Git Server from Scratch in Go

Using awesome gliderlabs/ssh library

… in 7 minutes of less

GopherCon 2025
@iefserge

gliderlabs/ssh

README BSD-3-Clause license

gliderlabs/ssh

> The Glider Labs SSH server package is dope. —@bradfitz, Go team member

This Go package wraps the crypto/ssh package with a higher-level API for building SSH servers. The goal of the API was to make it as simple as using net/http, so the API is very similar:

```
package main

import (
    "github.com/gliderlabs/ssh"
    "io"
    "log"
)

func main() {
    ssh.Handle(func(s ssh.Session) {
        io.WriteString(s, "Hello world\n")
    })

    log.Fatal(ssh.ListenAndServe(":2222", nil))
}
```

@iefserge

Sergii Iefremov

AXIOS

Gitpatch

Basic SSH server can we run git clone on it?

```
1  package main
2
3  import (
4          "fmt"
5          "log"
6
7          "github.com/gliderlabs/ssh"
8  )
9
10 func main() {
11         ssh.Handle(func(s ssh.Session) {
12                 fmt.Printf("Running: %s\n", s.RawCommand())
13         })
14
15         log.Fatal(ssh.ListenAndServe(":2222", nil,
16                 ssh.HostKeyFile("gitssh_host_key"), ssh.NoPty()))
17 }
```

Git clone

- 1) git clone ssh://git@localhost:2222/demo
- 2) git clone git@localhost:demo (on port 22)

```
/Volumes/dev/gitssh-demo ❯ go run main.go
Running: git-upload-pack '/demo'
```

NAME

git-upload-pack - Send objects packed back to git-fetch-pack

SYNOPSIS

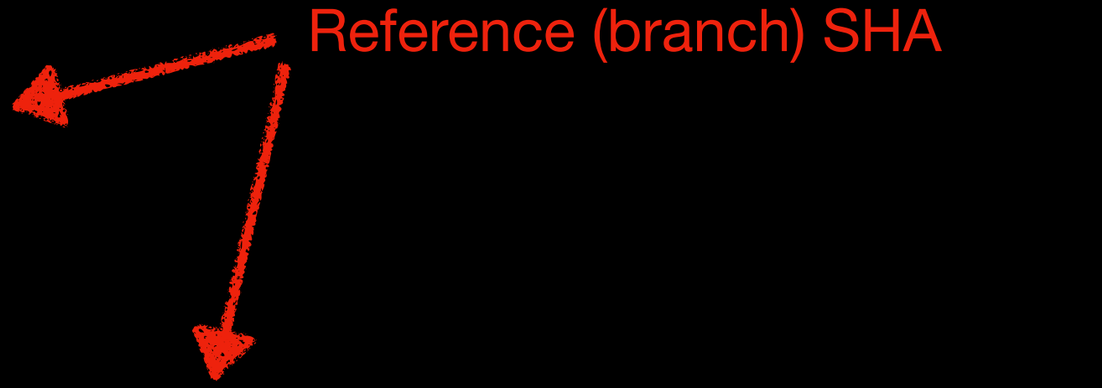
```
git-upload-pack [--[no-]strict] [--timeout=<n>] [--stateless-rpc]
                [--advertise-refs] <directory>
```

DESCRIPTION

Invoked by **git fetch-pack**, learns what objects the other side is missing, and sends them after packing.

This command is usually not invoked directly by the end user. The UI for the protocol is on the **git fetch-pack** side, and the program pair is meant to be used to pull updates from a remote repository. For push operations, see **git send-pack**.

