

EP 2 Gitlab Tutorial

Diese Anleitung hilft Ihnen Ihre Angabe in Git zu finden und die Abgabe Ihrer Lösung mittels Git durchzuführen.

Benötigte Software

Die benötigte Software ist auf den Rechnern im Informatik-Labor bereits installiert. Wir empfehlen Ihnen diese auch auf ihren privaten Geräten zu installieren, da Sie diese Tools auch in vielen anderen Lehrveranstaltungen verwenden werden.

IntelliJ IDEA

Wir empfehlen die Verwendung von *IntelliJ IDEA Community-Edition*.

- Im Angabe-Template wird ein IntelliJ IDEA-Projekt mitgeliefert.
- Auf den Labor-Rechnern wird bei den Übungen IntelliJ IDEA verwendet.
- Für die Programmiertests wird IntelliJ IDEA verwendet.
- TutorInnen können Ihnen bei Problemen im Tutorium (Pong Mi 14:30-16:30) helfen.

Bei anderen Entwicklungsumgebungen kann von Seiten des LVA Teams kein Support angeboten werden.

Java

Wir verwenden *Java Development Kit (JDK) 21*. Sie können **Oracle JDK 21** verwenden oder OpenJDK 21 über den Package Manager installieren. (Die Bilder in diesem Tutorial wurden allerdings mit einer älteren Version gemacht, genauso bei der restlichen Software.)

Sie können auch ein JDK mit neuerer Version verwenden. Achten Sie dabei darauf, das Sprachlevel in [Einrichtung in IntelliJ IDEA](#) auf 21 zu setzen.

Das **JDK** enthält den Java Compiler und eine Java Virtual Machine, um die vom Compiler erzeugten Dateien auszuführen. Es ist ausreichend ein JDK zu installieren, eine zusätzliche Installation einer JRE (Java Runtime Environment) ist nicht notwendig.

Git

IntelliJ IDEA benötigt ein installiertes *Git* Executable. Am einfachsten kann *Git* von der [Git Download-Seite](#) oder über dem Package Manager Ihres Systems installiert werden.

Es gibt jedoch auch verschiedene grafische Oberflächen (GUIs) für *Git*, die das Executable oft mitliefern. Die Installation einer solchen GUI ist nicht notwendig, da *IntelliJ IDEA* bereits eine *Git*-Oberfläche anbietet.

IntelliJ IDEA erkennt unter Windows, wenn *Git* im *Windows Subsystem for Linux (WSL)* installiert ist und verwendet diese Installation automatisch.

GitLab

Wir verwenden einen **GitLab-Server (b3.complang.tuwien.ac.at)** zur Verwaltung Ihrer Git-Repositories, über die auch die Angaben bereitgestellt werden. Neben dem Zugriff direkt über die IDE bietet GitLab auch ein Web-Interface.

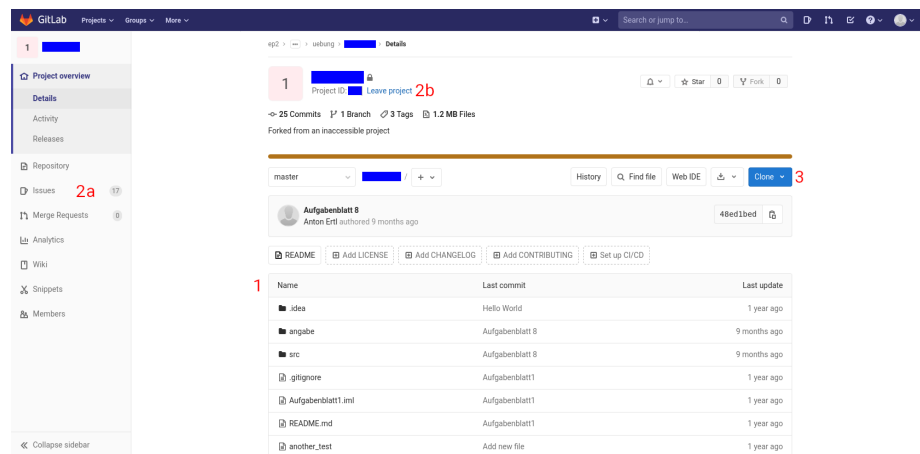
Ein Account wurde für alle Teilnehmenden der LVA automatisch erstellt. Sie können mittels **Passwort vergessen** jederzeit ein neues Passwort setzen. Dabei kann es einige Stunden dauern, bis Sie die Passwort-Änderungs-Mail erhalten.

Repository

Ihr gesamter Source-Code wird in einem sogenannten Repository gespeichert, über das auch regelmäßig die neuen Angaben veröffentlicht werden. Eine Liste Ihrer Repositories für die Übung wird auf der GitLab-Startseite nach dem Login angezeigt:



Der Link führt Sie zur Projektseite Ihres Repositories:



Die Übersichtsseite sowie die über die Navigation erreichbaren Unterseiten bieten eine Fülle an Funktionen, die Sie für EP2 üblicherweise nicht brauchen.

