**LB2**

**Web-Server**

|  |  |
| --- | --- |
| **Version:** | 0.2 |
|  |  |
| **Modul:** | M300 |
| **Ansprechpartner:**  **Lehrperson:** | Name: Nury Schmed E-Mail: [nury.schmed@edu.tbz.ch](mailto:nury.schmed@edu.tbz.ch)  Name: Marcello Calisto |
|  |  |
| **Status:** | In Bearbeitung |
| **Bemerkung:** |  |

**Inhaltsverzeichnis**

[Inhaltsverzeichnis 2](#_Toc43359371)

[Glossar 3](#_Toc43359372)

[1 Einleitung 4](#_Toc43359373)

[1.1 Zielsetzung und Zweck 4](#_Toc43359374)

[1.2 Überblick über das Dokument 4](#_Toc43359375)

[1.3 Git 4](#_Toc43359376)

[1.3.1 Verantwortlichkeiten 5](#_Toc43359377)

[1.3.2 Vertreterregelungen 5](#_Toc43359378)

[2 Netzwerk 6](#_Toc43359379)

[2.1 Verkabelung 6](#_Toc43359380)

# Glossar

**Inhalt:**

In diesem Abschnitt werden Fachwörter und Befehle zu dieser Dokumentation erfasst und erklärt.

Wörter welche mit einem Stern\* markiert sind hier aufgeführt.

|  |  |
| --- | --- |
| **Begriffe / befehle** | **Beschreibung** |
| Git |  |
| Git Bash |  |
| Git Hub |  |
| Repository |  |
| Masterverzeichniss |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Einleitung

In Diesem Modul lernen wir die Versionierungsverwaltung kennen und das Staging über ein Client Config-files, über welches der Client präzise konfigurierbar ist. Da die Wirtschaftliche Zukunft sich mehr nach Therminal und direktes Codieren und weniger auf die GUI richten, lernen wir alles genannte ohne GUI zu bedienen.

## Zielsetzung und Zweck

In diesem Dokument zeige ich auf wie ich ein Versionierungsverwaltete Ablage aufsetze. In dieser Ablage wird ein Config-file erstellt, welches einen Client als Web-Server aufsetzt. Auf diesem Web-server werde ich eine .HTML Datei hochladen um meine Web-site nicht kahl zu lassen. Das ganze wird aufgesetzt ohne GUI.

## Überblick über das Dokument

Die Netzwerkdokumentation wurde sehr praxisorientiert erstellt und wird ebenso praxisorientiert gepflegt, um auch fachfremden Personen bzw. Mitarbeitern einen leichten Einstieg bzw. Übersicht zu ermöglichen. Dazu ist die Dokumentation in folgende Kapitel unterteilt:

**Netzwerk**Beschreibung der relevanten Komponenten des Netzwerkes wie Verkabelungen oder Hubs

**IT-Inventar**

Tabellarische Übersicht sämtlicher Komponenten der IT-Infrastruktur

**Serverlandschaft**

Beschreibung der Server wie z.B. Exchange-Server oder FTP-Server

**Clients**

Übersicht der Clients in der IT-Infrastruktur

**Peripherie**

Umfangreiche Beschreibung der Peripheriegeräte wie Drucker oder Scanner

**Softwaresysteme**

Beschreibung der relevanten IT-Softwaresysteme aus den Bereichen XXX und XXX

**TK-Anlage**

Details zur TK-Anlage im IT-Verbund

**Sicherheit**

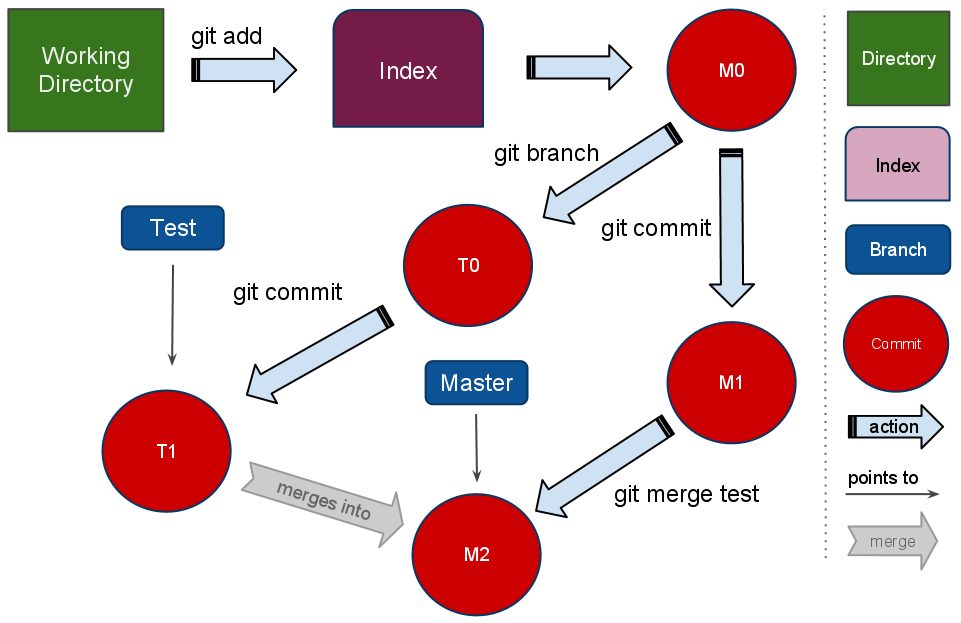
Die relevanten Sicherheitsmaßnahmen in der IT-Infrastruktur

**Benutzerdokumentation**

Einteilung der Benutzer in OUs und in Exchange Gruppen

## Git

Git ist eine Software zur Versionierungsverwaltung von Dateien/Projekten. Mit Git legt man ein Repository an welches als „synchronisierungs Ordner“ fungiert. Alle abgelegten Dateien / Verzeichnisse welche sich im Repository befinden, egal ob man Sie über Git Bash oder selber im File-Explorer anlegt, werden in die Cloud zum Git Hub Repository hochgeladen auf Befehl.



### Vagrant

Vagrant ist eine Software zum Erstellen und Verwalten [virtueller Maschinen](https://de.wikipedia.org/wiki/Virtuelle_Maschine). Vagrant ermöglicht einfache [Softwareverteilung](https://de.wikipedia.org/wiki/Softwareverteilung) insbesondere in der [Software-](https://de.wikipedia.org/wiki/Softwareentwicklung) und [Webentwicklung](https://de.wikipedia.org/wiki/Webentwicklung) und dient als [Wrapper](https://de.wikipedia.org/wiki/Wrapper_(Software)) zwischen [Virtualisierungssoftware](https://de.wikipedia.org/wiki/Virtualisierungssoftware) wie [VirtualBox](https://de.wikipedia.org/wiki/VirtualBox), [KVM](https://de.wikipedia.org/wiki/Kernel-based_Virtual_Machine)/[QEMU](https://de.wikipedia.org/wiki/QEMU), [VMware](https://de.wikipedia.org/wiki/VMware) und [Hyper-V](https://de.wikipedia.org/wiki/Hyper-V) und [Software-Configuration-Management](https://de.wikipedia.org/wiki/Software-Configuration-Management)-Anwendungen beziehungsweise Systemkonfigurationswerkzeugen wie [Chef](https://de.wikipedia.org/w/index.php?title=Chef_(Software)&action=edit&redlink=1), [Saltstack](https://de.wikipedia.org/wiki/Saltstack) und [Puppet](https://de.wikipedia.org/wiki/Puppet_(Software)) Vertreterregelungen

# Konfiguration

## Code im Bash

81 git config --global user.name ulangtaun

Hier melde ich mich im Bash an mit meinem User

82 git remote add origin https://ulangtaun:password@  
github.com/ulangtaun/M300-Services.git

Hier verlinke ich meinen User an ein existierendes Repository

(es popt ein kleines Fenster auf um sich zu verifizieren)

87 ssh -T ulangtaun@github.com

Hier teste ich meine Verbindung, ob diese Erfolgreich aufgebaut wurde

93 cd "E:/Ramonans NB/Doku Git-bash/Gitr"

Hier wechsle ich das Verzeichnis zum gewünschten Zielort für das Repository

94 git init

Mit «init» initialisiere ich das Masterverzeichniss\*

95 git clone https://github.com/ulangtaun/M300-Services.git

Hier Kopiere ich das Repository, welches in der Cloud existiert auf mein Lokales Masterverzeichniss. Danach ist mein Master Verzeichniss mein lokales Repository

96 ls

Mit «ls» kontrolliere ich nun ob ich wirklich mein Repository heruntergeladen habe

99 README.md

Ich teste die funktionierende Versionskontrolle indem ich das README.md bearbeite

100 git status

Hier wird Überprüft ob es Veränderungen gibt, welche noch nicht mit der Cloud synchronisiert sind

101 git add .

Mit dem «.» binde ich alle Veränderungen für die Initialisierung ein

103 git commit "test veraenderung"

104 git commit README.md"test veraenderung"

105\* git commit README.md "tes

106 git commit README.md -a "test veraenderung"

107 git status

108 git push

109 git push

110 git push

111 git status

112 git add .

113 git commit -m "test veraenderung"

114 git push

115 git status

116 cd vagrant

117 rm Vagrantfile

118 cd ..

119 git status

120 git add .

121 git commit -m "Vagrant delete"

122 git push

123 mkdir vagrant

124 cd vagrant/

125 vagrant init ubuntu/xenial64

126 l

127 ls

128 git status

129 cd ..

130 git status

131 git add .

132 git commit -m "Vagrant Web-Server"

133 git push

134 vagrant up

135 vagrant global-status

136 cd vagrant/

137 vagrant up

138 vagrant status

139 git status

140 git add .

141 git commit -m "Web-Server erstellt und gestarted"

142 git push

143 git status

144 git add .

145 git commit -m "unbekannte veränderung"

146 git status

147 git push

148 git status

149 cd ..

150 git status

151 cd vagrant/

152 vagrant ssh

153 cd ..

154 git status

155 git add .

156 git status

157 git commit -m "HTML ändern"

158 git push

159 git status

160 git add .

161 git commit -m "Website anpassung"

162 git push

163 git status