

# Agile Database Development

David E. Wheeler  
iovation

PGCon 2013  
Ottawa



Text: Attribution-Noncommercial-Share Alike 3.0 United States:  
<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>  
Images licensed independently and © Their respective owners.



# Database Development

David E. Wheeler  
iovation

PGCon 2013  
Ottawa



Text: Attribution-Noncommercial-Share Alike 3.0 United States:  
<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>  
Images licensed independently and © Their respective owners.



**David E. Wheeler**

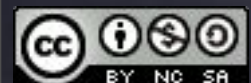
**PGCon 2013  
Ottawa**



Text: Attribution-Noncommercial-Share Alike 3.0 United States:  
<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>  
Images licensed independently and © Their respective owners.

**David E. Wheeler**

**PGCon 2013  
Ottawa**



License: Attribution-NonCommercial-Share Alike 3.0 United States: <http://creativecommons.org/licenses/by-nc-sa/3.0/us/>

# Build My VC-Funded App for Me for Free

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



PGCon 2013  
Ottawa



License: Attribution-NonCommercial-Share Alike 3.0 United  
States: <http://creativecommons.org/licenses/by-nc-sa/3.0/us/>

# 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 waaaay too long

# This is Genius

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

# This is Genius

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





# Flipr

antisocial network

# 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





© Duncan Davidson

<http://flic.kr/p/2honiQ> © 2007 James Duncan Davidson. All rights reserved. Used with permission.



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](http://metacpan.org/module/pg_prove)



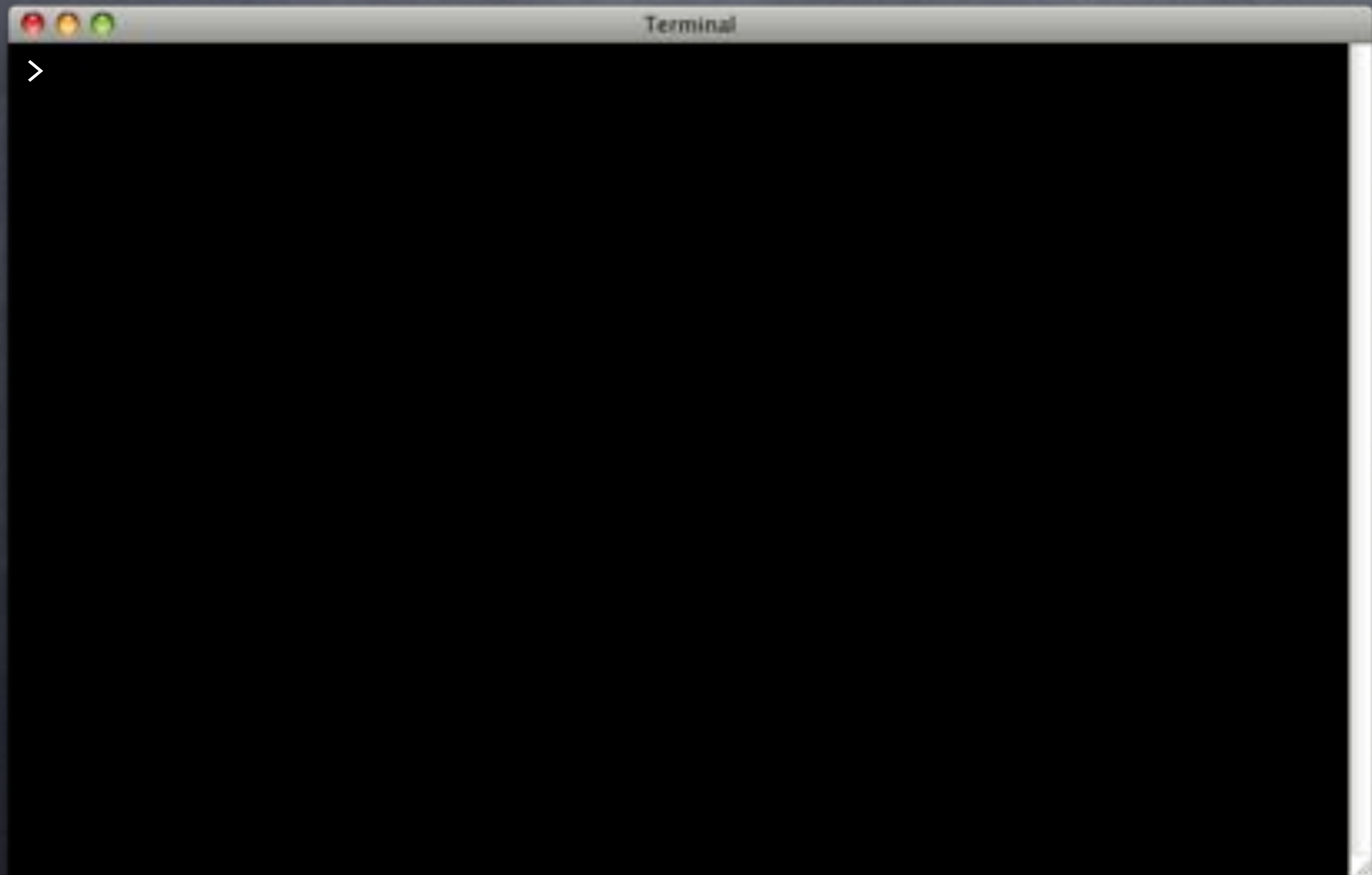
We good?



Let's do this thang.

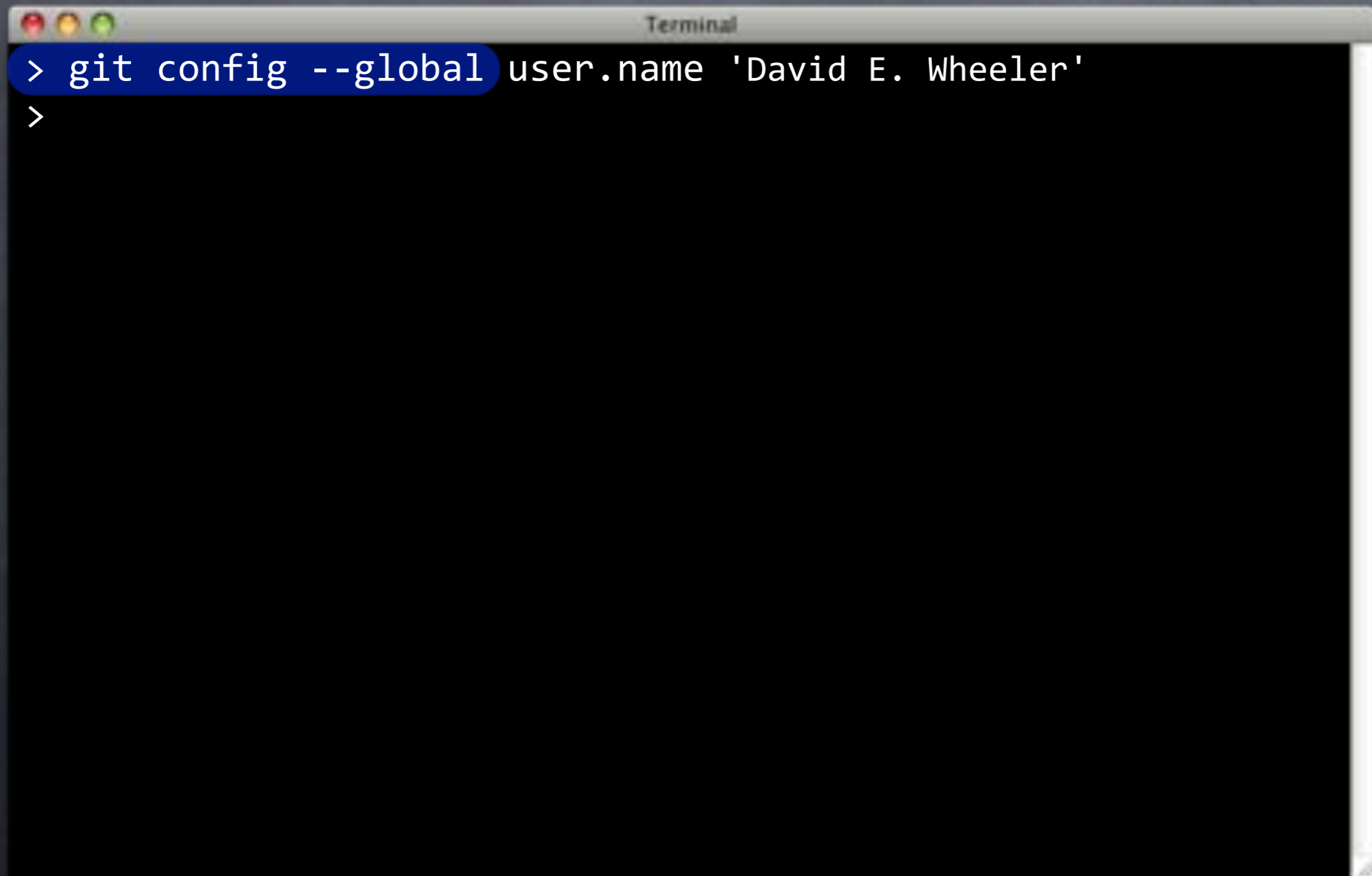


# Who Am I?



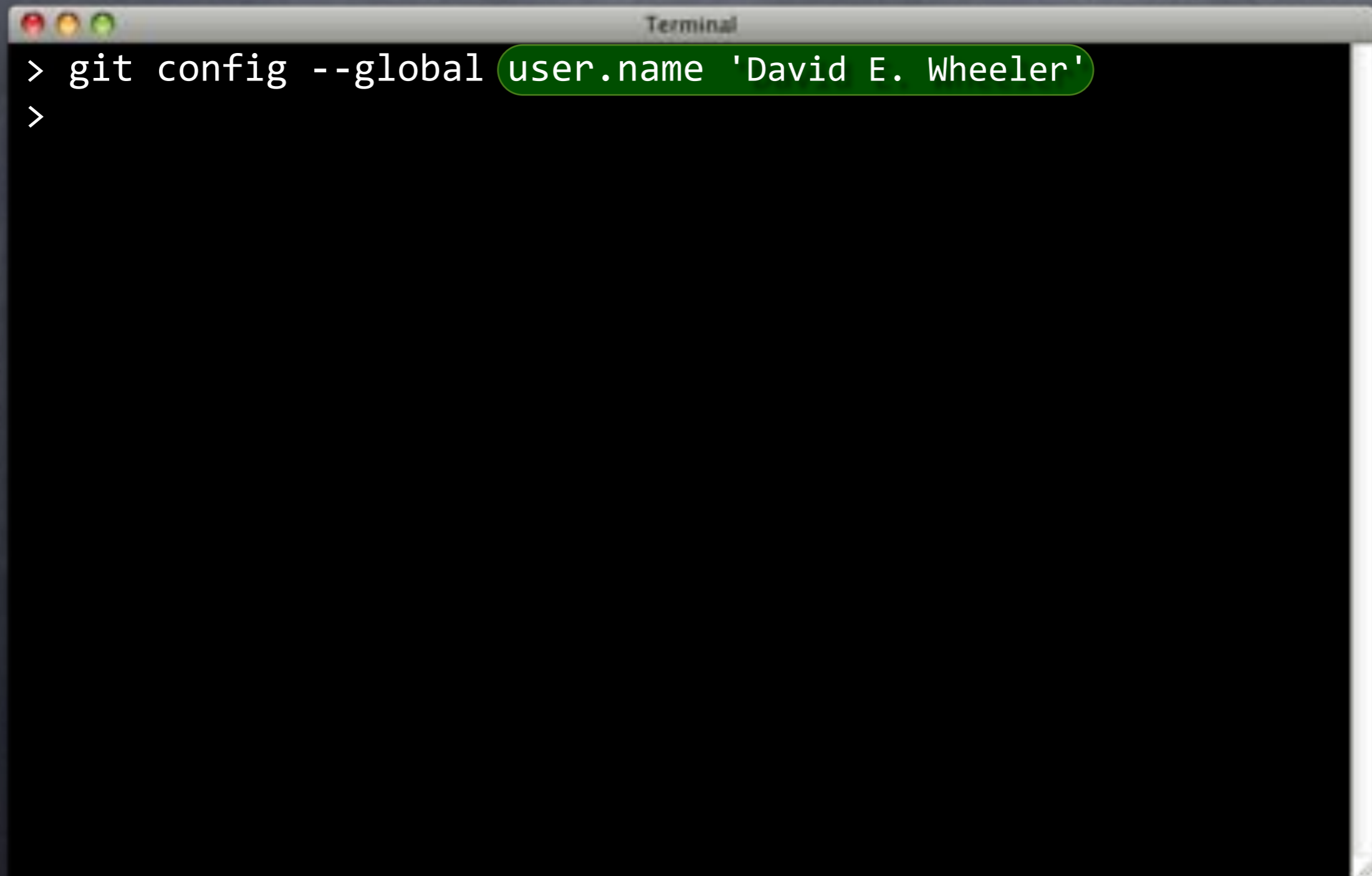


# Who Am I?

A terminal window titled "Terminal" with a dark background and a light gray title bar. The window contains a single line of text: "> git config --global user.name 'David E. Wheeler'". The text is white, and the prompt character ">" is positioned at the beginning of the line. The terminal window has three small colored circles (red, yellow, green) in the top-left corner of the title bar.

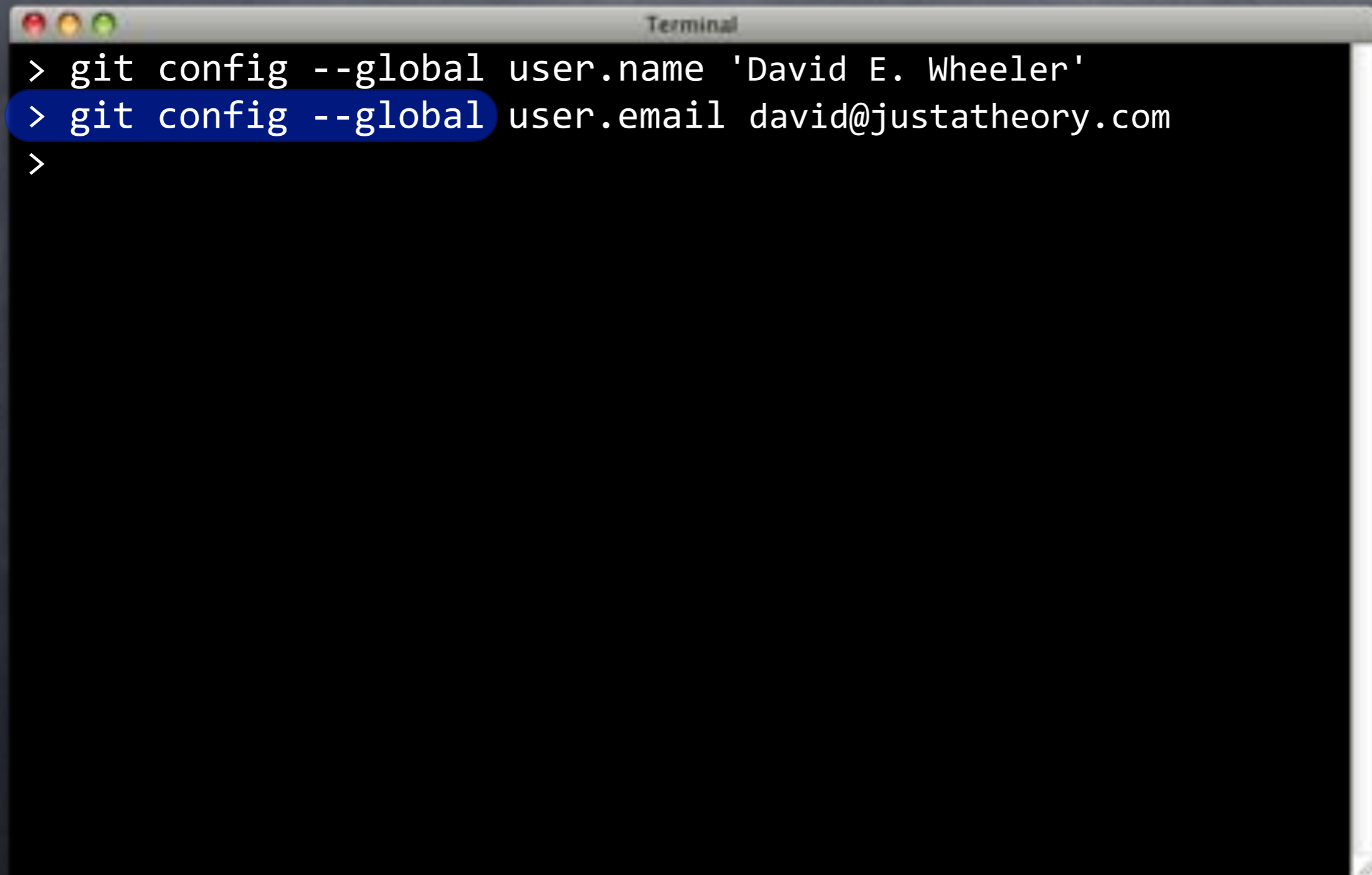
```
> git config --global user.name 'David E. Wheeler'
```

# Who Am I?

A terminal window titled "Terminal" with a dark background. The window contains the command `> git config --global user.name 'David E. Wheeler'` followed by a new prompt `>`. The text `user.name 'David E. Wheeler'` is highlighted with a green rounded rectangle.

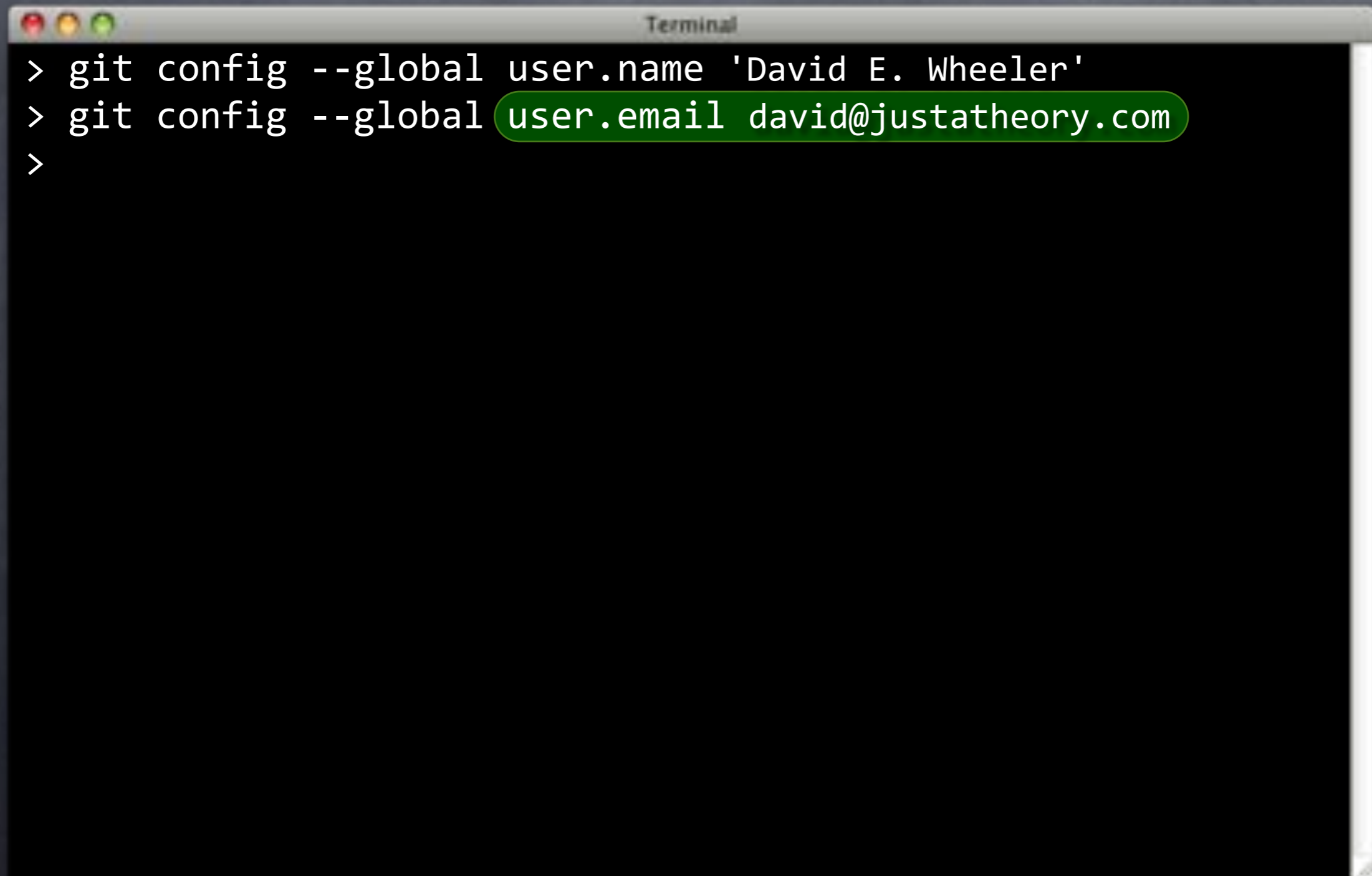
```
> git config --global user.name 'David E. Wheeler'  
>
```

# Who Am I?

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows three lines of text: a prompt followed by a git config command for the user name, a second prompt followed by a git config command for the user email (highlighted in blue), and a third prompt.

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

# Who Am I?

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal contains three lines of text: a prompt followed by the command "git config --global user.name 'David E. Wheeler'", a second prompt followed by "git config --global user.email david@justatheory.com", and a third prompt. The second line is highlighted with a green background.

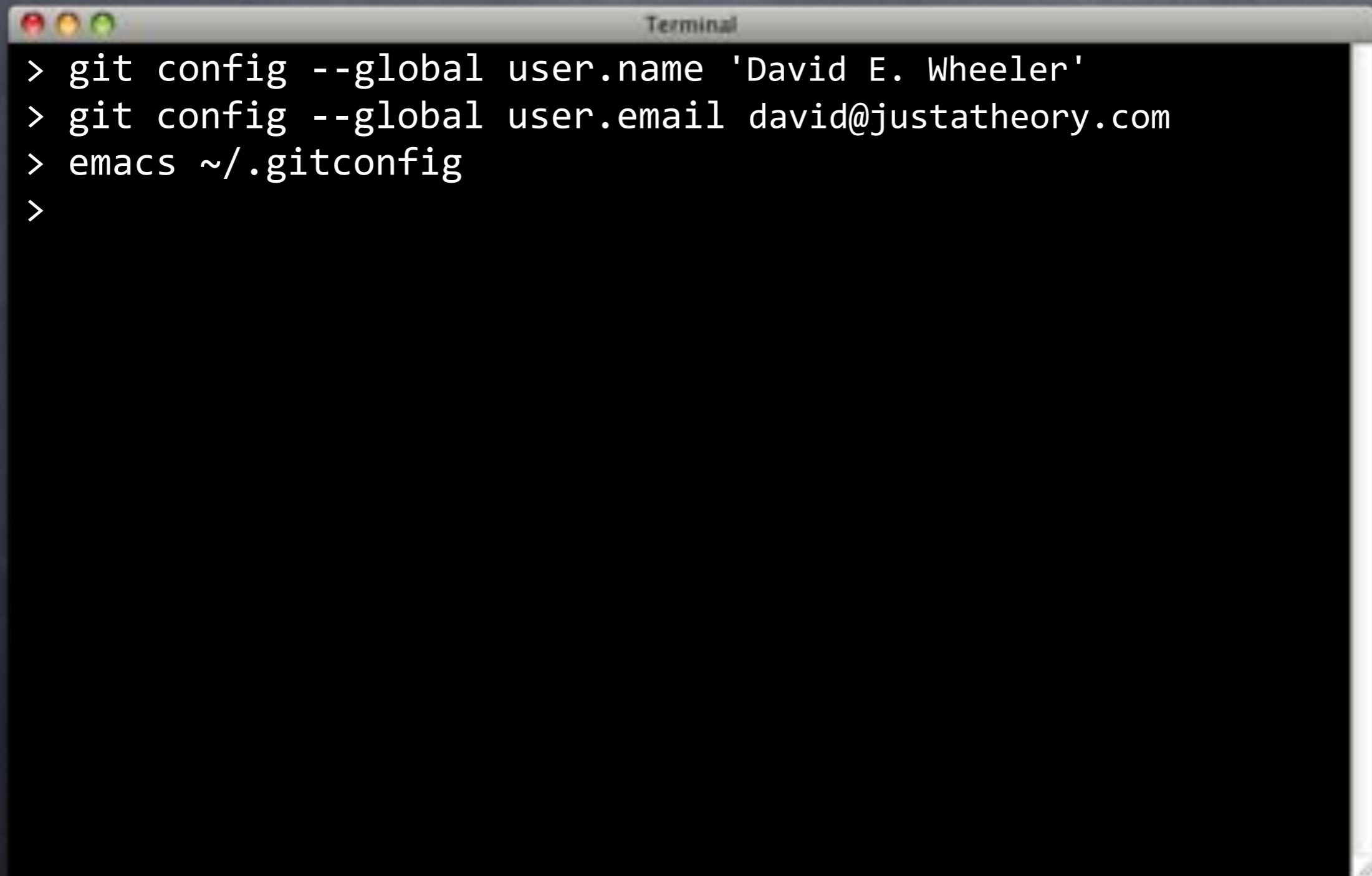
```
> git config --global user.name 'David E. Wheeler'  
> git config --global user.email david@justatheory.com  
>
```

# 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?

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal contains the following commands:

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

# ~/.gitconfig



The image shows a screenshot of an Emacs editor window. The window title is "Emacs". The main content area is dark green and contains the following text:

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

The status bar at the bottom of the window shows the file path "~/.gitconfig" and the encoding "All (SQL[ansi])".

# ~/.gitconfig



```
[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

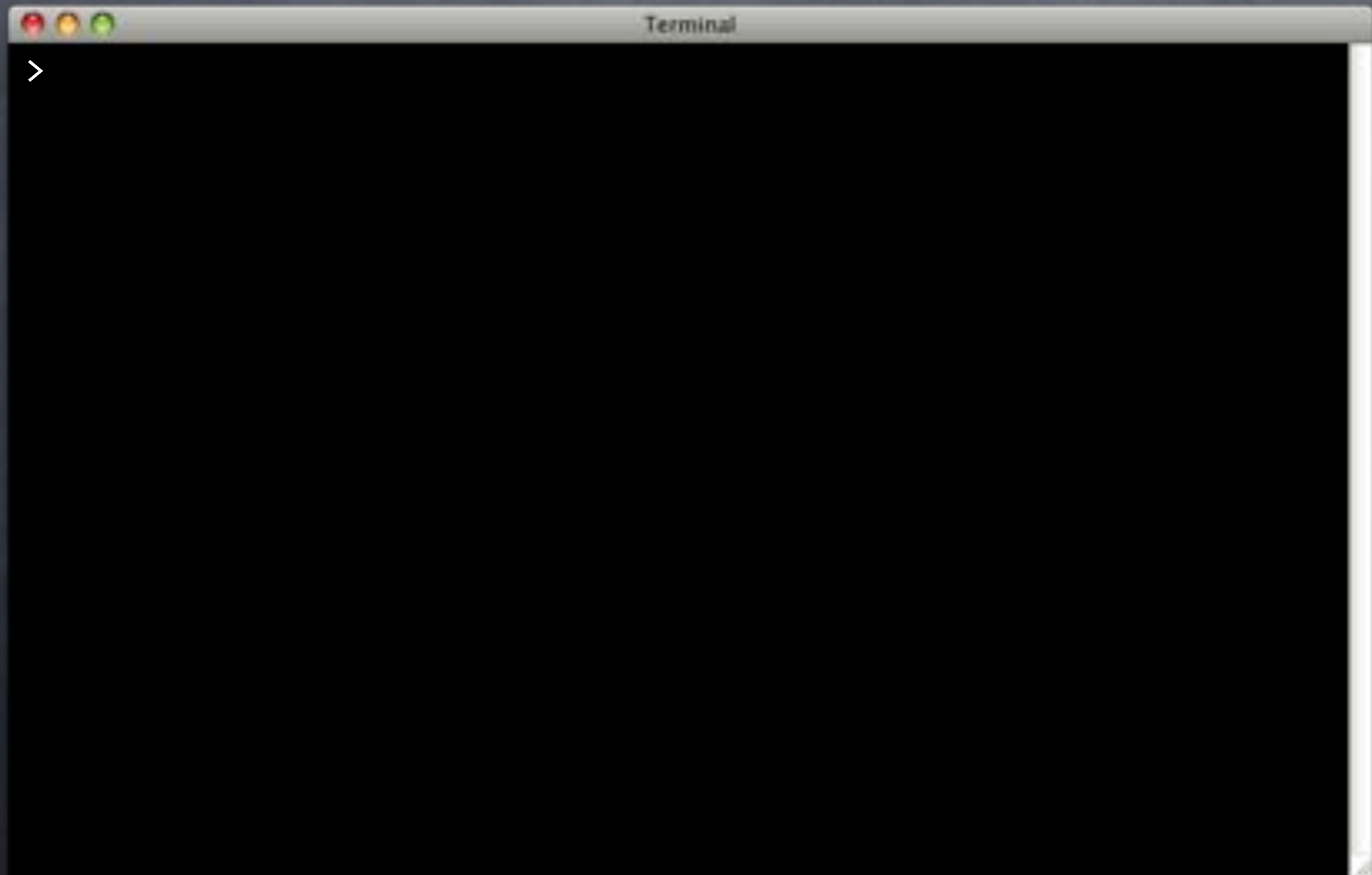
# Create a Remote

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

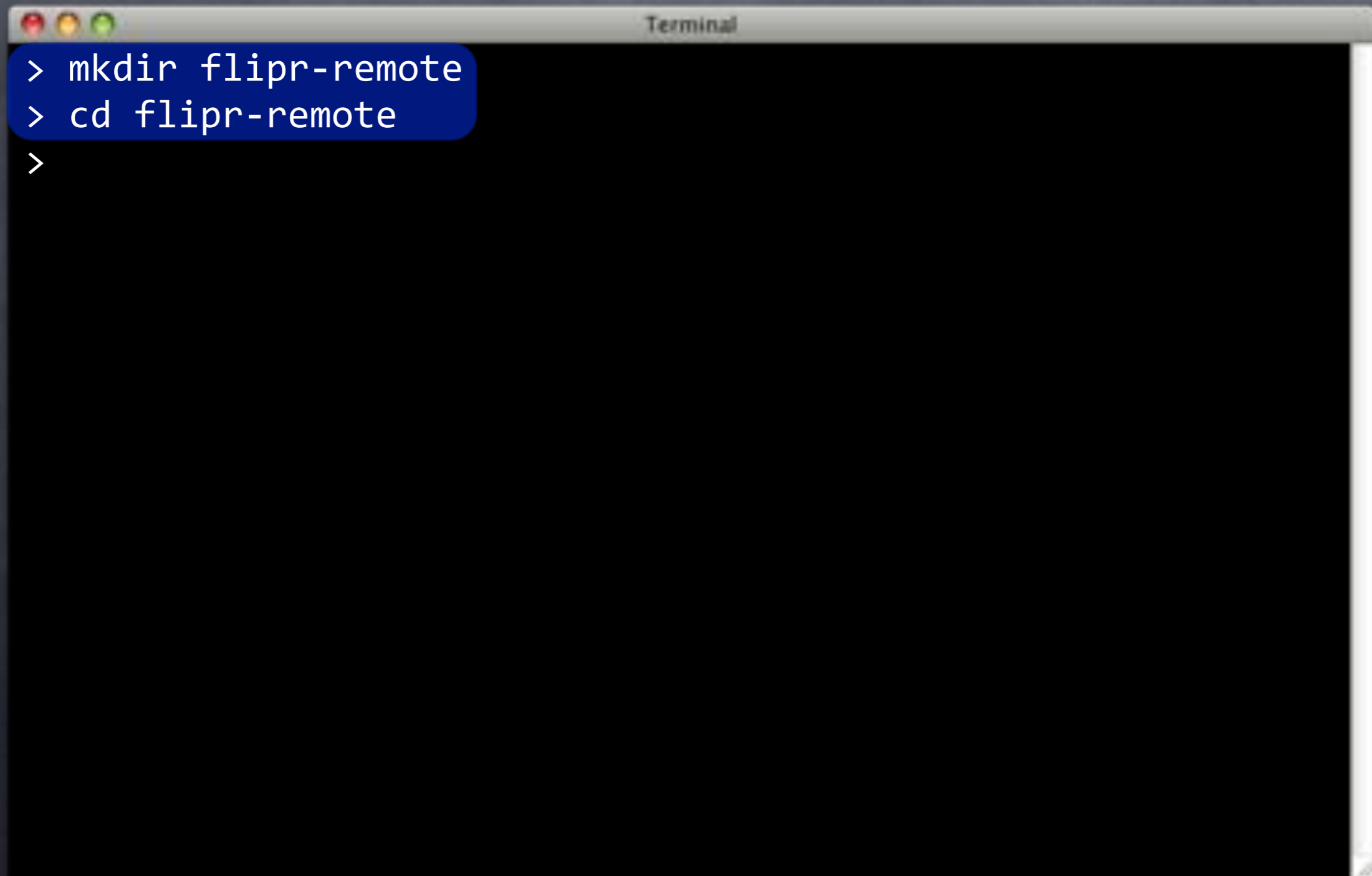
**Quick setup** — if you've done this kind of thing before

🍏 Setup in Mac or HTTP SSH `git@github.com:theory/agile-flipr-test.git`

# Or a Local Remote



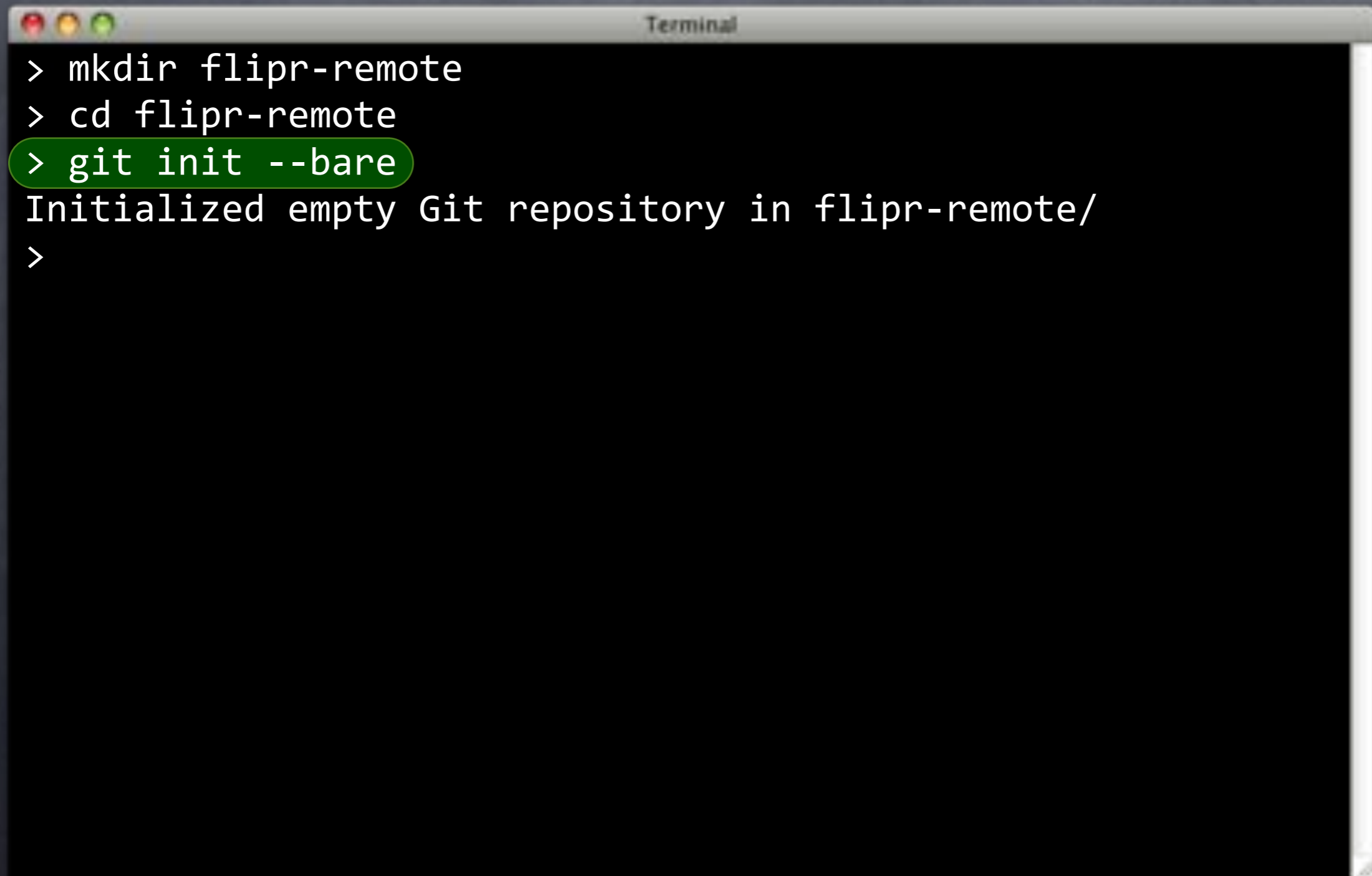
# Or a Local Remote

A screenshot of a macOS Terminal window. The window title is "Terminal". The terminal content shows three lines of commands: the first line is "> mkdir flipr-remote", the second line is "> cd flipr-remote", and the third line is ">". The first two lines are highlighted with a blue background.

```
Terminal  
> mkdir flipr-remote  
> cd flipr-remote  
>
```



# Or a Local Remote

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows a sequence of commands: `> mkdir flipr-remote`, `> cd flipr-remote`, and `> git init --bare`. The last command is highlighted with a green background. Below the command, the output reads "Initialized empty Git repository in flipr-remote/" followed by a prompt `>`.

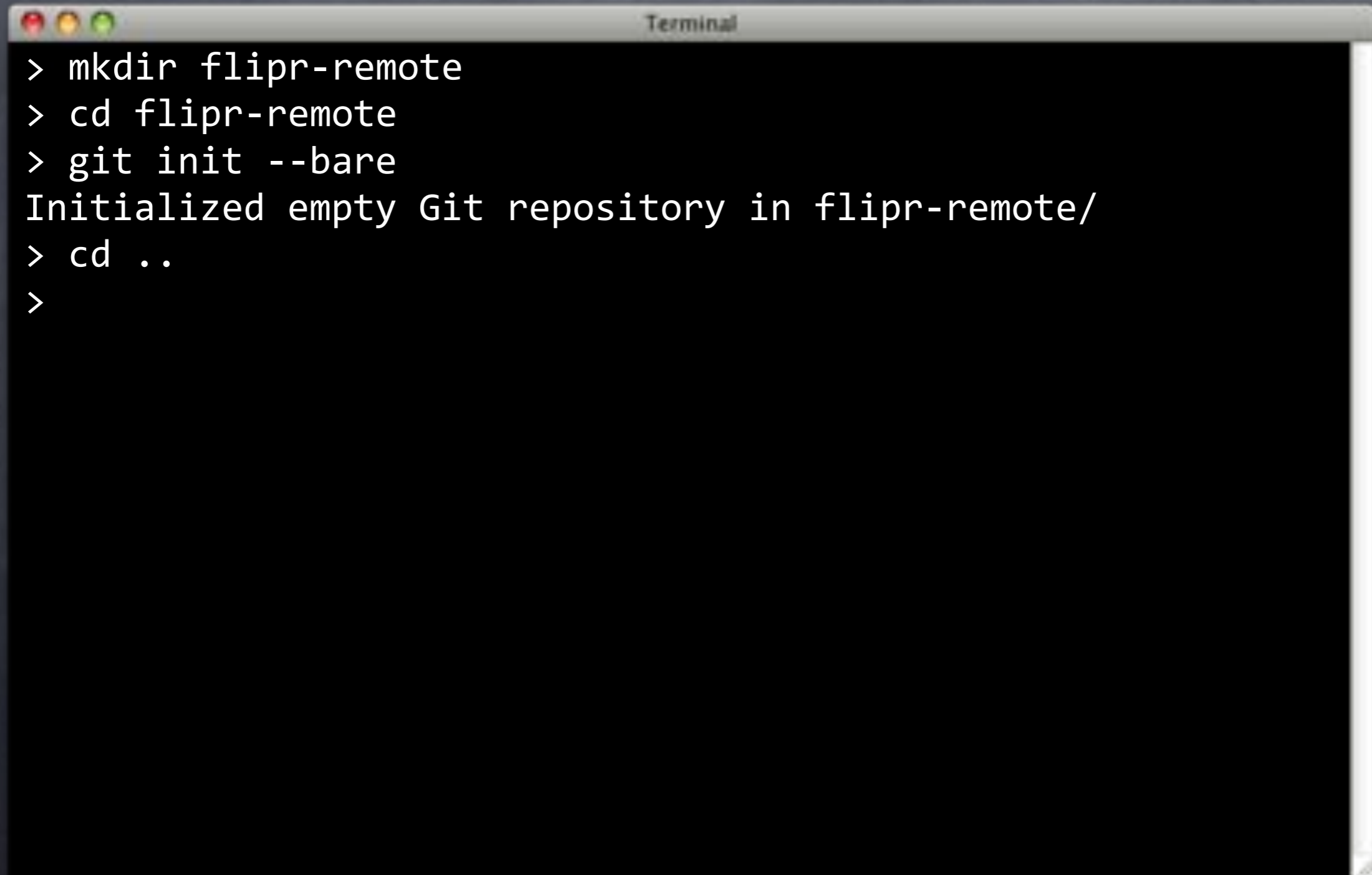
```
Terminal
> mkdir flipr-remote
> cd flipr-remote
> git init --bare
Initialized empty Git repository in flipr-remote/
>
```

# Or a Local Remote

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

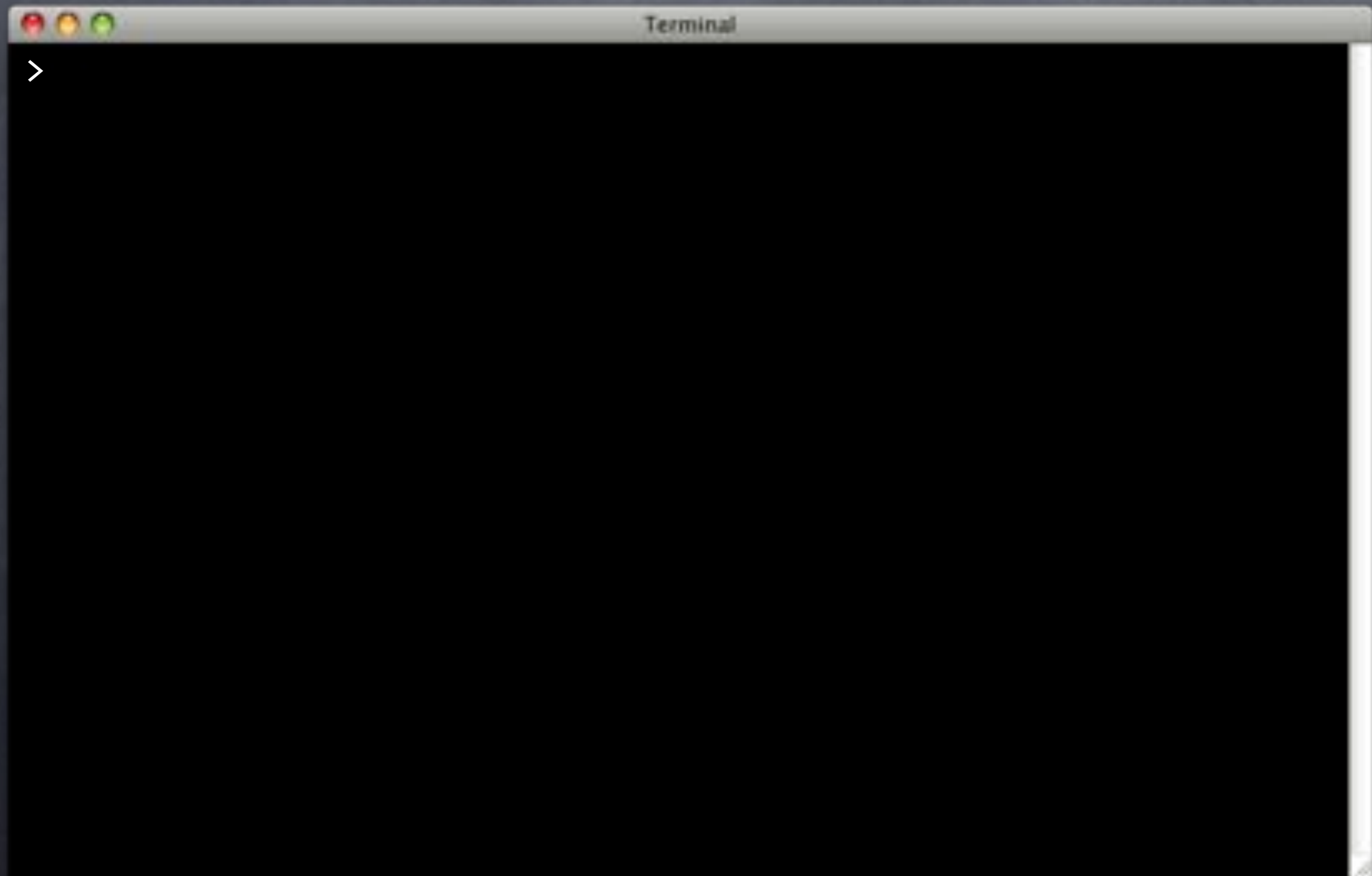
Remote URL is simply the path to this directory.

# Or a Local Remote

A terminal window with a title bar that says "Terminal". The window contains a series of shell commands and their output. The commands are: > mkdir flipr-remote, > cd flipr-remote, > git init --bare. The output of the last command is "Initialized empty Git repository in flipr-remote/". The terminal then shows > cd .. and > on separate lines.

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

# Gitiup



# Gitup

A terminal window titled "Terminal" with a dark background and light text. The window shows a sequence of commands and their output. The first two commands, "mkdir flipr-db" and "cd flipr-db", are highlighted with a blue background. The third command, "git init", is followed by the output "Initialized empty Git repository in flipr-db/.git/" and a new prompt character ">".

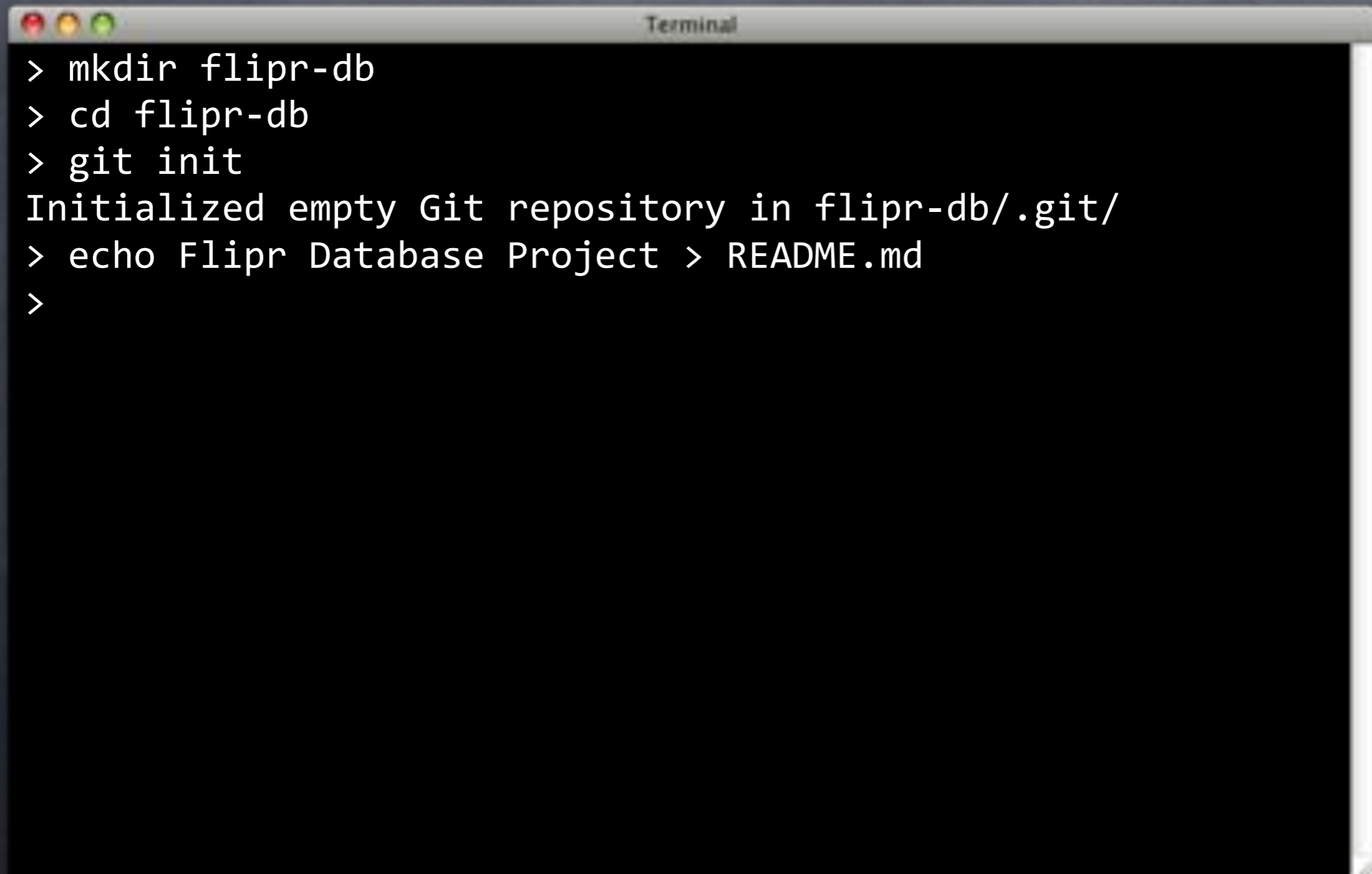
```
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
>
```

# Gitup

A terminal window titled "Terminal" with a dark background and light text. The window shows a sequence of commands: > mkdir flipr-db, > cd flipr-db, and > git init. The last command is highlighted with a green background. Below the last command, the output "Initialized empty Git repository in flipr-db/.git/" is displayed, followed by a new prompt >.

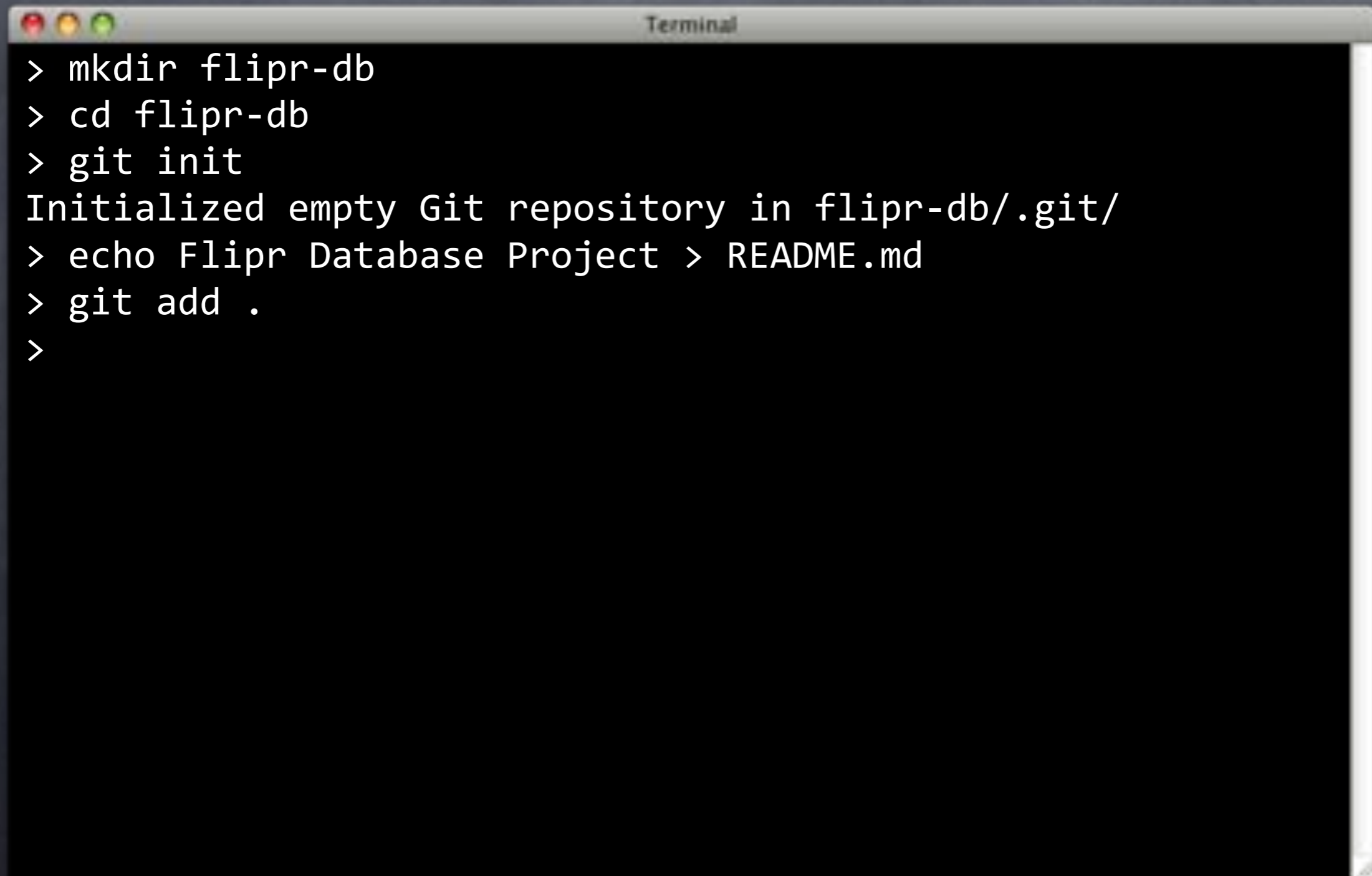
```
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
>
```

# Gitup

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows a sequence of commands and their output for setting up a Git repository. The commands are: `mkdir flipr-db`, `cd flipr-db`, `git init`, and `echo Flipr Database Project > README.md`. The output of `git init` is "Initialized empty Git repository in flipr-db/.git/".

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

# Gitup

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows a series of commands and their output for setting up a Git repository. The commands are: `mkdir flipr-db`, `cd flipr-db`, `git init`, `echo Flipr Database Project > README.md`, and `git add .`. The output for `git init` is "Initialized empty Git repository in flipr-db/.git/".

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



# Gitup

```
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
>
```

# Gitup

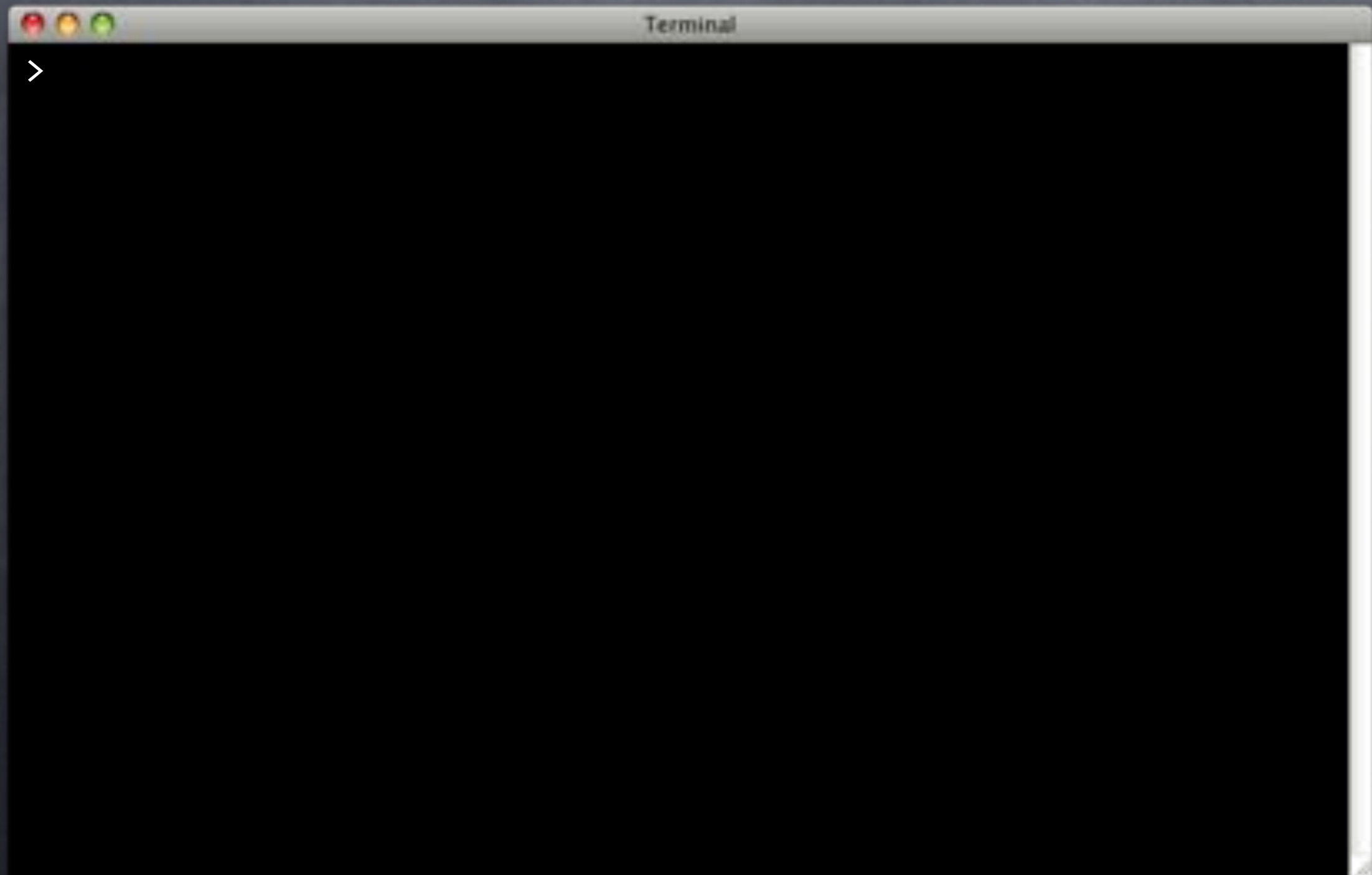
```
Terminal
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
> echo Flipr Database Project > README.md
> git add .
> git commit -m 'Initialize repo, add README.'
→ [master (root-commit) 2c4b510] Initialize repo, add README.
1 file changed, 1 insertion(+)
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

# Gitup

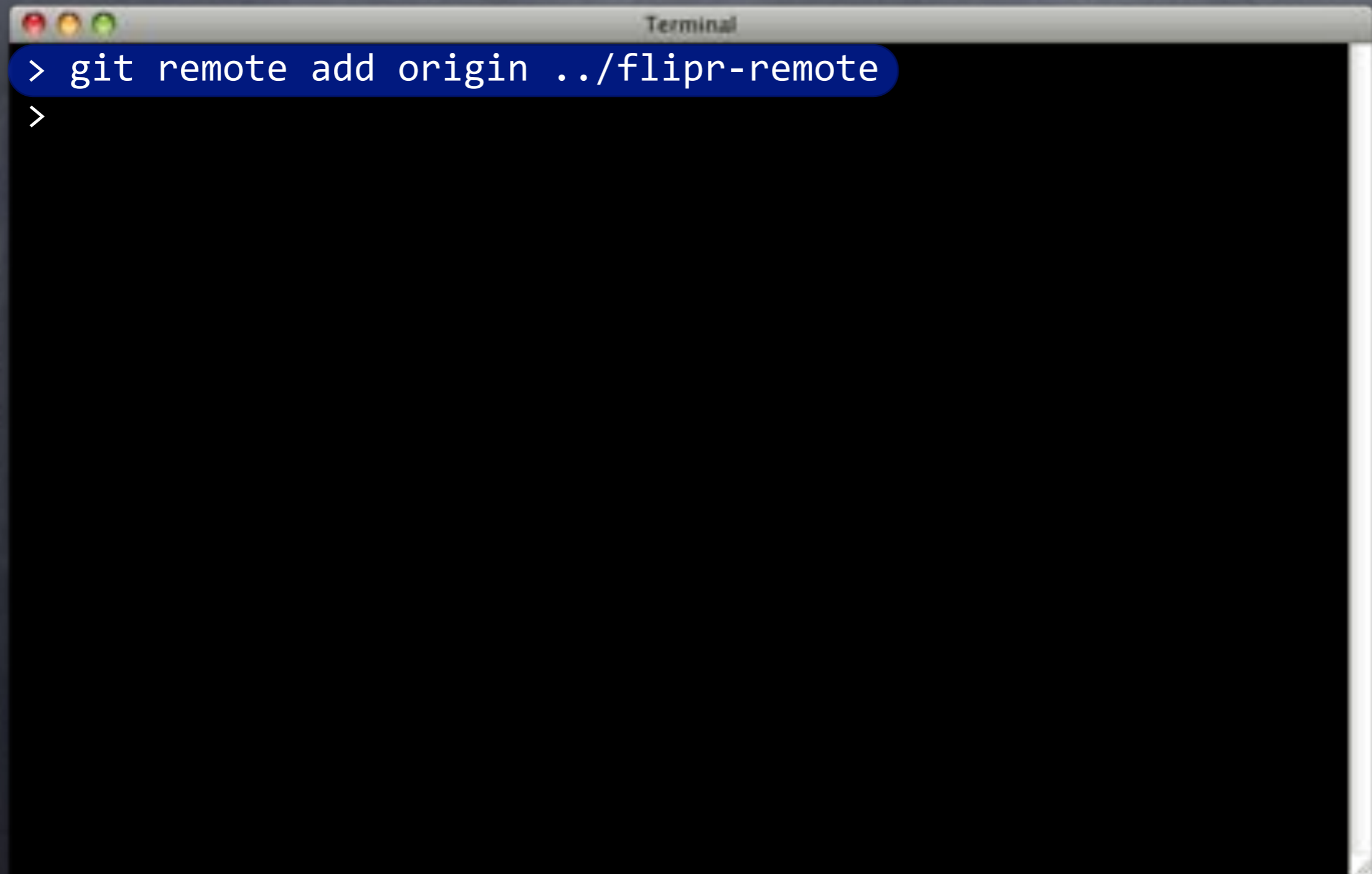
```
Terminal
> mkdir flipr-db
> cd flipr-db
> git init
Initialized empty Git repository in flipr-db/.git/
> echo Flipr Database Project > README.md
> git add .
> git commit -m 'Initialize repo, add README.'
[master (root-commit) 2c4b510] Initialize repo, add README.
1 file changed, 1 insertion(+)
 create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

We have a  
Git repo.

# Origin



# Origin

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) in the top-left corner. The main area of the terminal is black with white text. The text shows a prompt character ">" followed by the command "git remote add origin ../flipr-remote". The command is highlighted with a blue background. Below the command, there is another prompt character ">" on a new line.

```
> git remote add origin ../flipr-remote  
>
```

# Origin

A terminal window titled "Terminal" with a dark background. The first line of text is highlighted in blue and contains the command: `> git remote add origin ../flipr-remote`. The second line shows a prompt character `>`. A blue callout box with a white border and a pointer to the `origin` parameter in the command contains the text "Or remote URL" in white, bold, sans-serif font.

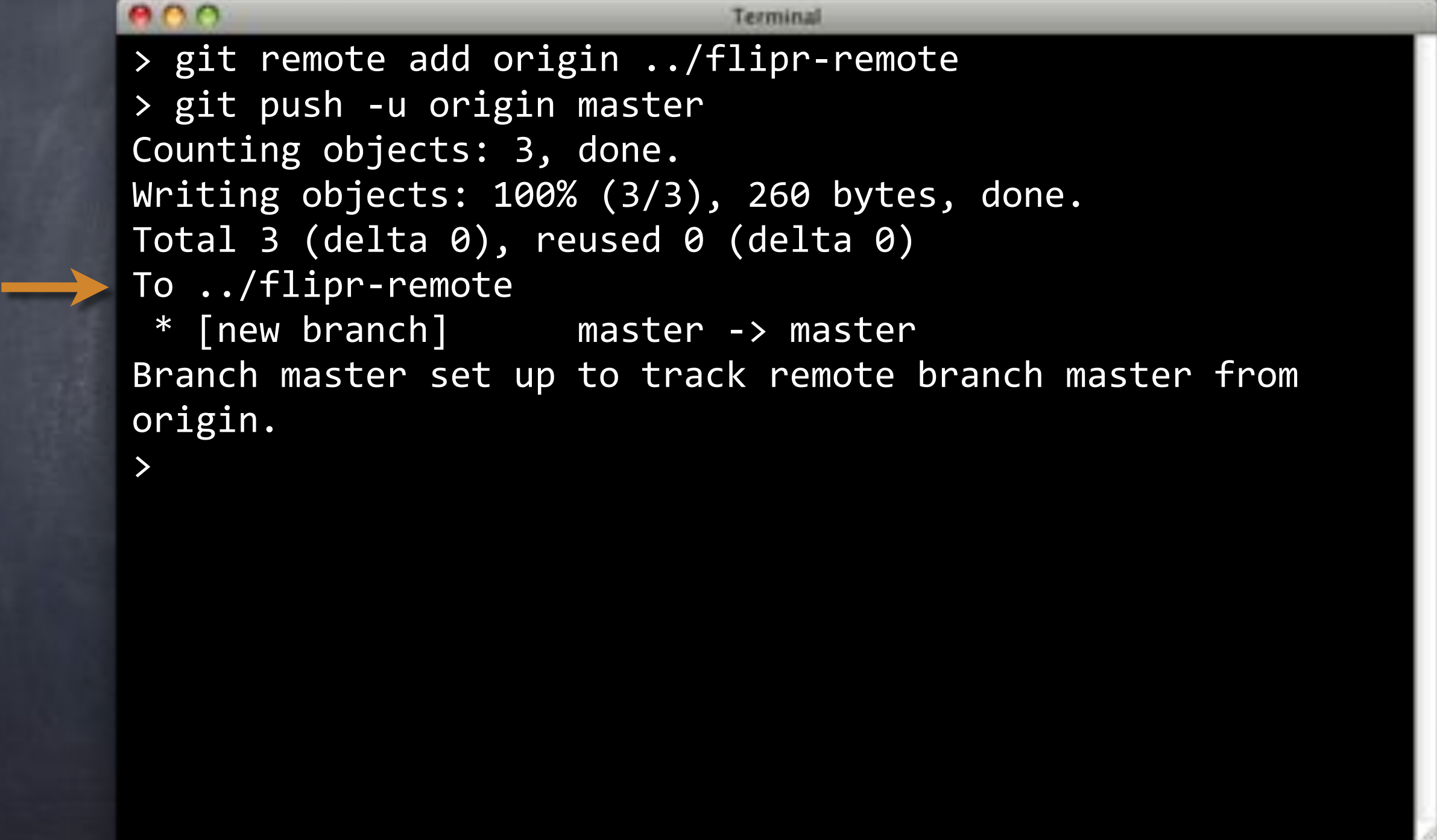
```
> git remote add origin ../flipr-remote
>
```

Or remote URL

# 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.
>
```

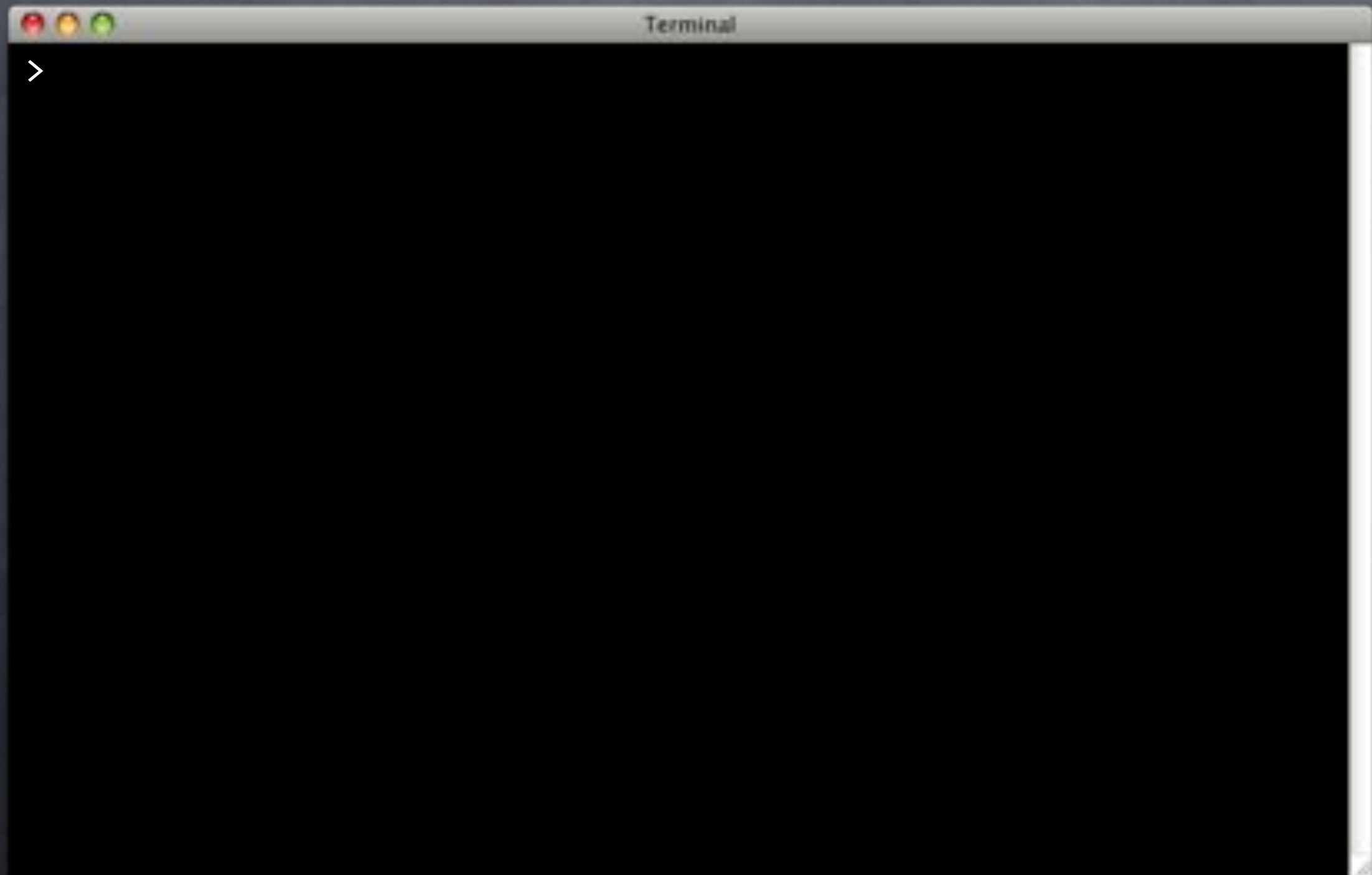


# Origin

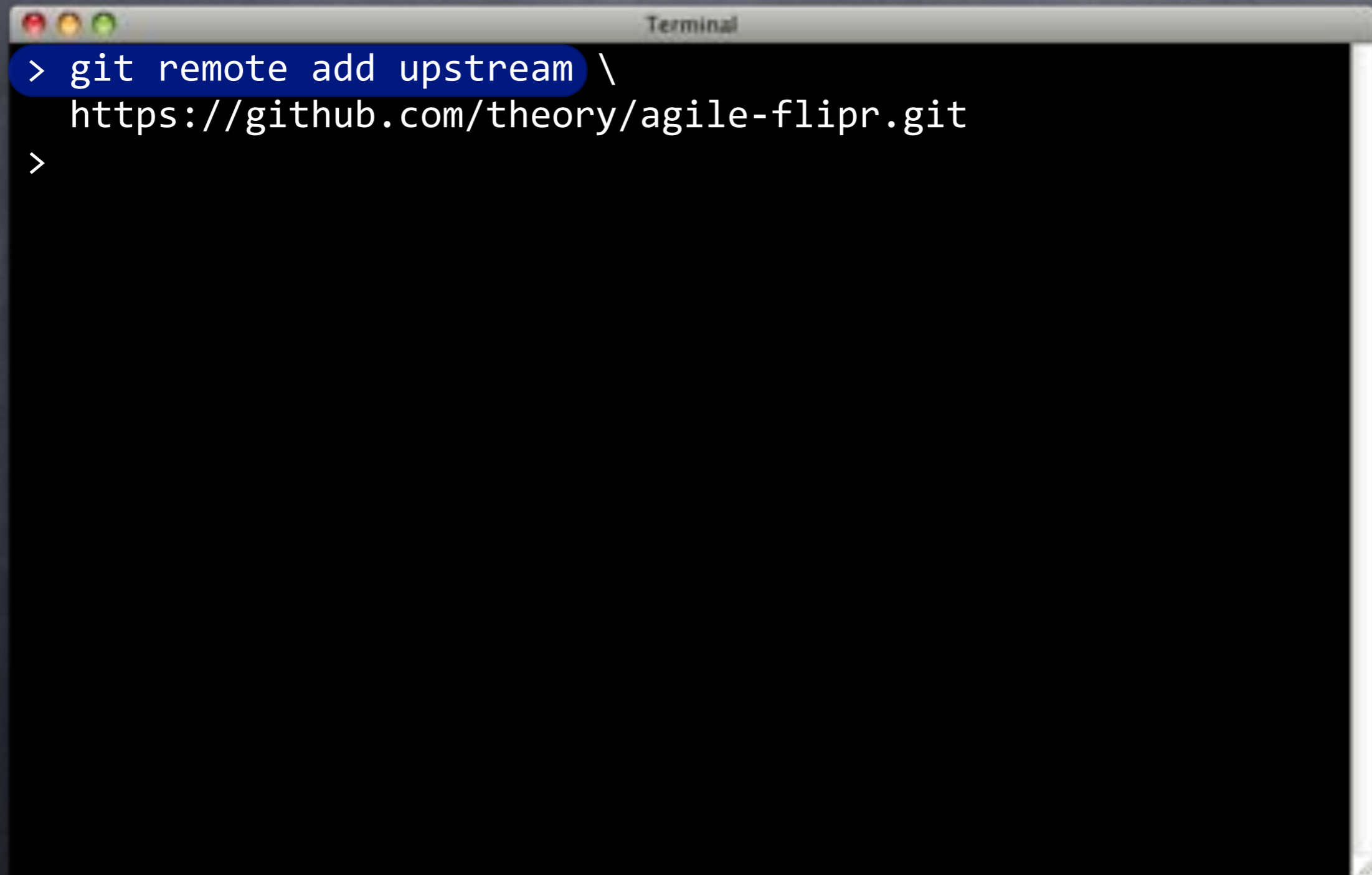
```
Terminal
> git remote add origin ../flipr-remote
> git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 260 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To ../flipr-remote
 * [new branch]      master -> master
Branch master set up to track remote branch master from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



# Swimming Upstream



# Swimming Upstream

A terminal window titled "Terminal" with a dark background and light text. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The text inside the terminal shows a command being entered: "> git remote add upstream \" followed by the URL "https://github.com/theory/agile-flipr.git" on the next line, and another ">" on the third line. The text "git remote add upstream \" is highlighted with a blue background.

```
> git remote add upstream \  
https://github.com/theory/agile-flipr.git  
>
```

# 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**



# PiSHA1



# 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

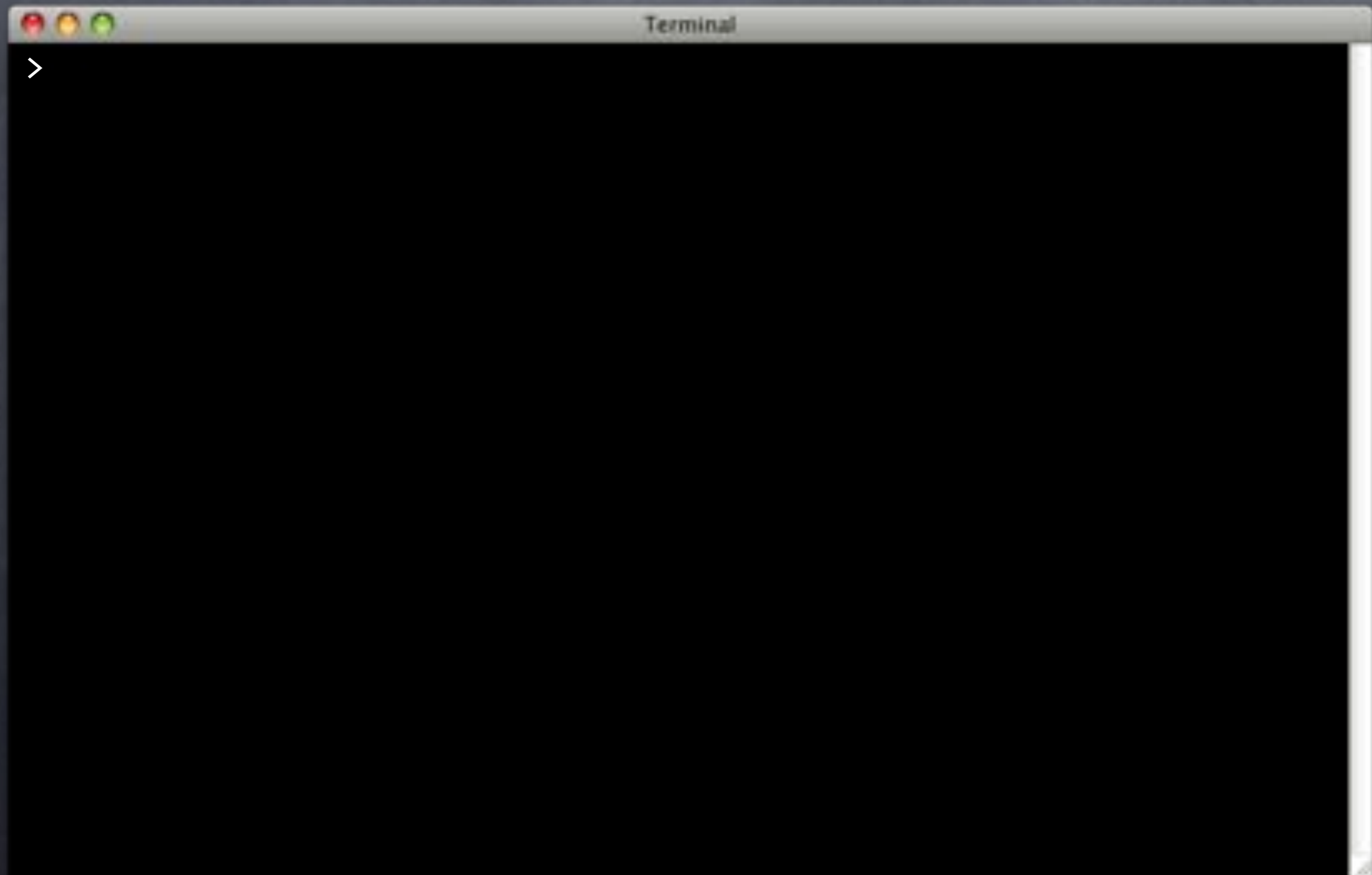


# PiSHA1

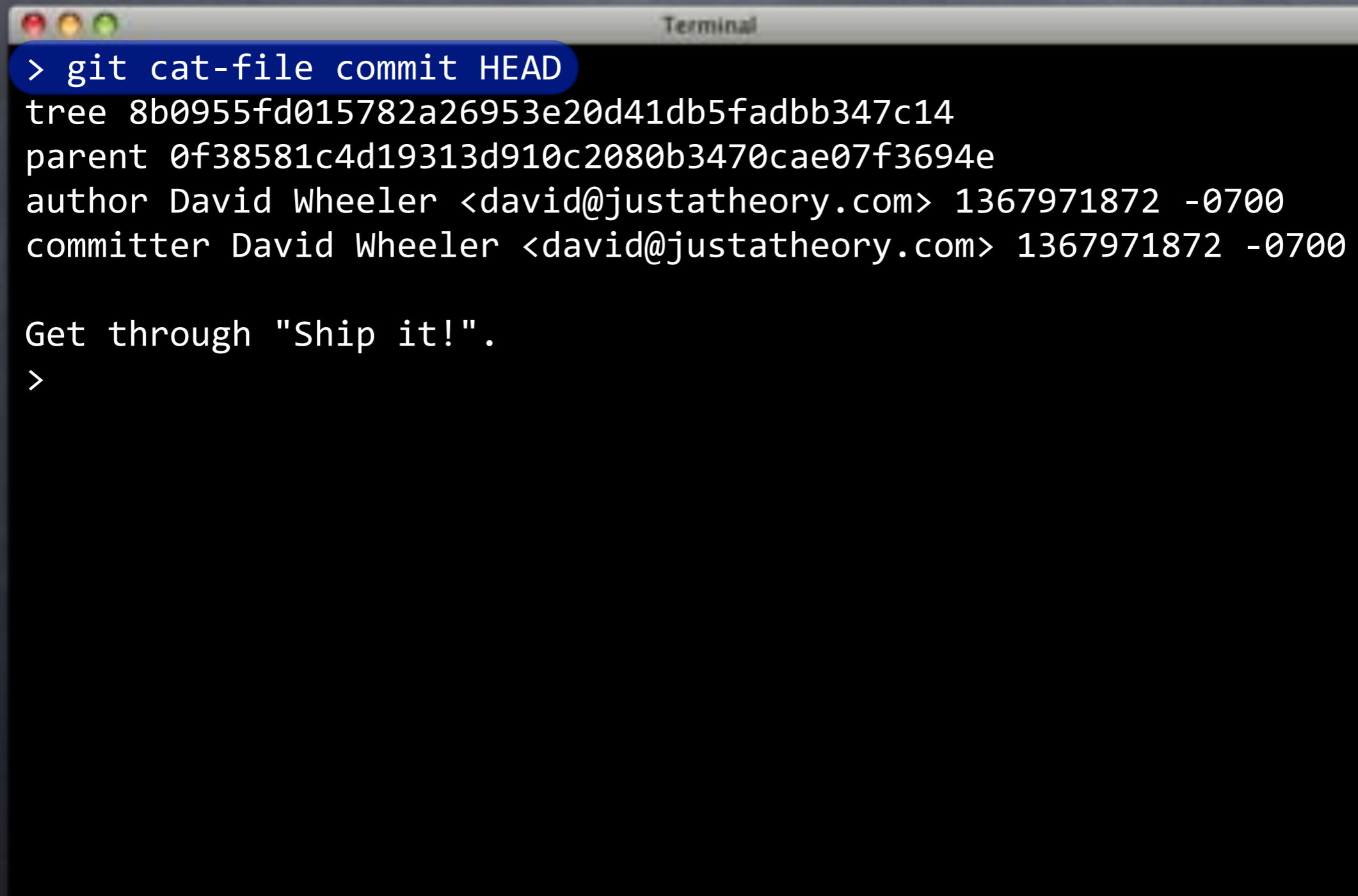
- SHA1 ID for every object
  - commit, tag, tree, blob
- Hashed commit text includes:
  - tree ID
  - parent ID
  - author
  - committer
  - message



# Making a hash of it



# Making a hash of it

A terminal window titled "Terminal" with a dark background and light text. The window shows the command `> git cat-file commit HEAD` highlighted in a blue bar. The output is: `tree 8b0955fd015782a26953e20d41db5fadbb347c14`, `parent 0f38581c4d19313d910c2080b3470cae07f3694e`, `author David Wheeler <david@justatheory.com> 1367971872 -0700`, and `committer David Wheeler <david@justatheory.com> 1367971872 -0700`. Below the output, the text "Get through 'Ship it!'" is displayed, followed by a prompt character `>`.

```
> 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



```
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
```

An orange arrow points to the first line of the terminal output.

Get through "Ship 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



```
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
```

An orange arrow points to the 'author' line in the terminal output.

Get through "Ship 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

```
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



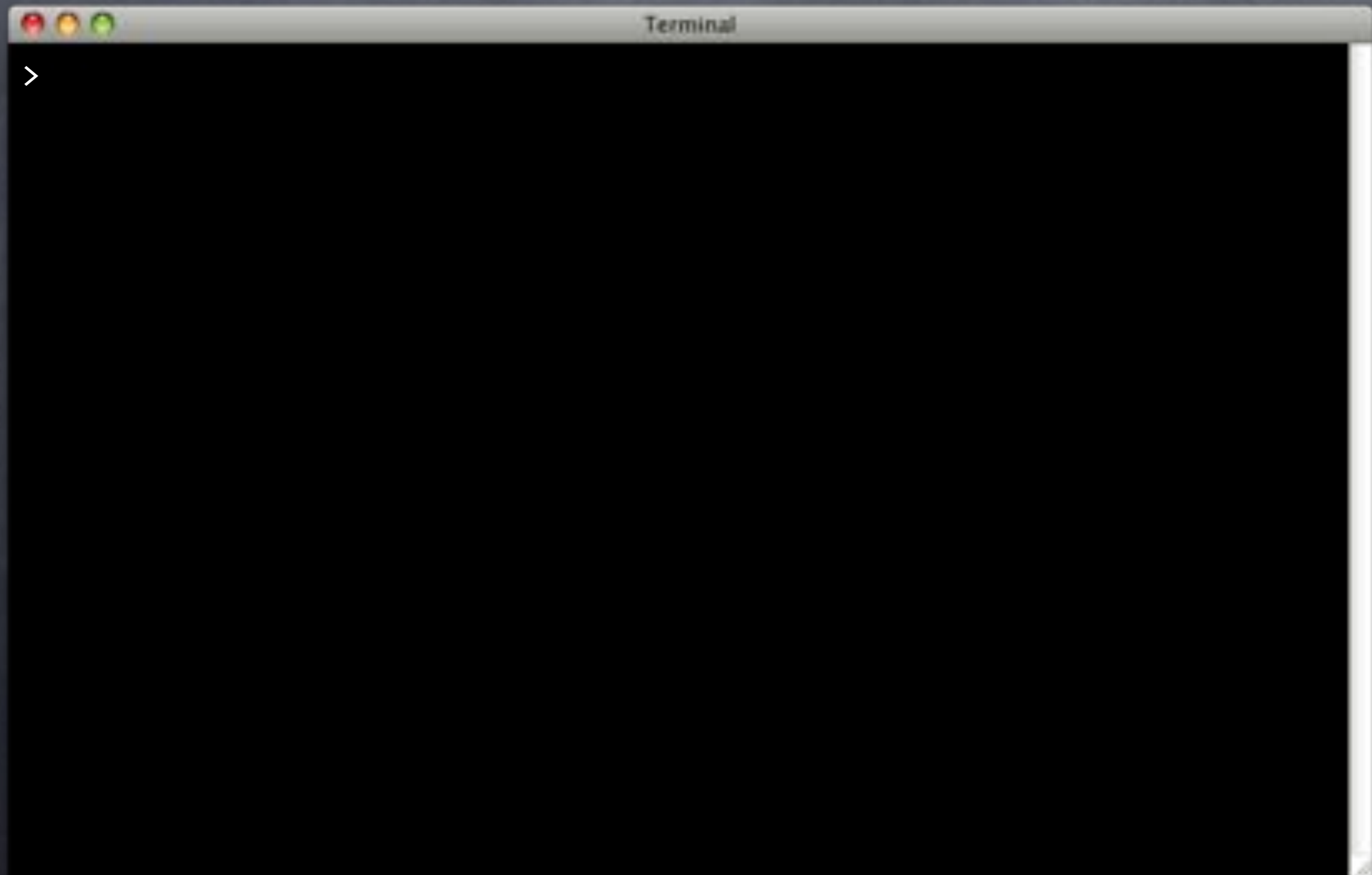
# Your Turn

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





# Who am I again?

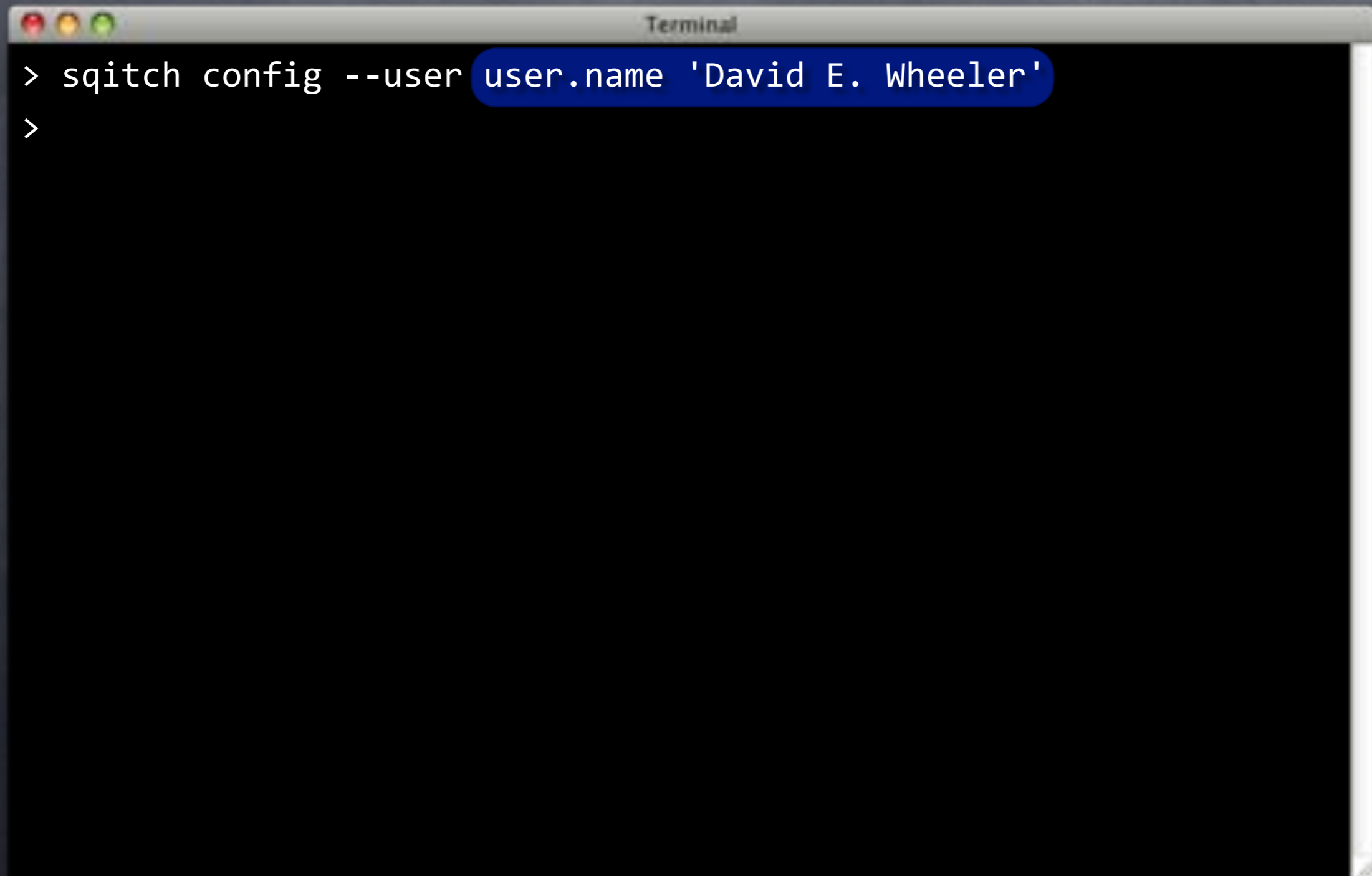


# Who am I again?

A terminal window titled "Terminal" with a dark background and a light gray title bar. The window contains two lines of text: the first line is a command prompt followed by the command "sqitch config --user user.name 'David E. Wheeler'", and the second line is a command prompt followed by a greater-than sign. The text "sqitch config --user" in the first line is highlighted with a green background.

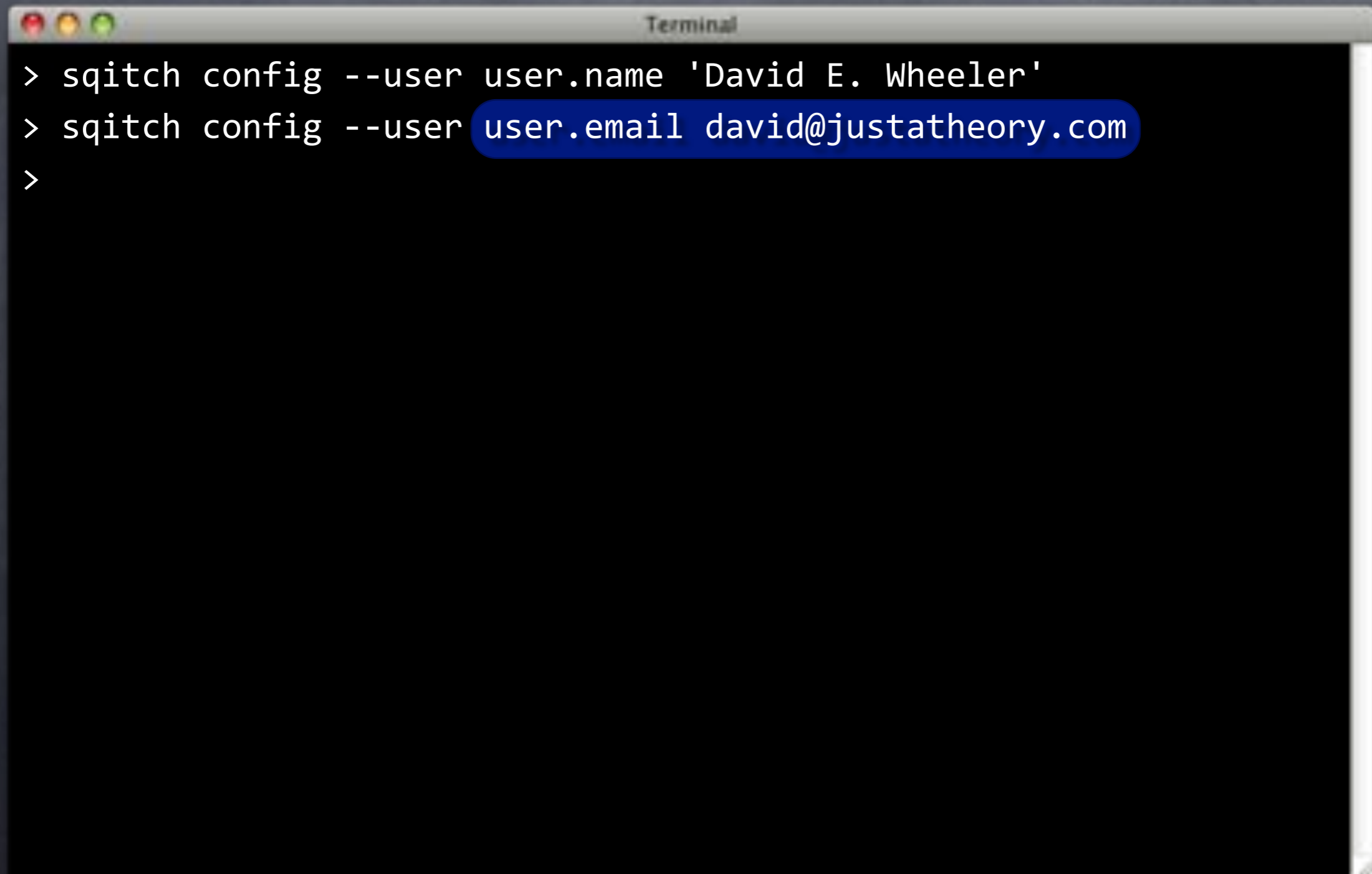
```
> sqitch config --user user.name 'David E. Wheeler'  
>
```

# Who am I again?

A terminal window titled "Terminal" with a dark background and a light gray title bar. The window contains two lines of text: the first line is a command prompt followed by the command "sqitch config --user user.name 'David E. Wheeler'", and the second line is a command prompt followed by a greater-than sign. The text "user.name 'David E. Wheeler'" is highlighted with a blue background.

```
> sqitch config --user user.name 'David E. Wheeler'  
>
```

# Who am I again?

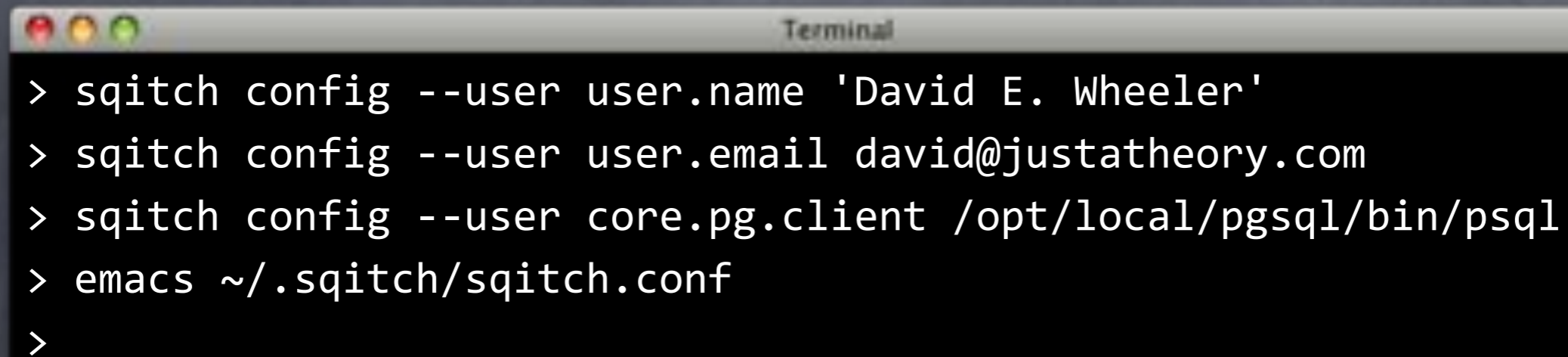
A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows three lines of shell commands. The second line, `user.email david@justatheory.com`, is highlighted with a blue background. The prompt character `>` is visible at the start of each line.

```
> sqitch config --user user.name 'David E. Wheeler'  
> sqitch config --user user.email david@justatheory.com  
>
```

# 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
>
```

# Who am I again?

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal contains a series of shell commands for configuring sqitch. The commands are: 1. sqitch config --user user.name 'David E. Wheeler' 2. sqitch config --user user.email david@justatheory.com 3. sqitch config --user core.pg.client /opt/local/pgsql/bin/psql 4. emacs ~/.sqitch/sqitch.conf 5. A blank line with a prompt character >.

```
> 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



```
[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

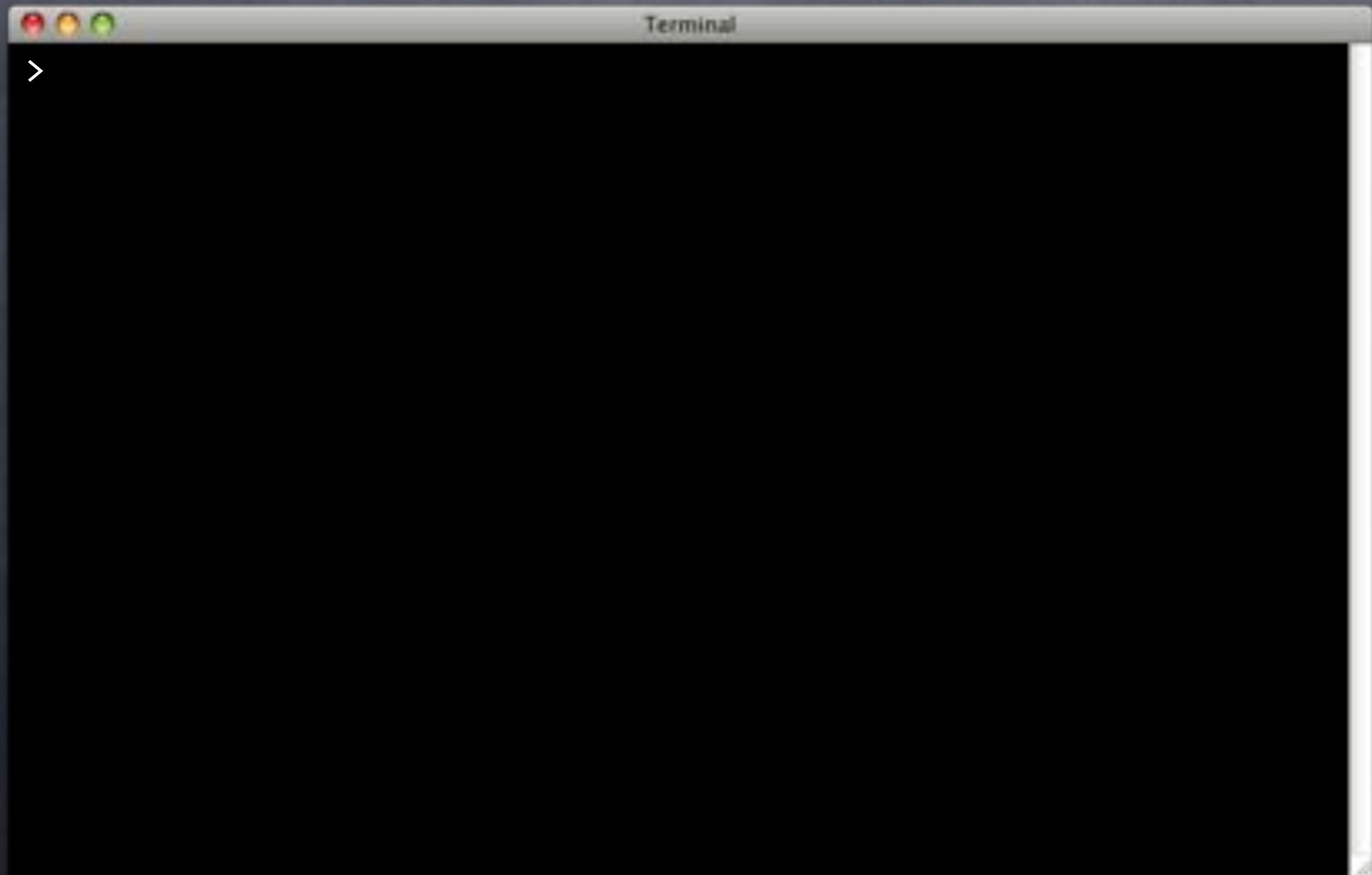
```
[core "pg"]
  client = /opt/local/pgsql/bin/psql
[user]
  name = David E. Wheeler
  email = david@justatheory.com
```

## Good for all projects

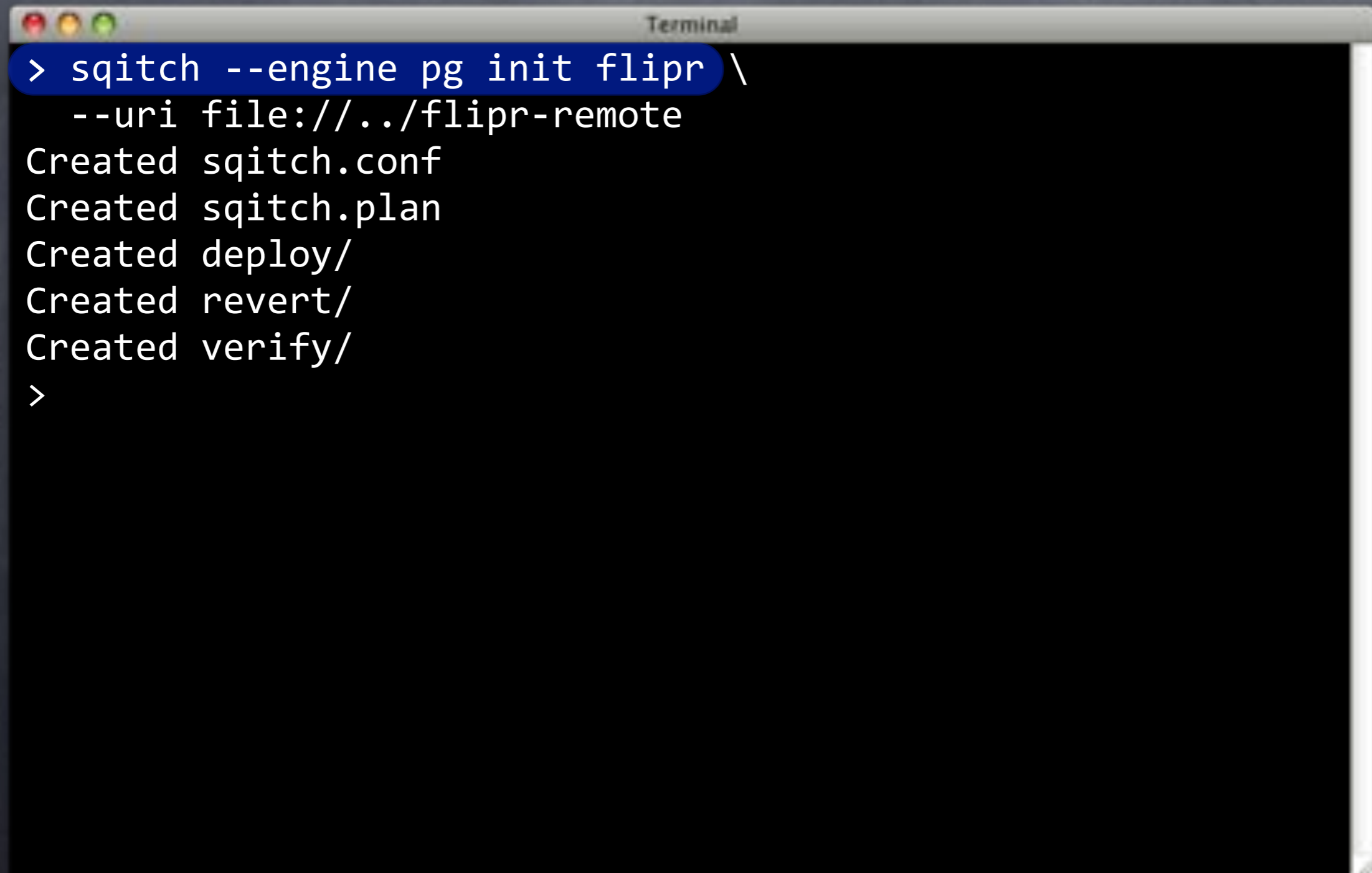
~/.sqitch/sqitc All (SQL[ansi])



# Scratch that Sqitch

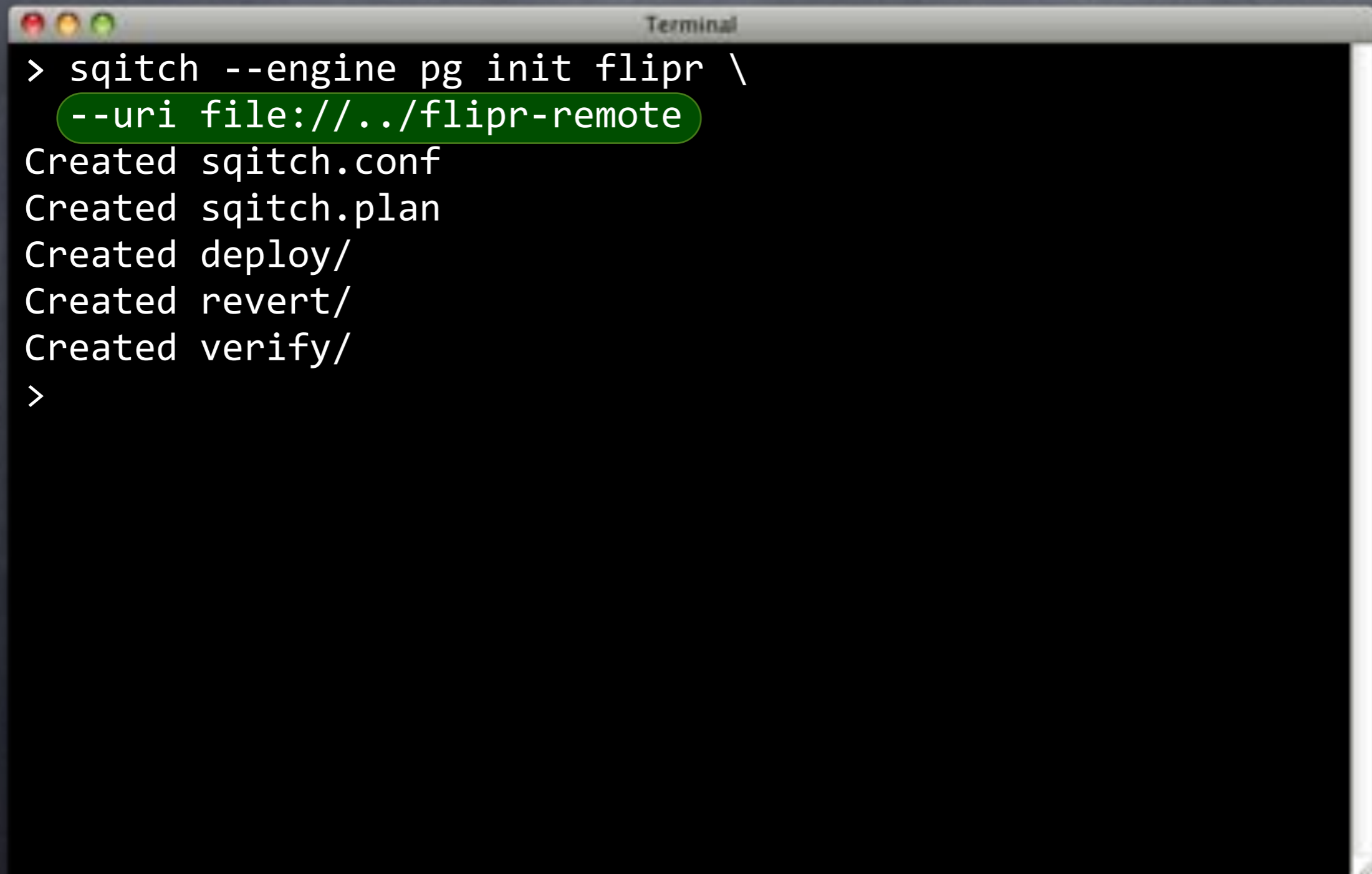


# Scratch that Sqitch

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows the execution of the sqitch command to initialize a project named "flipr". The command is highlighted in a blue box. The output shows the creation of several files and directories.

