

CI Workflow auf Basis von GitHub

Inhalt

[1 Allgemeines Vorgehen 3](#_Toc509579121)

[2 Mit dem Workflow arbeiten 4](#_Toc509579122)

[2.1 Erste Schritte mit einem Repository 4](#_Toc509579123)

[2.2 Aufgabe beginnen 5](#_Toc509579124)

[2.3 Aufgabe bearbeiten 6](#_Toc509579125)

[2.4 Änderungen verteilen 6](#_Toc509579126)

[2.5 Status ansehen 7](#_Toc509579127)

[2.6 Fehleranalyse 9](#_Toc509579128)

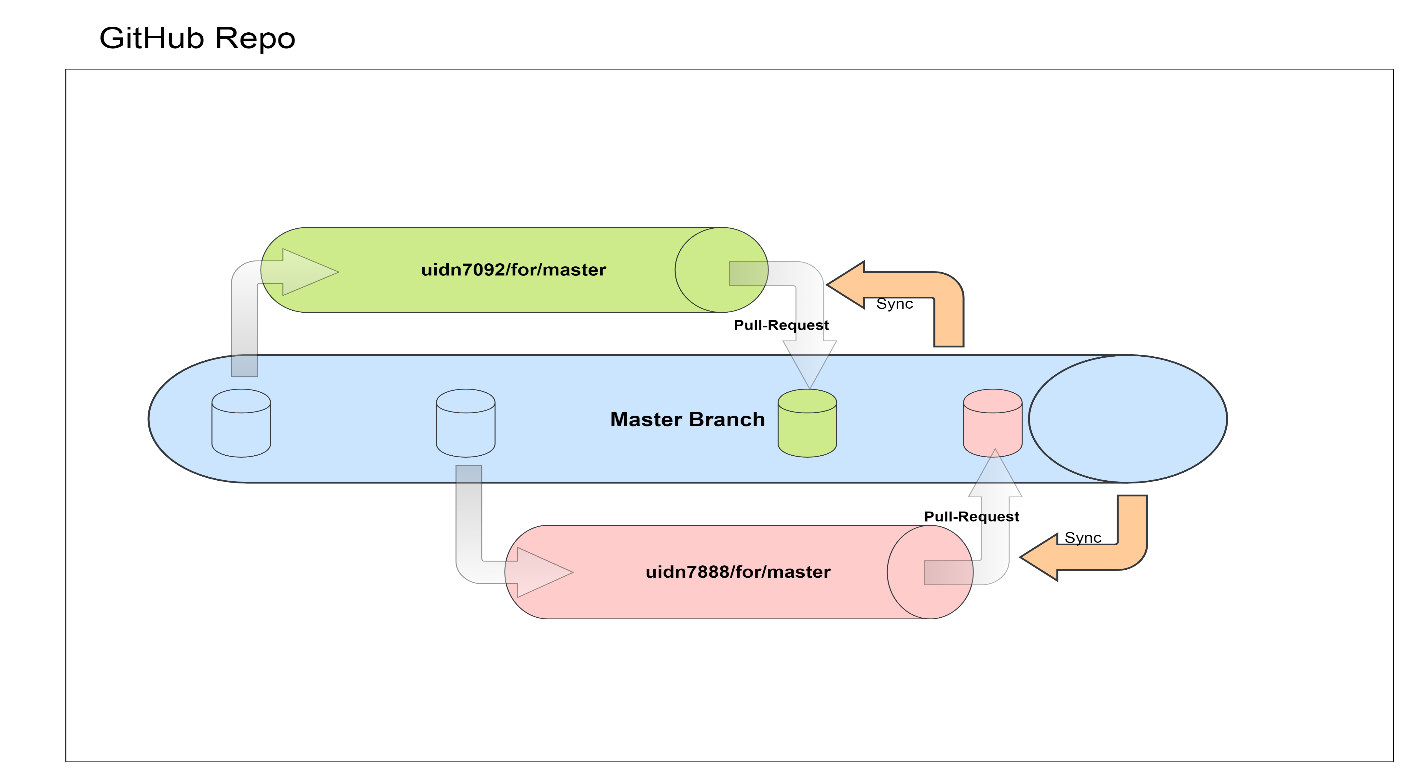
# Allgemeines Vorgehen

Die Entwicklung findet auf Branches statt, die möglichst kurzlebig sein sollen und normalerweise personenbezogen sind (in der folgenden Darstellung grün und rot dargestellt). Sie können aber auch für bestimmte Bugfixes oder Features erstellt werden.

