# Vorbereitung:

## Mit dem Linux Server verbinden (Win 10, MacOS, Linux)

In Powershell (Windows) oder Terminal (MacOS, Linux):

 ssh *benutzername*@zam.chrishofer.net -p 765

**Hinweis:** Bunter kursiv geschriebener Text ist ein Platzhalter und muss durch den gewünschten Text ersetzt werden.

Nur beim allerersten Mal wird man gefragt ob man diesem Rechner vertrauen möchte – hier mit yes bestätigen.

Nach erfolgreicher Passworteingabe sieht man einen Prompt **benutzername@localhost:~$.**

Zum Beenden der Verbindung **exit** eingeben oder + **d** eingeben.

## Vorbereitung: Mit dem Linux Server verbinden (ältere Windows Versionen)

Die Software putty installieren ( <https://www.putty.org> ) und starten: Dort unter Host Name **zam.chrishofer.net** eintragen und unter Port **765**. Danach Open betätigen. (Man kann diese Einstellungen auch einen Namen unter **Saved Sessions** geben und mit **Save** speichern.)

Nur beim allerersten Mal wird man gefragt ob man diesem Rechner vertrauen möchte – hier mit yes bestätigen. Danach wird man nach Benutzername und Passwort gefragt.

Nach erfolgreicher Passworteingabe sieht man einen Prompt **benutzername@localhost:~$.**

Zum Beenden der Verbindung **exit** eingeben oder + **d** eingeben

Probieren Sie folgende Befehle aus und finden Sie heraus was diese bewirken. Machen Sie sich dazu Notizen. Tauschen Sie sich in Ihrer Kleingruppe aus. (Tipp: Falls Befehl nichts ausgibt können Sie bei diesen Beispielen meist mit den Befehlen **ls** oder **pwd** nachvollziehen was passiert ist.)

Die ersten beiden Beispiele sind bereits beschrieben:

echo "Wie geht es dir?"

echo "Wie geht es dir?" > wie.txt

echo "Wie geht es dir?" >> wie.txt

curl -o geschichte.txt http://zam.chrishofer.net/maerchen.txt

cp geschichte.txt geschichte2.txt

mv geschichte2.txt ../geschichte3.txt

Zusatzaufgabe: Wie unterscheiden sich **more** bzw. **less** von **cat**?

# Linux Cheat Sheet

Mit Ordnern/Verzeichnissen (directories) arbeiten

### Zuerst immer entscheiden ob wir unsere Verzeichnisangabe

**absolut** (Bezug auf root Verzeichnis / ) oder **relativ** (Bezug aktuelle Position) umsetzen möchten.

Unter Linux ist das Trennzeichen zwischen Verzeichnissen / (Schrägstrich, Forward Slash)

### Folgende besondere Verzeichnisse gibt es:

/ root Verzeichnis (oberstes Verzeichnis)

/home/chris home Verzeichnis (jede/r Benutzer/in hat eigenes home Verzeichnis)

. aktuelles Verzeichnis

.. übergeordnetes Verzeichnis

~ steht für das home Verzeichnis des aktuellen Users

Anweisung Beschreibung

Pwd Zeigt aktuelles Verzeichnis (print working directory)

Ls Zeigt Inhalt des aktuellen Verzeichnisses

ls -l Zeigt Inhalt des aktuellen Verzeichnisses mit Details

man ls

cd .. geht ins nächst höhere Verzeichnis

cd /public\_html

cd *name* Ins Verzeichnis namewechseln

cd ~ wechselt ins Home Verzeichnis

mkdir *name* Erzeugt ein Verzeichnis mit dem Namen *name*

rmdir *name* Entfernt ein bereits bestehendes leeres Verzeichnis mit dem Namen *name*

Passwd Passwort ändern

More *datei* Seitenweises Anzeigen der Textdatei *datei*

Less *datei* Anzeige einer Textdatei *datei* mit Navigation

cat *datei* Anzeigen einer Textdatei *datei (cat tagebuch.txt)*

echo *text* Text *text* auf der Standardausgabe ausgeben

echo *text* > *datei* Standardausgabe (hier konkret *text*) in *datei* speichern

echo *text* >> *datei* Standardausgabe (hier konkret *text*) wird in *datei* hinten angehängt

cp *datei* 1 *datei* 2 Datei *datei* 1nach *datei* 2 kopieren

mv *datei* 1 *datei* 2 Datei *datei* 1 verschieben/umbenennen nach *datei* 2

rm *datei* Datei *datei* löschen

rm –rf *datei* Ordner bzw. Datei *datei* und alle darin liegenden Ordner/Dateien löschen(Achtung aufpassen!)