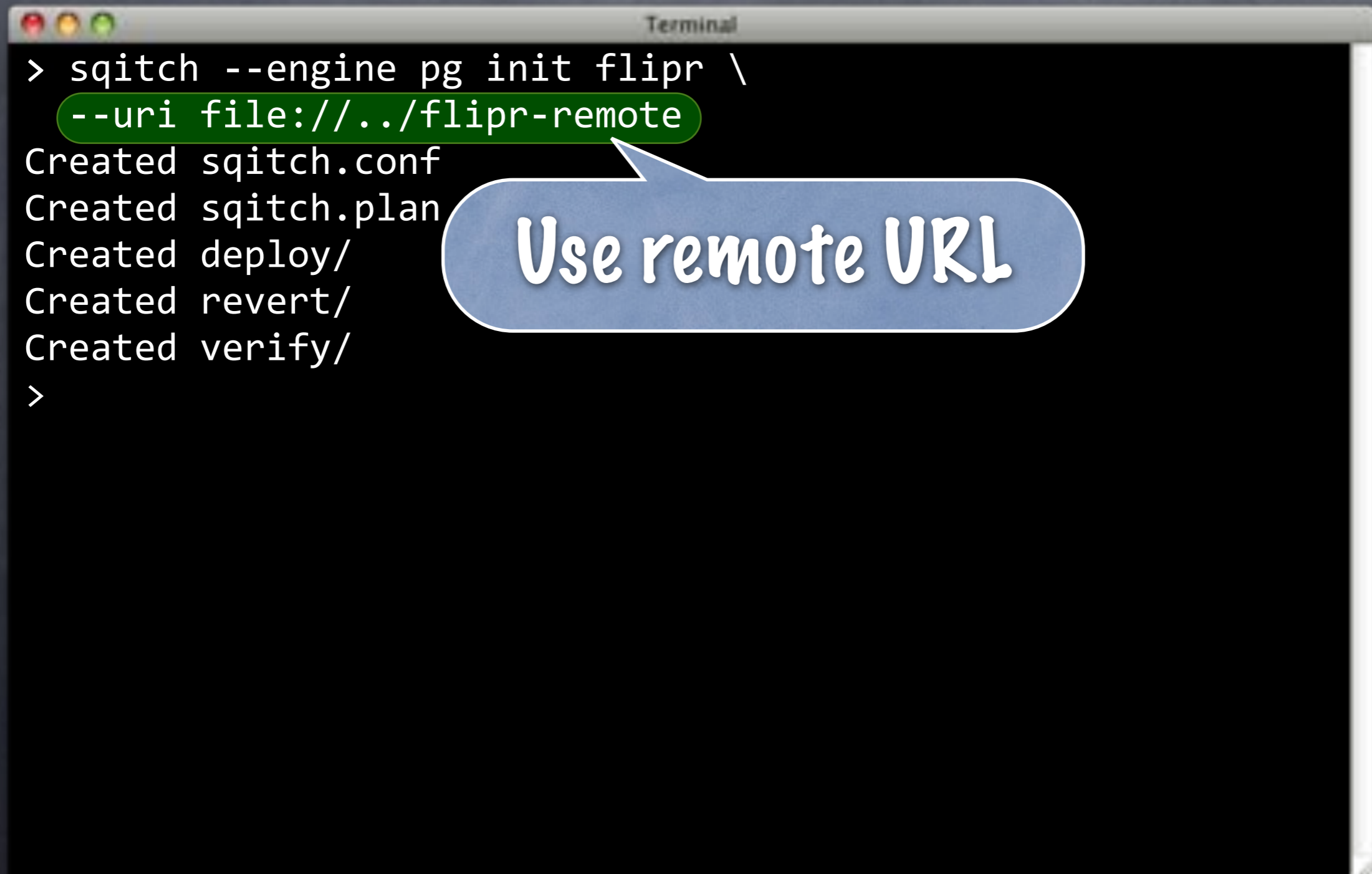
```
> sqitch --engine pg init flipr \  
  --uri file:///../flipr-remote  
Created sqitch.conf  
Created sqitch.plan  
Created deploy/  
Created revert/  
Created verify/  
>
```

# Scratch that Sqitch

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows a command being executed: `> sqitch --engine pg init flipr \` followed by `--uri file:///../flipr-remote` which is highlighted in green. The output shows the creation of several files and directories: `Created sqitch.conf`, `Created sqitch.plan`, `Created deploy/`, `Created revert/`, and `Created verify/`. The prompt `>` is shown at the end of the output.

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

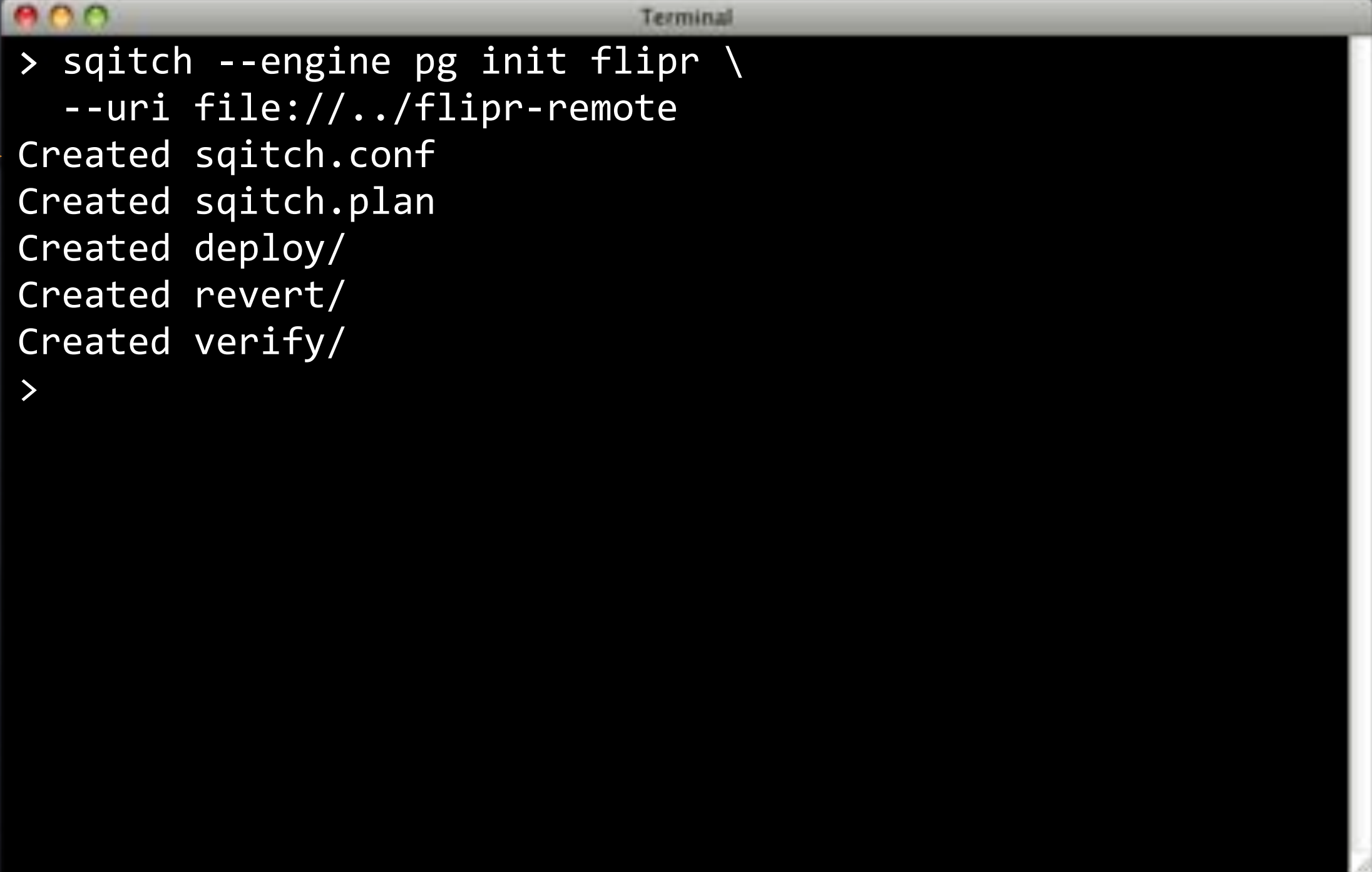
# Scratch that Sqitch



```
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



```
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/  
>
```



# Scratch that Sqitch

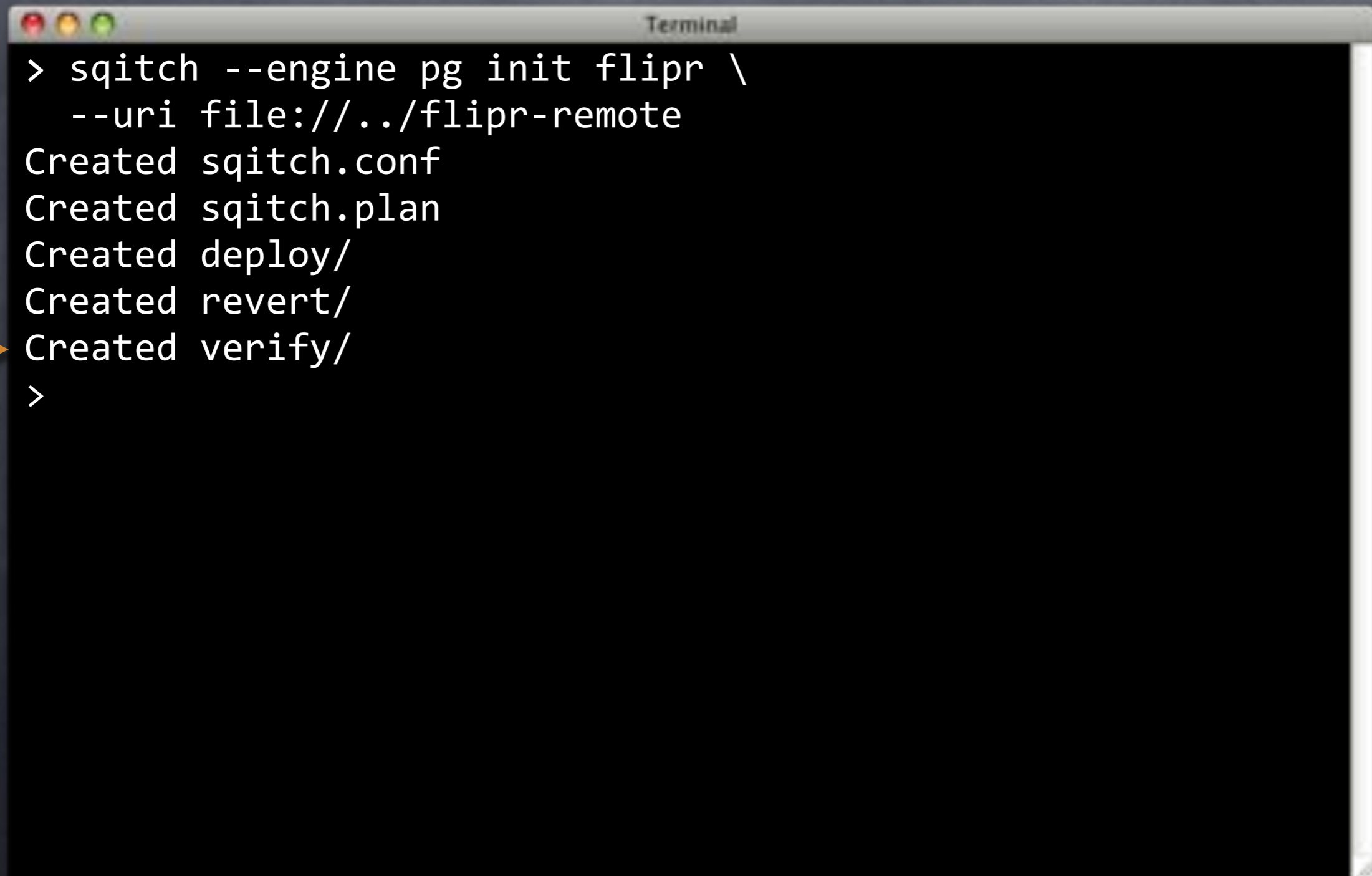
```
Terminal  
> sqitch --engine pg init flipr \  
  --uri file:///../flipr-remote  
Created sqitch.conf  
Created sqitch.plan  
→ Created deploy/  
Created revert/  
Created verify/  
>
```

# 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/
>
```

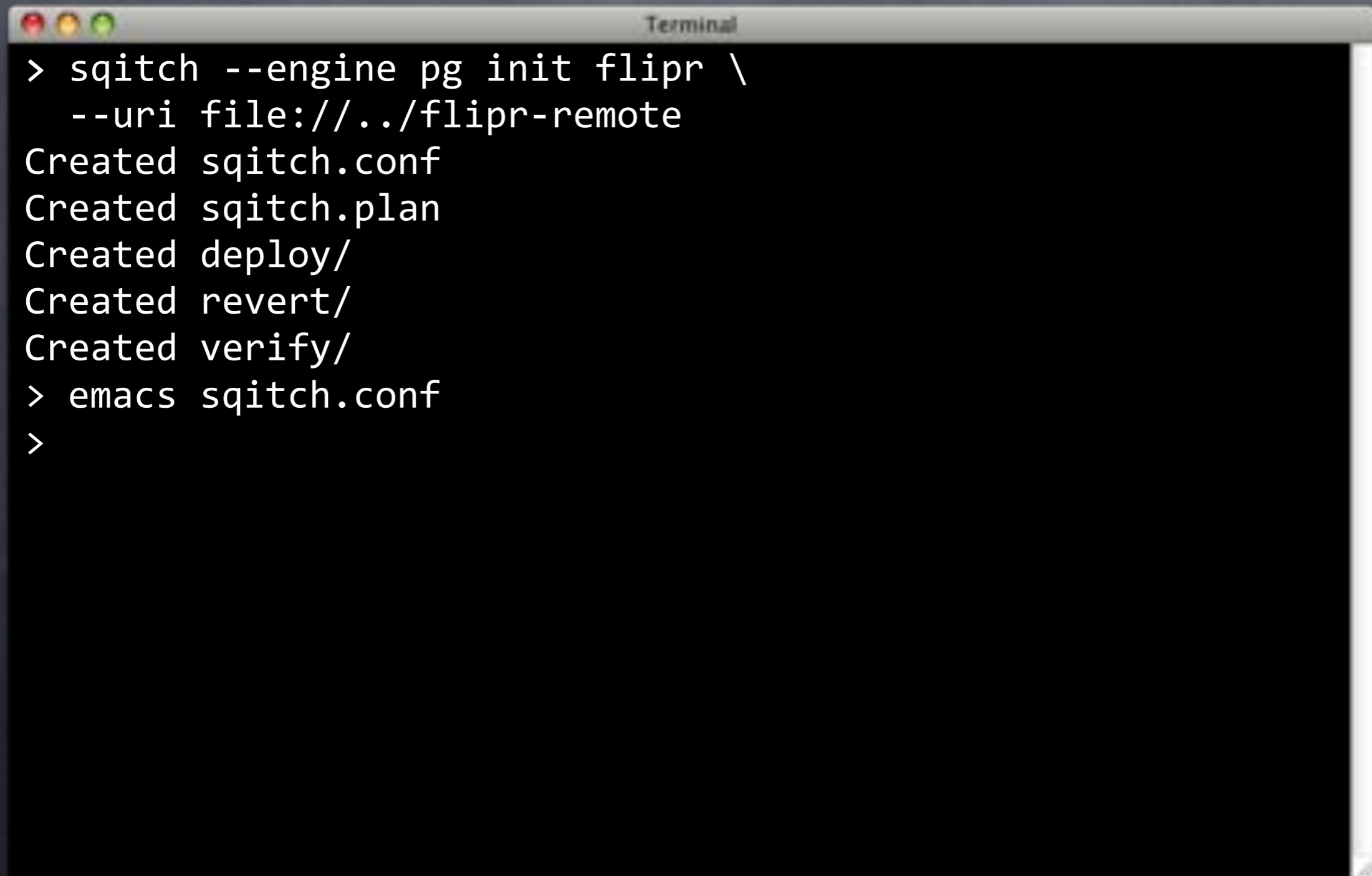


# Scratch that Sqitch

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows the execution of the 'sqitch' command to initialize a project named 'flipr'. The output lists several files and directories created: sqitch.conf, sqitch.plan, deploy/, revert/, and verify/. An orange arrow points to the 'Created verify/' line.

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

# Scratch that Sqitch

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal displays the following text:

```
> 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

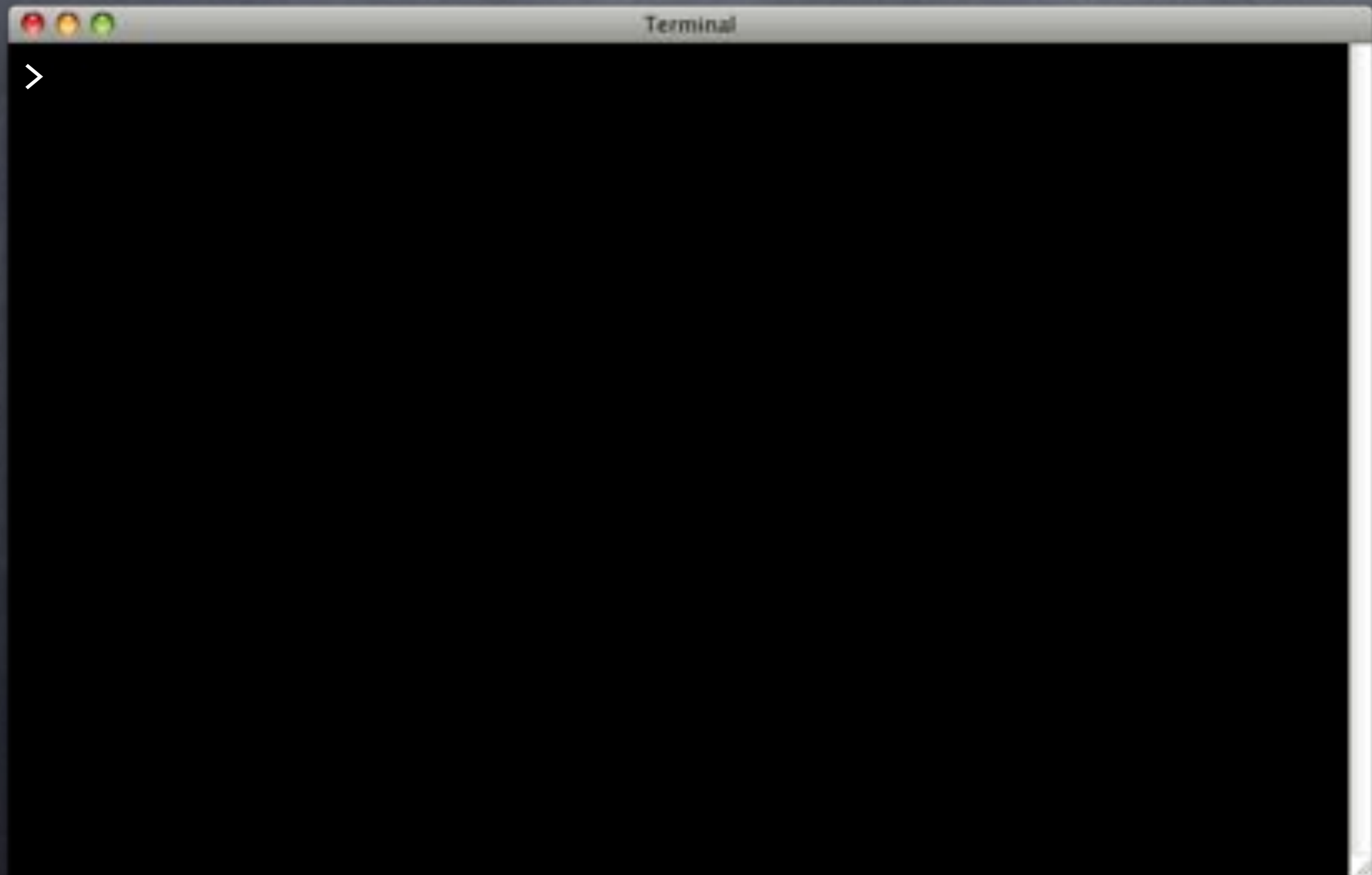
```
[core]
  engine = pg
  # plan_file = sqitch.plan
  # top_dir = .
  # deploy_dir = deploy
  # revert_dir = revert
  # verify_dir = verify
  # extension = sql
# [core "pg"]
  # client = /usr/local/pgsql/bin/psql
  # username =
  # password =
  # db_name =
  # host =
  # port =
  # sqitch_schema = sqitch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

# sqitch.conf

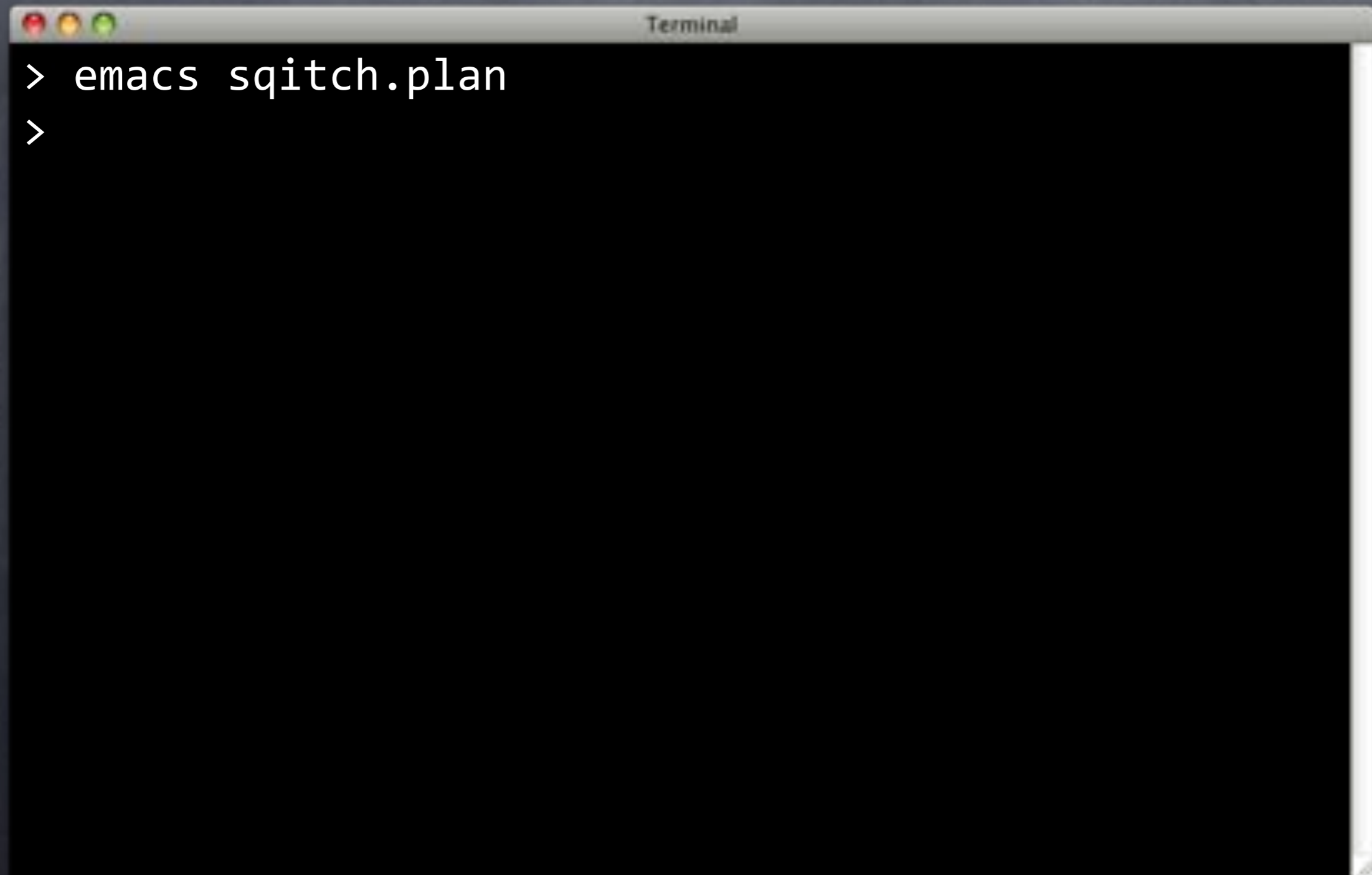
```
[core]
  engine = pg
  # plan_file =
  # top_dir = .
  # deploy_dir = deploy
  # revert_dir = revert
  # verify_dir = verify
  # extension = sql
# [core "pg"]
  # client = /usr/local/pgsql/bin/psql
  # username =
  # password =
  # db_name =
  # host =
  # port =
  # sqitch_schema = sqitch
```

--engine pg

# What's the Plan Man?

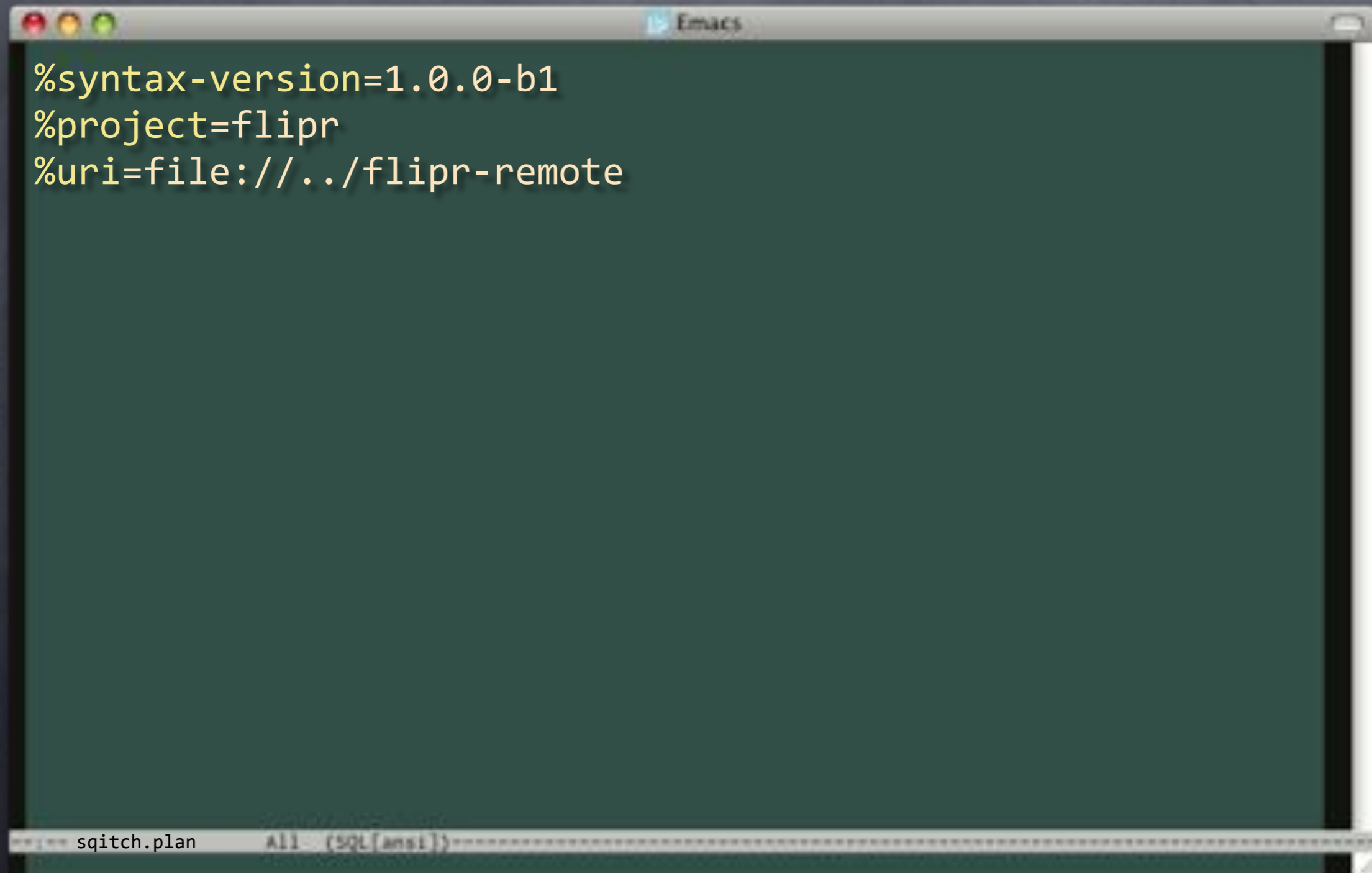


# What's the Plan Man?

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a prompt character followed by the command "emacs sqitch.plan" and another prompt character on the next line.

```
> emacs sqitch.plan  
>
```

# sqitch.plan



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

sqitch.plan All (SQL[ansi])

# sqitch.plan

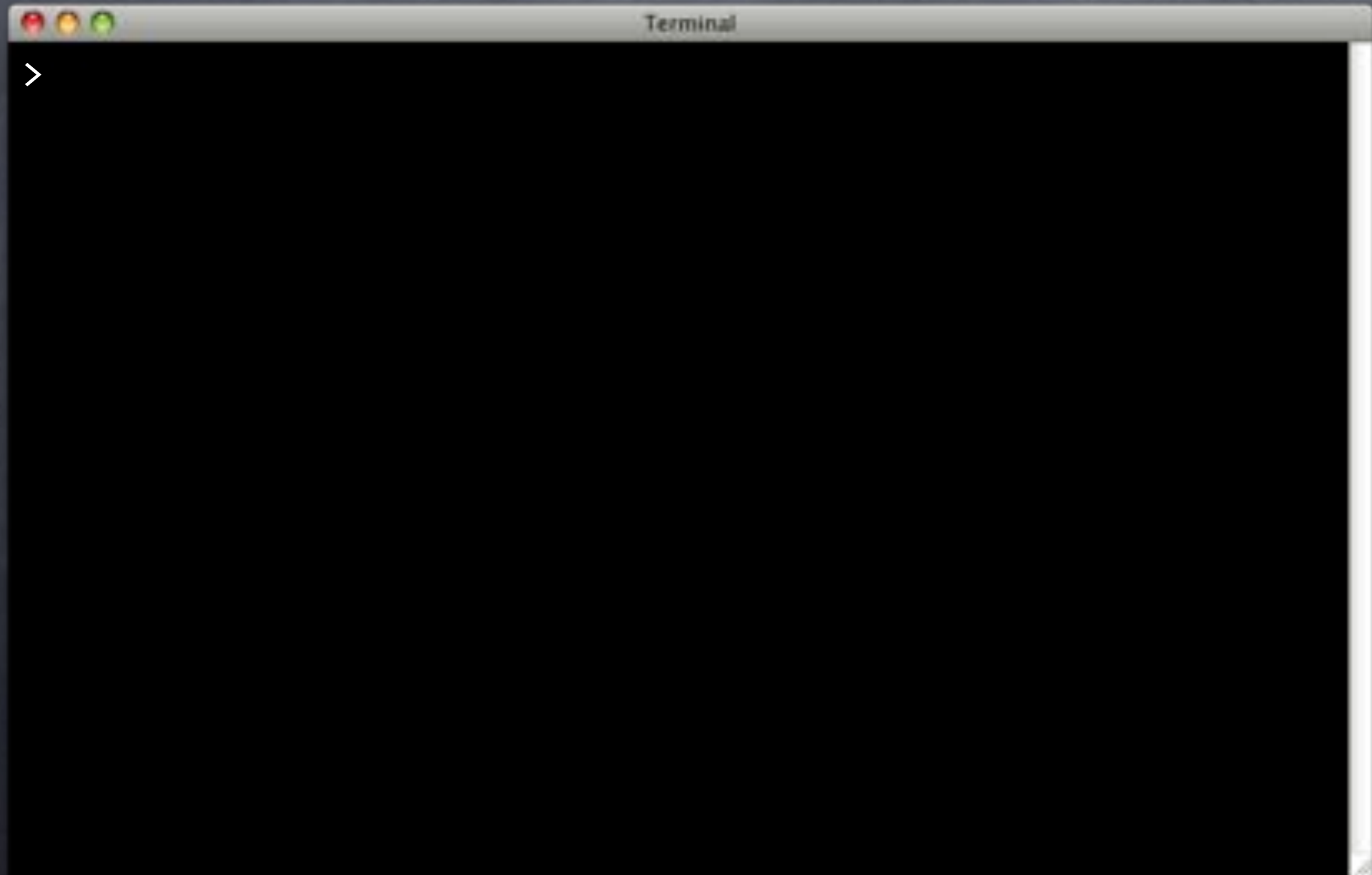
```
%syntax-version=1.0.0-b1
%project=flipr
%uri=file:///../flipr-remote
```

Identified

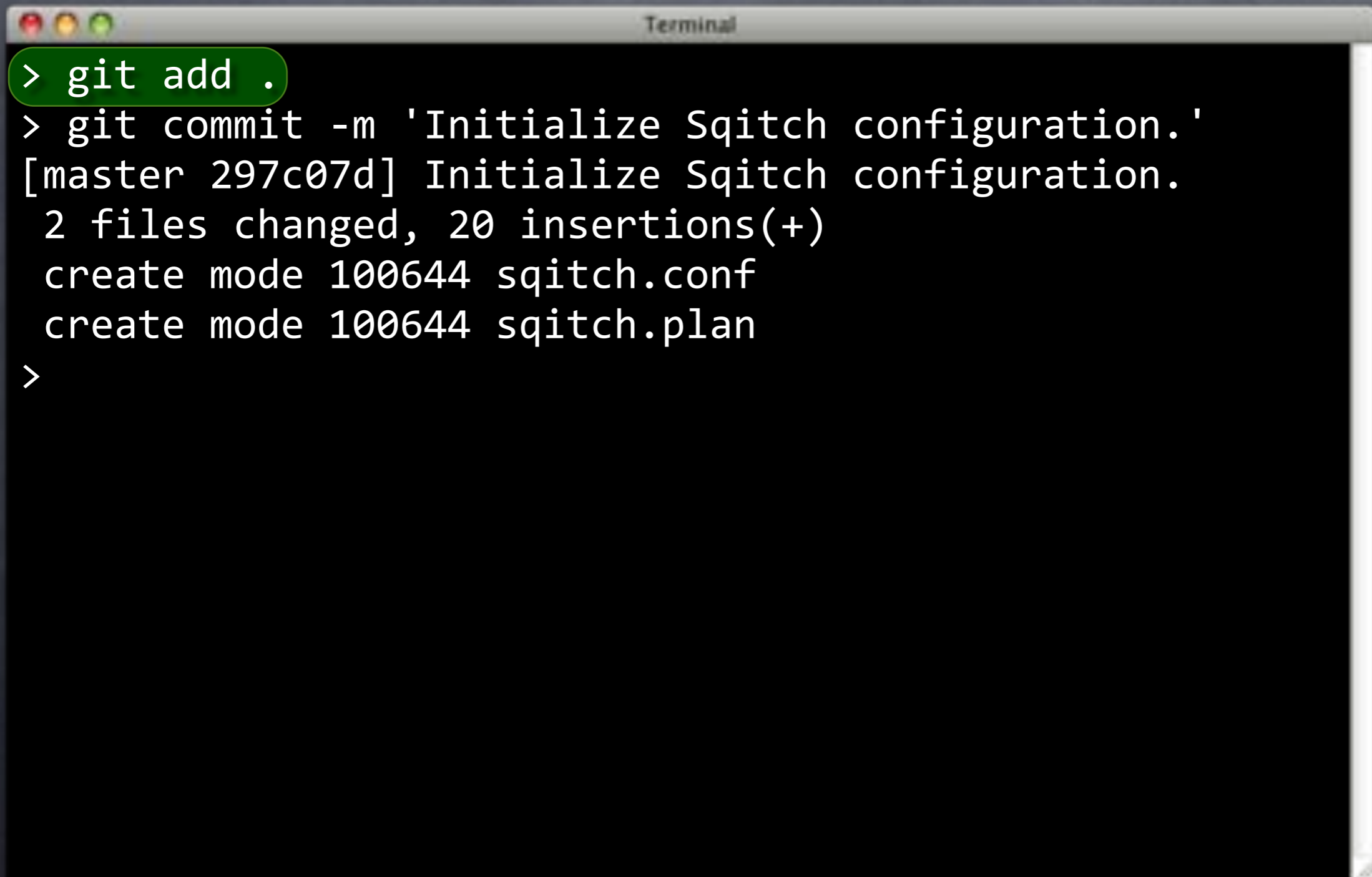
sqitch.plan All (SQL[ansi])



# Make It So

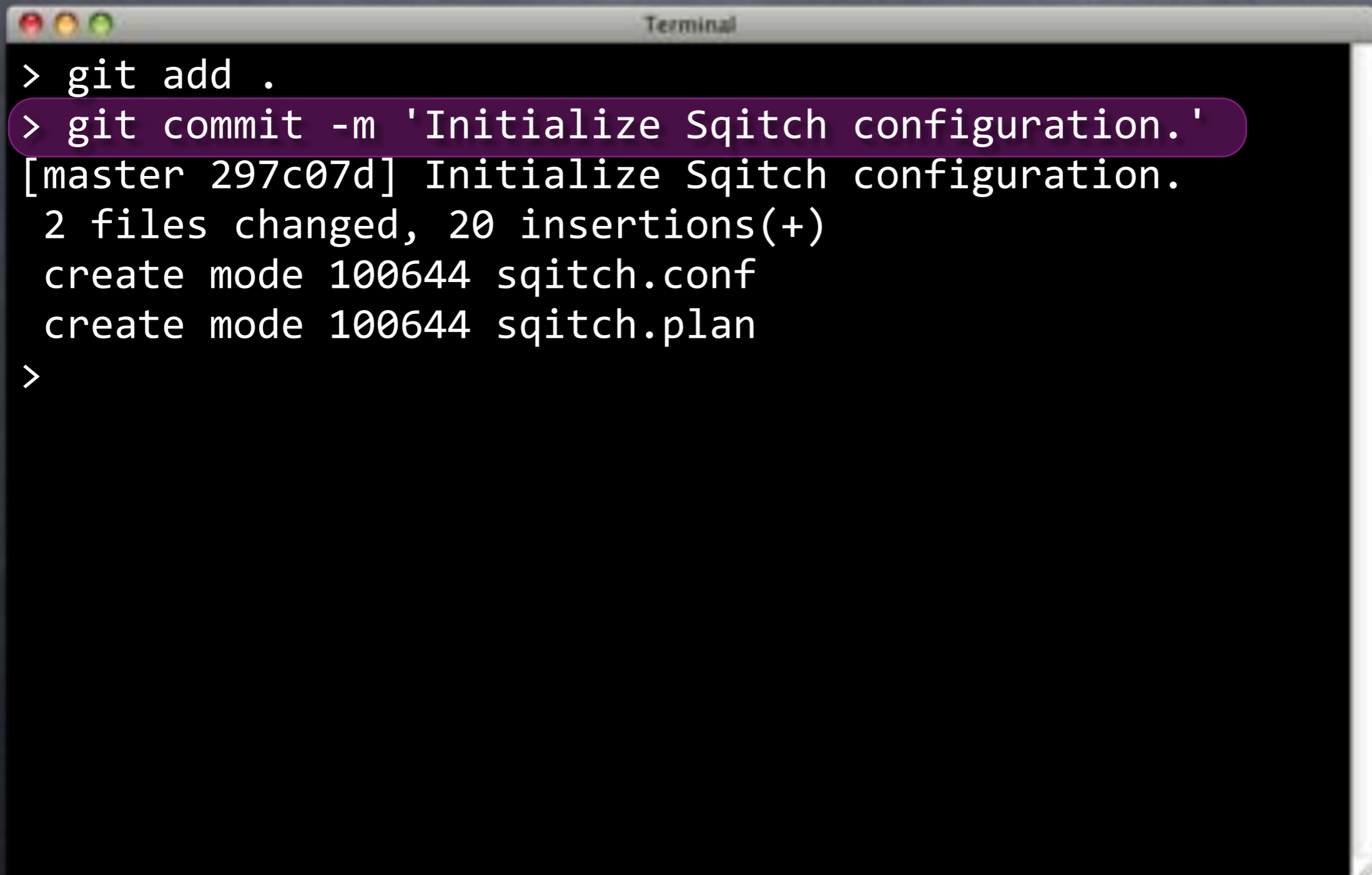


# Make It So

A terminal window titled "Terminal" with a dark background and light text. The first command, "> git add .", is highlighted with a green background. The subsequent command is "> git commit -m 'Initialize Sqitch configuration.'". The output shows the commit message "[master 297c07d] Initialize Sqitch configuration.", followed by "2 files changed, 20 insertions(+)", and the creation of two files: "create mode 100644 sqitch.conf" and "create mode 100644 sqitch.plan". A final prompt ">" is visible at the bottom.

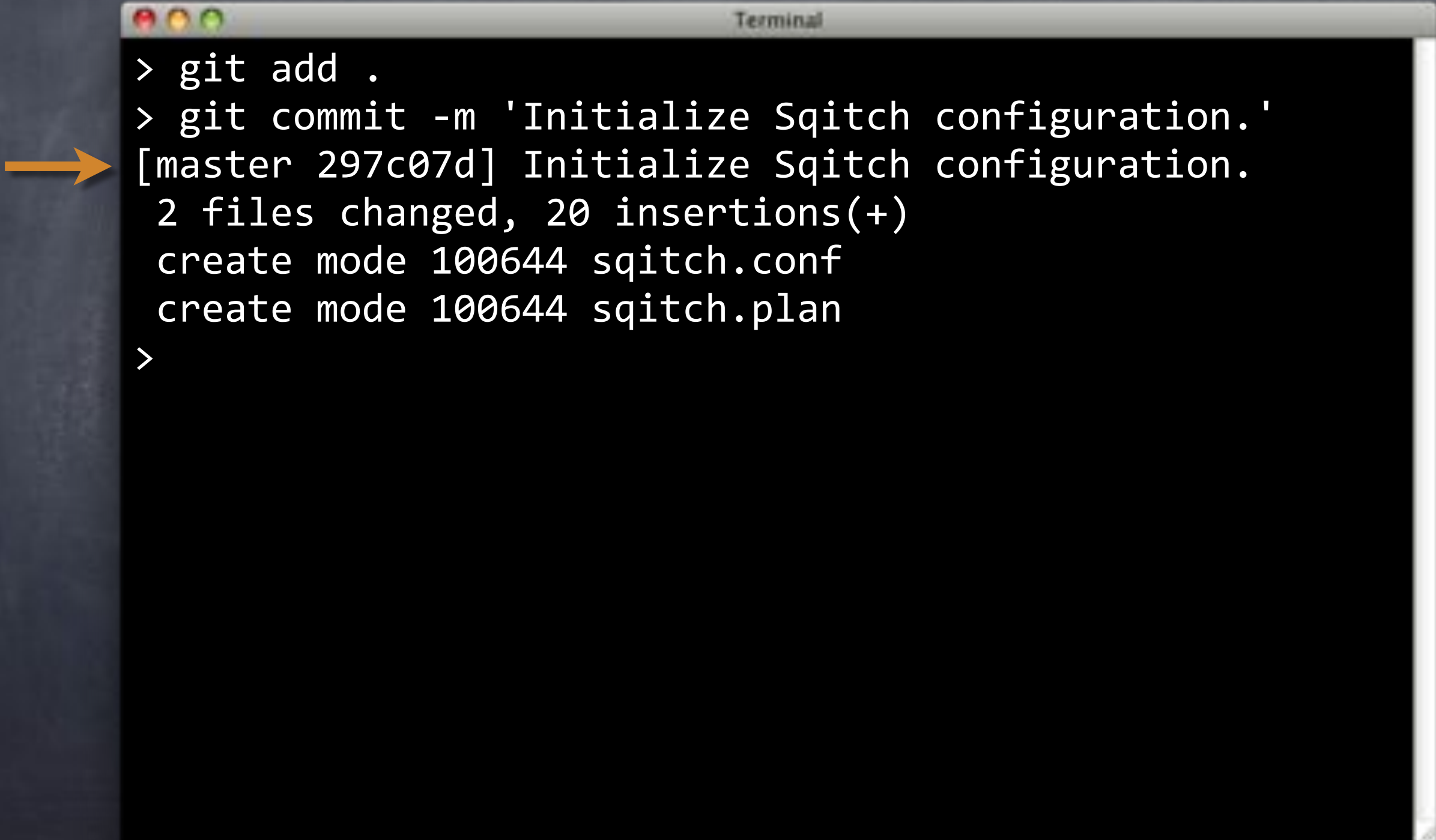
```
> git add .
> git commit -m 'Initialize Sqitch configuration.'
[master 297c07d] Initialize Sqitch configuration.
 2 files changed, 20 insertions(+)
 create mode 100644 sqitch.conf
 create mode 100644 sqitch.plan
>
```

# Make It So

A terminal window titled "Terminal" with a dark background and light text. The window shows a sequence of git commands and their output. The first command is "git add ." followed by "git commit -m 'Initialize Sqitch configuration.'" which is highlighted with a purple background. The output shows the commit message, the current branch and hash, and details about the files added.

```
> 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

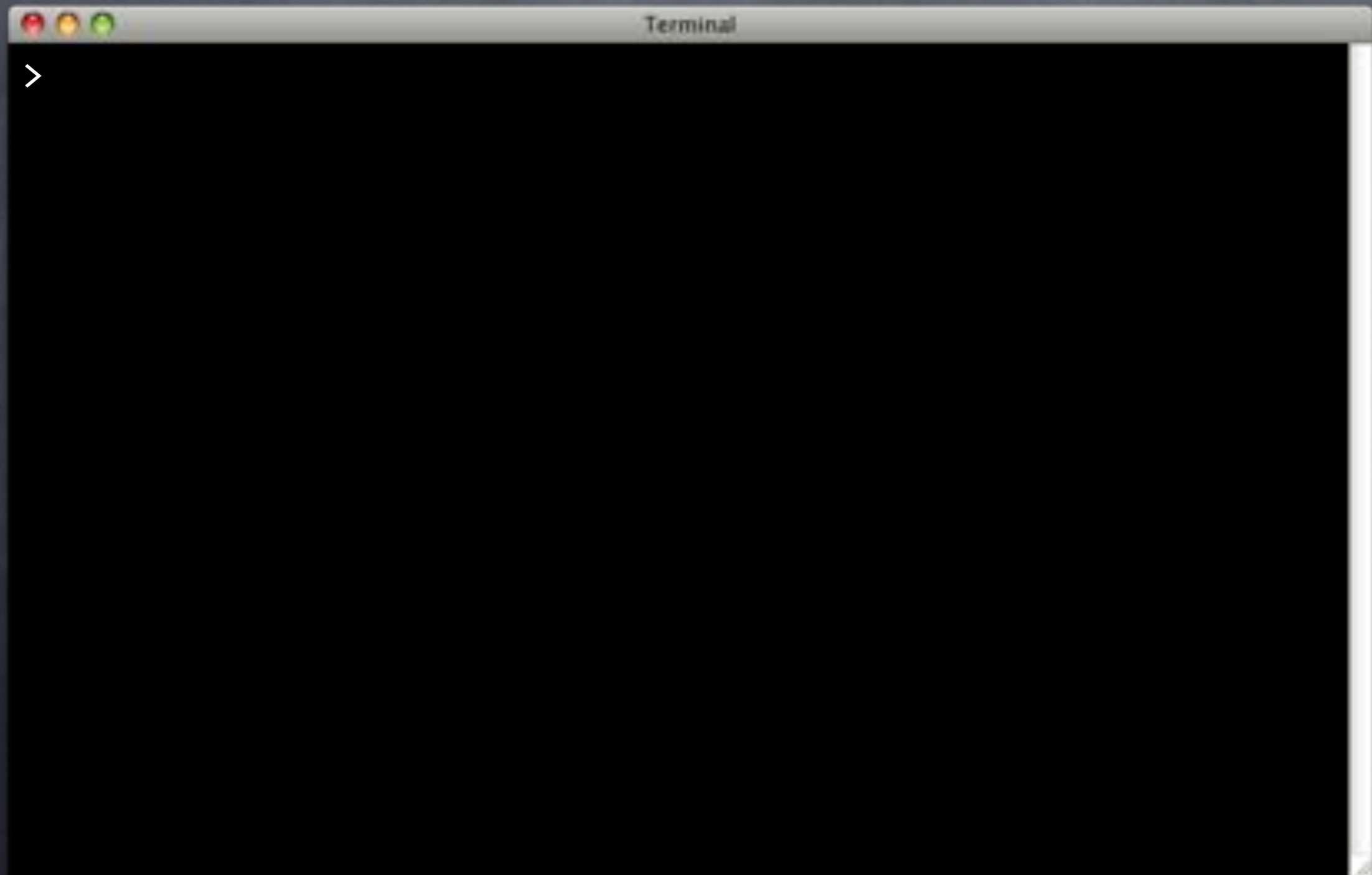
```
Terminal
> git add .
> git commit -m 'Initialize Sqitch configuration.'
[master 297c07d] Initialize Sqitch configuration.
 2 files changed, 20 insertions(+)
 create mode 100644 sqitch.conf
 create mode 100644 sqitch.plan
> git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
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?

```
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?

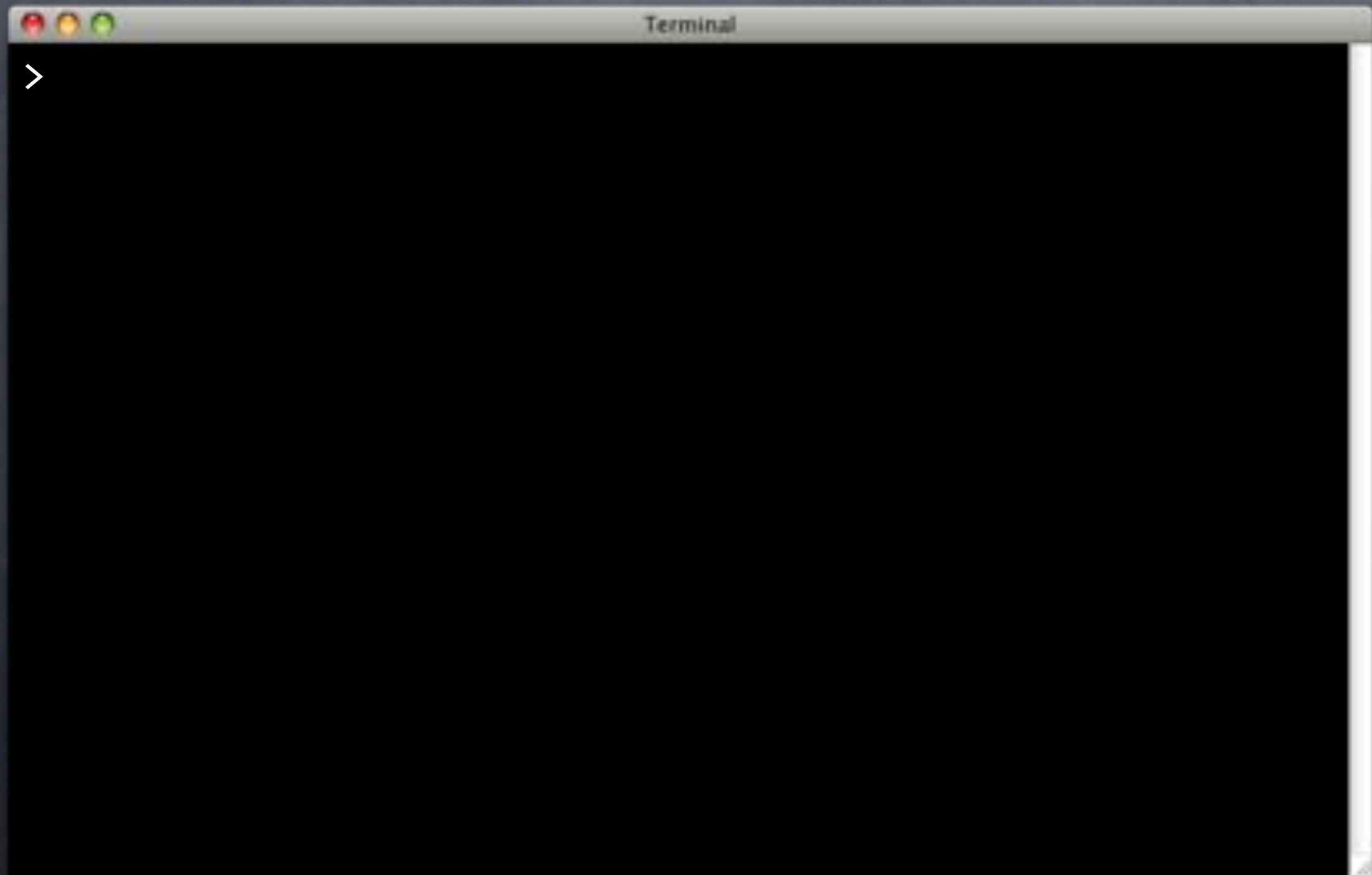
```
Terminal
> git log
commit a4f765f88c1875ffe9427e73f72a6b66ce363ed4
Author: David E. Wheeler <david@justatheory.com>
Date: Tue May 21 15:00:33 2013 -0400

    Initialize Sqitch configuration.

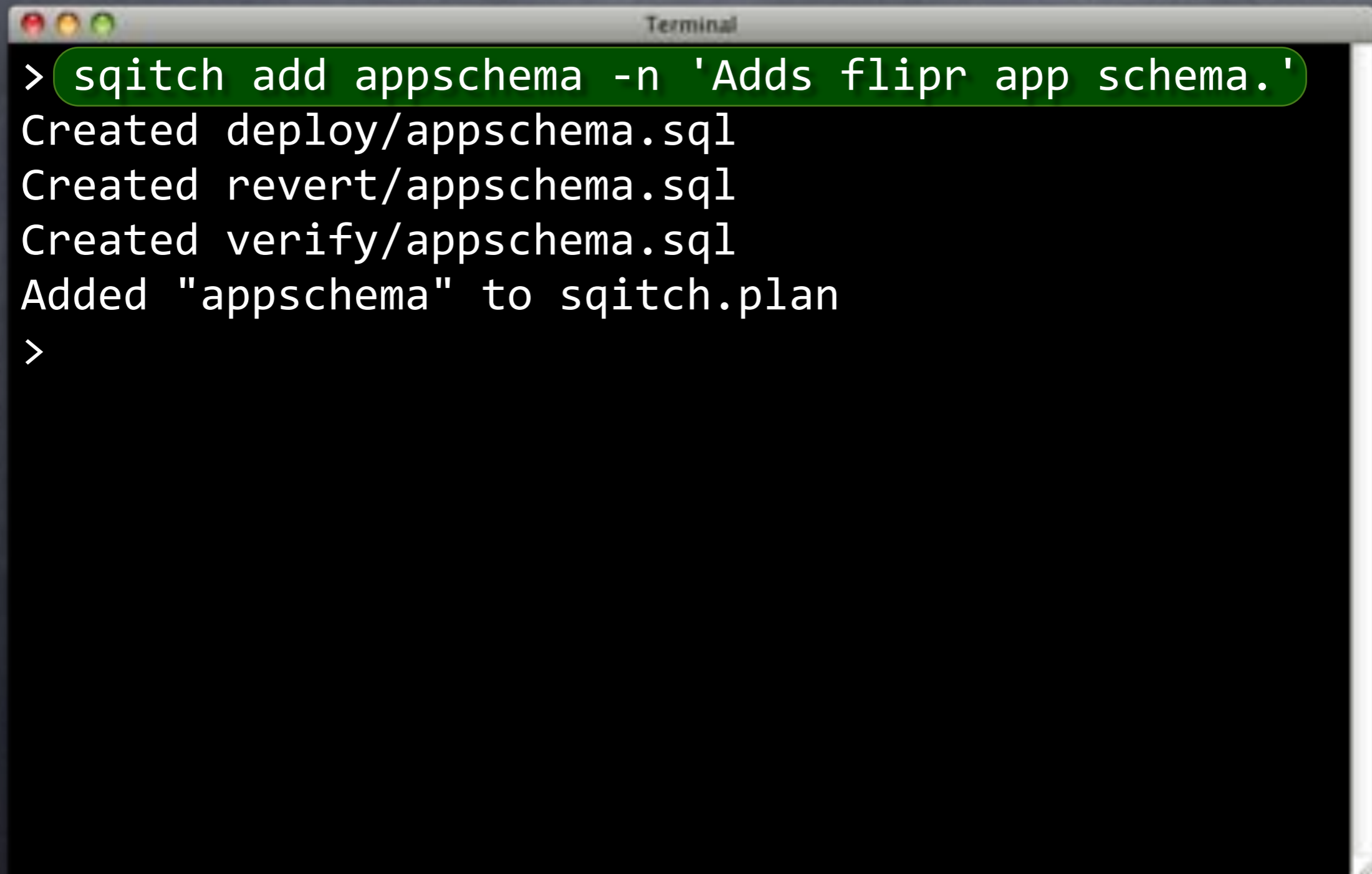
→ commit 2c4b51017929634a30f4a74e20268c05effdfcb3
Author: David E. Wheeler <david@justatheory.com>
Date: Tue May 21 14:59:16 2013 -0400

    Initialize repo, add README.
>
```

# First Change

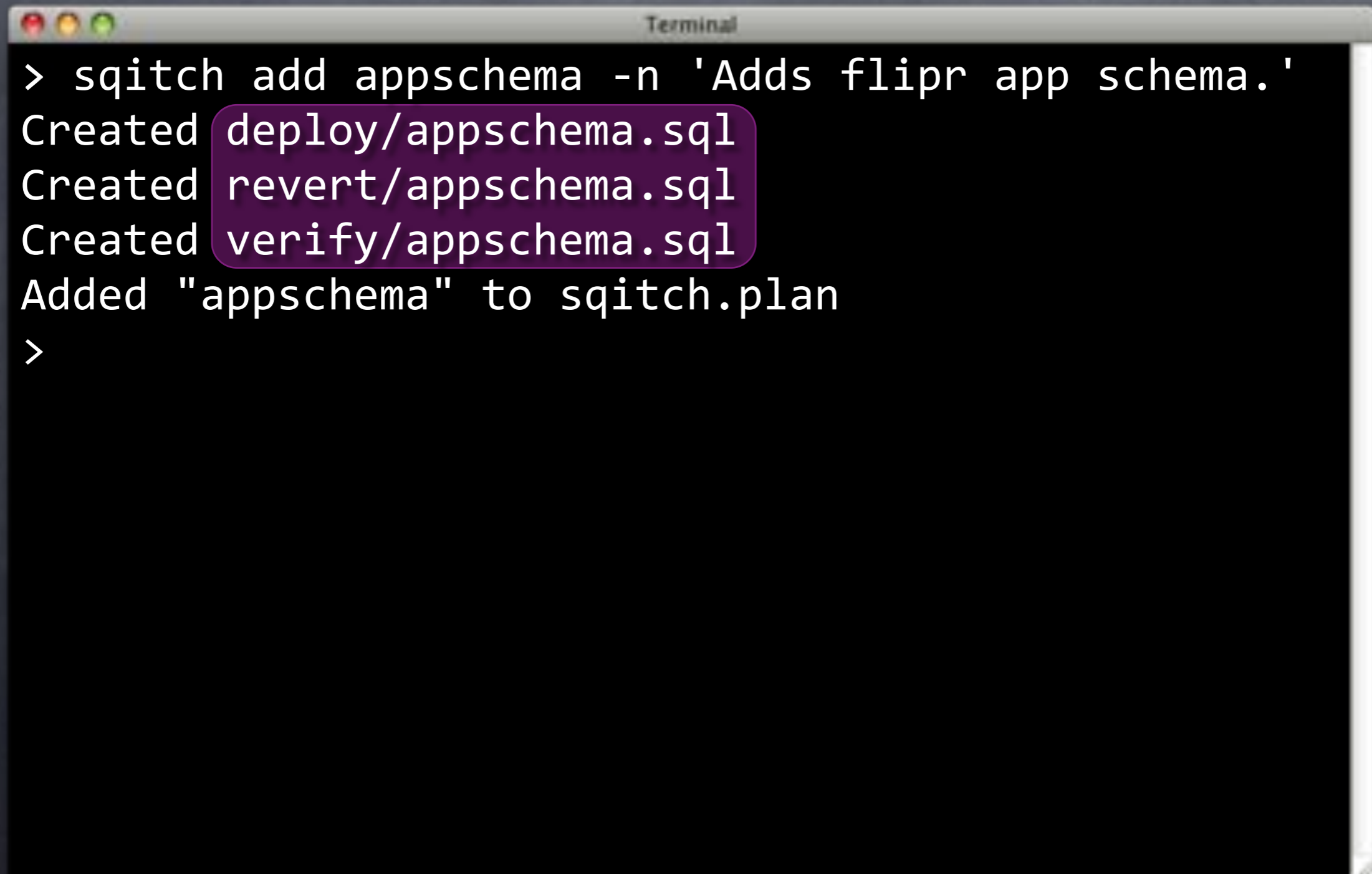


# First Change

A terminal window titled "Terminal" with a dark background and a light gray title bar. The terminal shows the execution of the command `sqitch add appschema -n 'Adds flipr app schema.'` which is highlighted in a green box. The output shows the creation of three SQL files: `deploy/appschema.sql`, `revert/appschema.sql`, and `verify/appschema.sql`, followed by the message "Added 'appschema' to sqitch.plan" and a prompt character `>`.

```
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

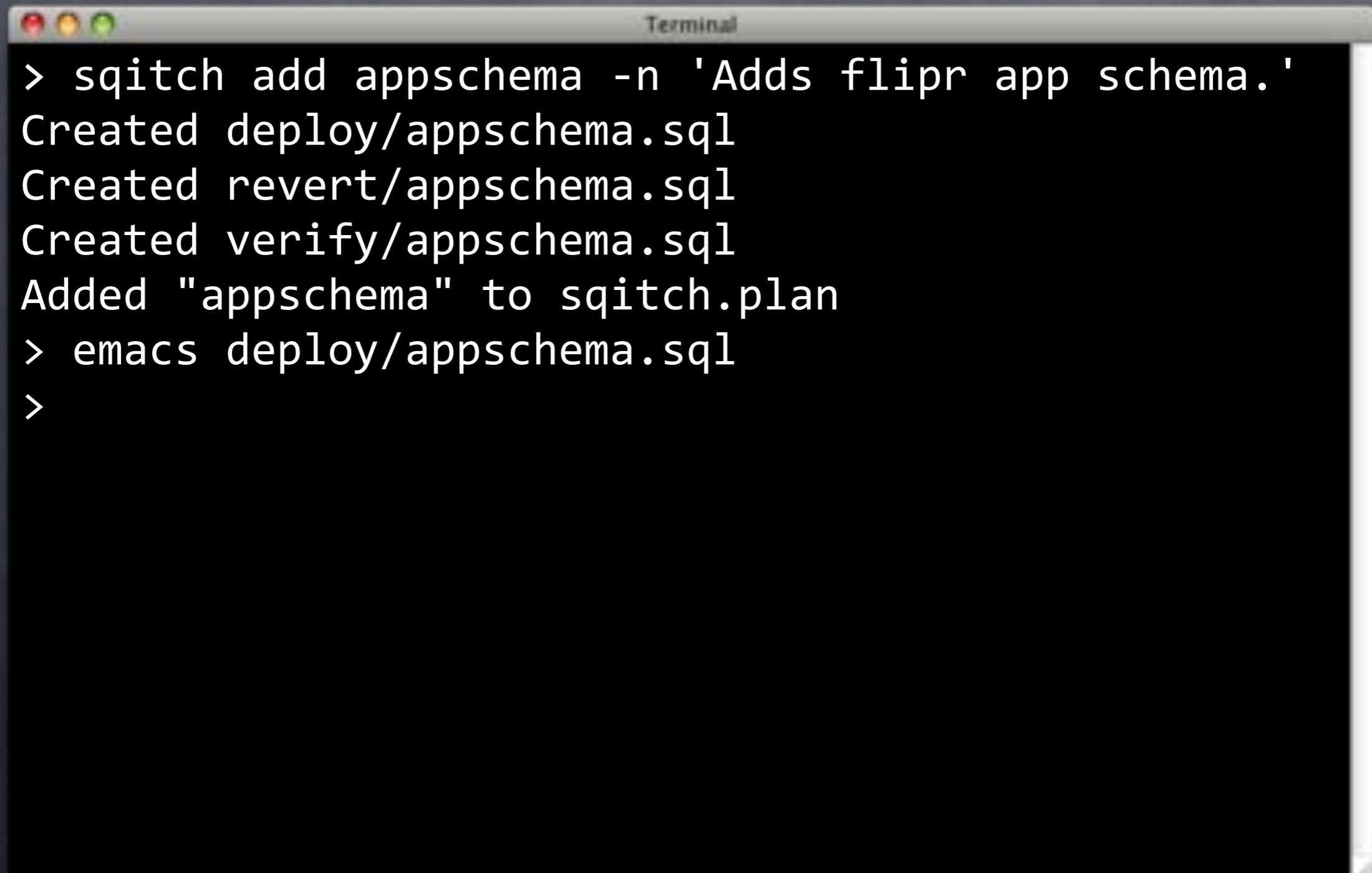
A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows the execution of the command `sqitch add appschema -n 'Adds flipr app schema.'`. The output consists of four lines: `Created deploy/appschema.sql`, `Created revert/appschema.sql`, `Created verify/appschema.sql`, and `Added "appschema" to sqitch.plan`. The first three lines are highlighted with a purple background. The prompt `>` is visible at the beginning and end of the output.

```
> 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
>
```

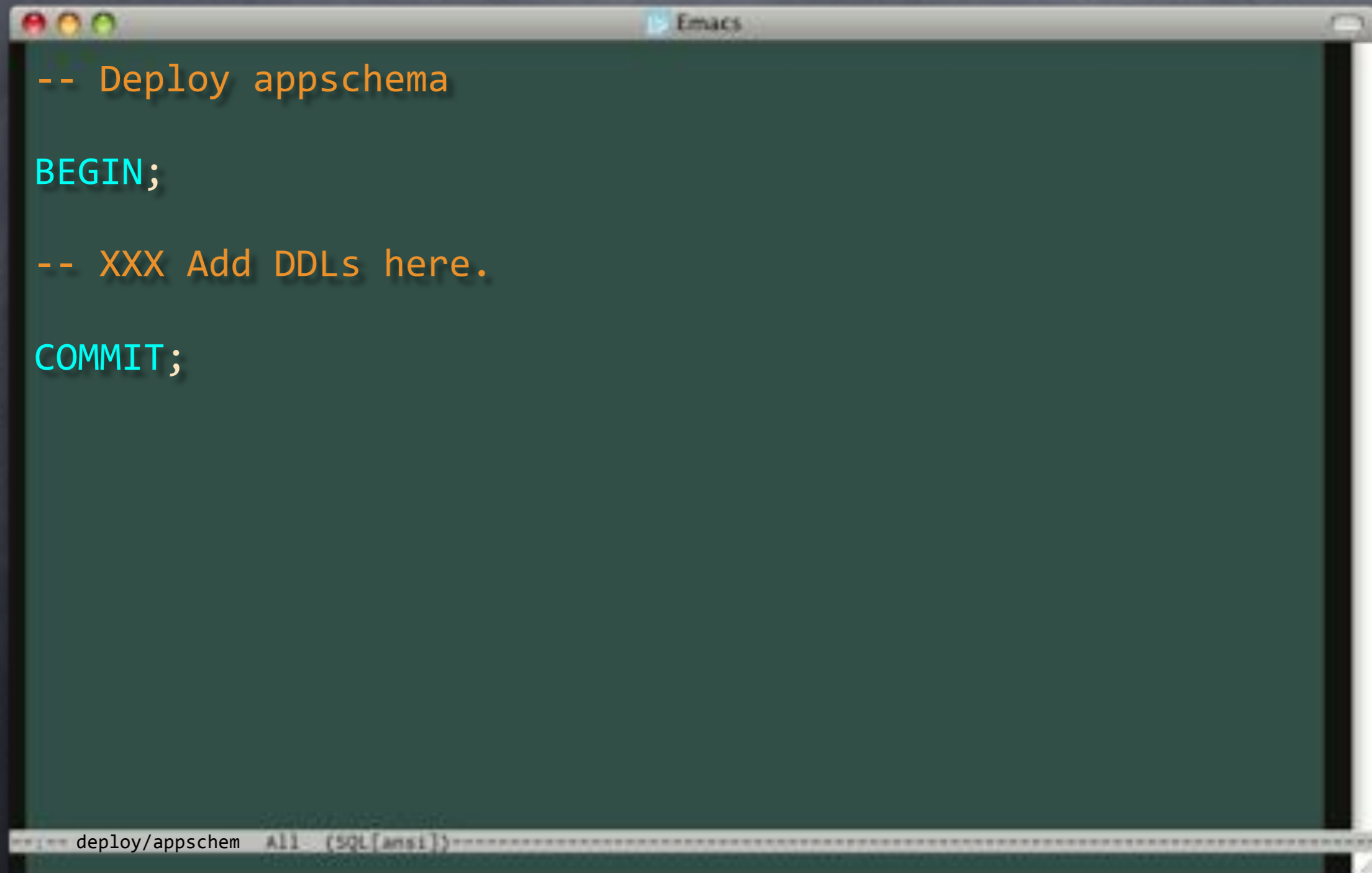
# First Change

A terminal window titled "Terminal" with a standard macOS window title bar (red, yellow, green buttons). The terminal displays the following text:

```
> 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



```
-- Deploy appschema

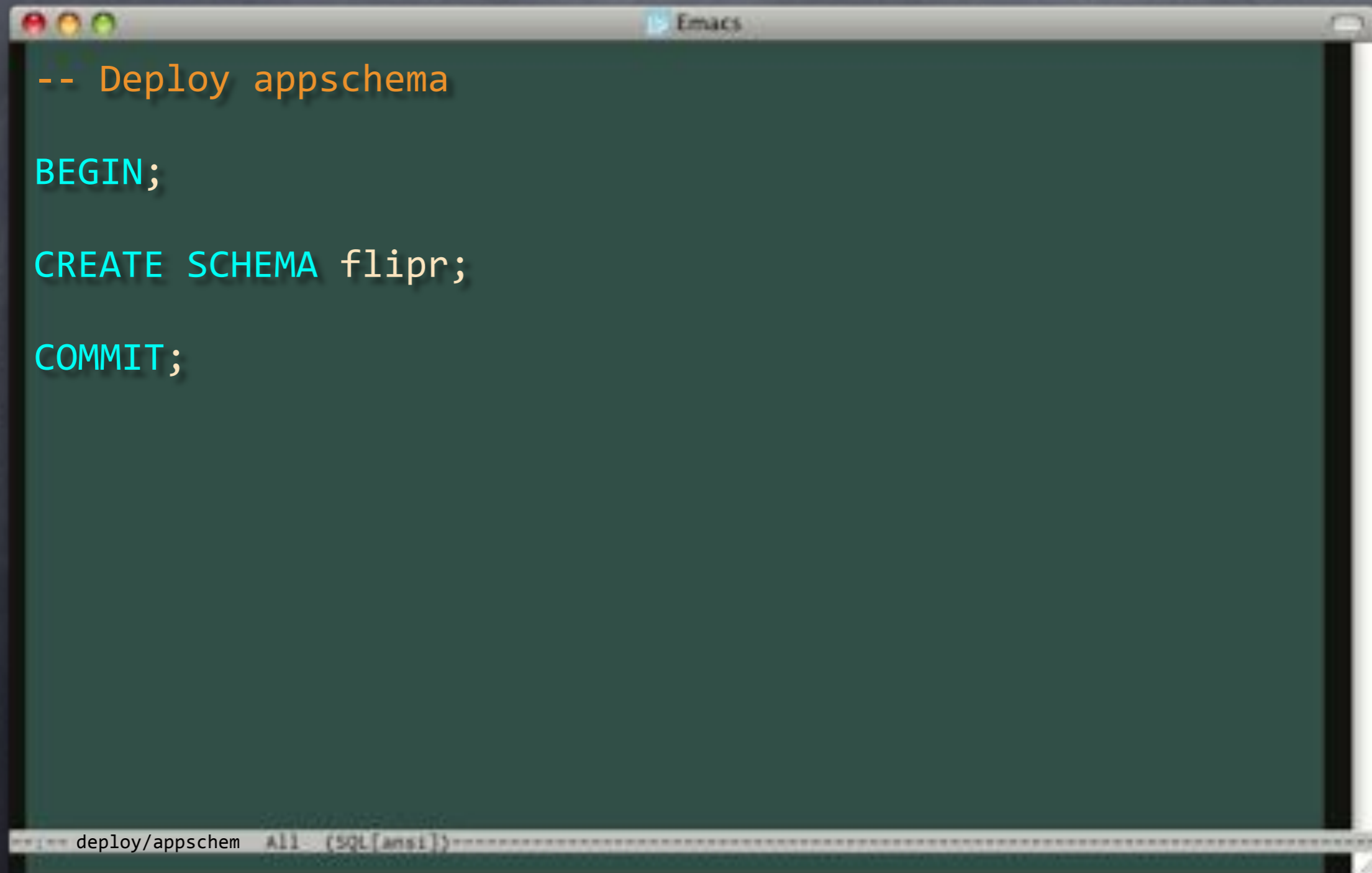
BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

The screenshot shows an Emacs editor window with a dark green background. The code is displayed in a monospaced font with syntax highlighting: comments are orange, SQL keywords are cyan, and the rest is white. The window title bar shows 'Emacs' and standard macOS window controls. The status bar at the bottom indicates the file path 'deploy/appschem', the buffer name 'All', and the encoding '(SQL[ansi])'.

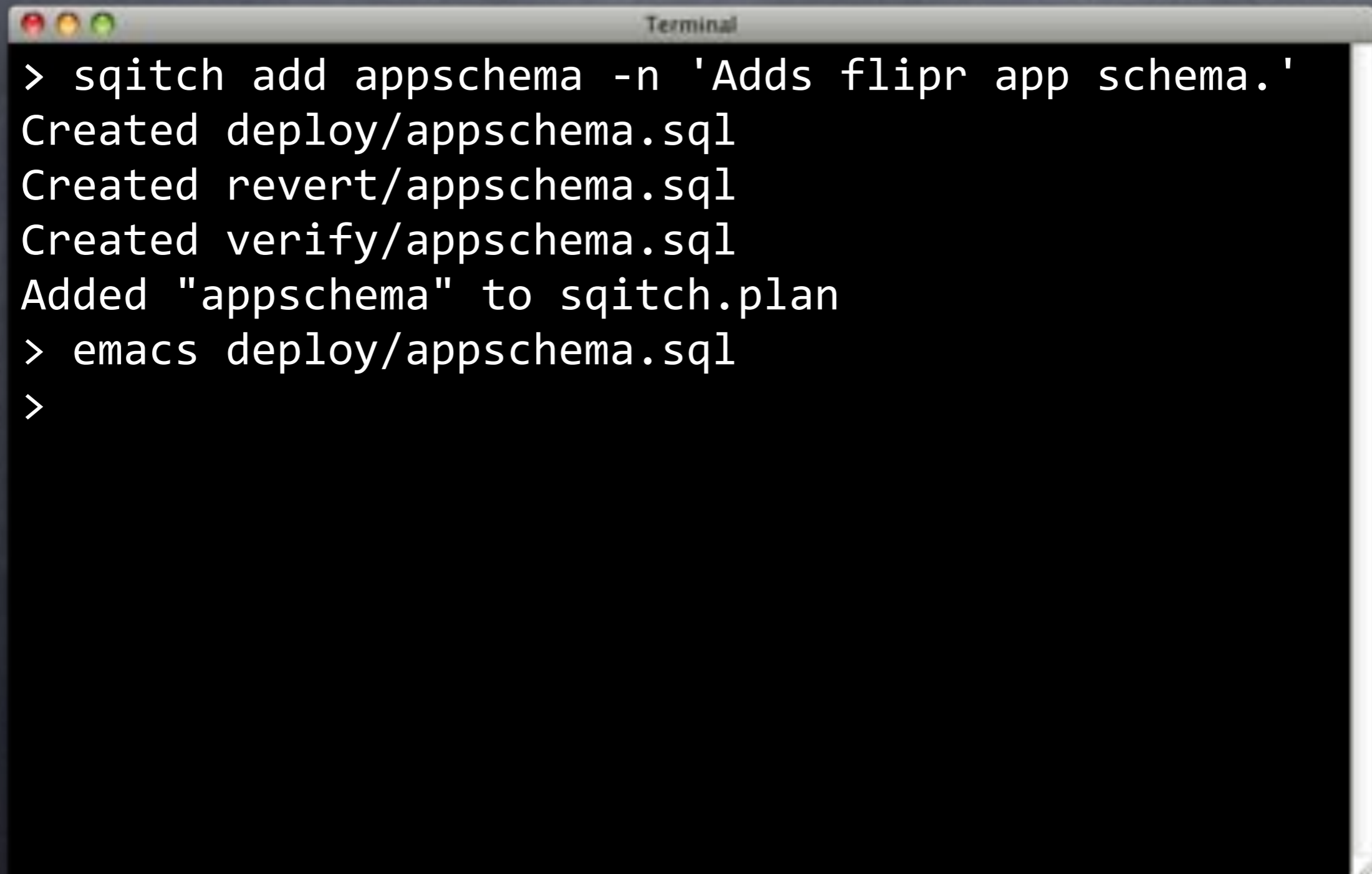
# deploy/appschema.sql



```
-- Deploy appschema  
  
BEGIN;  
  
CREATE SCHEMA flipr;  
  
COMMIT;
```

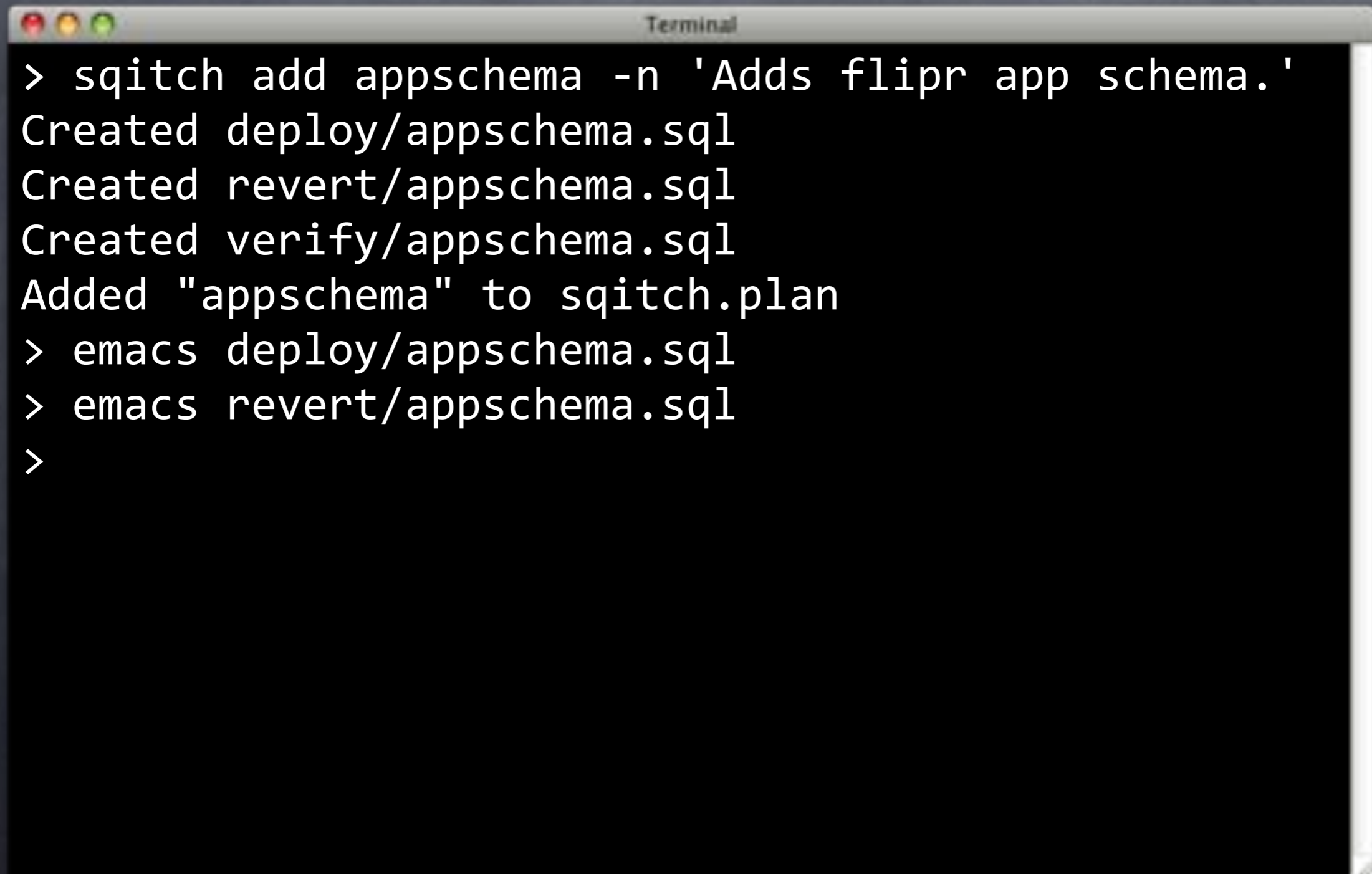
The image shows a screenshot of an Emacs editor window. The window title is "Emacs". The editor content is a SQL script with the following lines: "-- Deploy appschema", "BEGIN;", "CREATE SCHEMA flipr;", and "COMMIT;". The status bar at the bottom of the window shows "deploy/appschem All (SQL[ansi])".

# First Change

A terminal window titled "Terminal" with a standard macOS window title bar (red, yellow, green buttons). The terminal displays the following commands and output:

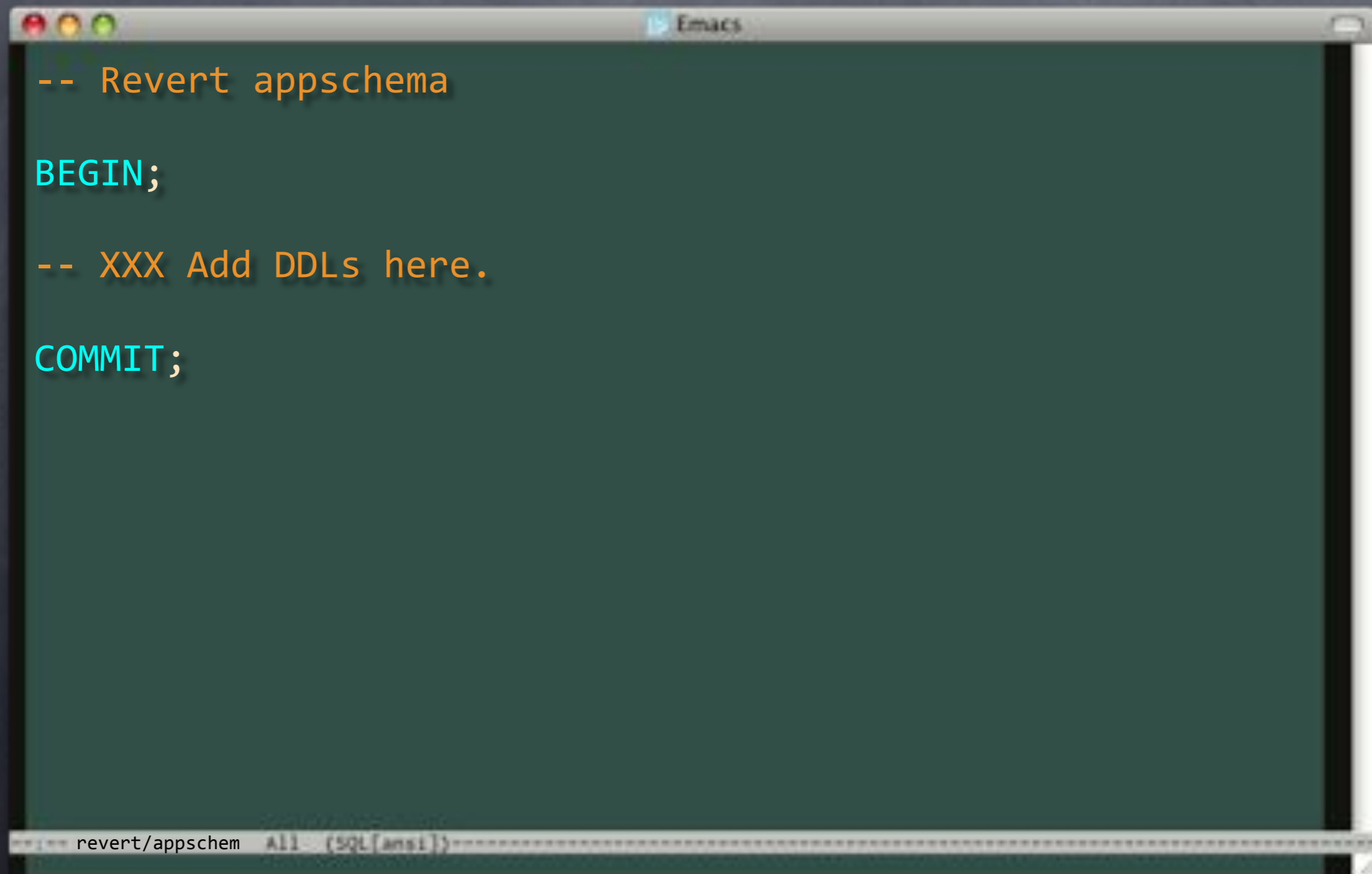
```
> sqitch add appschema -n 'Adds flipr app schema.'  
Created deploy/appschema.sql  
Created revert/appschema.sql  
Created verify/appschema.sql  
Added "appschema" to sqitch.plan  
> emacs deploy/appschema.sql  
>
```

# First Change

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows the execution of the 'sqitch add' command to create a new schema named 'appschema'. The output indicates that three SQL files were created: 'deploy/appschema.sql', 'revert/appschema.sql', and 'verify/appschema.sql'. The 'appschema' entry was also added to the 'sqitch.plan' file. Finally, the user opens the 'deploy/appschema.sql' and 'revert/appschema.sql' files in Emacs.

```
> 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



```
-- Revert appschema

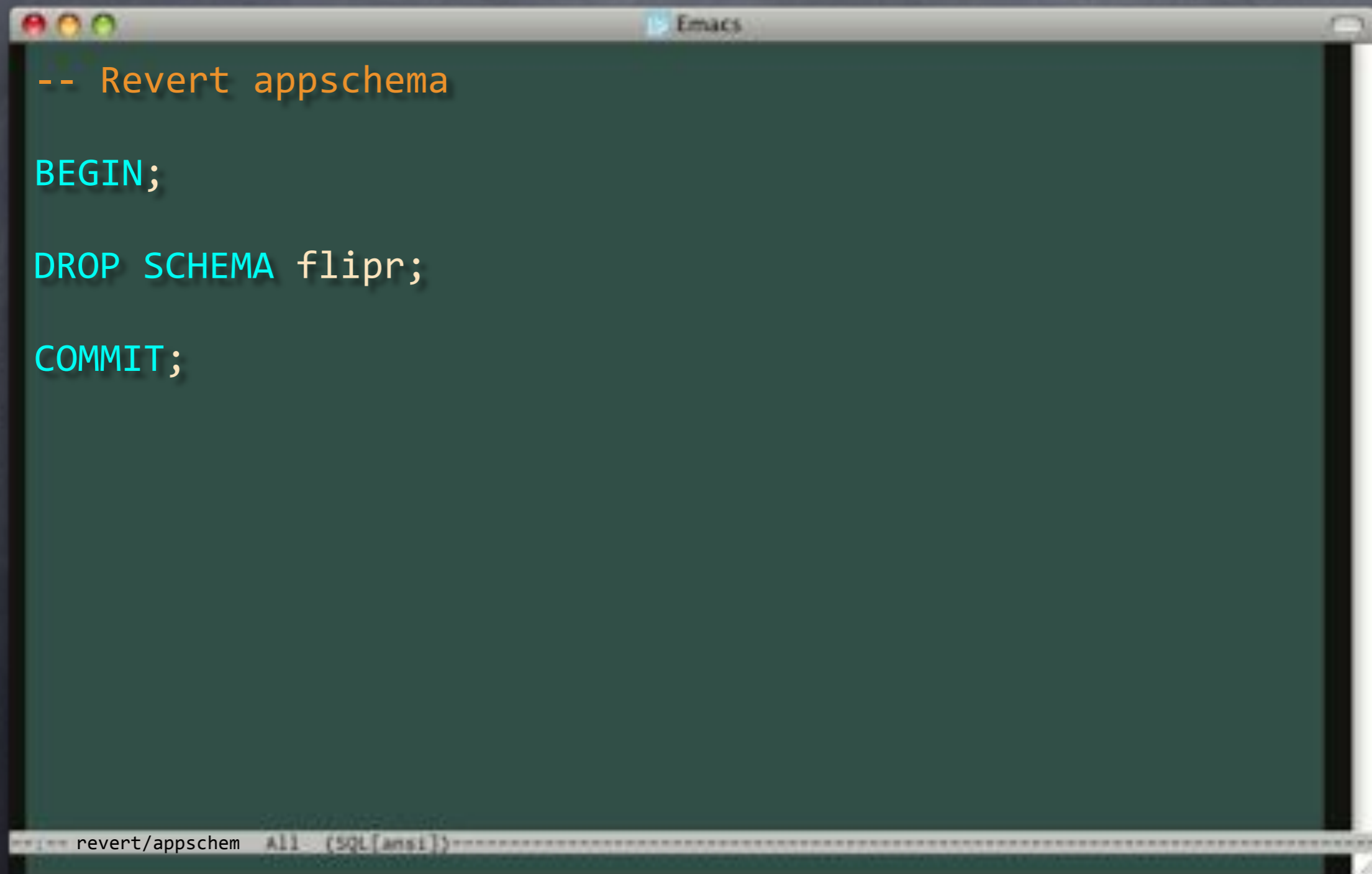
BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

The screenshot shows an Emacs editor window with a dark green background. The code is color-coded: comments are orange, and SQL keywords are cyan. The status bar at the bottom of the window displays "revert/appschem All (SQL[ansi])".

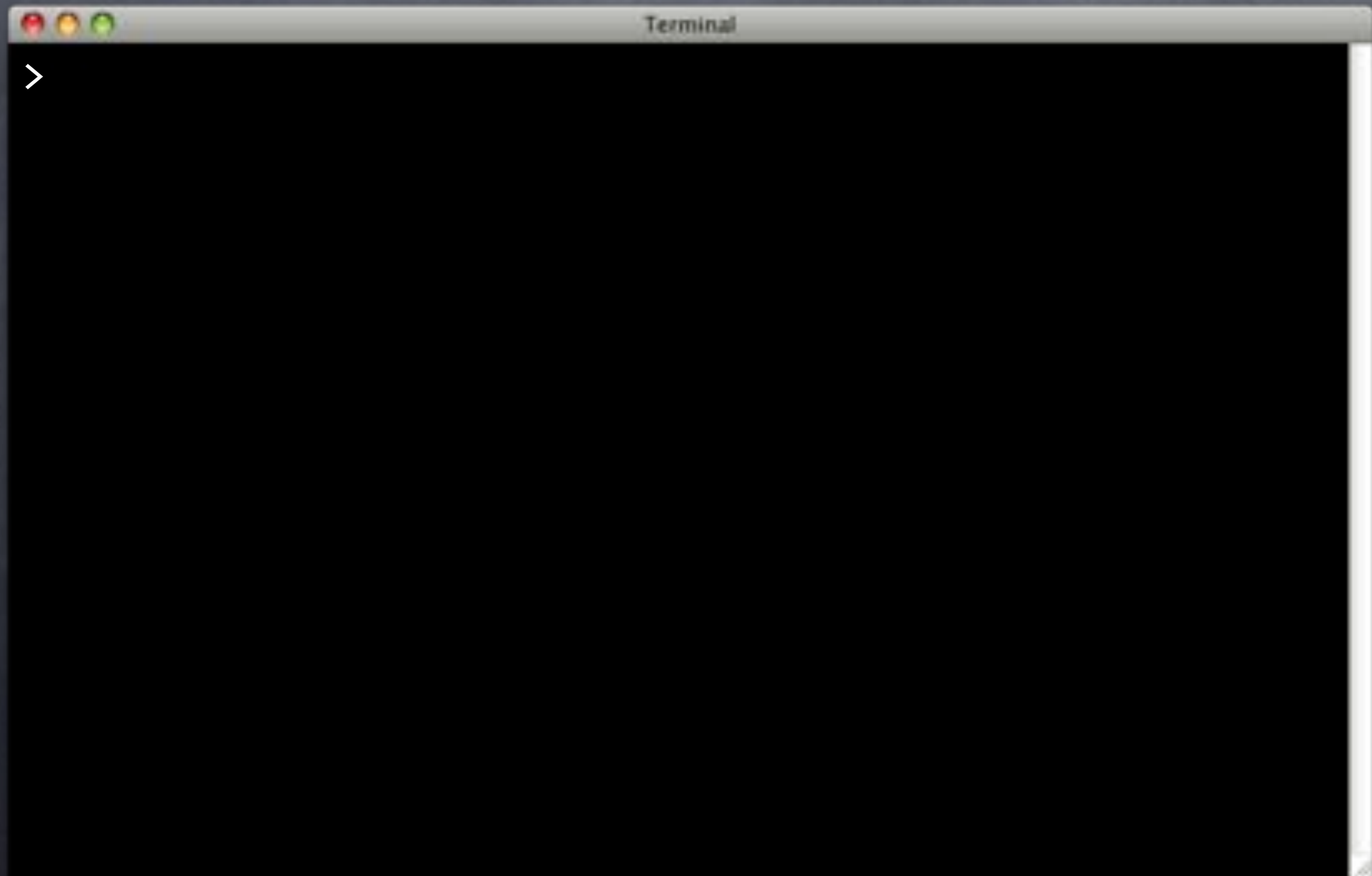
# revert/appschema.sql

A screenshot of an Emacs editor window. The window title bar shows "Emacs" and standard macOS window controls (red, yellow, green buttons). The main editing area has a dark green background and contains the following SQL code:

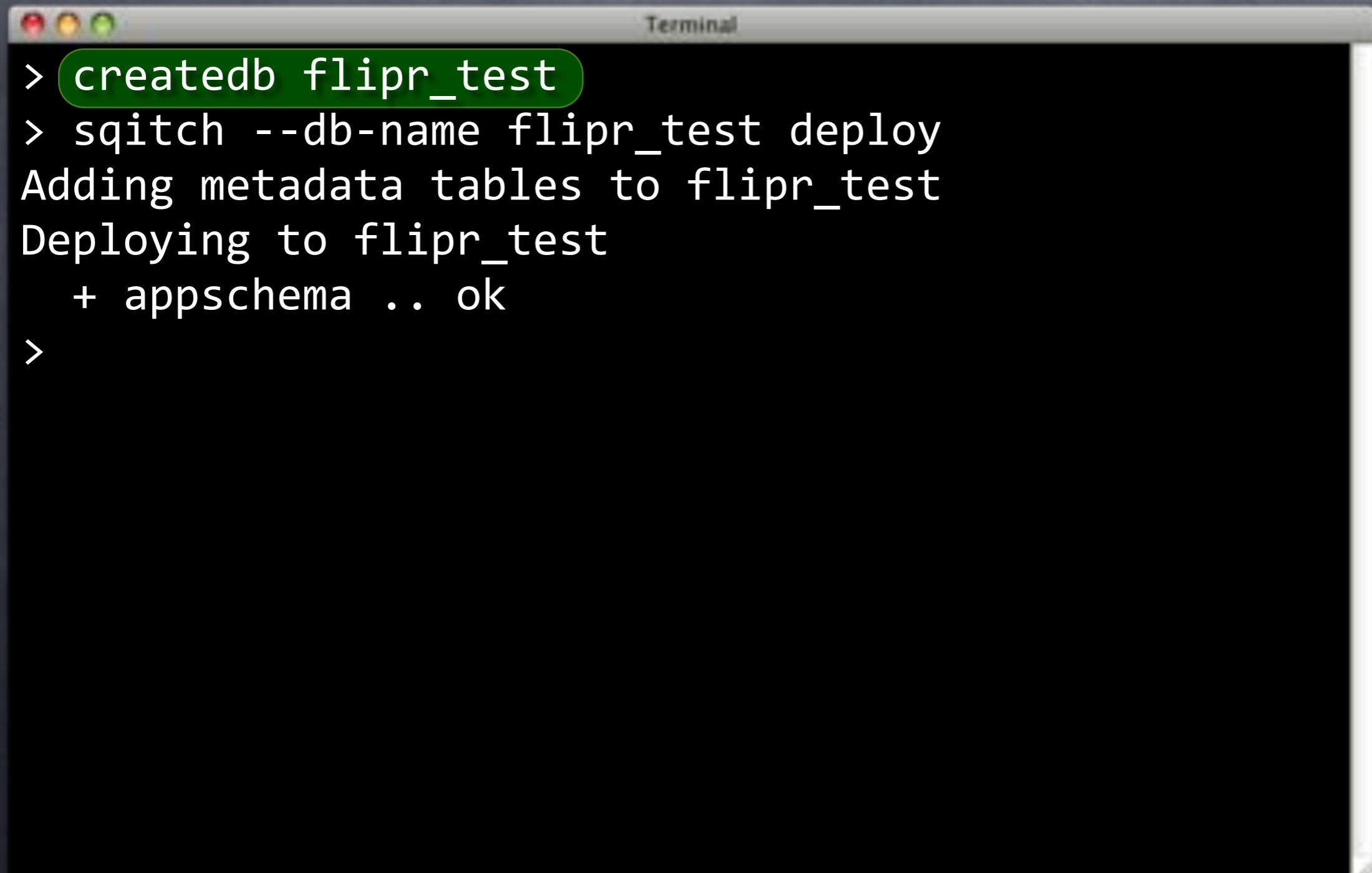
```
-- Revert appschema  
  
BEGIN;  
  
DROP SCHEMA flipr;  
  
COMMIT;
```

The status bar at the bottom of the window shows "revert/appschem All (SQL[ansi])".

# Make it So!



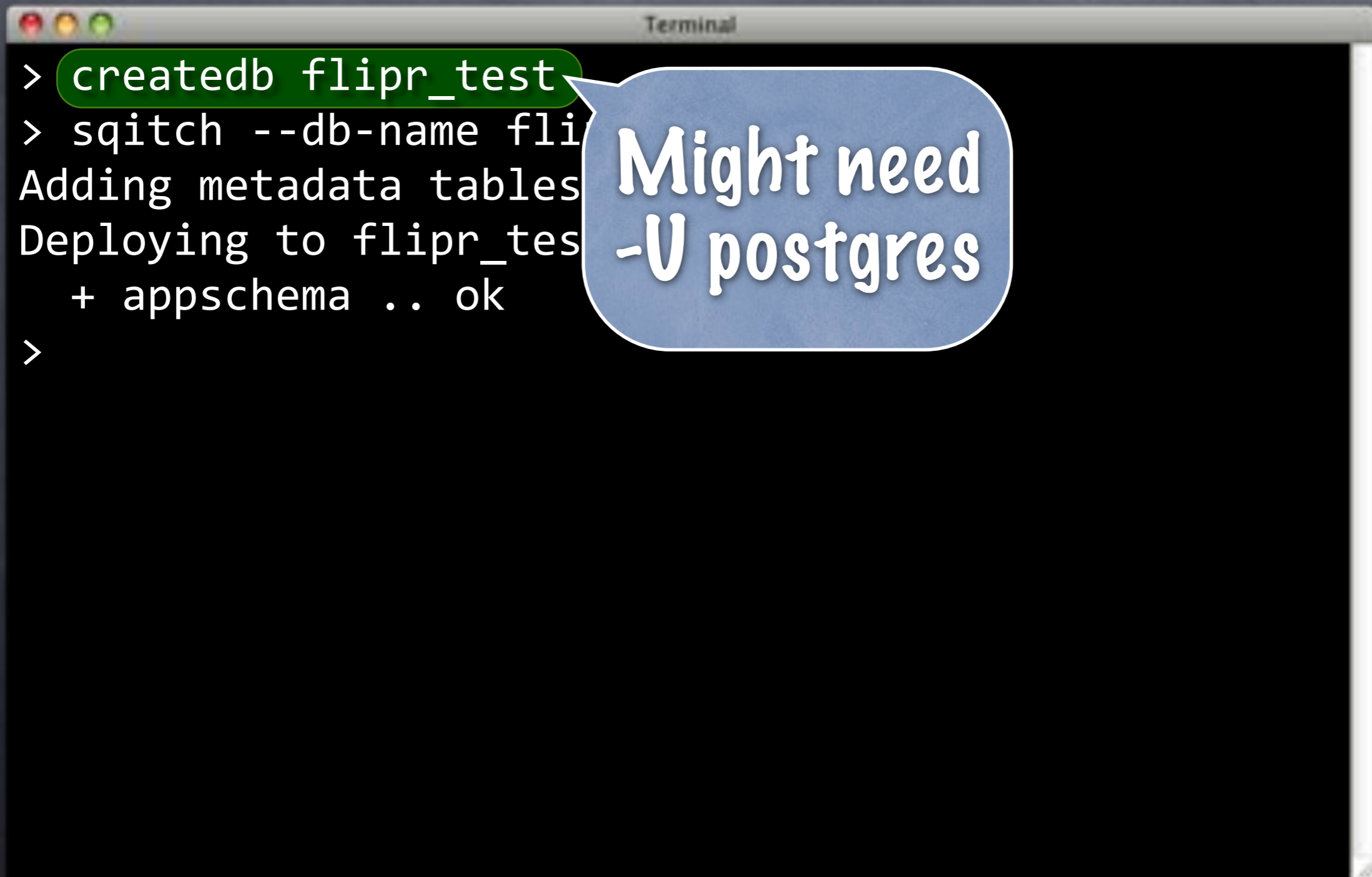
# Make it So!

A terminal window titled "Terminal" with a dark background and light text. The window shows a sequence of commands and their outputs. The first command, "createdb flipr\_test", is highlighted with a green background. The second command, "sqitch --db-name flipr\_test deploy", is followed by three lines of output: "Adding metadata tables to flipr\_test", "Deploying to flipr\_test", and "+ appschema .. ok". The terminal ends with a prompt character ">".

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



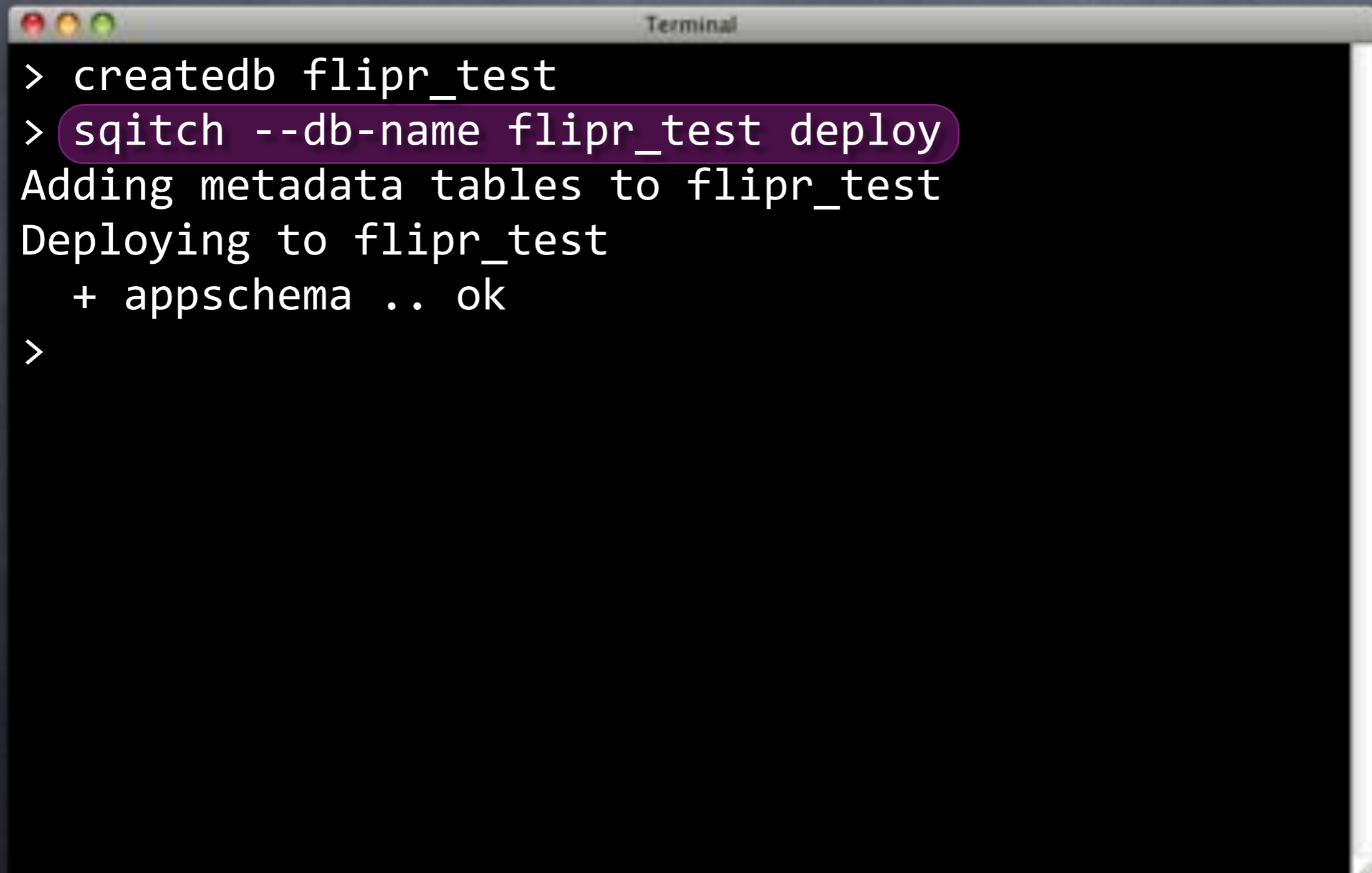
# Make it So!

