# Cheat Sheets

Contents

[Cheat Sheets 1](#_Toc180502704)

[Linux Operating System 2](#_Toc180502705)

[Basic System Commands 2](#_Toc180502706)

[Differences between <sudo su>, <sudo -i>, <sudo /bin/bash> 2](#_Toc180502707)

[Work with File Systems 2](#_Toc180502708)

[Catalog /proc 3](#_Toc180502709)

[Users and Groups 3](#_Toc180502710)

[Packages 3](#_Toc180502711)

[Network 3](#_Toc180502712)

[Basic Network Commands 3](#_Toc180502713)

[??? 5](#_Toc180502714)

[Kernel 5](#_Toc180502715)

[Loaders (GRUB/GRUB2) 6](#_Toc180502716)

[Initialization and Processes 6](#_Toc180502717)

[MUST HAVE Commands 6](#_Toc180502718)

[Monitoring and Debugging Commands 6](#_Toc180502719)

[Open files in Linux (management) 6](#_Toc180502720)

[??? 8](#_Toc180502721)

[Configuration Files 8](#_Toc180502722)

[System Modules 8](#_Toc180502723)

[systemd 8](#_Toc180502724)

[Journaling 9](#_Toc180502725)

[Server Security 9](#_Toc180502726)

[Optimization and Automation 9](#_Toc180502727)

[Virtualization 9](#_Toc180502728)

[Utilities 9](#_Toc180502729)

[Telnet 9](#_Toc180502730)

[Archiving (zip/gzip/unzip/rar) 10](#_Toc180502731)

[IP (utility) 11](#_Toc180502732)

[???? 12](#_Toc180502733)

[Bash 12](#_Toc180502734)

[Virtualization 12](#_Toc180502735)

[KVM (console commands) 12](#_Toc180502736)

[Containerization 13](#_Toc180502737)

[Docker / Docker Compose 13](#_Toc180502738)

[Git 13](#_Toc180502739)

[Java 18](#_Toc180502740)

[JavaScript 18](#_Toc180502741)

[Node.js 18](#_Toc180502742)

[Python 18](#_Toc180502743)

[Virtual Environment 18](#_Toc180502744)

## 

## Linux Operating System

### Basic System Commands

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### Differences between <sudo su>, <sudo -i>, <sudo /bin/bash>

|  |  |
| --- | --- |
| su … |  |
| sudo … |  |
| sudo su |  |
| sudo -i |  |
| sudo /bin/bash |  |

Команда su позволяет изменить текущего пользователя терминала на другого. Команда sudo выполняет команду от имени root пользователя. Теперь разберем сам вопрос.

sudo su - выполняет команду su от имени суперпользователя. Таким образом, сначала используется sudo чтобы получить права суперпользователя, а затем пользователь меняется на root с помощью su. При этом вы останетесь в той же директории потому, что выполняется ваш .bashrc. А также .profile пользователя root поэтому вы окажетесь в окружении root.

sudo -i - похожая команда, опция -i указывает утилите что нужно переключиться в консоль от имени суперпользователя. На этот раз вы уже окажетесь в домашней папке root, и будет выполнен его .bashrc и .profile.

sudo /bin/bash - запускает еще одну оболочку bash от имени суперпользователя. Файлы конфигурации не читаются, но выполняется только .bashrc вашего пользователя. Вы не окажетесь в окружении root, а просто останетесь в своем окружении с правами суперпользователя.

### Work with File Systems

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

#### Catalog /proc

|  |  |
| --- | --- |
| /proc/cmdline | OS boot parameters |
| /proc/consoles | Current console info |
| /proc/devices | Devices drivers, working at the moment |
| /proc/dma | Current DMA channels |
| /proc/fb | Framebuffer devices |
| /proc/filesystems | Supported file systems by the current kernel version |
| /proc/iomem | Current map of the system memory for devices |
| /proc/ioports | Registered ports ranges and devices, using them |
| /proc/loadavg | System load in the time aspect + number of running processes + PID of the last process |
| /proc/locks | Files locked by kernel |
| /proc/meminfo | System memory info |
| /proc/misc | Applied devices drivers |
| /proc/modules | Loaded kernel modules (in the current moment) |
| /proc/mounts | Mount points, used by system |
| /proc/partitions | Available partitions for the system |
| /proc/pci | PCI-devices info |
| /proc/stat | Records and stat info saved since the latest restart |
| /proc/swaps | SWAP info |
| /proc/uptime | System working time in seconds |
| /proc/version | Kernel version, GCC version + installed Linux info |
|  |  |

