Rheinische Friedrich-Wilhelms-Universität Bonn - Institut für Informatik III

Übungen zur Vorlesung

Softwaretechnologie

- Wintersemester 2017/18 -

Dr. Günter Kniesel

Übungsblatt 1

Zu bearbeiten bis: 20.10.2017, 16:00 Uhr

Fragen zu Übungsaufgaben respektive zur Vorlesung können Sie auf der Mailingliste swt-tutoren@lists.iai.uni-bonn.de, bzw. swt-vorlesung@lists.iai.uni-bonn.de stellen.

Aufgabe 1. *Branch & merge sowie vollständig dezentralisiertes Arbeiten* (15 Punkte)

Tipps zur Vorgehensweise in SmartGit sind mit SG gekennzeichnet.

a) **(1 Punkt)** Clonen Sie ssh://gitolite-se-swt@git.iai.uni-bonn.de/swt2017_readonly. Danach sollen Sie in Ihrer lokalen Kopie die Datei "briand-kellogg.txt" vorfinden. Darin steht ein Text der aus drei Abschnitten besteht.

Jeder von Ihnen soll *parallel zu den anderen Gruppenmitgliedern* auf seinem eigenen Rechner genau einen Abschnitt dieser Datei bearbeiten. Ordnen Sie sich selbst die Abschnitte zu, indem Sie ihre Vornamen alphabetisch sortieren. Der 1. Vorname bearbeitet den 1. Abschnitt, der 2. Vorname den zweiten Abschnitt, und so weiter. Wenn Sie mehr als 3 Teammitglieder sind, geht es zyklisch weiter. Dann gibt es mehrere Leute, die den gleichen Abschnitt bearbeiten.

b) **(3 Punkte)** Führen Sie folgende Änderungen durch:

a) Entsprechend Ihrer vereinbarten Aufgabenteilung pro Abschnitt, korrigieren Sie die Rechtschreibfehler in "ihrem" Abschnitt der Datei *briand-kellogg.txt*
b) Committen Sie die Änderungen in ihr lokales Repository.

Sie können nun ihre Änderungen nicht in das "readonly" repository zurück pushen, da Sie dazu sinnvollerweise keine Rechte haben (sonst würden diejenigen die das Repository später klonen nichts mehr zu korrigieren haben). Stattdessen arbeiten Sie nun mit Ihrem Teamkollegen auf dem eigenen Gruppenrepository weiter. Dazu müssen Sie als erstes:

c) **(1 Punkt)** Das Gruppenrepository als Remote eintragen (SG: "Remote" > "Add…" > *URL eintragen* > *Name für den Remote eintragen, z.B. MeineGruppe* > "Verify repository connection" *wegklicken* > "Add"). Bevor sie fortfahren ist zu beachten, dass sie nicht einfach in den Master-Branch des Remote-Repository pushen können ohne ihn vorher zu pullen. Ein einfacher Pull führt jedoch hier dazu, dass sie aus dem readonly-Repository pullen. Hier gibt es eine Möglichkeit das Problem zu lösen :

-> Sie können ihren lokalen Master-Branch synchron zu dem im Repository halten. Dazu Rechtsklick auf Master-Branch → Set Tracked Branch. Nun können sie normal fortfahren.

d) **(5 Punkte)** Versuchen Sie nun zu pushen (SG: "Push").Falls Sie der Gruppenschnellste waren, geht das problemlos. Ansonsten wird Git eine Fehlermeldung ausgeben. Dann müssen Sie zusätzlich:

(1) Die Änderungen die Ihre Kollegen schon gepusht haben zu sich pullen (SG: "Pull"). Git führt dabei automatisch einen "Merge" durch (SG: Sie können in den Optionen des "Pull"-Dialogfensters steuern ob ein "Merge" oder ein "Rebase" durchgeführt wird).

(2) Lösen Sie eventuell auftretende Konflikte. Sprechen Sie gegebenenfalls mit ihrem am Konflikt beteiligten Kollegen, wenn unklar ist, welche Konfliktlösung Sinn macht.

- SG: Konflikte werden angezeigt durch ein Dateisymbol mit einem roten Punkt in dem sich ein Ausrufezeichen befindet.
- SG: Den Konfliktlösungseditor von SmartGit starten sie durch Doppelklick auf die betroffene Datei oder durch Rechtsklick → *Conflict Solver*. Darin können Sie alle Änderungen der Datei nachverfolgen und die Konflikte auflösen indem Sie pro Änderung entscheiden, welcher Stand (ihrer oder der des Anderen) übernommen wird.

(3) Schauen Sie sich im Konfliktlösungseditor von SmartGit oder den Meldungen auf der Git-Bash-Kommandozeile genau um, so dass Sie verstehen, wann genau ein Konflikt auftritt. Versuchen Sie insbesondere zu verstehen welche der folgenden Fälle in Git zutreffen oder nicht:

1.i. Wenn die gleiche Datei von unterschiedlichen Leuten bearbeitet wurde gibt es einen Konflikt.
1.ii. Wenn die gleiche Zeile einer Datei von unterschiedlichen Leuten bearbeitet wurde gibt es einen Konflikt.
1.iii. Wenn in der gleichen Zeile von unterschiedlichen Leuten unterschiedliche Änderungen durchgeführt wurden gibt es einen Konflikt.
1.iv. Wenn … <Ihre eige*ne Formulierung, falls keine der obigen passt*> …

(4) Wenn alle Konflikte in der Datei gelöst sind speichern Sie die Datei und führen ein „Stage“ durch. Dadurch wird Git mitgeteilt, dass der neue Stand als Lösung des Konflikts betrachtet werden und Teil des nächsten „Commit“ sein soll. (SG: Der Konfliktlösungseditor fragt beim Speichern immer, ob für die Konfliktlösung ein "Stage" durchgeführt werden soll. Antworten Sie mit "Ja".)