Normalerweise arbeitet nur eine Person an einem Branch. Es ist aber auch möglich, in kleinen Teams zusammen an einem Branch zu arbeiten. Hierzu sollte jeder, der an dem Branch arbeitet, die lokalen Commits möglichst häufig in den Branch pushen.

Sobald das entwickelte Feature bereit für die Weitergabe ist, wird es über einen sogenannten Pull Request in den Master gemerged. Der Master selbst ist gesperrt, sodass die Entwicklung tatsächlich nur in Branches stattfinden kann.

Entwicklungsstände und Funktionen sollten nicht „halbfertig“ über Feature-Branches geteilt, sondern in fertigen Ständen über den Master verteilt werden.

Der Vorteil dieses Vorgehens liegt in einem sehr stabil wachsenden Master und einer Reduzierung der Merge-Konflikte.

Der Workflow unterstützt zwei Vorgehensweisen:

1. Mit automatischer Erzeugung der Pull Requests

Beim ersten Push in einen Branch, der diesem Vorgehen folgt, wird automatisch einen Pull Request für den Branch auf GitHub angelegt.

Anschließend werden die für das Repository festgelegten Prüfungen durchgeführt. Der Workflow gibt vor, dass zumindest geprüft wird, ob die mit dem Push gelieferten Änderungen nach Zusammenführung mit dem Stand im Master kompilierfähig wären und die vorhandenen Unit-Tests erfolgreich durchgeführt werden könnten.

Bei jedem weiteren Push werden die Änderungen zum offenen Pull Request hinzugefügt und erneut geprüft. Bis der Pull Request geschlossen wird.

1. Mit manueller Erstellung der Pull Requests

Beim Pushen in einen Branch, der diesem Vorgehen folgt, startet der Workflow nur die für das Repository festgelegten Prüfungen.

Der Pull Request muss manuell über GitHub erstellt werden. Alles andere verläuft analog zu 1.

# Mit dem Workflow arbeiten

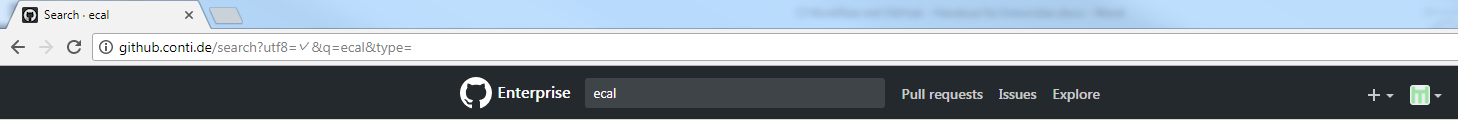
Die Arbeitsschritte werden in diesem Dokument anhand der Git Bash erklärt. Auf alle GUI Clients einzugehen, die es für Git gibt, würde den Rahmen dieses Dokuments sprengen.

## Erste Schritte mit einem Repository

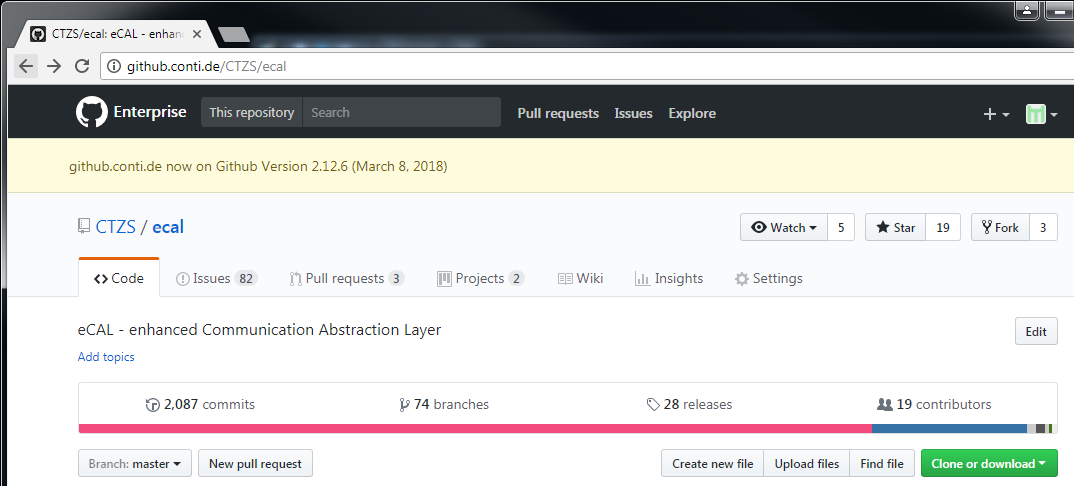
Als Erstes muss eine lokale Kopie des Remote Repositories auf dem Entwicklerrechner angelegt werden. Dafür folgende Schritte ausführen:

