

1. Vorbereitung

Zur Durchführung der Veranstaltung und zur Anfertigung der Seminararbeit werden einige Werkzeuge vorausgesetzt:

XAMPP (V 7.3)	Lokaler Webserver (Apache) und lokale Datenbank (MariaDB) Download: https://www.apachefriends.org/de/download.html
Visual Studio Code	Plattformübergreifender Quelltext-Editor Download: https://code.visualstudio.com/download
Node.js	Serverseitige Javascript-Laufzeitumgebung. Download: https://nodejs.org/en/download/
NPM	Paketmanager für den Zugriff auf Node.js-Module
GIT	Software zur verteilten Versionsverwaltung von Dateien Download: https://git-scm.com/download/

Aufgabe 1 - Einrichtung einer lokalen Arbeitsumgebung:

1. Installieren sie die Anwendungen XAMPP und Visual Studio Code auf ihrem Rechner. Auf die Installation von sinnvollen Erweiterungen wird später eingegangen. sie können die Anwendungen testweise starten. Ein zusätzlicher Funktionstest erfolgt dann, wenn wir die Anwendungen erstmalig einsetzen.
2. Installieren sie die Anwendungen Node.js+NPM auf ihrem Rechner. Verzichten sie aktuell auf die Installation von Erweiterungen. Informationen zur Installation auf einem Windows-basierten System finden sie unter <https://docs.microsoft.com/de-de/windows/nodejs/setup-on-windows>.
3. Öffnen sie auf ihrem Rechner eine Konsole und geben sie zur Überprüfung folgende Kommandos ein:

```
node -version
```

```
npm -version
```

Die Ausführung der Kommandos zeigt die Version der jeweiligen Software und könnte wie folgt aussehen. Die bei Ihnen angezeigten Versionen können höher sein!

```
C:\Users\fampe>node --version
v12.16.0

C:\Users\fampe>npm --version
6.13.4
```

Neben geeigneten Werkzeugen helfen uns einige Internetdienste, die Zusammenarbeit innerhalb der Gruppe zu verbessern. So können wir gemeinsam genutzte Dateien zentral für alle Gruppenmitglieder erreichbar auf GitHub hosten. Ein kostenloser Account ermöglicht die Erstellung von beliebig vielen öffentlichen Repositories und wenigen privaten. Die maximale Anzahl der Mitwirkenden ist bei privaten Repositories auf drei beschränkt. Für unsere Zwecke ist ein öffentliches Repository momentan ausreichend. Beachten Sie, dass ich sie nicht verpflichten möchte einen kostenlosen Account zu erstellen – ein Account ist aber sicherlich hilfreich. Wir werden später noch optional einen Account bei stackblitz.com anlegen. Beide Accounts können verknüpft werden. Mehr dazu später.

Aufgabe 2 – Installation GIT:

1. Installieren sie Git auf Ihrem Rechner und legen sie sich einen Account bei GitHub an. Die notwendigen Links können sie der Tabelle entnehmen. Die Installation eines zusätzlichen GUI-Clients ist nicht notwendig.
2. Öffnen sie eine Konsole, ein Terminal oder die Git-Bash und geben sie folgendes Kommando ein:

```
git --version
```

Die Ausgabe könnte wie folgt aussehen:

```
C:\Users\fampe>git --version
git version 2.26.0.windows.1
```

3. Erstellen sie einen Account bei GitHub. Besuchen sie dazu die Seite <https://github.com/> und wählen sie „sign up“.
4. Wir müssen nun einige GIT-Einstellungen tätigen, die uns später erlauben, Dateien über GitHub zu verwalten. Öffnen sie ein Terminal und geben sie folgende Kommandos ein:

```
git config --global user.email "IhreEmailBeiGitHub"
git config --global user.name "IhrBenutzernameBeiGitHub"
```