A terminal window titled "Terminal" with a dark background and light text. The first command, "> createdb flipr\_test", is highlighted with a green background. The second command, "> sqitch --db-name flipr\_test", is followed by the output "Adding metadata tables" and "Deploying to flipr\_test + appschema .. ok". A blue callout bubble with white text points to the first command, containing the text "Might need -U postgres".

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

Might need  
-U postgres

# Make it So!

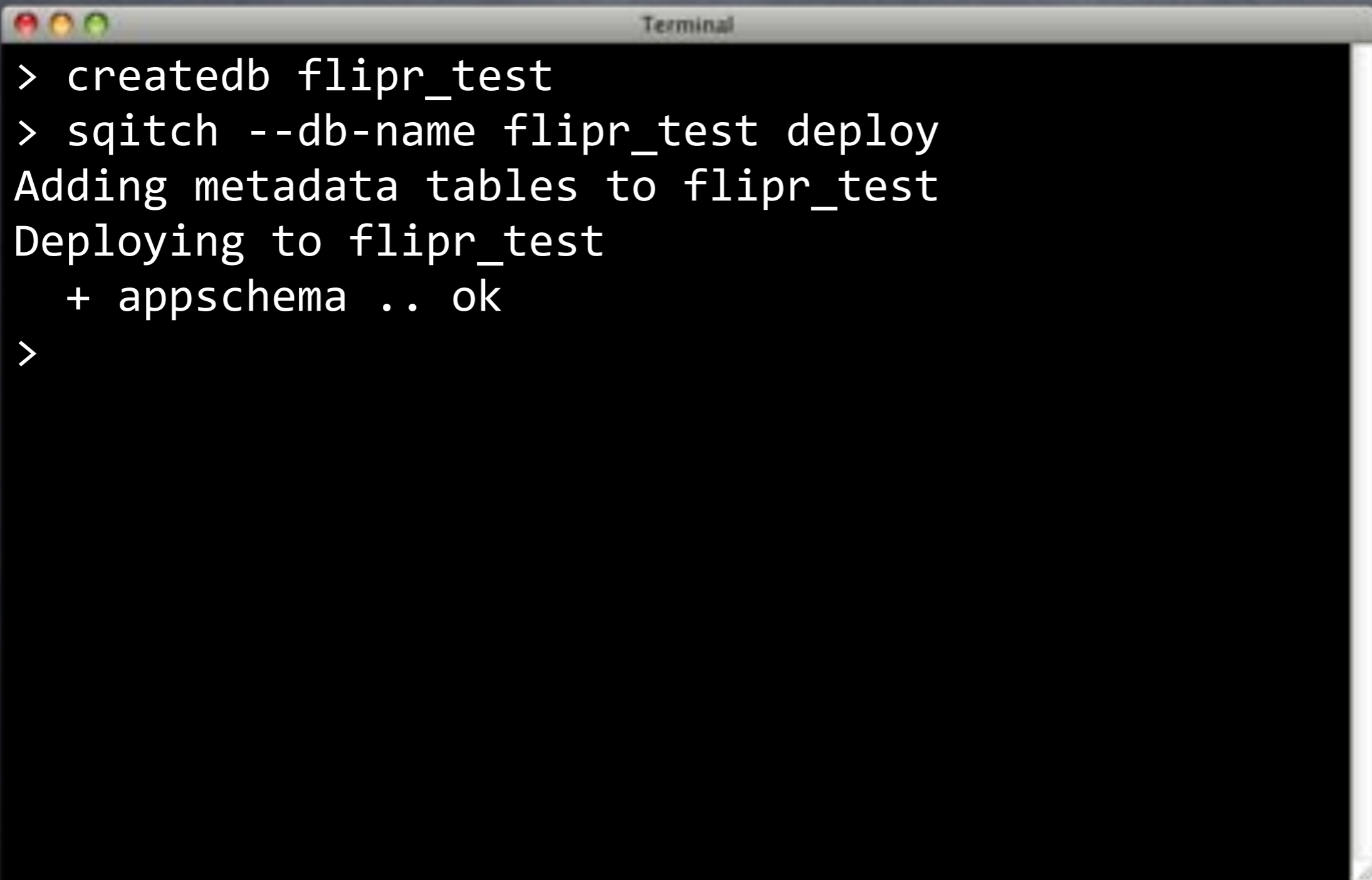
A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows a sequence of commands and their outputs. The first command is "createdb flipr\_test". The second command is "sqitch --db-name flipr\_test deploy", which is highlighted with a purple background. The output of this command is "Adding metadata tables to flipr\_test" and "Deploying to flipr\_test + appschema .. ok". The terminal ends with a prompt character ">".

```
> 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
>
```

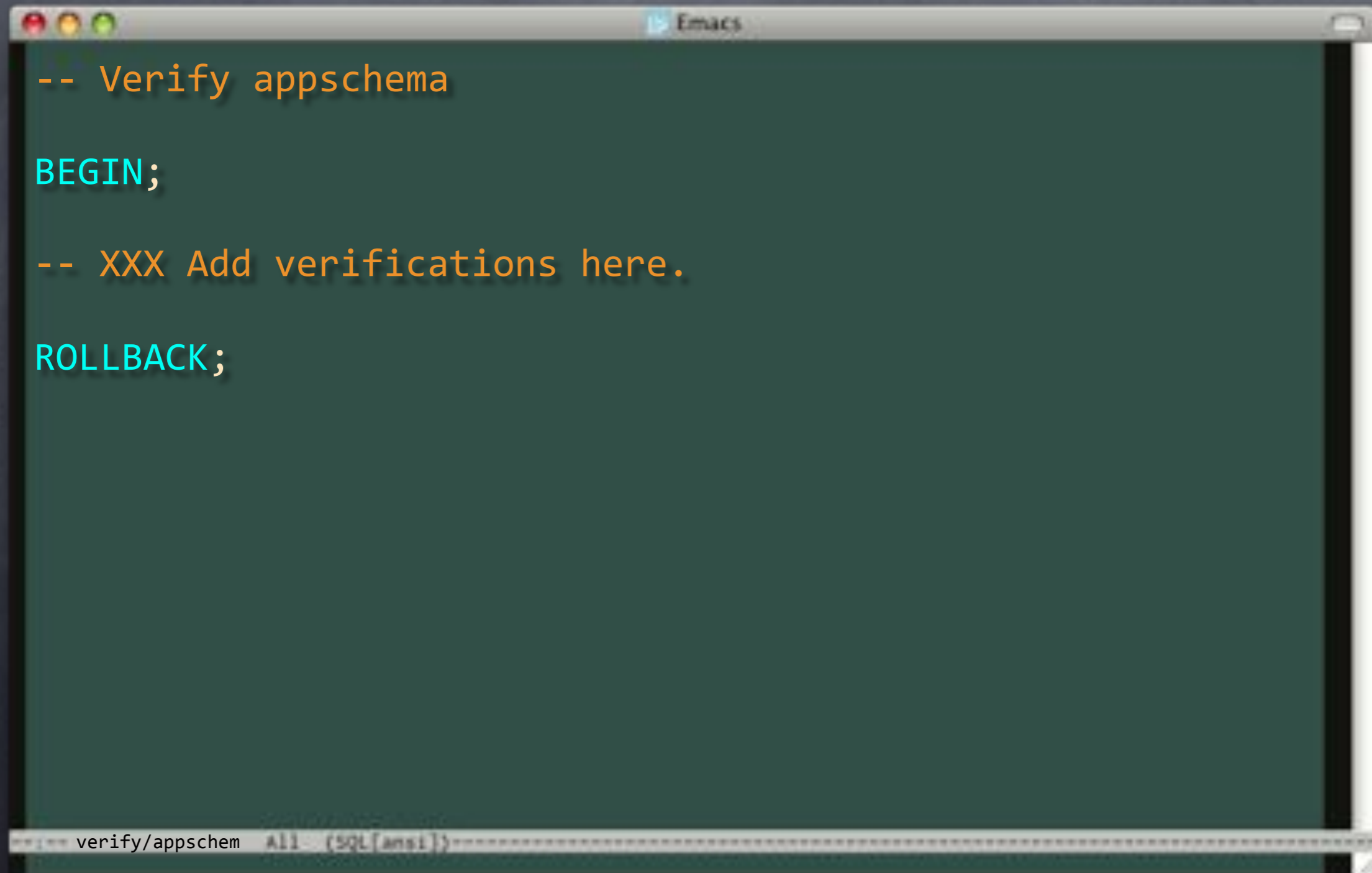
Trust, but verify

# 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



```
-- Verify appschema

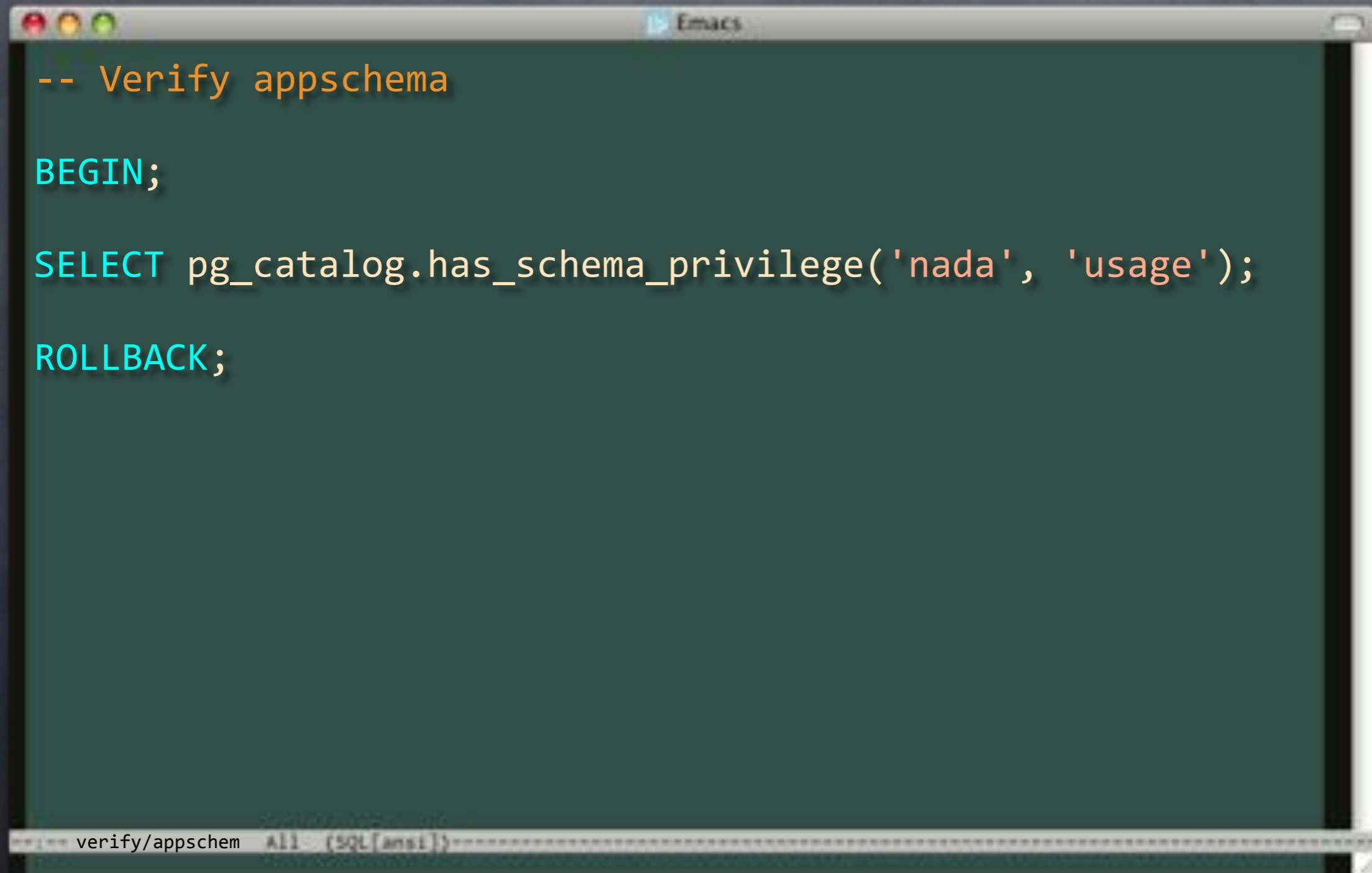
BEGIN;

-- XXX Add verifications here.

ROLLBACK;
```

The screenshot shows an Emacs editor window with a dark green background. The code is color-coded: comments are orange, and SQL keywords are cyan. The window title bar shows 'Emacs' and standard macOS window controls. The status bar at the bottom indicates the file path 'verify/appschem', the buffer name 'All', and the mode '(SQL[ansi])'.

# verify/appschema.sql

A screenshot of an Emacs editor window. The window title bar shows the Emacs logo and the text "Emacs". The editor content is a dark green background with SQL code in various colors: orange for comments, cyan for keywords, and white for identifiers and literals. The code is as follows:

```
-- Verify appschema  
  
BEGIN;  
  
SELECT pg_catalog.has_schema_privilege('nada', 'usage');  
  
ROLLBACK;
```

The status bar at the bottom of the window shows "verify/appschem All (SQL[ansi])".

# verify/appschema.sql

```
-- Verify appschema
```

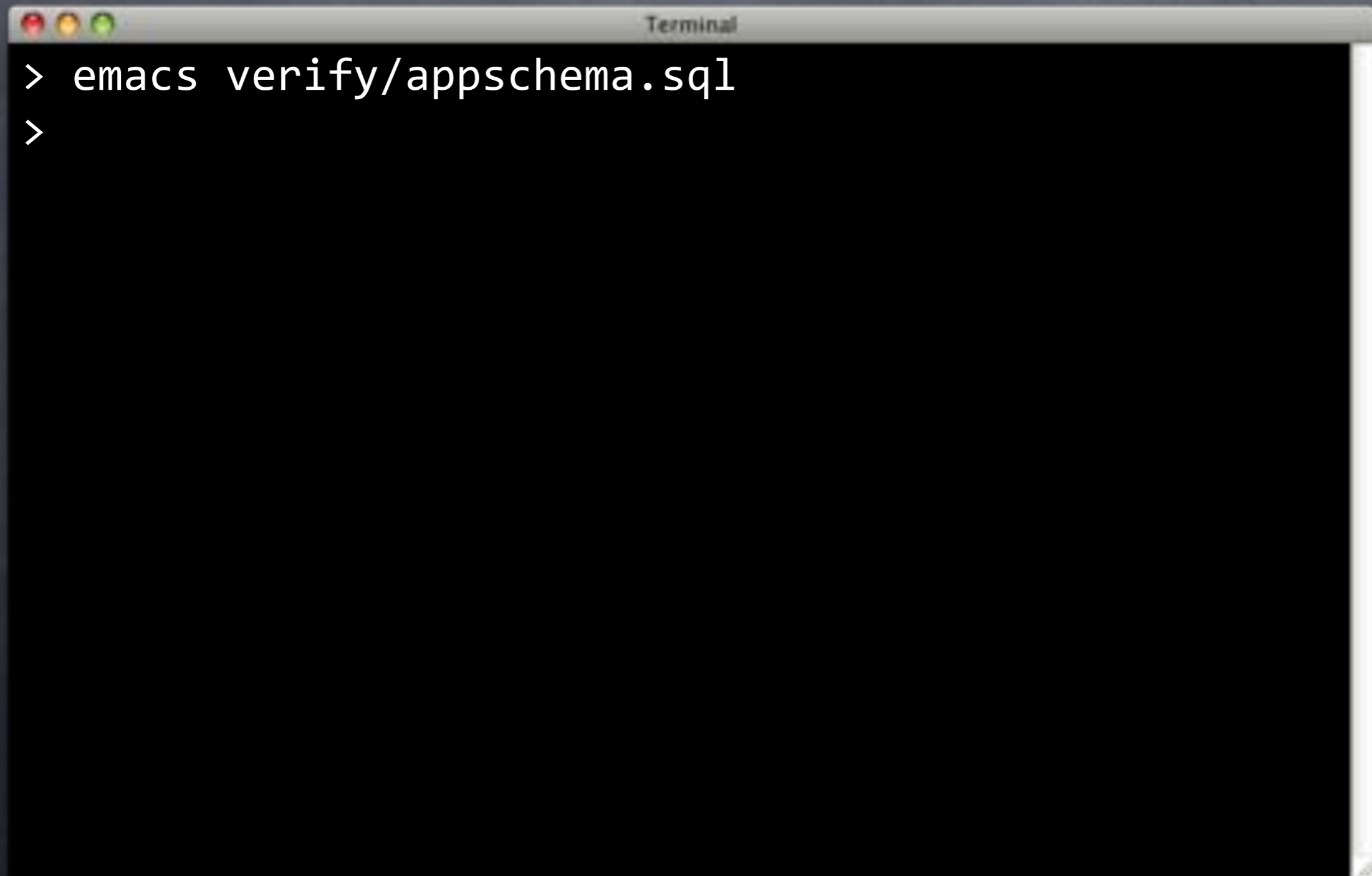
```
BEGIN;
```

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

```
ROLLBACK;
```

Let's try  
it, first

# Trust, But Verify

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows a prompt character followed by the command "emacs verify/appschema.sql" and a second prompt character on the next line.

```
> emacs verify/appschema.sql  
>
```

# 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
>
```

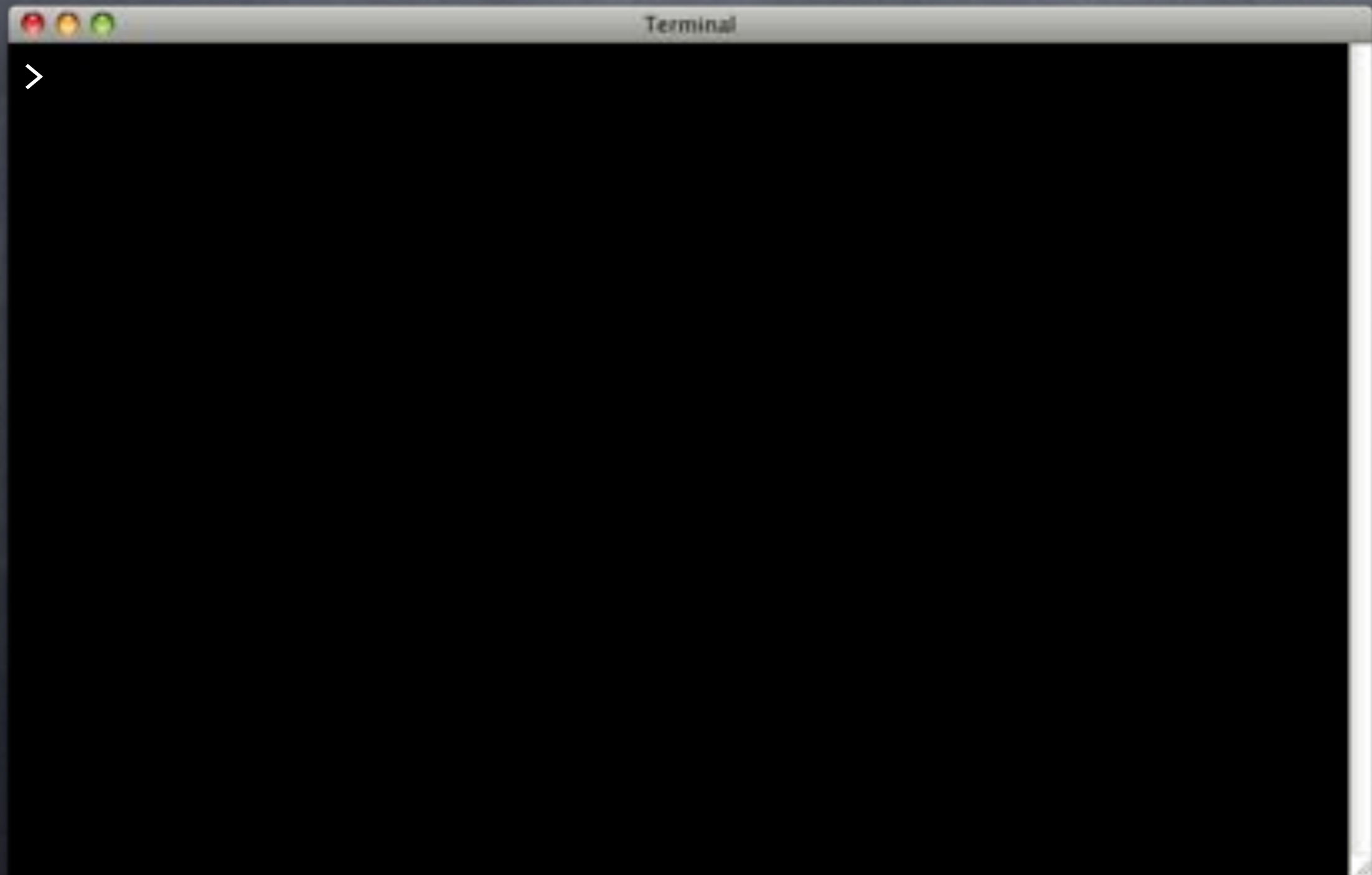
# Trust, But Verify

```
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

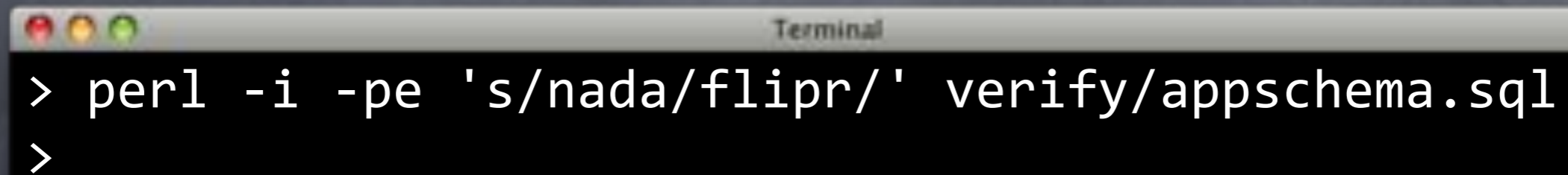
Verify Summary Report
-----
Changes: 1
Errors: 1
Verify failed
>
```



# Trust, But Verify



# Trust, But Verify

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a command being entered: 

```
> perl -i -pe 's/nada/flipr/' verify/appschema.sql
```

 followed by a second prompt character 

```
>
```

 on the next line.

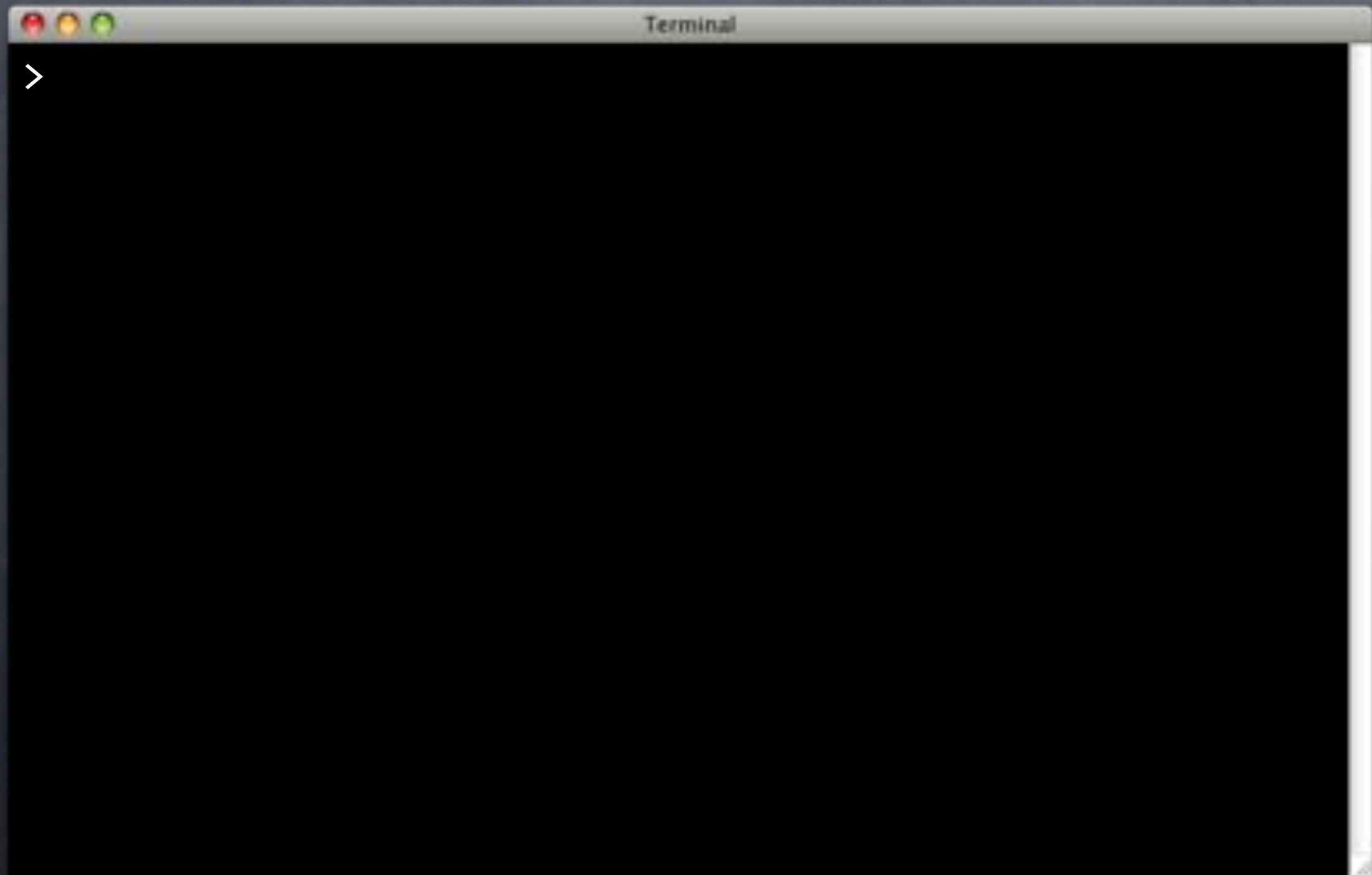
```
> perl -i -pe 's/nada/flipr/' verify/appschema.sql
>
```

# Trust, But Verify

```
Terminal
> perl -i -pe 's/nada/flipr/' verify/appschemata.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)
>
```



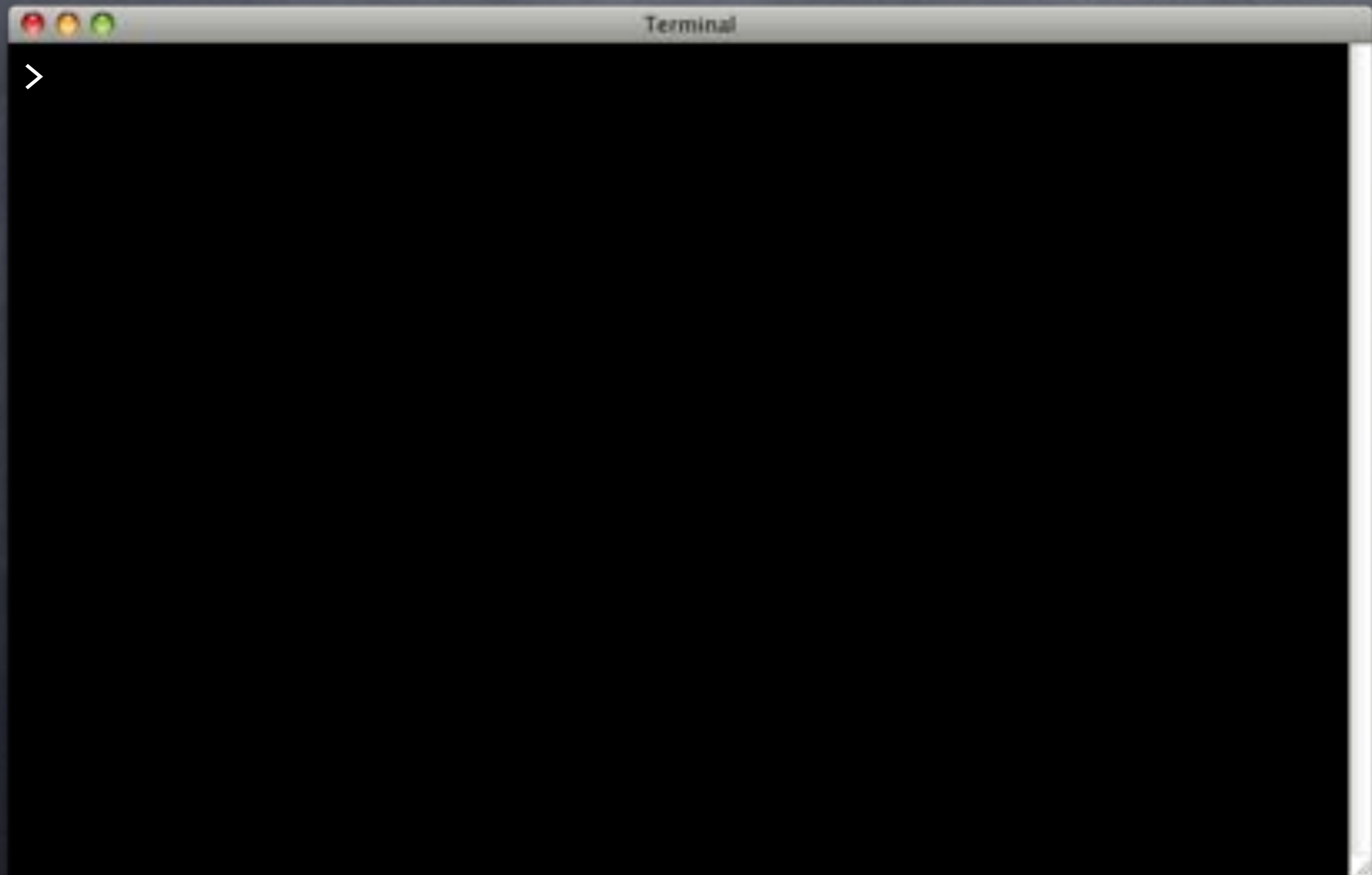
# How's it Look?

```
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)
>
```

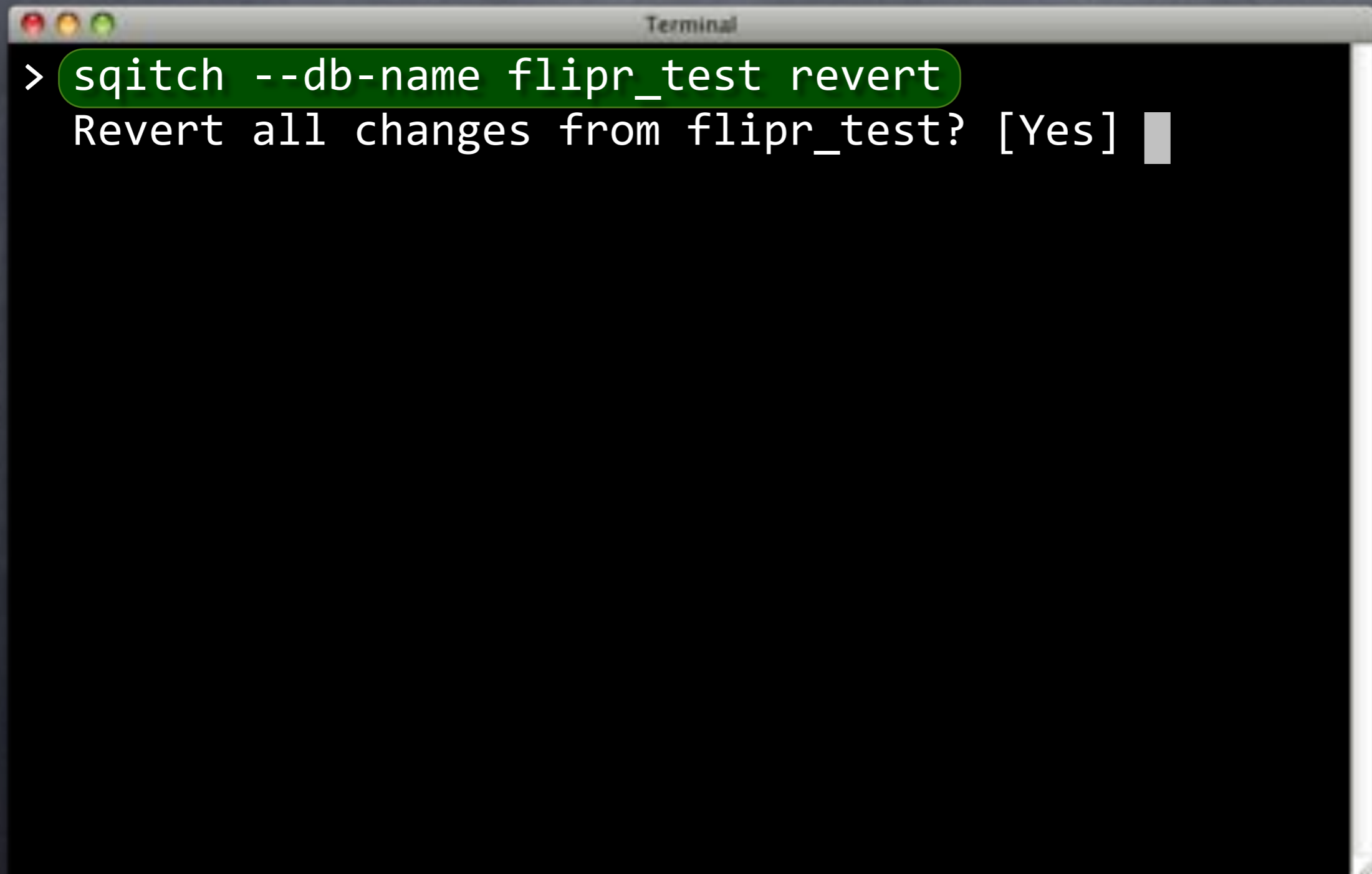
# How's it Look?

```
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

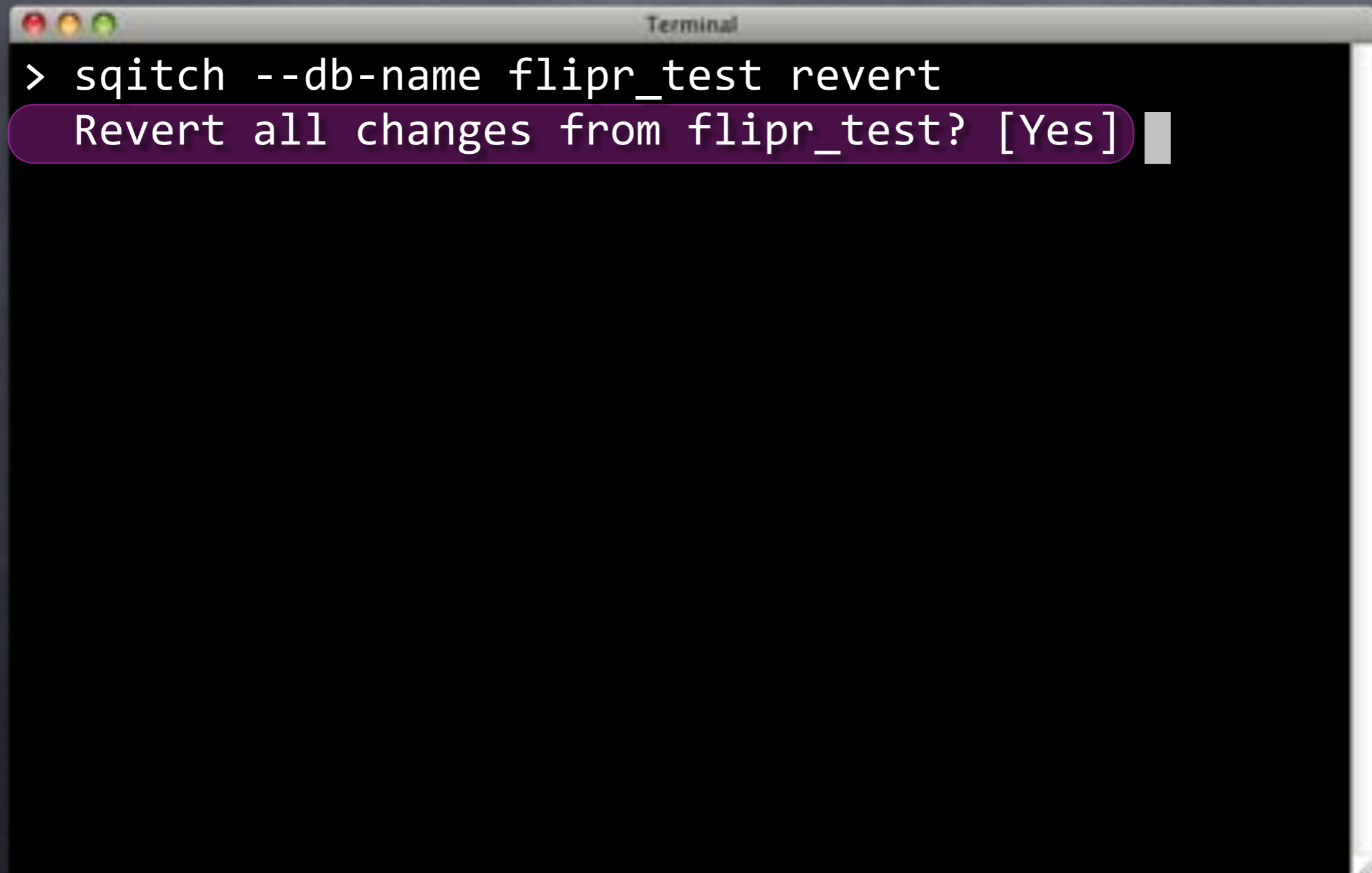


# Go Back



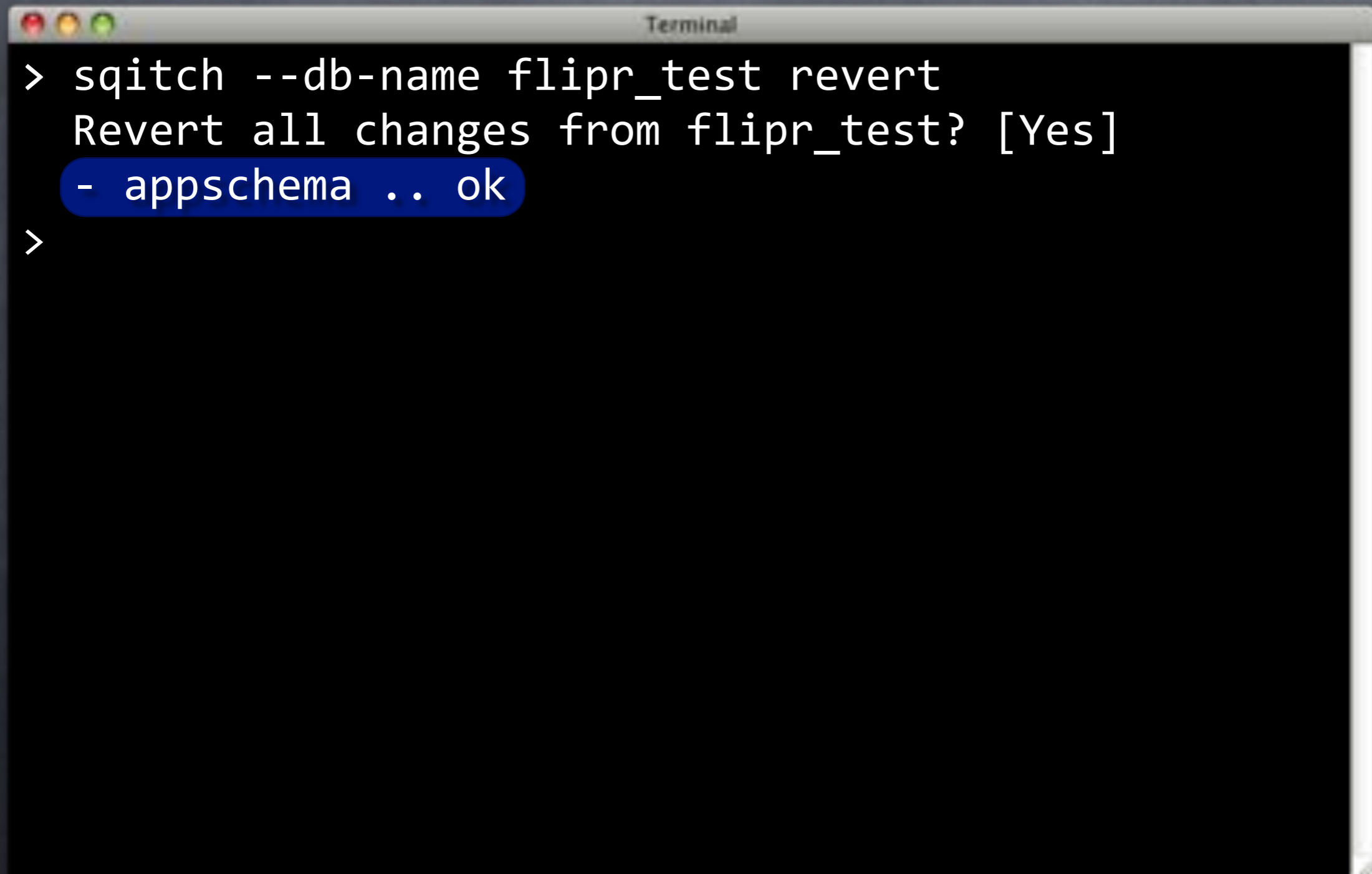
```
Terminal  
> sqitch --db-name flipr_test revert  
Revert all changes from flipr_test? [Yes] █
```

# 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
-----+-----
>
```

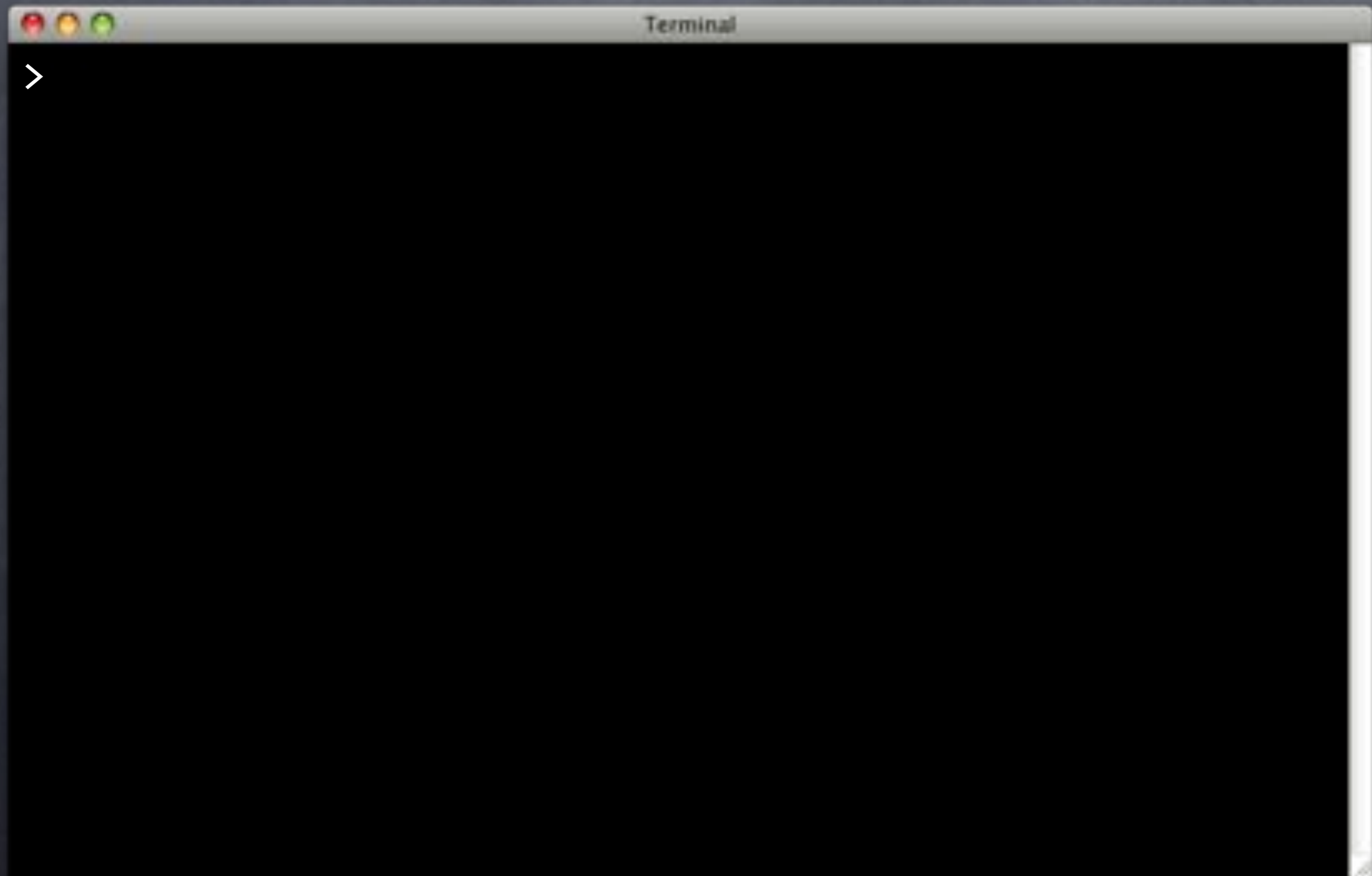
# Go Back

```
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

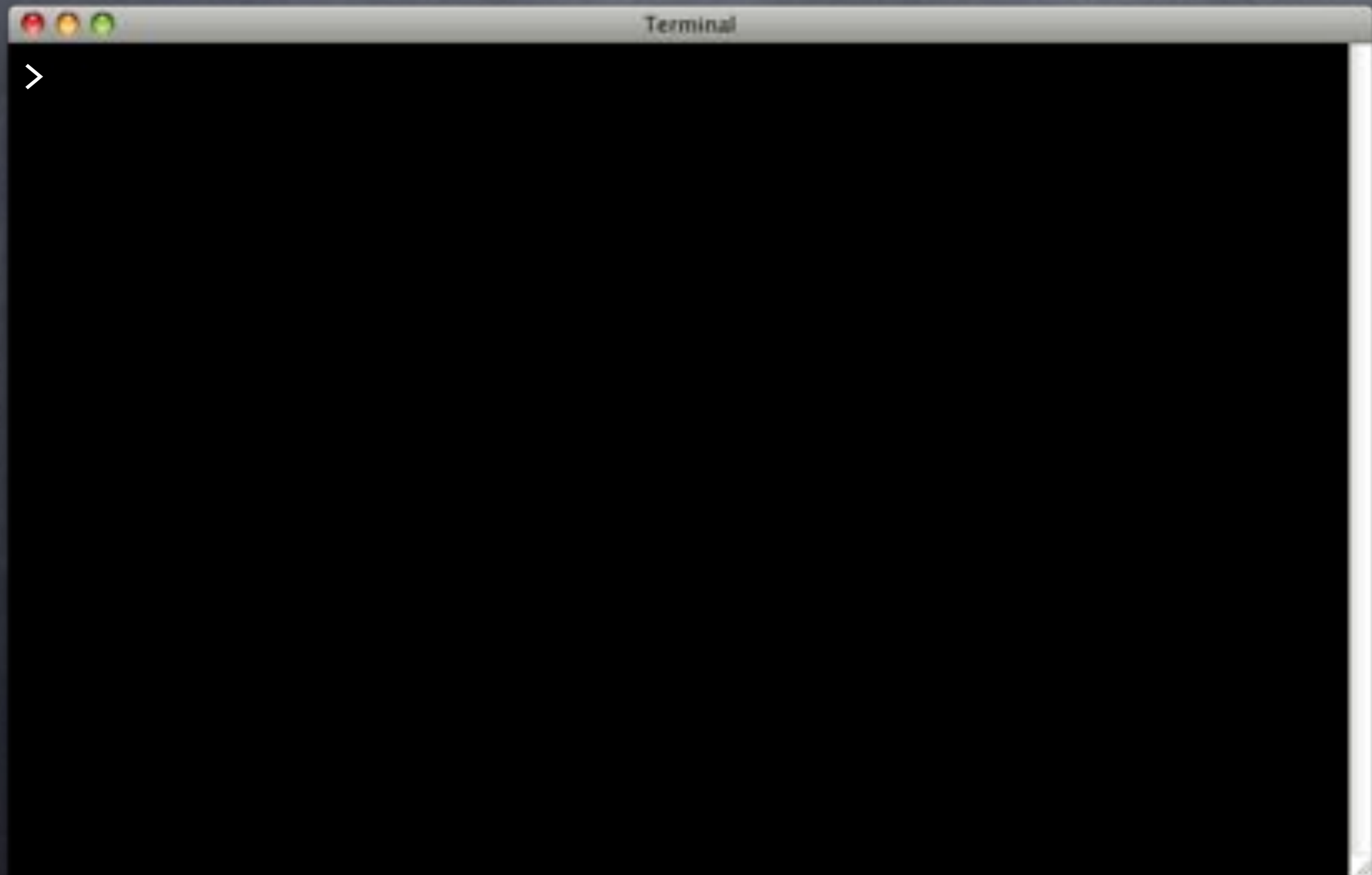
```
Terminal
> sqitch -d flipr_test log
On database flipr_test
Revert 748346dfe73cf2af32a8b7088fd75ad8d7aecda3
Name:      appschema
Committer: David E. Wheeler <david@justatheory.com>
Date:      2013-05-21 15:07:07 -0400

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!



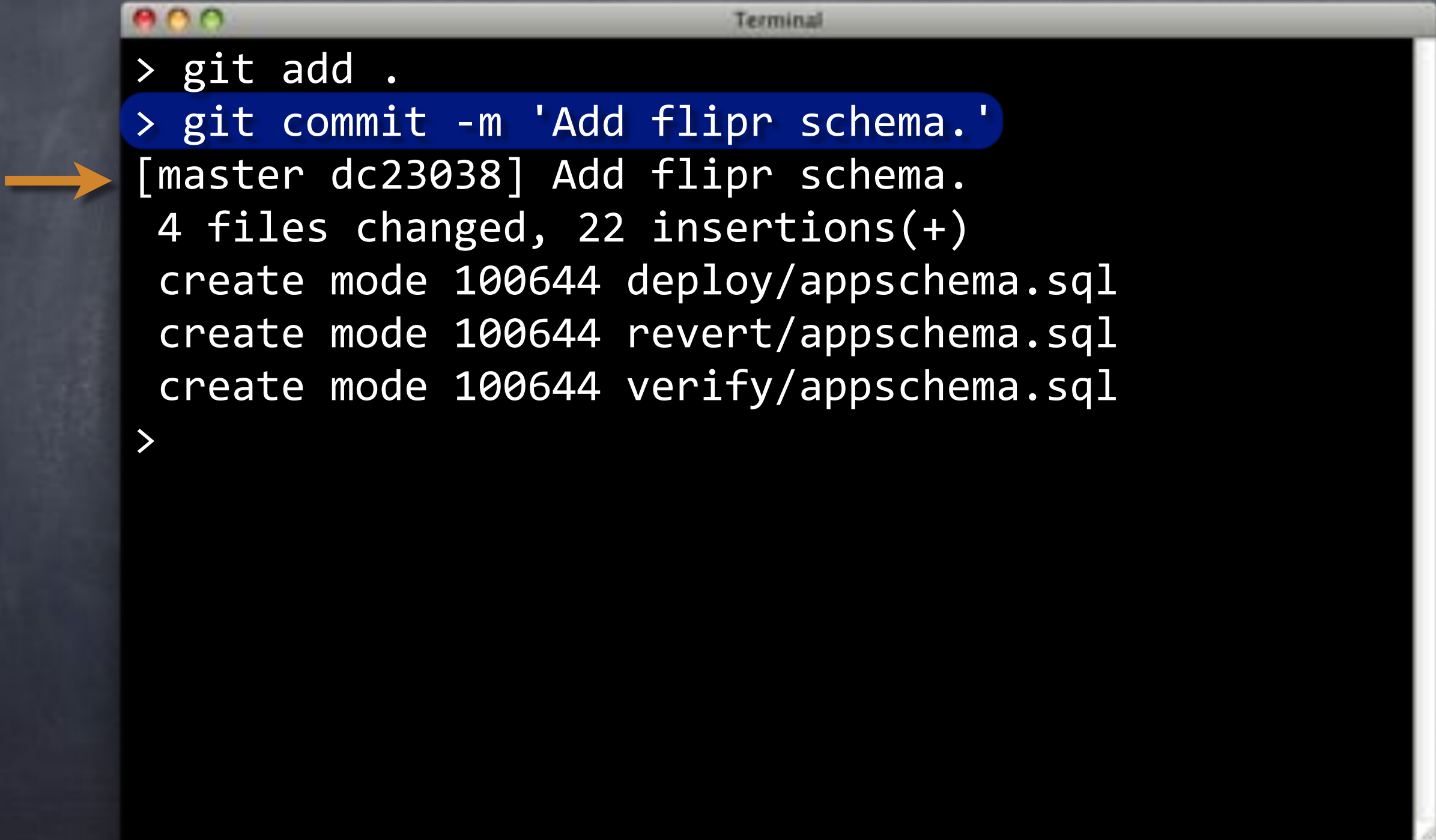
# Commit It!

```
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
>
```



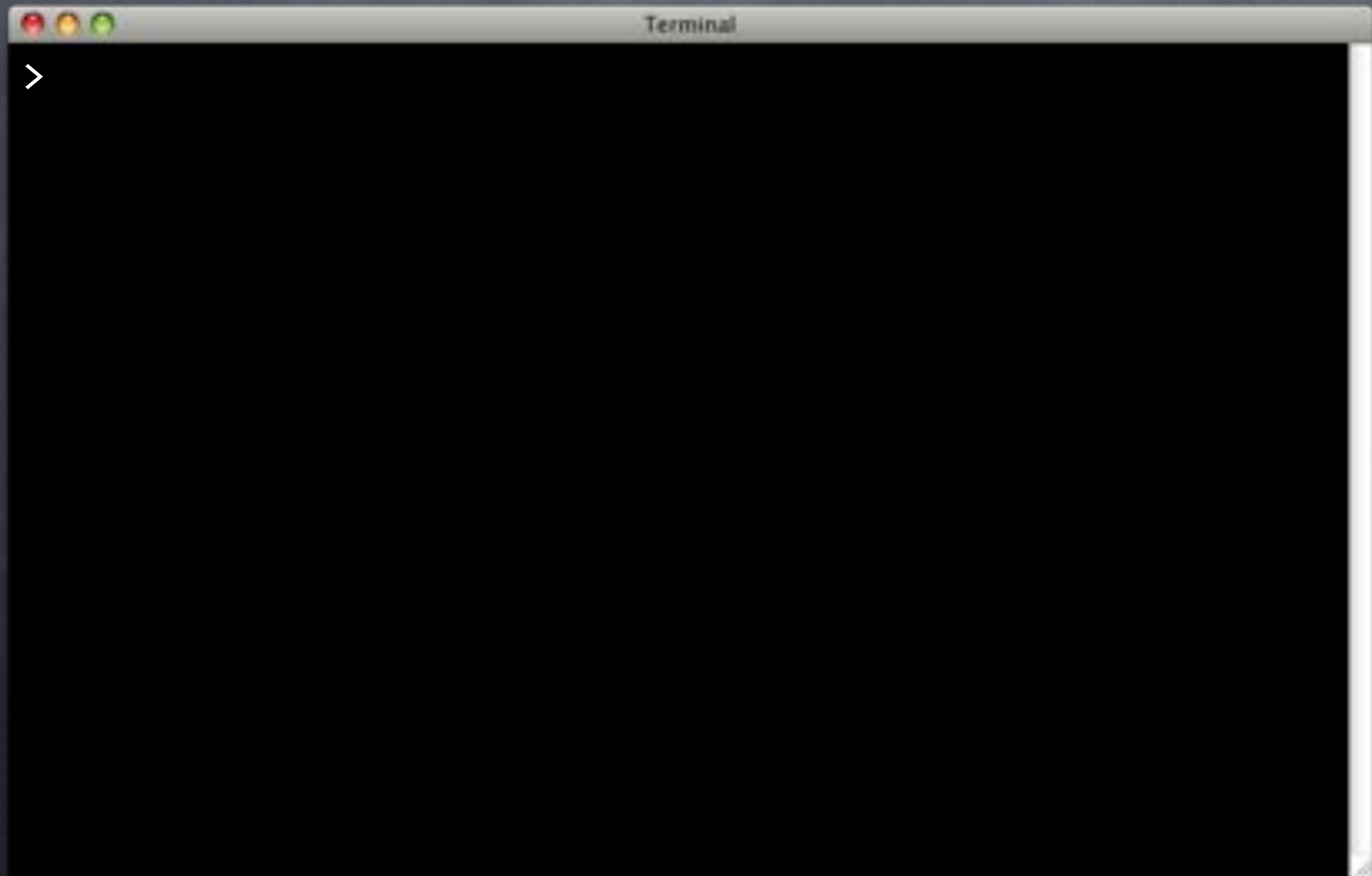
# Commit It!

```
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
```

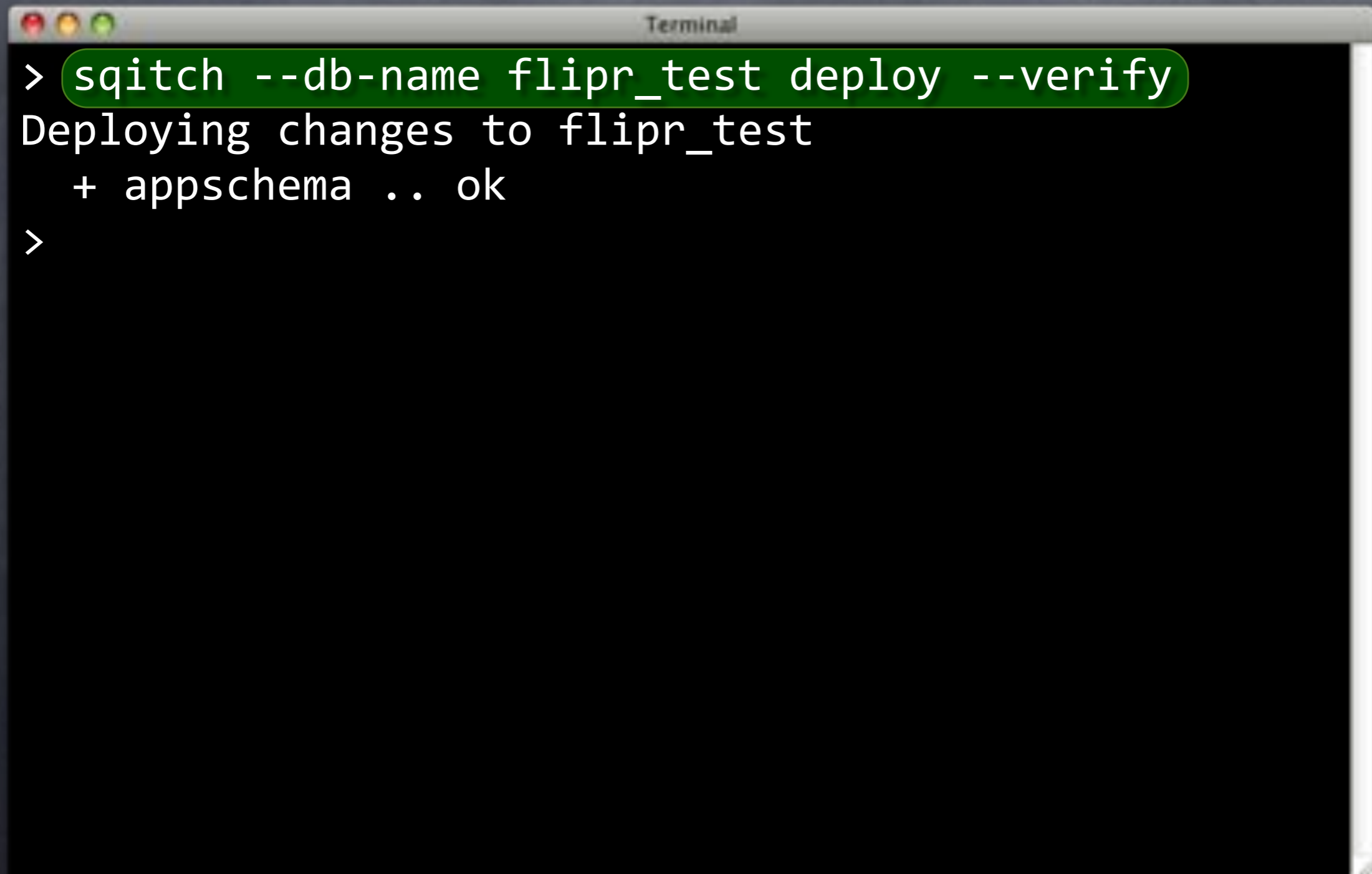
# Commit It!

```
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

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a command being executed and its output. The command is highlighted with a green background.

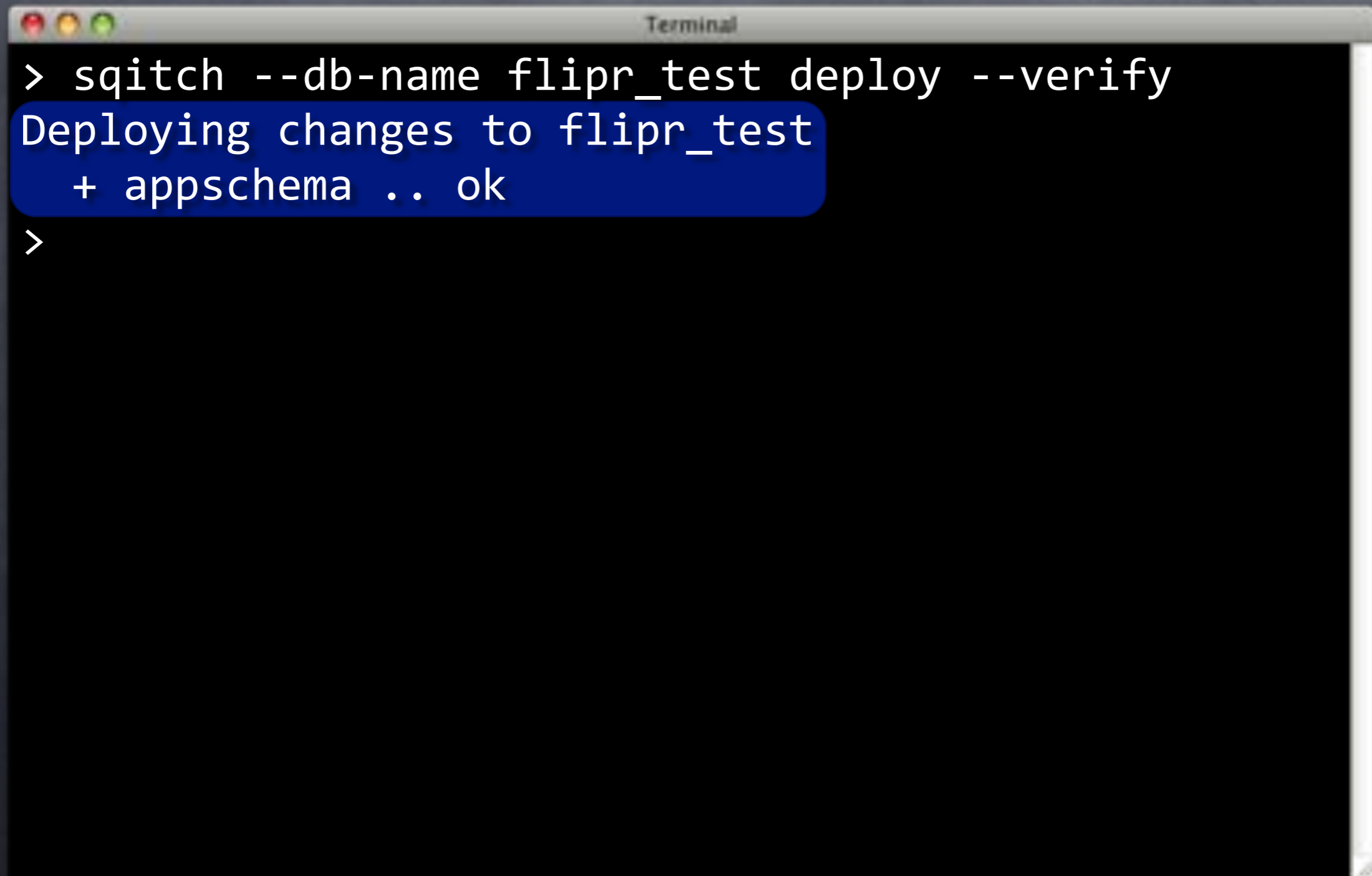
```
> sqitch --db-name flipr_test deploy --verify
Deploying changes to flipr_test
+ appschema .. ok
>
```

# Redeploy

Integrated!

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

# Redeploy

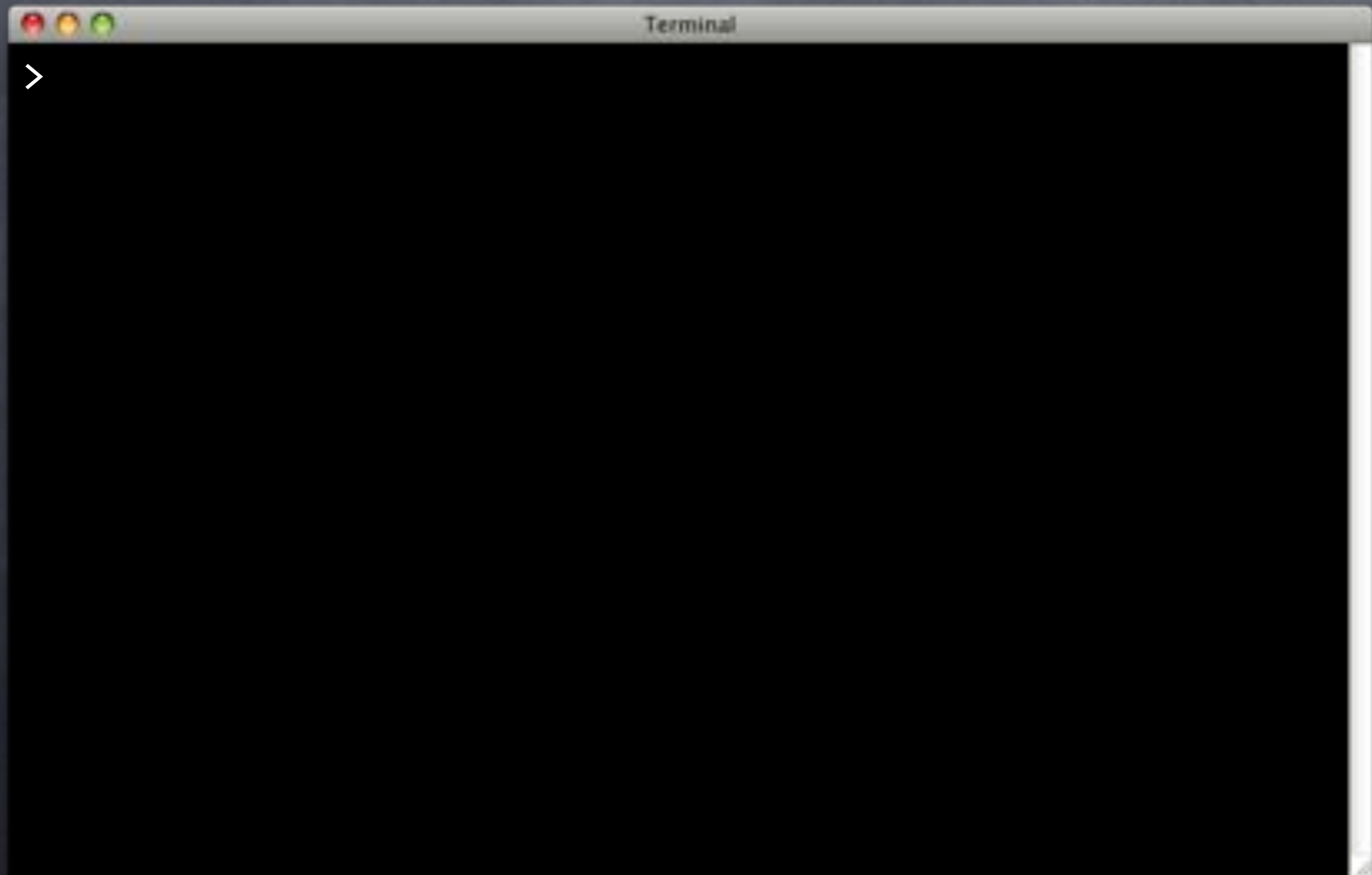
A terminal window titled "Terminal" with a standard macOS window header (red, yellow, green buttons). The terminal shows a command being executed: `> sqitch --db-name flipr_test deploy --verify`. The output is: `Deploying changes to flipr_test` followed by `+ appschema .. ok`. A blue highlight box covers the output text. Below the output, there is a prompt character `>`.

```
> 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

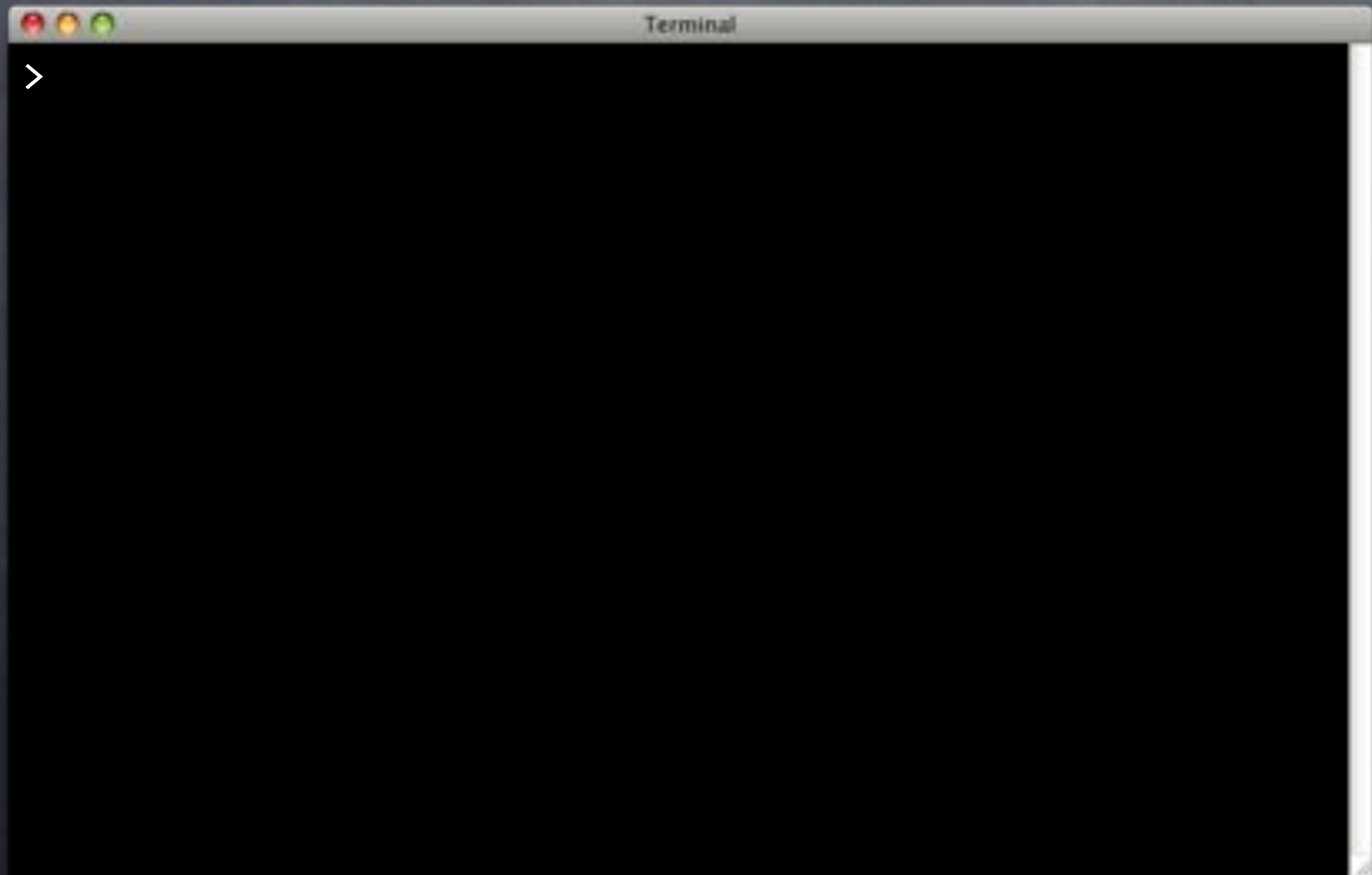




# Status Update

```
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



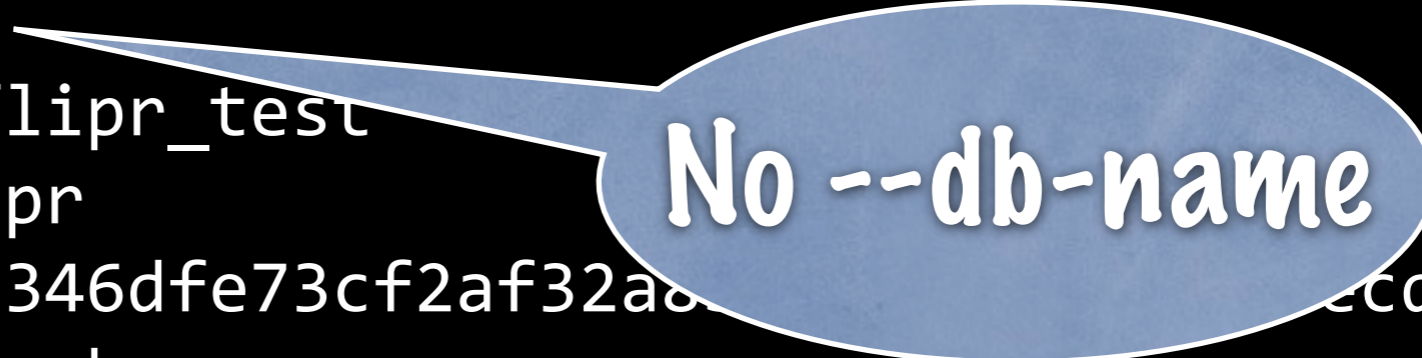
# Save My Fingers

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows a command being entered: "> sqitch config core.pg.db\_name flipr\_test" followed by a blank line and another prompt ">".

```
> sqitch config core.pg.db_name flipr_test
>
```

# Save My Fingers

```
Terminal
> sqitch config core.pg.db_name flipr_test
> sqitch status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a8...ccda3
# 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

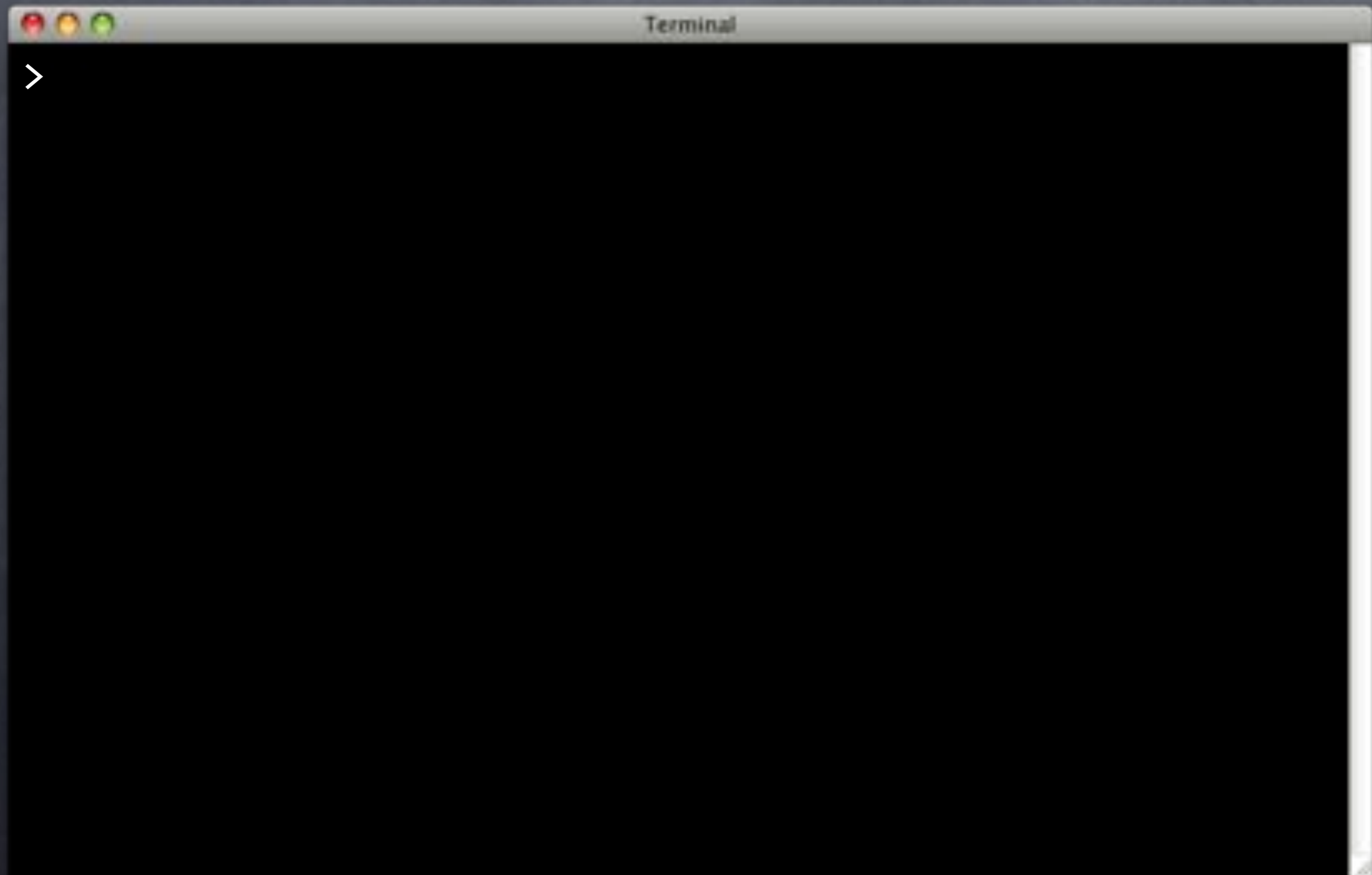
# Save My Fingers

```
Terminal
> sqitch config core.pg.db_name flipr_test
> sqitch status
# On database flipr_test
# Project:  flipr
# Change:   748346dfe73cf2af32a0...ccda3
# 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
>
```

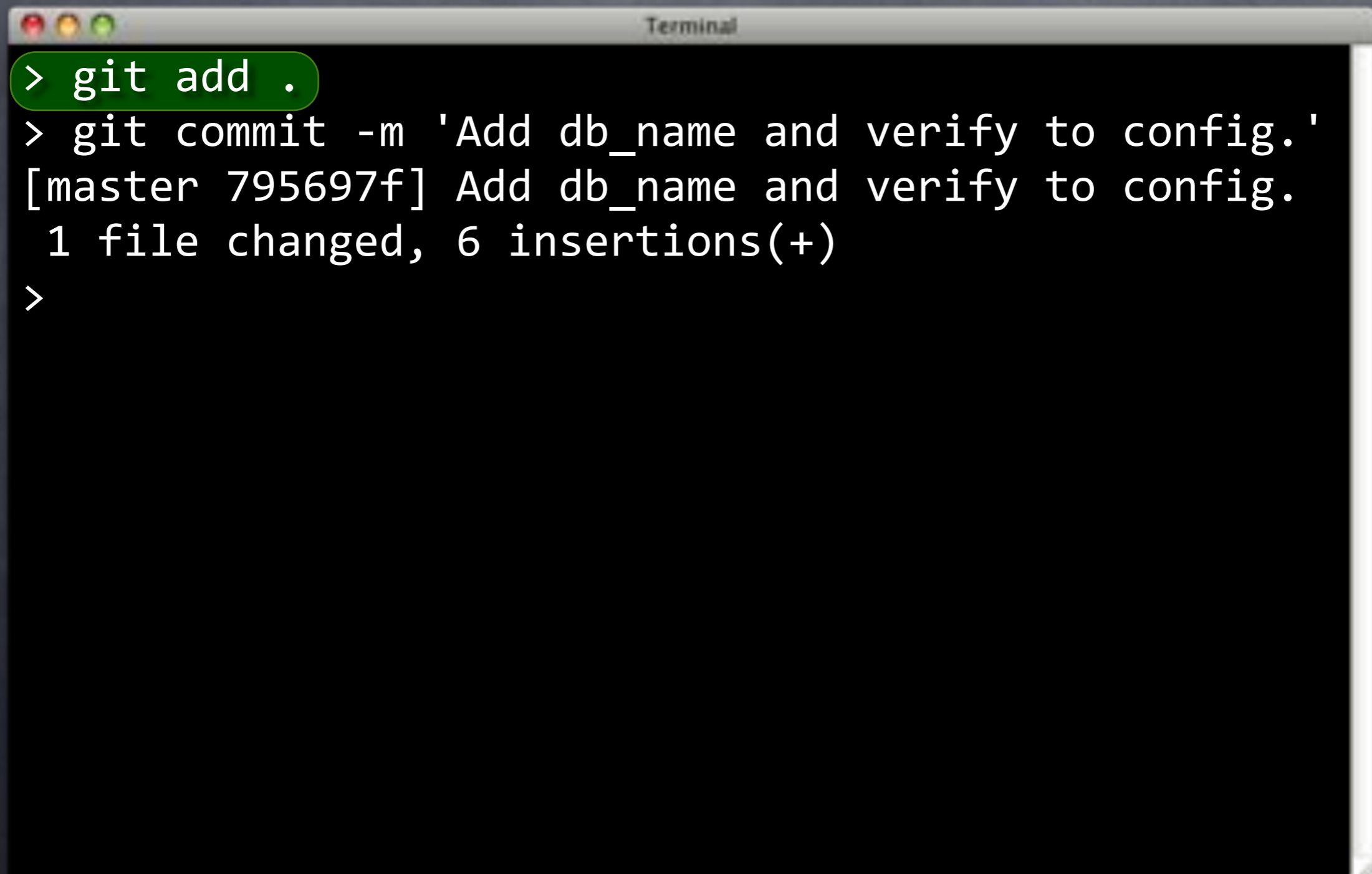
No --db-name

Always verify.

# Commit Config

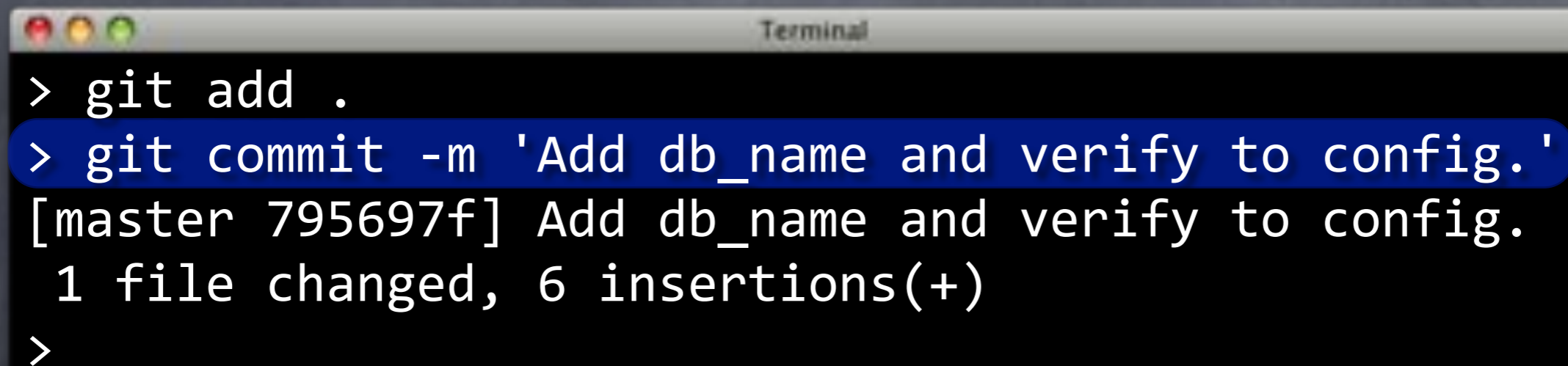


# Commit Config

A terminal window titled "Terminal" with a dark background and light text. The window shows a sequence of git commands and their output. The first command, "git add .", is highlighted with a green background. The second command, "git commit -m 'Add db\_name and verify to config.'", is followed by the output: "[master 795697f] Add db\_name and verify to config. 1 file changed, 6 insertions(+)", and a final prompt character ">".

```
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(+)
>
```

# Commit Config

A terminal window titled "Terminal" with a dark background and white text. The window shows a sequence of git commands and their output. The second command, "git commit -m 'Add db\_name and verify to config.'", is highlighted with a blue background. The output shows the commit hash [master 795697f], the commit message, and the file changes: "1 file changed, 6 insertions(+)".

```
> 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(+)  
>
```



# 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

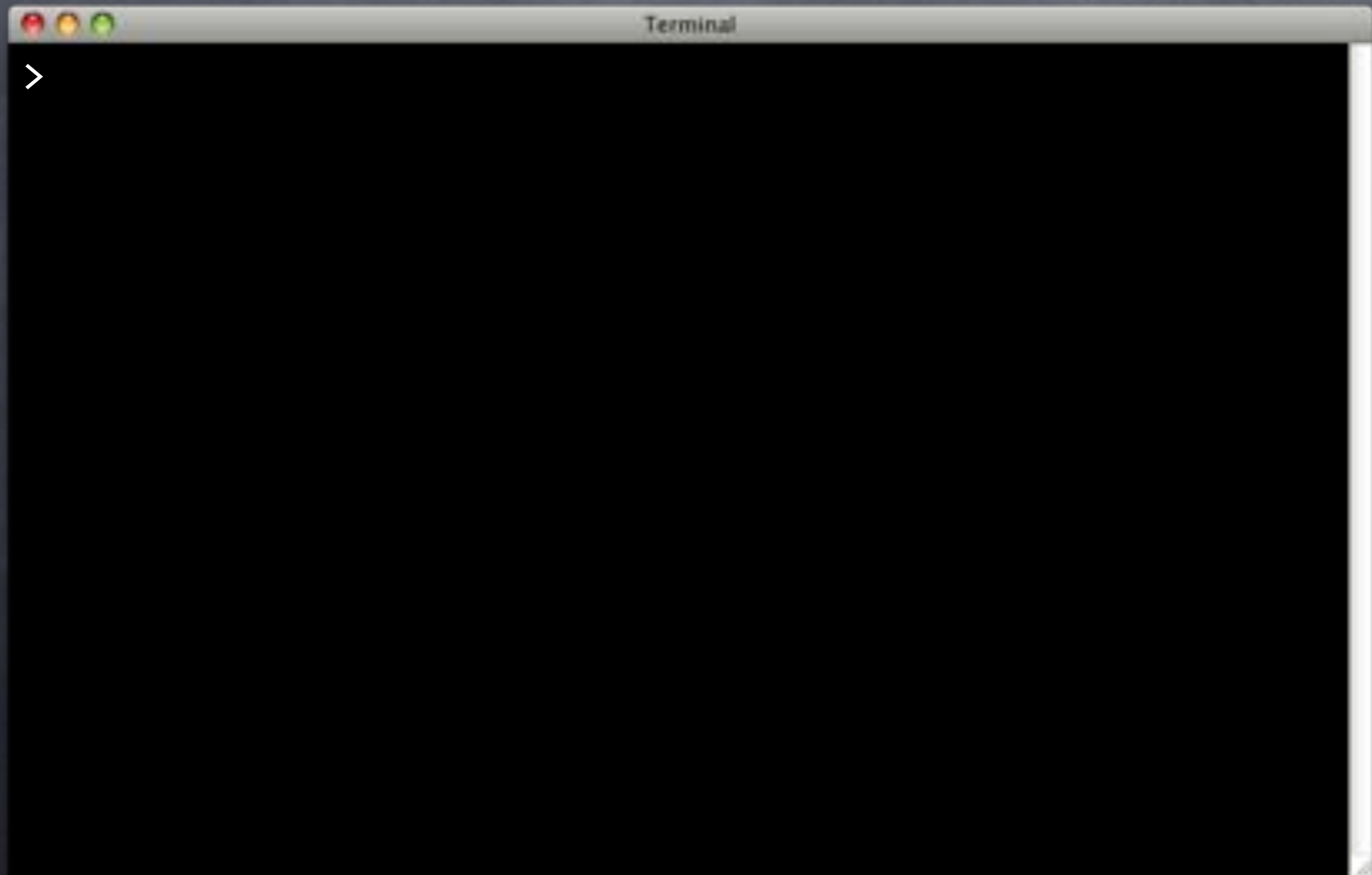


# SHAzbat

- 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.
>
```

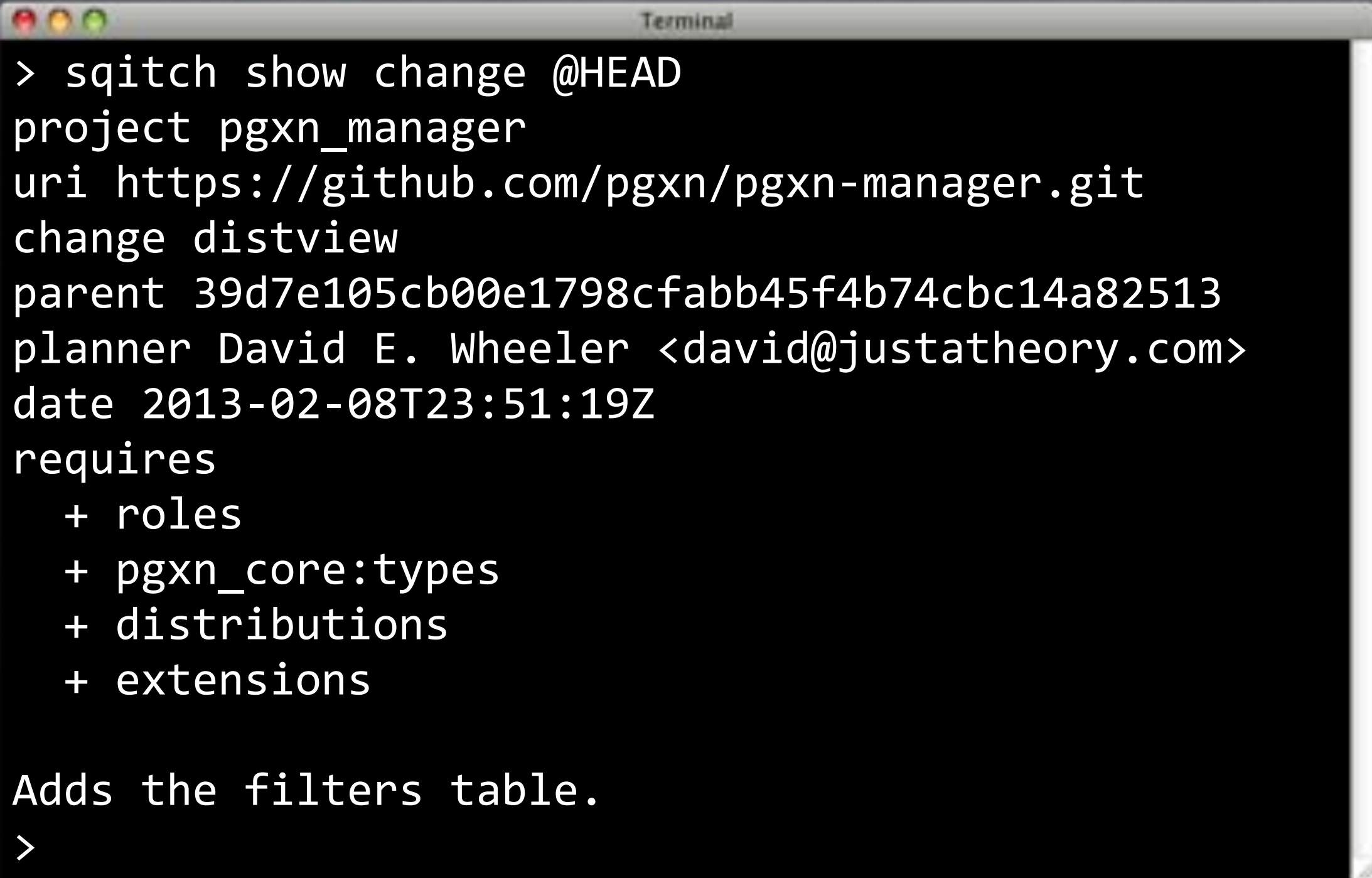
# SHAsome

```
Terminal
> sqitch show change @HEAD
project pgxn_manager
uri https://github.com/pgxn/
change distview
parent 39d7e105cb00e1798cfabb4514074c0c14a82513
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

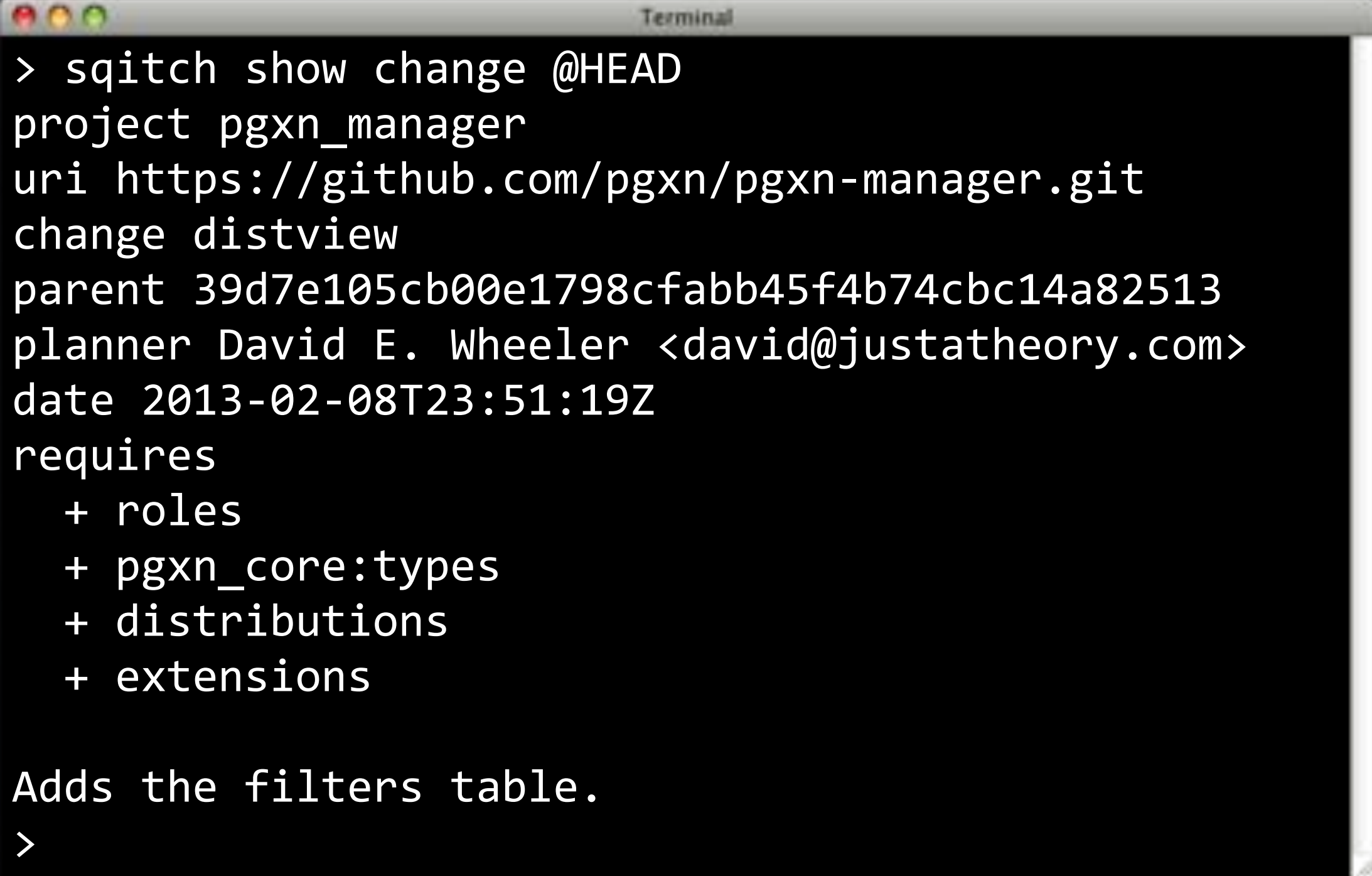
# SHAsome



```
> 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.
>
```

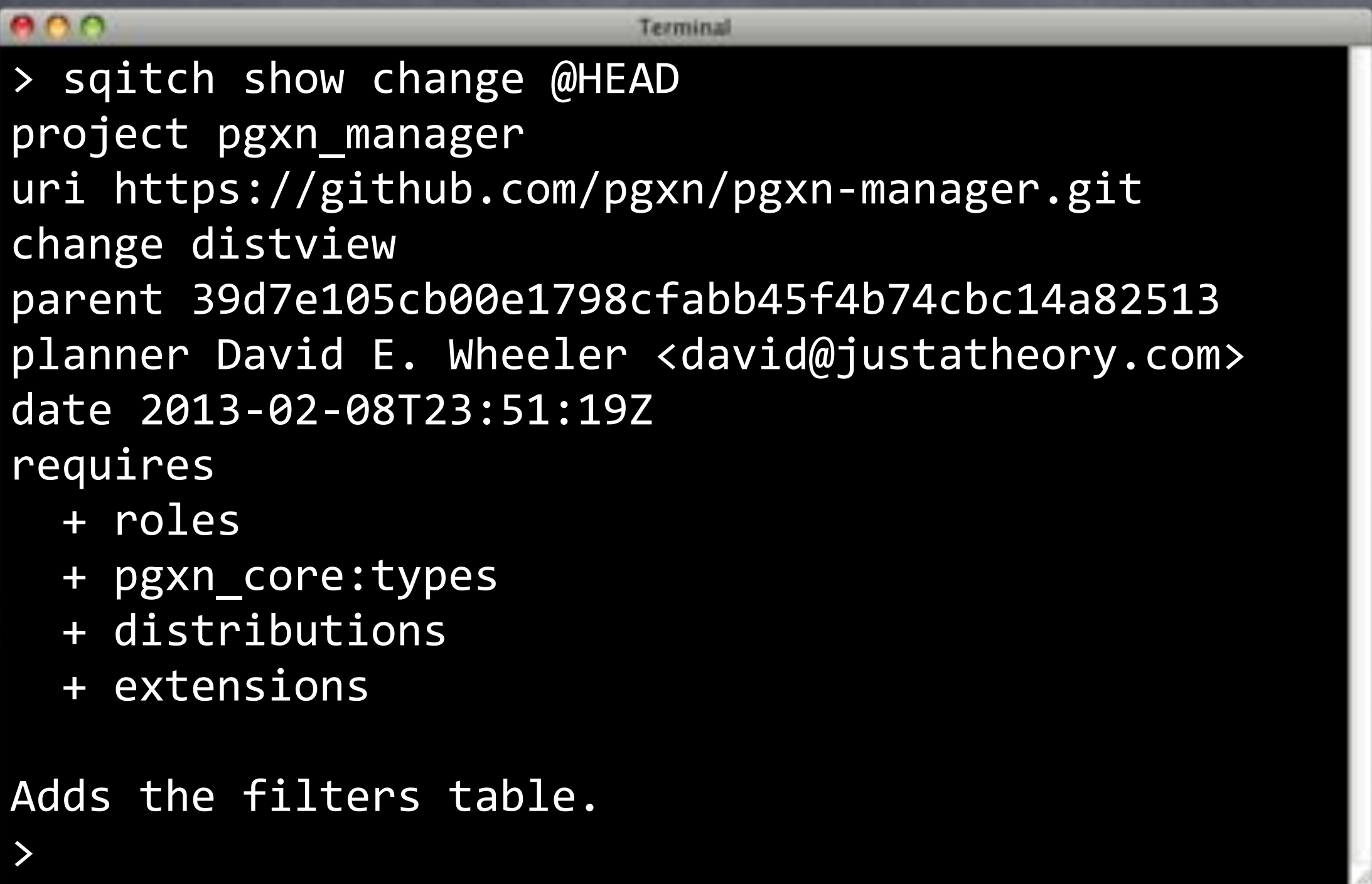
# SHAsome



```
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



```
> 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

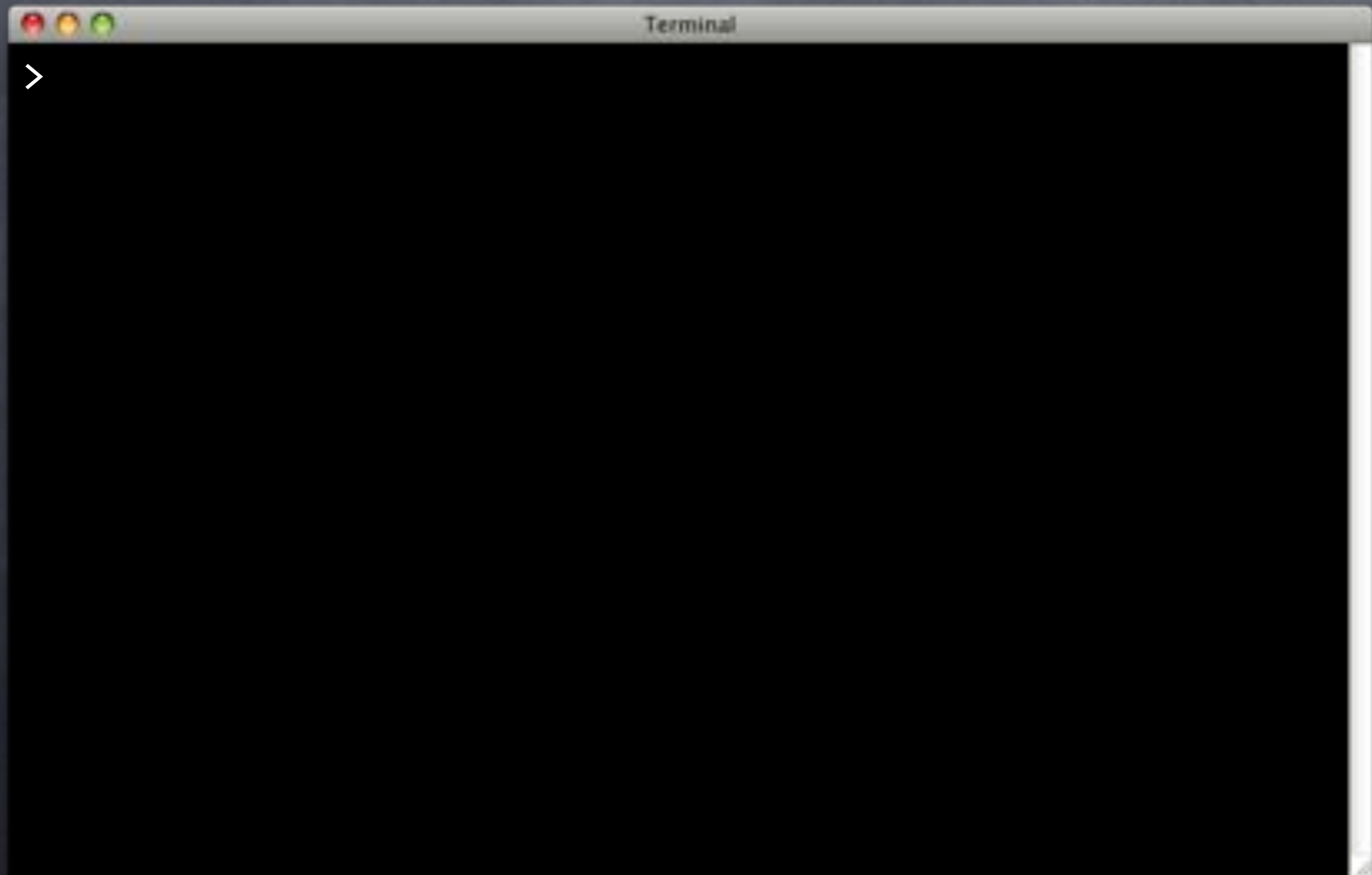


# Your Turn

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

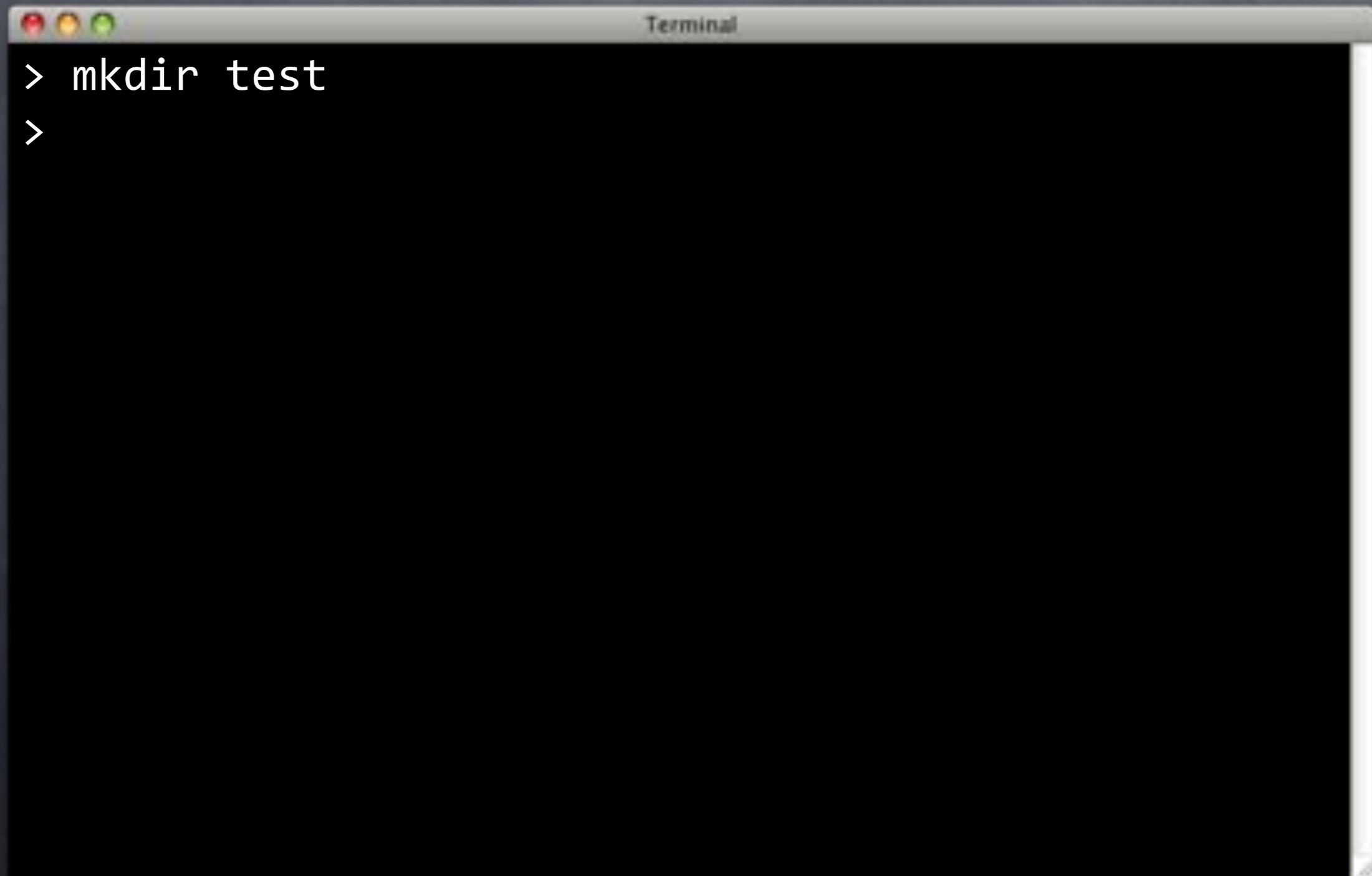


# Add Tests



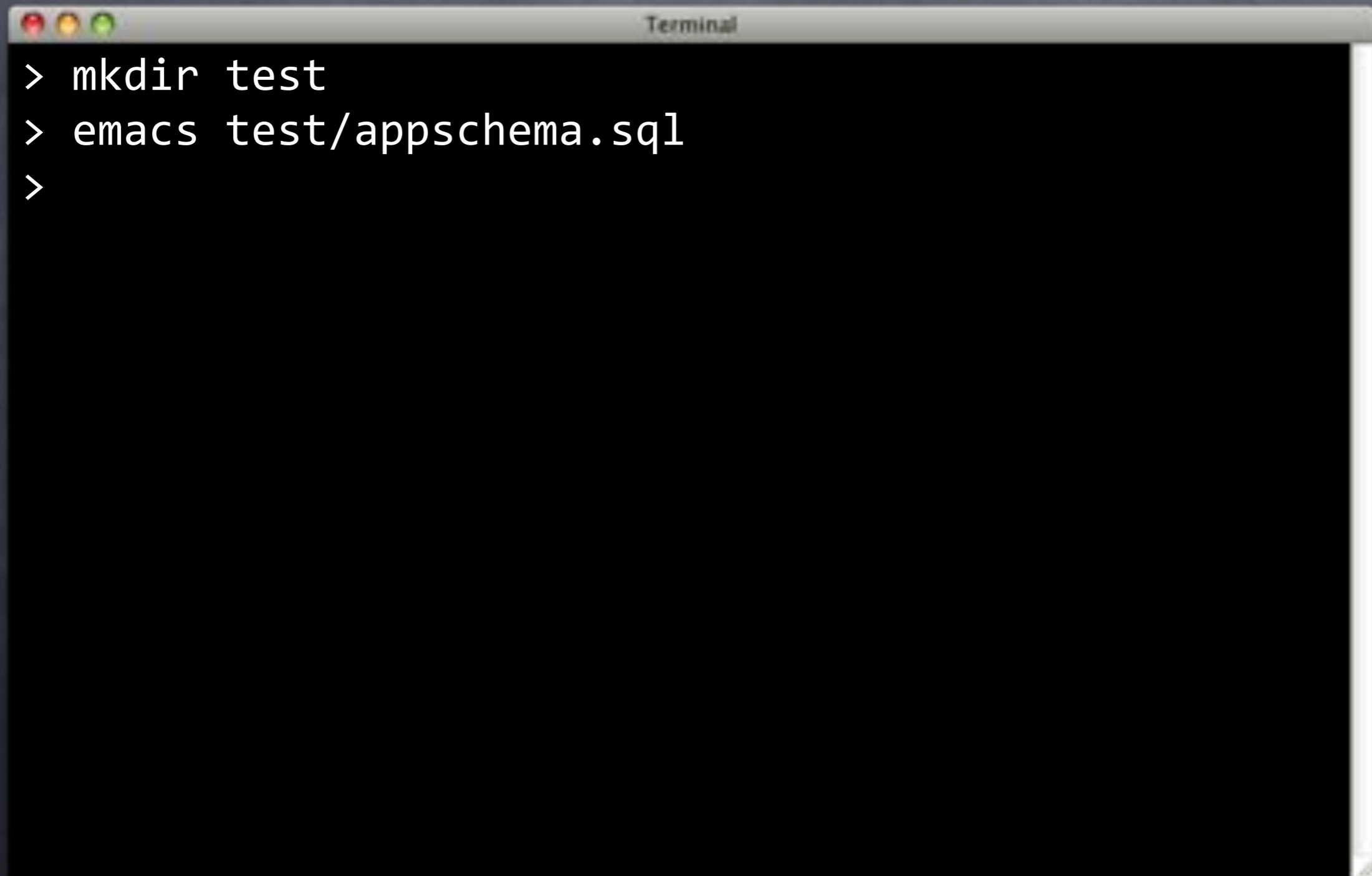


# Add Tests

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a prompt character ">" followed by the command "mkdir test" on the first line, and a second prompt character ">" on the second line, indicating the command has been executed and the prompt is ready for the next input.

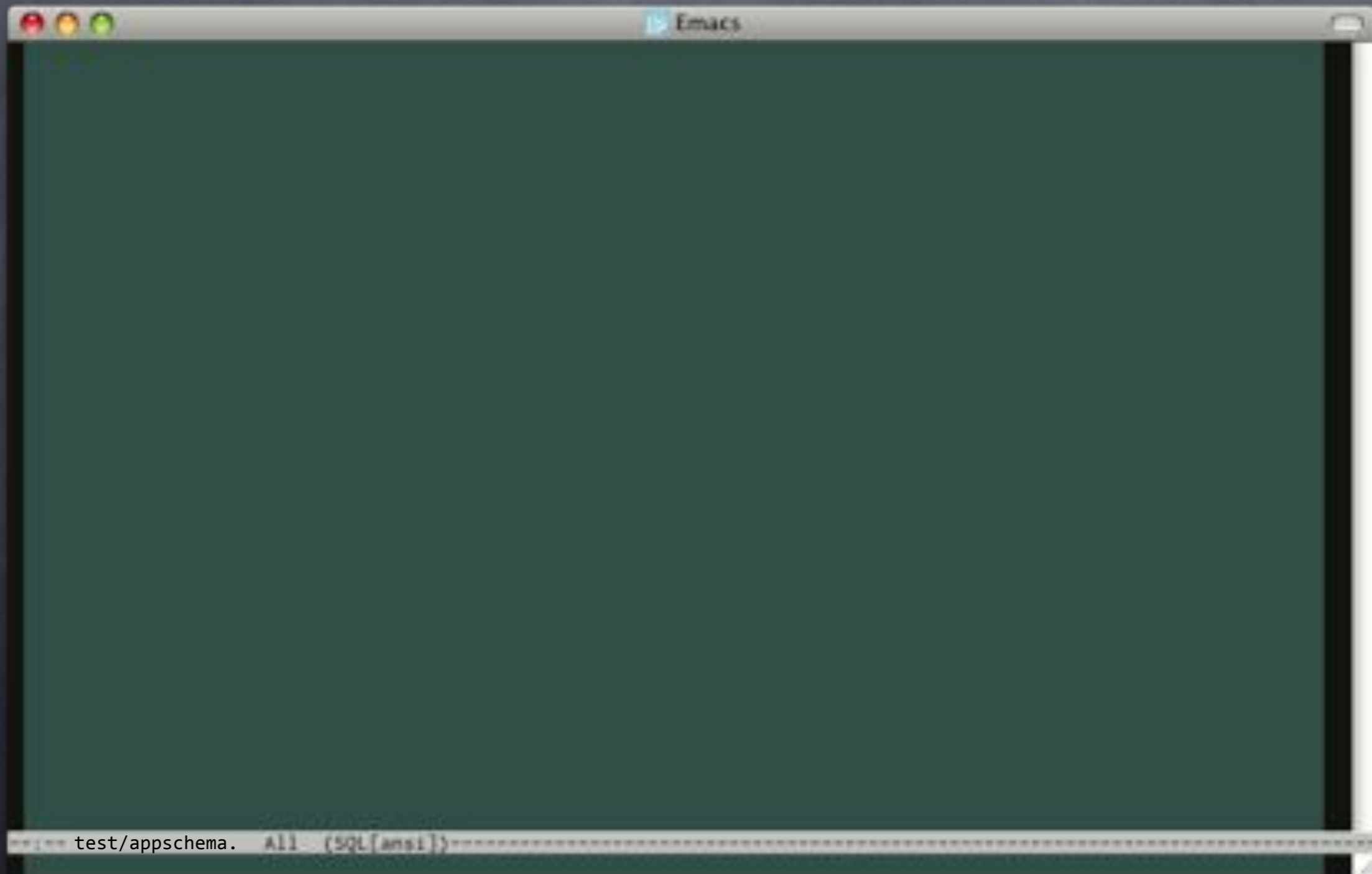
```
> mkdir test  
>
```

# Add Tests

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows three lines of shell commands: a prompt followed by "mkdir test", a prompt followed by "emacs test/appschema.sql", and a final prompt character ">".

