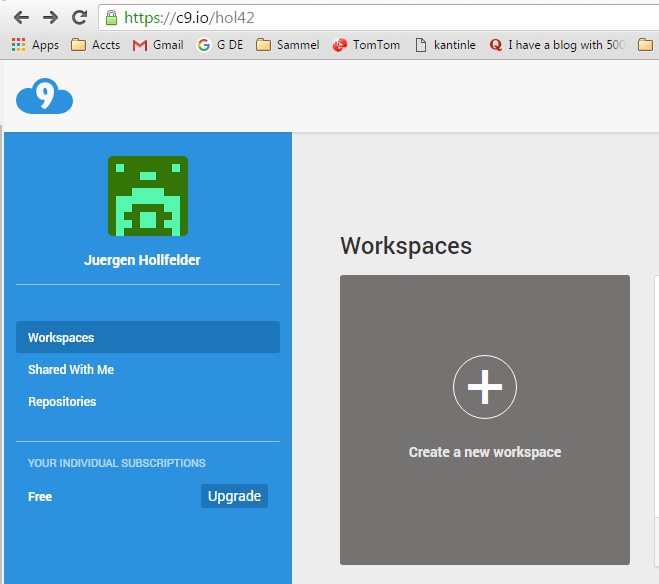
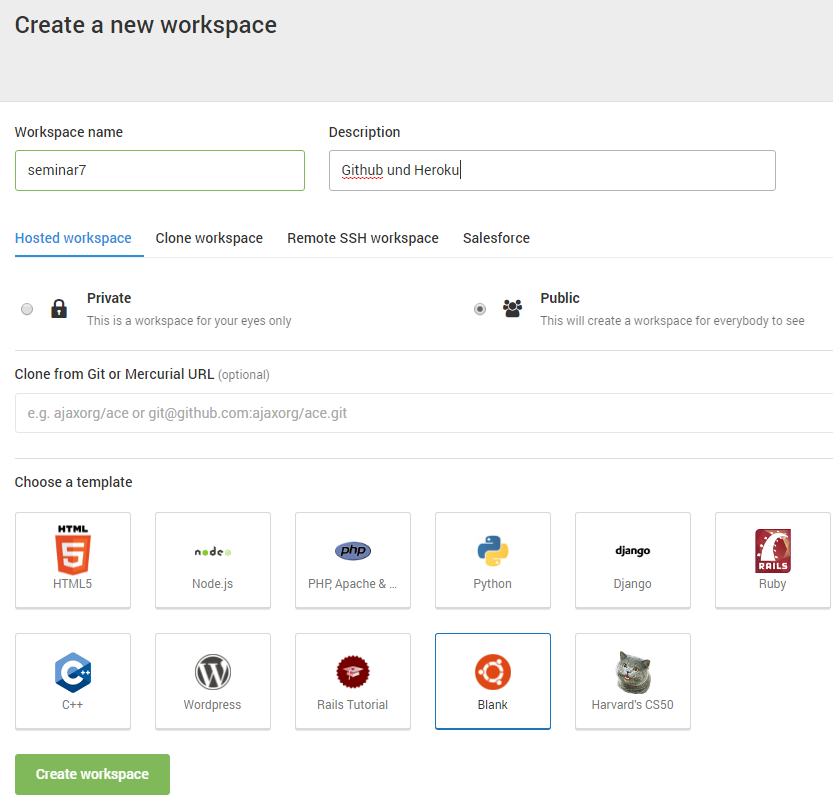
Seminar 7

1) Cloud9: Neuen Workspace anlegen

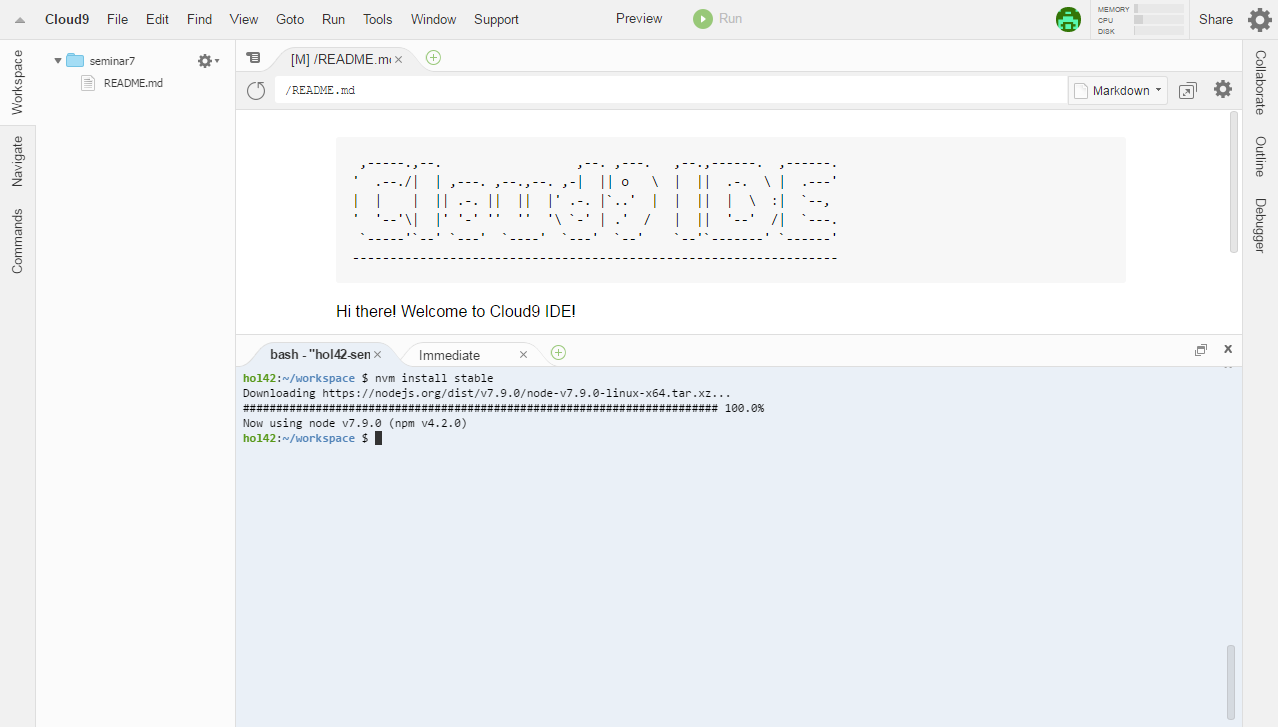


2) Blank Ubuntu reicht



3) Abwarten

4) Node auf neueste Version updaten

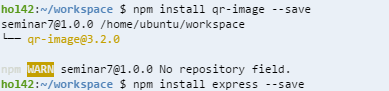


5) Als Testprojekt wollen wir serverseitig QR Codes produzieren, die wir zurückgeben