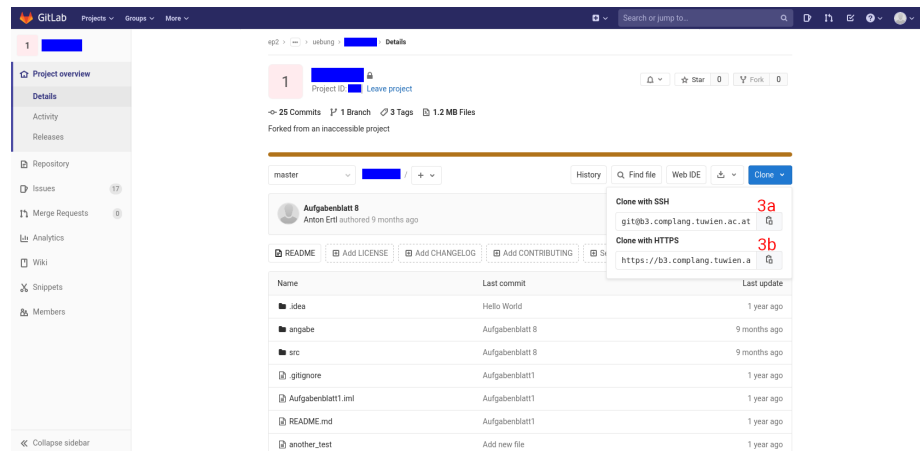
Für Sie relevant sind:

Dateiliste (1) *Git* speichert intern die Versionen der Dateien, die Sie bei einem [Commit](#) speichern. *Gitlab* zeigt Ihnen die aktuelle Version der Dateien in dieser Ansicht an. Die Dateien, die hier angezeigt werden, werden für die Abgaben berücksichtigt. Stellen Sie also sicher, dass Sie die gewünschte Version zum Zeitpunkt der Abgabe mittels eines [Pushes](#) auf *Gitlab* geladen haben.

Issues (2a) Ihre Tutorin oder ihr Tutor kann Ihnen über so genannte Issues Feedback zu ihren Aufgaben und den Übungstests geben. Im Normalfall wird das Feedback zum Aufgabenblatt jedoch über die Bewertung in TUWEL gegeben. In Ausnahmefällen kann hier aber noch ein ausführlicheres Feedback eingetragen werden. Bei der Erzeugung eines Issues wird Ihnen eine E-Mail zugesandt. Sie müssen diesen Menüpunkt also nicht regelmäßig überprüfen.

Leave project/Projekt verlassen 2b: ACHTUNG: Über diesen Link entfernen Sie ihre Mitgliedschaft im Projekt und können das Projekt nicht weiter bearbeiten. Klicken Sie **NICHT** auf diesen Link. Sollte es Ihnen doch passieren, melden Sie sich bei ihrer Tutorin oder ihrem Tutor.

Durch Klick auf *Clone (3)* öffnet sich ein Dialog, der URLs für den Import mit *IntelliJ* enthält:

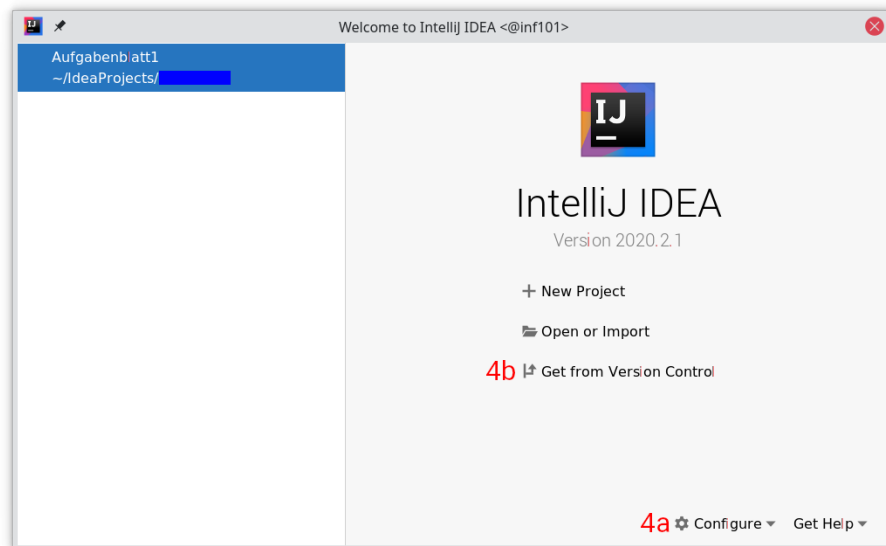


Clone with HTTPS (3b) ist die einfachere Methode und wird anschließend erklärt.

Clone with SSH (3a) wird von erfahrenen Benutzern bevorzugt und im [Anhang](#) erklärt.

