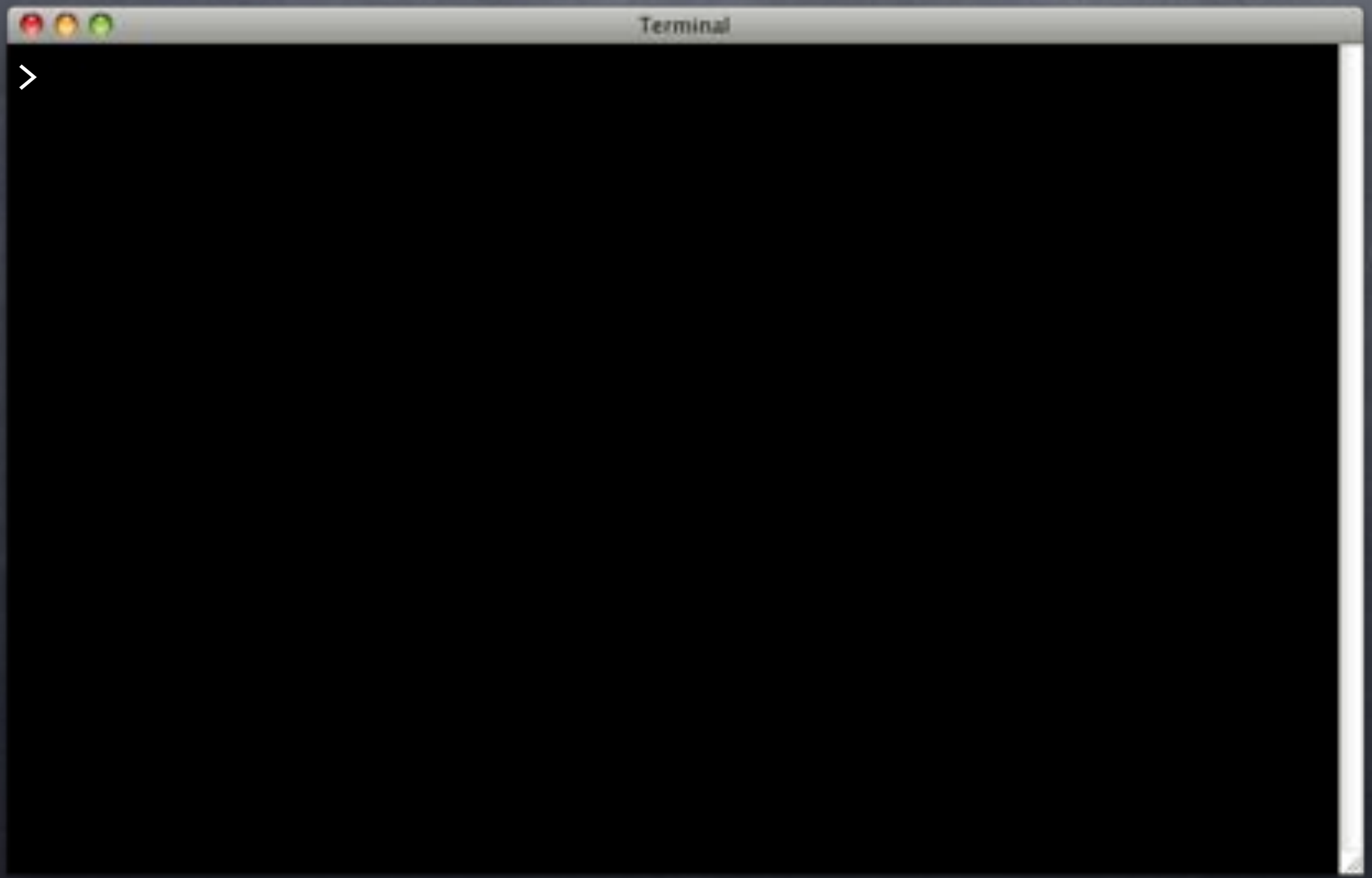
Flip Out

- Create flips branch
- Create flips table
  - flip_id SERIAL PK
  - nickname FK
  - body TEXT
  - timestamptz
- Use TDDD

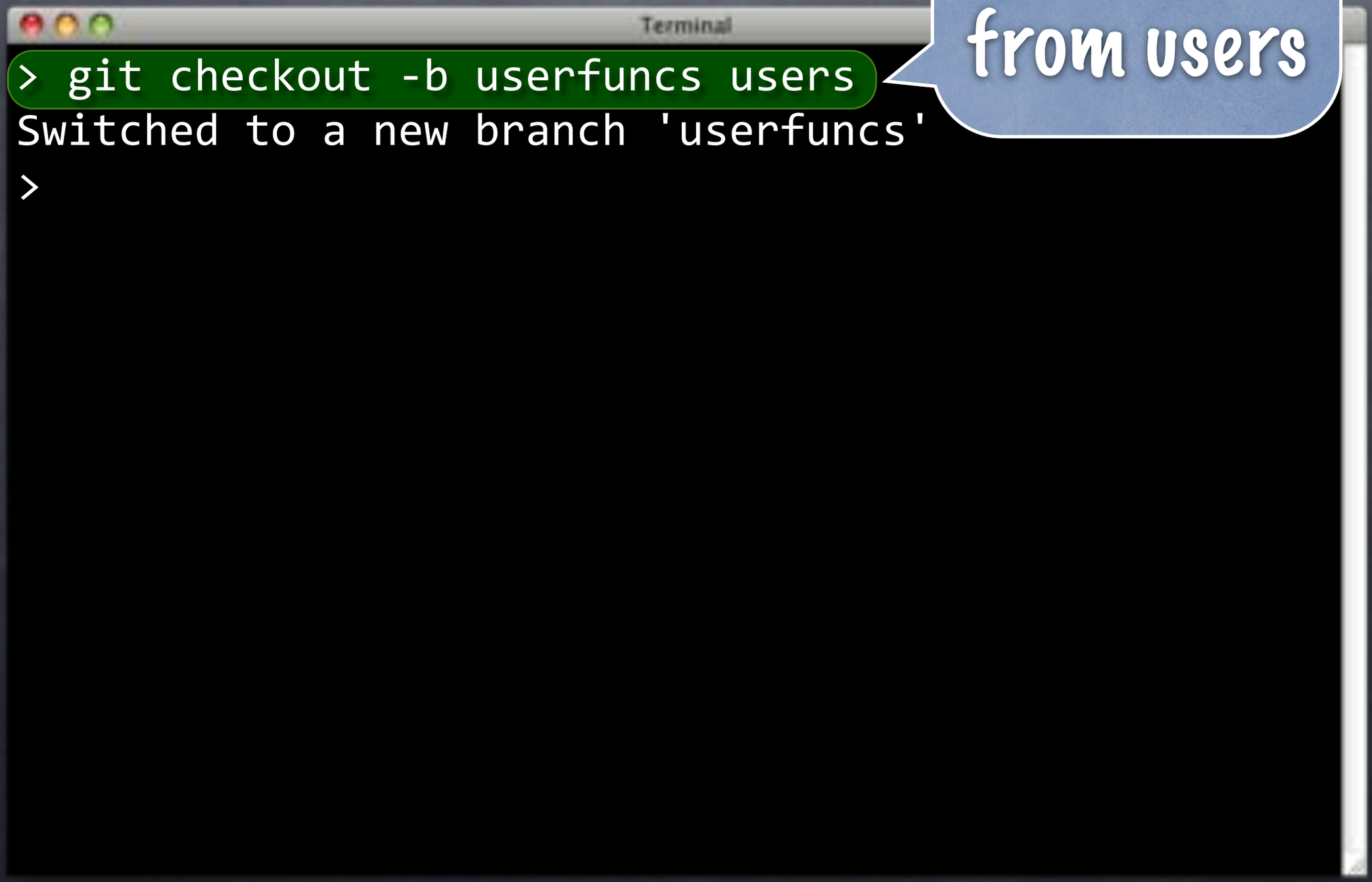
Flip Out

- Create flips branch
- Create flips table
  - flip_id SERIAL PK
  - nickname FK
  - body TEXT
  - timestamptz
- Use TDDD
- https://github.com/theory/agile-flipr.git

Functional Testing



Functional Testing



Functional Testing

Terminal

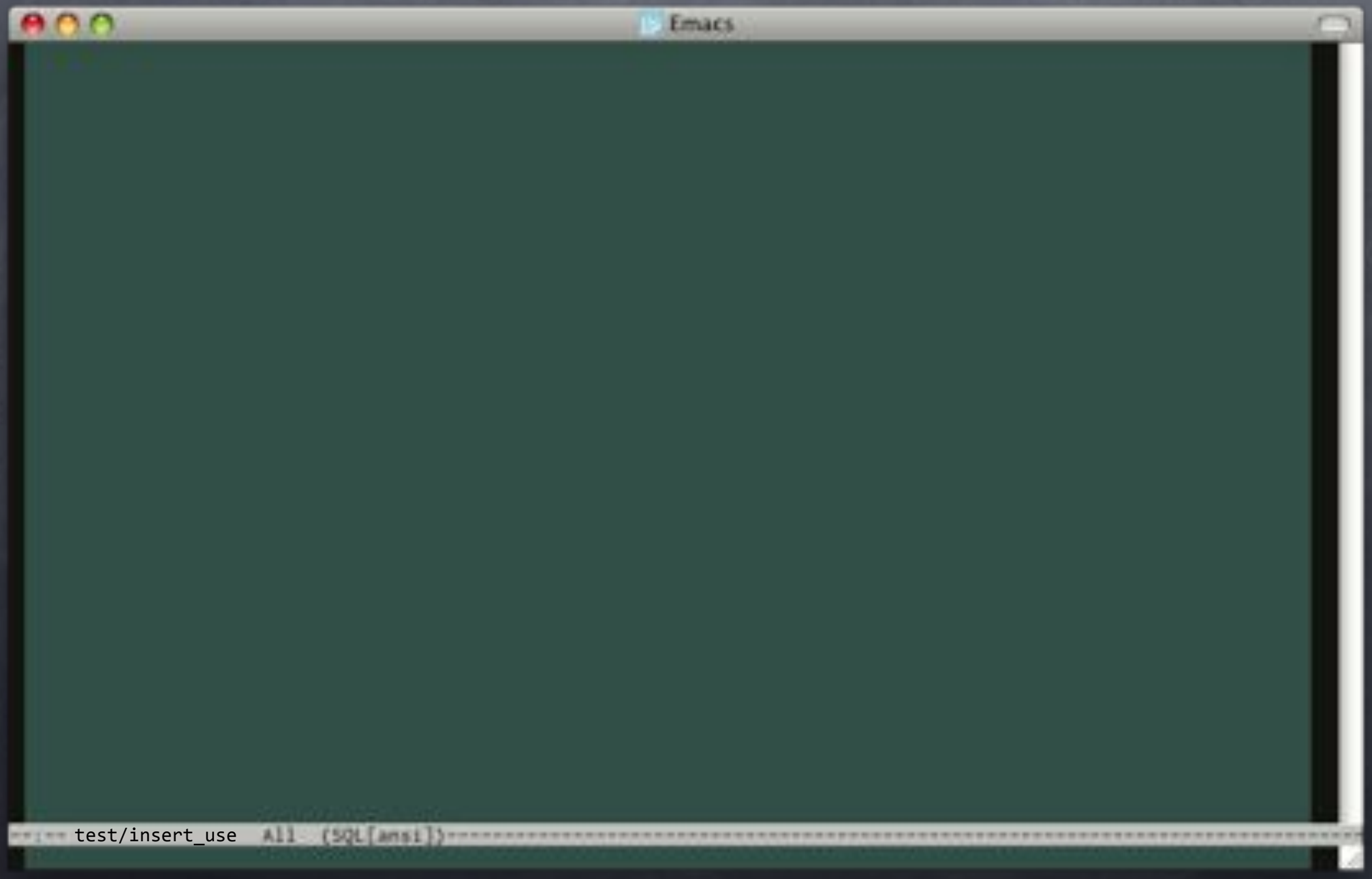
```
> git checkout -b userfuncs users
Switched to a new branch 'userfuncs'
> sqitch add insert_user -r users -r appschema \
      -n 'Creates a function to insert a user.'
Created deploy/insert_user.sql
Created revert/insert_user.sql
Created verify/insert_user.sql
Added "insert_user [users appschema]" to sqitch.plan
>
```

Functional Testing

Terminal

```
> git checkout -b userfuncs users
Switched to a new branch 'userfuncs'
> sqitch add insert_user -r users -r appschema \
      -n 'Creates a function to insert a user.'
Created deploy/insert_user.sql
Created revert/insert_user.sql
Created verify/insert_user.sql
Added "insert_user [users appschema]" to sqitch.plan
> emacs test/insert_user.sql
>
```

test/insert_user.sql



test/insert_user.sql

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;
SET search_path TO flipr,public;
```

-:** test/insert_use All (SQL[ansi])----------

test/insert_user.sql

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;
SET search_path TO flipr,public;
```

-:** test/insert_use All (SQL[ansi])----------

test/insert_user.sql

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;
SET search_path TO flipr,public;

-- Plan the tests.
BEGIN;
select plan(11);
```

-:** test/insert_use All (SQL[ansi])

test/insert_user.sql

Emacs

```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;
SET search_path TO flipr,public;

-- Plan the tests.
BEGIN;
select plan(11);
```

-:** test/insert_use All (SQL[ansi])

Emacs

```
SELECT plan(11);
```

```
-:**-  test/insert_use   All   (SQL[ansi])----------------------------------------------------------------------------
```

Emacs

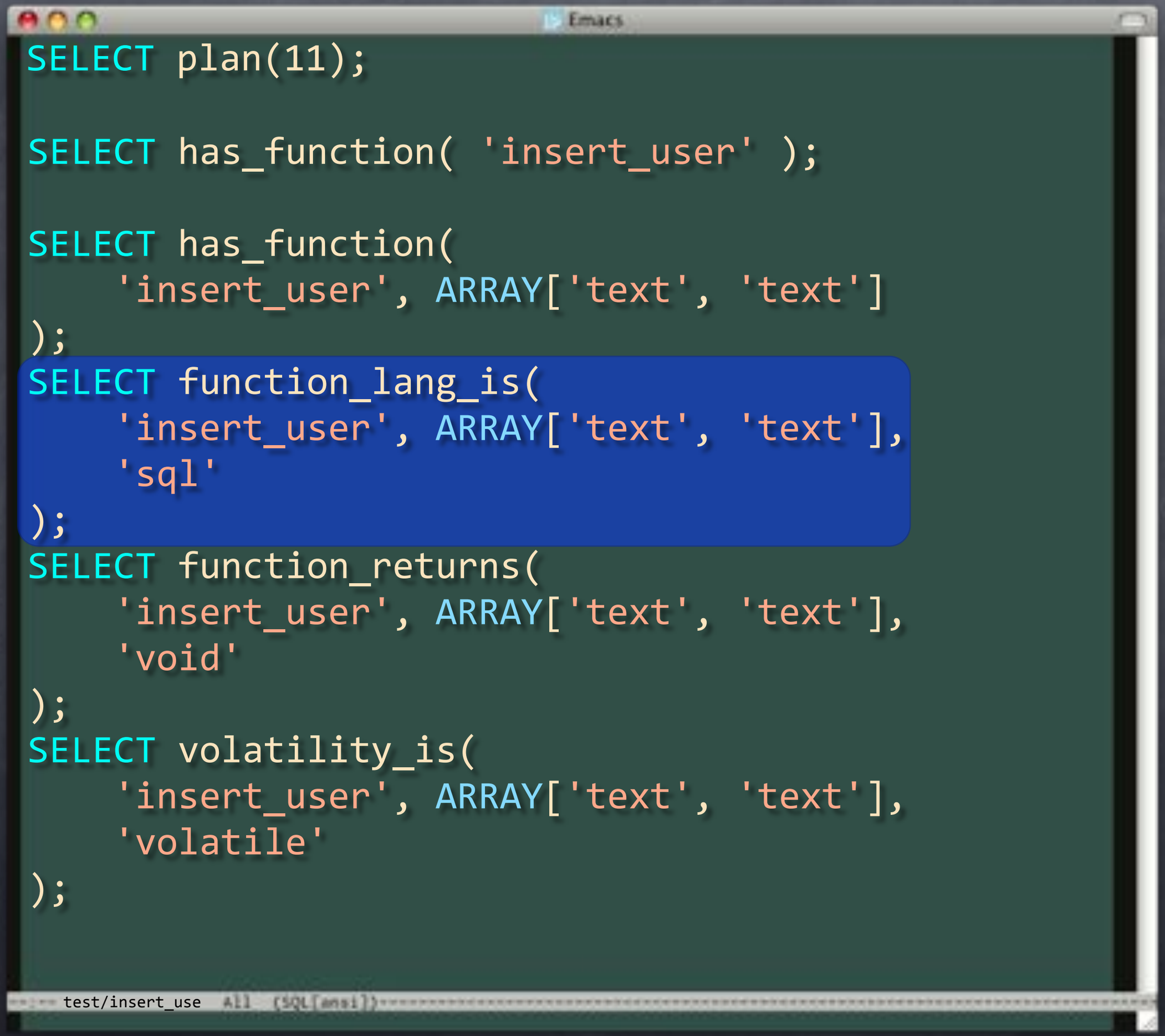
```
SELECT plan(11);

SELECT has_function( 'insert_user' );

SELECT has_function(
    'insert_user', ARRAY['text', 'text']
);
SELECT function_lang_is(
    'insert_user', ARRAY['text', 'text'],
    'sql'
);
SELECT function_returns(
    'insert_user', ARRAY['text', 'text'],
    'void'
);
SELECT volatility_is(
    'insert_user', ARRAY['text', 'text'],
    'volatile'
);
```

-:**- test/insert_use   All   (SQL[ansi])

Emacs

```
SELECT plan(11);

SELECT has_function( 'insert_user' );

SELECT has_function(
    'insert_user', ARRAY['text', 'text']
);
SELECT function_lang_is(
    'insert_user', ARRAY['text', 'text'],
    'sql'
);
SELECT function_returns(
    'insert_user', ARRAY['text', 'text'],
    'void'
);
SELECT volatility_is(
    'insert_user', ARRAY['text', 'text'],
    'volatile'
);
```

-:**- test/insert_use All (SQL[ansi])


Emacs
SELECT plan(11);
SELECT has_function('insert_user');
SELECT has_function(
'insert_user', ARRAY['text', 'text']
);
SELECT function_lang_is(
'insert_user', ARRAY['text', 'text'],
'sql'
);
SELECT function_returns(
'insert_user', ARRAY['text', 'text'],
'void'
);
SELECT volatility_is(
'insert_user', ARRAY['text', 'text'],
'volatile'
);
test/insert_use All (SQL[ansi])

Emacs

```
SELECT plan(11);

SELECT has_function( 'insert_user' );

SELECT has_function(
    'insert_user', ARRAY['text', 'text']
);
SELECT function_lang_is(
    'insert_user', ARRAY['text', 'text'],
    'sql'
);
SELECT function_returns(
    'insert_user', ARRAY['text', 'text'],
    'void'
);
SELECT volatility_is(
    'insert_user', ARRAY['text', 'text'],
    'volatile'
);
```

-:**- test/insert_use   All   (SQL[ansi])

Emacs

```
SELECT plan(11);

SELECT has_function( 'insert_user' );

SELECT has_function(
    'insert_user', ARRAY['text', 'text']
);
SELECT function_lang_is(
    'insert_user', ARRAY['text', 'text'],
    'sql'
);
SELECT function_returns(
    'insert_user', ARRAY['text', 'text'],
    'void'
);
SELECT volatility_is(
    'insert_user', ARRAY['text', 'text'],
    'volatile'
);
```

-:**- test/insert_use   All   (SQL[ansi])

test/insert_user.sql

Emacs

```
    'volatile'
);
```

-:**- test/insert_use   All   (SQL[ansi])----------

test/insert_user.sql

Emacs

```
    'volatile'
);

SELECT lives_ok(
    $$ SELECT insert_user('theory', 'foo') $$,
    'Insert a user'
);

SELECT row_eq(
    'SELECT * FROM users',
    ROW('theory', md5('foo'), NOW())::users,
    'The user should have been inserted'
);
```

-:-- test/insert_use   All   (SQL[ansi])

test/insert_user.sql

Emacs

```
    'volatile'
);

SELECT lives_ok(
    $$ SELECT insert_user('theory', 'foo') $$,
    'Insert a user'
);

SELECT row_eq(
    'SELECT * FROM users',
    ROW('theory', md5('foo'), NOW())::users,
    'The user should have been inserted'
);
```

-:**- test/insert_use   All   (SQL[ansi])----------

test/insert_user.sql

Emacs

```
    'volatile'
);

SELECT lives_ok(
    $$ SELECT insert_user('theory', 'foo') $$,
    'Insert a user'
);

SELECT row_eq(
    'SELECT * FROM users',
    ROW('theory', md5('foo'), NOW())::users,
    'The user should have been inserted'
);
```

-:-- test/insert_use   All   (SQL[ansi])

test/insert_user.sql

Emacs

```
    'volatile'
);

SELECT lives_ok(
    $$ SELECT insert_user('theory', 'foo') $$,
    'Insert a user'
);

SELECT row_eq(
    'SELECT * FROM users',
    ROW('theory', md5('foo'), NOW())::users,
    'The user should have been inserted'
);
```

-:**- test/insert_use All (SQL[ansi])

test/insert_user.sql

Emacs

```
    'volatile'
);

SELECT lives_ok(
    $$ SELECT insert_user('theory', 'foo') $$,
    'Insert a user'
);

SELECT row_eq(
    'SELECT * FROM users',
    ROW('theory', md5('foo'), NOW())::users,
    'The user should have been inserted'
);
```

-:-- test/insert_use   All   (SQL[ansi])

test/insert_user.sql

Emacs

```
    'The user should have been inserted'
);
```

-:** test/insert_use   All   (SQL[ansi])----------

test/insert_user.sql

Emacs

```
    'The user should have been inserted'
);
SELECT lives_ok(
    $$ SELECT insert_user('strongrrl', 'w00t') $$,
    'Insert another user'
);
SELECT bag_eq(
    'SELECT * FROM users',
    $$ VALUES
        ('theory',    md5('foo'),  NOW()),
        ('strongrrl', md5('w00t'), NOW())
    $$,
    'Both users should be present'
);
```

-:**- test/insert_use All (SQL[ansi])----------

test/insert_user.sql

Emacs

```
    'The user should have been inserted'
);
SELECT lives_ok(
    $$ SELECT insert_user('strongrrl', 'w00t') $$,
    'Insert another user'
);
SELECT bag_eq(
    'SELECT * FROM users',
    $$ VALUES
        ('theory',    md5('foo'),  NOW()),
        ('strongrrl', md5('w00t'), NOW())
    $$,
    'Both users should be present'
);
```

-:**- test/insert_use All (SQL[ansi])----------

test/insert_user.sql

Emacs

```
    'The user should have been inserted'
);
SELECT lives_ok(
    $$ SELECT inse                trongrrl', 'w00t') $$,
    'Insert anot
);
SELECT bag_eq(
    'SELECT * FROM users',
    $$ VALUES
        ('theory',    md5('foo'),  NOW()),
        ('strongrrl', md5('w00t'), NOW())
    $$,
    'Both users should be present'
);
```

Includes dupes

test/insert_use All (SQL[ansi])

test/insert_user.sql

Emacs

```
    'The user should have been inserted'
);
SELECT lives_ok(
    $$ SELECT insert_user('strongrrl', 'w00t') $$,
    'Insert another user'
);
SELECT bag_eq(
    'SELECT * FROM users',
    $$ VALUES
        ('theory',    md5('foo'),  NOW()),
        ('strongrrl', md5('w00t'), NOW())
    $$,
    'Both users should be present'
);
```

-:**- test/insert_use All (SQL[ansi])----------

test/insert_user.sql

Emacs

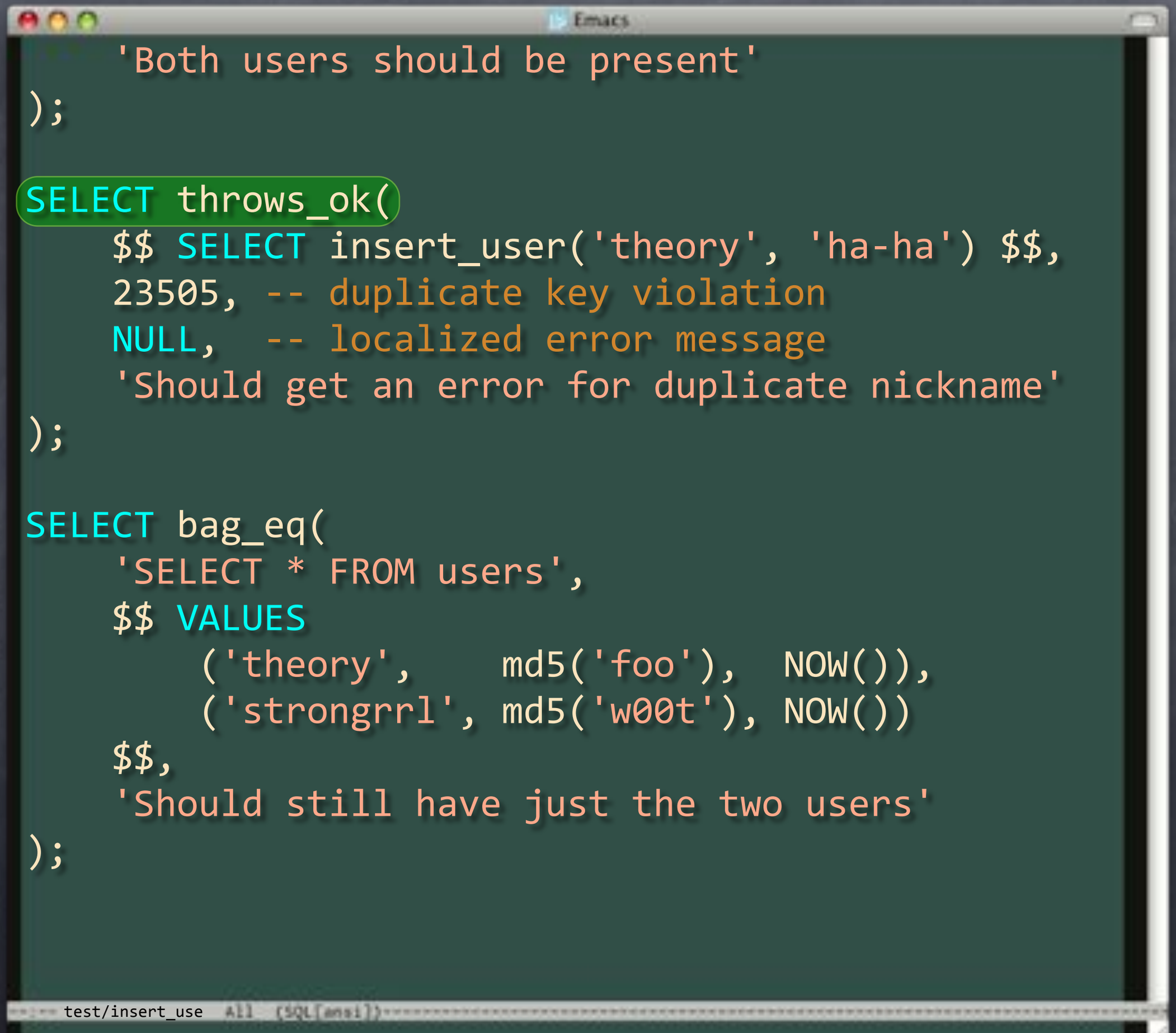
```
    'The user should have been inserted'
);
SELECT lives_ok(
    $$ SELECT insert_user('strongrrl', 'w00t') $$,
    'Insert another user'
);
SELECT bag_eq(
    'SELECT * FROM users',
    $$ VALUES
        ('theory',    md5('foo'),  NOW()),
        ('strongrrl', md5('w00t'), NOW())
    $$,
    'Both users should be present'
);
```

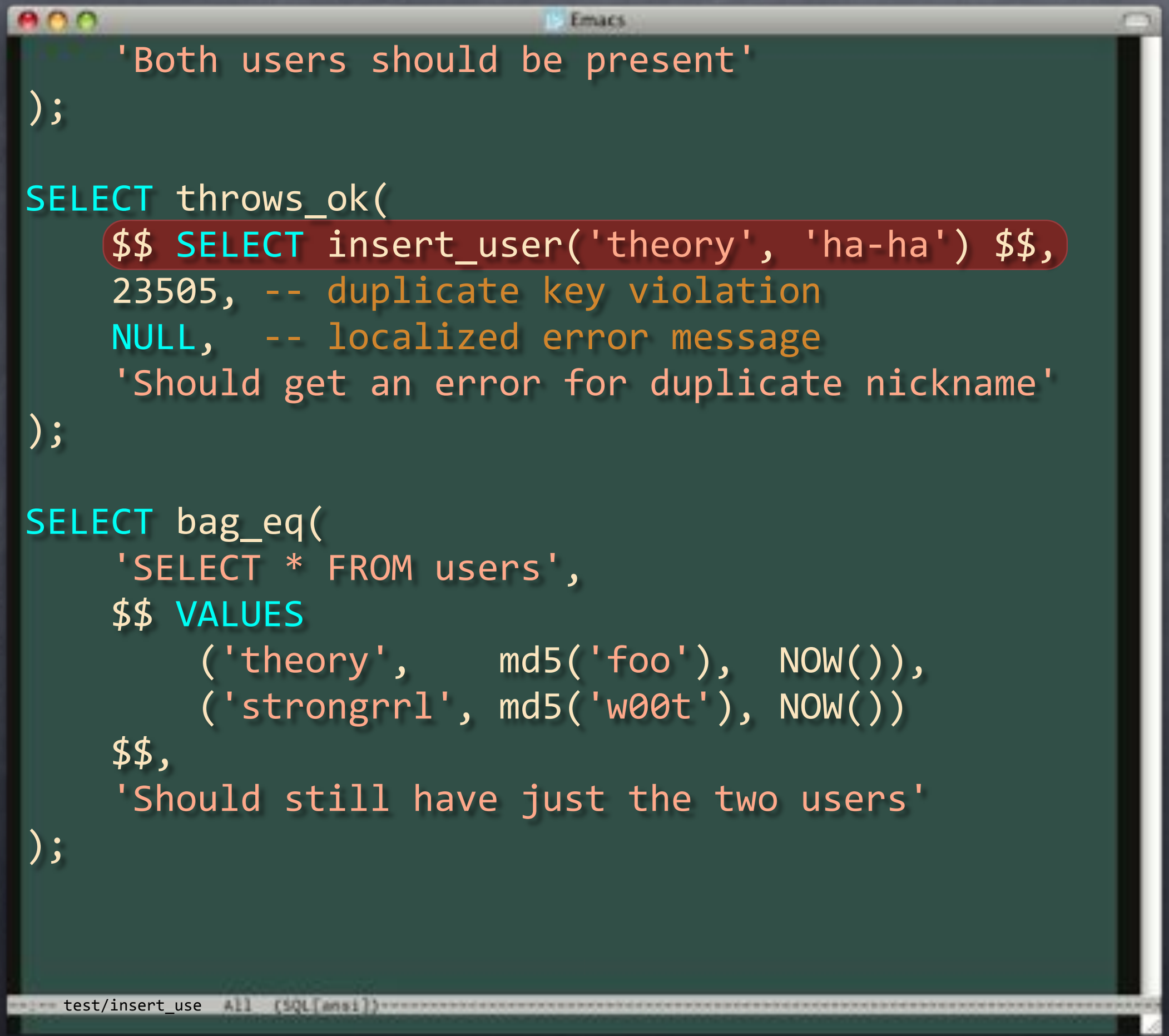
-:**- test/insert_use   All   (SQL[ansi])----------

Emacs

```
    'Both users should be present'
);
```

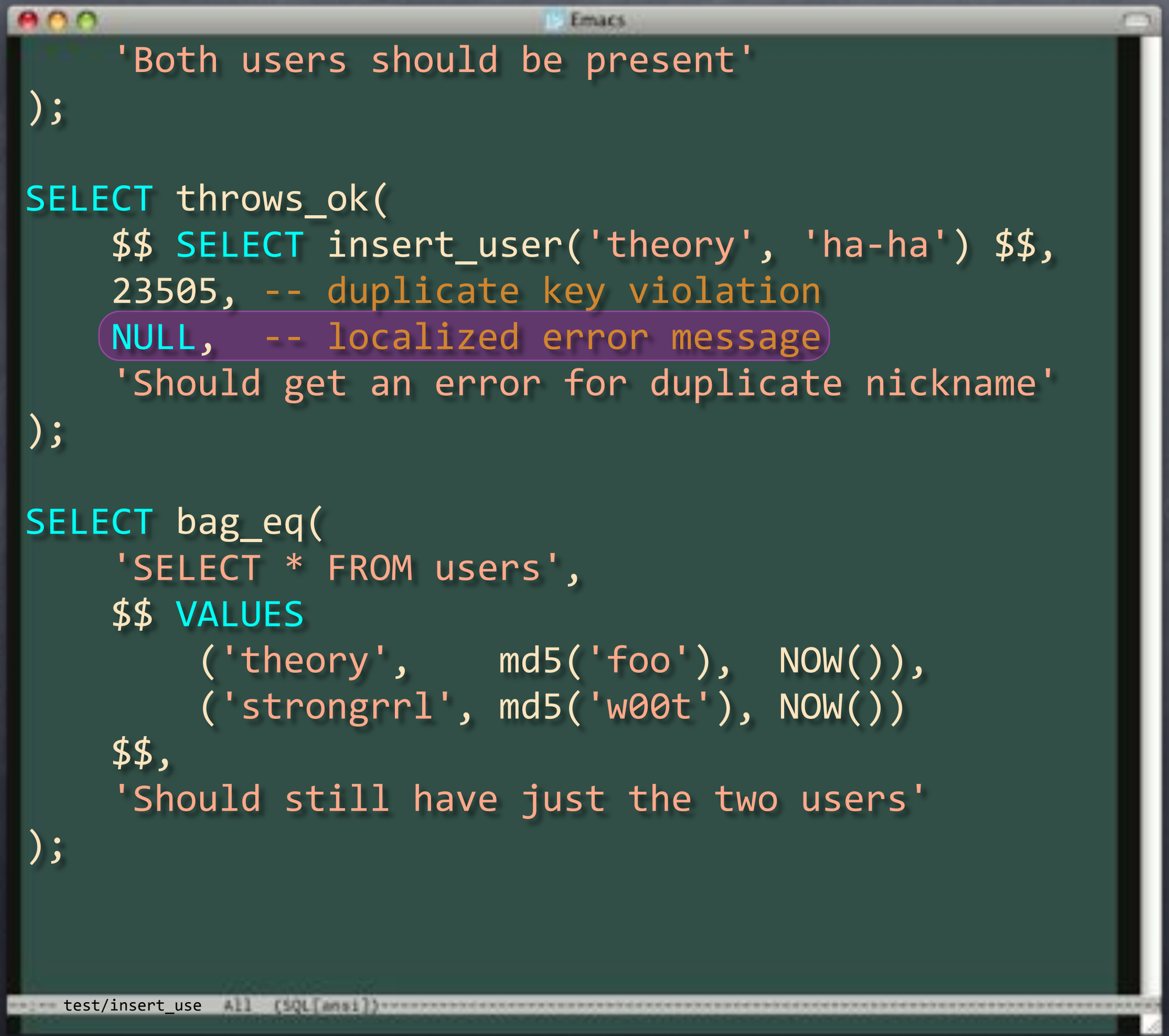
-:**- test/insert_use   All   (SQL[ansi])----------

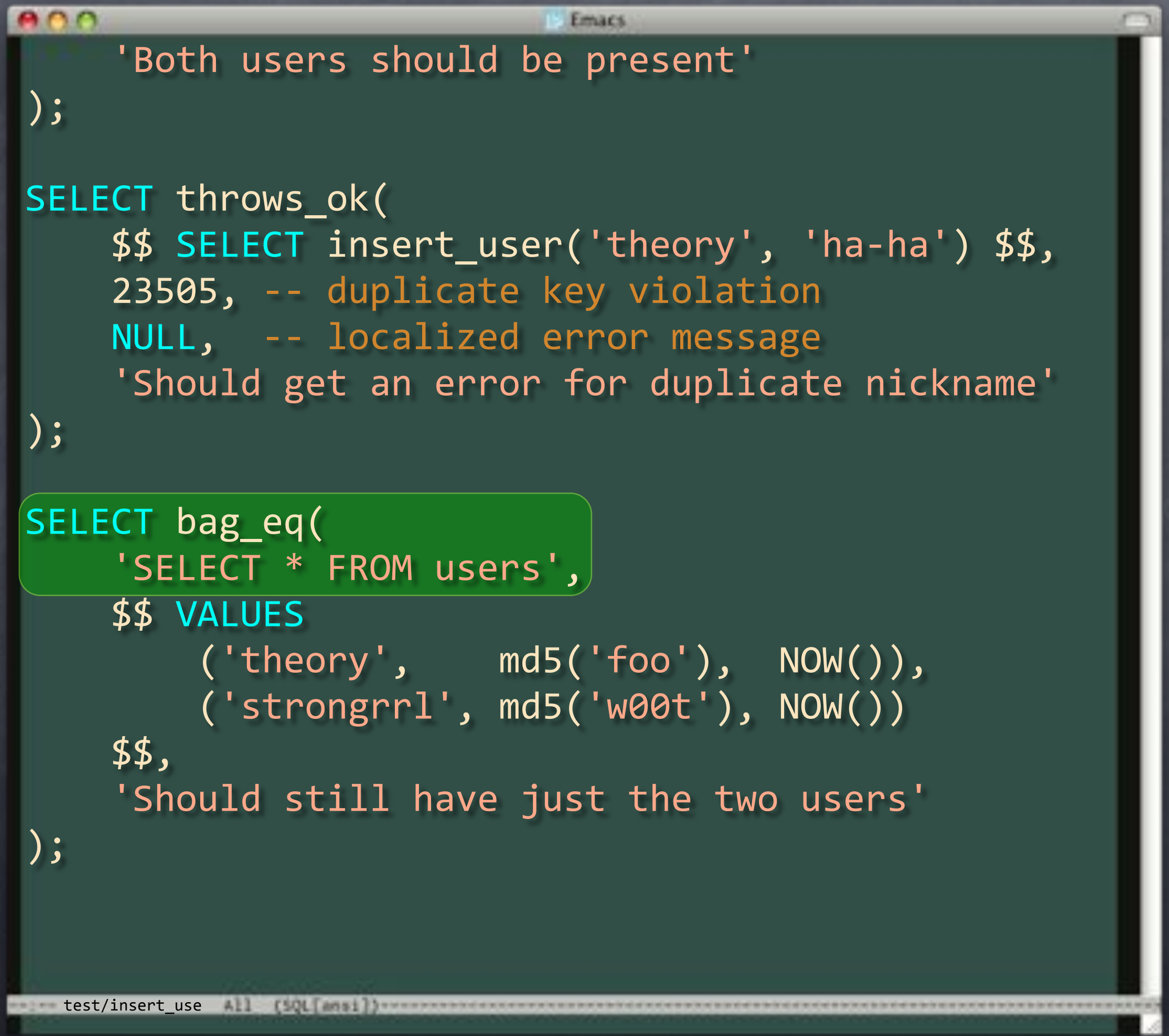

Emacs
'Both users should be present'
);
SELECT throws_ok(
$$ SELECT insert_user('theory', 'ha-ha') $$,
23505, -- duplicate key violation
NULL, -- localized error message
'Should get an error for duplicate nickname'
);
SELECT bag_eq(
'SELECT * FROM users',
$$ VALUES
('theory', md5('foo'), NOW()),
('strongrrl', md5('w00t'), NOW())
$$,
'Should still have just the two users'
);
test/insert_use All (SQL[ansi])

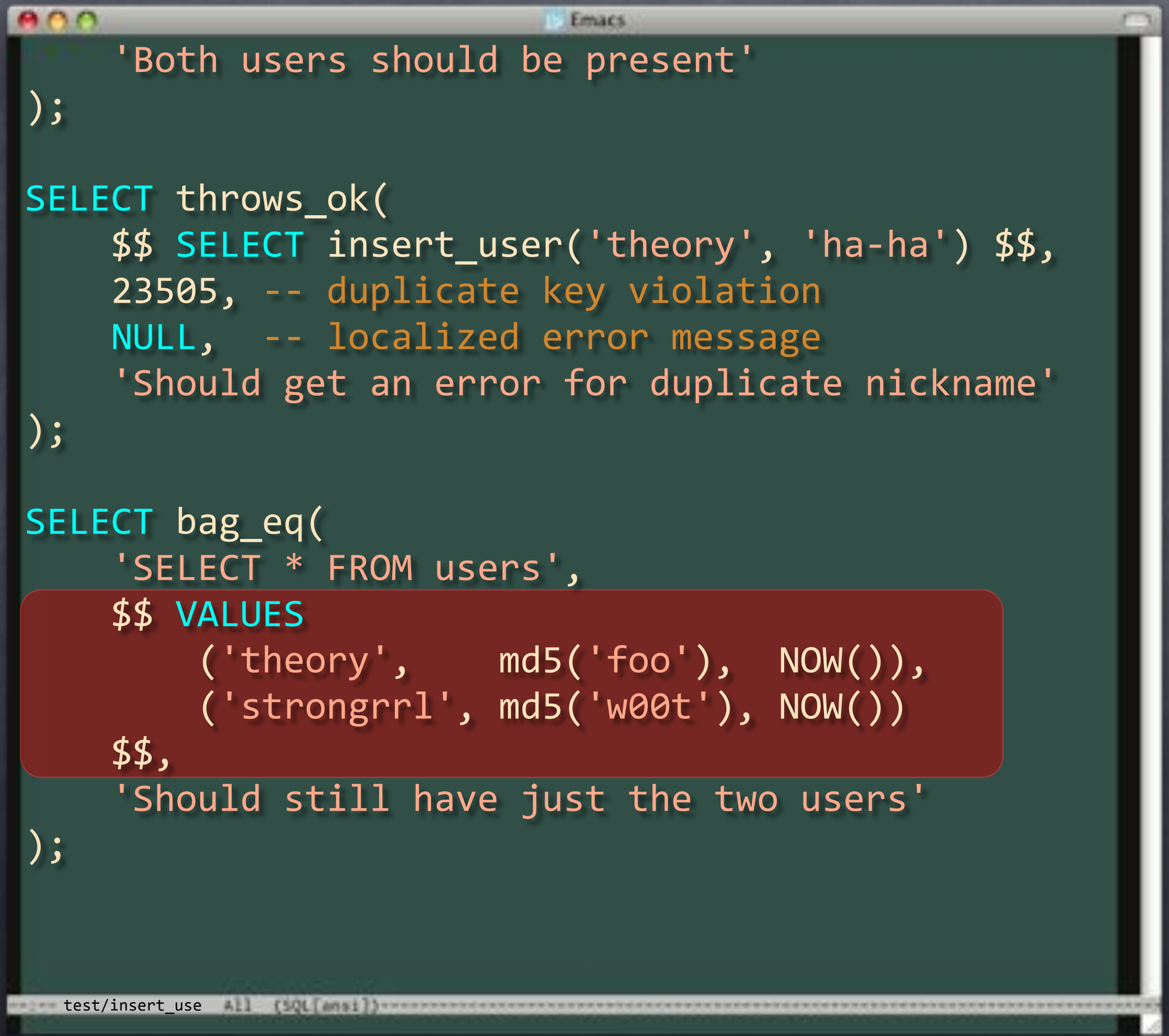

Emacs
'Both users should be present'
);
SELECT throws_ok(
$$ SELECT insert_user('theory', 'ha-ha') $$,
23505, -- duplicate key violation
NULL, -- localized error message
'Should get an error for duplicate nickname'
);
SELECT bag_eq(
'SELECT * FROM users',
$$ VALUES
('theory', md5('foo'), NOW()),
('strongrrl', md5('w00t'), NOW())
$$,
'Should still have just the two users'
);
test/insert_use All (SQL[ansi])

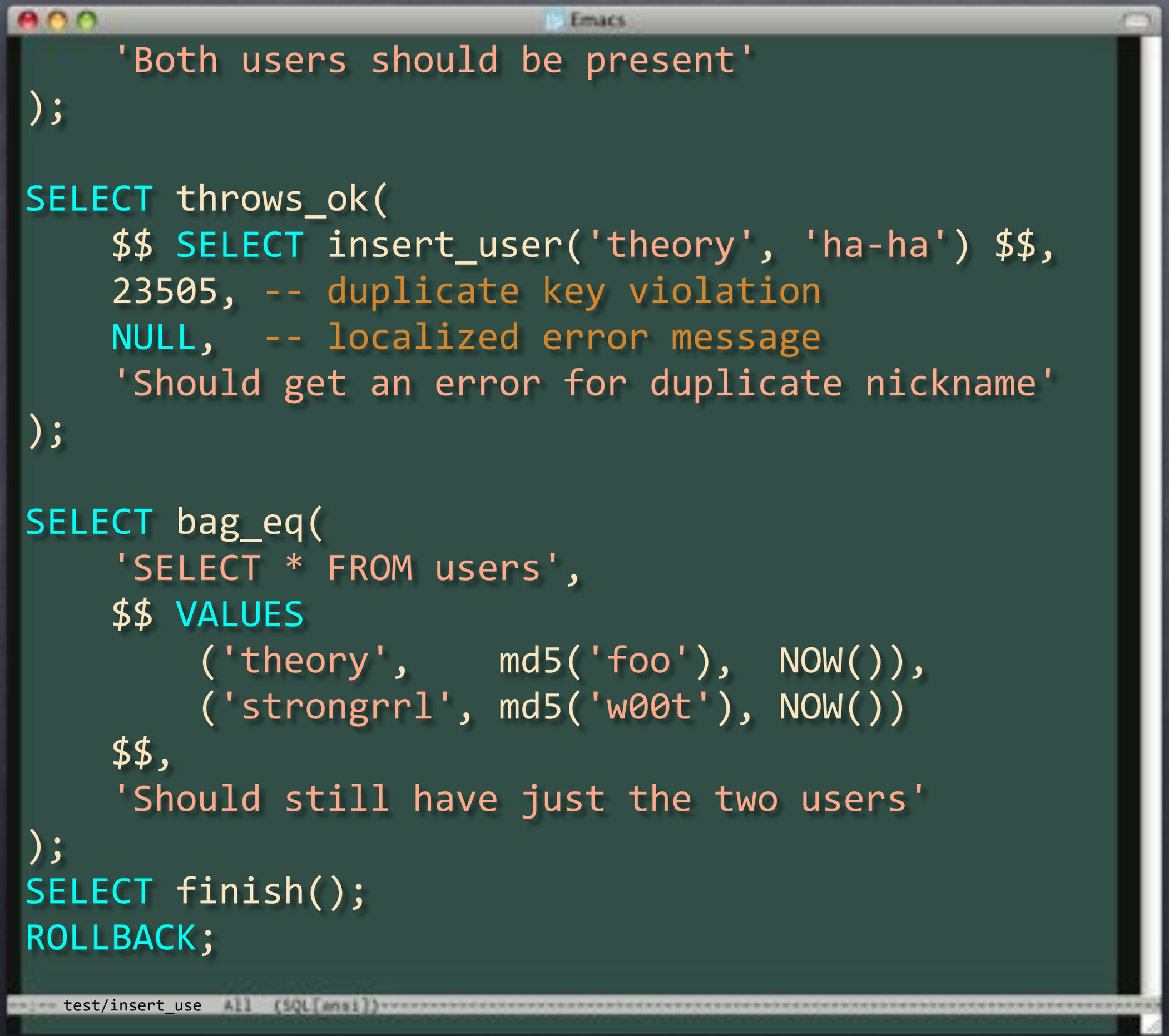
```
    'Both users should be present'
);

SELECT throws_ok(
    $$ SELECT insert_user('theory', 'ha
    23505, -- duplicate key violation
    NULL,  -- localized error message
    'Should get an error for duplicate nickname'
);

SELECT bag_eq(
    'SELECT * FROM users',
    $$ VALUES
        ('theory',    md5('foo'),  NOW()),
        ('strongrrl', md5('w00t'), NOW())
    $$,
    'Should still have just the two users'
);
```

Appendix A

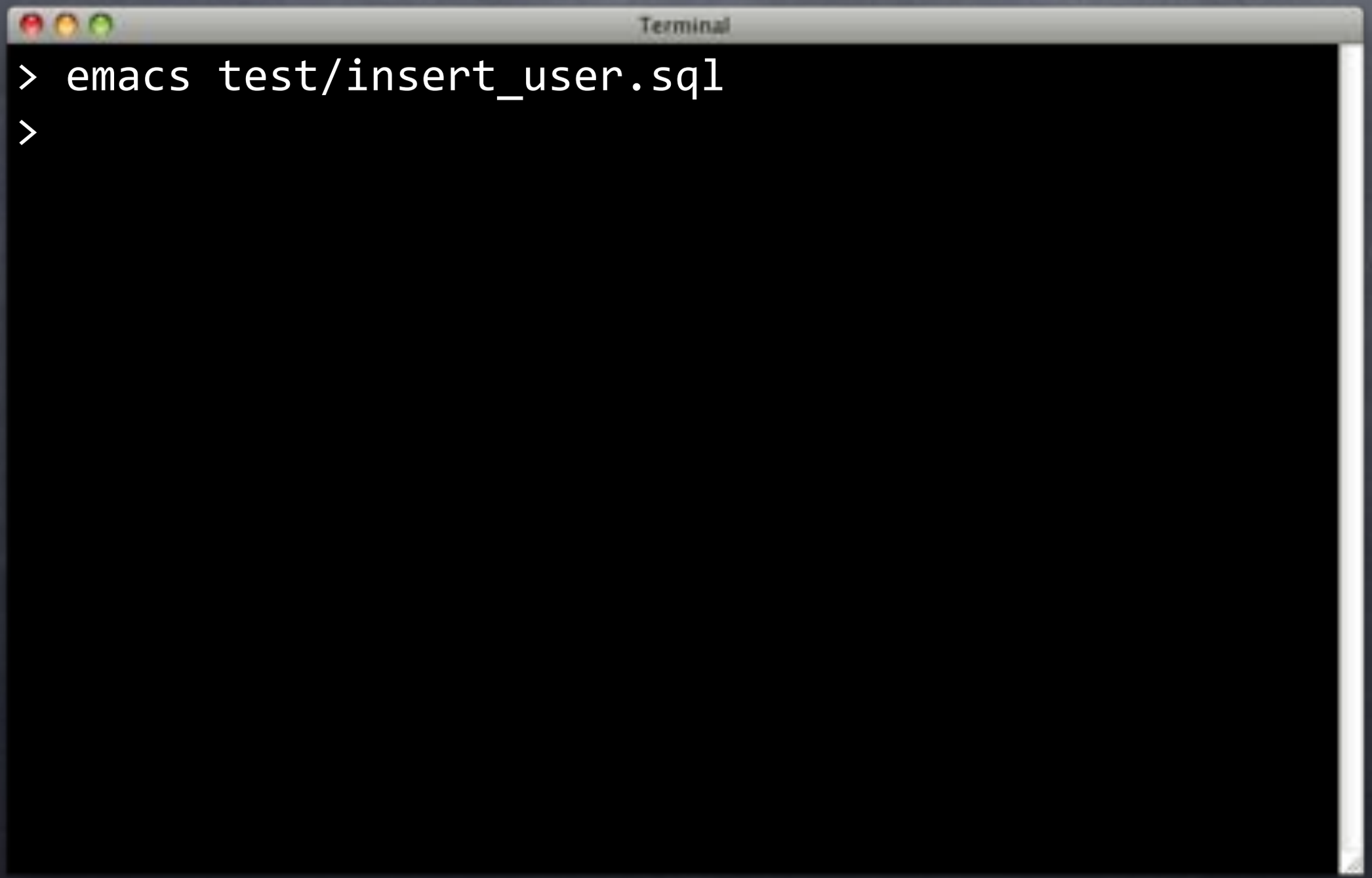

Emacs
'Both users should be present'
);
SELECT throws_ok(
$$ SELECT insert_user('theory', 'ha-ha') $$,
23505, -- duplicate key violation
NULL, -- localized error message
'Should get an error for duplicate nickname'
);
SELECT bag_eq(
'SELECT * FROM users',
$$ VALUES
('theory', md5('foo'), NOW()),
('strongrrl', md5('w00t'), NOW())
$$,
'Should still have just the two users'
);
test/insert_use All (SQL[ansi])


Emacs
'Both users should be present'
);
SELECT throws_ok(
$$ SELECT insert_user('theory', 'ha-ha') $$,
23505, -- duplicate key violation
NULL, -- localized error message
'Should get an error for duplicate nickname'
);
SELECT bag_eq(
'SELECT * FROM users',
$$ VALUES
('theory', md5('foo'), NOW()),
('strongrrl', md5('w00t'), NOW())
$$,
'Should still have just the two users'
);
test/insert_use All (SQL[ansi])

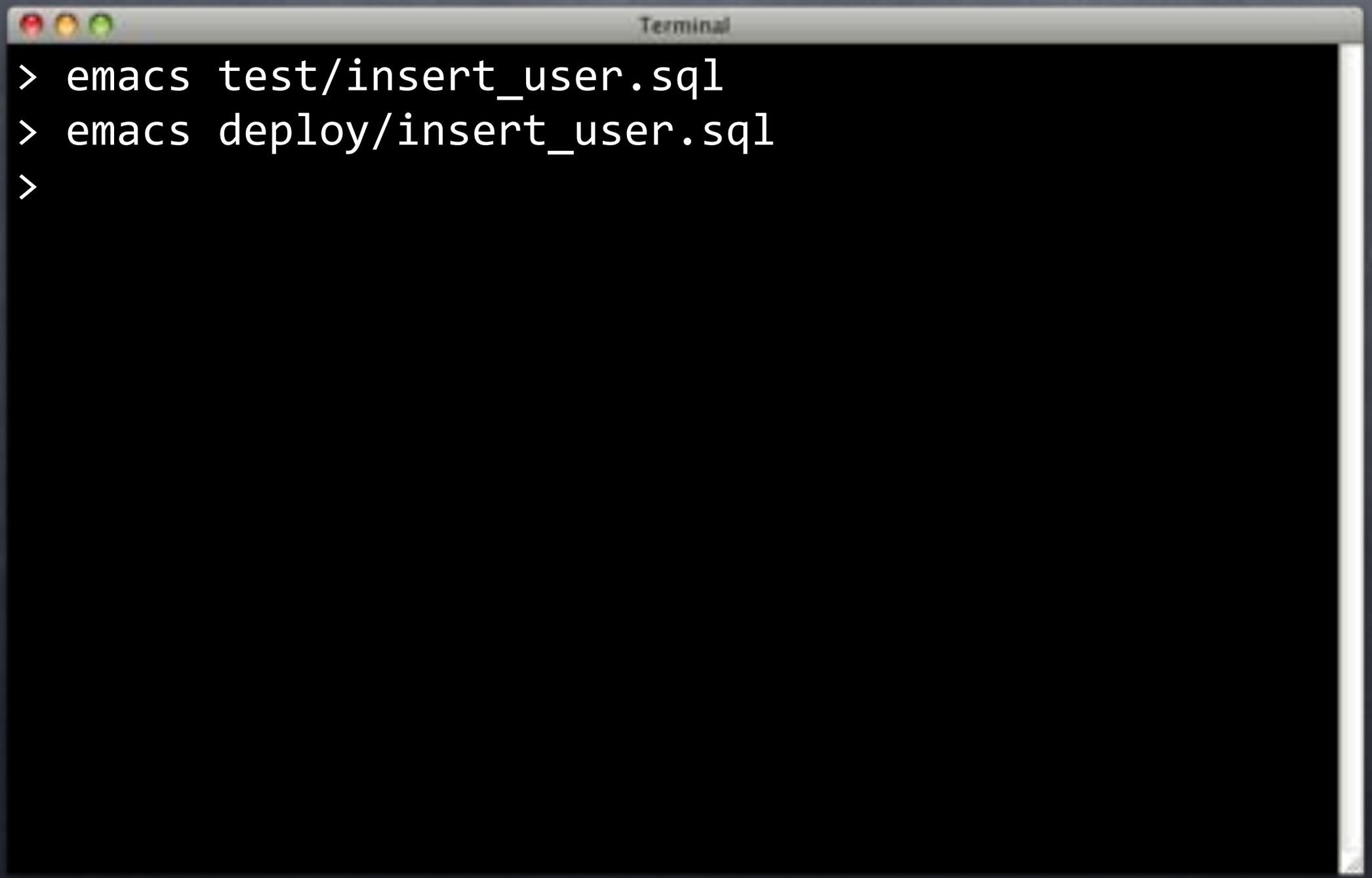

Emacs
'Both users should be present'
);
SELECT throws_ok(
$$ SELECT insert_user('theory', 'ha-ha') $$,
23505, -- duplicate key violation
NULL, -- localized error message
'Should get an error for duplicate nickname'
);
SELECT bag_eq(
'SELECT * FROM users',
$$ VALUES
('theory', md5('foo'), NOW()),
('strongrrl', md5('w00t'), NOW())
$$,
'Should still have just the two users'
);
test/insert_use All (SQL[ansi])


Emacs
'Both users should be present'
);
SELECT throws_ok(
$$ SELECT insert_user('theory', 'ha-ha') $$,
23505, -- duplicate key violation
NULL, -- localized error message
'Should get an error for duplicate nickname'
);
SELECT bag_eq(
'SELECT * FROM users',
$$ VALUES
('theory', md5('foo'), NOW()),
('strongrrl', md5('w00t'), NOW())
$$,
'Should still have just the two users'
);
SELECT finish();
ROLLBACK;
test/insert_use All (SQL[ansi])

Functional Testing



Functional Testing



deploy/insert_user.sql

Emacs

