GitLab

Git LFS Deep Dive - CREATE

Create Deep Dive

Francisco Javier López - Senior Backend Engineer

April 3, 2019

Table of Contents

Git and Binary files

- Git doesn’t track binary files (audio, video, or image file) the same way it does with text files
- A change in an binary file requires a new copy of the file in the repository
- It will make the history grow bigger and bigger
- Over time, this will decrease the speed to perform regular operations: clone, fetch, or pull

What is Git LFS?

- Open source project
- Git extension that provides some tools to handle LFS files
- Sets the specification for LFS clients and servers
- Replaces binary files with (text) pointers
- Introduces the concept of LFS Server
  - Pointers -> Git repository
  - Binary files -> LFS Server

Git LFS Pointers

```
version https://git-lfs.github.com/spec/v1
oid sha256:42c3dd42a403e9b474b4bab7f543a8dc92356b74829a009c36588acf7f3b79ea
size 1876
```

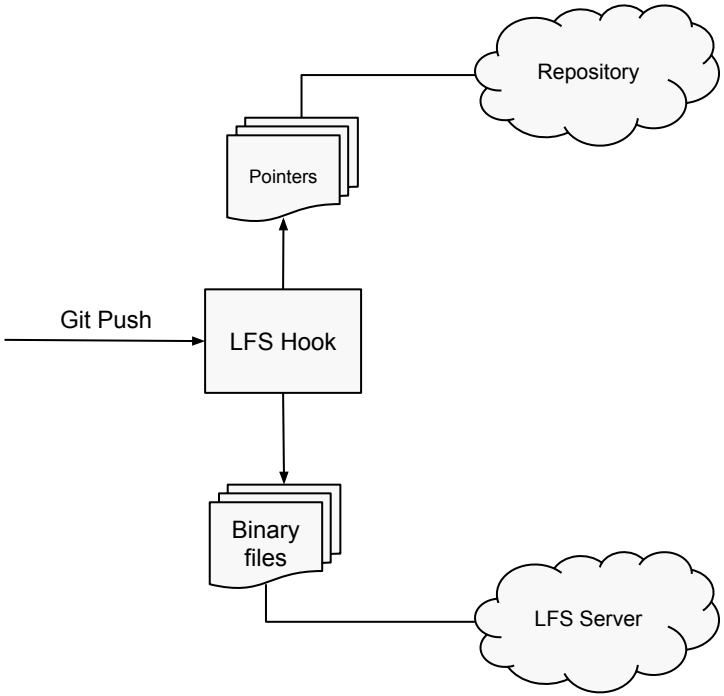
- **Version:** URL that identifies the file spec
- **Oid:** hashed unique identifier of the file
  - *Sha256* is the only one supported at the moment
  - Identical files get always the same oid
- **Size:** file size in bytes

How Git LFS works

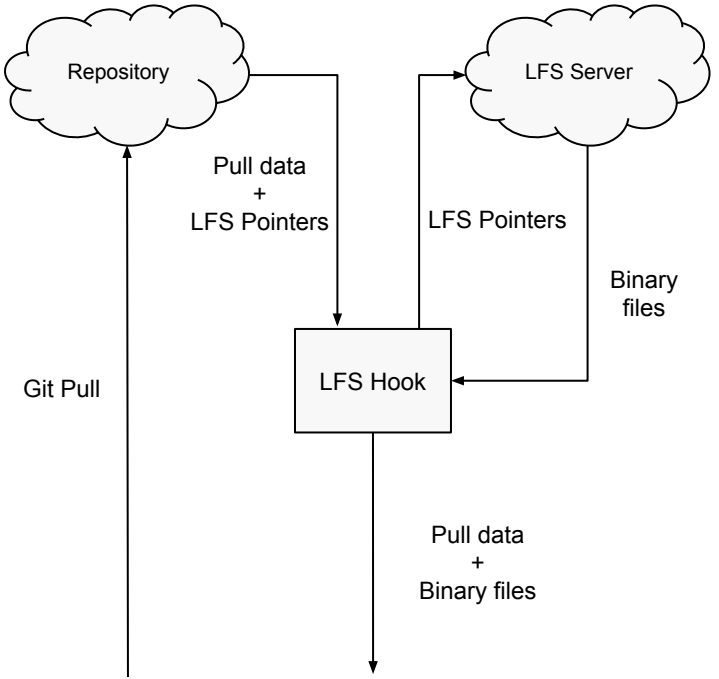
- The entry *lfs* is added to the repository config file
  - This entry stores the URL of the LFS server
  - By default the Git repository url will be used
- You select which files to track, ie. *git lfs track "*.png"*
  - Only works for new files.
  - To track existing files in the repository: *git lfs migrate*
  - A new file *.gitattributes* is added to the repository
- Also provides file locking capabilities
- How does it handle this process?
  - Through Git hooks that executes Git LFS commands under the hood