curl–o *datei* url Lädt die Datei unter der Adresse url und speichert sie unter dem Namen *datei* (danach evtl. return Taste betätigen um Ausgabe zu entfernen)

man anweisung Zeigt Hilfe für Anweisung anweisung an

grep wort *datei* Zeige alle Zeilen in der Datei *datei* an welche wort enthalten

grep –R wort\* Durchsuche alle Unterverzeichnisse nach wort

# Nano Texteditor

Der Texteditor kann mit nano *dateiname* eine Textdatei mit dem Namen dateiname öffnen bzw. bearbeiten. Die wichtigsten Shortcuts sind:



+ X Editor beenden

+ O Datei speichern

 + U Undo (letzter Schritt rückgängig)

+ K aktuelle Zeile ausschneiden

Zu gewünschter Position navigieren

+ U an aktueller Stelle einfügen

### Ausgewählter Bereich ausschneiden oder kopieren bzw. einfügen:

 + A Auswahl des markierten Bereichs beginnen

Zum Ende des markierten Bereichs navigieren

+ K Markierten Bereich ausschneiden **ODER** + 6 Markierten Bereich kopieren

Zu gewünschter Position navigieren

+ U An aktueller Stelle einfügen

# GIT Konfigurieren

(nur einmal auf einem Rechner notwendig)

git config --global user.email "christian.hofer@campus02.at"

git config --global user.name "Christian Hofer"

git config --global init.defaultBranch main

Mit folgendem Befehl müssen Sie Passwörter nicht erneut eingeben – Achtung Passwort wird in Klartext auf Computer gespeichert!

git config --global credential.helper store

ACHTUNG: Falls 2 Faktor Authentifizierung aktiviert wurde nicht das Github Passwort eingeben. Sie müssen ein eigenes Passwort unter Settings -> Developer settings -> Personal access tokens -> Neuer Token (repo Scope auswählen) erzeugen. In diesem Fall unbedingt Passwort speichern. (Siehe auch: https://docs.github.com/en/github/authenticating-to-github/creating-a-personal-access-token )

### Grundkonzept

**working directory (Arbeitsverzeichnis):**

Alle Änderungen geschehen im Arbeitsverzeichnis (working directory).

Die Änderungen oder neue Dateien werden im Arbeitsverzeichnis gespeichert.

**stage area (Stage-Bereich):**

Git hat im Gegensatz zu anderen Versionskontrollsystemen den Stage-Bereich (Index) eingeführt. Es ist eine Zwischenebene zwischen Arbeitsverzeichnis und Repository. Um veränderte oder neue Dateien über Git zu speichern, werden sie zuerst mit git add dateiname im Stage „gespeichert“.

**local repository:** Vom Stage-Bereich werden die Dateien mit git commit ins lokale Repository verschoben. Mit diesem Befehl wird automatisch ein Texteditor geöffnet, in dem ein Kommentar zu der Datei angehängt werden kann. Die Commits ergeben zusammen die Versionsgeschichte.

**remote repository:** Erst mit dem Befehl git push wird das Repository am Server aktualisiert, die veränderten und neuen Dateien werden hochgeladen. Wichtig ist dabei, dass der Branch, über dem die Dateien hochgeladen werden, am Server vorhanden sein muss. Wenn er am Server noch nicht vorhanden ist, muss er beim ersten push angelegt werden.

### Workflow



### Wichtige Befehle in git

Initial für jedes Projekt (git Repository)

git clone URL Ein Repository wird vom Server lokal herunterkopiert.

git status Der Git-Status wird angezeigt, d. h. welcher Branch aktuell ist, welche Dateien verändert oder neu erstellt wurden, welche Dateien im Stage-Bereich sind.

git add dateiname Die veränderte bzw. neue Datei dateiname wird dem Stage-Bereich hinzugefügt (Vorbereitung für Commit).

git commit Dateien aus dem Stage-Bereich werden dem aktuellen Branch im lokalen Repository permanent hinzugefügt.

git commit –m "Commit Message" analog nur Commit Nachricht wird sofort spezifiziert.

git branch Die Namen aller lokalen Branches werden angezeigt. Der aktuelle Branch, auf dem man arbeitet, ist mit \* markiert.

git branch branch\_name Der Branch branch\_name wird lokal neu angelegt.

git checkout branch\_name Wechseln in den lokalen Branch branch\_name

git merge branch\_name Die veränderten Inhalte inbranch\_name werden in den aktuellen Branch übernommen.

git push Dateien des aktuellen Branches des lokalen Repositories werden an den Server geschickt. Der Branch muss bereits am Server vorhanden sein.

git push -u origin branch\_name Beim ersten Push des neuen lokalen Branchesbranch\_name muss der Server-Branch angegeben werden.

git pull Die aktuelle Version des Branches wird vom Server geholt und die Dateien oder Branches aktualisiert.

git fetch Die aktuellen Versionen aller Branches werden vom Server geholt, aber die Dateien oder Branches werden nicht aktualisiert.

git fetch –p wie vorher, aber alle Branches, die inzwischen nicht mehr am Server existieren, werden gelöscht.

git show Der aktuellste Commit wird nochmals mit Commit-ID, AutorIn, Datum, Uhrzeit, Commit-Nachricht, Zusammenfassung der Veränderungen angezeigt.

git log Die letzten Commits werden angezeigt. Mögliche Parameter **–graph** und **–oneline**

git diff Wenn eine Datei verändert wurde, können (vor dem Stagen) die Unterschiede mit diesem Befehl ausgegeben werden. Wenn die veränderte Datei bereits im Stage-Bereich ist, können die Unterschiede zum Repository mit dem zusätzlichen Parameter **--staged** angezeigt werden.