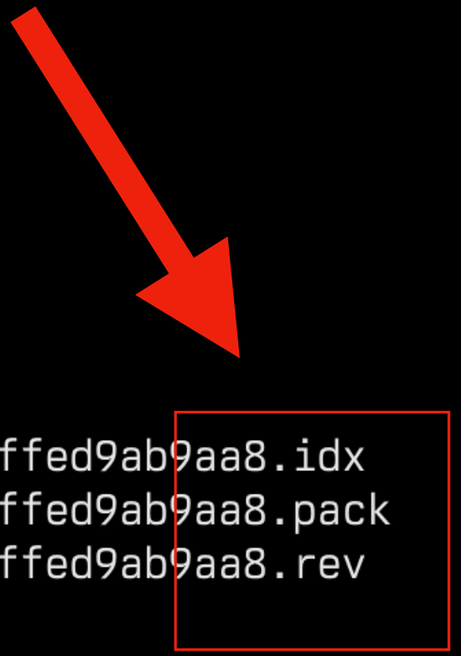
```
6           "io"
7           "log"
8           "os"
9
10          "github.com/gliderlabs/ssh"
11  )
12
13  func pkt(s string) string {
14          return fmt.Sprintf("%04x%s\n", len(s)+5, s)
15  }
16
17  func main() {
18          ssh.Handle(func(s ssh.Session) {
19                  ref, err := os.ReadFile(".git/refs/heads/main")
20                  if err != nil {
21                          s.Exit(1)
22                          return
23                  }
24                  ref = bytes.TrimRight(ref, "\n")
25
26                  io.WriteString(s, pkt(fmt.Sprintf("%s HEAD\x00object-format=sha1 symref=HEAD:refs/heads/main", ref)))
27                  io.WriteString(s, pkt(fmt.Sprintf("%s refs/heads/main", ref)))
28                  io.WriteString(s, "0000")
29          })
30
31          log.Fatal(ssh.ListenAndServe(":2222", nil,
32                  ssh.HostKeyFile("gitssh_host_key"), ssh.NoPty()))
33  }
```



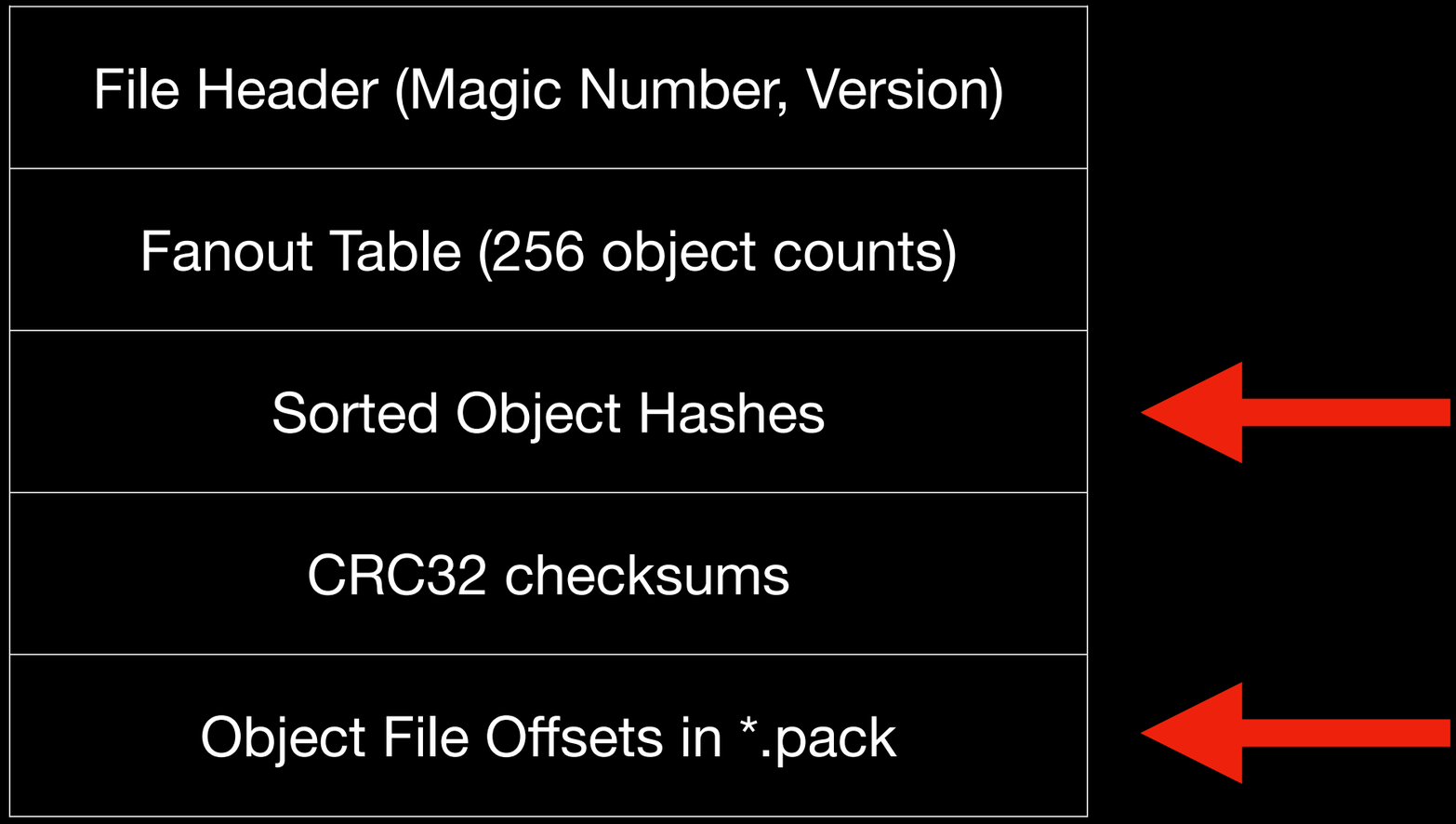
```
dev/tmp/tmp-demo-gitssh ❱ GIT_TRACE_PACKET=1 git clone git@localhost:demo
Cloning into 'demo'...
02:33:45.186357 pkt-line.c:85          packet:        clone< a823d20b5c6ce9db809af2f8133be4e0cb974bc4 HEAD\0o
02:33:45.186653 pkt-line.c:85          packet:        clone< a823d20b5c6ce9db809af2f8133be4e0cb974bc4 refs/he
02:33:45.186658 pkt-line.c:85          packet:        clone< 0000
02:33:45.188192 pkt-line.c:85          packet:        clone> want a823d20b5c6ce9db809af2f8133be4e0cb974bc4
02:33:45.188198 pkt-line.c:85          packet:        clone> want a823d20b5c6ce9db809af2f8133be4e0cb974bc4
02:33:45.188200 pkt-line.c:85          packet:        clone> 0000
```