How Git LFS works (II)

Git Push



Git Pull



Authentication

- Git LFS uses HTTP Basic Authentication
  - For security reasons HTTPS is encouraged
- Where do these credentials come from?
  - From the Git remote or LFS url
  - Git credentials
  - When the repository remote is SSH
    - SSH connects to the repository
    - Command *git-lfs-authenticate* (handled by Gitlab Shell)
      - Gitlab Shell connects to the internal API (*/lfs_authenticate*)
      - A token is created with an expiration time inside
    - The following header with token is sent to the user

```
{
  "href": "https://gitlab.com/gitlab-org/foo",
  "header": {
    "Authorization": "Created token"
  },
  "expires_in": 1800
}
```

Authentication (II)

- The authentication is handled by *Projects::GitHttpClientController*
  - Checks if the authorization header is set
  - Calls *Gitlab::Auth.find_for_git_client*
    - Iterate over different authentication methods trying the login and password from the header
    - Returns a *Gitlab::Auth::Result* with the result

Batch API

Endpoints

- **POST info/lfs/objects**
  - Used to request the ability to transfer LFS files
    - If the user is not authorized no information about the LFS files will be sent
  - If everything ok, then the transfer will be through a different endpoint
  - Used for both uploading and downloading
  - Necessary headers:

```
Accept: application/vnd.git-lfs+json
Content-Type: application/vnd.git-lfs+json
```

Batch API (II)

Request

```
{
  "operation": "upload",
  "transfers": [ "basic" ],
  "ref": { "name": "refs/heads/master" },
  "objects": [
    {
      "oid": "11111111",
      "size": 5,
    }
  ]
}
```

- **Operation:** *upload* / *download*
- **Transfers:** List of the client transfer adapters (only supported *basic* at the moment)
- **Ref:** optional
- **Objects:** Array of LFS objects