### Users and Groups

TBD

### Packages

TBD

### Network

#### Basic Network Commands

|  |  |
| --- | --- |
| Шпаргалка по работе с сетью в linux  Краткая шпаргалка по работе с сетью в Linux, которая охватывает основные команды для диагностики и управления сетевыми интерфейсами.  1. Информация о сетевых интерфейсах  - ip a или ip addr — показывает информацию о всех сетевых интерфейсах.  - ifconfig — старый способ просмотра интерфейсов. В современных системах заменён на ip a.  - ip link show — информация о сетевых интерфейсах (без IP адресов).  2. Настройка сетевого интерфейса  - ip link set dev eth0 up/down — включить или отключить сетевой интерфейс (заменить eth0 на нужный интерфейс).  - ip addr add 192.168.1.100/24 dev eth0 — назначить IP-адрес интерфейсу eth0.  - ip addr del 192.168.1.100/24 dev eth0 — удалить IP-адрес с интерфейса.  3. Маршрутизация  - ip route show — показать таблицу маршрутизации.  - ip route add default via 192.168.1.1 — установить шлюз по умолчанию (заменить на нужный IP).  - ip route del default — удалить маршрут по умолчанию.  4. DNS  - cat /etc/resolv.conf — показать используемые DNS-сервера.  - nslookup example.com — получить IP-адрес по доменному имени.  - dig example.com — расширенная диагностика DNS-запросов.  5. Проверка сетевых подключений  - ping 192.168.1.1 — отправить ICMP-запрос к узлу (заменить на нужный IP или домен).  - traceroute example.com — показать маршрут до узла.  - mtr example.com — комбинированная команда ping и traceroute с обновлением в реальном времени.  6. Прослушивание и управление портами  - ss -tuln — показать открытые порты и активные соединения.  - netstat -tuln — устаревшая, но часто используемая команда для тех же целей.  - lsof -i :80 — показать процессы, использующие порт 80.  7. Проверка доступных сетей и соединений  - iwconfig — информация о беспроводных интерфейсах (для Wi-Fi).  - nmcli — управление соединениями через NetworkManager.  - nmtui — текстовый интерфейс для управления сетью через NetworkManager.  8. Тестирование и диагностика  - tcpdump -i eth0 — захват пакетов с интерфейса eth0.  - curl ifconfig.me — получить внешний IP-адрес.  - wget http://example.com/file.zip — скачать файл по HTTP/HTTPS.  9. Firewall (iptables)  - iptables -L — показать правила файрвола.  - iptables -A INPUT -p tcp --dport 22 -j ACCEPT — разрешить доступ по SSH (порт 22).  - iptables -D INPUT -p tcp --dport 22 -j ACCEPT — удалить правило.  10. Проброс портов  - iptables -t nat -A PREROUTING -p tcp --dport 8080 -j REDIRECT --to-port 80 — перенаправить запросы на порт 8080 к порту 80. |  |
|  |  |
| Работа с сетью  Отладка Linux  # ethtool eth0 - Показать Ethernet статус  # ethtool -s eth0 speed 100 duplex full - Принудительная установка режима 100Mbit Full duplex  # ethtool -s eth0 autoneg off - Отключить автоопределение  # ethtool -p eth1 - Мигать индикатором сетевой карты - если поддерживается  # ip link show - Список сетевых интерфейсов в Linux (подобна ifconfig)  # ip link set eth0 up - Активировать сетевой интерфейс (или отключить). Аналог "ifconfig eth0 up"  # ip addr show - Список всех IP адресов в Linux ( аналог ifconfig)  # ip neigh show - Тоже что и arp -a  Печать таблицы маршрутизации  # route -n # Linux или используйте "ip route"  # netstat -rn # Linux, BSD и UNIX  # route print # Windows  Используйте "add -p" что-бы сделать маршрут постоянным.Настройка дополнительных IP адресов Linux  # ifconfig eth0 192.168.50.254 netmask 255.255.255.0 # Первый IP адрес  # ifconfig eth0:0 192.168.51.254 netmask 255.255.255.0 # Второй IP адрес  # ip addr add 192.168.50.254/24 dev eth0 # Эквивалентные команды для ip  # ip addr add 192.168.51.254/24 dev eth0 label eth0:1 |  |
|  |  |