1. In einem Browser GitHub öffnen: <http://github.conti.de/>
2. Zum gewünschten Repository navigieren

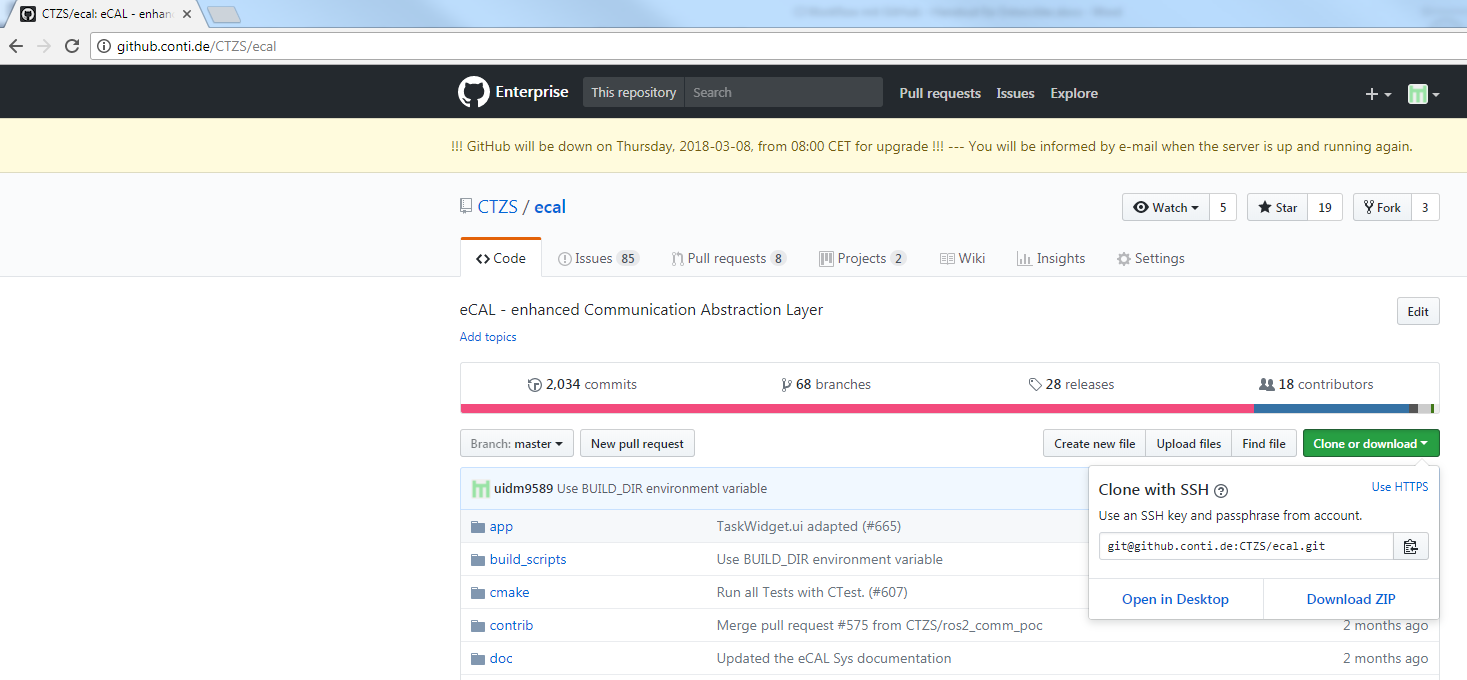
(z.B. den Repository-Namen in die Suchleiste eingeben):



1. In der Repository-Ansicht rechts auf „Clone or download“ klicken.



1. Im anschließend angezeigten Bereich die Clone URL des Repositories kopieren:



1. In einer Git Bash in das Verzeichnis wechseln, in dem die lokale Kopie des Repositories angelegt werden soll.
2. Den Befehl „git clone“ eingeben und die URL anfügen, die in Schritt 4 kopiert wurde.

