3. Praktikumstermin: Automatischer Build mit Jenkins

Jenkins (https://jenkins.io) ist ein mächtiges Tool, das unter anderem den aktuellen Code aus Ihrem Git-Verzeichnis abrufen, analysieren, kompilieren und testen kann. Dieses Tool werden wir heute kennenlernen, damit Sie bequem die Doxygen-Dokumentation erstellen können, aber auch viele weitere Möglichkeiten für die folgenden Praktikumstermine haben. Nach diesem Praktikumstermin haben Sie den vollständigen Arbeitszyklus für das weitere Praktikum kennengelernt und eingeübt.

Wie in jedem Praktikumstermin arbeiten Sie weiter am CocktailPro. Die Projektdateien sind natürlich in Git versioniert.

3.1. Vorbereitung - Kommentare im Code

Der Code ist vollständig und sinnvoll gemäß der vorgegebenen Richtlinien für Doxygen dokumentiert. Alle Teammitglieder haben in einem eigenen Branch einen Teil des Codes kommentiert. Dieser Branch lässt sich auf Ihrer lokalen IDE problemlos kompilieren und ausführen. Dann (und erst dann!) haben Sie Ihren Branch gemerged und auf dem zentralen Git-Server eingecheckt. Sie haben über das Webfrontend des Git_Servers (https://code.fbi.h-da.de) geprüft, dass der Code auch tatsächlich im Master-Branch des zentralen Repository vorliegt. Die zugehörigen Tickets in GitLab sind geschlossen.

3.2. Anbindung & Nutzung von Jenkins: Automatisches Bauen des Master-Branches

Jenkins

Jenkins ist ein mächtiges Tool, das Sie im Lauf des Praktikums besser kennen lernen werden.

Im Labor läuft ein Jenkins-Server und steht allen Teams mit je einem Projekt zur Verfügung. Die Oberfläche ist webbasiert. Ihr Betreuer zeigt Ihnen den Zugang zu Ihrem Jenkins-Projekt.
Da Jenkins im gesamten Praktikum Ihr Frontend für die Erstellung der Software sein wird, legen Sie am besten eine Verknüpfung zu Ihrem Jenkins-Projekt im Browser oder auf dem Desktop an. **Achtung!** Der Jenkins Server (https://cs12.fbi.h-da.de/jenkins) ist aus Sicherheitsgründen nur innerhalb des Hochschul-Netzwerks erreichbar. Um auch von zu Hause Jenkins aufrufen zu können, müssen Sie eine VPN-Software installieren und einrichten. Beachten Sie dazu die Hinweise am Anfang dieses Dokuments.

Jenkins - Der erste Build

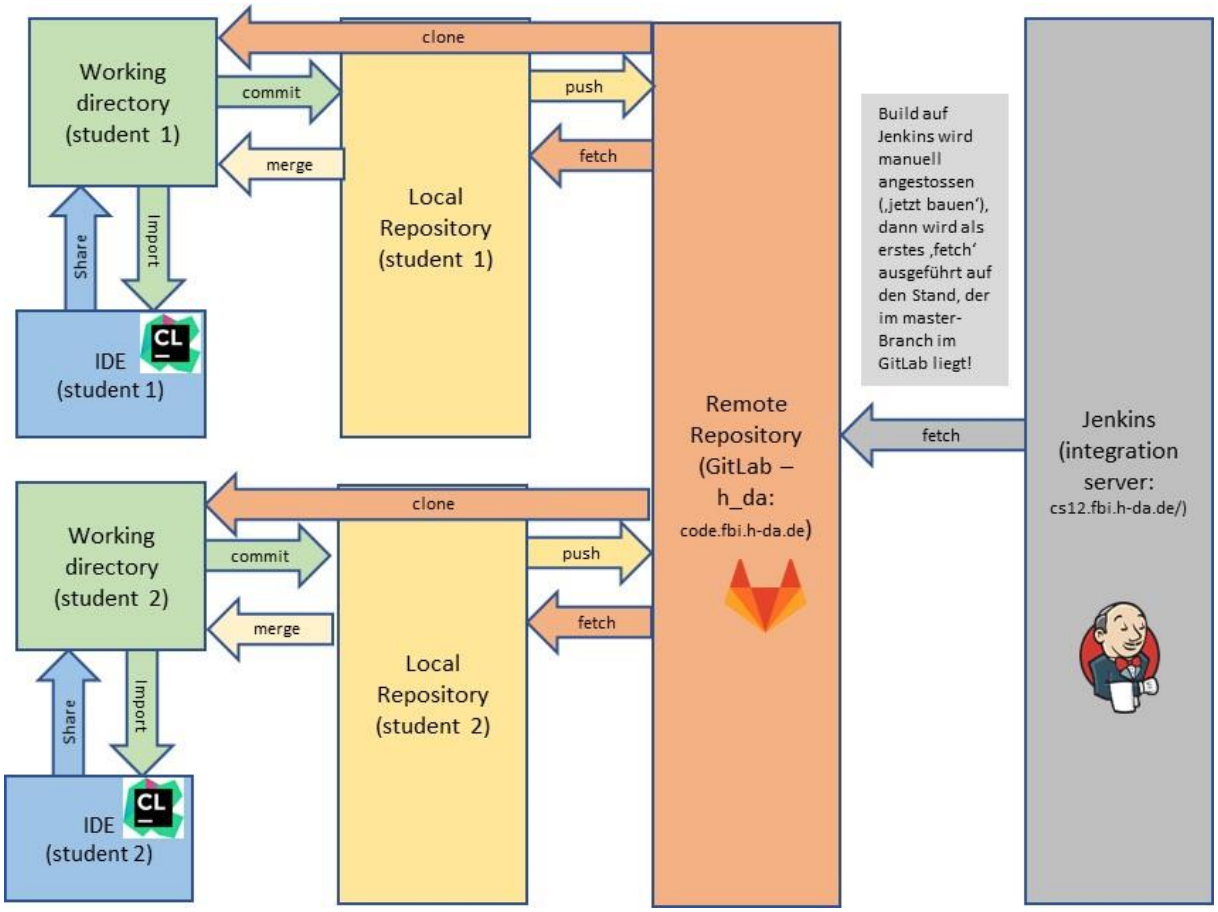
Ihr Jenkins-Projekt ist bereits für das Praktikum konfiguriert. Damit Jenkins den Quellcode aus dem Git-Repository abruft und diverse Aufgaben durchführt, klicken Sie auf "Jetzt bauen". Der Build sollte erfolgreich verlaufen.

"Spielen" Sie ruhig ein wenig mit Jenkins (aber bitte ändern Sie nicht die Konfiguration).

- Schauen Sie sich die verschiedenen Menüpunkte an.
- Suchen Sie die Ausgaben des Compilers in der Konsolen-Ausgabe

Ist Ihnen aufgefallen, dass Ihre Kommentare im Quellcode fehlen? Bisher verwendet Jenkins das ursprünglich zugelieferte Projekt und nicht Ihren kommentierten Quellcode. Das werden wir gemeinsam ändern. Ihre Betreuer werden gemeinsam mit Ihnen die Anbindung von Jenkins an den Quellcode Ihrer Gruppe vornehmen.
BITTE befolgen Sie dabei genau die Anweisungen Ihrer Betreuer. Dieser Schritt wird nur einmal von genau einem Gruppenmitglied gemacht!

Danach sind Sie bereit für den kompletten CI / CD Arbeitszyklus:



Doxygen-Dokumentation

Mit Hilfe von Doxygen (www.doxygen.org) können Kommentare aus Quellcode extrahiert und zu einer gut lesbaren Dokumentation aufbereitet werden. Details zur Dokumentation mit Doxygen finden Sie unter www.doxygen.org/manual/docblocks.html.

Beim Build hat Jenkins bereits eine Doxygen-Dokumentation erzeugt. In der Jenkins-Oberfläche für Ihr Projekt finden Sie die Einträge "Doxygen Document" und "Doxygen Warnungen".

Schauen Sie sich beide Ergebnisse an und achten Sie besonders auf die Warnungen. Bis zum nächsten Praktikumstermin müssen alle verbleibenden Doxygen-Warnungen durch (sinnvolle) Kommentare beseitigt werden.

3.3. Der vollständige CI / CD Arbeitszyklus

Verstehen Sie, wie Sie mit Jenkins arbeiten müssen!

- Aktualisieren Sie Ihr lokales Verzeichnis mit dem Quellcode
- Erzeugen Sie einen Branch und veröffentlichen Sie diesen im zentralen Repository
- Bauen Sie einen Syntaxfehler in eine Quellcode-Datei ein
- Checken Sie die fehlerhafte Datei ein (Commit)

- Mergen Sie die fehlerhafte Datei in Ihr lokales Repository
- Übertragen Sie die Änderung an das zentrale Repository (Push)
- Starten Sie den Build mit Jenkins
- Schauen Sie was passiert.

Anschließend machen Sie die fehlerhafte Änderung rückgängig und stellen sicher, dass der Jenkins-Build wieder erfolgreich durchläuft.

Überlegen Sie gemeinsam mit Ihren Teammitgliedern, wie Sie arbeiten müssen, damit Ihr CocktailPro für Sie und Ihre Kollegen möglichst immer kompilierbar und lauffähig ist. Leiten Sie daraus eine Regel für Ihre Zusammenarbeit ab!

Schreiben Sie die Regel hier auf (und befolgen Sie die Regel in Zukunft!):

Üben Sie den Arbeitszyklus

Bauen Sie das Projekt in Ihrer lokalen IDE und sehen Sie, ob es Compiler-Warnungen gibt.

Entfernen Sie jetzt die Warnungen, die der Compiler liefert, um den Arbeitszyklus und den Umgang mit Ihrer Build-Umgebung zu üben. Teilen Sie die Warnungen so auf, dass jedes Teammitglied mindestens eine Problemstelle korrigiert. Alle Teammitglieder verwenden einen Branch für die Korrektur und checken den Code schließlich in das gemeinsame Repository ein.

3.4. Statische Codeanalyse mit CppCheck

CppCheck ist ein Tool, das Quellcode auf verdächtige und problematische Konstrukte untersucht. Dabei werden frühzeitig potenzielle Fehler entdeckt, die in der Betriebsphase zu Abstürzen oder unerwartetem Verhalten führen können. Jenkins hat diese Analyse bereits durchgeführt.

In der Jenkins-Oberfläche finden Sie "CppCheck Results". Schauen Sie sich die Befunde an. Bis zum nächsten Termin sollen Sie alle diese Probleme verstehen und beheben.

Auf der rechten Seite im Jenkins sehen Sie auch eine Trend-Darstellung für CppCheck. Wenn Sie die Probleme beheben und Jenkins starten, sollte die Grafik entsprechende Änderungen anzeigen.

Dokumentieren Sie in einer gemeinsamen Datei "CppCheck-Fixes.txt", welche Maßnahme Sie für welche Meldung durchgeführt haben. Erstellen Sie die anfangs leere Datei und checken Sie die Datei im Hauptverzeichnis ihres Git-Projekts ein.

3.5. Nachbereitung

Folgende Abnahmepunkte müssen bis zum Abgabetermin für den nächsten Praktikumstermin erfüllt sein:

3.5.1. Doxygen

Beseitigen Sie eventuell verbliebene Doxygen Warnungen. In der Trendgrafik "Doxygen-Warnungen" auf der rechten Seite in Jenkins sehen Sie (und Ihr Betreuer!) schnell, ob es noch Probleme gibt. Prüfen Sie in Jenkins, ob die erzeugte Doxygen-Doku die erwarteten Inhalte hat bzw. ob noch etwas fehlt.

3.5.2. Compiler-Warnungen

Verstehen und beseitigen Sie alle eventuell verbliebenen Warnungen des Compilers.

3.5.3. CPP-Check

Beseitigen Sie bis zum nächsten Termin alle Probleme, die CppCheck bemängelt.
Dokumentieren Sie in der Datei "CppCheck-Fixes.txt", welche Maßnahme Sie für welche Meldung durchgeführt haben.

3.5.4. Arbeit mit Git

Verwenden Sie Git, Tickets und Branches. Achten Sie darauf, dass jedes Teammitglied im eigenen Branch nachweislich bei der Mängelbehebung mitgemacht hat.

3.5.5. CI / CD Arbeitszyklus

Achten Sie darauf, dass zum Abgabetermin ein lauffähiger Entwicklungsstand in Git eingecheckt ist, der die obigen Anforderungen erfüllt. Bauen Sie dazu zur Sicherheit (rechtzeitig vor der Abgabe) nochmal das gesamte Projekt auf dem Jenkins-Server und kontrollieren Sie die Ergebnisse.