#### ???

TBD

### Kernel

TBD

### Loaders (GRUB/GRUB2)

TBD

### Initialization and Processes

TBD

### MUST HAVE Commands

#### Monitoring and Debugging Commands

|  |  |
| --- | --- |
| top | Show running processes and used resources with autoupdate |
| ps -eafw | Show running processes and used resources + other info – show once |
| ps -e -o pid,args --forest | Show PIDs and processes as a tree |
| pstree | Show processes tree |
| kill -9 98989  kill -KILL 98989 | Kill the process with PID = 98989 (no data consistency) |
| kill -TERM 98989 | Kill the process with PID = 98989 (gracefully) |
| kill -1 98989  kill -HUP 98989 | Force the process with PID = 98989 to reload it’s config |
| kill -s SIGSTOP 98989 | Stop running process with PID = 98989 |
| kill -s SIGCONT 98989 | Continue stopped process with PID = 98989 |
| lsof -p 98989 | Show files opened by the process with PID = 98989 |
| lsof /home/user1 | Show files opened from the directory /home/user1 |
| strace -c ls > /dev/null | Show list of the system calls created and received by the process ls |
| strace -f -e open ls > /dev/null | Show calls to the libraries |
| watch -n1 ‘cat /proc/interrupts’ | Show interrupts in the real time |
| last reboot | Show the system’s reboot history |
| last user1 | Show history of registration user1 in the system and time spent in the system |
| lsmod | Show loaded kernel modules |
| free -m | Show the RAM state in megabytes |
| smartctl -A /dev/had | Control the hard drive /dev/had state via SMART |
| smartctl -i /dev/had | Check whether SMART is available on the hard drive /dev/had |
| tail /var/log/dmesg | Show the last 10 records from the kernel load journal |
| tail /var/log/messages | Show the last 10 records from the system journal |
|  |  |

#### Open files in Linux (management)

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
| Как найти количество открытых файлов в Linux  Значение сохраняется в  # cat /proc/sys/fs/file-max  Проверка Hard лимит в Linux  # ulimit -Hn  Проверка Soft лимит в Linux  # ulimit -Sn  Чтобы просмотреть значения Hard и Soft для разных пользователей, переключите пользователя с «su» на пользователя, ограничения которого вы хотите проверить.  Например:  # su root  $ ulimit -Sn  $ ulimit -Hn  Как проверить количество системных дескрипторов файлов в Linux  На серверах, некоторые из ваших приложений могут потребовать более высокий лимит для открытых дескрипторов файлов. Хорошим примером этого являются службы MySQL / MariaDB или веб-сервер Apache.  Вы можете увеличить лимит открытых файлов в Linux, отредактировав опцию ядра fs.file-max. Для этого используйте утилиту sysctl.  Sysctl используется для настройки параметров ядра.  Например, чтобы увеличить ограничение на количество открых файла до 500000, вы можете использовать следующую команду:  # sysctl -w fs.file-max=500000  Вы можете проверить текущее значение для количества открытых файлов с помощью следующей команды:  $ cat /proc/sys/fs/file-max  С помощью указанной команды изменения, которые вы внесли, будут оставаться активными только до следующей перезагрузки. Если вы хотите применить их на постоянной основе, вам придется отредактировать следующий файл:  # vi /etc/sysctl.conf  Добавьте следующую команду:  fs.file-max=500000  Конечно, вы можете изменить число в соответствии с вашими потребностями. Чтобы снова проверить изменения, используйте:  # cat /proc/sys/fs/file-max  Чтобы изменения вступили в силу, пользователям необходимо будет выйти из системы и снова войти в систему. Если вы хотите немедленно применить ограничения, вы можете использовать следующую команду:  # sysctl -p  Установка количества открытых файлов для пользователя в Linux  Вышеприведенные примеры показали, как установить ограничения, но вы можете установить лимит для каждого пользователя. Для этой цели, с правами пользователя root, вам необходимо отредактировать следующий файл:  # vi /etc/security/limits.conf  Если вы администратор Linux, предлагаем Вам ознакомиться с этим файлом. Прочитайте все комментарии в нем, поскольку он обеспечивает гибкость в плане управления системными ресурсами путем установки лимита пользователям на разных уровнях.  Команды, которые вы должны добавить, имеют следующие параметры:  <domain>        <type>  <item>  <value>  Вот пример установки Soft и Hard лимитов для пользователя dimon:  ## Example hard limit for max opened files  dimon hard nofile 4096  ## Example soft limit for max opened files  dimon soft nofile 1024 |  |
|  |  |
|  |  |