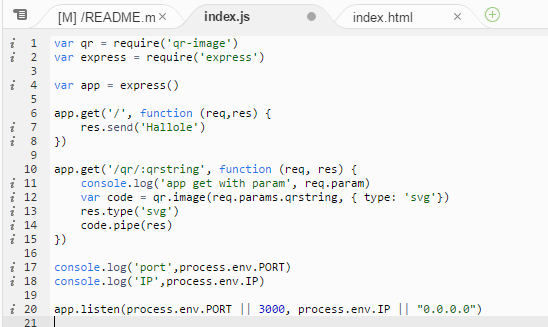
Ein NodeProjekt anlegen:



und dann noch qr-image und express zufügen



6) index.js anlegen und eintippen



var qr = require('qr-image')

var express = require('express')

var app = express()

app.get('/', function (req,res) {

res.send('Hallole')

})

app.get('/qr/:qrstring', function (req, res) {

console.log('app get with param', req.param)

var code = qr.image(req.params.qrstring, { type: 'svg'})

res.type('svg')

code.pipe(res)

})

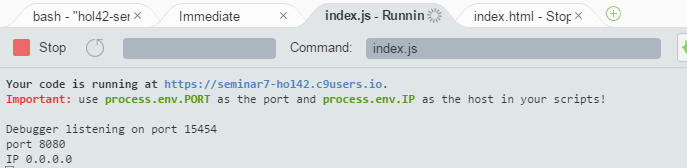
console.log('port',process.env.PORT)

console.log('IP',process.env.IP)

app.listen(process.env.PORT || 3000, process.env.IP || "0.0.0.0")

7) testen

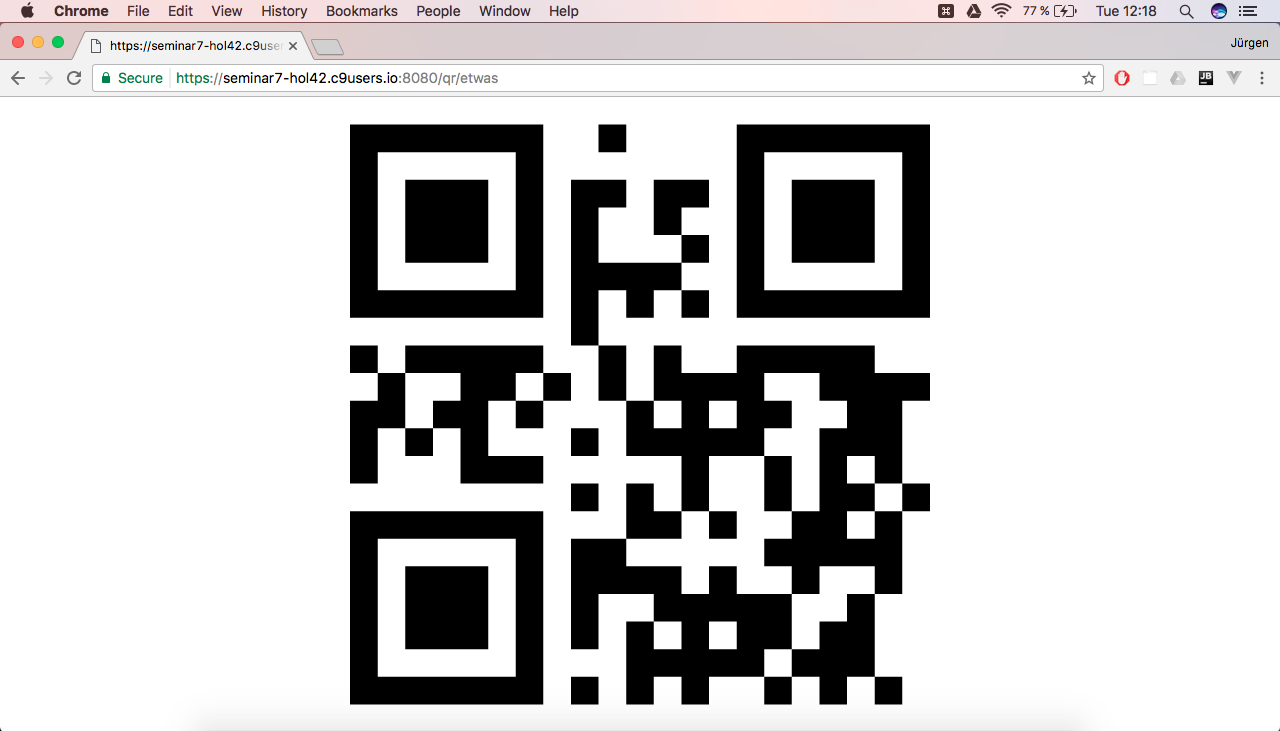
Run klicken



Jetzt im Browser eingeben

<https://seminar7-hol42.c9users.io:8080/qr/versuch>

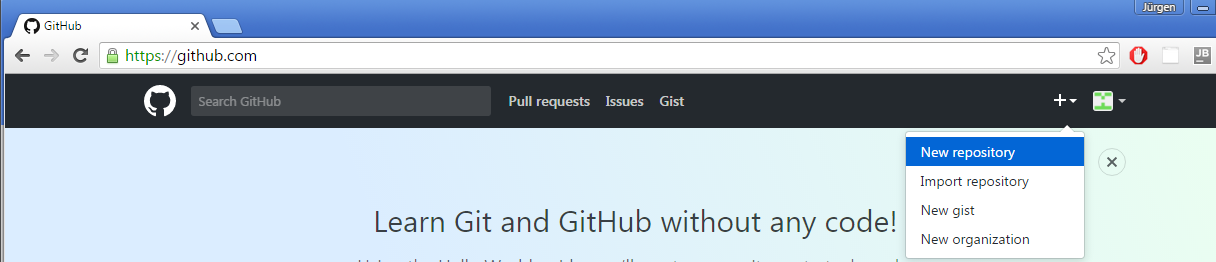
Das funktioniert nicht im internen Netz. Aber ausserhalb schon. Firewall Problem!

