Cheat Sheet: Advanced Linux Commands

RED HAT® DEVELOPER PROGRAM

Lets Get Started...

This cheat sheet should help you get started with developing a (web) application on Red Hat Enterprise Linux (RHEL). We'll assume you have a VM running RHEL, by - for example - having run through the steps in the "Using Vagrant to Get Started with RHEL" blog

As an example scenario, we are going to pretend we are developing a LAMP (Linux, Apache, MariaDB and PHP) application on single machine running Red Hat Enterprise Linux 7. As a first step, we're going to install Apache, PHP and MariaDB (the drop-in replacement for MySQL that's shipped with Red Hat Enterprise Linux 7), and start the appropriate services:

`# yum -y install httpd mariadb-server php-mysql php`	Installs the correct packages to start developing a LAMP application: the Apache webserver, the base packages for PHP, and a MariaDB server, including MySQL bindings for PHP.
`$ systemctl status httpd`	Show information about httpd, including process ID, child processes, time since startup, what man pages are available, the most recent log messages, and more.
`# systemctl start httpd mariadb`	Start the httpd and mariadb services. Instead of 'start', you can also use stop or restart, for obvious use cases.
`# systemctl enable httpd mariadb`	Enable the httpd and mariadb services to start at next boot. You can also use disable, mask or unmask.

So the framework is installed and services should be running; let's check if everything is ok by checking out the logs. (You must either be a member of the 'adm' group on the system, or run these commands with 'sudo' prepended to them to see all log messages.)

`$ journalctl -f -l`	Show and keep open (-f) the system log, allowing you to see new messages scrolling by. The -l flag prevents truncating of long lines.
`$ journalctl -f -l -u httpd -u mariadb`	Same as above, but only for log messages from the httpd and mariadb services.
`$ journalctl -f -l -u httpd -u mariadb --since -300`	Same as above, only for log messages that are less than 300 seconds (5 minutes) old

Now in order to test our app in the VM, we need the IP address of the server. For that we want to see the IP address configured for the first network card, called 'eth0' in most virtual machines:

`$ nmcli d`	Show the status of all network interfaces
`$ nmcli d show eth0`	Show details of network interface eth0; alternatively you can use 'ip a s eth0'
`# nmcli d connect eth0`	Bring up the network interface eth0. You can use 'disconnect' to bring the interface down.

Now let's drop an example PHP file in /var/www/html to see if everything works

`$ cat << EOF > /var/www/html/test.php`
`<?php`
`  phpinfo();`
`?>`
`EOF`
	All text between the first line and EOF will be added to /var/www/html/test.php. Any existing content in that file will be overwritten. This is called a 'heredoc'.

Now we can download the test.php file from either the same machine, or our local workstation:

`$ curl http://www.someapp.org/test.php` `$ curl http://10.0.0.10/test.php`	Use the 'curl' command to perform a download of the test.php file at www.someapp.org or 10.0.0.10, respectively
`$ curl http://localhost:80/someapp/api -v`	Fetch sent and received HTTP GET status, API response payload from the local host
`$ curl https://localhost:443/someapp/api -v -F "arg1=foo" -F "arg2=bar"`	Fetch sent and received HTTPS POST status, API response payload from the local host
`$ host www.someapp.org`	Use the 'host' command to test DNS name resolution; you might need to run 'yum -y install bind-utils' for this command to work.

Generally, files in `/var/www/html` are owned by apache.In a dev environment, you might want to make those files owned by apache and a developer group. Here are some commands that are useful to make that a reality.

`# chown apache:developers test.php`	Change ownership of test.php to "apache" and the "developers" group. (You can only change ownership of a file to another user if you are the superuser, "root".)
`# chmod u+rw,g+rw,o+r test.php`	Change the mode of test.php to allow owner (u) and users in the group (g) to read and write (+rw) it, and the rest of the world (o) to just read (+r) it.
`# chmod g+rw test.php`	Allow users in the group of test.php to read and write it

`# chown -R :developers /var/www/html`	Change ownership of /var/www/html and all files in that directory to the developers group.
`# chmod g+s /var/www/html`	Special command to make sure that all files created in /var/www/html are owned by the group that own /var/www/html; it sets to so-called sticky bit.

Maybe you have a script that you want to use on that server, too. You'll need to make it executable first:

`$ chmod 755 somescript`	Allow the owner of somescript to read, write and execute it, and the rest of the world to just read and execute it.
`$ chmod +x somefile`	Allow execution of somefile

Red Hat Enterprise Linux 7 ships with a security feature called SELinux. SELinux basically labels all files, and then whitelists what labels a program (e.g. Apache) is allowed to read.

`$ ls -lZ test.php`	Show the SELinux label of test.php. Files in /var/www/html need to be labeled httpd_sys_content_t (content readable by Apache) or httpd_sys_rw_content_t (content readable and writable by Apache).
`# ausearch -sv no --comm httpd`	Search the audit log for recently denied events triggered by Apache ('httpd'). Useful for debugging an application that might be running into SELinux related problems.
`# restorecon -FvR /var/www/html`	Use this command to restore the default labels on all files under /var/www/html if different from those mentioned above.
`$ getenforce`	Show what mode SELinux is in: Disabled, Permissive or Enforcing. Switch SELinux to enforcing mode with 'setenforce 1'.
`# semanage fcontext -l | grep '/var/www'`	View all SELinux rules that potentially apply to /var/www in the extensive SELinux docs. Install the policycoreutils-python package with yum to get the 'semanage' command.

If you have a database on a separate server, you need to allow Apache to initiate network connections, which SELinux denies by default. This is done by setting an SELinux boolean.

`$ getsebool -a`	Show all available SELinux boolean settings
`# setsebool httpd_can_network_connect_db 1`	Tell SELinux to allow httpd to make connections to databases on other servers. Use the -P flag to make permanent.

The above should hopefully get you started with developing on RHEL, but you can do so much more! For example, here are some commands to run a program in the background in your shell.

`$ ./someprogram &`	Start someprogram in the background. You can also just start someprogram and hit CTRL-Z to suspend it and send it to the background.
`$ jobs`	Show all background jobs in current shell; add -l for more information on the jobs.
`$ bg [number]`	Continue suspended job (i.e. a job suspended with CTRL-Z) in the background.
`$ fg [number]`	Bring a background job to the foreground again.

And if you need to get an idea on how your application or system is performing, you might like these commands

`$ free`	Show the amount of free memory. Please note it's not necessarily a problem if Linux seems to use a lot of memory!
`$ vmstat 3`	Every three seconds, show statistics about the system, like utilization, memory in use, etc.
`$ iotop`	Show 'top' like output for disk i/o. Must be root to run this. First install the iotop package with yum.
`$ ps xauww`	Show the system process list

Finally, maybe you want to use Java instead of PHP. These two commands install some programs you might want to use in that case

`# subscription-manager repos --enable rhel-server-rhscl-7-rpms`	Enable the Software Collections repositories to install packages from (required for Maven)
`# yum -y install java-1.8.0-openjdk-devel tomcat maven30 git`	Single command to install your Java compiler, Tomcat webserver, maven and git.