IntelliJ IDEA

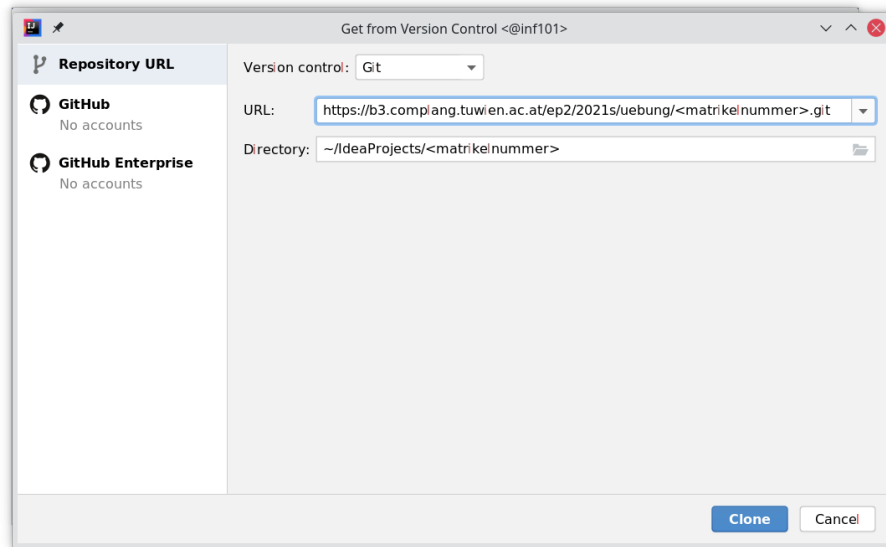
Wenn Sie *IntelliJ IDEA* das erste Mal öffnen, wird Ihnen folgendes Fenster angezeigt:



Projekt Importieren

Um das Projekt zu importieren klicken Sie auf *Get from Version Control* (4b). Sollten Sie bereits ein Projekt geöffnet haben, können Sie das Projekt über das Menü importieren: *File > New > Project from Version Control* öffnet dasselbe Fenster wie (4b).

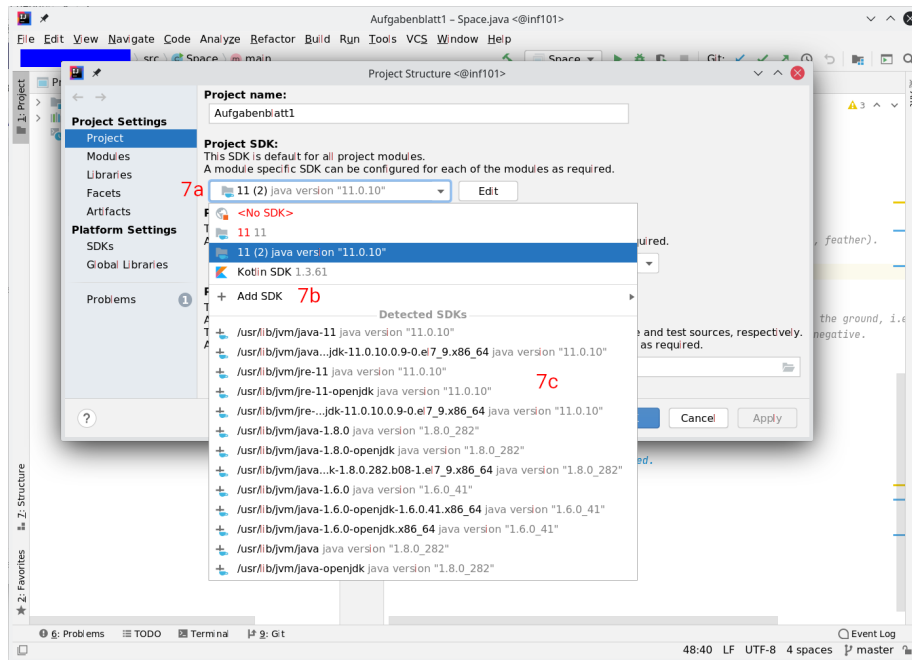
Sie müssen dazu die URL des Repositories (aus 3b) angeben und können optional das lokale Verzeichnis für das Projekt verändern. Klicken Sie anschließend auf *Clone* um das Importieren abzuschließen. Wenn Sie *Clone with HTTPS* verwenden, werden Sie nach den Zugangsdaten für *GitLab* gefragt.