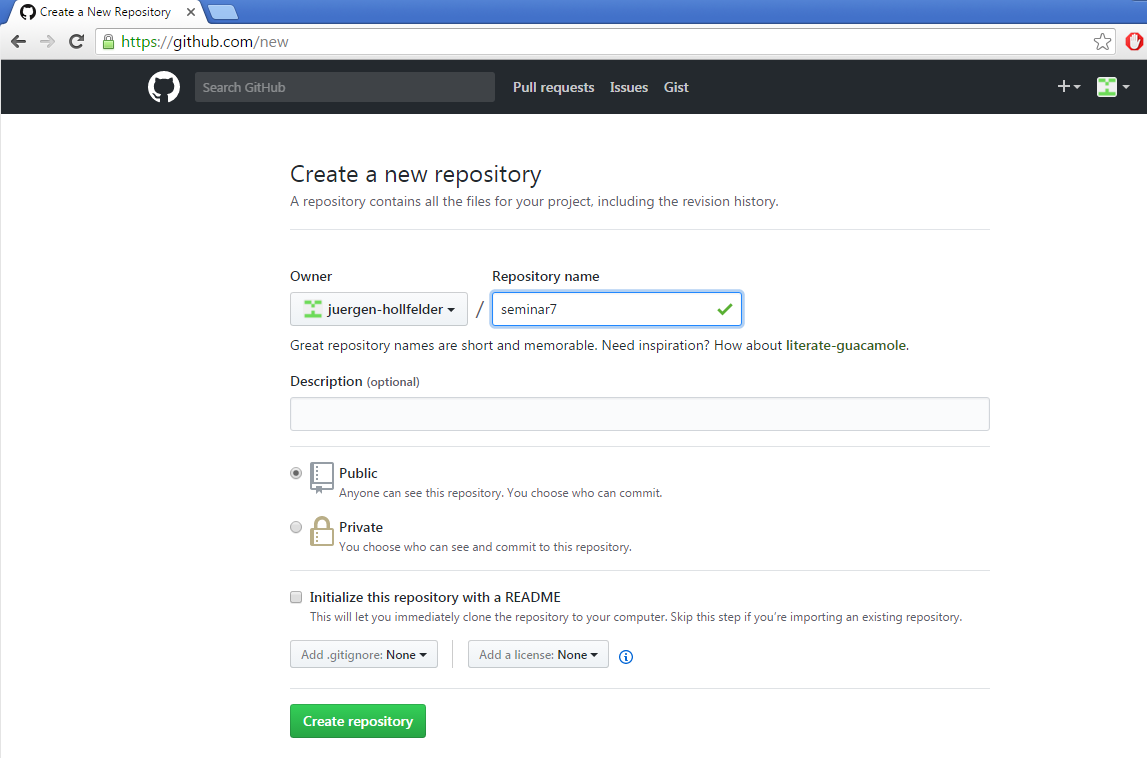


Jetzt kommt der eigentliche Seminarteil

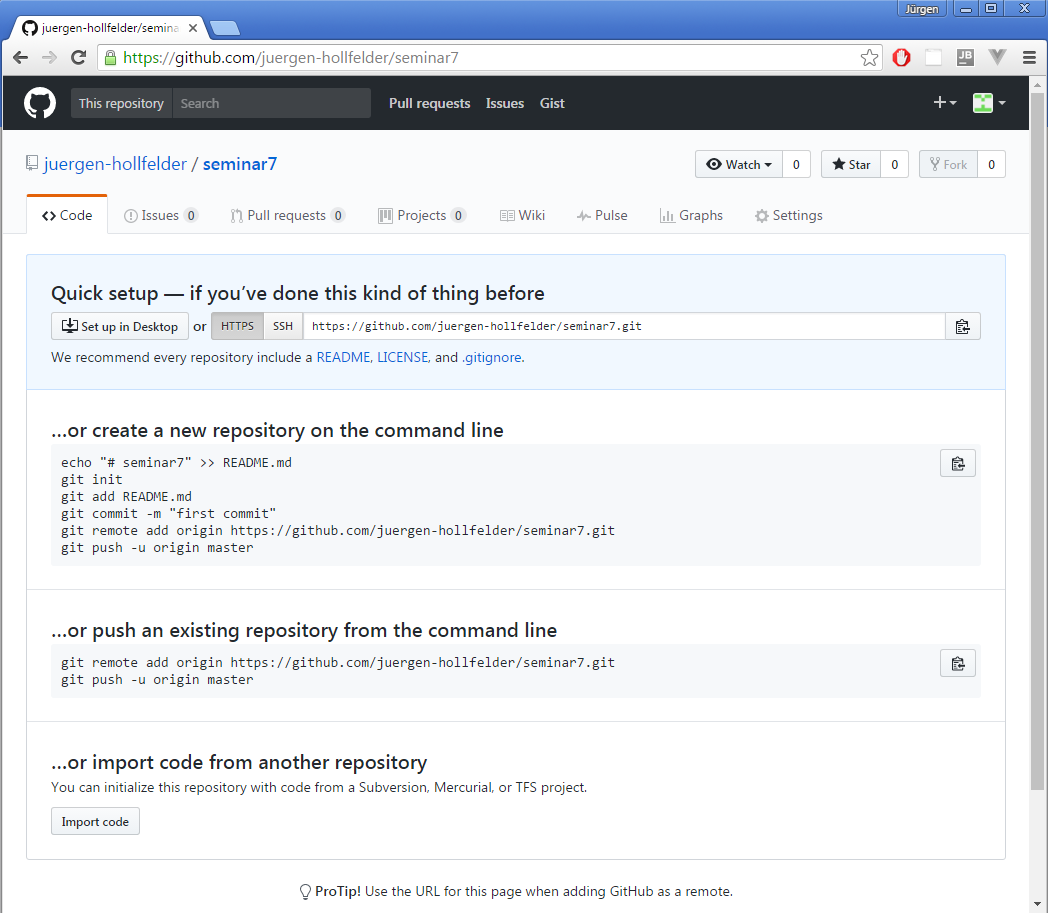
8) Fertige App in Github stellen

8.1) Repository erstellen

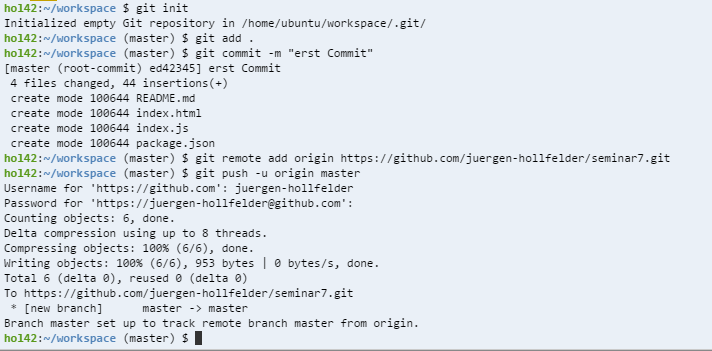




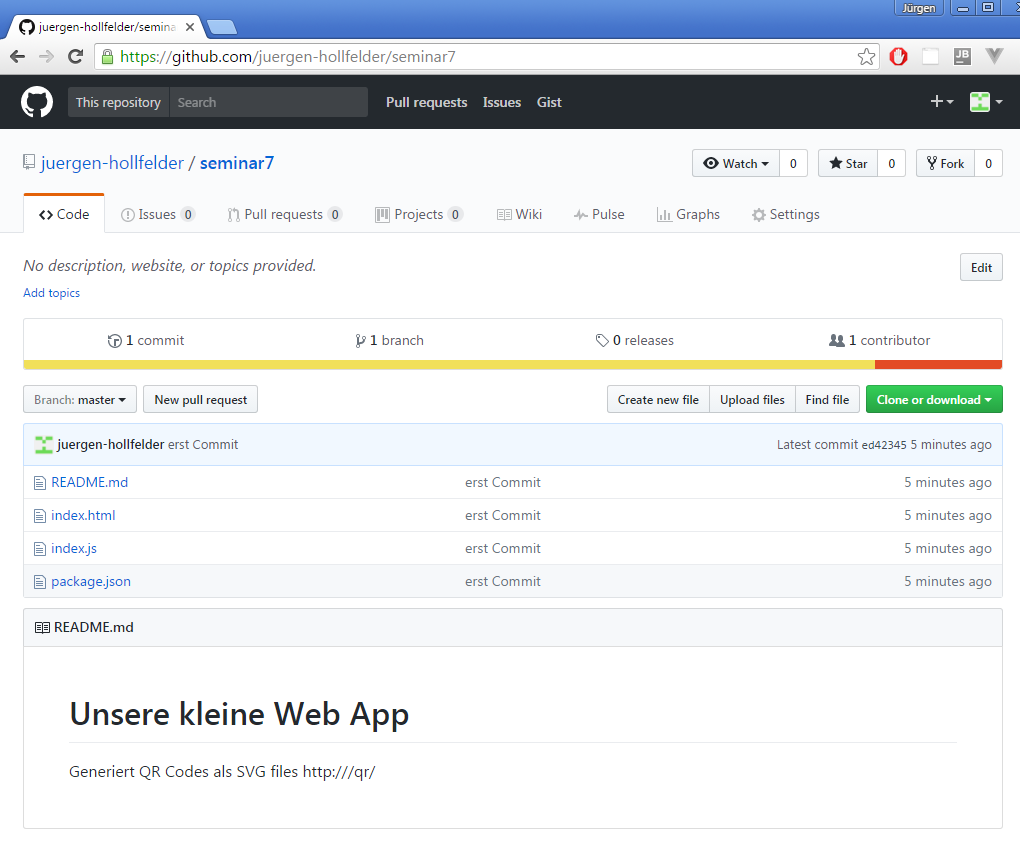
8.2) den Anweisungen folgen



Ein README.me haben wir schon, dass kann man natürlich noch vorher anpassen



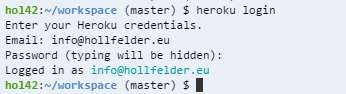
Jetzt ist alles da:



Jetzt sollten wir die App ja noch irgendwo auf einem zentral erreichbaren Server hosten. Dazu bietet sich Heroku an.

9) Bei Heroku ein Account erstellen

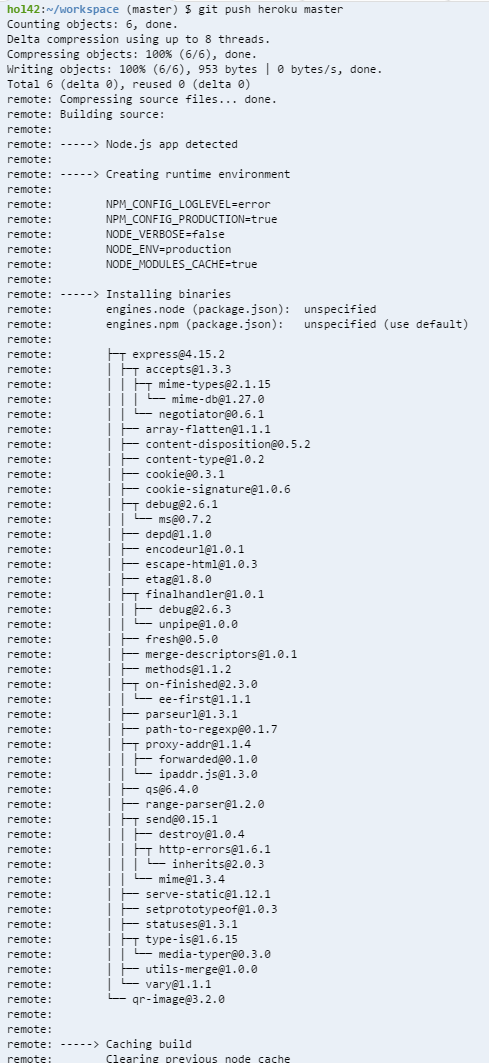
10) Bei Heroku anmelden. Heroku ist vorinstalliert in der Cloud9 Ubuntu Maschine

