Build Your Own Go CI Server

18:00 03 August 2017

Marwan Sulaiman - github.com/marwan-at-work

Developer at Work & Co

Overview

- What is CI/CD?
- Why do you want it?
- How do you do it?

What is Continuous Integration?

*[it] is the practice of merging all developer working copies to a shared *mainline* several times a day*

-Wikipedia

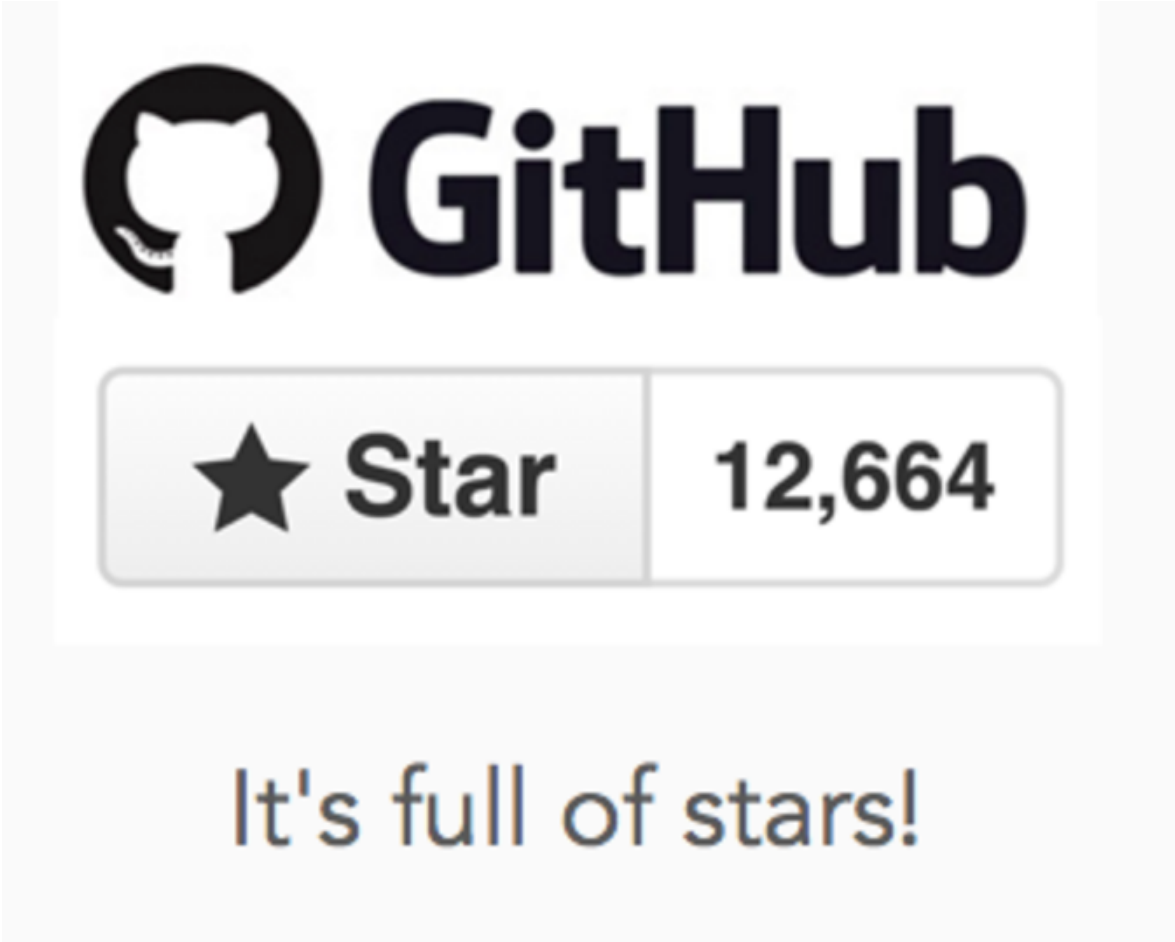
What is Continuous Integration?

it is automated building/testing of your repository's pull request so you merge with more confidence