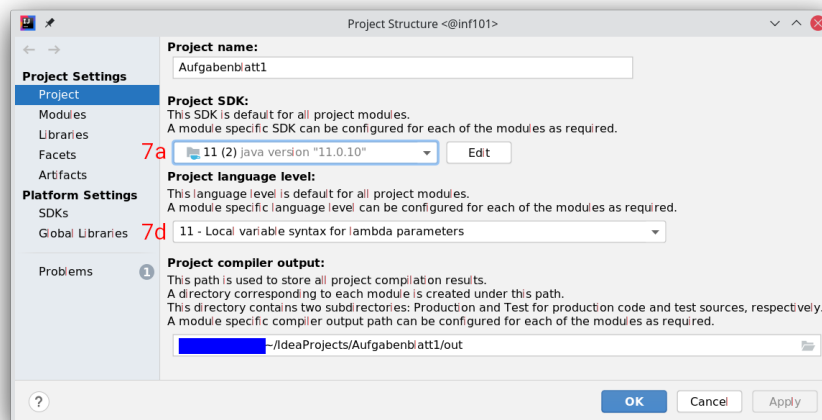
Einrichtung in IntelliJ IDEA

Das importierte Projekt enthält keine Vorkonfiguration zum verwendeten JDK (Java Development Kit), nachdem hier Abweichungen zwischen verschiedenen Rechnern bestehen können. Sie müssen diesen daher selbst setzen, bevor Sie das Projekt verwenden können.

Öffnen Sie dazu *File > Project Structure* und setzen dort unter **(7a)** das zu verwendende JDK 21. Falls das JDK nicht automatisch von IntelliJ IDEA erkannt wurde, müssen Sie unter *Add SDK* **(7b)** manuell den Pfad zur Java-Installation angeben (üblicherweise in `C:\Program Files\Java` für Windows bzw. `/usr/lib/jvm` für Linux-Systeme).



In *Detected SDKs* (7c) wird Ihnen eine Liste der erkannten JDKs und JREs angezeigt ((7c) zeigt eine Liste der installierten JDKs und JREs aus früheren Jahren). Wählen Sie aus diesen Einträgen einen Eintrag mit der Version 21 und ohne "jre" im Namen aus. (In der gezeigten Liste ist der Eintrag `jre-11-openjdk` kein JDK!).



Achten Sie auch darauf, dass Sie das richtige Sprachlevel (*Project Language Level* (7d)) wählen. Das Sprachlevel dieser LVA ist dieses Jahr 21 (das Bild stammt

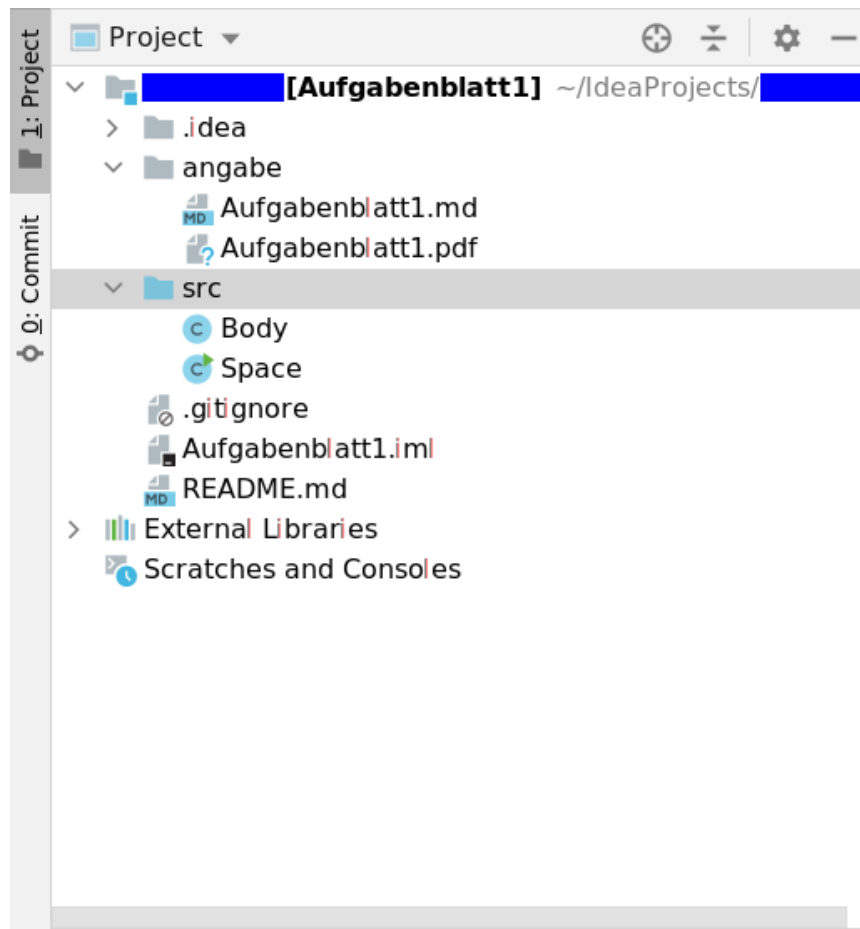
aus einem früheren Jahr).

Wenn Sie ein zu niedriges Sprachlevel verwenden, können Sie einige Sprachfunktionen nicht verwenden. Wenn Sie ein zu hohes Sprachlevel wählen, kann es sein, dass Sie Übersetzungsfehler bekommen, die Ihnen *IntelliJ IDEA* vor dem Übersetzen nicht anzeigt.