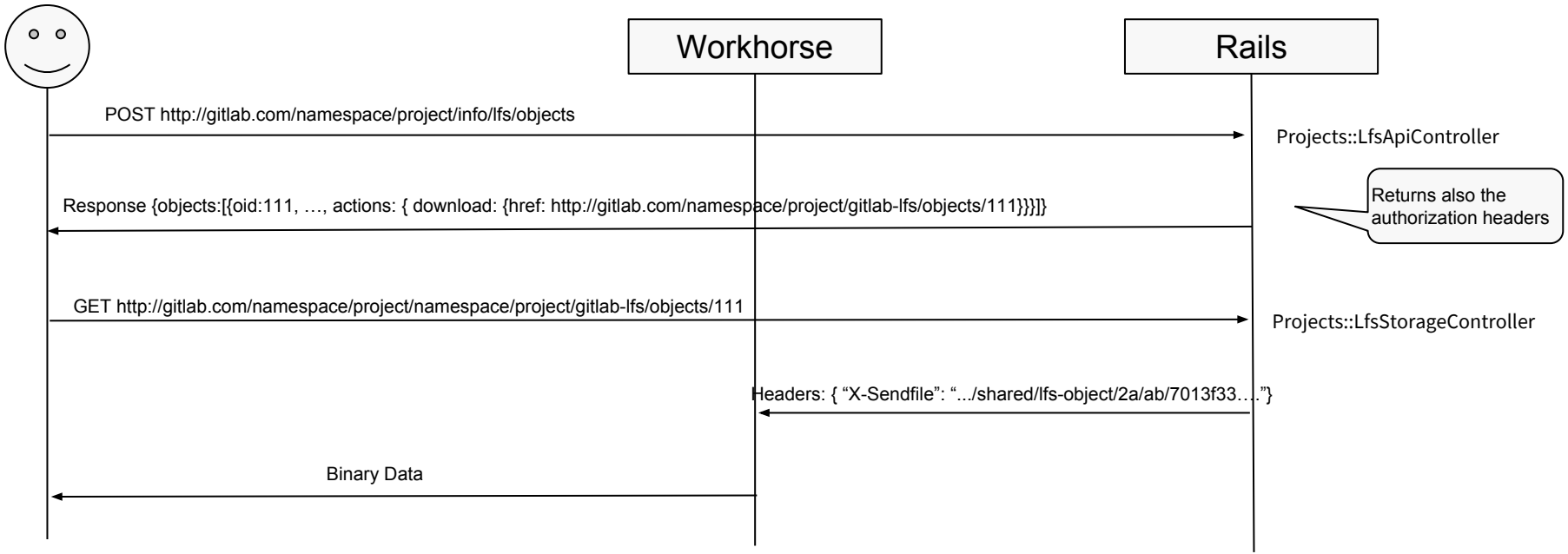
Batch API (III)

Response

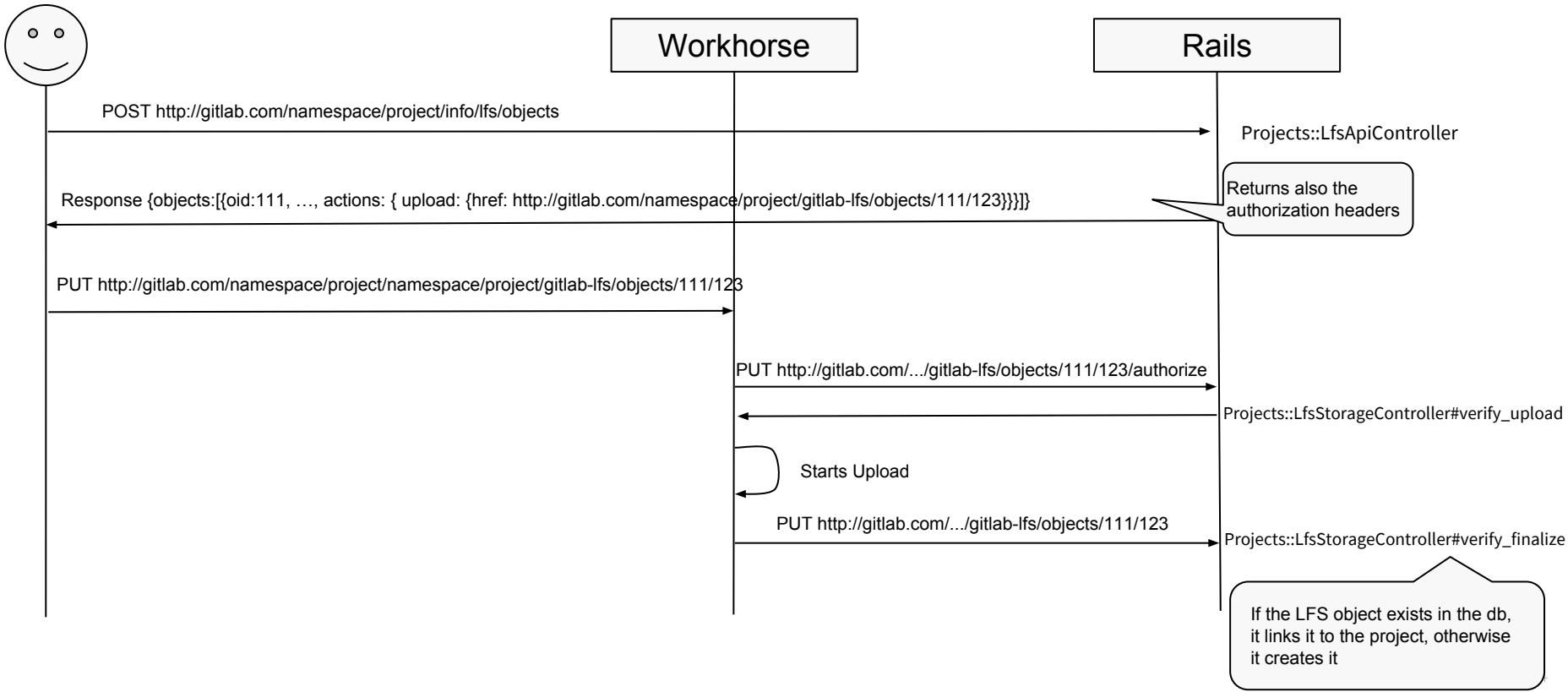
```
{
  "transfer": "basic",
  "objects": [
    {
      "oid": "1111111",
      "size": 5,
      "authenticated": true,
      "actions": {
        "download": {
          "href": "https://gitlab.com",
          "header": {
            "Authorization": "1234"
          },
          "Expires_at": "2019-04-03T11:16:07Z",
        }
      }
    }
  ]
}
```

