

So, yeah, we're switching to Git.

So, yeah, we're switching to Git.

Why git?

So, yeah, we're switching to Git.

Why git?

Because many people already like git.

So, yeah, we're switching to Git.

Why git?

Because many people already like git.
Employees.

So, yeah, we're switching to Git.

Why git?

Because many people already like git.
Employees. Clients.

So, yeah, we're switching to Git.

Why git?

Because many people already like git.
Employees. Clients. Open source projects.

So, yeah, we're switching to Git.

Why git?

Because many people already like git.
Employees. Clients. Open source projects.

S1 doesn't code anymore, so he lets us use
whichever version-control system we want.
And some of us told him we want Git! :)

Git vs Svn

- * Git is new and popular

Git vs Svn

- * Git is new and popular (among developers and among our clients)

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)
- * Git is powerful

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)
- * Git is powerful (this slide system runs on git's history-rewriting magic)

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)
- * Git is powerful (this slide system runs on git's history-rewriting magic)

example:

```
void say_hello() {  
    printf("hello");  
}
```

```
int main() {  
    say_hello();  
    return 0;  
}
```

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)
- * Git is powerful (this slide system runs on git's history-rewriting magic)

example:

```
void say_hello() {  
    printf("hello");  
}  
  
int main() {  
    for(int i=0; i<10; ++i) {  
        say_hello();  
    }  
    return 0;  
}
```

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)
- * Git is powerful (this slide system runs on git's history-rewriting magic)

example:

```
void say_hello() {  
    printf("hello");  
}  
  
int main(int argc, char** argv) {  
    int n = (argc) > 1  
        ? atoi(argv[1])  
        : 10;  
    for(int i=0; i<n; ++i) {  
        say_hello();  
    }  
    return 0;  
}
```


Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)
- * Git is powerful (this slide system runs on git's history-rewriting magic)

but...

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)
- * Git is powerful (this slide system runs on git's history-rewriting magic)

but...

- * Git is new
- * Git is distributed
- * Git is powerful

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)
- * Git is powerful (this slide system runs on git's history-rewriting magic)

but...

- * Git is new (you will have to learn a new tool)
- * Git is distributed
- * Git is powerful

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)
- * Git is powerful (this slide system runs on git's history-rewriting magic)

but...

- * Git is new (you will have to learn a new tool)
- * Git is distributed (you will forget to push your changes)
- * Git is powerful

Git vs Svn

- * Git is new and popular (among developers and among our clients)
- * Git is distributed (but we don't care about that)
- * Git is powerful (this slide system runs on git's history-rewriting magic)

but...

- * Git is new (you will have to learn a new tool)
- * Git is distributed (you will forget to push your changes)
- * Git is powerful (and more complicated)

Git vs Svn

done pending

Git vs Svn

Don't panic

we're here to help.

Git vs Svn

gone gone

we're here to help.
your colleagues are here to help.

Git vs Svn

Google powered

we're here to help.
your colleagues are here to help.
Google is here to help.

Besides, it's not really that complicated:

- * `svn log` becomes `git log`
- * `svn add` becomes `git add`
- * `svn diff` becomes `git diff`
- * `svn mv` becomes `git mv`
- * `svn rm` becomes `git rm`
- * `svn status` becomes `git status`
- * `svn commit` becomes `git commit`

Besides, it's not really that complicated:

- * `svn log` becomes `git log`
- * `svn add` becomes `git add`
- * `svn diff` becomes `git diff`
- * `svn mv` becomes `git mv`
- * `svn rm` becomes `git rm`
- * `svn status` becomes `git status`
- * `svn commit` becomes `git add ; git commit ; git push`

```
git add ; git commit ; git push
```

Live example.

```
ls -A
```

A simple "hello world"

```
make
```

It works, but...

```
cat hello.c
```

Let's fix this silly mistake.

```
(edit example-repo/hello.c)
```

```
git add ; git commit ; git push
```

Live example.

```
ls -A  
.git  
hello.c  
Makefile
```

A simple "hello world"

```
make
```

It works, but...

```
cat hello.c
```

Let's fix this silly mistake.

```
(edit example-repo/hello.c)
```

```
git add ; git commit ; git push
```

Live example.

```
ls -A
```

A simple "hello world"

```
make
gcc hello.c -o hello
./hello
hello world!
```

It works, but...

```
cat hello.c
```

Let's fix this silly mistake.

```
(edit example-repo/hello.c)
```

```
git add ; git commit ; git push
```

Live example.

```
ls -A
```

A simple "hello world"

```
make
```

It works, but...

```
cat hello.c
int main() {
    printf("hello world!");
    return 0;
}
```