Git Storage

```
gitssh-demo main* ❱ git repack -a
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 10 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), done.
Total 9 (delta 2), reused 9 (delta 2), pack-reused 0 (from 0)
gitssh-demo main* ❱ ls -l .git/objects/pack
total 24
-r--r--r--@ 1 se  staff  1324 Aug 20 21:19 pack-908d4c53a4f9e7583e625f7600c47ffed9ab9aa8.idx
-r--r--r--@ 1 se  staff  3013 Aug 20 21:19 pack-908d4c53a4f9e7583e625f7600c47ffed9ab9aa8.pack
-r--r--r--@ 1 se  staff    88 Aug 20 21:19 pack-908d4c53a4f9e7583e625f7600c47ffed9ab9aa8.rev
```



*.idx file


File Header (Magic Number, Version)
Fanout Table (256 object counts)
Sorted Object Hashes
CRC32 checksums
Object File Offsets in *.pack

func Index(out *error) iter.Seq2[string, idxObject]

```go
type IdxObject struct {
        Pack   string
        Offset int
}

func Index(out *error) iter.Seq2[string, IdxObject] {
        return func(yield func(string, IdxObject) bool) {
                entries, err := os.ReadDir(".git/objects/pack")
                if err != nil {
                        *out = fmt.Errorf("reading pack dir: %w", err)
                        return
                }
                fileIndex := slices.IndexFunc(entries, func(e os.DirEntry) bool { return strings.HasSuffix(e.Name(), ".idx") })
                if fileIndex < 0 {
                        return
                }
                data, err := os.ReadFile(".git/objects/pack/" + entries[fileIndex].Name())
                if err != nil {
                        *out = fmt.Errorf("reading idx file: %w", err)
                        return
                }
                hashOffsets := 256*4 + 8
                objectCount := int(binary.BigEndian.Uint32(data[hashOffsets-4 : hashOffsets]))
                packOffsets := hashOffsets + objectCount*20 + objectCount*4
                for i := range objectCount {
                        // Object hash.
                        hash := hex.EncodeToString(data[hashOffsets+i*20 : hashOffsets+(i+1)*20])
                        if !yield(hash, IdxObject{
                                // Idx file name.
                                Pack: ".git/objects/pack/" + strings.TrimSuffix(entries[fileIndex].Name(), ".idx") + ".pack",
                                // Location of this object in *.pack file.
                                Offset: int(binary.BigEndian.Uint32(data[packOffsets+i*4 : packOffsets+(i+1)*4])),
                        }) {
                                return
                        }
                }
        }
}
```