```
-- Deploy insert_user
-- requires: users
-- requires: appschema

BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

-:** deploy/insert_u All (SQL[ansi])

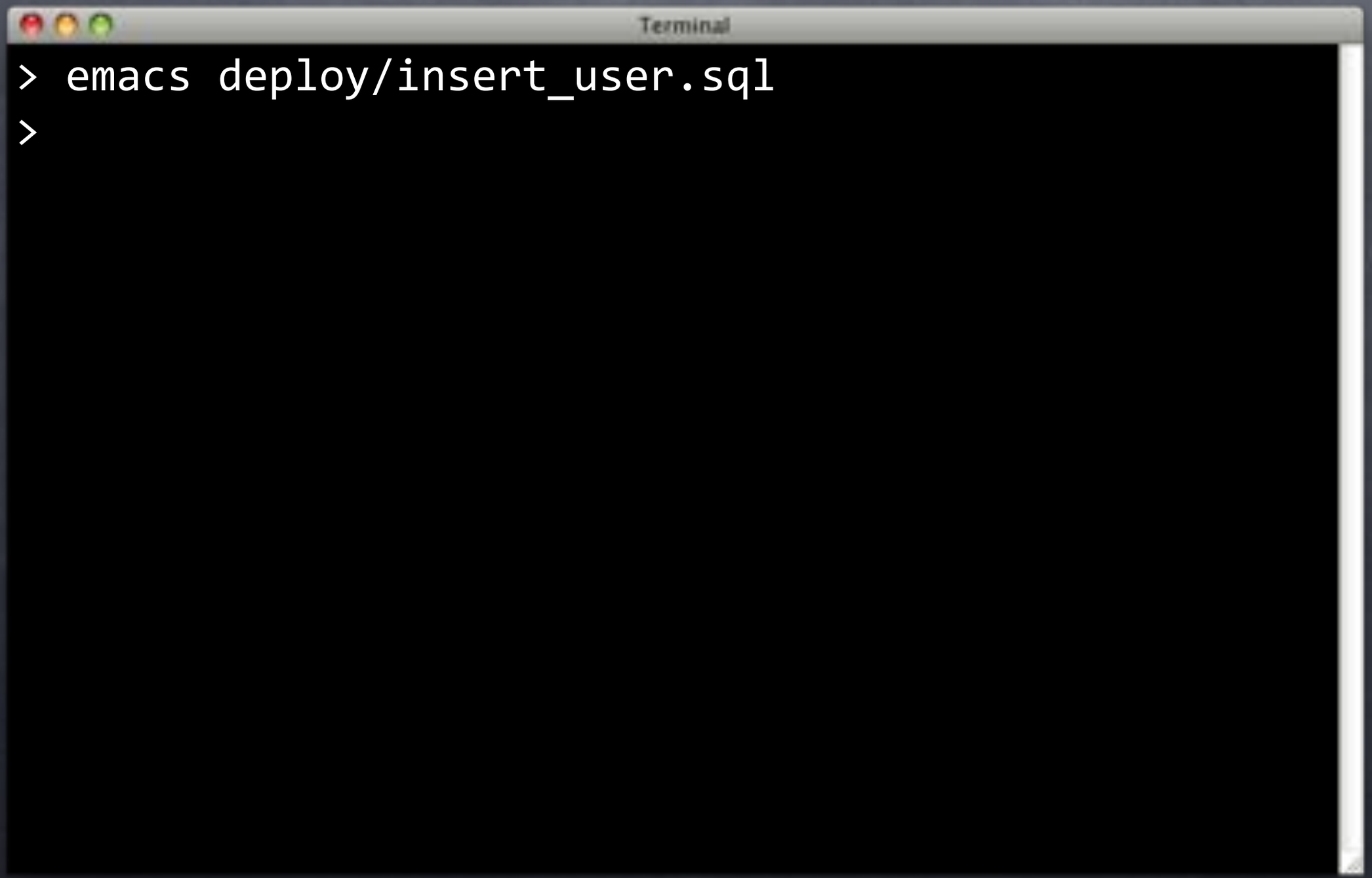
deploy/insert_user.sql

Emacs

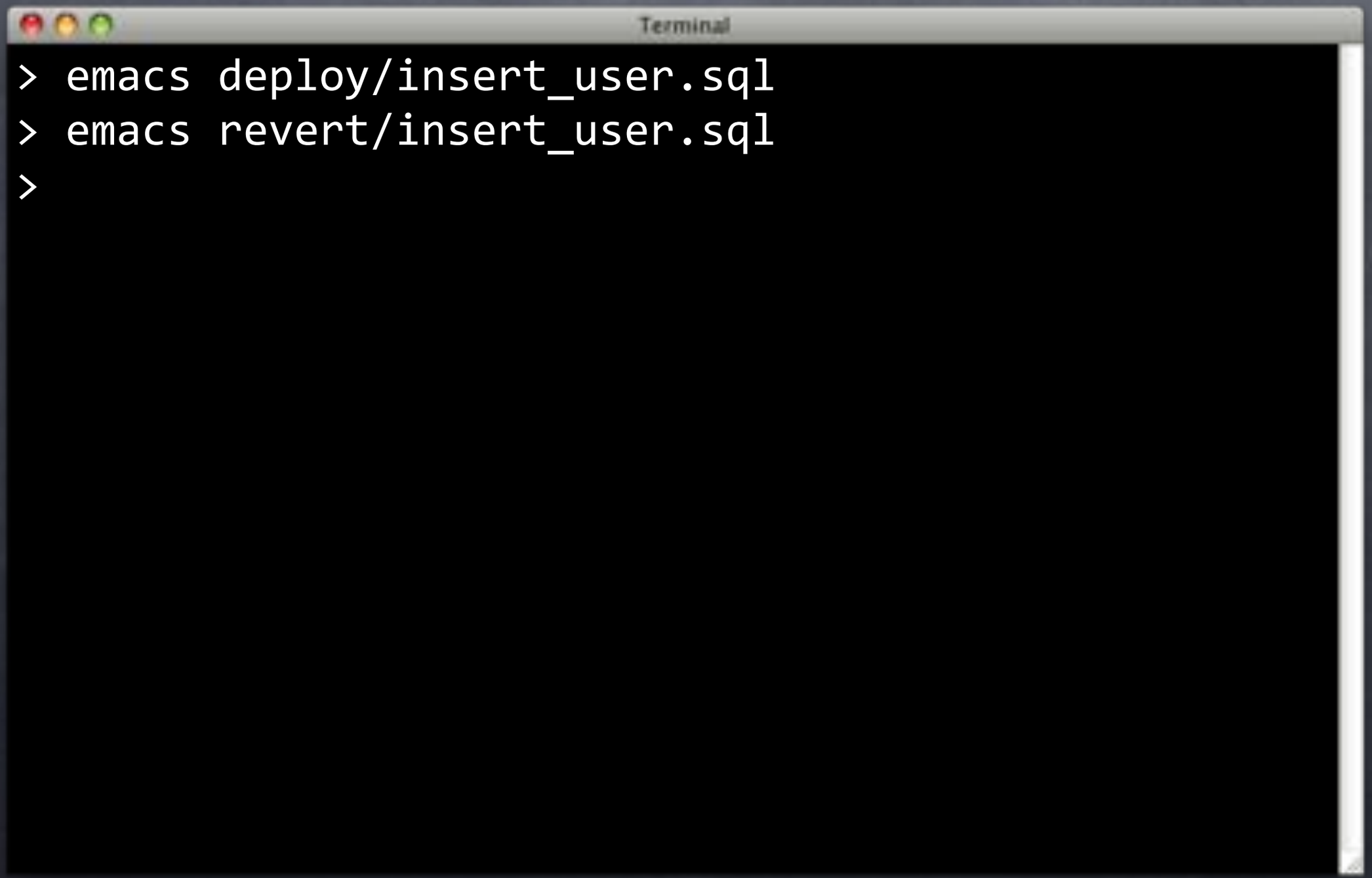
```
-- Deploy insert_user
-- requires: users
-- requires: appschema

BEGIN;

CREATE OR REPLACE FUNCTION flipr.insert_user(
    nickname TEXT,
    password TEXT
) RETURNS VOID LANGUAGE SQL SECURITY DEFINER AS $$
    INSERT INTO flipr.users VALUES($1, md5($2));
$$;

COMMIT;
```

-:** deploy/insert_u All (SQL[ansi])

Functional Testing



Functional Testing



revert/insert_user.sql

Emacs

```
-- Revert insert_user

BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

```
-:**-  revert/insert_u   All   (SQL[ansi])----------------------------------------------------------------------------
```

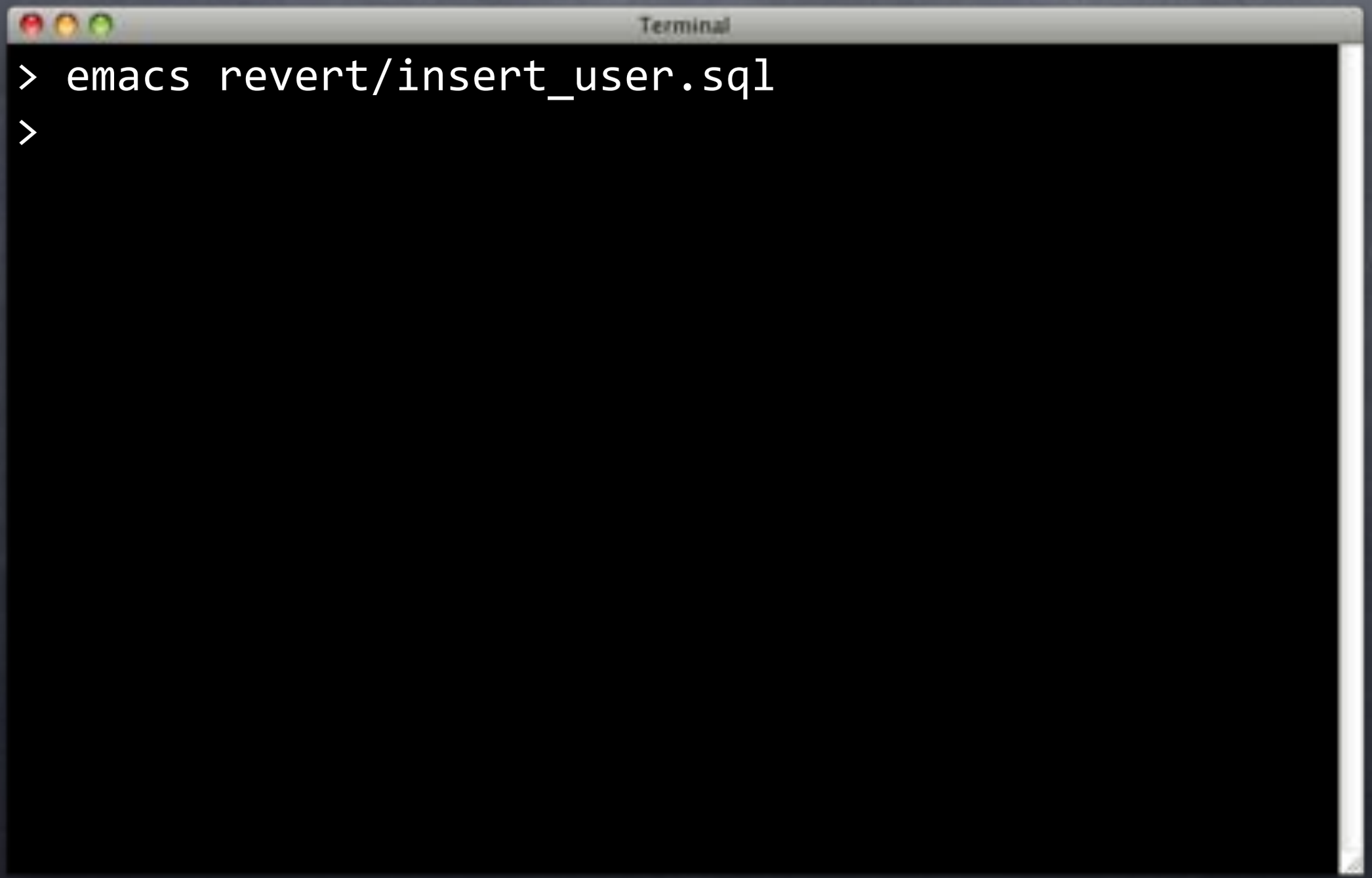
revert/insert_user.sql

Emacs

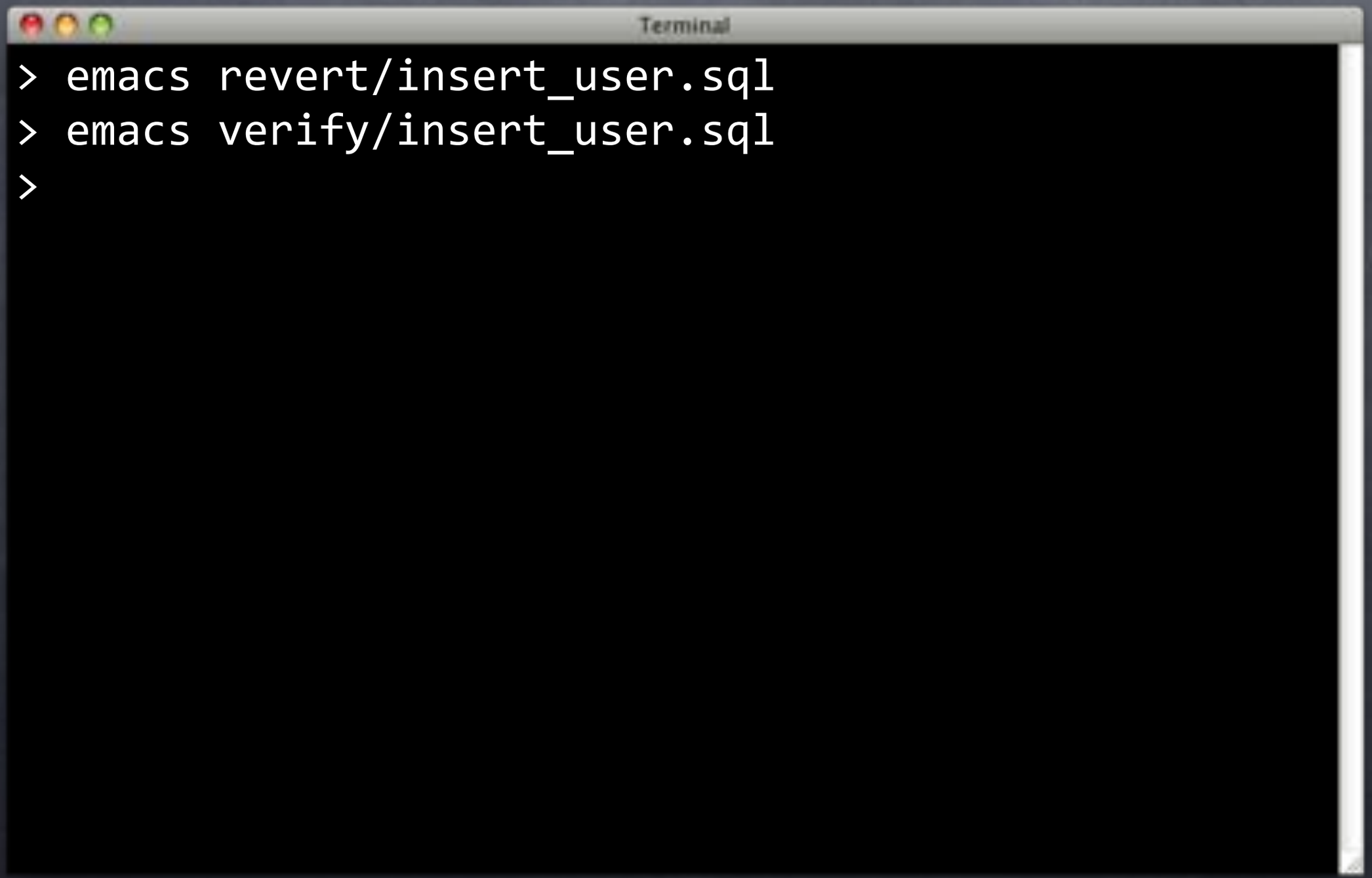
```
-- Revert insert_user

BEGIN;

DROP FUNCTION flipr.insert_user(TEXT, TEXT);

COMMIT;
```

-:** revert/insert_u All (SQL[ansi])

Functional Testing



Functional Testing



verify/insert_user.sql

Emacs

```
-- Verify insert_user

BEGIN;

-- XXX Add DDLs here.

ROLLBACK;
```

-:** verify/insert_u All (SQL[ansi])----------

verify/insert_user.sql

Emacs

```
-- Verify insert_user

BEGIN;

SELECT has_function_privilege(
    'flipr.insert_user(text, text)',
    'execute'
);

ROLLBACK;
```

-:** verify/insert_u All (SQL[ansi])

verify/insert_user.sql

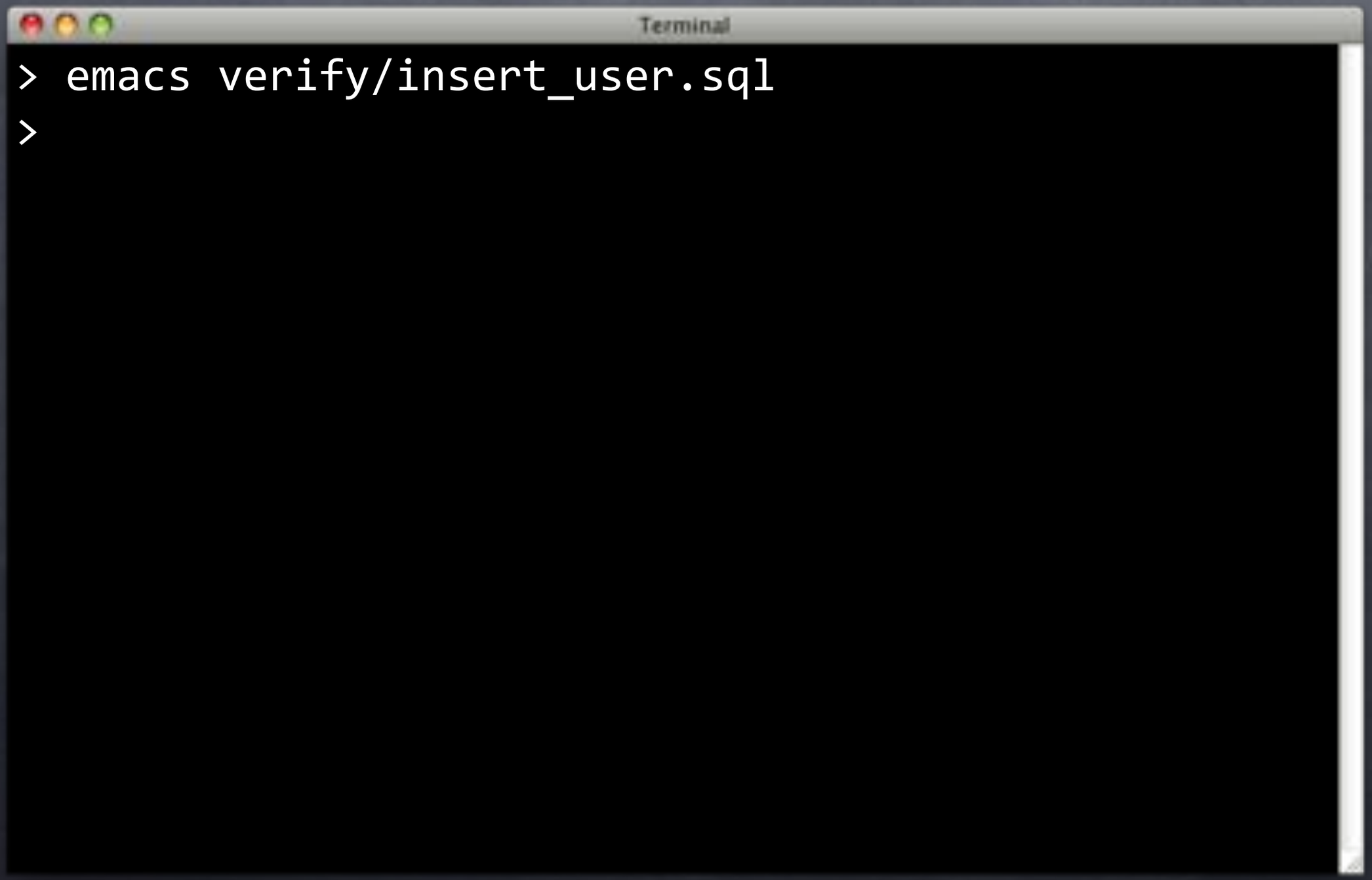
Emacs

```
-- Verify insert_user

BEGIN;

SELECT has_function_privilege(
    'flipr.insert_user(text, text)',
    'execute'
);

ROLLBACK;
```

Convenient!

-:** verify/insert_u All (SQL[ansi])----------

We Good?

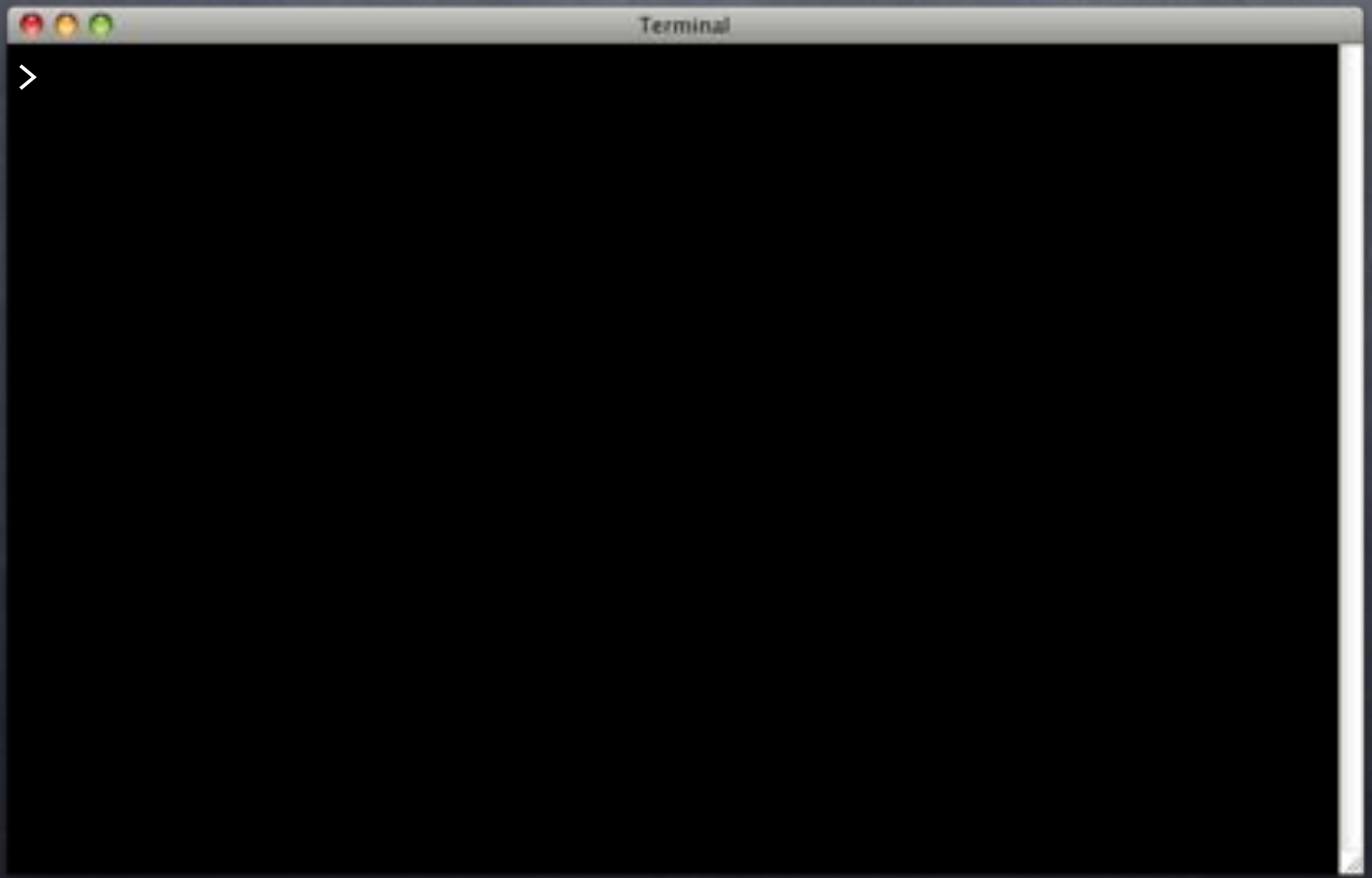


We Good?

Terminal

```
> emacs verify/insert_user.sql
> pg_prove -d flipr_test test/*.sql
test/appschema.sql .... ok
test/insert_user.sql .. ok
test/users.sql ........ ok
All tests successful.
Files=3, Tests=27,  0 wallclock secs
Result: PASS
>
```

Commitment

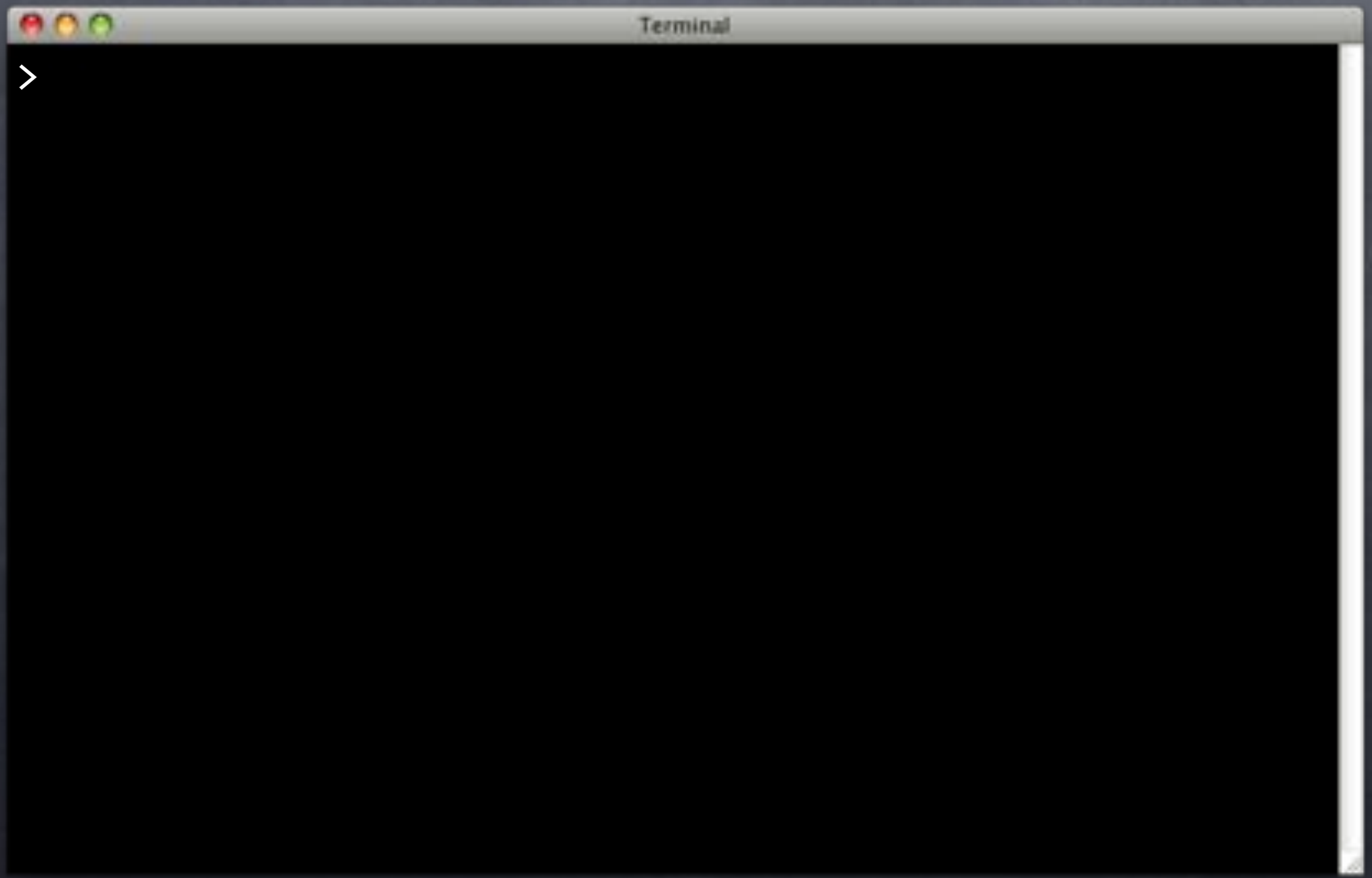


Commitment

Terminal

```
> git add .
> git commit -m 'Add `insert_user()`.'
[userfuncs 40eabe1] Add `insert_user()`.
 6 files changed, 101 insertions(+)
 create mode 100644 -n
 create mode 100644 deploy/insert_user.sql
 create mode 100644 revert/insert_user.sql
 create mode 100644 test/insert_user.sql
 create mode 100644 verify/insert_user.sql
>
```

Push It Real Good...



Push It Real Good...

Terminal

```
> git push origin --set-upstream userfuncs
Counting objects: 18, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 1.96 KiB, done.
Total 12 (delta 1), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      userfuncs -> userfuncs
Branch userfuncs set up to track remote branch
userfuncs from origin.
>
```

Push It Real Good...

Terminal

```
> git push origin --set-upstream userfuncs
Counting objects: 18, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 1.96 KiB, done.
Total 12 (delta 1), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      userfuncs -> userfuncs
Branch userfuncs set up to track remote branch
userfuncs from origin.
>
```

Push It Real Good...

Terminal

```
> git push origin --set-upstream userfuncs
Counting objects: 18, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 1.96 KiB, done.
Total 12 (delta 1), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      userfuncs -> userfuncs
Branch userfuncs set up to track remote branch
userfuncs from origin.
>
```

Reset, Mes Amis!

Reset, Mes Amis!

- git checkout users

Reset, Mes Amis!

- git checkout users
- git reset --hard upstream/users

Reset, Mes Amis!

- git checkout users
- git reset --hard upstream/users
- git checkout -b userfuncs

Reset, Mes Amis!

- git checkout users
- git reset --hard upstream/users
- git checkout -b userfuncs
- git reset --hard insert_user

None Shall Pass

None Shall Pass

- Create change_pass()

None Shall Pass

- Create change_pass()
- Params:

None Shall Pass

- Create change_pass()
- Params:
  - nickname

None Shall Pass

- Create change_pass()
- Params:
  - nickname
  - old_pass

None Shall Pass

- Create change_pass()
- Params:
  - nickname
  - old_pass
  - new_pass

None Shall Pass

- Create change_pass()
- Params:
  - nickname
  - old_pass
  - new_pass
- Only update if old pass correct

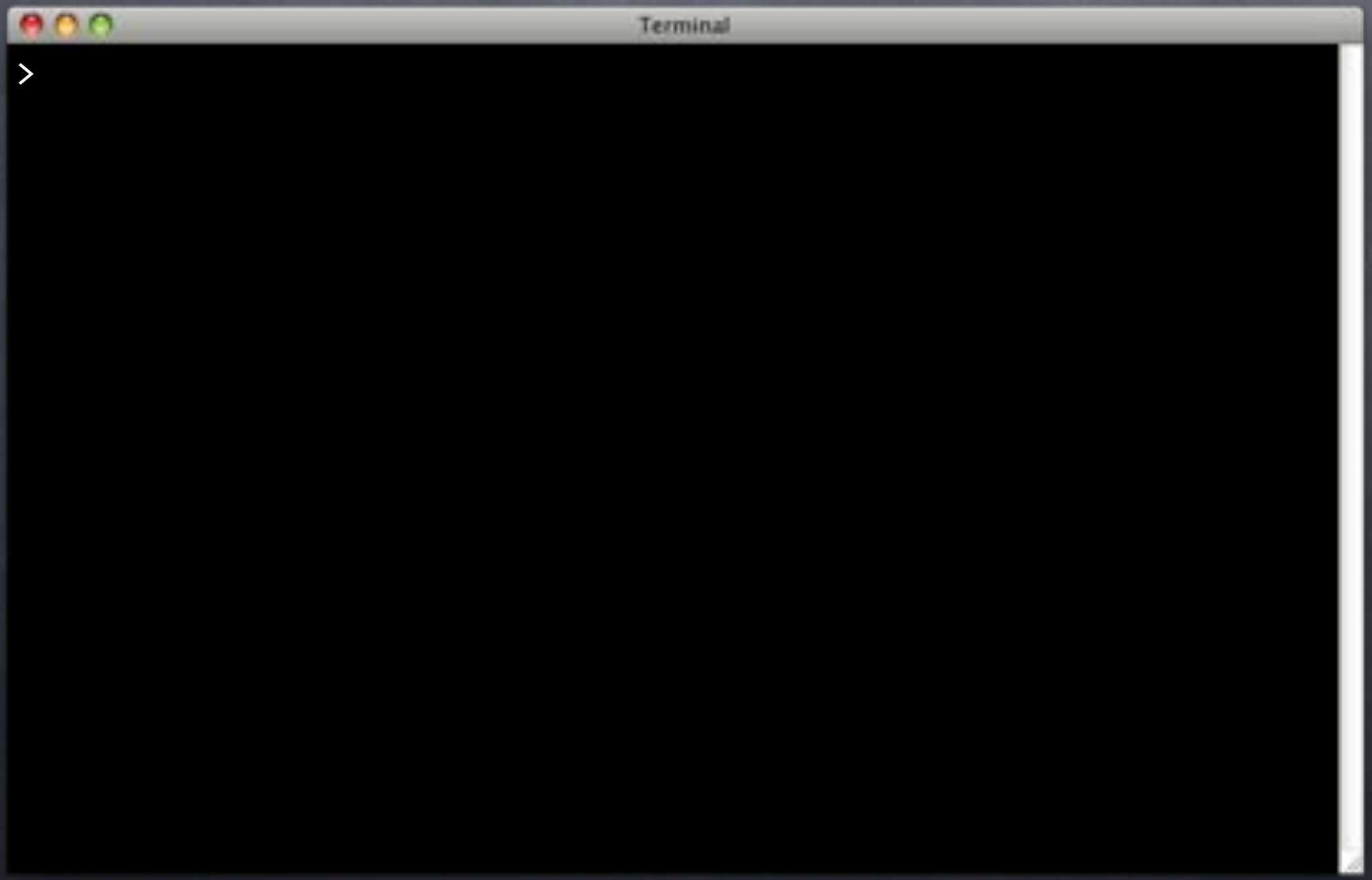
None Shall Pass

- Create change_pass()
- Params:
  - nickname
  - old_pass
  - new_pass
- Only update if old pass correct
- Use TDDD

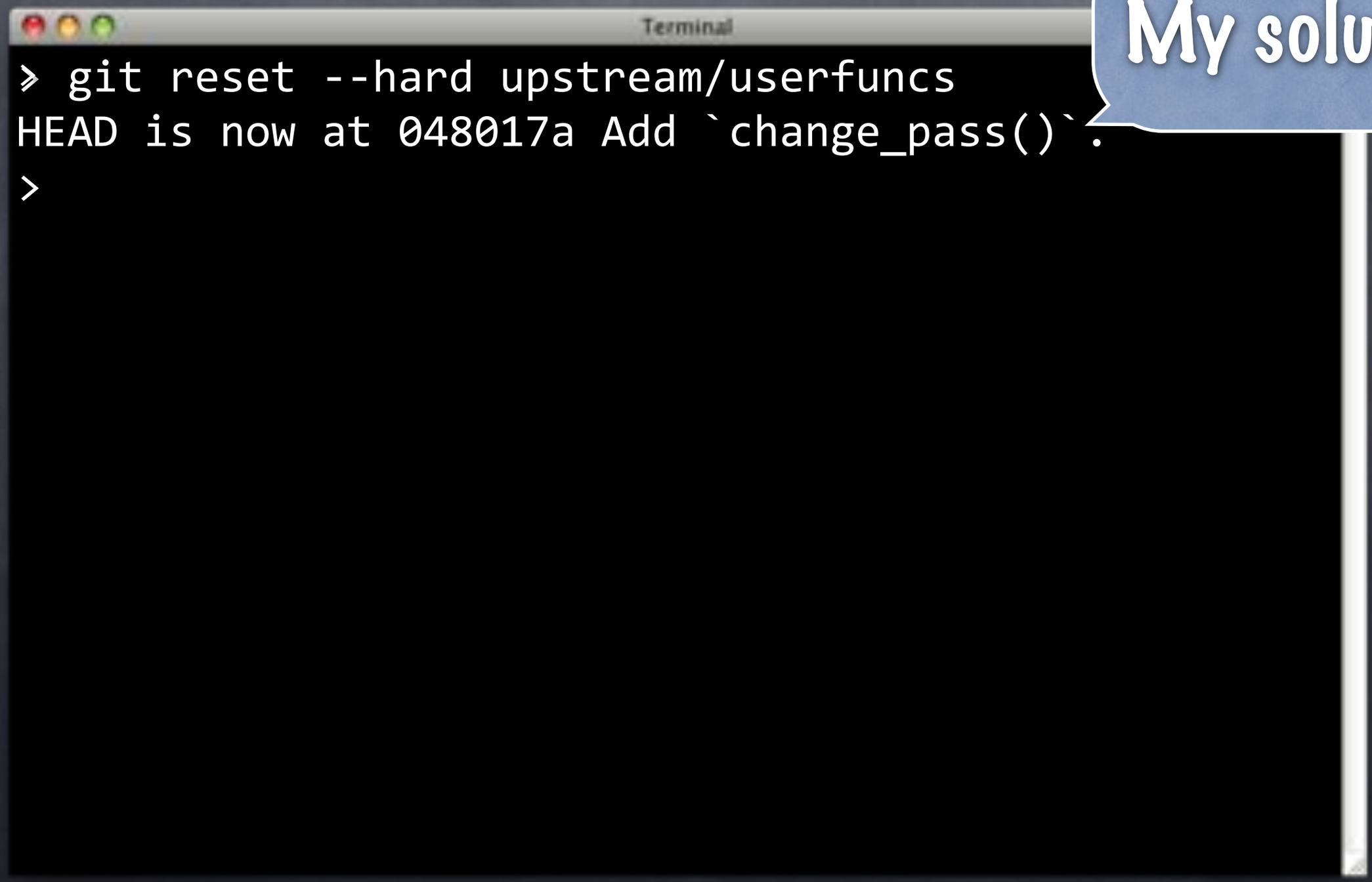
None Shall Pass

- Create change_pass()
- Params:
  - nickname
  - old_pass
  - new_pass
- Only update if old pass correct
- Use TDDD
- https://github.com/theory/agile-flipr.git

Resets

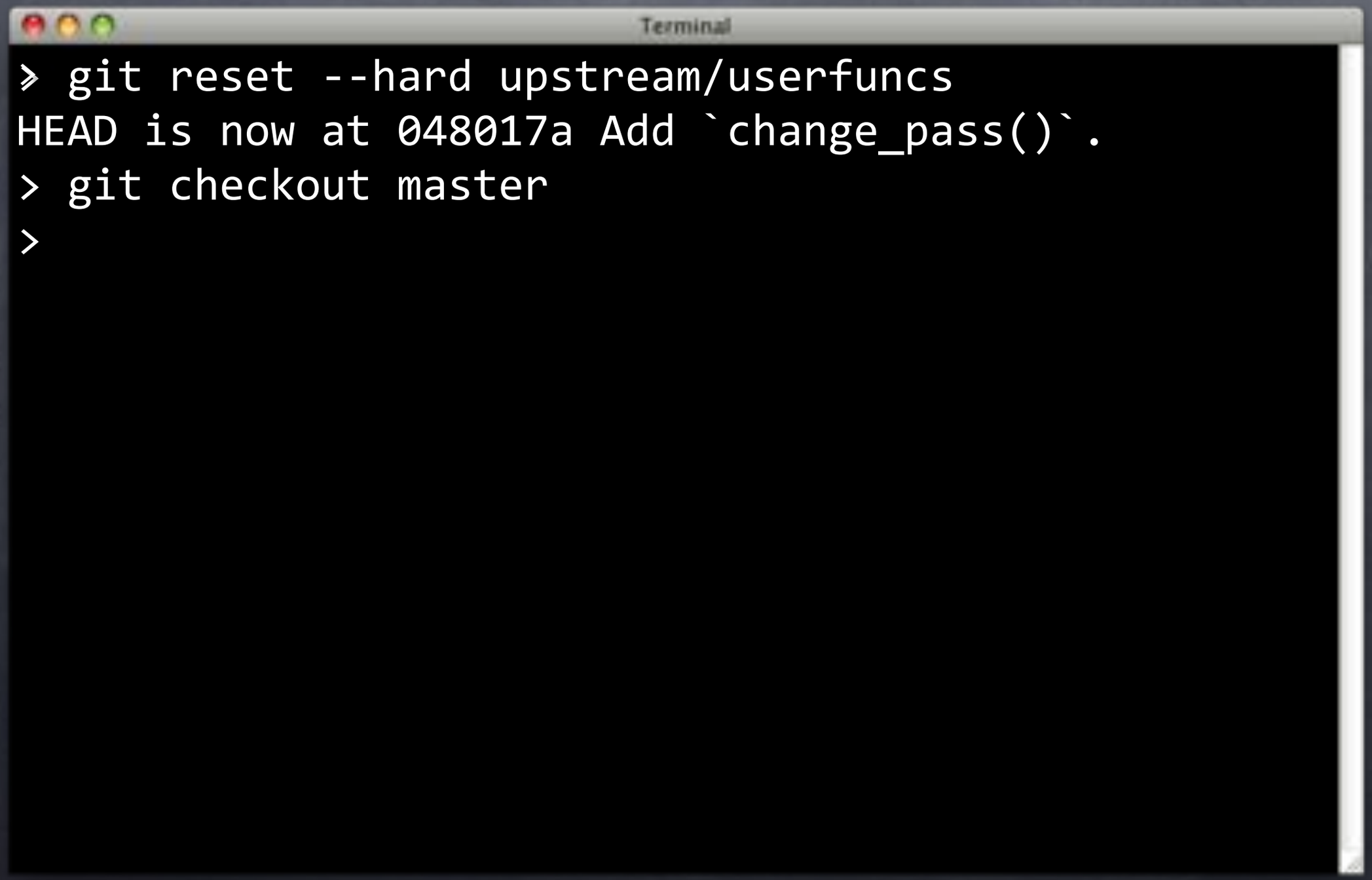


Resets

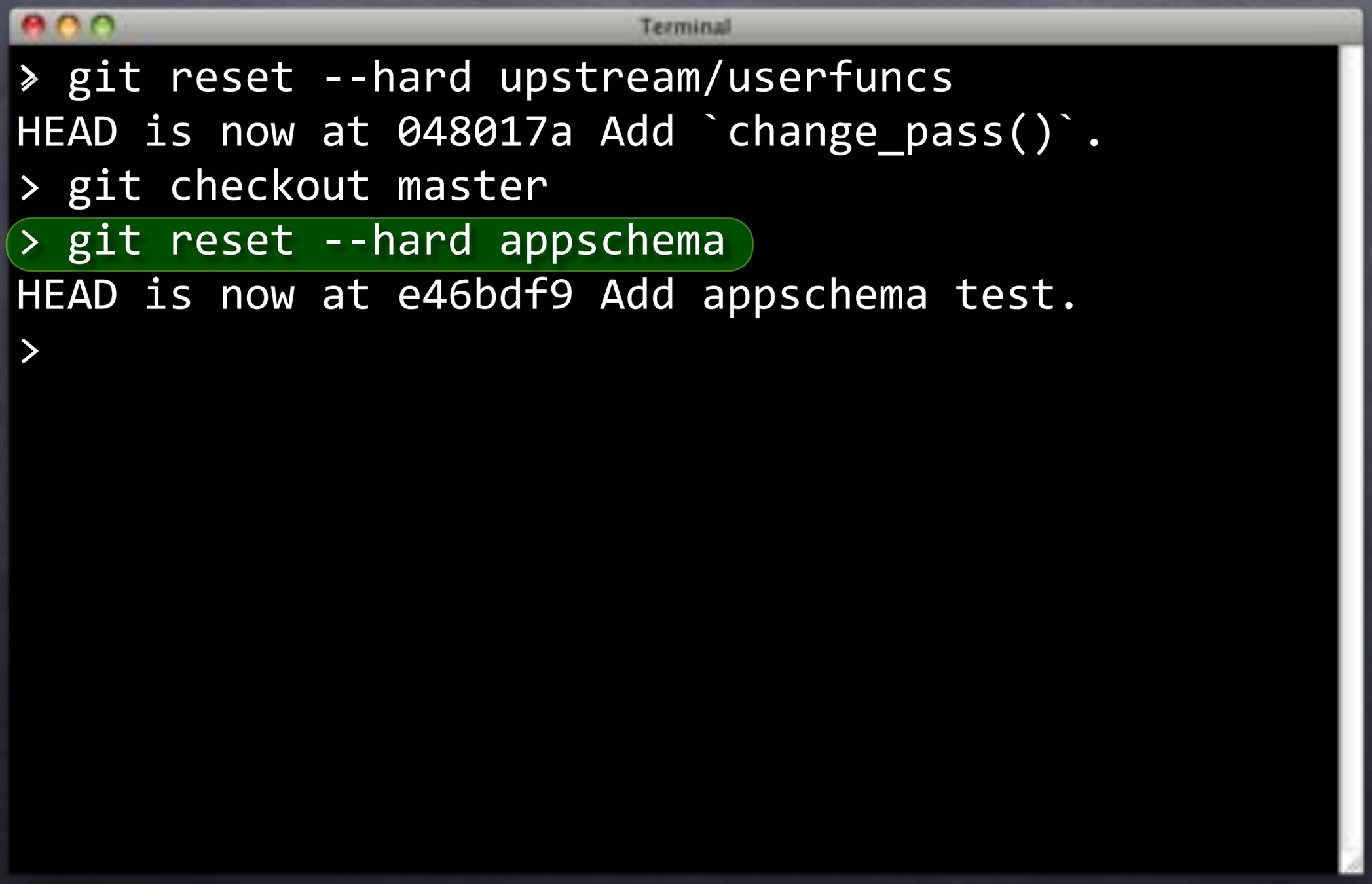


My solution

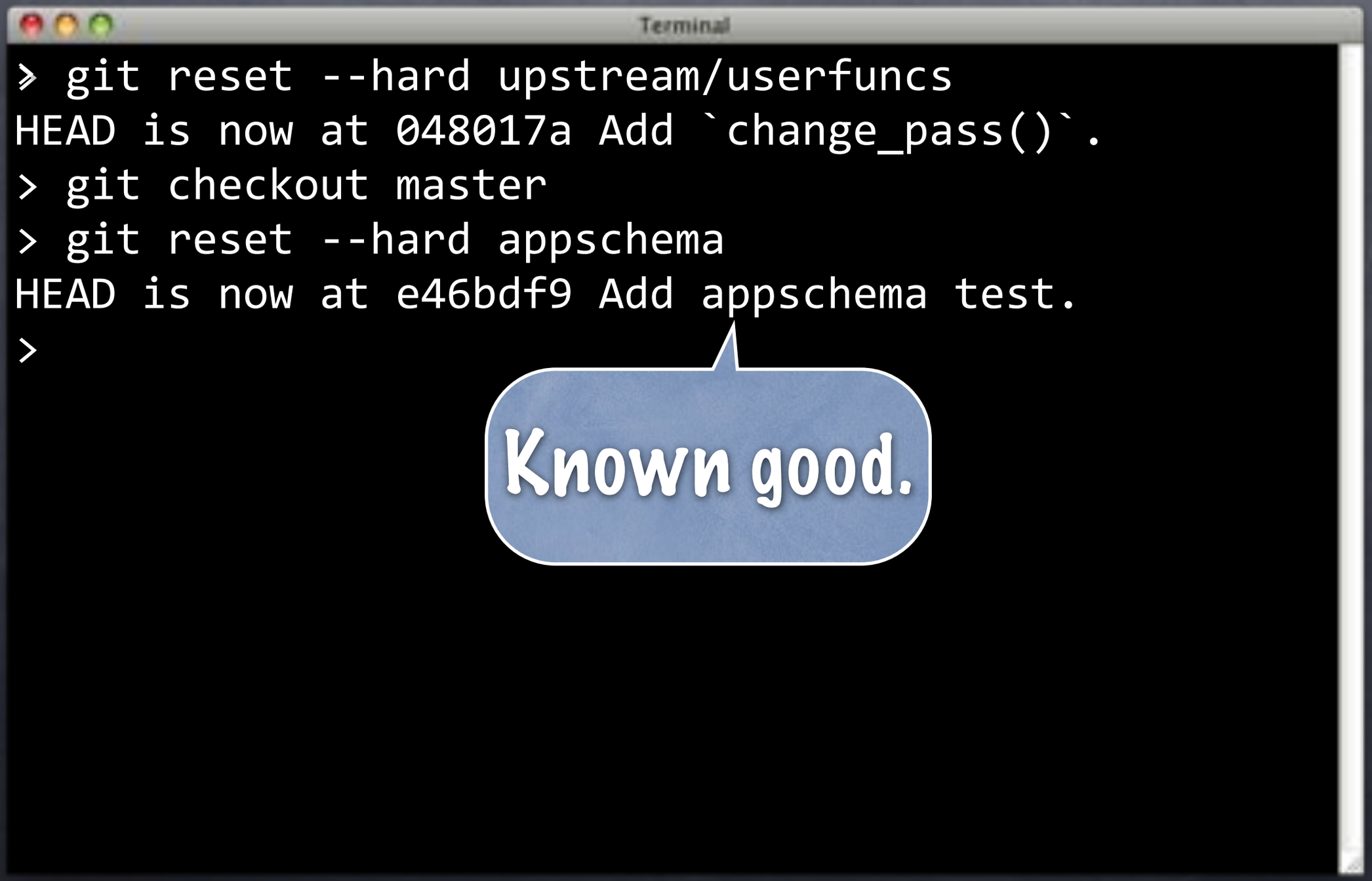
Resets


Terminal
» git reset --hard upstream/userfuncs
HEAD is now at 048017a Add `change_pass()`.
> git checkout master
>

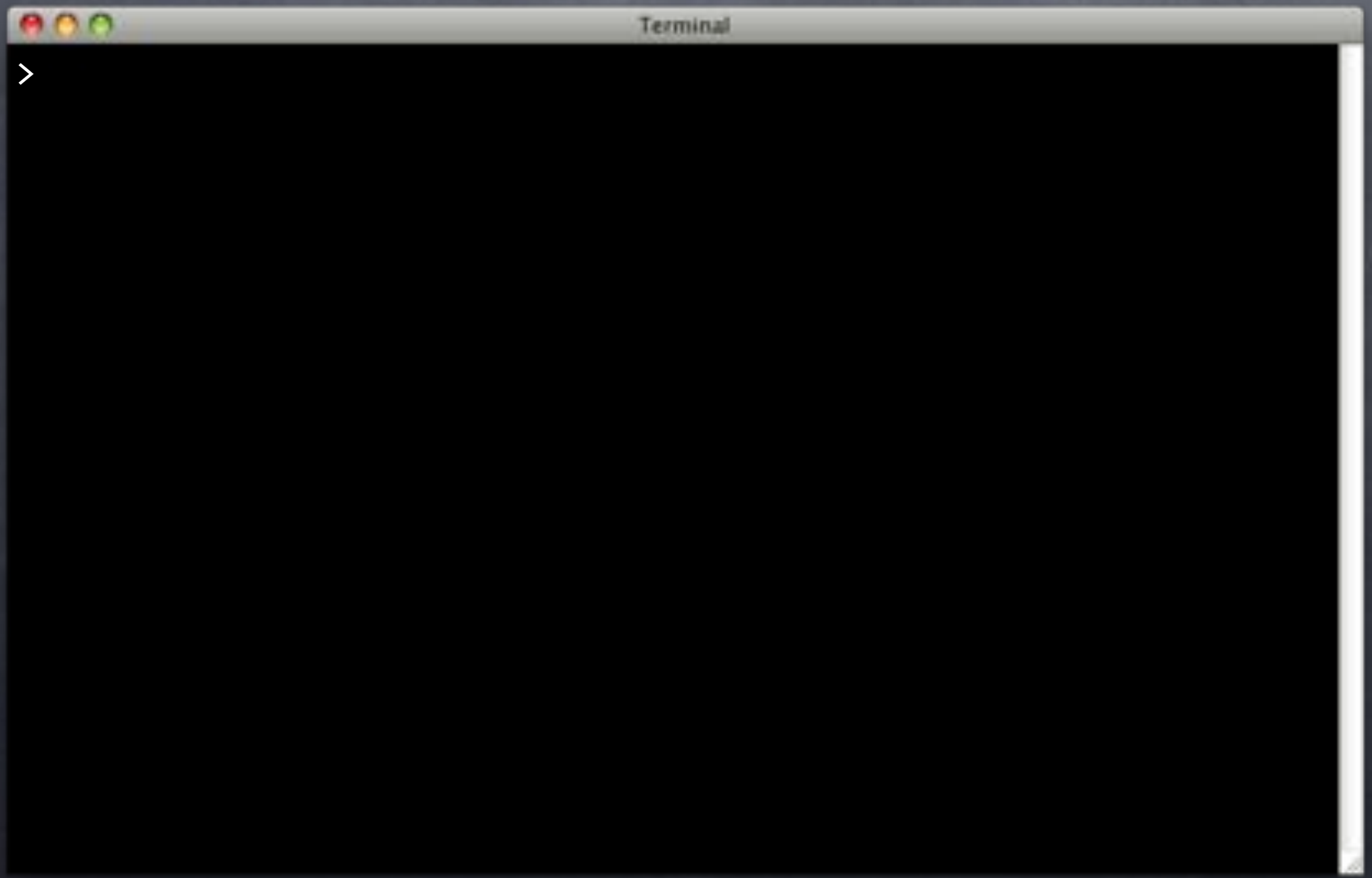
Resets


Terminal
> git reset --hard upstream/userfuncs
HEAD is now at 048017a Add `change_pass()`.
> git checkout master
> git reset --hard appschema
HEAD is now at e46bdf9 Add appschema test.
>

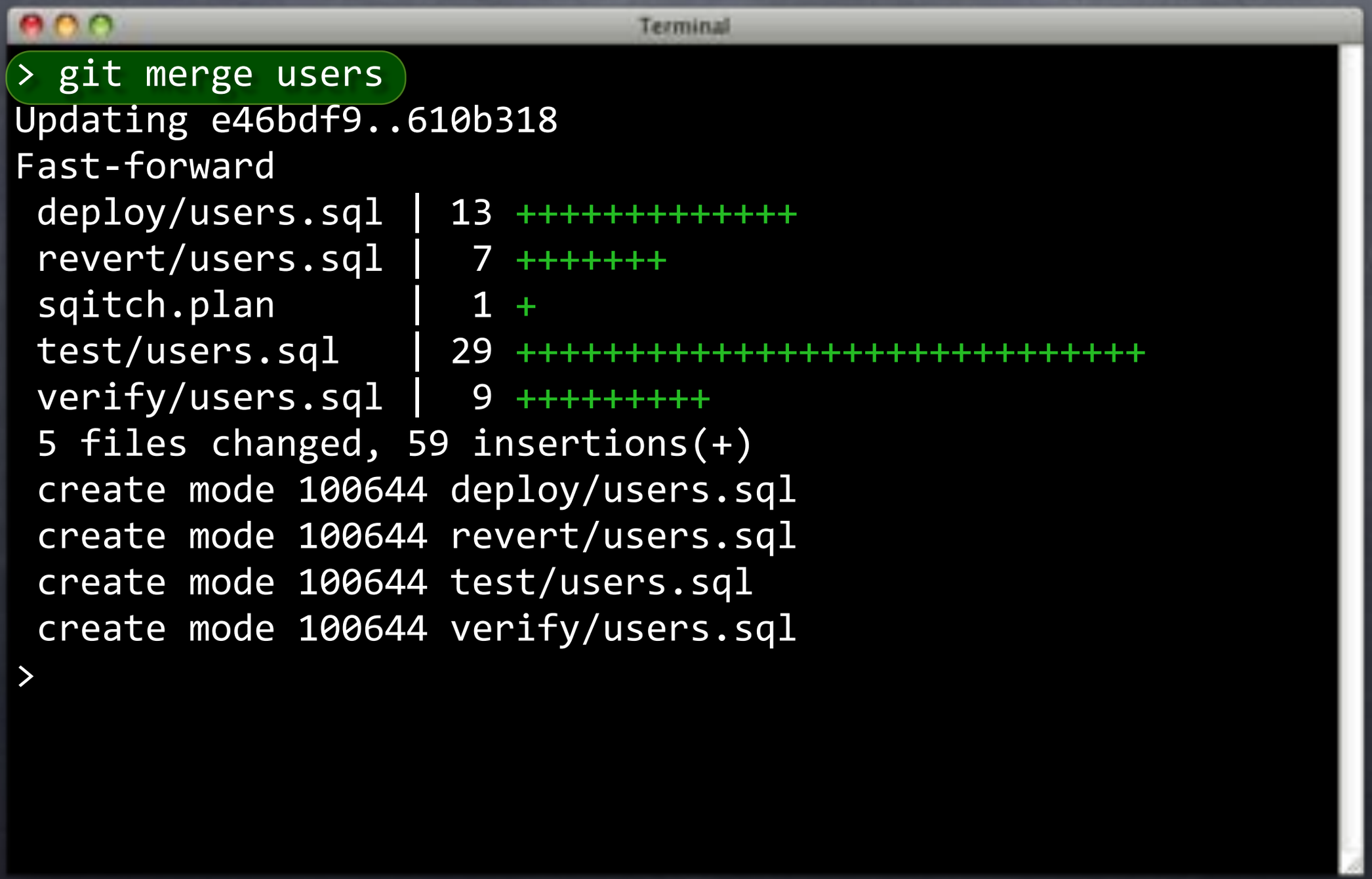
Resets


Terminal
> git reset --hard upstream/userfuncs
HEAD is now at 048017a Add `change_pass()`.
> git checkout master
> git reset --hard appschema
HEAD is now at e46bdf9 Add appschema test.
>
Known good.

Mergers and Acquisitions



Mergers and Acquisitions



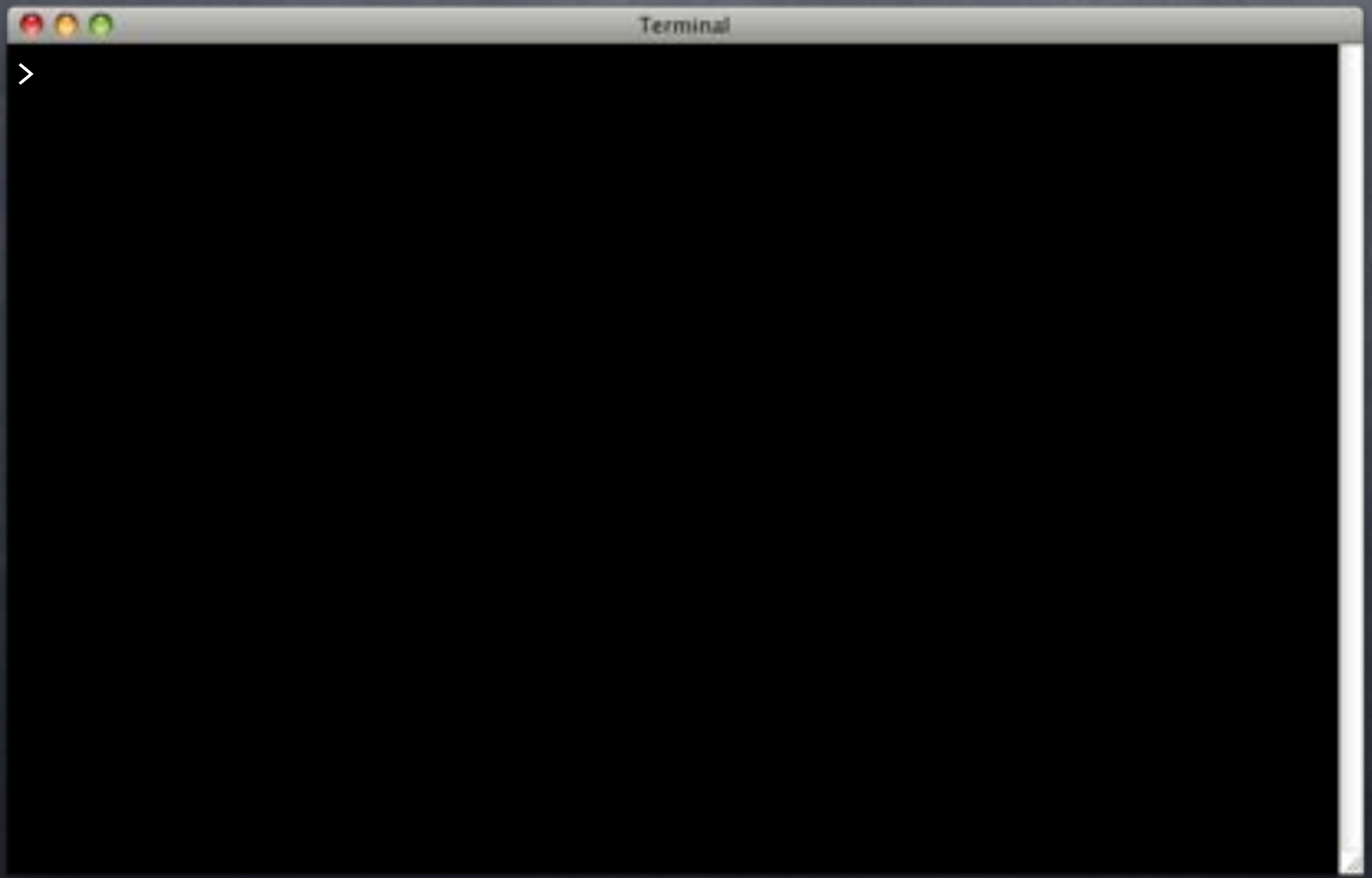
Mergers and Acquisitions

Terminal

