

<div>General</div> <div>cal: prints calender</div> <div>hostname: name of computer</div> <div>id: UID (userid), GID (group)</div> <div>ps: current processes</div> <div>uname: name of operating system</div> <div>tty: terminal device used</div> <div>who: all users logged in</div> <div>whoami: current user</div> <div>man [command]: help manuel</div> <div>less [filename]: print file page per page</div> <div>which [comm]: show program path</div> <div>bc: binary calculator</div>	<div>Filemanagment</div> <div>ls:</div> <div>-a: hidden files</div> <div>-l longlist, mit Rechte</div> <div>-d */ directories in current only</div> <div>-R list recursively</div> <div>mkdir -p --Unterverzeichnisse autom. erstellen</div> <div>rmdir dirnam --del emty dir</div> <div>rm -d --dir -r --rekursiv</div> <div>mv, cp</div> <div>mv f1 f2 dir1 -- f1 &amp; f2 in dir1 schieben</div> <div>mv -i --fragt ob überschreiben</div> <div>mv f1 dir1/fnew -- f1 in dir 1 schieben und umbenennen</div> <div>cp file1 file2 -- kopieren und umbenennen</div> <div>cp -R dir1 dir2 -- dir1 rekursiv kopieren</div> <div>cp -i file1 dir/fileA -- file1 kopieren und umbenennen -- ggf nachfragen (-i)</div>																												
<div>Wildcards</div> <div>* -- beliebige Zeichenfolge</div> <div>? -- genau 1 Zeichen</div> <div>a[abc] -- aa, ab, ac</div> <div>a[!abc] -- ad, ae... a[jedes ASCII außer abc]</div> <div>\ escape - "ab\?cd" -&gt; "ab?cd"</div> <div>mkdir home/{dir1,dir2,dir3} -- erstellt 3 dirs</div>	<div>Benutzerverwaltung</div> <div>-mit Schreibrecht auf Verzeichnis kann man darin löschen / umbenennen</div> <div>umask: Rechte enziehen: umask 022 = entzieht G und o das Writerecht</div> <div>-Gruppe sudo: alle darin dürfen sudo aufrufen</div> <div>-sudo adduser (--system) username (--ingroup groupname)</div> <div>-sudo deluser (--remove-home) username</div> <div>-sudo usermod -aG groupname username //add user to group</div> <div>-sudo usermod -g groupname username //change primary group</div> <div>-sudo deluser username groupname //rem. user from group</div> <div>-sudo addgroup groupname</div> <div>-sudo delgroup groupname</div> <div>-sudo usermod -l newname oldname //rename user - wenn user ausgelogged</div> <div>-sudo passwd username</div> <div>root: standardmäßig in Ubuntu deaktiviert - passwort setzen</div> <div>- dann: sudo passwd -l (lock) -u (unlock) root</div>																												
<div>Rechteverwaltung</div> <div>-neue Verzeichnisse / files: umask 022 //danach werden neue Verzeichnisse mit 755 und files mit 644 erstellt</div> <div>-SUID/SGID: s statt x bei user: ausführender Benutzer hat während Ausführung die UID / GID des Besitzers, Bsp.: passwd</div> <div>-SUID bei Verzeichnis macht keinen Sinn</div> <div>-SGID bei VZ erbt diese Gruppe - sinnvoll für Teamverzeichnisse</div> <div>-Sticky-Bit: bei VZ kann jeder mit Schreibrecht Daten löschen oder umbenennen. - Lösung: Sticky-Bit: verhindert das. nur root und owner können löschen und umbenennen</div> <div>-chmod -- erstes bit x000 legt fest. StickyBit: 1; SGID: 2; SUID:4</div>	<div>-chmod -R g-x,u+wx,o=r filename</div> <div>-chmod 1755 -- stickybit -- chmod +t</div> <div>-chmod 2755/3755/4755..</div> <div>Eigentümer / Gruppe ändern</div> <div>-chown username filename //nur root</div> <div>-chgrp groupname filename // root oder owner wenn in Gruppe</div> <div>-chown user:group files //beliebig setzen möglich</div> <div>-chown -R root /home //rekursiv</div>																												
<div>Git diff</div> <table><tr><th>Befehl</th><th>Änderungen von</th><th>→</th><th>auf</th></tr><tr><td>git diff</td><td>Staging Area</td><td></td><td>Arbeitskopie</td></tr><tr><td>git diff --cached</td><td>HEAD</td><td></td><td>Staging Area</td></tr><tr><td>git diff HEAD</td><td>HEAD</td><td></td><td>Arbeitskopie</td></tr><tr><td>git diff HEAD^ HEAD</td><td>Vorgänger von HEAD</td><td></td><td>HEAD</td></tr><tr><td>git diff branch1 main</td><td>branch1</td><td></td><td>main</td></tr><tr><td>git diff branch1...main</td><td>Ursprung von branch1</td><td></td><td>main</td></tr></table>	Befehl	Änderungen von	→	auf	git diff	Staging Area		Arbeitskopie	git diff --cached	HEAD		Staging Area	git diff HEAD	HEAD		Arbeitskopie	git diff HEAD^ HEAD	Vorgänger von HEAD		HEAD	git diff branch1 main	branch1		main	git diff branch1...main	Ursprung von branch1		main	
Befehl	Änderungen von	→	auf																										
git diff	Staging Area		Arbeitskopie																										
git diff --cached	HEAD		Staging Area																										
git diff HEAD	HEAD		Arbeitskopie																										
git diff HEAD^ HEAD	Vorgänger von HEAD		HEAD																										
git diff branch1 main	branch1		main																										
git diff branch1...main	Ursprung von branch1		main																										

<p><b><i>GIT Setup</i></b></p> <ul style="list-style-type: none"> <li>- git <b>init</b>: neuer Workspace</li> <li>- git <b>clone</b>: kopiert bestehendes repo --&gt; neuer workspace             <ul style="list-style-type: none"> <li>-o benennt das Ursprungsrepo --&gt; wichtig für syncs</li> <li>-b auszucheckender branch</li> </ul> </li> <li>- git <b>config</b> -l --vorhandene sets             <ul style="list-style-type: none"> <li>--unset --entfernt setting</li> <li>--system (all repo aller users) --global (alle repos des users) --local (akt. repo, default)</li> </ul> </li> <li>- <b>Alias</b> setzen: git config --global alias.newcommand "orig. command"</li> <li>- git config --global <b>user.name</b>/ user.email "user name"/"email"</li> </ul>	<p><b><i>Datenstrukturen:</i></b></p> <ul style="list-style-type: none"> <li>- <b>Commit</b> speichert: Stand von Arbeitsverzeichnis: Metadaten (wer, wann etc.), <b>parent</b> ID, tree ID (hashwerte)</li> <li>- <b>tree</b> speichert: hashes von Blobs</li> <li>- <b>blob</b>: Abbild einer ganzen Datei - wenn ein Hash einer zu committenden Datei schon existiert, wird einfach der bereits erstellte Blob verwendet, ansonsten wird ein neuer Blob erstellt</li> </ul>
<p><b><i>Zustände: untracked, unmodified, modified, staged</i></b></p> <ul style="list-style-type: none"> <li>- git <b>rm</b> file: löscht und added, also = rm file; git add file</li> <li>- git <b>mv</b> file file1 = mv ..; git add file; add file1: Git erkennt Verschiebungen durch den gleichn Hashwert</li> <li>- git <b>status</b>: Zustand der files im WD</li> <li>- git <b>diff</b> (--cached) --vergleicht WD mit Index bzw. Index mit repo (--cached)</li> <li>- <b>.gitignore</b>: *bel. string; ** bel. Unterverzeichnisse; ? bel. char; [abc]:ein Zeichen der Menge; mit !file wird td getrackt</li> <li>- git <b>commit</b>: -m -a(alles vom index) --<b>amend</b>(zum letzten commit dazu)</li> </ul>	<p><b><i>Commits referenzieren</i></b></p> <ul style="list-style-type: none"> <li>- ^ und ~: beides Vorgänger. HEAD~2 = HEAD~~ / ^^</li> <li>- HEAD^^2 = 2 commits zurück, beim 2. aber anderer branch</li> <li>- auch main^ geht</li> <li>- git <b>log</b>: Historie</li> <li>- git <b>show</b>: infos zu commit: z.B. git show HEAD^^ oder hash</li> </ul>
<p><b><i>Rückgängig machen</i></b></p> <p>git <b>diff</b>: Änderungen zwischen commits</p> <ul style="list-style-type: none"> <li>- git <b>checkout</b>: bewegt HEAD und neues WD</li> <li>- git <b>reset</b> --soft (ändert nur head) --mixed (default, ändert staging) --hard (ändert WD)</li> <li>- git <b>restore</b> -S (--staged; repo to staging) -W (--worktree; staging to WD)</li> <li>- git <b>revert</b> (erzeugt neuen commit, inhalt gleich altem Commit)</li> <li>- bei geteilten repos immer revert</li> <li>- git <b>reflog</b>: alte Aktionen</li> </ul>	<p><b><i>Files Zusammenführen</i></b></p> <ul style="list-style-type: none"> <li>- git <b>branch</b> //neuer branch</li> <li>- git <b>checkout</b> // Head -- branch/commit --b legt branch an</li> <li>- git <b>switch</b> -c (legt branch an) -d (detach head auf beliebigen comm)</li> <li>- git <b>merge</b> feature (Quellbranch) (Zielbranch ist der der ausgewählt)</li> <li>- git <b>rebase</b> (neue Basis) -i (interactive)</li> <li>- git <b>cherry-pick</b> (-n no commit)</li> <li>- git <b>stash</b>: Zwischenspeicher in einem Stack</li> </ul>
<p><b><i>Remote repositories</i></b></p> <ul style="list-style-type: none"> <li>- git <b>clone</b>: lokale Kopie von entferntem repo             <ul style="list-style-type: none"> <li>- Name ist standardmäßig origin, kann durch -o &lt;name&gt; geändert werden</li> <li>- beim clone werden alle branches runtergeladen und als <b>remote tracking branch</b> gespeichert, automatisch wird der branch ausgecheckt auf den der head im remote repo zeigt</li> <li>- Beim auschecken eines solchen wird ein <b>local tracking branch</b> erstellt, auf dem man arbeiten kann</li> </ul> </li> <li>- einfach git switch -c &lt;branchname&gt;</li> <li>- git switch -c &lt;newname&gt; origin/&lt;remotename&gt;</li> </ul>	<ul style="list-style-type: none"> <li>- git <b>remote</b> -v (list entfernte repos) <b>show</b> (details zu ent. repo) <b>add</b> (ent. repo hinzufügen) <b>remove</b> (ent. repo. entf.)</li> <li>- git <b>fetch</b>: aktualisiert rem. track. branches, noch nicht gemerged</li> <li>- git <b>pull</b>: fetch und merge</li> <li>- git <b>push</b>: -u (set upstream (macht zu local tracking branch)) -d (del, löscht branch im rem. repo) -f (<b>force push</b>)</li> </ul> <p>Leere Verzeichnisse können nicht versioniert werden! Deshalb brauchen wir darin eine versteckte Datei z.B. gitkeep, um sie zu versionieren.</p>
<p><b><i>Docker</i></b></p> <ul style="list-style-type: none"> <li>- docker <b>ps</b>: processes running?</li> <li>- docker <b>pull</b> {image}</li> <li>- docker <b>images</b>: local avail. images</li> <li>- docker <b>run -it -rm</b> (remove when ended) {image} (creates new container)</li> <li>- docker <b>container ps -a</b> (list all containers)</li> <li>- docker container <b>inspect</b> {container} (config file)</li> <li>- docker <b>start / stop</b> {container}</li> <li>- docker <b>attach</b> {container} (attaches terminal to PID1 process)</li> <li>- docker <b>exec</b> -it {container} bash (starts proc bash, not directly PID1)</li> <li>- exit only termin. the bash -- more safe</li> </ul>	<p><b><i>persisting data</i></b></p> <ul style="list-style-type: none"> <li>- docker <b>volume</b> ls (list volumes) - used to save outside of container layer</li> <li>- docker run -it --<b>mount</b> source=wse-volume, destination=/wse-data/ ubuntu</li> <li>- docker run -it -v wse-volume:/wse-data(:ro (read only)) ubuntu</li> </ul> <p><b>Bind mounting:</b></p> <ul style="list-style-type: none"> <li>- docker run -it --rm -v \${PWD}/data:/wse-data ubuntu</li> <li>- mounts a directory from the host system</li> </ul>