```
> mkdir test
> emacs test/appschema.sql
>
```

# pgTAP Basics



# pgTAP Basics

```
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

```
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

```
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

```
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

```
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

```
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

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

BEGIN;

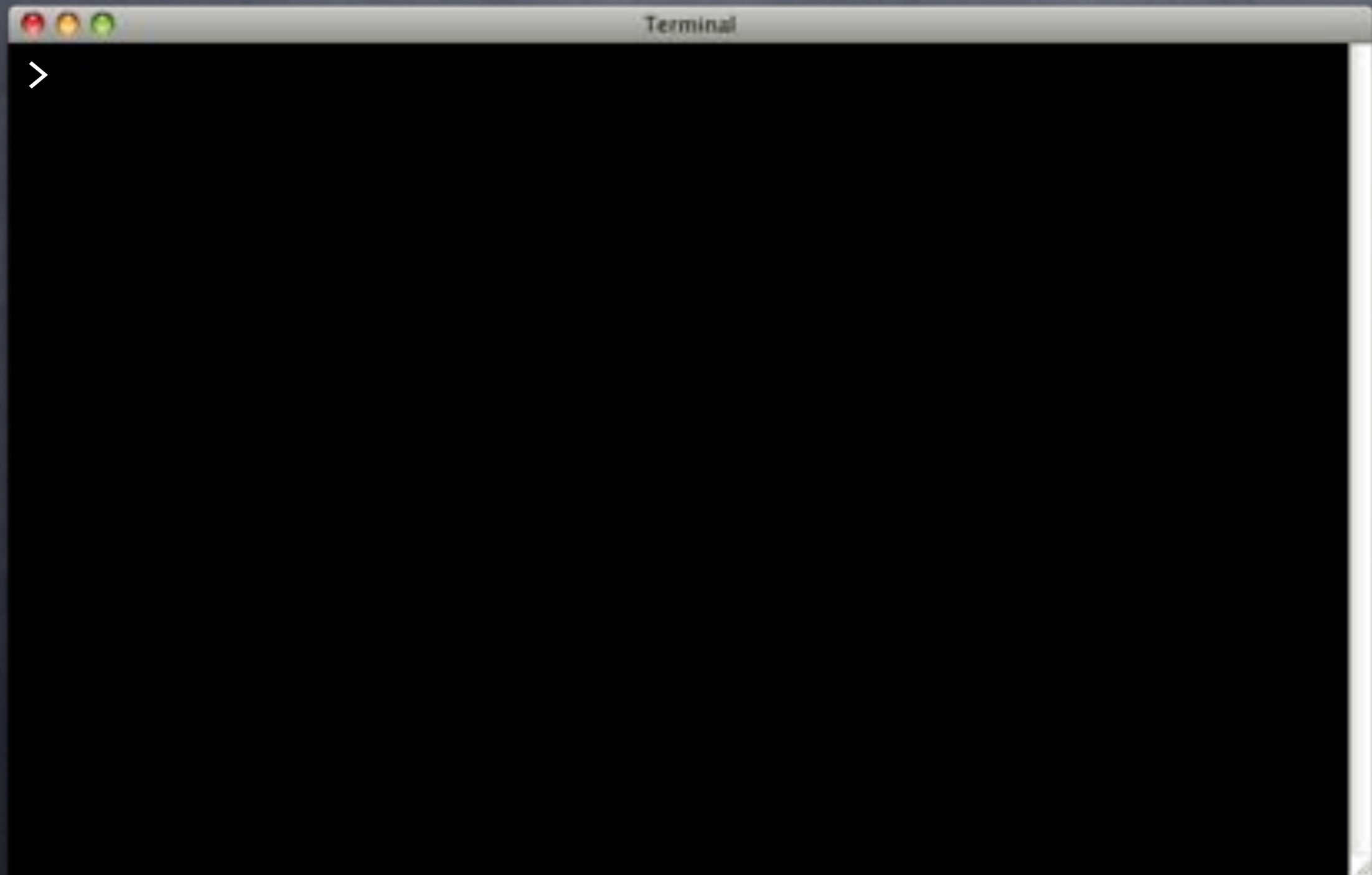
-- Plan the tests.
SELECT plan(1);

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

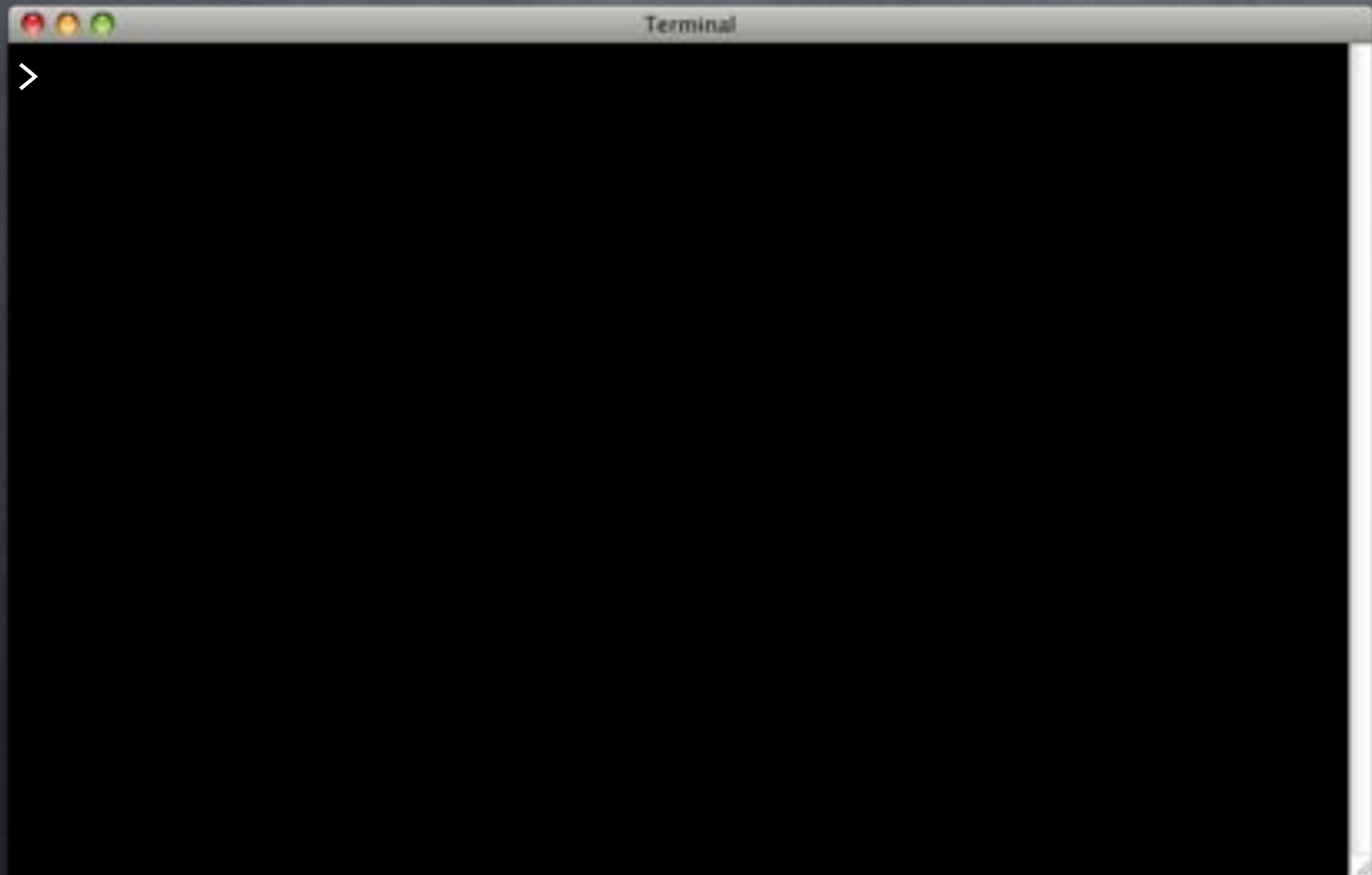
Test Summary Report
-----
test/appschema.sql (Wstat: 0 Tests: 1 Failed: 1)
  Failed test: 1
Files=1, Tests=1, 0 wallclock secs
Result: FAIL
```

# Run the Test

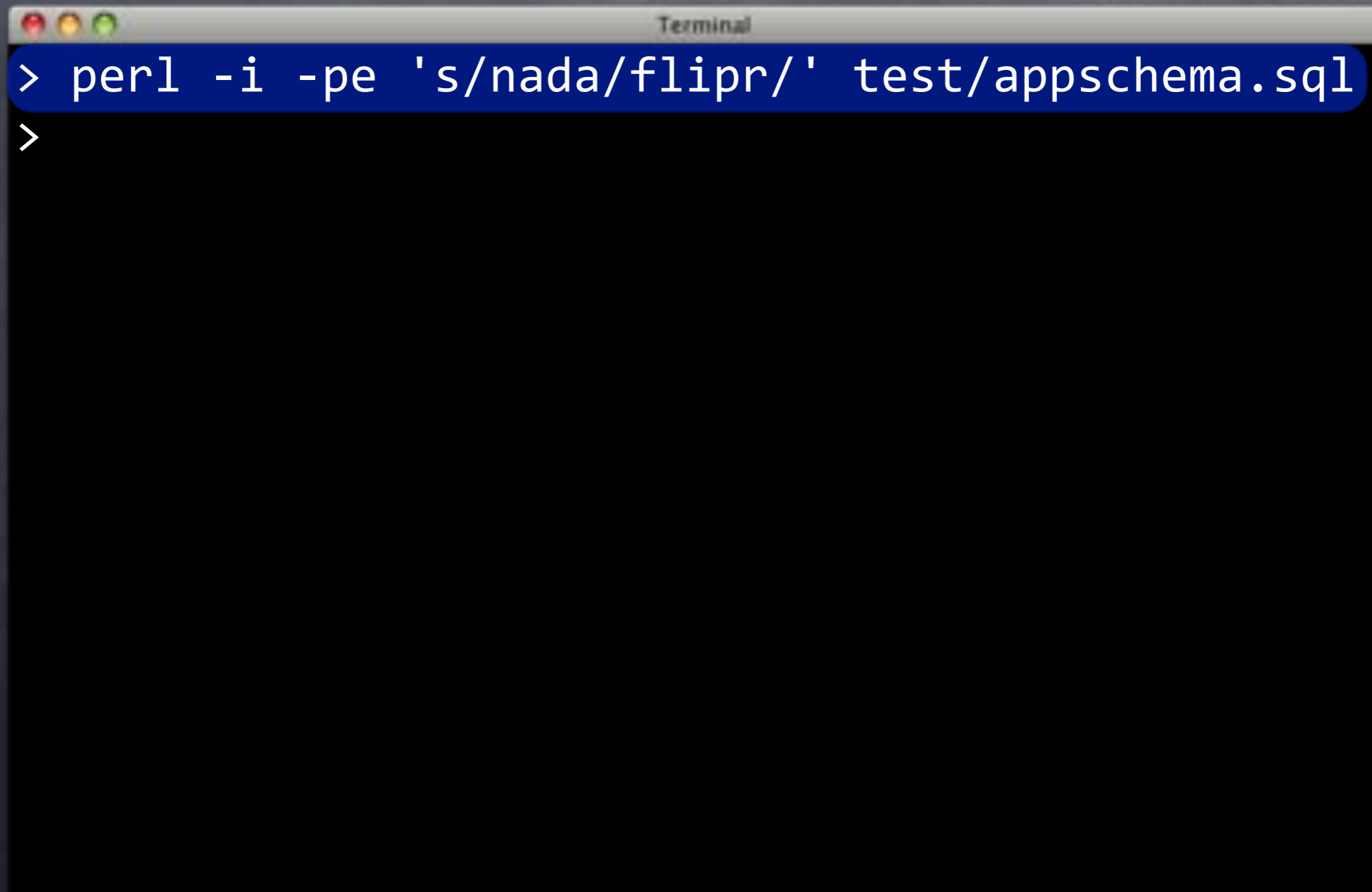
```
Terminal
> pg_prove -v -d flipr_test test/appschema.sql
test/appschema.sql ..
1..1
not ok 1 - Schema nada should exist
# Failed test 1: "Schema nada should exist"
# Looks like you failed 1 test of 1
Failed 1/1 subtests

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

A terminal window titled "Terminal" with a dark background and a light gray title bar. The window contains a single line of text: `> perl -i -pe 's/nada/flipr/' test/appschemata.sql`. The text is highlighted in a blue selection box. Below the highlighted line, there is a white prompt character `>` on the next line.

```
Terminal  
> perl -i -pe 's/nada/flipr/' test/appschemata.sql  
>
```



# First Pass

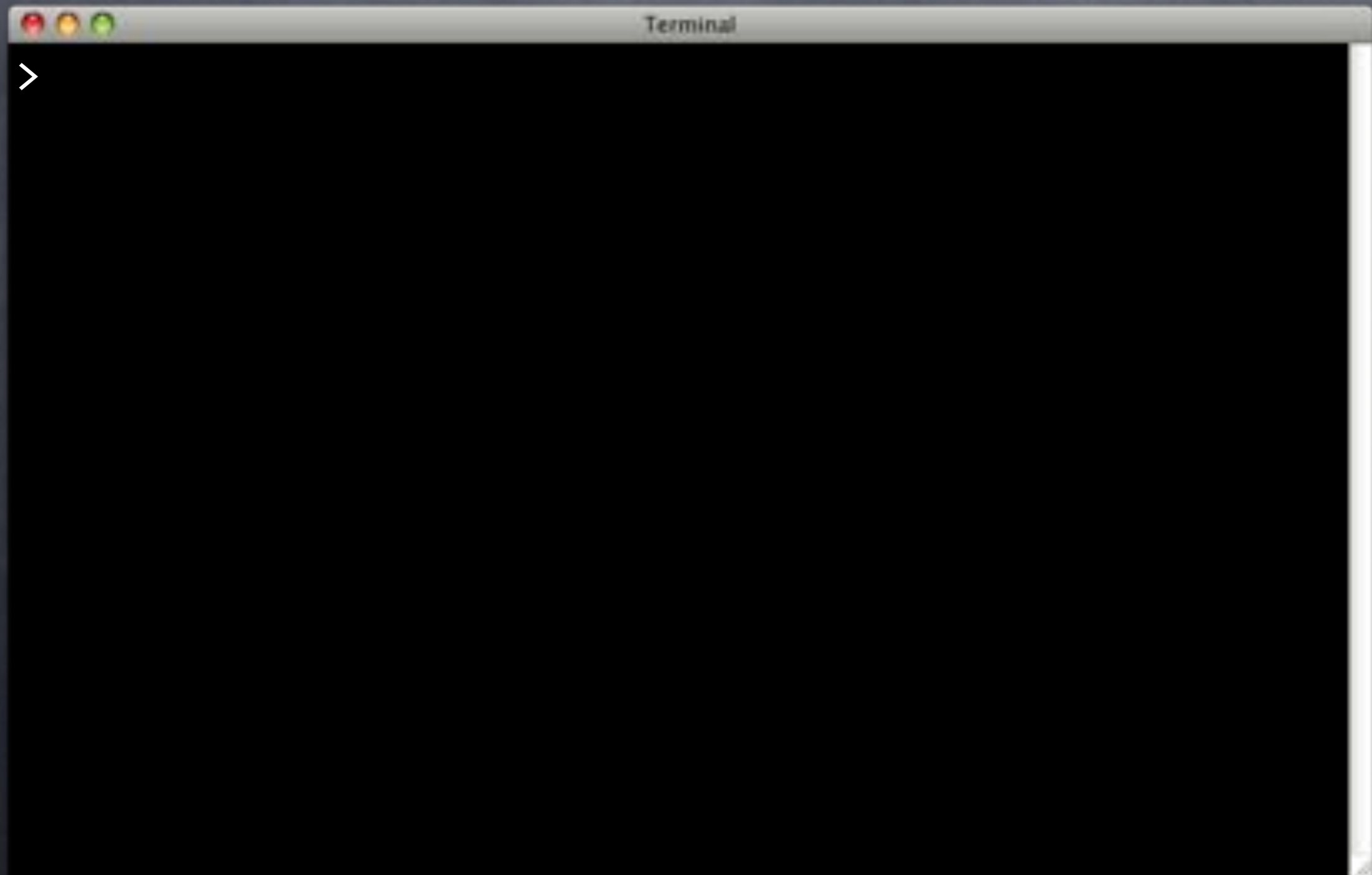
```
Terminal
> perl -i -pe 's/nada/flipr/' test/appschemata.sql
> pg_prove -v -d flipr_test test/appschemata.sql
test/appschemata.sql ..
1..1
ok 1 - Schema flipr should exist
ok
All tests successful.
Files=1, Tests=1, 0 wallclock secs
Result: PASS
```

# First Pass

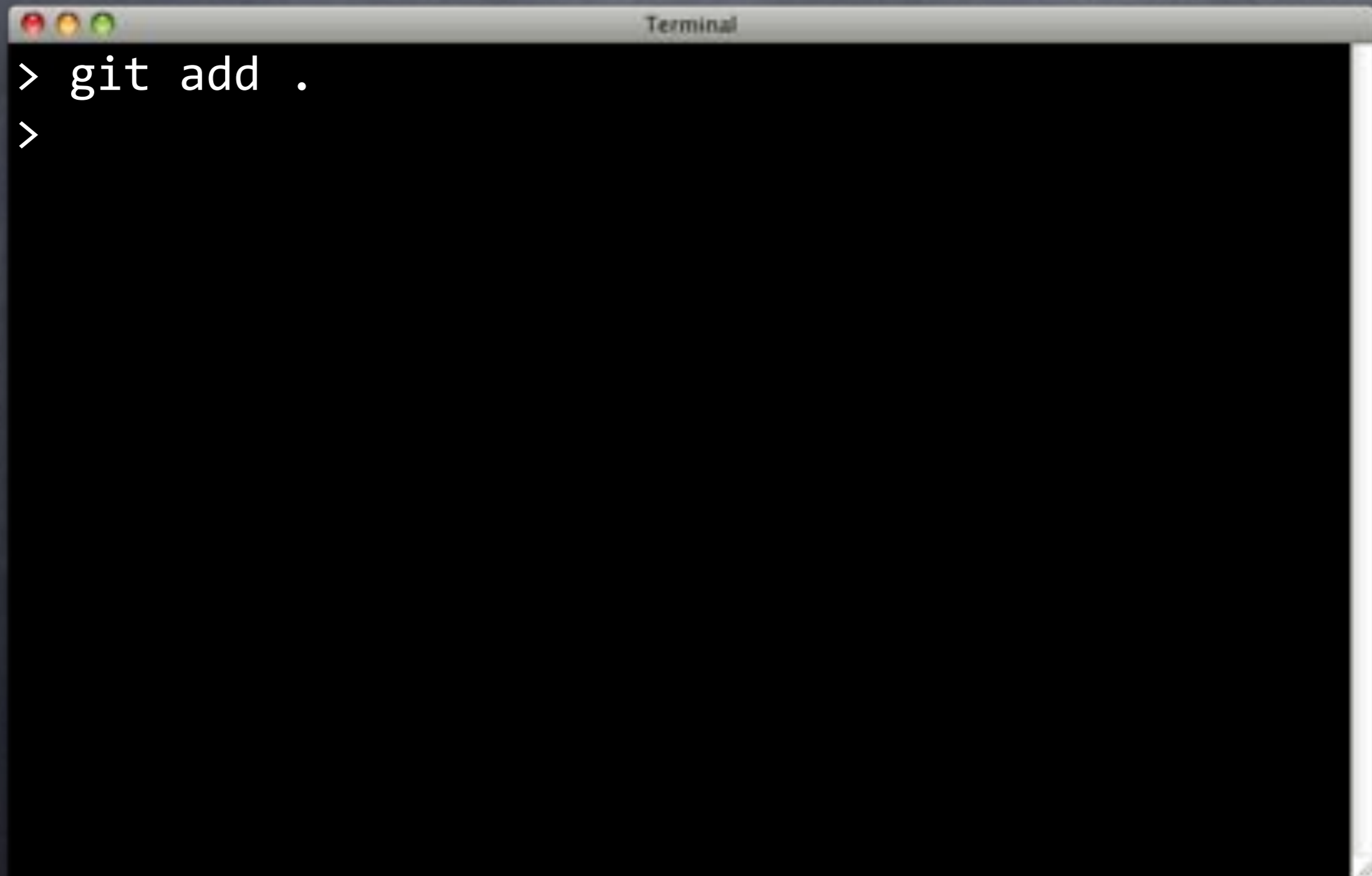
```
Terminal
> perl -i -pe 's/nada/flipr/' test/appschemata.sql
> pg_prove -v -d flipr_test test/appschemata.sql
test/appschemata.sql ..
1..1
ok 1 - Schema flipr should exist
ok
All tests successful.
Files=1, Tests=1, 0 wallclock secs
Result: PASS
```

WOOT!

# Pass it On



# Pass it On

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a prompt character ">" followed by the command "git add ." on the first line, and a second prompt character ">" on the second line, indicating the command has been entered and the terminal is waiting for further input.

```
> git add .  
>
```

# 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...





# TOD





**“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**



TDD is an act  
of design.







**TDD is an act of  
documentation.**





Okay.





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

Big jump!



# Database Design

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

Pricing 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?”**



Okay.





# 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

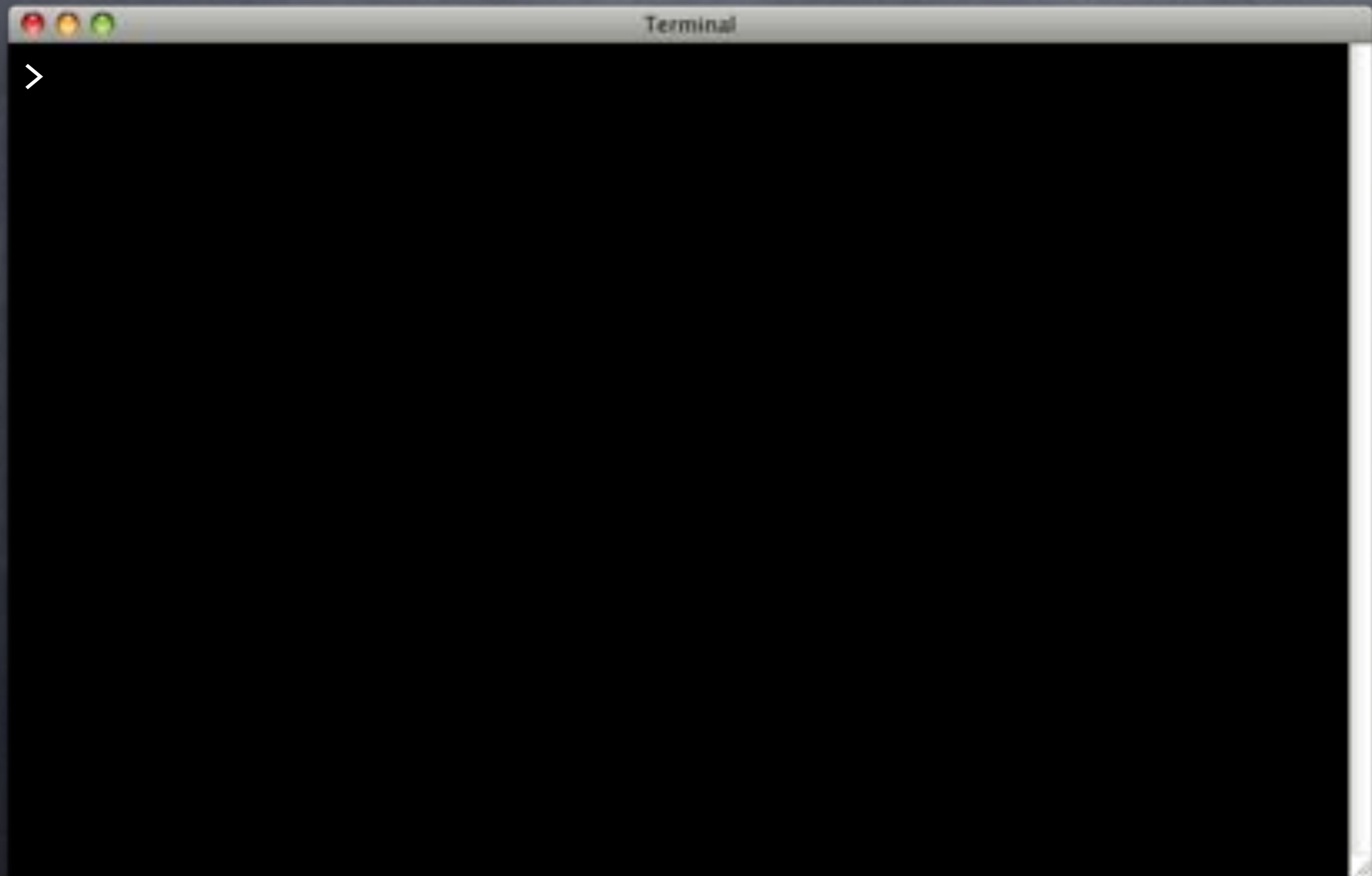


# TDD How

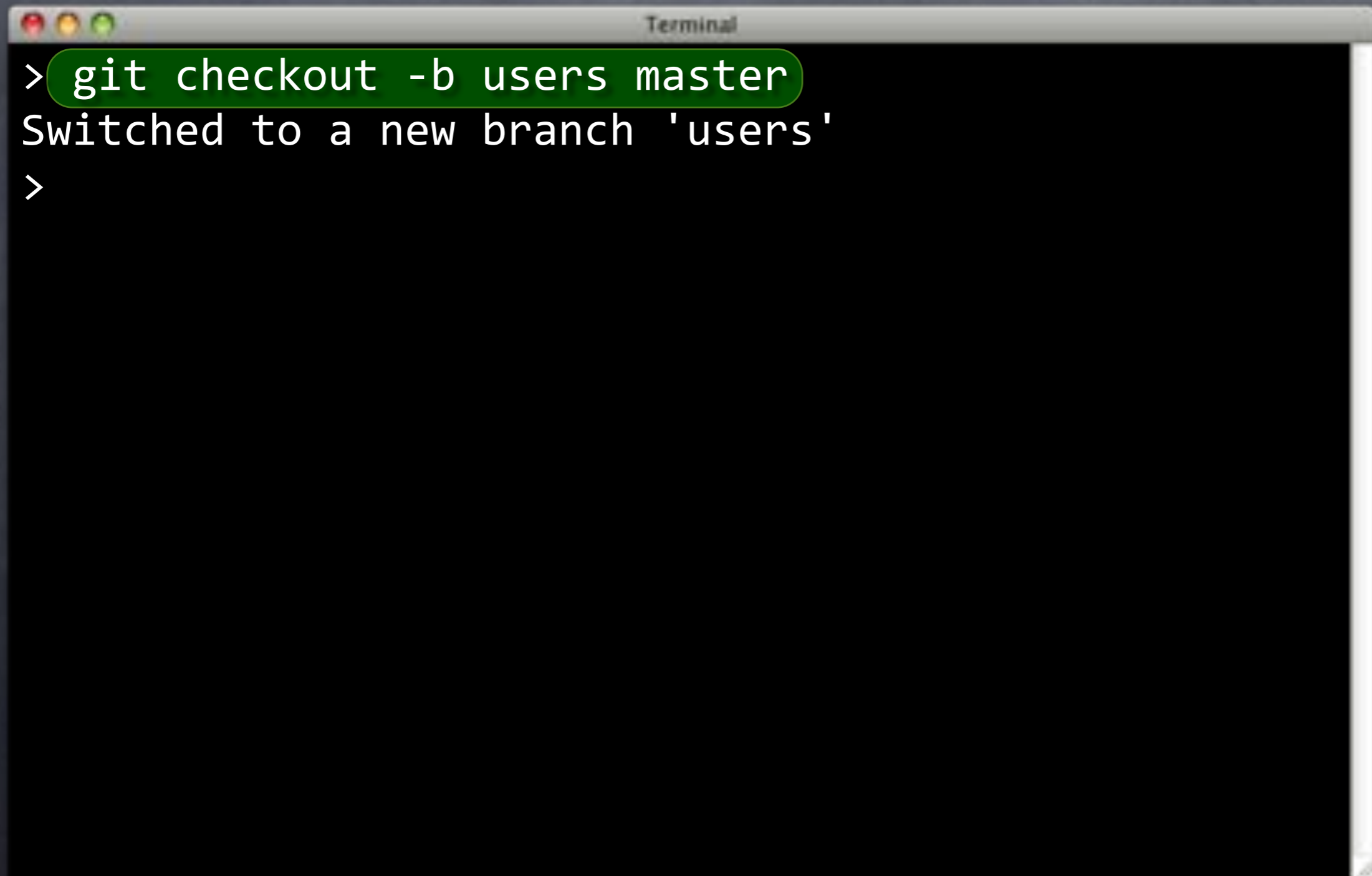
- Ideally separate from app tests
  - May be many apps
  - DB should present interface to all
  - Apps may use different permissions
- Ideally use DB test Framework
- Like...pgTAP!



# Branching Out



# Branching Out

A terminal window titled "Terminal" with a dark background. The command `git checkout -b users master` is entered and highlighted in green. The output is "Switched to a new branch 'users'", followed by a prompt character `>`.

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

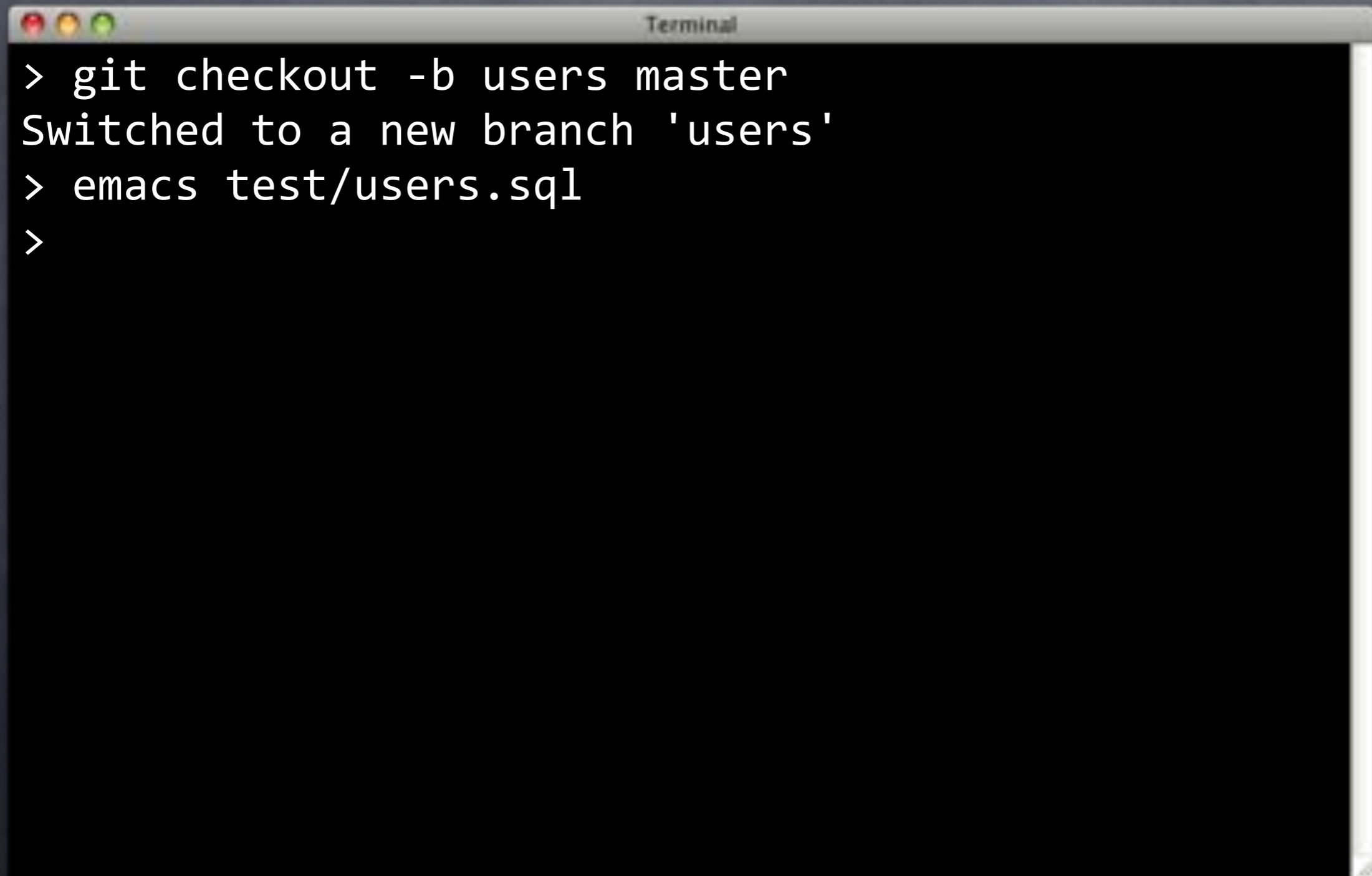
# Branching Out

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

Branching off  
from others



# Branching Out

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a sequence of commands: a git checkout command to create a new branch, a confirmation message, and an emacs command to open a file.

```
> git checkout -b users master
Switched to a new branch 'users'
> emacs test/users.sql
>
```

# Table For One



# Table For One

```
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;
```

# Table For One

```
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;
```

# Table For One

```
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;
```

# Table For One

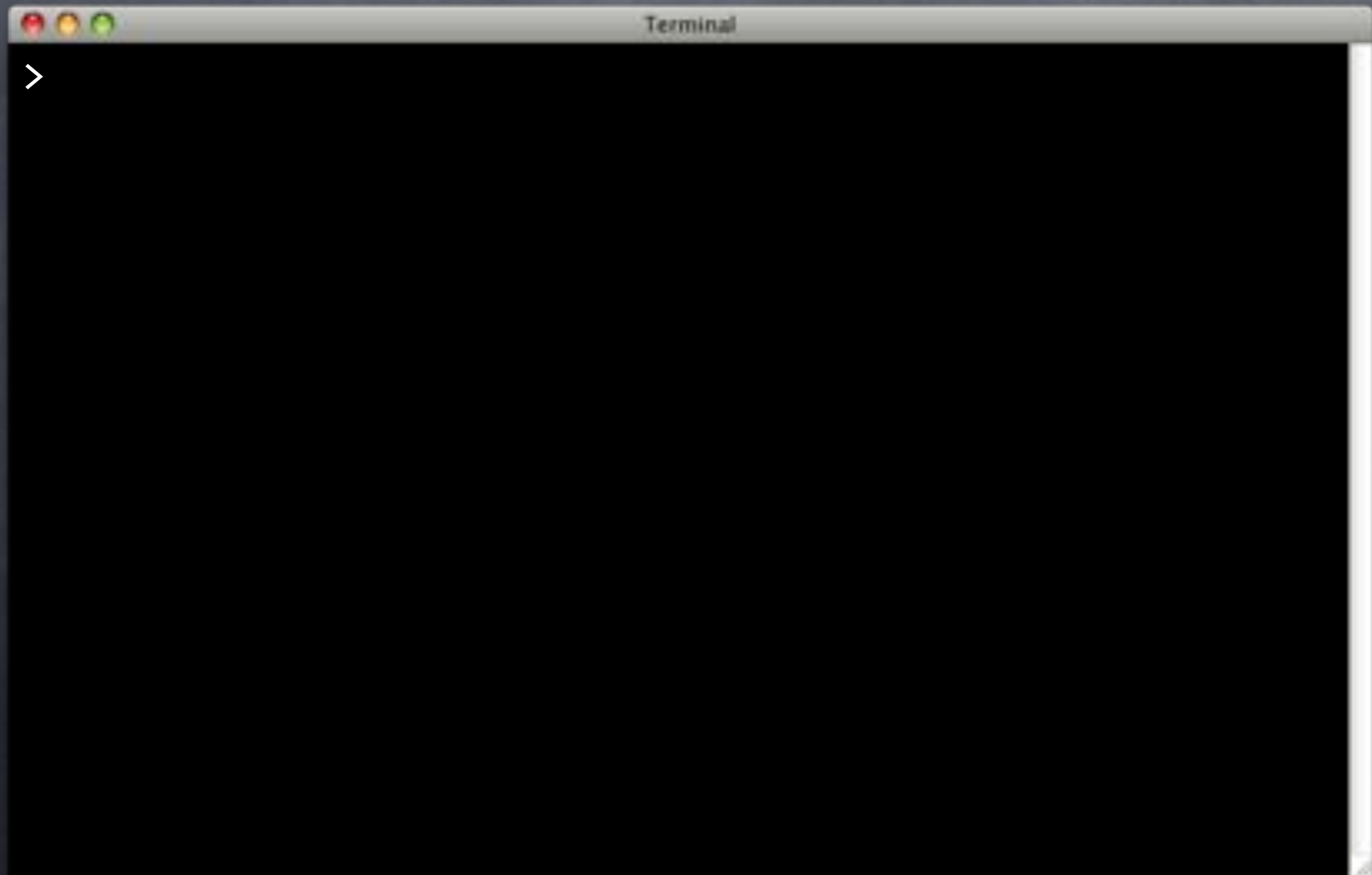
```
SET client_min_messages TO warning;
CREATE EXTENSION IF NOT EXISTS pgtap;
RESET client_min_messages;

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

SELECT has_table( 'users' );

SELECT finish();
ROLLBACK;
```

# 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
>
```

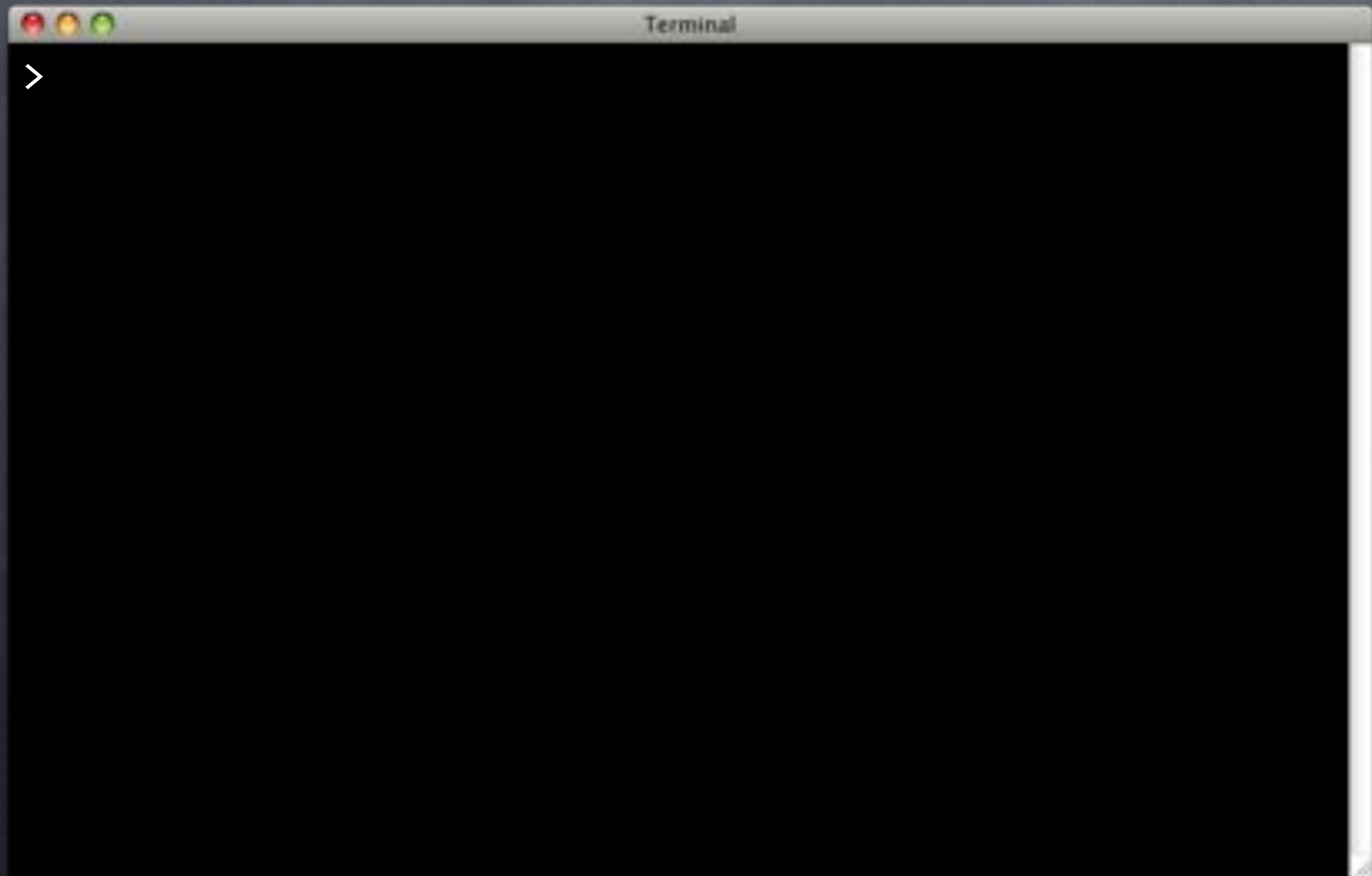
# Run 'Em

```
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, 1 failure(s)
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

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

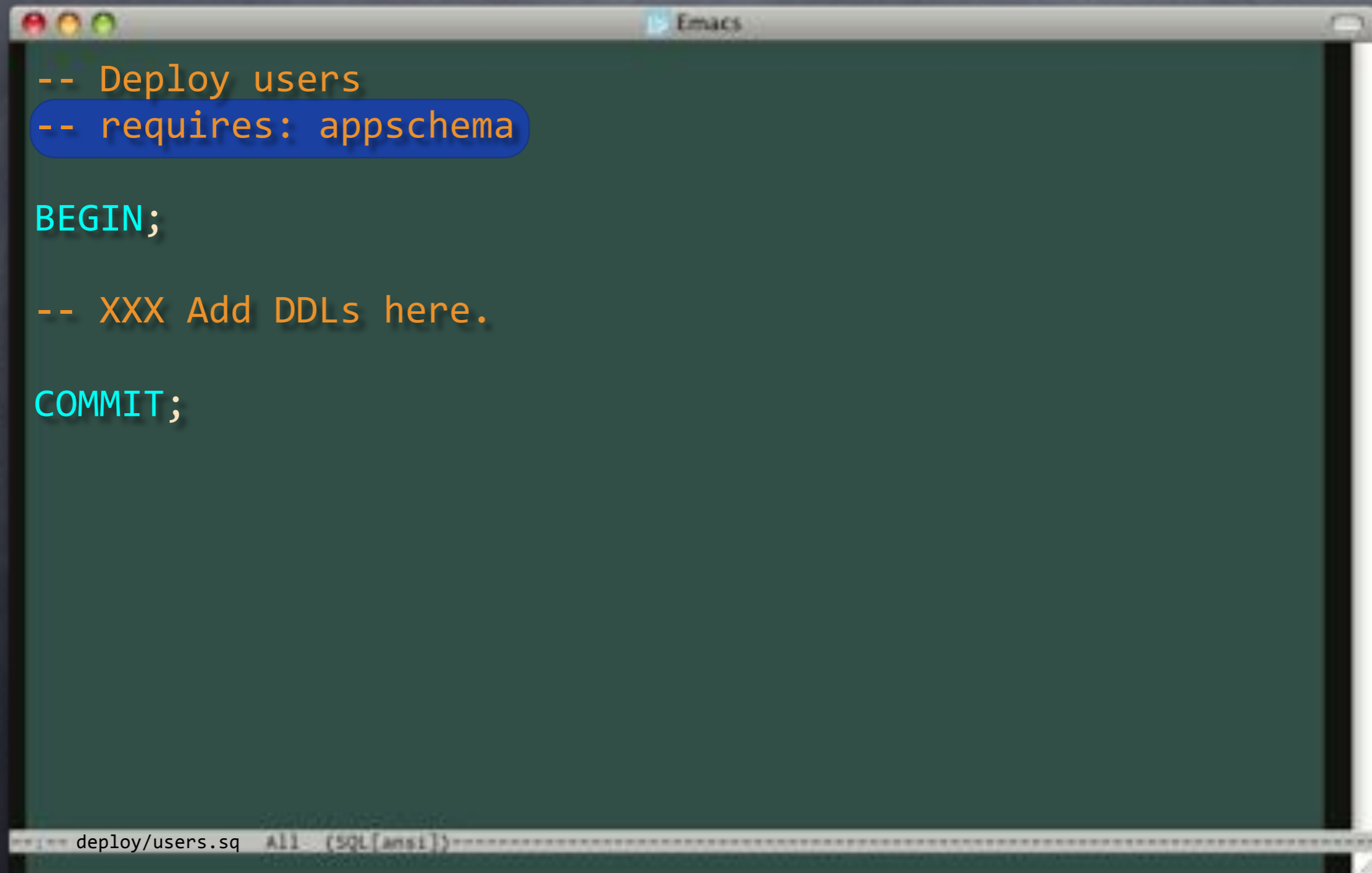
BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

deploy/users.sql All (SQL[ansi])

# deploy/users.sql



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

BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

The screenshot shows an Emacs editor window with a dark green background. The code is color-coded: comments are orange, SQL keywords are cyan, and the requirement 'requires: appschema' is highlighted in a blue rounded rectangle. The window title bar shows 'Emacs' and standard macOS window controls. The status bar at the bottom indicates the file path 'deploy/users.sql' and the mode '(SQL[ansi])'.



# deploy/users.sql

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

BEGIN;

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

COMMIT;
```

deploy/users.sql All (SQL[ansi])

# deploy/users.sql

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

BEGIN;

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

COMMIT;
```

Bare  
Minimum

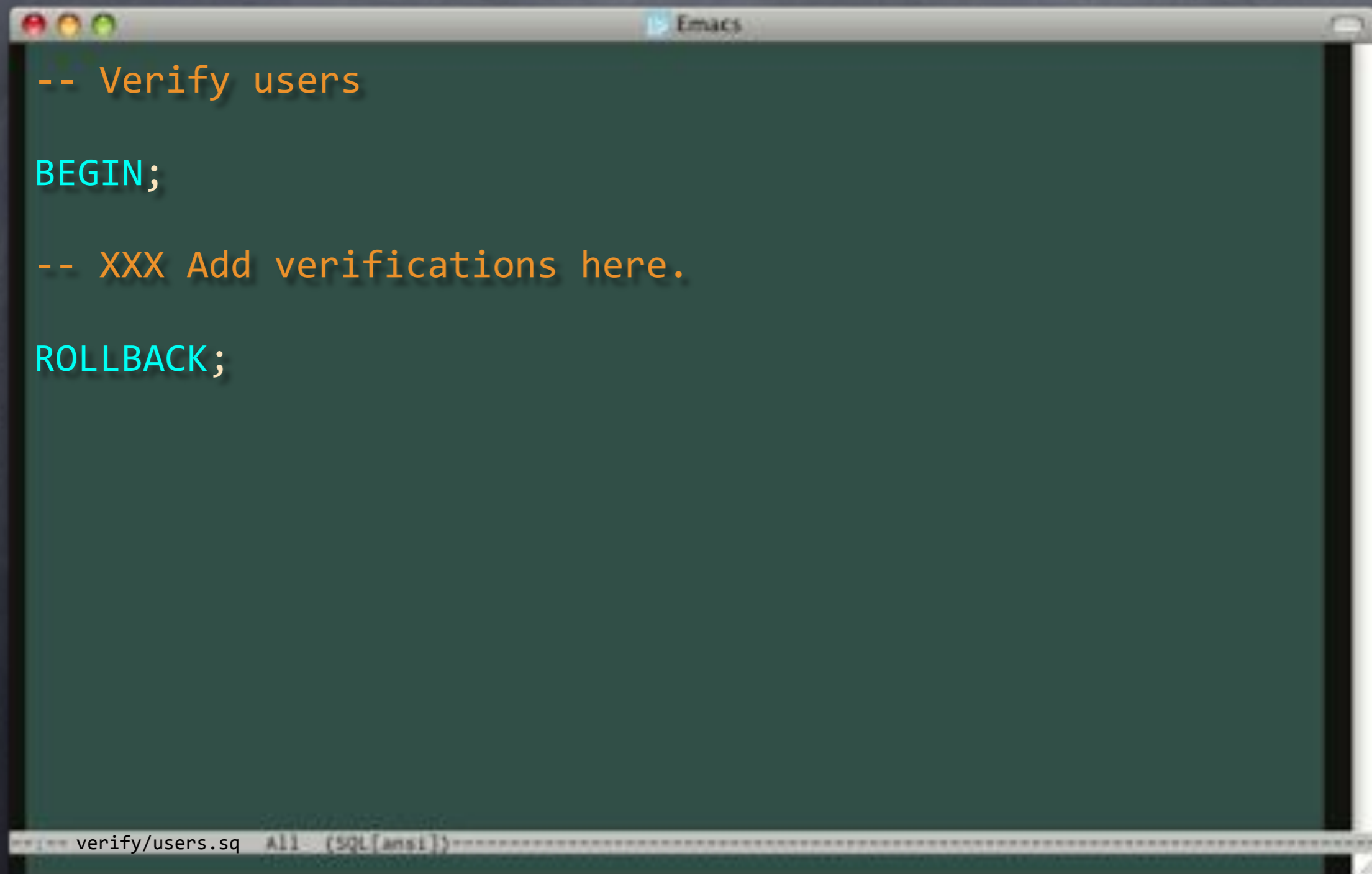
# Verily, Users

```
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
>
```

# Verify, 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



```
-- Verify users

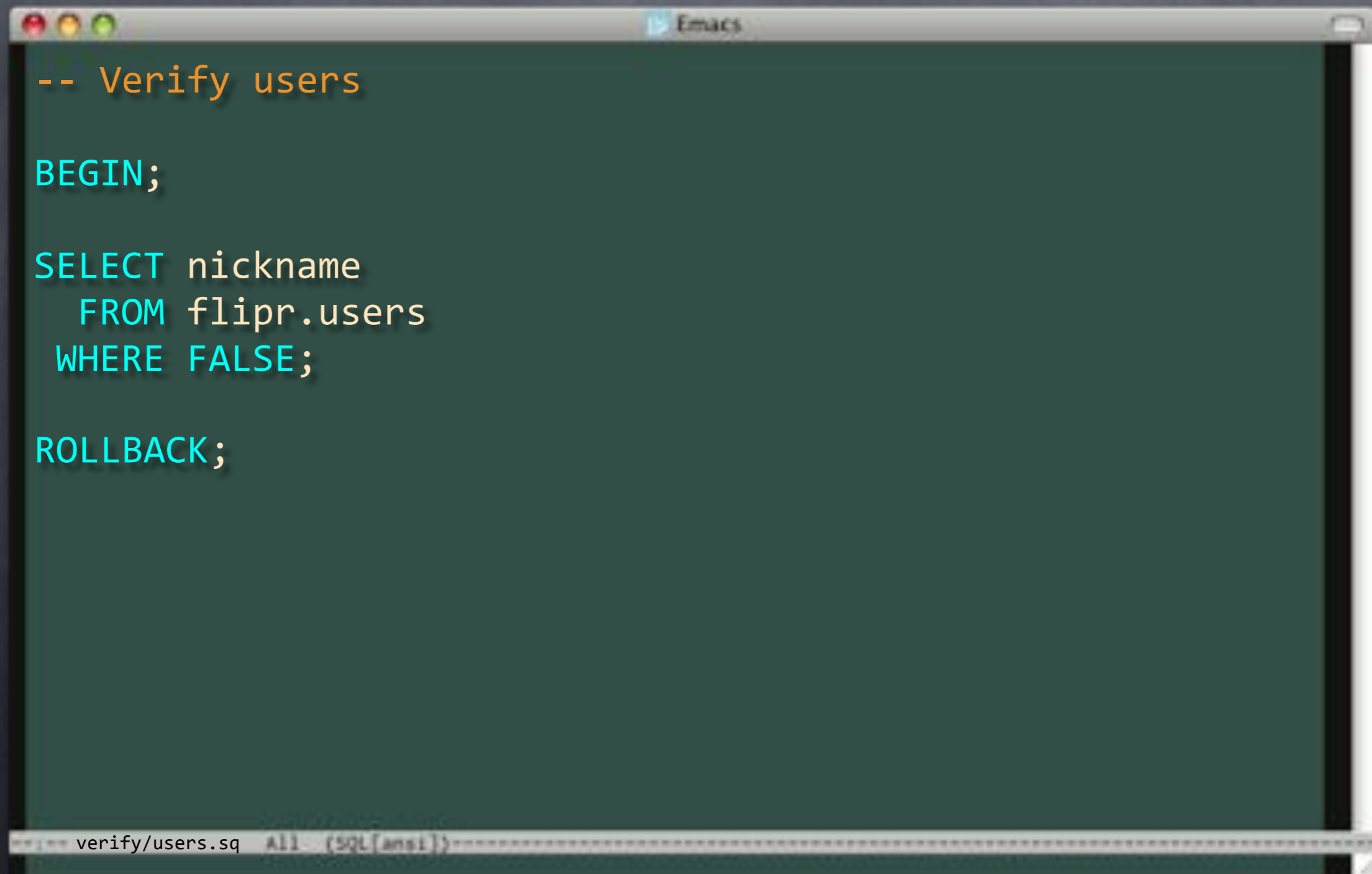
BEGIN;

-- XXX Add verifications here.

ROLLBACK;
```

The screenshot shows an Emacs editor window with a dark green background. The title bar at the top reads "Emacs". The code is displayed in a monospaced font with syntax highlighting: comments are in orange, and SQL keywords are in cyan. The status bar at the bottom of the window shows "verify/users.sq" and "All (SQL[ansi])".

# verify/users.sql

A screenshot of an Emacs editor window. The window title bar shows the Emacs logo and the text "Emacs". The editor content is a dark green background with SQL code in a light blue font. The code is: 

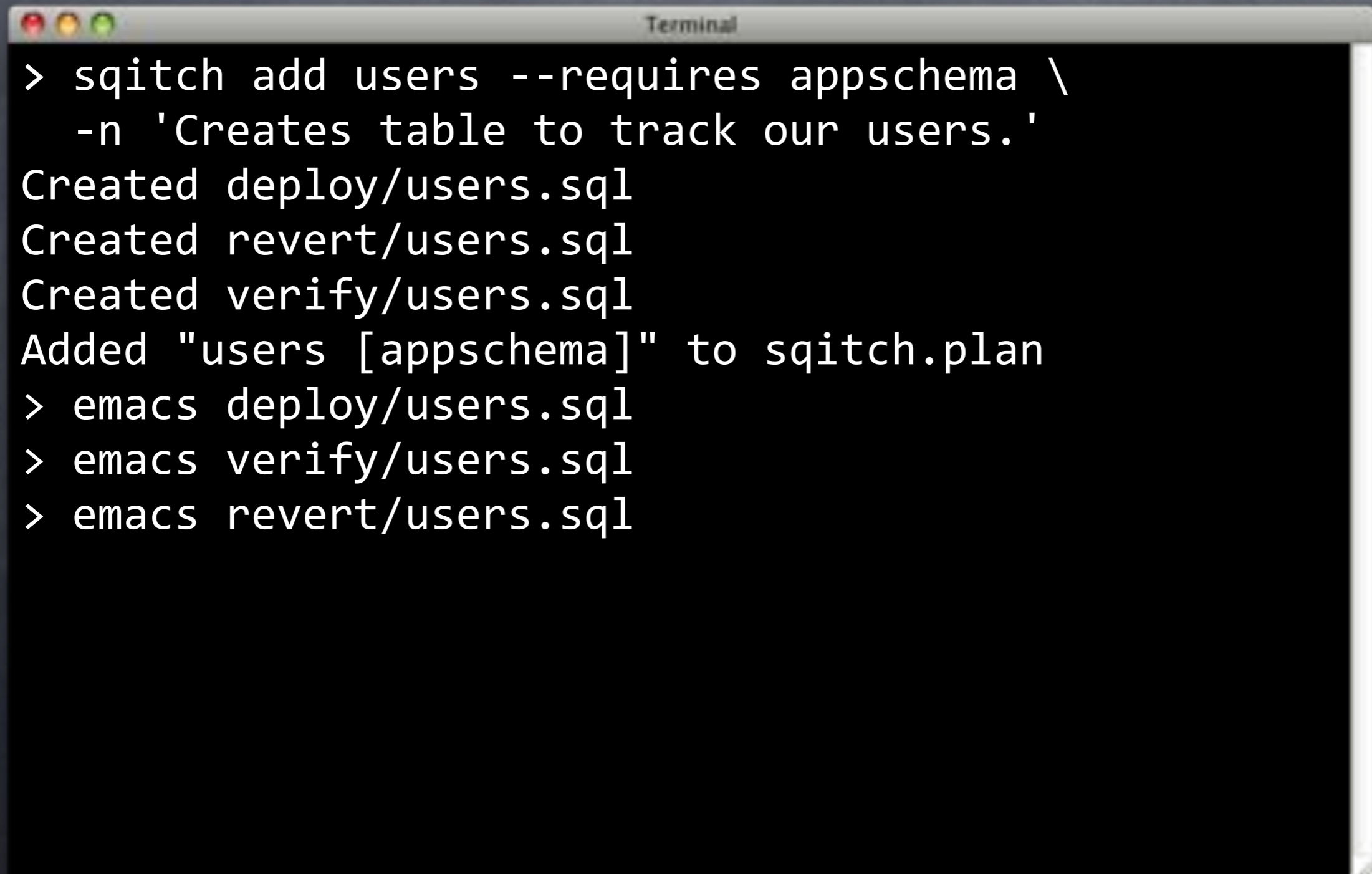
```
-- Verify users  
  
BEGIN;  
  
SELECT nickname  
  FROM flipr.users  
 WHERE FALSE;  
  
ROLLBACK;
```

 At the bottom of the window, the status bar shows "verify/users.sq" and "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
>
```

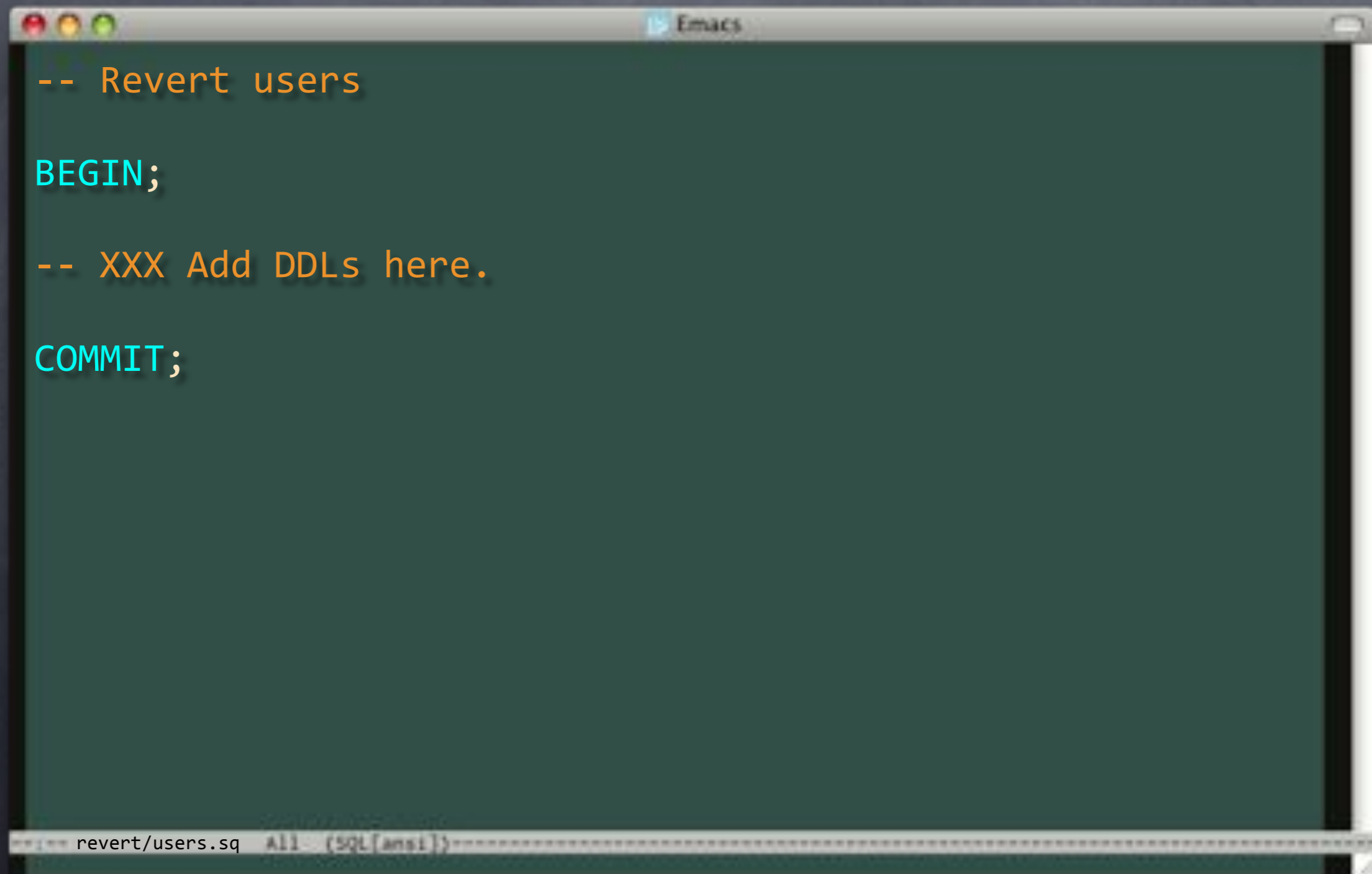
# Unusered



```
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



```
-- Revert users

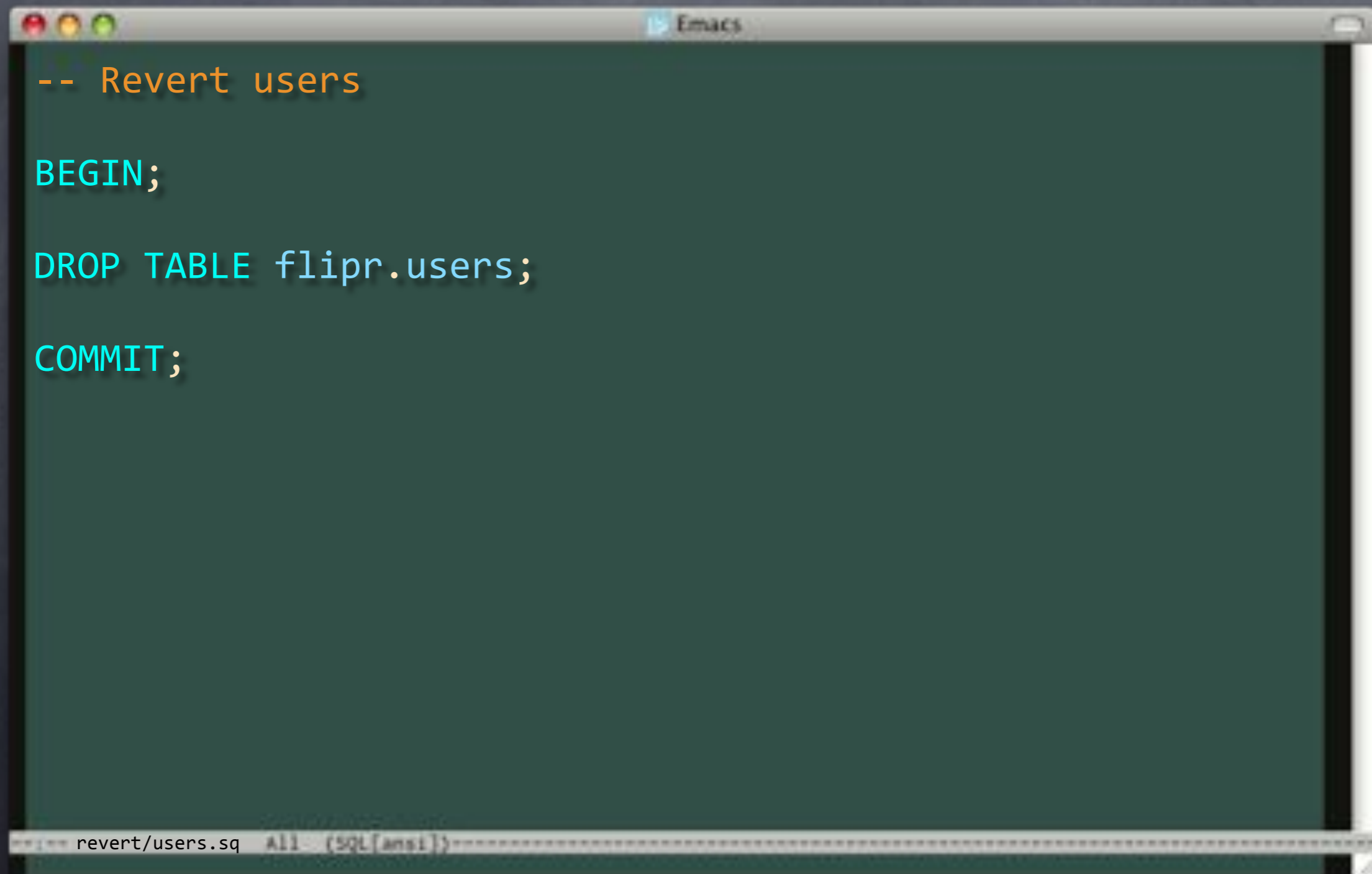
BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

The screenshot shows an Emacs editor window with a dark green background. The title bar at the top reads "Emacs". The code is displayed in a monospaced font with syntax highlighting: orange for comments, cyan for SQL keywords, and black for the rest of the code. The status bar at the bottom of the window shows "revert/users.sql" and "All (SQL[ansi])".

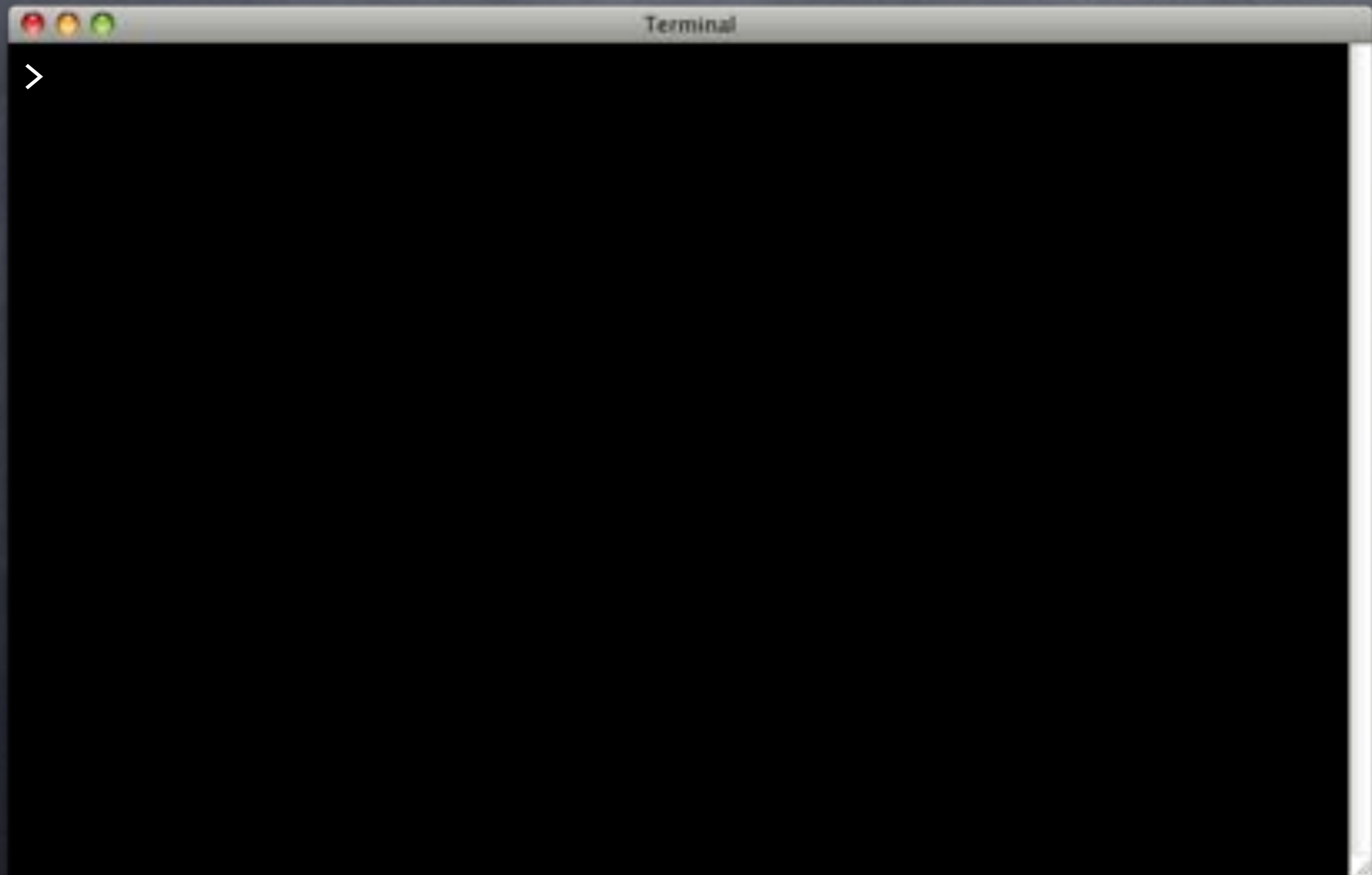
# revert/users.sql



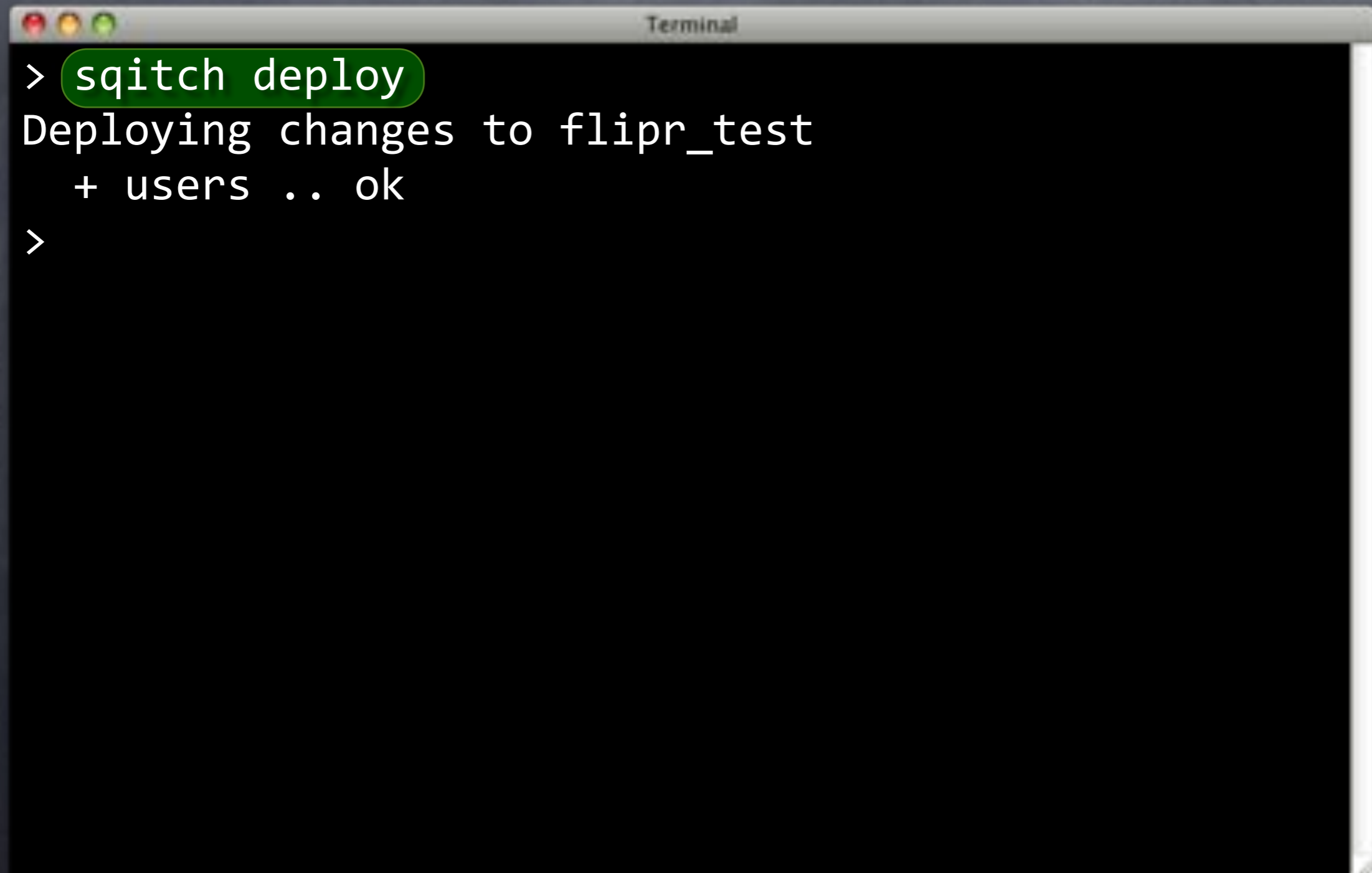
```
-- Revert users  
  
BEGIN;  
  
DROP TABLE flipr.users;  
  
COMMIT;
```

The screenshot shows an Emacs editor window with a dark green background. The title bar at the top reads "Emacs". The code is displayed in a monospaced font with syntax highlighting: the comment "-- Revert users" is in orange, and the SQL keywords "BEGIN;", "DROP TABLE flipr.users;", and "COMMIT;" are in cyan. The status bar at the bottom of the window shows "revert/users.sql" and "All (SQL[ansi])".

# Make Users

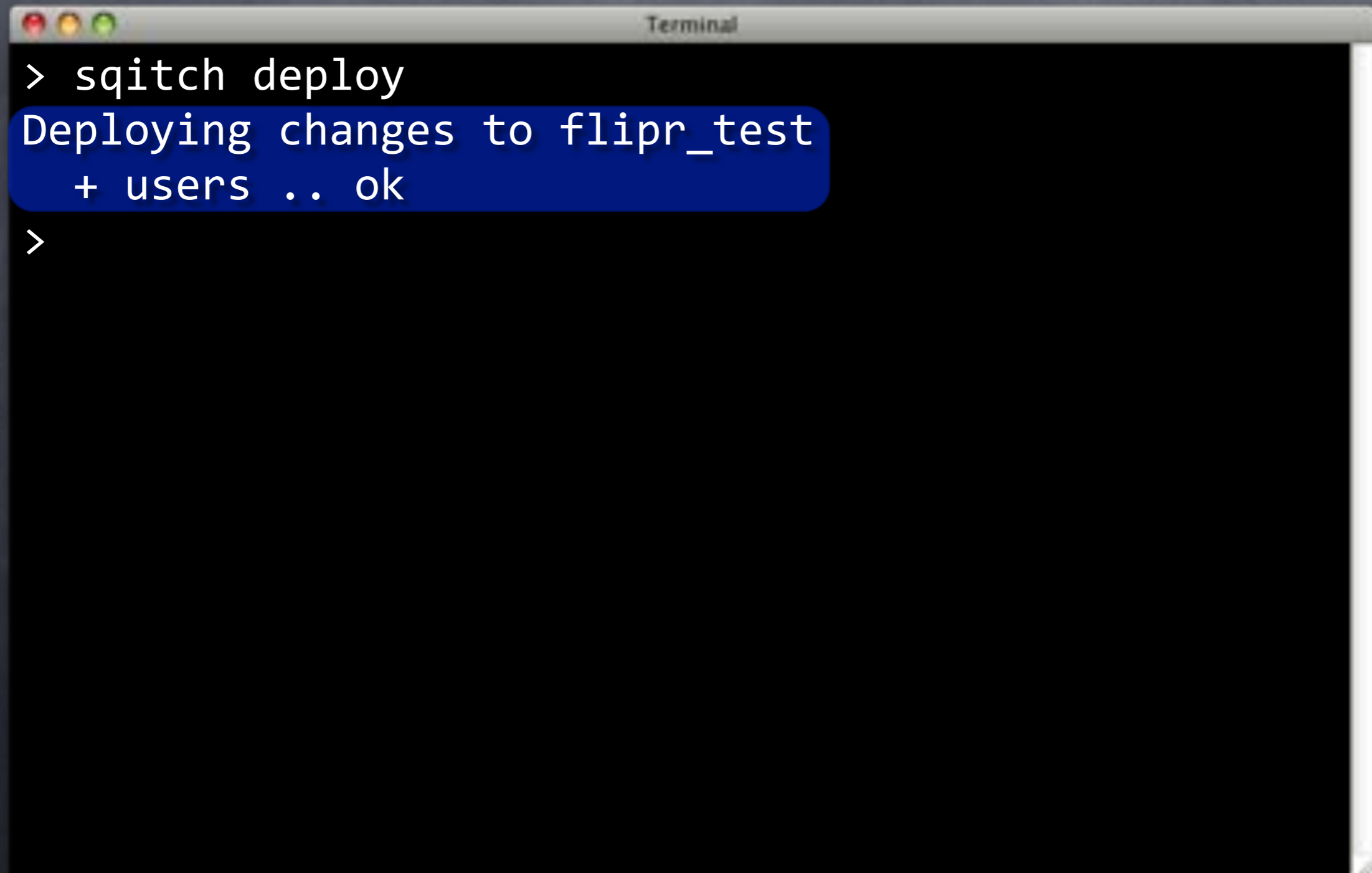


# Make Users



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

# Make Users

A terminal window titled "Terminal" with a dark background and light text. The window shows a command prompt with the following text:

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

The text "Deploying changes to flipr\_test + users .. ok" is highlighted with a blue background.

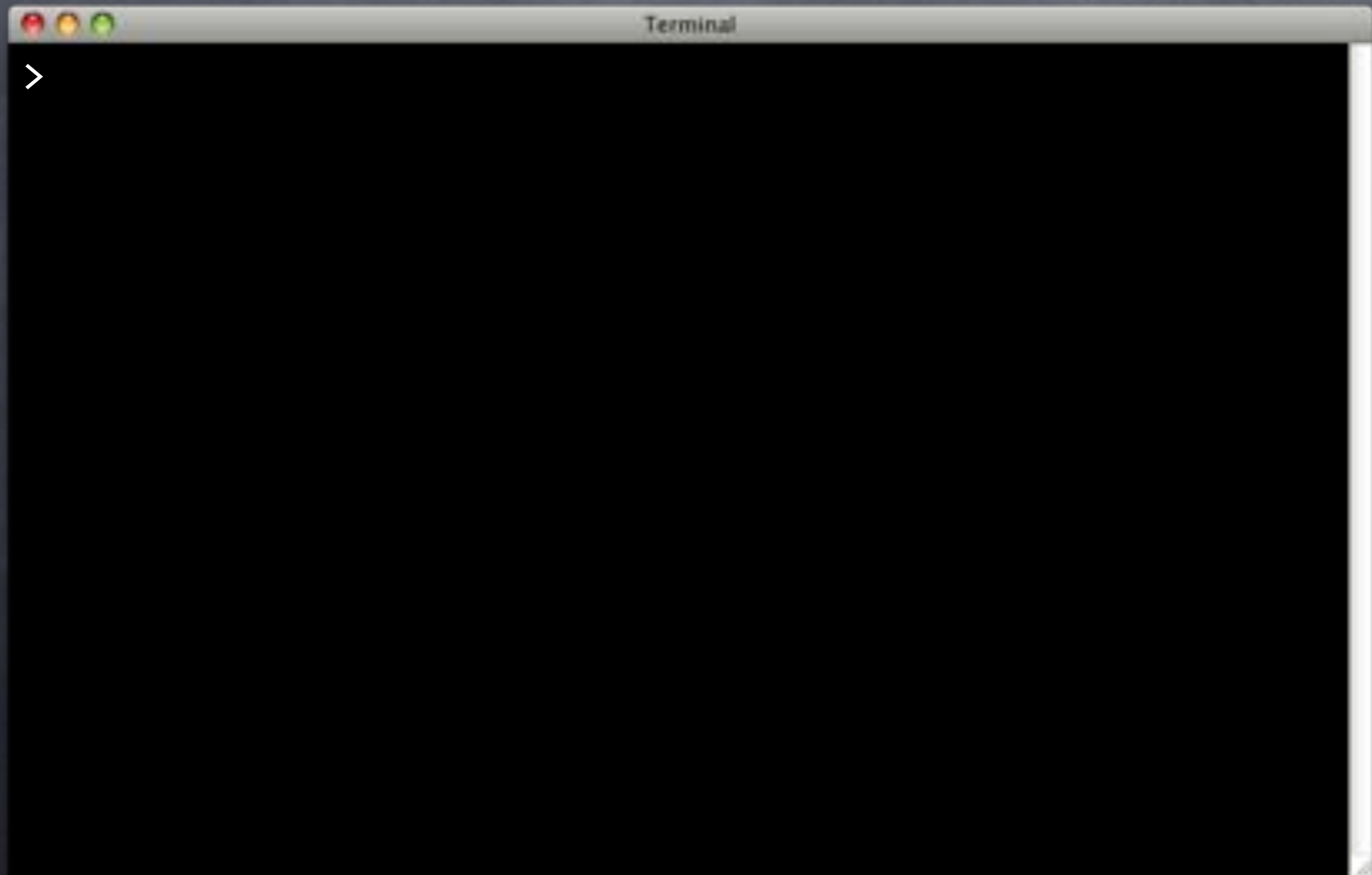
# Make Users

```
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

```
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;
```

# Columnist

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

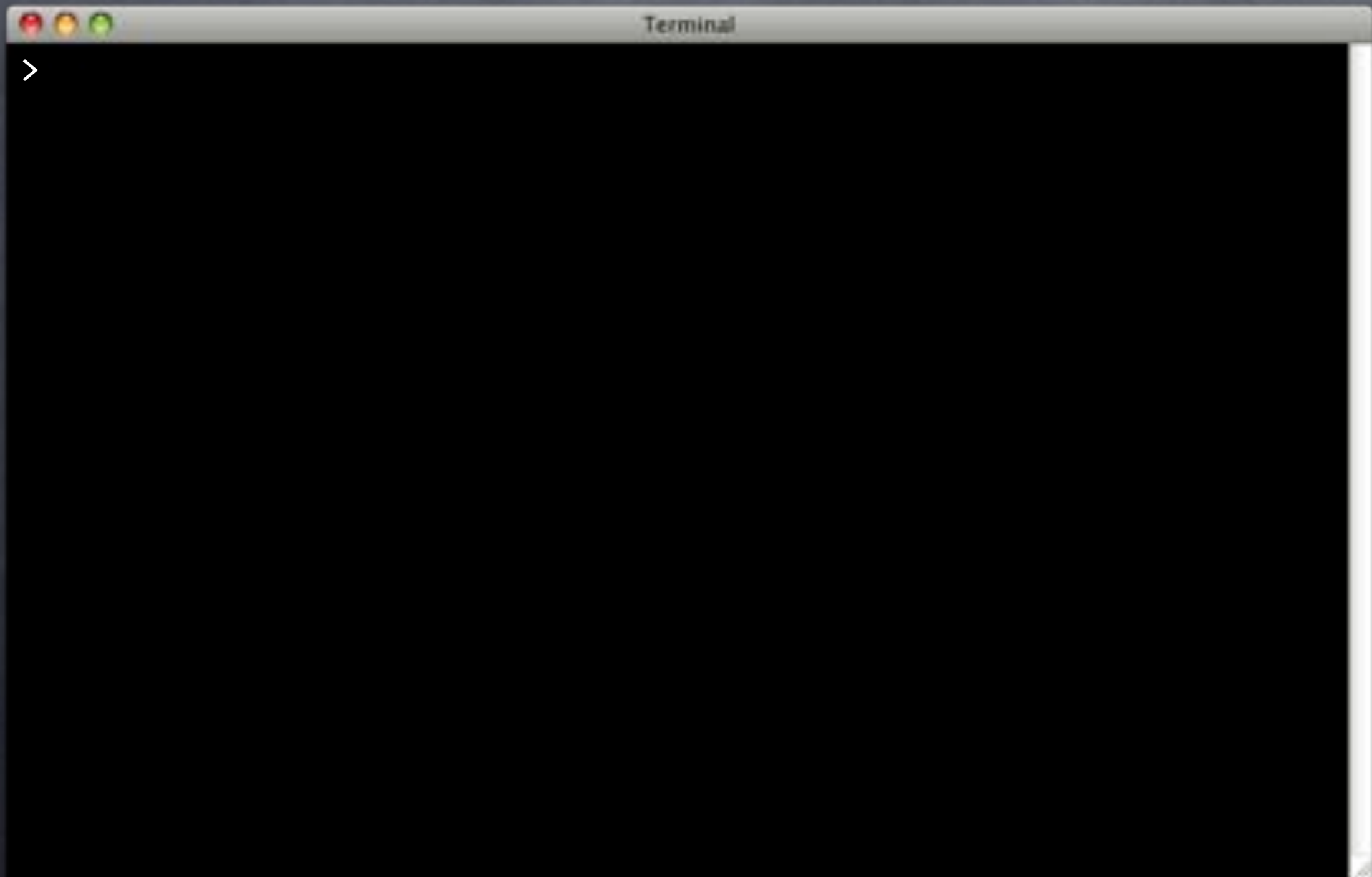
BEGIN;
SELECT no_plan();
SET search_path TO flipr,public;

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
```



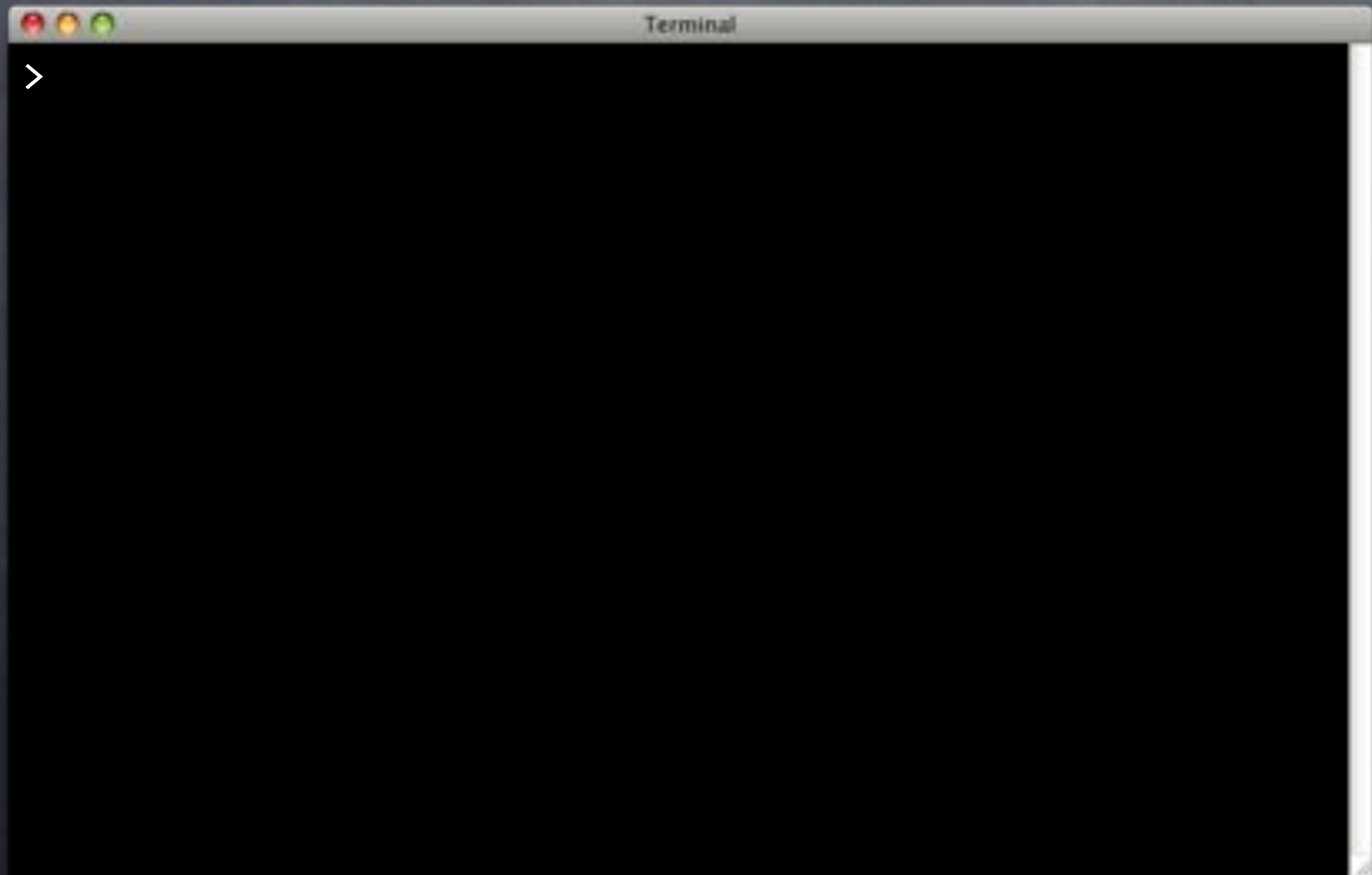
# Dead Again

```
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 2 out of 4 tests.
Failed 2/4 subtests

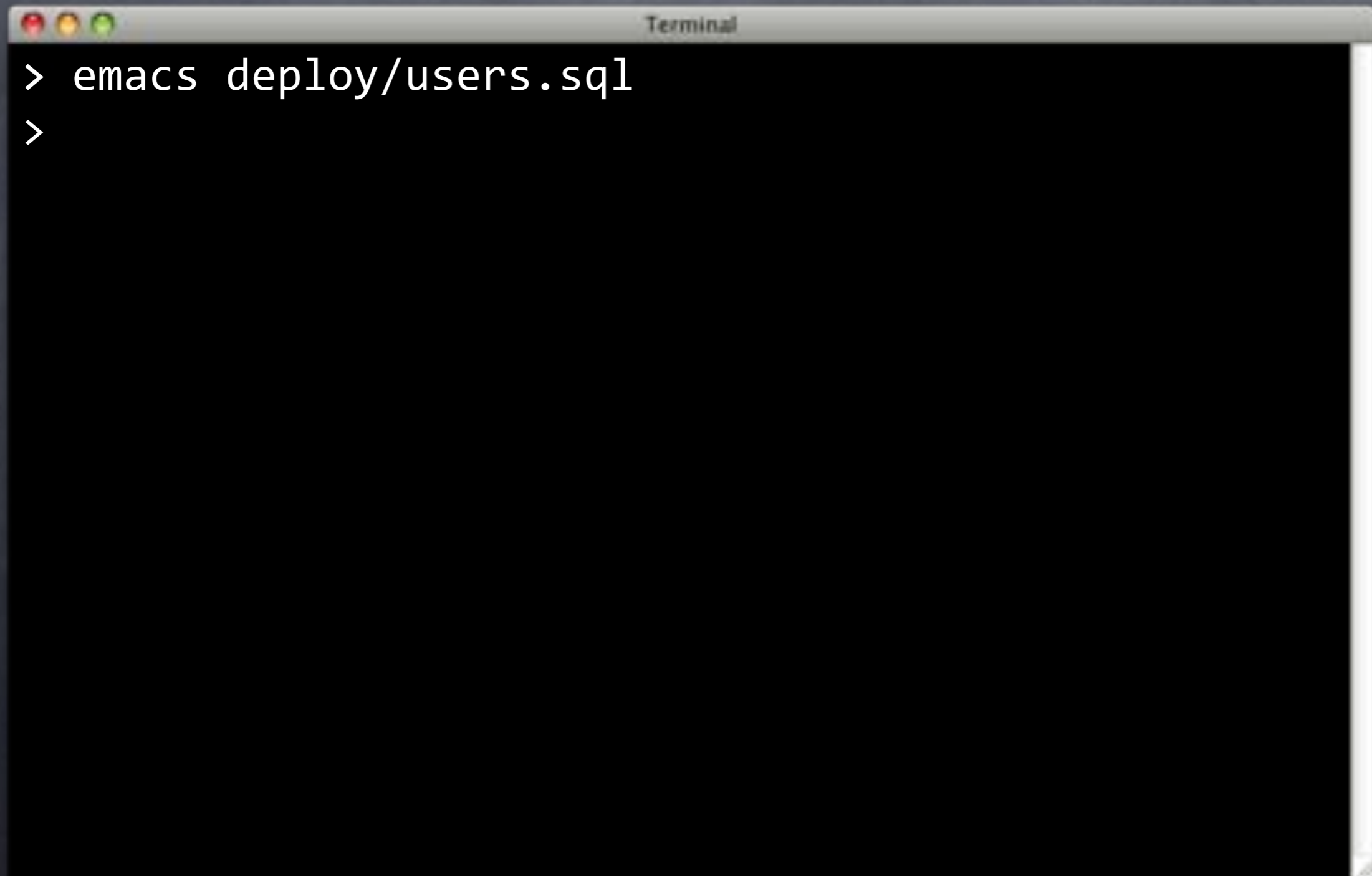
Test Summary Report
-----
test/users.sql (Wstat: 0 tests: 4 failed: 2)
Failed tests: 3-4
Files=1, Tests=4, 0 wallclock secs
Result: FAIL
```

Guess we should add them.