Projektstruktur

Die Angaben werden kurz nach der Abgabe der vorherigen Übung auf ihren Repositories eingespielt. Sie finden die Angaben dann jeweils im Verzeichnis **angabe** über die Projekt-Ansicht in *IntelliJ IDEA*. Die Angabe als Markdown-Datei verfügbar (z.B. `Aufgabenblatt1.md`, die Sie direkt in *IntelliJ IDEA* betrachten können.

Im Ordner `src` befinden sich die Dateien mit dem Quellcode für die Aufgaben. Beachten Sie bitte die Hinweise in den Angaben zur Erstellung neuer Dateien, um Konflikte mit zukünftigen Angaben zu vermeiden.



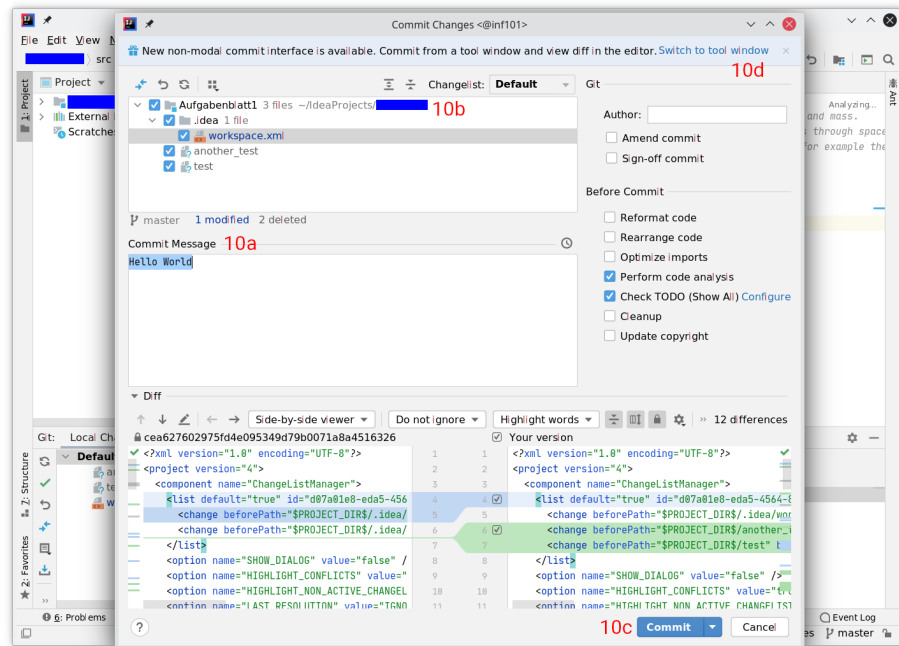
Commit

Mit einem Commit wird in *Git* ein "Schnappschuss" der Dateien in ihrem Projekt erstellt und lokal gespeichert. Wir empfehlen Ihnen, möglichst oft einen Commit zu machen, um keine Änderungen zu verlieren. Mit *Git* ist es auch möglich, eine frühere Version eines Projektes wiederherzustellen. Falls Sie eine Wiederherstellung benötigen, fragen Sie bitte bei Ihrer Tutorin oder Ihrem Tutor nach.

ACHTUNG: Damit die Änderungen, die Sie gemacht haben, auch für eine Abgabe berücksichtigt werden können, müssen Sie diese auch pushen (siehe dazu Abschnitt [Push](#)).

Verwenden Sie das grüne Häkchen rechts oben um den Commit Dialog zu öffnen. In neueren Versionen von *IntelliJ IDEA* wird Ihnen angeboten, das *Commit Tool Window* zu benutzen. Beide Varianten bieten die für EP2 benötigte Funktionalität. Um das *Commit Tool Window* in Zukunft zu benutzen, klicken

Sie auf *Switch to tool window* (10d), das *Commit Tool Window* wird dann geöffnet und kann über einen Reiter links wieder geöffnet werden.