Beispiel: **git clone git@github.conti.de:CTZS/ecal.git**

## Aufgabe beginnen

Als Erstes muss ein neuer Branch erstellt werden. Dafür müssen folgende Schritte ausgeführt werden:

1. Branch-Bezeichnung wählen

Über die Branch-Bezeichnung wird entschieden, welche der beiden Vorgehensweisen, die der Workflow unterstützt, angewendet wird.

1. automatische Erzeugung der Pull Requests

In diesem Fall muss die Branch-Bezeichnung dem Muster

**{identifier}/for/{base}** folgen.

{identifier} steht für einen beliebigen Text (z.B. die User ID, die Issue Nummer oder den Feature Namen), darf aber /for/ nicht enthalten.

{base} steht für den Ziel Branch des Pull Requests (z.B. master)

Beispiel: uidm9589/for/master

1. manuelle Erstellung der Pull Requests

In diesem Fall darf der Branch-Name den Text **/for/** nicht enthalten.

1. Lokalen Branch anlegen

Über Git Bash mit dem Befehl:

**git checkout –b <branch-name>**

Beispiel: git checkout –b uidm9589/for/master

1. Branch in das Remote Repository einfügen

Über Git Bash mit dem Befehl:

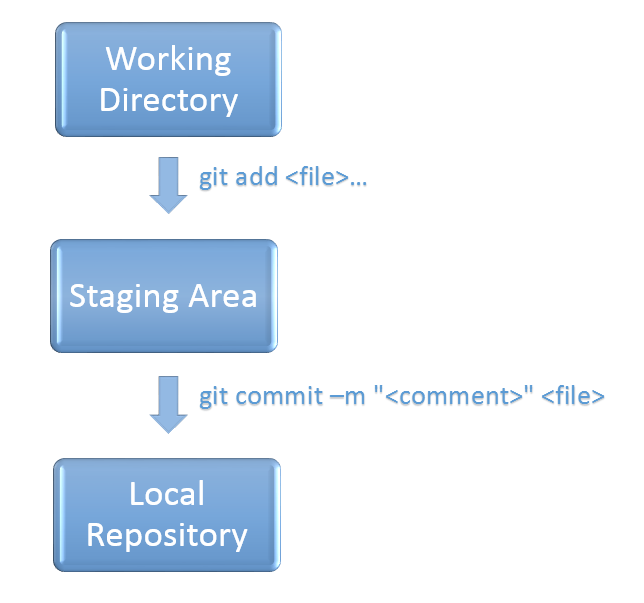
**git push --set-upstream origin <branch-name>**

Beispiel: git push --set-upstream origin uidm9589/for/master

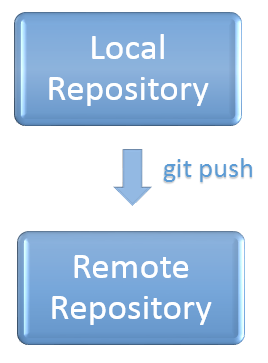
Die Option --set-upstream sorgt dabei dafür, dass der lokale Branch mit dem Remote Branch assoziiert wird, so dass man die Zielangabe bei allen folgenden Befehlen weglassen kann.

## Aufgabe bearbeiten

Änderungen im lokalen Arbeitsverzeichnis durchführen und wie folgt sichern:



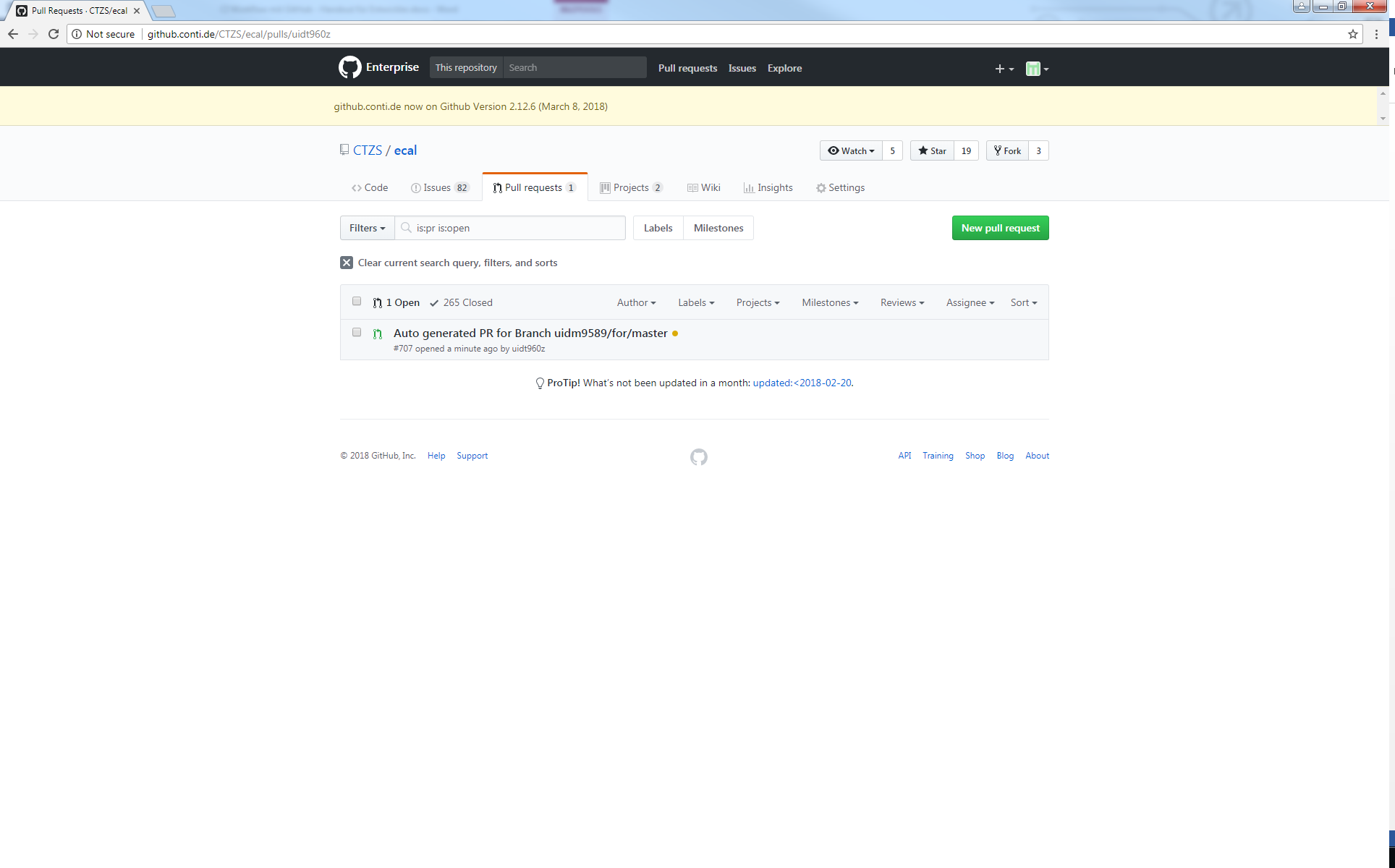
## Änderungen verteilen

Sobald ein stabiler Stand erreicht wurde, kann dieser in den Branch im Remote Repository gepushed werden.

## Status ansehen

Sobald ein Pull Request erstellt wurde (automatisch oder manuell), kann der Stand der durch den Workflow durchgeführten Prüfungen über GitHub verfolgt werden.