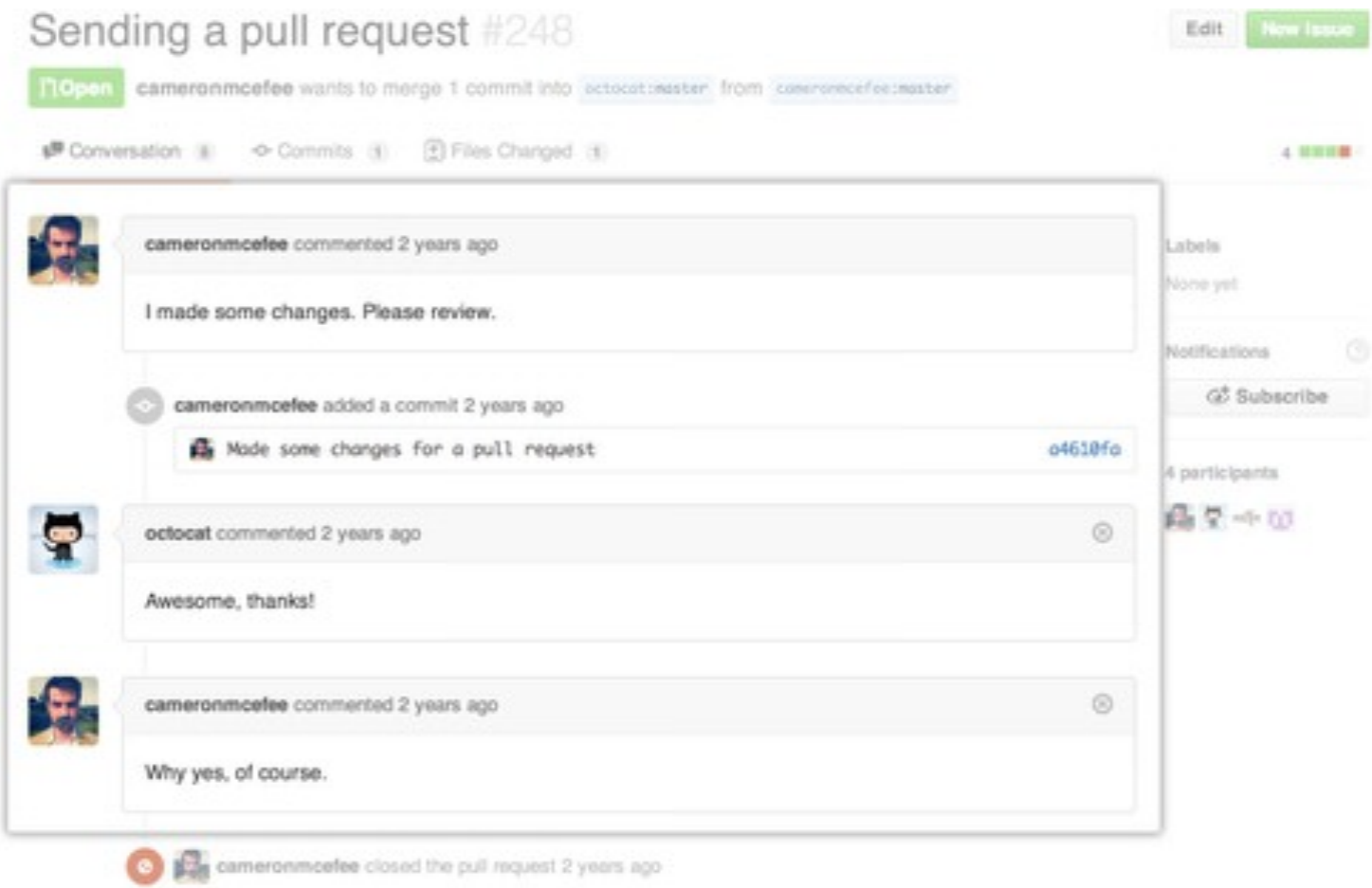
-Me

Set the Scene

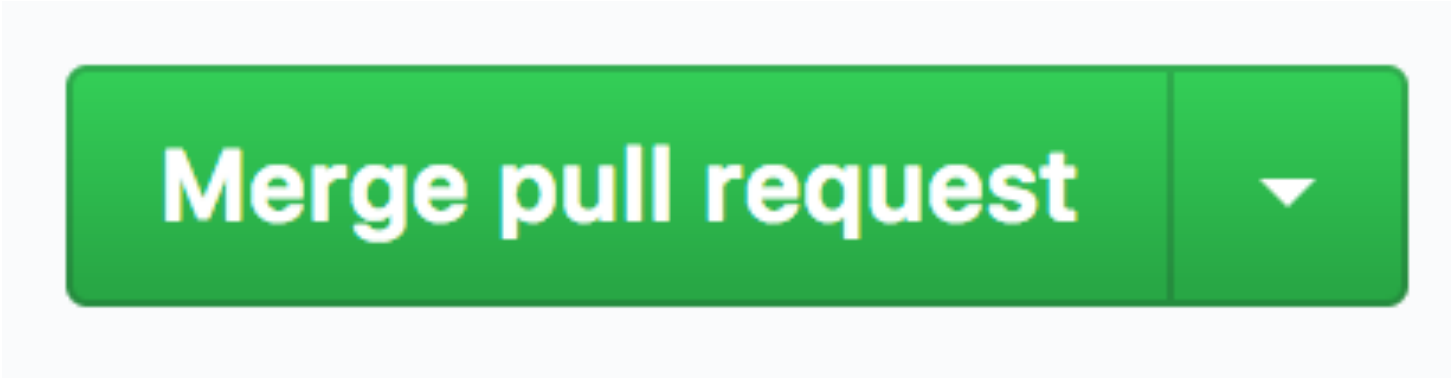
Your new repo just got popular



You get a pull request



Deceptively friendly green button alert:



Two things must happen before merge.

1. You review the code (done by you).

2. Check out the pull request commit, build/run the tests (done by someone/something else).

Like a good citizen, you review code.

```
230 +               if pkg.hasOption("tag") {
231 +                       rev, _ = pkg.options["tag"].(string)
232 +               }
233 +               if pkg.hasOption("commit") {
234 +                       rev, _ = pkg.options["commit"].(string)
235 +               }
236 +
237 +               var pc gps.ProjectConstraint
238 +               pi := gps.ProjectIdentifier{
239 +                       ProjectRoot: gps.ProjectRoot(pkg.name),
240 +               }
241 +
242 +               pc.Ident = pi
243 +
244 +               if rev != "" {
245 +                       pc.Constraint, err = g.sm.InferConstraint(rev, pc.Ident)
```

carolynvs 5 days ago Collaborator
When `revision == ""`, you should set the constraint to `gps.Any()`.

mattn 5 days ago **Member**
Ah, I didn't know gps.Any(). Okay, will do it in later.

carolynvs 5 days ago • edited Collaborator
After thinking about this more, I think the right answer is to do that in `InferConstraint`, so that when an empty string is passed, it returns `gps.Any()`. I'll submit a PR for that in a bit.