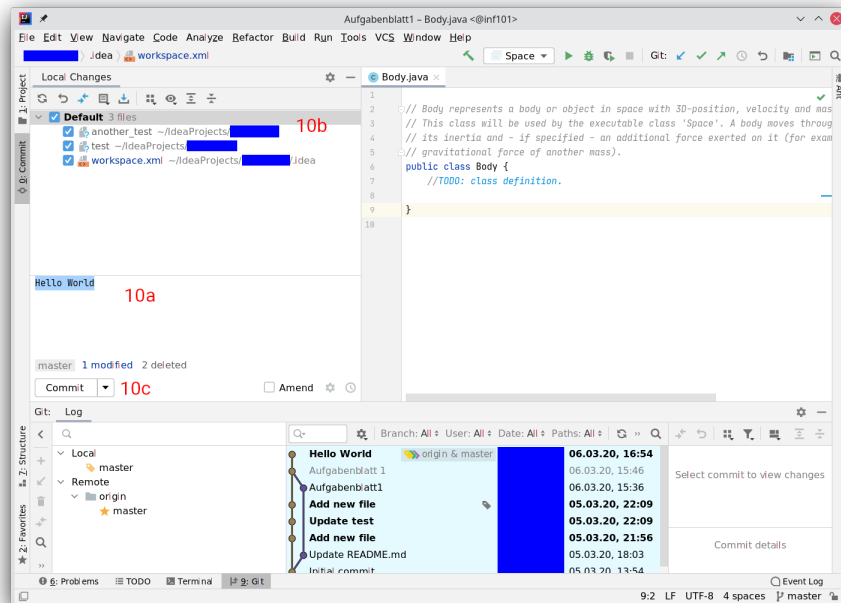


Sowohl der Dialog als auch das *Commit Tool Window* benötigen dieselben Eingaben, um einen Commit durchführen zu können:

Commit Message (10a) Geben Sie hier eine Nachricht ein, mit der Sie die Änderungen, die Sie durchgeführt haben, beschreiben. Es empfiehlt sich, hier genau zu sein, damit Sie Änderungen leicht wiederfinden können.

Changelist (10b) Geben Sie hier die Dateien an, die Sie in diesem "Schnappschuss" inkludieren möchten. In den meisten Fällen können Sie hier alle Dateien auswählen, die Ihnen angeboten werden. Im *Commit Tool Window* kann es vorkommen, dass Sie jedes Mal die Changelist **Default** neu anwählen müssen.

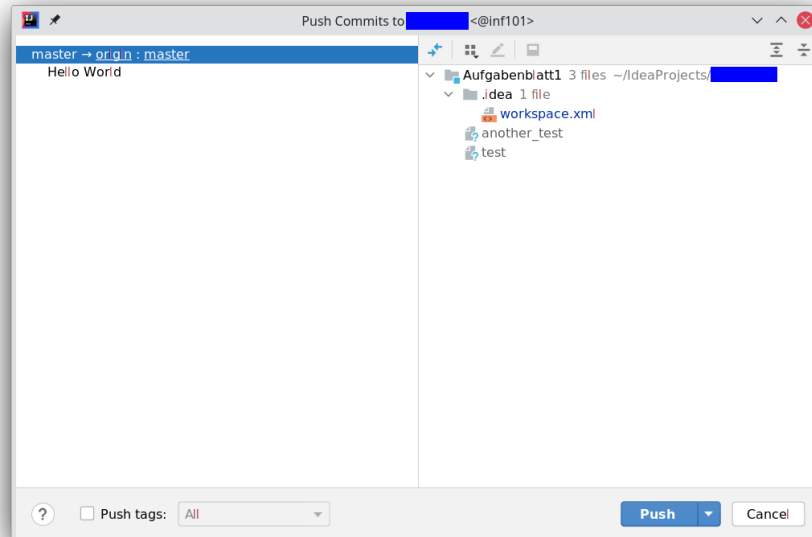
Commit (10c) Sobald Sie die Dateien ausgewählt und die *Commit Message* gewählt haben, können Sie den Commit erzeugen. Sie können auch durch das Klicken auf den Pfeil nach unten im rechten Teil der Schaltfläche *Commit and Push* auswählen. Dadurch wird direkt ein **Push** eingeleitet.



Wir empfehlen, oft und auch für kleine Änderungen einen Commit zu machen und diesen zu pushen.

Push

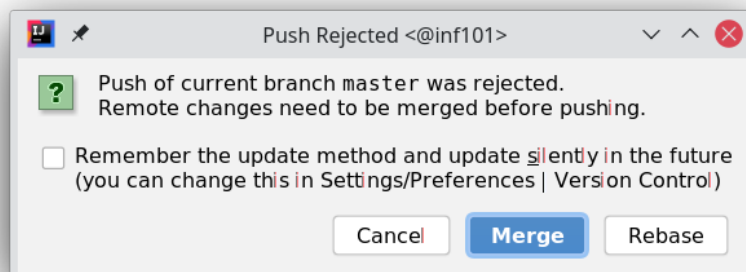
Damit die Commits nicht nur lokal gespeichert werden, müssen diese gepusht werden. Dadurch werden Commits auf den Git-Server übertragen. Sie können entweder direkt *Commit and Push* verwenden (empfohlen) oder den Push Dialog über das Menü *VCS > Git > Push* öffnen.



Mit einem Klick auf die Schaltfläche *Push* werden die Commits auf den *Gitlab* Server übertragen. Die anderen Optionen für die *Push* Schaltfläche werden Sie für EP2 voraussichtlich nicht benötigen.

Nach einem Push können Sie ihre lokalen Dateien mit den Dateien auf der *Gitlab* Weboberfläche vergleichen um sicherzustellen, dass der Push erfolgreich war.