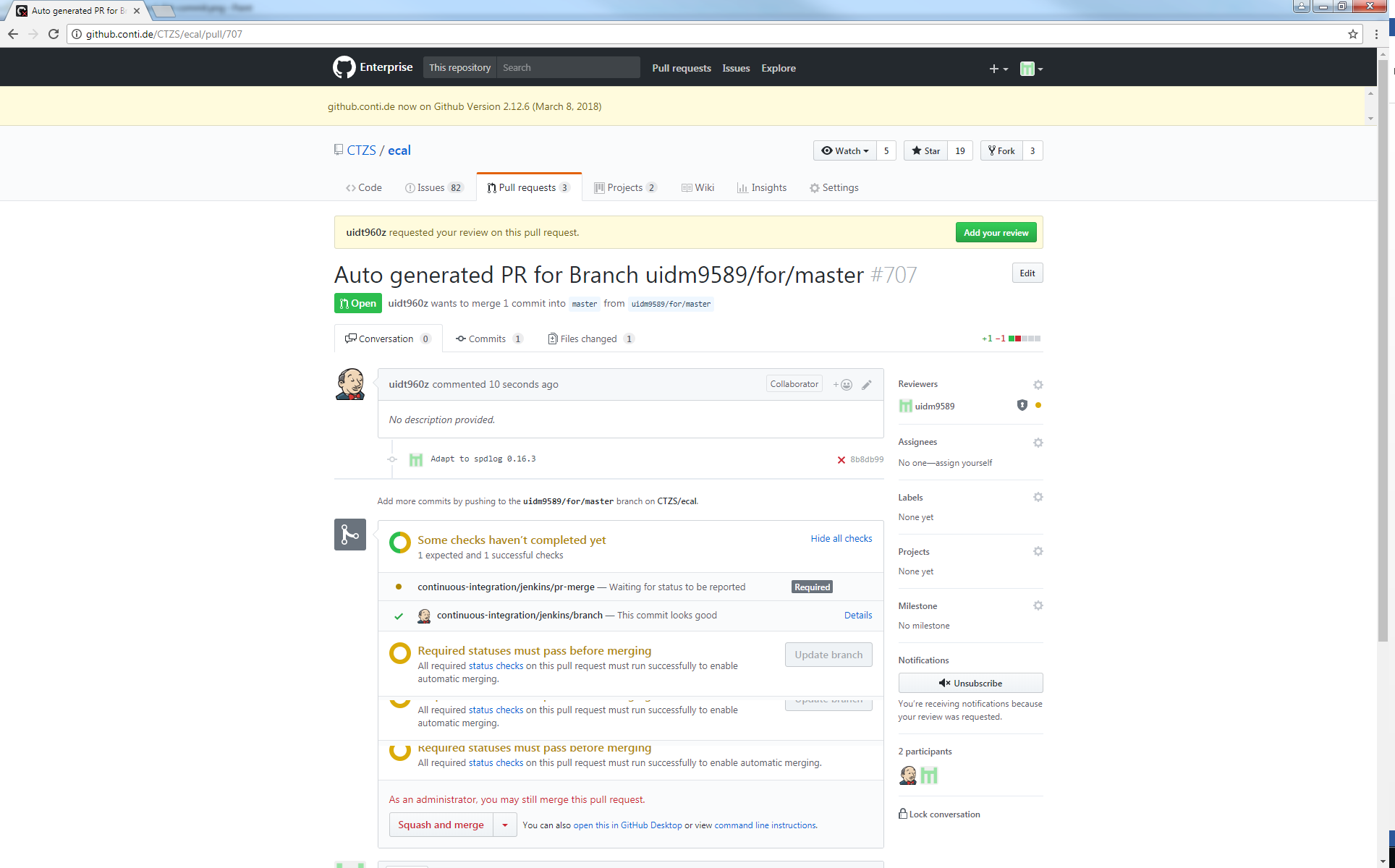
1. GitHub öffnen: <http://github.conti.de/>.
2. Zum gewünschten Repository navigieren.
3. Auf den Reiter „Pull requests“ wechseln:



1. Detailansicht durch Klick auf die **Bezeichnung des Pull Requests** öffnen

(im Beispiel ist das „Auto generated PR for Branch uidm9589/for/master“).

Im unteren (in der folgenden Darstellung rot markierten) Bereich wird der **Status** der für das Repository festgelegten Prüfungen angezeigt.

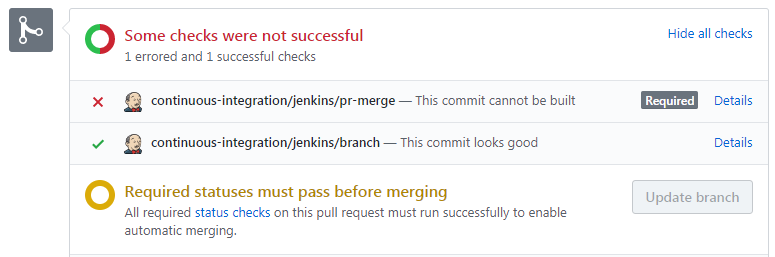


Die Prüfung, ob die Änderungen nach Zusammenführung mit dem Stand im Master kompilierfähig wären und die vorhandenen Unit-Tests erfolgreich durchgeführt werden könnten, verbirgt sich hinter der Bezeichnung „**continuous-integration/jenkins/pr-merge**“ (in der folgenden Darstellung mit einem roten Pfeil gekennzeichnet).