```
> git merge users
Updating e46bdf9..610b318
Fast-forward
 deploy/users.sql | 13 +++++++++++++
 revert/users.sql |  7 +++++++
 sqitch.plan      |  1 +
 test/users.sql   | 29 +++++++++++++++++++++++++++++
 verify/users.sql |  9 +++++++++
 5 files changed, 59 insertions(+)
 create mode 100644 deploy/users.sql
 create mode 100644 revert/users.sql
 create mode 100644 test/users.sql
 create mode 100644 verify/users.sql
>
```

So far...

Mergers and Acquisitions



Mergers and Acquisitions

Terminal

```
> git merge flips
Updating 610b318..fbf8469
Fast-forward
 deploy/flips.sql | 15 +++++++++++++++
 revert/flips.sql |  7 +++++++
 sqitch.plan      |  1 +
 test/flips.sql   | 38 ++++++++++++++++++++++++++++++++++++++
 verify/flips.sql | 12 ++++++++++++
 5 files changed, 73 insertions(+)
 create mode 100644 deploy/flips.sql
 create mode 100644 revert/flips.sql
 create mode 100644 test/flips.sql
 create mode 100644 verify/flips.sql
>
```

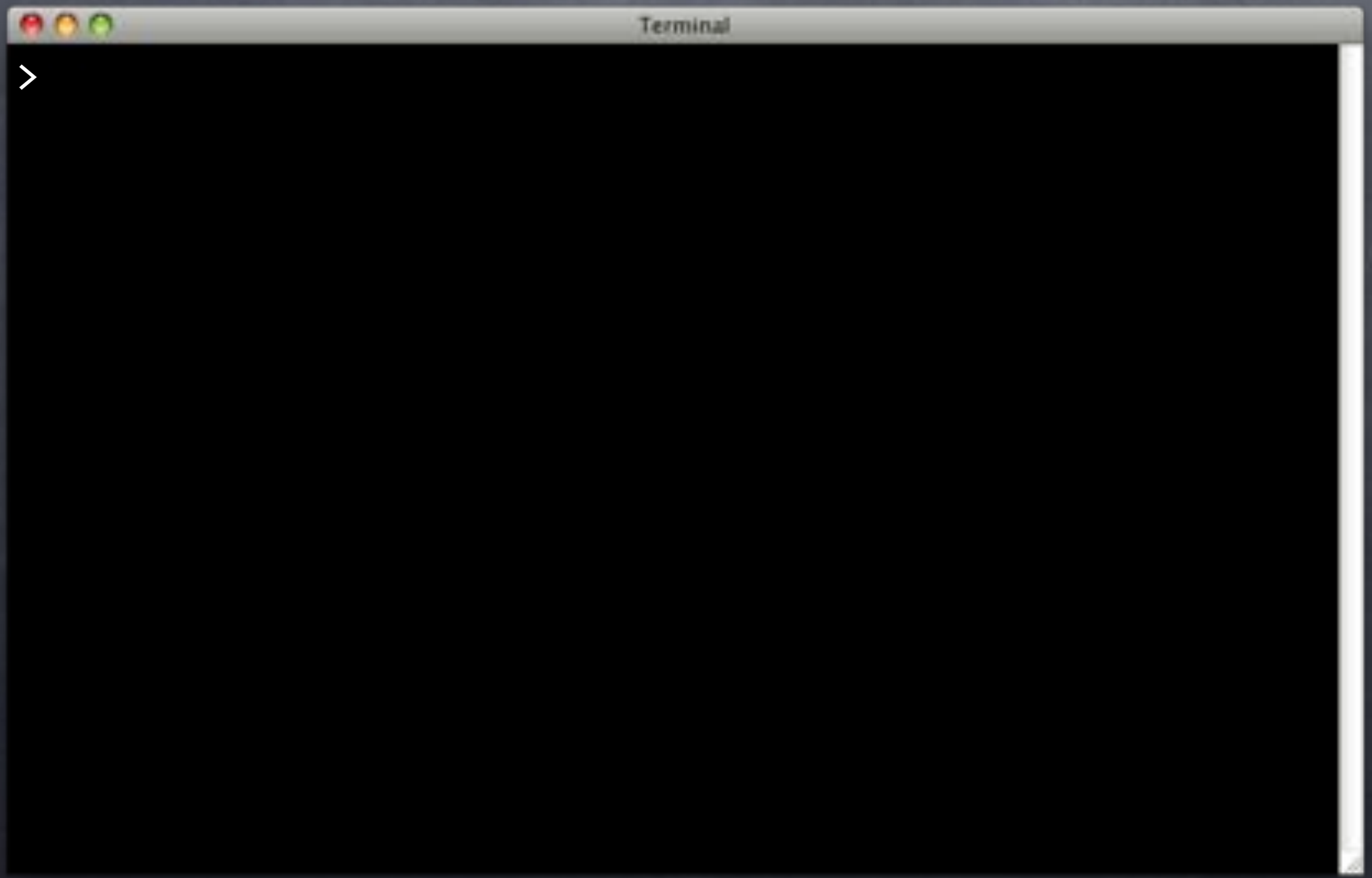
Mergers and Acquisitions

Terminal

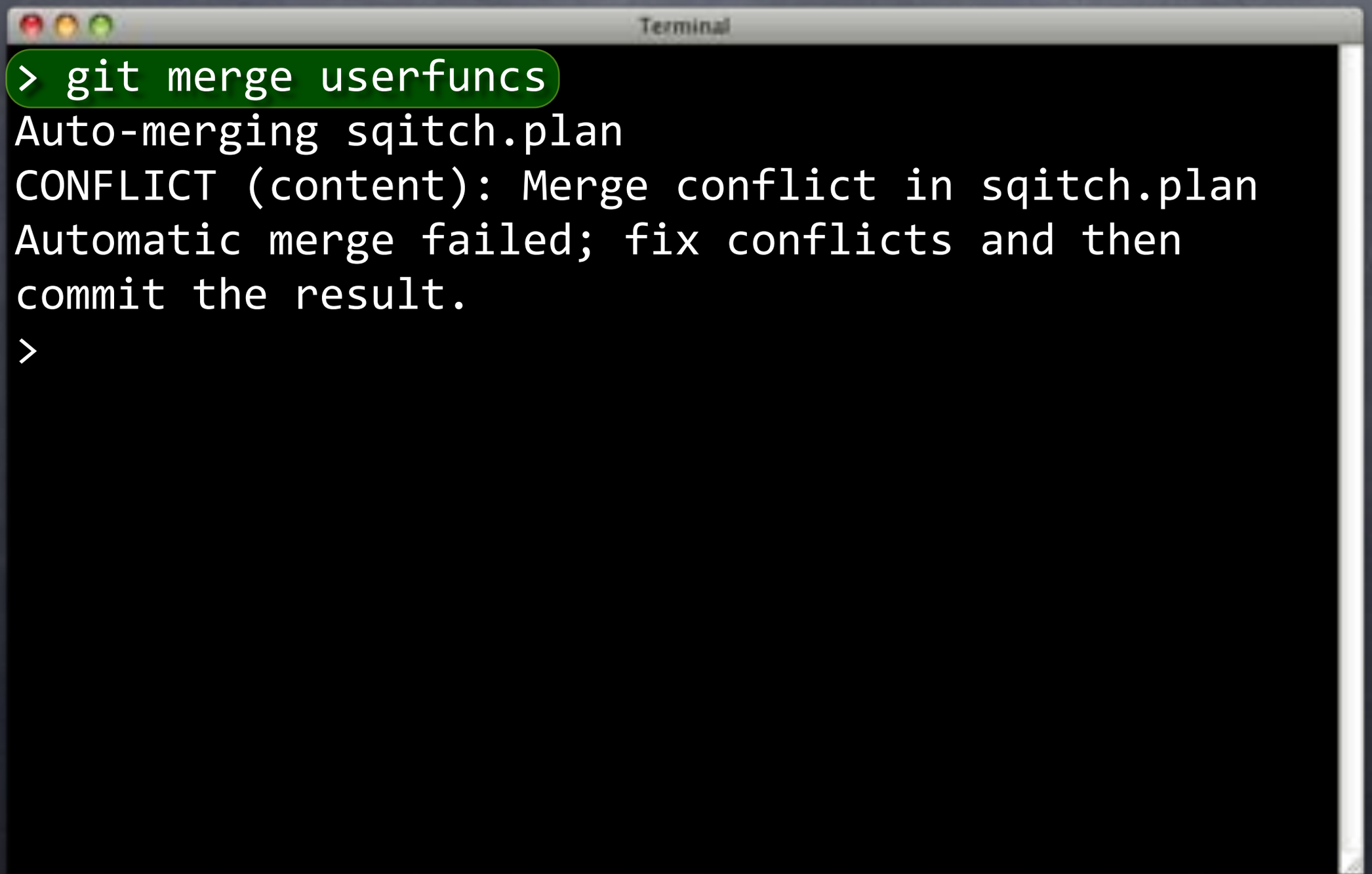
```
> git merge flips
Updating 610b318..fbf8469
Fast-forward
 deploy/flips.sql | 15 +++++++++++++++
 revert/flips.sql |  7 +++++++
 sqitch.plan      |  1 +
 test/flips.sql   | 38 ++++++++++++++++++++++++++++++++++++++
 verify/flips.sql | 12 ++++++++++++
 5 files changed, 73 insertions(+)
 create mode 100644 deploy/flips.sql
 create mode 100644 revert/flips.sql
 create mode 100644 test/flips.sql
 create mode 100644 verify/flips.sql
>
```

So good...

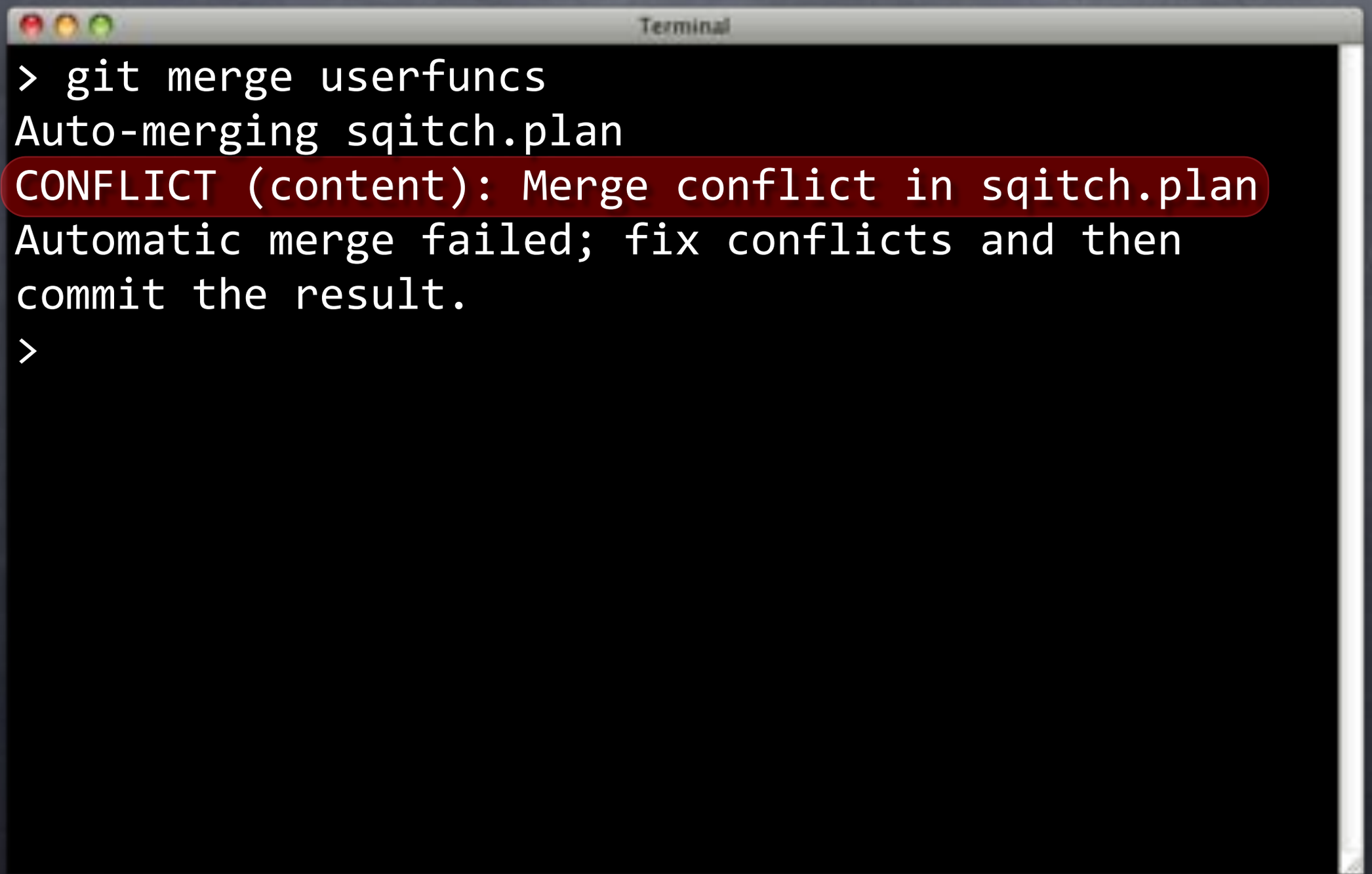
Mergers and Acquisitions



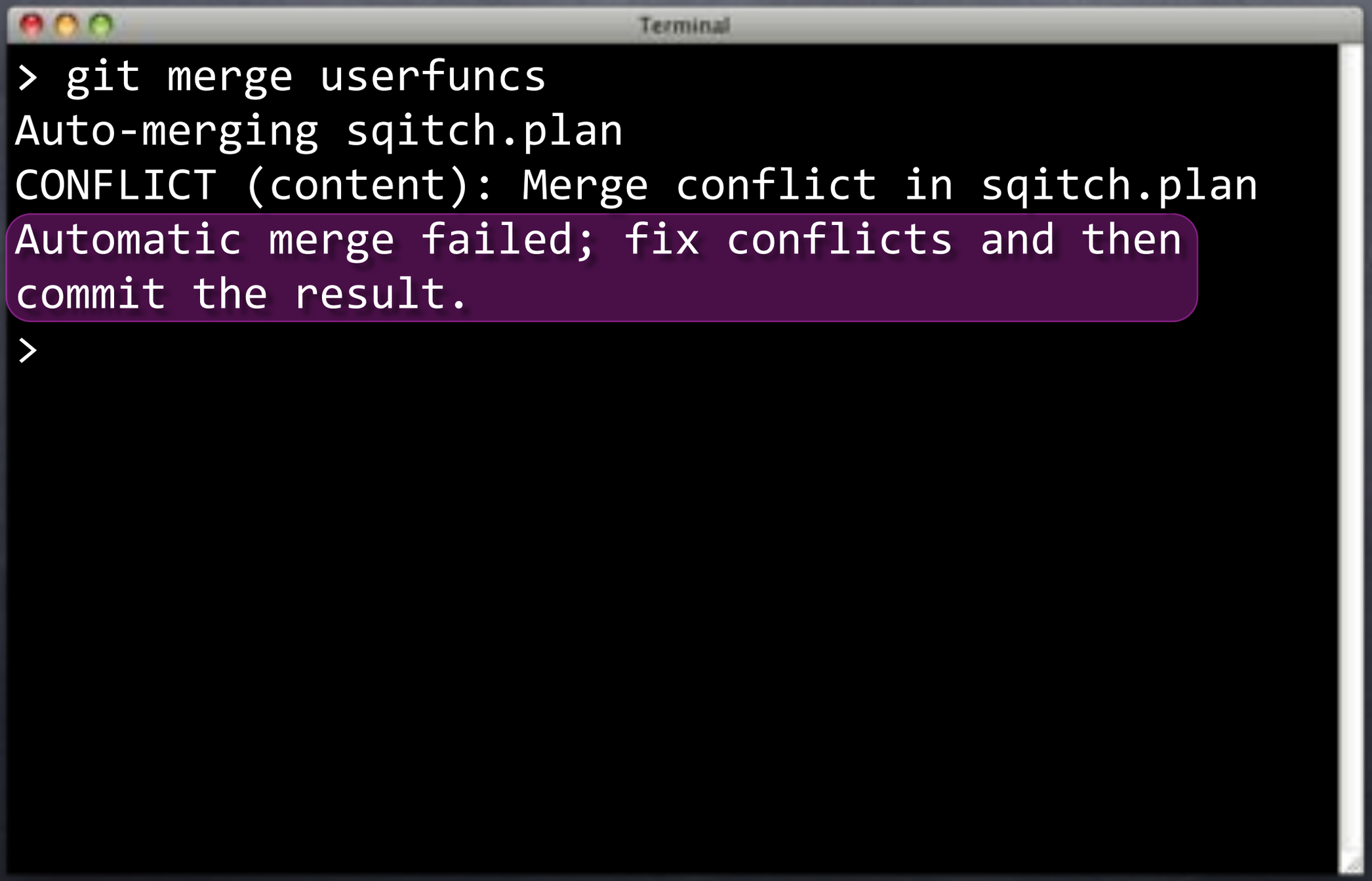
Mergers and Acquisitions



Mergers and Acquisitions



Mergers and Acquisitions



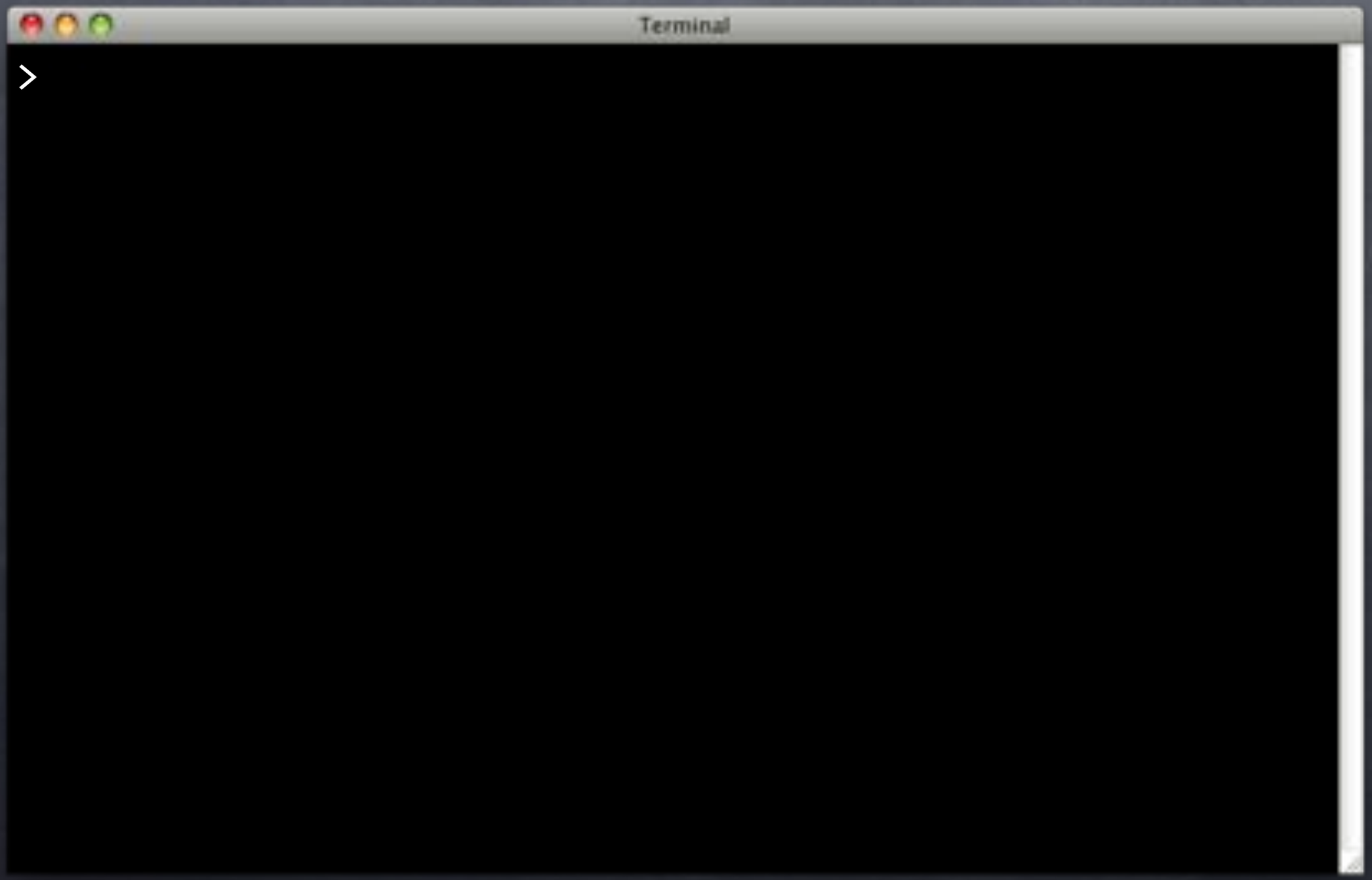
Mergers and Acquisitions

Terminal

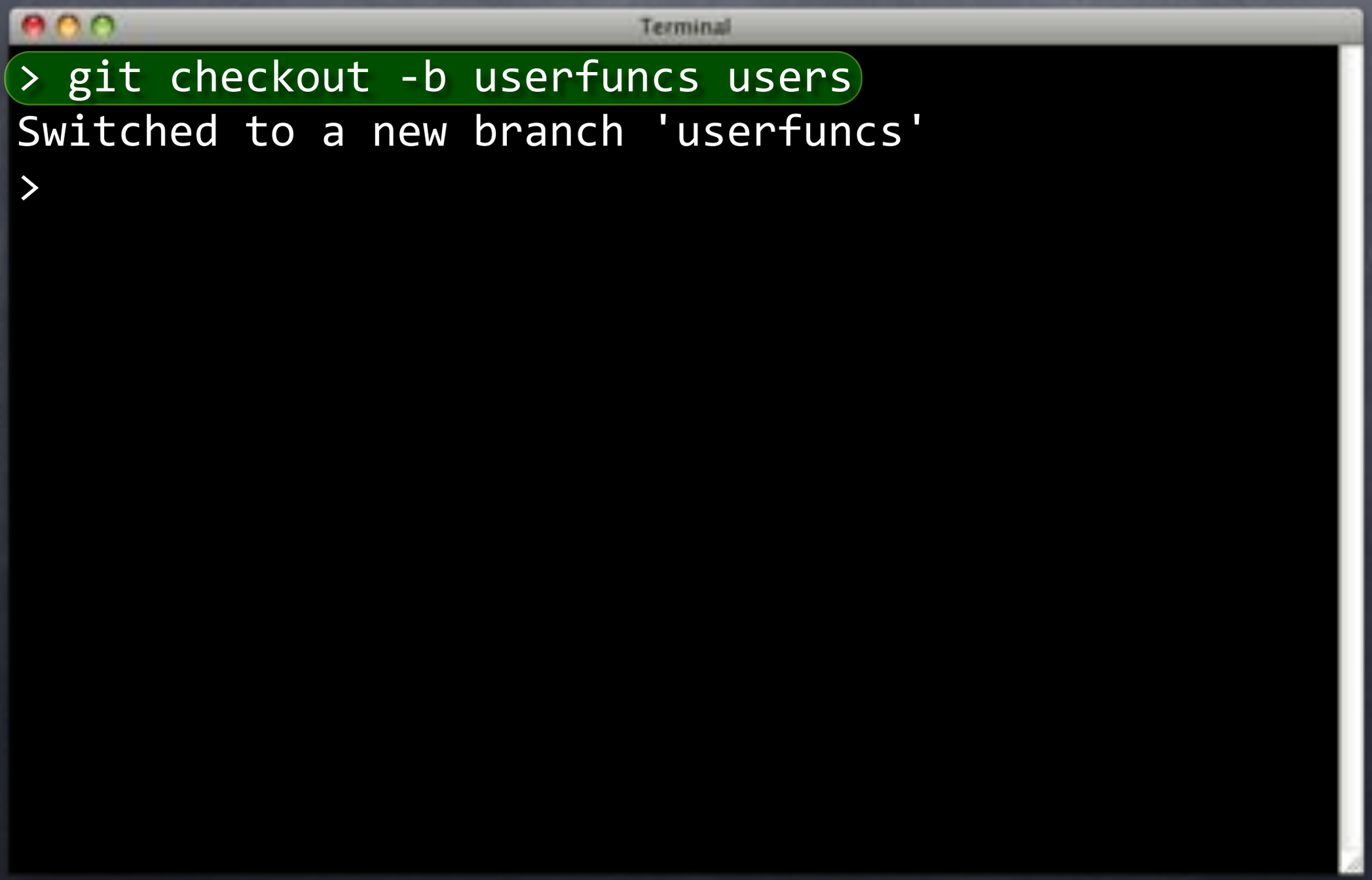
```
> git merge userfuncs
Auto-merging sqitch.plan
CONFLICT (content): Merge conflict in sqitch.plan
Automatic merge failed; fix conflicts and then
commit the result.
>
```

Wha???

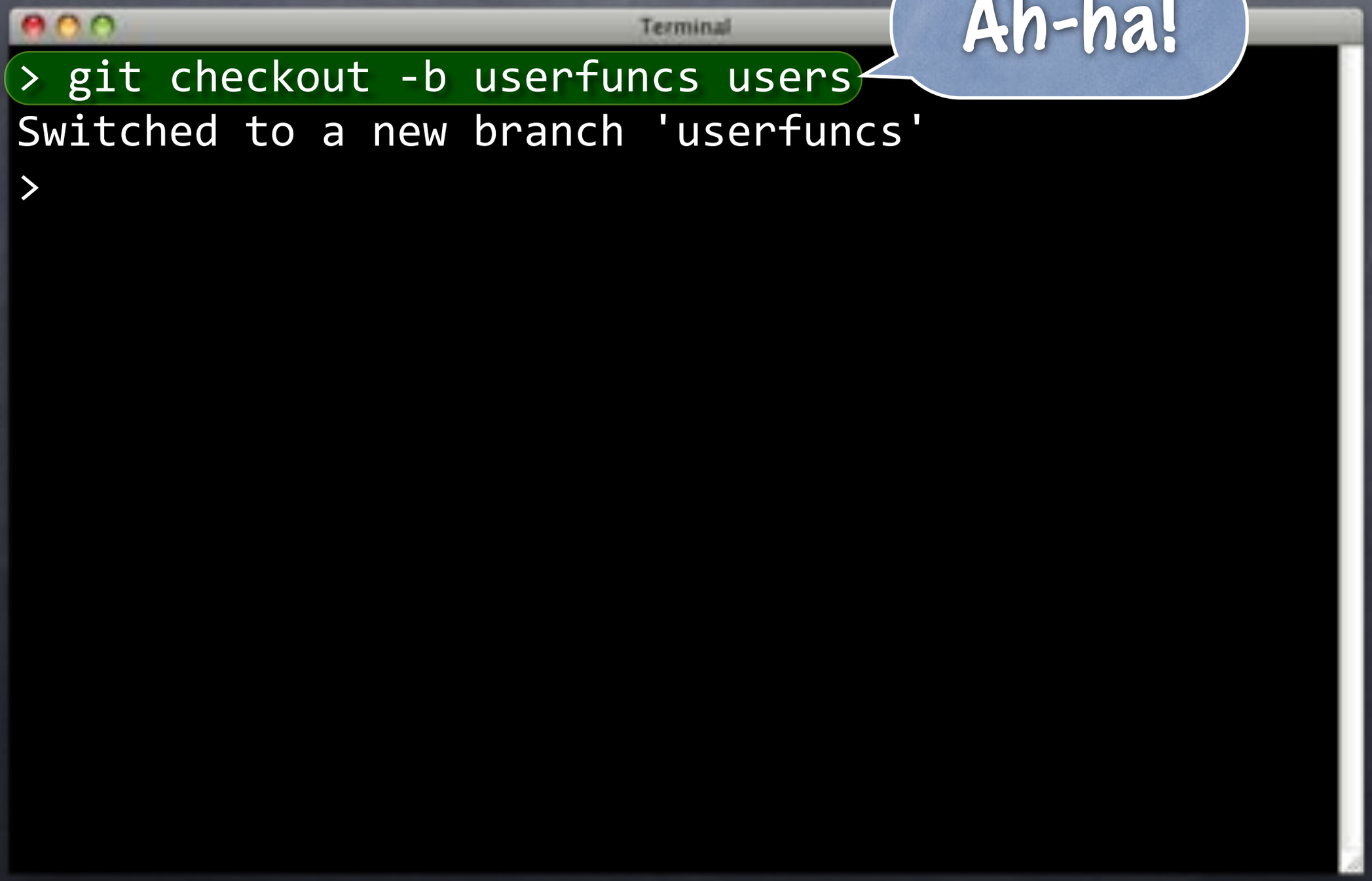
Back in Time...



