How GitLab Works
Saito

@saitowu

github.com/saitowu
GitLab?
GitLab?

Self Hosted Git Management Application
Projects (6)

GitLab HQ
Last activity: 7 minutes ago

gitlab-recipes
Last activity: about 22 hours ago

GITLAB.wiki
Last activity: about 22 hours ago

gitlab_meta
Last activity: about 22 hours ago

blog.gitlabhq.com
Last activity: about 13 days ago

GitLab Site
Last activity: about 1 month ago

News Feed

Homepage  Blog  @gitlabhq
happy to use it.
happy to use it.
want to know how it works?
repo access

backend
Backend
Can be backend:
Can be backend: protocol
Can be backend: protocol

file => (RW)
git => (R)
ssh => (RW)
http => (RW)
Can be backend: protocol

file => (RW)    x

file => (R)     x

ssh => (RW)

http => (RW)
Can be backend: protocol

file => (RW)  x  #=> remote
git => (R)    x  #=> authentication
ssh => (RW)   
http => (RW)
SSH:
SSH:

normal: authorized_keys
SSH:

normal: authorized_keys
SSH:
SSH:

gitosis/gitolite: authorized_keys
SSH:

gitosis/gitolite: authorized_keys

SSH magic command
SSH:

```
command="/var/git/gitolite/src/gitolite-shell gitlab", no-port-forwarding, no-X11-forwarding, no-agent-forwarding, no-pty
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAAGQC1QXCUnXUDKBFnumI9Pw1kJQ8rI5qYUvwTkrXpLJvjd/vukKaqNK7SWVECnfj6LGr5p054GwuD4+yR5S1LNjxq0kLhhEvCHLMgJ5C8Yzw8t74v+6gGXe0wMN9gsmAURktyje/iGb2j10sAy9LaQwhGzGTfduAIkfoe3cm7hw== vagrant@precise32
```

gitosis/gitolite: authorized_keys

SSH magic command
SSH:
SSH:

sshd: not authorized_keys
SSH:

sshd: not authorized_keys

but patched sshd
SSH:

sshd: not authorized_keys

but patched sshd
SSH: sshd: not authorized_keys

but patched sshd
SSH:

sshd: not authorized_keys

but patched sshd

norbauser / openssh-for-git

Twisted <= instead of sshd
SSH:

GitLab v1.0: gitosis #=> python
SSH:

GitLab v1.0: gitosis #=> python

GitLab v2.0: gitolite #=> perl
HTTP:
HTTP:

before Git v1.6.6: dumb
HTTP:

before Git v1.6.6: dumb

after Git v1.6.6: smart!!!
HTTP:

```ruby
SERVICES = [
  ["POST", 'service_rpc', "(.*+)/git-upload-pack$", 'upload-pack'],
  ["POST", 'service_rpc', "(.*+)/git-receive-pack$", 'receive-pack'],
  ["GET", 'get_info_refs', "(.*+)/info/refs$"],
  ["GET", 'get_text_file', "(.*+)/HEAD$"],
  ["GET", 'get_text_file', "(.*+)/objects/info/alternates$"],
  ["GET", 'get_text_file', "(.*+)/objects/info/http-alternates$"],
  ["GET", 'get_info_packs', "(.*+)/objects/info/packs$"],
  ["GET", 'get_text_file', "(.*+)/objects/info/[\^/]*$"],
  ["GET", 'get_loose_object', "(.*+)/objects/[0-9a-f]{2}/[0-9a-f]{38}$"],
  ["GET", 'get_pack_file', "(.*+)/objects/pack/pack-[0-9a-f]{40}\packs$"],
  ["GET", 'get_idx_file', "(.*+)/objects/pack/pack-[0-9a-f]{40}\idx$"],
]
```

HTTP:

git-upload-pack #=> send

git-receive-pack #=> receive
HTTP: smart

git-upload-pack #=> send

git-receive-pack #=> receive
HTTP:

GitLab v2.7: smart HTTP protocol
Repo
Access
Grit: command-line wrapper
Rugged:
ruby bindings to libgit2
Grit vs Rugged
Grit vs Rugged

win
Grit vs Rugged

lose  win
Grit vs Rugged

lose win

github will migrate from grit to rugged too
Repo
Browser
Browser
Must Know:
Must Know:

Git object model
Must Know:

Git object model

Git encoding strategy
Must Know:

Git object model
Git encoding strategy
Ruby encoding
Git Object Model:
Git Object Model:

4 basic objects: blob tree commit tag
Git Object Model:

4 basic objects: blob tree commit tag

stored in: .git/objects
Git Object Model:
Git Object Model:

git show <sha>
Git Object Model:

git show `<sha>`
git cat-file -p `<sha>"
Git Object Model:

helloworld git:master  >  git show 7394b8cc9ca
tree 7394b8cc9ca

readme.md
helloworld git:master  >  git cat-file -p 7394b8cc9ca
100644 blob 3b18e512dba79e4c8300dd08aeb37f8e728b8dad  readme.md

http://saito.im/note/Git-Internals/
Git Encoding Strategy:

At the core level, git is character encoding agnostic.

http://www.kernel.org/pub/software/scm/git/docs/v1.5.0.2/git-log.html#_discussion
Git Encoding Strategy:
Git Encoding Strategy:

filename
Git Encoding Strategy:

filename

blob contents
Git Encoding Strategy:

- filename
- blob contents
- commit message
Git Encoding Strategy:

filename  #=>  tree blob
blob contents
commit message
Git Encoding Strategy:

filename  #=>  tree blob
blob contents  #=>  blob
commit message
Git Encoding Strategy:

filename  #=>  tree blob

blob contents  #=>  blob

commit message  #=>  commit tag
Git Encoding Strategy:
Git Encoding Strategy:

no need #=> blob contents
Git Encoding Strategy:

no need $\Rightarrow$ blob contents

need $\Rightarrow$ commit message, filename
Git Encoding Strategy:

IMHO: `#=>` hg hybrid strategy & svn

no need `#=>` blob contents

need `#=>` commit message, filename
Ruby Encoding:
detect encoding:
Ruby Encoding:
detect encoding: the longer the better
Ruby Encoding:

detect encoding: the longer the better

filename
Ruby Encoding:

detect encoding: the longer the better

filename
blob contents
Ruby Encoding:

detect encoding: the longer the better

filename
blob contents
commit message
Ruby Encoding:

detect encoding: the **longer the better**

filename  #=> too short

blob contents
c North message
Ruby Encoding:

detect encoding: the **longer the better**

filename  #=> **too short**

blob contents  #=> **good to detect**

commit message
Ruby Encoding:

detect encoding: the longer the better

filename  #=> too short
blob contents  #=> good to detect
commit message  #=> not long enough
Ruby Encoding:

Character encoding detection:
Ruby Encoding:
Character encoding detection:
charlock_holmes
Ruby Encoding:

Character encoding detection:

charlock_holmes

rchardet19
Ruby Encoding:
Character encoding detection:
charlock_holmes #=> libicu
rchardet19
Ruby Encoding:

Character encoding detection:

charlock_holmes #=> libicu
rchardet19  #=> pure ruby
Ruby Encoding:
Ruby Encoding:

blob contents:
Ruby Encoding:

blob contents:

filename and commit message:
Ruby Encoding:

blob contents: detect and encode

filename and commit message:
Ruby Encoding:

blob contents: detect and encode
filename and commit message: hard to deal with
Ruby Encoding:
Ruby Encoding:

CSI
Ruby Encoding:

CSI  #=>  Character Set Independent
Ruby Encoding:

CSI  #=>  Character Set Independent

USC
Ruby Encoding:

CSI  =>  Character Set Independent

USC  =>  Universal Character Set
Ruby Encoding:

CSI  #=> Character Set Independent

 #=> Ruby

USC  #=> Universal Character Set
Ruby Encoding:

CSI => Character Set Independent

Ruby

USC => Universal Character Set

Java Python Perl
Ruby Encoding:
Ruby Encoding:

MRI 1.8:
Ruby Encoding:

MRI 1.8: defective encoding
Ruby Encoding:

MRI 1.8: defective encoding

MRI 1.9:
Ruby Encoding:

MRI 1.8: defective encoding

MRI 1.9: Encoding Class
Ruby Encoding:

MRI 1.8: defective encoding
MRI 1.9: Encoding Class
pros: regexp length
Ruby Encoding:

MRI 1.8: defective encoding

MRI 1.9: Encoding Class

pros: regexp length

cons: incompatible character encodings
Hooks
Post-receive:
Post-receive:

push event to redis
Post-receive:

push event to redis
resque it
Post-receive:

scalability : Web Hooks
Post-receive:

interoperability : API
Post-receive:

interoperability : API

based on grape
More
Git: pull-request
Git: pull-request

git format-patch <br>..<br>
Git: pull-request

git format-patch <br>..

git apply --check <patch>
Git: pull-request

`git format-patch <br>..<br>`

`git apply --check <patch>`

`git am <patch>`
Git: cherry-pick

`git cherry-pick <sha>`
One
More
Thing
Gitlab CI
Thanks