- **Transfer:** Client transfer adapter. Same from the request
- **Objects:** List of objects:
  - **Oid**
  - **Size**
  - **Authenticated**: indicates whether the request for the object is authenticated.
  - **Actions**:
    - **Operation:** *upload*/*download*
    - **HRef:** URL where the LFS file can be accessed
    - **Header:** Optional hash to apply to the request
    - **Expires_in / Expires_at:** indicates when then transfer will expire

Batch API (IV). Gitlab LFS Downloads



Batch API (IV). Gitlab LFS Uploads



File Locking API

Endpoints

- **POST info/lfs/locks**
  - Creates a lock
  - Note: this is the first version of this API, so only single branch locking is supported
- **GET info/lfs/locks**
  - List all the locks
- **POST info/lfs/locks/:id/unlock**
  - Allows to remove locks
  - Through this endpoint locks from other uses can be removed if the param `force=true`
    - We ensure that only project maintainers can remove locks set by other users

File Locking API (II). Verify Locks

- **POST info/lfs/locks/verify**
  - Used to check if any existing lock can affect a Git push

```
Locking support detected on remote "origin". Consider enabling it with:

$ git config lfs.http://gitlab.com/user/project.git/info/lfs.locksverify true
```

- The response is splitted into *ours* and *theirs*
  - *Ours:* locks created by the user that makes the request
  - *Theirs*: locks owned by other users
- When pushing:
  - If any of the files matches any of the locks of the user (*ours*)
    - The locks will be listed at the end of the push
    - Git push succeeds
  - If any of the files matches any of the locks of the other users (*theirs)*
    - Git push halts

File Locking API (III). Verify Locks

Request

```
// POST https://gitlab.com/locks/verify
{
  "cursor": "optional lock cursor",
  "limit": 20,
  "ref": {
    "name": "refs/heads/master"
  }
}
```

- All params are optional

Response

```
{
  "ours": [
    {
      "id": "example-uuid",
      "path": "/foo/image.png",
      "locked_at": "2019-04-03T12:35:00+00:00",
      "owner": {
        "name": "User Name"
      }
    }
  ],
  "theirs": [],
  "next_cursor": "next lock ID",
}
```

Code Dive

Questions?

Thank you

Francisco Javier López - Senior Backend Engineer
fjlopez@gitlab.com