# MOAR Deploy



# MOAR Deploy

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows a prompt character followed by the command "emacs deploy/users.sql" and a second prompt character on the next line.

```
> emacs deploy/users.sql  
>
```

# deploy/users.sql

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

BEGIN;

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

COMMIT;
```

deploy/users.sql All (SQL[ansi])

# deploy/users.sql

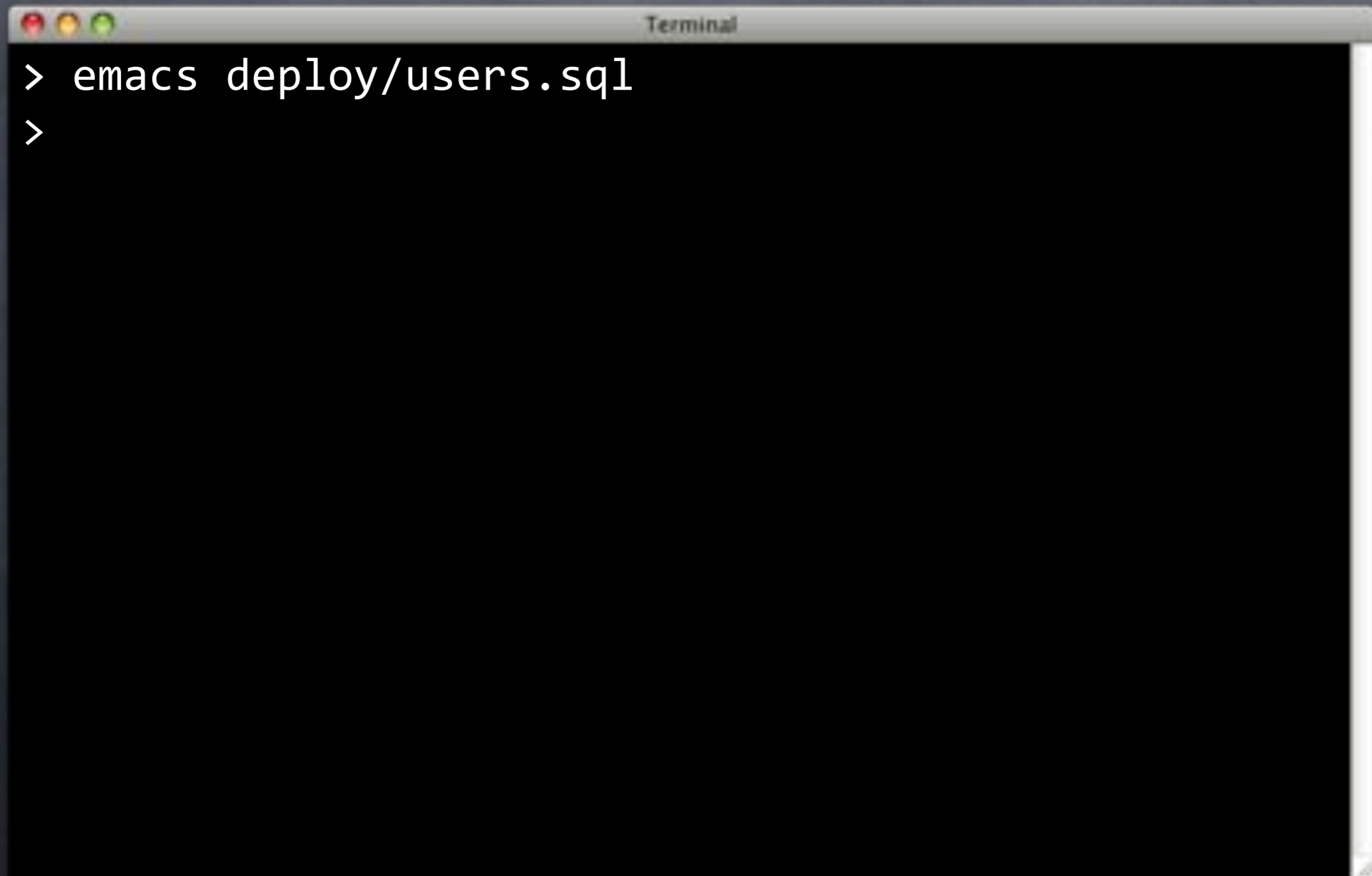
```
-- Deploy users
-- requires: appschema

BEGIN;

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

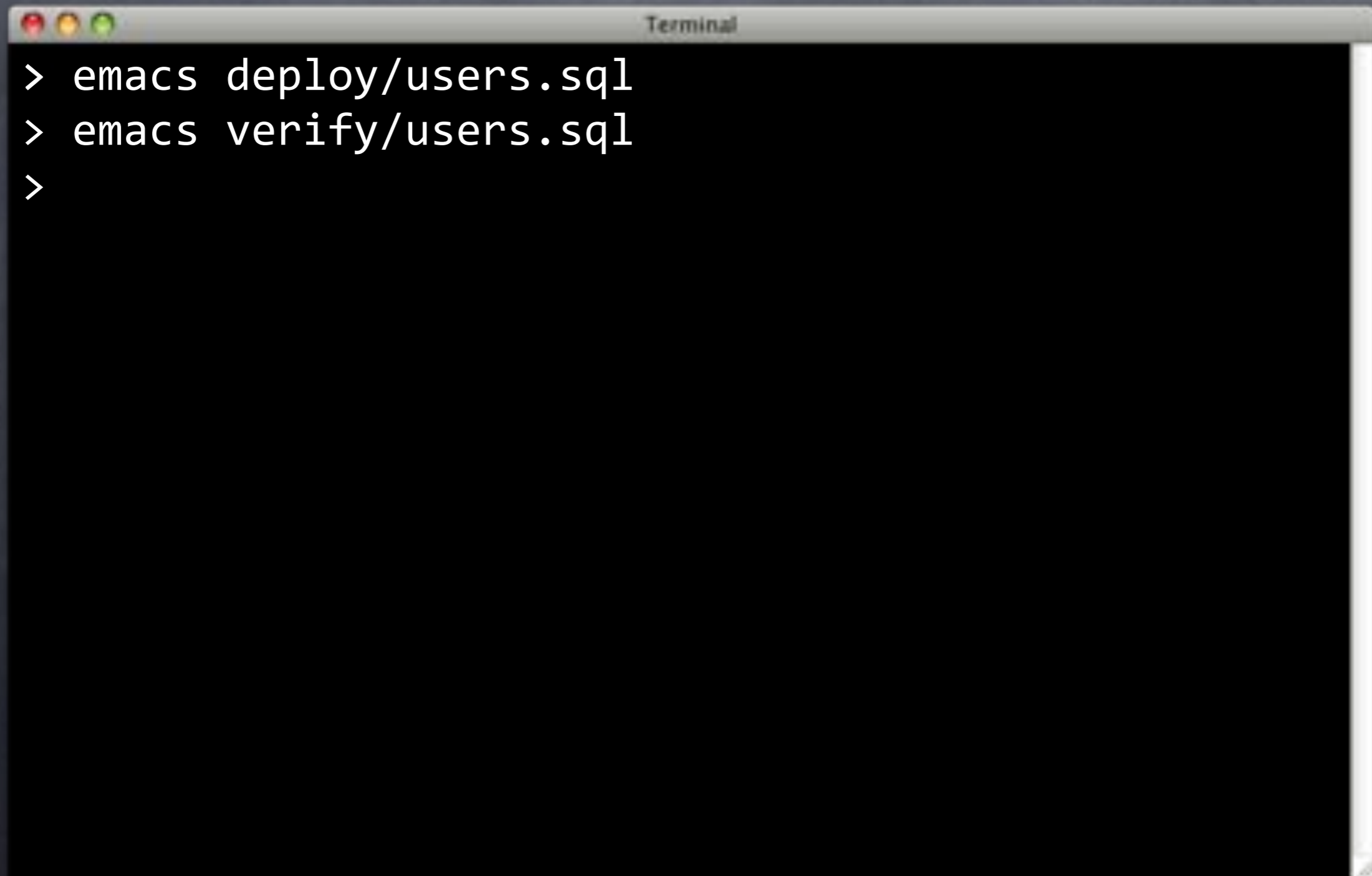
COMMIT;
```

# Update Verify

A terminal window titled "Terminal" with a standard macOS window header (red, yellow, green buttons). The terminal content shows a command prompt followed by the command to open a file in emacs.

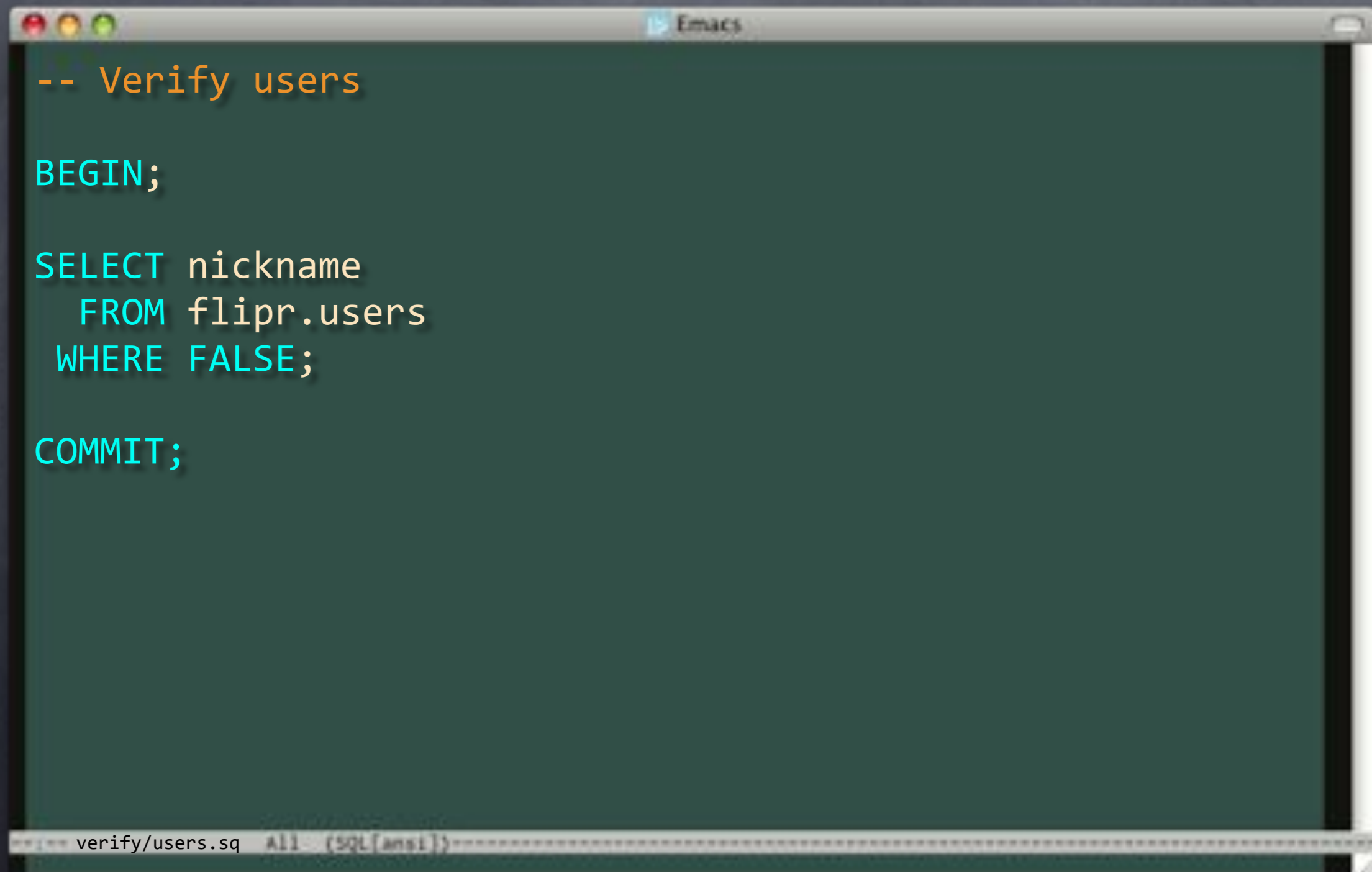
```
> emacs deploy/users.sql  
>
```

# Update Verify

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows three lines of text, each starting with a greater-than sign (>).

```
> emacs deploy/users.sql  
> emacs verify/users.sql  
>
```

# verify/users.sql

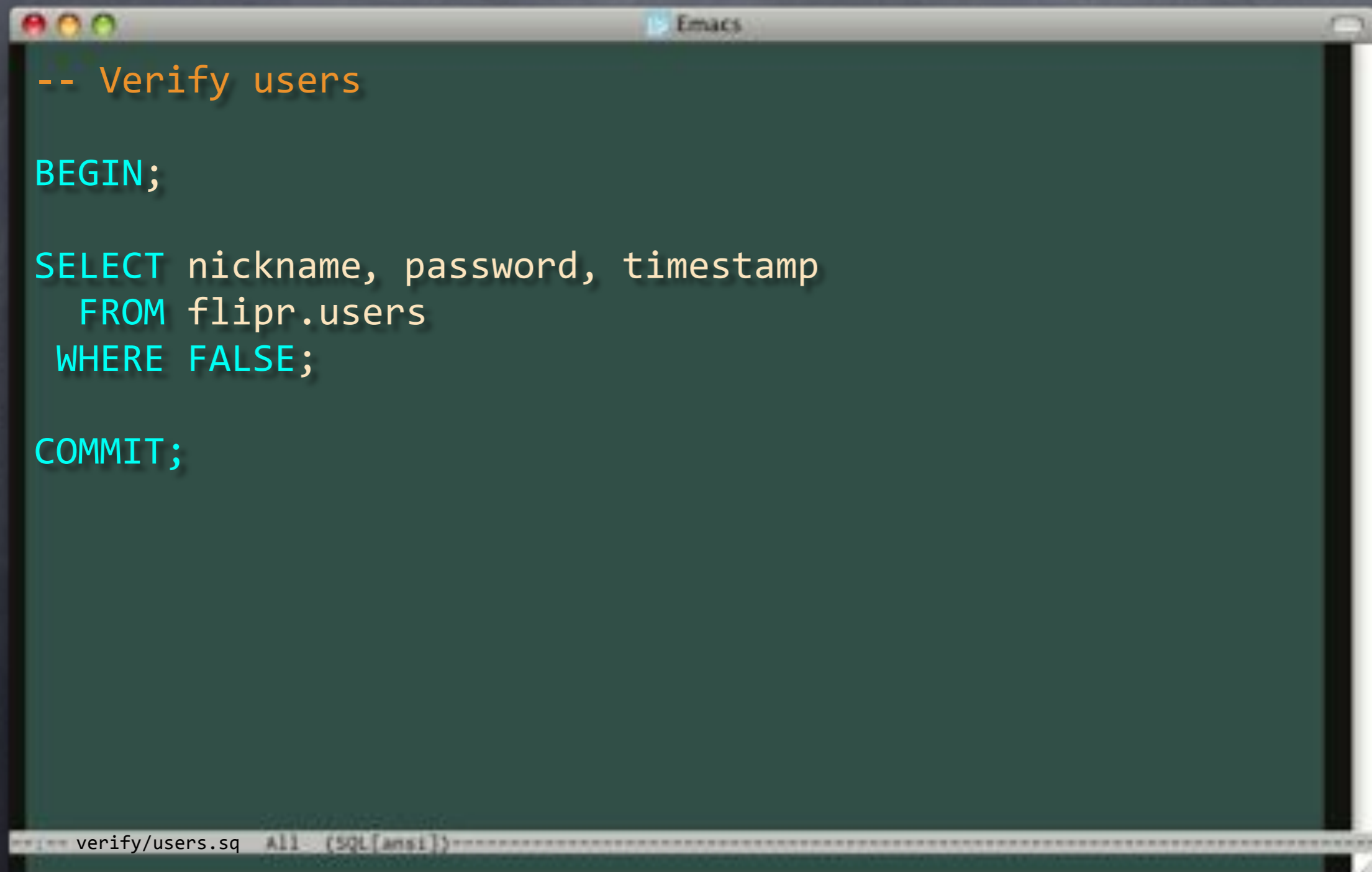
A screenshot of an Emacs editor window. The window title bar shows "Emacs" and standard macOS window controls (red, yellow, green buttons). The main content area is a dark green background with SQL code in a light blue/cyan font. The code is: 

```
-- Verify users  
  
BEGIN;  
  
SELECT nickname  
  FROM flipr.users  
 WHERE FALSE;  
  
COMMIT;
```

 At the bottom of the window, a status bar shows "verify/users.sq" and "All (SQL[ansi])".



# verify/users.sql

A screenshot of an Emacs editor window. The window title bar shows the Emacs logo and the text "Emacs". The editor content is a SQL script with the following lines: "-- Verify users", "BEGIN;", "SELECT nickname, password, timestamp", "FROM flipr.users", "WHERE FALSE;", "COMMIT;". The status bar at the bottom of the window shows "verify/users.sq", "All", and "(SQL[ansi])".

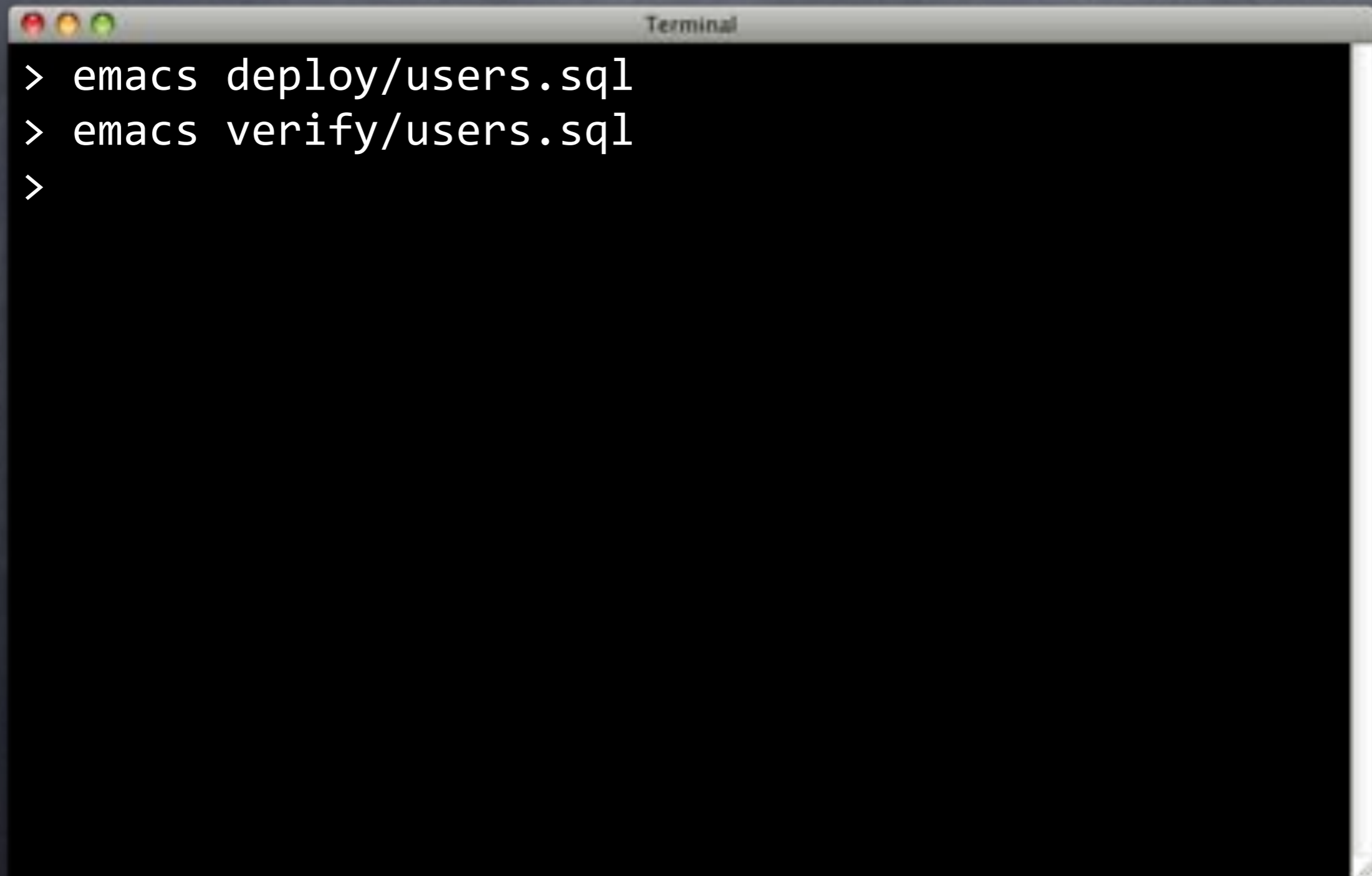
```
-- Verify users

BEGIN;

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

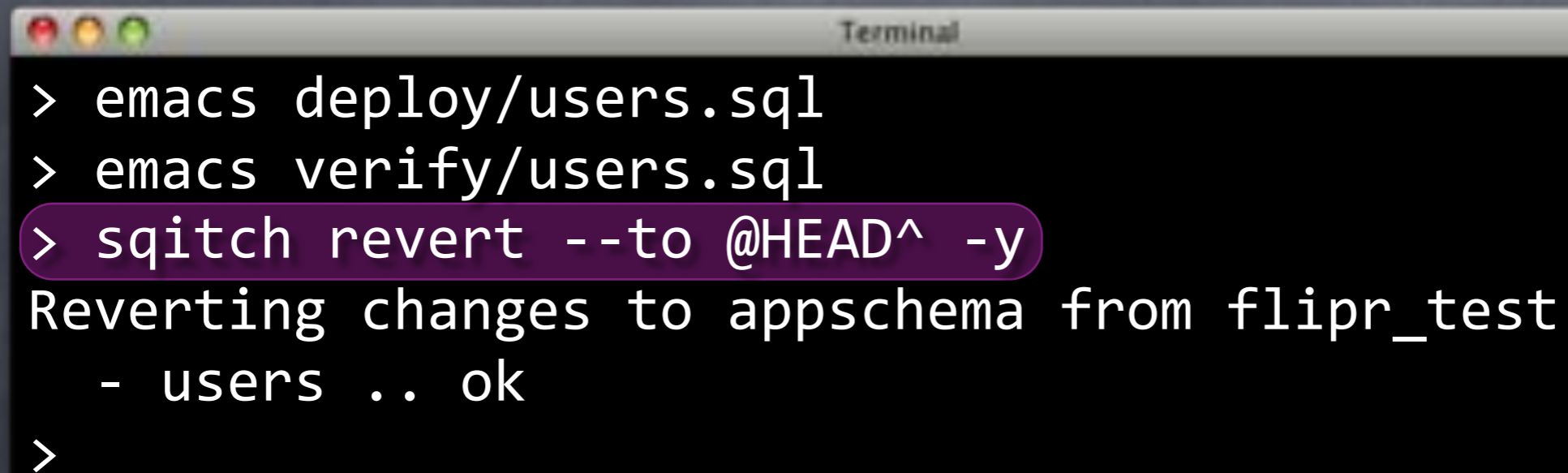
COMMIT;
```

# Revert Overhead



```
Terminal  
> emacs deploy/users.sql  
> emacs verify/users.sql  
>
```

# 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
>
```

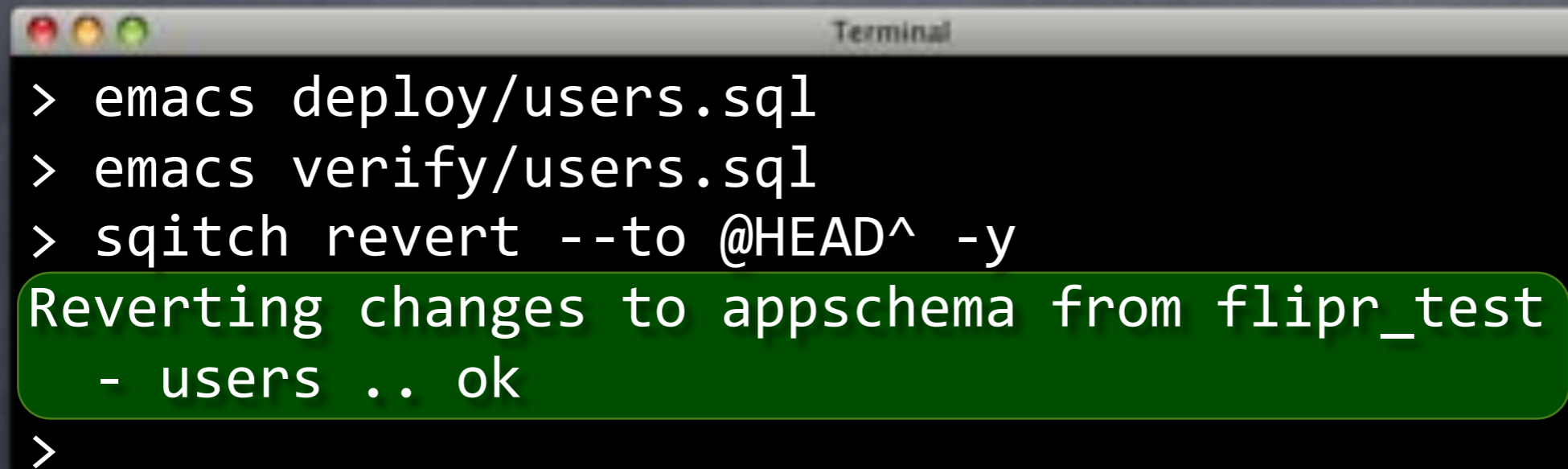
# Revert Overhead

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows a sequence of commands and their output. The command `sqitch revert --to @HEAD^ -y` is highlighted with a purple background. A blue speech bubble points to this command with the text "Yes, really." The output shows the process of reverting changes to the 'users' table from the 'flipr\_test' branch to the '@HEAD^' commit, resulting in 'users .. ok'.

```
> 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



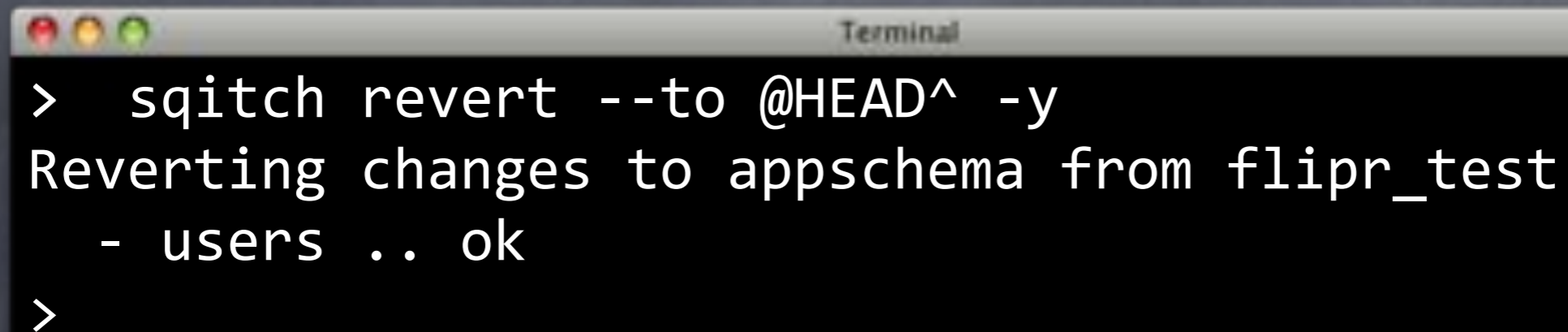
# Specifying Changes

- `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



```
Terminal
> sqitch revert --to @HEAD^ -y
Reverting changes to appschema from flipr_test
- users .. ok
>
```

# Whither Users

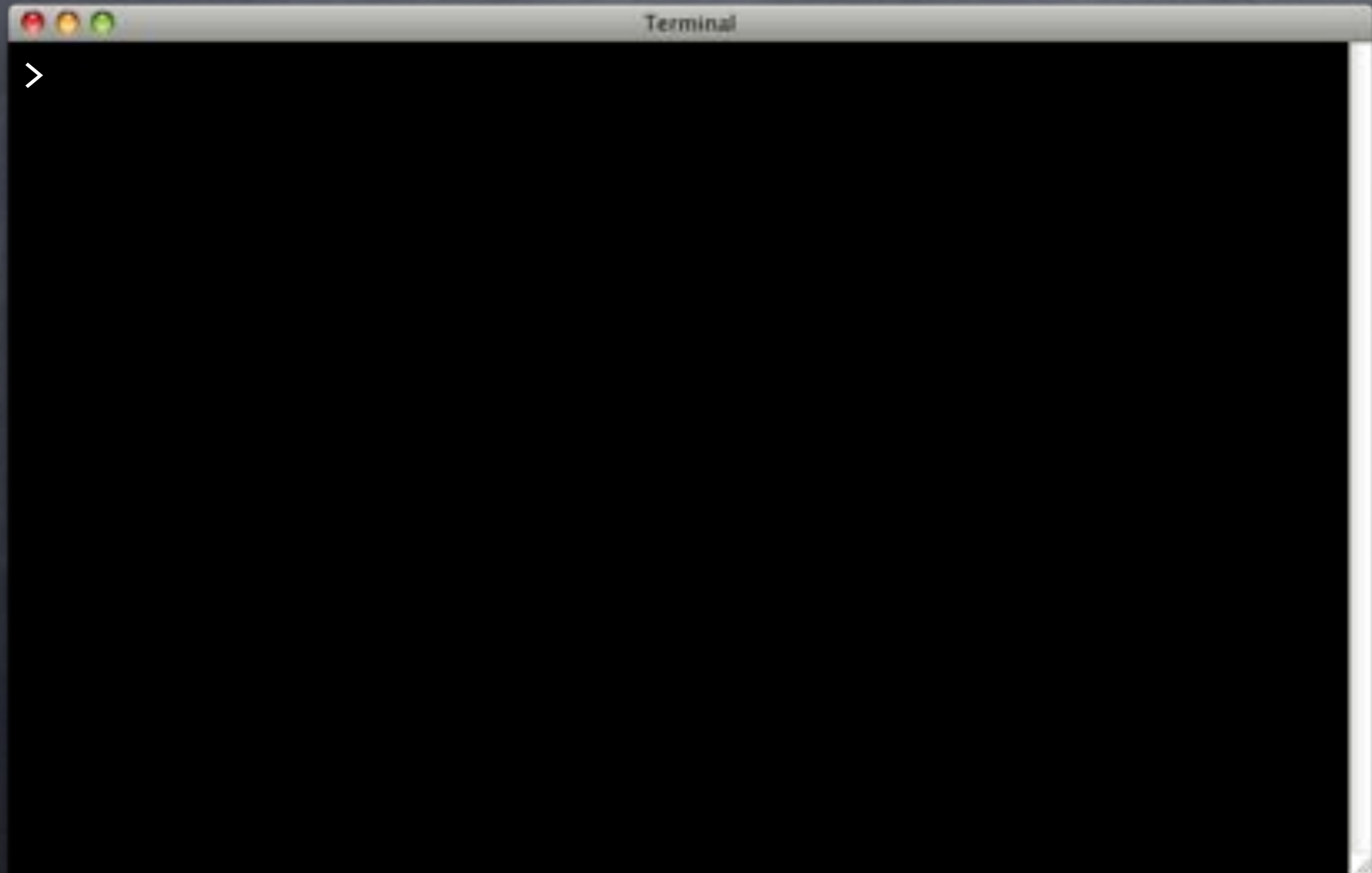
```
Terminal
> sqitch revert --to @HEAD^ -y
Reverting changes to appschema from flipr_test
- users .. ok
> psql -d flipr_test -c '\d flipr.users'
Did not find any relation named "flipr.users".
>
```

# Whither Users

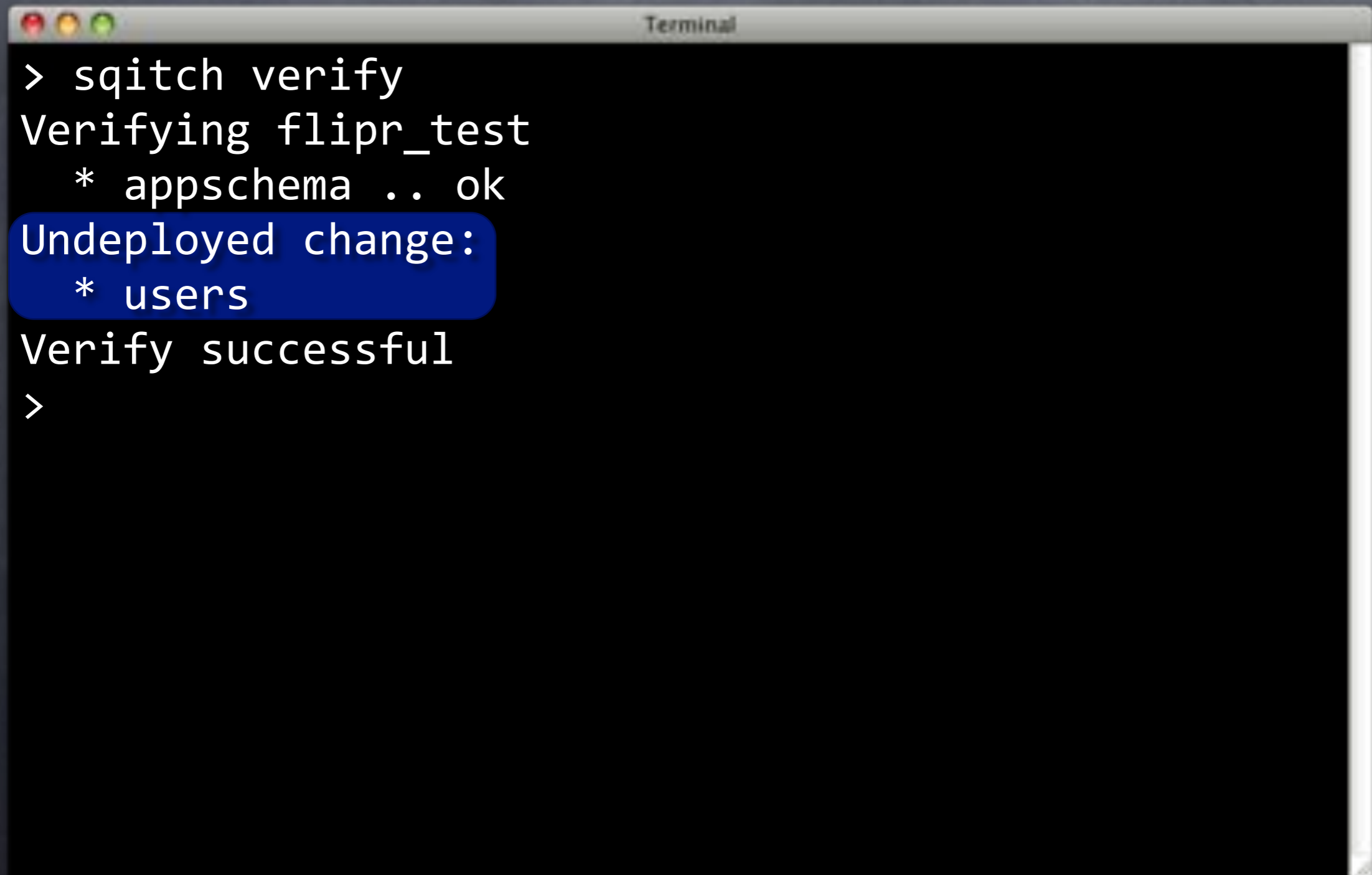
```
Terminal
> sqitch revert --to @HEAD^ -y
Reverting changes to appschema from flipr_test
- users .. ok
> psql -d flipr_test -c '\d flipr.users'
Did not find any relation named "flipr.users".
> sqitch status
# On database flipr_test
# Project:   flipr
# Change:    748346dfe73cf2af32a8b7088fd75ad8d7aecda3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

```
Undeployed change:
* users
```

# Whither Users

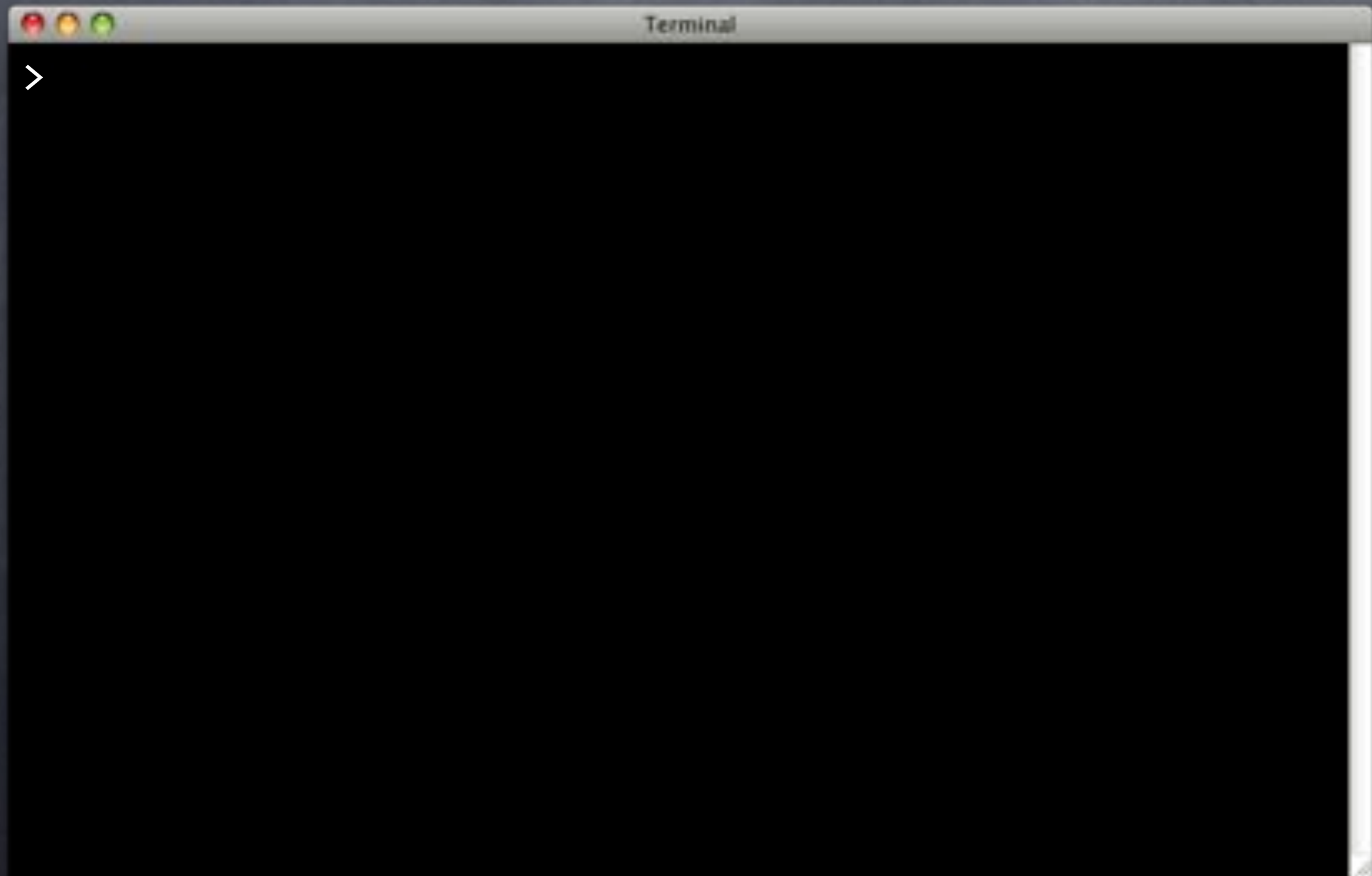


# Whither Users

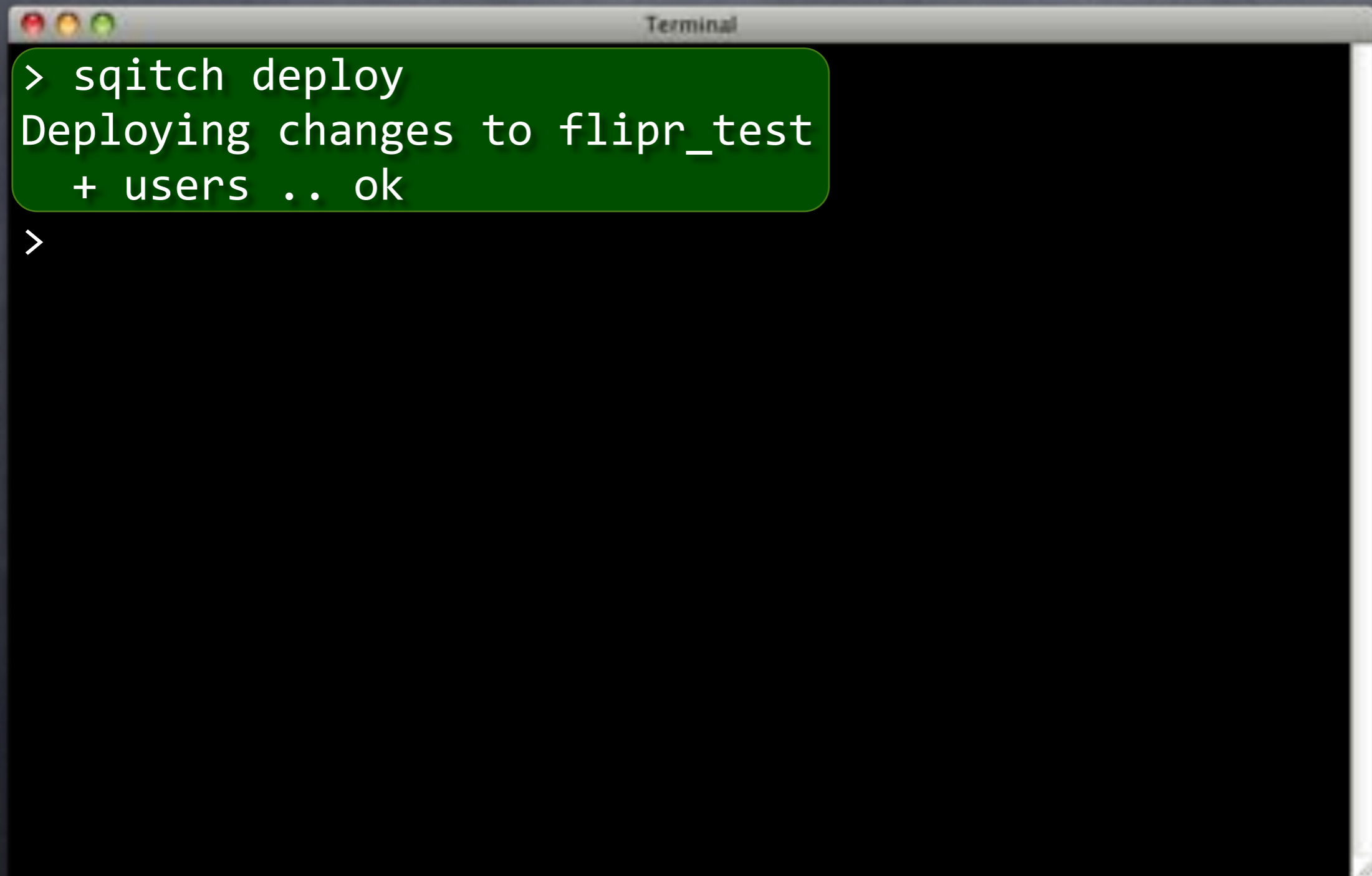
A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal displays the output of the command 'sqitch verify'. The output shows 'Verifying flipr\_test' followed by '\* appschema .. ok'. A blue highlight box surrounds the text 'Undeployed change: \* users'. The terminal then shows 'Verify successful' and a prompt '>'.

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

# Back At It

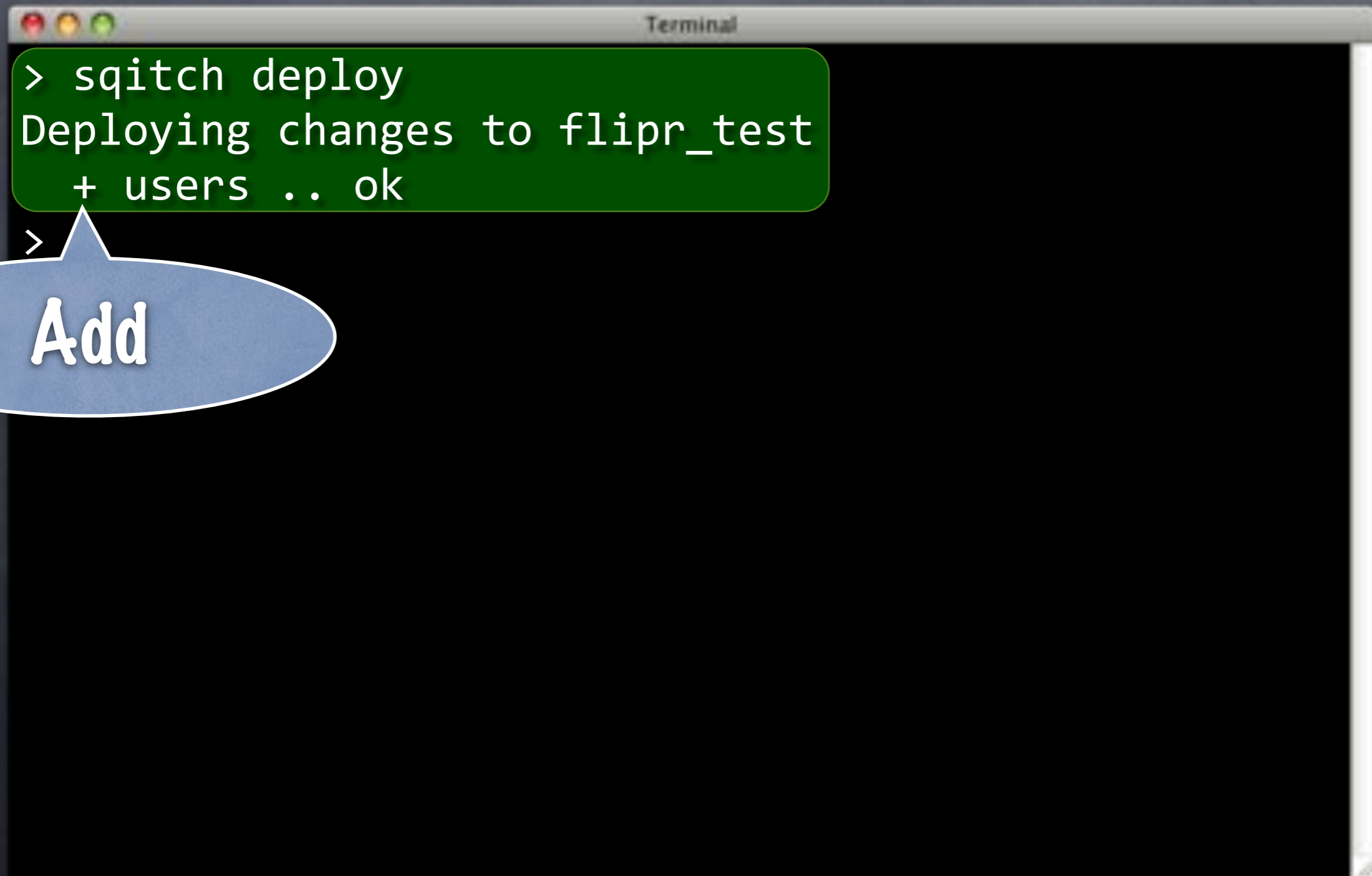


# Back At It

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content is highlighted in a green rounded rectangle. The text inside the terminal shows a successful deployment command and its output.

```
> 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
```

```
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;
```

```
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;
```

```
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;
```

```
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;
```

```
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' );

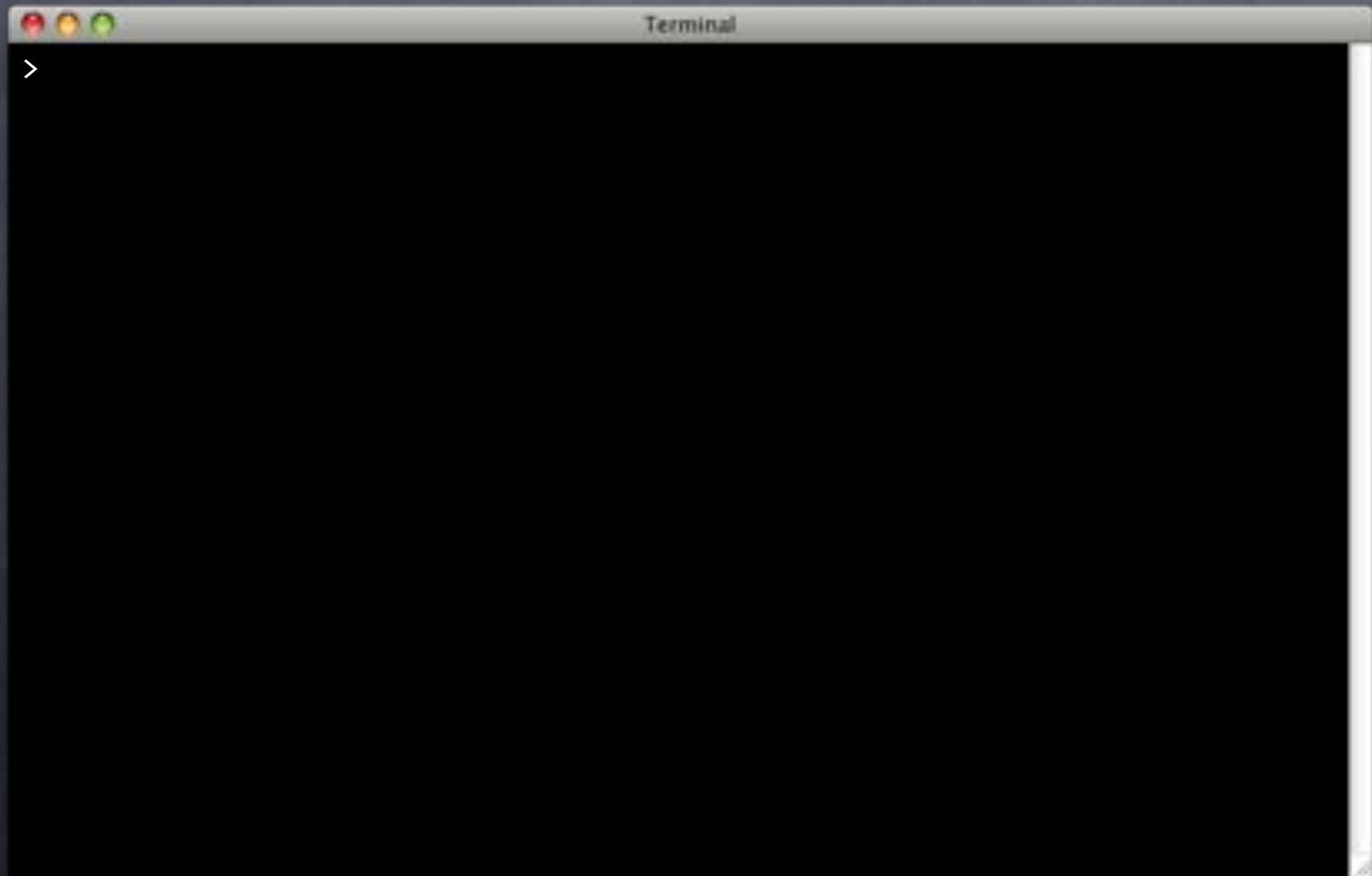
SELECT has_column( 'users', 'password' );
SELECT col_type_is( 'users', 'password', 'text' );
SELECT col_not_null( 'users', 'password' );
SELECT col_hasnt_default( 'users', 'password' );

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;
```



# Columnny



# Columnny

```
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
```

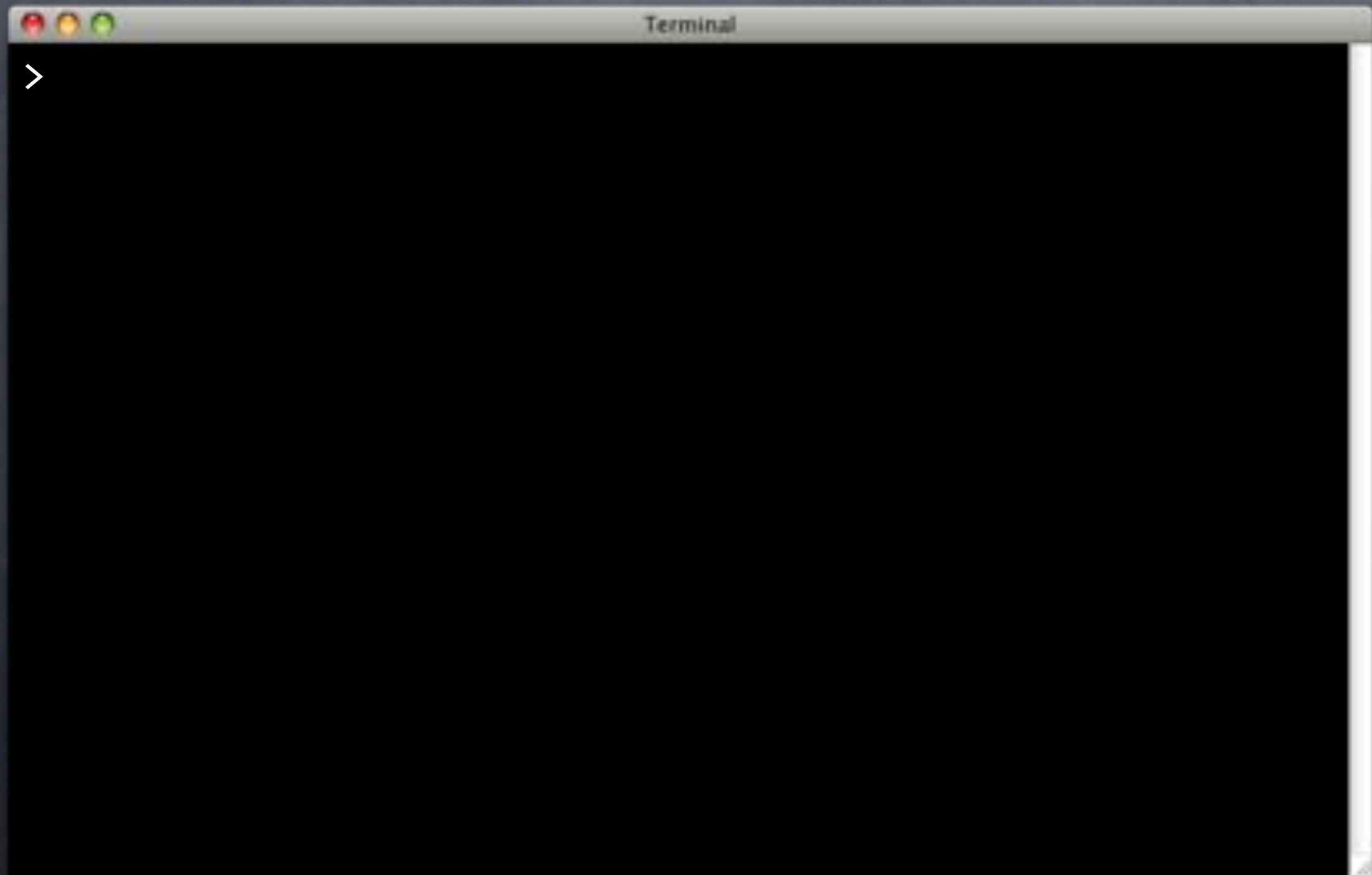
# Columnny

```
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

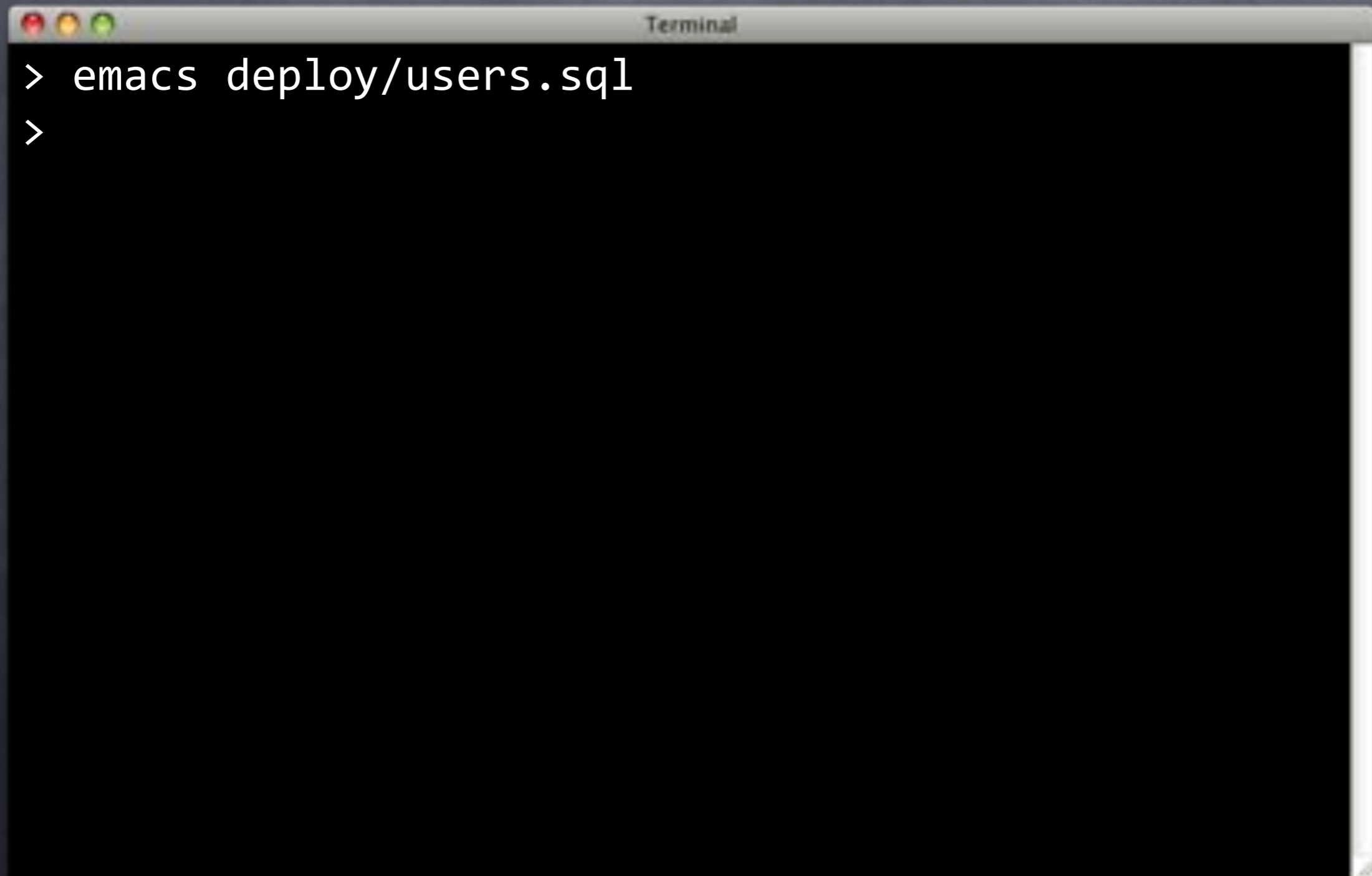
Test Summary Report
-----
test/users.sql (Wstat: 0 Tests: 15 Failed: 6)
  Failed tests:  2, 6, 9, 13-15
Files=1, Tests=15,  0 wallclock secs
Result: FAIL
```

Let's make  
it so.

# User Typography



# User Typography

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows a prompt character ">" followed by the command "emacs deploy/users.sql" on the first line, and another prompt character ">" on the second line.

```
> emacs deploy/users.sql  
>
```

# deploy/users.sql

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

BEGIN;

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

COMMIT;
```

# deploy/users.sql

```
-- 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.sql

```
-- 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.sql

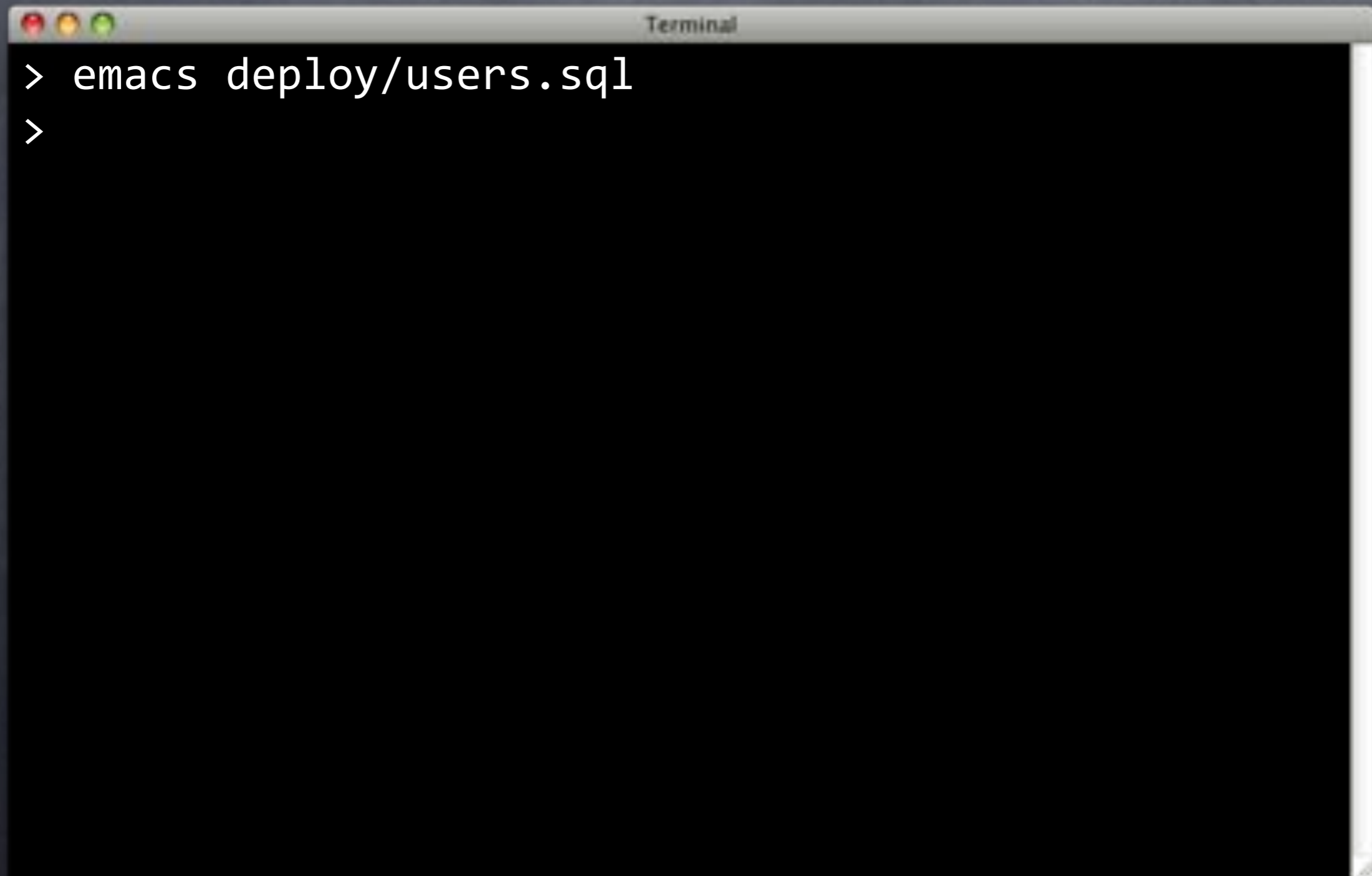
```
-- Deploy users
-- requires: appschema

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;
```

# User Typography

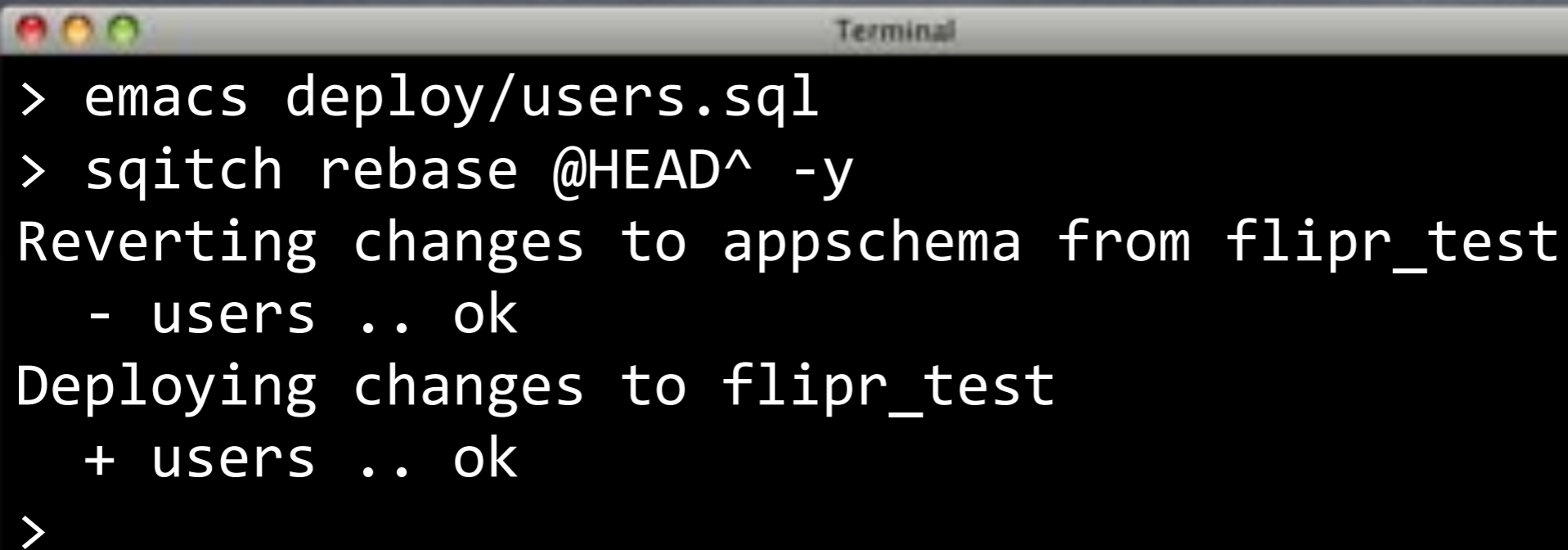
A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows a prompt character ">" followed by the command "emacs deploy/users.sql" on the first line, and another prompt character ">" on the second line.

```
> emacs deploy/users.sql  
>
```

# 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
>
```

# 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
>
```

# 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
>
```

# User Typography

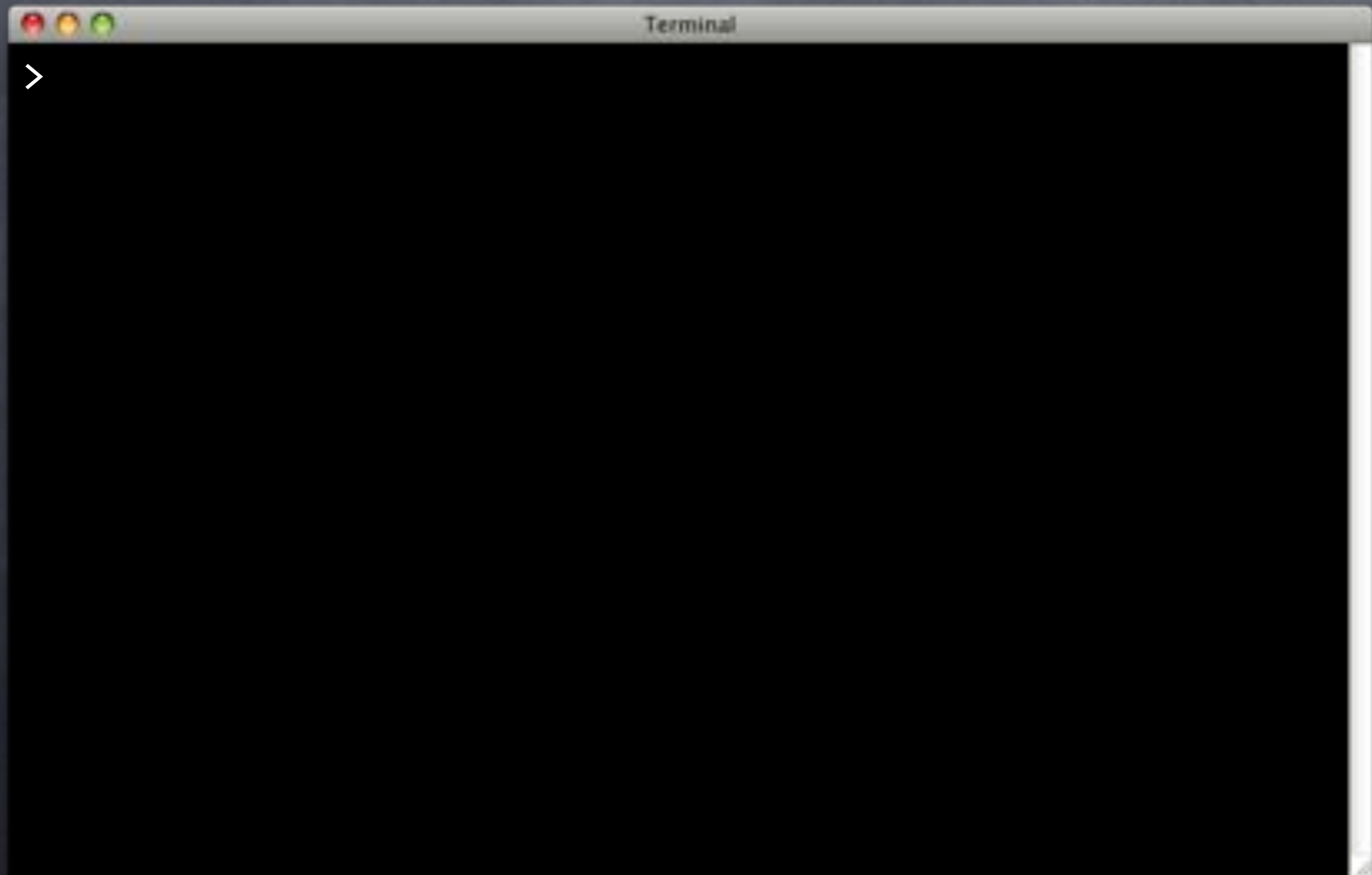
```
Terminal
> emacs deploy/users.sql
> sqitch rebase @HEAD^ -y
Reverting changes to appschema from flipr_test
- users .. ok
Deploying changes to flipr_test
+ users .. ok
> pg_prove -d flipr_test test/users.sql
test/users.sql .. ok
All tests successful.
Files=1, Tests=15, 0 wallclock secs
Result: PASS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

# 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
>
```

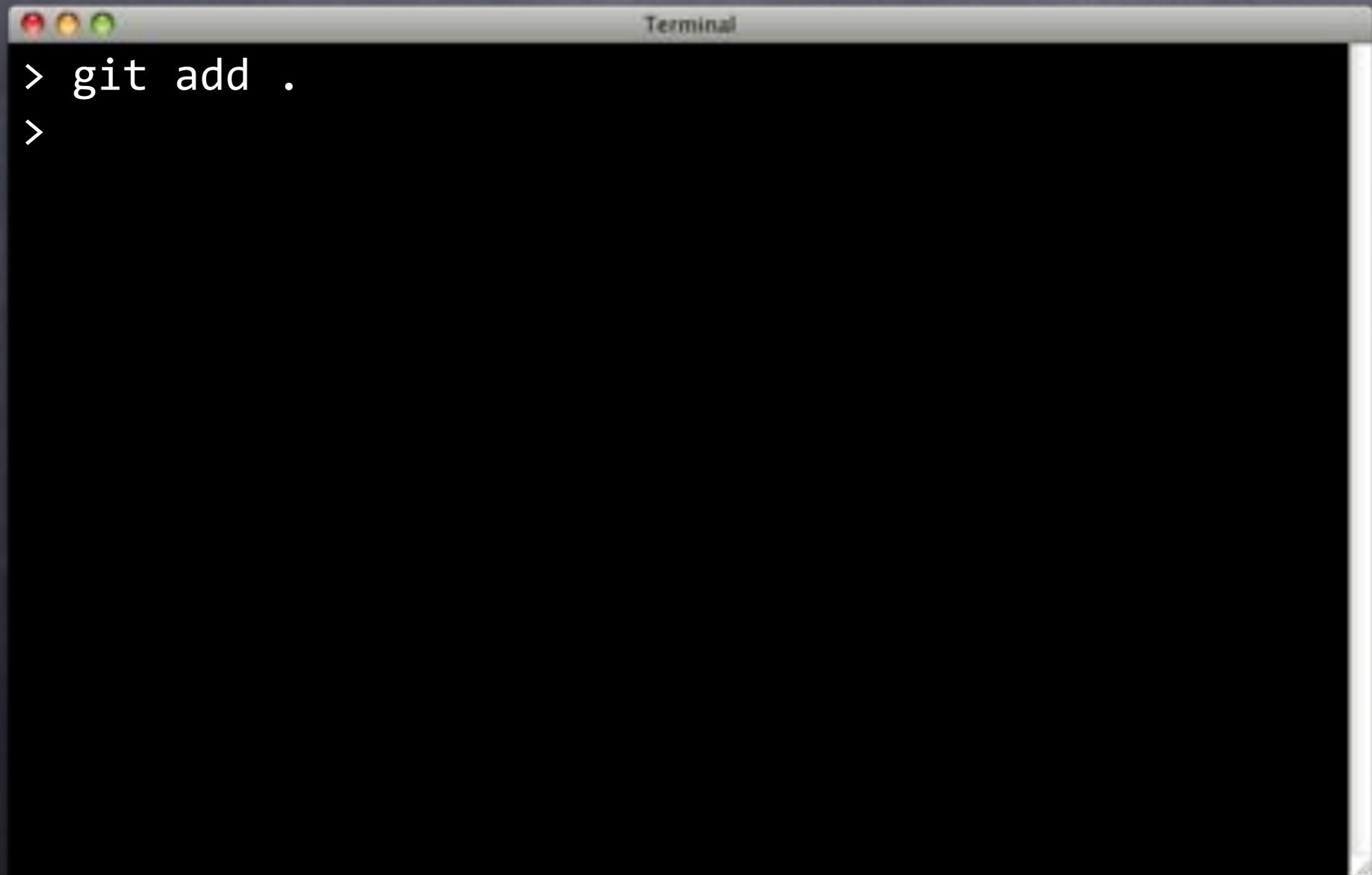
**Boom.**

# Additives



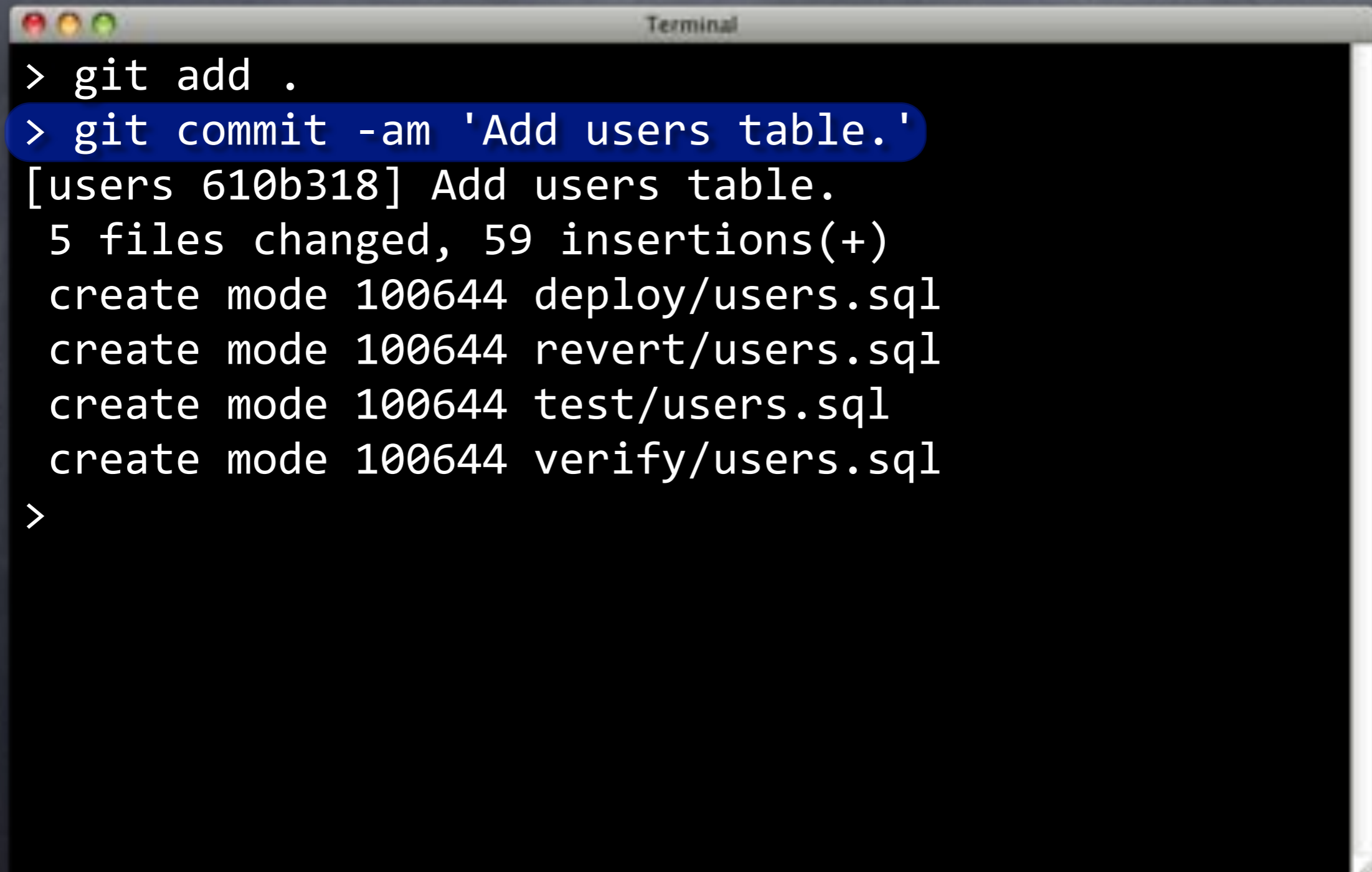


# Additives

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a prompt character followed by the command "git add ." and another prompt character on the next line.

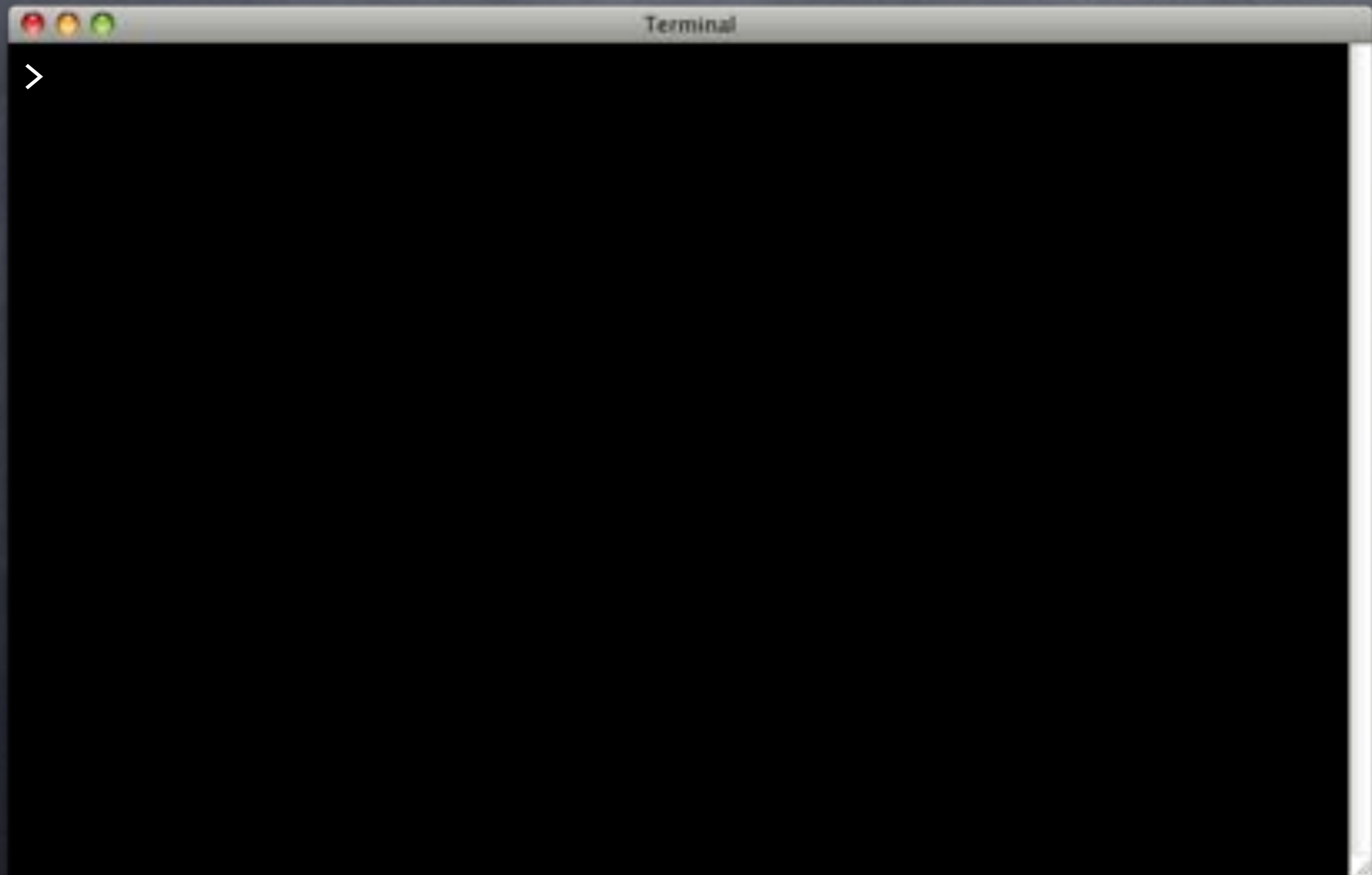
```
> git add .  
>
```

# Additives

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows the following text:

```
> 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

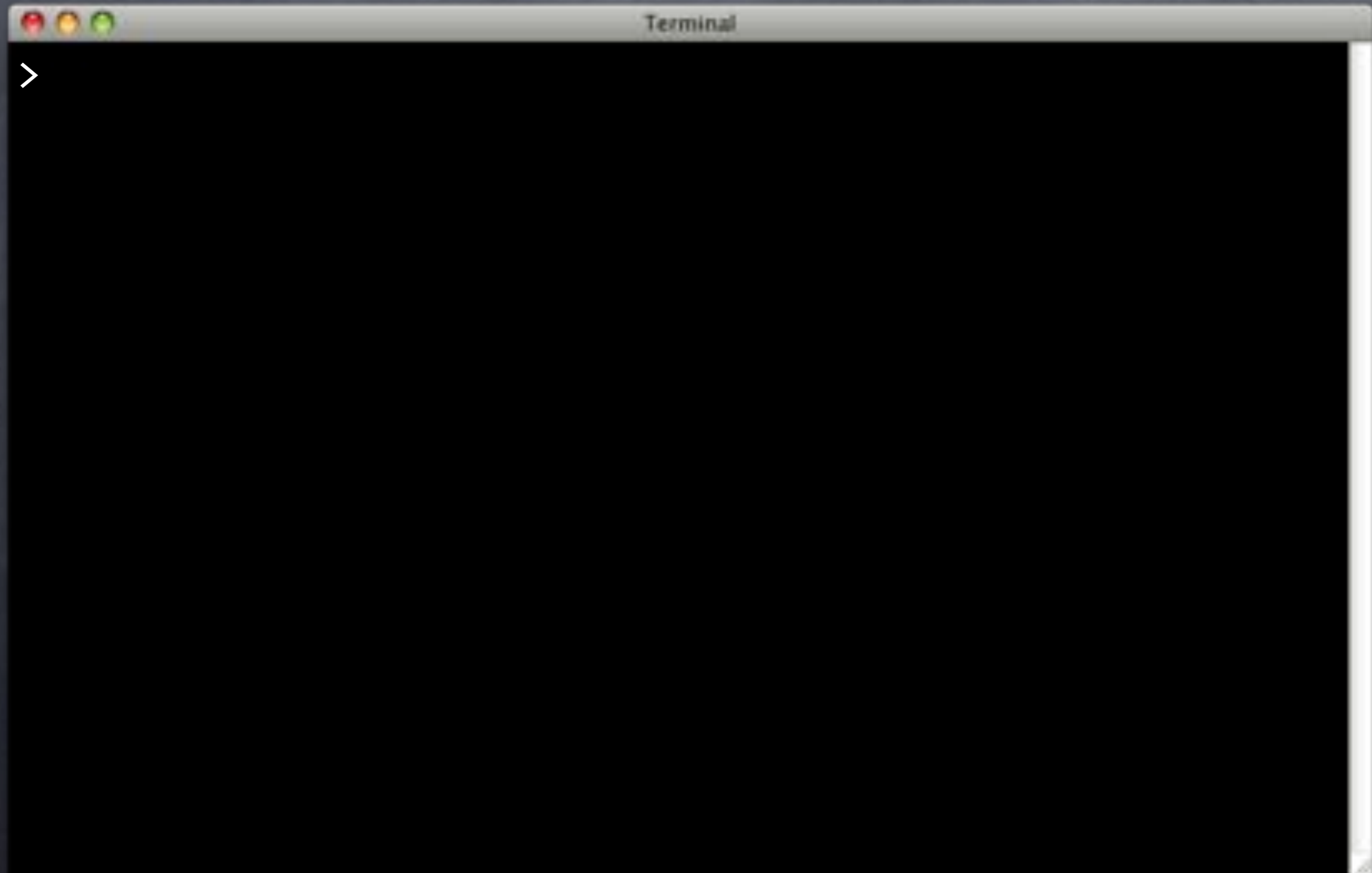


# Flip Out

- Create flips branch
- Create flips table
  - flip\_id SERIAL PK
  - nickname FK
  - body TEXT
  - timestamptz
- Use TDD
- <https://github.com/theory/agile-flipr.git>



# Functional Testing



# Functional Testing

Branches  
from users

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



# 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

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

# test/insert\_user.sql

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

# test/insert\_user.sql

```
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\_user.sql

```
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);
```

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

```
test/insert_use All (SQL[ansi])
```



```
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'  
);
```

```
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'  
);
```

```
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'
);
```

```
SELECT plan(11);

SELECT has_function( 'insert_user' );

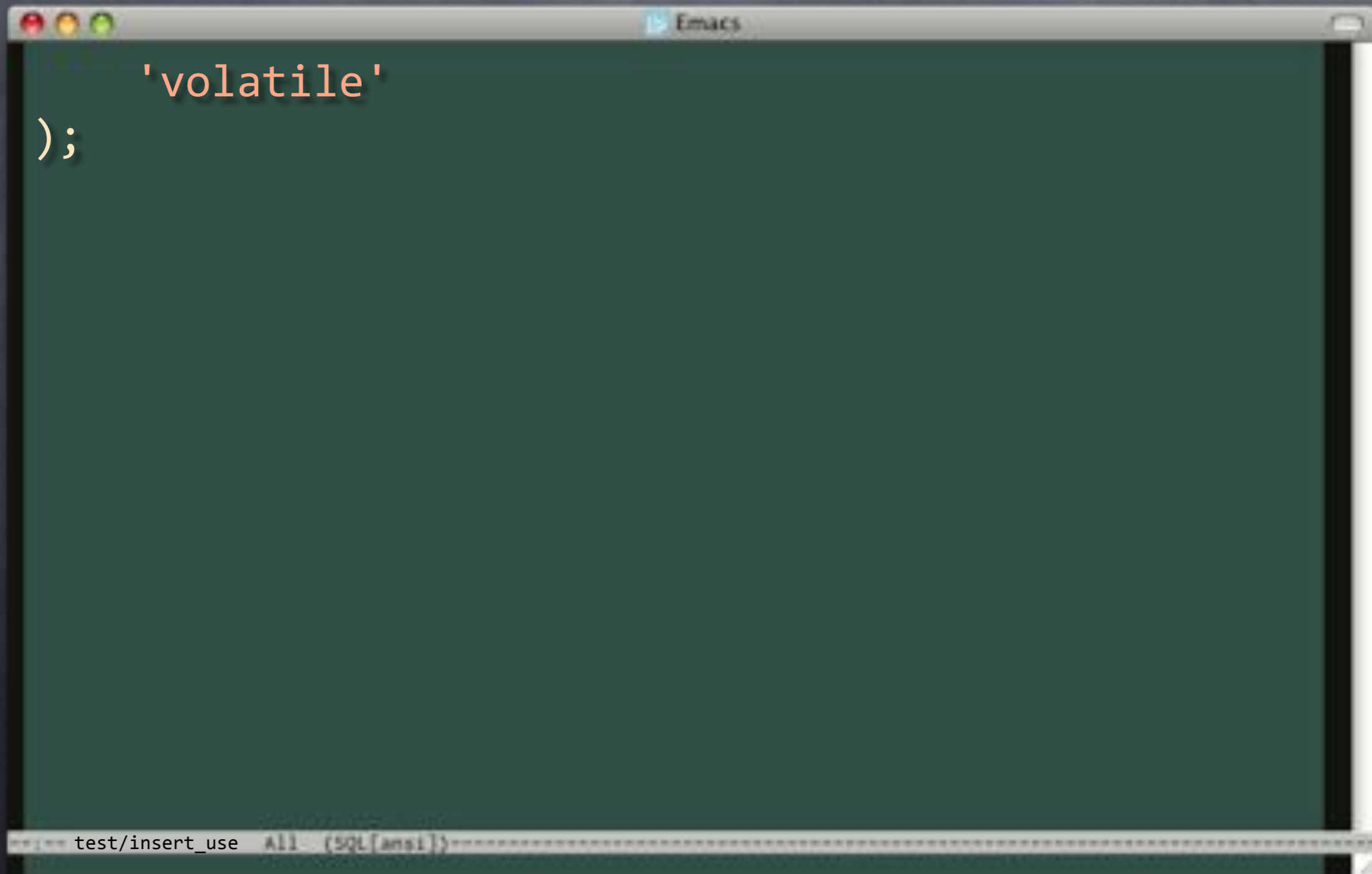
SELECT has_function(
    'insert_user', ARRAY['text', 'text']
);
SELECT function_lang_is(
    'insert_user', ARRAY['text', 'text'],
    'sql'
);
SELECT function_returns(
    'insert_user', ARRAY['text', 'text'],
    'void'
);
SELECT volatility_is(
    'insert_user', ARRAY['text', 'text'],
    'volatile'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

```
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\_user.sql



The image shows a screenshot of an Emacs editor window. The window title bar indicates the file path 'test/insert\_user.sql'. The editor content displays the following SQL code:

```
'volatile'  
);
```

The status bar at the bottom of the window shows the file path 'test/insert\_use', the buffer name 'All', and the encoding '(SQL[ansi])'.

# test/insert\_user.sql

```
        '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\_user.sql

```
        '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\_user.sql

```
        '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\_user.sql

```
        '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])

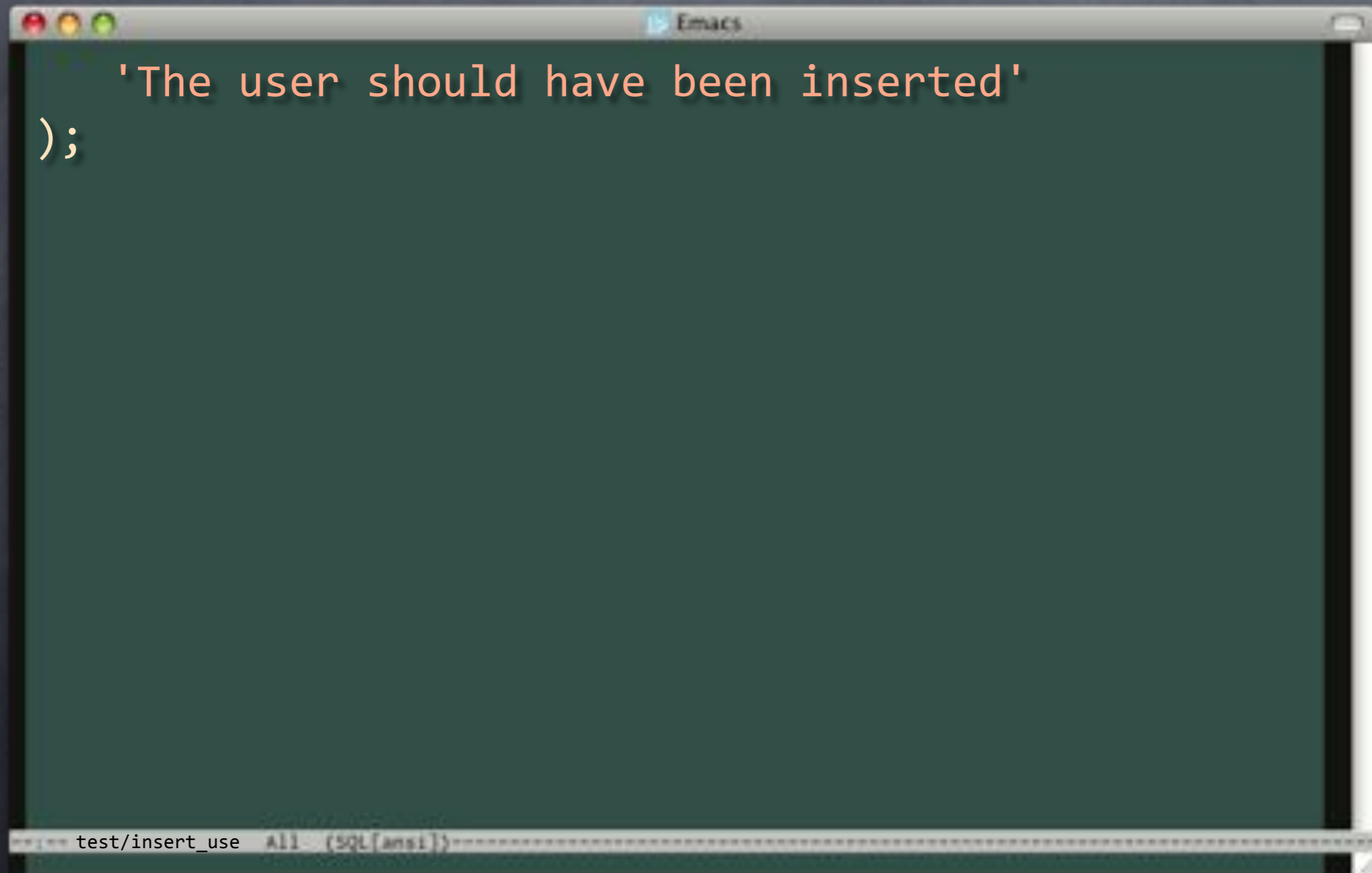
# test/insert\_user.sql

```
        '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\_user.sql

A screenshot of an Emacs editor window. The window title bar shows "Emacs" and standard macOS window controls (red, yellow, green buttons). The main editing area has a dark green background and contains the following SQL code:

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

The status bar at the bottom of the window displays "test/insert\_use All (SQL[ansi])".

# test/insert\_user.sql

```
'The user should have been inserted'
);
SELECT lives_ok(
  $$ SELECT insert_user('strongrr1', 'w00t') $$,
  'Insert another user'
);
SELECT bag_eq(
  'SELECT * FROM users',
  $$ VALUES
    ('theory', md5('foo'), NOW()),
    ('strongrr1', md5('w00t'), NOW())
  $$,
  'Both users should be present'
);
```

test/insert\_use All (SQL[ansi])

# test/insert\_user.sql

```
'The user should have been inserted'
);
SELECT lives_ok(
  $$ SELECT insert_user('strongrr1', 'w00t') $$,
  'Insert another user'
);
SELECT bag_eq(
  'SELECT * FROM users',
  $$ VALUES
    ('theory', md5('foo'), NOW()),
    ('strongrr1', md5('w00t'), NOW())
  $$,
  'Both users should be present'
);
```

test/insert\_use All (SQL[ansi])

# test/insert\_user.sql

```
'The user should have been inserted'
);
SELECT lives_ok(
  $$ SELECT insert('strongrr1', 'w00t') $$,
  'Insert another user'
);
SELECT bag_eq(
  'SELECT * FROM users',
  $$ VALUES
    ('theory', md5('foo'), NOW()),
    ('strongrr1', md5('w00t'), NOW())
  $$,
  'Both users should be present'
);
```

Includes  
dupes

# test/insert\_user.sql

```
'The user should have been inserted'
);
SELECT lives_ok(
  $$ SELECT insert_user('strongrr1', 'w00t') $$,
  'Insert another user'
);
SELECT bag_eq(
  'SELECT * FROM users',
  $$ VALUES
    ('theory', md5('foo'), NOW()),
    ('strongrr1', md5('w00t'), NOW())
  $$,
  'Both users should be present'
);
```

test/insert\_use All (SQL[ansi])



# test/insert\_user.sql

```
'The user should have been inserted'
);
SELECT lives_ok(
  $$ SELECT insert_user('strongrr1', 'w00t') $$,
  'Insert another user'
);
SELECT bag_eq(
  'SELECT * FROM users',
  $$ VALUES
    ('theory',      md5('foo'),      NOW()),
    ('strongrr1',  md5('w00t'),      NOW())
  $$,
  'Both users should be present'
);
```

test/insert\_use All (SQL[ansi])

```
Emacs  
'Both users should be present'  
);  
  
test/insert_use All (SQL[ansi])
```

```
    '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',
    $$ VALUES
        ('theory', md5('foo'), NOW()),
        ('strongrr1', md5('w00t'), NOW())
    $$,
    'Should still have just the two users'
);
```

```
    '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',
    $$ VALUES
        ('theory', md5('foo'), NOW()),
        ('strongrr1', md5('w00t'), NOW())
    $$,
    'Should still have just the two users'
);
```

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

SELECT throws_ok(
    $$ SELECT insert_user('theory', 'ha-1', '2013-01-01',
23505, -- duplicate key violation
NULL, -- localized error message
    'Should get an error for duplicate nickname'
);

SELECT bag_eq(
    'SELECT * FROM users',
    $$ VALUES
        ('theory', md5('foo'), NOW()),
        ('strongrr1', md5('w00t'), NOW())
    $$,
    'Should still have just the two users'
);
```

## Appendix A

```
    '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',
    $$ VALUES
        ('theory', md5('foo'), NOW()),
        ('strongrr1', md5('w00t'), NOW())
    $$,
    'Should still have just the two users'
);
```

```
    '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',
  $$ VALUES
    ('theory', md5('foo'), NOW()),
    ('strongrr1', md5('w00t'), NOW())
  $$,
  'Should still have just the two users'
);
```

```

    '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',
    $$ VALUES
        ('theory', md5('foo'), NOW()),
        ('strongrr1', md5('w00t'), NOW())
    $$,
    'Should still have just the two users'
);

```

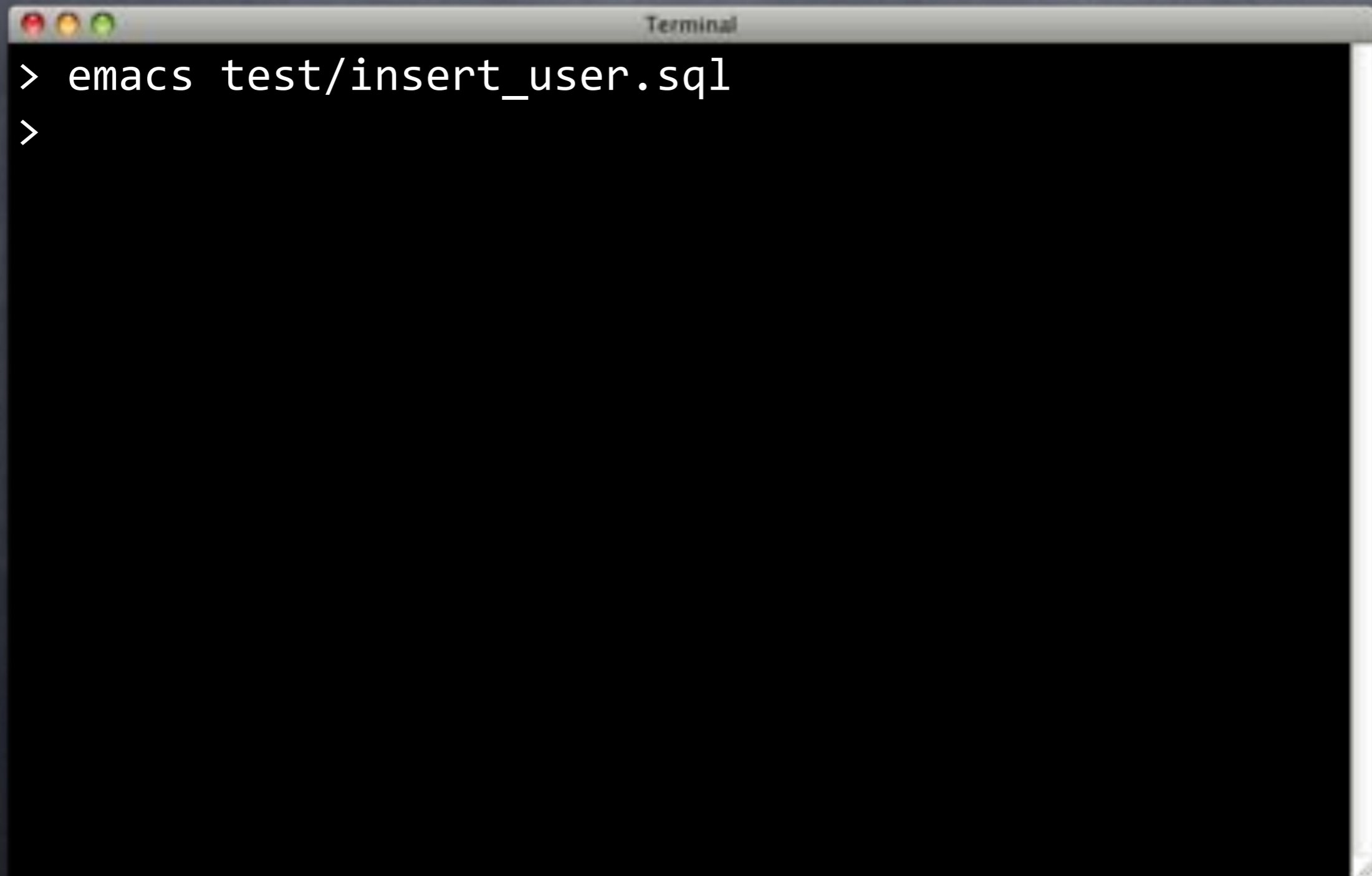


```
    'Both users should be present'
);

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

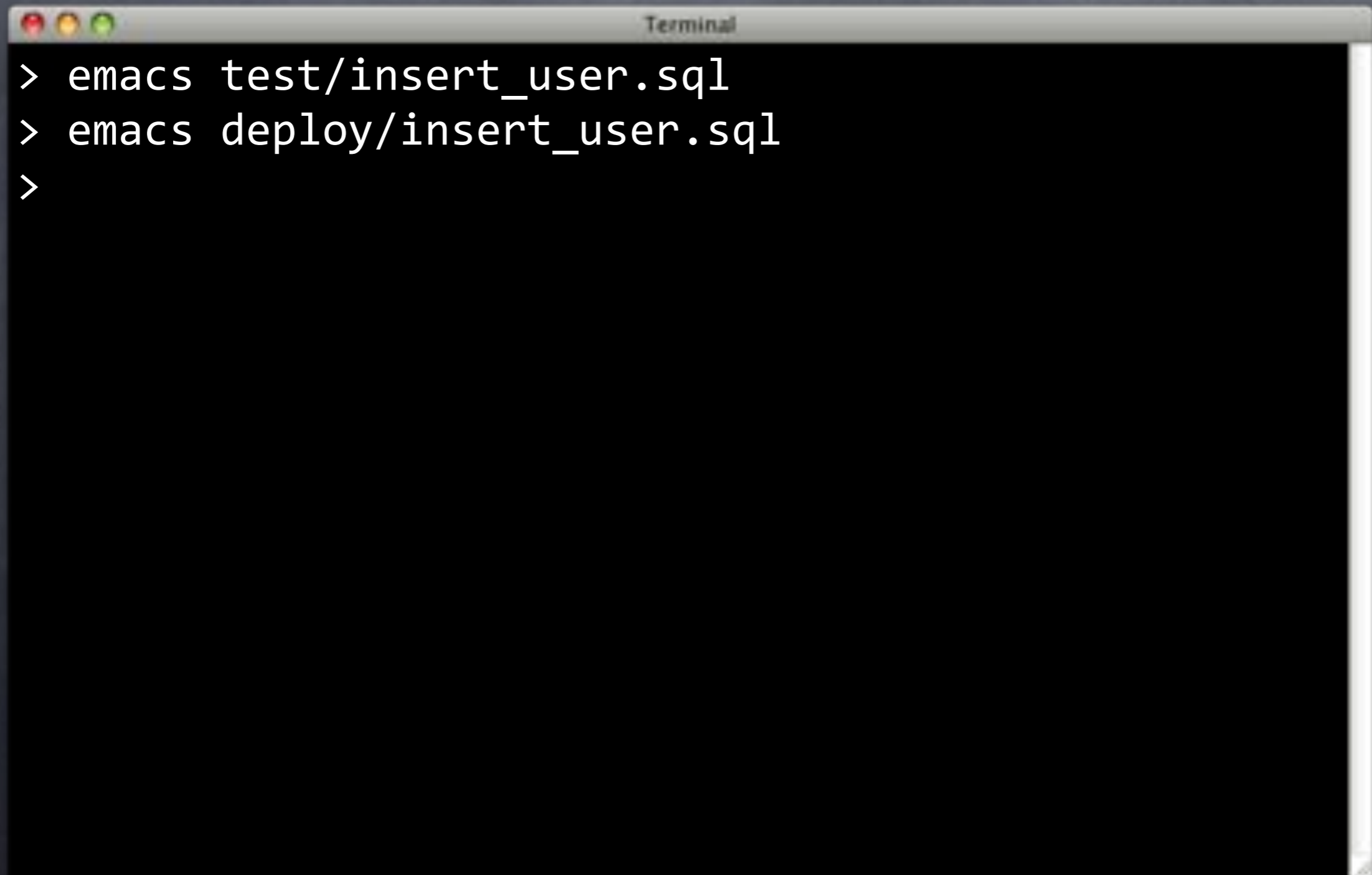
SELECT bag_eq(
    'SELECT * FROM users',
    $$ VALUES
        ('theory', md5('foo'), NOW()),
        ('strongrr1', md5('w00t'), NOW())
    $$,
    'Should still have just the two users'
);
SELECT finish();
ROLLBACK;
```

# Functional Testing

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows a prompt character followed by the command "emacs test/insert\_user.sql" and a second prompt character on the next line.

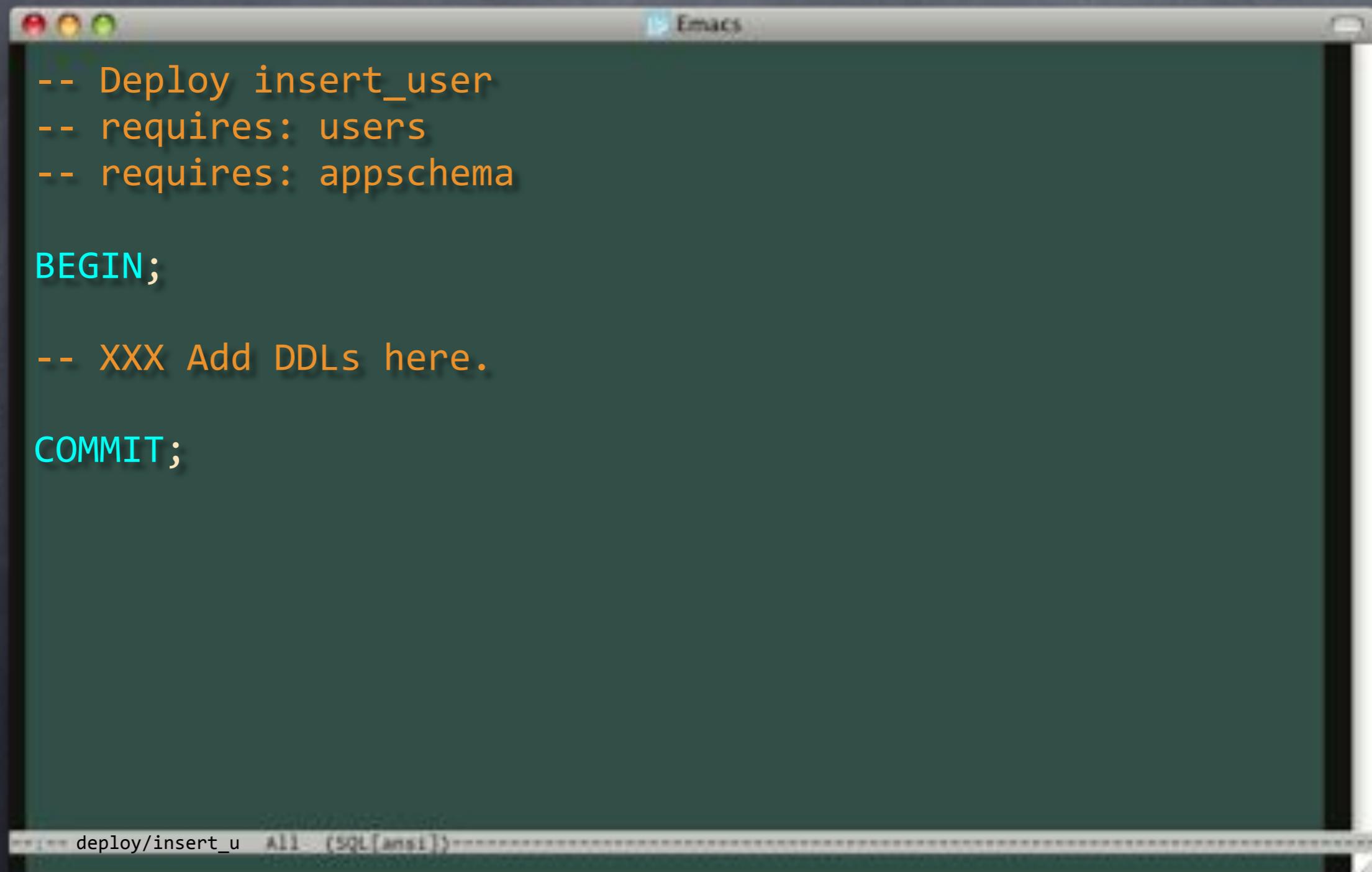
```
> emacs test/insert_user.sql  
>
```

# Functional Testing

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows three lines of shell commands, each preceded by a greater-than sign (>).

```
> emacs test/insert_user.sql
> emacs deploy/insert_user.sql
>
```

# deploy/insert\_user.sql

A screenshot of an Emacs editor window. The window title is "Emacs". The background is a dark green color. The text is as follows:

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

BEGIN;

-- XXX Add DDLs here.