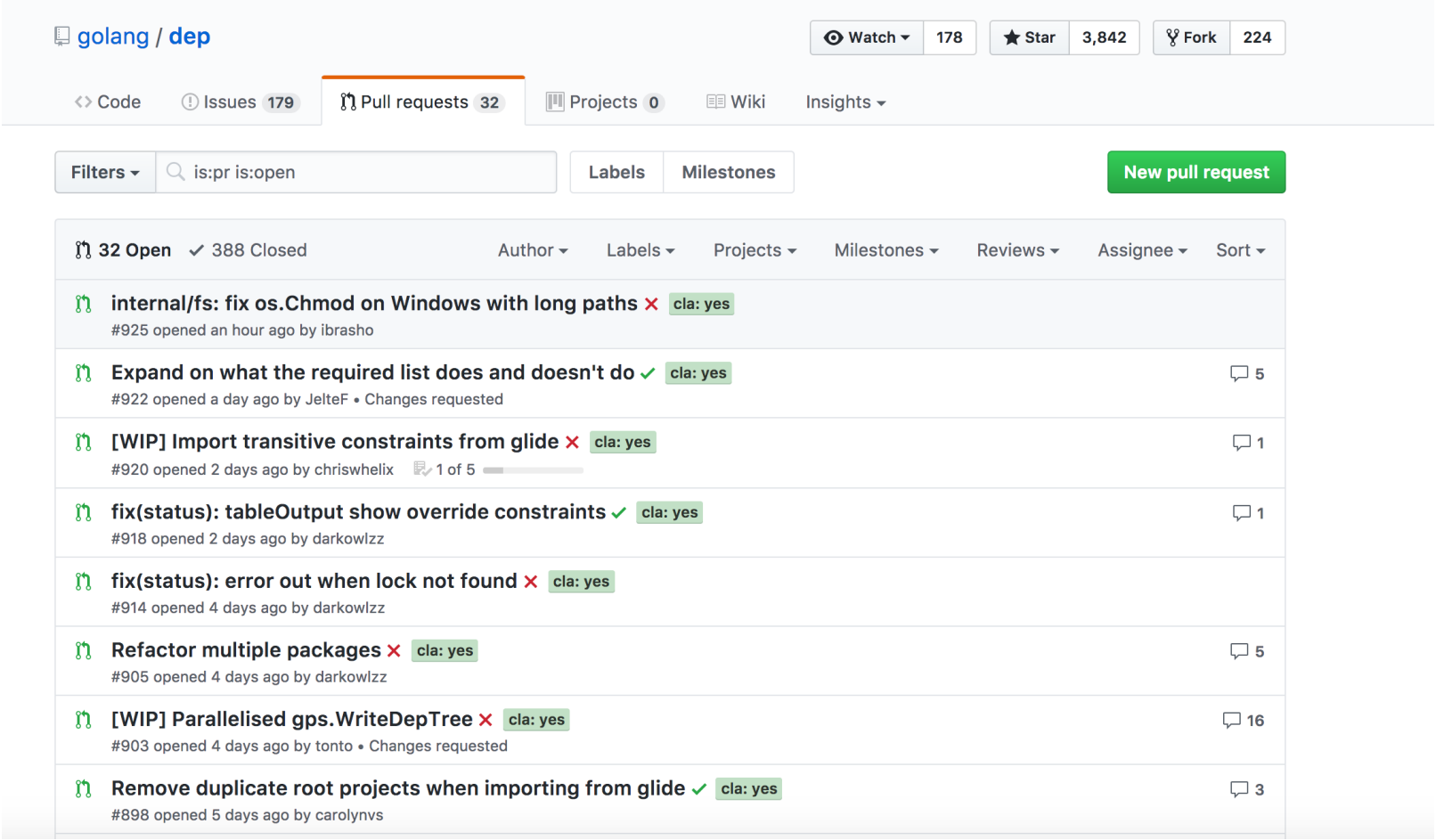
carolynvs 5 days ago Collaborator
I've submitted #901 so that all importers get this for free when calling `InferConstraint`.

Reply...