#### ???

TBD

### Configuration Files

TBD

### System Modules

TBD

#### systemd

|  |  |
| --- | --- |
| Basic systemctl commands | |
| **systemctl status <service\_name>** | Check the service status |
| **systemctl start|stop <service\_name>** | Start/stop the specified service |
| **systemctl restart|reload <service\_name>** | Restart/reload (only if the service is running) the specified service |
| **systemctl enable <service\_name>** | Turn on the “autostart” for the specified service on boot |
| **systemctl disable <service\_name>** | Turn off the “autostart” for the specified service on boot |
| **systemctl list-units --type=service** | Show **running** services |
| **systemctl list-units --all --type=service** | Show all **running** and **stopped** services |
| **systemctl list-units --state=active** | Show all active units (services, mount, timer, etc.) |
| **systemctl mask|unmask <service\_name>** | Mask/unmask the service specified (mask – disable run) |
| Manage system state | |
| **systemctl reboot** | Reboot the system |
| **systemctl poweroff** | Shutdown the system |
| **systemctl suspend** | Put the system into sleep mode (suspend) |
| **systemctl hibernate** | Put the system into hibernate mode |
| **systemctl hybrid-sleep** | Put the system into the hybrid sleep mode |
| Work with units-files | |
| **systemctl show -p FragmentPath <service\_name>** | Show path to the unit-file of the service specified |
| **systemctl edit [--full] <service\_name>** | Edit service’s unit-file. File will be opened in the editor. Flags: --full – if you use this flag, you will edit the full version of the unit-file for service |
| **systemctl daemon-reload** | Re-activate all units after changes (reload unit-files) |
| **systemctl cat <service\_name>** | Show the service’s configuration |
| **systemctl list-dependencies <service\_name>** | Show all dependencies for the service specified |
|  | |
| **systemctl** |  |
| **systemctl** |  |
| **systemctl** |  |
| **systemctl** |  |
|  | |
| **systemctl** |  |
| **systemctl** |  |
| **systemctl** |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### 

### Journaling

TBD

### Server Security

TBD

### Optimization and Automation

TBD

### Virtualization

TBD

## Utilities

### Telnet

|  |  |
| --- | --- |
| telnet <hostname> <port> | Basic usage |
| telnet example.com 80 | Basic usage - example |
| telnet <dns-server> 53 | Check DNS server |
| Ctrl + ]  *quit*  *help*  *open <hostname> <port>*  *close*  *display*  *mode*  *send*  *status*  *set / unset* | Close session (move to the command mode), below see some of the commands:  - type <quit> after move to the command mode – close connection  - show possible commands  - open new connection to the host specified  - close connection  - current connection parameters  - switch current mode, for ex. *line* or *`character`*  - send control symbols, for ex. *send escape*  - current connection status  - sets / unsets the current connection parameters, for ex. *set echo* / *unset echo* |
| telnet <HTTP server> <port>  *GET / HTTP/1.1*  *Host: example.com* | Sending text request after connection to the HTTP server |
|  |  |

### Archiving (zip/gzip/unzip/rar)

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Полезные команды Linux. Архивация и сжатие файлов  # gzip -9 file1  Поместить файл file1 в архив gzip с максимальной степенью сжатия  # rar a file1.rar file1 file2 dir1  Создать rar архив file1.rar, заархивировав файлы: file1, file2 и директорию: dir1  # rar a file1.rar test\_file  Упаковать в rar архив file1.rar файл test\_file  # rar x file.rar  Разархивировать rar архив file.rar  # bzip2 file1  Сжимает файл file1  # bunzip2 file1.bz2  Разжимает файл file1.bz2  # gzip file1  Сжимает файл file1  # gunzip file1.gz  Разжимает файл file1.gz  # tar -cvf archive.tar file1 file2 dir1  Создать tar архив archive.tar, упаковав в него файлы file1, file2 и директорию dir1  # tar -cvf archive.tar file  Упаковать в tar-архив archive.tar, файл file  # tar -tf archive.tar  Вывести содержимое tar архива  # tar -xvf archive.tar  Распаковать tar архив  # tar -xvf archive.tar -C /tmp  Распаковать архив в /tmp  # tar -cvfz archive.tar.gz dir1  Создать tar архив и сжать его с помощью программы gzip  # tar -xvfz archive.tar.gz  Разжать tar архив и распаковать его  # tar -cvfj archive.tar.bz2 dir1  Создать архив и сжать его с помощью bzip2 ( ключ -j работает не во всех \*nix системах )  # tar -xvfj archive.tar.bz2  Разжать архив и распаковать его ( ключ -j работает не во всех \*nix системах )  # zip file1.zip file1  Создать сжатый zip-архив  # zip -r file1.zip file1 file2 dir1  Запаковать в архив несколько файлов и/или директорий  # unzip file1.zip  Разжать и распаковать zip-архив  # unrar x file1.rar  Распаковать rar-архив |  |
|  |  |
|  |  |