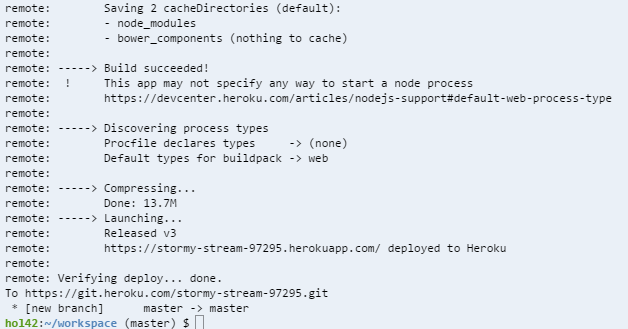


11) Heroku app erstellen



12) Deploy





13) Starten



14) Logs anschauen



15) Ausprobieren

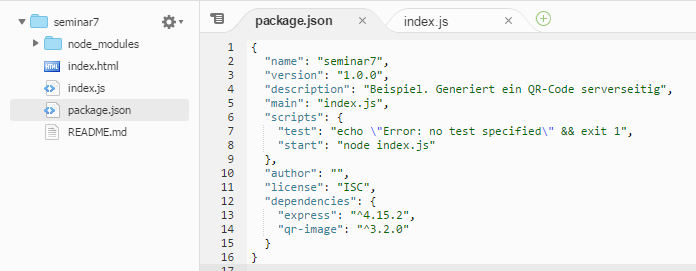
<http://stormy-stream-97295.herokuapp.com/>

16) Fehlermeldungen?!

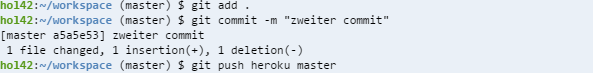
npm ERR! argv "/app/.heroku/node/bin/node" "/app/.heroku/node/bin/npm" "start"

Heroku versucht die App zu starten mit **npm start**. Was bei **npm start** passieren soll muss aber definiert werden.

17) package.json anpassen



18) Nochmal GIT (lokal) update und Heroku Update



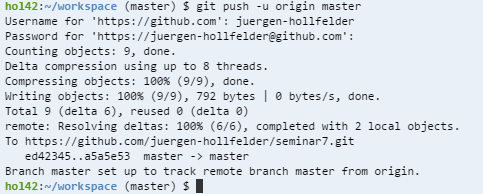
19) Neuer Test, jetzt geht’s



20) Jetzt haben wir die Source wo?