Während eine Prüfung durchgeführt wird, wird links daneben ein gelber Punkt angezeigt.

Sobald die Prüfung abgeschlossen ist, wird stattdessen ein grüner Haken (bei erfolgreicher Prüfung) oder ein rotes Kreuz (für eine fehlgeschlagene Prüfung) angezeigt.

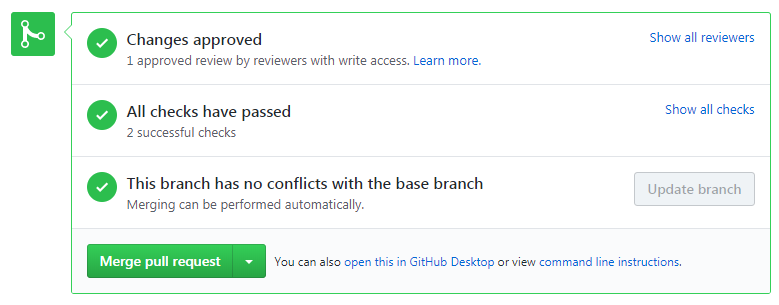
Schlägt die Prüfung der Kompilierfähigkeit oder ein Unit-Test fehl, wird automatische eine E-Mail an die betroffenen Entwickler versandt. In GitHub wird der Fehlschlag im Pull Request wie folgt angezeigt:



Rechts neben der Bezeichnung der Prüfung wird der Link **„Details“** angezeigt, über den man zur Fehleranalyse zu Jenkins wechseln kann.

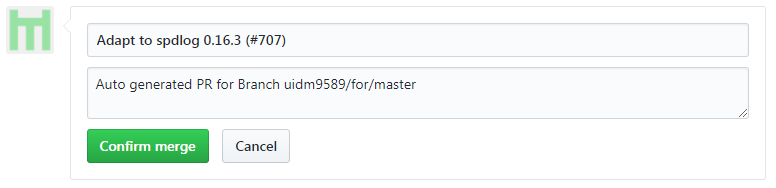
Die Fehlerbehebung erfolgt analog zur Bearbeitung (siehe Kapitel 2.3). Sobald alle Änderungen durchgeführt worden sind, um den Fehler zu beheben, können diese in den Branch im Remote Repository gepushed werden. Die für das Repository festgelegten Prüfungen werden dann erneut durchgeführt und die Status-Ansicht des Pull Request in GitHub entsprechend aktualisiert.

Wenn alle für das Repository festgelegten Prüfungen erfolgreich beendet wurden, wird der Merge-Button freigeschaltet:

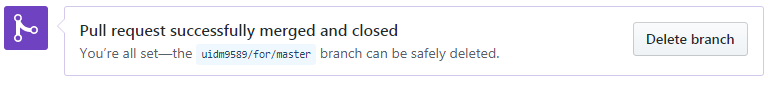


Über den Button „**Merge pull request**“ können die Änderungen für die Überführung in den Master vorbereitet werden.

Anschließend kann die Commit-Message und die Beschreibung für den Master festgelegt werden, bevor der Merge über den Button „**Confirm merge**“ gestartet wird:



Die Anzeige in GitHub sollte anschließend wie folgt aussehen:



Zum Abschluss sollte der Branch gelöscht werden, damit der Plattenplatz in der Build-Umgebung wieder freigegeben wird.

## Fehleranalyse

Über die Pull Request Anzeige in GitHub oder die im Fehlerfall versandte E-Mail kann zur Fehleranalyse der entsprechende Job in Jenkins aufgerufen werden.

Die vom Workflow durchgeführten Prüfungen werden sowohl für Linux als auch für Windows durchgeführt. Zur besseren Lesbarkeit sind die zugehörigen Konsolenausgaben jeweils am Anfang der Zeile durch [linux] bzw. [windows] gekennzeichnet.

Kompilerfehler und -warnungen werden gesammelt und können über den Menüpunkt „Parsed Console Output“ eingesehen werden.