Git the basics

Bart Trojanowski, bart@jukie.net

Jukie Networks Inc.

July 9th, 2008
Concepts
  SCM components
  SCM operations
  Decentralization

GIT History

Repository
  Structure
  Objects

Using GIT
  Commands
  Commit
  Inspection
  Branching
  Merging
  Rebasin
  Remotes
  GUI tools
  Loose ends

Git the basics
  Bart
  Trojanowski, bart@jukie.net
  Concepts
    SCM components
    SCM operations
    Decentralization
  GIT History
  Repository
    Structure
    Objects
  Using GIT
    Commands
    Commit
    Inspection
    Branching
    Merging
    Rebasin
    Remotes
    GUI tools
    Loose ends
-
Next...

Concepts

SCM components
SCM operations
Decentralization

GIT History

Repository

Structure
Objects

Using GIT

Commands
Commit
Inspection
Branching
Merging
Rebasings
Remotes
GUI tools
Loose ends
Concepts

- Source Control Management
  - track changes to files
  - repository / database of changes
  - working directory / current state

- Centralized SCM
  - server: single database
  - client: working directory & state

- Decentralized SCM
  - anyone can be a server
  - repository coupled with working directory
  - complete history
  - disconnected operation
Concepts

- Source Control Management
  - track changes to files
  - repository / database of changes
  - working directory / current state

- Centralized SCM
  - server: single database
  - client: working directory & state

- Decentralized SCM
  - anyone can be a server
  - repository coupled with working directory
  - complete history
  - disconnected operation
Concepts

- Source Control Management
  - track changes to files
  - repository / database of changes
  - working directory / current state

- Centralized SCM
  - server: single database
  - client: working directory & state

- Decentralized SCM
  - anyone can be a server
  - repository coupled with working directory
  - complete history
  - disconnected operation
SCM components

Working tree

- directories
- files
SCM components

Repository contents

► files
SCM components

Repository contents

- files
- commits
SCM components

Repository contents

- files
- commits
- ancestry
SCM components
directed acyclic graph

"DAG"
SCM components

References

► tags
SCM components

References

- tags
- branches

Git the basics
Bart
Trojanowski,
bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History
Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
SCM components

HEAD

- current checkout
- points to branch
SCM components

HEAD
- current checkout
- points to branch
- sometimes detached
SCM components

Index

► “staging area”
► what is to be committed
SCM operations

Bootstrap
  ▶ init
  ▶ checkout
  ▶ switch branch

Modify
  ▶ add, delete, rename
  ▶ commit

Information
  ▶ status
  ▶ diff
  ▶ log

Reference
  ▶ tag
  ▶ branch
SCM operations

Bootstrap
- init
- checkout
- switch branch

Modify
- add, delete, rename
- commit

Information
- status
- diff
- log

Reference
- tag
- branch
SCM operations

Bootstrap
  ▶ init
  ▶ checkout
  ▶ switch branch
Modify
  ▶ add, delete, rename
  ▶ commit
Information
  ▶ status
  ▶ diff
  ▶ log
Reference
  ▶ tag
  ▶ branch
SCM operations

Bootstrap
  ▶ init
  ▶ checkout
  ▶ switch branch

Modify
  ▶ add, delete, rename
  ▶ commit

Information
  ▶ status
  ▶ diff
  ▶ log

Reference
  ▶ tag
  ▶ branch
Centralized SCM

- operations require server
  - single point of failure
  - bottleneck
more SCM operations

Decentralized

- clone
- pull, fetch
- push
anyone can be a server
Decentralization

- public repository
Decentralization

▶ make a local clone
Decentralization

- local cloning is lightweight
Decentralization

- push changes between any repositories
Decentralization

- publish changes to public server
Decentralization

- share changes with trusted peers
Is Decentralization any good?

- non-intrusive micro-commits
- detached operation
- no single point of failure
- backups are trivial
Next...

- Concepts
  - SCM components
  - SCM operations
  - Decentralization

GIT History

- Repository
  - Structure
  - Objects

Using GIT

- Commands
- Commit
- Inspection
- Branching
- Merging
- Rebasings
- Remotes
- GUI tools
- Loose ends
Birth of GIT

- 2002
  - Linus uses BitKeeper for tracking Linux
  - BK gets better
  - Linux development scales better
- April 6, 2005
  - BitMover drops free license
  - Linus writes his own SCM, GIT
- April 18, 2005
  - GIT can merge
- June 16, 2005
  - GIT is officially used to track Linux
- Feb 14, 2007
  - GIT 1.5.0 is released
  - major usability effort
Birth of GIT

- 2002
  - Linus uses BitKeeper for tracking Linux
  - BK gets better
  - Linux development scales better

- April 6, 2005
  - BitMover drops free license
  - Linus writes his own SCM, GIT

- April 18, 2005
  - GIT can merge

- June 16, 2005
  - GIT is officially used to track Linux

- Feb 14, 2007
  - GIT 1.5.0 is released
  - major usability effort
Birth of GIT

- 2002
  - Linus uses BitKeeper for tracking Linux
  - BK gets better
  - Linux development scales better
- April 6, 2005
  - BitMover drops free license
  - Linus writes his own SCM, GIT
- April 18, 2005
  - GIT can merge
- June 16, 2005
  - GIT is officially used to track Linux
- Feb 14, 2007
  - GIT 1.5.0 is released
  - major usability effort
Birth of GIT

- 2002
  - Linus uses BitKeeper for tracking Linux
  - BK gets better
  - Linux development scales better
- April 6, 2005
  - BitMover drops free license
  - Linus writes his own SCM, GIT
- April 18, 2005
  - GIT can merge
- June 16, 2005
  - GIT is officially used to track Linux
- Feb 14, 2007
  - GIT 1.5.0 is released
  - major usability effort
Birth of GIT

- **2002**
  - Linus uses BitKeeper for tracking Linux
  - BK gets better
  - Linux development scales better
- **April 6, 2005**
  - BitMover drops free license
  - Linus writes his own SCM, GIT
- **April 18, 2005**
  - GIT can merge
- **June 16, 2005**
  - GIT is officially used to track Linux
- **Feb 14, 2007**
  - GIT 1.5.0 is released
  - major usability effort
GIT gets better

And then realize that nothing is perfect. Git is just *closer* to perfect than any other SCM out there.

-Linus
Next...