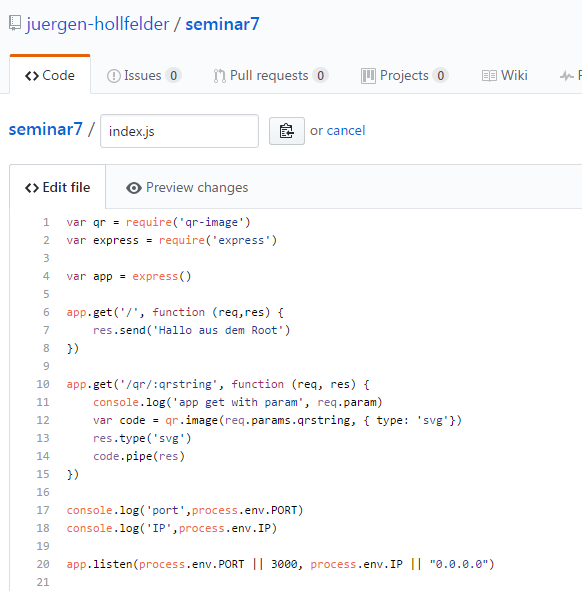
Nur bei Heroku!

21) Gut, dann pushen wir es auch zu GitHub

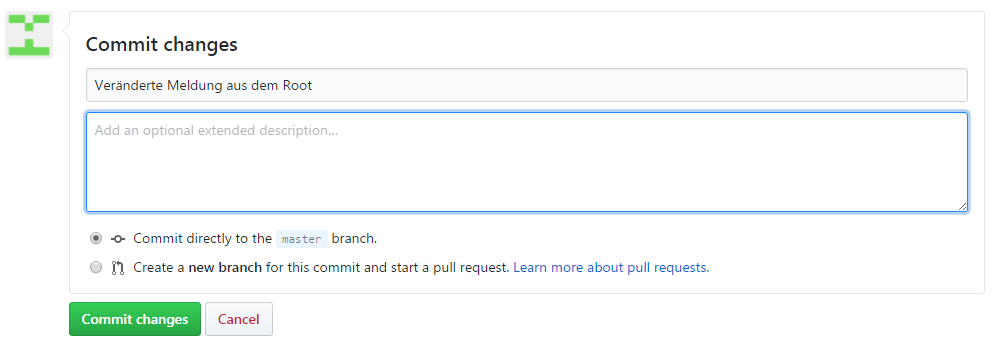


22) Was aber wenn jemand anderes etwas verändert hat? Jetzt wollen wir das mal simulieren.

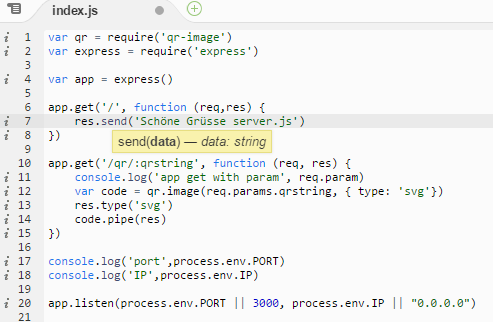
Im Github GUI tatsächlich in index.js etwas verändern:



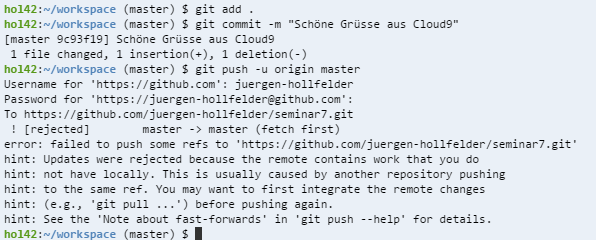
Mit direktem Commit:



In der Cloud9 auch dort verändern:

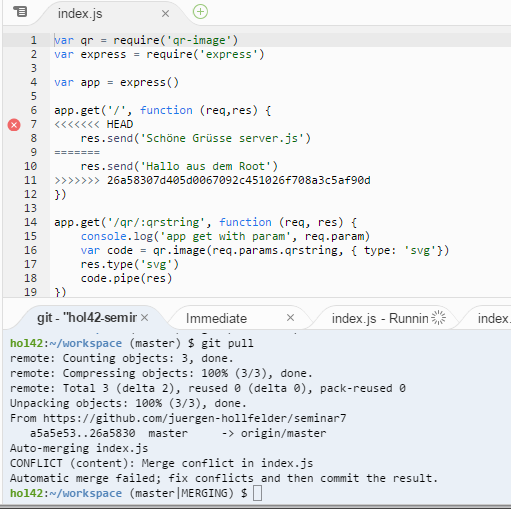


Das dann auch committen



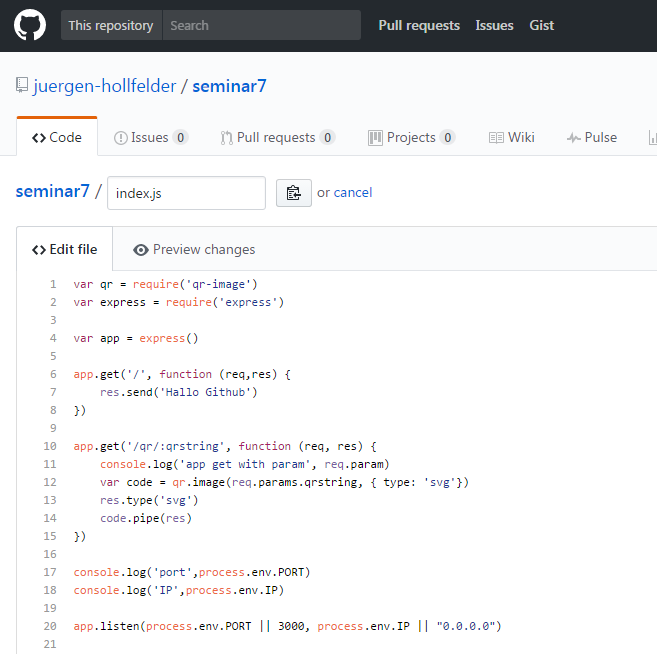
Es gibt ein Konflikt. Was jetzt?

GIT Pull zeigt einem wo man abweicht



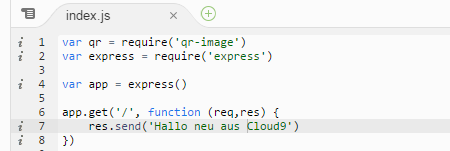
Nochmal

a) Ändern in Github:



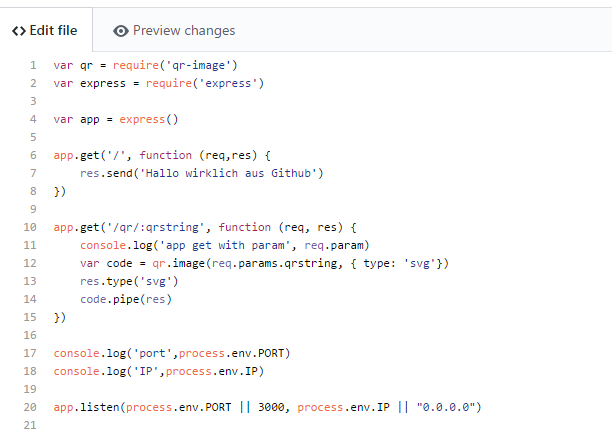
+ Commit

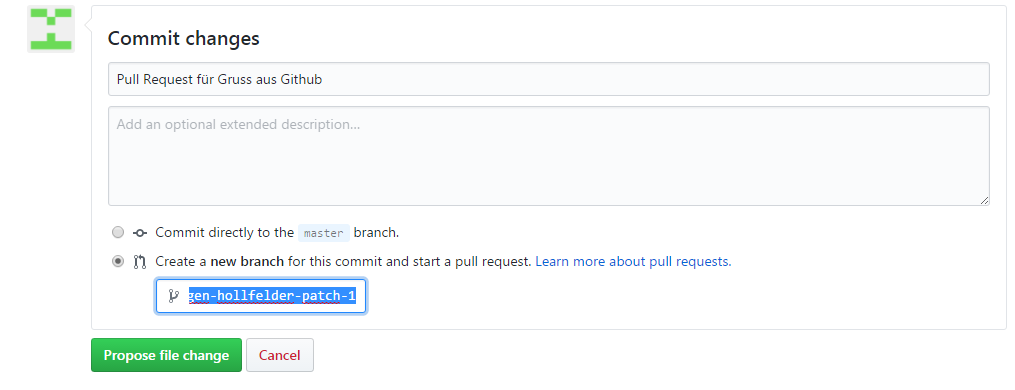
B) Gleichzeitig ändern in Cloud 9:

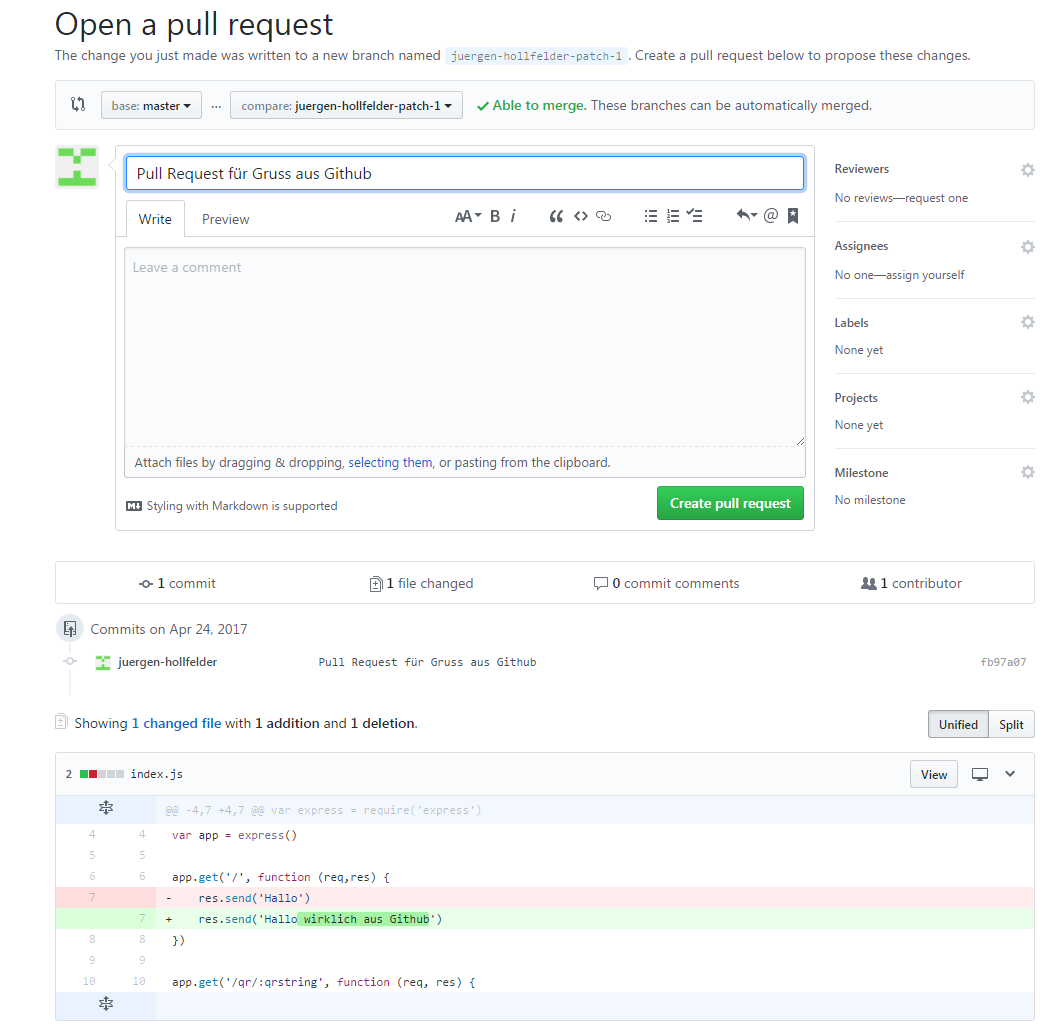




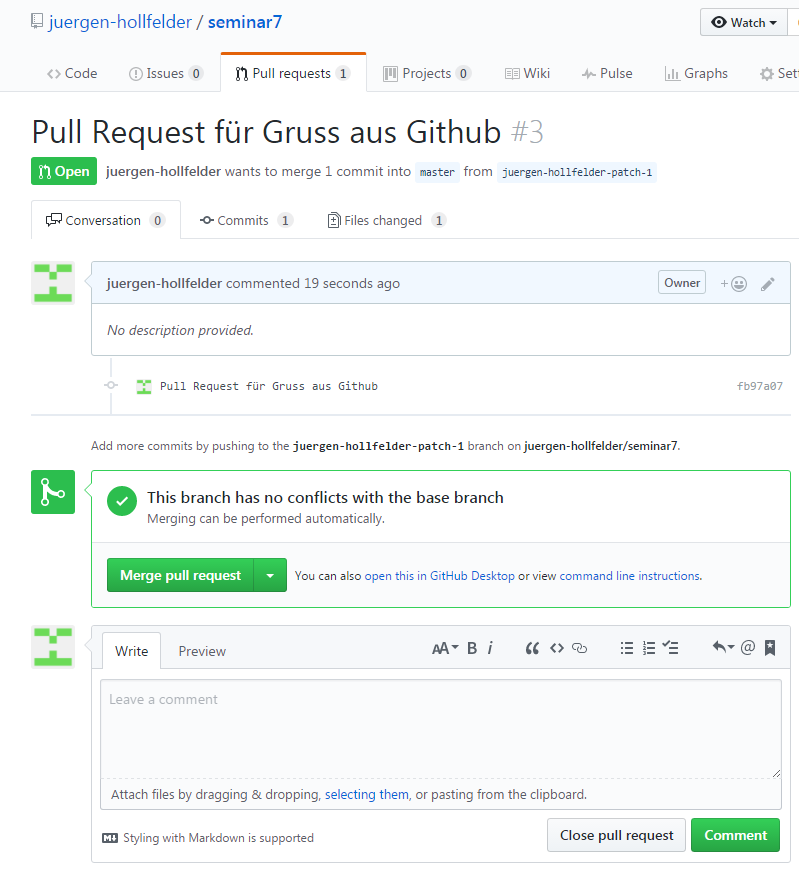
Oder man macht es so:



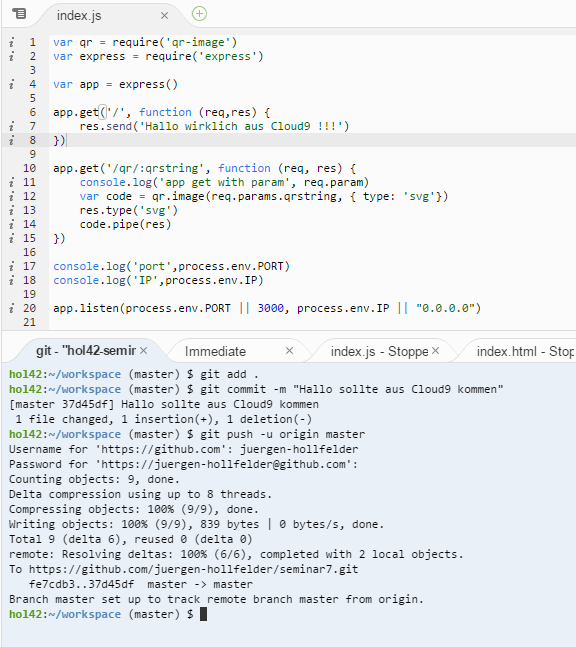




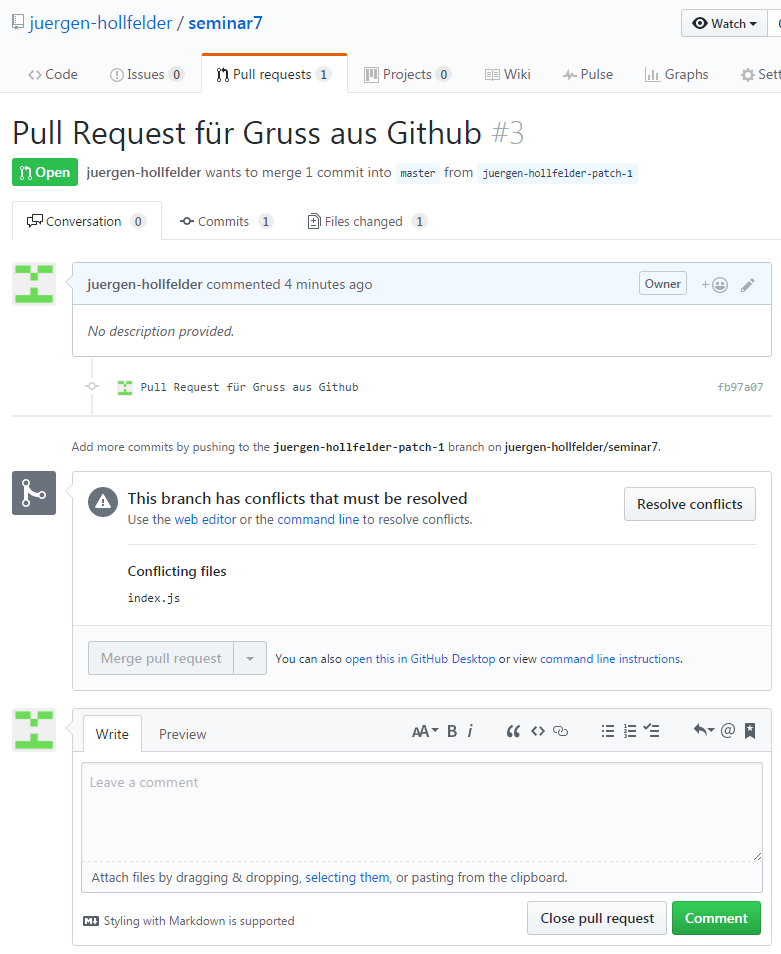
Solange es kein Konflikt gibt könnte man einfach mergen und fertig:



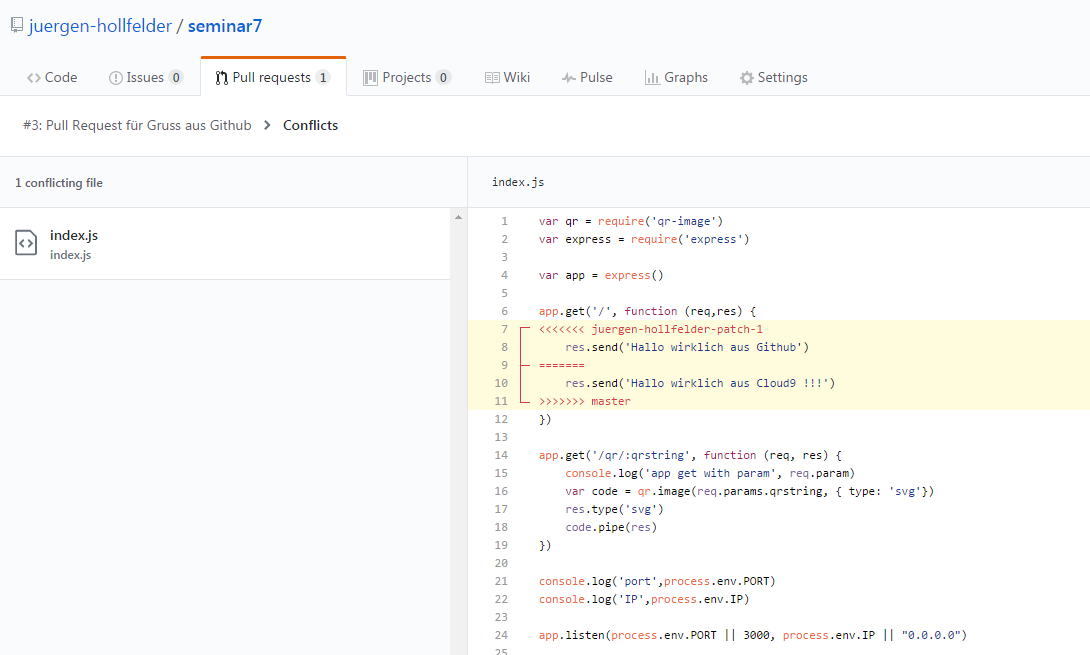
Aber sagen wir mal wir ändern in der Cloud 9 wieder etwas:



Jetzt sieht das anders aus, jetzt erkennt Github den Konflikt:



Klick auf „Resolve conflicts“:

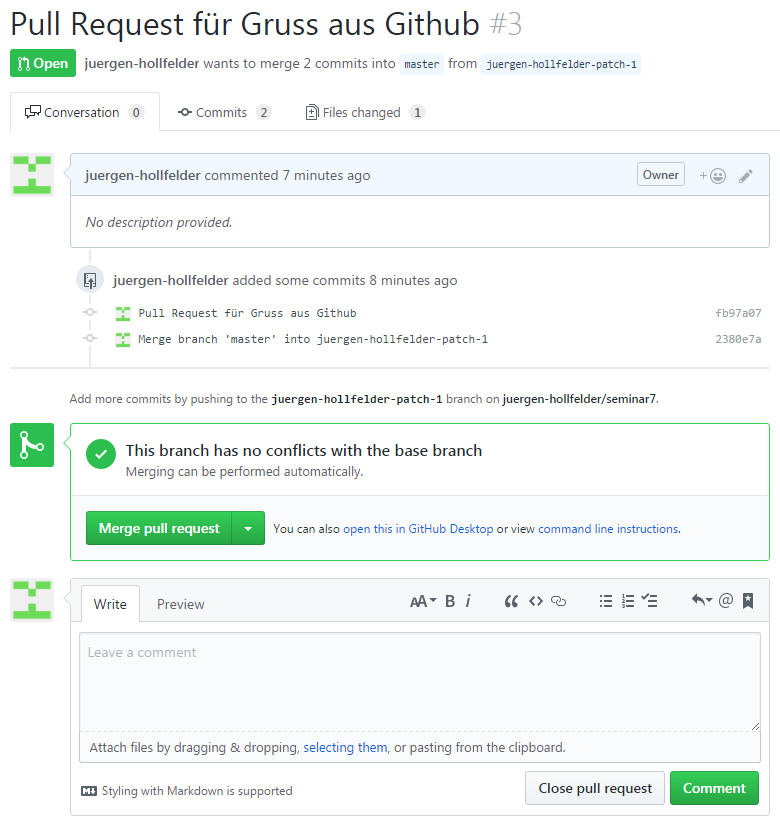


Edit:



Klick auf „Mark as resolved“ und „Commit changes“

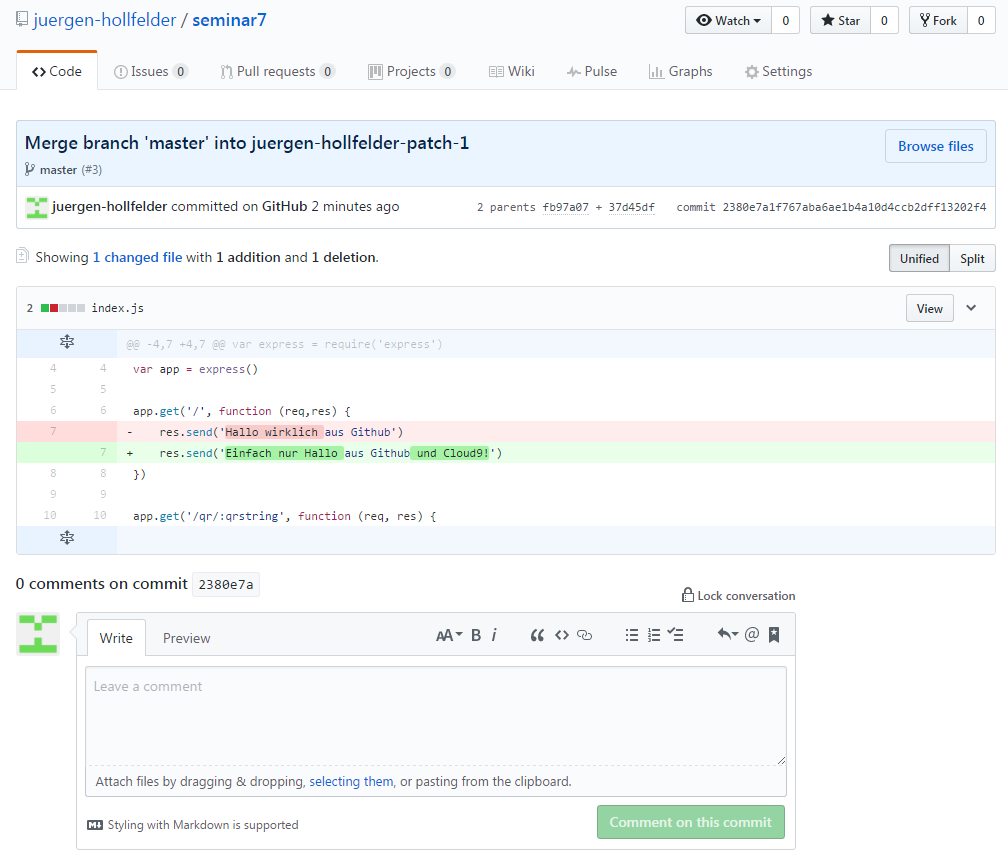
Jetzt ist alles wieder gut:



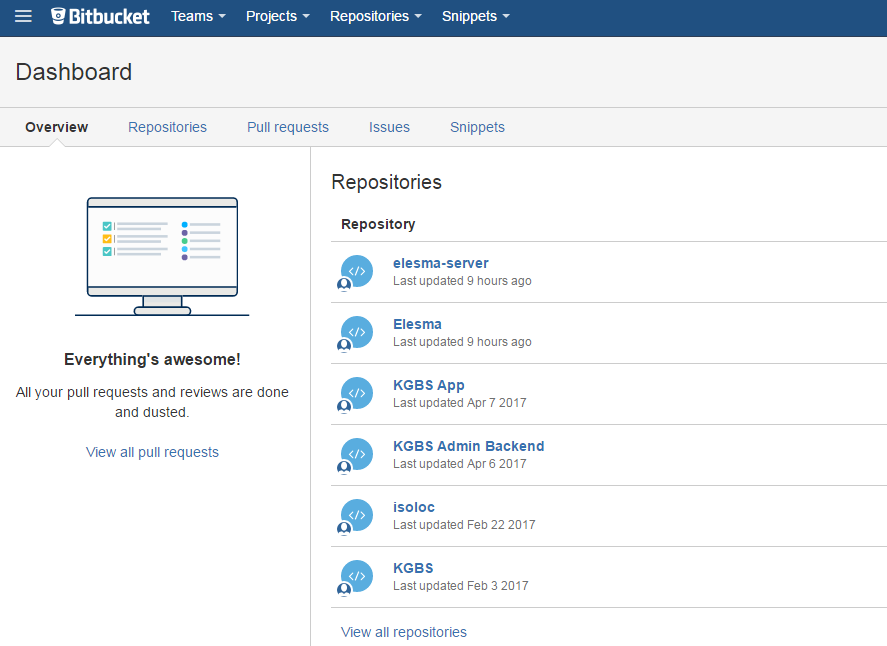
Klick auf „Merge pull request“ und es ist soweit fertig, den Branch der durch den Pull Request angelegt wurde, kann man löschen, das ist so üblich.



Man kann jetzt auch sehen, das der Pull-Request ein in den Master „hineingemergt“ wurde:



Es muss nicht immer Github sein. Der Nachteil bei Github ist, dass alle Repositories öffentlich sind, ausser man zahlt eine Gebühr. Bei Bitbucket erhält man Git Repositories für umsonst:



Ich persönlich finde es etwas besser und mit dem lokalen Tool „Atlassian Source Tree“ kommt man auch ganz gut zurecht.

Dort gibt es auch sehr gute Tutorials

<https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud>