Transporting Changes

Bundles

Bundles are binary compressed archives that contain a series of commits. This format can be easily transmitted via email or USB stick.

Creating a bundle

Create a bundle from a range of commit treeish

```
git bundle create catchup.bundle HEAD~8..HEAD
```

Create a bundle from a range of time

```
git bundle create catchup.bundle --since=10.days master
```

Create a bundle from an entire branch

```
git bundle create catchup.bundle master
```

Visualize the Bundle

Show contents of a bundle

```
git ls-remote catchup.bundle
```

Or just the HEADs

```
git bundle list-heads catchup.bundle
```

Verify the Bundle

```
git bundle verify catchup.bundle
```

Retrieve the Bundle Contents

Pull in the blobs from a bundle as if they were a remote, but don't merge them. We'll retrieve them to a new local branch that represents the bundle.

```
git fetch catchup.bundle master:catchupmaster
```

Or start a new repo named `temprepo` via `clone` with the contents of the bundle put into the `master` branch

```
git clone catchup.bundle -b master temprepo
```

Merge in the Remote Contents

```
git merge catchupmaster master
```

Reference

ProGit section on `git bundle`

Patching

Build the Patch

Build an email that contains the patch

```
git email-patch
```

or to generate a patch file for every file on the current branch that differs from the master branch

```
git format-patch master
```

or reroute it all to a unified file

```
git format-patch master --stdout > myfix.patch
```

Visualize the Patch

To view the contents of the patch before applying it

```
git apply --stat myfix.patch
```

Test the patch application for conflicts

```
git apply --check myfix.patch
```

Apply a Patch

```
git apply myfix.patch
```

or using apply-mailbox to apply a series of patches

```
git am -3 myfix.patch
```

and if any conflicts are encountered, to continue the process

```
git am --resolved
```

or to sign off on the patch using your credentials

```
git am --signoff
```