COMMIT;
```

At the bottom of the window, there is a status bar with the text "deploy/insert\_u All (SQL[ansi])".

# deploy/insert\_user.sql

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

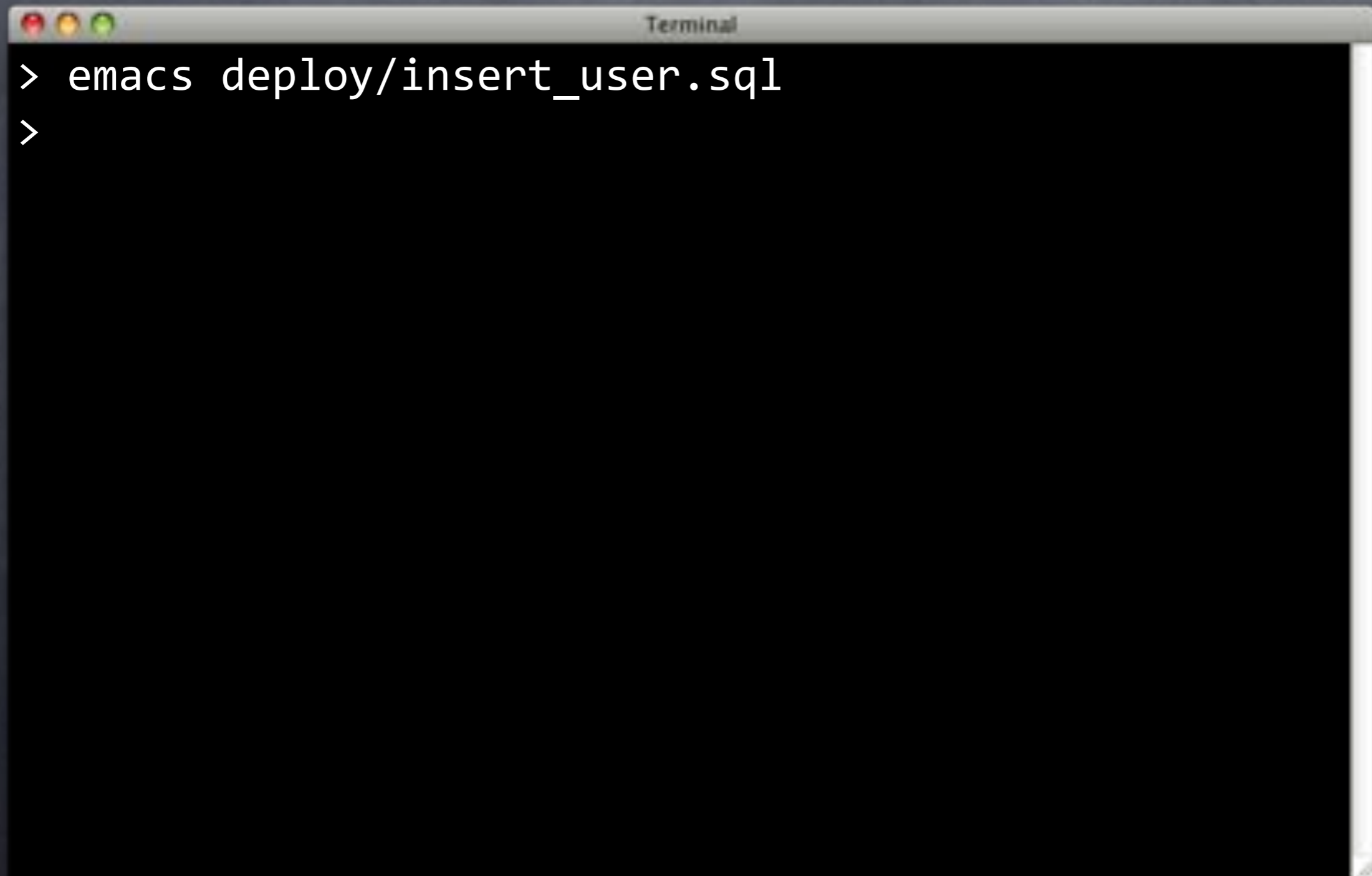
```
BEGIN;
```

```
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;
```

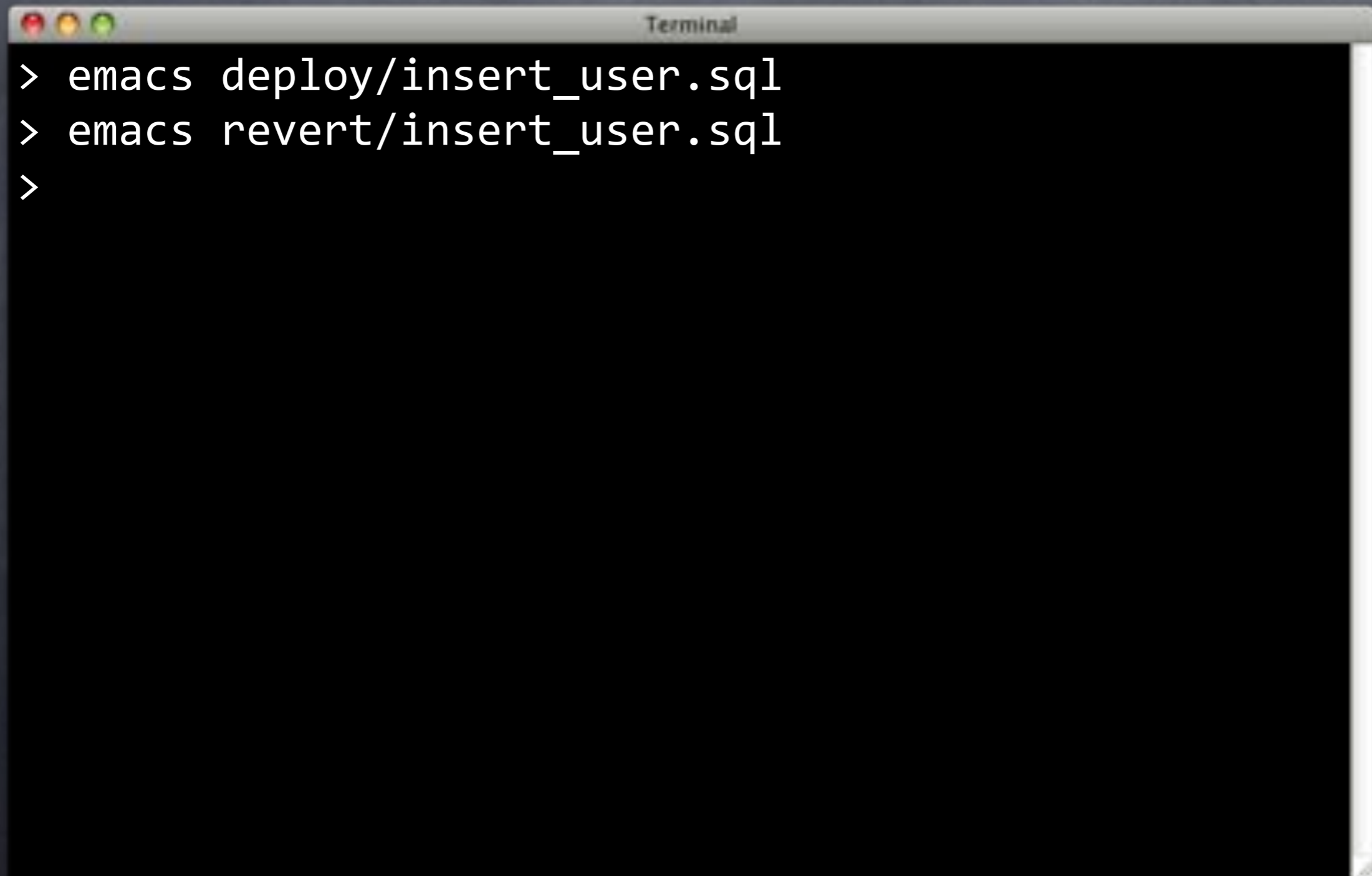
```
deploy/insert_u All (SQL[ansi])
```

# Functional Testing

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows a prompt character followed by the command "emacs deploy/insert\_user.sql" and a second prompt character on the next line.

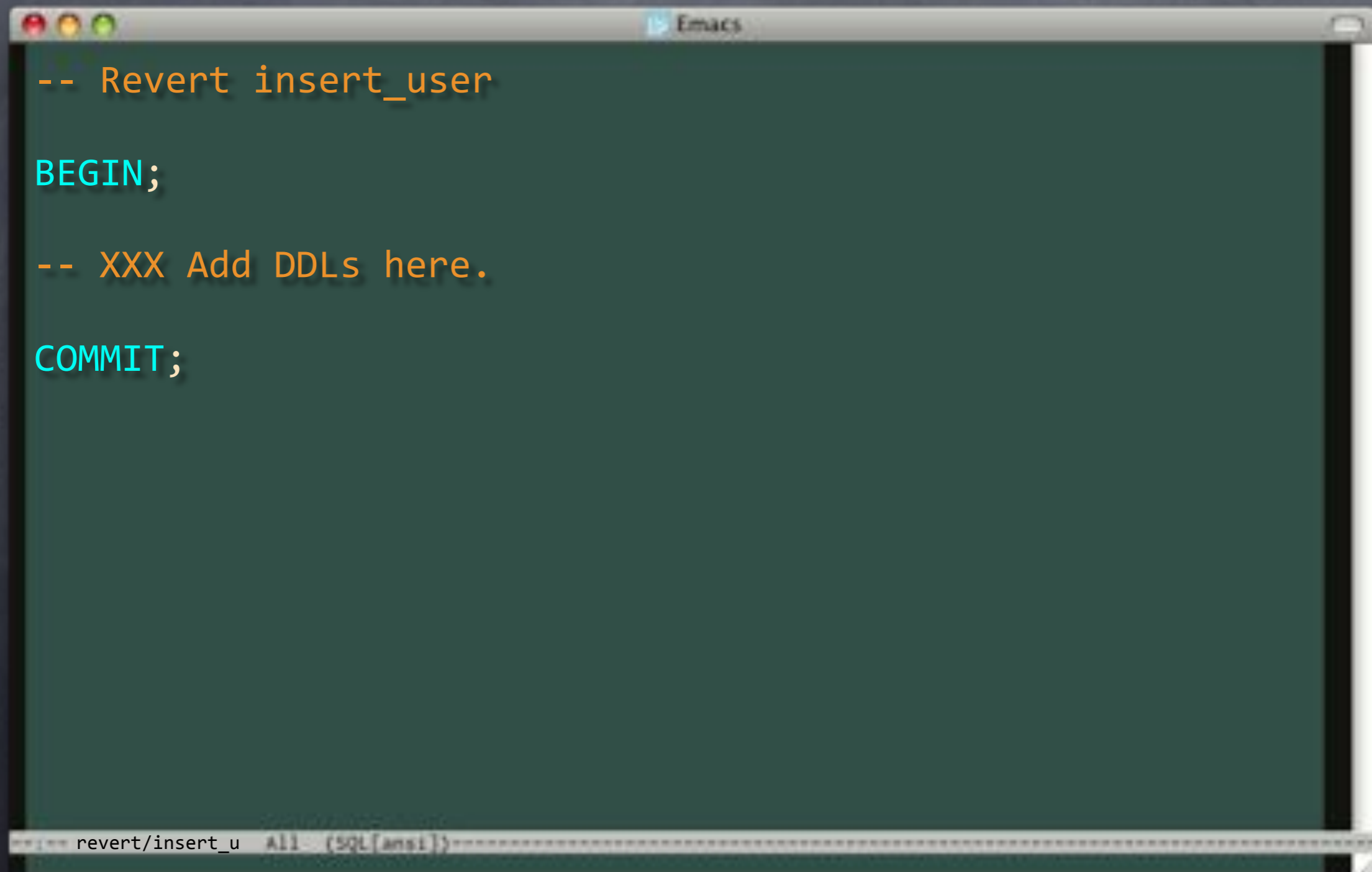
```
> emacs deploy/insert_user.sql  
>
```

# Functional Testing

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows three lines of commands, each starting with a greater-than sign (>).

```
> emacs deploy/insert_user.sql
> emacs revert/insert_user.sql
>
```

# revert/insert\_user.sql

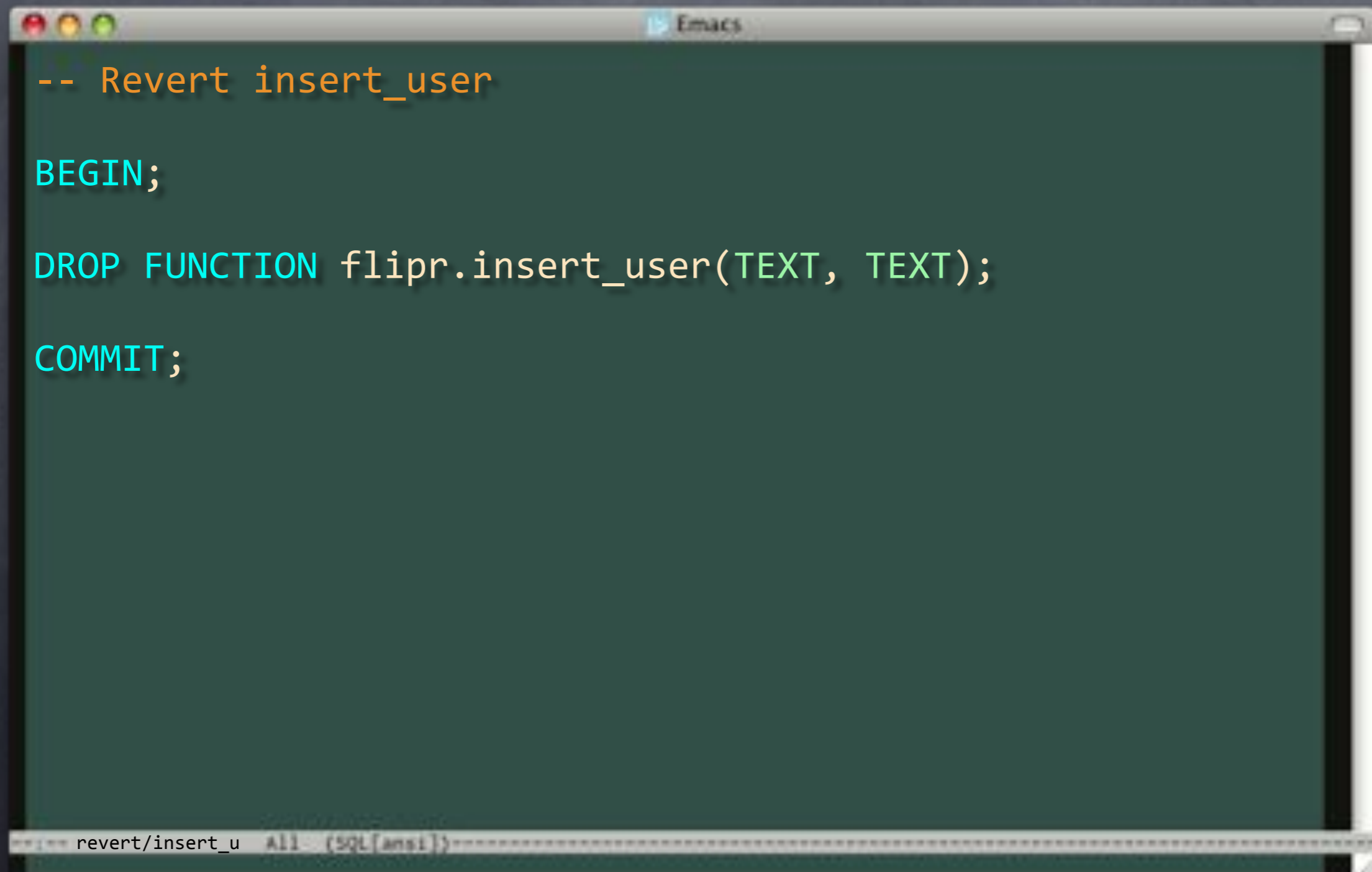
A screenshot of an Emacs editor window. The window title is "Emacs". The background is a dark green color. The text is as follows:

```
-- Revert insert_user  
  
BEGIN;  
  
-- XXX Add DDLs here.  
  
COMMIT;
```

The status bar at the bottom shows "revert/insert\_u All (SQL[ansi])".



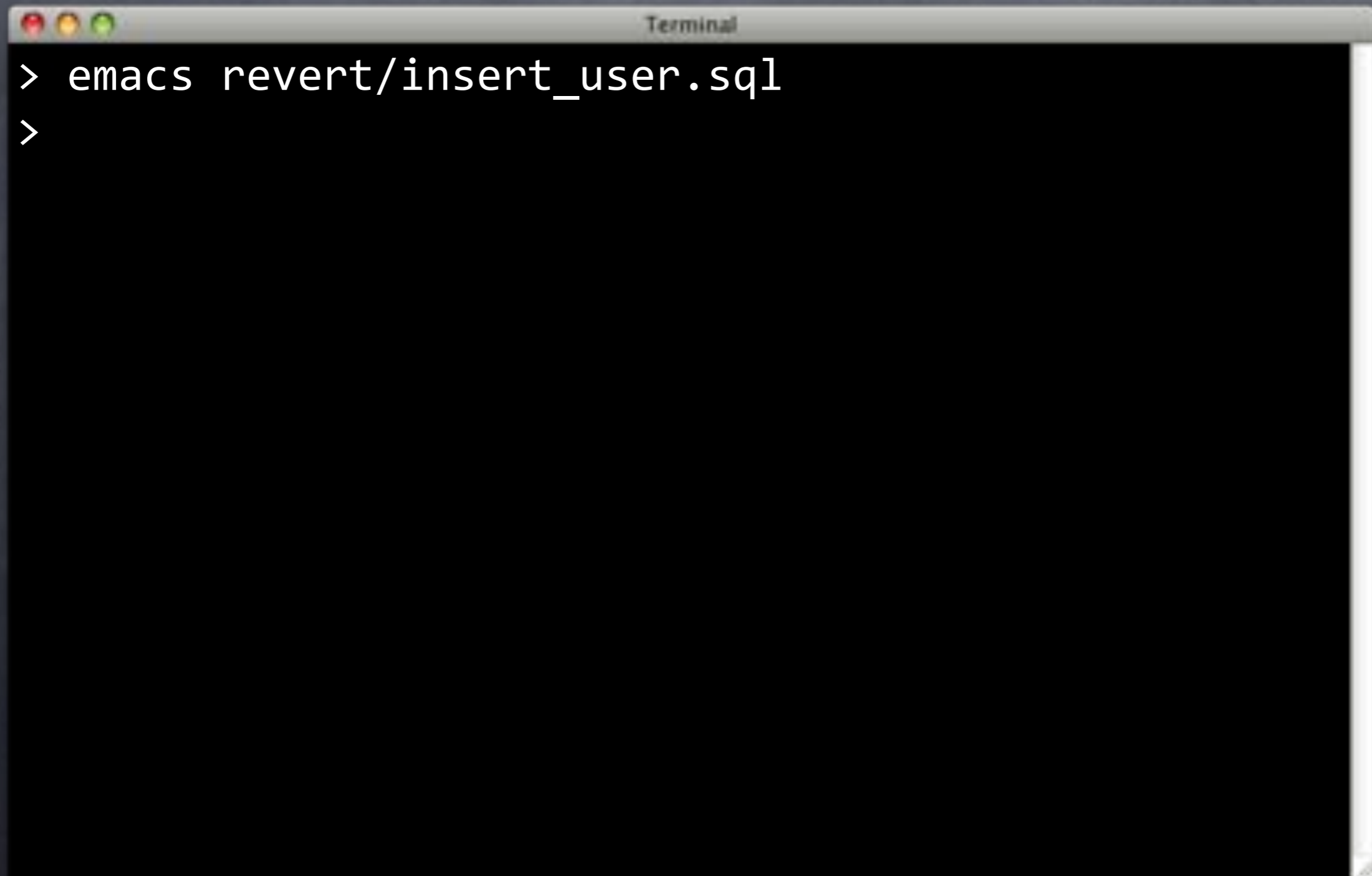
# revert/insert\_user.sql

A screenshot of an Emacs editor window. The window title is "Emacs". The background is a dark green color. The text is as follows:

```
-- Revert insert_user  
  
BEGIN;  
  
DROP FUNCTION flipr.insert_user(TEXT, TEXT);  
  
COMMIT;
```

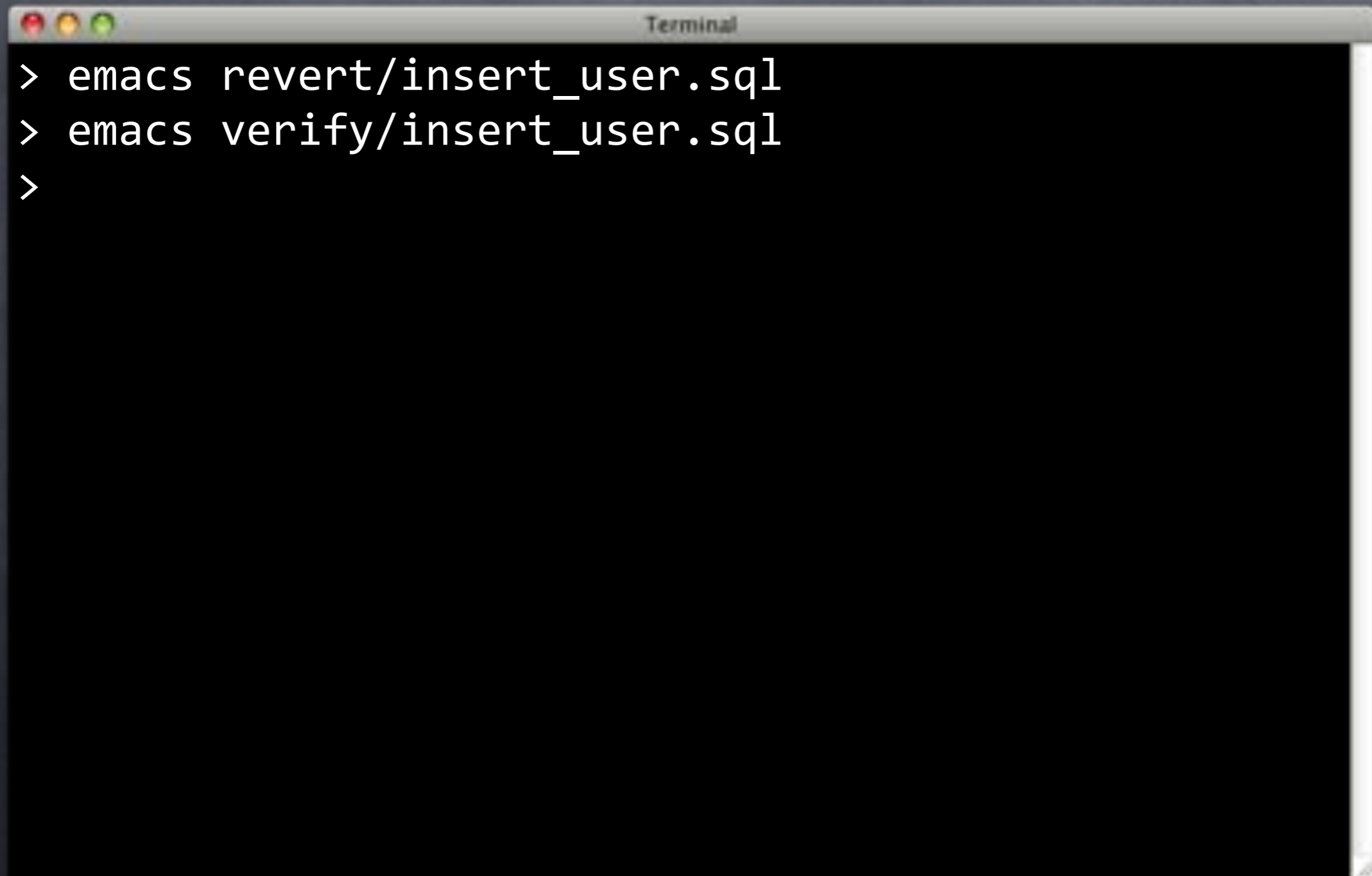
At the bottom of the window, there is a status bar with the text "revert/insert\_u All (SQL[ansi])".

# Functional Testing

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows a prompt character followed by the command "emacs revert/insert\_user.sql" and a second prompt character on the next line.

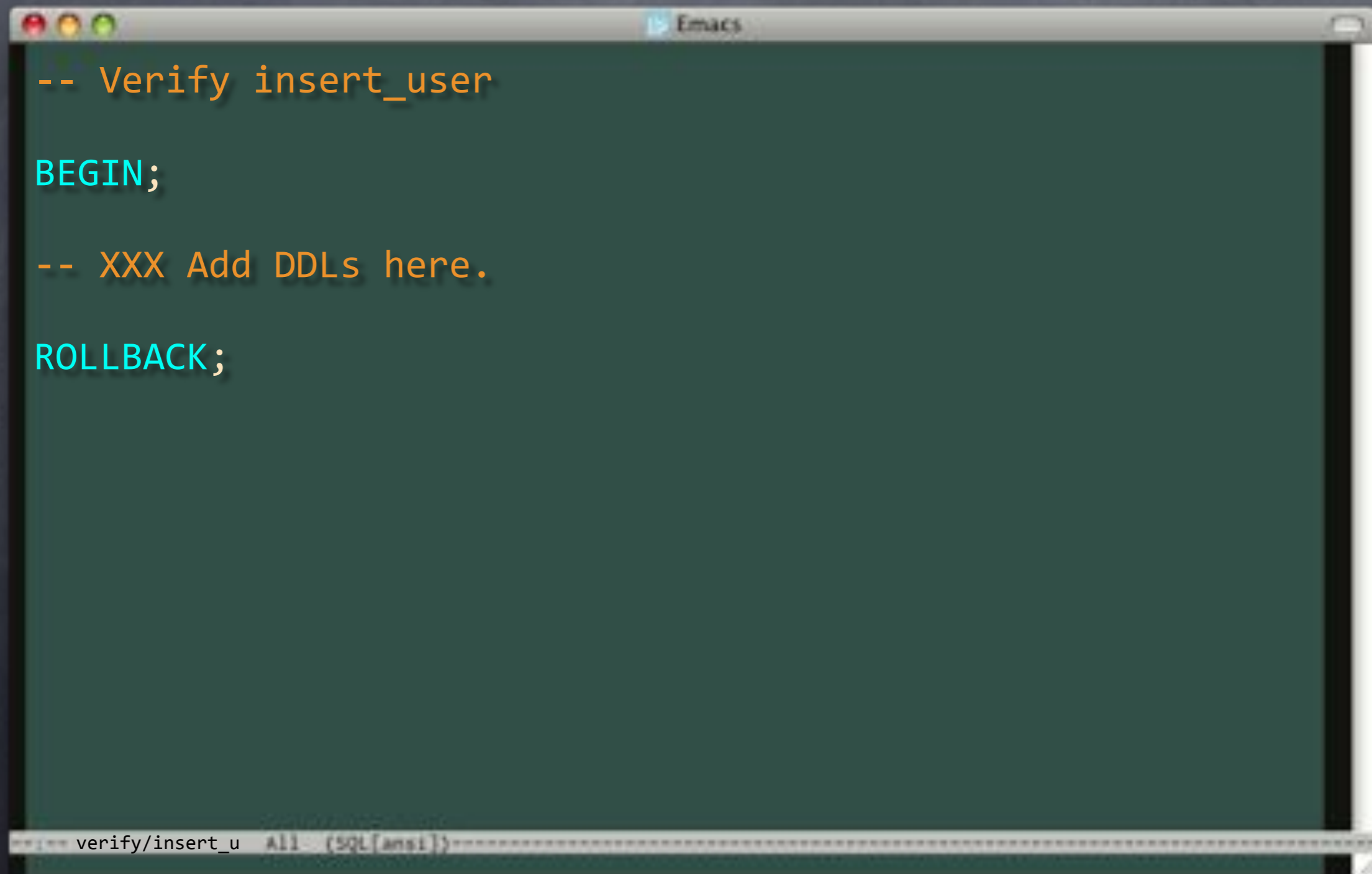
```
> emacs revert/insert_user.sql  
>
```

# Functional Testing

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows three lines of commands, each starting with a greater-than sign (>).

```
Terminal  
> emacs revert/insert_user.sql  
> emacs verify/insert_user.sql  
>
```

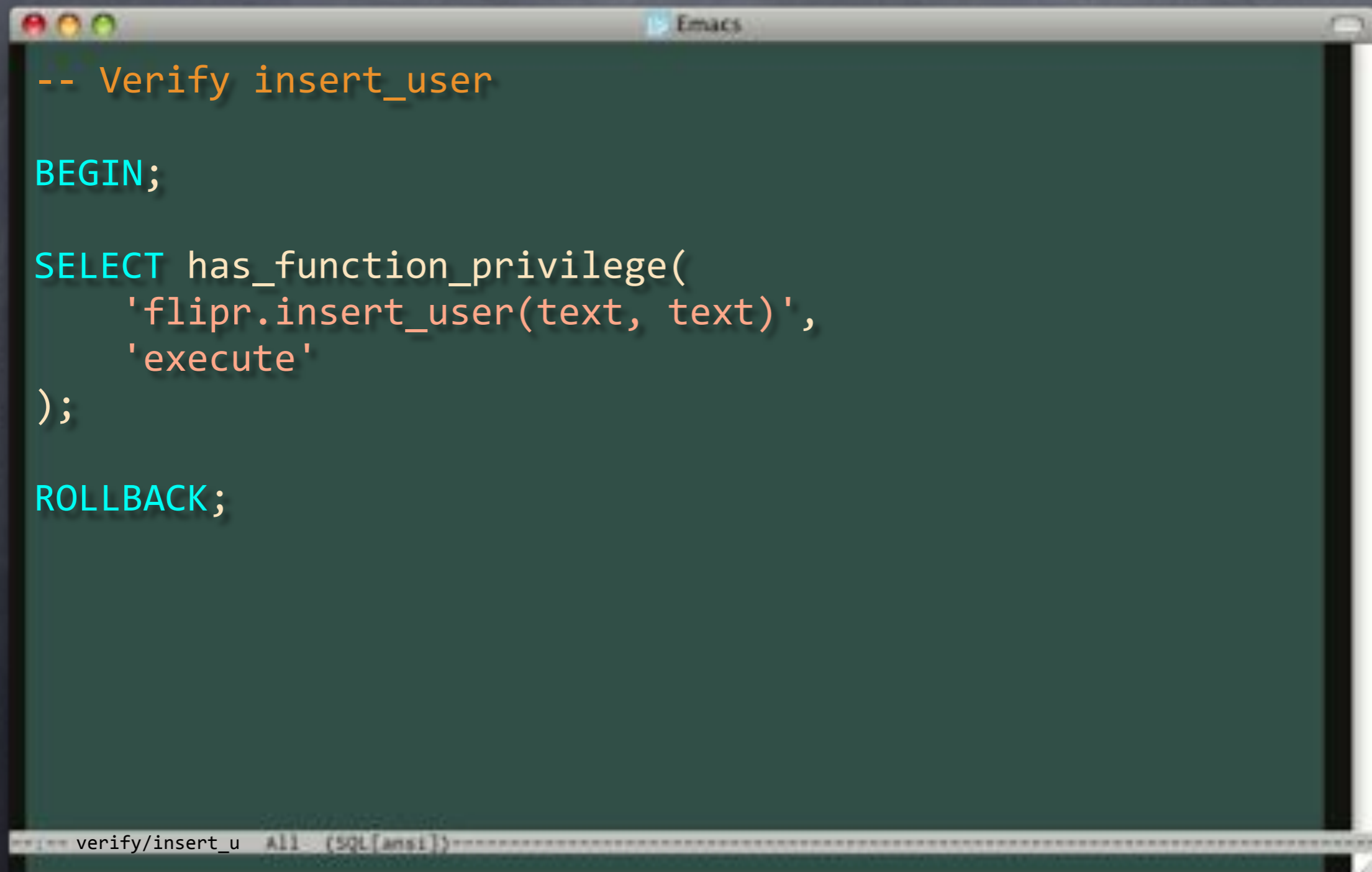
# verify/insert\_user.sql

A screenshot of an Emacs editor window. The window title bar shows the Emacs logo and the text "Emacs". The editor content is on a dark green background and contains the following SQL code:

```
-- Verify insert_user  
  
BEGIN;  
  
-- XXX Add DDLs here.  
  
ROLLBACK;
```

The status bar at the bottom of the window shows "verify/insert\_u" followed by "All (SQL[ansi])".

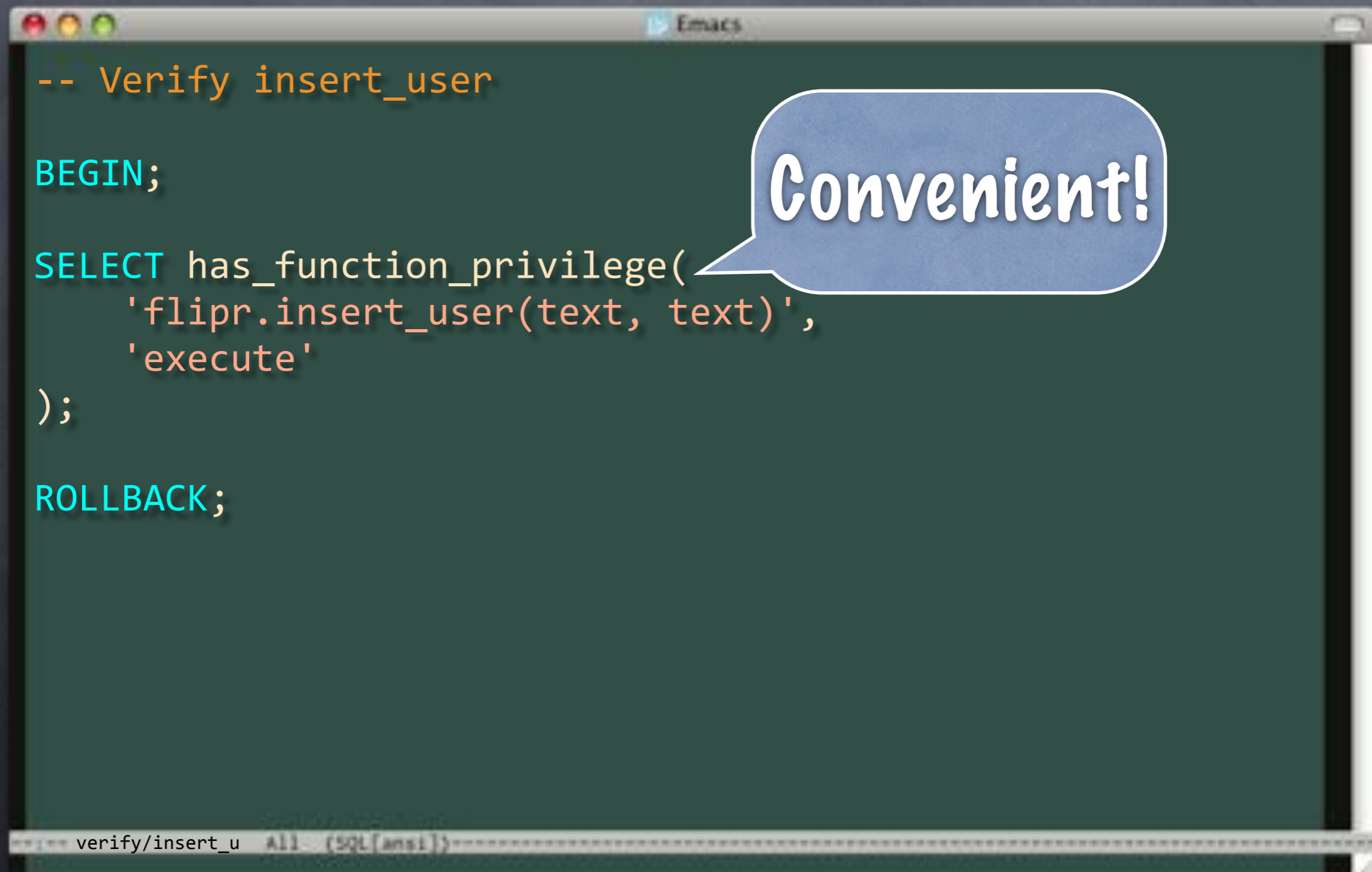
# verify/insert\_user.sql

A screenshot of an Emacs editor window. The window title is "Emacs". The background is a dark green color. The code is displayed in a light green font. The code is as follows:

```
-- Verify insert_user  
  
BEGIN;  
  
SELECT has_function_privilege(  
    'flipr.insert_user(text, text)',  
    'execute'  
);  
  
ROLLBACK;
```

The status bar at the bottom of the window shows "verify/insert\_u All (SQL[ansi])".

# verify/insert\_user.sql

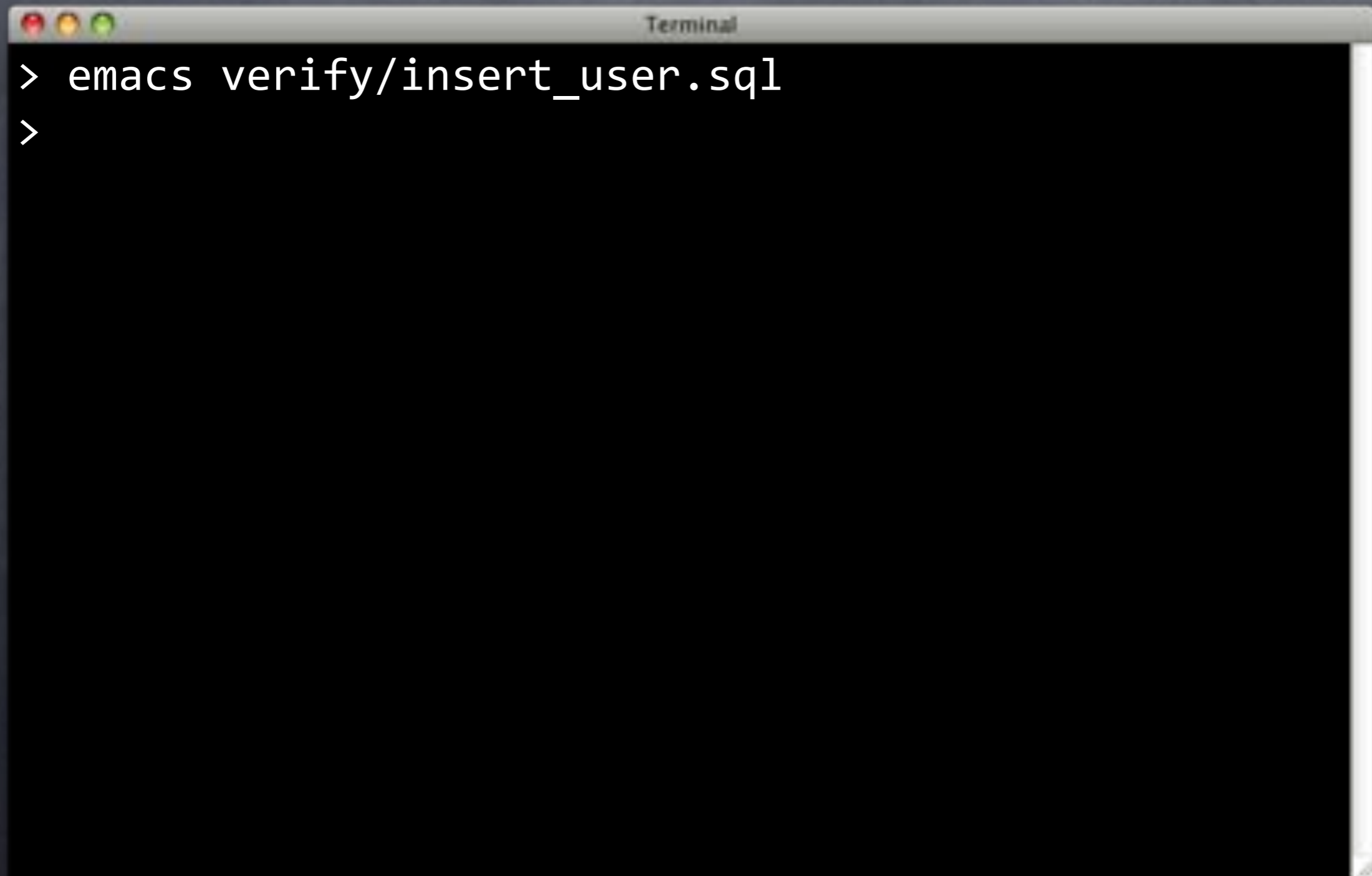


```
-- Verify insert_user  
  
BEGIN;  
  
SELECT has_function_privilege(  
    'flipr.insert_user(text, text)',  
    'execute'  
);  
  
ROLLBACK;
```

Convenient!

verify/insert\_u All (SQL[ansi])

# We Good?

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a prompt character ">" followed by the command "emacs verify/insert\_user.sql" on the first line, and a second prompt character ">" on the second line.

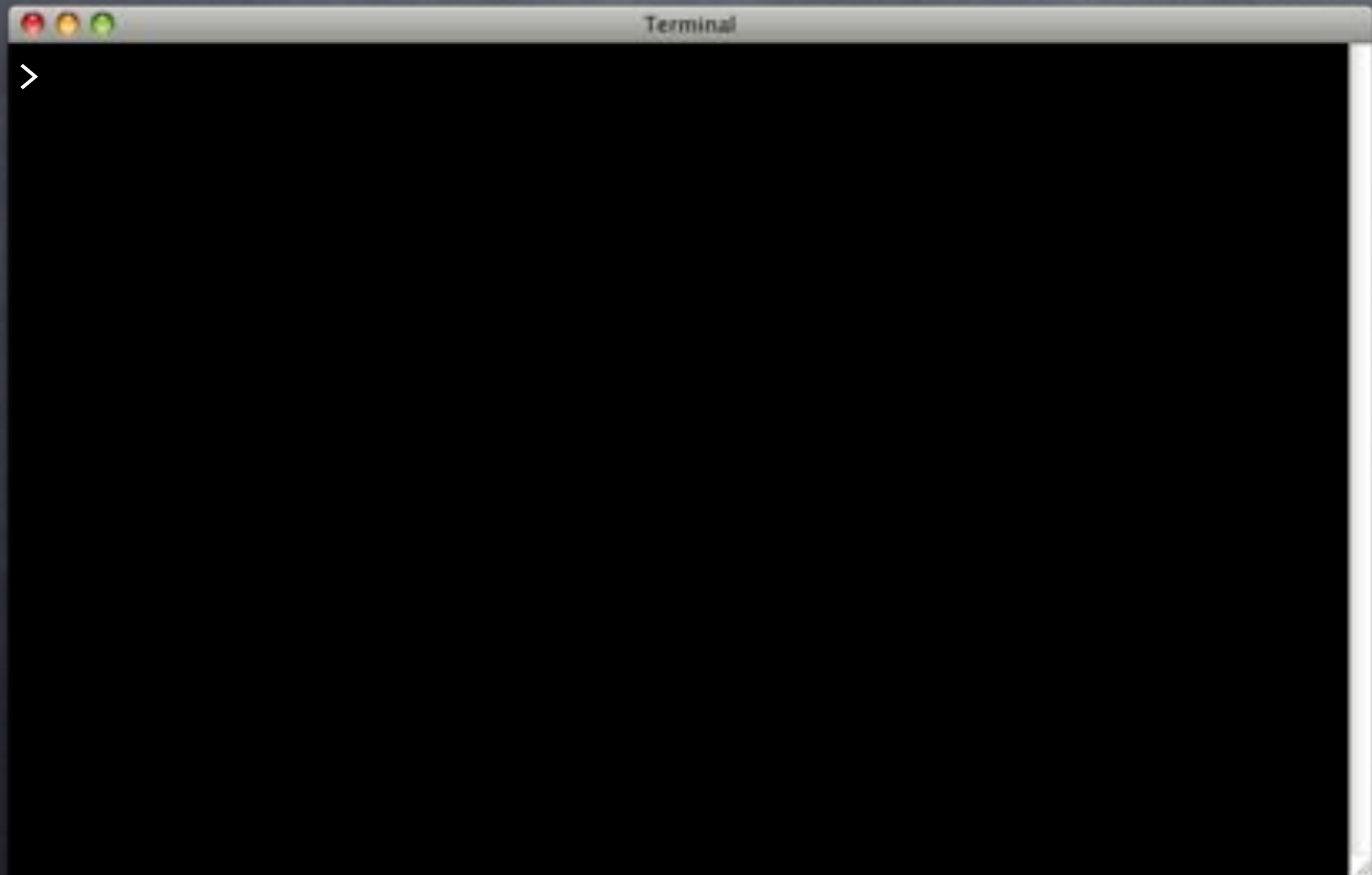
```
> emacs verify/insert_user.sql  
>
```

# 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
>
```



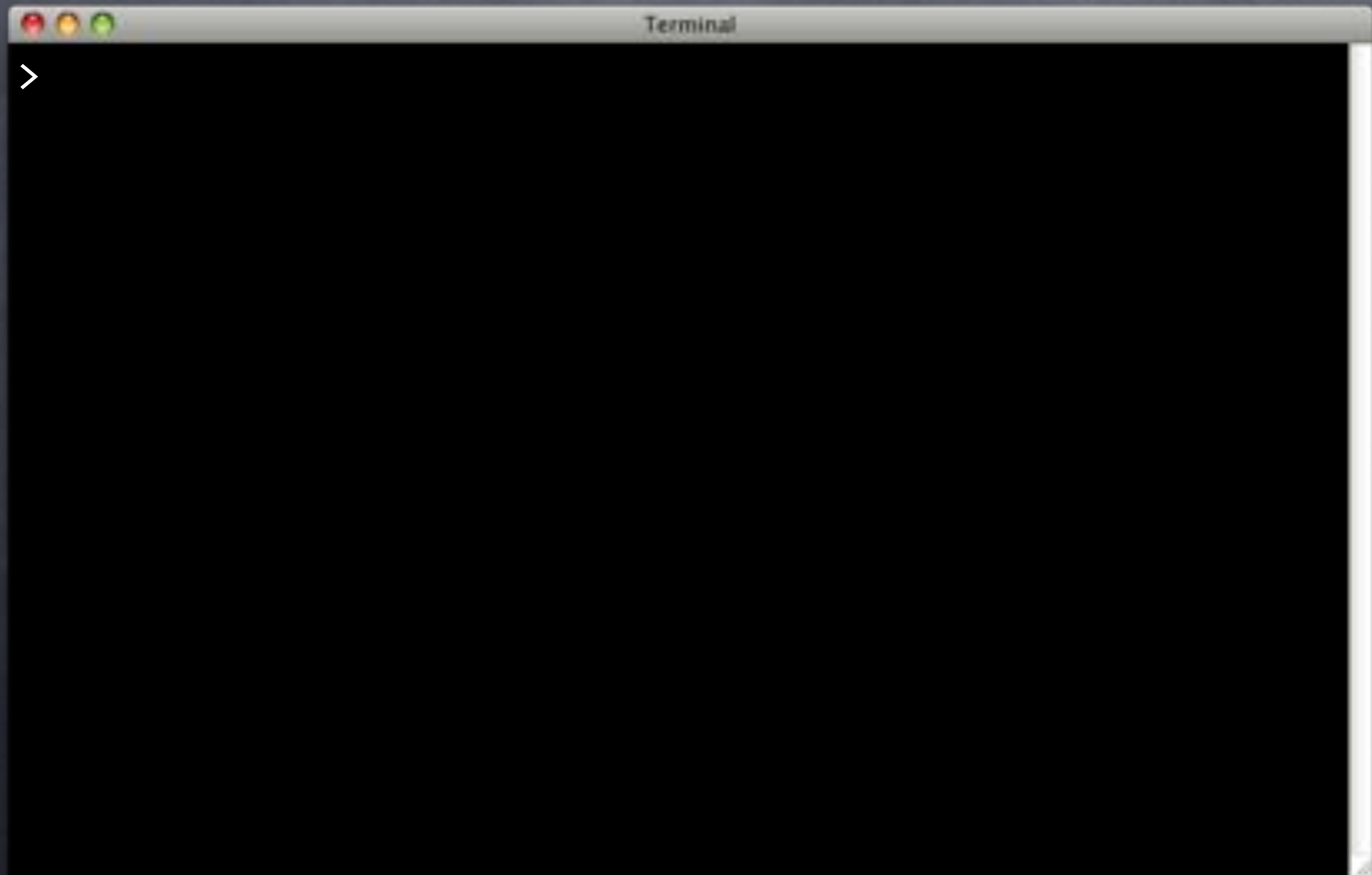
# Commitment



# 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 TDD

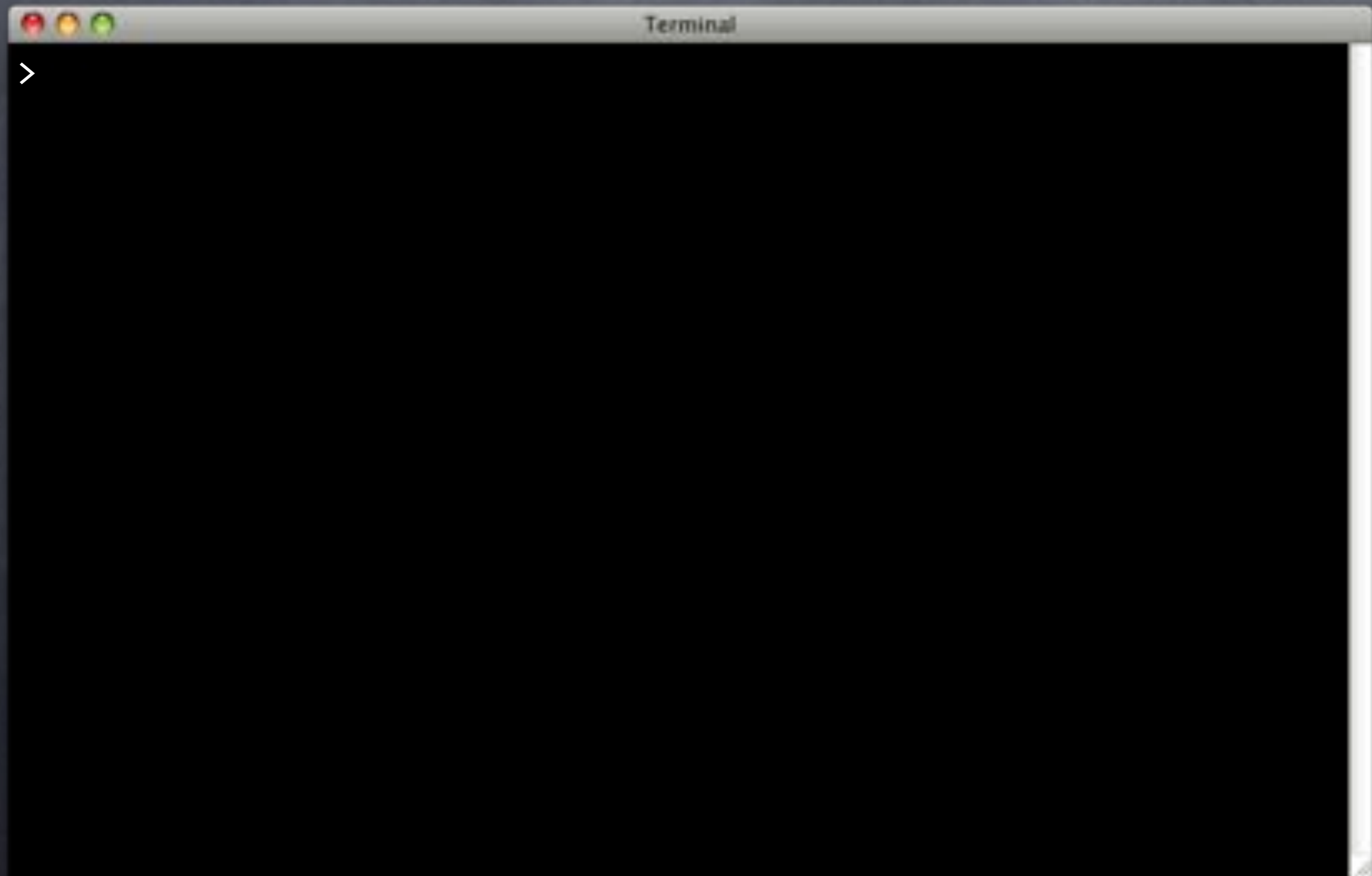


# None Shall Pass

- 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

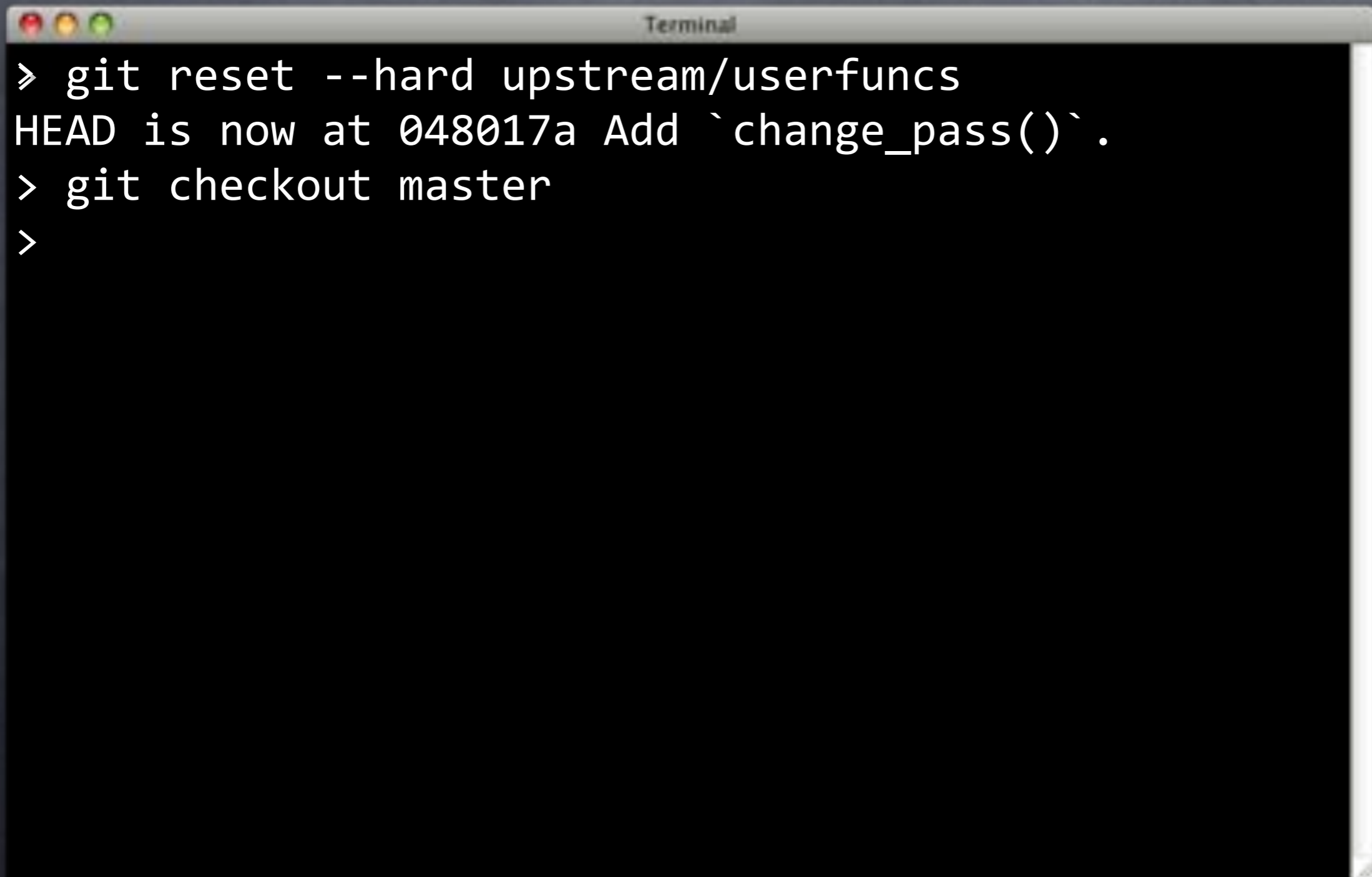


# Resets

My solution

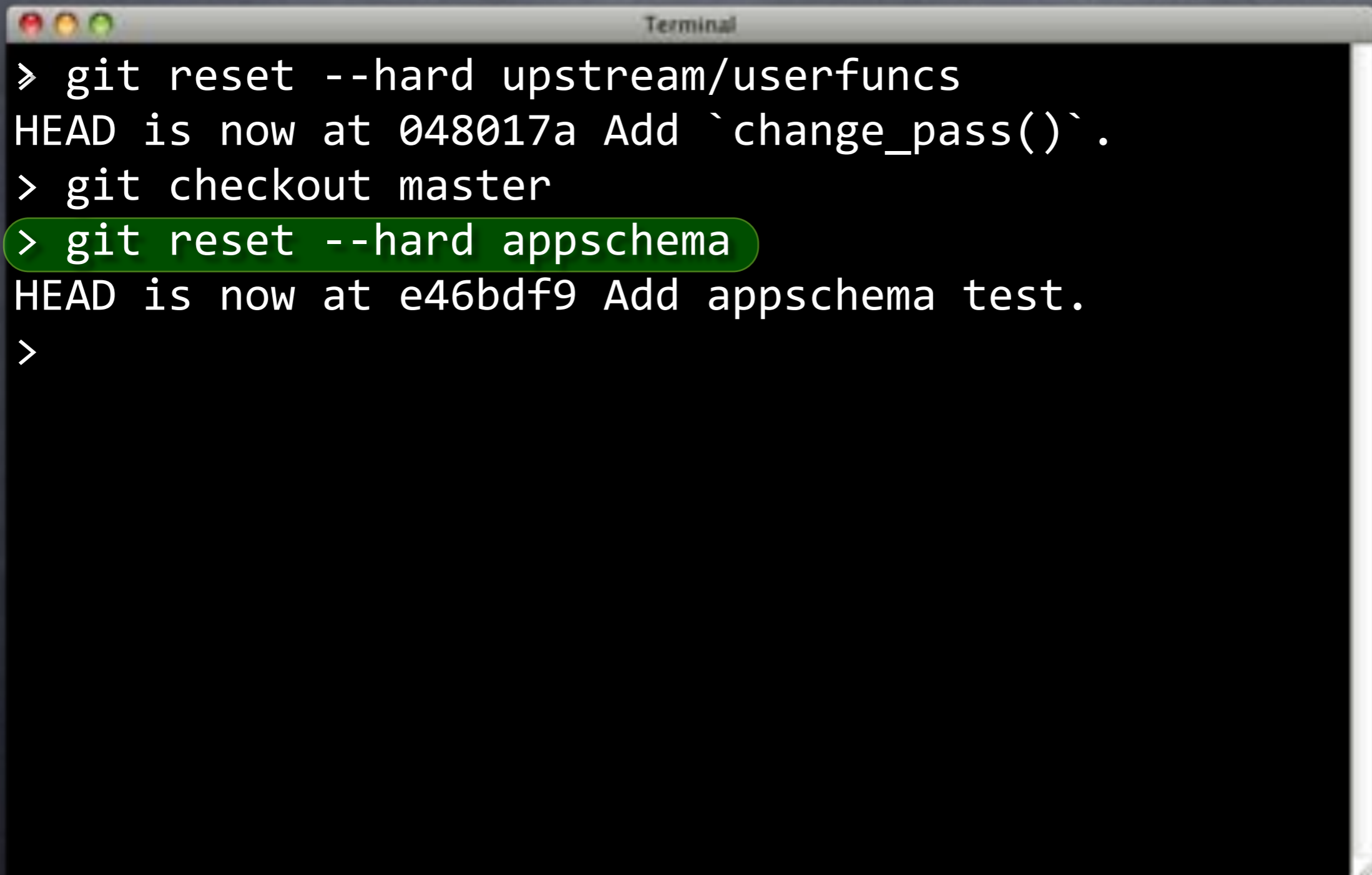
```
Terminal  
➤ git reset --hard upstream/userfuncs  
HEAD is now at 048017a Add `change_pass()`  
>
```

# Resets

A terminal window with a title bar that says "Terminal". The window contains the following text:

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

# Resets

A terminal window titled "Terminal" with a standard macOS window header (red, yellow, green buttons). The terminal shows a sequence of git commands and their outputs. The command `git reset --hard appschema` is highlighted with a green background.

```
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.
>
```

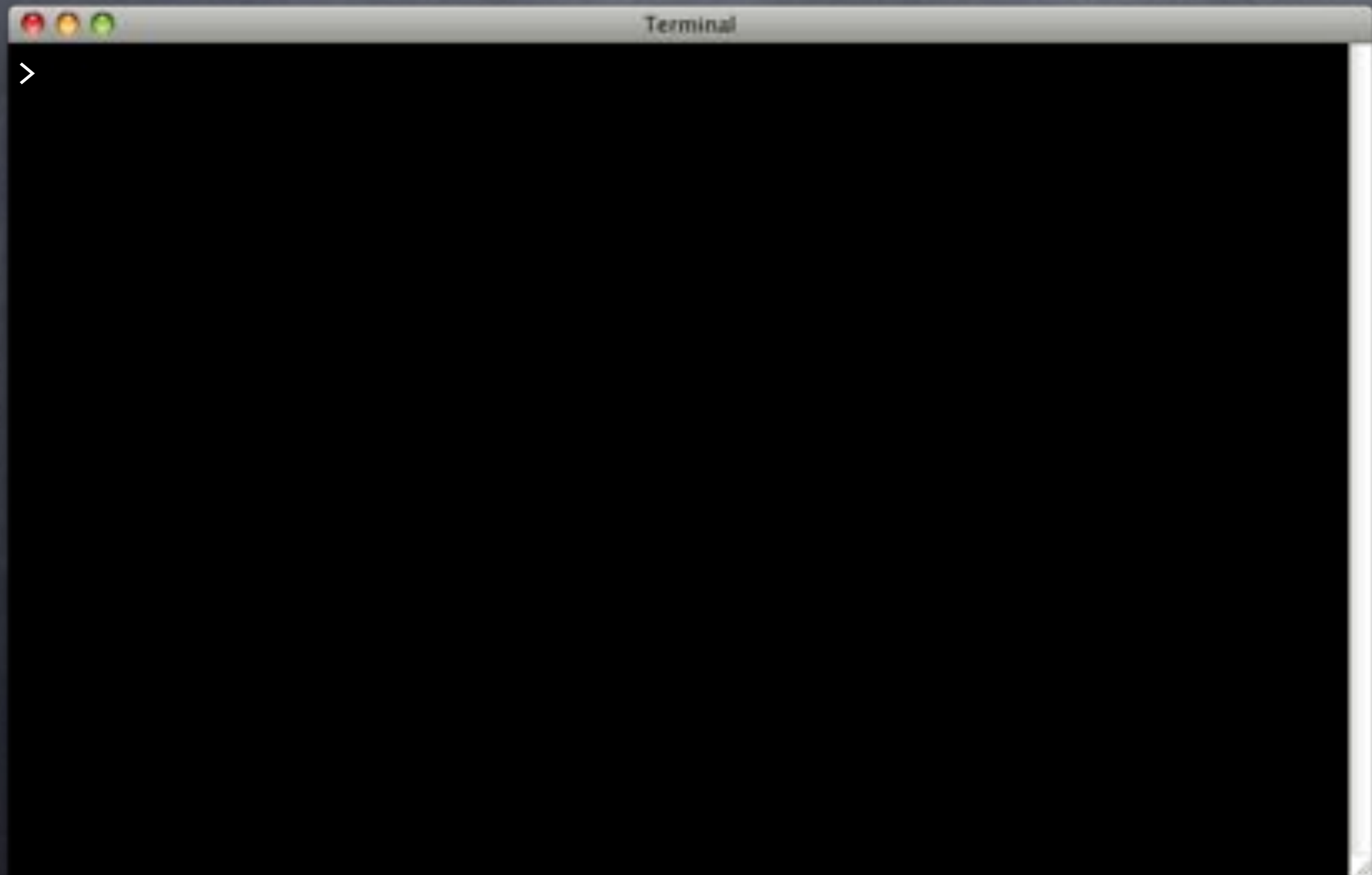


# Resets

```
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

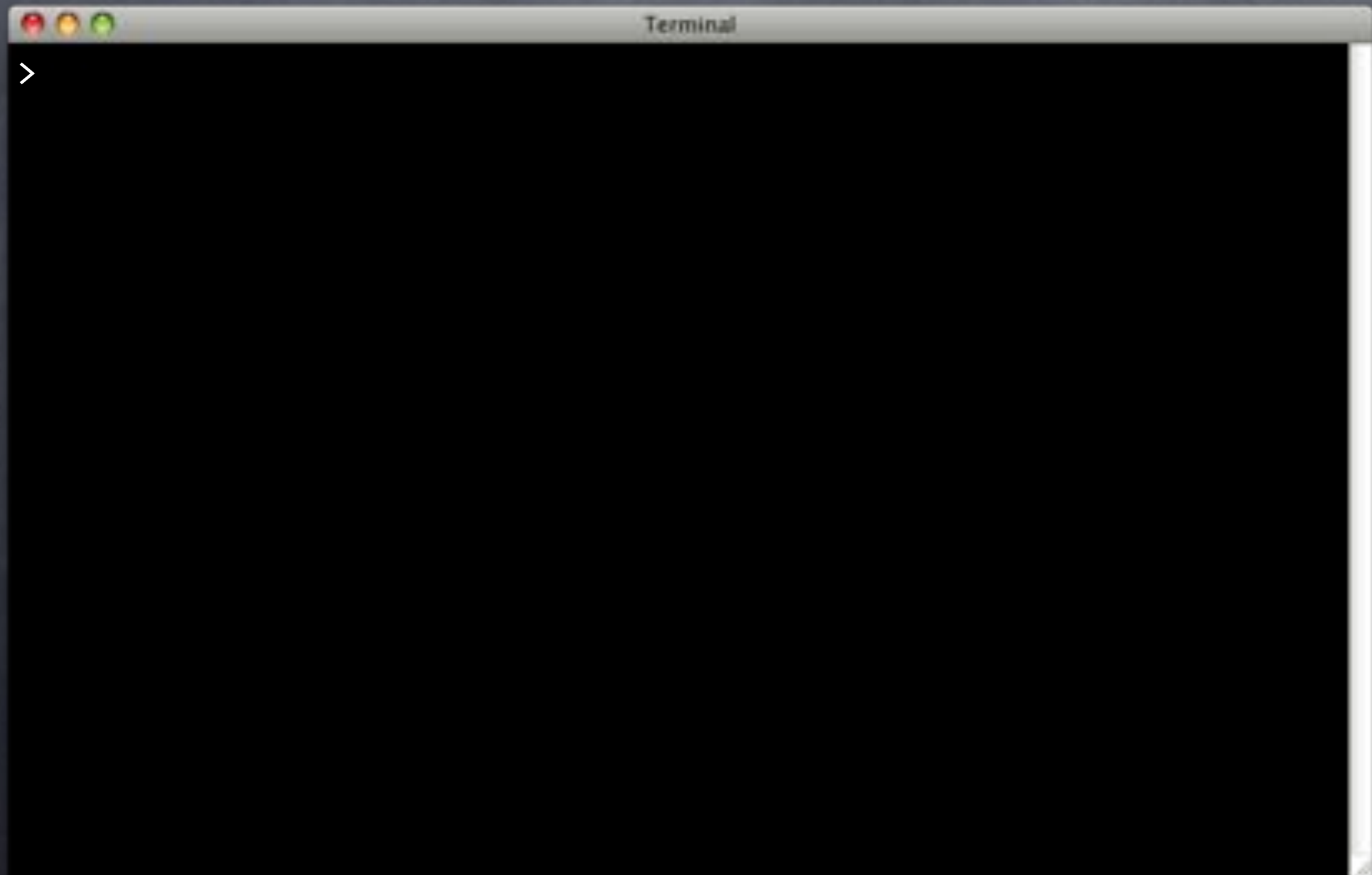
```
Terminal
> git merge users
Updating e46bdf9..610b318
Fast-forward
 deploy/users.sql | 13 ++++++++
 revert/users.sql |  7 +++++
 sqitch.plan      |  1 +
 test/users.sql   | 29 ++++++++
 verify/users.sql |  9 +++++
5 files changed, 59 insertions(+)
create mode 100644 deploy/users.sql
create mode 100644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

# 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

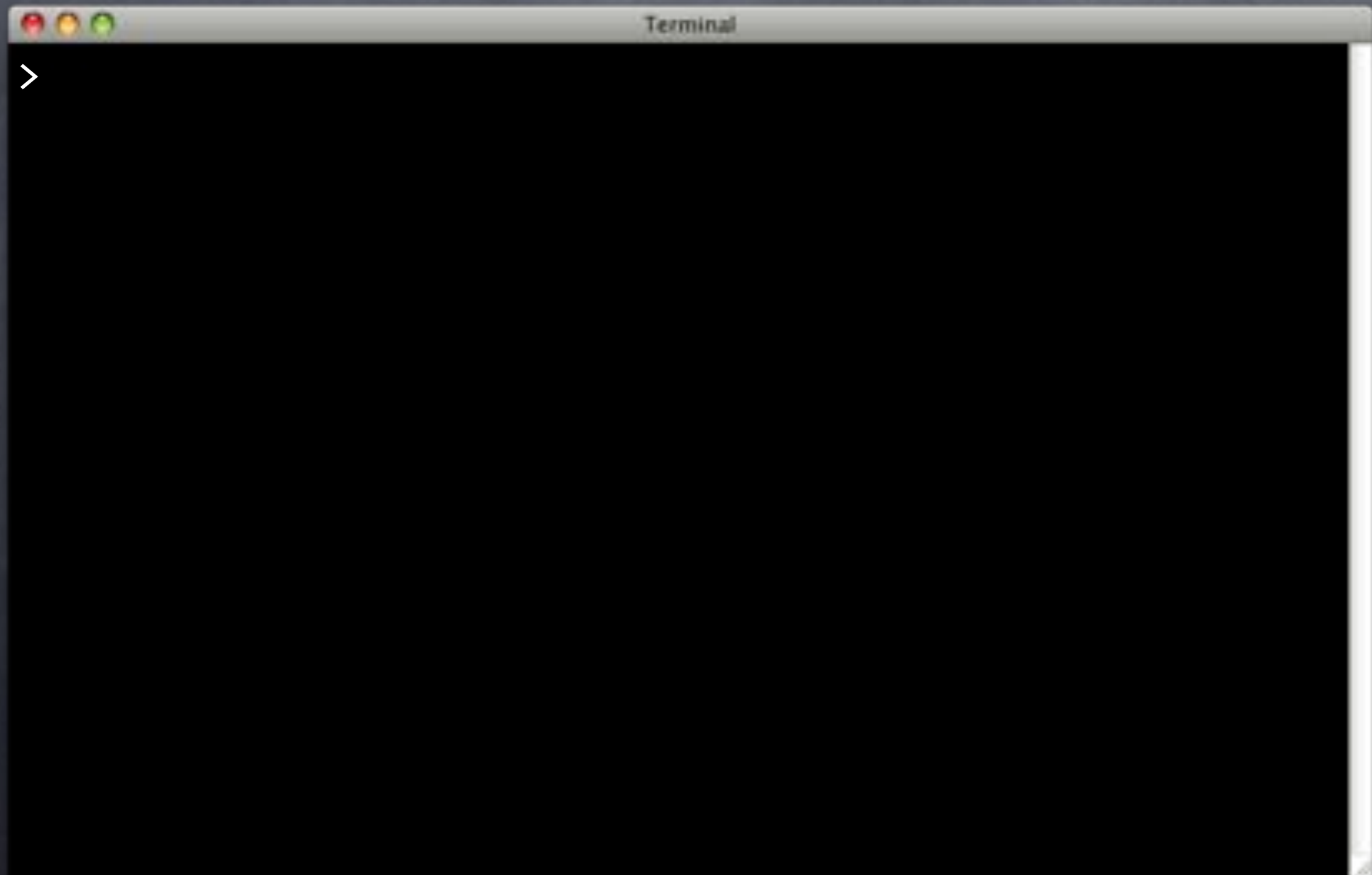
```
Terminal
> git merge flips
Updating 610b318..fbf8469
Fast-forward
 deploy/flips.sql | 15 ++++++
 revert/flips.sql |  7 +++++
 sqitch.plan      |  1 +
 test/flips.sql   | 38 ++++++
 verify/flips.sql | 12 +++++
5 files changed, 73 insertions(+)
create mode 100644 deploy/flips.sql
create mode 100644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

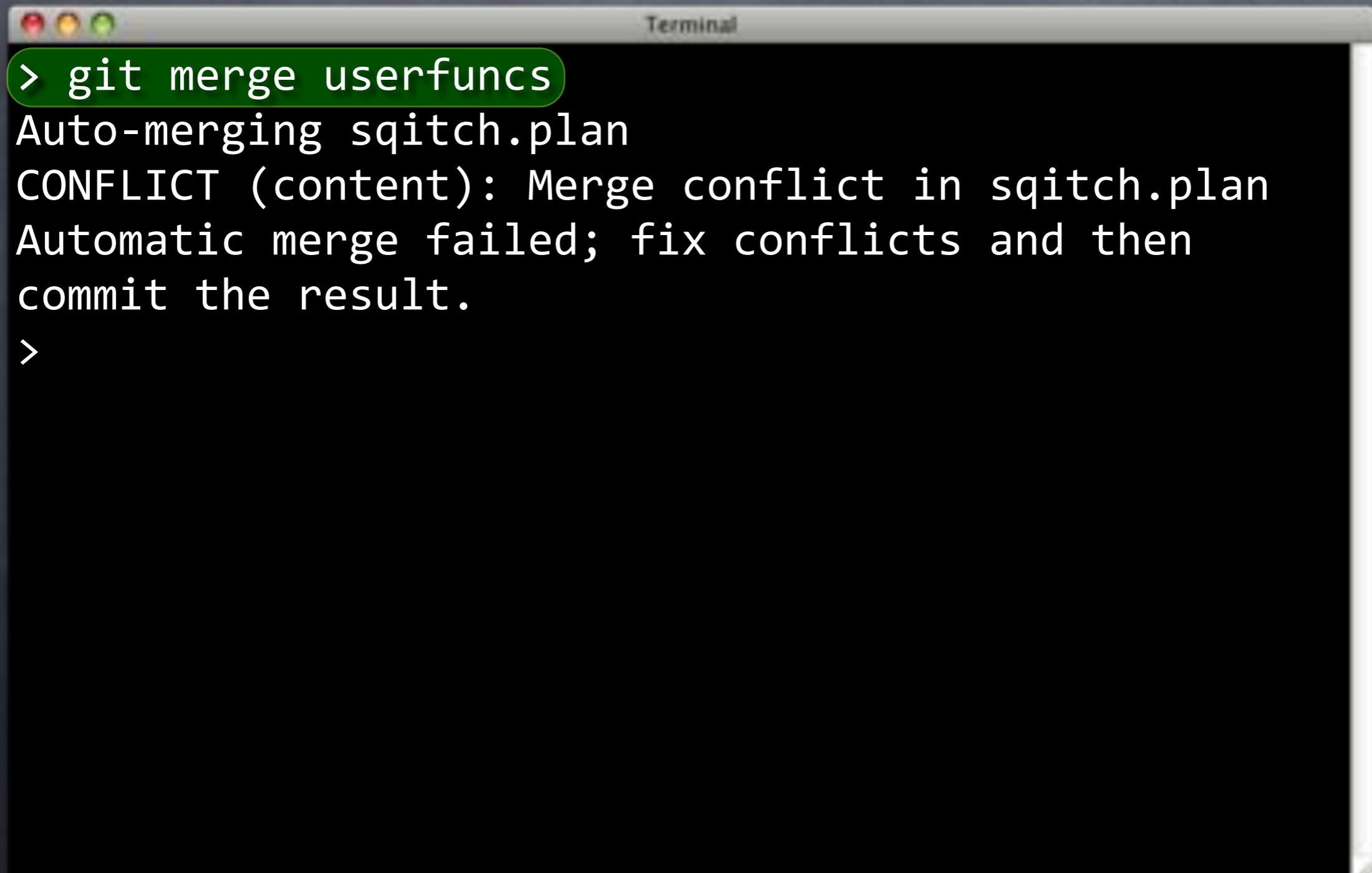
So good...

# Mergers and Acquisitions



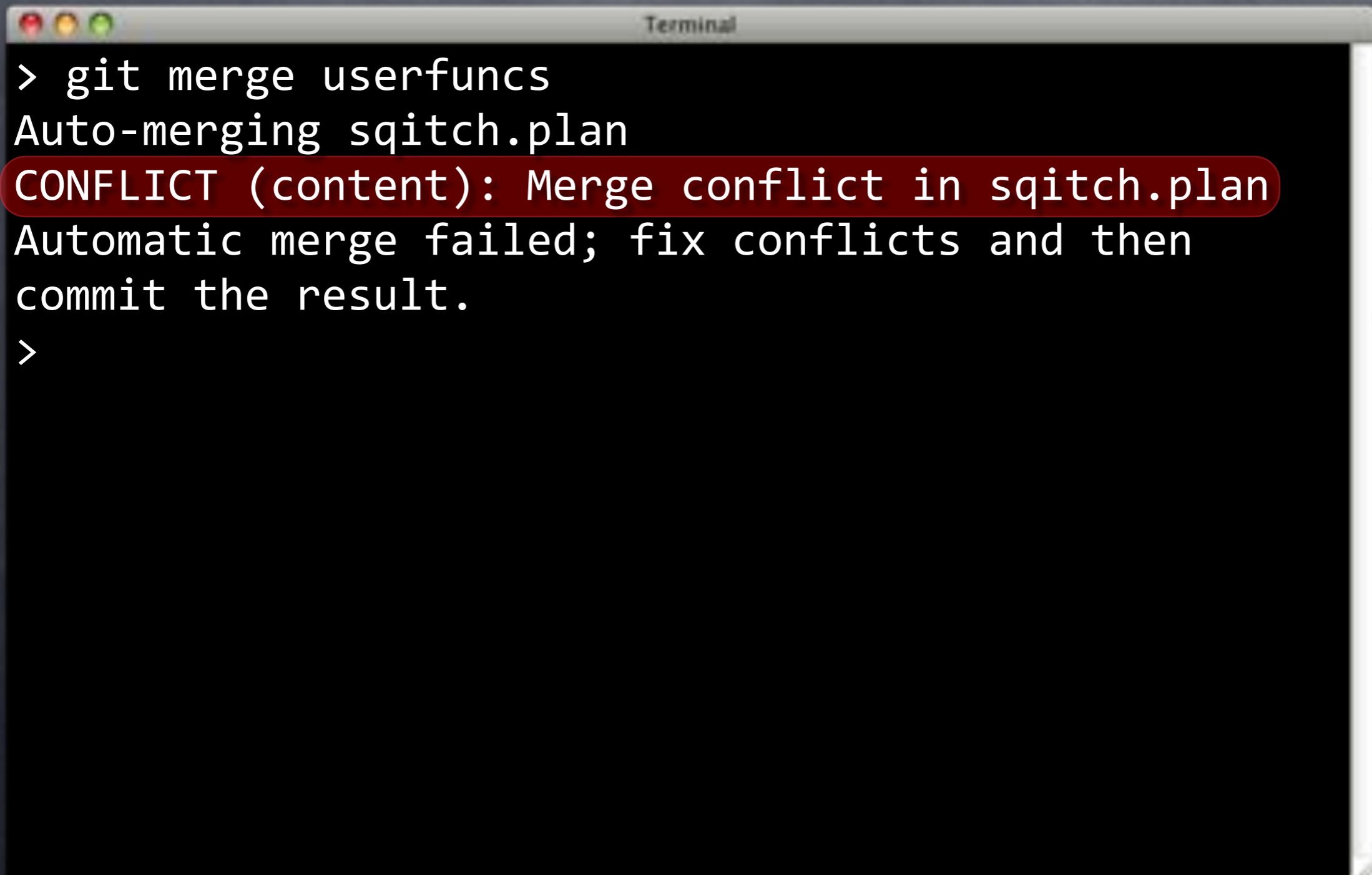


# Mergers and Acquisitions

A terminal window titled "Terminal" with a dark background and light text. The command `> git merge userfuncs` is highlighted in green. The output shows an auto-merge of `sqitch.plan` failing due to a conflict.

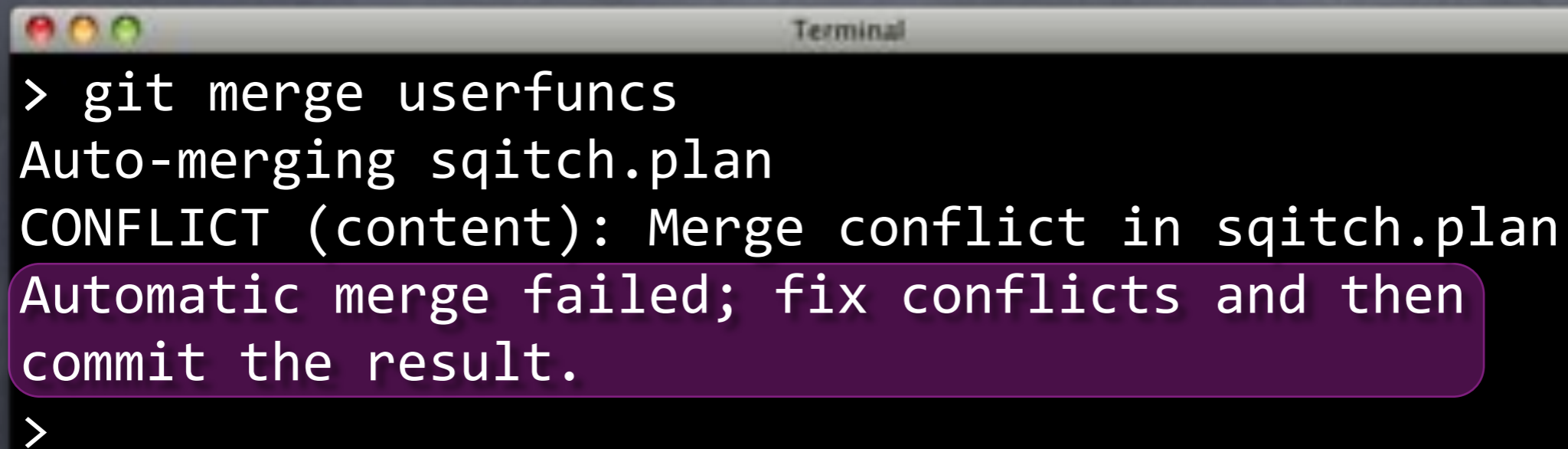
```
> git merge userfuncs
Auto-merging sqitch.plan
CONFLICT (content): Merge conflict in sqitch.plan
Automatic merge failed; fix conflicts and then
commit the result.
>
```

# Mergers and Acquisitions



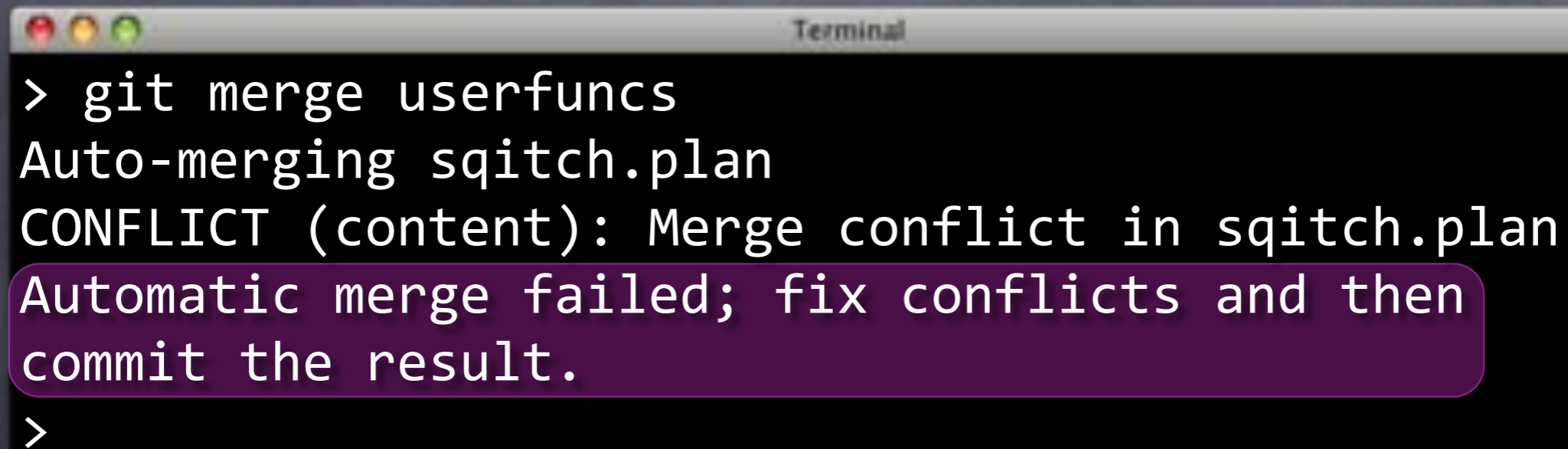
```
Terminal
> git merge userfuncs
Auto-merging sqitch.plan
CONFLICT (content): Merge conflict in sqitch.plan
Automatic merge failed; fix conflicts and then
commit the result.
>
```

# 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.
>
```

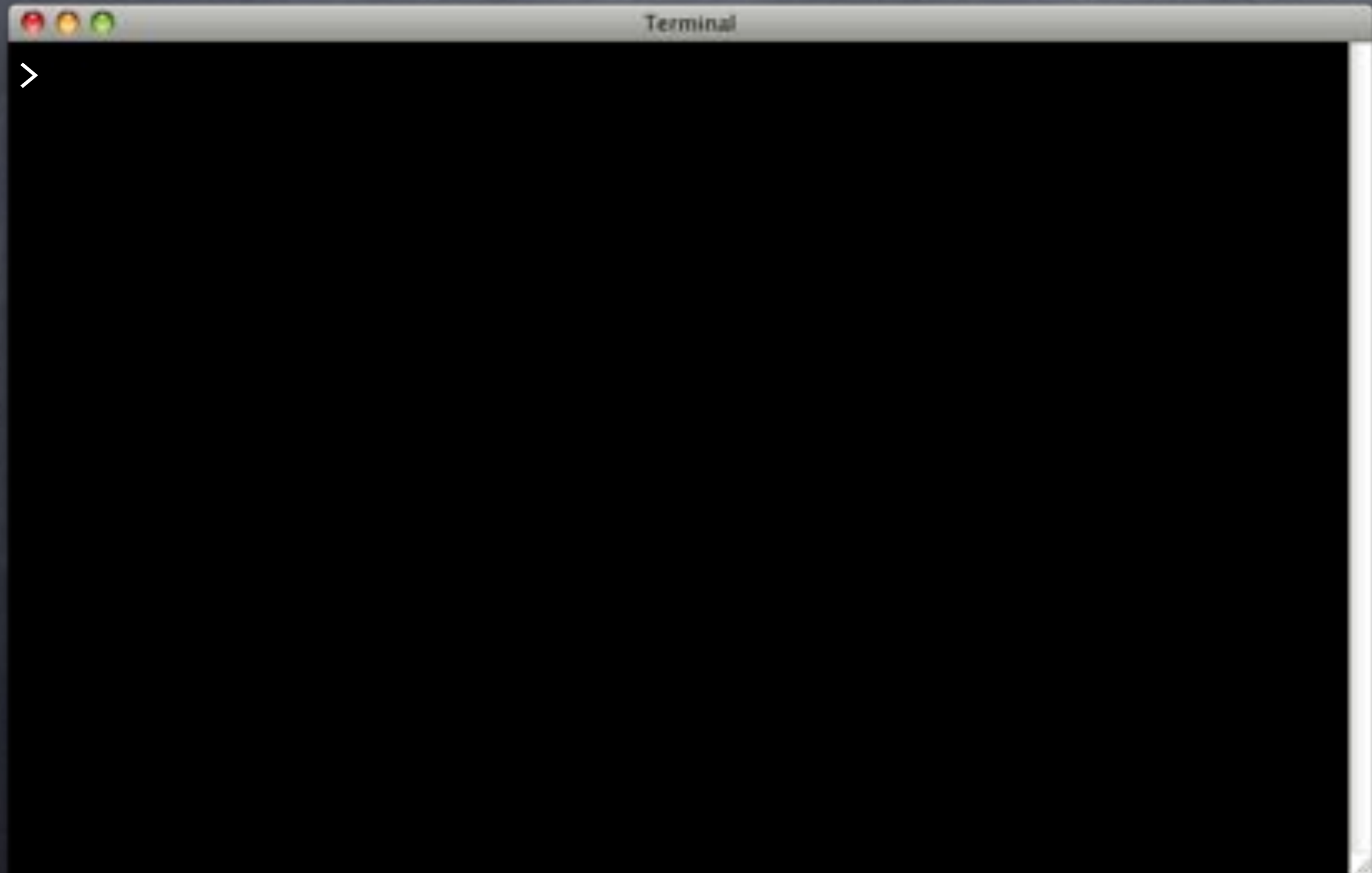
# Mergers and Acquisitions



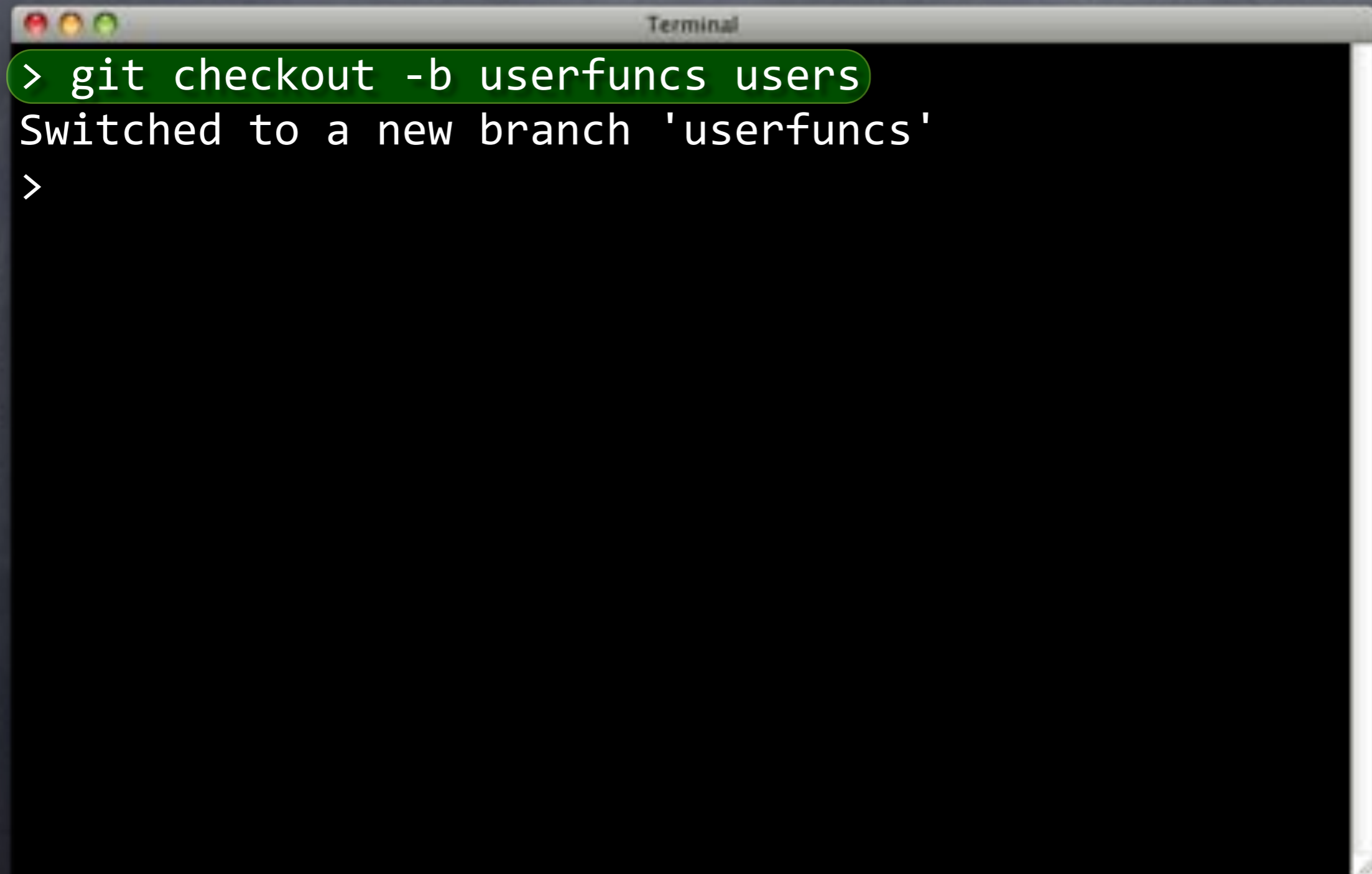
```
Terminal
> git merge userfuncs
Auto-merging sqitch.plan
CONFLICT (content): Merge conflict in sqitch.plan
Automatic merge failed; fix conflicts and then
commit the result.
>
```

Wha???

# Back in Time...



# Back in Time...

A terminal window titled "Terminal" with a dark background and a light gray title bar. The window contains the following text: a green prompt character followed by the command "git checkout -b userfuncs users" on a green background; the output "Switched to a new branch 'userfuncs'"; and a second green prompt character on a new line.

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

# Back in Time...

Ah-ha!