Wenn Sie auf mehreren Geräten arbeiten (z.B. Laborrechnern) oder Sie einen Push durchführen, nachdem die Angabe eingespielt wurde, kann es vorkommen, dass Sie folgende Fehlermeldung angezeigt bekommen:

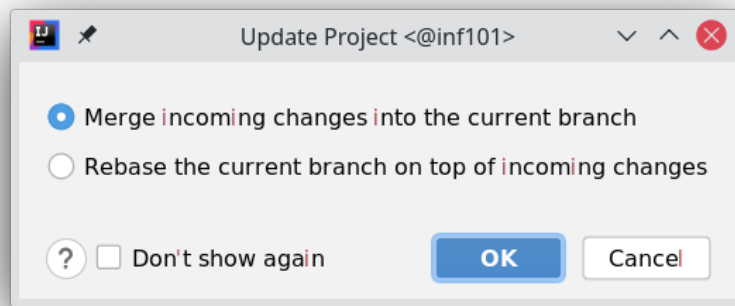


Diese Fehlermeldung bedeutet, dass Änderungen auf dem *Gitlab* Server durchge-

führt worden sind, die Sie noch nicht lokal gespeichert haben. Damit keine Inkonsistenzen in ihrem Repository entstehen können, müssen Sie diese zuerst lokal speichern und in ihr Repository integrieren. Klicken Sie dazu auf die Schaltfläche *Merge*. *Git* und *IntelliJ IDEA* werden dann versuchen, die verschiedenen Versionen der Dateien automatisch zusammenzuführen. Sollte das nicht funktionieren, müssen sie die Konflikte manuell beheben (*IntelliJ IDEA* bietet dafür sehr gutes Tooling an).

Pull

Nachdem neue Angaben veröffentlicht wurden oder wenn Sie den Rechner wechseln, müssen Sie Pull verwenden. Hier können Sie den blauen Pfeil rechts oben verwenden. Am besten gewöhnen Sie sich an, immer Pull zu verwenden, bevor Sie mit der Arbeit beginnen.



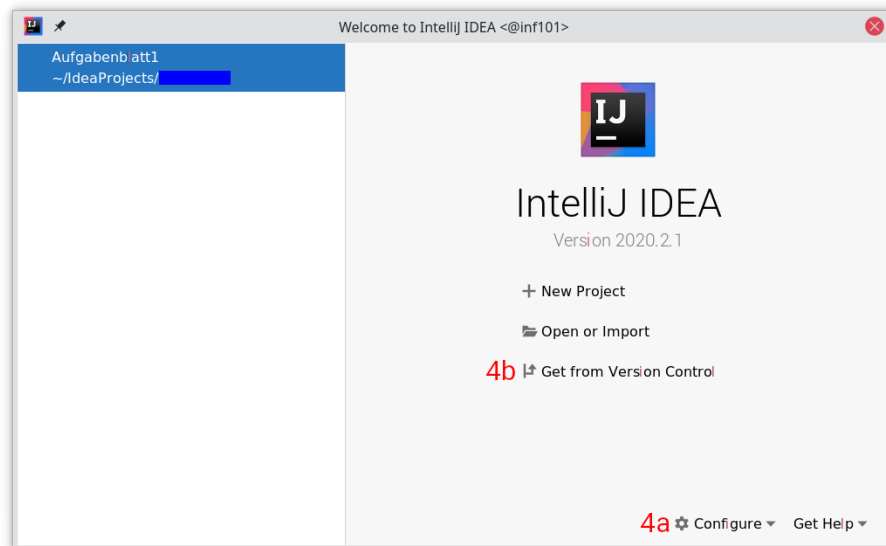
Wir wünschen Ihnen viel Erfolg beim Bearbeiten Ihrer Übungsaufgaben.

Appendix

Einstellungen

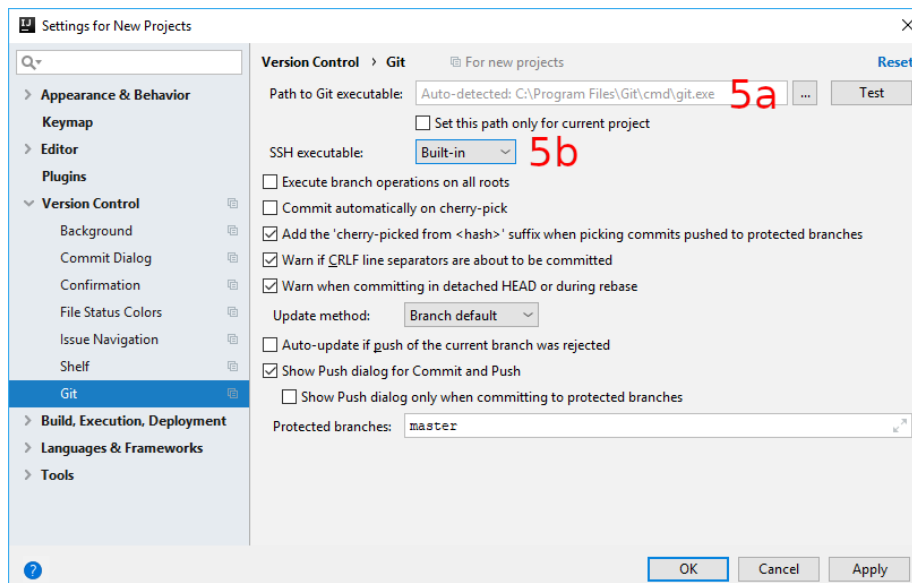
Im Bestfall wurde das installierte Git Executable automatisch erkannt (**5a**), ansonsten müssen Sie hier den Pfad zu `git.exe` (bzw. `git` unter UNIX-basierten Systemen) manuell angeben.

Navigieren Sie dazu unter *Configure* (**4a**) > *Settings* (oder alternativ *File* > *Other Settings* > *Settings for New Projects*, falls ein Projekt geöffnet ist) zu *Version Control* > *Git*.



Falls Sie *Clone with SSH* verwenden wollen, müssen Sie außerdem einstellen, welche SSH-Implementierung verwendet werden soll (5b):

- Unter Windows ist es meist am einfachsten, den in IntelliJ IDEA integrierten SSH-Client (*Built-in*) zu nutzen.
- Falls ein SSH-Client auf Ihrem System installiert ist (auf Linux-Systemen meist standardmäßig) ist dessen Verwendung (*Native*) zu bevorzugen.



SSH-Authentifizierung mit Keys

Anmerkung: Für diese LVA ist es nicht zwingend erforderlich, SSH-Keys zu verwenden, *Clone with HTTPS* ist an sich ausreichend. Es ist allerdings empfehlenswert, eine solche Einrichtung vorzunehmen, da Sie sie mit hoher Wahrscheinlichkeit früher oder später in anderen LVAs oder außerhalb der Universität benötigen werden.

SSH-Keys ermöglichen eine Authentifizierung bei SSH-Servern, ohne Passwörter zu übertragen. Für GitLab stellen sie die einzige Möglichkeit der Authentifizierung für *Clone with SSH* dar.

Falls Sie bereits SSH-Keys haben, können Sie diese einfach weiterverwenden und müssen keine neuen generieren (Ausnahme: DSA-Keys werden seit mehreren Jahren als unsicher angesehen und von unserer GitLab-Installation daher nicht akzeptiert). Fahren Sie in diesem Fall direkt mit dem Abschnitt **SSH-Key in GitLab importieren** fort.

Wenn Sie mehrere Rechner verwenden, müssen Sie die nachfolgenden Schritte auf jedem Rechner durchführen.

SSH-Key generieren SSH-Keys können beispielsweise mit dem Commandline-Tool `ssh-keygen` generiert werden. Dieses wird üblicherweise mit SSH-Clients mitinstalliert und ist auch in der Git-Installation von git-scm.com enthalten.

Öffnen Sie dazu ein Terminal-Fenster oder das Programm *Git Bash* und führen Sie `ssh-keygen` aus.

Die Default-Parameter können unverändert genutzt werden (außer gegebenenfalls bei stark veralteten Programmversionen). Für nähere Informationen zu den verfügbaren Optionen sei auf die [Dokumentation von ssh-keygen](#) verwiesen.


`ssh-keygen` fragt gegebenenfalls nach, ob existierende Keys überschrieben werden sollen. Sie sollten existierende Keys nicht überschreiben, es sei denn Sie sind sich sicher, dass diese nirgendwo verwendet werden.

SSH-Key in GitLab importieren Zuletzt müssen Sie den erstellten Key in GitLab importieren.

Navigieren Sie dazu in das Verzeichnis, in dem die Keys erstellt wurden (standardmäßig `.ssh` im Home-Verzeichnis des Users). Öffnen Sie `id_*type*.pub` mit einem Texteditor und kopieren Sie den Inhalt.

ACHTUNG: Geben Sie auf keinen Fall den Inhalt der Datei `id_*type*` an oder diesen weiter. Dadurch ist es anderen Personen möglich sich als Sie auszugeben!

In GitLab gelangen Sie über *Settings* im User-Menü rechts oben zum Formular für SSH-Keys. Fügen Sie dort den Public Key ein (**9a**).


 GitLab

Projects ▾

Groups ▾

More ▾

Search or jump to...



User Settings

Profile

Account

Applications

Chat

Access Tokens

Emails

Password

Notifications

SSH Keys

GPG Keys

Preferences

Active Sessions

Authentication log

<< Collapse sidebar

User Settings ▾

SSH Keys

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

To add an SSH key you need to [generate one](#) or use an [existing key](#).

Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.

ssh-rsa ABCDE... me@my-pc

9a

Title

me@my-pc

Name your individual key via a title

Add key

Your SSH keys (0)

There are no SSH keys with access to your account.