Setzen sie an den gekennzeichneten Stellen ihre E-Mail und ihren Benutzernamen ein, den sie bei der Anmeldung bei GitHub angegeben haben. Beispiel:

```
C:\Users\fampe>git config --global user.email github@top69.de
C:\Users\fampe>git config --global user.name "tomperw"
```

5. Der Abgleich eines lokalen Repository mit einem entfernten auf GitHub gehosteten Repository erfolgt über SSH. Dazu ist es notwendig, einen SSH-Schlüssel zu erzeugen. Das Vorgehen wird für unterschiedliche Plattformen auf folgender Seite beschrieben:

<https://help.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

Die Punkte drei bis sieben auf der folgenden Seite beschreibt, wie sie den eben erzeugten Schlüssel in ihren GitHub-Account eintragen: <https://help.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account>. Teilen Sie das Projekt mit weiteren Teilnehmern, muss jeder Teilnehmer die Schritte durchführen, also einen Schlüssel erzeugen und diesen dem GitHub-Account bekannt machen.

Git ist ein verteiltes Versionsverwaltungssystem, das das gemeinsame Bearbeiten von Dateien erlaubt. Zentrale Verwaltungseinheiten sind die Repositories. Ein Repository ist ein Verzeichnis zur Speicherung und Beschreibung von Dateien. Jeder Benutzer besitzt eine lokale Kopie des gesamten Repository auf seinem Rechner, inklusive der Versionsgeschichte. So können die meisten Aktionen lokal und ohne Netzwerkzugriff ausgeführt werden. Es wird nicht zwischen lokalen Entwicklungszweigen und Entwicklungszweigen entfernter Repositories unterschieden. In der Regel arbeitet man auf einem lokalen Repository und gleicht dies mit einem entfernten Repository ab. Auf dieses entfernte Repository haben alle Mitwirkenden Zugriff. Häufig wird für jedes Projekt ein separates Repository angelegt. Ein typischer Ablauf könnte wie folgt aussehen:

- a) Mitarbeiter A legt ein neues, leeres lokales Git-Repository an
- b) Dem Repository werden von Mitarbeiter A Dateien hinzugefügt, die im weiteren Verlauf von Mitarbeiter A verändert werden
- c) Ist ein Stand erreicht, dann werden die Änderungen von Mitarbeiter A „committed“ und damit in das lokale Repository übernommen.
- d) Nun kann der lokale Stand des Repository des Mitarbeiters A mit einen entfernten Repository abgeglichen werden.
- e) Mitarbeiter B klonet das von Mitarbeiter A angelegt Git-Repository.
- f) In seinem Arbeitsverzeichnis befindet sich nun eine Kopie aller im Repository verwalteten Dateien, sein lokales Repository. Diese Dateien können nun von Mitarbeiter B modifiziert werden. Zusätzlich kann Mitarbeiter B dem Repository weitere Dateien hinzufügen oder bestehende Dateien aus dem Repository entfernen.
- g) Ist ein gewünschter Stand erreicht, werden die Änderungen von Mitarbeiter B „committed“ und damit in das lokale Repository des Mitarbeiters B übernommen.
- h) Nun kann der lokale Stand des Repository des Mitarbeiters B mit einen entfernten Repository abgeglichen werden.
- i) Um den neuen Arbeitstag sicher mit dem neuesten Stand zu beginnen, gleichen Mitarbeiter A und Mitarbeiter B ihren ihre lokalen Repository mit dem zentralen entfernten Repository ab.

Aufgabe 3 – Das Arbeiten mit GIT/GitHub:

Wir wollen testweise ein lokales und eine entferntes Repository auf GitHub anlegen und die Dateien abgleichen. Das Beispiel bildet das oben beschriebene Szenario ab. Sie lernen dabei die wichtigsten Kommandos in ihrer Grundform kennen. Jedes vorgestellte Kommando kann durch zahlreiche Parameter ergänzt werden. In der Regel kommen wir mit den „Grundformen“ aus. Die notwendigen Befehle sind in Teilen Windows-spezifisch, sollten sich aber einfach auf andere Plattformen übertragen lassen.