About the Author

Maxim Burgerhout is a solution architect in the Red Hat Benelux team. He is often spotted talking about systems management and infrastructure, including infrastructure automation, implementing self-service deployments and configuration management.

In the past, he's been involved in various migrations from legacy Unix to Red Hat Enterprise Linux. Those migrations always involved making developers feel at home on the new platform by providing the right tools and documentation and getting them up to speed quickly.

Maxim has done some development in Ruby, PHP and Python in the past and is currently learning Java, because, well, just because.

@MaximBurgerhout Linkedin

Opensource.com: Linux Common Commands Cheat Sheet

BY SETH KENLON

Use this handy Linux command cheat sheet for executing common tasks such as navigating files, installing software, and starting services.

NAVIGATE FILES

LIST DIRECTORIES (WITH TYPE INDICATOR)

```
$ ls --file-type
```

CHANGE DIRECTORY TO ``EXAMPLE``

```
$ cd example
```

MOVE UP ONE DIRECTORY

```
$ cd ..
```

MOVE UP TWO DIRECTORIES

```
$ cd ../..
```

CHANGE TO HOME DIRECTORY

```
$ cd ~
```

GET CURRENT DIRECTORY

```
$ pwd
```

GET ABSOLUTE PATH TO A FILE OR FOLDER

```
$ readlink -f example
```

GET FILE TYPE OF ``EXAMPLE.EXT``

```
$ file example.ext
```

INSTALLING SOFTWARE

- On Fedora and CentOS, `[COMMAND]` is `dnf`
- On Ubuntu and Debian, `[COMMAND]` is `apt`
- On OpenSUSE, `[COMMAND]` is `zypper`
- Other distributions may use different commands

SEARCH FOR AN APPLICATION CALLED EXAMPLE

```
$ sudo [COMMAND] search example
```

INSTALL AN APPLICATION CALLED EXAMPLE

```
$ sudo [COMMAND] install example
```

UNINSTALL AN APPLICATION CALLED EXAMPLE

```
$ sudo [COMMAND] remove example
```

SERVICES

START SERVICES

```
$ sudo systemctl start example
```

STOP SERVICES

```
$ sudo systemctl stop example
```

GET STATUS OF SERVICES

```
$ sudo systemctl status example
```

FILE MANAGEMENT

COPY A FILE IN PLACE

```
$ cp example.txt example-1.txt
```

COPY A FILE TO DOCUMENTS

```
$ cp example.txt ~/Documents/example-1.txt
```

MOVE A FILE TO DOCUMENTS

```
$ mv example.txt ~/Documents
```

CREATE A DIRECTORY (FOLDER)

```
$ mkdir example
```

REMOVE AN EMPTY DIRECTORY

```
$ rmdir example
```

SAFELY REMOVE A FILE

```
$ trash example.txt
```

REMOVE A FILE (WITHOUT TRASH COMMAND)

```
$ mv example.txt ~/.local/share/Trash/files
```

PERMANENTLY DELETE A FILE

```
$ shred example.txt
```

DOWNLOAD A FILE FROM AN NETWORK LOCATION

```
$ wget http://example.com/file
```


A sysadmin's daily task involve managing servers and the data center's network. Following utilities and commands would help a sysadmin manage networks using linux from basic to advanced level.

Ping	
As the name suggests, ping is used to check the end-to-end connectivity between the system that you are pinging it to from your system. It uses ICMP Echo packets that travel back when a ping is successful. This might be a very first step to check any system/network connectivity. Ping can be with IPv4 and IPv6 addresses both. To know more about IP addresses and how to get your system's IP, refer to the article: https://opensource.com/article/18/5/how-find-ip-address-linux	
IPv4- ping <ip address>/<fqdn>	fqdn stands for fully qualified domain name, this can be your website-name.com or your server like server-name.company.com
IPv6- ping6 <ip address>/<fqdn>	
Also, you can use it to resolve names of websites to their corresponding IP address.	
Traceroute	
This is a nice utility for tracing the full network path from your system to other. Ping check's the end-to-end connectivity, traceroute utility tell you all the router IPs which come in the path when you try to reach the end system/website/server. Usually it is the second step after ping for any network connection debugging.	
traceroute <ip address>/<fqdn>	
Telnet	
Use this to telnet to any server.	
telnet <ip address>/<fqdn>	
Netstat	
Network statistics (netstat) utility is used to troubleshoot network connection problems with ability to check interface/port statistics, routing tables, protocol stats, etc. Any sysadmin's must-have tool!	
netstat -l	shows the list of all the ports which are in listening mode
netstat -a	shows all ports, to specify only tcp use '-at' (for udp use '-au')
netstat -r	provides routing table
netstat -s	provides summary of statistics for each protocol
netstat -i	displays TX/RX packet statistics for each interface
Nmcli	
A very good utility for managing network connections,configurations,etc. It can be used to control Network Manager and modify network configuration details of any device.	
nmcli device	lists all devices on the system
nmcli device show <interface>	shows network related details of the specified interface
nmcli connection	to check connection of the device
nmcli connection down <interface>/nmcli connection up <interface>	this command shuts/starts the specified interface
nmcli con add type vlan con-name <connection-name> dev <interface> id <vlan-number> ipv4 <ip/cidr> gw4 <gateway-ip>	this commmand adds a vlan interface with the specified vlan number, ip and a gateway to a particular interface
Routing	
There are many commands to check and configure routing. Some useful ones and their short description is as shown below:	
ip route	shows all the current routes configured for respective interfaces.
route add default gw <gateway-ip>	to add a default gateway to the routing table
route add -net <network ip/cidr> gw <gateway ip> <interface>	to add a new network route to the routing table. There are many other routing parameters like adding a default route, default gateway, etc.
route del -net <network ip/cidr>	to delete a particular route entry from the routing table.
ip neighbor	this shows the current neighbor table. It can be used to add/change/delete new neighbors.
arp	this is another similar utility like ip neighbor. It maps IP address of a system to its corresponding MAC (Media Access Control) address. In networking, ARP stands for Address Resolution Protocol.

Tcpdump

Linux provides many packet capturing tools like tcpdump, wireshark, tshark etc. They are used to capture the network traffic in packets which are transmitted/received and hence very useful for a sysadmin to debug any packet loss or related issues. For CLI enthusiasts, tcpdump is a great tool and for GUI users Wireshark is a great utility to capture and analyze packets. Tcpdump is a linux built-in utility to capture network traffic. It can be used to capture/show traffic on specific ports, protocols, etc.

tcpdump -i <interface-name>	shows live packets from the specified interface. Packets can be saved in a file by the adding '-w' flag & name of the output file to the command ex- tcpdump -w <output-file.> -i <interface-name>
tcpdump -i <interface> src <source-ip>	to capture packets from a particular source IP
tcpdump -i <interface> dst <destination-ip>	to capture packets from a particular destination IP
tcpdump -i <interface> port <port-number>	to capture traffic for a specific port number like 53, 80, 8080, etc.
tcpdump -i <interface> <protocol>	to capture traffic for a particular protocol like tcp, udp, etc.

Iptables

this is a firewall-like packet filtering utility which can allow/block certain traffic. The scope of this utility is very wide so we will discuss few of the useful ones.

iptables -L	lists all existing iptables rules
iptables -F	delete all the existing rules

The below commands allow traffic from the specified port number to the specified interface:

iptables -A INPUT -i <interface> -p tcp –dport <port-number> -m state –state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -o <interface> -p tcp -sport <port-number> -m state – state ESTABLISHED -j ACCEPT

To allow loopback access to the system:

iptables -A INPUT -i lo -j ACCEPT

iptables -A OUTPUT -o lo -j ACCEPT

Nslookup

this tool is used to obtain IP address mapping of a website/domain and vice versa. It can also be used to obtain information on your DNS server, all DNS records of the website, shown in one of the examples below. Similar tool to nslookup is dig (Domain Information Groper) utility.

nslookup <website-name.com>	this command shows the IP address of your DNS server in Server field and below that it gives the IP address of the website you are trying to reach
nslookup -type=any <website-name.com>	shows all the available records for the specified website/domain

Network/Interface Debugging

To troubleshoot interface connectivity or related network issues, here is a quick summary of the necessary commands/files.

netstat	utility for network statistics
ss	utility for dumping socket statistics
nmap <ip-address>	to scan network ports, discover hosts, MAC address detection, much more. Stands for Network Mapper.
ip addr/ifconfig -a	command to provide IP addresses and related info of all the interfaces of a system
ssh -vvv user@<ip/domain>	used to ssh to another server with the specified ip/domain and username. The '-vvv' flag provides "triple-verbose" details of the processes going on while ssh'ing to the server
ethtool -S <interface>	to check the statistics for a particular interface
ifup <interface>/ifdown <interface>	to start/shut the specified interface
systemctl restart network	to restart network service for the system
/etc/sysconfig/network-scripts/<interface-name>	interface configuration file used to set IP, network, gateway, etc. for the specified interface. DHCP mode can be set here.
/etc/hosts	this file contains custom host/domain to IP mappings
/etc/resolv.conf	used to specify DNS nameserver IP of the system
/etc/ntp.conf	used to specify NTP server domain

Learn more in our sysadmin's guide to SELinux, by Alex Callejas: https://red.ht/2zpWppY

CONCEPTS

SELinux = LABELING system

Every process, file, directory, system object has a LABEL.

Policy rules control access between labeled processes and labeled objects.

The kernel enforces these rules.

Labeling → files, process, ports, etc. (system objects)

Type enforcement → Isolates processes from each other based on types

LABELING

Label format:

user:role:type:level (optional)

user → identity known to the policy authorized for a specific set of roles and a specific MLS/MCS range

role → attribute of RBAC, serves as an intermediary between domains and SELinux users

type → attribute of type enforcement, defines a domain for processes and a type for files

level → attribute of MLS/MCS, pair of levels, written as lowlevel-highlevel if the levels differ, or lowlevel if the levels are identical

TYPE ENFORCEMENT

Targeted:	Processes that are targeted run in a confined domain, and processes that are not targeted run in an unconfined domain
Multi-level security (mls):	Control processes (domains) based on the level of the data they will be using
Multi-category security (mcs):	Protects like processes from each other (like VMs, OpenShift Gears, SELinux sandboxes, containers, etc.)

SELINUX MODES @ BOOT

Kernel parameters:

enforcing=0 → boot in permissive mode

selinux=0 → kernel to not load any part of the SELinux infrastructure

autorelabel=1 → forces the system to relabel

If you need to relabel the entire system:

 # touch /.autorelabel

 # reboot

If the system labeling contains a large amount of errors, you might need to boot in permissive mode for the autorelabel to succeed.

SELINUX STATES

enforcing	SELinux security policy is enforced
permissive	SELinux prints warnings instead of enforcing
disabled	No SELinux policy is loaded

CHECK STATUS:

Configuration file:	Check if SELinux is enabled:	**# getenforce**
/etc/selinux/config	SELinux status tool:	**# sestatus**
	Enable/disable SELinux (temporarily):	**# setenforce [1|0]**

EXAMPLE OF LABELING: APACHE WEB SERVER		
Binary	**/usr/sbin/httpd**	**httpd_exec_t**
Configuration directory	**/etc/httpd**	**httpd_config_t**
Logfile directory	**/var/log/httpd**	**httpd_log_t**
Content directory	**/var/www/html**	**httpd_sys_content_t**
Startup script	**/usr/lib/systemd/system/httpd.service**	**httpd_unit_file_d**
Process running	**/usr/sbin/httpd -DFOREGROUND**	**httpd_t**
Ports (netstat -tulpnZ)	80/tcp, 443/tcp	**httpd_t**
Port type (semanage port -l)	80, 81, 443, 488, 8008, 8009, 8443, 9000	**http_port_t**

CHECK/CREATE/MODIFY SELINUX CONTEXTS/LABELS:

Many commands accept the argument -Z to view, create, and modify context:

- ls -Z
- id -Z
- ps -Z
- netstat -Z
- cp -Z
- mkdir -Z

Contexts are set when files are created based on their parent directory's context (with a few exceptions). RPMs can set contexts as part of installation.

TROUBLESHOOTING			
SELinux tools:	**# yum -y install setroubleshoot setroubleshoot-server**		← Reboot or restart auditd after you install
Logging:	**/var/log/messages**	**/var/log/audit/audit.log**	**/var/lib/setroubleshoot/setroubleshoot_database.xml**
journalctl	List all logs related to setroubleshoot:	**# journalctl -t setroubleshoot --since=14:20**	
	List all logs related to a particular SELinux label:	**# journalctl _SELINUX_CONTEXT=system_u:system_r:policykit_t:s0**	
ausearch	Look for SELinux errors in the audit log:	**# ausearch -m AVC,USER_AVC,SELINUX_ERR -ts today -i**	
	Search for SELinux AVC messages for a particular service:	**# ausearch -m avc -c httpd -i**	
Edit/modify labels (semanage)	know the label:	**# semanage fcontext -a -t httpd_sys_content_t '/srv/myweb(/.*)?'**	
	know the file with the equivalent labeling:	**# semanage fcontext -a -e /srv/myweb /var/www**	
	Restore the context (for both cases):	**# restorecon -vR /srv/myweb**	
Edit/modify labels (chcon)	know the label:	**# chcon -t httpd_system_content_t /var/www/html/index.html**	Note: If you move instead of copy a file, the file keeps its original context.
	know the file with the equivalent labeling:	**# chcon --reference /var/www/html/ /var/www/html/index.html**	
	Restore the context (for both cases):	**# restorecon -vR /var/www/html/index.html**	
Add new port to service:	**# semanage port -a -t http_port_t -p tcp 8585**		← SELinux needs to know
Booleans	Booleans allow parts of SELinux policy to be changed at runtime without any knowledge of SELinux policy writing.		
To see all booleans:	**# getsebool -a**	To see the description of each one:	**# semanage boolean -l**
To set a boolean execute:	**# setsebool [boolean] [1|0]**	To configure it permanently, add -P:	Example : # setseebol httpd_enable_ftp_server 1 -P

Git Cheat Sheet

RED HAT® DEVELOPER PROGRAM

Table of contents

The basics

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Installation

Install git client wherever you need to run.

Windows	https://windows.github.com
Mac	https://mac.github.com
Fedora	$yum install git (up to Fedora 21) $dnf install git (Fedora 22 and newer)
Debian/Ubuntu	$apt-get install git
Git for all platforms	http://git-scm.com

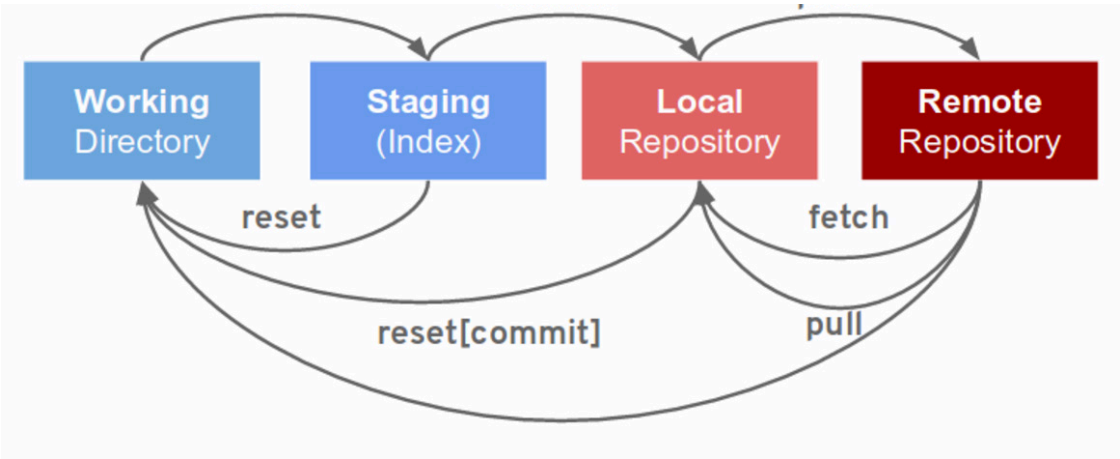
Next, configure user information used across all local repositories.

```
# set a name that is identifiable for credit when reviewing version history
$ git config --global user.name "[firstname lastname]"

# set an email address that will be associated with each history marker
$ git config --global user.email "[valid-email]"

# set automatic command line coloring for Git for easy reviewing
$ git config --global color.ui auto
```

Git commands sequence



Git CLI

Create

```
# Clone an existing repository
$ git clone https://github.com/bparees/openshift-jee-sample.git

# Create a new local repository
$ git init
```

Local changes

```
# Changed files in your working directory
$ git status

# Changes to tracked files
$ git diff

# Add all current changes to the next commit
$ git add .

# Add some changes in <file> to the next commit
$ git add -p <file>

# Commit all local changes in tracked files
$ git commit -a

# Commit previously staged changes
$ git commit
```

Commit history

```
# Show all commits, starting with newest
$ git log

# Show changes over time for a specific file
$ git log -p <file>

# Who changed what and when in <file>
$ git blame <file>
```

Branches & tags

```
# List all existing branches
$ git branch -av

# Switch HEAD branch
$ git checkout <branch>

# Create a new branch based on your current HEAD
$ git branch <new-branch>

# Delete a local branch
$ git branch -d <branch>

# Mark the current commit with a tag
$ git tag <tag-name>
```

Update & publish

```
# List all currently configured remotes
$ git remote -v

# Add new remote repository, named <remote>
$ git remote add <shortname> <url>

# Download all changes from <remote>, but don‘t integrate into HEAD
$ git fetch <remote>

# Download changes and directly merge/integrate into HEAD
$ git pull <remote> <branch>

# Publish local changes on a remote
$ git push <remote> <branch>
```

Merge & rebase

```
# Merge <branch> into your current HEAD
$ git merge <branch>

# Rebase your current HEAD onto <branch>
$ git rebase <branch>

# Abort a rebase
$ git rebase --abort

# Use your configured merge tool to solve conflicts
$ git mergetool

# Use your editor to manually solve conflicts and (  after resolving) mark
file as resolved
$ git add <resolved-file>
$ git rm <resolved-file>
```

Undo

```
# Discard all local changes in your working directory
$ git reset --hard HEAD

# Discard local changes in a specific file
$ git checkout HEAD <file>

# Revert a commit (by producing a new commit with contrary changes)
$ git revert <commit>

# Reset your HEAD pointer to a previous commit and discard all changes
since then
$ git reset --hard <commit>

# Reset your HEAD pointer to a previous commit and preserve all changes as
unstaged changes
$ git reset <commit>
```

Quickstart using Git

1. Start a new app project or work on an existing app project

```
# The “git init” command in a specific folder of your new project creates a
new and empty Git repository
$ git init

# The “git clone” command is used to download an existing repository from a
remote server.
$ git clone <remote-url>
```

2. Do your stuff on files

```
# Update your codes or add new files or delete some of them via  your
favorite editor
$ vi hellowWorld.java
…
Change your codes or contents
…
Save and Quit <wq>

# (Optional) Rebase your current HEAD onto <branch>
$ git rebase <branch>
```

3. Keep looking over your work

```
# The “git status” command tells you what happened since the last commit
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
    modified:   ansible/vars.yml
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    ansible/helloworld-msa.retry
no changes added to commit (use "git add" and/or "git commit -a")
```

4. Add files to your staging

```
# The “git add” command enables you to add all changed files to your
staging area
$ git add .

# If you need to add a specific file to your staging, you have to describe
the file name explicitly
$ git add <filename>
```

5. Commit all staged changes

```
# A commit wraps up all the changes you previously staged with the “git
  add” command. To record this set of changes in Git’s database, you
  execute the “git commit” command with a short and informative message
$ git commit -m "message"
```

6. Confirm your work

```
# Running the “git status” command right after a commit proves to you: only
the changes that you added to the Staging Area were committed
$ git status
```

7. Inspect the commit history

```
# The “git log” command lists all the commits that were saved in
chronological order. This allows you to see which changes were made in
detail and helps you comprehend how the project evolved
$ git log
commit 6a8110589c25eac8acd7223d2bf91995c0b72db8
Author: Daniel Oh <doh@redhat.com>
Date:   Fri Mar 24 13:27:03 2017 -0400
  Update Zipkin to 0.1.9
commit 40380a55163a7e93366c01e37d1d0ad5a3afc848
Author: Daniel Oh <doh@redhat.com>
Date:   Thu Mar 23 21:44:09 2017 -0400
  Remove the need to adjust SCC
# Publish local changes on a remote
$ git push
```

About the Author

Daniel Oh is an AppDev Specialist Solution Architect, Agile & DevOps CoP Manager at Red Hat and has specialty about JBoss, Java EE, Containers, Agile methodology, DevOps, PaaS(OpenShift), Containerized application design, MSA, and Mobile application platform.

@danieloh30
https://www.linkedin.com/in/daniel-oh-083818112

Linux Commands Cheat Sheet

Easy to use Linux shortcuts
for developers.

RED HAT DEVELOPERS

ssh [ip or hostname] **"vagrant ssh" in the same directory as the Vagrantfile to shell into the box/machine (assumes you have successfully "vagrant up")**	Secure shell, an encrypted network protocol allowing for remote login and command execution On Windows: PuTTY and WinSCP An "ssh.exe" is also available via Cygwin as well as with a Git installation.
pwd	Print Working Directory Displays the full path name
whoami	Displays your logged in user id
cd / **cd target** **cd ~**	Change directory to the root of the filesystem Change directory to "target" directory Change directory to your home directory
ls **ls -l** **ls -la**	Directory listing Long listing, displays file ownership Displays hidden files/directories

```
[vagrant@rhel-cdk /]$ ls
bin  boot  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  pro
[vagrant@rhel-cdk /]$ ls -l
total 62
lrwxrwxrwx.   1 root    root        7 Mar  8 20:36 bin -> usr/bin
dr-xr-xr-x.   4 root    root     1024 Mar 12 19:26 boot
drwxr-xr-x.  18 root    root     3100 Mar 12 19:49 dev
drwxr-xr-x.  85 root    root     4096 Mar 12 19:31 etc
drwxr-xr-x.   3 root    root     4096 Mar  8 20:54 home
lrwxrwxrwx.   1 root    root        7 Mar  8 20:36 lib -> usr/lib
lrwxrwxrwx.   1 root    root        9 Mar  8 20:36 lib64 -> usr/lib64
drwx------.   2 root    root    16384 Mar  8 20:34 lost+found
drwxr-xr-x.   2 root    root     4096 May 25  2015 media
drwxr-xr-x.   2 root    root     4096 May 25  2015 mnt
drwxr-xr-x.   3 root    root     4096 Mar  8 20:50 opt
dr-xr-xr-x. 166 root    root        0 Mar 12 19:26 proc
dr-xr-x---.   3 root    root     4096 Mar 12 19:30 root
drwxr-xr-x.  31 root    root     1000 Mar 12 19:31 run
lrwxrwxrwx.   1 root    root        8 Mar  8 20:36 sbin -> usr/sbin
drwxr-xr-x.   2 root    root     4096 May 25  2015 srv
dr-xr-xr-x.  13 root    root        0 Mar 12 19:26 sys
drwxrwxrwt.   7 root    root     4096 Mar 12 20:31 tmp
drwxr-xr-x.  13 root    root     4096 Mar  8 20:36 usr
drwxr-xr-x.   3 vagrant vagrant  4096 Mar 12 19:25 vagrant
drwxr-xr-x.  19 root    root     4096 Mar 12 19:26 var
[vagrant@rhel-cdk /]$
```

clear	Clear the terminal screen

cat file.txt	Displays the contents of file.txt to standard out
cat /etc/system-release	Displays the contents of the system-release file - what version of RHEL, Centos or Fedora are you running?

```
[vagrant@rhel-cdk etc]$ cat /etc/system-release
Red Hat Enterprise Linux Server release 7.2 (Maipo)
[vagrant@rhel-cdk etc]$
```

cat longfile.txt | more	Displays the contents of the file with forward paging

```
Count is 0
Count is 1
Count is 2
Count is 3
Count is 4
Count is 5
Count is 6
Count is 7
Count is 8
Count is 9
Count is 10
Count is 11
Count is 12
Count is 13
Count is 14
Count is 15
Count is 16
Count is 17
Count is 18
Count is 19
Count is 20
Count is 21
Count is 22
Count is 23
Count is 24
Count is 25
Count is 26
Count is 27
--More--
```

less longfile.txt	Scroll forward: Ctrl-f Scroll backward: Ctrl-b End of file: G Quit less: q
man cat	Man pages, the user manual. In this case, it will describe the cat command

```
CAT(1)

NAME
       cat - concatenate files and print on the standard output

SYNOPSIS
       cat [OPTION]... [FILE]...

DESCRIPTION
       Concatenate FILE(s), or standard input, to standard output.

       -A, --show-all
              equivalent to -vET

       -b, --number-nonblank
              number nonempty output lines, overrides -n

       -e     equivalent to -vE

       -E, --show-ends
              display $ at end of each line

       -n, --number
              number all output lines

       -s, --squeeze-blank
              suppress repeated empty output lines

       -t     equivalent to -vT

       -T, --show-tabs
              display TAB characters as ^I
 Manual page cat(1) line 1 (press h for help or q to quit)
```

cp source_file.js target_file.js **cp -r ~/source_dir ~/target_dir**	Copies a specific file Copies all files and sub-dirs

mkdir my_directory	Create the directory "my_directory"

```
[vagrant@rhel-cdk ~]$ mkdir my_directory
[vagrant@rhel-cdk ~]$ ls -la
total 32
drwx------. 5 vagrant vagrant 4096 Mar 12 21:39 .
drwxr-xr-x. 3 root    root    4096 Mar  8 20:54 ..
-rw-r--r--. 1 vagrant vagrant   18 Jul  8  2015 .bash_logout
-rw-r--r--. 1 vagrant vagrant  193 Jul  8  2015 .bash_profile
-rw-r--r--. 1 vagrant vagrant  231 Jul  8  2015 .bashrc
drwxr-xr-x. 2 vagrant vagrant 4096 Mar 12 19:26 .docker
drwxrwxr-x. 2 vagrant vagrant 4096 Mar 12 21:39 my_directory
drwx------. 2 vagrant vagrant 4096 Mar 12 19:26 .ssh
[vagrant@rhel-cdk ~]$
```

rm myfile.js **rm -rf my_directory/**	Removes a specific file Removes a directory, recursively

mv [source_file] [target_file]	Move file or directory

ps -ef	Displays information about a selection of the active processes

```
[vagrant@rhel-cdk etc]$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 19:26 ?        00:00:02 /usr/lib/systemd/systemd
root         2     0  0 19:26 ?        00:00:00 [kthreadd]
root         3     2  0 19:26 ?        00:00:01 [ksoftirqd/0]
root         5     2  0 19:26 ?        00:00:05 [kworker/0:0H]
root         7     2  0 19:26 ?        00:00:00 [migration/0]
root         8     2  0 19:26 ?        00:00:00 [rcu_bh]
root         9     2  0 19:26 ?        00:00:00 [rcuob/0]
root        10     2  0 19:26 ?        00:00:00 [rcuob/1]
root        11     2  0 19:26 ?        00:00:19 [rcu_sched]
root        12     2  0 19:26 ?        00:00:14 [rcuos/0]
root        13     2  0 19:26 ?        00:00:16 [rcuos/1]
root        14     2  0 19:26 ?        00:00:00 [watchdog/0]
root        15     2  0 19:26 ?        00:00:00 [watchdog/1]
root        16     2  0 19:26 ?        00:00:00 [migration/1]
root        17     2  0 19:26 ?        00:00:00 [ksoftirqd/1]
root        19     2  0 19:26 ?        00:00:00 [kworker/1:0H]
root        20     2  0 19:26 ?        00:00:00 [khelper]
root        21     2  0 19:26 ?        00:00:00 [kdevtmpfs]
root        22     2  0 19:26 ?        00:00:00 [netns]
root        23     2  0 19:26 ?        00:00:00 [perf]
root        24     2  0 19:26 ?        00:00:00 [writeback]
root        25     2  0 19:26 ?        00:00:00 [kintegrityd]
root        26     2  0 19:26 ?        00:00:00 [bioset]
root        27     2  0 19:26 ?        00:00:00 [kblockd]
root        28     2  0 19:26 ?        00:00:00 [md]
root        29     2  0 19:26 ?        00:00:02 [kworker/0:1]
root        34     2  0 19:26 ?        00:00:00 [khungtaskd]
root        35     2  0 19:26 ?        00:00:01 [kswapd0]
root        36     2  0 19:26 ?        00:00:00 [ksmd]
```

./runthisthing

Execute a program or shell script in your current working directory (pwd)
Executable items are have an "x" in their long listing (ls -la)

```
[vagrant@rhel-cdk bin]$ pwd
/usr/bin
[vagrant@rhel-cdk bin]$ ls -la | grep cp
-rwxr-xr-x.   1 root root      155136 Nov 25 10:55 cp
-rwxr-xr-x.   1 root root      141632 Jul  8  2015 cpio
-rwxr-xr-x.   1 root root      768592 Jul 15  2015 cpp
-rwxr-xr-x.   1 root root       67928 Jan 23 05:07 cpupower
-rwxr-xr-x.   1 root root       58352 Aug 21  2015 lscpu
-rwxr-xr-x.   1 root root       11400 Sep 15 05:51 rpm2cpio
-rwxr-xr-x.   1 root root       70360 Jan 13 10:22 scp
[vagrant@rhel-cdk bin]$
```

```
[vagrant@rhel-cdk ~]$ cd ~
[vagrant@rhel-cdk ~]$ ls -la
total 32
drwx------. 4 vagrant vagrant 4096 Mar 12 21:04 .
drwxr-xr-x. 3 root    root    4096 Mar  8 20:54 ..
-rw-r--r--. 1 vagrant vagrant   18 Jul  8  2015 .bash_logout
-rw-r--r--. 1 vagrant vagrant  193 Jul  8  2015 .bash_profile
-rw-r--r--. 1 vagrant vagrant  231 Jul  8  2015 .bashrc
drwxr-xr-x. 2 vagrant vagrant 4096 Mar 12 19:26 .docker
-rwxrwxr-x. 1 vagrant vagrant  107 Mar 12 21:04 runthisthing
drwx------. 2 vagrant vagrant 4096 Mar 12 19:26 .ssh
[vagrant@rhel-cdk ~]$
```

./runthisthing &

Execute a program or shell script as a background task

ps -ef | grep runthisthing

Find a particular process by name. The "|" is a pipe, redirects the output of the left-side command to the standard input of the right-side command

```
[vagrant@rhel-cdk ~]$ ./runthisthing &
[3] 22047
[vagrant@rhel-cdk ~]$ ps -ef | grep runthisthing
vagrant  22047 14771 99 21:11 pts/0    00:00:05 /bin/bash ./runthisthing
vagrant  22060 14771  0 21:11 pts/0    00:00:00 grep --color=auto runthisthing
[vagrant@rhel-cdk ~]$
```

kill -9 [pid]

```
[vagrant@rhel-cdk ~]$ ps -ef | grep runthisthing
vagrant  22047 14771 99 21:11 pts/0    00:00:21 /bin/bash ./runthisthing
vagrant  22081 14771  0 21:11 pts/0    00:00:00 grep --color=auto runthisthing
[vagrant@rhel-cdk ~]$ kill -9 22047
[vagrant@rhel-cdk ~]$ ps -ef | grep runthisthing
vagrant  22096 14771  0 21:11 pts/0    00:00:00 grep --color=auto runthisthing
[3]   Killed                  ./runthisthing
[vagrant@rhel-cdk ~]$
```

ip -4 a

Shows the IPv4 address for all NICs

top What is eating your CPU

```
top - 21:17:06 up  1:51,  1 user,  load average: 0.41, 0.67, 0.49
Tasks: 157 total,   4 running, 151 sleeping,   2 stopped,   0 zombie
%Cpu(s): 47.4 us,  3.7 sy,  0.0 ni, 48.5 id,  0.2 wa,  0.0 hi,  0.2 si,  0.0 st
KiB Mem :  1017056 total,   108588 free,   378700 used,   529768 buff/cache
KiB Swap:  1572860 total,  1559232 free,    13628 used.   436384 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
22526 vagrant   20   0  113116   1184   1008 R 100.0  0.1   0:08.06 runthisthing
13368 root      20   0 1448088 233012  15992 S  10.0 22.9  12:36.47 openshift
13168 root      20   0 1377296  44816  11004 S   1.3  4.4   1:10.37 docker
  453 root      20   0   43588   9560   6892 S   0.3  0.9   0:10.46 systemd-journal
    1 root      20   0   43896   5532   3568 S   0.0  0.5   0:02.06 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.03 kthreadd
    3 root      20   0       0      0      0 R   0.0  0.0   0:01.45 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:06.25 kworker/0:0H
    7 root      rt   0       0      0      0 S   0.0  0.0   0:00.66 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/1
   11 root      20   0       0      0      0 S   0.0  0.0   0:24.92 rcu_sched
   12 root      20   0       0      0      0 R   0.0  0.0   0:18.26 rcuos/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:21.08 rcuos/1
   14 root      rt   0       0      0      0 S   0.0  0.0   0:00.19 watchdog/0
   15 root      rt   0       0      0      0 S   0.0  0.0   0:00.20 watchdog/1
   16 root      rt   0       0      0      0 S   0.0  0.0   0:00.72 migration/1
   17 root      20   0       0      0      0 S   0.0  0.0   0:01.08 ksoftirqd/1
   19 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   20 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 khelper
   21 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kdevtmpfs
   22 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   23 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 perf
   24 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
   25 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
```

which [executable] Where is the executable located

```
[vagrant@rhel-cdk ~]$ which docker
/usr/bin/docker
[vagrant@rhel-cdk ~]$ which oc
/usr/bin/oc
[vagrant@rhel-cdk ~]$ which top
/usr/bin/top
[vagrant@rhel-cdk ~]$
```

echo "Stuff" > target_file.txt **echo "more" >> target_file.txt**	single > redirects the output to the file "target_file.txt" A double >> appends

```
[vagrant@rhel-cdk ~]$ cd ~
[vagrant@rhel-cdk ~]$ ls -la
total 32
drwx------. 4 vagrant vagrant 4096 Mar 12 21:29 .
drwxr-xr-x. 3 root    root    4096 Mar  8 20:54 ..
-rw-r--r--. 1 vagrant vagrant   18 Jul  8  2015 .bash_logout
-rw-r--r--. 1 vagrant vagrant  193 Jul  8  2015 .bash_profile
-rw-r--r--. 1 vagrant vagrant  231 Jul  8  2015 .bashrc
drwxr-xr-x. 2 vagrant vagrant 4096 Mar 12 19:26 .docker
-rwxrwxr-x. 1 vagrant vagrant  114 Mar 12 21:11 runthisthing
drwx------. 2 vagrant vagrant 4096 Mar 12 19:26 .ssh
[vagrant@rhel-cdk ~]$ echo "stuff" > target_file.txt
[vagrant@rhel-cdk ~]$ cat target_file.txt
stuff
[vagrant@rhel-cdk ~]$ echo "more" >> target_file.txt
[vagrant@rhel-cdk ~]$ cat target_file.txt
stuff
more
[vagrant@rhel-cdk ~]$ ls -la
total 36
drwx------. 4 vagrant vagrant 4096 Mar 12 21:29 .
drwxr-xr-x. 3 root    root    4096 Mar  8 20:54 ..
-rw-r--r--. 1 vagrant vagrant   18 Jul  8  2015 .bash_logout
-rw-r--r--. 1 vagrant vagrant  193 Jul  8  2015 .bash_profile
-rw-r--r--. 1 vagrant vagrant  231 Jul  8  2015 .bashrc
drwxr-xr-x. 2 vagrant vagrant 4096 Mar 12 19:26 .docker
-rwxrwxr-x. 1 vagrant vagrant  114 Mar 12 21:11 runthisthing
drwx------. 2 vagrant vagrant 4096 Mar 12 19:26 .ssh
-rw-rw-r--. 1 vagrant vagrant   11 Mar 12 21:29 target_file.txt
[vagrant@rhel-cdk ~]$
```

echo $PATH	Displays the $PATH environment variable

```
[vagrant@rhel-cdk ~]$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/vagrant/.local/bin:/home/vagrant/bin
```

env	Displays all ENV variables
export PATH=$PATH:/anoth-erdir	Adds "anotherdir" to your PATH, just for your current session
sudo find . -name [file]	Find a file or directory by name

```
[vagrant@rhel-cdk /]$ cd /
[vagrant@rhel-cdk /]$ sudo find . -name vagrant
./var/spool/mail/vagrant
./vagrant
./vagrant/src/booker/vagrant
./etc/sudoers.d/vagrant
./home/vagrant
[vagrant@rhel-cdk /]$
```

grep -i stuff `find . -name \*.txt -print`	Find the string "stuff" in all the .txt files

```
[vagrant@rhel-cdk /]$ cd ~
[vagrant@rhel-cdk ~]$ echo "Stuff" > target_file.txt
[vagrant@rhel-cdk ~]$ echo "more" >> target_file.txt
[vagrant@rhel-cdk ~]$ ls
my_directory  target_file.txt
[vagrant@rhel-cdk ~]$ grep -i stuff `find . -name \*.txt -print`
"Stuff"
[vagrant@rhel-cdk ~]$
```

head [file]	Output the first part of file (first 10 lines)
curl developers.redhat.com	Retrieve the content from developers.redhat.com
source myenvsetting_script.sh	How to add something to the PATH and make it stick By default a new shell is launched to run a script, therefore env changes are not visible to your current shell.

```
[vagrant@rhel-cdk ~]$ ls
myenvsetting_script.sh  mystuff
[vagrant@rhel-cdk ~]$ cat myenvsetting_script.sh
export MY_STUFF=/home/vagrant/mystuff
export PATH=$PATH:$MY_STUFF/bin
[vagrant@rhel-cdk ~]$ source myenvsetting_script.sh
[vagrant@rhel-cdk ~]$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/vagrant/.local/bin
:/home/vagrant/bin:/home/vagrant/mystuff/bin
[vagrant@rhel-cdk ~]$
```

Note: the path uses ":" as a separator vs ";" in the Windows world

sudo yum -y install net-tools	"yum" is the installation tool for Fedora, Centos and RHEL. This command installs "net-tools" which has many handy utilities like netstat

```
[vagrant@rhel-cdk ~]$ sudo yum -y install net-tools
Loaded plugins: product-id, search-disabled-repos, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package net-tools.x86_64 0:2.0-0.17.20131004git.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package    Arch    Version                     Repository              Size
================================================================================
Installing:
 net-tools x86_64 2.0-0.17.20131004git.el7   rhel-7-server-eus-rpms 304 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 304 k
Installed size: 917 k
Downloading packages:
net-tools-2.0-0.17.20131004git.el7.x86_64.rpm        | 304 kB   00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : net-tools-2.0-0.17.20131004git.el7.x86_64              1/1
  Verifying  : net-tools-2.0-0.17.20131004git.el7.x86_64              1/1

Installed:
  net-tools.x86_64 0:2.0-0.17.20131004git.el7

Complete!
[vagrant@rhel-cdk ~]$
```

sudo netstat -anp | grep tcp | grep LISTEN

Lists the various in-use ports and the process using it

```
[vagrant@rhel-cdk ~]$ sudo netstat -anp | grep tcp | grep LISTEN
tcp        0      0 127.0.0.1:10443         0.0.0.0:*               LISTEN      27967/haproxy
tcp        0      0 127.0.0.1:10444         0.0.0.0:*               LISTEN      27967/haproxy
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      27967/haproxy
tcp        0      0 0.0.0.0:1936            0.0.0.0:*               LISTEN      27967/haproxy
tcp        0      0 0.0.0.0:53              0.0.0.0:*               LISTEN      13368/openshift
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      907/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1651/master
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      27967/haproxy
tcp        0      0 0.0.0.0:8443            0.0.0.0:*               LISTEN      13368/openshift
tcp6       0      0 :::2376                 :::*                    LISTEN      13168/docker
tcp6       0      0 :::53866                :::*                    LISTEN      13368/openshift
tcp6       0      0 :::10250                :::*                    LISTEN      13368/openshift
tcp6       0      0 :::33900                :::*                    LISTEN      13368/openshift
tcp6       0      0 :::9101                 :::*                    LISTEN      14636/haproxy_expor
tcp6       0      0 :::52944                :::*                    LISTEN      13368/openshift
tcp6       0      0 :::38549                :::*                    LISTEN      13368/openshift
tcp6       0      0 :::22                   :::*                    LISTEN      907/sshd
tcp6       0      0 :::7001                 :::*                    LISTEN      13368/openshift
tcp6       0      0 ::1:25                  :::*                    LISTEN      1651/master
tcp6       0      0 :::4001                 :::*                    LISTEN      13368/openshift
tcp6       0      0 :::32869                :::*                    LISTEN      13368/openshift
[vagrant@rhel-cdk ~]$
```

sudo netstat -anp | grep 2376

Lists the process listening on port 2376

```
[vagrant@rhel-cdk ~]$ sudo netstat -anp | grep 2376
tcp6       0      0 :::2376                 :::*                    LISTEN      13168/docker
[vagrant@rhel-cdk ~]$
```

This is particularly useful when another process is hanging out on a port you need, like if you started Apache on 80 or Tomcat on 8080.

wget https://someurl.com/somefile.tar.gz

wget is a useful utility for downloading files from any website. If installation is required, simply sudo yum -y install wget

```
[vagrant@rhel-cdk ~]$ sudo yum -y install wget
Loaded plugins: product-id, search-disabled-repos, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package wget.x86_64 0:1.14-10.el7_0.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package          Arch              Version                  Repository      Size
================================================================================
Installing:
 wget             x86_64            1.14-10.el7_0.1          rhel-7-server-eus-rpms
                                                                              546 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 546 k
Installed size: 2.0 M
Downloading packages:
wget-1.14-10.el7_0.1.x86_64.rpm                                | 546 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : wget-1.14-10.el7_0.1.x86_64                                  1/1
  Verifying  : wget-1.14-10.el7_0.1.x86_64                                  1/1

Installed:
  wget.x86_64 0:1.14-10.el7_0.1

Complete!
[vagrant@rhel-cdk ~]$
```

tar -xf somefile.tar.gz
tar -xf somefile.tar.gz -C ~/somedir

Extracts/expands (think unzip) into current directory
Expands into the "somedir" directory

RHEL

Red Hat Enterprise Linux 8

We are delighted to introduce you to Red Hat Enterprise Linux 8. If you're familiar with previous versions of Red Hat Enterprise Linux, you'll find RHEL 8 more intuitive to pick up and use. However, there are a few new features and changes that you'll want to be aware of, so we hope this cheat sheet will help you quickly explore and begin your RHEL 8 application development.

SIMPLIFIED SOFTWARE PACKAGING AND INSTALLATION

Installing and using RHEL 8 is much easier than previous releases. Previously, there were server, workstation, and desktop variants, but RHEL 8 uses one installation medium for all variants. The RHEL 8 has also been simplified with fewer repos - they are:

`BaseOS` - primarily core operating system packages with support for the lifetime of the OS

`Appstream` - user-space applications and components, including numerous Application Streams (see below)

`CodeReady Builder`- additional libraries and tools for developers

`Supplementary` - 3rd party support only

Compilers, runtimes, web/database servers, and development tools will generally be delivered as Application Streams from the AppStream repo. See below for more info.

WORKNG WITH CONTAINERS

To enable container management without the need for daemons, Red Hat has introduced a set of tools for your Linux container application development:

Buildah allows you to build a container without any daemon or docker.

Podman allows you to manage containers without the daemon dependency it's also docker cli compatible.

`# podman pull`

RHEL 8 compatible images can be found here

`# yum install -y podman`

`# alias docker=podman`

type to use podman in place of docker

RED HAT UNIVERSAL BASE IMAGE (UBI)

Derived from Red Hat Enterprise Linux, the Red Hat Universal Base Image (UBI) provides a freely redistributable, enterprise-grade base container image on which developers can build and deliver their applications. This means you can containerize your app in UBI and deploy it anywhere. Of course, it will be more secure and Red Hat supported when deployed on RHEL or Red Hat OpenShift, but now you have more options. There are separate UBI 7 and UBI 8 versions for RHEL 7 and 8, respectively. You can obtain a number of RHEL container images from the Red Hat container catalog.

BASIC RED HAT ENTERPRISE LINUX COMMANDS

The most basic tasks that you might need after the operating system has been installed include:

`# yum search string`

search for packages matching a specific string

`# yum install package_name`

install a package

`# yum update package_name`

update a package

`# yum remove package_name`

`# yum history undo last`

uninstall a package and any packages that depend on it

`$ yum list all`

list information on all installed and available packages

`$ yum list installed`

list all installed packages

`# subscription-manager repos --list`

list all available repositories

`$ yum repolist`

list all currently enabled repositories

`# subscription-manager repos --enable repository`

enable a repository

`# subscription-manager repos --disable repository`

disable a repository

INTRODUCING APPLICATION STREAMS

RHEL 8 Beta introduces *Application Streams* where we deliver user space packages (e.g. compilers, scripting languages, databases, etc.) on a cadence that makes sense for each package.

In RHEL 8, Applications Streams are mostly packaged as Modules, but a few are non-module RPMs. A module is a set of RPM packages that can or must be installed together. A typical module can contain packages with an application, packages with the application's specific dependency libraries, packages with documentation for the application, and packages with helper utilities. Modules can have one or more streams - different versions of the module.

Terms and terminology:

***Application Stream* (or simply *stream*)** - refers to content. PHP 7.2 is an application stream. PHP 7.3 is an application stream.

Module - is the packaging format. PHP is packaged as a module.

Module Stream - different versions of a component packaged as a module. PHP 7.2 is an application stream packaged in a module stream.

appstream - is the name of the RHEL 8 repo where you can find Application Streams.

For even more information about Application Streams and modules, see Introducing Application Streams in RHEL 8.

FINDING AND EXPLORING MODULES

The following are common module commands.

`$ yum module list`

list all modules

`$ yum module list installed`

list installed modules

`$ yum module provides package`

find which module provides a package

`$ yum module info module`

examine details of a module

`$ yum module info --profile module:stream`

list packages installed by profiles of a module

`$ yum module list module`

display the current status of a module

WORKING WITH MODULES

The following commands must run with administrator privileges. Note also that some operations with modules require changes to many packages.

`# yum module enable module:stream`

enable a specific stream without installing packages

`# yum module install module:stream/profile`

install a specific stream

`# yum module remove module && yum module disable module`

disable a module stream and remove all packages provided by it

INSTALLING SPECIFIC APPLICATION STREAMS

The following table lists the most interesting Application Streams available in RHEL 8.

.NET Core 2.1	`$ sudo yum install dotnet`
Ant 1.1	`$ sudo yum install ant`
Buildah 1.5 & Podman 1.0	`$ sudo yum install buildah podman`
Clang/LLVM 7.0	`$ sudo yum install llvm-toolset`
GCC 8.2 plus complementarytools	`$ sudo yum group install "Development Tools"`
GO 1.11	`$ sudo yum install go-toolset`
HTTPD 2.4	`$ sudo yum install httpd`
MariaDB 10.3	`$ sudo yum install mariadb`
Maven 3.5	`$ sudo yum install maven`
MySQL 8	`$ sudo yum install mysql`
Nginx 1.14	`$ sudo yum install nginx`
Node.js 10	`$ sudo yum install nodejs`
OpenJDK 11	`$ sudo yum install java-11-openjdk-devel`
OpenJDK 8	`$ sudo yum install java-1.8.0-openjdk-devel`
PCP 4.3	`$ sudo yum install pcp-zeroconf`
Perl 5.26 & 5.24	`$ sudo yum install perl`
PHP 7.2	`$ sudo yum install php`
PostgreSQL 10.5	`$ sudo yum install postgresql`
PostgreSQL 9.6	`$ sudo yum module install postgresql:9.6`
Python 2.7	`$ sudo yum install python2 or yum module install python27`
Python 3.6	`$ sudo yum install python3 or yum module install python36`

INSTALLING SPECIFIC APPLICATION STREAMS (cont)

Redis 5	`$ sudo yum install redis`
Ruby 2.5	`$ sudo yum install ruby`
Rust 1.31	`$ sudo yum install rust-toolset`
Scala 2.10	`$ sudo yum install scala`
Subversion 1.1	`$ sudo yum install subversion`
Swig 3	`$ sudo yum install swig`
Systemtap 4.0	`$ sudo yum install systemtap`
Valgrind 3.14	`$ sudo yum install valgrind`
Varnish 6	`$ sudo yum install varnish`

MORE INFORMATION

For more information about RHEL 8, visit the Red Hat Developer website.

Note: if `sudo` isn't enabled for your user ID, see How to enable sudo on Red Hat Enterprise Linux. During system installation, checking the box Make this user administrator enables `sudo` for your user ID.

Authors: Mike Guerette, Vladimir Slavik