# HTML-Seite

### Typische HTML-Elemente:

•normaler Text, Zeilenumbrüche werden nicht berücksichtigt

text text

•Überschrift, Nummerierung gibt an, um welche Überschriftes handelt

(h1,h2,… h6)

<h1>text</h1>

•Absatz

<p>text</p>

<p class=“absatz1“>text</p> (spezifischer Absatz, für eigenen Style!)

•Liste (unorderd List, unsortiert, mit Punkten)

<ul>

 <li>text</li>

 <li>text</li>

</ul>

•Liste (ordered List, sortiert, 1., 2., 3. …)

<ol>

 <li>text</li>

 <li>text</li>

</ol>

•Bild einfügen

<img src=“<http://web.seite/bildname.jpg>“

 width=“300“

 height=“300“ (nicht angeben wenn ohne Verzerrung gewünscht)

 alt=“Bildbeschreibung“>

•Link einfügen

 <a href=“https://Linkadresse.at> LINK </a>

### HTML-Struktur

<!DOCTYPE html>

<html> (<html lang=“de“> verwendet Deutsche Sprache für kompletten Inhalt)

 <head>

 <meta charset=“UTF-8“> (bei Verwendung von äöü)

 <title>Titeltext für Lesezeichen, Suchmaschinen etc.</title>

 <style> ... Design-Regeln (Farben, Schriftart etc.) ... </style>

 ODER <link rel="stylesheet" href="styles.css">

 </head>

 <body>

 <h1>Überschrift</h1>

 <h2>kleinere Überschrift</h2>

 <p> ... Text ... </p>

<img src=“<http://web.seite/bildname.jpg>“

 width=“300“

 height=“300“ (nicht angeben wenn ohne Verzerrung gewünscht)

 alt=“Bildbeschreibung“>

<a href=“https://Linkadresse.at> LINK </a>

 <p.Absatz1> ... Text ... </p>

 <footer>

 <p>© <a href=<https://www.campus02.at></a></p>

 </footer>

 </body>

</html>

# CSS:

### Design-Regeln:

Stehen innerhalb des <style>-Elements.

Zwei Bestandteile:

•welches Element ist gemeint?

•was soll mit dem Element gemacht werden?

Beispiele:

body { background-color: #ccccff oder #8080ff;

font-family: sans-serif; }

h1 {color: #00004d;

 letter-spacing: 3px; }

p {background-color: #f080ee oder white;

 font-style: normal }

p.absatz1 { column-count: 2;

 color: black; }

li { font-weight: bold/italic; font-size: 80%; margin-top: 0.5em; }

Schrift:

font-family Schriftart Arial, TimesNewRoman, sans-serif

font-size Schriftgröße 20px

color Schriftfarbe red, green, white oder #ccccff

font-variant Schriftvariante normal, small-caps

font-weight Schriftgewicht normal, bold, bolder, lighter

font-Style Schriftstil normal, oblique, italic

Textgestaltung:

Text-align Ausrichtung left, right, center, justify (Blocksatz)

Line-height Zeilenabstand 5px

Text-decoration Ausgestaltung underline, overline, line-through, blink

Letter-spacing Zeichenabstand 3px

Text-transform Textart capitalize, uppercase, lowercase, none

Bilder:

Background Hintergrundfarbe red, green, #ccccff

Background-image Hintergrundbild none, URL

Background-repeat Kacheln repeat, repeat-x, repeat-y, no-repeat

### Darstellung von Umlauten:

& + Buchstabe (ohne Punkten) + uml;

Ä => &Auml; ä => &auml;

Ö => &Ouml; ö => &ouml;

Ü => &Uuml; ü => &uuml;

ß => &szlig;

Sonstiges:

<b> fett geschrieben </b>

<i> italic geschrieben </i>

<em> wird hervorgehoben, emphasis </em>

<sup> hochgestellte Schrift², superscript </sup>

<sub> tiefgestellte Schrift2, subscript </sub>

<del> durchgestrichener Text </del>

<small> kleingedrucktes </small>

<br> erzw. Zeilenumbruch, break

<hr> Trennlinie, horizontal ruler

<!-- Kommentar ohne Wirkung im Browser -->

border: 2px solid powderblue;
  padding: 30px;

margin: 50px;