1. Öffnen sie ein Terminal, eine Git-Bash oder eine Kommandozeile und erstellen sie an beliebiger Stelle einen neuen Ordner „GitTest“. Wechseln sie in das Verzeichnis.

```
mkdir GitTest
cd GitTest
```

- Erstellen sie ein neues Repository

```
git init
```

Dieses Kommando legt im aktuellen Verzeichnis ein weiteres *verstecktes* Verzeichnis namens „.git“ mit Metainformationen zum Repository an. Im weiteren Verlauf kommen wir mit diesem versteckten Verzeichnis nicht in Berührung.

- Erstellen sie im Verzeichnis „GitTest“ eine neue Datei namens „test1.txt“ und dem Inhalt „Hallo Welt“.

```
echo "Hallo Welt" >> test.txt
```

Anmerkung:
Sie können dazu auch einen einfachen Editor verwenden.

- Nehmen sie diese Datei in ihr Repository auf.

```
git add test.txt
```

- Lassen sie sich den Status ihres Repository anzeigen.

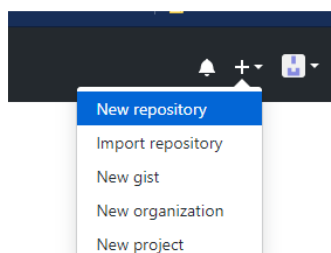
```
git status
```

- Übernehmen sie nun die Änderungen in das lokale Repository

```
git commit -m "Erstes commit"
```

Wir haben nun ein lokales Repository erstellt und wollen dies mit einem entfernten, auf GitHub gehosteten Repository abgleichen. Im ersten Schritt müssen wir dazu ein neues Repository auf GitHub erstellen.

- Melden sie sich auf GitHub an und wählen sie in der Tool-Leiste rechts oben das Pluszeichen und den Eintrag „New repository“ .




- Wählen sie ein Name für das Repository (hier: „meintest“) und optional eine Beschreibung. Bestätigen sie durch Klick auf die Schaltfläche „Create repository“.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner Repository name *

 tomperrrw / meintest ✓

Great repository names are short and memorable. Need inspiration? How about [friendly-goggles](#)?

Description (optional)

Dies ist ein Test

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.





☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None Add a license: None ⓘ

[Create repository](#)

9. Auf der folgenden Seite finden sie im oberen blauen Kasten eine Adresse auf das Repository. Kopieren sie sich die SSH-Adresse in die Zwischenablage, sie benötigen diese gleich in Punkt 10.

Quick setup — if you've done this kind of thing before

 Set up in Desktop or  HTTPS  SSH git@github.com:tomperrrw/meintest.git 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

Die Erstellung des leeren, entfernten Repository ist nun beendet, lassen sie das Fenster noch offen. Im nächsten Schritt wollen wir das lokale mit dem entfernten Repository abgleichen.

10. Im ersten Schritt müssen wir dem lokalen System das Remote-Repository bekannt machen. Dies wird durch das folgende Kommando erledigt:

```
git remote add origin git@github.com:tomperrrw/meintest.git
```

Mit „origin“ wird dem „Remote-Repository“ ein Name gegeben. Dieser ist frei wählbar, häufig aber mit „origin“ bezeichnet. Der letzte Parameter gibt die Adresse des entfernten Repository an. Dieser Vorgang muss nur einmal durchgeführt werden, die Verknüpfung wird in den Metadaten (siehe oben) hinterlegt. Alle bereits angelegten Remote-Repositories können über das Kommando `git remote -v` aufgelistet werden. Mit dem Kommando `git remote show origin` erhält man weitere Informationen zum entfernten Repository.