Concepts
- SCM components
- SCM operations
- Decentralization

GIT History

Repository
- Structure
- Objects

Using GIT
- Commands
- Commit
- Inspection
- Branching
- Merging
- Rebasings
- Remotes
- GUI tools
- Loose ends
Structure

repository

index

staging area

<table>
<thead>
<tr>
<th>Commit ID</th>
<th>Type</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>100644</td>
<td>file</td>
<td>bar</td>
</tr>
<tr>
<td>100644</td>
<td>file</td>
<td>baz</td>
</tr>
<tr>
<td>100644</td>
<td>file</td>
<td>sub/fi</td>
</tr>
<tr>
<td>100644</td>
<td>file</td>
<td>sub/foo</td>
</tr>
</tbody>
</table>

Bart Trojanowski, bart@jukie.net
Structure

repository

index

work tree

files you edit

100644 20b024 0 bar
100644 1d52a6 0 baz
100644 20b024 0 sub/fi
100644 43dbe0 0 sub/foo

|-- bar
|  |-- baz
|  |  |-- sub
|  |     |-- fi
|  |     |-- foo
“staging”

- add
- remove
- rename
**Structure**

**repository**

**index**

```
100644 20b024 0   bar
100644 1d52a6 0   baz
100644 20b024 0   sub/fi
100644 43dbe0 0   sub/foo
```

**work tree**

```
|−− bar
|   |−− baz
|   |−− sub
|   |   |−− fi
|   |   |−− foo
```

“committing”

**commit**
Structure

```
|−− bar
| −− baz
| −− sub
    | −− fi
    | −− foo
```

```
100644 20b024 0   bar
100644 1d52a6 0   baz
100644 20b024 0   sub/fi
100644 43dbe0 0   sub/foo
```

“reading tree”

- checkout
- read-tree
- reset
Structure

repository | index | work tree

```
100644 20b024 0   bar
100644 1d52a6 0   baz
100644 20b024 0   sub/fi
100644 43dbe0 0   sub/foo
```

```
|-- bar
   |-- baz
   `-- sub
      `-- fi
      `-- foo
```

“checking out”

checkout
checkout-index
reset
The repository

.git
|-- HEAD current checkout reference
|-- config repo private config
|-- description repo description
|-- hooks
| `-- ... hooking scripts
|-- index changes to commit
|-- info
| |-- exclude repo private
| `-- refs refs?
|-- logs
| `-- ... “reflog” data
|-- objects
| |-- XX
| | `-- ... loose objects
| |-- info
| | `-- packs info about packs
| `-- pack
| | `-- ... packs and indexes
`-- refs
| `-- heads
| | `-- master master branch
| `-- tags
| | `-- ... tags
The repository

.git/config
  ▶ repository config

.git/description
  ▶ describes the repository useful for gitweb

.git/info/exclude
  ▶ patterns to ignore
The repository

.git/config

- repository config

.git:description

- describes the repository useful for gitweb

.git/info/exclude

- patterns to ignore
The repository

.git/config
  ▶ repository config

.git/description
  ▶ describes the repository useful for gitweb

.git/info/exclude
  ▶ patterns to ignore
Objects

.git/objects

|-- 23
|  |-- d4bd826aba9e29aaace9411cc175b784edc399
|-- 76
|  |-- 49f82d40a98b1ba59057798e47aab2a99a11d3
|-- c4
|  |-- aaefaa8a48ad4ad379dc1002b78f1a3e4ceabc
|-- e7
|  |-- 4be61128eef713459ca4e32398d689fe80864e
|-- info
|  |-- packs
`-- pack
   |-- pack-b7b026b1a0b0f193db9dea0b0d7367d25d3a68cc.idx
   `-- pack-b7b026b1a0b0f193db9dea0b0d7367d25d3a68cc.pack
Objects

.git/objects

|-- 23
|  `-- d4bd826aba9e29aaace9411cc175b784edc399
|-- 76
|  `-- 49f82d40a98b1ba59057798e47aab2a99a11d3
|-- c4
|  `-- aaefaa8a48ad4ad379dc1002b78f1a3e4ceabc
|-- e7
|  `-- 4be61128eef713459ca4e32398d689fe80864e
|-- info
|  `-- packs
`-- pack

|-- pack-b7b026b1a0b0f193db9dea0b0d7367d25d3a68cc.idx
`-- pack-b7b026b1a0b0f193db9dea0b0d7367d25d3a68cc.pack

"loose objects"
Objects

.git/objects
|-- 23
| `-- d4bd826aba9e29aaace9411cc175b784edc399
|-- 76
| `-- 49f82d40a98b1ba59057798e47aab2a99a11d3
|-- c4
| `-- aaefaa8a48ad4ad379dc1002b78f1a3e4ceabc
|-- e7
| `-- 4be61128eef713459ca4e32398d689fe80864e
|-- info
| `-- packs
`-- pack
    |-- pack-b7b026b1a0b0f193db9dea0b0d7367d25d3a68cc.idx
    `-- pack-b7b026b1a0b0f193db9dea0b0d7367d25d3a68cc.pack

“pack file”
Objects

content addressable

data
Objects

content addressable

![Diagram of an object with fields: type, size, data, and SHA1 hash]

SHA1 hash: 52a0ff44aba8599f43a5d821c421af316cb7305
Objects

content addressable

<table>
<thead>
<tr>
<th>type</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
</tr>
</tbody>
</table>

52a0ff44aba8599f43a5d821c421af316cb7305
Objects

content addressable

```
52a0ff44aba8599f43a5d821c421af316cb7305
```
Objects

4 types
  ▶ blobs

<table>
<thead>
<tr>
<th>&quot;blob&quot;</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>file data</td>
<td></td>
</tr>
</tbody>
</table>
Objects

4 types
  ▶ blobs
  ▶ trees
Objects

4 types

- blobs
- trees
Objects

4 types
  ► blobs
  ► trees
Objects

4 types

- blobs
- trees
- commits

```
commit  cadbae
parent  b051d6
author  Someone <address>
committer Another <address>
commit message

tree 205f6b
040000 tree 205f6b... somedir
100644 blob 9daef... somefile

blob 257cc5
040000 blob 257cc5... other

file dat
```

Git the basics

Bart
Trojanowski, bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History

Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
Objects

4 types

- blobs
- trees
- commits
Objects

4 types

- blobs
- trees
- commits
Objects

4 types
- blobs
- trees
- commits
- tags
## Objects

<table>
<thead>
<tr>
<th>type</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
</tr>
</tbody>
</table>
Objects

immutable

<table>
<thead>
<tr>
<th>type</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>type</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>data'</td>
<td></td>
</tr>
</tbody>
</table>
Objects

immutable

<table>
<thead>
<tr>
<th>type</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
</tr>
</tbody>
</table>

- SHA1: 8a9fd66b8bb939638564ebfdecc23b5d58070fc8
Next...

Concepts
  SCM components
  SCM operations
  Decentralization

GIT History

Repository
  Structure
  Objects

Using GIT
  Commands
  Commit
  Inspection
  Branching
  Merging
  Rebasing
  Remotes
  GUI tools
  Loose ends

Git the basics

Bart
Trojanowski,
bart@jukie.net
Git commands

$ git <options> <command> <options>
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td></td>
</tr>
<tr>
<td>am</td>
<td></td>
</tr>
<tr>
<td>annotate</td>
<td></td>
</tr>
<tr>
<td>apply</td>
<td></td>
</tr>
<tr>
<td>archimport</td>
<td></td>
</tr>
<tr>
<td>archive</td>
<td></td>
</tr>
<tr>
<td>bisect</td>
<td></td>
</tr>
<tr>
<td>blame</td>
<td></td>
</tr>
<tr>
<td>branch</td>
<td></td>
</tr>
<tr>
<td>bundle</td>
<td></td>
</tr>
<tr>
<td>cat-file</td>
<td></td>
</tr>
<tr>
<td>check-attr</td>
<td></td>
</tr>
<tr>
<td>check-ref-format</td>
<td></td>
</tr>
<tr>
<td>checkout</td>
<td></td>
</tr>
<tr>
<td>checkout-index</td>
<td></td>
</tr>
<tr>
<td>cherry</td>
<td></td>
</tr>
<tr>
<td>cherry-pick</td>
<td></td>
</tr>
<tr>
<td>citool</td>
<td></td>
</tr>
<tr>
<td>clean</td>
<td></td>
</tr>
<tr>
<td>clone</td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td></td>
</tr>
<tr>
<td>commit-tree</td>
<td></td>
</tr>
<tr>
<td>config</td>
<td></td>
</tr>
<tr>
<td>count-objects</td>
<td></td>
</tr>
<tr>
<td>cvsexportcommit</td>
<td></td>
</tr>
<tr>
<td>cvssimport</td>
<td></td>
</tr>
<tr>
<td>cvsserver</td>
<td></td>
</tr>
<tr>
<td>daemon</td>
<td></td>
</tr>
<tr>
<td>describe</td>
<td></td>
</tr>
<tr>
<td>diff</td>
<td></td>
</tr>
<tr>
<td>diff-files</td>
<td></td>
</tr>
<tr>
<td>diff-index</td>
<td></td>
</tr>
<tr>
<td>diff-tree</td>
<td></td>
</tr>
<tr>
<td>fast-export</td>
<td></td>
</tr>
<tr>
<td>fast-import</td>
<td></td>
</tr>
<tr>
<td>fetch</td>
<td></td>
</tr>
<tr>
<td>fetch-pack</td>
<td></td>
</tr>
<tr>
<td>filter-branch</td>
<td></td>
</tr>
<tr>
<td>fmt-merge-msg</td>
<td></td>
</tr>
<tr>
<td>for-each-ref</td>
<td></td>
</tr>
<tr>
<td>format-patch</td>
<td></td>
</tr>
<tr>
<td>fsck</td>
<td></td>
</tr>
<tr>
<td>fsck-objects</td>
<td></td>
</tr>
<tr>
<td>gc</td>
<td></td>
</tr>
<tr>
<td>get-tar-commit-id</td>
<td></td>
</tr>
<tr>
<td>grep</td>
<td></td>
</tr>
<tr>
<td>gui</td>
<td></td>
</tr>
<tr>
<td>hash-object</td>
<td></td>
</tr>
<tr>
<td>http-fetch</td>
<td></td>
</tr>
<tr>
<td>http-push</td>
<td></td>
</tr>
<tr>
<td>imap-send</td>
<td></td>
</tr>
<tr>
<td>index-pack</td>
<td></td>
</tr>
<tr>
<td>init</td>
<td></td>
</tr>
<tr>
<td>init-db</td>
<td></td>
</tr>
<tr>
<td>instaweb</td>
<td></td>
</tr>
<tr>
<td>log</td>
<td></td>
</tr>
<tr>
<td>lost-found</td>
<td></td>
</tr>
<tr>
<td>ls-files</td>
<td></td>
</tr>
<tr>
<td>ls-remote</td>
<td></td>
</tr>
<tr>
<td>ls-tree</td>
<td></td>
</tr>
<tr>
<td>mailinfo</td>
<td></td>
</tr>
<tr>
<td>mailsplit</td>
<td></td>
</tr>
<tr>
<td>merge</td>
<td></td>
</tr>
<tr>
<td>merge-base</td>
<td></td>
</tr>
<tr>
<td>merge-file</td>
<td></td>
</tr>
<tr>
<td>merge-index</td>
<td></td>
</tr>
<tr>
<td>merge-one-file</td>
<td></td>
</tr>
<tr>
<td>merge-resolve</td>
<td></td>
</tr>
<tr>
<td>merge-subtree</td>
<td></td>
</tr>
<tr>
<td>merge-tree</td>
<td></td>
</tr>
<tr>
<td>mergetool</td>
<td></td>
</tr>
<tr>
<td>mktag</td>
<td></td>
</tr>
<tr>
<td>mktree</td>
<td></td>
</tr>
<tr>
<td>mv</td>
<td></td>
</tr>
<tr>
<td>name-rev</td>
<td></td>
</tr>
<tr>
<td>pack-objects</td>
<td></td>
</tr>
<tr>
<td>pack-redundant</td>
<td></td>
</tr>
<tr>
<td>pack-refs</td>
<td></td>
</tr>
<tr>
<td>parse-remote</td>
<td></td>
</tr>
<tr>
<td>patch-id</td>
<td></td>
</tr>
<tr>
<td>peek-remote</td>
<td></td>
</tr>
<tr>
<td>prune</td>
<td></td>
</tr>
<tr>
<td>prune-packed</td>
<td></td>
</tr>
<tr>
<td>pull</td>
<td></td>
</tr>
<tr>
<td>push</td>
<td></td>
</tr>
<tr>
<td>quitimport</td>
<td></td>
</tr>
<tr>
<td>read-tree</td>
<td></td>
</tr>
<tr>
<td>rebase</td>
<td></td>
</tr>
<tr>
<td>receive-pack</td>
<td></td>
</tr>
<tr>
<td>reflog</td>
<td></td>
</tr>
<tr>
<td>relink</td>
<td></td>
</tr>
<tr>
<td>remote</td>
<td></td>
</tr>
<tr>
<td>repack</td>
<td></td>
</tr>
<tr>
<td>repo-config</td>
<td></td>
</tr>
<tr>
<td>request-pull</td>
<td></td>
</tr>
<tr>
<td>rerere</td>
<td></td>
</tr>
<tr>
<td>reset</td>
<td></td>
</tr>
<tr>
<td>rev-list</td>
<td></td>
</tr>
<tr>
<td>rev-parse</td>
<td></td>
</tr>
<tr>
<td>revert</td>
<td></td>
</tr>
<tr>
<td>rm</td>
<td></td>
</tr>
<tr>
<td>send-email</td>
<td></td>
</tr>
<tr>
<td>send-pack</td>
<td></td>
</tr>
<tr>
<td>sh-setup</td>
<td></td>
</tr>
<tr>
<td>shell</td>
<td></td>
</tr>
<tr>
<td>shortlog</td>
<td></td>
</tr>
<tr>
<td>show</td>
<td></td>
</tr>
<tr>
<td>show-branch</td>
<td></td>
</tr>
<tr>
<td>show-index</td>
<td></td>
</tr>
<tr>
<td>show-ref</td>
<td></td>
</tr>
<tr>
<td>stash</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td></td>
</tr>
<tr>
<td>stripespace</td>
<td></td>
</tr>
<tr>
<td>submodule</td>
<td></td>
</tr>
<tr>
<td>svn</td>
<td></td>
</tr>
<tr>
<td>symbolic-ref</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td></td>
</tr>
<tr>
<td>tar-tree</td>
<td></td>
</tr>
<tr>
<td>unpack-file</td>
<td></td>
</tr>
<tr>
<td>unpack-objects</td>
<td></td>
</tr>
<tr>
<td>update-index</td>
<td></td>
</tr>
<tr>
<td>update-ref</td>
<td></td>
</tr>
<tr>
<td>update-server-info</td>
<td></td>
</tr>
<tr>
<td>upload-archive</td>
<td></td>
</tr>
<tr>
<td>upload-pack</td>
<td></td>
</tr>
<tr>
<td>var</td>
<td></td>
</tr>
<tr>
<td>verify-pack</td>
<td></td>
</tr>
<tr>
<td>verify-tag</td>
<td></td>
</tr>
<tr>
<td>whatchanged</td>
<td></td>
</tr>
<tr>
<td>write-tree</td>
<td></td>
</tr>
<tr>
<td>gitk</td>
<td></td>
</tr>
</tbody>
</table>
Every day use...

add
fetch

branch

gc

grep

checkout

clone
init

commit

config
log

diff
merge

rm

mv
show

pull
tag

clone
init

commit

config
log

diff
merge

remote
reset

Clone
init

commit

config
log

diff
merge

remote
reset
Some GUI tools...

- add
- fetch
- rm
- branch
- gc
- show
- checkout
- grep
- status
- citool
- pull
- tag
- clone
- init
- push
- commit
- grep
- rebase
- config
- log
- remote
- diff
- merge
- reset
- Gitk
Occasional use...

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>revert</td>
</tr>
<tr>
<td>am</td>
<td>rm</td>
</tr>
<tr>
<td>annotate</td>
<td>shell</td>
</tr>
<tr>
<td>apply</td>
<td>shortlog</td>
</tr>
<tr>
<td>archive</td>
<td>show</td>
</tr>
<tr>
<td>bisect</td>
<td>show-branch</td>
</tr>
<tr>
<td>blame</td>
<td></td>
</tr>
<tr>
<td>branch</td>
<td></td>
</tr>
<tr>
<td>bundle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>gc</td>
</tr>
<tr>
<td></td>
<td>grep</td>
</tr>
<tr>
<td>checkout</td>
<td>gui</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>cherry</td>
<td></td>
</tr>
<tr>
<td>cherry-pick</td>
<td></td>
</tr>
<tr>
<td>citool</td>
<td></td>
</tr>
<tr>
<td>clean</td>
<td></td>
</tr>
<tr>
<td>clone</td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td></td>
</tr>
<tr>
<td>config</td>
<td></td>
</tr>
<tr>
<td></td>
<td>push</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>describe</td>
<td></td>
</tr>
<tr>
<td>diff</td>
<td></td>
</tr>
<tr>
<td>merge</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rebase</td>
</tr>
<tr>
<td></td>
<td>reflog</td>
</tr>
<tr>
<td></td>
<td>update-server-info</td>
</tr>
<tr>
<td></td>
<td>remote</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>gitk</td>
</tr>
</tbody>
</table>
And the plumbing...
Help

```
  git help
    ▶ list of common commands

  git <command> -h
    ▶ brief help output

  man git-<command>
  git help <command>
  git <command> --help
    ▶ manual page
```
Help

```
git help
  ➤ list of common commands

git <command> -h
  ➤ brief help output

man git-<command>
git help <command>
git <command> --help
  ➤ manual page
```
Help

```sh
git help
  ▶ list of common commands

git <command> -h
  ▶ brief help output

man git-<command>
git help <command>
git <command> --help
  ▶ manual page
```
Configuration

$HOME/.gitconfig

$ git config --global user.name "Your Name"
$ git config --global user.email you@domain.tld

$ git config --global color.pager true
$ git config --global color.ui auto
Configuration

$HOME/.gitconfig

$ git config --global user.name "Your Name"
$ git config --global user.email you@domain.tld

$ git config --global color.pager true
$ git config --global color.ui auto
Configuration

$HOME/.gitconfig

$ git config --global user.name "Your Name"
$ git config --global user.email you@domain.tld

$ git config --global color.pager true
$ git config --global color.ui auto
Configuration

```
$ cat .gitconfig

[user]
  name = "Bart Trojanowski"
  email = "bart@jukie.net"
  signingkey = 2289688F

[core]
  pager = less -FRSX
  editor = vim

[color]
  ui = auto

[merge]
  tool = vimdiff
```
Git the basics
Bart
Trojanowski,
bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History
Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
-
Bootstrapping

$ git init

- ran in project workspace
- creates .git directory
Staging

What to commit?

- additions
  - $ git add file
  - $ git add .

- removal
  - $ git rm file

- renames
  - $ git mv old new
Staging

What to commit?

- additions
  - $ git add file
  - $ git add .

- removal
  - $ git rm file

- renames
  - $ git mv old new
Staging

What to commit?

- additions
  
  $ git add file
  $ git add .

- removal
  
  $ git rm file

- renames
  
  $ git mv old new
Staging

What to commit?

- additions
  $ git add file
  $ git add .

- removal
  $ git rm file

- renames
  $ git mv old new
Staging

$ cat .gitignore

*.o
*~
Committing

$ git commit -a -m 'some comment'

- will create a commit of all or only staged items
Bootstrapping

$ mkdir project
$ cd project
$ git init
First commit

$ echo test > test
Stage

$ echo test > test
$ git add test
Commit

$ echo test > test
$ git add test

$ git commit -m 'test'
Created initial commit 6f01040: test
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 test
Stage another

$ echo test > test
$ git add test

$ git commit -m 'test'

$ mkdir dir
$ echo foo > dir/foo
$ git add dir/foo
Commit another

$ echo test > test
$ git add test

$ git commit -m 'test'

$ mkdir dir
$ echo foo > dir/foo
$ git add dir/foo

$ git commit -m 'foo'
Created commit 52a0ff4: foo
  1 files changed, 1 insertions(+), 0 deletions(-)

create mode 100644 dir/foo
Repository status

$ git status

shows...

- staged
- unstaged
- untracked
Diffs

$ git diff
  ▶ changes between index and working files

$ git diff --staged
  ▶ changes between HEAD and index

$ git diff HEAD
  ▶ changes between HEAD and working files

$ git diff $commit $commit
  ▶ changes between two commits
Diffs

$ git diff
  ▶ changes between index and working files

$ git diff --staged
  ▶ changes between HEAD and index

$ git diff HEAD
  ▶ changes between HEAD and working files

$ git diff $commit $commit
  ▶ changes between two commits
Diffs

$ git diff
  ▶ changes between index and working files

$ git diff --staged
  ▶ changes between HEAD and index

$ git diff HEAD
  ▶ changes between HEAD and working files

$ git diff $commit $commit
  ▶ changes between two commits
Diffs

```
$ git diff
  ▶ changes between index and working files

$ git diff --staged
  ▶ changes between HEAD and index

$ git diff HEAD
  ▶ changes between HEAD and working files

$ git diff $commit $commit
  ▶ changes between two commits
```
Object references

specific commit ID...

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>full hash</td>
<td>6bb1270ffe60cbefef87266d2d4abe4218d9c68</td>
</tr>
<tr>
<td>short hash</td>
<td>6bb127</td>
</tr>
<tr>
<td>tag</td>
<td>v1.5.6.1</td>
</tr>
<tr>
<td>local branch</td>
<td>master</td>
</tr>
<tr>
<td>remote branch</td>
<td>origin/master</td>
</tr>
<tr>
<td>by message</td>
<td>&quot;~/some text&quot;</td>
</tr>
<tr>
<td>checkout</td>
<td>HEAD</td>
</tr>
<tr>
<td>last fetch</td>
<td>FETCH_HEAD</td>
</tr>
<tr>
<td>previous head</td>
<td>ORIG_HEAD</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Object references

a commit before HEAD

\[
\text{HEAD}^\wedge \quad == \quad \text{HEAD}^\sim 1
\]
Object references

few commits before HEAD

\[ \text{HEAD}^{\ldots} \quad == \quad \text{HEAD}^{\sim 3} \]
Object references

few commits before master

\[ \text{master}^{\cdots} = \text{master}^{-3} \]
Object references

what was it yesterday?

@\{yesterday\} == HEAD@\{yesterday\}
Object references

how about my-other-branch on June 1st?

my-other-branch@{June.1}
Object references

master a few changes ago?

master@{3}
Show me!
review last commit...

$ git show

commit 83b2d051814e884a8e264127ed47552a5dcf6c1d
Author: Bart Trojanowski <bart@jukie.net>
Date: Thu Jul 3 21:44:39 2008 -0400

changed one line

diff --git a/test b/test
index 808a2c4..99810fa 100644
--- a/test
+++ b/test
@@ -1,3 +1,3 @@
   Some old text before the change.
-Some text for removal.
+Replacement line.
   Some old text after the change.
Show me!

just the stats...

$ git show --stat

commit 83b2d051814e884a8e264127ed47552a5dcaf6c1d
Author: Bart Trojanowski <bart@jukie.net>
Date:   Thu Jul 3 21:44:39 2008 -0400

    changed one line

test |  2 +–
1 files changed, 1 insertions(+), 1 deletions(–)
Show me!

SVN’esq status information...

$ git show --name-status

commit 3d3d2989b817af3fd4fa6d63f200113bd6c94bdb
Author: Bart Trojanowski <bart@jukie.net>
Date: Thu Jul 3 22:59:13 2008 -0400

    something more interesting

A    sub/bar
D    sub/foo
M    test
Show me!
review any other commit...

$ git show HEAD
$ git show HEAD~^
$ git show master~10
$ git show master@{May.16}
...

Git the basics

Bart
Trojanowski,
bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History

Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
Show me!
show a file (or tree) in history...

$ git show HEAD:file
contents...
Logs

See commit history...

```
$ git log
commit 3d3d2989b817af3fd4fa6d63f200113bd6c94bdb
Author: Bart Trojanowski <bart@jukie.net>
Date: Thu Jul 3 22:59:13 2008 -0400

    most recent commit

commit 83b2d051814e884a8e264127ed47552a5dcf6c1d
Author: Bart Trojanowski <bart@jukie.net>
Date: Thu Jul 3 21:44:39 2008 -0400

    second most recent

commit 1cc1b35a611c39f49842e2ca28d40886c1ae9b7c
Author: Bart Trojanowski <bart@jukie.net>
Date: Thu Jul 3 21:44:05 2008 -0400

    middle commit

commit 411515f51a78d66a27a7d56ebe9f70dbd2bff008
Author: Bart Trojanowski <bart@jukie.net>
Date: Thu Jul 3 21:43:36 2008 -0400

    second oldest

commit 52a0ff44aba8599f43a5d821c421af316cb73051
Author: Bart Trojanowski <bart@jukie.net>
Date: Mon Jun 30 21:44:55 2008 -0400

    oldest commit
```
Logs

```
git log is awesome!
```

Git the basics

Bart
Trojanowski,
bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History
Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
Logs

limit by range...

$ git log tag..branch
$ git log HEAD~10..
$ git log branch1 branch2 ~common

$ git log -10
$ git log -10 master@[yesterday]

$ git log --since="May 1" --until="June 1"
Logs

limit by commit attributes...

```bash
$ git log --author=fred
$ git log --committer=joe
$ git log --grep="commit.*message.*text"
```
Logs

search for a change...

$ git log -S"some code change"
$ git log --pickaxe-regex -S"some.*code.*change"
Logs

limit by changes to specific path...

$ git log -- some/file
Faster grep
another way to search...

$ git grep -e "pattern" -- some/file

$ git grep -e "pattern" branch -- some/file
Faster grep
another way to search...

$ git grep -e "pattern" -- some/file

$ git grep -e "pattern" branch -- some/file
Git the basics
Bart
Trojanowski,
bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History

Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
-
References

things that point to commits

lightweight

mutable

disposable

3 basic types
References

things that point to commits

lightweight

mutable

disposable

3 basic types
References

- things that point to commits
- lightweight
- mutable
- disposable

3 basic types
References

things that point to commits

lightweight

mutable

disposable

3 basic types
References

things that point to commits

lightweight

mutable

disposable

3 basic types
local branches

- $ git branch -l
  
  branch1
  branch2
  * master

- .git/refs/heads/<branch>
References

tags

- $ git tag -l
  tag1
  tag2
  tag3

- .git/refs/tags/<tag>
References

remote branches

- `$ git branch -r`
  - fred/master
  - joe/master
  - joe/another-branch

- `.git/refs/remotes/<remote>/<branch>`
Creating branches

$ git branch name commit

- new branch “name” on HEAD or specified commit
Switching branches

$ git checkout -f name

- checkout files from “name” branch
- optionally force overwriting changed files
Create and switch

$ git checkout -b name commit

- checkout files from “name” branch
Switching with changes

$ git checkout name
error: You have local changes to 'filename'; cannot switch branches.

$ git checkout -m name
  ▶ merge outstanding diff onto branch "name"
  ▶ can result in conflict
Switching with changes

$ git checkout name
error: You have local changes to 'filename'; cannot switch branches.

$ git checkout -m name
  ▶ merge outstanding diff onto branch “name”
  ▶ can result in conflict
Working on branches

start at some tree
Working on branches

$ git checkout -b bug-fix
Working on branches

$ git commit -a -m ‘B’
Working on branches

$ git commit -a -m‘C’
Working on branches

you have a *wicked* idea
Working on branches

you have a *wicked* idea

$ git checkout -b wicked master
Working on branches

$ git commit -a -m‘D’
Working on branches

$ git commit -a -m ‘E’
Working on branches

you’re getting somewhere
Working on branches

you're getting somewhere

$ git tag -a -m 'got somewhere' good
Working on branches

manager asks about the bug
Working on branches

manager asks about the bug

$ git checkout bug-fix
Working on branches

$ git commit -a -m 'F'

[Diagram of Git branch structure]
Working on branches

your mind is elsewhere...
Working on branches

your mind is elsewhere...

$ git checkout wicked
Working on branches

...so you finish off the *wicked* feature

```
$ git commit -a -m"G"
```
Working on branches

feature’s done

bug is fixed

...time to merge
Working on branches

feature’s done

bug is fixed

...time to merge
Working on branches

feature’s done

bug is fixed

... time to merge
Working on branches

$ git checkout master
Working on branches

$ git reset --hard bug-fix
Working on branches

$ git merge wicked
Git the basics
Bart
Trojanowski, bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History

Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
Merging

$ git merge <branch> ...

- merge multiple branches
- creates commit with 2+ parents
- can cause conflicts
  ie: require user intervention

Git the basics
Bart
Trojanowski, bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History
Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
-
Merging

one more example
Merging

two topic branches
Merging

```
$ git checkout -b three two
```
Merging

$ git checkout -b three two
$ git merge one
Merging
Merging

$ git checkout three
Merging

$ git checkout three
$ git merge one two
Merging

$ git checkout three
$ git merge one two

“octopus”
Rebasing

```
$ git rebase <branch>
```
- moves new work onto a new baseline
- can cause conflicts
  ie: require user intervention
Rebasing

merge

take two identical trees

rebase
Rebasing

merge

$ git merge master
Rebasing

merge

that's easy

rebase

Git the basics
Bart
Trojanowski,
bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History
Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends

master
test
master
test

A
B
C
D
E
F
A
B
C
Rebasing

merge

```
$ git rebase master
```
Rebasing

merge

$ git rebase master
Rebasing

merge

```
$ git rebase master
```

rebase
Rebasing

merge

$ git rebase master
Rebasing

merge

test

master → C → B → A

F → E → D

rebase

test

E' → D' → E → D

master → C → B → A

$ git rebase master
Rebasing

merge

test

F

E

D

master

C

B

A

$ git rebase master

rebase

test

E'

D'

E

D

master

C

B

A

$ git rebase master
Rebasing

merge

rebase

$ git rebase master
Rebasing

merge

$ git rebase master
Rebasing

merge

<table>
<thead>
<tr>
<th>master</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

rebase

<table>
<thead>
<tr>
<th>master</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E'</td>
<td>D'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>test</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Git the basics
Bart Trojanowski, bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History
Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
Cloning

$ git clone <remote>

- replicates remote repository
- populates new repository
- checkout new working tree
Git URLs

▶ local repo

/home/git/project.git/
file:///home/git/project.git/

▶ http protocol

http://repo.or.cz/r/git.git

▶ native git protocol

git://repo.or.cz/git.git

▶ ssh protocol

ssh://bart@jukie.net/~git/project.git/
bart@jukie.net/~git/project.git/
Git URLs

- local repo
  /home/git/project.git/
  file:///home/git/project.git/

- http protocol
  http://repo.or.cz/r/git.git

- native git protocol
  git://repo.or.cz/git.git

- ssh protocol
  ssh://bart@jukie.net/~/git/project.git/
  bart@jukie.net/~/git/project.git/
Git URLs

- **local repo**
  
  /home/git/project.git/
  file:///home/git/project.git/

- **http protocol**
  
  http://repo.or.cz/r/git.git

- **native git protocol**
  
  git://repo.or.cz/git.git

- **ssh protocol**
  
  ssh://bart@jukie.net/~/git/project.git/
  bart@jukie.net/~/git/project.git/
Git URLs

- local repo
  /home/git/project.git/
  file:///home/git/project.git/

- http protocol
  http://repo.or.cz/r/git.git

- native git protocol
  git://repo.or.cz/git.git

- ssh protocol
  ssh://bart@jukie.net/~git/project.git/
  bart@jukie.net/~git/project.git/
Working with remotes

repo.or.cz

```
+ master
  |   F
  |   C
  |   B
  |   A
```

git://repo.or.cz/project.git
Working with remotes

repo.or.cz

```
$ git clone ssh+git://repo.or.or.cz/project.git
```
Working with remotes

repo.or.cz local

$ git clone ssh+git://repo.or.cz/project.git
Initialized project/.git
Initialized empty Git repository in /tmp/project/.git/
remote: Counting objects: 77575, done.
remote: Compressing objects: 100% (26407/26407), done.
remote: Total 77575 (delta 55750), reused 71007 (delta 49775)
Receiving objects: 100% (77575/77575), 22.87 MiB | 798 KiB/s, done.
Resolving deltas: 100% (55750/55750), done.
Checking out files: 100% (1396/1396), done.
Working with remotes

repo.or.cz local

$ git clone ssh+git://repo.or.cz/project.git
Initialize project/.git
Initialized empty Git repository in /tmp/project/.git/
remote: Counting objects: 77575, done.
remote: Compressing objects: 100% (26407/26407), done.
remote: Total 77575 (delta 55750), reused 71007 (delta 49775)
Receiving objects: 100% (77575/77575), 22.87 MiB | 798 KiB/s, done.
Resolving deltas: 100% (55750/55750), done.
Checking out files: 100% (1396/1396), done.
Working with remotes

repo.or.cz    local

```
$ git clone ssh+git://repo.or.cz/project.git build-tree
Initialize build-tree/.git
Initialized empty Git repository in /tmp/build-tree/.git/
remote: Counting objects: 77575, done.
remote: Compressing objects: 100% (26407/26407), done.
remote: Total 77575 (delta 55750), reused 71007 (delta 49775)
Receiving objects: 100% (77575/77575), 22.87 MiB | 798 KiB/s, done.
Resolving deltas: 100% (55750/55750), done.
Checking out files: 100% (1396/1396), done.
```
Working with remotes

repo.or.cz  local

$ git clone ssh+git://repo.or.cz/project.git
Initialized project/.git
Initialized empty Git repository in /tmp/project/.git/
remote: Counting objects: 77575, done.
remote: Compressing objects: 100% (26407/26407), done.
remote: Total 77575 (delta 55750), reused 71007 (delta 49775)
Receiving objects: 100% (77575/77575), 22.87 MiB | 798 KiB/s, done.
Resolving deltas: 100% (55750/55750), done.
Checking out files: 100% (1396/1396), done.
$ git clone ssh+git://repo.or.cz/project.git

Initialized project/.git
Initialized empty Git repository in /tmp/project/.git/
remote: Counting objects: 77575, done.
remote: Compressing objects: 100% (26407/26407), done.
remote: Total 77575 (delta 55750), reused 71007 (delta 49775)
Receiving objects: 100% (77575/77575), 22.87 MiB | 798 KiB/s, done.
Resolving deltas: 100% (55750/55750), done.
Checking out files: 100% (1396/1396), done.
Working with remotes

$ git clone ssh+git://repo.or.cz/project.git
Initialized project/.git
Initialized empty Git repository in /tmp/project/.git/
remote: Counting objects: 77575, done.
remote: Compressing objects: 100% (26407/26407), done.
remote: Total 77575 (delta 55750), reused 71007 (delta 49775)
Receiving objects: 100% (77575/77575), 22.87 MiB | 798 KiB/s, done.
Resolving deltas: 100% (55750/55750), done.
Checking out files: 100% (1396/1396), done.

Git the basics
Bart Trojanowski, bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History
Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
$ git clone ssh+git://repo.or.cz/project.git

Initialized project/.git
Initialized empty Git repository in /tmp/project/.git/
remote: Counting objects: 77575, done.
remote: Compressing objects: 100% (26407/26407), done.
remote: Total 77575 (delta 55750), reused 71007 (delta 49775)
Receiving objects: 100% (77575/77575), 22.87 MiB | 798 KiB/s, done.
Resolving deltas: 100% (55750/55750), done.
Checking out files: 100% (1396/1396), done.
Working with remotes

"origin"

local

$ git clone ssh+git://repo.or.cz/project.git
Initialized project/.git
Initialized empty Git repository in /tmp/project/.git/
remote: Counting objects: 77575, done.
remote: Compressing objects: 100% (26407/26407), done.
remote: Total 77575 (delta 55750), reused 71007 (delta 49775)
Receiving objects: 100% (77575/77575), 22.87 MiB | 798 KiB/s, done.
Resolving deltas: 100% (55750/55750), done.
Checking out files: 100% (1396/1396), done.
Working with remotes

```
$ git branch -a
* master
  origin/master

$ git tag -l
v0.1.0
v0.2.0
```
Working with remotes

"origin"

```
master

F
C  v0.2.0
B  v0.1.0
A  v0.1.0
```

`/tmp/project/`
```
|−− git/
|−−− bar
|−−−− sub/
|−−−−− foo
```

```
$ git branch -a
* master
  origin/master

$ git tag -l
  v0.1.0
  v0.2.0
```
Working with remotes

"origin"

```
A → v0.1.0
B → v0.1.0
C → v0.2.0
D
```

```
master
H
F
G
```

```
/local

master
```

```
origin/master

F
C
B
A
```

```
/tmp/project/
|--git/
|   |--bar
|   `--sub/
     `--foo
```

local
$ git fetch
Working with remotes

"origin"

local

$ git fetch
remote: Counting objects: 236, done.
remote: Compressing objects: 100% (190/190), done.
remote: Total 190 (delta 170), reused 0 (delta 0)
Receiving objects: 100% (190/190), 69.53 KiB, done.
Resolving deltas: 100% (170/170), completed with 40 local objects.
From mail.jukie.net:work/oclug/intro-to-git
  573ff80..06e3703  master  ->  origin/master
$ git fetch
remote: Counting objects: 236, done.
remote: Compressing objects: 100% (190/190), done.
remote: Total 190 (delta 170), reused 0 (delta 0)
Receiving objects: 100% (190/190), 69.53 KiB, done.
Resolving deltas: 100% (170/170), completed with 40 local objects.
From mail.jukie.net:work/oclug/intro-to-git
  573ff80..06e3703 master -> origin/master
Working with remotes

```
$ git fetch
remote: Counting objects: 236, done.
remote: Compressing objects: 100% (190/190), done.
remote: Total 190 (delta 170), reused 0 (delta 0)
Receiving objects: 100% (190/190), 69.53 KiB, done.
Resolving deltas: 100% (170/170), completed with 40 local objects.
From mail.jukie.net:work/oclug/intro-to-git
  573ff80..06e3703 master -> origin/master
```
Working with remotes

"origin"

local

... no disk changes?
Working with remotes

"origin"

local

git fetch only updates DAG
Working with remotes

"origin"

```
|−− master
|−− H
|−− G
|−− F
| −− C
| −− B
| −− A
`−− v0.1.0
```

```
|−− origin/master
|−− H
|−− G
|−− F
| −− C
| −− B
| −− A
`−− v0.1.0
```

git merge origin/master

```
f
```

fast-forwards master to match origin/master
and updates the working tree
Working with remotes

"origin"

local

```
/git merge origin/master
fast-forwards master to match origin/master
and updates the working tree
```
Working with remotes

"origin"

local

```
git merge origin/master
fast-forwards master to match origin/master
and updates the working tree
```
Working with remotes

"origin"

local

```
$ git fetch + git merge

= $ git pull
```
Working with remotes

"origin"

local

```
git fetch + git merge
= git pull
```
Working with remotes

"origin"

local

let’s make some local commits
Working with remotes

"origin"

```
$ git commit -a -m 'I'
```

local

```
$ git commit -a -m 'I'
```
Working with remotes

"origin"

```
$ git commit -a -m 'I'
Created commit b618aed: I
  2 files changed, 11 insertions(+), 20 deletions(-)
```
Working with remotes

"origin"

local

/tmp/project/
|-- .git/
|-- bar
|   |-- fi
|   |-- foo

master
H
G
F
C
B
A
v0.1.0
origin/master
I
H
G
F
C
B
A
v0.2.0

Git the basics
Bart Trojanowski, bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History

Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
Working with remotes

"origin"

master
H
F
G
C
B
A
v0.2.0
v0.1.0

origin/master

master

H
F
G
C
B
A
v0.2.0
v0.1.0

/local

tmp/project/

\-\- .git/
\-\- bar
\-\- sub/
\-\- fi
\-\- foo

$ git push
### Working with remotes

```
$ git push
Counting objects: 9, done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 810 bytes, done.
Total 5 (delta 4), reused 0 (delta 0)
refs/heads/master: 9ddc135 -> 15b67c0
To ssh+git://repo.or.cz/project.git
  9ddc135..15b67c0  master -> master
```
$ git push
Counting objects: 9, done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 810 bytes, done.
Total 5 (delta 4), reused 0 (delta 0)
refs/heads/master: 9ddc135 -> 15b67c0
To ssh+git://repo.or.cz/project.git
  9ddc135..15b67c0  master -> master
Working with remotes

"origin"

```
A (v0.1.0)
B (v0.1.0)
C (v0.2.0)

/private/project/
|--.git/
|-- bar
|-- sub/
| -- fi
| -- foo
```

local

```
A (v0.1.0)
B (v0.1.0)
C (v0.2.0)

/private/project/
|--.git/
```

Git the basics

Bart
Trojanowski,
bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History
Repository
Structure
Objects

Using GIT
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
-
Working with remotes

"origin"

... meanwhile, elsewhere on the internet ...

foo.com
Working with remotes

"origin"

foo.com

Fred clones the project.
Working with remotes

"origin"

Fred clones the project.
Working with remotes

"origin"

...and makes some changes.

foo.com
Working with remotes

"origin"

Fred cannot push.
Working with remotes
"origin"

```
From: fred@foo.com
Subject: I fixed a bug

Please pull from

    http://foo.com/project.git/

-Fred
```
Working with remotes

"origin"

local

foo.com

From: fred@foo.com
Subject: I fixed a bug

Please pull from

http://foo.com/project.git/

-Fred
Working with remotes

"origin"

local

foo.com

$ git remote add fred http://foo.com/project.git/
Working with remotes

"origin"

local

foo.com

$ git remote add fred http://foo.com/project.git/
Working with remotes

"origin"

```
$ git remote add fred http://foo.com/project.git/
```
Working with remotes

"origin"

local

"fred"

$ git fetch fred
Working with remotes

"origin"

local

"fred"

$ git fetch fred
Working with remotes

"origin"

local

"fred"

$ git fetch fred
Working with remotes

"origin"

master
I
H
G
F
C
B
A
v0.1.0
v0.2.0

local

master
K
I
H
G
F
C
B
A
v0.1.0
v0.2.0

"fred"

master
J
I
H
G
F
C
B
A
v0.1.0
v0.2.0

/git/
|-- bar
|-- sub/
|   |-- fi
|   |-- foo

/tmp/project/
Working with remotes

"origin"

"fred"

$ git log fred/master
Working with remotes

"origin"

local

"fred"

$ git log fred/master
$ git log fred/master ~master
Working with remotes

"origin"

local

"fred"

```
$ git log fred/master
$ git log fred/master ^master
$ git log -p fred/master ^master
```
Working with remotes

"origin"

local

"fred"

$ git checkout -b fred-fix fred/master
Working with remotes

"origin"

local

"fred"

$ git checkout -b fred-fix fred/master
Working with remotes

"origin"

local

"fred"

$ git checkout master

Git the basics
Bart
Trojanowski,
bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History
Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasings
Remotes
GUI tools
Loose ends
Working with remotes

"origin"

local

"fred"

```bash
$ git checkout master
$ git merge fred-fix
```
$ git checkout master
$ git merge fred-fix
$ git branch -d fred-fix
Working with remotes

"origin"

$ git push
Working with remotes

"origin"

local

"fred"

$ git push
Working with remotes

"origin"

```
+ master
|   L
|   K
|   J
|   I
|   H
|   G
|   F
|   C
|   B
|   A
```

```
$ git push
```
Working with remotes

"origin"

```
A ← v0.1.0
B  v0.1.0
C ← v0.2.0
D  v0.2.0
E ← v0.1.0
```

local

```
A ← v0.1.0
B  v0.1.0
C ← v0.2.0
D  v0.2.0
E ← v0.1.0
```

"fred"

```
A ← v0.1.0
B  v0.1.0
C ← v0.2.0
D  v0.2.0
E ← v0.1.0
```

/git/
  | ← bar
  | ← sub/
  | ← fi
  | ← foo

/tmp/project/
  | ← git/
  | ← bar
  | ← sub/
  | ← fi
  | ← foo

---

Git the basics
Bart Trojanowski, bart@jukie.net

Concepts
SCM components
SCM operations
Decentralization

GIT History
Repository
Structure
Objects

Using GIT
Commands
Commit
Inspection
Branching
Merging
Rebasing
Remotes
GUI tools
Loose ends
GUI tools

- **gitk**
  sexy view of your revision tree

- **git gui**
  perform trivial tasks from a GUI
  ex: add/rm files, make commits, branch, etc
GUI tools

- **gitk**
  sexy view of your revision tree

- **git gui**
  perform trivial tasks from a GUI
  ex: add/rm files, make commits, branch, etc
gitk

how sexy?
let git-case-assign assign to no one
load git-case-common from the location of the exec (closes 3ed4c7)
arguments for git-case-new
use gen_short_case_id() to shorten case ID's
Documentation/comment-format-proposal
git-case-assign cleanup
git-case-merge will now merge (closes 0dc37c)
Merge commit origin/aidan/master
  git-case-assign: Extend it to allow assigning to others
    Simplify add_file_to_index
    possible fix for git-case-merge
  Work around problem with in second argument to add_file_to_index.
    Merge branch 'master' of git+ssh://dave@tachyon.jukie.net/home/git/gi
groups, git-case-edit-description was storing the comment in the descri
    git-case-edit-description
    Allow git 1.5.3, as it's in etch-backports
    git-case-merge is broken, see case 0dc37c

Author: Bart Trojanowski <bart@jukie.net> 2008-04-16 16:04:24
Commit: Bart Trojanowski <bart@jukie.net> 2008-04-16 16:04:24
Parent: 8d768ae33c8f585b69a190c9b626e4f25c5f4ef50578d32f382 [arguments for git-case-
Child: c893e62d6b0b8bab42ac4e3f49f6d6ef1cd32e88 [let git-case-assign ass
Branches: master, remotes/origin/aidan/master, remotes/origin/master
Follows: None
Precedes:

load git-case-common from the location of the exec (closes 3ed4c7)
Remember *loose objects* ?
Loose objects
.git/objects

|-- 23
|  `-- d4bd826aba9e29aaace9411cc175b784edc399
|-- 76
|  `-- 49f82d40a98b1ba59057798e47aab2a99a11d3
|-- c4
|  `-- aaeafa8a48ad4ad379dc1002b78f1a3e4ceabc
|-- e7
|  `-- 4be61128eef713459ca4e32398d689fe80864e
|-- info
| `-- packs
 `-- pack
  |-- pack-b7b026b1a0b0f193db9dea0b0d7367d25d3a68cc.idx
  `-- pack-b7b026b1a0b0f193db9dea0b0d7367d25d3a68cc.pack
Loose objects

$ git gc
Counting objects: 636, done.
Compressing objects: 100% (635/635), done.
Writing objects: 100% (636/636), done.
Total 636 (delta 486), reused 0 (delta 0)
Removing duplicate objects: 100% (256/256), done.

▶ GC happens automatically when you get to 6700 objects (configurable)
Loose objects

$ git gc
Counting objects: 636, done.
Compressing objects: 100% (635/635), done.
Writing objects: 100% (636/636), done.
Total 636 (delta 486), reused 0 (delta 0)
Removing duplicate objects: 100% (256/256), done.

▶ GC happens automatically when you get to 6700 objects (configurable)
Stash
Stashing

- you’re hacking

- got debug code you don’t want to commit

- need to work on something else
Stashing

- you’re hacking
- got debug code you don’t want to commit
- need to work on something else
Stashing

- you’re hacking
- got debug code you don’t want to commit
- need to work on something else
Stashing

- `git stash "description"`
- `do that other thing`
- `git stash apply`
Stashing

- `git stash "description"`

- do that other thing

- `git stash apply`
Stashing

- `git stash "description"`

- do that other thing

- `git stash apply`
Further reading

git.or.cz

git.or.cz/gitwiki

GitCasts.com
(Scott Chacon)
Further reading

- git.or.cz
- git.or.cz/gitwiki

GitCasts.com
(Scott Chacon)
Further reading

*git.or.cz*
*git.or.cz/gitwiki*

*GitCasts.com*
*(Scott Chacon)*
Further reading

http://www.jukie.net/~bart/blog/

git://tachyon.jukie.net/intro-to-git.git/
Further reading

http://www.jukie.net/~bart/blog/

git://tachyon.jukie.net/intro-to-git.git/
Thank you.