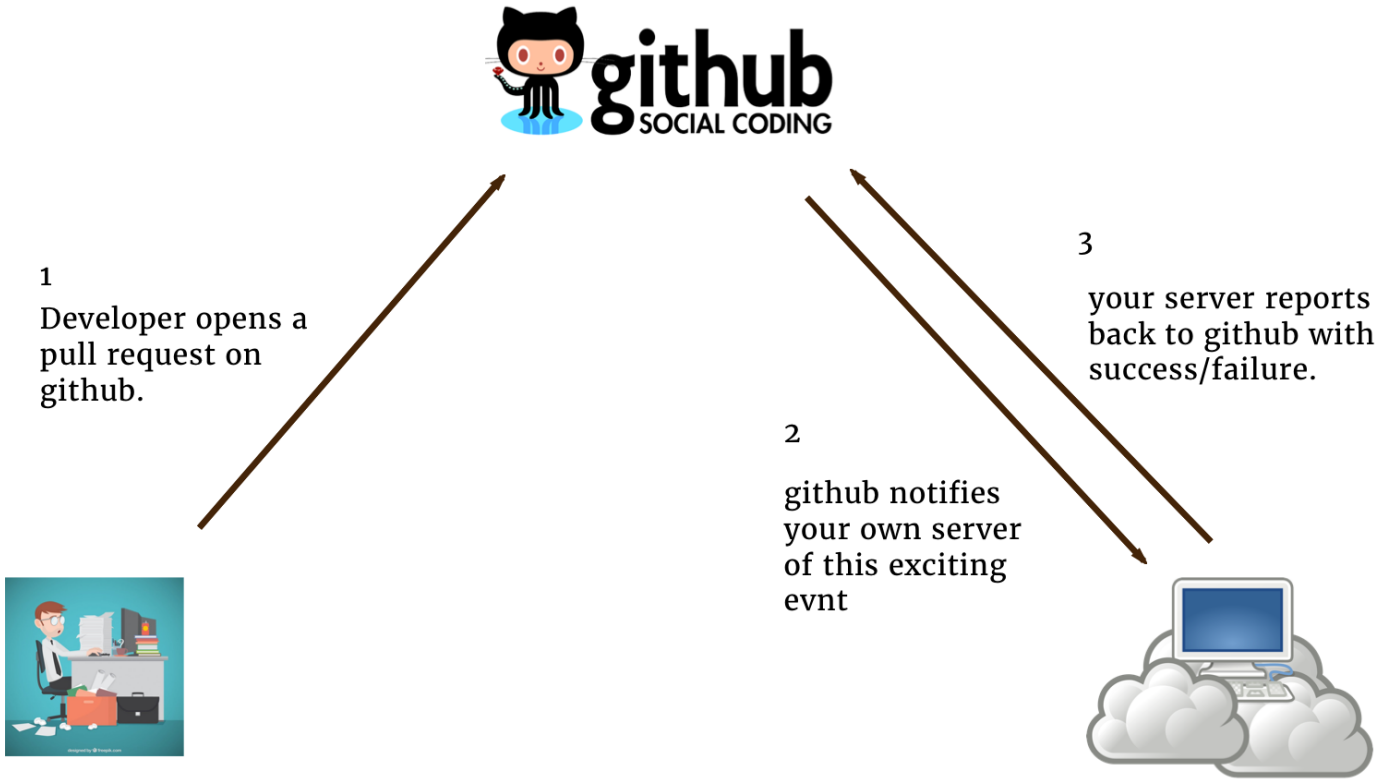
Start a new conversation

```
246 +                       if err != nil {
```

Someone/Something Else Running The Tests



The flow:


github
SOCIAL CODING
1
Developer opens a pull request on github.
2
github notifies your own server of this exciting evnt
3
your server reports back to github with success/failure.

Why have a CI?

- Impractical to manually build/test every pull request.
- Near-impossible to build/test in various environments (differnet Go versions, different OS's).
- Catch bugs, errors, and regressions before merging to master.
- Report useful information to the contributor (i.e. signing a CLA).

Why build your own CI?

- Customization
- Granularity
- Curiosity.

How do I do it?

Webhooks

This repository Search Pull requests Issues Marketplace Gist

marwan-at-work / gowatch

Unwatch 1 Star 0 Fork 1

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

- Options
- Collaborators
- Branches
- **Webhooks**
- Integrations & services
- Deploy keys

Webhooks / **Add webhook**

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *

https://www.my-ci-server.com/github-webhook

Content type

application/json

Secret

••••••••

By default, we verify SSL certificates when delivering payloads. Disable SSL verification

Which events would you like to trigger this webhook?

- Just the `push` event.
- Send me **everything**.
- Let me select individual events.

ngrok

```
./ngrok http 3000
```

```
ngrok by @inconshreveable

Session Status                online
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://bbe1f402.ngrok.io -> localhost:3000
Forwarding                    https://bbe1f402.ngrok.io -> localhost:3000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

CI Server

```
package main

import "net/http"

func main() {
    http.HandleFunc("/webhook", webhookHandler)

    http.ListenAndServe(":3000", nil)
}

func webhookHandler(w http.ResponseWriter, r *http.Request) {...}
```

Helper packages

```
// github client and webhook parser.
import "github.com/google/go-github/github"

// git implementation in Go
import "gopkg.in/src-d/go-git.v4"

// library to issue docker requests
import "github.com/docker/docker/client"
```

webhookHandler

```
import "github.com/google/go-github/github"

func webhookHandler(w http.ResponseWriter, r *http.Request) {

    // func ValidatePayload(r *http.Request, secretKey []byte) (payload []byte, err error)
    payload, err := github.ValidatePayload(r, []byte("supersecret"))
    if err != nil {
        fmt.Fprintln(w, err)
        return
    }

    ...
}
```

webhookHandler() - cont'd

```
func webhookHandler(w http.ResponseWriter, r *http.Request) {
    ...