### IP (utility)

|  |  |
| --- | --- |
| Команда ifconfig считается устаревшей и постепенно вытесняется командой ip  ip addr show - отобразить список сетевых интерфейсов с адресами. Возможно сокращение параметров команды - ip a sh или ip a s  ip addr list permanent - отобразить только статические ip адреса  ip addr list dynamic - отобразить только динамические ip адреса  ip addr add 1.1.1.13/24 dev eth0 - установить ip адрес для интерфейса eth0  ip addr del 1.1.1.13/24 dev eth0 - удалить ip адрес интерфейса eth0  ip addr flush dev eth0 - удалить все ip адреса интерфейса eth0  ip addr help - отобразить справку по команде ip addr  ip route show - отобразить сведения о маршрутах. В сокращенном варианте - ip r s  ip route show table 255 - отобразить все маршруты из таблицы 255 ip route get 10.10.20.0/24 - отобразить маршрут к этой сети ip route get 10.10.20.0/24 from 192.168.1.100 - отобразить маршрут к этой сети от указного интерфейса с IP=192.168.1.100.  ip route add 10.10.20.0/24 via 192.168.5.1 - создать маршрут к заданной сети через интерфейс c IP=192.168.5.1  ip route delete 10.10.20.0/24 - удалить указанный маршрут.  ip route del 10.10.20.0/24 via 192.168.50.100 - удалить указанный маршрут.  ip route add default via 192.168.50.100 - создать маршрут по умолчанию.  ip route add 10.10.20.0/24 dev eth0 - создать маршрут к указной сети .  ip route add table nnn 10.10.20.0/24 dev eth0 - создать маршрут в специальной таблице маршрутов с номером nnn.  ip route add to blackhole 10.10.0.0/16- создать фиктивный маршрут.  ip route add unreachable 10.10.0.0/16 - создать недостижимый маршрут.  ip link show - отобразить список сетевых интерфейсов.  ip link show eth0 - отобразить состояние eth0  ip link list up - отобразить статус всех включенных интерфейсов  ip link set eth1 up - включить интерфейс eth1  ip link set eth1 down - выключить интерфейс eth1  nslookup - запуск nslookup в интерактивном режиме.  nslookup mail.ru - отобразить сведения для домена mail.ru с использованием сервера DNS из сетевых настроек.  nslookup mail.ru 8.8.8.8 - отобразить сведения для домена mail.ru с использованием сервера DNS с адресом 8.8.8.8 (публичный DNS-сервер Google).  nslookup –query=mx mail.ru - отобразить запись типа MX для домена mail.ru. |  |
|  |  |
|  |  |

## ????

TBD

## Bash

TBD

## Virtualization

### KVM (console commands)

|  |  |
| --- | --- |
| virsh list [--all] | Show list of running VMs, with the flag --all – show all VMs (even powered off) |
| virsh shutdown <vm name> | Shutdown the VM by name <vm name> |
| virsh start <vm name> | Start the VM by name <vm name> |
| virsh suspend <vm name> | Suspend the VM by name <vm name> |
| virsh resume <vm name> | Resume the VM by name <vm name> |
| virsh reboot <vm name> | Reboot the VM by name <vm name> |
| virsh destroy <vm name> | Destroy the VM by name <vm name> |
| virsh undefined <vm name> | Remove VM from the list and remove all files related, usually called after **virsh destroy <vm>** |
| virsh vcpuinfo <vm name> | Show info about VM CPU for VM with the name <vm name> |
| virsh domid <vm name> | Get the ID (identity) of the VM >vm name> |
|  |  |
|  |  |
|  |  |
| # virsh domuuid <vm name> — получить UUID виртуальной машины  # virsh dominfo <vm name> — получить сведения о виртуальной машине  # virsh domstate <vm name> — просмотр состояния виртуальной машины  # virsh dumpxml <vm name> — вывести файл конфигурации указанной виртуальной машины в XML формате |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Containerization