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

# 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



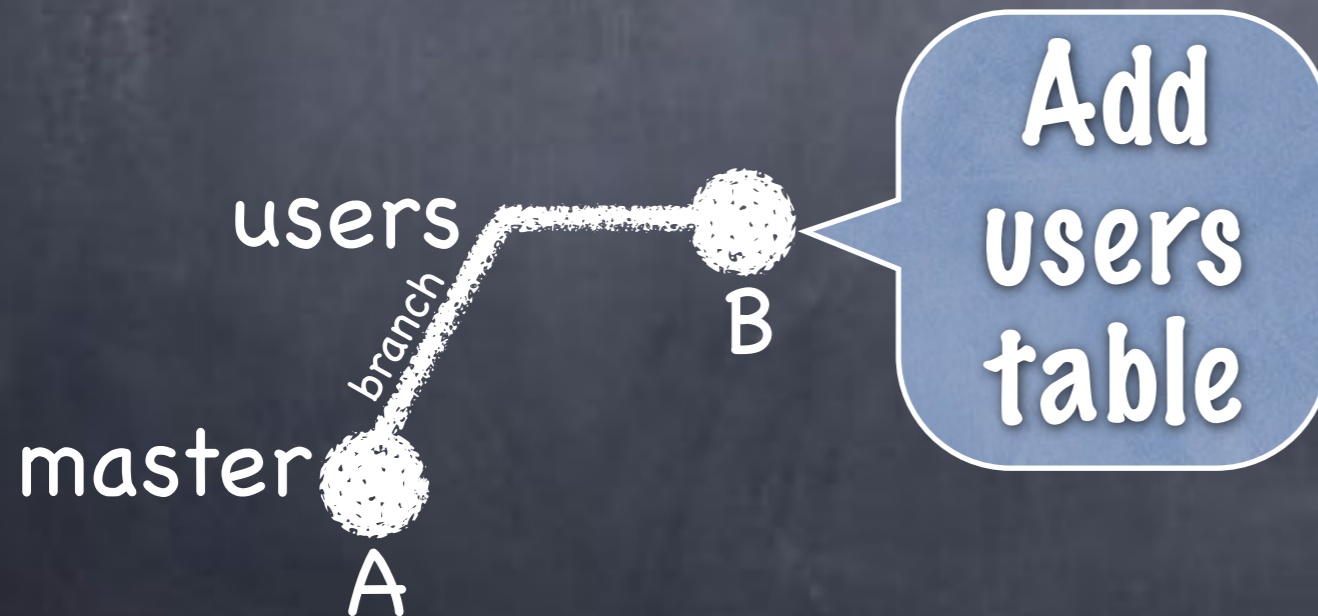
# Backbrancher

master   
A

# Backbrancher

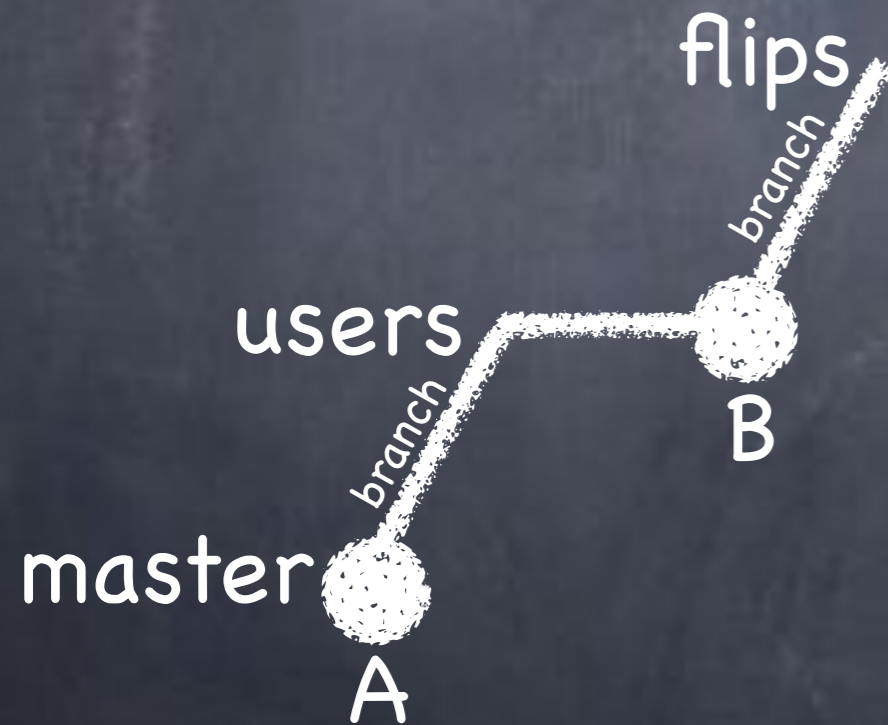


# Backbrancher

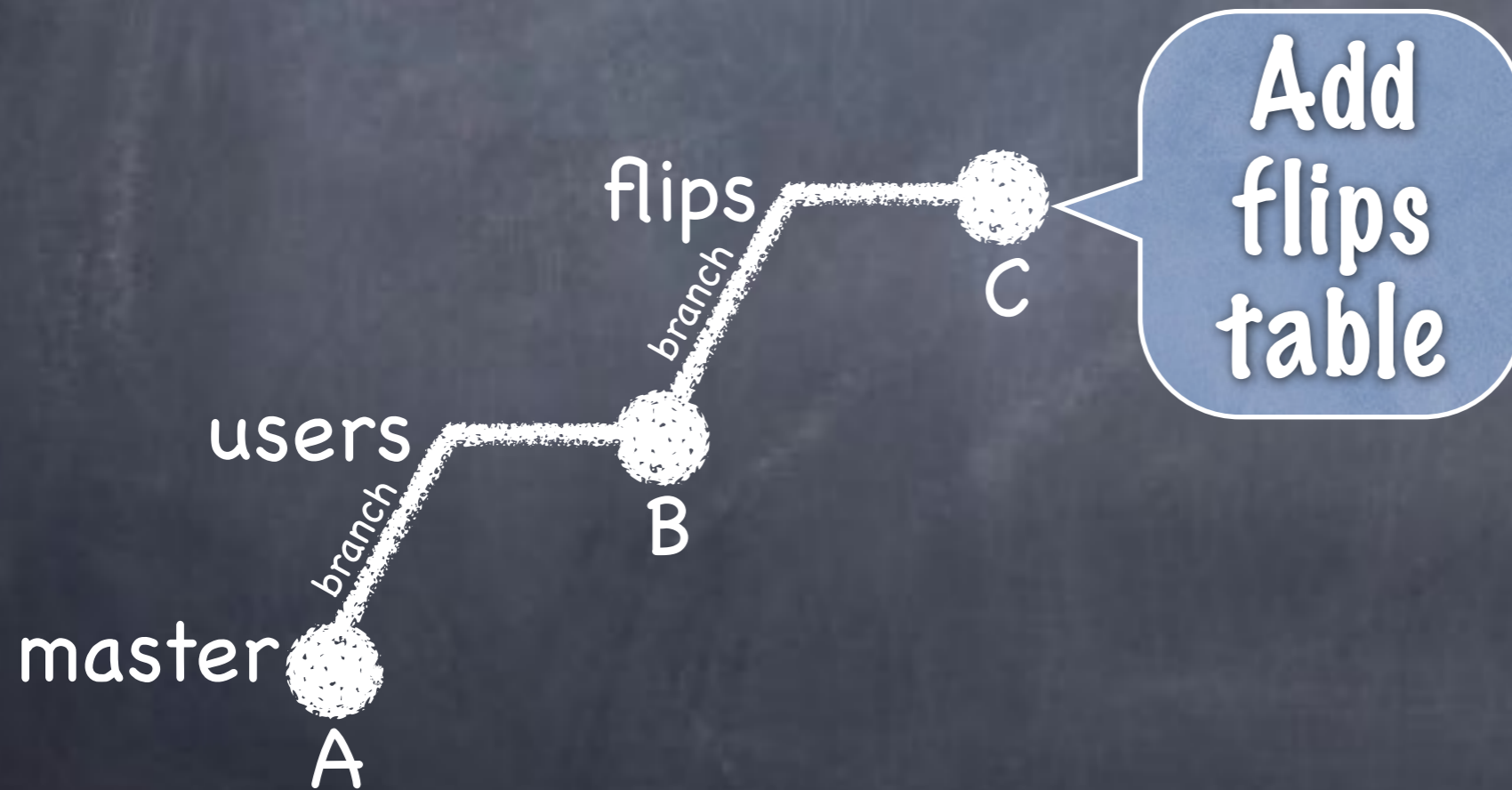




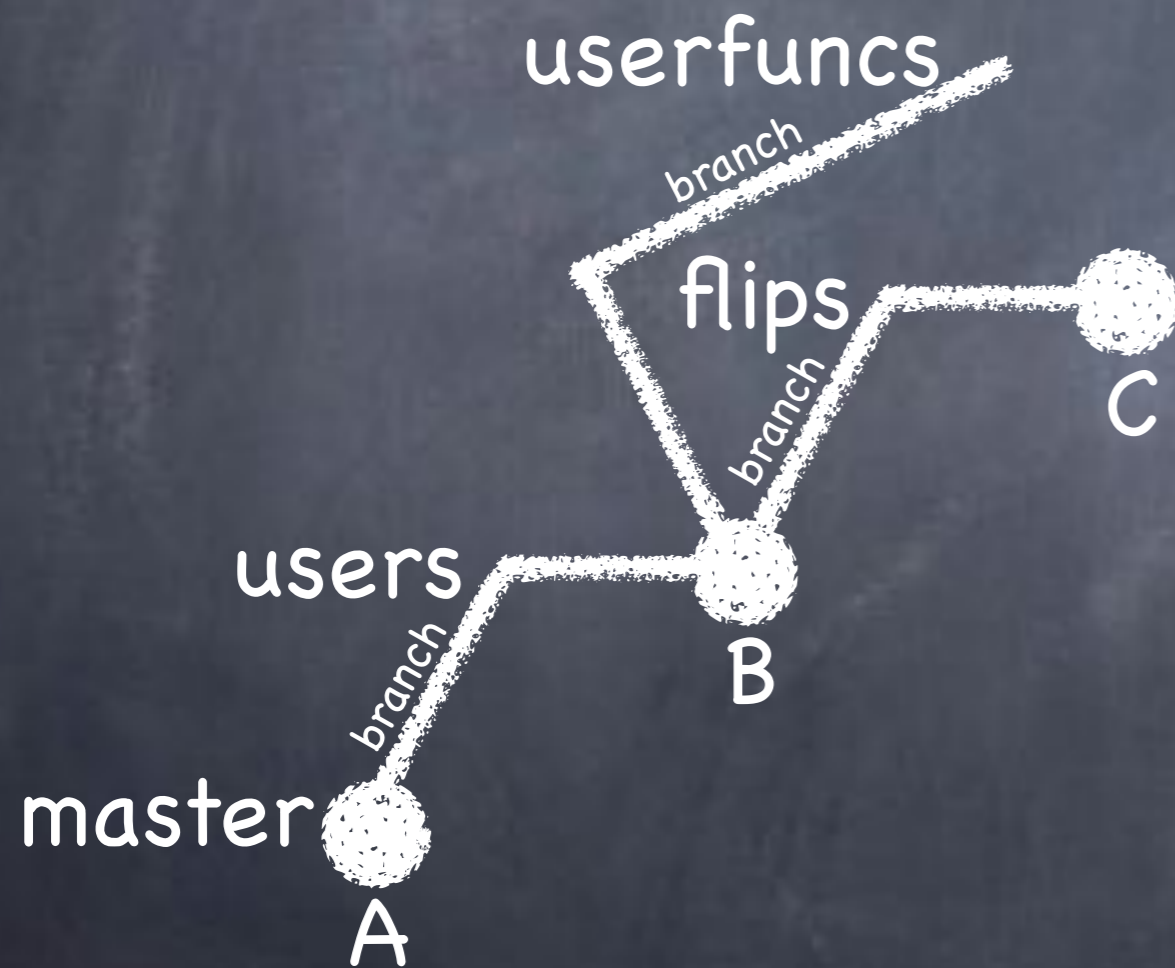
# Backbrancher



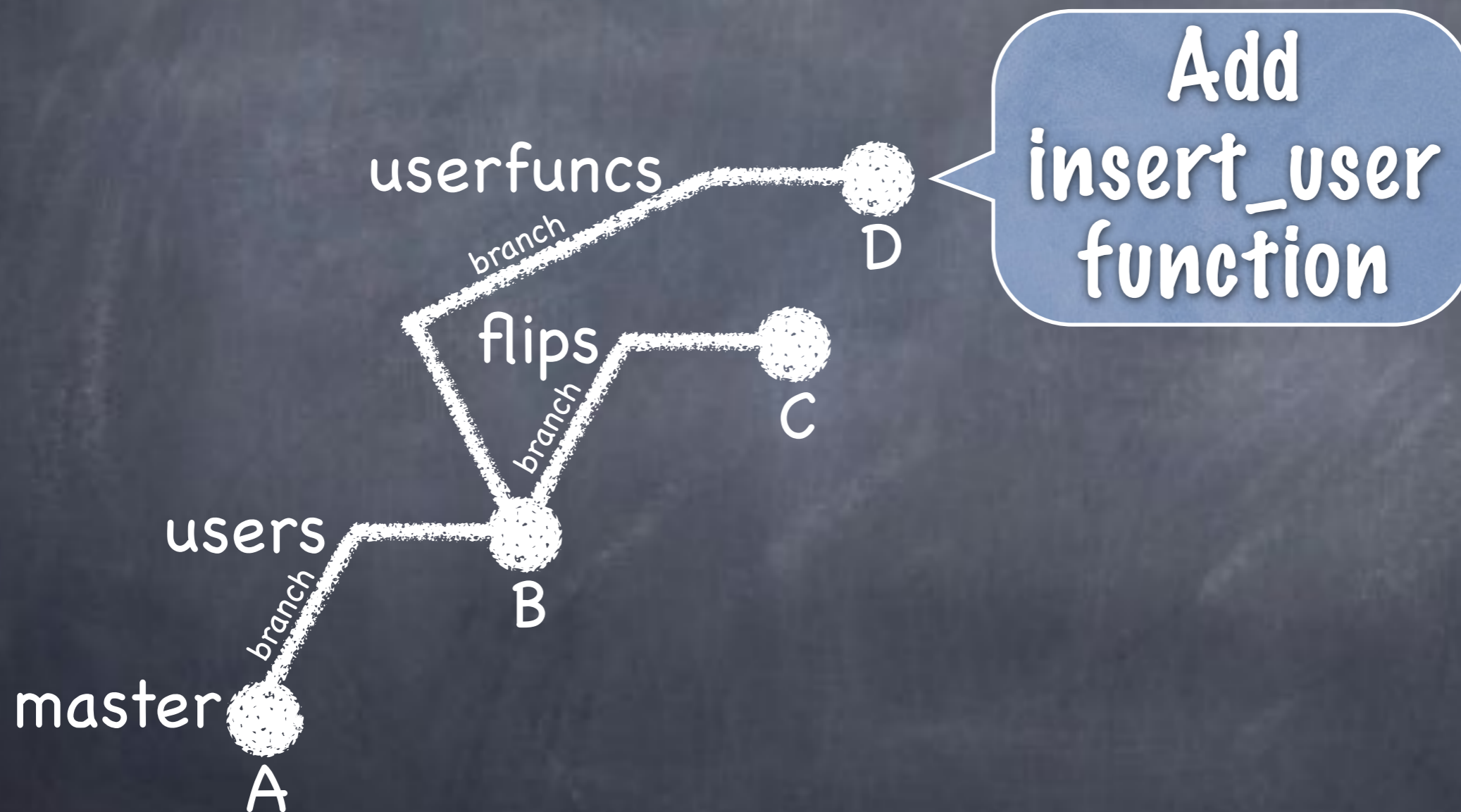
# Backbrancher



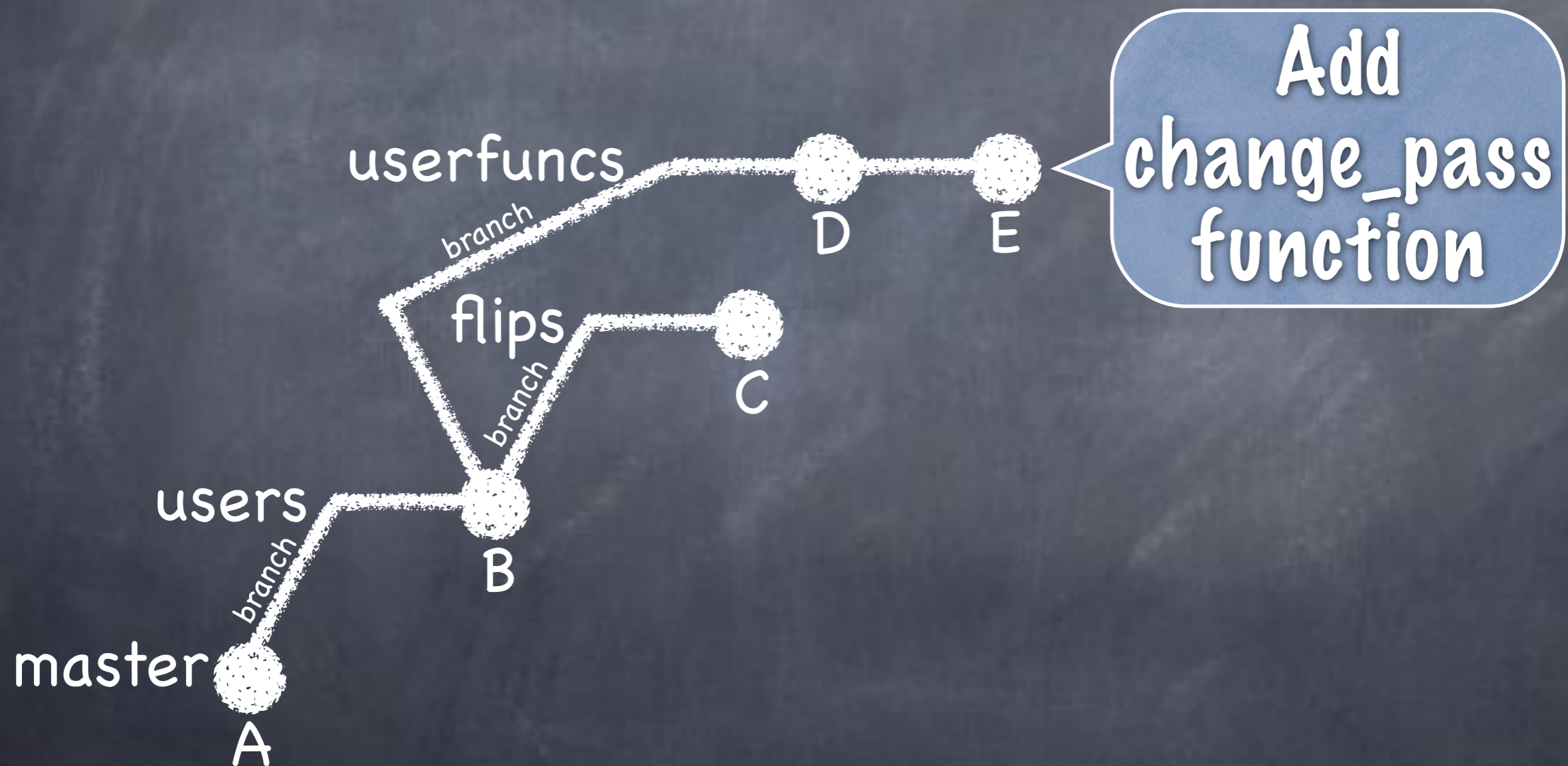
# Backbrancher



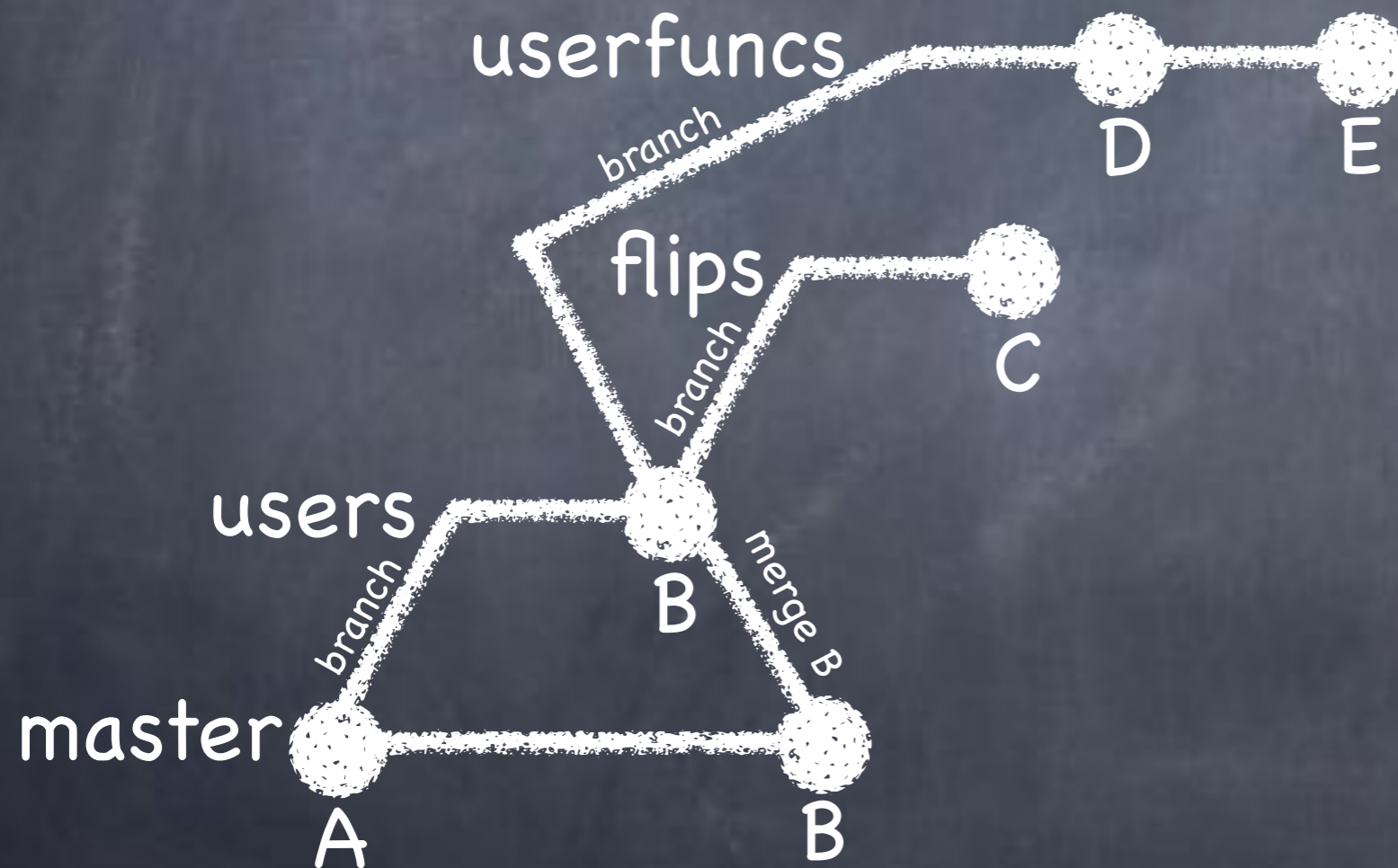
# Backbrancher



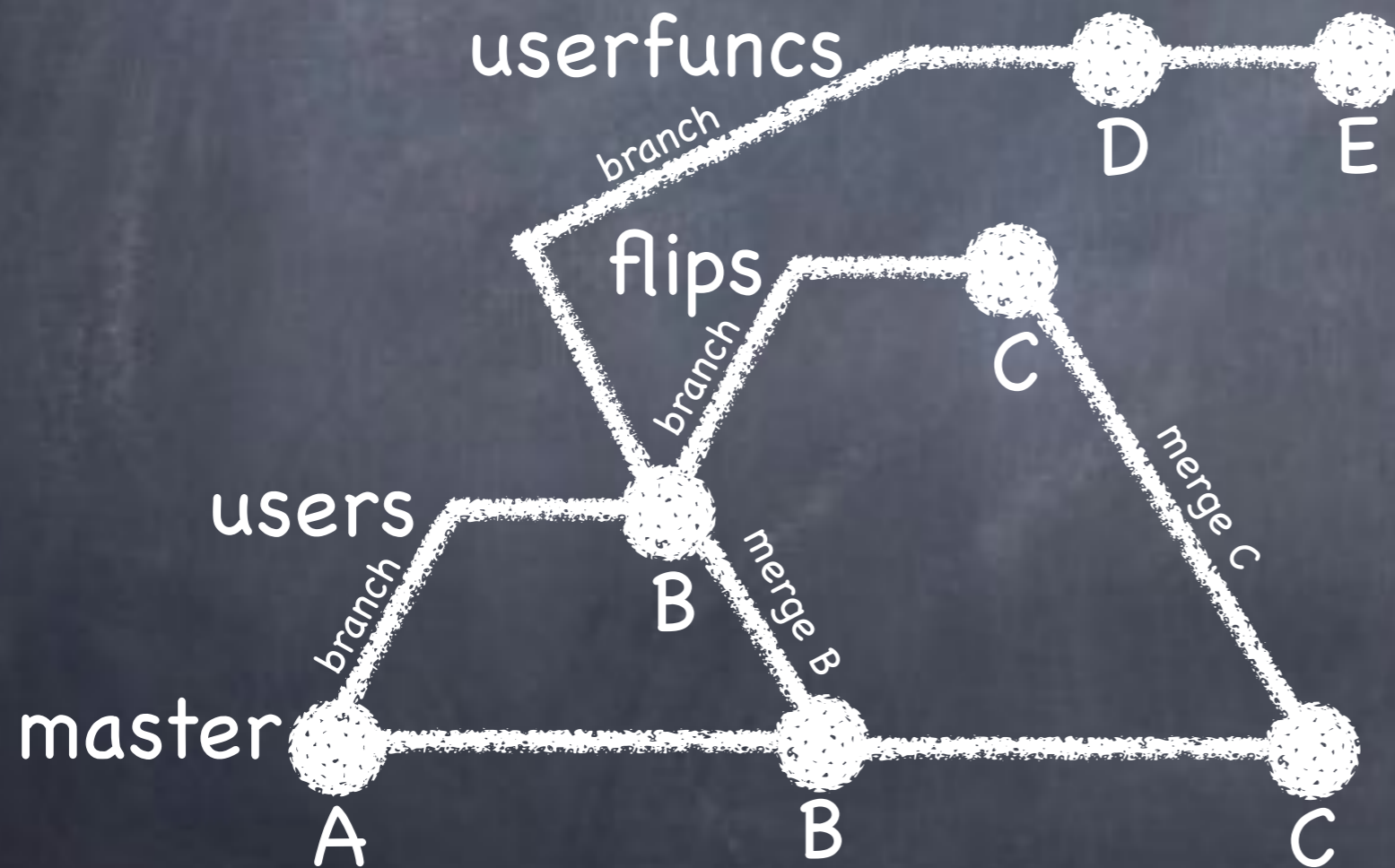
# Backbrancher



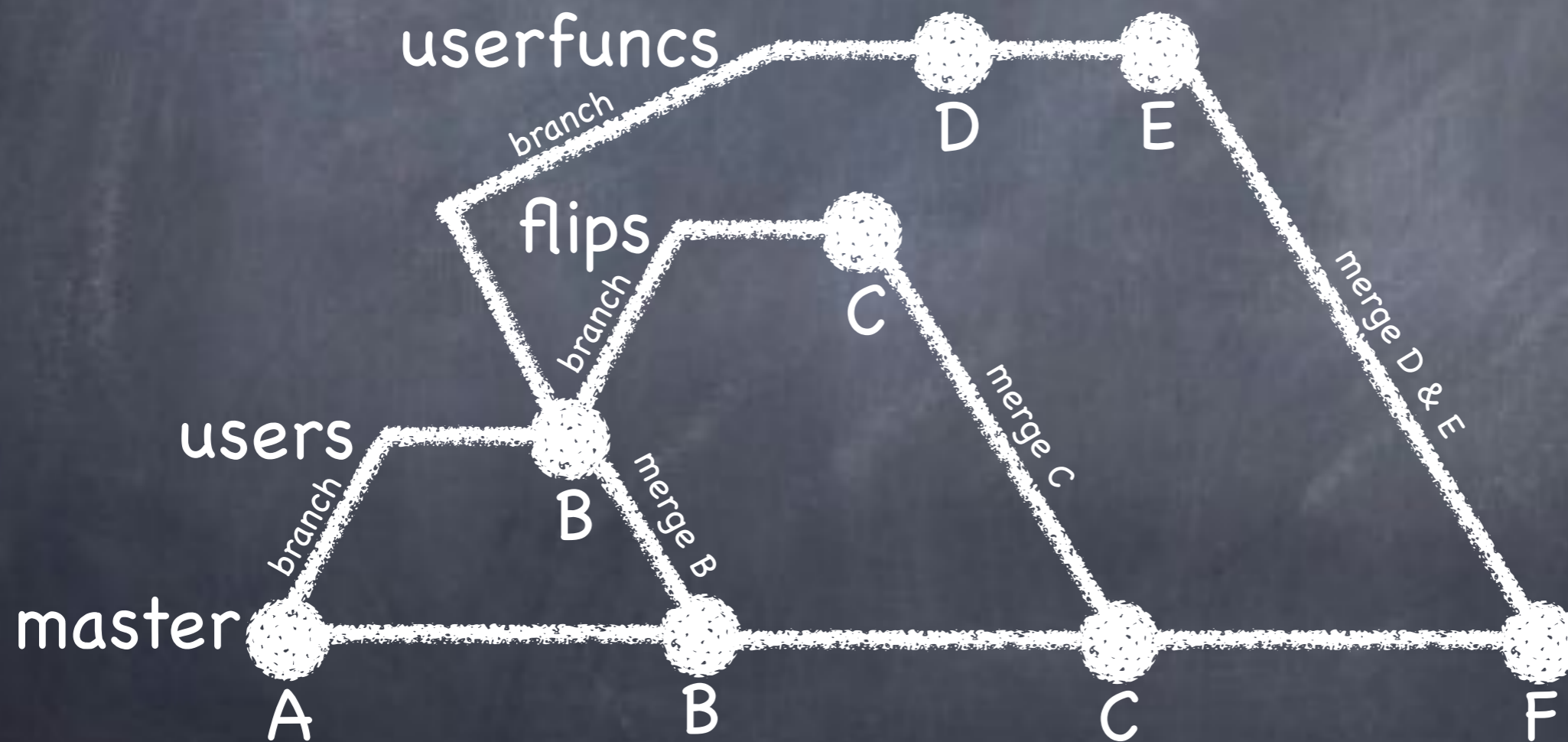
# Backbrancher



# Backbrancher

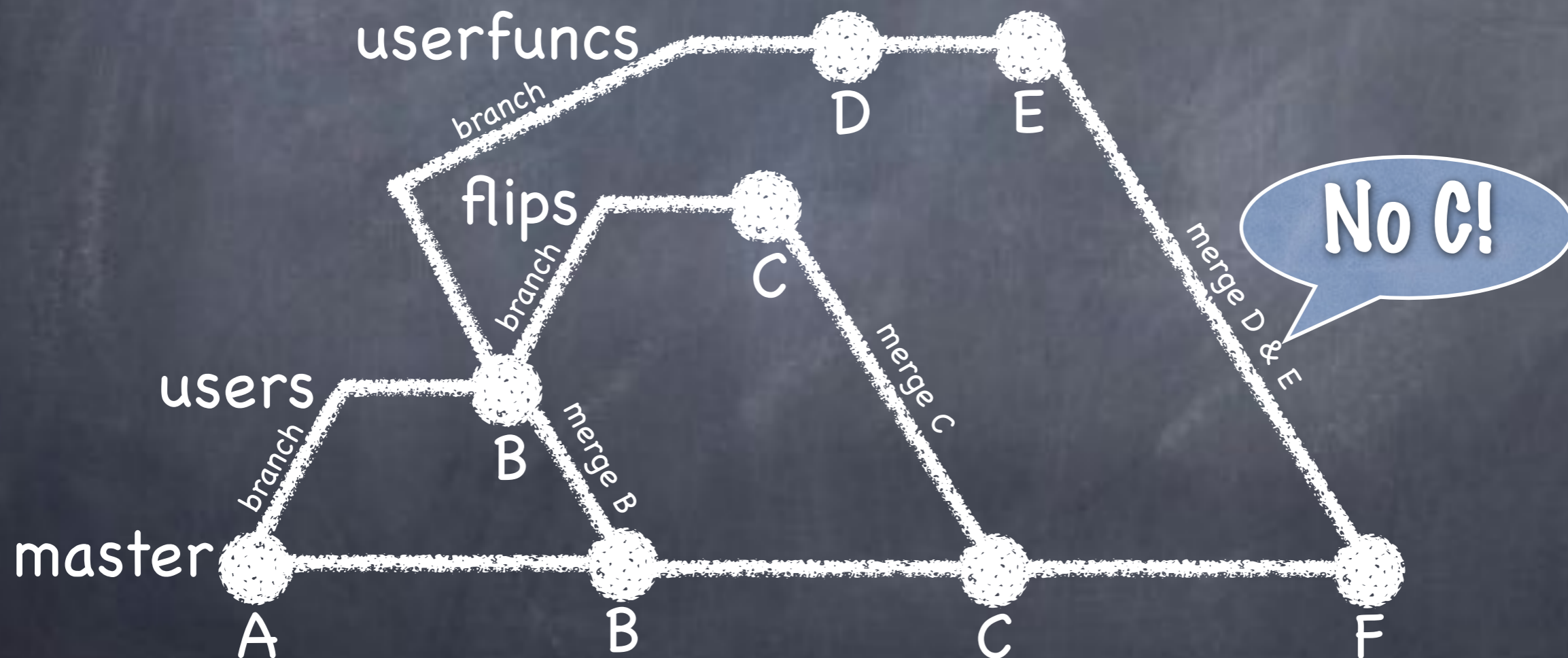


# Backbrancher

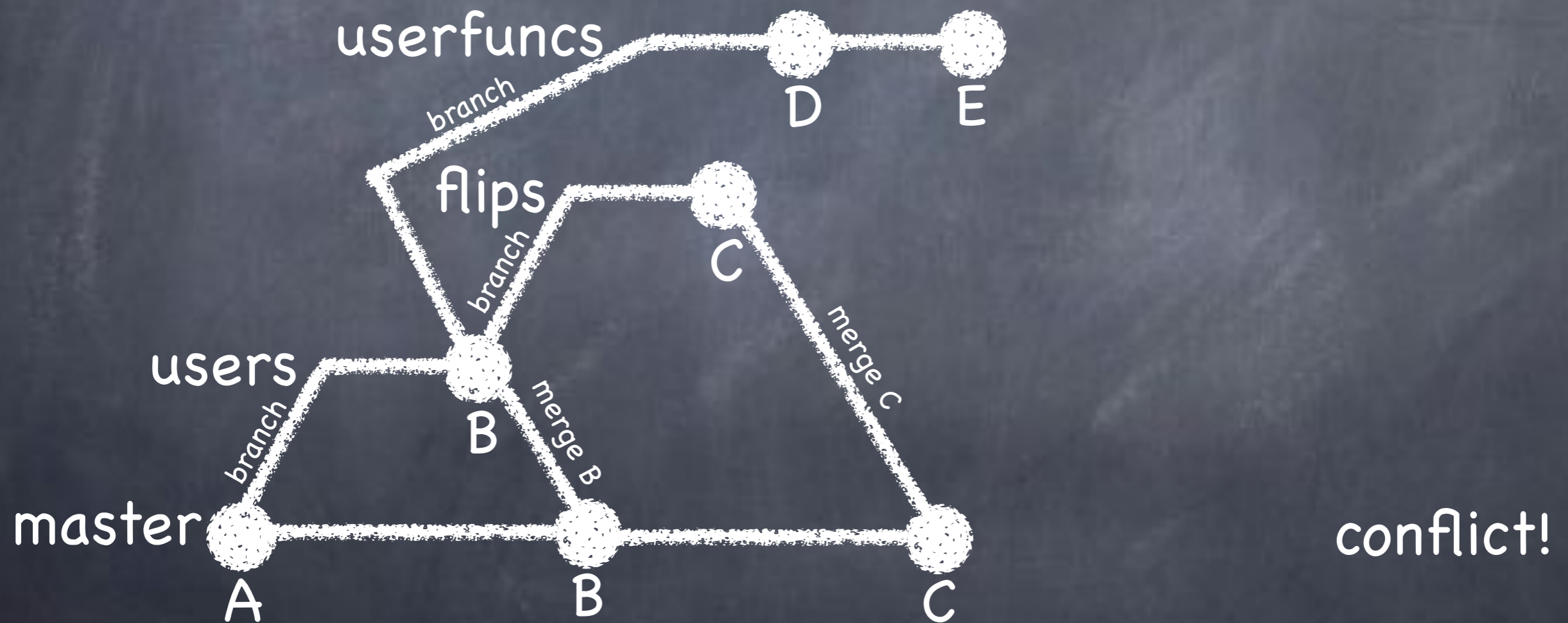




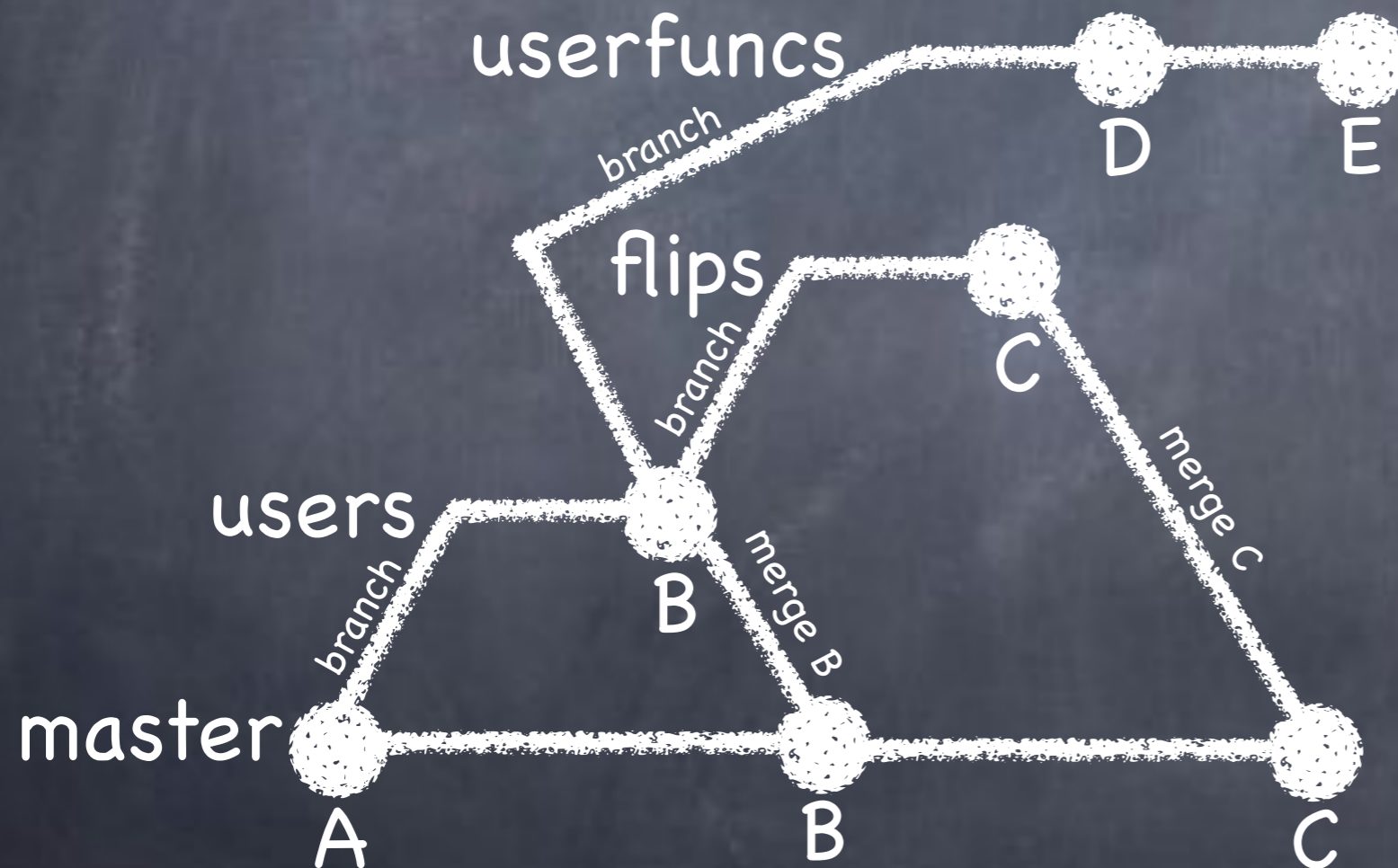
# Backbrancher



# Backbrancher



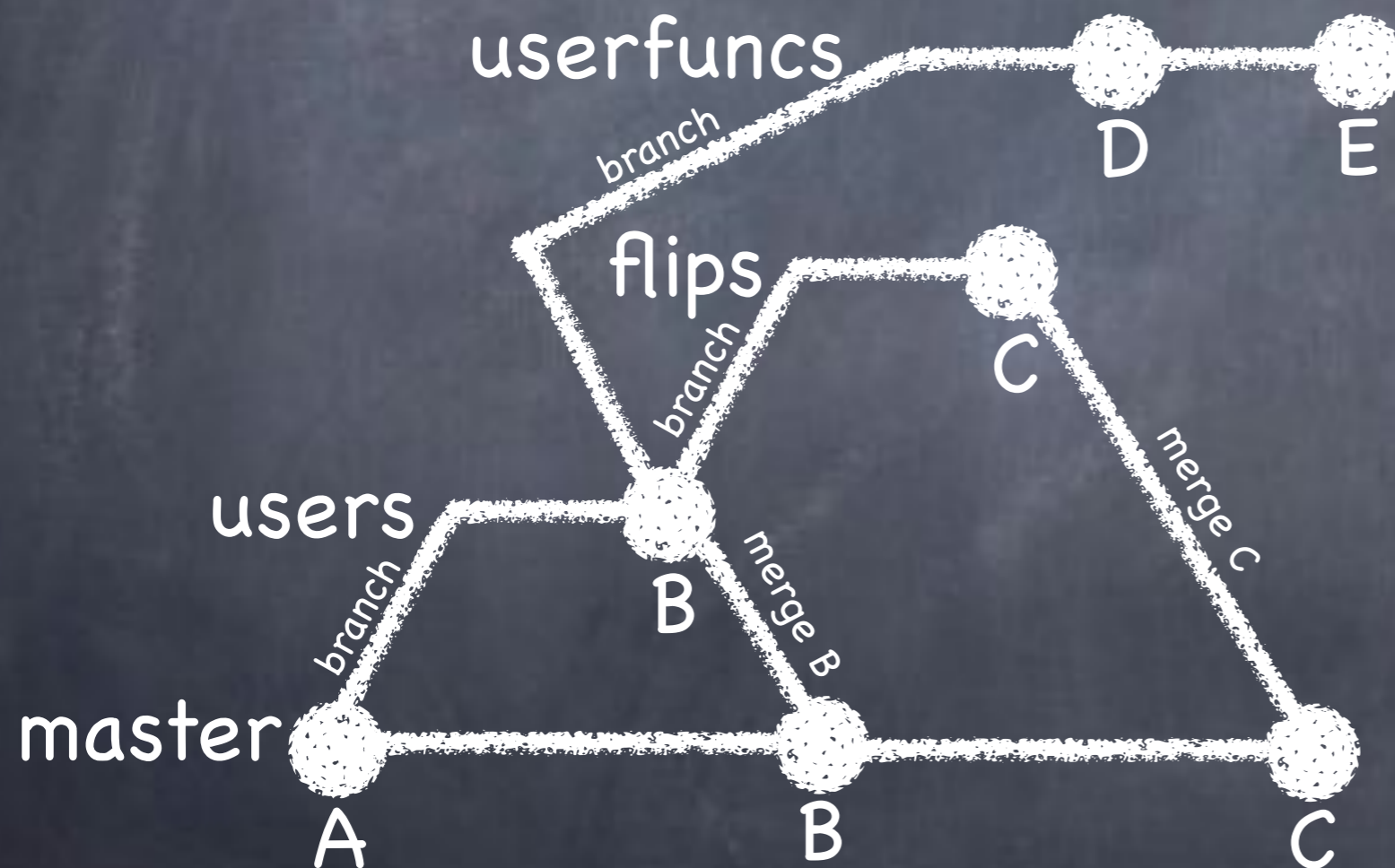
# Backbrancher



Now  
what?

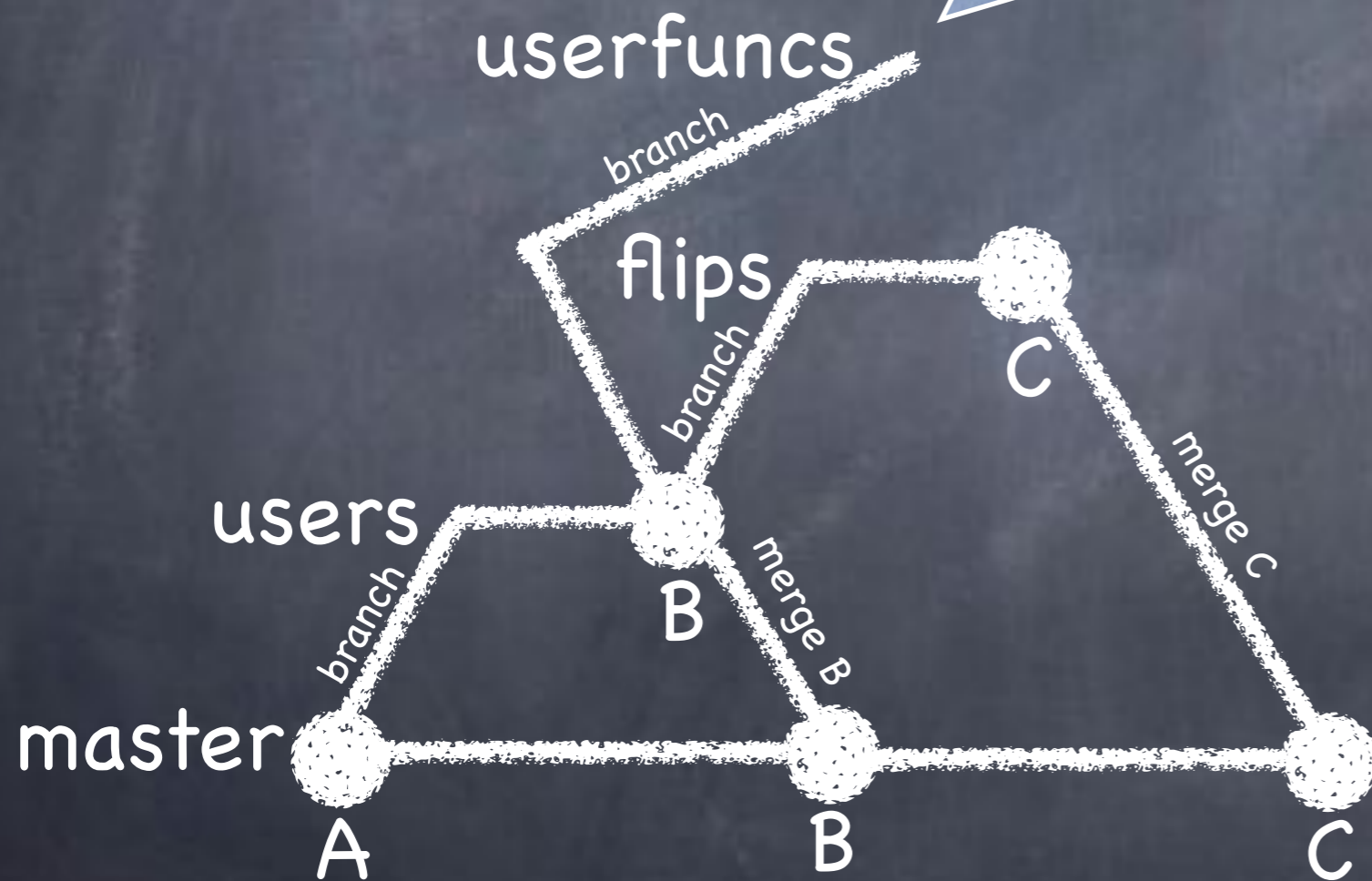
conflict!

# Rebase master

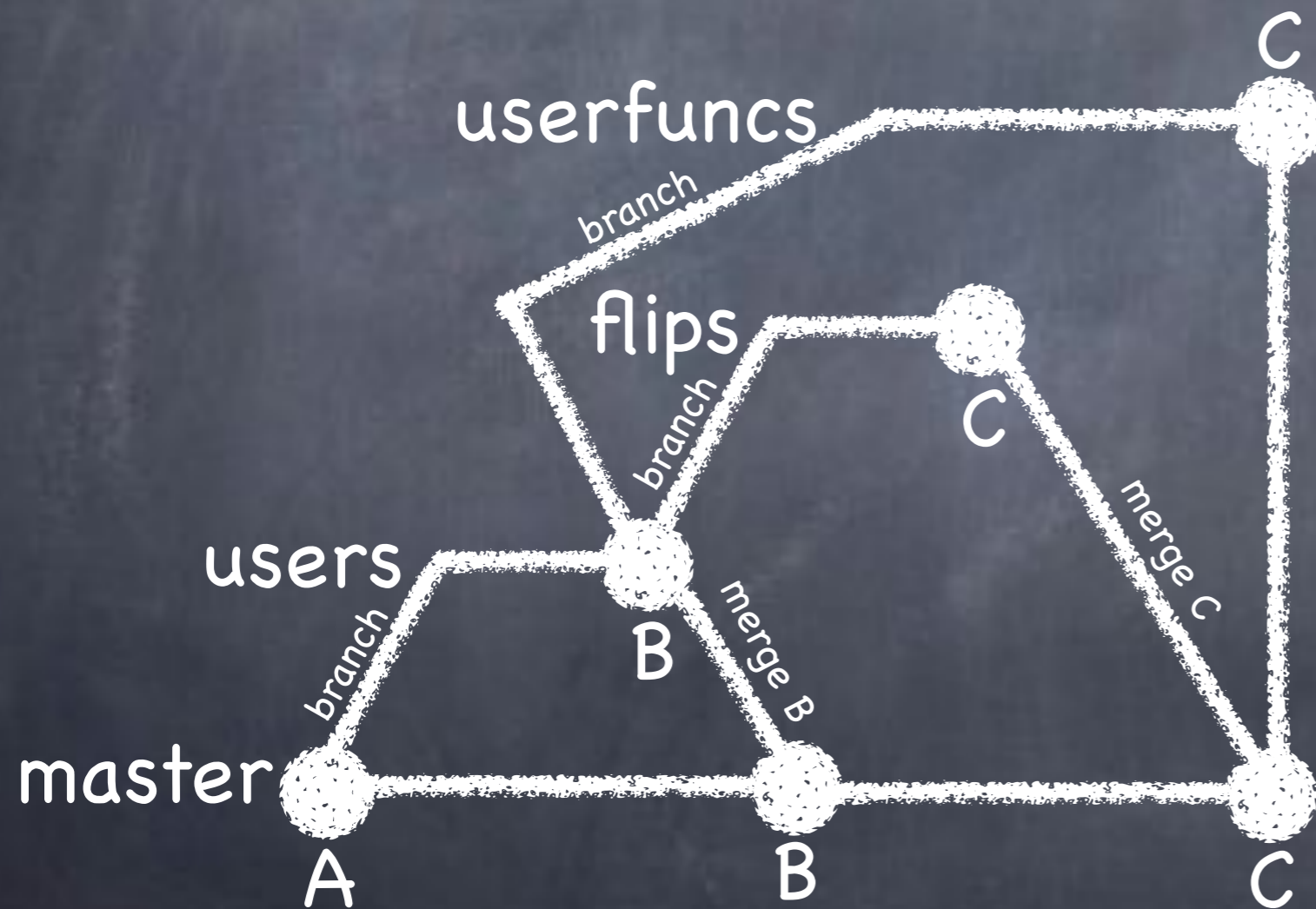


# Rebase master

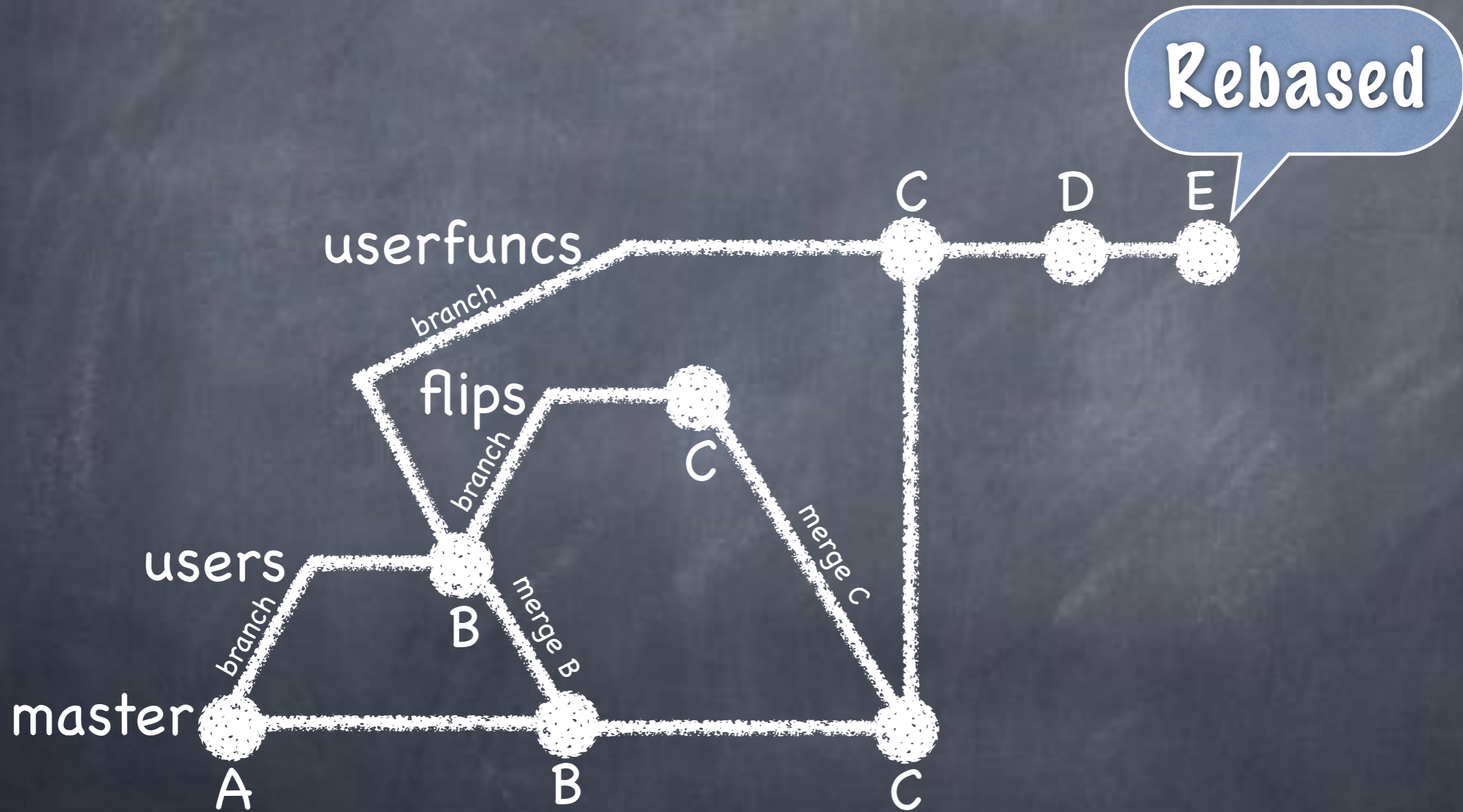
Last common  
with master: B



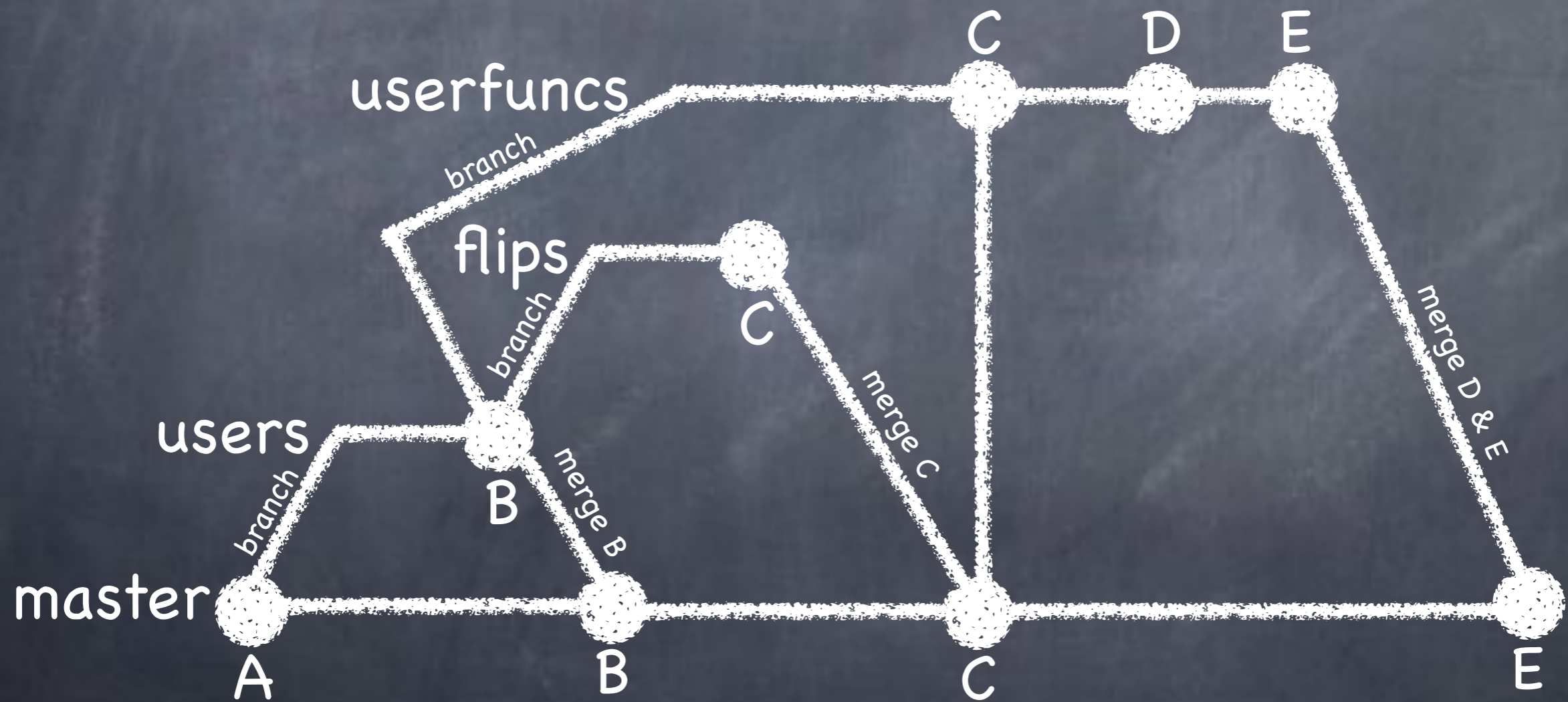
# Rebase master



# Rebase master

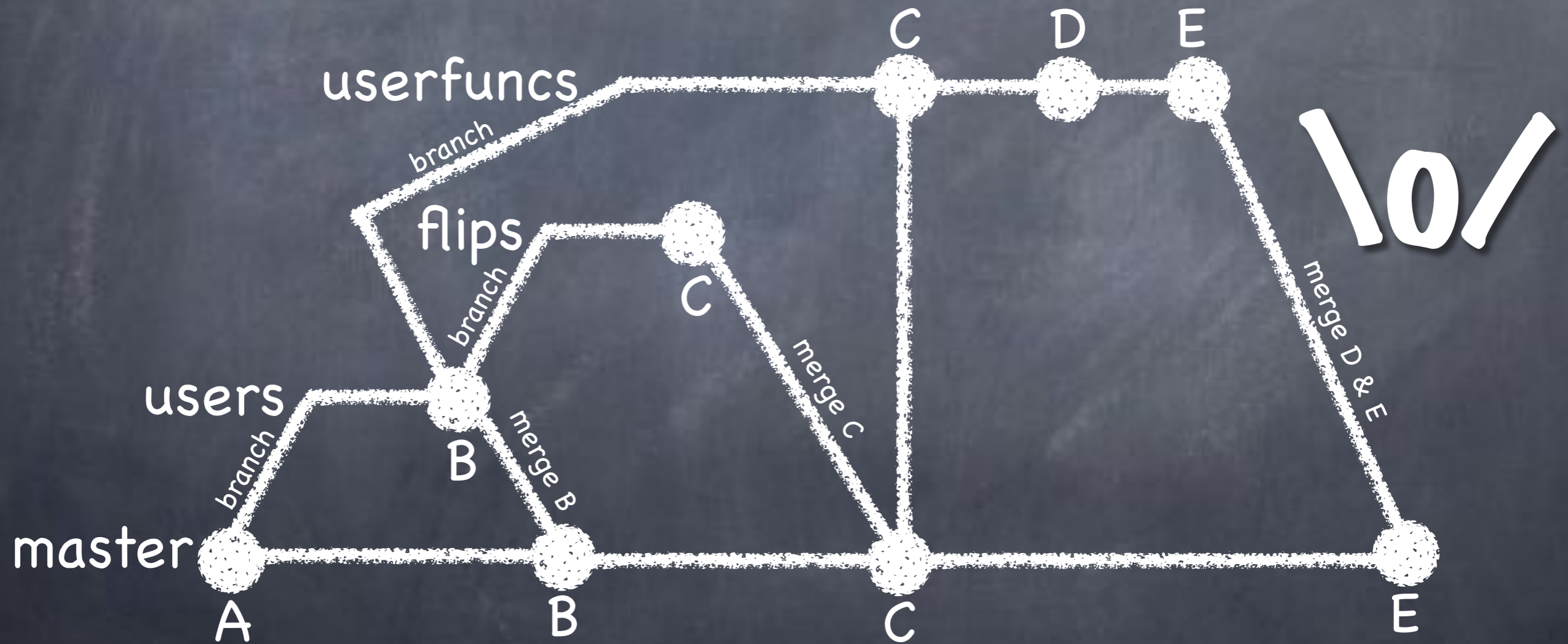


# Rebase master

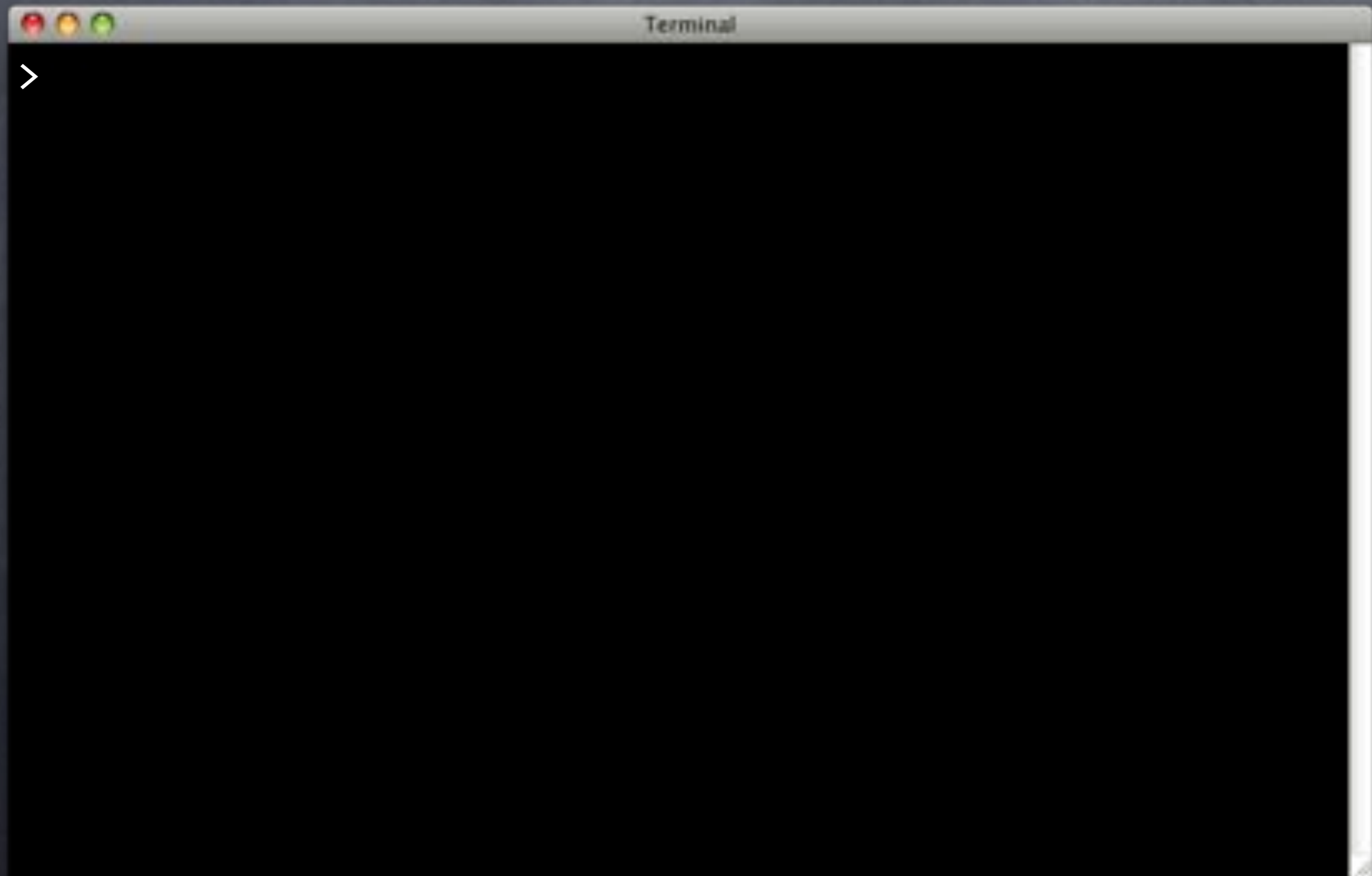





# Rebase master



# Reset



# Reset

A terminal window titled "Terminal" with a dark background. The command `> git reset --hard HEAD` is entered and highlighted in green. The output is `HEAD is now at fbf8469 Add flips table.` followed by a new prompt `>`.

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

Use with  
care!

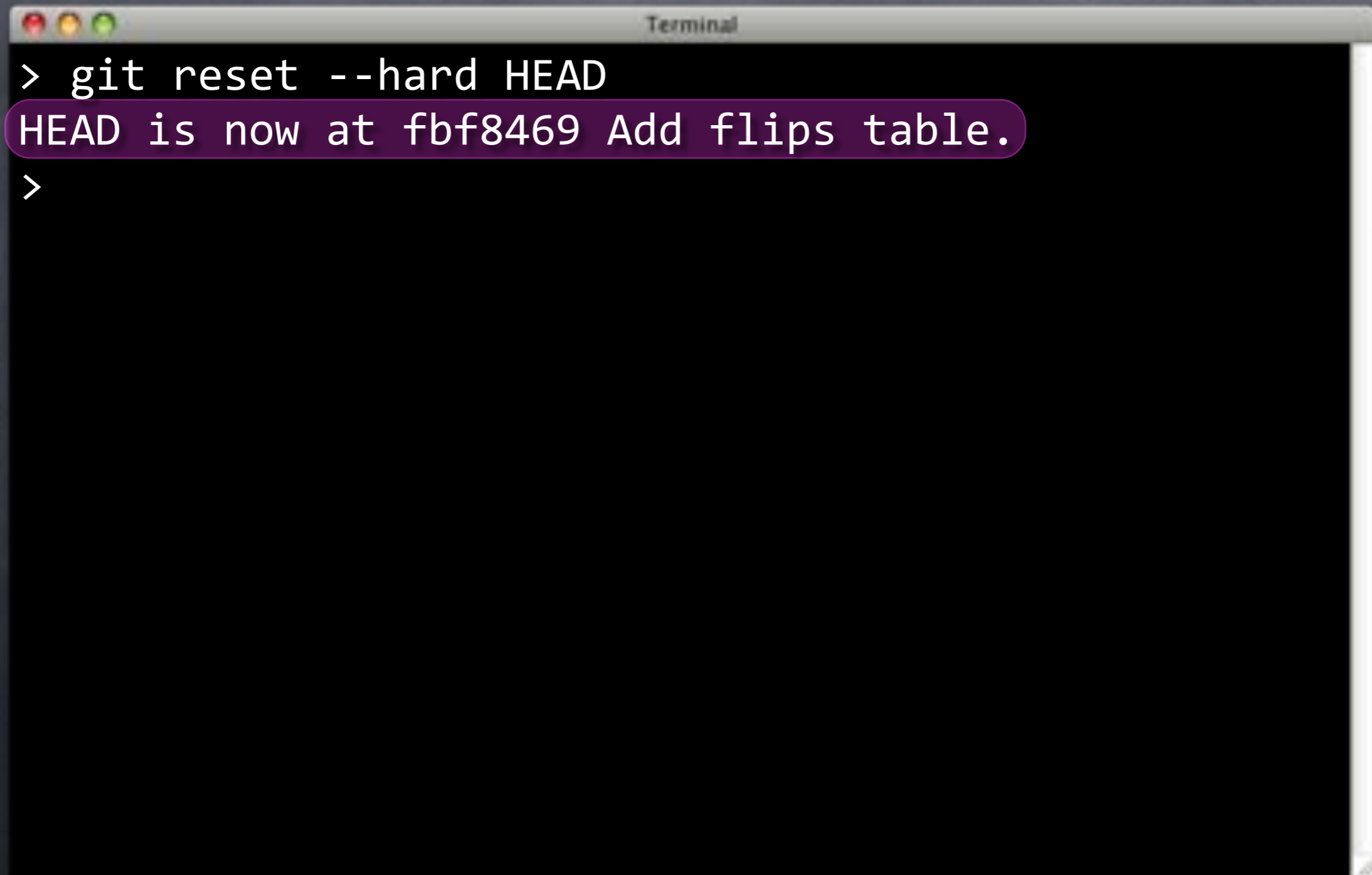
# Reset

```
> git reset --hard HEAD
```

```
HEAD is now at fbf8469 Add flips table.
```

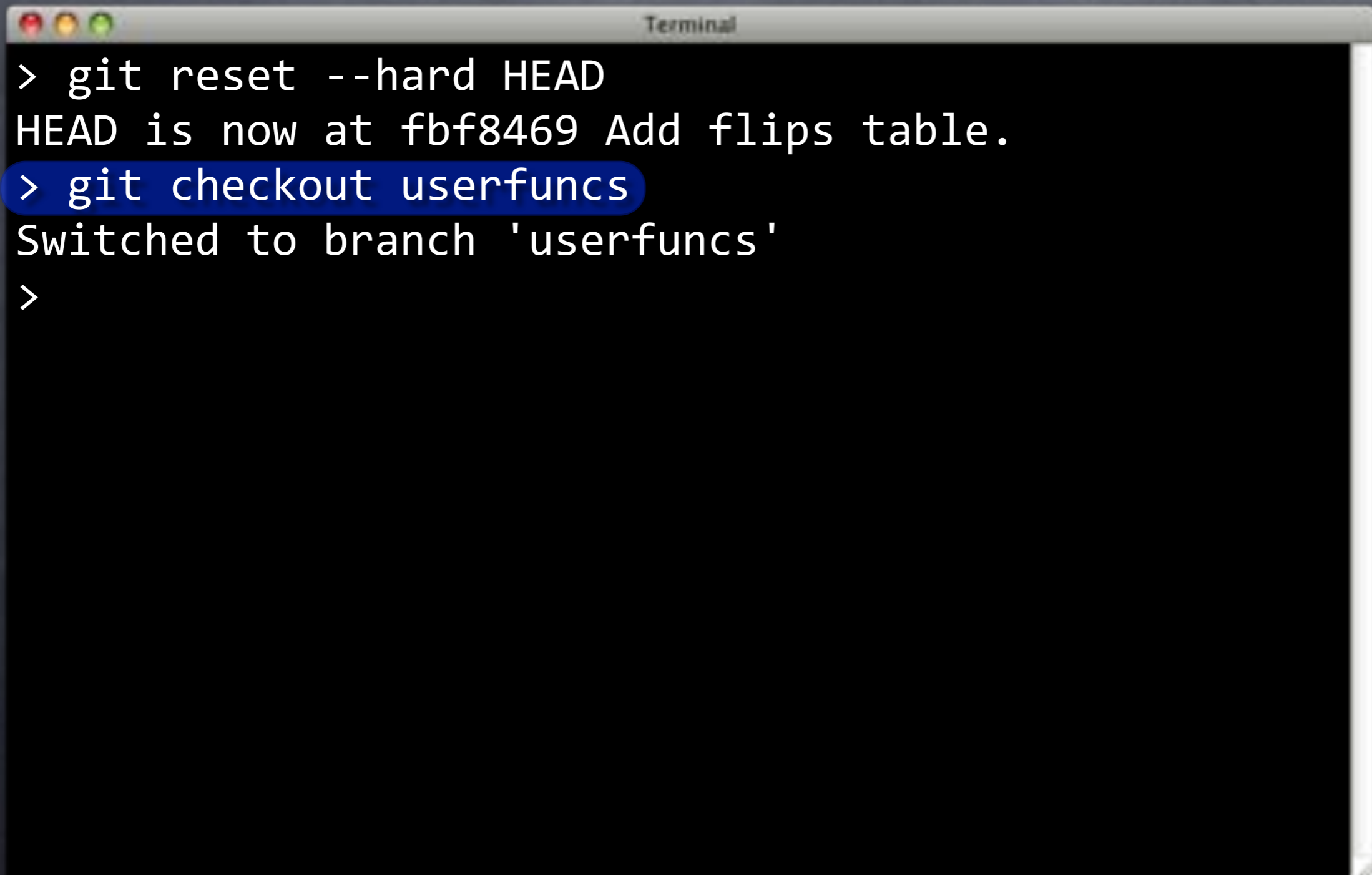
```
>
```

# Reset

A terminal window titled "Terminal" with a dark background and light text. The window shows a command prompt with the command `> git reset --hard HEAD`. The output is `HEAD is now at fbf8469 Add flips table.`, which is highlighted with a purple background. A second prompt `>` is visible below the output.

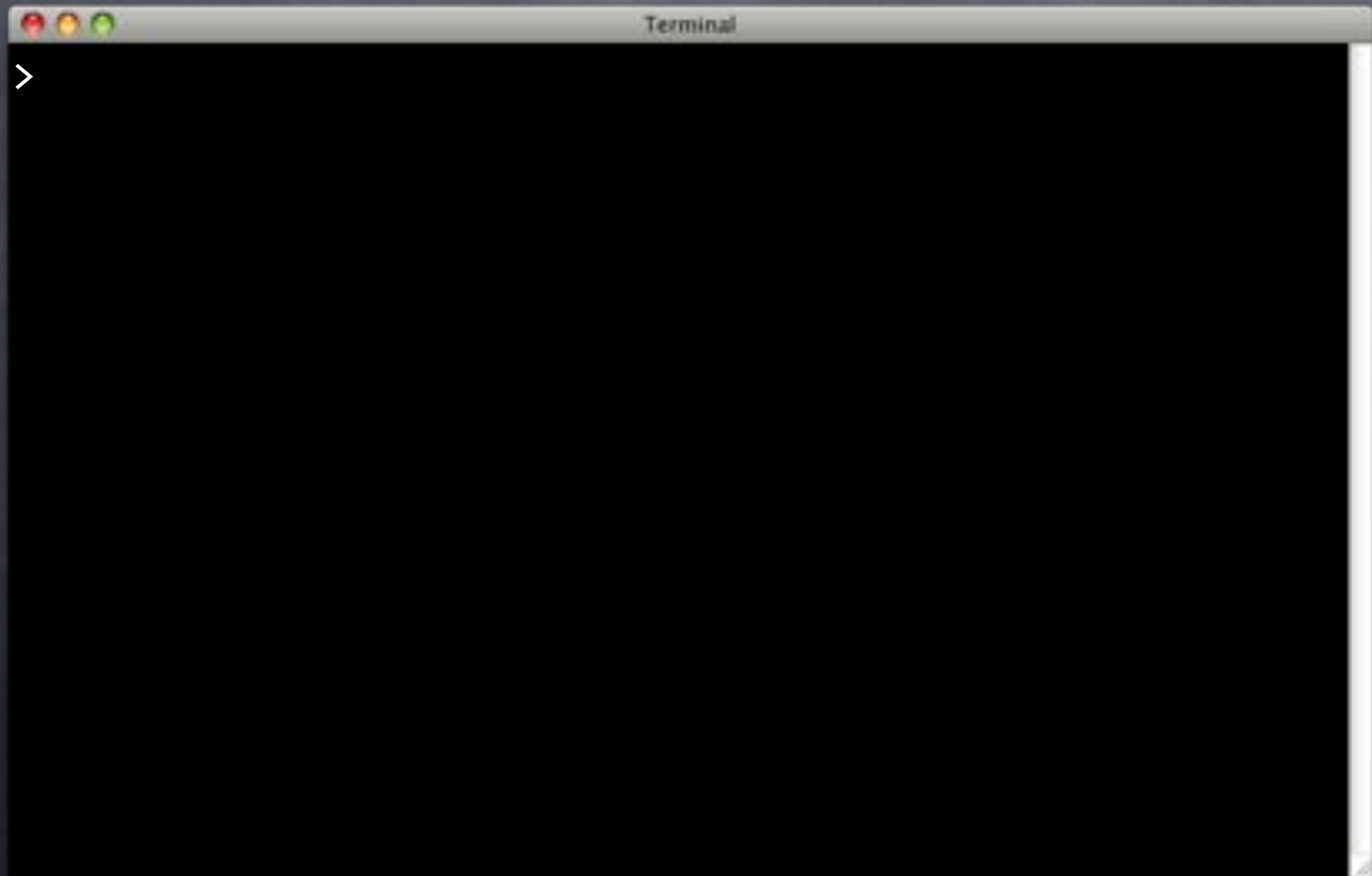
```
> git reset --hard HEAD
HEAD is now at fbf8469 Add flips table.
>
```

# Reset

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal shows the following text:

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

# Rebase



# Rebase

```
> 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.

```
> 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 --continue"  
If you prefer to skip this patch, run "git rebase --skip"  
To check out the original branch and stop rebasing, run "git rebase --abort".  
>

# Rebase

Here comes D...

```
> git rebase master
First, rewinding head to replay your work on top of the latest upstream changes.
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 --continue".  
If you prefer to skip this patch, run "git rebase --skip".  
To check out the original branch and stop rebasing, run "git rebase --abort".  
>

# 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
```

D'oh!

```
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 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 --continue"  
If you prefer to skip this patch, run "git rebase --skip"  
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



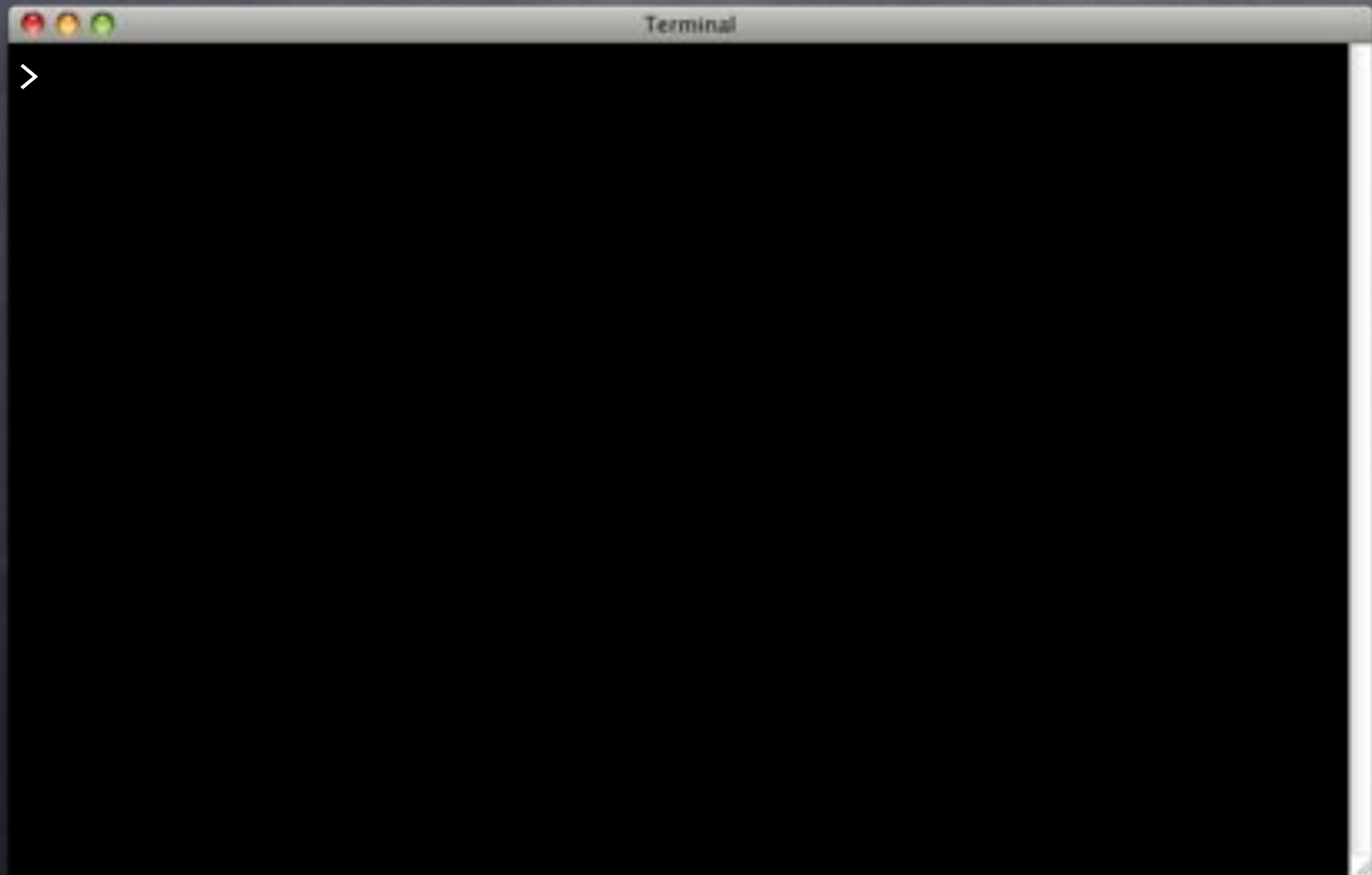


# Wha Happen?

- 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
    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 →

# 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
++=====
+ 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
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

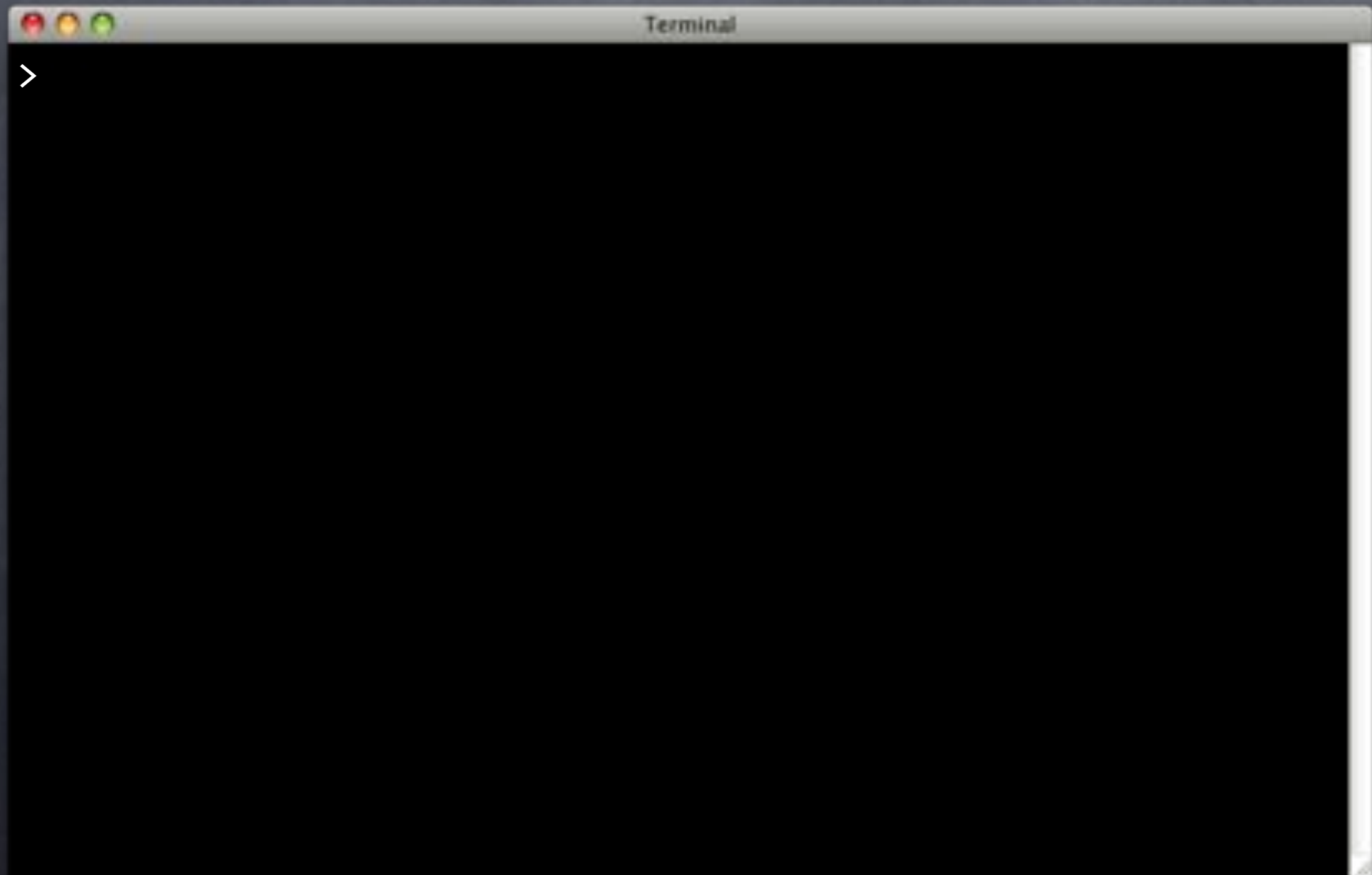


# Hallelunion

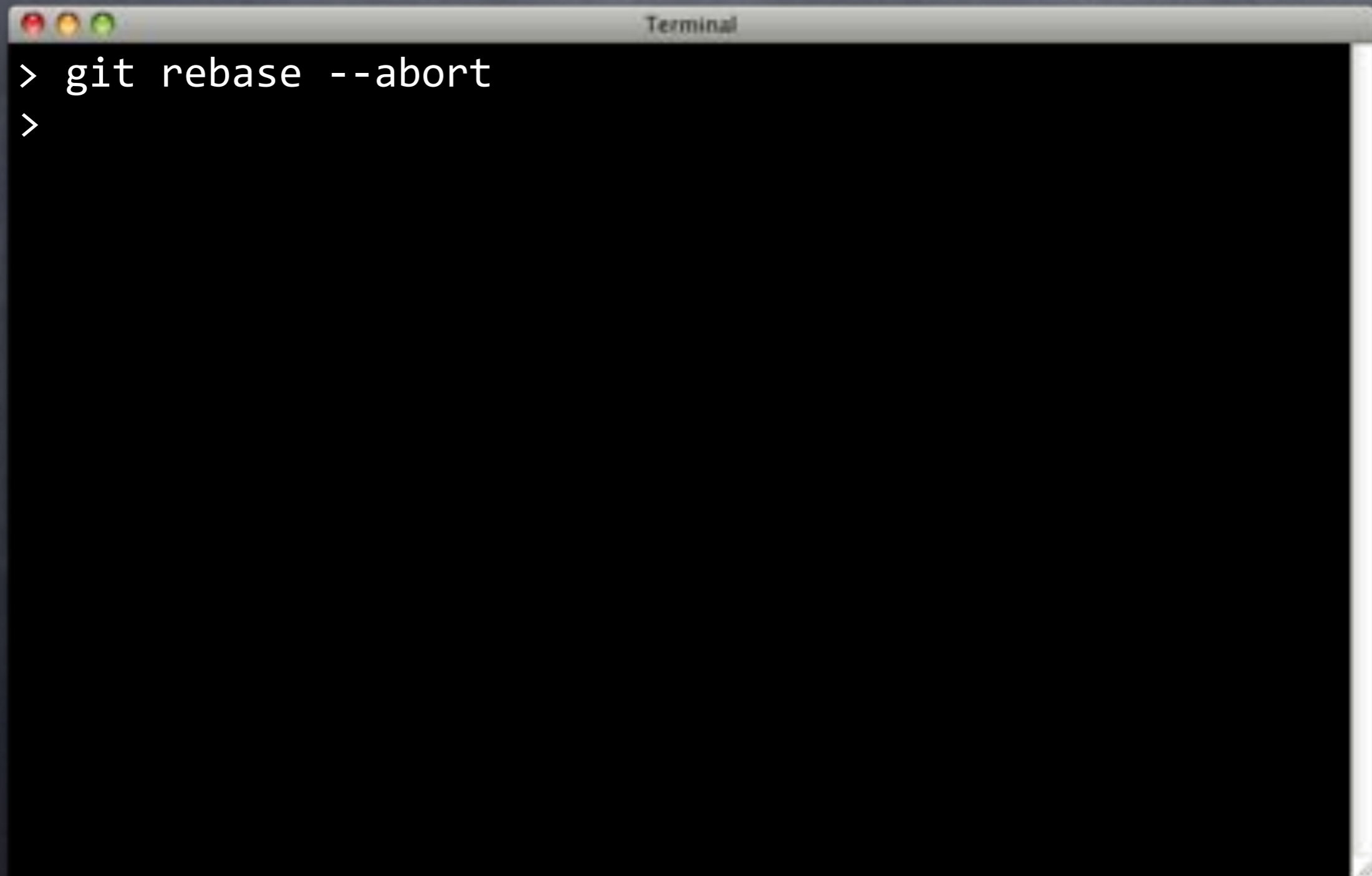
- Just appends lines
- Exactly how changes work
- Let's clean up our mess
- And try again



# Reemerge

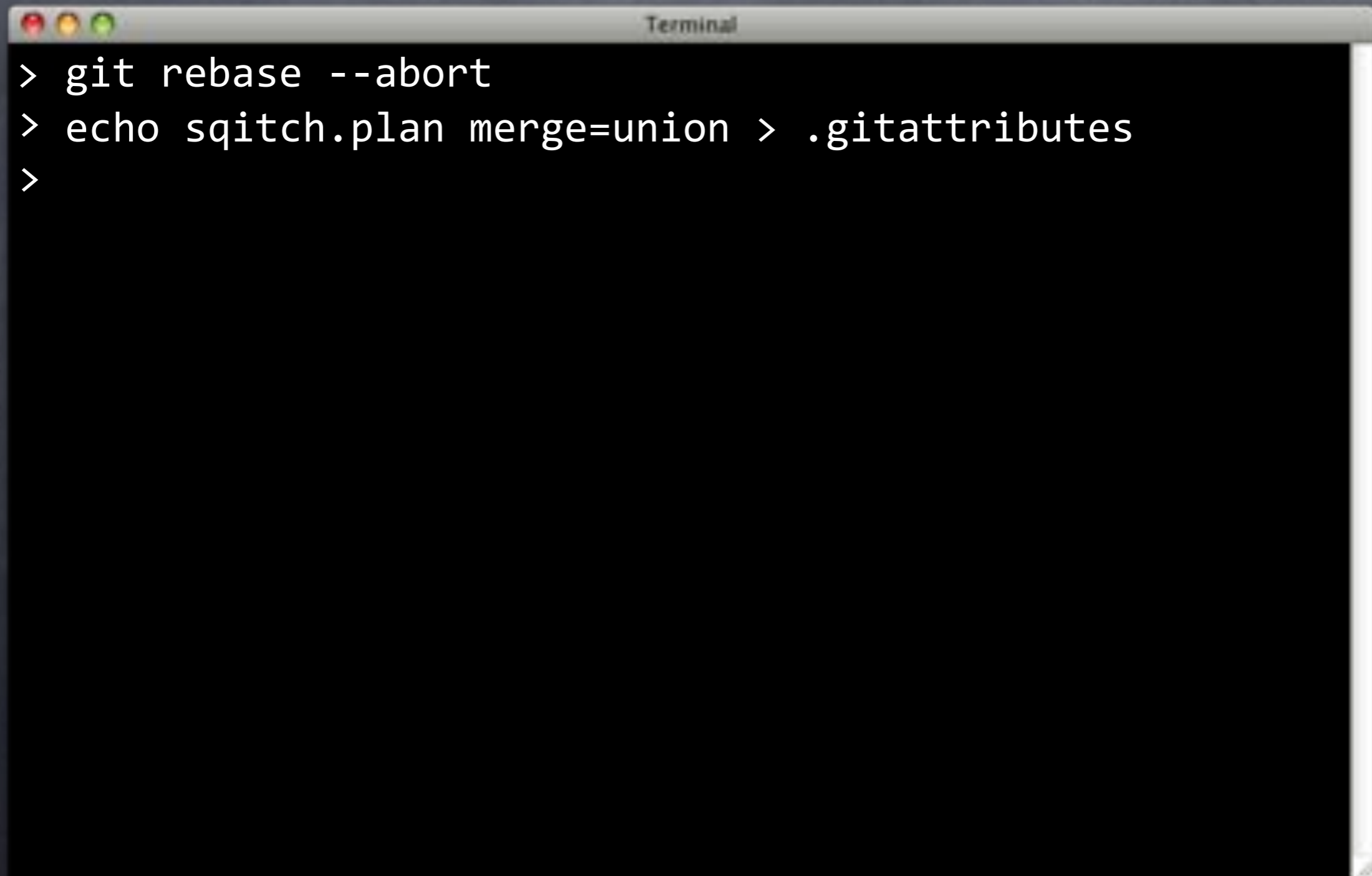


# Reemerge

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a prompt character followed by the command "git rebase --abort" and another prompt character on the next line.

```
> git rebase --abort  
>
```

# Reemerge

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal contains three lines of text, each starting with a greater-than sign (>).

```
> 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 merge C contributes
> 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=us
> 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 work on top of the latest upstream changes.
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
>
```

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
>
```

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 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 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?

```
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?

```
%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?

```
%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?

```
%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?

```
%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?

# Perfect

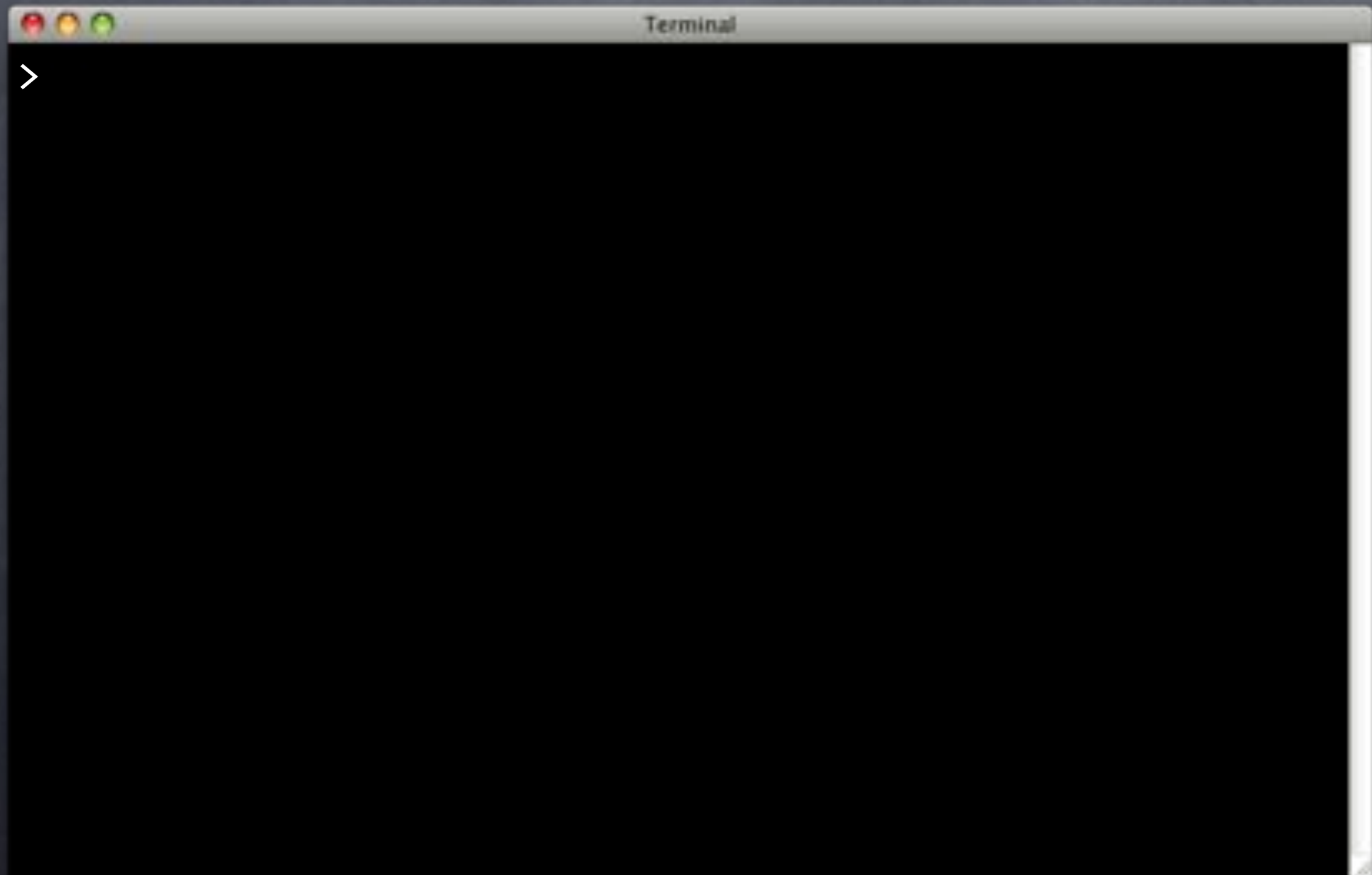
```
%syntax-version=1.0.0-b2
%project=flipr
%uri=file:///../flipr-remote
```

- `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])

# 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

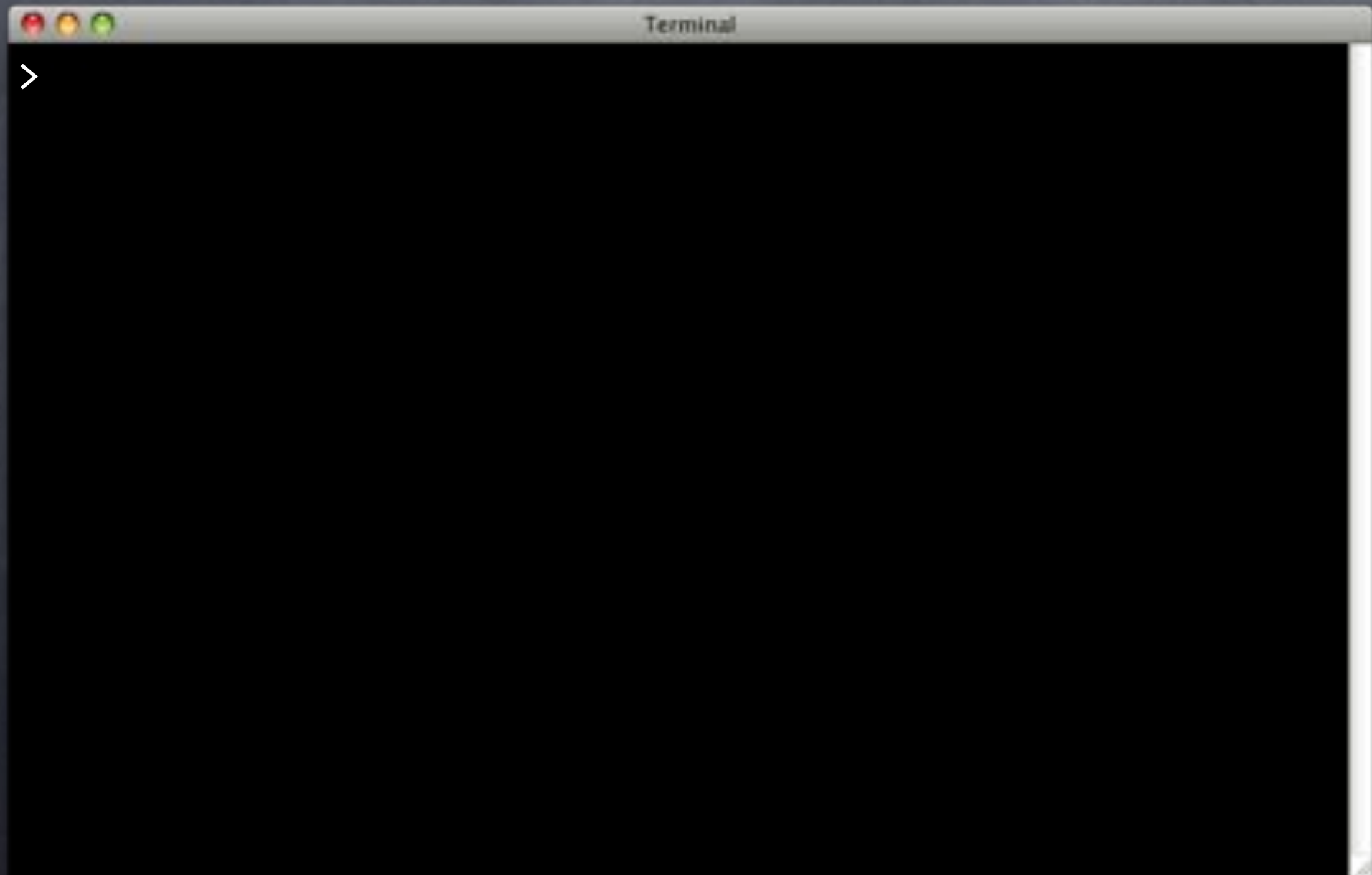
```
Terminal
> sqitch rebase -y
Reverting all changes from flipr_test
- change_pass .. ok
- insert_user .. ok
- users ..... ok
- appschema .... ok
Deploying changes to flipr_test
+ appschema .... ok
+ users ..... ok
+ flips ..... ok
+ insert_user .. ok
+ change_pass .. ok
>
```

# Work It

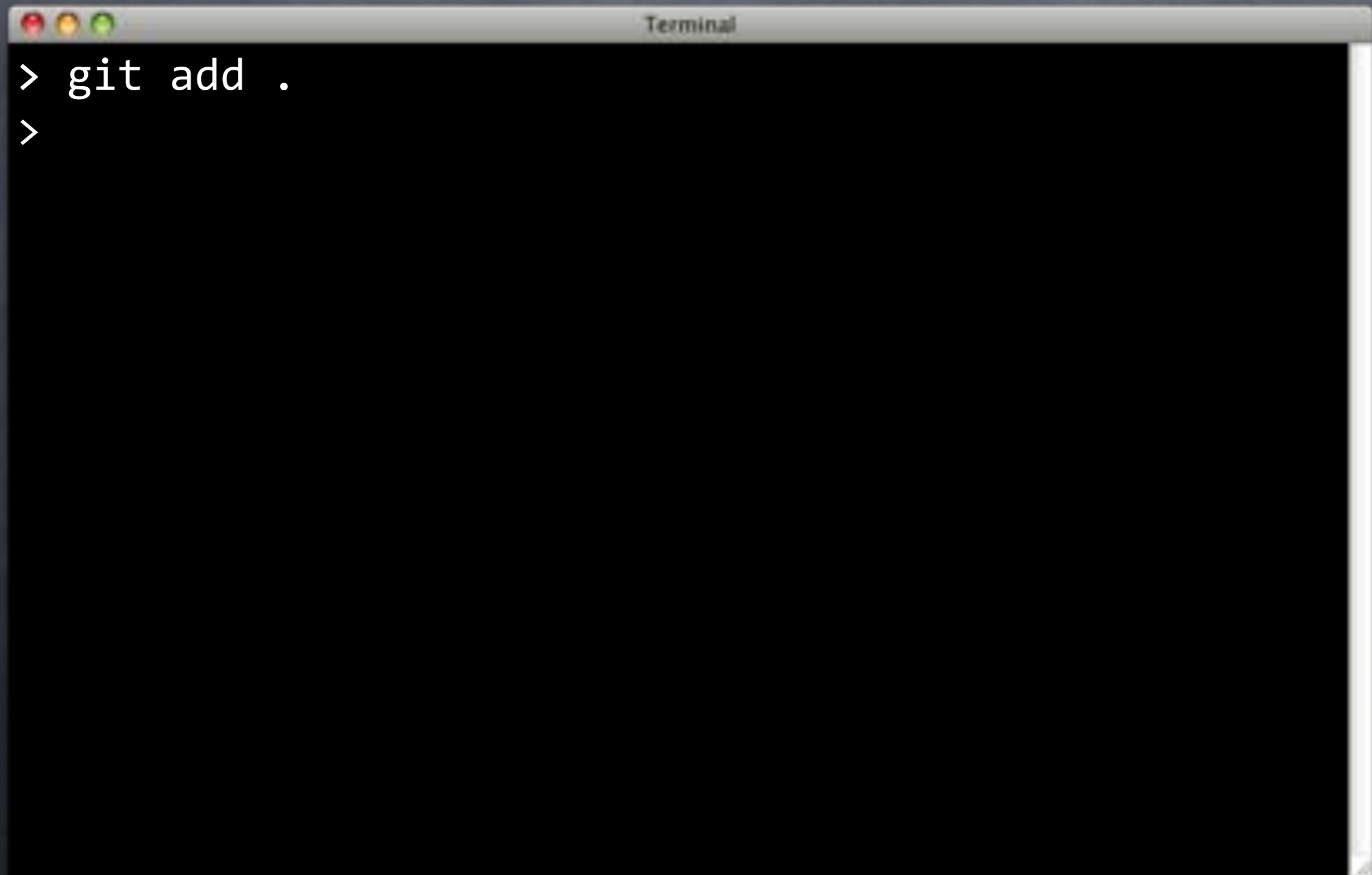
```
Terminal
> sqitch rebase -y
Reverting all changes from flipr_test
- change_pass .. ok
- insert_user .. ok
- users ..... ok
- appschema .... ok
Deploying changes to flipr_test
+ appschema .... ok
+ users ..... ok
+ flips ..... ok
+ insert_user .. ok
+ change_pass .. ok
>
```

# Awesomsauce

# Make it So



# Make it So

A screenshot of a macOS Terminal window. The window title is "Terminal". The terminal content shows the command `> git add .` followed by a new prompt `>` on the next line. The terminal background is black, and the text is white. The window has a standard macOS title bar with red, yellow, and green window control buttons on the left.

```
> git add .  
>
```

# 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

```
Terminal
> git add .
> git commit -m 'merge for `sqitch.plan`.'
[userfuncs (root)]: 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)
>
```

Overwrite history

> 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

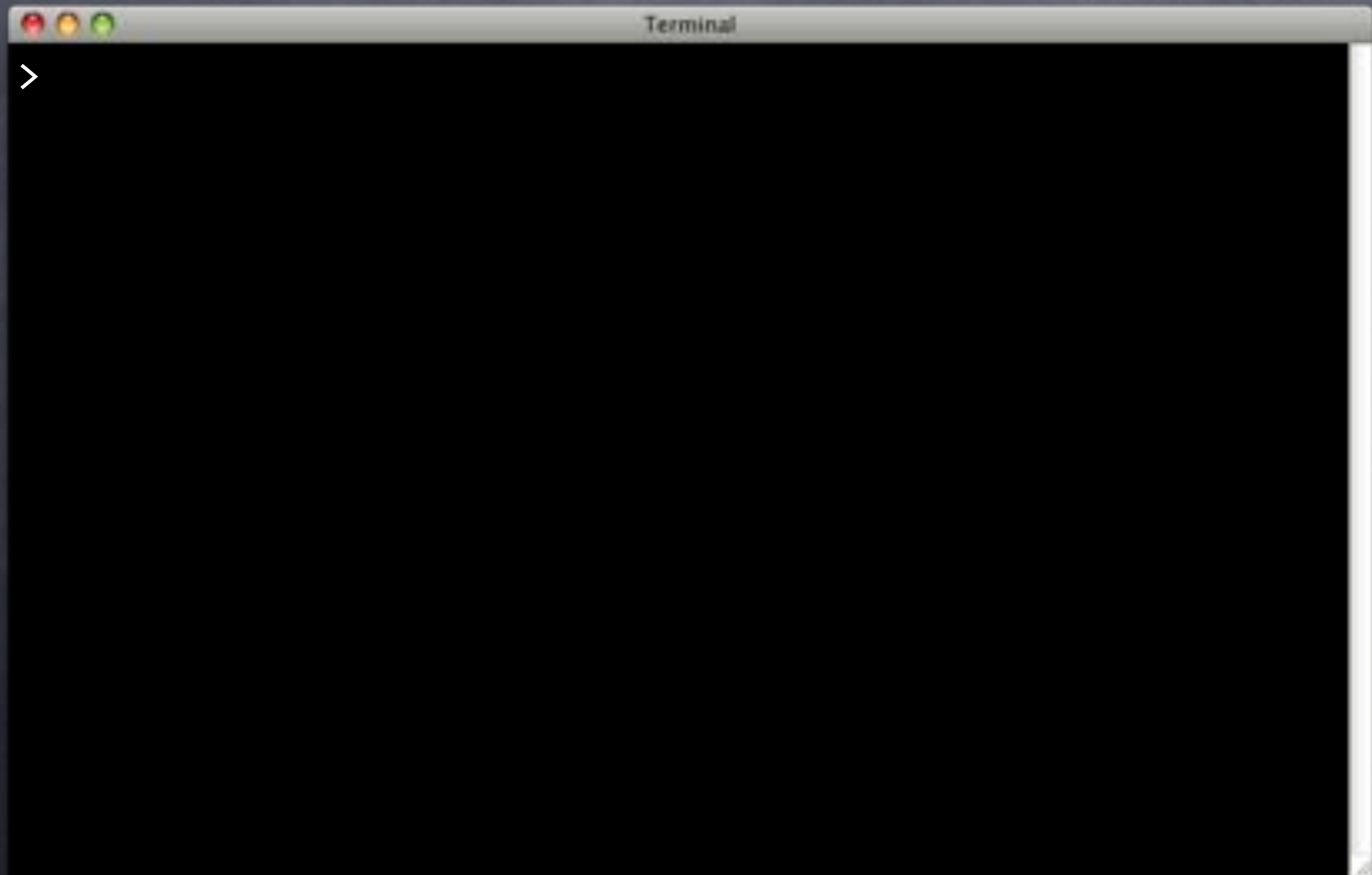
>

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

```
> 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
>
```

```
> 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
>
```

# 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
>
```





# 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

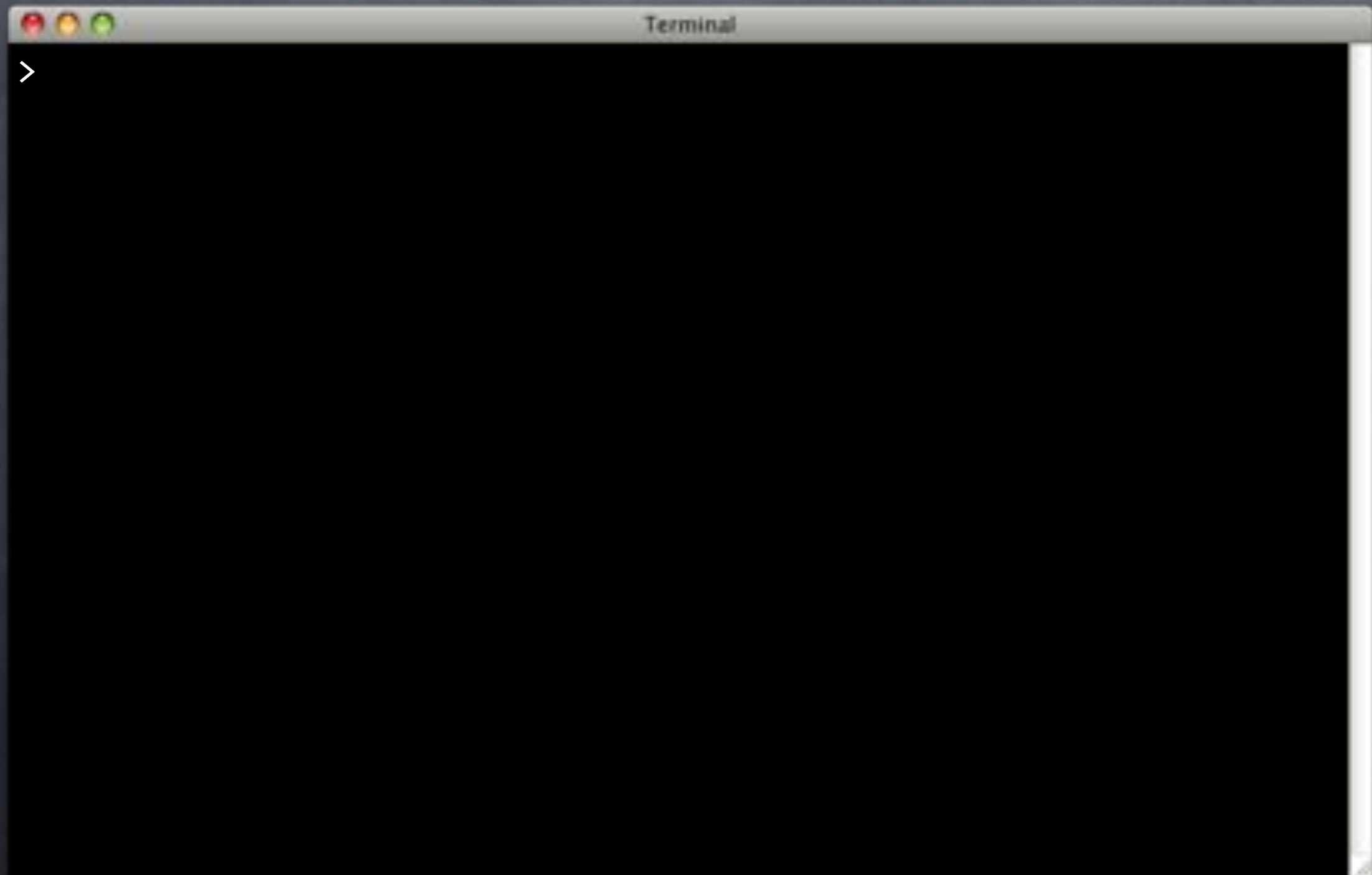


# Ship Shape

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

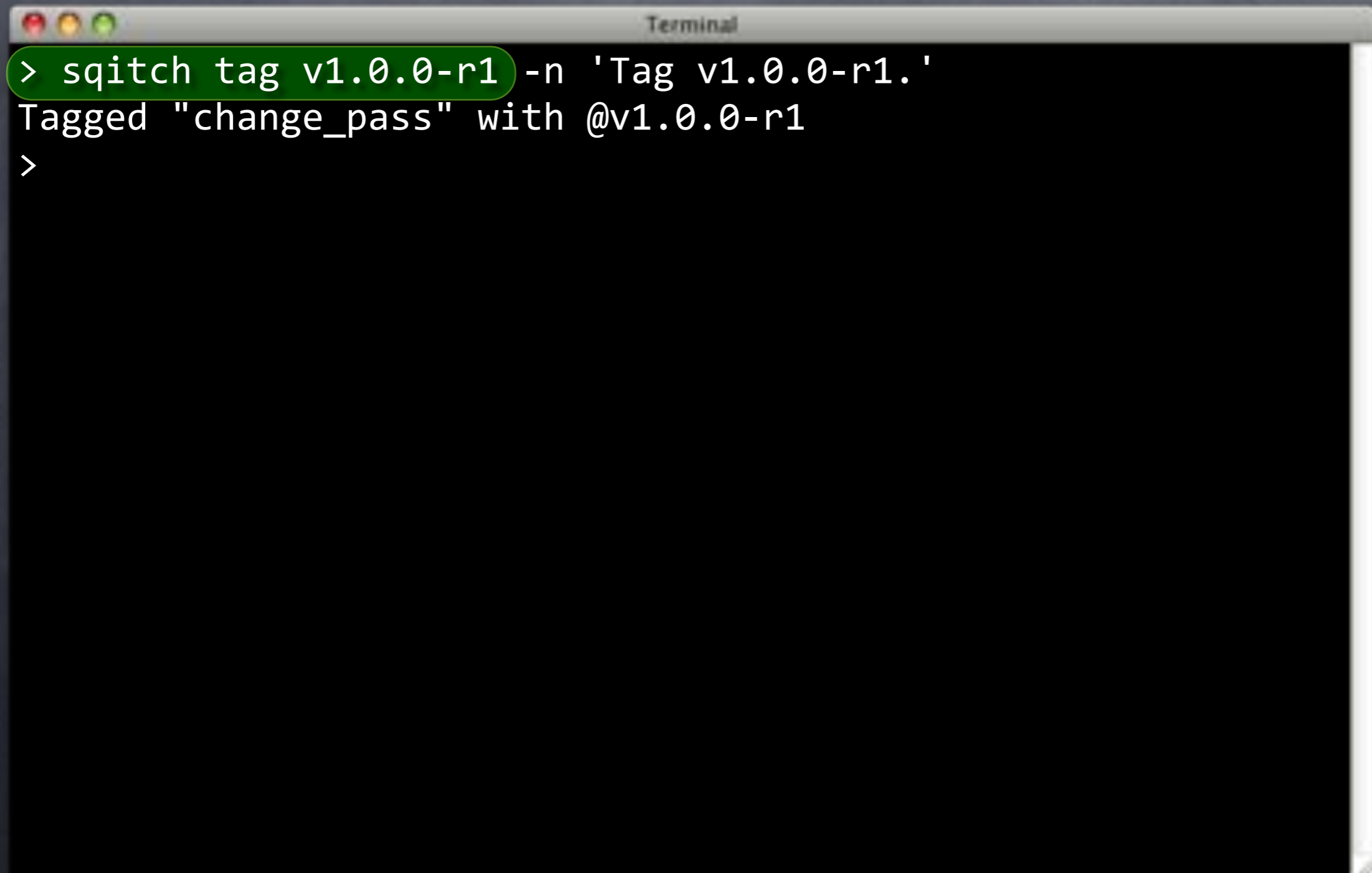


# You're It





# 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.

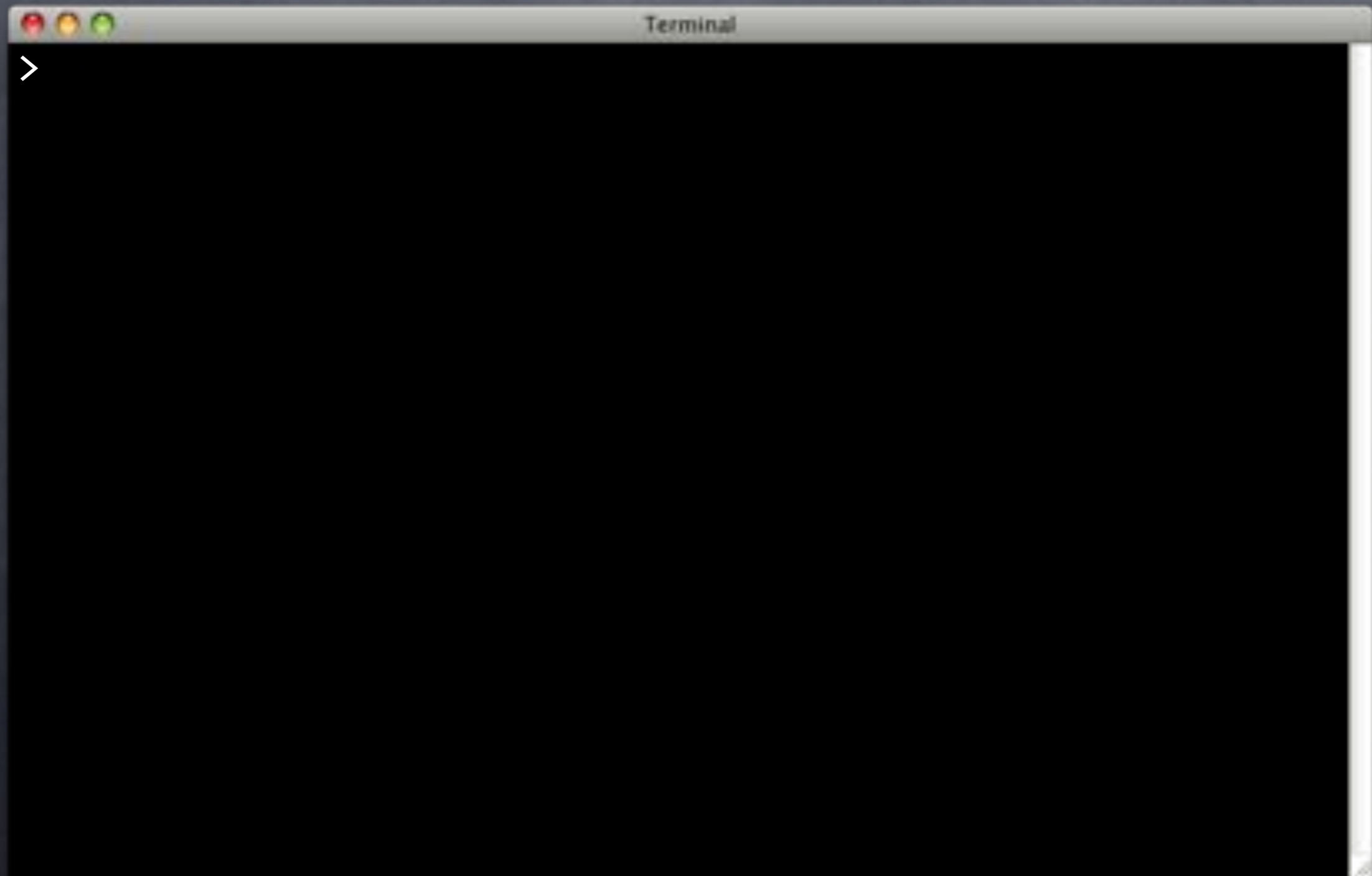
# You're It

```
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
>
```

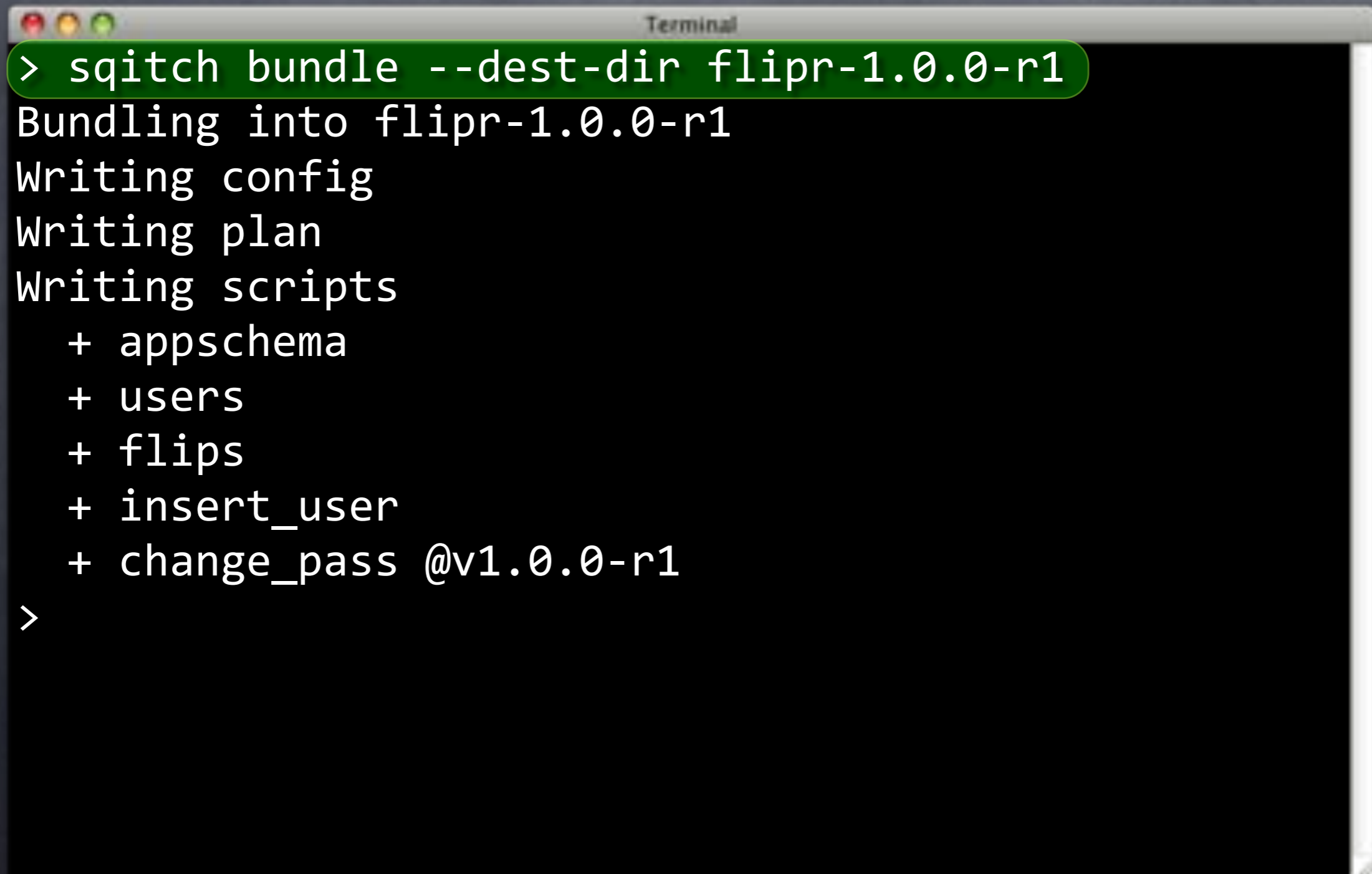
# You're It

```
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

A terminal window titled "Terminal" with a dark background and a light gray title bar. The terminal shows the execution of the command "sqitch bundle --dest-dir flipr-1.0.0-r1". The command is highlighted in a green box. The output shows the bundling process, including writing config, plan, and scripts, and listing the bundled components: appschema, users, flips, insert\_user, and change\_pass @v1.0.0-r1. The terminal ends with a prompt character ">".

```
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
>
```

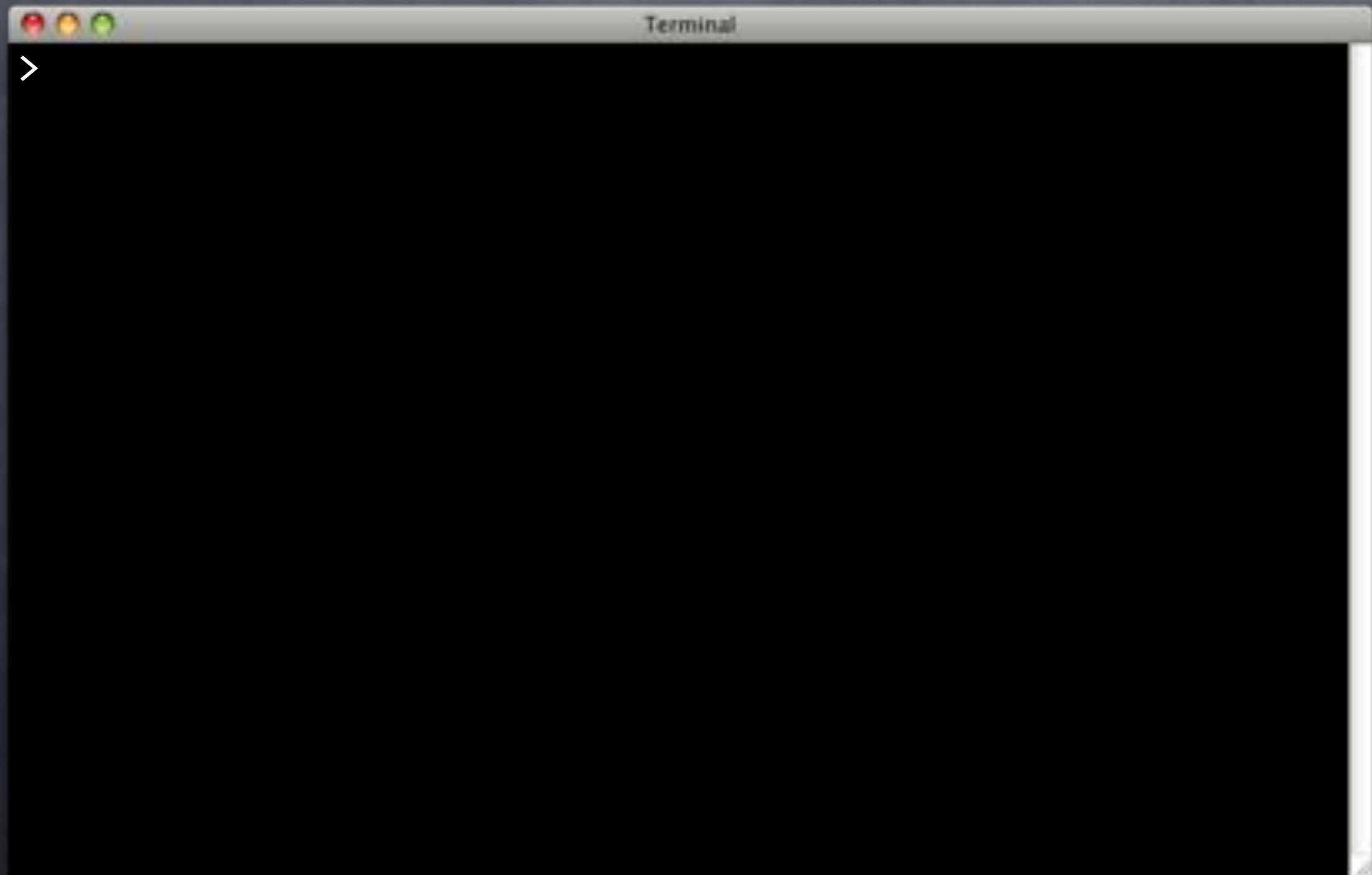
# Bundle Up

```
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?

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a prompt character followed by the command "cd flipr-1.0.0-r1" and another prompt character on the next line.

```
> cd flipr-1.0.0-r1  
>
```

# Bundled?

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows three lines of shell commands: a prompt followed by "cd flipr-1.0.0-r1", another prompt followed by "createdb flipr\_qa", and a third prompt. The terminal background is black and the text is white.

```
> cd flipr-1.0.0-r1
> createdb flipr_qa
>
```

# 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

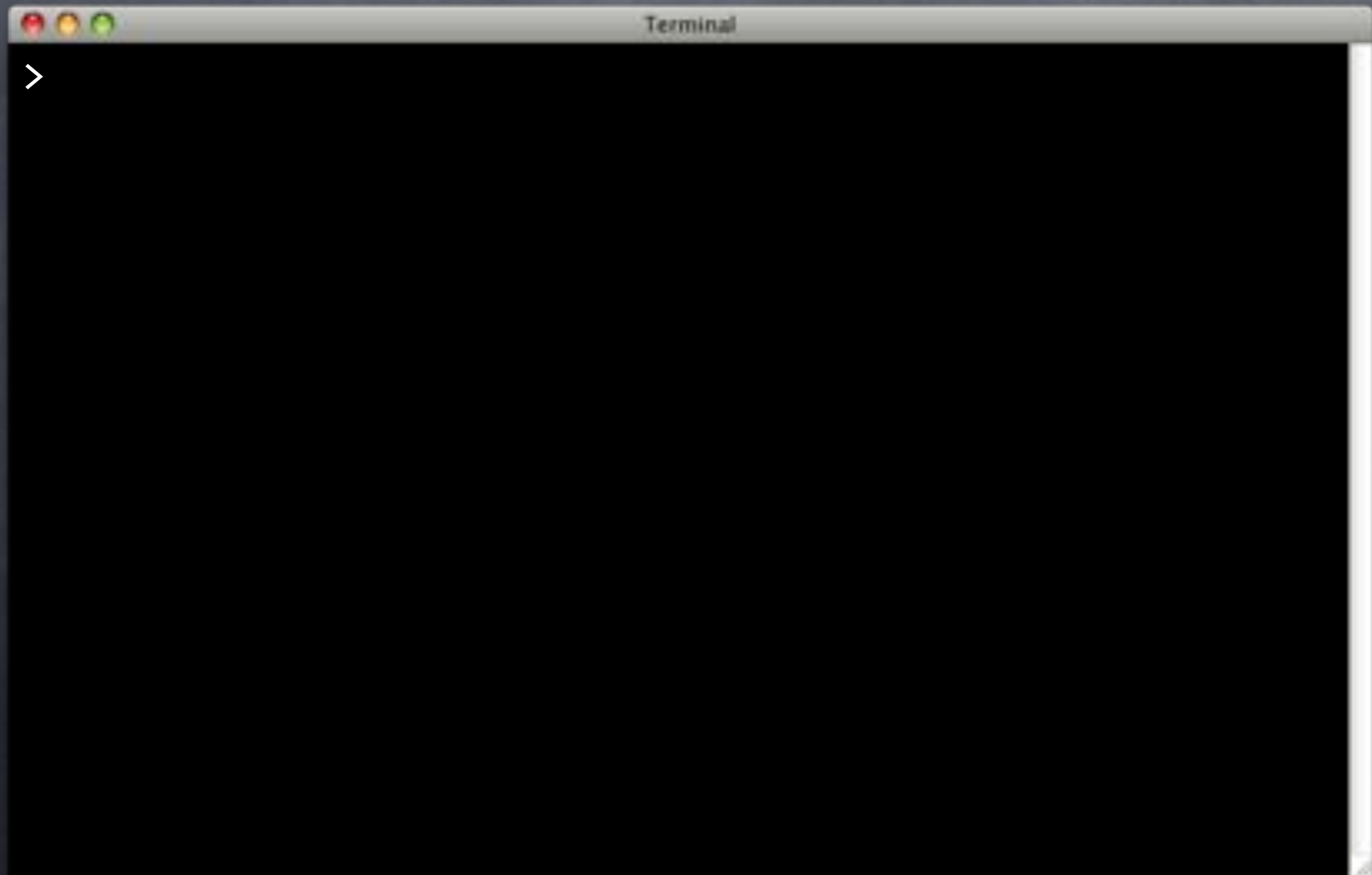


# Merge Madness

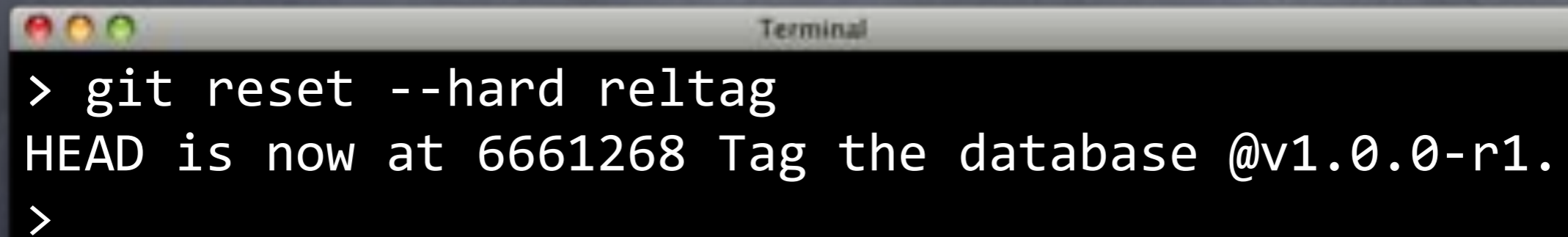
- Merge everything
- Back to master
  - users
  - flips
  - userfuncs
- Union merge sqitch.plan
- Bundle and ship
- <https://github.com/theory/agile-flipr.git>



# Ruh-Roh



# Ruh-Roh

A terminal window with a title bar that says "Terminal". The window contains the following text:

```
> git reset --hard re1tag  
HEAD is now at 6661268 Tag the database @v1.0.0-r1.  
>
```



# Ruh-Roh

```
Terminal
> git reset --hard reitag
HEAD is now at 6661268 Tag the database
> 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

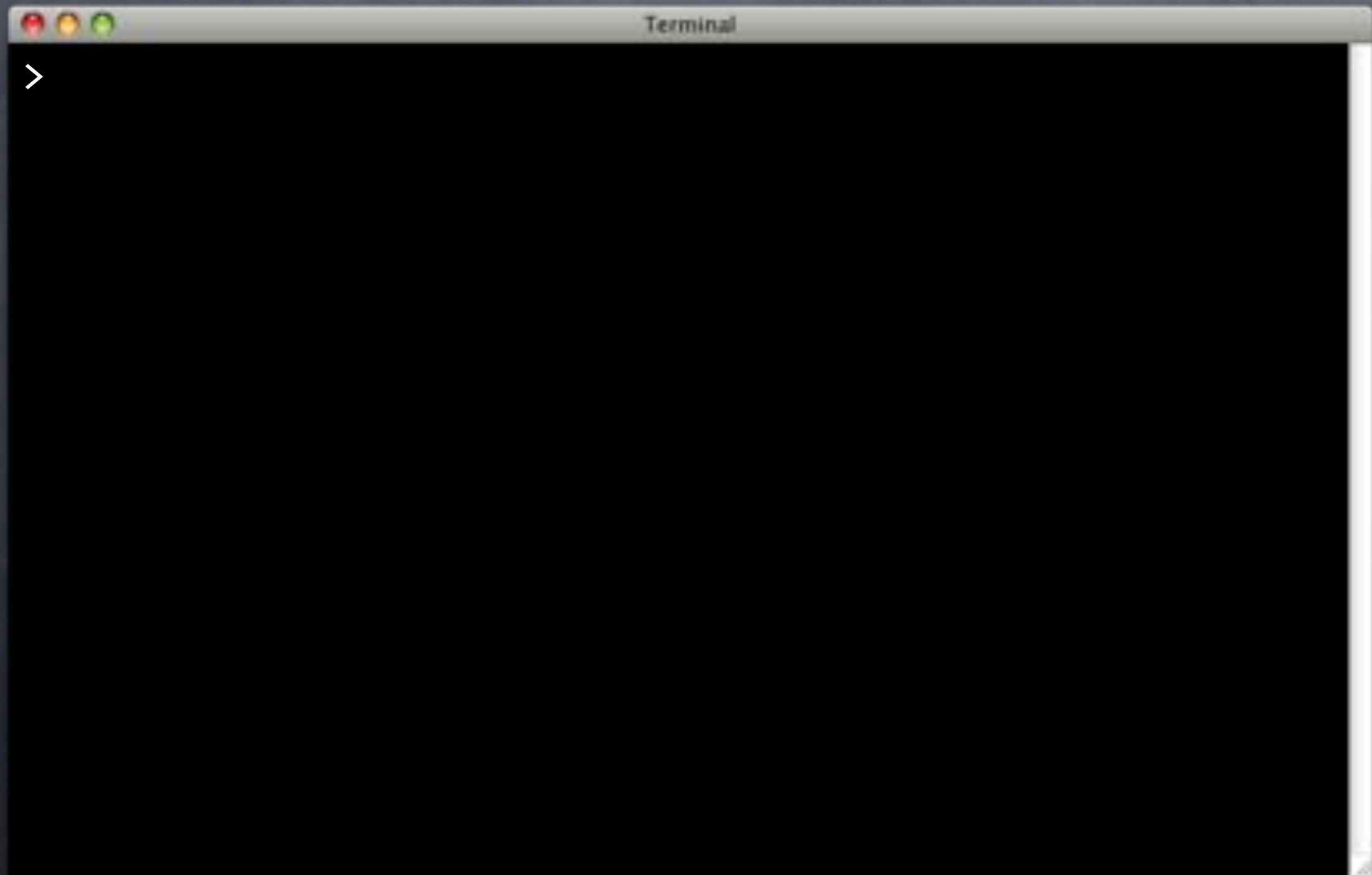
```
Terminal
> git reset --hard reltag
HEAD is now at 6661268 Tag the database @v1.0.0-r1.
> psql -d flipr_test -c "
    SELECT flipr.insert_user('foo', 'secr3t'),
           flipr.insert_user('bar', 'secr3t');
    SELECT nickname, password FROM flipr.users;
"
nickname | password
-----+-----
foo    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

# Ruh-Roh

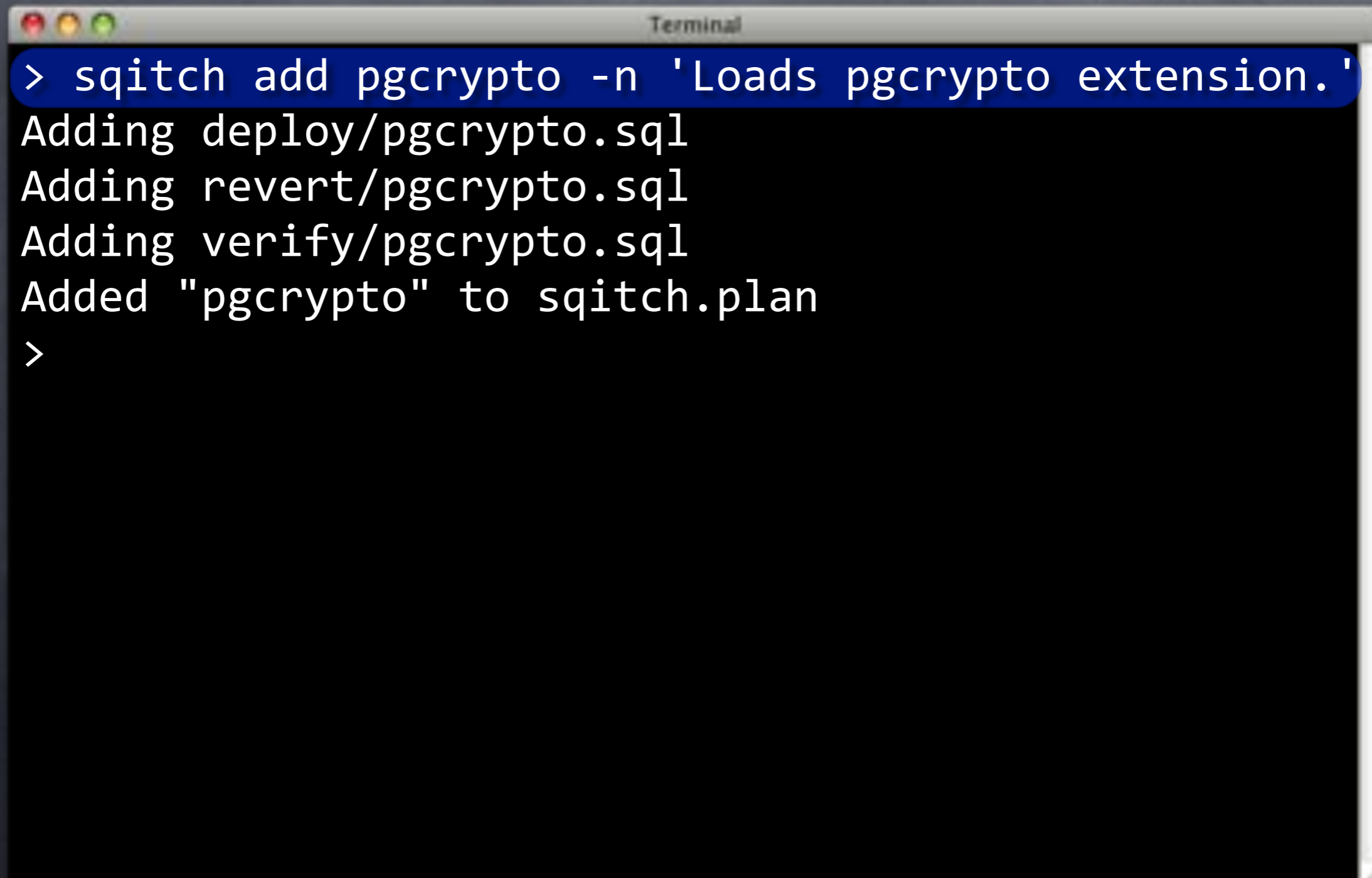
```
Terminal
> git reset --hard reltag
HEAD is now at 6661268 Tag the database @v1.0.0-r1.
> psql -d flipr_test -c "
    SELECT flipr.insert_user('foo', 'secr3t'),
           flipr.insert_user('bar', 'secr3t');
    SELECT nickname, password FROM flipr.users;
"
nickname | password
-----+-----
foo    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

**Not good.**

# PGCryptonite

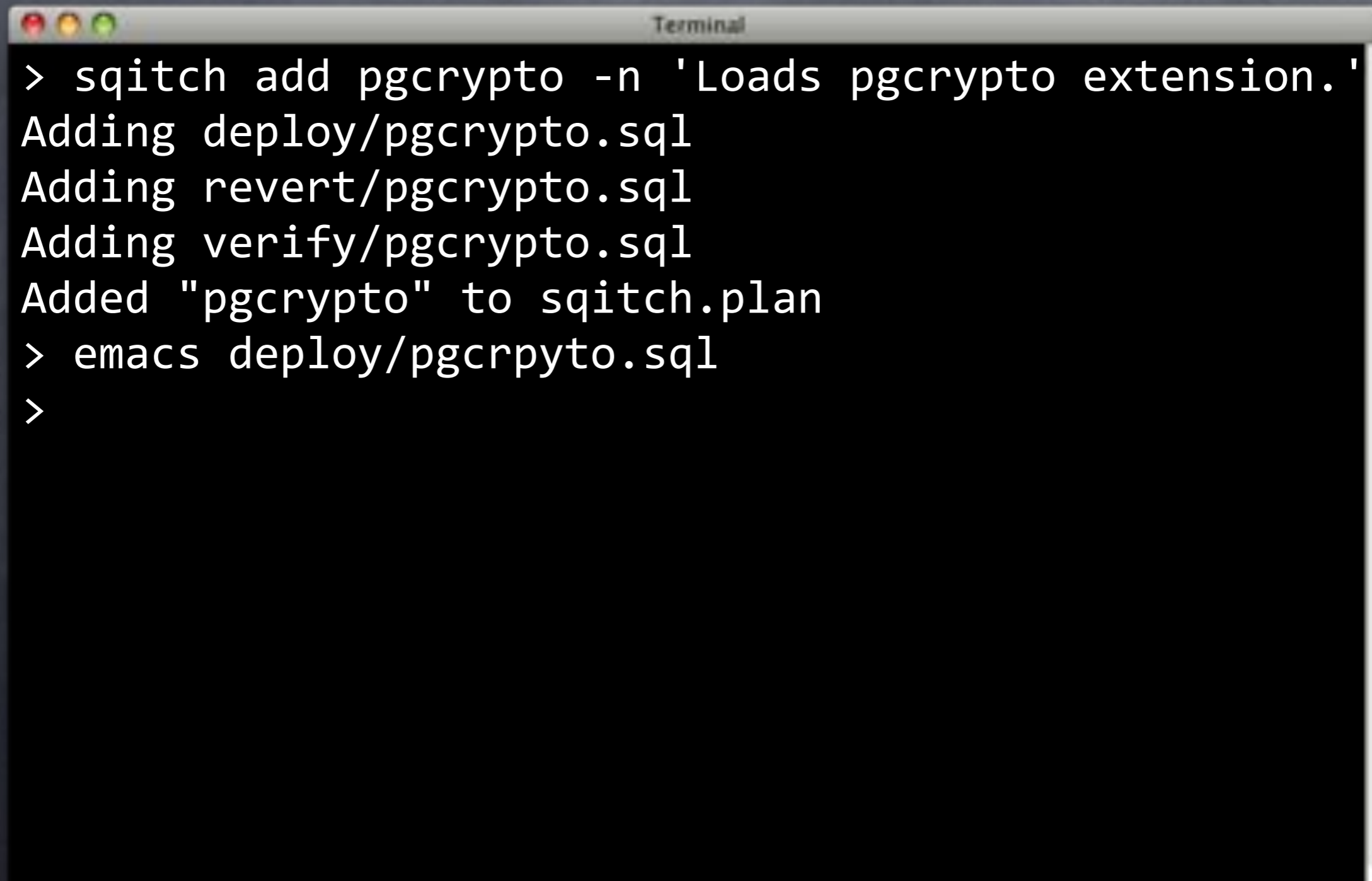


# PGCryptonite



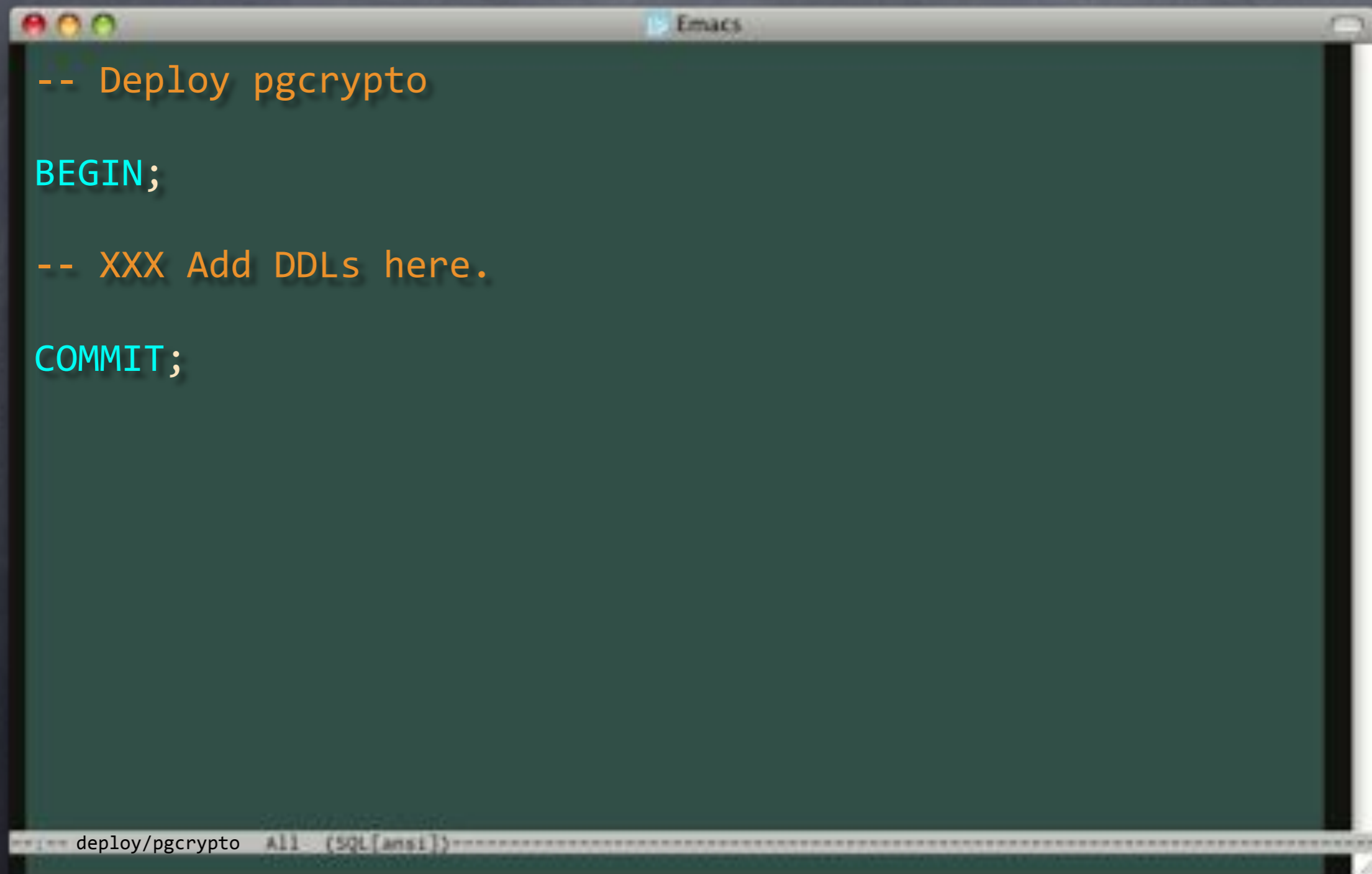
```
Terminal
> sqitch add pgcrypto -n 'Loads pgcrypto extension.'
Adding deploy/pgcrypto.sql
Adding revert/pgcrypto.sql
Adding verify/pgcrypto.sql
Added "pgcrypto" to sqitch.plan
>
```

# 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
>
```

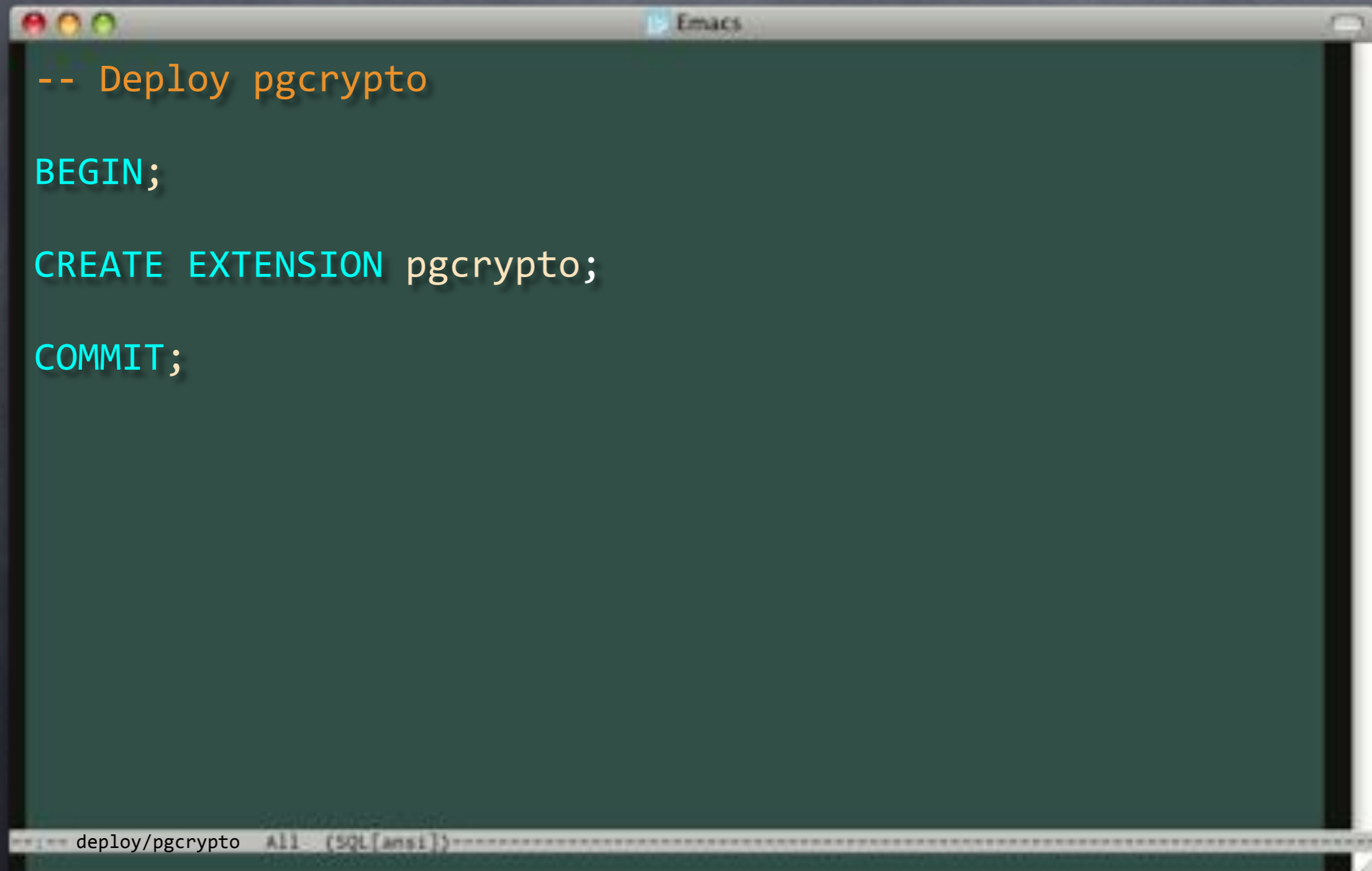
# deploy/pgcrypto.sql

A screenshot of an Emacs editor window. The window title is "Emacs". The background is a dark green color. The text is as follows:

```
-- Deploy pgcrypto  
  
BEGIN;  
  
-- XXX Add DDLs here.  
  
COMMIT;
```

At the bottom of the window, there is a status bar with the text "deploy/pgcrypto All (SQL[ansi])".

# deploy/pgcrypto.sql



```
-- Deploy pgcrypto

BEGIN;

CREATE EXTENSION pgcrypto;

COMMIT;
```

The screenshot shows an Emacs editor window with a dark green background. The title bar at the top reads "Emacs". The code is displayed in a monospaced font with syntax highlighting: the comment "-- Deploy pgcrypto" is in orange, and the SQL keywords "BEGIN;", "CREATE EXTENSION pgcrypto;", and "COMMIT;" are in cyan. The status bar at the bottom of the window shows "deploy/pgcrypto All (SQL[ansi])".



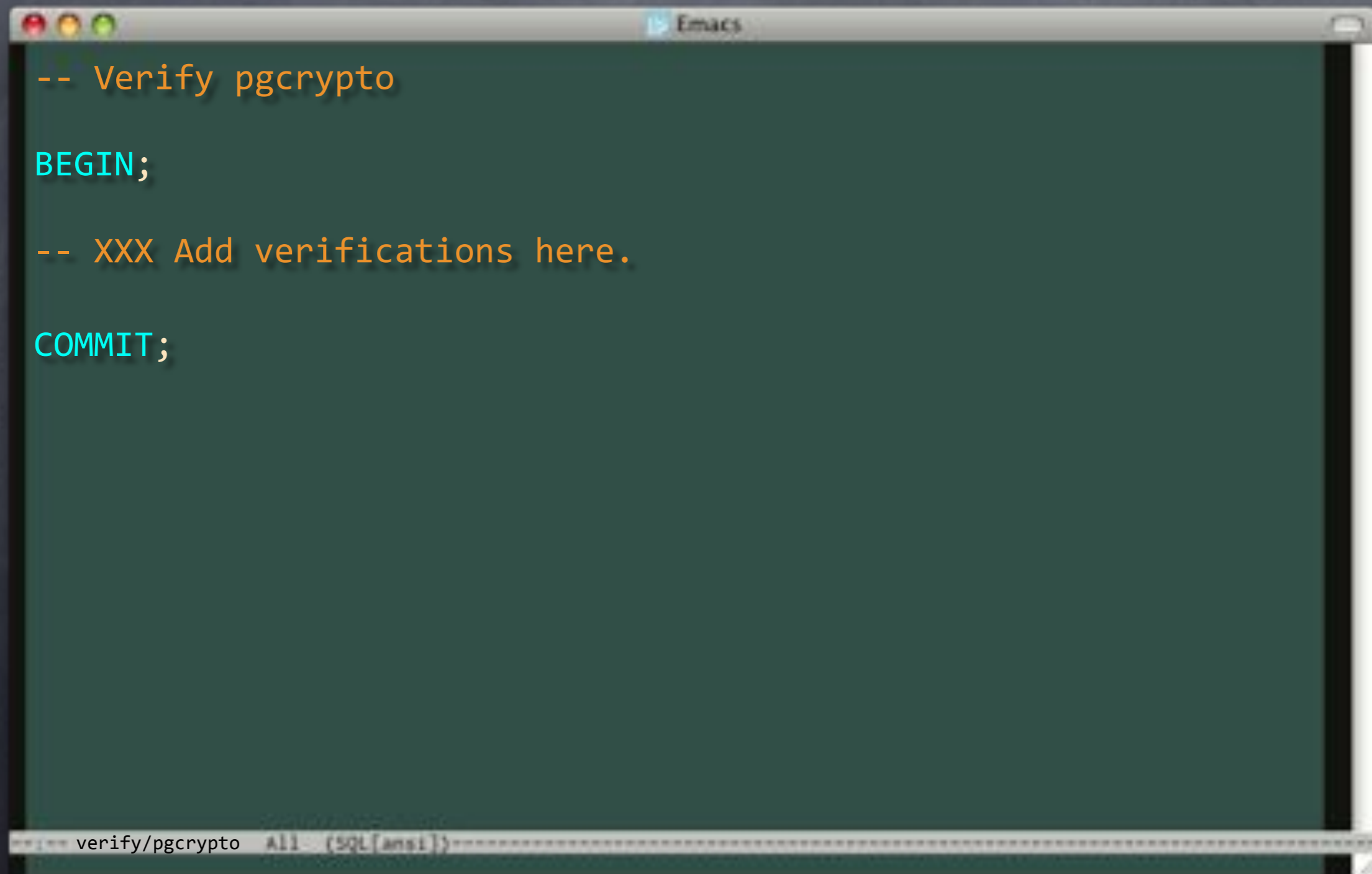
# PGCryptonite

```
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
>
```

# 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



```
-- Verify pgcrypto

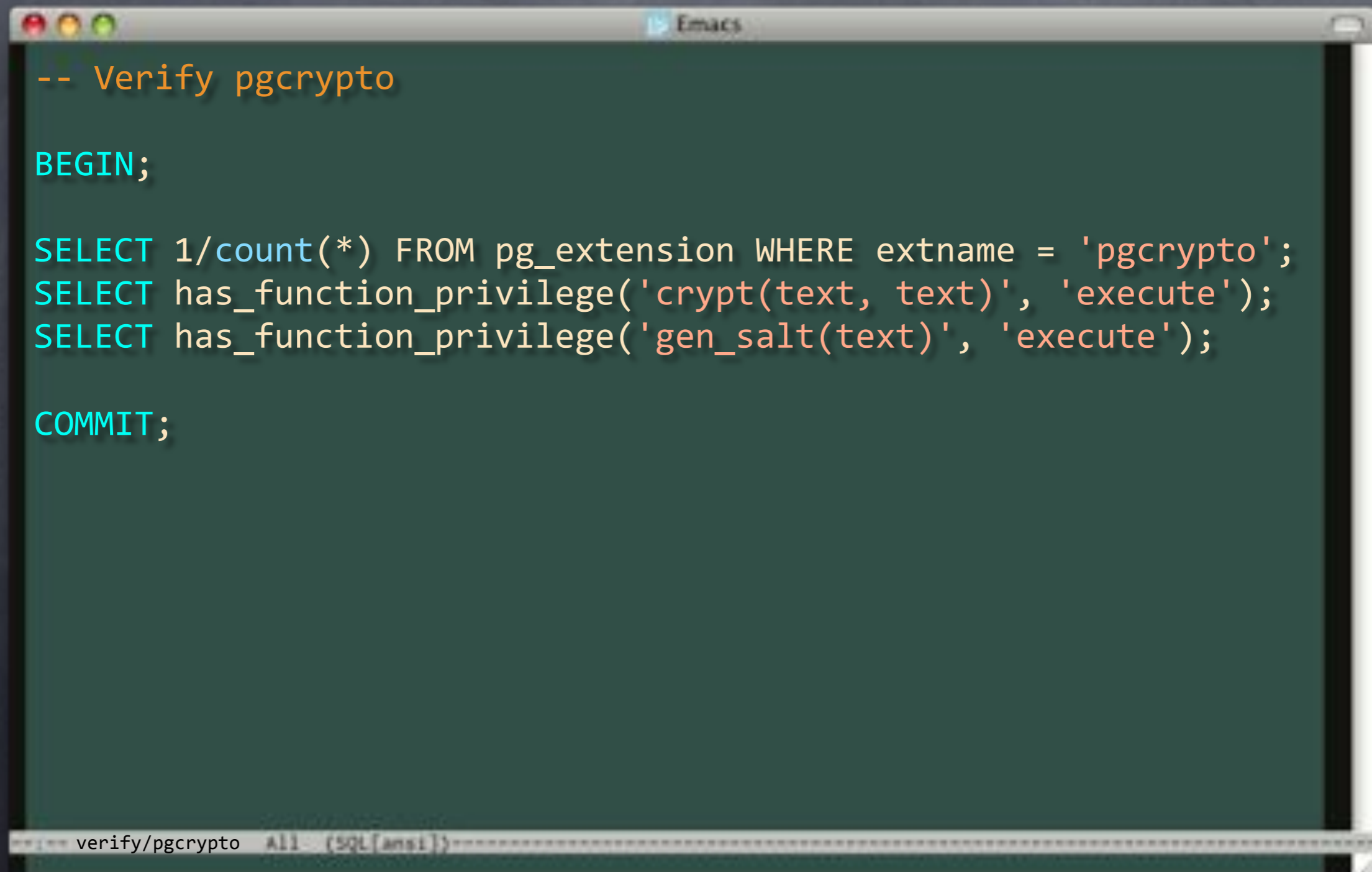
BEGIN;

-- XXX Add verifications here.

COMMIT;
```

The screenshot shows an Emacs editor window with a dark green background. The code is displayed in a monospaced font with syntax highlighting: comments are orange, and SQL keywords are cyan. The window title bar shows 'Emacs' and standard macOS window controls. The status bar at the bottom indicates the file path 'verify/pgcrypto', the buffer name 'All', and the encoding '(SQL[ansi])'.

# verify/pgcrypto.sql



```
-- 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

```
-- 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.sql

```
-- 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;
```

# 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

>

```
> 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' );
```

```
> 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' );
```

```
SELECT lives_ok(
  $$ SELECT insert_user('strongrr1', 'w00t') $$,
  'Insert another user'
);

-SELECT bag_eq(
-  'SELECT * FROM users',
-  $$ VALUES
-    ('theory', md5('foo'), NOW()),
-    ('strongrr1', md5('w00t'), NOW())
-  $$,
-  '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 = 'strongrr1'
+  AND password = crypt('w00t', password)
+), 'The second user should have been inserted' );

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

```
SELECT lives_ok(  
    $$ SELECT insert_user('strongrr1', 'w00t') $$,  
    'Insert another user'  
);
```

```
-SELECT bag_eq(  
-    'SELECT * FROM users',  
-    $$ VALUES  
-        ('theory', md5('foo'), NOW()),  
-        ('strongrr1', md5('w00t'), NOW())  
-    $$,  
-    '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 = 'strongrr1'  
+    AND password = crypt('w00t', password)  
+), 'The second user should have been inserted' );
```

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

```
SELECT lives_ok(
  $$ SELECT insert_user('strongrr1', 'w00t') $$,
  'Insert another user'
);

-SELECT bag_eq(
-  'SELECT * FROM users',
-  $$ VALUES
-    ('theory', md5('foo'), NOW()),
-    ('strongrr1', md5('w00t'), NOW())
-  $$,
-  '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 = 'strongrr1'
+  AND password = crypt('w00t', password)
+), 'The second user should have been inserted' );

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

```
SELECT lives_ok(
  $$ SELECT insert_user('strongrr1', 'w00t') $$,
  'Insert another user'
);

-SELECT bag_eq(
-  'SELECT * FROM users',
-  $$ VALUES
-    ('theory', md5('foo'), NOW()),
-    ('strongrr1', md5('w00t'), NOW())
-  $$,
-  '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 = 'strongrr1'
+  AND password = crypt('w00t', password)
+), 'The second user should have been inserted' );

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

```
@@ -56,14 +56,8 @@ SELECT throws_ok(  
    'Should get an error for duplicate nickname'  
);  
  
-SELECT bag_eq(  
-    'SELECT * FROM users',  
-    $$ VALUES  
-        ('theory',    md5('foo'),    NOW()),  
-        ('strongrr1', md5('w00t'), NOW())  
-    $$,  
-    'Should still have just the two users'  
-);  
+SELECT is(COUNT(*)::INT, 2, 'Should still have two users')  
+ FROM flipr.users;  
  
SELECT finish();  
ROLLBACK;  
>
```

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

```
-SELECT bag_eq(  
-    'SELECT * FROM users',  
-    $$ VALUES  
-        ('theory', md5('foo'), NOW()),  
-        ('strongrr1', md5('w00t'), NOW())  
-    $$,  
-    'Should still have just the two users'  
-);
```

```
+SELECT is(COUNT(*)::INT, 2, 'Should still have two users')  
+ FROM flipr.users;
```

```
SELECT finish();  
ROLLBACK;
```

```
>
```

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

```
-SELECT bag_eq(  
-    'SELECT * FROM users',  
-    $$ VALUES  
-        ('theory', md5('foo'), NOW()),  
-        ('strongrr1', md5('w00t'), NOW())  
-    $$,  
-    'Should still have just the two users'  
-);
```

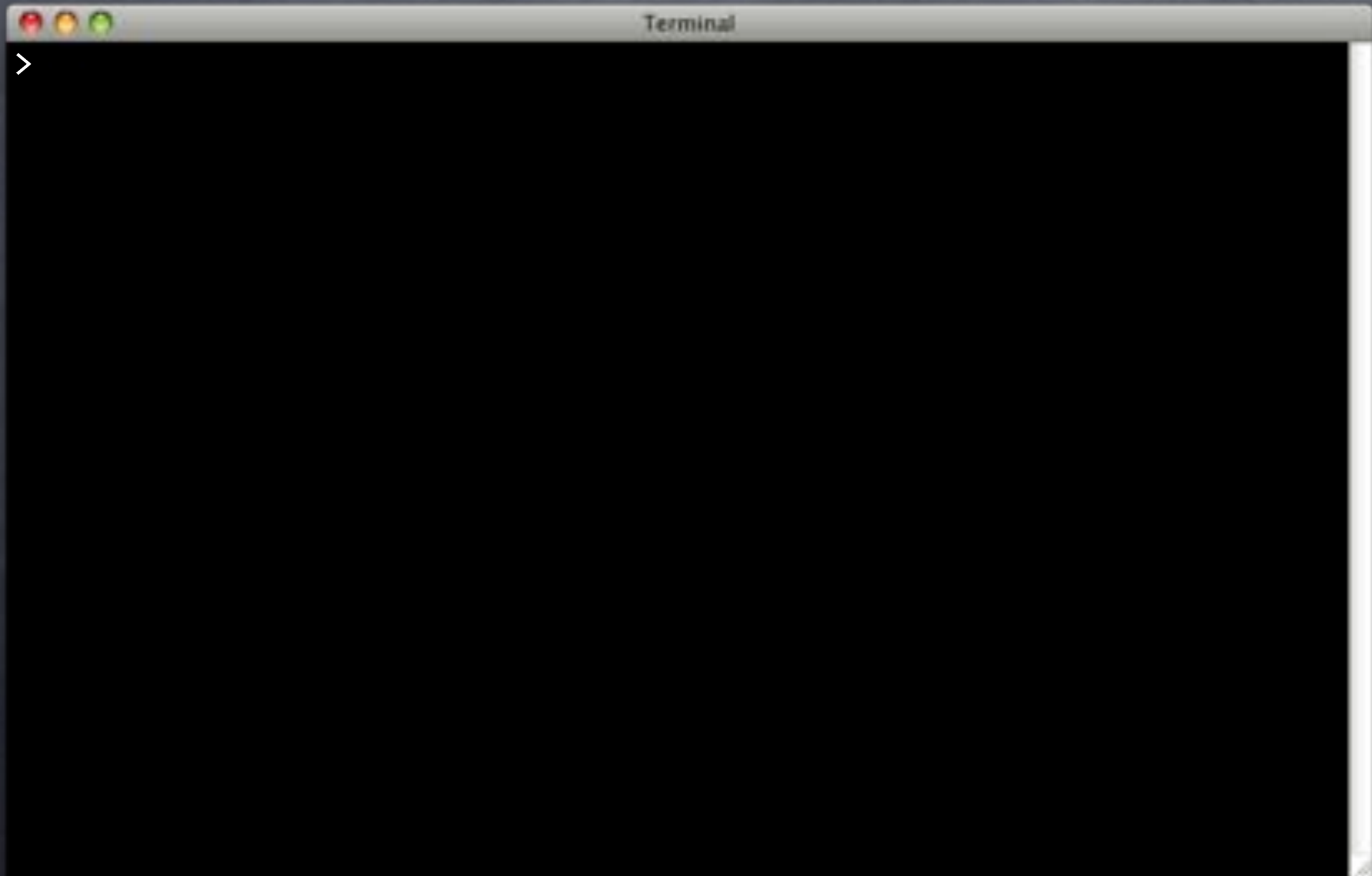
```
+SELECT is(COUNT(*)::INT, 2, 'Should still have two users')  
+ FROM flipr.users;
```

```
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.

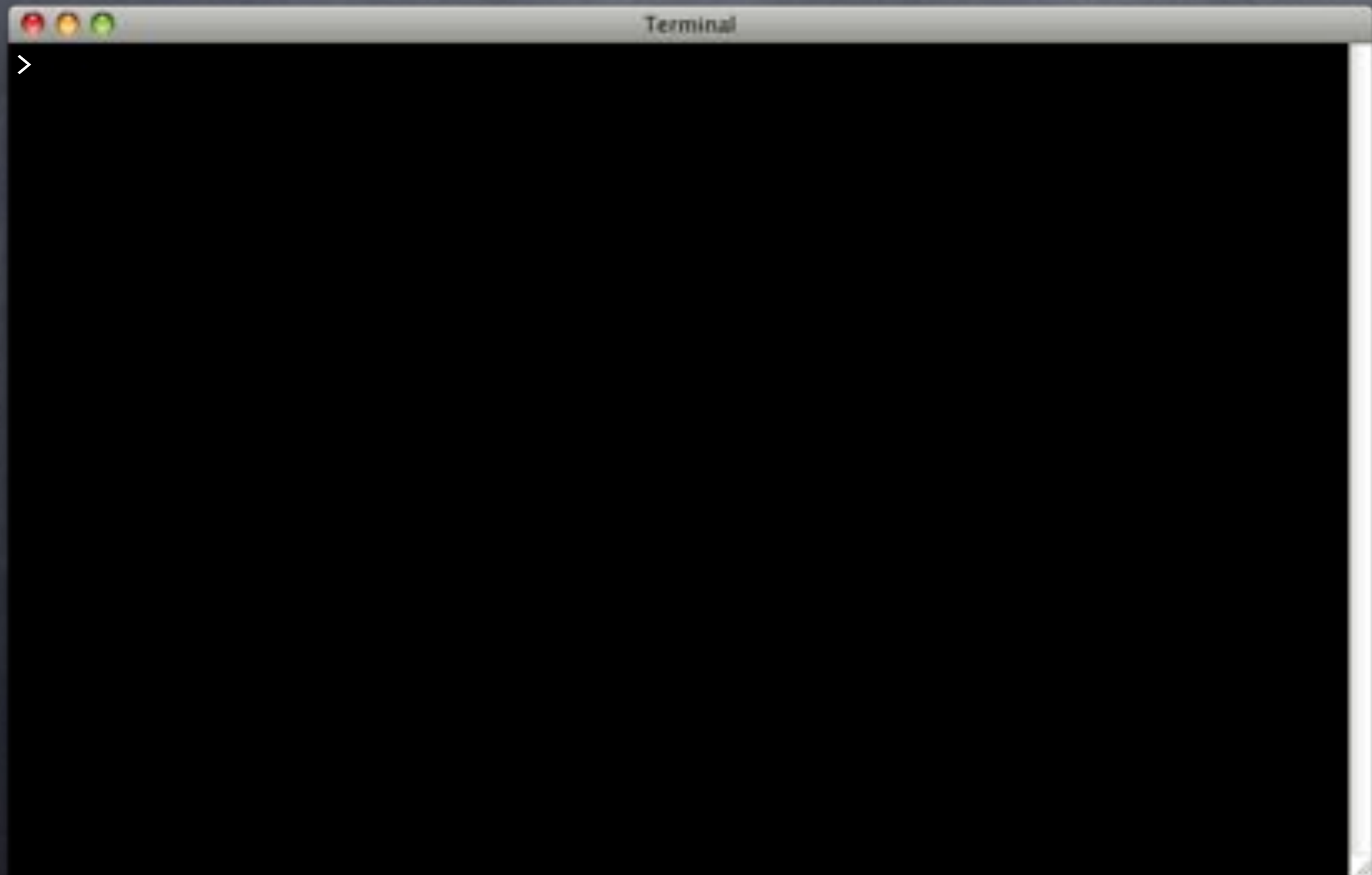
Back when discussing  
traditional migration  
systems, I said...



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

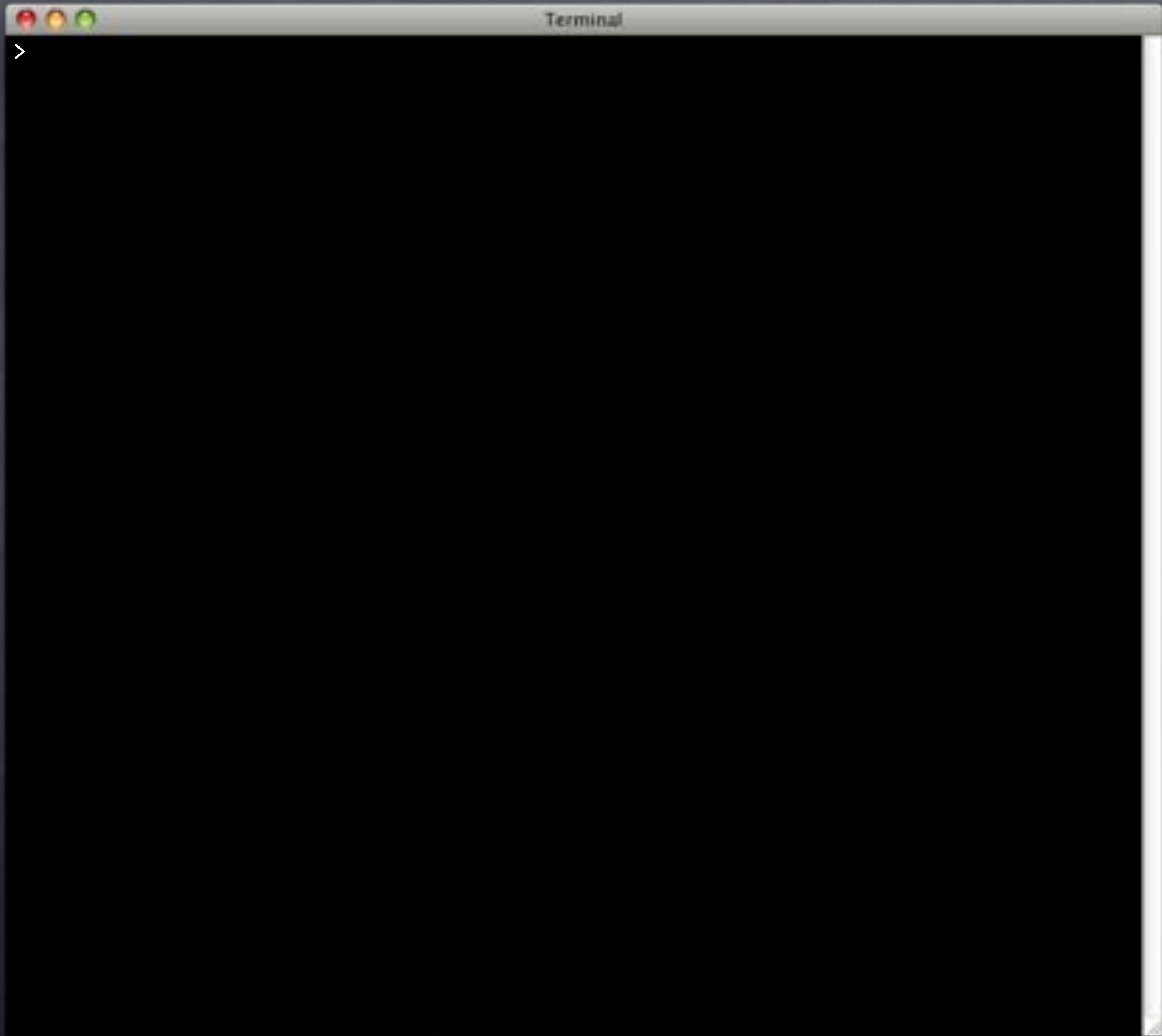
- Copy insert\_user.sql to new deploy file
- Change that new file
- Copy insert\_user.sql to new revert file
- Test it
- Do the same for change\_pass.sql



# 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
> 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 $$
+  INSERT INTO users values($1, md5($2));
+$$;
```

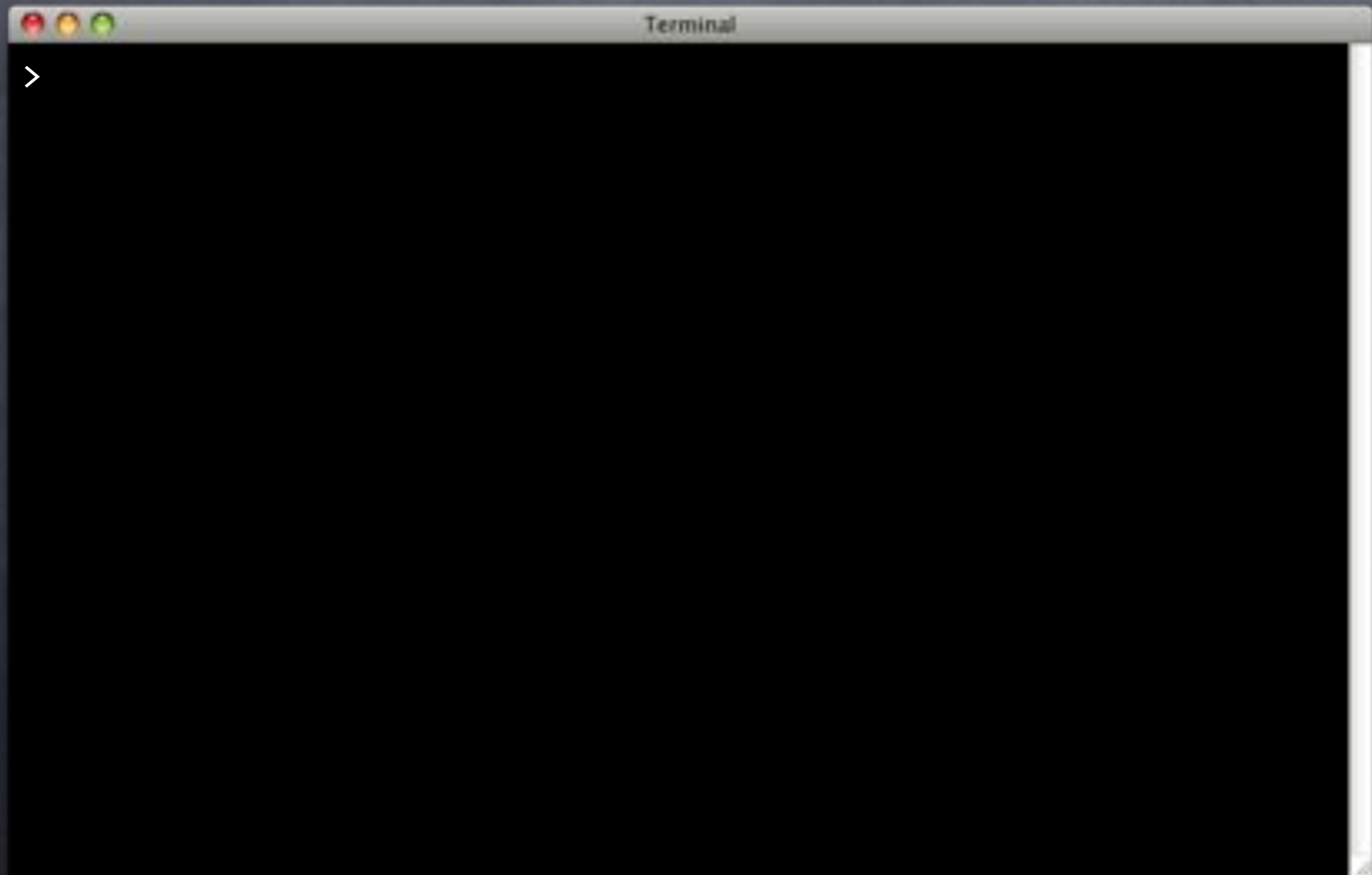
```
Terminal
> git diff HEAD^
diff --git a/deploy/insert_user_crypt.sql b/deploy/insert_user_crypto.sql
new file mode 100644
index 0000000..fa8d0c6
--- /dev/null
+++ b/deploy/insert_user_crypt.sql
@@ -0,0 +1,8 @@
+-- requires: users, appuser, pgcrypto
+
+CREATE OR REPLACE FUNCTION insert_user(
+  nickname 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
>
```

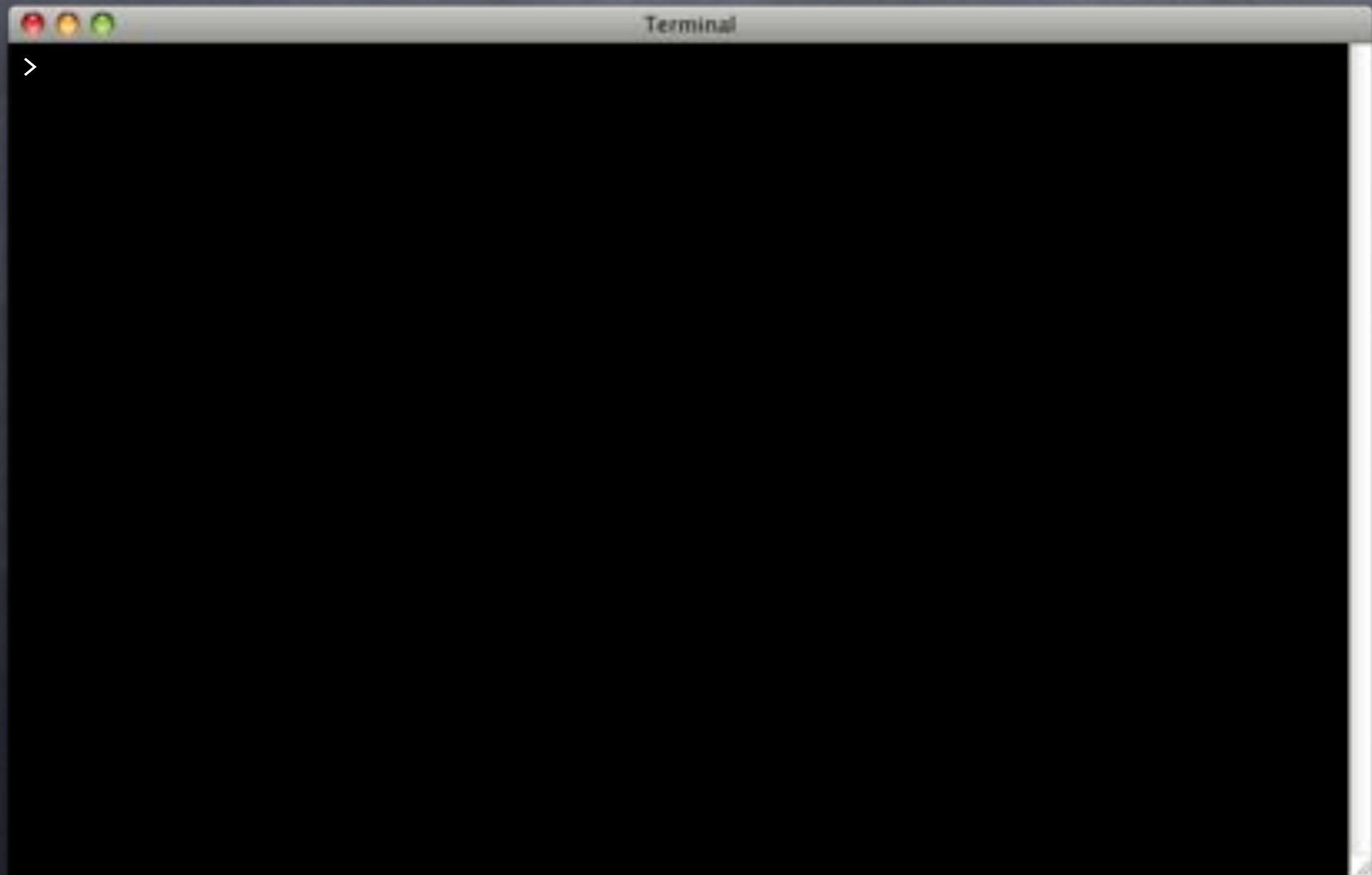
# Rework It

```
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 committed
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   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 committed)
#
#   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 -a")
>
```

# 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
>
```

As of  
@v1.0.0-r1

# Same Files?

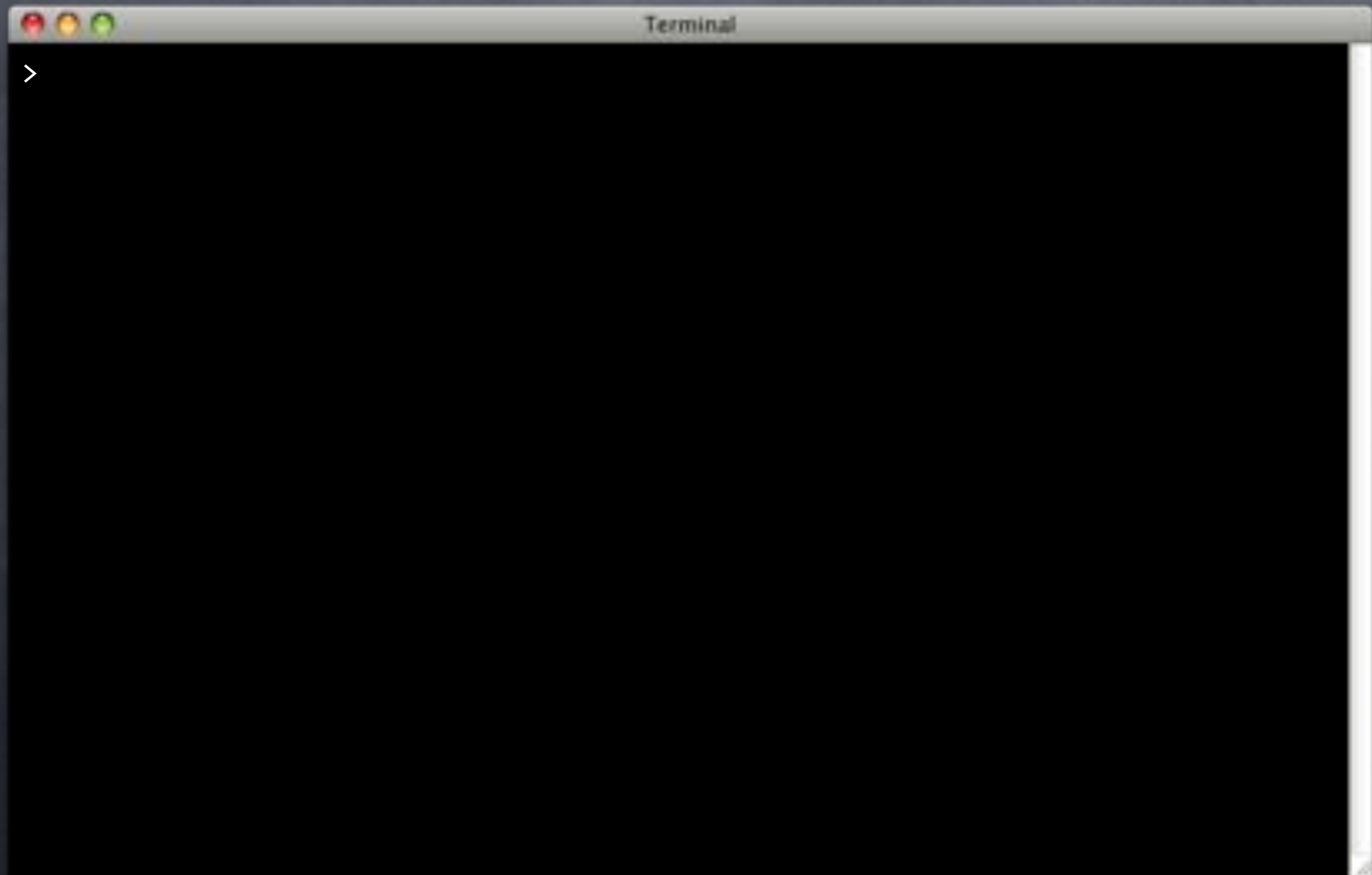
```
Terminal
> git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
# modified:   revert/insert_user.sql
# modified:   sqitch.plan
# modified:   test/insert_user.sql
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   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 -a")
>
```

# Same Files?

```
Terminal
> git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
# 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 committed)
#
#   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" to update)
>
```

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

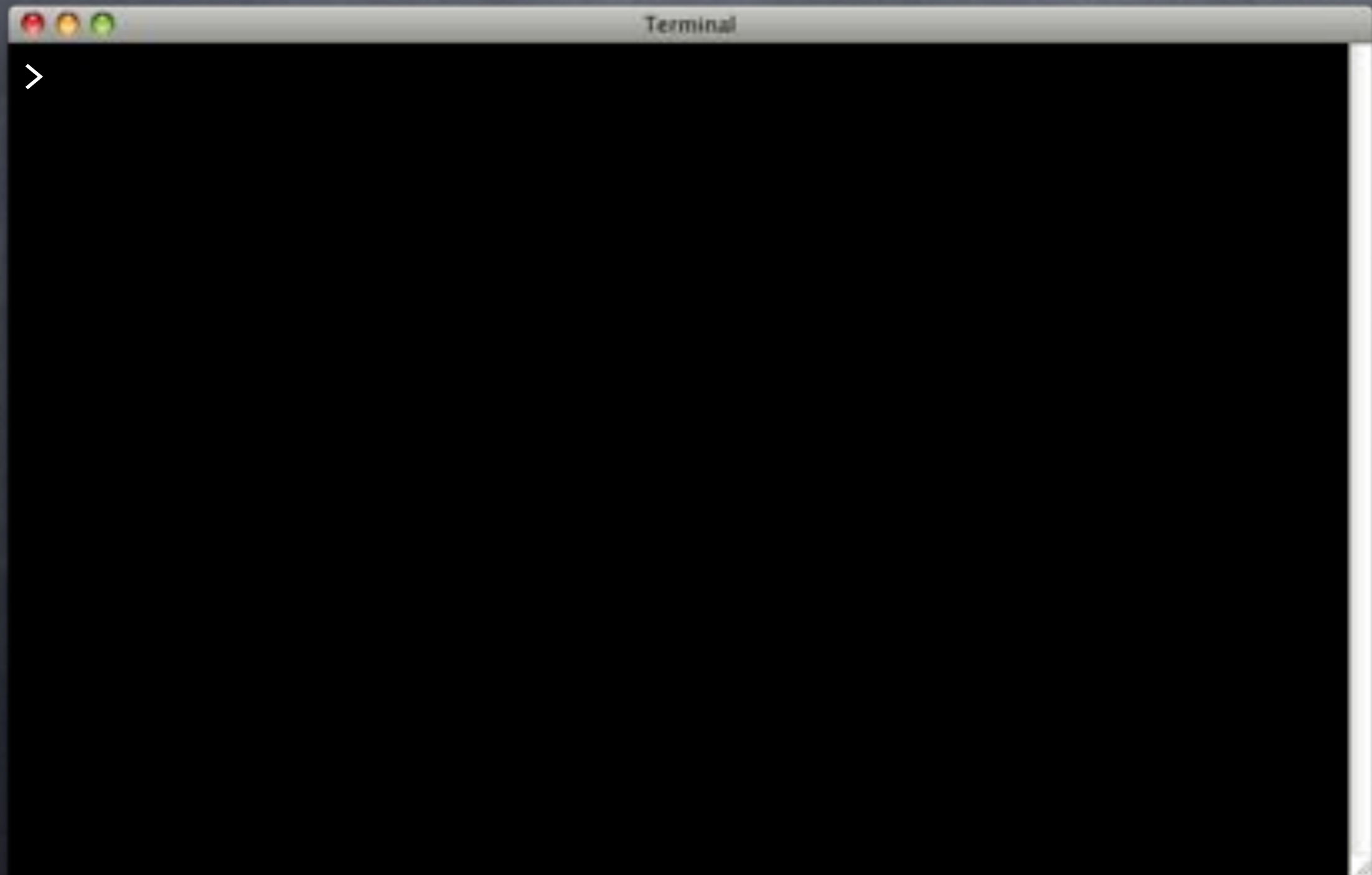
BEGIN;

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

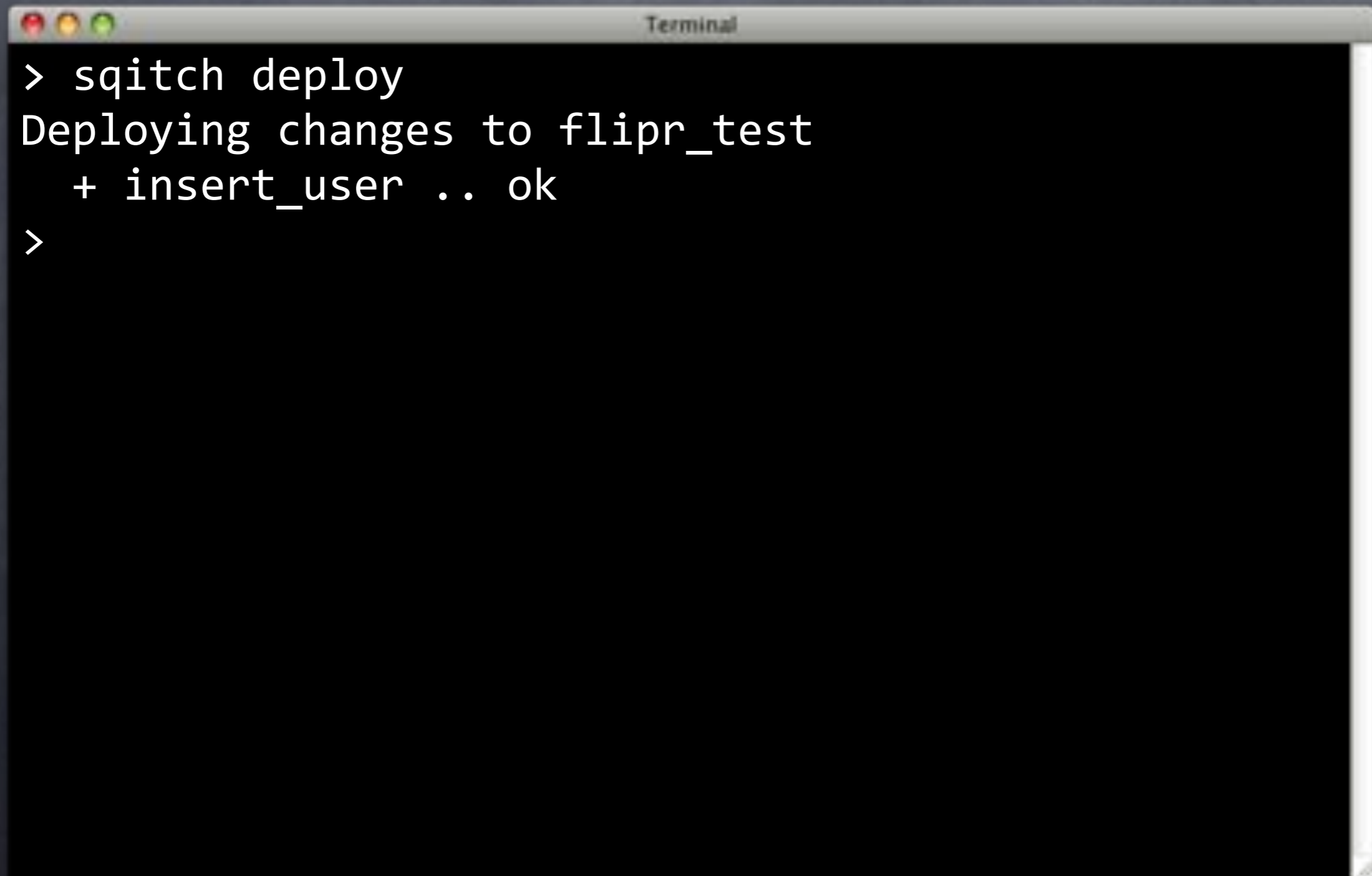
COMMIT;
>
```



# Send it Up!



# Send it Up!

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a command being executed and its output.

```
> 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      | $1$nK047p03$YRXYTt4NoNncTThLyxzEq1
bar      | $1$LbVUs/p.$LVbvPlkD8rJlixW2nS3WP0
(2 rows)
>
```

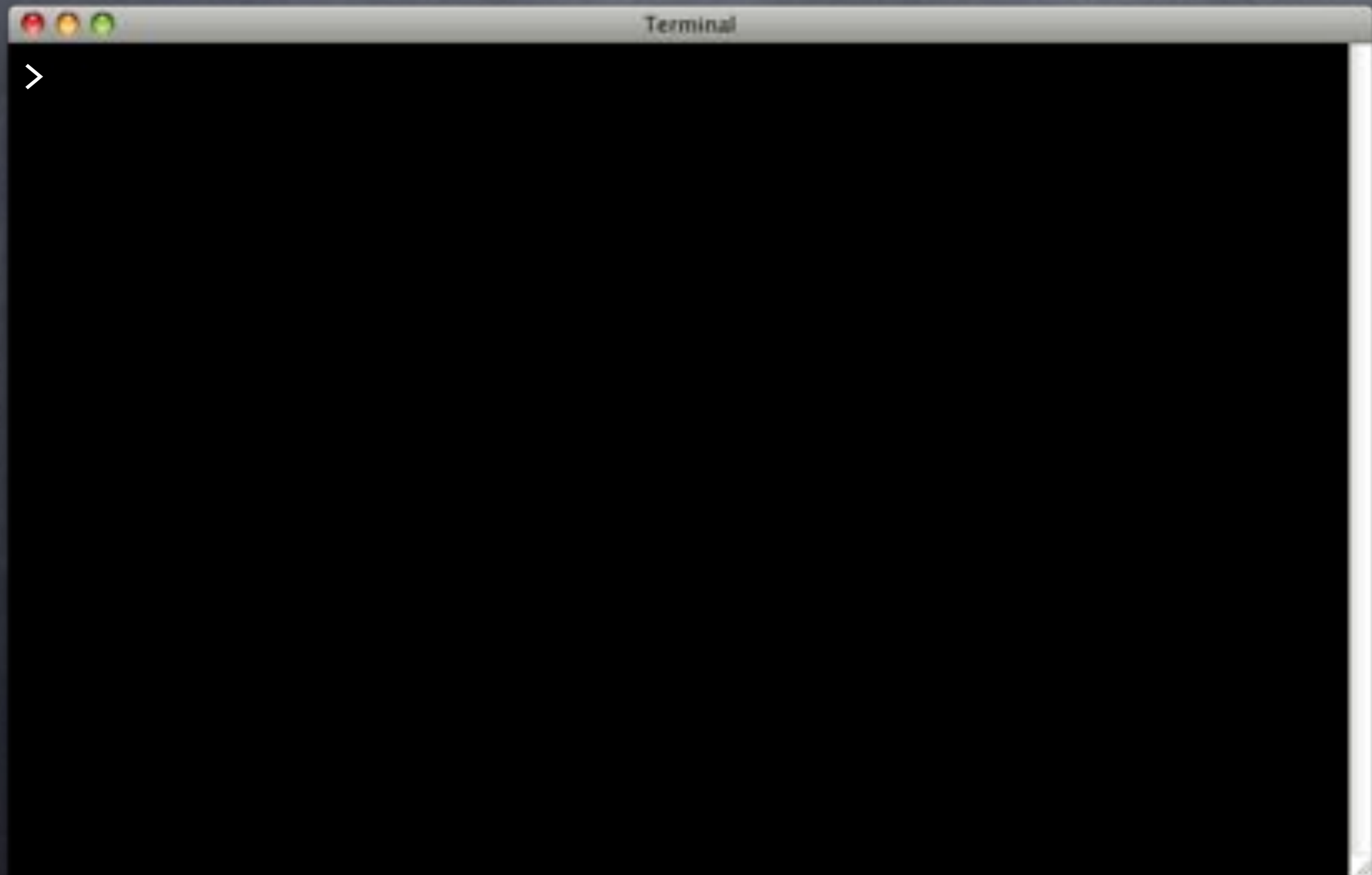
# Send it Up!

```
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      | $1$nK047p03$YRXYTt4NoNncTThLyxzEq1
bar      | $1$LbVUs/p.$LVbvPlkD8rJlixW2nS3WP0
(2 rows)
>
```

✓

\$1\$nK047p03\$YRXYTt4NoNncTThLyxzEq1  
\$1\$LbVUs/p.\$LVbvPlkD8rJlixW2nS3WP0

# Can We Go Back?



# Can We Go Back?

A terminal window with a title bar that says "Terminal". The window contains a command to revert a commit and insert a new commit.

```
> sqitch revert --to @HEAD^ -y  
- insert_user .. ok  
>
```

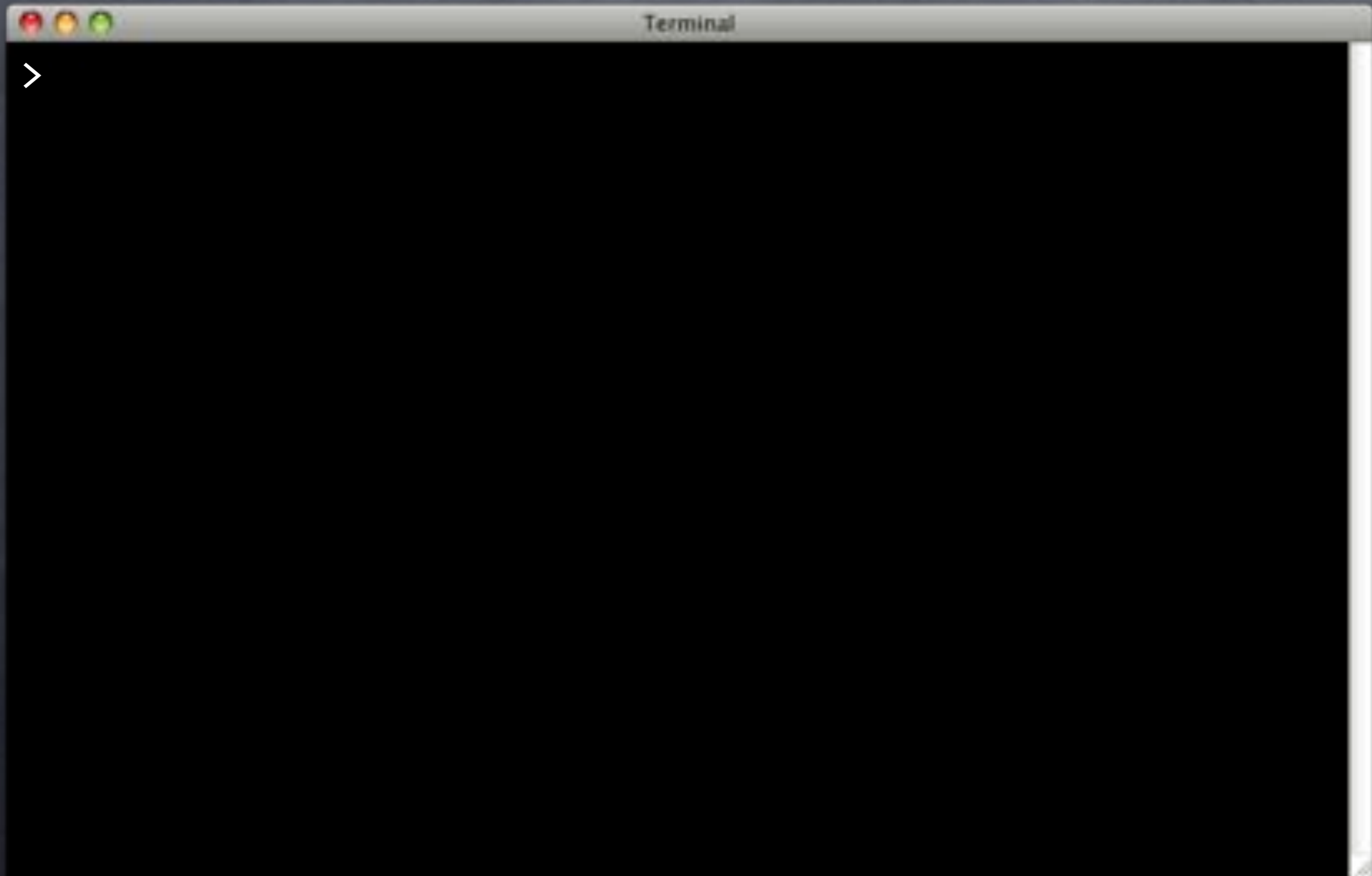
# Can We Go Back?

```
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

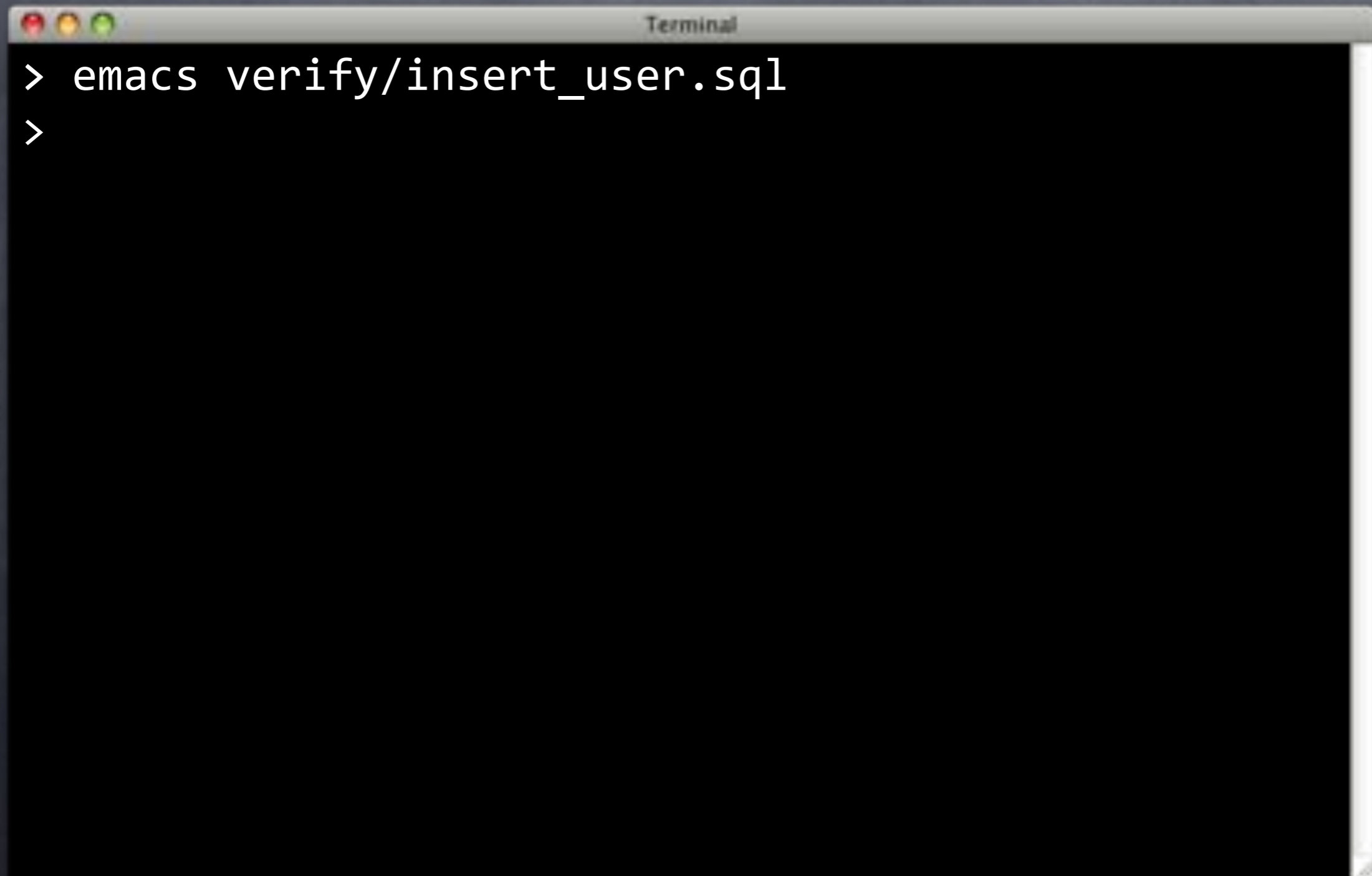
(2 rows)

# Verify How?



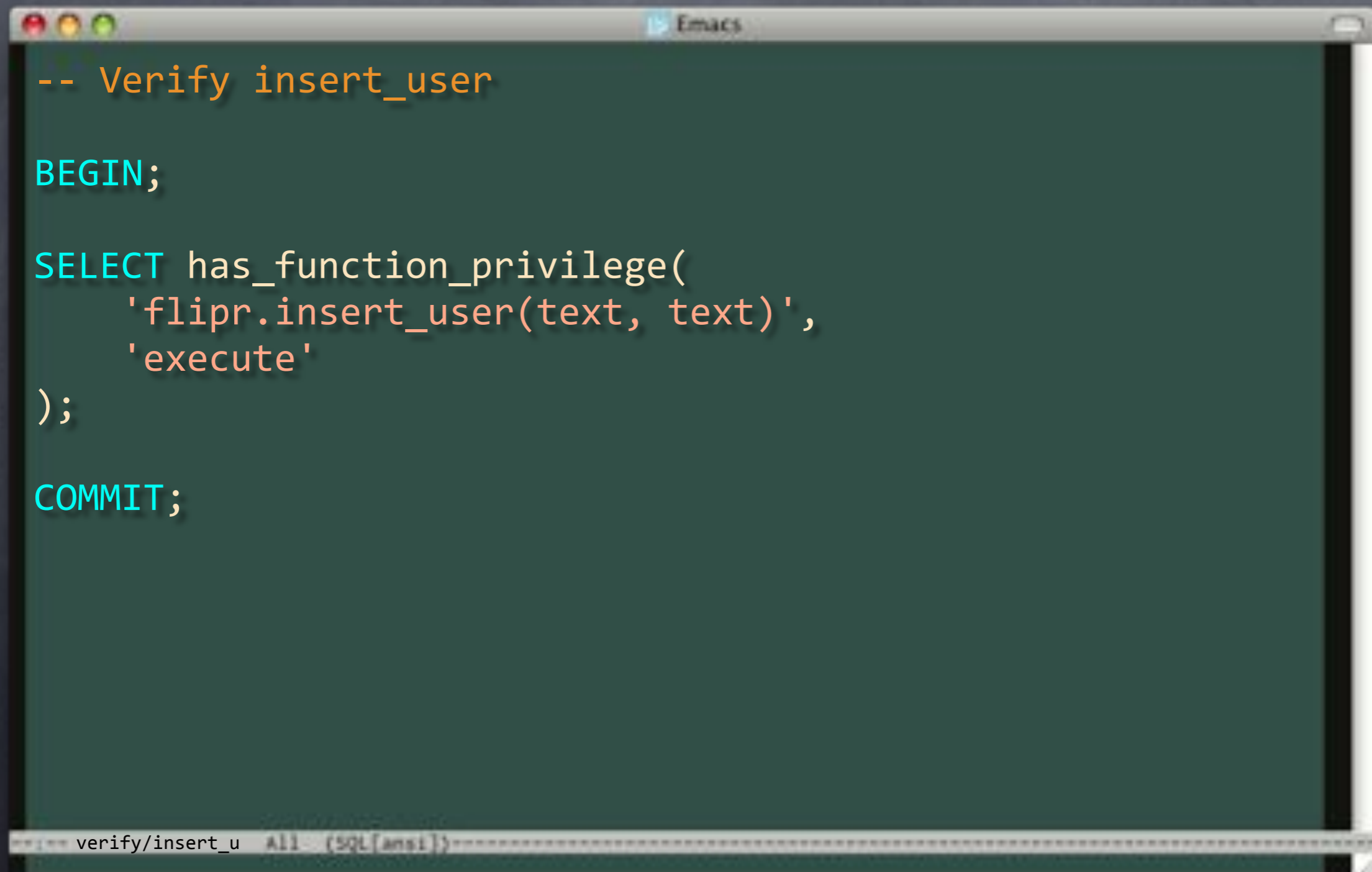


# Verify How?

A terminal window titled "Terminal" with a standard macOS-style title bar (red, yellow, green buttons). The terminal content shows a shell prompt followed by the command to open a file in emacs.

```
> emacs verify/insert_user.sql  
>
```

# verify/insert\_user.sql

The image shows a screenshot of an Emacs editor window. The window title is "Emacs". The background of the editor is dark green. The code is as follows:

```
-- Verify insert_user  
  
BEGIN;  
  
SELECT has_function_privilege(  
    'flipr.insert_user(text, text)',  
    'execute'  
);  
  
COMMIT;
```

The status bar at the bottom of the window shows "verify/insert\_u All (SQL[ansi])".

# verify/insert\_user.sql

```
-- 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

```
-- Verify insert_user

BEGIN;

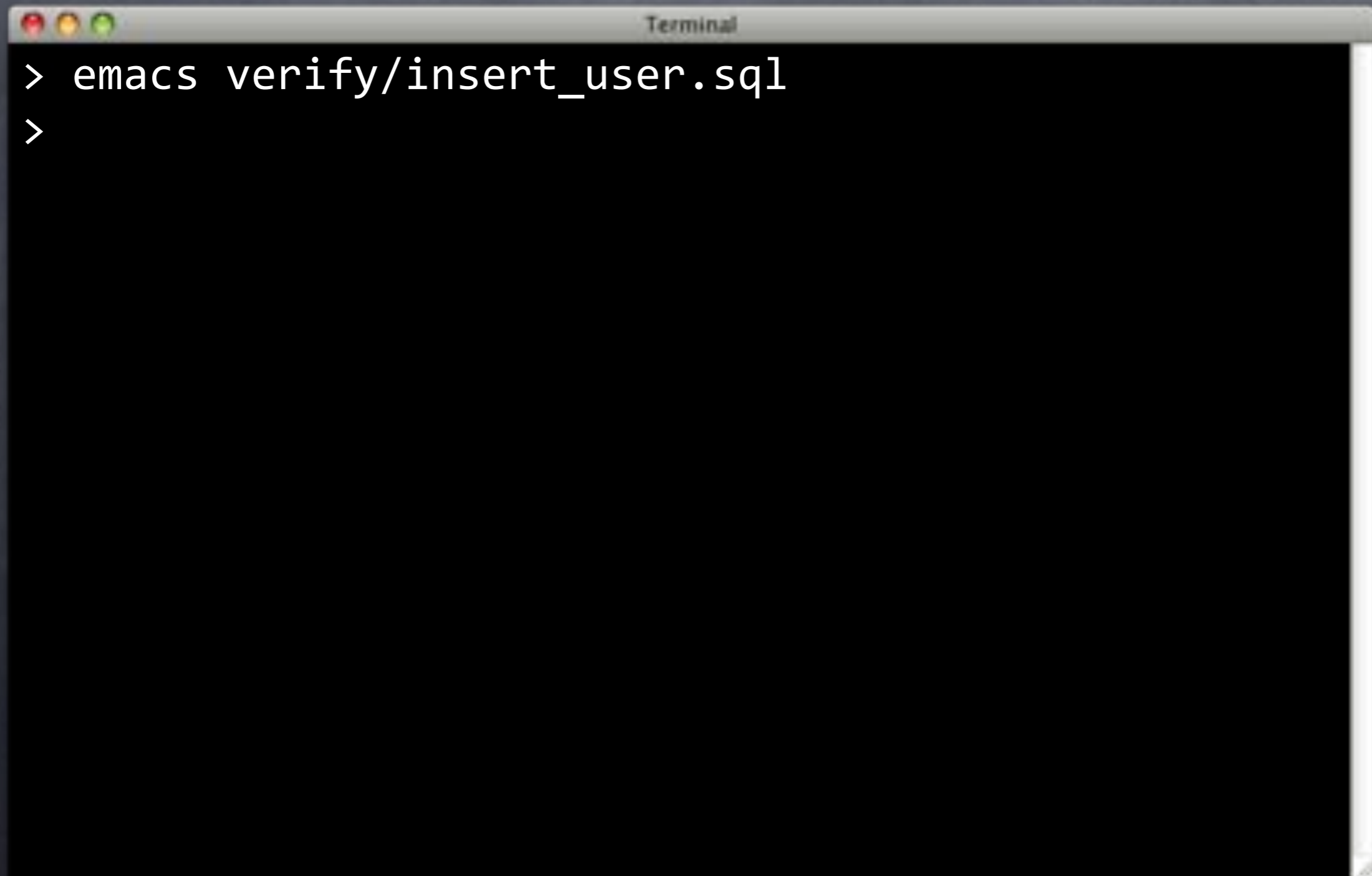
SELECT has_function_privilege(
    'flipr.insert_user(text, text)',
    'execute'
);

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

COMMIT;
```

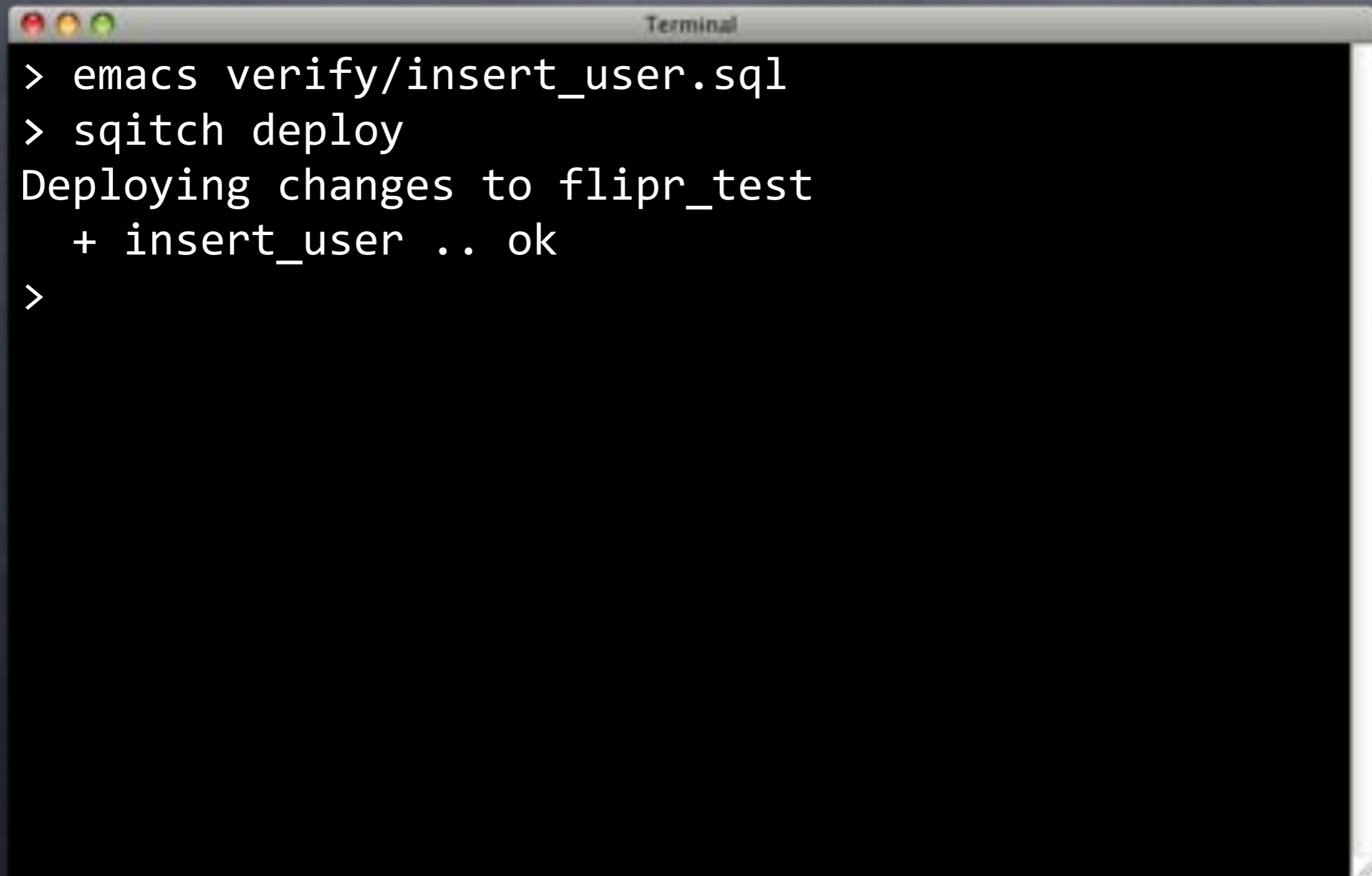
Yeah,  
compare  
source.

# Let's Go!

A terminal window with a title bar that says "Terminal". The window has three colored window control buttons (red, yellow, green) on the left. The terminal content shows a prompt character ">" followed by the command "emacs verify/insert\_user.sql" on the first line, and another prompt character ">" on the second line.

```
> emacs verify/insert_user.sql  
>
```

# Let's Go!

A terminal window titled "Terminal" with a standard macOS window header (red, yellow, green buttons). The terminal content shows a sequence of commands and their output: a prompt followed by "emacs verify/insert\_user.sql", another prompt followed by "sqitch deploy", the output "Deploying changes to flipr\_test", a sub-output "+ insert\_user .. ok", and a final prompt.

```
> 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
>
```

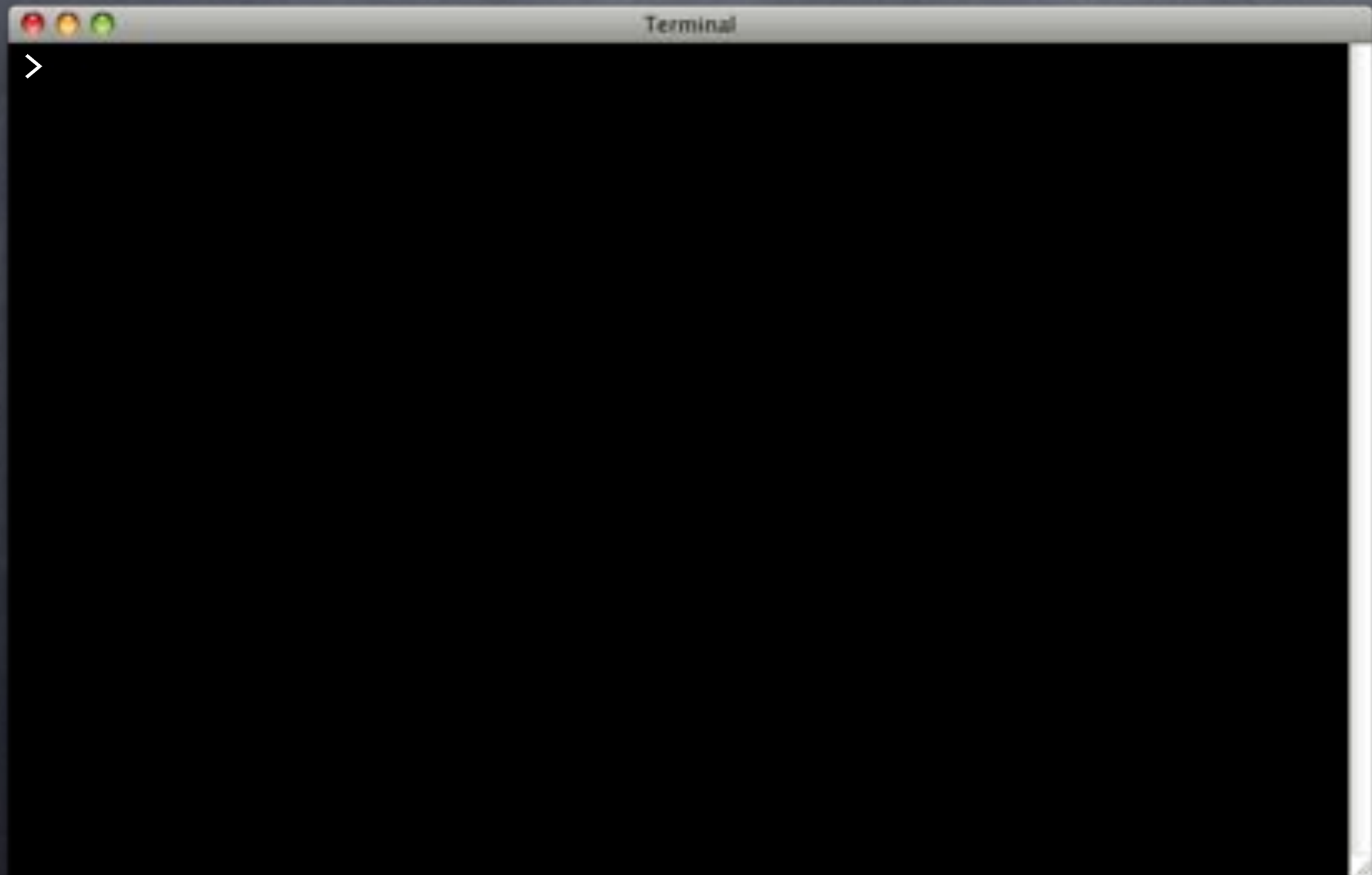
# Let's Go!

```
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 `pgcrypto`



# Change It Up

- `git reset --hard insert_user2`
- Rework `change_pass`
- Use `pgcrypto`
- Test first!





# Change It Up

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



# Change It Up

- `git reset --hard insert_user2`
- Rework `change_pass`
- Use `pgcrypto`
- 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



Text: Attribution-Noncommercial-Share Alike 3.0 United States:  
<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>  
Images licensed independently and © Their respective owners.