    // func ParseWebHook(messageType string, payload []byte) (interface{}, error)
    event, err := github.ParseWebHook(github.WebHookType(r), payload)
    if err != nil {
        fmt.Fprintln(w, err)
        return
    }

    ...
}
```

webhookHandler() - cont'd

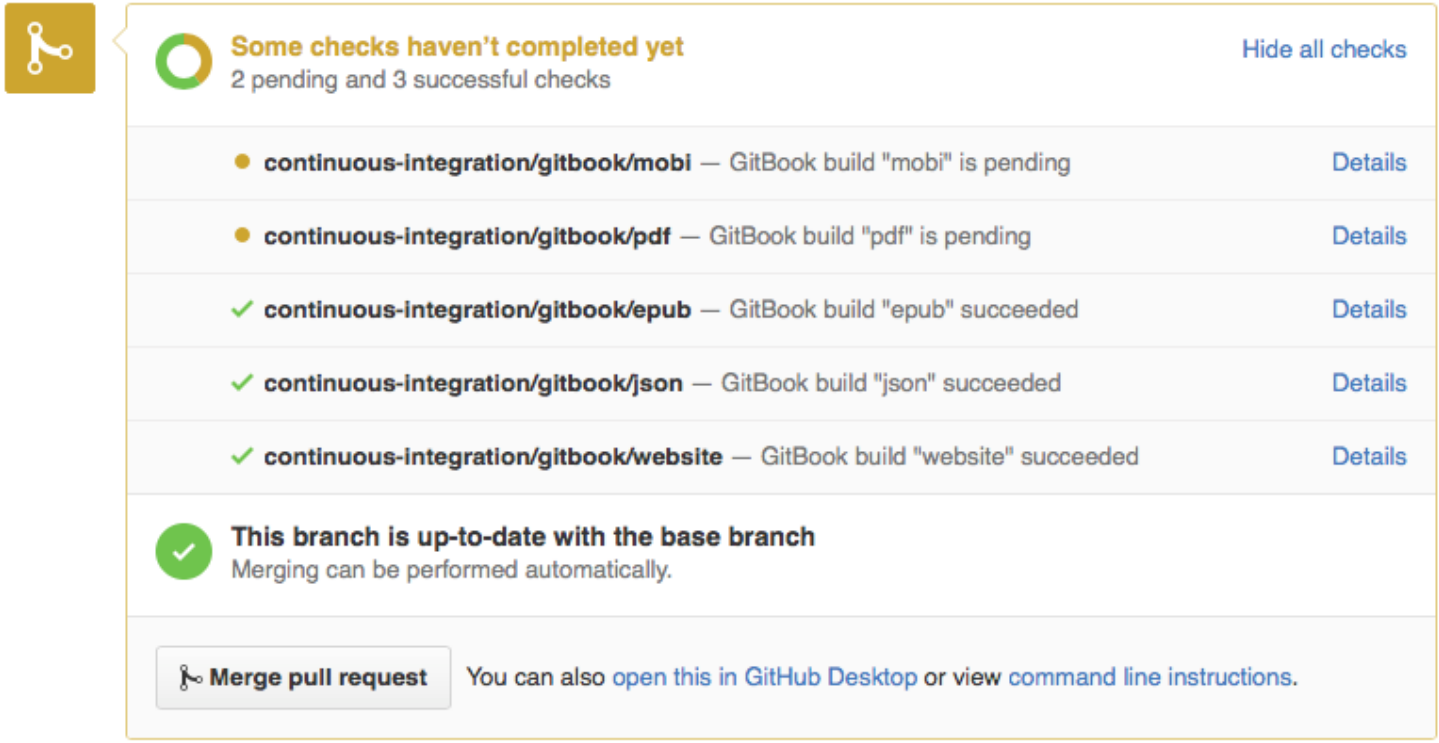
```
func webhookHandler(w http.ResponseWriter, r *http.Request) {
    ...

    switch pre := event.(type) {
    case *github.PullRequestEvent:
        action := pre.GetAction()
        if action == "opened" || action == "reopened" || action == "synchronize" {
            err = buildPR(pre)
            ...
        }
    default:
        fmt.Fprintf(w, "Uninteresting event: %T\n", event)
        return
    }
}
```

buildPR()

```
func buildPR(pre *github.PullRequestEvent) error {
    status := "pending"
    reportStatus(status, pre *github.PullRequestEvent)

    ...
}
```


Some checks haven't completed yet
2 pending and 3 successful checks
Hide all checks
continuous-integration/gitbook/mobi — GitBook build "mobi" is pending
Details
continuous-integration/gitbook/pdf — GitBook build "pdf" is pending
Details
continuous-integration/gitbook/epub — GitBook build "epub" succeeded
Details
continuous-integration/gitbook/json — GitBook build "json" succeeded
Details
continuous-integration/gitbook/website — GitBook build "website" succeeded
Details
This branch is up-to-date with the base branch
Merging can be performed automatically.
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.

reportStatus()

```
import "golang.org/x/oauth2"

func reportStatus(status string, pre *github.PullRequestEvent) {
    ts := oauth2.StaticTokenSource(&oauth2.Token{AccessToken: os.Getenv("GITHUB_TOKEN")})
    tc := oauth2.NewClient(oauth2.NoContext, ts)
    c := github.NewClient(tc)

    owner := pre.PullRequest.Head.Repo.Owner.GetLogin()
    repoName := pre.PullRequest.Head.Repo.GetName()
    ref := pre.PullRequest.Head.GetSHA()

    // func (s *RepositoriesService) CreateStatus(ctx context.Context, owner, repo, ref string, status
    c.Repositories.CreateStatus(context.Background(), owner, repoName, ref, &github.RepoStatus{
        State: &status,
    })
}
```

buildPR() - clone the repository

```
func buildPR(pre *github.PullRequestEvent) error {
    ...

    pathToRepo, err := cloneRepo(pre)
    if err != nil {
        return err
    }
    defer os.RemoveAll(pathToRepo)

    ...
}
```

cloneRepo() - create a temp directory

```
func cloneRepo(pre *github.PullRequestEvent) (string, error) {
    ...

    repoName := pre.PullRequest.Head.Repo.GetName()
    tempDir, err := ioutil.TempDir("", repoName) // tempDir == "/var/jXei/T/repoName-fxjso"
    if err != nil {
        return "", err
    }

    ...
}
```

cloneRepo() - clone the repo

```
import "gopkg.in/src-d/go-git.v4"
import "gopkg.in/src-d/go-git.v4/plumbing"

func cloneRepo(pre *github.PullRequestEvent) (string, error) {
    ...

    cloneURL := pre.PullRequest.Head.Repo.GetCloneURL()
    branchName := pre.PullRequest.Head.GetRef()

    repo, err := git.PlainClone(tempDir, false, &git.CloneOptions{
        URL:               gitURL,
        Progress:          os.Stdout,
        RecurseSubmodules: git.DefaultSubmoduleRecursionDepth,
        ReferenceName:     plumbing.ReferenceName(fmt.Sprintf("refs/heads/%v", branchName)),
    })
    if err != nil {
        return "", err
    }

    ...
}
```

cloneRepo() - checkout the pull-request commit

```
func cloneRepo(pre *github.PullRequestEvent) (string, error) {
    ...

    wt, err := repo.Worktree()
    if err != nil {
        return "", err
    }

    commitSha := pre.PullRequest.Head.GetSHA()
    err = wt.Checkout(&git.CheckoutOptions{
        Hash: plumbing.NewHash(commitSha),
    })
    if err != nil {
        return "", err
    }

    return tempDir, nil
}
```

buildPR() - build the Dockerfile inside the repo.

```
func buildPR(pre *github.PullRequestEvent) error {
    ...

    err = buildImage(pathToRepo)

    return err
}
```

buildImage() - create a docker client.

```
import docker "github.com/docker/docker/client"

func buildImage(repoDir string) error {
    c, err := docker.NewEnvClient()
    if err != nil {
        return err
    }

    ...
}
```

buildImage() - tar up the whole repo.

```
import "github.com/docker/docker/pkg/archive"

func buildImage(repoDir string) error {
    ...

    buildCtx, err := archive.TarWithOptions(repoDir, &archive.TarOptions{
        Compression: archive.Uncompressed,
    })
    if err != nil {
        return errors.Wrap(err, "could not create build context")
    }
    defer buildCtx.Close()

    ...
}
```

buildImage() - send the tar to the Docker daemon.

```
import "github.com/docker/docker/pkg/archive"

func buildImage(repoDir string) error {
    ...

    resp, err := c.ImageBuild(ctx, buildCtx, types.ImageBuildOptions{
        Dockerfile: "./Dockerfile",
        Tags:       []string{"built_by_my_ci"},
    })
    if err != nil {
        return errors.Wrap(err, "could not send image build request")
    }

    ...
}
```

buildImage() - send the tar to the Docker daemon.

```
import "github.com/docker/docker/pkg/jsonmessage"
import "github.com/docker/docker/pkg/term"

func buildImage(repoDir string) error {
    ...

    fd, isTerm := term.GetFdInfo(os.Stdout)
    err = jsonmessage.DisplayJSONMessagesStream(resp.Body, os.Stdout, fd, isTerm, nil)

    return err
}
```

webhookHandler() - cont'd

```
func webhookHandler(w http.ResponseWriter, r *http.Request) {
    ...

    switch pre := event.(type) {
    case *github.PullRequestEvent:
        action := pre.GetAction()
        if action == "opened" || action == "reopened" || action == "synchronize" {
            err = buildPR(pre)
            status := "success"
            if err != nil {
                status = "error"
            }

            reportStatus(status, pre *github.PullRequestEvent)
        }
    default:
        fmt.Fprintf(w, "Uninteresting event: %T", event)
        return
    }
}
```

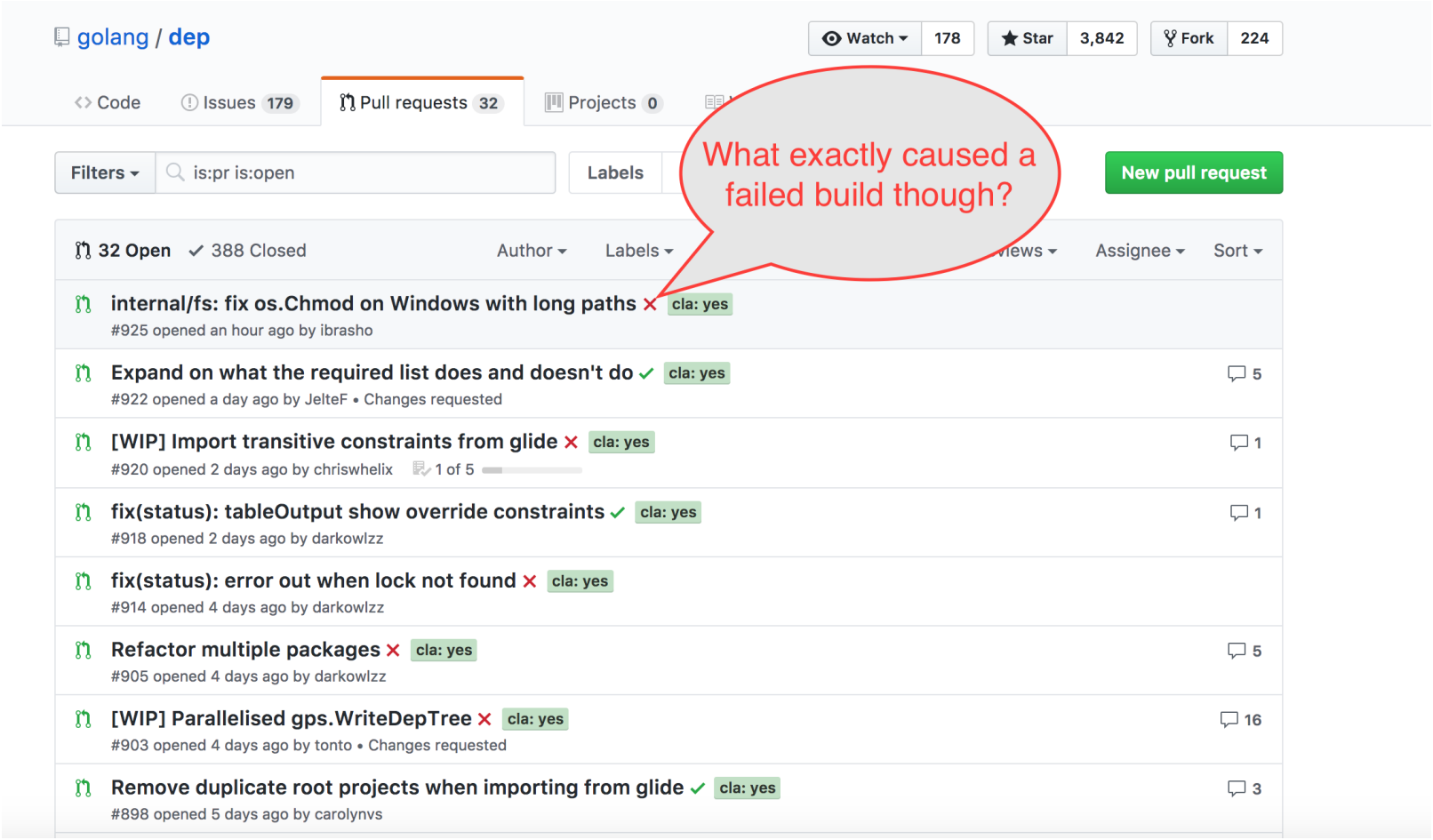
RECAP

- We get a webhook request.
- We parse the payload to get all the info we need about the pull request.
- We clone the repo to a temp directory
- We cd into that repo && `docker build` it.
- report back to github success/failure.

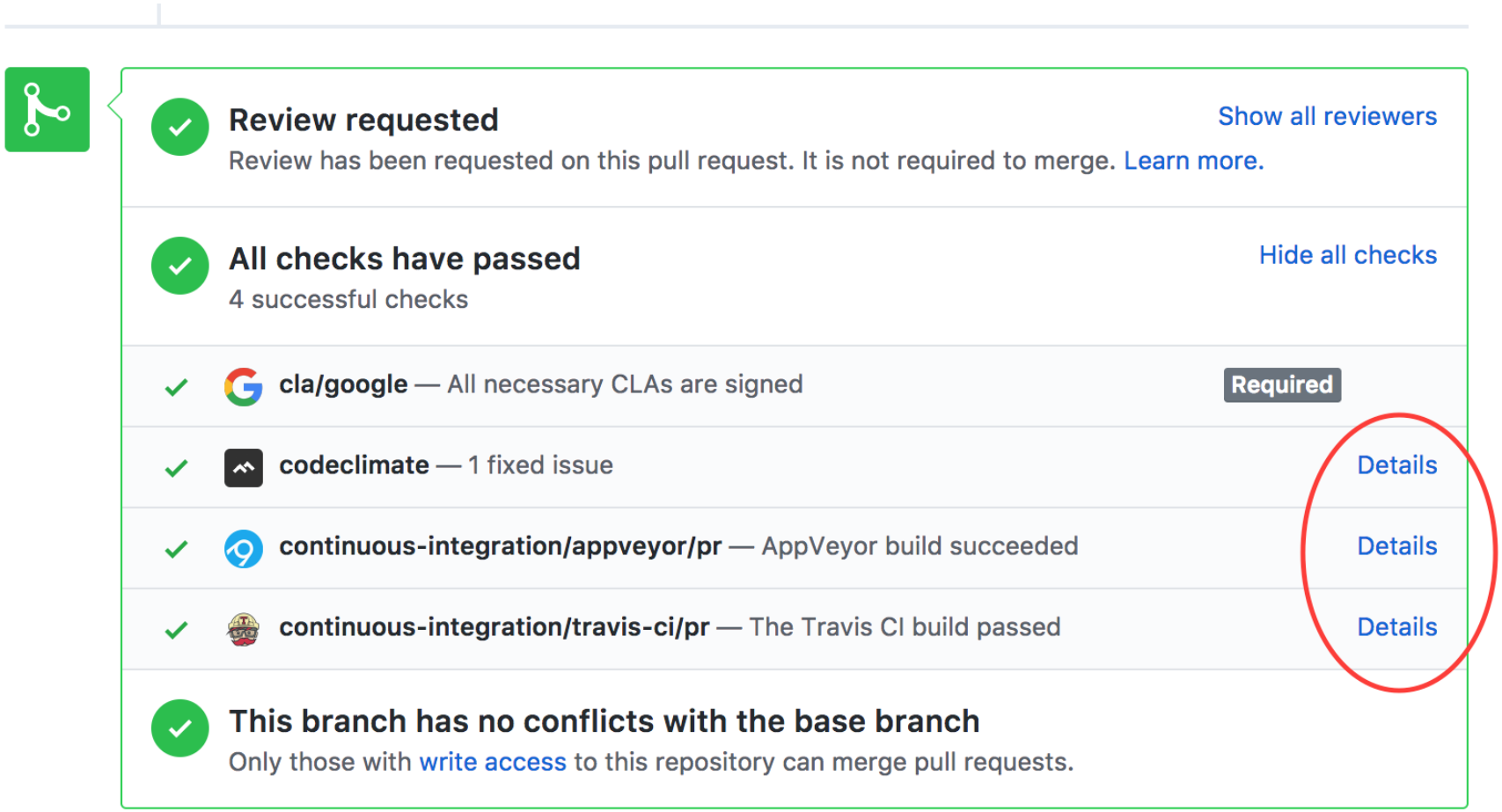
Demo

One Last, Important, Missing Piece: Persisting Logs

TargetURL


golang / dep
Watch 178
Star 3,842
Fork 224
Code
Issues 179
Pull requests 32
Projects 0
What exactly caused a failed build though?
Filters
is:pr is:open
Labels
New pull request
32 Open
388 Closed
Author
Labels
Assignee
Sort
internal/fs: fix os.Chmod on Windows with long paths
cla: yes
#925 opened an hour ago by ibrasho
Expand on what the required list does and doesn't do
cla: yes
#922 opened a day ago by JelteF • Changes requested
5
[WIP] Import transitive constraints from glide
cla: yes
#920 opened 2 days ago by chriswhelix
1 of 5
1
fix(status): tableOutput show override constraints
cla: yes
#918 opened 2 days ago by darkowlzz
1
fix(status): error out when lock not found
cla: yes
#914 opened 4 days ago by darkowlzz
Refactor multiple packages
cla: yes
#905 opened 4 days ago by darkowlzz
5
[WIP] Parallelised gps.WriteDepTree
cla: yes
#903 opened 4 days ago by tonto • Changes requested
16
Remove duplicate root projects when importing from glide
cla: yes
#898 opened 5 days ago by carolynvs
3

TargetURL


Review requested
Show all reviewers
Review has been requested on this pull request. It is not required to merge. Learn more.
All checks have passed
Hide all checks
4 successful checks
cla/google — All necessary CLAs are signed
Required
codeclimate — 1 fixed issue
Details
continuous-integration/appveyor/pr — AppVeyor build succeeded
Details
continuous-integration/travis-ci/pr — The Travis CI build passed
Details
This branch has no conflicts with the base branch
Only those with write access to this repository can merge pull requests.

TargetURL

```
func reportStatus(status string, pre *github.PullRequestEvent) {
    ...

    link := "https://www.my-ci-server.com/logs?id=123"

    c.Repositories.CreateStatus(context.Background(), owner, repoName, ref, &github.RepoStatus{
        State: &status,
        TargetURL: &link, // a link to this particular build logs.
    })
}
```

io.Writer

```
repo, err := git.PlainClone(tempDir, false, &git.CloneOptions{
    URL:               gitURL,
    Progress:          os.Stdout, // io.Writer <----
    RecurseSubmodules: git.DefaultSubmoduleRecursionDepth,
    ReferenceName:     plumbing.ReferenceName(fmt.Sprintf("refs/heads/%v", branchName)),
})

err = jsonmessage.DisplayJSONMessagesStream(
    resp.Body,
    os.Stdout, // io. Writer <----
    fd,
    isTerm,
    nil,
)
```

io.Writer implemented

```go
// Handles logs logic of a build process
type Logger struct {
    ID  string
    URL string
}

// implement the io.Writer
func (l *Logger) Write(p []byte) (int, error) {
    persistToDB(l.ID, p)

    logToTerminal(l.ID, p)

    tweetIt(l.ID, p)

    sendItToMyMom(l.ID, p)

    return len(p), nil
}
```

io.Writer used

```
myLogger: &Logger{ID: id}

repo, err := git.PlainClone(tempDir, false, &git.CloneOptions{
    URL:               gitURL,
    Progress:          myLogger, // io.Writer <----
    RecurseSubmodules: git.DefaultSubmoduleRecursionDepth,
    ReferenceName:     plumbing.ReferenceName(fmt.Sprintf("refs/heads/%v", branchName)),
})

err = jsonmessage.DisplayJSONMessagesStream(
    resp.Body,
    myLogger, // io. Writer <----
    fd,
    isTerm,
    nil,
)
```

Demo 2

Thank You

Marwan Sulaiman - github.com/marwan-at-work

Developer at Work & Co