1. List files in .git directory

2. Open *.idx file

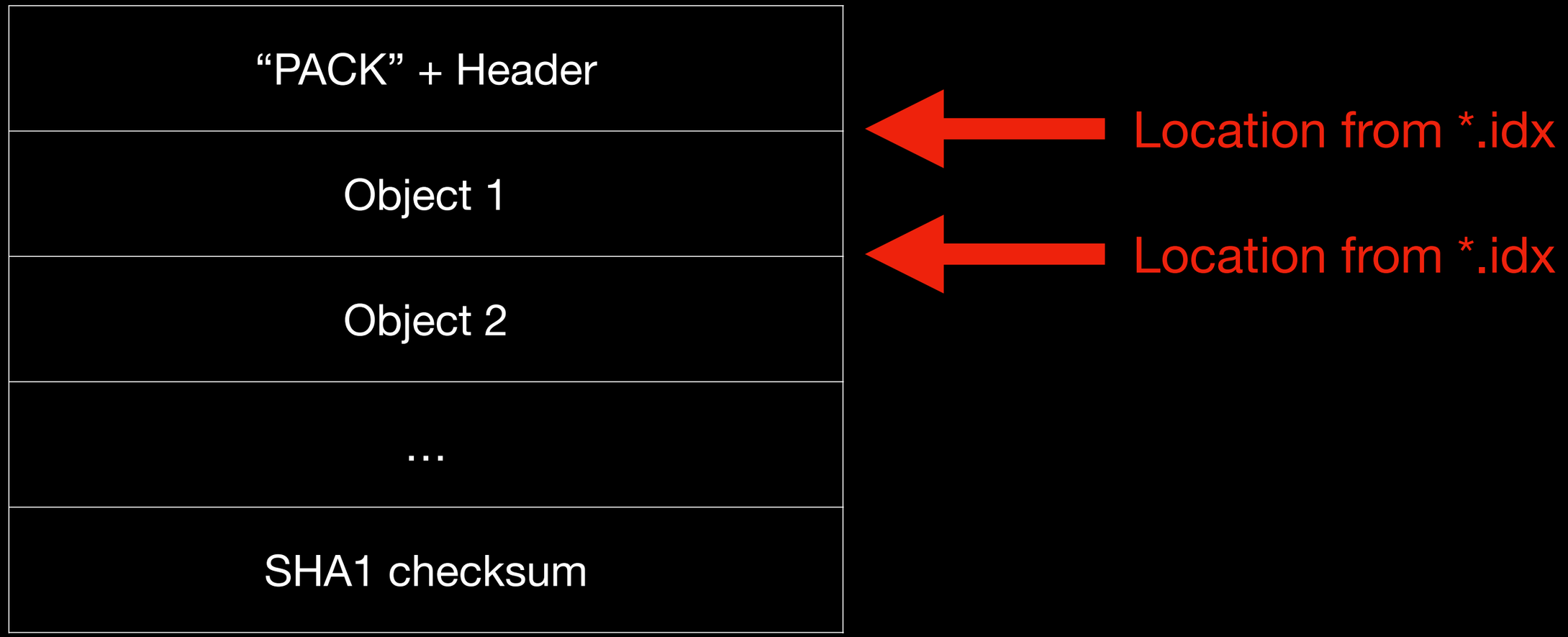
3. Iterate and yield objects [hash, obj]

```
20 func main() {
21         var err error
22         for hash, object := range githelpers.Index(&err) {
23                 fmt.Printf("%s at %d\n", hash, object.Offset)
24         }
25         if err != nil {
26                 slog.Error("failed reading objects", "error", err)
27         }
28 }
```

```
gitssh-demo main* ❱ go run main.go
457bd1fabf8c701a77df762553e24cada665ec11 at 2711
4ee01a0f8c61bdaaf71087d7401eb9ba92fd437d at 2849
695b935747ac2c82ac353080913310e063d48996 at 2326
7a183d8f4919d977b018ef849a4338c2e6a24031 at 2889
882a897d144ee8e403a739be0ad408fbaa408a5f at 1635
889fee6a5705b6698e35c6bd808ea23481c33c80 at 830
a823d20b5c6ce9db809af2f8133be4e0cb974bc4 at 12
f3e3fb70fa15f6d29b1c94aa64434a298352aca1 at 1608
42a753c9c199dacc42a655dc1a901143ca7db351 at 1806
```

*.pack file

used for both storage and transfer



```
51              count := 0
52              packObject := map[string][]githelpers.IdxObject{}
53              for _, object := range githelpers.Index(&err) {
54                      packObject[object.Pack] = append(packObject[object.Pack], object)
55                      count++
56              }
57              if err != nil {
58                      s.Exit(1)
59                      return
60              }
61
62              h := sha1.New()
63              sh := io.MultiWriter(s, h)
64              sh.Write([]byte("PACK"))
65              binary.Write(sh, binary.BigEndian, uint32(2))
66              binary.Write(sh, binary.BigEndian, uint32(count))
67              for pack, objects := range packObject {
68                      if err := writeObjects(sh, pack, objects); err != nil {
69                              slog.Error("writing objects", "pack", pack, "error", err)
70                              s.Exit(1)
71                      }
72              }
73
74              s.Write(h.Sum(nil))
75      })
```

1. Checksum and MultiWriter

2. Write objects

```go
func writeObjects(w io.Writer, pack string, objects []githelpers.IdxObject) error {
	f, err := os.Open(pack)
	if err != nil {
		return fmt.Errorf("opening packfile: %w", err)
	}
	defer f.Close()

	slices.SortFunc(objects, func(a, b githelpers.IdxObject) int { return a.Offset - b.Offset })

	for _, object := range objects {
		sectionReader := io.NewSectionReader(f, int64(object.Offset), math.MaxInt64)
		reader := bufio.NewReader(sectionReader)

		// Read and copy object header.
		if _, err := githelpers.CopyObjectHeader(reader, w); err != nil {
			return fmt.Errorf("reading header: %w", err)
		}

		// Read and copy object body.
		zlibReader, err := zlib.NewReader(io.TeeReader(reader, w))
		if err != nil {
			return fmt.Errorf("creating zlib reader: %w", err)
		}

		// Read object and also write it into writer.
		if _, err := io.Copy(io.Discard, zlibReader); err != nil {
			return fmt.Errorf("copying object: %w", err)
		}
		zlibReader.Close()
	}

	return nil
}
```

1. Sort objects

2. Set up readers

3. zlib and TeeReader

4. Stream objects

```
dev/tmp/tmp-demo-gitssh ❯ git clone git@localhost:demo
Cloning into 'demo'...
fatal: pack has bad object at offset 3291: inflate returned -5
fatal: fetch-pack: invalid index-pack output
```

Issue with zlib + io.TeeReader

- zlib needs ReadByte()
  - **io.TeeReader()** doesn’t implement it

compress/flate/inflate.go

```
// The actual read interface needed by [NewReader].
// If the passed in [io.Reader] does not also have ReadByte,
// the [NewReader] will introduce its own buffering.
type Reader interface {
  io.Reader
  io.ByteReader
}
```

- Solution: replace **io.TeeReader()** -> custom **teeByteReader()**
  - Read(p []byte) (int, error)
  - ReadByte() (byte, error)

```
@@ -141,7 +136,7 @@ func writeObjects(w io.Writer, pack string, objects []githelpers.IdxObject) erro
                }

                // Read and copy object body.
-               zlibReader, err := zlib.NewReader(io.TeeReader(reader, w))
+               zlibReader, err := zlib.NewReader(&teeByteReader{zlibInputReader: reader, teeWriter: w})
                if err != nil {
                        return fmt.Errorf("creating zlib reader: %w", err)
                }
```

```
dev/tmp/tmp-demo-gitssh ❱ git clone git@localhost:demo
Cloning into 'demo'...
Receiving objects: 100% (33/33), 13.29 KiB | 156.00 KiB/s, done.
Resolving deltas: 100% (14/14), done.
```

```
dev/tmp/tmp-demo-gitssh ❱ ls demo
githelpers go.mod     go.sum     main.go
```

Full source on GitHub:

https://github.com/iefserge/gitssh-demo

Thank you!