11. Im letzten Schritt können nun die Repositories abgeglichen werden. Haben sie ihren SSH-Schlüssel mit einem Kennwort gesichert, dann werden sie im Rahmen der Ausführung des folgenden Kommandos zur Eingabe dieses Kennwortes aufgefordert.

```
git push origin master
```

Mit „master“ wird dabei der Hauptentwicklungszweig bezeichnet. Dieser wird beim Erstellen des Repository automatisch angelegt. Das Arbeiten mit Entwicklungszweigen (branches) wird an dieser Stelle nicht betrachtet. Wir gleichen immer die Hauptentwicklungszweige der beteiligten Repositories ab, auch wenn die Praxis anders aussieht. Sichten Sie dazu die zusätzlichen Informationen.

12. Aktualisieren sie nun die GitHub-Seite im Browser. Im Reiter „Code“ sollte die eben erstellte Datei „test.txt“ aufgelistet sein. Diese können sie nun auch in der GitHub-Oberfläche einsehen oder editieren.

Die Konsole könnte nach Abarbeitung der Schritte folgendes Aussehen haben:

```
C:\>mkdir GitTest

C:\>cd GitTest

C:\GitTest>git init
Initialized empty Git repository in C:/GitTest/.git/

C:\GitTest>echo "Hallo Welt" >> test.txt

C:\GitTest>git add test.txt

C:\GitTest>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test.txt

C:\GitTest>git commit -m "Erstes Commit"
[master (root-commit) 70de405] Erstes Commit
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt

C:\GitTest>git remote add origin git@github.com:tomperrw/meintest.git

C:\GitTest>git push origin master
Enter passphrase for key '/c/Users/fampe/.ssh/id_rsa':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 221 bytes | 221.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:tomperrw/meintest.git
 * [new branch]      master -> master

C:\GitTest>
```

Im letzten Schritt wollen wir das entfernte Repository in einem neuen Verzeichnis klonen. Wir simulieren damit den Mitarbeiter B.

1. Öffnen sie ein weiteres Terminal, eine weitere Konsole.
2. Klonen sie das entfernte Repository. Im Rahmen der Ausführung des Kommandos müssen sie gegebenenfalls noch einmal das Kennwort zur Sicherung ihres SSH-Schlüssels eingeben.

```
git clone git@github.com:tomperrw/meintest.git
```

Das System erstellt nun eigenständig ein Verzeichnis mit dem Namen des entfernten Repositories (hier: „meintest“).

3. Wechseln sie in das Verzeichnis und lassen sie sich den Inhalt der Datei test.txt anzeigen.

```
cd meintest
echo test.txt
```

Zusammenfassung:

```
C:\>git clone git@github.com:tomperrw/meintest.git
Cloning into 'meintest'...
Enter passphrase for key '/c/Users/fampe/.ssh/id_rsa':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

C:\>cd meintest

C:\meintest>dir
Datenträger in Laufwerk C: ist OS
Volumeseriennummer: CE01-1F3B

Verzeichnis von C:\meintest

09.04.2020  21:23    <DIR>          .
09.04.2020  21:23    <DIR>          ..
09.04.2020  21:23                15 test.txt
               1 Datei(en),               15 Bytes
               2 Verzeichnis(se), 436.198.699.008 Bytes frei

C:\meintest>echo test.txt
test.txt
```

Mittlerweile hat Mitarbeiter A nicht mehr den aktuellen Stand, da Mitarbeiter B möglicherweise neue Dateien hinzugefügt bzw. bestehende Dateien modifiziert und bereits in das entfernte Repository übertragen hat. Mitarbeiter A kann den aktuellen Stand mit folgendem Kommando übernehmen:

```
git pull origin
```

Auf eine detaillierte Beschreibung aller GIT-Kommandos wurde an dieser Stelle verzichtet. In Internet gibt es zahlreiche Ressourcen, die Kommandos inklusive der Parameter ausführlich beschreiben. Eine Liste mit weiterführenden Informationen finden sie unten. Auch auf den begleitenden Seiten ist ein Artikel aus der c't aus dem Jahr 2015 bereitgestellt. Das Beispiel soll den Ablauf, den Workflow beschreiben. Nicht nur aufgrund der aktuell herrschenden Bewegungseinschränkungen bietet es sich an, diese Technik für das gemeinsame Arbeiten an der Seminararbeit zu nutzen. Dabei ist die Nutzung nicht auf das Teilen von Programmsourcen beschränkt. Es lassen sich so alle Arten von Dokumenten teilen.

Weiterführende und vertiefende Informationen:

- Eine fünfteiligen Videosammlung zu diesem Themenkomplex finden sie unter <https://www.youtube.com/watch?v=-U-eUHI6euM>.
- Referenzen der wichtigsten GIT-Kommandos stehen auch als Apps zur Verfügung. Für die Android-Welt kann ich folgende App vorschlagen: <https://play.google.com/store/apps/details?id=io.github.easyintent.quickref&hl=de>
- Wichtige GIT-Befehle + Informationen zum Workflow (https://pgi-jcns.fz-juelich.de/pub/doc/git_gitflow.pdf)
- GitHub bietet selbst zahlreiche Tutorials an: <https://guides.github.com/>
- Eine ausführliche Referenz aller Kommandos finden sie unter <https://git-scm.com/docs>
- Eine auf das Wesentliche zusammengefasste Übersicht: <https://rogerdudler.github.io/git-guide/>

Aufgabe 4 – Arbeiten mit GIT/GitHub mit mehreren Teilnehmern auf mehreren Rechnern:

Bilden sie Gruppen. Sinnvollerweise sollten die Gruppen bereits so gewählt werden, wie sie ihre Seminararbeit abgeben. Ein Gruppenmitglied legt ein lokales Repository namens „Test“ an. Füllen sie dieses mit einigen Testverzeichnissen / -dateien. Erstellen sie im zweiten Schritt im eigenen Account ein GitHub-Repository namens „Seminararbeit“ und laden sie die anderen Gruppenmitglieder als Mitwirkende (Collaborator) ein. Hinweis: Alle Gruppenmitglieder müssen ebenfalls über einen GitHub-Account verfügen. Wechseln sie innerhalb von GitHub in das Repository und wählen sie den Reiter „Settings“. Springen sie hier in den Menüeintrag „Manage Access“. Siehe auch <https://help.github.com/en/github/setting-up-and-managing-your-github-user-account/inviting-collaborators-to-a-personal-repository>. Alle anderen Gruppenmitglieder sollen nun das entfernte Repository klonen. Fügen sie im Anschluss weitere Dateien hinzu. „Commiten“ sie und führen sie einen Abgleich aus. Nach dem Anlegen bzw. Klonen eines lokalen Repository können sie zwischenzeitlich von anderen Mitwirkenden getätigten Änderungen mittels `git pull origin` in ihr lokales Repository aufnehmen. „origin“ ist hierbei der Name des entfernten Repositories (siehe Punkt 10 in Aufgabe 3).

Hinweis:

Nach Abschluss der Übung können Sie das Repository wieder entfernen. Aktivieren Sie dazu den Reiter „Settings“ im entsprechenden Repository und bewegen sie sich in die „Danger Zone“. Dort befindet sich eine Schaltfläche „Delete this repository“.

Danger Zone

Make this repository private Hide this repository from the public.	Make private
Transfer ownership Transfer this repository to another user or to an organization where you have the ability to create repositories.	Transfer
Archive this repository Mark this repository as archived and read-only.	Archive this repository
Delete this repository Once you delete a repository, there is no going back. Please be certain.	Delete this repository

Aufgabe 5 – Kostenlosen Webservice + Datenbank einrichten

Sichten Sie dazu das Dokument „Einrichtung bplaced-Accounts“.