Back in Time...


Terminal
> git checkout -b userfuncs users
Switched to a new branch 'userfuncs'
>

Back in Time...


Terminal
> git checkout -b userfuncs users
Switched to a new branch 'userfuncs'
>
Ah-ha!

Branching Out

Branching Out

- users branched from master

Branching Out

- users branched from master
- flips branched from users

Branching Out

- users branched from master
- flips branched from users
- userfuncs branched from users

Branching Out

- users branched from master
- flips branched from users
- userfuncs branched from users
- users and flips merged to master

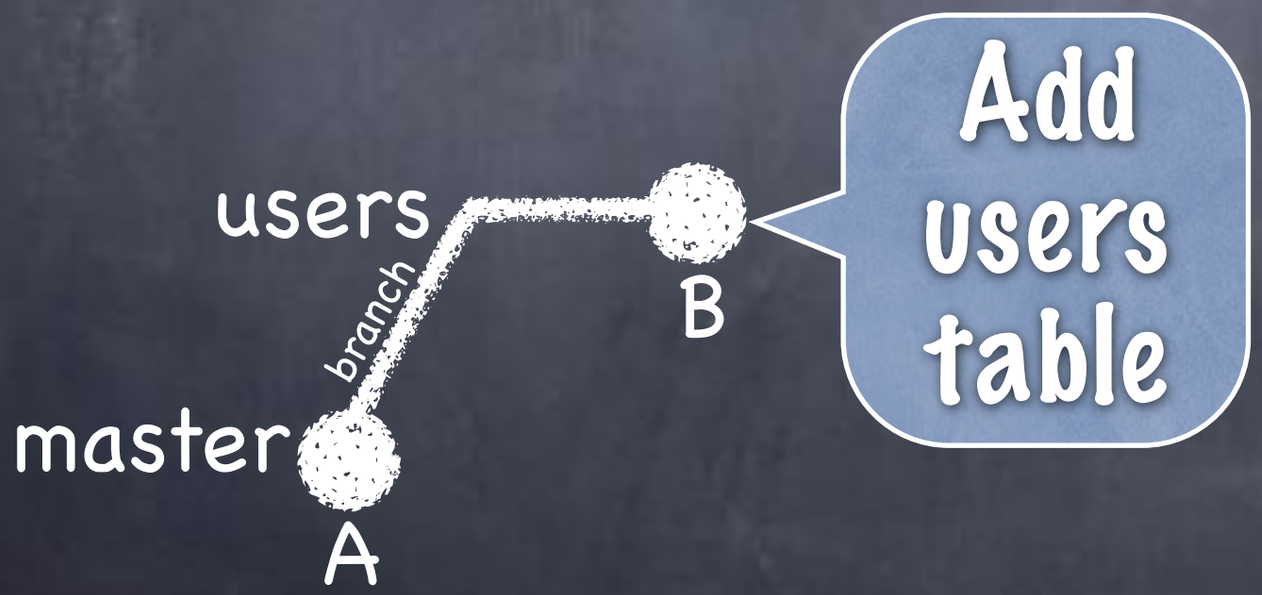
Branching Out

- users branched from master
- flips branched from users
- userfuncs branched from users
- users and flips merged to master
- userfuncs unaware of flips

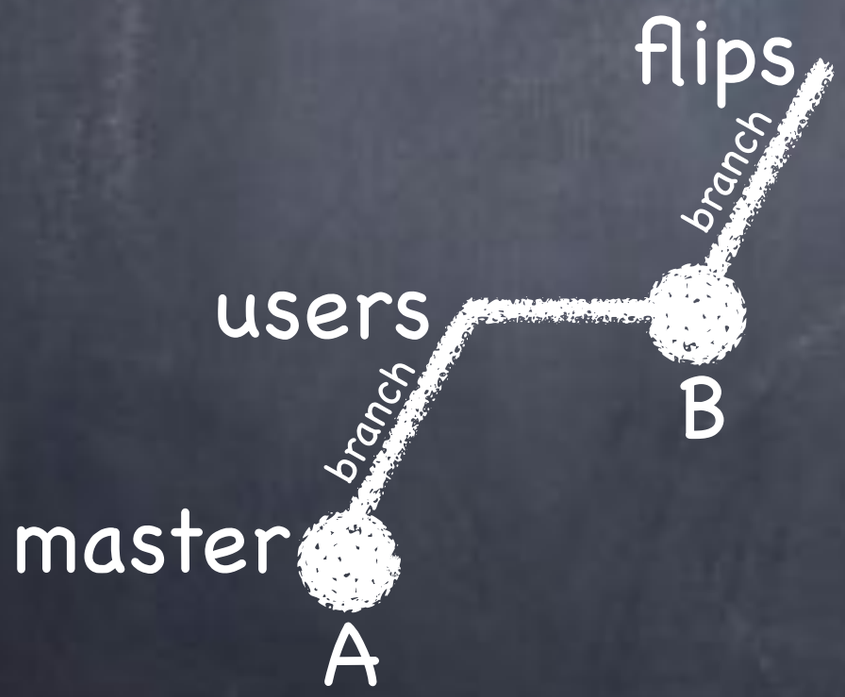
Backbrancher

Backbrancher

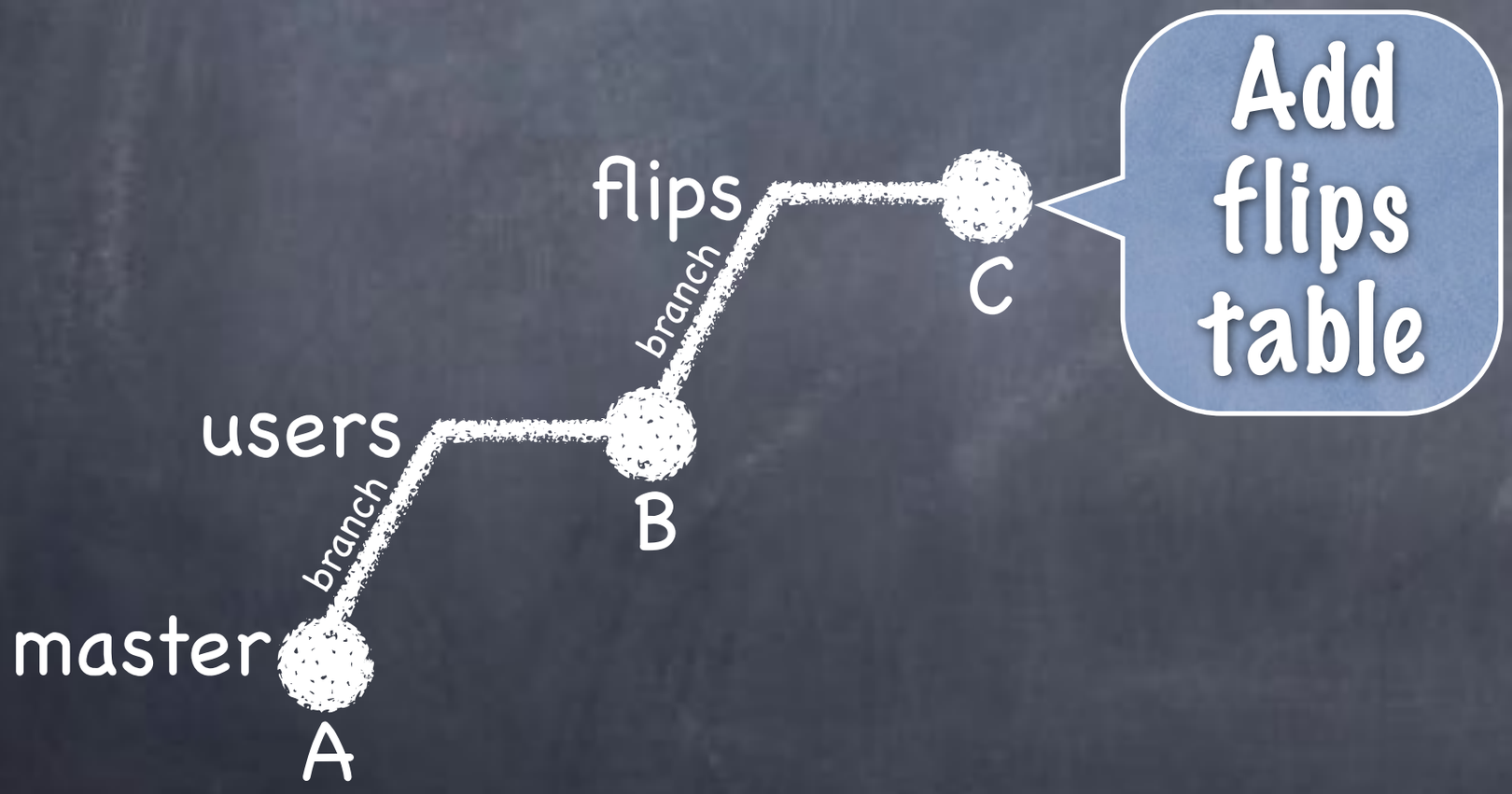
Backbrancher



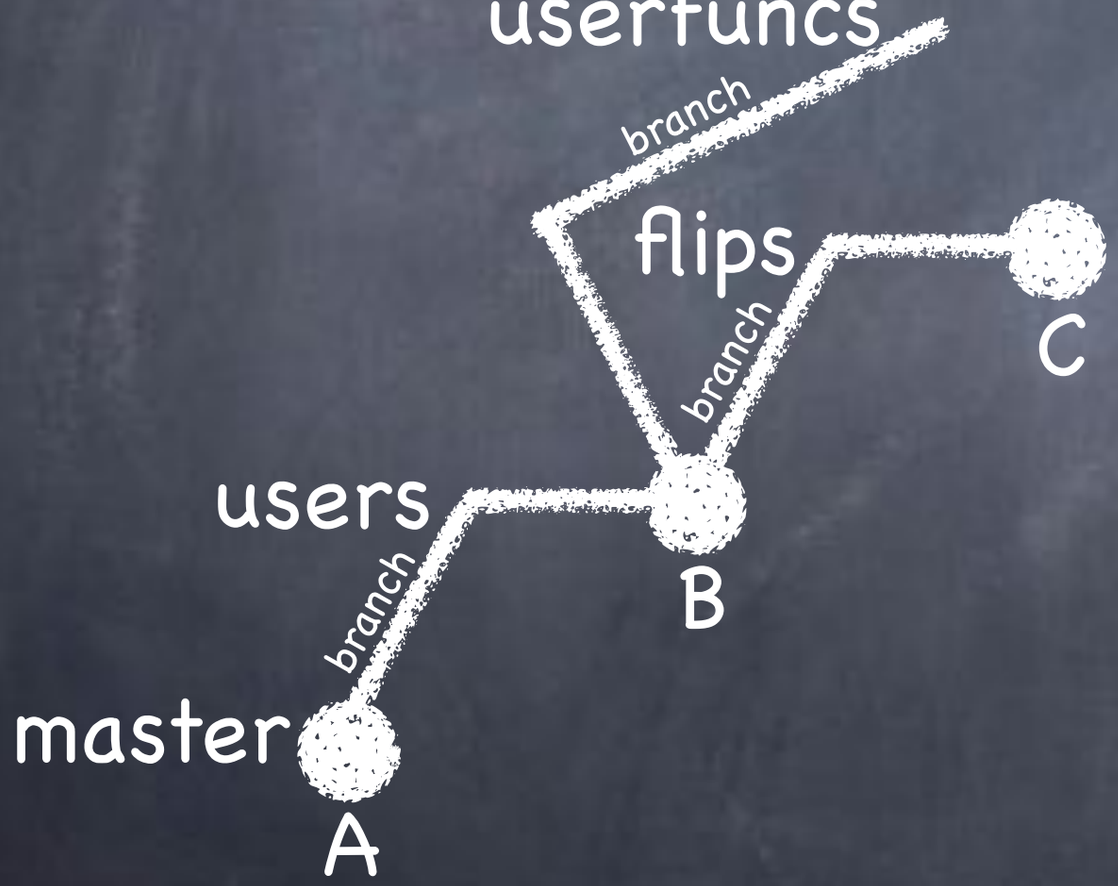
Backbrancher



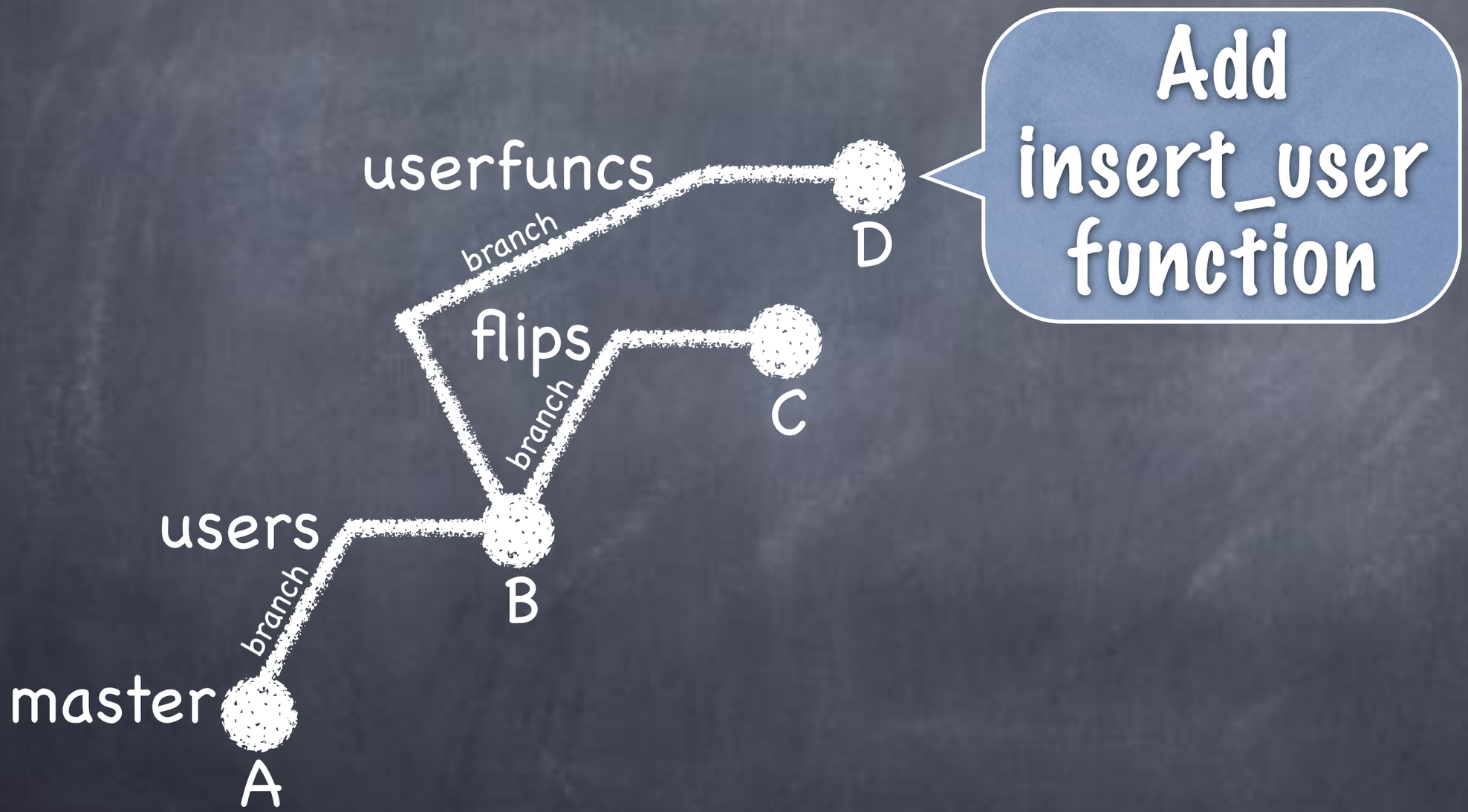
Backbrancher



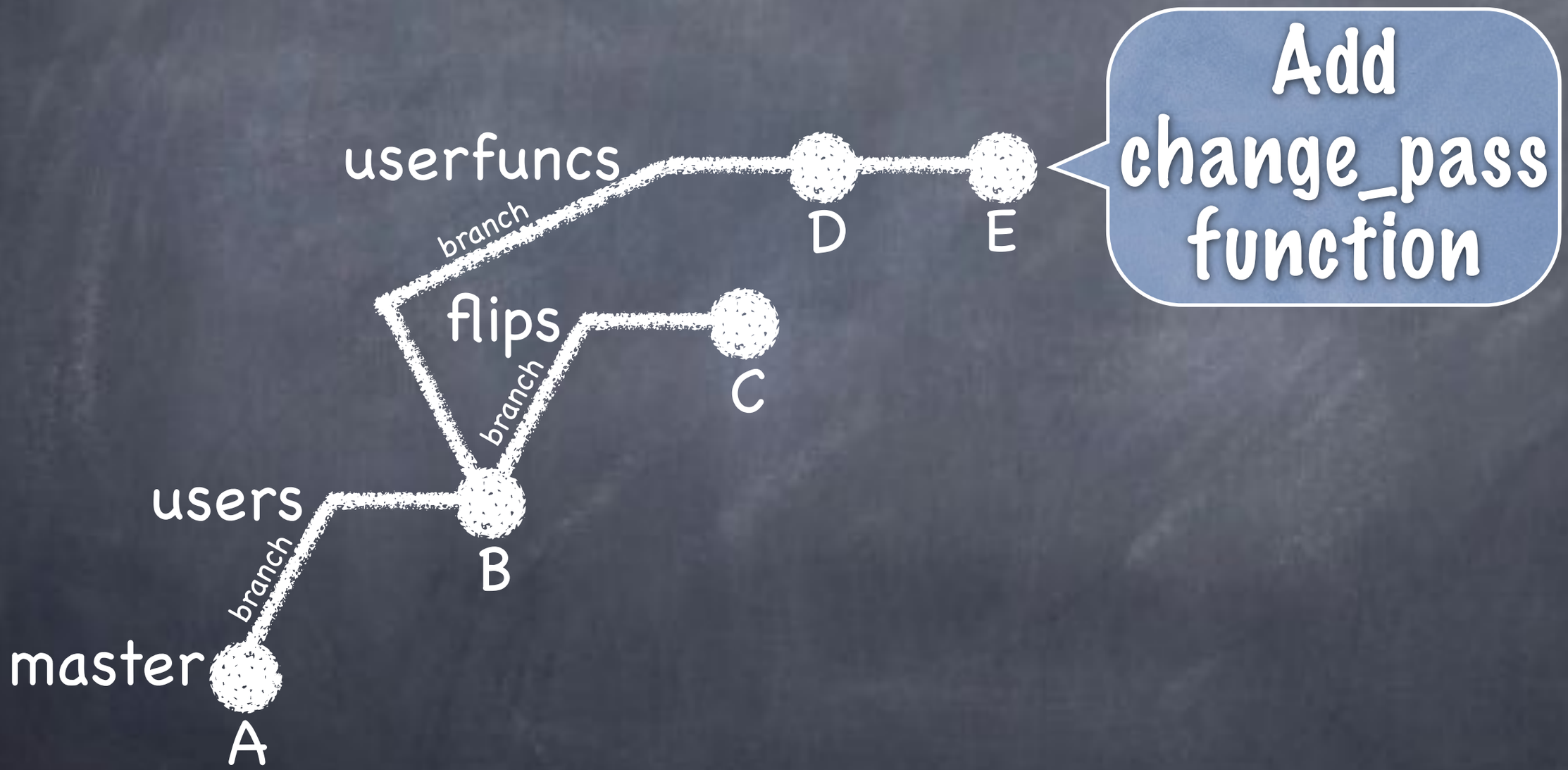
Backbrancher



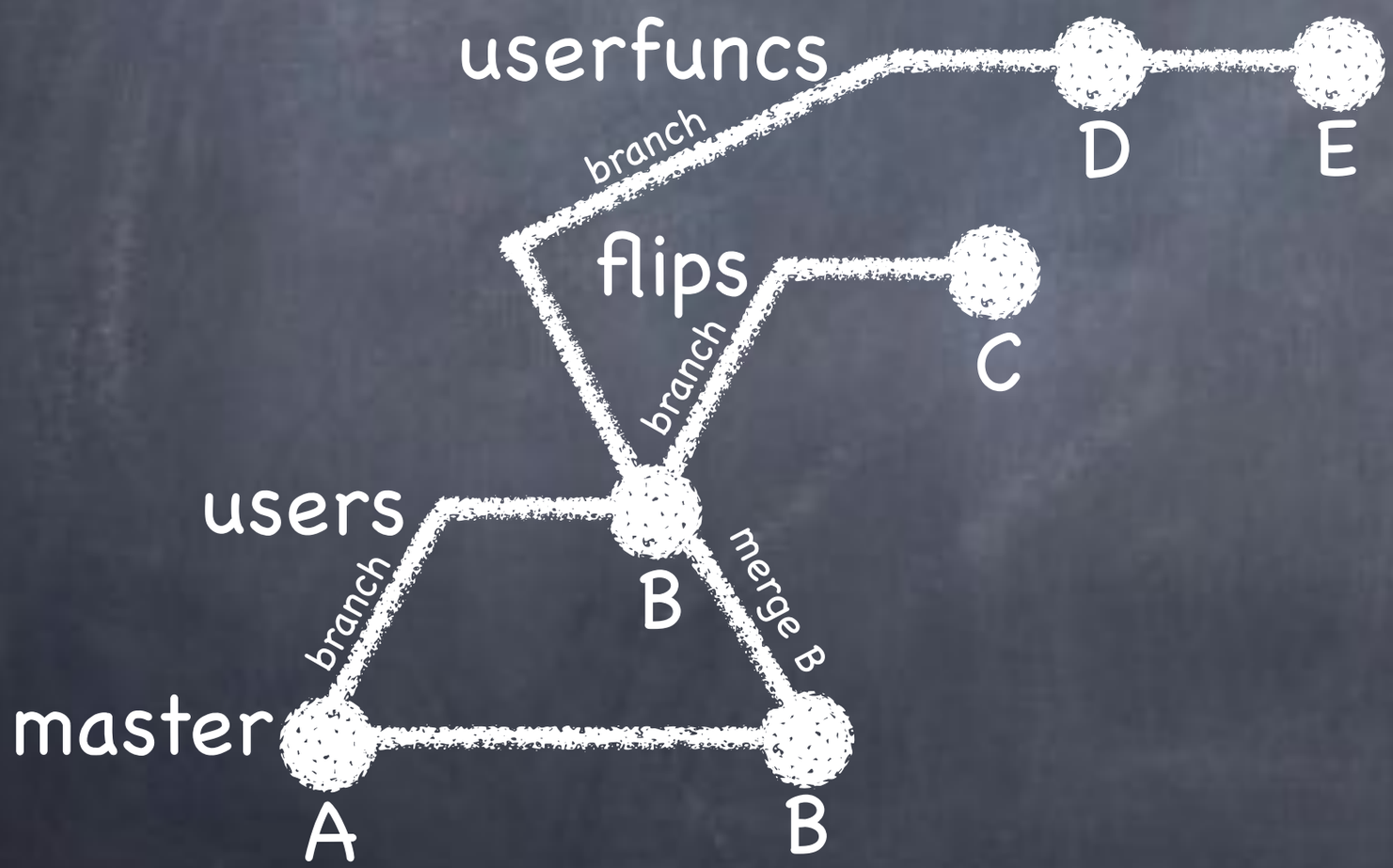
Backbrancher



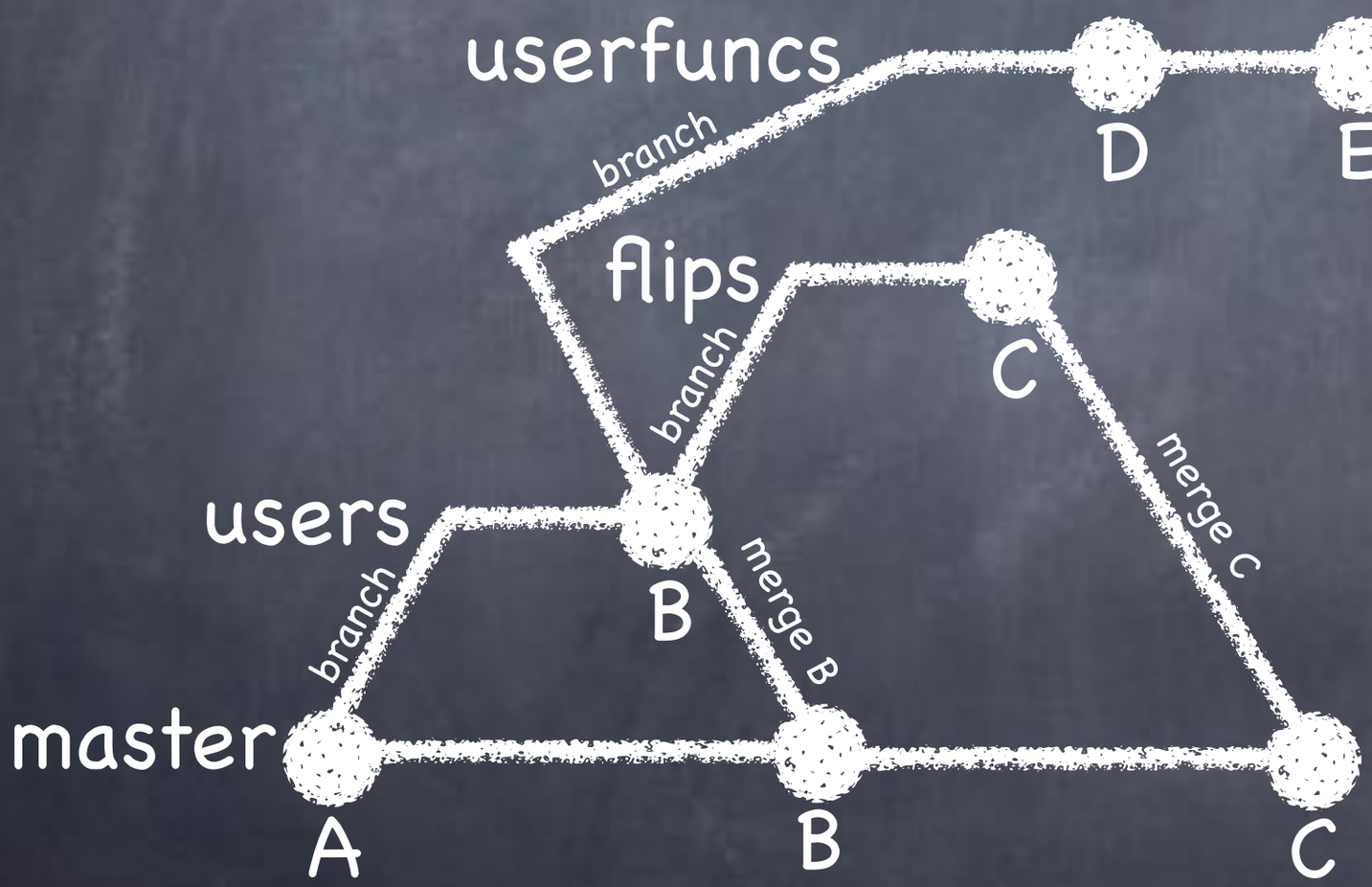
Backbrancher



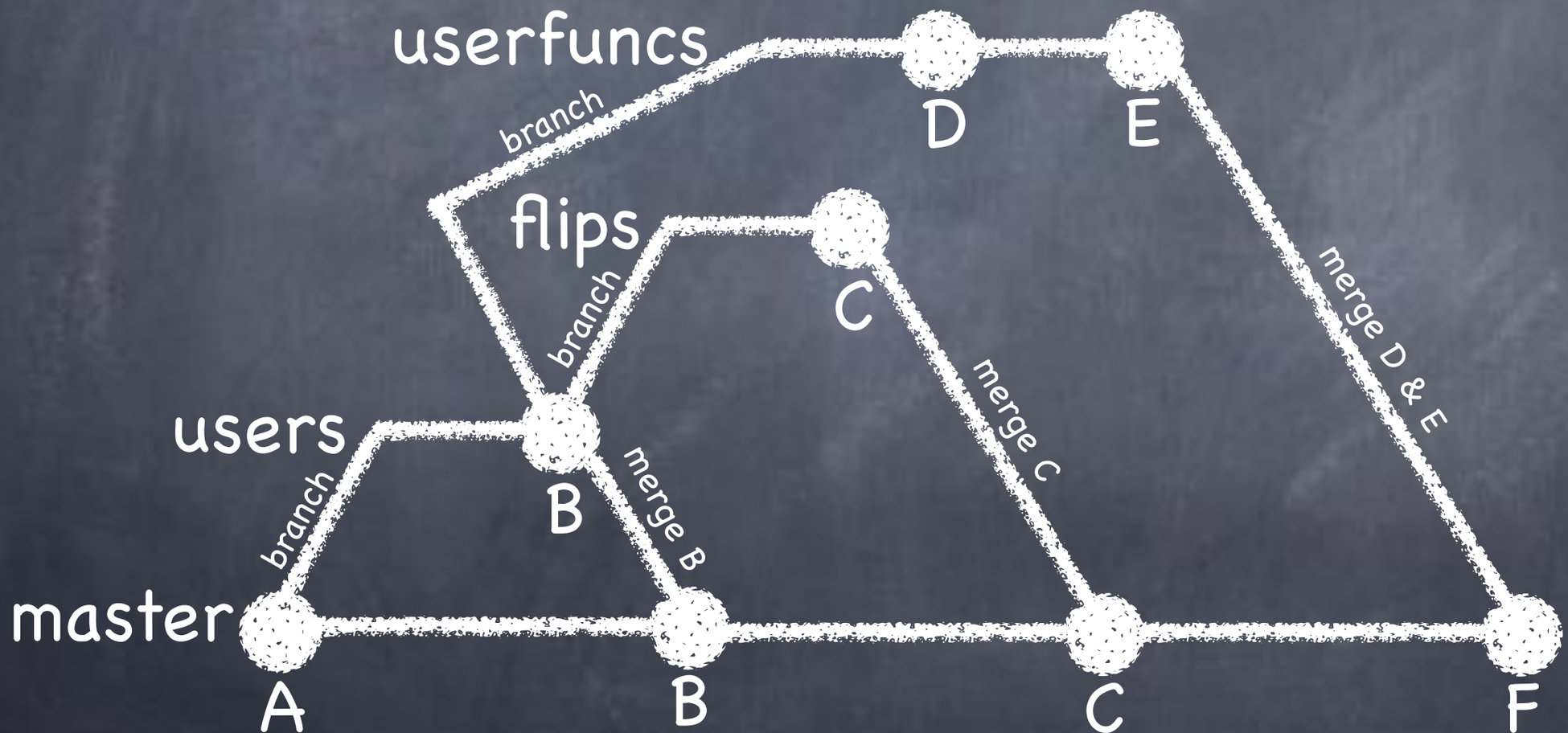
Backbrancher



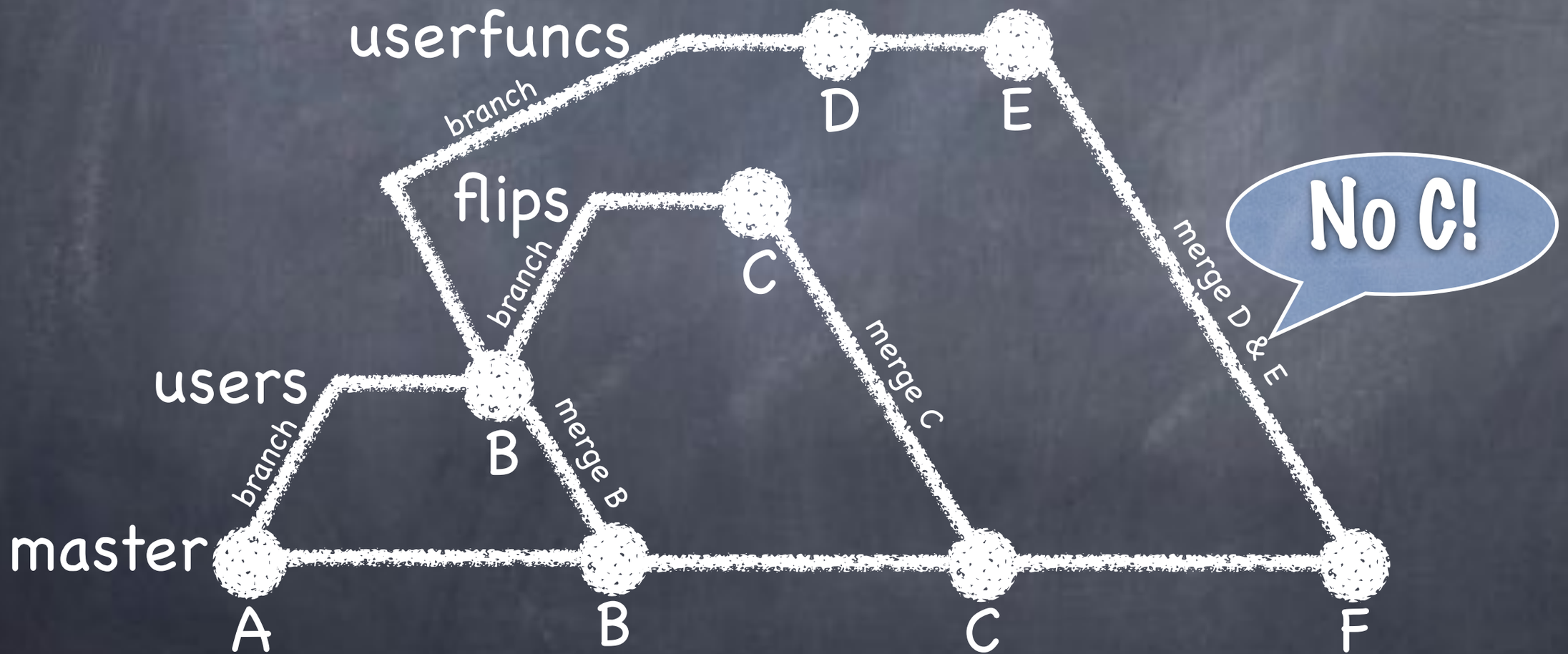
Backbrancher



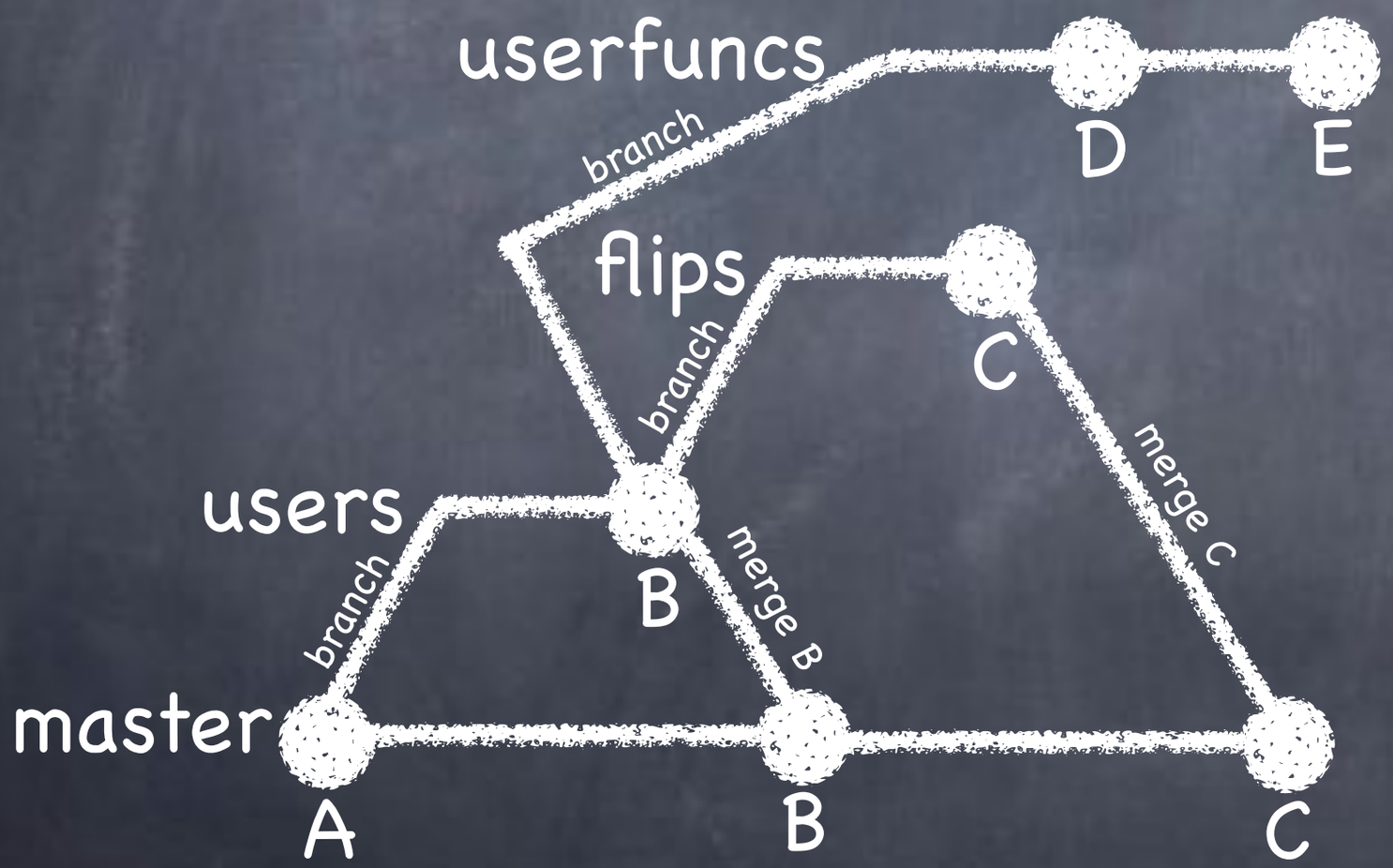
Backbrancher



Backbrancher

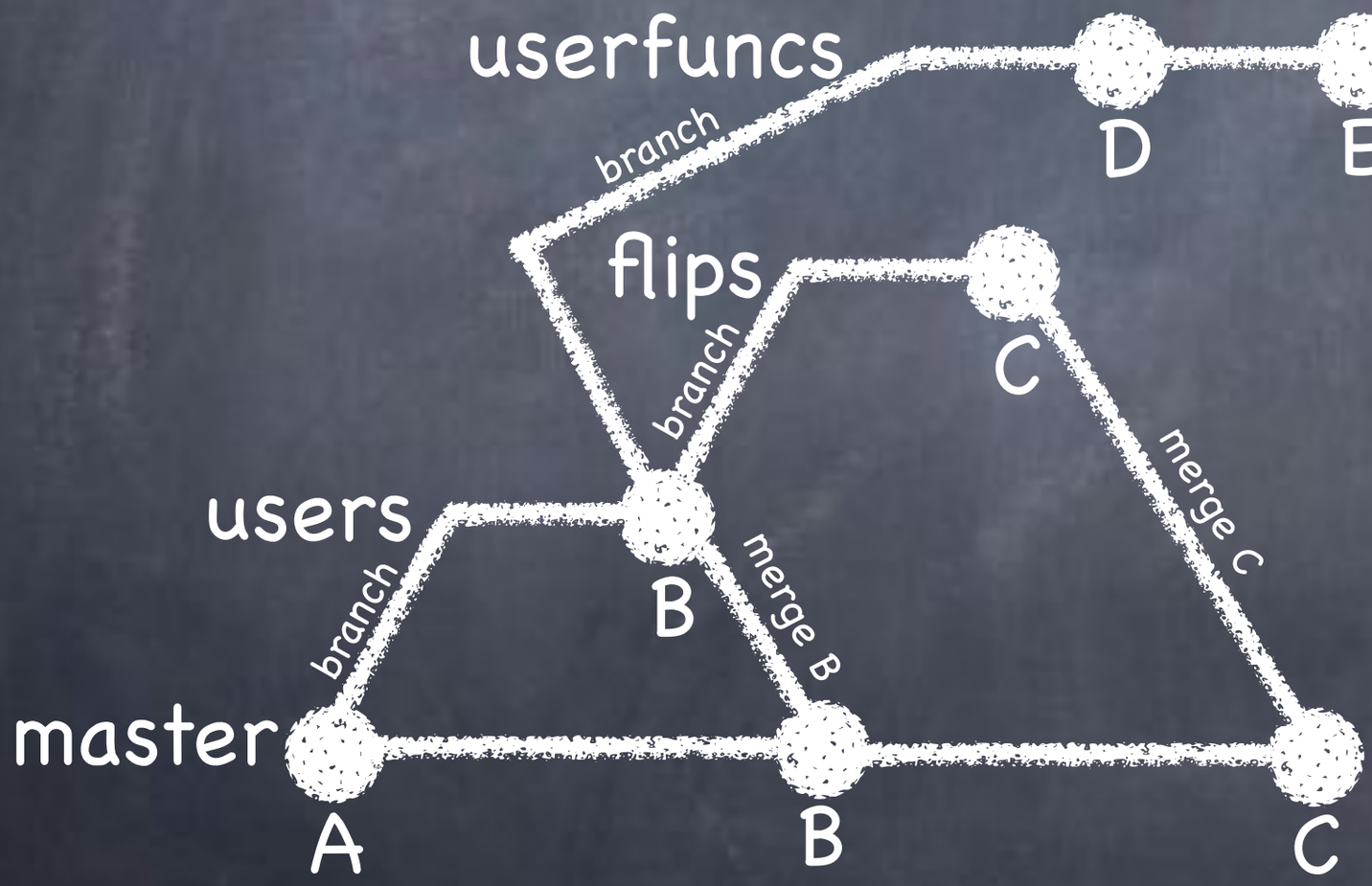


Backbrancher



conflict!

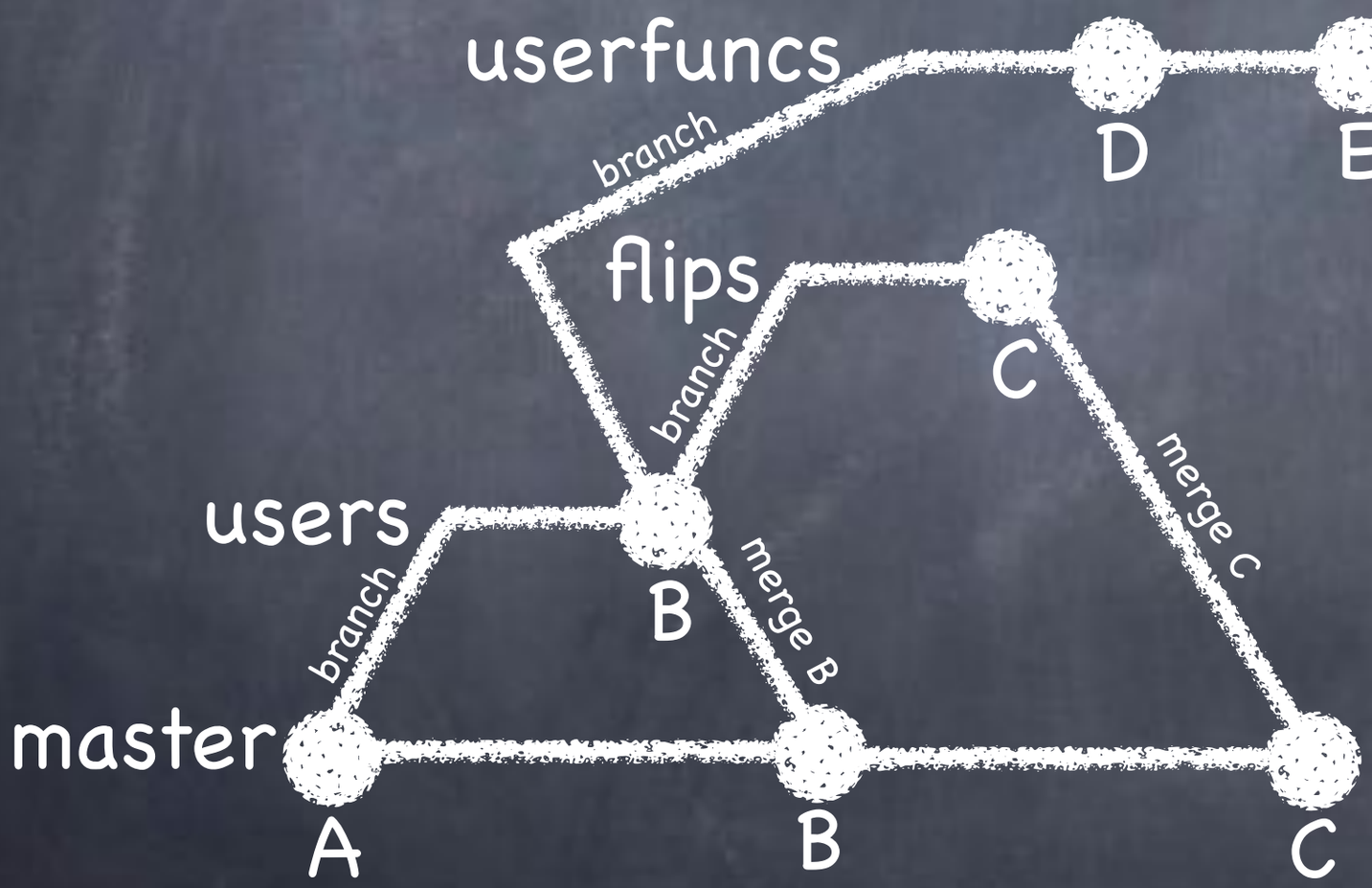
Backbrancher



Now what?

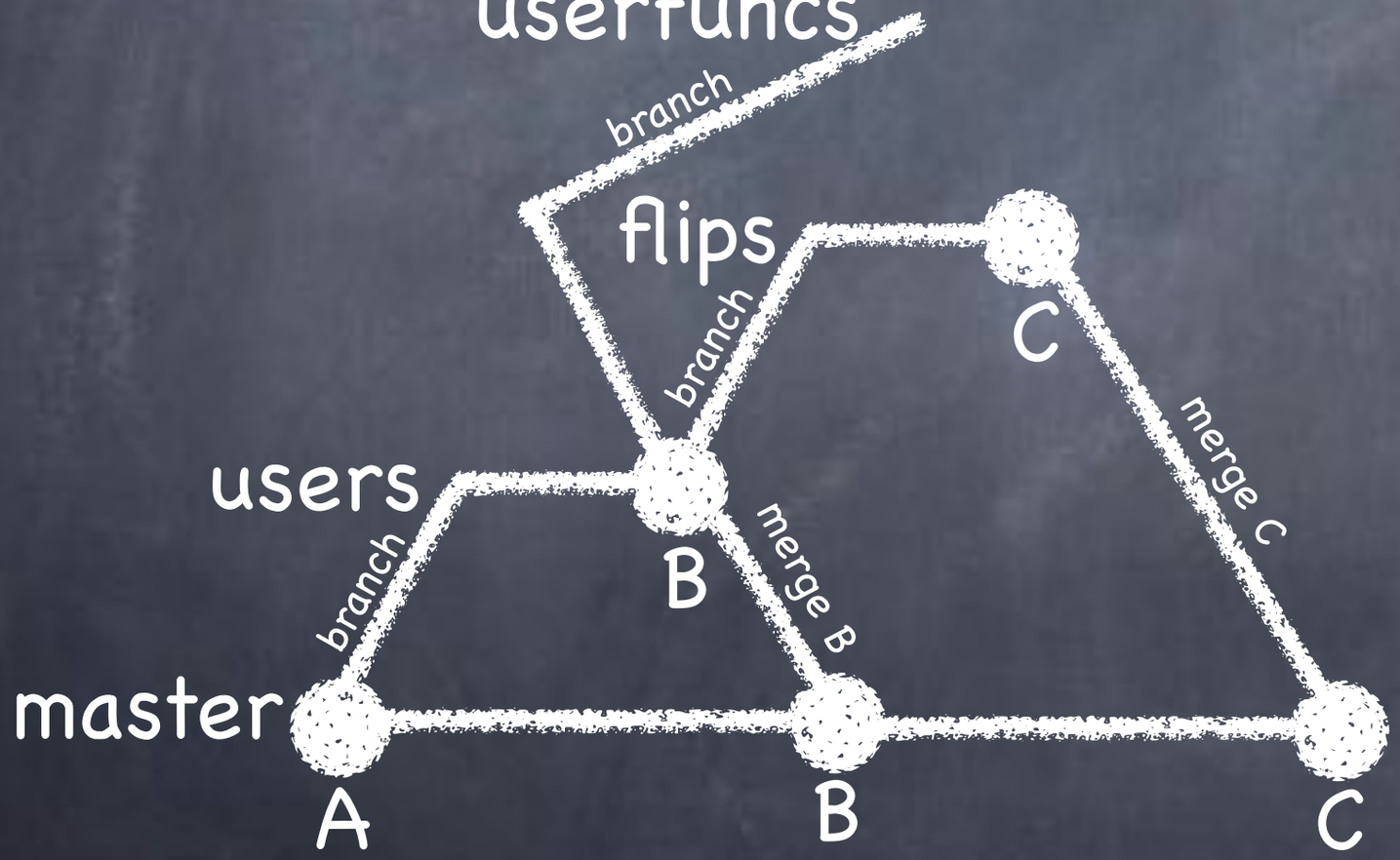
conflict!

Rebase master

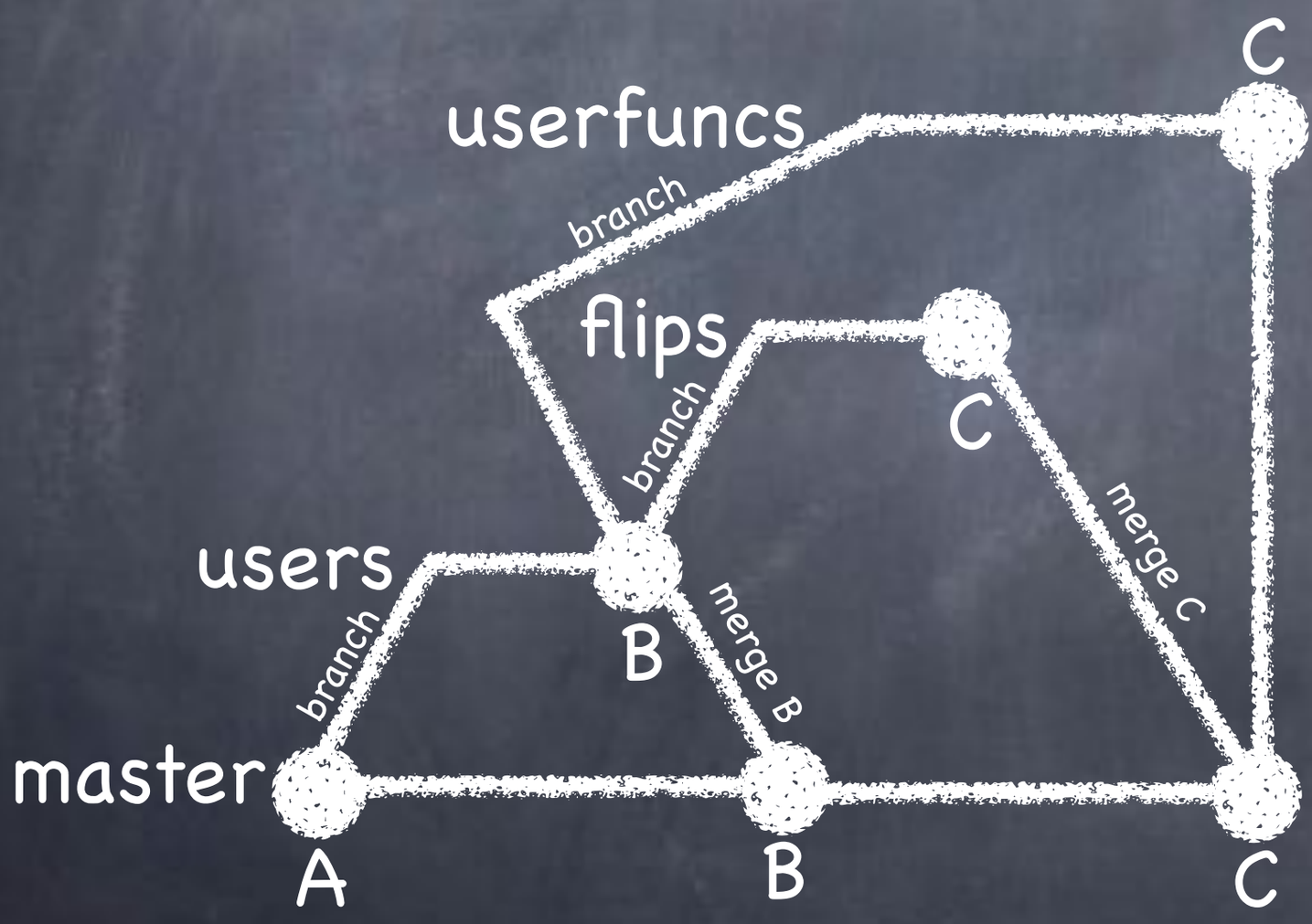


Rebase master

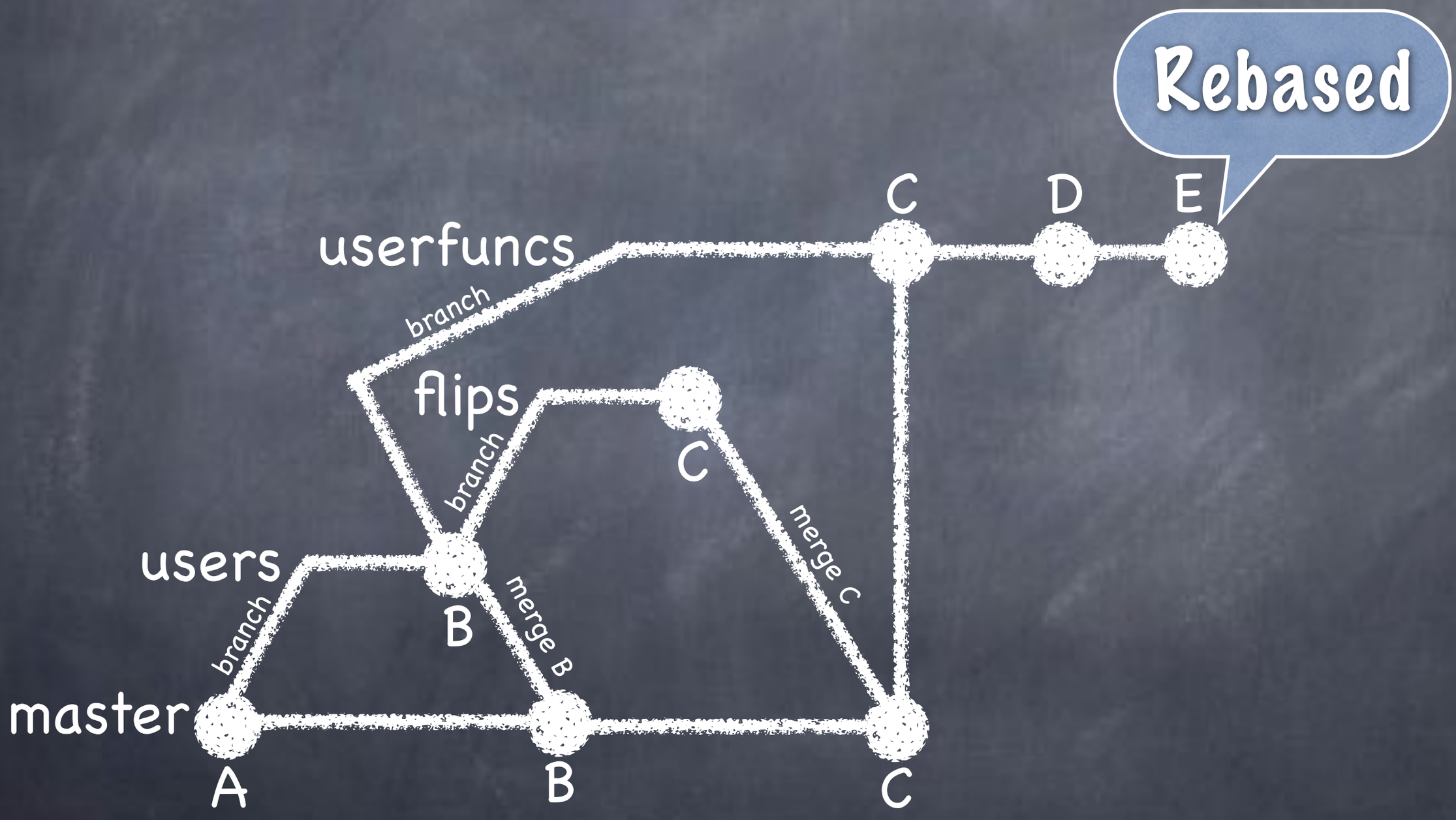
Last common with master: B



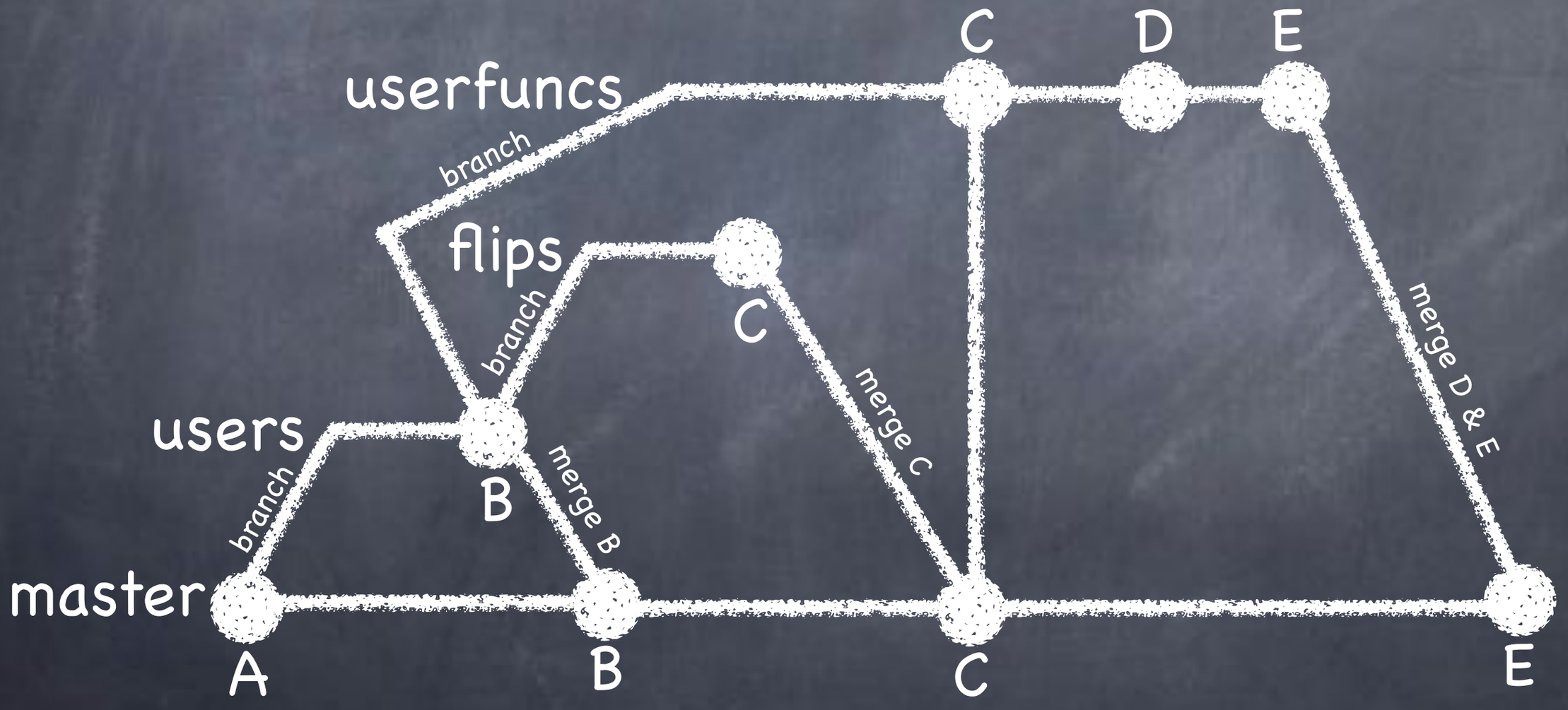
Rebase master



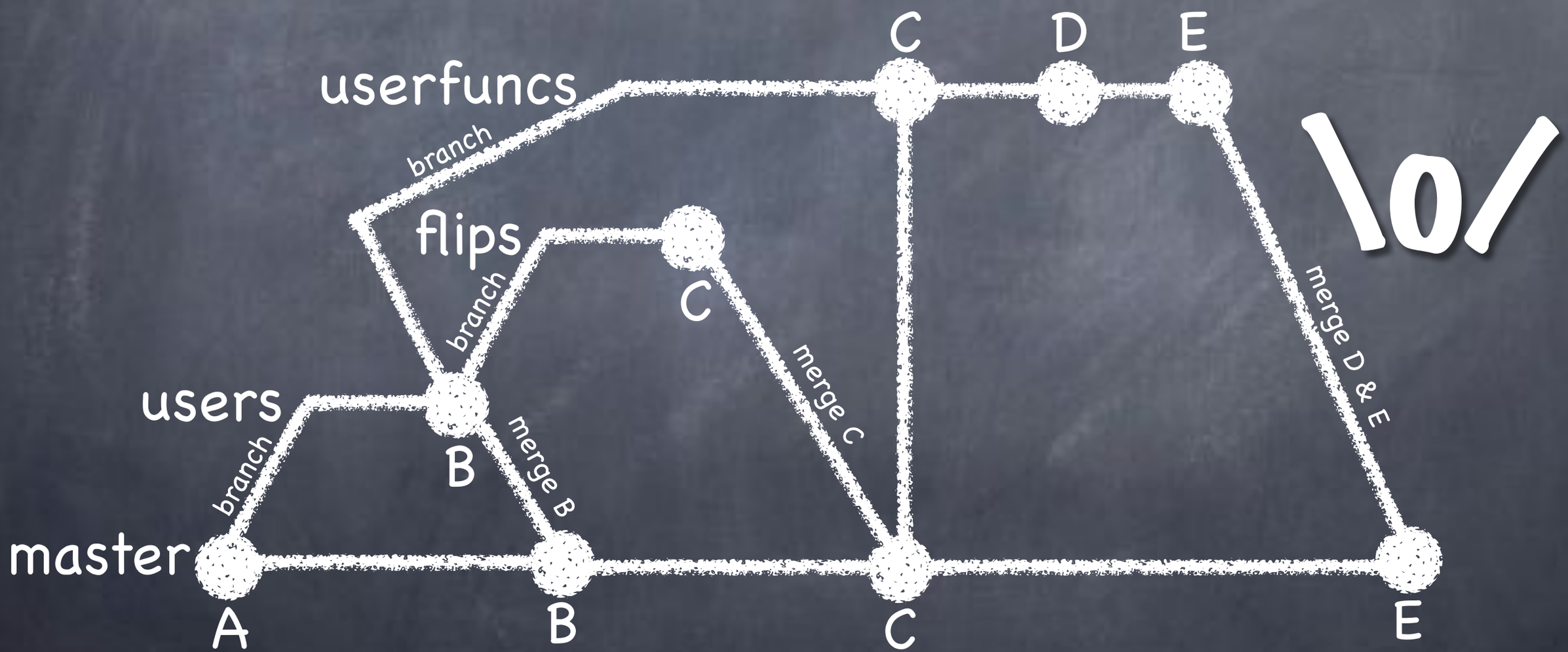
Rebase master



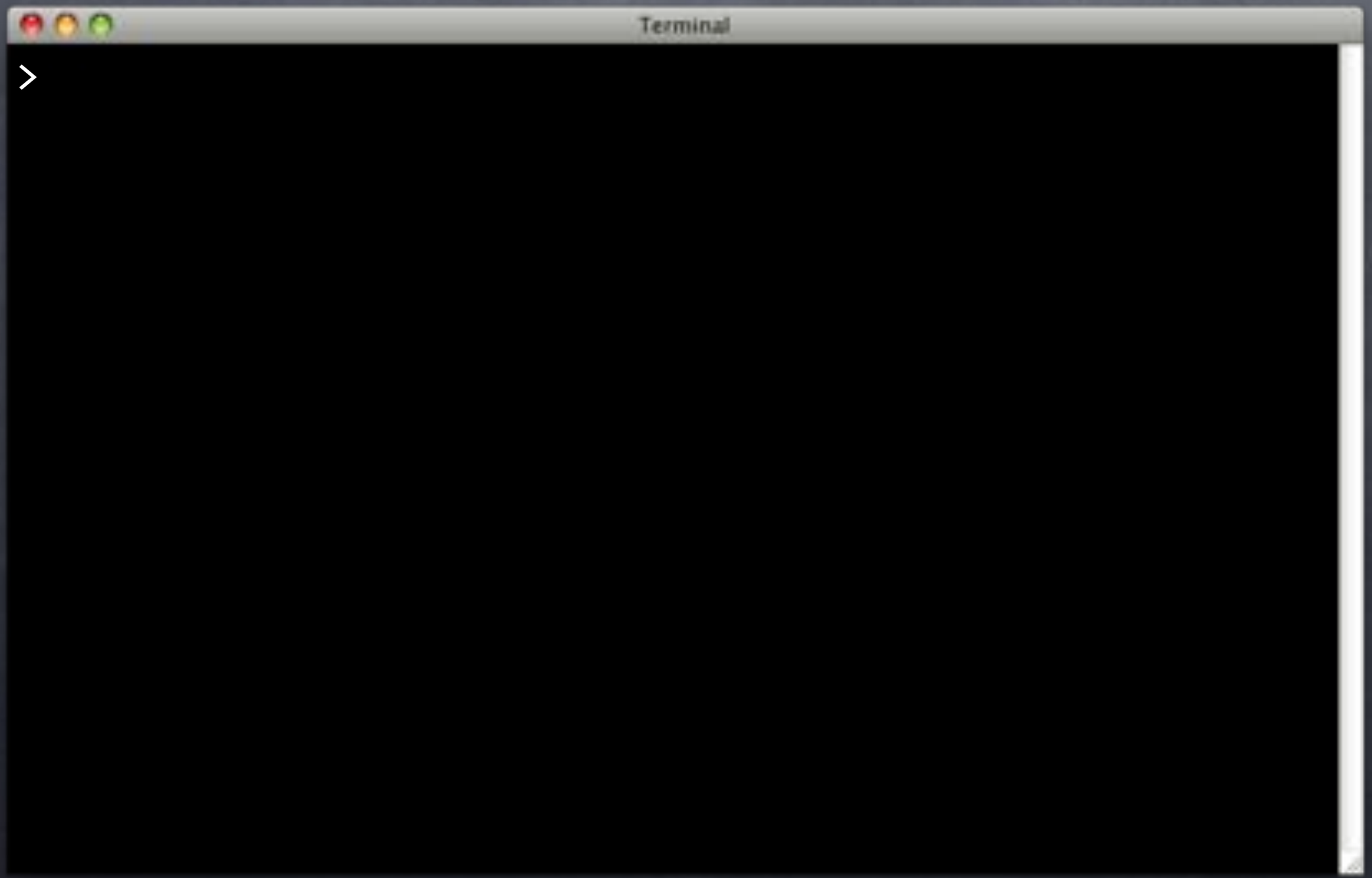
Rebase master



Rebase master



Reset



Reset


Terminal
> git reset --hard HEAD
HEAD is now at fbf8469 Add flips table.
>

Reset

Use with care!

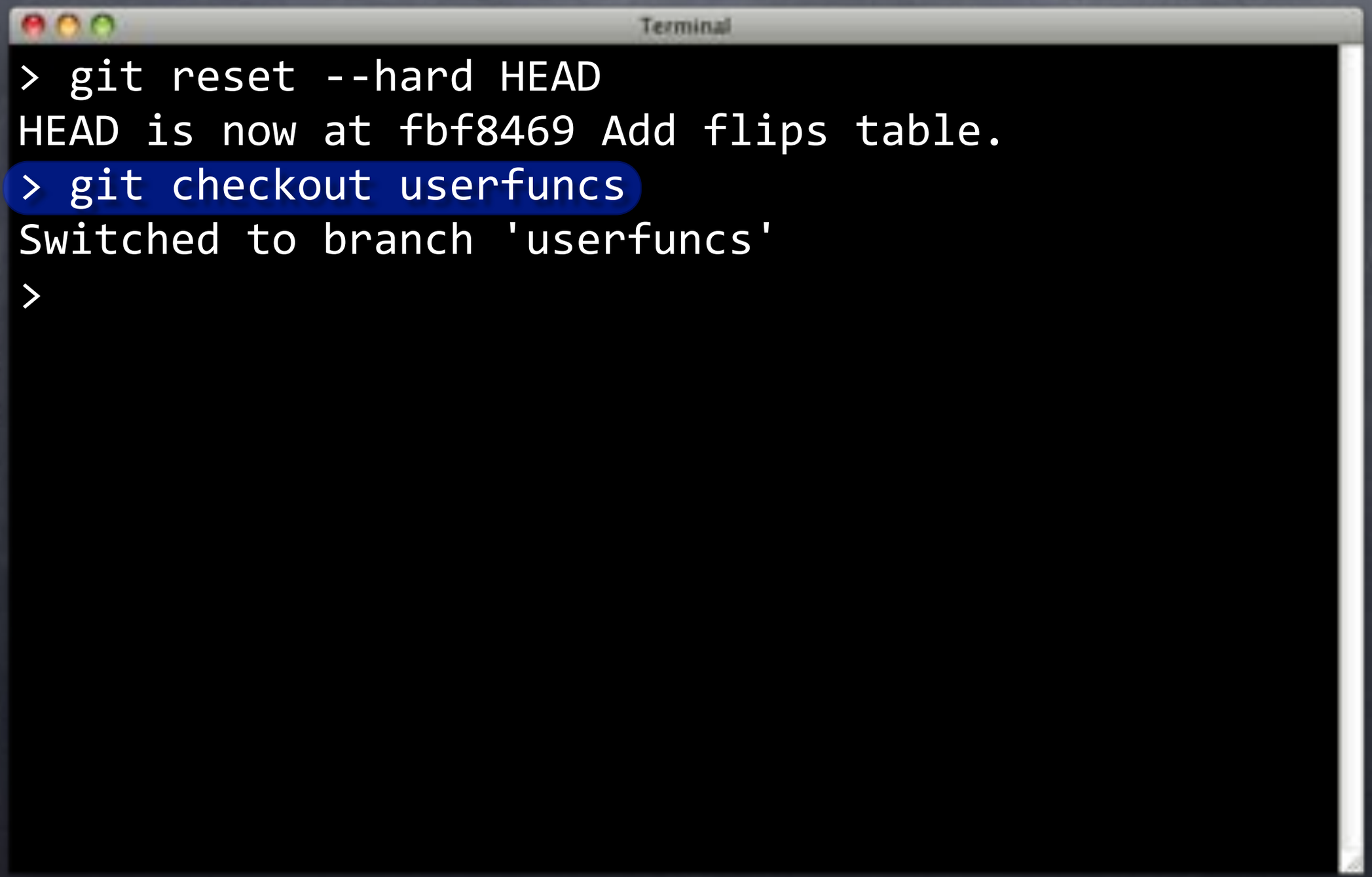
Terminal

```
> git reset --hard HEAD
HEAD is now at fbf8469 Add flips table.
>
```

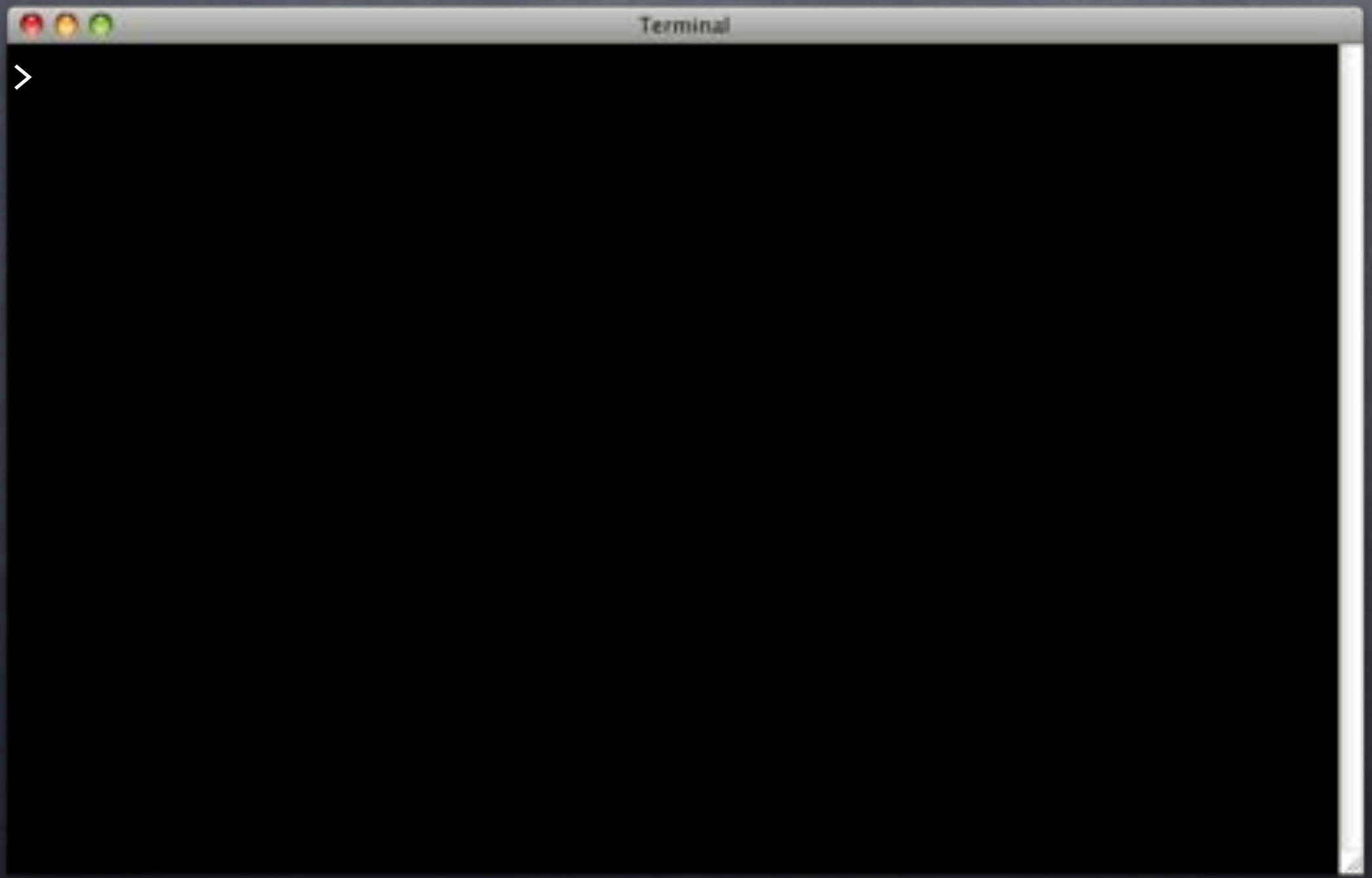
Reset


Terminal
> git reset --hard HEAD
HEAD is now at fbf8469 Add flips table.
>

Reset


Terminal
> git reset --hard HEAD
HEAD is now at fbf8469 Add flips table.
> git checkout userfuncs
Switched to branch 'userfuncs'
>

Rebase



Rebase

Terminal

```
> git rebase master
First, rewinding head to replay your work on top of it...
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
CONFLICT (content): Merge conflict in sqitch.plan
Failed to merge in the changes.
Patch failed at 0001 Add `insert_user()`.
The copy of the patch that failed is found in:
   /Users/david/Desktop/flipr-db/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --contin
If you prefer to skip this patch, run "git rebase --skip" ins
To check out the original branch and stop rebasing, run "git
rebase --abort".
>
```

Rebase

Back to B, apply C.

Terminal

```
> git rebase master
First, rewinding head to replay your work on top of it...
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
CONFLICT (content): Merge conflict in sqitch.plan
Failed to merge in the changes.
Patch failed at 0001 Add `insert_user()`.
The copy of the patch that failed is found in:
   /Users/david/Desktop/flipr-db/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --contin
If you prefer to skip this patch, run "git rebase --skip" ins
To check out the original branch and stop rebasing, run "git
rebase --abort".
>
```

Rebase

That's D

```
> git rebase master
First, rewinding head to replay your work on top of it...
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
CONFLICT (content): Merge conflict in sqitch.plan
Failed to merge in the changes.
Patch failed at 0001 Add `insert_user()`.
The copy of the patch that failed is found in:
   /Users/david/Desktop/flipr-db/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --contin
If you prefer to skip this patch, run "git rebase --skip" ins
To check out the original branch and stop rebasing, run "git
rebase --abort".
>
```

Rebase

Terminal

```
> git rebase master
First, rewinding head to replay yo
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
CONFLICT (content): Merge conflict in sqitch.plan
Failed to merge in the changes.
Patch failed at 0001 Add `insert_user()`.
The copy of the patch that failed is found in:
   /Users/david/Desktop/flipr-db/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --contin
If you prefer to skip this patch, run "git rebase --skip" ins
To check out the original branch and stop rebasing, run "git
rebase --abort".
>
```

Here comes D...

Rebase

Terminal

```
> git rebase master
First, rewinding head to replay your work on top of it...
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
CONFLICT (content): Merge conflict in sqitch.plan
Failed to merge in the changes.
Patch failed at 0001 Add `insert_user()`.
The copy of the patch that failed is found in:
   /Users/david/Desktop/flipr-db/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --contin
If you prefer to skip this patch, run "git rebase --skip" ins
To check out the original branch and stop rebasing, run "git
rebase --abort".
>
```

D'oh!

Rebase

Terminal

```
> git rebase master
First, rewinding head to replay your work on top of it...
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
CONFLICT (content): Merge conflict in sqitch.plan
Failed to merge in the changes.
Patch failed at 0001 Add `insert_user()`.
The copy of the patch that failed is found in:
   /Users/david/Desktop/flipr-db/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --contin
If you prefer to skip this patch, run "git rebase --skip" ins
To check out the original branch and stop rebasing, run "git
rebase --abort".
>
```

Wha Happen?

Wha Happen?

- Same conflict

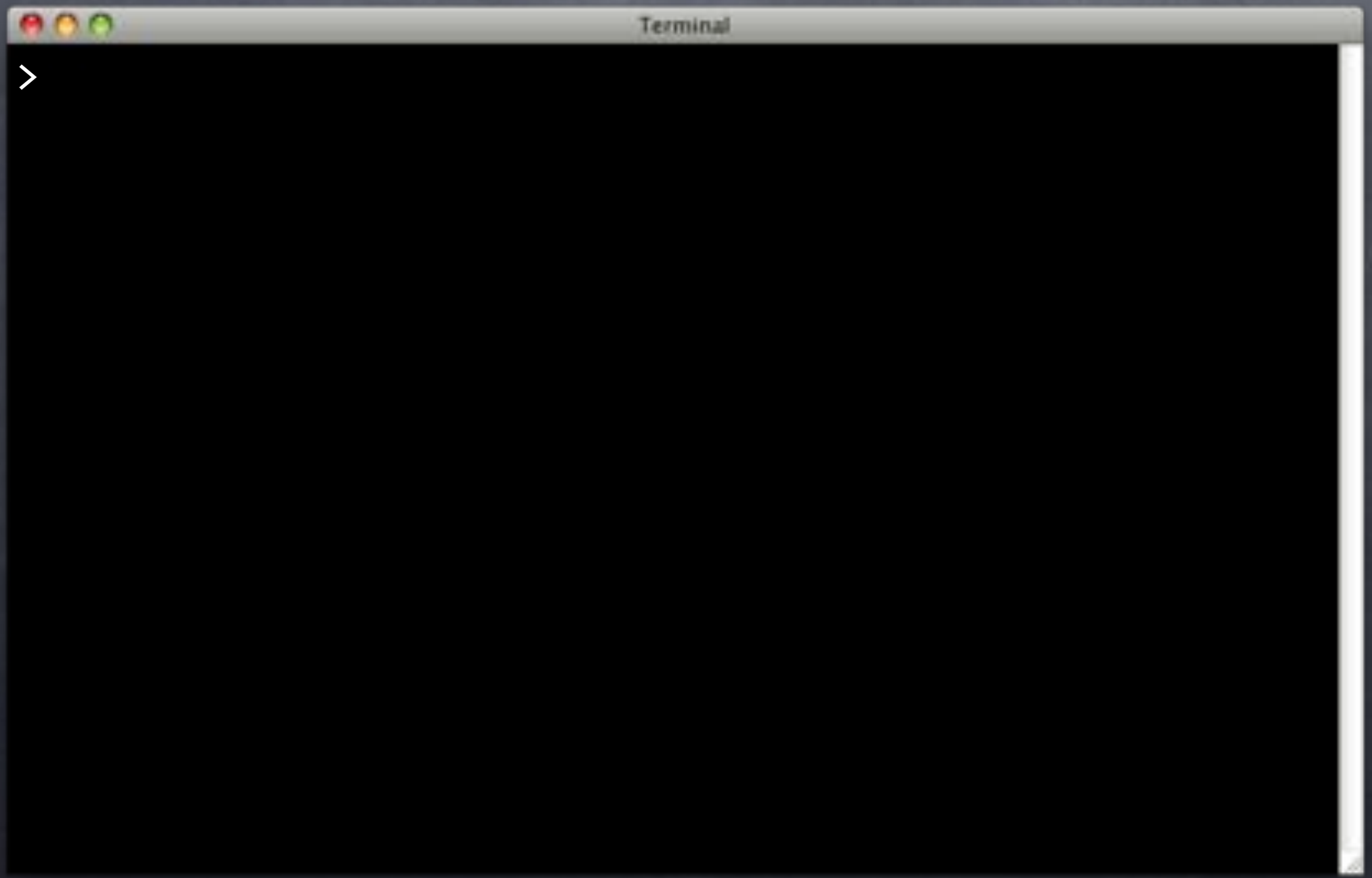
Wha Happen?

- Same conflict
- Both branches modified sqitch.plan

Wha Happen?

- Same conflict
- Both branches modified sqitch.plan
- The same line!

Wha Happen?



Wha Happen?

Terminal

```
> git diff
diff --cc sqitch.plan
index 0d5a515,96b1b7e..0000000
--- a/sqitch.plan
+++ b/sqitch.plan
@@@ -4,4 -4,4 +4,8 @@@

  appschema 2013-05-15T06:03:09Z David E. Wheeler <d
  users [appschema] 2013-05-17T04:59:01Z David E. Wh
++<<<<<<< HEAD
  flips [appschema users] 2013-05-18T00:04:56Z David
++=======
+ insert_user [users appschema] 2013-05-20T20:53:56Z
++>>>>>>> Add `insert_user()`.
>
```

Wha Happen?

Terminal

```
> git diff
diff --cc sqitch.plan
index 0d5a515,96b1b7e..0000000
--- a/sqitch.plan
+++ b/sqitch.plan
@@@ -4,4 -4,4 +4,8 @@@

  appschema 2013-05-15T06:03:09Z David E. Wheeler <d
  users [appschema] 2013-05-17T04:59:01Z David E. Wh
++<<<<<<< HEAD
  flips [appschema users] 2013-05-18T00:04:56Z David
++=======
+ insert_user [users appschema] 2013-05-20T20:53:56Z
++>>>>>>> Add `insert_user()`.
>
```

Wha Happen?

Terminal

```
> git diff
diff --cc sqitch.plan
index 0d5a515,96b1b7e..0000000
--- a/sqitch.plan
+++ b/sqitch.plan
@@@ -4,4 -4,4 +4,8 @@@

   appschema 2013-05-15T06:03:09Z David E. Wheeler <d
B  users [appschema] 2013-05-17T04:59:01Z David E. Wh
++<<<<<<< HEAD
   flips [appschema users] 2013-05-18T00:04:56Z David
++=======
+ insert_user [users appschema] 2013-05-20T20:53:56Z
++>>>>>>> Add `insert_user()`.
>
```

Wha Happen?

Terminal

```
> git diff
diff --cc sqitch.plan
index 0d5a515,96b1b7e..0000000
--- a/sqitch.plan
+++ b/sqitch.plan
@@@ -4,4 -4,4 +4,8 @@@

  appschema 2013-05-15T06:03:09Z David E. Wheeler <d
  users [appschema] 2013-05-17T04:59:01Z David E. Wh
++<<<<<<< HEAD
  flips [appschema users] 2013-05-18T00:04:56Z David
++=======
+ insert_user [users appschema] 2013-05-20T20:53:56Z
++>>>>>>> Add `insert_user()`.
>
```

B

C

Wha Happen?

Terminal

```
> git diff
diff --cc sqitch.plan
index 0d5a515,96b1b7e..0000000
--- a/sqitch.plan
+++ b/sqitch.plan
@@@ -4,4 -4,4 +4,8 @@@

   appschema 2013-05-15T06:03:09Z David E. Wheeler <d
B  users [appschema] 2013-05-17T04:59:01Z David E. Wh
++<<<<<<< HEAD
C  flips [appschema users] 2013-05-18T00:04:56Z David
++=======
D  + insert_user [users appschema] 2013-05-20T20:53:56Z
++>>>>>>> Add `insert_user()`.
>
```

Scratch that Sqitch

Scratch that Sqitch

- Screwed either way?

Scratch that Sqitch

- Screwed either way?
- Fortunately, this is Git

Scratch that Sqitch

- Screwed either way?
- Fortunately, this is Git
- Tell it to treat Sqitch plans differently

Scratch that Sqitch

- Screwed either way?
- Fortunately, this is Git
- Tell it to treat Sqitch plans differently
- Changes on single lines

Scratch that Sqitch

- Screwed either way?
- Fortunately, this is Git
- Tell it to treat Sqitch plans differently
- Changes on single lines
- Only appended to plan file

Scratch that Sqitch

- Screwed either way?
- Fortunately, this is Git
- Tell it to treat Sqitch plans differently
- Changes on single lines
- Only appended to plan file
- Use the “union” merge

Re: Union Merge

Re: Union Merge

Run 3-way file level merge for text files, but take lines from both versions, instead of leaving conflict markers. This tends to leave the added lines in the resulting file in random order and the user should verify the result. Do not use this if you do not understand the implications.

—Git Manual

Hallelunion

Hallelunion

- Just appends lines

Hallelunion

- Just appends lines
- Exactly how changes work

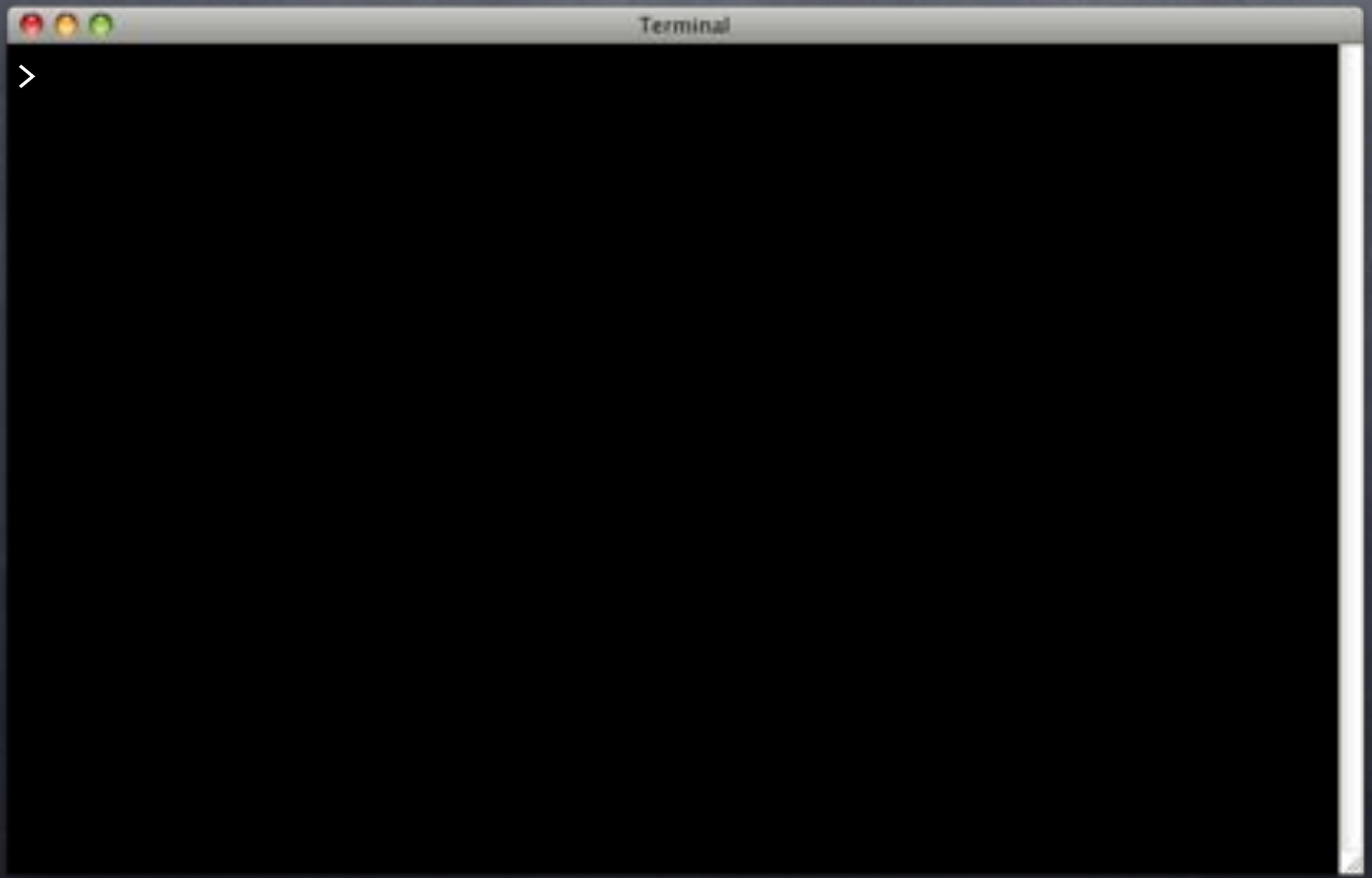
Hallelunion

- Just appends lines
- Exactly how changes work
- Let's clean up our mess

Hallelunion

- Just appends lines
- Exactly how changes work
- Let's clean up our mess
- And try again

Reemerge



Reemerge

Reemerge

Terminal

```
> git rebase --abort
> echo sqitch.plan merge=union > .gitattributes
>
```

Reemerge

Terminal

```
> git rebase --abort
> echo sqitch.plan merge=union > .gitattributes
> git rebase master
First, rewinding head to replay your work on top of it..
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M  sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
Applying: Add `change_pass()`.
Using index info to reconstruct a base tree...
M  sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
>
```

Reemerge

Back to B, applies C

```
> git rebase --abort
> echo sqitch.plan merg                    ibutes
> git rebase master
First, rewinding head to replay your work on top of it..
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
Applying: Add `change_pass()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
>
```

Reemerge

Terminal

```
> git rebase --abort
> echo sqitch.plan merge=u                    es
> git rebase master
First, rewinding head to replay your work on top of it...
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
Applying: Add `change_pass()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
>
```

That's D

Reemerge

Terminal

```
> git rebase --abort
> echo sqitch.plan merge=union > .gitattributes
> git rebase master
First, rewinding head to replay your w
Applying: Add `insert_user()`.
Using index info to reconstruct a base
M  sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
Applying: Add `change_pass()`.
Using index info to reconstruct a base tree...
M  sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
>
```

Union merge

Reemerge

Terminal

```
> git rebase --abort
> echo sqitch.plan merge=union > .gitattributes
> git rebase master
First, rewinding head to replay your work on top of it..
Applying: Add `insert_user()`.
Using index info to reconstr                    ..
M   sqitch.plan
Falling back to patching b                      ...
Auto-merging sqitch.plan
Applying: Add `change_pass()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
>
```

That's E

Reemerge

Terminal

```
> git rebase --abort
> echo sqitch.plan merge=union > .gitattributes
> git rebase master
First, rewinding head to replay your work on top of it..
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
Applying: Add `change_pass()`.
Using index info to reconstruct a base
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
>
```

Union merge

Reemerge

Terminal

```
> git rebase --abort
> echo sqitch.plan merge=union > .gitattributes
> git rebase master
First, rewinding head to replay your work on top of it..
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
Applying: Add `change_pass()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
>
```

Success!

Reemerge

Terminal

```
> git rebase --abort
> echo sqitch.plan merge=union > .gitattributes
> git rebase master
First, rewinding head to replay your work on top of it..
Applying: Add `insert_user()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
Applying: Add `change_pass()`.
Using index info to reconstruct a base tree...
M   sqitch.plan
Falling back to patching base and 3-way merge...
Auto-merging sqitch.plan
> emacs sqitch.plan
>
```

Success!

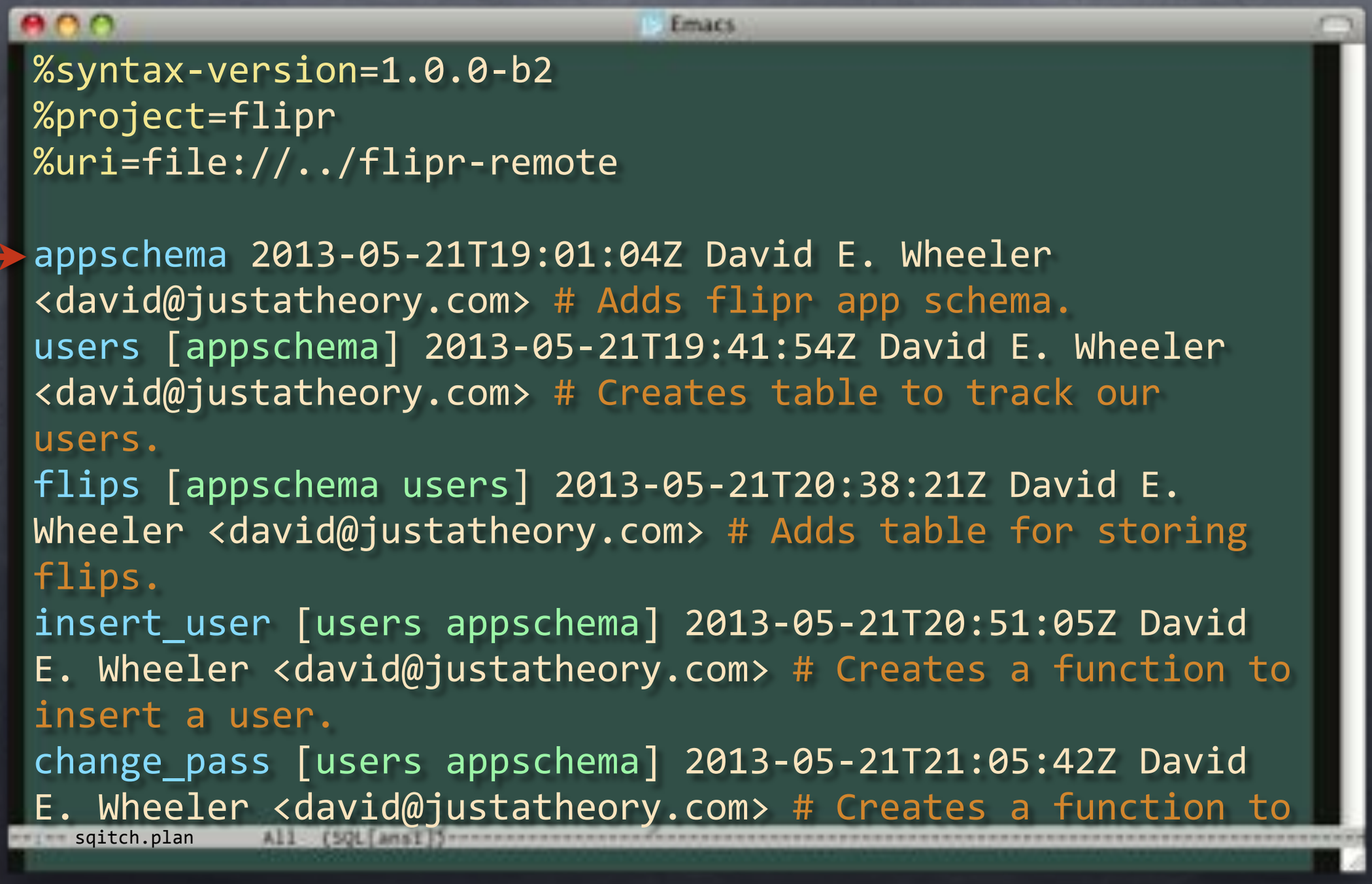
What's the Plan, Man?

Emacs

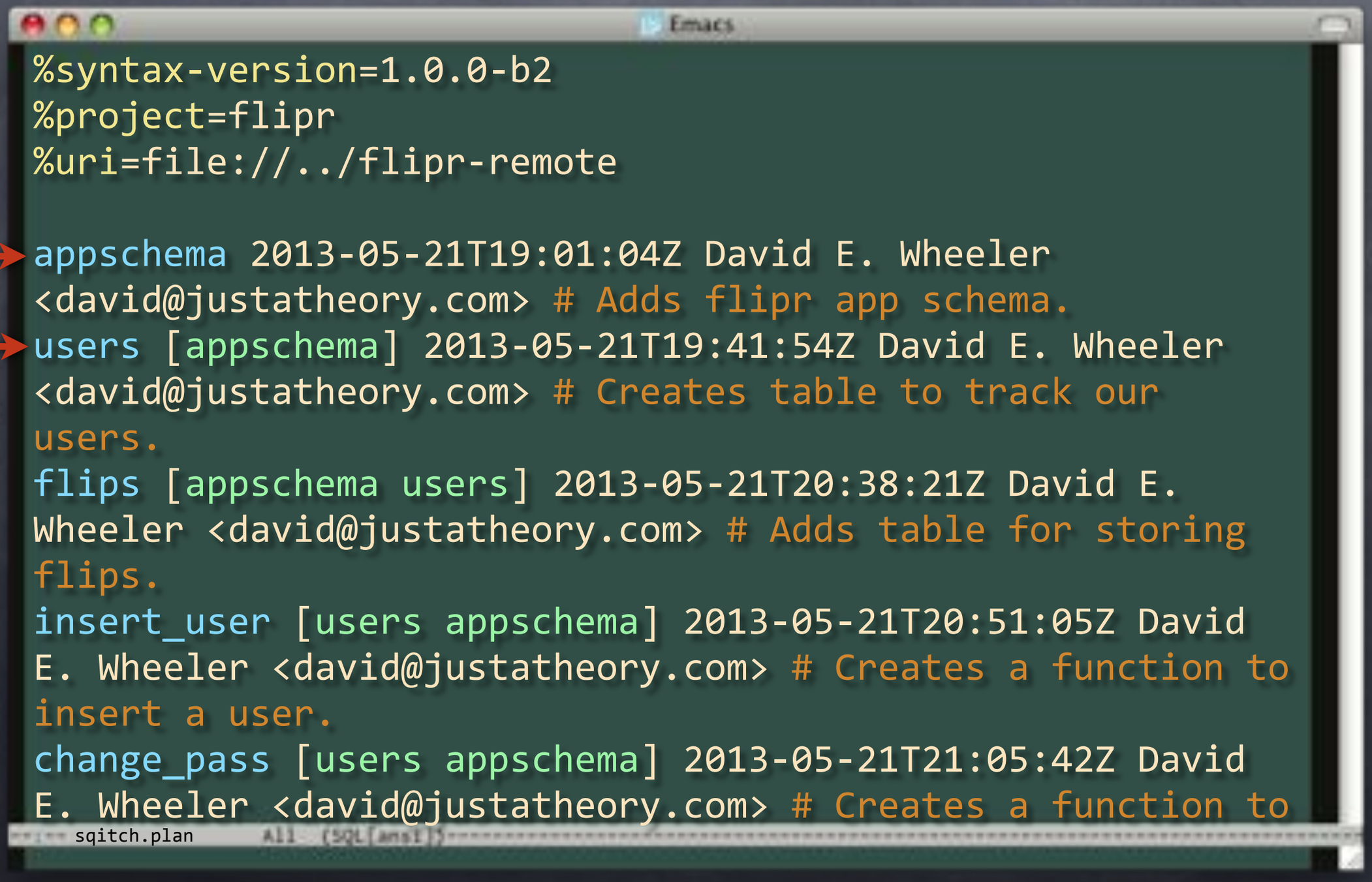
```
%syntax-version=1.0.0-b2
%project=flipr
%uri=file://../flipr-remote

appschema 2013-05-21T19:01:04Z David E. Wheeler
<david@justatheory.com> # Adds flipr app schema.
users [appschema] 2013-05-21T19:41:54Z David E. Wheeler
<david@justatheory.com> # Creates table to track our
users.
flips [appschema users] 2013-05-21T20:38:21Z David E.
Wheeler <david@justatheory.com> # Adds table for storing
flips.
insert_user [users appschema] 2013-05-21T20:51:05Z David
E. Wheeler <david@justatheory.com> # Creates a function to
insert a user.
change_pass [users appschema] 2013-05-21T21:05:42Z David
E. Wheeler <david@justatheory.com> # Creates a function to
```

sqitch.plan All (SQL[ansi])

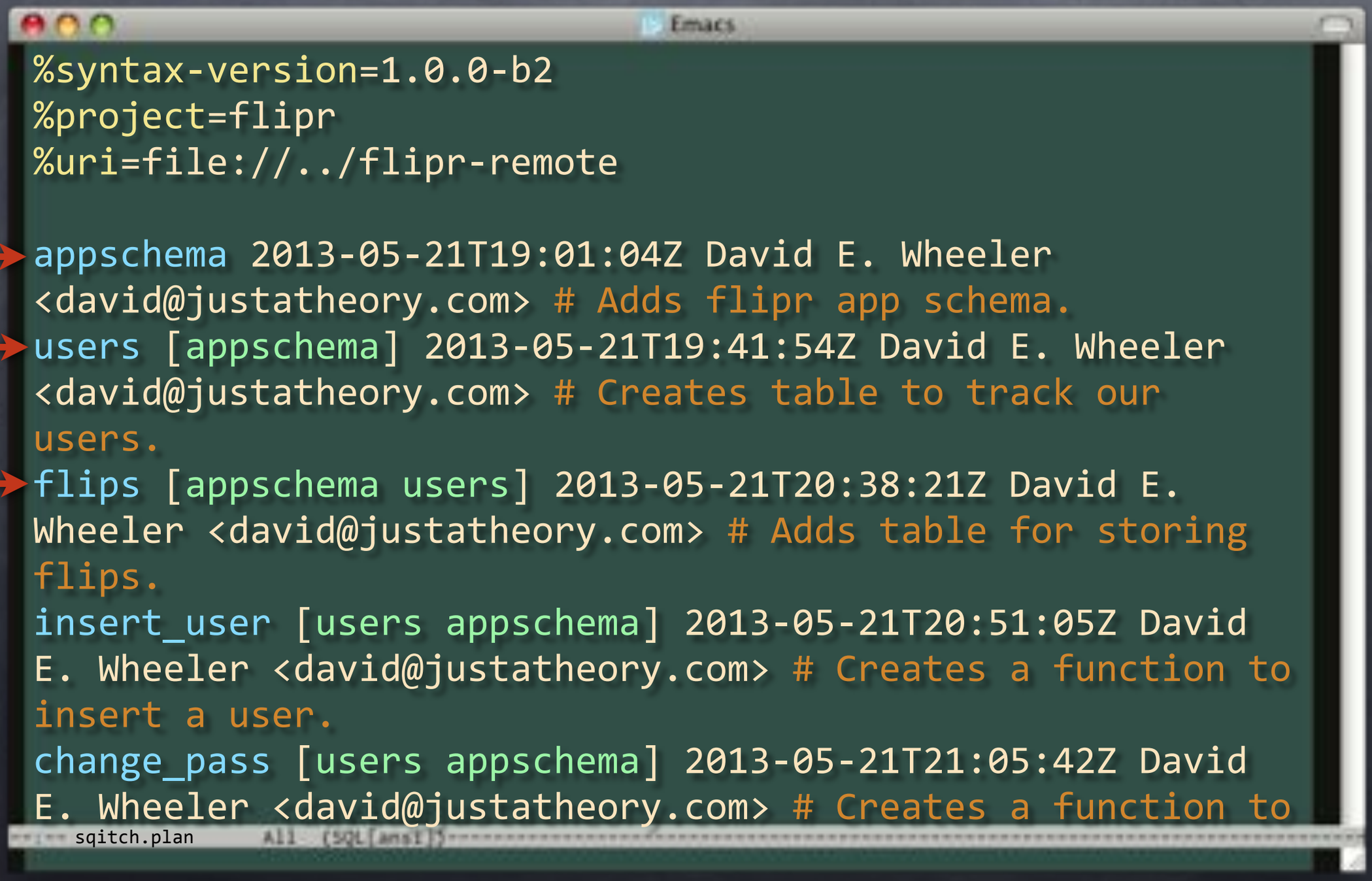
What's the Plan, Man?



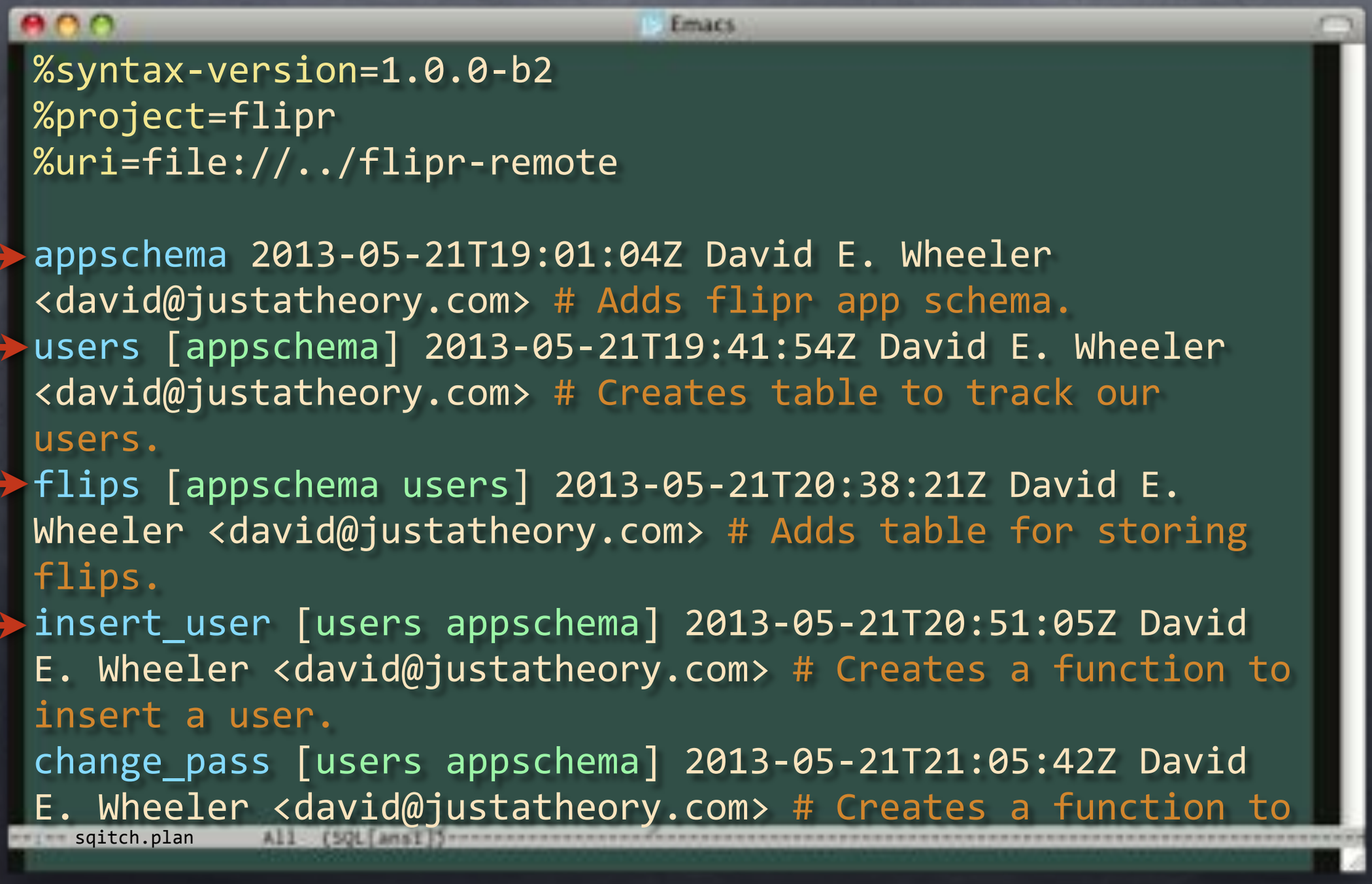
What's the Plan, Man?



What's the Plan, Man?



What's the Plan, Man?



What's the Plan, Man?

Emacs

```
%syntax-version=1.0.0-b2
%project=flipr
%uri=file://../flipr-remote

appschema 2013-05-21T19:01:04Z David E. Wheeler
<david@justatheory.com> # Adds flipr app schema.
users [appschema] 2013-05-21T19:41:54Z David E. Wheeler
<david@justatheory.com> # Creates table to track our
users.
flips [appschema users] 2013-05-21T20:38:21Z David E.
Wheeler <david@justatheory.com> # Adds table for storing
flips.
insert_user [users appschema] 2013-05-21T20:51:05Z David
E. Wheeler <david@justatheory.com> # Creates a function to
insert a user.
change_pass [users appschema] 2013-05-21T21:05:42Z David
E. Wheeler <david@justatheory.com> # Creates a function to
```

sqitch.plan All (SQL[ansi])

What's the Plan, Man?

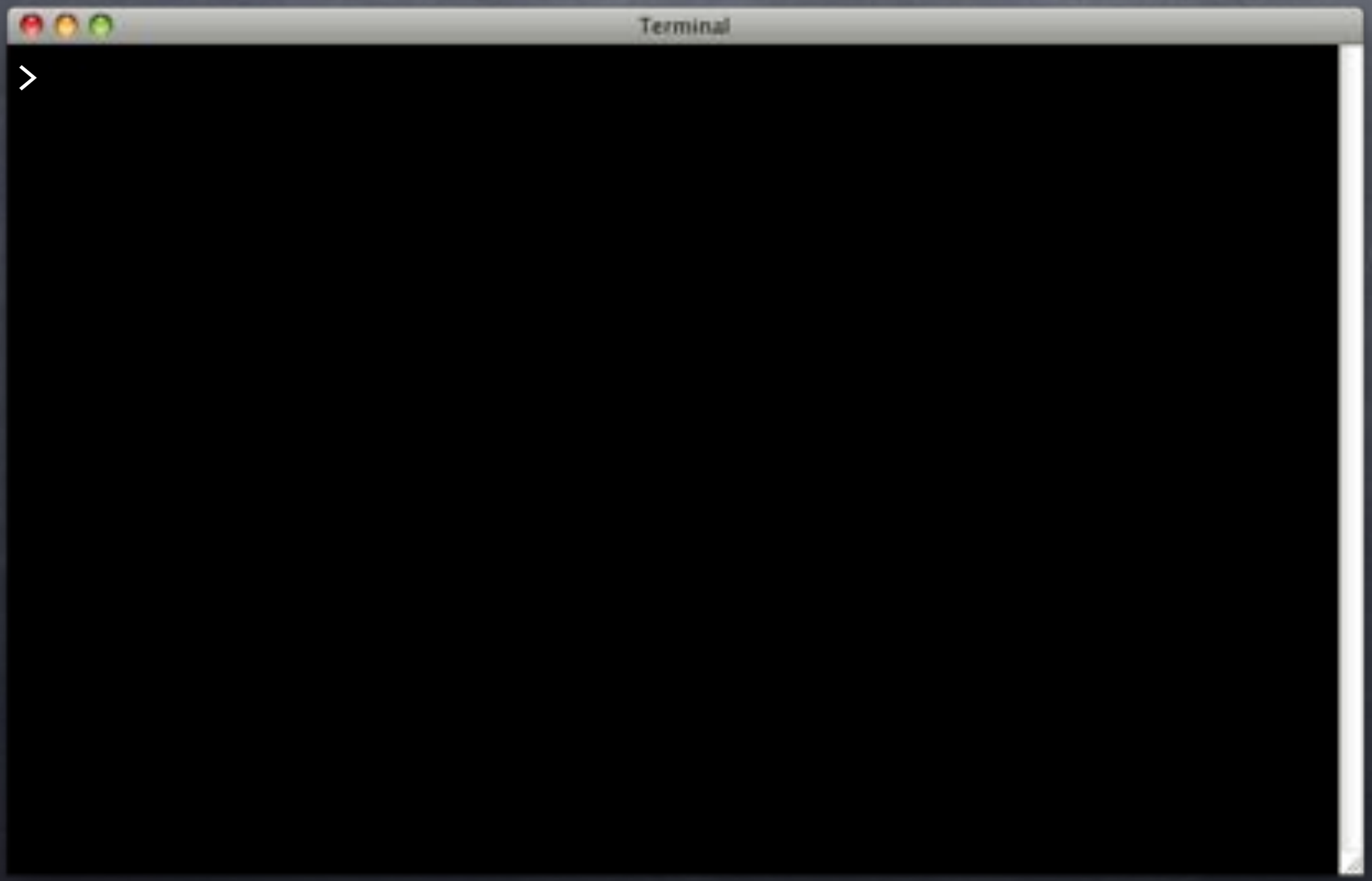
Emacs

```
%syntax-version=1.0.0-b2
%project=flipr
%uri=file://../flipr-remote

appschema 2013-05-21T19:01:04Z David E. Wheeler
<david@justatheory.com> # Adds flipr app schema.
users [appschema] 2013-05-21T19:41:54Z David E. Wheeler
<david@justatheory.com> # Creates table to track our
users.
flips [appschema users] 2013-05-21T20:38:21Z David E.
Wheeler <david@justatheory.com> # Adds table for storing
flips.
insert_user [users appschema] 2013-05-21T20:51:05Z David
E. Wheeler <david@justatheory.com> # Creates a function to
insert a user.
change_pass [users appschema] 2013-05-21T21:05:42Z David
E. Wheeler <david@justatheory.com> # Creates a function to
```

sqitch.plan All (SQL[ansi])

Perfect

Work It

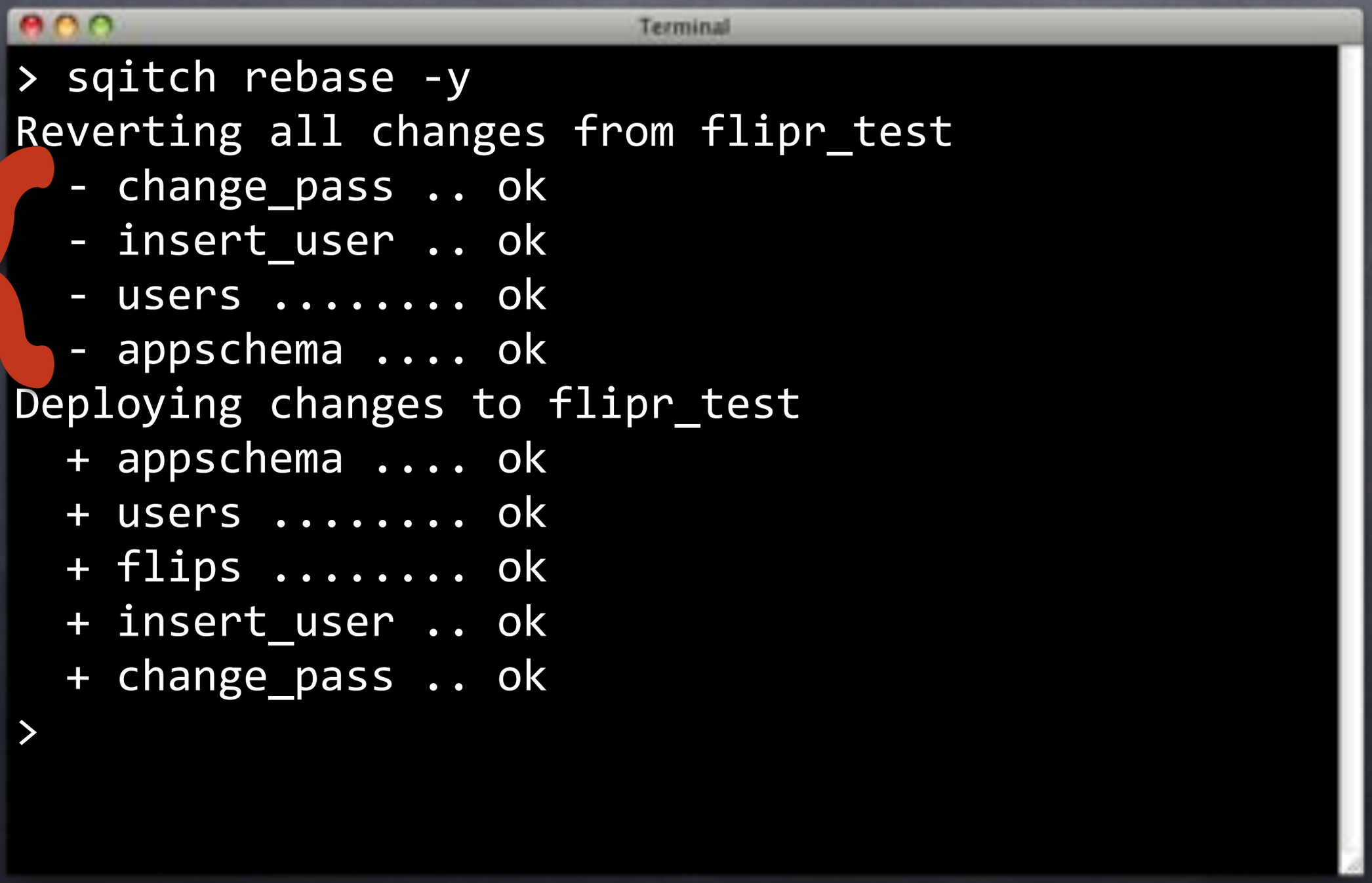


Work It

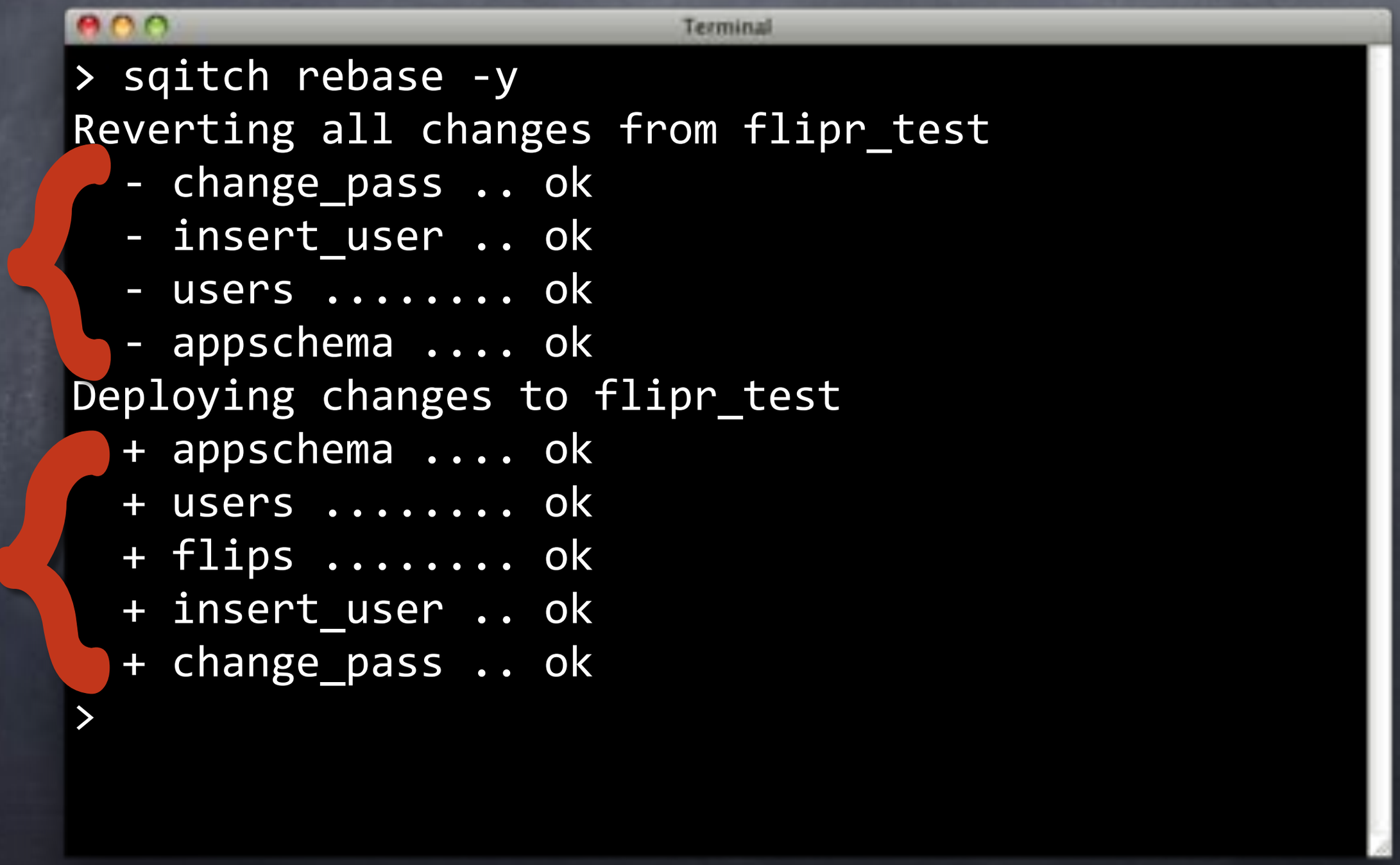
Terminal

```
> sqitch rebase -y
Reverting all changes from flipr_test
  - change_pass .. ok
  - insert_user .. ok
  - users ........ ok
  - appschema .... ok
Deploying changes to flipr_test
  + appschema .... ok
  + users ........ ok
  + flips ........ ok
  + insert_user .. ok
  + change_pass .. ok
>
```

Work It


Terminal
> sqitch rebase -y
Reverting all changes from flipr_test
- change_pass .. ok
- insert_user .. ok
- users ..... ok
- appschema .... ok
Deploying changes to flipr_test
+ appschema .... ok
+ users ..... ok
+ flips ..... ok
+ insert_user .. ok
+ change_pass .. ok
>

Work It


Terminal
> sqitch rebase -y
Reverting all changes from flipr_test
- change_pass .. ok
- insert_user .. ok
- users ..... ok
- appschema .... ok
Deploying changes to flipr_test
+ appschema .... ok
+ users ..... ok
+ flips ..... ok
+ insert_user .. ok
+ change_pass .. ok
>

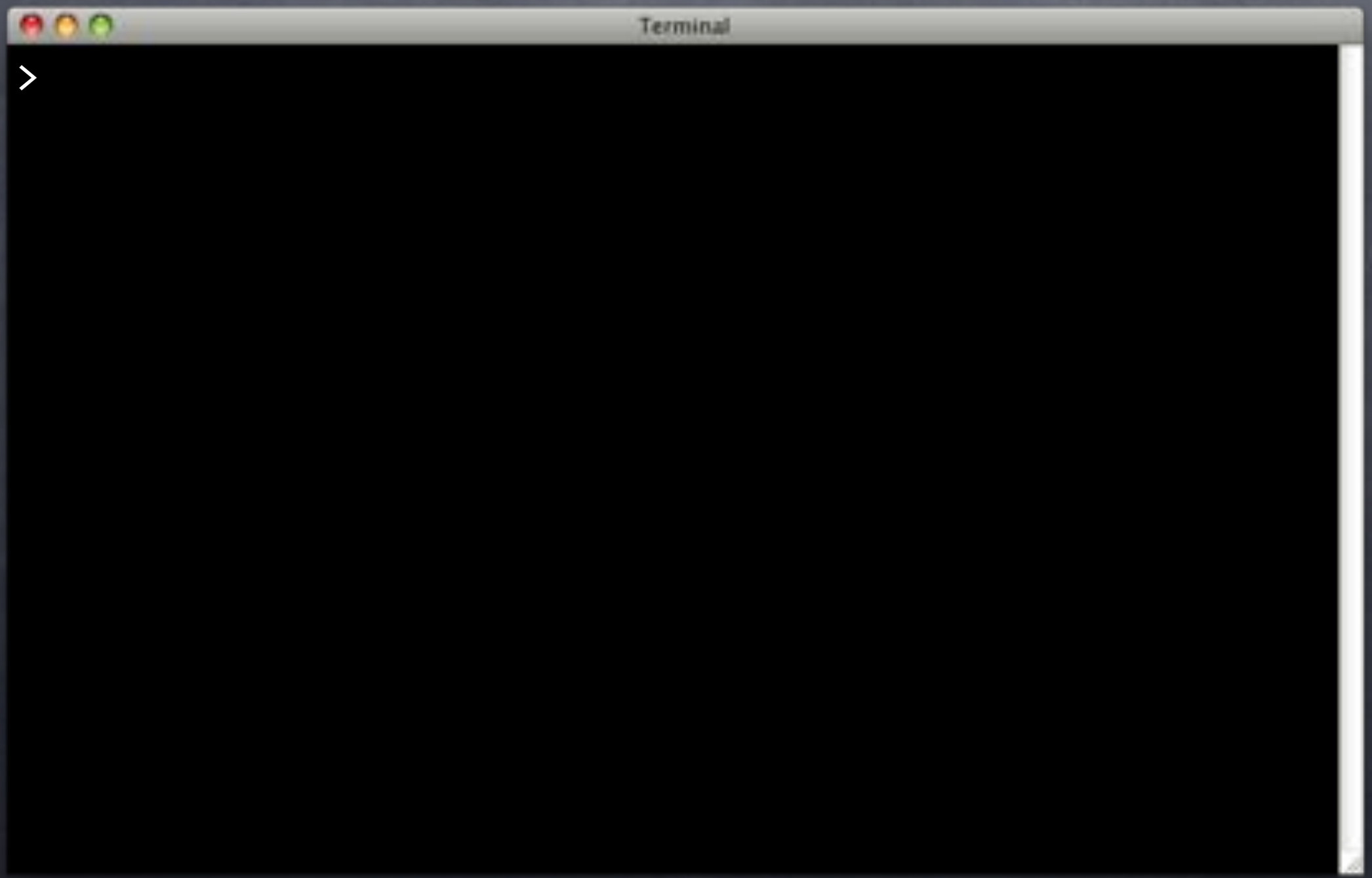
Work It

Terminal

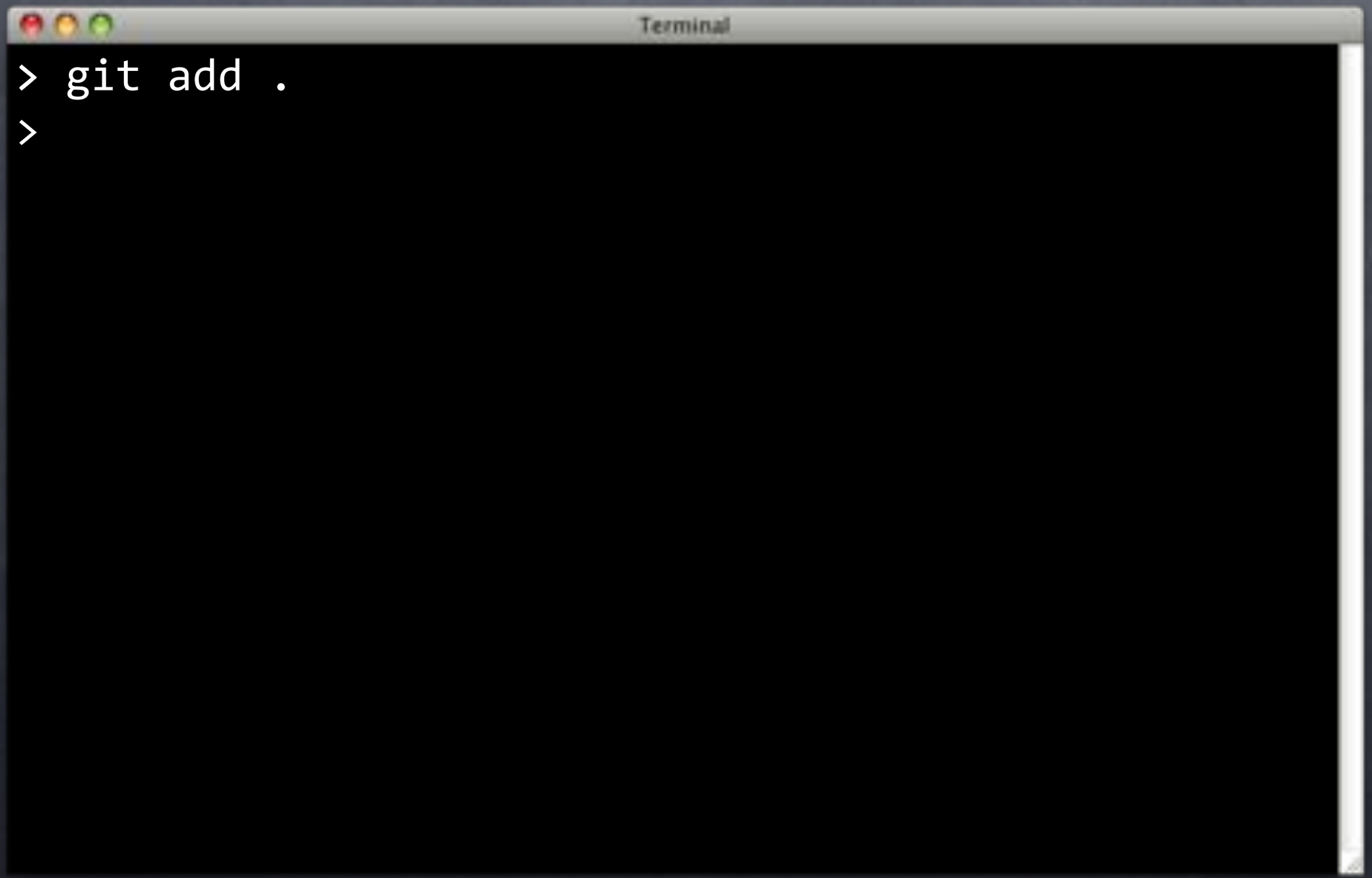
```
> sqitch rebase -y
Reverting all changes from flipr_test
  - change_pass .. ok
  - insert_user .. ok
  - users ........ ok
  - appschema .... ok
Deploying changes to flipr_test
  + appschema .... ok
  + users ........ ok
  + flips ........ ok
  + insert_user .. ok
  + change_pass .. ok
>
```

Awesomsauce

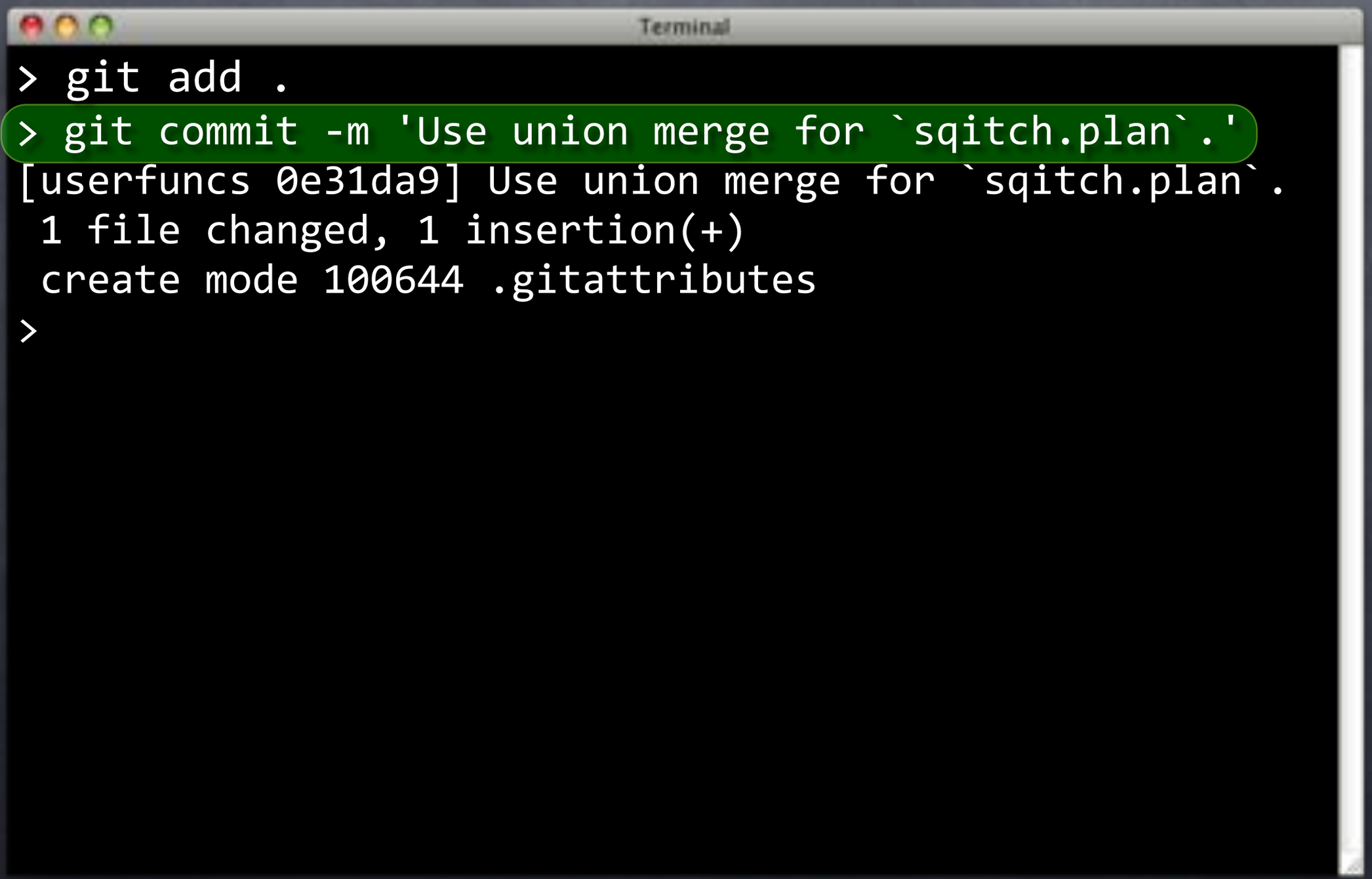
Make it So



Make it So



Make it So


Terminal
> git add .
> git commit -m 'Use union merge for `sqitch.plan`.'
[userfuncs 0e31da9] Use union merge for `sqitch.plan`.
 1 file changed, 1 insertion(+)
 create mode 100644 .gitattributes
>

Make it So

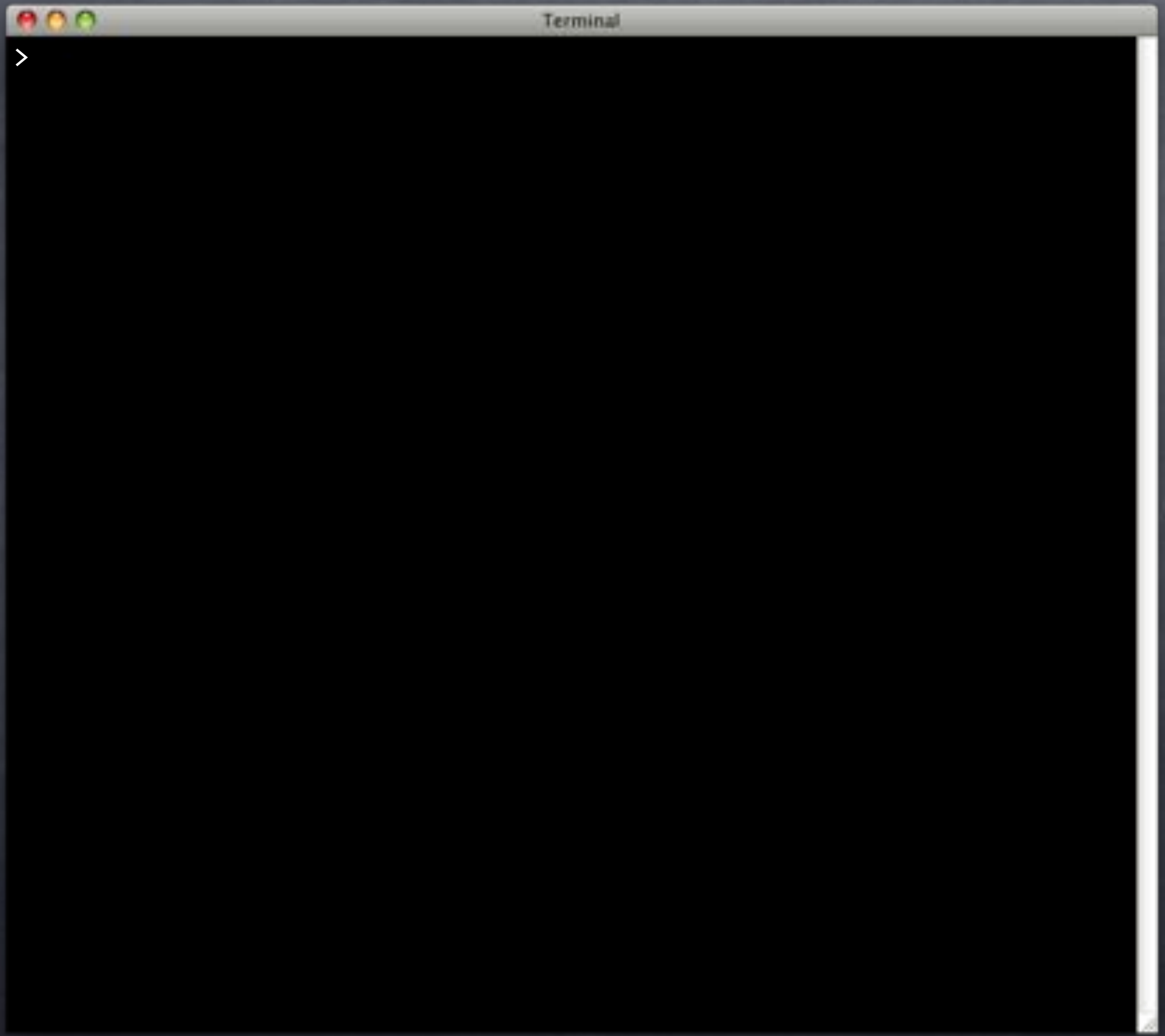
Overwrite history

```
> git add .
> git commit                                  ge for `sqitch.plan`.'
[userfuncs (                                  merge for `sqitch.plan`.
 1 file chan                                  4)
 create mode 100644 .gitattributes
> git push --force origin userfuncs
Counting objects: 31, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (24/24), done.
Writing objects: 100% (25/25), 3.83 KiB, done.
Total 25 (delta 7), reused 0 (delta 0)
To ../flipr-remote
 + e7477c...0e31da userfuncs -> userfuncs (forced update
>
```

Make it So

Terminal

```
> git add .
> git commit -m 'Use union merge for `sqitch.plan`.'
[userfuncs 0e31da9] Use union merge for `sqitch.plan`.
 1 file changed, 1 insertion(+)
 create mode 100644 .gitattributes
> git push --force origin userfuncs
Counting objects: 31, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (24/24), done.
Writing objects: 100% (25/25), 3.83 KiB, done.
Total 25 (delta 7), reused 0 (delta 0)
To ../flipr-remote
 + e7477c...0e31da userfuncs -> userfuncs (forced update
>
```


Terminal
>

Terminal

```
> git checkout master
Switched to branch 'master'
>
```

Terminal

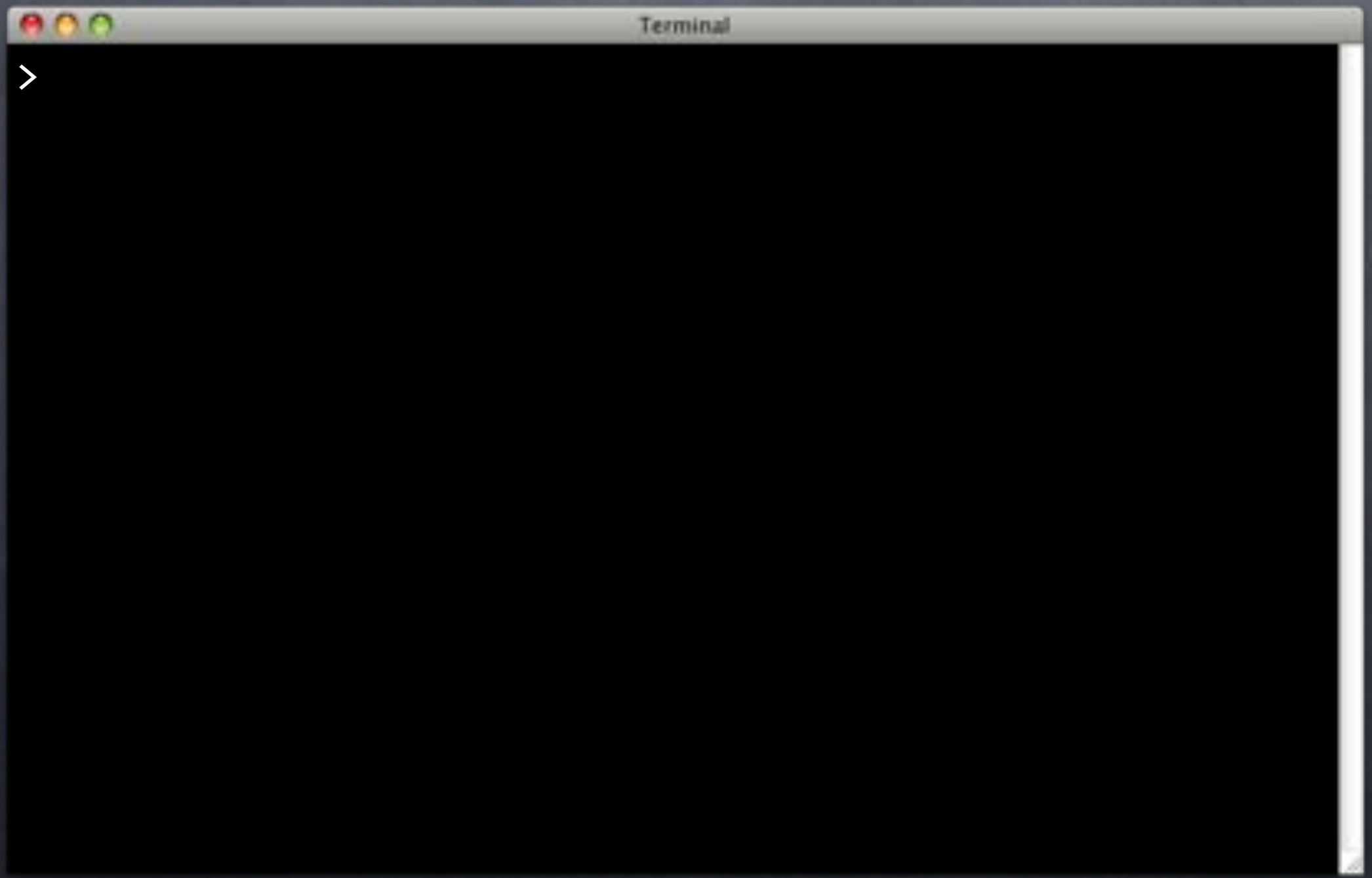
```
> git checkout master
Switched to branch 'master'
> git merge userfuncs
Updating fbf8469..0e31da9
Fast-forward
 .gitattributes          |  1 +
 deploy/change_pass.sql  | 21 ++++++++++++++++++++
 deploy/insert_user.sql  | 14 ++++++++++++
 revert/change_pass.sql  |  7 ++++++
 revert/insert_user.sql  |  7 ++++++
 sqitch.plan             |  2 ++
 test/change_pass.sql    | 52 ++++++++++++++++++++++++++++++++++++++++++++++++
 test/insert_user.sql    | 69 ++++++++++++++++++++++++++++++++++++++++++++++++++++++
 verify/change_pass.sql  |  7 ++++++
 verify/insert_user.sql  | 10 +++++++++
 10 files changed, 190 insertions(+)
 create mode 100644 .gitattributes
 create mode 100644 deploy/change_pass.sql
 create mode 100644 deploy/insert_user.sql
 create mode 100644 revert/change_pass.sql
 create mode 100644 revert/insert_user.sql
 create mode 100644 test/change_pass.sql
 create mode 100644 test/insert_user.sql
 create mode 100644 verify/change_pass.sql
 create mode 100644 verify/insert_user.sql
>
```

Terminal

```
> git checkout master
Switched to branch 'master'
> git merge userfuncs
Updating fbf8469..0e31da9
Fast-forward
 .gitattributes          |  1 +
 deploy/change_pass.sql  | 21 ++++++++++++++++++++
 deploy/insert_user.sql  | 14 ++++++++++++
 revert/change_pass.sql  |  7 ++++++
 revert/insert_user.sql  |  7 ++++++
 sqitch.plan             |  2 ++
 test/change_pass.sql    | 52 ++++++++++++++++++++++++++++++++++++++++++++++++
 test/insert_user.sql    | 69 ++++++++++++++++++++++++++++++++++++++++++++++++++++++
 verify/change_pass.sql  |  7 ++++++
 verify/insert_user.sql  | 10 +++++++++
 10 files changed, 190 insertions(+)
 create mode 100644 .gitattributes
 create mode 100644 deploy/change_pass.sql
 create mode 100644 deploy/insert_user.sql
 create mode 100644 revert/change_pass.sql
 create mode 100644 revert/insert_user.sql
 create mode 100644 test/change_pass.sql
 create mode 100644 test/insert_user.sql
 create mode 100644 verify/change_pass.sql
 create mode 100644 verify/insert_user.sql
>
```

\0/

Pusher



Pusher

Terminal

```
> git push
Counting objects: 20, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (14/14), 2.31 KiB, done.
Total 14 (delta 2), reused 0 (delta 0)
To ../flipr-remote
   fbf8469..0e31da9  master -> master
>
```


Flipr
antisocial network

Ship it!

Ship Shape

Ship Shape

- Good work so far

Ship Shape

- Good work so far
- People gonna flip out

Ship Shape

- Good work so far
- People gonna flip out
- Let's tag a dev release

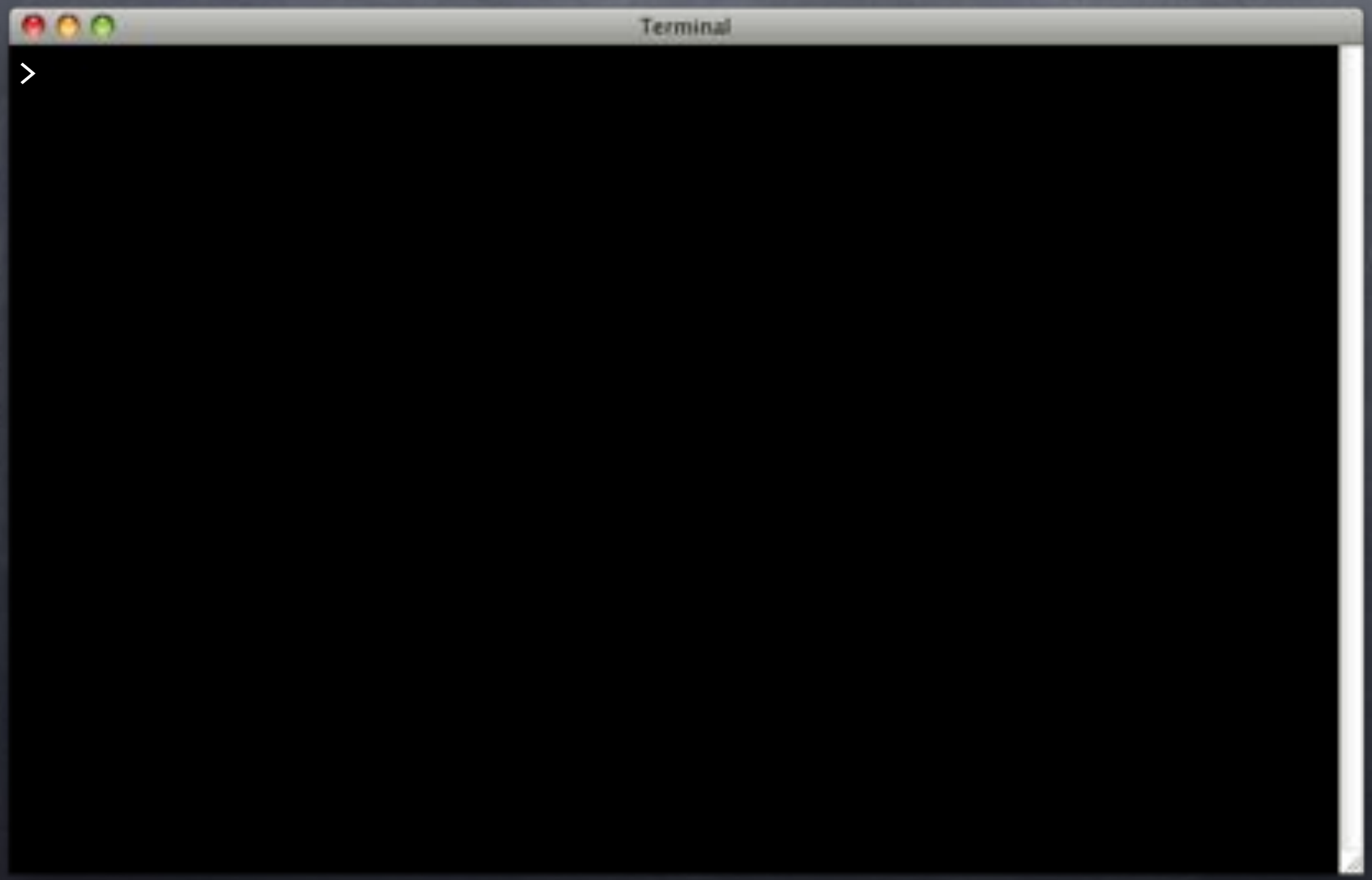
Ship Shape

- Good work so far
- People gonna flip out
- Let's tag a dev release
- Bundle it up

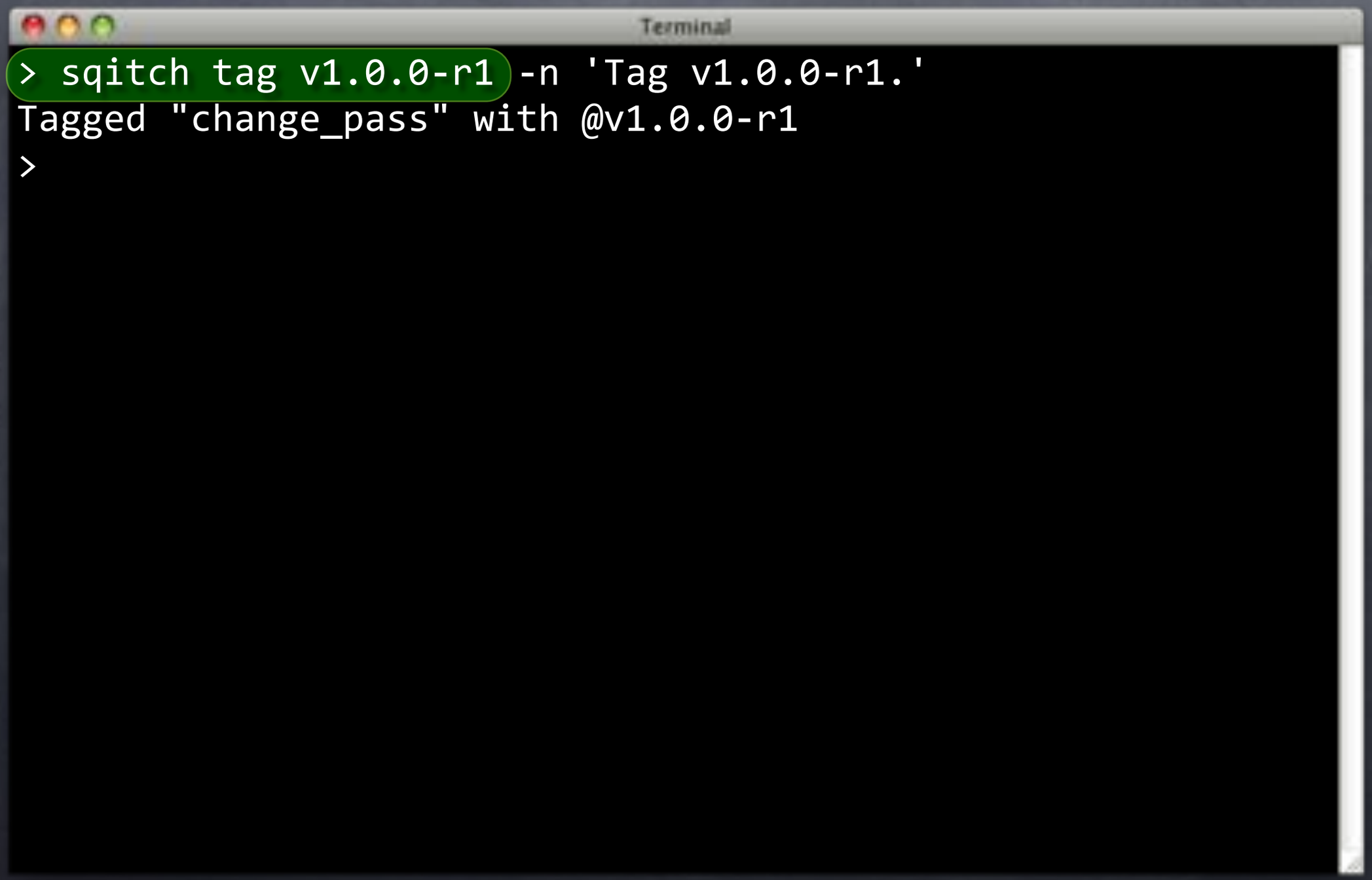
Ship Shape

- Good work so far
- People gonna flip out
- Let's tag a dev release
- Bundle it up
- And ship it

You're It



You're It


Terminal
> sqitch tag v1.0.0-r1 -n 'Tag v1.0.0-r1.'
Tagged "change_pass" with @v1.0.0-r1
>

You're It

Terminal

```
> sqitch tag v1.0.0-r1 -n 'Tag v1.0.0-r1.'
Tagged "change_pass" with @v1.0.0-r1
> git commit -am 'Tag the database @v1.0.0-r1.'
[master 6661268] Tag the database @v1.0.0-r1.
 1 file changed, 1 insertion(+)
>
```

You're It

Terminal

```
> sqitch tag v1.0.0-r1 -n 'Tag v1.0.0-r1.'
Tagged "change_pass" with @v1.0.0-r1
> git commit -am 'Tag the database @v1.0.0-r1.'
[master 6661268] Tag the database @v1.0.0-r1.
 1 file changed, 1 insertion(+)
> git tag v1.0.0-r1 -am 'Tag v1.0.0-r1.'
>
```

In sync.

You're It

Terminal

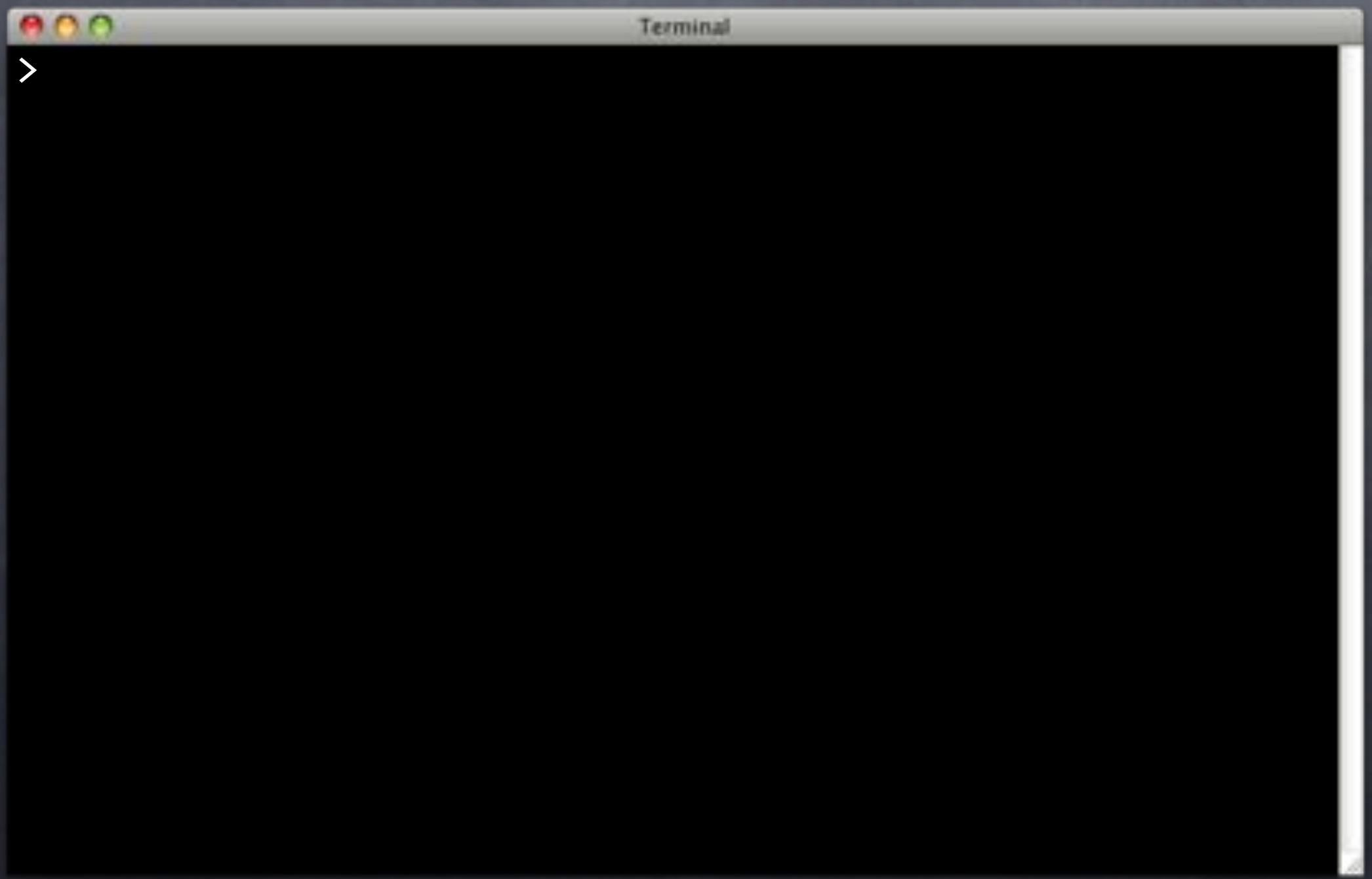
```
> sqitch tag v1.0.0-r1 -n 'Tag v1.0.0-r1.'
Tagged "change_pass" with @v1.0.0-r1
> git commit -am 'Tag the database @v1.0.0-r1.'
[master 6661268] Tag the database @v1.0.0-r1.
 1 file changed, 1 insertion(+)
> git tag v1.0.0-r1 -am 'Tag v1.0.0-r1.'
> git push
Counting objects: 5, done.
Writing objects: 100% (3/3), 346 bytes, done.
Total 3 (delta 2), reused 0 (delta 0)
To ../flipr-remote
   0e31da9..6661268  master -> master
>
```

You're It

Terminal

```
> sqitch tag v1.0.0-r1 -n 'Tag v1.0.0-r1.'
Tagged "change_pass" with @v1.0.0-r1
> git commit -am 'Tag the database @v1.0.0-r1.'
[master 6661268] Tag the database @v1.0.0-r1.
 1 file changed, 1 insertion(+)
> git tag v1.0.0-r1 -am 'Tag v1.0.0-r1.'
> git push
Counting objects: 5, done.
Writing objects: 100% (3/3), 346 bytes, done.
Total 3 (delta 2), reused 0 (delta 0)
To ../flipr-remote
   0e31da9..6661268  master -> master
> git push --tags
To ../flipr-remote
 * [new tag]         v1.0.0-r1 -> v1.0.0-r1
>
```

Bundle Up



Bundle Up

Terminal

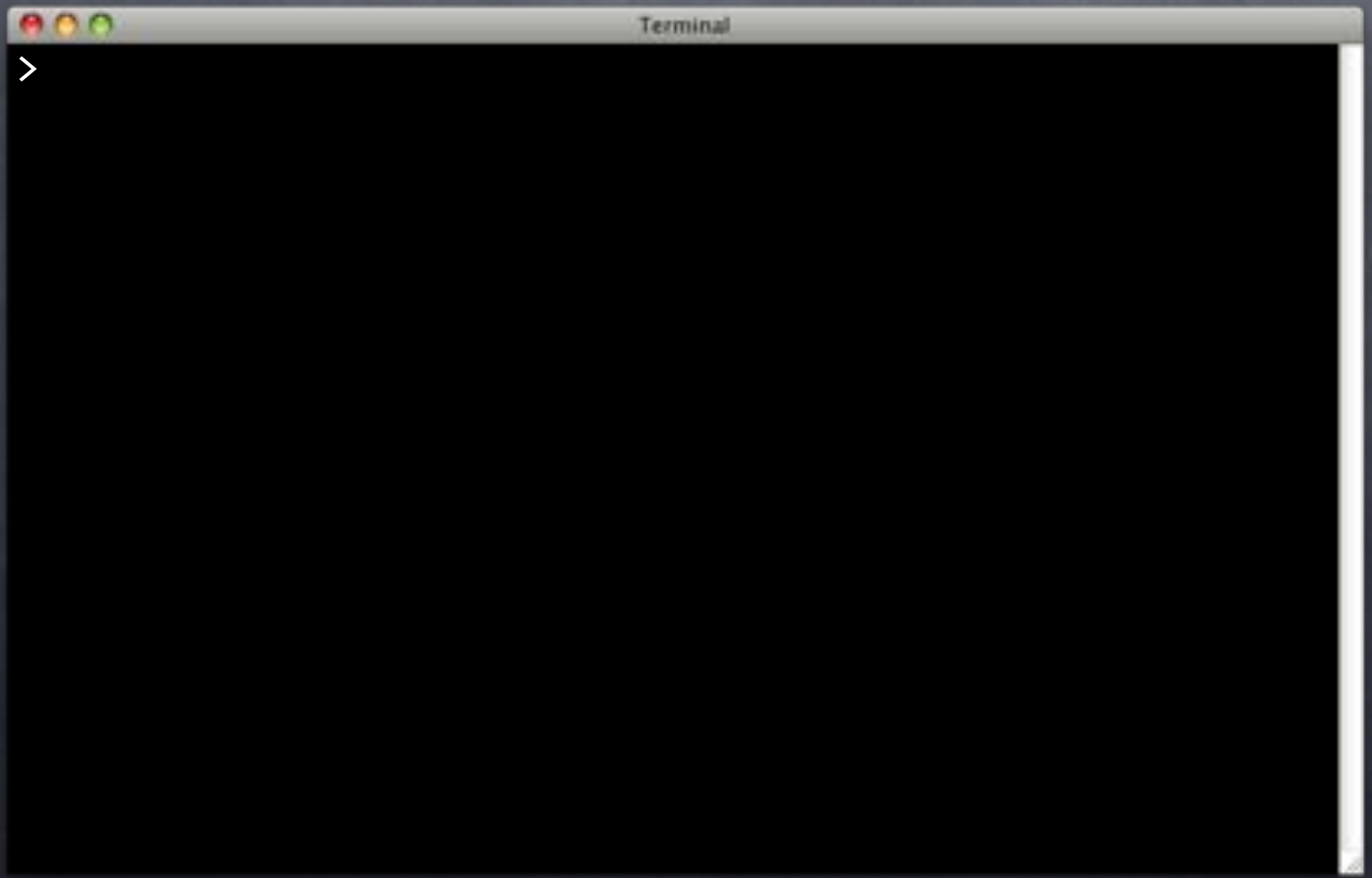
```
> sqitch bundle --dest-dir flipr-1.0.0-r1
Bundling into flipr-1.0.0-r1
Writing config
Writing plan
Writing scripts
  + appschema
  + users
  + flips
  + insert_user
  + change_pass @v1.0.0-r1
>
```

Bundle Up

Terminal

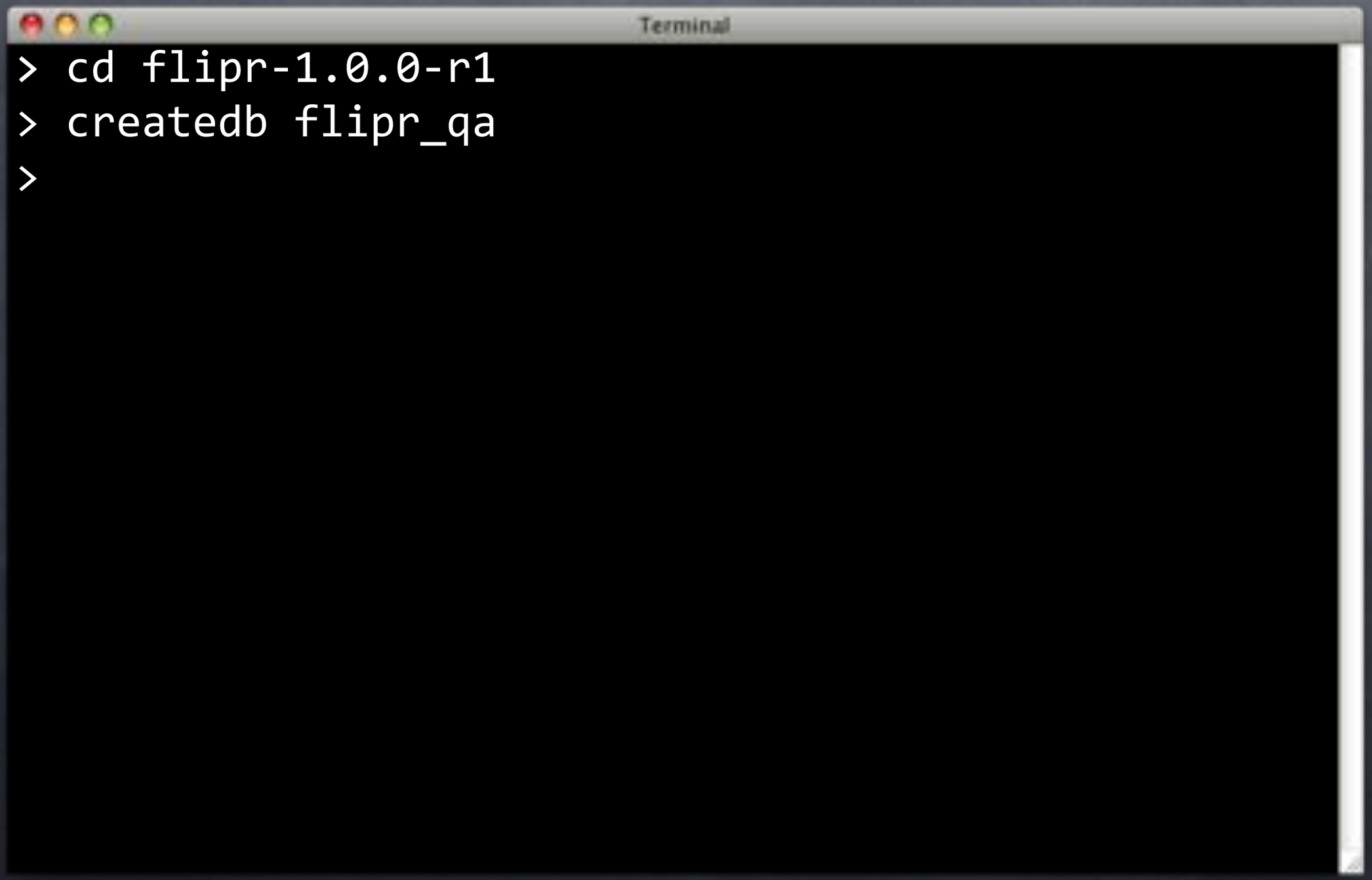
```
> sqitch bundle --dest-dir flipr-1.0.0-r1
Bundling into flipr-1.0.0-r1
Writing config
Writing plan
Writing scripts
  + appschema
  + users
  + flips
  + insert_user
  + change_pass @v1.0.0-r1
>
```

Bundled?



Bundled?

Bundled?



Bundled?

Terminal

```
> cd flipr-1.0.0-r1
> createdb flipr_qa
> sqitch -d flipr_qa deploy
Adding metadata tables to flipr_qa
Deploying changes to flipr_qa
  + appschema ................ ok
  + users ..................... ok
  + flips ..................... ok
  + insert_user @v1.0.0-r1 .. ok
>
```

Bundled?

Terminal

```
> cd flipr-1.0.0-r1
> createdb flipr_qa
> sqitch -d flipr_qa deploy
Adding metadata tables to flipr_qa
Deploying changes to flipr_qa
  + appschema ................ ok
  + users ..................... ok
  + flips ..................... ok
  + insert_user @v1.0.0-r1 .. ok
>
```

Ship it!

Merge Madness

Merge Madness

- Merge everything

Merge Madness

- Merge everything
- Back to master

Merge Madness

- Merge everything
- Back to master
  - users

Merge Madness

- Merge everything
- Back to master
  - users
  - flips

Merge Madness

- Merge everything
- Back to master
  - users
  - flips
  - userfuncs

Merge Madness

- Merge everything
- Back to master
  - users
  - flips
  - userfuncs
- Union merge sqitch.plan

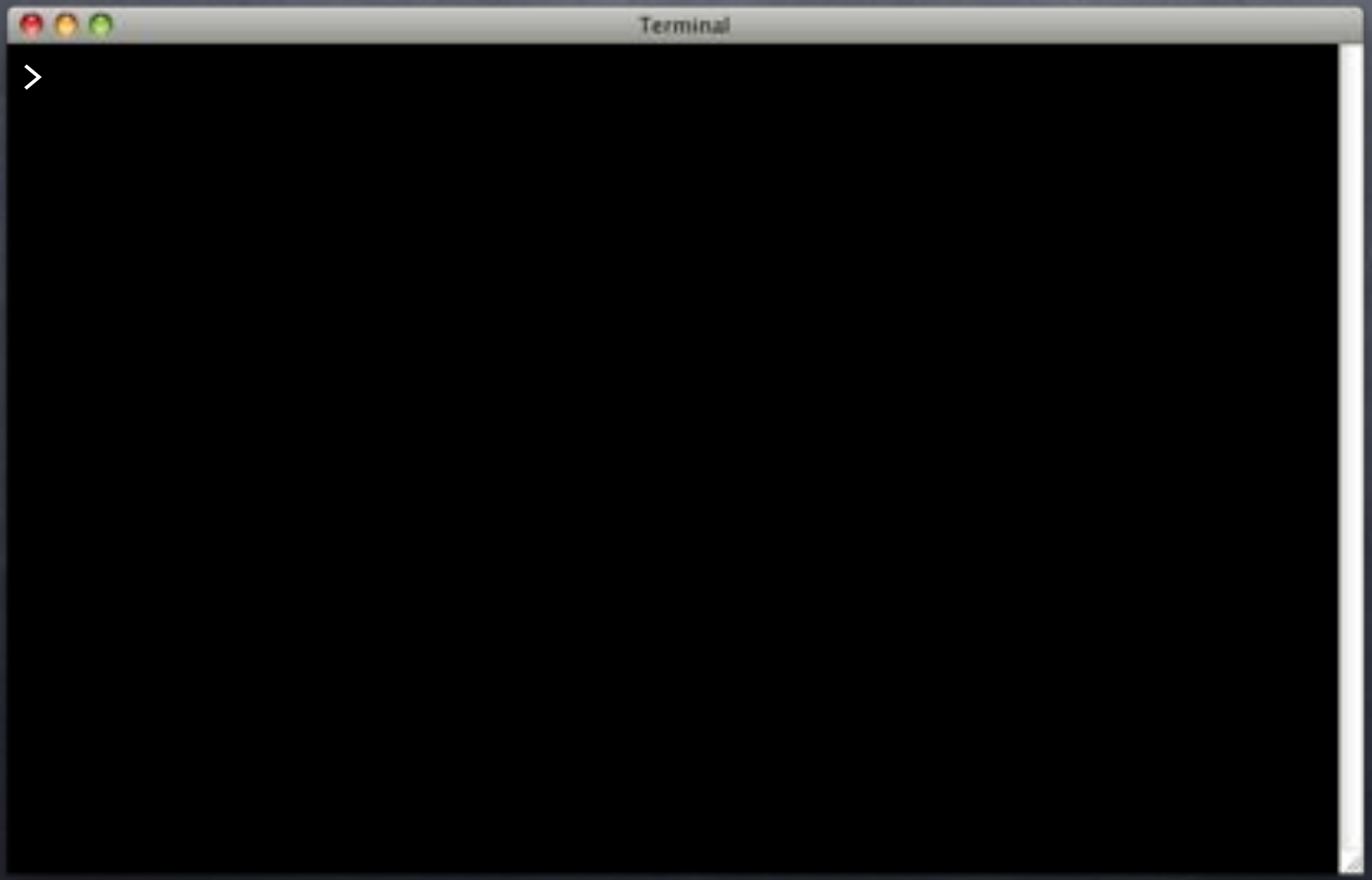
Merge Madness

- Merge everything
- Back to master
  - users
  - flips
  - userfuncs
- Union merge sqitch.plan
- Bundle and ship

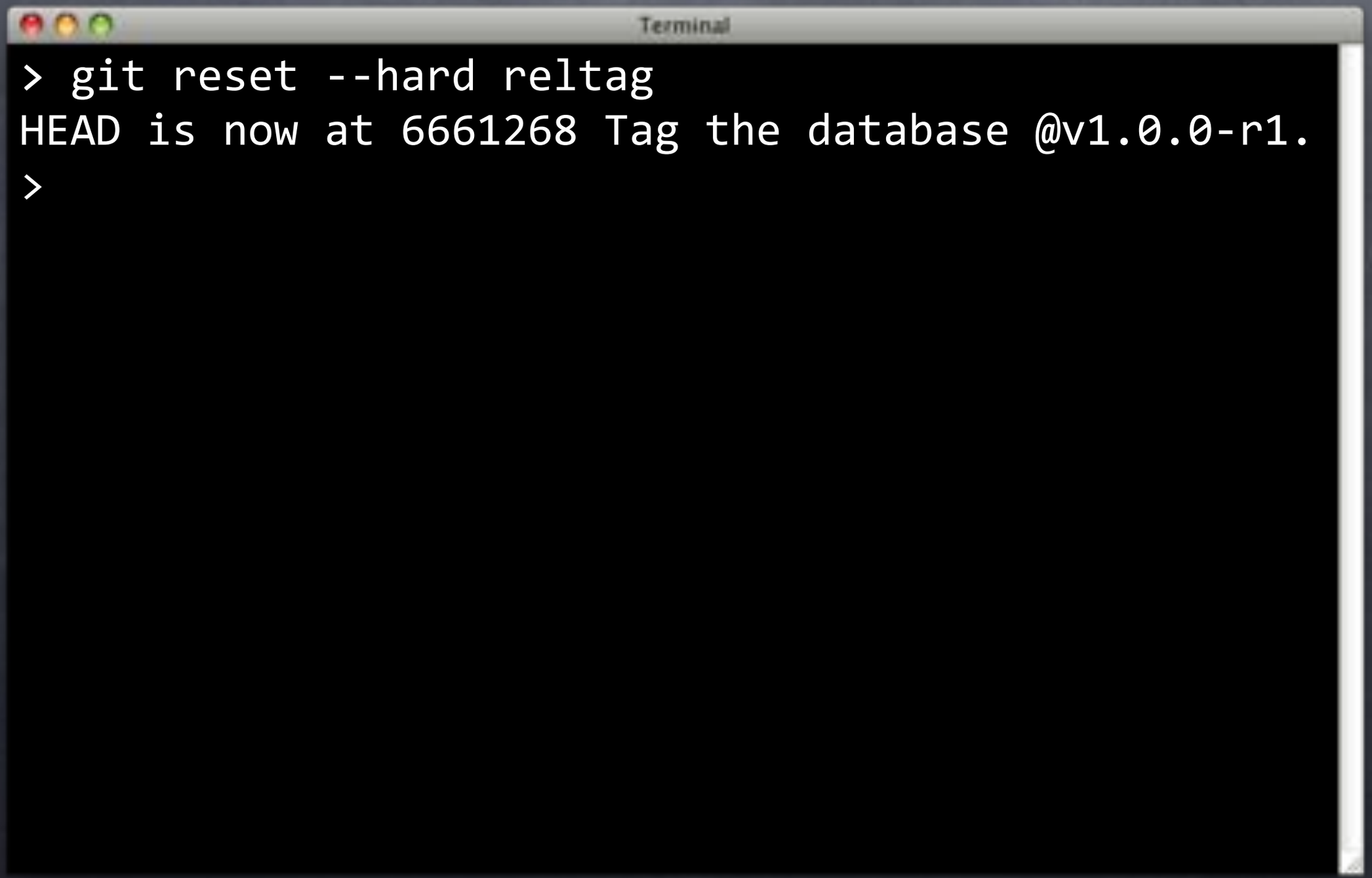
Merge Madness

- Merge everything
- Back to master
  - users
  - flips
  - userfuncs
- Union merge sqitch.plan
- Bundle and ship
- https://github.com/theory/agile-flipr.git

Ruh-Roh



Ruh-Roh


Terminal
> git reset --hard reltag
HEAD is now at 6661268 Tag the database @v1.0.0-r1.
>

Ruh-Roh

Terminal

```
> git reset --hard reltag
HEAD is now at 6661268 Tag the databas
> psql -d flipr_test -c "
    SELECT flipr.insert_user('foo', 'secr3t'),
           flipr.insert_user('bar', 'secr3t');
    SELECT nickname, password FROM flipr.users;
"
 nickname |             password
----------+----------------------------------
 foo      | 9695da4dd567a19f9b92065f240c6725
 bar      | 9695da4dd567a19f9b92065f240c6725
(2 rows)
>
```

Same password

Ruh-Roh

Terminal

```
> git reset --hard reltag
HEAD is now at 6661268 Tag the database @v1.0.0-r1.
> psql -d flipr_test -c "
    SELECT flipr.insert_user('foo', 'secr3t'),
           flipr.insert_user('bar', 'secr3t');
    SELECT nickname, password FROM flipr.users;
"
 nickname |              password
----------+----------------------------------
 foo      | 9695da4dd567a19f9b92065f240c6725
 bar      | 9695da4dd567a19f9b92065f240c6725
(2 rows)
>
```

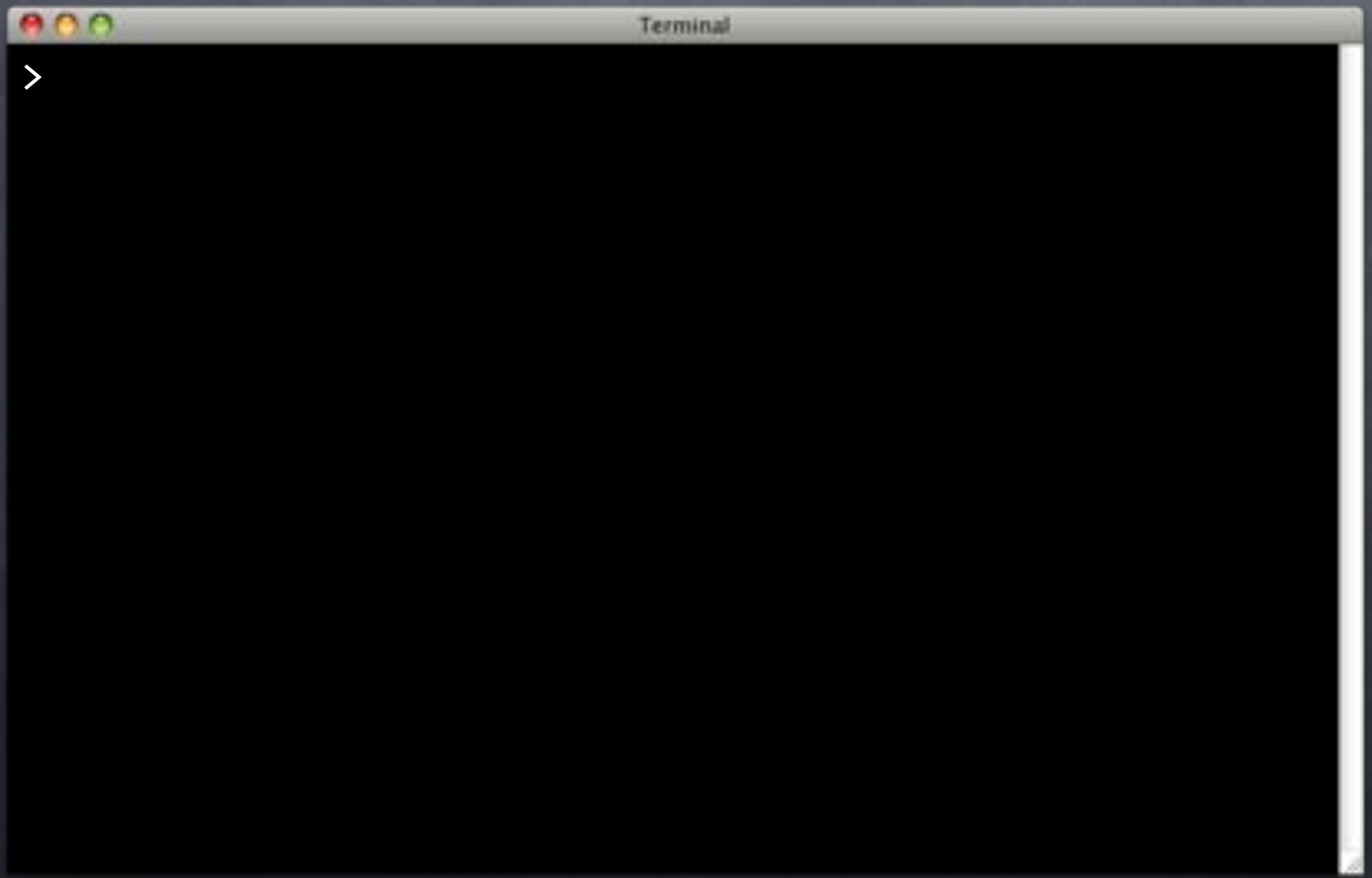
Ruh-Roh

Terminal

```
> git reset --hard reltag
HEAD is now at 6661268 Tag the database @v1.0.0-r1.
> psql -d flipr_test -c "
    SELECT flipr.insert_user('foo', 'secr3t'),
           flipr.insert_user('bar', 'secr3t');
    SELECT nickname, password FROM flipr.users;
"
 nickname |              password
----------+----------------------------------
 foo      | 9695da4dd567a19f9b92065f240c6725
 bar      | 9695da4dd567a19f9b92065f240c6725
(2 rows)
>
```

Not good.

PGCryptonite



PGCryptonite

Terminal

```
> sqitch add pgcrypto -n 'Loads pgcrypto extension.'
Adding deploy/pgcrypto.sql
Adding revert/pgcrypto.sql
Adding verify/pgcrypto.sql
Added "pgcrypto" to sqitch.plan
>
```

PGCryptonite

Terminal

```
> sqitch add pgcrypto -n 'Loads pgcrypto extension.'
Adding deploy/pgcrypto.sql
Adding revert/pgcrypto.sql
Adding verify/pgcrypto.sql
Added "pgcrypto" to sqitch.plan
> emacs deploy/pgcrpyto.sql
>
```

deploy/pgcrypto.sql

Emacs

```
-- Deploy pgcrypto

BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

-:** deploy/pgcrypto All (SQL[ansi])----------

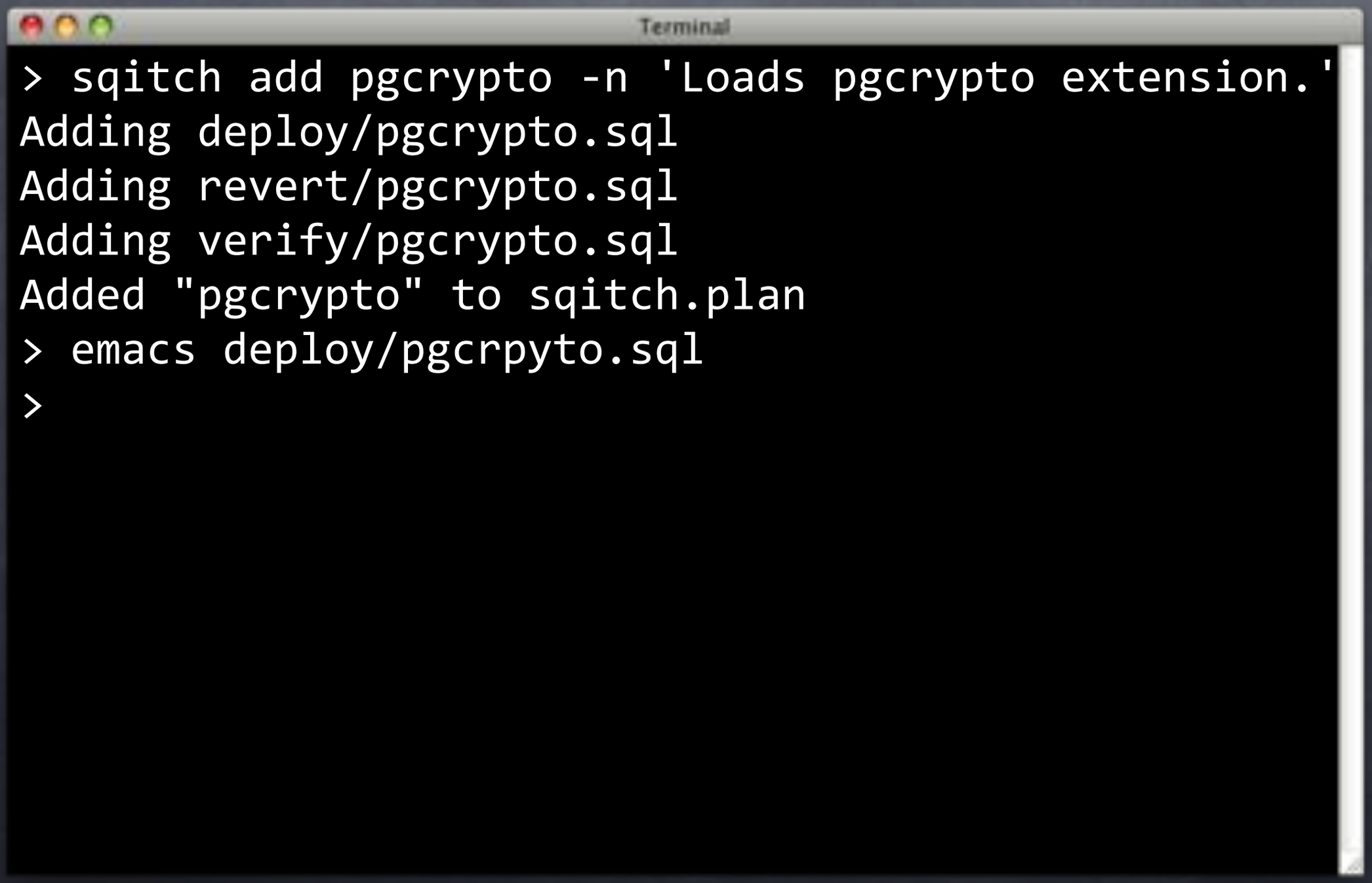
deploy/pgcrypto.sql

Emacs

```
-- Deploy pgcrypto

BEGIN;

CREATE EXTENSION pgcrypto;

COMMIT;
```

-:** deploy/pgcrypto All (SQL[ansi])----------

PGCryptonite



PGCryptonite

Terminal

```
> sqitch add pgcrypto -n 'Loads pgcrypto extension.'
Adding deploy/pgcrypto.sql
Adding revert/pgcrypto.sql
Adding verify/pgcrypto.sql
Added "pgcrypto" to sqitch.plan
> emacs deploy/pgcrpyto.sql
> emacs verify/pgcrpyto.sql
>
```

verify/pgcrypto.sql

Emacs

```
-- Verify pgcrypto

BEGIN;

-- XXX Add verifications here.

COMMIT;
```

-:** verify/pgcrypto All (SQL[ansi])----------

verify/pgcrypto.sql

Emacs

```
-- Verify pgcrypto

BEGIN;

SELECT 1/count(*) FROM pg_extension WHERE extname = 'pgcrypto';
SELECT has_function_privilege('crypt(text, text)', 'execute');
SELECT has_function_privilege('gen_salt(text)', 'execute');

COMMIT;
```

-:** verify/pgcrypto All (SQL[ansi])

verify/pgcrypto.sql

Emacs

```
-- Verify pgcrypto

BEGIN;

SELECT 1/count(*) FROM pg_extension WHERE extname = 'pgcrypto';
SELECT has_function_privilege('crypt(text, text)', 'execute');
SELECT has_function_privilege('gen_salt(text)', 'execute');

COMMIT;
```

-:**- verify/pgcrypto All (SQL[ansi])----------

verify/pgcrypto.sql

Emacs

```
-- Verify pgcrypto

BEGIN;

SELECT 1/count(*) FROM pg_extension WHERE extname = 'pgcrypto';
SELECT has_function_privilege('crypt(text, text)', 'execute');
SELECT has_function_privilege('gen_salt(text)', 'execute');

COMMIT;
```

-:-- verify/pgcrypto   All   (SQL[ansi])

You Know the Drill

You Know the Drill

- Write revert script

You Know the Drill

- Write revert script
- Add test

You Know the Drill

- Write revert script
- Add test
- Use has_function()

You Know the Drill

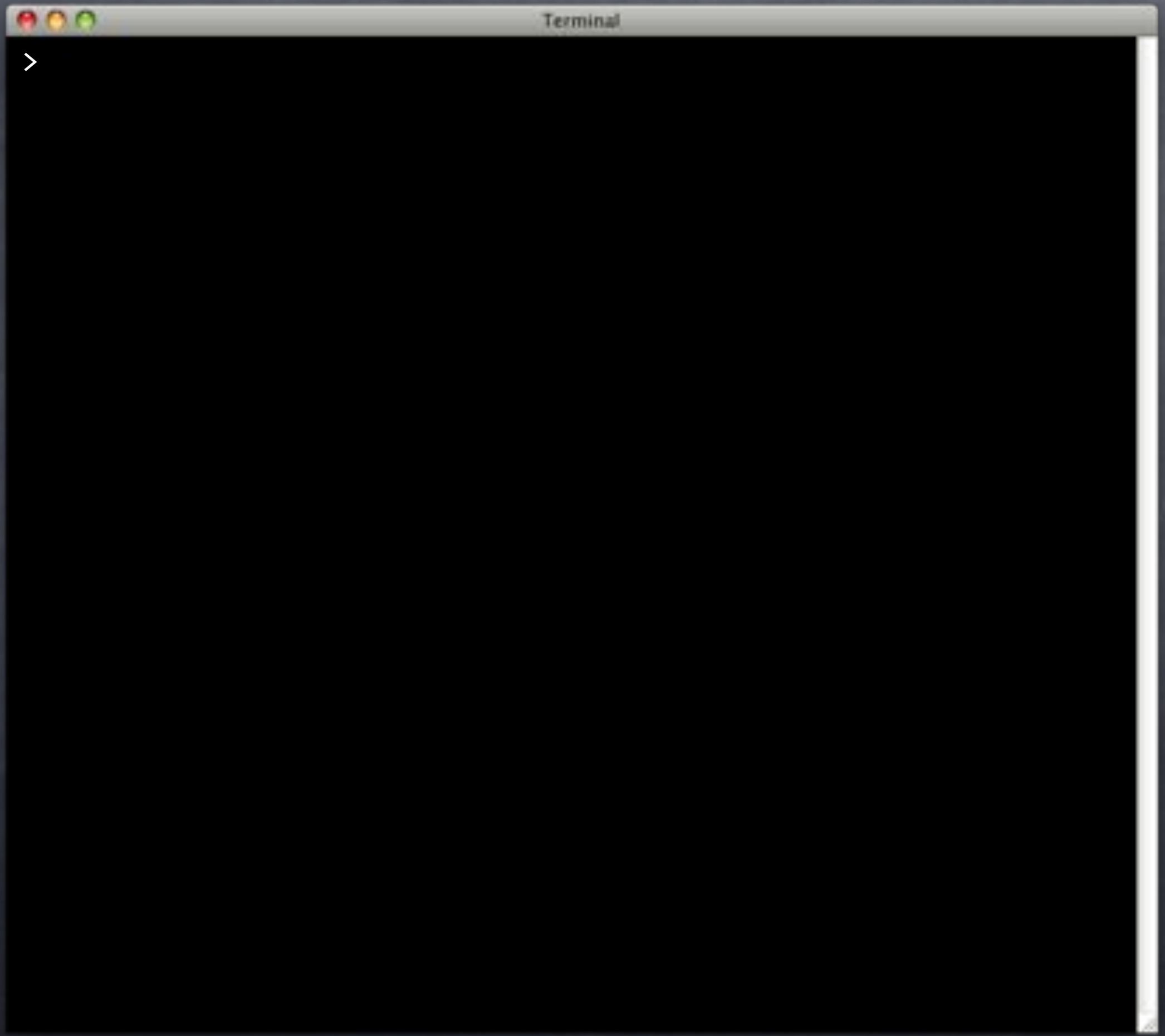
- Write revert script
- Add test
  - Use has_function()
- Commit

You Know the Drill

- Write revert script
- Add test
  - Use has_function()
- Commit
- Push

You Know the Drill

- Write revert script
- Add test
  - Use has_function()
- Commit
- Push
- Modify the insert_user test


Terminal
>

Terminal

```
> git diff test/insert_user.sql
@@ -5,7 +5,7 @@ SET search_path TO flipr,public;

 -- Plan the tests.
 BEGIN;
-SELECT plan(11);
+SELECT plan(12);

 SELECT has_function( 'insert_user' );
 SELECT has_function(
@@ -29,25 +29,25 @@ SELECT lives_ok(
     'Insert a user'
 );

-SELECT row_eq(
-    'SELECT * FROM users',
-    ROW('theory', md5('foo'), NOW())::users,
-    'The user should have been inserted'
-);
+SELECT ok( EXISTS(
+    SELECT 1 FROM flipr.users
+     WHERE nickname = 'theory'
+       AND password = crypt('foo', password)
+), 'The user should have been inserted' );
```

Terminal

```
> git diff test/insert_user.sql
@@ -5,7 +5,7 @@ SET search_path TO flipr,public;

 -- Plan the tests.
 BEGIN;
-SELECT plan(11);
+SELECT plan(12);

 SELECT has_function( 'insert_user' );
 SELECT has_function(
@@ -29,25 +29,25 @@ SELECT lives_ok(
     'Insert a user'
 );

-SELECT row_eq(
-    'SELECT * FROM users',
-    ROW('theory', md5('foo'), NOW())::users,
-    'The user should have been inserted'
-);
+SELECT ok( EXISTS(
+    SELECT 1 FROM flipr.users
+     WHERE nickname = 'theory'
+       AND password = crypt('foo', password)
+), 'The user should have been inserted' );
```

Terminal

```
> git diff test/insert_user.sql
@@ -5,7 +5,7 @@ SET search_path TO flipr,public;

 -- Plan the tests.
 BEGIN;
-SELECT plan(11);
+SELECT plan(12);

 SELECT has_function( 'insert_user' );
 SELECT has_function(
@@ -29,25 +29,25 @@ SELECT lives_ok(
     'Insert a user'
 );

-SELECT row_eq(
-    'SELECT * FROM users',
-    ROW('theory', md5('foo'), NOW())::users,
-    'The user should have been inserted'
-);
+SELECT ok( EXISTS(
+    SELECT 1 FROM flipr.users
+     WHERE nickname = 'theory'
+       AND password = crypt('foo', password)
+), 'The user should have been inserted' );
```

Terminal

```
 SELECT lives_ok(
     $$ SELECT insert_user('strongrrl', 'w00t') $$,
     'Insert another user'
 );

-SELECT bag_eq(
-    'SELECT * FROM users',
-    $$ VALUES
-        ('theory',    md5('foo'),  NOW()),
-        ('strongrrl', md5('w00t'), NOW())
-    $$,
-    'Both users should be present'
-);
+SELECT is(COUNT(*)::INT, 2, 'There should be two users')
+  FROM flipr.users;
+
+SELECT ok( EXISTS(
+    SELECT 1 FROM flipr.users
+     WHERE nickname = 'strongrrl'
+       AND password = crypt('w00t', password)
+), 'The second user should have been inserted' );

 SELECT throws_ok(
     $$ SELECT insert_user('theory', 'ha-ha') $$,
```

Terminal

```
 SELECT lives_ok(
     $$ SELECT insert_user('strongrrl', 'w00t') $$,
     'Insert another user'
 );

-SELECT bag_eq(
-    'SELECT * FROM users',
-    $$ VALUES
-        ('theory',    md5('foo'),  NOW()),
-        ('strongrrl', md5('w00t'), NOW())
-    $$,
-    'Both users should be present'
-);
+SELECT is(COUNT(*)::INT, 2, 'There should be two users')
+  FROM flipr.users;
+
+SELECT ok( EXISTS(
+    SELECT 1 FROM flipr.users
+     WHERE nickname = 'strongrrl'
+       AND password = crypt('w00t', password)
+), 'The second user should have been inserted' );

 SELECT throws_ok(
     $$ SELECT insert_user('theory', 'ha-ha') $$,
```

Terminal

```
 SELECT lives_ok(
     $$ SELECT insert_user('strongrrl', 'w00t') $$,
     'Insert another user'
 );

-SELECT bag_eq(
-    'SELECT * FROM users',
-    $$ VALUES
-        ('theory',    md5('foo'),  NOW()),
-        ('strongrrl', md5('w00t'), NOW())
-    $$,
-    'Both users should be present'
-);
+SELECT is(COUNT(*)::INT, 2, 'There should be two users')
+  FROM flipr.users;
+
+SELECT ok( EXISTS(
+    SELECT 1 FROM flipr.users
+     WHERE nickname = 'strongrrl'
+       AND password = crypt('w00t', password)
+), 'The second user should have been inserted' );

 SELECT throws_ok(
     $$ SELECT insert_user('theory', 'ha-ha') $$,
```

Terminal

```
 SELECT lives_ok(
     $$ SELECT insert_user('strongrrl', 'w00t') $$,
     'Insert another user'
 );

-SELECT bag_eq(
-    'SELECT * FROM users',
-    $$ VALUES
-        ('theory',    md5('foo'),  NOW()),
-        ('strongrrl', md5('w00t'), NOW())
-    $$,
-    'Both users should be present'
-);
+SELECT is(COUNT(*)::INT, 2, 'There should be two users')
+  FROM flipr.users;
+
+SELECT ok( EXISTS(
+    SELECT 1 FROM flipr.users
+     WHERE nickname = 'strongrrl'
+       AND password = crypt('w00t', password)
+), 'The second user should have been inserted' );

 SELECT throws_ok(
     $$ SELECT insert_user('theory', 'ha-ha') $$,
```

Terminal

```
@@ -56,14 +56,8 @@ SELECT throws_ok(
      'Should get an error for duplicate nickname'
  );

-SELECT bag_eq(
-    'SELECT * FROM users',
-    $$ VALUES
-        ('theory',    md5('foo'),  NOW()),
-        ('strongrrl', md5('w00t'), NOW())
-    $$,
-    'Should still have just the two users'
-);
+SELECT is(COUNT(*)::INT, 2, 'Should still have two users')
+  FROM flipr.users;

  SELECT finish();
  ROLLBACK;
>
```

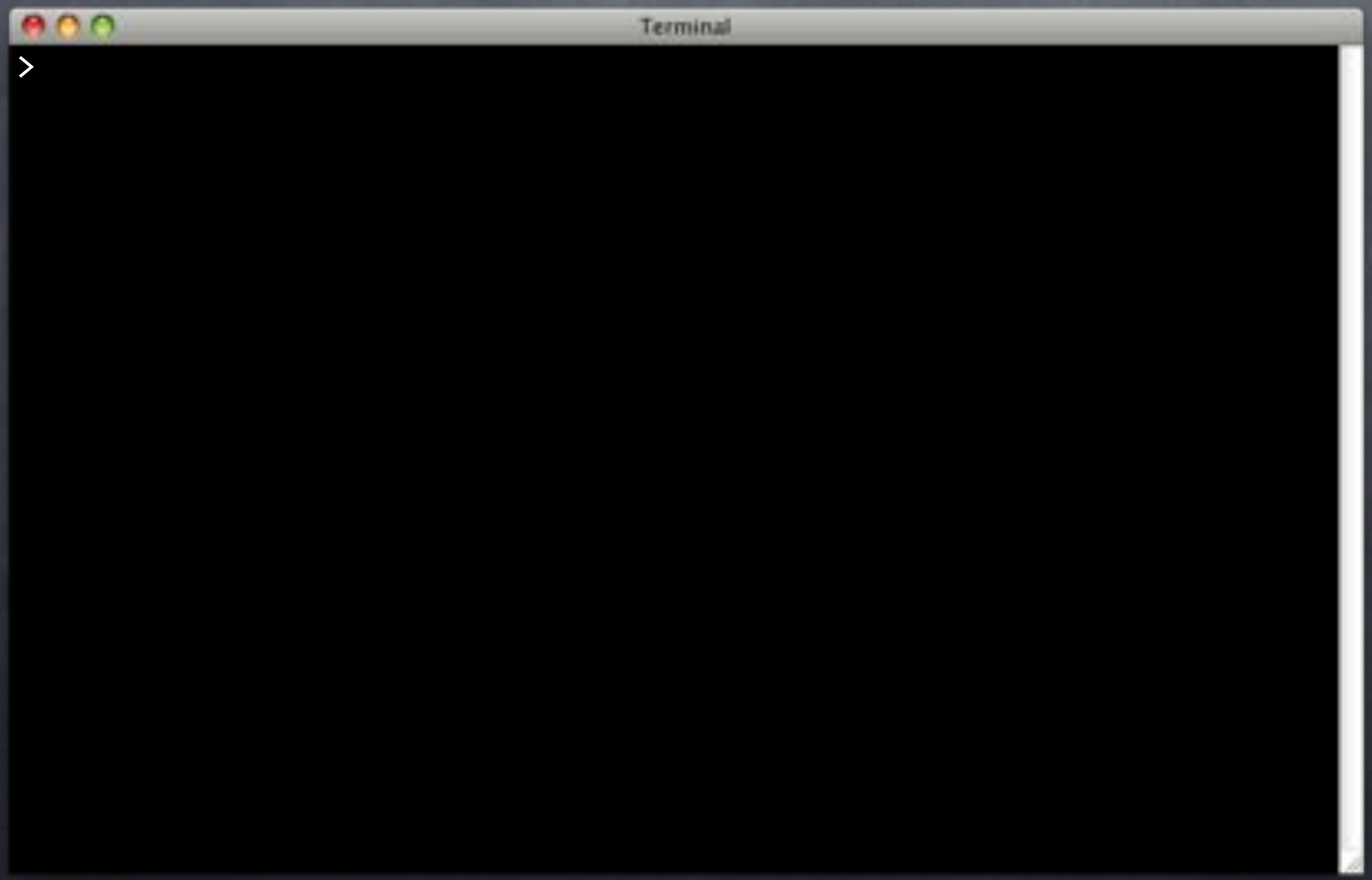
Terminal

```
@@ -56,14 +56,8 @@ SELECT throws_ok(
      'Should get an error for duplicate nickname'
  );

-SELECT bag_eq(
-    'SELECT * FROM users',
-    $$ VALUES
-        ('theory',    md5('foo'),  NOW()),
-        ('strongrrl', md5('w00t'), NOW())
-    $$,
-    'Should still have just the two users'
-);
+SELECT is(COUNT(*)::INT, 2, 'Should still have two users')
+  FROM flipr.users;

  SELECT finish();
  ROLLBACK;
>
```

Terminal

```
@@ -56,14 +56,8 @@ SELECT throws_ok(
      'Should get an error for duplicate nickname'
  );

-SELECT bag_eq(
-    'SELECT * FROM users',
-    $$ VALUES
-        ('theory',    md5('foo'),  NOW()),
-        ('strongrrl', md5('w00t'), NOW())
-    $$,
-    'Should still have just the two users'
-);
+SELECT is(COUNT(*)::INT, 2, 'Should still have two users')
+  FROM flipr.users;

  SELECT finish();
  ROLLBACK;
>
```

FAIL



FAIL

Terminal

```
> pg_prove -d flipr_test test/insert_user.sql
test/insert_user.sql .. 1/12
# Failed test 7: "The user should have been inserted"
# Failed test 10: "The second user should have been inserted"
# Looks like you failed 2 tests of 12
test/insert_user.sql .. Failed 2/12 subtests

Test Summary Report
-------------------
test/insert_user.sql (Wstat: 0 Tests: 12 Failed: 2)
  Failed tests:  7, 10
Files=1, Tests=12,  1 wallclock secs
Result: FAIL
>
```

FAIL

Terminal

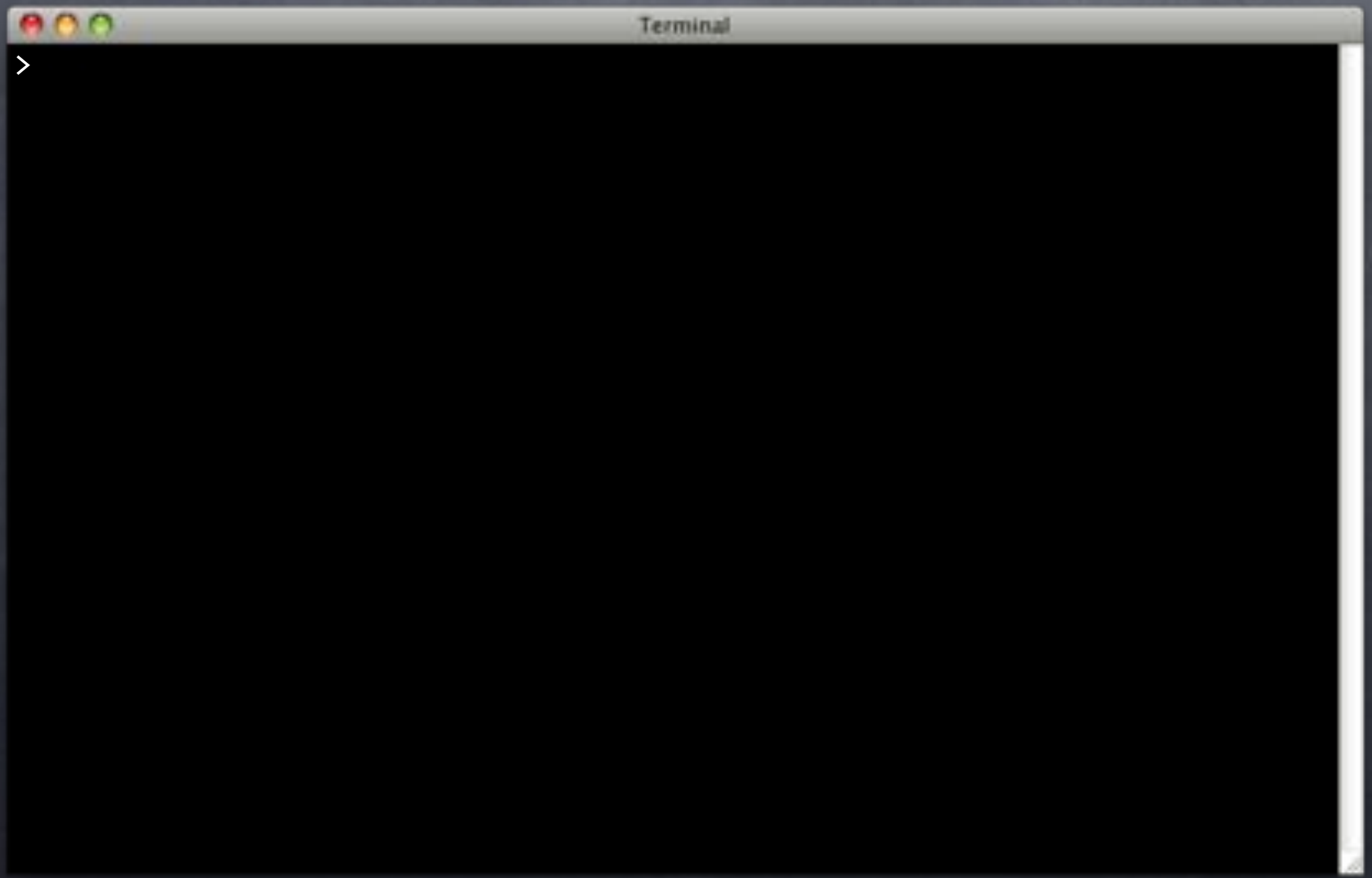
```
> pg_prove -d flipr_test test/insert_user.sql
test/insert_user.sql .. 1/12
# Failed test 7: "The user should have been inserted"
# Failed test 10: "The second user should have been inserted"
# Looks like you failed 2 tests of 12
test/insert_user.sql .. Failed 2/12 subtests

Test Summary Report
-------------------
test/insert_user.sql (Wstat: 0 Tests: 12 Failed: 2)
  Failed tests:  7, 10
Files=1, Tests=12,  1 wallclock secs
Result: FAIL
>
```

As expected.

Back when discussing traditional migration systems, I said...

Managing procedures is a PITA!

Consider this Change



Consider this Change

Terminal

```
> git diff
diff --git a/deploy/insert_user.sql b/deploy/insert_user.sql
index eb30fed..5c28d02 100644
--- a/deploy/insert_user.sql
+++ b/deploy/insert_user.sql
@@ -8,7 +8,7 @@ CREATE OR REPLACE FUNCTION flipr.insert_user(
     nickname TEXT,
     password TEXT
 ) RETURNS VOID LANGUAGE SQL SECURITY DEFINER AS $$
-    INSERT INTO flipr.users VALUES($1, md5($2));
+    INSERT INTO flipr.users values($1, crypt($2, gen_salt('md5')));
 $$;

 COMMIT;
```

Consider this Change

Terminal

```
> git diff
diff --git a/deploy/insert_user.sql b/deploy/insert_user.sql
index eb30fed..5c28d02 100644
--- a/deploy/insert_user.sql
+++ b/deploy/insert_user.sql
@@ -8,7 +8,7 @@ CREATE OR REPLACE FUNCTION flipr.insert_user(
     nickname TEXT,
     password TEXT
 ) RETURNS VOID LANGUAGE SQL SECURITY DEFINER AS $$
-    INSERT INTO flipr.users VALUES($1, md5($2));
+    INSERT INTO flipr.users values($1, crypt($2, gen_salt('md5')));
 $$;

 COMMIT;
```

Simple, right?

Not So Much

Not So Much

- Copy insert_user.sql to new deploy file

Not So Much

- Copy insert_user.sql to new deploy file
- Change that new file

Not So Much

- Copy insert_user.sql to new deploy file
- Change that new file
- Copy insert_user.sql to new revert file

Not So Much

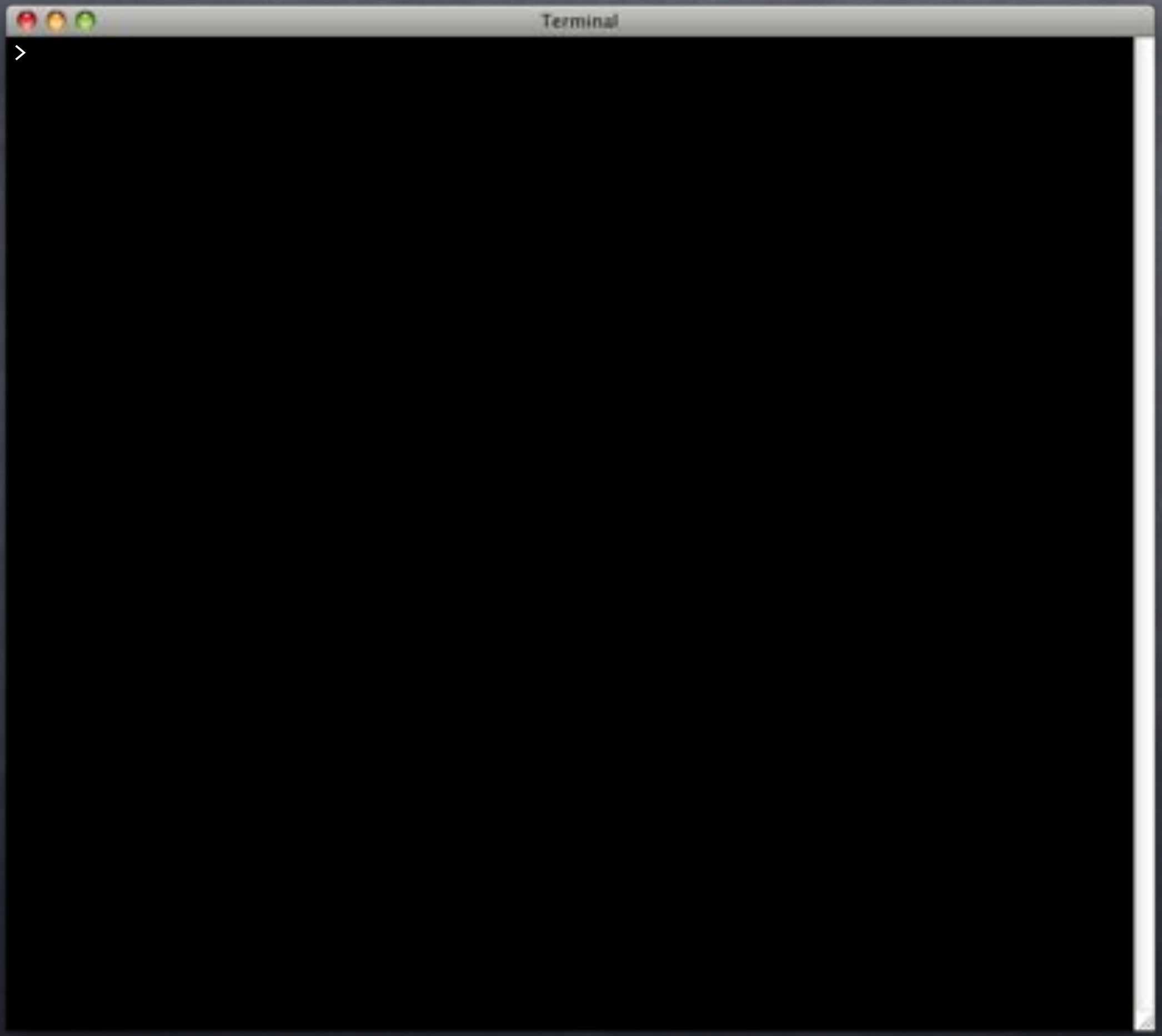
- Copy insert_user.sql to new deploy file
- Change that new file
- Copy insert_user.sql to new revert file
- Test it

Not So Much

- Copy insert_user.sql to new deploy file
- Change that new file
- Copy insert_user.sql to new revert file
- Test it
- Do the same for change_pass.sql

Not So Much

- Copy insert_user.sql to new deploy file
- Change that new file
- Copy insert_user.sql to new revert file
- Test it
- Do the same for change_pass.sql
- The problem with that...


Terminal
>

Terminal

```
> git diff HEAD^
diff --git a/deploy/insert_user_crypt.sql b/deploy/insert_user_crypto.sql
new file mode 100644
index 0000000..fa8d0c6
--- /dev/null
+++ b/deploy/insert_user_crypt.sql
@@ -0,0 +1,8 @@
+-- requires: users, appuser, pgcrypto
+
+CREATE OR REPLACE FUNCTION insert_user(
+    nickname TEXT,
+    password TEXT
+) RETURNS VOID LANGUAGE SQL AS $$
+    INSERT INTO users values($1, crypt($2, gen_salt('md5')));
+$$;
diff --git a/revert/insert_user_crypt.sql b/revert/insert_user_crypto.sql
new file mode 100644
index 0000000..a7f4e31
--- /dev/null
+++ b/revert/insert_user_crypt.sql
@@ -0,0 +1,8 @@
+-- requires: users, appuser
+
+CREATE OR REPLACE FUNCTION insert_user(
+    nickname TEXT,
+    password TEXT
+) RETURNS VOID LANGUAGE SQL AS $$
+    INSERT INTO users values($1, md5($2));
+$$;
```

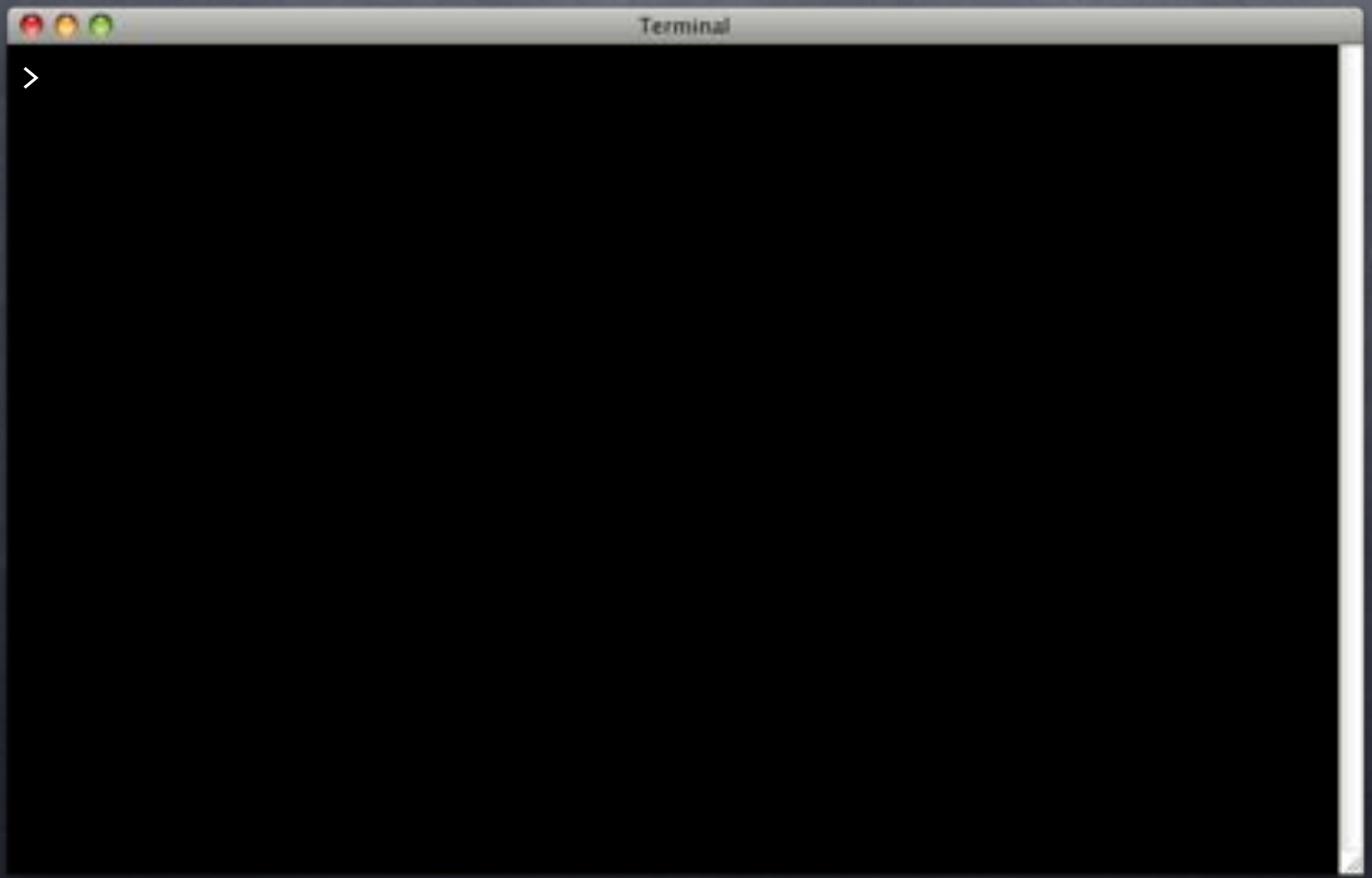
Terminal

```
> git diff HEAD^
diff --git a/deploy/insert_user_crypt.sql b/deploy/insert_user_crypto.sql
new file mode 100644
index 0000000..fa8d0c6
--- /dev/null
+++ b/deploy/insert_user_crypt.sql
@@ -0,0 +1,8 @@
+-- requires: users, appuser, pgcrypto
+
+CREATE OR REPLACE FUNCTION insert_user(
+    nickname TEXT,
+    password TEXT
+) RETURNS VOID LANGUAGE SQL AS $$
+    INSERT INTO users values($1, crypt($2, gen_salt('md5')));
+$$;
diff --git a/revert/insert_user_crypt.sql b/revert/insert_user_crypto.sql
new file mode 100644
index 0000000..a7f4e31
--- /dev/null
+++ b/revert/insert_user_crypt.sql
@@ -0,0 +1,8 @@
+-- requires: users, appuser
+
+CREATE OR REPLACE FUNCTION insert_user(
+    nickname TEXT,
+    password TEXT
+) RETURNS VOID LANGUAGE SQL AS $$
+    INSERT INTO users values($1, md5($2));
+$$;
```

Oy.

Let Sqitch do the work.

Rework It



Rework It

Terminal

```
> sqitch rework insert_user -r pgcrypto \
  -n 'Changes insert_user to use pgcrypto.'
Added "insert_user [insert_user@v1.0.0-r1 pgcrypto]" to
sqitch.plan.
Modify these files as appropriate:
  * deploy/insert_user.sql
  * revert/insert_user.sql
  * verify/insert_user.sql
>
```

Rework It

Terminal

```
> sqitch rework insert_user -r pgcrypto \
  -n 'Changes insert_user to use pgcrypto.'
Added "insert_user [insert_user@v1.0.0-r1 pgcrypto]" to
sqitch.plan.
Modify these files as appropriate:
  * deploy/insert_user.sql
  * revert/insert_user.sql
  * verify/insert_user.sql
>
```

Rework It

Terminal

```
> sqitch rework insert_user -r pgcrypto \
  -n 'Changes insert_user to use pgcrypto.'
Added "insert_user [insert_user@v1.0.0-r1 pgcrypto]" to
sqitch.plan.
Modify these files as appropriate:
  * deploy/insert_user.sql
  * revert/insert_user.sql
  * verify/insert_user.sql
>
```

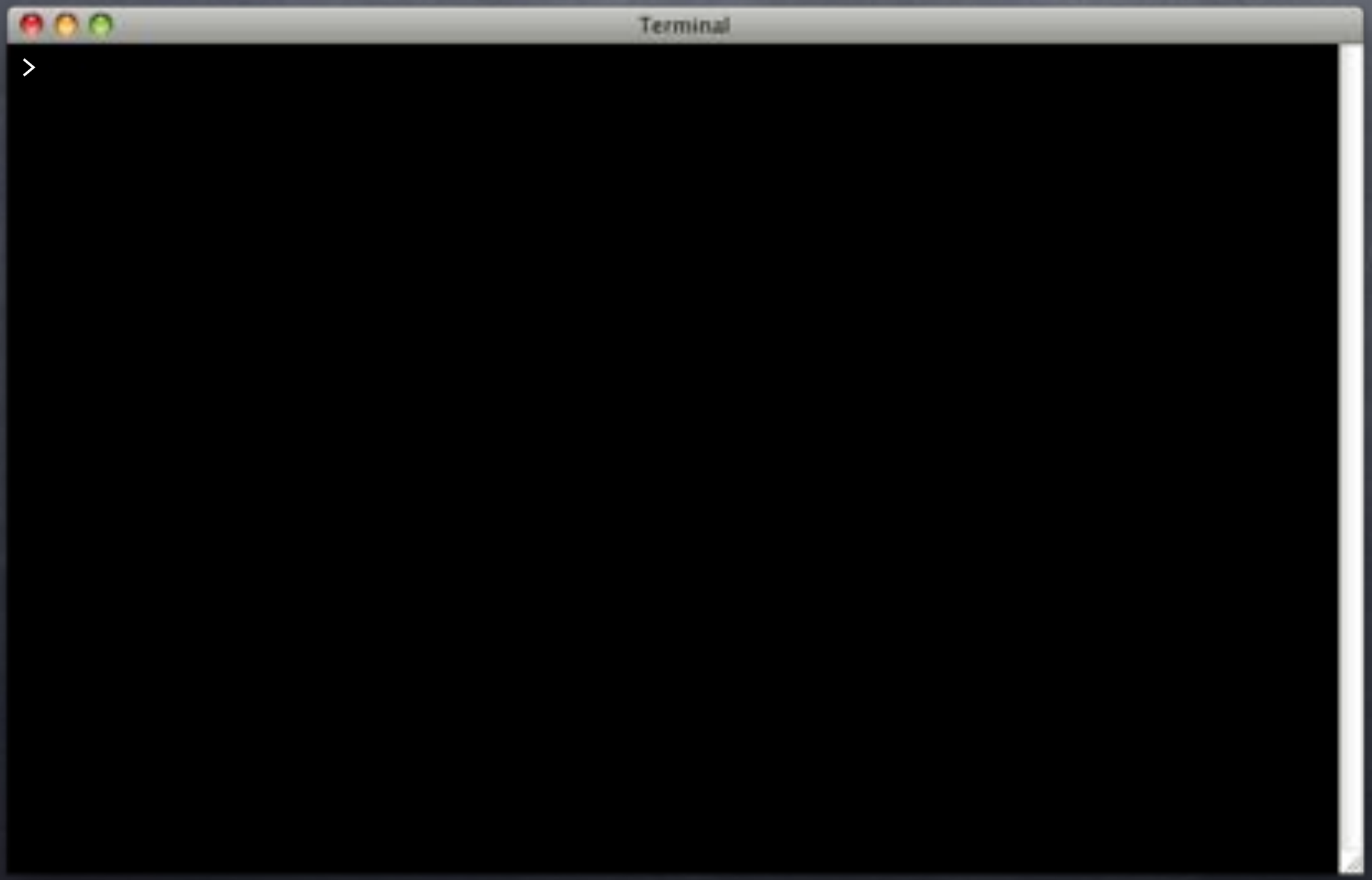
Rework It

Terminal

```
> sqitch rework insert_user -r pgcrypto \
  -n 'Changes insert_user to use pgcrypto.'
Added "insert_user [insert_user@v1.0.0-r1 pgcrypto]" to
sqitch.plan.
Modify these files as appropriate:
  * deploy/insert_user.sql
  * revert/insert_user.sql
  * verify/insert_user.sql
>
```

Same files?

Same Files?



Same Files?

Terminal

```
> git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committe
#   (use "git checkout -- <file>..." to discard changes in w
#
#   modified:   revert/insert_user.sql
#   modified:   sqitch.plan
#   modified:   test/insert_user.sql
#
# Untracked files:
#   (use "git add <file>..." to include in what will be comm
#
#   deploy/insert_user@v1.0.0-r1.sql
#   revert/insert_user@v1.0.0-r1.sql
#   verify/insert_user@v1.0.0-r1.sql
no changes added to commit (use "git add" and/or "git commit
>
```

Same Files?

Terminal

```
> git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committe
#   (use "git checkout -- <file>..." to discard changes in w
#
#   modified:   revert/insert_user.sql
#   modified:   sqitch.plan
#   modified:   test/insert_user.sql
#
# Untracked files:
#   (use "git add <file>..." to include in what will be comm
#
#   deploy/insert_user@v1.0.0-r1.sql
#   revert/insert_user@v1.0.0-r1.sql
#   verify/insert_user@v1.0.0-r1.sql
no changes added to commit (use "git add" and/or "git commit
>
```

Same Files?

Terminal

```
> git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committe
#   (use "git checkout -- <file>..." to discard changes in w
#
#   modified:   revert/insert_user.sql
#   modified:   sqitch.plan
#   modified:   test/insert_user.sql
#
# Untracked files:
#   (use "git add <file>..." to include in what will be comm
#
#   deploy/insert_user@v1.0.0-r1.sql
#   revert/insert_user@v1.0.0-r1.sql
#   verify/insert_user@v1.0.0-r1.sql
no changes added to commit (use "git add" an
>
```

As of @v1.0.0-r1

Same Files?

Terminal

```
> git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committe
#   (use "git checkout -- <file>..." to discard changes in w
#
#   modified:   revert/insert_user.sql
#   modified:   sqitch.plan
#   modified:   test/insert_user.sql
#
# Untracked files:
#   (use "git add <file>..." to include in what will be comm
#
#   deploy/insert_user@v1.0.0-r1.sql
#   revert/insert_user@v1.0.0-r1.sql
#   verify/insert_user@v1.0.0-r1.sql
no changes added to commit (use "git add" and/or "git commit
>
```

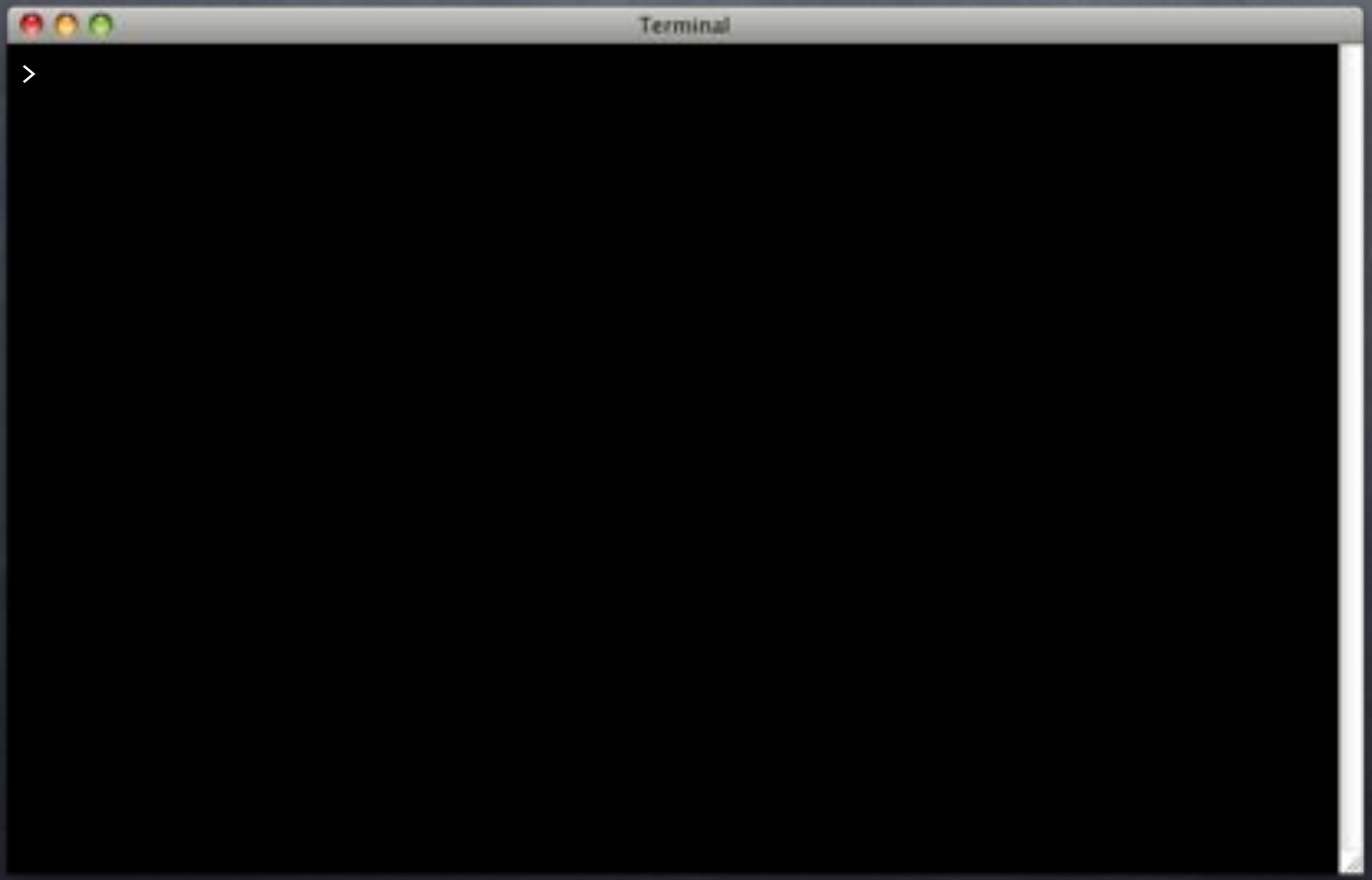
Same Files?

Terminal

```
> git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committe
#   (use "git checkout -- <file>..." to discard changes in w
#
#   modified:   revert/insert_user.sql
#   modified:   sqitch.plan
#   modified:   test/insert_user.sql
#
# Untracked files:
#   (use "git add <file>..." to include in what will be comm
#
#   deploy/insert_user@v1.0.0-r1.sql
#   revert/insert_user@v1.0.0-r1.sql
#   verify/insert_user@v1.0.0-r1.sql
no changes added to commit (use "git add" and/or "git commit
>
```

Previous deploy becomes revert

What's the Diff?

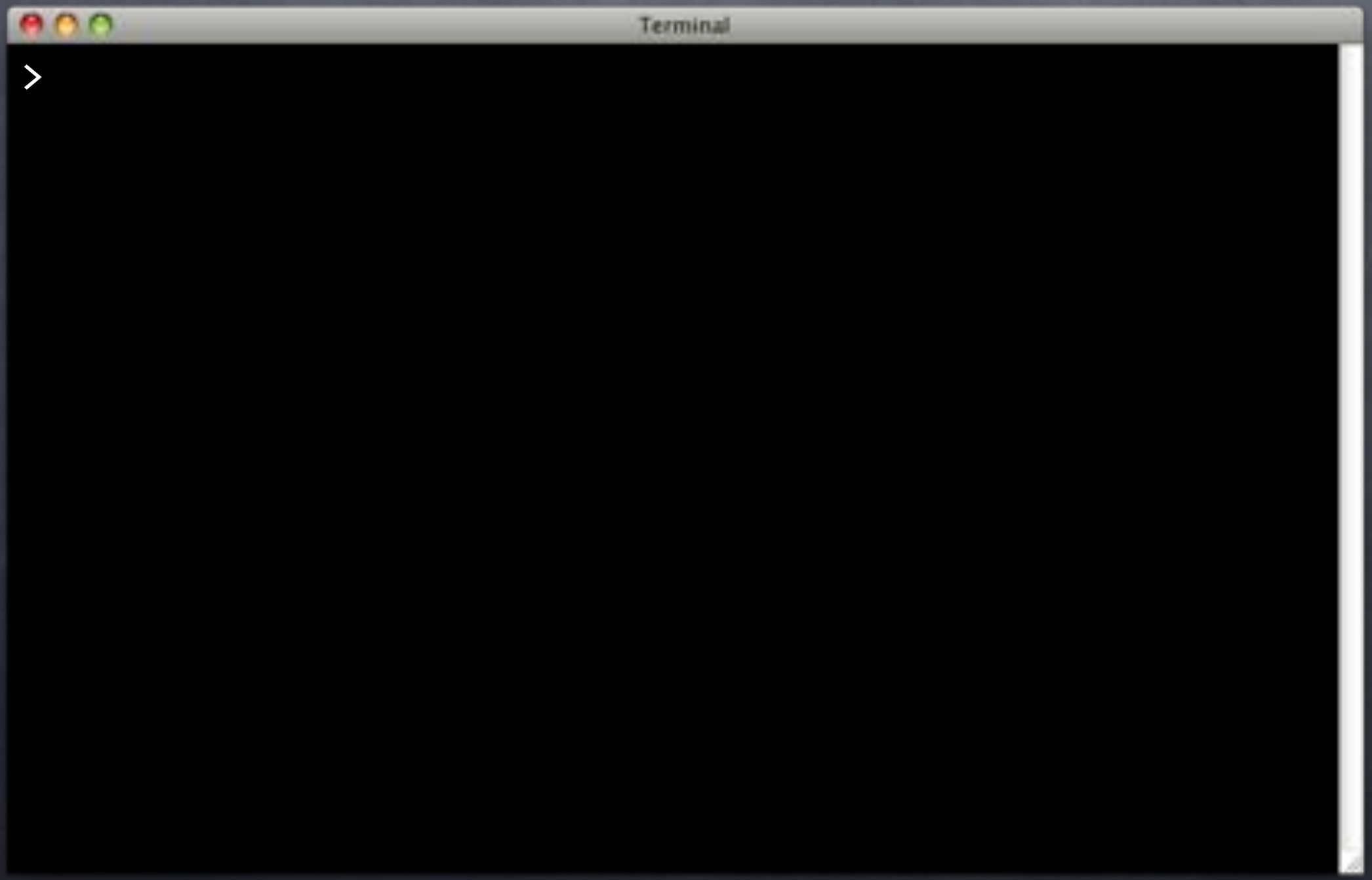


What's the Diff?

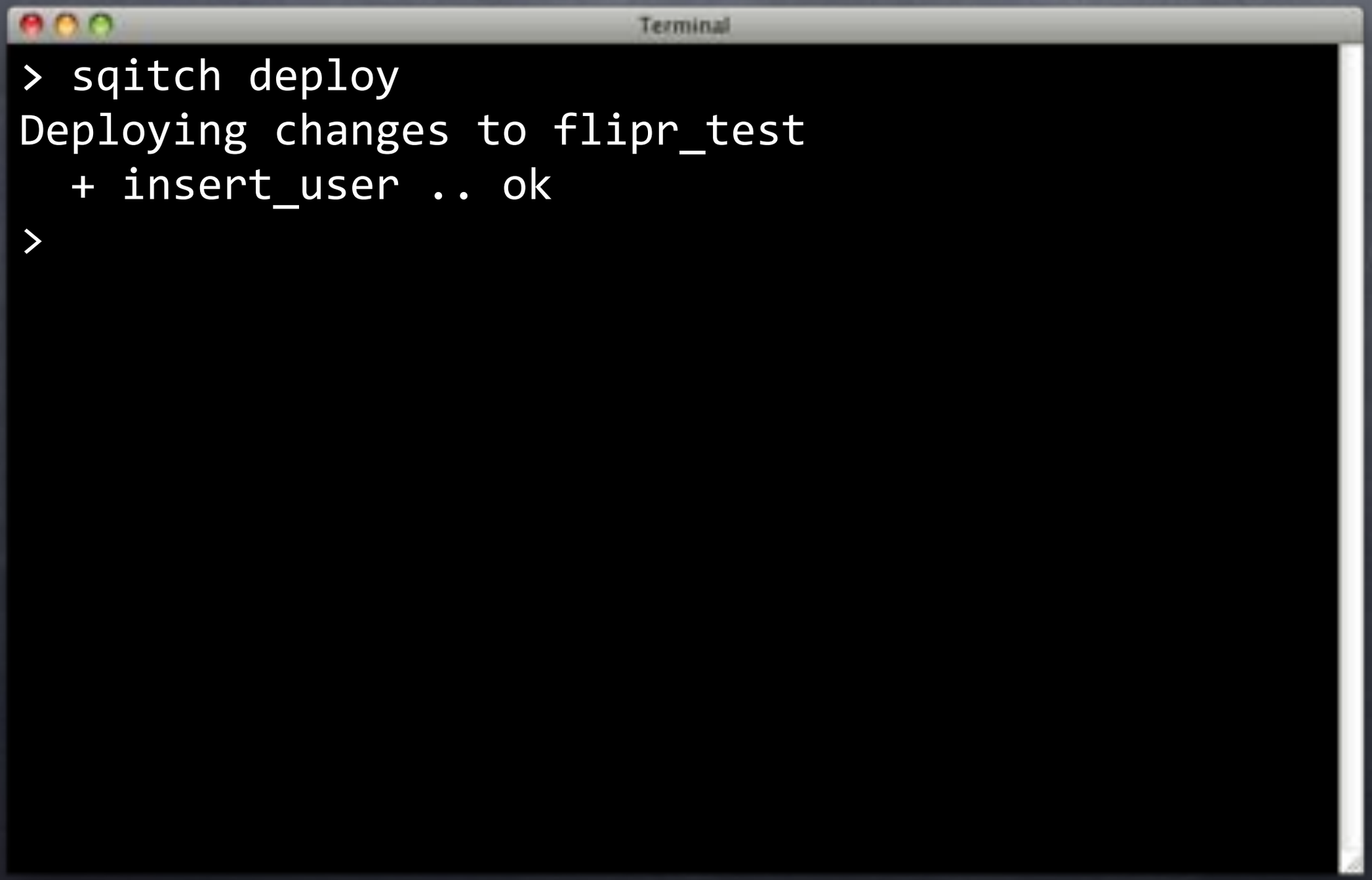
Terminal

```
> diff -u deploy/insert_user.sql
diff --git a/deploy/insert_user.sql b/deploy/insert_user.sql
@@ -1,6 +1,7 @@
 -- Deploy insert_user
 -- requires: users
 -- requires: appschema
+-- requires: appschema

 BEGIN;

@@ -8,7 +9,7 @@ CREATE OR REPLACE FUNCTION flipr.insert_user(
     nickname TEXT,
     password TEXT
 ) RETURNS VOID LANGUAGE SQL SECURITY DEFINER AS $$
-    INSERT INTO flipr.users VALUES($1, md5($2));
+    INSERT INTO flipr.users values($1, crypt($2, gen_salt('md5')));
 $$;

 COMMIT;
>
```

Send it Up!



Send it Up!


Terminal
> sqitch deploy
Deploying changes to flipr_test
  + insert_user .. ok
>

Send it Up!

Terminal

```
> sqitch deploy
Deploying changes to flipr_test
  + insert_user .. ok
> psql -d flipr_test -c "
    DELETE FROM flipr.users;
    SELECT flipr.insert_user('foo', 'secr3t'),
           flipr.insert_user('bar', 'secr3t');
    SELECT nickname, password FROM flipr.users;
"
 nickname |                 password
----------+------------------------------------
 foo      | $1$nKO47p03$YRXYTt4NoNncTThLyxzEq1
 bar      | $1$LbVUs/p.$LVbvPlkD8rJlixW2nS3WP0
(2 rows)
>
```

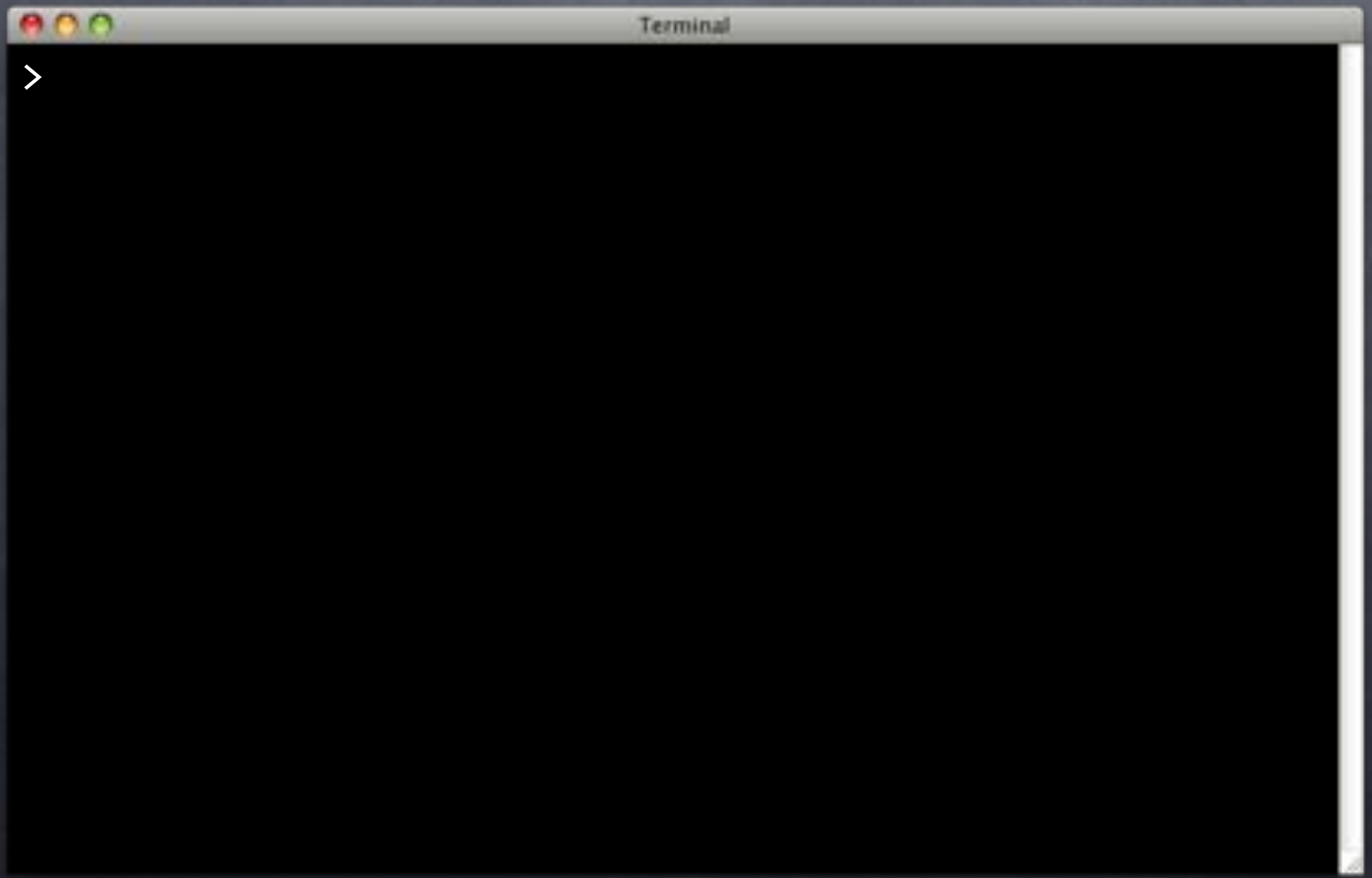
Send it Up!

Terminal

```
> sqitch deploy
Deploying changes to flipr_test
  + insert_user .. ok
> psql -d flipr_test -c "
    DELETE FROM flipr.users;
    SELECT flipr.insert_user('foo', 'secr3t'),
           flipr.insert_user('bar', 'secr3t');
    SELECT nickname, password FROM flipr.users;
"
 nickname |                 password
----------+-------------------------------------
 foo      | $1$nKO47p03$YRXYTt4NoNncTThLyxzEq1
 bar      | $1$LbVUs/p.$LVbvPlkD8rJlixW2nS3WP0
(2 rows)
>
```

\o/

Can We Go Back?



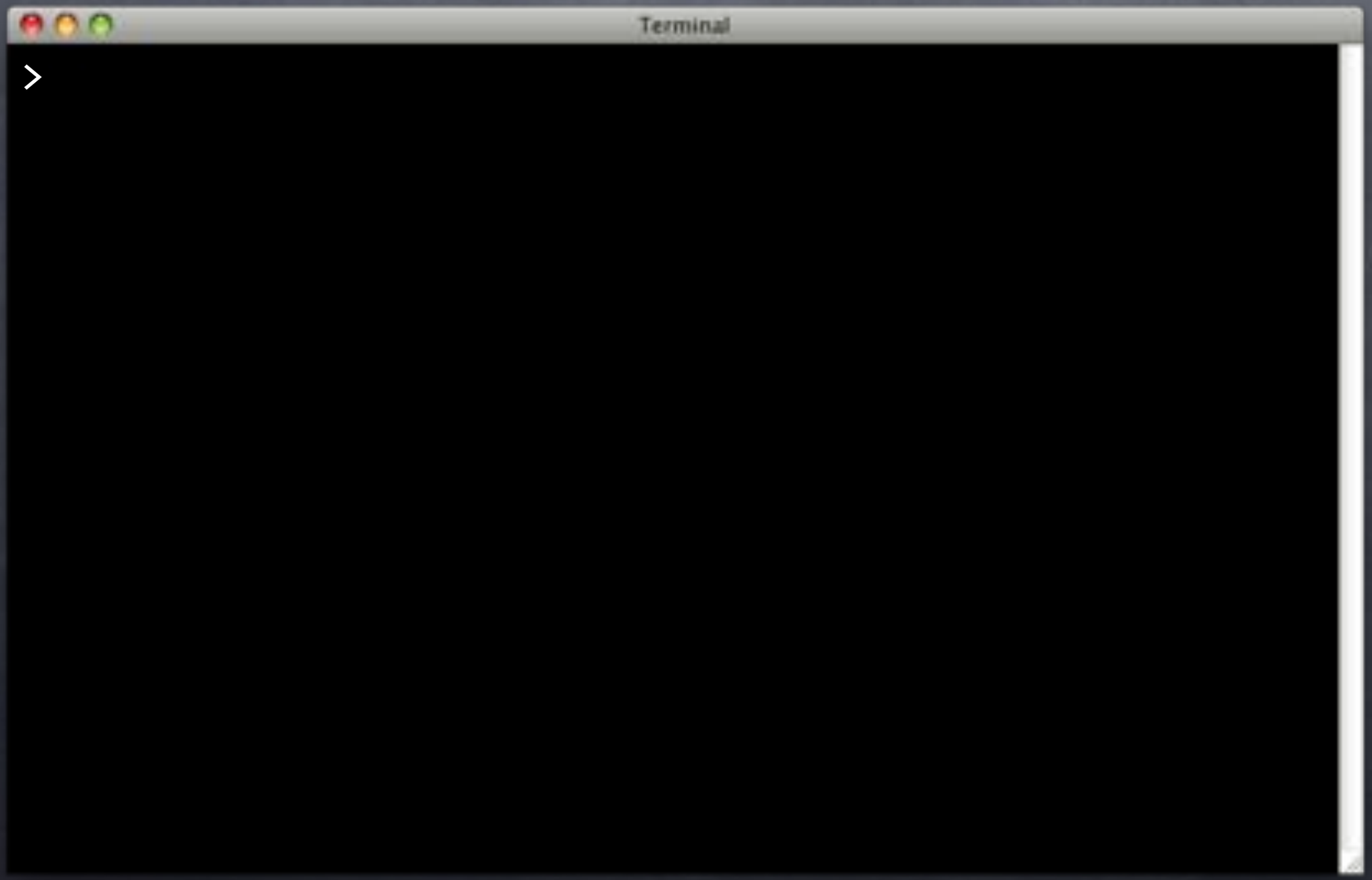
Can We Go Back?

Can We Go Back?

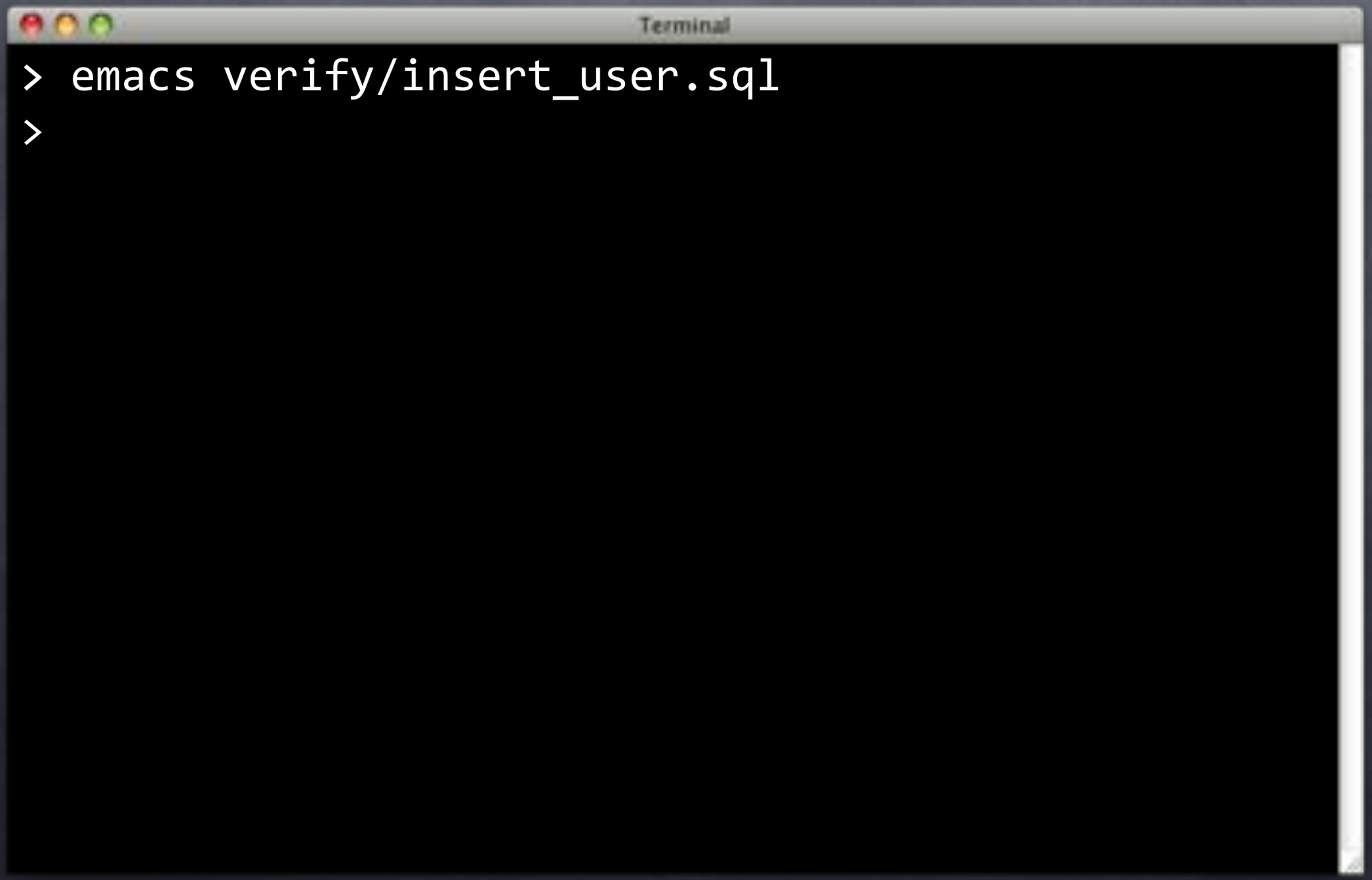
Terminal

```
> sqitch revert --to @HEAD^ -y
  - insert_user .. ok
> psql -d flipr_test -c "
    DELETE FROM flipr.users;
    SELECT flipr.insert_user('foo', 'secr3t'),
           flipr.insert_user('bar', 'secr3t');
    SELECT nickname, password FROM flipr.users;
"
 nickname |             password
----------+----------------------------------
 foo      | 9695da4dd567a19f9b92065f240c6725
 bar      | 9695da4dd567a19f9b92065f240c6725
(2 rows)
```

Verify How?



Verify How?



verify/insert_user.sql

Emacs

```
-- Verify insert_user

BEGIN;

SELECT has_function_privilege(
    'flipr.insert_user(text, text)',
    'execute'
);

COMMIT;
```

-:** verify/insert_u All (SQL[ansi])

verify/insert_user.sql

Emacs

```
-- Verify insert_user

BEGIN;

SELECT has_function_privilege(
    'flipr.insert_user(text, text)',
    'execute'
);

SELECT 1/COUNT(*)
  FROM pg_catalog.pg_proc
 WHERE proname = 'insert_user'
   AND pg_get_functiondef(oid)
       LIKE $$%crypt($2, gen_salt('md5'))%$$;

COMMIT;
```

-:** verify/insert_u All (SQL[ansi])

verify/insert_user.sql

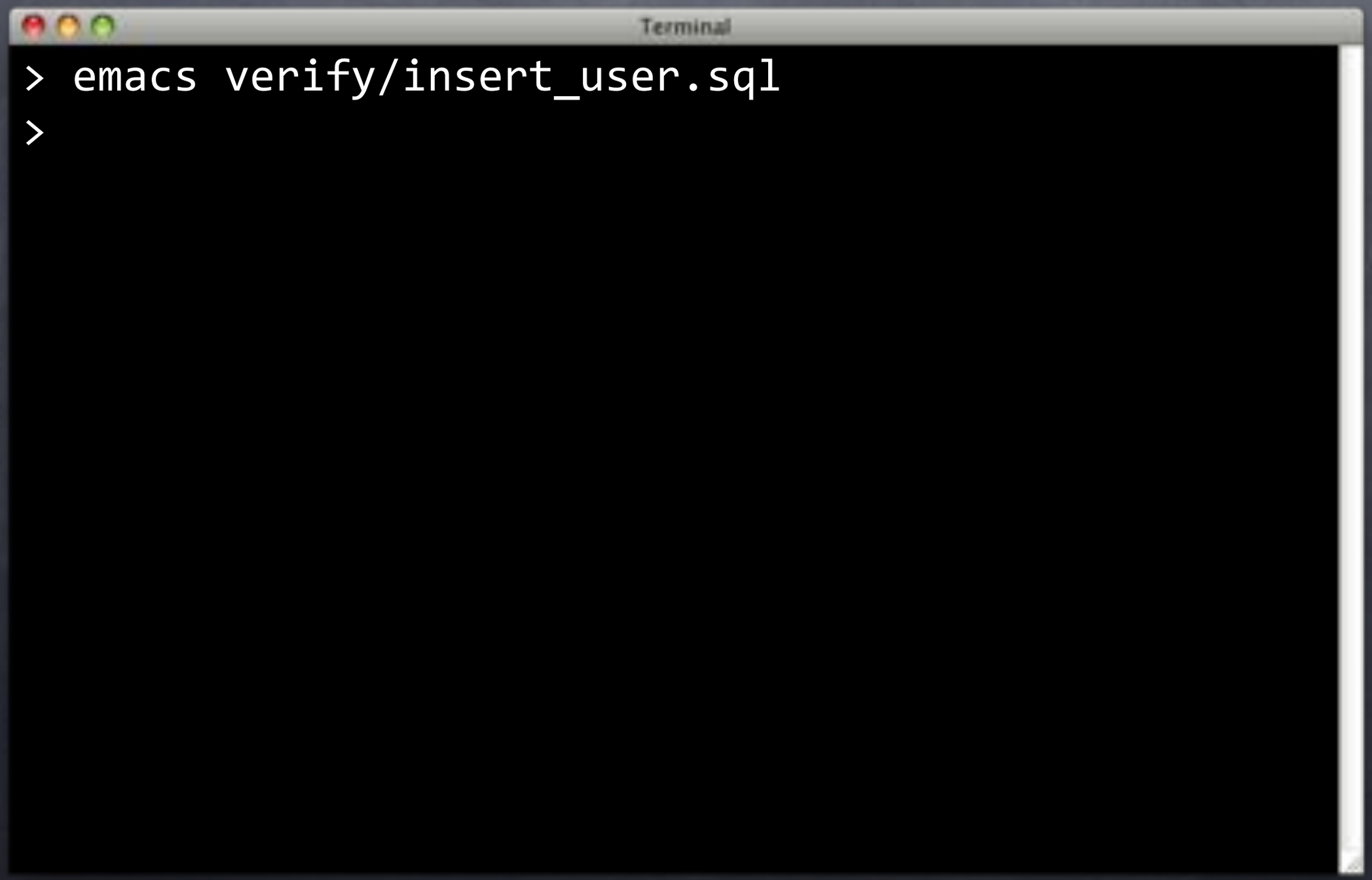
Emacs

```
-- Verify insert_user

BEGIN;

SELECT has_function_privilege(
    'flipr.insert_user(text, text)',
    'execute'
);

SELECT 1/COUNT(*)
  FROM pg_catalog.pg_proc
 WHERE proname = 'insert_user'
   AND pg_get_functiondef(oid)
       LIKE $$%crypt($2, gen_salt('md5'))%$$;

COMMIT;
```

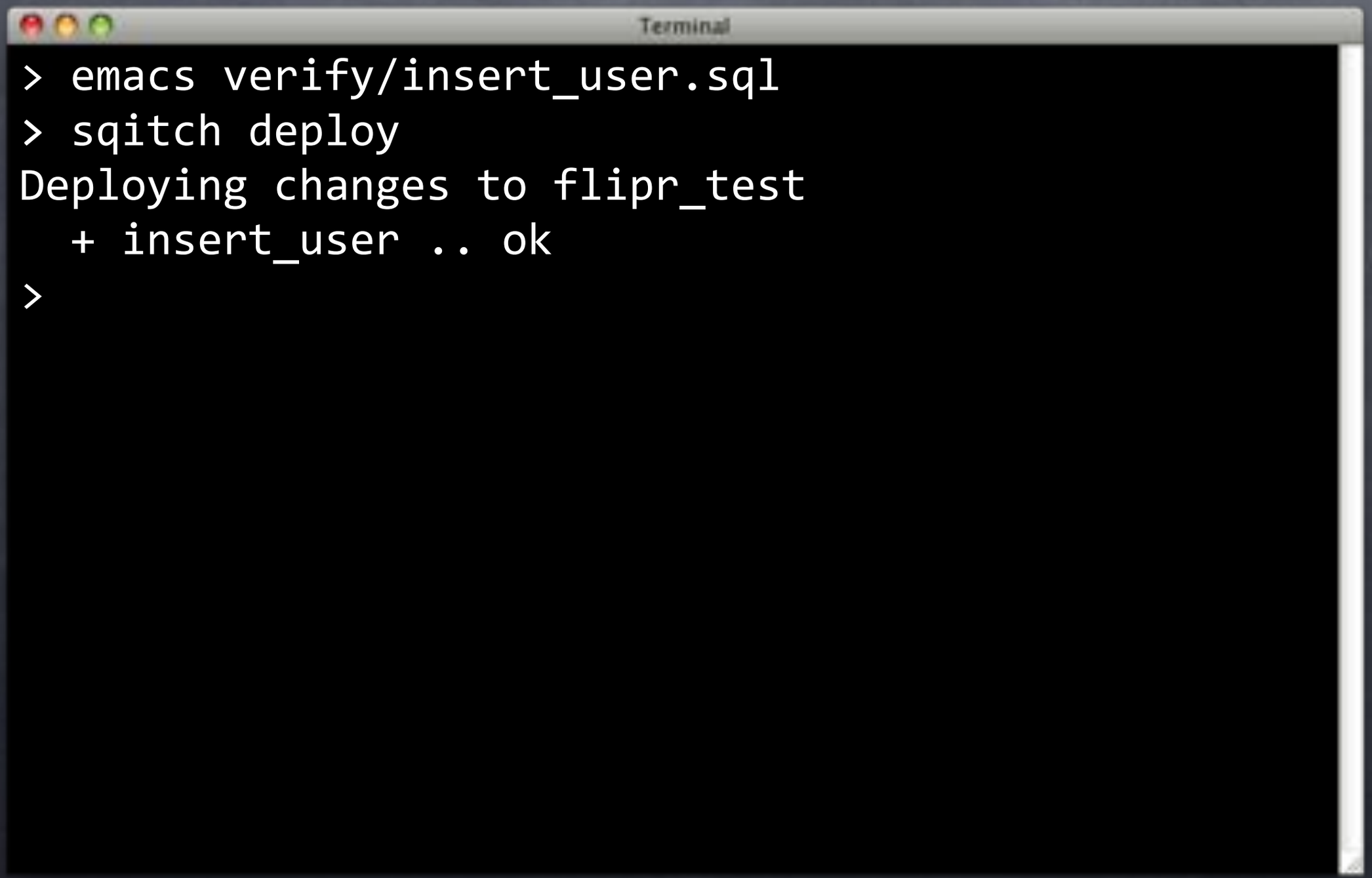
Yeah, compare source.

-:** verify/insert_u All (SQL[ansi])

Let's Go!



Let's Go!


Terminal
> emacs verify/insert_user.sql
> sqitch deploy
Deploying changes to flipr_test
  + insert_user .. ok
>

Let's Go!

Terminal

```
> emacs verify/insert_user.sql
> sqitch deploy
Deploying changes to flipr_test
  + insert_user .. ok
> psql -d flipr_test -c 'DELETE FROM flipr.users'
DELETE 2
>
```

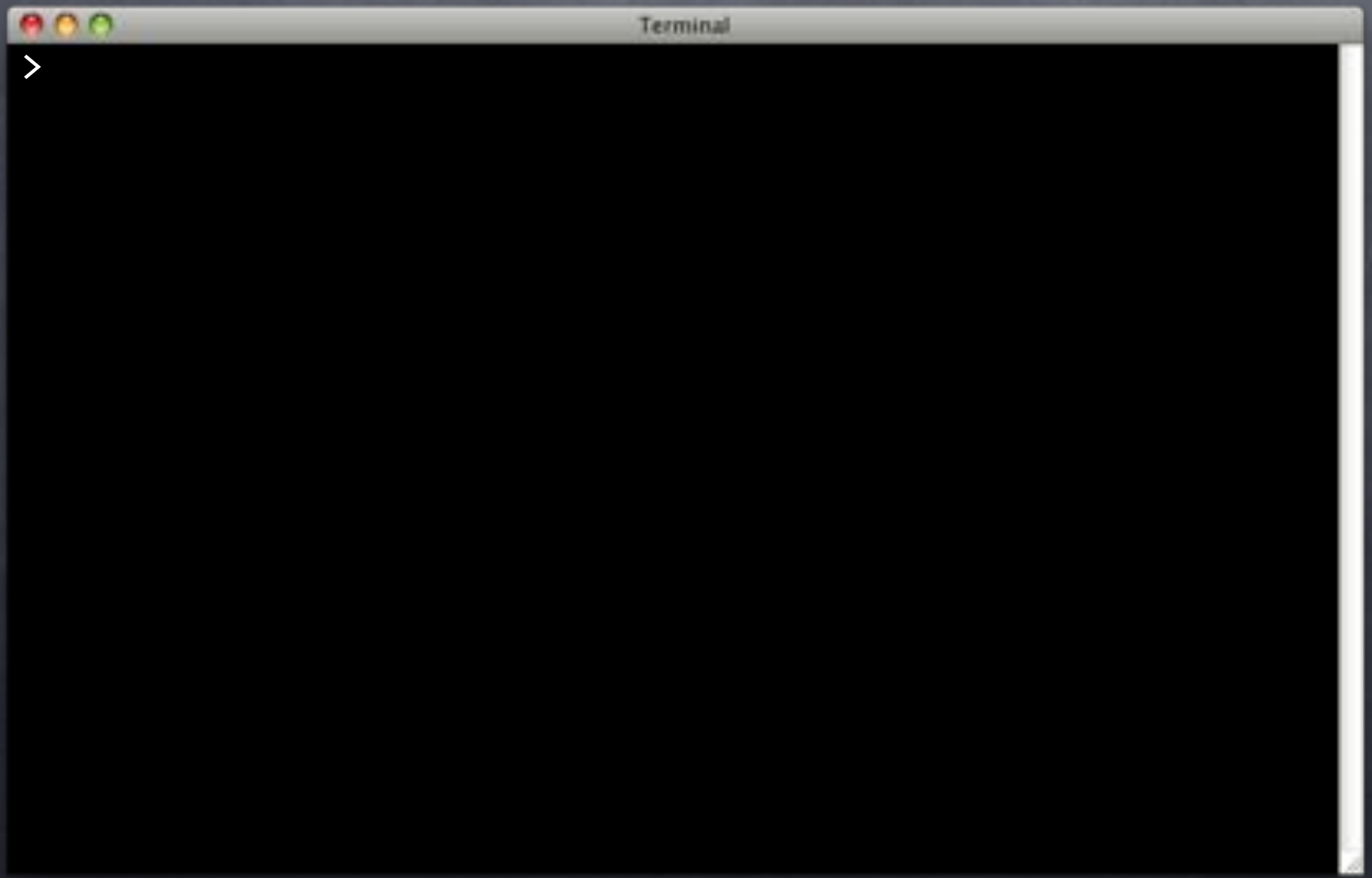
Let's Go!

Terminal

```
> emacs verify/insert_user.sql
> sqitch deploy
Deploying changes to flipr_test
  + insert_user .. ok
> psql -d flipr_test -c 'DELETE FROM flipr.users'
DELETE 2
> pg_prove -d flipr_test test/insert_user.sql
test/insert_user.sql .. ok
All tests successful.
Files=1, Tests=12,  0 wallclock secs
Result: PASS
>
```

Shazam!

Add, Commit, Push, Go



Add, Commit, Push, Go

Terminal

```
> git add .
> git commit -m 'Update insert_user to use pgcrypto.'
[master 59f5823] Update insert_user to use pgcrypto.
 8 files changed, 66 insertions(+), 25 deletions(-)
 create mode 100644 deploy/insert_user@v1.0.0-r1.sql
 create mode 100644 revert/insert_user@v1.0.0-r1.sql
 create mode 100644 verify/insert_user@v1.0.0-r1.sql
>
```

Add, Commit, Push, Go

Terminal

```
> git add .
> git commit -m 'Update insert_user to use pgcrypto.'
[master 59f5823] Update insert_user to use pgcrypto.
 8 files changed, 66 insertions(+), 25 deletions(-)
 create mode 100644 deploy/insert_user@v1.0.0-r1.sql
 create mode 100644 revert/insert_user@v1.0.0-r1.sql
 create mode 100644 verify/insert_user@v1.0.0-r1.sql
> git push
Counting objects: 17, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 1.75 KiB, done.
Total 10 (delta 6), reused 0 (delta 0)
To ../flipr-remote
   5f4c29a..59f5823  master -> master
>
```

Change It Up

Change It Up

- git reset --hard insert_user2

Change It Up

- git reset --hard insert_user2
- Rework change_pass

Change It Up

- git reset --hard insert_user2
- Rework change_pass
- Use pgcrypo

Change It Up

- git reset --hard insert_user2
- Rework change_pass
- Use pgcrypo
- Test first!

Change It Up

- git reset --hard insert_user2
- Rework change_pass
- Use pgcrypo
- Test first!
- Bundle, tag, release

Change It Up

- git reset --hard insert_user2
- Rework change_pass
- Use pgcrypo
- Test first!
- Bundle, tag, release
- https://github.com/theory/agile-flipr.git

I'm afraid I have some bad news...

DEADPOOL

Antisocial Networking Startup Flipr Heads To The Deadpool

by **Michael Arrington** on November 2, 2010

28 Comments

I loved this site.

Flipr, an online “antisocial networking” site that encouraged users to alienate each other in order to increase their antisocial cred, is shutting down. The startup’s homepage now consists of a letter to Flipr users instructing them to download their “flips” by November 30, at which point nearly all of the service’s features will be taken offline and data deleted.

In the letter, Flipr CEO David Wheeler writes that despite ample venture funding and a dedicated team of database developers, the site underestimated people’s willingness to be assholes. This is not something I can relate to, although from what I’ve been told by more polite society, it is indeed the case. Such a shame.

Read More

I'm afraid it's true.

I'm sorry I have no money left to pay you.

I hope you enjoyed working here.

And that you're able to use the skills you've gained in your next job.

Git Skillz

Git Skillz

- Branching

Git Skillz

- Branching
- Diffing

Git Skillz

- Branching
- Diffing
- Rebasing

Git Skillz

- Branching
- Diffing
- Rebasing
- Merging

Git Skillz

- Branching
- Diffing
- Rebasing
- Merging
- Committing

Git Skillz

- Branching
- Diffing
- Rebasing
- Merging
- Committing
- Pushing

MOAR Git

MOAR Git

- Bisecting

MOAR Git

- Bisecting
- Blaming

MOAR Git

- Bisecting
- Blaming
- Pull requests

MOAR Git

- Bisecting
- Blaming
- Pull requests
- Submitting patches

MOAR Git

- Bisecting
- Blaming
- Pull requests
- Submitting patches
- Rewriting history

MOAR Git

- Bisecting
- Blaming
- Pull requests
- Submitting patches
- Rewriting history
- Log formatting

MOAR Git

- Bisecting
- Blaming
- Pull requests
- Submitting patches
- Rewriting history
- Log formatting
- git help --all

pgTAP Skillz

pgTAP Skillz

- TDDD

pgTAP Skillz

- TDDD
- Schema testing

pgTAP Skillz

- TDDD
- Schema testing
- Scalar testing

pgTAP Skillz

- TDDD
- Schema testing
- Scalar testing
- Functional testing

pgTAP Skillz

- TDDD
- Schema testing
- Scalar testing
- Functional testing
- Relational testing

MOAR pgTAP

MOAR pgTAP

- Testing privileges

MOAR pgTAP

- Testing privileges
- Mocking interfaces

MOAR pgTAP

- Testing privileges
- Mocking interfaces
- Custom test functions

MOAR pgTAP

- Testing privileges
- Mocking interfaces
- Custom test functions
- xUnit-style testing

MOAR pgTAP

- Testing privileges
- Mocking interfaces
- Custom test functions
- xUnit-style testing
  - Tests maintained in functions

MOAR pgTAP

- Testing privileges
- Mocking interfaces
- Custom test functions
- xUnit-style testing
  - Tests maintained in functions
- http://pgtap.org/documentation.html

Sqitch Skillz

Sqitch Skillz

- Adding changes

Sqitch Skillz

- Adding changes
- Dependency management

Sqitch Skillz

- Adding changes
- Dependency management
- Deploying changes

Sqitch Skillz

- Adding changes
- Dependency management
- Deploying changes
- Verifying changes

Sqitch Skillz

- Adding changes
- Dependency management
- Deploying changes
- Verifying changes
- Reverting changes

Sqitch Skillz

- Adding changes
- Dependency management
- Deploying changes
- Verifying changes
- Reverting changes
- Rebasing changes

Sqitch Skillz

- Adding changes
- Dependency management
- Deploying changes
- Verifying changes
- Reverting changes
- Rebasing changes
- Reworking changes

Sqitch Skillz

- Adding changes
- Dependency management
- Deploying changes
- Verifying changes
- Reverting changes
- Rebasing changes
- Reworking changes

MOAR Sqitch

MOAR Sqitch

- Cross-project dependencies

MOAR Sqitch

- Cross-project dependencies
  - Multiple projects, one database

MOAR Sqitch

- Cross-project dependencies
  - Multiple projects, one database
- Changing Branches

MOAR Sqitch

- Cross-project dependencies
  - Multiple projects, one database
- Changing Branches
  - Checkout command

MOAR Sqitch

- Cross-project dependencies
  - Multiple projects, one database
- Changing Branches
  - Checkout command
  - Reverts to last common change

MOAR Sqitch

- Cross-project dependencies
  - Multiple projects, one database
- Changing Branches
  - Checkout command
  - Reverts to last common change
  - Changes Git branch

MOAR Sqitch

- Cross-project dependencies
  - Multiple projects, one database
- Changing Branches
  - Checkout command
  - Reverts to last common change
  - Changes Git branch
  - Deploys

MOAR Sqitch

- Cross-project dependencies
  - Multiple projects, one database
- Changing Branches
  - Checkout command
  - Reverts to last common change
  - Changes Git branch
  - Deploys
- sqitch help

Good luck out there.

Agile Database Development

David E. Wheeler
iovation

PGCon 2013
Ottawa