(5) Anschließend committen Sie. Dadurch werden die durch das „stage“ kummulierten Konfliktlösungsänderungen (es kann ja sein, dass Sie viele Konflikte in vielen Dateien haben) gemeinsam ins Repository übertragen.

(6) Versuchen Sie nun erneut zu pushen Sofern nicht erneut jemand schneller war sollte es nun klappen. Ansonsten müssen Sie noch mal (1) – (5) durchführen.

e) **(5 Punkte)** Erzeugen Sie eine Datei "Konfliktdefinition.txt"

(1) (3 Punkte) Schreiben Sie darin das auf was ihr Fazit aus Schritt d)(3) war. Wenn sie der Erste waren, der Schritt (1) bis (5) nicht durchführen musste, schreiben Sie auf, was Sie aus der Vorlesung hinsichtlich der Konfliktbegriffes verstanden haben.

(2) (2 Punkte) Committen Sie die Datei und Pushen Sie sie.

(3) Falls andere schneller waren gehen Sie wie in Punkt d) vor. Wenn Konflikte auftreten, überschreiben Sie nicht die Vorschläge der Anderen, sondern sorgen Sie bei der

Konfliktlösung dafür, dass Ihr Formulierungsvorschlag als erkennbarer eigener Abschnitt unter den anderen steht (fügen Sie z.B. den eigenen Namen vor dem Abschnitt hinzu).

Aufgabe 2. *Dateioperationen mit Git* (12 Punkte)

In dieser Aufgabe erstellen Sie ein eigenes Repository und Erzeugen darin Dateien, benennen Sie um und Löschen manche wieder. Diese Aufgabe löst jeder für sich und fügt daher dem Ordner "Dateien" den eigenen Vornamen hinzu also z.B. "Dateien_Guenter".

a) **(1 Punkt)** Erzeugen sie lokal einen Ordner „Dateien_*Vorname*“ an, darin die Dateien „Datei 0.txt“ und „Datei 1.txt“ mit beliebigem, nicht-leeren Inhalt.

b) **(1 Punkt)** Erzeugen Sie lokal ein Git-Repository (SG: *Repository → Add or Create*…).

c) **(1 Punkt)** Führen Sie einen Commit der beiden Dateien durch. Erzeugen Sie nun einen Branch und wechseln Sie in diesen Branch.

d) **(1 Punkt)** Fügen Sie (auf Systemebene) in dem Ordner eine neue Datei „Datei 2.txt“ hinzu. Wechseln Sie nach SmartGit. Sehen Sie die neue Datei? Sehen Sie die alte Datei auch noch? Finden Sie heraus worüber Sie steuern können, ob sie alle Dateien / neue Dateien / veränderte Dateien im SmartGit-Fenster "Files" sehen.

e) **(1 Punkt)** Welches Symbol hat die neue Datei? Teilen Sie git mit, dass sie auch ins Repository übernommen werden soll, indem Sie die Datei auswählen und einen "Stage" durchführen. Wie verändert sich das Dateisymbol?

f) **(1 Punkt)** Schreiben Sie die Antworten zu d) und e) in die erste Datei (Datei 0.txt). Wie verändert sich das Dateisymbol nach dem Speichern? Schreiben sie auch die Antwort hierzu in die Datei.

g) **(1 Punkt)** Führen Sie nun einen Commit, mit den Änderungen, durch.

h) **(1 Punkt)** Benennen Sie nun (auf Systemebene) „Datei 2.txt“ in „Antworten.txt“ um, "Datei 1.txt" in "Eins.txt" und "Datei 0.txt" in "Null.txt". Was bemerken Sie, wenn Sie nach SmartGit zurückkehren? Wie lässt sich Ihre Beobachtung erklären? Schreiben Sie es in die Datei „Antworten.txt“ und führen Sie einen weiteren commit durch, der alle Änderungen beinhaltet (SG: Achten Sie darauf beim commit alle Dateien die sich geändert haben zu selektieren.).

i) **(1 Punkt)** Führen Sie folgende Schritte in einem Commit durch: Ändern Sie den Dateinamen von „Antworten.txt“ in „Antworten_final.txt“. Anschließend schreiben Sie folgende Überschrift in Ihre Datei: „Finale Antworten [DATUM VON HEUTE]“. Was fällt Ihnen auf? Schreiben Sie Ihre Antworten ebenfalls in „Antworten_final.txt“ und commiten Sie letztendlich alle Ihre Änderungen.

j) **(1 Punkt)** Wechseln Sie in den Hauptbranch (master) zurück, wählen Sie dabei beim Checkouts, *falls* Sie danach gefragt werden „save stash“ aus. Ändern Sie den Inhalt von "Datei 1.txt".

k) (1 Punkt) Führen Sie ein "merge" mit dem Stand des anderen branches durch. Vergleichen Sie genau was problemlos geht und wo Konflikte auftreten. Achten Sie vor allem darauf, wie Git mit den beiden umbenannten Dateien umgeht. Können Sie den Unterschied erklären?

l) **(1 Punkt)** Comitten Sie und pushen Sie das Ergebnis in ihr Gruppenrepository.