Let's fix this silly mistake.

```
(edit example-repo/hello.c)
```

```
git add ; git commit ; git push
```

Live example.

```
ls -A
```

A simple "hello world"

```
make
```

It works, but...

```
cat hello.c
int main() {
    printf("hello world!\n");
    return 0;
}
```

Let's fix this silly mistake.


```
git add ; git commit ; git push  
^
```

Status before the add:

```
git status
```

Step 1: git add

```
git add hello.c
```

Status after the add:

```
git status
```

```
git add ; git commit ; git push
^
```

Status before the add:

```
git status
# On branch master
# Changes not staged for commit:
#       modified:   hello.c
#
...
```

Step 1: git add

```
git add hello.c
```

Status after the add:

```
git status
```

```
git add ; git commit ; git push  
^
```

Status before the add:

```
git status
```

Step 1: git add

```
git add hello.c  
(no output)
```

Status after the add:

```
git status
```

```
git add ; git commit ; git push
^
```

Status before the add:

```
git status
```

Step 1: git add

```
git add hello.c
```

Status after the add:

```
git status
# On branch master
# Changes to be committed:
#   modified:   hello.c
#
...
```

```
git add ; git commit ; git push
          ^
```

Status before the commit:

```
git status
```

Step 2: git commit

```
git commit -m "missing newline"
```

Status after the commit:

```
git status
```

```
git add ; git commit ; git push
      ^
```

Status before the commit:

```
git status
# On branch master
# Changes to be committed:
#   modified:   hello.c
#
...
```

Step 2: git commit

```
git commit -m "missing newline"
```

Status after the commit:

```
git status
```

```
git add ; git commit ; git push
      ^
```

Status before the commit:

```
git status
```

Step 2: git commit

```
git commit -m "missing newline"
[master 6bcea2f] missing newline
1 files changed, 1 insertions(+), 1 deletions(-)
```

Status after the commit:

```
git status
```

```
git add ; git commit ; git push
          ^
```

Status before the commit:

```
git status
```

Step 2: git commit

```
git commit -m "missing newline"
```

Status after the commit:

```
git status
```

(no "Changes to be committed" section)


```
git add ; git commit ; git push
      ^
```

Our commits:

```
git log --oneline
```

Upstream commits:

```
git log --oneline origin/master
```

```
git add ; git commit ; git push
      ^
```

Our commits:

```
git log --oneline
53fd141 missing newline
21722a1 hello world, with a few intentional mistakes
4d430e1 no files yet
```

Upstream commits:

```
git log --oneline origin/master
```

```
git add ; git commit ; git push
      ^
```

Our commits:

```
git log --oneline
```

Upstream commits:

```
git log --oneline origin/master
21722a1 hello world, with a few intentional mistakes
4d430e1 no files yet
```

```
git add ; git commit ; git push
      ^
```

Status before the push:

```
git status
```

Step 3: git push

```
git push
```

Status after the push:

```
git status
```

```
git add ; git commit ; git push
      ^
```

Status before the push:

```
git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
...
```

Step 3: git push

```
git push
```

Status after the push:

```
git status
```

```
git add ; git commit ; git push
      ^
```

Status before the push:

```
git status
```

Step 3: git push

```
git push
To ../example-remote
21722a1..53fd141 master -> master
```

Status after the push:

```
git status
```

```
git add ; git commit ; git push
      ^
```

Status before the push:

```
git status
```

Step 3: git push

```
git push
```

Status after the push:

```
git status
```

(no "ahead by 1 commit" message)

Of course, things don't always go that smoothly.

What if your working copy is not up-to-date?


```
git add ; git commit ; git push
      ^
```

Status before the push:

```
git status
```

```
git log --oneline
```

First attempt: git push

```
git push
```

```
git add ; git commit ; git push
      ^
```

Status before the push:

```
git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
...
```

```
git log --oneline
```

First attempt: git push

```
git push
```

```
git add ; git commit ; git push
      ^
```

Status before the push:

```
git status
```

```
git log --oneline
53fd141 missing newline
21722a1 hello world, with a few intentional mistakes
4d430e1 no files yet
```

First attempt: git push

```
git push
```

```
git add ; git commit ; git push
      ^
```

Status before the push:

```
git status
```

```
git log --oneline
```

First attempt: git push

```
git push
```

To ../example-remote

```
! [rejected]          master -> master (non-fast-forward)
```

```
...
```

```
git add ; git commit ; git^pull ; git push
```

Our commits:

```
git log --oneline
```

Merge the upstream changes ("rebase" is a mode that is closer to svn's way):

```
git pull --rebase
```

The updated commits:

```
git log --oneline
```

```
git add ; git commit ; git pull ; git push
```

Our commits:

```
git log --oneline
53fd141 missing newline
21722a1 hello world, with a few intentional mistakes
4d430e1 no files yet
```

Merge the upstream changes ("rebase" is a mode that is closer to svn's way):

```
git pull --rebase
```

The updated commits:

```
git log --oneline
```

```
git add ; git commit ; git pull ; git push
      ^
```

Our commits:

```
git log --oneline
```

Merge the upstream changes ("rebase" is a mode that is closer to svn's way):

```
git pull --rebase
From ../example-remote
 21722a1..6bb51ae master    -> origin/master
...
Auto-merging hello.c
```

The updated commits:

```
git log --oneline
```

```
git add ; git commit ; git^pull ; git push
```

Our commits:

```
git log --oneline
```

Merge the upstream changes ("rebase" is a mode that is closer to svn's way):

```
git pull --rebase
```

The updated commits:

```
git log --oneline
2d927f8 missing newline
6bb51ae add a comment
21722a1 hello world, with a few intentional mistakes
4d430e1 no files yet
```



```
git add ; git commit ; git pull ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

```
git add ; git commit ; git pull ; git^push
```

Status before the push:

```
git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
...
```

```
git log --oneline
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

```
git add ; git commit ; git pull ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
c805c01 missing newline
6bb51ae add a comment
21722a1 hello world, with a few intentional mistakes
4d430e1 no files yet
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

```
git add ; git commit ; git pull ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
```

```
git log --oneline origin/master
```

```
6bb51ae add a comment
```

```
21722a1 hello world, with a few intentional mistakes
```

```
4d430e1 no files yet
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

```
git add ; git commit ; git pull ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

To ../example-remote

```
6bb51ae..c805c01 master -> master
```

Status after the push:

```
git status
```

```
git add ; git commit ; git pull ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

(no "ahead by 1 commit" message)

Of course, things don't always go that smoothly.

What if your commit conflicts with someone else's?

```
git add ; git commit ; git pull ; git push
```

Our commits:

```
git log --oneline
```

Merge the upstream changes:

```
git pull --rebase
```



```
git add ; git commit ; git pull ; git push
```

Our commits:

```
git log --oneline
53fd141 missing newline
21722a1 hello world, with a few intentional mistakes
4d430e1 no files yet
```

Merge the upstream changes:

```
git pull --rebase
```

```
git add ; git commit ; git pull ; git push
```

Our commits:

```
git log --oneline
```

Merge the upstream changes:

```
git pull --rebase
From ../example-remote
21722a1..a506fad master -> origin/master
...
Auto-merging hello.c
CONFLICT (content): Merge conflict in hello.c
```

When you have resolved this problem run "git rebase --continue".

```
git add ; git commit ; git pull ; resolve ; git push
      ^
```

Status before resolving:

```
git status
```

Conflict markers:

```
cat hello.c
```

Mark as resolved:

```
git add hello.c
```

```
git add ; git commit ; git pull ; resolve ; git push
      ^
```

Status before resolving:

```
git status
# Not currently on any branch.
# Unmerged paths:
#   both modified:   hello.c
...
```

Conflict markers:

```
cat hello.c
```

Mark as resolved:

```
git add hello.c
```

```
git add ; git commit ; git pull ; resolve ; git push
      ^
```

Status before resolving:

```
git status
```

Conflict markers:

```
cat hello.c
int main() {
<<<<<< HEAD
    printf("HELLO WORLD!");
||||||| merged common ancestors
    printf("hello world!");
=====
    printf("hello world!\n");
>>>>>> missing newline
    return 0;
}
```

Mark as resolved:

```
git add hello.c
```

```
git add ; git commit ; git pull ; resolve ; git push
      ^
```

Status before resolving:

```
git status
```

Conflict markers:

```
cat hello.c
```

Mark as resolved:

```
git add hello.c
(no output)
```

```
git add ; git commit ; git pull ; resolve ; git push
      ^
```

Status before continuing:

```
git status
```

```
git log --oneline
```

The command we were told to type after resolving:

```
git rebase --continue
```

Status after continuing:

```
git status
```

```
git log --oneline
```

```
git add ; git commit ; git pull ; resolve ; git push
      ^
```

Status before continuing:

```
git status
# Not currently on any branch.
# Changes to be committed:
#   modified:   hello.c
...
```

```
git log --oneline
```

The command we were told to type after resolving:

```
git rebase --continue
```

Status after continuing:

```
git status
```

```
git log --oneline
```



```
git add ; git commit ; git pull ; resolve ; git push
      ^
```

Status before continuing:

```
git status
```

```
git log --oneline
a506fad ALL CAPS!!
21722a1 hello world, with a few intentional mistakes
4d430e1 no files yet
```

The command we were told to type after resolving:

```
git rebase --continue
```

Status after continuing:

```
git status
```

```
git log --oneline
```

```
git add ; git commit ; git pull ; resolve ; git push
      ^
```

Status before continuing:

```
git status
```

```
git log --oneline
```

The command we were told to type after resolving:

```
git rebase --continue
```

Applying: missing newline

Status after continuing:

```
git status
```

```
git log --oneline
```

```
git add ; git commit ; git pull ; resolve ; git push
      ^
```

Status before continuing:

```
git status
```

```
git log --oneline
```

The command we were told to type after resolving:

```
git rebase --continue
```

Status after continuing:

```
git status
```

```
# On branch master
```

```
# Your branch is ahead of 'origin/master' by 1 commit.
```

```
...
```

```
git log --oneline
```

```
git add ; git commit ; git pull ; resolve ; git push
      ^
```

Status before continuing:

```
git status
```

```
git log --oneline
```

The command we were told to type after resolving:

```
git rebase --continue
```

Status after continuing:

```
git status
```

```
git log --oneline
```

```
94fcd09 missing newline
```

```
a506fad ALL CAPS!!
```

```
21722a1 hello world, with a few intentional mistakes
```

```
4d430e1 no files yet
```

```
git add ; git commit ; git pull ; resolve ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

```
git log --oneline origin/master
```

```
git add ; git commit ; git pull ; resolve ; git^push
```

Status before the push:

```
git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
...
```

```
git log --oneline
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

```
git log --oneline origin/master
```

```
git add ; git commit ; git pull ; resolve ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
0e6fcb3 missing newline
a506fad ALL CAPS!!
21722a1 hello world, with a few intentional mistakes
4d430e1 no files yet
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

```
git log --oneline origin/master
```

```
git add ; git commit ; git pull ; resolve ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
```

```
git log --oneline origin/master
```

```
a506fad ALL CAPS!!
```

```
21722a1 hello world, with a few intentional mistakes
```

```
4d430e1 no files yet
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

```
git log --oneline origin/master
```



```
git add ; git commit ; git pull ; resolve ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

To ../example-remote

```
a506fad..0e6fcb3 master -> master
```

Status after the push:

```
git status
```

```
git log --oneline origin/master
```

```
git add ; git commit ; git pull ; resolve ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

(no "ahead by 1 commit" message)

```
git log --oneline origin/master
```

```
git add ; git commit ; git pull ; resolve ; git^push
```

Status before the push:

```
git status
```

```
git log --oneline
```

```
git log --oneline origin/master
```

Second attempt: git push

```
git push
```

Status after the push:

```
git status
```

```
git log --oneline origin/master
```

```
0e6fcb3 missing newline
```

```
a506fad ALL CAPS!!
```

```
21722a1 hello world, with a few intentional mistakes
```

```
4d430e1 no files yet
```

/ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$	/ \$ \$	/ \$ \$	/ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$
_____ \$ \$ _____ /	\$ \$	\$ \$	\$ \$ _____ /
\$ \$	\$ \$	\$ \$	\$ \$
\$ \$	\$ \$ \$ \$ \$ \$ \$ \$ \$ \$	\$ \$	\$ \$ \$ \$ \$
\$ \$	\$ \$ _____	\$ \$	\$ \$ _____ /
\$ \$	\$ \$	\$ \$	\$ \$
\$ \$	\$ \$	\$ \$	\$ \$ \$ \$ \$ \$ \$ \$ \$ \$
_____ /	_____ /	_____ /	_____ /

/ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$	/ \$ \$	/ \$ \$	/ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$
\$ \$ _____ /	\$ \$ \$	\$ \$	\$ \$ _____ \$ \$
\$ \$	\$ \$ \$ \$	\$ \$	\$ \$ \ \$ \$
\$ \$ \$ \$ \$	\$ \$ \$ \$	\$ \$	\$ \$ \$ \$
\$ \$ _____ /	\$ \$ \$ \$ \$ \$	\$ \$	\$ \$ \$ \$
\$ \$	\$ \$ \ \$ \$ \$	\$ \$	\$ \$ \$ \$
\$ \$ \$ \$ \$ \$ \$ \$ \$ \$	\$ \$ \ \$ \$	\$ \$	\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ /
_____ /	_____ / \ _____ /	_____ /	_____ /