TBD

### Docker / Docker Compose

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Git

|  |  |
| --- | --- |
| Git :: Configuration + Help | |
| **git help <command>**  **git <command> --help** | Gets help on specified git command (commands are equivalent). On Windows this will open web-browser with help screen. |
| **git config --list** | Lists all config parameters (name=value list) |
| **git config --global user.name** | Sets up user name for current user (use key **--system** for set up value for all users) |
| **git config --global user.email** | Sets up user email for current user (use key **--system** for set up value for all users) |
| **git config --global core.editor <editor>** | Select text editor for git messages (commits, tags, merges etc) |
| **git config --global credential.helper cache** | Sets up credentials cache. It saves credential info for some time in memory (you don't have to enter credentials on every remote operation). Default timeout is 15 minutes. |
| **git config --global credential.helper “cache --timeout=3600”** | The same as above, but with different timeout. Timeout specified in seconds. |
| **git config --global credential.helper wincred** | The same as above, but works **for Windows** (credentials store, for msysgit 1.8.1 and above) |
| **git config --global alias.<name> <command>** | Create short alias <name> for command <command> (next in your work you can use <name> instead of <command>). Very useful for convenience work process. If <command> contains spaces (composite command) put it in quotes (double quotes on windows). |
| **<Ignoring file(s)>** | 1. Create file .gitignore in working folder and adds there files that should be ignored (not tracked by Git system) 2. Edit $GIT\_DIR/info/exclude and add ignored files there |
| Git :: Repository creation + init | |
| **git init** | Initialize repository in current folder (creates new local repository) |
| **git clone <URL> [dest-folder]** | Creates a copy (clone) existing repository located at <URL> in folder with default name (repo name). If you specify **[dest-folder]**, repository will be cloned to that folder. For clone you may use different protocols: **local**, **https://**, **git://**, **ssh**. |
| **Git :: Files operations** | |
| **git add <path/file>** | Adds new file/path to Git repository or indexes modified file. You can specify more than one file, separate them by space. Also you may use the following wildcards/patterns:   * \* - any count of any symbols * [abc] – one of abc (a or b or c) * [0-9] – values range (in this example from 0 to 9) * ? - any one symbol * \*\* - nested folders, for example a/\*\*/z means file z nested in folder a and any number of other folders between them |
| **git rm <file>** | Removes **<file>** both from Git tracked files list (indexed area) and from work folder. You can manually delete file from work folder and index file removing operation by executing git command [git rm]. You may use patterns/wildcards with this command. Important options:   * **-f** use this option when you have already edited and indexed a file (force removing) * **--cached** remove file from indexed area, but leave it in working folder (doesn't physically delete file, file becames untracked) |
| **git mv <source> <dest>** | Moves/renames file <source> to <dest>. This command is equivalent to sequence: [**mv <source> <dest>** → **git rm <source>** → **git add <dest>**] (you may manually move a file and then index changes by two git commands) |
| **git checkout -- <file>** | Discard changes in working directory – **you lose your local changes!** |
| **git checkout [revision] <file>** | Restores **<file>** from current branch HEAD (if no [revision] specified) or specified revision **[revision]**. Command overwrites not commited (not indexed) local changes (**you lose local changes!**). |
| **Git :: Working tree** | |
| **git status**   |  |  | | --- | --- | | **M** | modified | | **C** | copied and modified | | **R** | renamed and modified | | **A** | added | | **D** | deleted | | **U** | conflicts after merge | | **??** | untracked | | Shows current repository status. Important options are:   * **-v**, **--verbose** be verbose * **-s**, **--short** show status concisely   When you use -s/--short option git uses flags for files' statuses – see table on the left.  Command **git status** outputs this flags in two columns: [**MM**] or [**<space>M**] etc (using other flags). Flag in a first column means: changes are indexed (ready for commit and **will be included** in commit); flag in a second column means: changes are not indexed (not ready for commit and **won't be included** in commit). |
| **git diff [--stat] [path/file]**  *(show not indexed)* | Show **all** or specified **[path/file]** differences between HEAD and work folder – shows **not indexed** (and **not committed**) changes. Option **--stat** used for short output (summary). |
| **git diff --cached | --staged [--stat] [path/file]**  *(show indexed, not committed)* | Show **all** or specified **[path/file]** differences between HEAD and indexed area – shows **indexed** but **not committed** changes (changes that are ready for commit). Option **--stat** used for short output (summary). |
| **git diff HEAD [--stat] [path/file]**  *(show all)* | Show **all** or specified **[path/file]** differences between **not indexed** and **committed** changes for HEAD (show all changes from previous commit). Option **--stat** used for short output (summary). |
| **git diff <source-rev> <dest-rev>**  **[--stat] [path/file]** | Show **all** or specified **[path/file]** differences between **committed** changes in direction from **<source-rev>** to **<dest-rev>**. If you omit **<dest-rev>** git will use HEAD. Option **--stat** used for short output (summary). As **<source/dev-rev>** you may use branch name. |
| **git add <path/file>** | Stage path or file for the commit (indexing). |
| **git reset HEAD <path/file>** | Cancel indexing for path/file (unstage path/file, remove it from indexed area). |
| **git reset --soft HEAD**  **git reset --hard HEAD** | ????? |
| **git commit** | Commits to local repository all indexed (with [git add] command) changes. Important options are:   * **-m** specify commit message (in quotes). If you commit without this option git will open text editor for entering commit message * **-a** auto stage (indexing) **all** tracked and modified files before commit (don't need to explicitly call [git add] for indexing tracked files) * **--amend** replaces last commit for a new one (you may add forgotten file etc.) * **--dry-run** do not create a commit, but show a list of paths that are to be committed, paths with local changes that will be left uncommitted and paths that are untracked |
| **git clean** | ????? |
| **git revert <rev>** | ????? |

|  |  |
| --- | --- |
| **Git :: History (log) + browsing** | |
| **git log [path/file]**  *values for <format> option in* ***--pretty=format:''<format>''***   |  |  | | --- | --- | | %H | full hash for commit | | %h | short hash code for commit | | %T | tree hash code | | %t | tree short hash | | %P | parent commits hash | | %p | parent commits short hash | | %an | author name | | %ae | author's email | | %ad | original create date (consider **-date=** option) | | %ar | original create date, relative | | %cn | version creator's name | | %ce | version creator's email | | %cd | version creation date | | %cr | version creation date, relative | | %s | commit comment | | Show commit log (full log or for specified [path/file]) in a descending order (the last commit first). Some important/useful options are:   * **-p** show differences for every commit * **-(n)**  show the specified number → (n) of last commits * **--since**, **--after** show only commits after specified date * **--until**, **--before** show only commits before specified date * **--author** show commits for specified author * **--committer** show commits for specified committer * **--grep** show commits with commit message containing specified string * **-S<string>**  show commits where specified <string> was deleted or inserted * **--no-merges** show commits without merges * **--all-match** turn multiple parameters (see above) in state AND, default state is OR * **--stat** show short statistics for every commit * **--shortstat** show line with only changes/insertions/deletions from **--stat** option * **--name-only** show changed files names list after commit info * **--name-status** show changed files list with info about operation add/change/delete * **--abbrev-commit** show only some first (about 6) symbols for SHA-1 sum, instead all 40 * **--relative-date** show date in relative format(for example: “2 weeks ago”) * **--graph** add ASCII graph to history info * **--oneline** show log in one line for every commit * **--decorate** show branches and HEAD pointers * **--all** show all commits history * **--pretty=<value>** change output format (**oneline**, **short**, **full**, **fuller**). * **--pretty=format:''<format>''** set specified output format. Format parameters see on the left side (blue table). |
| **git log --pretty=format:”%h - %an, %ar : %s”** – pretty formatted log output | |
| **git log --pretty=format:”%h %s” --graph** – pretty formatted output with ASCII graph | |
| **git log --oneline --decorate**  – show one line log with branches pointers | |
| **git log --oneline --decorate --graph --all** – show commits history with branches pointers and graph | |
| **[alias]**  **lg1 = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' –all**  **lg2 = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(bold yellow)%d%C(reset)%n''%C(white)%s%C(reset) %C(dim white)- %an%C(reset)' --all**  **lg = !"git lg1"** | |
| **gitk** | The Git repository browser (graph browser). Important options:   * **--all** show all refs (branches, tags, etc.) * **--branches[=pattern]**, **--tags[=pattern]**, **--remotes[pattern]** – pretends as if all the branches (tags, remote branches, etc) are listed on the command line as <commit>. If <pattern> is given, limit refs to ones matching given shell glob. If pattern lacks ?, \*, or [, /\* at the end is implied. |
| **git blame <file>** | Show what revision and author last modified each line of a <file>. Important options are:   * **--show-stat** include additional statistics at the end of blame output * **--reverse** walk history forward instead of backward |
| **Git :: Remote repositories** | |
| **git remote** | Shows all existing remote repositories (links). Some important options are:   * **-v** show URLs for remote aliases * **-vv** ????? |
| **git remote add <alias> <url>** | Adds pointer <alias> to remote repository <url> in list of tracked repositories |
| **git remote rename <old> <new>** | Rename alias <old> to alias <new>. Alias content (URL) doesn't affected. |
| **git remote show <alias>** | Show full info about remote repository <alias>. |
| **git remote rm <alias>** | Remove link (alias) to remote repository <alias>. |
| **git fetch [alias]** | Fetch data from remote repository (default or named by [alias]) and updates all remote branches pointers. **Command doesn't merge fetched data! Don't forget to merge if you need it!** |
| **git pull [alias]** | Pull data from remote repository (default or named by [alias]) and merge with local repository current branch ([git pull] = [git fetch] + [git merge]). **Be careful: command automatically merges fetched remote data!** |
| **git push [alias] [branch name]** | Push committed data from local repository (all branches or concrete [branch name]) to remote repository (default or named by [alias]). |
| **git push -u [alias] [branch name]** |  |
| **git push [alias] <tag>** | Push tag to remote repository (default or named by [alias]). |
| **git push [alias] --tags** | Push all tags to remote repository (default or named by [alias]). |
| **Git :: Branches and Tags** | |
| **git branch <name>** | Create new branch <name> (**just create new pointer, doesn't switch HEAD!**). For switching HEAD use **git checkout <name>** - see below. |
| **git branch -v** | Shows the latest commit in every branch |
| **git branch -d <name>** | Deletes branch <name>. If this branch contains unique (not merged in any other branch) data, git doesn't allow it. Use option -D instead. |
| **git branch --merged** | Shows all branches, that **were** merged with current branch |
| **git branch --no-merged** | Shows all branches, that **were not** merged with current branch |
|  |  |
|  |  |
|  |  |
| **git checkout [-b] <name>** | Switch HEAD to branch <name>. **This command changes content of working folder.** If you specify option **-b** command create branch <name> and switch to it (it's short for [**git branch**] + [**git checkout**]) |
|  |  |
| **git tag** | Show all existing tags |
| **git tag <tag>** | Create **lightweight** tag for last commit in HEAD branch. |
| **git tag -a <tag> [-m <comment>] [revision]** | Create **annotated** tag named <tag>. Optionally (if you specify option **-m**) add message <comment> to tag. If you omit message <comment> but add option **-m** git opens text editor for adding tag message. If you specify **[revision]** (first 6-8 symbols of commit sha-1 sum) tag will be created for specified commit. |
| **git show <tag>** | Show commit data and (optional) tag data for tag <tag>. Only commit data is shown for **lightweight** tags, full data (commit and tag) is shown for **annotated** tags. |
| **git merge** |  |
| **git mergetool** |  |
| **git push <origin> --delete <branchname>** | Remove remote branch. |
|  |  |

## Java

TBD

## JavaScript

TBD

## Node.js

* <https://nodejs.org> – Node.js itself
* <https://www.npmjs.com> – Node Package Manager
* <https://github.com/nvm-sh/nvm> - Node Version Manager for Linux environment (POSIX-compliant bash script to manage multiple active node.js versions)
* <https://github.com/coreybutler/nvm-windows> - NVM for Windows environment

|  |  |
| --- | --- |
| npm init -y | Initialization Node.js packaje in the current directory (package.json will be created). |
| node <js script> | Execution of the Node.js application. |
| npm install [package(s) name(s)] [flag(s)] | Install the latest version of the node package. If specified package name as: <package>@<version> - the specified version will be installed. If no package name specified, packages specified in the **package.json** will be installed. Flags are relevant for all modes: **--global**|**-g** – globally, **--save**|**-S**|<no flags> - as runtime and development time dependency, **--save-dev**|**-D** – development time dependency. Can be specified multiple multiple packages, separated by space. |
| npm list | Show all installed packages. Flags: **-g** – show globally installed packages. |
| npm view <package> versions | Show all available package versions. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Python

TBD

### Virtual Environment

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |